



**ARM Cortex™ -M0**

**32-BIT 微控制器**

**NuMicro M051™ 系列  
技术参考手册**

## 目录

<b>1</b>	<b>概述</b> .....	<b>11</b>
<b>2</b>	<b>特征</b> .....	<b>12</b>
<b>3</b>	<b>框图</b> .....	<b>16</b>
<b>4</b>	<b>选型表</b> .....	<b>17</b>
<b>5</b>	<b>管脚配置</b> .....	<b>18</b>
5.1	QFN 33 pin .....	18
5.2	LQFP 48 pin .....	19
5.3	管脚描述 .....	20
<b>6</b>	<b>功能描述</b> .....	<b>23</b>
6.1	ARM® Cortex™ -M0内核 .....	23
6.2	系统管理器 .....	25
6.2.1	概述 .....	25
6.2.2	系统复位 .....	25
6.2.3	系统电源分配 .....	25
6.2.4	系统存储器映射 .....	27
6.2.5	系统存储器映射表 .....	29
6.2.6	系统管理器控制寄存器 .....	30
6.2.7	系统定时器(SysTick) .....	60
6.2.8	嵌套向量中断控制器 (NVIC) .....	65
6.2.9	系统控制器寄存器映射 .....	89
6.3	时钟控制器 .....	96
6.3.1	概述 .....	96
6.3.2	时钟发生器框图 .....	96
6.3.3	系统时钟 & SysTick 时钟 .....	97
6.3.4	AHB 时钟源选择 .....	98
6.3.5	外围设备时钟源选择 .....	99
6.3.6	掉电模式(深度休眠模式) 时钟 .....	100
6.3.7	分频器输出 .....	101
6.3.8	时钟控制寄存器映射 .....	102
6.3.9	时钟控制寄存器描述 .....	103
6.4	通用I/O .....	124
6.4.1	概述 .....	124
6.4.2	Port 0-4 控制器寄存器映射 .....	126
6.4.3	Port 0-4 控制器寄存器描述 .....	130
6.5	I2C 总线控制器 (主机/从机) .....	144
6.5.1	简介 .....	144
6.5.2	特征 .....	144
6.5.3	功能描述 .....	145

6.5.4	I2C 协议寄存器	149
6.5.5	I2C 控制器寄存器映射	153
6.5.6	I2C 控制器寄存器描述	154
6.5.7	操作模式	162
6.5.8	5种操作模式下的数据传输流	163
6.6	PWM发生器和捕捉定时器	169
6.6.1	简介	169
6.6.2	特征	170
6.6.3	PWM 框图	171
6.6.4	PWM 功能描述	175
6.6.5	PWM 控制器寄存器映射	182
6.6.6	PWM 控制器寄存器描述	185
6.7	串行外围设备接口(SPI)控制器	210
6.7.1	简介	210
6.7.2	特性	210
6.7.3	SPI 框图	211
6.7.4	SPI 功能描述	212
6.7.5	SPI 时序波形图	218
6.7.6	SPI编程例程	221
6.7.7	SPI串行总线控制寄存器映射	223
6.7.8	SPI控制寄存器描述	224
6.8	定时器控制器	233
6.8.1	简介	233
6.8.2	特征	233
6.8.3	定时器控制器框图	234
6.8.4	定时器操作模式	235
6.8.5	定时器控制器寄存器映射	236
6.9	看门狗定时器 (WDT)	242
6.9.1	简介	242
6.9.2	特征	244
6.9.3	WDT 框图	244
6.9.4	看门狗定时器控制寄存器映射	245
6.10	UART接口控制器	248
6.10.1	简介	248
6.10.2	特性	250
6.10.3	UART 框图	251
6.10.4	IrDA 模式	254
6.10.5	RS-485 模式	256
6.10.6	UART 接口控制寄存器映射	258
6.10.7	UART接口控制寄存器描述	260
6.11	模拟数字转换(ADC)	283
6.11.1	简介	283
6.11.2	特征	283

6.11.3	ADC框图	284
6.11.4	ADC操作步骤	286
6.11.5	ADC 寄存器映射	291
6.11.6	ADC 寄存器描述	292
6.12	外部总线接口 (EBI)	304
6.12.1	简介	304
6.12.2	特性	304
6.12.3	EBI 框图	305
6.12.4	操作步骤	306
6.12.5	EBI 控制器寄存器映射	312
6.12.6	EBI 控制器寄存器描述	312
6.13	Flash内存控制器(FMC)	315
6.13.1	简介	315
6.13.2	特性	315
6.13.3	FMC 框图	316
6.13.4	FMC内存结构	317
6.13.5	启动选择	319
6.13.6	Data Flash	320
6.13.7	在系统编程(ISP)	321
6.13.8	FMC控制寄存器映射	325
6.13.9	FMC控制器寄存器描述	326
<b>7</b>	<b>USER 配置</b>	<b>336</b>
<b>8</b>	<b>典型应用电路</b>	<b>338</b>
<b>9</b>	<b>电气特性</b>	<b>339</b>
9.1	绝对最大额定值	339
9.2	DC电气特性	340
9.3	AC 电气特性	344
9.3.1	外部晶振	344
9.3.2	外部振荡器	344
9.3.3	晶振的典型应用电路	345
9.3.4	内部 22.1184 MHz RC振荡器	346
9.3.5	内部 10kHz RC 振荡器	346
9.4	模拟量特性	347
9.4.1	600kHz sps 12-bit SARADC特性	347
9.4.2	LDO规格 & Power 管理	348
9.4.3	低压复位说明	349
9.4.4	欠压检测说明	349
9.4.5	上电复位说明(5V)	349
9.5	SPI 动态特性	350
<b>10</b>	<b>封装尺寸</b>	<b>352</b>



10.1 LQFP-48 (7x7x1.4mm <sup>2</sup> Footprint 2.0mm) .....	352
10.2 QFN-33 (5X5 mm <sup>2</sup> , Thickness 0.8mm, Pitch 0.5 mm).....	353
<b>11 版本历史</b> .....	<b>354</b>

## LIST OF FIGURES

图 5.1-1 NuMicro™ M051 系列框图 .....	16
图 5.1-1 NuMicro™ 命名规则 .....	17
图 5.1-1 NuMicro™ M051 系列QFN33 引脚图 .....	18
图 5.2-1 NuMicro™ M051 系列 LQFP-48引脚图 .....	19
图 6.1-1 功能框图 .....	23
图 6.2-1 NuMicro M051™ 系列电源分配图 .....	26
图 6.3-1 时钟发生器框图 .....	96
图 6.3-2 系统时钟框图 .....	97
图 6.3-3 SysTick时钟控制框图 .....	97
图 6.3-4 AHB 时钟源HCLK .....	98
图 6.3-5 外设时钟源选择PCLK .....	99
图 6.3-6分频器的时钟源 .....	101
图 6.3-7分频器框图 .....	101
图 6.4-1 推挽输出 .....	124
图 6.4-2 开漏输出 .....	125
图 6.4-3 准双端I/O 模式 .....	125
图 6.5-1 I2C 总线时序 .....	144
图 6.5-2 I2C 协议 .....	145
图 6.5-3 主机向从机传输数据 .....	146
图 6.5-4 主机读取从机的数据 .....	146
图 6.5-5 START 和 STOP 条件 .....	147
图 6.5-6 I2C总线上的位传输 .....	148
图 6.5-7 I2C总线上的应答信号 .....	148
图 6.5-8 I2C 数据移位 .....	150
图 6.5-9: I2C 超时计数器框图 .....	152
图 6.5-10 Legend for the following four figures .....	163
图 6.5-11 主机发送模式 .....	164
图 6.5-12主机接收模式 .....	165
图 6.5-13从机发送模式 .....	166
图 6.5-14从机接收模式 .....	167

图 6.5-15 广播呼叫模式.....	168
图 6.6-1 PWM发生器 0 时钟源控制.....	171
图 6.6-2 PWM 发生器0结构框图 .....	171
图 6.6-3 PWM 发生器 2 时钟源控制.....	172
图 6.6-4 PWM 发生器 2结构框图 .....	172
图 6.6-5 PWM 发生器 4 时钟源控制.....	173
图 6.6-6 PWM 发生器 4 结构框图 .....	173
图 6.6-7 PWM 发生器 6 时钟源控制.....	174
图 6.6-8 PWM 发生器 6 结构框图 .....	174
图 6.6-9 PWM定时器内部比较器输出.....	175
图 6.6-10 PWM定时器操作时序 .....	176
图 6.6-11 PWM 双缓存图解.....	176
图 6.6-12 PWM 控制器输出占空比.....	177
图 6.6-13死区发生器操作 .....	177
图 6.6-14 捕捉操作时序.....	178
图 6.6-16 PWM A组 PWM-定时器中断结构图.....	179
图 6.6-17 PWM B组 PWM-定时器中断结构图.....	179
图 6.7-1 SPI 框图.....	211
图 6.7-2 SPI主机模式应用框图.....	212
图 6.7-3 SPI从机模式应用框图.....	212
图 6.7-4 一次传输两个Transactions (Burst Mode) .....	214
图 6.7-5 字节重排列.....	215
图 6.7-6 字节休眠时序波形.....	216
图 6.7-7 可调串行时钟频率.....	217
图 6.7-8 主机模式下SPI 时序 .....	218
图 6.7-9 主机模式下SPI 时序(Alternate Phase of SPICLK) .....	219
图 6.7-10 从机模式下SPI 时序 .....	219
图 6.7-11 从机模式下SPI 时序(Alternate Phase of SPICLK) .....	220
图 6.8-1定时器控制器框图.....	234
图 6.8-2定时器控制的时钟源 .....	234
图 6.9-1 中断时序与复位信号时序.....	243

图 6.9-2 看门狗定时器时钟控制 .....	244
图 6.9-3 看门狗定时框图 .....	244
图 6.10-1 UART 时钟控制框图 .....	251
图 6.10-2 UART 框图 .....	252
图 6.10-3 自动流控制框图 .....	253
图 6.10-4 IrDA 框图 .....	254
图 6.10-5 IrDA TX/RX 时序框图 .....	255
图 6.10-6 RS-485 帧结构 .....	257
图 6.11-1 ADC 控制器框图 .....	284
图 6.11-2 ADC 时钟控制 .....	285
图 6.11-3 ADC 转换器自校准时序框图 .....	286
图 6.11-4 单次转换模式时序图 .....	287
图 6.11-5 单周期扫描下使能通道转换时序图 .....	288
图 6.11-6 使能通道的连续扫描时序图 .....	289
图 6.11-7 A/D 转换结果监控框图 .....	290
图 6.11-8 A/D 控制器中断 .....	290
图 6.11-9 ADC 单端输入转换电压和转换结果图 .....	293
图 6.11-10 ADC 差分输入转换电压和转换结果图 .....	294
图 6.12-1 EBI 框图 .....	305
图 6.12-2 16位EBI数据宽度与16位器件连接 .....	306
图 6.12-3 8位EBI数据宽度与8位设备连接 .....	307
图 6.12-4 16位数据宽度的时序控制波形 .....	309
图 6.12-5 8位数据宽度时序控制波形 .....	310
图 6.12-6 插入空闲周期的时序控制波形 .....	311
图 6.13-1 Flash内存控制框图 .....	316
图 6.13-2 Flash内存组成 .....	318
图 6.13-3 上电时启动选择(BS) .....	319
图 6.13-4 Flash内存结构 .....	320
图 6.13-5 ISP 时钟源控制 .....	321
图 6.13-6 ISPGo 时序 .....	322
图 6.13-7 ISP 软件编程流程 .....	323



图 9.3-1 典型晶振应用电路.....	345
图 9.5-1 SPI 主机时序 .....	351
图 9.5-2 SPI 从机时序 .....	351

## LIST OF TABLES

表 5.3-1 NuMicro™ M051 系列引脚描述 .....	22
表 6.2-1 片上模块的地址空间分配 .....	27
表 6.2-2 片上模块的地址空间分配 .....	28
表 6.2-3 异常模式 .....	67
表 6.2-4 系统中断映射 .....	67
表 6.2-5 向量表格式 .....	68
表 6.3-1 掉电模式控制表 .....	105
表 6.9-1 看门狗定时溢出间隔选择 .....	242
表 6.10-1 UART 波特率方程 .....	248
表 6.10-2 UART 波特率设置表 .....	249
表 6.10-3 软件模式下 UART 中断源和标志表 ( Software Mode) .....	275
表 6.10-4 波特率方程表 .....	278
表 6.13-1 Flash 存储器地址映射 .....	317
表 6.13-2 ISP Mode .....	324

## 1 概述

NuMicro M051™ 系列是以ARM® Cortex™-M0为内核的32位微控制器，应用于工业控制和需要丰富通信接口的领域。Cortex™-M0是ARM最新的32位嵌入式处理器，成本仅相当于传统的8位微控制器。NuMicro M051™ 系列包括M052, M054, M058 和 M0516.

NuMicro M051™ 内核运行频率最高可达50MHz，适用于需要高性能CPU的工业领域。NuMicro M051™ 系列内嵌有 8K/16K/32K/64K-字节的flash存储器，4K字节数据flash存储器，用于在系统编程（ISP）的4K字节 flash存储器，及 4K字节SRAM存储器。

许多系统级外设模块，如I/O端口、EBI (外部总线接口)、Timer、UART、SPI、I2C、PWM、ADC，看门狗定时器和欠压检测功能，NuMicro M051™ 系列集成这些功能模块可以减少系统外围元器件数量，节省电路板空间和系统成本。这些功能使NuMicro M051™ 系列适用于广泛应用。

同时，NuMicro M051™ 系列带有ISP（在系统编程）和ICP（在电路编程）功能，允许用户直接在电路板上对程序存储器进行升级。

## 2 特征

- 内核
  - ARM® Cortex™ -M0内核运行频率可达50MHz.
  - 一个 24位系统定时器。
  - 支持低功耗睡眠模式.
  - 单指令32位硬件乘法器.
  - 嵌套向量中断控制器NVIC可管理32个中断，支持4个优先级。
  - 支持串行调试（SWD）接口，2 个观察点/4 断点。
- 内建一组 LDO支持宽电压工作范围：2.5V~5.5V
- 存储器
  - 8KB/16KB/32KB/64KB Flash用于存储程序代码(APROM)
  - 4KB Flash用于存储数据(DataFlash)
  - 4KB Flash用于存储ISP引导代码 (LDRROM)
  - 4KB字节SRAM用作内部高速暂存存储器
- 时钟控制
  - 可编程的系统时钟源
  - 4~24 MHz外部晶振输入
  - 22.1184MHz内部振荡器（精度可达1%）
  - 低功耗10KHz 的振荡器用于看门狗及睡眠模式唤醒
  - PLL支持CPU最高运行在50MHz
- I/O 端口
  - 在LQFP48管脚封装中，最多支持40个标准I/O端口（GPIO）
  - 4种I/O工作模式：
    - ◆ 准双向模式
    - ◆ 推挽输出模式
    - ◆ 开漏输出模式
    - ◆ 高阻抗输入模式
  - 可选施密特触发输入
  - I/O管脚可被配置为边沿/电平触发模式的中断源
  - 较强的拉电流驱动能力和灌电流承受能力
- 定时器



- 4组32位定时器，均带有24位向上计数定时器和8位预分频计数器
- 每个定时器有独立的时钟源
- 24位定时器值可由定时器数据寄存器（TDR）读出
- 提供3种工作模式：单脉冲模式，周期模式，开关模式。
- 看门狗定时器
  - 多路时钟源
  - 支持在掉电模式和休眠模式下的唤醒功能
  - 看门狗定时溢出的中断/复位选择
- PWM
  - 内建4个16位PWM发生器，可输出8路PWM或4对互补PWM
  - 每个PWM发生器配有一个时钟源选择器，一个时钟分频器，一个8位时钟预分频器，和一个用于互补PWM的死区发生器
  - PWM中断与PWM周期同步
  - 16位捕捉定时器(共享PWM定时器)提供输入的上升/下降沿的捕捉功能
  - 支持捕捉中断
- UART
  - 最多两组UART设备
  - 可编程波特率发生器
  - 带缓冲的接收器和发送器，均带有15bytes的FIFO
  - 可选择支持流程控制(CTS 和 RTS)
  - 支持 IrDA(SIR) 功能
  - 支持RS485功能
- SPI
  - 最高支持2组SPI设备
  - 支持 SPI主机/从机模式
  - 主机模式时钟频率高达20 MHz/ 从机模式时钟频率高达10MHz
  - 全双工同步串行数据传输
  - 数据长度（从1到32位）可改变
  - 可设置MSB或LSB优先的传输模式
  - Rx可在串行时钟的上升/下降沿接收数据
  - Tx可在串行时钟的上升/下降沿发送数据
  - 32位传输模式下的字节睡眠模式

- I2C
  - 支持主机/从机模式
  - 主从机之间双向数据传输
  - 多主机总线支持（无中心主机）
  - 多主机间同时传输数据仲裁，避免总线上串行数据损坏
  - 串行时钟同步使得不同比特率的器件可以通过一条串行总线传输数据
  - 可用同步时钟控制总线上数据暂停及恢复传送。
  - 可编程配置的时钟可适应多样化的传输速率控制。
  - I2C总线上支持多地址识别 (4组从机地址带屏蔽选项)

- ADC
  - 12位逐次逼近式模数转换器ADC，转换速率达 600k SPS
  - 8通道单端输入或4通道差分输入
  - 单次转换模式/突发模式/单周期扫描模式/连续扫描模式
  - 每通道有独立的存放转换结果的寄存器
  - 支持转换值监测（或比较）的阈值电压检测
  - 转换开始可由软件或外部触发
- EBI (外部总线接口) 支持
  - 可访问的空间: 8位模式下为64KB，16位模式下为128KB
  - 支持8bit/16bit 数据宽度
- 在系统编程（ISP）与在电路编程（ICP）
- 欠压检测
  - 支持四级检测电压: 4.5V/3.8V/2.7V/2.2V
  - 支持欠压中断和复位选择
- LVR（低电压复位）
  - 阈值电压: 2.0V
- 工作温度: -40℃~85℃
- 封装:
  - 无铅封装 (RoHS)
  - 48-pin LQFP, 33-pin QFN

## 3 框图

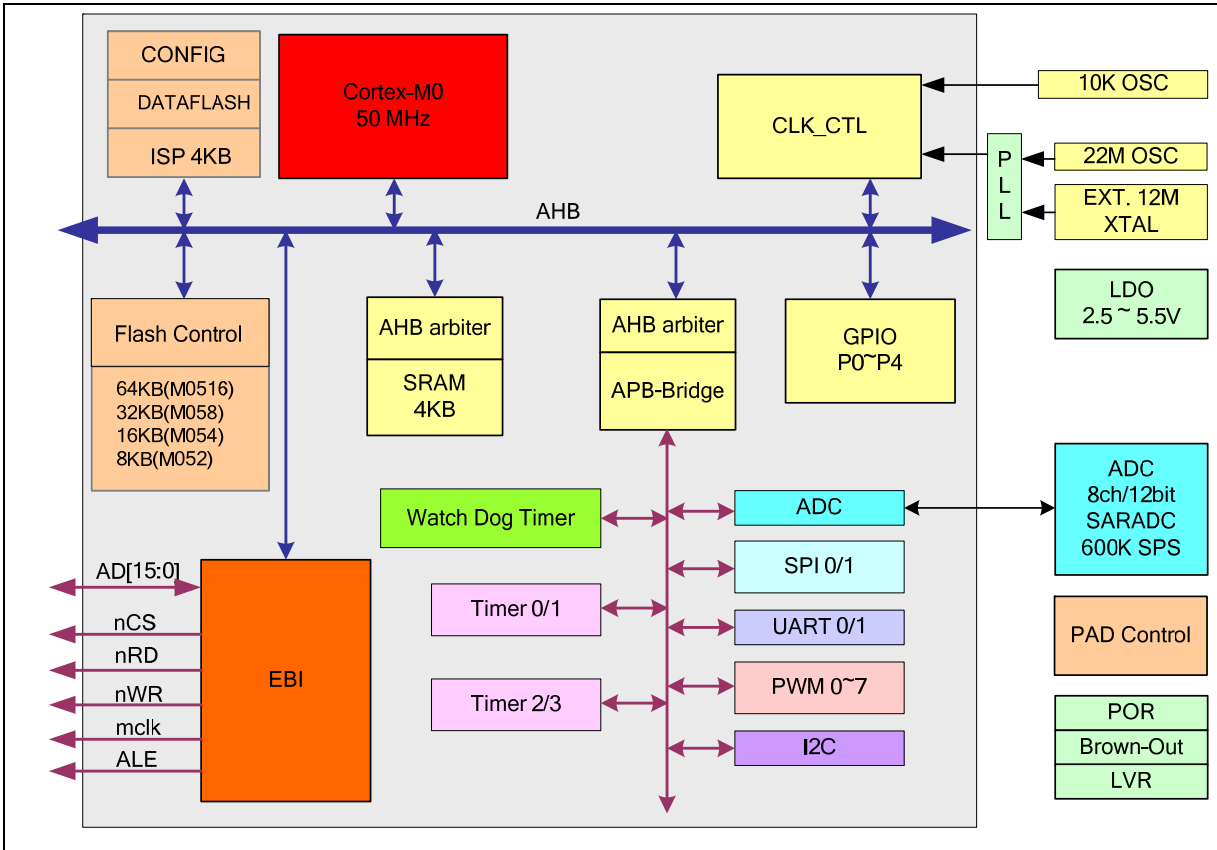


图 5.1-1 NuMicro™ M051 系列框图



## 4 选型表

NuMicro M051™ 系列产品选型指南

型号	APROM	RAM	Data Flash	LDROM	I/O	Timer	通讯接口			PWM	ADC	EBI	ISP ICP	封装
							UART	SPI	I2C					
M052LAN	8KB	4KB	4KB	4KB	40	4x32-bit	2	2	1	8	8x12-bit	v	v	LQFP48
M052ZAN	8KB	4KB	4KB	4KB	24	4x32-bit	2	1	1	5	5x12-bit		v	QFN 33
M054LAN	16KB	4KB	4KB	4KB	40	4x32-bit	2	2	1	8	8x12-bit	v	v	LQFP48
M054ZAN	16KB	4KB	4KB	4KB	24	4x32-bit	2	1	1	5	5x12-bit		v	QFN 33
M058LAN	32KB	4KB	4KB	4KB	40	4x32-bit	2	2	1	8	8x12-bit	v	v	LQFP48
M058ZAN	32KB	4KB	4KB	4KB	24	4x32-bit	2	1	1	5	5x12-bit		v	QFN 33
M0516LAN	64KB	4KB	4KB	4KB	40	4x32-bit	2	2	1	8	8x12-bit	v	v	LQFP48
M0516ZAN	64KB	4KB	4KB	4KB	24	4x32-bit	2	1	1	5	5x12-bit		v	QFN 33

表- NuMicro™ M051 系列产品选型指南

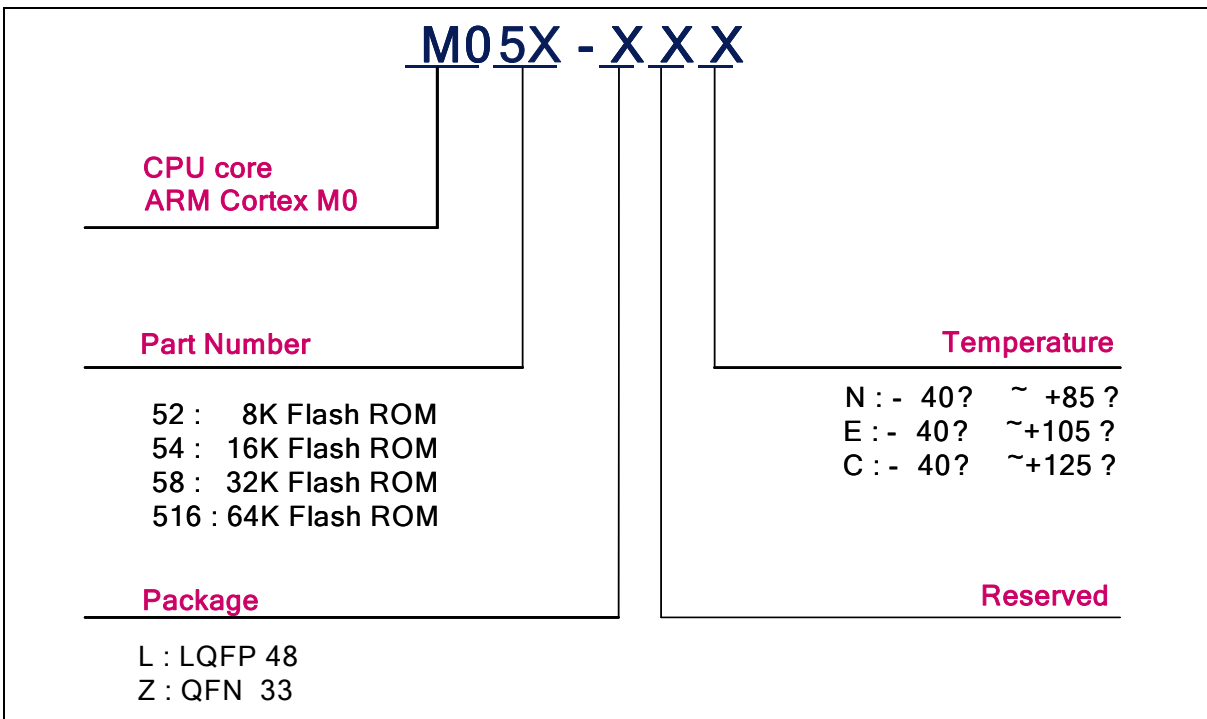


图 5.1-1 NuMicro™ 命名规则

## 5 管脚配置

### 5.1 QFN 33 pin

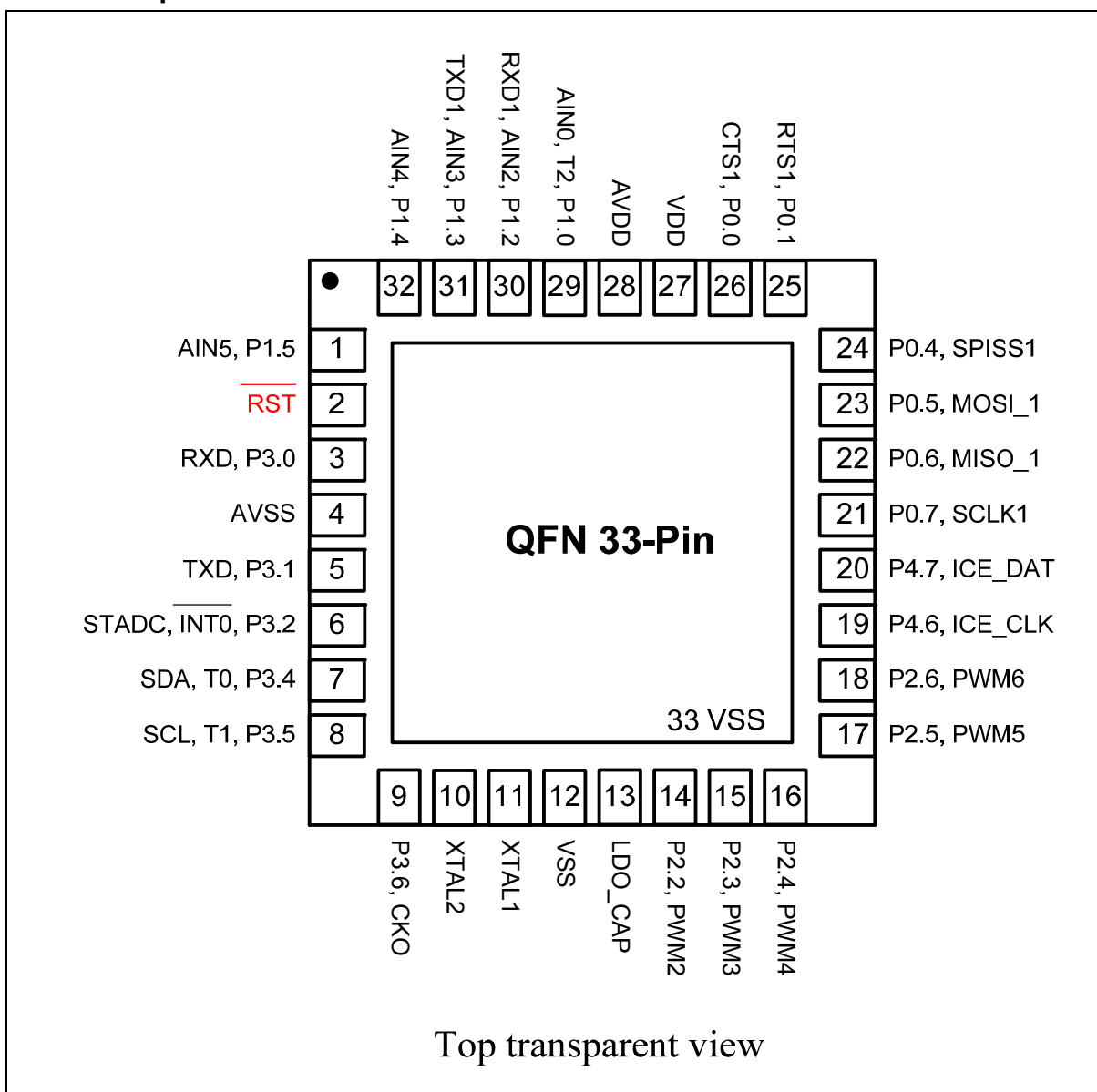


图 5.1-1 NuMicro™ M051 系列QFN33 引脚图

## 5.2 LQFP 48 pin

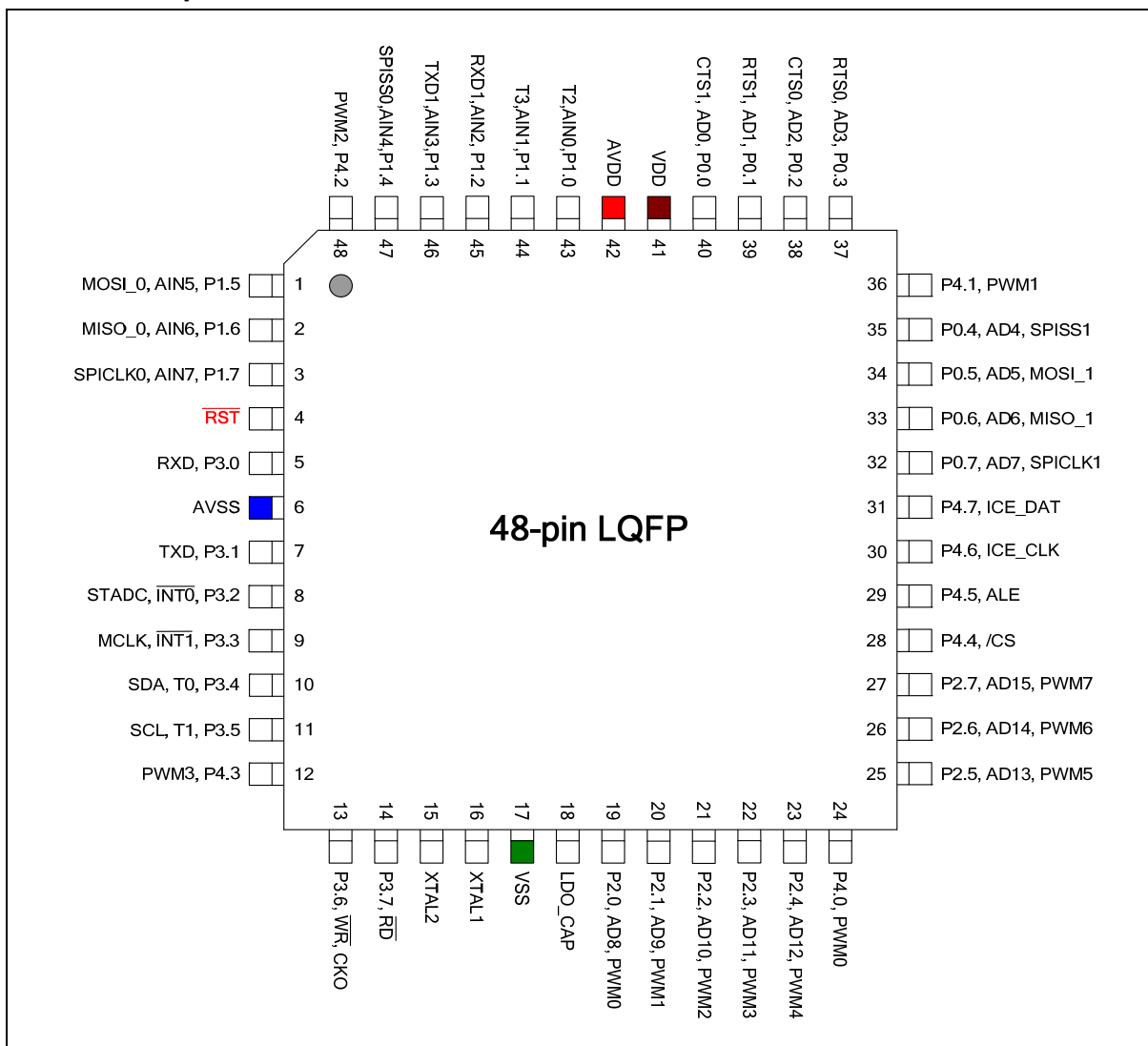


图 5.2-1 NuMicro™ M051 系列 LQFP-48引脚图

## 5.3 管脚描述

管脚号		符号	复用功能		类型 <sup>[1]</sup>	描述
QFN33	LQFP48		1	2		
11	16	XTAL1			I (ST)	<b>晶振脚1:</b> 内部反向放大器输入管脚。当 FOSC[1:0] (CONFIG3[1:0]) 都为1 (默认值) 时, 系统时钟由外部晶振或谐振器输入。
10	15	XTAL2			O	<b>晶振脚2:</b> 内部反向放大器输出管脚。此管脚输出晶振脚1的反向信号。
27	41	VDD			P	<b>电源输入脚:</b> 提供 <b>数字</b> 电源V <sub>DD</sub> 。
12 33	17	VSS			P	<b>地:</b> <b>数字</b> 电源地
28	42	AVDD			P	<b>模拟电源输入脚:</b> 提供 <b>模拟</b> 电源 AV <sub>DD</sub>
4	6	AVSS			P	<b>地:</b> <b>模拟</b> 电源地
13	18	LDO_C AP			P	<b>LDO:</b> LDO 输出管脚 <b>注: 必须外接 10uF 电容。</b>
2	4	/RST			I (ST)	<b>复位脚:</b> /RST管脚为施密特出发输入管脚, 用于芯片复位。当该管脚上接入“低”电位, 保持768个内部22MHzRC晶振时钟周期后, 芯片复位。/RST管脚具有上拉电阻, 对该管脚通过外部电容接地, 就可以完成上电复位。
26	40	P0.0	CTS1	AD0	D, I/O	<b>端口0:</b> 端口1是8位的, 具有四种输出模式, 2种输入模式的管脚。并与下列功能复用, 包括CTS1, RTS1, CTS0, RTS0, SPISS1, MOSI_1, MISO_1, 及SPCLK1。 当外部总线接口 (EBI) 被使能时, P0可复用为AD[7:0]。该管脚同时为SPI0的SPISS1, MOSI_1, MISO_1, 及 SCLK1脚。 CTS0/1: 清除UART0/1输入引脚发送引脚
25	39	P0.1	RTS1	AD1	D, I/O	
NC	38	P0.2	CTS0	AD2	D, I/O	
NC	37	P0.3	RTS0	AD3	D, I/O	
24	35	P0.4	SPISS1	AD4	D, I/O	



管脚号		符号	复用功能		类型 <sup>[1]</sup>	描述
QFN33	LQFP48		1	2		
23	34	P0.5	MOSI_1	AD5	D, I/O	RTS0/1: UART0/1发送输出信号请求引脚
22	33	P0.6	MISO_1	AD6	D, I/O	
21	32	P0.7	SPI_SCLK_1	AD7	D, I/O	
29	43	P1.0	T2	AIN0	I/O	<p><b>端口1:</b> 端口1是8位的，具有四种输出模式，2种输入模式的管脚。并与下列功能复用，包括T2, T3, RXD1, TXD1, SPISS0, MOSI_0, MISO_0, 及SCLK0。</p> <p>T2: Timer2的外部输入管脚</p> <p>T3: Timer3的外部输入管脚</p> <p>端口1的管脚同时为SPI1的SPISS0, MOSI_0, MISO_0, 及SCLK1脚。</p> <p>AIN0~AIN7: 用于12位ADC的模拟信号输入脚</p> <p>RXD1/TXD1: 供UART1使用</p>
NC	44	P1.1	T3	AIN1	I/O	
30	45	P1.2	RXD1	AIN2	I/O	
31	46	P1.3	TXD1	AIN3	I/O	
32	47	P1.4	SPISS0	AIN4	I/O	
1	1	P1.5	MOSI_0	AIN5	I/O	
NC	2	P1.6	MISO_0	AIN6	I/O	
NC	3	P1.7	SPICLK0	AIN7	I/O	
NC	19	P2.0	PWM0	AD8	D, I/O	
NC	20	P2.1	PWM1	AD9	D, I/O	
14	21	P2.2	PWM2	AD10	D, I/O	<p><b>端口2:</b> 端口2是8位的，具有4种输出模式，2种输入模式的管脚。并与下列功能复用。</p> <p>当外部总线接口（EBI）被使能时，P2可复用为AD[7:0]。PWM0~PWM7 用于PWM输出功能</p>
15	22	P2.3	PWM3	AD11	D, I/O	
16	23	P2.4	PWM4	AD12	D, I/O	
17	25	P2.5	PWM5	AD13	D, I/O	
18	26	P2.6	PWM6	AD14	D, I/O	
NC	27	P2.7	PWM7	AD15	D, I/O	
3	5	P3.0	RXD		I/O	
5	7	P3.1	TXD		I/O	<p><b>端口3:</b> 端口3是8位的，具有4种输出模式，2种输入模式的管脚。并与下列功能复用。包括RXD, TXD,</p>

管脚号		符号	复用功能		类型 <sup>[1]</sup>	描述
QFN33	LQFP48		1	2		
6	8	P3.2	$\overline{\text{INT0}}$	STADC	I/O	$\overline{\text{INT0}}$ , $\overline{\text{INT1}}$ , T0, T1, $\overline{\text{WR}}$ 及 $\overline{\text{RD}}$ 。 T0: Timer0的外部输入管脚 T1: Timer1的外部输入管脚 RXD/TXD: 供UART0使用SDA/SCK: 供I2C功能使用 MCLK: EBI 时钟输出脚 CKO: HCLK 时钟输出 STADC: ADC 外部触发信号脚
NC	9	P3.3	$\overline{\text{INT1}}$	MCLK	I/O	
7	10	P3.4	T0	SDA	I/O	
8	11	P3.5	T1	SCL	I/O	
9	13	P3.6	$\overline{\text{WR}}$	CKO	I/O	
NC	14	P3.7	$\overline{\text{RD}}$		I/O	
NC	24	P4.0	PWM0		I/O	<b>端口4:</b> 端口4是8位的, 具有4种输出模式, 2种输入模式的管脚。并与下列功能复用。包括/CS, ALE, ICE_CLK 及ICE_DAT。 /CS EBI 的片选信号脚。 ALE (地址锁存使能脚): 用于分别锁存P2, P0上的地址信号。 ICE_CLK/ICE_DAT: 用于JTAG仿真。 当EBI使能, P4.0-P4.3可用作PWM0-3。
NC	36	P4.1	PWM1		I/O	
NC	48	P4.2	PWM2		I/O	
NC	12	P4.3	PWM3		I/O	
NC	28	P4.4	/CS		I/O	
NC	29	P4.5	ALE		I/O	
19	30	P4.6	ICE_CLK		I/O	
20	31	P4.7	ICE_DAT		I/O	

表 5.3-1 NuMicro™ M051 系列引脚描述

[1] 管脚类型 I= 数字输入, O=数字输出; I/O: 准双向; P=电源管脚; ST: Schmitt 触发器.

## 6 功能描述

### 6.1 ARM® Cortex™-M0 内核

Cortex™-M0处理器是32位多级可配置的RISC处理器。它有AMBA AHB-Lite接口和嵌套向量中断控制器（NVIC），具有可选的硬件调试功能，可以执行Thumb指令，并与其它Cortex-M系列兼容。

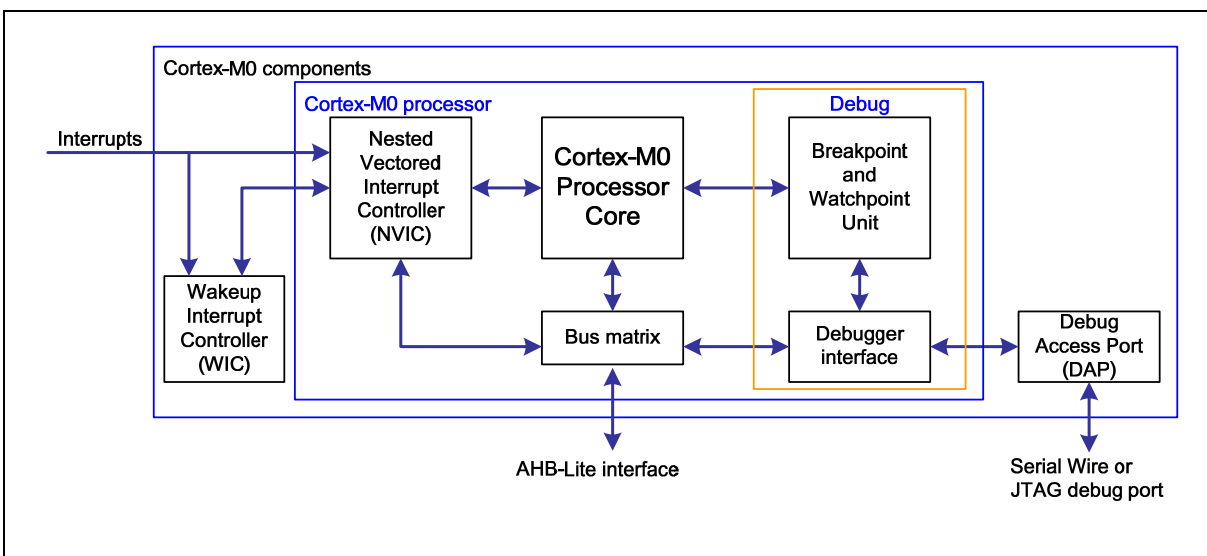


图 6.1-1 功能框图

设备提供:

低门数处理器特征:

- ARMv6-M Thumb 指令集。
- Thumb-2 技术。
- ARMv6-M 兼容 24-bit SysTick 定时器。
- 32-bit 硬件乘法器。
- 系统支持小端（little-endian）数据。
- 准确而及时的中断处理能力。
- 可以禁用和重启的多路加载/存储和多周期乘法可以实现快速中断处理。
- 兼容 C 应用程序二进制接口的异常兼容模式（C-ABI）。  
ARMv6-M（C-ABI）兼容异常模式允许用户使用纯 C 函数实现中断处理。
- （WFI）与（WFE）指令可以进入低功耗的休眠模式。

NVIC 特征:

- 32 个外部中断，具有 4 级优先级。
- 不可屏蔽中断输入（NMI）。



- 支持电平和脉冲中断触发。
- 中断唤醒控制器(WIC)，支持极低功耗休眠模式。

#### 调试

- 四个硬件断点。
- 两个观察点。
- 用于非侵入式代码分析的程序计数采样寄存器（PCSR）。
- 单步向量捕获能力。

#### 总线接口:

- 单一 32 位的 AMBA-3 ABH-Lite 系统接口，向所有的系统外设和存储器提供简单的集成。
- 支持 DAP(Debug Access Port)的单一 32 位的从机端口。



## 6.2 系统管理器

### 6.2.1 概述

系统管理器包括如下功能

- 系统复位
- 系统存储器映射
- 用于管理产品ID, 芯片复位及片上模块复位, 多功能管脚控制的系统管理寄存器
- 系统定时器 (SysTick)
- 嵌套向量中断控制器(NVIC)
- 系统控制寄存器

### 6.2.2 系统复位

下列情况发生时, 系统复位, 复位标志由寄存器**RSTRC**读出.

- 上电复位(POR)
- 复位脚 (/RESET) 上有低电平
- 看门狗复位(WDT)
- 低电压复位(LVR)
- 欠压检测复位(BOD)
- CPU复位
- 系统复位

### 6.2.3 系统电源分配

该器件的电源分为三个部分:

- 由AVDD 和AVSS提供的模拟电源, 为模拟部分工作提供电压。
- 由VDD与VSS提供的数字电源, 为内部稳压器提供电压, 内部稳压器向数字操作与I/O管脚提供固定的2.5V电压。

内部电压管理器 (LDO) 的输出, 需要在相应管脚附近接一颗电容。下图为该器件电压分配图:

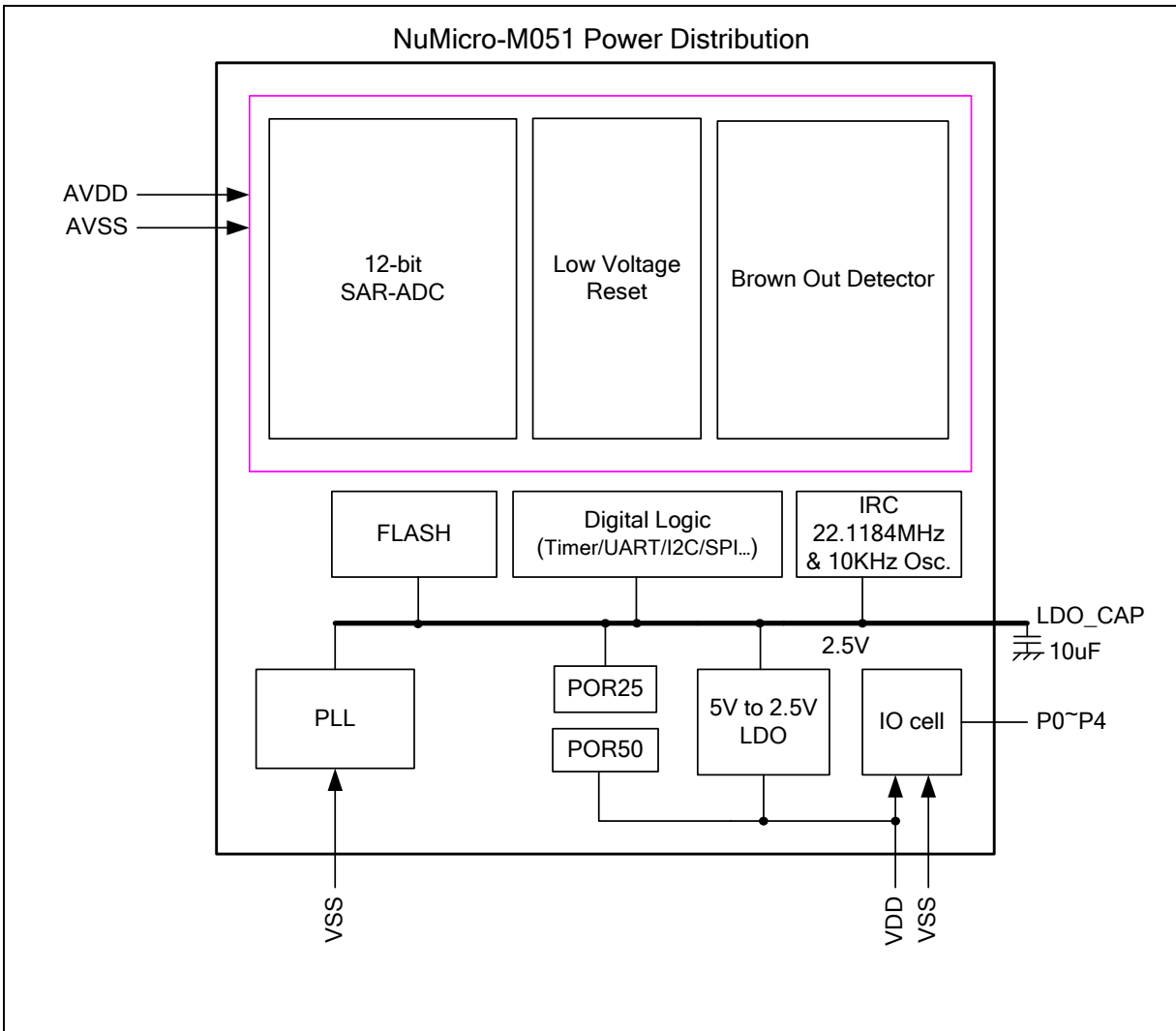


图 6.2-1 NuMicro M051™ 系列电源分配图

## 6.2.4 系统存储器映射

NuMicro M051™ 提供4G字节的寻址空间。内存地址分配情况见表 6.2-1。详细的寄存器地址分配和编程将在后续的讲述各个独立的片上外设的章节被描述。NuMicro M051™ 系列仅支持小端数据格式。

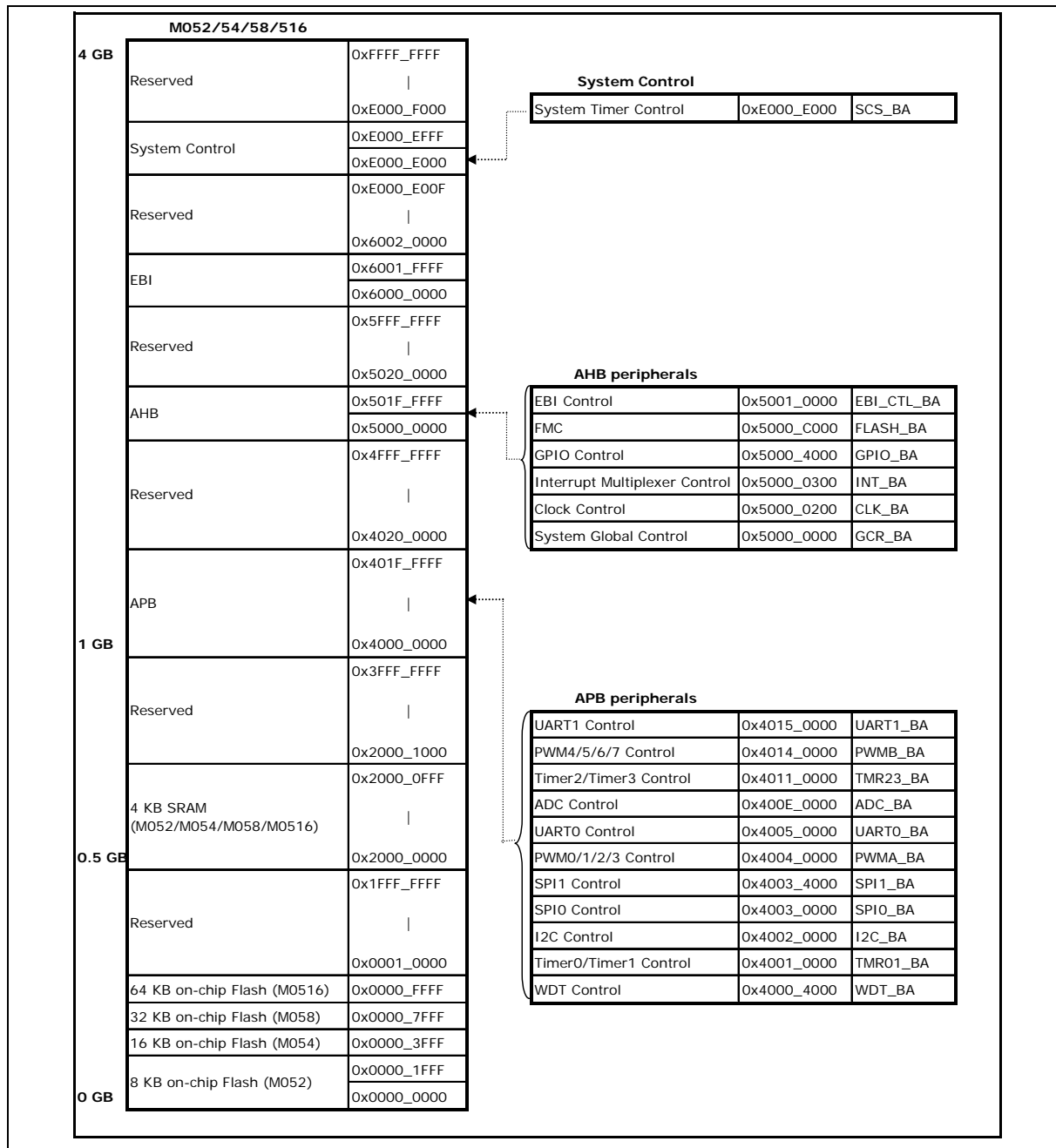
表 6.2-1片上模块的地址空间分配

地址空间	标志	模块
<b>Flash &amp; SRAM 内存空间</b>		
0x0000_0000 – 0x0000_FFFF	FLASH_BA	FLASH内存空间(64KB)
0x2000_0000 – 0x2000_0FFF	SRAM_BA	SRAM内存空间(4KB)
<b>AHB模块空间(0x5000_0000 – 0x501F_FFFF)</b>		
0x5000_0000 – 0x5000_01FF	GCR_BA	系统全局控制寄存器
0x5000_0200 – 0x5000_02FF	CLK_BA	时钟控制寄存器
0x5000_0300 – 0x5000_03FF	INT_BA	多路中断控制寄存器
0x5000_4000 – 0x5000_7FFF	GPIO_BA	GPIO (P0~P4) 控制寄存器
0x5000_C000 – 0x5000_FFFF	FMC_BA	Flash 内存控制寄存器
0x5001_0000 – 0x5001_3FFF	EBI_CTL_BA	EBI 控制寄存器 (128KB)
<b>EBI 空间 (0x6000_0000 ~ 0x6001_FFFF)</b>		
0x6000_0000 – 0x6001_FFFF	EBI_BA	EBI 空间
<b>APB模块空间(0x4000_0000 ~ 0x400F_FFFF)</b>		
0x4000_4000 – 0x4000_7FFF	WDT_BA	看门狗控制寄存器
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 控制寄存器
0x4002_0000 – 0x4002_3FFF	I2C_BA	I2C接口控制寄存器
0x4003_0000 – 0x4003_3FFF	SPI0_BA	带主/从功能的SPI0控制寄存器
0x4003_4000 – 0x4003_7FFF	SPI1_BA	带主/从功能的SPI1 控制寄存器
0x4004_0000 – 0x4004_3FFF	PWMA_BA	PWM0/1/2/3 控制寄存器
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0控制寄存器
0x400E_0000 – 0x400E_FFFF	ADC_BA	ADC控制寄存器
0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3控制寄存器

0x4014_0000 – 0x4014_3FFF	PWMB_BA	PWM4/5/6/7控制寄存器
0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 控制寄存器
<b>System Control Space (0xE000_E000 ~ 0xE000_EFFF)</b>		
0xE000_E010 – 0xE000_E0FF	SCS_BA	System 定时器控制寄存器
0xE000_E100 – 0xE000_ECFF	SCS_BA	外部中断控制器控制寄存器
0xE000_ED00 – 0xE000_ED8F	SCS_BA	System控制寄存器

表 6.2-2 片上模块的地址空间分配

## 6.2.5 系统存储器映射表



## 6.2.6 系统管理器控制寄存器

寄存器	偏移量	R/W	描述	复位后的值
<b>GCR_BA = 0x5000_0000</b>				
<b>PDID</b>	GCR_BA+0x00	R	器件ID寄存器	0x0000_5200
<b>RSTSRC</b>	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX
<b>IPRSTC1</b>	GCR_BA+0x08	R/W	外设复位控制寄存器1	0x0000_0000
<b>IPRSTC2</b>	GCR_BA+0x0C	R/W	外设复位控制寄存器2	0x0000_0000
<b>BODCR</b>	GCR_BA+0x18	R/W	欠压检测控制寄存器	0x0000_008X
<b>PORCR</b>	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_00xx
<b>P0_MFP</b>	GCR_BA+0x30	R/W	P0 复用功能和输入类型控制寄存器	0x0000_0000
<b>P1_MFP</b>	GCR_BA+0x34	R/W	P1 复用功能和输入类型控制寄存器	0x0000_0000
<b>P2_MFP</b>	GCR_BA+0x38	R/W	P2 复用功能和输入类型控制寄存器	0x0000_0000
<b>P3_MFP</b>	GCR_BA+0x3C	R/W	P3 复用功能和输入类型控制寄存器	0x0000_0000
<b>P4_MFP</b>	GCR_BA+0x40	R/W	P4 输入类型控制寄存器	0x0000_00C0
<b>REGWRPROT</b>	GCR_BA+0x100	R/W	寄存器写保护控制寄存器	0x0000_0000

## 器件ID寄存器(PDID)

寄存器	偏移量	R/W	描述	复位后的值
PDID	GCR_BA+0x00	R	器件ID寄存器	0x0000_5200 <sup>[1]</sup>

[1] 复位后每个器件都有一个独一无二的号码。

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

Bits	描述
[31:0]	<p><b>PDID</b></p> <p>产品器件识别码。 该寄存器反映器件的识别码。S/W可以读该寄存器识别所使用的器件。 例如, M052LAN PDID 的识别码是0x0000_5200.</p>

NuMicro M051™ 系列	产品器件识别码
M052LAN	0x00005200
M054LAN	0x00005400
M058LAN	0x00005800
M0516LAN	0x00005A00
M052ZAN	0x00005203
M054ZAN	0x00005403
M058ZAN	0x00005803
M0516ZAN	0x00005A03



## 系统复位源寄存器(RSTSRC)

该寄存器提供一些信息用于识别引起芯片上次操作复位的复位源。

寄存器	偏移量	R/W	描述	复位后的值
RSTSRC	GCR_BA+04	R/W	系统复位源寄存器	0x0000_00XX

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
RSTS_CPU	RSTS_PMU	RSTS_MCU	RSTS_BOD	RSTS_LVR	RSTS_WDT	RSTS_RESET	RSTS_POR

Bits	描述	
[31:8]	保留	保留
[7]	RSTS_CPU	<p>当软件向CPU_RST (IPRSTCR1[1])写入“1”，复位Cortex-M0 CPU 内核和FLASH 控制器（FMC）时，RSTS_CPU 标志由硬件置位。.</p> <p>1= 软件置CPU_RST为1时， Cortex-M0 CPU 内核与FMC复位.</p> <p>0= CPU无复位</p> <p>向该位写1清零.</p>
[6]	保留	保留
[5]	RSTS_MCU	<p>RSTS_MCU由来自MCU Cortex_M0 的“复位信号“置位，以表示当前的复位源.</p> <p>1= MCU Cortex_M0 在软件向SYSRESTREQ(AIRCR[2])写1时，发出复位信号以复位系统.</p> <p>0= MCU无复位</p> <p>向该位写1清零.</p>



[4]	<b>RSTS_BOD</b>	<p>RSTS_BOD标志位由欠压检测模块的“复位信号”置1，用于表示当前复位源。</p> <p>1: 欠压检验模块发出复位信号使系统复位. 0: BOD无复位 向该位写1清零.</p>
[3]	<b>RSTS_LVR</b>	<p>RSTS_LVR标志位由低压复位模块的“复位信号”置1，用于表示当前复位源。</p> <p>1: 低压 LVR 模块发出复位信号使系统复位. 0: LVR无复位 向该位写1清零.</p>
[2]	<b>RSTS_WDT</b>	<p>RSTS_WDT 标志位由看门狗模块的“复位信号”置1，用于说明当前复位源。</p> <p>1: 看门狗模块发出复位信号使系统复位. 0: 没有看门狗复位信号 向该位写1清零</p>
[1]	<b>RSTS_RESET</b>	<p>RSTS_RESET标志位由/RESET脚的“复位信号”置1，用于说明当前复位源。</p> <p>1: /RESET脚上发出复位信号使系统复位. 0: 没有/RESET复位信号 向该位写1清零.</p>
[0]	<b>RSTS_POR</b>	<p>RSTS_POR标志位由POR模块的“复位信号”置1，用于说明当前的复位源。</p> <p>1: 上电复位POR发出复位信号使系统复位. 0: 没有POR复位信号 向该位写1清零.</p>

### 外设复位控制寄存器1 (IPRSTC1)

寄存器	偏移量	R/W	描述	复位后的值
IPRSTC1	GCR_BA+08	R/W	外设复位控制寄存器1	0x0000_0000

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

Bits	描述	
[31:4]	保留	保留
[3]	EBI_RST	<p><b>EBI 控制器复位</b></p> <p>设置该位为“1”，产生复位信号到EBI。用户需要置0才能释放复位状态</p> <p>该位是受保护的位，修改该位时，需要依次向0x5000_0100写入“59h”，“16h”，“88h”解除寄存器保护。参考寄存器 REGWRPROT，地址GCR_BA + 0x100。</p> <p>0= 正常工作 1= EBI IP 复位</p>
[2]	保留	保留
[1]	CPU_RST	<p><b>CPU内核复位</b></p> <p>该位置1，CPU内核和Flash存储控制器复位。两个时钟周期后，该位自动清零。</p> <p>该位是受保护的位，修改该位时，需要依次向0x5000_0100写入“59h”，“16h”，“88h”解除寄存器保护，参考寄存器 REGWRPROT，地址GCR_BA + 0x100。</p> <p>0：正常 1：复位CPU</p>

[0]	CHIP_RST	<p><b>芯片复位</b></p> <p>该位置1，芯片复位，包括CPU内核和所有外设均复位。，两个时钟周期后，该位自动清零。</p> <p>CHIP_RST与POR复位相似, 所有片上模块都复位，芯片设置从FLASH重载</p> <p>CHIP_RST与上电复位一样，所有的芯片模块都复位，芯片设置从flash重新加载</p> <p>该位是受保护的位，修改该位时，需要依次向0x5000_0100写入“59h”，“16h”，“88h”解除寄存器保护。参考寄存器 REGWRPROT，地址GCR_BA + 0x100.</p> <p>0：正常 1：复位芯片</p>
-----	----------	---



## 外设复位控制寄存器2 (IPRSTC2)

置“1”，产生异步复位信号给相应的IP。清零时，从复位状态恢复

寄存器	偏移量	R/W	描述	复位后的值
IPRST2	GCR_BA+0C	R/W	外设复位控制寄存器2	0x0000_0000

31	30	29	28	27	26	25	24
保留			ADC_RST	保留			
23	22	21	20	19	18	17	16
保留		PWM47_RST	PWM03_RST	保留		UART1_RST	UART0_RST
15	14	13	12	11	10	9	8
保留		SPI1_RST	SPI0_RST	保留			I2C_RST
7	6	5	4	3	2	1	0
保留		TMR3_RST	TMR2_RST	TMR1_RST	TMR0_RST	GPIO_RST	保留

Bits	描述	
[31:29]	保留	保留
[28]	ADC_RST	<b>ADC控制器复位</b> “0”: ADC 模块正常工作 “1”: ADC 模块复位
[27:22]	保留	保留
[21]	PWM47_RST	<b>PWM4~7 控制器复位</b> 0= PWM4~7 模块正常工作 1= PWM4~7 模块复位
[20]	PWM03_RST	<b>PWM0~3 控制器复位</b> 0= PWM0~3 模块正常工作 1= PWM0~3 模块复位
[19:18]	保留	保留

[17]	<b>UART1_RST</b>	<b>UART1控制器复位</b> 0= UART1 正常工作 1= UART1 模块复位
[16]	<b>UART0_RST</b>	<b>UART0控制器复位</b> 0= UART0 正常工作 1= UART0 模块复位
[15:14]	保留	保留
[13]	<b>SPI1_RST</b>	<b>SPI1控制器复位</b> 0= SPI1 正常工作 1= SPI1 模块复位
[12]	<b>SPI0_RST</b>	<b>SPI0控制器复位</b> 0= SPI0 正常工作 1= SPI0 模块复位
[11:9]	保留	保留
[8]	<b>I2C_RST</b>	<b>I2C控制器复位</b> 0= I2C 模块 正常工作 1= I2C 模块复位
[7:6]	保留	保留
[5]	<b>TMR3_RST</b>	<b>Timer3控制器复位</b> 0= Timer3 正常工作 1= Timer3 模块复位
[4]	<b>TMR2_RST</b>	<b>Timer2控制器复位</b> 0= Timer2 正常工作 1= Timer2 模块复位
[3]	<b>TMR1_RST</b>	<b>Timer1控制器复位</b> 0= Timer1 正常工作 1= Timer1 模块复位

[2]	<b>TMR0_RST</b>	<b>Timer0控制器复位</b> 0= Timer0 正常工作 1= Timer0 复位
[1]	<b>GPIO_RST</b>	<b>GPIO (P0~P4) 控制器复位</b> 0= GPIO 正常工作 1= GPIO 复位
[0]	保留	保留

## 欠压检测控制寄存器(BODCR)

BODCR 控制寄存器的部分值在flash配置时已经初始化和写保护，用户如果需要改变BODCR的值，需要依次向控制寄存器 0x5000\_0100 写入“59h”，“16h” “88h”，解除寄存器保护。参考寄存器 REGWRPROT，其地址为GCR\_BA+0x100

寄存器	偏移量	R/W	描述	复位后的值
BODCR	GCR_BA+18	R/W	欠压检测控制寄存器	0x0000_008X

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

Bits	描述	
[31:8]	保留	保留
[7]	LVR_EN	<p>低压复位使能（写保护位）</p> <p>输入电源电压低于LVR电路设置时，LVR复位。LVR默认配置下LVR复位是使能的，典型的LVR值为2.0V</p> <p>1= 使能低电压复位功能，使能该位100US后，LVR功能生效(默认)</p> <p>0= 禁用低电压复位功能</p>
[6]	BOD_OUT	<p>欠压检测输出的状态位</p> <p>1 = 欠压检测输出状态为 1，表示检测到的电压低于 BOD_VL 设置。若 BOD_EN 是“0”，该位保持为“0”</p> <p>0 =欠压检测输出状态为0，表示检测到的电压高于BOD_VL设置</p>
[5]	BOD_LPM	<p>低压模式下的欠压检测（写保护位）</p> <p>1 = 使能 BOD 低压模式</p> <p>0 = BOD 工作于正常模式(默认)</p> <p>BOD 在正常模式下消耗电流约为100uA，低压模式下减少到当前的1/10，但BOD响应速度变慢。</p>

[4]	<b>BOD_INTF</b>	<p><b>欠压检测中断标志</b></p> <p>1= 欠压检测到V<sub>DD</sub> 下降到BOD_VL 的设定电压或V<sub>DD</sub> 升到BOD_VL 的设定电压，该位设置为1，如果欠压中断被使能，则发生欠压中断。</p> <p>0= 没有检测到任何电压由V<sub>DD</sub> 下降或上升至BOD_VL 设定值。</p>															
[3]	<b>BOD_RSTEN</b>	<p><b>欠压复位使能（上电初始化和写保护位）</b></p> <p>1 = 使能欠压复位功能，当欠压检测功能使能后，检测的电压低于阈值电压，芯片发生复位</p> <p>默认值由用户在配置flash控制寄存器时的config0 bit[20]设置</p> <p>0 = 使能欠压中断功能，当欠压检测功能使能后，检测的电压低于阈值电压，就发送中断信号给MCU Cortex-M0</p> <p>当BOD_EN使能，且中断被声明时，该中断会持续到将BOD_EN设置为"0"。通过禁用CPU中的NVIC以禁用BOD中断或者通过禁用BOD_EN禁用中断源可禁用CPU响应中断，如果需要BOD功能时，可重新使能BOD_EN功能</p>															
[2:1]	<b>BOD_VL</b>	<p><b>欠压检测阈值电压 电压选择（上电初始化和写保护位）</b></p> <p>默认值 由用户在配置FLASH控制寄存器config0 bit[22:21]时设定</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 25%;">BOV_VL[1]</th> <th style="width: 25%;">BOV_VL[0]</th> <th style="width: 50%;">欠压值</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">4.5V</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">3.8V</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">2.7V</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">2.2V</td> </tr> </tbody> </table>	BOV_VL[1]	BOV_VL[0]	欠压值	1	1	4.5V	1	0	3.8V	0	1	2.7V	0	0	2.2V
BOV_VL[1]	BOV_VL[0]	欠压值															
1	1	4.5V															
1	0	3.8V															
0	1	2.7V															
0	0	2.2V															
[0]	<b>BOD_EN</b>	<p><b>欠压检测使能（上电初始化和写保护位）</b></p> <p>默认值 由用户在配置FLASH控制寄存器config0 bit[23]时设定</p> <p>1 = 使能欠压检测功能</p> <p>0 = 禁用欠压检测功能</p>															





## 上电复位控制寄存器(PORCR)

寄存器	偏移量	R/W	描述	复位后的值
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_00xx

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

Bits	描述	
[31:16]	保留	保留
[15:0]	POR_DIS_CODE	<p>该寄存器用于使能上电复位控制。</p> <p>上电时，POC电路产生复位信号使整个芯片复位，但是电源部分的干扰可能引起POR重新有效。如果将POR_DIS_CODE 设置为0x5AA5，POR复位功能被禁用，直到电源电压很低，设置POR_DIS_CODE 为其他值，或者由芯片的其他复位功能引起复位时，POR功能重新有效，这些复位功能包括：</p> <p>/RESET引脚复位，看看门狗，LVR复位，BOD复位，ICE复位命令和软件复位。</p> <p>该寄存器是受保护的寄存器，写该位需要先向地址0x5000_0100依次写入“59h”，“16h”，“88h”解除寄存器保护。参考寄存器REGWRPROT的设置，其地址为GCR_BA + 0x100。</p>

## 多功能端口0控制寄存器 (P0\_MFP)

寄存器	偏移量	R/W	描述	复位后的值
P0_MFP	GCR_BA+30	R/W	P0复用功能与输入类型控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
P0_TYPE[7:0]							
15	14	13	12	11	10	9	8
P0_ALT[7:0]							
7	6	5	4	3	2	1	0
P0_MFP[7:0]							

Bits	描述																
[31:24]	保留	保留															
[23:16]	P0_TYPE <sub>n</sub>	<b>P0[7:0]史密特触发输入功能使能</b> 1= 使能P0[7:0] I/O史密特触发输入功能 0= 禁用P0[7:0] I/O史密特触发功能输入															
[15]	P0_ALT[7]	P0.7 复用功能选择 P0.7 的功能取决于P0_MFP[7] 和 P0_ALT[7]. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>P0_ALT[7]</th> <th>P0_MFP[7]</th> <th>P0.7 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P0.7</td> </tr> <tr> <td>0</td> <td>1</td> <td>AD7(EBI)</td> </tr> <tr> <td>1</td> <td>0</td> <td>SPICLK1(SPI1)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P0_ALT[7]	P0_MFP[7]	P0.7 的功能	0	0	P0.7	0	1	AD7(EBI)	1	0	SPICLK1(SPI1)	1	1	保留
P0_ALT[7]	P0_MFP[7]	P0.7 的功能															
0	0	P0.7															
0	1	AD7(EBI)															
1	0	SPICLK1(SPI1)															
1	1	保留															

[14]	<b>P0_ALT[6]</b>	<p>P0.6复用功能选择</p> <p>P0.6的功能取决于P0_MFP[6] 和 P0_ALT[6].</p> <table border="1" data-bbox="527 422 1084 716"> <thead> <tr> <th>P0_ALT[6]</th> <th>P0_MFP[6]</th> <th>P0.6 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P0.6</td> </tr> <tr> <td>0</td> <td>1</td> <td>AD6(EBI)</td> </tr> <tr> <td>1</td> <td>0</td> <td>MISO_1(SPI1)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P0_ALT[6]	P0_MFP[6]	P0.6 的功能	0	0	P0.6	0	1	AD6(EBI)	1	0	MISO_1(SPI1)	1	1	保留
P0_ALT[6]	P0_MFP[6]	P0.6 的功能															
0	0	P0.6															
0	1	AD6(EBI)															
1	0	MISO_1(SPI1)															
1	1	保留															
[13]	<b>P0_ALT[5]</b>	<p>P0.5复用功能选择</p> <p>P0.5的功能取决于 P0_MFP[5] 和 P0_ALT[5].</p> <table border="1" data-bbox="527 825 1084 1110"> <thead> <tr> <th>P0_ALT[5]</th> <th>P0_MFP[5]</th> <th>P0.5 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P0.5</td> </tr> <tr> <td>0</td> <td>1</td> <td>AD5(EBI)</td> </tr> <tr> <td>1</td> <td>0</td> <td>MOSI_1(SPI1)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P0_ALT[5]	P0_MFP[5]	P0.5 的功能	0	0	P0.5	0	1	AD5(EBI)	1	0	MOSI_1(SPI1)	1	1	保留
P0_ALT[5]	P0_MFP[5]	P0.5 的功能															
0	0	P0.5															
0	1	AD5(EBI)															
1	0	MOSI_1(SPI1)															
1	1	保留															
[12]	<b>P0_ALT[4]</b>	<p>P0.4复用功能选择</p> <p>of P0.4 的功能取决于 P0_MFP[4] 和 P0_ALT[4].</p> <table border="1" data-bbox="527 1218 1084 1493"> <thead> <tr> <th>P0_ALT[4]</th> <th>P0_MFP[4]</th> <th>P0.4 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P0.4</td> </tr> <tr> <td>0</td> <td>1</td> <td>AD4(EBI)</td> </tr> <tr> <td>1</td> <td>0</td> <td>SPISS1(SPI1)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P0_ALT[4]	P0_MFP[4]	P0.4 的功能	0	0	P0.4	0	1	AD4(EBI)	1	0	SPISS1(SPI1)	1	1	保留
P0_ALT[4]	P0_MFP[4]	P0.4 的功能															
0	0	P0.4															
0	1	AD4(EBI)															
1	0	SPISS1(SPI1)															
1	1	保留															

[11]	<b>P0_ALT[3]</b>	<p>P0.3复用功能选择</p> <p>P0.3的功能取决于P0_MFP[3] 和 P0_ALT[3].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P0_ALT[3]</th> <th style="width: 25%;">P0_MFP[3]</th> <th style="width: 50%;">P0.3的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P0.3</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AD3(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>RTS0(UART0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P0_ALT[3]	P0_MFP[3]	P0.3的功能	0	0	P0.3	0	1	AD3(EBI)	1	0	RTS0(UART0)	1	1	保留
P0_ALT[3]	P0_MFP[3]	P0.3的功能															
0	0	P0.3															
0	1	AD3(EBI)															
1	0	RTS0(UART0)															
1	1	保留															
[10]	<b>P0_ALT[2]</b>	<p>P0.2复用功能选择</p> <p>P0.2的功能取决于P0_MFP[2] 和 P0_ALT[2].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P0_ALT[2]</th> <th style="width: 25%;">P0_MFP[2]</th> <th style="width: 50%;">P0.2的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P0.2</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AD2(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>CTS0(UART0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P0_ALT[2]	P0_MFP[2]	P0.2的功能	0	0	P0.2	0	1	AD2(EBI)	1	0	CTS0(UART0)	1	1	保留
P0_ALT[2]	P0_MFP[2]	P0.2的功能															
0	0	P0.2															
0	1	AD2(EBI)															
1	0	CTS0(UART0)															
1	1	保留															
[9]	<b>P0_ALT[1]</b>	<p>P0.1复用功能选择</p> <p>P0.1 的功能取决于P0_MFP[1] 和 P0_ALT[1].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P0_ALT[1]</th> <th style="width: 25%;">P0_MFP[1]</th> <th style="width: 50%;">P0.1的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P0.1</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AD1(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>RTS1(UART1)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P0_ALT[1]	P0_MFP[1]	P0.1的功能	0	0	P0.1	0	1	AD1(EBI)	1	0	RTS1(UART1)	1	1	保留
P0_ALT[1]	P0_MFP[1]	P0.1的功能															
0	0	P0.1															
0	1	AD1(EBI)															
1	0	RTS1(UART1)															
1	1	保留															
[8]	<b>P0_ALT[0]</b>	<p>P0.0复用功能选择</p> <p>P0.0的功能取决于 P0_MFP[0] 和 P0_ALT[0].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P0_ALT[0]</th> <th style="width: 25%;">P0_MFP[0]</th> <th style="width: 50%;">P0.0的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P0.0</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AD0(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>CTS1(UART1)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P0_ALT[0]	P0_MFP[0]	P0.0的功能	0	0	P0.0	0	1	AD0(EBI)	1	0	CTS1(UART1)	1	1	保留
P0_ALT[0]	P0_MFP[0]	P0.0的功能															
0	0	P0.0															
0	1	AD0(EBI)															
1	0	CTS1(UART1)															
1	1	保留															

[7:0]	<b>P0_MFP[7:0]</b>	P0复用功能选择 P0 的功能取决于 P0_MFP 和 P0_ALT. 参考P0_ALT 的详细描述.
-------	--------------------	---

## 多功能端口1控制寄存器 (P1\_MFP)

寄存器	偏移量	R/W	描述	复位后的值
P1_MFP	GCR_BA+34	R/W	P1复用功能与输入类型控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
P1_TYPE[7:0]							
15	14	13	12	11	10	9	8
P1_ALT[7:0]							
7	6	5	4	3	2	1	0
P1_MFP[7:0]							

Bits	描述																
[31:24]	保留	保留															
[23:16]	P1_TYPEn	<b>P1[7:0] 输入史密特触发功能使能</b> 1= 使能P1[7:0] I/O输入史密特触发功能 0= 禁用P1[7:0] I/O输入史密特触发功能															
[15]	P1_ALT[7]	<b>P1.7复用功能选择</b> P1.7的功能取决于P1_MFP[7] 和 P1_ALT[7]. <table border="1" style="margin-left: 20px; width: 100%;"> <thead> <tr> <th>P1_ALT[7]</th> <th>P1_MFP[7]</th> <th>P1.7 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P1.7</td> </tr> <tr> <td>0</td> <td>1</td> <td>AIN7(ADC)</td> </tr> <tr> <td>1</td> <td>0</td> <td>SPICLK0(SPI0)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P1_ALT[7]	P1_MFP[7]	P1.7 的功能	0	0	P1.7	0	1	AIN7(ADC)	1	0	SPICLK0(SPI0)	1	1	保留
P1_ALT[7]	P1_MFP[7]	P1.7 的功能															
0	0	P1.7															
0	1	AIN7(ADC)															
1	0	SPICLK0(SPI0)															
1	1	保留															

[14]	P1_ALT[6]	<p>P1.6复用功能选择</p> <p>P1.6 的功能取决于 P1_MFP[6] 和 P1_ALT[6].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P1_ALT[6]</th> <th style="width: 25%;">P1_MFP[6]</th> <th style="width: 50%;">P1.6 的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P1.6</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AIN6(ADC)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>MISO_0(SPI0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P1_ALT[6]	P1_MFP[6]	P1.6 的功能	0	0	P1.6	0	1	AIN6(ADC)	1	0	MISO_0(SPI0)	1	1	保留
P1_ALT[6]	P1_MFP[6]	P1.6 的功能															
0	0	P1.6															
0	1	AIN6(ADC)															
1	0	MISO_0(SPI0)															
1	1	保留															
[13]	P1_ALT[5]	<p>P1.5复用功能选择</p> <p>P1.5的功能取决于P1_MFP[5] 和 P1_ALT[5].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P1_ALT[5]</th> <th style="width: 25%;">P1_MFP[5]</th> <th style="width: 50%;">P1.5 的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P1.5</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AIN5(ADC)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>MOSI_0(SPI0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P1_ALT[5]	P1_MFP[5]	P1.5 的功能	0	0	P1.5	0	1	AIN5(ADC)	1	0	MOSI_0(SPI0)	1	1	保留
P1_ALT[5]	P1_MFP[5]	P1.5 的功能															
0	0	P1.5															
0	1	AIN5(ADC)															
1	0	MOSI_0(SPI0)															
1	1	保留															
[12]	P1_ALT[4]	<p>P1.4 复用功能选择</p> <p>P1.4的功能取决于P1_MFP[4] 和 P1_ALT[4].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P1_ALT[4]</th> <th style="width: 25%;">P1_MFP[4]</th> <th style="width: 50%;">P1.4的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P1.4</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AIN4(ADC)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>SPISS0(SPI0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P1_ALT[4]	P1_MFP[4]	P1.4的功能	0	0	P1.4	0	1	AIN4(ADC)	1	0	SPISS0(SPI0)	1	1	保留
P1_ALT[4]	P1_MFP[4]	P1.4的功能															
0	0	P1.4															
0	1	AIN4(ADC)															
1	0	SPISS0(SPI0)															
1	1	保留															
[11]	P1_ALT[3]	<p>P1.3 复用功能选择</p> <p>P1.3的功能取决于P1_MFP[3] and P1_ALT[3].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P1_ALT[3]</th> <th style="width: 25%;">P1_MFP[3]</th> <th style="width: 50%;">P1.3的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P1.3</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AIN3(ADC)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>TXD1(UART1)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P1_ALT[3]	P1_MFP[3]	P1.3的功能	0	0	P1.3	0	1	AIN3(ADC)	1	0	TXD1(UART1)	1	1	保留
P1_ALT[3]	P1_MFP[3]	P1.3的功能															
0	0	P1.3															
0	1	AIN3(ADC)															
1	0	TXD1(UART1)															
1	1	保留															

[10]	P1_ALT[2]	<p>P1.2 复用功能选择</p> <p>P1.2的功能取决于P1_MFP[2] and P1_ALT[2].</p> <table border="1" data-bbox="526 422 1105 667"> <thead> <tr> <th>P1_ALT[2]</th> <th>P1_MFP[2]</th> <th>P1.2的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P1.2</td> </tr> <tr> <td>0</td> <td>1</td> <td>AIN2(ADC)</td> </tr> <tr> <td>1</td> <td>0</td> <td>RXD1(UART1)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P1_ALT[2]	P1_MFP[2]	P1.2的功能	0	0	P1.2	0	1	AIN2(ADC)	1	0	RXD1(UART1)	1	1	保留
P1_ALT[2]	P1_MFP[2]	P1.2的功能															
0	0	P1.2															
0	1	AIN2(ADC)															
1	0	RXD1(UART1)															
1	1	保留															
[9]	P1_ALT[1]	<p>P1.1 复用功能选择</p> <p>P1.1的功能取决于P1_MFP[1] and P1_ALT[1].</p> <table border="1" data-bbox="526 772 1105 1018"> <thead> <tr> <th>P1_ALT[1]</th> <th>P1_MFP[1]</th> <th>P1.1的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P1.1</td> </tr> <tr> <td>0</td> <td>1</td> <td>AIN1(ADC)</td> </tr> <tr> <td>1</td> <td>0</td> <td>T3(Timer3)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P1_ALT[1]	P1_MFP[1]	P1.1的功能	0	0	P1.1	0	1	AIN1(ADC)	1	0	T3(Timer3)	1	1	保留
P1_ALT[1]	P1_MFP[1]	P1.1的功能															
0	0	P1.1															
0	1	AIN1(ADC)															
1	0	T3(Timer3)															
1	1	保留															
[8]	P1_ALT[0]	<p>P1.0 复用功能选择</p> <p>P1.0的功能取决于P1_MFP[0] and P1_ALT[0].</p> <table border="1" data-bbox="526 1123 1105 1369"> <thead> <tr> <th>P1_ALT[0]</th> <th>P1_MFP[0]</th> <th>P1.0的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P1.0</td> </tr> <tr> <td>0</td> <td>1</td> <td>AIN0(ADC)</td> </tr> <tr> <td>1</td> <td>0</td> <td>T2(Timer2)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P1_ALT[0]	P1_MFP[0]	P1.0的功能	0	0	P1.0	0	1	AIN0(ADC)	1	0	T2(Timer2)	1	1	保留
P1_ALT[0]	P1_MFP[0]	P1.0的功能															
0	0	P1.0															
0	1	AIN0(ADC)															
1	0	T2(Timer2)															
1	1	保留															
[7:0]	P1_MFP[7:0]	<p>P1 复用功能选择</p> <p>P1的功能取决于P1_MFP 和 P1_ALT.</p> <p>参考P1_ALT 的详细描述.</p>															



## 多功能端口2控制寄存器(P2\_MFP)

寄存器	偏移量	R/W	描述	复位后的值
P2_MFP	GCR_BA+38	R/W	P2复用功能与输入类型控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
P2_TYPE[7:0]							
15	14	13	12	11	10	9	8
P2_ALT[7:0]							
7	6	5	4	3	2	1	0
P2_MFP[7:0]							

Bits	描述																
[31:24]	保留	保留															
[23:16]	P2_TYPEn	<b>P2[7:0] 输入史密特触发功能使能</b> 1= 使能P2[7:0] I/O输入史密特触发功能 0= 禁用P2[7:0] I/O输入史密特触发功能															
[15]	P2_ALT[7]	<b>P2.7 复用功能选择</b> P2.7的功能取决于P2_MFP[7] and P2_ALT[7]. <table border="1" style="margin-left: 20px; width: 100%;"> <thead> <tr> <th>P2_ALT[7]</th> <th>P2_MFP[7]</th> <th>P2.7 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P2.7</td> </tr> <tr> <td>0</td> <td>1</td> <td>AD15(EBI)</td> </tr> <tr> <td>1</td> <td>0</td> <td>PWM7(PWM generator 6)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P2_ALT[7]	P2_MFP[7]	P2.7 的功能	0	0	P2.7	0	1	AD15(EBI)	1	0	PWM7(PWM generator 6)	1	1	保留
P2_ALT[7]	P2_MFP[7]	P2.7 的功能															
0	0	P2.7															
0	1	AD15(EBI)															
1	0	PWM7(PWM generator 6)															
1	1	保留															

[14]	P2_ALT[6]	<p>P2.6 复用功能选择</p> <p>P2.6的功能取决于P2_MFP[6] and P2_ALT[6].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P2_ALT[6]</th> <th style="width: 25%;">P2_MFP[6]</th> <th style="width: 50%;">P2.6 的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P2.6</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AD14(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>PWM6(PWM generator 6)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P2_ALT[6]	P2_MFP[6]	P2.6 的功能	0	0	P2.6	0	1	AD14(EBI)	1	0	PWM6(PWM generator 6)	1	1	保留
P2_ALT[6]	P2_MFP[6]	P2.6 的功能															
0	0	P2.6															
0	1	AD14(EBI)															
1	0	PWM6(PWM generator 6)															
1	1	保留															
[13]	P2_ALT[5]	<p>P2.5 复用功能选择</p> <p>P2.5的功能取决于P2_MFP[5] and P2_ALT[5].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P2_ALT[5]</th> <th style="width: 25%;">P2_MFP[5]</th> <th style="width: 50%;">P2.5 的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P2.5</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AD13(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>PWM5(PWM generator 4)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P2_ALT[5]	P2_MFP[5]	P2.5 的功能	0	0	P2.5	0	1	AD13(EBI)	1	0	PWM5(PWM generator 4)	1	1	保留
P2_ALT[5]	P2_MFP[5]	P2.5 的功能															
0	0	P2.5															
0	1	AD13(EBI)															
1	0	PWM5(PWM generator 4)															
1	1	保留															
[12]	P2_ALT[4]	<p>P2.4 复用功能选择</p> <p>P2.4的功能取决于P2_MFP[4] and P2_ALT[4].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P2_ALT[4]</th> <th style="width: 25%;">P2_MFP[4]</th> <th style="width: 50%;">P2.4的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P2.4</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AD12(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>PWM4(PWM generator 4)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P2_ALT[4]	P2_MFP[4]	P2.4的功能	0	0	P2.4	0	1	AD12(EBI)	1	0	PWM4(PWM generator 4)	1	1	保留
P2_ALT[4]	P2_MFP[4]	P2.4的功能															
0	0	P2.4															
0	1	AD12(EBI)															
1	0	PWM4(PWM generator 4)															
1	1	保留															
[11]	P2_ALT[3]	<p>P2.3 复用功能选择</p> <p>P2.3的功能取决于P2_MFP[3] and P2_ALT[3].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P2_ALT[3]</th> <th style="width: 25%;">P2_MFP[3]</th> <th style="width: 50%;">P2.3的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P2.3</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>AD11(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>PWM3(PWM generator 2)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P2_ALT[3]	P2_MFP[3]	P2.3的功能	0	0	P2.3	0	1	AD11(EBI)	1	0	PWM3(PWM generator 2)	1	1	保留
P2_ALT[3]	P2_MFP[3]	P2.3的功能															
0	0	P2.3															
0	1	AD11(EBI)															
1	0	PWM3(PWM generator 2)															
1	1	保留															

[10]	P2_ALT[2]	<p>P2.2 复用功能选择</p> <p>P2.2的功能取决于P2_MFP[2] and P2_ALT[2].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">P2_ALT[2]</th> <th style="text-align: center;">P2_MFP[2]</th> <th style="text-align: center;">P2.2的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">P2.2</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">AD10(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">PWM2(PWM generator 2)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">保留</td> </tr> </tbody> </table>	P2_ALT[2]	P2_MFP[2]	P2.2的功能	0	0	P2.2	0	1	AD10(EBI)	1	0	PWM2(PWM generator 2)	1	1	保留
P2_ALT[2]	P2_MFP[2]	P2.2的功能															
0	0	P2.2															
0	1	AD10(EBI)															
1	0	PWM2(PWM generator 2)															
1	1	保留															
[9]	P2_ALT[1]	<p>P2.1 复用功能选择</p> <p>P2.1的功能取决于P2_MFP[1] and P2_ALT[1].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">P2_ALT[1]</th> <th style="text-align: center;">P2_MFP[1]</th> <th style="text-align: center;">P2.1的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">P2.1</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">AD9(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">PWM1(PWM generator 0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">保留</td> </tr> </tbody> </table>	P2_ALT[1]	P2_MFP[1]	P2.1的功能	0	0	P2.1	0	1	AD9(EBI)	1	0	PWM1(PWM generator 0)	1	1	保留
P2_ALT[1]	P2_MFP[1]	P2.1的功能															
0	0	P2.1															
0	1	AD9(EBI)															
1	0	PWM1(PWM generator 0)															
1	1	保留															
[8]	P2_ALT[0]	<p>P2.0 复用功能选择</p> <p>P2.0的功能取决于P2_MFP[0] and P2_ALT[0].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">P2_ALT[0]</th> <th style="text-align: center;">P2_MFP[0]</th> <th style="text-align: center;">P2.0的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">P2.0</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">AD8(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">PWM0(PWM generator 0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">保留</td> </tr> </tbody> </table>	P2_ALT[0]	P2_MFP[0]	P2.0的功能	0	0	P2.0	0	1	AD8(EBI)	1	0	PWM0(PWM generator 0)	1	1	保留
P2_ALT[0]	P2_MFP[0]	P2.0的功能															
0	0	P2.0															
0	1	AD8(EBI)															
1	0	PWM0(PWM generator 0)															
1	1	保留															
[7:0]	P2_MFP[7:0]	<p>P2 复用功能选择</p> <p>P2的功能取决于P2_MFP 和 P2_ALT.</p> <p>参考P2_ALT 详细描述.</p>															

## 多功能端口3控制寄存器(P3\_MFP)

寄存器	偏移量	R/W	描述	复位后的值
P3_MFP	GCR_BA+3C	R/W	P3复用功能与输入类型控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
P3_TYPE[7:0]							
15	14	13	12	11	10	9	8
P3_ALT[7:0]							
7	6	5	4	3	2	1	0
P3_MFP[7:0]							

Bits	描述													
[31:24]	保留	保留												
[23:16]	P3_TYPEn	<b>P3[7:0] 输入史密特触发功能使能</b> 1= 使能P3[7:0] I/O输入史密特触发功能 0= 禁用P3[7:0] I/O输入史密特触发功能												
[15]	P3_ALT[7]	P3.7 复用功能选择 P3.7的功能取决于P3_MFP[7] and P3_ALT[7]. <table border="1" style="margin-left: 20px; width: 100%;"> <thead> <tr> <th>P3_ALT[7]</th> <th>P3_MFP[7]</th> <th>P3.7 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P3.7</td> </tr> <tr> <td>0</td> <td>1</td> <td>RD(EBI)</td> </tr> <tr> <td>1</td> <td>x</td> <td>保留</td> </tr> </tbody> </table>	P3_ALT[7]	P3_MFP[7]	P3.7 的功能	0	0	P3.7	0	1	RD(EBI)	1	x	保留
P3_ALT[7]	P3_MFP[7]	P3.7 的功能												
0	0	P3.7												
0	1	RD(EBI)												
1	x	保留												

[14]	P3_ALT[6]	<p>P3.6 复用功能选择</p> <p>P3.6的功能取决于P3_MFP[6] 和 P3_ALT[6].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">P3_ALT[6]</th> <th style="width: 20%;">P3_MFP[6]</th> <th style="width: 60%;">P3.6 的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P3.6</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>WR(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>CKO(Clock Driver output)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P3_ALT[6]	P3_MFP[6]	P3.6 的功能	0	0	P3.6	0	1	WR(EBI)	1	0	CKO(Clock Driver output)	1	1	保留
P3_ALT[6]	P3_MFP[6]	P3.6 的功能															
0	0	P3.6															
0	1	WR(EBI)															
1	0	CKO(Clock Driver output)															
1	1	保留															
[13]	P3_ALT[5]	<p>P3.5 复用功能选择</p> <p>P3.5的功能取决于P3_MFP[5] 和 P3_ALT[5].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">P3_ALT[5]</th> <th style="width: 20%;">P3_MFP[5]</th> <th style="width: 60%;">P3.5 的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P3.5</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>T1(Timer1)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>SCL(I2C)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P3_ALT[5]	P3_MFP[5]	P3.5 的功能	0	0	P3.5	0	1	T1(Timer1)	1	0	SCL(I2C)	1	1	保留
P3_ALT[5]	P3_MFP[5]	P3.5 的功能															
0	0	P3.5															
0	1	T1(Timer1)															
1	0	SCL(I2C)															
1	1	保留															
[12]	P3_ALT[4]	<p>P3.4 复用功能选择</p> <p>P3.4的功能取决于P3_MFP[4] 和 P3_ALT[4].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">P3_ALT[4]</th> <th style="width: 20%;">P3_MFP[4]</th> <th style="width: 60%;">P3.4的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P3.4</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>T0(Timer0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>SDA(I2C)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>保留</td> </tr> </tbody> </table>	P3_ALT[4]	P3_MFP[4]	P3.4的功能	0	0	P3.4	0	1	T0(Timer0)	1	0	SDA(I2C)	1	1	保留
P3_ALT[4]	P3_MFP[4]	P3.4的功能															
0	0	P3.4															
0	1	T0(Timer0)															
1	0	SDA(I2C)															
1	1	保留															
[11]	P3_ALT[3]	<p>P3.3 复用功能选择</p> <p>P3.3的功能取决于P3_MFP[3] 和 P3_ALT[3].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">P3_ALT[3]</th> <th style="width: 20%;">P3_MFP[3]</th> <th style="width: 60%;">P3.3的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P3.3</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>/INT1</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>MCLK(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">x</td> <td>保留</td> </tr> </tbody> </table>	P3_ALT[3]	P3_MFP[3]	P3.3的功能	0	0	P3.3	0	1	/INT1	1	0	MCLK(EBI)	1	x	保留
P3_ALT[3]	P3_MFP[3]	P3.3的功能															
0	0	P3.3															
0	1	/INT1															
1	0	MCLK(EBI)															
1	x	保留															

[10]	P3_ALT[2]	<p>P3.2 复用功能选择</p> <p>P3.2的功能取决于P3_MFP[2] and P3_ALT[2].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 33%;">P3_ALT[2]</th> <th style="width: 33%;">P3_MFP[2]</th> <th style="width: 33%;">P3.2的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">P3.2</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">/INT0</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">保留</td> </tr> </tbody> </table>	P3_ALT[2]	P3_MFP[2]	P3.2的功能	0	0	P3.2	0	1	/INT0	1	1	保留
P3_ALT[2]	P3_MFP[2]	P3.2的功能												
0	0	P3.2												
0	1	/INT0												
1	1	保留												
[9]	P3_ALT[1]	<p>P3.1 复用功能选择</p> <p>P3.1的功能取决于P3_MFP[1] and P3_ALT[1].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 33%;">P3_ALT[1]</th> <th style="width: 33%;">P3_MFP[1]</th> <th style="width: 33%;">P3.1的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">P3.1</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">TXD(UART0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">x</td> <td style="text-align: center;">保留</td> </tr> </tbody> </table>	P3_ALT[1]	P3_MFP[1]	P3.1的功能	0	0	P3.1	0	1	TXD(UART0)	1	x	保留
P3_ALT[1]	P3_MFP[1]	P3.1的功能												
0	0	P3.1												
0	1	TXD(UART0)												
1	x	保留												
[8]	P3_ALT[0]	<p>P3.0 复用功能选择</p> <p>P3.0的功能取决于P3_MFP[0] and P3_ALT[0].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 33%;">P3_ALT[0]</th> <th style="width: 33%;">P3_MFP[0]</th> <th style="width: 33%;">P3.0的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">P3.0</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">RXD(UART0)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">x</td> <td style="text-align: center;">保留</td> </tr> </tbody> </table>	P3_ALT[0]	P3_MFP[0]	P3.0的功能	0	0	P3.0	0	1	RXD(UART0)	1	x	保留
P3_ALT[0]	P3_MFP[0]	P3.0的功能												
0	0	P3.0												
0	1	RXD(UART0)												
1	x	保留												
[7:0]	P3_MFP[7:0]	<p>P3 复用功能选择</p> <p>P3的功能取决于P3_MFP 和 P3_ALT.</p> <p>参考 P3_ALT详细描述.</p>												



## 多功能端口4控制寄存器(P4\_MFP)

寄存器	偏移量	R/W	描述	复位后的值
P4_MFP	GCR_BA+40	R/W	P4复用功能与输入类型控制寄存器	0x0000_00C0

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
P4_TYPE[7:0]							
15	14	13	12	11	10	9	8
P4_ALT[7:0]							
7	6	5	4	3	2	1	0
P4_MFP[7:0]							

Bits	描述													
[31:24]	保留	保留												
[23:16]	P4_TYPEn	<b>P4[7:0] 输入史密特触发功能使能</b> 1= 使能P4[7:0] I/O输入史密特触发功能 0= 禁用P4[7:0] I/O输入史密特触发功能												
[15]	P4_ALT[7]	P4.7 复用功能选择 P4.7的功能取决于P4_MFP[7] and P4_ALT[7]. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>P4_ALT[7]</th> <th>P4_MFP[7]</th> <th>P4.7 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P4.7</td> </tr> <tr> <td>0</td> <td>1</td> <td>ICE_DAT(ICE)</td> </tr> <tr> <td>1</td> <td>x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[7]	P4_MFP[7]	P4.7 的功能	0	0	P4.7	0	1	ICE_DAT(ICE)	1	x	保留
P4_ALT[7]	P4_MFP[7]	P4.7 的功能												
0	0	P4.7												
0	1	ICE_DAT(ICE)												
1	x	保留												
[14]	P4_ALT[6]	P4.6 复用功能选择 P4.6的功能取决于P4_MFP[6] and P4_ALT[6]. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>P4_ALT[6]</th> <th>P4_MFP[6]</th> <th>P4.6 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P4.6</td> </tr> <tr> <td>0</td> <td>1</td> <td>ICE_CLK(ICE)</td> </tr> <tr> <td>1</td> <td>x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[6]	P4_MFP[6]	P4.6 的功能	0	0	P4.6	0	1	ICE_CLK(ICE)	1	x	保留
P4_ALT[6]	P4_MFP[6]	P4.6 的功能												
0	0	P4.6												
0	1	ICE_CLK(ICE)												
1	x	保留												

[13]	P4_ALT[5]	<p>P4.5 复用功能选择</p> <p>P4.5的功能取决于P4_MFP[5] and P4_ALT[5].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P4_ALT[5]</th> <th style="width: 25%;">P4_MFP[5]</th> <th style="width: 50%;">P4.5 的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P4.5</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>ALE(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[5]	P4_MFP[5]	P4.5 的功能	0	0	P4.5	0	1	ALE(EBI)	1	x	保留
P4_ALT[5]	P4_MFP[5]	P4.5 的功能												
0	0	P4.5												
0	1	ALE(EBI)												
1	x	保留												
[12]	P4_ALT[4]	<p>P4.4 复用功能选择</p> <p>P4.4的功能取决于P4_MFP[4] and P4_ALT[4].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P4_ALT[4]</th> <th style="width: 25%;">P4_MFP[4]</th> <th style="width: 50%;">P4.4的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P4.4</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>/CS(EBI)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[4]	P4_MFP[4]	P4.4的功能	0	0	P4.4	0	1	/CS(EBI)	1	x	保留
P4_ALT[4]	P4_MFP[4]	P4.4的功能												
0	0	P4.4												
0	1	/CS(EBI)												
1	x	保留												
[11]	P4_ALT[3]	<p>P4.3 复用功能选择</p> <p>P4.3的功能取决于P4_MFP[3] and P4_ALT[3].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P4_ALT[3]</th> <th style="width: 25%;">P4_MFP[3]</th> <th style="width: 50%;">P4.3的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P4.3</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>PWM3(PWM generator 2)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[3]	P4_MFP[3]	P4.3的功能	0	0	P4.3	0	1	PWM3(PWM generator 2)	1	x	保留
P4_ALT[3]	P4_MFP[3]	P4.3的功能												
0	0	P4.3												
0	1	PWM3(PWM generator 2)												
1	x	保留												
[10]	P4_ALT[2]	<p>P4.2 复用功能选择</p> <p>P4.2的功能取决于P4_MFP[2] and P4_ALT[2].</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">P4_ALT[2]</th> <th style="width: 25%;">P4_MFP[2]</th> <th style="width: 50%;">P4.2的功能</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>P4.2</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>PWM2(PWM generator 2)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[2]	P4_MFP[2]	P4.2的功能	0	0	P4.2	0	1	PWM2(PWM generator 2)	1	x	保留
P4_ALT[2]	P4_MFP[2]	P4.2的功能												
0	0	P4.2												
0	1	PWM2(PWM generator 2)												
1	x	保留												



[9]	P4_ALT[1]	<p>P4.1 复用功能选择</p> <p>P4.1的功能取决于P4_MFP[1] and P4_ALT[1].</p> <table border="1" data-bbox="527 422 1110 646"> <thead> <tr> <th>P4_ALT[1]</th> <th>P4_MFP[1]</th> <th>P4.1的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P4.1</td> </tr> <tr> <td>0</td> <td>1</td> <td>PWM1(PWM generator 0)</td> </tr> <tr> <td>1</td> <td>x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[1]	P4_MFP[1]	P4.1的功能	0	0	P4.1	0	1	PWM1(PWM generator 0)	1	x	保留
P4_ALT[1]	P4_MFP[1]	P4.1的功能												
0	0	P4.1												
0	1	PWM1(PWM generator 0)												
1	x	保留												
[8]	P4_ALT[0]	<p>P4.0 复用功能选择</p> <p>P4.0的功能取决于P4_MFP[0] and P4_ALT[0].</p> <table border="1" data-bbox="527 745 1110 974"> <thead> <tr> <th>P4_ALT[0]</th> <th>P4_MFP[0]</th> <th>P4.0的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P4.0</td> </tr> <tr> <td>0</td> <td>1</td> <td>PWM0(PWM generator 0)</td> </tr> <tr> <td>1</td> <td>x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[0]	P4_MFP[0]	P4.0的功能	0	0	P4.0	0	1	PWM0(PWM generator 0)	1	x	保留
P4_ALT[0]	P4_MFP[0]	P4.0的功能												
0	0	P4.0												
0	1	PWM0(PWM generator 0)												
1	x	保留												
[7:0]	P4_MFP[7:0]	<p>P4 复用功能选择</p> <p>P4的功能取决于P4_MFP 和 P4_ALT.</p> <p>参考P4_ALT 详细描述.</p>												



## 寄存器锁定键地址寄存器(REGWRPROT)

有些系统控制寄存器需要被保护起来，以防止误操作而影响芯片运行，这些寄存器在上电复位到用户解除寄存器保护之前是锁定的。用户可以连续依次写入“59h”，“16h”“88h”到寄存器REGWRPROT（地址0x5000\_0100）来解除寄存器保护。

解除寄存器保护，用户可以检测0x5000\_0100的bit0，“1”表示保护被解除，“0”表示保护未解除。用户可以更新目标寄存器的值，向“0x5000\_0100”写入任何值，就可以重新使能寄存器保护。读该寄存器可获取REGPROTDIS的状态。

寄存器	偏移量	R/W	描述	复位后的值
REGWRPROT	GCR_BA+100	R/W	寄存器锁定键地址寄存器	0x0000_0000

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

Bits	描述		
[31:16]	保留	保留	
[7:1]	REGWRPROT	<b>寄存器写保护码(只写)</b> 一些写保护寄存器可以通过向此处依次写入“59h”，“16h”，“88h”来解除写保护，这个时序完成之后，REGPROTDIS位将被置1，写保护寄存器可以正常写入数据。	
[0]	REGPROTDIS	<b>寄存器写保护(只读)</b> 1 = 解除写保护以写入受保护的寄存器 0 = 使能寄存器的写保护，不能向受保护寄存器写入数据。 受保护的寄存器有：	
		<b>Registers</b>	<b>Address</b>
		IPRSTC1	0x5000_0008
		<b>Note</b>	

		<b>BODCR</b>	0x5000_0018	
		<b>PORCR</b>	0x5000_001C	
		<b>PWRCON</b>	0x5000_0200	在清除电源唤醒中断时bit[6]不被保护
		<b>APBCLK bit[0]</b>	0x5000_0208	bit[0]: 看门狗时钟使能
		<b>CLKSEL0</b>	0x5000_0210	HCLK和CPU STCLK时钟源选择
		<b>CLK_SEL1 bit[1:0]</b>	0x5000_0214	看门狗时钟源选择
		<b>ISPCON</b>	0x5000_C000	Flash在系统（ISP）编程控制寄存器
		<b>WTCR</b>	0x4000_4000	
		<b>FATCON</b>	0x5000_C018	

## 6.2.7 系统定时器(SysTick)

Cortex-M0 包含一个集成的系统定时器, SysTick. SysTick 提供一种简单, 24位写清零, 递减计数, 计数至0后自动重载的计数器, 可灵活控制。计数器可作为实时操作系统的滴答定时器或者一个简单的计数器。

使能后, 定时器从SysTick 当前寄存器(SYST\_CVR)的值向下计数到0, 在下一个时钟边沿, 重新加载寄存器(SYST\_RVR)的值。当计数器减到0时, 标志位COUNTFLAG置位, 读系统定时器的控制与状态寄存器 (SYST\_CSR) 将清零标志位COUNTFLAG。

复位后, SYST\_CVR 的值未知。使能前, 软件应该向寄存器写入0。这样确保定时器以SYST\_RVR中的值计数, 而非任意值。

若SYST\_RVR 是0, 在重新加载后, 定时器将保持当前值0, [这种机制可以用来在不使用系统定时器的使能位的情形下禁用系统定时器。](#)

详情请参考“ARM® Cortex™-M0 Technical Reference Manual”与“ARM® v6-M Architecture Reference Manual”。

## 6.2.7.1 系统定时器控制寄存器映射

R: 只读, W: 只写, R/W: 可读写, W&C: 写1清零

寄存器	偏移量	R/W	描述	复位后的值
SCS_BA = 0xE000_E000				
SYST_CSR	SCS_BA + 010	R/W	SysTick控制与状态	0x0000_0004
SYST_RVR	SCS_BA + 014	R/W	SysTick重装载值	0xFFFF_XXXX
SYST_CVR	SCS_BA + 018	R/W	SysTick 当前值	0xFFFF_XXXX

### SysTick 控制与状态 (SYST\_CSR)

寄存器	偏移量	R/W	描述	复位后的值
SYST_CSR	SCS_BA+0x10	R/W	SysTick控制与状态	0x0000_0004

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

Bits	描述	
[31:17]	保留	保留
[16]	<b>COUNTFLAG</b>	从上次该寄存器读取, 如果定时器计数到0, 则返回1. 计数由1到0时, COUNTFLAG 置位. 在读该位或向系统定时器当前值寄存器 (SYST_CVR) 写时, COUNTFLAG 被清零。
[15:3]	保留	保留

[2]	<b>CLKSRC</b>	1= 内核时钟用于SysTick. 0= 时钟源可选, 参考 <a href="#">STCLK S</a> .
[1]	<b>TICKINT</b>	1: 向下计数到0将引起SysTick 异常而挂起。清SysTick 当前值寄存器的值将不会导致SysTick 挂起。 0: 向下计数到0不会引起SysTick异常而挂起。软件通过设置COUNTFLAG 来确定是否已经发生计数到0
[0]	<b>ENABLE</b>	1: 计数器运行于多脉冲方式。 0: 禁用计数器

## SysTick重载值寄存器 (SYST\_RVR)

寄存器	偏移量	R/W	描述	复位后的值
SYST_RVR	SCS_BA + 014	R/W	SysTick重载值寄存器	0XXXXX_XXXX

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

Bits	描述	
[31:24]	保留	保留
[23:0]	<b>RELOAD</b>	当计数器达到0时，值加载到当前值寄存器。

## SysTick当前值寄存器 (SYST\_CVR)

寄存器	偏移量	R/W	描述	复位后的值
SYST_CVR	SCS_BA + 018	R/W	SysTick当前值寄存器	0XXXXX_XXXX

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

Bits	描述	
[31:24]	保留	保留
[23:0]	CURRENT	当前计数值，为采样时刻的计数器的值，计数器不提供读修改写保护功能，该寄存器为写清零软件写入任何值将清寄存器为0。这些位不支持读为零(read as zero)，参见系统重装载值寄存器 (SYST_RVR)。



### 6.2.8 嵌套向量中断控制器 (NVIC)

Cortex-M0提供中断控制器，作为异常模式的组成部分，称之为“嵌套向量中断控制器(NVIC)”。它与处理器内核紧密联系，并具有以下特性：

- 支持嵌套和向量中断
- 自动保存和恢复处理器状态
- 可动态改变优先级
- 简化的精确的中断延迟

NVIC区分和处理所有支持的异常，所有异常在“处理器模式”处理。NVIC结构支持具有四级优先级的32个 (IRQ[31:0])离散中断。所有的中断和大多数系统异常可以配置为不同优先级。当中断发生时，NVIC将比较新中断与当前中断的优先级，如果新中断优先级高，则立即处理新中断。

当接受任何中断时，ISR的开始地址可从内存的向量表中取得。不需要确定哪个中断被响应，也不要软件分配相关中断服务程序 (ISR) 的起始地址。当起始地址取得时，NVIC将自动保存处理状态，包括以下寄存器“PC, PSR, LR, R0~R3, R12” 的值到栈中。在ISR结束时，NVIC 将从栈中恢复相关寄存器的值，进行正常操作，因此减少了处理中断请求的时间。

NVIC支持末尾连锁“Tail Chaining”，有效处理背对背中断“back-to-back interrupts”，即无需保存和恢复当前状态从而减少在切换当前ISR至挂起的ISR时的延迟时间NVIC还支持晚到“Late Arrival”，改善同时发生的ISR的效率。当较高优先级中断请求发生在当前ISR开始执行之前（保存处理器状态和获取起始地址阶段），NVIC将立即选择处理更高优先级的中断，从而提高了实时性。

详情请参考“ARM® Cortex™-M0 Technical Reference Manual”与“ARM® v6-M Architecture Reference Manual”。

## 6.2.8.1 异常模式和系统中断映射

NuMicro M051™ 系列支持下表所列的异常模式。与所有中断一样，软件可以对其中一些中断设置4级优先级。最高优先级为“0”，最低优先级为“3”，所有用户可配置的优先级的默认值为“0”。注意：优先级为“0”在整个系统中为第4优先级，排在“Reset”，“NMI”与“Hard Fault”之后。

异常号	向量地址	中断号 (Bit in Interrupt Registers)	中断名	源IP	中断描述	掉电
1-15	0x00-0x3C	-	-	-	系统异常	
16	0x40	0	BOD_OUT	Brown-Out	欠压检测中断	Yes
17	0x44	1	WDT_INT	WDT	看门狗定时器中断	Yes
18	0x48	2	EINT0	GPIO	P3.2 脚上的外部信号中断	Yes
19	0x4C	3	EINT1	GPIO	P3.3 脚上的外部信号中断	Yes
20	0x50	4	GP01_INT	GPIO	P0[7:0] / P1[7:0] 外部信号中断	Yes
21	0x54	5	GP234_INT	GPIO	P2[7:0]/P3[7:0]/P4[7:0] 外部信号中断，除 P32 和 P33	Yes
22	0x58	6	PWMA_INT	PWM0~3	PWM0, PWM1, PWM2 和 PWM3 中断	No
23	0x5C	7	PWMB_INT	PWM4~7	PWM4, PWM5, PWM6 和 PWM7 中断	No
24	0x60	8	TMR0_INT	TMR0	Timer 0 中断	No
25	0x64	9	TMR1_INT	TMR1	Timer 1 中断	No
26	0x68	10	TMR2_INT	TMR2	Timer 2 中断	No
27	0x6C	11	TMR3_INT	TMR3	Timer 3 中断	No
28	0x70	12	UART0_INT	UART0	UART0 中断	Yes
29	0x74	13	UART1_INT	UART1	UART1 中断	Yes
30	0x78	14	SPI0_INT	SPI0	SPI0 中断	No
31	0x7C	15	SPI1_INT	SPI1	SPI1 中断	No
32-33	0x80-0x84	16-17	-	-	-	-
34	0x88	18	I2C_INT	I2C	I2C 中断	No
35-43	0x8C-0xAC	19-27	-	-	-	-
44	0xB0	28	PWRWU_INT	CLKC	从掉电状态唤醒的时钟控制器中断	Yes
45	0xB4	29	ADC_INT	ADC	ADC 中断	No

46-47	0xB8-0xBC	30-31	-	-	-	
-------	-----------	-------	---	---	---	--

表 6.2-3 异常模式

异常名称	向量号	优先级
Reset	1	-3
NMI	2	-2
Hard Fault	3	-1
保留	4 ~ 10	保留
SVCall	11	可配置
保留	12 ~ 13	保留
PendSV	14	可配置
SysTick	15	可配置
Interrupt (IRQ0 ~ IRQ31)	16 ~ 47	可配置

表 6.2-4 系统中断映射

### 6.2.8.2 向量表

响应中断时，处理器自动从向量表中取出ISR的起始地址。对于ARMv6-M，向量表的基地址为0x00000000。向量表包括复位后栈的初始值，所有异常处理器的入口地址。向量号表示处理异常的先后次序。

向量表字偏移量	描述
0	SP_main -主堆栈指针
Vector Number	异常入口指针，用向量号表示

表 6.2-5 向量表格式

### 6.2.8.3 操作说明

通过写相应中断使能设置寄存器或清使能寄存器，可以使能NVIC中断或禁用NVIC中断,这些寄存器通过写1使能和写1清零，寄存器读取返回当前相应中断的使能状态，当中断禁用时，中断声明将使中断挂起，因此中断不被激活，如果在禁用时中断被激活，该中断就保持在激活状态，直到通过复位或异常返回来清除。清使能位可以阻止新的相应中断被激活。

NVIC 中断可以使用互补的寄存器对来挂起/解除挂起以使能/禁用这些中断，这些寄存器分别为Set-Pending Register 与 Clear-Pending，可以写1使能和写1清零，这些寄存器读取返回当前相应中断的状态。寄存器Clear-Pending 在中断响应时不影响执行状态。

NVIC中断通过更新32位寄存器中的各个8位字段（每个寄存器支持4个中断）来分配中断的优先级与NVIC相关的通用寄存器都可以在内存系统控制空间的一块区域设置，下一节将作出描述。

### 6.2.8.4 NVIC 控制寄存器

R: 只读, W: 只写, R/W: 可读写, W&C: 写1清零

寄存器	偏移量	R/W	描述	复位后的值
SCS_BA = 0xE000_E000				
NVIC_ISER	SCS_BA + 100	R/W	IRQ0 ~ IRQ31 设置使能控制寄存器	0x0000_0000
NVIC_ICER	SCS_BA + 180	R/W	IRQ0 ~ IRQ31 清使能控制寄存器	0x0000_0000
NVIC_ISPR	SCS_BA + 200	R/W	IRQ0 ~ IRQ31 设置挂起控制寄存器	0x0000_0000
NVIC_ICPR	SCS_BA + 280	R/W	IRQ0 ~ IRQ31 清挂起控制寄存器	0x0000_0000
NVIC_IPR0	SCS_BA + 400	R/W	IRQ0 ~ IRQ3 优先级 控制寄存器	0x0000_0000
NVIC_IPR1	SCS_BA + 404	R/W	IRQ4 ~ IRQ7 优先级控制寄存器	0x0000_0000
NVIC_IPR2	SCS_BA + 408	R/W	IRQ8 ~ IRQ11 优先级控制寄存器	0x0000_0000
NVIC_IPR3	SCS_BA + 40C	R/W	IRQ12 ~ IRQ15 优先级控制寄存器	0x0000_0000
NVIC_IPR4	SCS_BA + 410	R/W	IRQ16 ~ IRQ19 优先级控制寄存器	0x0000_0000
NVIC_IPR5	SCS_BA + 414	R/W	IRQ20 ~ IRQ23 优先级控制寄存器	0x0000_0000
NVIC_IPR6	SCS_BA + 418	R/W	IRQ24 ~ IRQ27 优先级控制寄存器	0x0000_0000
NVIC_IPR7	SCS_BA + 41C	R/W	IRQ28 ~ IRQ31 优先级控制寄存器	0x0000_0000

## IRQ0 ~ IRQ31 设置使能控制寄存器 (NVIC\_ISER)

寄存器	偏移量	R/W	描述	复位的值
NVIC_ISER	SCS_BA + 100	R/W	IRQ0 ~ IRQ31 设置使能控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	SETENA	使能1个或多个中断，每位代表从IRQ0 ~ IRQ31的中断号(向量号：16 ~ 47). 写1使能相关中断 写0无效 寄存器读取返回当前使能状态.

## IRQ0 ~ IRQ31清使能控制寄存器 (NVIC\_ICER)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ICER	SCS_BA + 180	R/W	IRQ0 ~ IRQ31清使能控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	<b>CLRENA</b>	禁用1个或多个中断，每位代表从IRQ0 ~ IRQ31的中断号 (向量号： 16 ~ 47). 写1禁用相应中断 写0无效 寄存器读取返回当前使能状态.

## IRQ0 ~ IRQ31 设置挂起控制寄存器 (NVIC ISPR)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ISPR	SCS_BA + 200	R/W	IRQ0 ~ IRQ31 设置挂起控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	SETPEND	软件写1，挂起相应中断。每位代表从IRQ0 ~ IRQ31 的中断号(向量号： 16 ~ 47). 写0无效 寄存器读取返回当前挂起状态



**IRQ0 ~ IRQ31清挂起控制寄存器 (NVIC\_ICPR)**

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ICPR	SCS_BA + 280	R/W	IRQ0 ~ IRQ31清挂起控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CLRPEND	写1清除相应中断挂起，每位代表从IRQ0 ~ IRQ31的中断号 (向量号: 16 ~ 47). 写0无效. 寄存器读取返回当前挂起状态.

## IRQ0 ~ IRQ3中断优先级寄存器 (NVIC\_IPR0)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR0	SCS_BA + 400	R/W	IRQ0 ~ IRQ3中断优先级寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_3		保留					
23	22	21	20	19	18	17	16
PRI_2		保留					
15	14	13	12	11	10	9	8
PRI_1		保留					
7	6	5	4	3	2	1	0
PRI_0		保留					

Bits	描述	
[31:30]	PRI_3	IRQ3优先级 “0”表示最高优先级& “3”表示最低优先级
[23:22]	PRI_2	IRQ2优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_1	IRQ1优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_0	IRQ0优先级 “0”表示最高优先级& “3”表示最低优先级

## IRQ4 ~ IRQ7中断优先级寄存器 (NVIC IPR1)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR1	SCS_BA + 404	R/W	IRQ4 ~ IRQ7中断优先级寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_7		保留					
23	22	21	20	19	18	17	16
PRI_6		保留					
15	14	13	12	11	10	9	8
PRI_5		保留					
7	6	5	4	3	2	1	0
PRI_4		保留					

Bits	描述	
[31:30]	PRI_7	IRQ7优先级 “0”表示最高优先级&“3”表示最低优先级
[23:22]	PRI_6	IRQ6优先级 “0”表示最高优先级&“3”表示最低优先级
[15:14]	PRI_5	IRQ5优先级 “0”表示最高优先级&“3”表示最低优先级
[7:6]	PRI_4	IRQ4优先级 “0”表示最高优先级&“3”表示最低优先级

## IRQ8 ~ IRQ11中断优先级寄存器 (NVIC\_IPR2)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR2	SCS_BA + 408	R/W	IRQ8 ~ IRQ11中断优先级寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_11		保留					
23	22	21	20	19	18	17	16
PRI_10		保留					
15	14	13	12	11	10	9	8
PRI_9		保留					
7	6	5	4	3	2	1	0
PRI_8		保留					

Bits	描述
[31:30]	PRI_11 IRQ11优先级 “0”表示最高优先级& “3”表示最低优先级
[23:22]	PRI_10 IRQ10优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_9 IRQ9优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_8 IRQ8优先级 “0”表示最高优先级& “3”表示最低优先级

## IRQ12 ~ IRQ15中断优先级寄存器 (NVIC\_IPR3)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR3	SCS_BA + 40C	R/W	IRQ12 ~ IRQ15中断优先级寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_15		保留					
23	22	21	20	19	18	17	16
PRI_14		保留					
15	14	13	12	11	10	9	8
PRI_13		保留					
7	6	5	4	3	2	1	0
PRI_12		保留					

Bits	描述
[31:30]	<b>PRI_15</b> IRQ15优先级 “0”表示最高优先级& “3”表示最低优先级
[23:22]	<b>PRI_14</b> IRQ14优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	<b>PRI_13</b> IRQ13优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	<b>PRI_12</b> IRQ12优先级 “0”表示最高优先级& “3”表示最低优先级

## IRQ16 ~ IRQ19中断优先级寄存器 (NVIC\_IPR4)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR4	SCS_BA + 410	R/W	IRQ16 ~ IRQ19中断优先级寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_19		保留					
23	22	21	20	19	18	17	16
PRI_18		保留					
15	14	13	12	11	10	9	8
PRI_17		保留					
7	6	5	4	3	2	1	0
PRI_16		保留					

Bits	描述	
[31:30]	PRI_19	IRQ19优先级 “0”表示最高优先级& “3”表示最低优先级
[23:22]	PRI_18	IRQ18优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_17	IRQ17优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_16	IRQ16优先级 “0”表示最高优先级& “3”表示最低优先级

## IRQ20 ~ IRQ23中断优先级寄存器 (NVIC\_IPR5)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR5	SCS_BA + 414	R/W	IRQ20 ~ IRQ23中断优先级寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_23		保留					
23	22	21	20	19	18	17	16
PRI_22		保留					
15	14	13	12	11	10	9	8
PRI_21		保留					
7	6	5	4	3	2	1	0
PRI_20		保留					

Bits	描述
[31:30]	PRI_23 IRQ23优先级 “0”表示最高优先级& “3”表示最低优先级
[23:22]	PRI_22 IRQ22优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_21 IRQ21优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_20 IRQ20优先级 “0”表示最高优先级& “3”表示最低优先级

**IRQ24 ~ IRQ27中断优先级寄存器 (NVIC\_IPR6)**

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR6	SCS_BA + 418	R/W	IRQ24 ~ IRQ27中断优先级寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_27		保留					
23	22	21	20	19	18	17	16
PRI_26		保留					
15	14	13	12	11	10	9	8
PRI_25		保留					
7	6	5	4	3	2	1	0
PRI_24		保留					

Bits	描述	
[31:30]	PRI_27	IRQ27优先级 “0”表示最高优先级& “3”表示最低优先级
[23:22]	PRI_26	IRQ26优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_25	IRQ25优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_24	IRQ24优先级 “0”表示最高优先级& “3”表示最低优先级



## IRQ28 ~ IRQ31中断优先级寄存器 (NVIC\_IPR7)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR7	SCS_BA + 41C	R/W	IRQ28 ~ IRQ31中断优先级寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_31		保留					
23	22	21	20	19	18	17	16
PRI_30		保留					
15	14	13	12	11	10	9	8
PRI_29		保留					
7	6	5	4	3	2	1	0
PRI_28		保留					

Bits	描述
[31:30]	PRI_31 IRQ31优先级 “0”表示最高优先级& “3”表示最低优先级
[23:22]	PRI_30 IRQ30优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_29 IRQ29优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_28 IRQ28优先级 “0”表示最高优先级& “3”表示最低优先级

## 6.2.8.5 中断源控制寄存器

除了NVIC相关的中断控制寄存器外，NuMicro M051™系列还有一些特殊控制寄存器执行中断功能，包括“中断源识别”，“NMI 源选择”与“中断测试模式”。描述如下：

R: 只读, W: 只写, R/W: 可读写, W&C: 写1清零

寄存器	偏移量	R/W	描述	复位后的值
INT_BA = 0x5000_0300				
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) 中断源识别	0xxxxx_xxxx
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别	0xxxxx_xxxx
IRQ2_SRC	INT_BA+0x08	R	IRQ2 ((EINT0) 中断源识别	0xxxxx_xxxx
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别	0xxxxx_xxxx
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (P0/1) 中断源识别	0xxxxx_xxxx
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (P2/3/4) 中断源识别	0xxxxx_xxxx
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) 中断源识别	0xxxxx_xxxx
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (PWMB) 中断源识别	0xxxxx_xxxx
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别	0xxxxx_xxxx
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别	0xxxxx_xxxx
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别	0xxxxx_xxxx
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别	0xxxxx_xxxx
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (URT0) 中断源识别	0xxxxx_xxxx
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (URT1) 中断源识别	0xxxxx_xxxx
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别	0xxxxx_xxxx
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) 中断源识别	0xxxxx_xxxx
IRQ16_SRC	INT_BA+0x40	保留	保留	0xxxxx_xxxx
IRQ17_SRC	INT_BA+0x44	保留	保留	0xxxxx_xxxx
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I2C) 中断源识别	0xxxxx_xxxx
IRQ19_SRC	INT_BA+0x4C	保留	保留	0xxxxx_xxxx
IRQ20_SRC	INT_BA+0x50	保留	保留	0xxxxx_xxxx

<b>IRQ21_SRC</b>	INT_BA+0x54	保留	保留	0XXXXX_XXXX
<b>IRQ22_SRC</b>	INT_BA+0x58	保留	保留	0XXXXX_XXXX
<b>IRQ23_SRC</b>	INT_BA+0x5C	保留	保留	0XXXXX_XXXX
<b>IRQ24_SRC</b>	INT_BA+0x60	保留	保留	0XXXXX_XXXX
<b>IRQ25_SRC</b>	INT_BA+0x64	保留	保留	0XXXXX_XXXX
<b>IRQ26_SRC</b>	INT_BA+0x68	保留	保留	0XXXXX_XXXX
<b>IRQ27_SRC</b>	INT_BA+0x6C	保留	保留	0XXXXX_XXXX
<b>IRQ28_SRC</b>	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源识别	0XXXXX_XXXX
<b>IRQ29_SRC</b>	INT_BA+0x74	R	IRQ29 (ADC) 中断源识别	0XXXXX_XXXX
<b>IRQ30_SRC</b>	INT_BA+0x78	保留	保留	0XXXXX_XXXX
<b>IRQ31_SRC</b>	INT_BA+0x7C	保留	保留	0XXXXX_XXXX
<b>NMI_SEL</b>	INT_BA+0x80	R/W	NMI中断源选择控制寄存器	0x0000_0000
<b>MCU_IRQ</b>	INT_BA+0x84	R/W	MCU IRQ号识别寄存器	0x0000_0000

## 中断源识别寄存器(IRQn\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQn_SRC	INT_BA+0x00	R	MCU IRQ0 (BOD) 中断源识别	0xXXXX_XXXX
	INT_BA+0x7C		MCU IRQ31 (保留) 中断源识别	

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

地址	INT-Num	Bits	描述
INT_BA+0x00	0	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: BOD_INT
INT_BA+0x04	1	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: WDT_INT
INT_BA+0x08	2	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: EINT0 -P3.2上的外部中断0
INT_BA+0x0C	3	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: EINT1 - P3.3上的外部中断1

INT_BA+0x10	4	[2:0]	Bit2: 1'b0 Bit1: P1_INT Bit0: P0_INT
INT_BA+0x14	5	[2:0]	Bit2: P4_INT Bit1: P3_INT Bit0: P2_INT
INT_BA+0x18	6	[3:0]	Bit3: PWM3_INT Bit2: PWM2_INT Bit1: PWM1_INT Bit0: PWM0_INT
INT_BA+0x1C	7	[3:0]	Bit3: PWM7_INT Bit2: PWM6_INT Bit1: PWM5_INT Bit0: PWM4_INT
INT_BA+0x20	8	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: TMR0_INT
INT_BA+0x24	9	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: TMR1_INT
INT_BA+0x28	10	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: TMR2_INT
INT_BA+0x2C	11	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: TMR3_INT
INT_BA+0x30	12	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: URT0_INT
INT_BA+0x34	13	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: URT1_INT

INT_BA+0x38	14	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: SPI0_INT
INT_BA+0x3C	15	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: SPI1_INT
INT_BA+0x40	16	[2:0]	保留
INT_BA+0x44	17	[2:0]	保留
INT_BA+0x48	18	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: I2C_INT
INT_BA+0x4C	19	[2:0]	保留
INT_BA+0x50	20	[2:0]	保留
INT_BA+0x54	21	[2:0]	保留
INT_BA+0x58	22	[2:0]	保留
INT_BA+0x5C	23	[2:0]	保留
INT_BA+0x60	24	[2:0]	保留
INT_BA+0x64	25	[2:0]	保留
INT_BA+0x68	26	[2:0]	保留
INT_BA+0x6C	27	[2:0]	保留
INT_BA+0x70	28	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: PWRWU_INT
INT_BA+0x74	29	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: ADC_INT
INT_BA+0x78	30	[2:0]	保留
INT_BA+0x7C	31	[2:0]	保留

## NMI中断源选择控制寄存器(NMI\_SEL)

寄存器	偏移量	R/W	描述	复位后的值
NMI_SEL	INT_BA+0x80	R/W	NMI中断源选择控制寄存器	0x0000_0000

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

Bits	描述	
[31:5]	保留	保留
[4:0]	NMI_SEL	Cortex-M0的NMI 中断源可以从interrupt[31:0]中选择一个 NMI_SEL bit[4:0] 用于选择NMI 中断源

## MCU中断请求源寄存器(MCU\_IRQ)

寄存器	偏移量	R/W	描述	复位后的值
MCU_IRQ	INT_BA+0x84	R/W	MCU中断请求源寄存器	0x0000_0000

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

Bits	描述
[31:0]	<p><b>MCU_IRQ 源寄存器</b></p> <p>MCU_IRQ 从外围设备收集所有中断，同步对Cortex-M0产生中断。以下两种模式均可中断Cortex-M0，正常模式与测试模式。</p> <p>MCU_IRQ从每一个外设收集中断，同步他们，然后触发Cortex-M0中断。</p> <p>MCU_IRQ[n] 是“0”：置 MCU_IRQ[n] 为“1”，向Cortex_M0 NVIC[n]发生一个中断。</p> <p>MCU_IRQ[n] 是“1”：(意味着有中断请求) 置位MCU_bit[n]将清中断</p> <p>MCU_IRQ[n]是“0”：无效。</p>



### 6.2.9 系统控制器寄存器映射

Cortex-M0的主要控制与状态特征由系统控制寄存器中的系统控制寄存器集中管理，包括CPUID，Cortex-M0中断优先级和Cortex-M0电源管理。

更多详情请参考“ARM® Cortex™-M0 Technical Reference Manual”与“ARM® v6-M Architecture Reference Manual”。

**R:** 只读, **W:** 只写, **R/W:** 可读写, **W&C:** 写1清零

寄存器	偏移量	R/W	描述	复位后的值
SCS_BA = 0xE000_E000				
<b>CPUID</b>	SCS_BA + D00	R	CPUID Base 寄存器	0x0000_0000
<b>ICSR</b>	SCS_BA + D04	R/W	中断控制状态寄存器	0x0000_0000
<b>AIRCR</b>	SCS_BA+ 0xD0C	R/W	中断应用和复位控制寄存器	0xFA05_0000
<b>SCR</b>	SCS_BA + D10	R/W	系统控制寄存器	0x0000_0000
<b>SHPR2</b>	SCS_BA + D1C	R/W	系统处理器优先级寄存器2	0x0000_0000
<b>SHPR3</b>	SCS_BA + D20	R/W	系统处理器优先级寄存器3	0x0000_0000

## CPUID Base寄存器(CPUID)

寄存器	偏移量	R/W	描述	复位后的值
CPUID	SCS_BA + D00	R	CPUID Base寄存器	0x 410CC200

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

Bits	描述	
[31:24]	IMPLEMENTER	由ARM分配执行码。( ARM = 0x41)
[23:20]	保留	保留
[19:16]	PART	ARMv6-M 读取值为0xC
[15:4]	PARTNO	读取值为 0xC20.
[3:0]	REVISION	读取值为 0x0

中断控制状态寄存器(ICSR)

寄存器	偏移量	R/W	描述	复位后的值
ICSR	SCS_BA + D04	R/W	中断控制状态寄存器	0x 00000000

31	30	29	28	27	26	25	24
NMIPENDSET	保留		PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	保留
23	22	21	20	19	18	17	16
ISRPREEMPT	ISRPENDING	保留	VECTPENDING[8:4]				
15	14	13	12	11	10	9	8
VECTPENDING[3:0]			保留				VECTACTIVE[8]
7	6	5	4	3	2	1	0
VECTACTIVE[7:0]							

Bits	R/W	描述	
[31]	R/W	NMIPENDSET	设置该位将激活NMI，由于NMI是最高优先级，只要一声明，就被激活。由当前状态读回（1为挂起，0为未挂起）。
[28]	R/W	PENDSVSET	设置PendSV 中断。通常用于请求内容切换。由当前状态读回（1为挂起，0为未挂起）。
[27]	W	PENDSVCLR	写1 清PendSV 中断。
[26]	R/W	PENDSTSET	设置挂起SysTick。由当前状态读回（1为挂起，0为未挂起）。
[25]	W	PENDSTCLR	写1清除挂起SysTick。
[23]	R	ISRPREEMPT	如果置位，挂起异常生效，由调试停止状态退出。
[22]	R	ISRPENDING	表示外部配置中断是否挂起。
[20:12]	R	VECTPENDING	表示最高优先级挂起异常号，挂起状态包括内存使能和掩膜寄存器，不包括PRIMASK。值为0时代表没有异常挂起。
[8:0]	R	VECTACTIVE	0: 线程模式 value > 1: 当前执行异常处理的异常号

## 中断应用和复位控制寄存器(AIRCR)

寄存器	偏移量	R/W	描述	复位后的值
AIRCR	SCS_BA+0xD0C	R/W	中断应用和复位控制寄存器	0xFA05_0000

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

Bits	描述	
[31:16]	VECTORKEY	写该寄存器时，该值为0x05FA, 否则写动作结果不可预测。
[15:3]	保留	保留
[2]	SYSRESETREQ	该位写1，产生复位信号给芯片表示有复位请求。 该位只写，在复位时自动清零。
[1]	VECTCLRACTIVE	该位置1，清除所有有效状态以配置异常。 该位只写，只有在内核挂起时可写。 注:重新初始化堆栈是调试器 (debugger) 的任务
[0]	保留	保留

### 系统控制寄存器(SCR)

寄存器	偏移量	R/W	描述	复位后的值
SCR	SCS_BA + D10	R/W	系统控制寄存器	0x 00000000

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

Bits	描述	
[4]	SEVONPEND	当使能时，不活动到挂起的中断，包括在唤醒事件列表中（WFE指令）。
[2]	SLEEPDEEP	提示从休眠中唤醒需要较长时间。
[1]	SLEEPONEXIT	设置为1，在异常返回到线程模式时，内核进入休眠状态。这是复位后进入的异常环境和工作模式，是代码执行的基础

## 系统处理器优先级寄存器2 (SHPR2)

寄存器	偏移量	R/W	描述	复位后的值
SHPR2	SCS_BA + D1C	R/W	系统处理器优先级寄存器2	0x 00000000

31	30	29	28	27	26	25	24
PRI_11		保留					
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留							

Bits	描述
[31:30]	<p><b>PRI_11</b></p> <p>系统处理器的优先级11 – SVCall</p> <p>“0”表示最高优先级&amp; “3”表示最低优先级</p>

## 系统处理器优先级寄存器3 (SHPR3)

寄存器	偏移量	R/W	描述	复位后的值
SHPR3	SCS_BA + D20	R/W	系统处理器优先级寄存器3	0x 00000000

31	30	29	28	27	26	25	24
PRI_15		保留					
23	22	21	20	19	18	17	16
PRI_14		保留					
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留							

Bits	描述
[31:30]	<p><b>PRI_15</b></p> <p>系统处理器优先级15 – SysTick</p> <p>“0”表示最高优先级 &amp; “3”表示最低优先级</p>
[23:22]	<p><b>PRI_14</b></p> <p>系统处理器优先级14 – PendSV</p> <p>“0”表示最高优先级 &amp; “3”表示最低优先级</p>

## 6.3 时钟控制器

### 6.3.1 概述

时钟控制器为芯片提供时钟，包括系统时钟和所有外设时钟。时钟控制器利用独立的时钟开关控制位、时钟源选择位、4个时钟分频位，可以实现电源控制功能。当设置CPU掉电模式使能位(PWR\_DOWN\_EN)且CPU Cortex-M0执行WFI或WFE指令时，芯片将进入掉电模式。在掉电模式下，控制器关闭外部晶振和内部振荡器，以降低功耗到最低。

### 6.3.2 时钟发生器框图

时钟发生器由如下4个时钟源组成：

- 一个外部 4~24 MHz 晶振
- 一个内部 22.1184 MHz RC 振荡器
- 一个可编程的 PLL FOUT(PLL由外部4~24MHz晶振和内部22.1184MHz振荡器组成)
- 一个内部 10KHz 振荡器

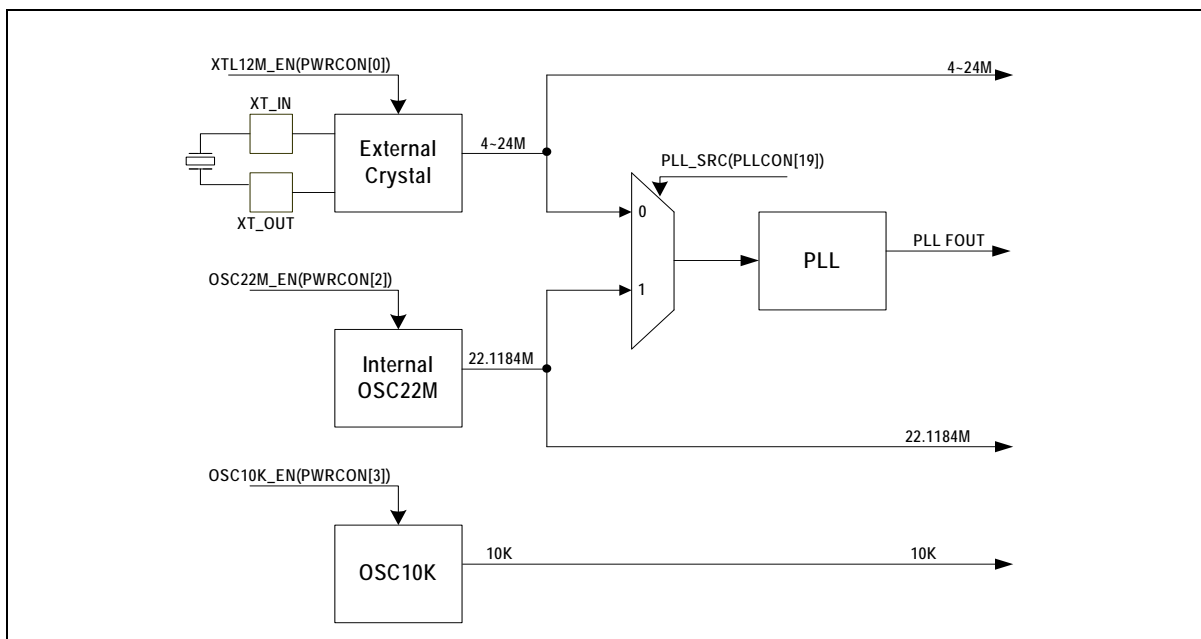


图 6.3-1 时钟发生器框图



### 6.3.3 系统时钟 & SysTick 时钟

系统时钟有4个时钟源，由时钟发生器产生。时钟源切换取决于寄存器HCLK\_S(CLKSEL0[2:0])，如下图所示。

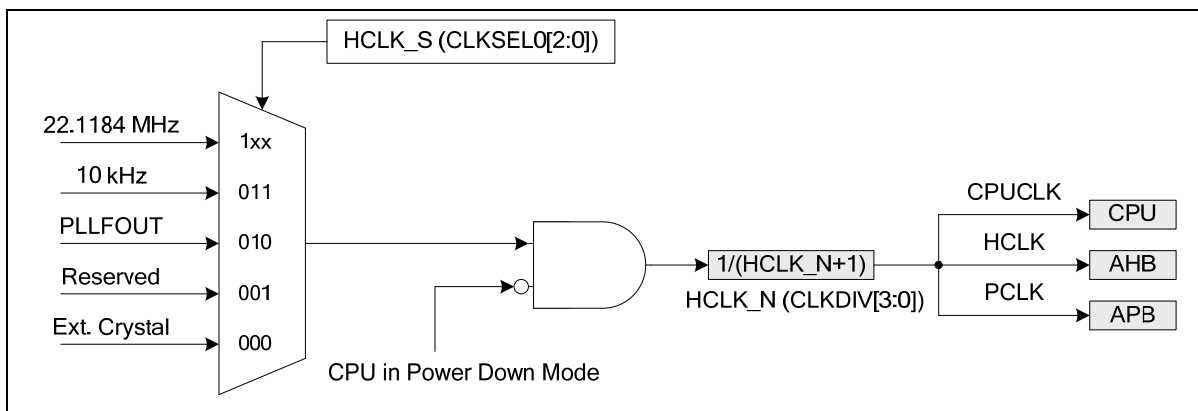


图 6.3-2 系统时钟框图

SysTick 时钟 (STCLK) 有4个时钟源，由时钟发生器产生。时钟源切换取决于寄存器STCLK\_S(CLKSEL0[5:3])，如下图所示。

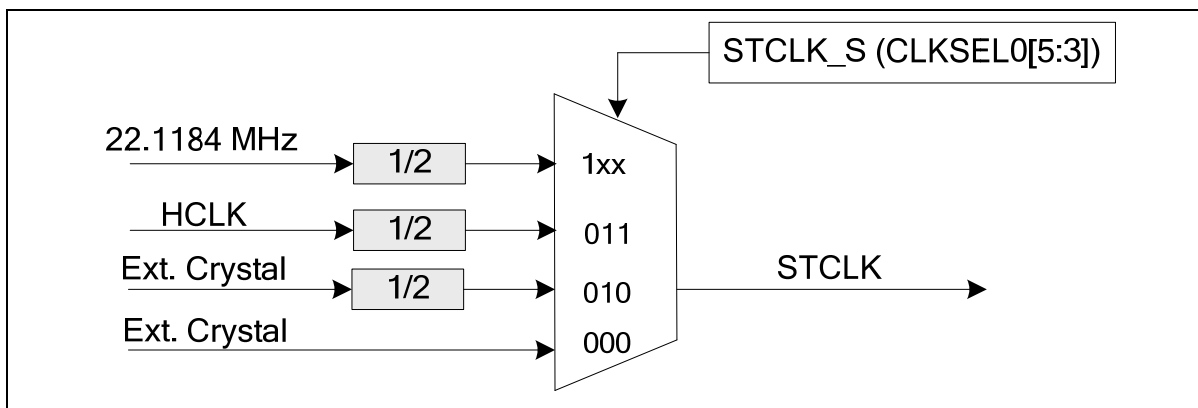


图 6.3-3 SysTick时钟控制框图

6.3.4 AHB 时钟源选择

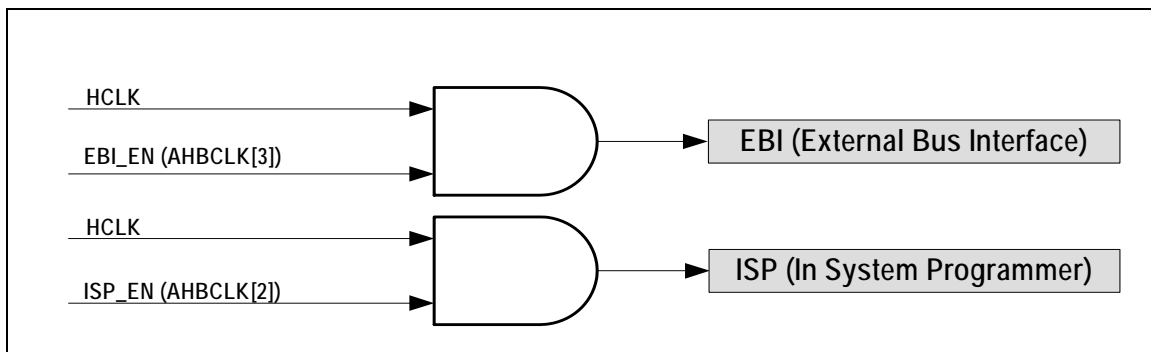


图 6.3-4 AHB 时钟源HCLK

## 6.3.5 外围设备时钟源选择

不同的外设，其时钟依赖于不同的时钟源切换. 参阅寄存器CLKSEL1 & APBCLK 的描述chapter 6.3.9.

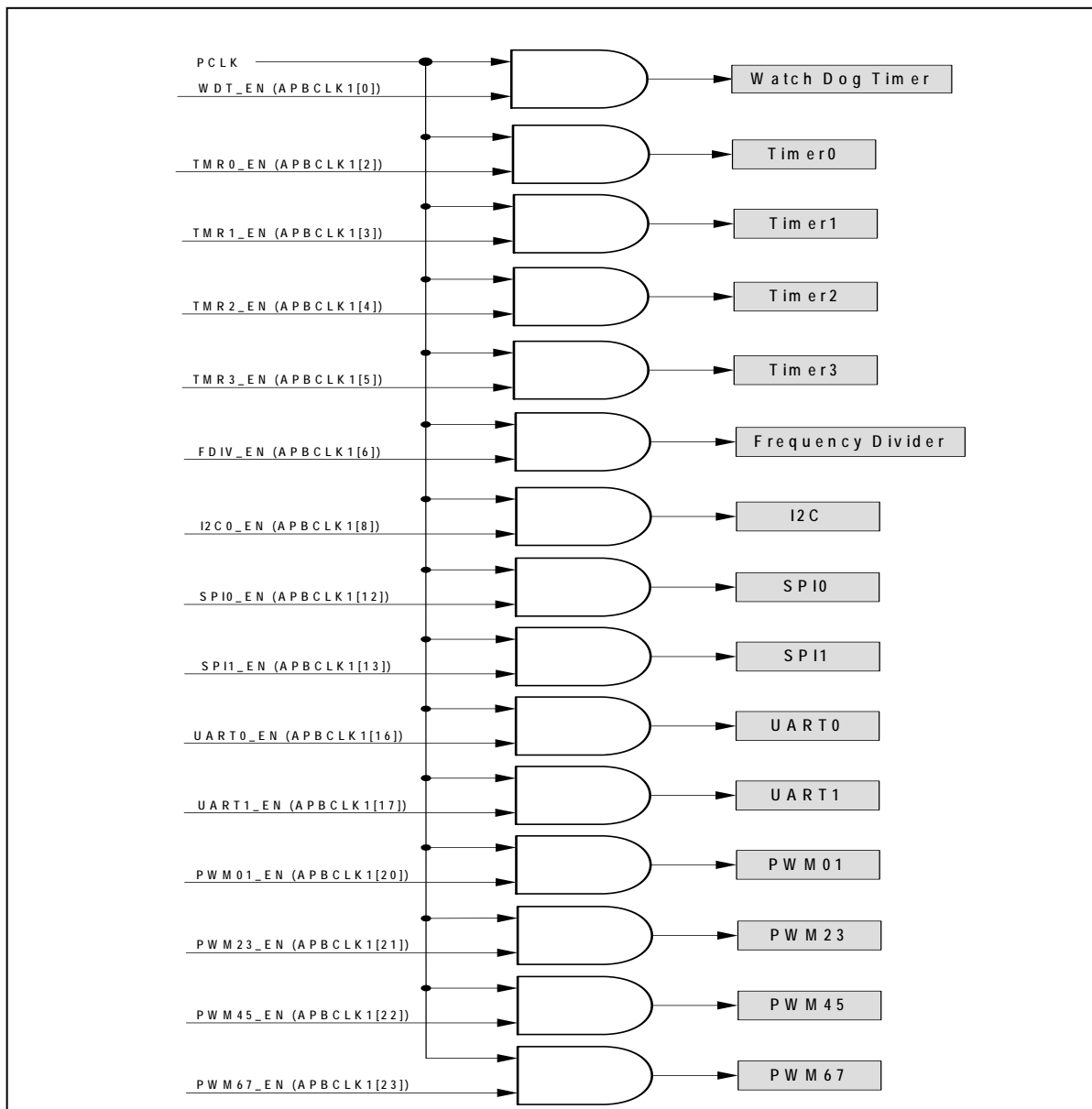


图 6.3-5 外设时钟源选择PCLK

## 6.3.6 掉电模式(深度休眠模式) 时钟

进入掉电模式后，大部分时钟源、外设时钟和系统时钟被禁用，也有一些时钟源与外设时钟仍在工作。

如下时钟仍在工作：

- 时钟发生器
  - 内部 10K 振荡器时钟
- 外设时钟 (当这些IP采用内部10KHz振荡器作时钟源时)
  - 看门狗时钟
  - Timer 0/1/2/3 时钟
  - PWM 时钟

### 6.3.7 分频器输出

该器件包含分频器，由16级2分频移位寄存器组成。16级2分频寄存器的输出由一个16到1的多路复用器映射到P3.6。有16种分频选择，从  $F_{in}/2^1$  到  $F_{in}/2^{17}$ ，其中  $F_{in}$  为输入到时钟分频器的时钟频率

输出公式： $F_{out} = F_{in}/2^{(N+1)}$ ，其中  $F_{in}$  为输入时钟频率， $F_{out}$  为时钟分频输出频率， $N$  为 FSEL(FRQDIV[3:0])中的4位值

当 FDIV\_EN(FRQDIV[4]) 被设为高，链计数器开始计数，当FDIV\_EN(FRQDIV[4])置0时，链计数器持续计数直到分频时钟达到低状态并停留在低状态。

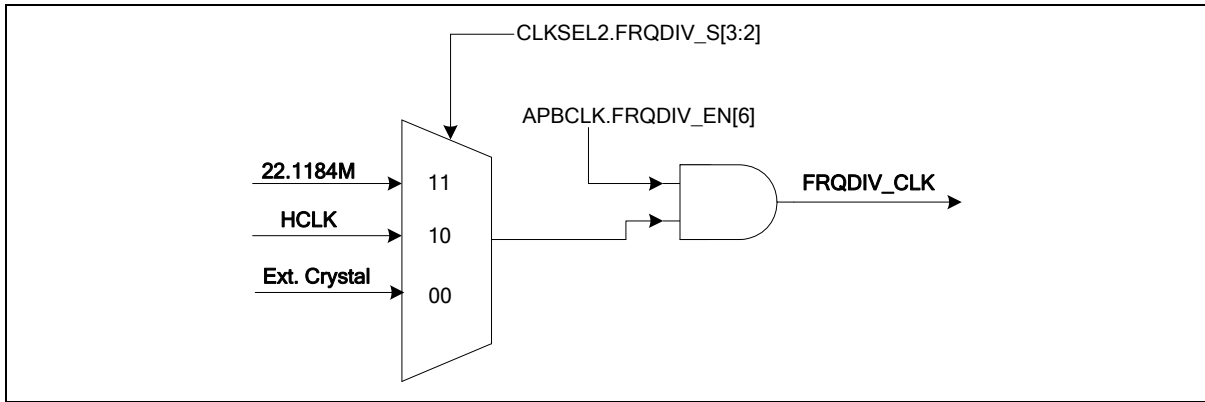


图 6.3-6分频器的时钟源

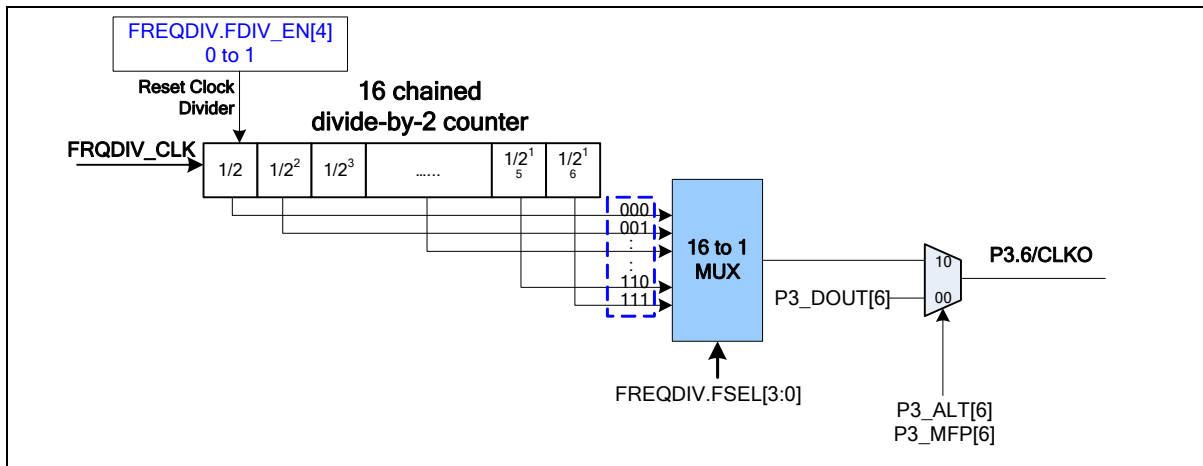


图 6.3-7分频器框图

### 6.3.8 时钟控制寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
PWRCON	CLK_BA + 00	R/W	系统掉电控制寄存器	0x0000_000X
AHBCLK	CLK_BA + 04	R/W	AHB 设备时钟使能控制寄存器	0x0000_0001
APBCLK	CLK_BA + 08	R/W	APB 设备时钟使能控制寄存器	0x0000_000x
CLKSTATUS	CLK_BA + 0C	R/W	时钟状态监控寄存器	0x0000_00XX
CLKSEL0	CLK_BA + 10	R/W	时钟源选择控制寄存器0	0xFFFF_FFFX
CLKSEL1	CLK_BA + 14	R/W	时钟源选择控制寄存器1	0xFFFF_FFFX
CLKSEL2	CLK_BA + 1C	R/W	时钟源选择控制寄存器2	0xFFFF_FFFX
CLKDIV	CLK_BA_ + 18	R/W	时钟分频数目寄存器	0x0000_0000
PLLCON	CLK_BA + 20	R/W	PLL控制寄存器	0x0005_C22E
FRQDIV	CLK_BA + 24	R/W	分频器控制寄存器	0x0000_0000

## 6.3.9 时钟控制寄存器描述

### 掉电控制寄存器 (PWRCON)

除BIT[6]外, PWRCON的其他位都受保护。解锁这些位, 需要向0x5000\_0100依次写入“59h”, “16h”, “88h”. 参考寄存器REGWRPROT, 其地址是GCR\_BA + 0x100

寄存器	偏移量	R/W	描述	复位后的值
PWRCON	CLK_BA + 00	R/W	系统掉电控制寄存器	0x0000_000X

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							PD_WAIT_CPU
7	6	5	4	3	2	1	0
PWR_DOWN_EN	PD_WU_STS	PD_WU_INT_EN	PD_WU_DLY	OSC10K_EN	OSC22M_EN	保留	XTL12M_EN

Bits	描述	
[31:9]	保留	保留
[8]	PD_WAIT_CPU	<p><b>控制进入掉电模式的条件</b></p> <p>1 = 在PWR_DOWN_EN置1与CPU执行WFE/WFI指令时, 芯片进入掉电模式。 0 = PWR_DOWN_EN 置1时, 芯片进入掉电模式</p>
[7]	PWR_DOWN_EN	<p><b>使能系统掉电模式</b></p> <p>该位置“1”, 使能芯片的掉电模式, 激活芯片的掉电行为取决于PD_WAIT_CPU位</p> <p>(a) PD_WAIT_CPU 为“0”, 置位PWR_DOWN_EN 后, 芯片进入掉电模式。 (b) PD_WAIT_CPU 为“1”, 在CPU的休眠模式有效时, 芯片仍在运行, 然后才进入掉电模式</p> <p>芯片由掉电模式唤醒, 该位自动清零, 在下次掉电时, 用户需要重新置位该位。</p> <p>掉电 模式下, 外部晶振(4~24MHz)与22.1184 MHz振荡器被禁用, 10KH振荡器的使能不受该位控制</p> <p>掉电时, PLL 与系统时钟也被禁用, 时钟源选择被忽略. 如果外设以10KHz振荡器作为时钟, 则其时钟不受该位控制</p> <p>1 = 芯片立即进入掉电模式 或等待CPU休眠命令WFI</p>

		0 = 芯片工作于正常模式或CPU进入空闲模式 (休眠模式)
[6]	PD_WU_STS	<b>芯片掉电唤醒状态标志</b> 若“掉电唤醒”置位, 表明芯片从掉电模式恢复 如果GPIO(P0~P4), 和 UART 唤醒, 该标志置位 写1清零.
[5]	PD_WU_INT_EN	<b>掉电模式唤醒的中断使能</b> 0 = 禁用 1 = 使能。从掉电唤醒时, 产生中断.
[4]	PD_WU_DLY	<b>唤醒延迟计数器使能.</b> 当芯片从掉电模式唤醒时, 该时钟控制将延迟一定时钟周期以等待系统时钟稳定. 当芯片工作于外部晶振 (4~24MHz), 延迟时间为4096 个时钟周期, 工作于22.1184MHZ时, 延迟256 个时钟周期。 1 = 使能时钟周期延迟 0 = 禁用时钟周期延迟
[3]	OSC10K_EN	<b>内部 10KHz 振荡器控制</b> 1 = 使能10KHz 振荡器 0 =禁用10KHz 振荡器
[2]	OSC22M_EN	<b>内部 22.1184MHz振荡器控制</b> 1 = 使能22.1184MHz振荡器 0 =禁用22.1184MHz振荡器
[1]	保留	保留
[0]	XTL12M_EN	<b>外部 12MHz晶振控制</b> 该位的默认值由flash控制器用户配置寄存器config0 [26:24] 设置。当默认时钟源为外部晶振 (4~24MHz). 该位自动置1 1 = 使能晶振 0 =禁用晶振



寄存器 指令模式	PWR_DOWN_EN	PD_WAIT_CPU	CPU 运行 WFE/WFI指令	时钟门控
正常运行模式	1'b0	1'b0	NO	所有时钟根据寄存器配置正常运行
IDLE 模式 (CPU 进入休眠模式)	1'b0	1'b0	YES	仅CPU内部时钟运行
Power_down 模式	1'b1	1'b0	NO	大部分时钟停止运行, 仅外部10K与WDT/Timer/PWM/ADC 可能运行.
Power_down Mode (CPU 进入深度休眠模式)	1'b1	1'b1	YES	大部分时钟停止运行, 仅外部10K与WDT/Timer/PWM/ADC 可能运行

表 6.3-1 掉电模式控制表

**AHB设备时钟使能控制寄存器 (AHBCLK)**

该寄存器各位用于使能/禁用系统与AHB设备时钟禁用

寄存器	偏移量	R/W	描述	复位后的值
AHBCLK	CLK_BA + 04	R/W	AHB设备时钟使能控制寄存器	0x0000_0001

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

Bits	描述	
[31:4]	保留	保留
[3]	EBI_EN	<b>EBI 控制器时钟使能控制.</b> 1 = 使能EBI控制器时钟. 0 = 禁用EBI控制器时钟.
[2]	ISP_EN	<b>Flash ISP 控制器时钟使能控制.</b> 1 = 使能 the Flash ISP控制器时钟. 0 = 禁用the Flash ISP控制器时钟.
[1:0]	保留	保留

## APB 设备时钟使能控制寄存器 (APBCLK)

该寄存器A的各位用于使能/禁用APB设备和外设时钟.

寄存器	偏移量	R/W	描述	复位后的值
APBCLK	CLK_BA + 08	R/W	APB设备时钟使能控制寄存器	0x0000_000X

31	30	29	28	27	26	25	24
保留			ADC_EN	保留			
23	22	21	20	19	18	17	16
PWM67_EN	PWM45_EN	PWM23_EN	PWM01_EN	保留		UART1_EN	UART0_EN
15	14	13	12	11	10	9	8
保留		SPI1_EN	SPI0_EN	保留			I2C_EN
7	6	5	4	3	2	1	0
保留	FDIV_EN	TMR3_EN	TMR2_EN	TMR1_EN	TMR0_EN	保留	WDT_EN

Bits	描述	
[31:29]	保留	保留
[28]	ADC_EN	<b>使能ADC时钟控制</b> 1 = 使能 ADC 时钟 0 = 禁用 ADC 时钟
[27:24]	保留	保留
[23]	PWM67_EN	<b>PWM_67 时钟使能</b> 1 = 使能 PWM67 时钟 0 = 禁用 PWM67 时钟
[22]	PWM45_EN	<b>PWM_45时钟使能</b> 1 = 使能 PWM45 时钟 0 = 禁用 PWM45 时钟
[21]	PWM23_EN	<b>PWM_23时钟使能</b> 1 = 使能 PWM23 时钟 0 = 禁用 PWM23 时钟

[20]	<b>PWM01_EN</b>	<b>PWM_01时钟使能</b> 1 = 使能 PWM01时钟 0 = 禁用 PWM01 时钟
[19:18]	保留	保留
[17]	<b>UART1_EN</b>	<b>UART1 时钟使能</b> 1 = 使能 UART1 时钟 0 = 禁用 UART1 时钟
[16]	<b>UART0_EN</b>	<b>UART0 时钟使能</b> 1 = 使能 UART0 时钟 0 = 禁用 UART0 时钟
[15:14]	保留	保留
[13]	<b>SPI1_EN</b>	<b>SPI1 时钟使能</b> 1 = 使能 SPI1 时钟 0 = 禁用 SPI1 时钟
[12]	<b>SPI0_EN</b>	<b>SPI0 时钟使能</b> 1 = 使能 SPI0 时钟 0 = 禁用 SPI0 时钟
[11:9]	保留	保留
[8]	<b>I2C_EN</b>	<b>I2C 时钟使能</b> 1 = 使能 I2C 时钟 0 = 禁用 I2C 时钟
[7]	保留	保留
[6]	<b>FDIV_EN</b>	<b>分频器输出时钟使能控制</b> 0 = 禁用 1 = 使能
[5]	<b>TMR3_EN</b>	<b>Timer3 时钟使能控制</b> 0 = 禁用 1 = 使能

[4]	TMR2_EN	<b>Timer2 时钟使能控制</b> 0 = 禁用 1 = 使能
[3]	TMR1_EN	<b>Timer1 时钟使能控制</b> 0 = 禁用 1 = 使能
[2]	TMR0_EN	<b>Timer0 时钟使能控制</b> 0 = 禁用 1 = 使能
[1]	保留	保留
[0]	WDT_EN	<b>Watch Dog 时钟使能.</b> 该位是受保护的位，对该位编程时，需要向0x5000_0100 依次写入“59h”，“16h”，“88h”来解除寄存器写保护，参考寄存器REGWRPROT 默认值由flash控制器用户可配置寄存器congig0 bit[31]设置 0 = 禁用 1 = 使能

**时钟状态寄存器 (CLKSTATUS)**

该寄存器各位用于监控芯片时钟是否稳定，时钟切换是否失败。

寄存器	偏移量	R/W	描述	复位后的值
CLKSTATUS	CLK_BA + 0C	R/W	时钟状态监控寄存器	0x0000_00XX

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
CLK_SW_FAIL	保留		OSC22M_STB	OSC10K_STB	PLL_STB	保留	XTL12M_STB

Bits	描述	
[31:8]	保留	-
[7]	CLK_SW_FAIL	<p>时钟切换失败标志</p> <p>1 = 时钟切换失败</p> <p>0 = 时钟切换成功</p> <p>当目标切换时钟源不稳定时，该位置位写1清零。</p>
[6:5]	保留	-
[4]	OSC22M_STB	<p>OSC22M (内部 22.1184 MHz)时钟源稳定标志 (只读)</p> <p>1 = OSC22M 时钟稳定</p> <p>0 = OSC22M 时钟不稳定或没有使能</p>
[3]	OSC10K_STB	<p>OSC10K 时钟源稳定标志 (只读)</p> <p>1 = OSC10K 时钟稳定</p> <p>0 = OSC10K 时钟不稳定或没有使能</p>

[2]	PLL_STB	<b>PLL 时钟源稳定标志（只读）</b> 1 = PLL 时钟稳定 0 = PLL 时钟不稳定或没有使能
[1]	保留	-
[0]	XTL12M_STB	<b>XTL12M 时钟源稳定标志（只读）</b> 1 = XTL12M 时钟稳定 0 = XTL12M 时钟不稳定或没有使能

## 时钟源选择控制寄存器0 (CLKSELO)

寄存器	偏移量	R/W	描述	复位后的值
CLKSELO <sup>[1]</sup>	CLK_BA + 10	R/W	时钟源选择控制寄存器0	0xFFFF_FFFX <sup>[2]</sup>

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

Bits	描述	
[31:6]	保留	保留
[5:3]	STCLK_S	<p><b>MCU Cortex_M0 SysTick 时钟源选择.</b></p> <p>受保护位, 对该位编程时, 需要向0x5000_0100 依次写入“59h”, “16h”, “88h” 来解除寄存器写保护, 参考寄存器REGWRPROT, 地址为GCR_BA + 0x100</p> <p>000 = 外部晶振 (4~24MHz)</p> <p>001 = 保留</p> <p>010 = 外部晶振 (4~24MHz) 1/2分频</p> <p>011 = HCLK/2</p> <p>1xx = 内部 22.1184MHz 振荡器 1/2分频.</p>



[2:0]	HCLK_S	<p><b>HCLK 时钟源选择.</b></p> <p>注:</p> <ol style="list-style-type: none"> <li>1. 在时钟切换到相关时钟源(当前和新选)时必须打开</li> <li>2. 任何复位后, 加载用户配置寄存器<b>CFOSC(Config0[26:24])</b>的值, 默认值可为 000b 或 111b.</li> <li>3. 受保护位, 对该位编程时, 需要向0x5000_0100 依次写入“59h”, “16h”, “88h”来解除寄存器写保护, 参考寄存器REGWRPROT, 地址为GCR_BA + 0x100</li> </ol> <p>00 =外部晶振 (4~24MHz)          001 = 保留          010 = PLL 时钟          011 = 内部 10KHz 振荡器时钟          111 = 内部 22.1184MHz振荡器时钟</p> <p>Others = 保留.</p>
-------	--------	--

## 时钟源选择控制寄存器1 (CLKSEL1)

在时钟切换前，相关的时钟源（当前和新选），必须打开。

寄存器	偏移量	R/W	描述	复位后的值
CLKSEL1	CLK_BA + 14	R/W	时钟源选择控制寄存器1	0xFFFF_FFFX

31	30	29	28	27	26	25	24
PWM23_S		PWM01_S		保留		UART_S	
23	22	21	20	19	18	17	16
保留		TMR3_S		保留		TMR2_S	
15	14	13	12	11	10	9	8
保留		TMR1_S		保留		TMR0_S	
7	6	5	4	3	2	1	0
保留				ADC_S		WDT_S	

Bits	描述	
[31:30]	PWM23_S	<b>PWM3 与 PWM2的时钟源选择.</b> PWM3 与 PWM2使用相同的时钟源 和相同的分频 00 = 外部晶振 (4~24MHz) 01 = 保留 10 = HCLK 11 = 内部 22.1184MHz 振荡器
[29:28]	PWM01_S	<b>PWM1 与 PWM0的时钟源选择.</b> PWM1 与 PWM0使用相同的时钟源 和相同的分频 00 = 外部晶振 (4~24MHz) 01 = 保留 10 = HCLK 11 = 内部22.1184MHz 振荡器
[27:26]	保留	保留

[25:24]	<b>UART_S</b>	<b>UART时钟源选择.</b> 00 =外部晶振 (4~24MHz) 01 = PLL 1x =内部22.118422MHz 振荡器
[23]	保留	保留
[22:20]	<b>TMR3_S</b>	<b>TIMER3 时钟源选择.</b> 000 =外部晶振 (4~24MHz) 001 =保留 010 = HCLK 011 = 外部触发时钟 1xx =内部 22.1184MHz 振荡器
[19]	保留	保留
[18:16]	<b>TMR2_S</b>	<b>TIMER2时钟源选择.</b> 000 =外部晶振 (4~24MHz) 001 =保留 010 = HCLK 011 = 外部触发时钟 1xx =内部 22.1184 MHz 振荡器
[15]	保留	保留
[14:12]	<b>TMR1_S</b>	<b>TIMER1 时钟源选择.</b> 000 =外部晶振 (4~24MHz) 001 =保留 010 = HCLK 011 = 外部触发时钟 1xx =内部22.1184MHz 振荡器
[11]	保留	保留
[10:8]	<b>TMR0_S</b>	<b>TIMERO 时钟源选择.</b> 000 =外部晶振 (4~24MHz) 001 =保留 010 = HCLK 011 = 外部触发时钟 1xx =内部 22.1184MHz 振荡器
[7:4]	保留	保留

[3:2]	<b>ADC_S</b>	<b>ADC 时钟源选择.</b> 00 =外部晶振 (4~24MHz) 01 = PLL 1x =内部 122.1184 MHz 振荡器
[1:0]	<b>WDT_S</b>	<b>WDG CLK 时钟源选择.</b> 受保护位, 对该位编程时, 需要向0x5000_0100 依次写入“59h”, “16h”, “88h” 来解除寄存器写保护, 参考寄存器REGWRPROT, 地址为GCR_BA + 0x100 00 =外部晶振 (4~24MHz) 01 =保留 10 = HCLK/2048时钟 11 = 内部 10KHz振荡器时钟

## 时钟源选择控制寄存器 (CLKSEL2)

在时钟切换前, 相关的时钟源(当前和新选的), 必须打开.

寄存器	偏移量	R/W	描述	复位后的值
CLKSEL2	CLK_BA + 1C	R/W	时钟源选择控制寄存器 2	0xFFFF_FFFX

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
PWM67_S		PWM45_S		FRQDIV_S		保留	

Bits	描述	
[31:8]	保留	保留
[7:6]	PWM67_S	<b>PWM6 与 PWM7的时钟源选择.</b> – PWM6 与 PWM7使用相同的时钟源 和相同的分频器 00 = 外部晶振 (4~24MHz) 01 = 保留 10 = HCLK 11 = 内部 22.1184MHz 振荡器
[5:4]	PWM45_S	<b>PWM4 与 PWM5的时钟源选择.</b> – PWM4 与 PWM5使用相同的时钟源 和相同的分频器 00 = 外部晶振 (4~24MHz) 01 = 保留 10 = HCLK 11 = 内部22.1184MHz 振荡器

[3:2]	FRQDIV_S	时钟分频器时钟源选择 00 = 外部晶振 (4~24MHz) 01 = 保留 10 = HCLK 11 = 内部 22.1184MHz 振荡器
[1:0]	保留	保留

## 时钟分频寄存器(CLKDIV)

寄存器	偏移量	R/W	描述	复位后的值
CLKDIV	CLK_BA_ + 18	R/W	时钟分频寄存器	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:16]	ADC_N	ADC时钟频率=ADC时钟源频率/ (ADC_N + 1)
[15:12]	保留	保留
[11:8]	UART_N	UART时钟频率 = (UART时钟源频率) / (UART_N + 1)
[7:4]	保留	保留
[3:0]	HCLK_N	HCLK 时钟频率 = (HCLK时钟源频率) / (HCLK_N + 1)

### PLL控制寄存器 (PLLCON)

The PLL的参考时钟源来自外部晶振（4~24MHz）的外部时钟输入或22.1184MHz的内部振荡器，该寄存器用于控制PLL的输出频率和PLL的操作模式

寄存器	偏移量	R/W	描述	复位后的值
PLLCON	CLK_BA + 20	R/W	PLL控制寄存器	0x0005_C22E

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

Bits	描述	
[19]	PLL_SRC	<b>PLL时钟源选择</b> 1 = PLL 时钟源为22.1184 MHz 振荡器 0 = PLL时钟源为外部晶振（4~24MHz）
[18]	OE	<b>PLL OE (FOUT enable)引脚控制</b> 0 = 使能 PLL FOUT 1 = PLL FOUT 为低
[17]	BP	<b>PLL 旁路控制</b> 0 = PLL 正常模式 (默认) 1 = PLL 时钟输出与时钟输入相同(XTALin)
[16]	PD	<b>掉电模式.</b> 设置PWRCON的IDLE位为"1", PLL进入掉电模式 0 = PLL正常模式 (默认) 1 = PLL 掉电模式
[15:14]	OUT_DV	PLL 输出分频控制引脚 (PLL_OD[1:0])



[13:9]	<b>IN_DV</b>	PLL输入分频控制引脚(PLL_R[4:0])
[8:0]	<b>FB_DV</b>	PLL反馈分频控制引脚(PLL_F[8:0])

## 输出时钟频率设置

$$F_{OUT} = F_{IN} \times \frac{NF}{NR} \times \frac{1}{NO}$$

约束条件:

1. 3.2MHz < FIN < 150MHz
2. 800KHz <  $\frac{FIN}{2 * NR}$  < 8MHz
3. 800KHz <  $\frac{FIN}{2 * NR}$  < 8MHz, 120MHz < FCO is preferred.

符号	说明
<i>F<sub>OUT</sub></i>	输出时钟频率
<i>F<sub>IN</sub></i>	输入 (参考) 时钟频率
<i>N<sub>R</sub></i>	输入分频 (IN_DV + 2)
<i>N<sub>F</sub></i>	反馈分频 (FB_DV + 2)
<i>N<sub>O</sub></i>	OUT_DV = "00" : NO = 1 OUT_DV = "01" : NO = 2 OUT_DV = "10" : NO = 2 OUT_DV = "11" : NO = 4

## 默认频率设置

PLLCON默认值: 0xC22E

FIN = 12 MHz

NR = (1+2) = 3

NF = (46+2) = 48

NO = 4

F<sub>OUT</sub> = 12/4 x 48 x 1/3 = 48 MHz

## 频率分频器控制寄存器(FRQDIV)

寄存器	偏移量	R/W	描述	复位后的值
FRQDIV	CLK_BA+ 24	R/W	频率分频器控制寄存器	0x0000_0000

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

Bits	描述	
[31:5]	保留	保留
[4]	FDIV_EN	频率分频器使能位 0 = 禁用频率分频 1 = 使能频率分频
[3:0]	FSEL	分频器输出频率选择位 输出频率的公式是 $F_{out} = F_{in}/2^{(N+1)},$ F <sub>in</sub> 为输入时钟频率，F <sub>out</sub> 为分频器输出时钟频率，N 为FSEL[3:0]的值。

## 6.4 通用I/O

### 6.4.1 概述

这款MCU有40个通用I/O引脚，并复用为特殊功能引脚。40个引脚分配在P0, P1, P2, P3 和 P4五个端口上，每个端口最多8个引脚。每个引脚都是独立的，都有相应的寄存器来控制引脚工作模式与数据

I/O引脚上的I/O类型可由软件独立地配置为输入，输出，开漏或准双端模式。所有的I/O引脚处于准双端模式，端口数据寄存器Px\_DOUT[7:0]的值为0x000\_00FF。每个I/O引脚配置为弱上拉，即用一颗110KΩ~300KΩ的上拉电阻接到V<sub>DD</sub>（5.0V 到 2.5V）上。

#### 6.4.1.1 输入模式的说明

设置 Px\_PMD(PMDn[1:0]) 为00b，Px[n]为输入模式，I/O引脚为三态（高阻态），没有输出驱动能力。Px\_PIN的值反映相应端口的状态。

#### 6.4.1.2 输出模式的说明

设置Px\_PMD(PMDn[1:0])为 2'b01，Px[n]为输出模式，I/O支持数字输出功能，有拉电流/灌电流能力。Px\_DOUT 相应位的值被送到相应引脚上。

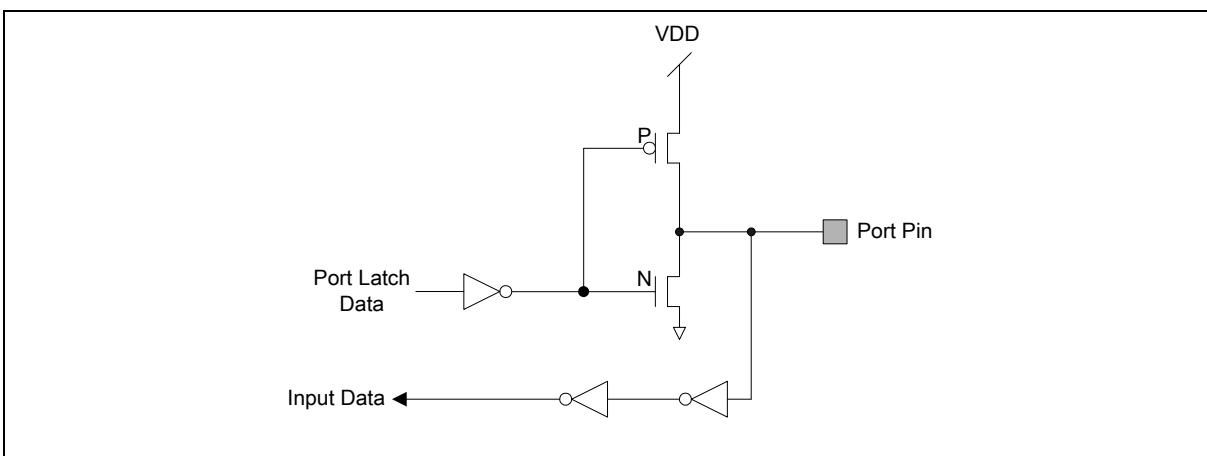


图 6.4-1 推挽输出

### 6.4.1.3 开漏模式的说明

设置 Px\_PMD(PMDn[1:0])为 2'b10, Px[n]为开漏模式, I/O支持数字输出功能, 仅有灌电流能力, 用于驱动到高状态外接上一颗上拉电阻. 如果Px\_DOUT相应位bit [n]的值为“0”, 引脚上输出低电平. 如果Px\_DOUT 相应位bit [n]的值为“1”, 该引脚输出为高电平, 可以由内部上拉电阻或外部上拉电阻控制。

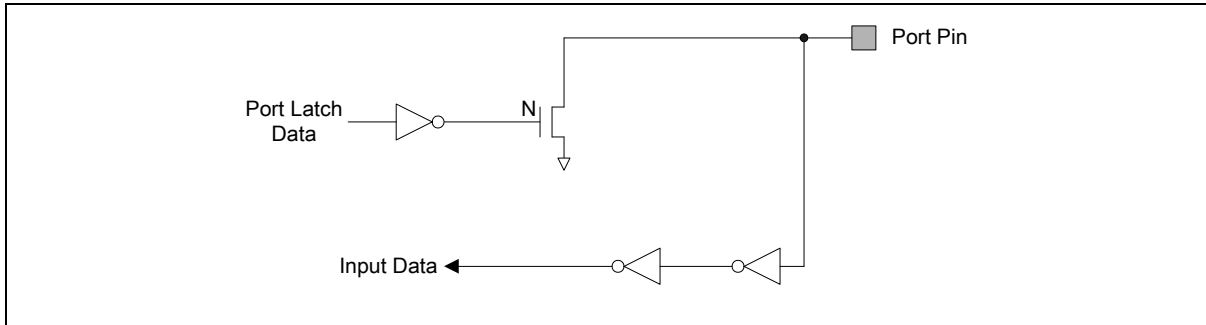


图 6.4-2 开漏输出

### 6.4.1.4 准双端模式的说明

设置Px\_PMD(PMDn[1:0]) 为 2'b11, Px[n] 为准双端模式, I/O同时支持数字输出和输入功能, 但拉电流仅达数百uA. 要实现数字输入, 需要先将Px\_DOUT 相应位置1. 准双端输出是80C51及其派生产品所共有的模式. 若Px\_DOUT相应位bit[n]为“0”, 引脚上输出为“低电平”. 若Px\_DOUT相应位bit[n]为“1”, 该引脚将核对引脚值. 若引脚值为高, 没有任何动作, 若引脚值为低, 该引脚在2个时钟周期内置高, 然后禁用强输入驱动, 引脚状态由内部上拉电阻控制. 注: 准双端模式的拉电流能力仅有200uA到30uA(相应VDD的电压从5.0V到2.5V)

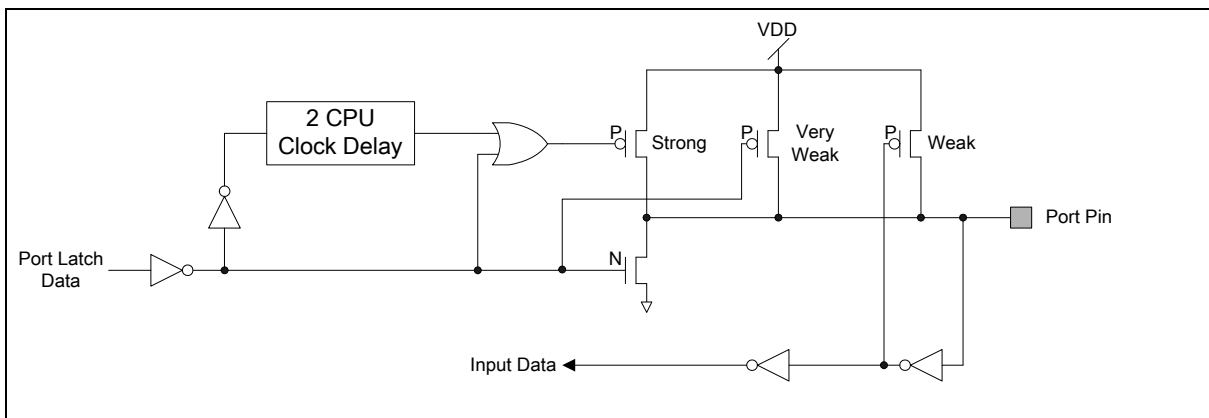


图 6.4-3 准双端 I/O 模式

## 6.4.2 Port 0-4 控制器寄存器映射映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>GP_BA = 0x5000_4000</b>				
P0_PMD	GP_BA+0x000	R/W	P0 Bit模式控制	0x0000_FFFF
P0_OFFD	GP_BA+0x004	R/W	P0 Bit OFF数字使能	0x0000_0000
P0_DOUT	GP_BA+0x008	R/W	P0数据输出值	0x0000_00FF
P0_DMASK	GP_BA+0x00C	R/W	P0数据输出写屏蔽	0x0000_0000
P0_PIN	GP_BA+0x010	R	P0管脚数值	0x0000_00XX
P0_DBEN	GP_BA+0x014	R/W	P0防反弹使能	0x0000_0000
P0_IMD	GP_BA+0x018	R/W	P0中断模式控制	0x0000_0000
P0_IEN	GP_BA+0x01C	R/W	P0中断使能	0x0000_0000
P0_ISRC	GP_BA+0x020	R/WC	P0 中断源标志	0xXXXX_XXXX
P1_PMD	GP_BA+0x040	R/W	P1 Bit模式使能	0x0000_FFFF
P1_OFFD	GP_BA+0x044	R/W	P1 Bit OFF数字使能	0x0000_0000
P1_DOUT	GP_BA+0x048	R/W	P1数据输出值	0x0000_00FF
P1_DMASK	GP_BA+0x04C	R/W	P1 数据输出写屏蔽	0x0000_0000
P1_PIN	GP_BA+0x050	R	P1管脚数值	0x0000_00XX
P1_DBEN	GP_BA+0x054	R/W	P1防反弹使能	0x0000_0000
P1_IMD	GP_BA+0x058	R/W	P1 中断模式控制	0x0000_0000
P1_IEN	GP_BA+0x05C	R/W	P1 中断使能	0x0000_0000
P1_ISRC	GP_BA+0x060	R/WC	P1 中断源标志	0xXXXX_XXXX
P2_PMD	GP_BA+0x080	R/W	P2 Bit 模式使能	0x0000_FFFF
P2_OFFD	GP_BA+0x084	R/W	P2 Bit OFF数字使能	0x0000_0000
P2_DOUT	GP_BA+0x088	R/W	P2数据输出值	0x0000_00FF
P2_DMASK	GP_BA+0x08C	R/W	P2数据输出写屏蔽	0x0000_0000
P2_PIN	GP_BA+0x090	R	P2 管脚数值	0x0000_00XX

P2_DBEN	GP_BA+0x094	R/W	P2防反弹使能	0x0000_0000
P2_IMD	GP_BA+0x098	R/W	P2中断模式控制	0x0000_0000
P2_IEN	GP_BA+0x09C	R/W	P2中断使能	0x0000_0000
P2_ISRC	GP_BA+0x0A0	R/WC	P2中断源标志	0xXXXX_XXXX
P3_PMD	GP_BA+0x0C0	R/W	P3 Bit模式使能	0x0000_FFFF
P3_OFFD	GP_BA+0x0C4	R/W	P3 Bit OFF数字使能	0x0000_0000
P3_DOUT	GP_BA+0x0C8	R/W	P3数据输出值	0x0000_00FF
P3_DMASK	GP_BA+0x0CC	R/W	P3 数据输出写屏蔽	0x0000_0000
P3_PIN	GP_BA+0x0D0	R	P3管脚数值	0x0000_00XX
P3_DBEN	GP_BA+0x0D4	R/W	P3防反弹使能	0x0000_0000
P3_IMD	GP_BA+0x0D8	R/W	P3中断模式控制	0x0000_0000
P3_IEN	GP_BA+0x0DC	R/W	P3中断使能	0x0000_0000
P3_ISRC	GP_BA+0x0E0	R/WC	P3中断源标志	0xXXXX_XXXX
P4_PMD	GP_BA+0x100	R/W	P4 Bit模式使能	0x0000_FFFF
P4_OFFD	GP_BA+0x104	R/W	P4 Bit OFF数字使能	0x0000_0000
P4_DOUT	GP_BA+0x108	R/W	P4数据输出值	0x0000_00FF
P4_DMASK	GP_BA+0x10C	R/W	P4 数据输出写屏蔽	0x0000_0000
P4_PIN	GP_BA+0x110	R	P4 E管脚数值	0x0000_00XX
P4_DBEN	GP_BA+0x114	R/W	P4 防反弹使能	0x0000_0000
P4_IMD	GP_BA+0x118	R/W	P4 中断模式控制	0x0000_0000
P4_IEN	GP_BA+0x11C	R/W	P4 中断使能	0x0000_0000
P4_ISRC	GP_BA+0x120	R/WC	P4 中断源标志	0xXXXX_XXXX
DBNCECON	GP_BA+0x180	R/W	防反弹周期控制	0x0000_0020
P00_DOUT	GP_BA+0x200	R/W	P0.0数据输出值	0x0000_0001
P01_DOUT	GP_BA+0x204	R/W	P0.1数据输出值	0x0000_0001
P02_DOUT	GP_BA+0x208	R/W	P0.2数据输出值	0x0000_0001
P03_DOUT	GP_BA+0x20C	R/W	P0.3数据输出值	0x0000_0001
P04_DOUT	GP_BA+0x210	R/W	P0.4数据输出值	0x0000_0001

P05_DOUT	GP_BA+0x214	R/W	P0.5数据输出值	0x0000_0001
P06_DOUT	GP_BA+0x218	R/W	P0.6数据输出值	0x0000_0001
P07_DOUT	GP_BA+0x21C	R/W	P0.7数据输出值	0x0000_0001
P10_DOUT	GP_BA+0x220	R/W	P1.0数据输出值	0x0000_0001
P11_DOUT	GP_BA+0x224	R/W	P1.1数据输出值	0x0000_0001
P12_DOUT	GP_BA+0x228	R/W	P1.2数据输出值	0x0000_0001
P13_DOUT	GP_BA+0x22C	R/W	P1.3数据输出值	0x0000_0001
P14_DOUT	GP_BA+0x230	R/W	P1.4数据输出值	0x0000_0001
P15_DOUT	GP_BA+0x234	R/W	P1.5数据输出值	0x0000_0001
P16_DOUT	GP_BA+0x238	R/W	P1.6数据输出值	0x0000_0001
P17_DOUT	GP_BA+0x23C	R/W	P1.7数据输出值	0x0000_0001
P20_DOUT	GP_BA+0x240	R/W	P2.0数据输出值	0x0000_0001
P21_DOUT	GP_BA+0x244	R/W	P2.1数据输出值	0x0000_0001
P22_DOUT	GP_BA+0x248	R/W	P2.2数据输出值	0x0000_0001
P23_DOUT	GP_BA+0x24C	R/W	P2.3数据输出值	0x0000_0001
P24_DOUT	GP_BA+0x250	R/W	P2.4数据输出值	0x0000_0001
P25_DOUT	GP_BA+0x254	R/W	P2.5数据输出值	0x0000_0001
P26_DOUT	GP_BA+0x258	R/W	P2.6数据输出值	0x0000_0001
P27_DOUT	GP_BA+0x25C	R/W	P2.7数据输出值	0x0000_0001
P30_DOUT	GP_BA+0x260	R/W	P3.0数据输出值	0x0000_0001
P31_DOUT	GP_BA+0x264	R/W	P3.1数据输出值	0x0000_0001
P32_DOUT	GP_BA+0x268	R/W	P3.2数据输出值	0x0000_0001
P33_DOUT	GP_BA+0x26C	R/W	P3.3数据输出值	0x0000_0001
P34_DOUT	GP_BA+0x270	R/W	P3.4数据输出值	0x0000_0001
P35_DOUT	GP_BA+0x274	R/W	P3.5数据输出值	0x0000_0001
P36_DOUT	GP_BA+0x278	R/W	P3.6数据输出值	0x0000_0001
P37_DOUT	GP_BA+0x27C	R/W	P3.7数据输出值	0x0000_0001
P40_DOUT	GP_BA+0x280	R/W	P4.0数据输出值	0x0000_0001



P41_DOUT	GP_BA+0x284	R/W	P4.1 数据输出值	0x0000_0001
P42_DOUT	GP_BA+0x288	R/W	P4.2 数据输出值	0x0000_0001
P43_DOUT	GP_BA+0x28C	R/W	P4.3 数据输出值	0x0000_0001
P44_DOUT	GP_BA+0x290	R/W	P4.4 数据输出值	0x0000_0001
P45_DOUT	GP_BA+0x294	R/W	P4.5 数据输出值	0x0000_0001
P46_DOUT	GP_BA+0x298	R/W	P4.6 数据输出值	0x0000_0001
P47_DOUT	GP_BA+0x29C	R/W	P4.7 数据输出值	0x0000_0001

## 6.4.3 Port 0-4 控制器寄存器描述

### Port 0-4 I/O 模式控制(Px\_PMD)

寄存器	偏移量	R/W	描述	复位后的值
P0_PMD	GP_BA+0x000	R/W	P0 Pin I/O 模式控制	0x0000_FFFF
P1_PMD	GP_BA+0x040	R/W	P1 Pin I/O 模式控制	0x0000_FFFF
P2_PMD	GP_BA+0x080	R/W	P2 Pin I/O 模式控制	0x0000_FFFF
P3_PMD	GP_BA+0x0C0	R/W	P3 Pin I/O 模式控制	0x0000_FFFF
P4_PMD	GP_BA+0x100	R/W	P4 Pin I/O 模式控制	0x0000_FFFF

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
PMD7		PMD6		PMD5		PMD4	
7	6	5	4	3	2	1	0
PMD3		PMD2		PMD1		PMD0	

Bits	描述	
[31:16]	保留	保留
[2n+1 :2n]	<b>PMDn</b>	<b>Px I/O Pin[n]</b> 模式控制 Px的I/O类型 00 = Px [n] 输入模式. 01 = Px [n] 输出模式. 10 = Px [n] 开漏模式. 11 = Px [n] 准双端模式 x=0~4, n = 0~7

### Port 0-4 Bit OFF 数字寄存器使能 (Px\_OFFD)

寄存器	偏移量	R/W	描述	复位后的值
P0_OFFD	GP_BA+0x004	R/W	P0 Pin OFF 数字使能	0x0000_0000
P1_OFFD	GP_BA+0x044	R/W	P1 Pin OFF 数字使能	0x0000_0000
P2_OFFD	GP_BA+0x084	R/W	P2 Pin OFF 数字使能	0x0000_0000
P3_OFFD	GP_BA+0x0C4	R/W	P3 Pin OFF 数字使能	0x0000_0000
P4_OFFD	GP_BA+0x104	R/W	P4 Pin OFF 数字使能	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:16]	OFFD	<b>OFFD: Px Pin[n] OFF 数字输入通道使能</b> 1 = 禁用 IO 数字输入通道 (数字输入连接到低电平) 0 = 使能 IO 数字输入通道 x=0~4, n = 0~7
[15:0]	保留	保留

### Port 0-4数据输出值(Px DOUT)

寄存器	偏移量	R/W	描述	复位后的值
P0_DOUT	GP_BA+0x008	R/W	P0数据输出值	0x0000_00FF
P1_DOUT	GP_BA+0x048	R/W	P1数据输出值	0x0000_00FF
P2_DOUT	GP_BA+0x088	R/W	P2数据输出值	0x0000_00FF
P3_DOUT	GP_BA+0x0C8	R/W	P3数据输出值	0x0000_00FF
P4_DOUT	GP_BA+0x108	R/W	P4数据输出值	0x0000_00FF

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

Bits	描述	
[31:8]	保留	保留
[n]	DOUT[n]	<p><b>Px Pin[n] 输出值</b></p> <p>Px配置成输出，输入，和准双端模式时，这些位控制Px引脚状态。</p> <p>1 = 相应的输出模式使能位设置时，Px Pin[n] 为高。</p> <p>0 =相应的输出模式使能位设置时，Px Pin[n] 为低。</p> <p>x=0~4, n = 0~7</p>

## Port0-4数据输出写屏蔽(Px\_DMASK)

寄存器	偏移量	R/W	描述	复位后的值
P0_DMASK	GP_BA+0x00C	R/W	P0数据输出写屏蔽	0xFFFF_XX00
P1_DMASK	GP_BA+0x04C	R/W	P1数据输出写屏蔽	0xFFFF_XX00
P2_DMASK	GP_BA+0x08C	R/W	P2数据输出写屏蔽	0xFFFF_XX00
P3_DMASK	GP_BA+0x0CC	R/W	P3数据输出写屏蔽	0xFFFF_XX00
P4_DMASK	GP_BA+0x10C	R/W	P4数据输出写屏蔽	0xFFFF_XX00

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

Bits	描述	
[31:8]	保留	保留
[n]	DMASK[n]	<p><b>Px 数据输出写屏蔽</b></p> <p>用于保护相应寄存器Px_DOUT bit[n]. 当设置DMASK bit[n] 为“1”，相应DOUTn bit 被保护，写信号被屏蔽时，不能向保护位写数据</p> <p>0 = 相应的Px_DOUT [n] 位未被屏蔽</p> <p>1 = 相应的Px_DOUT [n] 位被屏蔽</p> <p>x=0~4, n = 0~7</p>

## Port 0-4管脚数据(Px PIN)

寄存器	偏移量	R/W	描述	复位后的值
P0_PIN	GP_BA+0x010	R	P0管脚数据	0x0000_00XX
P1_PIN	GP_BA+0x050	R	P1管脚数据	0x0000_00XX
P2_PIN	GP_BA+0x090	R	P2管脚数据	0x0000_00XX
P3_PIN	GP_BA+0x0D0	R	P3管脚数据	0x0000_00XX
P4_PIN	GP_BA+0x110	R	P4管脚数据	0x0000_00XX

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

Bits	描述	
[31:8]	保留	保留
[n]	PIN[n]	<b>Px 管脚数据</b> 这些位的值为各个Px真实状态的反映 x=0~4, n = 0~7

## Port 0-4防反弹使能(Px DBEN)

寄存器	偏移量	R/W	描述	复位后的值
P0_DBEN	GP_BA+0x014	R/W	P0防反弹使能	0XXXXX_XX00
P1_DBEN	GP_BA+0x054	R/W	P1防反弹使能	0XXXXX_XX00
P2_DBEN	GP_BA+0x094	R/W	P2防反弹使能	0XXXXX_XX00
P3_DBEN	GP_BA+0x0D4	R/W	P3防反弹使能	0XXXXX_XX00
P4_DBEN	GP_BA+0x114	R/W	P4防反弹使能	0XXXXX_XX00

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

Bits	描述	
[31:8]	保留	保留
[n]	DBEN[n]	<p><b>Px输入信号防反弹使能</b></p> <p>DBEN[n] 用于使能相应位的防反弹功能。如果输入信号脉冲宽度不能被两个连续的防反弹采样周期所采样，则输入信号被被视为信号反弹，从而不触发中断。</p> <p>DBEN[n] 仅用于边沿触发 中断，不用于电平触发中断</p> <p>0 = 禁用 bit[n] 防反弹功能</p> <p>1 = 使能 bit[n] 防反弹功能</p> <p>防反弹功能对于边沿触发中断有效，对于电平触发中断模式，防反弹功能使能位不起作用。</p> <p>x=0~4, n = 0~7</p>

## Port 0-4中断模式控制(Px\_IMD)

寄存器	偏移量	R/W	描述	复位后的值
P0_IMD	GP_BA+0x018	R/W	P0中断模式控制	0xFFFF_XX00
P1_IMD	GP_BA+0x058	R/W	P1中断模式控制	0xFFFF_XX00
P2_IMD	GP_BA+0x098	R/W	P2中断模式控制	0xFFFF_XX00
P3_IMD	GP_BA+0x0D8	R/W	P3中断模式控制	0xFFFF_XX00
P4_IMD	GP_BA+0x118	R/W	P4中断模式控制	0xFFFF_XX00

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

Bits	描述	
[31:8]	保留	保留
[n]	IMD[n]	<p><b>Port 0-4中断模式控制</b></p> <p>IMD[n] 用于控制电平触发或边沿触发的中断。若中断由边沿触发，触发源是控制防反弹，如果是中断由电平触发，触发源由一个时钟采样并产生中断</p> <p>0 = 边沿触发中断 1 = 电平触发中断</p> <p>设置引脚为电平触发中断，仅需要在寄存器Px_IEN 设置一个电平，若设置为既有电平触发，又有边沿触发，设置将被忽略，不会产生中断</p> <p>防反弹功能对于边沿触发中断有效，对于电平触发中断无效。</p> <p>x=0~4, n = 0~7</p>



## Port 0-4中断使能控制(Px\_IEN)

寄存器	偏移量	R/W	描述	复位后的值
P0_IEN	GP_BA+0x01C	R/W	P0中断使能	0x0000_0000
P1_IEN	GP_BA+0x05C	R/W	P1中断使能	0x0000_0000
P2_IEN	GP_BA+0x09C	R/W	P2中断使能	0x0000_0000
P3_IEN	GP_BA+0x0DC	R/W	P3中断使能	0x0000_0000
P4_IEN	GP_BA+0x11C	R/W	P4中断使能	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[n+16]	IR_EN[n]	<p><b>Port 0-4 输入上升沿或输入高电平的中断使能</b></p> <p>IR_EN[n] 用于使能相应Px[n]输入的中断。置“1”也可以使能引脚唤醒功能 设置 IR_EN[n] 位为“1”:</p> <p>如果中断是电平触发模式，输入Px[n]的状态为高电平时，产生中断。 如果中断是边沿触发模式，输入Px[n]的状态由低电平到高电平变化时，产生中断</p> <p>1 = 使能Px[n] 高电平或由低电平到高电平变化的中断 0 = 禁用Px[n] 高电平或由低电平到高电平变化的中断.</p> <p>x=0~4, n = 0~7</p>
[15:8]	保留	保留

[n]	IF_EN[n]	<p><b>Port 0-4输入下降沿或输入低电平的中断使能</b></p> <p>IF_EN[n] 用于使能相应Px[n]输入的中断。置“1”也可以使能引脚唤醒功能 设置 IF_EB[n] 位为“1”:</p> <p>如果中断是电平触发模式，输入Px[n]的状态为低电平时，产生中断。 如果中断是边沿触发模式，输入Px[n]的状态由高电平到低电平变化时，产生中断。</p> <p>1 =使能Px[n]低电平或由高电平到低电平变化的中断 0 =禁用Px[n]低电平或由高电平到低电平变化的中断</p> <p>x=0~4, n = 0~7</p>
-----	----------	--

## Port 0-4 中断触发源(Px ISRC)

寄存器	偏移量	R/W	描述	复位后的值
P0_ISRC	GP_BA+0x020	R/WC	P0中断触发源	0x0000_0000
P1_ISRC	GP_BA+0x060	R/WC	P1中断触发源	0x0000_0000
P2_ISRC	GP_BA+0x0A0	R/WC	P2中断触发源	0x0000_0000
P3_ISRC	GP_BA+0x0E0	R/WC	P3中断触发源	0x0000_0000
P4_ISRC	GP_BA+0x120	R/WC	P4中断触发源	0x0000_0000

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

Bits	描述	
[31:8]	保留	保留
[n]	ISRC[n]	<p><b>Port 0-4 中断触发源</b></p> <p>读：</p> <p>1 = Px[n]产生中断</p> <p>0 = Px[n]没有中断</p> <p>写：</p> <p>1= 清相应的中断标志</p> <p>0= 无动作</p> <p>x=0~4, n = 0~7</p>

## 中断防反弹周期控制(DBNCECON)

寄存器	偏移量	R/W	描述	复位后的值
DBNCECON	GP_BA+0x180	R/W	外部中断防反弹控制	0x0000_0020

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

Bits	描述
[5]	<p><b>ICLK_ON</b></p> <p><b>中断时钟On模式</b> 如果禁用pin[n]中断，设置该位为“0”将禁用中断产生时钟电路</p> <p>0 = 如果中断P0/1/2/3/4[n]被禁用，禁用时钟 1 = 总是使能中断产生时钟电路</p> <p>n=0~7</p>
[4]	<p><b>DBCLKSRC</b></p> <p><b>防反弹计数器时钟源选择</b></p> <p>1 = 防反弹计数器时钟源为内部10KHz 时钟 0 = 防反弹计数器时钟源为 HCLK</p>

[3:0]	DBCLKSEL	防反弹采样周期选择	
		DBCLKSEL	描述
		0	采样中断输入每 1 个时钟周期一次
		1	采样中断输入每 2 个时钟周期一次
		2	采样中断输入每4 个时钟周期一次
		3	采样中断输入每8 个时钟周期一次
		4	采样中断输入每16 个时钟周期一次
		5	采样中断输入每32 个时钟周期一次
		6	采样中断输入每64 个时钟周期一次
		7	采样中断输入每128 个时钟周期一次
		8	采样中断输入每256 个时钟周期一次
		9	采样中断输入每 2*256 个时钟周期一次
		10	采样中断输入每4*256个时钟周期一次
		11	采样中断输入每8*256 个时钟周期一次
		12	采样中断输入每16*256 个时钟周期一次
		13	采样中断输入每32*256 个时钟周期一次
14	采样中断输入每64*256 个时钟周期一次		
15	采样中断输入每128*256 个时钟周期一次		



## GPIO 端口 [P0/P1/P2/P3/P4] I/O 位输出控制 (Pxx\_DOUT)

寄存器	偏移量	R/W	描述	复位后的值
P0x_DOUT	GP_BA+0x200 - GP_BA+0x21C	R/W	P0 Pin I/O位输出控制	0x0000_0001
P1x_DOUT	GP_BA+0x220 - GP_BA+0x23C	R/W	P1 Pin I/O位输出控制	0x0000_0001
P2x_DOUT	GP_BA+0x240 - GP_BA+0x25C	R/W	P2 Pin I/O位输出控制	0x0000_0001
P3x_DOUT	GP_BA+0x260 - GP_BA+0x27C	R/W	P3 Pin I/O位输出控制	0x0000_0001
P4x_DOUT	GP_BA+0x280 - GP_BA+0x29C	R/W	P4 Pin I/O位输出控制	0x0000_0001

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

Bits	描述	
[0]	Pxx_DOUT	<b>Pxx I/O位输出控制</b> 设置该位可以控制一个GPIO管脚的输出值 1 = 设置相应的GPIO位为高

		0 = 设置相应的GPIO位为低
--	--	------------------

## 6.5 I2C 总线控制器 (主机/从机)

### 6.5.1 简介

I2C为双线，双向串行总线，为设备之间的数据通讯提供了简单有效的方法。标准I2C是多主机总线，包括冲突检测和仲裁机制以防止在两个或多个主机试图同时控制总线时发生的数据冲突。

数据在主机与从机间通过SCL时钟线控制在SDA数据上实现一字节一字节的同步传输，每个字节为8位长度，一个SCL时钟脉冲传输一个数据位，数据由最高位MSB首先传输，每个传输字节后跟随一个应答位，每个位在SCL为高时采样；因此，SDA线只有在SCL为低时才可以改变，在SCL为高时SDA保持稳定。当SCL为高时，SDA线上的跳变视为命令中断（START 或 STOP），参考下图I2C总线时序。

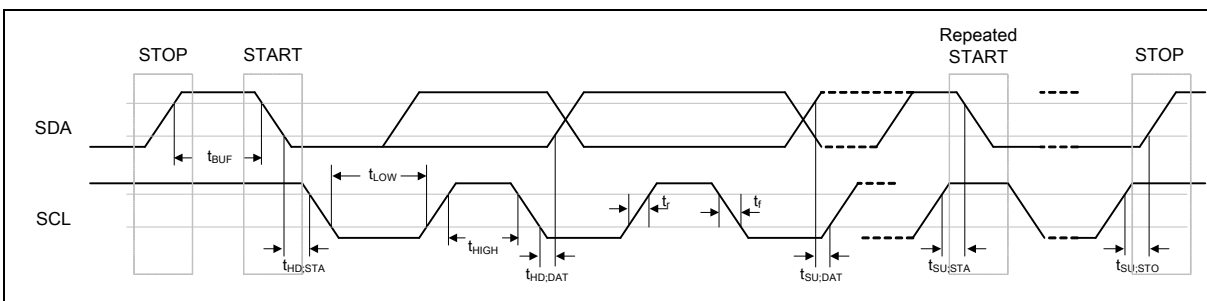


图 6.5-1 I2C 总线时序

该设备符合I2C总线规范，I2C端口自动处理字节传输，将I2CON的ENS1位设置为1，可以使能该端口。I2C H/W 接口: SDA (Px.y, 串行数据线) 与 SCL (Px.y, 串行时钟线)。引脚Px.y 与 Px.y 用于 I2C操作需要上拉电阻，因为这两个引脚为开漏脚。在作为 I2C 端口使用时，用户必须先将引脚设置为I2C功能。

### 6.5.2 特征

I2C总线通过SDA 及 SCL在连接在总线上的设备间传输数据，总线的主要特征:

- 支持主机和从机模式
- 主从机之间双向数据传输
- 多主机总线支持 (无中心主机)
- 总线上多主机间连续传输数据，实现无间断仲裁
- 总线上不同传输速率设备间，实现同步传输
- 总线上数据暂停传送及恢复传送时，实现同步时钟功能
- 内建14位溢出计数器，当I2C总线上闲置并计数器溢位时，产生I2C中断。
- 需要外部上拉确保高电平输出
- 可编程的时钟适用于不同速率控制
- 支持7位地址模式



- I2C总线上支持多地址辨识 (4组从机地址带mask选项)

### 6.5.3 功能描述

#### 6.5.3.1 I2C协议

通常标准I2C传输协议包含四个部分:

- 1) 起始信号或重复起始信号的产生
- 2) 从机地址和R/W位传输
- 3) 数据传输
- 4) 停止信号的产生

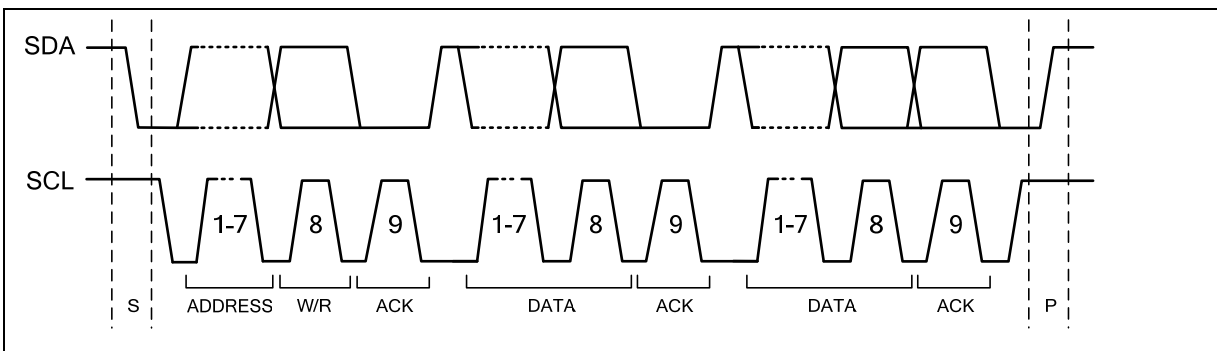


图 6.5-2 I2C 协议

### 6.5.3.2 I2C总线上的数据传输

主机发出，从机接收7位地址(一个字节)

传输方向未改变

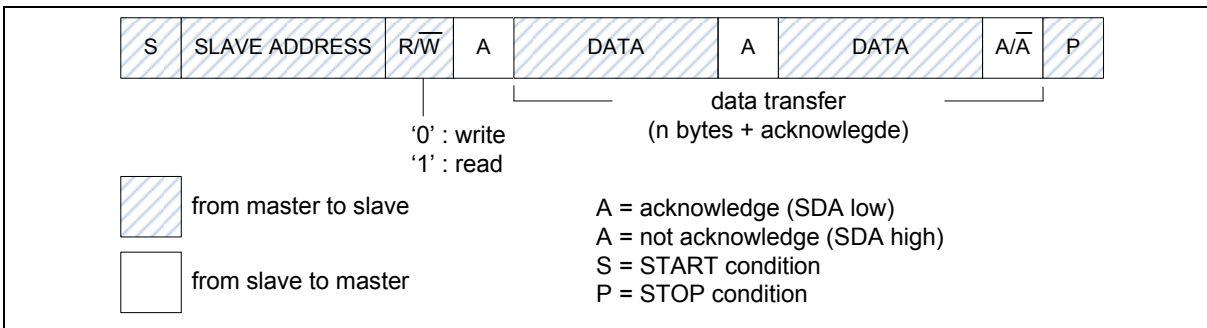


图 6.5-3 主机向从机传输数据

第一个字节后主机紧接着由从机读取数据(内容为从机地址)

传输方向改变

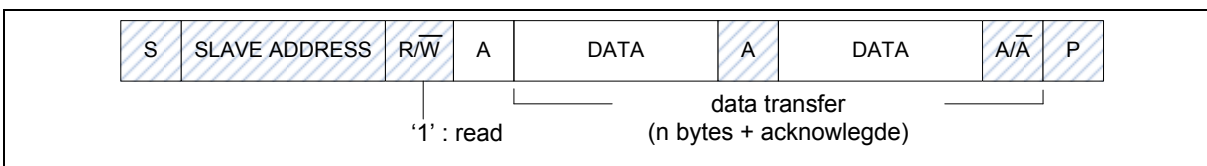


图 6.5-4 主机读取从机的数据

### 6.5.3.3 起始或重复起始信号

当总线处于空闲状态下，说明没有主机对总线发起传输请求(SCL 和SDA线同时为高)，主机可以对总线发出起始信号。起始信号，通常表示为**S-bit**，当SCL线为高时，SDA线上信号由高至低，标示总线上产生起始信号，新的传输开始。

重复起始信号 (Sr) 是指在两个START信号间不存在STOP信号。主机用这种方式来和从机在不同的传输方向并且不释放总线的情形下通讯（如从写向一个设备到从该设备读取）。

#### 停止信号

主机向总线发出停止信号结束数据传送。停止信号，通常用**P-bit**表示，当SCL线为高时，SDA线上出现由低到高的信号，被定义为停止信号。

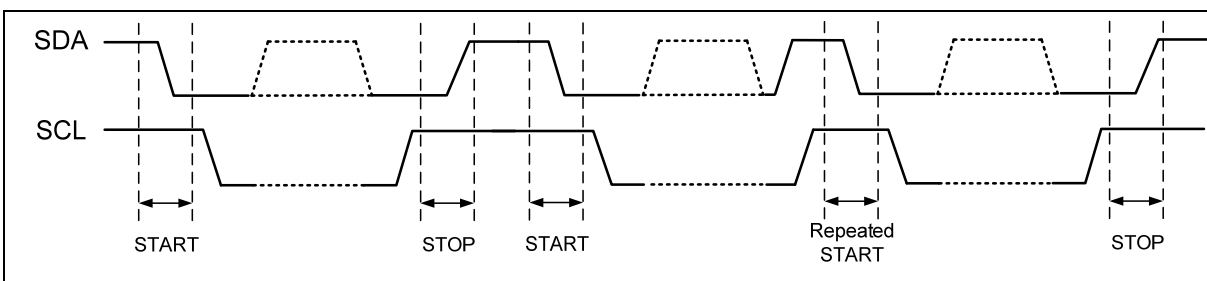


图 6.5-5 START 和 STOP 条件

#### 6.5.3.4 从机地址传输

起始信号后传输第一个字节是从机地址，从机地址的头7位组是呼叫地址，紧跟7位地址后的是RW位。RW位决定了从机是接收还是发送。系统当中不会有两台从机有相同的地址。地址匹配的从机会在SCL的第9个时钟周期拉低SDA作为应答信号。

#### 6.5.3.5 数据传输

当从机地址被成功识别，就可以根据RW所决定的方向，开始一字节一字节的数据传输，每字节最后带一个应答信号，如果从机上产生无应答信号(NACK)，主机产生停止信号，或者产生重复起始信号开始新一轮的数据传输。

当主机作为接收器件时，发生无应答信号(NACK)，从机释放SDA线，使主机产生停止信号或重复起始信号。

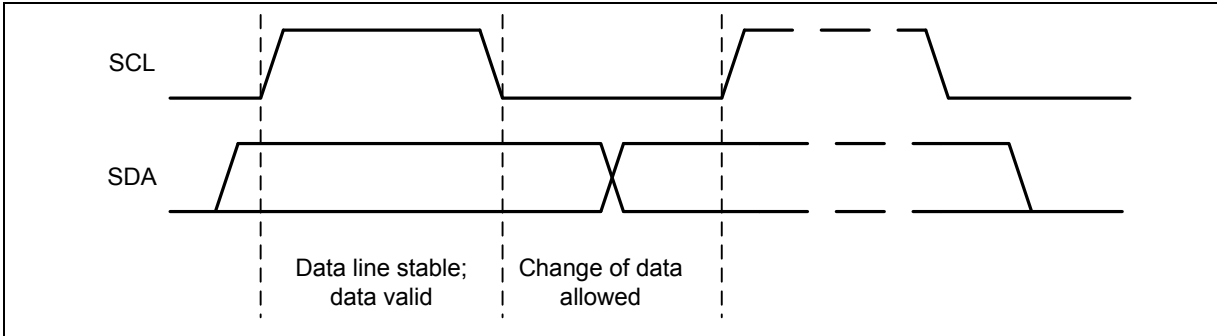


图 6.5-6 I2C 总线上的位传输

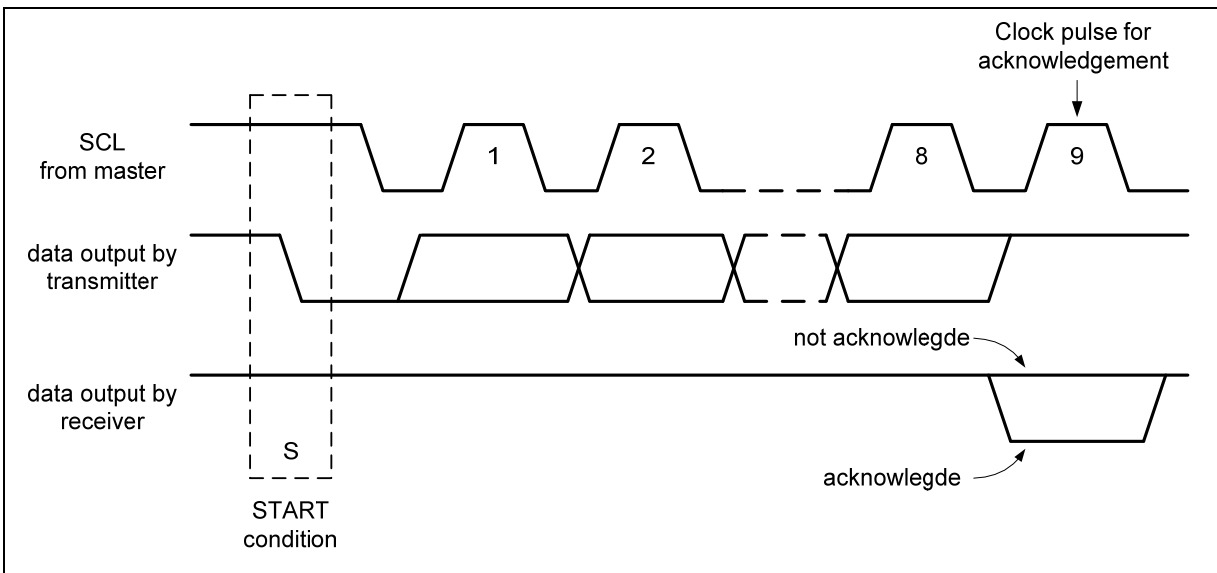


图 6.5-7 I2C 总线上的应答信号

### 6.5.4 I2C 协议寄存器

CPU的下列特殊功能寄存器，用于I2C通讯：I2CON (控制寄存器), I2CSTATUS (状态寄存器), I2CDAT (数据寄存器), I2CADDRn (地址寄存器, n=0~3), I2CADMn (地址屏蔽寄存器, n=0~3) I2CLK (时钟速率寄存器) 和 I2CTOC (超时计数寄存器)。寄存器的第31位至第8位都是保留的，不具备任何功能，读回值为0。

当ENS1(I2CON[6])置1, I2C端口使能后，内部状态由 I2CON 和 I2C总线逻辑状态控制。当有新的状态发生后，会存储到 I2CSTATUS, I2C 中断标志(SI(I2CON[3])) 也会自动置位。若此时 EI(I2CON[7]) 被设定为高, I2C中断会被响应。I2CSTATUS [7:3] 内存储状态码，在SI由软件清除之前, I2CSTATUS的低三位始终保持低。在NUC1XX上的地址为 4002\_0000。

#### 6.5.4.1 地址寄存器(I2CADDR)

I2C端口内建4个从机地址寄存器I2CADDRn (n=0~3)。这四个寄存器当I2C作为主机时值无效。在从机模式下, I2CADDRn [7:1]必须装入MCU自身从机地址，当I2CADDR地址与接收的从机地址符合时, I2C硬件起作用。

I2C 口支持广播呼叫功能。当GC位(I2CADDRn [0])被置位, 硬件会响应广播呼叫地址(00H)。清GC位可禁止广播呼叫功能。

当GC位被置位, 当I2C作为从机时, 可以在I2C总线上的主机发出广播呼叫地址后, 按照00H地址接收信号, 然后按照广播呼叫模式工作。

I2C总线控制器支持多地址辨识, 最多可辨识4组屏蔽地址I2CADMn (n=0~3)。当地址屏蔽寄存器某一位置1, 意味着接收到的相应地址将被忽略。当该位保持0, 说明接收到的响应地址完全符合地址寄存器内的值。

#### 6.5.4.2 数据寄存器(I2CDAT)

该寄存器存储的内容是准备发送的或刚接收的串行数据一个字节的的数据。只要不在移位处理的过程中, CPU可以读写访问8位寄存器I2CDAT [7:0]。当I2C的状态设定后并且串行中断标志(SI)置位, I2CDAT[7:0]中的数据一直是稳定的。在数据移出的过程中, 总线上的数据同时也在移动; I2CDAT的内容一直是总线上出现的最后一个字节。在主机发送数据从机接收数据的模式中, 不需要仲裁来保证I2CDAT [7:0]中的数据正确。

移位寄存器包含I2CDAT [7:0]和应答位, 共9位, 应答位由I2C的硬件控制, CPU不能访问。I2CDAT [7:0]中的串行数据和应答位在串行时钟SCL的上升沿移出。当一个字节被移位到I2CDAT [7:0]后, I2CDAT [7:0]中的串行数据是可以使用的, 应答位(ACK或NACK) 由控制逻辑在第9个时钟返回。串行数据在每一个下降沿(SCL时钟)从I2CDAT [7:0]移出输出, 在每一个上升沿(SCL时钟)数据移进I2CDAT [7:0]。

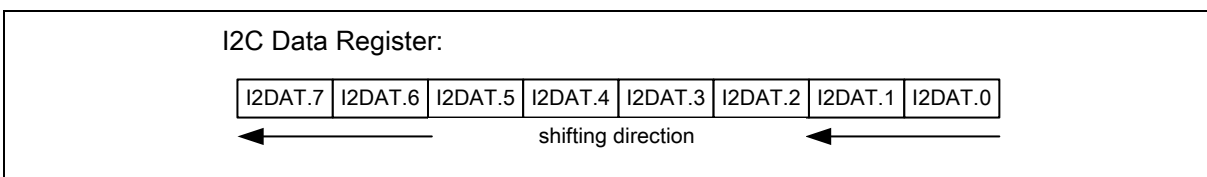


图 6.5-8 I2C 数据移位

#### 6.5.4.3 控制寄存器(I2CON)

CPU可以直接读写寄存器I2CON [7:0]的各个位，I2CON 有2位会受到硬件的影响：SI位在I2C硬件请求一个串行中断时被置位，STO位在总线上出现停止条件或者ENS1=0时被清除。

EI	中断使能
ENS1	置位允许I2C串行功能。当ENS1=1，使能I2C功能。复用管脚SDA与SCL必须设置为I2C功能。
STA	I2C开始控制位。置1，STA为高，进入主机模式。如果总线为空闲，I2C 硬件向总线发送起始信号或者重复起始信号。
STO	I2C停止控制位，为主机模式时，STO位置位，将在I2C总线上输出停止信号。当检测到总线上出现停止信号，I2C硬件清除STO标志。在从机模式，STO标志被置位将复位I2C硬件至不可寻址从机模式。这意味着该设备不再处于从机接收模式，不能从主发送设备接收数据。
SI	I2C中断标志。当有新的SIO状态在I2CSTATUS寄存器中时，SI标志由硬件置位，如果EI（I2CON [7]）置位，则产生I2C中断请求。SI 必须由软件清零。向该位写1清零。
AA	声明应答控制位。若在地址或数据接收之前，AA=1，则在下列情况下：1.从机应答主机发送主机的地址信息 2.接收设备应答发送设备发送的数据 时将在SCL时钟的应答时钟脉冲间返回应答信号（SDA为低）；若在地址或数据接收之前，AA=0，则在SCL的应答时钟脉冲间不会返回应答信号（SDA为高）。

#### 6.5.4.4 状态寄存器(I2CSTATUS)

I2CSTATUS [7:0] 是一个8-位只读寄存器。低3位一直为0。其余是状态码。有26个可能的状态码，均在6.5.6章节中列出。当I2CSTATUS [7:0]的内容是F8H，没有串行中断请求。所有的其它I2CSTATUS [7:3]值对应I2C 端口的状态。当每一个进入状态，就会产生状态中断请求(SI = 1)。在SI被硬件置位 1个机器周期后，有效状态码出现在I2CSTATUS [7:3]中，并保持稳定至SI被软件清除的下一个机器周期。

另外，00H状态表示总线错误。总线错误发生在起始或停止信号出现在帧结构非法的位置。非法位置比如是在串行传输地址字节出现数据字节或应答位。为了将I2C从总线错误中恢复，需要置位STO，清除SI从而进入不可寻址从机模式，进而清除STO并等待新的通信。I2C总线在总线错误时不能识别停止信号。

#### 6.5.4.5 I2C 时钟波特率位 (I2CLK)

当SIO在主机模式下，I2C数据的波特率由I2CLK[7:0]寄存器设定。波特率在从机模式下时是不重要的；

在从机模式下，SIO将自动与主机I2C设备时钟频率同步，频率可高达1MHz。

I2C数据波特率设定是： $I2C \text{ 的数据波特率} = I2C = PCLK / (4x(I2CLK[7:0]+1))$ ，如果PCLK=16MHz，I2CLK[7:0]= 40(28H)，I2C的数据波特率 $I2C = 16MHz / (4X (40 +1)) = 97.5K \text{位/秒}$ 。

#### 6.5.4.6 The I2C超时计数寄存器 (I2CTOC)

当总线闲置时，MCU提供一个14位超时的计数器。当计数功能使能后，计数器开始计数直至发生超时，此时TIF置1，并向CPU产生I2C中断或者清除ENT1为0关闭计数功能。当超时计数器使能，对SI标志置高会使计数器复位，再对SI清零会重新开始计数。如果I2C总线闲置，会使I2STATUS及SI标志在一段时间内不再更新。该14位超时计数器同样会发生溢出并向CPU发出I2C中断请求。参考下图，用户可通过对TIF位软件写1清0。

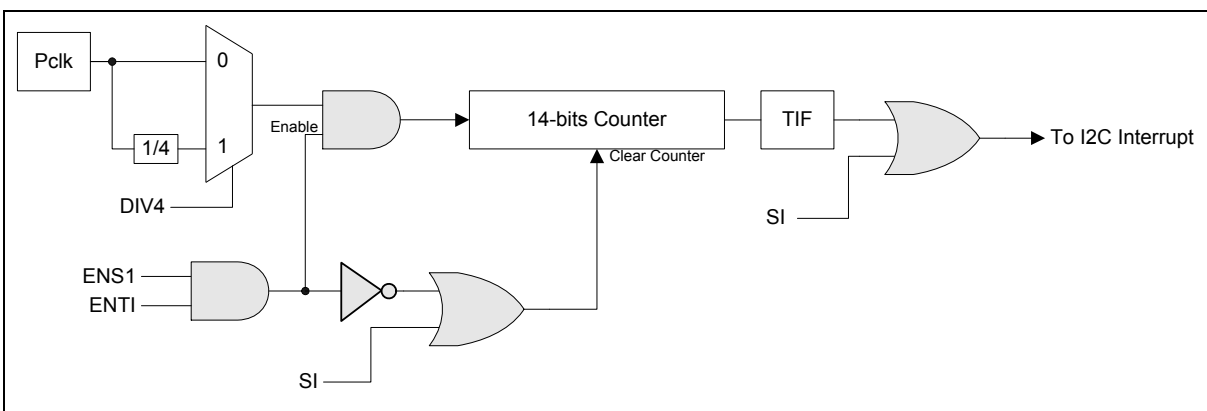


图 6.5-9: I2C 超时计数器框图



### 6.5.5 I2C 控制器寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
I2CON	I2C_BA+0x00	R/W	I2C控制寄存器	0x0000_0000
I2CADRR0	I2C_BA+0x04	R/W	I2C 从机地址寄存器0	0x0000_0000
I2CDAT	I2C_BA+0x08	R/W	I2C 数据寄存器	0x0000_0000
I2CSTATUS	I2C_BA+0x0C	R	I2C 状态寄存器	0x0000_00F8
I2CLK	I2C_BA+0x10	R/W	I2C 时钟时钟分频寄存器	0x0000_0000
I2CTOC	I2C_BA+0x14	R/W	I2C 超时控制寄存器	0x0000_0000
I2CADDR1	I2C_BA+0x18	R/W	从机地址寄存器1	0x0000_0000
I2CADDR2	I2C_BA+0x1C	R/W	从机地址寄存器2	0x0000_0000
I2CADDR3	I2C_BA+0x20	R/W	从机地址寄存器3	0x0000_0000
I2CADM0	I2C_BA+0x24	R/W	从机隐藏地址寄存器0	0x0000_0000
I2CADM1	I2C_BA+0x28	R/W	从机隐藏地址寄存器1	0x0000_0000
I2CADM2	I2C_BA+0x2C	R/W	从机隐藏地址寄存器2	0x0000_0000
I2CADM3	I2C_BA+0x30	R/W	从机隐藏地址寄存器3	0x0000_0000

## 6.5.6 I2C 控制器寄存器描述

### I2C控制寄存器(I2CON)

寄存器	偏移量	R/W	描述	复位后的值
I2CON	I2C_BA+0x00	R/W	I2C 控制寄存器	0x0000_0000

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

Bits	描述	
[31:8]	保留	保留
[7]	EI	使能中断 1 = 使能CPU中断功能 0 = 禁用CPU中断功能
[6]	ENSI	<b>I2C 控制使能位</b> 1 = 使能 0 = 禁用 当ENS=1 I2C串行功能使能，SDA和SCL 管脚必须设置为I2C功能
[5]	STA	<b>I2C 起始控制位</b> STA置1，进入主机模式，如果总线处于空闲状态，I2C硬件会送出起始信号或重复起始信号。
[4]	STO	<b>I2C 停止控制位</b> 在主机模式下，置位STO将向总线传输停止条件，进而I2C硬件会检查总线状态。一旦检测到停止条件，该位将被硬件自动清零。在从机模式下，置位STO会将I2C硬件复位至不可寻址从机模式。这意味着该设备不再处于从机接收模式，不能从主发送设备接收数据。

[3]	SI	<p><b>I2C 中断标志位</b></p> <p>I2CSTATUS 寄存器有新的SIO状态时，硬件置位SI标志。如果EI (I2CON [7])已经置位，就产生I2C中断请求。SI 必须由软件清零。向该位写1清零。</p>
[2]	AA	<p><b>接收应答控制位</b></p> <p>若在地址或数据接收之前，AA=1，则在下列情况下：1.从机应答主机发送主机的地址信息 2.接收设备应答发送设备发送的数据 时将在SCL时钟的应答时钟脉冲间返回应答信号（SDA为低）；若在地址或数据接收之前，AA=0，则在SCL的应答时钟脉冲间不会返回应答信号（SDA为高）。</p>
[1:0]	保留	保留

**I2C 数据寄存器 (I2CDAT)**

寄存器	偏移量	R/W	描述	复位后的值
I2CDAT	I2C_BA+0x08	R/W	I2C 数据寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
I2CDAT[7:0]							

Bits	描述	
[31:8]	保留	保留
[7:0]	I2CDAT	I2C 数据寄存器 Bit[7:0] 为8位I2C串行端口的传输数据.

**I2C状态寄存器 (I2CSTATUS)**

寄存器	偏移量	R/W	描述	复位后的值
I2CSTATUS	I2C_BA+0x0C	R/W	I2C 状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
I2CSTATUS[7:0]							

Bits	描述	
[31:8]	保留	保留
[7:0]	I2CSTATUS	<b>I2C状态寄存器</b> 低三位始终是0；高5位包含状态码。含状态码有26可能；当I2CSTATUS的值是F8H，表示没有串行中断请求；其它的所有的I2CSTATUS值可以反映I2C的状态。当进入这些状态时会产生一个状态中断请求(SI=1)。一个有效的状态码在SI被硬件设为'1'后一个周期内反映到I2CSTATUS中，并保持稳定至SI被软件清零的下一个周期。另外，状态码是00H时表示总线错误；当'起始'或'结束'时出现帧结构的非法位置时会产生总线错误。比如在串行传输地址字节中出现的数据字节或应答位就是非法的。

## I2C波特率控制寄存器(I2CLK)

寄存器	偏移量	R/W	描述	复位后的值
I2CLK	I2C_BA+0x10	R/W	I2C时钟分频寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
I2CLK[7:0]							

Bits	描述	
[31:8]	保留	保留
[7:0]	I2CLK	I2C 波特率控制寄存器 I2C波特率 = PCLK / (4x(I2CLK+1)).

## I2C超时计数寄存器 (I2CTOC)

寄存器	偏移量	R/W	描述	复位后的值
I2CTOC	I2C_BA+0x14	R/W	I2C超时计数寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留					ENTI	DIV4	TIF

Bits	描述	
[31:3]	保留	保留
[2]	ENTI	<b>超时计数使能/禁用</b> 1 = 使能 0 = 禁用 当计数器被使能，SI被清0后，14位超时计数寄存器开始计数。对SI置1会使计数器复位，在SI清零后计数器重新开始计数
[1]	DIV4	<b>超时计数输入时钟除4</b> 1 = 使能 0 = 禁用 使能后，溢出时间延长4倍。
[0]	TIF	<b>超时标志</b> 1 = 超时由硬件置位，可引发CPU的中断 0 = 软件清零。

## I2C 从机地址寄存器(I2CADDRx)

寄存器	偏移量	R/W	描述	复位后的值
I2CADDR0	I2C_BA+0x04	R/W	I2C 从机地址寄存器0	0x0000_0000
I2CADDR1	I2C_BA+0x18	R/W	I2C从机地址寄存器1	0x0000_0000
I2CADDR2	I2C_BA+0x1C	R/W	I2C 从机地址寄存器2	0x0000_0000
I2CADDR3	I2C_BA+0x20	R/W	I2C 从机地址寄存器3	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
I2CADDR[7:1]							GC

Bits	描述	
[31:8]	保留	保留
[7:1]	I2CADDR	I2C地址寄存器: 主机模式下, 该寄存器的值无效, 从机模式下, 高七位为MCU自身地址, I2C硬件会匹配是否与该值相符.
[0]	GC	广播呼叫功能. 0: 禁用广播呼叫功能. 1: 允许广播呼叫功能



## I2C 从机隐藏地址寄存器(I2CADMx)

寄存器	偏移量	R/W	描述	复位后的值
I2CADM0	I2C_BA+0x24	R/W	I2C s从机隐藏地址寄存器0	0x0000_0000
I2CADM1	I2C_BA+0x28	R/W	I2C从机隐藏地址寄存器1	0x0000_0000
I2CADM2	I2C_BA+0x2C	R/W	I2C从机隐藏地址寄存器2	0x0000_0000
I2CADM3	I2C_BA+0x30	R/W	I2C从机隐藏地址寄存器3	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
I2CADMx[7:1]							保留

Bits	描述	
[31:8]	保留	保留
[7:1]	I2CADMx	I2C 隐藏地址寄存器: 1 = 允许隐藏(接收到任何地址不予辨识) 0 = 禁用隐藏 (接收到的地址必须完全符合正确的地址内容) I2C总线支持多隐藏地址辨识。当设置允许隐藏时, 接收到的从机地址是否正确不予处理, 当选择为禁用隐藏是, 从机地址必须完全符合其真实的地址才给与响应.
[0]	保留	保留

### 6.5.7 操作模式

片上I2C端口支持5种操作模式：主机发送，主机接收，从机发送，从机接收和广播呼叫模式。

在实际应用中，I2C端口可以作为主机和从机。在从机模式，I2C端口寻找自身从机地址和广播呼叫地址，如果有地址被检测到，从机准备接收或发送数据（通过设置AA位），应答信号在第9个时钟脉冲时发送，此时，如果中断已经使能，则发生中断请求。在主控芯片要成为总线主机时，在进入主机模式之前，硬件等待总线空闲以使主机不产生中断，在主机模式，如果总线仲裁丢失，I2C立即切换到从机模式，并检测自身从机地址。

#### 6.5.7.1 主机发送模式

当SCL线上输出时钟信号时，SDA线上输出数据。主机产生起始信号，第一个发送的字节包含从设备的地址（7位）和数据传输方向位（1位）。在该模式下，方向位(R/W)为0，在图6.5-11中以“W”表示。这样传输内容为SLA+W，串行发送一次传输8位。在每个字节发送完成时，将接收到一个应答位。起始和停止条件将被输出以表明串行传输的开始和结束。

#### 6.5.7.2 主机接收模式

第一个发送的字节包含从设备的地址（7位）和数据传输方向位（1位）。在该模式下，方向位(R/W)为0，在图6.5-11中以“W”表示。这样传输内容为SLA+W，串行发送一次传输8位。在每个字节发送完成时，将接收到一个应答位。起始和停止条件将被输出以表明串行传输的开始和结束。

#### 6.5.7.3 从机接收模式

在该模式下，串行数据和串行时钟通过SDA和SCL接收。在接收到一个字节后，一个应答位将被发送。起始和停止条件将被确认为串行传输的开始和结束。地址识别将在从地址和数据传输方向位接收到时由硬件执行。

#### 6.5.7.4 从机发送模式

第一个字节在从机接收模式下接收和处理。在该模式下，数据传输方向位用来标示传输方向是颠倒的。SCL线上接收时钟信号，SDA脚上输出数据。起始和停止条件将被确认为串行传输的开始和停止。

## 6.5.8 5种操作模式下的数据传输流

5种操作模式：主机发送，主机接收，从机发送，从机接收和广播呼叫模式。在SI位清除后，I2CON中的STA，STO和AA位将决定SIO硬件下一次的的状态。一个新的动作完成后，将更新新的状态码并置位SI标志。如果I2C中断控制位EI（I2CON [7]）置位，新状态码对应的动作和软件行为将在中断服务子程序中执行。

数据传输每种模式如下图所示。

\*\*\* Legend for the following five figures:

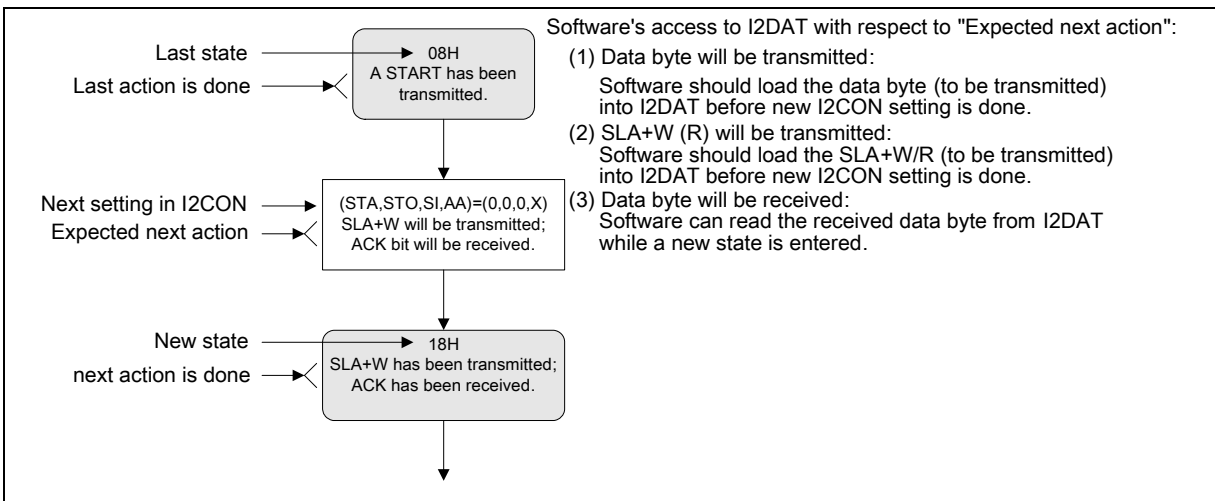


图 6.5-10 Legend for the following four figures

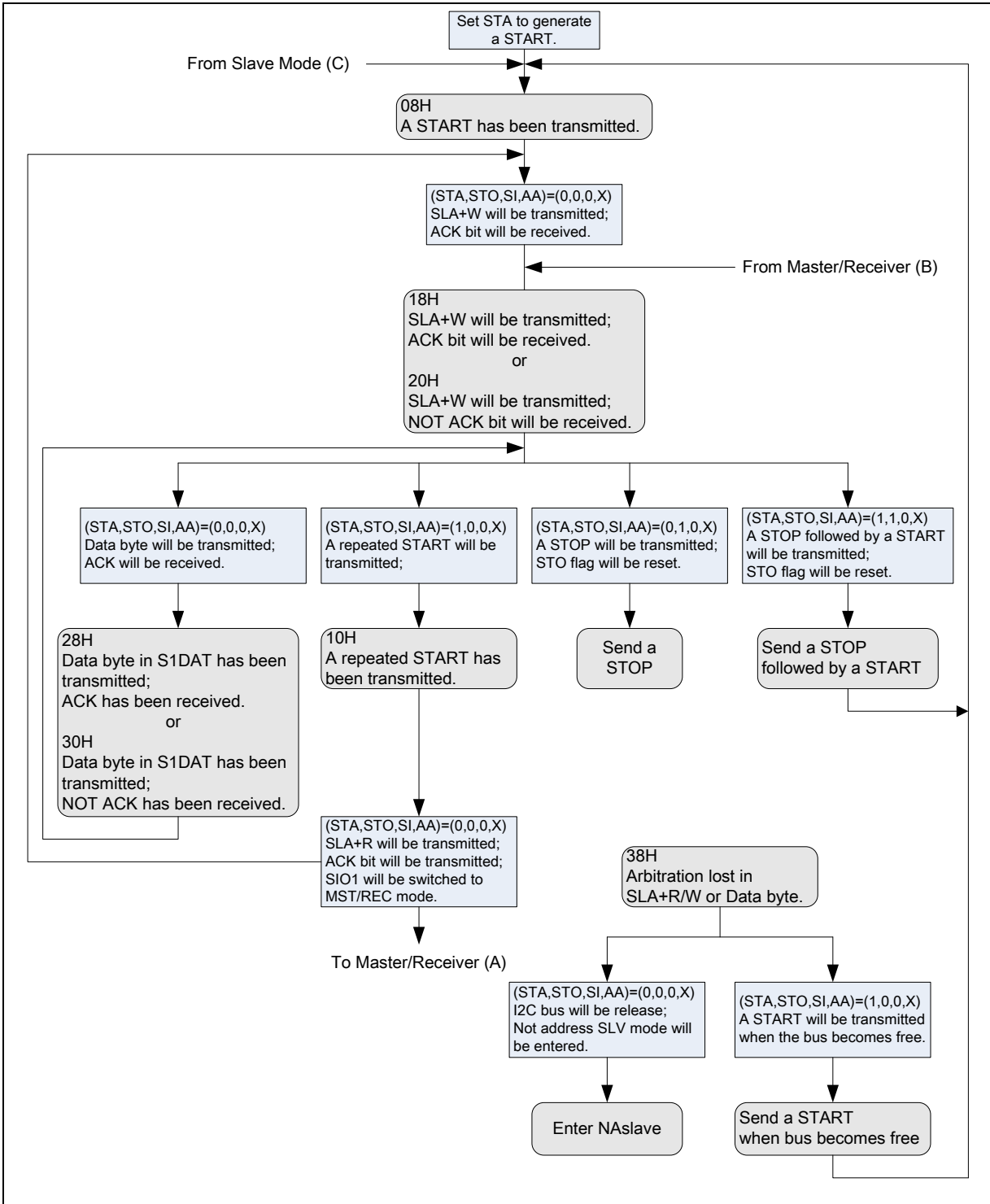


图 6.5-11 主机发送模式

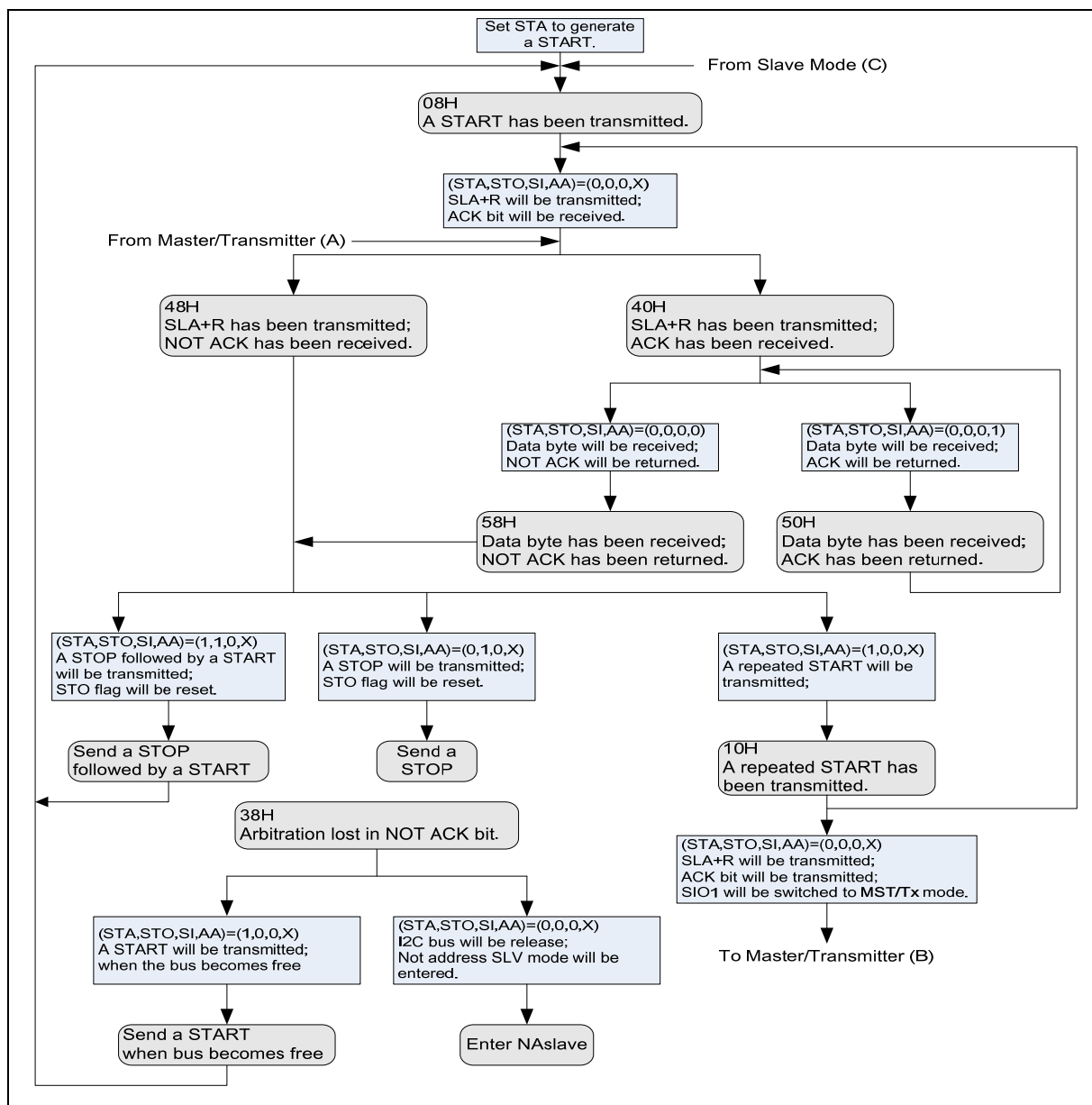


图 6.5-12 主机接收模式

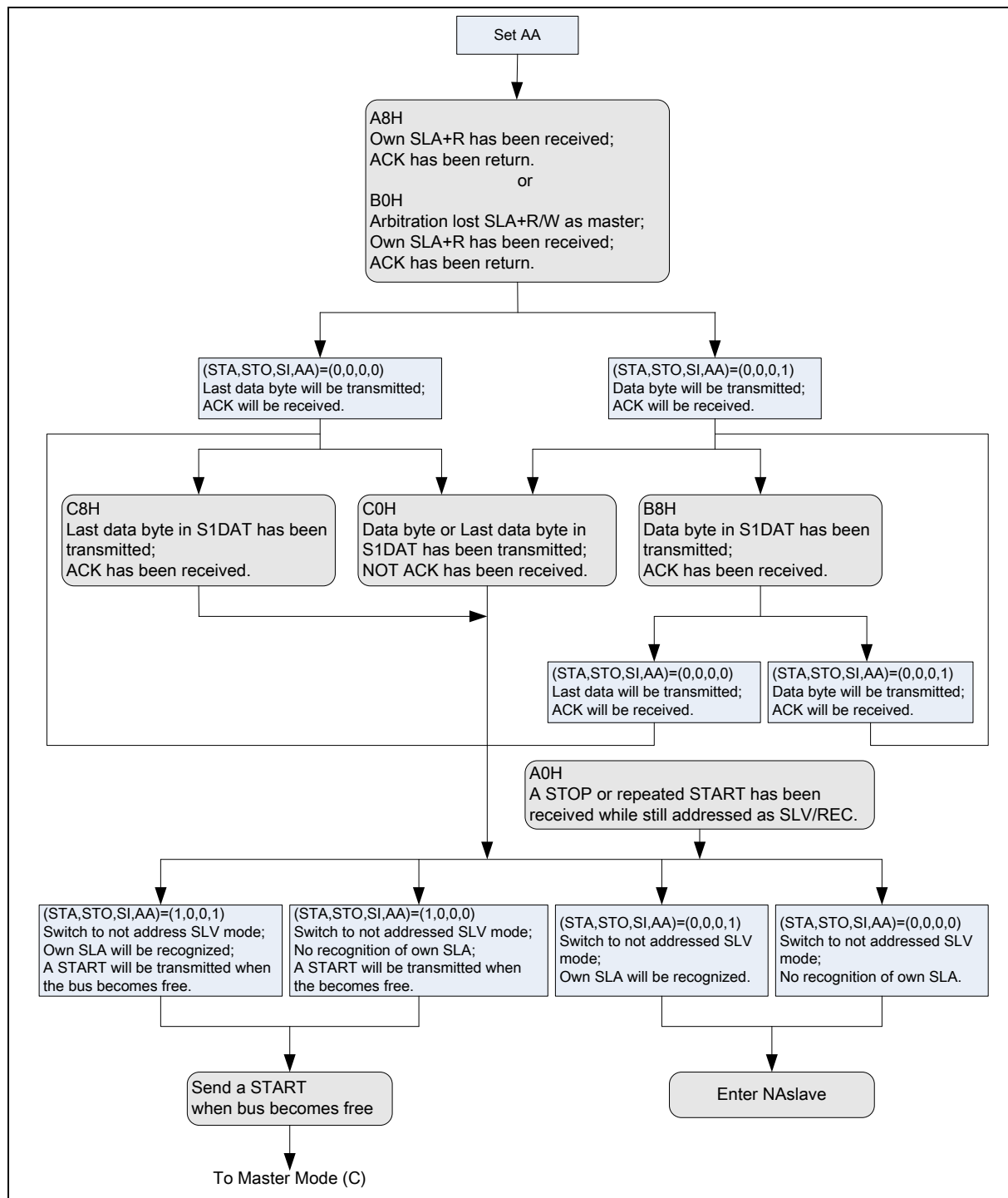


图 6.5-13 从机发送模式

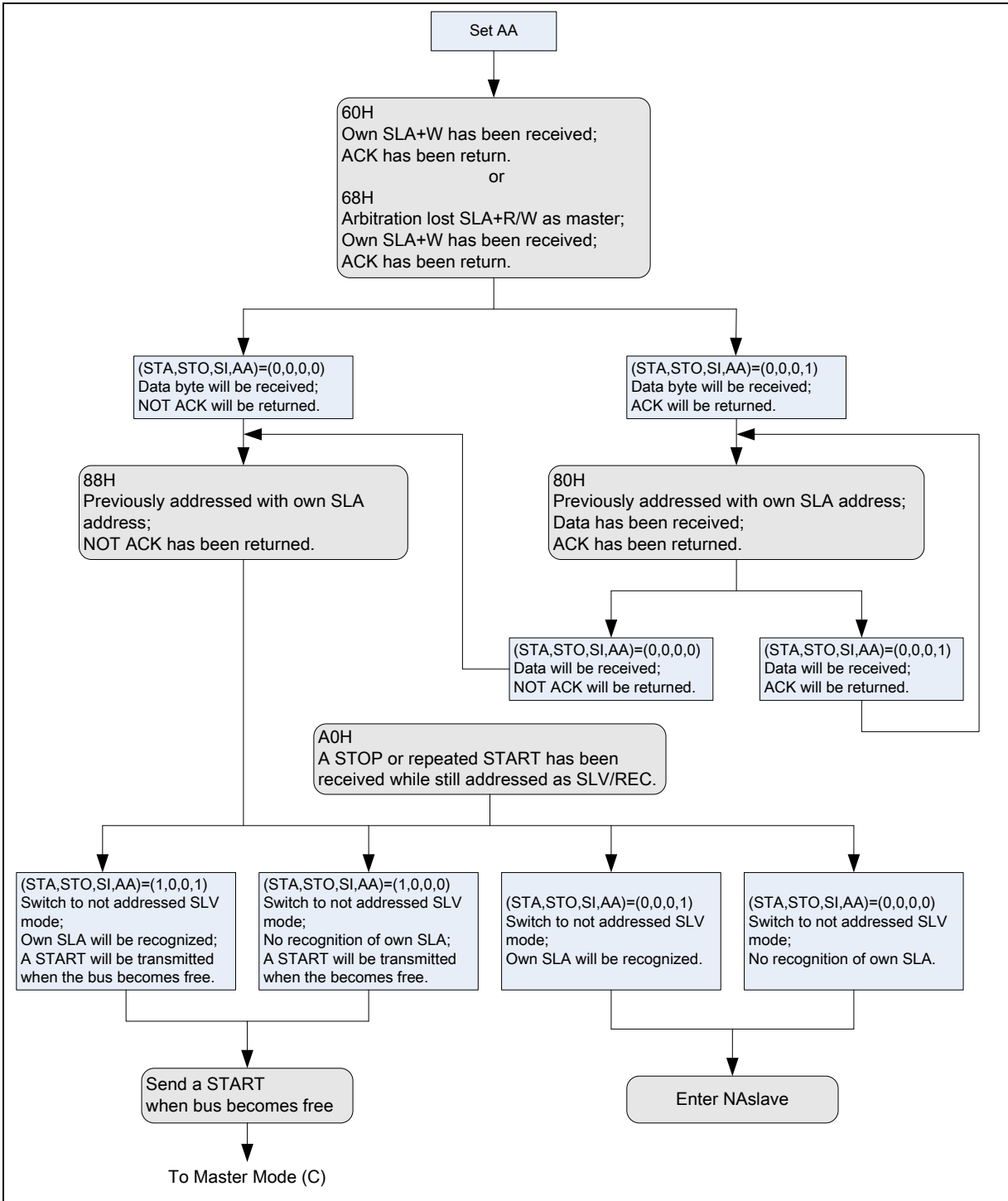


图 6.5-14 从机接收模式

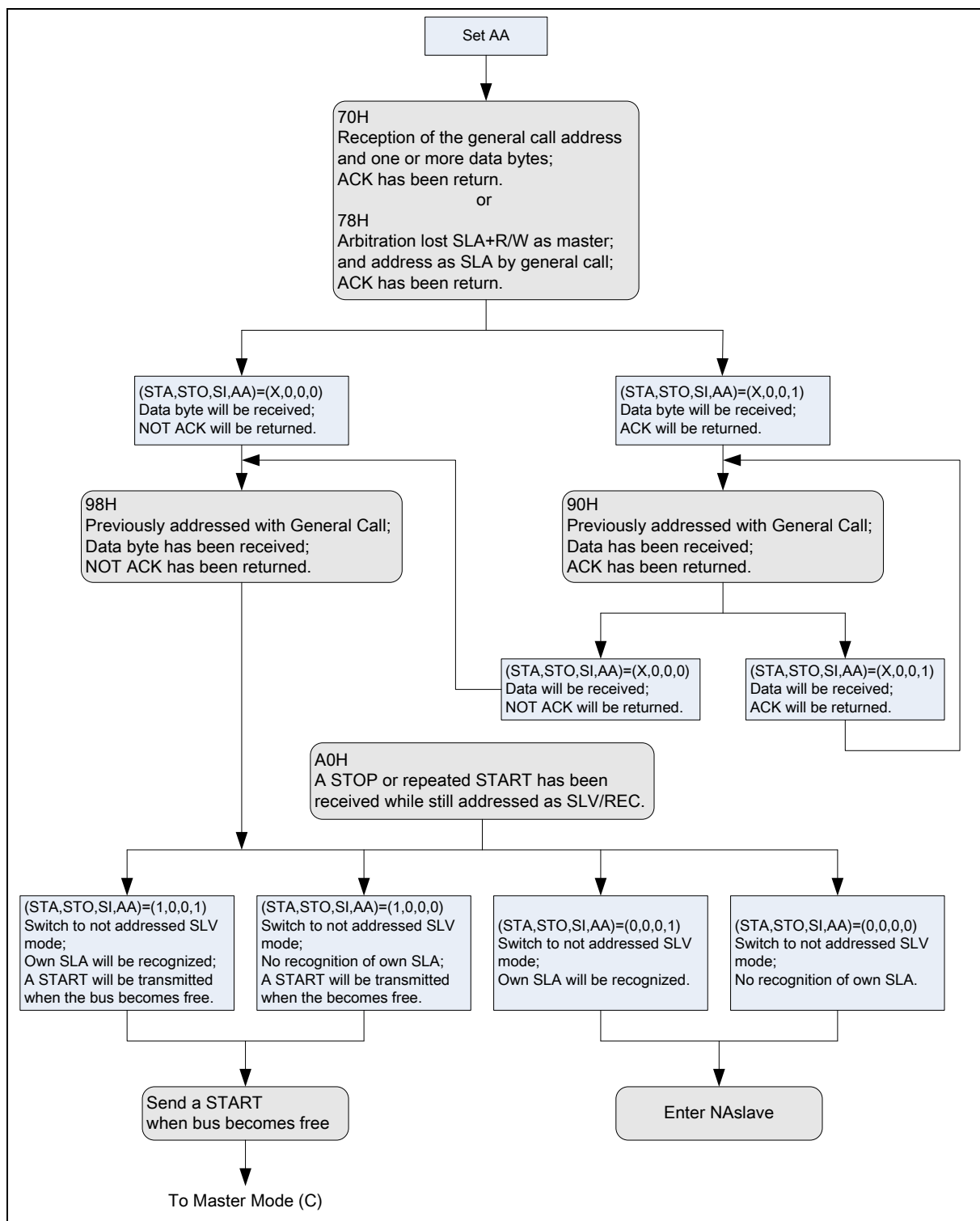


图 6.5-15 广播呼叫模式



## 6.6 PWM发生器和捕捉定时器

### 6.6.1 简介

NuMicro M051™ 系统有2个PWM组，支持4组PWM发生器，可配置成8个独立的PWM输出，PWM0~PWM7，或者4个互补的PWM对，(PWM0, PWM1), (PWM2, PWM3), (PWM4, PWM5) 和 (PWM6, PWM7)，带4个可编程的死区发生器。

每组PWM发生器带有8位预分频，一个时钟分频提供5种分频(1, 1/2, 1/4, 1/8, 1/16)，两个PWM定时器包括2个时钟选择，两个16位PWM向下计数计数器用于PWM周期控制，两个16位比较器用于PWM占空比占空比控制以及死区发生器。4组PWM发生器提供8个独立的PWM中断标志，这些中断标志当PWM向下计数周期达到零时由硬件置位。PWM发生器可以配置为单触发模式或自动重载模式连续输出PWM波形。

当PCR.DZEN01置位，PWM0 与 PWM1形成互补的PWM功能，这一对PWM的时序，周期，占空比周期占空比 和死区时间由PWM0定时器和死区发生器0决定。同样，PWM 互补对(PWM2, PWM3), (PWM4, PWM5) 与 (PWM6, PWM7) 分别由 PWM2, PWM4 与 PWM6 定时器和死区发生器2, 4, 6控制，参考下图查看PWM定时器内部结构。

为防止PWM输出不稳定波形，16位向下计数计数器和16位比较器采用双缓存器。当用户向计数器/比较器寄存器内写入值，只有当计数器/比较器的值计数到0后，才会被重新写入计数器/比较器。该双缓冲功能可以避免PWM输出时产生干扰波形。

当16位向下计数计数器达到0时，中断请求产生。如果PWM定时器被配置为连续模式，当向下计数器达到0时，会自动重新导入设定值(CNRx)并重新开始递减计数。如果定时器设为单触发模式，向下计数器停止计数，并产生中断请求。

比较器数据用于设定脉宽度，当向下计数器的值与比较寄存器的值相同时，计数器控制逻辑可以改变输出至高电平。

PWM定时器可复用为数字捕捉功能。输出模块的输入捕捉功能使能，PWM的输出引脚将切换至捕捉输入模式。捕捉器0和PWM0使用同一个定时器，捕捉器1和PWM1使用另一组定时器，以此类推。因此在使用捕捉功能之前，用户必须预先配置PWM定时器。捕捉功能使能后，在输入通道的上升沿将PWM计数器值锁存至CRLR，在输入通道的下降沿锁存PWM计数器值至CFLR捕捉通道0中断是可编程的，设定CCR0.CRL\_IE0[1] (上升沿锁存中断使能)和CCR0.CFL\_IE0[2] (下降沿锁存中断使能)，可以使捕捉器通道0作为中断源。同样设定CCR0.CRL\_IE1[17]和CCR0.CFL\_IE1[18]，可以设定通道1，通道2 和3同样通过设定CCR1[1],CCR1[2] 和CCR1[17], CCR1[18] 对于每一组，不管捕捉何时产生中断0/1/2/3，PWM计数器0/1/2/3都将在该时刻重载。

最大的捕捉频率受捕捉中断延迟限制。捕捉中断产生时，软件至少要执行三个步骤：读PIIRx 以得到中断源，读PWM\_CRLx/PWM\_CFLx(x=0 and 3) 以得到捕捉值，写1清PIIRx。如果中断延迟要花T0完成，捕捉信号不必在T0之间发送。这样最大的捕捉频率将为1/T0。例如：

HCLK = 50 MHz, PWM\_CLK = 25 MHz, 中断延迟时间 900 ns

因此最大的捕捉频率将是1/900ns ≈ 1000 kHz

## 6.6.2 特征

### 6.6.2.1 PWM功能特性:

PWM 组有两个PWM发生器. 每个PWM发生器支持一个8位的预分频, 一个时钟分频, 两个PWM定时器 (向下计数), 一个死区发生器和两个PWM输出.

- 最高16位分辨率
- PWM 中断与PWM周期同步
- 单触发模式或自动重载模式
- 2个PWM组 (PWMA/PWMB) 支持8个PWM通道

### 6.6.2.2 捕捉功能模块特性:

- 与PWM发生器共享时序控制逻辑
- 8 路捕捉输入通道与8个PWM输出通道复用
- 每个通道支持1个上升沿锁存寄存器(CRLR), 一个下降沿锁存寄存器 (CFLR) 和捕捉中断标志 (CAPIF<sub>x</sub>)

## 6.6.3 PWM 框图

下图为PWM对的结构图(Timer 0&1 为一对, timer 2&3为另外一对, 诸如此类.).

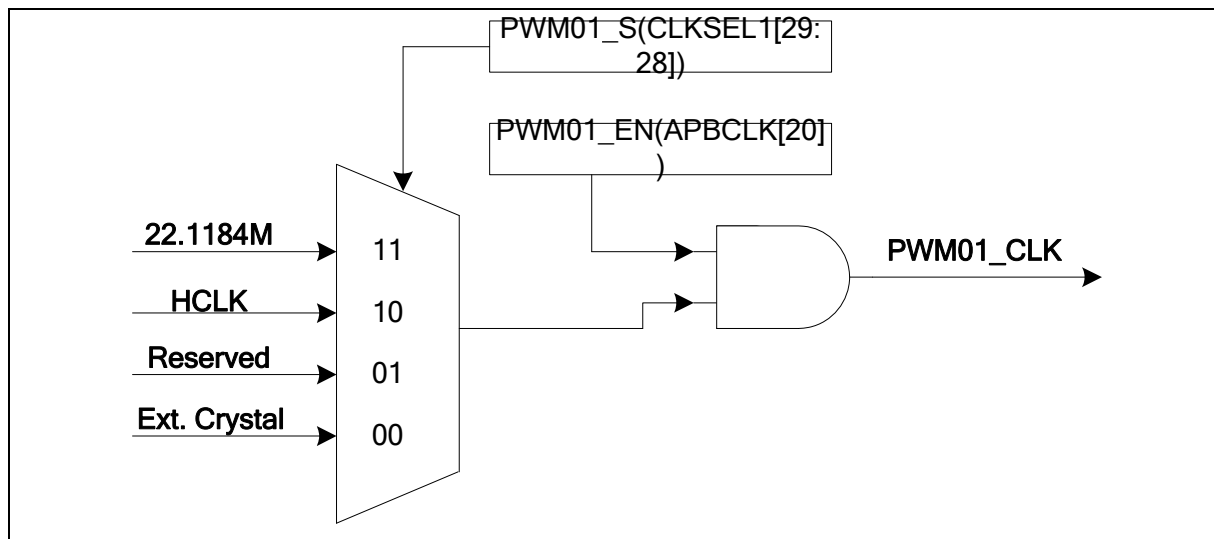


图 6.6-1 PWM 发生器 0 时钟源控制

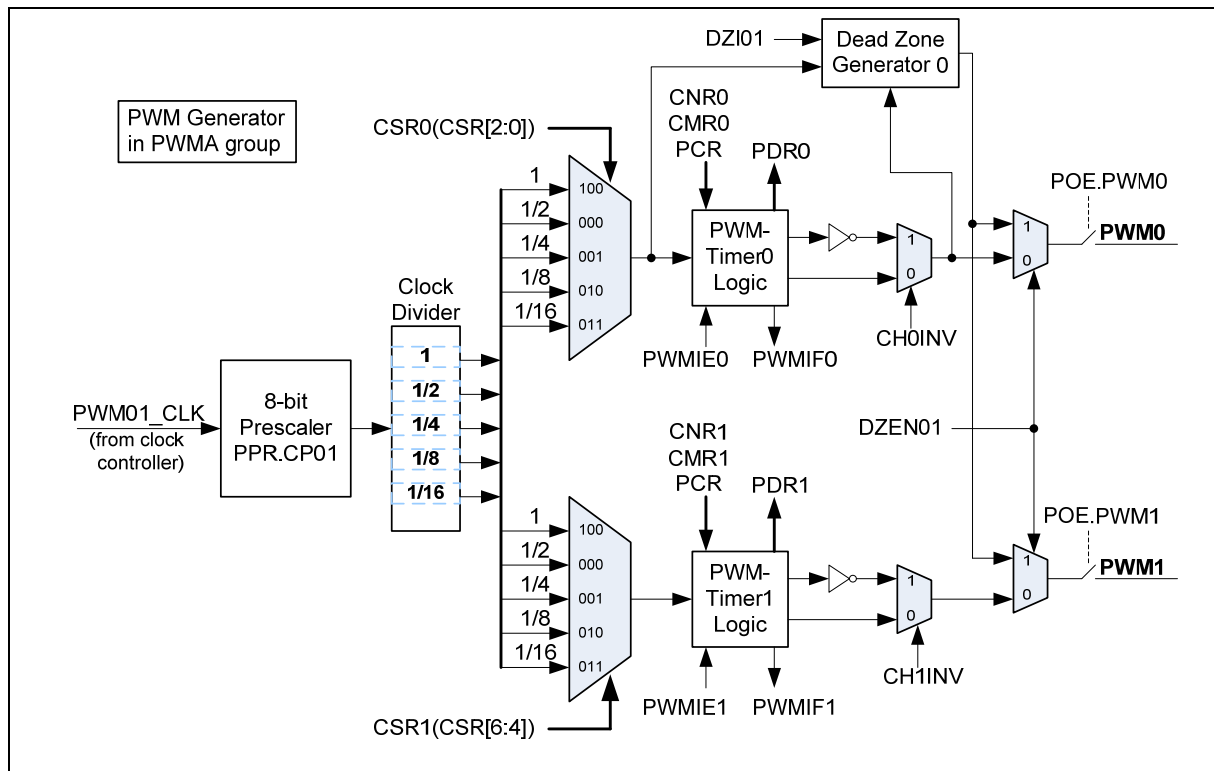


图 6.6-2 PWM 发生器 0 结构框图

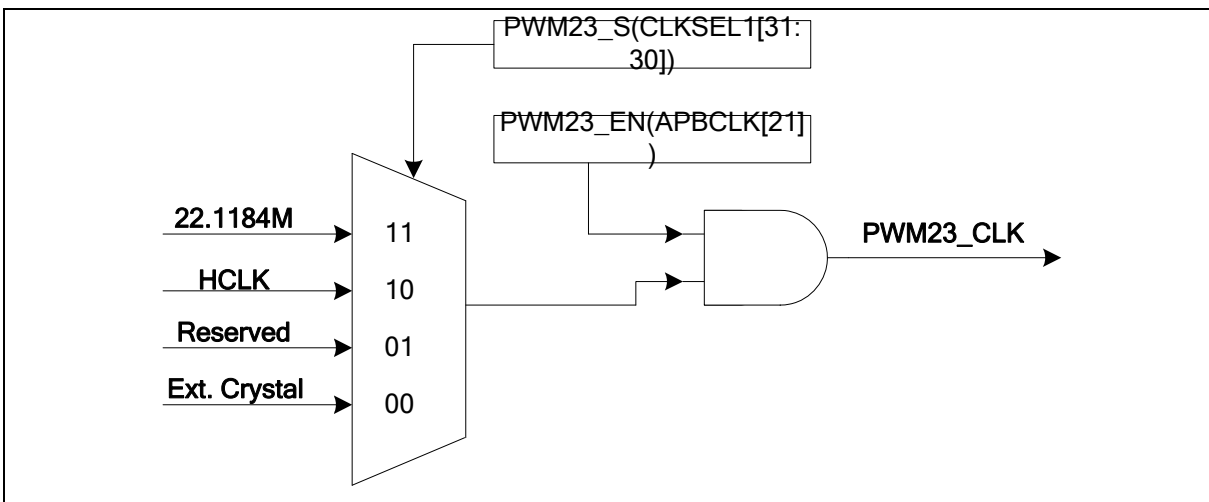


图 6.6-3 PWM 发生器 2 时钟源控制

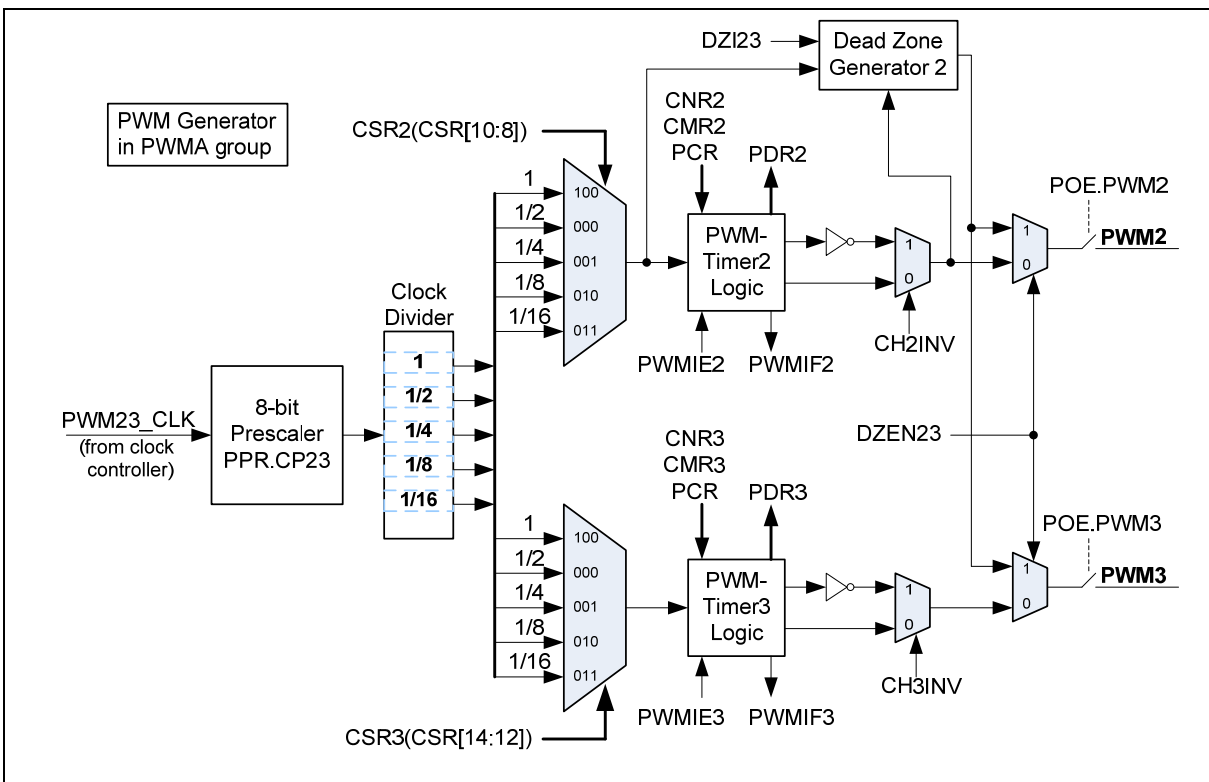


图 6.6-4 PWM 发生器 2 结构框图

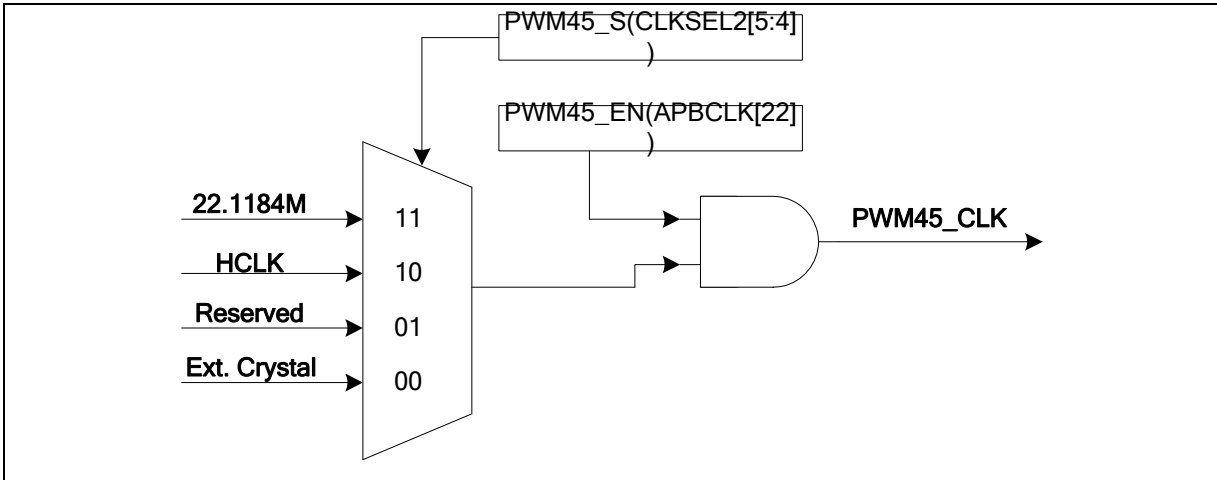


图 6.6-5 PWM 发生器 4 时钟源控制

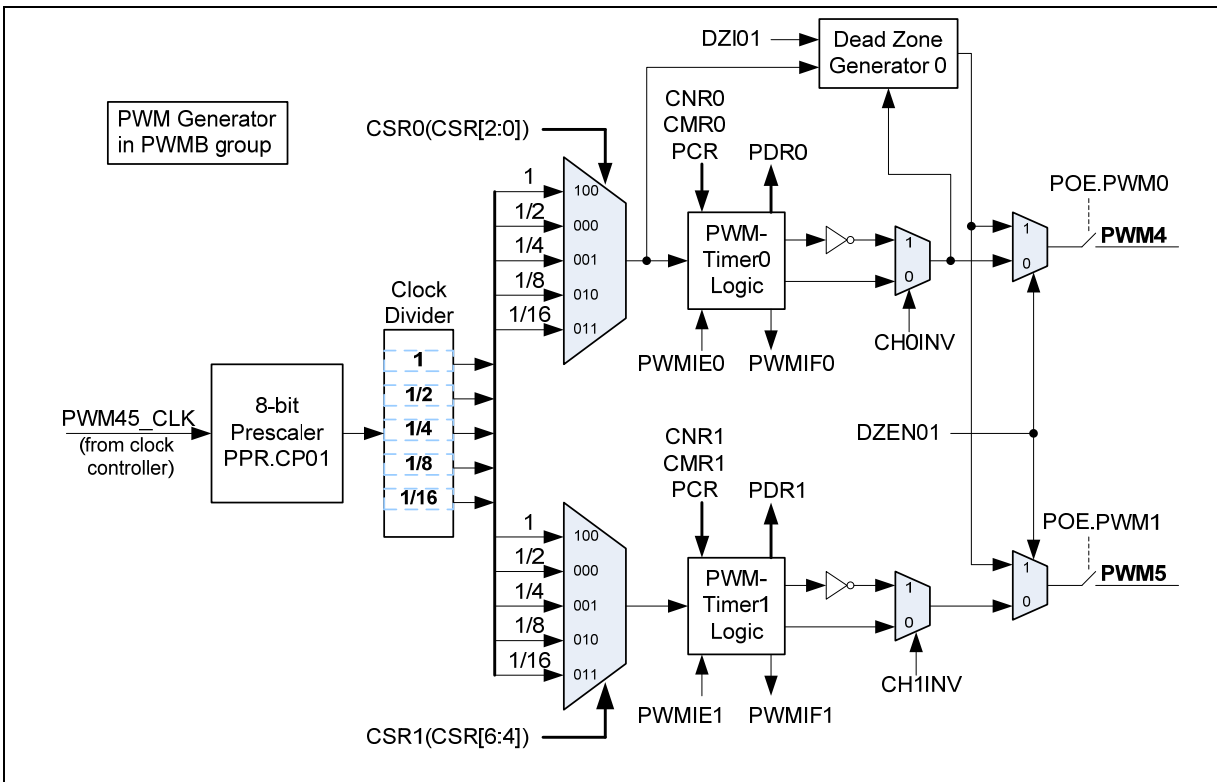


图 6.6-6 PWM 发生器 4 结构框图

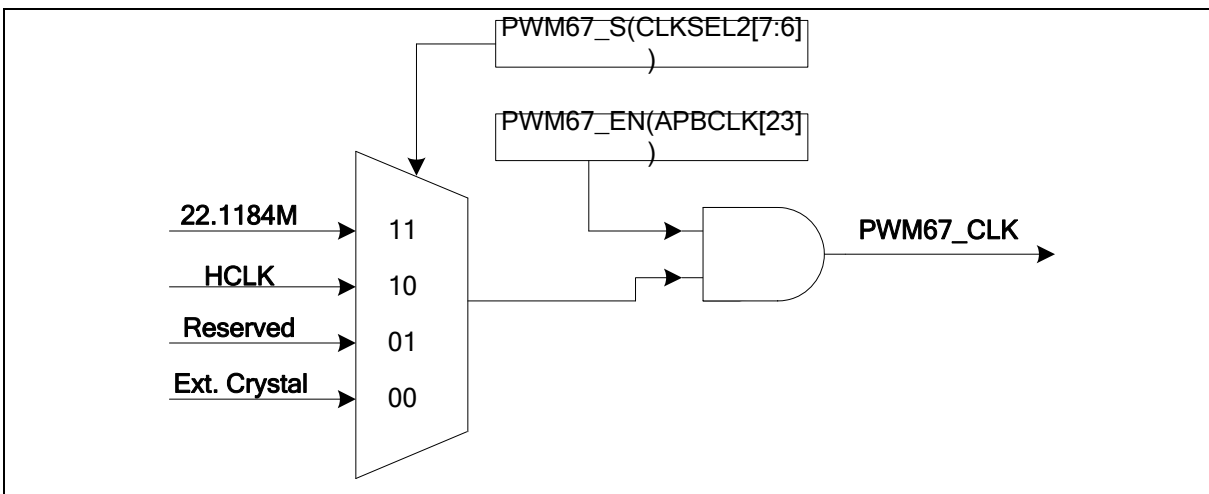


图 6.6-7 PWM 发生器 6 时钟源控制

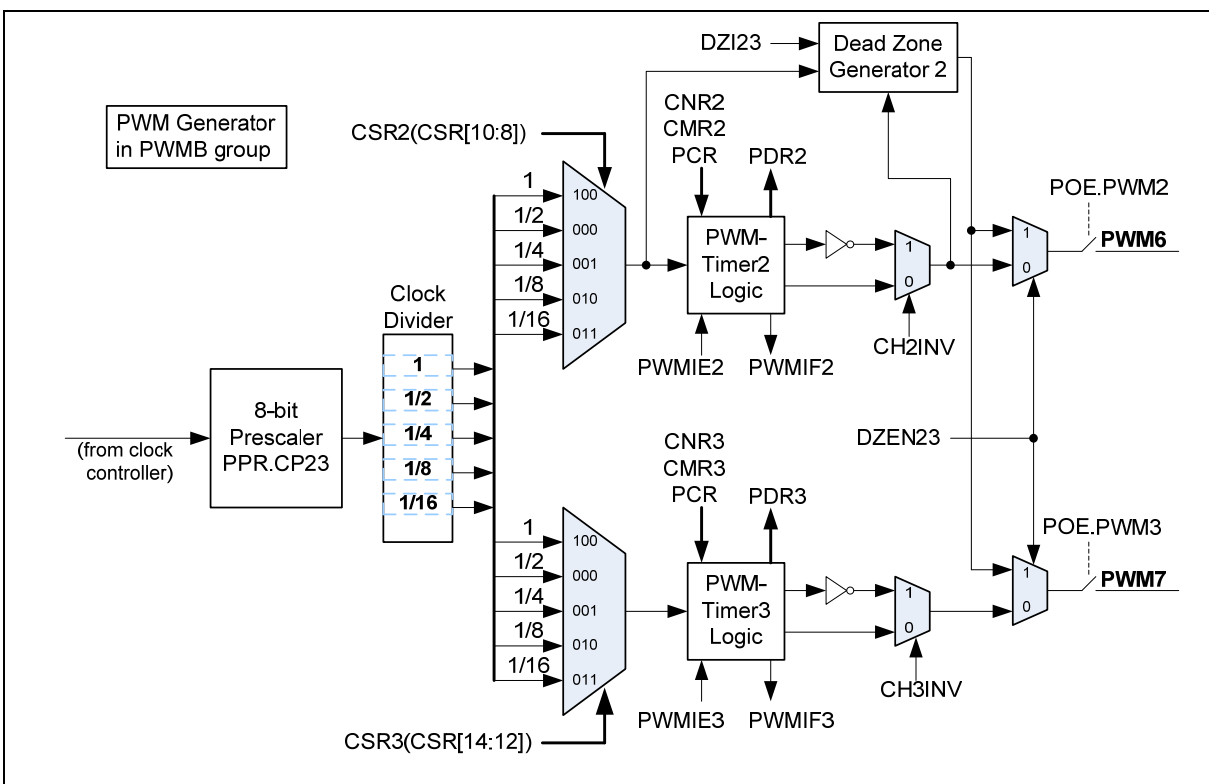


图 6.6-8 PWM 发生器 6 结构框图

## 6.6.4 PWM 功能描述

### 6.6.4.1 PWM-定时器操作

PWM 周期和占空比控制由向下计数的PWM寄存器(CNR)以及PWM比较寄存器(CMR)控制。PWM定时器工作时序如图6.6-10所示。脉宽调制的公式如下，相应的周期如图6.6-9所示。注意：相应的GPIO管脚必须配置成PWM功能(使能POE 和禁用CAPENR)。

PWM 频率 =  $\text{PWM}_{xy\_CLK} / (\text{prescale} + 1) * (\text{clock divider}) / (\text{CNR} + 1)$ ; xy代表01, 23, 45 或 67, 取决于所选择的PWM通道。

- 占空比 =  $(\text{CMR} + 1) / (\text{CNR} + 1)$
- $\text{CMR} \geq \text{CNR}$ : PWM输出为高
- $\text{CMR} < \text{CNR}$ : PWM低脉宽 =  $(\text{CNR} - \text{CMR}) \text{ unit}^1$ ; PWM高脉宽 =  $(\text{CMR} + 1) \text{ unit}$
- $\text{CMR} = 0$ : PWM 低脉宽 =  $(\text{CNR}) \text{ unit}$ ; PWM高脉宽 = 1 unit

注: 1. unit = 一个PWM时钟周期

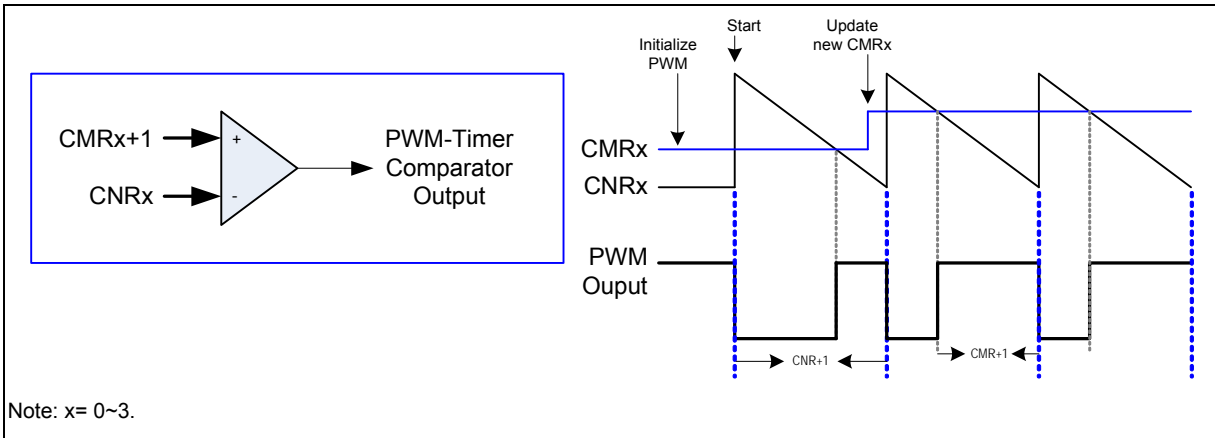


图 6.6-9 PWM 定时器内部比较器输出

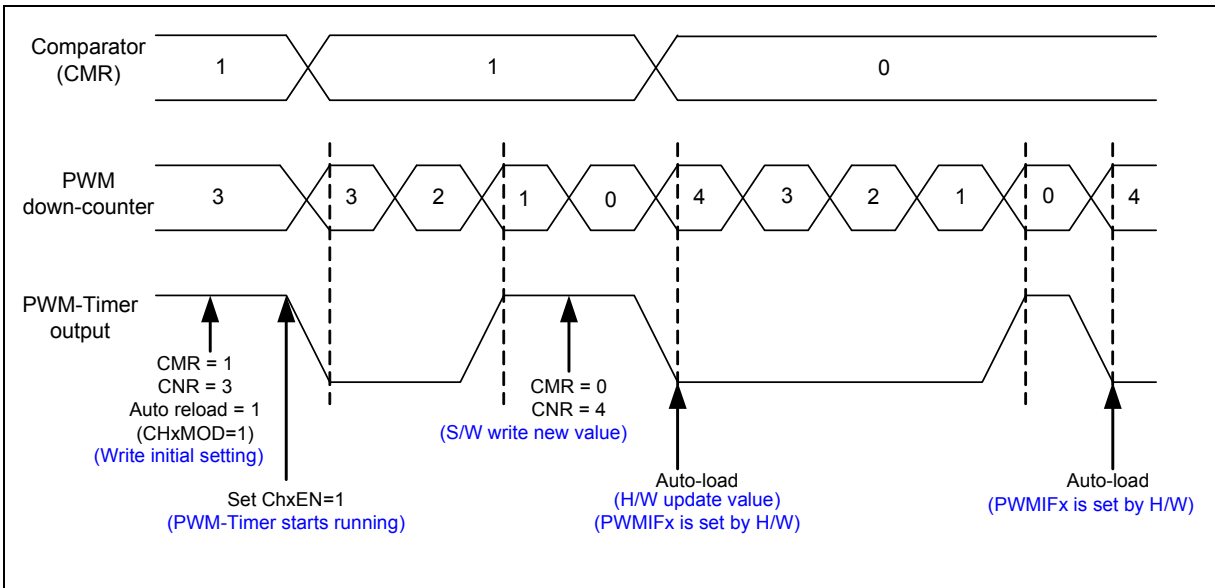


图 6.6-10 PWM 定时器操作时序

### 6.6.4.2 PWM双缓存, 自动重载以及单触发模式

NuMicro M051™ 系列PWM定时器具有双缓存功能。重载值将在下一个周期开始时更新，避免影响当前计数器工作。PWM计数器值可写入CNRx，并可从PDRx内读出。

PWM 控制寄存器(PCR)的CH0MOD 位定义PWM0是自动重载模式或是单触发模式。CH0MOD置1，自动重载模式下，当PWM计数器计到0，MCU自动重载CNR0 值到PWM 计数器。如果CNR0 设定为0，PWM计数器计数到0后，计数器将停止计数。如果此时CH0MOD设定为0，计数器立即停止计数。PWM1~PWM7 运行状态与PWM0 相同。

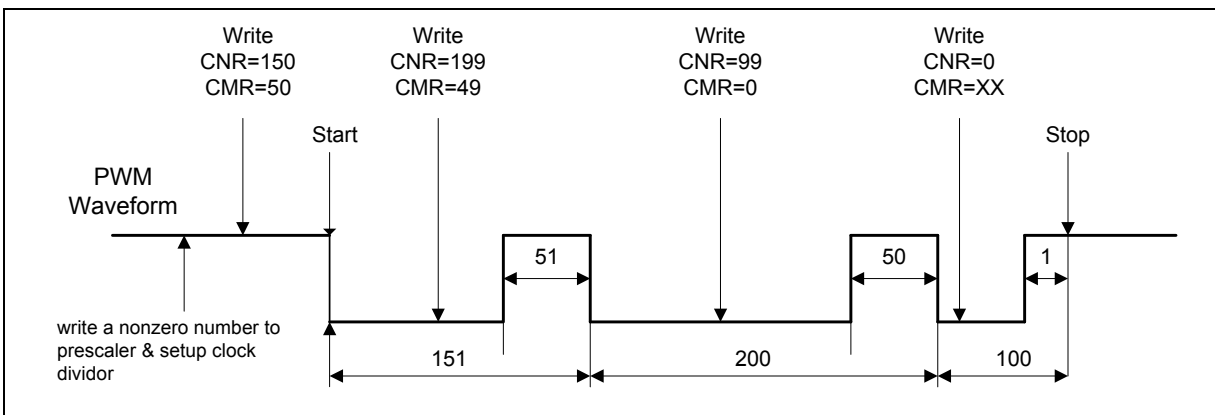


图 6.6-11 PWM 双缓存图解



### 6.6.4.3 可调占空比

双缓存允许CMRx自当前运行时被改写。写入值将在下个周期内生效。

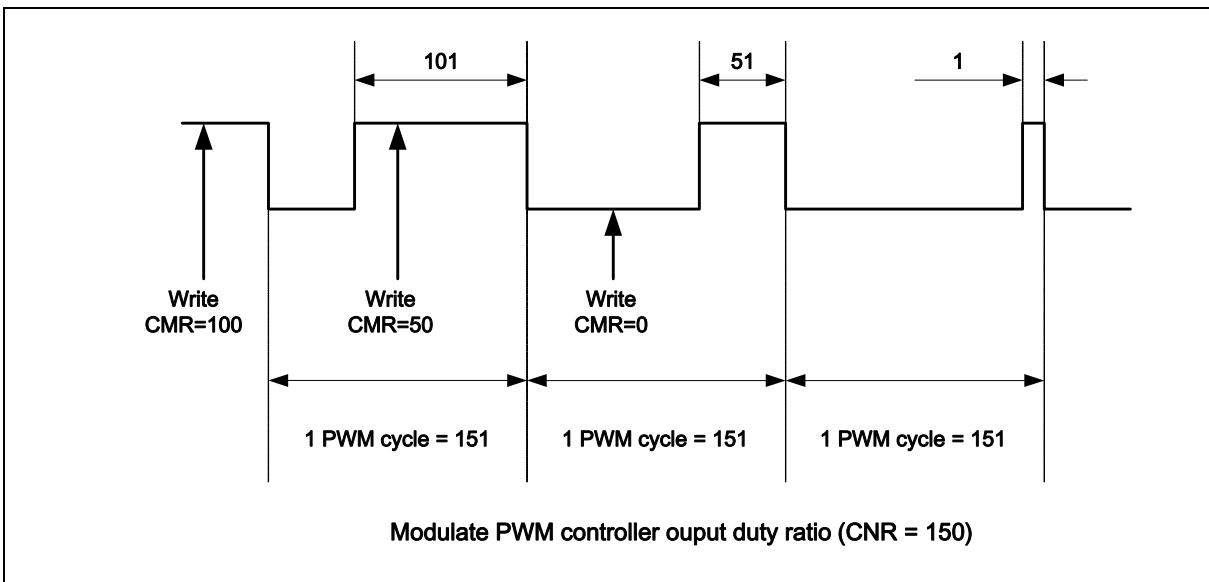


图 6.6-12 PWM 控制器输出占空比

### 6.6.4.4 死区发生器

NuMicro M051™ 系列提供PWM死区发生器，用于保护器件电源。该功能产生可编程的延迟时间到PWM上升沿输出，用户通过编程PPRx.DZI确定死区间隔。

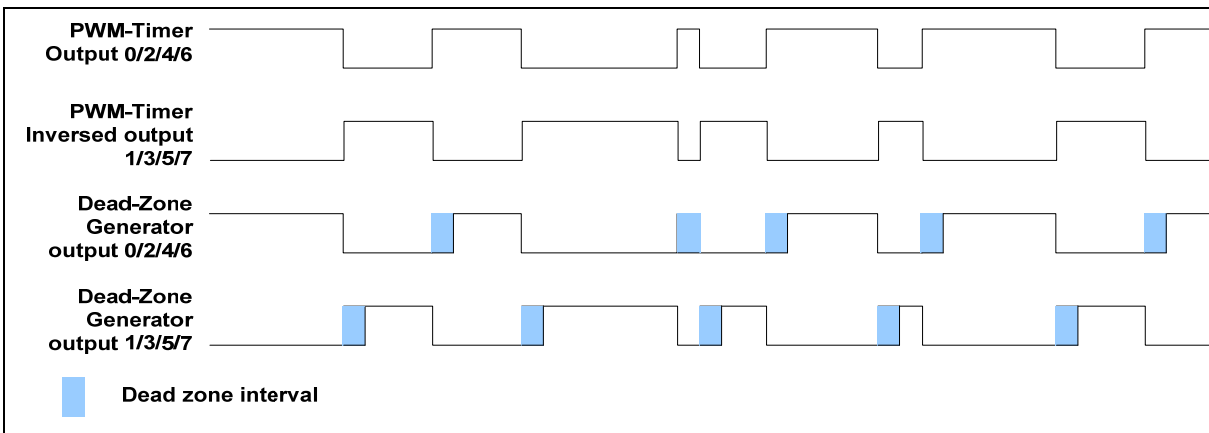


图 6.6-13 死区发生器操作

### 6.6.4.5 捕捉操作

捕捉器0和PWM0使用同一个定时器，捕捉器1和PWM1使用另一组定时器，以此类推。在使用捕捉功能

之前，必须预先配置PWM定时器。在输入信号的上升沿将PWM计数器的值锁定至CRLRx，在下降沿将PWM计数器的值锁定至CFLRx。捕捉通道0是可编程的，设定CCR0[1] (上升沿触发中断有效)和CCR0[2](下降沿触发中断有效),可以决定其中断发生条件。同样设定CCR0[17] 和CCR0[18]，可以设定通道1。通道2 和3同样通过设定CCR1[1],CCR1[2] 和CCR1[17], CCR1[18]来设定。无论捕捉模块何时触发中断，PWM计数器都将以CNRx值重置。注：相应的GPIO管脚必须配置成捕捉功能(禁用POE和使能CAPENR)。

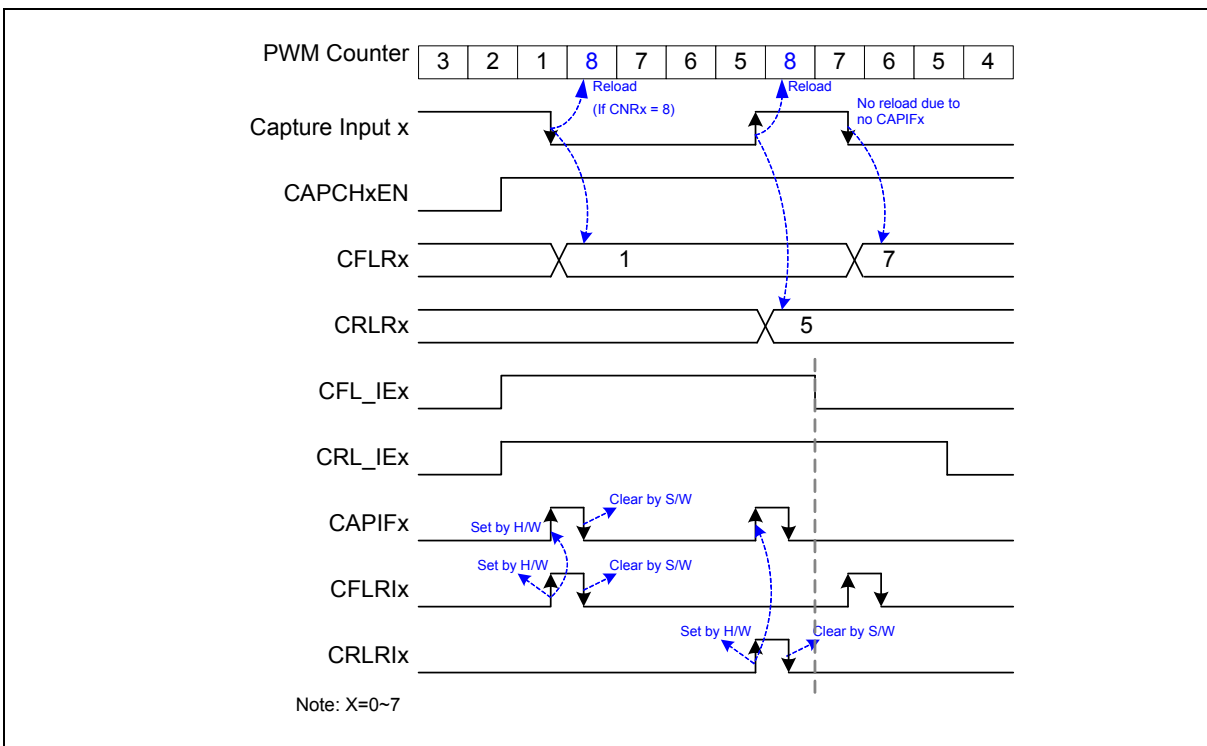


图 6.6-14 捕捉操作时序

在上述范例中，CNR 为8:

1. 捕捉中断标志(CAPIFx)置位时，PWM计数器将重装载CNRx.
2. 通道低脉冲宽为(CNR + 1 - CRLR).
3. 通道高脉冲宽为(CNR + 1 - CFLR).

#### 6.6.4.6 PWM-定时器中断结构

提供8个PWM中断，PWM0\_INT~PWM7\_INT，对于增强型中断控制寄存器(AIC)可分为PWMA\_INT 与PWMB\_INT。PWM 0 与捕捉器0共用同一组中断。PWM1 与捕捉器1 共用同一组中断，以此类推。因此，同一通道的PWM中断和捕捉中断不能同时发生。图 6.6-15和图6.6-16显示 PWM定时器中断结构如下。

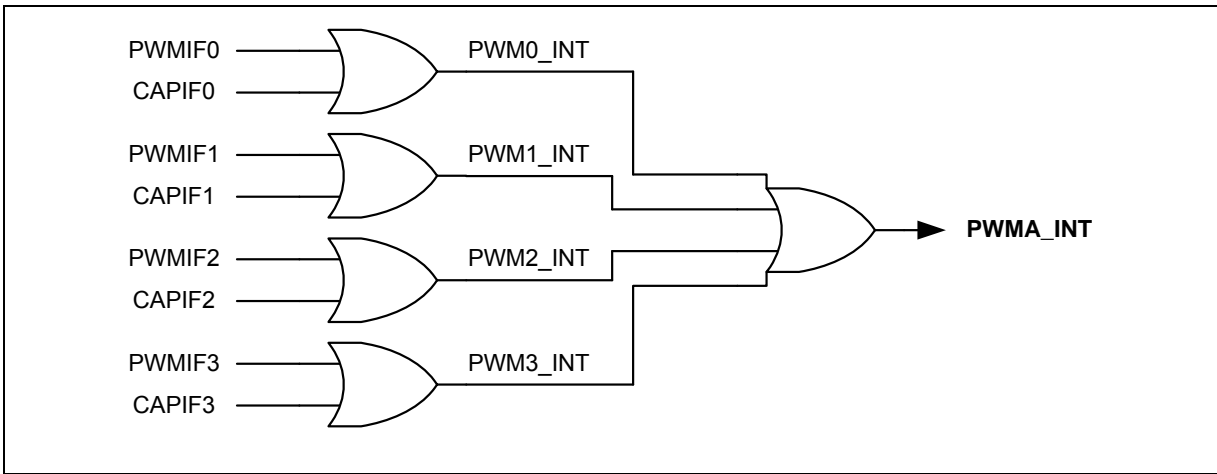


图 6.6-16 PWM A 组 PWM-定时器中断结构图

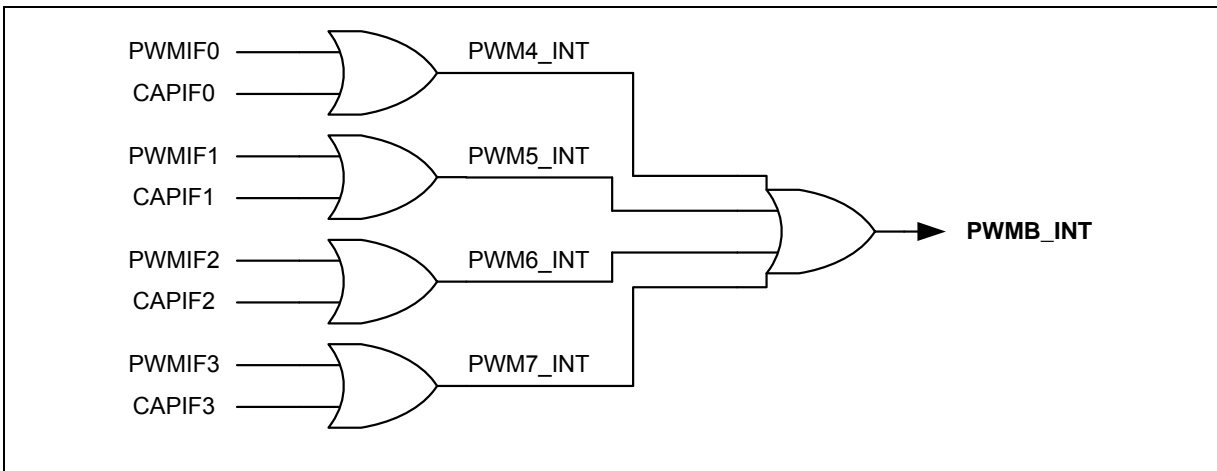


图 6.6-17 PWM B 组 PWM-定时器中断结构图

#### 6.6.4.7 PWM-定时器开启步骤

下列步骤用于启动PWM.

1. 配置时钟选择器(CSR)
2. 配置预分频器 (PPR)
3. 配置反向打开/关闭, 死区打开/关闭, 自动重载/单触发模式以及PWM定时器关闭 (PCR)
4. 配置比较器寄存器(CMR) 设定PWM占空比
5. 配置PWM计数器寄存器 (CNR) 设定PWM周期.
6. 配置中断使能寄存器(PIER)
7. 配置相应的GPIO管脚为PWM功能(使能 POE 和禁用CAPENR).
8. 使能PWM定时器开始运行 置PCR中的 CHxEN 为1

#### 6.6.4.8 PWM-定时器关闭步骤

方式 1:

设定16位向下计数计数器(CNR)为0, 并查看PDR (当前16位向下计数器的值) 状态. 当PDR达到0, 关闭PWM定时器 (PCR的CHxEN位). **(推荐)**

方式 2:

设定16位向下计数计数器(CNR)为0, 当中断条件发生, 在中断内关闭PWM定时器(PCR的CHxEN位). **(推荐)**

方式 3:

直接关闭PWM定时器(PCR的CHxEN位). (不推荐)

不推荐方式3的原因是: 禁用CHxEN将立即停止PWM输出信号, 引起PWM输出占空比的改变, 这可能导致电机电路控制异常.

## 6.6.4.9 捕捉开始步骤

- 配置时钟选择器(CSR)
- 配置预分频器(PPR)
- 配置通道使能, 上升/下降沿中断使能以及输入信号反向打开/关闭 (CCR0, CCR1)
- 配置PWM计数器寄存器(CNR)
- 配置相应GPIO管脚为捕捉功能(禁用 POE和使能 CAPENR).
- 使能PWM定时器 (置PCR中的 CHxEN 为1)

## 6.6.5 PWM 控制器寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>PWMA_BA = 0x4004_0000 (PWMA组)</b>				
<b>PWMB_BA = 0x4014_0000 (PWMB组)</b>				
PPR	PWMA_BA+0x00	R/W	PWMA组预分频寄存器	0x0000_0000
	PWMB_BA+0x00	R/W	PWMB组 预分频寄存器	0x0000_0000
CSR	PWMA_BA+0x04	R/W	PWMA组时钟选择寄存器	0x0000_0000
	PWMB_BA+0x04	R/W	PWMB组时钟选择寄存器	0x0000_0000
PCR	PWMA_BA+0x08	R/W	PWMA组控制寄存器	0x0000_0000
	PWMB_BA+0x08	R/W	PWMB组控制寄存器	0x0000_0000
CNR0	PWMA_BA+0x0C	R/W	PWMA组计数寄存器0	0x0000_0000
	PWMB_BA+0x0C	R/W	PWMB组计数寄存器0	0x0000_0000
CMR0	PWMA_BA+0x10	R/W	PWMA组比较寄存器0	0x0000_0000
	PWMB_BA+0x10	R/W	PWMB组比较寄存器0	0x0000_0000
PDR0	PWMA_BA+0x14	R	PWMA组数据寄存器0	0x0000_0000
	PWMB_BA+0x14	R	PWMB组数据寄存器0	0x0000_0000
CNR1	PWMA_BA+0x18	R/W	PWMA组计数寄存器1	0x0000_0000
	PWMB_BA+0x18	R/W	PWMB组计数寄存器1	0x0000_0000
CMR1	PWMA_BA+0x1C	R/W	PWMA组比较寄存器1	0x0000_0000
	PWMB_BA+0x1C	R/W	PWMB组比较寄存器1	0x0000_0000
PDR1	PWMA_BA+0x20	R	PWMA组数据寄存器1	0x0000_0000
	PWMB_BA+0x20	R	PWMB组数据寄存器1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWMA组计数寄存器2	0x0000_0000
	PWMB_BA+0x24	R/W	PWMB组计数寄存器2	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWMA组 比较寄存器2	0x0000_0000
	PWMB_BA+0x28	R/W	PWMB组 比较寄存器 2	0x0000_0000

<b>PDR2</b>	PWMA_BA+0x2C	R	PWMA组 数据寄存器 2	0x0000_0000
	PWMB_BA+0x2C	R	PWMB组数据寄存器 2	0x0000_0000
<b>CNR3</b>	PWMA_BA+0x30	R/W	PWMA组 计数寄存器 3	0x0000_0000
	PWMB_BA+0x30	R/W	PWMB组 计数寄存器3	0x0000_0000
<b>CMR3</b>	PWMA_BA+0x34	R/W	PWMA组比较寄存器3	0x0000_0000
	PWMB_BA+0x34	R/W	PWMB组比较寄存器3	0x0000_0000
<b>PDR3</b>	PWMA_BA+0x38	R	PWMA组数据寄存器3	0x0000_0000
	PWMB_BA+0x38	R	PWMB组数据寄存器3	0x0000_0000
<b>PIER</b>	PWMA_BA+0x40	R/W	PWMA组中断使能寄存器	0x0000_0000
	PWMB_BA+0x40	R/W	PWMB组中断使能寄存器	0x0000_0000
<b>PIIR</b>	PWMA_BA+0x44	R/C	PWMA组中断标志寄存器	0x0000_0000
	PWMB_BA+0x44	R/C	PWMB组中断标志寄存器	0x0000_0000
<b>CCR0</b>	PWMA_BA+0x50	R/W	PWMA组 捕捉控制寄存器 0	0x0000_0000
	PWMB_BA+0x50	R/W	PWMB组捕捉控制寄存器0	0x0000_0000
<b>CCR2</b>	PWMA_BA+0x54	R/W	PWMA组捕捉控制寄存器2	0x0000_0000
	PWMB_BA+0x54	R/W	PWMB组捕捉控制寄存器2	0x0000_0000
<b>CRLR0</b>	PWMA_BA+0x58	R/W	PWMA组捕捉上升沿锁存寄存器(Channel 0)	0x0000_0000
	PWMB_BA+0x58	R/W	PWMB组捕捉上升沿锁存寄存器(Channel 0)	0x0000_0000
<b>CFLR0</b>	PWMA_BA+0x5C	R/W	PWMA组捕捉下降沿锁存寄存器(Channel 0)	0x0000_0000
	PWMB_BA+0x5C	R/W	PWMB组捕捉下降沿锁存寄存器(Channel 0)	0x0000_0000
<b>CRLR1</b>	PWMA_BA+0x60	R/W	PWMA组捕捉上升沿锁存寄存器(Channel 1)	0x0000_0000
	PWMB_BA+0x60	R/W	PWMB组捕捉上升沿锁存寄存器(Channel 1)	0x0000_0000
<b>CFLR1</b>	PWMA_BA+0x64	R/W	PWMA组捕捉下降沿锁存寄存器(Channel 1)	0x0000_0000
	PWMB_BA+0x64	R/W	PWMB组捕捉下降沿锁存寄存器(Channel 1)	0x0000_0000
<b>CRLR2</b>	PWMA_BA+0x68	R/W	PWMA组捕捉上升沿锁存寄存器(Channel 2)	0x0000_0000
	PWMB_BA+0x68	R/W	PWMB组捕捉上升沿锁存寄存器(Channel 2)	0x0000_0000
<b>CFLR2</b>	PWMA_BA+0x6C	R/W	PWMA组捕捉下降沿锁存寄存器(Channel 2)	0x0000_0000
	PWMB_BA+0x6C	R/W	PWMB组捕捉下降沿锁存寄存器(Channel 2)	0x0000_0000

<b>CRLR3</b>	PWMA_BA+0x70	R/W	PWMA组捕捉上升沿锁存寄存器(Channel 3)	0x0000_0000
	PWMB_BA+0x70	R/W	PWMB组捕捉上升沿锁存寄存器(Channel 3)	0x0000_0000
<b>CFLR3</b>	PWMA_BA+0x74	R/W	PWMA组捕捉下降沿锁存寄存器(Channel 3)	0x0000_0000
	PWMB_BA+0x74	R/W	PWMB组捕捉下降沿锁存寄存器(Channel 3)	0x0000_0000
<b>CAPENR</b>	PWMA_BA+0x78	R/W	PWMA组 捕捉输入 0~3 使能寄存器	0x0000_0000
	PWMB_BA+0x78	R/W	PWMB组捕捉输入 0~3 使能寄存器	0x0000_0000
<b>POE</b>	PWMA_BA+0x7C	R/W	PWMA组 通道0~3输出使能	0x0000_0000
	PWMB_BA+0x7C	R/W	PWMB组通道0~3输出使能	0x0000_0000



## 6.6.6 PWM 控制器寄存器描述

### PWM 预分频寄存器 (PPR)

寄存器	偏移量	R/W	描述	复位后的值
PPR	PWMA_BA+0x00	R/W	PWM A组预分频寄存器	0x0000_0000
	PWMB_BA+0x00	R/W	PWM B组预分频寄存器	0x0000_0000

31	30	29	28	27	26	25	24
DZI23							
23	22	21	20	19	18	17	16
DZI01							
15	14	13	12	11	10	9	8
CP23							
7	6	5	4	3	2	1	0
CP01							

Bits	描述	
[31:24]	DZI23	<p><b>PWM2 与 PWM3的死区间隔寄存器</b>(PWM2与PWM3对应于PWMA组, PWM6与PWM7对应于PWMB组)</p> <p>该8位寄存器决定死区长度.</p> <p>每单位死区时间长度由相应的CSR位决定.</p>
[23:16]	DZI01	<p><b>PWM0 与 PWM1的死区间隔寄存器</b>(PWM0与PWM1对应于PWMA组, PWM4与PWM5对应于PWMB组)</p> <p>该8位寄存器决定死区长度.</p> <p>每单位死区时间长度由相应的CSR位决定.</p>
[15:8]	CP23	<p><b>PWM定时器2 &amp; 3的时钟预分频 2</b> (PWM counter 2 &amp; 3对应于A组与PWM counter 6 &amp; 7对应于B组)</p> <p>时钟输入相应PWM计数器之前, 根据(CP23 + 1)分频</p> <p>如果CP23=0, 预分频器2输出时钟停止。PWM 计数器2和3也停止.</p>
[7:0]	CP01	<p><b>PWM定时器0 &amp; 1的时钟预分频0</b>(PWM counter 0 &amp; 1对应于A组与PWM counter 4 &amp; 5对应于B组)</p>

		时钟输入相应PWM计数器之前，根据(CP01 + 1)分频 如果CP01=0, 预分频器0输出时钟停止。PWM计数器0和1也停止.
--	--	--



## PWM 时钟选择寄存器(CSR)

寄存器	偏移量	R/W	描述	复位后的值
CSR	PWMA_BA+0x04	R/W	PWM A组时钟选择寄存器	0x0000_0000
	PWMB_BA+0x04	R/W	PWM B组时钟选择寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留	CSR3			保留	CSR2		
7	6	5	4	3	2	1	0
保留	CSR1			保留	CSR0		

Bits	描述													
[31:15]	保留	保留												
[14:12]	CSR3	定时器 3 时钟源选择(PWM定时器3对应于A组 and PWM定时器7对应于B组) 为PWM定时器选择时钟输入.												
		<table border="1" style="width: 100%;"> <thead> <tr> <th>CSR3 [14:12]</th> <th>输入时钟分频</th> </tr> </thead> <tbody> <tr> <td>100</td> <td>1</td> </tr> <tr> <td>011</td> <td>16</td> </tr> <tr> <td>010</td> <td>8</td> </tr> <tr> <td>001</td> <td>4</td> </tr> <tr> <td>000</td> <td>2</td> </tr> </tbody> </table>	CSR3 [14:12]	输入时钟分频	100	1	011	16	010	8	001	4	000	2
		CSR3 [14:12]	输入时钟分频											
		100	1											
		011	16											
		010	8											
001	4													
000	2													
[11]	保留	保留												
[10:8]	CSR2	定时器 2 时钟源选择(PWM定时器2 对应于A组 and PWM定时器6 对应于B组) 为PWM定时器选择时钟输入. (表格同CSR3)												

[7]	保留	保留
[6:4]	CSR1	定时器 1 时钟源选择(PWM定时器1 对应于A组 and PWM定时器5 对应于B组) 为PWM定时器选择时钟输入. (表格同CSR3)
[3]	保留	保留
[2:0]	CSR0	定时器 0 时钟源选择(PWM定时器0 对应于A组 and PWM定时器4 对应于B组) 为PWM定时器选择时钟输入. (表格同CSR3)

## PWM 控制寄存器(PCR)

寄存器	偏移量	R/W	描述	复位后的值
PCR	PWMA_BA+0x08	R/W	PWM A组控制寄存器 (PCR)	0x0000_0000
	PWMB_BA+0x08	R/W	PWM B组控制寄存器(PCR)	0x0000_0000

31	30	29	28	27	26	25	24
保留				CH3MOD	CH3INV	保留	CH3EN
23	22	21	20	19	18	17	16
保留				CH2MOD	CH2INV	保留	CH2EN
15	14	13	12	11	10	9	8
保留				CH1MOD	CH1INV	保留	CH1EN
7	6	5	4	3	2	1	0
保留		DZEN23	DZEN01	CH0MOD	CH0INV	保留	CH0EN

Bits	描述	
[31:28]	保留	保留
[27]	CH3MOD	<p><b>PWM-定时器3 自动重载/单触发模式选择</b>(PWM定时器3 对应于A组 and PWM定时器7 对应于B组)</p> <p>1 = 自动重载模式</p> <p>0 = 单触发模式</p> <p>注: 如果该位由0置1, 会使CNR3 和CMR3 清位</p>
[26]	CH3INV	<p><b>PWM定时器3反向打开/关闭</b>(PWM定时器3 对应于A组 and PWM定时器7 对应于B组)</p> <p>1 = 反向打开</p> <p>0 = 反向关闭</p>
[25]	保留	保留
[24]	CH3EN	<p><b>PWM定时器3 使能/禁用</b>(PWM定时器3 对应于A组 and PWM定时器7 对应于B组)</p> <p>1 = 使能相应PWM定时器开始运行</p> <p>0 = 停止相应PWM定时器运行</p>

[23:20]	保留	保留
[19]	<b>CH2MOD</b>	<b>PWM-定时器2 自动重载/单触发模式选择</b> (PWM定时器2 对应于A组 and PWM定时器6 对应于B组) 1 = 自动重载模式 0 = 单触发模式 注: 如果该位由0置1, 会使CNR2和CMR2清位.
[18]	<b>CH2INV</b>	<b>PWM-定时器2反向打开/关闭</b> (PWM定时器2 对应于A组 and PWM定时器6 对应于B组) 1 = 反向打开 0 = 反向关闭
[17]	保留	保留
[16]	<b>CH2EN</b>	<b>PWM定时器2 使能/禁用</b> (PWM定时器2 对应于A组 and PWM定时器6 对应于B组) 1 = 使能相应PWM定时器开始运行 0 = 停止相应PWM定时器运行
[15:12]	保留	保留
[11]	<b>CH1MOD</b>	<b>PWM定时器1 自动重载/单触发模式选择</b> (PWM定时器1 对应于A组 and PWM定时器5 对应于B组) 1 = 自动重载模式 0 = 单触发模式 注: 如果该位由0置1, 会使CNR1和CMR1清位.
[10]	<b>CH1INV</b>	<b>PWM-定时器1反向打开/关闭</b> (PWM定时器1 对应于A组 and PWM定时器5 对应于B组) 1 = 反向打开 0 = 反向关闭
[9]	保留	保留
[8]	<b>CH1EN</b>	<b>PWM定时器1 使能/禁用</b> (PWM定时器1 对应于A组 and PWM定时器5 对应于B组) 1 = 使能相应PWM定时器开始运行 0 = 停止相应PWM定时器运行
[7:6]	保留	保留
[5]	<b>DZEN23</b>	<b>死区发生器2使能/禁用</b> (PWM2 and PWM3 pair 对应于 PWMA组, PWM6 and PWM7 pair 对应于 PWMB组) 1 = 使能

		<p>0 = 禁用</p> <p>注: 当死区发生器使能, PWM A组的PWM2与PWM3将成为互补对, PWM B组的PWM6与PWM7将成为互补对.</p>
[4]	<b>DZEN01</b>	<p>死区发生器<b>0</b>使能/禁用(PWM0 and PWM1 pair 对应于 PWMA组, PWM4 and PWM5 pair 对应于 PWMB组)</p> <p>1 = 使能</p> <p>0 = 禁用</p> <p>注: 当死区发生器使能, PWM A组的PWM0与PWM1将成为互补对, PWM B组的PWM4与PWM5将成为互补对.</p>
[3]	<b>CH0MOD</b>	<p><b>PWM-定时器 0 自动加载/单触发模式选择</b> (PWM定时器0 对应于A组 and PWM定时器4 对应于B组)</p> <p>1 = 自动重载模式</p> <p>0 = 单触发模式</p> <p>注: 如果该位由0置1, 会使CNR0和CMR0清位.</p>
[2]	<b>CH0INV</b>	<p><b>PWM-定时器0反向打开/关闭</b> (PWM定时器0 对应于A组 and PWM定时器4 对应于B组)</p> <p>1 = 反向打开</p> <p>0 = 反向关闭</p>
[1]	保留	保留
[0]	<b>CH0EN</b>	<p><b>PWM-定时器0 使能/禁用</b> (PWM定时器0 对应于A组 and PWM定时器4 对应于B组)</p> <p>1 = 使能相应PWM定时器开始运行</p> <p>0 = 停止相应PWM定时器运行</p>



## PWM 计数器寄存器3-0 (CNR3-0)

寄存器	偏移量	R/W	描述	复位后的值
CNR0	PWMA_BA+0x0C	R/W	PWM A组计数器寄存器0	0x0000_0000
	PWMB_BA+0x0C	R/W	PWM B组计数器寄存器0	0x0000_0000
CNR1	PWMA_BA+0x18	R/W	PWM A组计数器寄存器1	0x0000_0000
	PWMB_BA+0x18	R/W	PWM B组计数器寄存器1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWM A组计数器寄存器2	0x0000_0000
	PWMB_BA+0x24	R/W	PWM B组计数器寄存器2	0x0000_0000
CNR3	PWMA_BA+0x30	R/W	PWM A组计数器寄存器3	0x0000_0000
	PWMB_BA+0x30	R/W	PWM B组计数器寄存器3	0x0000_0000

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

Bits	描述	
[31:16]	保留	保留
[15:0]	CNRx	<p><b>PWM 计数器/定时器载入值</b></p> <p>CNR 决定PWM的周期.</p> <ul style="list-style-type: none"> <li>● PWM 频率 = <math>PWM_{xy\_CLK} / (\text{prescale} + 1) * (\text{clock divider}) / (\text{CNR} + 1)</math>; xy代表 01, 23, 45或 67, 取决于所选择的PWM通道.</li> <li>● 占空比 = <math>(\text{CMR} + 1) / (\text{CNR} + 1)</math>.</li> <li>● <math>\text{CMR} \geq \text{CNR}</math>: PWM输出高.</li> <li>● <math>\text{CMR} &lt; \text{CNR}</math>: PWM低脉冲宽度 = <math>(\text{CNR} - \text{CMR})</math> unit; PWM高脉冲宽度 =</li> </ul>



		<p>(CMR+1) unit.</p> <ul style="list-style-type: none"><li>● CMR = 0: PWM脉冲宽度 = (CNR) unit; PWM脉冲宽度 = 1 unit</li></ul> <p>(Unit = one PWM clock cycle)</p> <p>注: CNR写入数据后将在下一个PWM周期生效.</p>
--	--	--

## PWM 比较器寄存器 3-0 (CMR3-0)

寄存器	偏移量	R/W	描述	复位后的值
CMR0	PWMA_BA+0x10	R/W	PWM A组比较器寄存器0	0x0000_0000
	PWMB_BA+0x10	R/W	PWM B组比较器寄存器0	0x0000_0000
CMR1	PWMA_BA+0x1C	R/W	PWM A组比较器寄存器1	0x0000_0000
	PWMB_BA+0x1C	R/W	PWM B组比较器寄存器1	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWM A组比较器寄存器2	0x0000_0000
	PWMB_BA+0x28	R/W	PWM B组比较器寄存器2	0x0000_0000
CMR3	PWMA_BA+0x34	R/W	PWM A组比较器寄存器3	0x0000_0000
	PWMB_BA+0x34	R/W	PWM B组比较器寄存器3	0x0000_0000

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

Bits	描述	
[31:16]	保留	保留
[15:0]	CMRx	<p><b>PWM 比较器寄存器</b></p> <p>CMR 决定 PWM 的占空比.</p> <ul style="list-style-type: none"> <li>● PWM 频率 = <math>PWM_{xy\_CLK} / (\text{prescale} + 1) * (\text{clock divider}) / (\text{CNR} + 1)</math>; xy代表 01, 23, 45 or 67, 取决于所选择的PWM通道.</li> <li>● 占空比 = <math>(\text{CMR} + 1) / (\text{CNR} + 1)</math>.</li> <li>● <math>\text{CMR} \geq \text{CNR}</math>: PWM 输出高.</li> <li>● <math>\text{CMR} &lt; \text{CNR}</math>: PWM低脉冲宽度 = <math>(\text{CNR} - \text{CMR})</math> unit; PWM高脉冲宽度 = <math>(\text{CMR} + 1)</math> unit.</li> </ul>

		<ul style="list-style-type: none"><li>● CMR = 0: PWM低脉冲宽度= (CNR) unit; PWM 高脉冲宽度 = 1 unit (Unit = 一个PWM时钟周期)</li></ul> <p>Note: CNR写入数据后将在下一个PWM周期生效.</p>
--	--	---



### PWM 数据寄存器 3-0 (PDR 3-0)

寄存器	偏移量	R/W	描述	复位后的值
PDR0	PWMA_BA0+0x14	R	PWM A组数据寄存器0	0x0000_0000
	PWMB_BA0+0x14	R	PWMB组数据寄存器0	0x0000_0000
PDR1	PWMA_BA0+0x20	R	PWMA组数据寄存器1	0x0000_0000
	PWMB_BA0+0x20	R	PWMB组数据寄存器1	0x0000_0000
PDR2	PWMA_BA0+0x2C	R	PWMA组数据寄存器2	0x0000_0000
	PWMB_BA0+0x2C	R	PWMB组数据寄存器2	0x0000_0000
PDR3	PWMA_BA0+0x38	R	PWMA组数据寄存器3	0x0000_0000
	PWMB_BA0+0x38	R	PWMB组数据寄存器3	0x0000_0000

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

Bits	描述	
[31:16]	保留	保留
[15:0]	PDRx	<b>PWM 数据寄存器</b> 用户查询PDR可知16位计数器当前值.

## PWM 中断使能寄存器 (PIER)

寄存器	偏移量	R/W	描述	复位后的值
PIER	PWMA_BA+0x40	R/W	PWM A组中断使能寄存器	0x0000_0000
	PWMB_BA+0x40	R/W	PWM B组中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留				PIER3	PIER2	PIER1	PIER0

Bits	描述	
[31:4]	保留	保留
[3]	<b>PWMIE3</b>	<b>PWM 通道3中断使能</b> 1 = 使能 0 = 禁用
[2]	<b>PWMIE2</b>	<b>PWM 通道2中断使能</b> 1 = 使能 0 = 禁用
[1]	<b>PWMIE1</b>	<b>PWM 通道1中断使能</b> 1 = 使能 0 = 禁用
[0]	<b>PWMIE0</b>	<b>PWM 通道0中断使能</b> 1 = 使能 0 = 禁用

**PWM 中断标志寄存器(PIIR)**

寄存器	偏移量	R/W	描述	复位后的值
PIIR	PWMA_BA+0x44	R/W	PWM A组中断标志寄存器	0x0000_0000
	PWMB_BA+0x44	R/W	PWM B组中断标志寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留				PWMIF3	PWMIF2	PWMIF1	PWMIF0

Bits	描述	
[31:4]	保留	保留
[3]	PWMIF3	<b>PWM 通道3中断状态</b> 当PWM3向下计数至0时，硬件将该位置1。软件写1清该位。
[2]	PWMIF2	<b>PWM 通道 2 中断状态</b> 当PWM2向下计数至0时，硬件将该位置1。软件写1清该位。
[1]	PWMIF1	<b>PWM 通道 1 中断状态</b> 当PWM1向下计数至0时，硬件将该位置1。软件写1清该位。
[0]	PWMIF0	<b>PWM 通道 0中断状态</b> 当PWM0向下计数至0时，硬件将该位置1。软件写1清该位。

注：用户可通过对PIIR相应的位写1来对中断标志清零。

## 捕捉控制寄存器(CCR0)

寄存器	偏移量	R/W	描述	复位后的值
CCR0	PWMA_BA+0x50	R/W	PWM A组捕捉控制寄存器	0x0000_0000
	PWMB_BA+0x50	R/W	PWM B组捕捉控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
CFLRI1	CRLRI1	保留	CAPIF1	CAPCH1EN	FL_IE1	RL_IE1	INV1
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
CFLRI0	CRLRI0	保留	CAPIF0	CAPCH0EN	FL_IE0	RL_IE0	INV0

Bits	描述	
[31:24]	保留	保留
[23]	CFLRI1	<b>CFLR1锁定方向标志位</b> 在PWM输入通道1的下降沿, CFLR1 锁存PWM向下计数器, 并且该位由硬件置位。写1清该位。
[22]	CRLRI1	<b>CRLR1锁定方向标志位</b> 在PWM输入通道1的上升沿, CFLR1 锁存PWM向下计数器, 并且该位由硬件置位。写1清该位。
[5]	保留	保留
[20]	CAPIF1	<b>捕捉器1中断标志</b> 如果PWM组通道1上升沿锁定中断使能(CRL_IE1=1), PWM组通道1的向上传输将使CAPIF1为高; 同样, 如果下降沿锁定中断使能(CFL_IE1=1), 向下传输将使CAPIF1 为高。 该标志由软件写1清零。
[19]	CAPCH1EN	<b>捕捉器通道1传输使能/禁用</b> 1 = 使能PWM组通道1的捕捉功能。

		<p>0 = 禁用PWM组通道1的捕捉功能</p> <p>使能时，捕捉锁定PWM计数器并保存CRLR（上升沿锁定）和CFLR（上升沿锁定）。</p> <p>禁用时，捕捉器不更新CRLR和CFLR，并禁用PWM组通道1中断。</p>
[18]	CFL_IE1	<p><b>PWM 组通道1下降沿锁定中断使能</b></p> <p>1 = 使能向下锁定中断</p> <p>0 = 禁用向下锁定中断</p> <p>使能时，如果捕捉器检测到PWM组通道1有下降沿，捕捉器产生中断。</p>
[17]	CRL_IE1	<p><b>PWM 组通道1上升沿锁定中断使能</b></p> <p>1 = 使能向上锁定中断</p> <p>0 = 禁用向上锁定中断</p> <p>使能时，如果捕捉器检测到PWM组通道1有上升沿，捕捉器产生中断。</p>
[16]	INV1	<p><b>通道1反向打开/关闭</b></p> <p>1 = 反向打开。输入到寄存器的信号与通道上的实际信号电平反向。</p> <p>0 = 反向关闭</p>
[15:8]	保留	保留
[7]	CFLR10	<p><b>CFLR0锁定方向标志位</b></p> <p>在PWM输入通道0的下降沿，CFLR0 锁存PWM向下计数器，并且该位由硬件置位。</p> <p>写1清该位</p>
[6]	CRLR10	<p><b>CRLR0锁定方向标志位</b></p> <p>在PWM输入通道0的上升沿，CRLR0 锁存PWM向下计数器，并且该位由硬件置位。</p> <p>写1清该位。</p>
[5]	保留	保留
[4]	CAPIF0	<p><b>捕捉器0中断标志</b></p> <p>如果PWM组通道1上升沿锁定中断使能(CRL_IE0=1)，PWM组通道1的上升沿将使CAPIF0为高；同样，如果使能下降沿锁定中断(CFL_IE0=1)，下降沿将使CAPIF0为高。</p> <p>该标志由软件写1清零。</p>
[3]	CAPCH0EN	<p><b>捕捉器通道0传输使能/禁用</b></p> <p>1 = 使能PWM组通道0的捕捉功能。</p> <p>0 = 禁用PWM组通道0的捕捉功能</p>



		使能时，捕捉锁定PWM计数器并保存CRLR（向上锁定）和CFLR（向下锁定）。 禁用时，捕捉器不更新CRLR和CFLR，并禁用PWM组通道0中断。
[2]	<b>CFL_IE0</b>	<b>通道0下降沿锁定中断使能</b> 1 = 使能下降沿锁定中断 0 = 禁用下降沿锁定中断 使能时，捕捉器检测到PWM组通道0有向上传输，捕捉器产生中断。
[1]	<b>CRL_IE0</b>	<b>PWM 组通道0上升沿锁定中断使能</b> 1 = 使能上升沿锁定中断 0 = 禁用上升沿锁定中断 使能时，如果捕捉器检测到PWM组通道0有向上传输，捕捉器产生中断。
[0]	<b>INVO</b>	<b>通道0反向打开/关闭</b> 1 = 反向打开。输入到寄存器的信号与通道上的实际信号点平反向。 0 = 反向关闭

## 捕捉控制寄存器(CCR2)

寄存器	偏移量	R/W	描述	复位后的值
CCR2	PWMA_BA+0x54	R/W	PWM A组捕捉控制寄存器	0x0000_0000
	PWMB_BA+0x54	R/W	PWM B组捕捉控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
CFLRI3	CRLRI3	保留	CAPIF3	CAPCH3EN	FL_IE3	RL_IE3	INV3
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
CFLRI2	CRLRI2	保留	CAPIF2	CAPCH2EN	FL_IE2	RL_IE2	INV2

Bits	描述	
[31:24]	保留	保留
[23]	CFLRI3	<b>CFLR3锁定方向标志位</b> 在PWM输入通道3的下降沿, CFLR3锁定PWM向下计数器的值, 同时该位由硬件置位.写1清该位.
[22]	CRLRI3	<b>CRLR3锁定方向标志位</b> 在PWM输入通道3的下降沿, CFLR3锁定PWM向下计数器的值, 同时该位由硬件置位.写1清该位.
[21]	保留	保留
[20]	CAPIF3	<b>捕捉器1中断标志</b> 如果PWM组通道3上升沿锁定中断使能(CRL_IE3=1), PWM组通道3的上升沿传输将使CAPIF3为高; 同样, 如果下降沿锁定使能(CFL_IE3=1), 向下下降沿传输将使CAPIF3 为高. 该标志由软件写1清零.
[19]	CAPCH3EN	<b>捕捉器通道3传输使能/禁用</b> 1 = 使能PWM组通道3的捕捉功能. 0 = 禁用PWM组通道3的捕捉功能

		<p>使能时，捕捉锁定PWM计数器并保存CRLR（上升沿锁定）和CFLR（下降沿锁定）。</p> <p>禁用时，捕捉器不更新CRLR和CFLR，并禁用PWM组通道3中断。</p>
[18]	<b>CFL_IE3</b>	<p><b>PWM 组通道3下降沿锁定中断使能</b></p> <p>1 = 使能下降沿锁定中断</p> <p>0 = 禁用下降沿锁定中断</p> <p>使能时，捕捉器检测到PWM组通道3有向下传输，捕捉器产生中断。</p>
[17]	<b>CRL_IE3</b>	<p><b>PWM 组通道3上升沿锁定中断使能</b></p> <p>1 = 使能上升沿锁定中断</p> <p>0 = 禁用上升沿锁定中断</p> <p>使能时，如果捕捉器检测到PWM组通道3有上升沿，捕捉器产生中断。</p>
[16]	<b>INV3</b>	<p><b>通道3反向打开/关闭</b></p> <p>1 = 反向打开。输入到寄存器的信号与通道上的实际信号电平反向。</p> <p>0 = 反向关闭</p>
[15:8]	保留	保留
[7]	<b>CFLR2</b>	<p><b>CFLR2锁定方向标志位</b></p> <p>在PWM输入通道2的下降沿，CFLR2锁定PWM向下计数器的值，硬件置位。写1清该位。</p>
[6]	<b>CRLR2</b>	<p><b>CRLR2锁定方向标志位</b></p> <p>在PWM输入通道2的下降沿，CRLR2锁定PWM向下计数器的值，硬件置位。写1清该位。</p>
[5]	保留	保留
[4]	<b>CAPIF2</b>	<p><b>捕捉器2中断指示标志</b></p> <p>如果PWM组通道2上升沿锁定中断使能(CRL_IE2=1)，PWM组通道2的上升沿将使CAPIF2为高；同样，如果使能(CFL_IE2=1)，下降沿将使CAPIF2为高。</p> <p>该标志由软件写1清零。</p>
[3]	<b>CAPCH2EN</b>	<p><b>捕捉器通道2传输使能/禁用</b></p> <p>1 = 使能PWM组通道2的捕捉功能。</p> <p>0 = 禁用PWM组通道2的捕捉功能</p> <p>使能时，捕捉锁定PWM计数器并保存CRLR（上升沿锁定）和CFLR（下降沿）。</p> <p>禁用时，捕捉器不更新CRLR和CFLR，并禁用PWM组通道2中断。</p>
[2]	<b>CFL_IE2</b>	<p><b>PWM 组通道2下降沿锁定中断使能</b></p>

		<p>1 = 使能下降沿锁定中断</p> <p>0 = 禁用下降沿锁定中断</p> <p>使能时，捕捉器检测到PWM组通道2有下降沿，捕捉器产生中断。</p>
[1]	<b>CRL_IE2</b>	<p><b>PWM 组通道2上升沿锁定中断使能</b></p> <p>1 = 使能上升沿锁定中断</p> <p>0 = 禁用上升沿锁定中断</p> <p>使能时，如果捕捉器检测到PWM组通道2有上升沿，捕捉器产生中断。</p>
[0]	<b>INV2</b>	<p><b>通道2反向打开/关闭</b></p> <p>1 = 反向打开。输入到寄存器的信号与通道上的实际信号电平反向。</p> <p>0 = 反向关闭</p>

## 捕捉上升沿锁存寄存器3-0 (CRLR3-0)

寄存器	偏移量	R/W	描述	复位后的值
CRLR0	PWMA_BA+0x58	R	PWM A组捕捉上升沿锁存寄存器(channel 0)	0x0000_0000
	PWMB_BA+0x58	R	PWM B组捕捉上升沿锁存寄存器(channel 0)	0x0000_0000
CRLR1	PWMA_BA+0x60	R	PWM A组捕捉上升沿锁存寄存器(channel 1)	0x0000_0000
	PWMB_BA+0x60	R	PWM B组捕捉上升沿锁存寄存器(channel 1)	0x0000_0000
CRLR2	PWMA_BA+0x68	R	PWM A组捕捉上升沿锁存寄存器(channel 2)	0x0000_0000
	PWMB_BA+0x68	R	PWM B组捕捉上升沿锁存寄存器(channel 2)	0x0000_0000
CRLR3	PWMA_BA+0x70	R	PWM A组捕捉上升沿锁存寄存器(channel 3)	0x0000_0000
	PWMB_BA+0x70	R	PWM B组捕捉上升沿锁存寄存器(channel 3)	0x0000_0000

注: 当CPU时钟低于PWM/Capture时钟时, 不能对CRLRx进行改写.

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

Bits	描述	
[31:16]	保留	保留
[15:0]	CRLRx	捕捉上升沿锁存寄存器 通道0/1/2/3 上升沿时, 锁存PWM计数器.

## 捕捉下降沿锁存寄存器3-0 (CFLR3-0)

寄存器	偏移量	R/W	描述	复位后的值
CFLR0	PWMA_BA+0x5C	R	PWM A组捕捉下降锁存寄存器(channel 0)	0x0000_0000
	PWMB_BA+0x5C	R	PWM B组捕捉下降锁存寄存器(channel 0)	0x0000_0000
CFLR1	PWMA_BA+0x64	R	PWM A组捕捉下降锁存寄存器(channel 1)	0x0000_0000
	PWMB_BA+0x64	R	PWM B组捕捉下降锁存寄存器(channel 1)	0x0000_0000
CFLR2	PWMA_BA+0x6C	R	PWM A组捕捉下降锁存寄存器(channel 2)	0x0000_0000
	PWMB_BA+0x6C	R	PWM B组捕捉下降锁存寄存器(channel 2)	0x0000_0000
CFLR3	PWMA_BA+0x74	R	PWM A组捕捉下降锁存寄存器(channel 3)	0x0000_0000
	PWMB_BA+0x74	R	PWM B组捕捉下降锁存寄存器(channel 3)	0x0000_0000

注: 当CPU时钟低于PWM/Capture时钟时, 不能对 CFLRx 进行改写.

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

Bits	描述	
[31:16]	保留	保留
[15:0]	CFLRx	捕捉下降沿锁存寄存器 通道01/2/3 下降沿时, 锁存PWM计数器.

## 捕捉输入使能寄存器(CAPENR)

寄存器	偏移量	R/W	描述	复位后的值
-----	-----	-----	----	-------



<b>CAPENR</b>	PWMA_BA+0x78	R/W	PWM A组捕捉输入0~3 使能寄存器	0x0000_0000
	PWMB_BA+0x78	R/W	PWM B组捕捉输入0~3 使能寄存器	0x0000_0000

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

Bits	描述	
[3:0]	<b>CAPENR</b>	<p>捕捉输入使能寄存器</p> <p>4组捕捉输入。Bit0~Bit3 用于控制每个输入的打开 / 关闭。</p> <p>0 = 关闭(PWMx 复用脚输入对捕捉器不产生影响)</p> <p>1 = 打开 (PWMx 复用脚将影响捕捉器功能.)</p> <p><b>CAPENR</b></p> <p><u>Bit 3210 用于PWM A组</u></p> <p>Bit xxx1 → 捕捉通道0 从 P2 [0] 输入</p> <p>Bit xx1x → 捕捉通道1 从 P2 [1] 输入</p> <p>Bit x1xx → 捕捉通道2 从 P2 [2] 输入</p> <p>Bit 1xxx → 捕捉通道3 从 P2 [3] 输入</p> <p><u>Bit 3210用于PWM B组</u></p> <p>Bit xxx1 → 捕捉通道0 从 P2 [4] 输入</p> <p>Bit xx1x → 捕捉通道1 从 P2 [5] 输入</p> <p>Bit x1xx → 捕捉通道2 从 P2 [6] 输入</p> <p>Bit 1xxx → 捕捉通道3 从 P2 [7] 输入</p>

## PWM输出使能寄存器 (POE)

寄存器	偏移量	R/W	描述	复位后的值
POE	PWMA_BA+0x7C	R/W	PWM A组输出使能寄存器 (通道0~3)	0x0000_0000
	PWMB_BA+0x7C	R/W	PWM B组输出使能寄存器 (通道0~3)	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留				PWM3	PWM2	PWM1	PWM0

Bits	描述	
[3]	PWM3	<b>PWM 通道3输出使能寄存器</b> 1 = 使能PWM通道3输出 0 = 禁用PWM通道3输出 注: GPIO相应管脚必须切换到PWM功能
[2]	PWM2	<b>PWM 通道2输出使能寄存器</b> 1 = 使能PWM通道2输出 0 = 禁用PWM通道2输出 注: GPIO相应管脚必须切换到PWM功能
[1]	PWM1	<b>PWM 通道1输出使能寄存器</b> 1 = 使能PWM通道1输出 0 = 禁用PWM通道1输出 注: GPIO相应管脚必须切换到PWM功能
[0]	PWM0	<b>PWM 通道0输出使能寄存器</b> 1 = 使能PWM通道0输出



		0 = 禁用PWM通道0输出 注: GPIO相应管脚必须切换到PWM功能
--	--	---

## 6.7 串行外围设备接口(SPI)控制器

### 6.7.1 简介

SPI 接口是工作于全双工模式下的同步串行数据传输接口。器件通过4线双端接口工作于主机/从机模式。NuMicro M051™系列包括最多2组SPI控制器，将从外设接收到的数据进行串并转换，或将要发送到外设的数据进行并串转换。每组SPI控制器都可被设置成主机；也可设置为被外围主机控制的从机。

### 6.7.2 特性

- 两组SPI传送控制器
- 支持主/从机模式
- 最大传输字可达32位，一次最多可传输2个字，即一次最多可传输64位数据
- 支持burst操作模式，在一次传输过程中，发送/接收可执行两次字传输
- 支持MSB 或 LSB 优先传输
- 字节或字空闲功能
- 主机模式下可输出多种串行时钟频率
- 主机模式下支持两个可编程的串行时钟频率

6.7.3 SPI 框图

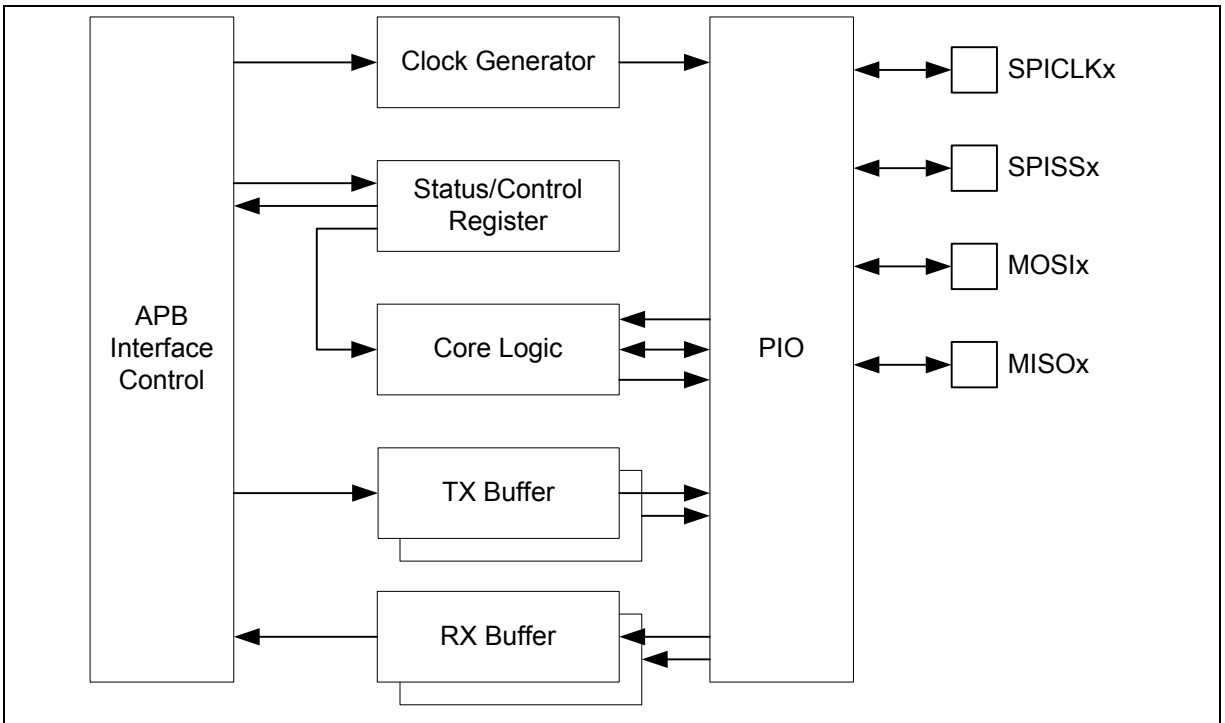


图 6.7-1 SPI 框图

## 6.7.4 SPI 功能描述

### 主/从模式

SPI控制器可通过设置**SLAVE**位(SPI\_CNTRL[18])配置为主机或从机模式，来与外设从机或主机通信.主机模式与从机模式的应用框图如下.

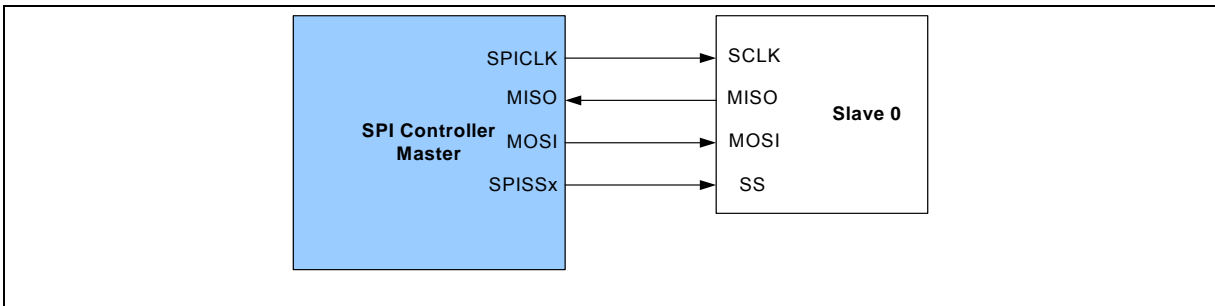


图 6.7-2 SPI主机模式应用框图

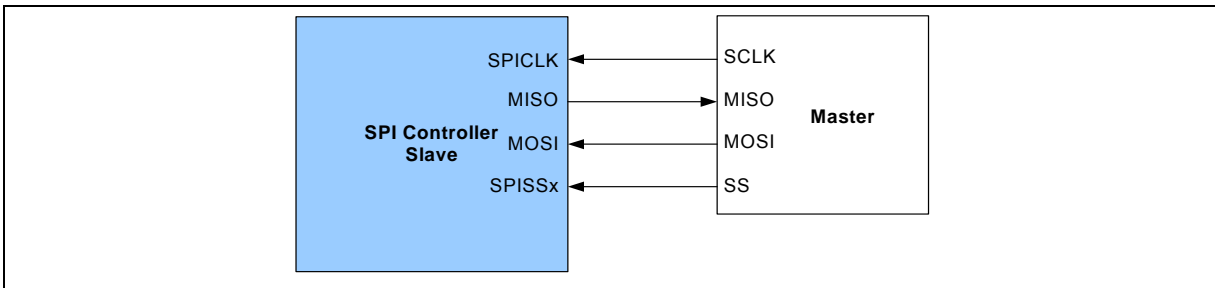


图 6.7-3 SPI从机模式应用框图

### 从机选择

在主机模式下，SPI控制器能通过从机选择输出脚SPISS驱动一个外围从机设备。从机模式下，外围的主机设备驱动从机信号是通过SPISS输入到SPI控制器的。在主机/从机模式下，从机选择信号的有效电平可以在SS\_LVL位(SPI\_SSR[2])被编程为低有效或高有效，SS\_LTRIG位(SPI\_SSR[4])配置从机选择信号SPISS为电平触发或边沿触发。触发条件的选择取决于所连接的从机/主机的外设类型。

### 电平触发/边沿触发

在从机模式下，从机选择信号可以配置成电平触发或边沿触发。边沿触发，数据传输从有效边沿开始。如果主机不发送稳定边沿信号给从机，传输将不能完成，从机的中断标志将不会被置位。电平触发，下面两个情况可以中止传输过程，并使从机的中断标志被置位。一，如果主机设置从机选择管脚为非有效电平，将迫使从机中止当前传输而不管已经传输多少位，并且中断标志将置位。用户可以读取LTRIG\_FLAG判断数据是否传输完毕。二，如果传输位数与TX\_NUM和TX\_BIT\_LEN的设置匹配时，从机的中断标志置位。

### 自动从机选择

在主机模式下，如果AUTOSS(SPI\_SSR[3])置位，从机选择信号自动产生，并根据SSR[0](SPI\_SSR[0])是否使能，输出到SPISS引脚上，这意味着，从机选择信号(由SSR[0]寄存器使能)由SPI控制器在发送/接收开始(由GO\_BUSY位设置)时生效，在传输结束时失效。当AUTOSS位清零时，可以手动置位与清零寄存器SPI\_SSR[0]的相关位，来声明或取消从机选择输出信号。在SS\_LVL位(SPI\_SSR[2])配置了从机选择输出信号的有效电平。

### 串行时钟

在主机模式下，配置驱动器寄存器(SPI\_DIVIDER[15:0])来编程由SPICLK引脚输出的串行时钟频率。如果VARCLK\_EN bit(SPI\_CNTRL[23])使能，串行时钟也支持频率可调功能，在这种情况下，通过配置DIVIDER和DIVIDER2(SPI\_DIVIDER[31:16])，串行时钟每一位的输出频率可被编程为一种或两种不同频率。每一位的可调频率是由VARCLK(SPI\_VARCLK[3:0])寄存器定义的。在从机模式下，外设主机设备通过此SPI控制器的SPICLK输入口驱动串行时钟。

### 时钟极性

在主机模式下，CLKP位(SPI\_CNTRL[11])配置为串行时钟的空闲状态。当CLKP = 1，输出SPICLK在高电平下为空闲状态。CLKP = 0时，输出SPICLK在低电平下为空闲状态。对于可调串行时钟，仅CLKP=0时有效。

### 发送/接收位长度

传输字的长度在Tx\_BIT\_LEN 位(SPI\_CNTRL[7:3])中配置. 发送/接收的传输字可达32位长度.

### Burst 模式/脉冲模式

SPI可通过设置TX\_NUM (SPI\_CNTRL [9:8])为0X01, 切换到burst模式. burst 模式下, SPI 可以在一次传输中进行两次发送/接收处理。SPI burst 模式波形图如下:

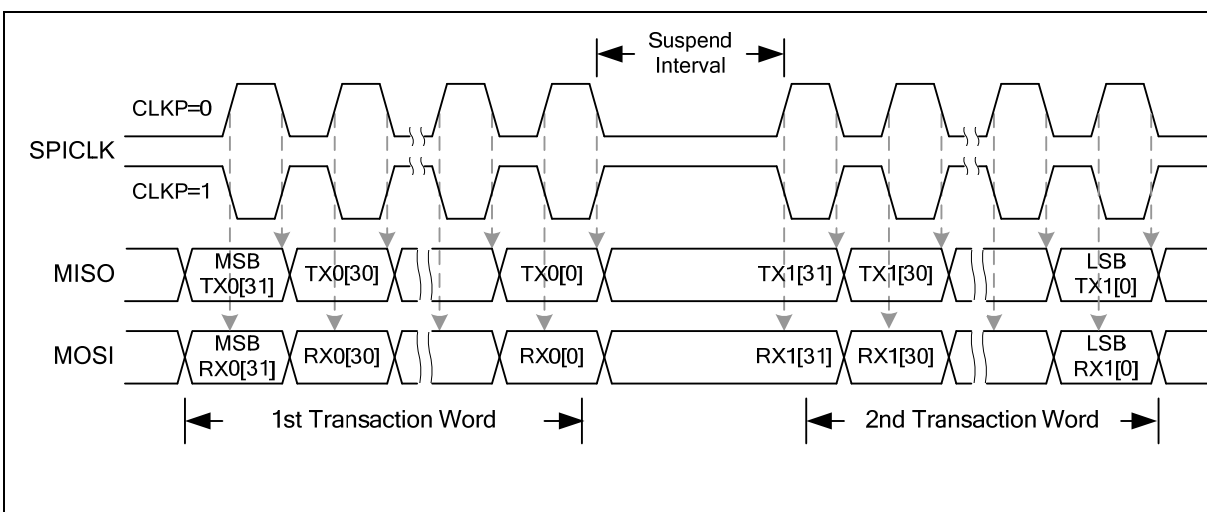


图 6.7-4 一次传输两个Transactions (Burst Mode)

### LSB First

LSB 位(SPI\_CNTRL[10]) 定义是从LSB还是从MSB开始发送/接收数据.

### 发送边界

Tx\_NEG 位 (SPI\_CNTRL[2]) 定义数据发送是在串行时钟SPICLK的下降沿还是上升沿.

### 接收边界

Rx\_NEG 位 (SPI\_CNTRL[1]) 定义数据接收是在串行时钟SPICLK的下降沿还是上升沿.

## 字休眠

在主机模式下，SP\_CYCLE (SPI\_CNTRL[15:12])的4位在两个连续传输字之间的提供可配置为2~17个串行时钟周期的休眠间隔。休眠间隔指从前一次传输字的最后一个时钟下降沿到下一次传输字的第一个时钟上升沿（CLKP = 0），如果CLKP = 1，间隔为前一次传输字的上升沿到下一次传输字的下降沿。SP\_CYCLE的默认值为0x0（2个串行时钟周期），如果Tx\_NUM = 0x00，设置这些位不会影响数据传输过程。

## 字节重排序

当设置为MSB优先时（LSB = 0），且REORDER使能，TX\_BIT\_LEN = 32位模式下，存储在TX缓存与RX缓存中的数据将按[BYTE0, BYTE1, BYTE2, BYTE3]的次序重新排列，发送/接收数据将变成BYTE0, BYTE1, BYTE2, BYTE3的顺序。如果Tx\_BIT\_LEN 设置为24-位模式时，TX 缓存与RX 缓存的数据将被重新排列为[unknown byte, BYTE0, BYTE1, BYTE2]，在MSB优先时，BYTE0, BYTE1, BYTE2将逐步发送/接收。16位模式与上面相同。

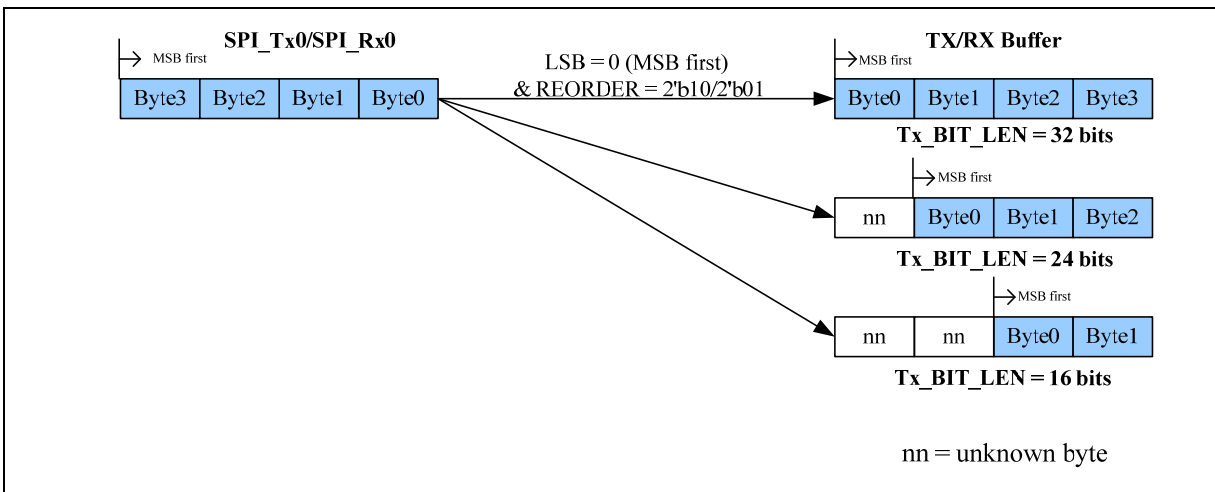


图 6.7-5 字节重排列

## 字节休眠

主机模式下，如果置(SPI\_CNTRL[19])为1，硬件将在两个连续传输字节之间插入2~17个串行时钟周期的休眠间隔。休眠字节的设置与字休眠在SP\_CYCLE寄存器的普通位一样，注意当使能字节休眠时，TX\_BIT\_LEN必须设置为0x00（一个传输字32位）。

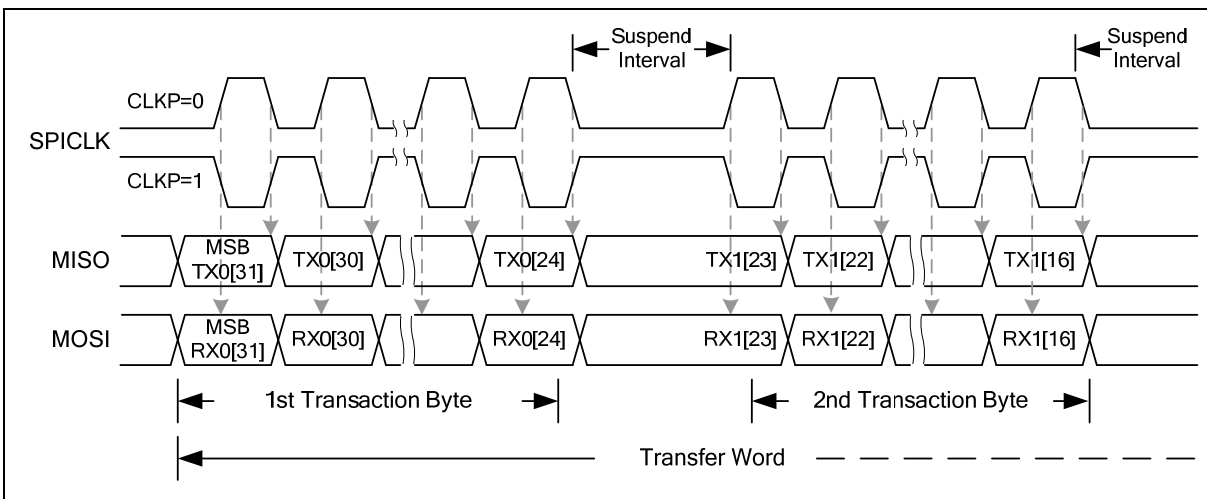


图 6.7-6 字节休眠时序波形

表 11-1 字节顺序和字节休眠条件

REORDER	描述
00	禁用字节重排序功能和字节休眠。
01	使能字节重排序功能，并在每个字节之间插入一个字节的休眠间隔（2~17串行时钟周期）。TX_BIT_LEN 的设置必须配置成0x00（32 bits/ word）
10	使能字节重排序功能但不禁用字节休眠功能。
11	禁用字节重排序功能，但在每个字节之间插入一个休眠间隔（2~17串行时钟周期）。TX_BIT_LEN的设置必须配置成 0x00（32 bits/ word）

## 中断

数据传输完毕时，每一个SPI控制器会产生独立的中断和相应的中断标志IF (SPI\_CNTRL[16])被置位。如果中断使能位IE (SPI\_CNTRL[17]) 置位，则中断事件标志将向CPU提出中断请求。中断标志可由软件写1清零。



## 可调串行时钟频率

在主机模式下，如果使能可调时钟控制位VARCLK\_EN (SPI\_CNTRL [23])，串行时钟的输出可以编程为可调种时钟频率。频率格式在寄存器VARCLK (SPI\_VARCLK [31:0]) 里定义。如果VARCLK 的该位为‘0’，输出频率取决于DIVIDER (SPI\_DIVIDER[15:0])，如果VARCLK 的该位为‘1’，输出频率取决于DIVIDER2 (SPI\_DIVIDER[31:16])。下图为串行时钟(SPICLK)，VARCLK，DIVIDER和DIVIDER2的时序。VARCLK的相邻两位占一个时钟周期。VARCLK [31:30] 占SPICLK的第一个时钟周期。VARCLK [29:28] 占SPICLK的第二个时钟周期，诸如此类。在VARCLK中选择时钟源，且必须在切换到下一个时钟源前1个周期被设置。例如，在SPICLK有5个CLK1，VARCLK将在MSB设置9个‘0’。第10个将设置为‘1’以切换到下一个时钟源CLK2。注意当使能VARCLK\_EN 位，TX\_BIT\_LEN 必须设置成0x10 (仅16 bits 模式)。

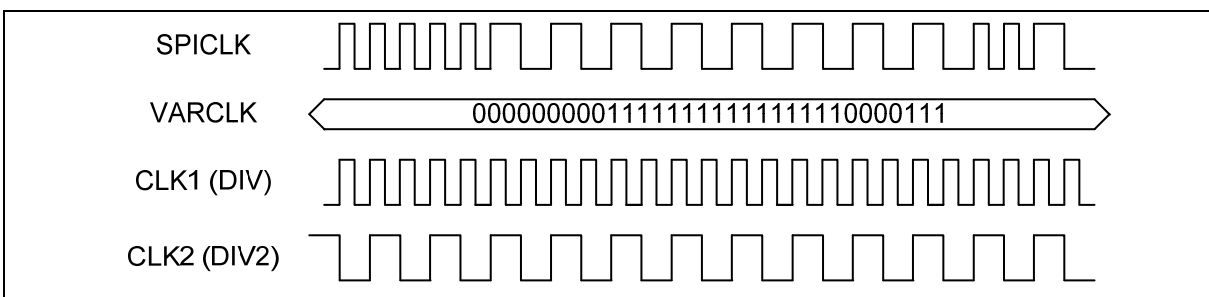


图 6.7-7 可调串行时钟频率

## 6.7.5 SPI 时序波形图

在主机/从机模式下，设备/从机选择信号(SPISS)的有效电平可以在**SS\_LVL**位(SPI\_SSR[2])被编程为低电平有效或高电平有效，在**SS\_LTRIG**位(SPI\_SSR[4])中配置SPISSx0/1为电平触发还是边沿触发。串行时钟(SPICLK)的空闲状态可以通过**CLKP**位(SPI\_CNTRL[11])配置为高状态或低状态。在**Tx\_BIT\_LEN**(SPI\_CNTRL[7:3])中配置传输字的长度，在**Tx\_NUM**(SPI\_CNTRL[8])中配置传输的数目，在**LSB bit**(SPI\_CNTRL[10])中配置发送/接收数据是MSB还是LSB优先。寄存器**Tx\_NEG/Rx\_NEG**(SPI\_CNTRL[2:1])的设置，可以使用户选择发送/接收数据时串行时钟的边沿。四个SPI时序图及相关设置如下。

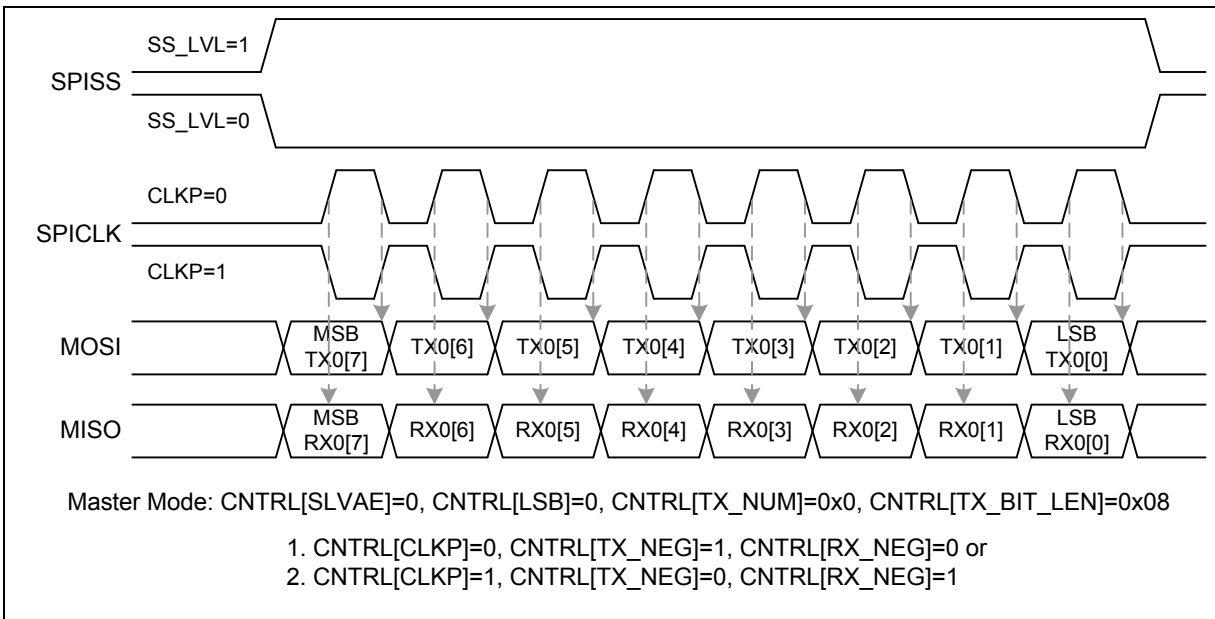


图 6.7-8 主机模式下 SPI 时序

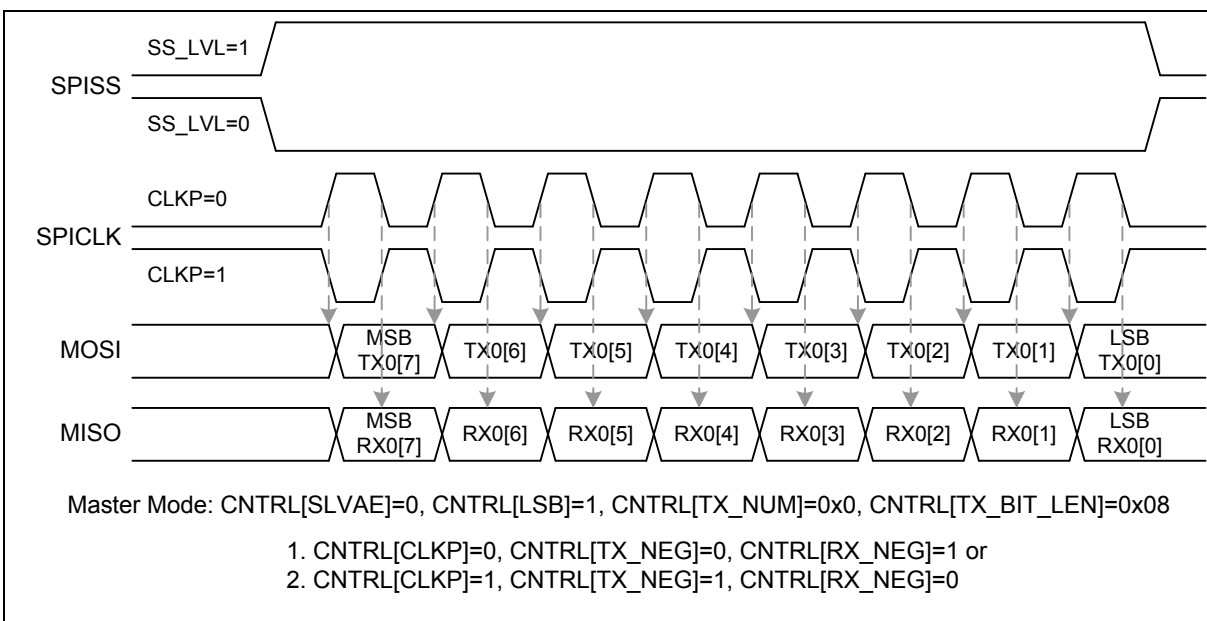


图 6.7-9 主机模式下 SPI 时序(Alternate Phase of SPICLK)

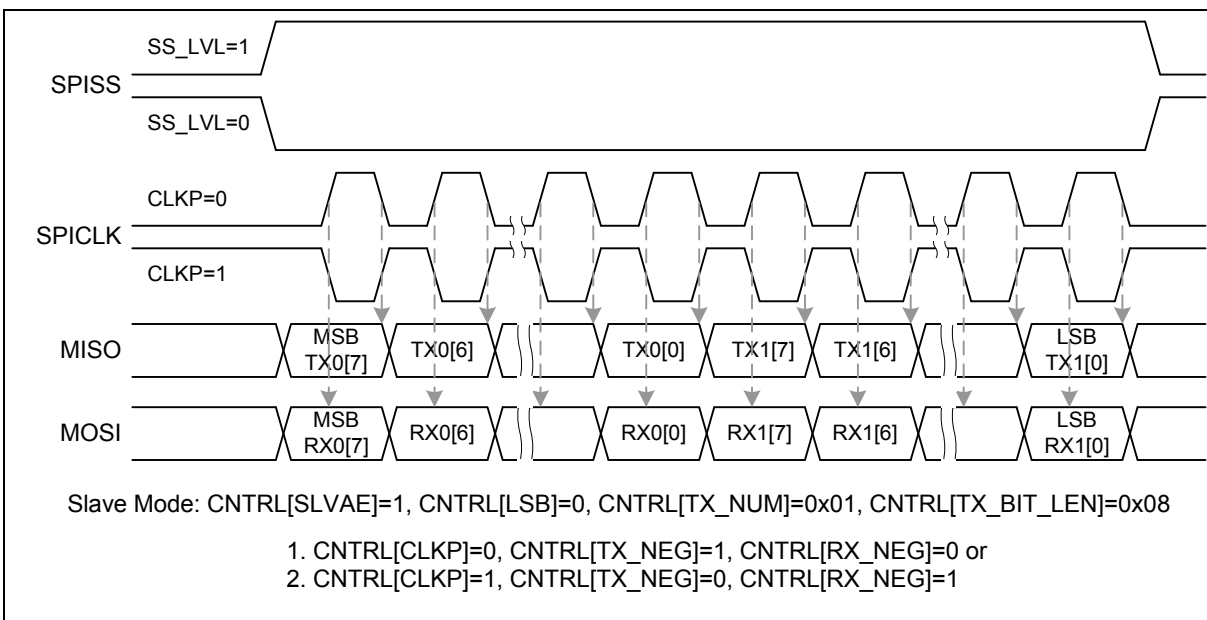


图 6.7-10 从机模式下 SPI 时序

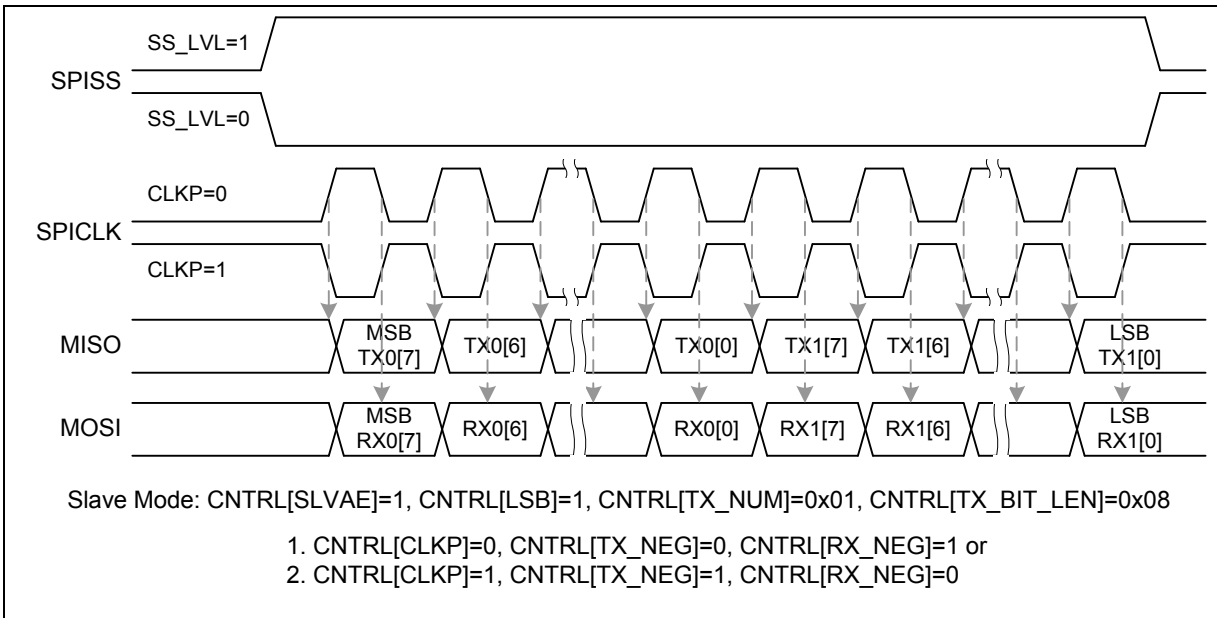


图 6.7-11 从机模式下 SPI 时序(Alternate Phase of SPICLK)

### 6.7.6 SPI编程例程

■ 例 1, SPI 作为主机时, 按如下说明与一个从机设备通信:

- 数据在时钟上升沿锁存
- 数据在时钟下降沿传输
- 数据先传送MSB
- SPICLK低电平时空闲
- 每次输出/接收一个字节
- 从机选择信号低有效

基本上, 在执行这些步骤之前, 需要详细查阅片外从机的说明。

- 1) 设置DIVIDER (SPI\_DIVIDER[15:0])寄存器, 配置串行时钟输出频率.
- 2) 向SPI\_SSR写入适当的值, 用于主机模式的相应设置
  1. 禁用自动从机选择位AUTOSS (SPI\_SSR[3] = 0)
  2. 配置从机选择有效电平位SS\_LVL (SPI\_SSR[2] = 0) 使从机选择信号为低电平触发输出
  3. 通过设置相关从机选择寄存器SSR[0] (SPI\_SSR[0]), 激活片外从设备来使I/O引脚上的从机选择信号生效
- 3) 向寄存器SPI\_CNTRL 写入相应设置, 控制SPI主机
  1. 通过SLAVE 位 (SPI\_CNTRL[18] = 0)设置PSI控制器为主机设备
  2. 通过CLKP位(SPI\_CNTRL[11] = 0) 设置串行时钟为空闲状态
  3. 通过Tx\_NEG位(SPI\_CNTRL[2] = 1)选择数据在串行时钟的下降沿传输
  4. 通过Rx\_NEG位(SPI\_CNTRL[1] = 0)选择数据在串行时钟的上升沿锁存
  5. 通过Tx\_BIT\_LEN位域 (SPI\_CNTRL[7:3] = 0x08)设置传输字的长度为8位
  6. 通过 Tx\_NUM (SPI\_CNTRL[9:8] = 0x0) 设置为一次字传输
  7. 通过 MSB 位 (SPI\_CNTRL[10] = 1) 设置为 MSB 传输, 不必关心 SP\_CYCLE (SPI\_CNTRL[15:12])的设置, 因为在本例中没有burst模式
- 4) 如果SPI主机发送一个字节的的数据到外设, 所写字节数据将被传输到寄存器Tx0[7:0] (SPI\_Tx0[7:0]).
- 5) 如果SPI主机从外设接收一个字节的的数据, 不必关心什么数据被传输, 只需要向寄存器SPI\_Tx0[7:0] 写入0xFF .
- 6) 使能GO\_BUSY 位(SPI\_CNTRL[0] = 1), 以开始数据传输。
- 7) 等待SPI中断发生 (IE使能), 或检测GO\_BUSY 位直到其被硬件自动清零
- 8) 从寄存器Rx0[7:0] (SPI\_Rx0[7:0])读出所接收到的一个字节的的数据.

9) 重复步骤 4) 继续其他数据的传输或设置**SSR[0]**为0 以停止外设.

■ 例 2, SPI控制器作为从机设备, 由外设主机设备控制, 片外主机设备依如下说明通过SPI接口与片上SPI从机通信:

- 数据在时钟上升沿锁存
- 数据在时钟下降沿传输
- 数据先传送 LSB
- SPICLK高电平时空闲
- 每次输出/接收一个字节
- 从机选择信号高有效

基本上, 在执行这些步骤之前, 需要详细查阅片外主机的说明。

- 1) 向寄存器SPI\_SSR写入正确值设置为从机模式, 配置从机选择有效电平位**SS\_LVL** (SPI\_SSR[2] = 1)与从机选择电平触发位**SS\_LTRIG** (SPI\_SSR[4] = 1)为从机选择信号输入选择高电平和电平触发.
- 2) 向寄存器SPI\_CNTRL 写入相应配置以控制SPI从机
  1. 通过**SLAVE**位(SPI\_CNTRL[18] = 1)设置SPI控制器为从机设备
  2. 通过**CLKP**位(SPI\_CNTRL[11] = 1)选择串行时钟空闲状态
  3. 通过 **Tx\_NEG**位(SPI\_CNTRL[2] = 1)选择数据传输发生在串行时钟下降沿
  4. 通过**Rx\_NEG**位(SPI\_CNTRL[1] = 0)选择数据锁存在串行时钟的上升沿
  5. 通过**Tx\_BIT\_LEN**位field (SPI\_CNTRL[7:3] = 0x08)设置字传输长度为8位
  6. 通过**Tx\_NUM** (SPI\_CNTRL[9:8] = 0x0)设置为仅一次字传输
  7. 通过 **LSB**位(SPI\_CNTRL[10] = 1)设置为LSB传输, 在本例中由于没有burst模式, 因而不必关心.
- 3) 如果SPI从机传输一个字节数据到外设主机, 所写的数字将传输到寄存器**Tx0[7:0]** (SPI\_Tx0[7:0]).
- 4) 如果SPI从机仅从外设主机接收一字节数据, 用户不必关心什么数据将被传输, 只需要向寄存器SPI\_Tx0[7:0]写入0xFF.
- 5) 使能**GO\_BUSY** bit (SPI\_CNTRL[0] = 1), 等待外设的从机选择触发输入和串行时钟输入, 开始数据传输到SPI接口.
 

--等待SPI中断发生 (IE使能), 或检测**GO\_BUSY** 位直到其被硬件自动清零--
- 6) 从**Rx[7:0]** (SPI\_Rx0[7:0]) 寄存器中读出所接收到的一个字节的数字
- 7) 重复步骤3) 继续其他数据传输或禁用**GO\_BUSY** 位停止数据传输.

## 6.7.7 SPI串行总线控制寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
SPI0_BA = 0x4003_0000 SPI1_BA = 0x4003_4000				
SPI_CNTRL	SPIx_BA + 0x00	R/W	控制及状态寄存器	0x0000_0004
SPI_DIVIDER	SPIx_BA + 0x04	R/W	时钟分频寄存器	0x0000_0000
SPI_SSR	SPIx_BA + 0x08	R/W	从机选择寄存器	0x0000_0000
SPI_RX0	SPIx_BA + 0x10	R	数据接收寄存器0	0x0000_0000
SPI_RX1	SPIx_BA + 0x14	R	数据接收寄存器1	0x0000_0000
SPI_TX0	SPIx_BA + 0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1	SPIx_BA + 0x24	W	数据发送寄存器1	0x0000_0000
SPI_VARCLK	SPIx_BA + 0x34	R/W	可调时钟类型控制寄存器	0x007F_FF87

注 1: 由软件编写程CNTRL 寄存器, GO\_BUSY 位必须最后写入。

## 6.7.8 SPI控制寄存器描述

## SPI 控制与状态寄存器(SPI\_CNTRL)

寄存器	偏移量	R/W	描述	复位后的值
SPI_CNTRL	SPIx_BA + 0x00	R/W	控制与状态寄存器	0x0000_0004

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
VARCLK_EN	保留		REORDER		SLAVE	IE	IF
15	14	13	12	11	10	9	8
SP_CYCLE				CLKP	LSB	TX_NUM	
7	6	5	4	3	2	1	0
TX_BIT_LEN					TX_NEG	RX_NEG	GO_BUSY

Bits	描述	
[31:24]	保留	保留
[23]	VARCLK_EN	<p>可调多时钟使能 (仅主机)</p> <p>0 = 串行时钟输出仅由DIVIDER的值决定.</p> <p>1 = 串行时钟输出可变. 输出频率由VARCLK, DIVIDER, 和 DIVIDER2的值决定.</p> <p>注: 当使能VARCLK_EN, TX_BIT_LEN 必须设置成 0x10 (16 bits 模式)</p>
[22:21]	保留	保留
[20:19]	REORDER	<p>重排序模式选择</p> <p>00 = 禁用字节重排序和字节休眠功能.</p> <p>01 = 使能字节重排序, 并在每个字节之间插入一个字节休眠间隔 (2~17 串行时钟周期). TX_BIT_LEN 必须设置成0x00. (32 bits/word)</p> <p>10 = 使能字节重排序功能, 但禁用字节休眠功能.</p> <p>11 = 禁用字节重排序功能, 但在每个字节之间插入一个休眠间隔(2~17 串行时钟周期). TX_BIT_LEN 必须设置成0x00. (32 bits/word)</p>



[18]	SLAVE	<b>从机模式选择</b> 0 = 主机模式 1 = 从机模式
[17]	IE	<b>中断使能</b> 0 = 禁用MICROWIRE/SPI 中断. 1 = 使能MICROWIRE/SPI中断.
[16]	IF	<b>中断标志</b> 0 = 表示传输未结束 1 = 表示传输完成。当SPI使能，该位置1。 <b>注：</b> 该位写1清零。
[15:12]	SP_CYCLE	<b>休眠间隙 (仅主机模式)</b> 该四位用于编辑增加在两次连续传输内的间隔时间。如果CLKP=0，间隔时间从当前传输的最后一个时钟下降沿到下次传输的第一个时钟上升沿。如果CLKP=1，间隔时间从时钟上升沿到时钟下降沿。默认值为0x0。当Tx_NUM = 00, 该位无效。下列公式可获得所需的间隔时间。 $(SP\_CYCLE[3:0] + 2) * SPI\text{时钟周期}$ SP_CYCLE = 0x0 ... 2个SPI时钟周期 SP_CYCLE = 0x1 ... 3个SPI时钟周期 ..... SP_CYCLE = 0xe ... 16个SPI时钟周期 SP_CYCLE = 0xf ... 17个SPI时钟周期
[11]	CLKP	<b>时钟极性</b> 0 = SCLK 低电平空闲. 1 = SCLK 高电平空闲.
[10]	LSB	<b>优先传送LSB</b> 0 = 优先发送/接收MSB (具体是SPI_TX0/1和SPI_RX0/1 寄存器的哪一位取决于TX_BIT_LEN的值). 1 = 优先发送 LSB (SPI_TX0/1的bit 0), 接收到的首位数居被送入 Rx 寄存器的LSB位置(SPI_RX0/1的bit 0)..
[9:8]	TX_NUM	<b>发送/接收数量</b> 该寄存器用于标示一次成功传输中，传输的数量。 00 = 每次传输仅完成一次发送/接收 01 = 每次传输完成两次发送/接收 10 = 保留. 11 = 保留.

[7:3]	<b>TX_BIT_LEN</b>	<p><b>传输位长度</b></p> <p>该寄存器用于标示一次传输中，完成的传输长度，最高纪录32位。</p> <p>Tx_BIT_LEN = 0x01 ... 1位            Tx_BIT_LEN = 0x02 ... 2位            .....            Tx_BIT_LEN = 0x1f ... 31位            Tx_BIT_LEN = 0x00 ... 32位</p>
[2]	<b>TX_NEG</b>	<p><b>发送数据边沿反向位</b></p> <p>0 = SDO 信号在SPICLK的上升沿改变。            1 = SDO 信号在SPICLK的下降沿改变.</p>
[1]	<b>RX_NEG</b>	<p><b>接收数据边沿反向位</b></p> <p>0 = SDI 信号在SPICLK上升沿锁存            1 = SDI 信号在SPICLK下降沿锁存.</p>
[0]	<b>GO_BUSY</b>	<p><b>通讯或忙状态标志</b></p> <p>0 =在SPI正在通讯时对该位写0会使数据传输停止            1 =主机模式下，对该位写1开启SPI数据传输；从机模式下，对该位写1表明从机已准备好与主机的通讯。</p> <p><b>注:</b> 在对CNTRL寄存器的GO_GOBY置1之前，必须先配置相应的寄存器。在传输过程中再对其他寄存器进行配置，无法影响传输过程。</p>

## SPI 分频寄存器 (SPI\_DIVIDER)

寄存器	偏移量	R/W	描述	复位后的值
SPI_DIVIDER	SPIx_BA + 0x04	R/W	时钟分频寄存器(仅主机模式)	0x0000_0000

31	30	29	28	27	26	25	24
DIVIDER2[15:8]							
23	22	21	20	19	18	17	16
DIVIDER2[7:0]							
15	14	13	12	11	10	9	8
DIVIDER[15:8]							
7	6	5	4	3	2	1	0
DIVIDER[7:0]							

Bits	描述	
[31:16]	<b>DIVIDER2</b>	<p><b>时钟2分频寄存器</b> (仅主机模式)</p> <p>系统时钟, PCLK的第2个频率分频器, 产生串行时钟输出SPICLK. 可以根据下列方程获得期望的频率:</p> $f_{sclk} = \frac{f_{psclk}}{(DIVIDER2 + 1) * 2}$
[15:0]	<b>DIVIDER</b>	<p><b>时钟分频寄存器</b>(仅主机模式)</p> <p>系统时钟, PCLK的分频器产生串行时钟输出SPICLK。根据下列方程获得期望的频率</p> $f_{sclk} = \frac{f_{psclk}}{(DIVIDER + 1) * 2}$ <p>从机模式, 由主机提供的SPI时钟周期, 可以大于或等于PCLK周期的5倍. 换言之, SPI时钟的最大频率为从机PCLK的1/5.</p>

## SPI从机选择寄存器(SPI\_SSR)

寄存器	偏移量	R/W	描述	复位后的值
SPI_SSR	SPI0_BA + 0x08	R/W	从机选择寄存器	0x0000_0000

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

Bits	描述	
[31:6]	保留	保留
[5]	LTRIG_FLAG	<p><b>电平触发标志</b></p> <p>在从机模式下SS_LTRIG置位，该标志能够表示接收到的位数量是否达到要求。</p> <p>1: 接收数量和接收位长度达到TX_NUM 及TX_BIT_LEN内的值。</p> <p>0: 接收数量或接收位长度没有达到TX_NUM 及TX_BIT_LEN内的值。</p> <p>注：该位只读</p>
[4]	SS_LTRIG	<p><b>从机电平触发选择（从机模式）</b></p> <p>0: 从机输入边沿触发。该为默认值</p> <p>1: 从机选择由电平触发。根据 SS_LVL选择是高电平/低电平触发。</p>
[3]	AUTOSS	<p><b>自动从机选择(主机模式)</b></p> <p>0 = 该位清位，从机选择信号是否生效，由设置或清除SSR[0]寄存器决定。</p> <p>1 = 该位置位，SPISS0/1信号自动产生。这说明在SSR[0]寄存器内的从机选择信号，在置位GO_BUSY开始发送/接收时有SPI控制器声明，并且在传输结束后解除声明。</p>
[2]	SS_LVL	<p><b>从机选择激活电平</b></p> <p>该位决定SPISS0/1寄存器内从机选择信号根据哪个电平激活</p> <p>0 = SPISS0/1 从机选择低电平/下降沿时激活。</p>

		1 = The SPISS0/1从机选择高电平/上升沿时激活。
[1]	保留	保留
[0]	<b>SSR</b>	<p><b>从机选择寄存器(主机模式)</b></p> <p>当AUTOSS位被清除，对SSR位写1，将会激活SPISSx线，写0线上返回至非活动状态。</p> <p>当AUTOSS位被设置，对SSR位写1，将会使SPISSx线上在传输/接受数据时自动驱动至激活状态。在其他时间驱动至非活动状态(由SS_LVL决定激活电平)。</p> <p>注：SPISSx通常在从机模式下被定义为设备/从机选择输入。</p>

## SPII数据接收寄存器 (SPI\_RX)

寄存器	偏移量	R/W	描述	复位后的值
SPI_RX0	SPIx_BA + 0x10	R	数据接收寄存器 0	0x0000_0000
SPI_RX1	SPIx_BA + 0x14	R	数据接收寄存器1	0x0000_0000

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

Bits	描述
[31:0]	<p><b>数据接收寄存器</b></p> <p>数据接收寄存器内保存最后一次传输所接收的数据。数据的长度根据 SPI_CNTRL 寄存器内配置的长度决定。例如, Tx_BIT_LEN 设定为 0x08 且 Tx_NUM 设定为 0x0, Rx0[7:0] 内保存传输数据。</p> <p><b>注:</b> 数据接收寄存器为只读寄存器。</p>

## SPI 数据发送寄存器(SPI\_TX)

寄存器	偏移量	R/W	描述	复位后的值
SPI_TX0	SPIx_BA + 0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1	SPIx_BA + 0x24	W	数据发送寄存器1	0x0000_0000

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

Bits	描述	
[31:0]	TX	<p><b>数据发送寄存器</b></p> <p>数据发送寄存器内存储下一次被发送的数据。数据的长度根据CNTRL寄存器内配置的长度决定。例如，Tx_BIT_LEN设定为 0x08 且Tx_NUM 设定为0x0，Tx0[7:0] 内的数据将被发送。如果[Tx_BIT_LEN 设定为 0x00 且Tx_NUM 设定为 0x1，模块将用同种设置确保 2 个 32 位数据发送/接收，（顺序是 TX0[31:0], TX1 [31:0]）。</p>

**SPI 可调时钟类型寄存器(SPI\_VARCLK)**

寄存器	偏移量	R/W	描述	复位后的值
SPI_VARCLK	SPIx_BA + 0x34	R/W	多时钟类型寄存器	0x007F_FF87

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

Bits	描述	
[31:0]	VARCLK	<p><b>可调时钟类型</b></p> <p>该值为SPI时钟频率类型. VARCLK为'0', SPICLK的输出频率取决于DIVIDER的值. VARCLK 为 '1', SPICLK的输出频率取决于DIVIDER2. 参考寄存器 <a href="#">SPI_DIVIDER</a>.</p> <p>参考图.6-47 对应于可调时钟时序框图。</p> <p>注: 仅适用于 CLKP = 0.</p>



## 6.8 定时器控制器

### 6.8.1 简介

定时器控制器包括4组32位的定时器, TIMER0~TIMER3, 方便用户的定时器控制应用. 定时器模块可支持例如频率测量, 计数, 间隔时间测量, 时钟产生, 延迟时间等功能. 定时器可在计时溢出时产生中断信号, 也可在操作过程中提供计数的当前值。

### 6.8.2 特征

- 4 组 32-位定时器, 带24位向上定时器和一个8位的预分频计数器
- 每个定时器都有独立的时钟源
- 24位向上计数器, 通过TDR (定时器数据寄存器) 可读取
- 3种工作模式: 单脉冲模式(one-shot), 周期模式(periodic), 开关模式(toggle)操作模式

### 6.8.3 定时器控制器框图

每个通道带一个8位预分频计数器，一个24位向上计数器，一个24位比较寄存器和一个中断请求信号。参阅图 6.8-1的定时器控制框图。每个通道有3个时钟源选项，图 6.8-2 为时钟源控制功能。可软件编程配置8位预分频系数来确定24位向上计数器的时钟周期。

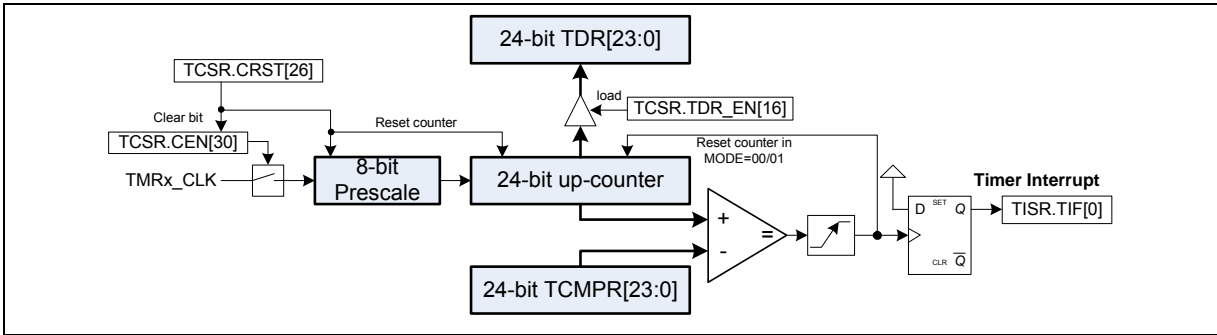


图 6.8-1 定时器控制器框图

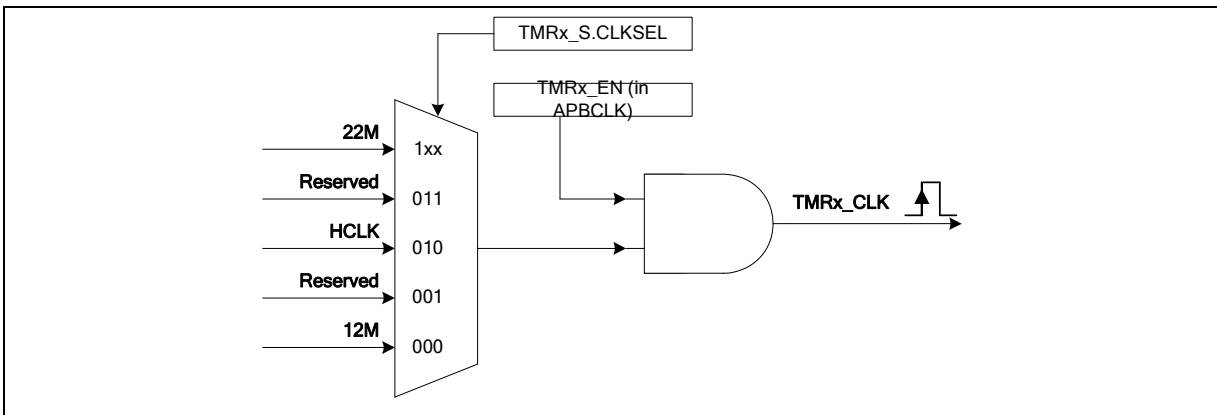


图 6.8-2 定时器控制的时钟源

#### 6.8.4 定时器操作模式

定时器控制器提供3种工作模式，单脉冲(one-shot)模式、周期(periodic)模式、开关 (toggle) 模式。每种操作功能模式如下所示：

##### 6.8.4.1 单脉冲模式

如果定时器工作在单脉冲模式且CEN (定时器使能位) 置1, 定时器的计数器开始计数. 一旦定时器计数器的值达到定时器比较寄存器 (TCMPR) 的值, 且IE (中断使能位) 置1, 则定时器中断标志置位, 产生中断信号并送到NVIC通知CPU. 表明定时器计数发生溢出. 如果IE (interrupt enable bit) 置0, 无中断信号发生. 在此工作模式下, 一旦定时器计数器的值达到定时器比较寄存器 (TCMPR) 的值, 定时器计数器的值返回初始值且CEN (定时器使能位)由定时器控制器清零. 一旦定时器计数器的值达到定时器比较寄存器(TCMPR) 的值, 定时器计数操作停止,。也就是说, 在编程比较寄存器 (TCMPR) 的值与CEN(定时器使能位)置1后, 定时器操作定时器计数和与TCMPR值的比较仅执行一次.因此, 该操作称为单脉冲模式.

##### 6.8.4.2 周期模式

如果定时器工作在周期模式且CEN (定时器使能位)置1, 定时器计数器开始计数. 一旦定时器计数器的值达到定时器比较寄存器 (TCMPR)的值, 且IE (中断使能位) 设置为1, 则定时器中断标志置位且产生中断信号, 并发送到NVIC通知CPU. 表示定时器计数溢出发生. 如果IE (中断使能位)设置为0, 无中断信号发生. 在该工作模式下, 一旦定时器计数器的值达到定时器比较寄存器(TCMPR) 的值, 定时器计数器的值返回计数初始值且CEN 保持为1 (持续使能计数). 定时器计数器重新开始计数. 如果软件清除中断标志, 一旦定时器计数器的值与定时器比较寄存器(TCMPR)的值匹配且IE (中断使能位)设置为1 由中断标志置位, 产生中断信号并送到NVIC再次通知CPU. 也就是说, 定时器操作定时器计数和与TCMPR比较功能是周期性进行的. 直到CEN设置为0, 定时器计数操作才会停止.中断信号的产生也是周期性的. 因此, 这种操作模式称为周期模式.

##### 6.8.4.3 开关模式

如果定时器工作在开关模式且CEN (定时器使能位)置1, 定时器计数器开始计数. 一旦定时器计数器的值与定时器比较寄存器TCMPR的值匹配时, 且IE (中断使能位)设置为 1, 则定时器中断标志置位,产生中断信号并送到NVIC通知CPU. 表示定时器发生计数溢出. 相应开关输出(tout) 信号置1. 在这种操作模式, 一旦定时器计数器的值与定时器比较寄存器TCMPR的值匹配, 定时器计数器的值返回到计数初始值且CEN 保持为 1 (持续使能计数). 定时器计数器重新开始计数. 如果中断标志由软件清除, 一旦定时器计数器的值与定时器比较寄存器中TCMPR的值匹配且IE (中断使能位) 置1, 则定时器中断标志置位, 发生中断信号, 并送到NVIC再次通知CPU. 相应开关输出 (tout)信号置0. 定时器计数操作在CEN设置为0之后才停止. 因此, t开关输出 (tout)信号 以50%的占空比反复改变. 所以这种操作模式称为开关模式.

## 6.8.5 定时器控制器寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>TMR_BA01 = 0x4001_0000</b>				
<b>TMR_BA23 = 0x4011_0000</b>				
<b>TCSR0</b>	TMR_BA01+00	R/W	Timer0控制和状态寄存器	0x0000_0005
<b>TCMPR0</b>	TMR_BA01+04	R/W	Timer0比较寄存器	0x0000_0000
<b>TISR0</b>	TMR_BA01+08	R/W	Timer0 中断状态寄存器	0x0000_0000
<b>TDR0</b>	TMR_BA01+0C	R	Timer0 数据寄存器	0x0000_0000
<b>TCSR1</b>	TMR_BA01+20	R/W	Timer1 控制和状态寄存器	0x0000_0005
<b>TCMPR1</b>	TMR_BA01+24	R/W	Timer1 比较寄存器	0x0000_0000
<b>TISR1</b>	TMR_BA01+28	R/W	Timer1 中断状态寄存器	0x0000_0000
<b>TDR1</b>	TMR_BA01+2C	R	Timer1 数据寄存器	0x0000_0000
<b>TCSR2</b>	TMR_BA23+00	R/W	Timer2 控制和状态寄存器	0x0000_0005
<b>TCMPR2</b>	TMR_BA23+04	R/W	Timer2 比较寄存器	0x0000_0000
<b>TISR2</b>	TMR_BA23+08	R/W	Timer2 中断状态寄存器	0x0000_0000
<b>TDR2</b>	TMR_BA23+0C	R	Timer2 数据寄存器	0x0000_0000
<b>TCSR3</b>	TMR_BA23+20	R/W	Timer3 控制和状态寄存器	0x0000_0005
<b>TCMPR3</b>	TMR_BA23+24	R/W	Timer3 比较寄存器	0x0000_0000
<b>TISR3</b>	TMR_BA23+28	R/W	Timer3 中断状态寄存器	0x0000_0000
<b>TDR3</b>	TMR_BA23+2C	R	Timer3 数据寄存器	0x0000_0000

## 定时器控制寄存器 (TCSR)

寄存器	偏移量	R/W	描述	复位后的值
TCSR0	TMR_BA01+000	R/W	Timer0 控制与状态寄存器	0x0000_0005
TCSR1	TMR_BA01+020	R/W	Timer1控制与状态寄存器	0x0000_0005
TCSR2	TMR_BA23+000	R/W	Timer2控制与状态寄存器	0x0000_0005
TCSR3	TMR_BA23+020	R/W	Timer3控制与状态寄存器	0x0000_0005

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

Bits	描述			
[31]	保留	保留		
[30]	CEN	<p>计数器使能位</p> <p>0 = 停止/暂停计数</p> <p>1 = 开始计数</p> <p>注1: 在停止状态, 设置CEN为1, 使能24位计数器从上次停止的计数值继续计数.</p> <p>注 2: 在单脉冲模式下 (MODE[28:27]=00b), 当相应的定时中断产生时 (IE[29]=1b), 该位由硬件自动清零.</p>		
[29]	IE	<p>中断使能</p> <p>0 = 禁用定时器中断</p> <p>1 = 使能定时器中断</p> <p>如果定时器中断使能, 当计数值与TCMPR寄存器内数值相同时, 触发中断.</p>		
[28:27]	MODE	<p>定时器工作模式</p> <table border="1" style="width: 100%; margin-top: 5px;"> <tr> <td style="width: 50%;">模式</td> <td style="width: 50%;">定时器工作模式</td> </tr> </table>	模式	定时器工作模式
模式	定时器工作模式			

		00	当定时器配置为单脉冲模式(one-shot)时, 定时器溢出仅触发中断一次(IE 使能),进入中断后CEN自动清除为0.
		01	T当定时器配置为周期模式(period)时, 定时器每次溢出都触发中断(IE 使能).
		10	定时器工作于开关mode. IE使能, 产生周期性的中断信号. 开关信号 (tout) 前后改变50%的占空比.
		11	保留
[26]	CRST	<b>计数器重置</b> 设置该位将重置定时器计数器, 预分频并使CEN为0. 0 = 无动作. 1 = 重置定时器的预分频计数器, 内部24位向上计数器和CEN位	
[25]	CACT	<b>定时器工作状态 (只读)</b> 该位表示当前定时器计数器的状态。 0 = 定时器未工作。 1 = 定时器工作中。	
[24:17]	保留	保留	
[16]	TDR_EN	<b>数据锁存使能</b> 当置位TDR_EN, TDR (Timer数据寄存器) 将不断更新为24位向上计数器的值 1 = 使能Timer数据寄存器 更新 0 = 禁用Timer数据寄存器 更新	
[15:8]	保留	保留	
[7:0]	PRESCALE	<b>预分频计数器</b> 时钟输入根据Prescale数值+1进行预分频。如果PRESCALE =0, 不进行预分频.	

## 定时器比较寄存器(TCMPR)

寄存器	偏移量	R/W	描述	复位后的值
TCMPR0	TMR_BA01+004	R/W	Timer0 比较寄存器	0x0000_0000
TCMPR1	TMR_BA01+024	R/W	Timer1比较寄存器	0x0000_0000
TCMPR2	TMR_BA23+004	R/W	Timer2比较寄存器	0x0000_0000
TCMPR3	TMR_BA23+024	R/W	Timer3比较寄存器	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:0]	<b>TCMP</b>	<p><b>定时器比较值</b></p> <p>TCMP是24位比较寄存器。当内部24位向上计数器的值与TCMP的值匹配时, 如果TCSR.IE[29]=1, 就产生定时器中断请求. TCMP的值为定时器计数周期.</p> <p>定时溢出周期= (Period of timer clock input) * (8-bit Prescale + 1) * (24-bit TCMP)</p> <p><b>注1:</b> 不能在TCMP里写0x0或0x1, 否则内核将运行到未知状态.</p> <p><b>注2:</b> 无论CEN为0或1, 软件向该寄存器写入新的值, TIMER将退出当前计数并使用新的值, 开始重新计数.</p>

### 定时器中断状态寄存器(TISR)

寄存器	偏移量	R/W	描述	复位后的值
TISR0	TMR_BA01+08	R/W	Timer0 中断状态寄存器	0x0000_0000
TISR1	TMR_BA01+28	R/W	Timer1 中断状态寄存器	0x0000_0000
TISR2	TMR_BA23+08	R/W	Timer2 中断状态寄存器	0x0000_0000
TISR3	TMR_BA23+28	R/W	Timer3 中断状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	保留	保留
[0]	TIF	定时器中断标志 定时器中断状态位。 当内部24位计数器与TCMP的值匹配时，TIF由硬件置位，写1清该位。



### Timer数据寄存器(TDR)

寄存器	偏移量	R/W	描述	复位后的值
TDR0	TMR_BA01+0C	R/W	Timer0数据寄存器	0x0000_0000
TDR1	TMR_BA01+2C	R/W	Timer1数据寄存器	0x0000_0000
TDR2	TMR_BA23+0C	R/W	Timer2数据寄存器	0x0000_0000
TDR3	TMR_BA23+2C	R/W	Timer3数据寄存器	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:0]	TDR	<p>定时器数据寄存器</p> <p>TCSR.TDR_EN 置1时，内部24位定时器的值加载到TDR中，用户可以读取该寄存器的值获取24位计时器的值。</p>

## 6.9 看门狗定时器 (WDT)

### 6.9.1 简介

看门狗定时器是在软件运行至未知状态时执行系统复位功能，可以防止系统无限制地挂机，除此之外，看门狗定时器还可将CPU由掉电模式唤醒。看门狗定时器包含一个18位的自由运行的计数器，可编程其定时溢出间隔。表 6-6 为看门狗定时溢出间隔选择，图6.9-1为看门狗中断信号与复位信号的时序。

设置WTE(WDTCR[7])使能看门狗定时器和WDT计数器开始计数。当计数器达到选择的定时溢出间隔，如果看门狗定时器中断使能位WTIE置位，看门狗定时器中断标志WTIF被立即置位，并请求WDT中断，同时，在溢出有一个有指定延时( $1024 * T_{WDT}$ )。用户必须在特定的延时时间结束前设置WTR(WDTCR[0]) (看门狗定时器复位)为高，重置18位WDT计数器，防止CPU复位。WTR在WDT计数重置后自动由硬件清零。通过设置WTIS(WDTCR[10:8])可选择8个定时溢出间隔。如果在特定延迟时间终止后，WDT计数没有被清零，看门狗定时将置位看门狗定时器重置标志(WTRF)为高并使CPU复位。这个复位将持续63个WDT时钟，然后CPU重启，并从复位向量(0x0000 0000)执行程序。看门狗复位后WTRF位不会被清除。用户可用软件查询WTRF来识别复位源。WDT还提供唤醒功能。当芯片掉电，且看门狗唤醒使能位(WDTR[4])置位时，如果WDT计数器在特定延时后没有被清零，芯片就会由掉电状态唤醒。

WTIS	Timeout Interval Selection $T_{TIS}$	Interrupt周期 $T_{INT}$	WTR Timeout Interval (WDT_CLK=12 MHz) Min. $T_{WTR}$ ~ Max. $T_{WTR}$
000	$2^4 * T_{WDT}$	$1024 * T_{WDT}$	1.33 us ~ 86.67 us
001	$2^6 * T_{WDT}$	$1024 * T_{WDT}$	5.33 us ~ 90.67 us
010	$2^8 * T_{WDT}$	$1024 * T_{WDT}$	21.33 us ~ 106.67 us
011	$2^{10} * T_{WDT}$	$1024 * T_{WDT}$	85.33 us ~ 170.67 us
100	$2^{12} * T_{WDT}$	$1024 * T_{WDT}$	341.33 us ~ 426.67 us
101	$2^{14} * T_{WDT}$	$1024 * T_{WDT}$	1.36 ms ~ 1.45 ms
110	$2^{16} * T_{WDT}$	$1024 * T_{WDT}$	5.46 ms ~ 5.55 ms
111	$2^{18} * T_{WDT}$	$1024 * T_{WDT}$	21.84 ms ~ 21.93 ms

表 6.9-1看门狗定时溢出间隔选择

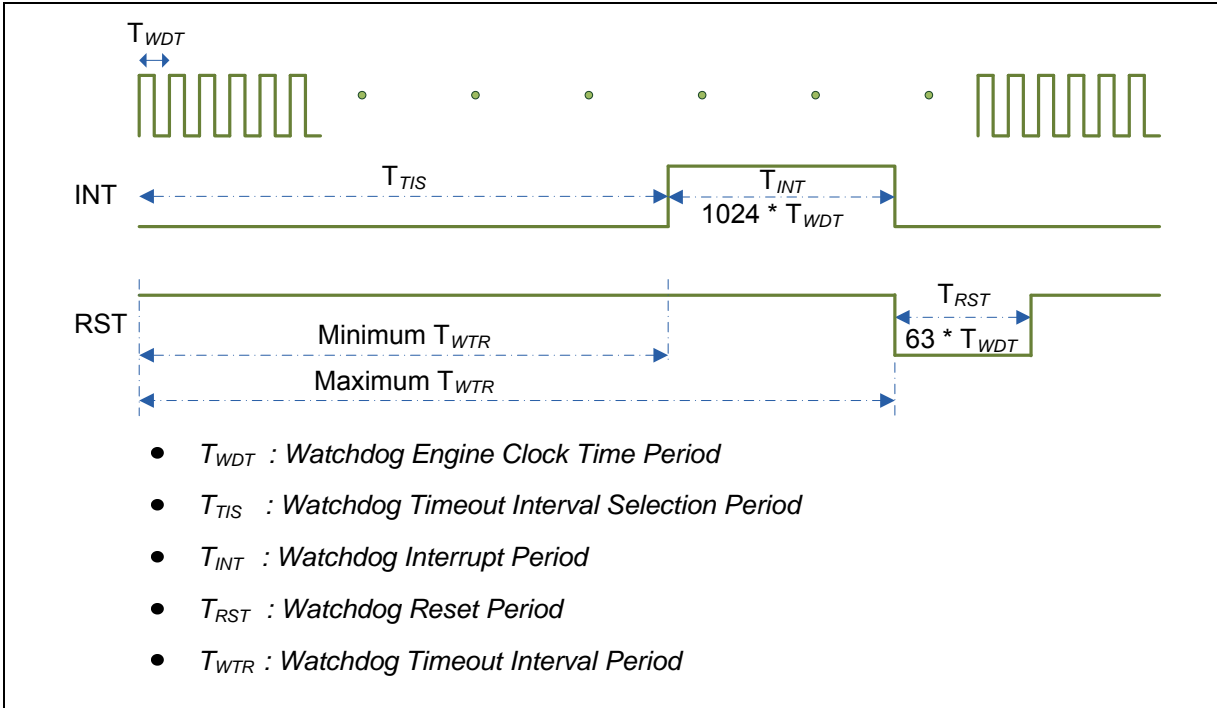


图 6.9-1 中断时序与复位信号时序

## 6.9.2 特征

- 18-位自由运行的计数器以防止CPU在在延迟时间结束之前发生看门狗定时器复位。
- 溢出时间间隔可选( $2^4 \sim 2^{18}$ )，溢出时间范围在 86.67 us ~ 21.93 ms (如果WDT\_CLK = 12 MHz)。
- 复位周期 = WDT\_CLK \* 63, 如果WDT\_CLK = 12 MHz。

## 6.9.3 WDT 框图

看门狗定时器时钟控制和框图如下。

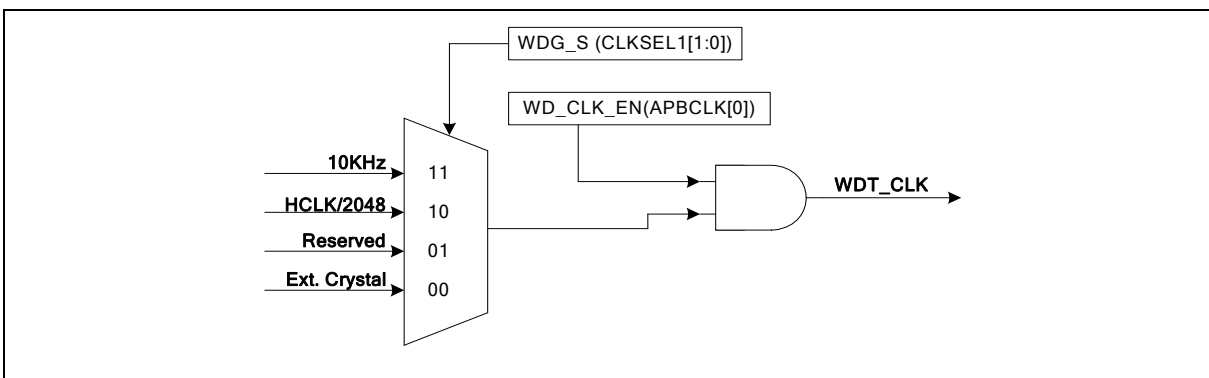


图 6.9-2 看门狗定时器时钟控制

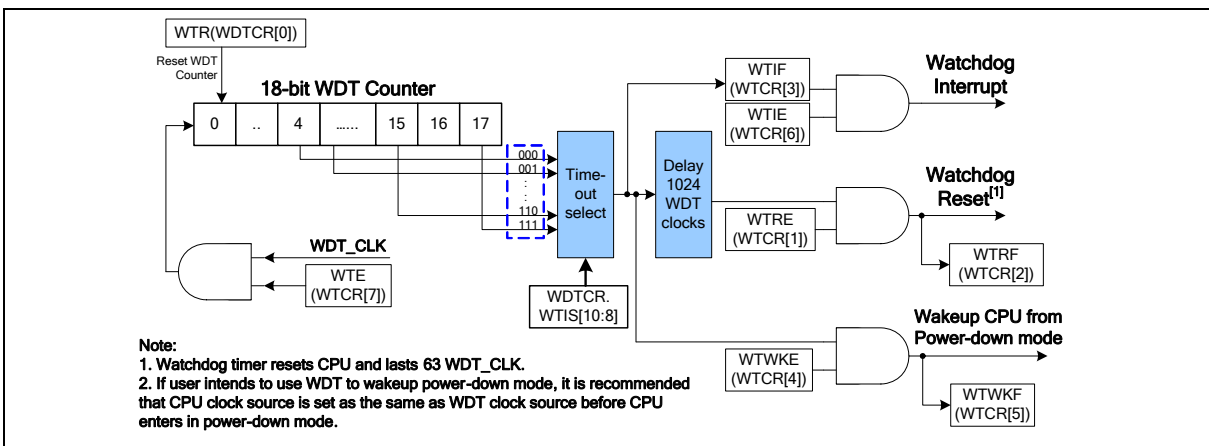


图 6.9-3 看门狗定时框图



## 6.9.4 看门狗定时器控制寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>WDT_BA = 0x4000_4000</b>				
WTCR	WDT_BA+00	R/W	看门狗定时器控制寄存器	0x0000_0700

### 看门狗定时器控制寄存器r (WTCR)

寄存器	偏移量	R/W	描述	复位后的值
WTCR	WDT_BA+000	R/W	看门狗定时器控制寄存器	0x0000_0700

注:

该寄存器所有位都写保护。要编程时，需要开锁时序，依次向寄存器REGWRPROT写入“59h”，“16h”，与“88h”，RegLockAddr 的地址为GCR\_BA + 0x100

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留					WTIS		
7	6	5	4	3	2	1	0
WTE	WTIE	WTWKF	WTWKE	WTIF	WTRF	WTRE	WTR

Bits	描述									
[31:11]	保留	保留								
[10:8]	WTIS	看门狗定时器间隔选择 选择看门狗定时器的定时溢出间隔。								
		<table border="1" style="width: 100%; text-align: center;"> <tr> <th>WTIS</th> <th>溢出间隔选择</th> <th>中短周期</th> <th>WTR溢出间隔 (WDT_CLK=12 MHz)</th> </tr> <tr> <td>000</td> <td><math>2^4 * T_{WDT}</math></td> <td><math>(2^4 + 1024) * T_{WDT}</math></td> <td>1.33 us ~ 86.67 us</td> </tr> </table>	WTIS	溢出间隔选择	中短周期	WTR溢出间隔 (WDT_CLK=12 MHz)	000	$2^4 * T_{WDT}$	$(2^4 + 1024) * T_{WDT}$	1.33 us ~ 86.67 us
		WTIS	溢出间隔选择	中短周期	WTR溢出间隔 (WDT_CLK=12 MHz)					
000	$2^4 * T_{WDT}$	$(2^4 + 1024) * T_{WDT}$	1.33 us ~ 86.67 us							

		001	$2^6 * T_{WDT}$	$(2^6 + 1024) * T_{WDT}$	5.33 us ~ 90.67 us
		010	$2^8 * T_{WDT}$	$(2^8 + 1024) * T_{WDT}$	21.33 us ~ 106.67 us
		011	$2^{10} * T_{WDT}$	$(2^{10} + 1024) * T_{WDT}$	85.33 us ~ 170.67 us
		100	$2^{12} * T_{WDT}$	$(2^{12} + 1024) * T_{WDT}$	341.33 us ~ 426.67 us
		101	$2^{14} * T_{WDT}$	$(2^{14} + 1024) * T_{WDT}$	1.36 ms ~ 1.45 ms
		110	$2^{16} * T_{WDT}$	$(2^{16} + 1024) * T_{WDT}$	5.46 ms ~ 5.55 ms
		111	$2^{18} * T_{WDT}$	$(2^{18} + 1024) * T_{WDT}$	21.84 ms ~ 21.93 ms
[7]	<b>WTE</b>	<b>看门狗定时器使能</b> 0 = 禁用看门狗定时器功能(该动作重置内部计数器) 1 = 使能看门狗定时器			
[6]	<b>WTIE</b>	<b>看门狗定时器中断使能</b> 0 = 禁用看门狗定时器中断 1 = 使能看门狗定时器中断			
[5]	<b>WTWKF</b>	<b>看门狗定时器唤醒标志</b> 如果看门狗定时器引起CPU从掉电模式下唤醒，该位将被置高。 0 = 看门狗定时器不能引起CPU唤醒。 1 = CPU 由休眠或掉电模式被看门狗定时溢出唤醒 注：写1清零。			
[4]	<b>WTWKE</b>	<b>看门狗定时器唤醒功能使能位</b> 0 = 禁用看门狗唤醒CPU功能。 1 = 使能看门狗唤醒CPU功能。			
[3]	<b>WTIF</b>	<b>看门狗定时器中断标志</b> 如果看门狗定时器中断使能，该位由硬件置位表示看门狗定时器中断已发生。 0= 不发生看门狗定时器中断 1= 发生看门狗定时器中断 注：写1清零。			
[2]	<b>WTRF</b>	<b>看门狗定时器复位标志</b> 当看门狗定时器引发复位，该位被置位，通过读取该位可以确认复位是否由看门狗引起。该位软件写1清零。如果 <b>WTRE</b> 禁用，看门狗定时器溢出对该位无影响。 0 = 复位不是由看门狗定时器产生。 1 = 看门狗定时器引发复位			

		注: 写1清零.
[1]	WTRE	<b>看门狗定时器复位使能</b> 设定该位使能看门狗定时器复位功能。 0 = 禁用看门狗定时器复位功能 1 = 使能看门狗定时器复位功能
[0]	WTR	<b>清看门狗定时器</b> 设置该位清看门狗定时器。 0: 写0无效 1: 重置看门狗定时器的内容 <b>NOTE:</b> 写1清零.

## 6.10 UART接口控制器

NuMicro M051™ 提供2个通用异步收/发器 (UART) 通道, UART0~1支持普通速度, 支持流控制.

### 6.10.1 简介

通用异步收/发器(UART) 对从同外设收到数据的时候执行串到并的转换, 对从CPU收到数据的时候执行并到串的转换. 该串口同时支持IrDA SIR 功能和RS-485模式. 每个UART通道有**5种类型的中断**, 它们是, 发送FIFO 空中断(Int\_THRE), 接收阈值到达中断(Int\_RDA), 线状态中断 (奇偶校验错误或者帧错误r或者打断中断) (Int\_RLS), 接收缓冲器溢出中断中断(Int\_Tout), 调制解调器/唤醒状态中断(Int\_Modem), 缓存错误中断(INT\_BUF\_ERR). 中断号12 (中断向量为28支持UART0的中断), 中断号13 (中断向量为29) 支持UART1的中断, 参考嵌套向量中断控制器.

UART0~1内嵌一个15字节发送FIFO (TX\_FIFO) 和一个15字节接收 FIFO (RX\_FIFO)。CPU可以随时读UART的状态.返回的状态信息包括正在被UART执行的传输操作的类型和条件, 在接收数据时还可能发生3个错误(奇偶校验错误、帧错误、钳制中断)状况。UART包括一个可编程的波特率发生器, 它可以将输入时钟分频来得到收发器需要的时钟. 波特率公式是  $\text{Baud Rate} = \text{UART\_CLK} / M * [\text{BRD} + 2]$ . 其中M和BRD在波特率分频寄存器UA\_BAUD中配置. 图表6.10-1和6.10-2分别列出了波特率方程和UART波特率设置表。

Mode	DIV_X_EN	DIV_X_ONE	Divider X	BRD	波特率公式
0	0	0	B	A	$\text{UART\_CLK} / [16 * (A+2)]$
1	1	0	B	A	$\text{UART\_CLK} / [(B+1) * (A+2)]$ , B must $\geq 8$
2	1	1	Don't care	A	$\text{UART\_CLK} / (A+2)$ , A must $\geq 3$

表 6.10-1 UART 波特率方程

系统时钟 = 22.1184 MHz			
波特率	模式0	模式1	模式2
921600	x	A=0,B=11	A=22
460800	A=1	A=1,B=15 A=2,B=11	A=46
230400	A=4	A=4,B=15 A=6,B=11	A=94
115200	A=10	A=10,B=15 A=14,B=11	A=190
57600	A=22	A=22,B=15 A=30,B=11	A=382



38400	A=34	A=62,B=8 A=46,B=11 A=34,B=15	A=574
19200	A=70	A=126,B=8 A=94,B=11 A=70,B=15	A=1150
9600	A=142	A=254,B=8 A=190,B=11 A=142,B=15	A=2302
4800	A=286	A=510,B=8 A=382,B=11 A=286,B=15	A=4606

表 6.10-2 UART波特率设置表

UART0与UART1 控制器支持自动流控制功能，它使用 2 种低电平信号，/CTS (clear-to-send, 允许发送)和 /RTS (request-to-send, 请求发送)，来控制UART 和外部驱动器(ex: Modem)之间的数据流传递。UART 将禁用接收数据直到 UART 向外发出/RTS信号。当 Rx FIFO 的值和RTS\_TRI\_LEV (UA\_FCR [19:16]) 相等，/RTS信号不再发出。当UART控制器从外部驱动器侦测到 /CTS，UART 向外发送数据。如果 /CTS 未被侦测，UART 将不向外发送数据。

UART 控制器提供 串行 IrDA (SIR, 串行红外) 功能 (用户需置位rDA\_EN (UA\_FUN\_SEL[1:0])使能 IrDA 功能)。SIR 配置短程红外异步串行传输模式为1 开始位, 8 数据位, 和1 停止位。最大数据速率为 115.2 Kbps (半双工)。IrDA SIR 包括 IrDA SIR 编码/解码协议。IrDA SIR 只是半双工协议。因此不能同时传输和接收数据。IrDA SIR 物理层规定在传输和接收之间至少要有10ms 输出延时。该特性必须由软件执行。

UART控制的另一功能是支持RS-485 9位模式，由RTS控制方向或通过软件编程GPIO (P0.3 对应于 RTS0 and P0.1 对应于 RTS 1) 执行该功能。RS-485模式通过设置UA\_FUN\_SEL 选定。在RS-485模式下，RX与TX的许多特性与UART相同。

### 6.10.2 特性

- 全双工，异步通信
- 分别接收/发送15字节 (UART0/UART1) FIFO数据装载区
- 支持硬件自动流控制/流控制功能(CTS, RTS)和可编程的RTS流控制触发电平(UART0 与UART1 支持)
- 可编程的接收缓冲触发电平
- 每个通道都支持独立的可编程的波特率发生器
- 支持CTS 唤醒功能(UART0 与 UART1 支持)
- 支持7位接收缓冲计时溢出检测功能
- 通过设置UA\_TOR [DLY] 可以编程在上一个停止与下一个开始位之间的延迟时间
- 支持打断错误，帧错误，奇偶校验错误和接收/发送缓冲溢出检测功能
  - ◆ 可编程串行接口特性
- 可编程的数据位, 5, 6, 7, 8位
- 可编程的奇偶校验位, 偶校验、奇校验、无校验位或stick校验位 发生和检测
- 可编程停止位, 1, 1.5, 或 2 停止位 发生
- 支持IrDA SIR 功能
- 普通模式下支持 3/16位持续时间
- 支持RS-485 模式.
- 支持 RS-485 9位模式
- 支持由RTS提供的硬件或软件直接使能控制

### 6.10.3 UART 框图

UART 时钟控制和框图如图图6.10-1和 图6.10-2.

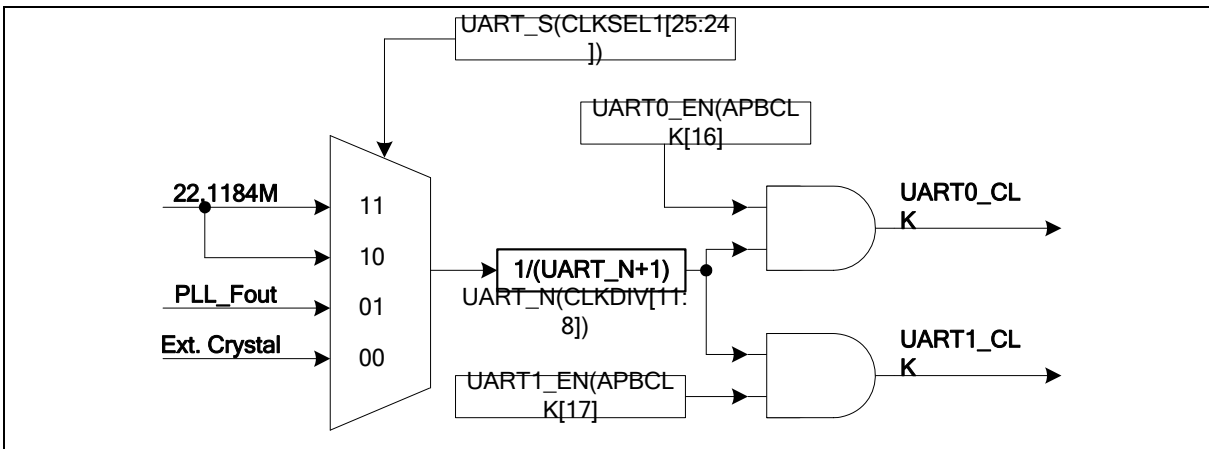


图 6.10-1 UART 时钟控制框图

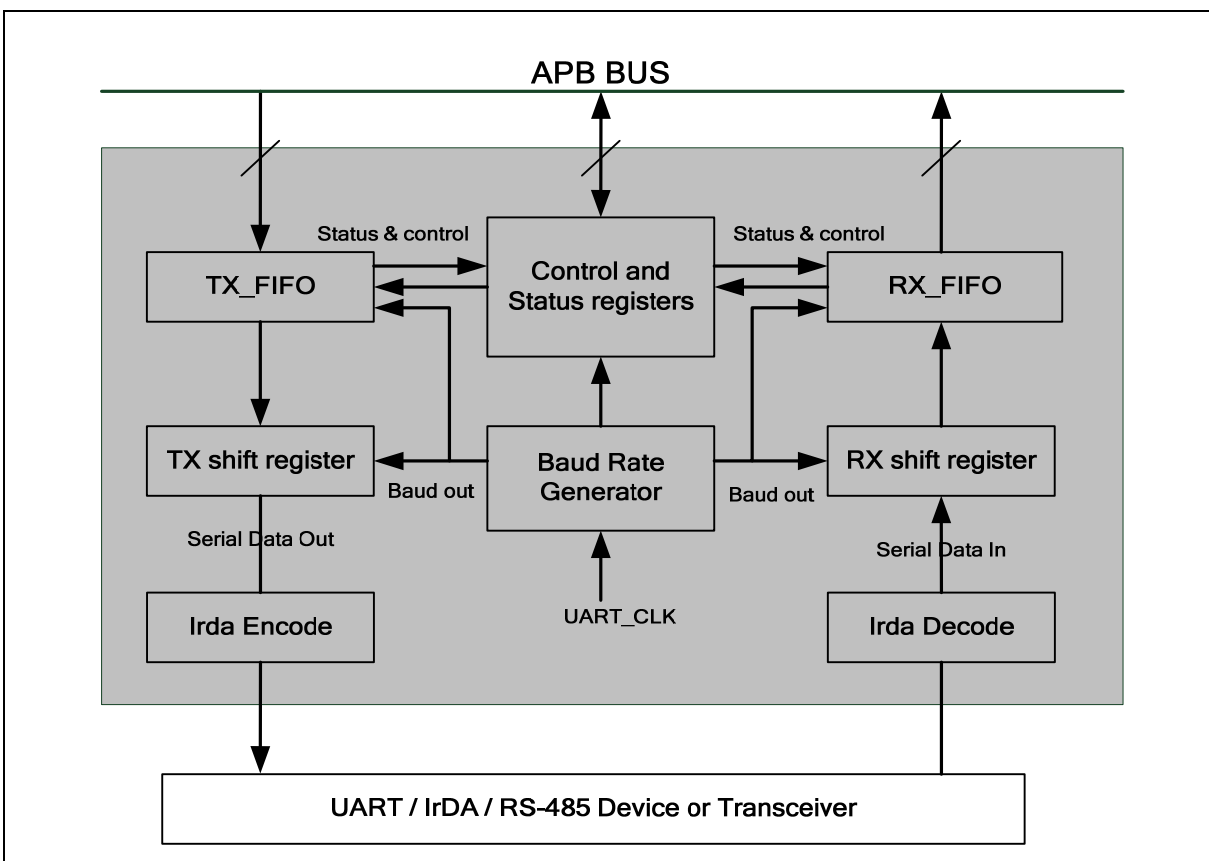


图 6.10-2 UART 框图

**TX\_FIFO**

发送用一个15字节的FIFO做缓存来降低向CPU申请的中断数量.

**RX\_FIFO**

接收用一个15字节(每个字节加3个比特的错误比特)的FIFO做缓存来降低向CPU申请的中断数量.

**TX移位寄存器**

此模块控制移动准备发送的数据串行输出.

**RX移位寄存器**

此模块控制串行移动接收到的数据.

**Modem控制寄存器**

寄存器控制与MODEM 或者数据传输转换器或者一个MODEM模拟器的接口.

**波特率发生器**

将外部时钟除以一个除数来分频产生期望的内部时钟,参考波特率方程.

**IrDA 编码**

IrDA 编码控制模块.

**IrDA 解码**

IrDA 解码控制模块.

**控制和状态寄存器**

此寄存器设定, 包括 FIFO 控制寄存器 (UA\_FCR), FIFO 状态寄存器 (UA\_FSR), 和线控制寄存器 (UA\_LCR). 时间溢出控制寄存器 (UA\_TOR) 应用于识别时间溢出中断控制. 该寄存器包括中断控制使能寄存器 (UA\_IER) 和中断状态寄存器 (UA\_ISR) 来使能或者禁用中断响应并且识别发生的中断. 有7种中断: FIFO为空中断 (INT\_THRE), 接收阈值到达中断 (INT\_RDA), 线状态中断 (校验错误, 帧错误和打断中断) (INT\_RLS), 定时溢出中断 (INT\_Tout), MODEM/唤醒状态中断 (INT\_Modem) 和缓冲错误中断 (INT\_Buf\_Err) .

下图为自动流控制框图.

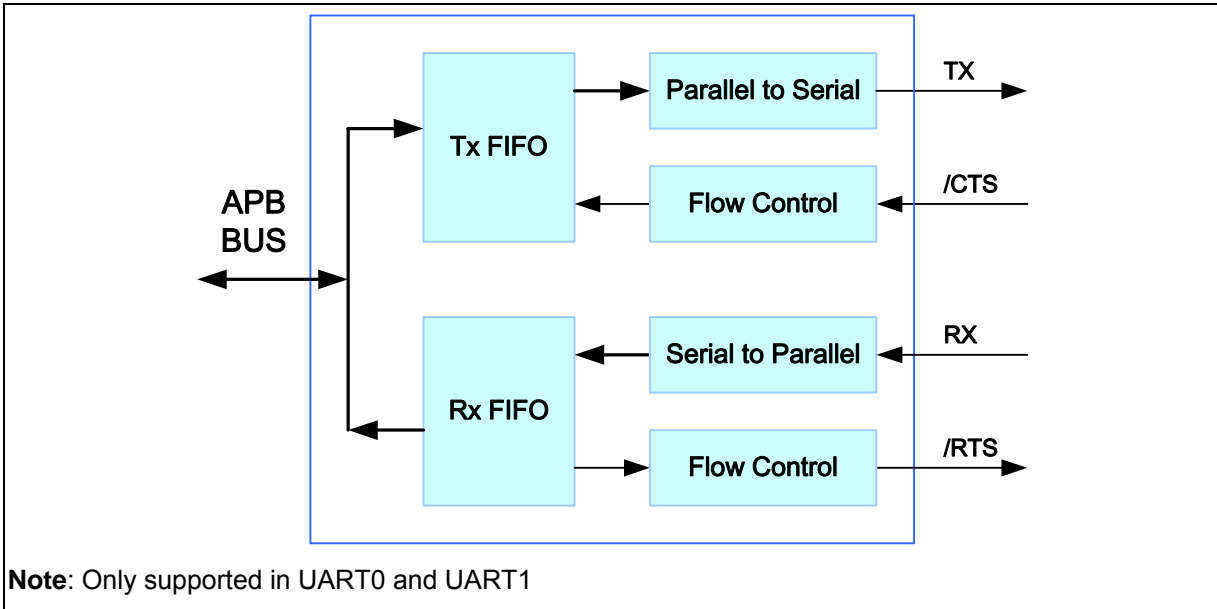


图 6.10-3 自动流控制框图

### 6.10.4 IrDA 模式

UART 支持 IrDA SIR (串行红外) 传输编码 和接收解码, IrDA 模式通过设定 **UA\_FUN\_SEL** 寄存器的 **IrDA\_EN** 位.

IrDA 模式下, **UA\_BAUD[DIV\_X\_EN]** 寄存器需禁用.

波特率 =  $\text{Clock} / (16 * \text{BRD})$ , BRD 为 **UA\_BAUD** 寄存器中的波特率分频器.

下图为IrDA控制框图.

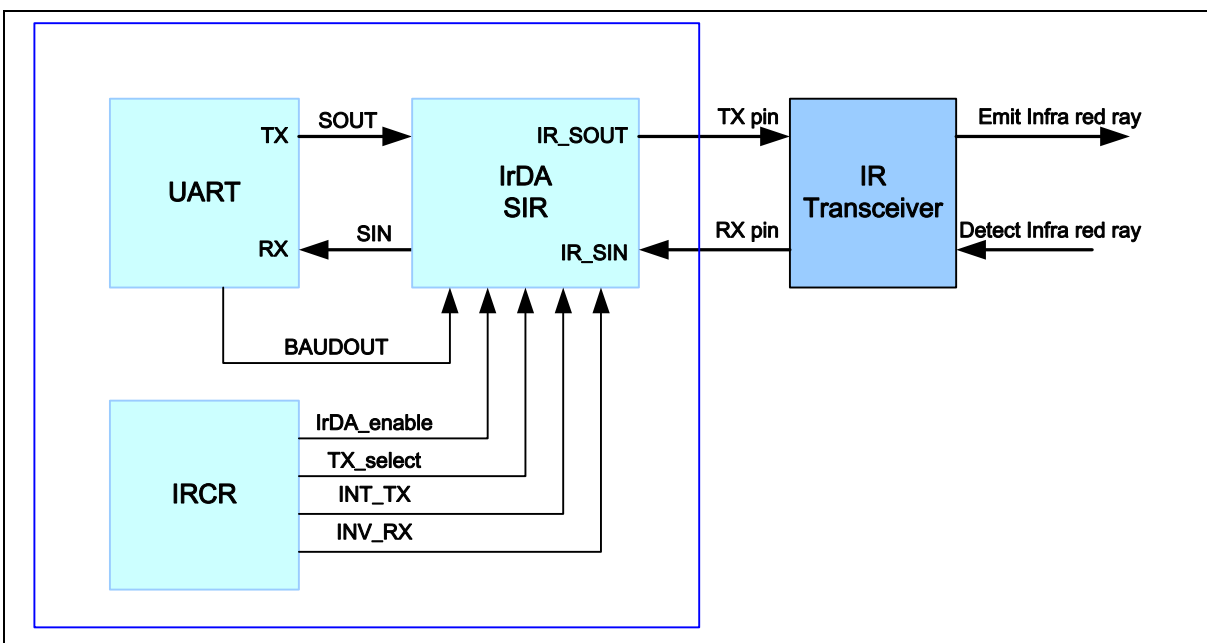


图 6.10-4 IrDA 框图

#### 6.10.4.1 IrDA SIR 传输编码

IrDA SIR发送编码器以非归零(NRZ) 调制方式从 UART 输出比特流。IrDA SIR 物理层指定使用归零调制方式, 反向 (RZI) 调制配置表现在 0 作为下列脉冲. 被调整的脉冲输出脉冲到外部输出驱动器和红外发射二极管

在正常模式下, 传输脉冲的宽度为 3/16 波特率周期.

#### 6.10.4.2 IrDA SIR接收解码

IrDA SIR 接收解码器以归零return-to-zero方式解调由输入探测器输入的比特流, 并输出 NRZ比特流到 UART 作为数据输入. 解码器在空闲模式输入通常为高. (因此, IRCR bit 5 默认设定为高)

当解码器输入为低时, 起始位将被检测到。

### 6.10.4.3 IrDA SIR 运作

IrDA SIR 编码/解码 提供 UART 数据流和半双工串行 SIR接口间互相转换的功能。图6. 10-15是IrDA编码/解码 波形图

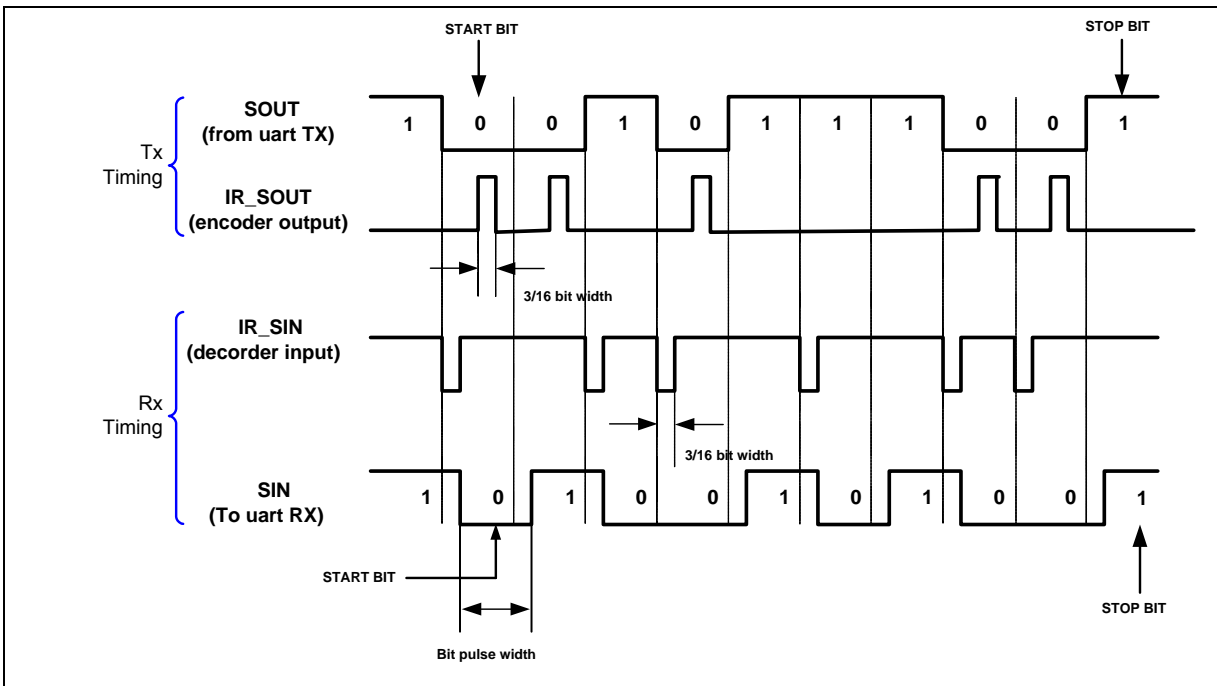


图 6.10-5 IrDA TX/RX 时序框图

### 6.10.5 RS-485 模式

UART 支持RS-485 9位模式. 设置UA\_FUN\_SEL选择RS-485模式. RS-485 RTS驱动控制通过异步串口的RTS控制信号, 使能RS-485 驱动器实现. . RS-485 模式, RX与TX的许多特性与UART一样.

RS-485 模式下, 通过设置校验位 (第9位)为1来识别一个地址字符, 控制器可以配置成可寻址的RS-485从机模式和RS-485主机发送器. 对于数据字符, 校验位设置为0. 软件可通过设置寄存器UA\_LCR 控制第9位 (PBE, EPE 和 SPE置位, 第9位发送0, PBE 和 SPE 置位, EPE清零, 第9位发送1). 该控制器支持三种操作模式: RS-485 普通模式(NMM), RS-485 自动地址识别模式 (AAD) 和RS-485 自动方向控制模式(AUD), 可通过编程UA\_RS-485\_CSR寄存器选择其中一种工作模式, 通过设置UA\_TOR [DLY] 可以设置上一个停止与下一个开始位之间的延迟时间. 下图为RS-485帧结构

#### RS-485 普通模式 (NMM)

RS-485 普通模式, 首先必须软件配置在地址位之前检测到的数据是否存储于RX-FIFO中. 如果想软件忽略地址位之前检测到的所有数据, 则接下来是设置UART\_FCR[RS485\_RX\_DIS]和使能UA\_RS-485[RS485\_NMM], 这样, 接收器在忽略所有数据直至检测到地址字节 (bit9=1) 并将地址字节数据存储于RX-FIFO中. . 如果想软件接收检测到地址位之前的所有数据, 接下来就是禁止UART\_FCR [RS485\_RX\_DIS]和使能UA\_RS-485[RS485\_NMM], 这样, 接收器就会接收所有数据. 如果检测到地址位, 就会向CPU发出一个中断申请, 根据UA\_RS-485\_FCR [RX\_DIS]的设置由软件决定接收器是否接收后面的数据位. 如果使能接收器接收, 所有接收的数据都将被接收并存储于RX-FIFO中, 如果接收器接收, 则接收到的所有数据都将被忽略直至下一个地址位被检测到. 若软件设置UA\_RS-485禁用接收器, 当检测到下一个地址字节, 控制器清UA\_RS-485\_FCR [RX\_DIS]位, 地址字节数据存储到RX-FIFO.

#### RS-485 自动地址识别模式 (AAD)

RS-485 自动地址识别模式, 接收在检测到地址字节 (bit9=1) 和地址字节数据与UA\_ALT\_CSR [ADDR\_MATCH]的值相匹配之前, 忽略所有数据. 地址字节数据存储于RX-FIFO. 所有接收字节数据将被接受, 存储于RX-FIFO 直到地址字节或数据字节不匹配UA\_ALT\_CSR [ADDR\_MATCH] 的值为止.

#### RS-485 自动方向模式 (AUD)

RS-485控制器的另一个功能是自动方向控制. RS-485 通过RTS驱动控制异步串口的控制信号, 使能RS-485 驱动器实现. RS-485 模式. RTS 连接到RS-485 驱动器, 使能RTS线为高 (逻辑1),使能RS-485 驱动器. 设置RTS为低 (逻辑0), 使驱动器进入tri-state状态. 用户通过设置寄存器UA\_MCR 中的LEV\_RTS位改变 RTS 驱动电平.

编程流程示例:

1. 设置寄存器UA\_FUN\_SEL中的FUN\_SEL位选择RS-485.
2. 设置寄存器UA\_FCR 中的RX\_DIS 位使能或禁用RS-485 接收器
3. 设置RS-485\_NMM 或 RS-485\_AAD 模式.
4. 如果选择RS-485\_AAD 模式, ADDR\_MATCH设置成自动地址匹配值.
5. 设置RS-485\_AUD选择自动方向控制.



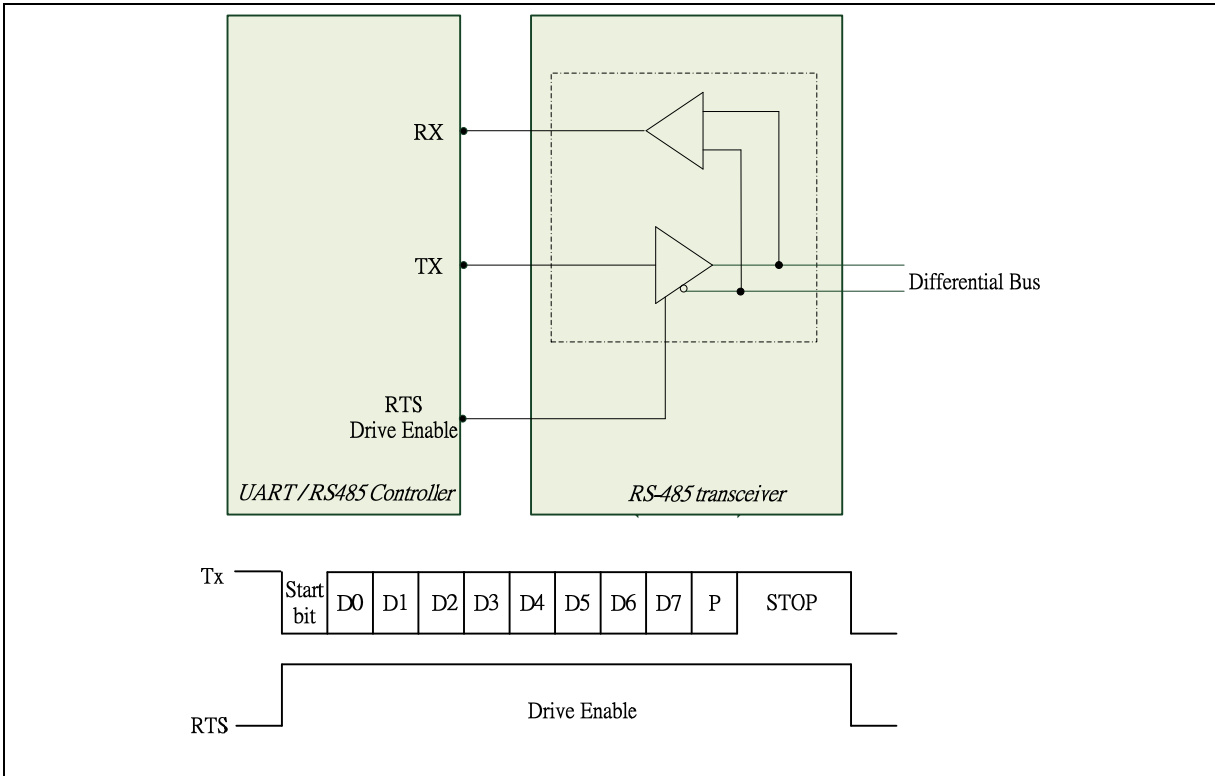


图 6.10-6 RS-485 帧结构



## 6.10.6 UART 接口控制寄存器映射

R:只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>UART 基地址:</b>				
<b>Channel0 : UART0_BA = 0x4005_0000</b>				
<b>Channel1 : UART1_BA = 0x4015_0000</b>				
<b>UA_RBR</b>	UART0_BA+0x00	R	UART0接收数据缓存寄存器.	Undefined
	UART1_BA+0x00	R	UART1接收数据缓存寄存器.	Undefined
<b>UA_THR</b>	UART0_BA+0x00	W	UART0发送保持寄存器.	Undefined
	UART1_BA+0x00	W	UART1发送保持寄存器.	Undefined
<b>UA_IER</b>	UART0_BA+0x04	R/W	UART0中断使能寄存器.	0x0000_0000
	UART1_BA+0x04	R/W	UART1中断使能寄存器.	0x0000_0000
<b>UA_FCR</b>	UART0_BA+0x08	R/W	UART0 FIFO 控制寄存器.	0x0000_0000
	UART1_BA+0x08	R/W	UART1 FIFO 控制寄存器.	0x0000_0000
<b>UA_LCR</b>	UART0_BA+0x0C	R/W	UART0 Line 控制寄存器.	0x0000_0000
	UART1_BA+0x0C	R/W	UART1 Line 控制寄存器.	0x0000_0000
<b>UA_MCR</b>	UART0_BA+0x10	R/W	UART0 Modem 控制寄存器.	0x0000_0000
	UART1_BA+0x10	R/W	UART1 Modem 控制寄存器.	0x0000_0000
<b>UA_MSR</b>	UART0_BA+0x14	R/W	UART0 Modem 状态寄存器.	0x0000_0000
	UART1_BA+0x14	R/W	UART1 Modem 状态寄存器.	0x0000_0000
<b>UA_FSR</b>	UART0_BA+0x18	R/W	UART0 FIFO 状态寄存器.	0x1040_4000
	UART1_BA+0x18	R/W	UART1 FIFO 状态寄存器.	0x1040_4000
<b>UA_ISR</b>	UART0_BA+0x1C	R/W	UART0 Interrupt 状态寄存器.	0x0000_0002
	UART1_BA+0x1C	R/W	UART1 Interrupt 状态寄存器.	0x0000_0002
<b>UA_TOR</b>	UART0_BA+0x20	R/W	UART0 定时溢出寄存器	0x0000_0000
	UART1_BA+0x20	R/W	UART1定时溢出寄存器	0x0000_0000
<b>UA_BAUD</b>	UART0_BA+0x24	R/W	UART0 波特率分频寄存器	0x0F00_0000

	UART1_BA+0x24	R/W	UART1 波特率分频寄存器	0x0F00_0000
UA_IRCR	UART0_BA+0x28	R/W	UART0 IrDA 控制寄存器.	0x0000_0040
	UART1_BA+0x28	R/W	UART1 IrDA 控制寄存器.	0x0000_0040
UA_ALT_CSR	UART0_BA+0x2C	R/W	UART0 控制/状态寄存器	0x0000_0000
	UART1_BA+0x2C	R/W	UART1控制/状态寄存器	0x0000_0000
UA_FUN_SEL	UART0_BA+0x30	R/W	UART0 功能选择寄存器	0x0000_0000
	UART1_BA+0x30	R/W	UART1功能选择寄存器	0x0000_0000

## 6.10.7 UART接口控制寄存器描述

### 接收缓冲寄存器(UA\_RBR)

寄存器	偏移量	R/W	描述	复位后的值
UA_RBR	UART0_BA+0x00	R	UART0接收缓冲寄存器	Undefined
	UART1_BA+0x00	R	UART1接收缓冲寄存器	Undefined

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

Bits	描述	
[31:8]	保留	保留
[7:0]	RBR	接收缓冲寄存器（只读） 通过读此寄存器, UART 将返回一组从 Rx引脚接收到的 8-位数据 (LSB优先).

## 发送保持寄存器(UA\_THR)

寄存器	偏移量	R/W	描述	复位后的值
UA_THR	UART0_BA+0x00	W	UART0发送保持寄存器	Undefined
	UART1_BA+0x00	W	UART1发送保持寄存器	Undefined

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

Bits	描述	
[31:8]	保留	保留
[7:0]	THR	发送保持寄存器 通过写该寄存器, UART 将通过Tx引脚 (LSB优先) 发送 8-位数据.

## 中断使能寄存器(UA\_IER)

寄存器	偏移量	R/W	描述	复位后的值
UA_IER	UART0_BA+0x04	R/W	UART0中断使能寄存器	0x0000_0000
	UART1_BA+0x04	R/W	UART1中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留		AUTO_CTS_EN	AUTO_RTS_EN	TIME_OUT_EN	保留		
7	6	5	4	3	2	1	0
保留	WAKE_EN	BUF_ERR_IEN	RTO_IEN	MODEM_IEN	RLS_IEN	THRE_IEN	RDA_IEN

Bits	描述	
[31:14]	保留	保留
[13]	AUTO_CTS_EN	<b>CTS 自动流控制使能</b> 1 = 使能 CTS 自动流控制. 0 = 禁用 CTS 自动流控制. 当 CTS 自动流控制使能, 当CTS输入有效时UART将向外部驱动器发送数据 (UART 将不发送数据 只到CTS 被证实.
[12]	AUTO_RTS_EN	<b>RTS 自动流控制使能</b> 1 = 使能 RTS 自动流控制. 0 = 禁用 RTS自动流控制. 当 RTS 自动流使能, Rx FIFO中接收的字节数和 UA_FCR[RTS_Tri_Lev]相等, UART 将使RTS信号失效
[11]	TIME_OUT_EN	<b>计时溢出 计数器使能</b> 1 = 使能计时溢出计数器. 0 = 禁用计时溢出计数器.
[10:7]	保留	保留
[6]	WAKE_EN	<b>唤醒 CPU 功能使能</b>

		<p>0 = 禁用 UART 唤醒 CPU 功能</p> <p>1 = 使能唤醒功能, 当系统在深度睡眠模式下, 外部 /CTS 的改变将 CPU 从深度睡眠模式下唤醒.</p>
[5]	保留	保留
[4]	RTO_IEN	<p><b>Rx 计时溢出中断使能</b></p> <p>0 = 禁用INT_tout中断</p> <p>1 = 使能INT_tout 中断</p>
[3]	MODEM_IEN	<p><b>调制解调器中断状态使能</b></p> <p>0 = 禁用off INT_MOS中断</p> <p>1 = 使能INT_MOS中断</p>
[2]	RLS_IEN	<p><b>接收线上中断状态使能</b></p> <p>0 = 禁用off INT_RLS中断</p> <p>1 = 使能INT_RLS中断</p>
[1]	THRE_IEN	<p><b>发送保持寄存器空中断使能</b></p> <p>0 = 禁用 INT_THRE中断</p> <p>1 = 使能INT_THRE中断</p>
[0]	RDA_IEN	<p><b>可接收数据中断使能 .</b></p> <p>0 = 禁用INT_RDA中断</p> <p>1 = 使能INT_RDA中断</p>



## FIFO 控制寄存器 (UA\_FCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_FCR	UART0_BA+0x08	R/W	UART0 FIFO控制寄存器	0x0000_0000
	UART1_BA+0x08	R/W	UART1 FIFO控制寄存器.	0x0000_0000

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

Bits	描述											
[31:20]	保留	保留										
[19:16]	RTS_TRI_LEV	<p>RTS触发自动流程控制使用</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>RTS_TRI_LEV</th> <th>Trigger Level (Bytes)</th> </tr> </thead> <tbody> <tr> <td>0000</td> <td>01</td> </tr> <tr> <td>0001</td> <td>04</td> </tr> <tr> <td>0010</td> <td>08</td> </tr> <tr> <td>0011</td> <td>14</td> </tr> </tbody> </table> <p>注: 该寄存器用于自动RTS流控制..</p>	RTS_TRI_LEV	Trigger Level (Bytes)	0000	01	0001	04	0010	08	0011	14
RTS_TRI_LEV	Trigger Level (Bytes)											
0000	01											
0001	04											
0010	08											
0011	14											
[15:9]	保留	保留										
[8]	RX_DIS	<p>接收器禁用寄存器.</p> <p>接收器禁用或使能(置1禁用接收器)</p> <p>1: 禁用接收器</p> <p>0: 使能接收器</p> <p>注: 该位用于RS-485 普通模式. 必须在设置UA_ALT_CSR [RS-485_NMM]之前被</p>										



		设置好.										
[7:4]	<b>RFITL</b>	<p><b>RX FIFO 中断 (INT_RDA)触发级别</b></p> <p>FIFO 接收字节数与RFITL匹配时, RDA_IF 将被置位 (如果UA_IER [RDA_IEN]使能, 将产生中断).</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 50%;">RFITL</th> <th style="width: 50%;">INTR_RDA Trigger Level (Bytes)</th> </tr> </thead> <tbody> <tr> <td>0000</td> <td>01</td> </tr> <tr> <td>0001</td> <td>04</td> </tr> <tr> <td>0010</td> <td>08</td> </tr> <tr> <td>其它</td> <td>14</td> </tr> </tbody> </table>	RFITL	INTR_RDA Trigger Level (Bytes)	0000	01	0001	04	0010	08	其它	14
RFITL	INTR_RDA Trigger Level (Bytes)											
0000	01											
0001	04											
0010	08											
其它	14											
[3]	保留	保留										
[2]	<b>TFR</b>	<p><b>TX软件复位</b></p> <p>当 Tx_RST 置位, 发送FIFO中的所有字节和Tx内部状态将被清零</p> <p>0 = 该位写 0 将无效.</p> <p>1 =该位置位将复位 Tx 内部机器和指令状态.</p> <p><b>注:</b> 该位自动清零需要至少3个UART时钟周期</p>										
[1]	<b>RFR</b>	<p><b>Rx 软件复位</b></p> <p>当 Rx_RST 置位, 接收FIFO中所有字节和Rx内部状态机都将被清零</p> <p>0 = 该位写 0 将无效.</p> <p>1 = 该位置位将复位 Rx 内部状态机和指令状态.</p> <p><b>注:</b> 该位自动清零需要至少3个UART时钟周期</p>										
[0]	保留	保留										



## Line Control Register (UA\_LCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_LCR	UART0_BA+0x0C	R/W	UART0线控制寄存器	0x0000_0000
	UART1_BA+0x0C	R/W	UART1线控制寄存器	0x0000_0000

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

Bits	描述	
[31:7]	保留	保留
[6]	BCB	<b>钳制控制位</b> 该位置位, 串行数据输出 (Tx) 将被迫间隔发送数据 (逻辑0). 该位仅作用于Tx 对传输逻辑不起作用.
[5]	SPE	<b>Stick 校验使能</b> 0 = 禁用 stick 奇偶使能 1 = 当 PBE, EPE 和 SPE 置位, 校验位传输, 检测被清除. 当 PBE 和 SPE 置位并且 EPE 清除, 校验位传输, 检测有效
[4]	EPE	<b>偶校验使能</b> 0: 数据位和校验位中共有奇数个逻辑1被传输和检测 . 1: 数据位和校验位中共有偶数个逻辑1被传输和检测 该位仅当第三位 (校验位使能) 位置位有效..
[3]	PBE	<b>校验使能位</b> 0 = 当传输时校验位没有产生 (只发送了数据) 产生或检测 (只接收了数据).

		1 =串行数据的最后一位和停止位之间的，就是生成的校验位，校验时检测此位										
[2]	<b>NSB</b>	<p>“STOP bit” 数目</p> <p>0= 传递数据时 1个停止位产生</p> <p>1=传递数据时1. 5个停止位产生(5位数据传输长度被选择);</p> <p>2个停止位产生 6, 7-和 8位数据传输长度被选择.</p>										
[1:0]	<b>WLS</b>	<p>字长度选择</p> <table border="1" data-bbox="558 625 1052 882"> <thead> <tr> <th>WLS[1:0]</th> <th>字长度</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>5 bits</td> </tr> <tr> <td>01</td> <td>6 bits</td> </tr> <tr> <td>10</td> <td>7 bits</td> </tr> <tr> <td>11</td> <td>8 bits</td> </tr> </tbody> </table>	WLS[1:0]	字长度	00	5 bits	01	6 bits	10	7 bits	11	8 bits
WLS[1:0]	字长度											
00	5 bits											
01	6 bits											
10	7 bits											
11	8 bits											

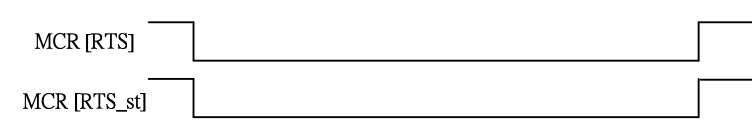
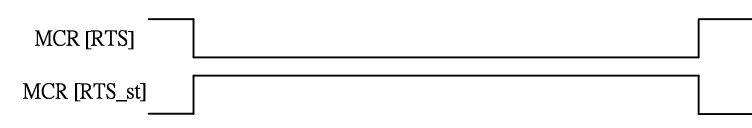
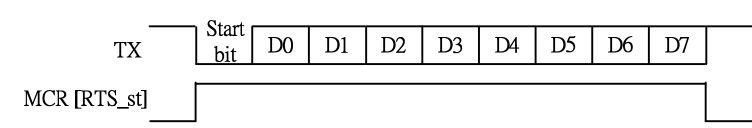
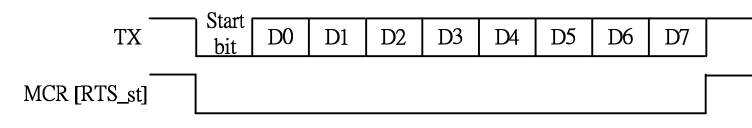


## MODEM 控制寄存器 (UA\_MCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_MCR	UART0_BA+0x10	R/W	UART0调制解调器控制寄存器	0x0000_0000
	UART1_BA+0x10	R/W	UART1 调制解调器控制寄存器	0x0000_0000

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

Bits	描述																
[31:14]	保留	保留															
[13]	RTS_ST	<b>RTS Pin 状态(只读)</b> 该位表示 RTS 引脚状态.															
[12:10]	保留	保留															
[9]	LEV_RTS	<b>RTS 触发电平</b> 该位改变 RTS 触发电平. 0= 低电平触发 1= 高电平触发 <b>UART模式:</b> <table border="1" style="margin-top: 10px; width: 100%;"> <thead> <tr> <th>Input1</th> <th>Input0</th> <th>Output</th> </tr> </thead> <tbody> <tr> <td>LEV_RTS (MCR.BIT9)</td> <td>RTS (MCR.BIT1)</td> <td>RTS_ST (MCR.BIT13, RTS Pin)</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	Input1	Input0	Output	LEV_RTS (MCR.BIT9)	RTS (MCR.BIT1)	RTS_ST (MCR.BIT13, RTS Pin)	0	0	1	0	1	0	1	0	0
Input1	Input0	Output															
LEV_RTS (MCR.BIT9)	RTS (MCR.BIT1)	RTS_ST (MCR.BIT13, RTS Pin)															
0	0	1															
0	1	0															
1	0	0															

		<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="width: 33%; text-align: center;">1</td> <td style="width: 33%; text-align: center;">1</td> <td style="width: 33%; text-align: center;">1</td> </tr> </table> <p><i>UART Mode : MCR[Lev_RTS] = 1</i></p>  <p><i>UART Mode : MCR[Lev_RTS] = 0</i></p>  <p><b>RS-485 Mode:</b></p> <table border="1" style="margin: auto; border-collapse: collapse;"> <thead> <tr> <th style="width: 33%;">Input1</th> <th style="width: 33%;">Input0</th> <th style="width: 33%;">Output</th> </tr> </thead> <tbody> <tr> <td>LEV_RTS (MCR.BIT9)</td> <td>Tx</td> <td>RTS_ST (MCR.BIT13, RTS Pin)</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">x</td> <td style="text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">x</td> <td style="text-align: center;">1</td> </tr> </tbody> </table> <p><i>RS-485 Mode : MCR[Lev_RTS] = 1</i></p>  <p><i>RS-485 Mode : MCR[Lev_RTS] = 0</i></p> 	1	1	1	Input1	Input0	Output	LEV_RTS (MCR.BIT9)	Tx	RTS_ST (MCR.BIT13, RTS Pin)	0	x	0	1	x	1
1	1	1															
Input1	Input0	Output															
LEV_RTS (MCR.BIT9)	Tx	RTS_ST (MCR.BIT13, RTS Pin)															
0	x	0															
1	x	1															
[8:2]	保留	保留															
[1]	<b>RTS</b>	<p><b>RTS (Request-To-Send) 信号</b></p> <p>0: 使 RTS 管脚为 1 (如果 Lev_RTS 设定低电平触发).</p> <p>1: 使 RTS 管脚为 0 (如果 Lev_RTS 设定低电平触发).</p> <p>0: 使 RTS 管脚为 0 (如果 Lev_RTS 设定高电平触发).</p> <p>1: 使 RTS 管脚为 1 (如果 Lev_RTS 设定高电平触发).</p>															
[0]	保留	保留															

## Modem状态寄存器 (UA MSR)

寄存器	偏移量	R/W	描述	复位后的值
UA_MSR	UART0_BA+0x14	R/W	UART0 调制解调器状态寄存器	0x0000_0000
	UART1_BA+0x14	R/W	UART1 调制解调器状态寄存器	0x0000_0000

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

Bits	描述
[31:9]	保留
[8]	<p><b>LEV_CTS</b></p> <p><b>CTS 触发电平</b>                      该位可改变 CTS 触发电平控制TX_FIFO发送数据。                      0=低电平触发                      1=高电平触发</p>
[7:5]	保留
[4]	<p><b>CTS_ST</b></p> <p><b>CTS 引脚状况(只读)</b>                      该位表示 CTS 引脚状态。</p>
[3:1]	保留
[0]	<p><b>DCTS_F</b></p> <p><b>检测侦测 CTS 状态改变标志位(只读)</b>                      只要 CTS 输入状态改变 该位置位, 并且在UA_IER[MODEM_IEN]置位时还会向CPU产生调制解调器中断  <b>注:</b> 该位只读, 可写 '1' 清除。</p>

## FIFO 状态寄存器(UA\_FSR)

寄存器	偏移量	R/W	描述	复位后的值
UA_FSR	UART0_BA+0x18	R/W	UART0 FIFO状态寄存器.	0x1040_4000
	UART1_BA+0x18	R/W	UART1 FIFO状态寄存器	0x1040_4000

31	30	29	28	27	26	25	24
保留			TE_FLAG	保留			
23	22	21	20	19	18	17	16
TX_OVER	TX_EMPTY	TX_POINTER					
15	14	13	12	11	10	9	8
RX_OVERL	RX_EMPTY	RX_POINTER					
7	6	5	4	3	2	1	0
保留	BIF	FEF	PEF	RS-485_ADD_DE TF	保留		

Bits	描述	
[31:29]	保留	保留
[28]	TE_FLAG	<p><b>发送器空闲标志位 (只读)</b></p> <p>当 Tx FIFO(UA_THR) 为空或最后一个字节的停止位被传送到之后, 该位由硬件自动置位.</p> <p>当 Tx FIFO(UA_THR) 不为空或最后一个字节未传输完时, 该位由硬件保持为0</p> <p>注: 该位只读..</p>
[27:24]	保留	保留
[23]	TX_OVER	<p><b>发送 FIFO 溢出(只读)</b></p> <p>该位表示TX FIFO是否溢出.</p> <p>如果发送数据的字节数大于TX_FIFO (UA_RBR) 的大小, UART0/UART1为15个字节, 该位将置位, 否则由硬件清零.</p>
[22]	TX_EMPTY	<p><b>发送FIFO 为空(只读)</b></p> <p>该位表示 Tx FIFO 是否为空.</p> <p>当 Tx FIFO的最后一个字节传输到发送移位寄存器时, 硬件置位该位. 当写数据到THR (Tx FIFO非空) 清除.</p>
[21:16]	TX_POINTER	<b>TX FIFO指针(只读)</b>

		该位表示Tx FIFO 缓冲指示器. 当CPU 写 1字节到 UA_THR, Tx_Pointer 增 1.当 Tx FIFO 传输 1 字节到发送移位寄存器, Tx_Pointer 减1.
[15]	RX_OVER	<b>接收器 FIFO 溢出 (只读)</b> 该位表示RX FIFO 是否溢出. 如果接收数据的字节数大于RX_FIFO (UA_RBR) 的大小, UART0/UART1为15个字节, 该位置位, 否则由硬件清零.
[14]	RX_EMPTY	<b>接收FIFO 为空(只读)</b> 该位表示 Rx FIFO是否为空. 当 Rx FIFO 最后字节 从CPU中读取, 硬件置位 该位. 当 UART 接收到新数据 该位清除.
[13:8]	RX_POINTER	<b>Rx FIFO 指针(只读)</b> 该位表示 Rx FIFO 缓冲指示器. 当UART 从外部设备接收到 1 字节数据, Rx_Pointer增 1. 当 Rx FIFO 通过 CPU 读 1字节数据, Rx_Pointer 减1.
[7]	保留	保留
[6]	BIF	<b>钳制中断标志位 (只读)</b> 当接收数据的输入时, 保持在空状态(逻辑0)状态的时间大于输入全字传输(即即起始位 + 数据位 + 校验位 + 停止位的所有时间)的时间, 该位置1. 无论CPU何时向该位写1都会使该位重置. 注: 该位只读, 但可以写1清零.
[5]	FEF	<b>帧错误 标志位 (只读)</b> 当接收的字符串没有正确的停止位(即检测到跟在最后一个数据位或校验位后面的停止位为逻辑0)时, 该位置位. 该位在CPU向其写1时清零 注: 该位只读, 但可以写1清零.
[4]	PEF	<b>奇偶校验错误 标志位(只读)</b> 当接收到的字符串的校验位无效时, 该位将置位, CPU 写 1 到 该位 复位 注: 该位只读, 但可以写1清零.
[3]	RS-485_ADD_DETF	<b>RS-485地址字节检测标志 (只读)</b> RS-485模式, 只要接收器检测到地址字节接收到了地址字节字符 (第9位为1), 该位与UA_ALT_CSR均将置位. 只要CPU写1到该位就复位. 注: 该位用于RS-485 模式. 注: 该位只读, 但可写1清零.
[2:0]	保留	保留



## 中断状态控制寄存器(UA\_ISR)

寄存器	偏移量	R/W	描述	复位后的值
UA_ISR	UART0_BA+0x1C	R/W	UART0中断状态控制寄存器	0x0000_0002
	UART1_BA+0x1C	R/W	UART1中断状态控制寄存器	0x0000_0002

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留			TOUT_INT	MODEM_INT	RLS_INT	THRE_INT	RDA_INT
7	6	5	4	3	2	1	0
保留			TOUT_IF	MODEM_IF	RLS_IF	THRE_IF	RDA_IF

Bits	描述
[31:13]	保留
[12]	<b>TOUT_INT</b> 计时溢出状态指示中断控制器(只读) 将RTO_IEN 和Tout_IF进行“与”(AND), 然后在该位输出
[11]	<b>MODEM_INT</b> 调制解调器 状态指示中断控制器(只读). 将Modem_IEN 和Modem_IF进行“与”(AND), 然后在该位输出
[10]	<b>RLS_INT</b> 接收 Line 中断状态指示中断控制器(只读). 将RLS_IEN 和RLS_IF进行“与”(AND), 然后在该位输出
[9]	<b>THRE_INT</b> 发送保持寄存器为空中断指示中断控制器 (只读). THRE_IEN 和 THRE_IF进行 “与” (AND), 然后在该位输出
[8]	<b>RDA_INT</b> 接收数据中断指示中断控制器 (只读). RDA_IEN 和 RDA_IF 输入进行 “与” (AND), 然后在该位输出
[7:5]	保留
[4]	<b>TOUT_IF</b> 计时溢出中断标志 (只读) 当 Rx FIFO非空且无动作, 同时时间溢出计数器和TOIC 相等 该位置位. 若 UA_IER [TOUT_IEN] 使能, 计时溢出中断产生.

		注: 该位只读, 用户可读UA_RBR (Rx is in active)清空.
[3]	MODEM_IF	<p><b>调制解调器中断标志 (只读)</b></p> <p>当 CTS引脚状态(DCTS=1)改变 该位置位.若UA_IER [MODEM_IEN]使能, 调制解调器中断产生.</p> <p>注: 写1清该位到0.</p>
[2]	RLS_IF	<p><b>接收线 状态标志位 (只读).</b></p> <p>当 Rx 接收数据有奇偶校验错误、帧错误、打断错误时,该位置位, framing error 或 break error (至少3 位, BIF, FEF 和 PEF, 置位). 若UA_IER [RLS_IEN] 使能, RLS 中断产生.</p> <p>注: 在RS-485模式, 该位包括“接收器检测任何地址字节接收到的地址字节符号 (第9位为1)</p> <p>注: 写1清该位0.</p>
[1]	THRE_IF	<p><b>发送保持寄存器空中断标志 (只读).</b></p> <p>当TX FIFO 的最后一个数据发送到发送器移位寄存器, 该位置位. 如果UA_IER [THRE_IEN] 使能, THRE中断产生.</p> <p>注: 该位只读, 写数据到THR 清零该位 (TX FIFO not empty).</p>
[0]	RDA_IF	<p><b>接收数据中断标志(只读).</b></p> <p>当RX FIFO 中的字节数等于RFITL , RDA_IF 置位. 如果使能 UA_IER [RDA_IEN], RDA 中断产生.</p> <p>注: 该位只读, 当RX FIFO 的未读取字节数少于阈值 (RFITL)时该位清零</p>

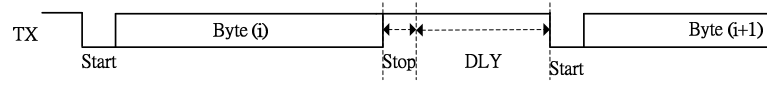
UART 中断源	中断使能位	中断指示 中断控制	中断标志位	标志位清除
RX计时溢出t中断 INT_TOUT	RTO_IEN	TOUT_INT	TOUT_IF	Read UA_RBR
调制解调器 状态中断 INT_MODEM	MODEM_IEN	MODEM_INT	MODEM_IF = (DCTSIF)	Write '1' to DCTSIF
接收线状态中断 INT_RLS	RLS_IEN	RLS_INT	RLS_IF = (BIF or FEF or PEF)	Write '1' to BIF/FEF/PEF
发送保持寄存器为空中断 INT_THRE	THRE_IEN	THRE_INT	THRE_IF	Write UA_THR
接收数据中断 INT_RDA	RDA_IEN	RDA_INT	RDA_IF	Read UA_RBR

表 6.10-3 软件模式下UART 中断源和标志表 ( Software Mode)

## Time out 寄存器 (UA TOR)

寄存器	偏移量	R/W	描述	复位后的值
UA_TOR	UART0_BA + 0x20	R/W	UART0 定时溢出寄存器	0x0000_0000
	UART1_BA + 0x20	R/W	UART1 定时溢出寄存器	0x0000_0000

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

Bits	描述	
[31:16]	保留	保留
[15:8]	DLY	<p><b>TX 延迟时间值</b></p> <p>该位用于编程上一停止位与下一开始位之间的延迟时间。</p> 
[6:0]	TOIC	<p><b>定时溢出中断比较器</b></p> <p>当 RX FIFO 接收到新数据后定时计数器复位和开始计数(定时器时钟频率=波特率). 一旦定时溢出计数器 (TOUT_CNT)和定时溢出中断比较器 (TOIC) 相等, 且 UA_IER [RTO_IEN]使能, 接收器定时溢出中断产生 (INTR_TOUT) . 一个新的输入数据字或RX FIFO 为空将清 INT_TOUT.</p>



## 波特率分频寄存器(UA\_BAUD)

寄存器	偏移量	R/W	描述	复位后的值
UA_BAUD	UART0_BA+0x24	R/W	UART0波特率分频寄存器	0x0F00_0000
	UART1_BA+0x24	R/W	UART1波特率分频寄存器	0x0F00_0000

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

Bits	描述	
[31:30]	保留	保留
[29]	DIV_X_EN	<p>分频 X 使能器</p> <p>BRD = 波特率分频值, 波特率方程如下:  <b>波特率 = Clock / [ M * (BRD + 2) ]</b>; 默认 M 为 16.</p> <p>0 = 禁用分频器 X (the equation of M = 16)</p> <p>1 = 使能分频器 X (the equation of M = X+1, but DIVIDER_X [27:24] must &gt;= 8).</p> <p>详见表6.10-4</p> <p>注: 在IrDA 模式下 该位禁用.</p>
[28]	DIV_X_ONE	<p>分频系数X等于1</p> <p>0 = 分频系数M = X (M = X+1,但DIVIDER_X[27:24]必须大于或等于8)</p> <p>1 = 分频系数M = 1 (M = 1, 但BRD [15:0] 必须大于或等于3).</p> <p>参考下表.</p>
[27:24]	DIVIDER_X	<p>分频 X</p> <p>波特率分频: M = X+1.</p>
[23:16]	保留	保留

[15:0]	<b>BRD</b>	波特率分频器 这些位表示波特率分频器
--------	------------	-----------------------

模式	DIV_X_EN	DIV_X_ONE	DIVIDER X	BRD	波特率公式
0	Disable	0	B	A	$UART\_CLK / [16 * (A+2)]$
1	Enable	0	B	A	$UART\_CLK / [(B+1) * (A+2)]$ , B must $\geq 8$
2	Enable	1	Don't care	A	$UART\_CLK / (A+2)$ , A must $\geq 3$

表 6.10-4 波特率方程表

## IrDA 控制器寄存器 (IRCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_IRCR	UART0_BA+0x28	R/W	UART0 IrDA控制寄存器.	0x0000_0040
	UART1_BA+0x28	R/W	UART1 IrDA控制寄存器	0x0000_0040

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

Bits	描述	
[31:7]	保留	保留
[6]	INV_RX	<b>INV_RX</b> 1= Rx 输入信号反转 0= 无反转
[5]	INV_TX	<b>INV_TX</b> 1= Tx 输出信号反转 0=无反转
[4:2]	保留	保留
[1]	TX_SELECT	<b>TX_SELECT</b> 1: 使能IrDA 发送器 0: 使能 IrDA 接收器
[0]	保留	保留

注：在 IrDA 模式，寄存器 UA\_BAUD[DIV\_X\_EN]必须禁用 (波特方程为 Clock / 16 \* (BRD))

## UART 控制/状态寄存器 (UA\_ALT\_CSR)

寄存器	偏移量	R/W	描述	复位后的值
UA_ALT_CSR	UART0_BA+0x2C	R/W	UART0 控制/状态寄存器	0x0000_0000
	UART1_BA+0x2C	R/W	UART1控制/状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
ADDR_MATCH							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
RS-485_ADD_EN	保留				RS-485_AUD	RS-485_AAD	RS-485_NMM
7	6	5	4	3	2	1	0
保留							

Bits	描述	
[31:24]	ADDR_MATCH	地址匹配值寄存器 该位包含RS-485 地址匹配值。 注: 该位用于RS-485自动地址识别模式。
[23:16]	保留	保留
[15]	RS-485_ADD_EN	RS-485 地址识别使能 该位用于使能RS-485地址识别模式。 1: 使能地址识别模式 0: 禁用地址识别模式 注: 该位用于RS-485的所有模式。
[14:11]	保留	保留
[10]	RS-485_AUD	RS-485 自动方向模式 (AUD) 1: 使能RS-485 自动方向操作模式 (AUO) 0: 禁用 RS-485 自动方向操作模式(AUO) 注: RS-485_AAD 或 RS-485_NMM 操作模式下有效。
[9]	RS-485_AAD	RS-485 自动地址识别操作模式 (AAD) 1: 使能RS-485 自动地址识别操作(AAD)



		0: 禁用 RS-485 自动地址识别操作模式(AAD) 注: RS-485_NMM 操作模式下无效.
[8]	RS-485_NMM	<b>RS-485 普通操作模式 (NMM)</b> 1: 使能 RS-485 普通操作模式 (NMM) 0: 禁用 RS-485 普通操作模式 (NMM) 注: RS-485_AAD 操作模式下无效.
[7:0]	保留	保留

## UART 功能选择寄存器 (UA\_FUN\_SEL)

寄存器	偏移量	R/W	描述	复位后的值
UA_FUN_SEL	UART0_BA+0x30	R/W	UART0功能选择寄存器	0x0000_0000
	UART1_BA+0x30	R/W	UART1功能选择寄存器	0x0000_0000

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

Bits	描述	
[31:2]	保留	保留
[1:0]	<b>FUN_SEL</b>	功能选择使能 00 = UART 01 = 保留 10 = IrDA 11 = RS-485

## 6.11 模拟数字转换(ADC)

### 6.11.1 简介

NuMicro M051™ 系列包含一个8通道12位的逐次逼近式模拟 – 数字转换器 (SAR A/D转换器). A/D 转换器支持四种工作模式: 单次转换模式、突发转换模式、单周期扫描模式和连续扫描模式. 开始A/D 转换可软件设定和外部STADC/P3.2引脚启动。

### 6.11.2 特征

- 模拟输入电压: 0~0~AVDD(最大to 5.0V).
- 12-bits 分辨率和 10-bits 精确度保证.
- 多达 8 路单端输入通道或4路差分输入通道.
- 最大 ADC 时钟频率 16MHz.
- 高达600k SPS 转换速率.
- 四种运作模式
  - 单次转换模式: A/D转换在指定通道完成一次转换.
  - 单周期扫描模式: A/D 转换在指定通道完成一个周期 (从低序号通道到高序号通道) 转换.
  - 连续扫描模式: A/D 转换器连续执行单周期扫描模式直到软件停止A/D转换.
  - 突发模式: A/D 转换 采样和转换在指定单个通道进行, 并将结果存入FIFO.
- A/D转换开始条件
  - 软件向ADST 位写1
  - 外部引脚STADC触发
- 每通道转换结果存储在相应数据寄存器内, 并带有有效或超出限度的标志.
- 转换结果可和指定的值相比较 当转换值和设定值相匹配时, 用户可设定是否产生中断请求.
- 通道 7 支持 2 输入源: 外部模拟电压, 内部固定参考电压.
- 支持自身校正功能以减少转换的误差.

6.11.3 ADC框图

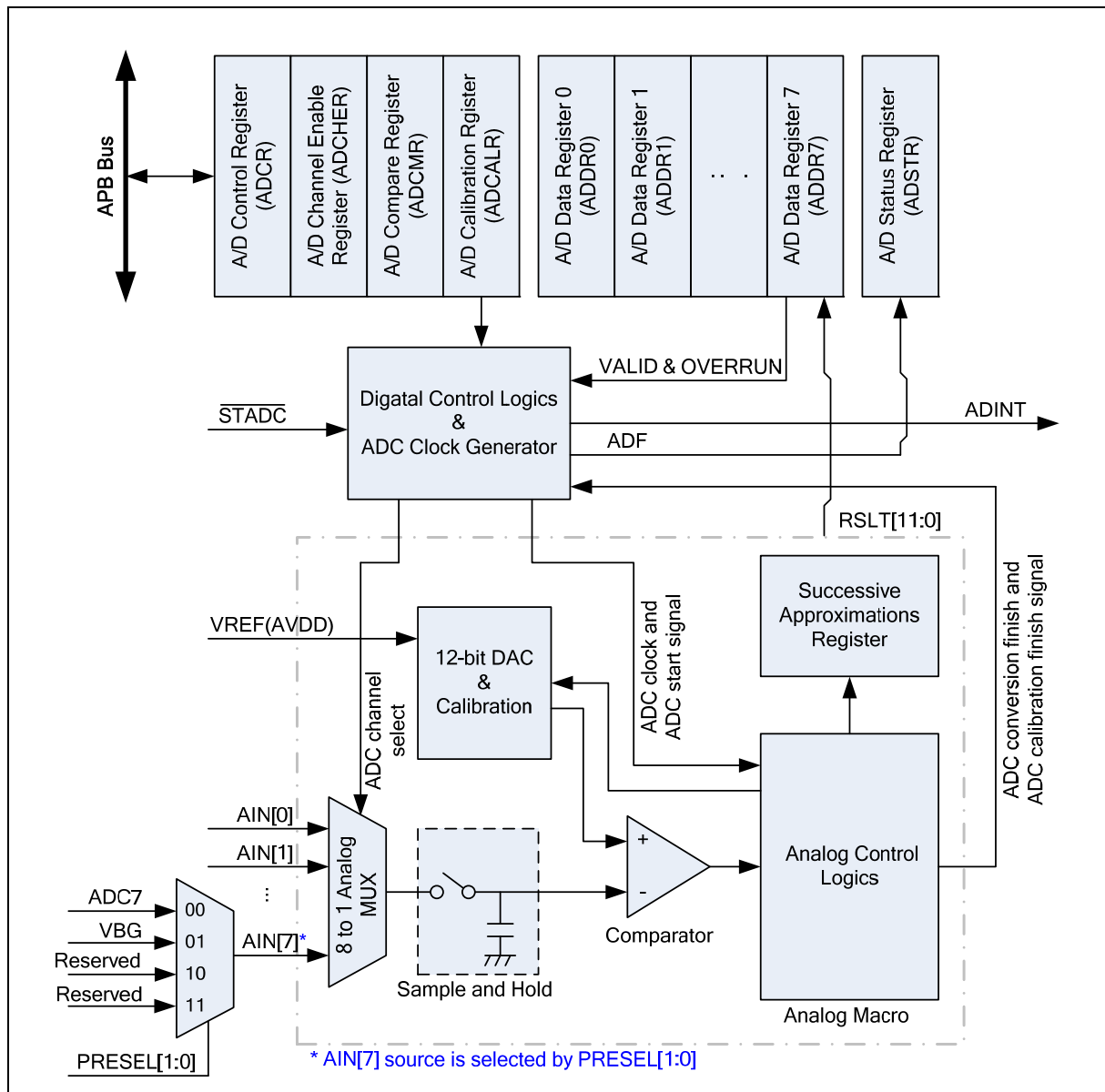


图 6.11-1 ADC 控制器框图

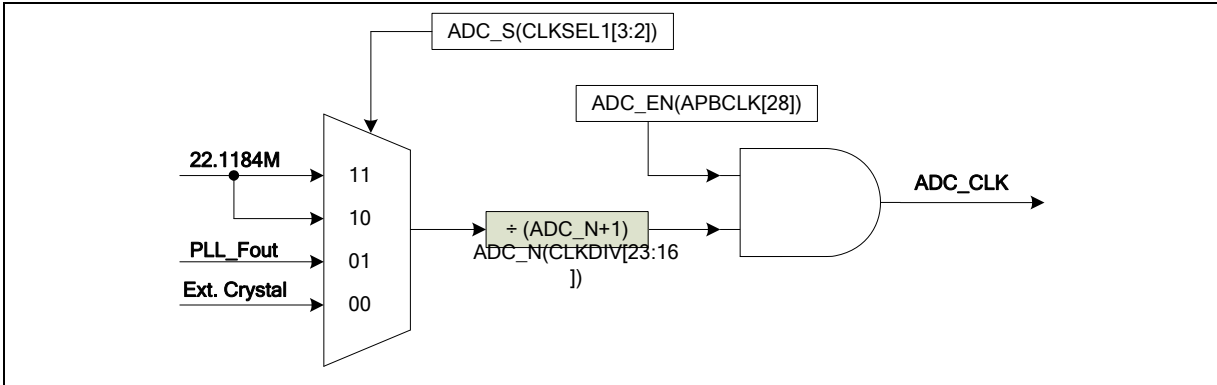


图 6.11-2 ADC 时钟控制

### 6.11.4 ADC操作步骤

A/D转换器为12位逐次逼近式。A/D 具有自身校正功能 减少转换的误差, 用户可写 1 到 CALEN 位 ( ADCALR 寄存器) 使能自身校正功能, 当内部校正完成 CAL\_DONE 置位。ADC 具有4种操作模式: 单次转换模式、突发转换模式、单周期扫描模式和连续扫描模式。当改变工作模式或使能的模拟输入通道时, 为了防止错误的操作, 软件需清 ADST 位为 0 (ADCR register)。

#### 6.11.4.1 自校准

当系统上电或要在单通道模式与差分输入模式间切换时, 就需要ADC自校正以减小转换误差。用户置位 CALEN位(ADCALR 寄存器) 使能自身校正功能。这个过程在内部执行, 需要127 ADC 时钟完成校正。CALEN 置位后, 软件需等待 CAL\_DONE 位通过内部硬件置位。时序图如下:

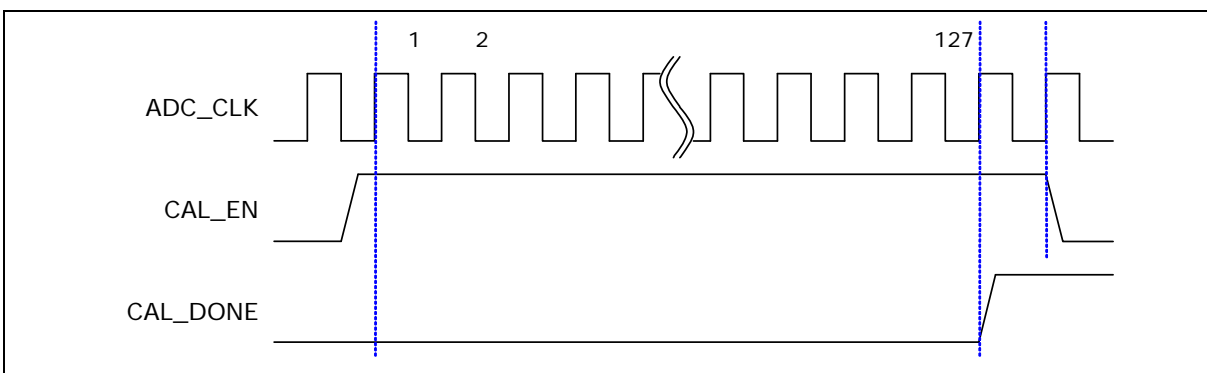


图 6.11-3 ADC 转换器自校准时序框图

#### 6.11.4.2 ADC 时钟发生器

最大采样率达600K。ADC有三个时钟源, 可由ADC\_S (CLKSEL[3:2])选择, ADC时钟频率由一个8位分频器按如下公式进行8位预分频:

The ADC 时钟频率 = (ADC 时钟源频率) / (ADC\_N + 1);

8位ADC\_N 在寄存器CLKDIV[23:16]中设置。

通常来说, 软件设置ADC\_S 与 ADC\_N 获得 16MHZ 或稍低于16MHZ的频率。

因为带隙电压源驱动能力不强, 建议转换速率低于15 kHz。

#### 6.11.4.3 单次转换模式

在单次转换模式下, A/D 转换相应通道上只执行一次, 运作流程如下:

1. 当通过软件或外部触发输入使 ADCR 的ADST 置位开始A/D 转换。
2. 当 A/D 转换完成, A/D 转换的数据值将存储于的A/D数据寄存器中
3. A/D 转换完成, ADSR 的ADF 位置1. 若此时ADIE 位置1, 将产生ADINT 中断请求。

4. A/D转换期间, ADST 位保持为1. A/D 转换结束, ADST 位自动清 0, A/D 转换器进入空闲模式

注: 在单次转换模式时, 如果软件使能多于一个通道, 序号最小的通道被转换, 其他通道被忽略.

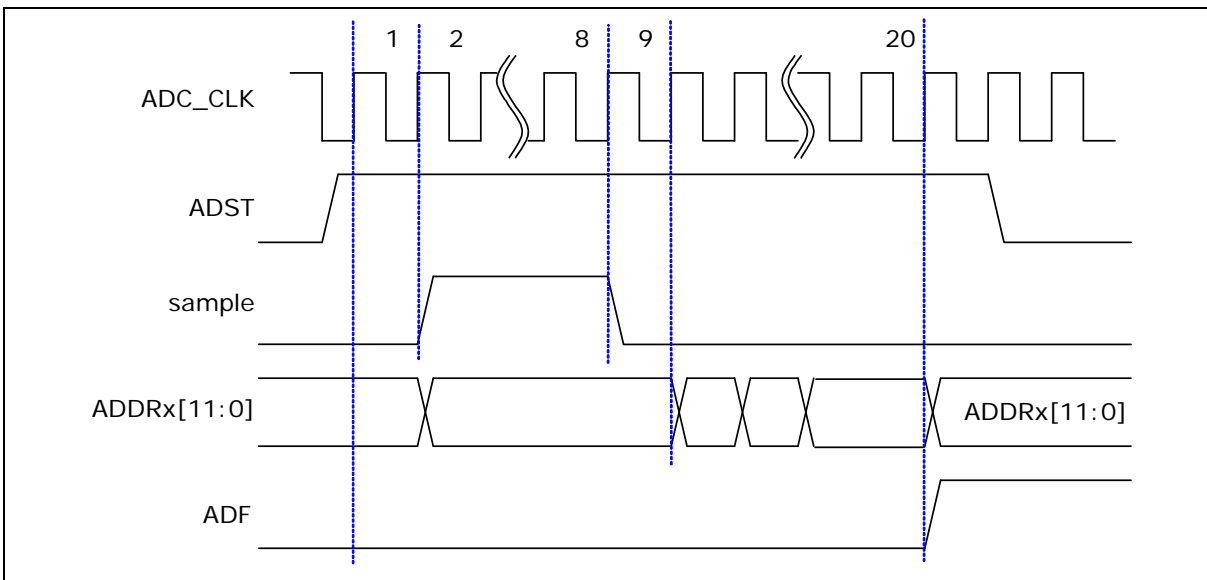


图 6.11-4 单次转换模式时序图

#### 6.11.4.4 突发模式

突发模式, A/D 转换会采样和转换指定的单个通道,并有序存储在FIFO (最多8次采样). 操作步骤如下:

1. 软件或外部触发置ADCR的ADST位为1, 从序号最小的通道开始A/D 转换.
2. 当使能的通道A/D转换完成, 结果有序送入FIFO, 可以从A/D数据寄存器0得到.
3. 当FIFO中多于4个采样, ADSR的ADF位置1. 如果此时ADIE位置1, 在A/D转换完成时就会产生ADINT中断请求.
4. ADST保持为1时, 重复步骤2到步骤3. 当ADST位清零时, A/D转换停止, A/D转换器进入空闲状态.

注:在突发模式下, 如果软件使能多个通道, 则序号最小通道进行转换, 其他通道不转换.

#### 6.11.4.5 单周期扫描模式

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

1. 软件或外部触发使ADCR 寄存器的 ADST 位置位, 开始 从最小序号通道到最大序号通道的 A/D 转换.
2. 每路 A/D 转换完成后, A/D 转换数值将有序装载到相应数据寄存器中.
3. 当所选择的通道转换完成后, ADSR的ADF位置1, 如果此时ADIE位也置1, 在A/D转换结束后就会有ADINT中断请求.
4. A/D 转换结束, ADST 位自动清 0, A/D 转换器进入空闲模式. 如果在所有被使能通道完成转换前

ADST 清 0，A/D 转换将完成当前转换，并且序号最小的通道的结果将不可预知。

使能通道(0, 2, 3 and 7) 单周期模式时序图如下:

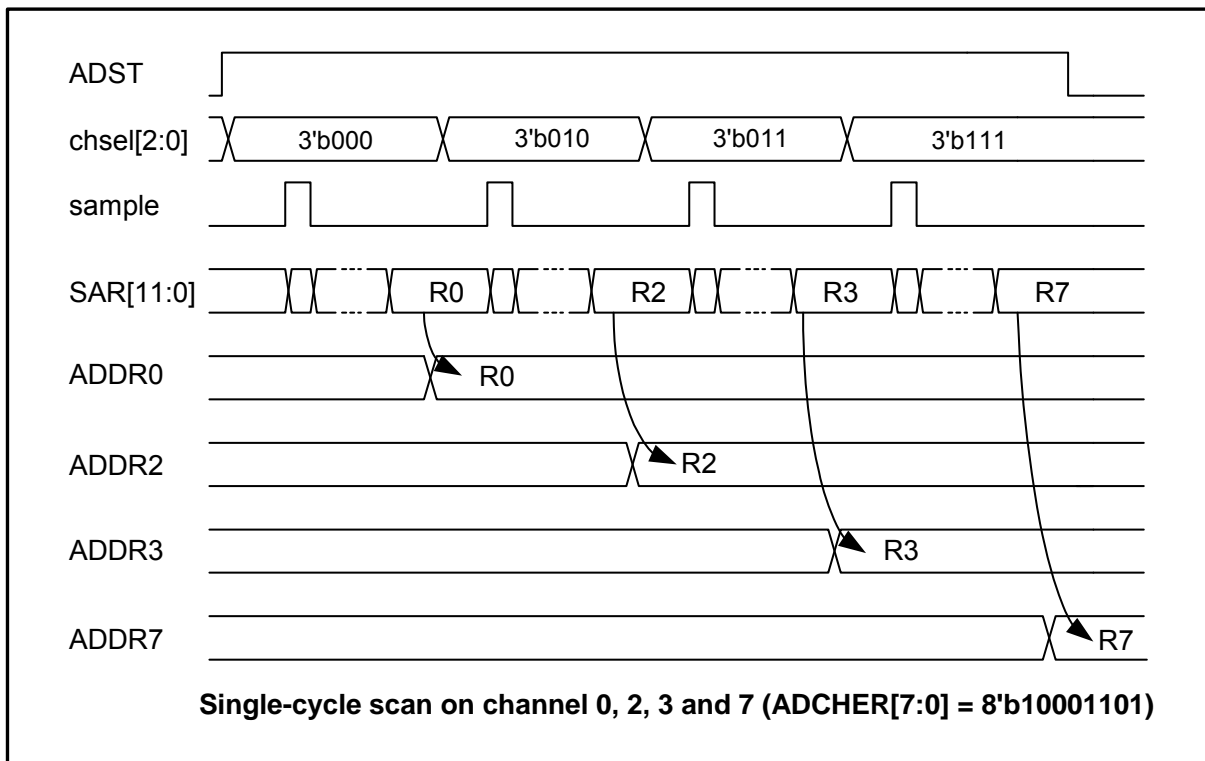


图 6.11-5 单周期扫描下使能通道转换时序图

#### 6.11.4.6 连续扫描模式

连续扫描模式，使能ADCHER的CHEN位，A/D转换在指定通道进行(最多选中8个ADC通道). 操作步骤如下:

1. 通过软件或外部触发使ADCR 寄存器的 ADST 位置位, 开始 最小序号通道到最大序号通道的 A/D 转换.
2. 每路 A/D 转换完成后, A/D 转换数值将装载到相应数据寄存器中.
3. 当被选择的通道数 都完成了一次转换后, ADF 位 (ADSR 寄存器) 置1. 若此时 ADIE 置1, 当 A/D 转换完成后, 将产生 ADINT 中断请求. 转换又从被使能的第一个通道开始.
4. 只要ADST位保持为1, 就重复步骤2到3. 当ADST位被清0, A/D 转换停止, A/D 转换器进入空闲状态. 当ADST清0, ADC 控制器将完成当前转换, 被使能的最小序号ADC通道的结果将不可预料。



使能通道(0, 2, 3和7) 连续扫描模式时序图如下:

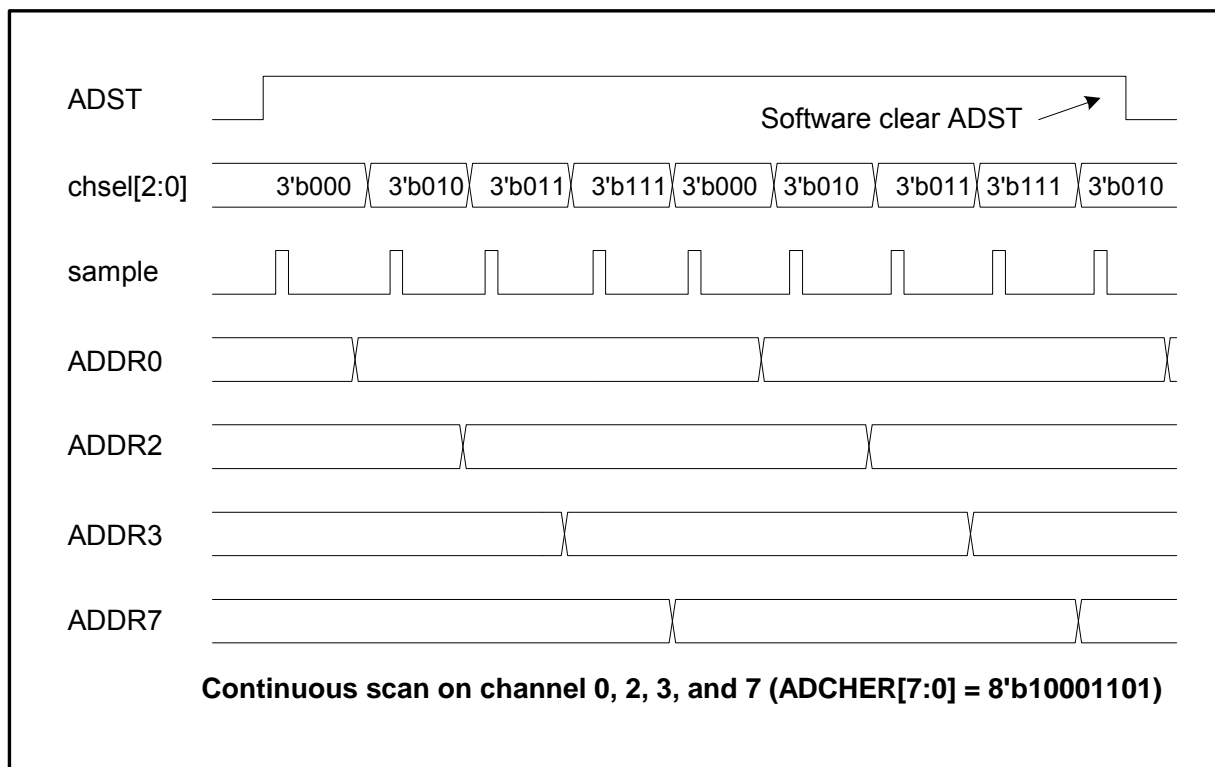


图 6.11-6 使能通道的连续扫描时序图

#### 6.11.4.7 输入采样和 A/D 转换时间

A/D 转换可通过外部引脚触发请求.当 ADCR.TRGEN 置位, 使能 ADC 外部触发功能, 配置 TRGS[1:0] 位为 00b 选择从 STADC 引脚输入外部触发. 软件设定 TRGCOND[1:0] 选择触发方式上升沿/下降沿 或低电平/高电平触发. 8位取样计数器应用于抗尖峰脉冲. 若选择 电平触发条件, STADC 需保持定义的电平状态 至少 8 个 PCLK 周期. 在第9个 PCLK 时钟来临时 ADST 位置位, 开始新的转换电平触发模式状态下, 如果外部触发输入状态为拉低 (或高电平), 进行连续转换. 仅当外部触发条件消失 才停止. 若选择边沿触发模式, 高低电平状态至少需保持 4 PLCKs 周期. 脉冲低于该值时, 将被忽略.

#### 6.11.4.8 比较模式下 AD 转换结果监控

NuMicro M051™ 系列提供2个比较寄存器 ADCMPR0 和 1, 可最多监控 A/D 转换模块的 2路指定通道转换结果值, 可参考图 6.11-7.. 可通过软件设定 CMPCH(ADCMPRx[5:0])选择监控通道, CMPCOND 位检查转换置小于或大于 等于 CMPD[11:0] 指定值. 当被 CMPCH 指定的通道完成转换时, 比较就被自动触发且执行一次. 当比较结果和设定值相匹配, 比较匹配计数器将加 1, 否则比较匹配计数器就清 0. 当计数器的值和设定值 (CMPMATCNT+1) 匹配, CMPF 位将置 1, 如果 CMPIE 置位 将产生 ADINT 中断请求. 在扫描模式下 软件可应用于监控外部模拟输入引脚电压变化而不会增加程序负载. .具体逻辑框图如下:

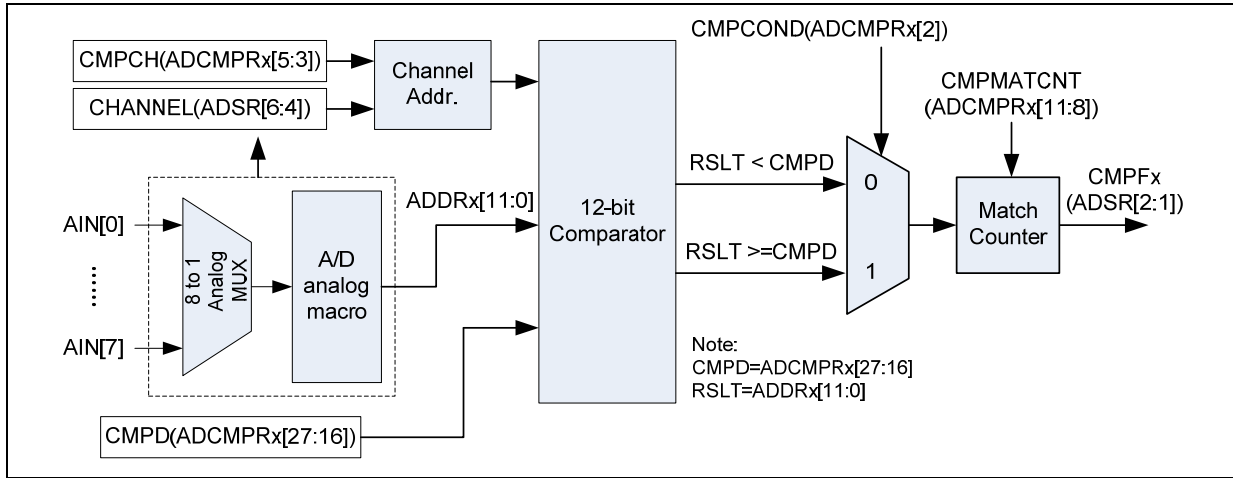


图 6.11-7 A/D 转换结果监控框图

6.11.4.9 中断源

A/D 转换结束时，A/D转换器产生ADF转换完成标志（ADSR寄存器）。若 ADIE 位（ADCR 寄存器）置位，将产生 ADINT 中断请求。若 CMPIE 位使能，当 A/D 转换结果同 ADCMPR 寄存器设定值相匹配，将置位ADINT，产生一个监控中断，系统可清 CMPF 和 ADF 位 禁用中断请求

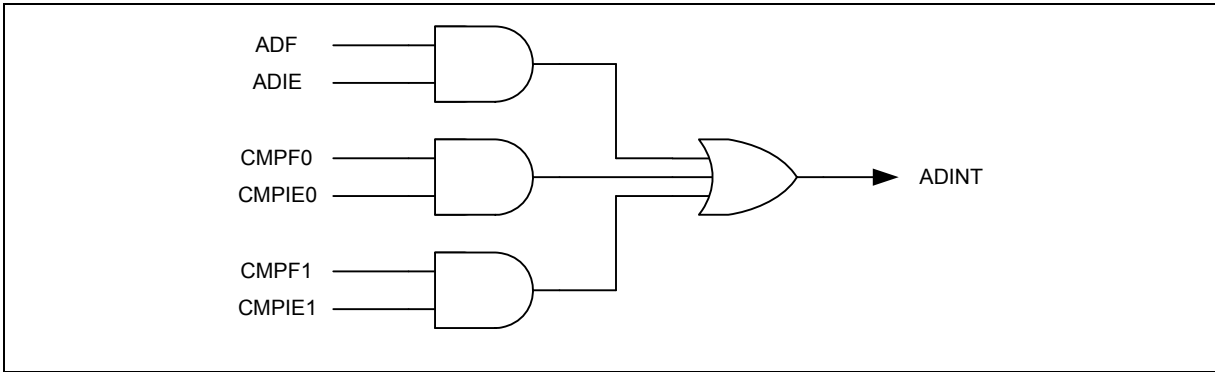


图 6.11-8 A/D 控制器中断

## 6.11.5 ADC 寄存器映射

R: 只读, W: 只写, R/W: 可读写, C: 仅在值为0时可写

寄存器	偏移量	R/W	描述	复位后的值
<b>ADC_BA = 0x400E_0000</b>				
ADDR0	ADC_BA+0x00	R	A/D数据寄存器 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D数据寄存器 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D数据寄存器 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D数据寄存器 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D数据寄存器 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D数据寄存器 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D数据寄存器 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D数据寄存器 7	0x0000_0000
ADCR	ADC_BA+0x20	R/W	A/D控制寄存器	0x0000_0000
ADCHER	ADC_BA+0x24	R/W	A/D 通道使能寄存器	0x0000_0000
ADCMPR0	ADC_BA+0x28	R/W	A/D 比较寄存器0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	A/D比较寄存器1	0x0000_0000
ADSR	ADC_BA+0x30	R/W	A/D 状态寄存器	0x0000_0000
ADCALR	ADC_BA+0x34	R/W	A/D 校准寄存器	0x0000_0000



## 6.11.6 ADC 寄存器描述

### A/D数据寄存器 (ADDR0 ~ ADDR7)

寄存器	偏移量	R/W	描述	复位后的值
ADDR0	ADC_BA+0x00	R	A/D数据寄存器 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D数据寄存器 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D数据寄存器 2	0x0000_0000
ADDR3	ADC_BA+0x0c	R	A/D数据寄存器 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D数据寄存器 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D数据寄存器 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D数据寄存器 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D数据寄存器 7	0x0000_0000

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

Bits	描述	
[31:18]	保留	-
[17]	VALID	有效标志位(只读) 1 = RSLT[11:0] 位数据 有效. 0 = RSLT[11:0] 位数据 无效. 相应模拟通道 转换完成后, 将该位置位, 读ADDR 寄存器后, 该位由硬件清除.
[16]	OVERRUN	结束运行标志位(只读)

		1 = RSLT[11:0] 数据 被覆盖. 0 = RSLT[11:0] 数据 新近转换结果. 新的转换结果装载至寄存器之前, 若 RSLT[11:0] 的数据没有被读取, OVERRUN 将置 1. 读ADDR 寄存器后, 该位由硬件清除..
[15:12]	保留	-
[11:0]	<b>RSLT</b>	<b>A/D 转换结果</b> 包括 12 位AD 转换结果.

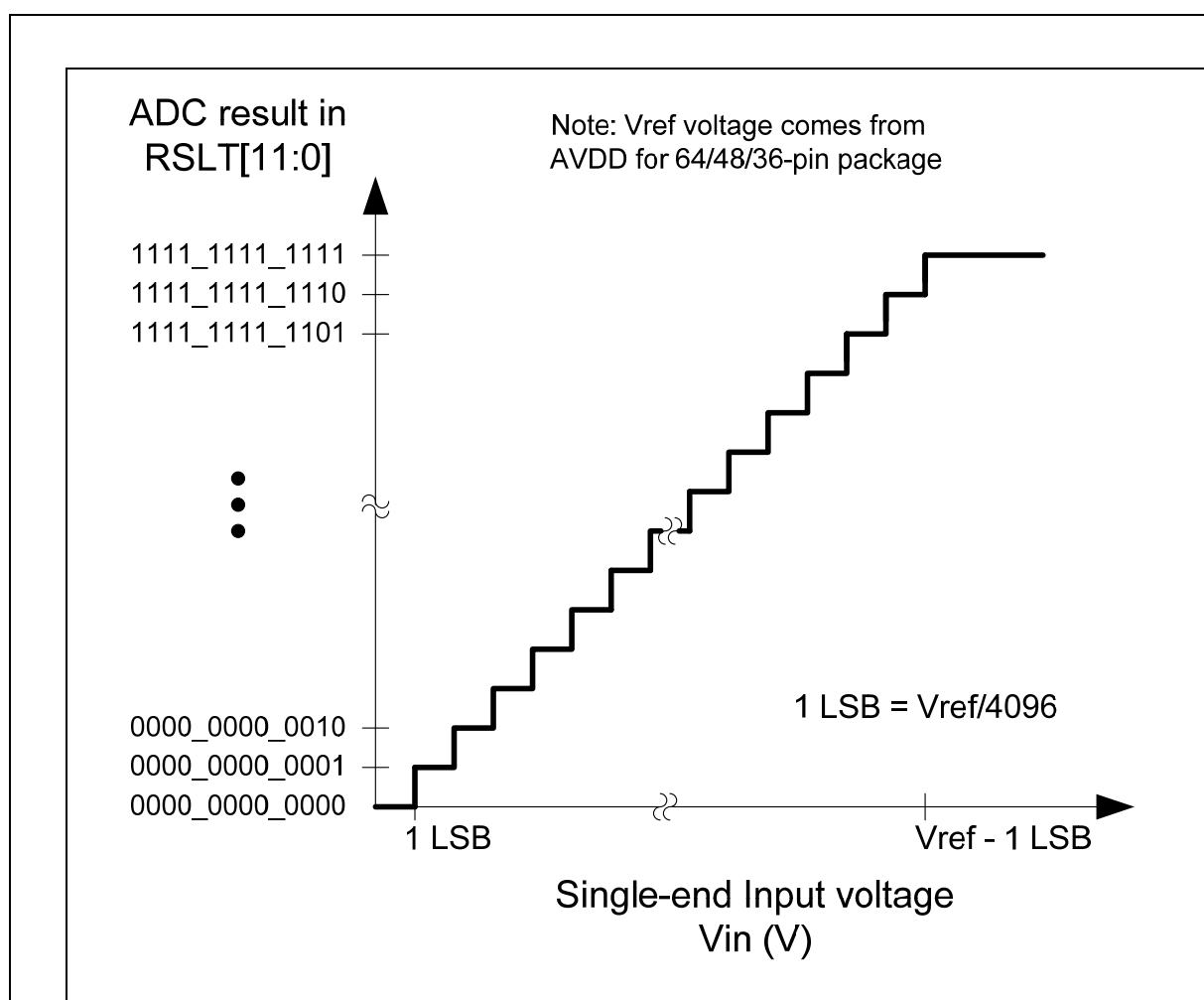


图 6.11-9 ADC 单端输入转换电压和转换结果图

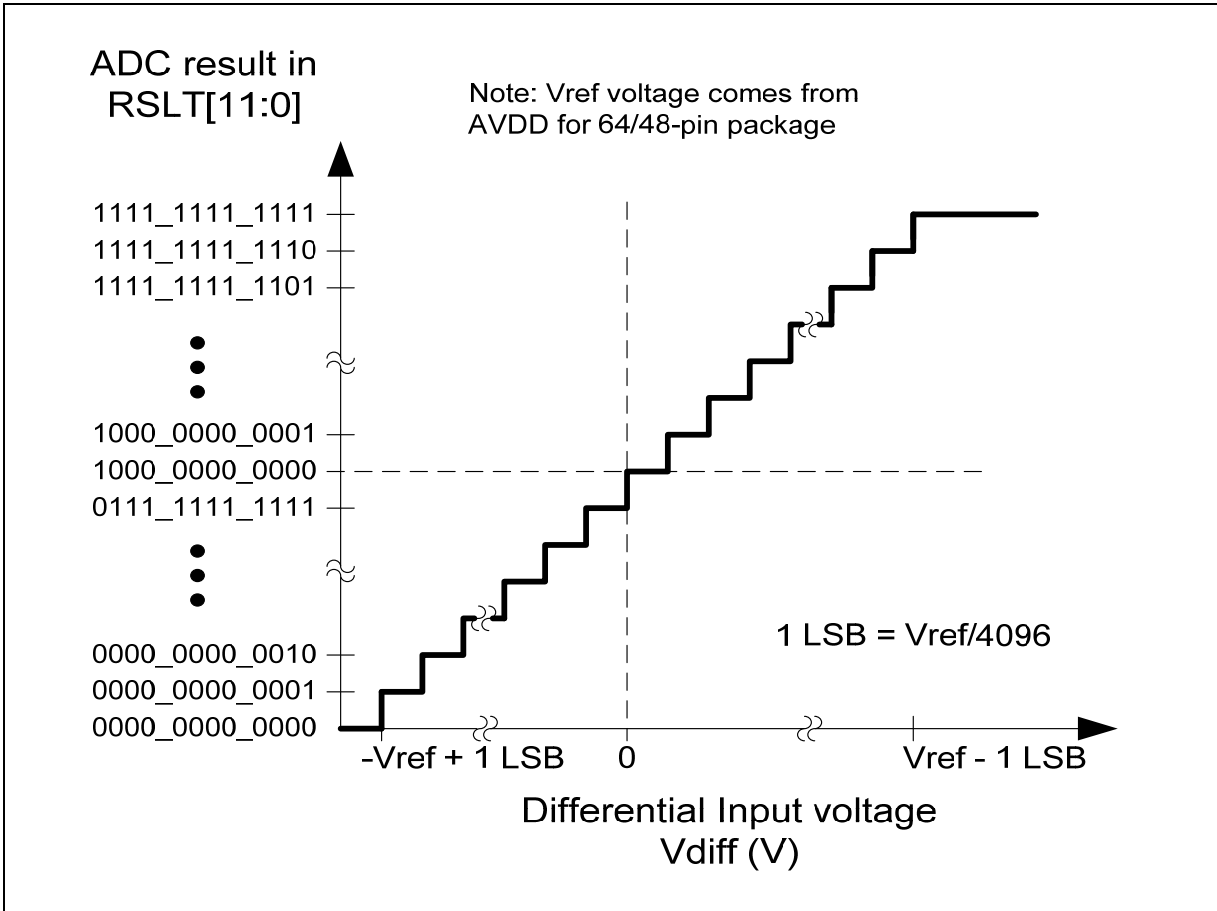


图 6.11-10 ADC差分输入转换电压和转换结果图



## A/D 控制寄存器 (ADCR)

寄存器	偏移量	R/W	描述	复位后的值
ADCR	ADC_BA+0x20	R/W	ADC 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				ADST	DIFFEN	保留	TRGEN
7	6	5	4	3	2	1	0
TRGCOND		TRGS		ADMD		ADIE	ADEN

Bits	描述																	
[31:12]	保留																	
[11]	<p><b>ADST</b></p> <p><b>A/D 转换开始</b> 1 = 转换开始. 0 = 转换结束或A/D转进入空闲状态.</p> <p>ADST 位置位有下列 2 种方式: 软件设定和外部STADC引脚. 单次转换模式和单周期扫描模式下, 转换完成后, ADST 将被硬件自动清除在连续扫描模式下, A/D 转换将一直进行 只到软件写0到该位或系统复位.</p>																	
[10]	<p><b>DIFFEN</b></p> <p><b>A/D 差分输入模式使能</b> 1 = A/D为差分输入模式 0 = A/D为单端输入模式</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th rowspan="2"></th> <th colspan="2">模拟输入</th> </tr> <tr> <th>V<sub>plus</sub></th> <th>V<sub>minus</sub></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>AIN0</td> <td>AIN1</td> </tr> <tr> <td>1</td> <td>AIN2</td> <td>AIN3</td> </tr> <tr> <td>2</td> <td>AIN4</td> <td>AIN5</td> </tr> <tr> <td>3</td> <td>AIN6</td> <td>AIN7</td> </tr> </tbody> </table>		模拟输入		V <sub>plus</sub>	V <sub>minus</sub>	0	AIN0	AIN1	1	AIN2	AIN3	2	AIN4	AIN5	3	AIN6	AIN7
	模拟输入																	
	V <sub>plus</sub>	V <sub>minus</sub>																
0	AIN0	AIN1																
1	AIN2	AIN3																
2	AIN4	AIN5																
3	AIN6	AIN7																

		<p>差分输入电压(<math>V_{diff} = V_{plus} - V_{minus}</math>)</p> <p>注: 在差分输入模式下, 只需要在ADCHER使能两个相应通道之一. 转换结果将放置于相应的使能通道的寄存器里, 如果差分输入对两个通道都使能, ADC在扫描模式下转换两次, 然后将转换结果存入两个相应的数据寄存器.</p>
[9]	保留	-
[8]	TRGE	<p><b>外部触发使能</b></p> <p>使能或禁用A/D 转换 (通过外部STADC引脚)</p> <p>1= 使能</p> <p>0= 禁用</p>
[7:6]	TRGCOND	<p><b>外部触发条件</b></p> <p>该 2 位 决定外部STADC 引脚触发为(电平触发还是边沿触发. 该信号必须保持至少8 PCLKs的稳定状态; 边沿触发下, 至少保持4 PCLKs 的高电平或低电平状态.</p> <p>00 = 低电平</p> <p>01 = 高电平</p> <p>10 = 下降沿</p> <p>11 = 上升沿</p>
[5:4]	TRGS	<p><b>硬件触发生源</b></p> <p>00 =设定外部STADC引脚启动A/D 转换</p> <p>其它 =保留</p> <p>改变 TRGS 前, 软件需要禁用TRGE 和 ADST.</p> <p>在硬件触发模式下, STADC外部引脚触发置位ADST 位.</p>
[3:2]	ADMD	<p><b>A/D 转换模式</b></p> <p>00 = 单次转换</p> <p>01 = 突发转换</p> <p>10 = 单周期扫描</p> <p>11 = 连续扫描</p> <p>当改变转换模式时, 软件要首先禁用 ADST 位</p> <p>注: 在突发模式下, A/D转换结果总是存储在数据寄存器0中</p>
[1]	ADIE	<p><b>A/D 中断使能</b></p> <p>1 = 使能 A/D 中断功能</p> <p>0 = 禁用 A/D 中断功能</p> <p>如果ADIE置位, A/D 转换结束产生中断请求.</p>
[0]	ADEN	<p><b>A/D 转换使能</b></p> <p>1 = 使能</p> <p>0 = 禁用</p> <p>开始 A/D 转换功能时,该位需 置位. 该位为 0 将禁用 A/D 转换模拟电路的电源供给.</p>



## A/D通道使能寄存器(ADCHER)

寄存器	偏移量	R/W	描述	复位后的值
ADCHER	ADC_BA+0x24	R/W	A/D通道使能	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留						PRESEL[1:0]	
7	6	5	4	3	2	1	0
CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0

Bits	描述	
[31:10]	保留	-
[9:8]	PRESEL[1:0]	模拟输入通道7选择 00= 外部模拟输入 01= 内部参考源电压 10= 保留 11= 保留
[7]	CHEN7	模拟输入通道 7使能 1 = 使能 0 = 禁用
[6]	CHEN6	模拟输入通道 6使能 1 = 使能 0 = 禁用
[5]	CHEN5	模拟输入通道 5使能 1 = 使能 0 = 禁用
[4]	CHEN4	模拟输入通道 4使能

		1 = 使能 0 = 禁用
[3]	CHEN3	模拟输入通道 3使能 1 = 使能 0 = 禁用
[2]	CHEN2	模拟输入通道 2使能 1 = 使能 0 = 禁用
[1]	CHEN1	模拟输入通道 1使能 1 = 使能 0 = 禁用
[0]	CHEN0	模拟输入通道 0使能 1 = 使能 0 = 禁用 当 CHEN1~7 设定为 0时，该位使能。 在单一模式下，软件使能多通道，仅最小序号通道进行转换，其他通道将被忽视。

## A/D 比较寄存器 0/1 (ADCMR0/1)

寄存器	偏移量	R/W	描述	复位后的值
ADCMR0	ADC_BA+0x28	R/W	A/D比较寄存器0	0x0000_0000
ADCMR1	ADC_BA+0x2C	R/W	A/D比较寄存器1	0x0000_0000

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

Bits	描述	
[31:28]	保留	-
[27:16]	<b>CMPD</b>	<b>比较数值</b> 此12 位数值将和 指定通道的转换结果相比较,在扫描模式下(不增加程序负载)可用软件监控外部模拟输入 引脚电压转换.
[15:12]	保留	-
[11:8]	<b>CMPMATCNT</b>	<b>比较匹配值</b> 当指定A/D通道的转换值和比较条件CMPCOND[2]相匹配, 内部计数器将相应的加1.当内部计数器的值达到设定值时, (CMPMATCNT +1) 硬件将置位CMPF 位.
[5:3]	<b>CMPCH</b>	<b>Compare 通道选择</b> 000 = 选择比较通道0 转换结果. 001 = 选择比较通道1 转换结果. 010 = 选择比较通道2 转换结果. 011 = 选择比较通道3 转换结果. 100 = 选择比较通道4 转换结果. 101 = 选择比较通道5 转换结果. 110 = 选择比较通道6 转换结果.

		111 = 选择比较通道7 转换结果.
[2]	<b>CMPCOND</b>	<p><b>比较条件</b></p> <p>1= 设置比较条件即当 12 位 A/D 转换结果大于或等于 12 位 CMPD(ADCMPrx[27:16]), 内部匹配计数器加1.</p> <p>0= 设置比较条件即当 12 位 A/D 转换结果小于 12 位 CMPD(ADCMPrx[27:16]), 内部匹配计数器减1.</p> <p>注: 当内部计数器的值达到(CMPMATCNT +1), CMPF 置位.</p>
[1]	<b>CMPIE</b>	<p><b>比较中断使能</b></p> <p>1 = 使能比较功能中断</p> <p>0 = 禁用比较功能中断</p> <p>如果使能比较功能, 且比较条件与CMPCOND和CMPMATCNT的设置匹配, CMPF位有效, 同时, 如果CMPIE 置1, 产生比较中断请求.</p>
[0]	<b>CMPEN</b>	<p><b>比较使能</b></p> <p>1 = 使能比较.</p> <p>0 = 禁用比较.</p> <p>当转换数据装载到ADDR 寄存器时, 该位置位使能ADC控制器比较CMPD[11:0]与特定通道的转换值.</p>

**A/D 状态寄存器 (ADSR)**

寄存器	偏移量	R/W	描述	复位后的值
ADSR	ADC_BA+0x30	R/W	ADC状态寄存器	undefined

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
OVERRUN							
15	14	13	12	11	10	9	8
VALID							
7	6	5	4	3	2	1	0
保留	CHANNEL			BUSY	CMPF1	CMPF0	ADF

Bits	描述	
[31:24]	保留	-
[23:16]	OVERRUN	结束运行标志 (只读) ADDRx的OVERRUN位的镜像 ADC工作于突发模式, 若FIFO超出限度, OVERRUN[7:0] 全部置1.
[15:8]	VALID	数据有效标志位 (只读) ADDRx的VALID位的镜像 ADC工作于突发模式, 若FIFO失效 VALID [7:0] 全部置1.
[7]	保留	-
[6:4]	CHANNEL	当前转换通道 这3位在BUSY=1时 表示进行转换中的通道. 当BUSY=0, 表示可进行下次转换的通道. 只读位.
[3]	BUSY	忙/空闲 1 = A/D 转换器忙碌 0 = A/D 转换器空闲 该位是ADST 位(ADCR). 的镜像

		只读位.
[2]	<b>CMPF1</b>	<p><b>比较标志位</b></p> <p>选择 A/D 转换通道 结果和ADCMPR1相匹配 该位置1. 写 1 清该位.</p> <p>1 = ADDR 转换结果和 ADCMPR1相匹配</p> <p>0 = ADDR 转换结果和 ADCMPR1不匹配</p>
[1]	<b>CMPF0</b>	<p><b>比较标志位</b></p> <p>选择 A/D 转换通道 结果和ADCMPR 0相匹配 该位置1. 写 1 清该位.</p> <p>1 = ADDR 转换结果和 ADCMPR0相匹配</p> <p>0 = ADDR 转换结果和 ADCMPR0不匹配</p>
[0]	<b>ADF</b>	<p><b>A/D转换结束标志位</b></p> <p>状态标志位 指示A/D 转换结束.</p> <p>ADF 在下列三个条件时置位:</p> <ol style="list-style-type: none"> <li>1. 单次转换模式下A/D转换结束时</li> <li>2. 扫描模式下在所有指定通道A/D转换结束时.</li> <li>3. 突发模式下, FIFO存储多于4个转换结果</li> </ol> <p>该标志写1清零.</p>

## A/D 校准寄存器 (ADCALR)

寄存器	偏移量	R/W	描述	复位后的值
ADCALR	ADC_BA+0x34	R/W	A/D 校准寄存器	0x0000_0000

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

Bits	描述	
[31:2]	保留	-
[1]	<b>CALDONE</b>	<p><b>校准完成标志 (只读)</b></p> <p>1 = A/D 转换 自校准完成.</p> <p>0 = A/D转换无自校准 或 自校准进行中 (若CALEN 位置位).</p> <p>CALEN 位写0, CALDONE 位将由硬件立即清零, 该位只读.</p>
[0]	<b>CALEN</b>	<p><b>自身校准 功能</b></p> <p>1 = 使能 自校准</p> <p>0 = 禁用 自校准</p> <p>软件置位 该位使能 A/D 转换执行自校准 功能.需要 127 ADC 时钟 完成校准功能. CALDONE 置位后 该位需保持为高, 清该位将禁用自校准 功能.</p>

## 6.12 外部总线接口 (EBI)

### 6.12.1 简介

NuMicro M051™ 系统配备一个外部总线接口 (EBI), 以供外部设备使用。

为保护外部设备与芯片的连接, EBI支持地址总线与数据总线复用模式. 且地址锁存使能 (ALE)信号能区分地址与数据周期.

### 6.12.2 特性

外部总线接口有下列功能:

1. 支持外部设备最大64K字节 (8位数据宽度)/128K字节(16位数据宽度)
2. 支持可调外部总线基本时钟 (MCLK)
3. 支持8位或 16 位数据宽度
4. 支持可调的数据访问时间 (tACC), 地址锁存使能时间(tALE) 和地址保持时间(tAHD)
5. 支持地址总线和数据总线复用以保护地址管脚
6. 支持可配置的空闲周期用于不同访问条件: 写命令结束 (W2X), 连续读 (R2R)。



6.12.3 EBI 框图

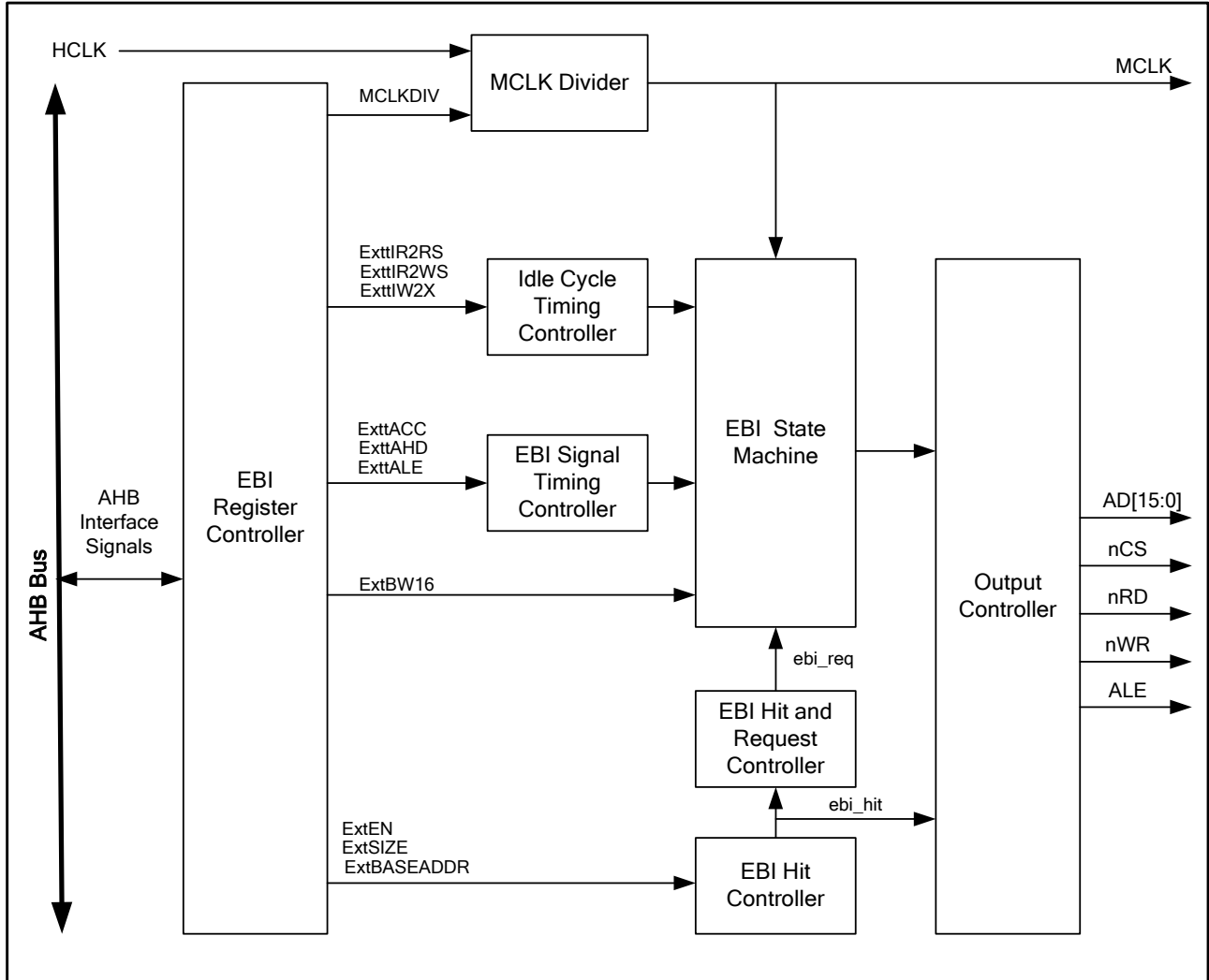


图 6.12-1 EBI 框图

## 6.12.4 操作步骤

### 6.12.4.1 EBI Area and Address Hit

NuMicro M051™ 系列EBI地址在0x6000\_0000 ~ 0x6001\_FFFF，总共内存空间为128K字节。当系统申请的地址在EBI的内存空间，相应的EBI片选信号有效，EBI状态机工作。

对于8-bit device (64Kbyte)，EBI 所占地址是0x6000\_0000 ~ 0x6000\_FFFF 和 0x6001\_0000 ~ 0x6001\_FFFF。

### 6.12.4.2 EBI 数据宽度连接

NuMicro M051™ 系列EBI都支持地址总线和数据总线复用。对于外部器件，地址与数据总线分开，与器件的连接需要额外的逻辑单元来锁存地址。这样，ALE需要连接到锁存器（如74HC373）上。Ad为锁存器的输入引脚，锁存器的输出连接到外部器件的地址总线上。对于16位设备，AD [15:0] 由地址线与16位数据线共用。对于8位设备，仅AD [7:0] 由地址与8位数据共用，AD [15:8]作地址，直接与8位器件连接。

对于8位数据宽度，NuMicro M051™ 系统地址[15:0] 作为设备地址[15:0]。对于16位数据宽度，NuMicro M051™ 系统地址[16:1] 作为设备地址[15:0]，在 NuMicro M051™ 系统中地址位bit [0] 不用。

EBI bit width	System address (AHBADR)	EBI address (AD)
8 bit	AHBADR[15:0]	AD[15:0]
16 bit	AHBADR[16:1]	AD[15:0]

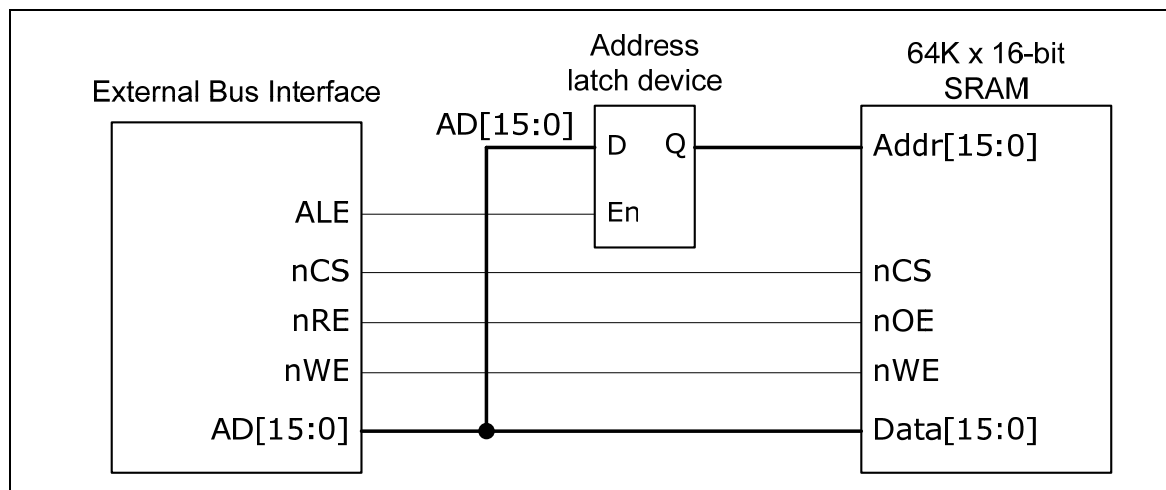


图 6.12-2 16位EBI数据宽度与16位器件连接

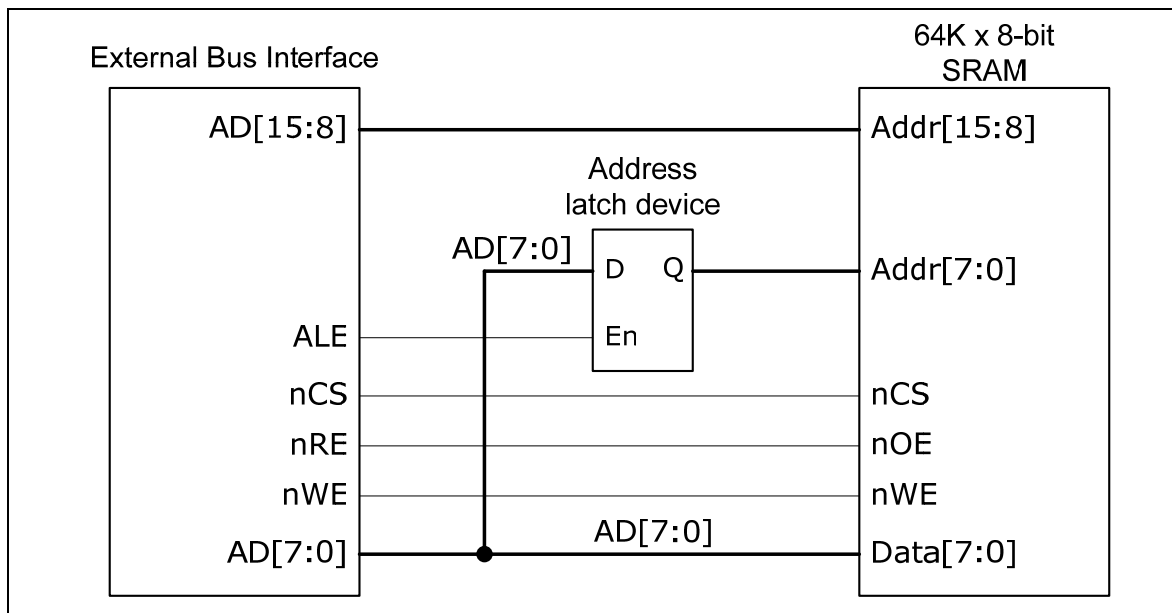


图 6.12-3 8位EBI数据宽度与8位设备连接

当系统访问数据宽度大于EBI的数据宽度，EBI控制器通过多次执行EBI访问来完成操作。例如，如果系统通过EBI器件请求32位数据，如果EBI为8位数据宽度，EBI控制器将访问4次来完成操作。

#### 6.12.4.3 EBI 操作控制

##### MCLK 控制

NuMicro M051™ 系列中，EBI工作时，通过MCLK同步所有EBI信号。当NuMicro M051™ 系列以较低工作频率连接到外部器件，MCLK 可以通过设置寄存器EBICON 中的MCLKDIV分频，最小可达HCLK/32。因此，NuMicro M051™ 可以适用于宽频率范围的EBI 设备。如果 MCLK 设置为HCLK/1，EBI 信号可以与MCLK的上升沿同步，也可以与MCLK的下降沿同步。

##### 操作与访问时序控制

开始访问时，片选 (nCS)置低并等待一个MCLK地址建立时间(tASU)以使地址稳定。地址稳定后ALE置高并保持一段时间 (tALE) 以用于地址锁存。地址锁存后，ALE 置低并等待一个MCLK 的周期锁存保持时间 (tLHD) 和另一个MCLK 的周期 (tA2D) 用于总线转换（地址到数据）。然后当读时nRD 置低或写时nWR 置低。在保持访问时间(tACC用于读取输出稳定或者完成写入)后置高用于EBI 信号保持数据访问时间 (tAHD)和片选置高，地址由当前访问控制释放。

NuMicro M051™ 系列提供灵活的EBI 时序控制以用于不同外部设备。在NuMicro M051™ EBI 的时序控制中，tASU, tLHD 和 tA2D固定为1个MCLK周期，tAHD 可以通过设置寄存器EXTIME的ExttAHD在1~8 MCLK 周期调节，tACC可以通过设置寄存器EXTIME的ExttACC在1~32 MCLK周期调节，tALE可以通过寄存器EBICON的tALE在1~8 MCLK 周期调节。

参数	值	单位	描述
tASU	1	MCLK	地址锁存建立时间.
tALE	1 ~ 8	MCLK	ALE 高电平时间. 由EBICON的ExttALE控制.
tLHD	1	MCLK	地址锁存保持时间.
tA2D	1	MCLK	地址到数据的延迟 (总线转换时间).
tACC	1 ~ 32	MCLK	数据访问时间. 由EXTIME的ExttACC控制.
tAHD	1 ~ 8	MCLK	数据访问保持时间. 由EXTIME 的ExttAHB 控制.
IDLE	1 ~ 15	MCLK	空闲周期.由EXTIME 的ExttIR2R, ExttIR2W 和 ExttIW2X 控制.

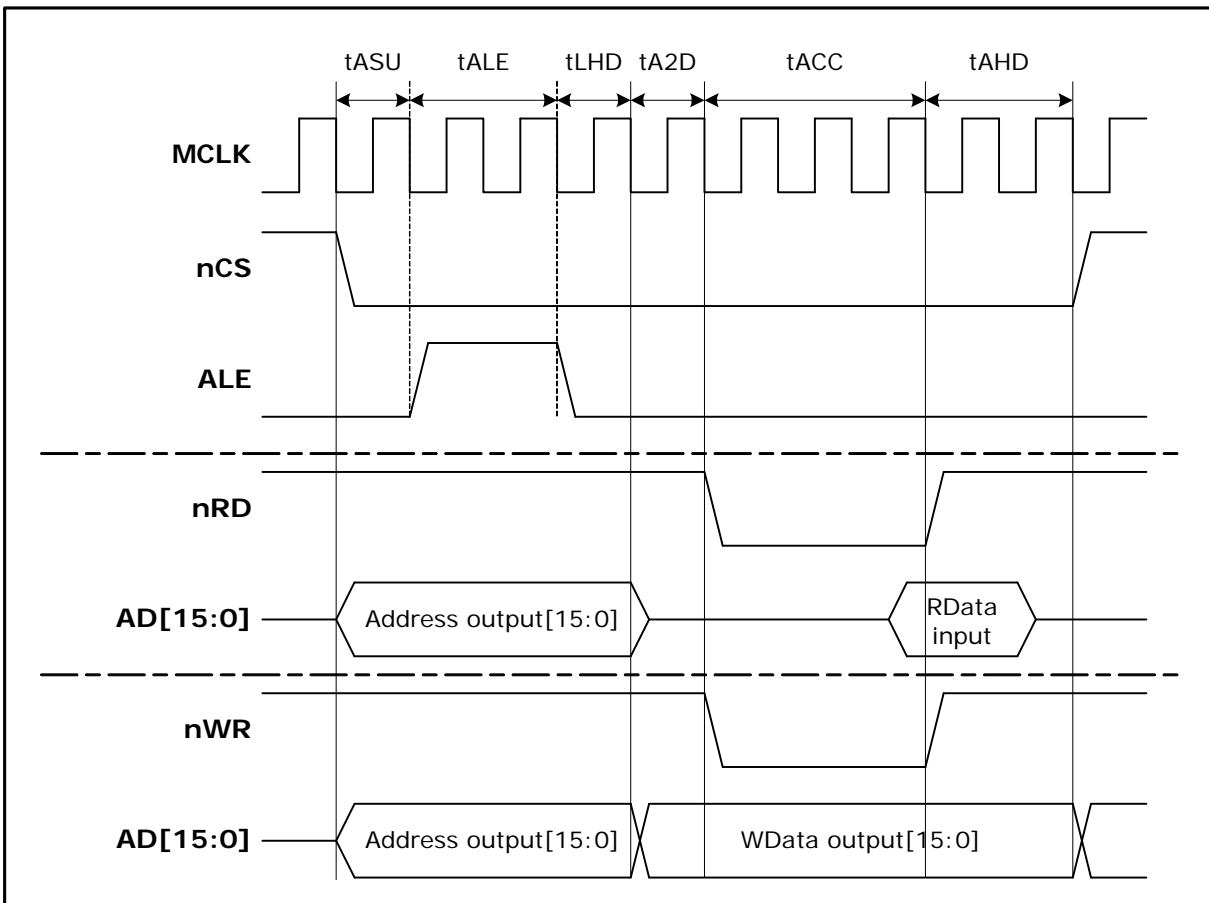


图 6.12-4 16位数据宽度的时序控制波形

上述时序波形是以16位数据宽度为例. 此例中, AD 总线用作地址 [15:0] 和数据 [15:0]. 当 ALE 置高, AD 为地址输出. 在地址锁存后, ALE 置低并且 AD 总线转换成高阻以等等设备输出数据 (在读取访问操作时), 或用于写数据输出.

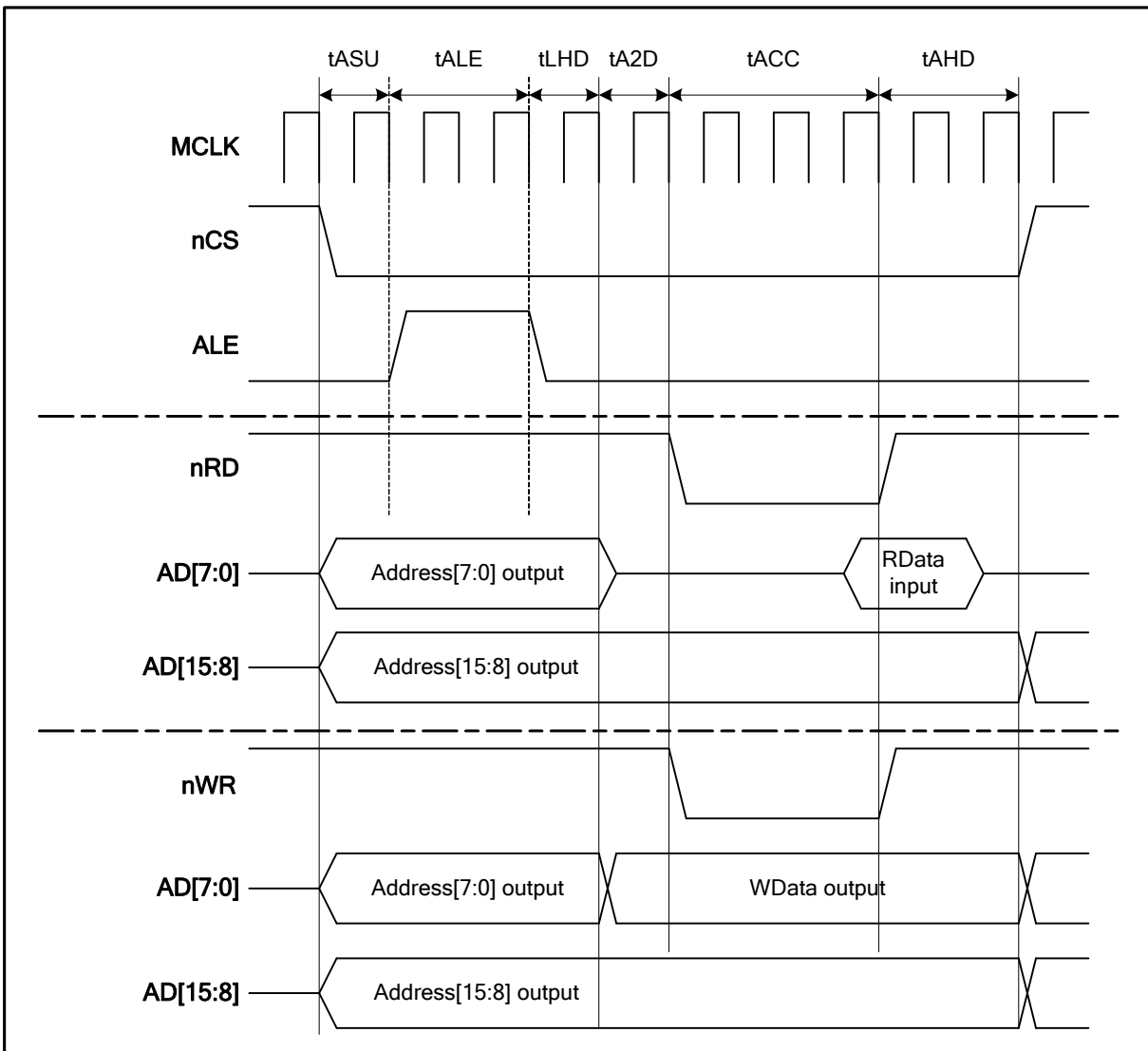


图 6.12-5 8位数据宽度时序控制波形

上述时序波形是以8位数据宽度为例。与16位数据宽度不同的是AD[15:8]的使用。在8位数据宽度的设置，AD[15:8] 固定为地址位 [15:8] 的输出，所以外部锁存仅需要8位宽度。

### 插入空闲周期

当EBI 连续访问时，如果设备访问速度较慢，可能会有总线冲突。NuMicro M051™ 支持额外空闲周期以解决该问题。在空闲周期，EBI的所有控制信号无效。下图为空闲周期波形图：

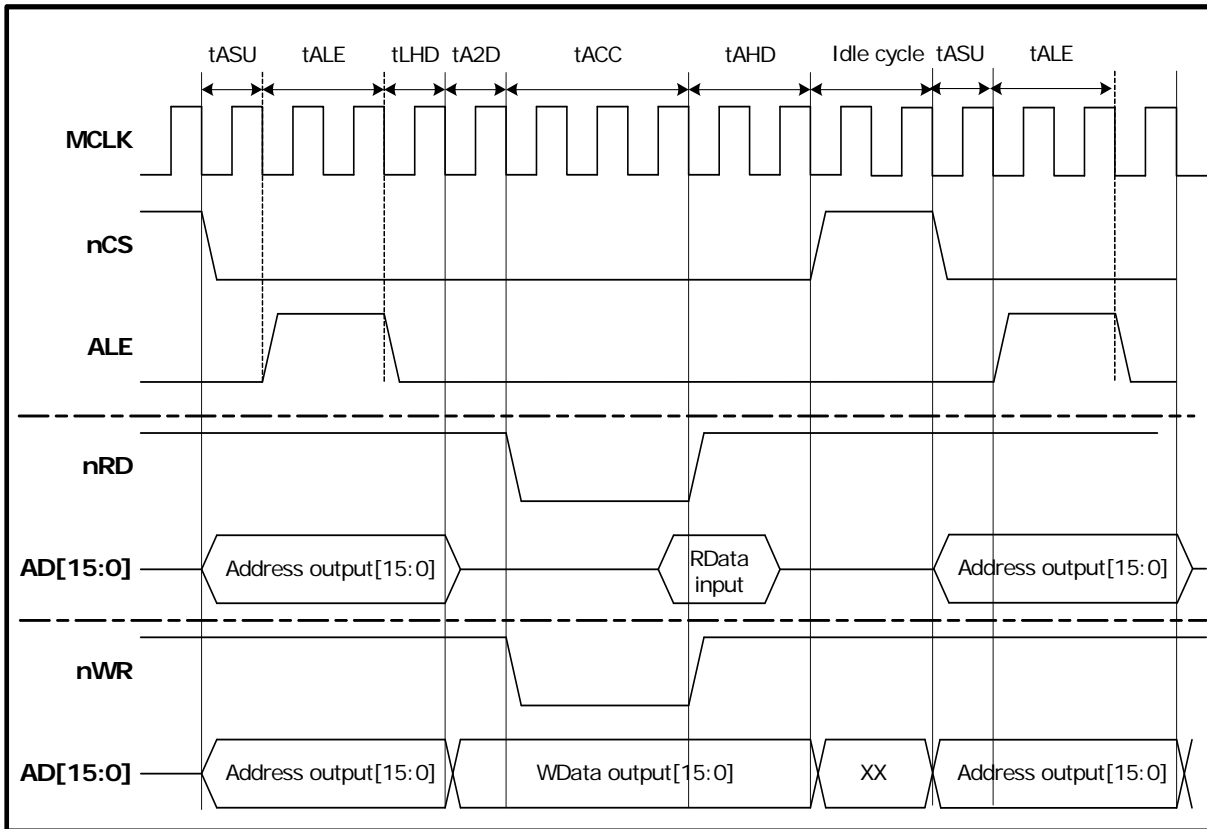


图 6.12-6 插入空闲周期的时序控制波形

在下面两种情况下，EBI可插入空闲周期：

- 写访问之后
- 读访问之后与下一个读访问之前

通过设置寄存器EXTIME的ExtIW2X, ExtIR2R, 与 ExtIR2W, 空闲周期可设定在0~15 MCLK.



## 6.12.5 EBI 控制器寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>EBI_CTL_BA = 0x5001_0000</b>				
EBICON	EBI_CTL_BA+0x00	R/W	外部总线接口控制寄存器	0x0000_0000
EXTIME	EBI_CTL_BA+0x04	R/W	外部总线接口时序控制寄存器	0x0000_0000

## 6.12.6 EBI 控制器寄存器描述

### 外部总线接口控制寄存器(EBICON)

寄存器	偏移量	R/W	描述	复位后的值
EBICON	EBI_CTL_BA+0x00	R/W	外部总线接口控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留					ExttALE		
15	14	13	12	11	10	9	8
保留					MCLKDIV		
7	6	5	4	3	2	1	0
保留						ExtBW16	ExtEN

Bits	描述	
[31:19]	保留	保留
[18:16]	ExttALE	ALE的扩展时间 通过ExttALE控制地址锁存ALE 时间宽度 (tALE) $tALE = (ExttALE+1)*MCLK$
[15:11]	保留	保留
[10:8]	MCLKDIV	外部输出时钟分频器



		<p>由 MCLKDIV 控制EBI输出时钟的频率，见下表:</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">MCLKDIV</th> <th style="width: 10%;">Output clock (MCLK)</th> </tr> </thead> <tbody> <tr><td>000</td><td>HCLK/1</td></tr> <tr><td>001</td><td>HCLK/2</td></tr> <tr><td>010</td><td>HCLK/4</td></tr> <tr><td>011</td><td>HCLK/8</td></tr> <tr><td>100</td><td>HCLK/16</td></tr> <tr><td>101</td><td>HCLK/32</td></tr> <tr><td>11X</td><td>默认</td></tr> </tbody> </table> <p>注: 默认输出时钟为HCLK/1</p>	MCLKDIV	Output clock (MCLK)	000	HCLK/1	001	HCLK/2	010	HCLK/4	011	HCLK/8	100	HCLK/16	101	HCLK/32	11X	默认
MCLKDIV	Output clock (MCLK)																	
000	HCLK/1																	
001	HCLK/2																	
010	HCLK/4																	
011	HCLK/8																	
100	HCLK/16																	
101	HCLK/32																	
11X	默认																	
[7:2]	保留	保留																
[1]	<b>ExtBW16</b>	<p><b>EBI 数据宽度为 16位</b></p> <p>该位配置数据总是8位还是16位.</p> <p>0 = EBI 数据宽度为8位</p> <p>1 = EBI 数据宽度为16位</p>																
[0]	<b>ExtEN</b>	<p><b>EBI 使能</b></p> <p>该位使能EBI.</p> <p>0 = 禁用EBI</p> <p>1 = 使能EBI</p>																

### 外部总线接口时序控制寄存器(EXTIME)

寄存器	偏移量	R/W	描述	复位后的值
<b>EXTIME</b>	EBI_CTL_BA+0x04	R/W	外部总线接口时序控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留				<b>ExtIR2R</b>			
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
<b>ExtIW2X</b>				保留	<b>ExttAHD</b>		

7	6	5	4	3	2	1	0
ExttACC					保留		

Bits	描述	
[31:28]	保留	保留
[27:24]	<b>ExttR2R</b>	<b>读与读之间的空闲状态周期</b> 当读完成且下一个动作也是读, 插入空闲状态周期且nCS在ExtIW2X非零时返回高. 空闲状态周期 = (ExttR2R*MCLK)
[23:16]	保留	保留
[15:12]	<b>ExttW2X</b>	<b>写之后的空闲状态</b> 当写完成, 插入空闲状态且nCS在ExtIW2X非零时返回高. Idle state cycle = (ExttW2X*MCLK)
[11]	保留	保留
[10:8]	<b>ExttAHD</b>	<b>EBI 数据访问保持时间</b> ExttAHD 配置数据访问保持时间(tAHD). $tAHD = (ExttAHD + 1) * MCLK$
[7:3]	<b>ExttACC</b>	<b>EBI 数据访问时间</b> ExttACC 配置数据访问时间 (tACC). $tACC = (ExttACC + 1) * MCLK$
[2:0]	保留	保留

## 6.13 Flash内存控制器(FMC)

### 6.13.1 简介

NuMicro M051™ 系列具有 64K/32K/16K/8K 字节的片上 FLASH EEPROM，用于存储应用程序 (APROM)，用户可以通过 ISP/IAP 更新 FLASH 中的程序。在系统编程 (ISP) 允许用户更新焊接在 PCB 板上的芯片中的程序。上电后，通过设置 Config0 的启动选择 (CBS) 确定 Cortex-M0 CPU 从 APROM 或 LDROM 读取代码。此外，NuMicro M051™ 系列为用户提供额外的 4k 字节的数据 FLASH，以供用户在芯片于 64/32/16/8K 字节 APROM 模式下系统掉电之前存储一些基于应用的数据。

### 6.13.2 特性

- 高达 50MHz 的零等待连续地址访问
- 64/32/16/8KB 应用程序内存 (APROM)
- 4kB 在系统编程 (ISP) 加载程序内存 (LDROM)
- 可配置的 4kB 数据 FLASH，带有 512 字节页擦除单元
- 在系统编程 (ISP)/在应用编程 (IAP) 更新片上 Flash EPROM
- 在电路编程 (ISP) 采用串行调试接口 (SWDB) 接口

### 6.13.3 FMC 框图

FLASH内存控制器由AHB从接口，ISP控制逻辑，烧写接口和FLASH宏接口时序控制逻辑组成. FLASH内存控制器框图如下所示:

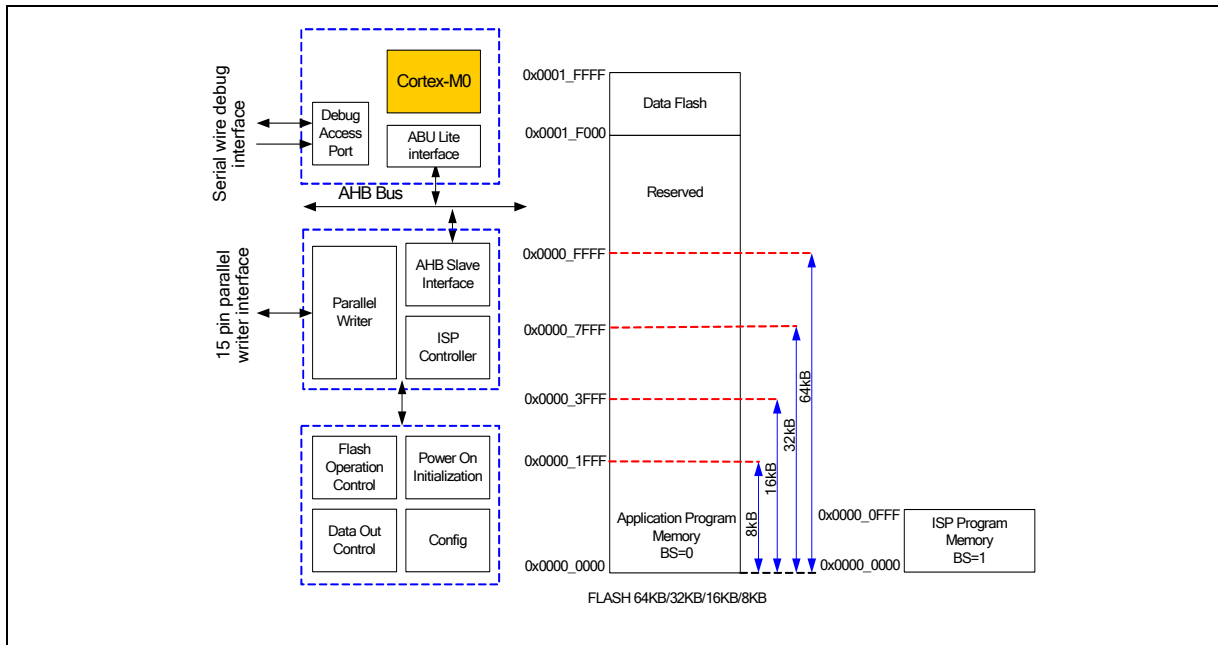


图 6.13-1 Flash 内存控制框图

#### 6.13.4 FMC内存结构

NuMicro M051™ 的flash 内存由程序内存(64/32/16/8KB)，数据FLASH，ISP加载程序内存，用户配置块组成. 用户配置块提供几个字节控制系统逻辑，如flash安全加密，启动选择，欠压电平，数据FLASH基地址等. 在上电期间，由FLASH内存加载到相应的控制寄存器中，用户根据烧写器的应用要求设置这些位，不用将芯片从PCB板上取下，数据FLASH的开始地址和大小可由用户根据应用配置，对于64/32/16/8KB的FLASH内存器件，其大小为4KB，开始地址为0x0001\_F000.

区块名称	大小	开始地址	结束地址
AP-ROM	8/16/32/64KB	0x0000_0000	0x0000_1FFF (8KB) 0x0000_3FFF (16KB) 0x0000_7FFF (32KB) 0x0000_FFFF (64KB)
Data Flash	4KB	0x0001_F000	0x0001_FFFF
LD-ROM	4KB	0x0010_0000	0x0010_0FFF
User Configuration	1 Words	0x0030_0000	0x0030_0000

表 6.13-1 Flash存储器地址映射

Flash内存组成如下所示:

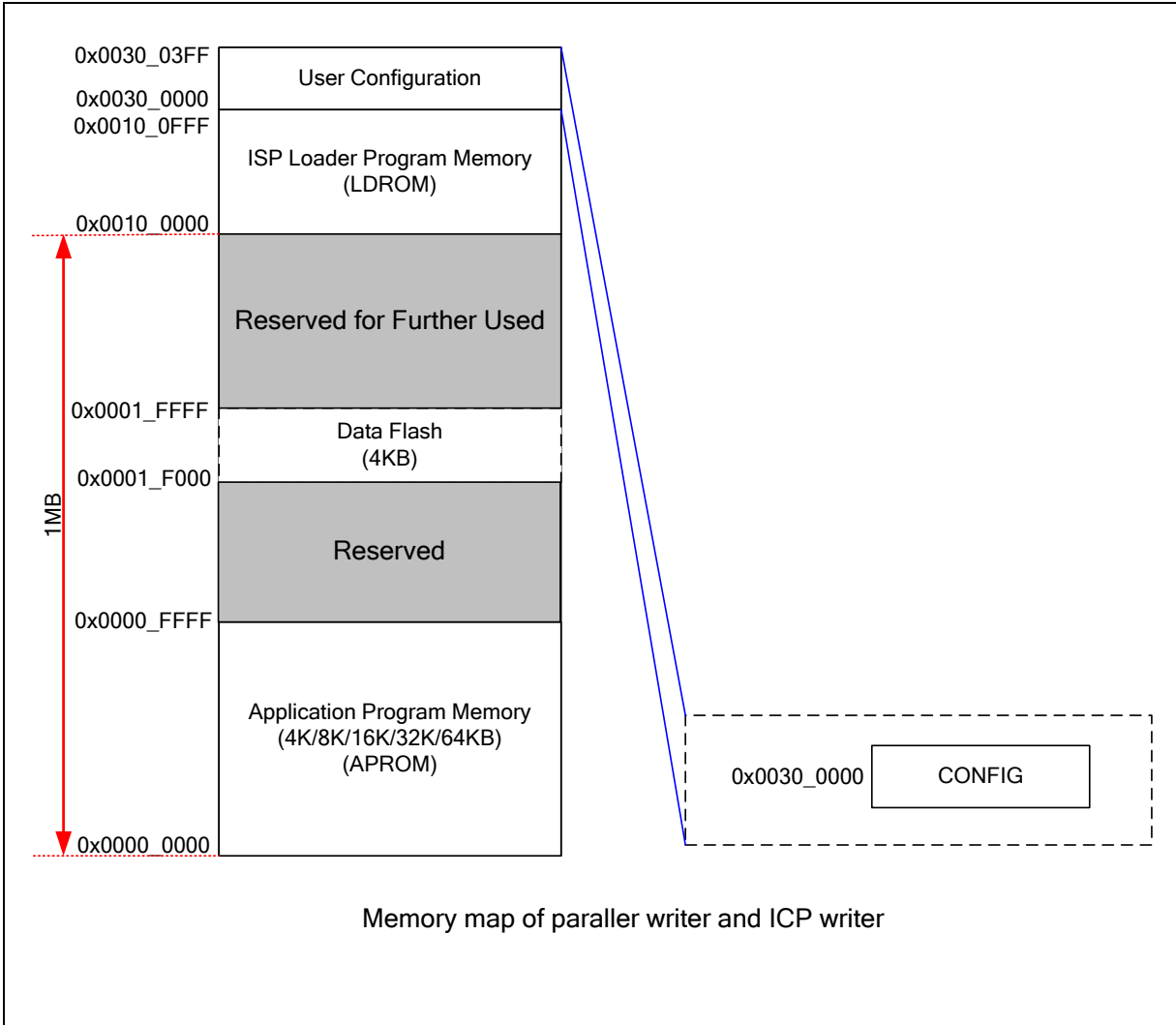


图 6.13-2 Flash 内存组成

## 6.13.5 启动选择

NuMicro M051™ 提供在系统编程 (ISP) 特征, 允许用户直接更新PCB板上芯片中的程序. 提供4kB程序内存专门用于存储ISP固件. 用户设置Config0的(CBS)以选择从APROM或LDRROM启动.

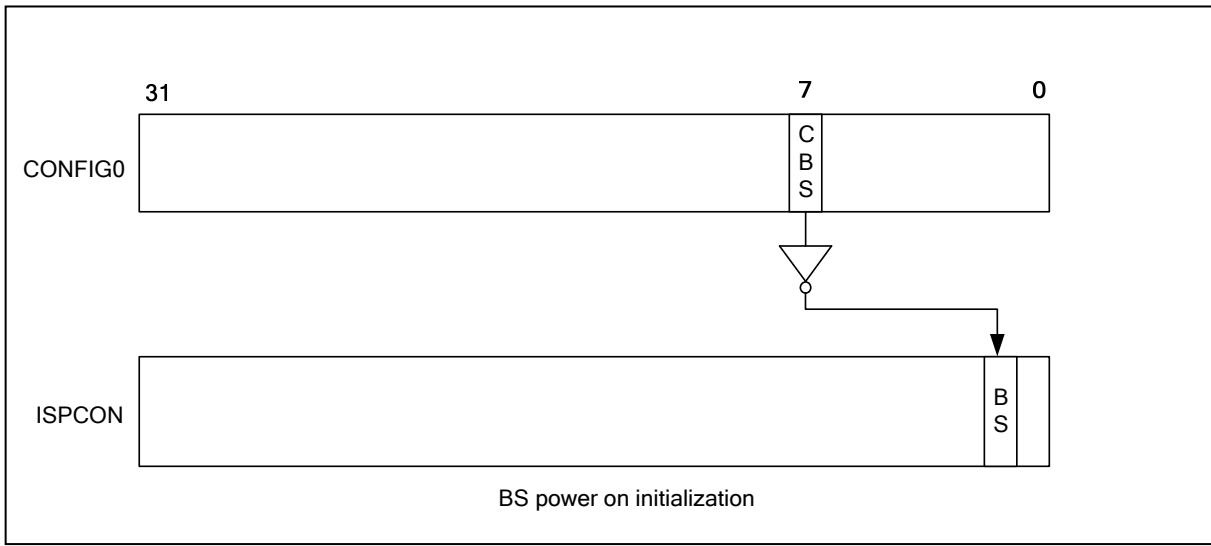


图 6.13-3 上电时启动选择(BS)

## 6.13.6 Data Flash

NuMicro M051™ 为用户提供数据FLASH. 通过ISP程序读/写. 擦除单元为512字节. 若要改变一个字, 需要先把所有128字拷贝到另外页或SRAM中. 对于8/16/32/64KB的flash器件, 数据FLASH的大小为4KB, 开始地址为0x0001\_F000.

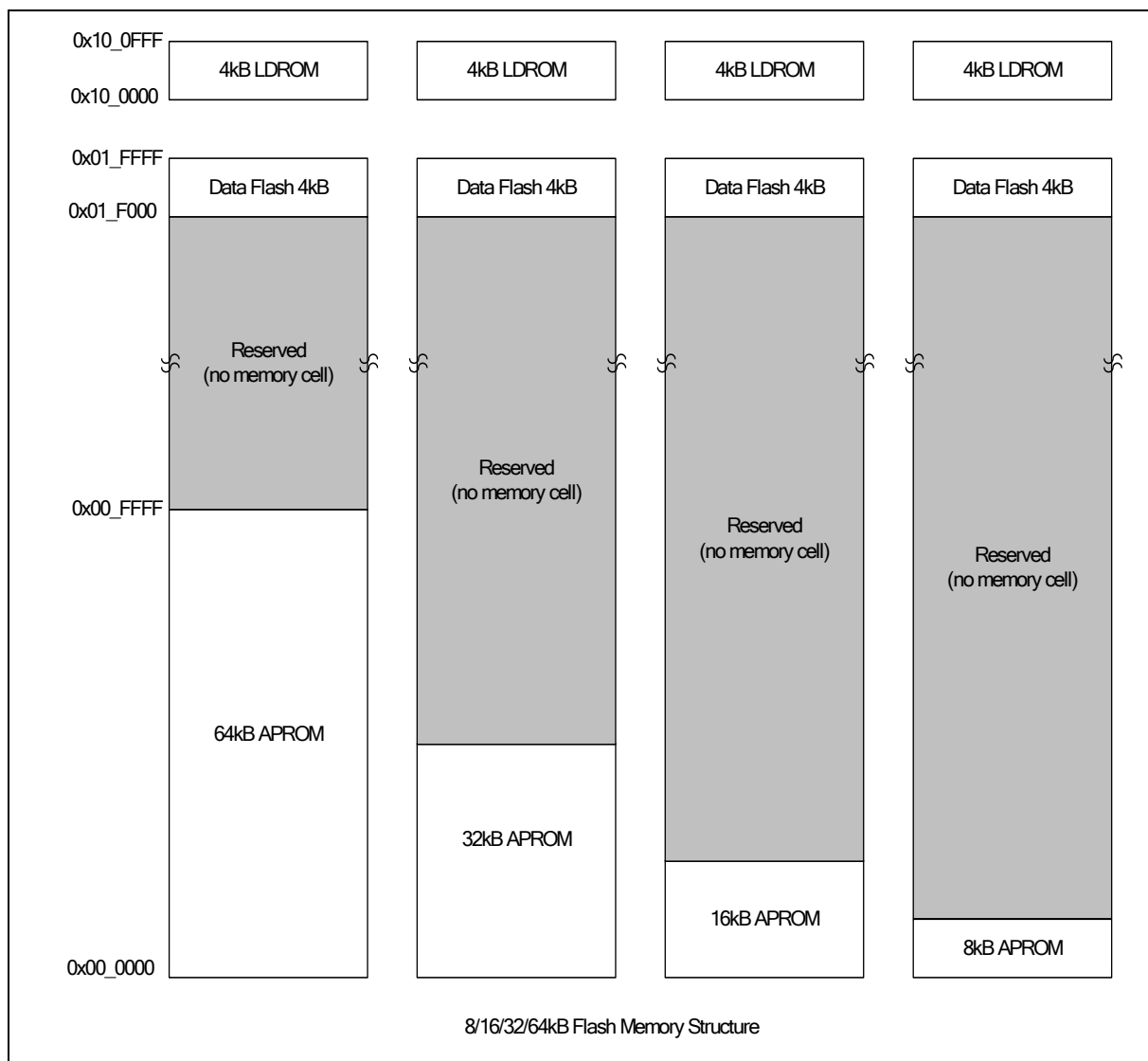


图 6.13-4 Flash 内存结构



### 6.13.7 在系统编程(ISP)

注: 使用ISP功能之前, 先设置ISP\_EN(AHBCLK[2])打开ISP时钟.

#### ■ ISP 时钟源框图

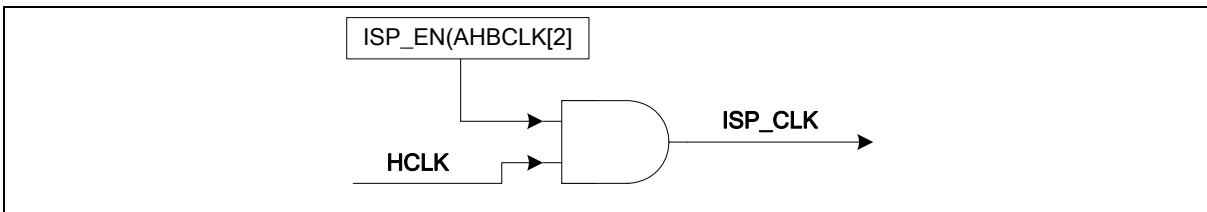


图 6.13-5 ISP 时钟源控制

程序内存和数据FLASH支持硬件编程和在系统编程 (ISP). 硬件编程模式在该产品进入批量生产状态时采用批量写, 以减小编程开销和上市时间. 若产品还在开发阶段或终端用户需要升级固件时, 硬件编程模式不是很方便, ISP模式能更好地适用于这种情况. NuMicro M051™支持 ISP 模式, 即通过软件控制来对器件重新编程. 而且, 这也使得更新固件得以广泛应用.

ISP 可以在没有将微控制器从系统中取下来的情况下执行编程. 各种接口使得LDR0M更容易更新程序代码. 最常用的方法是通过UART连接到LDR0M中的固件来执行ISP, PC一般都是通过串口传输新的APROM代码. LDR0M接收后, 通过ISP, 重新对APROM编程. Nuvoton 提供用于NuMicro M051™的ISP 固件和PC 应用程序. 用户采用Nuvoton ISP工具可以非常方便地执行ISP.

#### ■ ISP程序

NuMicro M051™ 支持从APROM 或 LDR0M启动由用户配置位 (CBS) 定义. 用户想更新APROM中的应用程序时, 可以写BS=1, 并开始软件复位使芯片由LDR0M启动. 向ISPEN写入1开始ISP功能. 在向ISPCON 寄存器写数据之前, SW 需要向全局控制寄存器 (GCR, 0x5000\_0100) 的寄存器REGWRPROT写入0x59, 0x16 和 0x88, 这个过程用于保护FLASH内存免受意外更改.

向ISPGO写入数据后, 要检查几个错误条件. 如果错误条件产生时, ISP操作失败, 其失败标志置位, ISPPFF标志由软件清零, 而不会在下次ISP操作时被覆盖, 即使ISPPFF保持为“1”, 下一次ISP也可以开始. 建议在每次ISP操作后, 如果ISPPFF被设置为1了, 就软件检查ISPPFF并将其清零.

如果ISPGO置位, CPU将等待ISP操作结束, 在此期间, 外设仍然在工作, 如果有中断请求时, CPU仍然会先执行完ISP后再响应中断.

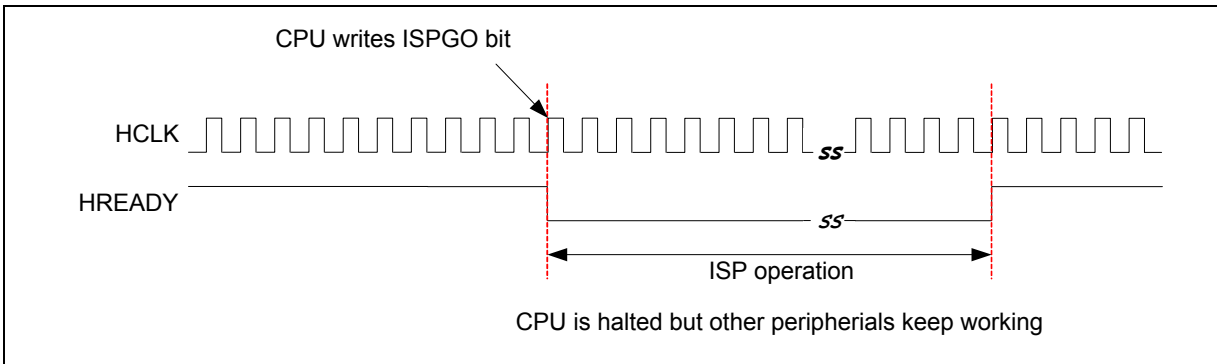


图 6.13-6 ISPGo 时序

注：NuMicro M051™ 允许用户通过ISP更新CONFIG的值，基于对应用程序安全考虑，软件在擦除CONFIG时，要先页擦除APROM，否则CONFIG不能被擦除。

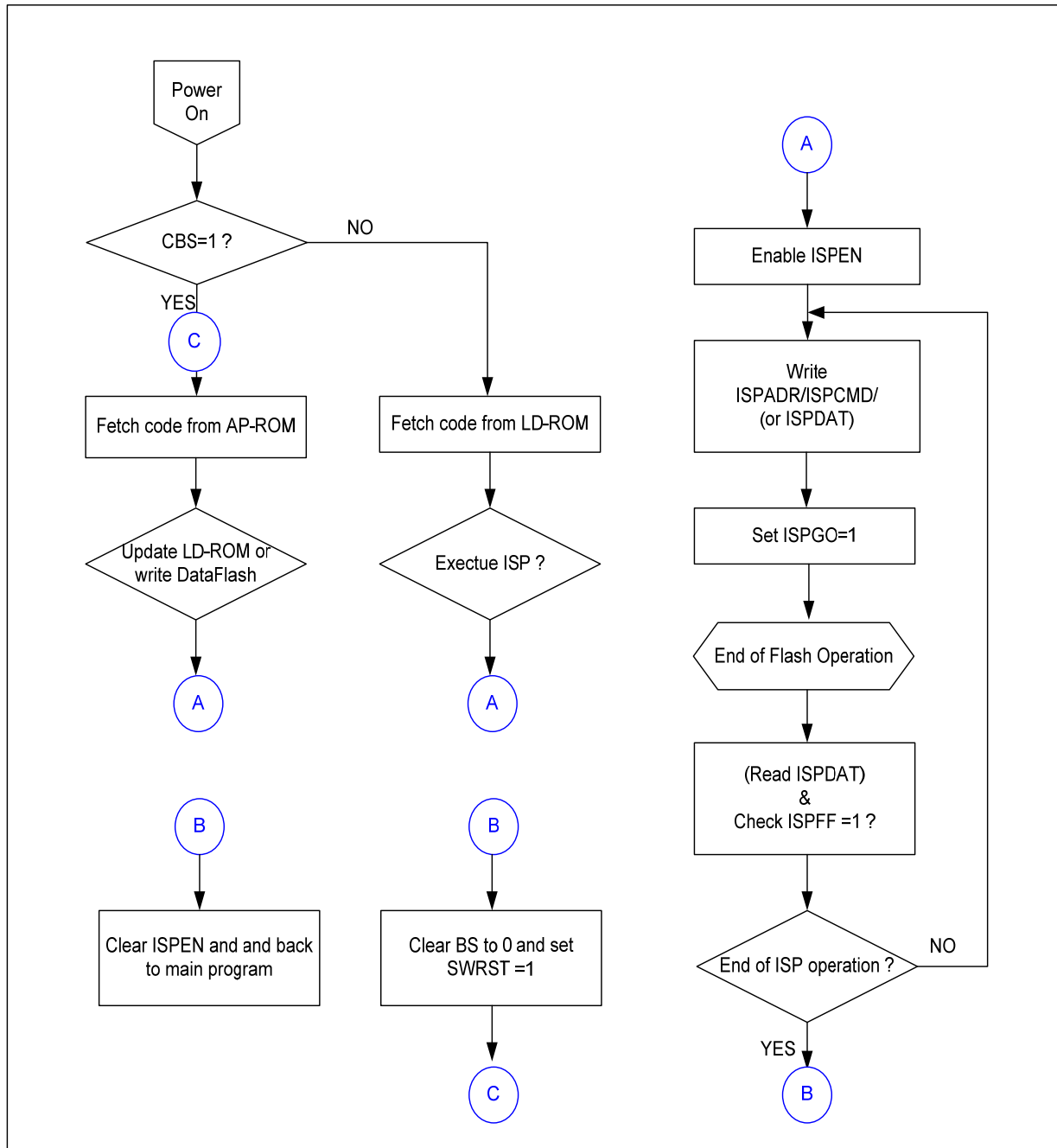


图 6.13-7 ISP 软件编程流程

ISP Mode	ISPCMD			ISPADR			ISPDAT
	FOEN	FCEN	FCTRL[3:0]	A21	A20	A[19:0]	D[31:0]
Standby	1	1	X	x	x	x	x
Read Company ID	0	0	1011	x	x	x	Data out D[31:0] = 0x0000_00DA
FLASH Page Erase	1	0	0010	0	A20 <sup>#1</sup>	Address in A[19:0]	x
FLASH Program	1	0	0001	0	A20 <sup>#1</sup>	Address in A[19:0]	Data in D[31:0]
FLASH Read	0	0	0000	0	A20 <sup>#1</sup>	Address in A[19:0]	Data out D[31:0]
CONFIG Page Erase	1	0	0010	1	1	Address in A[19:0]	x
CONFIG Program	1	0	0001	1	1	Address in A[19:0]	Data in D[31:0]
CONFIG Read	0	0	0000	1	1	Address in A[19:0]	Data out D[31:0]

表 6.13-2 ISP Mode

注1: A20=0 对应于 APROM和DATA, A20=1, 对应于 LDR0M

### 6.13.8 FMC控制寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>基地址 (FMC_BA) : 0x5000_C000</b>				
ISPCON	FMC_BA+0x000	R/W	ISP控制寄存器	0x0000_0000
ISPADR	FMC_BA+0x004	R/W	ISP地址寄存器	0x0000_0000
ISPDAT	FMC_BA+0x008	R/W	ISP数据寄存器	0x0000_0000
ISPCMD	FMC_BA+0x00C	R/W	ISP命令寄存器	0x0000_0000
ISPTRG	FMC_BA+0x010	R/W	ISP触发寄存器	0x0000_0000
DFBADR	FMC_BA+0x014	R	数据Flash 起始地址	0x0000_0000 0x0001_F000
FATCON	FMC_BA+0x018	R/W	FLASH访问窗口控制寄存器	0x0000_0000



## 6.13.9 FMC控制器寄存器描述

### ISP 控制寄存器(ISPCON)

寄存器	偏移量	R/W	描述	复位后的值
ISPCON	FMC_BA+0x00	R/W	ISP 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留	ET2	ET1	ET0	保留	PT2	PT1	PT0
7	6	5	4	3	2	1	0
SWRST	ISPPF	LDUEN	CFGUEN	保留		BS	ISPEN

Bits	描述																																					
[31:15]	保留	保留																																				
[14:12]	ET[2:0]	Flash 擦除时间																																				
		<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>ET[2]</th> <th>ET[1]</th> <th>ET[0]</th> <th>擦除时间 (ms)</th> </tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>20 (默认)</td> </tr> <tr> <td>0</td><td>0</td><td>1</td><td>25</td> </tr> <tr> <td>0</td><td>1</td><td>0</td><td>30</td> </tr> <tr> <td>0</td><td>1</td><td>1</td><td>35</td> </tr> <tr> <td>1</td><td>0</td><td>0</td><td>3</td> </tr> <tr> <td>1</td><td>0</td><td>1</td><td>5</td> </tr> <tr> <td>1</td><td>1</td><td>0</td><td>10</td> </tr> <tr> <td>1</td><td>1</td><td>1</td><td>15</td> </tr> </tbody> </table>	ET[2]	ET[1]	ET[0]	擦除时间 (ms)	0	0	0	20 (默认)	0	0	1	25	0	1	0	30	0	1	1	35	1	0	0	3	1	0	1	5	1	1	0	10	1	1	1	15
		ET[2]	ET[1]	ET[0]	擦除时间 (ms)																																	
		0	0	0	20 (默认)																																	
		0	0	1	25																																	
		0	1	0	30																																	
		0	1	1	35																																	
		1	0	0	3																																	
		1	0	1	5																																	
1	1	0	10																																			
1	1	1	15																																			
[11]	保留	保留																																				
[8:10]	PT[2:0]	Flash 编程时间																																				

		PT[2]	PT[1]	PT[0]	编程时间(us)
		0	0	0	40
		0	0	1	45
		0	1	0	50
		0	1	1	55
		1	0	0	20
		1	0	1	25
		1	1	0	30
		1	1	1	35
[7]	SWRST	<b>软件复位</b> 写1执行软件复位。 复位完成后由硬件清零			
[6]	ISPPF	<b>ISP失败标志</b> 当ISP满足下列条件时，该位由硬件置位： (1) APROM对自身写入。 (2) LDROM对自身写入。 (3) 目标地址无效，如超过正常范围。 注：写1清零。			
[5]	LDUEN	<b>LDROM更新使能</b> LDROM 更新使能位。 1 = MCU在APROM中运行时，LDROM可以被更新。 0 = 禁用LDROM更新			
[4]	CFGUEN	<b>配置更新使能</b> 写1使能S/W通过ISP更新配置位，不管此时程序是运行在APROM还是LDROM。 1 = 使能配置更新 0 = 禁用配置更新			
[2]	保留	保留			
[1]	BS	<b>启动选择</b> 该位为保护位，置位/清零该位选择下次是由LDROM启动还是由APROM启动，该位可作为MCU启动状态标志,用于检查MCU是由LDROM还是APROM启动的。上电复位后，该位初始值为config0的CBS的取反值；其他复位时保			

		持不变。 1 = 由LDROM启动 0 = 由 APROM启动
[0]	ISPEN	<b>ISP 使能</b> 该位是保护位，ISP 使能位，设置该位可以使能ISP功能。 1 = 使能 ISP 功能 0 = 禁用 ISP 功能



## ISP 地址 (ISPADR)

寄存器	偏移量	R/W	描述	复位后的值
ISPADR	FMC_BA+ 0x04	R/W	ISP 地址寄存器	0x0000_0000

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

Bits	描述	
[31:0]	ISPADR	ISP 地址 NuMicro M051™ 系列内置 32kx32 的 flash, 仅支持字编程. 执行 ISP 功能时, ISPADR[1:0] 必须为 00b.

## ISP 数据寄存器(ISPDAT)

寄存器	偏移量	R/W	描述	复位后的值
ISPDAT	FMC_BA+ 0x08	R/W	ISP 数据寄存器	0x0000_0000

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

Bits	描述	
[31:0]	ISPDAT	<b>ISP 数据</b> ISP操作之前, 写数据到该寄存器 ISP读操作后, 可从该寄存器读数据

## ISP 命令 (ISPCMD)

寄存器	偏移量	R/W	描述	复位后的值
ISPCMD	FMC_BA+ 0x0C	R/W	ISP命令寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留		FOEN	FCEN	FCTRL3	FCTRL2	FCTRL1	FCTRL0

Bits	描述																																				
[31:6]	保留	保留																																			
[5:0]	FOEN, FCEN, FCTRL	<p><b>ISP 命令</b></p> <p>ISP命令表如下:</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>操作模式</th> <th>FOEN</th> <th>FCEN</th> <th colspan="4">FCTRL[3:0]</th> </tr> </thead> <tbody> <tr> <td>待机</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>读</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>编程</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>Page Erase 页擦除</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	操作模式	FOEN	FCEN	FCTRL[3:0]				待机	1	1	0	0	0	0	读	0	0	0	0	0	0	编程	1	0	0	0	0	1	Page Erase 页擦除	1	0	0	0	1	0
操作模式	FOEN	FCEN	FCTRL[3:0]																																		
待机	1	1	0	0	0	0																															
读	0	0	0	0	0	0																															
编程	1	0	0	0	0	1																															
Page Erase 页擦除	1	0	0	0	1	0																															

## ISP触发控制寄存器(ISPTRG)

寄存器	偏移量	R/W	描述	复位后的值
ISPTRG	FMC_BA+ 0x10	R/W	ISP触发控制寄存器	0x0000_0000

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

Bits	描述	
[31:1]	保留	保留
[0]	ISPGO	<p><b>ISP开始触发</b></p> <p>写 1 开始ISP操作，当ISP操作结束后，该位由硬件自动清零.</p> <p>1 = ISP 即将执行</p> <p>0 = ISP 操作结束</p>

### 数据FLASH基地址寄存器(DFBADR)

寄存器	地址	R/W/C	描述	复位后的值
DFBADR	FMC_BA+ 0x14	R	数据FLASH基地址	0x0001_F000

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

Bits	描述
[31:0]	<p><b>DFBADR</b></p> <p><b>数据FLASH基地址</b></p> <p>该寄存器为数据FLASH开始地址寄存器,只读.</p> <p>对于 8/16/32/64KB flash 器件, 数据flash的大小为4KB , 由硬件决定起始地址为0x0001_F000.</p>



## Flash访问时间控制寄存器 (FATCON)

寄存器	偏移量	R/W	描述	复位后的值
FATCON	FMC_BA + 0x18	R/W	Flash访问时间控制寄存器	0x0000_0000

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

Bits	描述															
[31:5]	保留	保留														
[4]	L_SPEED	<p><b>Flash 低速模式使能</b></p> <p>1 = 无等待的Flash 访问(0等待状态)</p> <p>0 = 当FLASH访问不连续的地址时插入等待状态.</p> <p>注: 仅当HCLK ≤ 25 MHz设置该位. 如果HCLK &gt; 25 MHz, CPU 取到错误的代码, 导致失败.</p>														
[3:1]	FATS	<p><b>Flash 访问时间窗口选择</b></p> <p>这些位用于决定flash h感测放大器有效期.</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>FATS</th> <th>访问时间窗口 (ns)</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>40 (默认)</td> </tr> <tr> <td>001</td> <td>50</td> </tr> <tr> <td>010</td> <td>60</td> </tr> <tr> <td>011</td> <td>70</td> </tr> <tr> <td>100</td> <td>80</td> </tr> <tr> <td>101</td> <td>90</td> </tr> </tbody> </table>	FATS	访问时间窗口 (ns)	000	40 (默认)	001	50	010	60	011	70	100	80	101	90
FATS	访问时间窗口 (ns)															
000	40 (默认)															
001	50															
010	60															
011	70															
100	80															
101	90															

		110	100
		111	保留
[0]	<b>FPSEN</b>	<b>Flash 省电使能</b> 片上flash内存访问时间约为40ns, 如果CPU 时钟低于50 MHz, s/w 使能flash省电功能. 1 = 使能flash省电功能 0 = 禁用flash省电功能	



## 7 USER 配置

**CONFIG (Address = 0x0030\_0000)**

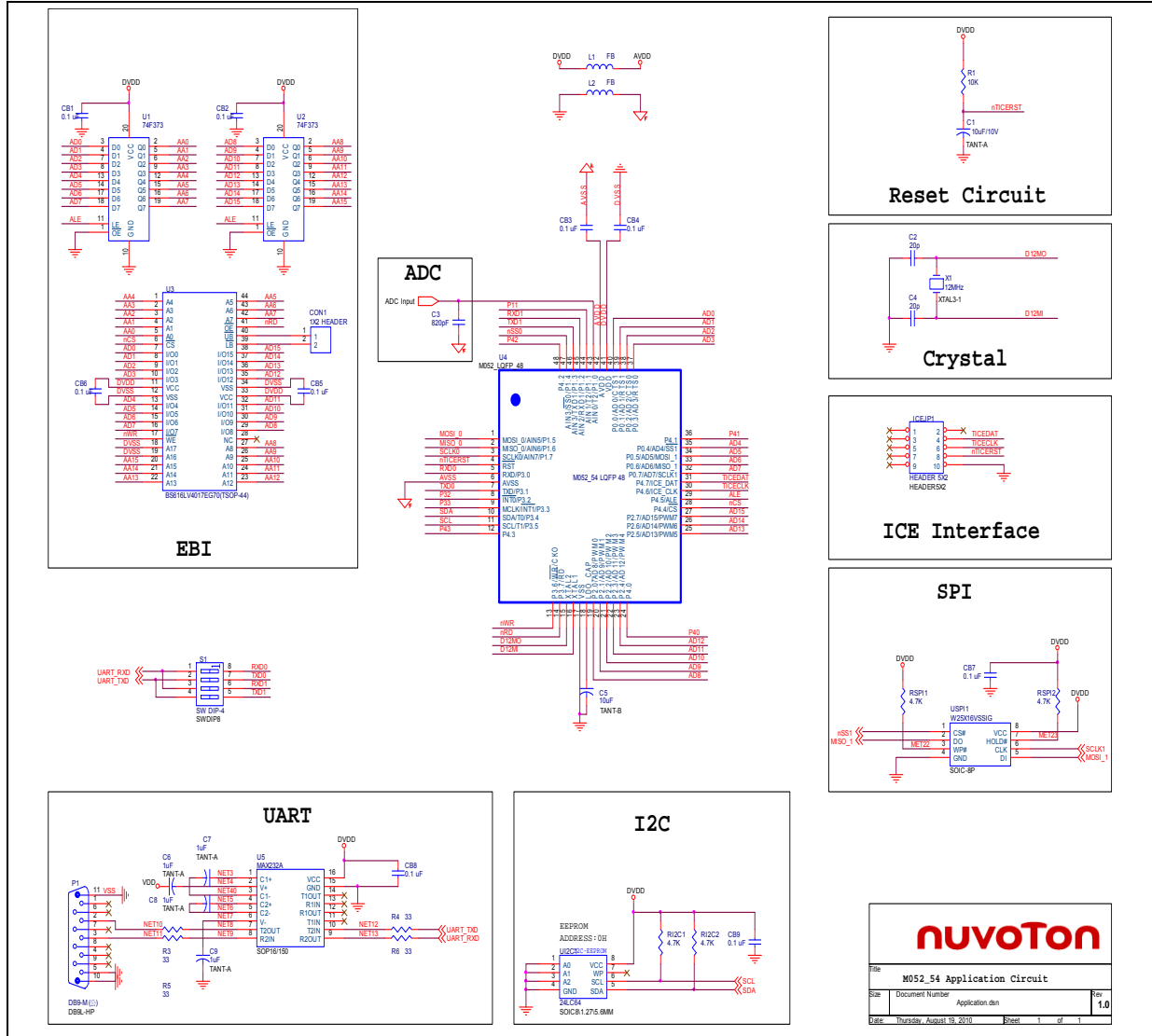
31	30	29	28	27	26	25	24
保留			CKF	保留	CFOSC		
23	22	21	20	19	18	17	16
CBODEN	CBOV1	CBOV0	CBORST	保留			
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
CBS	保留					LOCK	保留

Bits	描述									
[31:29]	保留	保留								
[28]	CKF	<b>XT1时钟滤波器使能</b> 0 = 禁用时钟滤波器 1 = 使能 XT1时钟滤波器								
[27]	保留	保留								
[26:24]	CFOSC	<b>复位后CPU 时钟源选择</b> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <tr> <td style="width: 30%;">FOSC[2:0]</td> <td>时钟源</td> </tr> <tr> <td>000</td> <td>外部晶振时钟 (4 ~ 24MHz)</td> </tr> <tr> <td>111</td> <td>内部 RC 22.1184 MHz 振荡器时钟</td> </tr> <tr> <td>其他</td> <td style="text-align: center;">保留</td> </tr> </table> 复位发生后，加载CFOSC 的值到CLKSEL0.HCLK_S[2:0].	FOSC[2:0]	时钟源	000	外部晶振时钟 (4 ~ 24MHz)	111	内部 RC 22.1184 MHz 振荡器时钟	其他	保留
		FOSC[2:0]	时钟源							
		000	外部晶振时钟 (4 ~ 24MHz)							
		111	内部 RC 22.1184 MHz 振荡器时钟							
其他	保留									
[23]	CBODEN	<b>欠压检测使能</b> 0= 上电后使能欠压检测 1= 上电后禁用欠压检测								
[22:21]	CBOV1-0	<b>欠压电压选择</b> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <tr> <td style="width: 30%;">CBOV1</td> <td style="width: 30%;">CBOV0</td> <td>欠压电压</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">4.5V</td> </tr> </table>	CBOV1	CBOV0	欠压电压	1	1	4.5V		
		CBOV1	CBOV0	欠压电压						
1	1	4.5V								



		1	0	3.8V	
		0	1	2.7V	
		0	0	2.2V	
[20]	<b>CBORST</b>	<b>欠压复位使能</b> 0 = 上电后使能欠压复位 1 = 上电后禁用欠压复位			
[19:8]	保留	保留			
[7]	<b>CBS</b>	<b>配置启动选择</b> 0 = 芯片从LDR0M启动 1 = 芯片从APROM启动			
[6:2]	保留	保留			
[1]	<b>LOCK</b>	<b>安全锁</b> 0 = Flash 数据锁定 1 = Flash 数据不锁定.  当锁定了flash数据, 仅有器件ID, Config0 和 Config1 可以通过烧录器和ICP通过串行调试接口读出。读出。其他数据锁定在0xFFFFFFFF. ISP 可以不管LOCK是否锁定都能读出数据.			
[0]	保留	保留			

## 8 典型应用电路



<b>nuvoTon</b>		
File:	M052_54 Application Circuit	
Sub:	Document Number	Application.dsn
Rev:		1.0
Date:	Thursday, August 19, 2010	Sheet 1 of 1

## ARM Cortex™ -M0

## 32-BIT 微控制器

## 9 电气特性

## 9.1 绝对最大额定值

参数	符号	最小值	最大值	单位
直流电源电压	VDD-VSS	-0.3	+7.0	V
输入电压	VIN	VSS-0.3	VDD+0.3	V
晶振频率	1/t <sub>CLCL</sub>	0	40	MHz
工作温度	TA	-40	+85	°C
贮存温度	TST	-55	+150	°C
V <sub>DD</sub> 最大流入电流		-	120	mA
V <sub>SS</sub> 最大流出电流			120	mA
单一管脚最大灌电流			35	mA
单一管脚最大源电流			35	mA
所有管脚最大灌电流总和			100	mA
所有管脚最大源电流总和			100	mA

注: 上表所列的条件中, 其极限值可能对器件的稳定有反作用.

## 9.2 DC电气特性

(VDD-VSS=3.3V, TA = 25°C, F<sub>OSC</sub> = 50MHz 在无特别说明的情况下.)

参数	符号	明细表				测试条件
		最小值	典型值	最大值	单位	
工作电压	V <sub>DD</sub>	2.5		5.5	V	V <sub>DD</sub> = 2.5V ~ 5.5V , 最高可达50 MHz
电源地	V <sub>SS</sub> AV <sub>SS</sub>	-0.3			V	
LDO 输出电压	V <sub>LDO</sub>	-10%	2.45	+10%	V	V <sub>DD</sub> > 2.7V
模拟输入带宽	V <sub>BG</sub>	-5%	1.26	+5%	V	V <sub>DD</sub> = 2.5V ~ 5.5V
模拟工作电压	AV <sub>DD</sub>	0		V <sub>DD</sub>	V	
普通模式下的工作电流 (50Mhz)	I <sub>DD1</sub>		32		mA	V <sub>DD</sub> = 5.5V@50 MHz, 使能所有的IP和PLL XTAL=12 MHz
	I <sub>DD2</sub>		24		mA	V <sub>DD</sub> = 5.5V@50 MHz, 禁用所有的IP, 使能PLL XTAL=12 MHz
	I <sub>DD3</sub>		31		mA	V <sub>DD</sub> = 3V@50 MHz, 使能所有的IP和PLL XTAL=12 MHz
	I <sub>DD4</sub>		23		mA	V <sub>DD</sub> = 3V@50 MHz, 禁用所有的IP, 使能PLL XTAL=12 MHz
普通模式下的工作电流 (12Mhz)	I <sub>DD5</sub>		17		mA	V <sub>DD</sub> = 5.5V@ 12MHz, 使能所有的IP, 禁用PLL, XTAL=12 MHz
	I <sub>DD6</sub>		14		mA	V <sub>DD</sub> = 5.5V@12 MHz, 禁用所有的IP和PLL, XTAL=12 MHz
	I <sub>DD7</sub>		16		mA	V <sub>DD</sub> = 3V@12 MHz, 使能所有的IP, 禁用PLL, XTAL=12 MHz
	I <sub>DD8</sub>		13		mA	V <sub>DD</sub> = 3V@12 MHz, 禁用所有的IP和PLL, XTAL=12 MHz

普通模式下的工作电流 (4Mhz)	$I_{DD9}$	12		mA	$V_{DD} = 5.5V@4\text{ MHz}$ , 使能所有的IP , 禁用PLL, XTAL=4MHz
	$I_{DD10}$	10		mA	$V_{DD} = 5.5V@4\text{ MHz}$ , 禁用所有的IP和PLL, XTAL=4MHz
	$I_{DD11}$	10		mA	$V_{DD} = 3V@4\text{ MHz}$ , 使能所有的IP , 禁用PLL, XTAL=4MHz
	$I_{DD12}$	9		mA	$V_{DD} = 3V@4\text{ MHz}$ , 禁用所有的IP和PLL, XTAL=4 MHz
空闲模式下的工作电流 (50Mhz)	$I_{IDLE1}$	19		mA	$V_{DD} = 5.5V@50\text{ MHz}$ , <b>使能所有的IP和PLL</b> XTAL=12 MHz
	$I_{IDLE2}$	11		mA	$V_{DD} = 5.5V@50\text{ MHz}$ , 禁用所有的IP , 使能PLL XTAL=12MHz
	$I_{IDLE3}$	18		mA	$V_{DD} = 3V@50\text{ MHz}$ , <b>使能所有的IP和PLL</b> XTAL=12 MHz
	$I_{IDLE4}$	10		mA	$V_{DD} = 3V@50\text{ MHz}$ , 禁用所有的IP , 使能PLL XTAL=12 MHz
空闲模式下的工作电流 (12Mhz)	$I_{IDLE5}$	10		mA	$V_{DD} = 5.5V@12\text{ MHz}$ , 使能所有的IP , 禁用PLL, XTAL=12 MHz
	$I_{IDLE6}$	7		mA	$V_{DD} = 5.5V@12\text{ MHz}$ , 禁用所有的IP和PLL, XTAL=12 MHz
	$I_{IDLE7}$	9		mA	$V_{DD} = 3V@12\text{ MHz}$ , 使能所有的IP , 禁用PLL, XTAL=12 MHz
	$I_{IDLE8}$	6		mA	$V_{DD} = 3V@12\text{ MHz}$ , 禁用所有的IP和PLL, XTAL=12 MHz
空闲模式下的工作电流 (4Mhz)	$I_{IDLE9}$	5		mA	$V_{DD} = 5.5V@4\text{ MHz}$ , 使能所有的IP , 禁用PLL, XTAL=4 MHz
	$I_{IDLE10}$	4		mA	$V_{DD} = 5.5V@4\text{ MHz}$ , 禁用所有的IP和PLL, XTAL=4 MHz
	$I_{IDLE11}$	4		mA	$V_{DD} = 3V@4\text{ MHz}$ ,

						使能所有的IP, 禁用PLL, XTAL=4 MHz
	$I_{DLE12}$		3		mA	$V_{DD} = 3V@4\text{ MHz}$ , 禁用所有的IP和PLL, XTAL=4 MHz
掉电模式下旁路电流 (深度休眠模式)	$I_{PWD1}$		15		$\mu\text{A}$	$V_{DD} = 5.5V$ , 无负载 @禁用BOV 功能
	$I_{PWD2}$		11		$\mu\text{A}$	$V_{DD} = 3.0V$ , 无负载 @禁用BOV 功能
P0/1/2/3/4输入电流 (准双向模式)	$I_{IN1}$		-50	-60	$\mu\text{A}$	$V_{DD} = 5.5V, V_{IN} = 0.4V$
P0/1/2/3/4输入漏电流	$I_{LK}$	-2	-	+2	$\mu\text{A}$	$V_{DD} = 5.5V, 0 < V_{IN} < V_{DD}$
P0/1/2/3/4逻辑1至0转换时电流(准双向模式)	$I_{TL}^{[3]}$	-650	-	-200	$\mu\text{A}$	$V_{DD} = 5.5V, V_{IN} < 2.0V$
P0/1/2/3/4输入低电压 (TTL输入)	$V_{IL1}$	-0.3	-	0.8	V	$V_{DD} = 4.5V$
		-0.3	-	0.6		$V_{DD} = 2.5V$
P0/1/2/3/4输入高电压 (TTL输入)	$V_{IH1}$	2.0	-	$V_{DD} + 0.2$	V	$V_{DD} = 5.5V$
		1.5	-	$V_{DD} + 0.2$		$V_{DD} = 3.0V$
输入低电压XT1 <sup>[2]</sup>	$V_{IL3}$	0	-	0.8	V	$V_{DD} = 4.5V$
		0	-	0.4		$V_{DD} = 3.0V$
输入高电压XT1 <sup>[2]</sup>	$V_{IH3}$	3.5	-	$V_{DD} + 0.2$	V	$V_{DD} = 5.5V$
		2.4	-	$V_{DD} + 0.2$		$V_{DD} = 3.0V$
RESET脚 负向门槛电压 (Schmitt输入)	$V_{ILS}$	-0.5	-	$0.3V_{DD}$	V	
RESET脚 正向门槛电压 (Schmitt输入)	$V_{IHS}$	$0.7V_{DD}$	-	$V_{DD} + 0.5$	V	
/RST 内部上拉电阻	$R_{RST}$	40		150	K $\Omega$	
P0/1/2/3/4 负向门槛电压 (Schmitt输入)	$V_{ILS}$	-0.5	-	$0.2V_{DD}$	V	
P0/1/2/3/4 正向门槛电压 (Schmitt输入)	$V_{IHS}$	$0.4V_{DD}$	-	$V_{DD} + 0.5$	V	
P0/1/2/3/4 源电流(准双向模式)	$I_{SR11}$	-300	-370	-450	$\mu\text{A}$	$V_{DD} = 4.5V, V_S = 2.4V$
	$I_{SR12}$	-50	-70	-90	$\mu\text{A}$	$V_{DD} = 2.7V, V_S = 2.2V$
	$I_{SR12}$	-40	-60	-80	$\mu\text{A}$	$V_{DD} = 2.5V, V_S = 2.0V$
P0/1/2/3/4源电流(推挽模式)	$I_{SR21}$	-20	-24	-28	mA	$V_{DD} = 4.5V, V_S = 2.4V$
	$I_{SR22}$	-4	-6	-8	mA	$V_{DD} = 2.7V, V_S = 2.2V$
	$I_{SR22}$	-3	-5	-7	mA	$V_{DD} = 2.5V, V_S = 2.0V$
P0/1/2/3/4灌电流(准双向模式)	$I_{SK1}$	10	16	20	mA	$V_{DD} = 4.5V, V_S = 0.45V$



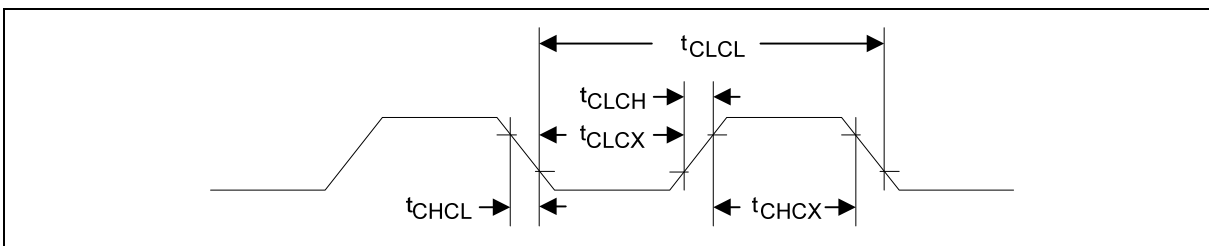
和推挽模式)	$I_{SK1}$	7	10	13	mA	$V_{DD} = 2.7V, V_S = 0.45V$
	$I_{SK1}$	6	9	12	mA	$V_{DD} = 2.5V, V_S = 0.45V$
欠压电压 BOV_VL [1:0] =00b	$V_{BO2.2}$	2.1	2.2	2.3	V	
欠压电压 BOV_VL [1:0] =01b	$V_{BO2.7}$	2.6	2.7	2.8	V	
欠压电压 BOV_VL [1:0] =10b	$V_{BO3.8}$	3.7	3.8	3.9	V	
欠压电压 BOV_VL [1:0] =11b	$V_{BO4.5}$	4.4	4.5	4.6	V	
BOD电压迟滞范围	$V_{BH}$	30	-	150	mV	$V_{DD} = 2.5V \sim 5.5V$

**Notes:**

1. /RST 脚为史密特触发输入.
2. XTAL1 为CMOS输入.
3. P0, P1, P2, P3 和 P4管脚被外部由1驱动到0时, 可作来输出电流的源端, 在 $V_{DD}=5.5V$ 时, 输出电流达到最大值,  $V_{in}$  接近2V.

## 9.3 AC 电气特性

### 9.3.1 外部晶振



注: 占空比为 50%.

参数	符号	最小值	典型值	最大值	单位	条件
时钟高电平时间	$t_{CHCX}$	20	-	125	nS	
时钟低电平时间	$t_{CLCX}$	20	-	125	nS	
时钟上升沿时间	$t_{CLCH}$	-	-	10	nS	
时钟下降沿时间	$t_{CHCL}$	-	-	10	nS	

### 9.3.2 外部振荡器

参数	条件	最小值	典型值	最大值	单位
输入时钟频率	外部晶振	4	12	24	MHz
温度	-	-40	-	85	°C
$V_{DD}$	-	2.5	5	5.5	V
工作电流	12 MHz@ $V_{DD} = 5V$	-	5	-	mA



### 9.3.3 晶振的典型应用电路

晶振	C1	C2
4 MHz ~ 24 MHz	可选 (取决于晶振规格)	

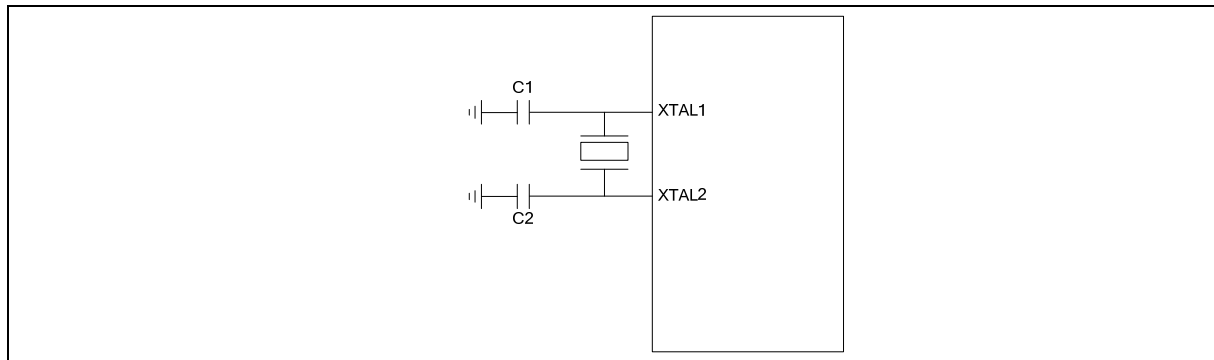


图 9.3-1 典型晶振应用电路

## 9.3.4 内部 22.1184 MHz RC振荡器

参数	条件	最小值	典型值	最大值	单位
电压 <sup>[1]</sup>	-	2.5	-	5.5	V
中心频率	-	-	22.1184	-	MHz
经过校准标准 内部频率偏差率	+25° C; V <sub>DD</sub> =5V	-1	-	+1	%
	-40° C ~+85° C; V <sub>DD</sub> =2.5V~5.5V	-3	-	+3	%
未经校准标准 内部频率偏差率	-40° C ~+85° C; V <sub>DD</sub> =2.5V~5.5V	-25	-	+25	%
工作电流	V <sub>DD</sub> =5V	-	500	-	uA

## 9.3.5 内部 10kHz RC 振荡器

参数	条件	最小值	典型值	最大值	单位
电压 <sup>[1]</sup>	-	2.5	-	5.5	V
中心频率	-	-	10	-	kHz
经过校准标准 内部频率偏差率	+25° C; V <sub>DD</sub> =5V	-30	-	+30	%
	-40° C ~+85° C; V <sub>DD</sub> =2.5V~5.5V	-50	-	+50	%
工作电流	V <sub>DD</sub> =5V	-	5	-	uA

注:

- 内部的工作电压来自LDO.

## 9.4 模拟量特性

## 9.4.1 600kHz sps 12-bit SARADC特性

参数	符号	最小值	典型值	最大值	单位
分辨率	-	-	-	12	Bit
非线性差分错误	DNL	-	±1.2	-	LSB
非线性整型错误	INL	-	±1.5	-	LSB
补偿错误	EO	-	+4	10	LSB
增益错误 (传输增益)	EG	-	+7	1.005	-
一致性	-	Guaranteed			-
ADC 时钟频率	FADC	-	-	20	MHz
校准时间	TCAL	-	127	-	Clock
取样时间	TS	-	7	-	Clock
转换时间	TADC	-	13	-	Clock
采样率	FS	-	-	600	k sps
工作电压	V <sub>LDO</sub>	-	2.5	-	V
	V <sub>ADD</sub>	3	-	5.5	V
工作电流(平均)	I <sub>DD</sub>	-	0.5	-	mA
	I <sub>DDA</sub>	-	1.5	-	mA
输入电压范围	V <sub>IN</sub>	0	-	V <sub>DD</sub>	V
电容	C <sub>IN</sub>	-	5	-	pF

## 9.4.2 LDO规格 &amp; Power 管理

参数	最小值	典型值	最大值	单位	备注
输入电压	2.7	5	5.5	V	V <sub>DD</sub> 输入电压
输出电压 (bypass=0)	-10%	2.45	+10%	V	LDO输出电压
输出电压 (bypass=1)	-10%	输入电压	+10%	V	输入电压 < 2.7V
静态电流 (PD=0, bypass=0)	-	100	-	uA	
静态电流 (PD=1, bypass=0)	-	5	-	uA	
静态电流 (PD=1, bypass=1)	-	5	-	uA	
Iload (PD=0)	-	-	100	mA	
Iload (PD=1)	-	-	100	uA	
Cbp	-	1u	-	F	Resr=1ohm
负载	-	250p	-	F	

注:

- 1、建议接一颗10uF 或更大的电容和一颗 100nF 旁路电容在VDD引脚与最近的VSS引脚之间.
- 2、为保证电源稳定, 要在LDO与最近的VSS之间接一颗4.7uF 或更大的电容. 再加一颗100nF 的旁路电容在LDO与VSS之间有助于抑制输出噪声Note.

## 9.4.3 低压复位说明

参数	条件	最小值	典型值	最大值	单位
操作电压	-	1.7	-	5.5	V
静态电流	VDD5V=5.5V	-	-	5	uA
极限电压	温度=25°C	1.7	2.0	2.3	V
	温度=-40°C	-	2.4	-	V
	温度=85°C	-	1.6	-	V
迟滞	-	0	0	0	V

## 9.4.4 欠压检测说明

参数	条件	最小值	典型值	最大值	单位
操作电压	-	2.5	-	5.5	V
静态电流	AVDD=5.5V	-	-	125	μA
温度	-	-40	25	85	°C
欠压电压	BOV_VL[1:0]=11	4.4	4.5	4.6	V
	BOV_VL [1:0]=10	3.7	3.8	3.9	V
	BOV_VL [1:0]=01	2.6	2.7	2.8	V
	BOV_VL [1:0]=00	2.1	2.2	2.3	V
迟滞	-	30m	-	150m	V

## 9.4.5 上电复位说明(5V)

参数	条件	最小值	典型值	最大值	单位
复位电压	V+	-	2	-	V
静态电流	Vin>复位电压	-	1	-	nA

## 9.5 SPI 动态特性

符号	参数	最小值	典型值	最大值	单位
SPI 主机模式 (VDD = 4.5V ~ 5.5V, 30pF 负载电容)					
$t_{DS}$	数据建立时间	35	-	-	ns
$t_{DH}$	数据保持时间	0	-	-	ns
$t_V$	数据输出有效时间	-	-	10	ns
SPI 主机模式 (VDD = 3.0V ~ 3.6V, 30pF 负载电容)					
$t_{DS}$	数据建立时间	45	-	-	ns
$t_{DH}$	数据保持时间	0	-	-	ns
$t_V$	数据输出有效时间	-	-	16	ns
SPI 从机模式 (VDD = 4.5V ~ 5.5V, 30pF 负载电容)					
$t_{DS}$	数据建立时间	0	-	-	ns
$t_{DH}$	数据保持时间	$2 \cdot PCLK + 4$	-	-	ns
$t_V$	数据输出有效时间	-	-	$2 \cdot PCLK + 40$	ns
SPI 从机模式 (VDD = 3.0V ~ 3.6V, 30pF 负载电容)					
$t_{DS}$	数据建立时间	0	-	-	ns
$t_{DH}$	数据保持时间	$2 \cdot PCLK + 5$	-	-	ns
$t_V$	数据输出有效时间	-	-	$2 \cdot PCLK + 50$	ns

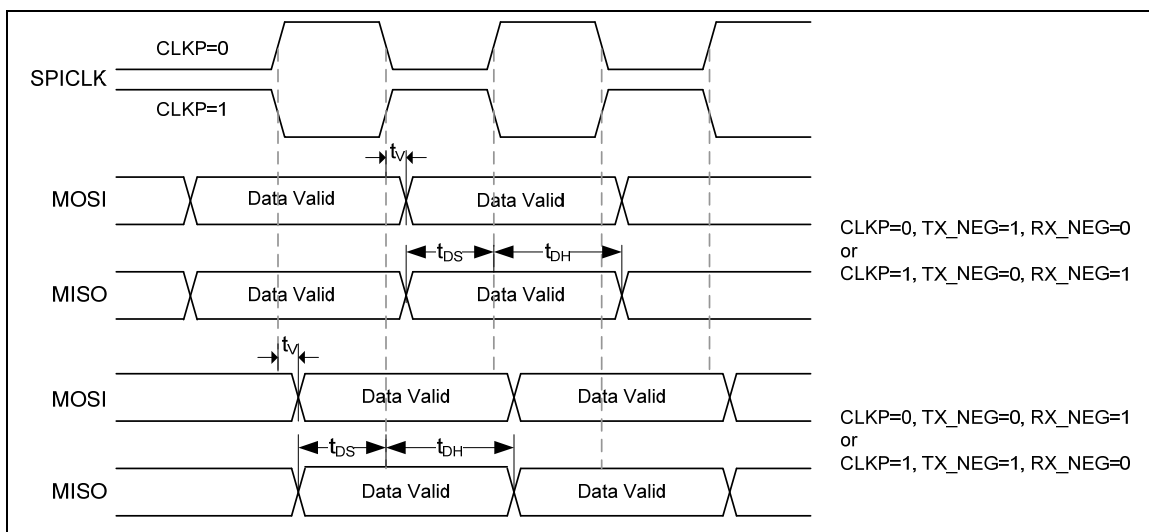


图 9.5-1 SPI 主机时序

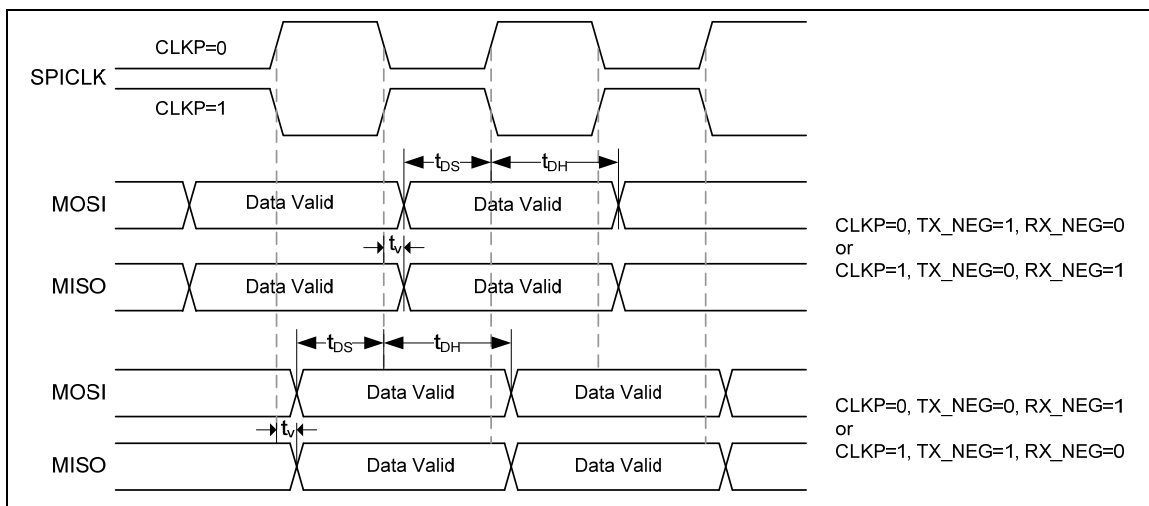
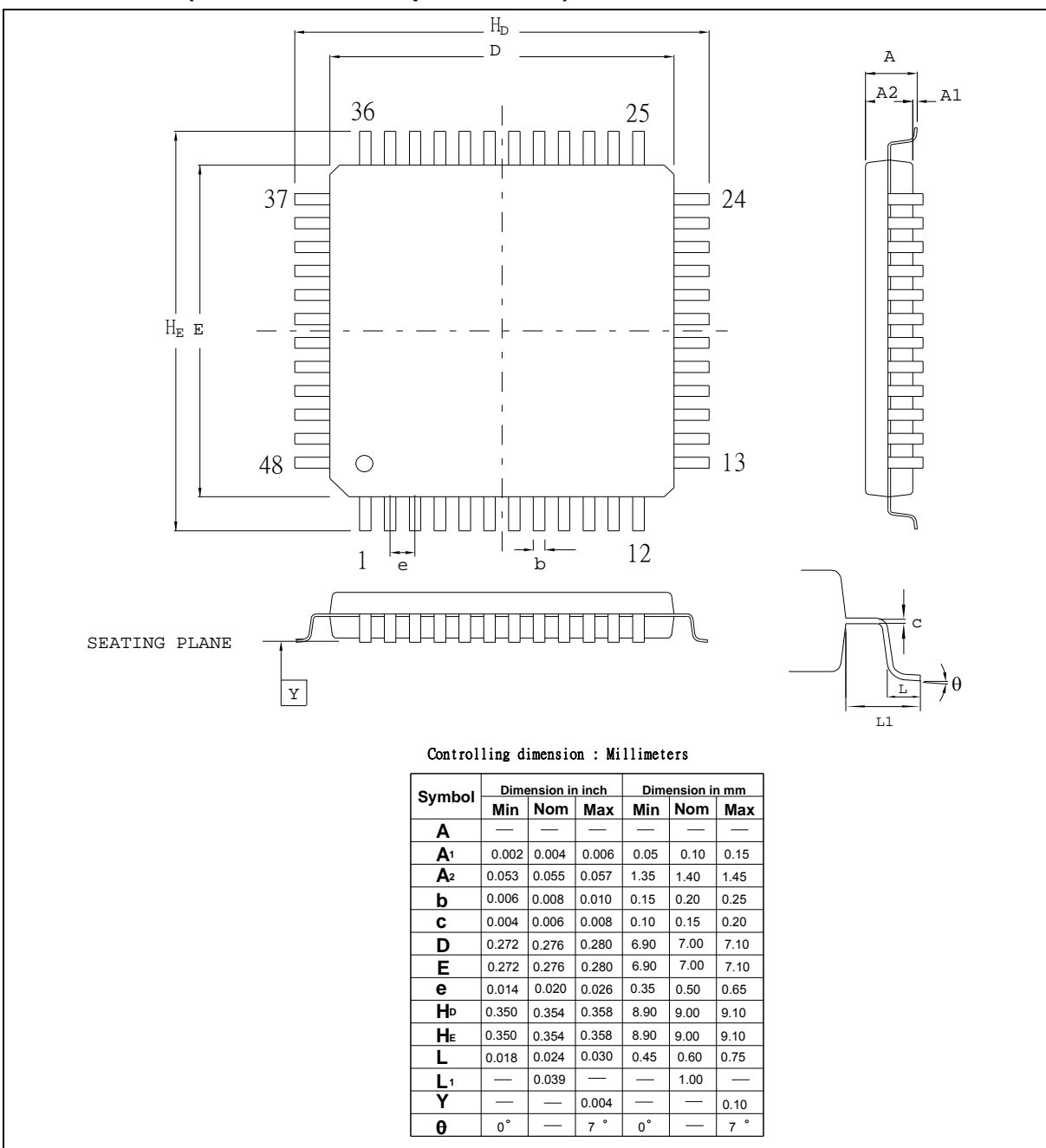


图 9.5-2 SPI 从机时序

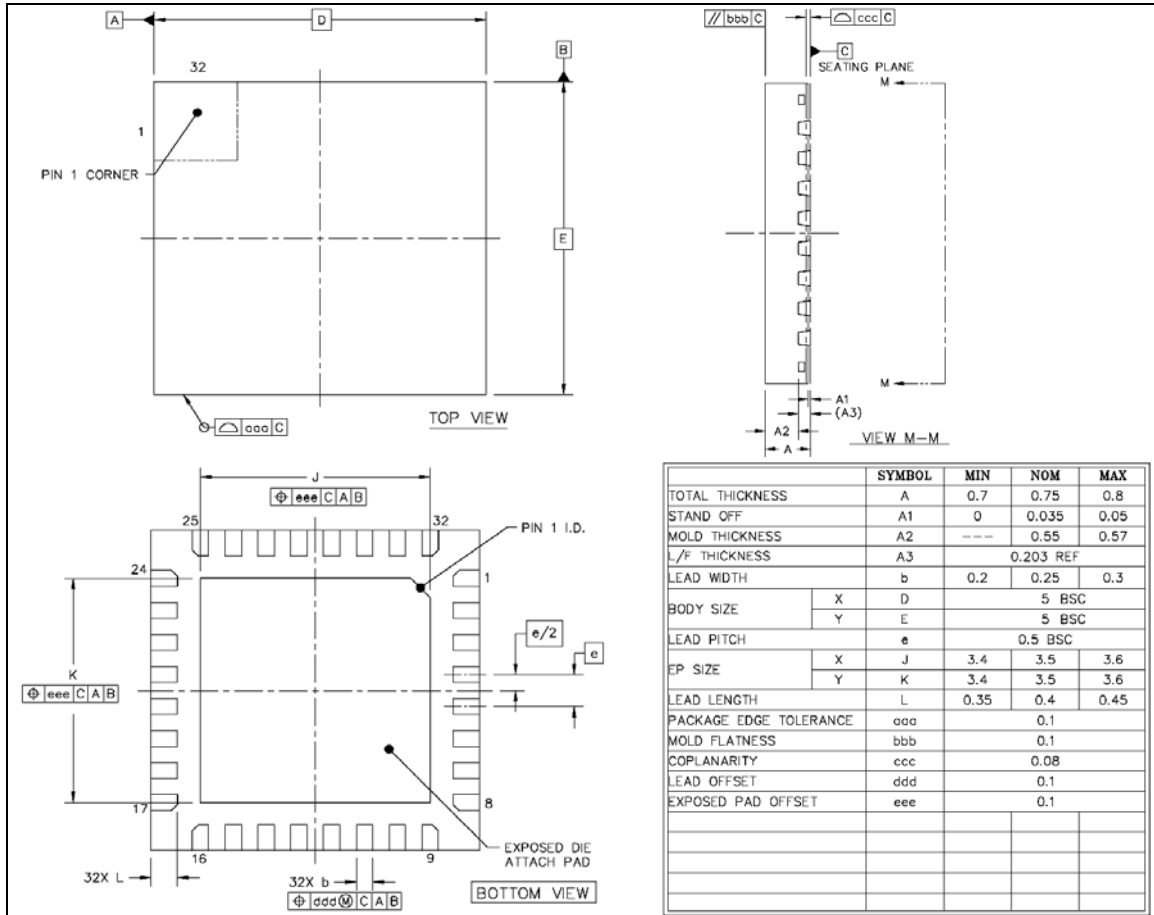
## 10 封装尺寸

### 10.1 LQFP-48 (7x7x1.4mm<sup>2</sup> Footprint 2.0mm)





## 10.2 QFN-33 (5X5 mm<sup>2</sup>, Thickness 0.8mm, Pitch 0.5 mm)



## 11 版本历史

版本	日期	页	描述
V1.0	Aug 23, 2010	-	初次发行版本
V1.1	Sep. 8, 2010	6.10	1. 修改 UART的FIFO大小（从16bytes 到 15bytes）。 2. 校正UART中断的总线错误。
V1.2	Sep. 14, 2010	6.10	1. 校正CTS触发电平的定义 (UA_MSR.8).

## Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Further more, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

---

Please note that all data and specifications are subject to change without notice. All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.

---

*Please note that all data and specifications are subject to change without notice.  
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*