ARM Cortex™-M0

32-BIT MICROCONTROLLER

NuMicro™ Family
Mini51 系列
技术参考手册

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

目录

List of Figures

List of Tables

1 概述

NuMicro MINI51™ 系列是32位的微处理器，内嵌ARM® Cortex™-M0内核，可用于工业控制和需要高性能、低功耗的应用. Cortex™-M0是ARM最新的微处理器，有32位的性能，但是价格只相当于传统的8位单片机.

NuMicro MINI51™系列最快可以跑到24MHz. 因而可以支持很广范围的工业控制和需要高性能CPU的场合. NuMicro MINI51™ 系列内嵌4K/8K/16K字节程序flash，数据flash大小可配置(与程序flash共享) ,2K字节ISP flash，2K字节SRAM.

为了降低成本，减小空间，NuMicro MINI51™ 系列内嵌了很多外设，像：I/O口、定时器、UART、SPI、I2C、PWM、ADC、看门狗和低电压检测，这使NuMicro MINI51™ 系列可以用于更广泛的应用.

另外，NuMicro MINI51™ 系列还配备ISP (In-System Programming) 和 ICP (In-Circuit Programming) 功能，让用户可以升级固件而不必将芯片从板子上取下.

2 特性

- 内核
  - ARM® Cortex™-M0 核，最高跑到 24 MHz
  - 一个 24比特系统定时器
  - 支持低功耗Idle 模式
  - 一个单指令周期硬件乘法器
  - 支持32个外部中断的NVIC, 每个中断有4级优先级
  - 支持串行调试接口 (SWD)，有2个监视点(watchpoints)/4个断点(breakpoints)
- 内嵌LDO 可支持宽电压输入: 2.5 V to 5.5 V
- 内存
  - 4KB/8KB/16KB Flash 内存用来存放应用程序 (APROM)
  - 可配置的数据 flash(Data Flash)
  - 2KB启动代码空间 (LDROM)
  - 内嵌2KB SRAM (SRAM)
- 支持In-System Programming (ISP) & In-Circuit Programming (ICP)
- 时钟控制
  - 系统时钟源可编程
    - 正在运行代码时可以切换时钟源
  - 4 ~ 24 MHz crystal oscillator (HXT)
  - 32.768K crystal oscillator (LXT)，可用于系统时钟和在掉电模式(power down mode)下唤醒CPU (如果外设选择32.768K作为时钟源的话)
  - 22.1184 MHz 内部 oscillator (HIRC) (25^{0}C, 5V,1% 误差)
    - 在-40^{0}C to 85^{0}C，利用外部32.768K晶振可以动态矫正到22 MHz +/- 1%
  - 10 KHz 内部低功耗oscillator (LIRC)，给看门狗和掉电模式下唤醒CPU提供时钟源(如果外设选择10K作为时钟源的话)
- I/O口
  - LQFP-48封装，最多 30个通用 (GPIO) 脚
  - 软件可以配置I/O口为以下模式
    - 准双向输入/输出模式
    - 推挽输出
    - 开漏输出
    - 输入模式，带内部高阻
  - 可选择施密特触发输入模式

- 定时器
  - 两个24-bit 定时器，有8-bit预分频
    - 支持 事件计数功能
    - 支持toggle输出模式
    - 脉冲宽度测量模式下，支持外部触发
      - 脉冲宽度捕获模式下，支持外部触发
- 看门狗定时器
  - 时钟源和超时周期可选择
  - 掉电和idle模式下支持唤醒CPU功能
  - 当超时发生时，可以选择发生中断还是复位CPU
- PWM
  - 内嵌最多3个16位PWM发生器，提供6个独立的PWM输出或者3组互补的PWM输出
  - 支持边沿对齐和中心对齐
  - 支持故障侦测
  - 每个PWM发生器有单独的时钟源，时钟除频，8比特预分频和死区发生器
  - 每个PWM周期可以发生中断
- UART
  - 一组 UART
  - 两个16字节的接收和发送缓冲区
  - 流控功能(CTSn 和 RTSn)
  - 支持 IrDA (SIR) 功能
  - 波特率可编程，最快可达 1/16 系统时钟
  - 支持 RS-485 功能
- SPI
  - 一组SPI
  - 主模式最高可达12 MHz, 从模式最高可达4 MHz
  - 支持SPI主/从模式
  - 全双工同步串行数据传输
  - 每笔传输比特长度可配置，范围 1到 32 比特
  - MSB 或者 LSB优先
  - 发送和 接收边沿独立，都可以上升沿也可以下降沿
  - 32比特长度下，支持字节suspend 功能

- I^2C
  - 支持主/从模式
  - 主和从之间双向数据传输
  - 支持多主总线 (无核心主设备)
  - 同时发起传输的主设备之间仲裁，防止数据被破坏
  - 串行时钟同步，允许同一个总线上的设备有不同的比特率
  - 串行时钟同步可以用做一个握手机制，挂起或者重启串行传输
  - 时钟源可编程以方便波特率控制
  - 支持多地址识别(四个从地址，有掩码功能)
- ADC
  - 10-bit SAR型 ADC，速率 150K SPS
  - 最多8个single-end输入通道，一个内部band-gap输入
  - 可由软件或者外部引脚触发一次转换
- Analog Comparator
  - 2组模拟比较器。支持可编程的16级内部参考电压
  - 内嵌比较器参考电压(CRV)
- BOD R复位
  - 3种检测电压选择: 3.8V/2.7V/2.0V (缺省 2.0V)
  - BOD中断还是复位可选择
- 96比特唯一序列号 (Unique ID)
- 工作温度:-40℃~85℃
- 封装:
  - Green package (RoHS)
  - LQFP 48-pin (7x7), QFN 33-pin (5x5), QFN 33-pin (4x4)

3 产品型号和引脚配置

3.1 NuMicro Mini51™系列选型表

Part number	APROM	RAM	Data Flash	ISP Loader ROM	I/O	Timer	Connectivity			Comp.	PWM	ADC	ISP ICP	IRC 22.1184 MHz	Package
							UART	SPI	I^2C						
MINI51LAN	4 KB	2 KB	Configurable	2 KB	up to 30	2x32-bit	1	1	1	2	6	8x10-bit	v	v	LQFP48
MINI51ZAN	4 KB	2 KB	Configurable	2 KB	up to 29	2x32-bit	1	1	1	2	6	8x10-bit	v	v	QFN33(5x5)
MINI51TAN	4 KB	2 KB	Configurable	2 KB	up to 29	2x32-bit	1	1	1	2	6	8x10-bit	v	v	QFN33(4x4)
MINI52LAN	8 KB	2 KB	Configurable	2 KB	up to 30	2x32-bit	1	1	1	2	6	8x10-bit	v	v	LQFP48
MINI52ZAN	8 KB	2 KB	Configurable	2 KB	up to 29	2x32-bit	1	1	1	2	6	8x10-bit	v	v	QFN33(5x5)
MINI52TAN	8 KB	2 KB	Configurable	2 KB	up to 29	2x32-bit	1	1	1	2	6	8x10-bit	v	v	QFN33(4x4)
MINI54LAN	16 KB	2 KB	Configurable	2 KB	up to 30	2x32-bit	1	1	1	2	6	8x10-bit	v	v	LQFP48
MINI54ZAN	16 KB	2 KB	Configurable	2 KB	up to 29	2x32-bit	1	1	1	2	6	8x10-bit	v	v	QFN33(5x5)
MINI54TAN	16 KB	2 KB	Configurable	2 KB	up to 29	2x32-bit	1	1	1	2	6	8x10-bit	v	v	QFN33(4x4)

表 3.1-1 NuMicro Mini51™系列产品选型表

3.2 引脚配置

3.2.1 LQFP 48-pin

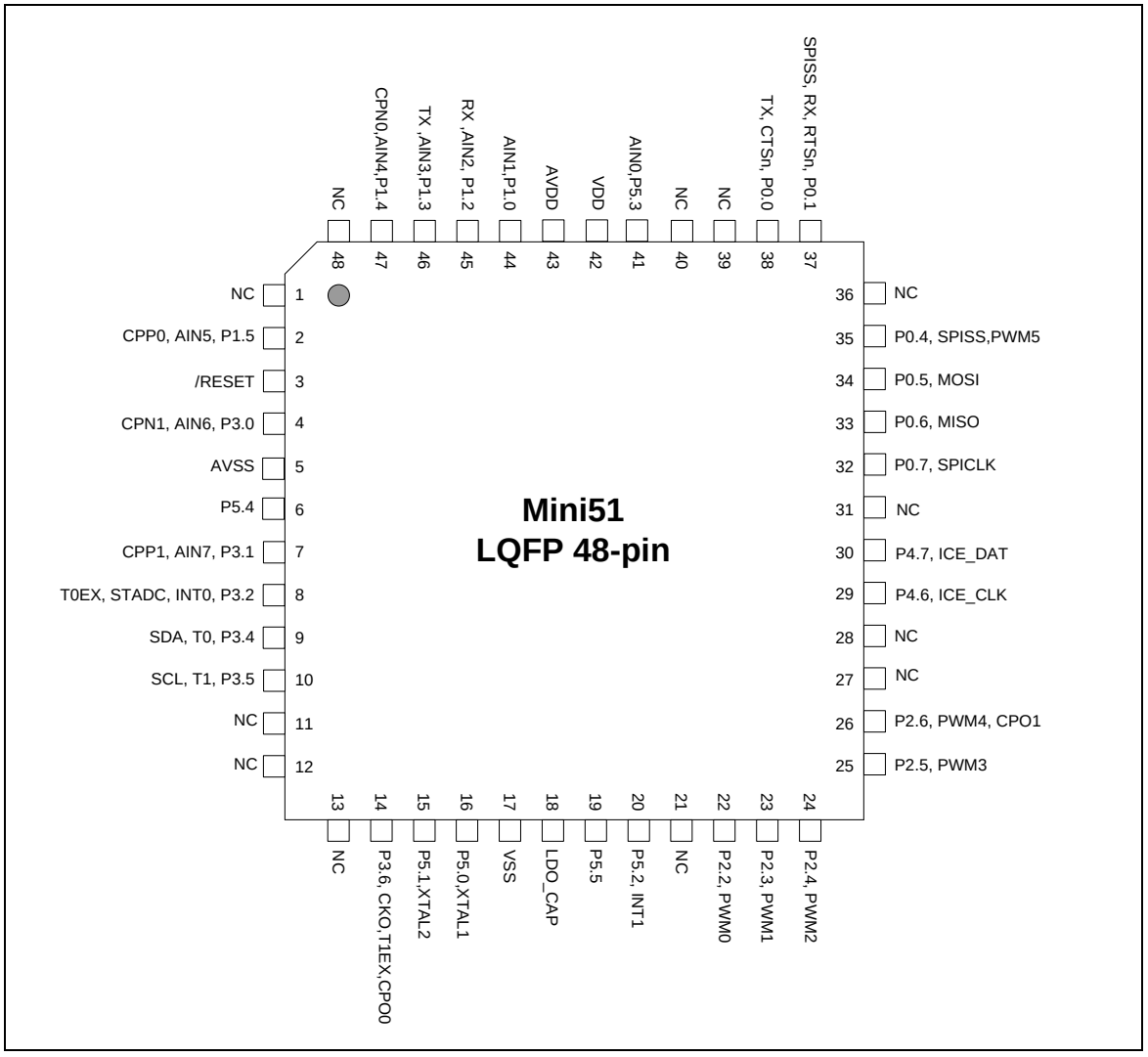


图 3.2-1 NuMicro Mini51™系列 LQFP 48-pin 图

3.2.2 QFN 33-pin

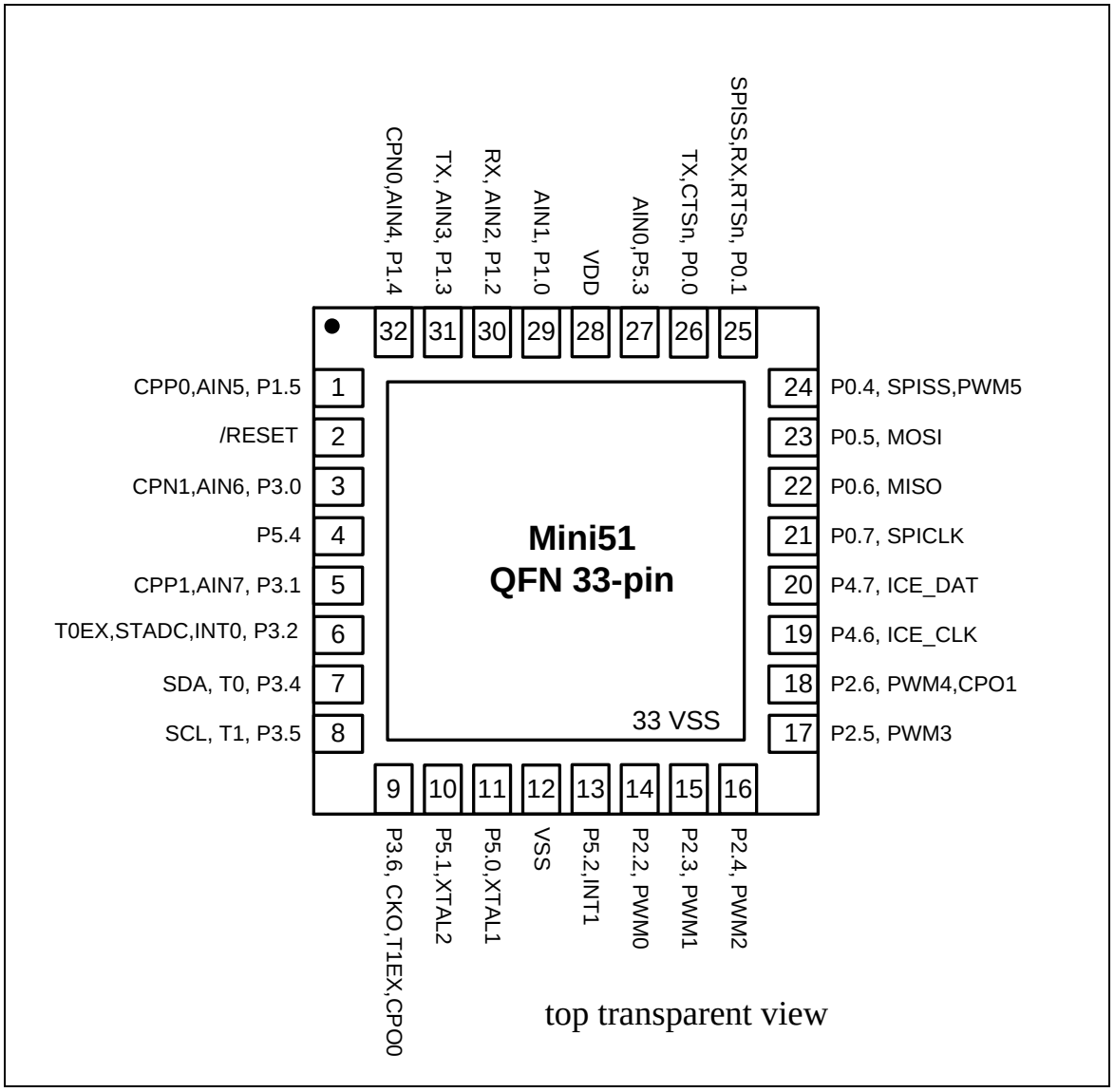


图 3.2-2 NuMicro Mini51™ 系列 QFN 33-pin 图

3.3 引脚描述

Pin Number		Pin Name	Pin Type	描述
LQFP 48	QFN 33			
1		NC		Not connected pin
2	1	P1.5	**I/O**	General purpose input/output digital pin
		AIN5	**AI**	ADC analog input pin
		CPP0	**AI**	Analog comparator Positive input pin
3	2	/RESET	**I(ST)**	This pin is a Schmitt trigger input pin for hardware device reset. A "**Low**" on this pin for 768 clock counter of Internal RC 22.1184 MHz while the system clock is running will reset the device. /RESET pin has an internal pull-up resistor allowing power-on reset by simply connecting an external capacitor to GND.
4	3	P3.0	**I/O**	General purpose input/output digital pin
		AIN6	**AI**	ADC analog input pin
		CPN1	**AI**	Analog comparator negative input pin
5		AVSS	**AP**	Ground pin for analog circuit
6	4	P5.4	**I/O**	General purpose input/output digital pin
7	5	P3.1	**I/O**	General purpose input/output digital pin
		AIN7	**AI**	ADC analog input pin
		CPP1	**AI**	Analog comparator positive input pin
8	6	P3.2	**I/O**	General purpose input/output digital pin
		INT0	**I**	External interrupt 0 input pin
		STADC	**I**	ADC external trigger input pin
		T0EX	**I**	Timer 0 external capture/reset trigger input pin
9	7	P3.4	**I/O**	General purpose input/output digital pin
		T0	**I/O**	Timer 0 external event counter input pin
		SDA	**I/O**	I^2C data input/output pin
10	8	P3.5	**I/O**	General purpose input/output digital pin
		T1	**I/O**	Timer 1 external event counter input pin
		SCL	**I/O**	I^2C clock input/output pin
11		NC		Not connected pin
12		NC		Not connected pin

Pin Number		Pin Name	Pin Type	描述
LQFP 48	QFN 33			
13		NC		Not connected pin
14	9	P3.6	**I/O**	General purpose input/output digital pin
		CPO0	**O**	Analog comparator output pin
		CKO	**O**	Frequency Divider output pin
		T1EX	**I**	Timer 1 external capture/reset trigger input pin
15	10	P5.1	**I/O**	General purpose input/output digital pin
		XTAL2	**O**	This is the output pin from the internal inverting amplifier. It emits the inverted signal of XTAL1.
16	11	P5.0	**I/O**	General purpose input/output digital pin
		XTAL1	**I**	This is the input pin to the internal inverting amplifier. The system clock could be from external crystal or resonator.
17	12	VSS	**P**	Ground pin for digital circuit
	33			
18		LDO_CAP	**P**	LDO output pin
19		P5.5	**I/O**	General purpose input/output digital pin User program must enable pull-up resistor in QFN33 package.
20	13	P5.2	**I/O**	General purpose input/output digital pin
		INT1	**I**	External interrupt 1 input pin
21		NC		Not connected pin
22	14	P2.2	**I/O**	General purpose input/output digital pin
		PWM0	**O**	PWM0 output of PWM unit
23	15	P2.3	**I/O**	General purpose input/output digital pin
		PWM1	**O**	PWM1 output of PWM unit
24	16	P2.4	**I/O**	General purpose input/output digital pin
		PWM2	**O**	PWM2 output of PWM unit
25	17	P2.5	**I/O**	General purpose input/output digital pin
		PWM3	**O**	PWM3 output of PWM unit
26	18	P2.6	**I/O**	General purpose input/output digital pin
		PWM4	**O**	PWM4 output of PWM unit
		CPO1	**O**	Analog comparator output pin

Pin Number		Pin Name	Pin Type	描述
LQFP 48	QFN 33			
27		NC		Not connected pin
28		NC		Not connected pin
29	19	P4.6	**I/O**	General purpose input/output digital pin
		ICE_CLK	**I**	Serial wired debugger clock pin
30	20	P4.7	**I/O**	General purpose input/output digital pin
		ICE_DAT	**I/O**	Serial wired debugger data pin
31		NC		Not connected pin
32	21	P0.7	**I/O**	General purpose input/output digital pin
		SPICLK	**I/O**	SPI serial clock pin
33	22	P0.6	**I/O**	General purpose input/output digital pin
		MISO	**I/O**	SPI MISO (master in/slave out) pin
34	23	P0.5	**I/O**	General purpose input/output digital pin
		MOSI	**O**	SPI MOSI (master out/slave in) pin
35	24	P0.4	**I/O**	General purpose input/output digital pin
		SPISS	**I/O**	SPI slave select pin
		PWM5	**O**	PWM5 output of PWM unit
36		NC		Not connected pin
37	25	P0.1	**I/O**	General purpose input/output digital pin
		RTSn	**O**	UART RTS pin
		RX	**I**	UART data receiver input pin
		SPISS	**I/O**	SPI slave select pin
38	26	P0.0	**I/O**	General purpose input/output digital pin
		CTSn	**I**	UART CTS pin
		TX	**O**	UART transmitter output pin
39		NC		Not connected pin
40		NC		Not connected pin
41	27	P5.3	**I/O**	General purpose input/output digital pin
		AIN0	**AI**	ADC analog input pin
42	28	VDD	**P**	Power supply for digital circuit

Pin Number		Pin Name	Pin Type	描述
LQFP 48	QFN 33			
43		AVDD	**P**	Power supply for analog circuit
44	29	P1.0	**I/O**	General purpose input/output digital pin
		AIN1	**AI**	ADC analog input pin
45	30	P1.2	**I/O**	General purpose input/output digital pin
		AIN2	**AI**	ADC analog input pin
		RX	**I**	UART data receiver input pin
46	31	P1.3	**I/O**	General purpose input/output digital pin
		AIN3	**AI**	ADC analog input pin
		TX	**O**	UART transmitter output pin
47	32	P1.4	**I/O**	General purpose input/output digital pin
		AIN4	**I/O**	PWM5: PWM output/Capture input
		CPN0	**AI**	Analog comparator negative input pin
48		NC		Not connected pin

表 3.3-1 NuMicro Mini51™系列引脚描述

[1] I/O 类型描述. I: 输入, O: 输出, I/O: 准双向, D: 开漏, P: 电源引脚, ST: Schmitt trigger, A: 模拟输入.

4 方块图

4.1 NuMicro Mini51™ 方块图

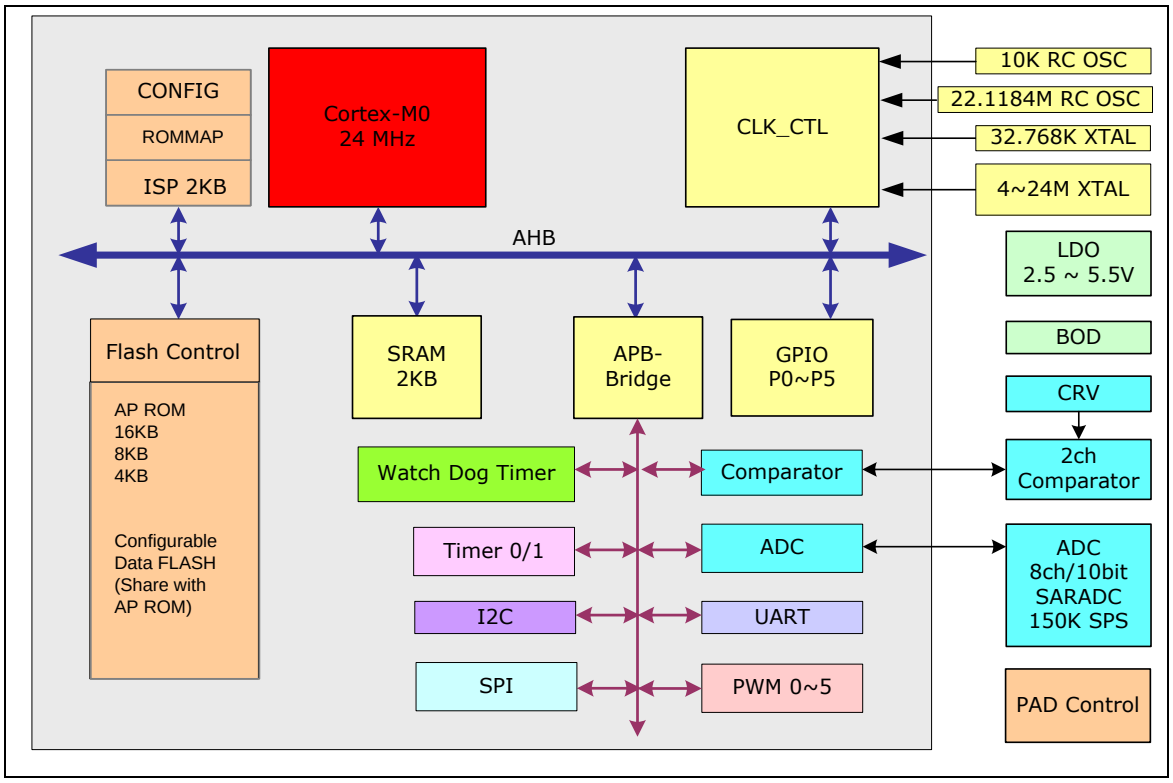


图 4.1-1 NuMicro Mini51™系列方块图

5 功能描述

5.1 内存组织

5.1.1 概述

NuMicro MINI51™ 系列有4G字节的地址空间. 芯片上每个模块在内存的位置如表5.1-1所示. 详细的寄存器描述、地址以及编程将在下面每个模块的章节单独叙述. NuMicro MINI51™ 系列只支持小端数据格式..

5.1.2 系统内存映射

芯片上每个控制器的内存地址安排如下表所示.

地址空间	符号	控制器
Flash & SRAM内存空间		
0x0000_0000 – 0x0000_3FFF	FLASH_BA	FLASH内存空间(16KB)
0x2000_0000 – 0x2000_07FF	SRAM_BA	SRAM 内存空间 (2KB)
AHB控制器空间(0x5000_0000 – 0x501F_FFFF)		
0x5000_0000 – 0x5000_01FF	GCR_BA	系统管理控制寄存器
0x5000_0200 – 0x5000_02FF	CLK_BA	时钟控制寄存器
0x5000_0300 – 0x5000_03FF	INT_BA	中断复用控制寄存器
0x5000_4000 – 0x5000_7FFF	GP_BA	GPIO控制寄存器
0x5000_C000 – 0x5000_FFFF	FMC_BA	Flash 内存控制寄存器
APB1 控制器空间(0x4000_0000 – 0x401F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	看门狗控制寄存器
0x4001_0000 – 0x4001_3FFF	TMR_BA	定时器0/定时器1 控制寄存器
0x4002_0000 – 0x4002_3FFF	I2C_BA	I^2C控制寄存器
0x4003_0000 – 0x4003_3FFF	SPI_BA	SPI控制寄存器
0x4004_0000 – 0x4004_3FFF	PWM_BA	PWM控制寄存器
0x4005_0000 – 0x4005_3FFF	UART_BA	UART控制寄存器
0x400D_0000 – 0x400D_3FFF	CMP_BA	模拟比较器控制寄存器
0x400E_0000 – 0x400E_3FFF	ADC_BA	模数转换 (ADC) 控制寄存器
系统控制(SCS)空间(0xE000_E000 – 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SCS_BA	系统定时器(SysTick)控制寄存器
0xE000_E100 – 0xE000_ECFF	SCS_BA	Nested Vectored Interrupt Control Registers(外部中断控制寄存器)
0xE000_ED00 – 0xE000_ED8F	SCB_BA	系统控制寄存器

表 5.1-1 片上各模块地址空间分配

5.2 Nested Vectored Interrupt Controller (NVIC)

5.2.1 概述

Cortex™-M0 CPU 内核提供一个中断控制器作为异常模型的完整部分, 命名为“递归向量中断控制器(NVIC)”.

5.2.2 特性

- 支持递归和向量中断
- 自动处理器状态保存和恢复
- 动态优先级改变
- 降低中断延迟，并且延迟时间确定

NVIC 支持有4级优先级.所有异常都是在”Handler Mode”下被处理的.NVIC支持32个(IRQ[31:0]) 外部中断输入.所有中断和大多数系统异常可以配置优先级.当中断发生时,NVIC将比较新来的中断和正在处理的中断的优先级.如果新来的中断优先级比较高，新来的中断将抢占正在处理的中断.

发生中断时,中断处理函数(ISR) 的起始地址将从内存中向量表取得.不需要由软件查看发生了何种中断再决定要跳去哪个地址. 拿到起始地址之后,NVIC将自动保存上下文，包括这些寄存器“PC, PSR, LR, R0~R3, R12” 到堆栈. 中断处理函数结束之后, NVIC 将从堆栈自动恢复上下文，然后继续正常运行. 因此它将花费更少并且确定的时间来处理中断请求.

NVIC 支持 “Tail Chaining” 的方式处理中断，一个中断处理完毕不要恢复现场马上处理另一个，这样可以降低中断等待时间，使中断处理更加有效率. NVIC 也支持 “Late Arrival” 的中断处理方式.如果当前中断正在保存上下文还没有进入中断处理函数，一个更高优先级的中断发生，NVIC将处理更高优先级的中断，并且不用再保存一次上下文，因而可以提高实时性.

更多细节，请参考文档 “ARM® Cortex™-M0 Technical Reference Manual” 和 “ARM® v6-M Architecture Reference Manual”.

5.2.3 异常模型和系统中断映射

下表列出了NuMicro MINI51™ 系列支持的异常模型. 一些异常和所有中断可以设定4级优先级. “0”为最高优先级,”3”为最低优先级.所有用户可以设定的中断，缺省的优先级是”0”. 注意优先级”0”的优先级，在整个系统中是在“复位”, “NMI” 和 “Hard Fault”之后的第4级优先级.

Exception Name	异常号	优先级
Reset	1	-3
NMI	2	-2

Exception Name	异常号	优先级
Hard Fault	3	-1
预留	4 ~ 10	预留
SVCall	11	可配置
预留	12 ~ 13	预留
PendSV	14	可配置
SysTick	15	可配置
Interrupt (IRQ0 ~ IRQ31)	16 ~ 47	可配置

表 5.2-1 异常模型

Exception Number	IRQ Number (Bit in Interrupt Registers)	Exception Name	Source IP	中断描述	Power Down wake-up
1 ~ 15	-	-	-	System exceptions	-
16	0	**BOD_OUT**	Brownout	Brownout low voltage detected interrupt	Yes
17	1	**WDT_INT**	WDT	Watchdog Timer interrupt	Yes
18	2	**EINT0**	GPIO	External signal interrupt from P3.2 pin	Yes
19	3	**EINT1**	GPIO	External signal interrupt from P5.2 pin	Yes
20	4	**GP0/1_INT**	GPIO	External signal interrupt from GPIO group P0~P1	Yes
21	5	**GP2/3/4_INT**	GPIO	External signal interrupt from GPIO group P2~P4 except P3.2	Yes
22	6	**PWM_INT**	PWM	PWM interrupt	No
23	7	**BRAKE_INT**	PWM	PWM interrupt	No
24	8	**TMR0_INT**	TMR0	Timer 0 interrupt	Yes
25	9	**TMR1_INT**	TMR1	Timer 1 interrupt	Yes
26 ~ 27	10 ~ 11	-	-	-	
28	12	**UART_INT**	UART	UART interrupt	Yes
29	13	-	-	-	
30	14	**SPI_INT**	SPI	SPI interrupt	No
31	15	-	-	-	

Exception Number	IRQ Number (Bit in Interrupt Registers)	Exception Name	Source IP	中断描述	Power Down wake-up
32	16	**GP5_INT**	GPIO	External signal interrupt from GPIO group P5 except P5.2	Yes
33	17	**HFIRC_TRIM_INT**	HFIRC	HFIRC trim interrupt	No
34	18	**I2C_INT**	I^2C	I^2C interrupt	No
35 ~ 40	19 ~ 24	**-**	-	-	
41	25	**ACMP_INT**	ACMP	Analog Comparator 0 or 1 interrupt	Yes
42 ~ 43	26 ~ 27	**-**	-	-	
44	28	**PWRWU_INT**	CLKC	Clock controller interrupt for chip wake-up from power-down state	Yes
45	29	**ADC_INT**	ADC	ADC interrupt	No
46 ~ 47	30 ~ 31	**-**	-	-	

表 5.2-2 系统中断映射

5.2.4 向量表

收到中断信号时,处理器将自动从中断向量表取得中断处理函数的起始地址.对于ARMv6-M 来说,向量表的基地址固定在地址0x0000_0000的位置. 向量表包含复位时堆栈的初始值和所有异常处理函数的入口地址. 前一页的向量号定义了相应异常处理函数在向量表入口地址的顺序.

向量表字(Word) 偏移	描述
0x00	Initial Stack Pointer value
Exception Number × 0x04	Exception Entry Pointer using that Exception Number

表 5.2-3 向量表格式

5.2.5 操作描述

通过写中断Set-Enable 或者中断Clear-Enable 寄存器相应的比特域 ,NVIC可以使能和禁止中断. 寄存器使用写-1-使能和写-1-禁止的机制, 两个寄存器都可以读回当前相应中断的使能状态. 当中断被禁止时，发起中断将导致中断变成待处理状态，然而，中断不会激活.如果在中断禁止时中断是激活的，它将保持激活，直到复位或者异常返回. 清除使能位将阻止相应中断新的激活.

使用Set-Pending 寄存器和Clear-Pending 寄存器，NVIC中断可以是pended/un-pended状态. 这两个寄存器使用写-1-使能和写-1-清除机制, 两个寄存器都可以读回相应中断的当前pended状态. 激活的中断，写Clear-Pending 寄存器无效.

通过设定32比特寄存器(每个寄存器支持4个中断)中的8个比特可以设定一个NVIC 中断的优先级.

所有的NVIC寄存器都在系统控制空间(SCS)可以访问到，细节在下一节描述.

5.2.6 NVIC 控制寄存器

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移	R/W	描述	复位值
SCS_BA = 0xE000_E000				
NVIC_ISER	SCS_BA+0x100	R/W	IRQ0 ~ IRQ31 Set-Enable控制寄存器	0x0000_0000
NVIC_ICER	SCS_BA+0x180	R/W	IRQ0 ~ IRQ31 Clear-Enable控制寄存器	0x0000_0000
NVIC_ISPR	SCS_BA+0x200	R/W	IRQ0 ~ IRQ31 Set-Pending控制寄存器	0x0000_0000
NVIC_ICPR	SCS_BA+0x280	R/W	IRQ0 ~ IRQ31 Clear-Pending控制寄存器	0x0000_0000
NVIC_IPR0	SCS_BA+0x400	R/W	IRQ0 ~ IRQ3 优先级控制寄存器	0x0000_0000
NVIC_IPR1	SCS_BA+0x404	R/W	IRQ4 ~ IRQ7 优先级控制寄存器	0x0000_0000
NVIC_IPR2	SCS_BA+0x408	R/W	IRQ8 ~ IRQ11 优先级控制寄存器	0x0000_0000
NVIC_IPR3	SCS_BA+0x40C	R/W	IRQ12 ~ IRQ15 优先级控制寄存器	0x0000_0000
NVIC_IPR4	SCS_BA+0x410	R/W	IRQ16 ~ IRQ19 优先级控制寄存器	0x0000_0000
NVIC_IPR5	SCS_BA+0x414	R/W	IRQ20 ~ IRQ23 优先级控制寄存器	0x0000_0000
NVIC_IPR6	SCS_BA+0x418	R/W	IRQ24 ~ IRQ2 优先级控制寄存器	0x0000_0000
NVIC_IPR7	SCS_BA+0x41C	R/W	IRQ28 ~ IRQ31 优先级控制寄存器	0x0000_0000

IRQ0 ~ IRQ31 Set-Enable控制寄存器(NVIC_ISER)

寄存器	偏移	R/W	描述	复位值
NVIC_ISER	SCS _BA+0x100	R/W	IRQ0 ~ IRQ31 Set-Enable控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	**SETENA[31:0]**	使能一个或者多个中断.每个比特代表一个中断号IRQ0 ~ IRQ31 (向量号16 ~ 47). 写 “1” 将使能相应的中断. 写 “0”没有作用. 这个寄存器可以读回当前中断的使能状态.

IRQ0 ~ IRQ31 Clear-Enable控制寄存器(NVIC_ICER)

寄存器	偏移	R/W	描述	复位值
NVIC_ICER	SCS _BA+0x180	R/W	IRQ0 ~ IRQ31 Clear-Enable控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CLRENA[31:0]	禁止一个或者多个中断.每个比特代表一个中断号IRQ0 ~ IRQ31 (向量号16 ~ 47). 写 “1” 将禁止相应的中断. 写 “0”没有作用. 这个寄存器可以读回当前中断的使能状态.

IRQ0 ~ IRQ31 Set-Pending 控制寄存器(NVIC_ISPR)

寄存器	偏移	R/W	描述	复位值
NVIC_ISPR	SCS _BA+0x200	R/W	IRQ0 ~ IRQ31 Set-Pending 控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	SETPEND[31:0]	软件控制写”1”将pend相应的中断.每个比特代表一个中断号IRQ0 ~ IRQ31 (向量号16 ~ 47). 写 “0”没有作用. 这个寄存器可以读回当前中断的pend状态.

IRQ0 ~ IRQ31 Clear-Pending 控制寄存器(NVIC_ICPR)

寄存器	偏移	R/W	描述	复位值
NVIC_ICPR	SCS _BA+0x280	R/W	IRQ0 ~ IRQ31 Clear-Pending 控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CLRPEND[31:0]	软件控制写”1”将un-pend相应的中断.每个比特代表一个中断号IRQ0 ~ IRQ31 (向量号16 ~ 47). 写 “0”没有作用. 这个寄存器可以读回当前中断的pend状态.

IRQ0 ~ IRQ3中断优先级寄存器(NVIC_IPR0)

寄存器	偏移	R/W	描述	复位值
NVIC_IPR0	SCS _BA+0x400	R/W	IRQ0 ~ IRQ3中断优先级控制寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_3[1:0]	**IRQ3的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	**预留**
[23:22]	PRI_2[1:0]	**IRQ2的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:16]	-	**预留**
[15:14]	PRI_1[1:0]	**IRQ1的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[13:8]	-	**预留**
[7:6]	PRI_0[1:0]	**IRQ0的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[5:0]	-	**预留**

IRQ4 ~ IRQ7中断优先级寄存器(NVIC_IPR1)

寄存器	偏移	R/W	描述	复位值
NVIC_IPR1	SCS _BA+0x404	R/W	IRQ4 ~ IRQ7中断优先级控制寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_7[1:0]	**IRQ7的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	**预留**
[23:22]	PRI_6[1:0]	**IRQ6的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:16]	-	**预留**
[15:14]	PRI_5[1:0]	**IRQ5的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[13:8]	-	**预留**
[7:6]	PRI_4[1:0]	**IRQ4的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[5:0]	-	**预留**

IRQ8 ~ IRQ11中断优先级寄存器(NVIC_IPR2)

寄存器	偏移	R/W	描述	复位值
NVIC_IPR2	SCS _BA+0x408	R/W	IRQ8 ~ IRQ11中断优先级控制寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_11[1:0]	**IRQ11的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	**预留**
[23:22]	PRI_10[1:0]	**IRQ10的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:16]	-	**预留**
[15:14]	PRI_9[1:0]	**IRQ9的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[13:8]	-	**预留**
[7:6]	PRI_8[1:0]	**IRQ8的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[5:0]	-	**预留**

IRQ12 ~ IRQ15中断优先级寄存器(NVIC_IPR3)

寄存器	偏移	R/W	描述	复位值
NVIC_IPR3	SCS _BA+0x40C	R/W	IRQ12 ~ IRQ15中断优先级控制寄存器	0x0000_0000

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

Bits	描述	
[31:30]	**PRI_15[1:0]**	**IRQ15的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	**预留**
[23:22]	**PRI_14[1:0]**	**IRQ14的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:16]	-	**预留**
[15:14]	**PRI_13[1:0]**	**IRQ13的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[13:8]	-	**预留**
[7:6]	**PRI_12[1:0]**	**IRQ12的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[5:0]	-	**预留**

IRQ16 ~ IRQ19中断优先级寄存器(NVIC_IPR4)

寄存器	偏移	R/W	描述	复位值
NVIC_IPR4	SCS _BA+0x410	R/W	IRQ16 ~ IRQ19中断优先级控制寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_19[1:0]	**IRQ19的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	预留
[23:22]	PRI_18[1:0]	**IRQ18的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:16]	-	预留
[15:14]	PRI_17[1:0]	**IRQ17的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[13:8]	-	预留
[7:6]	PRI_16[1:0]	**IRQ16的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[5:0]	-	预留

IRQ20 ~ IRQ23中断优先级寄存器(NVIC_IPR5)

寄存器	偏移	R/W	描述	复位值
NVIC_IPR5	SCS _BA+0x414	R/W	IRQ20 ~ IRQ23中断优先级控制寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_23[1:0]	**IRQ23的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	预留
[23:22]	PRI_22[1:0]	**IRQ22的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:16]	-	预留
[15:14]	PRI_21[1:0]	**IRQ21的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[13:8]	-	预留
[7:6]	PRI_20[1:0]	**IRQ20的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[5:0]	-	预留

IRQ24 ~ IRQ27中断优先级寄存器(NVIC_IPR6)

寄存器	偏移	R/W	描述	复位值
NVIC_IPR6	SCS _BA+0x418	R/W	IRQ24 ~ IRQ27中断优先级控制寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_27[1:0]	**IRQ27的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	**预留**
[23:22]	PRI_26[1:0]	**IRQ26的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:16]	-	**预留**
[15:14]	PRI_25[1:0]	**IRQ25的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[13:8]	-	**预留**
[7:6]	PRI_24[1:0]	**IRQ24的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[5:0]	-	**预留**

IRQ28 ~ IRQ31中断优先级寄存器(NVIC_IPR7)

寄存器	偏移	R/W	描述	复位值
NVIC_IPR7	SCS _BA+0x41C	R/W	IRQ28 ~ IRQ31中断优先级控制寄存器	0x0000_0000

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

Bits	描述	
[31:30]	**PRI_31[1:0]**	**IRQ31的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	**预留**
[23:22]	**PRI_30[1:0]**	**IRQ30的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:16]	-	**预留**
[15:14]	**PRI_29[1:0]**	**IRQ29的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[13:8]	-	**预留**
[7:6]	**PRI_28[1:0]**	**IRQ28的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[5:0]	-	**预留**

5.2.7 中断源控制寄存器

除了NVIC中断控制寄存器, NuMicro MINI51™ 系列也实现了一些特别的控制寄存器以便于中断功能的使用, 包括“中断源标识”, ”NMI 源选择” 和“中断模式”. 描述如下.

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移	R/W	描述	复位值
INT_BA = 0x5000_0300				
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (Brownout) 中断源标识	0xXXXX_XXXX
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源标识	0xXXXX_XXXX
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 中断源标识	0xXXXX_XXXX
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源标识	0xXXXX_XXXX
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (GP0/1) 中断源标识	0xXXXX_XXXX
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GP2/3/4) 中断源标识	0xXXXX_XXXX
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWM) 中断源标识	0xXXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (BRAKE) 中断源标识	0xXXXX_XXXX
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源标识	0xXXXX_XXXX
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源标识	0xXXXX_XXXX
IRQ10_SRC	INT_BA+0x28	-	预留	0xXXXX_XXXX
IRQ11_SRC	INT_BA+0x2C	-	预留	0xXXXX_XXXX
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (UART) 中断源标识	0xXXXX_XXXX
IRQ13_SRC	INT_BA+0x34	-	预留	0xXXXX_XXXX
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI) 中断源标识	0xXXXX_XXXX
IRQ15_SRC	INT_BA+0x3C	-	预留	0xXXXX_XXXX
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (GP5) 中断源标识	0xXXXX_XXXX
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (HFIRC trim) 中断源标识	0xXXXX_XXXX
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I^2C) 中断源标识	0xXXXX_XXXX
IRQ19_SRC	INT_BA+0x4C	-	预留	0xXXXX_XXXX
IRQ20_SRC	INT_BA+0x50	-	预留	0xXXXX_XXXX
IRQ21_SRC	INT_BA+0x54	-	预留	0xXXXX_XXXX
IRQ22_SRC	INT_BA+0x58	-	预留	0xXXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	-	预留	0xXXXX_XXXX

寄存器	偏移	R/W	描述	复位值
INT_BA = 0x5000_0300				
IRQ24_SRC	INT_BA+0x60	-	预留	0xXXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (ACMP) 中断源标识	0xXXXX_XXXX
IRQ26_SRC	INT_BA+0x68	-	预留	0xXXXX_XXXX
IRQ27_SRC	INT_BA+0x6C	-	预留	0xXXXX_XXXX
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源标识	0xXXXX_XXXX
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (ADC) 中断源标识	0xXXXX_XXXX
IRQ30_SRC	INT_BA+0x78	-	预留	0xXXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	-	预留	0xXXXX_XXXX
NMI_CON	INT_BA+0x80	R/W	NMI 中断源选择控制寄存器	0x0000_0000
MCU_IRQ	INT_BA+0x84	R/W	MCU中断号标识寄存器	0x0000_0000

中断源标识寄存器(IRQn_SRC)

寄存器	偏移	R/W	描述	复位值
IRQn_SRC	INT_BA+0x00 …….. INT_BA+0x7C	R	MCU IRQ0 (BOD) 中断源标识 ： MCU IRQ31 (预留) 中断源标识	0xXXXX_XXXX

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

Bits	Address	IRQ No.	描述
[31:3]	-	-	**预留.**
[2:0]	INT_BA+0x00	0	Bit1~2: 预留 Bit0: BOD_INT
[2:0]	INT_BA+0x04	1	Bit1~2: 预留 Bit0: WDT_INT
[2:0]	INT_BA+0x08	2	Bit1~2: 预留 Bit0: EINT0 – 外部中断 0，来自P3.2
[2:0]	INT_BA+0x0C	3	Bit1~2: 预留 Bit0: EINT1 – 外部中断1，来自P5.2.
[2:0]	INT_BA+0x10	4	Bit2: 预留 Bit1: P1_INT Bit0: P0_INT
[2:0]	INT_BA+0x14	5	Bit2: P4_INT Bit1: P3_INT Bit0: P2_INT
[2:0]	INT_BA+0x18	6	Bit1~2: 预留 Bit0: PWM_INT

Bits	Address	IRQ No.	描述
[2:0]	INT_BA+0x1C	7	Bit1~2: 预留 Bit0: BRAKE_INT
[2:0]	INT_BA+0x20	8	Bit1~2: 预留 Bit0: TMR0_INT
[2:0]	INT_BA+0x24	9	Bit1~2: 预留 Bit0: TMR1_INT
[2:0]	INT_BA+0x28	10	预留
[2:0]	INT_BA+0x2C	11	预留
[2:0]	INT_BA+0x30	12	Bit1~2: 预留 Bit0: UART_INT
[2:0]	INT_BA+0x34	13	预留
[2:0]	INT_BA+0x38	14	Bit1~2: 预留 Bit0: SPI_INT
[2:0]	INT_BA+0x3C	15	预留
[2:0]	INT_BA+0x40	16	Bit1~2: 预留 Bit0: P5_INT
[2:0]	INT_BA+0x44	17	Bit1~2: 预留 Bit0: HFIRC_TRIM_INT
[2:0]	INT_BA+0x48	18	Bit1~2: 预留 Bit0: I2C_INT
[2:0]	INT_BA+0x4C	19	预留.
[2:0]	INT_BA+0x50	20	预留
[2:0]	INT_BA+0x54	21	预留
[2:0]	INT_BA+0x58	22	预留
[2:0]	INT_BA+0x5C	23	预留
[2:0]	INT_BA+0x60	24	预留
[2:0]	INT_BA+0x64	25	Bit1~2: 预留 Bit0: ACMP_INT
[2:0]	INT_BA+0x68	26	预留
[2:0]	INT_BA+0x6C	27	预留
[2:0]	INT_BA+0x70	28	Bit1~2: 预留 Bit0: PWRWU_INT

Bits	Address	IRQ No.	描述
[2:0]	INT_BA+0x74	29	Bit1~2: 预留 Bit0: ADC_INT
[2:0]	INT_BA+0x78	30	预留
[2:0]	INT_BA+0x7C	31	预留

NMI 中断源选择控制寄存器 (NMI_SEL)

寄存器	偏移	R/W	描述	复位值
NMI_SEL	INT_BA+0x80	R/W	NMI s中断源选择控制寄存器	0x0000_0000

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

Bits	描述	
[31:9]	-	预留
[8]	NMI_SEL_EN	NMI 中断源选择使能 (写保护) 设定这个比特将使能NMI_SEL来选择Cortex-M0的NMI中断源.
[7:5]	-	预留
[4:0]	NMI_SEL[4:0]	NMI 中断源选择 Cortex™-M0 CPU 的NMI中断可以从interrupt[31:0]选择. NMI_SEL[4:0] 用来选择NMI的中断源.

MCU中断请求源寄存器(MCU_IRQ)

寄存器	偏移	R/W	描述	复位值
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							
23	22	21	20	19	18	17	16
MCU_IRQ							
15	14	13	12	11	10	9	8
MCU_IRQ							
7	6	5	4	3	2	1	0
MCU_IRQ							

Bits	描述	
[31:0]	MCU_IRQ[31:0]	**MCU中断请求源寄存器** 这个寄存器的每个比特代表相应的中断源 MCU_IRQ 收集来自外设的所有中断并产生同步中断给Cortex-M0内核. Cortex-M0有两种中断模式, 正常模式和测试模式. MCU_IRQ收集来自外设的所有中断同步它们并产生中断给Cortex-M0. 当MCU_IRQ[n]比特是“0”时: 设定MCU_IRQ[n] 比特为“1”将产生一个中断给 Cortex-M0 NVIC[n]. 当MCU_IRQ[n]比特是 “1”时 (意味着相应的中断已经发生), 设定MCU_bit[n]比特为”1”将清除中断.写”0无效

5.3 系统管理

5.3.1 概述

系统管理章节包含下面的内容

- 系统内存映射
- 系统定时器 (SysTick)
- Nested Vectored Interrupt 控制器 (NVIC)
- PID系统管理寄存器
- 芯片及模块复位和多功能引脚配置系统管理寄存器
- Brownout 和芯片其它控制寄存器
- 外设中断源标识

5.3.2 系统复位

系统复位包含下面的事件.从寄存器**RSTSRC** 可以读到这些事件标志.

- 上电复位 (POR)
- /RESET脚上低电平复位
- 看门狗超时复位 (WDT)
- Brownout-Detected 复位(BOD)
- Cortex™-M0 CPU 复位
- 系统复位

5.3.3 系统电源分布

此系列芯片电源分布分3块.

- 模拟电源；AVDD 和 AVSS，负责给模拟部分供电
- 数字电源：VDD 和 VSS，通过内部稳压器固定到1.8V，负责给数字和I/O引脚部分供电
- 内嵌电容，用于内部电压稳压

内部电压稳压的输入，就是LDO_CAP，需要外接一个尽量靠近相应引脚的电容. 图5.3-1 显示了芯片的电源架构.

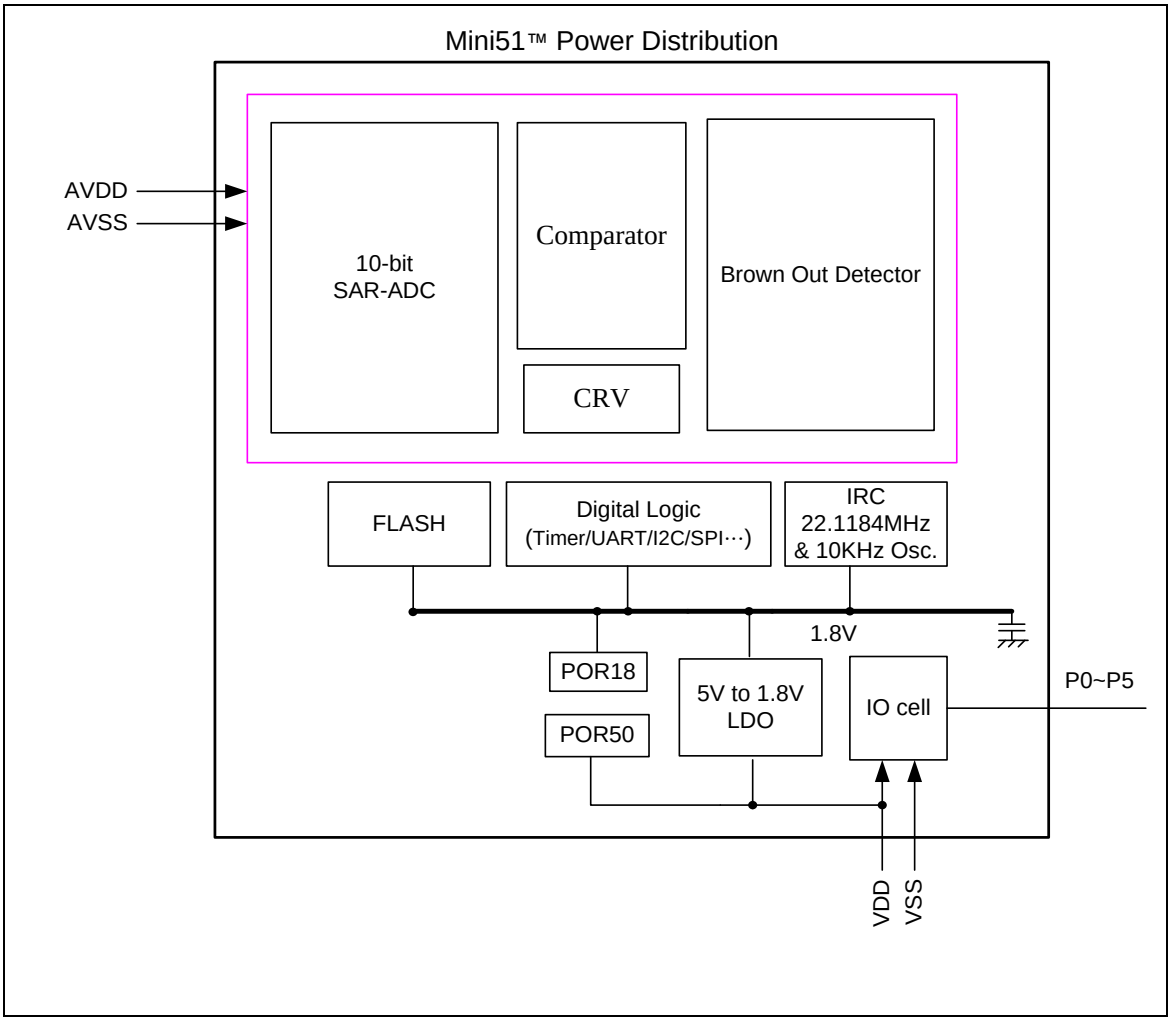


图 5.3-1 NuMicro Mini51™系列电源分布图

5.3.4 内存映射表

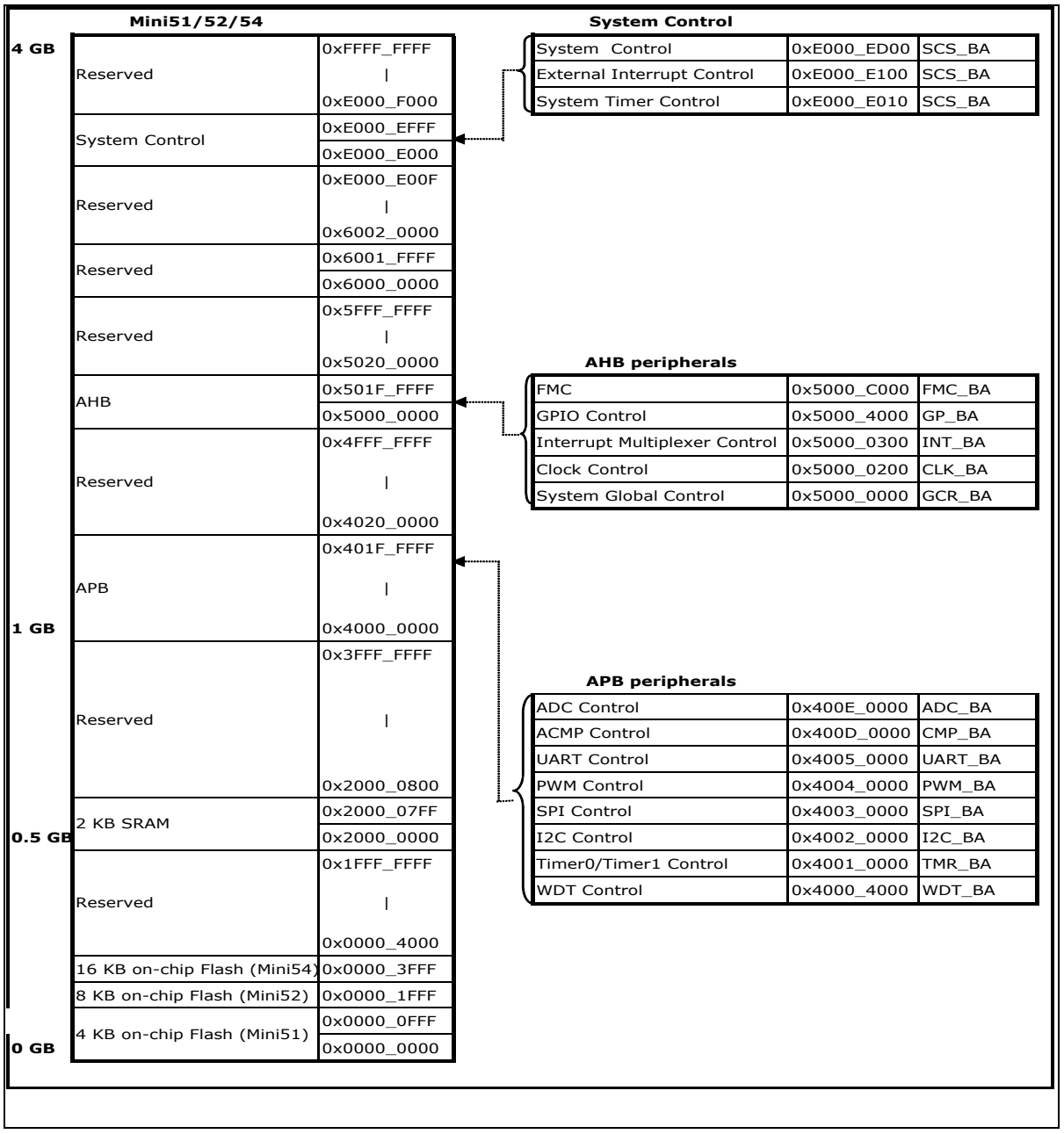


表 5.3-1 内存映射表

5.3.5 系统管理控制寄存器

R: 只读, **W**: 只写, **R/ W**: 可读写

寄存器	偏移	R/W	描述	复位值
GCR_BA = 0x5000_0000				
PDID	GCR_BA+0x00	R	Part Device Identification Number 寄存器	0xXXXX_XXXX[1]
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX
IPRSTC1	GCR_BA+0x08	R/W	IP 复位控制寄存器 1	0x0000_0000
IPRSTC2	GCR_BA+0x0C	R/W	IP 复位控制寄存器 2	0x0000_0000
BODCR	GCR_BA+0x18	R/W	Brownout 检测控制寄存器	0x00XX_000X
P0_MFP	GCR_BA+0x30	R/W	P0 多功能和输入类型控制寄存器	0x0000_0000
P1_MFP	GCR_BA+0x34	R/W	P1 多功能和输入类型控制寄存器	0x0000_0000
P2_MFP	GCR_BA+0x38	R/W	P2 多功能和输入类型控制寄存器	0x0000_0000
P3_MFP	GCR_BA+0x3C	R/W	P3 多功能和输入类型控制寄存器	0x0000_0000
P4_MFP	GCR_BA+0x40	R/W	P4 多功能和输入类型控制寄存器	0x0000_00C0
P5_MFP	GCR_BA+0x44	R/W	P5 多功能和输入类型控制寄存器	0x0000_0000
IRCTRIMCTL	GCR_BA+0x80	R/W	HIRC 修正控制寄存器	0x0000_0000
IRCTRIMIEN	GCR_BA+0x84	R/W	HIRC 修正中断使能寄存器	0x0000_0000
IRCTRIMINT	GCR_BA+0x88	R/W	HIRC 修正中断状态寄存器	0x0000_0000
RegLockAddr	GCR_BA+0x100	R/W	寄存器写保护寄存器	0x0000_0000

Part Device Identification Number 寄存器(PDID)

寄存器	偏移	R/W	描述	复位值
PDID	GCR_BA+0x00	R	Part Device Identification number 寄存器	0xXXXX_XXXX[1]

[1] Every part number has a unique default reset value.

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

Bits	描述	
[31:0]	PDID[31:0]	**Product Device Identification Number** 这个寄存器反映了设备的part number 码. 软件可以读这个寄存器来识别正在使用的是哪一个设备. 例如, MINI51LAN PDID 码是 “0x00205100”.

NuMicro Mini51™ series	Part Device Identification Number
MINI51LAN	0x00205100
MINI51ZAN	0x00205103
MINI51TAN	0x00205104
MINI52LAN	0x00205200
MINI52ZAN	0x00205203
MINI52TAN	0x00205204
MINI54LAN	0x00205400
MINI54ZAN	0x00205403
MINI54TAN	0x00205404

系统复位源寄存器(RSTSRC)

这个寄存器指示上一次的芯片复位源.

寄存器	偏移	R/W	描述	复位值
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX

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

Bits	描述	
[31:8]	-	**预留**
[7]	RSTS_CPU	如果软件写”1”到**CPU_RST** (IPRSTC1[1])来复位Cortex™-M0 CPU核和Flash内存控制器(FMC),硬件将设定**RSTS_CPU** 标志来指示上次的复位源. 1 = Cortex™-M0 CPU 核和FMC被软件写”1”到CPU_RST而复位 0 = 软件没有写”1”到CPU_RST复位Cortex™-M0 CPU 核和FMC. 软件写 “1” 可以清除这个比特.
[6]	-	**预留**
[5]	RSTS_MCU	来自Cortex-M0内核的“复位信号”导致硬件设置**RSTS_MCU**比特来指示上次的复位源. 1 = 写“1” 到比特SYSRESETREQ (AIRCR[2])导致Cortex-M0核发起的复位信号复位了整个系统，AIRCR是Application Interrupt and Reset Control Register, 地址 = 0xE000ED0C，在Cortex-M0内核的系统控制寄存器(SCS)空间. 0 = 没有来自Cortex-M0内核的复位信号. 软件写 “1” 可以清除这个比特.
[4]	RSTS_BOD	Brownout-Detected 模块发出“复位信号”导致硬件设置**RSTS_BOD** 标志位来指示上次的复位源 1 = Brownout-Detected模块发出复位信号导致系统被复位. 0 = BOD没有发出过复位信号. 软件写 “1” 可以清除这个比特.
[3]	-	**预留**

Bits	描述	
[2]	**RSTS_WDT**	看门狗模块发出”复位信号”导致硬件设置**RSTS_WDT** 标志位来指示上次的复位源 1 = 看门狗模块发出过复位信号复位系统. 0 = 看门狗没有发出过复位信号. 软件写 “1” 可以清除这个比特.
[1]	**RSTS_RESET**	RESET脚发出”复位信号”导致硬件设置**RSTS_RESET** 标志位来指示上次的复位源 1 = /RESET脚发出过复位信号复位系统. 0 = /RESET脚没有发出过复位信号. 软件写 “1” 可以清除这个比特.
[0]	**RSTS_POR**	Power-On 复位 (POR)模块或者比特CHIP_RST (IPRSTC1[0])被置发出”复位信号”,硬件将设置 **RSTS_POR** 标志位来指示上次的复位源 1 = Power-On-复位 (POR) 或者 CHIP_RST=1 发出过复位信号. 0 = POR没有发出过复位信号. 软件写 “1” 可以清除这个比特.

IP 复位控制寄存器1 (IPRSTC1)

寄存器	偏移	R/W	描述	复位值
IPRSTC1	GCR_BA+0x08	R/W	IP 复位控制寄存器1	0x0000_0000

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

Bits	描述	
[31:2]	-	预留
[1]	CPU_RST	**CPU 内核复位** 设置这个比特将导致CPU核和Flash 内存控制器(FMC)被复位，2个时钟周期之后，这个比特将自动清”0”. **这个比特是写保护比特,** 修改这个比特需要一个开锁序列，依次写“59h”, “16h”, “88h”到地址0x5000_0100可以解锁这个比特. 请参考寄存器 **RegLockAddr，** 地址 GCR_BA + 0x100. 1 = 复位CPU. 0 = 正常..
[0]	CHIP_RST	**CHIP one shot 复位** 设置这个比特将复位整个芯片，包括CPU核和所有的外设，2个时钟周期之后，这个比特将自动清”0”. CHIP_RST 和POR复位是一样的, 所有的芯片模块将被复位并且芯片设定也会从flash用户配置区域重新加载. **这个比特是写保护比特,** 修改这个比特需要一个开锁序列，依次写“59h”, “16h”, “88h”到地址0x5000_0100可以解锁这个比特. 请参考寄存器 **RegLockAddr，** 地址 GCR_BA + 0x100. 1 = 复位整个芯片. 0 = 正常.

IP 复位控制寄存器 2 (IPRSTC2)

设定这些比特为“1”将对相应外设产生异步复位信号. 用户需要再次写 “0”来使相关IP退出复位状态.

寄存器	偏移	R/W	描述	复位值
IPRSTC2	GCR_BA+0x0C	R/W	IP 复位控制寄存器 2	0x0000_0000

31	30	29	28	27	26	25	24
-			ADC_RST	-			
23	22	21	20	19	18	17	16
-	ACMP_RST	-	PWM_RST	-			UART_RST
15	14	13	12	11	10	9	8
-			SPI_RST	-			I2C_RST
7	6	5	4	3	2	1	0
-				TMR1_RST	TMR0_RST	GPIO_RST	-

Bits	描述	
[31:29]	-	预留
[28]	ADC_RST	ADC 控制器复位 1 = ADC 外设复位. 0 = ADC 外设正常工作.
[27:23]	-	预留
[22]	ACMP_RST	ACMP 控制器复位 1 = ACMP 外设复位. 0 = ACMP 外设正常工作.
[21]	-	预留
[20]	PWM_RST	PWM 控制器复位 1 = PWM 外设复位. 0 = PWM 外设正常工作.
[19:17]	-	预留
[16]	UART_RST	UART 控制器复位 1 = UART 外设复位. 0 = UART 外设正常工作.
[15:13]	-	预留

Bits	描述	
[12]	**SPI_ RST**	**SPI 控制器复位** 1 = SPI 外设复位. 0 = SPI 外设正常工作.
[11:9]	**-**	**预留**
[8]	**I2C _RST**	**I^2C 控制器复位** 1 = I^2C 外设复位. 0 = I^2C 外设正常工作.
[7:4]	**-**	**预留**
[3]	**TMR1_RST**	**Timer1 控制器复位** 1 = Timer1 外设复位. 0 = Timer1 外设正常工作.
[2]	**TMR0_RST**	**Timer0控制器复位** 1 = Timer0 外设复位. 0 = Timer0 外设正常工作.
[1]	**GPIO_RST**	**GPIO (P0~P5) 控制器复位** 1 = GPIO 外设复位. 0 = GPIO 外设正常工作.
[0]	**-**	**预留**

Brownout Detector 控制寄存器(BODCR)

一部分BODCR控制寄存器的值由flash用户配置区域初始化并且是写保护的.如果用户需要改变写保护的内容，需要一个解锁序列.解锁序列就是依次写数据“59h”, “16h”, “88h” 到写保护寄存器，地址“0x5000_0100”.在这3个数据之间，任何其它的数据写操作都将导致解锁序列退出.

用户可以查看寄存器0x5000_0100的比特0，确定是否解锁成功, “1” 表示解锁, “0” 表示没有解锁, 解锁后用户就可以更新被写保护的寄存器了. 写任何数据到地址“0x5000_0100”,将重新对写保护寄存器上锁.

寄存器	偏移	R/W	描述	复位值
BODCR	GCR_BA+0x18	R/W	Brownout Detector 控制寄存器	0x00XX_000X

31	30	29	28	27	26	25	24
-							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BOD38_TRIM				**BOD27_TRIM**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
-							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
-	**BOD_OUT**	-	**BOD_INTF**	**BOD_RSTEN**	**BOD_VL**		-

Bits	描述	
[31:7]	-	**预留**
[6]	**BOD_OUT**	**Brownout Detector** 输出状态 1 = Brownout Detector输出 “1”, 表示检测到电压低于BOD检测电压(BOD_VL). 0 = Brownout Detector输出“0”, 表示检测到电压低于BOD检测电压(BOD_VL).
[5]	-	**预留**
[4]	**BOD_INTF**	**Brownout Detector 中断标志** 1 = 当Brownout Detector 探测到V_{DD} 下降掉到BOD_VL电压以下，或者V_{DD}电压升高高于BOD_VL时，这个比特将被置为“1”. 如果Brownout中断使能，Brownout中断将发生. 0 = Brownout Detector 没有检测到任何V_{DD} 电压穿过BOD_VL 设定的情形.

Bits	描述	
[3]	**BOD_RSTEN**	**Brownout 复位使能（被初始化 & 写保护)** 1 = 使能Brownout “复位”功能, 当Brownout Detector 功能被使能并且检测到电压低于极限电压(BOD_VL)时，将复位芯片. 缺省值由用户配置区域寄存器config0 bit[20]决定. 0 = 使能Brownout “中断”功能, 当Brownout Detector 功能被使能并且检测到电压低于极限电压(BOD_VL)时，将发送中断给Cortex™-M0 CPU. 当 BOD_EN 被使能并且中断发生, 中断将一直保持到BOD_EN被设为”0”为止.可以通过关闭NVIC BOD中断或者关闭BOD_EN来阻止中断，需要BOD功能的时候再将BOD_EN打开.
[2:1]	**BOD_VL[1:0]**	**Brownout Detector 极限电压选择（被初始化 & 写保护)** 缺省值由用户配置区域寄存器config0 bit[22:21]决定.

BOD_VL[1]	BOD_VL[0]	Brownout 电压
1	1	Disable 2.7V and 3.8V
1	0	3.8V
0	1	2.7V
0	0	预留

[0]	-	**预留**

Port0 多功能控制寄存器(P0_MFP)

寄存器	偏移	R/W	描述	复位值
P0_MFP	GCR_BA+0x30	R/W	P0多功能和输入类型控制寄存器	0x0000_0000

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

Bits	描述	
[31:24]	-	预留
[23:16]	P0_TYPE[n]	**使能P0[7:0] 输入施密特触发功能** 1 = 使能P0[7:0] I/O 输入施密特触发功能. 0 = 关闭P0[7:0] I/O 输入施密特触发功能.
[15]	P0_ ALT[7]	**P0.7 功能选择** P0.7 脚的功能依靠 P0_MFP[7] 和P0_ALT[7] 的值.

P0_ALT[7]	P0_MFP[7]	P0.7 function
0	0	P0.7
0	1	预留
1	0	SPICLK (SPI)
1	1	预留

<table>
<tr><th>Bits</th><th colspan="2">描述</th></tr>
<tr><td>[14]</td><td>P0_ ALT[6]</td><td>P0.6 功能选择
P0.6 脚的功能依靠 P0_MFP[6] 和P0_ALT[6] 的值.
<table>
<tr><td>P0_ALT[6]</td><td>P0_MFP[6]</td><td>P0.6 功能</td></tr>
<tr><td>0</td><td>0</td><td>P0.6</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>MISO (SPI)</td></tr>
<tr><td>1</td><td>1</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[13]</td><td>P0_ ALT[5]</td><td>P0.5 功能选择
P0.5 脚的功能依靠P0_MFP[5] 和 P0_ALT[5]的值..
<table>
<tr><td>P0_ALT[5]</td><td>P0_MFP[5]</td><td>P0.5 功能</td></tr>
<tr><td>0</td><td>0</td><td>P0.5</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>MOSI (SPI)</td></tr>
<tr><td>1</td><td>1</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[12]</td><td>P0_ ALT[4]</td><td>P0.4 功能选择
P0.4脚的功能依靠P0_MFP[4] 和 P0_ALT[4] 的值.
<table>
<tr><td>P0_ALT[4]</td><td>P0_MFP[4]</td><td>P0.4 功能</td></tr>
<tr><td>0</td><td>0</td><td>P0.4</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>SPISS (SPI)</td></tr>
<tr><td>1</td><td>1</td><td>PWM5 (PWM)</td></tr>
</table></td></tr>
<tr><td>[11:10]</td><td>-</td><td>预留</td></tr>
<tr><td>[9]</td><td>P0_ ALT[1]</td><td>P0.1 功能选择
P0.1脚的功能依靠P0_MFP[1] 和 P0_ALT[1] 的值.
<table>
<tr><td>P0_ALT[1]</td><td>P0_MFP[1]</td><td>P0.1 功能</td></tr>
<tr><td>0</td><td>0</td><td>P0.1</td></tr>
<tr><td>0</td><td>1</td><td>SPISS (SPI)</td></tr>
<tr><td>1</td><td>0</td><td>RTSn (UART)</td></tr>
<tr><td>1</td><td>1</td><td>RX (UART)</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Bits</th><th colspan="2">描述</th></tr>
<tr><td>[8]</td><td>P0_ ALT[0]</td><td>P0.0 功能选择
P0.0脚的功能依靠P0_MFP[0] 和 P0_ALT[0] 的值.
<table>
<tr><td>P0_ALT[0]</td><td>P0_MFP[0]</td><td>P0.0 功能</td></tr>
<tr><td>0</td><td>0</td><td>P0.0</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>CTSn (UART)</td></tr>
<tr><td>1</td><td>1</td><td>TX (UART)</td></tr>
</table></td></tr>
<tr><td>[7:0]</td><td>P0_MFP[7:0]</td><td>P0 多功能选择
P0脚的功能依靠P0_MFP 和 P0_ALT的值.
细节请参考P0_ALT 的描述.</td></tr>
</table>

Port1多功能控制寄存器(P1_MFP)

寄存器	偏移	R/W	描述	复位值
P1_MFP	GCR_BA+0x34	R/W	P1 多功能和输入类型控制寄存器	0x0000_0000

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

Bits	描述	
[31:24]	-	预留
[23:16]	P1_TYPE[n]	使能P1[7:0] 输入施密特触发功能 1 = 使能P1[7:0] I/O 输入施密特触发功能. 0 = 禁止P1[7:0] I/O 输入施密特触发功能.
[15:14]	-	预留
[13]	P1_ ALT[5]	P1.5 功能选择 P1.5脚的功能依靠P1_MFP[5] 和 P1_ALT[5] 的值.

P1_ALT[5]	P1_MFP[5]	P1.5 功能
0	0	P1.5
0	1	AIN5 (ADC)
1	0	预留
1	1	CPP0 (CMP)

<table>
<tr><th>Bits</th><th>描述</th><th></th></tr>
<tr><td>[12]</td><td>P1_ ALT[4]</td><td>P1.4 功能选择
P1.4脚的功能依靠P1_MFP[4] 和P1_ALT[4] 的值.
<table><tr><td>P1_ALT[4]</td><td>P1_MFP[4]</td><td>P1.4 功能</td></tr><tr><td>0</td><td>0</td><td>P1.4</td></tr><tr><td>0</td><td>1</td><td>AIN4 (ADC)</td></tr><tr><td>1</td><td>0</td><td>预留</td></tr><tr><td>1</td><td>1</td><td>CPN0 (CMP)</td></tr></table></td></tr>
<tr><td>[11]</td><td>P1_ ALT[3]</td><td>P1.3 功能选择
P1.3脚的功能依靠P1_MFP[3] 和P1_ALT[3] 的值.
<table><tr><td>P1_ALT[3]</td><td>P1_MFP[3]</td><td>P1.3 功能</td></tr><tr><td>0</td><td>0</td><td>P1.3</td></tr><tr><td>0</td><td>1</td><td>AIN3 (ADC)</td></tr><tr><td>1</td><td>0</td><td>TX (UART)</td></tr><tr><td>1</td><td>1</td><td>预留</td></tr></table></td></tr>
<tr><td>[10]</td><td>P1_ ALT[2]</td><td>P1.2 功能选择
P1.2脚的功能依靠P1_MFP[2] 和P1_ALT[2] 的值.
<table><tr><td>P1_ALT[2]</td><td>P1_MFP[2]</td><td>P1.2 功能</td></tr><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>RX (UART)</td></tr><tr><td>1</td><td>1</td><td>预留</td></tr></table></td></tr>
<tr><td>[9]</td><td>-</td><td>预留</td></tr>
<tr><td>[8]</td><td>P1_ ALT[0]</td><td>P1.0 功能选择
P1.0脚的功能依靠1_MFP[0] 和 P1_ALT[0] 的值.
<table><tr><td>P1_ALT[0]</td><td>P1_MFP[0]</td><td>P1.0 功能</td></tr><tr><td>0</td><td>0</td><td>P1.0</td></tr><tr><td>0</td><td>1</td><td>AIN1 (ADC)</td></tr><tr><td>1</td><td>0</td><td>预留</td></tr><tr><td>1</td><td>1</td><td>预留</td></tr></table></td></tr>
<tr><td>[7:0]</td><td>P1_MFP[7:0]</td><td>P1 多功能选择
P1脚的功能依靠P1_MFP和P1_ALT的值.
细节请参考P1_ALT 的描述.</td></tr>
</table>

Port2 多功能控制寄存器(P2_MFP)

寄存器	偏移	R/W	描述	复位值
P2_MFP	GCR_BA+0x38	R/W	P2 多功能和输入类型控制寄存器	0x0000_0000

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

Bits	描述	
[31:24]	-	预留
[23:16]	P2_TYPE[n]	使能P2[7:0] 输入施密特触发功能 1 =使能P2[7:0] I/O 输入施密特触发功能. 0 =禁止P2[7:0] I/O 输入施密特触发功能.
[15]	-	预留
[14]	P2_ ALT[6]	P2.6 功能选择 P2.6 脚的功能依靠P2_MFP[6] 和P2_ALT[6] 的值.

P2_ALT[6]	P2_MFP[6]	P2.6 功能
0	0	P2.6
0	1	预留
1	0	PWM4 (PWM)
1	1	CPO1

<table>
<tr><th>Bits</th><th>描述</th><th></th></tr>
<tr><td>[13]</td><td>P2_ ALT[5]</td><td>P2.5 功能选择
P2.5 脚的功能依靠P2_MFP[5] 和P2_ALT[5] 的值.
<table>
<tr><td>P2_ALT[5]</td><td>P2_MFP[5]</td><td>P2.5 功能</td></tr>
<tr><td>0</td><td>0</td><td>P2.5</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>PWM3 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[12]</td><td>P2_ ALT[4]</td><td>P2.4 功能选择
P2.4 脚的功能依靠P2_MFP[4] 和P2_ALT[4] 的值.
<table>
<tr><td>P2_ALT[4]</td><td>P2_MFP 4]</td><td>P2.4 功能</td></tr>
<tr><td>0</td><td>0</td><td>P2.4</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>PWM2 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[11]</td><td>P2_ ALT[3]</td><td>P2.3 功能选择
P2.3 脚的功能依靠P2_MFP[3] 和P2_ALT[3] 的值.
<table>
<tr><td>P2_ALT[3]</td><td>P2_MFP[3]</td><td>P2.3 功能</td></tr>
<tr><td>0</td><td>0</td><td>P2.3</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>PWM1 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[10]</td><td>P2_ ALT[2]</td><td>P2.2 功能选择
P2.2脚的功能依靠P2_MFP[2] 和P2_ALT[2] 的值.
<table>
<tr><td>P2_ LT[2]</td><td>P2_MFP[2]</td><td>P2.2 功能</td></tr>
<tr><td>0</td><td>0</td><td>P2.2</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>PWM0 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[9:8]</td><td>-</td><td>预留</td></tr>
<tr><td>[7:0]</td><td>P2_MFP[7:0]</td><td>P2 多功能选择
P2脚的功能依靠P2_MFP和P2_ALT的值.
细节请参考P2_ALT 的描述.</td></tr>
</table>

Port3 多功能控制寄存器(P3_MFP)

寄存器	偏移	R/W	描述	复位值
P3_MFP	GCR_BA+0x3C	R/W	P3 多功能和输入类型控制寄存器	0x0000_0000

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

Bits	描述	
[31:24]	-	预留
[23:16]	P3_TYPE[n]	使能P3[7:0] 输入施密特触发功能 1 =使能P3[7:0] I/O 输入施密特触发功能. 0 =禁止P3[7:0] I/O 输入施密特触发功能.
[15]	-	预留
[14]	P3_ ALT[6]	P3.6 功能选择 P3.6 脚的功能依靠P3_MFP[6] 和P3_ALT[6] 的值.

P3_ALT[6]	P3_MFP[6]	P3.6 功能
0	0	P3.6
0	1	T1EX
1	0	CKO (时钟除频输出)
1	1	CPO0 (CMP)

<table>
<tr><th>Bits</th><th>描述</th><th></th></tr>
<tr><td>[13]</td><td>P3_ ALT[5]</td><td>P3.5 功能选择
P3.5脚的功能依靠P3_MFP[5] 和P3_ALT[5] 的值.
<table>
<tr><td>P3_ALT[5]</td><td>P3_MFP[5]</td><td>P3.5 功能</td></tr>
<tr><td>0</td><td>0</td><td>P3.5</td></tr>
<tr><td>0</td><td>1</td><td>T1 (Timer1)</td></tr>
<tr><td>1</td><td>0</td><td>SCL (I²C)</td></tr>
<tr><td>1</td><td>1</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[12]</td><td>P3_ ALT[4]</td><td>P3.4 功能选择
P3.4脚的功能依靠P3_MFP[4] 和P3_ALT[4] 的值.
<table>
<tr><td>P3_ALT[4]</td><td>P3_MFP[4]</td><td>P3.4 功能</td></tr>
<tr><td>0</td><td>0</td><td>P3.4</td></tr>
<tr><td>0</td><td>1</td><td>T0 (Timer0)</td></tr>
<tr><td>1</td><td>0</td><td>SDA (I²C)</td></tr>
<tr><td>1</td><td>1</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[11]</td><td>-</td><td>预留</td></tr>
<tr><td>[10]</td><td>P3_ ALT[2]</td><td>P3.2 功能选择
P3.2脚的功能依靠P3_MFP[2] 和P3_ALT[2] 的值.
<table>
<tr><td>P3_ALT[2]</td><td>P3_MFP[2]</td><td>P3.2 功能</td></tr>
<tr><td>0</td><td>0</td><td>P3.2</td></tr>
<tr><td>0</td><td>1</td><td>INT0</td></tr>
<tr><td>1</td><td>0</td><td>T0EX</td></tr>
<tr><td>1</td><td>1</td><td>STADC (ADC)</td></tr>
</table></td></tr>
<tr><td>[9]</td><td>P3_ ALT[1]</td><td>P3.1 功能选择
P3.1脚的功能依靠P3_MFP[1] 和P3_ALT[1] 的值.
<table>
<tr><td>P3_ALT[1]</td><td>P3_MFP[1]</td><td>P3.1 功能</td></tr>
<tr><td>0</td><td>0</td><td>P3.1</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>CPP1</td></tr>
<tr><td>1</td><td>1</td><td>AIN7 (ADC)</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Bits</th><th colspan="2">描述</th></tr>
<tr><td>[8]</td><td>P3_ ALT[0]</td><td>P3.0 功能选择
P3.0脚的功能依靠P3_MFP[0] 和P3_ALT[0] 的值.
<table>
<tr><td>P3_ALT 0]</td><td>P3_MFP[0]</td><td>P3.0 功能</td></tr>
<tr><td>0</td><td>0</td><td>P3.0</td></tr>
<tr><td>0</td><td>1</td><td>预留</td></tr>
<tr><td>1</td><td>0</td><td>CPN1</td></tr>
<tr><td>1</td><td>1</td><td>AIN6 (ADC)</td></tr>
</table></td></tr>
<tr><td>[7:0]</td><td>P3_MFP[7:0]</td><td>P3 多功能选择
P3脚的功能依靠P3_MFP 和P3_ALT的值.
细节请参考P3_ALT 的描述.</td></tr>
</table>

Port4 多功能控制寄存器(P4_MFP)

寄存器	偏移	R/W	描述	复位值
P4_MFP	GCR_BA+0x40	R/W	P4多功能和输入类型控制寄存器	0x0000_00C0

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

Bits	描述	
[31:24]	-	预留
[23:16]	P4_TYPE[n]	**使能P4[7:0] 输入施密特触发功能** 1 =使能P4[7:0] I/O 输入施密特触发功能. 0 =禁止P4[7:0] I/O 输入施密特触发功能.
[15]	P4_ ALT[7]	**P4.7 功能选择** P4.7脚的功能依靠P4_MFP[7] 和 P4_ALT[7] 的值. P4_ALT[7] / P4_MFP[7] / P4.7 功能 0 / 0 / P4.7 0 / 1 / ICE_DAT (ICE) 1 / x / 预留
[14]	P4_ ALT[6]	**P4.6 功能选择** P4.6脚的功能依靠P4_MFP[6] 和P4_ALT[6] 的值. P4_ALT[6] / P4_MFP[6] / P4.6 功能 0 / 0 / P4.6 0 / 1 / ICE_CLK (ICE) 1 / x / 预留
[13:8]	-	预留

Bits	描述	
[7:0]	**P4_MFP[7:0]**	**P4 多功能选择** P4脚的功能依靠P4_MFP 和P4_ALT的值. 细节请参考P4_ALT 的描述.

Port5 多功能控制寄存器(P5_MFP)

寄存器	偏移	R/W	描述	复位值
P5_MFP	GCR_BA+0x44	R/W	P5 多功能和输入类型控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
P5_TYPE							
15	14	13	12	11	10	9	8
P5_ALT							
7	6	5	4	3	2	1	0
P5_MFP							

Bits	描述	
[31:24]	-	预留
[23:16]	P5_TYPE[n]	使能P5[7:0] 输入施密特触发功能 1 =使能P5[7:0] I/O 输入施密特触发功能. 0 =禁止P5[7:0] I/O 输入施密特触发功能.
[15:14]	-	预留
[13]	P5_ ALT[5]	P5.5 功能选择 P5.5脚的功能依靠P5_MFP[5] 和P5_ALT[5] 的值. P5_ALT[5] / P5_MFP[5] / P5.5 功能 0 / 0 / P5.5 0 / 1 / 预留 1 / x / 预留
[12]	P5_ ALT[4]	P5.4 功能选择 P5.4脚的功能依靠P5_MFP[4] 和P5_ALT[4] 的值. P5_ALT[4] / P5_MFP[4] / P5.4 功能 0 / 0 / P5.4 0 / 1 / 预留 1 / x / 预留

<table>
<tr><th>Bits</th><th colspan="2">描述</th></tr>
<tr><td>[11]</td><td>P5_ ALT[3]</td><td>P5.3 功能选择
P5.3脚的功能依靠P5_MFP[3] 和P5_ALT[3] 的值.
<table>
<tr><td>P5_ALT[3]</td><td>P5_MFP[3]</td><td>P5.3 功能</td></tr>
<tr><td>0</td><td>0</td><td>P5.3</td></tr>
<tr><td>0</td><td>1</td><td>AIN0 (ADC)</td></tr>
<tr><td>1</td><td>x</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[10]</td><td>P5_ ALT[2]</td><td>P5.2 功能选择
P5.2脚的功能依靠P5_MFP[2] 和P5_ALT[2] 的值.
<table>
<tr><td>P5_ALT[2]</td><td>P5_MFP[2]</td><td>P5.2 功能</td></tr>
<tr><td>0</td><td>0</td><td>P5.2</td></tr>
<tr><td>0</td><td>1</td><td>INT1</td></tr>
<tr><td>1</td><td>x</td><td>预留</td></tr>
</table></td></tr>
<tr><td>[9]</td><td>P5_ ALT[1]</td><td>P5.1 功能选择
P5.1脚的功能依靠P5_MFP[1] 和P5_ALT[1] 的值.
<table>
<tr><td>P5_ALT[1]</td><td>P5_MFP[1]</td><td>P5.1 功能</td></tr>
<tr><td>0</td><td>0</td><td>P5.1</td></tr>
<tr><td>0</td><td>1</td><td>XTAL2</td></tr>
<tr><td>1</td><td>x</td><td>预留</td></tr>
</table>
注意: 使能外部XTAL功能，PWRCON 比特 [1:0] (XTLCLK_EN)也必须设定: 外部12 MHz 或者 32 KHz Crystal Oscillator 控制寄存器.</td></tr>
<tr><td>[8]</td><td>P5_ ALT[0]</td><td>P5.0 功能选择
P5.0脚的功能依靠P5_MFP[0] 和 P5_ALT[0] 的值.
<table>
<tr><td>P5_ALT[0]</td><td>P5_MFP[0]</td><td>P5.0 功能</td></tr>
<tr><td>0</td><td>0</td><td>P5.0</td></tr>
<tr><td>0</td><td>1</td><td>XTAL1</td></tr>
<tr><td>1</td><td>x</td><td>预留</td></tr>
</table>
注意: 使能外部XTAL功能，PWRCON 比特 [1:0] (XTLCLK_EN)也必须设定：外部12 MHz 或者 32 KHz Crystal Oscillator 控制寄存器.</td></tr>
<tr><td>[7:0]</td><td>P5_MFP[7:0]</td><td>P5 多功能选择
P5脚的功能依靠P5_MFP 和 P5_ALT的值.
细节请参考P5_ALT 的描述.</td></tr>
</table>

HFIRC 修正控制寄存器(IRCTRIMCTL)

寄存器	偏移	R/W	描述	复位值
IRCTRIMCTL	GCR_BA+0x80	R/W	HIRC 修正控制寄存器	0x0000_0000

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

Bits	描述	
[31:8]	-	预留
[7:6]	-	预留. 保持缺省值 “00”
[5:4]	TRIM_LOOP[1:0]	修正计算循环 这个域定义了修正值计算是基于多少个32.768 KHz 时钟. 例如, 如果TRIM_LOOP被设为“00”, 自动修正电路将基于4个32.768 KHz 时钟的平均频率计算修正值. 00 = 基于4个32.768 KHz时钟的平均频率计算修正值. 01 =基于8个32.768 KHz时钟的平均频率计算修正值. 10 =基于16个32.768 KHz时钟的平均频率计算修正值. 11 =基于32个32.768 KHz时钟的平均频率计算修正值.
[3:1]	-	预留
[0]	TRIM_SEL	修正频率选择 这个比特用来使能HFIRC自动修正功能. 当设定这个比特为“1”时, HFIRC自动修正功能将基于外部32.768KHz参考时钟自动修正HFIRC到22M. 自动修正期间,如果32.768 KHz 时钟错误被探测到或者达到修正重试限制次数，这个域将被自动清成 “0”. 0 = 禁止HFIRC 自动修正功能. 1 = 使能HFIRC自动修正功能并修正HFIRC 到 22 MHz.

HFIRC 修正中断使能寄存器(IRCTRIMIEN)

寄存器	偏移	R/W	描述	复位值
IRCTRIMIEN	GCR_BA+0x84	R/W	HIRC 修正中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-							
7	6	5	4	3	2	1	0
-					32K_ERR_IEN	TRIM_FAIL_IEN	-

Bits	描述	
[31:3]	-	**预留**
[2]	**32K_ERR_IEN**	**32.768 KHz 时钟错误中断使能** 在自动修正期间，如果32.768KHz时钟不精确，这个比特控制是否CPU将发生中断. 如果这个比特为高, 并且32K_ERR_INT被设, 中断将发生通知CPU 32.768KHz频率不精确. 0 = 禁止32K_ERR_INT 触发CPU中断. 1 =使能 32K_ERR_INT 触发CPU中断..
[1]	**TRIM_FAIL_IEN**	**修正失败中断使能** 这个比特控制当HFIRC修正值更新限制计数达到限定值，但是HFIRC频率仍然没有锁定在22时是否发生中断. 如果这个比特为高并且TRIM_FAIL_INT被设，中断将发生通知CPU HFIRC修正更新限制计数已经达到. 0 = 禁止 TRIM_FAIL_INT 触发中断给CPU. 1 =使能 TRIM_FAIL_INT 触发中断给CPU.
[0]	-	**预留**

HFIRC 修正中断状态寄存器 (IRCTRIMINT)

寄存器	偏移	R/W	描述	复位值
IRCTRIMINT	GCR_BA+0x88	R/W	HIRC 修正中断状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-							
7	6	5	4	3	2	1	0
-					32K_ERR_INT	TRIM_FAIL_INT	FREQ_LOCK

Bits	描述	
[31:3]	-	预留
[2]	32K_ERR_INT	**32.768 KHz 时钟错误中断状态** 这个比特指示32.768 KHz 时钟频率是否不精确. 一旦这个比特被设, 自动修正操作将停止并且TRIM_SEL将被硬件自动清成 “0”. 如果这个比特被设并且32K_ERR_IEN为高, 中断将发生通知CPU 32.768KHz时钟频率不精确. 写 “1” 清0. 0 = 32.768 KHz 时钟频率不精确. 1 = 32.768 KHz时钟频率精确.
[1]	TRIM_FAIL_INT	**修正失败中断状态** 这个比特指示HFIRC 修正值更新限制次数已经达到，但是HIRC频率一直无法锁定.一旦这个比特被设，自动修正操作将被停止并且TRIM_SEL将被硬件自动清成 “0”. 如果这个比特被设并且TRIM_FAIL_IEN为高, 中断将发生通知CPU HFIRC 修正值更新限制次数已经达到. 这个比特写 “1” 清0. 0 =修正值更新限制次数没有达到. 1 =修正值更新限制次数已经达到并且HFIRC频率没有锁定.
[0]	FREQ_LOCK	**HIRC 频率锁定状态** 这个比特指示HIRC频率锁定状态. 只是一个状态比特不触发任何中断.

写保护寄存器 (RegLockAddr)

一些系统控制寄存器需要写保护以防止被无意写到干扰到芯片的正常运行. 这些系统控制寄存器上电复位时是锁住的. 用户写这些寄存器之前需要一个解锁序列. 解锁序列就是依次写数据 “59h”, “16h” “88h” 到写保护寄存器，地址 “0x5000_0100”. 在此期间任何其他的数据写入都将退出解锁序列，使解锁失败.

解锁之后, 用户可以查看地址“0x5000_0100” 比特 0是否解锁成功: “1” 解锁, “0”没有解锁. 之后用户就可以更新被保护的寄存器 。解锁之后写任何值到地址”0x5000_0100”将重新上锁.

写这个寄存器可以打开RegUnLock比特，读可以得到RegUnLock 比特的状态.

寄存器	偏移	R/W	描述	复位值
RegLockAddr	GCR_BA+0x100	R/W	寄存器写保护寄存器	0x0000_0000

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

Bits	描述	
[31:1]	-	**预留**
[0]	**RegUnLock**	1 = 写保护寄存器已经解锁. 0 = 写保护寄存器是加锁的. 写任何数据到写保护寄存器不起作用. 写保护的寄存器有: IPRSTC1 – 地址x5000_0008 (IP 复位控制寄存器1) BODCR –地址0x5000_0018 (Brownout detector 控制寄存器) PORCR –地址0x5000_0024 (上电复位控制寄存器) PWRCON –地址0x5000_0200 (bit[6] 没有保护，可用来清除唤醒中断) APBCLK[0] –地址0x5000_0208 (看门狗时钟使能) CLKSEL0 –地址0x5000_0210 (HCLK 和 CPU STCLK时钟源选择) CLKSEL1[1:0] –地址0x5000_0214 (看门狗时钟源选择) NMI_SEL[8] –地址0x5000_380 (NMI 中断源使能) ISPCON –地址0x5000_C000 (Flash ISP 控制 存器) WTCR –地址0x4000_4000 (看门狗定时器控制寄存器)

5.4 时钟控制器

5.4.1 概述

时钟控制器为整个芯片提供时钟源, 包括系统时钟和所有外设的时钟. 时钟控制器也实现电源控制功能：独立的时钟开/关控制,时钟源选择和一个4比特的时钟除频器. CPU设定掉电使能位(PWR_DOWN_EN) 和Cortex-M0执行WFI指令之后，芯片将进入掉电模式. 之后, 芯片将等待唤醒中断触发离开掉电模式. 掉电模式下, 时钟控制器关闭外部晶振和内部22.1184 MHz RC振荡器来降低系统功耗.

5.4.2 Clock Generator

时钟发生器包括以下3个时钟源:

- 一个外部12 MHz (HXT) 或者32 KHz (LXT) crystal
- 一个内部22.1184 MHz RC oscillator (HIRC)
- 一个内部10 KHz oscillator (LIRC)

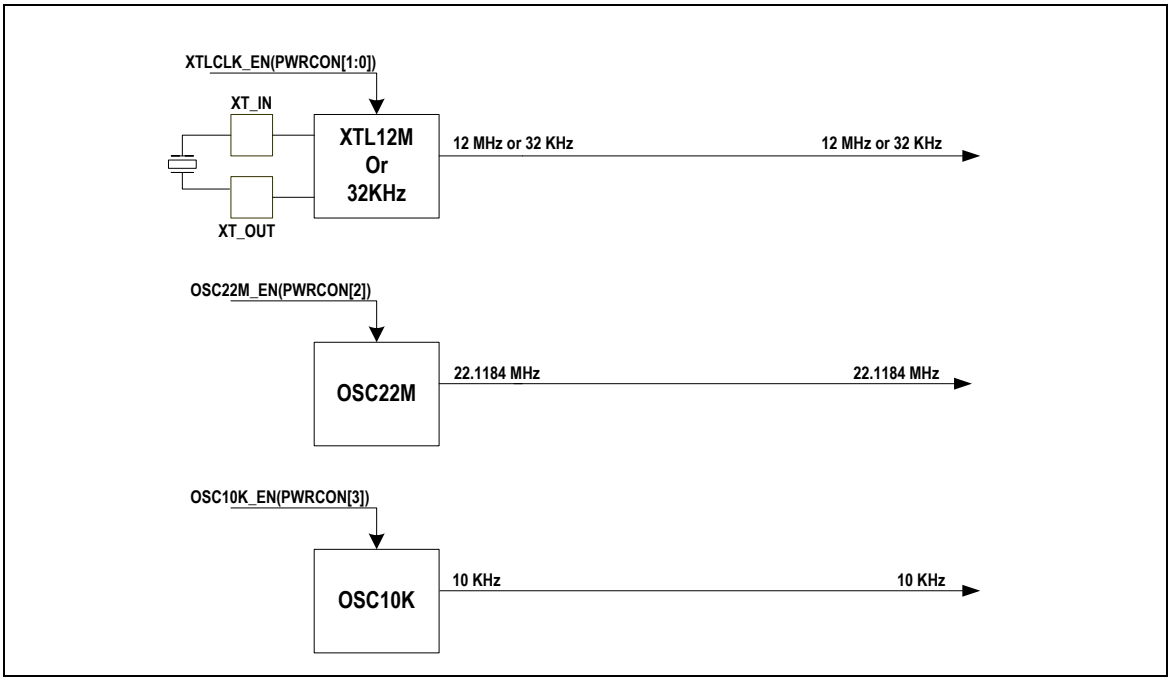


图 5.4-1 时钟发生器方块图

5.4.3 系统时钟& SysTick 时钟

系统时钟有3个时钟源来自时钟发生器. 时钟源切换依靠寄存器HCLK_S (CLKSEL0[2:0])的设定. 方块图如下.

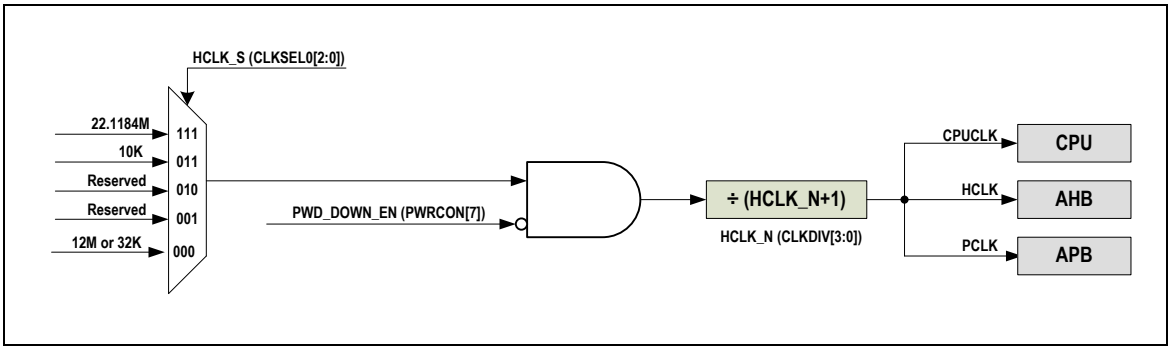


图 5.4-2 系统时钟方块图

Cortex-M0内核中的SysTick时钟源可以选择CPU时钟或者外部时钟 (SYST_CSR[2]). 如果使用外部时钟， SysTick时钟 (STCLK)有4 个时钟源. 时钟源切换依靠寄存器STCLK_S (CLKSEL0[5:3])的设定. 方块图如图5.4-3所示.

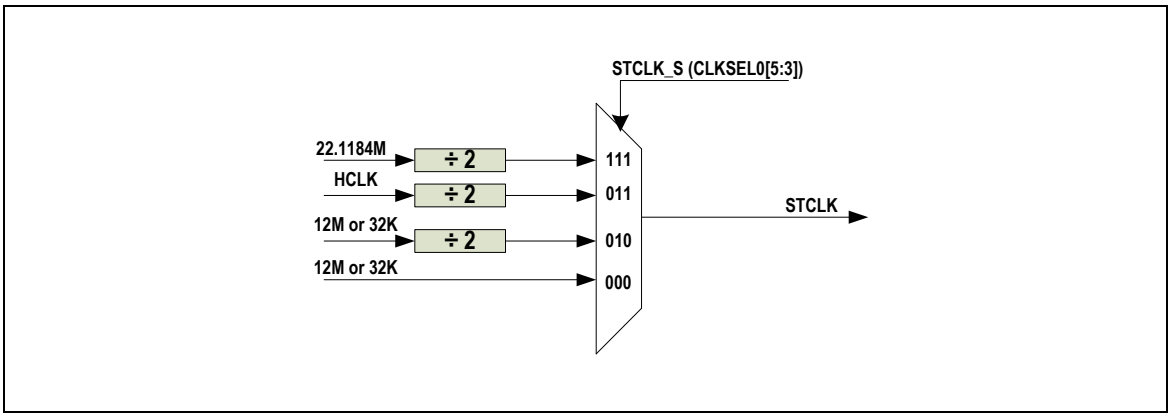


图 5.4-3 SysTick 时钟控制方块图

5.4.4 AHB 时钟源选择

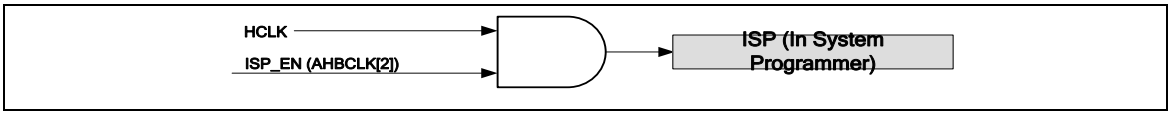


图 5.4-4 AHB 总线 HCLK 的时钟源

5.4.5 外设时钟源选择

不同的外设有不同的时钟源选择. 请参考**錯誤! 找不到參照來源。**章 CLKSEL1 & APBCLK 寄存器描述.

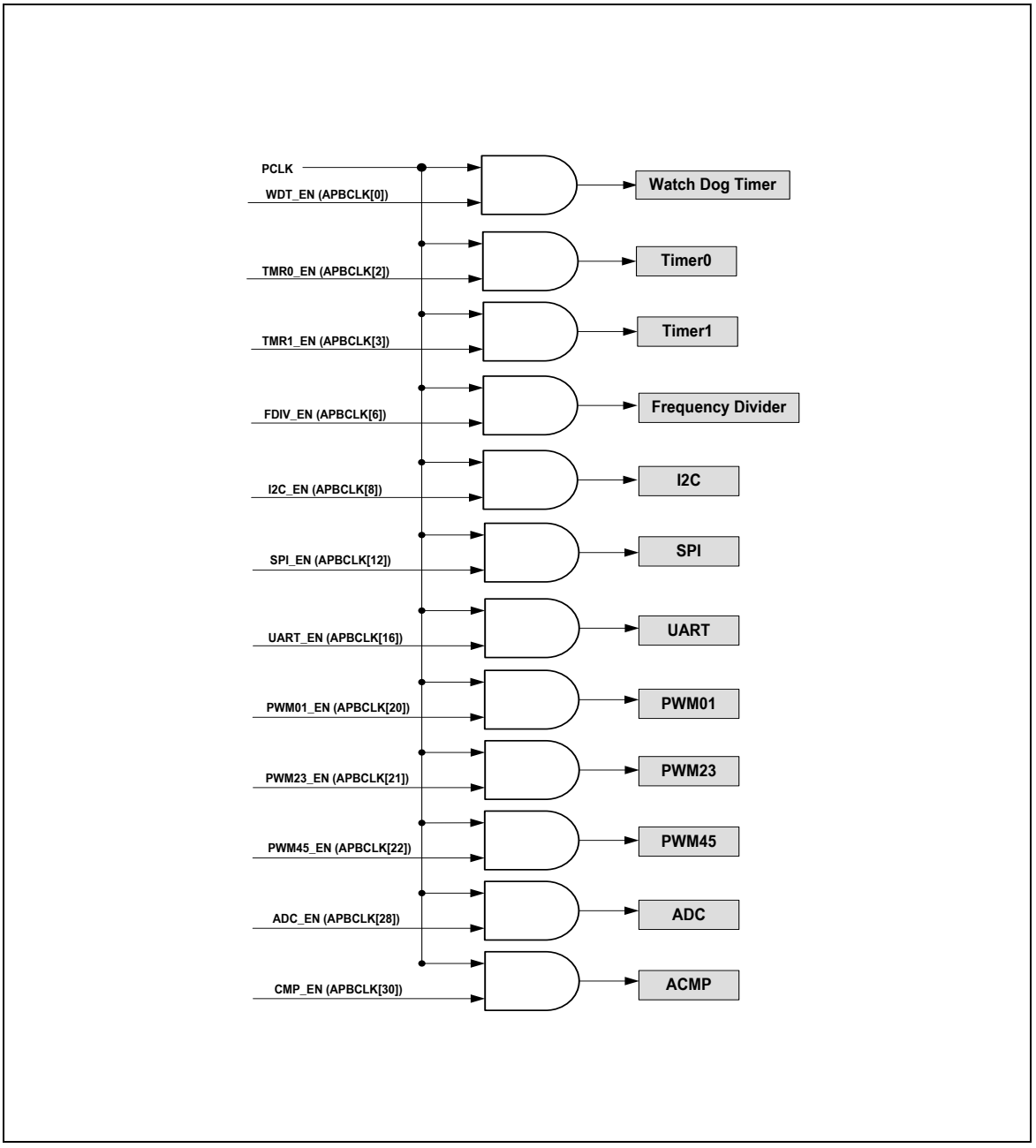


图 5.4-5 外设时钟源选择(PCLK)

	Ext. CLK (12M or 32K)	IRC22.1184M	IRC10K	PCLK
WDT	Yes	No	Yes	Yes
Timer0	Yes	Yes	Yes	Yes
Timer1	Yes	Yes	Yes	Yes
I^2C	No	No	No	Yes
SPI	No	No	No	Yes
UART	Yes	Yes	No	No
PWM	No	No	No	Yes
ADC	Yes	Yes	No	Yes
ACMP	No	No	No	Yes

表 5.4-1 外设时钟源选择表

5.4.6 掉电(睡眠)模式下的时钟

进入掉电模式时, 一些时钟源、外设时钟和系统时钟将被关闭.也有一些时钟源和外设时钟将保持激活..

下列时钟将保持激活:

- 时钟发生器
  - 内部 10 KHz RC oscillator (LIRC) clock
  - 外部 32.768 KHz crystal oscillator (LXT) clock (If PD_32K = “1” and XTLCLK_EN[1:0] = “10b”)
- 外设时钟(当外设采用10 KHz 作为时钟源时)
  - 看门狗时钟
  - Timer 0/1时钟

5.4.7 频率除频输出

用户可以选择某个时钟源除频之后从P3.6 IO脚输出。这个功能由一个2的倍数频率除频器完成，除频器有16级. 因而有16个输出频率选择：$F_{in}/2^1$ 到 $F_{in}/2^{17}$ ，Fin 是频率除频输出的时钟源.

输出公式 $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$, F_{in}是频率除频输出的时钟源, F_{out} 是频率除频输出的时钟频率，N是4-bit FREQDIV.FSEL[3:0]的值.

当 FREQDIV.FDIV_EN[4] 设为高时，上升沿将复位链表计数器并开始计数. 当 FREQDIV.FDIV_EN[4] 写0时, 链表计数器继续计数直到除频时钟到低电平并维持低电平.

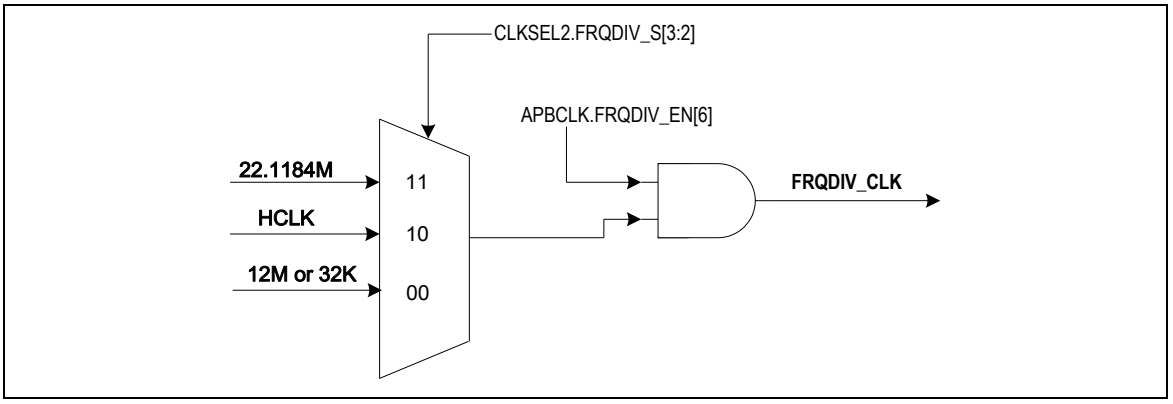


图 5.4-6 频率除频输出时钟源

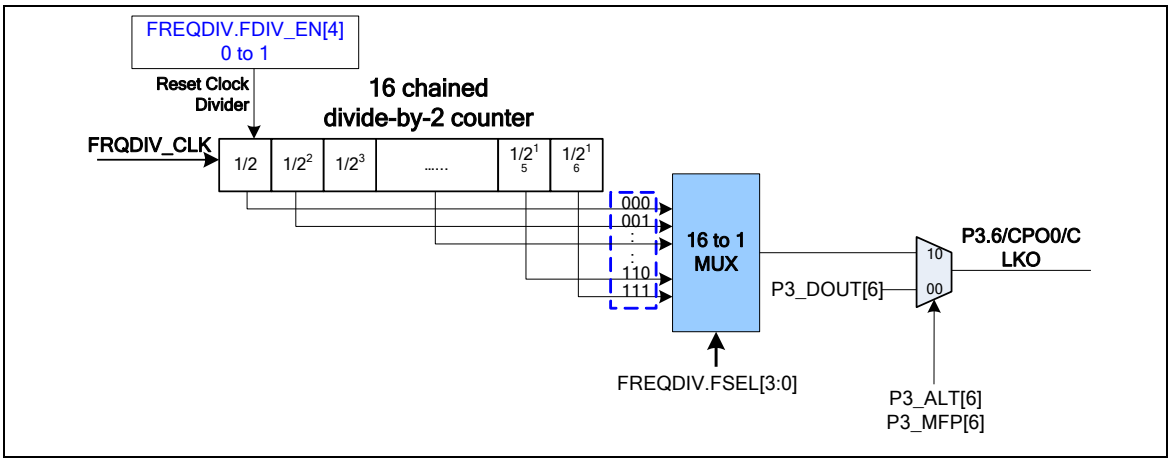


图 5.4-7 时钟除频方块图

5.4.8 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	R/W	描述	复位值
CLK_BA = 0x5000_0200				
PWRCON	CLK_BA+0x00	R/W	系统电源控制寄存器	0x0000_001C
AHBCLK	CLK_BA+0x04	R/W	AHB 设备时钟使能控制寄存器	0x0000_0005
APBCLK	CLK_BA+0x08	R/W	APB 设备时钟使能控制寄存器	0x0000_0001
CLKSTATUS	CLK_BA+0x0C	R/W	时钟状态监控寄存器	0x0000_0018
CLKSEL0	CLK_BA+0x10	R/W	时钟源选择控制寄存器0	0x0000_003F
CLKSEL1	CLK_BA+0x14	R/W	时钟源选择控制寄存器1	0xAFFF_FFFF
CLKDIV	CLK_BA+0x18	R/W	时钟除频寄存器	0x0000_0000
CLKSEL2	CLK_BA+0x1C	R/W	时钟源选择控制寄存器2	0x0000_00EF
FRQDIV	CLK_BA+0x24	R/W	频率除频控制寄存器	0x0000_0000

5.4.9 时钟控制寄存器描述

电源 控制寄存器(PWRCON)

除了BIT[6], 所有其它比特都是写保护的, 编程这些比特需要一个解锁序列, 依次写“59h”, “16h”, “88h” 到地址0x5000_0100 将解锁这些比特. 参考寄存器**RegLockAddr，** 地址GCR_BA + 0x100.

寄存器	偏移	R/W	描述	复位值
PWRCON	CLK_BA+0x00	R/W	系统电源控制寄存器	0x0000_001C

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-						PD_32K	-
7	6	5	4	3	2	1	0
PWR_DOWN	PD_WU_STS	WINT_EN	WU_DLY	OSC10K_EN	OSC22M_EN	XTLCLK_EN	

Bits	描述	
[31:10]	-	预留
[9]	PD_32K	这个比特控制掉电时 32.768KHz crystal oscillator 是否保持激活. 1 = 如果 XTLCLK_EN[1:0]=”10b”, 掉电时 32.768 KHz 晶振振荡器 (LXT) 将保持激活. 0 = 掉电时不起作用.
[8]	-	预留
[7]	PWR_DOWN_EN	**系统掉电激活或者使能位** 当芯片从掉电被唤醒时,这个比特会自动被清除,下次掉电需要用户再次设定这个比特. 掉电时, LDO、外部晶振和内部22M.1184 OSC都将被硬件自动关闭. 内部10K不受控制. **注意：当PWR_DOWN_EN =”1”时，如果XTLCLK_EN[1:0]=”10b”(使能 32 KHz 外部Crystal Oscillator)，则外部crystal oscillator 不会被关闭.** 掉电时, 所有AMBA 时钟 (HCLKx, CPU clock 和PCLKx) 不管选择的是什么时钟源都将被禁止. 但如果外设选择10K或者32K作为时钟源，外设时钟将不受这个比特控制. 1 = 芯片立即进入掉电模式或者等待CPU Idle 命令(WFI). 0 = 芯片在正常操作模式或者CPU进入Idle 模式.

Bits	描述	
[6]	**PD_WU_STS**	**掉电唤醒中断状态** 被”掉电唤醒事件”置位, 指示从掉电模式恢复. 如果GPIO, UART, WDT, ACMP, Timer 或者BOD 唤醒事件发生,这个标志将被置位，. 写“1”清除. 注意: 这个比特只有在PD_WU_INT_EN (PWRCON[5])等于”1”的情况下才起作用.
[5]	**PD_WU_INT_EN**	**掉电唤醒中断使能 (写保护)** 0 = 禁止. 1 = 使能. 当掉电唤醒时，中断将发生. 当PD_WU_STS和 PD_WU_INT_EN 都为高时，中断将发生.
[4]	**WU_DLY**	**使能唤醒延迟计数器 (写保护)** 当芯片从掉电模式唤醒时, 时钟控制器将延迟一定的时钟周期等待系统时钟稳定. 当芯片工作在12M的时候，延迟时间是4096个时钟周期；**当芯片工作在32.768 KHz的时候，延迟时间也是4096个时钟周期**；当芯片工作在22M的时候，延迟时间是16个时钟周期. 1 = 使能唤醒延迟. 0 = 禁止唤醒延迟.
[3]	**OSC10K_EN**	**内部10 KHz Oscillator (LIRC) 控制** 1 = 使能10KHz Oscillation. 0 = 禁止10KHz Oscillation..
[2]	**OSC22M_EN**	**内部22.1184 MHz Oscillator (HIRC) 控制** 1 = 使能22.1184 MHz Oscillation. 0 = 禁止22.1184 MHz Oscillation. 注意: OSC22M_EN 比特缺省就是 “1”.
[1:0]	**XTLCLK_EN[1:0]**	**外部 12 MHz (HXT) 或者 32 KHz (LXT) Crystal Oscillator 控制** 缺省时钟源来自内部22 MHz. 这两个比特缺省被设成“00” ， XTAL1 和 XTAL2 引脚都当作GPIO用. 00 = XTAL1和 XTAL2 都当作 GPIO, 关闭 XTL32K & XTAL12M (缺省). 01 = 使能 XTAL12M (HXT). 10 = 使能 XTAL32K (LXT). 11 = XTAL1作为外部时钟输入引脚, XTAL2作为GPIO. **注意:** 为了使能外部XTAL 功能，必须也设定P5_MFP寄存器的 P5_ALT[1:0] 和 P5_MFP[1:0] 比特.

寄存器 / 指令模式	PWR_DOWN_EN	CPU 执行 WFI 指令	关闭时钟
正常运行模式	0	NO	通过控制寄存器关闭所有时钟
IDLE 模式	0	YES	仅关闭CPU时钟
掉电模式	1	YES	大部分时钟被关闭，只有10K和WDT/Timer/PWM/ADC 外设时钟还活着.

表 5.4-2 掉电模式控制表

当芯片进入掉电模式时,用户可以通过一些中断源来唤醒芯片. 在设置PWR_DOWN_EN比特(PWRCON[7])之前，用户应该使能相关的中断源和NVIC 中断使能比特(NVIC_ISER)，以保证芯片可以成功被唤醒.

AHB 设备时钟使能控制寄存器（AHBCLK）

这些寄存器比特可以用来使能/禁止AHB总线上外设的时钟.

寄存器	偏移	R/W	描述	复位值
AHBCLK	CLK_BA+0x04	R/W	AHB 设备时钟使能控制寄存器	0x0000_0005

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

Bits	描述	
[31:3]	-	预留
[2]	ISP_EN	Flash ISP 控制器时钟使能控制 1 = 使能Flash ISP外设的时钟. 0 = 禁止Flash ISP外设的时钟.
[1]	-	预留
[0]	-	预留. 必须保持这个比特总是为 “1”，如果设成“0” 结果不可预期.

APB 设备时钟使能控制寄存器(APBCLK)

这些寄存器比特可以用来使能/禁止APB总线上外设的时钟.

寄存器	偏移	R/W	描述	复位值
APBCLK	CLK_BA+0x08	R/W	APB 设备时钟使能控制寄存器	0x0000_0001

31	30	29	28	27	26	25	24
-	CMP_EN	-	ADC_EN	-			
23	22	21	20	19	18	17	16
-	PWM45_EN	PWM23_EN	PWM01_EN	-			UART_EN
15	14	13	12	11	10	9	8
-			SPI_EN	-			I2C_EN
7	6	5	4	3	2	1	0
-	FDIV_EN	-		TMR1_EN	TMR0_EN	-	WDT_EN

Bits	描述	
[31]	-	预留
[30]	CMP_EN	模拟比较器时钟使能控制 1 = 使能模拟比较器的时钟. 0 = 禁止模拟比较器的时钟.
[29]	-	预留
[28]	ADC_EN	模数转换(ADC) 时钟使能控制 1 = 使能模数转换器的时钟. 0 = 禁止模数转换器的时钟
[27:23]	-	预留
[22]	PWM45_EN	PWM_45时钟使能控制 1 = 使能PWM45的时钟. 0 = 禁止PWM45的时钟.
[21]	PWM23_EN	PWM_23时钟使能控制 1 = 使能PWM23的时钟. 0 = 禁止PWM23的时钟.
[20]	PWM01_EN	PWM_01时钟使能控制 1 = 使能PWM01的时钟. 0 = 禁止PWM01的时钟.

Bits	描述	
[19:17]	-	**预留**
[16]	**UART_EN**	**UART时钟使能控制** 1 = 使能UART的时钟. 0 = 禁止UART的时钟.
[15:13]	-	**预留**
[12]	**SPI_EN**	**SPI时钟使能控制** 1 = 使能SPI的时钟. 0 = 禁止SPI的时钟.
[11:9]	-	**预留**
[8]	**I2C_EN**	**I^2C时钟使能控制** 1 = 使能I2C的时钟. 0 = 禁止I2C的时钟.
[7]	-	**预留**
[6]	**FDIV_EN**	**时钟除频输出时钟使能控制** 1 = 使能除频输出的时钟. 0 = 禁止除频输出的时钟.
[5:4]	-	**预留**
[3]	**TMR1_EN**	**Timer1时钟使能控制** 1 = 使能定时器1的时钟. 0 = 禁止定时器1的时钟.
[2]	**TMR0_EN**	**Timer0时钟使能控制** 1= 使能定时器0的时钟. 0 = 禁止定时器0的时钟.
[1]	-	**预留**
[0]	**WDT_EN**	**看门狗时钟使能控制** **这个比特是写保护的,** 编程这个比特需要一个解锁序列, 依次写 “59h”, “16h”, “88h” 到地址 “0x5000_0100”就可以解锁这个比特. 参考寄存器 **RegLockAddr** ， 地址GCR_BA + 0x100. 1 =使能看门狗的时钟. 0 =禁止看门狗的时钟.

时钟状态寄存器(CLKSTATUS)

这些寄存器比特可以用来监控是否芯片时钟源是稳定的, 否则时钟切换将失败.

寄存器	偏移	R/W	描述	复位值
CLKSTATUS	CLK_BA+0x0C	R/W	时钟状态监控寄存器	0x0000_0018

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

Bits	描述	
[31:8]	-	预留
[7]	CLK_SW_FAIL	时钟切换失败标志 1 = 时钟切换失败. 0 =时钟切换成功. 当要切换的目标时钟源不稳定的时候这个比特将被置. 写“1”清除.
[6:5]	-	预留
[4]	OSC22M_STB	OSC22M 时钟源稳定标志 1 = OSC22M 时钟是稳定的. 0 = OSC22M时钟不稳定或者没有使能.
[3]	OSC10K_STB	OSC10K时钟源稳定标志 1 = OSC10K时钟是稳定的. 0 = OSC10K时钟不稳定或者没有使能.
[2:1]	-	预留
[0]	XTL_STB	XTL12M or XTL32K时钟源稳定标志 1 = XTL12M 或者 XTL32K时钟是稳定的. 0 = XTL12M 或者 XTL32K 时钟不稳定或者没有使能.

时钟源选择控制寄存器 0 (CLKSEL0)

在时钟切换之前，相关时钟源(当前时钟源和要选择的时钟源)必须打开

寄存器	偏移	R/W	描述	复位值
CLKSEL0	CLK_BA+0x10	R/W	时钟源选择控制寄存器 0	0x0000_003F

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

Bits	描述	
[31:6]	-	**预留**
[5:3]	**STCLK_S[2:0]**	**Cortex™-M0 CPU SysTick 时钟源选择** 如果 SYST_CSR[2]=0, SysTick 使用下表的时钟源 **这些比特都是写保护的**, 编程这些比特需要一个解锁序列, 依次写"59h", "16h", "88h" 到地址0x5000_0100 将解锁这些比特. 参考寄存器**RegLockAddr，** 地址GCR_BA + 0x100. 000 = 时钟源来自外部12 MHz 或者 32 KHz 晶振. 001 = 预留. 010 =时钟源来自外部12 MHz 或者 32 KHz 晶振 **/2.** 011 = 时钟源来自 HCLK**/2.** 111 = 时钟源来自内部 22 MHz RC振荡器**/2.** 100, 101, 110 = 预留. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz 晶振.

Bits	描述	
[2:0]	**HCLK_S[2:0]**	**HCLK时钟源选择** **这些比特都是写保护的**, 编程这些比特需要一个解锁序列, 依次写“59h”, “16h”, “88h” 到地址0x5000_0100 将解锁这些比特. 参考寄存器**RegLockAddr，** 地址GCR_BA + 0x100. 000 =时钟源来自外部12 MHz 或者 32 KHz 晶振. 001 = 预留. 010 = 预留. 011 =时钟源来自内部10 KHz 振荡器. 111 =时钟源来自内部22 MHz 振荡器. Others = 预留. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz 晶振.

时钟源选择控制寄存器 1（CLKSEL1）

在时钟切换之前，相关时钟源(当前时钟源和要选择的时钟源)必须打开.

寄存器	偏移	R/W	描述	复位值
CLKSEL1	CLK_BA+0x14	R/W	时钟源选择控制寄存器1	0xAFFF_FFFF

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

Bits	描述	
[31:30]	**PWM23_S[1:0]**	**PWM2 和 PWM3时钟源选择** PWM2 和PWM3 使用相同的时钟源，它们也使用相同的预分频器. 00 = 预留. 01 =预留. 10 = 时钟源来自HCLK. 11 =预留. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz 晶振.
[29:28]	**PWM01_S[1:0]**	**PWM0 和 PWM1时钟源选择** PWM0 和PWM1 使用相同的时钟源，它们也使用相同的预分频器. 00 = 预留. 01 = 预留. 10 = 时钟源来自HCLK. 11 = 预留. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz 晶振.
[27:26]	-	预留

Bits	描述	
[25:24]	**UART_S[1:0]**	**UART时钟源选择** 00 = 时钟源来自外部12 MHz 或者 32 KHz 晶振. 01 = 预留. 10 = 时钟源来自内部22 MHz 振荡器. 11 = 预留. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz 晶振**.**
[23:15]	**-**	**预留**
[14:12]	**TMR1_S[2:0]**	**TIMER1时钟源选择** 000 = 时钟源来自外部 12 MHz 或者 32 KHz晶振. 001 = 时钟源来自内部 10 KHz 振荡器. 010 = 时钟源来自HCLK. 011 = 时钟源来自 外部 引脚触发. 111 = 时钟源来自内部 22 MHz 振荡器. 100, 101, 110 =预留. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz晶振**.**
[11]	**-**	**预留**
[10:8]	**TMR0_S[2:0]**	**TIMER0时钟源选择** 000 =时钟源来自外部 12 MHz 或者 32 KHz晶振. 001 = 时钟源来自内部 10 KHz 振荡器. 010 = 时钟源来自HCLK. 011 = 时钟源来自 外部 引脚触发. 111 = 时钟源来自内部 22 MHz 振荡器. 100, 101, 110 =预留. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz晶振**.**
[7:4]	**-**	**预留**
[3:2]	**ADC_S[1:0]**	**ADC时钟源选择** 00 = 时钟源来自外部 12 MHz 或者32 KHz晶振. 01 = 预留. 10 = 时钟源来自 HCLK clock. 11 = 时钟源来自 内部 22 MHz 振荡器. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz晶振.

Bits	描述	
[1:0]	**WDT_S[1:0]**	**WDT CLK时钟源选择** **这些比特都是写保护的**, 编程这些比特需要一个解锁序列, 依次写“59h”, “16h”, “88h” 到地址0x5000_0100 将解锁这些比特. 参考寄存器**RegLockAddr，** 地址GCR_BA + 0x100. 00 = 时钟源来自外部 12 MHz 或者32 KHz晶振. 01 = 预留. 10 = 时钟源来自HCLK/2048. 11 = 时钟源来自内部 10 KHz 振荡器. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz晶振.

时钟源选择控制寄存器2(CLKSEL2)

在时钟切换之前，相关时钟源(当前时钟源和要选择的时钟源)必须打开.

寄存器	偏移	R/W	描述	复位值
CLKSEL2	CLK_BA+0x1C	R/W	时钟源选择控制寄存器 2	0x0000_00EF

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

Bits	描述	
[31:6]	-	预留
[5:4]	PWM45_S[1:0]	PWM4 和PWM5 使用相同的时钟源，它们也使用相同的预分频器. 00 = 预留. 01 = 预留. 10 = 时钟源来自HCLK. 11 = 预留. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz crystal clock.
[3:2]	FRQDIV_S[1:0]	**Clock Divider时钟源选择** 00 = 时钟源来自外部 12 MHz 或者 32 KHz crystal clock. 01 = 预留. 10 = 时钟源来自HCLK. 11 = 时钟源来自内部 22 MHz oscillator clock. **注意:** 设定 PWRCON[1:0] 可以选择12 MHz 或者 32 KHz crystal clock.
[1:0]	-	预留

时钟除频寄存器(CLKDIV)

寄存器	偏移	R/W	描述	复位值
CLKDIV	CLK_BA+0x18	R/W	时钟除频寄存器	0x0000_0000

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

Bits	描述	
[31:24]	-	预留
[23:16]	ADC_N[7:0]	ADC 时钟除频值 ADC输入时钟频率 = (ADC 时钟源频率) / (ADC_N + 1).
[15:12]	-	预留
[11:8]	UART_N[3:0]	UART时钟除频值 UART输入时钟频率 = (UART时钟源频率) / (UART_N + 1).
[7:4]	-	预留
[3:0]	HCLK_N[3:0]	HCLK时钟除频值 HCLK输入时钟频率 = (HCLK时钟源频率) / (HCLK_N + 1).

频率除频 控制寄存器(FRQDIV)

寄存器	偏移	R/W	描述	复位值
FRQDIV	CLK_BA+0x24	R/W	频率除频控制寄存器	0x0000_0000

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

Bits	描述	
[31:5]	**-**	**预留**
[4]	**DIVIDER_EN**	**频率除频输出使能比特** 0 = 禁止频率除频输出器. 1 = 使能频率除频输出器.
[3:0]	**FSEL[3:0]**	**频率除频输出器频率选择比特** 输出频率公式 **$F_{out} = F_{in}/2^{(N+1)}$,** F_{in} 是除频输出的时钟源. F_{out} 是除频之后输出的时钟. N 就是这 4 个比特FSEL[3:0]的值.

5.5 模拟比较器 (CMP)

5.5.1 概述

NuMicro MINI51™ 系列包含两个比较器. 可以在某些不同的条件下使用. 当正输入大于负输入时比较器输出逻辑”1” ;否则输出”0”. 当比较器输出值改变时,每个比较器都可以配置发生中断. 方块图如图5.5-1所示.

注意:在模拟比较功能使能之前,模拟输入引脚必须被配置为输入模式.

5.5.2 特性

- 模拟输入电压范围: 0 ~ 5.0 V
- 支持迟滞功能
- 两个模拟比较器负端支持选择内部参考电压输入
- 任何比较器都可以请求比较器中断

5.5.3 方块图

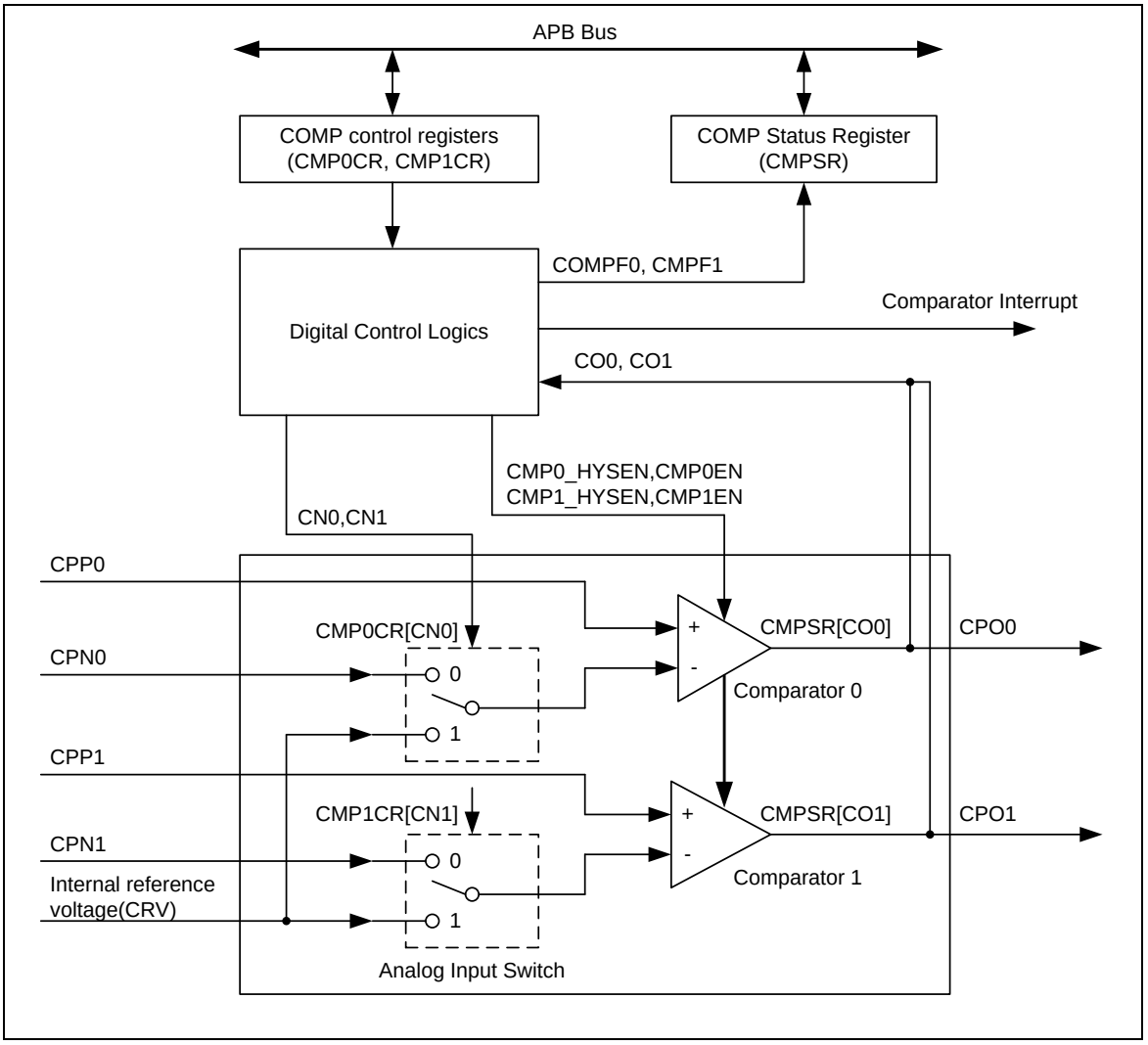


图 5.5-1 模拟比较器方块图

5.5.4 功能描述

5.5.4.1 *中断源*

比较器由PCLK采样在CMPSR寄存器的CO1(CO2)产生输出. 如果CMP0CR (CMP1CR)寄存器的CMP0IE (CMP1IE)比特被设,当比较器的输出CO0 (CO1)状态改变的时候,比较器标志位CMPF0 (CMPF1) 将被设 ,比较器中断将发生. 软件可以写”0”到CMPF0 和 CMPF1来清除中断.

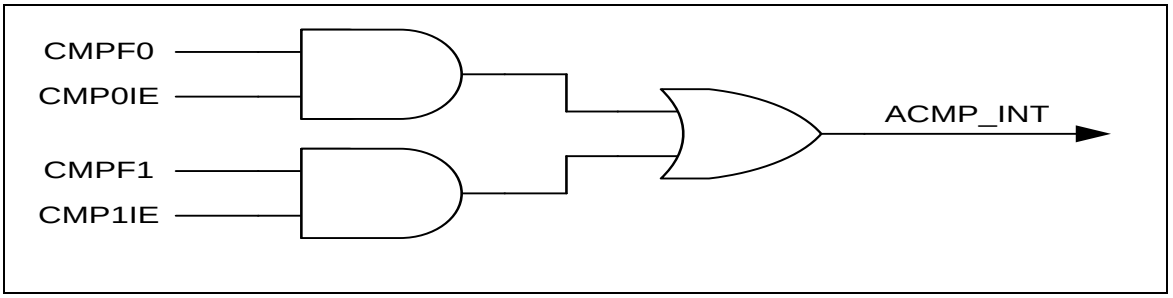


图 5.5-2 比较器中断源

5.5.5 比较器参考电压(CRV)

5.5.5.1 CRV 介绍

比较器参考电压模块 (CRV) 负责为比较器产生参考电压. CRV 模块由阶梯寄存器和模拟切换寄存器组成, 通过设定CRVS[3:0] 寄存器可以设定CRV输出电压,通过设定OUL_SEL寄存器可以选择比较器的参考电压.

5.5.5.2 CRV 的特性包括:

- 通过设定CRVS[3:0]寄存器用户可以设定参考电压.
- 当设定OUT_SEL=0 (选择Band-gap 1.35 V 输出)时，自动关闭阶梯电阻以降低功耗.

CRV模块方块图如下所示:

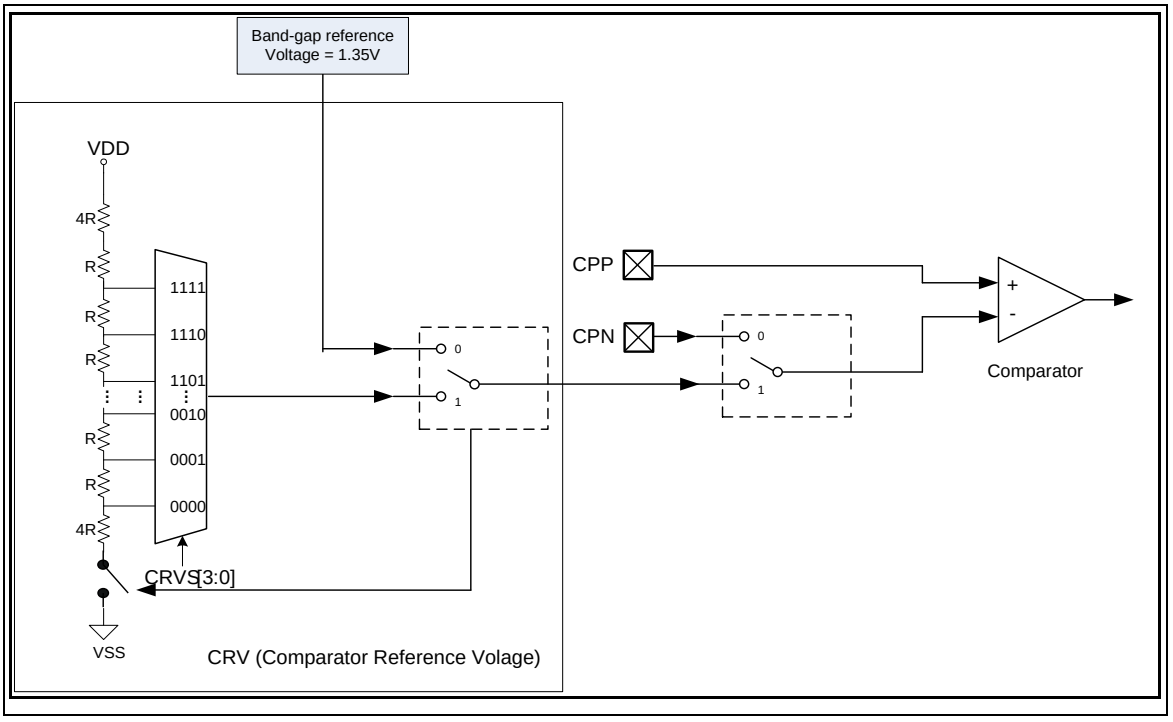


图 5.5-3 比较器参考电压方块图

5.5.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移	R/W	描述	复位值
CMP_BA = 0x400D_0000				
CMP0CR	CMP_BA+0x00	R/W	比较器0 控制寄存器	0x0000_0000
CMP1CR	CMP_BA+0x04	R/W	比较器1 控制寄存器	0x0000_0000
CMPSR	CMP_BA+0x08	R/W	比较器状态寄存器	0x0000_0000
CMPRVCR	CMP_BA+0x0C	R/W	比较器参考电压控制寄存器	0x0000_0000

5.5.7 寄存器描述

比较器0 控制寄存器(CMP0CR)

寄存器	偏移	R/W	描述	复位值
CMP0CR	CMP_BA+0x00	R/W	比较器0 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-							
7	6	5	4	3	2	1	0
-			CN0	-	CMP0_HYSEN	CMP0IE	CMP0EN

Bits	描述	
[31:5]	-	预留
[4]	CN0	**比较器0负端输入选择** 1 = 内部比较器参考电压(Vref=1.35V 或者来自 CRV 的设定值) 被选择当作比较器负端输入. 0 = 比较器引脚CPN0 被选择当作比较器负端输入.
[3]	-	预留
[2]	CMP0_HYSEN	**比较器0 滞后使能** 1 = 使能 CMP0 迟滞功能，典型滞后范围20mV. 0 = 禁止CMP0 迟滞功能 (缺省值).
[1]	CMP0IE	**比较器0中断使能** 1 = 使能 CMP0 的中断功能. 0 = 禁止 CMP0 的中断功能. CMP0转换完成之后,如果CMP0IE 比特被设为 “1”,中断将发生.
[0]	CMP0EN	**比较器0 使能** 1 = 使能. 0 = 禁止. 在CMP0EN 比特被设之后,需要等待10us，比较器才会输出.

比较器1 控制寄存器(CMP1CR)

寄存器	偏移	R/W	描述	复位值
CMP1CR	CMP_BA+0x04	R/W	比较器1 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-							
7	6	5	4	3	2	1	0
-			CN1	-	CMP1_HYSEN	CMP1IE	CMP1EN

Bits	描述	
[31:5]	-	预留
[4]	CN1	**比较器1负端输入选择** 1 = 内部比较器参考电压(Vref=1.35V 或者来自 CRV 的设定值) 被选择当作比较器负端输入. 0 = 比较器引脚CPN1被选择当作比较器负端输入.
[3]	-	预留
[2]	CMP1_HYSEN	**比较器1滞后使能** 1 = 使能 CMP1 迟滞功能，典型滞后范围20mV. 0 = 禁止CMP1 迟滞功能 (缺省值).
[1]	CMP1IE	**比较器1中断使能** 1 = 使能 CMP1 的中断功能. 0 = 禁止 CMP1的中断功能. CMP1转换完成之后,如果CMP1IE 比特被设为 “1”,中断将发生.
[0]	CMP1EN	**比较器1 使能** 1 = 使能. 0 = 禁止. 在CMP0EN 比特被设之后,需要等待10us，比较器才会输出.

比较器状态寄存器(CMPSR)

寄存器	偏移	R/W	描述	复位值
CMPSR	CMP_BA+0x08	R/W	比较器状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-							
7	6	5	4	3	2	1	0
-				CO1	CO0	CMPF1	CMPF0

Bits	描述	
[31:4]	-	预留
[3]	CO1	**比较器1 输出** 软件读这个比特可以得知当前模拟比较器的输出状态. 当比较器被禁止时(CMP1EN = 0)被清除.
[2]	CO0	**比较器0 输出** 软件读这个比特可以得知当前模拟比较器的输出状态. 当比较器被禁止时(CMP1EN = 0)被清除.
[1]	CMPF1	**比较器1 标志** 当模拟比较器1输出改变时硬件将设置这个比特. 如果CMP1IE 被使能的话,中断将发生. 这个比特写“1”清除.
[0]	CMPF0	**比较器0标志** 当模拟比较器0输出改变时硬件将设置这个比特. 如果CMP0IE 被使能的话,中断将发生. 这个比特写“1”清除.

CMPRV 控制寄存器(CMPRVCR)

寄存器	偏移	R/W	描述	复位值
CMPRVCR	CMP_BA+0x0C	R/W	比较器参考电压控制寄存器	0x0000_0000

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

Bits	描述	
[31:8]	-	预留
[7]	OUT_SEL	**CRV模块输出选择** 1= 选择 CRVS 设定电压. 0= 选择 Band-gap 1.35 V 电压.
[6:4]	-	预留
[3:0]	CRVS[3:0]	**比较器参考电压设定** **CRVS** = VDD x (1/6+CRV[3:0]/24)

5.6 模数转换器 (ADC)

5.6.1 概述

NuMicro MINI51™ 系列包含一个10 bit逐次逼近型模数转换器(SAR A/D 转换器) ,有8个输入通道. A/D 转换可以由软件或者外部STADC/P3.2 脚触发.

注意:在ADC功能使能前,模拟输入脚必须配置为输入模式.

5.6.2 特性

- 模拟输入电压范围: 0 ~ Vref (最大 5.0 V)
- 10比特分辨率,8比特精度
- 最多8个单端(single-end)模拟输入通道
- 最大 ADC 时钟频率是 6 MHz
- 最高转换率150K SPS
- A/D 转换每次在一个指定的通道转换一次I
- A/D转换可以这样开始
  - 软件写“1” 到ADST 比特
  - 外部 STADC引脚
- 转换结果放到数据寄存器,带有效位和溢出指示
- 转换结果可以和特定的值比较,当转换结果等于比较寄存器的设定时,用户可以选择是否产生中断
- 通道 7 支持 2 种输入源: 外部模拟电压输入和内部固定的带隙(band-gap)电压

5.6.3 方块图

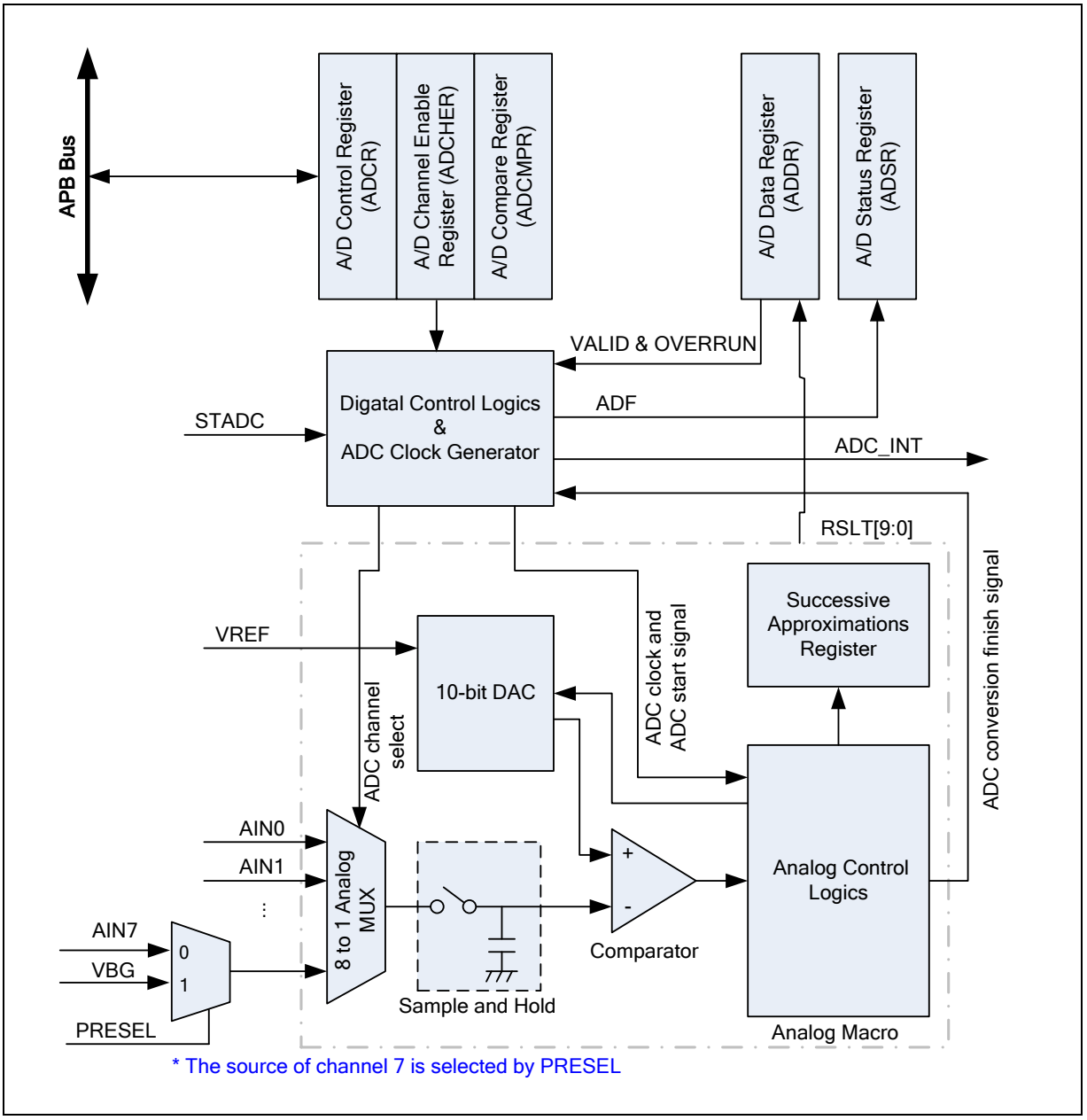


图 5.6-1 ADC 控制器方块图

5.6.4 操作过程

A/D 转换器是10比特逐次逼近型的. 当改变模拟输入使能通道时,为了避免操作错误,软件必须清除ADCR 寄存器的ADST位。如果ADST位被清除,A/D 转换器立即丢弃当前转换结果,转换器将进入空闲状态.

5.6.4.1 ADC 时钟发生器

ADC最大采样率150 K. 通过一个2比特的ADC_S (CLKSEL1[3:2])选择时钟源, ADC时钟频率等于ADC 时钟源频率除以一个8比特的预分频,公式如下:

ADC时钟频率= (ADC 时钟源频率) / (ADC_N+1);
8-bit ADC_N 在寄存器CLKDIV[23:16]中.

一般来说, 软件可以设置ADC_S 和 ADC_N 得到6 MHz 或者更低的频率.

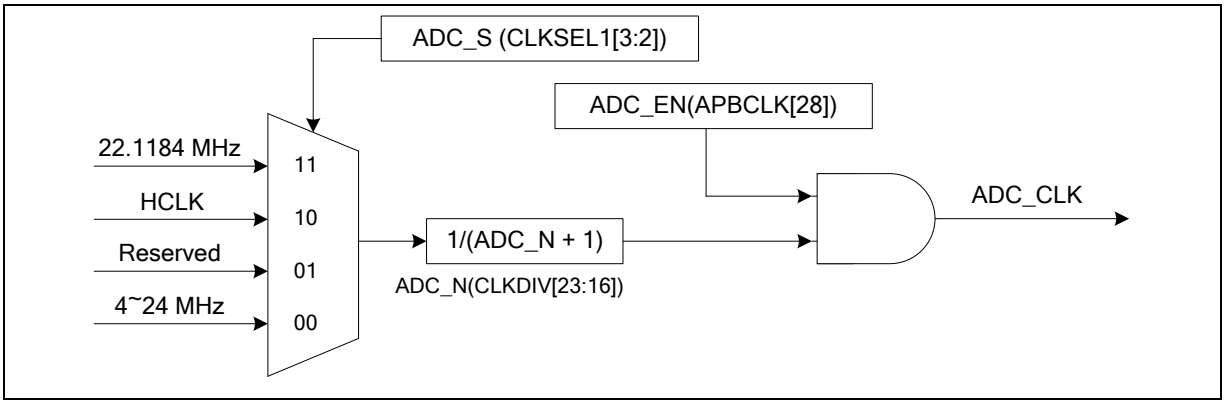


图 5.6-2 ADC 时钟控制

5.6.4.2 操作

A/D 转换器一次只可以在一个指定的通道上转换一次. 操作如下:

1. 当ADCR寄存器的ADST比特由软件或者外部触发引脚设成“1”时, A/D 开始转换.
2. A/D 转换完成时, 结果将存入A/D 数据寄存器(ADDR).
3. ADSR 寄存器的ADF比特将被设成 “1”. 如果ADCR寄存器的ADIE比特也比设成“1”, ADC 中断将发生.
4. 在A/D转换期间ADST 比特保持 “1”. 当A/D 转换结束时, ADST 比特自动清成“0” ,A/D 转换器进入空闲状态.

注意: 如果软件使能了多个通道,最小号码的通道将被选择,其它的通道被忽略.

5.6.4.3 外部触发输入采样

A/D 转换可以由外部引脚触发. 当ADCR.TRGEN 置为高时, ADC 外部触发功能被使能,触发输入源来自STADC引脚. 软件可以设置TRGCOND 来选择触发条件是**下降还是上升沿**. 一个8-bit的采样计数器用来消除毛刺. 如果边沿触发条件发生,高或者低脉冲必须保持4个PCLK时钟.否则视为无效脉冲.

5.6.4.4 比较功能监控转换结果

NuMicro MINI51™ 系列 ADC 控制器有 2 组比较寄存器, ADCMPR0 和 ADCMPR1, 用来监控 A/D 转换

的结果, 参考 图 5.6-3 A/D 转换结果监控逻辑图

. 通过设定CMPCH (ADCMPRx[5:0]) ,软件可以选择监控哪个通道. CMPCOND 比特可以用来设置比较结果小于还是大于(等于)指定的值(CMPD[9:0]). 当CMPCH 指定的通道转换完成, 比较动作自动触发一次. 当比较结果与设定匹配时, 比较计数值将增加1, 否则比较计数值将被清成“0”. 当计数值等于 (CMPMATCNT+1) 时,CMPF比特将被设为 “1”, 如果CMPIE 比特被使能,一个 ADC_INT 中断将发生. 细节逻辑图如图 5.6-3所示.

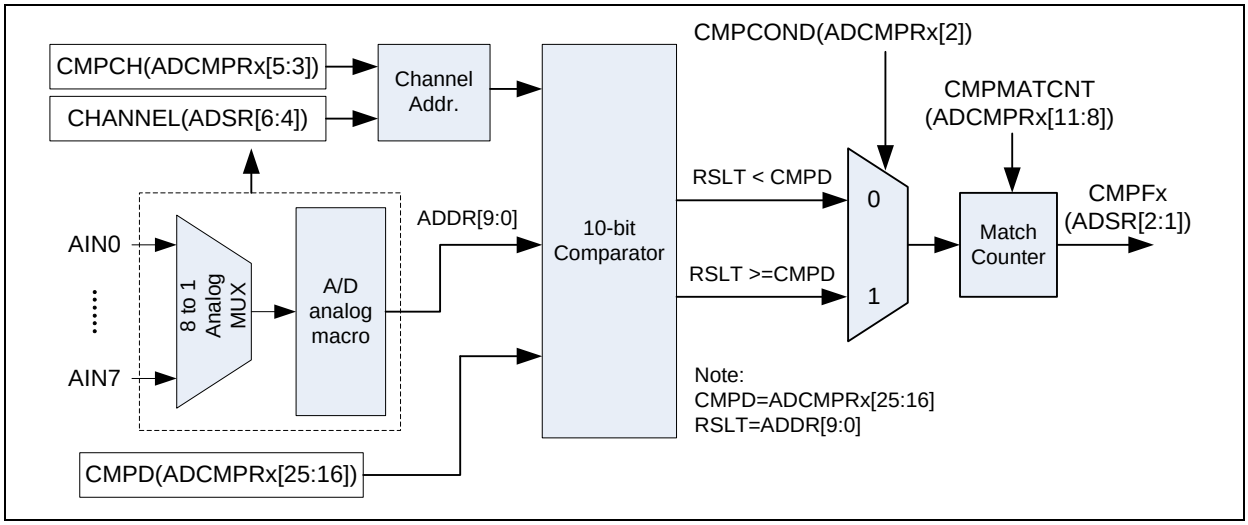


图 5.6-3 A/D 转换结果监控逻辑图

5.6.4.5 中断源

ADC 中断有3个中断源. 当一次ADC 转换完成时, A/D 转换结束标志ADF将被设成 “1”. CMPF0 和 CMPF1 是比较功能的比较标志. 当转换结果与ADCMPR0/1的设定匹配时, 相应的标志将被设成“1”. 当任一个标志, ADF, CMPF0 和 CMPF1, 被设成 “1”并且相应的中断使能位, ADCR的ADIE 和 ADCMPR0/1 的CMPIE, 也被设成“1”, ADC中断将发生. 软件可以通过清除这些标志位来清除中断.

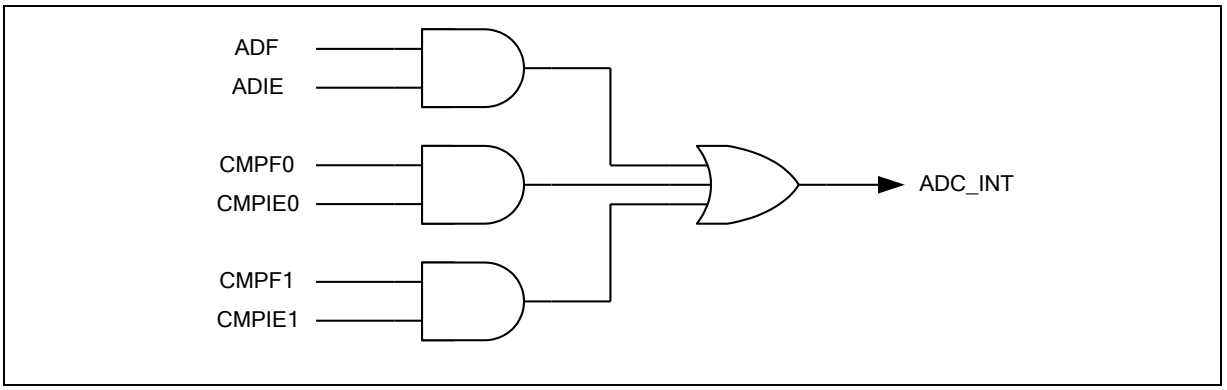


图 5.6-4 A/D 控制器中断

5.6.5 ADC 寄存器映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移	R/W	描述	复位值
ADC_BA = 0x400E_0000				
ADDR	ADC_BA+0x00	R	A/D 数据寄存器	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

5.6.6 ADC 寄存器描述

A/D 数据寄存器(ADDR)

寄存器	偏移	R/W	描述	复位值
ADDR	ADC_BA+0x00	R	A/D 数据寄存器	0x0000_0000

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

Bits	描述	
[31:18]	-	预留
[17]	VALID	有效标志 1 = RSLT[9:0] 中的数据有效. 0 = RSLT[9:0] 中的数据无效. 当ADC转换完成时这个比特将被设成 “1” ,ADDR寄存器被读之后,硬件将其清成0.
[16]	OVERRUN	溢出标志 1 = RSLT[9:0]中的数据被覆盖. 0 = RSLT[9:0] 中的数据没有被覆盖. 如果RSLT[9:0] 中的数据没有读走,新的转换已经完成,数据又写入这个寄存器, OVERRUN 将被设成1”. ADDR寄存器被读之后,硬件将其清成0.
[15:10]	-	预留
[9:0]	RSLT[9:0]	A/D 转换结果 这个域包含ADC转换的结果.

A/D控制寄存器(ADCR)

寄存器	偏移	R/W	描述	复位值
ADCR	ADC_BA+0x20	R/W	ADC 控制寄存器	0x0000_0000

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

Bits	描述	
[31:12]	-	**预留**
[11]	ADST	**A/D 转换开始** 1 = 开始转换. 0 = 停止转换，A/D转换器进入空闲状态. ADST比特可以由两个方法设成 “1”: 软件和外部触发引脚STADC. 转换完成时,ADST将由硬件自动清“0”.
[10:9]	-	**预留**
[8]	TRGEN	**外部触发使能** 使能或者禁止外部STADC引脚触发A/D转换. 1 = 使能. 0 = 禁止.
[7]	-	**预留**
[6]	TRGCOND	**外部触发条件** 这个比特决定外部触发引脚STADC是下降沿还是上升沿触发ADC转换. 信号必须保持稳定至少4 个PCLKs时钟高电平和低电平. 0 = 下降沿. 1 = 上升沿.
[5:2]	-	**预留**
[1]	ADIE	**A/D 中断使能**

Bits	描述	
		1 = 使能A/D 中断. 0 = 禁止A/D 中断. 如果ADIE比特被设成”1”,A/D 转换结束后,中断将发生.
[0]	**ADEN**	**A/D 转换器使能** 1 = 使能. 0 = 禁止. 开始A/D转换前, 这个比特应该设成 “1”. 清成“0” 将关闭A/D 模数转换电路的电源.

A/D 通道使能寄存器(ADCHER)

寄存器	偏移	R/W	描述	复位值
ADCHER	ADC_BA+0x24	R/W	A/D 通道使能	0x0000_0000

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

Bits	描述	
[31:9]	-	预留
[8]	PRESEL	**模拟输入通道 7选择** 0 = 输入模拟信号. 1 = 输入Band-gap (VBG) 电压. 注意: 当软件选择band-gap 电压作为ADC通道7的模拟输入时, ADC 时钟频率限制低于300 KHz.
[7]	CHEN7	**使能模拟输入通道7** 1 = 使能. 0 = 禁止
[6]	CHEN6	**使能模拟输入通道6** 1 =使能. 0 =禁止.
[5]	CHEN5	**使能模拟输入通道5** 1 =使能. 0 =禁止.
[4]	CHEN4	**使能模拟输入通道4** 1 =使能.

Bits	描述	
		0 =禁止.
[3]	**CHEN3**	**使能模拟输入通道3** 1 =使能. 0 =禁止.
[2]	**CHEN2**	**使能模拟输入通道2** 1 =使能. 0 =禁止.
[1]	**CHEN1**	**使能模拟输入通道1** 1 =使能. 0 =禁止.
[0]	**CHEN0**	**使能模拟输入通道0** 1 =使能. 0 =禁止. **注意:** 如果软件使能了多于一个通道，最低编号的通道将被选择，其它使能的通道将被忽略.这就意味着通道0有最高权限.

A/D 比较寄存器 0/1 (ADCMPR0/1)

寄存器	偏移	R/W	描述	复位值
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

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

Bits	描述	
[31:26]	-	**预留**
[25:16]	**CMPD[9:0]**	**比较数据** 这10个比特的值用来和特定通道的转换结果比较.
[15:12]	-	**预留**
[11:8]	**CMPMATCNT [3:0]**	**比较匹配次数** 当特定A/D 通道模数转换结果和CMPCOND[2]定义的比较条件匹配时, 内部匹配计数器的值将加1. 当内部匹配计数器的值等于(CMPMATCNT +1)时, CMPFx比特将被置位 .
[7:6]	-	**预留**
[5:3]	**CMPCH[2:0]**	**比较通道选择** **CMPCH[2:0]** / **比较通道选择** **000** 与通道0 的转换结果进行比较. **001** 与通道1的转换结果进行比较. **010** 与通道2 的转换结果进行比较. **011** 与通道3 的转换结果进行比较. **100** 与通道4 的转换结果进行比较.

Bits	描述	
		101 与通道5 的转换结果进行比较 **110** 与通道6 的转换结果进行比较 **111** 与通道7 的转换结果进行比较.
[2]	**CMPCOND**	**比较条件** 1 = 当10比特的A/D转换结果大于或者等于CMPD (ADCMPRx[27:16])中10比特的值时, 内部匹配计数器的值将加1. 0 = 当10比特的A/D转换结果小于CMPD (ADCMPRx[27:16])中10比特的值时,内部匹配计数器的值将加1. **注意:** 当内部计数器的值等于(CMPMATCNT +1)时, CMPFx 比特将被置.
[1]	**CMPIE**	**比较中断使能** 1 =使能比较功能中断. 0 =禁止比较功能中断. 如果比较功能被使能，比较条件符合CMPCOND 和 CMPMATCNT的设定, CMPFx比特将被设, 同时, 如果CMPIE被设成“1”, 比较中断将发生.
[0]	**CMPEN**	**比较使能** 1 = 使能比较功能. 0 =禁止比较功能. 设置这个比特为“1”，当特定通道的转换结果存入ADDR寄存器的时候,将和CMPD[9:0]的值比较.

A/D 状态寄存器(ADSR)

寄存器	偏移	R/W	描述	复位值
ADSR	ADC_BA+0x30	R/W	ADC 状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
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
-	CHANNEL			BUSY	CMPF1	CMPF0	ADF

Bits	描述	
[31:17]	-	预留
[16]	OVERRUN	溢出标志 是ADDR寄存器OVERRUN比特的映射比特.
[15:9]	-	预留
[8]	VALID	数据有效标志 是ADDR寄存器VALID比特的映射比特.
[7]	-	预留
[6:4]	CHANNEL [2:0]	当前转换通道 当BUSY=1时,这个域反映当前转换通道； 当BUSY=0时, 这个域将等于”0”. 该比特是只读的.
[3]	BUSY	BUSY/IDLE 1 = A/D 转换器正忙. 0 = A/D 转换器空闲. 是ADCR寄存器ADST比特的映射比特. 这个比特是只读的.
[2]	CMPF1	比较标志1 当选择的通道A/D转换的结果与ADCMPR1寄存器的设定匹配时,这个比特将被设成“1”.这个比特写 “1”清除.

Bits	描述	
		1 = ADDR中的转换结果与ADCMPR1的设定匹配. 0 = ADDR中的转换结果与ADCMPR1的设定不匹配.
[1]	**CMPF0**	**比较标志0** 当选择的通道A/D转换的结果与ADCMPR0寄存器的设定匹配时,这个比特将被设成“1”.这个比特写 “1”清除.. 1 = ADDR中的转换结果与ADCMPR0的设定匹配. 0 = ADDR中的转换结果与ADCMPR0的设定不匹配.
[0]	**ADF**	**A/D 转换结束标志** 一个状态标志,用来指示A/D转换结束. A/D转换结束时,ADF被设成“1”. 这个比特写 “1”清除.

5.7 FLASH 内存控制器(FMC)

5.7.1 概述

NuMicro MINI51™ 系列内嵌4K/8K/16K 字节片上FLASH EPROM,用作应用程序内存(APROM) ,可以通过ISP更新. 当芯片焊到PCB板子上以后,In System Programming (ISP) 功能使用户能更新应用程序内存. 芯片上电以后,Cortex-M0 CPU 从APROM还是LDROM取代码运行,取决于Config0中启动选项(CBS)的设置. 同时, NuMicro MINI51™ 系列也提供数据Flash区域, 数据Flash与原本的程序内存共享,开始地址可配置,由用户设置Config1来定义. 数据flash 的大小由用户根据需要来定义.

5.7.2 特性

- AHB接口兼容
- 最高跑到24 MHz,非连续地址读访问时零等待
- 4K/8K/16KB 应用程序内存 (APROM)
- 2KB in system programming (ISP) 程序装载内存(LDROM)
- 数据flash开始地址可编程,页擦除单位512字节
- n System Program (ISP) 可以更新片上Flash EPROM

5.7.3 方块图

Flash内存控制器由AHB从接口,ISP控制逻辑,写接口,flash接口时序控制逻辑组成. flash 内存控制器的方块图如下所示:

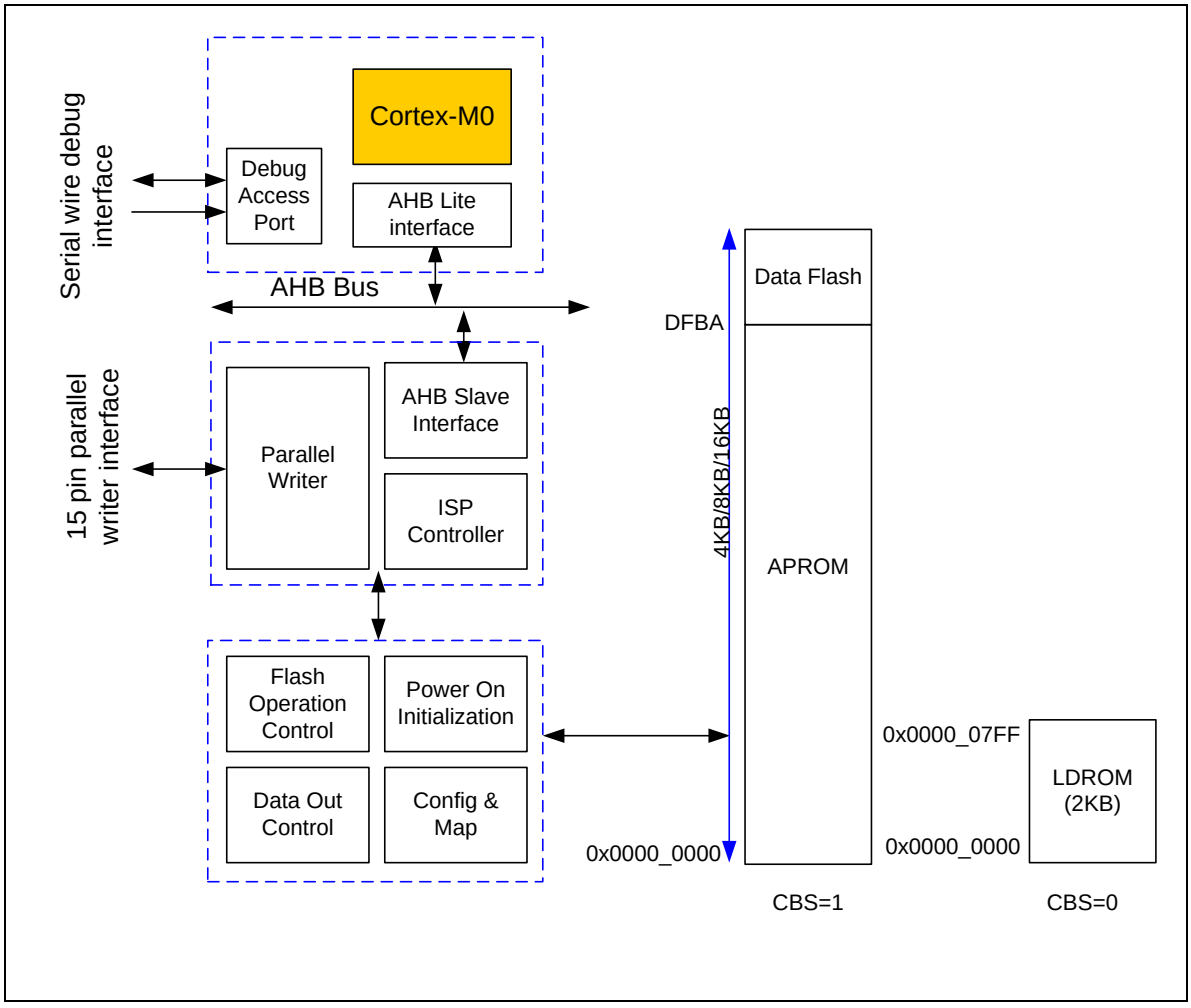


图 5.7-1 Flash 内存控制器方块图

5.7.4 功能描述

5.7.4.1 Flash 内存组织

The NuMicro MINI51™ flash 内存由编程内存 (4K/8K/16KB), 数据 flash, ISP 装载程序内存,用户配置区域组成. 用户配置区域提供几个字节控制系统逻辑,像flash安全加密,启动选择, Brownout检测电压,数据flash基地址等. 这些比特用作上电设定. 芯片上电时,从flash内存加载到相应的控制寄存器. 用户可以根据应用的需要设定这些比特. 数据flash的起始地址和大小可以由用户根据应用自己定义.

表 5.7-1 内存地址映射

Block Name	大小	起始地址	结束地址
AP-ROM	(4-0.5*N)KB / (8-0.5*N)KB / (16-0.5*N)KB	0x0000_0000	DFBA-1 (if DFEN=0)
Data Flash	0.5*N KB	DFBA	0x0000_0FFF / 0x0000_1FFF / 0x0000_3FFF
Reserved for future use		0x0000_4000	0x000F_FFFF
LD-ROM	2 KB	0x0010_0000	0x0010_07FF
用户配置空间	2 words	0x0030_0000	0x0030_0004

Flash 内存组织如下所示:

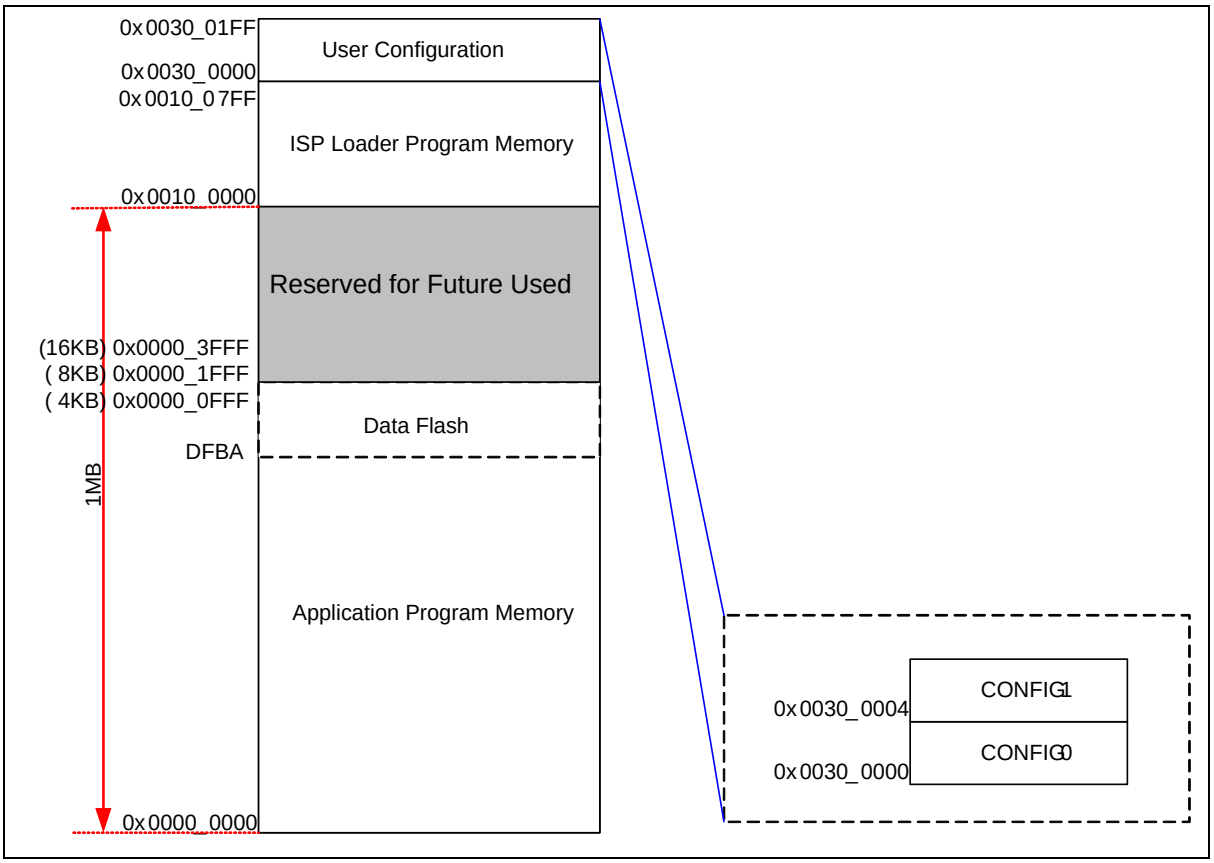


图 5.7-2 内存组织

5.7.4.2 启动选择

NuMicro MINI51™ 提供in system programming (ISP) 特性,当芯片焊到PCB板上以后,用户可以更新程序内存. 专用的2KB 程序内存用来存放ISP 固件. 通过配置Config0的CBS比特,用户能选择程序从APROM还是LDROM运行. 启动选择有两种.

表 5.7-2 启动选择表

CBS	启动选择
1	CPU 从APROM启动, flash访问范围包括APROM 和 Data Flash; 除非通过ISP否则LDROM 不能直接访问. 这种模式下APROM 不可写.
0	CPU 从LDROM启动, flash访问范围只有LDROM 2KB;除非通过ISP ,否则APROM 不能直接写. 这种模式下APROM 可写.

5.7.4.3 Data Flash

NuMicro MINI51™ 提供数据flash,使用户可以存数据. 可以通过ISP过程读/写. 擦除单位是512个字节. 当一个字需要改变时,所有128个字要提前拷贝到另一页或者SRAM中. 如果Config0的DFEN使能,数据flash的基地址由DFBA定义. 例如:对于4K/2K/1K/0KB Data Flash, DFBA设定值如下表所示.

表 5.7-3 Data Flash 表

Data Flash	4KB (DFEN=0)	2KB (DFEN=0)	1KB (DFEN=0)	0KB (DFEN=1)
16K Flash	DFBA=0x0000_3000	DFBA=0x0000_3800	DFBA=0x0000_3C00	DFEN=1
8K Flash	DFBA=0x0000_1000	DFBA=0x0000_1800	DFBA=0x0000_1C00	DFEN=1
4K Flash	Forbidden	DFBA=0x0000_0800	DFBA=0x0000_0C00	DFEN=1

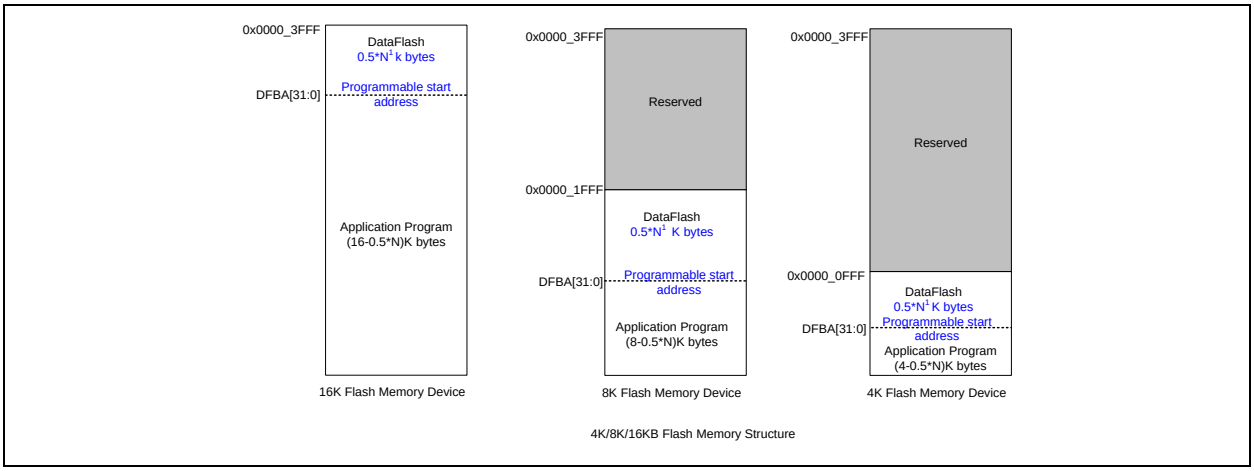


图 5.7-3 Flash 内存结构

5.7.4.4 用户配置区域

Config0 (Address = 0x0030_0000)

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

Bits	描述	
[31:29]	-	预留
[28]	CKF	**HXT/LXT 时钟过滤使能** 0 = 禁止HXT/LXT 时钟过滤. 1 = 使能HXT/LXT时钟过滤
[27:23]	-	预留
[22:21]	CBOV[1:0]	**Brownout电压选择** CBOV[1] / CBOV[0] / Brownout voltage 1 / 1 / Disable 2.7V/3.8V 1 / 0 / 3.8V 0 / 1 / 2.7V 0 / 0 / 2.7V
[20]	CBORST	**Brown Out 复位 Enable复位使能** 0 = 上电时使能brown out 复位. 1 = 上电时禁止brown out 复位.
[19:8]	-	预留
[7]	CBS	**配置启动选择** CBS = 1: CPU 从APROM启动，flash访问范围包括APROM 和 Flash; 除非通过ISP否则LDROM 不能直接访问. 这种模式下APROM 不可写.

Bits	描述		
		0	CPU 从LDROM启动, flash访问范围只有LDROM 2KB,除非通过ISP ,否则APROM 不能直接写. 这种模式下APROM 可写.
[6:2]	-	预留	
[1]	LOCK	安全锁 0 = Flash 数据被锁. 1 = Flash 数据没有被锁. 当flash 数据被锁时, 只有device ID, unique ID, Config0 和 Config1 可以被writer和ICP串行调试接口读到. 其它数据都被锁成0xFFFFFFFF. ISP 不管LOCK比特的设定,任何时候都可以读数据.	
[0]	DFEN	数据Flash 使能 0 = 使能数据 flash. 1 = 禁止数据 flash.	

Config1 (地址 = 0x0030_0004)

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

Bits	描述	
[31:18]	-	预留
[17:0]	DFBA[17:0]	**Data Flash 基地址** 数据flash 基地址由用户定义. 因为片上flash擦除单位是512个字节, 比特8-0被强制成 “0”.

例如:

表 5.7-4 Data Flash 配置

Data Flash	4KB (DFEN=0)	2KB (DFEN=0)	1KB (DFEN=0)	0KB (DFEN=1)
16K Flash	DFBA=0x0000_3000	DFBA=0x0000_3800	DFBA=0x0000_3C00	DFEN=1
8K Flash	DFBA=0x0000_1000	DFBA=0x0000_1800	DFBA=0x0000_1C00	DFEN=1
4K Flash	Forbidden	DFBA=0x0000_0800	DFBA=0x0000_0C00	DFEN=1

5.7.4.5 In System Program (ISP)

程序内存和数据flash都是硬件和in system programming (ISP)可编程的. 当产品进入量产阶段时,硬件编程模式使用gang-writers 来降低编程成本和时间. 然而, 如果产只是在开发阶段或者终端产品用户需要更新固件, 硬件编程方法就比较困难和不太方便. ISP 方法将使其变得容易和方便. NuMicro MINI51™ 支持ISP模式,允许设备在软件的控制下可以重新编程. 而且, 可以更新应用程序固件的能力使应用范围更广.

ISP不需要将微控制器从系统上解除就可以实现. 各种接口使LDROM固件取得新的程序代码很容易. 实现ISP最常见的方法是LDROM中的固件+UART. 一般来说, PC 通过串口传输新的APROM程序. LDROM 固件收到之后通过ISP命令重新编程到APROM中. Nuvoton 为NuMicro MINI51™系列提供 ISP 固件和PC 应用程序. 使用户通过Nuvoton ISP 工具可以很方便地实现ISP功能.

ISP 过程

NuMicro MINI51™ 支持从APROM还是LDROM启动,由用户配置比特(CBS)决定. 如果用户想更新APROM中的应用程序, 可以写BS=1 然后启动软件复位使芯片从LDROM启动. 第一步启动ISP功能：写ISPEN 为 “1”. 写ISPCON寄存器之前,软件需要依次写0x59, 0x16 和 0x88到解锁寄存器RegLockAddr(在GCR, 0x5000_0100中). 这个过程可以保护flash在上电/断电时防止由于无意的写操作导致数据被破坏.

软件写ISPGO比特之后,几个错误条件将被检测. 如果错误条件发生, ISP 操作不会启动并且ISP 失败标志将被设. ISPFF标志由软件清除, 下次ISP操作不会重写. 即使ISPFF为”1”,下一个ISP 过程也可以启动. 建议软件检查ISPFF比特,并在每次ISP操作之后清除它(如果被设).

当 ISPGO 比特被设时, CPU 将等待ISP 操作完成, 在此期间, 外设将保持正常工作. 如果发生任何中断, CPU等ISP过程完成之后才会处理.

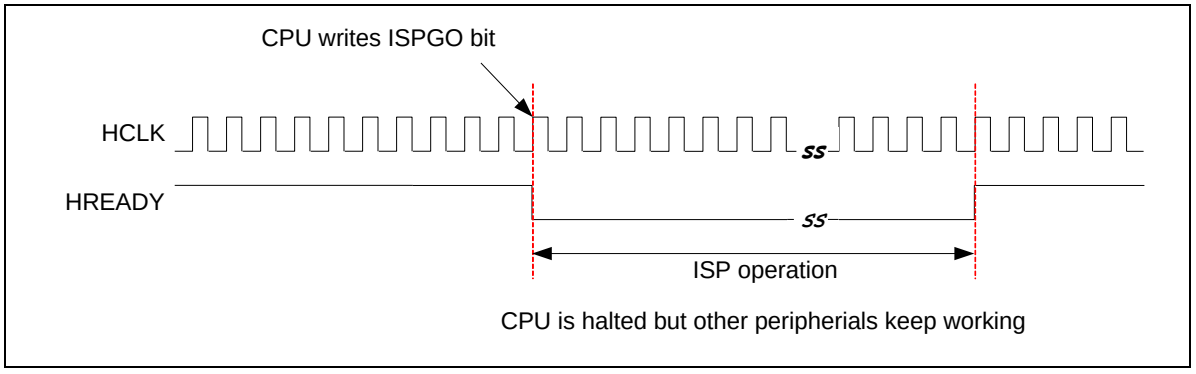


图 5.7-4 ISP 过程

注意: NuMicro MINI51™ 允许用户使用ISP更新 用户配置区域的值.

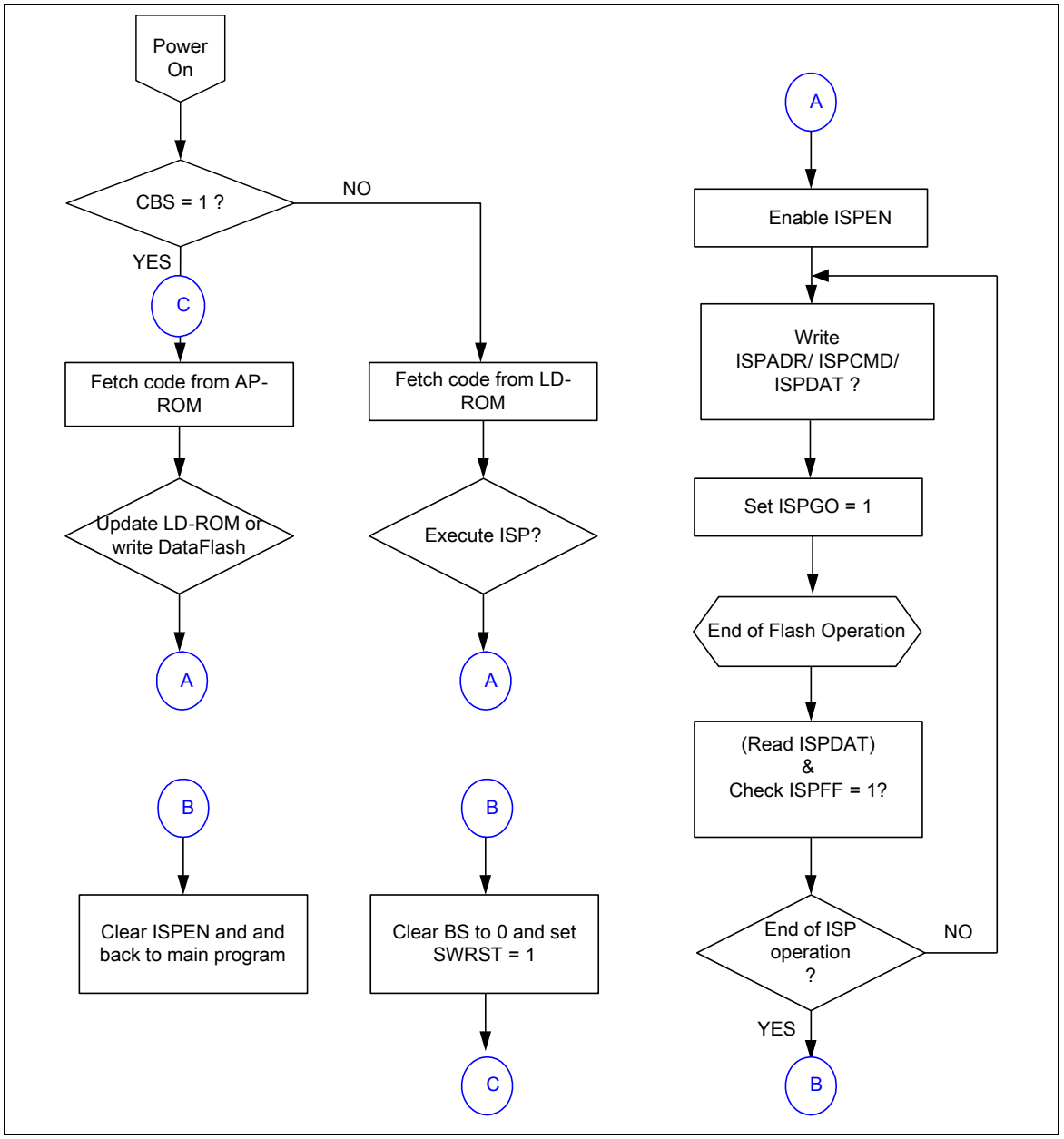


图 5.7-5 ISP 操作流程

表 5.7-5 ISP 命令表

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
Read Device ID	0	0	1100	x	x	Address in A[19:0] = 0x00000	Data out D[31:0] = Device ID
Read Unique ID	0	0	0100	x	x	Address in A[19:0] = 0x00000 0x00004 0x00008	Data out D[31:0] = Unique ID
FLASH Page Erase	1	0	0010	0	A20	Address in A[19:0]	x
FLASH Program	1	0	0001	0	A20	Address in A[19:0]	Data in D[31:0]
FLASH Read	0	0	0000	0	A20	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]

5.7.5 Flash 控制存器映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移	R/W	描述	复位值
FMC_BA = 0x5000_C000				
ISPCON	FMC_BA+0x00	R/W	ISP 控制寄存器	0x0000_0000
ISPADR	FMC_BA+0x04	R/W	ISP 地址寄存器	0x0000_0000
ISPDAT	FMC_BA+0x08	R/W	ISP 数据寄存器	0x0000_0000
ISPCMD	FMC_BA+0x0C	R/W	ISP 命令寄存器	0x0000_0000
ISPTRG	FMC_BA+0x10	R/W	ISP 触发寄存器	0x0000_0000
DFBA	FMC_BA+0x14	R	Data Flash 起始地址	0x0000_3800

5.7.6 Flash 控制寄存器描述

ISP 控制寄存器(ISPCON)

寄存器	偏移	R/W	描述	复位值
ISPCON	FMC_BA+0x00	R/W	ISP 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-	ET			-	PT		
7	6	5	4	3	2	1	0
SWRST	ISPFF	LDUEN	CFGUEN			BS	ISPEN

Bits	描述	
[31:15]	-	预留
[14:12]	ET[2:0]	Flash 擦除时间 ET[2] / ET[1] / ET[0] / 擦除时间 (ms) 0 / 0 / 0 / 20 (default) 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]	-	预留

<table>
<tr><th>Bits</th><th>描述</th><th></th></tr>
<tr><td>[10:8]</td><td>PT[2:0]</td><td>Flash 编程时间
<table>
<tr><th>PT[2]</th><th>PT[1]</th><th>PT[0]</th><th>编程时间(us)</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>40</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>45</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>50</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>55</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>20</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>25</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>30</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>35</td></tr>
</table></td></tr>
<tr><td>[7]</td><td>SWRST</td><td>软件复位
写 “1” 到该比特启动软件复位.
复位完成之后,硬件自动将该比特清0.</td></tr>
<tr><td>[6]</td><td>ISPFF</td><td>ISP 失败标志
触发ISP时遇到下列情况之一硬件将设置这个比特:
(1) APROM 写自己.
(2) LDROM 写自己.
(3) 当MCU从APROM启动时,擦除/编程用户配置区域.
(4) 目标地址非法,例如:超过有效范围.
写 “1” 清除.</td></tr>
<tr><td>[5]</td><td>LDUEN</td><td>LDROM更新使能
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</td></tr>
<tr><td>[4]</td><td>CFGUEN</td><td>配置更新使能
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</td></tr>
<tr><td>[3:2]</td><td>-</td><td>预留</td></tr>
</table>

Bits	描述	
[1]	**BS**	**启动选择** 设置/清除这个比特可以选择CPU下次从LDROM/APROM启动. 这个比特也是的MCU启动状态标志, 可以用来查看MCU从哪里启动的. 上电复位之后,这个比特由Config0的CBS比特的值取反之后初始化; 其它复位这个比特的值保持不变. 1 =从 LDROM启动. 0 =从APROM启动.
[0]	**ISPEN**	**ISP 使能** ISP 功能使能比特. 设置这个比特将使能ISP功能. 1 = 使能 ISP 功能. 0 = 禁止 ISP 功能.

ISP 地址(ISPADR)

寄存器	偏移	R/W	描述	复位值
ISPADR	FMC_BA+0x04	R/W	ISP 地址寄存器	0x0000_0000

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

Bits	描述	
[31:0]	**ISPADR[31:0]**	**ISP 地址** NuMicro MINI51™ 系列只支持字(word)编程. ISP操作时,ISPADR[1:0] 必须为00b.

ISPDAT (ISP 数据寄存器)

寄存器	偏移	R/W	描述	复位值
ISPDAT	FMC_BA+0x08	R/W	ISP 数据寄存器	0x0000_0000

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

Bits	描述	
[31:0]	ISPDAT[31:0]	**ISP 数据** ISP写操作之前将数据写到这个寄存器. ISP读操作之后,从这个寄存器读数据.

ISP 命令(ISPCMD)

寄存器	偏移	R/W	描述	复位值
ISPCMD	FMC_BA+0x0C	R/W	ISP 命令寄存器	0x0000_0000

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

Bits	描述	
[31:6]	-	**预留**
[5:0]	**FOEN, FCEN, FCTRL[3:0]**	**ISP 命令** ISP 命令.

操作模式	FOEN	FCEN	FCTRL[3:0]			
Read	0	0	0	0	0	0
Program	1	0	0	0	0	1
Page Erase	1	0	0	0	1	0
Read CID	0	0	1	0	1	1
Read DID	0	0	1	1	0	0
Read UID	0	0	0	1	0	0

ISP 触发控制寄存器(ISPTRG)

寄存器	偏移	R/W	描述	复位值
ISPTRG	FMC_BA+0x10	R/W	ISP 触发控制寄存器	0x0000_0000

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

Bits	描述	
[31:1]	-	预留
[0]	ISPGO	ISP 触发 写“1” 启动ISP操作,当ISP操作完成时,硬件将自动清除该比特. 1 = ISP 正在工作. 0 = ISP 操作已经完成.

Data Flash 基地址寄存器(DFBA)

寄存器	偏移	R/W	描述	复位值
DFBA	FMC_BA+0x14	R	Data flash 基地址	0x0000_3800

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

Bits	描述	
[31:0]	DFBA[31:0]	**Data Flash 基地址** 这个寄存器指示数据flash的起始地址.是只读的. 数据flash 的起始地址由用户定义. 因为片上flash的擦除单位是512个字节,比特8-0强制为“0”.

例如:

Data Flash	4KB (DFEN=0)	2KB (DFEN=0)	1KB (DFEN=0)	0KB (DFEN=1)
16K Flash	DFBA=0x0000_3000	DFBA=0x0000_3800	DFBA=0x0000_3C00	DFEN=1
8K Flash	DFBA=0x0000_1000	DFBA=0x0000_1800	DFBA=0x0000_1C00	DFEN=1
4K Flash	Forbidden	DFBA=0x0000_0800	DFBA=0x0000_0C00	DFEN=1

5.8 通用 I/O

5.8.1 概述

NuMicro™ Mini51共有30个通用I/O引脚，和其他的某些特定功能复用. 这30个脚分配在6个GPIO口中，分别命名为P0, P1, P2, P3, P4 和 P5. 30个引脚中的每一个都是独立的，有相应的寄存器比特控制引脚的工作模式.

每一个I/O引脚的类型是独立的，可以由软件配置成输入, 输出, 开漏, 或者准双向模式. 复位之后，所有I/O脚的缺省模式都是输入模式，数据输出寄存器Px_DOUT[n]复位成"1".对于准双向模式，每个I/O引脚内部配备一个弱上拉电阻，阻值范围大概110KΩ~300KΩ(V_{DD} 从5.0V 到2.5V).

5.8.2 特性

- 4种I/O模式:
  - 准双向
  - 推挽输出
  - 开漏
  - 高阻态输入模式
- TTL/施密特输入可选择
- 所有I/O 脚都可以配置为中断源，支持边沿/电平触发
- 支持高电流输入和高电流输出

5.8.3 功能描述

5.8.3.1 输入模式说明

设定Px_PMD (PMDn[1:0]) 为 00b, Px[n] 脚就会被设置成输入模式.I/O引脚为三态（高阻）模式，没有输出电流的能力。Px_PIN 寄存器的值反映当前相应I/O引脚的状态.

5.8.3.2 输出模式说明

设定Px_PMD (PMDn[1:0]) 为01b, Px[n] 脚就会被设置成输出模式.I/O 引脚支持数字输出并且有输入/输出电流的能力.寄存器Px_DOUT 的值将驱动到相应引脚上.

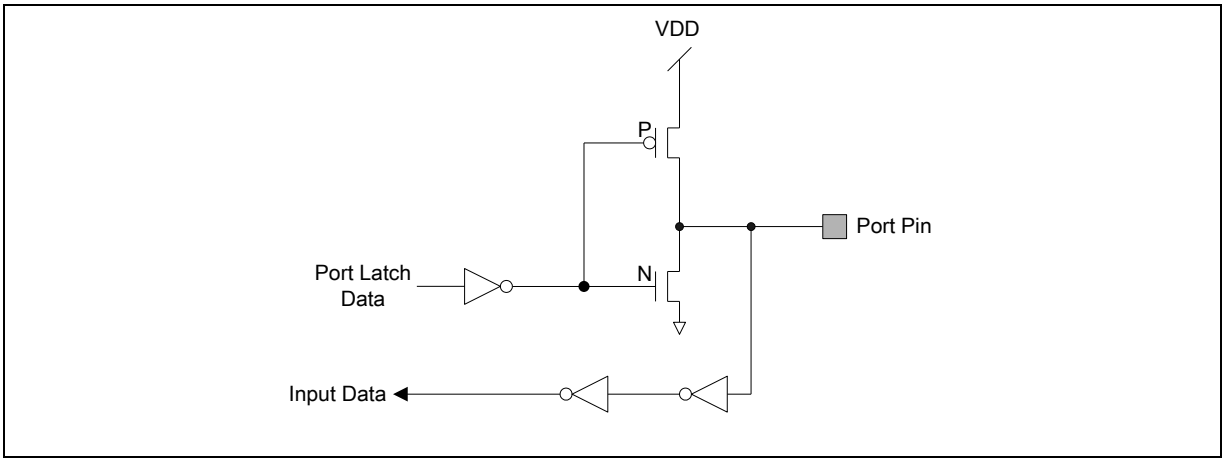


图 5.8-1 推挽输出

5.8.3.3 开漏模式说明

设定Px_PMD (PMDn[1:0])为10b, Px[n] 脚就会被设置成开漏模式.I/O引脚的数字输出功能只支持输入电流的能力, 为了输出高电平，需要额外加上拉电阻. 如果寄存器Px_DOUT 的某个比特[n]是”0”, 引脚上将输出低电平；如果寄存器Px_DOUT 的某个比特[n]是”1”,引脚将由内部或者外部上拉电阻控制输出高电平.

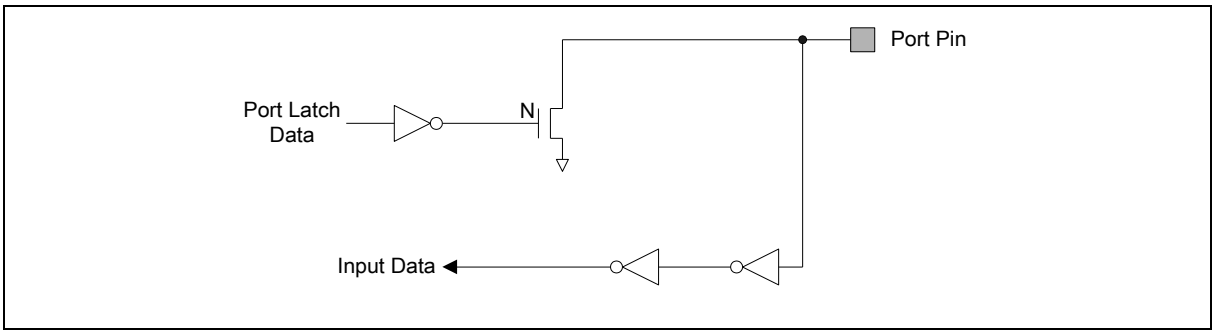


图 5.8-2 开漏输出

5.8.3.4 准双向模式说明

设定Px_PMD (PMDn[1:0]) 为 “11b”, Px[n] 脚就会被设置成准双向模式.I/O 引脚同时支持数字输出和输入功能，但是输出电流的能力只有几百uA. 使用数字输入功能之前，Px_DOUT寄存器相应位必须被设成”1”. 准双向输出模式在80C51和其大多衍生产品上是很常见的. 如果寄存器Px_DOUT 的某个比特[n]是”0”,引脚上将输出低电平；如果寄存器Px_DOUT 的某个比特[n]是”1”,相应引脚将检测自身状态值，如果引脚状态值是高，不采取任何动作.如果引脚状态值是低，将驱动2个时钟周期的高电流输出，然后关闭高电流输出，此后引脚的状态由内部上拉电阻控制. 注意，准双向模式下驱动电流的能力几乎只有200uA到30uA(VDD 从5.0 V to 2.5 V).

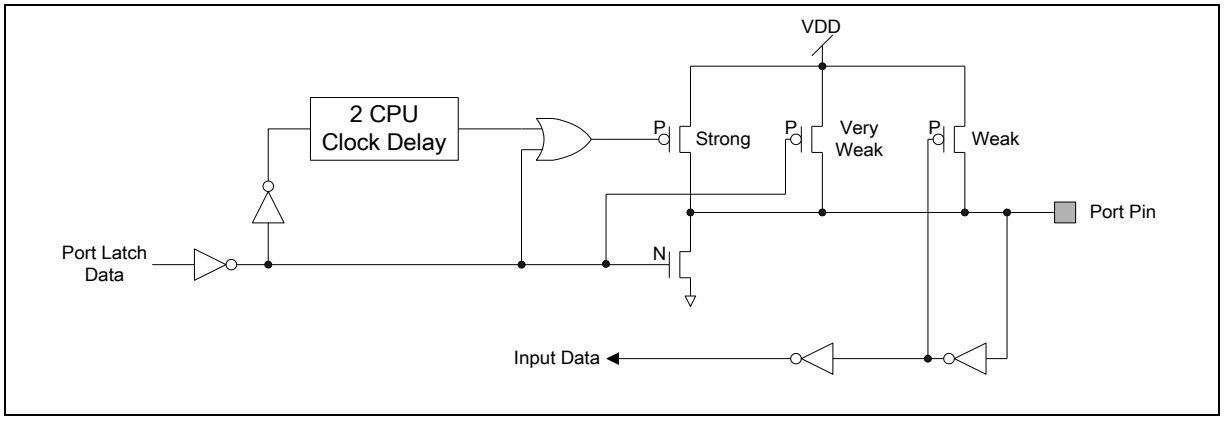


图 5.8-3 准双向 I/O 模式

5.8.4 Port 0-5 控制寄存器映射

R: 只读, **W:** 只写, **R/W:** 可读写

寄存器	偏移	R/W	描述	复位值
GP_BA = 0x5000_4000				
P0_PMD	GP_BA+0x000	R/W	P0 端口I/O模式控制	0x0000_0000
P0_OFFD	GP_BA+0x004	R/W	P0 端口断开数字通路使能	0x0000_0000
P0_DOUT	GP_BA+0x008	R/W	P0 端口数据输出寄存器	0x0000_00F3
P0_DMASK	GP_BA+0x00C	R/W	P0 端口数据输出寄存器写屏蔽	0x0000_0000
P0_PIN	GP_BA+0x010	R	P0 端口引脚状态	0x0000_00XX
P0_DBEN	GP_BA+0x014	R/W	P0 端口De-bounce 使能	0x0000_0000
P0_IMD	GP_BA+0x018	R/W	P0 端口中断模式控制	0x0000_0000
P0_IEN	GP_BA+0x01C	R/W	P0 端口中断使能寄存器	0x0000_0000
P0_ISRC	GP_BA+0x020	R/W	P0 端口中断触发源指示	0x0000_0000
P1_PMD	GP_BA+0x040	R/W	P1端口I/O模式控制	0x0000_0000
P1_OFFD	GP_BA+0x044	R/W	P1端口断开数字通路使能	0x0000_0000
P1_DOUT	GP_BA+0x048	R/W	P1端口数据输出寄存器	0x0000_003D
P1_DMASK	GP_BA+0x04C	R/W	P1端口数据输出寄存器写屏蔽	0x0000_0000
P1_PIN	GP_BA+0x050	R	P1端口引脚状态	0x0000_00XX
P1_DBEN	GP_BA+0x054	R/W	P1端口 De-bounce 使能寄存器	0x0000_0000
P1_IMD	GP_BA+0x058	R/W	P1端口中断模式控制寄存器	0x0000_0000
P1_IEN	GP_BA+0x05C	R/W	P1端口中断使能寄存器	0x0000_0000
P1_ISRC	GP_BA+0x060	R/WC	P1端口中断触发源指示寄存器	0x0000_0000
P2_PMD	GP_BA+0x080	R/W	P2端口I/O模式控制	0x0000_0000
P2_OFFD	GP_BA+0x084	R/W	P2端口断开数字通路使能	0x0000_0000
P2_DOUT	GP_BA+0x088	R/W	P2端口数据输出寄存器	0x0000_007C
P2_DMASK	GP_BA+0x08C	R/W	P2端口数据输出寄存器写屏蔽	0x0000_0000
P2_PIN	GP_BA+0x090	R	P2端口引脚状态	0x0000_00XX
P2_DBEN	GP_BA+0x094	R/W	P2 端口De-bounce 使能寄存器	0x0000_0000
P2_IMD	GP_BA+0x098	R/W	P2端口中断模式控制寄存器	0x0000_0000

寄存器	偏移	R/W	描述	复位值
GP_BA = 0x5000_4000				
P2_IEN	GP_BA+0x09C	R/W	P端口2中断使能寄存器	0x0000_0000
P2_ISRC	GP_BA+0x0A0	R/W	P2端口中断触发源指示寄存器	0x0000_0000
P3_PMD	GP_BA+0x0C0	R/W	P3端口I/O模式控制	0x0000_0000
P3_OFFD	GP_BA+0x0C4	R/W	P3端口断开数字通路使能	0x0000_0000
P3_DOUT	GP_BA+0x0C8	R/W	P3端口数据输出寄存器	0x0000_0077
P3_DMASK	GP_BA+0x0CC	R/W	P3端口数据输出寄存器写屏蔽	0x0000_0000
P3_PIN	GP_BA+0x0D0	R	P3端口引脚状态	0x0000_00XX
P3_DBEN	GP_BA+0x0D4	R/W	P3 端口De-bounce 使能寄存器	0x0000_0000
P3_IMD	GP_BA+0x0D8	R/W	P3端口中断模式控制寄存器	0x0000_0000
P3_IEN	GP_BA+0x0DC	R/W	P3端口中断使能寄存器	0x0000_0000
P3_ISRC	GP_BA+0x0E0	R/W	P3端口中断触发源指示寄存器	0x0000_0000
P4_PMD	GP_BA+0x100	R/W	P4 端口I/O模式控制	0x0000_0000
P4_OFFD	GP_BA+0x104	R/W	P4端口断开数字通路使能	0x0000_0000
P4_DOUT	GP_BA+0x108	R/W	P4端口数据输出寄存器	0x0000_00C0
P4_DMASK	GP_BA+0x10C	R/W	P4端口数据输出寄存器写屏蔽	0x0000_0000
P4_PIN	GP_BA+0x110	R	P4端口引脚状态	0x0000_00XX
P4_DBEN	GP_BA+0x114	R/W	P4 端口De-bounce 使能寄存器	0x0000_0000
P4_IMD	GP_BA+0x118	R/W	P4端口中断模式控制寄存器	0x0000_0000
P4_IEN	GP_BA+0x11C	R/W	P4端口中断使能寄存器	0x0000_0000
P4_ISRC	GP_BA+0x120	R/W	P4端口中断触发源指示寄存器	0x0000_0000
P5_PMD	GP_BA+0x140	R/W	P5 端口I/O模式控制	0x0000_0000
P5_OFFD	GP_BA+0x144	R/W	P5 端口断开数字通路使能	0x0000_0000
P5_DOUT	GP_BA+0x148	R/W	P5端口数据输出寄存器	0x0000_003F
P5_DMASK	GP_BA+0x14C	R/W	P端口5数据输出寄存器写屏蔽	0x0000_0000
P5_PIN	GP_BA+0x150	R	P5端口引脚状态	0x0000_00XX
P5_DBEN	GP_BA+0x154	R/W	P5 端口De-bounce 使能寄存器	0x0000_0000
P5_IMD	GP_BA+0x158	R/W	P5端口中断模式控制寄存器	0x0000_0000
P5_IEN	GP_BA+0x15C	R/W	P5端口中断使能寄存器	0x0000_0000

寄存器	偏移	R/W	描述	复位值
GP_BA = 0x5000_4000				
P5_ISRC	GP_BA+0x160	R/W	P5端口中断触发源指示寄存器	0x0000_0000
DBNCECON	GP_BA+0x180	R/W	中断De-bounce周期 控制	0x0000_0020
P00_DOUT	GP_BA+0x200	R/W	P0.0数据输出	0x0000_0001
P01_DOUT	GP_BA+0x204	R/W	P0.1数据输出	0x0000_0001
P04_DOUT	GP_BA+0x210	R/W	P0.4数据输出	0x0000_0001
P05_DOUT	GP_BA+0x214	R/W	P0.5数据输出	0x0000_0001
P06_DOUT	GP_BA+0x218	R/W	P0.6数据输出	0x0000_0001
P07_DOUT	GP_BA+0x21C	R/W	P0.7数据输出	0x0000_0001
P10_DOUT	GP_BA+0x220	R/W	P1.0数据输出	0x0000_0001
P12_DOUT	GP_BA+0x228	R/W	P1.2数据输出	0x0000_0001
P13_DOUT	GP_BA+0x22C	R/W	P1.3数据输出	0x0000_0001
P14_DOUT	GP_BA+0x230	R/W	P1.4数据输出	0x0000_0001
P15_DOUT	GP_BA+0x234	R/W	P1.5数据输出	0x0000_0001
P22_DOUT	GP_BA+0x248	R/W	P2.2数据输出	0x0000_0001
P23_DOUT	GP_BA+0x24C	R/W	P2.3数据输出	0x0000_0001
P24_DOUT	GP_BA+0x250	R/W	P2.4数据输出	0x0000_0001
P25_DOUT	GP_BA+0x254	R/W	P2.5数据输出	0x0000_0001
P26_DOUT	GP_BA+0x258	R/W	P2.6数据输出	0x0000_0001
P30_DOUT	GP_BA+0x260	R/W	P3.0数据输出	0x0000_0001
P31_DOUT	GP_BA+0x264	R/W	P3.1数据输出	0x0000_0001
P32_DOUT	GP_BA+0x268	R/W	P3.2数据输出	0x0000_0001
P34_DOUT	GP_BA+0x270	R/W	P3.4数据输出	0x0000_0001
P35_DOUT	GP_BA+0x274	R/W	P3.5数据输出	0x0000_0001
P36_DOUT	GP_BA+0x278	R/W	P3.6数据输出	0x0000_0001
P46_DOUT	GP_BA+0x298	R/W	P4.6数据输出	0x0000_0001
P47_DOUT	GP_BA+0x29C	R/W	P4.7数据输出	0x0000_0001
P50_DOUT	GP_BA+0x2A0	R/W	P5.0数据输出	0x0000_0001

寄存器	偏移	R/W	描述	复位值
GP_BA = 0x5000_4000				
P51_DOUT	GP_BA+0x2A4	R/W	P5.1数据输出	0x0000_0001
P52_DOUT	GP_BA+0x2A8	R/W	P5.2数据输出	0x0000_0001
P53_DOUT	GP_BA+0x2AC	R/W	P5.3数据输出	0x0000_0001
P54_DOUT	GP_BA+0x2B0	R/W	P5.4数据输出	0x0000_0001
P55_DOUT	GP_BA+0x2B4	R/W	P5.5数据输出	0x0000_0001

注意: 在QFN-33封装，为了使功耗最小，软件必须设定un-bonding 输出引脚P5.5 成输出模式.

5.8.5 Port 0-5 控制寄存器描述

Port 0-5 端口I/O引脚模式控制(Px_PMD)

寄存器	偏移	R/W	描述	复位值
P0_PMD	GP_BA+0x000	R/W	P0 端口I/O 模式控制	0x0000_0000
P1_PMD	GP_BA+0x040	R/W	P1端口I/O 模式控制	0x0000_0000
P2_PMD	GP_BA+0x080	R/W	P2端口I/O 模式控制	0x0000_0000
P3_PMD	GP_BA+0x0C0	R/W	P3端口I/O 模式控制	0x0000_0000
P4_PMD	GP_BA+0x100	R/W	P4端口I/O 模式控制	0x0000_0000
P5_PMD	GP_BA+0x140	R/W	P5端口I/O 模式控制	0x0000_0000

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

Bits	描述	
[31:16]	-	**预留**
[2n+1:2n]	**PMDn**	**Px Pin[n] I/O模式 控制** 决定每个I/O 引脚的类型. **PMDn[1:0]** / **Pin I/O 模式 (x = 0~5, n = 0~7)** **00** Px pin[n] 是输入模式. **01** Px pin[n] 是输出模式. **10** Px pin[n] 是开漏模式. **11** Px pin[n] 是准双向模式.
注意: P0_PMD[7:4] 预留.		

Bits	描述
P1_PMD[15:12], [3:2] 预留. P2_PMD[15:14], [3:0] 预留. P3_PMD[15:14], [7:6] 预留. P4_PMD[11:0] 预留. P5_PMD[15:12] 预留.	

Port 0-5 端口断开数字通路使能(Px_OFFD)

寄存器	偏移	R/W	描述	复位值
P0_OFFD	GP_BA+0x004	R/W	P0端口断开数字通路使能	0x0000_0000
P1_OFFD	GP_BA+0x044	R/W	P1端口断开数字通路使能	0x0000_0000
P2_OFFD	GP_BA+0x084	R/W	P2端口断开数字通路使能	0x0000_0000
P3_OFFD	GP_BA+0x0C4	R/W	P3端口断开数字通路使能	0x0000_0000
P4_OFFD	GP_BA+0x104	R/W	P4端口断开数字通路使能	0x0000_0000
P5_OFFD	GP_BA+0x144	R/W	P5端口断开数字通路使能	0x0000_0000

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

Bits	描述	
[31:24]	-	**预留**
[23:16]	**OFFD[n]**	**OFFD: Px Pin[n] 断开数字通路使能** 1 = 禁止IO 数字输入通路 (数字输入固定在低电平). 0 = 使能IO 数字输入通路. x = 0~5, n = 0~7
[15:0]	-	**预留**
注意: P0_OFFD[19:18] 预留. P1_OFFD[23:22], [17] 预留. P2_OFFD[23], [17:16] 预留. P3_OFFD[23], [19] 预留. P4_OFFD[21:16] 预留. P5_OFFD[23:22] 预留.		

Port 0-5 端口数据输出寄存器(Px_DOUT)

寄存器	偏移	R/W	描述	复位值
P0_DOUT	GP_BA+0x008	R/W	P0端口数据输出寄存器	0x0000_00F3
P1_DOUT	GP_BA+0x048	R/W	P1 端口数据输出寄存器	0x0000_003D
P2_DOUT	GP_BA+0x088	R/W	P2端口 数据输出寄存器	0x0000_007C
P3_DOUT	GP_BA+0x0C8	R/W	P3 端口数据输出寄存器	0x0000_0077
P4_DOUT	GP_BA+0x108	R/W	P4 端口数据输出寄存器	0x0000_00C0
P5_DOUT	GP_BA+0x148	R/W	P5端口 数据输出寄存器	0x0000_003F

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

Bits	描述	
[31:8]	**-**	**预留**
[n]	**DOUT[n]**	**Px Pin[n] 输出值** 当Px.Pin[n]引脚配置成输出，开漏或准双向模式时，这些位控制Px.Pin[n]引脚的状态 1 = 如果被配置成输出模式,Px.Pin[n] 将驱动输出高电平. 0 = Px Pin[n] 将输出低电平. x = 0~5, n = 0~7
注意: P0_DOUT[3:2] 预留. P1_DOUT [7:6], [1] 预留. P2_DOUT [7], [1:0] 预留. P3_DOUT [7], [3] 预留. P4_DOUT [5:0] 预留. P5_DOUT [7:6] 预留.		

Port 0-5 端口数据输出写掩码寄存器(Px_DMASK)

寄存器	偏移	R/W	描述	复位值
P0_DMASK	GP_BA+0x00C	R/W	P0 端口数据输出写屏蔽	0x0000_0000
P1_DMASK	GP_BA+0x04C	R/W	P1端口数据输出写屏蔽	0x0000_0000
P2_DMASK	GP_BA+0x08C	R/W	P2端口数据输出写屏蔽	0x0000_0000
P3_DMASK	GP_BA+0x0CC	R/W	P3端口数据输出写屏蔽	0x0000_0000
P4_DMASK	GP_BA+0x10C	R/W	P4端口数据输出写屏蔽	0x0000_0000
P5_DMASK	GP_BA+0x14C	R/W	P5端口数据输出写屏蔽	0x0000_0000

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

Bits	描述	
[31:8]	**-**	**预留**
[n]	**DMASK[n]**	**Px 输出写屏蔽** 这些位可以用来保护相应的Px_DOUT pin[n]位.当DMASK bit[n]为"1"的时候，相应的DOUTn引脚被写保护，写Px_DOUT pin[n]位不起作用. 0 = 相应的Px_DOUT[n] 位可以被更新. 1 = 相应的Px_DOUT[n] 位被保护. x = 0~5, n = 0~7
注意: P0_DMASK[3:2] 预留. P1_DMASK [7:6], [1] 预留. P2_DMASK [7], [1:0] 预留. P3_DMASK [7], [3] 预留. P4_DMASK [5:0] 预留. P5_DMASK [7:6] 预留.		

Port 0-5 端口引脚值寄存器e (Px_PIN)

寄存器	偏移	R/W	描述	复位值
P0_PIN	GP_BA+0x010	R	P0 端口引脚值	0x0000_00XX
P1_PIN	GP_BA+0x050	R	P1端口引脚值	0x0000_00XX
P2_PIN	GP_BA+0x090	R	P2 端口引脚值	0x0000_00XX
P3_PIN	GP_BA+0x0D0	R	P3 端口引脚值	0x0000_00XX
P4_PIN	GP_BA+0x110	R	P4 端口引脚值	0x0000_00XX
P5_PIN	GP_BA+0x150	R	P5 端口引脚值	0x0000_00XX

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

Bits	描述	
[31:8]	**-**	**预留**
[n]	**PIN[n]**	**Px Pin[n] 引脚值** 每个比特的值都反映了相应的 Px pin[n] 的状态. (x = 0~5, n = 0~7)
注意: P0_PIN[3:2] 预留. P1_PIN [7:6], [1] 预留. P2_PIN [7], [1:0] 预留. P3_PIN [7], [3] 预留. P4_PIN [5:0] 预留. P5_PIN [7:6] 预留.		

Port 0-5 端口De-bounce使能寄存器(Px_DBEN)

寄存器	偏移	R/W	描述	复位值
P0_DBEN	GP_BA+0x014	R/W	P0 端口De-bounce 使能	0x0000_0000
P1_DBEN	GP_BA+0x054	R/W	P1 端口De-bounce 使能	0x0000_0000
P2_DBEN	GP_BA+0x094	R/W	P2端口 De-bounce 使能	0x0000_0000
P3_DBEN	GP_BA+0x0D4	R/W	P3 端口De-bounce 使能	0x0000_0000
P4_DBEN	GP_BA+0x114	R/W	P4端口 De-bounce 使能	0x0000_0000
P5_DBEN	GP_BA+0x154	R/W	P5 端口De-bounce 使能	0x0000_0000

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

Bits	描述	
[31:8]	-	**预留**
[n]	**DBEN[n]**	**Px 输入信号De-bounce 使能** DBEN[n] 用来使能相应引脚的de-bounce 功能. 如果输入信号脉冲宽度不能被连续两个de-boune采样周期被采样，输入信号的转变就会被当作信号bounce，而不会触发中断. DBEN[n] 只用在“边沿触发” 中断, 对“电平触发” 中断不起作用. 0 = 禁止pin[n] de-bounce功能. 1 = 使能pin[n] de-bounce功能. de-bounce 功能只有在边沿触发中断才有效.如果中断模式是电平触发, de-bounce 功能被忽略. x = 0~5, n = 0~7

Bits	描述
注意:	
P0_DBEN[3:2] 预留.	
P1_DBEN [7:6], [1] 预留.	
P2_DBEN [7], [1:0] 预留.	
P3_DBEN [7], [3] 预留.	
P4_DBEN [5:0] 预留.	
P5_DBEN [7:6] 预留.	

Port 0-5 端口中断模式控制寄存器(Px_IMD)

寄存器	偏移	R/W	描述	复位值
P0_IMD	GP_BA+0x018	R/W	P0端口中断模式控制	0x0000_0000
P1_IMD	GP_BA+0x058	R/W	P1端口中断模式控制	0x0000_0000
P2_IMD	GP_BA+0x098	R/W	P2端口中断模式控制	0x0000_0000
P3_IMD	GP_BA+0x0D8	R/W	P3端口中断模式控制	0x0000_0000
P4_IMD	GP_BA+0x118	R/W	P4端口中断模式控制	0x0000_0000
P5_IMD	GP_BA+0x158	R/W	P5端口中断模式控制	0x0000_0000

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

Bits	描述	
[31:8]	-	**预留**
[n]	**IMD[n]**	**Port 0-5 中断模式控制** IMD[n] 用来控制中断是边沿触发还是电平触发.如果中断是边沿触发,DEBN寄存器将用于控制输入信号的de-bounce. 如果中断是电平触发，输入源在一个时钟内被采样到，就将触发中断. 0 = 边沿触发中断. 1 = 电平触发中断. 如果设定引脚中断为电平触发，则寄存器Px_IEN中只有一种电平可以设定. 如果设定两个高/低电平都触发中断，设定无效，将没有中断发生. de-bounce 功能只在边沿触发时才有效. 如果中断触发模式是电平触发，de-bounce使能位将被忽略. x = 0~5, n = 0~7

Bits	描述
注意:	
P0_IMD[3:2] 预留.	
P1_IMD [7:6], [1] 预留.	
P2_IMD [7], [1:0] 预留.	
P3_IMD [7], [3] 预留.	
P4_IMD [5:0] 预留.	
P5_IMD [7:6] 预留.	

Port 0-5 端口中断使能寄存器(Px_IEN)

寄存器	偏移	R/W	描述	复位值
P0_IEN	GP_BA+0x01C	R/W	P0 端口中断使能	0x0000_0000
P1_IEN	GP_BA+0x05C	R/W	P1端口中断使能	0x0000_0000
P2_IEN	GP_BA+0x09C	R/W	P2端口中断使能	0x0000_0000
P3_IEN	GP_BA+0x0DC	R/W	P3端口中断使能	0x0000_0000
P4_IEN	GP_BA+0x11C	R/W	P4端口中断使能	0x0000_0000
P5_IEN	GP_BA+0x15C	R/W	P5端口中断使能	0x0000_0000

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

Bits	描述	
[31:24]	-	**预留**
[n+16]	**IR_EN[n]**	**Port 0-5中断使能，同时设定是上升沿还是高电平触发** IR_EN[n] 用来使能相应Px[n]输入脚的中断. 设为 “1” 也同时使能唤醒CPU的功能. 当设定IR_EN[n] 比特为 “1”时: 如果中断是电平触发模式, Px[n]输入“高”电平将触发中断. 如果中断是边沿触发模式, Px[n] 输入状态从“低到高” 的上升沿将触发中断. 1 = 使能Px[n] 高电平或者低到高上升沿触发中断. 0 = 禁止Px[n] 高电平或者低到高上升沿触发中断. x = 0~5, n = 0~7
[15:8]	-	**预留**

Bits	描述	
[n]	**IF_EN[n]**	**Port 0-5中断使能，同时设定是下降沿还是低电平触发** IF_EN[n] 用来使能相应Px[n]输入引脚的中断. 设为 “1” 也同时使能唤醒CPU的功能. 当设定IF_EB[n] 比特为“1”时: 如果中断是电平触发模式, 输入“低”电平将触发中断. 如果中断是边沿触发模式, 输入状态从“高到低” 的下降沿将触发中断. 1 = 使能 Px[n] 低电平或者高到低的下降沿触发中断. 0 = 禁止 Px[n] 低电平或者高到低的下降沿触发中断. x = 0~5, n = 0~7
注意: P0_IEN[19:18], [3:2] 预留. P1_IEN [23:22], [17], [7:6], [1] 预留. P2_IEN [23], [17:16], [7], [1:0] 预留. P3_IEN [23], [19], [7], [3] 预留. P4_IEN [21:16], [5:0] 预留. P5_IEN [23:22], [7:6] 预留.		

Port 0-5端口中断触发源指示寄存器(Px_ISRC)

寄存器	偏移	R/W	描述	复位值
P0_ISRC	GP_BA+0x020	R/W	P0端口中断触发源指示	0x0000_0000
P1_ISRC	GP_BA+0x060	R/W	P1端口中断触发源指示	0x0000_0000
P2_ISRC	GP_BA+0x0A0	R/W	P2端口中断触发源指示	0x0000_0000
P3_ISRC	GP_BA+0x0E0	R/W	P3端口中断触发源指示	0x0000_0000
P4_ISRC	GP_BA+0x120	R/W	P4端口中断触发源指示	0x0000_0000
P5_ISRC	GP_BA+0x160	R/W	P5端口中断触发源指示	0x0000_0000

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

Bits	描述	
[31:8]	-	预留
[n]	ISRC[n]	**Port 0-5中断触发源指示** 读 : 1 = 指示 Px[n] 产生了中断. 0 = Px[n]没有产生中断. 写 : 1 = 清除相应的中断标志. 0 = 不起作用. x = 0~5, n = 0~7

Bits	描述
注意:	
P0_ISRC [3:2] 预留.	
P1_ISRC [7:6], [1] 预留.	
P2_ISRC [7], [1:0] 预留.	
P3_ISRC [7], [3] 预留.	
P4_ISRC [5:0] 预留.	
P5_ISRC [7:6] 预留.	

中断De-bounce 周期控制寄存器(DBNCECON)

寄存器	偏移	R/W	描述	复位值
DBNCECON	GP_BA+0x180	R/W	外部中断 De-bounce 周期控制	0x0000_0020

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

Bits	描述	
[31:6]	-	预留
[5]	ICLK_ON	中断时钟使能位 如果pin[n]的中断是禁止的，设置该比特为“0” 将关闭中断发生电路的时钟. 0 = 如果P0/1/2/3/4[n]中断是禁止的，关闭时钟. 1 =中断发生器的时钟总是使能的. n = 0~7
[4]	DBCLKSRC	De-bounce 计数器时钟源选择 1 = De-bounce 计数器时钟源是内部10 KHz. 0 = De-bounce 计数器时钟源是HCLK.

Bits	描述	
[3:0]	**DBCLKSEL [3:0]**	**De-bounce 采样周期选择** DBCLKSEL / 描述 0 每个时钟周期采样一次中断输入 1 每2个时钟周期采样一次中断输入 2 每4个时钟周期采样一次中断输入 3 每8个时钟周期采样一次中断输入 4 每16个时钟周期采样一次中断输入 5 每32个时钟周期采样一次中断输入 6 每64个时钟周期采样一次中断输入 7 每128个时钟周期采样一次中断输入 8 每256个时钟周期采样一次中断输入 9 每2*256个时钟周期采样一次中断输入 10 每4*256个时钟周期采样一次中断输入 11 每8*256个时钟周期采样一次中断输入 12 每16*256个时钟周期采样一次中断输入 13 每32*256个时钟周期采样一次中断输入 14 每64*256个时钟周期采样一次中断输入 15 每128*256个时钟周期采样一次中断输入

GPIO Port [P0/P1/P2/P3/P4/P5] I/O 位输出控制寄存器(P[x][n]_DOUT)

P[x][n]_DOUT: x = 0~5, n = 0~7

寄存器	偏移	R/W	描述	复位值
P0[n]_DOUT	GP_BA+0x200 - GP_BA+0x21C	R/W	P0 Pin I/O 比特输入/输出控制. For P0, n=0,1,4,5,6,7	0x0000_0001
P1[n]_DOUT	GP_BA+0x220 - GP_BA+0x23C	R/W	P1 Pin I/O 比特输入/输出控制. For P1, n=0,2,3,4,5	0x0000_0001
P2[n]_DOUT	GP_BA+0x240 - GP_BA+0x25C	R/W	P2 Pin I/O 比特输入/输出控制. For P2, n=2,3,4,5,6	0x0000_0001
P3[n]_DOUT	GP_BA+0x260 - GP_BA+0x27C	R/W	P3 Pin I/O 比特输入/输出控制I. For P3, n=0,1,2,4,5,6	0x0000_0001
P4[n]_DOUT	GP_BA+0x280 - GP_BA+0x29C	R/W	P4 Pin I/O 比特输入/输出控制. For P4, n=6,7	0x0000_0001
P5[n]_DOUT	GP_BA+0x2A0 - GP_BA+0x2B4	R/W	P5 Pin I/O 比特输入/输出控制. For P5, n=0,1,2,3,4,5	0x0000_0001

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

Bits	描述	
[31:1]	-	**预留**

[0]	**P[x][n]_DOUT**	**$P_{[x][n]}$ I/O 脚比特输入/输出控制** 写该比特可以控制一个 GPIO 引脚的输出值. 1 = 设定相应的GPIO脚输出高电平. 0 =设定相应的GPIO脚输出低电平. 读这个寄存器可以得到IO引脚的状态. 1 = 相应GPIO脚的状态为高电平. 0 =相应GPIO脚的状态为低电平. 例如: 写 P01_DOUT[0] 将反映写的值到P0.1引脚, 读P01_DOUT[0]将返回P0.1引脚的值. x = 0~5, n = 0~7

5.9 I²C 串行接口控制器(主/从)

5.9.1 概述

I²C 是一个两线双向串行总线，为设备间数据交互提供简单和有效的方法. I²C 标准是一个多主总线，包含冲突检测和仲裁，两个或者多个主设备同时发送数据的时候，可以防止数据遭到破坏. 串行, 8-bit 双向数据传输，速度可达1.0 Mbps.

基于SCL上的时钟，数据在主设备和从设备之间一个字节一个字节的同步传输. 每个字节8个比特长. 每个比特有一个SCL时钟脉冲，MSB优先发送. 每传输完一个字节将收到一个应答比特.每个比特在SCL的高电平被采样; 因而, SDA 线只有在SCL处于低电平的时候才可以改变状态，在SCL为高电平的时候必须保持稳定. SCL为高电平时如果SDA上的状态改变将被解释为一个命令 (START或者 STOP). I2C总线的时序细节请参考下面的图.

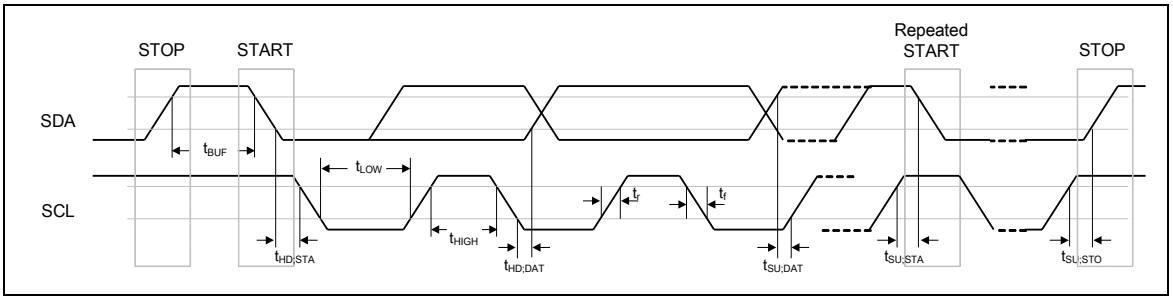


图 5.9-1 总线时序

芯片内的I2C逻辑提供串行接口，符合I2C总线的标准. 片内的I²C由硬件实现，自动处理字节传输. 为了使能I2C接口, I2CON寄存器的ENSI比特必须设成"1". I2C硬件接口有两根引脚: SDA (P3.4, 串行数据线) 和SCL (P3.5, 串行时钟线). 当设为开漏模式时，引脚P3.4 和 P3.5需要上拉电阻. 这两根I/O脚用作I2C时，需要先将其设定为I2C功能.

5.9.2 特性

I²C总线使用两根线(SDA 和 SCL) 在设备间传输数据. 主要特性有:

- 支持主/从模式
- 主从设备之间双向数据传输
- 多主总线(无核心主设备)
- 多个主设备同时传输时仲裁，防止总线上的数据遭到破坏
- 串行时钟同步允许同一个总线上的设备有不同的速率
- 串行时钟同步可以用作一个握手信号挂起和重新启动传输
- 内嵌一个14-bit 超时计数器，如果I2C总线挂起，导致超时计数器溢出，超时中断将发生
- 更快的输出上拉速度需要加外部上拉电阻
- 时钟可编程，允许更丰富的波特率控制

- 支持 7-bit 寻址模式
- I²C总线控制器支持多地址识别(4个从地址，带掩码功能)

5.9.3 I²C 协议

正常情况下, 标准的通讯包括4部分:

1) 产生START 或者 Repeated START 信号
2) 传输从设备地址和R/W比特
3) 传输数据
4) 产生STOP 信号

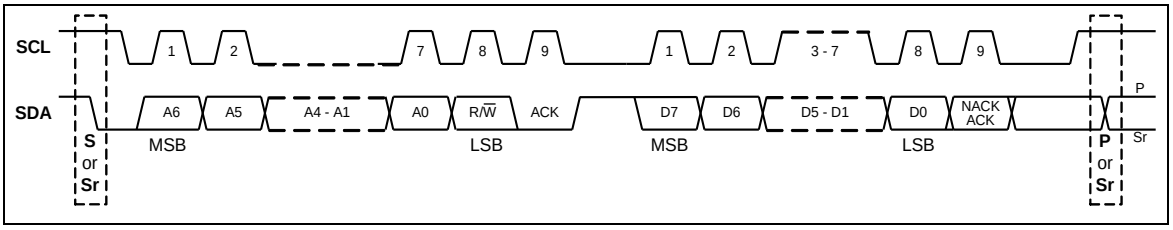


图 5.9-2 I²C 协议

5.9.3.1 I²C总线数据传输

主设备作为发送方用7比特地址寻址作为接收方的从设备.

传输方向不变.

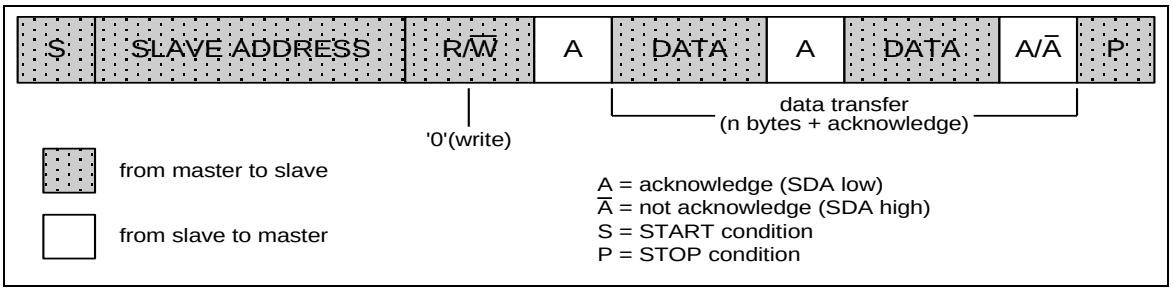


图 5.9-3 主设备发送数据到从设备

发送第一个字节(地址)之后，主设备立即改变传输方向，改为读从设备.

传输方向改变.

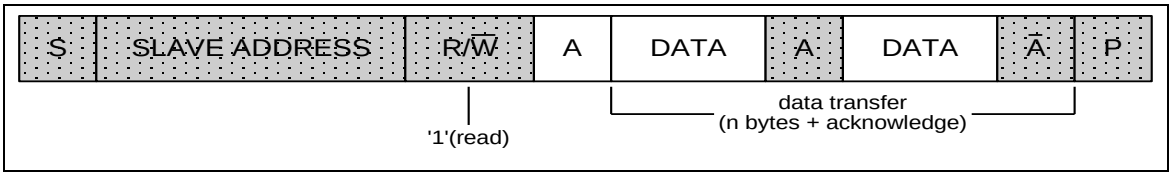


图 5.9-4 主设备从从设备读数据

5.9.3.2 START 或者 Repeated START 信号

当总线空闲时,意味着没有主设备占用总线 (SCL 和 SDA 都为高), 主设备可以通过发送START信号来初始化一次传输. START 信号, 常常称作S位, 定义为当SCL为高时SDA由高变低. START 信号表示一次新的数据传输开始.

Repeated START (Sr) 就是在两次START信号之间没有STOP信号. 主设备使用这个方法来跟不同的从设备通讯或者同一个从设备但是不同的数据传输方向(例如:写从设备变成读从设备) ， 而不用释放总线.

5.9.3.3 STOP 信号

主设备通过产生STOP信号来结束传输. STOP 信号, 常常称作P位, 定义为当SCL为高时SDA由低变高.

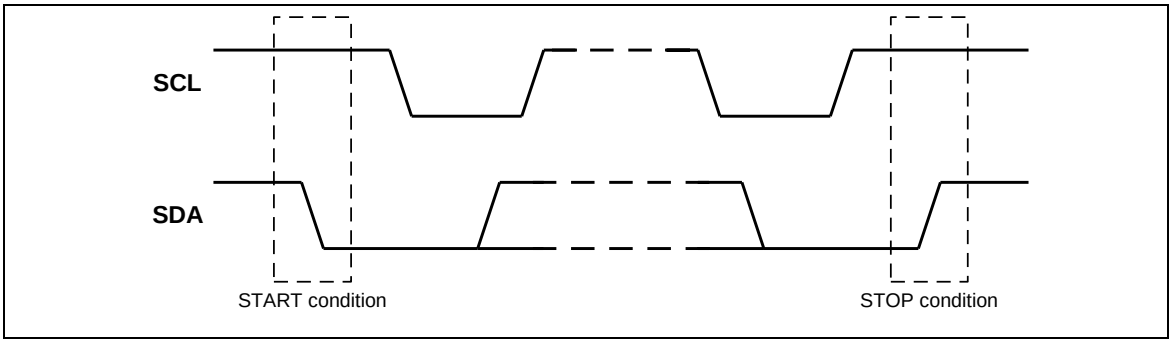


图 5.9-5 START 和 STOP 条件

5.9.3.4 从地址传输

START信号之后， 主设备发送的第一个字节就是从设备地址. 这是一个7比特的地址加一个RW比特. RW 比特通知从设备数据传输的方向. 系统中不可以有两个从设备有一样的地址.只有跟主设备发送的从设备地址匹配的从设备才可以在第9个SCL时钟周期时通过将SDA拉低来返回应答比特.

5.9.3.5 数据传输

一旦从地址寻址成功实现, 数据就可以按照主设备发送的RW位指定的方向按字节一个一个传输. 每个传输的字节后面在第 9 个 SCL 时钟周期都跟随一个应答比特. 如果从设备发 Not Acknowledge(NACK)信号,主设备可以产生STOP信号来终止数据传输，或者产生Repeated START 信号来开始一个新的传输周期.

如果主设备, 作为接收设备, 发Not Acknowledge (NACK) 信号给从设备, 从设备应该释放SDA线，然后主设备产生STOP信号或者Repeated START 信号.

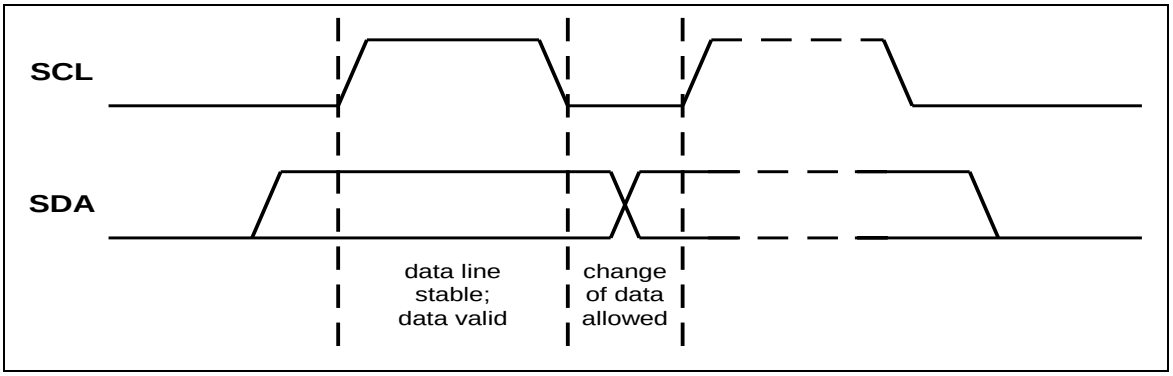


图 5.9-6 I^2C 总线上的比特传输

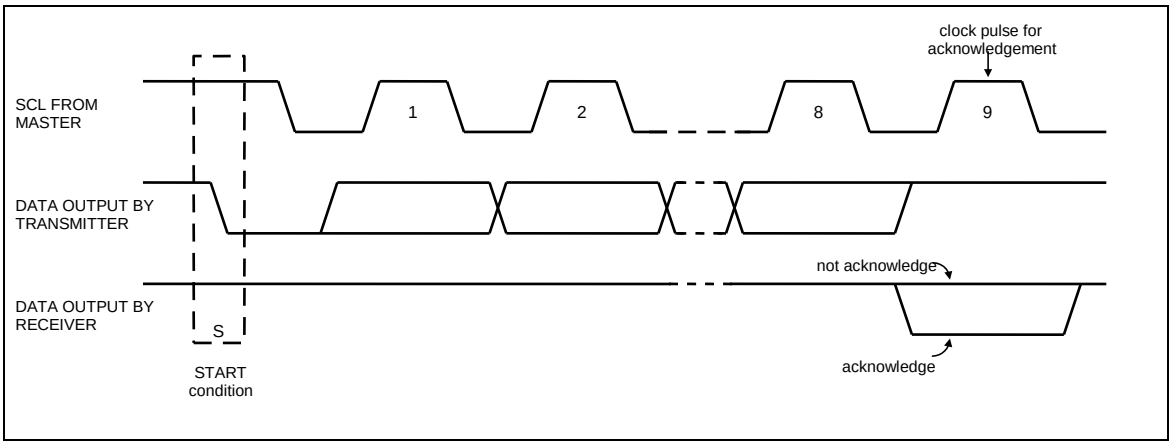


图 5.9-7 I²C 总线应答

5.9.4 I²C 协议寄存器

CPU通过下面13个特殊功能寄存器与I2C连接: 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 (地址掩码寄存器, n=0~3), I2CLK (时钟频率寄存器) 和 I2CTOC (超时计数器寄存器). 所有这些寄存器的比特 31 ~ 比特 8 都是预留的. 这些比特没有作用，读全部返回0.

当设置ENSI (I2CON[6])为高将I²C使能时,内部的状态机由寄存器I2CON和I2C硬件逻辑控制.一旦一个新的状态码产生,并且被存放到I2CSTATUS寄存器,I2C中断标志位SI(I2CON[3]) 将自动被置位.如果中断使能比特EI(I2CON[7]) 同时也为高,I2C中断将发生. I2CSTATUS[7:3]负责保存状态码，低3个比特总是0.状态码一直保持直到SI比特被软件清除.I²C 的基地址是 0x4002_0000.

5.9.4.1 地址寄存器(I2CADDR)

I²C 口配备4个从地址寄存器I2CADDRn (n=0~3). 当I2C工作在主模式时，这些寄存器的值不起作用. 当I2C工作在从模式时, I2CADDRn[7:1] 比特域必须装载MCU自己的从地址. 如果I2CADDR的值和收到的从地址匹配,I²C 硬件将响应.

I²C 支持“General Call” 功能. 如果GC 比特 (I2CADDRn[0]) 被设,I²C 硬件将响应General Call 地址 (00H). 清除GC 比特将关闭general call 功能.

当 GC 比特被设置为”1”并且I²C 工作在从模式时, 如果主设备发送general call地址到I2C总线上,MCU就可以接收general call地址，接下来遵循GC模式的状态.

I²C总线控制器支持多地址识别，有4个从地址寄存器并支持掩码I2CADMn (n=0~3). 如果掩码寄存器的某个比特设为1,意味着收到的从地址相应比特无关. 如果掩码寄存器的某个比特设为0,意味着收到的从地址相应比特应该与从地址寄存器相同.

5.9.4.2 数据寄存器 (I2CDAT)

这个寄存器包含将要发送或者刚刚收到的一个字节的数据. 当这个寄存器没有正在搬移数据时，CPU能直接读或者写这个8-bit (I2CDAT[7:0]) 寄存器. 当I2C在已知状态并且串行中断标志(SI)被设,只要SI位被置位，在I2CDAT[7:0] 中的数据将一直保持稳定. 数据被移出之后，总线上的数据同步被

移入; I2CDAT[7:0] 总是保存总线上最后出现的数据. 因而, 当仲裁失败时,I2C自动从主发送变成从接收模式，I2CDAT[7:0]中的数据将是正确的.

I I2CDAT[7:0] 和应答比特形成一个9-bit 的移位寄存器, 应答比特由I2C硬件控制，CPU不能访问. 除了应答比特，串行数据在SCL串行时钟的上升沿移入I2CDAT[7:0] 寄存器, 应答比特(ACK or NACK) 在第9个SCL时钟由硬件返回. 串行数据在SCL串行时钟的下降沿移出I2CDAT[7:0] 寄存器，在SCL串行时钟的上升沿移入I2CDAT[7:0] 寄存器.

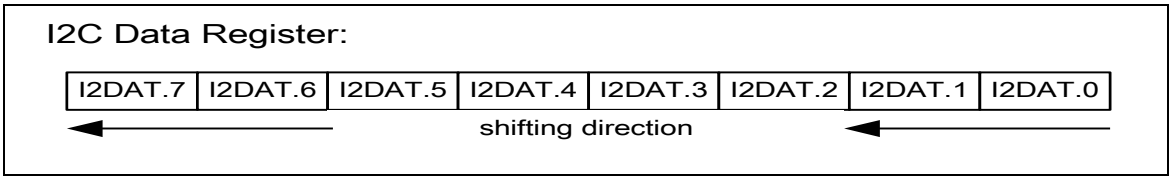


图 5.9-8 I²C 数据移位方向

5.9.4.3 控制寄存器(I2CON)

CPU能直接读或者写I2CON[7:0] 寄存器. 其中两个比特由硬件控制: 当I2C硬件请求串行中断的时候，SI 被设置;当STOP 条件出现在总线上时,STO比特被清除. 当ENSI = "0"时,STO 比特也会被清除.

EI 使能中断.

ENSI 使能I2C功能. 当ENSI=1 时,I²C 功能被使能.多功能引脚SDA和SCL必需先被设为I2C功能.

STA I²C START 信号控制比特. 设置 STA 比特为“1” I2C将进入主模式, 当总线空闲时，I²C硬件发送START 或者 repeat START 信号到总线上.

STO 主模式下I²C STOP 信号控制比特, 设置 STO 比特发送一个STOP 信号到总线上.I2C硬件将查看总线状态,如果STOP信号被检测到,硬件将自动清除这个比特. 从模式下,设置STO比特将复位I2C硬件到”未寻址”从模式. 这意味着从设备不再能从主设备接收数据.

SI I²C中断标志. 当I2CSTATUS寄存器存在一个新的I2C状态时,硬件将设置SI标志, 如果EI比特 (I2CON[7]) 被设置, I²C中断将发生. SI 标志必须由软件清除. 写“1” 到该比特就可以清除. 所有状态都在5.9.12节列出.

AA 发起应答控制比特. 如果在地址或者数据收到之前设置AA=1, 在SCL的应答周期,应答(SDA拉低)将返回:1.) 从设备应答主设备的寻址, 2.) 接收方应答数据发送方. 如果在地址或者数据收到之前设置AA=0, 在SCL的应答周期未应答NACK(SDA高电平)将返回.

5.9.4.4 状态寄存器(I2CSTATUS)

I2CSTATUS[7:0] 是一个8-bit 只读寄存器.最低3个比特总是”0”. 比特域I2CSTATUS[7:3] 包含状态码. 总共有27 个状态码. 所有状态都在5.9.12节列出. 如果I2CSTATUS[7:0] 包含“F8H”, 表示未寻址状态，没有中断发生. 所有其它的I2CSTATUS[7:3] 值都对应已定义的I2C状态. 进入每一个状态时，状态中断标志(SI = 1)将被硬件置位. 一个时钟周期之后，有效的状态码将放在I2CSTATUS[7:3]中，SI被软件清除之后状态码还将存在一个时钟周期.

另外, 状态“00H” 代表总线错误. 在一帧中如果START或者STOP信号存在于一个非法的位置，总线错误将发生. 在地址字节、数据字节或者应答比特都会有非法位置. 为了将I2C从总线错误恢复,应该设置STO位并且清除SI标志以进入未寻址从模式. 然后清除STO位释放总线等待新的通讯. 总线错误

发生时，I²C 不能识别停止信号.

5.9.4.5 I²C时钟频率寄存器(I2CLK)

I2C主模式下，总线的频率由I2CLK[7:0] 寄存器决定. 当I2C在从模式时这个寄存器不起作用. 从模式下,I2C将自动和主设备的时钟同步，频率最高1 MHz.

I2C的数据传输频率公式为:I²C总线频率 = PCLK / (4x (I2CLK[7:0] +1)). 如果PCLK=12 MHz, I2CLK[7:0] = 9 (0x09), 则I2C波特率= 12 MHz / (4x (9 +1)) = 300K 比特每秒. 方块图如下所示.

5.9.4.6 I²C超时计数器寄存器(I2CTOC)

I2C有一个14-bit 超时计数器，可以用来处理I2C总线挂起的情况. 如果超时计数器使能,计数器开始计数直到溢出(TIF=1) ，之后发中断通知CPU或者清除ENTI停止计数. 超时计数器使能时,设置SI标志为高将复位计数器，清除SI标志之后计数重新开始. 如果I²C 总线挂起,将导致I2CSTATUS和SI标志都不更新, 14-bit 超时计数器将溢出，发中断通知CPU. 参考下图14比特超时计数器. 用户可以写”1”到TIF比特将其清除.

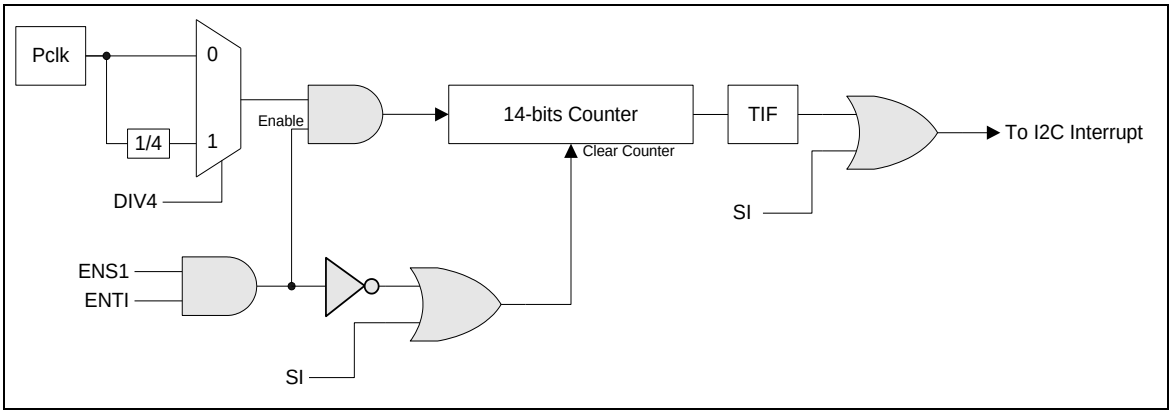


图 5.9-9 I²C 超时计数方块图

5.9.5 寄存器映射

R: 只读, **W:** 只写, **R/W:** 可读写

寄存器	偏移	R/W	描述	复位值
I2C_BA = 0x4002_0000				
I2CON	I2C_BA+0x00	R/W	I^2C 控制寄存器	0x0000_0000
I2CADRR0	I2C_BA+0x04	R/W	I^2C 从地址寄存器0	0x0000_0000
I2CDAT	I2C_BA+0x08	R/W	I^2C数据寄存器	0x0000_0000
I2CSTATUS	I2C_BA+0x0C	R	I^2C状态寄存器	0x0000_00F8
I2CLK	I2C_BA+0x10	R/W	I^2C 时钟除频寄存器	0x0000_0000
I2CTOC	I2C_BA+0x14	R/W	I^2C超时计数器寄存器	0x0000_0000
I2CADDR1	I2C_BA+0x18	R/W	I^2C 从地址寄存器1	0x0000_0000
I2CADDR2	I2C_BA+0x1C	R/W	I^2C 从地址寄存器2	0x0000_0000
I2CADDR3	I2C_BA+0x20	R/W	I^2C 从地址寄存器3	0x0000_0000
I2CADM0	I2C_BA+0x24	R/W	I^2C 从地址掩码寄存器 0	0x0000_0000
I2CADM1	I2C_BA+0x28	R/W	I^2C从地址掩码寄存器 1	0x0000_0000
I2CADM2	I2C_BA+0x2C	R/W	I^2C从地址掩码寄存器 2	0x0000_0000
I2CADM3	I2C_BA+0x30	R/W	I^2C从地址掩码寄存器 3	0x0000_0000

5.9.6 Register Description

I²C 控制寄存器(I2CON)

寄存器	偏移	R/W	描述	复位值
I2CON	I2C_BA+0x00	R/W	I²C 控制寄存器	0x0000_0000

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

Bits	描述	
[31:8]	-	预留
[7]	EI	中断使能 1 = 使能I²C中断. 0 = 禁止I²C 中断.
[6]	ENSI	I²C控制器使能比特 1 = 使能. 0 = 禁止. 置”1”可以使能I2C功能. 当ENSI=1 时将使能I²C 控制器. SDA和SCL多功能引脚也必须设成I2C功能.
[5]	STA	I²C START 控制比特 设置STA 为 “1” 将进入主模式, 在总线空闲时I²C 硬件会发送START 或者repeat START 信号到总线上.
[4]	STO	I²C STOP控制比特 主模式下, 设置STO为”1”将发送STOP 信号到总线上.此后I2C硬件将检查总线的状态,如果检测到STOP信号, 该比特将被硬件自动清0. 从模式下,设置STO将复位I2C硬件到”未寻址”从模式,. 这就意味着从设备不再能接收主设备发送的数据.

Bits	描述	
[3]	**SI**	**I^2C 中断标志** 当一个新的I2C状态存放到I2CSTATUS 寄存器时, SI 标志将被硬件置位,如果EI (I2CON[7]) 也被设为"1", I^2C 中断将发生. SI 必须由软件清除. 写"1" 可以清除该比特.
[2]	**AA**	**发起应答控制比特** 如果在地址或者数据收到之前设置AA=1, 在SCL的应答时钟周期,应答(SDA拉低)将返回:1.) 从设备应答主设备的寻址, 2.) 接收方应答数据发送方. 如果在地址或者数据收到之前设置AA=0, 在SCL的应答脉冲未应答NACK(SDA高电平)将返回.
[1:0]	**-**	**预留**

I²C 数据寄存器(I2CDAT)

寄存器	偏移	R/W	描述	复位值
I2CDAT	I2C_BA+0x08	R/W	I²C 数据寄存器	0x0000_0000

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

Bits	描述	
[31:8]	-	**预留**
[7:0]	**I2CDAT[7:0]**	**I²C 数据寄存器** Bit [7:0]用来存放8比特发送或者接收到的数据.

I²C 状态寄存器(I2CSTATUS)

寄存器	偏移	R/W	描述	复位值
I2CSTATUS	I2C_BA+0x0C	R	I²C状态寄存器	0x0000_00F8

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

Bits	描述	
[31:8]	-	**预留**
[7:0]	**I2CSTATUS[7:0]**	**I²C 状态寄存器** 用来存放I2C控制器的状态: 这个寄存器是只读寄存器.最低3个比特总是”0”.最高5个比特I2CSTATUS[7:3]包含了状态码. 总共有27 个状态码.如果I2CSTATUS[7:0] 包含“F8H”, 表示未寻址状态，没有中断发生. 所有其它的I2CSTATUS[7:3] 值都对应已定义的I2C状态. 进入每一个状态时，状态中断标志(SI = 1)将被硬件置位. 一个时钟周期之后，有效的状态码将放在I2CSTATUS[7:3] 中，SI被软件清除之后状态码还将存在一个时钟周期. 另外, 状态“00H” 代表总线错误. 在一帧中如果START或者STOP信号存在于一个非法的位置，总线错误将发生. 在地址字节、数据字节或者应答比特都会包含非法位置.

状态码	描述	状态码	描述
0x08	Start	0xA0	Slave Transmit Repeat Start or Stop
0x10	Master Repeat Start	0xA8	Slave Transmit Address ACK
0x18	Master Transmit Address ACK	0xB0	Slave Transmit Arbitration Lost
0x20	Master Transmit Address NACK	0xB8	Slave Transmit Data ACK
0x28	Master Transmit Data ACK	0xC0	Slave Transmit Data NACK
0x30	Master Transmit Data NACK	0xC8	Slave Transmit Last Data ACK
0xF8	Stop	0x60	Slave Receive Address ACK
0x38	Master Arbitration Lost	0x68	Slave Receive Arbitration Lost
0x40	Master Receive ACK	0x80	Slave Receive Data ACK
0x48	Master Receive NACK	0x88	Slave Receive Data NACK
0x50	Master Receive ACK	0x70	GC mode Address ACK
0x58	Master Receive NACK	0x78	GC mode Arbitration Lost
0x00	Bus error	0x90	GC mode Data ACK
		0x98	GC mode Data NACK

状态码的细节在5.9.12节描述.

I²C 波特率控制寄存器(I2CLK)

寄存器	偏移	R/W	描述	复位值
I2CLK	I2C_BA+0x10	R/W	I²C 时钟除频寄存器	0x0000_0000

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

Bits	描述	
[31:8]	**-**	**预留**
[7:0]	**I2CLK[7:0]**	**I²C 时钟除频寄存器** I²C 时钟频率控制比特: I²C数据传输波特率 = PCLK / (4x (I2CLK+1)).

I²C 超时计数器寄存器(I2CTOC)

寄存器	偏移	R/W	描述	复位值
I2CTOC	I2C_BA+0x14	R/W	I²C 超时计数器寄存器	0x0000_0000

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

Bits	描述	
[31:3]	-	预留
[2]	ENTI	**使能超时计数器** 1 = 使能. 0 = 禁止. 使能时, 当SI等于”0”,14-bit 超时计数器开始计数, 当SI等于”1”时,计数器将复位, 在SI被清除之后再次开始计数.
[1]	DIV4	**超时计数器输入时钟除以4** 1 = 使能. 0 = 禁止 使能时, 超时时钟周期将延长4倍.
[0]	TIF	**超时标志** 1 = 超时标志由硬件置位,可以触发中断. 0 = 软件可以写 “1” 来清除该比特.

I²C 从地址寄存器(I2CADDRx)

寄存器	偏移	R/W	描述	复位值
I2CADDR0	I2C_BA+0x04	R/W	I²C从地址寄存器0	0x0000_0000
I2CADDR1	I2C_BA+0x18	R/W	I²C从地址寄存器1	0x0000_0000
I2CADDR2	I2C_BA+0x1C	R/W	I²C从地址寄存器2	0x0000_0000
I2CADDR3	I2C_BA+0x20	R/W	I²C从地址寄存器3	0x0000_0000

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

Bits	描述	
[31:8]	-	**预留**
[7:1]	**I2CADDR[6:0]**	**I²C 从地址寄存器** 主模式下这个寄存器的内容不起作用.从模式下,最高的7个比特必须加载MCU自己的从地址. 如果任何一个地址匹配,I2C硬件将响应.
[0]	**GC**	**General Call 功能** 0 = 禁止General Call 功能. 1 = 使能 General Call 功能.

I²C从地址掩码寄存器(I2CADMx)

寄存器	偏移	R/W	描述	复位值
I2CADM0	I2C_BA+0x24	R/W	I²C从地址掩码寄存器0	0x0000_0000
I2CADM1	I2C_BA+0x28	R/W	I²C从地址掩码寄存器1	0x0000_0000
I2CADM2	I2C_BA+0x2C	R/W	I²C从地址掩码寄存器2	0x0000_0000
I2CADM3	I2C_BA+0x30	R/W	I²C从地址掩码寄存器3	0x0000_0000

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

Bits	描述	
[31:8]	-	**预留**
[7:1]	**I2CADMx[6:0]**	**I²C 从地址掩码寄存器** 1 = 使能掩码(不关心收到的地址相应比特是否匹配). 0 = 禁止掩码 (收到的地址相应比特必须与地址寄存器的值相同). I²C 总线控制器支持多地址识别，并且有4个地址掩码寄存器. 如果掩码寄存器中某个比特设为“1”, 意味着不关心收到的地址相应比特. 如果掩码寄存器中某个比特设为“0”, 意味着接收到的地址相应比特必须与地址寄存器相同.
[0]	-	**预留**

5.9.7 操作模式

片上I^2C 口支持5中操作模式: 主发送, 主接收, 从发送, 从接收, 和GC 呼叫.

一个给定的应用中,I2C口可以当作主模式也可以当作从模式. 从模式下, I^2C 硬件查看自己的从地址和GC呼叫的地址. 如果这些地址中的一个被探测到, 并且从设备愿意接收或者发送数据到主设备(通过设定AA比特),在第9个时钟周期应答脉冲将被发送, 如果中断使能,主和从设备都将发生中断. 当微控制器希望变成总线主模式时,硬件会等待直到总线空闲之后才进入主模式.以便从设备的动作不会被打断. 如果主模式下总线仲裁失败,I2C将立即自动转成从模式并且开始探测它自己的的从地址.

5.9.8 主发送模式

当SCL输出串行时钟的时候,数据由SDA串行输出. 发送的第一个字节包含从地址(7 比特)和一个数据方向比特. 这种情况下RW比特是"0",代表"写", 因而第一个传输的字节是SLA+W.串行数据一次发送8个比特.每个字节发送完之后,将收到应答比特. START 和 STOP 信号输出指示一次传输的开始和结束.

5.9.9 主接收模式

这种情况下RW比特是"1", 代表 "读". 因而第一个发送的字节是SLA+R. 当SCL输出时, 串行数据由SDA接收. 串行数据一次接收8个比特. 每个字节接收完之后, 将发送一个应答比特. START 和 STOP 信号输出指示一次传输的开始和结束.

5.9.10 从接收模式

串行数据和串行时钟由SDA和SCL接收. 收到一个字节之后,将发送一个应答比特. START 和 STOP 信号被识别作为一次传输的开始和结束. 收到从地址和方向位之后,地址识别由硬件实现.

5.9.11 从发送模式

收到的第一个字节当作从接收处理.然而在这个模式下,方向位将指示传输方向反向.当串行时钟由SCL输入时,串行数据通过SDA发送. START 和 STOP 信号被识别作为一次传输的开始和结束.

5.9.12 五种操作模式下数据传输流程

五种操作模式是: 主发送, 主接收, 从发送, 从接收, 和GC Call. SI比特被清除以后,I2CON寄存器的比特 STA, STO 和 AA决定I2C硬件的下一个状态. 新的动作完成以后, 新的状态将被更新,SI标志将被置.如果I2C中断控制比特EI(I2CON[7])被置位, 中断将发生,用户可以在中断处理函数里面处理新的状态码.

每种模式下数据传输流程图如下所示.

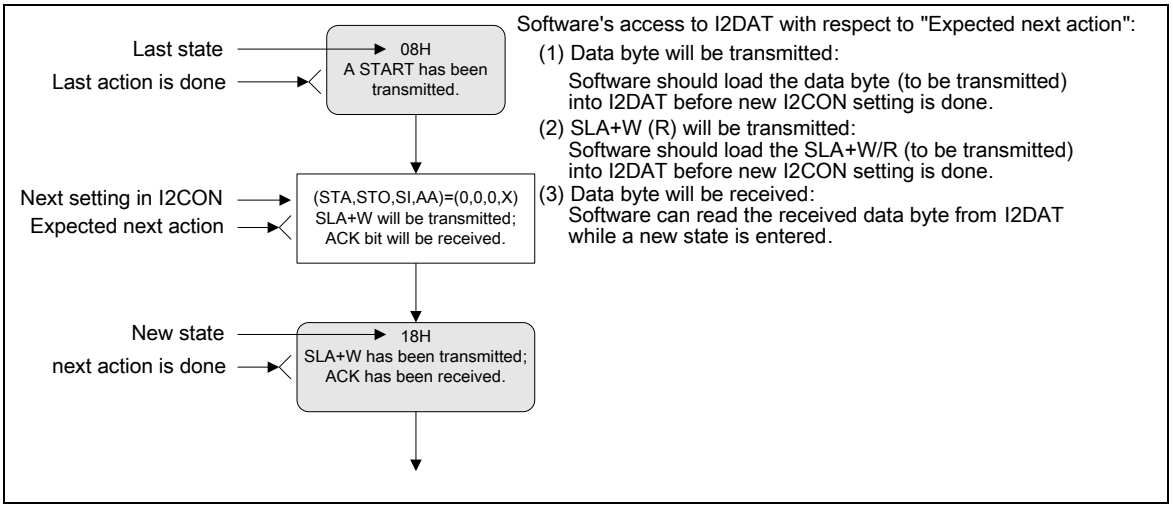


图 5.9-10 下面 4 张图的使用图例

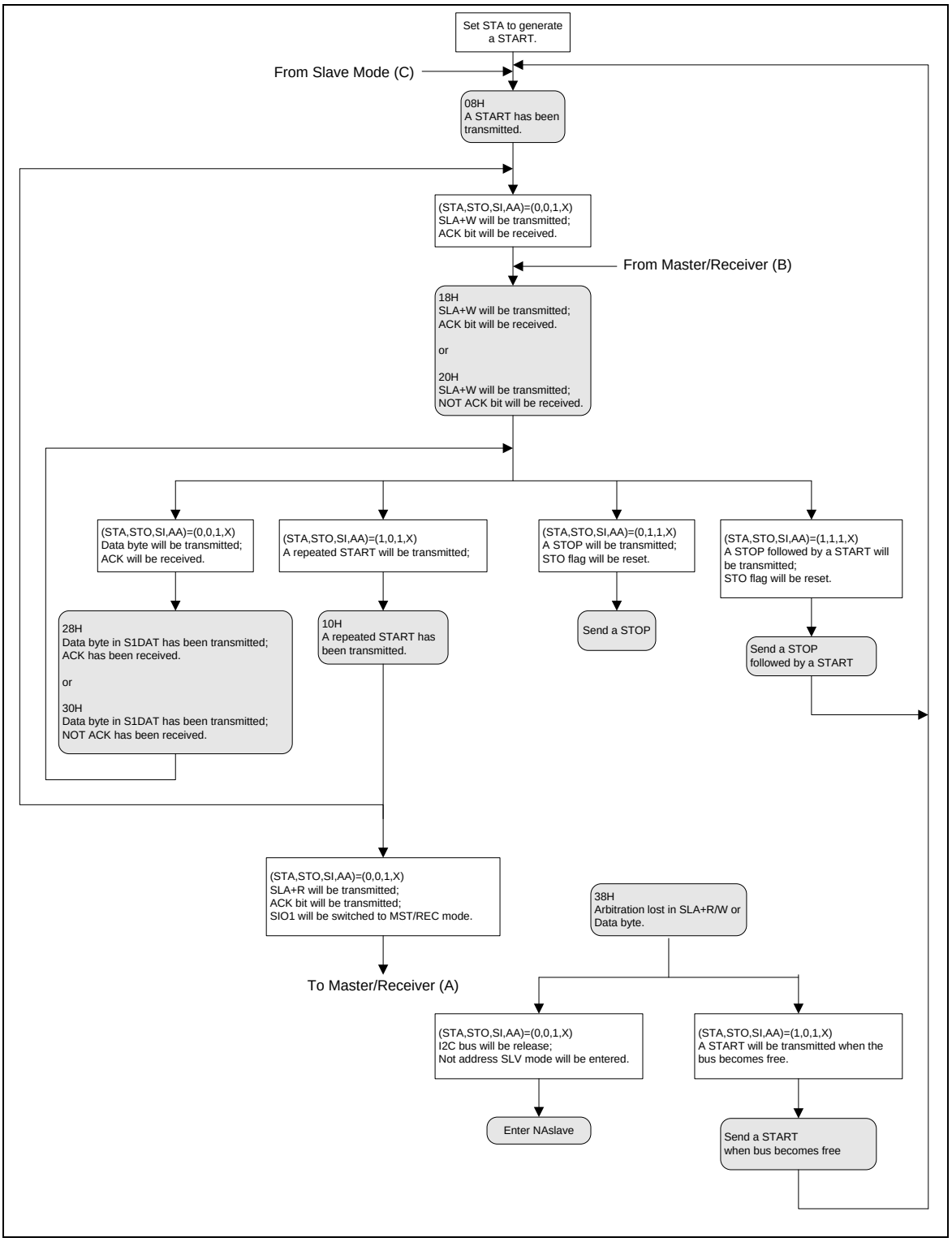

Set STA to generate a START.
From Slave Mode (C)
08H
A START has been transmitted.
(STA,STO,SI,AA)=(0,0,1,X)
SLA+W will be transmitted;
ACK bit will be received.
From Master/Receiver (B)
18H
SLA+W will be transmitted;
ACK bit will be received.
or
20H
SLA+W will be transmitted;
NOT ACK bit will be received.
(STA,STO,SI,AA)=(0,0,1,X)
Data byte will be transmitted;
ACK will be received.
(STA,STO,SI,AA)=(1,0,1,X)
A repeated START will be transmitted;
(STA,STO,SI,AA)=(0,1,1,X)
A STOP will be transmitted;
STO flag will be reset.
(STA,STO,SI,AA)=(1,1,1,X)
A STOP followed by a START will be transmitted;
STO flag will be reset.
28H
Data byte in S1DAT has been transmitted;
ACK has been received.
or
30H
Data byte in S1DAT has been transmitted;
NOT ACK has been received.
10H
A repeated START has been transmitted.
Send a STOP
Send a STOP followed by a START
(STA,STO,SI,AA)=(0,0,1,X)
SLA+R will be transmitted;
ACK bit will be transmitted;
SIO1 will be switched to MST/REC mode.
38H
Arbitration lost in SLA+R/W or Data byte.
To Master/Receiver (A)
(STA,STO,SI,AA)=(0,0,1,X)
I2C bus will be release;
Not address SLV mode will be entered.
(STA,STO,SI,AA)=(1,0,1,X)
A START will be transmitted when the bus becomes free.
Enter NAslave
Send a START when bus becomes free

图 5.9-11 主发送模式

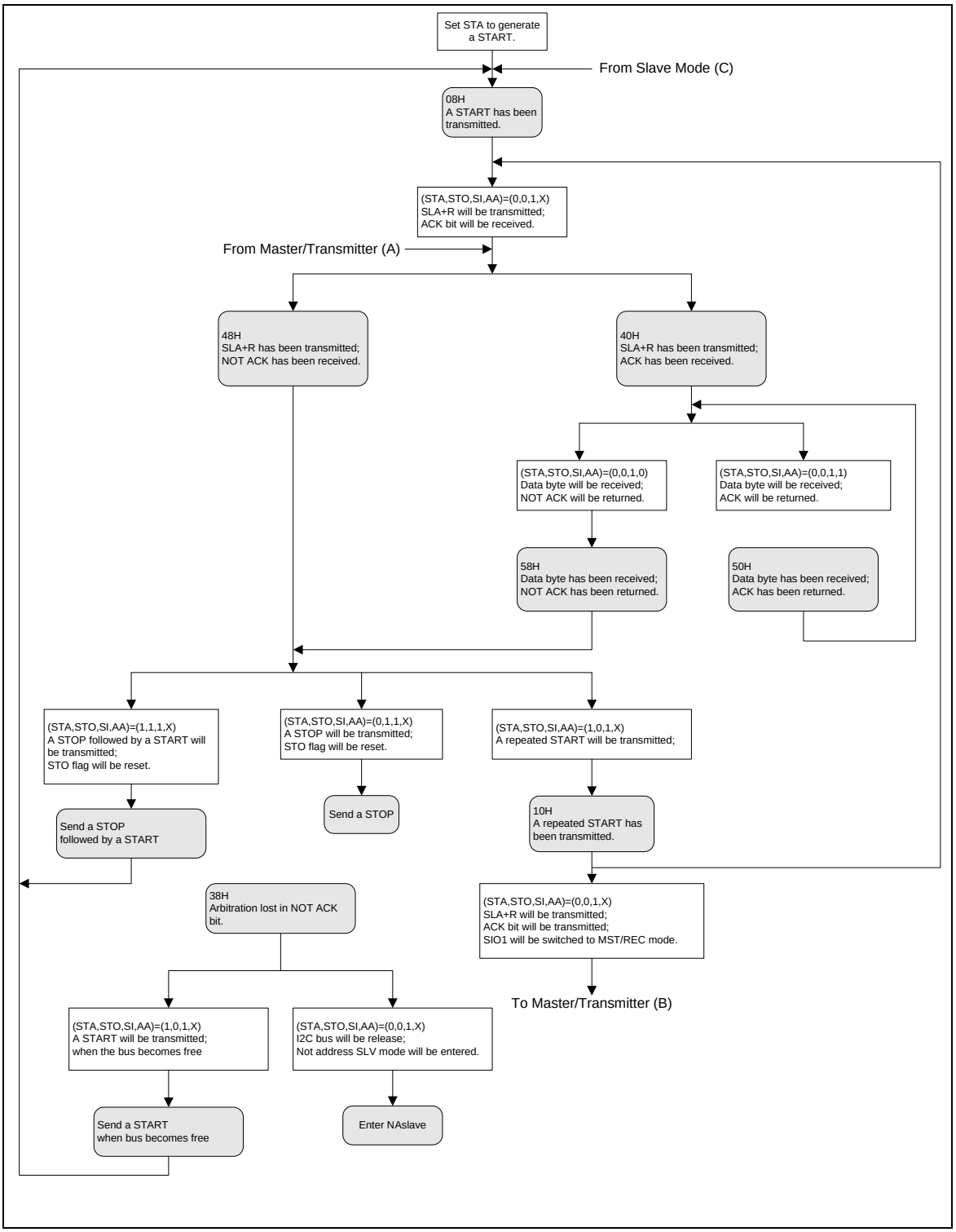

Set STA to generate a START.
From Slave Mode (C)
08H
A START has been transmitted.
(STA,STO,SI,AA)=(0,0,1,X)
SLA+R will be transmitted;
ACK bit will be received.
From Master/Transmitter (A)
48H
SLA+R has been transmitted;
NOT ACK has been received.
40H
SLA+R has been transmitted;
ACK has been received.
(STA,STO,SI,AA)=(0,0,1,0)
Data byte will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(0,0,1,1)
Data byte will be received;
ACK will be returned.
58H
Data byte has been received;
NOT ACK has been returned.
50H
Data byte has been received;
ACK has been returned.
(STA,STO,SI,AA)=(1,1,1,X)
A STOP followed by a START will be transmitted;
STO flag will be reset.
(STA,STO,SI,AA)=(0,1,1,X)
A STOP will be transmitted;
STO flag will be reset.
(STA,STO,SI,AA)=(1,0,1,X)
A repeated START will be transmitted;
Send a STOP followed by a START
Send a STOP
10H
A repeated START has been transmitted.
38H
Arbitration lost in NOT ACK bit.
(STA,STO,SI,AA)=(0,0,1,X)
SLA+R will be transmitted;
ACK bit will be transmitted;
SIO1 will be switched to MST/REC mode.
To Master/Transmitter (B)
(STA,STO,SI,AA)=(1,0,1,X)
A START will be transmitted;
when the bus becomes free
(STA,STO,SI,AA)=(0,0,1,X)
I2C bus will be release;
Not address SLV mode will be entered.
Send a START when bus becomes free
Enter NAslave

图 5.9-12 主接收模式

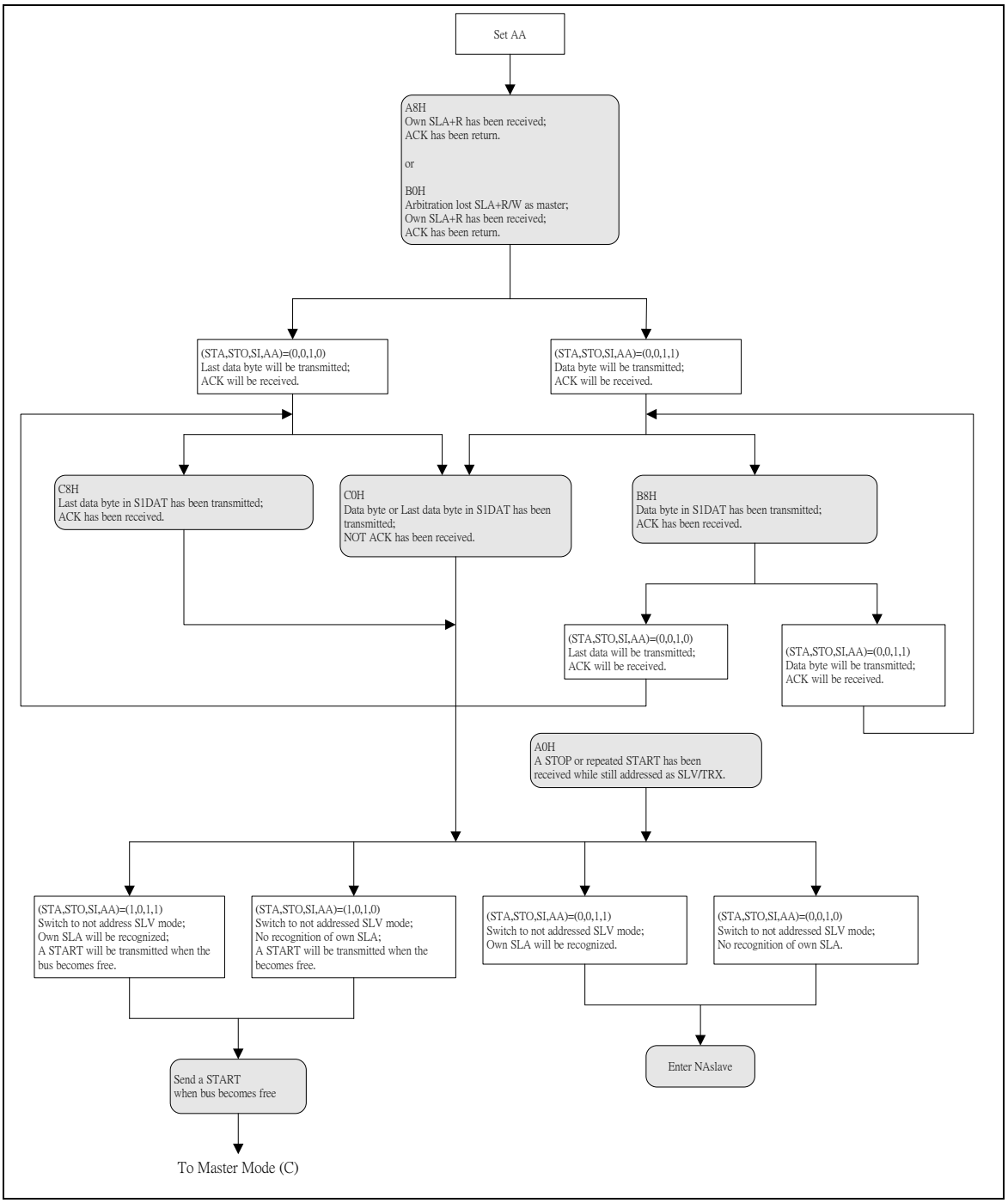

Set AA
A8H
Own SLA+R has been received;
ACK has been return.
or
B0H
Arbitration lost SLA+R/W as master;
Own SLA+R has been received;
ACK has been return.
(STA,STO,SI,AA)=(0,0,1,0)
Last data byte will be transmitted;
ACK will be received.
(STA,STO,SI,AA)=(0,0,1,1)
Data byte will be transmitted;
ACK will be received.
C8H
Last data byte in S1DAT has been transmitted;
ACK has been received.
C0H
Data byte or Last data byte in S1DAT has been transmitted;
NOT ACK has been received.
B8H
Data byte in S1DAT has been transmitted;
ACK has been received.
(STA,STO,SI,AA)=(0,0,1,0)
Last data will be transmitted;
ACK will be received.
(STA,STO,SI,AA)=(0,0,1,1)
Data byte will be transmitted;
ACK will be received.
A0H
A STOP or repeated START has been received while still addressed as SLV/TRX.
(STA,STO,SI,AA)=(1,0,1,1)
Switch to not address SLV mode;
Own SLA will be recognized;
A START will be transmitted when the bus becomes free.
(STA,STO,SI,AA)=(1,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA;
A START will be transmitted when the becomes free.
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed SLV mode;
Own SLA will be recognized.
(STA,STO,SI,AA)=(0,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA.
Send a START
when bus becomes free
Enter NAslave
To Master Mode (C)

图 5.9-13 从发送模式

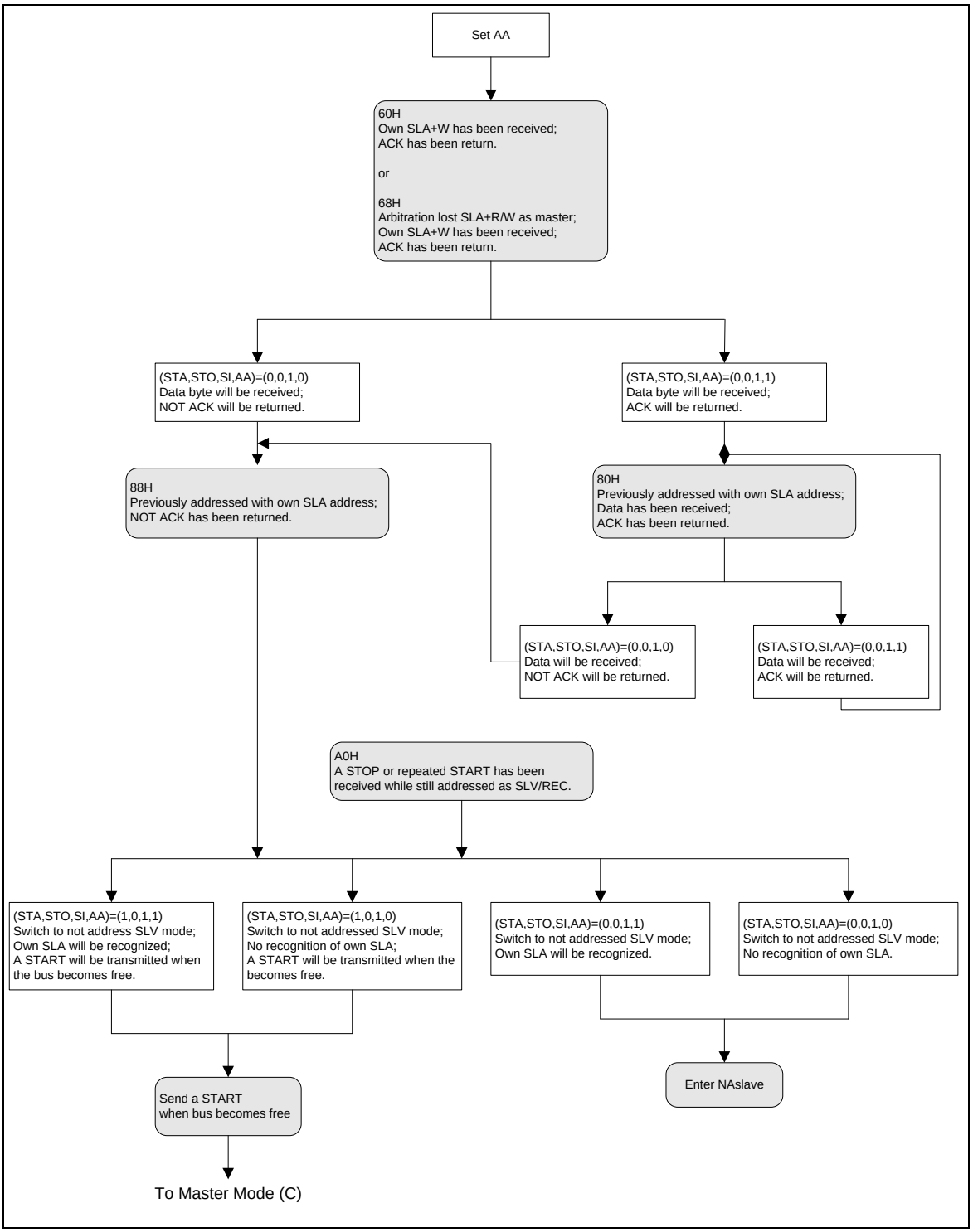

Set AA
60H
Own SLA+W has been received;
ACK has been return.
or
68H
Arbitration lost SLA+R/W as master;
Own SLA+W has been received;
ACK has been return.
(STA,STO,SI,AA)=(0,0,1,0)
Data byte will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(0,0,1,1)
Data byte will be received;
ACK will be returned.
88H
Previously addressed with own SLA address;
NOT ACK has been returned.
80H
Previously addressed with own SLA address;
Data has been received;
ACK has been returned.
(STA,STO,SI,AA)=(0,0,1,0)
Data will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(0,0,1,1)
Data will be received;
ACK will be returned.
A0H
A STOP or repeated START has been
received while still addressed as SLV/REC.
(STA,STO,SI,AA)=(1,0,1,1)
Switch to not address SLV mode;
Own SLA will be recognized;
A START will be transmitted when
the bus becomes free.
(STA,STO,SI,AA)=(1,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA;
A START will be transmitted when the
becomes free.
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed SLV mode;
Own SLA will be recognized.
(STA,STO,SI,AA)=(0,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA.
Send a START
when bus becomes free
Enter NAslave
To Master Mode (C)

图 5.9-14 从接收模式

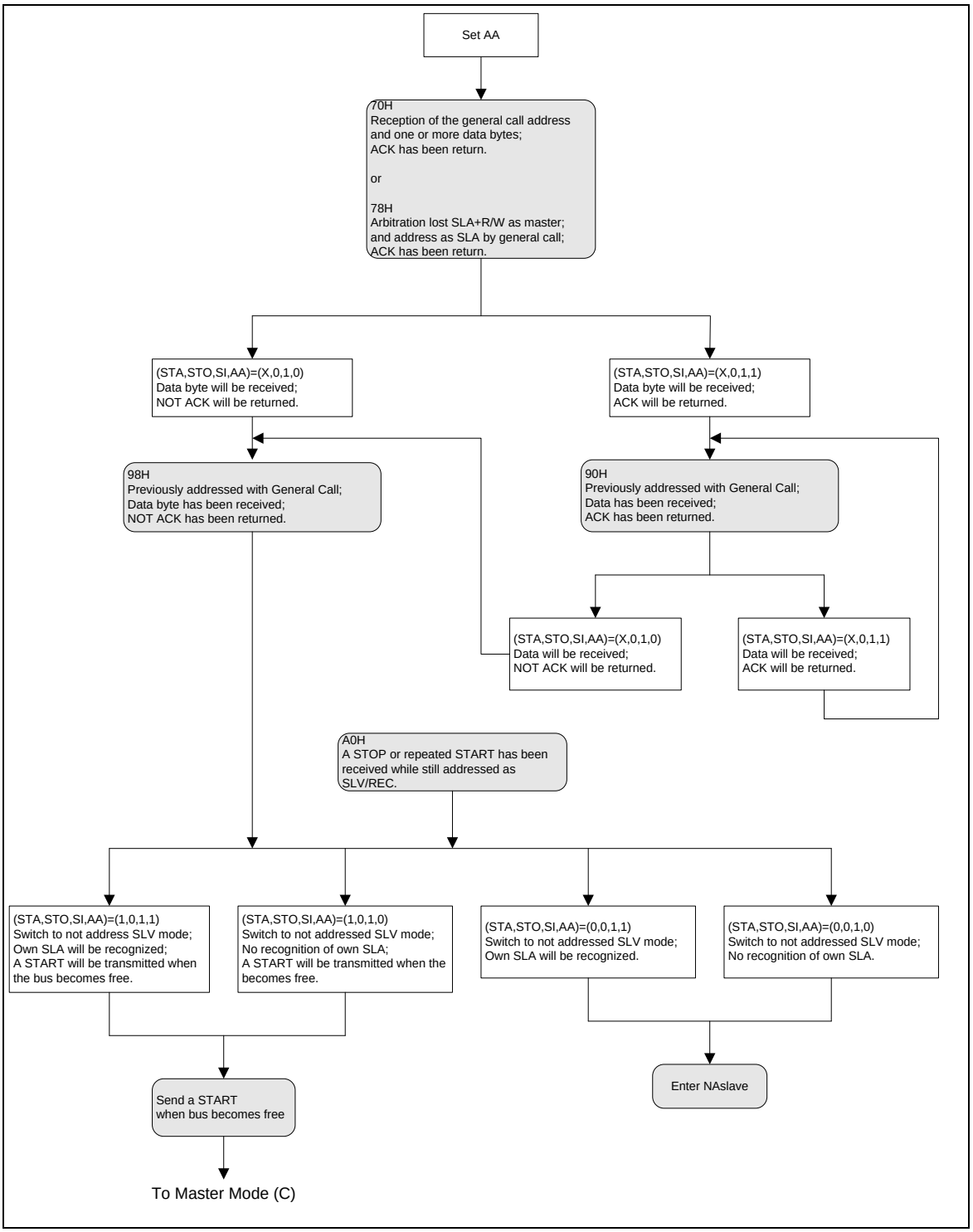

Set AA
70H
Reception of the general call address and one or more data bytes;
ACK has been return.
or
78H
Arbitration lost SLA+R/W as master;
and address as SLA by general call;
ACK has been return.
(STA,STO,SI,AA)=(X,0,1,0)
Data byte will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(X,0,1,1)
Data byte will be received;
ACK will be returned.
98H
Previously addressed with General Call;
Data byte has been received;
NOT ACK has been returned.
90H
Previously addressed with General Call;
Data has been received;
ACK has been returned.
(STA,STO,SI,AA)=(X,0,1,0)
Data will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(X,0,1,1)
Data will be received;
ACK will be returned.
A0H
A STOP or repeated START has been received while still addressed as SLV/REC.
(STA,STO,SI,AA)=(1,0,1,1)
Switch to not address SLV mode;
Own SLA will be recognized;
A START will be transmitted when the bus becomes free.
(STA,STO,SI,AA)=(1,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA;
A START will be transmitted when the becomes free.
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed SLV mode;
Own SLA will be recognized.
(STA,STO,SI,AA)=(0,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA.
Send a START
when bus becomes free
Enter NAslave
To Master Mode (C)

图 5.9-15 GC 模式

5.10 增强型 PWM 发生器

5.10.1 概述

NuMicro MINI51™ 系列内嵌一个PWM单元,特别设计用于驱动马达.PWM单元支持6个PWM发生器，可以配置成6个独立的PWM输出，PWM0~PWM5,或者3组互补的PWM对，(PWM0, PWM1), (PWM2, PWM3) 和 (PWM4, PWM5) ,死区时间可编程.

每组PWM发生器共享一个8-bit 预分频器, 一个时钟除频器提供5个除频频率 (1, 1/2, 1/4, 1/8, 1/16). 每个 PWM 输出有独立的16比特计数器用于控制PWM的周期, 和一个16-bit 比较器用于控制PWM的占空比. 6个PWM发生器提供6个独立的PWM中断标志,当PWM计数周期匹配时，中断标志由硬件置位. 每个PWM中断源有相应的使能位可以控制PWM中断的发生. PWM 发生器可以配置为one-shot 模式只产生一个PWM周期信号或者自动加载模式连续输出PWM波形.

5.10.2 特性

PWM 有下列特性:

- 6个独立的16比特PWM占空比控制单元,最多6个输出引脚:
  - 6 个独立的PWM 输出: PWM0, PWM1, PWM2, PWM3, PWM4, 和 PWM5
  - 3 组互补的PWM对,一对中的两个引脚输出的波形互补,插入的死区时间可以编程 (PWM0, PWM1), (PWM2, PWM3) 和 (PWM4, PWM5)
  - 3 组同步PWM对, 一对中的两个引脚相位同步: (PWM0, PWM1), (PWM2, PWM3) 和 (PWM4, PWM5)
- 组控制比特：PWM2和PWM4 与 PWM0同步
- One-shot (只有边沿对齐模式才支持)或者 Auto-reload PWM模式
- 最大16 比特解析度
- 支持边沿和中心对齐模式
- 互补的PWM对之间插入的死区时间可以编程
- PWM0 到 PWM5 的每个脚有独立的极性控制
- 硬件故障刹车保护
  - 2 种中断源类型:
    - 当下数型计数器比较匹配 (边沿和中心对齐)或者下溢(边沿对齐) 时中断同步发生
    - 当外部故障刹车发生时，中断发生
      - BKP0: EINT0
      - BKP1: EINT1 或者 CPO0
- 极性控制之前的PWM信号输出高电平.之后 PWM 口输出高或者低由极性控制寄存器控制.

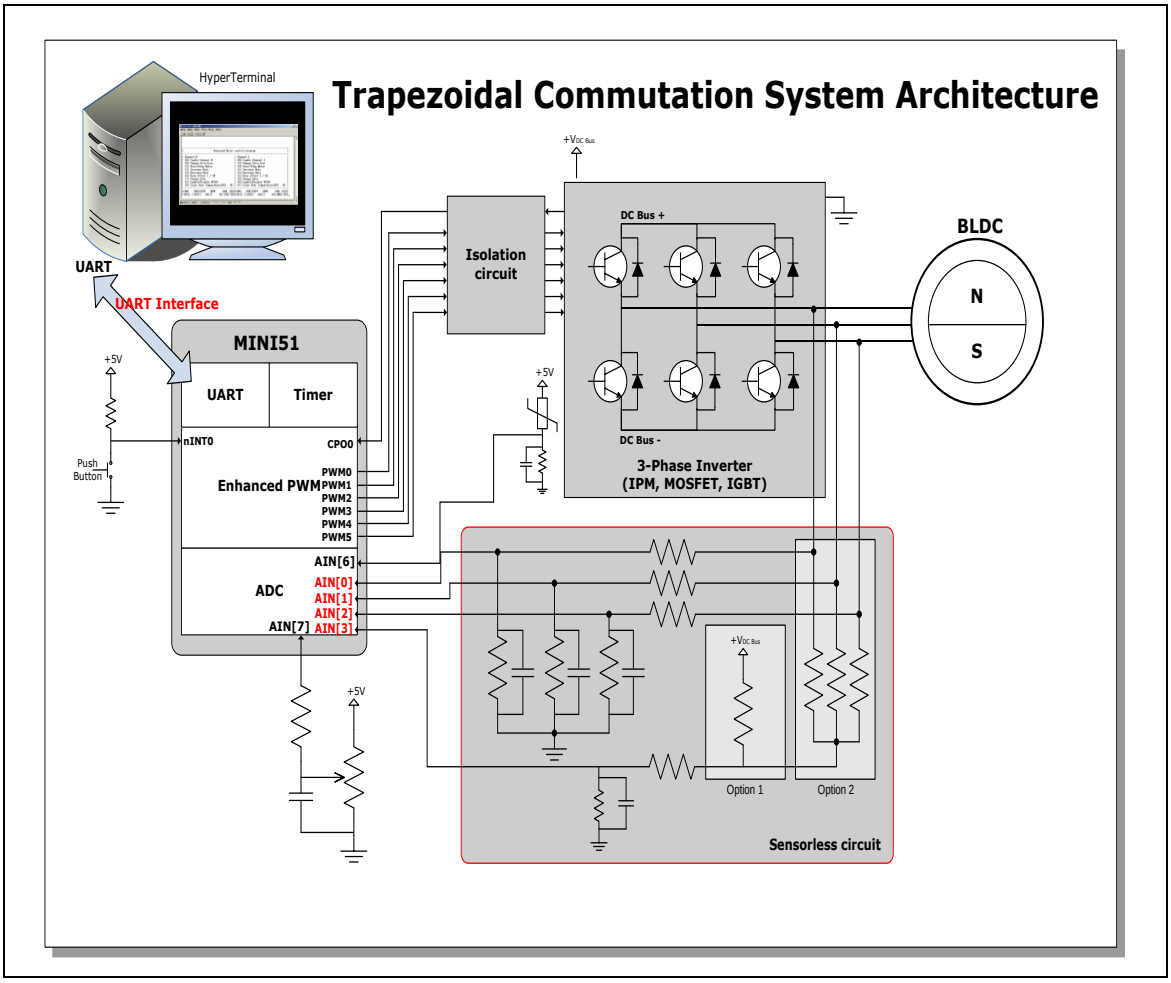

Trapezoidal Commutation System Architecture
HyperTerminal
UART
UART Interface
MINI51
UART
Timer
nINT0
CPO0
PWM0
PWM1
PWM2
PWM3
PWM4
PWM5
Enhanced PWM
AIN[6]
AIN[0]
AIN[1]
AIN[2]
AIN[3]
ADC
AIN[7]
Push Button
+5V
Isolation circuit
DC Bus +
DC Bus -
3-Phase Inverter (IPM, MOSFET, IGBT)
BLDC
N
S
Option 1
Option 2
Sensorless circuit

图 5.10-1 应用电路图

5.10.3 PWM 方块图

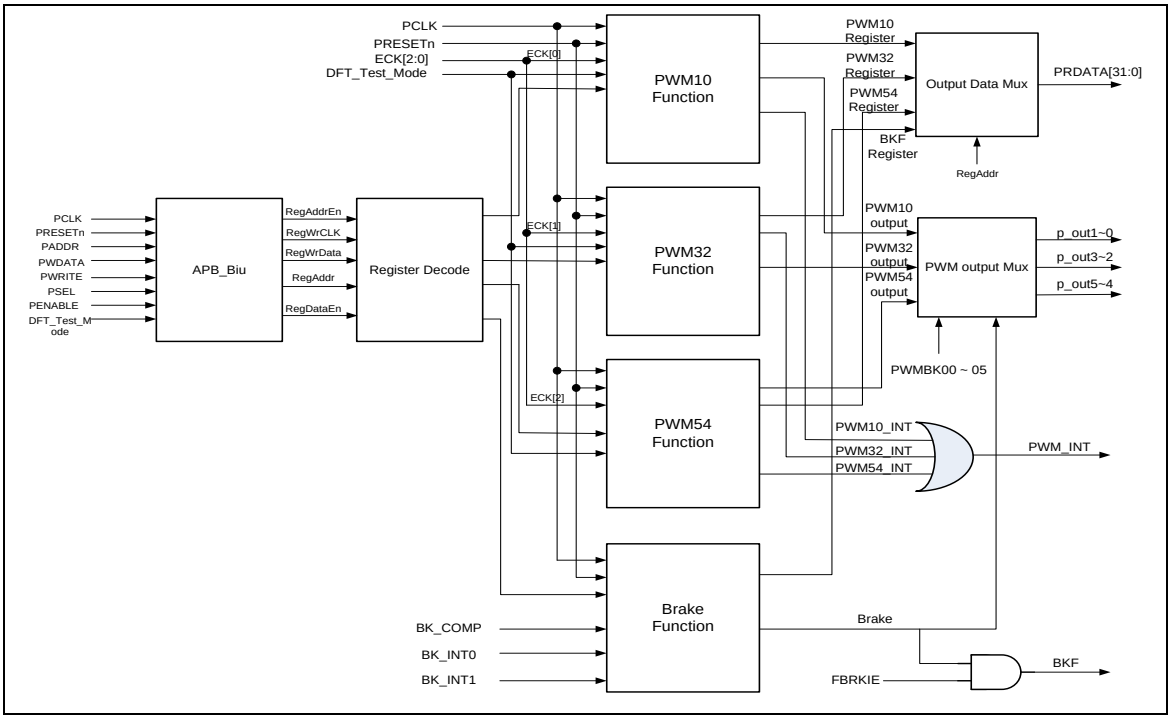


图 5.10-2 PWM 方块图

下图说明了PWM组架构 (PWM 0&1 为一对 ,PWM 2&3 为另一对,等等).

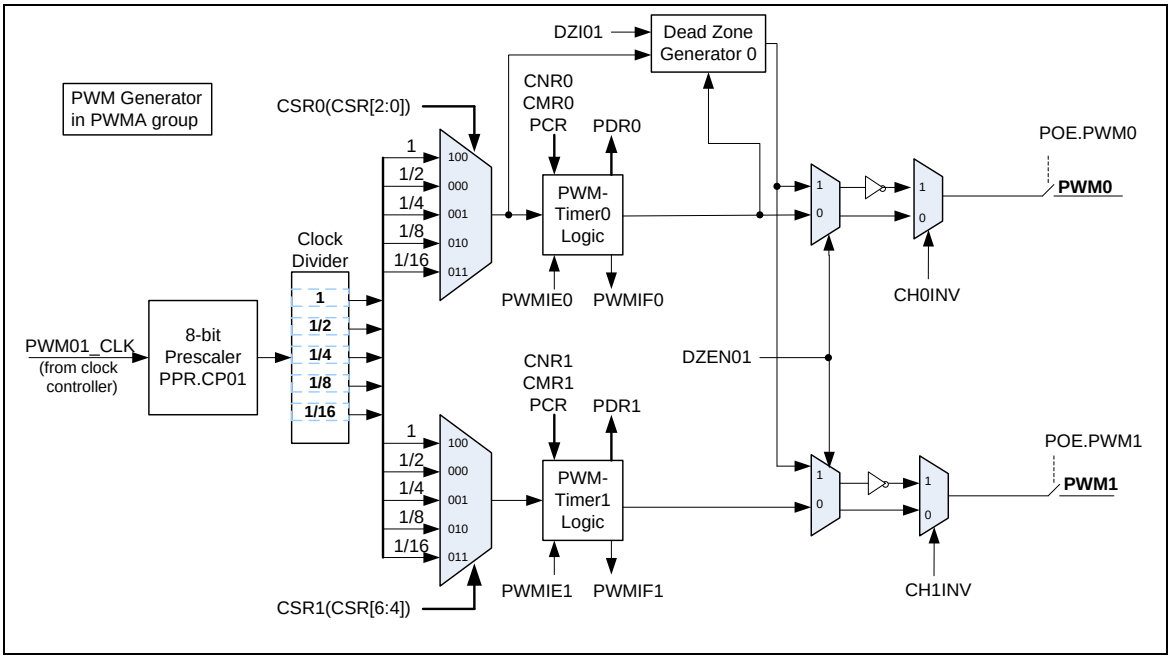


图 5.10-3 PWM 发生器 0 架构图

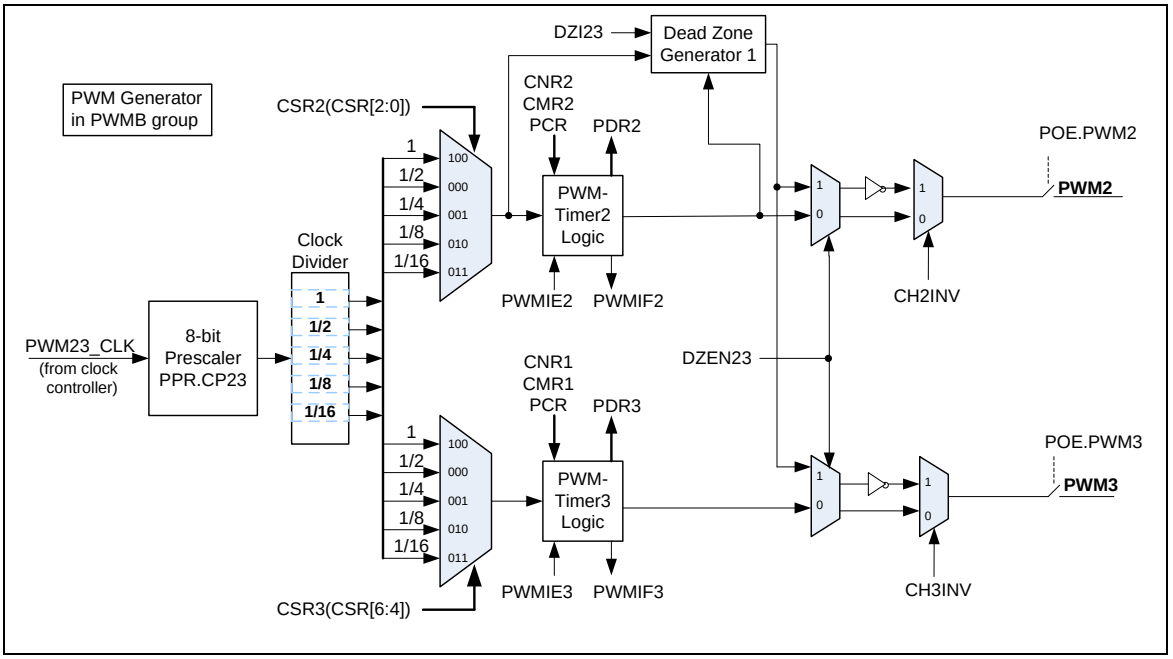


图 5.10-4 PWM 发生器 2 架构图

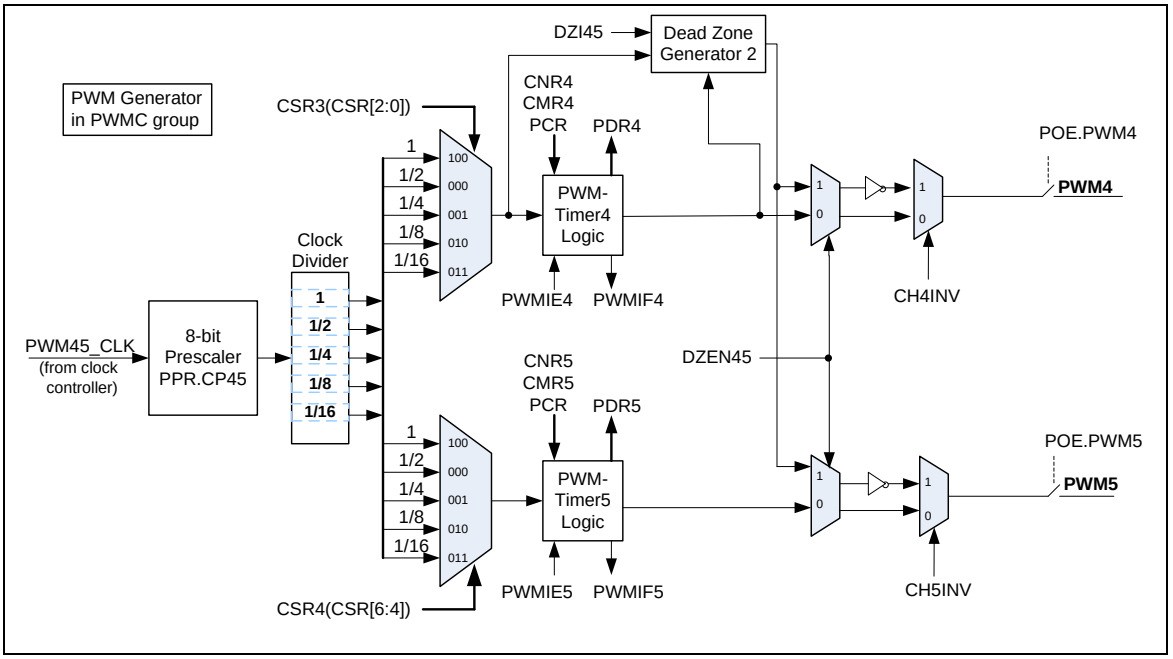


图 5.10-5 PWM 发生器 4 架构图

5.10.4 PWM 功能描述

5.10.4.1 PWM定时器操作

PWM支持2种操作模式: 边沿对齐和中心对齐模式.

下面的等式是每种操作模式下周期和占空比计算公式:

边沿对齐 (下数计数器)

占空比= (CMR+1) / (CNR+1)

Duty = (CMR+1) x (clock period)

周期= (CNR+1) x (clock period)

中心对齐(上数和下数计数器):

占空比= (CNR - CMR) / (CNR+1)

Duty = (CNR - CMR) x 2 x (clock period)

周期= (CNR+1) x 2 x (clock period)

边沿对齐 PWM (下数计数器)

边沿对齐PWM的输出模式: 16比特PWM计数器从CNRn开始下数直到与CMRn的值匹配, 这时PWM发生器将切换输出高电平. 计数器继续下数直到0, 这时PWM发生器将切换输出低电平,如果CHnMODE=1,CMRn和CNR的值将被重新加载，如果中断使能(PIER.n=1) PWM中断将触发.

图 5.10-6 和 图 5.10-7描述了边沿对齐PWM的时序和操作流程.

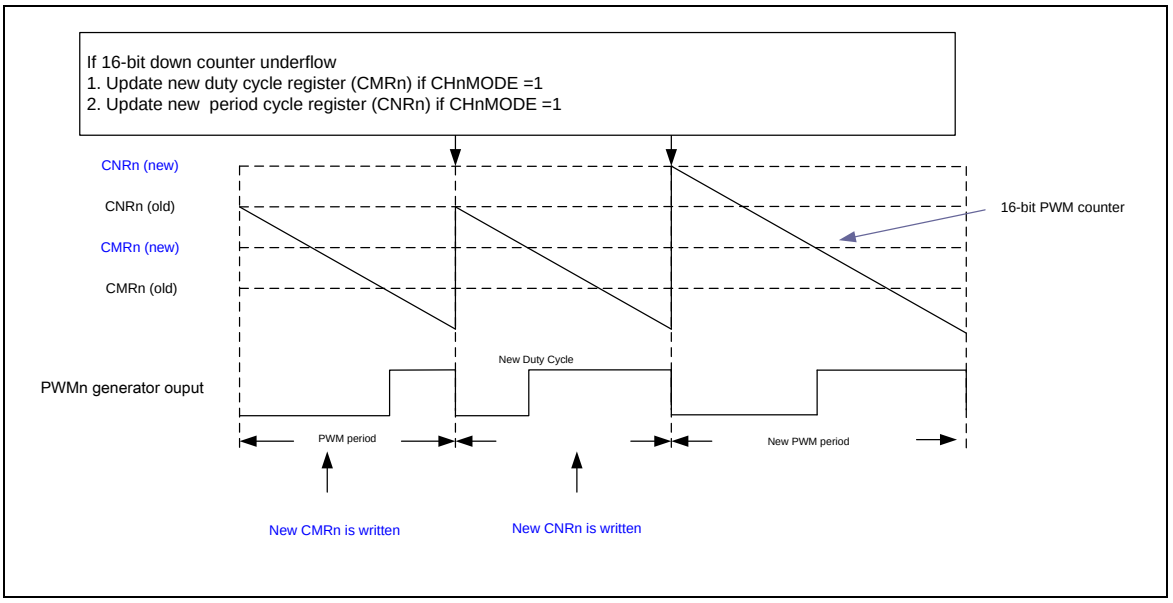


图 5.10-6 边沿对齐 PWM

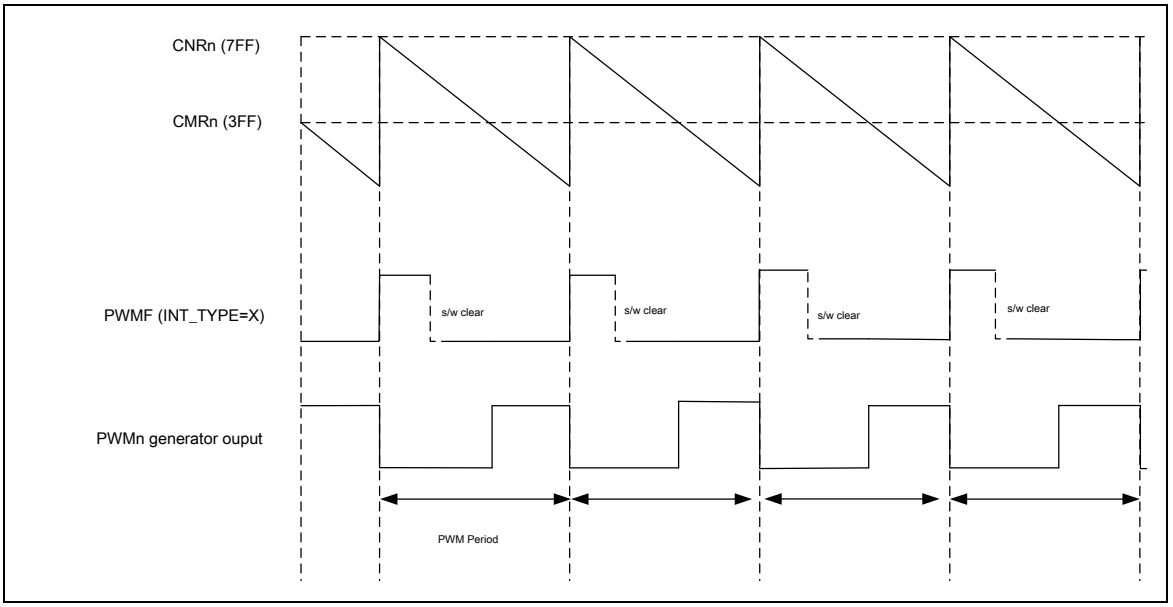


图 5.10-7 PWM 边沿对齐波形输出

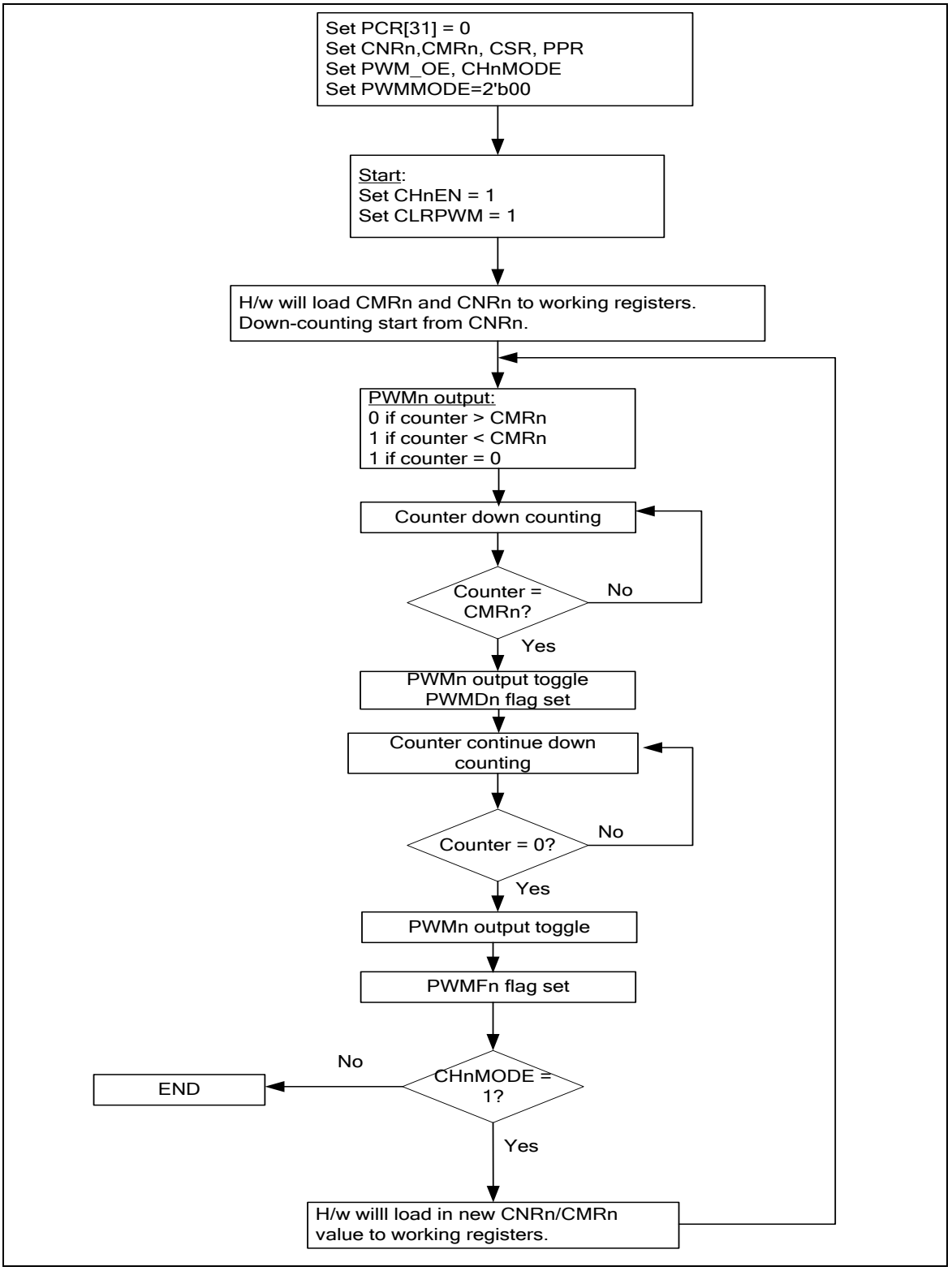


图 5.10-8 E 边沿对齐流程图

PWM 周期和占空比由PWM下数计数器寄存器(CNRn)和PWM比较器(CMRn)控制. PWM定时器时

序操作如图 5.10-10所示. 脉冲宽度调制遵守下面的公式, PWM定时器比较器说明如图 5.10-9所示. 注意相应的GPIO脚必须配置成PWM功能 (使能 PWMPOE) 用于PWM输出.

PWM频率= PWMxy_CLK/((prescale+1)*(clock divider)/(CNR+1)); xy取决于选择的PWM通道,可以是01, 23 或者 45

占空比= (CMR+1)/(CNR+1)

CMR >= CNR: PWM总是输出高

CMR < CNR: PWM 低电平宽度= (CNR-CMR) unit[1]; PWM 高电平宽度= (CMR+1) unit

CMR = 0: PWM 低电平宽度= (CNR) unit; PWM 高电平宽度= 1 unit

注意: 1. Unit =一个PWM时钟周期.

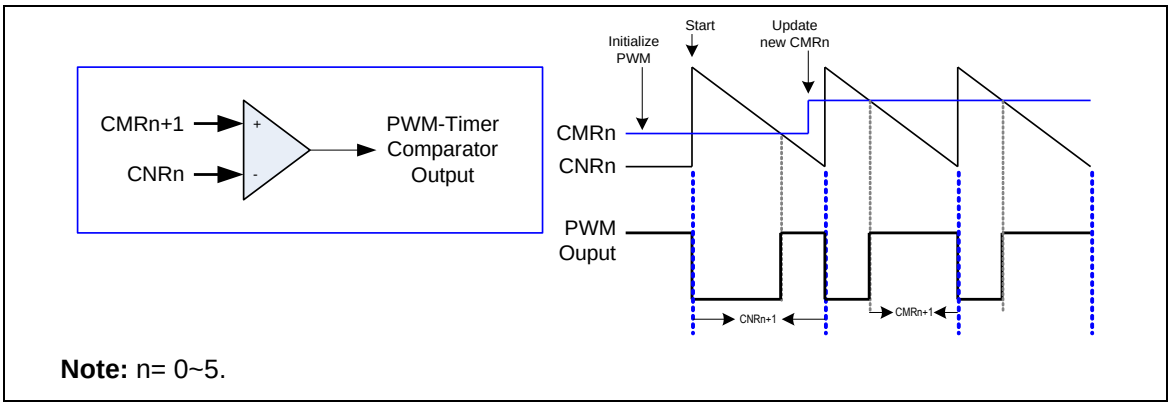


图 5.10-9 定时器的内部比较器输出

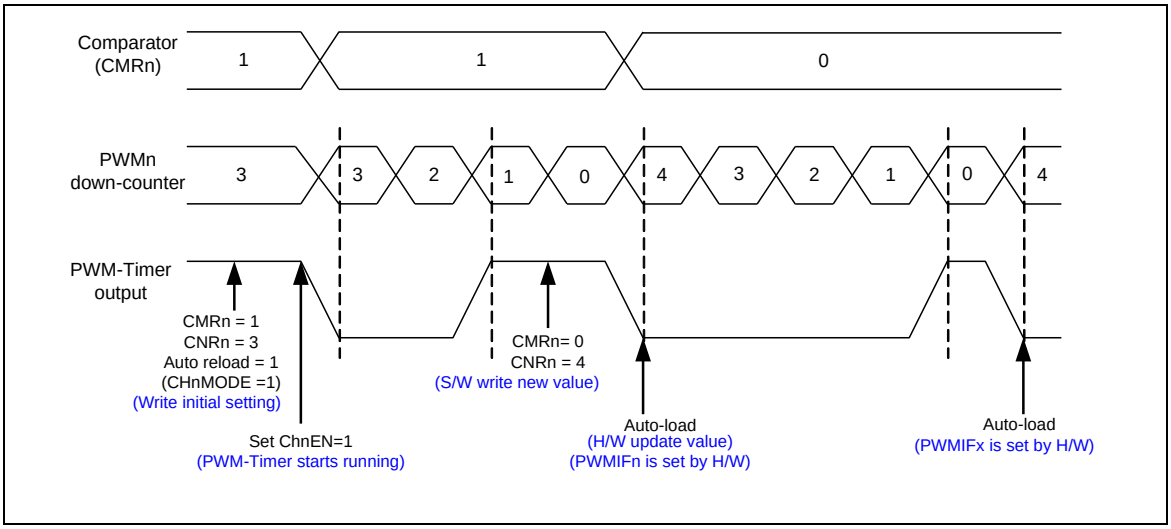


图 5.10-10 PWM 定时器操作时序

中心对齐 PWM (上/下数计数器)

当PWM定时器配置为上/下数型计数器模式时，中心对齐PWM信号将产生. PWM 计数器从0开始上数直到与CMRn的值匹配;PWM发生器将切换输出高.计数器继续数直到与CNRn的值匹配,此后计数器自动开始下数直到再次与CMRn的值匹配，此时PWM发生器将切换输出低电平. 如果CHnMODE = 1,一旦PWM 计数器下溢,PWM周期寄存器CNRn和占空比寄存器将自动更新.

中心对齐模式下, 如果INT_TYPE (PIER[17]) =0，当计数器下溢时(也就是每个周期的开始和结束)PWM周期中断将发生; 如果INT_TYPE (PIER[17]) =1，当上数计数器与CNRn的值匹配时(也就是PWM周期的中点)PWM周期中断将发生.

图 5.10-11, 图 5.10-12 和 图 5.10-13描述了中心对齐PWM的时序和操作流程.

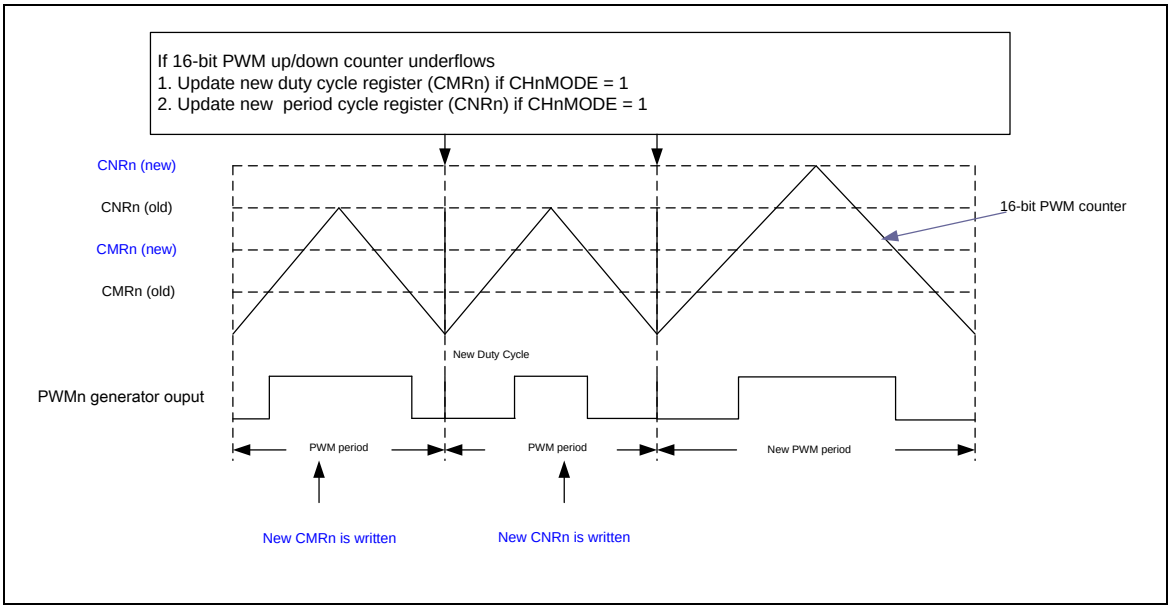


图 5.10-11 中心对齐模式

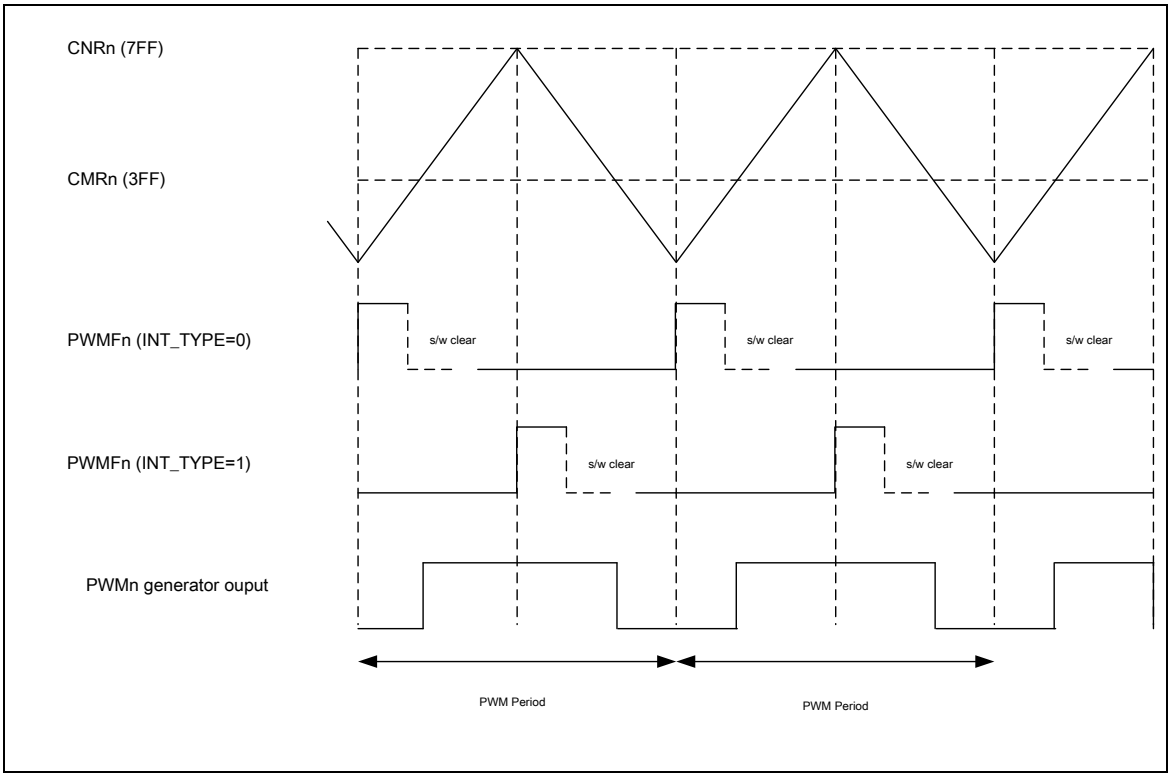


图 5.10-12 PWM 中心对齐波形输出

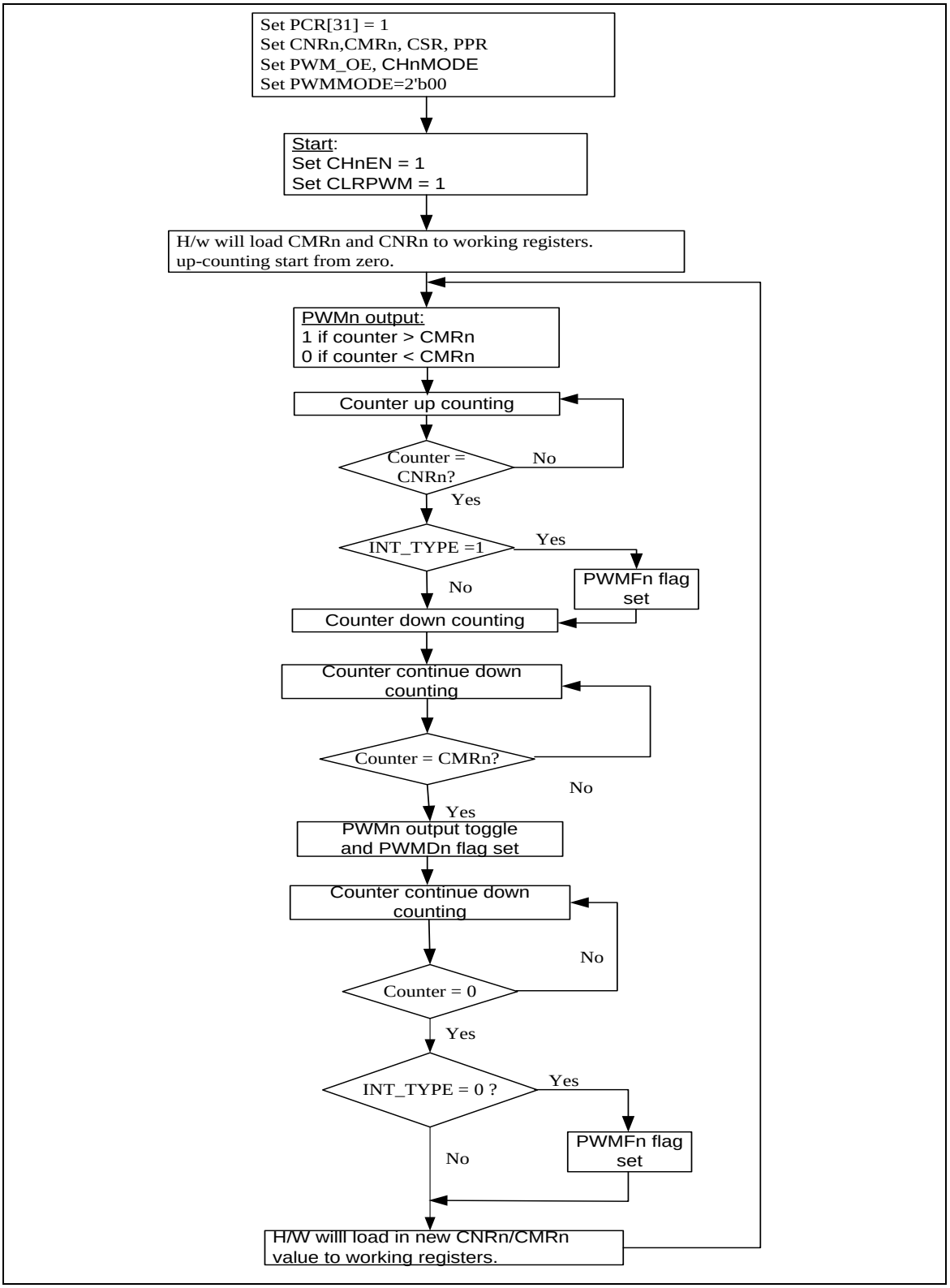


图 5.10-13 中心对齐操作流程(INT_TYPE = 0)

5.10.4.2 PWM 双缓冲, 自动加载和 One-shot 操作

NuMicro MINI51™ 系列 PWM 定时器有双缓冲功能，更新的值将在下个时钟周期起作用不会影响当前周期. PWM 计数器的值写到CNRn寄存器.

PWM 控制寄存器 (PCR)的比特CHnMOD定义PWM操作在自动加载还是 one-shot 模式. 如果CHnMOD 设为 “1”,当PWM计数器到0时,CNRn的值将自动加载到计数器.但是如果CNRn设为0,PWM计数器到0之后也将停止计数；如果CHnMOD 设为 “0”,当PWM计数器到0时,将立即停止计数..PWM0~PWM5 功能相同.

注意: 只有边沿对齐模式才支持One-shot操作

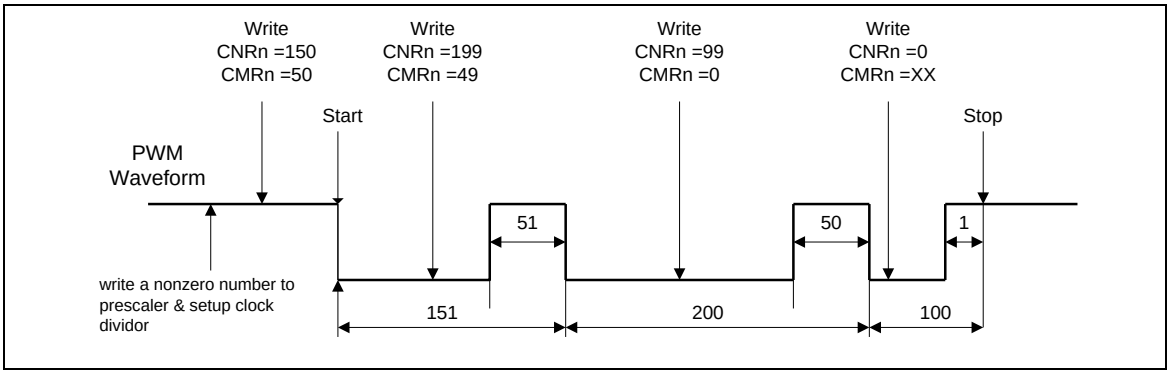


图 5.10-14 PWM 双缓冲说明

5.10.4.3 占空比调制

双缓冲功能允许在任何时间更新CMRn.新加载的值在下个周期起作用.

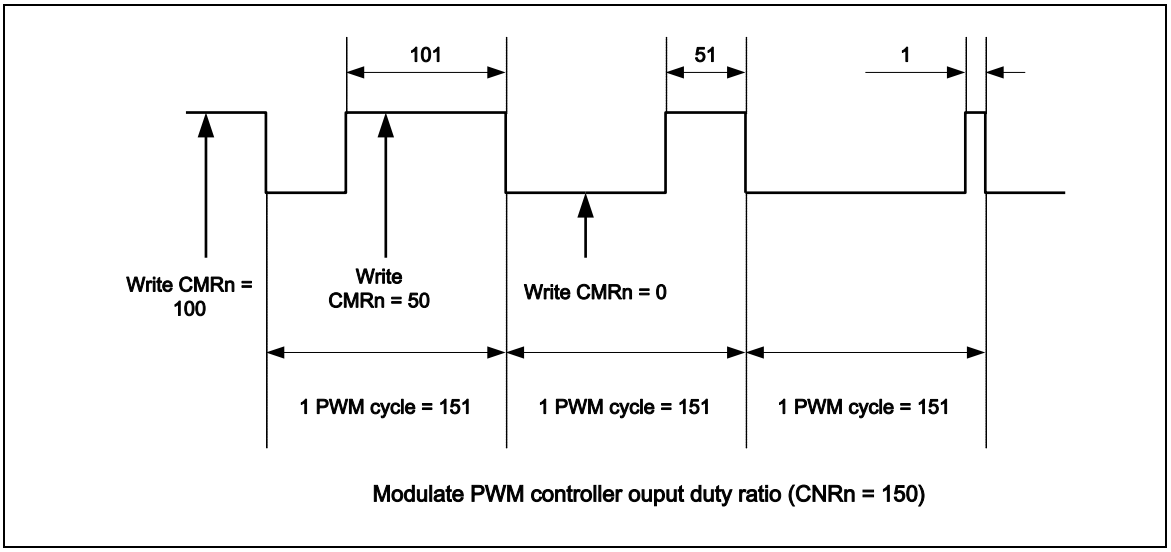


图 5.10-15 PWM 控制器输出占空比

5.10.5 PWM 操作模式

功能强大的PWM单元支持独立模式可以用于直流或者BLDC马达控制，带死区功能的互补模式可以用于交流感应马达和同步马达，同步模式可以使每对的两个引脚相位同步.而且,组模式可以强制PWM2和PWM4与PWM0同步，这样可以简化直流和BLDC占空比控制.

5.10.5.1 独立模式

当PWMMOD[1:0] = “00b”时使能独立模式.

WM缺省工作在独立模式, 6个PWM 通道独立输出. 每个通道有自己的占空比和周期控制.

Note: 在独立模式只有偶数通道(PWM0, PWM2, PWM4)可以使能反转 bit (CHnINV, n=0,2,4).

5.10.5.2 互补模式

当PWMMOD[1:0] = “01b”时使能互补模式.

这种模式有3组占空比-周期发生器, 总共有3组PWM输出引脚. 所有的6个PWM输出按照奇偶号组成对. 互补模式下, 内部的奇数PWM信号PGn必须总是和相应的偶数PWM信号互补. 例如, PG1 和PG0互补. PG3 和PG2互补,PG5和PG4互补. PWM模块的时基由它自己的16位定时器合并可选的预分频选项提供.

5.10.5.3 死区插入

互补模式下，死区发生器在关闭两个成对的引脚之中的一个到打开另外一个互补的引脚之间插入称为 “死区”的“off” 周期，.这可以防止电源切换时设备被破坏。互补对输出模式有一个8比特的下数计数器用来插入死区时间。互补输出一直延迟到计数器减到0.

死区时间可以按照下面的公式来计算:

dead-zone = PWM_CLK * (DZIxy[7:0]+1). xy可以是 01, 23, 45

下面的时序图说明一对PWM信号的死区插入.

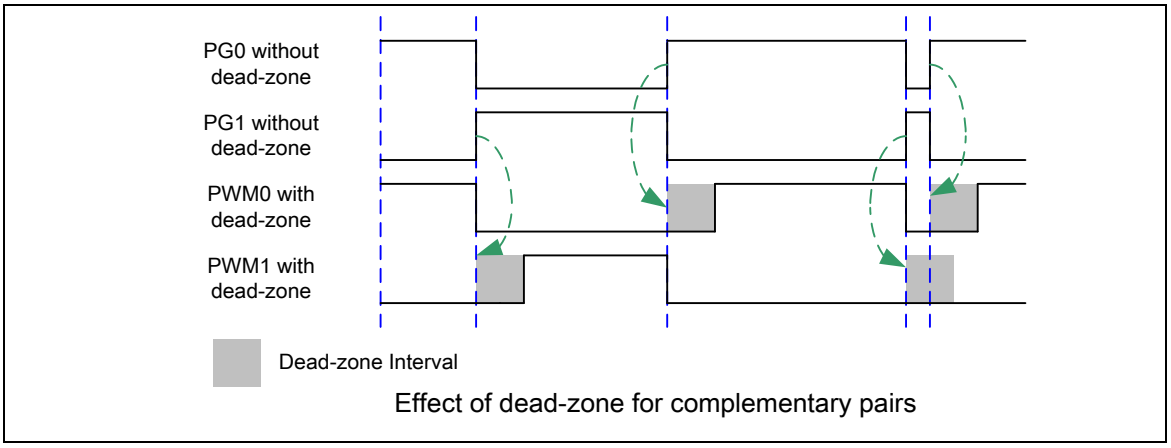


图 5.10-16 死区插入

在逆变器应用中，死区插入可以避免半整流桥的上下臂同时导通。因而死区控制对系统的正常运转至关重要。一个互补对中，在关闭一个PWM输出而打开另一个时，必须等待一定的时间而不能同时切换.

5.10.5.4 同步模式

当PWMMOD[1:0] = “10b”时使能同步模式.

同步模式下,每对PWM信号相位同步.

PG1=PG0, PG3=PG2 和 PG5=PG4.

5.10.5.5 组模式

当GRP (PCR[30]) = 1时使能组模式.

设备支持组模式控制. 该控制允许所有的偶PWM通道输出占空比都由PWM0控制.

如果 GRP = 1, (PG2, PG3) 和 (PG4, PG5) 两对将跟 (PG0, PG1)完全一样, 这就意味着:

PG4 = PG2 = PG0;

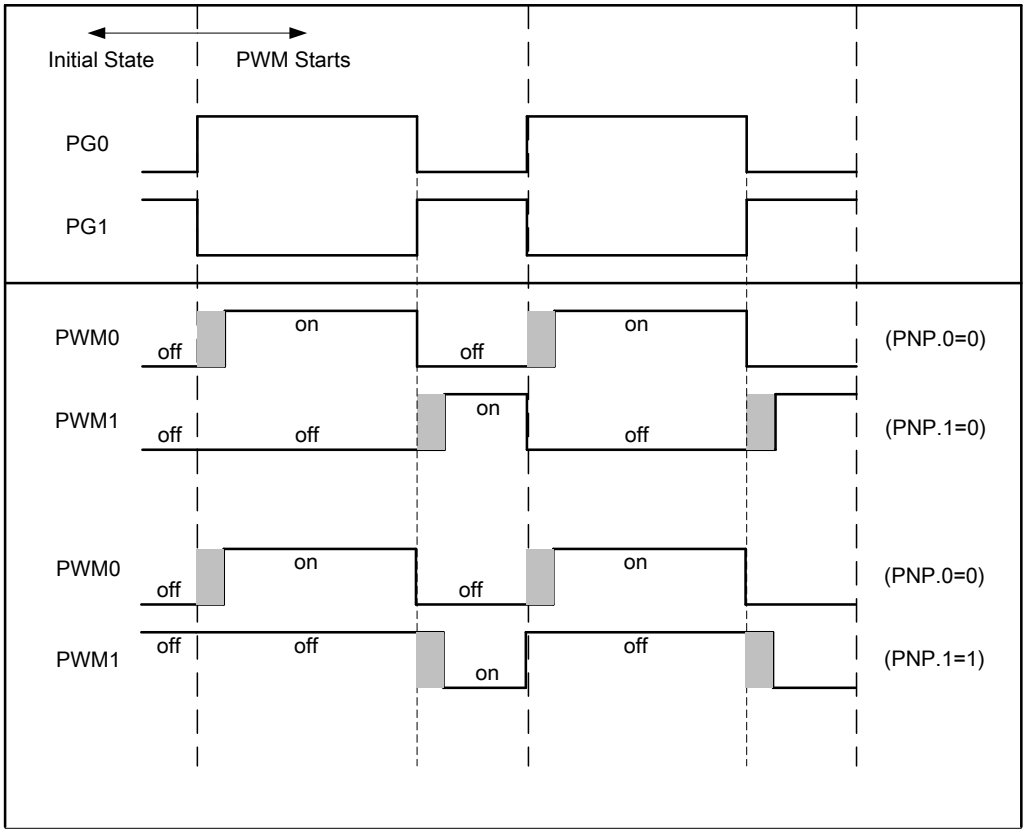
PG5 = PG3 = PG1 = 反向的 (PG0)(如果互补模式使能的话 (PWMMOD[1:0]=”01b”)

实际应用中, 请不要同时使能组和同步模式，因为同步模式电平不反转.

5.10.6 极性控制

PWM0 到 PWM5 的每个PWM口都有独立的极性控制来配置PWM输出激活时的极性.缺省情况下,PWM输出高.

下图显示了不同极性设定下PWM的初始状态.



NPx: Negative Polarity control bits; It controls the PWM output initial state and polarity

Dead-zone insertion; It is only effective in complementary mode

Note: Only Odd channels can be set inverter bit when dead-zone insertion and polarity control.

PWM output with dead-zone insertion and polarity control

图 5.10-17 初始状态和极性控制并在上升沿插入死区

5.10.7 PWM 马达控制中断架构

PWM单元有4种中断源: PWM 周期中断 (PWMPIF), 占空比中断(PWMDIF), 刹车0标志 (BKF0) 和刹车1标志(BKF1). 比特BRKIE (PIER[16]) 控制刹车中断使能; 比特PWMPIEn (PIER[0] ~ PIER[5]) 控制PWM周期中断使能; 比特PWMDIEn (PIER[8] ~ PIER[13]) 控制PWM占空比中断使能.

注意:所有的中断标志都由硬件置位，必须由软件清除..

图 5.10-18 演示了PWM中断的架构

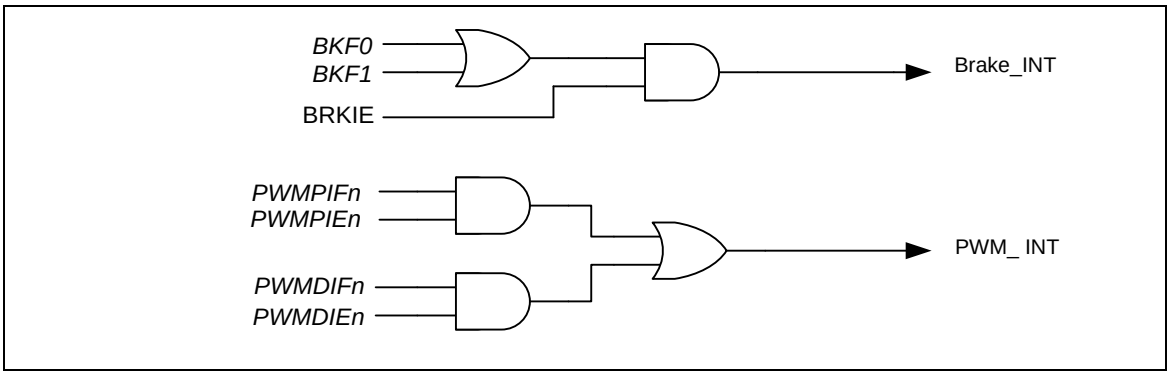


图 5.10-18 中断控制架构

5.10.8 PWM 刹车功能

这个设备支持2个外部刹车引脚: BKP0 和 BKP1 引脚. 刹车功能由PFBCON寄存器控制.

因为刹车发生时将自动设置BKF标志，用户可以轮询刹车标志位或者使能PWM刹车中断来确定是否发生了故障刹车.

注意:当刹车发生时, PWM0 ~ PWM5 使能比特将被硬件关闭，用户必须写PWM使能比特来退出刹车状态.

5.10.9 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	R/W	描述	复位值
PWM_BA = 0x4004_0000				
PPR	PWM_BA+0x00	R/W	PWM预分频寄存器	0x0000_0000
CSR	PWM_BA+0x04	R/W	PWM 时钟选择寄存器	0x0000_0000
PCR	PWM_BA+0x08	R/W	PWM控制寄存器	0x0000_0000
CNR0	PWM_BA+0x0C	R/W	PWM计数器寄存器0	0x0000_0000
CNR1	PWM_BA+0x10	R/W	PWM 计数器寄存器 1	0x0000_0000
CNR2	PWM_BA+0x14	R/W	PWM 计数器寄存器 2	0x0000_0000
CNR3	PWM_BA+0x18	R/W	PWM 计数器寄存器3	0x0000_0000
CNR4	PWM_BA+0x1C	R/W	PWM 计数器寄存器 4	0x0000_0000
CNR5	PWM_BA+0x20	R/W	PWM 计数器寄存器 5	0x0000_0000
CMR0	PWM_BA+0x24	R/W	PWM 比较器器寄存器0	0x0000_0000
CMR1	PWM_BA+0x28	R/W	PWM 比较器器寄存器 1	0x0000_0000
CMR2	PWM_BA+0x2C	R/W	PWM 比较器器寄存器 2	0x0000_0000
CMR3	PWM_BA+0x30	R/W	PWM 比较器器寄存器3	0x0000_0000
CMR4	PWM_BA+0x34	R/W	PWM 比较器器寄存器 4	0x0000_0000
CMR5	PWM_BA+0x38	R/W	PWM 比较器器寄存器 5	0x0000_0000
PIER	PWM_BA+0x54	R/W	PWM 中断使能寄存器	0x0000_0000
PIIR	PWM_BA+0x58	R/W	PWM 中断指示寄存器	0x0000_0000
PWMPOE	PWM_BA+0x5C	R/W	PWM通道 0~5输出使能寄存器	0x0000_0000
PFBCON	PWM_BA+0x60	R/W	PWM 故障刹车控制寄存器	0x0000_0000
PDZIR	PWM_BA+0x64	R/W	PWM死区间隔寄存器	0x0000_0000

5.10.10 PWM 控制器寄存器描述

PWM预分频寄存器(PPR)

寄存器	偏移	R/W	描述	复位值
PPR	PWM_BA+0x00	R/W	PWM预分频寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
CP45							
15	14	13	12	11	10	9	8
CP23							
7	6	5	4	3	2	1	0
CP01							

Bits	描述	
[31:24]	-	预留
[23:16]	CP45[7:0]	**时钟预分频器4 (PWM counter 4 & 5 共用)** 时钟在送给PWM计数器之前除以 (CP45 + 1). 如果CP45=0, 时钟预分频器 4时钟输出将停止. 所以相应的PWM计数器也将停止.
[15:8]	CP23[7:0]	**时钟预分频器2 (PWM counter 2 & 3共用)** 时钟在送给PWM计数器之前除以(CP23 + 1). 如果 CP23=0, 时钟预分频器 2时钟输出将停止. 所以相应的PWM计数器也将停止.
[7:0]	CP01[7:0]	**时钟预分频器0 (PWM counter 0 & 1共用)** 时钟在送给PWM计数器之前除以(CP01 + 1) 如果 CP01=0, 时钟预分频器 0时钟输出将停止. 所以相应的PWM计数器也将停止.

PWM 时钟选择寄存器(CSR)

寄存器	偏移	R/W	描述	复位值
CSR	PWM_BA+0x04	R/W	PWM时钟选择寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-	CSR5			-	CSR4		
15	14	13	12	11	10	9	8
-	CSR3			-	CSR2		
7	6	5	4	3	2	1	0
-	CSR1			-	CSR0		

Bits	描述	
[31:23]	-	**预留**
[22:20]	**CSR5[2:0]**	**Timer 5 时钟源选择** 选择PWM定时器5的输入时钟. **CSR5 [2:0]** / 输入时钟除以 100 / 1 011 / 16 010 / 8 001 / 4 000 / 2 101, 110, 111 / 预留
[19]	-	**预留**
[18:16]	**CSR4[2:0]**	**Timer 4时钟源选择** 选择PWM定时器4的输入时钟. (表与 CSR5相同)
[15]	-	**预留**
[14:12]	**CSR3[2:0]**	**Timer 3时钟源选择** 选择PWM定时器3的输入时钟. (表与 CSR5相同)

Bits	描述	
[11]	-	**预留**
[10:8]	**CSR2[2:0]**	**Timer 2时钟源选择** 选择PWM定时器2的输入时钟. (表与 CSR5相同)
[7]	-	**预留**
[6:4]	**CSR1[2:0]**	**Timer 1时钟源选择** 选择PWM定时器1的输入时钟. (表与 CSR5相同)
[3]	-	**预留**
[2:0]	**CSR0[2:0]**	**Timer 0时钟源选择** 选择PWM定时器0的输入时钟. (表与 CSR5相同)

PWM 控制寄存器(PCR)

寄存器	偏移	R/W	描述	复位值
PCR	PWM_BA+0x08	R/W	PWM 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PWMTYPE	GRP	PWMMOD		CLRPWM	DZEN45	DZEN23	DZEN01
23	22	21	20	19	18	17	16
CH5MOD	CH5INV	-	CH5EN	CH4MOD	CH4INV	-	CH4EN
15	14	13	12	11	10	9	8
CH3MOD	CH3INV	-	CH3EN	CH2MOD	CH2INV	-	CH2EN
7	6	5	4	3	2	1	0
CH1MOD	CH1INV	-	CH1EN	CH0MOD	CH0INV	DB_MODE	CH0EN

Bits	描述	
[31]	PWMTYPE	**PWM对齐类型选择** 0 = 边沿对齐类型. 1 = 中心对齐类型.
[30]	GRP	**Group比特** 1 = 统一PWM0，PWM2和PWM4的信号时序,使相位相同,都由PWM0控制. 0 = PWM0, PWM2 和 PWM4信号时序是独立的.
[29:28]	PWMMOD[1:0]	**PWM操作模式选择** PWMMOD[1:0] / 模式 00 / 独立模式 01 / 互补模式 10 / 同步模式 11 / 预留
[27]	CLRPWM	**清除PWM 定时器控制比特** 1 = 清除16位的PWM计数器为 “000H”. 0 = 不要清除PWM 计数器. **注意:**该比特由硬件自动清除.

Bits	描述	
[26]	**DZEN45**	**死区 4 发生器 使能/禁止(用于PWM4 和 PWM5 对)** 1 = 使能. 0 = 禁止. **注意:** 当死区发生器使能时, PWM4 和PWM5 变成互补对.
[25]	**DZEN23**	**死区 2 发生器 使能/禁止(用于PWM2 和 PWM3 对)** 1 = 使能. 0 = 禁止. **注意:** 当死区发生器使能时, PWM2 和PWM3 变成互补对.
[24]	**DZEN01**	**死区 0 发生器 使能/禁止(用于PWM0 和 PWM1 对)** 1 = 使能. 0 = 禁止. **注意:** 当死区发生器使能时, PWM0 和PWM1 变成互补对.
[23]	**CH5MOD**	**PWM-Timer 5 自动加载/One-Shot 模式** 1 = 自动加载模式. 0 = One-shot 模式. **注意:** 如果该比特有一个上升沿跳变, 将导致CNR5 和 CMR5 被清除.
[22]	**CH5INV**	**PWM-Timer 5 输出反转ON/OFF** 1 = 反转 ON. (在PWM独立模式只有偶数通道(PWM0, PWM2, PWM4)可以使能反转) 0 = 反转 OFF.
[21]	**-**	**预留**
[20]	**CH5EN**	**使能/禁止PWM-Timer 5开始计数** 1 = 使能PWM-Timer 开始计数. 0 = 停止PWM-Timer 计数.
[19]	**CH4MOD**	**PWM-Timer 4自动加载/One-Shot 模式** 1 = 自动加载模式. 0 = One-shot 模式. **注意:** 如果该比特有一个上升沿跳变, 将导致CNR4 和 CMR4 被清除.
[18]	**CH4INV**	**PWM-Timer 4 输出反转ON/OFF** 1 = 反转 ON. 0 = 反转 OFF.
[17]	**-**	**预留**
[16]	**CH4EN**	**使能/禁止PWM-Timer 4开始计数** 1 = 使能PWM-Timer 开始计数. 0 = 停止PWM-Timer 计数.

Bits	描述	
[15]	**CH3MOD**	**PWM-Timer 3 自动加载/One-Shot 模式** 1 = 自动加载模式. 0 = One-shot 模式. **注意:** 如果该比特有一个上升沿跳变, 将导致CNR3 和 CMR3 被清除.
[14]	**CH3INV**	**PWM-Timer 3 输出反转ON/OFF** 1 = 反转 ON. (在PWM独立模式只有偶数通道(PWM0, PWM2, PWM4)可以使能反转) 0 = 反转 OFF.
[13]	**-**	**预留**
[12]	**CH3EN**	**使能/禁止PWM-Timer 3开始计数** 1 = 使能PWM-Timer 开始计数. 0 = 停止PWM-Timer 计数.
[11]	**CH2MOD**	**PWM-Timer 2 自动加载/One-Shot 模式** 1 = 自动加载模式. 0 = One-shot 模式. **注意:** 如果该比特有一个上升沿跳变, 将导致CNR2 和 CMR2 被清除.
[10]	**CH2INV**	**PWM-Timer 2 输出反转ON/OFF** 1 = 反转 ON. 0 = 反转 OFF.
[9]	**-**	**预留**
[8]	**CH2EN**	**使能/禁止PWM-Timer 2开始计数** 1 = 使能PWM-Timer 开始计数. 0 = 停止PWM-Timer 计数.
[7]	**CH1MOD**	**PWM-Timer 1 自动加载/One-Shot 模式** 1 = 自动加载模式. 0 = One-shot 模式. **注意:** 如果该比特有一个上升沿跳变, 将导致CNR1 和 CMR1 被清除.
[6]	**CH1INV**	**PWM-Timer 1 输出反转ON/OFF** 1 = 反转 ON. (在PWM独立模式只有偶数通道(PWM0, PWM2, PWM4)可以使能反转) 0 = 反转 OFF.
[5]	**-**	**预留**
[4]	**CH1EN**	**使能/禁止PWM-Timer 1开始计数** 1 = 使能PWM-Timer 开始计数. 0 = 停止PWM-Timer 计数.

Bits	描述	
[3]	**CH0MOD**	**PWM-Timer 0 自动加载/One-Shot 模式** 1 = 自动加载模式. 0 = One-shot 模式. **注意:** 如果该比特有一个上升沿跳变, 将导致CNR0 和 CMR0 被清除.
[2]	**CH0INV**	**PWM-Timer 0 输出反转ON/OFF** 1 = 反转 ON. 0 = 反转 OFF.
[1]	**DB_MODE**	**PWM 调试模式配置比特(只在 DEBUG 模式下有作用)** 1 = 正常模式: 调试时,定时器继续正常计数，因为一个固定的占空比被应用于逆变器，可能一些情况下这是危险的， (不再发生中断). 0 = 安全模式: 定时器冻结并且PWM输出停止.退出调试模式时定时器将从停止的地方重新开始.
[0]	**CH0EN**	**使能/禁止PWM-Timer 0开始计数** 1 =使能PWM-Timer 开始计数. 0 =停止PWM-Timer计数.

PWM 计数器寄存器0-5 (CNR0-5)

寄存器	偏移	R/W	描述	复位值
CNR0	PWM_BA+0x0C	R/W	PWM计数器寄存器0	0x0000_0000
CNR1	PWM_BA+0x10	R/W	PWM 计数器寄存器1	0x0000_0000
CNR2	PWM_BA+0x14	R/W	PWM 计数器寄存器 2	0x0000_0000
CNR3	PWM_BA+0x18	R/W	PWM 计数器寄存器3	0x0000_0000
CNR4	PWM_BA+0x1C	R/W	PWM 计数器寄存器4	0x0000_0000
CNR5	PWM_BA+0x20	R/W	PWM 计数器寄存器5	0x0000_0000

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

Bits	描述	
[31:16]	-	**预留**
[15:0]	**CNRn** **n = 0 – 5**	**PWM 计数器/定时器加载值** CNRn 决定PWM的周期. 边沿对齐模式: ● PWM 频率 = PWMxy_CLK/((prescale+1)*(clock divider))/(CNRn+1); xy 依靠选择的PWM通道可以是01, 23, 45. ● 占空比 = (CMRn+1)/(CNRn+1). ● CMRn >= CNRn: PWM 输出总是高. ● CMRn < CNRn: PWM 低电平宽度= (CNRn-CMRn) unit; PWM 高电平宽度 = (CMR+1) unit. ● CMRn = 0: PWM低电平宽度= (CNRn) unit; PWM高电平宽度= 1 unit. 中心对齐模式: PWM 频率 = PWMxy_CLK/((prescale+1)*(clock divider))/ (2*CNRn+1); xy 依靠选择的PWM通道可以是01, 23, 45. ● 占空比 = (CNRn - CMRn)/(CNRn+1).

Bits	描述	
		● CMRn >= CNRn: PWM输出指示为低. ● CMRn < CNRn: PWM 低电平宽度= (CMRn + 1) x 2 unit; PWM 高电平宽度 = (CNRn - CMRn) x 2 unit. ● CMRn = 0: PWM 低电平宽度 = 2 unit; PWM 高电平宽度 = (CNRn) x 2 unit. (Unit = 一个PWM时钟周期) **注意: 1. 写CNRn 在下一个PWM周期起作用.** **2. n = 0, 1, 2, 3, 4, 5.**

PWM 比较器寄存器0-5 (CMR0-5)

寄存器	偏移	R/W	描述	复位值
CMR0	PWM_BA+0x24	R/W	PWM比较器寄存器0	0x0000_0000
CMR1	PWM_BA+0x28	R/W	PWM 比较器寄存器1	0x0000_0000
CMR2	PWM_BA+0x2C	R/W	PWM 比较器寄存器 2	0x0000_0000
CMR3	PWM_BA+0x30	R/W	PWM 比较器寄存器3	0x0000_0000
CMR4	PWM_BA+0x34	R/W	PWM 比较器寄存器 4	0x0000_0000
CMR5	PWM_BA+0x38	R/W	PWM 比较器寄存器 5	0x0000_0000

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

Bits	描述	
[31:16]	-	**预留**
[15:0]	**CMRn** **n = 0 – 5**	**PWM 比较器寄存器** CMR 定义PWM的占空比. 边沿对齐模式: ● PWM 频率 = PWMxy_CLK/((prescale+1)*(clock divider))/(CNRn+1); xy取决于选择的PWM通道可以是01, 23, 45. ● 占空比= (CMRn+1)/(CNRn+1). ● CMRn >= CNRn: PWM 总是输出高. ● CMRn < CNRn: PWM低电平宽度= (CNRn-CMRn) unit; PWM高电平宽度= (CMR+1) unit. ● CMRn = 0: PWM低电平宽度= (CNRn) unit; PWM高电平宽度= 1 unit 中心对齐模式: ● PWM 频率 = PWMxy_CLK/((prescale+1)*(clock divider)) /(2*CNRn+1); xy 取决于选择的PWM通道可以是01, 23, 45. ● 占空比= (CNRn - CMRn)/(CNRn+1).

Bits	描述	
		● CMRn >= CNRn: PWM输出总是低. ● CMRn < CNRn: PWM低电平宽度= (CMRn + 1) x 2 unit; PWM高电平宽度= (CNRn - CMRn) x 2 unit. ● CMRn = 0: PWM低电平宽度= 2 unit; PWM高电平宽度= (CNRn) x 2 unit (Unit = 一个PWM时钟周期) **注意: 1. 写CMRn 在下一个PWM周期起作用.** **2. n = 0, 1, 2, 3, 4, 5**.

PWM 中断使能寄存器(PIER)

寄存器	偏移	R/W	描述	复位值
PIER	PWM_BA+0x54	R/W	PWM 中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-						INT_TYPE	BRKIE
15	14	13	12	11	10	9	8
-		PWMDIE5	PWMDIE4	PWMDIE3	PWMDIE2	PWMDIE1	PWMDIE0
7	6	5	4	3	2	1	0
-		PWMPIE5	PWMPIE4	PWMPIE3	PWMPIE2	PWMPIE1	PWMPIE0

Bits	描述	
[31:18]	-	预留
[17]	INT_TYPE	PWM 中断类型选择 1 = 当PWM计数器与CNRn寄存器的值匹配时PWMPIFn 将被置位. 0 = 当PWM计数器下溢时PWMPIFn 将被置位. 注意: 当PWM工作在中心对齐模式时该比特才起作用.
[16]	BRKIE	使能故障刹车0 和 1 的中断 1 = 使能故障刹车中断. 0 =禁止故障刹车中断.
[15:14]	-	预留
[13]	PWMDIE5	PWM 通道 5 占空比中断使能 1 = 使能. 0 = 禁止.
[12]	PWMDIE4	PWM 通道 4 占空比中断使能 1 = 使能. 0 = 禁止.
[11]	PWMDIE3	PWM 通道 3 占空比中断使能 1 = 使能. 0 = 禁止.

Bits	描述	
[10]	**PWMDIE2**	**PWM 通道 2 占空比中断使能** 1 = 使能. 0 = 禁止.
[9]	**PWMDIE1**	**PWM 通道 1 占空比中断使能** 1 = 使能. 0 = 禁止.
[8]	**PWMDIE0**	**PWM 通道 0 占空比中断使能** 1 = 使能. 0 = 禁止.
[7:6]	**-**	**预留**
[5]	**PWMPIE5**	**PWM通道 5 周期中断使能** 1 =使能. 0 =禁止.
[4]	**PWMPIE4**	**PWM通道 4 周期中断使能** 1 =使能. 0 =禁止.
[3]	**PWMPIE3**	**PWM通道 3 周期中断使能** 1 =使能. 0 =禁止.
[2]	**PWMPIE2**	**PWM通道 2 周期中断使能** 1 =使能. 0 =禁止.
[1]	**PWMPIE1**	**PWM通道 1 周期中断使能** 1 =使能. 0 =禁止.
[0]	**PWMPIE0**	**PWM通道 0 周期中断使能** 1 =使能. 0 =禁止.

PWM 中断指示寄存器(PIIR)

寄存器	偏移	R/W	描述	复位值
PIIR	PWM_BA+0x58	R/W	PWM中断指示寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-						BKF1	BKF0
15	14	13	12	11	10	9	8
-		PWMDIF5	PWMDIF4	PWMDIF3	PWMDIF2	PWMDIF1	PWMDIF0
7	6	5	4	3	2	1	0
-		PWMPIF5	PWMPIF4	PWMPIF3	PWMPIF2	PWMPIF1	PWMPIF0

Bits	描述	
[31:18]	-	**预留**
[17]	BKF1	**PWM 刹车1 标志** 1 = 当PWM刹车在BKP1引脚探测到一个下降沿时,这个标志将被置为高. 0 = PWM 刹车在BKP1引脚没有探测到下降沿. **注意:** 该比特必须由软件写”1”来清除.
[16]	BKF0	**PWM 刹车0 标志** 1 = 当PWM刹车在BKP0引脚探测到一个下降沿时,这个标志将被置为高. 0 = PWM 刹车在BKP0没有探测到下降沿. **注意:** 该比特必须由软件写”1”来清除.
[15:14]	-	**预留**
[13]	PWMDIF5	**PWM 通道 5 占空比中断标志** 当通道5的 PWM计数器达到CMR5的值时该比特将被置,软件可以写 “1”清除.
[12]	PWMDIF4	**PWM 通道 4 占空比中断标志** 当通道4的 PWM计数器达到CMR4的值时该比特将被置,软件可以写 “1”清除.
[11]	PWMDIF3	**PWM 通道 3 占空比中断标志** 当通道3的 PWM计数器达到CMR3的值时该比特将被置,软件可以写 “1”清除.
[10]	PWMDIF2	**PWM 通道 2 占空比中断标志** 当通道2的 PWM计数器达到CMR2的值时该比特将被置,软件可以写 “1”清除.

Bits	描述	
[9]	**PWMDIF1**	**PWM 通道 1 占空比中断标志** 当通道1的 PWM计数器达到CMR1的值时该比特将被置,软件可以写 “1”清除.
[8]	**PWMDIF0**	**PWM 通道 0 占空比中断标志** 当通道0的 PWM计数器达到CMR0的值时这个比特将被置,软件可以写 “1”清除.
[7:6]	**-**	**预留**
[5]	**PWMPIF5**	**PWM 通道 5 周期中断标志** 当CNR5减到0时这个比特将被置,软件可以写 “1”清除
[4]	**PWMPIF4**	**PWM 通道 4 周期中断标志** 当CNR4减到0时这个比特将被置,软件可以写 “1”清除
[3]	**PWMPIF3**	**PWM 通道 3 周期中断标志** 当CNR3减到0时这个比特将被置, 软件可以写 “1”清除
[2]	**PWMPIF2**	**PWM 通道 2 周期中断标志** 当CNR2减到0时这个比特将被置, 软件可以写 “1”清除
[1]	**PWMPIF1**	**PWM 通道 1 周期中断标志** 当CNR1减到0时这个比特将被置, 软件可以写 “1”清除
[0]	**PWMPIF0**	**PWM 通道 0 周期中断标志** 当CNR0减到0时这个比特将被置, 软件可以写 “1”清除

注意: 用户能写”1”清除每个中断标志.

PWM 输出控制寄存器(PWMPOE)

寄存器	偏移	R/W	描述	复位值
PWMPOE	PWM_BA+0x5C	R/W	PWM 通道0~5的输出控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-							
7	6	5	4	3	2	1	0
-		PWM5	PWM4	PWM3	PWM2	PWM1	PWM0

Bits	描述	
[31:6]	-	预留
[5]	PWM5	**PWM 通道 5 输出使能** 1 = 使能 PWM 通道 5 输出到引脚上. 0 = 禁止 PWM 通道 5 输出到引脚上. **注意:** 相应的GPIO脚也必须切成PWM功能.
[4]	PWM4	**PWM 通道 4 输出使能** 1 = 使能 PWM 通道 4 输出到引脚上. 0 = 禁止 PWM 通道 4 输出到引脚上. **注意:** 相应的GPIO脚也必须切成PWM功能.
[3]	PWM3	**PWM 通道 3输出使能** 1 = 使能 PWM 通道 3 输出到引脚上. 0 = 禁止 PWM 通道 3 输出到引脚上. **注意:** 相应的GPIO脚也必须切成PWM功能.
[2]	PWM2	**PWM 通道 2 输出使能** 1 = 使能 PWM 通道 2 输出到引脚上. 0 = 禁止 PWM 通道 2 输出到引脚上. **注意:** 相应的GPIO脚也必须切成PWM功能.

Bits	描述	
[1]	**PWM1**	**PWM 通道 1输出使能** 1 = 使能 PWM 通道 1 输出到引脚上. 0 =禁止 PWM 通道 1 输出到引脚上. **注意:** 相应的GPIO脚也必须切成PWM功能.
[0]	**PWM0**	**PWM 通道 0 输出使能** 1 = 使能 PWM 通道 0 输出到引脚上. 0 = 禁止 PWM 通道 0 输出到引脚上. **注意:** 相应的GPIO脚也必须切成PWM功能.

PWM 故障刹车控制寄存器(PFBCON)

寄存器	偏移	R/W	描述	复位值
PFBCON	PWM_BA+0x60	R/W	PWM故障刹车控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-		PWMBKO5	PWMBKO4	PWMBKO3	PWMBKO2	PWMBKO1	PWMBKO0
23	22	21	20	19	18	17	16
-							
15	14	13	12	11	10	9	8
-							
7	6	5	4	3	2	1	0
BKF	-				CPO0BKEN	BKEN1	BKEN0

Bits	描述	
[31:30]	-	预留
[29]	PWMBKO5	PWM 通道 5 刹车时输出电平选择 1 = 当刹车条件发生时PWM 输出高电平. 0 =当刹车条件发生时PWM 输出低电平.
[28]	PWMBKO4	PWM 通道 4 刹车时输出电平选择 1 = 当刹车条件发生时PWM 输出高电平. 0 =当刹车条件发生时PWM 输出低电平.
[27]	PWMBKO3	PWM 通道 3 刹车时输出电平选择 1 = 当刹车条件发生时PWM 输出高电平. 0 =当刹车条件发生时PWM 输出低电平.
[26]	PWMBKO2	PWM 通道 2 刹车时输出电平选择 1 = 当刹车条件发生时PWM 输出高电平. 0 =当刹车条件发生时PWM 输出低电平.
[25]	PWMBKO1	PWM 通道 1刹车时输出电平选择 1 = 当刹车条件发生时PWM 输出高电平. 0 =当刹车条件发生时PWM 输出低电平.
[24]	PWMBKO0	PWM 通道 0 刹车时输出电平选择 1 = 当刹车条件发生时PWM 输出高电平. 0 =当刹车条件发生时PWM 输出低电平.

Bits	描述	
[23:8]	-	**预留**
[7]	**BKF**	**PWM 故障刹车事件标志(写“1”清除)** 1 = 当故障刹车条件发生时,PWM输出故障刹车状态. 0 =当故障刹车条件发生时,PWM输出初始状态.
[6:3]	-	**预留**
[2]	**CPO0BKEN**	**故障刹车1(BKP1)的刹车源选择** 1 = CPO0作为刹车1的刹车源. 0 = EINT1作为刹车1的刹车源.
[1]	**BKEN1**	**使能 BKP1 引脚触发故障刹车功能** 0 = 禁止BKP1 引脚触发刹车功能1. 1 =使能 BKP1 引脚触发刹车功能1，引脚上一个下降沿转变将触发刹车. BKP1刹车使用EINT1 输入引脚还是比较器 CPO0的输出由CPO0BKEN 决定. 也就是说，使能BKP1刹车功能以后，如果CPO0BKEN=0, EINT1引脚上一个下降沿转变将触发刹车; 如果CPO0BKEN=1, 比较器CPO0的输出改变将触发刹车
[0]	**BKEN0**	**使能 BKP0 引脚触发故障刹车功能** 0 =禁止 BKP0 引脚触发刹车功能0 1 =使能 BKP0引脚触发刹车功能0，引脚上一个下降沿转变将触发刹车(EINT0). BKP0刹车使用EINT0输入引脚，使能BKP0刹车功能以后，EINT0引脚上一个下降沿转变将触发刹车

PWM 死区间隔寄存器(PDZIR)

寄存器	偏移	R/W	描述	复位值
PDZIR	PWM_BA+0x64	R/W	PWM 死区间隔寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DZI45							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DZI23							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DZI01							

Bits	描述	
[31:24]	**-**	**预留**
[23:16]	**DZI45[7:0]**	**设定通道4和通道5**(PWM4 和 PWM5 对)**的死区间隔**. 这 8个比特决定了死区的长度. 死区长度单位由CSR比特决定.
[15:8]	**DZI23[7:0]**	**设定通道2和通道3**(PWM2 和 PWM3对)**的死区间隔**. 这 8个比特决定了死区的长度. 死区长度单位由CSR比特决定.
[7:0]	**DZI01[7:0]**	**设定通道0和通道1**(PWM0 和 PWM1 对)**的死区间隔**. 这 8个比特决定了死区的长度. 死区长度单位由CSR比特决定.

5.11 串行外设接口(SPI) 控制器

5.11.1 概述

串行外设接口(SPI) 是一个同步串行数据通讯协议,工作在全双工模式. 设备支持主/从模式通讯，4线双向接口. NuMicro MINI51™ 系列包含一组SPI控制器,从外设收到数据时执行串到并的转换,发送数据到外设时执行并到串的转换.SPI控制器可以设为主模式也可以设为从模式,由接在SPI上的外设控制.

5.11.2 特性

- 支持主或者从模式
- MSB或者 LSB 优先发送
- 字节或者word 暂停(Suspend)模式
- 主模式时输出的串行时钟频率可变
- 主模式时支持输出两个可编程的串行时钟频率

5.11.3 SPI 方块图

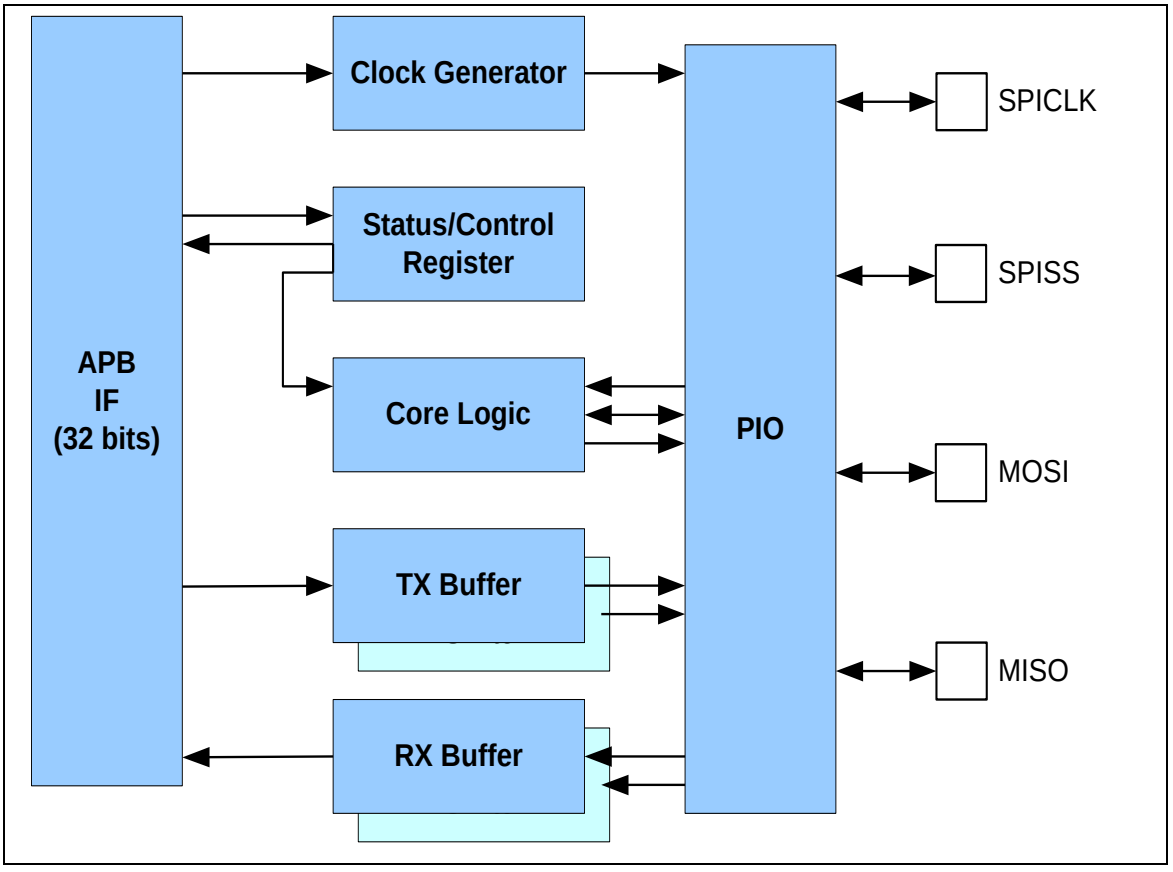


图 5.11-1 SPI 方块图

5.11.4 SPI 功能描述

5.11.4.1 主/从模式

通过设定**SLAVE** 比特(SPI_CNTRL[18]),SPI 控制器可以设为主或者从模式. 主/从模式应用方块图如下所示.

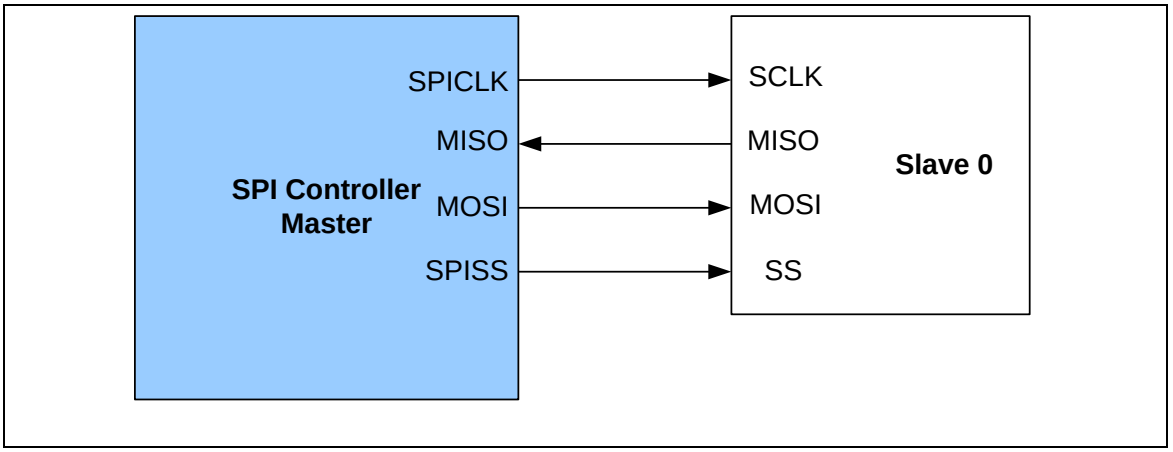


图 5.11-2 SPI 主模式应用方块图

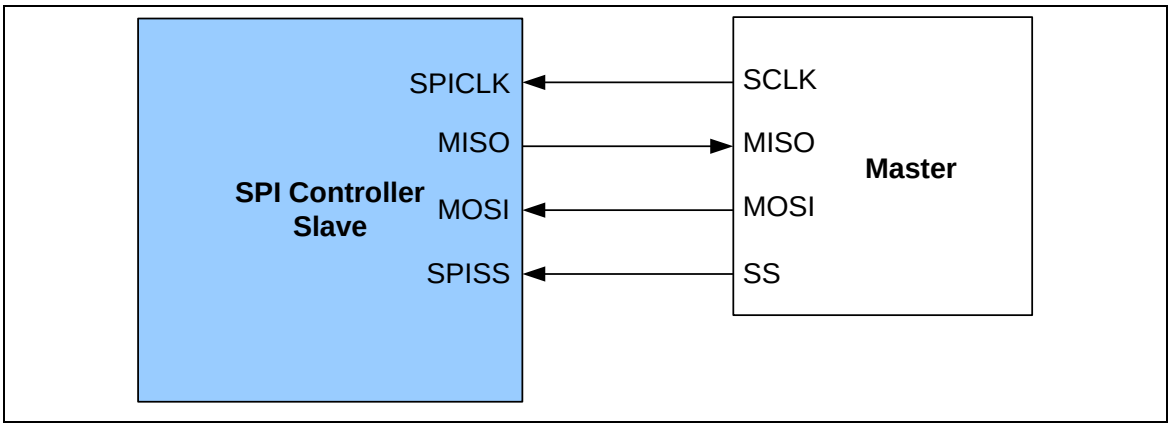


图 5.11-3 SPI 从模式应用方块图

5.11.4.2 从模式选择

主模式下, SPI 控制器可以通过SPISS片选驱动片外从设备. 从模式下, 片外主设备驱动SPISS片选信号给SPI控制器. 主/从模式下片选信号的激活电平可以通过SS_LVL 比特 (SPI_SSR[2])配置成低电平或者高电平激活, 比特SS_LTRIG (SPI_SSR[4]) 定义片选信号SPISS为电平触发还是边沿触发. 触发条件依靠连接的从/主设备.

5.11.4.3 自动片选

主模式下,如果ASS比特(SPI_SSR[3])被设, 将根据SSR比特 (SPI_SSR[0])自动产生片选信号到SPISS引脚. 这意味着通过设定GO_BUSY比特(SPI_CNTRL[0])开始收/发时,SPI控制器将自动设定片选信号.数据收发完成时自动取消片选. 如果ASS比特被清除,片选信号由软件通过设定和清除SPI_SSR[0]比特发起和取消. 片选信号激活电平由SS_LVL 比特 (SPI_SSR[2])决定.

5.11.4.4 串行时钟

主模式下,写一个除频值到DIVIDER (SPI_DIVIDER[15:0]) 寄存器来编程SPICLK输出时钟的频率. 如果VARCLK_EN 比特 (SPI_CNTRL[23]) 被打开,也支持可变频率功能, 这种情况下串行时钟的每个比特的输出频率可以编程为两个不同的频率中的一个, 这两个频率由DIVIDER 和DIVIDER2(SPI_DIVIDER[31:16]) 决定. 每个比特的可变频率由寄存器VARCLK (SPI_VARCLK[31:0]) 决定. 从模式下,片外主设备通过SPICLK驱动串行时钟输入到SPI控制器.

5.11.4.5 时钟极性

主模式下,CLKP 比特 (SPI_CNTRL[11]) 定义串行时钟空闲时的极性. 如果CLKP = 1,空闲时SPICLK输出高电平,否则如果CLKP = 0输出低电平. 可变串行时钟只能工作在CLKP = 0的情况.

5.11.4.6 收/发比特长度

每次传输的比特长度由Tx_BIT_LEN (SPI_CNTRL[7:3])定义. 最大一次传输32个比特.

5.11.4.7 收/发笔数

一次传输的收/发笔数由Tx_NUM (SPI_CNTRL[9:8])定义. 但是只能设为“00b” 或者 “01b” , 收/发一笔或者两笔.

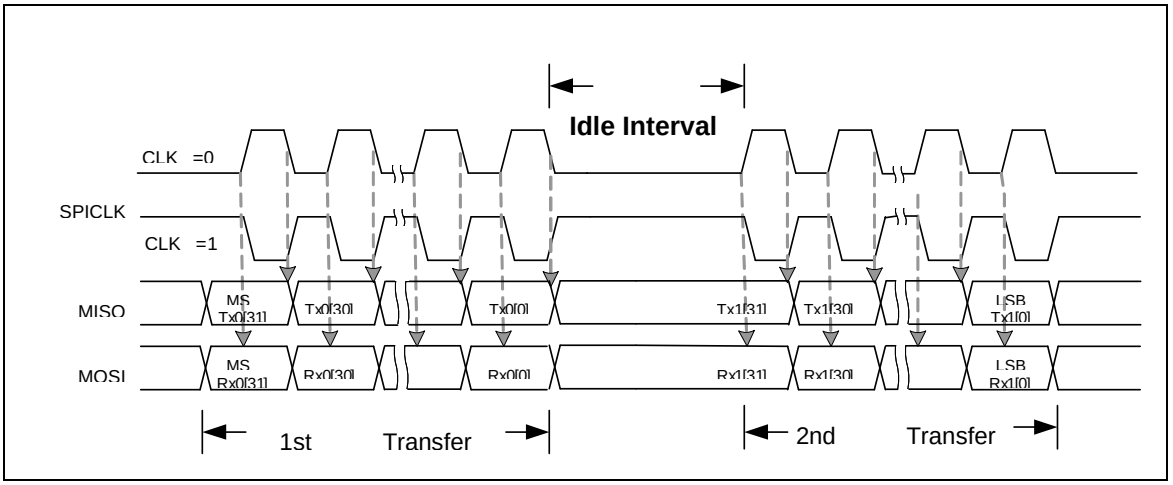


图 5.11-4 一次传输传输两笔(突发模式) 数据

5.11.4.8 LSB 优先

LSB 比特 (SPI_CNTRL[10]) 定义了数据传输LSB还是MSB优先发送.

5.11.4.9 发送沿

Tx_NEG 比特 (SPI_CNTRL[2]) 定义数据在SPICLK时钟的上升沿还是下降沿发送.

5.11.4.10 接收沿

Rx_NEG 比特 (SPI_CNTRL[1]) 定义数据在SPICLK时钟的上升沿还是下降沿接收.

5.11.4.11 Word 暂停(Suspend)

主模式下,SP_CYCLE (SPI_CNTRL[15:12]) 有4个比特域配置连续两次传输之间暂停2 ~ 17个串行时钟周期. 如果CLKP = 0,暂停间隔是从上次传输最后一个时钟下降沿到下次传输第一个时钟上升沿. 如果CLKP = 1,暂停间隔是从上次传输最后一个时钟上升沿到下次传输第一个时钟下降沿. SP_CYCLE 的缺省值是 “0” (2 个串行时钟周期), 但是如果Tx_NUM = “00b”,设定这些比特没有作用.

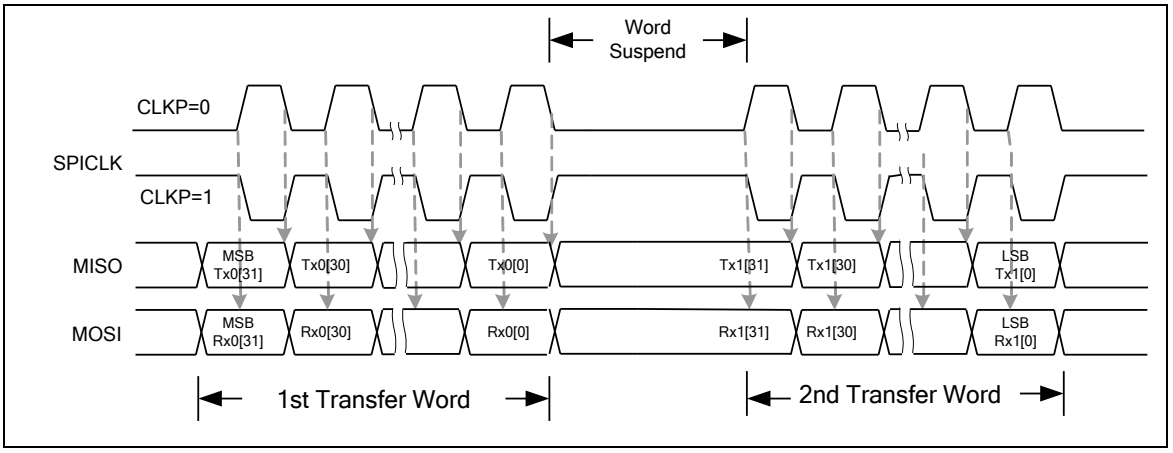


Figure 5.11-5 Word Suspend 模式

5.11.4.12 字节重新排序

当设定MSB优先(LSB = 0)并且REORDER使能时, 如果Tx_BIT_LEN = 32-bit,存放在TX和RX缓冲中的数据将按照[BYTE0, BYTE1, BYTE2, BYTE3]的顺序重新排序.收/发数据的顺序为BYTE0, BYTE1, BYTE2, 然后 BYTE3. 如果Tx_BIT_LEN = 24-bit,存放在TX和RX缓冲中的数据将按照[unknown byte, BYTE0, BYTE1, BYTE2]的顺序重新排序.收/发数据的顺序为BYTE0, BYTE1, BYTE2. 16-bit 模式的规则与上面相同.

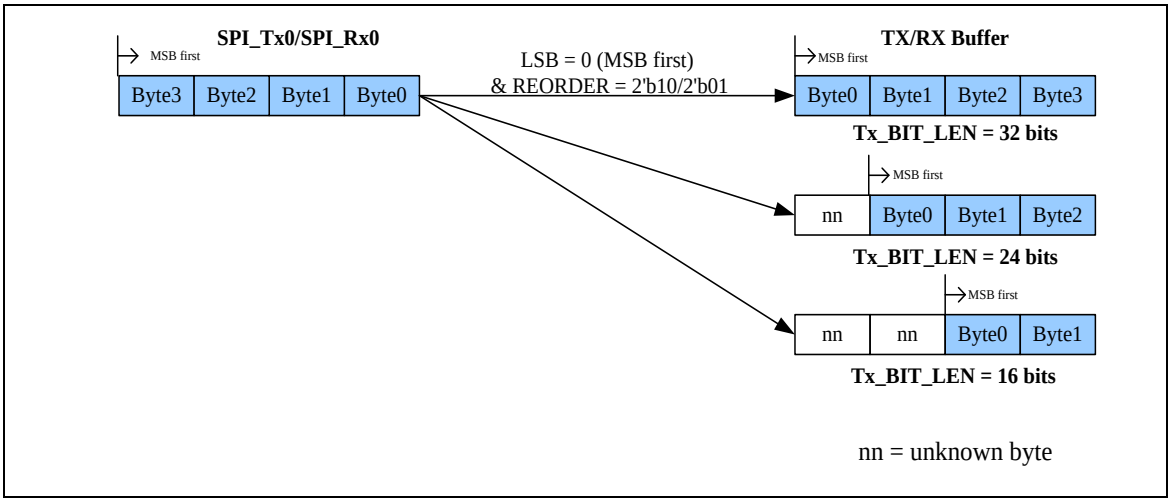


图 5.11-6 字节重新排序

5.11.4.13 字节 Suspend

主模式下, 如果 SPI_CNTRL[19] 设成 “1”, 硬件将在一次传输的连续2个字节之间插入2 ~ 17个串行时钟周期. 字节suspend 设定与word suspend设定相同都使用SP_CYCLE寄存器相同的比特域. 注意当使能字节suspend功能时, Tx_BIT_LEN必须编程为 “00H” (32 比特传输长度).

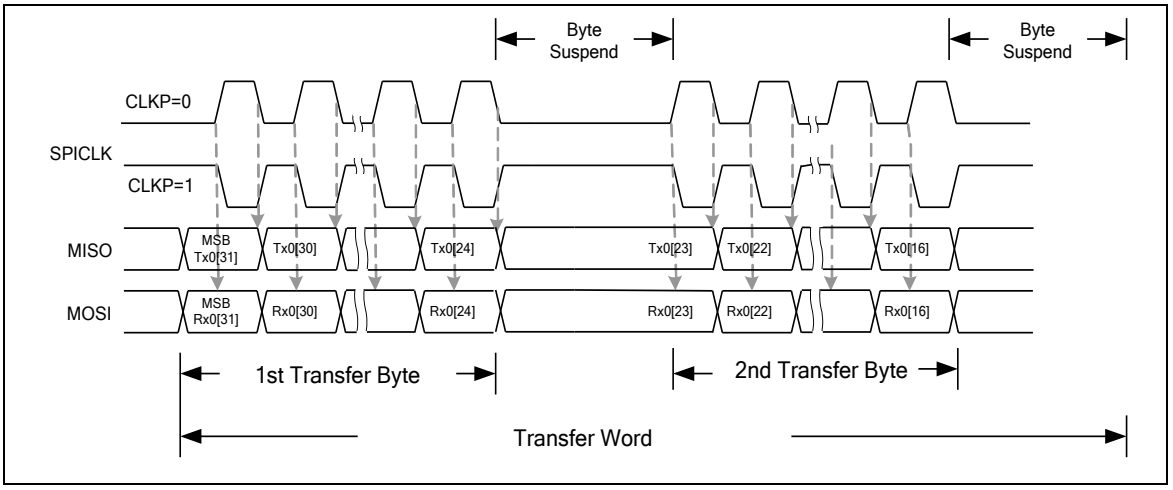


图 5.11-7 字节 Suspend 模式

寄存器	描述
00	禁止字节重新排序和字节 暂停功能.
01	使能字节重新排序和每个字节之间插入字节暂停间隔 (2~17 SPICLK 周期). Tx_BIT_LEN 必须设为 “00H” (32-bit/word).

10	使能字节重新排序, 但是禁止字节暂停功能.
11	禁止字节重新排序但是使能每个字节之间插入字节暂停间隔 (2~17 SPICLK 周期). Tx_BIT_LEN 必须设为 “00H” (32-bit/word).

表 5.11-1 字节顺序和字节时钟空闲内部条件

5.11.4.14 中断

当数据传输完成时,每个SPI控制器能产生独立的中断源并且相应的中断事件标志(SPI_CNTRL[16])将被设.如果中断使能IE (SPI_CNTRL[17]) 被设,中断事件标志将产生中断通知CPU. 中断事件标志写”1”清除.

5.11.4.15 可变串行时钟频率

主模式下,如果可变时钟使能位VARCLK_EN (SPI_CNTRL[23])被使能,串行时钟的输出可以编程为可变频率模型.频率模型格式由(SPI_VARCLK[31:0]) 寄存器定义.如果VARCLK的比特内容为”0”输出频率根据DIVIDER (SPI_DIVIDER[15:0])计算; 如果VARCLK的比特内容为”1”输出频率根据DIVIDER2 (SPI_DIVIDER[31:16])计算. 下图是串行时钟(SPICLK),VARCLK,DIVIDER和DIVIDER2之间关系的时序图. VARCLK每2个比特定义一个时钟周期. 比特域VARCLK[31:30] 定义SPICLK的第一个时钟周期. 比特域VARCLK[29:28] 定义SPICLK的第二个时钟周期，以此类推. 时钟源由寄存器VARCLK定义,硬件会自动在最前面插入与比特31相同的值. 例如, 如果SPICLK的前面有 5 个CLK1 周期, 在VARCLK的最高位 应该设 9 个”0”.第10个比特应该为”1". 注意当使能VARCLK_EN比特时, Tx_BIT_LEN 必须设为 “10H” (16比特模式).

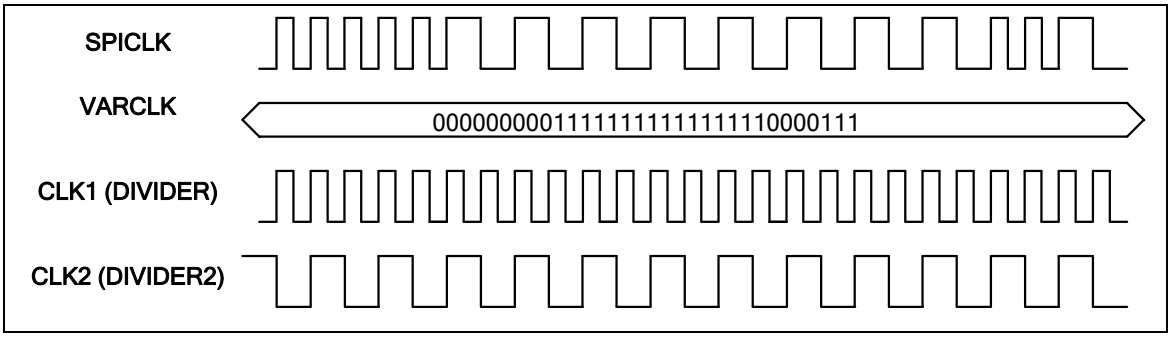


图 5.11-8 可变串行时钟频率

5.11.4.16 SPI 时序图

主/从模式下,片选信号(SPISS)的激活电平可以编程为高电平或者低电平,由SS_LVL 比特(SPI_SSR[2])控制, 但是SPISS 是电平还是边沿触发由SS_LTRIG 比特 (SPI_SSR[4])控制. 串行时钟(SPICLK) 空闲时的状态可以配置成高电平或者低电平,由CLKP 比特 (SPI_CNTRL[11])控制. 一次传输的比特长度由Tx_BIT_LEN (SPI_CNTRL[7:3])控制, 传输笔数由Tx_NUM (SPI_CNTRL[9:8])控制, 收/发数据MSB还是LSB优先由LSB 比特 (SPI_CNTRL[10])控制. 用户也可以选择串行时钟的哪一个沿发送/接收，由Tx_NEG/Rx_NEG (SPI_CNTRL[2:1]) 控制. 主/从模式下4个SPI时序图和相

关设定如下所示.

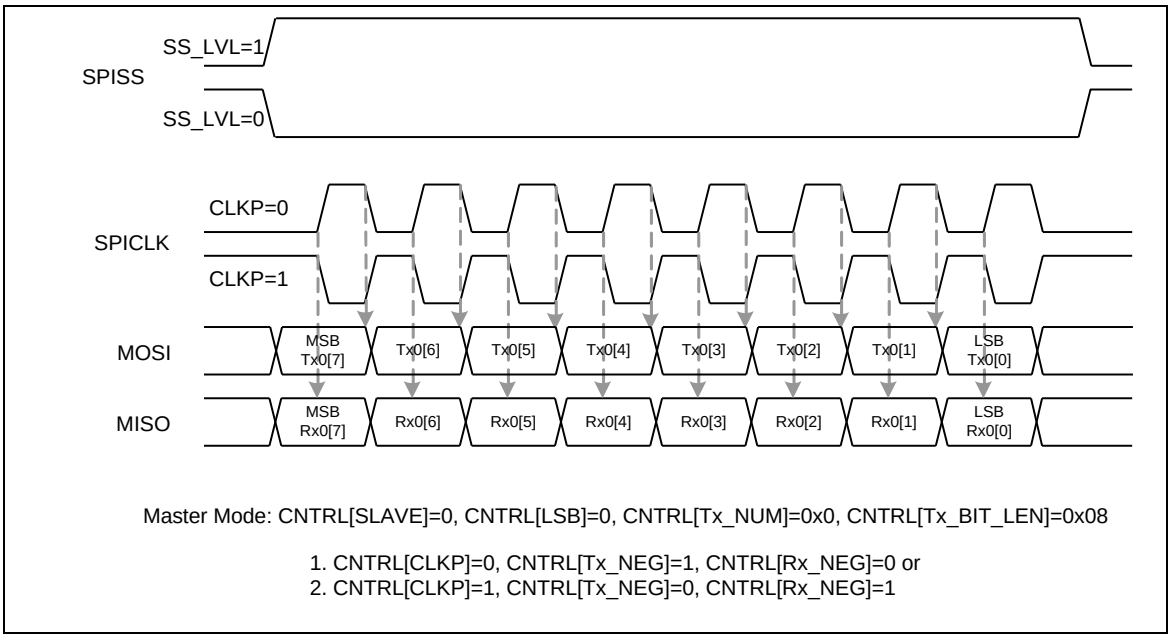


图 5.11-9 SPI 主模式时序

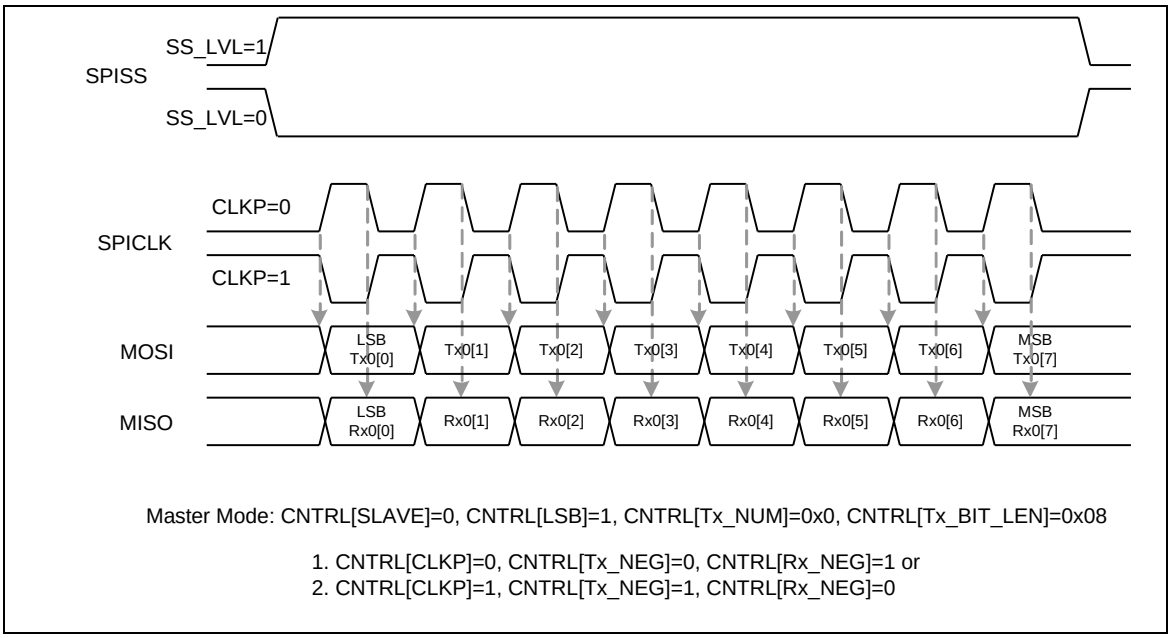


图 5.11-10 SPI 主模式时序(SPICLK 相位可选)

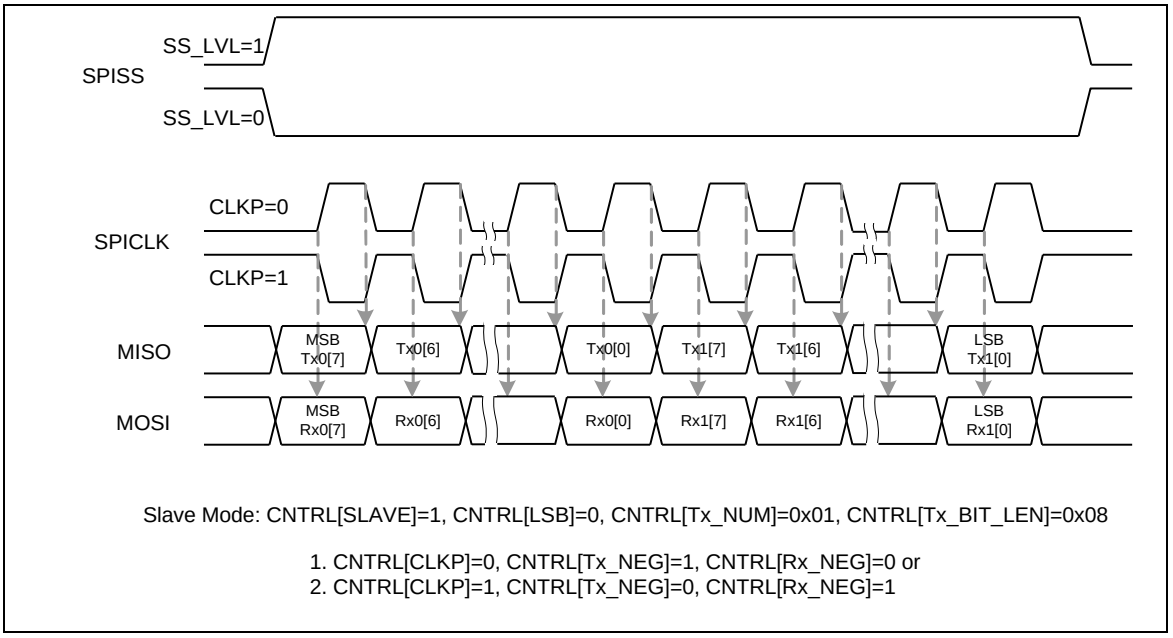


图 5.11-11 SPI 从模式时序

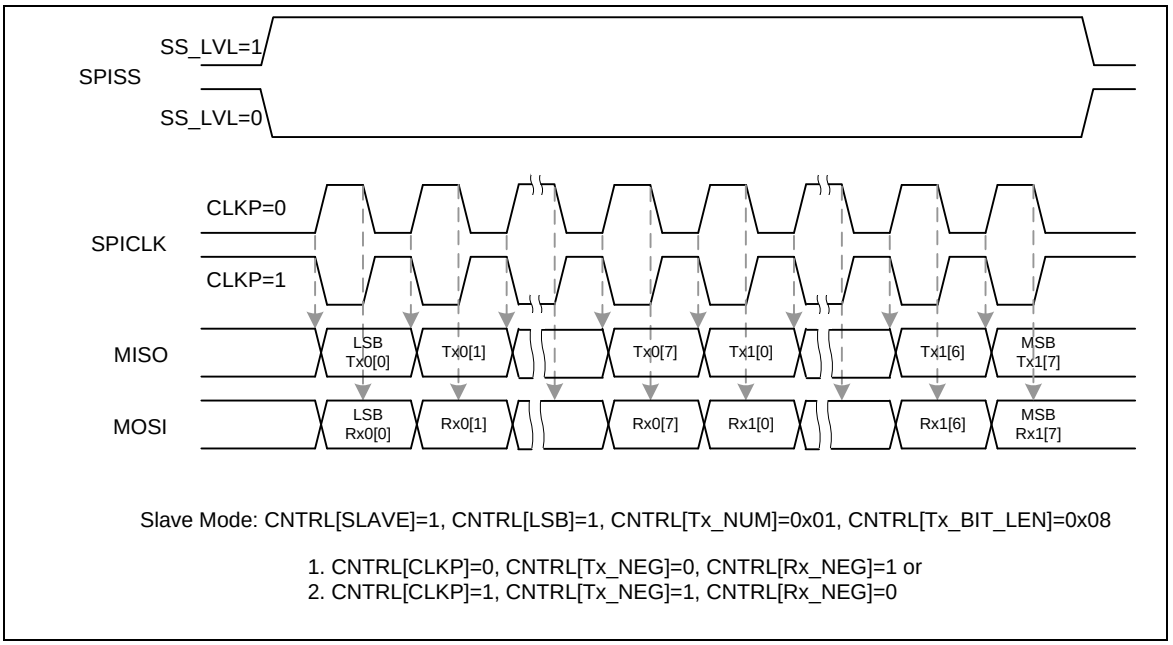


图 5.11-12 SPI 从模式时序(SPICLK 相位可选)

5.11.4.17 SPI编程实例

实例1, SPI控制器作为主设备,访问片外从设备,从设备有下列规格:

- 上升沿接收

- 下降沿发送
- MSB优先
- SPICLK空闲时为低电平
- 收/发长度为一个字节
- 片选信号低电平有效

基本上,下列步骤应该参考从设备的规格:

1) 写DIVIDER (SPI_DIVIDER[15:0]) 寄存器决定串行时钟的输出频率.
2) 写一个合适的值到 SPI_SSR 寄存器,设定主模式的相关设定.
    1. 使能自动片选ASS (SPI_SSR[3] = 1).
    2. 选择片选信号低电平有效SS_LVL (SPI_SSR[2] = 0).
    3. 设定片选寄存器比特SSR (SPI_SSR[0]),选择片选信号输出的IO脚.
3) 写控制寄存器SPI_CNTRL 来控制 SPI 主设备的行为.
    1. 设定SPI控制器做主设备:SLAVE 比特 (SPI_CNTRL[18] = 0).
    2. 迫使串行时钟输出低电平:CLKP 比特 (SPI_CNTRL[11] = 0).
    3. 下降沿发送数据:Tx_NEG 比特 (SPI_CNTRL[2] = 1).
    4. 上升沿接收:Rx_NEG 比特 (SPI_CNTRL[1] = 0).
    5. 设定每次传输的比特长度为8比特: Tx_BIT_LEN (SPI_CNTRL[7:3] = 0x08).
    6. 每次收/发只收/发一笔数据:Tx_NUM (SPI_CNTRL[9:8] = 0x0).
    7. 设定MSB 优先:LSB 比特 (SPI_CNTRL[10] = 0), 由于不是突发模式,不关心 SP_CYCLE (SPI_CNTRL[15:12]) 的设定.
4) 如果SPI主设备要发送数据到片外从设备,写一个字节到Tx0 (SPI_Tx0[7:0]) 寄存器.
5) 如果SPI主设备要从片外从设备接收数据, 则不关心发送的数据.写“FFH”到Tx0 (SPI_Tx0[7:0]) 寄存器就好.
6) 使能GO_BUSY 比特 (SPI_CNTRL[0] = 1) 来启动数据传输.
    - ◆ 等待SPI 中断(如果中断使能比特IE 被设置) 或者轮询GO_BUSY 比特直到变成“0”.
7) 从Rx0 (SPI_Rx0[7:0]) 寄存器读回收到的数据.
8) 返回 4) 继续下一个数据的传输或者设定SSR 成 “0” 取消片选引脚的选择.

实例 2, SPI 控制器作为从设备由片外主设备控制, 片外SPI 主设备具有下列特性:

- 上升沿接收
- 下降沿发送
- LSB 优先

- SPICLK 空闲时输出高电平
- 一次收/发一个字节
- 片选信号高电平有效

基本上, 下列步骤要参考片外主设备的特性,

1) 写一个合适的值到SPI_SSR 寄存器设定相关从设备的特性.

   通过设定片选电平比特 SS_LVL (SPI_SSR[2] = 1)和电平触发比特SS_LTRIG (SPI_SSR[4] = 1),选择片选高电平有效并且电平触发.

2) 写相关设定到SPI_CNTRL 寄存器控制SPI 从设备的行为.

   1. 设定SPI控制器作为从设备:SLAVE 比特 (SPI_CNTRL[18] = 1).
   2. 时钟空闲时高电平: CLKP比特(SPI_CNTRL[11] = 1).
   3. 下降沿发送: Tx_NEG比特(SPI_CNTRL[2] = 1).
   4. 上升沿接收:Rx_NEG比特(SPI_CNTRL[1] = 0).
   5. 一次传输比特长度为8比特:Tx_BIT_LEN (SPI_CNTRL[7:3] = 0x08).
   6. 每次传输一笔数据:Tx_NUM (SPI_CNTRL[9:8] = 0x0).
   7. LSB 优先:LSB比特(SPI_CNTRL[10] = 1), 由于不是突发模式不关心SP_CYCLE (SPI_CNTRL[15:12]) 的设定.

3) 如果SPI从设备要发送一个(被读) 字节到片外主设备, 写要发送的一个字节到Tx0 (SPI_Tx0[7:0]) 寄存器.

4) 如果SPI从设备要从片外主设备接收一个(被写) 字节, 则不关心要发送的数据，写“FFH”到SPI_Tx0[7:0] 寄存器就好.

5) 使能GO_BUSY 比特 (SPI_CNTRL[0] = 1) 等待片选和串行时钟从片外主设备输入.

   等待SPI 中断发生(如果中断使能比特IE 被设) 或者轮询GO_BUSY 比特直到被硬件清0.--

6) 从接收寄存器Rx (SPI_Rx0[7:0]) 寄存器读回收到的数据.

7) 返回 3) 继续下一个数据传输或者关闭GO_BUSY比特结束传输.

5.11.5 SPI 串行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	R/W	描述	复位值
SPI_BA = 0x4003_4000				
SPI_CNTRL	SPI_BA+0x00	R/W	控制和状态寄存器	0x0000_0004
SPI_DIVIDER	SPI_BA+0x04	R/W	时钟除频寄存器	0x0000_0000
SPI_SSR	SPI_BA+0x08	R/W	从设备选择 寄存器	0x0000_0000
SPI_RX0	SPI_BA+0x10	R	数据接收寄存器0	0x0000_0000
SPI_RX1	SPI_BA+0x14	R	数据接收寄存器1	0x0000_0000
SPI_TX0	SPI_BA+0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1	SPI_BA+0x24	W	数据发送寄存器1	0x0000_0000
SPI_VARCLK	SPI_BA+0x34	R/W	可变时钟模型寄存器	0x007F_FF87
SPI_CNTRL2	SPI_BA+0x3C	R/W	控制和状态寄存器2	0x0000_0000

注意 1:当软件编程SPI_CNTRL时, GO_BUSY 比特应该最后再写.

5.11.6 寄存器描述

SPI 控制和状态寄存器(SPI_CNTRL)

寄存器	偏移	R/W	描述	复位值
SPI_CNTRL	SPI_BA+0x00	R/W	控制和状态寄存器	0x0000_0004

31	30	29	28	27	26	25	24
-							
23	22	21	20	19	18	17	16
VARCLK_EN	-		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]	-	**预留**
[23]	VARCLK_EN	**可变时钟使能 (只用于主模式)** 0 = 串行时钟输出频率是固定的,只由**DIVIDER**的值决定. 1 = 串行时钟输出频率是可变的. 输出频率由**VARCLK**, **DIVIDER**, 和 **DIVIDER2**的值共同决定. **注意**:当使能**VARCLK_EN** 比特时, **Tx_BIT_LEN** 必须设为0x10 (16比特模式).
[22:21]	-	**预留**
[20:19]	REORDER[1:0]	**重新排序模式选择** REORDER: 描述 00: 禁止字节重新排序和字节 暂停功能 01: 使能字节重新排序和每个字节之间插入字节暂停间隔 (2~17 SPICLK 周期). Tx_BIT_LEN 必须设为 “00H” (32-bit/word). 10: 使能字节重新排序, 但是禁止字节暂停功能. 11: 禁止字节重新排序但是使能每个字节之间插入字节暂停间隔 (2~17 SPICLK 周期).
[18]	SLAVE	**从设备模式** 0 = 主设备. 1 = 从设备.

Bits	描述	
[17]	**IE**	**中断使能** 0 = 禁止 SPI 中断. 1 =使能 SPI 中断.
[16]	**IF**	**中断标志** 0 = 指示传输还没有完成. 1 = 传输已经完成. 如果中断使能这个比特将被设. **注意: 这个比特写"1"清除**.
[15:12]	**SP_CYCLE[3:0]**	**暂停间隔设定 (只用于主设备)** 这4个比特配置一次传输的两笔之间的暂停间隔. 如果CLKP = 0,暂停间隔是从上次传输最后一个时钟下降沿到下次传输第一个时钟上升沿. 如果CLKP = 1,暂停间隔是从上次传输最后一个时钟下降沿到下次传输第一个时钟下降沿. SP_CYCLE 的缺省值是 "0" (2个串行时钟周期), 但是如果Tx_NUM = "00b",设定这些比特没有作用. 暂停间隔公式如下: (SP_CYCLE[3:0] + 2) * SPICLK 时钟周期

SP_CYCLE 值	suspend 间隔
0000	2 SPICLK 时钟周期
0001	3 SPICLK 时钟周期
------	----------
1110	16 SPICLK 时钟周期
1111	17 SPICLK 时钟周期

[11]	**CLKP**	**时钟极性** 0 = SPICLK空闲时低电平. 1 = SPICLK空闲时高电平.
[10]	**LSB**	**LSB 优先** 0 = MSB 优先收/发 (SPI_Tx0/1 和 SPI_Rx0/1 寄存器中收发的比特数 由t Tx_BIT_LEN 定义). 1 = LSB 优先发送 (SPI_Tx0/1的比特0), 收到的第一个比特也会放在Rx 寄存器 (SPI_Rx0/1的比特0)LSB 的位置.
[9:8]	**TX_NUM[1:0]**	**收/发笔数** 定义一次收/发多少笔数据.

TX_NUM	描述
00	一次收发一笔数据.
01	一次收发两笔数据(触发模式).
10	预留
11	预留

<table>
<tr><th>Bits</th><th>描述</th><th></th></tr>
<tr><td></td><td></td><td>注意: 从模式下电平触发时,连续两次收发期间,片选引脚必须维持激活状态.</td></tr>
<tr><td>[7:3]</td><td>TX_BIT_LEN [4:0]</td><td>传输比特长度
定义一次传输收/发多少比特. 最大32比特.
<table>
<tr><th>TX_BIT_LEN</th><th>描述</th></tr>
<tr><td>00001</td><td>一次传输1 bit</td></tr>
<tr><td>00010</td><td>一次传输2 bit</td></tr>
<tr><td>00011</td><td>一次传输3 bits</td></tr>
<tr><td>------</td><td>----------</td></tr>
<tr><td>11111</td><td>一次传输31 bits</td></tr>
<tr><td>00000</td><td>一次传输32</td></tr>
</table></td></tr>
<tr><td>[2]</td><td>TX_NEG</td><td>下降沿发送
0 = 发送的数据在SPICLK时钟上升沿改变.
1 =发送的数据在SPICLK时钟下降沿改变.</td></tr>
<tr><td>[1]</td><td>RX_NEG</td><td>下降沿接收
0 =接收的数据在SPICLK时钟上升沿锁存.
1 =接收的数据在SPICLK时钟下降沿锁存.</td></tr>
<tr><td>[0]</td><td>GO_BUSY</td><td>Go and Busy Status
0 =如果SPI正在传输, 写“0”到这个比特将停止数据传输.
1= 主模式下, 写“1” 到这个比特启动一次SPI数据传输;从模式下,写“1” 到这个比特指示从设备已经准备好和主设备通讯. 数据传输期间, 保持这个比特为 “1”. 数据传输完成以后, 这个比特将被自动清除.
注意:写”1”到GO_BUSY之前,所有寄存器应该先设定好.</td></tr>
</table>

SPI 除频寄存器(SPI_DIVIDER)

寄存器	偏移	R/W	描述	复位值
SPI_DIVIDER	SPI_BA+0x04	R/W	时钟除频寄存器(只用于主模式)	0x0000_0000

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

Bits	描述	
[31:16]	DIVIDER2[15:0]	**时钟除频器2 寄存器(只用于主模式)** 这个域的值是系统时钟, PCLK,的2nd 频率除频器, 用来产生SPICLK的串行输出时钟. 输出频率的公式如下: $f_{sclk} = \frac{f_{psclk}}{(DIVIDER2+1)*2}$
[15:0]	DIVIDER[15:0]	**时钟除频器寄存器(只用于主模式)** 这个域的值是系统时钟, PCLK,的频率除频器, 用来产生SPICLK的串行输出时钟. 输出频率的公式如下: $f_{sclk} = \frac{f_{psclk}}{(DIVIDER+1)*2}$ 从模式下, 由主设备驱动的SPI时钟周期应该大于或者等于PCLK周期的5倍.换句话说,就是输入的SPI时钟频率最大不能超过从设备PCLK频率的1/5..

SPI从设备选择寄存器(SPI_SSR)

寄存器	偏移	R/W	描述	复位值
SPI_SSR	SPI_BA+0x08	R/W	从设备选择寄存器	0x0000_0000

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

Bits	描述	
[31:6]	-	**预留**
[5]	LTRIG_FLAG	**电平触发标志** 从模式下,当SS_LTRIG 比特被设时,这个比特可以被读用来指示接收到的比特数和传输笔数是否与期望一致. 1 = 传输笔数和传输的比特数与期望一致(由Tx_NUM和Tx_BIT_LEN定义). 0 =传输笔数或者传输的比特数与期望不符. **注意:** 这个比特是只读的.
[4]	SS_LTRIG	**片选触发电平(只用于从模式)** 0 = 输入的片选信号是边沿触发的. 1 =输入的片选信号是电平触发的.依靠SS_LVL 的设定决定信号是低电平还是高电平.
[3]	ASS	**自动片选 (只用于主模式)** 0 = 如果这个比特被清除, 片选信号通过设定和清除SSR寄存器的相关比特来发出和取消. 1 = 如果这个比特被设置, 片选信号由SPI控制器自动产生.这意味着设定GO_BUSY启动收/发时,SPI控制器将根据SSR寄存器的设定发出/取消片选信号.
[2]	SS_LVL	**片选信号激活电平** 定义片选信号(SPISS)的激活电平. 0 = 片选信号低电平/下降沿激活. 1 =片选信号高电平/上升沿激活.
[1]	-	**预留**
[0]	SSR	**片选寄存器(只用于主模式)**

Bits	描述	
		如果ASS 比特被清除, 写“1” 到这个域将激活SPISS片选信号, 写“0” 到这个域将取消片选. 如果ASS 比特被设定, 写“1” 到这个域将导致收/发时SPI控制器自动激活片选SPISS, 其它时间自动取消片选. (SPISS 的激活电平由SS_LVL 决定). **注意**: SPISS定义为从模式下从设备片选输入信号.

SPI 数据接收寄存器(SPI_RX)

寄存器	偏移	R/W	描述	复位值
SPI_RX0	SPI_BA+0x10	R	数据接收寄存器0	0x0000_0000
SPI_RX1	SPI_BA+0x14	R	数据接收寄存器1	0x0000_0000

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

Bits	描述	
[31:0]	**RX[31:0]**	**数据接收寄存器** 数据接收寄存器用来保存最后一次传输收到的数据. 有效位数依靠SPI_CNTRL 寄存器比特长度域的定义. 例如, 如果Tx_BIT_LEN 设成“08H” 并且 Tx_NUM 设成 “0”, Rx0[7:0] 保存的是收到的数据. **注意:** 数据接收寄存器是只读的.

SPI 数据发送寄存器(SPI_TX)

寄存器	偏移	R/W	描述	复位值
SPI_TX0	SPI_BA+0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1	SPI_BA+0x24	W	数据发送寄存器1	0x0000_0000

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

Bits	描述	
[31:0]	**TX[31:0]**	**数据发送寄存器** 数据发送寄存器用来保存要发送的数据.有效位数依靠CNTRL 寄存器中传输比特长度的定义. 例如, 如果Tx_BIT_LEN 被设为“08H” ,Tx_NUM 被设为“0”, Tx0[7:0] 将保存要发送的数据.如果Tx_BIT_LEN被设为“00H” ,Tx_NUM被设为“1”, Tx0[31:0]和Tx1[31:0]将保存要发送的数据 (顺序为Tx0[31:0], Tx1[31:0]).

SPI 可变时钟模型寄存器(SPI_VARCLK)

寄存器	偏移	R/W	描述	复位值
SPI_VARCLK	SPI_BA+0x34	R/W	可变时钟模型寄存器	0x007F_FF87

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

Bits	描述	
[31:0]	VARCLK[31:0]	**可变时钟模型** 这个域的值是SPI时钟的频率模型. 如果VARCLK某个比特的值是“0”, SPICLK根据DIVIDER的值输出频率. 如果VARCLK某个比特的值是“1”, SPICLK根据DIVIDER2的值输出频率. 请参考寄存器SPI_DIVIDER. 参考图 5.11-8 **注意**: 只可用于CLKP = 0 的情况.

SPI 控制与状态寄存器2 (SPI_CNTRL2)

这个寄存器比特[9:11]用于3线从设备模式，3线SPI从设备模式就是没有从设备片选信号

寄存器	偏移	R/W	描述	复位值
SPI_CNTRL	SPI_BA+0x3C	R/W	控制和状态寄存器 2	0x0000_0000

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

Bits	描述	
[31:12]	-	**预留**
[11]	SLV_START_INTSTS	**从设备开始传输中断状态** 用来指示无片选从模式传输已经开始. 1 =指示无片选从模式传输已经开始.传输结束将自动清0或者写”1”清除. 0 =指示无片选从模式传输还没有开始.
[10]	SSTA_INTEN	**从设备开始传输中断使能** 用来使能无片选从模式传输已经开始中断. 如果传输开始以后超过用户定义的时间传输还没有完成,用户可以设置SLV_ABORT 比特来迫使传输完成. 1 = 使能传输开始中断.. 0 =禁止传输开始中断.
[9]	SLV_ABORT	**终止无片选从模式** 正常情况下, 当接收到的数据满足期望的比特数和笔数时 ,有中断发生. 如果收到的比特数小于期望值并且超过一次传输时间不再有串行时钟输入时,用户可以设置这个比特来迫使完成当前传输然后会得到一个传输完成事件. **注意:** 当终止事件激活时这个比特会自动清“0”.
[8]	NOSLVSEL	**无片选从模式** 从模式时用来忽略片选.SPI控制器可以工作在3线模式，包括SPICLK, SPI_MISO, 和SPI_MOSI信号. 0 = 控制器为4-wire 双向接口. 1 = 从模式下控制器为3线双向接口. 当这个比特为“1”时, GO_BUSY比特激活并且串行时钟输入时，控制器就开始收/发数据. **注意:** 无片选信号时, SS_LTRIG, SPI_SSR[4], 应该设成 “1”
[7:1]	-	**预留**
[0]	DIV_ONE	**时钟除频固定除以1**

Bits	描述	
		这个寄存器用来设定系统时钟,PCLK,频率除以1功能,也就是SPICLK时钟频率将等于PCLK. 0 = SPICLK输出频率由DIVIDER 和 DIVIDER2 决定. 1 = 除频值固定为 “1”..

5.12 定时器控制器

5.12.1 概述

定时器模块包含两个通道, TIMER0~TIMER1,用户可以很容易的实现定时器控制. 定时器可以实现频率测量,间隔测量,时钟产生,时间延迟等功能.超时发生时定时器可以产生中断或者计时过程中返回当前值.

5.12.2 特性

- 2组32比特定时器，有24比特上数定时器和一个8比特预分频计数器
- 两个通道时钟源独立 (TMR0_CLK, TMR1_CLK)
- 支持one-shot, 周期, toggle 和 连续计数操作模式
- 超时周期= (定时器输入时钟频率) * (8比特预分频+ 1) * (24-bit TCMP)
- 最大计数周期= (1 / T MHz) * (2^8) * (2^{24}), T为时钟源周期
- 内部24比特上数计数值可以通过TDR (定时器数据寄存器)读到
- 支持数事件功能用来数外部输入引脚的事件个数
- 支持输入捕捉功能用来捕捉或者复位计数器的值

5.12.3 方块图

每个通道配备一个8比特预分频计数器，一个24比特上数计数器，一个24比特比较寄存器和一个中断请求信号. 参考图 5.12-1. 定时器控制器方块图. 每个通道的时钟源有5种选择.. 图 5.12-2说明了时钟源控制功能.

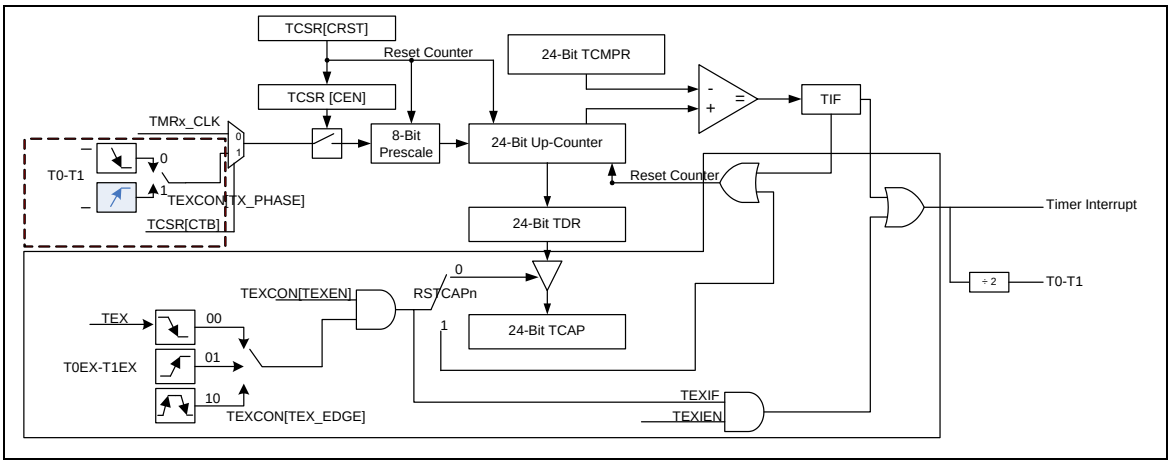


图 5.12-1 定时器控制器方块图

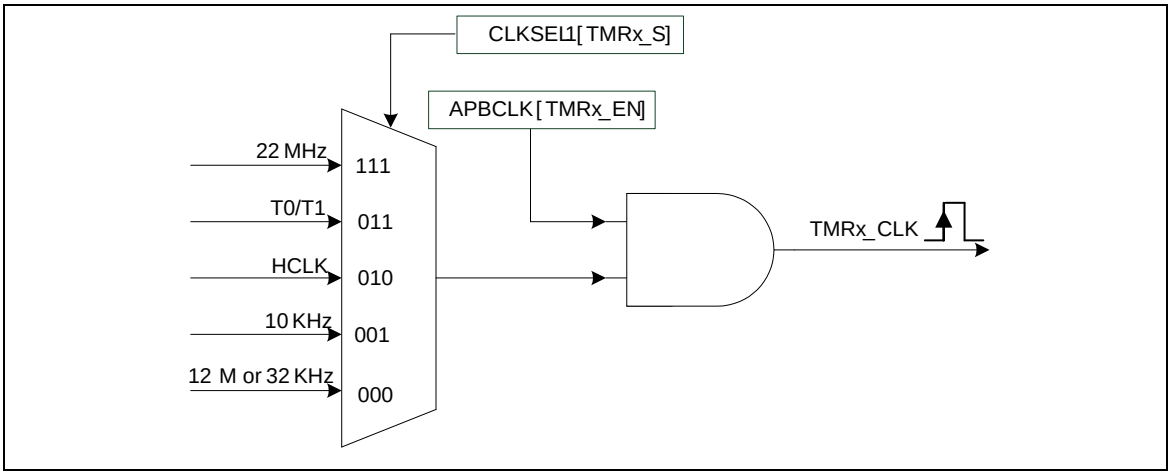


图 5.12-2 定时器控制器时钟源

5.12.4 功能描述

定时器控制器提供one-shot, 周期, toggle 和连续计数模式. 也提供事件计数功能来计数外部引脚上的事件和输入捕捉功能来捕捉或者复位定时器的计数值. 每种操作模式如下所述:

5.12.4.1 One –Shot 模式

如果定时器操作在one-shot 模式并且CEN (TCSR[30] timer enable bit) 被设为 “1”, 定时器计数器开始上数. 一旦定时器计数值达到比较寄存器 (TCMPR) 的值, 如果IE (TCSR[29] interrupt enable bit) 被设为 “1”, 定时器中断标志将被设并且中断将发生. 如果IE 被设为“0”, 没有中断发生. 这个操作模式下,一旦定时器计数值达到比较寄存器 (TCMPR) 的值, 定时器计数值将返回初始值,CEN被定时器控制器清成“0”. 一旦定时器计数值达到比较寄存器 (TCMPR) 的值,计数将停止. 也就是说, 编程定时器比较寄存器(TCMPR)并使能CEN之后,定时器计数并达到比较寄存器TCMPR的值只一次. 所以,这个操作模式称为 One-Shot 模式.

5.12.4.2 周期模式

如果定时器操作在周期模式并且CEN 被设成“1”, 定时器计数器开始计数. 一旦定时器计数值达到比较寄存器 (TCMPR) 的值, 如果IE (TCSR[29] interrupt enable bit) 被设为 “1”, 定时器中断标志将被设并且中断将发生. 如果IE 被设为“0”, 没有中断发生. 这个操作模式下,一旦定时器计数值达到比较寄存器 (TCMPR) 的值, 定时器计数值将返回初始值,CEN维持“1” (counting enable continuously)不变. 定时器计数器再次上数. 如果中断标志由软件清除, 一旦定时器计数值达到比较寄存器 (TCMPR) 的值并且 IE 被设为 “1”, 中断标志被设定时器将再次发生中断. 也就是说, 定时器计数并达到比较寄存器TCMPR的值周期性发生. 直到CEN被设为”0”定时器才会停止计数. 中断也会周期性产生.所以这种模式称为周期模式.

5.12.4.3 Toggle 模式

如果定时器操作在Toggle模式并且CEN 被设成“1”, 定时器计数器开始计数. 一旦定时器计数值达到比较寄存器 (TCMPR) 的值, 如果IE (TCSR[29] 中断使能位) 被设为 “1”, 定时器中断标志将被设并且中断将发生. 相应的切换输出 (tout) 信号被设成 “1”. 这个操作模式下,一旦定时器计数值达到比较寄存器 (TCMPR) 的值, 定时器计数值将返回初始值,CEN维持“1” (counting enable continuously)不变. 定时器计数器再次上数. 如果中断标志由软件清除, 一旦定时器计数值达到比较寄存器 (TCMPR) 的值并且 IE 被设为 “1”, 中断标志被设定时器将再次发生中断. 相应的切换输出 (tout) 信号被设成“0”. 直到CEN被设为”0”定时器才会停止计数. 因而, 切换输出(tout) 信号来回改变,占空比50%.所以这种模式称为Toggle模式.

5.12.4.4 连续计数模式

如果定时器操作在连续计数模式并且CEN 被设成“1”, 定时器计数器开始计数, 一旦定时器计数值达到比较寄存器 (TCMPR) 的值, 如果IE (TCSR[29] 中断使能位) 被设为 “1”, 定时器中断标志将被设并且中断将发生. 用户可以立即改变TCMPR的值不用关闭定时器计数器再重新计数. 例如,首先TCMPR 被设成80. (TCMPR 应该小于2^{24} -1 大于 1). 当TDR的值等于80的时候, 如果IE使能,定时器中断标志TIF将被设并且中断将发生. 但是CEN维持“1” (counting enable continuously) 不变并且TDR的值也不会变回0, 而是继续数81, 82, 83, • • • 直到 2^{24} -1, 0, 1, 2, 3, • • • 直到 2^{24} -1 一次又一次. 接下来, 如果用户编程TCMPR 为 200 ,TIF被清成 “0”, 当TDR的值等于200的时候, 如果IE使能,定时器中断标志TIF将被设并且中断将再次发生. 最后, 用户又编程TCMPR 等于 500 并再次清除TIF 成“0”, 当TDR的值等于500的时候, 如果IE使能,定时器中断标志TIF将被设并且中断将再次发生. 从应用的角度来看, 中断依靠TCMPR的值产生TCMPR. 这种模式下, 定时器连续计数. 所以这种模式称为连续计数模式.

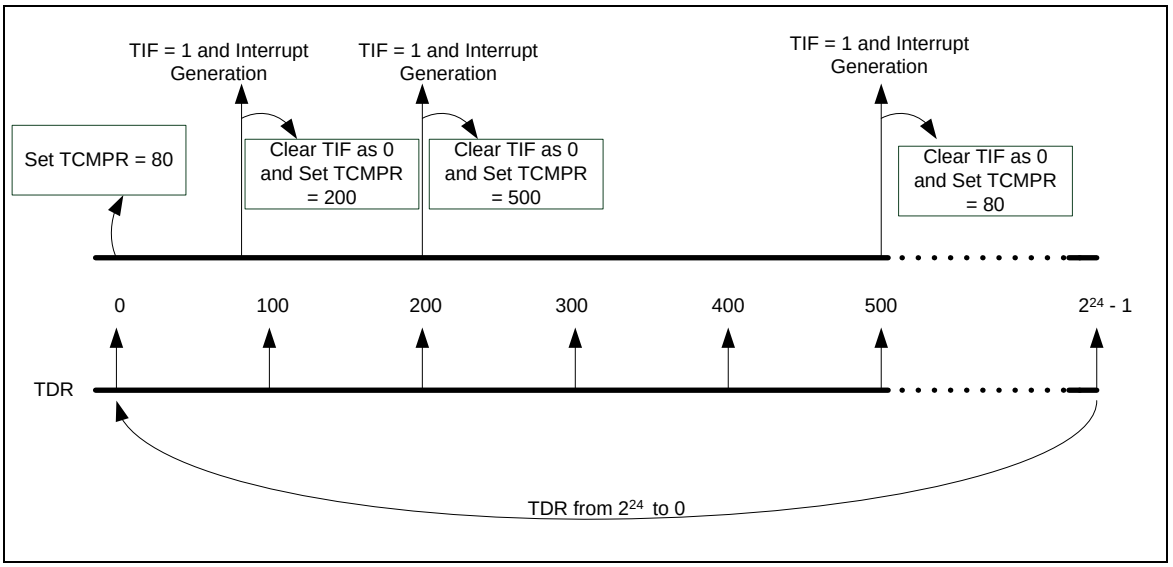


图 5.12-3 连续计数模式

5.12.4.5 事件计数功能

提供一个可以计数来自T0~T1引脚上的事件的应用.称为事件计数模式.事件计数模式下,定时器的时钟源, TMRx_CLK, 在 图 5.12-2中应该被设为HCLK .并且如果关闭计数de-bounce功能,输入事件计数源的频率应该小于HCLK 频率的1/3;如果打开de-bounce 功能的话应该小于HCLK的1/8.否则得到的TDR的值是不正确的. 通过TEXCON[7]可以使能或者关闭de-bounce功能, 通过TEXCON[0]可以设定T0~T1上升沿或者下降沿计数.

5.12.4.6 输入捕捉功能

提供输入捕获或者复位功能来捕捉或者复位定时器的计数值. 通过CAP_MODE (TEXCON[8])可以配置捕获功能为free-counting 捕获模式和trigger-counting 捕获模式. free-counting 捕获模式, 复位模式, trigger-counting 捕获模式描述如下:

5.12.4.7 Free-Counting捕捉模式

如果CAP_MODE被清成“0”, TEXEN (TEXCON[3]) 被设为“1” 并且 RSTCAPn 被设成 “0”, 当TEX (Timer External Pin) 引脚触发条件发生时,TDR 将被捕捉存入TCAP 寄存器. TEX 触发边沿由TEX_EDGE来选择.

5.12.4.8 复位模式

如果CAP_MODE被清成“0”, TEXEN (TEXCON[3]) 被设为“1” 并且RSTCAPn被设成“1”, 当TEX (Timer External Pin) 引脚触发条件发生时,TDR 将被复位成”0”. TEX 触发边沿由TEX_EDGE来选择.

5.12.4.9 Trigger-Counting 捕获模式

如果CAP_MODE被设为“1”, TEXEN (TEXCON[3]) 被设为“1” 并且RSTCAPn 被清成 “0”, TDR被复位成

0，当TEX (Timer External Pin) 引脚触发条件发生时,TDR开始计数; 当TEX (Timer External Pin) 引脚触发条件再次发生时TDR的值将被捕捉存入TCAP寄存器. TEX 触发边沿由TEX_EDGE来选择.

当 TEX 触发发生时, TEXIF (定时器外部中断标志) 被设为“1”, 如果TEXIEN (Timer External Interrupt Enable Bit) 为 “1”,中断将发生. 如果关闭计数de-bounce功能,输入事件的频率应该小于HCLK 频率的1/3; 如果打开计数de-bounce功能,输入事件的频率应该小于HCLK 频率的1/8. 由 TEXCON[7] 控制是否打开 TM0~TM1 de-bounce 功能.

表 5.12-1 输入捕捉模式操作

功能	CAP_MODE (TEXCON[8])	RSTCAPN (TEXCON[4])	TEX_EDGE (TEXCON[2:1])	操作描述
Free-Counting捕捉模式	0	0	00	定时器外部输入引脚1到0的转变被检测. TDR 被捕捉存入TCAP.
	0	0	01	定时器外部输入引脚0到1的转变被检测. TDR 被捕捉存入TCAP.
	0	0	10	定时器外部输入引脚0到1,1到0的转变都会被检测. TDR 被捕捉存入TCAP.
	0	0	11	预留
复位模式	0	1	00	定时器外部输入引脚1到0的转变被检测. TDR 被复位成0.
	0	1	01	定时器外部输入引脚0到1的转变被检测. TDR 被复位成0.
	0	1	10	定时器外部输入引脚0到1,1到0的转变都会被检测. TDR 被复位成0.
	0	1	11	预留
Trigger-Counting捕捉模式	1	0	00	Falling edge trigger: 定时器外部输入引脚第一次1到0转变将TDR复位成”0”并开始计数,第二次1到0转变停止计数.
	1	0	01	Rising edge trigger: 定时器外部输入引脚第一次0到1转变将TDR复位成”0”并开始计数,第二次0到1转变停止计数.
	1	0	10	Level change trigger: 定时器外部输入引脚第一次1到0转变将TDR复位成”0”并开始计数,第二次0到1转变停止计数.
	1	0	11	Level change trigger: 定时器外部输入引脚第一次0到1转变将

功能	CAP_MODE (TEXCON[8])	RSTCAPN (TEXCON[4])	TEX_EDGE (TEXCON[2:1])	操作描述
				TDR复位成”0”并开始计数,第二次1到0转变停止计数.

5.12.5 寄存器映射

R: 只读, **W:** 只写, **R/W:** 可读写

寄存器	偏移	R/W	描述	复位值
TMR_BA = 0x4001_0000				
TCSR0	TMR_BA+0x00	R/W	定时器0控制和状态寄存器	0x0000_0005
TCMPR0	TMR_BA+0x04	R/W	定时器0比较寄存器	0x0000_0000
TISR0	TMR_BA+0x08	R/W	定时器0 中断状态寄存器	0x0000_0000
TDR0	TMR_BA+0x0C	R	定时器0 数据寄存器	0x0000_0000
TCAP0	TMR_BA+0x10	R	定时器0 捕捉数据寄存器	0x0000_0000
TEXCON0	TMR_BA+0x14	R/W	定时器0外部控制寄存器	0x0000_0000
TEXISR0	TMR_BA+0x18	R/W	定时器0外部中断状态寄存器	0x0000_0000
TCSR1	TMR_BA+0x20	R/W	定时器1控制和状态寄存器	0x0000_0005
TCMPR1	TMR_BA+0x24	R/W	定时器1比较寄存器	0x0000_0000
TISR1	TMR_BA+0x28	R/W	定时器1 中断状态寄存器	0x0000_0000
TDR1	TMR_BA+0x2C	R	定时器1 数据寄存器	0x0000_0000
TCAP1	TMR_BA+0x30	R	定时器1 捕捉数据寄存器	0x0000_0000
TEXCON1	TMR_BA+0x34	R/W	定时器1外部控制寄存器	0x0000_0000
TEXISR1	TMR_BA+0x38	R/W	定时器1部中断状态寄存器	0x0000_0000

5.12.6 寄存器描述

定时器控制寄存器(TCSR)

寄存器	偏移	R/W	描述	复位值
TCSR0	TMR_BA+0x00	R/W	定时器0控制和状态寄存器	0x0000_0005
TCSR1	TMR_BA+0x20	R/W	定时器1控制和状态寄存器	0x0000_0005

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

Bits	描述	
[31]	DBGACK_TMR	**禁止ICE调试模式应答(写保护)** 0 = ICE 调试模式应答影响定时器计数. ICE调试模式下,定时器停止计数. 1 = 关闭ICE 调试模式应答. 无论ICE是否在调试模式下, 定时器继续计数..
[30]	CEN	**定时器使能比特** 1 = 开始计数. 0 = 停止/暂停计数. 注意1: 停止状态下, 设置CEN 为“1” 将使能24比特上数计数器,从上次的计数值继续计数. 注意2: one-shot (MODE [28:27] =00)模式下,当中断(IE[29] =1).发生时,这个比特由硬件自动清0.
[29]	IE	**中断使能比特** 1 = 使能定时器中断. 0 = 禁止定时器中断. 如果定时器中断被使能, 当上数计数器的值等于TCMPR时,定时器发出中断.
[28:27]	MODE[1:0]	**定时器操作模式**

Bits	描述	
		MODE: 定时器操作模式 00: 定时器操作在one-shot 模式. 相应的中断发生一次 (如果IE 使能)， CEN 由硬件自动清0. 01: 定时器操作在周期模式. 相应的中断周期性发生(如果IE 使能). 10: 定时器操作在 toggle模式. 中断信号周期性发生(如果IE 使能).相应的输出信号 (tout)来回切换 ,占空比50%. 11: 定时器操作在连续计数模式. 当TDR = TCMPR 时(如果IE 使能)中断将发生. 然而,24比特计数器继续计数. 细节请参考5.12.4.4节连续计数模式的描述.
[26]	**CRST**	**定时器复位比特** 设置这个比特将导致24位上数计数器,8比特预分频器复位也会导致CEN 等于 “0”. 0 = 没有作用. 1 = 复位定时器的 8比特预分频器,内部24比特上数计数器 和CEN 比特.
[25]	**CACT**	**定时器激活状态比特 (只读)** 这个比特指示上数计数器的状态. 0 = 定时器没有激活. 1 = 定时器是活动的.
[24]	**CTB**	**计数模式使能比特** 这个比特用来使能计数模式. 当定时器用作事件计数模式时,这个比特应该被设为”1”, 定时器将工作在事件计数模式. 计数器探测外部引脚的相位可以由TX_PHASE域选为上升/下降沿. 1 = 使能计数模式. 0 = 禁止计数模式.
[23]	**WAKE_EN**	**唤醒使能** 当WAKE_EN被设时,如果TIF 或者 TEXIF 被设, 定时器控制器将产生一个唤醒事件唤醒CPU. 0 = 禁止唤醒触发事件. 1 = 使能唤醒触发事件.
[22:17]	**-**	**预留**
[16]	**TDR_EN**	**数据加载使能** 当TDR_EN 被设时, 定时器计数时,24比特上数计数器的值将被连续更新到TDR (定时器数据寄存器). 1 = 定时器数据寄存器更新使能. 0 = 定时器数据寄存器更新禁止.
[15:8]	**-**	**预留**
[7:0]	**PRESCALE**	**Pre-Scale Counter** 时钟在输入给定时器之前可以除以PRESCALE+1来预分频。如果PRESCALE = “0”, 就

Bits	描述	
	[7:0]	不分频.

定时器比较寄存器(TCMPR)

寄存器	偏移	R/W	描述	复位值
TCMPR0	TMR_BA+0x04	R/W	定时器0比较寄存器	0x0000_0000
TCMPR1	TMR_BA+0x24	R/W	定时器1比较寄存器	0x0000_0000

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

Bits	描述	
[31:24]	**-**	**预留**
[23:0]	**TCMP[23:0]**	**定时器比较值** TCMP 是一个24比特比较值寄存器. 当内部24比特上数计数器的值等于TCMP 的值时, 如果中断使能位TCSR.IE[29]=1,中断将发生. 超时周期= (Period of timer clock input) * (8-bit PRESCALE + 1) * (24-bit TCMP) 注意1: 不要写“0” 或者 “1” 到 TCMP, 否则定时器将进入不知道的状态. 注意2: 当定时器操作在连续计数模式时, 如果软件写一个新的值到TCMP, 24-bit 上数计数器将连续计数; 当定时器操作在其它模式时, 如果软件写一个新的值到TCMP, 24-bit 上数计数器将重新开始计数并与新的TCMP的值比较.

定时器中断状态寄存器(TISR)

寄存器	偏移	R/W	描述	复位值
TISR0	TMR_BA+0x08	R/W	定时器0 中断状态寄存器	0x0000_0000
TISR1	TMR_BA+0x28	R/W	定时器1 中断状态寄存器	0x0000_0000

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

Bits	描述	
[31:2]	-	**预留**
[1]	**TWF**	**定时器唤醒标志** 如果定时器导致CPU从睡眠模式唤醒,这个比特将被置为高. 这个比特必须由软件写”1”清除. 0 = 定时器没有导致CPU唤醒. 1 = 定时器超时导致CPU从睡眠或者掉电模式唤醒.
[0]	**TIF**	**定时器中断标志** 这个比特指示定时器的中断状态. 当内部24比特上数计数器的值与比较寄存器(TCMP)的值匹配,而且定时器中断使能比特TCSR.IE[29] = “1”时,硬件将设置这个比特. TIF 这个比特写“1” 清除.

定时器数据寄存器(TDR)

寄存器	偏移	R/W	描述	复位值
TDR0	TMR_BA+0x0C	R	定时器0 数据寄存器	0x0000_0000
TDR1	TMR_BA+0x2C	R	定时器1 数据寄存器	0x0000_0000

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

Bits	描述	
[31:24]	**-**	**预留**
[23:0]	**TDR[23:0]**	定时器数据寄存器,用来存放当前的计数值.

定时器捕捉数据寄存器(TCAP)

寄存器	偏移	R/W	描述	复位值
TCAP0	TMR_BA+0x10	R	定时器0 捕捉数据寄存器	0x0000_0000
TCAP1	TMR_BA+0x30	R	定时器1捕捉数据寄存器	0x0000_0000

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

Bits	描述	
[31:24]	**-**	**预留**
[23:0]	**TCAP[23:0]**	**定时器捕捉数据寄存器** 当 TEXEN (TEXCON[3]) 被设, RSTCAPn (TEXCON[4]) 等于 “0”, 并且TEX引脚上按照TEX_EDGE (TEXCON[2:1])的设定发送了转变, 内部24比特上数计数器的值将被存到TCAP.用户可以读这个寄存器得到计数值.

定时器外部控制寄存器(TEXCON)

寄存器	偏移	R/W	描述	复位值
TEXCON0	TMR_BA+0x14	R/W	定时器0外部控制寄存器	0x0000_0000
TEXCON1	TMR_BA+0x34	R/W	定时器0外部控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
-							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
-							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
-							**CAP_MODE**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TCDB	**TEXDB**	**TEXIEN**	**RSTCAPN**	**TEXEN**	**TEX_EDGE**		**TX_PHASE**

Bits	描述	
[31:9]	**-**	**预留**
[8]	**CAP_MODE**	**捕获模式选择** 0 = 定时器计数复位功能或者 定时器捕获功能的free-counting 模式. 1 = 定时器捕获功能的Trigger-counting 模式.
[7]	**TCDB**	**定时器计数引脚De-bounce 使能位** 1 = 使能 De-bounce. 0 = 禁止 De-bounce. 如果这个比特被使能, de-bounce 电路将检测TM0~TM1引脚的边沿.
[6]	**TEXDB**	**定时器外部捕获引脚De-bounce 使能位** 1 = 使能 De-bounce. 0 = 禁止 De-bounce. 如果这个比特被使能, de-bounce 电路将检测TEX 引脚的边沿.
[5]	**TEXIEN**	**定时器外部中断使能位** 1 = 使能 定时器外部中断. 0 =禁止定时器外部中断. 如果定时器外部中断使能, 当TEX引脚根据TEX_EDGE (TEXCON[2:1])的设定发生转变时,定时器将发出外部中断信号通知CPU. 例如, 当TEXIEN = 1, TEXEN = 1, 并且 TEX_EDGE = 00时, TEX 引脚上1 到0 的转变将导致TEXIF (TEXISR[0]) 被设,然后中断将发生.

Bits	描述	
[4]	**RSTCAPN**	**定时器外部复位计数器/捕获模式选择T** 1 = TEX 转变用作定时器计数复位功能. 0 = TEX转变用作定时器捕获功能.
[3]	**TEXEN**	**定时器外部引脚使能** 这个比特使能TEX引脚上的复位/捕获功能. 1 = TEX引脚上的转变将导致捕获或者定时器计数复位. 0 = TEX 引脚被忽略.
[2:1]	**TEX_EDGE [1:0]**	**定时器外部引脚边沿探测** TEX_EDGE: 描述 00: 探测TEX引脚上1 到 0的转变 01: 探测TEX引脚上0 到 1的转变 10: TEX引脚上1 到 0 或者 0 到 1 的转变都将被探测 11: 预留.
[0]	**TX_PHASE**	**定时器外部计数相位** 这个比特指示外部计数引脚的相位. 1 = 外部计数引脚上上升沿将被计数. 0 =外部计数引脚上下降沿将被计数.

定时器外部中断状态寄存器(TEXISR)

寄存器	偏移	R/W	描述	复位值
TEXISR0	TMR_BA+0x18	R/W	定时器0外部中断状态寄存器	0x0000_0000
TEXISR1	TMR_BA+0x38	R/W	定时器1外部中断状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	-	**预留**
[0]	**TEXIF**	**定时器外部中断标志** 这个比特指示定时器外部中断的状态. TEXEN (TEXCON[3]) 等于 “1”, TEX引脚根据TEX_EDGE (TEXCON[2:1])的设定发生转变时硬件将设置这个比特. 这个比特写“1”清除. 例如, 当 TEXEN = 1, TEX_EDGE = 00时, TEX脚上1 到 0 的转变将导致TEXIF 被设.

5.13 UART 接口控制器

NuMicro MINI51™ 系列提供一个通用异步收/发器(UART). 支持流控功能.

5.13.1 概述

通用异步收/发器(UART) 从外设收到数据时执行串到并的转换, 从CPU收到数据时执行并到串的转换. UART 控制器也支持IrDA SIR 功能, 和RS-485 功能. UART 通道支持6种中断类型包括: 发送缓冲FIFO 空中断 (INT_THRE), 接收极限值达到中断(INT_RDA), 线状态中断(校验错误,帧错误,break 中断) (INT_RLS), 接收缓冲超时中断(INT_TOUT), MODEM/唤醒状态中断(INT_MODEM), 和缓冲错误中断(INT_BUF_ERR). 中断号12 (vector number is 28). 系统中断映射请参考Nested Vectored 中断控制器一章.

UART 内嵌一个16字节的发送缓冲(TX_FIFO) 和一个16字节的接收缓冲(RX_FIFO) 可以降低CPU 的中断数,任何时候CPU都可以读UART的状态. 报告的状态信息包括: UART正在进行的发送的类型和条件, 也包含接收数据时可能发生的4个错误条件(校验错误, 帧错误, break 中断和缓冲错误). UART 包含一个可编程的波特率发生器,将输入时钟除频产生收发需要的时钟. 比特率公式为:波特率 = UART_CLK / M * [BRD + 2], M 和 BRD 在波特率除频寄存器(UA_BAUD)中定义. 下表列出了各种条件下UART波特率公式.

表 5.13-1 波特率设定表

Mode	DIV_X_EN	DIV_X_ONE	Divider X	BRD	波特率公式
0	0	0	B	A	UART_CLK / [16 * (A+2)]
1	1	0	B	A	UART_CLK / [(B+1) * (A+2)] , B must >= 8
2	1	1	Don't care	A	UART_CLK / (A+2), A must >=3

表 5.13-2 UART 波特率设定表

System clock = 22.1184 MHz			
Baud rate	Mode0	Mode1	Mode2
921600	Not Support	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=574

		A=34,B=15	
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

5.13.1.1 自动流控

UART控制器支持自动流控功能,使用两个低电平信号: CTSn (clear-to-send) 和 RTSn (request-to-send), 来控制UART和外设(ex: Modem)之间的数据流.当自动流控使能时,直到发送RTSn信号到外设之后,UART才允许接收外设发来的数据. 当RX缓冲中的字节数等于RTS_TRI_LEV (UA_FCR[19:16])的值时, RTSn 信号将被取消. 直到探测到外设发来CTSn信号之后,UART才允许发送数据到外设. 如果CTSn信号消失,UART控制器将停止发送数据.

5.13.1.2 IrDA 模式

UART 控制器也支持串行IrDA (SIR, Serial Infrared) 功能(用户必须设置IrDA_EN (UA_FUN_SEL[1:0]) 来使能IrDA功能). SIR 规格定义了一个短距离、红外、异步串行传输模式, 有一个起始位，8个数据位，和一个停止位.最大速率115.2 Kbps (半双工). IrDA SIR 包含一个IrDA SIR 协议编码/解码器. IrDA SIR 协议是半双工的. 所以不能同时收/发. IrDA SIR 物理层规定了收/发切换最小需要10ms的延迟时间. 延迟特性必须由软件实现.

5.13.1.3 RS-485 模式

UART另一个可选的功能是RS-485 9 比特模式, 方向由RTSn引脚控制.用户也可以软件编程GPIO(P0.1 for RTSn)来控制方向. RS-485 模式通过设定UA_FUN_SEL 寄存器来选择. RS-485 驱动器由RTSn脚来控制. RS-485 模式的许多收/发特性都与UART相同.

5.13.2 特性

- 全双工, 异步通讯
- 收/发独立的16字节缓冲
- 支持硬件自动流控功能(CTSn, RTSn) ,RTSn 流控触发电平可编程
- 接收缓冲触发级别可编程
- 波特率可编程
- 支持CTSn唤醒功能
- 支持 7比特接收缓冲超时功能
- 通过设定寄存器UA_TOR[DLY] ,上一个停止位到下一个起始位之间的发送数据延迟时间可编程
- 支持break 错误, 帧错误, 校验错误和收/发溢出检测功能
- 完全可编程的串行接口特性
  - 数据位可编程: 5-, 6-, 7-, 8个比特
  - 校验位可编程: 奇, 偶, 无校验或者校验位粘连产生和检测
  - 停止位可编程: 1, 1.5, 或者 2个停止位
- 支持IrDA SIR 功能
  - 支持 3/16 比特周期
- 支持 RS-485 功能
  - 支持 RS-485 9比特模式
  - 支持硬件或者软件方向控制(RTSn 引脚)或则软件控制GPIO来控制传输方向

5.13.3 方块图

UART时钟控制方块图如下所示.

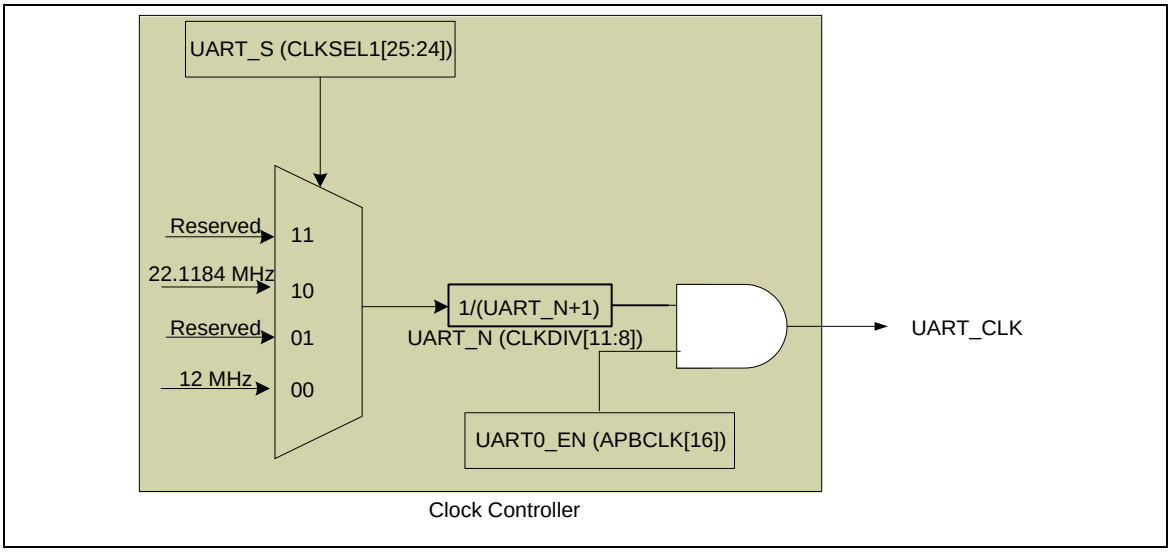


图 5.13-1 UART 时钟控制图

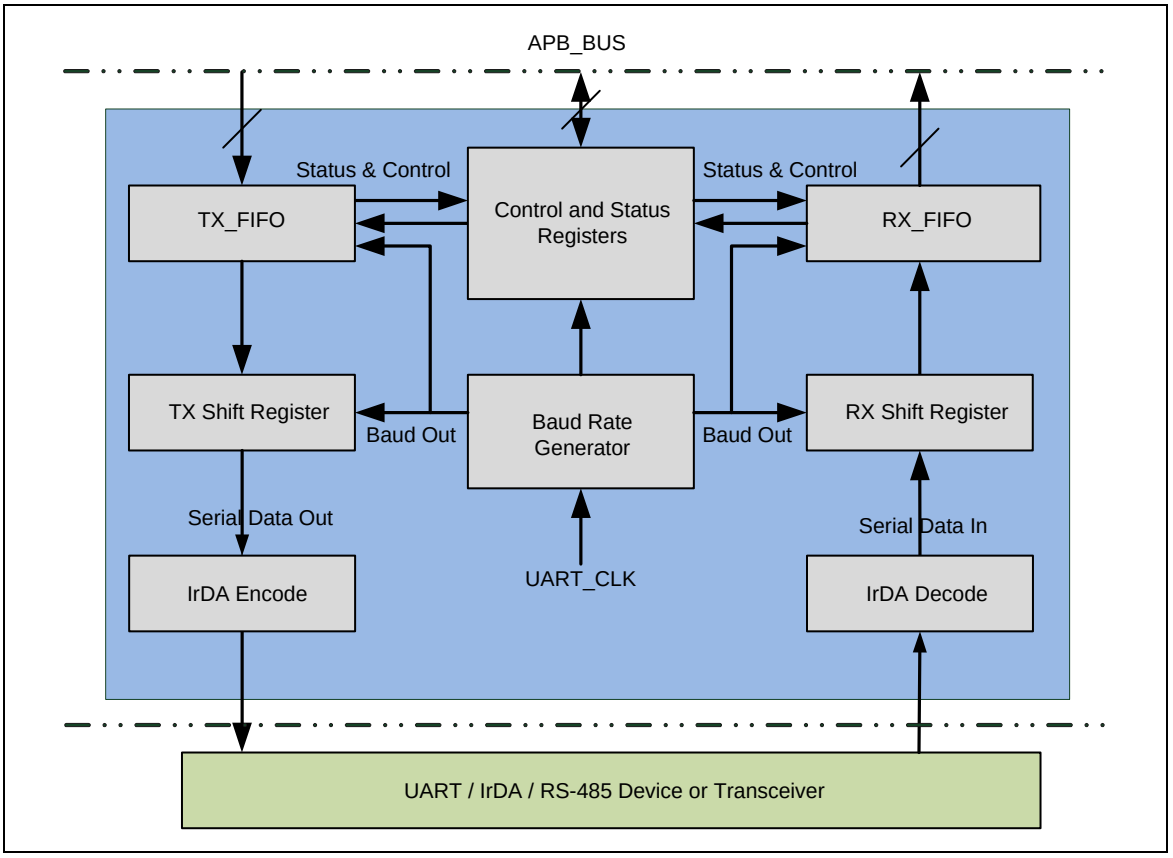


图 5.13-2 UART 方块图

TX_FIFO

有16个字节的发送缓冲用来降低CPU的中断数..

RX_FIFO

有16个字节(加3个比特的错误标志)的接收缓冲用来降低CPU的中断数.

TX shift Register

移位发送数据到串行控制块外.

RX shift Register

移位接收到的数据到串行控制块内.

Baud Rate Generator

将输入的时钟进行分频得到期望的波特率. 请参考波特率公式.

IrDA Encode

IrDA编码控制块.

IrDA Decode

IrDA解码控制块.

控制与状态寄存器

这些寄存器包括缓冲控制寄存器(UA_FCR),缓冲状态寄存器(UA_FSR),和线控制寄存器(UA_LCR). 超时控制寄存器(UA_TOR) 设定超时中断的条件. 这些寄存器也包括中断使能寄存器 (UA_IER) ,中断状态寄存器(UA_ISR) 使能或者禁止中断也鉴别中断源. 共有6种中断, 发送缓冲空中断 (INT_THRE), 接收极限级别达到中断 (INT_RDA), 线状态中断 (校验错误,帧错误, break 错误) (INT_RLS) , 超时中断 (INT_TOUT), MODEM/Wakeup 状态中断(INT_MODEM), 和缓冲错误中断 (INT_BUF_ERR).

5.13.4 功能描述

5.13.4.1 自动流控

下图演示了自动流控方块图.

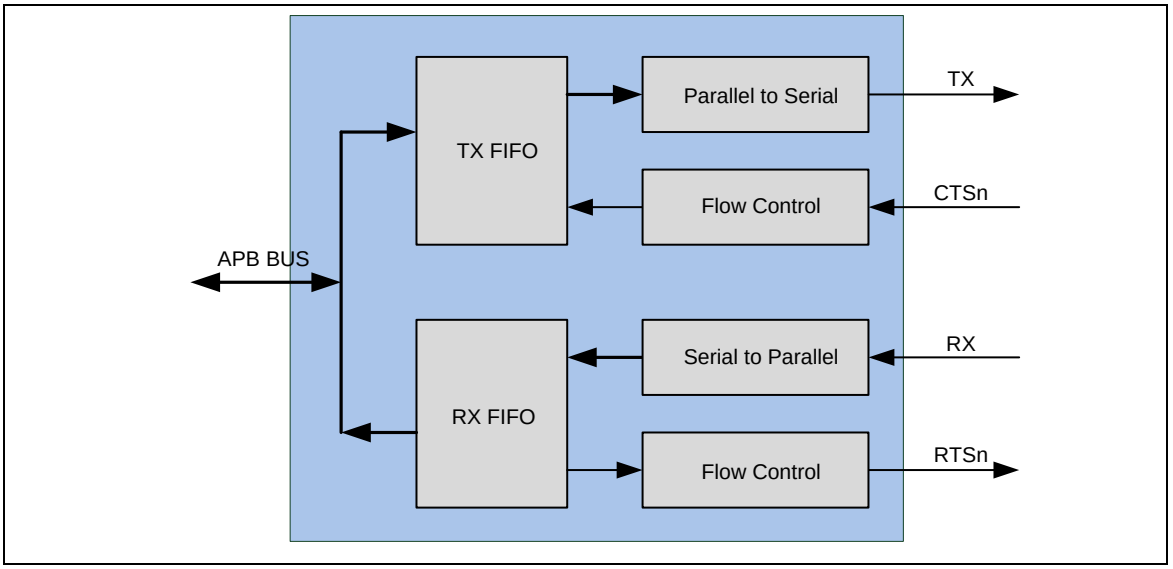


图 5.13-3 自动流控方块图

5.13.4.2 IrDA 模式

UART 支持IrDA SIR (串行红外) 发送编码和接收解码, IrDA 模式可以通过设定UA_FUN_SEL寄存器的IrDA_EN 比特来选择.

IrDA 模式下, UA_BAUD[DIV_X_EN] 比特必须被关闭.

波特率 = Clock / (16 * BRD), BRD 是 UA_BAUD 寄存器的波特率除数值.

下图演示了IrDA控制方块图.

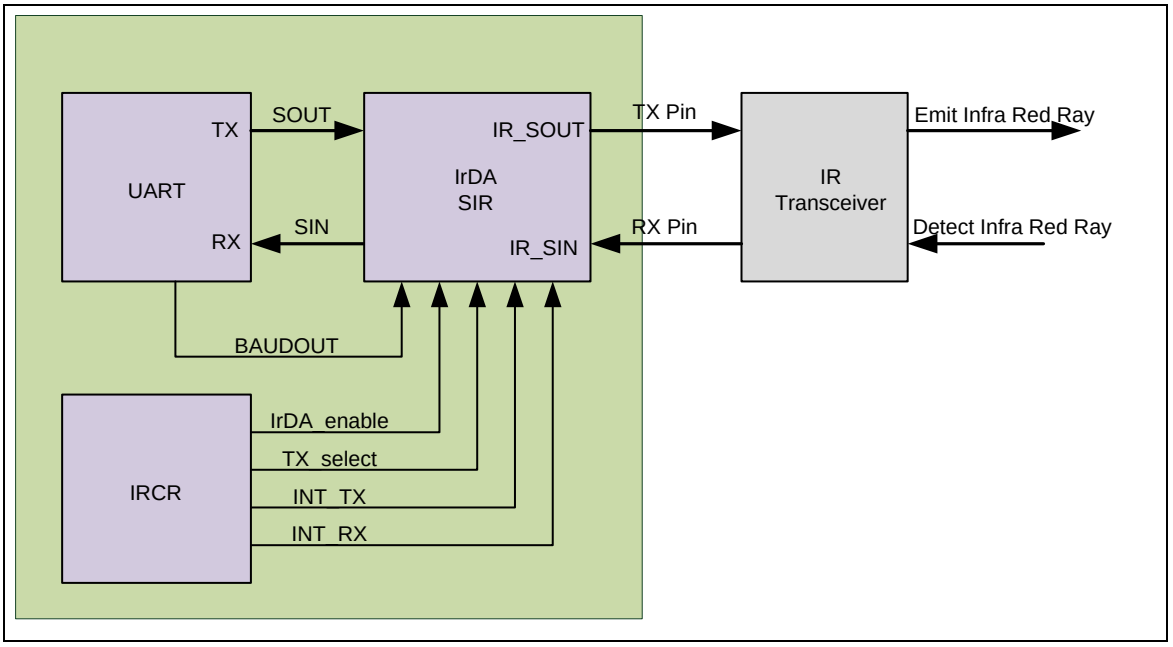


图 5.13-4 IrDA 方块图

IrDA SIR发送编码

IrDA SIR 发送编码器对从UART发出的Non-Return-to Zero (NRZ)码流进行调制. IrDA SIR 物理层规定使用Return-to-Zero, Inverted (RZI) 调制机制,逻辑“0” 表示红外光脉冲. 调制后的输出脉冲流发送到片外的驱动芯片和红外光发射二极管.

正常模式下,发送的脉冲宽度规定为波特率周期的3/16.

IrDA SIR接收解码

IrDA SIR 接收解码器解调从解码输入收到的return-to-zero 比特流,输出NRZ串行比特流给UART接收数据输入. 解码输入空闲时为高电平. (因此, IRCR[6] 缺省应该设为“1”)

当解码输入拉低时，认为收到起始比特.

IrDA SIR 操作

IrDA SIR 编码/解码提供UART数据流和半双工串行SIR接口之间的数据转换. 下图是IrDA 编码/解码波形图:

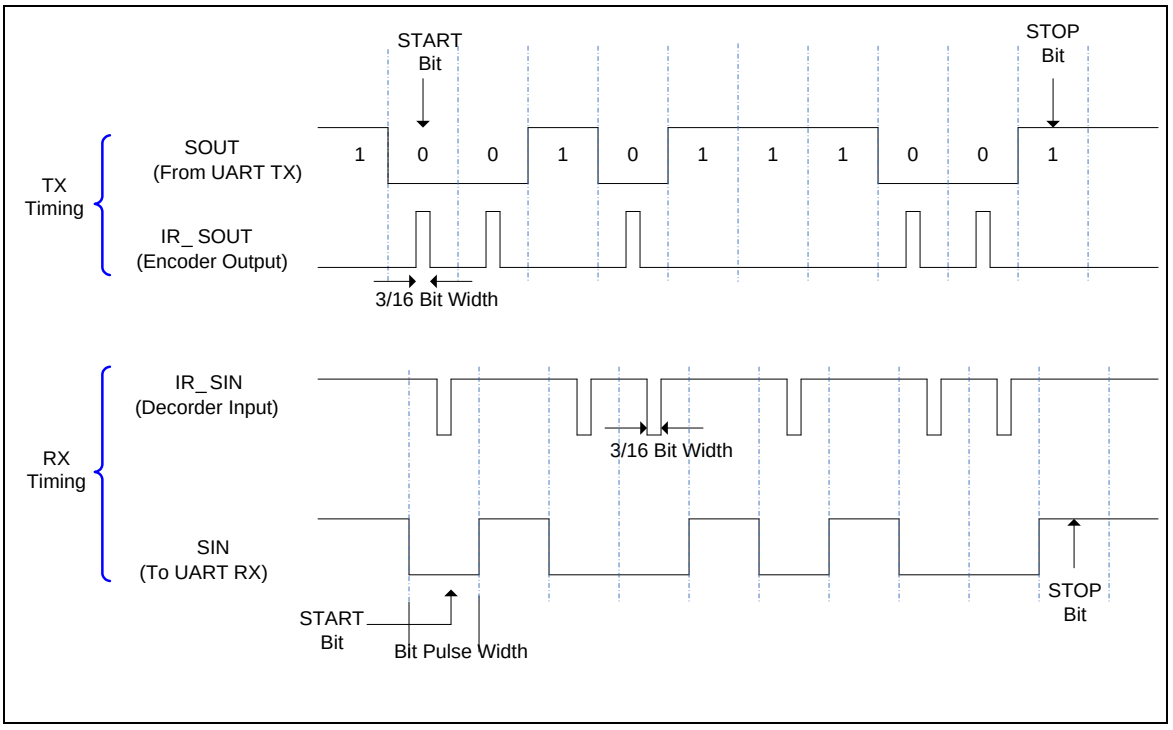


图 5.13-5 IrDA TX/RX 时序图

5.13.4.3 RS-485 功能模式

UART 支持RS-485 9比特模式. 设定UA_FUN_SEL 寄存器可以选择RS-485 模式. RS-485 驱动器由RTSn引脚控制. RS-485 模式下, 许多收/发特性与UART一样.

RS-485 发送模式, 控制器可以被配置成RS-485可寻址从模式.RS-485主发送器通过设定校验位(9th bit)为"1"来标识地址字符. 数据字符, 校验位被配置成"0". 软件可以编程UA_LCR 寄存器来控制第9个比特 (当PBE , EPE 和 SPE 都被设定时, 第9个比特发送0; 当PBE和 SPE被置位,EPE 被清除时, 第9个比特发送1). 控制器支持3种操作模式:RS-485 正常操作模式 (NMM), RS-485 自动地址检测操作模式(AAD) 和 RS-485 自动方向控制操作模式(AUD), 软件通过编程UA_ALT_CSR 寄存器可以选择任何操作模式, 并且软件可以控制上次停止位离开发送缓冲到设定UA_TOR[DLY] 寄存器取消之间的发送延迟时间.

RS-485正常操作模式(NMM)

RS-485 正常操作模式下,首先,软件必须决定在地址字节检测到之前数据是否存入接收缓冲. 如果软件希望检测到地址字节之前,忽略任何数据, 则流程为:设定UART_FCR[RX_DIS] ,然后使能UA_ALT_CSR[RS-485_NMM],之后接收器将忽略任何数据,直到检测到地址(bit9 =1)字节,然后地址字节将被存入接收缓冲. 如果软件希望检测到地址字节之前接收数据, 则流程为:禁止UART_FCR[RX_DIS] ,然后使能UA_ALT_CSR[RS-485_NMM] , 之后接收器将接收任何数据. 如果地址字节(bit9 =1)检测到, 可以发生中断通知CPU, 软件可以通过设定UA_ALT_CSR[RX_DIS]来决定是否接收下一个字节. 如果接收使能, 所有收到的数据都将被存入接收缓冲, 如果接收被禁止, 所有收到的数据都将被忽略,直到检测到下一个地址字节. 如果软件通过设定UA_ALT_CSR[RX_DIS]来

禁止接收,当检测到下一个地址字节时,控制器将清除UA_ALT_CSR[RX_DIS] 比特,地址字节将被存入接收缓冲.

RS-485自动地址检测操作模式(AAD)

自动地址检测操作模式下, 接收器将忽略任何数据,直到地址(bit9 =1)字节被检测到.如果地址字节与UA_ALT_CSR[ADDR_MATCH]的值匹配,地址字节将被存入接收缓冲.所有收到的数据都将存入接收缓冲,直到下一个接收到的地址字节与UA_ALT_CSR[ADDR_MATCH] 的值不匹配.

RS-485自动方向模式(AUD)

RS-485控制器另一个可选的模式就是自动方向控制功能. RS-485 驱动器由RTSn引脚自动控制方向. RTSn 线连到RS-485驱动器使能脚上,以便设定RTS为高(逻辑 “1”)时可以使能RS-485驱动器. 设定RTSn为低(逻辑“0”) 将驱动器置于3态状态. 用户可以设定UA_MCR 寄存器的LEV_RTS 来改变RTSn的驱动电平.

编程顺序:

1. 编程UA_FUN_SEL 寄存器的FUN_SEL 选择RS-485 功能.
2. 编程UA_FCR寄存器的RX_DIS 决定是否打开RS-485接收.
3. 选择RS-485_NMM 还是RS-485_AAD模式.
4. 如果选择RS-485_AAD 模式, 为了自动地址识别需要编程ADDR_MATCH.
5. 编程RS-485_AUD 决定自动方向控制.

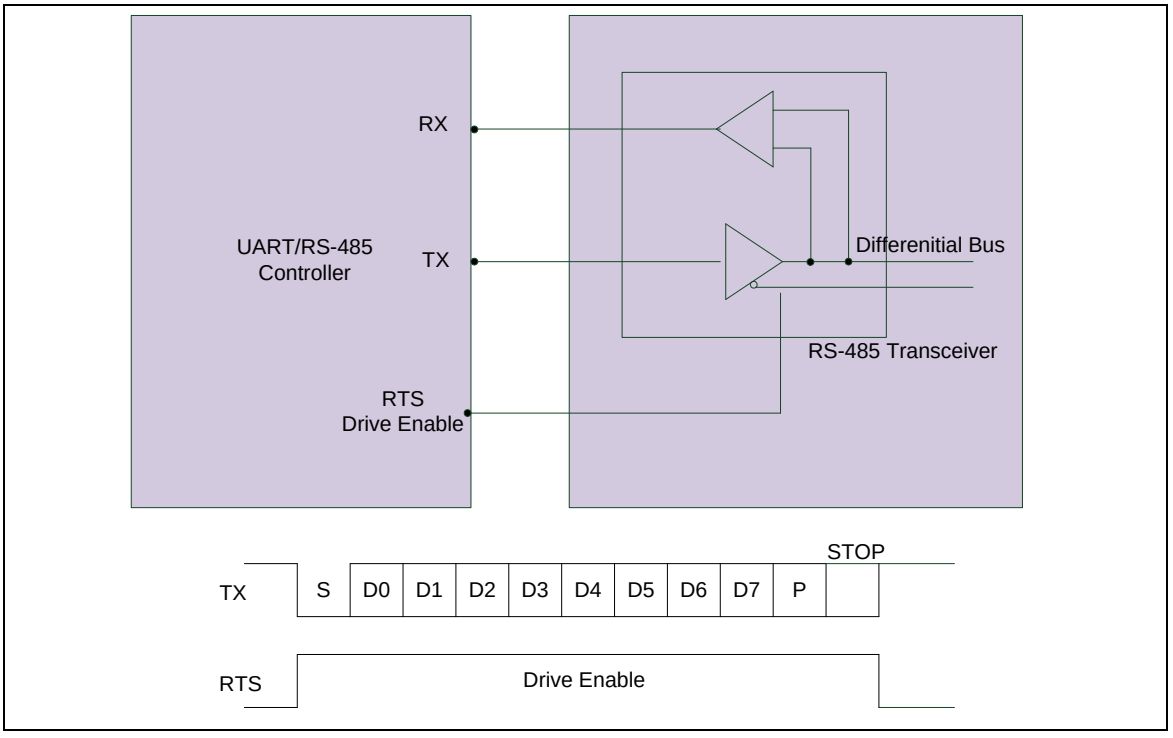


图 5.13-6 RS-485 帧结构

5.13.5 寄存器映射

R: 只读, **W:** 只写, **R/W:** 可读写

寄存器	偏移	R/W	描述	复位值
UART_BA = 0x4005_0000				
UA_RBR	UART_BA+0x00	R	UART接收缓冲寄存器	Undefined
UA_THR	UART_BA+0x00	W	UART发送保持寄存器	Undefined
UA_IER	UART_BA+0x04	R/W	UART中断使能寄存器	0x0000_0000
UA_FCR	UART_BA+0x08	R/W	UART缓冲控制寄存器	0x0000_0000
UA_LCR	UART_BA+0x0C	R/W	UART线控制寄存器	0x0000_0000
UA_MCR	UART_BA+0x10	R/W	UART Modem控制寄存器	0x0000_0000
UA_MSR	UART_BA+0x14	R/W	UART Modem状态寄存器	0x0000_0000
UA_FSR	UART_BA+0x18	R/W	UART缓冲状态寄存器	0x1040_4000
UA_ISR	UART_BA+0x1C	R/W	UART 中断状态寄存器	0x0000_0002
UA_TOR	UART_BA+0x20	R/W	UART超时寄存器	0x0000_0000
UA_BAUD	UART_BA+0x24	R/W	UART波特率除数寄存器	0x0F00_0000
UA_IRCR	UART_BA+0x28	R/W	UART IrDA控制寄存器	0x0000_0040
UA_ALT_CSR	UART_BA+0x2C	R/W	UART选择控制/状态寄存器	0x0000_0000
UA_FUN_SEL	UART_BA+0x30	R/W	UART功能选择寄存器	0x0000_0000

5.13.6 寄存器描述

接收缓冲寄存器(UA_RBR)

寄存器	偏移	R/W	描述	复位值
UA_RBR	UART_BA+0x00	R	UART接收缓冲寄存器	Undefined

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

Bits	描述	
[31:8]	-	**预留**
[7:0]	**RBR[7:0]**	**接收缓冲寄存器（只读）** 读这个寄存器,UART将返回从接收引脚(LSB 优先)收到的8个比特的数据.

发送保持寄存器(UA_THR)

寄存器	偏移	R/W	描述	复位值
UA_THR	UART_BA+0x00	W	UART 发送保持寄存器	Undefined

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

Bits	描述	
[31:8]	-	预留
[7:0]	THR[7:0]	**发送保持寄存器** 写这个寄存器, UART 将送出一个8比特的数据到TX引脚 (LSB 优先).

中断使能寄存器(UA_IER)

寄存器	偏移	R/W	描述	复位值
UA_IER	UART_BA+0x04	R/W	UART 中断使能寄存器	0x0000_0000

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

Bits	描述	
[31:14]	-	**预留**
[13]	AUTO_CTS_EN	**CTS自动流控使能** 1 = 使能CTSn 自动流控. 0 = 禁止CTSn自动流控. 当CTSn 自动流控使能时, 当CTSn信号有效时,UART 将发送数据到外设 (CTSn 信号无效时,UART不会发送数据).
[12]	AUTO_RTS_EN	**RTS Auto Flow 控制 Enable** 1 =使能RTSn 自动流控. 0 =禁止RTSn自动流控. 当RTSn 自动流控使能时， 如果接收缓冲内的字节数等于UA_FCR [RTS_TRI_LEV]的值,UART 将取消RTSn信号.
[11]	TIME_OUT_EN	**超时计数器使能** 1 = 使能超时计数器. 0 = 禁止超时计数器.
[10:7]	-	**预留**
[6]	WAKE_EN	**唤醒CPU 功能使能** 0 = 禁止 UART 唤醒 CPU 功能. 1 = 使能唤醒功能, 当系统在睡眠模式时, 外部CTSn 信号改变将唤醒.
[5]	BUF_ERR_IEN	**缓冲错误中断使能**

Bits	描述	
		0 = 禁止INT_BUF_ERR. 1 = 使能INT_BUF_ERR.
[4]	**RTO_IEN**	**RX 超时中断使能** 0 = 禁止 INT_TOUT. 1 = 使能 INT_TOUT.
[3]	**MODEM_IEN**	**Modem 状态中断使能** 0 = 禁止INT_MODEM. 1 = 使能INT_MODEM.
[2]	**RLS_IEN**	**接收线状态中断使能** 0 = 禁止INT_RLS. 1 = 使能INT_RLS.
[1]	**THRE_IEN**	**发送保持寄存器空中断使能** 0 = 禁止INT_THRE. 1 = 使能 INT_THRE.
[0]	**RDA_IEN**	**接收数据可得中断使能** 0 = 禁止INT_RDA. 1 = 使能 INT_RDA.

缓冲控制寄存器(UA_FCR)

寄存器	偏移	R/W	描述	复位值
UA_FCR	UART_BA+0x08	R/W	UART缓冲控制寄存器	0x0000_0000

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

Bits	描述	
[31:20]	-	预留
[19:16]	RTS_TRI_LEV [3:0]	**RTSn自动流控触发级别** RTS_TRI_LEV / 触发级别(字节) 0000 / 01 0001 / 04 0010 / 08 0011 / 14 Others / 14 **注意**: 这个域只用于自动RTSn流控.
[15:9]	-	预留
[8]	RX_DIS	**接收禁止寄存器** 控制接收是否被禁止(设 “1”禁止接收). 1 = 禁止接收. 0 = 使能接收. **注意:** 这个域只用于RS-485 正常操作模式. UA_ALT_CSR[RS-485_NMM]被编程之前, 这个比特应该先被编程..
[7:4]	RFITL[3:0]	**接收缓冲中断(INT_RDA) 触发级别** 当接收缓冲内的字节数等于RFITL时,RDA_IF将被置 (如果UA_IER[RDA_IEN] 被使能, 中断将发生).

<table>
<tr><th>Bits</th><th>描述</th><th></th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>RFITL</th><th>INTR_RDA触发级别(字节)</th></tr>
<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>
<tr><td>Others</td><td>14</td></tr>
</table>
</td></tr>
<tr><td>[3]</td><td>-</td><td>预留</td></tr>
<tr><td>[2]</td><td>TFR</td><td>发送域软件复位
当 TX_RST 被置时, 在发送缓冲中的所有字节和发送内部状态机都将被清除.
0 = 写“0” 无效.
1 = 写 “1” 将复位发送内部状态机和发送指针.
注意:在3个UART时钟周期之后这个比特将自动清0.</td></tr>
<tr><td>[1]</td><td>RFR</td><td>接收域软件复位
当 RX_RST 被置时, 在接收缓冲中的所有字节和接收内部状态机都将被清除.
0 = 写“0” 无效.
1 = 写 “1” 将复位接收内部状态机和接收指针.
注意:在3个UART时钟周期之后这个比特将自动清0.</td></tr>
<tr><td>[0]</td><td>-</td><td>预留</td></tr>
</table>

线控制寄存器(UA_LCR)

寄存器	偏移	R/W	描述	复位值
UA_LCR	UART_BA+0x0C	R/W	UART线控制寄存器	0x0000_0000

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

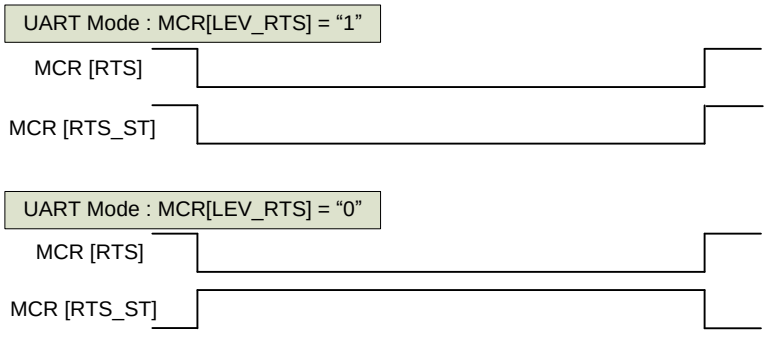
Bits	描述	
[31:7]	-	预留
[6]	BCB	**Break 控制比特** 当这个比特被置为 “1”时, 串行数据输出引脚(TX) 将被迫使处于Spacing State (逻辑“0”). 这个比特只是影响TX引脚不会影响内部发送逻辑.
[5]	SPE	**粘连校验使能** 0 = 禁止校验粘连. 1 = 当比特PBE, EPE 和 SPE 都被设时, 发送和接收检测的校验比特都是0. 当 PBE 和 SPE被设, EPE被清除时, 发送和接收检测的校验比特都是1.
[4]	EPE	**校验选择比特** 0 = 奇校验.发送和接收到的比特(加校验位),”1”的个数是奇数个. 1 = 偶校验. 发送和接收到的比特(加校验位),”1”的个数是偶数个. 只有当比特3 (校验使能比特) 被置时,这个比特才有作用.
[3]	PBE	**校验使能比特** 0 = 校验比特不会产生 (发送时) 或者检测(接收时) . 1 =在最后一个字节和停止位之间的串行数据校验比特将产生 (发送时) 或者检测.
[2]	NSB	**“停止位”的位数** 0 = 一个“停止位”将被产生. 1 = 当数据长度是5个比特时,一个半 “停止位” 将被产生; 当数据长度是6-，7-，8个比特时,两个“停止位”将被产生.
[1:0]	WLS[1:0]	**比特长度**

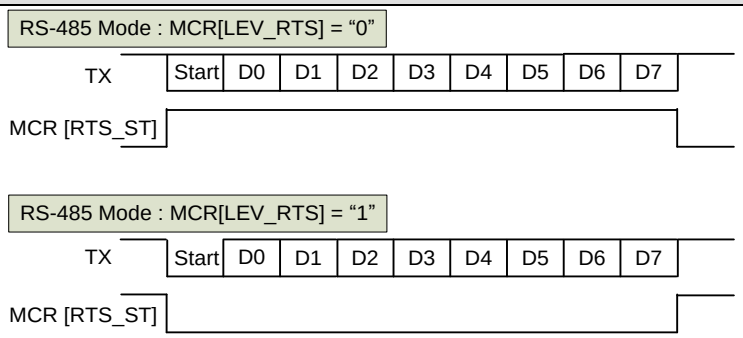
Bits	描述			
		WLS	比特长度	
		00	5 bits	
		01	6 bits	
		10	7 bits	
		11	8 bits	

MODEM控制寄存器(UA_MCR)

寄存器	偏移	R/W	描述	复位值
UA_MCR	UART_BA+0x10	R/W	UART Modem控制寄存器	0x0000_0000

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

Bits	描述	
[31:14]	-	预留
[13]	RTS_ST	RTSn 引脚状态 (只读) 这个比特是RTSn引脚的输出状态.
[12:10]	-	预留
[9]	LEV_RTS	RTSn 触发电平 这个比特可以改变 RTSn 的触发电平. 0 =低电平触发. 1 =高电平触发. 

Bits	描述	
		
[8:2]	-	预留
[1]	RTSn	**RTSn (Request-To-Send) 信号** LEV_RTS / RTSn / RTS_ST 0 (低电平触发) / 0 / 1 0 (低电平触发) / 1 / 0 1 (高电平触发) / 0 / 0 1 (高电平触发) / 1 / 1
[0]	-	预留

Modem状态寄存器(UA_MSR)

寄存器	偏移	R/W	描述	复位值
UA_MSR	UART_BA+0x14	R/W	UART Modem状态寄存器	0x0000_0000

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

Bits	描述	
[31:9]	-	**预留**
[8]	LEV_CTS	**CTSn 触发电平** 这个比特将改变CTSn 的触发电平. 0 = 低电平触发. 1 = 高电平触发.
[7:5]	-	预留
[4]	CTS_ST	**CTSn引脚状态 (只读)** 这个比特为CTSn引脚的状态.
[3:1]	-	**预留**
[0]	DCTSF	**检测CTSn 状态改变标志 (只读)** 无论何时只要CTSn的输入状态改变,这个比特将被设, 如果UA_IER[MODEM_IEN]使能, 将发生Modem 中断通知CPU. **注意**: 这个比特是只读的, 但是可以写"1" 清除.

缓冲状态寄存器(UA_FSR)

寄存器	偏移	R/W	描述	复位值
UA_FSR	UART_BA+0x18	R/W	UART 缓冲状态寄存器	0x1040_4000

31	30	29	28	27	26	25	24
-			TE_FLAG	-			TX_OVER_IF
23	22	21	20	19	18	17	16
TX_FULL	TX_EMPTY	TX_POINTER					
15	14	13	12	11	10	9	8
RX_FULL	RX_EMPTY	RX_POINTER					
7	6	5	4	3	2	1	0
-	BIF	FEF	PEF	RS-485_ ADD_DETF	-		RX_OVER_IF

Bits	描述	
[31:29]	-	预留
[28]	TE_FLAG	**发送缓冲空标志 (只读)** 当发送缓冲为空(UA_THR)并且最后一个字节的停止位已经被发送时,硬件将设置这个比特. 当发送缓冲非空或者最后一个字节还没有发送完时,这个比特将自动被清除.
[27:25]	-	预留
[24]	TX_OVER_IF	**发送缓冲溢出中断标志(只读)** 如果发送缓冲(UA_THR) 已经满了, 又写UA_THR 将导致这个比特被设为逻辑 “1”. **主语**: 这个比特是只读的, 但是可以写“1” 清除.
[23]	TX_FULL	**发送缓冲满(只读)** 这个比特指示发送缓冲是否已经满了. 当TX_POINTER 等于16时这个比特将被置位, 否则将由硬件清除.
[22]	TX_EMPTY	**发送缓冲空(只读)** 这个比特指示发送缓冲是否为空. 当发送缓冲中的最后一个字节已经被发送到发送移位寄存器时, 硬件将设置这个比特为高.当写数据到THR时, 硬件将自动清除这个比特.
[21:16]	TX_POINTER [5:0]	**发送缓冲指针 (只读)** 这个域指示了发送缓冲指针.当CPU写一个字节到UA_THR时, TX_POINTER将加1. 当发送缓冲中的一个字节发送到发送移位寄存器时, TX_POINTER 将减1.

Bits	描述	
[15]	**RX_FULL**	**接收缓冲满 (只读)** 这个比特指示接收缓冲是否已经满了. 当RX_POINTER 等于16时这个比特将被置位, 否则将由硬件清除.
[14]	**RX_EMPTY**	**接收缓冲 空(只读)** 这个比特指示接收缓冲是否为空. 当接收缓冲中的最后一个字节已经被CPU读走时, 硬件将设置这个比特为高.当UART收到数据时，硬件将自动清除这个比特.
[13:8]	**RX_POINTER [5:0]**	**接收缓冲指针(只读)** 这个域指示了接收缓冲指针.当UART从外设收到一个字节时, RX_POINTER将加1. 当CPU从接收缓冲读走一个字节时, RX_POINTER 将减1.
[7]	**-**	**预留**
[6]	**BIF**	**Break 中断标志 (只读)** 无论何时接收引脚(RX)被保持在“spacing state” (逻辑 0) 超过一个完整的字节(就是, “start bit” + data 比特 + parity + stop 比特的总时间)传输时间,这个比特将被置位.CPU写”1”清0. **注意**: 这个比特是只读的, 但是可以写”1” 清0.
[5]	**FEF**	**帧错误标志(只读)** 无论何时收到的字符没有有效的停止位时(就是, 停止位的位置探测到 “0”), 这个比特将被置为 “1”, CPU写”1”清0. **注意**: 这个比特是只读的, 但是可以写”1” 清0.
[4]	**PEF**	**校验错误标志 (只读)** 无论何时收到的字符没有有效的校验位时, 这个比特将被置为 “1”. CPU写”1”清0. **注意**: 这个比特是只读的, 但是可以写”1” 清0.
[3]	**RS-485_ ADD_DETF**	**RS-485 地址字节检测标志 (只读)** RS-485模式下,设置UA_ALT_CSR[RS-485_ADD_EN] 之后,无论何时接收器收到地址字节(bit9 = ‘1’),这个比特将被置为“1” , CPU写”1”清0.. **注意:** 这个域只用于RS-485 功能. **注意**: 这个比特是只读的, 但是可以写”1” 清0.
[2:1]	**-**	**预留**
[0]	**RX_OVER_IF**	**接收缓冲溢出标志 (只读)** 当接收缓冲溢出时,这个比特将被置. 如果收到的字节数大于 RX_FIFO (UA_RBR) 的大小, 16 个字节, 这个比特将被置. **注意**: 这个比特是只读的, 但是可以写”1” 清0.

中断状态控制寄存器(UA_ISR)

寄存器	偏移	R/W	描述	复位值
UA_ISR	UART_BA+0x1C	R/W	UART 中断状态寄存器	0x0000_0002

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

Bits	描述	
[31:14]	-	预留
[13]	BUF_ERR_INT	**缓冲错误中断指示 (只读)** 如果BUF_ERR_IEN 和 BUF_ERR_IF都被设，这个比特将被置为”1”.
[12]	TOUT_INT	**超时中断 (只读)** 如果TOUT_IEN 和TOUT_IF都被设，这个比特将被置为”1”. 1 = Tout中断发生. 0 = 没有Tout中断发生.
[11]	MODEM_INT	**MODEM 状态中断指示 (只读)** 如果MODEM_IEN 和MODEM_IF都被设，这个比特将被置为 “1”. 1 = Modem中断发生. 0 = 没有Modem中断发生.
[10]	RLS_INT	**接收线状态中断指示(只读)** 如果RLS_IEN和RLS_IF都被设，这个比特将被置为“1”. 1 = RLS 中断发生 0 = 没有RLS 中断发生.
[9]	THRE_INT	**发送保持寄存器空中断指示 (只读)** 如果THRE_IEN和THRE_IF都被设，这个比特将被置为“1”. 1 = THRE中断发生. 0 = 没有THRE中断发生.

Bits	描述	
[8]	**RDA_INT**	**接收数据可得中断指示 (只读)** 如果RDA_IEN和RDA_IF都被设，这个比特将被置为“1”. 1 = The RDA中断发生. 0 = 没有RDA中断发生.
[7:6]	**-**	**预留**
[5]	**BUF_ERR_IF**	**缓冲错误中断标志 (只读)** 当发送或者接收缓冲溢出(TX_OVER_IF或者RX_OVER_IF被设)时,这个比特将被置.当BUF_ERR_IF被设时，传输可能不正确. 如果UA_IER[BUF_ERR_IEN] 被使能，缓冲错误中断将发生. 注意: TX_OVER_IF 和 RX_OVER_IF 都被清除时,这个比特将被清除.
[4]	**TOUT_IF**	**超时中断标志 (只读)** 当接收缓冲非空,并且在TOIC的时间内没有再收到数据,这个比特将被置. 如果UA_IER[TOUT_IEN] 被使能, 超时中断将发生. 注意:这个比特是只读的,用户可以读UA_RBR (接收数据可得) 来清除.
[3]	**MODEM_IF**	**MODEM 中断标志(只读)** 当CTSn 引脚发生状态改变时 (DCTSF=1) ,这个比特将被置. 如果UA_IER[MODEM_IEN] 被使能, Modem中断将发生. 注意: 这个比特是只读的,当DCTSF 比特被写 “1” 清除时,这个比特将被清成0.
[2]	**RLS_IF**	**接收线中断标志 (只读)** 当接收数据有校验错误, 帧错误或者 break 错误时 (3个比特 BIF, FEF 和 PEF中至少一个比特被设) ,这个比特将被置. 如果UA_IER[RLS_IEN] 被使能, RLS中断将发生. 注意: RS-485功能下, 这个域包含“receiver detect any address byte received address byte character (bit9 = “1”) bit". 注意: 这个比特是只读的,当所有比特BIF, FEF 和 PEF都被清成0时,这个比特将被清成“0”.
[1]	**THRE_IF**	**发送保持寄存器空中断标志 (只读)** 当发送缓冲中的最后一个数据被传输到发送移位寄存器时这个比特将被设. 如果UA_IER [THRE_IEN] 被使能, THRE 中断将发生. **注意**: 这个比特是只读的,当往THR寄存器写数据的时候这个比特将被自动清除.
[0]	**RDA_IF**	**接收数据可得中断标志 (只读)** 当接收缓冲中的字节数等于RFITL时,RDA_IF 将被设. 如果 UA_IER[RDA_IEN] 被使能,RDA中断将发生. 注意: 这个比特是只读的, 当接收缓冲中的未读字节数掉到极限级别以下时 (RFITL),这个比特将被清除.

表 5.13-3 UART 中断源和标志表

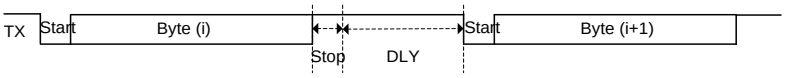
UART 中断源	中断使能比特	中断指示	中断标志	清除方法
缓冲区错误中断	BUF_ERR_IEN	BUF_ERR_INT	HW_BUF_ERR_IF =	Write ‘1’ to

UART 中断源	中断使能比特	中断指示	中断标志	清除方法
(INT_BUF_ERR)			(TX_OVER_IF or RX_OVER_IF or BIF or PEF or FEF)	TX_OVER_IF/ RX_OVER_IF/ BIF/PEF/FEF
接收超时中断 (INT_TOUT)	RTO_IEN	TOUT_INT	TOUT_IF	读 UA_RBR
Modem 状态中断 (INT_MODEM)	MODEM_IEN	MODEM_INT	MODEM_IF = (DCTSF)	写 “1”到 DCTSF
接收线状态中断 (INT_RLS)	RLS_IEN	RLS_INT	RLS_IF = (BIF or FEF or PEF or RS-485_ ADD_DETF)	写 “1” 到 BIF/FEF/PEF/ RS-485_ ADD_DETF
传输保持寄存器空中断 (INT_THRE)	THRE_IEN	THRE_INT	THRE_IF	写 UA_THR
接收数据有效中断 (INT_RDA)	RDA_IEN	RDA_INT	RDA_IF	读 UA_RBR

超时寄存器(UA_TOR)

寄存器	偏移	R/W	描述	复位值
UA_TOR	UART_BA+0x20	R/W	UART超时寄存器	0x0000_0000

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

Bits	描述	
[31:16]	-	预留
[15:8]	DLY[7:0]	**发送延迟时间** 这个域用来编程上一个停止位和下一个起始位之间的发送延迟时间 
[7:0]	TOIC[7:0]	**超时中断比较器** 无论何时接收缓冲收到一个新的数据,超时计数器复位并且开始计数 (计数时钟 = 波特率). 一旦超时计数器的值 (TOUT_CNT)等于TOIC的值,如果UA_IER[RTO_IEN]被使能, 接收超时中断 (INT_TOUT) 将发生. 新接收到一个字节或则接收缓冲为空将清除这个比特.

波特率除数寄存器(UA_BAUD)

寄存器	偏移	R/W	描述	复位值
UA_BAUD	UART_BA+0x24	R/W	UART波特率除数寄存器	0x0F00_0000

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

Bits	描述	
[31:30]	-	预留
[29]	DIV_X_EN	**除数 X 使能** BRD = 波特率除数, 波特率公式为 **波特率= Clock / [M * (BRD + 2)]**; M 的缺省值是16. 0 = 禁止除数 X (M = 16). 1 = 使能除数X (M = X+1, 但是 DIVIDER_X [27:24] 必须 >= 8). 更多信息请参考下表. **注意**: IrDA 模式下, 这个比特必须被禁止.
[28]	DIV_X_ONE	**除数 X 等于 1** 0 = 除数 M = X (M = X+1, 但是 DIVIDER_X[27:24] 必须 >= 8). 1 = 除数 M = 1 (M = 1, 但是 BRD [15:0] 必须 >= 3). 更多信息请参考下表.
[27:24]	DIVIDER_X[3:0]	**除数 X** 波特率除数 M = X+1.
[23:16]	-	预留
[15:0]	BRD[15:0]	**波特率除数** 这个域是波特率除数.

表 5.13-4 UART 波特率设定表

Mode	DIV_X_EN	DIV_X_ONE	Divider X	BRD	波特率公式
0	0	0	B	A	UART_CLK / [16 * (A+2)]
1	1	0	B	A	UART_CLK / [(B+1) * (A+2)] , B 必须 >= 8
2	1	1	Don't care	A	UART_CLK / (A+2), A 必须 >=3

IrDA控制寄存器(IRCR)

寄存器	偏移	R/W	描述	复位值
UA_IRCR	UART_BA+0x28	R/W	UART IrDA控制寄存器	0x0000_0040

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

Bits	描述	
[31:7]	-	预留
[6]	INV_RX	INV_RX 1 = 反转接收输入信号. 0 = 不要反转.
[5]	INV_TX	INV_TX 1 =反转发送输出信号. 0 =不要反转.
[4:2]	-	预留
[1]	TX_SELECT	TX_SELECT 1 = 使能 IrDA 发送. 0 = 使能 IrDA 接收. 注意: IrDA 模式下, UA_BAUD[DIV_X_EN] 比特必须被禁止 (波特率公式必须是 Clock / 16 * (BRD).
[0]	-	预留

UART 选择控制/状态寄存器(UA_ALT_CSR)

寄存器	偏移	R/W	描述	复位值
UA_ALT_CSR	UART_BA+0x2C	R/W	UART 选择控制/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
-							
15	14	13	12	11	10	9	8
RS485_ADD_EN	-				RS485_AUD	RS485_AAD	RS485_NMM
7	6	5	4	3	2	1	0
-							

Bits	描述	
[31:24]	**ADDR_MATCH [7:0]**	**地址匹配值寄存器** 这个域包含RS-485 的值匹配值. **注意:** 这个域只用于RS-485自动地址检测模式.
[23:16]	-	**预留**
[15]	**RS485_ADD_EN**	**RS-485 地址检测使能** 这个域用来使能RS-485 地址检测模式. 1 = 使能地址检测模式. 0 = 禁止地址检测模式.. **注意:** 这个域用于RS-485 任何模式.
[14:11]	-	**预留**
[10]	**RS485_AUD**	**RS-485 自动方向模式(AUD)** 1 = 使能 RS-485 自动方向操作模式 (AUD). 0 = 禁止RS-485自动方向操作模式(AUD). **注意:** 在RS-485_AAD或者RS-485_NMM 模式下可以使能这个比特.
[9]	**RS485_AAD**	**RS-485 自动地址检测操作模式 (AAD)** 1 =使能 RS-485 自动地址检测操作模式 (AAD). 0 = 禁止RS-485自动地址检测操作模式(AAD). **注意:** RS-485_NMM模式下不可以使能这个比特.

Bits	描述	
[8]	**RS485_NMM**	**RS-485 正常操作模式(NMM)** 1 =使能 RS-485 正常操作模式(NMM). 0 = 禁止RS-485正常操作模式(NMM). **注意:** RS-485_AAD 操作模式下不可以使能这个比特.
[7:0]	**-**	**预留**

UART功能选择寄存器(UA_FUN_SEL)

寄存器	偏移	R/W	描述	复位值
UA_FUN_SEL	UART_BA+0x30	R/W	UART 功能选择寄存器	0x0000_0000

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

Bits	描述	
[31:2]	-	预留
[1:0]	FUN_SEL[1:0]	功能选择

FUN_SEL	描述
00	UART 功能
01	预留.
10	使能 IrDA 功能
11	使能 RS-485 功能.

5.14 看门狗

5.14.1 概述

看门狗定时器的目的是为了在软件出问题之后实现系统复位. 这可以阻止系统无限期挂起. 而且, 看门狗支持将CPU从掉电模式唤醒. 看门狗定时器包含一个18比特计数器，超时间隔可编程. 下表显示看门狗超时间隔的选择和中断以及复位信号的时序.

设置WTE (WTCR[7]) 使能看门狗定时器并且WDT开始计数. 当计数器达到选择的超时间隔时, 看门狗定时器中断标志WT将被设,如果中断使能位WTIE也被设WDT中断将发生.同时一个特定的延迟时间 (1024 * T_{WDT}) 将被延迟. 用户必须设置WTR (WTCR[0]) (看门狗定时器复位) 为高,避免在延迟时间超时之前复位18比特的WDT计数器,避免CPU被看门狗定时器复位.有8个特定的超时间隔可以用看门狗定时器间隔选择比特WTIS(WTCR[10:8])来选则. 如果在特定的延迟时间超时以后,WDT 计数器没有被清除，看门狗定时器将设置看门狗定时器复位标志(WTRF) 为高来复位CPU. 复位将延迟63个WDT时钟(T_{RST}),然后CPU将从向量表(0x0000_0000)重新开始运行. WTRF比特不会被看门狗复位清除. 用户软件可以轮询 WTRF来识别复位源. WDT 也支持唤醒功能. 当芯片掉电时, 看门狗定时器唤醒功能使能比特 (WTCR[4])被设, 如果在特定的延迟时间超时以后,WDT 计数器没有被清除，芯片将被从掉电状态唤醒.

WTIS	WTR Timeout Interval T_{TIS}	Interrupt Period T_{INT}	WTR Timeout Interval (WDT_CLK = 10 KHz) T_{TIS}	WTR Reset Interval (WDT_CLK = 10 KHz) T_{WTR}
000	$2^4 * T_{WDT}$	1024 * T_{WDT}	1.6 ms	104 ms
001	$2^6 * T_{WDT}$	1024 * T_{WDT}	6.4 ms	108.8 ms
010	$2^8 * T_{WDT}$	1024 * T_{WDT}	25.6 ms	128 ms
011	$2^{10} * T_{WDT}$	1024 * T_{WDT}	102.4 ms	204.8 ms
100	$2^{12} * T_{WDT}$	1024 * T_{WDT}	407 ms	512 ms
101	$2^{14} * T_{WDT}$	1024 * T_{WDT}	1.638 s	1.741 s
110	$2^{16} * T_{WDT}$	1024 * T_{WDT}	6.553 s	6.6.656 s
111	$2^{18} * T_{WDT}$	1024 * T_{WDT}	26.214 s	26.316 s

表 5.14-1 看门狗超时间隔选择

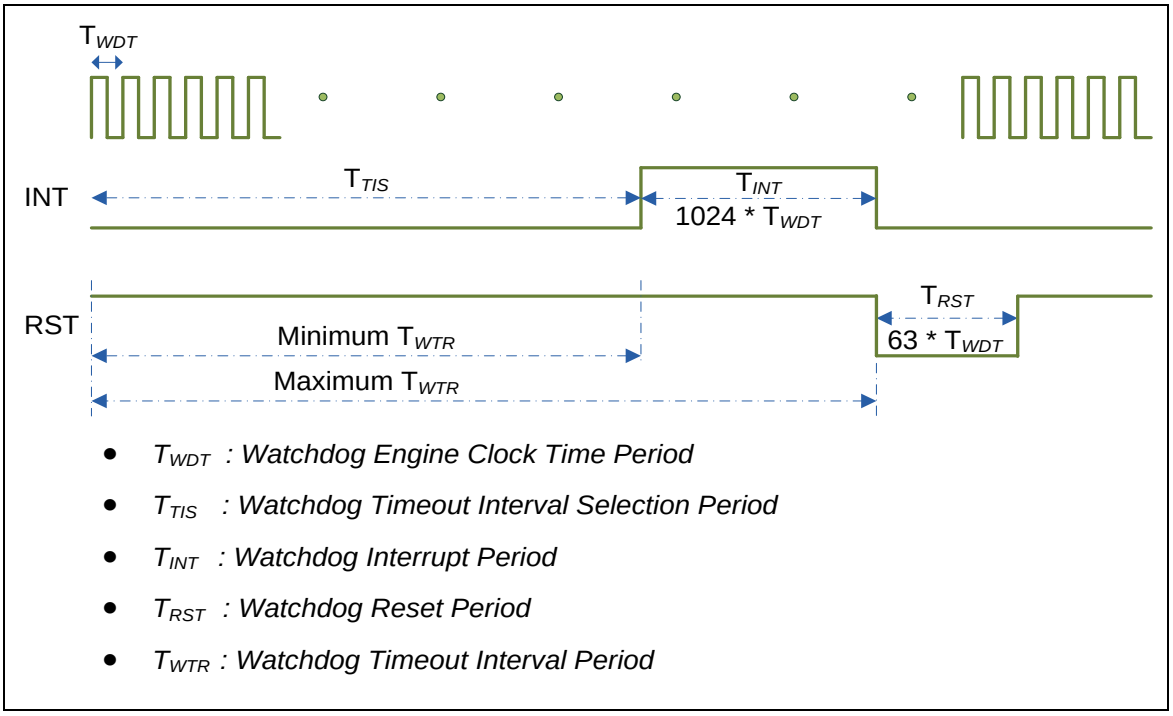


图 5.14-1 中断和复位信号时序

5.14.2 特性

- 在延迟时间超时之前,清除18比特计数器避免超时之前CPU被看门狗定时器复位.
- 超时间隔可选(2^4 ~ 2^{18}) ,超时间隔为 104 ms ~ 26.3168 s (如果WDT_CLK = 10 KHz).
- 复位 周期= (1 / 10 KHz) * 63,如果 WDT_CLK = 10 KHz.

5.14.3 方块图

T看门狗定时器时钟控制和方块图如下所示.

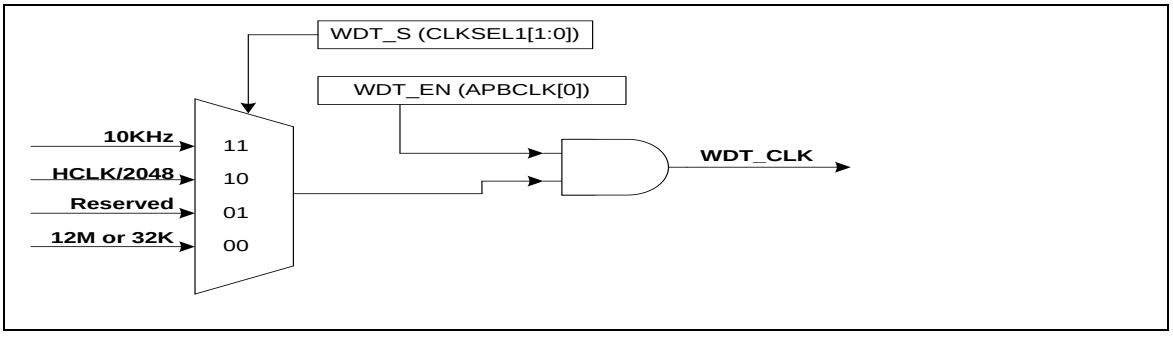


图 5.14-2 看门狗定时器时钟控制

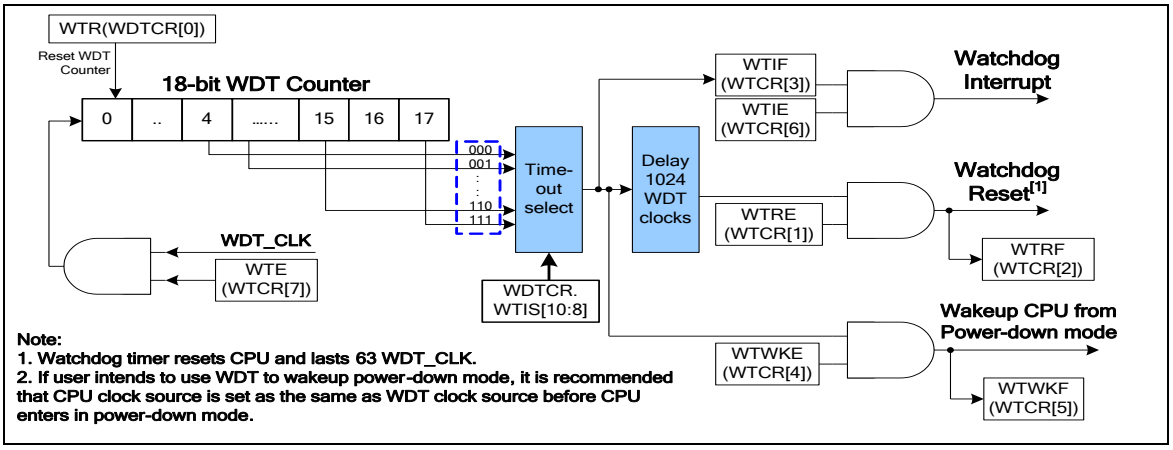


图 5.14-3 看门狗定时器方块图

5.14.4 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	R/W	描述	复位值
WDT_BA = 0x4000_4000				
WTCR	WDT_BA+0x00	R/W	看门狗定时器控制寄存器	0x0000_0700

5.14.5 寄存器描述

看门狗定时器控制寄存器(WTCR)

寄存器	偏移	R/W	描述	复位值
WTCR	WDT_BA+0x00	R/W	看门狗定时器控制寄存器	0x0000_0700

注意: 这个寄存器的所有比特都是写保护的. 编程需要一个开锁序列T, 依次写 “59h”, “16h”, and “88h” 到 RegLockAddr 寄存器,地址GCR_BA + 0x100.

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

Bits	描述

<table>
<tr><th>Bits</th><th colspan="2">描述</th></tr>
<tr><td>[31]</td><td>DBGACK_WDT</td><td>ICE 调试模式应答禁止 (写保护)
0 = ICE 调试模式应答影响看门狗计数.
当ICE调试模式应答时看门狗定时器将停止计数.
1 = ICE 调试模式应答禁止.
ICE调试模式下看门狗定时器将继续计数.</td></tr>
<tr><td>[30:11]</td><td>-</td><td>预留</td></tr>
<tr><td>[10:8]</td><td>WTIS[2:0]</td><td>看门狗定时器间隔选择
这3个比特选择看门狗定时器的超时间隔.
<table>
<tr><th>WTIS</th><th>超时间隔选择</th><th>中断周期</th><th>WTR超时时间 (WDT_CLK=10 kHz)</th></tr>
<tr><td>000</td><td>24 * TWDT</td><td>(24 + 1024) * TWDT</td><td>1.6 ms ~ 104 ms</td></tr>
<tr><td>001</td><td>26 * TWDT</td><td>(26 + 1024) * TWDT</td><td>6.4 ms ~ 108.8 ms</td></tr>
<tr><td>010</td><td>28 * TWDT</td><td>(28 + 1024) * TWDT</td><td>25.6 ms ~ 128 ms</td></tr>
<tr><td>011</td><td>210 * TWDT</td><td>(210 + 1024) * TWDT</td><td>102.4 ms ~ 204.8 ms</td></tr>
<tr><td>100</td><td>212 * TWDT</td><td>(212 + 1024) * TWDT</td><td>409.6 ms ~ 512 ms</td></tr>
<tr><td>101</td><td>214 * TWDT</td><td>(214 + 1024) * TWDT</td><td>1.6384 s ~ 1.7408 s</td></tr>
<tr><td>110</td><td>216 * TWDT</td><td>(216 + 1024) * TWDT</td><td>6.5536 s ~ 6.656 s</td></tr>
<tr><td>111</td><td>218 * TWDT</td><td>(218 + 1024) * TWDT</td><td>26.2144 s ~ 26.3168 s</td></tr>
</table></td></tr>
<tr><td>[7]</td><td>WTE</td><td>看门狗定时器使能
0 = 禁止看门狗定时器(内部计数器将复位).
1 = 使能看门狗定时器.</td></tr>
<tr><td>[6]</td><td>WTIE</td><td>看门狗定时器中断使能
0 = 禁止看门狗定时器中断.
1 =使能看门狗定时器中断.</td></tr>
<tr><td>[5]</td><td>WTWKF</td><td>看门狗定时器唤醒标志
如果看门狗定时器导致CPU从睡眠模式唤醒, 这个比特将被置为高. 这个比特必须由软件写“1”清除.
0 = 看门狗定时器没有导致CPU唤醒.
1 = 看门狗超时导致CPU从睡眠模式唤醒.</td></tr>
<tr><td>[4]</td><td>WTWKE</td><td>看门狗定时器唤醒功能使能标志
0 = 禁止看门狗定时器唤醒CPU功能.
1 = 使能唤醒功能,当看门狗定时器超时时,看门狗可以将CPU从睡眠模式唤醒.</td></tr>
</table>

Bits	描述	
[3]	**WTIF**	**看门狗定时器中断标志** 如果看门狗定时器中断被使能,超时发生时,应件将设置这个比特指示看门狗中断发生. 0 = 看门狗定时器中断没有发生. 1 = 看门狗定时器中断已经发生. **注意**: 这个比特可以写“1” 清除.
[2]	**WTRF**	**看门狗定时器复位标志** 当看门狗定时器导致复位时,硬件将设置这个比特.软件可以读这个比特决定复位源. 软件负责写 “1” 清除这个比特. 如果 **WTRE** 被禁止, 看门狗定时器不会影响这个比特. 0 = 看门狗定时器复位没有发生. 1 = 看门狗定时器复位发生. **注意**:写“1” 可以清除这个比特.
[1]	**WTRE**	**看门狗定时器复位使能** 设置这个比特将使能看门狗定时器复位功能. 0 = 禁止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**	**清除看门狗定时器** 设置这个比特将清除看门狗计数器. 0 = 写 “0” 到这个比特没有作用. 1 = 复位看门狗计数器的值. **注意**: 几个时钟周期之后这个比特会自动清除.

6 ARM® CORTEX™-M0 CORE

6.1 Overview

Cortex™-M0处理器是一个可配置、多级、32位RISC处理器.它有一个AMBA AHB-Lite接口和一个NVIC.还有一个可选择的硬件调试功能.这个处理器能执行Thumb指令，并且和其它的Cortex-M处理器兼容。支持两种模式：Thread 和 Handler模式.发生异常时将进入Handler模式 .异常返回只能在Handler模式下执行.系统复位时将进入Thread模式,异常返回时也可以进入Thread模式. 图 6.1-1显示了处理器的功能控制.

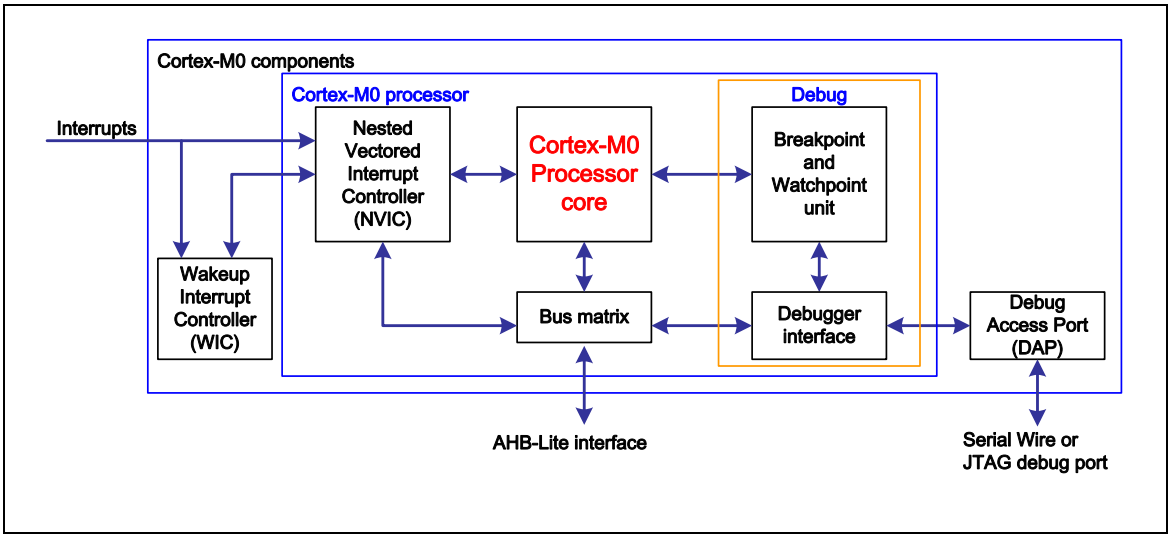


图 6.1-1 功能方块图

6.2 特性

- 低门数处理器，有下列特性:
  - The ARMv6-M Thumb®指令集
  - Thumb-2技术
  - 一个24-bit SysTick 定时器
  - 一个 32-bit硬件乘法器
  - 系统接口支持小端(little-endian)数据访问
  - 有确定性,固定延迟,中断处理的能力
  - Load/store-multiples 和 multi-cycle-multiplies指令能被中断并丢弃，便于快速的处理中断
  - C Application Binary Interface 兼容的异常模型

    ARMv6-M, C Application Binary Interface (C-ABI) 兼容的异常模型，使中断处理函数可以纯粹是C代码，无需汇编
  - Wait For Interrupt (WFI), Wait For Event (WFE) 指令可以进入低功耗idle模式, 或者从中断sleep-on-exit 特性返回
- NVIC 有下列特性:
  - 32个外部中断输入，每个中断有4级优先级
  - 专用的不可屏蔽中断:non-Maskable Interrupt (NMI)
  - 支持电平和边沿中断触发模式
  - 唤醒中断控制器 (WIC), 提供超低功耗idle 模式支持
- 调试:
  - 4个硬件断点
  - 2个 watch points
  - Program Counter Sampling Register (PCSR) for non-intrusive code profiling
  - 单步和向量捕捉能力
- 总线接口:
  - 单个32-bit AMBA-3 AHB-Lite系统接口，提供系统外设和内存的简单整合
  - 单个 32-bit slave port，支持 DAP (Debug Access Port)

6.3 系统定时器(SysTick)

Cortex-M0内核内嵌一个系统定时器, SysTick. SysTick 提供一个简单的, 24-bit 写清除, 递减型, 自装载(wrap-on-zero)计数器,同时具有灵活的控制机制. 这个计数器可以用作实时操作系统(RTOS)的tick时钟 或者当作简单的计数器.

当系统定时器使能时, 它将从SysTick Current Value 寄存器 (SYST_CVR)的值开始下数一直到0, 并在下一个时钟沿重新加载SysTick Reload Value 寄存器 (SYST_RVR) 中的值, 然后再递减. 当计数器的值变成0的时候, COUNTFLAG 状态位将被设. COUNTFLAG 比特读清0.

复位时SYST_CVR 的值是**未知**的. 使能计数器之前软件应该写这个寄存器以将它清成0. 这可以确保定时器从SYST_RVR开始数而不是一个任意的值.

如果SYST_RVR 的值是0, 定时器将保持0而停止计数.除了关闭定时器使能位，这个机制也可以用来关闭定时器.

细节信息, 请参考文档“ARM® Cortex™-M0 Technical Reference Manual” 和 “ARM® v6-M Architecture Reference Manual”.

6.3.1 系统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	R/W	描述	复位值
SCS_BA = 0xE000_E000				
SYST_CSR	SCS_BA+0x10	R/W	SysTick控制和状态	0x0000_0000
SYST_RVR	SCS_BA+0x14	R/W	SysTick 重新加载值	0x00XX_XXXX
SYST_CVR	SCS_BA+0x18	R/W	SysTick当前值	0x00XX_XXXX

6.3.2 系统定时器控制寄存器描述

SysTick控制和状态(SYST_CSR)

寄存器	偏移	R/W	描述	复位值
SYST_CSR	SCS_BA+0x10	R/W	SysTick控制和状态	0x0000_0000

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

Bits	描述	
[31:17]	-	预留
[16]	COUNTFLAG	自从上次寄存器被读之后，如果定时器计数到0将返回“1”. 计数值从1 到 0转变时，COUNTFLAG将被设. 读COUNTFLAG或者写Current Value 寄存器，COUNTFLAG 将被清成0.
[15:3]	-	预留
[2]	CLKSRC	1 = SysTick使用CPU的时钟作为时钟源. 如果没有外部参考时钟，读这个比特将得到”1”,忽略写. 0 = 时钟源可选, 参考STCLK_S.
[1]	TICKINT	1 = 计数到0 将导致SysTick 触发中断. 但是通过写SysTick Current Value 寄存器而将其清为0是不会触发中断的. 0 =计数到0 不会导致SysTick 触发中断. 软件可以使用COUNTFLAG 来决定计数器是否计数到0.
[0]	ENABLE	1 = 使能计数器. 计数器将工作在自动加载模式. 0 = 计数器将被禁止.

SysTick 重新加载寄存器(SYST_RVR)

寄存器	偏移	R/W	描述	复位值
SYST_RVR	SCS_BA+0x14	R/W	SysTick 重新加载值寄存器	0x00XX_XXXX

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

Bits	描述	
[31:24]	-	预留
[23:0]	RELOAD[23:0]	当计数器计数到0的时候，加载到Current Value 寄存器中的值.

SysTick Current Value 寄存器(SYST_CVR)

寄存器	偏移	R/W	描述	复位值
SYST_CVR	SCS _BA+0x18	R/W	SysTick 当前值寄存器	0x00XX_XXXX

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

Bits	描述	
[31:24]	-	预留
[23:0]	CURRENT[23:0]	当前计数器的值. 这是被采样时的当前计数器的值.计数器不提供读-修改-写-保护功能.这个寄存器是写清0的. 软件写任何值到这个寄存器都将清这个寄存器为“0”. 不支持RAZ (看 SysTick加载值寄存器).

6.4 系统控制寄存器

Cortex-M0 状态和操作模式由系统控制寄存器控制. 包括CPUID, Cortex-M0 中断优先级和 Cortex-M0 电源管理都可以由系统控制寄存器控制.

细节请参考文档“ARM® Cortex™-M0 Technical Reference Manual” 和 “ARM® v6-M Architecture Reference Manual”.

6.4.1 系统控制寄存器内存映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移	R/W	描述	复位值
SCS_BA = 0xE000_E000				
CPUID	SCS_BA+0xD00	R	CPUID Base寄存器	0x410C_C200
ICSR	SCS_BA+0xD04	R/W	中断控制状态寄存器	0x0000_0000
AIRCR	SCS_BA+0xD0C	R/W	应用程序中断和复位控制寄存器	0xFA05_0000
SCR	SCS_BA+0xD10	R/W	系统控制寄存器	0x0000_0000
SHPR2	SCS_BA+0xD1C	R/W	系统处理优先级寄存器2	0x0000_0000
SHPR3	SCS_BA+0xD20	R/W	系统处理优先级寄存器3	0x0000_0000

6.4.2 系统控制寄存器描述

CPUID Base 寄存器(CPUID)

寄存器	偏移	R/W	描述	复位值
CPUID	SCS_BA+0xD00	R	CPUID Base 寄存器	0x410C_C200

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

Bits	描述	
[31:24]	**IMPLEMENTER [7:0]**	ARM分配的Implementer代码. (ARM = 0x41).
[23:20]	**-**	**预留**
[19:16]	**PART[3:0]**	Reads as 0xC for ARMv6-M parts.
[15:4]	**PARTNO[11:0]**	Reads as 0xC20.
[3:0]	**REVISION[3:0]**	Reads as 0x0.

中断控制装态寄存器(ICSR)

寄存器	偏移	R/W	描述	复位值
ICSR	SCS_BA+0xD04	R/W	中断控制装态寄存器	0x0000_0000

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

Bits	描述	
[31]	NMIPENDSET	**NMI set-pending bit.** 写: 0 = 没有作用 1 = 改变NMI 异常状态为未处理. 读: 0 = 没有未处理的NMI 异常 1 = NMI 异常正等待处理. 因为NMI异常有最高的优先级，正常情况下只要一写”1”到这个比特,处理器将马上进入NMI异常处理函数,然后这个比特会被自动清成”0”. 这就意味着,如果在NMI异常处理函数中读这个比特返回1,只有一种可能就是在NMI异常处理函数中NMI中断信号再次发出.
[30:29]	-	**预留.**
[28]	PENDSVSET	**PendSV set-pending bit.** 写: 0 =没有作用. 1 = 改变PendSV异常状态为未处理. 读: 0 = 没有未处理的PendSV异常. 1 = PendSV异常正等待处理. 写1 到这个比特是唯一将PendSV 异常状态变成未处理的方法.

Bits	描述	
[27]	**PENDSVCLR**	**PendSV clear-pending bit (只写)** 写: 0 =没有作用. 1 = 清除PendSV 异常未处理状态.
[26]	**PENDSTSET**	**SysTick exception set-pending bit** 写: 0 = 没有作用. 1 = 改变SysTick 异常状态为未处理. 读: 0 = 没有未处理的SysTick 异常. 1 = SysTick 异常正等待处理.
[25]	**PENDSTCLR**	**SysTick exception clear-pending bit (只写)** 写: 0 =没有作用 1 = 清除SysTick异常未处理状态.
[24]	**-**	**预留.**
[23]	**ISRPREEMPT**	**Interrupt Preemption (只读)** 如果设定, 从调试停止状态退出时，一个未处理的异常将被处理.
[22]	**ISRPENDING**	**中断未处理标志 (只读)** 除了 NMI 和 Faults异常. 0 = 没有中断等待处理. 1 = 有中断正等待处理.
[21]	**-**	**预留.**
[20:12]	**VECTPENDING [8:0]**	**Vector pending indicator (只读)** 指示等待处理的最高优先级的异常: 0 =没有异常等待处理. 非0 = 等待处理的最高优先级异常的异常号.
[11:9]	**-**	**预留.**
[8:0]	**VECTACTIVE [8:0]**	**Vector active indicator (只读)** 这个域包含正在处理的异常号: 0 = Thread mode. 非0 = 当前正在处理的异常号.

应用程序中断和复位控制寄存器(AIRCR)

寄存器	偏移	R/W	描述	复位值
AIRCR	SCS_BA+0xD0C	R/W	应用程序中断和复位控制寄存器	0xFA05_0000

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

Bits	描述	
[31:16]	VECTORKEY [15:0]	写这个寄存器时, 这个域必须是0x05FA, 否则写将导致无法预期的后果.
[15:3]	-	预留
[2]	SYSRESETREQ	写“1”到这个比特将导致对芯片发起一个复位信号. 这个比特是只写，复位自清除.
[1]	VECTCLRACTIVE	设定这个比特为“1” 将清除所有固定和可配置异常的激活状态. 这个比特是只写的，只有当核是停止状态时才可以写. 注意: 重新初始化堆栈是调试器的责任.
[0]	-	预留

系统控制寄存器(SCR)

寄存器	偏移	R/W	描述	复位值
SCR	SCS_BA+0xD10	R/W	系统控制寄存器	0x0000_0000

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

Bits	描述	
[31:5]	-	**预留**
[4]	SEVONPEND	**Send Event on Pending bit** 0 = 只有使能的中断或者事件能唤醒处理器,禁止的中断不能唤醒. 1 = 使能的事件和所有的中断,包括禁止的中断,都可以将处理器唤醒. 当事件或者中断进入未处理状态时,事件信号将处理器从WFE状态唤醒.如果处理器没有正在等待事件,事件将被注册并影响下一个WFE. 处理器也可以由SEV指令或者外部事件唤醒.
[3]	-	**预留**
[2]	SLEEPDEEP	**使能Deep sleep模式** 这个比特控制处理器使用sleep还是deep sleep作为它的低功耗模式: 0 = sleep. 1 = deep sleep.
[1]	SLEEPONEXIT	**Sleep-on-exit enable** 当从Handler模式返回到Thread模式时, 这个比特控制sleep-on-exit: 0 = 当返回到Thread模式时不要进入sleep. 1 = 当返回到Thread模式时进入sleep, 或者deep sleep. 设置这个比特为1 ,使能中断驱动的应用程序避免返回到一个空的主应用(main application).
[0]	-	**预留**

系统处理函数优先级寄存器2 (SHPR2)

寄存器	偏移	R/W	描述	复位值
SHPR2	SCS_BA+0xD1C	R/W	系统处理函数优先级寄存器2	0x0000_0000

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

Bits	描述	
[31:30]	**PRI_11[1:0]**	**系统处理函数 11 – SVCall的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:0]	-	**预留**

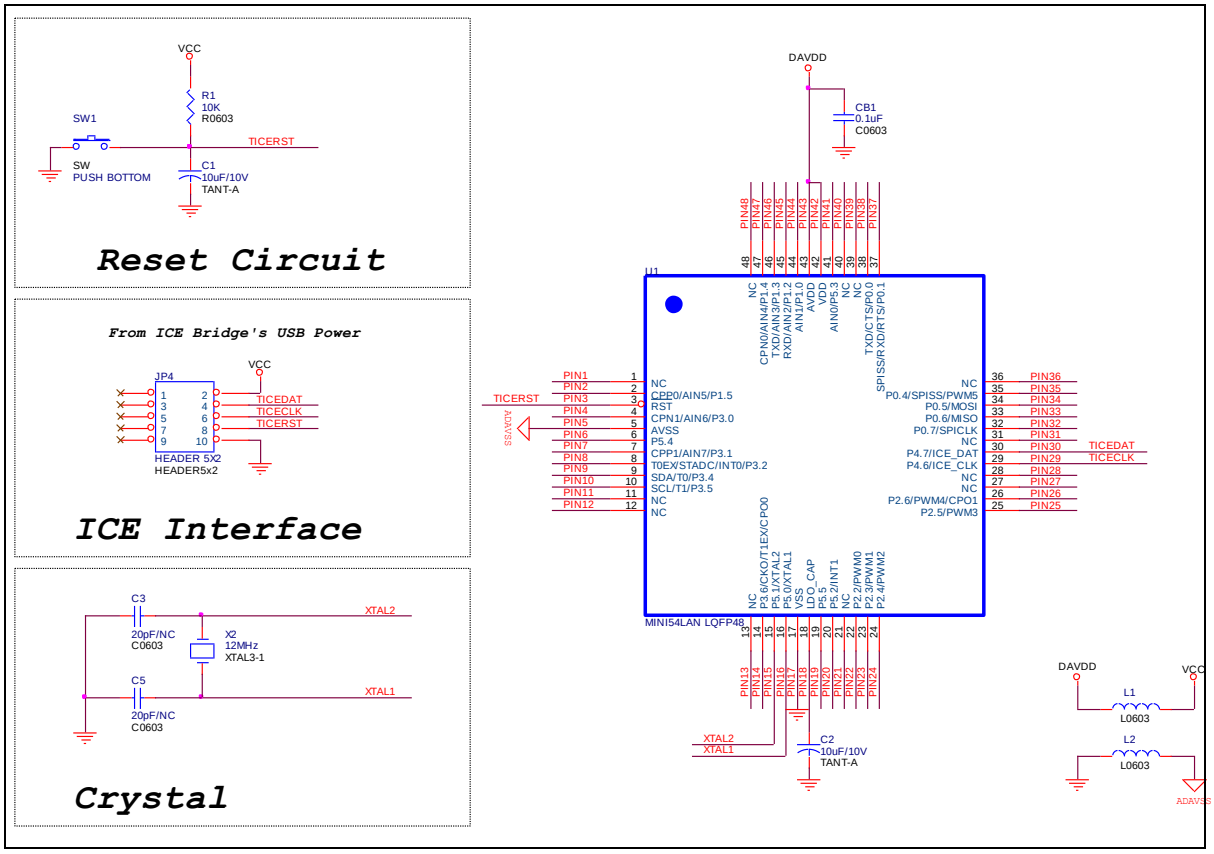
系统处理函数优先级寄存器3 (SHPR3)

寄存器	偏移	R/W	描述	复位值
SHPR3	SCS_BA+0xD20	R/W	系统处理函数优先级寄存器3	0x0000_0000

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

Bits	描述	
[31:30]	PRI_15[1:0]	**系统处理函数15 – SysTick的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[29:24]	-	预留
[23:22]	PRI_14[1:0]	**系统处理函数14 – PendSV的优先级** “0” 表示最高的优先级 & “3” 表示最低的优先级.
[21:0]	-	预留

7 应用电路


VCC
R1
10K
R0603
SW1
TICERST
SW
PUSH BOTTOM
C1
10uF/10V
TANT-A
Reset Circuit
From ICE Bridge's USB Power
JP4
VCC
TICEDAT
TICECLK
TICERST
HEADER 5X2
HEADER5x2
ICE Interface
C3
20pF/NC
C0603
X2
12MHz
XTAL3-1
XTAL2
C5
20pF/NC
C0603
XTAL1
Crystal
DAVDD
CB1
0.1uF
C0603
U1
MINI54LAN LQFP48
TICERST
TICEDAT
TICECLK
XTAL2
XTAL1
C2
10uF/10V
TANT-A
L1
L0603
L2
L0603
ADAVSS

8 电器特性

8.1 Absolute Maximum Ratings

SYMBOL	PARAMETER	MIN	MAX	UNIT
DC Power Supply	VDD–VSS	-0.3	+7.0	V
Input Voltage	VIN	VSS-0.3	VDD+0.3	V
Oscillator Frequency	$1/t_{CLCL}$	4	24	MHz
Operating Temperature	TA	-40	+85	°C
Storage Temperature	TST	-55	+150	°C
Maximum Current into VDD		-	120	mA
Maximum Current out of VSS			120	mA
Maximum Current sunk by a I/O pin			35	mA
Maximum Current sourced by a I/O pin			35	mA
Maximum Current sunk by total I/O pins			100	mA
Maximum Current sourced by total I/O pins			100	mA

Note: Exposure to conditions beyond those listed under absolute maximum ratings may adversely affects the lift and reliability of the device.

8.2 DC 电器特性

(VDD-VSS=5.0 V, TA = 25°C, FOSC = 24 MHz unless otherwise specified.)

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operation voltage	V_{DD}	2.5		5.5	V	V_{DD} =2.5 V ~ 5.5 V up to 24 MHz
V_{DD} rise rate to ensure internal operation correctly	V_{RISE}	0.05			V/mS	
Power Ground	V_{SS} AV_{SS}	-0.3			V	
LDO Output Voltage	V_{LDO}	-10%	1.8	+10%	V	V_{DD} = 2.5V ~ 5.5V
Analog Operating Voltage	AV_{DD}	0		V_{DD}	V	
Operating Current Normal Run Mode @ 24 MHz	I_{DD1}		9.5		mA	V_{DD} = 5.5V@24 MHz, enable all IP
	I_{DD2}		7.5		mA	V_{DD} = 5.5V@24 MHz, disable all IP
	I_{DD3}		7.5		mA	V_{DD} = 3.3V@24 MHz, enable all IP
	I_{DD4}		6		mA	V_{DD} = 3.3V@24 MHz, disable all IP
Operating Current Normal Run Mode @ 12 MHz	I_{DD5}		5.5		mA	V_{DD} = 5.5V@12 MHz, enable all IP
	I_{DD6}		4.5		mA	V_{DD} = 5.5V@12 MHz, disable all IP
	I_{DD7}		4		mA	V_{DD} = 3.3V@12 MHz, enable all IP
	I_{DD8}		3		mA	V_{DD} = 3.3V@12 MHz, disable all IP
Operating Current Normal Run Mode @ 4 MHz	I_{DD9}		3.6		mA	V_{DD} = 5.5V@4 MHz, enable all IP
	I_{DD10}		3.3		mA	V_{DD} = 5.5V@4 MHz, disable all IP
	I_{DD11}		1.7		mA	V_{DD} = 3.3V@4 MHz, enable all IP
	I_{DD12}		1.4		mA	V_{DD} = 3.3V@4 MHz, disable all IP
Operating Current Normal Run Mode @ 22.1184 MHz IRC	I_{DD13}		6.6		mA	V_{DD} = 5.5V@22.1184 MHz, enable all IP
	I_{DD14}		5		mA	V_{DD} = 5.5V@22.1184 MHz, disable all IP
	I_{DD15}		6.6		mA	V_{DD} = 3.3V@22.1184 MHz, enable all IP

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
	I_{DD16}		5		mA	V_{DD} = 3.3V@22.1184 MHz, disable all IP
Operating Current Normal Run Mode @ 32.768 KHz crystal oscillator	I_{DD17}		116		μA	V_{DD} = 5.5V@32.768 KHz, enable all IP
	I_{DD18}		113		μA	V_{DD} = 5.5V@32.768 KHz, disable all IP
	I_{DD19}		112		μA	V_{DD} = 3.3V@32.768 KHz, enable all IP
	I_{DD20}		100		μA	V_{DD} = 3.3V@32.768 KHz, disable all IP
Operating Current Normal Run Mode @ 10 KHz IRC	I_{DD21}		109		μA	V_{DD} = 5.5V@10 KHz, enable all IP
	I_{DD22}		108		μA	V_{DD} = 5.5V@10 KHz, disable all IP
	I_{DD23}		100		μA	V_{DD} = 3.3V@10 KHz, enable all IP
	I_{DD24}		98		μA	V_{DD} = 3.3V@10 KHz, disable all IP
Operating Current Idle Mode @ 24 MHz	I_{IDLE1}		5.5		mA	V_{DD} = 5.5V@24 MHz, enable all IP
	I_{IDLE2}		3.5		mA	V_{DD} = 5.5V@24 MHz, disable all IP
	I_{IDLE3}		3.8		mA	V_{DD} = 3.3V@24 MHz, enable all IP
	I_{IDLE4}		1.8		mA	V_{DD} = 3.3V@24 MHz, disable all IP
Operating Current Idle Mode @ 12 MHz	I_{IDLE5}		3.3		mA	V_{DD} = 5.5V@12 MHz, enable all IP
	I_{IDLE6}		2.6		mA	V_{DD} = 5.5V@12 MHz, disable all IP
	I_{IDLE7}		2		mA	V_{DD} = 3.3V@12 MHz, enable all IP
	I_{IDLE8}		1		mA	V_{DD} = 3.3V@12 MHz, disable all IP
Operating Current Idle Mode @ 4 MHz	I_{IDLE9}		3		mA	V_{DD} = 5.5V@4 MHz, enable all IP
	I_{IDLE10}		2.3		mA	V_{DD} = 5.5V@4 MHz, disable all IP
	I_{IDLE11}		1		mA	V_{DD} = 3.3V@4 MHz, enable all IP
	I_{IDLE12}		0.7		mA	V_{DD} = 3.3V@4 MHz, disable all IP
Operating Current Idle Mode @ 22.1184 MHz IRC	I_{IDLE13}		3.0		mA	V_{DD} = 5.5V@22.1184 MHz, enable all IP
	I_{IDLE14}		1.2		mA	V_{DD} = 5.5V@22.1184 MHz, disable all IP

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
	I_{IDLE15}		3.0		mA	V_{DD} = 3.3V@22.1184 MHz, enable all IP
	I_{IDLE16}		1.2		mA	V_{DD} = 3.3V@22.1184 MHz, disable all IP
Operating Current Idle Mode @ 32.768 KHz crystal oscillator	I_{IDLE17}		110		μA	V_{DD} = 5.5V@32.768 KHz, enable all IP
	I_{IDLE18}		107		μA	V_{DD} = 5.5V@32.768 KHz, disable all IP
	I_{IDLE19}		105		μA	V_{DD} = 3.3V@32.768 KHz, enable all IP
	I_{IDLE20}		102		μA	V_{DD} = 3.3V@32.768 KHz, disable all IP
Operating Current Idle Mode @ 10 KHz IRC	I_{IDLE21}		103		μA	V_{DD} = 5.5V@10 KHz, enable all IP
	I_{IDLE22}		102		μA	V_{DD} = 5.5V@10 KHz, disable all IP
	I_{IDLE23}		96		μA	V_{DD} = 3.3V@10 KHz, enable all IP
	I_{IDLE24}		95		μA	V_{DD} = 3.3V@10 KHz, disable all IP
Standby Current Power Down Mode	I_{PWD1}		10		μA	V_{DD} = 5.0V, CPU STOP All IP and Clock OFF
	I_{PWD2}		5		μA	V_{DD} = 3.3V, CPU STOP All IP and Clock OFF
Standby Current Power Down Mode with 32.768 KHz crystal enable	I_{PWD3}		12		μA	V_{DD} = 5.0V, CPU STOP All IP and Clock OFF except 32.768KHz crystal oscillator
	I_{PWD4}		7		μA	VDD = 3.3V, CPU STOP All IP and Clock OFF except 32.768KHz crystal oscillator
Input Current P0~P5 (Quasi-bidirectional mode)	I_{IN1}		-50	-60	μA	V_{DD} = 5.5 V, V_{IN} = 0 V or V_{IN}=V_{DD}
Input Current at /RESET[1]	I_{IN2}	-55	-45	-30	μA	V_{DD} = 3.3 V, V_{IN} = 0.45 V
Input Leakage Current PA, PB, PC, PD, PE	I_{LK}	-0.1	-	+0.1	μA	V_{DD} = 5.5 V, 0<V_{IN}<V_{DD}
Logic 1 to 0 Transition Current PA~PE (Quasi-bidirectional mode)	I_{TL} [3]	-650	-	-200	μA	V_{DD} = 5.5 V, V_{IN}<2.0 V

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Input Low Voltage P0~P5 (TTL input)	V_{IL1}	-0.3	-	0.8	V	V_{DD} = 4.5 V
		-0.3	-	0.6		V_{DD} = 2.5 V
Input High Voltage P0~P5 (TTL input)	V_{IH1}	2.0	-	V_{DD} +0.2	V	V_{DD} = 5.5 V
		1.5	-	V_{DD} +0.2		V_{DD} = 3.0 V
Input Low Voltage P0~P5, (Schmitt input)	V_{IL2}		0.4 V_{DD}		V	
Input High Voltage P0~P5, (Schmitt input)	V_{IH2}		0.6 V_{DD}		V	
Hysteresis voltage of P0~P5 (Schmitt input)	V_{HY}		0.2 V_{DD}		V	
Input Low Voltage XT1[*2]	V_{IL3}	0	-	0.8	V	V_{DD} = 4.5 V
		0	-	0.4		V_{DD} = 3.0 V
Input High Voltage XT1[*2]	V_{IH3}	3.5	-	V_{DD} +0.2	V	V_{DD} = 5.5 V
		2.4	-	V_{DD} +0.2		V_{DD} = 3.0 V
Internal /RESET pin pull up resistor	R_{RST}	40	-	100	KΩ	
Negative going threshold (Schmitt input), /RESET	V_{ILS}	-0.5	-	0.3 V_{DD}	V	
Positive going threshold (Schmitt input), /RESET	V_{IHS}	0.6 V_{DD}	-	V_{DD}+0.5	V	
Source Current P0~P5. (Quasi-bidirectional Mode)	I_{SR11}	-300	-370	-450	μA	V_{DD} = 4.5 V, V_S = 2.4 V
	I_{SR12}	-50	-70	-90	μA	V_{DD} = 2.7 V, V_S = 2.2 V
	I_{SR12}	-40	-60	-80	μA	V_{DD} = 2.5 V, V_S = 2.0 V
Source Current P0~P5, (Push-pull Mode)	I_{SR21}	-20	-24	-28	mA	V_{DD} = 4.5 V, V_S = 2.4 V
	I_{SR22}	-4	-6	-8	mA	V_{DD} = 2.7 V, V_S = 2.2 V
	I_{SR22}	-3	-5	-7	mA	V_{DD} = 2.5 V, V_S = 2.0 V
Sink Current P0~P5, (Quasi-bidirectional and Push-pull Mode)	I_{SK1}	10	16	20	mA	V_{DD} = 4.5 V, V_S = 0.45 V
	I_{SK1}	7	10	13	mA	V_{DD} = 2.7 V, V_S = 0.45 V
	I_{SK1}	6	9	12	mA	V_{DD} = 2.5 V, V_S = 0.45 V

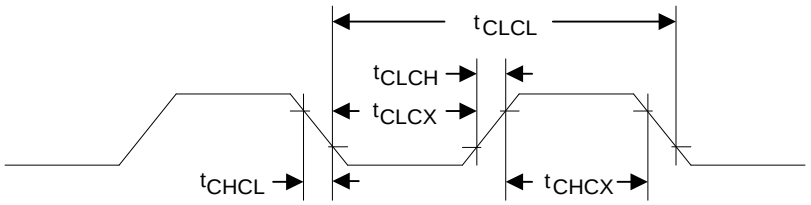
Note:

1. /RESET pin is a Schmitt trigger input.
2. Crystal Input is a CMOS input.
3. Pins of P0~P5 can source a transition current when they are being externally driven from 1 to 0. In the condition of V_{DD}=5.5 V, the transition current reaches its maximum value when V_{IN} approximates to 2 V.

8.3 AC 电器特性

8.3.1 External Input Clock

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
PARAMETER	tCHCX	20			nS	
Clock High Time	tCLCX	20			nS	
Clock Low Time	tCLCH			10	nS	
Clock Rise Time	tCHCL			10	nS	



Note: Duty cycle is 50%.

8.3.2 External 4~24 MHz XTAL Oscillator

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Oscillator frequency	f_{HXTAL}	4	12	24	MHz	VDD = 2.5V ~ 5.5V
Temperature	T_{HXTAL}	-40		+85	°C	
Operating current	I_{HXTAL}		TBD		mA	VDD = 5.0V

8.3.3 Typical Crystal Application Circuits

CRYSTAL	C1	C2
4 MHz ~ 24 MHz	Optional (Depend on crystal specification)	

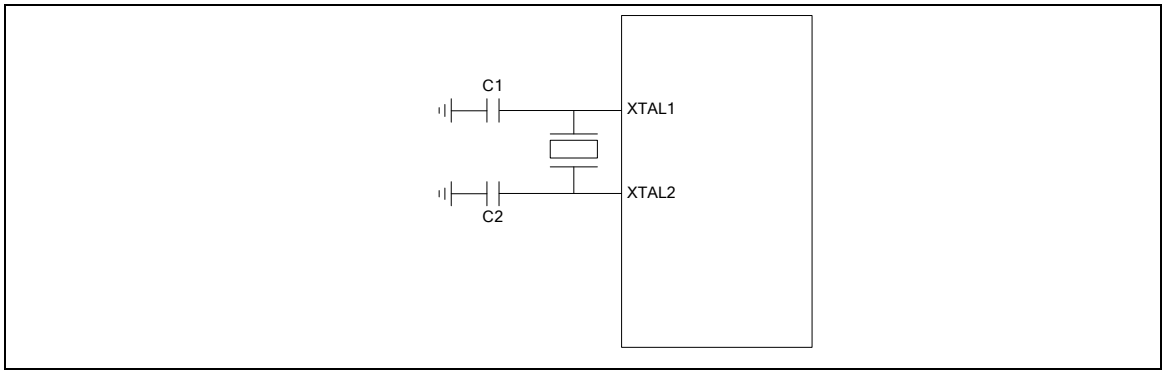


图 8.3-1 Typical Crystal Application Circuit

8.3.4 External 32.768 KHz XTAL Oscillator

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Oscillator frequency	fL_{XTAL}		32.768		KHz	VDD = 2.5V ~ 5.5V
Temperature	TL_{XTAL}	-40		+85	^{o}C	
Operating current	I_{HXTAL}		TBD		μA	VDD = 5.0V

8.3.5 Internal 22.1184 MHz RC Oscillator

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Supply voltage[1]	V_{HRC}		1.8		V	
Center Frequency	F_{HRC}	21.89	22.1184	22.34	MHz	25^{o}C, VDD = 5V
		20.57	22.1184	23.23	MHz	-40^{o}C~+85 ^{o}C, VDD = 2.5V~5.5V
		21.78	22.0	22.22	MHz	-40^{o}C~+85 ^{o}C, VDD = 2.5V~5.5V Enable 32.768K crystal oscillator and set TRIM_SEL = 1
Operating current	I_{HRC}		TBD		mA	

Note: Internal operation voltage comes from LDO

8.3.6 Internal 10 KHz RC Oscillator

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Supply voltage[1]	V_{LRC}		1.8		V	
Center Frequency	F_{LRC}	7	10	13	KHz	25°C, VDD = 5V
		5	10	15	KHz	-40°C~+85 °C, VDD = 2.5V~5.5V
Operating current	I_{LRC}		TBD		μA	VDD = 5V

Note: Internal operation voltage comes from LDO

.

8.4 模拟特性

8.4.1 Specification of Brown-Out Reset (BOD)

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operating voltage	V_{BOD}	2.0		5.5	V	
Operating current	I_{BOD}		5	15	μA	VDD = 5V Enable BOD27 and BOD38
BOD38 detection level	V_{B38dt}	3.6	3.8	4.0	V	25°C
BOD27 detection level	V_{B27dt}	2.6	2.7	2.8	V	25°C

8.4.2 Specification of Low Voltage Reset (LVR)

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operating voltage	V_{BOD}	2.0		5.5	V	
Operating current	I_{BOD}		1	2	μA	
Detection level	V_{LVR}		2.0		V	25°C
LVR always enable		1.6	2.0	2.4	V	-40°C ~ +85°C

8.4.3 Specification of Analog Comparator

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operating voltage	V_{BOD}	2.5	3.3	5.5	V	
Operating current	I_{CMP}		40	80	μA	
Input offset voltage	V_{OFFSET}		10	20	mV	
Output swing voltage	V_{swin}	0.1		V_{DD}-0.1	V	
Input common mode range (VCM)	V_{CM}	0.1		V_{DD}-0.1	V	
DC gain	G_{DC}		70		dB	
Propagation delay	T_{PDLY}		200		ns	VCM = 1.2V The difference voltage in CPPx and CPNx is 0.1V
Hysteresis	V_{HYS}		±10		mV	One bit control W/O & W. hysteresis @V_{CM}=0.2V ~ VDD-0.1V

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Stable time	T_{STBL}			2	μS	CPPx = 1.3V and CPNX = 1.2V

8.4.4 Analog Comparator Reference Voltage (CRV)

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operating voltage	V_{BOD}	2.5		5.5	V	
CRV step size	V_{STEP}		V_{DD}/24		V	VDD = 5V Enable BOD27 and BOD38
CRV output voltage absolute accuracy	A_{CRV}	-5		+5	%	
Unit resistor value	R_{CRV}		2K		ohm	

8.4.5 Specification of 10-bit ADC

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operating voltage	AV_{DD}	2.7		5.5	V	$AV_{DD} = V_{DD}$
Operating current	I_{ADC}			1	mA	$AV_{DD} = V_{DD}$ = 5V, F_{SPS} = 150K
Resolution	R_{ADC}			10	Bit	
Reference voltage	V_{REF}		A_{VDD}		V	V_{REF} connect to A_{VDD} in chip
ADC input voltage	V_{IN}	0		V_{REF}	V	
Conversion time	T_{CONV}	6.7			μS	
Sampling Rate	F_{SPS}	150K			Hz	V_{DD} = 5V, ADC clock = 6MHz Free running conversion
Integral Non-Linearity Error (INL)	INL			±1	LSB	
Differential Non-Linearity (DNL)	DNL			±1	LSB	
Gain error	E_G			±2	LSB	
Offset error	E_{OFFSET}			3	LSB	
Absolute error	E_{ABS}			4	LSB	
ADC Clock frequency	F_{ADC}	5K		6M	Hz	V_{DD} = 5V

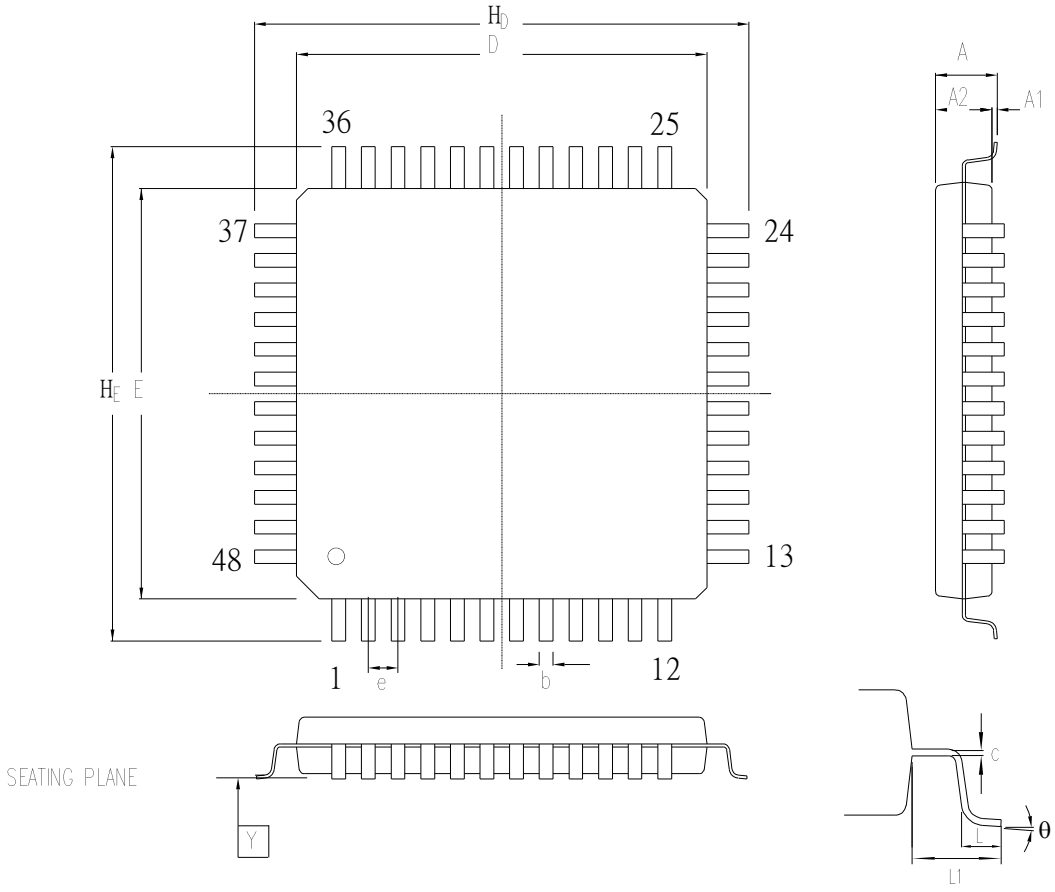
PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Clock cycle	AD_{CYC}	38			Cycle	
Bang-gap voltage	V_{BG}	1.27	1.35	1.44	V	

8.4.6 Flash Memory Characteristics

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Cycling (erase / write) Program memory	N_{CYC}	100			K cycle	
Data retention	T_{RET}	10			years	T_A = +85°C
Erase time of ISP mode	T_{ERASE}	2.3	2.5	2.7	mS	Erase time for one page
Program time of ISP mode	T_{PROG}	57	62	67	μS	Programming time for one word
Program current	I_{PROG}		3.3		mA	V_{DD} = 5.5V

9 PACKAGE DIMENSION

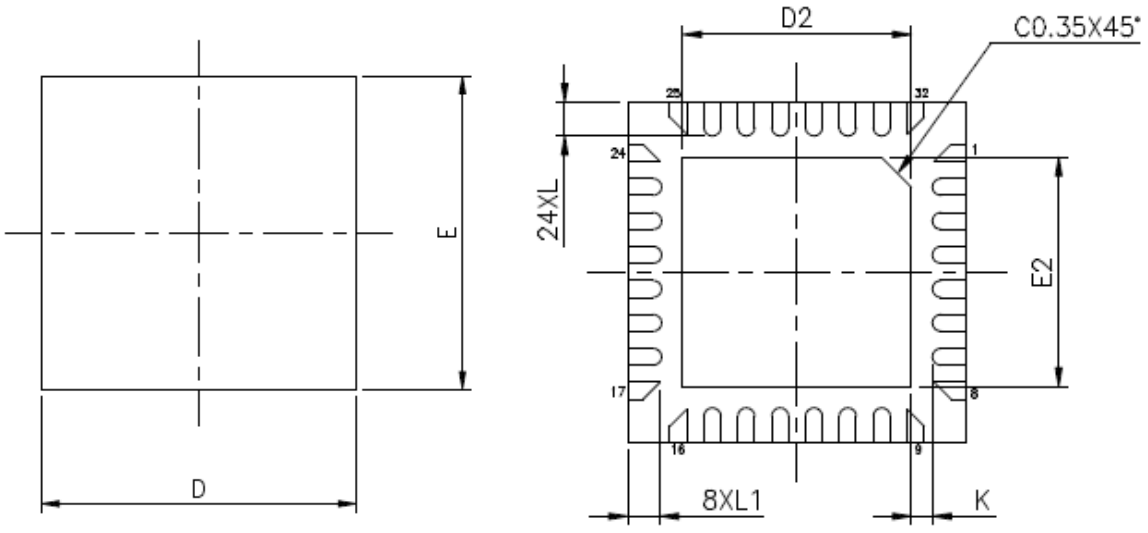
9.1 48-Pin LQFP

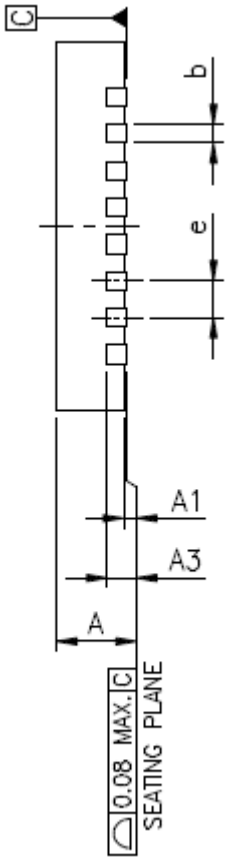


Controlling dimension : Millimeters

Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	—	—	—	—
A_1	0.002	0.004	0.006	0.05	0.10	0.15
A_2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.006	0.008	0.010	0.15	0.20	0.25
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.272	0.276	0.280	6.90	7.00	7.10
E	0.272	0.276	0.280	6.90	7.00	7.10
e	0.014	0.020	0.026	0.35	0.50	0.65
H_D	0.350	0.354	0.358	8.90	9.00	9.10
H_E	0.350	0.354	0.358	8.90	9.00	9.10
L	0.018	0.024	0.030	0.45	0.60	0.75
L_1	—	0.039	—	—	1.00	—
Y	—	—	0.004	—	—	0.10
θ	0	—	7	0	—	7

9.2 33-Pin QFN (4mm X 4mm)

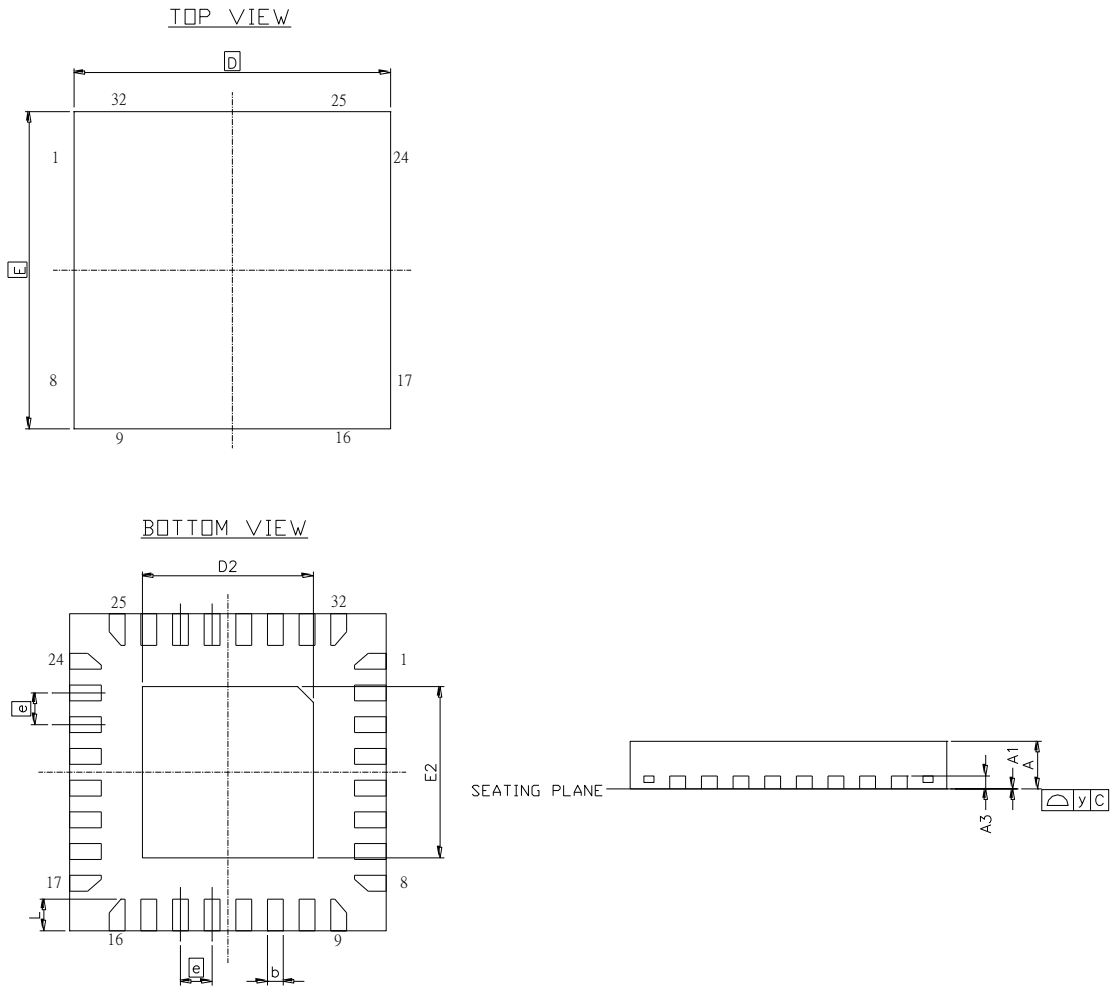




DIMENSION	MILLIMETERS		
SYMBOLS	MIN.	NOM.	MAX.
A	0.80	0.85	0.90
A1	0.00	0.02	0.05
A3	0.20 REF.		
b	0.15	0.20	0.25
D	4.00 BSC		
E	4.00 BSC		
e	0.40 BSC		
L	0.35	0.40	0.45
L1	0.332	0.382	0.432
K	0.20	—	—

	E2			D2		
PAD SIZE	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
114X114 MIL	2.60	2.70	2.75	2.60	2.70	2.75

9.3 33-Pin QFN (5mm X 5mm)



SYMBOL	DIMENSION (MM)			DIMENSION (INCH)		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	0.70	0.75	0.80	0.0275	0.0295	0.0315
A1	0	0.02	0.05	0	0.001	0.002
A3	0.20 REF			0.008 REF		
b	0.18	0.25	0.30	0.007	0.010	0.012
D	5.00 BSC			0.197 BSC		
E	5.00 BSC			0.197 BSC		
e	0.50 BSC			0.0197 BSC		
L	0.30	0.40	0.50	0.012	0.016	0.020
y	0.10			0.0039		

SYMBOL	D2			E2		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
DIMENSION (MM)	3.4	3.5	3.6	3.4	3.5	3.6

10 修订历史

日期	修订	修改
Dec 1, 2011	V1.02	Initial release of Chinese version.
Feb 1, 2012	V1.03	1. Add VDD rise rate specification. 2. Revise minimum ADC clock frequency specification. 3. Revise minimum and maximum specification of band-gap voltage. 4. Revise minimum and maximum specification of external input clock. 5. Add flash memory electrical characteristics. 6. Fix typo of figure 9.3-1 band-gap voltage. 7. Only even channels (PWM0, PWM2 and PWM4) can be set inverter bit (CHnINV, n=0,2,4) in independent mode.. 8. Modify the graphic in LEV_RTS (bit 9 of UA_MCR register) description. 9. Add CFGUEN and LDUEN bit description in ISPCON register

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.