ARM Cortex®-M0

32位微控制器

NuMicro™ 家族
NUC123 系列
技术参考手册

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

www.nuvoton.com

目录

图

表

1 概述

NuMicro™ NUC123 系列是 32位的内嵌 Corte®-M0 内核的微控制器，最高可运行至72 MHz, 内建36K/68K字节的Flash存储器，以及12K/20K字节 SRAM, 4K字节用于存储ISP引导代码的ROM。另外还有丰富的外设，如定时器，看门狗，Windowed Watchdog Timer, 带CRC计算功能的PDMA，UART, SPI/MICROWIRE, I^2C, I^2S, PWM Timer, GPIO, PS/2, USB 2.0全速设备, 10-bit ADC, 低电压复位控制器和欠压监测功能。

Product Line	UART	SPI	I^2C	USB	LIN	CAN	PS/2	I^2S
NUC123	•	•	•	•	-	-	•	•

表 1-1 支持的连接接口列表

2 特性

2.1 NuMicro™ NUC123 特性

- 内核
  - ARM® Cortex®-M0内核最高运行至72 MHz
  - 一个 24-位系统定时器
  - 低功耗掉电模式
  - 单指令周期32位硬件乘法器
  - 嵌套向量中断控制器NVIC 用于控制32个中断源，每个中断源可设置4个优先级
  - 支持串行线调试（SWD）带2个观察点/4个断点
- 内建LDO, 宽电压工作范围为2.5V 到 5.5V
- Flash 存储器
  - 36K/68K字节FLASH用于存储程序代码
  - 4KB flash用于存储ISP引导代码
  - 支持在系统编程 (ISP)方式更新应用程序
  - Flash为512 字节页擦除
  - 36KB 和 68KB 系统的数据Flash的地址和大小都可以配置
  - 通过SWD/ICE接口，支持2线 ICP升级方式
  - 支持外部编程器并行高速编程模式
- SRAM 存储器
  - 12K/20K 字节内建SRAM
  - 支持 PDMA 模式
- PDMA (Peripheral DMA)
  - 支持6个PDMA通道用于SRAM和周边设备例如：SPI, UART, I^2S, PWM 和 ADC之间的自动数据传输
  - 支持 CRC 计算功能，4种常见多项式可选：CRC-CCITT, CRC-8, CRC-16 和 CRC-32
- 时钟控制
  - 针对不同应用可灵活选择时钟
  - 内部 22.1184 MHz (调整到 1%)高速振荡器，可用于系统运行；低功耗内部 10 KHz 低速振荡器用于看门狗及掉电模式唤醒等功能
  - 支持一组PLL, 高至 144MHz, 用于高性能的系统运行
  - 外部 4~24 MHz 高速晶振输入用于精准的时序操作
- GPIO
  - 四种I/O模式:
    - 准双向模式
    - 推挽输出模式
    - 开漏输出模式
    - 高阻输入模式
  - TTL/Schmitt触发输入可选
  - I/O管脚可被配置为边沿/电平触发模式的中断源
  - 支持大电流驱动/灌入I/O模式
- Timer
  - 支持4组32位定时器，每个定时器包括一个24位向上计数定时器和一个8位预分频计数器

- 每个定时器有独立的时钟源
- 提供 one-shot, periodic, toggle 和Continuous Counting操作模式
- 支持事件计数功能

- Watchdog/Windowed-Watchdog Timer
  - 多个时钟源可选
  - 从1.6 ms ~ 26.0 sec有8个可选的超时周期(取决于所选的时钟源)
  - WDT可用作掉电模式的唤醒
  - 看门狗定时溢出可选择中断或者复位
  - windowed-watchdog timer 超时将发生中断
  - windowed-watchdog timer 超时可以复位或者重新加载一个unexpected time window
- PWM/Capture
  - 内建2组16位PWM发生器,可输出4路PWM或2组互补的PWM对
  - 每组PWM发生器配有一个时钟源选择器，一个时钟分频器，一个8位时钟预分频和一个用于互补配对PWM的死区发生器
  - 4路16位捕捉定时器(与PWM定时器共享)提供4路输入信号的上升/下降沿的捕捉功能
  - 支持捕捉(Capture)中断
- UART
  - 最多2组UART控制器
  - 支持流控(TXD, RXD, CTS and RTS)
  - UART0/1带 16-字节FIFO
  - 支持IrDA(SIR)功能
  - 支持 RS-485 9 位模式和方向控制.
  - 可编程波特率发生器频率高至1/16系统时钟
  - 支持PDMA模式
- SPI
  - 最高支持3组SPI控制器
  - 主机速率高达 32 Mbps ， 从机高达16Mbps(芯片工作在 3.3V)
  - 支持 SPI 主机/从机模式
  - 全双工同步串行数据传输
  - 传输数据长度从8 位 - 32位可变
  - 可设置MSB 或LSB 优先的传输模式
  - 当作为主机时有2条从机片选线，作为从机时有1条从机片选线
  - 支持16/24/32位传输模式下的字节睡眠模式
  - 支持 PDMA 模式
- I^2C
  - 最多支持2组I^2C 设备
  - 主/从模式
  - 主从机之间双向数据传输
  - 多主机总线支持 (无中心主机)
  - 多主机同时传输数据时仲裁，避免总线上串行数据损坏
  - 总线采用串行同步时钟,可实现设备之间以不同的速率传输
  - 串行同步时钟可作为握手方式控制总线上数据暂停及恢复传送
  - 可编程的时钟适用于不同速率控制
  - I^2C总线上支持多地址识别 (4个从机地址带mask选项)
  - 支持地址识别唤醒（只有第一个从地址）

- I^2S
  - 外部音频CODEC 接口
  - 可作主机也可作从机模式
  - 能处理8, 16,24 和 32 位word
  - 支持单声道和立体声的音频数据
  - 支持I^2S 和MSB justified数据格式
  - 提供两组8个字的FIFO数据缓存，一组用于发送，一组用于接收
  - 缓冲区超过可编程边界时，产生中断请求
  - 支持两组DMA请求，一组用于发送，另一组用于接收
- PS/2 设备控制器
  - 禁止主机通信和请求发送检测
  - 接收帧错误检测
  - 可编程的1到16字节的发送缓冲以减少CPU的负担
  - 数据接收的双缓冲
  - 软件可控 总线
- USB 2.0 全速设备
  - 一组USB 2.0 全速设备 12Mbps
  - 片上 USB 收发器
  - 提供一个中断源，4个中断事件
  - 支持控制、块、中断和等式传输
  - 总线上没有信号超过3ms，自动挂起功能
  - 提供8个可编程的端点
  - 内部有512 字节的内部SRAM 作为USB 缓存
  - 提供远程唤醒功能
- ADC
  - 10位SAR型ADC，转换速率达150K SPS
  - 最多8通道单端模式输入
  - 单一扫描模式/单周期扫描模式/连续扫描模式
  - 每个通道有独立的转换结果寄存器
  - 扫描使能的通道
  - 阈电压侦测
  - 软件编程或外部管脚触发开始转换
  - 支持 PDMA 模式
- 欠压检测(Brown-out detector)
  - 支持四级检测电压: 4.4V/3.7V/2.7V/2.2V
  - 支持欠压中断和复位选择
- 低压复位(Low Voltage Reset)
  - 阈电压: 2.0V
- 内建LDO
- 工作温度: -40℃~85℃

- 封装:
  - 无铅封装 (RoHS)
  - LQFP 64-pin
  - LQFP 48-pin
  - QFN 33-pin

3 料号列表及管脚配置

3.1 NuMicro™ NUC123xxxANx 产品选型指南

3.1.1 NuMicro™ NUC123 选型指南

Part number	Flash	SRAM	ISP ROM	I/O	Timer	Connectivity						I²S	Comp.	PWM	ADC	RTC	EBI	ISP ICP IAP	Package
						UART	SPI	I²C	USB	LIN	PS/2								
NUC123ZD4AN0	68 KB	20 KB	4 KB	Up to 20	4x32-bit	1	3	1	1	-	-	1	-	2	-	-	-	v	QFN33
NUC123ZC2AN1	36 KB	12 KB	4 KB	up to 20	4x32-bit	1	3	1	1	-	-	1	-	2	-	-	-	v	QFN33
NUC123LD4AN0	68 KB	20 KB	4 KB	up to 36	4x32-bit	2	3	2	1	-	1	1	-	4	8x10-bit	-	-	v	LQFP48
NUC123LC2AN1	36 KB	12 KB	4 KB	up to 36	4x32-bit	2	3	2	1	-	1	1	-	4	8x10-bit	-	-	v	LQFP48
NUC123SD4AN0	68 KB	20 KB	4 KB	up to 47	4x32-bit	2	3	2	1	-	1	1	-	4	8x10-bit	-	-	v	LQFP64
NUC123SC2AN1	36 KB	12 KB	4 KB	up to 47	4x32-bit	2	3	2	1	-	1	1	-	4	8x10-bit	-	-	v	LQFP64

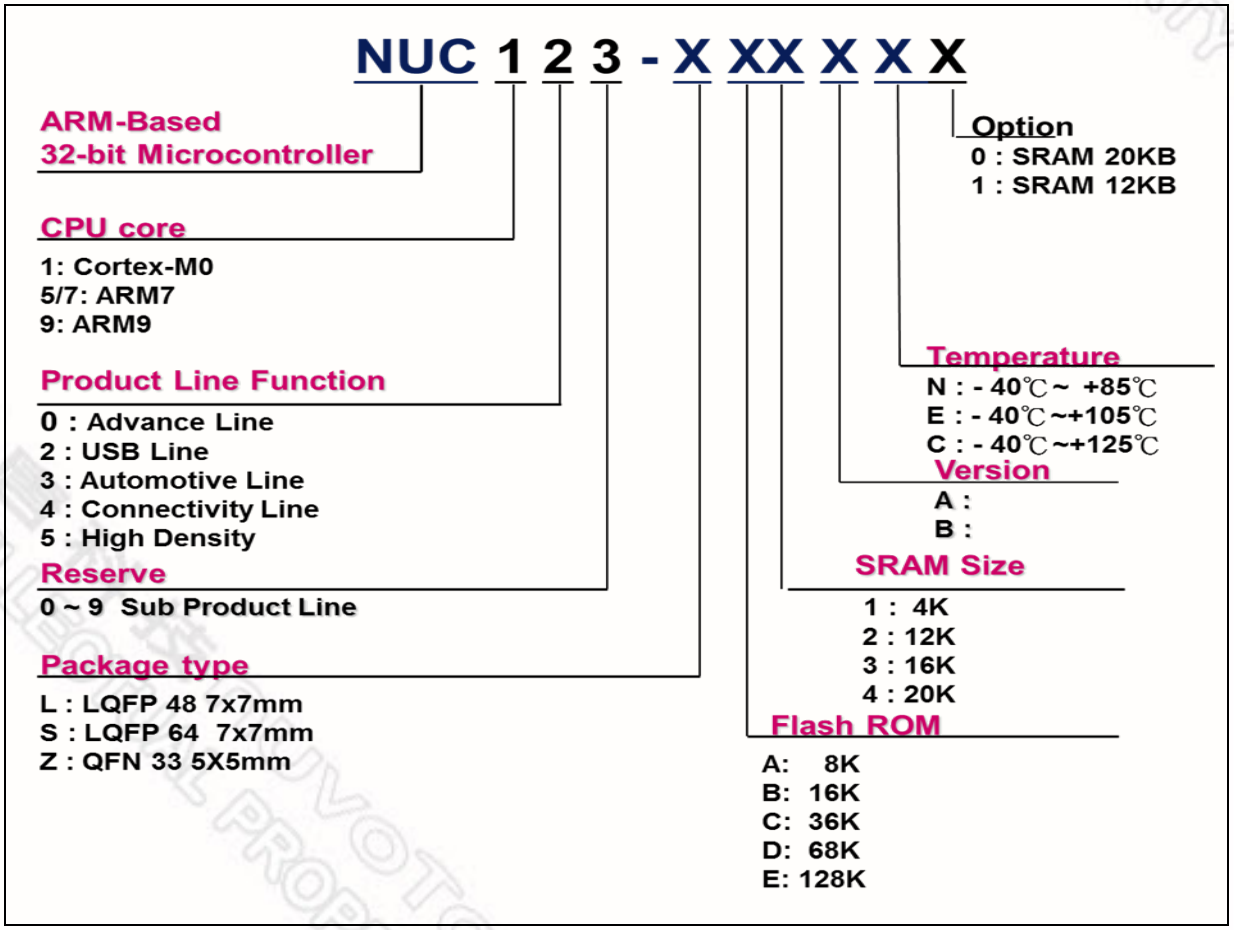


图 3-1 NuMicro™ NUC123 系列编号规则

3.2 管脚配置

3.2.1 NuMicro™ NUC123 管脚图

NuMicro™ NUC123SxxANx LQFP 64 pin

3.2.1.1

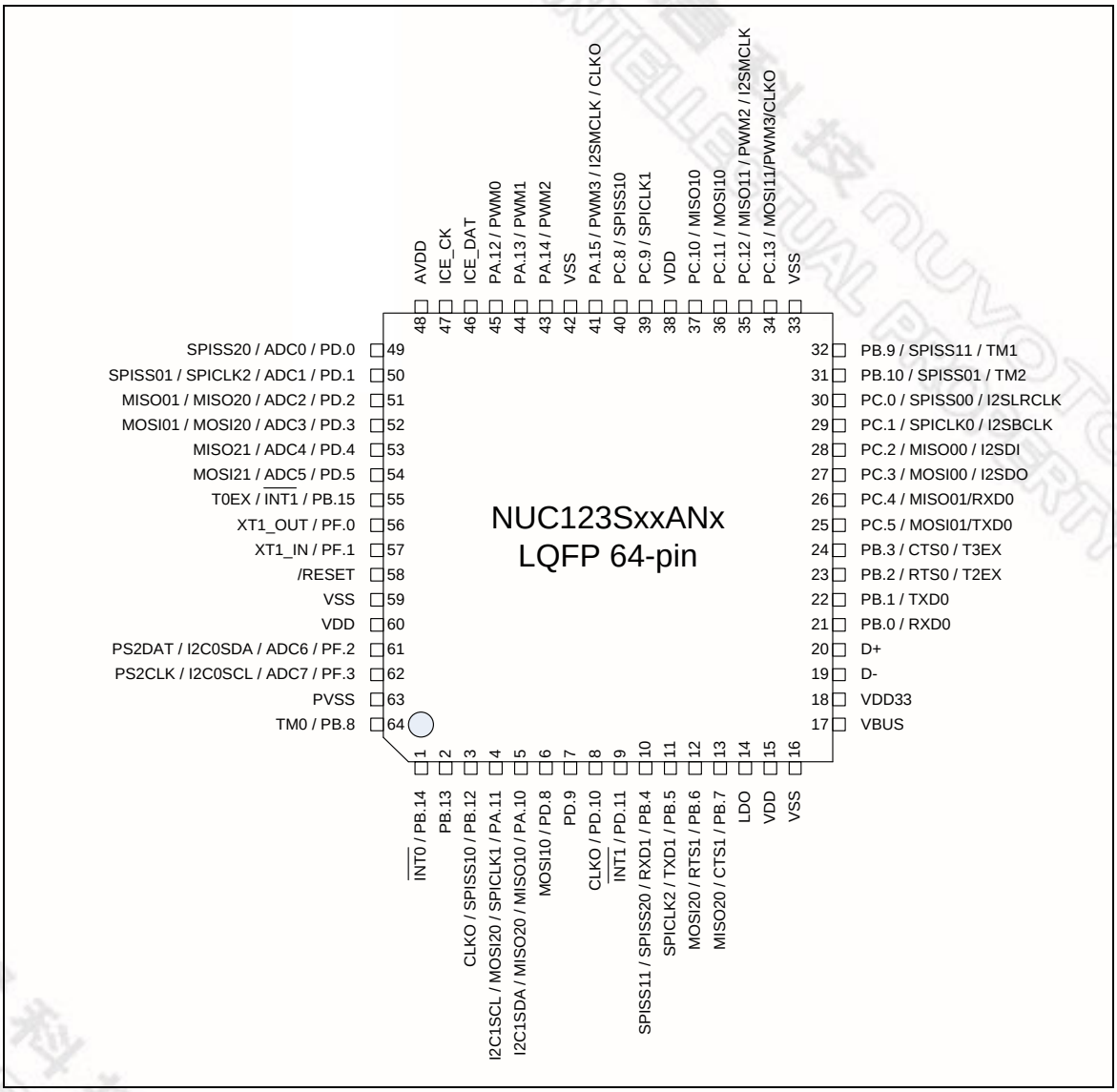


图 3-2 NuMicro™ NUC123SxxANx LQFP 64 管脚图

NuMicro™ NUC123LxxANx LQFP 48 pin

3.2.1.2

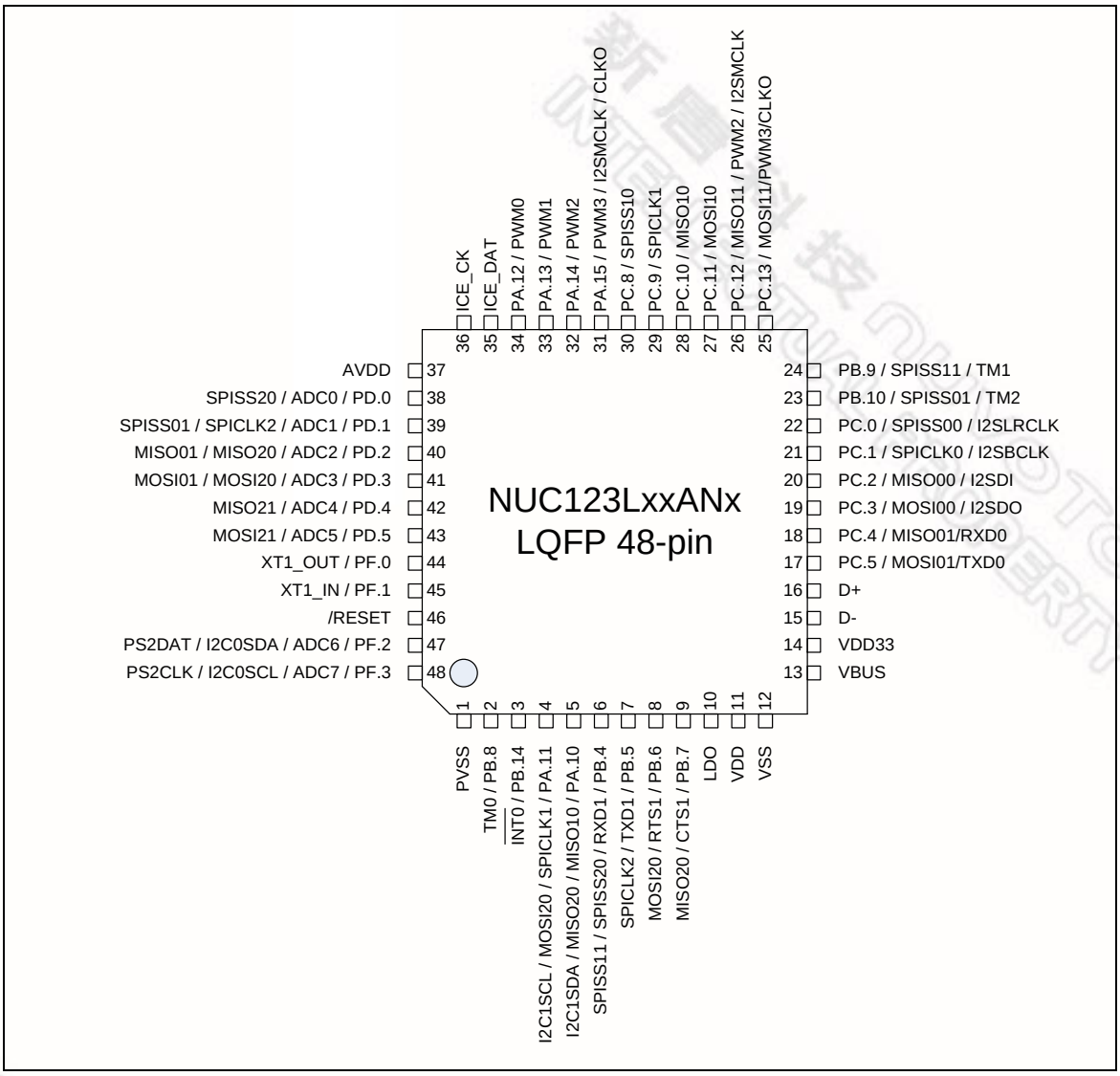


图 3-3 NuMicro™ NUC123LxxANx LQFP 48-pin 管脚图

3.2.1.3 NuMicro™ NUC123ZxxANx QFN 33 pin

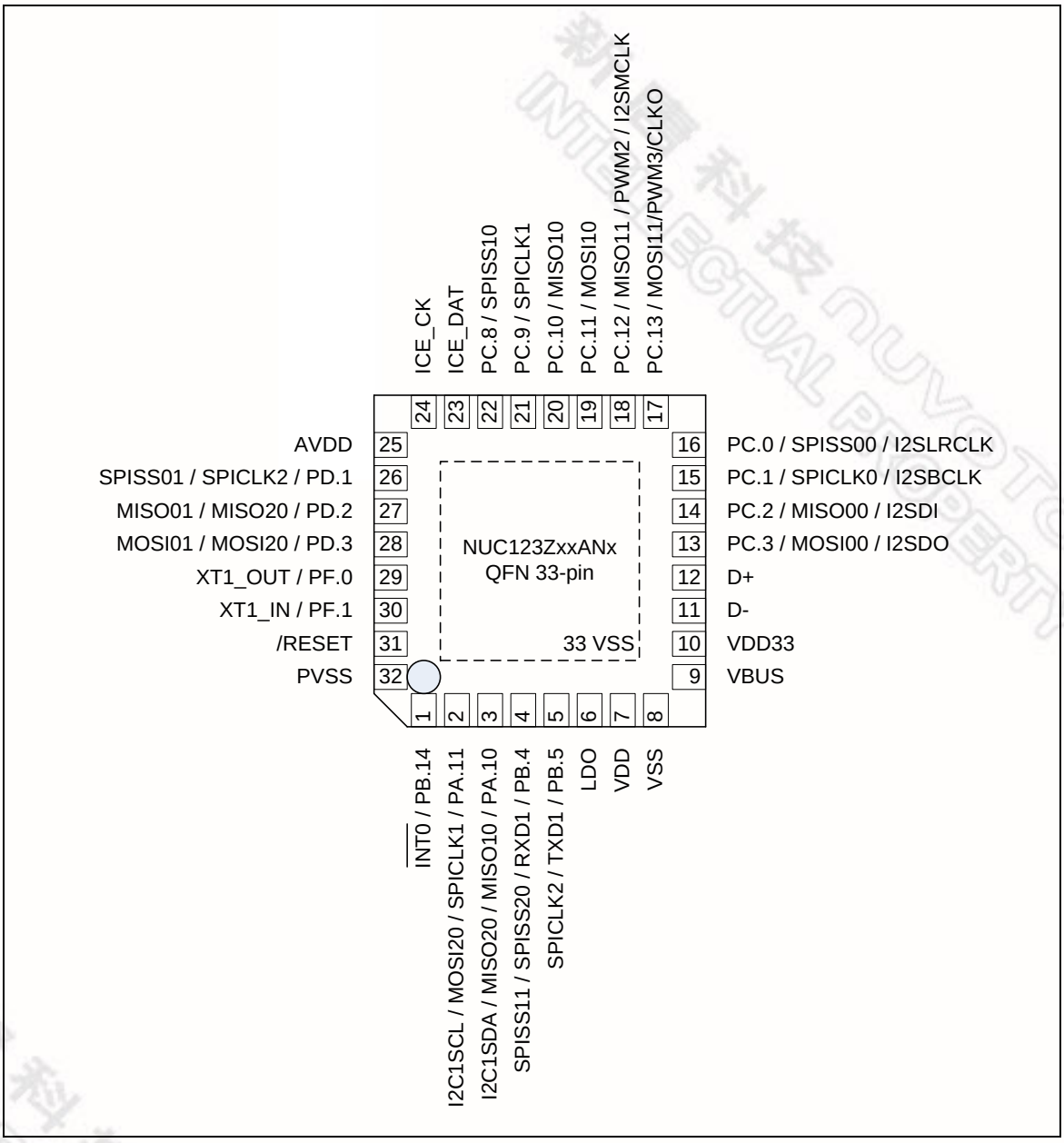


图 3-4 NuMicro™ NUC123ZxxANx QFN 33-pin 管脚图

3.2.2 管脚功能描述

3.2.3 NuMicro™ NUC123 管脚描述

Pin No			Pin Name	Type	Description
LQFP 64-pin	LQFP 48-pin	QFN 33-pin			
1	3	1	PB.14	**I/O**	Digital GPIO pin
			/INT0	**I**	External interrupt 0 input pin
2			PB.13	**I/O**	Digital GPIO pin
3			PB.12	**I/O**	Digital GPIO pin
			SPISS10	**I/O**	SPI1 1st slave select pin
			CLKO	**O**	Frequency Divider output pin
4	4	2	PA.11	**I/O**	Digital GPIO pin
			SPICLK1	**I/O**	SPI1 serial clock pin
			MOSI20	**I/O**	SPI2 1st MOSI (Master Out, Slave In) pin
			I2C1SCL	**I/O**	I^2C1 clock pin
5*	5*	3*	PA.10	**I/O**	Digital GPIO pin
			MISO10	**I/O**	SPI1 1st MISO (Master In, Slave Out) pin
			MISO20	**I/O**	SPI2 1st MISO (Master In, Slave Out) pin
			I2C1SDA	**I/O**	I^2C1 data input/output pin
6			PD.8	**I/O**	Digital GPIO pin
			MOSI10	**I/O**	SPI1 1st MOSI (Master Out, Slave In) pin
7			PD.9	**I/O**	Digital GPIO pin
8			PD.10	**I/O**	Digital GPIO pin
			CLKO	**O**	Frequency Divider output pin
9			PD.11	**I/O**	Digital GPIO pin
			/INT1	**I**	External interrupt 1 input pin
10	6	4	PB.4	**I/O**	Digital GPIO pin
			RXD1	**I**	UART1 data receiver input pin
			SPISS20	**I/O**	SPI2 1st slave select pin
			SPISS11	**I/O**	SPI1 2nd slave select pin
11	7	5	PB.5	**I/O**	Digital GPIO pin

			TXD1	**O**	UART1 data transmitter output pin
			SPICLK2	**I/O**	SPI2 serial clock pin
12	8		PB.6	**I/O**	Digital GPIO pin
			RTS1	**O**	UART1 request to send output pin
			MOSI20	**I/O**	SPI2 1st MOSI (Master Out, Slave In) pin
13	9		PB.7	**I/O**	Digital GPIO pin
			CTS1	**I**	UART1 clear to send input pin
			MISO20	**I/O**	SPI2 1st MISO (Master In, Slave Out) pin
14	10	6	LDO	**P**	LDO output pin
15	11	7	VDD	**P**	Power supply for I/O ports and LDO source for internal PLL and digital function. Voltage range is 2.5V ~ 5V.
16	12	8	VSS	**P**	Ground
17	13	9	VBUS	**USB**	Power supply from USB host or hub
18	14	10	VDD33	**USB**	Internal power regulator output 3.3V decoupling pin
19	15	11	D-	**USB**	USB differential signal D-
20	16	12	D+	**USB**	USB differential signal D+
21			PB.0	**I/O**	Digital GPIO pin
			RXD0	**I**	UART0 data receiver input pin
22			PB.1	**I/O**	Digital GPIO pin
			TXD0	**O**	UART0 data transmitter output pin
23			PB.2	**I/O**	Digital GPIO pin
			RTS0	**O**	UART0 request to send output pin
			T2EX	**I**	Timer2 external capture input pin
24			PB.3	**I/O**	Digital GPIO pin
			CTS0	**I**	UART0 clear to send input pin
			T3EX	**I**	Timer3 external capture input pin
25	17		PC.5	**I/O**	Digital GPIO pin
			MOSI01	**I/O**	SPI0 2nd MOSI (Master Out, Slave In) pin
			TXD0	**O**	UART0 data transmitter output pin
26	18		PC.4	**I/O**	Digital GPIO pin
			MISO01	**I/O**	SPI0 2nd MISO (Master In, Slave Out) pin
			RXD0	**I**	UART0 data receiver input pin

27	19	13	PC.3	**I/O**	Digital GPIO pin
			MOSI00	**I/O**	SPI0 1st MOSI (Master Out, Slave In) pin
			I2SDO	**O**	I2S data output pin
28	20	14	PC.2	**I/O**	Digital GPIO pin
			MISO00	**I/O**	SPI0 1st MISO (Master In, Slave Out) pin
			I2SDI	**I**	I2S data input pin
29	21	15	PC.1	**I/O**	Digital GPIO pin
			SPICLK0	**I/O**	SPI0 serial clock pin
			I2SBCLK	**I/O**	I2S bit clock pin
30	22	16	PC.0	**I/O**	Digital GPIO pin
			SPISS00	**I/O**	SPI0 1st slave select pin
			I2SLRCLK	**I/O**	I2S left/right channel clock pin
31	23		PB.10	**I/O**	Digital GPIO pin
			SPISS01	**I/O**	SPI0 2nd slave select pin
			TM2	**I/O**	Timer2 event counter input / toggle output pin
32	24		PB.9	**I/O**	Digital GPIO pin
			SPISS11	**I/O**	SPI1 2nd slave select pin
			TM1	**I/O**	Timer1 event counter input / toggle output pin
33			VSS	**P**	Ground
34	25	17	PC.13	**I/O**	Digital GPIO pin
			MOSI11	**I/O**	SPI1 2nd MOSI (Master Out, Slave In) pin
			PWM3	**I/O**	PWM3 PWM output / capture input pin
			CLKO	**O**	Frequency Divider output pin
35	26	18	PC.12	**I/O**	Digital GPIO pin
			MISO11	**I/O**	SPI1 2nd MISO (Master In, Slave Out) pin
			PWM2	**I/O**	PWM2 PWM output / capture input pin
			I2SMCLK	**O**	I2S master clock output pin
36	27	19	PC.11	**I/O**	Digital GPIO pin
			MOSI10	**I/O**	SPI1 1st MOSI (Master Out, Slave In) pin
37	28	20	PC.10	**I/O**	Digital GPIO pin
			MISO10	**I/O**	SPI1 1st MISO (Master In, Slave Out) pin

38			VDD	**P**	Power supply for I/O ports and LDO source for internal PLL and digital function. Voltage range is 2.5V ~ 5V.
39	29	21	PC.9	**I/O**	Digital GPIO pin
			SPICLK1	**I/O**	SPI1 serial clock pin
40	30	22	PC.8	**I/O**	Digital GPIO pin
			SPISS10	**I/O**	SPI1 1st slave select pin
41	31		PA.15	**I/O**	Digital GPIO pin
			PWM3	**I/O**	PWM3 PWM output / capture input pin
			I2SMCLK	**O**	I^2S master clock output pin
			CLKO	**O**	Frequency Divider output pin
42			VSS	**P**	Ground
43	32		PA.14	**I/O**	Digital GPIO pin
			PWM2	**I/O**	PWM2 PWM output / capture input pin
44	33		PA.13	**I/O**	Digital GPIO pin
			PWM1	**I/O**	PWM1 PWM output / capture input pin
45	34		PA.12	**I/O**	Digital GPIO pin
			PWM0	**I/O**	PWM0 PWM output / capture input pin
46	35	23	ICE_DAT	**I/O**	Serial wired debugger data pin
47	36	24	ICE_CK	**I**	Serial wired debugger clock input pin
48	37	25	AVDD	**AP**	Power supply for internal analog circuit
49	38		PD.0	**I/O**	Digital GPIO pin
			ADC0	**AI**	ADC channel 0 analog input pin
			SPISS20	**I/O**	SPI2 1st slave select pin
			SPISS11	**I/O**	SPI1 2nd slave select pin
50	39	26	PD.1	**I/O**	Digital GPIO pin
			ADC1	**AI**	ADC channel 1 analog input pin
			SPICLK2	**I/O**	SPI2 serial clock pin
			SPISS01	**I/O**	SPI0 2nd slave select pin
51	40	27	PD.2	**I/O**	Digital GPIO pin
			ADC2	**AI**	ADC channel 2 analog input pin
			MISO20	**I/O**	SPI2 1st MISO (Master In, Slave Out) pin
			MISO01	**I/O**	SPI0 2nd MISO (Master In, Slave Out) pin

52	41	28	PD.3	**I/O**	Digital GPIO pin
			ADC3	**AI**	ADC channel 3 analog input pin
			MOSI20	**I/O**	SPI2 1st MOSI (Master Out, Slave In) pin
			MOSI01	**I/O**	SPI0 2nd MOSI (Master Out, Slave In) pin
53	42		PD.4	**I/O**	Digital GPIO pin
			ADC4	**AI**	ADC channel 4 analog input pin
			MISO21	**I/O**	SPI2 2nd MISO (Master In, Slave Out) pin
54	43		PD.5	**I/O**	Digital GPIO pin
			ADC5	**AI**	ADC channel 5 analog input pin
			MOSI21	**I/O**	SPI2 2nd MOSI (Master Out, Slave In) pin
55			PB.15	**I/O**	Digital GPIO pin
			/INT1	**I**	External interrupt 1 input pin
			T0EX	**I**	Timer0 external capture input pin
56	44	29	PF.0	**I/O**	Digital GPIO pin
			XT1_OUT	**O**	External 4~24 MHz high speed crystal output pin
57	45	30	PF.1	**I/O**	Digital GPIO pin
			XT1_IN	**I**	External 4~24 MHz high speed crystal input pin
58	46	31	/RESET	**I**	External reset input: Low active, set this pin low reset chip to initial state. With internal pull-up.
59			VSS	**P**	Ground
60			VDD	**P**	Power supply for I/O ports and LDO source for internal PLL and digital circuit. Voltage range is 2.5 V ~ 5V.
61	47		PF.2	**I/O**	Digital GPIO pin
			ADC6	**AI**	ADC channel 6 analog input pin
			I2C0SDA	**I/O**	I^2C0 data input/output pin
			PS2DAT	**I/O**	PS/2 data pin
62	48		PF.3	**I/O**	Digital GPIO pin
			ADC7	**AI**	ADC channel 7 analog input pin
			I2C0SDA	**I/O**	I^2C0 clock pin
			PS2CLK	**I/O**	PS/2 clock pin
63	1	32	PVSS	**P**	PLL ground
64	2		PB.8	**I/O**	Digital GPIO pin
			TM0	**I/O**	Timer2 event counter input / toggle output pin

注意: Pin Type I=数字输入, O=数字输出; AI=模拟输入; P=Power Pin; AP=Analog Power

4 框图

4.1 NuMicro™ NUC123 框图

4.1.1 NuMicro™ NUC123 框图

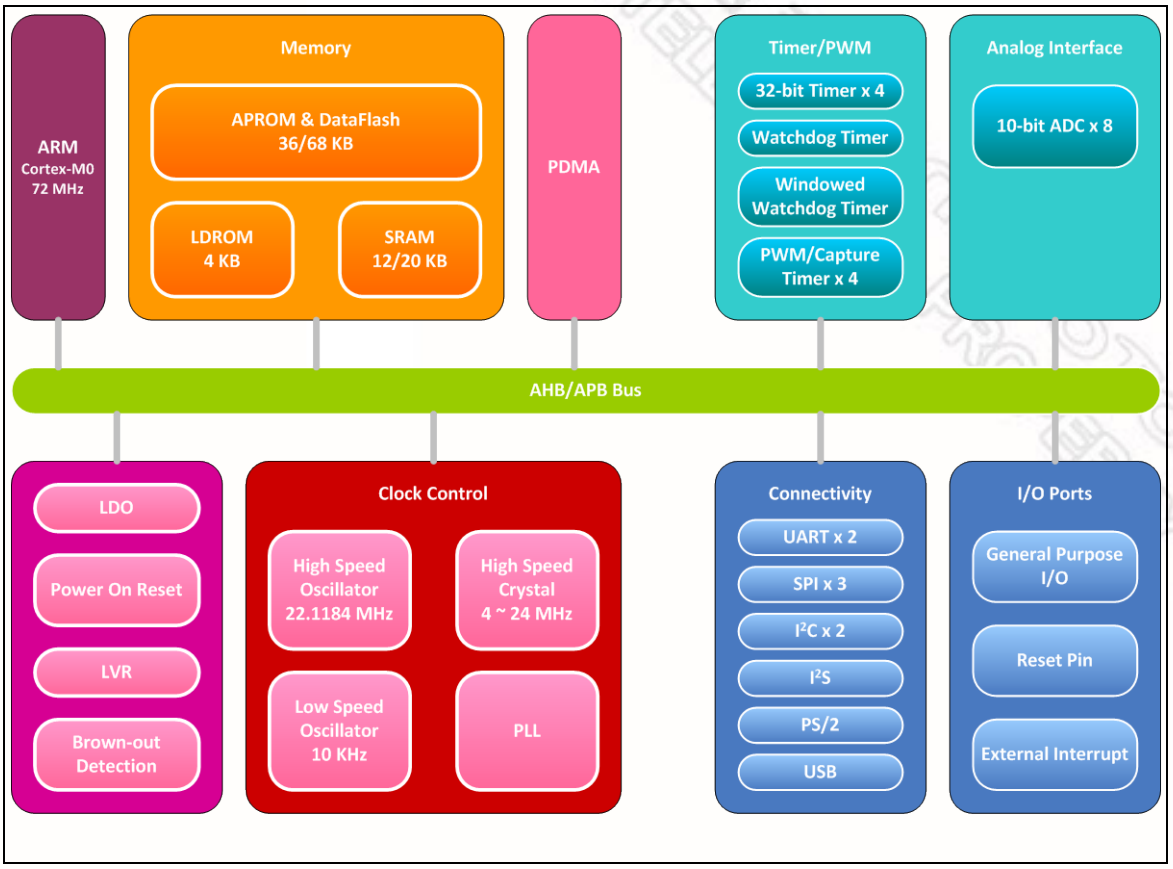


图 4-1 NuMicro™ NUC123 框图

5 功能描述

5.1 内存架构

5.1.1 概述

NuMicro™ NUC123 系列提供 4G-字节的寻址空间. 每个外设的内存地址如下图所示. 外设寄存器的详细描述，地址和编程细节在稍后章节将详细描述. NuMicro™ NUC123 系列只支持小端数据格式

5.1.2 系统内存映射

地址	标志	控制器
Flash 和 SRAM 内存空间		
0x0000_0000 – 0x0000_FFFF	FLASH_BA	FLASH 内存空间(64KB)
0x2000_0000 – 0x2000_4FFF	SRAM_BA	SRAM 内存空间(20KB)
AHB 控制器空间(0x5000_0000 – 0x501F_FFFF)		
0x5000_0000 – 0x5000_01FF	GCR_BA	系统全局控制寄存器
0x5000_0200 – 0x5000_02FF	CLK_BA	时钟控制寄存器
0x5000_0300 – 0x5000_03FF	INT_BA	多路中断控制寄存器
0x5000_4000 – 0x5000_7FFF	GPIO_BA	GPIO 控制寄存器
0x5000_8000 – 0x5000_BFFF	PDMA_BA	外设 DMA 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 ~ 0x400F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	Watchdog/Window Watchdog 定时器控制寄存器
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 控制寄存器
0x4002_0000 – 0x4002_3FFF	I2C0_BA	I^2C0 接口控制寄存器
0x4003_0000 – 0x4003_3FFF	SPI0_BA	带主/从功能的SPI0 控制寄存器
0x4003_4000 – 0x4003_7FFF	SPI1_BA	带主/从功能的SPI1 控制寄存器
0x4004_0000 – 0x4004_3FFF	PWMA_BA	PWM0/1/2/3 控制寄存器
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0 控制寄存器
0x4006_0000 – 0x4006_3FFF	USBD_BA	USB 2.0 FS 设备控制寄存器
0x400E_0000 – 0x400E_FFFF	ADC_BA	模数转换器 (ADC) 控制寄存器
APB2 控制器空间(0x4010_0000 ~ 0x401F_FFFF)		
0x4010_0000 – 0x4010_3FFF	PS2_BA	PS/2 接口控制寄存器
0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3 控制寄存器
0x4012_0000 – 0x4012_3FFF	I2C1_BA	I^2C1 接口控制寄存器
0x4013_0000 – 0x4013_3FFF	SPI2_BA	带主/从功能的SPI2控制寄存器

0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 控制寄存器
0x401A_0000 – 0x401A_3FFF	I2S_BA	I^2S 接口控制寄存器
System Controllers Space (0xE000_E000 ~ 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SCS_BA	系统定时器控制寄存器
0xE000_E100 – 0xE000_ECFF	SCS_BA	外部中断控制器控制寄存器
0xE000_ED00 – 0xE000_ED8F	SCS_BA	System控制寄存器

表 5-1 片上控制器的地址空间分配

5.2 嵌套向量中断控制器 (NVIC)

5.2.1 概述

Cortex-M0提供中断控制器，作为异常模式不可缺少的一部分，称之为“嵌套向量中断控制器(NVIC)”. NVIC和处理器内核紧密相连。

5.2.2 特性

- 支持嵌套和向量中断
- 自动保存和恢复处理器状态
- 动态改变优先级
- 简化的和确定的中断时间

NVIC依照优先级处理所有支持的异常，所有异常在“处理器模式“处理. NVIC结构支持32(IRQ[31:0])个离散中断，每个中断可以支持4级离散中断优先级. 所有的中断和大多数系统异常可以配置为不同优先级. 当中断发生时，NVIC将比较新中断与当前中断的优先级，如果新中断优先级高，则立即处理新中断.

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

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

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

5.2.3 异常模式和系统中断映射

NuMicro™ NUC123 支持表 5-2 所列的异常模式. 与所有中断一样，软件可以对其中一些中断设置4级优先级. 最高优先级为“0”，最低优先级为“3”，所有用户可配置的优先级的默认值为“0”. 注意：优先级为“0”在整个系统中为第4优先级，排在“Reset”, “NMI” 与“Hard Fault”之后.

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

表 5-2 异常模式

向量号	中断号 (Bit in Interrupt Registers)	中断名称	源 IP	中断描述
0 ~ 15	-	**-**	-	系统异常
16	0	**BOD_OUT**	Brown-Out	欠压检测中断
17	1	**WDT_INT**	WDT	看门狗定时器 中断
18	2	**EINT0**	GPIO	PB.14脚上的外部信号中断
19	3	**EINT1**	GPIO	PB.15脚上的外部信号中断
20	4	**GPAB_INT**	GPIO	PA[15:0] / PB[13:0]的外部信号中断
21	5	**GPCDF_INT**	GPIO	PC[15:0]/PD[15:0]/PF[15:0] 的外部信号中断
22	6	**PWMA_INT**	PWM0~3	PWM0, PWM1, PWM2 与 PWM3 中断
23	7	**保留**	保留	保留
24	8	**TMR0_INT**	TMR0	Timer 0中断
25	9	**TMR1_INT**	TMR1	Timer 1中断

26	10	**TMR2_INT**	TMR2	Timer 2中断
27	11	**TMR3_INT**	TMR3	Timer 3中断
28	12	**UART0_INT**	UART0	UART0 中断
29	13	**UART1_INT**	UART1	UART1中断
30	14	**SPI0_INT**	SPI0	SPI0中断
31	15	**SPI1_INT**	SPI1	SPI1中断
32	16	**SPI2_INT**	SPI2	SPI2中断
33	17	**保留**	保留	保留
34	18	**I2C0_INT**	I^2C0	I^2C0中断
35	19	**I2C1_INT**	I^2C1	I^2C1中断
36	20	**保留**	保留	保留
37	21	**保留**	保留	保留
38	22	**保留**	保留	保留
39	23	**USB_INT**	USBD	USB 2.0 FS设备中断
40	24	**PS2_INT**	PS/2	PS/2 中断
41	25	**保留**	保留	保留
42	26	**PDMA_INT**	PDMA	PDMA 中断
43	27	**I2S_INT**	I^2S	I^2S 中断
44	28	**PWRWU_INT**	CLKC	从掉电状态唤醒的时钟控制器中断
45	29	**ADC_INT**	ADC	ADC 中断
46	30	**保留**	保留	保留
47	31	**保留**	保留	保留

表 5-3 系统中断映射

5.2.4 向量表

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

向量表字偏移量	*描述*
0	SP_main – 主栈指针
向量号	异常入口指针，用向量号表示

表 5-4 向量表格式

5.2.5 操作说明

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

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

NVIC中断依次更新32位寄存器中的各个8位字段（每个寄存器支持4个中断）来设置中断的优先级.

与NVIC相关的通用寄存器都可以在内存系统控制空间寄存器（SCS_BA）其中的一块寄存器区域中设置，下一节将作出描述.

5.2.6 NVIC 控制寄存器

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
SCS 基地址: **SCS_BA = 0xE000_E000**				
NVIC_ISER	SCS_BA+0x100	R/W	IRQ0 ~ IRQ31 设置使能控制寄存器	0x0000_0000
NVIC_ICER	SCS_BA+0x180	R/W	IRQ0 ~ IRQ31 清使能控制寄存器	0x0000_0000
NVIC_ISPR	SCS_BA+0x200	R/W	IRQ0 ~ IRQ31 设置挂起控制寄存器	0x0000_0000
NVIC_ICPR	SCS_BA+0x280	R/W	IRQ0 ~ IRQ31 清挂起控制寄存器	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 ~ IRQ27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设置使能控制寄存器（NVIC_ISER）

寄存器	偏移量	R/W	描述	复位的值
NVIC_ISER	SCS _BA+0x100	R/W	IRQ0 ~ IRQ31中断使能控制寄存器	0x0000_0000

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

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

IRQ0 ~ IRQ31清使能控制寄存器(NVIC_ICER)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ICER	SCS _BA+0x180	R/W	IRQ0 ~ IRQ31禁止中断控制寄存器	0x0000_0000

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

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

IRQ0 ~ IRQ31设置/挂起控制寄存器（NVIC_ISPR）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ISPR	SCS _BA+0x200	R/W	IRQ0 ~ IRQ31设置挂起控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	SETPEND	软件写1，由软件控制发起相应中断.每位代表从IRQ0 ~ IRQ31 的中断号(向量号： 16 ~ 47). 写0无效 读取寄存器返回当前等待处理的中断状态.

IRQ0 ~ IRQ31清挂起控制寄存器（NVIC_ICPR）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ICPR	SCS _BA+0x280	R/W	IRQ0 ~ IRQ31清挂起控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CLRPEND	写1清除，由软件控制清除等待处理的中断，每位代表从IRQ0 ~ IRQ31的中断号 (向量号： 16 ~ 47). 写0无效. 读取寄存器返回当前等待处理的中断状态.

IRQ0 ~ IRQ3中断优先级寄存器（NVIC_IPR0）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR0	SCS _BA+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		保留					
23	22	21	20	19	18	17	16
PRI_2		保留					
15	14	13	12	11	10	9	8
PRI_1		保留					
7	6	5	4	3	2	1	0
PRI_0		保留					

Bits	描述	
[31:30]	PRI_3	IRQ3优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:24]	保留	保留
[23:22]	PRI_2	IRQ2优先级 “0” 表示最高优先级& “3” 表示最低优先级
[21:16]	保留	保留
[15:14]	PRI_1	IRQ1优先级 “0” 表示最高优先级& “3” 表示最低优先级
[13:8]	保留	保留
[7:6]	PRI_0	IRQ0优先级 “0” 表示最高优先级& “3” 表示最低优先级
[5:0]	保留	保留

IRQ4 ~ IRQ7中断优先级寄存器（NVIC_IPR1）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR1	SCS _BA+0x404	R/W	IRQ4 ~ IRQ7中断优先级寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_7	IRQ7优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:24]	保留	保留
[23:22]	PRI_6	IRQ6优先级 “0” 表示最高优先级& “3” 表示最低优先级
[21:16]	保留	保留
[15:14]	PRI_5	IRQ5优先级 “0” 表示最高优先级& “3” 表示最低优先级
[13:8]	保留	保留
[7:6]	PRI_4	IRQ4优先级 “0” 表示最高优先级& “3” 表示最低优先级
[5:0]	保留	保留

IRQ8 ~ IRQ11 中断优先级寄存器（NVIC_IPR2）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR2	SCS _BA+0x408	R/W	IRQ8 ~ IRQ11中断优先级寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_11	IRQ11优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:24]	保留	保留
[23:22]	PRI_10	IRQ10优先级 “0” 表示最高优先级& “3” 表示最低优先级
[21:16]	保留	保留
[15:14]	PRI_9	IRQ9优先级 “0” 表示最高优先级& “3” 表示最低优先级
[13:8]	保留	保留
[7:6]	PRI_8	IRQ8优先级 “0” 表示最高优先级& “3” 表示最低优先级
[5:0]	保留	保留

IRQ12 ~ IRQ15中断优先级寄存器（NVIC_IPR3）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR3	SCS _BA+0x40C	R/W	IRQ12 ~ IRQ15中断优先级寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_15	IRQ15优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:24]	保留	保留
[23:22]	PRI_14	IRQ14优先级 “0” 表示最高优先级& “3” 表示最低优先级
[21:16]	保留	保留
[15:14]	PRI_13	IRQ13优先级 “0” 表示最高优先级& “3” 表示最低优先级
[13:8]	保留	保留
[7:6]	PRI_12	IRQ12优先级 “0” 表示最高优先级& “3” 表示最低优先级
[5:0]	保留	保留

IRQ16 ~ IRQ19中断优先级寄存器（NVIC_IPR4）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR4	SCS _BA+0x410	R/W	IRQ16 ~ IRQ19中断优先级寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_19	IRQ19优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:24]	保留	保留
[23:22]	PRI_18	IRQ18优先级 “0” 表示最高优先级& “3” 表示最低优先级
[21:16]	保留	保留
[15:14]	PRI_17	IRQ17优先级 “0” 表示最高优先级& “3” 表示最低优先级
[13:8]	保留	保留
[7:6]	PRI_16	IRQ16优先级 “0” 表示最高优先级& “3” 表示最低优先级
[5:0]	保留	保留

IRQ20 ~ IRQ23中断优先级寄存器（NVIC_IPR5）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR5	SCS _BA+0x414	R/W	IRQ20 ~ IRQ23中断优先级寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_23	IRQ23优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:24]	保留	保留
[23:22]	PRI_22	IRQ22优先级 “0” 表示最高优先级& “3” 表示最低优先级
[21:16]	保留	保留
[15:14]	PRI_21	IRQ21优先级 “0” 表示最高优先级& “3” 表示最低优先级
[13:8]	保留	保留
[7:6]	PRI_20	IRQ20优先级 “0” 表示最高优先级& “3” 表示最低优先级
[5:0]	保留	保留

IRQ24 ~ IRQ27中断优先级寄存器（NVIC_IPR6）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR6	SCS _BA+0x418	R/W	IRQ24 ~ IRQ27中断优先级寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_27	IRQ27优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:24]	保留	保留
[23:22]	PRI_26	IRQ26优先级 “0” 表示最高优先级& “3” 表示最低优先级
[21:16]	保留	保留
[15:14]	PRI_25	IRQ25优先级 “0” 表示最高优先级& “3” 表示最低优先级
[13:8]	保留	保留
[7:6]	PRI_24	IRQ24优先级 “0” 表示最高优先级& “3” 表示最低优先级
[5:0]	保留	保留

IRQ28 ~ IRQ31中断优先级寄存器（NVIC_IPR7）

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR7	SCS _BA+0x41C	R/W	IRQ28 ~ IRQ31中断优先级寄存器	0x0000_0000

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

Bits	描述	
[31:30]	PRI_31	IRQ31优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:24]	保留	保留
[23:22]	PRI_30	IRQ30优先级 “0” 表示最高优先级& “3” 表示最低优先级
[21:16]	保留	保留
[15:14]	PRI_29	IRQ29优先级 “0” 表示最高优先级& “3” 表示最低优先级
[13:8]	保留	保留
[7:6]	PRI_28	IRQ28优先级 “0” 表示最高优先级& “3” 表示最低优先级
[5:0]	保留	保留

5.2.7 中断源控制寄存器

除了NVIC相关的中断控制寄存器外，NuMicro™ NUC123 还有一些特殊寄存器便于中断处理，包括“中断源识别”，”NMI 源选择”与 “中断测试模式”。描述如下.

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
INT 基地址: **INT_BA = 0x5000_0300**				
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) 中断源识别	0xXXXX_XXXX
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别	0xXXXX_XXXX
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 中断源识别	0xXXXX_XXXX
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别	0xXXXX_XXXX
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (GPA/B) 中断源识别	0xXXXX_XXXX
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GPC/D/F) 中断源识别	0xXXXX_XXXX
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) 中断源识别	0xXXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (保留) 中断源识别	0xXXXX_XXXX
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别	0xXXXX_XXXX
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别	0xXXXX_XXXX
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别	0xXXXX_XXXX
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别	0xXXXX_XXXX
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (URT0) 中断源识别	0xXXXX_XXXX
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (URT1) 中断源识别	0xXXXX_XXXX
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别	0xXXXX_XXXX
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) 中断源识别	0xXXXX_XXXX
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (SPI2) 中断源识别	0xXXXX_XXXX
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (保留) 中断源识别	0xXXXX_XXXX
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I^2C0) 中断源识别	0xXXXX_XXXX
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I^2C1) 中断源识别	0xXXXX_XXXX
IRQ20_SRC	INT_BA+0x50	R	IRQ20 (保留) 中断源识别	0xXXXX_XXXX
IRQ21_SRC	INT_BA+0x54	R	IRQ21 (保留) 中断源识别	0xXXXX_XXXX

IRQ22_SRC	INT_BA+0x58	R	IRQ22 (保留) 中断源识别	0xXXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (USBD) 中断源识别	0xXXXX_XXXX
IRQ24_SRC	INT_BA+0x60	R	IRQ24 (PS/2) 中断源识别	0xXXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (保留) 中断源识别	0xXXXX_XXXX
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (PDMA) 中断源识别	0xXXXX_XXXX
IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (I^2S) 中断源识别	0xXXXX_XXXX
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源识别	0xXXXX_XXXX
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (ADC) 中断源识别	0xXXXX_XXXX
IRQ30_SRC	INT_BA+0x78	R	IRQ30 (保留) 中断源识别	0xXXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	R	IRQ31 (保留) 中断源识别	0xXXXX_XXXX
NMI_SEL	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 中断号识别寄存器	0x0000_0000

IRQ0 (BOD) 中断源识别寄存器 (IRQ0_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: BOD_INT

IRQ1 (WDT) 中断源识别寄存器 (IRQ1_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: WDT_INT

IRQ2 (EINT0) 中断源识别寄存器 (IRQ2_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: EINT0 –PB.14上的外部中断0

IRQ3 (EINT1) 中断源识别寄存器 (IRQ3_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: EINT1 –PB.15 或PD.11上的外部中断1

IRQ4 (GPA/GPB) 中断源识别寄存器 (IRQ4_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (GPA/B) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: GPB_INT Bit0: GPA_INT

IRQ5 (GPC/GPD/ GPF) 中断源识别寄存器 (IRQ5_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GPC/D/F) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:4]	保留	保留
[3:0]	INT_SRC	Bit3: GPF_INT Bit2: 0 Bit1: GPD_INT Bit0: GPC_INT

IRQ6 (PWMA) 中断源识别寄存器 (IRQ6_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:4]	保留	保留
[3:0]	INT_SRC	Bit3: PWM3_INT Bit2: PWM2_INT Bit1: PWM1_INT Bit0: PWM0_INT

IRQ8 (TMR0) 中断源识别寄存器 (IRQ8_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: TMR0_INT

IRQ9 (TMR1) 中断源识别寄存器 (IRQ9_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: TMR1_INT

IRQ10 (TMR2) 中断源识别寄存器 (IRQ10_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: TMR2_INT

IRQ11 (TMR3) 中断源识别寄存器 (IRQ11_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: TMR3_INT

IRQ12 (UART0) 中断源识别寄存器 (IRQ12_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (UART0) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: URT0_INT

IRQ13 (UART1) 中断源识别寄存器 (IRQ13_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (UART1) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: UART1_INT

IRQ14 (SPI0) 中断源识别寄存器 (IRQ14_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: SPI0_INT

IRQ15 (SPI1) 中断源识别寄存器 (IRQ15_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: SPI1_INT

IRQ16 (SPI2) 中断源识别寄存器 (IRQ16_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (SPI2) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: SPI2_INT

IRQ18 (I^2C0) 中断源识别寄存器 (IRQ18_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I^2C0) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: I2C0_INT

IRQ19 (I^2C1) 中断源识别寄存器 (IRQ19_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I^2C1) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: I2C1_INT

IRQ23 (USBD) 中断源识别寄存器 (IRQ23_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (USBD) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: USB_INT

IRQ24 (PS/2) 中断源识别寄存器 (IRQ24_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ24_SRC	INT_BA+0x60	R	IRQ24 (PS/2) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: PS2_INT

IRQ26 (PDMA) 中断源识别寄存器 (IRQ26_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (PDMA) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: PDMA_INT

IRQ27 (I^2S) 中断源识别寄存器 (IRQ27_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (I^2S) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: I2S_INT

IRQ28 (PWRWU) 中断源识别寄存器 (IRQ28_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: PWRWU_INT

IRQ29 (ADC) 中断源识别寄存器 (IRQ29_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (ADC) 中断源识别寄存器	0xXXXX_XXXX

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

Bits	Description	
[31:3]	保留	保留
[2:0]	INT_SRC	Bit2: 0 Bit1: 0 Bit0: ADC_INT

NMI中断源选择控制寄存器 (NMI_SEL)

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

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

Bits	描述	
[31:5]	**保留**	保留
[4:0]	**NMI_SEL**	NMI 中断源选择 通过设置NMI_SEL可以在外围设备中断中选择Cortex-M0的NMI 中断.

MCU中断请求源寄存器 (MCU_IRQ)

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

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

Bits	描述	
[31:0]	MCU_IRQ	**MCU中断请求寄存器** MCU_IRQ 从外围设备收集所有中断，并向Cortex-M0内核产生同步中断，生成中断的模式有正常模式与测试模式. MCU_IRQ 从每个外围设备收集所有中断和同步这些中断，然后触发Cortex-M0中断. MCU_IRQ[n] 是 “0”时：置 MCU_IRQ[n] 为“1”，Cortex_M0 NVIC[n]发生一个中断. MCU_IRQ[n] 是 “1”时: (意味着有中断请求) 置MCU_IRQ[n]为1将清除中断；置MCU_IRQ[n]为0无效

5.3 系统管理器

5.3.1 概述

系统管理器包括如下功能:

- 系统复位
- 系统内存映射
- 产品ID、芯片复位、模块功能复位和多功能管脚控制的系统管理寄存器
- 系统定时器(SysTick)
- 嵌套向量中断控制器(NVIC)
- 系统控制寄存器

5.3.2 系统复位

下列情况发生时，系统复位，复位标志由寄存器 **RSTSRC** 读出.

- 上电复位
- 复位脚（/RESET）上有低电平
- 看门狗复位
- 低压复位
- 欠压检测器复位
- CPU 复位
- 系统复位

系统复位和上电复位使整个芯片复位，包括所有外设。系统复位与上电复位的区别在于外部晶振电路与ISPCON.BS位。系统复位不复位外部晶振电路和ISPCON.BS位，上电复位可以。

5.3.3 系统电源分配

该器件的电源分为三个部分.

- 由AVDD 和AVSS提供的模拟电源，为模拟部分工作提供电压.
- 由VDD与VSS提供的数字电源，提供一个固定的2.5V的数字电源，用于数字操作和I/O引脚的内部稳压电源.
- VBUS提供给USB 的电源，用于USB模块传输操作. (仅用于NuMicro™ NUC123)

内部电压调节器输出，LDO和VDD33，需要在相应的引脚上外接电容. 图 5-1 为NuMicro™ NUC123的电源分配图.

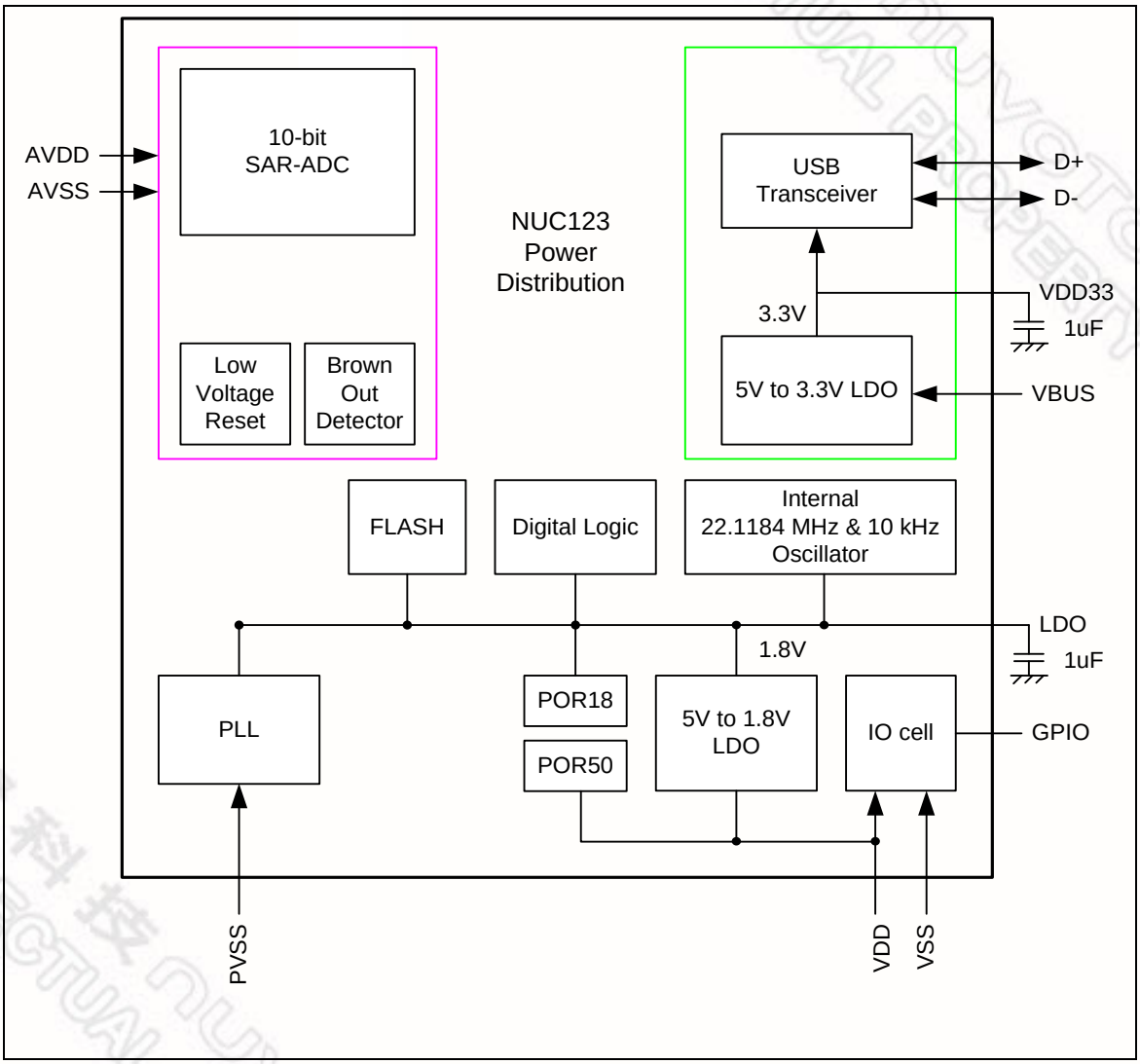


图 5-1 NuMicro™ NUC123 电源分配图

5.3.4 系统管理器控制寄存器

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
GCR 基地址: **GCR_BA = 0x5000_0000**				
PDID	GCR_BA+0x00	R	器件ID寄存器	0x0001_23XX[1]
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX
IPRSTC1	GCR_BA+0x08	R/W	外设复位控制寄存器1	0x0000_0000
IPRSTC2	GCR_BA+0x0C	R/W	外设复位控制寄存器2	0x0000_0000
BODCR	GCR_BA+0x18	R/W	欠压检测控制寄存器	0x0000_008X
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_00XX
GPA_MFP	GCR_BA+0x30	R/W	GPIOA多功能和输入类型控制寄存器	0x0000_0000
GPB_MFP	GCR_BA+0x34	R/W	GPIOB多功能和输入类型控制寄存器	0x0000_0000
GPC_MFP	GCR_BA+0x38	R/W	GPIOC多功能和输入类型控制寄存器	0x0000_0000
GPD_MFP	GCR_BA+0x3C	R/W	GPIOD多功能和输入类型控制寄存器	0x0000_0000
GPF_MFP	GCR_BA+0x40	R/W	GPIOF多功能和输入类型控制寄存器	0x0000_000X
ALT_MFP	GCR_BA+0x50	R/W	复用多功能引脚控制寄存器	0x0000_0000
ALT_MFP1	GCR_BA+0x54	R/W	复用多功能引脚控制寄存器1	0x0000_0000
GPA_IOCR	GCR_BA+0xC0	R/W	GPIOA IO 控制寄存器	0x0000_0000
GPB_IOCR	GCR_BA+0xC4	R/W	GPIOB IO 控制寄存器	0x0000_0000
GPD_IOCR	GCR_BA+0xCC	R/W	GPIOD IO 控制寄存器	0x0000_0000
REGWRPROT	GCR_BA+0x100	R/W	寄存器写保护寄存器	0x0000_0000

Note: [1] Dependents on part number.

器件ID寄存器(PDID)

寄存器	偏移量	R/W	描述	复位后的值
PDID	GCR_BA+0x00	R	器件ID寄存器	0x0001_23XX[1]

[1]每类器件都有一个唯一的默认复位值.

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

Bits	描述	
[31:0]	PDID	产品器件识别码 该寄存器反映器件的器件号码。软件可以读该寄存器来识别所使用的器件.

PART NUMBER	PACKAGE	FLASH	SRAM	PDID
NUC123ZD4AN0	QFN-33	68	20	0x0001_2355
NUC123ZC2AN1	QFN-33	36	12	0x0001_2345
NUC123LD4AN0	LQFP-48	68	20	0x0001_2335
NUC123LC2AN1	LQFP-48	36	12	0x0001_2325
NUC123SD4AN0	LQFP-64	68	20	0x0001_2315
NUC123SC2AN1	LQFP-64	36	12	0x0001_2305

系统复位源寄存器(RSTSRC)

该寄存器提供一些信息用于识别引起芯片上次复位操作的复位源.

寄存器	偏移量	R/W	描述	复位后的值
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX

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

Bits	描述	
[31:8]	保留	保留
[7]	RSTS_CPU	**如果软件写1到CPU_RST(IPRTC1[1])，复位Coterx-M0 CPU的核和Flash控制器，硬件会把RSTS_CPU标志位置‘1’** 1= Cortex-M0 CPU 内核与FMC因为软件置CPU_RST为1而复位. 0= CPU无复位 向该位写1清零.
为[6]	保留	保留
[5]	RSTS_SYS	**RSTS_SYS**标志位由来自MCU Cortex_M0 的“复位信号“置位，以表示导致之前复位的复位源. 1 = Cortex_M0 因为软件向SYSRESTREQ(AIRCR[2])写1，发出复位信号而复位系统,(AIRCR[2]寄存器地址 = 0xE000ED0C). 0 = Cortex_M0无复位 向该位写1清零.
[4]	RSTS_BOD	RSTS_BOD标志位由欠压检测模块的” 复位信号”置位，用于表示导致之前复位的复位源. 1 =欠压检验模块发出复位信号使系统复位 0 = BOD无复位

		向该位写1清零.
[3]	**RSTS_LVR**	RSTS_LVR标志位由低压复位模块的” 复位信号”置位，用于表示导致之前复位的复位源. 1= 低压 LVR 模块发出复位信号使系统复位. 0= LVR无复位 向该位写1清零.
[2]	**RSTS_WDT**	RSTS_WDT标志位由看门狗模块的” 复位信号”置位，用于说明导致之前复位的复位源. 1: 看门狗模块发出复位信号使系统复位. 0: 没有看门狗复位信号 向该位写1清零.
[1]	**RSTS_RESET**	RSTS_RESET 标志位由/RESET脚的” 复位信号”置位，用于说明导致之前复位的复位源. 1: /RESET脚上发出复位信号使系统复位. 0: 没有/RESET复位信号 向该位写1清零.
[0]	**RSTS_POR**	RSTS_POR 标志由POR的“复位信号”或CHIP_RST(IPRSTC1[0])位置位，用于说明导致之前复位的复位源. 1 = 上电复位 (POR) 或 CHIP_RST 发出复位信号使系统复位. 0 =没有POR或CHIP_RST复位信号 向该位写1清零.

外设复位控制寄存器1 (IPRSTC1)

寄存器	偏移量	R/W	描述	复位后的值
IPRSTC1	GCR_BA+0x08	R/W	IP复位控制寄存器1	0x0000_0000

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

Bits	描述	
[31:3]	保留	保留
[2]	PDMA_RST	**PDMA控制器复位** 该位置1，产生复位信号到PDMA，用户需要置0才能释放复位状态. 该位受保护, 编程该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100. 1 = PDMA 控制器复位 0 = PDMA 控制器正常工作
[1]	CPU_RST	**CPU 内核复位**(写保护位) 设置该位仅复位CPU内核和FMC，该位在2个时钟周期后自动清零 该位受保护, 编程该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100. 1 = CPU复位 0 = CPU 正常工作
[0]	CHIP_RST	**CHIP 复位** (写保护位) 设置该位复位整个芯片，包括CPU内核和所有外设，该位在2个时钟周期后，自动清零 CHIP_RST 与上电复位一样，所有芯片控制器都复位，芯片设置从flash重新加载. CHIP_RST 与 SYSRESETREQ的区别，参考section 5.2.2 该位受保护, 编程该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”, 参

		考寄存器 REGWRPROT,地址GCR_BA + 0x100 1 = CHIP 复位 0 = CHIP 正常工作

外设复位控制寄存器2 (IPRSTC2)

置"1" , 产生异步复位信号给相应的IP控制器. 该位置0相应的IP控制器从复位状态恢复

寄存器	偏移量	R/W	描述	复位后的值
IPRSTC2	GCR_BA+0x0C	R/W	外设复位控制寄存器2	0x0000_0000

31	30	29	28	27	26	25	24
保留		I2S_RST	ADC_RST	USBD_RST	保留		
23	22	21	20	19	18	17	16
PS2_RST	保留		PWM03_RST	保留		UART1_RST	UART0_RST
15	14	13	12	11	10	9	8
保留	SPI2_RST	SPI1_RST	SPI0_RST	保留		I2C1_RST	I2C0_RST
7	6	5	4	3	2	1	0
保留		TMR3_RST	TMR2_RST	TMR1_RST	TMR0_RST	GPIO_RST	保留

Bits	描述	
[31:30]	保留	保留
[29]	I2S_RST	**I²S控制器复位** 1 = I²S控制器复位 0 = I²S控制器正常工作
[28]	ADC_RST	**ADC 控制器复位** 1 = ADC控制器复位 0 = ADC控制器正常工作
[27]	USBD_RST	**USB 设备控制器复位** 1 = USB设备控制器复位 0 = USB 设备控制器正常工作
[26:24]	保留	保留
[23]	PS2_RST	**PS/2控制器复位** 1 = PS/2控制器复位 0 = PS/2控制器正常工作
[22:21]	保留	保留
[20]	PWM03_RST	**PWM03控制器复位** 1 = PWM03 控制器复位

		0 = PWM03 控制器正常工作
[19:18]	保留	保留
[17]	**UART1_RST**	**UART1控制器复位** 1 = UART1 控制器复位 0 = UART1 控制器正常工作
[16]	**UART0_RST**	**UART0控制器复位** 1 = UART0 控制器复位 0 = UART0 控制器正常工作
[15]	**保留**	保留
[14]	**SPI2_ RST**	**SPI2控制器复位**(Medium Density Only) 1 = SPI2 控制器复位 0 = SPI2 控制器正常工作
[13]	**SPI1_ RST**	**SPI1控制器复位** 1 = SPI1 控制器复位 0 = SPI1 控制器正常工作
[12]	**SPI0_ RST**	**SPI0控制器复位** 1 = SPI0 控制器复位 0 = SPI0 控制器正常工作
[11:10]	保留	保留
[9]	**I2C1_RST**	**I^2C1控制器复位** 1 = I^2C1 控制器复位 0 = I^2C1 控制器正常工作
[8]	**I2C0_RST**	**I^2C0控制器复位** 1 = I^2C0 控制器复位 0 = I^2C0 控制器正常工作
[7:6]	保留	保留
[5]	**TMR3_RST**	**Timer3控制器复位** 1 = Timer3 控制器复位 0 = Timer3 控制器正常工作
[4]	**TMR2_RST**	**Timer2控制器复位** 1 = Timer2 控制器复位 0 = Timer2 控制器正常工作
[3]	**TMR1_RST**	**Timer1控制器复位**

		1 = Timer1 控制器复位 0 = Timer1 控制器正常工作
[2]	**TMR0_RST**	**Timer0控制器复位** 1 = Timer0 控制器复位 0 = Timer0 控制器正常工作
[1]	**GPIO_RST**	**GPIO控制器复位** 1 = GPIO 控制器复位 0 = GPIO 控制器正常工作
[0]	保留	保留

欠压检测控制寄存器(BODCR)

BODCR 控制寄存器的部分位在flash配置时（config0寄存器）已经初始化，部分位是受保护的位. 编程这些写保护的位，需要向控制寄存器0x5000_0100依次写入“59h”, “16h” “88h”, 参考REGWRPROT（GCR_BA+0x100）

寄存器	偏移量	R/W	描述	复位后的值
BODCR	GCR_BA+0x18	R/W	欠压检测控制寄存器	0x0000_008X

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

Bits	描述	
[31:8]	保留	保留
[7]	LVR_EN	**低压复位使能**(写保护位) 当输入电源电压低于LVR电路设置时，LVR复位芯片。默认使能低电压复位功能。 1= 使能低电压复位功能，使能该位100us后，低电压复位输出稳定，LVR功能生效.(默认). 0= 禁用低电压复位功能 该位受保护, 编程该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100
[6]	BOD_OUT	**欠压检测输出的状态位** 1 =欠压检测输出状态为 1, 表示检测到的电压低于 BOD_VL的设置. 若BOD_EN 是“0”, 禁用BOD功能，该位通常响应必定为 “0” 0 =欠压检测输出状态为0. 表示检测电压高于BOD_VL的设置或BOD_EN为0
[5]	BOD_LPM	**低压模式下的欠压检测**(写保护位) 1 = 使能 BOD 低功耗模式 0 = BOD 工作于正常模式(默认) BOD 在正常模式下消耗电流约为100 uA, 低功耗模式下减少到当前的1/10，但BOD响应速度变慢. 该位受保护, 编程该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”来

		禁用寄存器保护。参考寄存器 REGWRPROT,地址GCR_BA + 0x100.
[4]	**BOD_INTF**	**欠压检测中断标志** 1= 欠压检测到VDD 下降到BOD_VL 的设定电压或VDD 升到BOD_VL 的设定电压，该位设置为1，如果欠压中断被使能，则发生欠压中断. 0= 没有检测到任何电压由 VDD 下降或上升至BOD_VL 设定值. 软件写1将该位清零.
[3]	**BOD_RSTEN**	**欠压复位使能**(写保护位) 1 =使能欠压复位功能 当同时使能欠压检测功能 (BOD_EN 为高) 和BOD 复位功能(BOD_RSTEN 为高)时, 如果检测到电压低于threshold (BOD_OUT 为高)，BOD 将发送信号复位芯片 0 = 使能欠压中断功能 当同时使能BOD功能 (BOD_EN high) 和 BOD 中断功能(BOD_RSTEN), 如果BOD_OUT为高BOD 将产生中断. BOD 中断保持直到BOD_EN 被设置为0. 可以通过禁止NVIC BOD中断或禁止BOD功能(设置 BOD_EN low)封锁BOD中断. 默认值由用户在配置flash控制寄存器时config0 bit[20]设置. 该位受保护, 修改该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”来禁用寄存器保护, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100.
[2:1]	**BOD_VL**	**欠压检测Threshold 电压选择**(写保护位) 缺省值 由用户在配置FLASH控制寄存器config0 bit[22:21]时设定 该位受保护, 修改该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”来禁用寄存器保护, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100.
[0]	**BOD_EN**	**欠压检测使能**(写保护位) 缺省值 由用户在配置FLASH控制寄存器config0 bit[23]时置位 1 = 使能欠压检测功能 0 =禁止欠压检测功能 该位受保护, 修改该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”来禁用寄存器保护, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100.

BOV_VL[1]	BOV_VL[0]	Brown out voltage
1	1	4.5
1	0	3.8V
0	1	2.7V
0	0	2.2V

上电复位控制寄存器(PORCR)

寄存器	偏移量	R/W	描述	复位后的值
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_00XX

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

Bits	描述	
[31:16]	保留	保留
[15:0]	POR_DIS_CODE	该寄存器用于上电复位使能控制(写保护位) 上电时，POR电路产生复位信号使整个芯片复位，但是电源部分的干扰可能引起POR重新有效。用户可以将POR_DIS_CODE 设置为0x5AA5, 禁用POR内部电路，以免造成不可预知的干扰，当设置POR_DIS_CODE 为其他值，或者由芯片的其他复位功能引起复位时，POR功能重新有效，这些复位功能包括： /RESET引脚复位，看门狗，LVR复位，BOD复位，ICE复位命令和软件复位 该位受保护, 修改该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”来禁用寄存器保护, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100

GPIOA多功能管脚控制寄存器(GPA_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPA_MFP	GCR_BA+0x30	R/W	GPIOA多功能与输入类型控制寄存器	0x0000_0000

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

Bits	描述	
[31:16]	GPA_TYPEn	1 = 使能GPIOA[15:0] I/O Schmitt触发输入 0 = 禁止GPIOA[15:0] I/O Schmitt触发输入
[15]	GPA_MFP15	**PA.15 Pin功能选择** 该pin功能取决于GPA_MFP15 与 PA15_MFP1 (ALT_MFP[9]). PA15_MFP1 (ALT_MFP[9]) / GPA_MFP[15] / PA.15 Function: 0 / 0 / GPIO 0 / 1 / PWM3 (PWM) 1 / 0 / CLKO (□lock Driver output) 1 / 1 / I2SMCLK (I^2S)
[14]	GPA_MFP14	**PA.14 Pin功能选择** 该pin功能取决于GPA_MFP14. GPA_MFP[14] / PA.14 Function: 0 / GPIO 1 / PWM2 (P□M)
[13]	GPA_MFP13	**PA.13 Pin功能选择** 该pin功能取决于 GPA_MFP13. GPA_MFP[13] / PA.13 Function: 0 / GPIO

		1 PWM1 (PWM)
[12]	**GPA_MFP12**	**PA.12 Pin功能选择** 该pin功能取决于GPA_MFP12. **GPA_MFP[12]** / **PA.12 Function** 0 GPIO 1 PWM0 (PWM)
[11]	**GPA_MFP11**	**PA.11 Pin 功能选择** 该pin功能取决于GPA_MFP11 与PA11_MFP1 (ALT_MFP[11]). **PA11_MFP1 (ALT_MFP[11])** / **GPA_MFP[11]** / **PA.11 Function** 0 0 GPIO 0 1 I2C1SCL (I^2C) 1 0 SPICLK1 (SPI1) 1 1 MOSI20 (SPI2)
[10]	**GPA_MFP10**	**PA.10 Pin 功能选择** 该pin功能取决于 GPA_MFP10 与 PA10_MFP1 (ALT_MFP[12]). **PA10_MFP1 (ALT_MFP[1□])** / **GPA_MFP[10]** / **PA.10 Function** 0 0 □PIO 0 1 I2C1SDA (I^2C1) 1 0 MISO10 (SPI1) 1 1 MISO20 (SPI2)
[9:0]	**保留**	保留

GPIOB多功能管脚控制寄存器(GPB_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPB_MFP	GCR_BA+0x34	R/W	GPIOB多功能与输入类型控制寄存器	0x0000_0000

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

Bits	描述	
[31:16]	GPB_TYPEn	1 = 使能GPIOB[15:0] I/O Schmitt触发输入 0 = 禁止GPIOB[15:0] I/OSchmitt触发输入
[15]	GPB_MFP15	**PB.15 Pin 功能选择** PB15_MFP1 (ALT_MFP[24]) 和 GPB_MFP[15] 决定 PB.15 功能

PB15_MFP1 (ALT_MFP[24])	GPB_MFP[15]	PB.15 Function
0	0	GPIO
0	1	/INT1
1	0	Reserved
1		T0EX (TMR0)

[14]	GPB_MFP14	**PB.14 Pin功能选择** 该引脚功能取决于 GPB_MFP14

GPB_MFP[14]	PB.14 Func□ion
0	GPIO
1	/I□T0

[13]	GPB_MFP13	**PB.13 Pin功能选择** 该引脚只当作GPIO引脚，没有和其它功能复用.

□PB_MFP[13]	PB.13 Function

		0 GPIO
[12]	**GPB_MFP12**	**PB.12 Pin功能选择** T该引脚功能取决于 GPB_MFP12 和PB12_MFP1 (ALT_MFP[10]). **PB12_MFP1 (ALT_MFP[10])** / **GPB_MFP[12]** / **PB.12 Function** 0 / 0 / GPIO 0 / 1 / SPISS10 (SPI1) 1 / 0 / Reserved 1 / 1 / CLKO (C□ock Dri□er output)
[11]	**保留**	保留
[10]	**GPB_MFP10**	**PB.10 Pin功能选择** 该引脚功能取决于 GPB_MFP10 和 PB10_MFP1 (ALT_MFP[0]). **PB10_MFP1 (ALT_MFP□0])** / **GPB_MFP[10]** / **PB.10 Function** 0 / 0 / GPIO 0 / 1 / TM2 1 / 0 / Reserved 1 / 1 / SPISS01 (SPI0)
[9]	**GPB_MFP9**	**PB.9 Pin功能选择** 该引脚功能取决于 GPB_MFP9 和 PB9_MFP1 (ALT_MFP[1]). **PB9_MFP1 (ALT_MFP[1])** / **GPB_MFP[9]** / **PB.9 Function** 0 / 0 / GPIO 0 / 1 / TM1 1 / 0 / Reserved 1 / 1 / SPISS11 (SPI1
[8]	**GPB_MFP8**	**PB.8 Pin功能选择** GPB_MFP[8] 决定 PB.8 功能 **GPB_MFP[8]** / **PB.8 Function** 0 / GPIO 1 / TM0
[7]	**GPB_MFP7**	**PB.7 Pin功能选择** 该引脚功能取决于GPB_MFP7 和 PB7_MFP1 (ALT_MFP[16]). **PB7_MFP1** / **GPB_MFP[7]** / **PB.7 Function**

		(ALT_MFP[16])		
		0	0	GPIO
		0	1	CTS1 (UART1)
		1	0	Reserved
			1	MISO20 (SPI2)
[6]	**GPB_MFP6**	**PB.6 Pin功能选择** 该引脚功能取决于GPB_MFP6 和 PB6_MFP1 (ALT_MFP[17]).		
		PB6_MFP1 (ALT_MF□[17])	**GPB_MFP[6]**	**PB.6 F□nction**
		0	0	GPIO
		0	1	RTS1 (UART1)
		1	0	Reserved
		1	1	MOSI20 (SPI2)
[5]	**GPB_MFP5**	**PB. 5 Pin功能选择** 该引脚等取决于 GPB_MFP[5] 和 PB5_MFP1 (ALT_MFP[18])		
		PB5_MFP1 (ALT_MFP[18])	**GPB_MFP[5]**	**PB.5 Function**
		0	0	GPIO
		0	1	TXD1 (UA□T1)
		1	0	Reserved
		1	1	SPICLK2 (SPI2)
[4]	**GPB_MFP4**	**PB.4 Pin功能选择** 该引脚等取决于 GPB_MFP[4] 和PB4_MFP1 (ALT_MFP[15])		
		PB4_MFP1 (ALT_MFP[15])	**GPB_MFP[4]**	**PB.4 Function**
		0	0	GPIO
		0	1	RXD1 (UART1)
		1	0	SPISS20 (SPI2)
		1	1	SPISS11 (SPI1)
[3]	**GPB_MFP3**	**PB.3 Pin功能选择** 该引脚功能取决于GPB_MFP3 和 PB3_MFP1 (ALT_MFP[27]).		
		PB3_MFP1 (ALT_MFP[27])	**GPB_MFP[3]**	**PB.3 Function**
		0	0	GPIO
		0	1	CTS0 (UART0)

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>T3EX (TMR3)</td></tr>
</table>
</td></tr>
<tr><td>[2]</td><td>GPB_MFP2</td><td>
PB.2 Pin功能选择
该引脚功能取决于GPB_MFP2 和 PB2_MFP1 (ALT_MFP[26]).
<table>
<tr><th>PB2_MFP1 (ALT_MFP[26])</th><th>GPB_MFP[2]</th><th>PB.2 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>RTS0 (UART0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>T2EX (TMR2)</td></tr>
</table>
</td></tr>
<tr><td>[1]</td><td>GPB_MFP1</td><td>
PB.1 Pin功能选择
1 = PB.1作为UART0的TXD0
0 = PB.1作为GPIOB[1]
</td></tr>
<tr><td>[0]</td><td>GPB_MFP0</td><td>
PB.0 Pin功能选择
1 = PB.0作为UART0的 RXD0
0 = PB.0作为GPIOB[0]
</td></tr>
</table>

GPIOC多功能管脚控制寄存器(GPC_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPC_MFP	GCR_BA+0x38	R/W	GPIOC多功能与输入类型控制寄存器	0x0000_0000

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

Bits	描述	
[31:16]	GPC_TYPEn	1 = 使能GPIOC[15:0] I/O Schmitt触发输入 0 = 禁止GPIOC[15:0] I/OSchmitt触发输入
[15:14]	Reserved	Reserved
[13]	GPC_MFP13	**PC.13 Pin 功能选择** PC13_MFP1 (ALT_MFP[21]) 和 GPC_MFP[13] 决定 PC.13 的功能

PC13_MFP1 (ALT_MFP[21])	GPC_MFP[13]	PC.13 Function
0	0	GPIO
0	1	MOSI11 (SPI1)
1	0	CLKO (Cloc□Driver output)
	1	PWM3 (PWM)

[12]	GPC_MFP12	**PC.12 Pin 功能选择** PC12_MFP1 (ALT_MFP[20]) 和 GPC_MFP[12] 决定 PC.12 的功能

PC12_MFP1 (ALT_MFP[20])	GPC_MFP[12]	PC.12 Function
	0	GPIO
0	1	MISO11 (SPI1)
1	0	I2SMCLK
1	1	PWM2 (PWM)

[11]	**GPC_MFP11**	**PC.11 Pin 功能选择** 1 = PC.11作为**SPI1 MOSI0** (主机输出，从机输入引脚-0) 0 = PC.11作为GPIOC[11]
[10]	**GPC_MFP10**	**PC.10 Pin 功能选择** 1 = PC.10作为**SPI1 MISO0** (主机输入，从机输出引脚-0) 0 = PC.10作为GPIOC[10]
[9]	**GPC_MFP9**	**PC.9 Pin 功能选择** 1 = PC.9作为 **SPI1 SPCLK** 0 = PC.9作为GPIOC[9]
[8]	**GPC_MFP8**	**PC.8 Pin功能选择** 该引脚功能取决于 GPC_MFP[8]. **GPC_□FP[8]** / **PC.8 Function** 0 / □PIO 1 / □PISS10 (SPI1)
[7:6]	**保留**	保留
[5]	**GPC_MFP5**	**PC.5 Pin 功能选择** PC5_MFP1 (ALT_MFP[30]) 和 GPC_MFP[5] 决定 PC.5 的功能. **PC5_MFP1 (AL□_MFP[30])** / **GPC_MFP[5]** / **PC.5 Function** 0 / 0 / GPIO 0 / 1 / MOSI01 (SPI0) 1 / 0 / Reserved 1 / 1 / TXD0 (UART0)
[4]	**GPC_MFP4**	**PC.4 Pin 功能选择** PC4_MFP1 (ALT_MFP[29]) 和 GPC_MFP[4] 决定 PC.4 的功能 **PC4_MFP1 (ALT_MFP[29])** / **GPC_MFP[4]** / **PC.4 Function** 0 / 0 / GPIO 0 / 1 / MISO01 (SPI0) 1 / 0 / Reserved 1 / 1 / RXD0 (UART0)
[3]	**GPC_MFP3**	**PC.3 Pin功能选择** PC3_MFP1 (ALT_MFP[8]) 和 GPC_MFP[3] 决定 PC.3 的功能. **PC3_MFP1** / **GP□_MFP[3]** / **PC.3 Function**

		(ALT_MFP[8])		
		0	0	GPIO
		0		MOSI00 (SPI0)
		1	0	Reserved
		1	1	I2SDO (I^2S)
[2]	**GPC_MFP2**	**PC.2 Pin功能选择** PC2_MFP1 (ALT_MFP[7]) 和 GPC_MFP[2] 决定 PC.2 的功能.		
		PC2_MFP1 (ALT_MFP[7])	**GPC_MFP[2]**	**PC.2 Function**
		0	0	GP□O
		0	1	MISO00 (SP□0)
		1	0	Reserved
		1	1	I2SDI (I^2□)
[1]	**GPC_MFP1**	**PC.1 Pin功能选择** PC1_MFP1 (ALT_MFP[6]) 和 GPC_MFP[1] 决定 PC.1 的功能.		
		PC1_MFP1 (ALT_MFP[6])	**GPC_MFP[1]**	**PC.1 F□nction**
			0	GPIO
		0	1	SPICLK0 (SPI0)
		1	0	Reserved
		1	1	I2SBCLK (I^2S)
[0]	**GPC_MFP0**	**PC.0 Pin功能选择** PC0_MFP1 (ALT_MFP[5]) 和 GPC_MFP[0] 决定 PC.0 的功能.		
		PC0_MFP1 (ALT_MFP□5])	**GPC_MFP[0]**	**PC.0 Function**
		0	0	GPIO
		0	1	SPISS00 (SPI0)
		1	0	Reserved
		1	1	I2SLRCLK (I^2□)

GPIOD多功能控制寄存器(GPD_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPD_MFP	GCR_BA+0x3C	R/W	GPIOD多功能与输入类型控制寄存器	0x0000_0000

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

Bits	描述	
[31:16]	GPD_TYPEn	1 = 使能GPIOD[15:0] I/O Schmitt触发输入 0 = 禁止GPIOD[15:0] I/OSchmitt触发输入
[15:12]	保留	保留
[11]	GPD_MFP11	**PD.11 Pin功能选择** 1= PD.11作为外部中断/INT1 0= PD.11作为GPIOD[11]
[10]	GPD_MFP10	**PD.10 Pin功能选择** 1= PD.10作为CLKO 0= PD.10作为GPIOD[10]
[9]	GPD_MFP9	**PD.9 Pin功能选择** 这个引脚只作为GPIO用 GPD_MFP[9] = 0 → PD.9 Function = GPIO
[8]	GPD_MFP8	**PD.8 Pin功能选择** 1 = PD.8作为**SPI1 MISO10** (SPI1 主机输出，从机输入引脚-0) 0 = PD.8作为GPIOD[8]
[7:6]	Reserved	Reserved
[5]	GPD_MFP5	**PD.5 Pin功能选择** PD5_MFP1 (ALT_MFP1[21]) 和 GPD_MFP[5] 决定 PD.5 的功能.

		PD5_MFP1 (ALT_M□P1[21]) / **GPD_MFP[5]** / **PD.5 Function** 0 / 0 / GPIO 0 / 1 / Reserved 1 / 0 / MOSI21 (SPI2) 1 / 1 / ADC5
[4]	**GPD_MFP4**	**PD.4 Pin功能选择** PD4_MFP1 (ALT_MFP1[20]) 和 GPD_MFP[4] 决定 PD.4 的功能 **PD4_MFP1 (ALT_MFP1[20])** / **GPD_MFP[4]** / **PD.4 Function** 0 / 0 / GPIO 0 / 1 / Reserved 1 / 0 / MISO21 (SPI2) 1 / 1 / ADC4
[3]	**GPD_MFP3**	**PD.3 Pin功能选择** PD3_MFP1 (ALT_MFP1[19]) 和 GPD_MFP[3] 决定 PD.3 的功能 **PD3_MPF1 (ALT_MFP1□19])** / **GPD_MFP[3]** / **PD.3 Function** 0 / 0 / □PIO 0 / 1 / MOSI01 (SPI0) 1 / 0 / MOSI20 (SPI2) 1 / 1 / ADC3
[2]	**GPD_MFP2**	**PD.2 Pin功能选择** PD2_MFP1 (ALT_MFP1[18]) 和 GPD_MFP[2] 决定 PD.2 的功能 **PD2_MFP1 (ALT_MFP1[18])** / **GPD_MFP[2]** / **PD.2 Functio** 0 / 0 / GPIO 0 / 1 / MISO01 (SPI0) 1 / 0 / MISO20 (SPI2) 1 / 1 / ADC2
[1]	**GPD_MFP1**	**PD.1 Pin功能选择** PD1_MFP1 (ALT_MFP1[17]) 和 GPD_MFP[1] 决定 PD.1的功能 **PD1_MFP1 (ALT_MFP1[17])** / **GPD_MFP[1]** / **PD.1 F□□ction** 0 / 0 / □PIO

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>0</td><td>1</td><td>SPISS01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>SPICLK2 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC1</td></tr>
</table>
保留
</td></tr>
<tr><td>[0]</td><td>GPD_MFP0</td><td>
PD.0 Pin功能选择
PD0_MFP1 (ALT_MFP1[16]) 和 GPD_MFP[0] 决定 PD.0的功能
<table>
<tr><th>PD0_MFP1 (ALT_MFP1[16])</th><th>GPD_MFP[0]</th><th>PD.0 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>SPISS20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC0</td></tr>
</table>
</td></tr>
</table>

GPIOF多功能控制寄存器(GPF_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPF_MFP	GCR_BA+0x40	R/W	GPIOF多功能与输入类型控制寄存器	0x0000_000X

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留				GPF_TYPE[3:0]			
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留				GPF_MFP3	GPF_MFP2	GPF_MFP1	GPF_MFP0

Bits	描述	
[31:20]	保留	保留
[19:16]	GPF_TYPEn	1 = 使能 GPIOF[3:0] I/O Schmitt触发输入 0 = 禁止 GPIOF[3:0] I/O Schmitt触发输入
[15:4]	保留	保留
[3]	GPF_MFP3	**PF.5 Pin功能选择** PF3_MFP1 (ALT_MFP1[27:26]) 和 GPF_MFP[3] 决定 PF.3 的功能. 该比特的缺省值为1
[2]	GPF_MFP2	**PF.2 Pin 功能选择** PF2_MFP1 (ALT_MFP1[25:24]) 和 GPF_MFP[2] 决定 PF.2 的功能. 该比特的缺省值为1

PF3_MFP1 (ALT_MFP1[27:26])	GPF_MFP[3]	PF.3 Funct□n
00	0	GPI
00	1	PS2CLK
01	1	保留
10	1	I2C0SCL(I^{2}C0)
11	1	ADC7

PF2_MFP1 (ALT_MFP1[25:24])	GPF_MFP[2]	PF.2 Function

		00	0	GPIO
		00	1	PS2DAT
		01	1	保留
		10	1	I2C0SDA
		11	1	ADC6
[1]	**GPF_MFP1**	**PF.1 Pin功能选择** 该比特的缺省值由Config0的CFOSC决定。如果CFOSC=000，该比特的缺省值为1，如果CFOSC=111，该比特的缺省值是0 1 = PF.1作为XT1_IN 0 = PF.1作为GPIOF[1]		
[0]	**GPF_MFP0**	**PF.0 Pin 功能选择** 该比特的缺省值由Config0的CFOSC决定。如果CFOSC=000，该比特的缺省值为1，如果CFOSC=111，该比特的缺省值是0 1 = PF.0作为XT1_OUT 0 = PF.0作为GPIOF[0]		

复用多功能管脚控制寄存器(ALT_MFP)

寄存器	偏移量	R/W	描述	复位后的值
ALT_MFP	GCR_BA+0x50	R/W	复用多功能管脚控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留	PC5_MFP1	PC4_MFP1	保留	PB3_MFP1	PB2_MFP1	保留	PB15_MFP1
23	22	21	20	19	18	17	16
保留		PC13_MFP1	PC12_MFP1	保留	PB5_MFP1	PB6_MFP1	PB7_MFP1
15	14	13	12	11	10	9	8
PB4_MFP1	保留		PA10_MFP1	PA11_MFP1	PB12_MFP1	PA15_MFP1	PC3_MFP1
7	6	5	4	3	2	1	0
PC2_MFP1	PC1_MFP1	PC0_MFP1	保留			PB9_MFP1	PB10_MFP1

<table>
<tr><th>Bits</th><th colspan="4">描述</th></tr>
<tr><td>[31]</td><td>保留</td><td colspan="3">保留</td></tr>
<tr><td rowspan="6">[30]</td><td rowspan="6">PC5_MFP1</td><td colspan="3">PC.5 Pin 功能选择
PC5_MFP1 (ALT_MFP[30]) 和 GPC_MFP[5] 决定 PC.5 的功能.</td></tr>
<tr><td>PC5_MFP1 (ALT_MFP[30])</td><td>GPC_MFP[5]</td><td>PC.5 Function</td></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td></td><td>MOSI01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>TXD0 (UART0)</td></tr>
<tr><td rowspan="6">[29]</td><td rowspan="6">PC4_MFP1</td><td colspan="3">PC.4 Pin功能选择
PC4_MFP1 (ALT_MFP[29]) 和GPC_MFP[4] 决定 PC.4的功能.</td></tr>
<tr><td>PC4_MFP1 (ALT_MFP[29])</td><td>GPC_MFP[4]</td><td>PC.4 Function</td></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>RXD0 (UART0)</td></tr>
<tr><td>[28]</td><td>保留</td><td colspan="3">保留</td></tr>
</table>

<table>
<tr><td>[27]</td><td>PB3_MFP1</td><td>PB.3 Pin功能选择
该引脚的功能取决于 GPB_MFP[3] 和 PB3_MFP1 (ALT_MFP[27]).
<table><tr><th>PB3_MFP1 (ALT_MFP[27])</th><th>GPB_MFP[3]</th><th>PB.3 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>CTS0 (UART□)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>T3EX (TM□3)</td></tr></table></td></tr>
<tr><td>[26]</td><td>PB2_MFP1</td><td>PB.2 Pin功能选择
该引脚的功能取决于GPB_MFP[2] 和 PB2_MFP1 (ALT_MFP[26]).
<table><tr><th>PB2_MFP1 (ALT_MFP[26])</th><th>GPB_MFP[2]</th><th>PB.2 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>RTS□(UART□)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>T2EX (TMR2)</td></tr></table></td></tr>
<tr><td>[25]</td><td>保留</td><td>保留</td></tr>
<tr><td>[24]</td><td>PB15_MFP1</td><td>PB.15 Pin 功能选择
PB15_MFP1 (ALT_MFP[24]) 和 GPB_MFP[15] 决定PB.15的功能.
<table><tr><th>PB15_MFP1 (ALT_MFP[24])</th><th>GPB_MFP[15]</th><th>PB.15 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>/INT1</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>T0EX (TMR0)</td></tr></table></td></tr>
<tr><td>[23:22]</td><td>保留</td><td>保留</td></tr>
</table>

<table>
<tr><td>[21]</td><td>PC13_MFP1</td><td>PC.13 Pin 功能选择
PC13_MFP1 (ALT_MFP[21]) 和 GPC_MFP[13] 决定 PC.13的功能.
<table><tr><th>PC13_MFP1 (ALT_MFP[21])</th><th>GPC_MFP[13]</th><th>PC.13 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>MOSI11 (SPI1)</td></tr><tr><td>1</td><td>0</td><td>CLKO (Clock□Driver output)</td></tr><tr><td>1</td><td>1</td><td>PWM3 (PWM)</td></tr></table></td></tr>
<tr><td>[20]</td><td>PC12_MFP1</td><td>PC.12 Pin功能选择
PC12_MFP1 (ALT_MFP[20]) 和 GPC_MFP[12] 决定PC.12的功能.
<table><tr><th>PC12_MFP1 (ALT_MFP[20])</th><th>GPC_MFP[12]</th><th>PC.12 Function</th></tr><tr><td>0</td><td></td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>MISO11 (SPI1)</td></tr><tr><td>1</td><td>0</td><td>I2SMCLK (I^{2}S)</td></tr><tr><td>1</td><td>1</td><td>PWM2 (PWM)</td></tr></table></td></tr>
<tr><td>[19]</td><td>保留</td><td>保留</td></tr>
<tr><td>[18]</td><td>PB5_MFP1</td><td>PB. 5 Pin 功能选择
该引脚的功能取决于GPB_MFP[5] 和 PB5_MFP1 (ALT_MFP[18]).
<table><tr><th>PB5_MFP1 (ALT_M□P[18])</th><th>GPB_MFP[5]</th><th>PB.5 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>TXD1 (UART1)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>SPICLK2 (SPI2)</td></tr></table></td></tr>
<tr><td>[17]</td><td>PB6_MFP1</td><td>PB.6 Pin 功能选择
该引脚的功能取决于GPB_MFP[6] 和 PB6_MFP1 (ALT_MFP[17]).
<table><tr><th>PB6_MFP1 (ALT_MFP[17])</th><th>G□B_MFP[6]</th><th>PB.6 Function</th></tr><tr><td>0</td><td>0</td><td>GPI</td></tr><tr><td>0</td><td>1</td><td>RTS1 (UART1)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td></td><td>1</td><td>MOSI20 (SPI2)</td></tr></table></td></tr>
</table>

<table>
<tr><td>[16]</td><td>PB7_MFP1</td><td>PB.7 Pin 功能选择
该引脚的功能取决于GPB_MFP[7] 和 PB7_MFP1 (ALT_MFP[16]).
<table><tr><th>PB7_MFP1
(ALT_MFP[16])</th><th>GPB_MFP[7]</th><th>PB.7 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>CTS1 (UART1)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>MISO20 (SPI2)</td></tr></table></td></tr>
<tr><td>[15]</td><td>PB4_MFP1</td><td>PB.4 Pin 功能选择
该引脚的功能取决于GPB_MFP[4] 和 PB4_MFP1 (ALT_MFP[15]).
<table><tr><th>PB4_MFP1
(ALT_MFP[15])</th><th>GPB_MFP[4]</th><th>PB.4 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td></td><td>1</td><td>RXD1 (UART1)</td></tr><tr><td>1</td><td>0</td><td>SPISS20 (SPI2)</td></tr><tr><td>1</td><td>1</td><td>SPISS11 (SPI1)</td></tr></table></td></tr>
<tr><td>[14:13]</td><td>保留</td><td>保留</td></tr>
<tr><td>[12]</td><td>PA10_MFP1</td><td>PA.10 Pin 功能选择
该引脚的功能取决于GPA_MFP[10] 和 PA10_MFP1 (ALT_MFP[12]).
<table><tr><th>PA10_MFP1
(A□T_MFP[□2])</th><th>GPA_MFP[10]</th><th>PA.10 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>I2C1SDA (I^{2}C1)</td></tr><tr><td>1</td><td>0</td><td>MISO10 (SPI1)</td></tr><tr><td>1</td><td></td><td>MISO20 (SPI2)</td></tr></table></td></tr>
<tr><td>[11]</td><td>PA11_MFP1</td><td>PA.11 Pin 功能选择
该引脚的功能取决于GPA_MFP[11] 和 PA11_MFP1 (ALT_MFP[11]).
<table><tr><th>PA11_MFP1
(ALT_MFP[11])</th><th>GPA_MFP[□1]</th><th>PA.11 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>I2C1SCL (I^{2}C)</td></tr><tr><td>1</td><td>0</td><td>SPICLK1 (SPI1)</td></tr><tr><td>1</td><td>1</td><td>MOSI20 (SPI2)</td></tr></table></td></tr>
</table>

<table>
<tr><td>[10]</td><td>PB12_MFP1</td><td>PB.12 Pin 功能选择
该引脚的功能取决于GPB_MFP[12] 和 PB12_MFP1 (ALT_MFP[10]).
<table><tr><th>PB12_MFP1 (ALT_MFP[10])</th><th>GPB_MFP[12]</th><th>PB.12 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td></td><td>SPISS10 (SPI1)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td></td><td>1</td><td>CLKO (Clock Driver output)</td></tr></table></td></tr>
<tr><td>[9]</td><td>PA15_MFP1</td><td>PA.15 Pin 功能选择
该引脚的功能取决于GPA_MFP[15] 和 PA15_MFP1 (ALT_MFP[9]).
<table><tr><th>PA15_MFP1 (ALT_MFP[9])</th><th>GPA_MFP[15]</th><th>PA.15 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>PWM3 (PWM)</td></tr><tr><td>1</td><td>0</td><td>CLKO (Clock Driver output)</td></tr><tr><td>1</td><td>1</td><td>I2SMCLK (I²S)</td></tr></table></td></tr>
<tr><td>[8]</td><td>PC3_MFP1</td><td>PC.3 Pin 功能选择
PC3_MFP1 (ALT_MFP[8]) 和 GPC_MFP[3] 决定 PC.3的功能.
<table><tr><th>PC3_MFP1 (ALT_MFP[8])</th><th>GPC_MFP[3]</th><th>PC.3 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td></td><td>MOSI00 (SPI0)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>I2SDO (I²S)</td></tr></table></td></tr>
<tr><td>[7]</td><td>PC2_MFP1</td><td>PC.2 Pin 功能选择
PC2_MFP1 (ALT_MFP[7]) 和 GPC_MFP[2] 决定PC.2的功能.
<table><tr><th>PC2_MFP1 (ALT_MFP[7])</th><th>GPC_MFP[2]</th><th>PC.2 Function</th></tr><tr><td>0</td><td></td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>MISO00 (SPI0)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>I2SDI (I²S)</td></tr></table></td></tr>
</table>

<table>
<tr><td>[6]</td><td>PC1_MFP1</td><td>PC.1 Pin 功能选择
PC1_MFP1 (ALT_MFP[6]) 和 GPC_MFP[1] 决定PC.1的功能.
<table><tr><th>PC1_MFP1 (ALT_MFP[6])</th><th>GPC_MFP[1]</th><th>PC.1 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>SPICLK0 (SPI0)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>I2SBCLK (I²S)</td></tr></table></td></tr>
<tr><td>[5]</td><td>PC0_MFP1</td><td>PC.0 Pin 功能选择
PC0_MFP1 (ALT_MFP[5]) 和 GPC_MFP[0] 决定PC.0的功能.
<table><tr><th>PC0_MFP1 (ALT_MFP[5]</th><th>GPC_MFP[0]</th><th>PC.0 Fu□ction</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>SPISS00 (SPI0)</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>I2SLRCLK (I²□)</td></tr></table></td></tr>
<tr><td>[4:2]</td><td>保留</td><td>保留</td></tr>
<tr><td>[1]</td><td>PB9_MFP1</td><td>PB.9 Pin 功能选择
该引脚的功能取决于GPB_MFP[9] 和 PB9_MFP1 (ALT_MFP[1]).
<table><tr><th>PB9_MFP1 (ALT_MFP[1])</th><th>GPB_MFP[9]</th><th>PB.9 Funct□on</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>TM1</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>SPISS11 (SPI□)</td></tr></table></td></tr>
<tr><td>[0]</td><td>PB10_MFP1</td><td>PB.10 Pin 功能选择
该引脚的功能取决于 GPB_MFP[10] 和 PB10_MFP1 (ALT_MFP[0]).
<table><tr><th>PB10_MFP1 (ALT_MFP[0])</th><th>GPB_□FP[10]</th><th>PB.10 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td></td><td>TM2</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>SPISS01 (SPI0)</td></tr></table></td></tr>
</table>

复用多功能管脚控制寄存器1 (ALT_MFP1)

寄存器	偏移	R/W	描述	复位后的值
ALT_MFP1	GCR_BA+0x54	R/W	复用多功能管脚控制寄存器1	0x0000_0000

31	30	29	28	27	26	25	24
保留				PF3_MFP1		PF2_MFP1	
23	22	21	20	19	18	17	16
保留		PD5_MFP1	PD4_MFP1	PD3_MFP1	PD2_MFP1	PD1_MFP1	PD0_MFP1
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留							

Bits	描述	
[31:28]	保留	保留
[27:26]	PF3_MFP1	**PF.3 Pin 功能选择** PF3_MFP1 (ALT_MFP1[27:26]) 和 GPF_MFP[3] 决定 PF.3的功能.

PF3_MFP1 (ALT_MFP1[2□:26])	GPF_MFP[3]	PF.3 F□nction
00	0	GPIO
00	1	PS2CLK
01	1	Reserved
10	1	I2C0SCL
11	1	ADC7

[25:24]	PF2_MFP1	**PF.2 Pin 功能选择** PF2_MFP1 (ALT_MFP1[25:24]) 和 GPF_MFP[2] 决定PF.2的功能.

PF2_MFP1 (ALT_MFP1[25:24])	GPF_MFP[2]	PF.2 Function
00	0	GPIO
00	1	PS2DAT
01	1	Reserevd
10	1	I2C0SDA
11	1	ADC6

<table>
<tr><td>[23:22]</td><td>保留</td><td>保留</td></tr>
<tr><td>[21]</td><td>PD5_MFP1</td><td>PD.5 Pin 功能选择
PD5_MFP1 (ALT_MFP1[21]) 和 GPD_MFP[5] 决定PD.5的功能.
<table>
<tr><th>PD5_MFP1 (ALT_M□P1[21])</th><th>GPD_MFP[5]</th><th>PD.5 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>MOSI21 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC5</td></tr>
</table></td></tr>
<tr><td>[20]</td><td>PD4_MFP1</td><td>PD.4 Pin 功能选择
PD4_MFP1 (ALT_MFP1[20]) 和 GPD_MFP[4] 决定PD.4的功能.
<table>
<tr><th>PD4_MFP1 (ALT_MFP1[20])</th><th>GPD_MFP[4]</th><th>PD.4 Function</th></tr>
<tr><td>0</td><td>0</td><td>□PIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>MISO21 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC4</td></tr>
</table></td></tr>
<tr><td>[19]</td><td>PD3_MFP1</td><td>PD.3 Pin 功能选择
PD3_MFP1 (ALT_MFP1[19]) 和 GPD_MFP[3] 决定PD.3的功能.
<table>
<tr><th>PD3_MPF1 (ALT□MFP1[19])</th><th>GPD_MFP[3]</th><th>PD.3 Funct□on</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>MOSI20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC3</td></tr>
</table></td></tr>
<tr><td>[18]</td><td>PD2_MFP1</td><td>PD.2 Pin 功能选择
PD2_MFP1 (ALT_MFP1[18]) 和 GPD_MFP[2] 决定PD.2的功能.
<table>
<tr><th>PD2_MFP1 (ALT_MFP1[18])</th><th>GPD_MFP[2]</th><th>PD.2 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>MISO20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC2</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[17]</td><td>PD1_MFP1</td><td>PD.1 Pin 功能选择
PD1_MFP1 (ALT_MFP1[17]) 和 GPD_MFP[1] 决定 PD.1的功能.
<table><tr><th>PD1_MFP1 (AL□_MFP1[17])</th><th>GPD_MFP[1]</th><th>PD.1 Functio</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>SPISS01 (SPI0)</td></tr><tr><td>1</td><td>0</td><td>SPICLK2 (SPI2)</td></tr><tr><td>1</td><td>1</td><td>ADC1</td></tr></table></td></tr>
<tr><td>[16]</td><td>PD0_MFP1</td><td>PD.0 Pin 功能选择
PD0_MFP1 (ALT_MFP1[16]) 和 GPD_MFP[0] 决定 PD.0 的功能.
<table><tr><th>PD0_MFP1 (ALT_MFP1[16□)</th><th>GPD_MFP[0]</th><th>PD.0 Function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>SPISS20 (SPI2)</td></tr><tr><td>1</td><td>1</td><td>ADC0</td></tr></table></td></tr>
<tr><td>[15:0]</td><td>保留</td><td>保留</td></tr>
</table>

GPIOA IO 控制寄存器 (GPA_IOCR)

寄存器	偏移	R/W	描述	复位后的值
GPA_IOCR	GCR_BA+0xC0	R/W	GPIOA IO 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				GPA11_DS	GPA10_DS	保留	
7	6	5	4	3	2	1	0
保留							

Bits	描述	
[31:12]	保留	保留
[11]	GPA11_DS	**PA.11 引脚驱动强度选择** 1 = PA.11 使能强驱动能力. 0 = PA.11 关闭强驱动能力.
[10]	GPA10_DS	**PA.10 引脚驱动强度选择** 1 = PA.10使能强驱动能力. 0 = PA.10关闭强驱动能力.
[9:0]	保留	保留

GPIOB IO控制寄存器(GPB_IOCR)

寄存器	偏移	R/W	描述	复位后的值
GPB_IOCR	GCR_BA+0xC4	R/W	GPIOB IO 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留	GPB14_DS	GPB13_DS	GPB12_DS	Reserved			GPB8_DS
7	6	5	4	3	2	1	0
GPB7_DS	GPB6_DS	GPB5_DS	GPB4_DS	保留			

Bits	Description	
[31:15]	保留	保留
[14]	GPB14_DS	**PB.14 引脚驱动强度选择** 1 = PB.14使能强驱动能力. 0 = PB.14关闭强驱动能力.
[13]	GPB13_DS	**PB.13 引脚驱动强度选择** 1 = PB.13使能强驱动能力. 0 = PB.13关闭强驱动能力.
[12]	GPB12_DS	**PB.12 引脚驱动强度选择** 1 = PB.12使能强驱动能力. 0 = PB.12关闭强驱动能力.
[11:9]	保留	保留
[8]	GPB8_DS	**PB.8 引脚驱动强度选择** 1 = PB.8使能强驱动能力. 0 = PB.8关闭强驱动能力.
[7]	GPB7_DS	**PB.7 引脚驱动强度选择** 1 = PB.7使能强驱动能力. 0 = PB.7关闭强驱动能力.
[6]	GPB6_DS	**PB.6引脚驱动强度选择** 1 = PB.6使能强驱动能力. 0 = PB.6关闭强驱动能力.

[5]	**GPB5_DS**	**PB.5引脚驱动强度选择** 1 = PB.5使能强驱动能力. 0 = PB.5关闭强驱动能力.
[4]	**GPB4_DS**	**PB.4引脚驱动强度选择** 1 = PB.4使能强驱动能力. 0 = PB.4关闭强驱动能力.
[3:0]	**保留**	保留

GPIOD IO 控制寄存器(GPD_IOCR)

寄存器	偏移	R/W	描述	复位后的值
GPD_IOCR	GCR_BA+0xCC	R/W	GPIOD IO 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				GPD11_DS	GPD10_DS	GPD9_DS	GPD8_DS
7	6	5	4	3	2	1	0
保留							

Bits	描述	
[31:12]	保留	保留
[11]	GPD11_DS	**PD.11引脚驱动强度选择** 1 = PD.11使能强驱动能力. 0 = PD.11关闭强驱动能力.
[10]	GPD10_DS	**PD.10引脚驱动强度选择** 1 = PD.10使能强驱动能力. 0 = PD.10关闭强驱动能力.
[9]	GPD9_DS	**PD.9引脚驱动强度选择** 1 = PD.9使能强驱动能力. 0 = PD.9关闭强驱动能力.
[8]	GPD8_DS	**PD.8引脚驱动强度选择** 1 = PD.8使能强驱动能力. 0 = PD.8关闭强驱动能力.
[7:0]	保留	保留

寄存器写保护控制寄存器(REGWRPROT)

有些系统控制寄存器需要被保护起来，以防止误操作而影响芯片运行，这些寄存器在上电复位到用户解锁之前是锁定的。用户可以连续依次写入“59h”, “16h” “88h”到寄存器REGWRPROT （地址：0x5000_0100）解锁定。在这三个数据之间写入任何其他数据，不同时序或写入其他地址都会中止整个时序，导致无法解锁.

解锁后，用户可以检测解锁指示位：0x5000_0100 的bit0，“1“表示已经解锁定，”0“表示锁定。用户可以更新目标寄存器的值，向“0x5000_0100”写入任何值，就可以重锁保护寄存器.

该位寄存器用于禁止/使能保护寄存器，读取得到REGPROTDIS状态

寄存器	偏移量	R/W	描述	复位后的值
REGWRPROT	GCR_BA+0x100	R/W	寄存器写保护控制寄存器	0x0000_0000

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

Bits	描述	
[31:16]	保留	保留
[7:0]	**REGWRPROT**	**寄存器写保护码**(Write Only) 写保护寄存器可以通过写入“59h”, “16h”, “88h” 解除锁定状态. 这个时序完成之后, REGPROTDIS 位将被置1，写保护寄存器可以正常写入数据.
[0]	**REGPROTDIS**	**寄存器写保护**(Read only) 1 = 解锁定以写入受保护的寄存器 0 = 锁定受保护的寄存器，不能向受保护寄存器写入数据. 受保护的寄存器有: **IPRSTC1**: 地址 0x5000_0008 **BODCR**: 地址0x5000_0018 **PORCR**: 地址0x5000_0024 **PWRCON**: 地址0x5000_0200 (在电源唤醒中断被清时，bit[6]不受保护)

		APBCLK bit[0]: 地址0x5000_0208 (bit[0] 是看门狗时钟使能) **CLKSEL0**: 地址0x5000_0210 (选择 HCLK 与CPU STCLK 时钟源) **CLKSEL1 bit[1:0]**: 地址0x5000_0214 (用于看门狗时钟源选择) **ISPCON**: 地址0x5000_C000 (Flash ISP 控制寄存器) **WTCR**: 地址0x4000_4000 **FATCON**: 地址0x5000_C018

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围设备时钟。该控制器还通过单独的时钟关或开，时钟源选择和分频器来进行功耗控制。CPU使能PWR_DOWN_EN位后，Cortex-M0 内核执行WFI指令，芯片将进入掉电模式。等唤醒中断发生，将退出掉电模式。在掉电模式下，时钟控制器关闭外部 4~24 MHz 高速晶振和内部 22.1184 MHz 高速振荡器，以降低整个系统的功耗。

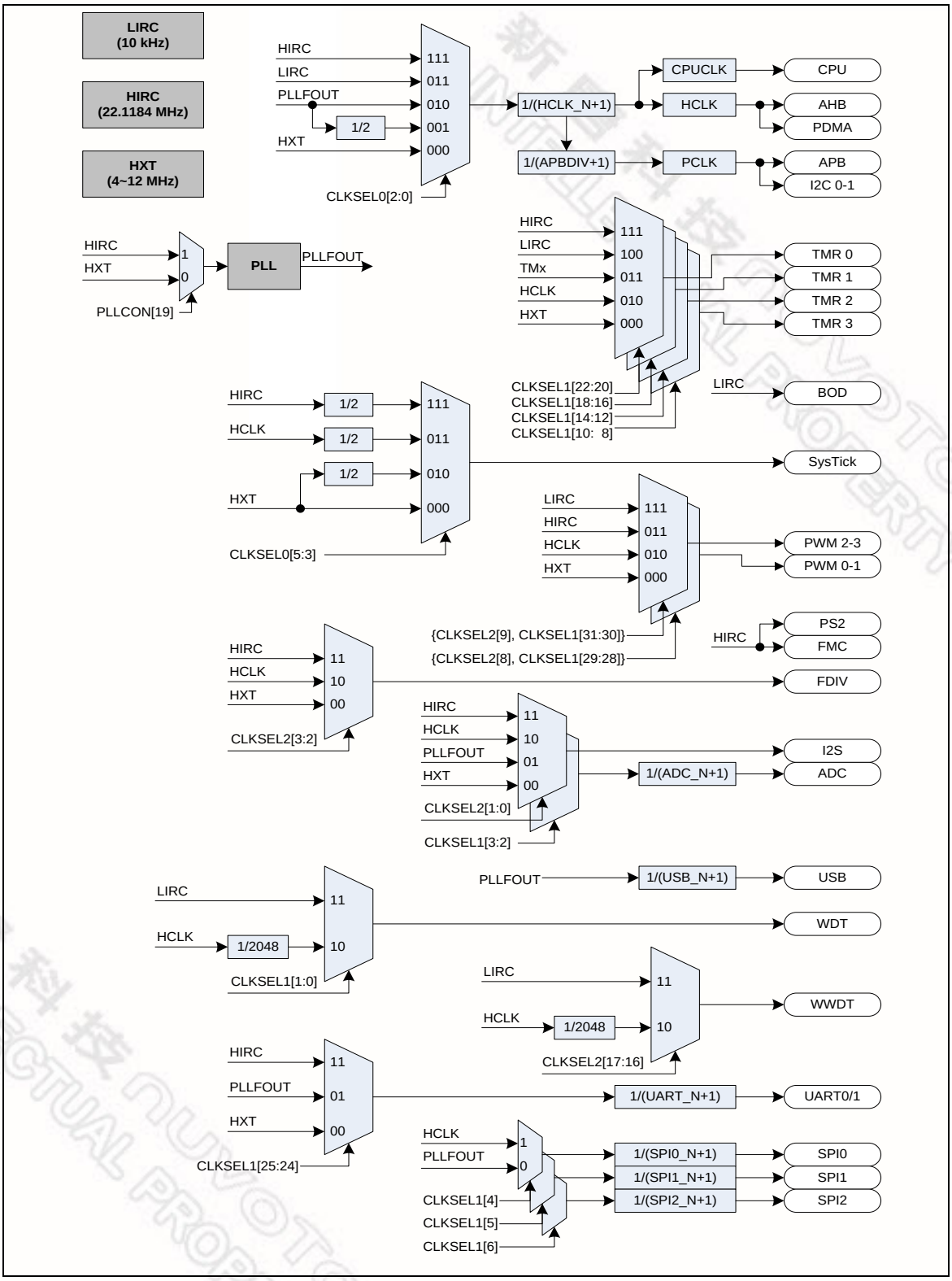


图 5-2 时钟发生器全局框图

5.4.2 时钟发生器

时钟发生器由如下4个时钟源组成:

- 一个外部 4~24 MHz 高速晶振
- 一个可编程的 PLL FOUT(PLL 由外部 4~24 MHz 高速晶振和内部 22.1184 MHz 高速振荡器提供时钟源)
- 一个内部 22.1184 MHz 高速振荡器
- 一个内部 10 KHz 低速振荡器

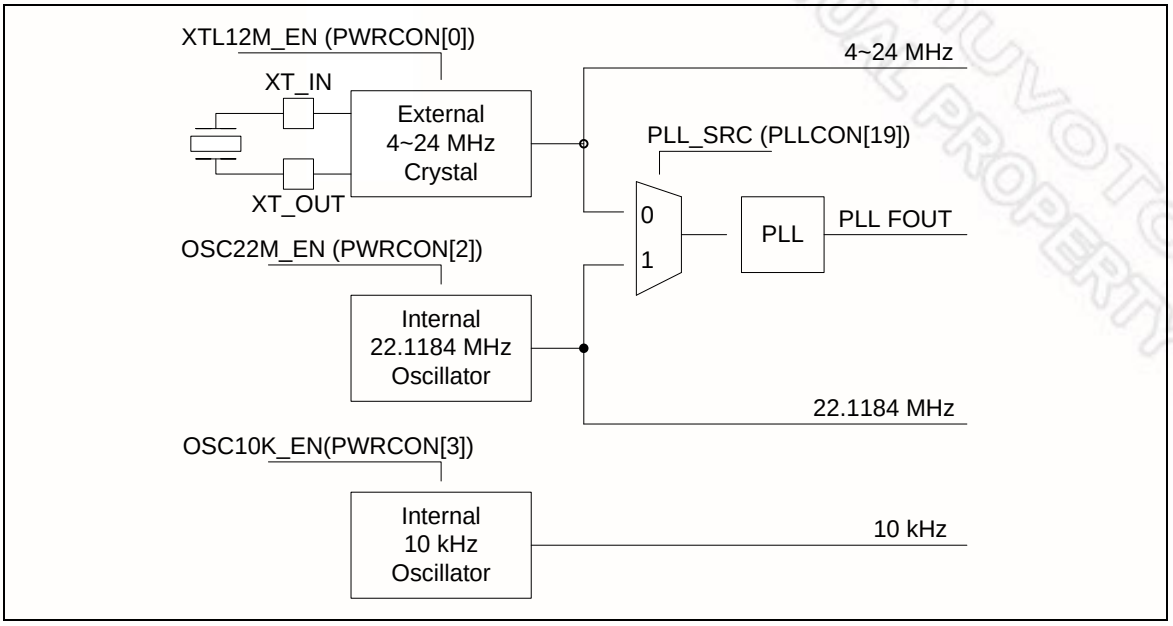


图 5-3 时钟发生器框图

5.4.3 系统时钟 和 SysTick 时钟

系统时钟有5个时钟源可选，由时钟发生器产生。时钟源切换取决于寄存器HCLK_S(CLKSEL0[2:0]), 如图 5-44所示。

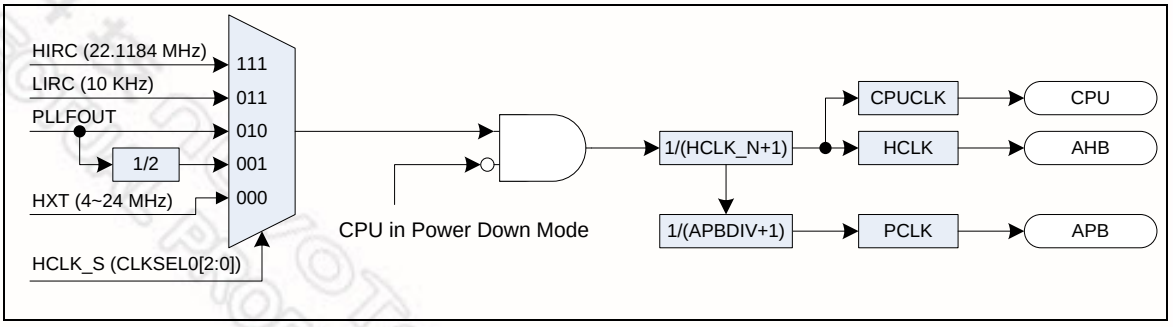


图 5-4 系统时钟框图

Cortex-M0内核的SysTick 时钟源可以选择CPU时钟或外部时钟(SYST_CSR[2]).如果使用外部时

钟，SysTick 时钟 (STCLK) 有5个时钟源。时钟源切换取决于寄存器 STCLK_S(CLKSEL0[5:3]. 框图如图 5-55.

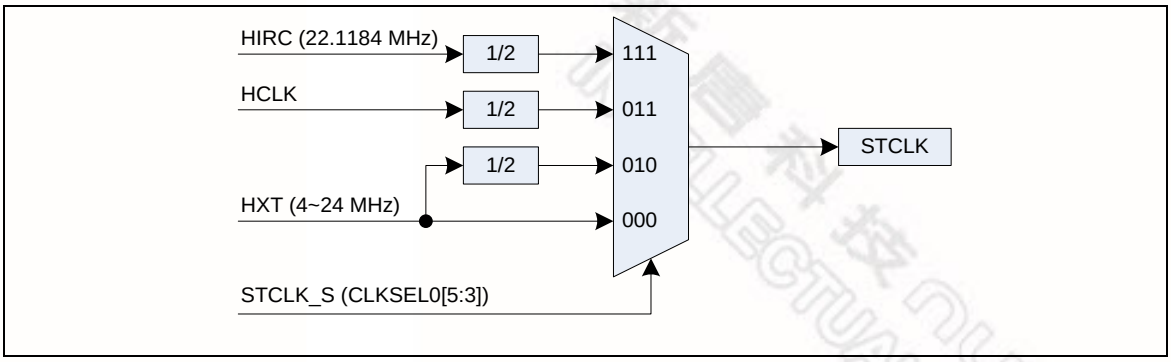


图 5-5 SysTick 时钟控制框图

5.4.4 外围设备时钟

不同的外设，其外围设备时钟有不同的时钟源可切换. 请参阅5.4.8节寄存器CLKSEL1 & CLKSEL2。

5.4.5 掉电模式时钟

当芯片进入掉电模式后，一些时钟源、外设时钟和系统时钟被关闭，也有一些时钟源与外设时钟仍在工作。

如下时钟仍在工作：

- 时钟发生器
  - 内部 10 KHz 低速振荡器时钟
- 外设时钟 (当这些外设采用内部10 KHz低速振荡器作为时钟源时)

5.4.6 分频器输出

该芯片包含一个由16级2分频移位寄存器组成的分频器。其中哪一级的值被输出由一个16选1的多路转换器选择，该多路转换器接到CLKO引脚上。因此有16种分频时钟选择，频率从 $F_{in}/2^{1}$ 到 $F_{in}/2^{16}$，其中 Fin 为输入到时钟分频器的时钟频率

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

当DIVIDER_EN (FRQDIV[4])置1时, 分级计数器开始计数。当DIVIDER_EN (FRQDIV[4])置0时，分级计数器持续计数直到分频时钟达到低电平并保持低电平.

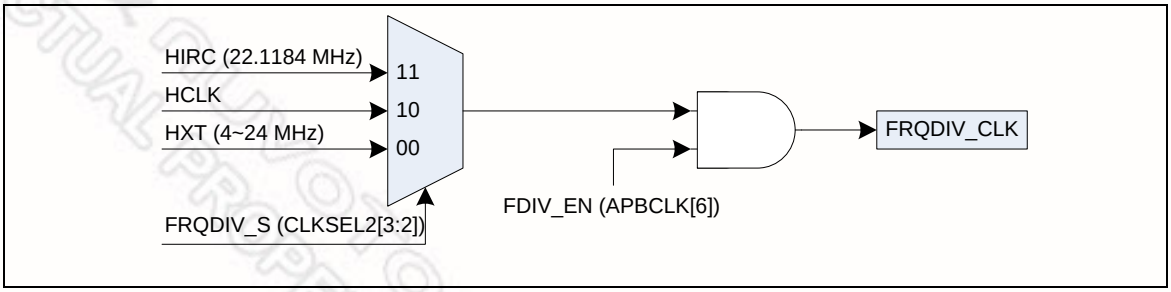


图 5-6 分频器的时钟源

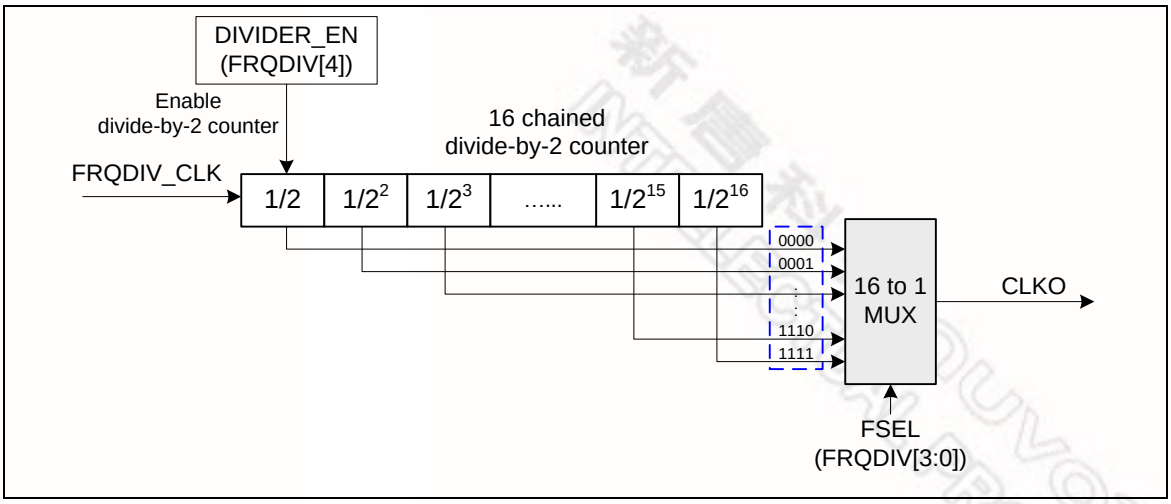

DIVIDER_EN
(FRQDIV[4])
Enable
divide-by-2 counter
16 chained
divide-by-2 counter
FRQDIV_CLK
1/2
1/2^2
1/2^3
.....
1/2^15
1/2^16
0000
0001
1110
1111
16 to 1
MUX
CLKO
FSEL
(FRQDIV[3:0])

图 5-7 分频器的框图

5.4.7 寄存器映射

R: read only, **W:** write only, **R/W:** both read and write

寄存器	偏移量	R/W	描述	复位后的值
CLK 基地址: **CLK_BA = 0x5000_0200**				
PWRCON	CLK_BA+0x00	R/W	系统掉电控制寄存器	0x0000_001X
AHBCLK	CLK_BA+0x04	R/W	AHB 设备时钟使能控制寄存器	0x0000_000D
APBCLK	CLK_BA+0x08	R/W	APB 设备时钟使能控制寄存器	0x0000_000X
CLKSTATUS	CLK_BA+0x0C	R/W	时钟状态监控寄存器(Low Density Only)	0x0000_00XX
CLKSEL0	CLK_BA+0x10	R/W	时钟源选择控制寄存器0	0x0000_003X
CLKSEL1	CLK_BA+0x14	R/W	时钟源选择控制寄存器1	0xFFFF_FFFF
CLKSEL2	CLK_BA+0x1C	R/W	时钟源选择控制寄存器2	0x0000_00FF
CLKDIV	CLK_BA+0x18	R/W	时钟分频寄存器	0x0000_0000
PLLCON	CLK_BA+0x20	R/W	PLL控制寄存器	0x0005_C22E
FRQDIV	CLK_BA+0x24	R/W	分频器控制寄存器	0x0000_0000
APBDIV	CLK_BA+0x2C	R/W	APB 分频控制寄存器	0x0000_0000

5.4.8 寄存器描述

掉电控制寄存器（PWRCON）

除BIT[6]外，PWRCON的其他位都是写保护的，解锁这些位，需要向地址0x5000_0100依次写入“59h”, “16h”, “88h”. 参考寄存器REGWRPROT，其地址是GCR_BA + 0x100

寄存器	偏移量	R/W	描述	复位后的值
PWRCON	CLK_BA+0x00	R/W	系统掉电控制寄存器	0x0000_001X

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

Bits	描述	
[31:9]	保留	保留
[8]	PD_WAIT_CPU	**控制进入掉电模式的条件** (写保护位) 1 =在 PD_WAIT_CPU 和 PWR_DOWN_EN 位都置 1 并且 CPU 执行 WFI 指令时，芯片进入掉电模式。 0 = 在PWR_DOWN_EN 置1时，芯片进入掉电模式
[7]	PWR_DOWN_EN	**系统掉电模式使能位** (写保护位) 该位置 “1”，使能芯片的掉电模式，激活芯片的掉电依赖于PD_WAIT_CPU 位 (a) 如果PD_WAIT_CPU 为 “0”,在置位PWR_DOWN_EN 后，芯片立即进入掉电. (b) 如果PD_WAIT_CPU 为 “1”, 直到CPU的休眠模式有效时,芯片仍在运行，然后才掉电 芯片由掉电唤醒，该位自动清零，用户需要重新设置该位才能使能下一次的掉电。 掉电模式下, 外部 4~24 MHz 高速晶振与内部 22.1184 MHz 高速振荡器被禁止，但内部 10 KHz 低速振荡器的使能不受该位控制 掉电时, PLL 与系统时钟被禁, 忽略时钟源选择. 如果外设时钟源选择10 kHz时钟，则外设的时钟不受掉电模式的控制. 1 = 芯片立即进入掉电模式 或等待CPU休眠命令WFI 0 = 执行WFI，芯片工作于正常模式或芯片进入idle mode(空闲模式)
[6]	PD_WU_STS	**芯片掉电唤醒状态标志** 设置“掉电唤醒”, 从掉电模式恢复

		GPIO, USB, UART, WDT, ACMP, BOD 被唤醒，该标志置位 写 1 清该位.
[5]	**PD_WU_INT_EN**	**掉电模式唤醒的中断使能** (写保护位) 0 = 禁止 1 = 使能 当PD_WU_STS 和 PD_WU_INT_EN 都为高时，产生中断.
[4]	**PD_WU_DLY**	**使能唤醒延时计数器** (写保护位) 芯片由掉电模式唤醒时，时钟控制会延迟一定时钟周期以等待系统时钟稳定. 当芯片工作在外部 4~24 MHz 高速晶振的条件下，延迟时钟周期为 4096 时钟周期, 当芯片工作在内部22.1184 MHz 时，延迟256 时钟周期. 1 = 使能时钟周期的延迟 0 = 禁止时钟周期的延迟
[3]	**OSC10K_EN**	**内部 10 KHz 低速振荡器控制** (写保护位) 1 = 使能内部 10 KHz 低速振荡器 0 = 禁止内部 10 KHz 低速振荡器
[2]	**OSC22M_EN**	**内部 22.1184 MHz 高速振荡器控制** (写保护位) 1 = 使能内部 22.1184 MHz 高速振荡器 0 = 禁止内部 22.1184 MHz 高速振荡器
[1]	**保留**	保留
[0]	**XTL12M_EN**	**外部 4~24 MHz 高速晶振控制**(写保护位) 该位的缺省值由flash控制器设置，用户配置寄存器config0 [26:24]. 当缺省时钟源为4~24 MHz晶振. 该位自动置1 1 = 使能外部 4~24 MHz 高速晶振 0 = 禁止外部 4~24 MHz 高速晶振

寄存器/指令 模式	PWR_DOWN_EN	PD_WAIT_CPU	CPU run WFI 指令	禁止时钟
正常运行模式	0	0	NO	通过控制寄存器禁止所有时钟
空闲模式 (CPU**进入休眠模式**)	0	0	YES	仅禁止CPU时钟
掉电模式	1	0	NO	大部分时钟停止运行，仅外部10K/及WDT/Timer/PWM可以运行.
掉电模式 (CPU**进入深度休眠模式**)	1	1	YES	大部分时钟停止运行，仅外部10K及WDT/Timer/PWM可以运行.

表 5-5 掉电模式控制表

当芯片进入掉电模式时，通过某些中断源用户可以将芯片唤醒。在置位

PWR_DOWN_EN(PWRCON[7])比特之前，用户应该使能相应的外设中断和NVIC IRQ中断使能位(NVIC_ISER)，以确保芯片进入掉电模式之后能被成功唤醒。

AHB 设备时钟使能控制寄存器 （AHBCLK）

该寄存器各位用于使能/禁止系统时钟, PDMA时钟.

寄存器	偏移量	R/W	描述	复位后的值
AHBCLK	CLK_BA+0x04	R/W	AHB设备时钟使能控制寄存器	0x0000_0001

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

Bits	描述	
[31:3]	保留	保留
[2]	ISP_EN	**Flash ISP控制器时钟使能控制** 1 = 使能 Flash ISP engine 时钟. 0 = 禁止 Flash ISP engine 时钟
[1]	PDMA_EN	**PDMA 控制器时钟使能控制.** 1 = 使能 PDMA engine 时钟. 0 = 禁止 PDMA engine 时钟
[0]	保留	保留

APB 设备时钟使能控制寄存器（APBCLK）

该寄存器各位用于使能/禁止外设控制器时钟.

寄存器	偏移量	R/W	描述	复位后的值
APBCLK	CLK_BA+0x08	R/W	APB 设备时钟使能控制寄存器	0x0000_000X

31	30	29	28	27	26	25	24
PS2_EN	保留	I2S_EN	ADC_EN	USBD_EN	保留		
23	22	21	20	19	18	17	16
保留		PWM23_EN	PWM01_EN	保留		UART1_EN	UART0_EN
15	14	13	12	11	10	9	8
保留	SPI2_EN	SPI1_EN	SPI0_EN	保留		I2C1_EN	I2C0_EN
7	6	5	4	3	2	1	0
保留	FDIV_EN	TMR3_EN	TMR2_EN	TMR1_EN	TMR0_EN	保留	WDT_EN

Bits	描述	
[31]	PS2_EN	**PS/2时钟使能控制** 1 = 使能 PS/2 时钟 0 = 禁止 PS/2 时钟
[30]	保留	保留
[29]	I2S_EN	**I^2S 时钟使能控制** 1 = 使能I^2S时钟 0 = 禁止I^2S时钟
[28]	ADC_EN	**ADC时钟使能控制** 1 = 使能 ADC 时钟 0 = 禁止 ADC 时钟
[27]	USBD_EN	**USB 2.0 FS 设备控制器时钟使能控制** 1 = 使能 USB 时钟 0 = 禁止 USB 时钟
[26:22]	保留	保留
[21]	PWM23_EN	**PWM_23时钟使能控制** 1 = 使能 PWM23 时钟 0 = 禁止 PWM23 时钟
[20]	PWM01_EN	**PWM_01时钟使能控制**

		1 = 使能 PWM01 时钟 0 = 禁止 PWM01 时钟
[19:18]	**保留**	保留
[17]	**UART1_EN**	**UART1时钟使能控制** 1 = 使能 UART1 时钟 0 = 禁止 UART1 时钟
[16]	**UART0_EN**	**UART0时钟使能控制** 1 = 使能 UART0 时钟 0 = 禁止 UART0 时钟
[15]	**保留**	保留
[14]	**SPI2_EN**	**SPI2时钟使能控制** (Medium Density Only) 1 = 使能 SPI2 时钟 0 = 禁止 SPI2 时钟
[13]	**SPI1_EN**	**SPI1时钟使能控制** 1 = 使能 SPI1 时钟 0 = 禁止 SPI1 时钟
[12]	**SPI0_EN**	**SPI0时钟使能控制** 1 = 使能 SPI0 时钟 0 = 禁止 SPI0 时钟
[11:10]	**保留**	保留
[9]	**I2C1_EN**	**I^2C1时钟使能控制** 1 = 使能 I^2C1 时钟 0 = 禁止 I^2C1 时钟
[8]	**I2C0_EN**	**I^2C0时钟使能控制** 1 = 使能 I^2C0 时钟 0 = 禁止 I^2C0 时钟
[7]	**保留**	保留
[6]	**FDIV_EN**	**分频器输出时钟使能控制** 1 = 使能 FDIV 时钟 0 = 禁止 FDIV 时钟
[5]	**TMR3_EN**	**Timer3时钟使能控制** 1 = 使能 Timer3 时钟 0 = 禁止 Timer3 时钟
[4]	**TMR2_EN**	**Timer2时钟使能控制** 1 = 使能 Timer2 时钟

		0 = 禁止 Timer2 时钟
[3]	**TMR1_EN**	**Timer1时钟使能控制** 1 = 使能 Timer1 时钟 0 = 禁止 Timer1 时钟
[2]	**TMR0_EN**	**Timer0时钟使能控制** 1 = 使能 Timer0 时钟 0 = 禁止 Timer0 时钟
[1]	**保留**	保留
[0]	**WDT_EN**	**看门狗定时器时钟使能控制** (写保护位) 该位是受保护的位，编程时，需要向0x5000_0100 依次写入“59h”, “16h”, “88h” 来解锁定，参考寄存器 REGWRPROT （地址GCR_BA+0x100）. 缺省值由flash控制设置，用户可配置寄存器congig0 bit[31] 1 = 使能看门狗定时器时钟 0 = 禁止看门狗定时器时钟

时钟状态寄存器（CLKSTATUS）

该寄存器各位用于监控芯片时钟源是否稳定，时钟切换是否失败

寄存器	偏移量	R/W	描述	复位后的值
CLKSTATUS	CLK_BA+0x0C	R/W	时钟状态监控寄存器	0x0000_00XX

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

Bits	描述	
[31:8]	保留	保留
[7]	CLK_SW_FAIL	**时钟切换失败标志** (写保护位) 1 = 时钟切换失败 0 = 时钟切换成功 该位在软件切换系统时钟源时更新时，如果切换的目标时钟稳定，该位设置为1'b0. 如果切换目标时钟不稳定，该位设置为1. 该位写1清零.
[6:5]	保留	保留
[4]	OSC22M_STB	**OSC22M时钟源稳定标志** 1 = OSC22M 时钟稳定 0 = OSC22M 时钟不稳定或没有使能 该位只读
[3]	OSC10K_STB	**OSC10K时钟源稳定标志** 1 = OSC10K 时钟稳定 0 = OSC10K 时钟不稳定或没有使能 该位只读
[2]	PLL_STB	**PLL 时钟源稳定标志** 1 = PLL 时钟稳定

		0 = PLL 时钟不稳定或没有使能 该位只读
[1]	**保留**	保留
[0]	**XTL12M_STB**	**XTL12M 时钟源稳定标志** 1 = XTL12M 时钟稳定 0 = XTL12M 时钟不稳定或没有使能 该位只读

时钟源选择控制寄存器0 （CLKSEL0）

寄存器	偏移量	R/W	描述	复位后的值
CLKSEL0	CLK_BA+0x10	R/W	时钟源选择控制寄存器0	0xFFFF_FFFX

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

Bits	描述	
[31:6]	**保留**	保留
[5:3]	**STCLK_S**	**Cortex_M0 SysTick 时钟源选择** (写保护位) 如果 SYST_CSR[2]=0, SysTick 使用下表的时钟源 该位受保护, 修改该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100. 000 = 外部 4~24 MHz 高速晶振 010 = 外部 4~24 MHz 高速晶振/2分频 011 = HCLK/2分频 111 = 内部 22.1184 MHz 高速振荡器/2分频
[2:0]	**HCLK_S**	**HCLK时钟源选择** (写保护位) 1. 时钟切换之前，相关时钟源(预选和新选) 必须都打开 2. 任何复位后，加载用户配置寄存器CFOSC(Config0[26:24])的值，因而缺省值为 000b 或 111b. 3. 该位受保护, 修改该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”, 参考寄存器 REGWRPROT,地址GCR_BA + 0x100. 000 = 外部 4~24 MHz 高速晶振 001 = PLL 时钟/2 010 = PLL 时钟 011 = 内部 10 KHz 低速振荡器 111 = 内部 22.1184 MHz 高速振荡器

时钟源选择控制寄存器1（CLKSEL1）

在时钟切换之前，相关的时钟源(预选和新选) 必须都打开.

寄存器	偏移量	R/W	描述	复位后的值
CLKSEL1	CLK_BA+0x14	R/W	时钟源选择控制寄存器1	0xFFFF_FFFF

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

Bits	描述	
[31:30]	**PWM23_S**	**PWM2与 PWM3的时钟源选择** PWM2与 PWM3使用相同的时钟源和相同的预分频 PWM2 和 PWM3 的时钟源由 PWM23_S[2:0] 决定，这个域由 CLKSEL2[9] 和 CLKSEL1[31:30]组合而成 000 = 外部 4~24 MHz 高速晶振 010 = HCLK 011 =内部 22.1184 MHz 高速振荡器 111 = 内部 10 KHz 低速振荡器
[29:28]	**PWM01_S**	**PWM0与 PWM1的时钟源选择** PWM0与 PWM1使用相同的时钟源和相同的分频 PWM0 和 PWM1 的时钟源由 PWM01_S[2:0] 决定，这个域由 CLKSEL2[8] 和 CLKSEL1[29:28]组合而成 000 = 外部 4~24 MHz 高速晶振 010 = HCLK 011 =内部 22.1184 MHz 高速振荡器 111 = 内部 10 KHz 低速振荡器
[27:26]	**保留**	保留
[25:24]	**UART_S**	**UART 时钟源选择** 00 = 外部 4~24 MHz 高速晶振 01 = PLL

		11 = 内部 22.1184 MHz 高速振荡器
[23]	**保留**	保留
[22:20]	**TMR3_S**	**TIMER3 时钟源选择** 000 = 外部 4~24 MHz 高速晶振 010 = HCLK 011 = 保留 100 =内部 10KHz 低速振荡器 111 = 内部 22.1184 MHz 高速振荡器
[19]	**保留**	保留
[18:16]	**TMR2_S**	**TIMER2时钟源选择** 000 = 外部 4~24 MHz 高速晶振 010 = HCLK 011 = 保留 100 =内部 10KHz 低速振荡器 111 = 内部 22.1184 MHz 高速振荡器
[15]	**保留**	保留
[14:12]	**TMR1_S**	**TIMER1时钟源选择** 000 = 外部 4~24 MHz 高速晶振 010 = HCLK 011 = 保留 100 =内部 10KHz 低速振荡器 111 = 内部 22.1184 MHz 高速振荡器
[11]	**保留**	保留
[10:8]	**TMR0_S**	**TIMER0时钟源选择** 000 = 外部 4~24 MHz 高速晶振 010 = HCLK 011 = 保留 100 =内部 10KHz 低速振荡器 111 = 内部 22.1184 MHz 高速振荡器
[7]	**保留**	保留
[6]	**SPI2_S**	**SPI2 时钟源选择** 1 = HCLK. 0 = PLL.
[5]	**SPI1_S**	**SPI1 时钟源选择** 1 = HCLK. 0 = PLL

[4]	**SPI0_S**	**SPI0 时钟源选择** 1 = HCLK. 0 = PLL.
[3:2]	**ADC_S**	**ADC时钟源选择** 00 = 外部 4~24 MHz 高速晶振 01 = PLL 10 = HCLK 11 = 内部 22.1184 MHz 高速振荡器
[1:0]	**WDT_S**	**看门狗定时器时钟源选择** (写保护位) 该位受保护, 修改该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”, 参考寄存器REGWRPROT,地址GCR_BA + 0x100. 10 = HCLK/2048 clock 11 = 内部 10 KHz 低速振荡器

时钟源选择控制寄存器2 （CLKSEL2）

在时钟切换前，相关的时钟源 (预选和新选) 必须都打开.

寄存器	偏移量	R/W	描述	复位后的值
CLKSEL2	CLK_BA+0x1C	R/W	时钟源选择控制寄存器2	0x0000_00FF

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留						WWDT_S	
15	14	13	12	11	10	9	8
保留						PWM23_S[2]	PWM01_S[2]
7	6	5	4	3	2	1	0
保留				FRQDIV_S		I2S_S	

Bits	描述	
[31:18]	保留	保留
[17:16]	WWDT_S	Windowed-Watchdog 定时器时钟源选择 (写保护位) 10 = HCLK/2048. 11 = 内部10 kHz 低速振荡器.
[15:10]	保留	保留
[9]	PWM23_S[2]	PWM2 和 PWM3 时钟源选择 Bit [2] PWM2 和 PWM3 使用相同的时钟源和预分频器. PWM2 和 PWM3 的时钟源由 PWM23_S[2:0] 决定，这个域由 CLKSEL2[9] 和 CLKSEL1[31:30]组合而成. 000 = 外部 4~24 MHz 高速晶振. 010 = HCLK. 011 = 内部 22.1184 MHz 高速振荡器. 111 = 内部 10 kHz 低速振荡器.

[8]	**PWM01_S[2]**	**PWM0 and PWM1 时钟源选择 Bit [2]** PWM0 和 PWM1使用相同的时钟源和预分频器. PWM0 和 PWM1 的时钟源由 PWM01_S[2:0] 决定，这个域由 CLKSEL2[8] 和 CLKSEL1[29:28]组合而成. 000 = 外部 4~24 MHz 高速晶振. 010 = HCLK. 011 = 内部 22.1184 MHz高速振荡器. 111 = 内部 10 kHz 低速振荡器.
[7:4]	**保留**	保留
[3:2]	**FRQDIV_S**	**时钟分频器时钟源选择** 00 = 外部 4~24 MHz 高速晶振 01 = 保留 10 = HCLK 11 = 内部 22.1184 MHz 高速振荡器
[1:0]	**I2S_S**	**I^2S时钟源选择** 00 = 外部 4~24 MHz 高速晶振 01 = PLL 10 = HCLK 11 = 内部 22.1184 MHz 高速振荡器

时钟分频寄存器(CLKDIV)

寄存器	偏移量	R/W	描述	复位后的值
CLKDIV	CLK_BA+0x18	R/W	时钟分频寄存器	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:16]	ADC_N	**ADC 时钟源的时钟除频数** ADC 时钟频率 = (ADC时钟源频率) / (ADC_N + 1)
[15:12]	保留	保留
[11:8]	UART_N	**UART时钟源的时钟除频数** UART 时钟频率 = (UART 时钟源频率) / (UART_N + 1)
[7:4]	USB_N	**USB 时钟由PLL时钟除频数** USB时钟频率= (PLL 频率) / (USB_N + 1)
[3:0]	HCLK_N	**HCLK时钟源的时钟除频数** HCLK 时钟频率 = (HCLK 时钟源频率) / (HCLK_N + 1)

PLL控制寄存器（PLLCON）

PLL的参考时钟源来自外部 4~24 MHz 高速晶振或内部 22.1184 MHz 高速振荡器。该寄存器用于控制PLL的输出频率和PLL的操作模式

寄存器	偏移量	R/W	描述	复位后的值
PLLCON	CLK_BA+0x20	R/W	PLL控制寄存器	0x0005_C22E

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

Bits	描述	
[31:20]	保留	保留
[19]	PLL_SRC	**PLL 时钟源选择** 1 = PLL 时钟源为内部 22.1184 MHz 高速振荡器 0 = PLL时钟源为外部 4~24 MHz 高速晶振
[18]	OE	**PLL OE (FOUT enable) 引脚控制** 0 = 使能PLL FOUT 1 = PLL FOUT 固定为低
[17]	BP	**PLL旁路控制** 0 = PLL 正常模式 (default) 1 = PLL输出时钟频率与输入时钟相同(XTALin)
[16]	PD	**掉电模式** 如果设置PWRCON寄存器的 PWR_DOWN_EN 位为1, PLL将进入掉电模式 0 = PLL正常模式 1 = PLL掉电模式(default)
[15:14]	OUT_DV	**PLL输出分频控制引脚** 参考下表公式
[13:9]	IN_DV	**PLL输入分频控制引脚** 参考下表公式

[8:0]	**FB_DV**	**PLL 反馈分频控制引脚** 参考下表公式

输出时钟频率设置

$$FOUT = FIN \times \frac{NF}{NR} \times \frac{1}{NO}$$

约束条件:

1. $3.2MHz < FIN < 150MHz$
2. $800KHz < \frac{FIN}{2 * NR} < 8MHz$
3. $100MHz < FCO = FIN \times \frac{NF}{NR} < 200MHz$
   $120MHz < FCO$ is preferred

符号	说明
FOUT	输出时钟频率
FIN	输入 (参考) 时钟频率
NR	输入分频(IN_DV + 2)
NF	反馈分频(FB_DV + 2)
NO	OUT_DV = “00” : NO = 1 OUT_DV = “01” : NO = 2 OUT_DV = “10” : NO = 2 OUT_DV = “11” : NO = 4

默认频率设置

默认值: 0xC22E
FIN = 12 MHz
NR = (1+2) = 3
NF = (46+2) = 48
NO = 4
FOUT = 12/4 x 48 x 1/3 = 48 MHz

频率分频器控制寄存器(FRQDIV)

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

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

Bits	描述	
[31:5]	**保留**	保留
[4]	**DIVIDER_EN**	**频率分频器使能位** 0 = 关闭频率分频 1 = 使能频率分频
[3:0]	**FSEL**	**分频器输出频率选择位** 输出频率的公式 $F_{out} = F_{in}/2^{(N+1)}$ F_{in}为输入时钟频率. F_{out}为分频器输出时钟频率. N为4位FSEL[3:0]的值.

APB 除频控制寄存器 (FRQDIV)

Register	Offset	R/W	Description	Reset Value
APBDIV	CLK_BA+ 2C	R/W	APB 时钟除频控制寄存器	0x0000_0000

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

Bits	描述	
[31:1]	保留	保留
[0]	APBDIV	APB 时钟除频控制 0: PCLK = HCLK 1: PCLK = HCLK / 2

5.5 FLASH 内存控制器 (FMC)

5.5.1 概述

NuMicro™ NUC123 具有68/36k 字节的片上FLASH，用于存储应用程序（APROM）和数据(数据flash),用户可以通过ISP更新这些FLASH. 在系统编程 (ISP) 允许用户更新焊接在PCB板上的芯片中的程序. 上电后，Config0的启动选择(CBS)决定 Cortex-M0 CPU 从APROM或LDROM读取代码. NuMicro™ NUC123 也提供了数据 flash ，在芯片掉电之前，用来存储数据.

NuMicro™ NUC123 还有另外一个灵活的特性：数据Flash的大小可以配置。数据Flash的大小由数据flash 大小可变使能位 (DFVSEN), Config0中的数据flash 使能位 (DFEN)和Config1中的数据Flash基地址 (DFBADR)决定. DFVSEN 等于 1时, 数据flash 大小固定为4K，起始地址为0x0001_f000, 并且APROM 的大小变成64/32K；DFVSEN 等于 0，DFEN 等于 1时, 数据flash 的大小为0，APROM 的大小为 68/36K 字节；DFVSEN 等于 0，DFEN 也等于 0时，APROM 和数据flash 共享68/36K 字节空间，数据Flash的起始地址由Config1 (DFBADR) 决定

5.5.2 特性

- 连续地址读访问零等待状态时，最高可达50 MHz，否则可达72M
- 68/36 KB 应用程序内存 (APROM) 和数据Flash
- 4KB 在系统编程 (ISP)加载程序内存(LDROM)
- 大小可配置的数据FLASH，带有512字节页擦除单元
- 在系统编程 (ISP) 用于更新片上FLASH

5.5.3 框图

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

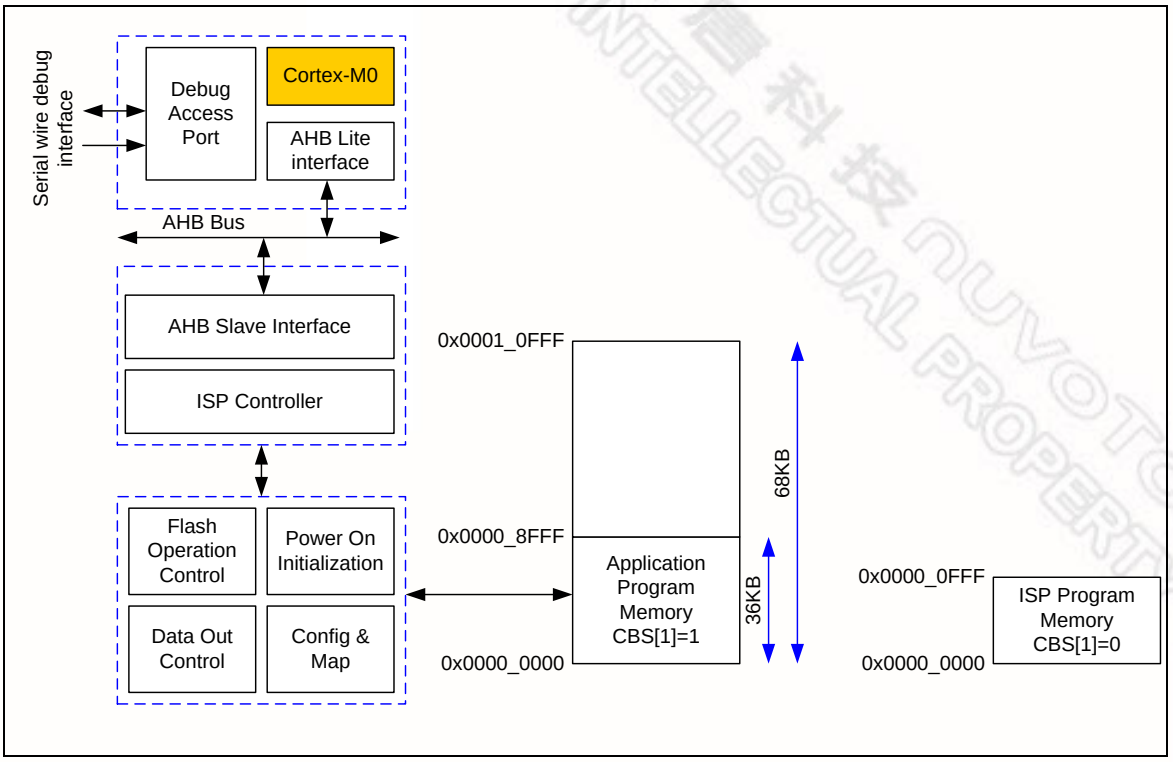


图 5-8 Flash 内存控制框图

5.5.4 Flash 内存结构

NuMicro™ NUC123 flash 内存由程序内存（64/32kB APROM），数据FLASH，ISP内存（LDROM），以及用户配置区域组成. 用户配置区域提供2个字控制系统逻辑，如flash安全加密，启动选择，欠压电平，数据FLASH基地址等. 在上电期间，由FLASH内存加载到相应的控制寄存器中，在芯片上PCB之前，用户可以用烧写器来设置这些位，.数据FLASH其大小为4KB，开始地址为0x0001_F000.

Block Name	Size	Start Address	End Address
AP-ROM	64 KB 32 KB	0x0000_0000	0x0000_FFFF (64 KB) 0x0000_7FFF (32 KB)
Data Flash	4 KB	0x0001_F000	0x0001_FFFF
LD-ROM	4 KB	0x0010_0000	0x0010_0FFF
User Configuration	2 words	0x0030_0000	0x0030_0004

表 5-6 内存地址映射 (DFVSEN = 1)

Block Name	Size	Start Address	End Address
AP-ROM	(36-0.5*N)/(68-0.5*N) KB	0x0000_0000	0x0001_0FFF (68KB, if DFEN=1) 0x0000_8FFF (36KB, if DFEN=1) DFBADR-1 (if DFEN=0)
Data Flash	0.5*N KB (if DFEN=0)	DFBADR (if DFEN=0)	0x0001_0FFF (68KB, if DFEN=0) 0x0000_8FFF (36KB, if DFEN=0)
LD-ROM	4 KB	0x0010_0000	0x0010_0FFF
User Configuration	2 words	0x0030_0000	0x0030_0004

注意: N 是被配置的数据Flash的页号. 一页为 512 bytes

表 5-7 内存地址映射 (DFVSEN = 0)

Flash 内存结构如下所示:

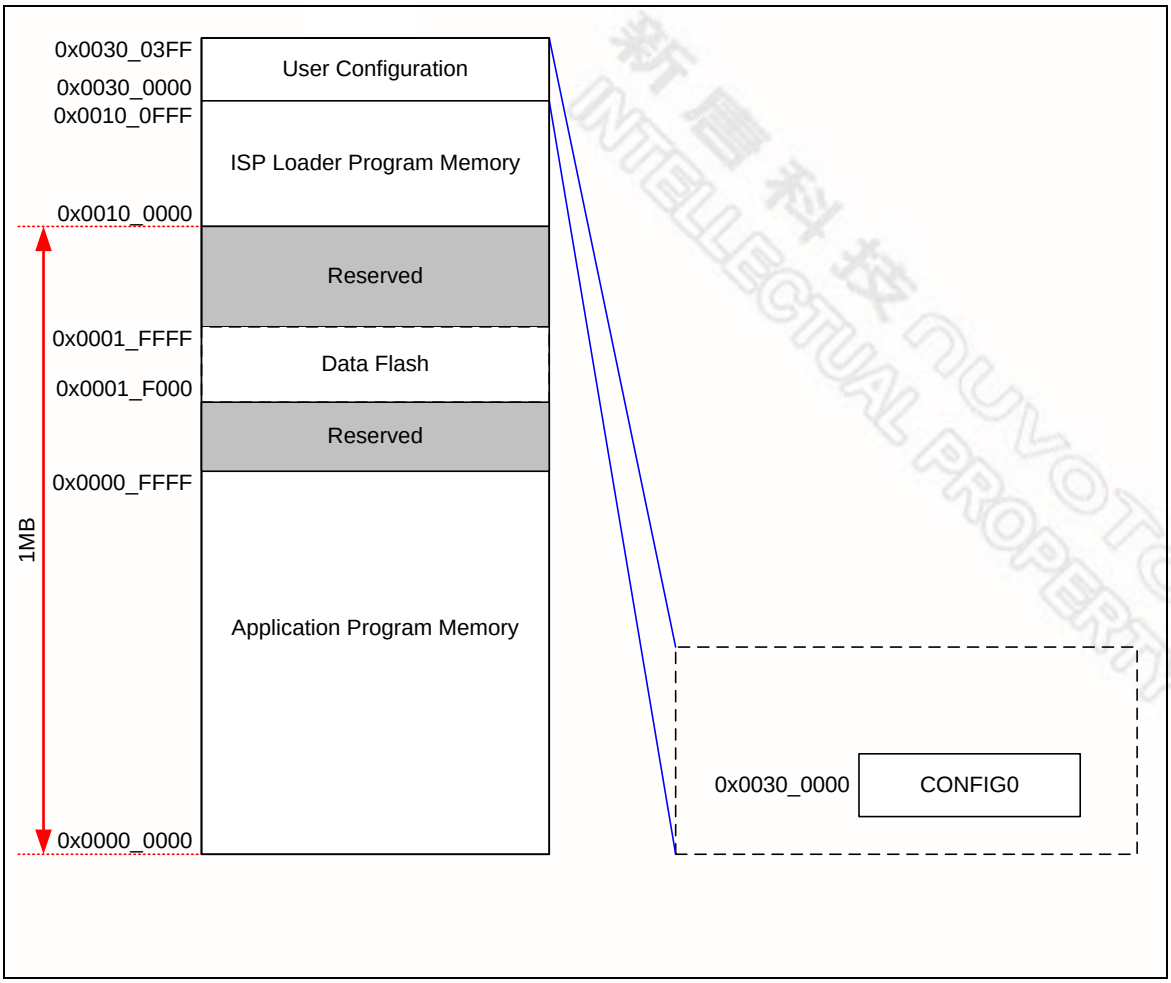


图 5-9 Flash 内存结构 (DFVSEN = 1)

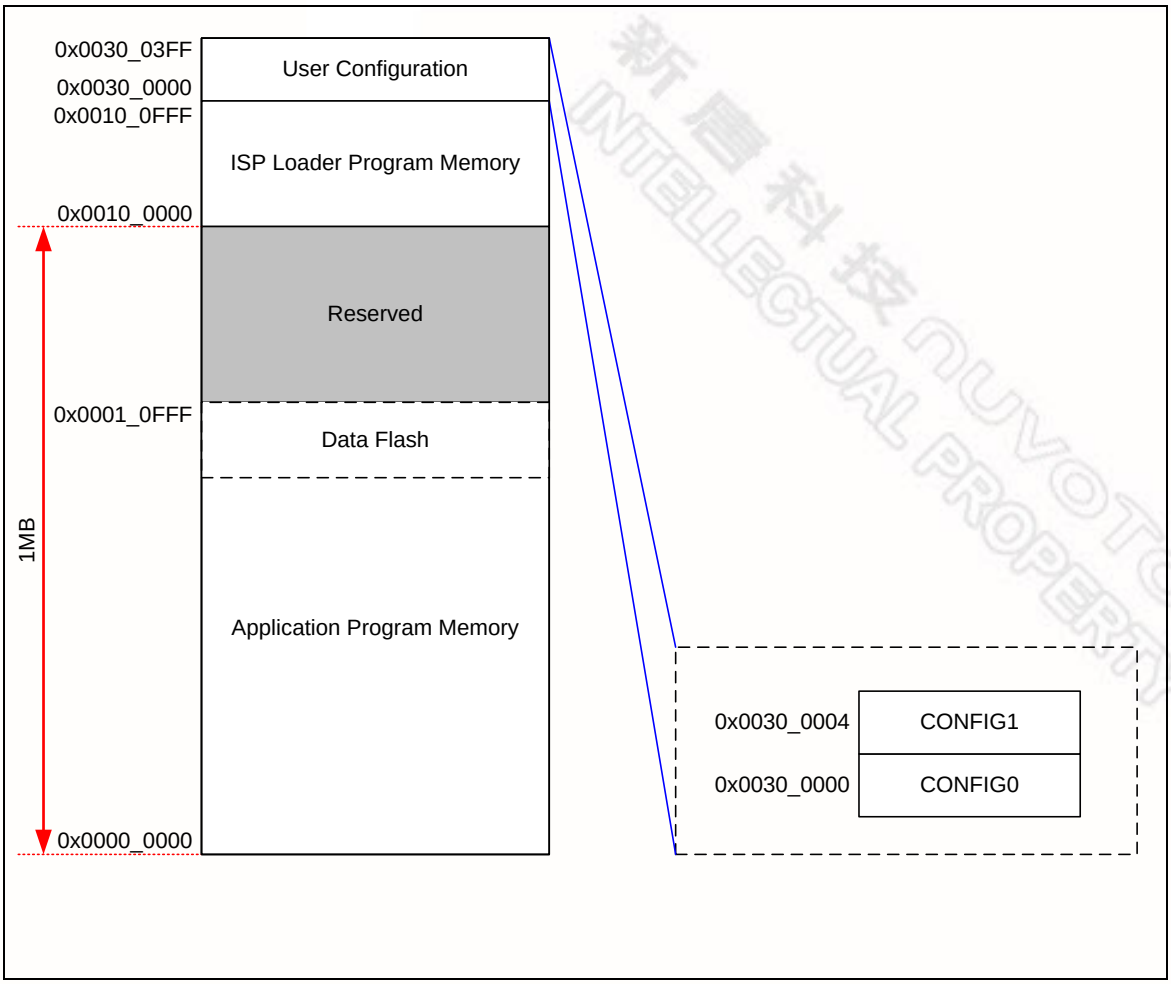


图 5-10 Flash 内存结构 (DFVSEN = 0)

5.5.5 启动选择

NuMicro™ NUC123 提供在系统编程 (ISP) 特征，允许用户直接更新PCB板上芯片中的程序. 提供4kB加载程序内存用于存储ISP固件. 用户可以通过设置Config0中的CBS位来选择从APROM还是从LDROM取指令来运行.

NuMicro™ NUC123 系列提供在系统编程 (ISP) 特性使客户通过单独的ISP固件可以更新程序空间。一个专用的4KB程序空间(LDROM)可以用来存放ISP固件。上电以后，用户可以通过Config0的CBS[1]选择从APROM启动还是LDROM启动.

除了设定从APROM还是LDROM启动，Config0的CBS位也可以用来控制上电后的系统内存映射。当CBS[1]=1并且CBS[1] = 1时，系统从APROM启动，并且APROM中的应用程序不能通过内存读的方式访问LDROM。当CBS[0] = 1 并且 set CBS[1] = 0时，系统从LDROM启动，并且LDROM中的应用程序不能通过内存读的方式访问APROM 。.. 图 5-11 显示了从APROM还是LDROM启动时的内存映射。.

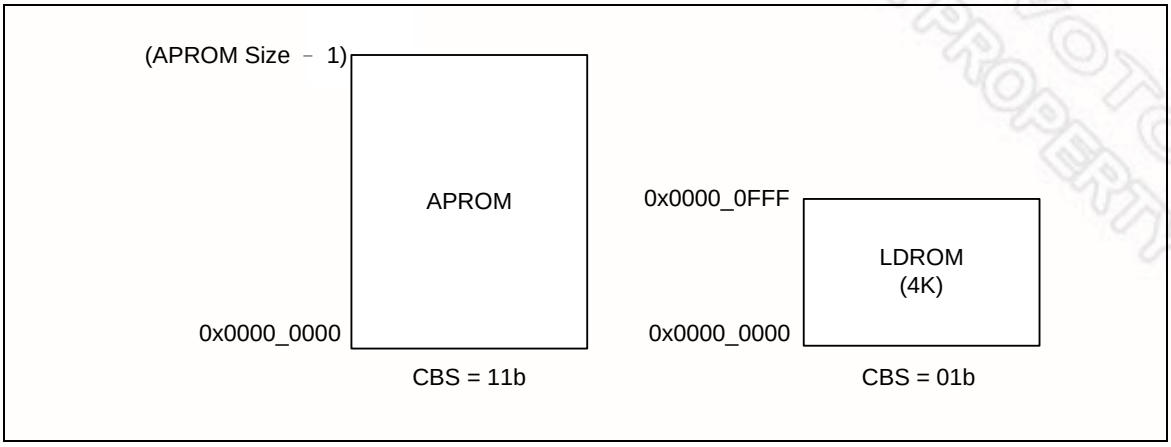


图 5-11 从APROM启动和从LDROM启动时程序的执行范围

对于应用程序来说，软件需要不改变启动模式，让执行在 APROM 中的代码调用 LDROM 中的函数，或者执行在 LDROM 中的代码调用 APROM 中的函数，此时 CBS[0]需要设为 0，这个功能称为在应用中编程（IAP）.

5.5.6 IAP (In Application Programming)

NuMicro™ NUC123 系列支持In-application-programming (IAP) 功能，用户无需复位系统就可以在APROM和LDROM之间切换执行代码。通过设定Config0的CBS[1:0]为10b 或者 00b ，然后将系统复位，即可以使能IAP功能。

CBS[1:0] = 10b的情况下，NuMicro™ NUC123 从APROM 启动，代码的执行地址范围包括所有的APROM 和LDROM。APROM 的存取地址空间保持不变，LDROM的存取地址空间也保持不变被映射到0x0010_0000~ 0x0010_0FFF。

CBS[1:0] = 00b的情况下，NuMicro™ NUC123 从LDROM 启动，代码的执行地址范围包括所有的LDROM 和几乎所有的APROM(除了前面512B)。执行代码不能访问APROM的第一页(前面512B)，因为第一页的执行地址缺省变成LDROM第一页的镜像。同时LDROM也被映射到0x0010_0000~ 0x0010_0FFF。

当IAR使能时，地址空间映射请参考下图。

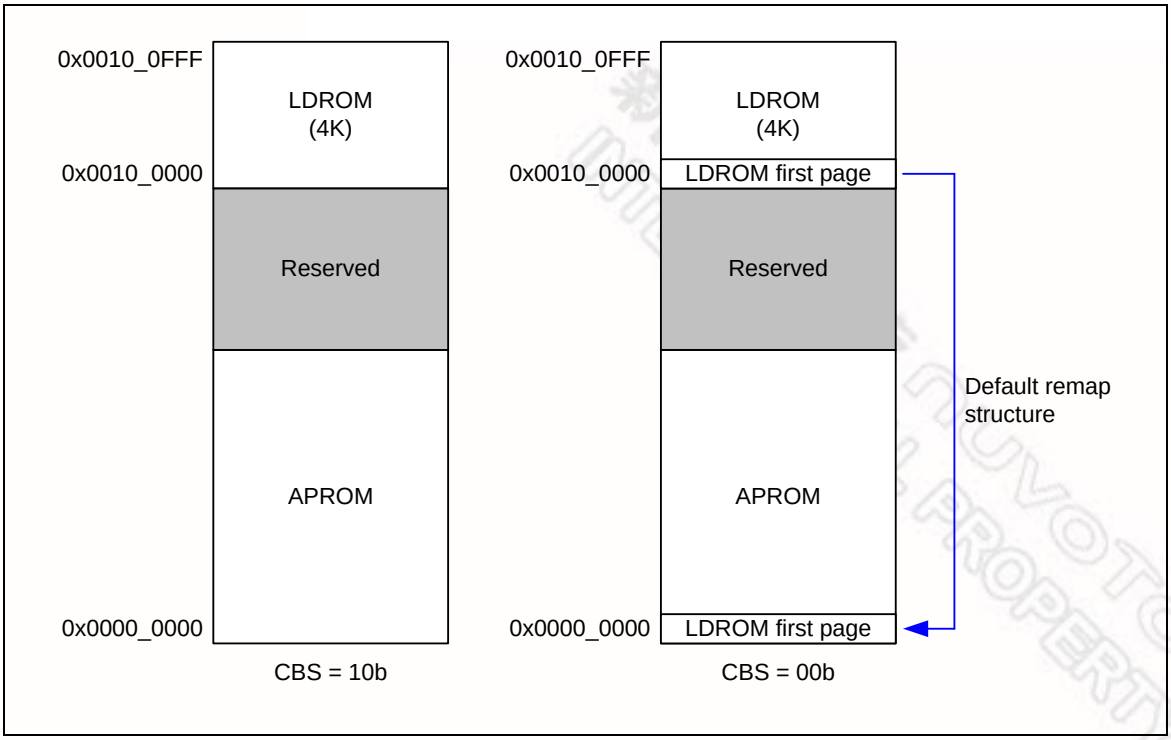


图 5-12 IAP 程序执行范围

如果IAP功能使能，芯片启动时，代码执行范围中的任何其它页任何时间都可以映射到第一页(0x0000_0000~0x0000_01FF)。将要映射的到第一页的地址写入ISPADR寄存器，将重新映射命令写入ISPCMD寄存器，执行ISP过程，用户就可以改变第一页的映射地址。通过ISPSTA寄存器的VECMAP域，用户可以检查映射是否成功.

5.5.7 数据Flash

NuMicro™ NUC123 为用户提供数据FLASH用来存储数据，可以通过ISP过程进行读/写. 擦除单位为512字节. 若要改变一个字，需要先把所有128个字拷贝到另外页或SRAM中.

当Config0中的数据Flash大小可变使能位(DFVSEN) 为1时, 应用程序内存(APROM) 空间为 64/32 K 字节，数据Flash大小为4 KB，起始地址固定为0x0001_F000；当DFVSEN 为 0时，应用程序内存(APROM) 空间为68/36 KB，数据Flash和APROM共享，大小由用户定义。如果Config0中的数据Flash使能位(DFEN) 为 1, 没有数据Flash，所有 68/36 KB都用作 APROM；如果Config0中的数据Flash使能位(DFEN) 为 0, 数据Flash和APROM共享，基地址由Config1中的数据Flash基地址位(DFBADR)决定，此时, 应用程序内存的大小为(68/36-0.5*N)KB ，数据Flash 的大小为 0.5*N KB。

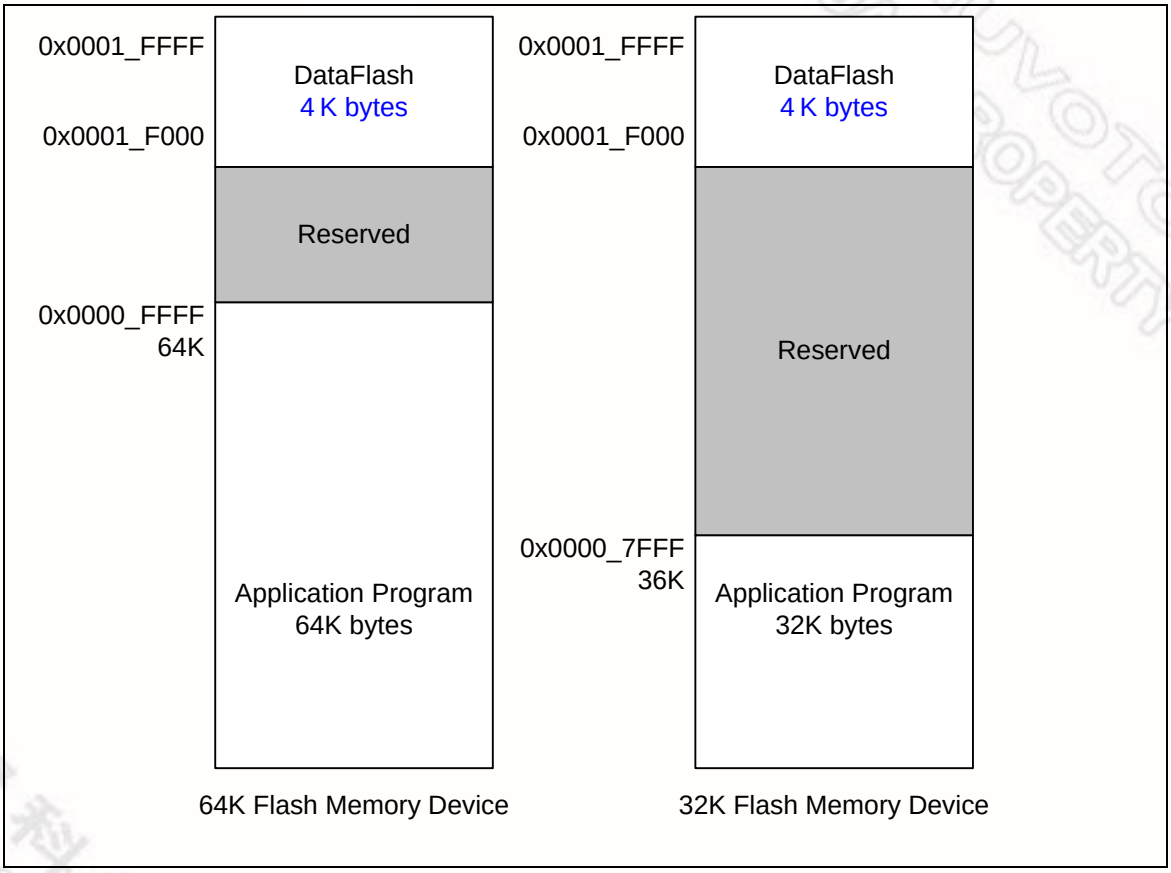


图 5-13 数据 Flash 内存结构 (DFVSEN = 1)

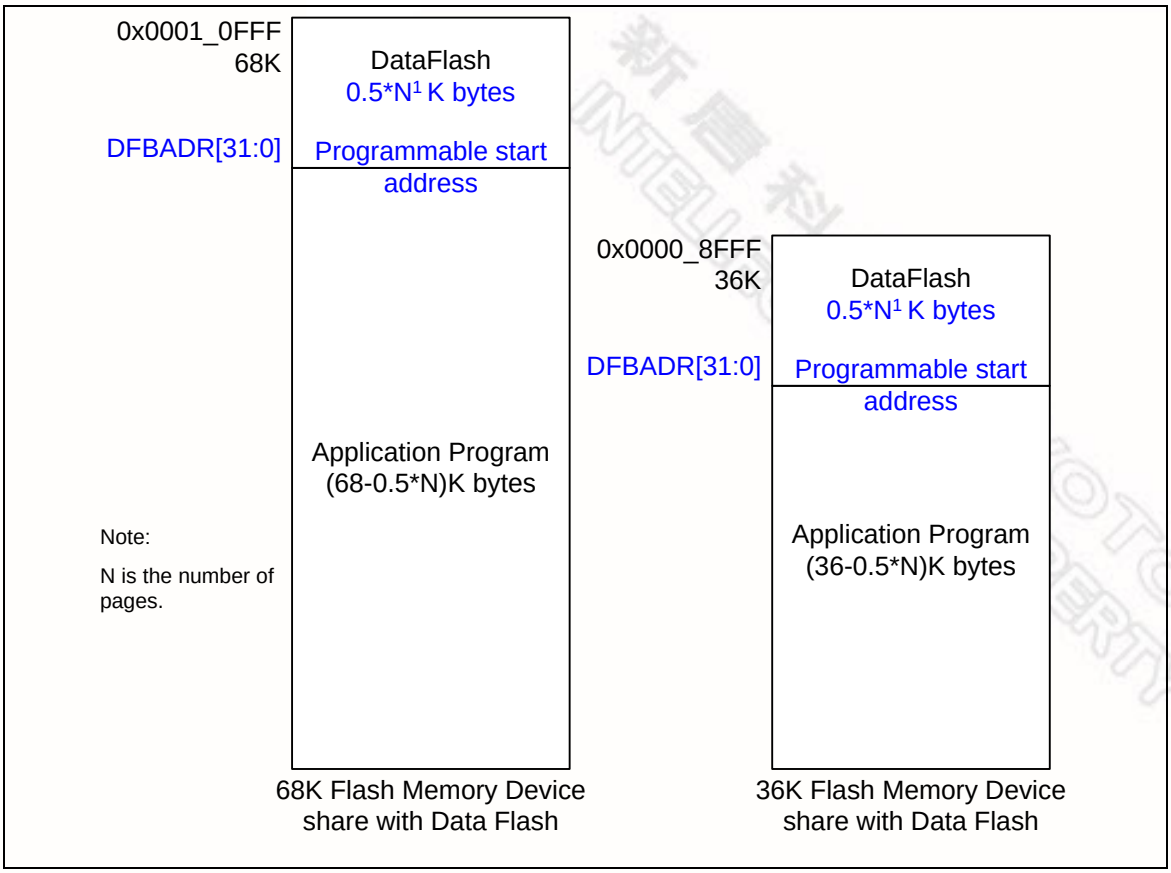


图 5-14 数据 Flash 内存结构 (DFVSEN = 0)

5.5.8 用户配置

Config0 (Address = 0x0030_0000)

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

Bits	描述	
[31]	**CWDTEN**	**看门狗硬件使能** 0 = 芯片上电时默认使能Window Watchdog Timer. 1 =芯片上电时默认关闭 Window Watchdog Timer.
[30]	**CWDTPDEN**	**看门狗时钟掉电时使能控制** 0 = 当芯片进入掉电模式时Watchdog Timer 时钟保持使能. 1 = 当芯片进入掉电模式时Watchdog Timer 的时钟由OSC10K_EN (PWRCON[3])控制. **注意:** 这个比特只有在CWDTEN 为0时才起作用
[29:27]	**保留**	保留
[26:24]	**CFOSC**	**复位后CPU时钟源选择** FO□C[2:0] / 时钟源 000 / 外部 4~24 MHz 高速晶振时钟 111 / 内部 22.1184 MHz 高速振荡器时钟 Others / 保留 复位后，CFOSC的值将被加载到系统寄存器CLKSEL0.HCLK_S[2:0].
[23]	**CBODEN**	**欠压检测使能** 0= 上电后，使能欠压检测 1= 上电后，禁用欠压检测

<table>
<tr><td>[22:21]</td><td>CBOV1-0</td><td>欠压电压选择
<table>
<tr><th>CBOV</th><th>CBOV0</th><th>Brown out □oltage</th></tr>
<tr><td>1</td><td>1</td><td>4.5V</td></tr>
<tr><td>1</td><td>0</td><td>3.8</td></tr>
<tr><td>0</td><td>1</td><td>2.7V</td></tr>
<tr><td>0</td><td>0</td><td>2□2V</td></tr>
</table></td></tr>
<tr><td>[20]</td><td>CBORST</td><td>欠压复位使能
0 = 上电后，使能欠压复位
1 = 上电后，禁用欠压复位</td></tr>
<tr><td>[19:8]</td><td>保留</td><td>保留</td></tr>
<tr><td>[7:6]</td><td>CBS</td><td>启动选项
<table>
<tr><th>CBS[1:0]</th><th>启动选择</th></tr>
<tr><td>11</td><td>芯片由APROM启动，程序执行范围只包括APROM。除非通过ISP，否则LDROM不能被APROM中的程序直接访问.
这个模式下APROM 是写保护的，需要使能ISPCON寄存器中的APUEN之后才可以更新。</td></tr>
<tr><td>01</td><td>芯片由LDROM启动, 程序执行范围只包括 LDROM; 除非通过ISP，否则APROM 不能被LDROM中的程序直接访问.
这个模式下APROM 可以被更新.</td></tr>
<tr><td>10</td><td>芯片由 APROM启动, 程序执行范围包括LDROM 和 APROM
LDROM 地址映射在0x0010_0000~0x0010_0FFF
此模式下APROM 可以被更新.
通过ISP命令，地址 0x0000_0000 ~ 0x0000_01FF 可以被重新映射到程序执行范围中的任何页.</td></tr>
<tr><td>00</td><td>芯片由LDROM启动, 程序执行范围包括LDROM 和 大部分APROM (除了第一个512 bytes, 因为第一个t 512 bytes 被映射到了 LDROM)
LDROM 地址被映射到 0x0010_0000 ~ 0x0010_0FFF, 第一页同时也被映射到地址 0x0000_0000 ~ 0x0000_01FF.
此模式下APROM 可以被更新.
通过ISP命令，地址 0x0000_0000 ~ 0x0000_01FF 可以被重新映射到程序执行范围中的任何页。</td></tr>
</table></td></tr>
<tr><td>[5:3]</td><td>保留</td><td>保留</td></tr>
<tr><td>[2]</td><td>DFVSEN</td><td>数据 Flash 大小可变使能
0 = 数据 flash的大小是可变的，基地址由 DFBADR (Config1)决定.
1 = 数据 flash 的大小固定为 4K bytes.</td></tr>
</table>

[1]	**LOCK**	**安全加密** 0 = 加密FLASH数据 1 = 解除Flash 数据加密 当FLASH数据被加密，仅有器件ID， Config0 和Config1 可被烧写器和ICP通过串口调试接口读取. 其他数据锁定为0xFFFFFFFF. 无论数据是否锁定，ISP 都可以读取
[0]	**DFEN**	**数据FLASH使能**(该位仅支持 128KB APROM 器件) 0 = 使能数据FLASH 1 = 禁用数据FLASH

Config1 (Address = 0x0030_0004)

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

Bits	描述	
[31:20]	**保留**	保留 (强行给这些保留位写入0x00)
[19:0]	**DFBADR**	**数据FLASH的基地址** (只有当DFEN等于0时这个寄存器才起作用) 如果DFEN等于0，数据Flash的基地址由用户定义。因为片上FLASH擦除单位为512字节, 所以强制bit 8-0位为0.

5.5.9 在系统编程 (ISP)

程序内存和数据FLASH都支持硬件编程和在系统编程 (ISP). 硬件编程模式采用批量写，以方便量产阶段进行编程。若产品还在开发阶段或终端用户需要升级固件时，硬件编程模式不是很方便，ISP模式能更好地适用于这种情况。NuMicro™ NUC123 支持 ISP 模式，即通过软件控制来对器件重新编程. 而且，这也使得应用程序得以广泛应用.

ISP 可以在不用将微控器从系统中取下来的情况下执行编程. 各种接口使得LDROM中的固件更容易取得程序代码. 执行ISP最常用的方法是在LDROM中的固件通过UART更新，PC一般都是通过串口传输新 的APROM代码. LDROM接收后，通过ISP命令，重新对APROM编程。Nuvoton 提供用于NuMicro™ NUC123 的ISP 固件和 PC 应用程序. 这让用户可以通过Nuvoton ISP工具非常方便地执行ISP.

5.5.10 ISP过程

NuMicro™ NUC123 支持从APROM 或 LDROM启动.，默认上电时的启动方式用户可以通过配置Config0的CBS位来定义。运行状态下，用户想更新APROM中的应用程序时，可以写ISPCON寄存器的BS=1，并发起软件复位使芯片由LDROM启动；反之用户想从LDROM切到APROM时，可以写BS=0，然后发起软件复位。 开始ISP功能的第一步是向ISPEN写1。在向ISPCON寄存器写数据之前，S/W 需要向全局控制寄存器（GCR, 0x5000_0100）的寄存器 RegLockAddr 依次写入0x59, 0x16 和 0x88，这个过程用于保护FLASH内存免受因为在上电或断电期间无意识的写操作而造成的损坏.

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

ISPGO置位时，CPU将等待ISP操作结束，在此期间，外设仍然在工作，如果有中断请求时，CPU仍然会先执行完ISP后再响应中断。ISP操作完成后，ISPGO比特会被硬件自动清0。通过查看这个比特，用户可以知道ISP操作是否已经完成。在写ISPGO为1之后，用户应该添加ISB指令来确保ISP过程之后的指令能正确执行。

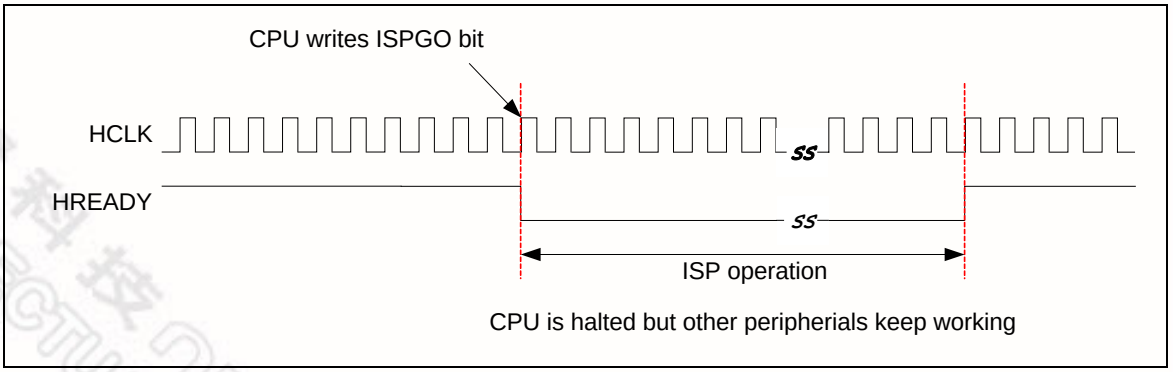


图 5-15 ISP操作期间CPU 停住

.

注意 NuMicro™ NUC123 系列允许用户通过ISP更新CONFIG的值，但是因为应用程序代码加密的问题，在擦除CONFIG区域之前，软件需要先擦除至少1页APROM，否则CONFIG将不允许擦除.

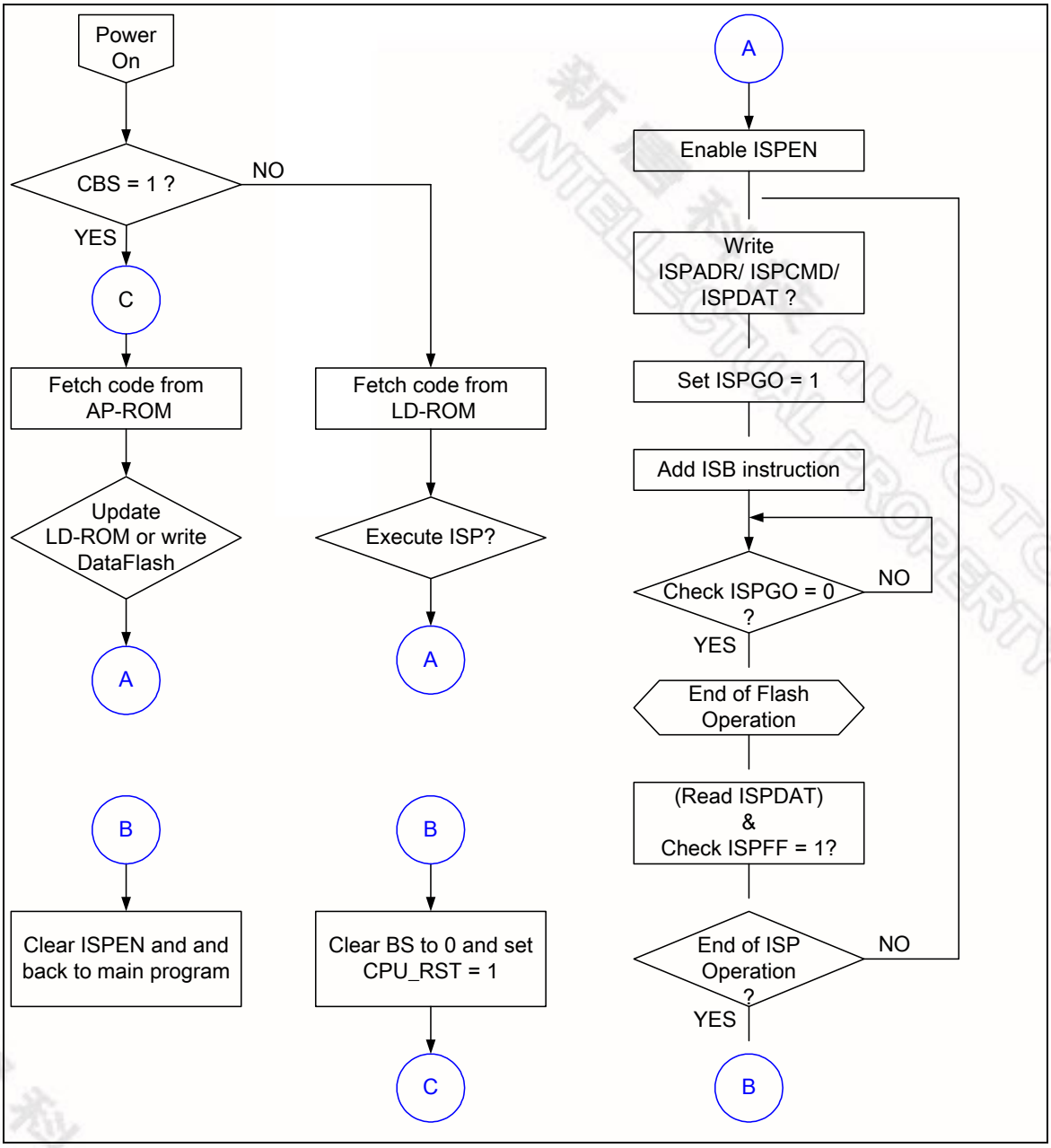


图 5-16 ISP 操作流程

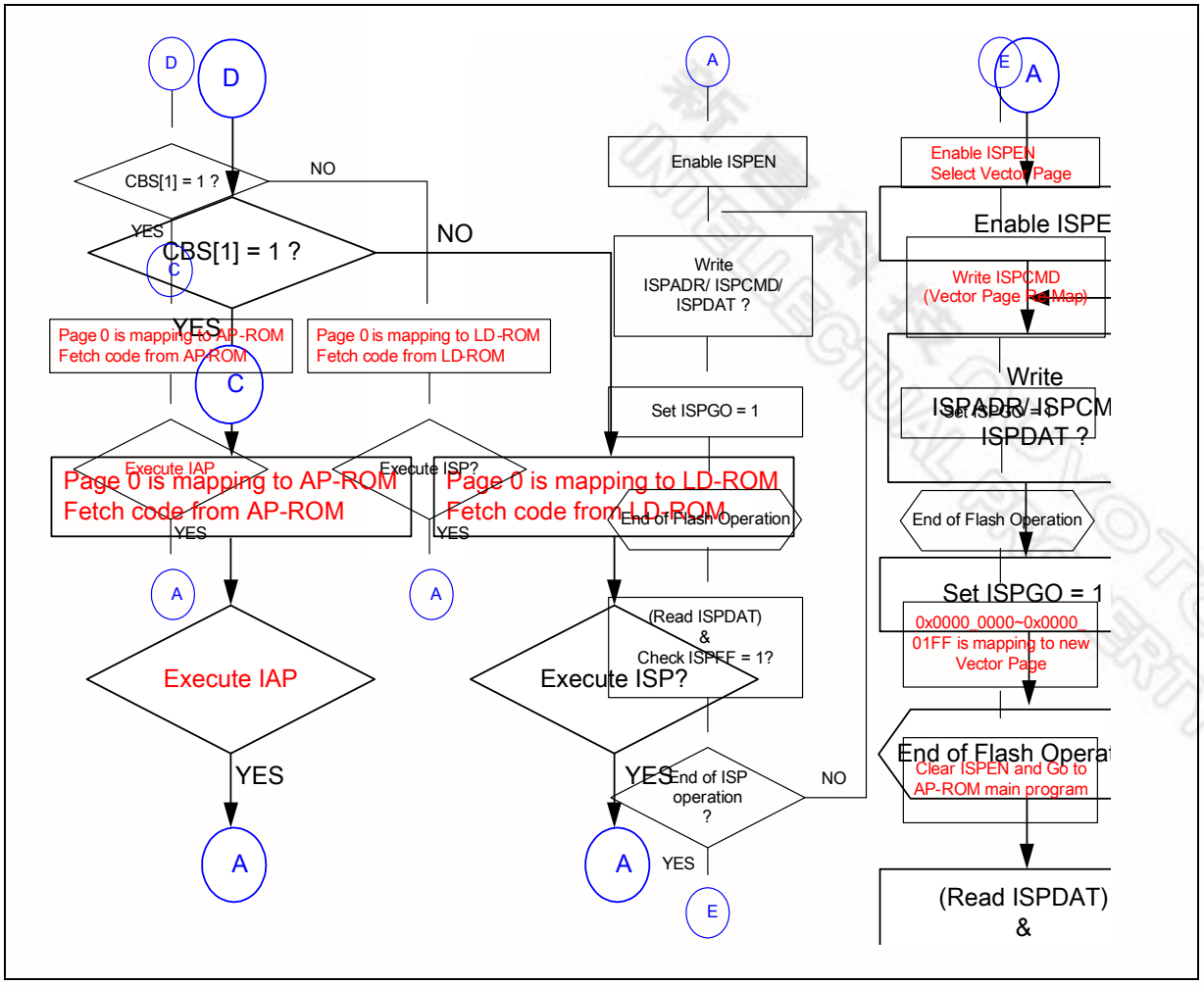


图 5-17 ISP 操作流程 (继续)

ISP Mode	ISPCMD			ISPADR			ISPDAT
	FOEN	FCEN	FCTRL[3:0]	A21	A20	A[19:0]	D[31:0]
Vector Page Re-Map	1	0	1110	0	A20	Address in A[19:0]	x
FLASH Page Erase	1	0	0010	0	A20	Address in A[19:0]	0xFFFF_FFFF
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]	0xFFFF_FFFF

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]
Read Unique ID	0	0	0100	0	0	Address in A[19:0] = 0x00_0000 0x00_0004 0x00_0008	Data out D[31:0]

表 5-8 ISP 模式命令

5.5.11 Flash 控制寄存器

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
FMC 基地址: **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
DFBADR	FMC_BA+0x14	R	数据FLASH开始地址	0xXXXX_XXXX
FATCON	FMC_BA+0x18	R/W	FLASH访问窗口控制寄存器	0x0000_0000
ISPSTA	FMC_BA+0x40	R/W	ISP状态寄存器	0x0000_0000

5.5.12 Flash控制寄存器描述

ISP 控制寄存器 (ISPCON)

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

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

Bits	描述	
[31:7]	**保留**	保留
[6]	**ISPFF**	**ISP失败标志** (写保护位) 当ISP满足下列条件时，该位由硬件置位: (1) APUEN等于0时，擦除或编程APROM. (2) LDUEN等于0时，擦除或编程LDROM. (3) CFGUEN等于0时，CONFIG 被擦除或编程. (4) 定义地址无效，如超过正常范围. (5) 在BOD使能时，如果BOD检测电压符合，执行ISP擦除或者写命令 (6) ISP命令无效 该位写 1 清除.
[5]	**LDUEN**	**LDROM更新使能** (写保护位) LDROM 更新使能位. 1 = LDROM 可以被更新. 0 = 禁止LDROM更新
[4]	**CFGUEN**	**配置区更新使能** (写保护位) 1 = 使能ISP更新配置位 0 = 禁止ISP更新配置位
[3]	**APUEN**	**APROM 更新使能 (**写保护位**)** 1 = 当芯片在APROM中运行时APROM 可以被更新. 0 = 当芯片在APROM中运行时APROM 不能被更新.
[2]	**保留**	保留

[1]	**BS**	**启动选择** (写保护位) 置位/清零该位选择下次是由LDROM启动还是由APROM启动, 该位也可作为MCU启动状态的标志,用于检查MCU是由LDROM还是APROM启动的. 这个比特在复位时（除了 CPU 复位 (RSTS_CPU 为 1) 或系统复位 (RSTS_SYS)）被初始化为 Config0 的 CBS 位的反转值，在其他复位时保持不变. 1 = 由LDROM启动 0 = 由APROM启动
[0]	**ISPEN**	**ISP 使能**(写保护位) ISP 使能位，设置该位可以使能ISP功能. 1 = 使能 ISP 功能 0 = 禁用 ISP 功能

ISP 地址(ISPADR)

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

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

Bits	描述	
[31:0]	**ISPADR**	**ISP地址** NuMicro™ NUC123 内嵌flash, 仅支持字编程. 执行ISP功能时，ISPARD[1:0] 必须为00b.

ISP 数据寄存器 (ISPDAT)

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

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

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

ISP 命令(ISPCMD)

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

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

Bits	描述	
[31:6]	保留	保留
[5]	FOEN	**ISP命令** ISP 命令表如下:
[4]	FCEN	
[3:0]	FCTRL	

操作模式	FOEN	FCEN	FCTRL[3:0]			
读	0	0	0	0	0	0
编程	1	0	0	0	0	1
页擦除	1	0	0	0	1	0
读UID	0	0	0	1	0	0

ISP触发控制寄存器(ISPTRG)

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

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

Bits	描述	
[31:1]	保留	保留
[0]	ISPGO	**ISP开始触发** 写 1 开始ISP操作，当ISP操作结束后，该位由硬件自动清零. 1 = ISP 正在执行 0 = ISP 操作结束

数据FLASH基地址寄存器(DFBADR)

寄存器	偏移量	R/W	描述	复位后的值
DFBADR	FMC_BA+ 0x14	R	数据FLASH基地址	0xXXXX_XXXX

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

Bits	描述	
[31:0]	**DFBADR**	**数据FLASH基地址** 该寄存器为数据FLASH开始地址寄存器，只读. DFVSEN等于0时，数据FLASH和APROM共享，数据Flash的大小由用户配置定义。芯片上电后该寄存器的默认值从Config1加载。 DFVSEN等于1时，数据Flash的大小固定为4K，地址固定为0x01_F000

Flash访问时间控制寄存器 (FATCON)

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

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

Bits	描述	
[31:7]	保留	保留
[6]	MFOM	中频优化模式（写保护位） 当芯工作频率低于 50 MHz时，通过设置该位为 1 芯片可以更高效的工作。 1 = 使能中频优化模式 0 = 禁用中频优化模式
[5]	保留	保留
[4]	LFOM	低频优化模式（写保护位） 当芯片操作频率低于 25 MHz时，通过设置该位为 1 芯片可以更高效的工作。 1 = 使能低频优化模式 0 = 禁用低频优化模式
[3:0]	保留	保留

ISP 状态寄存器 (ISPSTA)

寄存器	偏移	R/W	描述	复位后的值
ISPSTA	FMC_BA+0x40	R/W	ISP 状态寄存器	0x0000_0000

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

Bits	描述	
[31:21]	**保留**	**保留**
[20:9]	**VECMAP**	**向量表映射地址** 当前Flash地址空间 0x0000_0000~0x0000_01FF被映射到地址 {VEC}AP[11:0], “000000000b”} ~ {VEC}AP[11:0], “111111111b”} 该位只读
[8:7]	**保留**	**保留**
[6]	**ISPFF**	**ISP 失败标志**(写保护位) 当触发的ISP过程遇到下列情况时，该位由硬件置为1: (1) APUEN等于0时，擦除或编程APROM. (2) LDUEN等于0时， 擦除或编程LDROM. (3) CFGUEN等于0时，CONFIG 被擦除或编程. (4) 目的地址无效，如超过正常范围. (5) 在BOD使能时，如果BOD检测电压符合，执行ISP擦除或者写命令 (6) ISP命令无效 写1清0.
[5:3]	**保留**	**保留**

[2:1]	**CBS**	**配置启动选择** CBS[1:0] 启动选择 01 芯片从LDROM启动; APROM 不可用内存读的方式读 11 芯片从 APROM启动; LDROM不可用内存读的方式读 00 芯片从LDROM启动; LDROM 和 APROM 都是可读的 10 芯片从APROM启动; LDROM 和 APROM 都是可读的
[0]	**ISPBUSY**	**ISP 忙** 1 = ISP 操作正在进行中 0 = ISP 操作完成 该位只读

5.6 USB设备控制器 (USB)

5.6.1 概述

该器件有一组全速USB 2.0 设备控制器和收发器。符合USB 2.0全速设备规范，支持控制/块/中断/等时 传输类型。

在该设备控制器里，包含两个主要接口：APB总线和来自USB PHY收发器的USB总线. CPU通过APB总线编程控制寄存器。该控制器内置有512字节的SRAM作为数据缓存。输入或输出传输, 需要通过APB接口或SIE向SRAM写数据或从SRAM读数据. 用户需要通过BUFSEGx为每个端点设置有效的SRAM偏移地址用来缓存数据.

USB 设备控制器共有 8 个可配置的端点。每个端点可以配置为 IN 或者 OUT 类型。所有的操作包括 Control, Bulk, Interrupt 和 Isochronous 传输都可以支持。端点控制模块还可以用来管理数据序列同步，端点状态，当前起始地址，当前事务状态和每个端点的数据缓存状态。

该控制器有4个不同的中断事件. 包括唤醒功能，设备插拔事件，USB事件（如IN ACK, OUT ACK 等）, 和 BUS 事件（如suspend 和 resume, 等）。 任何事件都可以引起一个中断, 用户只需要在中断事件状态寄存器(USB_INTSTS)里检查相关事件标志以得知发生何种中断, 然后检测相关USB端点状态寄存器 (USB_EPSTS)以得知在该端点上发生哪种事件.

USB设备有一个软件禁用功能，用于模拟设备从主机插拔的情况。如果用户使能DRVSE0 位(USB_DRVSE0), USB控制器将迫使USB_DP和USB_DM输出低电平，USB功能被禁止。在关闭DRVSE0位后，主机将重新枚举USB设备.

参考: 通用串行总线规格修订版 1.1

5.6.2 特性

该通用串行总线（USB）用一个连接器来连接所有的USB外设到主机系统。下面是USB的一些特性.

- 兼容USB 2.0 全速规格
- 提供1个中断向量，4个中断事件(WAKEUP, FLDET, USB 和 BUS)
- 支持Control/Bulk/Interrupt/Isochronous传输类型
- 在没有总线活动超过3ms后支持总线挂起功能
- 提供8个端点，可配置为Control/Bulk/Interrupt/Isochronous传输类型，最大512字节的缓存
- 提供远程唤醒功能

5.6.3 框图

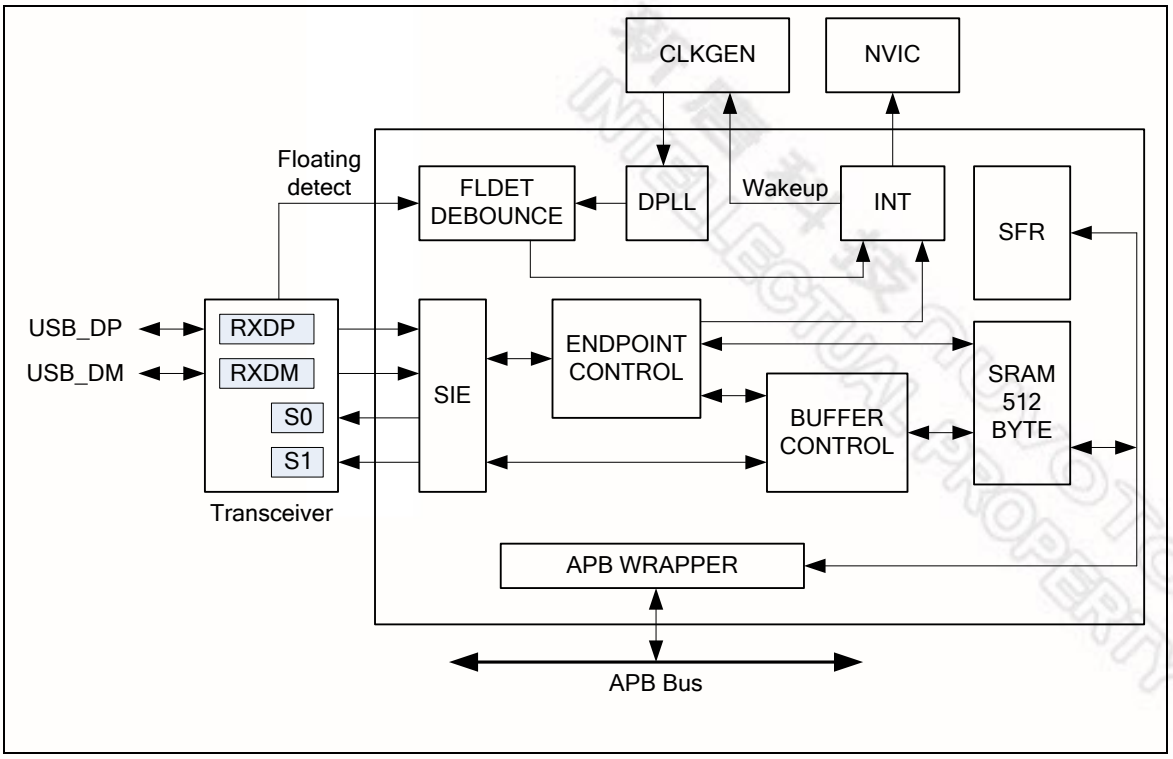


图 5-18 USB 框图

5.6.4 功能描述

5.6.4.1

SIE (Serial Interface Engine)

SIE是设备控制器的前端，处理大多数的USB 协议包。SIE典型处理信号向上发送到达事务处理层面。处理功能包括:

- 包识别，事务排序
- SOP, EOP, RESET, RESUME信号检测/产生
- Clock/Data分离
- NRZI 数据编解码与比特填充（bit-stuffing）
- CRC产生和校验 (for Token and Data)

5.6.4.2

- Packet ID (PID) 产生和校验/解码
- 串-并/并-串转换

端点控制

该控制器有8个端点，这8个端点0~7有2个不同的配置地址(详见5.5.4节的寄存器映射表)。每个端点可以配置成Control, Bulk, Interrupt, 或 Isochronous传输类型. 所有操作包括Control, Bulk, Interrupt 和 Isochronous 传输都在该模块内执行. 也可以用来管理数据时序同步，端点状态控制，当前端点地址，当前事务状态和每个端点的数据缓冲状态.

数字锁相环

USB数据的比特率为12 MHz, DPLL使用的48 MHz的频率由时钟控制器产生，用来锁定RXDP与RXDM的输入数据，12 MHz的比特率时钟也是由DPLL转换来的.

5.6.4.3 *插拔去抖动*

5.6.4.4 USB设备可以进行热插拔操作，为了监测USB设备拔出/插入的状态，在发生USB拔插中断时，设备控制器提供了硬件去抖动以防止在USB插拔时产生的抖动问题。拔插检测中断产生于USB进行插拔操作的10ms后，用户可以通过读取“USB_FLDET”寄存器的值，来应答USB插拔. “FLDET” 标志代表在当前总线上没有经过去抖动处理的状态。如果 FLDET 为 1，则意味着控制器已经插入 USB。若用户要通过这个标志来检测 USB 的状态（不通过中断的方式），则需要添加软件去抖动功能。

中断

5.6.4.5 该USB提供1个中断信号有4个中断事件的中断向量（WAKEUP, FLDET, USB, BUS）. WAKEUP中断被用来在掉电模式下唤醒系统时钟。（掉电模式功能在系统掉电控制寄存器PWRCON中定义.）FLDET 中断用于 USB 插拔。USB 事件告知用户一些USB请求,如IN ACK，OUT ACK等, BUS事件告知用户一些总线事件，如 暂停、复位等. 用户必须在USB设备控制器的中断使能寄存器（(USB_INTEN) ）设置相应的位以使能USB中断。

唤醒中断会在芯片进入掉电模式且唤醒事件发生时出现. 在芯片进入掉电模式后，D+和D-的任何变化都能唤醒芯片（假定USB唤醒功能使能时）。若这个变化不是有意而为的，那么只有唤醒中断会发生。在USB唤醒超过 20 ms后，若没有其它USB中断事件出现，唤醒中断将发生。下图为唤醒中断的控制流程.

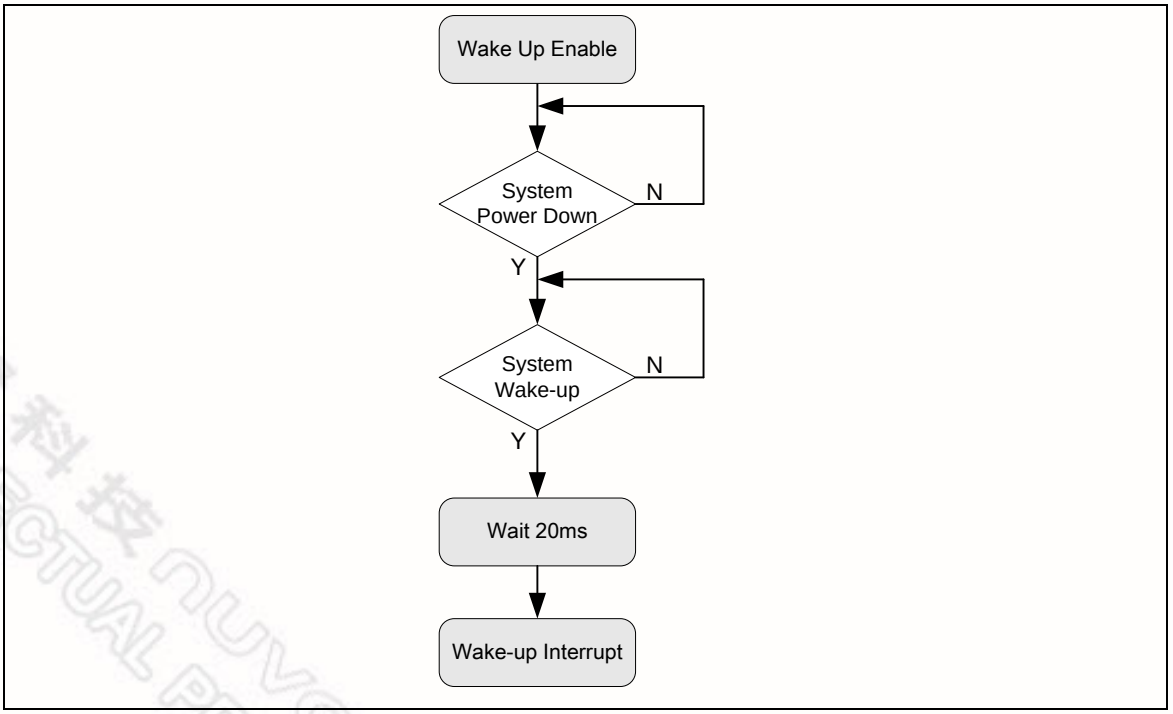


图 5-19 唤醒中断的操作流程

USB 中断告知用户所有总线上的USB事件，用户可以读特殊功能寄存器EPSTS

(USB_EPSTS[25:8]) 和 EPEVT5~0 (USB_INTSTS[21:16])来获知是哪类请求，对哪个端点采取响应.

与USB中断相似，BUS中断告之用户一些总线事件，如USB复位，中止，暂停和恢复，用户可以读寄存器“USB_ATTR”获取总线事件.

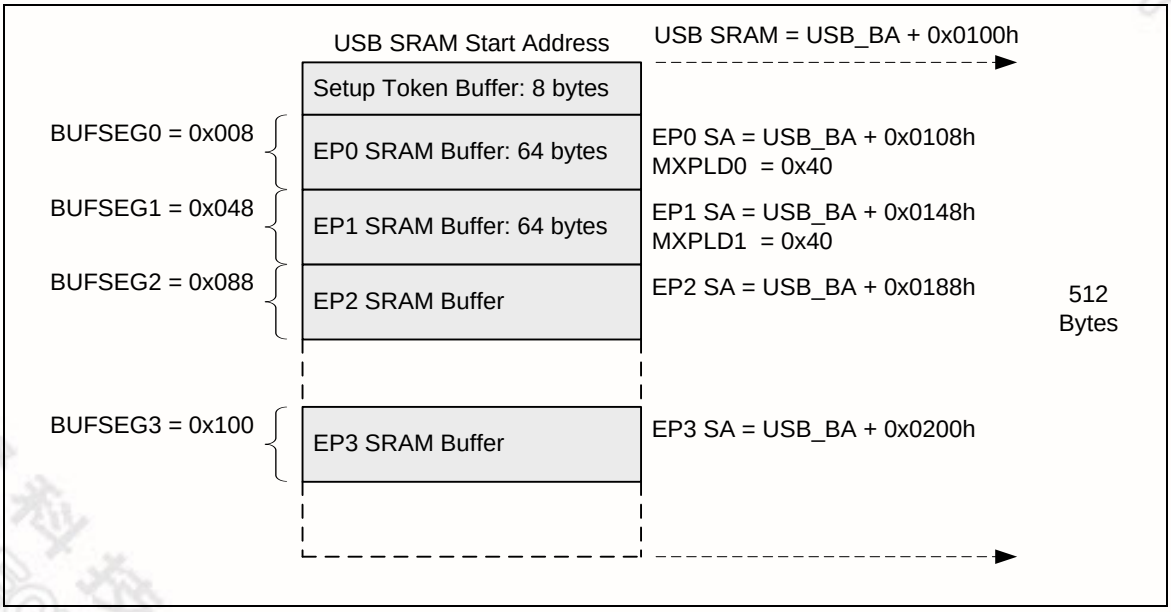
省电

5.6.4.6 当芯片进入掉电模式时，USB 自动关闭PHY收发器省电，此外，在特定环境下，用户可以将寄存器USB_ATTR［4］写入“0”关闭PHY进入省电状态.

缓冲控制

5.6.4.7 USB控制器有512 bytes SRAM和8个端点共享该缓冲. 在USB功能有效前，用户需要在缓冲段寄存器配置每个端点的有效起始地址. BUFFER CONTROL 用于控制每个端点的有效起始地址和SRAM的大小（在寄存器MXPLD定义）.

图 5-20 根据BUFSEG和MXPLD寄存器的内容描述每个端点的起始地址. 如果BUFSEG0 设置为0x08h 和MXPLD0 设置为0x40h, 端点0的SRAM 大小从USB_BA + 0x108h开始，到 USB_BA + 0x148h结束. (注: USB SRAM 基地址为USB_BA + 0x100h).



5.6.4.8 图 5-20 端点 SRAM 的结构

与USB外设的传输处理

用户可以采用中断或轮询USB_INTSTS的方式来监测USB传输，在USB传输发生时，USB_INTSTS由硬件设置，并向CPU发送中断请求（如果相关中断使能），在没有使能中断时，用户可以轮询USB_INTSTS来获取事件，以下是中断使能时的控制流程.

USB主机向设备控制器请求数据时，用户需要预先准备相关的数据到指定的端点缓存. 在将数据写入缓冲区后，用户需要写入实际数据长度到指定的MAXPLD寄存器. 一旦这个寄存器被写，内部信

号“In_Rdy” 会被设置，当收到主机发送的相关IN token之后，缓冲数据将被立刻传送。在传送指定数据之后，信号“In_Rdy ”会由硬件自动清除.

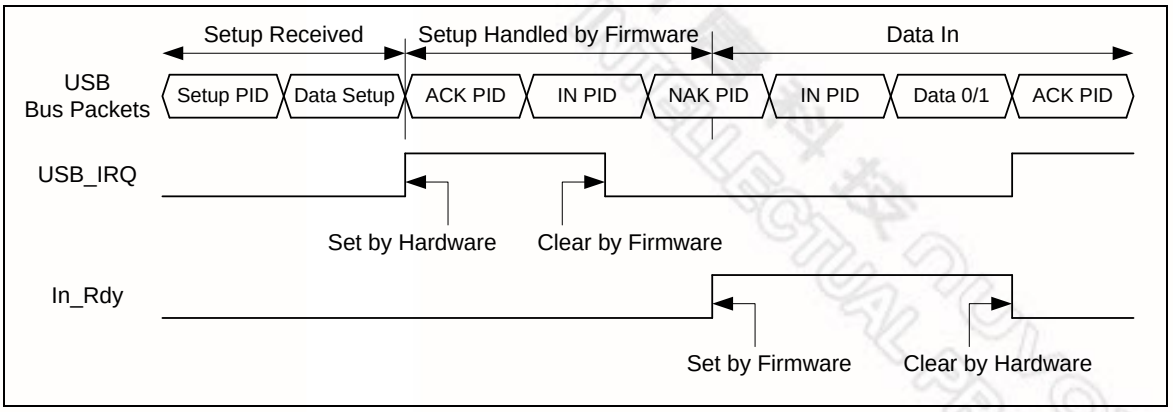


图 5-21 数据传入时间里传输流程图

USB主机要发送数据到设备控制器的OUT端点时，硬件将这些数据存在指定的端点缓存里，传输完成后，硬件在MAXPLD寄存器记录接收到的数据长度，并清除“Out_Rdy”信号，这避免硬件在用户没有取走当前数据时接收下一个数据。一旦用户处理完这次传输时，由软件写入相关的寄存器“MAXPLD”来设置 “Out_Rdy” 信号以接收下一次传输

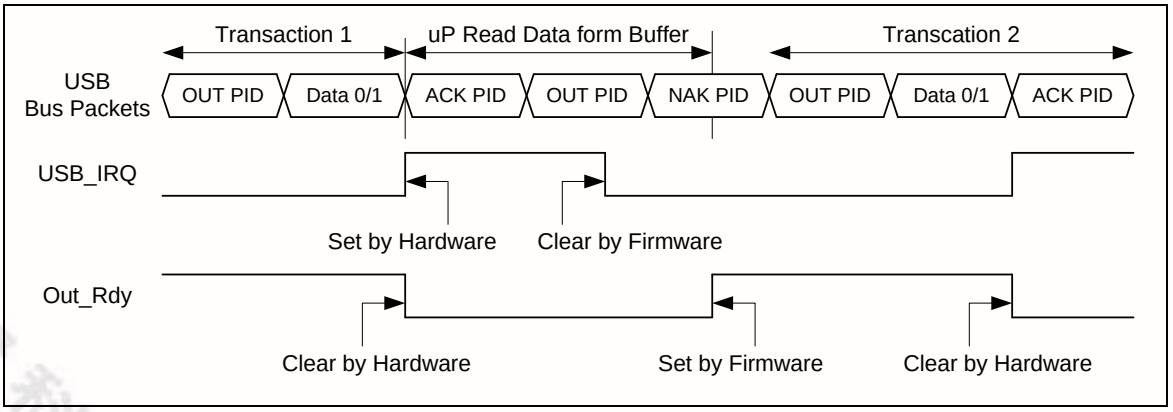


图 5-22 数据输出图

5.6.5 寄存器与内存映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
USB 基地址: **USB_BA = 0x4006_0000**				
USB_INTEN	USB_BA+0x000	R/W	USB 中断使能寄存器	0x0000_0000
USB_INTSTS	USB_BA+0x004	R/W	USB 中断事件状态寄存器	0x0000_0000
USB_FADDR	USB_BA+0x008	R/W	USB 设备功能地址寄存器	0x0000_0000
USB_EPSTS	USB_BA+0x00C	R	USB 端点状态寄存器	0x0000_00x0
USB_ATTR	USB_BA+0x010	R/W	USB 总线状态和归属寄存器	0x0000_0040
USB_FLDET	USB_BA+0x014	R	USB 插拔检测寄存器	0x0000_0000
USB_BUFSEG	USB_BA+0x018	R/W	Setup Token缓存偏移寄存器	0x0000_0000
USB_BUFSEG0	USB_BA+0x020	R/W	端点0的缓存偏移寄存器	0x0000_0000
USB_MXPLD0	USB_BA+0x024	R/W	端点0的最大有效载荷寄存器	0x0000_0000
USB_CFG0	USB_BA+0x028	R/W	端点0的配置寄存器	0x0000_0000
USB_CFGP0	USB_BA+0x02C	R/W	端点0的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG1	USB_BA+0x030	R/W	端点1的缓存偏移寄存器	0x0000_0000
USB_MXPLD1	USB_BA+0x034	R/W	端点1的最大有效载荷寄存器	0x0000_0000
USB_CFG1	USB_BA+0x038	R/W	端点1的配置寄存器	0x0000_0000
USB_CFGP1	USB_BA+0x03C	R/W	端点1的设置延迟与清除In/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG2	USB_BA+0x040	R/W	端点2的缓存偏移寄存器	0x0000_0000
USB_MXPLD2	USB_BA+0x044	R/W	端点2的最大有效载荷寄存器	0x0000_0000
USB_CFG2	USB_BA+0x048	R/W	端点2的配置寄存器	0x0000_0000
USB_CFGP2	USB_BA+0x04C	R/W	端点2的设置延迟与清除In/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG3	USB_BA+0x050	R/W	端点3的缓存偏移寄存器	0x0000_0000
USB_MXPLD3	USB_BA+0x054	R/W	端点3的最大有效载荷寄存器	0x0000_0000
USB_CFG3	USB_BA+0x058	R/W	端点3的配置寄存器	0x0000_0000
USB_CFGP3	USB_BA+0x05C	R/W	端点3的设置延迟与清除In/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG4	USB_BA+0x060	R/W	端点4的缓存偏移寄存器	0x0000_0000
USB_MXPLD4	USB_BA+0x064	R/W	端点4的最大有效载荷寄存器	0x0000_0000

USB_CFG4	USB_BA+0x068	R/W	端点4的配置寄存器	0x0000_0000
USB_CFGP4	USB_BA+0x06C	R/W	端点4的设置延迟与清除In/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG5	USB_BA+0x070	R/W	端点5的缓存偏移寄存器	0x0000_0000
USB_MXPLD5	USB_BA+0x074	R/W	端点5的最大有效载荷寄存器	0x0000_0000
USB_CFG5	USB_BA+0x078	R/W	端点5的配置寄存器	0x0000_0000
USB_CFGP5	USB_BA+0x07C	R/W	端点5的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_DRVSE0	USB_BA+0x090	R/W	USB 驱动 SE0 控制寄存器	0x0000_0001
USB_BUFSEG0	USB_BA+0x500	R/W	端点0的缓存偏移寄存器	0x0000_0000
USB_MXPLD0	USB_BA+0x504	R/W	端点0的最大有效载荷寄存器	0x0000_0000
USB_CFG0	USB_BA+0x508	R/W	端点0的配置寄存器	0x0000_0000
USB_CFGP0	USB_BA+0x50C	R/W	端点0的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG1	USB_BA+0x510	R/W	端点1的缓存偏移寄存器	0x0000_0000
USB_MXPLD1	USB_BA+0x514	R/W	端点1的最大有效载荷寄存器	0x0000_0000
USB_CFG1	USB_BA+0x518	R/W	端点1的配置寄存器	0x0000_0000
USB_CFGP1	USB_BA+0x51C	R/W	端点1的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG2	USB_BA+0x520	R/W	端点2的缓存偏移寄存器	0x0000_0000
USB_MXPLD2	USB_BA+0x524	R/W	端点2的最大有效载荷寄存器	0x0000_0000
USB_CFG2	USB_BA+0x528	R/W	端点2的配置寄存器	0x0000_0000
USB_CFGP2	USB_BA+0x52C	R/W	端点2的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG3	USB_BA+0x530	R/W	端点3的缓存偏移寄存器	0x0000_0000
USB_MXPLD3	USB_BA+0x534	R/W	端点3的最大有效载荷寄存器	0x0000_0000
USB_CFG3	USB_BA+0x538	R/W	端点3的配置寄存器	0x0000_0000
USB_CFGP3	USB_BA+0x53C	R/W	端点3的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG4	USB_BA+0x540	R/W	端点4的缓存偏移寄存器	0x0000_0000
USB_MXPLD4	USB_BA+0x544	R/W	端点4的最大有效载荷寄存器	0x0000_0000
USB_CFG4	USB_BA+0x548	R/W	端点4的配置寄存器	0x0000_0000
USB_CFGP4	USB_BA+0x54C	R/W	端点4的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG5	USB_BA+0x550	R/W	端点5的缓存偏移寄存器	0x0000_0000
USB_MXPLD5	USB_BA+0x554	R/W	端点5的最大有效载荷寄存器	0x0000_0000
USB_CFG5	USB_BA+0x558	R/W	端点5的配置寄存器	0x0000_0000

USB_CFGP5	USB_BA+0x55C	R/W	端点5的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG6	USB_BA+0x560	R/W	端点6的缓存偏移寄存器	0x0000_0000
USB_MXPLD6	USB_BA+0x564	R/W	端点6的最大有效载荷寄存器	0x0000_0000
USB_CFG6	USB_BA+0x568	R/W	端点6的配置寄存器	0x0000_0000
USB_CFGP6	USB_BA+0x56C	R/W	端点6的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000
USB_BUFSEG7	USB_BA+0x570	R/W	端点7的缓存偏移寄存器	0x0000_0000
USB_MXPLD7	USB_BA+0x574	R/W	端点7的最大有效载荷寄存器	0x0000_0000
USB_CFG7	USB_BA+0x578	R/W	端点7的配置寄存器	0x0000_0000
USB_CFGP7	USB_BA+0x57C	R/W	端点7的设置延迟与清除IN/Out准备好控制寄存器	0x0000_0000

内存类型	**地址**	**大小**	**描述**
USB_BA = 0x4006_0000			
SRAM	USB_BA+0x100 ~ USB_BA+0x2FF	512 Bytes	SRAM用于整个端点缓冲. 参考5.4.4.7 的端点SRAM结构和描述.

5.6.6 寄存器描述

USB中断使能寄存器(USB_INTEN)

寄存器	偏移量	R/W	描述	复位后的值
USB_INTEN	USB_BA+0x000	R/W	USB 中断使能寄存器	0x0000_0000

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

Bits	描述	
[31:16]	**保留**	保留
[15]	**INNAK_EN**	**激活收到IN token的时候的NAK 功能及其状态** 1 =当它置1并且收到INtoken时应答NAK，将发生INNAK中断，NAK状态被更新到端点状态寄存器USB_EPSTS，同时将发生中断（如果使能了USB事件中断的话） 0 =当它置0并且收到IN token应答NAK时不会发生中断，NAK状态不会被更新到端点状态寄存器USB_EPSTS。
[14:9]	**保留**	保留
[8]	**WAKEUP_EN**	**唤醒功能使能控制** 1 = 使能USB 唤醒CPU功能 0 = 禁止USB 唤醒CPU功能
[7:4]	**保留**	保留
[3]	**WAKEUP_IE**	**使能USB 唤醒CPU中断** 1 = 使能唤醒CPU中断 0 = 禁止唤醒CPU中断
[2]	**FLDET_IE**	**使能插拔检测中断** 1 = 使能插拔检测中断 0 = 禁止插拔检测中断
[1]	**USB_IE**	**使能USB 事件中断** 1 = 使能 USB 事件中断

		0 = 禁止USB 事件中断
[0]	**BUS_IE**	**使能总线事件中断** 1 = 使能总线事件中断 0 = 禁止总线事件中断

USB 中断事件状态寄存器(USB_INTSTS)

该寄存器是USB中断事件的标志寄存器，通过向相应位写1实现清零.

寄存器	偏移量	R/W	描述	复位后的值
USB_INTSTS	USB_BA+0x004	R/W	USB中断事件标志	0x0000_0000

31	30	29	28	27	26	25	24
SETUP	保留						
23	22	21	20	19	18	17	16
EPEVT7	EPEVT6	EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留				WAKEUP_STS	FLDET_STS	USB_STS	BUS_STS

Bits	描述	
[31]	SETUP	**Setup事件状态** 1 = Setup事件发生，向USB_INTSTS[31]写1清标志 0 = 无Setup事件
[30:24]	保留	保留
[23]	EPEVT7	**端点 7 的 USB 事件状态** 1 =端点7发生USB事件, 检查USB_EPSTS[31:29] 可知发生何种USB事件, 通过向 USB_INTSTS[23] 或USB_INTSTS[1] 写1清零. 0 =端点7没有事件发生.
[22]	EPEVT6	**端点 6 的 USB 事件状态** 1 =端点6发生USB事件, 检查USB_EPSTS[28:26] 可知发生何种USB事件, 通过向 USB_INTSTS[22] 或USB_INTSTS[1] 写1清零 0 =端点6没有事件发生.
[21]	EPEVT5	**端点 5的 USB 事件状态** 1 = 端点5发生USB事件, 检查 USB_EPSTS[25:23] 可知发生何种USB事件, 通过向 USB_INTSTS[21] 或 USB_INTSTS[1]写1清零 0 = 端点5没有事件发生
[20]	EPEVT4	**端点 4的USB 事件状态** 1 =端点4发生USB事件, 检查 USB_EPSTS[22:20]可知发生何种USB事件, 通过向 USB_INTSTS[20] 或 USB_INTSTS[1]写1清零 0 =端点4没有事件发生

[19]	**EPEVT3**	**端点 3's USB 事件状态** 1 =端点3发生USB事件, 检查 USB_EPSTS[19:17]可知发生何种USB事件, 通过向 USB_INTSTS[19] 或 USB_INTSTS[1]写1清零 0 = 端点3没有事件发生
[18]	**EPEVT2**	**端点 2's USB 事件状态** 1 =端点2发生USB事件, 检查 USB_EPSTS[16:14]可知发生何种USB事件, 通过向 USB_INTSTS[18] 或 USB_INTSTS[1]写1清零 0 =端点2没有事件发生
[17]	**EPEVT1**	**端点 1's USB 事件状态** 1 = 端点1发生USB事件, 检查 USB_EPSTS[13:11]可知发生何种USB事件, 通过向 USB_INTSTS[17] 或 USB_INTSTS[1]写1清零 0 = 端点1没有事件发生
[16]	**EPEVT0**	**端点 0's USB 事件状态** 1 =端点0发生USB事件, 检查 USB_EPSTS[10:8]可知发生何种USB事件, 通过向 USB_INTSTS[16] 或 USB_INTSTS[1]写1清零 0 =端点1没有事件发生
[15:4]	**保留**	保留
[3]	**WAKEUP_STS**	**唤醒中断状态** 1 =唤醒事件发生,向USB_INTSTS[3]写1清零 0 = 无唤醒中断状态
[2]	**FLDET_STS**	**插拔检测中断状态** 1 = USB总线上发生插入/拔除事件，向USB_INTSTS[2]写1清零. 0 = USB总线上没有插入/拔除事件
[1]	**USB_STS**	**USB 事件中断状态** USB 事件包括Setup Token, IN Token, OUT ACK, ISO IN, or ISO OUT. 1 = USB 事件发生, 检查EPSTS0~5[2:0] 可知何种USB事件发生, 向USB_INTSTS[1] 或 EPSTS0~5 和 SETUP (USB_INTSTS[31])写1清零 0 = 无USB事件发生
[0]	**BUS_STS**	**总线中断状态** 总线事件意味着总线上有挂起或重新开始功能. 1 = 总线事件发生; 检查USB_ATTR[3:0] 可知发生何种总线事件，向 USB_INTSTS[0] 写1清零. 0 = 无总线事件发生

USB 设备功能地址寄存器 (USB_FADDR)

一个7比特的值用作USB总线上的设备地址.

寄存器	偏移量	R/W	描述	复位后的值
USB_FADDR	USB_BA+0x008	R/W	USB设备功能地址寄存器	0x0000_0000

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

Bits	描述	
[31:7]	保留	保留
[6:0]	FADDR	USB设备地址。在主机分配地址之后，把地址写在这个寄存器

USB 端点状态寄存器 (USB_EPSTS)

寄存器	偏移量	R/W	描述	复位后的值
USB_EPSTS	USB_BA+0x00C	R	USB端点状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
EPSTS7[2:1]			EPSTS6[2:1]			EPSTS5[2:1]	
23	22	21	20	19	18	17	16
EPSTS5[0]	EPSTS4[2:0]			EPSTS3[2:0]			EPSTS2[2]
15	14	13	12	11	10	9	8
EPSTS2[1:0]		EPSTS1[2:0]			EPSTS0[2:0]		
7	6	5	4	3	2	1	0
OVERRUN	保留						

Bits	描述	
[31:29]	EPSTS7	**端点 7 总线状态** 这些位用于表示该端点当前的状态. 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[28:26]	EPSTS6	**端点 6 总线状态** 这些位用于表示该端点当前的状态. 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[25:23]	EPSTS5	**端点 5 总线状态** 这些位用于表示该端点当前的状态 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK

		011 = Setup ACK 111 = Isochronous transfer end
[22:20]	**EPSTS4**	**端点 4 总线状态** 这些位用于表示该端点当前的状态 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[19:17]	**EPSTS3**	**端点3 总线状态** 这些位用于表示该端点当前的状态 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[16:14]	**EPSTS2**	**端点2 总线状态** 这些位用于表示该端点当前的状态 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[13:11]	**EPSTS1**	**端点1总线状态** 这些位用于表示该端点当前的状态 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[10:8]	**EPSTS0**	**端点0 总线状态** 这些位用于表示该端点当前的状态 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK

		110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[7]	**OVERRUN**	**Overrun** 表示接收到的数据超过最大值. 1 = 表示主机Out Data大于寄存器MXPLD 的Max Payload 或Setup Data大于8 Bytes 0 = 没有溢出
[6:0]	**保留**	保留

USB 总线状态和属性寄存器(USB_ATTR)

寄存器	偏移量	R/W	描述	复位后的值
USB_ATTR	USB_BA+0x010	R/W	USB总线状态和属性寄存器	0x0000_0040

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留					BYTEM	PWRDN	DPPU_EN
7	6	5	4	3	2	1	0
USB_EN	保留	RWAKEUP	PHY_EN	TIMEOUT	RESUME	SUSPEND	USBRST

Bits	描述	
[31:11]	保留	保留
[10]	BYTEM	**CPU 访问USB SRAM 字节模式选择** 1 = Byte Mode：CPU只能以字节的方式访问USB SRAM. 0 = Word Mode：CPU只能以字的方式访问USB SRAM（4 bytes）.
[9]	PWRDN	**PHY收发器掉电，低电平有效** 1 = 打开PHY收发器相关电路 0 = 关闭PHY收发器相关电路
[8]	DPPU_EN	**使能USB_DP 的上拉电阻** 1 = USB_DP 的上拉电阻有效 0 = 禁止USB_DP 的上拉电阻
[7]	USB_EN	**使能USB 控制器** 1 = 使能USB 控制器 0 = 禁止USB控制器
[6]	保留	保留
[5]	RWAKEUP	**远程唤醒** 1 = 使USB总线到K状态 (USB_DP 拉低, USB_DM: 拉高), 用于远程唤醒 0 = 将USB总线由K状态释放 设备远程唤醒主机时，先将设备切到K状态，延迟一会儿将K状态释放
[4]	PHY_EN	**使能PHY 发送器功能** 1 = 使能PHY 送发器功能

		0 = 禁止PHY 发送器功能
[3]	**TIMEOUT**	**超时状态** 1 = 总线没有响应超过18 bits 时间 0 = 没有超时 该位只读
[2]	**RESUME**	**重启状态** 1 = 从挂起状态重启 0 = 没有总线重启 该位只读
[1]	**SUSPEND**	**挂起状态** 1 = 总线空闲超过 3ms, 或线缆拔出或主机休眠 0 = 总线没有挂起 该位只读
[0]	**USBRST**	**USB 复位状态** 1 = SE0 (single-ended 0)超过2.5us 总线复位 0 = 总线无复位 该位只读

插拔检测寄存器(USB_FLDET)

寄存器	偏移量	R/W	描述	复位后的值
USB_FLDET	USB_BA+0x014	R	悬空检测寄存器	0x0000_0000

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

Bits	描述	
[31:1]	**保留**	保留
[0]	**FLDET**	**设备悬空检测** 1 = USB控制器接入总线时，该位置1 0 = USB控制器没有接入USB 主机

Setup缓存偏移寄存器(USB_BUFSEG)

仅用于Setup token.

寄存器	偏移量	R/W	描述	复位后的值
USB_BUFSEG	USB_BA+0x018	R/W	Setup 标志的缓存偏移寄存器	0x0000_0000

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

Bits	描述	
[31:9]	**保留**	保留
[8:3]	**BUFSEG**	用于Setup token的USB SRAM中的有效的开始地址，开始地址为： USB_SRAM 地址 + { BUFSEG[8:3], 3'b000} USB_SRAM 地址 = USB_BA + 0x100h. 注: 只用于Setup token .
[2:0]	**保留**	保留

缓存偏移寄存器(BUFSEGx) x = 0~7

寄存器	偏移量	R/W	描述	复位后的值
USB_BUFSEG0	USB_BA+0x020	R/W	端点0的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG1	USB_BA+0x030	R/W	端点1的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG2	USB_BA+0x040	R/W	端点2的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG3	USB_BA+0x050	R/W	端点3的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG4	USB_BA+0x060	R/W	端点4的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG5	USB_BA+0x070	R/W	端点5的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG0	USB_BA+0x500	R/W	端点0的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG1	USB_BA+0x510	R/W	端点1的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG2	USB_BA+0x520	R/W	端点2的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG3	USB_BA+0x530	R/W	端点3的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG4	USB_BA+0x540	R/W	端点4的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG5	USB_BA+0x550	R/W	端点5的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG6	USB_BA+0x560	R/W	端点6的缓冲偏移寄存器	0x0000_0000
USB_BUFSEG7	USB_BA+0x570	R/W	端点7的缓冲偏移寄存器	0x0000_0000

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

Bits	描述	
[31:9]	**保留**	保留
[8:3]	**BUFSEGx**	每个端点在USB SRAM中的偏移，端点有效的开始地址是： USB_SRAM 地址 + { BUFSEG[8:3], 3'b000} USB_SRAM 地址 = USB_BA + 0x100h.

		参考5.4.4.7 端点的SRAM结构及其说明.
[2:0]	**保留**	保留

最大有效载荷寄存器(USB_MXPLDx) x = 0~7

寄存器	偏移量	R/W	描述	复位后的值
USB_MXPLD0	USB_BA+0x024	R/W	端点0最大有效载荷	0x0000_0000
USB_MXPLD1	USB_BA+0x034	R/W	端点1最大有效载荷	0x0000_0000
USB_MXPLD2	USB_BA+0x044	R/W	端点2最大有效载荷	0x0000_0000
USB_MXPLD3	USB_BA+0x054	R/W	端点3最大有效载荷	0x0000_0000
USB_MXPLD4	USB_BA+0x064	R/W	端点4最大有效载荷	0x0000_0000
USB_MXPLD5	USB_BA+0x074	R/W	端点5最大有效载荷	0x0000_0000
USB_MXPLD0	USB_BA+0x504	R/W	端点0最大有效载荷	0x0000_0000
USB_MXPLD1	USB_BA+0x514	R/W	端点1最大有效载荷	0x0000_0000
USB_MXPLD2	USB_BA+0x524	R/W	端点2最大有效载荷	0x0000_0000
USB_MXPLD3	USB_BA+0x534	R/W	端点3最大有效载荷	0x0000_0000
USB_MXPLD4	USB_BA+0x544	R/W	端点4最大有效载荷	0x0000_0000
USB_MXPLD5	USB_BA+0x554	R/W	端点5最大有效载荷	0x0000_0000
USB_MXPLD6	USB_BA+0x564	R/W	端点6最大有效载荷	0x0000_0000
USB_MXPLD7	USB_BA+0x574	R/W	端点7最大有效载荷	0x0000_0000

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

Bits	描述	
[31:9]	**保留**	保留
[8:0]	**MXPLD**	**最大有效载荷** 用于定义发送到主机(IN token)的数据长度或从主机接收到(OUT token)的实际数据长度. 也用于表示端点准备好发送(IN token)或接收数据(OUT token).

		(1). CPU写该寄存器, IN token, MXPLD 的值用于定义要发送的数据长度并表示数据缓冲已经准备好. OUT token, 表示控制器准备好接收主机的数据，MXPLD 的值表示可以从主机接收的最大数据长度. (2). CPU读该寄存器, IN token, MXPLD 的值 表示发送到主机的数据长度 OUT token, MXPLD 的值表示从主机实际接收到的数据长度. 注：一旦MXPLD 被写，数据包将在IN/OUT token到达后立即发送/接收.

配置寄存器(USB_CFGx) x = 0~7

寄存器	偏移量	R/W	描述	复位后的值
USB_CFG0	USB_BA+0x028	R/W	端点0的配置寄存器	0x0000_0000
USB_CFG1	USB_BA+0x038	R/W	端点1的配置寄存器	0x0000_0000
USB_CFG2	USB_BA+0x048	R/W	端点2的配置寄存器	0x0000_0000
USB_CFG3	USB_BA+0x058	R/W	端点3的配置寄存器	0x0000_0000
USB_CFG4	USB_BA+0x068	R/W	端点4的配置寄存器	0x0000_0000
USB_CFG5	USB_BA+0x078	R/W	端点5的配置寄存器	0x0000_0000
USB_CFG0	USB_BA+0x508	R/W	端点0的配置寄存器	0x0000_0000
USB_CFG1	USB_BA+0x518	R/W	端点1的配置寄存器	0x0000_0000
USB_CFG2	USB_BA+0x528	R/W	端点2的配置寄存器	0x0000_0000
USB_CFG3	USB_BA+0x538	R/W	端点3的配置寄存器	0x0000_0000
USB_CFG4	USB_BA+0x548	R/W	端点4的配置寄存器	0x0000_0000
USB_CFG5	USB_BA+0x558	R/W	端点5的配置寄存器	0x0000_0000
USB_CFG6	USB_BA+0x568	R/W	端点6的配置寄存器	0x0000_0000
USB_CFG7	USB_BA+0x578	R/W	端点7的配置寄存器	0x0000_0000

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

Bits	描述	
[31:10]	**保留**	保留
[9]	**CSTALL**	**清 STALL 响应** 1 =在setup阶段允许设备清stall 0 =禁止自动清 stall

[8]	**保留**	保留
[7]	**DSQ_SYNC**	**数据序列同步** 1 = DATA1 PID 0 = DATA0 PID 在IN token的传输中指明DATA0 或 DATA1 PID. H/W 基于该位自动触发.
[6:5]	**STATE**	**端点状态** 00 = 端点被关闭 01 = 输出端点 10 = 输入端点 11 = 无定义
[4]	**ISOCH**	**Isochronous 端点** 该位设置端点为Isochronous 端点, 无握手. 1 = Isochronous 端点 0 = 非Isochronous 端点
[3:0]	**EP_NUM**	**端点号** 用于定义该端点的端点号

额外配置寄存器(USB_CFGPx) x = 0~5

寄存器	偏移量	R/W	描述	复位后的值
USB_CFGP0	USB_BA+0x02C	R/W	设置端点0自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP1	USB_BA+0x03C	R/W	设置端点1自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP2	USB_BA+0x04C	R/W	设置端点2自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP3	USB_BA+0x05C	R/W	设置端点3自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP4	USB_BA+0x06C	R/W	设置端点4自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP5	USB_BA+0x07C	R/W	设置端点5自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP0	USB_BA+0x50C	R/W	设置端点0自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP1	USB_BA+0x51C	R/W	设置端点1自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP2	USB_BA+0x52C	R/W	设置端点2自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP3	USB_BA+0x53C	R/W	设置端点3自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP4	USB_BA+0x54C	R/W	设置端点4自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP5	USB_BA+0x55C	R/W	设置端点5自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP6	USB_BA+0x56C	R/W	设置端点6自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000
USB_CFGP7	USB_BA+0x57C	R/W	设置端点7自动回STALL与清除In/Out准备好控制寄存器	0x0000_0000

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

保留							
7	6	5	4	3	2	1	0
保留						SSTALL	CLRRDY

Bits	描述	
[31:2]	保留	保留
[1]	SSTALL	**设置 STALL** 1 = 设置设备自动响应STALL 0 = 禁止设备响应STALL
[0]	CLRRDY	**清除准备好标志** 用户设置寄存器MXPLD, 表示端点准备好发送或接收数据. 如果用户想要清除准备好标志，用户可以设置该位为1，该位自动清0. IN token, 写1清IN token时发送数据到USB的准备好信号. OUT token, 写1清OUT token时从USB接收数据的准备好信号. 该位只能写入1，且其返回值始终为0。

驱动 SE0 寄存器(USB_DRVSE0)

寄存器	偏移量	R/W	描述	复位后的值
USB_DRVSE0	USB_BA+0x090	R/W	USB PHY驱动SE0	0x0000_0001

31	30	29	28	27	26	25	24
保留							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
保留							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
保留							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
保留							**DRVSE0**

Bits	描述	
[31:1]	**保留**	保留
[0]	**DRVSE0**	**USB总线驱动**SE0 当USB_DP 和 USB_DM 两根线都拉低时，为SE0状态 1 = USB PHY驱动SE0状态 0 = 无

5.7 通用I/O (GPIO)

5.7.1 概述

NuMicro™ NUC123最多有 47 个通用 I/O 引脚和其他功能引脚共享，实际的I/O数量取决于芯片的配置。47个引脚分配在GPIOA, GPIOB, GPIOC, GPIOD 与 GPIOF五个端口上。GPIOA有6个引脚PA[15:10]，GPIOB有15个引脚PB[15:12]和PB[10:0]，GPIOC有12个引脚PC[13:8]和PC[5:0]，GPIOD有10个引脚PD[11:8]和PD[5:0]，GPIOF有4个引脚PF[3:0]。每个引脚都是独立的，都有相应的寄存器位来控制引脚功能模式与数据。

I/O引脚的I/O类型可由软件独立地配置为输入，输出，开漏或准双向模式。复位之后，所有引脚的I/O类型均为准双向模式，端口数据寄存器GPIOx_DOUT[15:0]的值为0x000_FFFF。每个I/O引脚有一个阻值为110KΩ~300KΩ的弱上拉电阻接到VDD 上，VDD从5.0 V 到2.5 V。

5.7.2 特性

- 四种 I/O 模式:
  - 准双向模式
  - 推挽输出
  - 开漏输出
  - 高阻态输入
- 可选的TTL/Schmitt 触发输入
- I/O可以配置为边沿/电平触发的中断源
- 支持high driver 和 high sink的 IO 模式

5.7.3 功能描述

5.7.3.1

输入模式说明

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

5.7.3.2

输出模式说明

设置 GPIOx_PMD (PMDn[1:0]) 为 01b，GPIOx port [n] 为输出模式，I/O支持数字输出功能，有source/sink电流能力。GPIOx_DOUT 相应位bit[n]的值被送到相应引脚上。

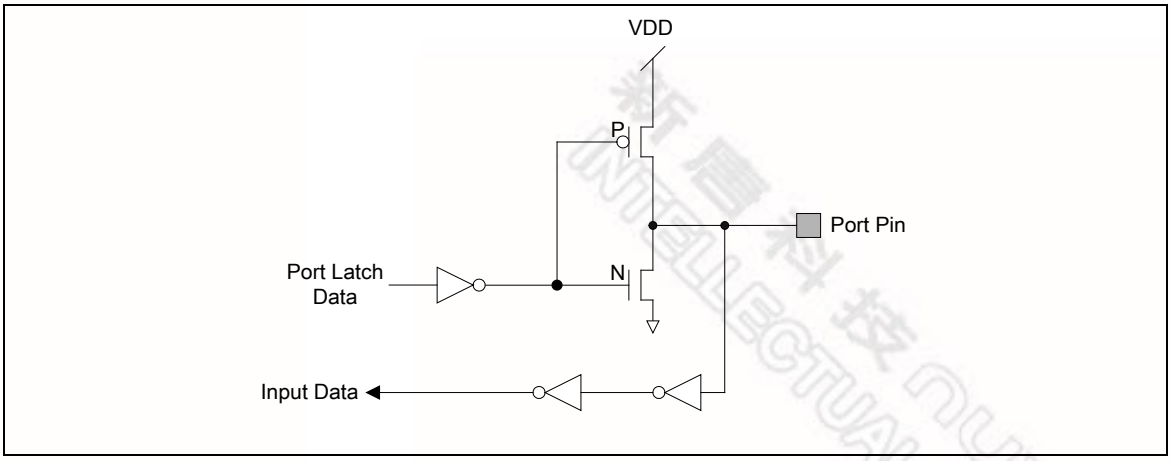


图 5-23 推挽输出

5.7.3.3 *开漏模式说明*

设置 GPIOx_PMD (PMDn[1:0]) 为 10b，GPIOx port [n]为开漏模式且I/O引脚数字输出功能仅支持灌电流，驱动到高电平需要一个外加上拉电阻。如果GPIOx_DOUT 相应位bit [n]的值为“0”，引脚上输出低。如果GPIOx_DOUT 相应位bit [n]的值为“1” ，该引脚输出为高，可以由外部上拉电阻控制。

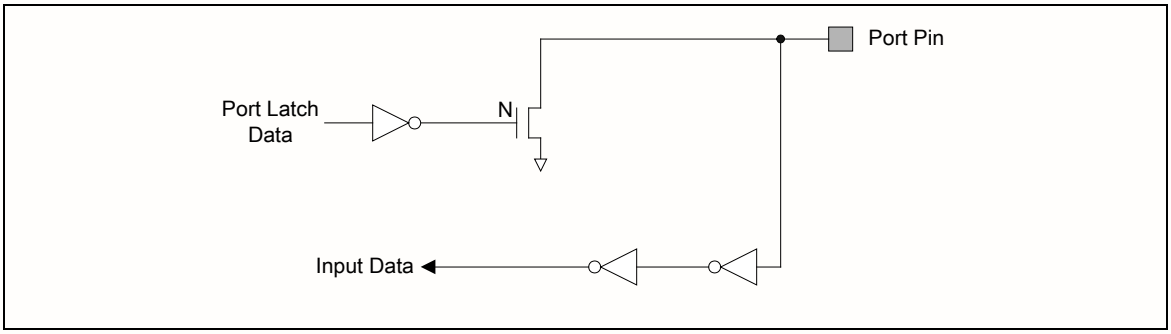


图 5-24 开漏输出

5.7.3.4 *准双向模式说明*

设置GPIOx_PMD (PMDn[1:0])为 11b，GPIOx port [n] 为准双向模式，I/O同时支持数字输出和输入功能，但source电流仅达数百uA。 要实现数字输入，需要先将GPIOx_DOUT 相应位置1。准双向输出是80C51及其派生产品常见的模式。若GPIOx_DOUT相应位bit[n]为”0” ，引脚上输出为“低” 。若GPIOx_DOUT相应位bit[n]为”1” ，该引脚将检测引脚值。若引脚值为高，没有任何动作，若引脚值为低，在该引脚上将驱动2个时钟周期的高电平，然后禁止强输出驱动，其后引脚状态由内部上拉电阻控制。注: 准双端模式source电流的大小仅有200 uA到30 uA(相应VDD的电压从5.0 V到2.5 V) 。

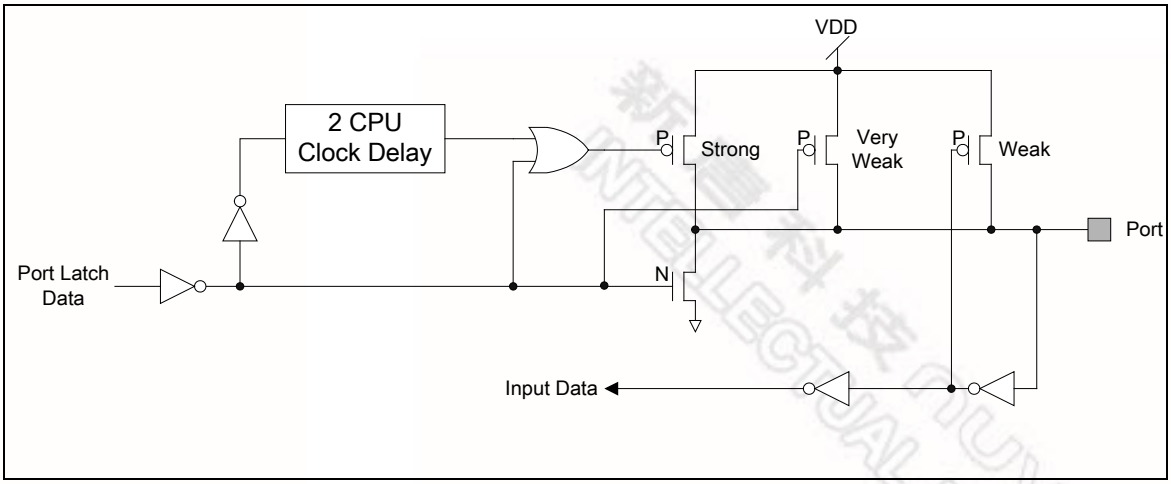


图 5-25 准双向 I/O 模式

5.7.4 寄存器映射

R: read only, **W:** write only, **R/W:** both read and write

寄存器	偏移量	R/W	描述	复位后的值
GP 基地址: **GP_BA = 0x5000_4000**				
GPIOA_PMD	GP_BA+0x000	R/W	GPIO 端口 A 引脚模式控制	0xFFFF_FFFF
GPIOA_OFFD	GP_BA+0x004	R/W	GPIO 端口 A 关闭数字通路寄存器	0x0000_0000
GPIOA_DOUT	GP_BA+0x008	R/W	GPIO 端口 A 数据输出寄存器	0x0000_FFFF
GPIOA_DMASK	GP_BA+0x00C	R/W	GPIO 端口 A 数据输出写屏蔽	0x0000_0000
GPIOA_PIN	GP_BA+0x010	R	GPIO 端口 A 引脚状态	0x0000_XXXX
GPIOA_DBEN	GP_BA+0x014	R/W	GPIO 端口 A 去抖动使能	0x0000_0000
GPIOA_IMD	GP_BA+0x018	R/W	GPIO 端口 A 中断模式控制	0x0000_0000
GPIOA_IEN	GP_BA+0x01C	R/W	GPIO 端口 A 中断使能	0x0000_0000
GPIOA_ISRC	GP_BA+0x020	R/W	GPIO 端口 A 中断源标志	0xXXXX_XXXX
GPIOB_PMD	GP_BA+0x040	R/W	GPIO 端口 B 引脚模式控制	0xFFFF_FFFF
GPIOB_OFFD	GP_BA+0x044	R/W	GPIO 端口 B 关闭数字通路寄存器	0x0000_0000
GPIOB_DOUT	GP_BA+0x048	R/W	GPIO 端口 B 数据输出寄存器	0x0000_FFFF
GPIOB_DMASK	GP_BA+0x04C	R/W	GPIO 端口 B 数据输出写屏蔽	0x0000_0000
GPIOB_PIN	GP_BA+0x050	R	GPIO 端口 B 引脚状态	0x0000_XXXX
GPIOB_DBEN	GP_BA+0x054	R/W	GPIO 端口 B 去抖动使能	0x0000_0000
GPIOB_IMD	GP_BA+0x058	R/W	GPIO 端口 B 中断模式控制	0x0000_0000
GPIOB_IEN	GP_BA+0x05C	R/W	GPIO 端口 B 中断使能	0x0000_0000
GPIOB_ISRC	GP_BA+0x060	R/W	GPIO 端口 B 中断源标志	0xXXXX_XXXX
GPIOC_PMD	GP_BA+0x080	R/W	GPIO 端口 C 引脚模式控制	0xFFFF_FFFF
GPIOC_OFFD	GP_BA+0x084	R/W	GPIO 端口 C 关闭数字通路寄存器	0x0000_0000
GPIOC_DOUT	GP_BA+0x088	R/W	GPIO 端口 C 数据输出寄存器	0x0000_FFFF
GPIOC_DMASK	GP_BA+0x08C	R/W	GPIO 端口 C 数据输出写屏蔽	0x0000_0000
GPIOC_PIN	GP_BA+0x090	R	GPIO 端口 C 引脚状态	0x0000_XXXX
GPIOC_DBEN	GP_BA+0x094	R/W	GPIO 端口 C 去抖动使能	0x0000_0000
GPIOC_IMD	GP_BA+0x098	R/W	GPIO 端口 C 中断模式控制	0x0000_0000

GPIOC_IEN	GP_BA+0x09C	R/W	GPIO 端口 C 中断使能	0x0000_0000
GPIOC_ISRC	GP_BA+0x0A0	R/W	GPIO 端口 C 中断源标志	0xXXXX_XXXX
GPIOD_PMD	GP_BA+0x0C0	R/W	GPIO 端口 D 引脚模式控制	0xFFFF_FFFF
GPIOD_OFFD	GP_BA+0x0C4	R/W	GPIO 端口 D 关闭数字通路寄存器	0x0000_0000
GPIOD_DOUT	GP_BA+0x0C8	R/W	GPIO 端口 D 数据输出寄存器	0x0000_FFFF
GPIOD_DMASK	GP_BA+0x0CC	R/W	GPIO 端口 D 数据输出写屏蔽	0x0000_0000
GPIOD_PIN	GP_BA+0x0D0	R	GPIO 端口 D 引脚状态	0x0000_XXXX
GPIOD_DBEN	GP_BA+0x0D4	R/W	GPIO 端口 D 去抖动使能	0x0000_0000
GPIOD_IMD	GP_BA+0x0D8	R/W	GPIO 端口 D 中断模式控制	0x0000_0000
GPIOD_IEN	GP_BA+0x0DC	R/W	GPIO 端口 D 中断使能	0x0000_0000
GPIOD_ISRC	GP_BA+0x0E0	R/W	GPIO 端口 D 中断源标志	0xXXXX_XXXX
GPIOF_PMD	GP_BA+0x140	R/W	GPIO 端口 F引脚模式控制	0xFFFF_FFFF
GPIOF_OFFD	GP_BA+0x144	R/W	GPIO 端口 F关闭数字通路寄存器	0x0000_0000
GPIOF_DOUT	GP_BA+0x148	R/W	GPIO 端口 F 数据输出寄存器	0x0000_FFFF
GPIOF_DMASK	GP_BA+0x14C	R/W	GPIO 端口 F 数据输出写屏蔽	0x0000_0000
GPIOF_PIN	GP_BA+0x150	R	GPIO 端口 F引脚状态	0x0000_XXXX
GPIOF_DBEN	GP_BA+0x154	R/W	GPIO 端口 F 去抖动使能	0x0000_0000
GPIOF_IMD	GP_BA+0x158	R/W	GPIO 端口 F 中断模式控制	0x0000_0000
GPIOF_IEN	GP_BA+0x15C	R/W	GPIO 端口 F 中断使能	0x0000_0000
GPIOF_ISRC	GP_BA+0x160	R/W	GPIO 端口 F 中断源标志	0xXXXX_XXXX
DBNCECON	GP_BA+0x180	R/W	去抖动周期控制	0x0000_0020
GPIOA10_DOUT	GP_BA+0x228	R/W	GPIO PA.10 数据输出/输入控制	0x0000_0001
GPIOA11_DOUT	GP_BA+0x22C	R/W	GPIO PA.11 数据输出/输入控制	0x0000_0001
GPIOA12_DOUT	GP_BA+0x230	R/W	GPIO PA.12 数据输出/输入控制	0x0000_0001
GPIOA13_DOUT	GP_BA+0x234	R/W	GPIO PA.13 数据输出/输入控制	0x0000_0001
GPIOA14_DOUT	GP_BA+0x238	R/W	GPIO PA.14 数据输出/输入控制	0x0000_0001
GPIOA15_DOUT	GP_BA+0x23C	R/W	GPIO PA.15 数据输出/输入控制	0x0000_0001
GPIOB0_DOUT	GP_BA+0x240	R/W	GPIO PB.0 数据输出/输入控制	0x0000_0001
GPIOB1_DOUT	GP_BA+0x244	R/W	GPIO PB.1 数据输出/输入控制	0x0000_0001
GPIOB2_DOUT	GP_BA+0x248	R/W	GPIO PB.2 数据输出/输入控制	0x0000_0001

GPIOB3_DOUT	GP_BA+0x24C	R/W	GPIO PB.3 数据输出/输入控制	0x0000_0001
GPIOB4_DOUT	GP_BA+0x250	R/W	GPIO PB.4 数据输出/输入控制	0x0000_0001
GPIOB5_DOUT	GP_BA+0x254	R/W	GPIO PB.5 数据输出/输入控制	0x0000_0001
GPIOB6_DOUT	GP_BA+0x258	R/W	GPIO PB.6 数据输出/输入控制	0x0000_0001
GPIOB7_DOUT	GP_BA+0x25C	R/W	GPIO PB.7 数据输出/输入控制	0x0000_0001
GPIOB8_DOUT	GP_BA+0x260	R/W	GPIO PB.8 数据输出/输入控制	0x0000_0001
GPIOB9_DOUT	GP_BA+0x264	R/W	GPIO PB.9 数据输出/输入控制	0x0000_0001
GPIOB10_DOUT	GP_BA+0x268	R/W	GPIO PB.10 数据输出/输入控制	0x0000_0001
GPIOB12_DOUT	GP_BA+0x270	R/W	GPIO PB.12 数据输出/输入控制	0x0000_0001
GPIOB13_DOUT	GP_BA+0x274	R/W	GPIO PB.13 数据输出/输入控制	0x0000_0001
GPIOB14_DOUT	GP_BA+0x278	R/W	GPIO PB.14 数据输出/输入控制	0x0000_0001
GPIOB15_DOUT	GP_BA+0x27C	R/W	GPIO PB.15 数据输出/输入控制	0x0000_0001
GPIOC0_DOUT	GP_BA+0x280	R/W	GPIO PC.0 数据输出/输入控制	0x0000_0001
GPIOC1_DOUT	GP_BA+0x284	R/W	GPIO PC.1 数据输出/输入控制	0x0000_0001
GPIOC2_DOUT	GP_BA+0x288	R/W	GPIO PC.2 数据输出/输入控制	0x0000_0001
GPIOC3_DOUT	GP_BA+0x28C	R/W	GPIO PC.3 数据输出/输入控制	0x0000_0001
GPIOC4_DOUT	GP_BA+0x290	R/W	GPIO PC.4 数据输出/输入控制	0x0000_0001
GPIOC5_DOUT	GP_BA+0x294	R/W	GPIO PC.5 数据输出/输入控制	0x0000_0001
GPIOC8_DOUT	GP_BA+0x2A0	R/W	GPIO PC.8 数据输出/输入控制	0x0000_0001
GPIOC9_DOUT	GP_BA+0x2A4	R/W	GPIO PC.9 数据输出/输入控制	0x0000_0001
GPIOC10_DOUT	GP_BA+0x2A8	R/W	GPIO PC.10 数据输出/输入控制	0x0000_0001
GPIOC11_DOUT	GP_BA+0x2AC	R/W	GPIO PC.11 数据输出/输入控制	0x0000_0001
GPIOC12_DOUT	GP_BA+0x2B0	R/W	GPIO PC.12 数据输出/输入控制	0x0000_0001
GPIOC13_DOUT	GP_BA+0x2B4	R/W	GPIO PC.13 数据输出/输入控制	0x0000_0001
GPIOD0_DOUT	GP_BA+0x2C0	R/W	GPIO PD.0 数据输出/输入控制	0x0000_0001
GPIOD1_DOUT	GP_BA+0x2C4	R/W	GPIO PD.1 数据输出/输入控制	0x0000_0001
GPIOD2_DOUT	GP_BA+0x2C8	R/W	GPIO PD.2 数据输出/输入控制	0x0000_0001
GPIOD3_DOUT	GP_BA+0x2CC	R/W	GPIO PD.3 数据输出/输入控制	0x0000_0001
GPIOD4_DOUT	GP_BA+0x2D0	R/W	GPIO PD.4 数据输出/输入控制	0x0000_0001
GPIOD5_DOUT	GP_BA+0x2D4	R/W	GPIO PD.5 数据输出/输入控制	0x0000_0001

GPIOD8_DOUT	GP_BA+0x2E0	R/W	GPIO PD.8 数据输出/输入控制	0x0000_0001
GPIOD9_DOUT	GP_BA+0x2E4	R/W	GPIO PD.9 数据输出/输入控制	0x0000_0001
GPIOD10_DOUT	GP_BA+0x2E8	R/W	GPIO PD.10 数据输出/输入控制	0x0000_0001
GPIOD11_DOUT	GP_BA+0x2EC	R/W	GPIO PD.11 数据输出/输入控制	0x0000_0001
GPIOF0_DOUT	GP_BA+0x340	R/W	GPIO PF.0 数据输出/输入控制	0x0000_0001
GPIOF1_DOUT	GP_BA+0x344	R/W	GPIO PF.1 数据输出/输入控制	0x0000_0001
GPIOF2_DOUT	GP_BA+0x348	R/W	GPIO PF.2 数据输出/输入控制	0x0000_0001
GPIOF3_DOUT	GP_BA+0x34C	R/W	GPIO PF.3 数据输出/输入控制	0x0000_0001

5.7.5 寄存器描述

GPIO 端口 [A/B/C/D/F] I/O 模式控制 (GPIOx_PMD)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_PMD	GP_BA+0x000	R/W	GPIO 端口 A Pin I/O 模式控制	0xFFFF_FFFF
GPIOB_PMD	GP_BA+0x040	R/W	GPIO 端口 B Pin I/O 模式控制	0xFFFF_FFFF
GPIOC_PMD	GP_BA+0x080	R/W	GPIO 端口 C Pin I/O 模式控制	0xFFFF_FFFF
GPIOD_PMD	GP_BA+0x0C0	R/W	GPIO 端口 D Pin I/O 模式控制	0xFFFF_FFFF
GPIOF_PMD	GP_BA+0x140	R/W	GPIO 端口 F Pin I/O 模式控制	0x0000_00FF

31	30	29	28	27	26	25	24
PMD15		PMD14		PMD13		PMD12	
23	22	21	20	19	18	17	16
PMD11		PMD10		PMD9		PMD8	
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	
[2n+1:2n]	PMDn	**GPIOx I/O Pin[n] 模式控制** 决定GPIOx的I/O 类型. 00 = GPIO port [n] 引脚为输入模式 01 = GPIO port [n] 引脚为输出模式 10 = GPIO port [n] 引脚为开漏模式 11 = GPIO port [n] 引脚为准双向模式 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留.

GPIO 端口 [A/B/C/D/F] 管脚 关闭 数字通路寄存器 (GPIOx_OFFD)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_OFFD	GP_BA+0x004	R/W	GPIO 端口 A 管脚关闭数字通路使能	0x0000_0000
GPIOB_OFFD	GP_BA+0x044	R/W	GPIO 端口 B 管脚关闭数字通路使能	0x0000_0000
GPIOC_OFFD	GP_BA+0x084	R/W	GPIO 端口 C 管脚关闭数字通路使能	0x0000_0000
GPIOD_OFFD	GP_BA+0x0C4	R/W	GPIO 端口 D 管脚关闭数字通路使能	0x0000_0000
GPIOF_OFFD	GP_BA+0x144	R/W	GPIO 端口 F 管脚关闭数字通路使能	0x0000_0000

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

Bits	描述	
[16:31]	**OFFD**	**GPIOx Pin[n] 关闭数字输入通道使能** 用于控制GPIO的数字输入通路是否使能. 如果输入为模拟信号，用户可以关闭输入通道防止漏电 1 = 关闭IO的数字输入通道(数字输入拉低) 0 = 使能IO数据输入通道 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留
[0:15]	**保留**	保留

GPIO 端口 [A/B/C/D/F] 数据输出值(GPIOx_DOUT)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_DOUT	GP_BA+0x008	R/W	GPIO 端口 A 数据输出值	0x0000_FFFF
GPIOB_DOUT	GP_BA+0x048	R/W	GPIO 端口 B 数据输出值	0x0000_FFFF
GPIOC_DOUT	GP_BA+0x088	R/W	GPIO 端口 C 数据输出值	0x0000_FFFF
GPIOD_DOUT	GP_BA+0x0C8	R/W	GPIO 端口 D 数据输出值	0x0000_FFFF
GPIOF_DOUT	GP_BA+0x148	R/W	GPIO 端口 F 数据输出值	0x0000_000F

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

Bits	描述	
[31:16]	**保留**	保留
[n]	**DOUT[n]**	**GPIOx Pin[n] 输出值** 在GPIO配置成输出，开漏和准双向模式时，控制GPIO相应引脚的状态. 1 = GPIO配置成输出，开漏和准双向模式时，GPIO port [A/B/C/D/F] Pin[n]为高 0 = GPIO配置成输出，开漏和准双向模式时，GPIO port [A/B/C/D/F] Pin[n]为低 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留

GPIO 端口 [A/B/C/D/F] 数据输出写屏蔽(GPIOx _DMASK)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_DMASK	GP_BA+0x00C	R/W	GPIO 端口 A 数据输出写屏蔽	0x0000_0000
GPIOB_DMASK	GP_BA+0x04C	R/W	GPIO 端口 B 数据输出写屏蔽	0x0000_0000
GPIOC_DMASK	GP_BA+0x08C	R/W	GPIO 端口 C 数据输出写屏蔽	0x0000_0000
GPIOD_DMASK	GP_BA+0x0CC	R/W	GPIO 端口 D 数据输出写屏蔽	0x0000_0000
GPIOF_DMASK	GP_BA+0x14C	R/W	GPIO 端口 F 数据输出写屏蔽	0x0000_0000

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

Bits	描述	
[31:16]	**保留**	保留
[n]	**DMASK[n]**	**端口 [A/B/C/D/F] 数据输出写屏蔽** 用于保护相应寄存器GPIOx_DOUT bit[n]. 当设置DMASK bit[n] 为 “1”时，相应DOUT[n] bit 被保护，写信号被屏蔽时，写数据到被保护位无效 1 = 保护相应的GPIO_DOUT[n] 位 0 =相应的GPIO_DOUT[n] 位可以被更新 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留

GPIO 端口 [A/B/C/D/F] 管脚数据(GPIOx _PIN)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_PIN	GP_BA+0x010	R	GPIO 端口 A 管脚数据	0x0000_XXXX
GPIOB_PIN	GP_BA+0x050	R	GPIO 端口 B 管脚数据	0x0000_XXXX
GPIOC_PIN	GP_BA+0x090	R	GPIO 端口 C 管脚数据	0x0000_XXXX
GPIOD_PIN	GP_BA+0x0D0	R	GPIO 端口 D 管脚数据	0x0000_XXXX
GPIOF_PIN	GP_BA+0x150	R	GPIO 端口 F 管脚数据	0x0000_000X

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

Bits	描述	
[31:16]	**保留**	保留
[n]	**PIN[n]**	**端口 [A/B/C/D/F] 管脚状态** 这些位的值为各个GPIO引脚真实状态的反映。如果值为“1”，表示相应引脚状态为高，否则为低 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留

GPIO 端口 [A/B/C/D/F] 去抖动使能(GPIOx _DBEN)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_DBEN	GP_BA+0x014	R/W	GPIO 端口 A 去抖动使能	0xXXXX_0000
GPIOB_DBEN	GP_BA+0x054	R/W	GPIO 端口 B 去抖动使能	0xXXXX_0000
GPIOC_DBEN	GP_BA+0x094	R/W	GPIO 端口 C 去抖动使能	0xXXXX_0000
GPIOD_DBEN	GP_BA+0x0D4	R/W	GPIO 端口 D 去抖动使能	0xXXXX_0000
GPIOF_DBEN	GP_BA+0x154	R/W	GPIO 端口 F 去抖动使能	0xXXXX_0000

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

Bits	描述	
[31:16]	**保留**	保留
[n]	**DBEN[n]**	**端口 [A/B/C/D/F] 输入信号去抖动使能** DBEN[n]用于使能相应位的去抖动功能. 如果输入信号脉冲宽度不能被两个连续的去抖动采样周期所采样，则被视为信号反弹，从而不触发中断. 去抖动时钟源由DBNCECON[4]控制, 一个去抖动周期由DBNCECON[3:0]控制 DBEN[n] 仅用于"边沿触发"中断，不能用于"电平触发"中断 1 = 使能 bit[n] 去抖动功能 0 = 禁用 bit[n] 去抖动功能 去抖动功能对于边沿触发中断有效, 对于电平触发中断模式，去抖动功能使能位不起作用. 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留

GPIO 端口 [A/B/C/D/F] 中断模式控制(GPIOx _IMD)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_IMD	GP_BA+0x018	R/W	GPIO 端口 A 中断模式控制	0x0000_0000
GPIOB_IMD	GP_BA+0x058	R/W	GPIO 端口 B 中断模式控制	0x0000_0000
GPIOC_IMD	GP_BA+0x098	R/W	GPIO 端口 C 中断模式控制	0x0000_0000
GPIOD_IMD	GP_BA+0x0D8	R/W	GPIO 端口 D 中断模式控制	0x0000_0000
GPIOF_IMD	GP_BA+0x158	R/W	GPIO 端口 F 中断模式控制	0x0000_0000

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

Bits	描述	
[31:16]	**保留**	保留
[n]	**IMD[n]**	**端口 [A/B/C/D/F] 边沿或电平检测中断控制** IMD[n] 用于控制电平触发或边沿触发中断. 若为边沿触发中断，触发源可由去抖动控制，如果是电平触发中断，输入源由一个HCLK时钟采样并产生中断 0 = 边沿触发中断 1 = 电平触发中断 如果设置引脚为电平触发模式，则在寄存器GPIOX_IEN中，只能设置一个电平(高电平或者低电平)；若设置了两个电平都触发中断，则设置被忽略，不会产生中断 去抖动功能对于边沿触发中断有效，对于电平触发中断无效. 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留

GPIO 端口 [A/B/C/D/F] 中断使能控制(GPIOx _IEN)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_IEN	GP_BA+0x01C	R/W	GPIO 端口 A 中断使能	0x0000_0000
GPIOB_IEN	GP_BA+0x05C	R/W	GPIO 端口 B 中断使能	0x0000_0000
GPIOC_IEN	GP_BA+0x09C	R/W	GPIO 端口 C 中断使能	0x0000_0000
GPIOD_IEN	GP_BA+0x0DC	R/W	GPIO 端口 D 中断使能	0x0000_0000
GPIOF_IEN	GP_BA+0x15C	R/W	GPIO 端口 F 中断使能	0x0000_0000

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

Bits	描述	
[n+16]	**IR_EN[n]**	**使能端口 [A/B/C/D/F] 输入上升沿或输入高电平中断** IR_EN[n] 用于使能相应GPIO_PIN[n]输入的中断. 置 “1” 也可以使能引脚唤醒功能 当设置 IR_EN[n] 位为 “1”: 如果中断是电平触发模式，输入PIN[n]的状态为高电平时，产生中断. 如果中断是边沿触发模式，输入PIN[n]的状态由低电平到高电平变化时，产生中断. 1 = 使能PIN[n]高电平或由低电平到高电平变化的中断 0 = 禁用PIN[n]高电平或由低电平到高电平变化的中断 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留
[n]	**IF_EN[n]**	**使能端口 [A/B/C/D/F] 输入下降沿或输入低电平的中断** IF_EN[n] 用于使能相应GPIO_PIN[n]输入的中断. 置 “1” 也可以使能引脚唤醒

		功能 当设置 IF_EN[n] 位为 “1”: 如果中断是电平触发模式，输入PIN[n]的状态为低电平时，产生中断. 如果中断是边沿触发模式，输入PIN[n]的状态由高电平到低电平变化时，产生中断. 1 = 使能PIN[n]低电平或由高电平到低电平变化的中断 0 = 禁用PIN[n]低电平或由高电平到低电平变化的中断 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留

GPIO 端口 [A/B/C/D/F] 中断触发源(GPIOx _ISRC)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_ISRC	GP_BA+0x020	R/W	GPIO 端口 A 中断触发源指标	0x0000_XXXX
GPIOB_ISRC	GP_BA+0x060	R/W	GPIO 端口 B 中断触发源指标	0x0000_XXXX
GPIOC_ISRC	GP_BA+0x0A0	R/W	GPIO 端口 C 中断触发源指标	0x0000_XXXX
GPIOD_ISRC	GP_BA+0x0E0	R/W	GPIO 端口 D 中断触发源指标	0x0000_XXXX
GPIOF_ISRC	GP_BA+0x160	R/W	GPIO 端口 F 中断触发源指标	0x0000_000X

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

Bits	描述	
[31:16]	**保留**	保留
[n]	**ISRC[n]**	**端口 [A/B/C/D/F] 中断触发源指标器** 读： 1 = GPIOx[n]产生中断 0 = GPIOx[n]没有中断 写： 1= 清相应的中断标志 0= 无动作 注： GPIOA: 有效值 15~10. 其它的保留. GPIOB: 有效值15~12, 10~0. 其它的保留. GPIOC: 有效值 13~8, 5~0. 其它的保留 GPIOD: 有效值11~8, 5~0. 其它的保留 GPIOF: 有效值 3~0. 其它的保留

中断去抖动周期控制(DBNCECON)

寄存器	偏移量	R/W	描述	复位后的值
DBNCECON	GP_BA+0x180	R/W	外部中断去抖动控制	0x0000_0020

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

Bits	描述	
[5]	ICLK_ON	**中断时钟On模式** 如果GPIO pin[n]中断功能被关闭，设置该位为“0” 将关闭中断产生电路的时钟 1 = 总是使能中断产生电路的时钟 0 = 如果中断GPIOA/B/C/D/F[n]被禁止，关闭相应的电路的时钟
[4]	DBCLKSRC	**去抖动计数器时钟源选择** 1 = 去抖动计数器时钟源为内部 10 KHz 时钟 0 = 去抖动计数器时钟源为 HCLK
[3:0]	DBCLKSEL	**去抖动采样周期选择**

DBCLKSEL	Des□riptio
0	采样中断输入信号, 每 1 clocks一次
1	采样中断输入信号, 每 2 clocks一次
2	采样中断输入信号, 每4 clocks一次
3	采样中断输入信号, 每8 cloc□s一次
4	采样中断输入信号, 每16 clocks一次
5	采样中断输入信号, 每32 clocks一次
6	采样中断输入信号, 每64 clocks一次
7	采样中断输入信号, 每128 clocks一次

		8	采样中断输入信号, 每256 clocks□次	
		9	采样中断输入信号, 每□2*256 clocks一	
		10	采样中断输入信号, 每4*256clocks一次	
		11	采样中断输入信号, 每8*256 cl□c□s一次	
		12	采样中断输入信号, 每16*2□6 c□ocks一次	
		13	采样中断输入信号, 每32*2□6 clocks一次	
		14	采样中断输入信号, 每64*256 clocks一次	
		15	采样中断输入信号, 每128*256 clocks一次	

GPIO 端口 [A/B/C/D/F] I/O 位输出/輸入控制(GPIOxx_DOUT)

寄存器	偏移量	R/W	描述	复位后的值
GPIOAx_DOUT	GP_BA+0x200 - GP_BA+0x23C	R/W	GPIO 端口 A Pin I/O位输出/輸入控制	0x0000_0001
GPIOBx_DOUT	GP_BA+0x240 - GP_BA+0x27C	R/W	GPIO 端口 B Pin I/O位输出/輸入控制	0x0000_0001
GPIOCx_DOUT	GP_BA+0x280 - GP_BA+0x2BC	R/W	GPIO 端口 C Pin I/O位输出/輸入控制	0x0000_0001
GPIODx_DOUT	GP_BA+0x2C0 - GP_BA+0x2FC	R/W	GPIO 端口 D Pin I/O位输出/輸入控制	0x0000_0001
GPIOFx_DOUT	GP_BA+0x340 - GP_BA+0x34C	R/W	GPIO 端口 F Pin I/O位输出/輸入控制	0x0000_0001

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

Bits	描述	
[0]	**GPIOxx_DOUT**	**GPIOxx I/O Pin 位输出/輸入控制** 写该位可以控制一个GPIO引脚的输出值 1 = 设置相应GPIO引脚为高 0 = 设置相应GPIO引脚为低 例如: 写GPIOA0_DOUT即把值写到GPIOA_DOUT[0]位上, 读GPIOA0_DOUT即读取GPIOA_PIN[0]的值.

5.8 I²C总线控制器 (Master/Slave) (I²C)

5.8.1 概述

I²C为双线，双向串行总线，通过简单有效的连线方式实现器件间的数据交换。I²C标准是多主机总线，包括冲突检测和仲裁以防止在两个或多个主机试图同时控制总线时发生的数据损坏。

数据在主机与从机之间通过SCL时钟同步在SDA数据线上实现一字节一字节的同步传输，每个字节为8位长度，一个SCL时钟脉冲传输一个数据位，数据由最高位MSB开始传输，每个传输字节后跟随一个应答位，每个位在SCL为高时采样；因此，SDA线只有在SCL为低时才可以改变，在SCL为高时SDA应保持稳定。当SCL为高时，SDA线上的跳变视为命令（START 或 STOP). 参考 图 5-26 I²C 总线时序。

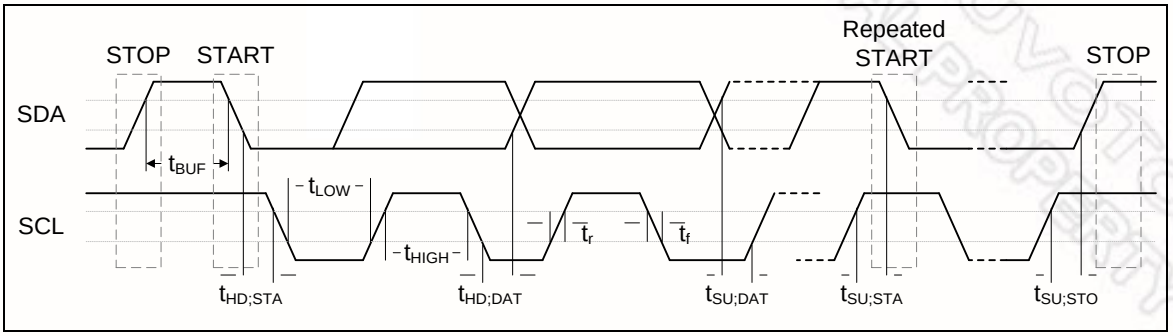


图 5-26 I²C 总线时序

片上I²C逻辑提供符合I²C总线标准的串行标准接口。I²C端口自动处理字节传输，将I2CON的ENS1位设置为1，即可使能该端口. I²C H/W 接口通过SDA与 SCL两个引脚连到I²C总线。当这两个引脚为开漏模式时，用于I²C操作的两个引脚需要接上拉电阻. 在I/O引脚作为I²C 端口使用时, 用户必须预先设置引脚功能为I²C功能。

5.8.2 特征

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

- 支持主机模式或从机模式
- 主从机之间双向数据传输
- 多主机总线支持 (无中心主机)
- 同时传输的多主机间进行仲裁，反之串行数据被破坏
- 总线上不同传输速率设备间，依靠串行时钟实现同步传输
- 串行时钟可以用作一个握手信号挂起和重新开始串行传输
- 内建14位超时计数器，当I²C总线挂起并且计数器溢位时， I²C中断将发生.
- 需要外部上拉确保高电平输出
- 可编辑的时钟适用于不同速率控制
- 支持7位地址模式

- I²C总线控制器支持多地址辨识 (4组从机地址带mask选项)
- 支持掉电唤醒功能

5.8.3 功能描述

I²C 协议

5.8.3.1 通常，标准I²C传输协议包含四个部分

1) 起始信号或重复起始信号
2) 从机地址传输和R/W位传输
3) 数据传输
4) 停止信号

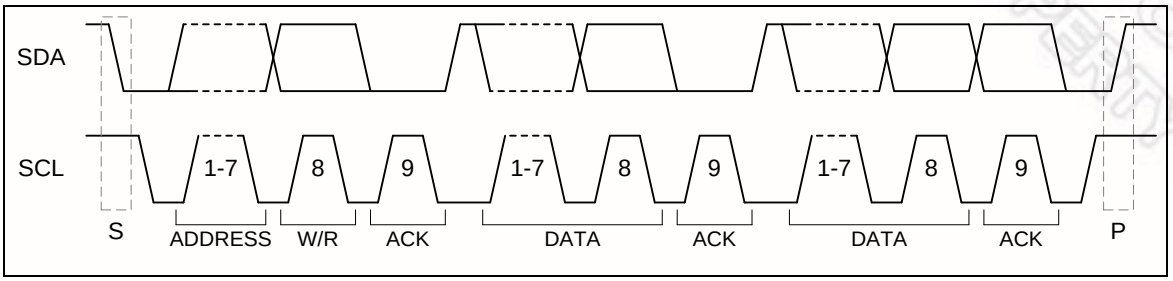


图 5-27 I²C 协议

5.8.3.2

I²C总线上数据传输

图5-27演示了主机发送数据到从机。主机首先通过一个7个bit的地址和1个bit的方向表明主机想发送数据到从机。从机返回应答信号之后，主机保持发送数据

传输方向不改变

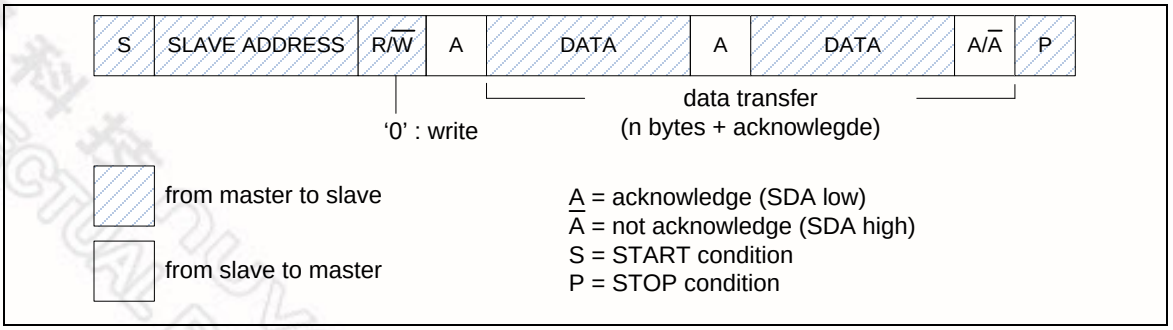


图 5-28 主机向从机传输数据

图5-28演示了主机从从机读数据。主机首先通过一个7个bit的地址和1个bit的方向表明主机想从从机收数据。从机返回应答信号之后，将开始发送数据到主机

第一个字节后（从机地址）主机紧接着由从机读取数据

传输方向改变

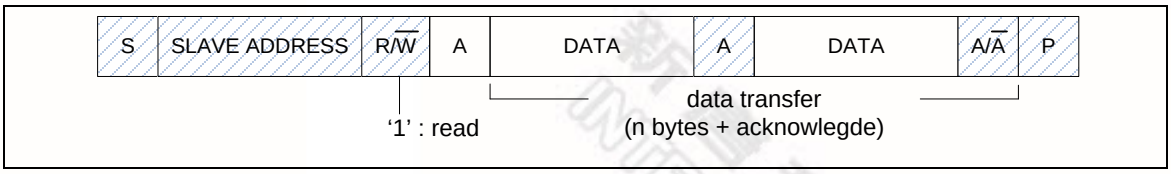


图 5-29 主机由从机读取地址

起始位或重复起始信号

5.8.3.3 当总线处于空闲状态时，说明没有主机对总线发起传输请求(SCL和SDA线同时为高)，主机可以通过发送一个START信号来发起传输请求。起始信号，通常表示为**S-bit**, 定义为当SCL线为高时，SDA线上信号**由高至低。**起始信号表示总线上新的传输开始.

重复起始信号 (Sr) 即在两个START信号之间没有STOP信号. 主机采用这种方法与另一个从机或相同的从机以不同传输方向进行通信 (例如：从写入设备到从设备读出) 而不释放总线.

STOP 信号

主机向总线发出停止信号结束数据传送。停止信号，通常用**P-bit**表示, 定义为当SCL线为高时，SDA线上信号由低到高，为停止信号.

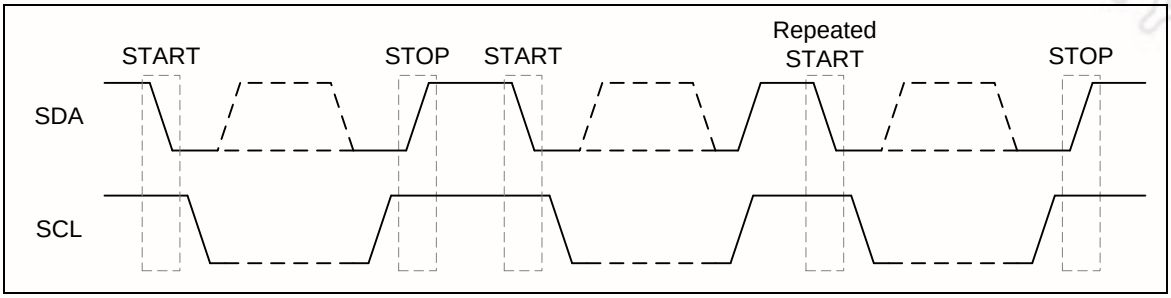


图 5-30 START 和 STOP 条件

5.8.3.4

从机地址传输

START信号后马上传输的第一个字节就是从机地址。这是一个7位从设备地址加一个R/W位，R/W位通知从机数据传输的方向。系统中没有两个从机有相同的地址，只有被主机寻址的从机会通过在第9个SCL时钟周期将SDL置低电平作为应答。

5.8.3.5

数据传输

当从机地址被成功识别时，就可以根据RW所指定的方向，开始一字节一字节的数据传输，每个传输字节在第9个时钟周期跟一个应答信号，如果从机上产生无响应信号**(NACK)**, 主机可以产生停止信号来终止数据传输，或者产生重复起始信号开始新一轮的数据传输。

当主机作为接收器件，发生无响应信号**(NACK)** 时,从机释放SDA线，使主机产生停止信号或重复起始信号.

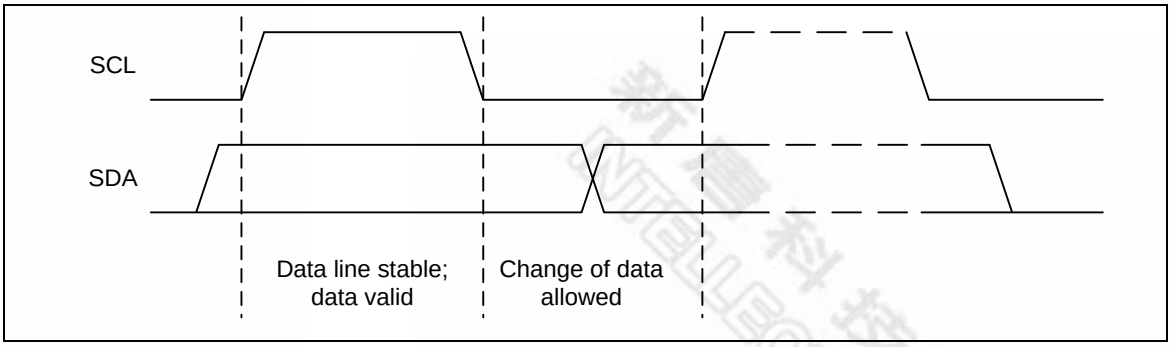


图 5-31 I²C 总线上位传输

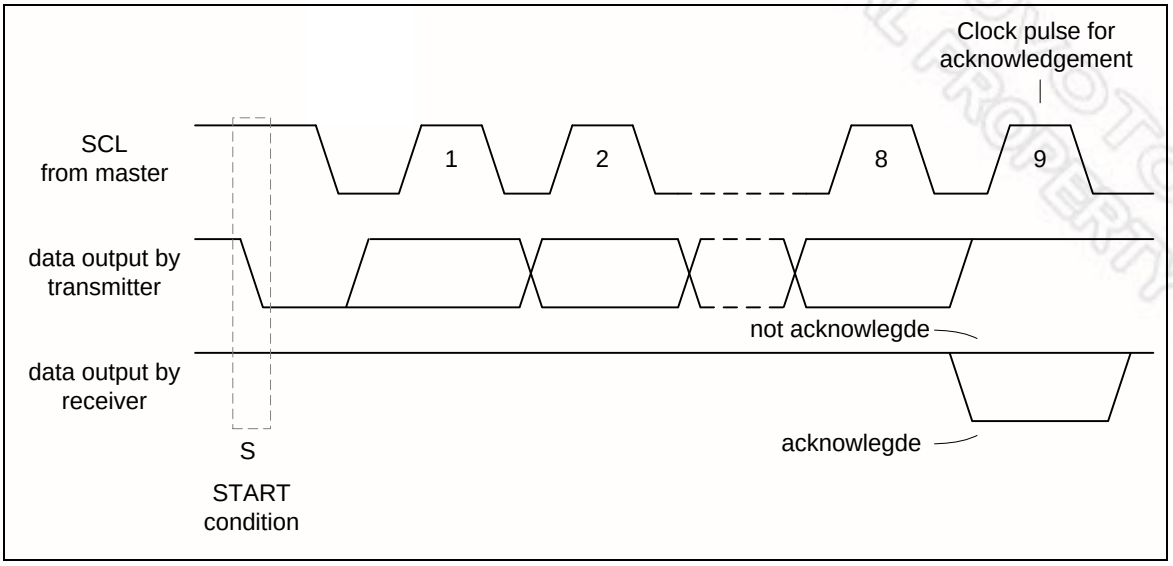


图 5-32 I²C 总线上应答信号

5.8.4 协议寄存器

CPU和I²C通过下列13个特殊功能寄存器通讯: **I2CON** (控制寄存器), **I2CSTATUS** (状态寄存器), **I2CDAT** (数据寄存器), **I2CADDRn** (地址寄存器,n=0~3),**I2CADRMn**（地址掩码寄存器，n=0~3） **I2CLK** (时钟速率寄存器) 和 **I2CTOC** (超时寄存器) 。所有I²C的寄存器的第8位至第31位都是保留的，不具备任何功能，返回值为0。

当ENS1(I2CON[6])置1，I²C口使能后，内部状态由 I2CON 和I²C逻辑硬件控制。当有新的状态码产生后，会存储到I2CSTATUS寄存器，I²C中断标志(SI(I2CON[3]))也会自动置起。若此时EI(I2CON[7]) 位为高，I²C中断会发生。I2CSTATUS[7:3]存储内部状态码，低3个比特总是0，在SI由软件清除之前，I2CSTATUS寄存器的内容保持稳定。寄存器基地址为0x4002_0000 和0x4012_0000.

地址寄存器(I2CADDR)

5.8.4.1 I²C 口内建4个从机地址寄存器I2CADDRn(n=0~3)。当I²C作为主机时，这四个寄存器的内容不相关。 在从机模式下，位字段I2CADDRn[7:1] 必须装载芯片自己的从机地址。当I2CADDR 地址与接收的从机地址符合时，I²C硬件将应答.

I²C口支持“广播呼叫”功能。当GC位(I2CADDRn [0])被置起，I²C端口硬件将应答广播呼叫的地址，清GC位可禁止“广播呼叫”功能.

当GC位被置且I²C处于从机模式时，主机发出广播呼叫地址到I²C总线后，可以接收广播呼叫地址00H，然后，它将遵守GC模式的状态。

I²C总线控制器有4个地址掩码寄存器I2CADRMn(n=0~3)并支持多地址识别。当地址掩码寄存器中某个比特置1时，表示接收到的相应地址位将被忽略。当该位为0时，说明接收到的相应地址位应该与地址寄存器内的值完全相符.

5.8.4.2

数据寄存器 (I2CDAT)

该寄存器包含一个准备发送或刚接收到的一个字节的数据。只要不在移位处理的过程，CPU可以直接读写I2CDAT[7:0]。当I²C处于一个确定的状态并且串行中断标志(SI)被设置， I2CDAT[7:0]中的数据就一直是稳定的。在数据被移出的过程中，总线上的数据同时被移入。I2CDAT的内容一直是总线上出现的最后一个字节。因此，在仲裁失败时，主发送从接收的模式下，I2CDAT[7:0]中的数据仍保持正确。

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

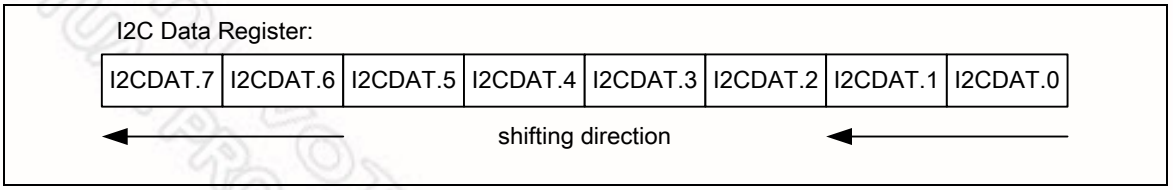


图 5-33 I²C 数据移位方向

控制寄存器(I2CON)

CPU 可以直接读或写寄存器I2CON [7:0]. 两个受硬件影响的位是: 在I²C硬件请求串行中断时SI置位； 在STOP状态出现在总线上时清STO位. 当ENS1 = "0"时，STO位也会被清除.

5.8.4.3

EI 使能中断.

ENS1 设置使能I²C串行功能控制器. 当ENS1=1 ，使能I²C 串行功能. SDA与SCL的多功能引脚必须设置成 I²C功能.

STA I²C START 控制位. STA为高时I²C进入主机模式。如果总线为空闲，硬件将发送一个START或repeat START 状态到总线。

STO I²C STOP 控制位。I²C为主机模式时，STO位置'1'， 设置STO来传送一个STOP状态到总线，I²C硬件将检测总线状态，如果一个STOP状态被检测到，这个标志将被硬件自动清除。在从机模式下，设置STO将复位I²C硬件为“未寻址”从机模式，这表示从机接收模式不再从主机传送装置接收数据。

SI I²C中断标志，当一个新的I²C状态出现在寄存器I2CSTATUS时，SI标志由硬件置位，并且如果EI (I2CON [7])位被设置，将产生I²C中断请求。SI标志由硬件置'1' ，SI必须由软件通过对该写‘1’来清‘0’。所有的状态均在5.6.6节列出

AA 应答控制位. 当AA=1 先于收到地址或数据被设时, 在以下两种情况：1.) 从机正在应答主机发送的地址，2.) 接收设备正在应答发送设备发送的数据，在SCL线上的应答时钟脉冲期间将返回一个应答 (SDA上的低电平)。当AA=0 先于收到地址或数据被设时，在SCL线上的应答时钟脉冲期间将返回一个无应答(NACK)信号

5.8.4.4

状态寄存器(I2CSTATUS)

I2CSTATUS［7:0］是一个8-位只读寄存器。低3位一直为0。I2CSTATUS [7:3]是状态码。有26个可能的状态码，所有的状态在5.6.6节列出。当I2CSTATUS[7:0]的内容是F8H，没有串行中断请求。所有的其它I2CSTATUS[7:3]值对应I²C 端口的状态。 当每进入一个状态时，就会产生状态中断请求(SI = 1). 在SI被硬件置'1'一个机器周期之后，有效状态码出现在I2CSTATUS[7:3]中；在SI被软件清除之后1个机器周期之内，I2CSTATUS[7:3]中的状态码仍维持有效.

另外，00H状态表示总线错误。当START或STOP条件出现在帧结构不正确的位置时，总线错误发生。例如：在传输地址字节时，数据字节时，或者应答位期间。要从总线错误中恢复I²C，应该设置STO，清除SI以进入未寻址从模式。然后清除STO，释放总线，等待新的通信。当总线产生错误时，I²C总线不能识别停止状态。

Master Mode		**Slave Mode**	
STATUS	**Description**	**STATUS**	**Description**
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
0x38	Master Arbitration Lost	0x60	Slave Receive Address ACK
0x40	Master Receive ACK	0x68	Slave Receive Arbitration Lost
0x48	Master Receive NACK	0x80	Slave Receive Data ACK
0x50	Master Receive ACK	0x88	Slave Receive Data NACK
0x58	Master Receive NACK	0x70	GC Mode Address ACK
0x00	Bus Error	0x78	GC Mode Arbitration Lost
		0x90	GC Mode Data ACK
		0x98	GC Mode Data NACK
0xF8	Bus Released **Note:** The status “0xF8” exists in both Master/Slave modes, and it will not raise any interrupt.		

表 5-9 I²C 状态码表

5.8.4.5 *I²C时钟波特率位 (I2CLK)*

当I²C在主机模式下，I²C数据的波特率由I2CLK[7:0]寄存器设定。I²C在从机模式下时是不重要的；在从机模式下，I²C将自动与主机I²C设备时钟频率同步。

数据波特率设定：I²C的数据波特率= (system clock) / (4x (I2CLK [7:0] +1))，如果system clock =16 MHz, I2CLK[7:0]= 40(28H)，I²C的数据波特率 = 16 MHz /(4X (40 +1)) = 97.5K位/秒.

5.8.4.6 *I²C超时计数寄存器(I2CTOC)*

当总线被锁死时，MCU提供一个14位超时计数器。当计数功能使能后，计数器开始计数直至溢出(TIF=1)，产生I²C中断或者通过清除ENT1位来关闭计数功能。当超时计数器使能时，对SI标志置高会使计数器复位，SI被清为0之后会重新开始计数。如果I²C总线锁死，会使I2CSTATUS及SI标志不再更新,该14位超时计数器会发生溢出从而产生I²C中断通知CPU，. 参考 图 5-34 14位超时计数器. 用户写1清TIF为0.

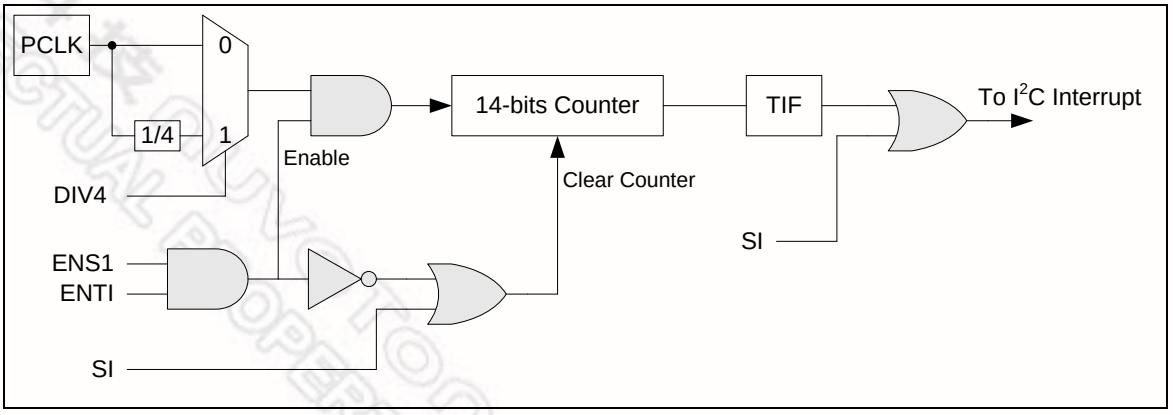


图 5-34 I²C 超时计数器框图

I²C 唤醒控制寄存器 (I2CWKUPCON)

进入睡眠模式时，其它的I²C主机可以通过寻址来唤醒我们的芯片。进入睡眠模式之前，用户必须配置相关设定.

5.8.4.7

WKUPEN 使能I²C 唤醒功能

I²C 唤醒状态寄存器 (I2CWKUPSTS)

当系统被其它的I²C主机唤醒时，WKUPIF 被置来标识这个事件

5.8.4.8

WKUPIF 唤醒中断标志

5.8.5 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
I2C 基地址: **I2Cx_BA = 0x4002_0000 + (0x10_0000 * x)** x = 0, 1				
I2CON	I2Cx_BA+0x00	R/W	I^2C 控制寄存器	0x0000_0000
I2CADDR0	I2Cx_BA+0x04	R/W	I^2C 从机地址寄存器0	0x0000_0000
I2CDAT	I2Cx_BA+0x08	R/W	I^2C 数据寄存器	0x0000_0000
I2CSTATUS	I2Cx_BA+0x0C	R	I^2C 状态寄存器	0x0000_00F8
I2CLK	I2Cx_BA+0x10	R/W	I^2C 时钟时钟分频寄存器	0x0000_0000
I2CTOC	I2Cx_BA+0x14	R/W	I^2C 超时控制寄存器	0x0000_0000
I2CADDR1	I2Cx_BA+0x18	R/W	从机地址寄存器1	0x0000_0000
I2CADDR2	I2Cx_BA+0x1C	R/W	从机地址寄存器2	0x0000_0000
I2CADDR3	I2Cx_BA+0x20	R/W	从机地址寄存器3	0x0000_0000
I2CADM0	I2Cx_BA+0x24	R/W	从机隐藏地址寄存器0	0x0000_0000
I2CADM1	I2Cx_BA+0x28	R/W	从机隐藏地址寄存器1	0x0000_0000
I2CADM2	I2Cx_BA+0x2C	R/W	从机隐藏地址寄存器2	0x0000_0000
I2CADM3	I2Cx_BA+0x30	R/W	从机隐藏地址寄存器3	0x0000_0000
I2CWKUPCON	I2Cx_BA+0x3C	R/W	I^2C 唤醒控制寄存器	0x0000_0000
I2CWKUPSTS	I2Cx_BA+0x40	R	I^2C 唤醒状态寄存器	0x0000_0000

5.8.6 寄存器描述

I²C控制寄存器(I2CON)

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

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

Bits	描述	
[31:8]	保留	保留
[7]	EI	**使能中断** 1 = 使能I²C中断功能 0 = 禁止I²C中断功能
[6]	ENSI	**I²C控制器使能位** 1 = 使能 0 = 禁止 当ENS1置1，I²C串行功能使能，SDA和SCL对应的GPIO管脚必须被设置成SDA，SCL功能.
[5]	STA	**I²C起始控制位** STA置1 ，进入主机模式，如果总线处于空闲状态，I²C硬件会送出起始信号或重复起始信号.
[4]	STO	**I²C 停止标志** 在主机模式下设置STO将在I²C总线上产生STOP时序，硬件会监测总线上的状态，当检测到STOP状态时，自动将此位清零。在从机模式下设置STO将复位I²C硬件为“未寻址”模式，也就是说它已经退出了从主机接收数据的从机接收模式
[3]	SI	**I²C 中断标志位.** 在I2CSTATUS寄存器上出现一个新的I²C信号时，SI将由硬件置位，如果EI (I2CON [7])位被置'1'， I²C中断请求，可以产生中断。SI必须由软件清除，之后I²C控制器才会进行下一步动作。
[2]	AA	**接收应答标志位** 在接收到地址或者数据之前将AA设为1，在应答时钟脉冲时，将返回应答信号(SDA将被拉为低电平)：1) I²C为从机模式，已经接收到主机发来的地址。2) I²C为接收模式，已经

		接收一个数据。。 在接收到地址或者数据时如果AA=0，在应答时钟脉冲时，将没有应答信号返回(SDA上高电平)。
[1:0]	**保留**	保留

I²C 数据寄存器 (I2CDAT)

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

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

Bits	描述	
[31:8]	**保留**	保留
[7:0]	**I2CDAT**	**I²C数据寄存器** Bit [7:0] 为8位I²C数据.

I²C 状态寄存器 (I2CSTATUS)

寄存器	偏移量	R/W	描述	复位后的值
I2CSTATUS	I2C_BA+0x0C	R/W	I²C状态寄存器	0x0000_00F8

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

Bits	描述	
[31:8]	**保留**	保留
[7:0]	**I2CSTATUS**	**I²C状态寄存器** 低三位始终是0；高5位包含状态码，状态码有26可能；当I2CSTATUS的值是F8H，表示没有串行中断请求；其它的所有的I2CSTATUS值对应I²C的状态。当进入这些状态时会产生一个状态中断请求(SI=1)。一个有效的状态码在SI被硬件设为'1'后一个周期内存放到I2CSTATUS中；在SI被软件清'0'后仍存在一个周期。另外，状态码是00H时表示总线错误；当'起始'或'结束'信号出现在帧结构的错误位置时会产生总线错误，例如：发送串行地址字节时，数据字节时和应答位时.

I²C 时钟分频寄存器(I2CLK)

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

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

Bits	描述	
[31:8]	保留	保留
[7:0]	I2CLK	I²C 时钟分频寄存器 I²C波特率 = （system clock ）/(4x(I2CLK+1))

I²C超时计数寄存器(I2CTOC)

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

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

Bits	描述	
[31:3]	保留	保留
[2]	ENTI	**超时计数使能/禁用** 1 = 使能 0 = 禁用 SI被清0时，14位超时计数寄存器开始计数；对SI置1会使计数器复位，并在SI清0后重新开始计数。
[1]	DIV4	**超时计数输入时钟除4** 1 = 使能 0 = 禁用 使能后，超时时间扩大4倍
[0]	TIF	**超时标志** 当超时发生时，该位由 H/W 置位，如果此时 I²C 的中断使能位 (EI) 置为1，则可引发 CPU 的中断。 S/W 可写 1 清除该位。 1 = 超时标志由硬件置位，可以导致CPU发生中断. 0 = 没有超时发生，软件能写"1"清除这个标志

I²C 从机地址寄存器 (I2CADDRx)

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

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

Bits	描述	
[31:8]	保留	保留
[7:1]	I2CADDR	I²C 地址寄存器 在主机模式下寄存器的值没有意义. 在从机模式下，高7位必须为MCU本身的地址。如果地址符合硬件会自动应答.
[0]	GC	广播呼叫功能. 0 = 禁用广播呼叫功能. 1 = 允许广播呼叫功能.

I²C从机地址掩码寄存器 (I2CADMx)

寄存器	偏移量	R/W	描述	复位后的值
I2CADM0	I2C_BA+0x24	R/W	I²C 从机地址掩码寄存器0	0x0000_0000
I2CADM1	I2C_BA+0x28	R/W	I²C 从机地址掩码寄存器1	0x0000_0000
I2CADM2	I2C_BA+0x2C	R/W	I²C 从机地址掩码寄存器2	0x0000_0000
I2CADM3	I2C_BA+0x30	R/W	I²C 从机地址掩码寄存器3	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
保留							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
保留							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I2CADM[7:1]							保留

Bits	描述	
[31:8]	**保留**	保留
[7:1]	**I2CADM**	**I²C地址掩码寄存器:** 1 = 使能掩码 (接收到的相应地址位被忽略) 0 = 禁用掩码 (接收到的地址位必须与地址内容完全符合) I²C总线支持多地址辨识。当打开掩码时，接收到的从机地址相应位是否正确不予处理，当选择为禁用隐码时，从机地址相应位必须完全符合收到的地址才给与响应。
[0]	**保留**	保留

I²C 唤醒控制寄存器 (I2WKUPCON)

寄存器	偏移量	R/W	描述	复位后的值
I2CWKUPCON	I2C_BA+0x3C	R/W	I²C 唤醒控制寄存器	0x0000_0000

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

Bits	描述	
[31:1]	**保留**	保留
[0]	**WKUPEN**	**I²C 唤醒功能使能** 1 = 使能I²C 唤醒功能. 0 = 关闭I²C 唤醒功能.

I²C 唤醒状态寄存器 (I2WKUPSTS)

寄存器	偏移量	R/W	描述	复位后的值
I2CWKUPSTS	I2C_BA+0x40	R	I²C 唤醒状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	**保留**	保留
[0]	**WKUPIF**	**唤醒中断标志** 1 = I²C发生了唤醒中断. 0 = I²C没有发生唤醒中断. 该位写”1”清除

5.8.7 操作模式

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

在实际应用中，I²C端口可以作为主机和从机。在从机模式，I²C端口寻找自身从机地址和广播呼叫地址，如果有地址被检测到，从机将接收或发送数据（通过设置AA位），应答信号在第9个时钟脉冲时发送，此时，如果中断已经使能，则发生中断请求。当芯片希望成为总线主机时，在进入主机模式之前，硬件等待总线空闲以使可能发生的从机动作不被中断。在主机模式，如果总线仲裁失败，I²C立即切换到从机模式，同时开始检测自身从机地址.

在SI位清除后，I2CON中的STA，STO和AA位将决定I²C硬件的下一个状态。一个新的动作完成后，将更新一个新的状态码并置位SI标志。如果I²C中断控制位EI（I2CON［7］）置位，可在中断服务子程序中根据新的状态码执行相应的动作。

每种模式下的数据传输见 **Error! Reference source not found.** 到 **Error! Reference source not found.**.

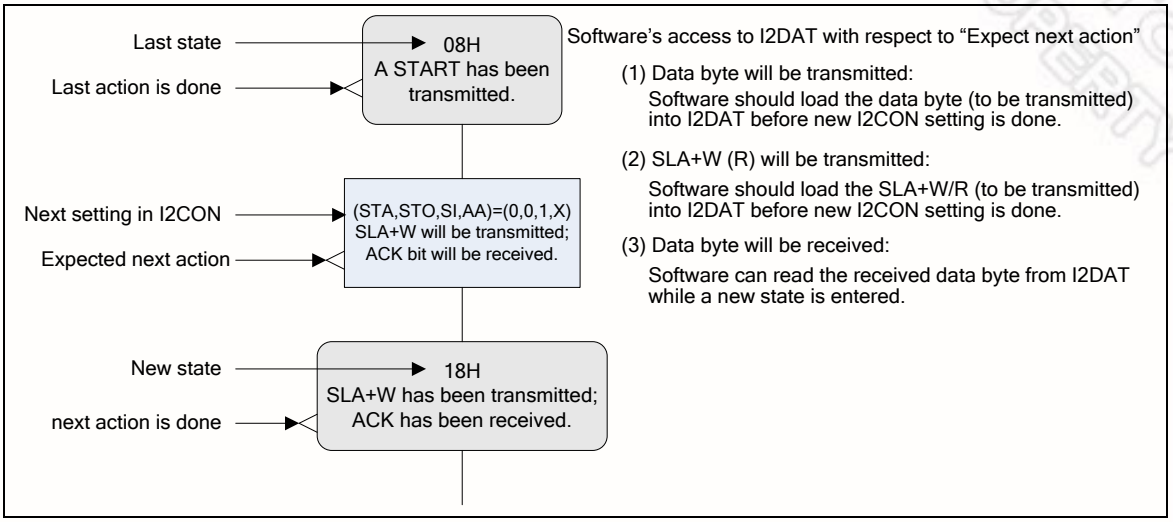
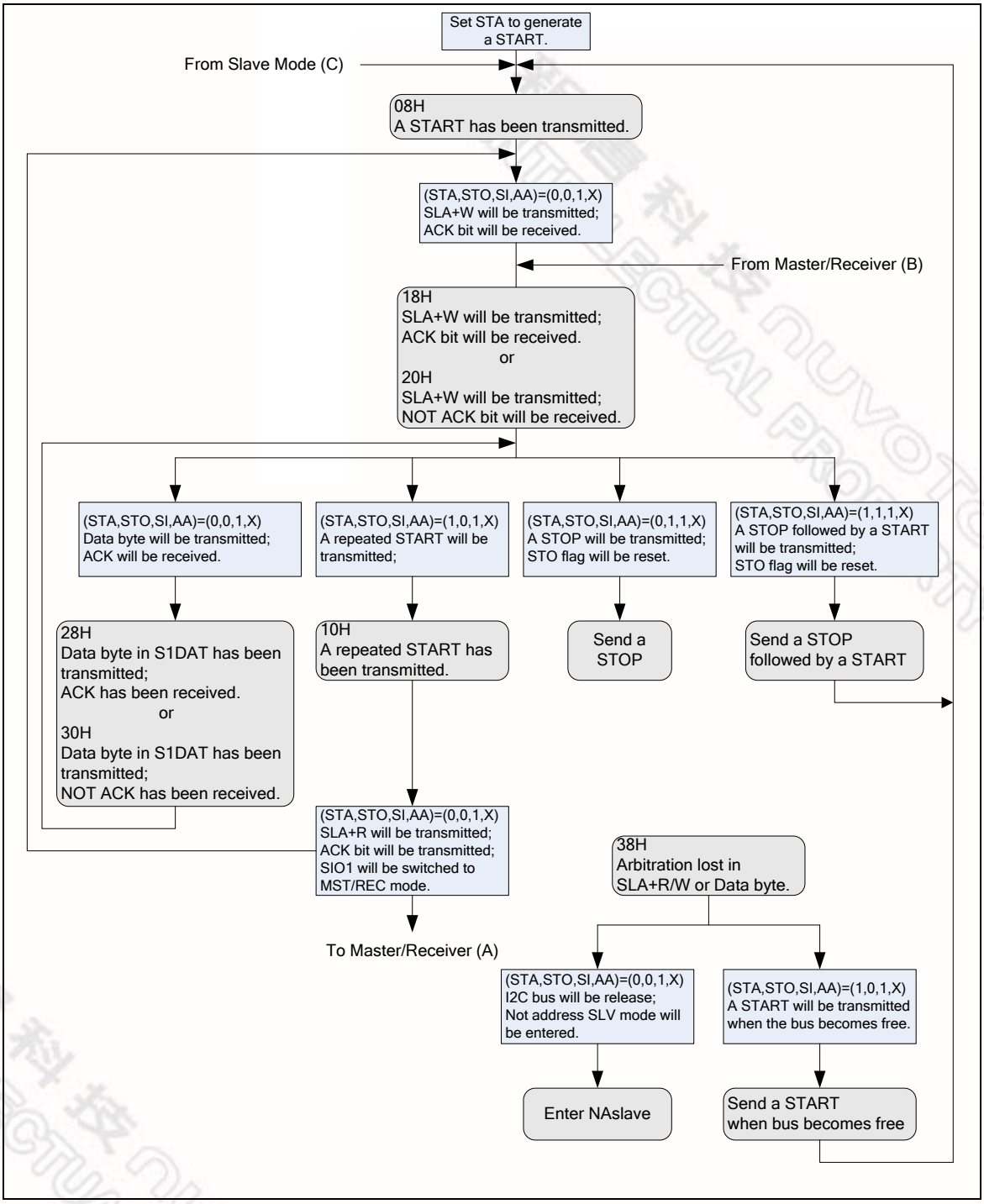


图 5-35 接下来的五个图的图例说明

5.8.7.1

主机发送模式

当SCL线上输出时钟信号时，SDA线上输出数据。置位STA，主机产生起始信号，之后主机向总线传输的第一个字节为从机地址（一般为7位）+方向位。主机传输模式下方向位(R/W)为0，用"W"表示，因而第一个传输的字节为SLA+W。串行数据一次传输8个比特。每个字节传输完成后，将收到应答（ACK）信号。输出起始或停止信号，标志开始传输或者停止传输



5.8.7.2

图 5-36 主机发送模式

主机接收模式

这种情况下方向位(R/W) 为1 ，用”R”表示。因而第一个传输的字节为SLA+R. 当SCL线上输出时钟信号时，SDA线上输出数据。串行数据一次接收8位，每接收到一个字节，响应一个应答信号，输出起始或者停止信号，标志开始传输或者停止传输.

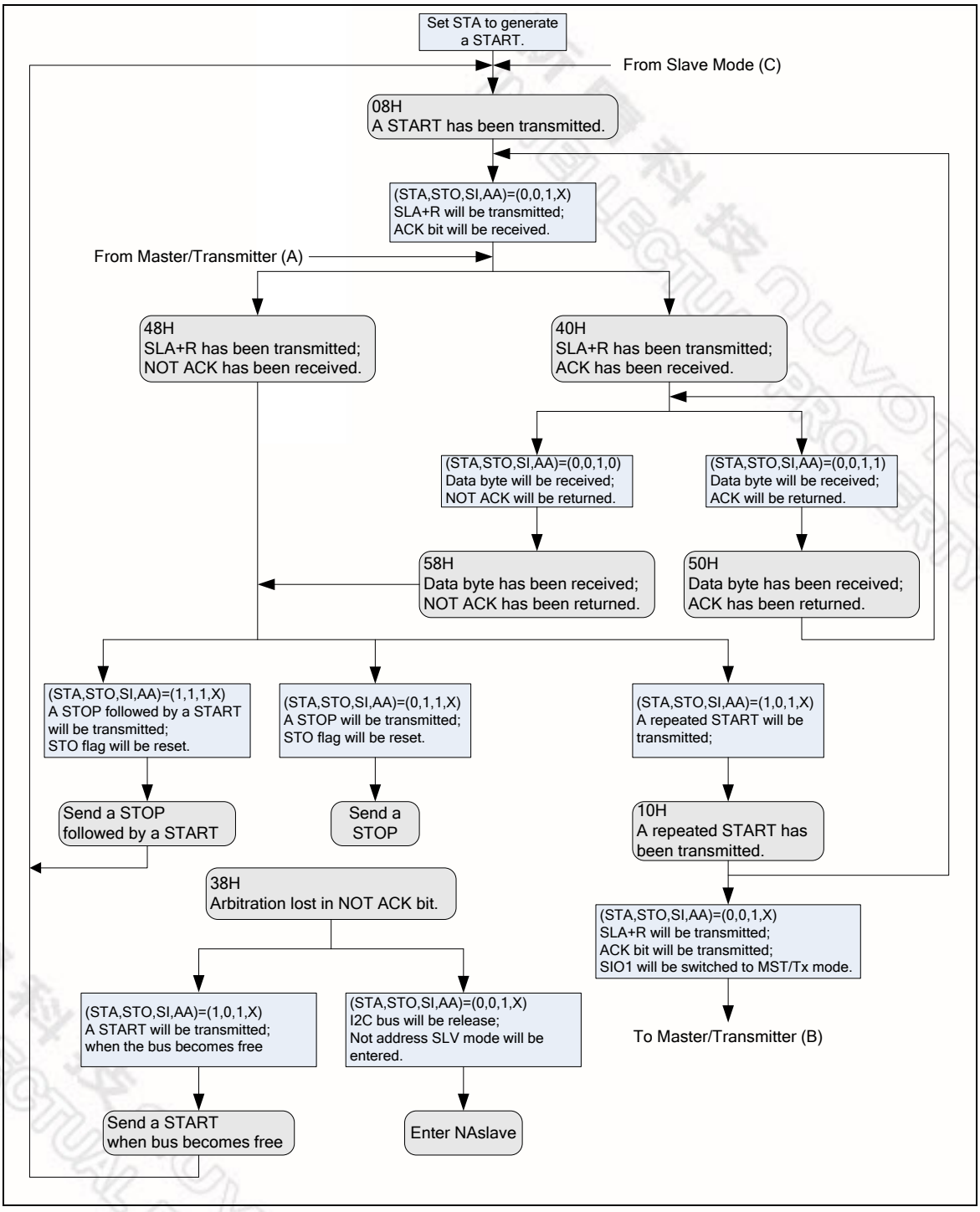

Set STA to generate a START.
From Slave Mode (C)
08H
A START has been transmitted.
(STA,STO,SI,AA)=(0,0,1,X)
SLA+R will be transmitted;
ACK bit will be received.
From Master/Transmitter (A)
48H
SLA+R has been transmitted;
NOT ACK has been received.
40H
SLA+R has been transmitted;
ACK has been received.
(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/Tx mode.
(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.
To Master/Transmitter (B)
Send a START
when bus becomes free
Enter NAslave

图 5-37 主机接收模式

从机接收模式

5.8.7.3 从串行线上接收SDA和SCL信号。每接收完一个字节，回复一个响应信号。识别起始或结束标志，作为传输的开始和结束。从机地址识别由硬件完成.

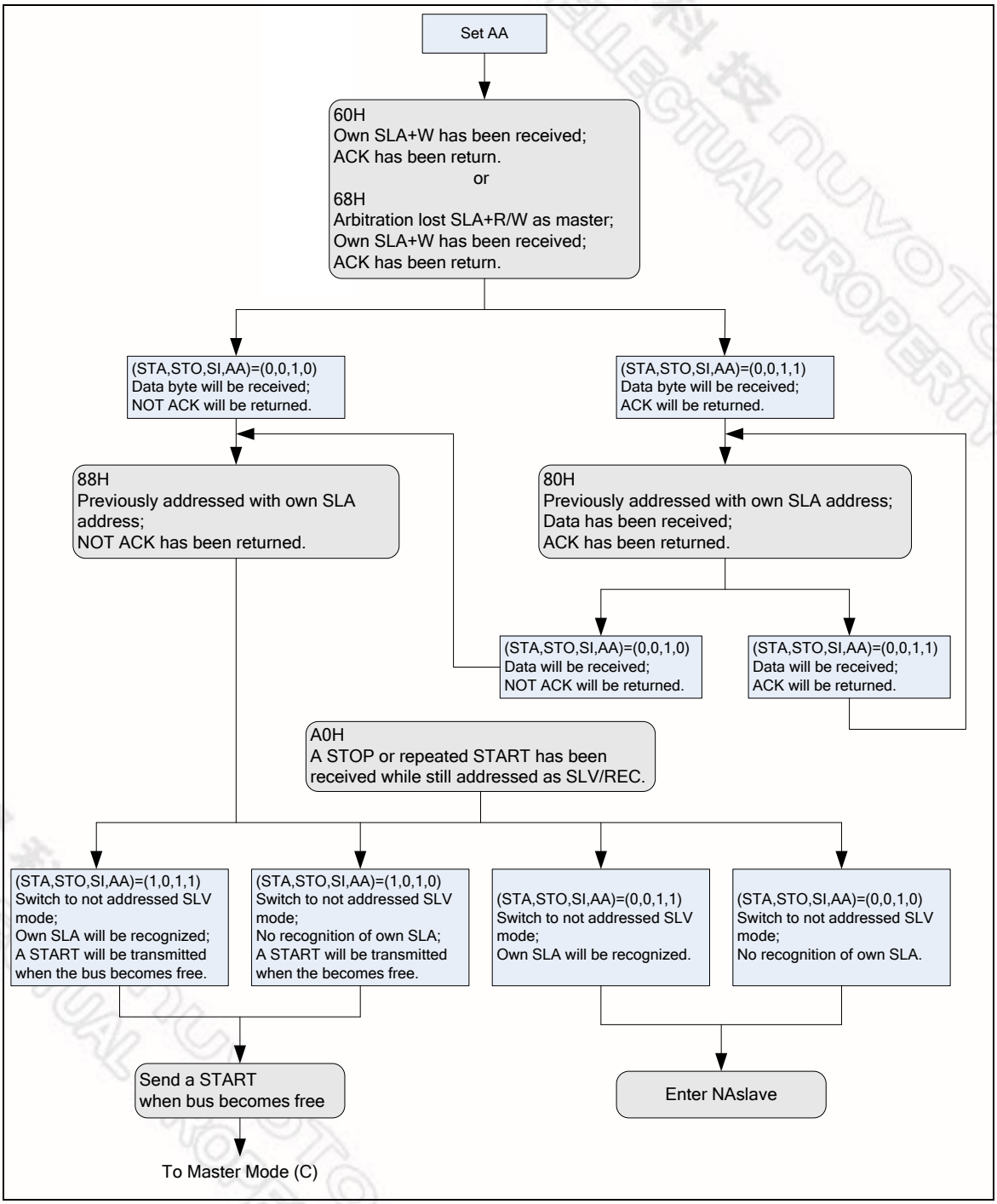


图 5-38 从机接收模式

从机发送模式

5.8.7.4 第一个字节为地址字节加方向位，此时从机仍为接收模式，然后传输方向反转，当SCL线上接收到时钟信号时，SDA脚上输出数据，并辨识是否有起始或停止标志.

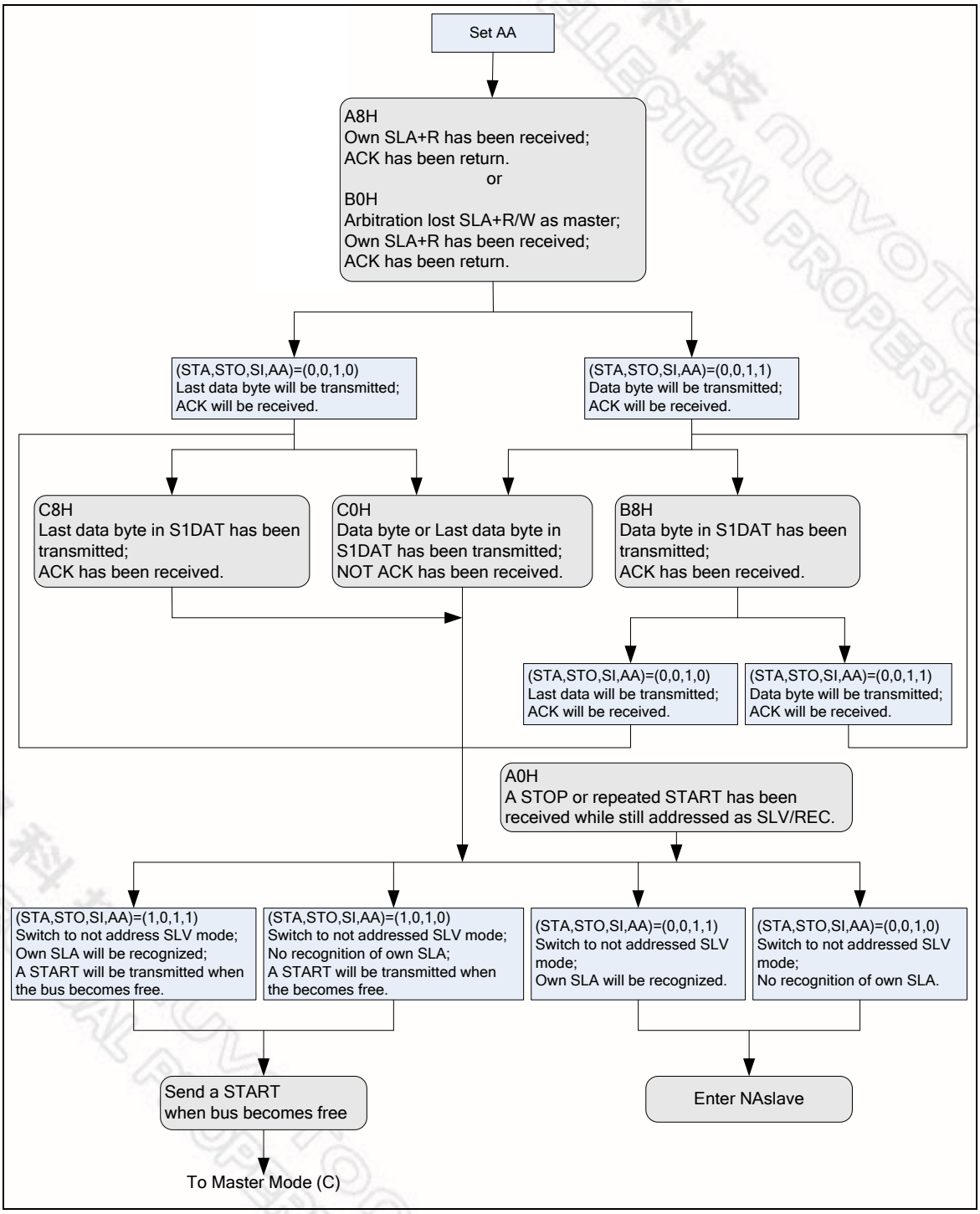


图 5-39 从机发送模式

全呼叫 (GC) 模式

5.8.7.5

如**Error! Reference source not found.**所示, 如果GC位（I2CADDRn[0]）被置， 硬件I^2C 端口将响应GC呼叫地址 (00H)。 清除GC位将关闭全呼叫功能。当GC位被置并且I^2C 工作在从模式，它将接收来自主机的全呼叫地址00H，然后它将遵守GC模式的状态变化。串行数据和串行时钟通过SDA和SCL接收。每收到一个字节都会返回应答位。START和STOP为传输的开始和结束。收到从机地址和方向位之后，地址识别由硬件执行。.

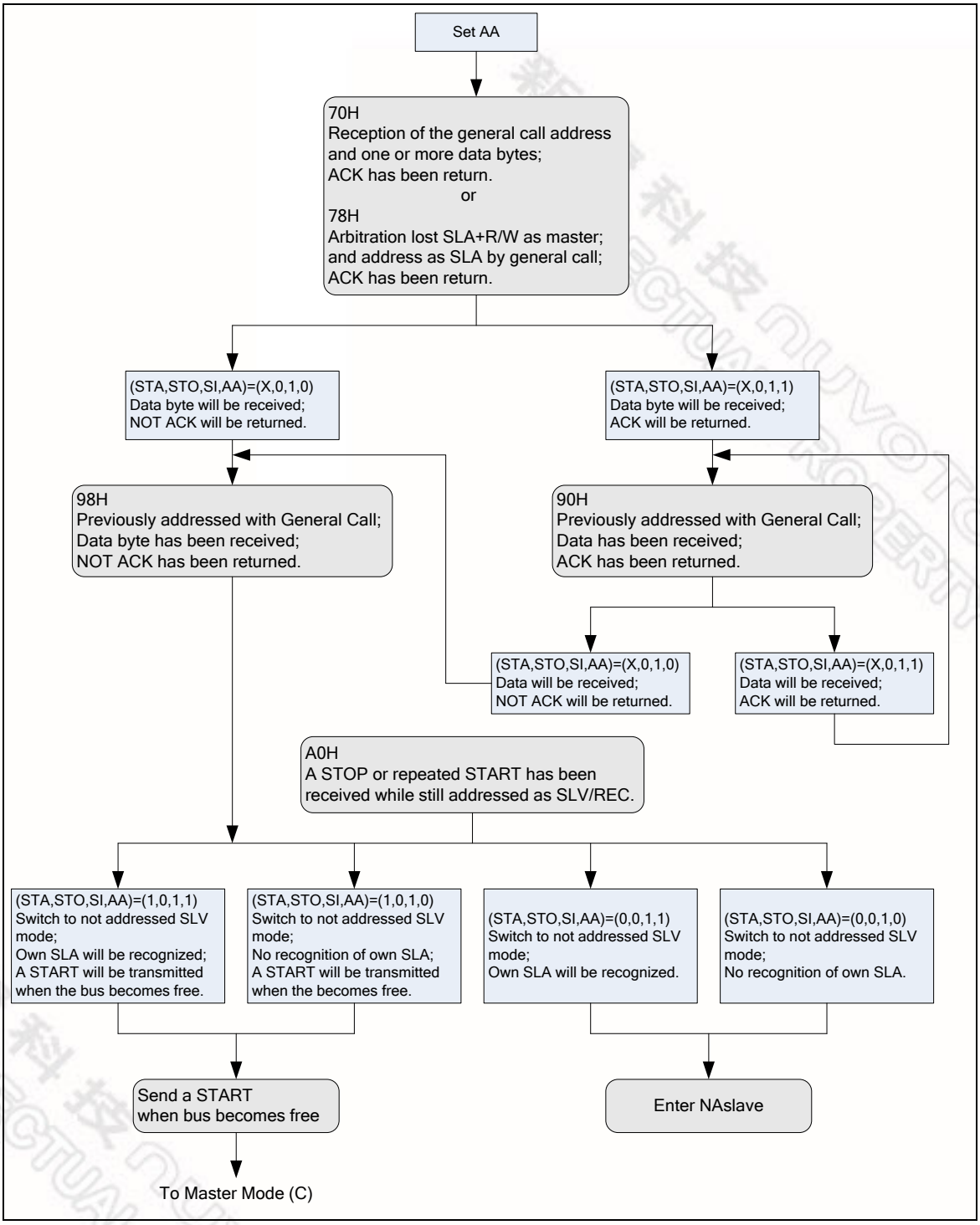


图 5-40 全呼模式

5.8.7.1 *随机读EEPROM范例*

配置I²C相关寄存器.

1. 配置寄存器“GPA_MFP” & “ALT_MFP” 将相应引脚配置为I²C功能
2. 使能I²C APB时钟，如果使用I2C0，“APBCLK”寄存器中的I2C0_EN=1 ，否则 I2C1_EN=1
3. 复位I²C ，如果使用I2C0，则“IPRSTC2” 寄存器中 I2C0_RST=1，然后I2C0_RST=0 ；如果使用I2C1则I2C1_RST=1，然后I2C1_RST=0
4. I2CON寄存器中ENS1 = 1，使能I²C 控制器
5. 写”I2CLK”寄存器，决定I²C总线SCL的频率
6. 写0x00040000到 “NVIC_ISER” 寄存器，使能I²C 中断
7. “I2CON”寄存器中 EI=1，使能I²C 中断
8. 写“I2CADDR0”寄存器，设置I²C从机地址

随机读EEPROM，NUC123作为主机，EEPROM作为从机。该操作分为两步：发送和接收。图 5-41 显示了 EEPROM 随机读操作的流程.

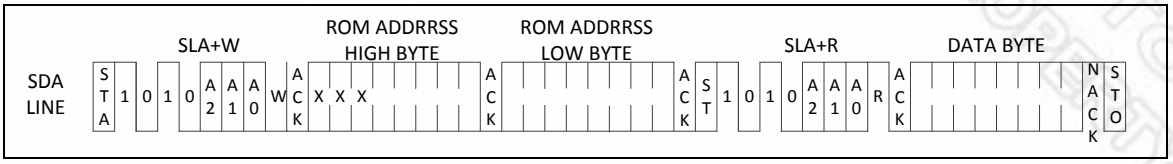


图 5-41 EEPROM 随机读操作

■ 发送操作

I²C 控制器首先发送START信号，然后发送 SLA (从机地址)+W 到 EERPOM ，并发送要访问的地址 (16 bits) 到EEPROM。详细步骤如下:

1. 发送一个START信号，并等待I2CSTATUS变成0x08

   ---发送START信号 (STA, STO, SI, AA) = (1, 0, 1, X)

2. I2CSTAUS = 0x08，表示START信号已经发送成功。然后写7-bit的EEPROM从机地址和读/写控制位(写)到I2CDAT, 寄存器，并清除SI位，然后等待EEPROM返回应答信号.

   --- I2C0DAT=SLA+W, 然后清除 SI (STA, STO, SI, AA) = (0, 0, 1, X);

3. I2CSTAUS = 0x18，表示收到EEPROM返回的ACK信号。然后写要读的EEPROM地址高字节到I2CDAT寄存器，并清除SI位，然后等待应答信号。如果I2CSTATUS=0x20，表示收到NACK，表示EEPROM地址不正确或者EEPROM不存在。此时应该发送STOP信号停止传输过程.

   --- 收到ACK (0x18): I2CDAT=要读的地址高字节并清除SI (STA, STO, SI, AA) = (0,0,1,X);

   --- 收到NACK (0x20): 发送STOP停止传输(STA, STO, SI, AA) = (0, 1, 1, X)

4. I2CSTATUS=0x28, 表示高地址发送成功，然后发送EEPROM低字节并清除SI位，等待ACKan ACK is received. Send data pointer low byte to EEPROM and wait next ACK back. 。如果 I2CSTATUS=0x30，表示NACK，高地址字节没有应答 I²C 应该停止传输.

   --- 收到ACK (0x28): I2CDATA=要读的地址低字节，并清除SI位 (STA, STO, SI, AA) = (0, 0, 1, X);

   --- 收到NACK (0x30): 发送STOP停止传输(STA, STO, SI, AA) = (0, 1, 1, X)

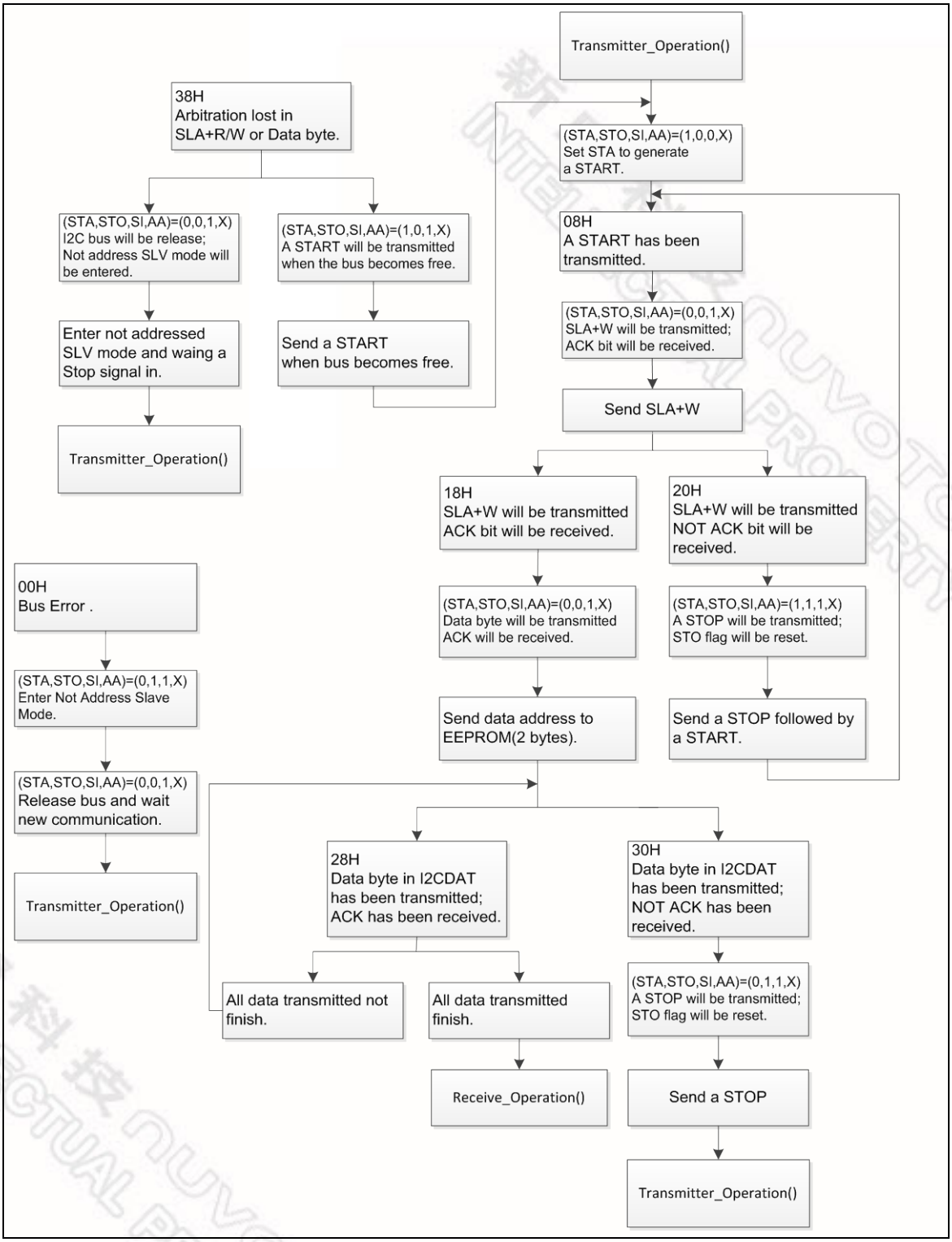


图 5-42 随机读发送操作

■ 接收操作

要读的EEPROM地址字节发送成功以后，接下来就需要从EEPROM接收数据了。. I²C 控制器发送SLA+R 到 EERPOM ，收到ACK之后，EEPROM就会返回第一个字节的数据。只要I²C 控制器返回ACK，接收操作就是连续的，直到I²C 控制器返回NACK，表示读操作结束。详细步骤如下：.

1. 当I2CSTATUS=0x28时，表示低字节发送成功，接下来发送 RESTART 信号..。 如果I2CSTATUS=0x30, 表示NACK，I²C 控制器应该发送STOP信号停止传输.

   --- 收到ACK (0x28): 清除SI位并发送RESTART信号(STA, STO, SI, AA) = (1, 0, 1, X)

   --- 收到NACK (0x30): 清除SI位并发送STOP信号(STA, STO, SI, AA) = (0, 1, 1, X)

2. I2CSTATUS=0x10, RESTART信号发送成功，然后写 7-bit EEPROM 从机地址和读/写控制位 (读) 到I2CDATA 寄存器.

   --- 写I2CDATA=SLA+R, 并清除 SI位 (STA, STO, SI, AA) = (0, 0, 1, X)

3. I2CSTATAUS=0x40时，表示SLA+R发送成功，本范例只接收1个字节，所以准备发送NACK给EEPROM（如果希望读多个字节，就回ACK给EEPROM）. .

   --- 清除SI位，并回NACK给EEPROM(STA, STO, SI, AA) = (0, 0, 1, 0)

   --- 清除SI位，并回ACK给EEPROM(STA, STO, SI, AA) = (0, 0, 1, 1)

4. I2CSTATUS=0x58时, 表示数据已经收到，NACK已经返回。发送STOP信号结束传输.

   --- 清除SI位，并发送STOP信号(STA, STO, SI, AA) = (0, 1, 1, 0)

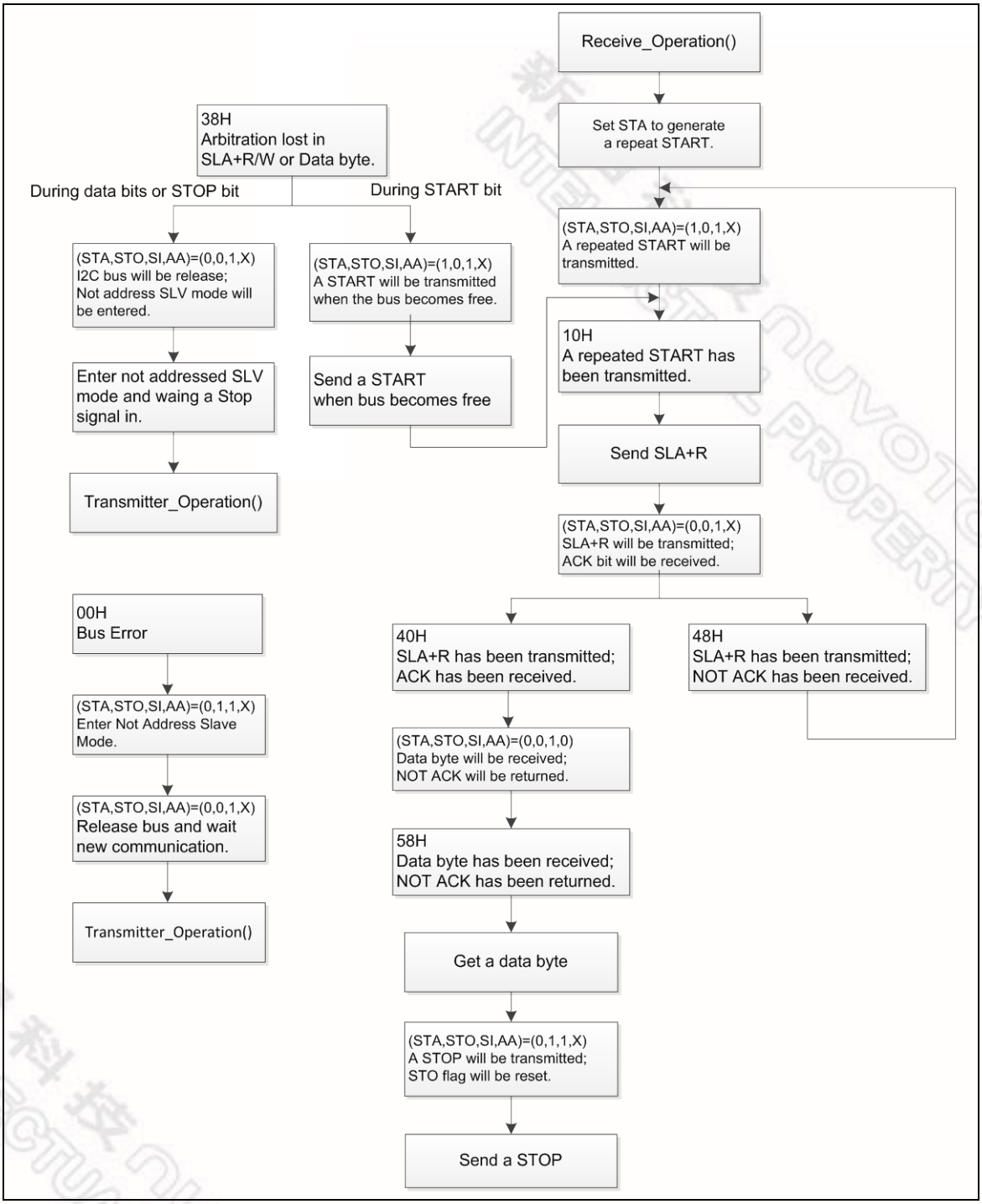


图 5-43 随机读接收操作

■ **其它事件 (多主机)**

在一些应用中，同一个I²C上有2个甚至多个主机将访问从机，并且主机有可能同时发送数据。NuMicro™的I²C控制器支持真正的多主机I²C总线，包含冲突检测和仲裁防止数据被破坏。.

1. 当 I2CSTATUS=0x38时, 表示"仲裁失败"。仲裁失败事件可能在发送START信号，数据或者STOP信号时发生。如果在发送START时收到该状态，I²C可以试着再次发送START信号。其他情况下，I²C应该发送STOP信号结束传输。.

   --- 发送 START 期间, 尝试再次发送START(STA, STO, SI, AA) = (1, 0, 1, X)

   --- 其它数据期间，发送STOP信号结束传输 (STA, STO, SI, AA) = (0, 0, 1, X)

2. 当 I2CSTATUS=0x00时, 表示 "总线错误" 。为了从总线错误状态恢复I²C总线，需要先将STO置为1，并清除SI，然后再将STO清为0。

   --- 首先将STO置为1并清除SI，I²C控制器将发送STOP信号 (STA, STO, SI, AA) = (0, 1, 1, X) 然后清除SI位(STA, STO, SI, AA) = (0, 0, 1, X)

5.9 PWM发生器和捕捉定时器 (PWM)

5.9.1 简介

该芯片有1组PWM 支持1组PWM发生器，可以配置成4个独立的PWM输出PWM0~PWM3, 或2组互补的PWM对, (PWM0, PWM1)与(PWM2, PWM3), 带2个可编程的死区发生器.

每组PWM发生器带有8位预分频，一个时钟除频提供5级时钟除频(1, 1/2, 1/4, 1/8, 1/16), 两个PWM定时器包括2个时钟选择，两个16位PWM向下计数计数器用于PWM周期控制，两个16位比较器用于PWM 占空比控制以及一个死区发生器. PWM发生器提供4个独立的PWM中断标志，当PWM向下计数周期达到零时触发中断。每个PWM中断源有独立的使能位。PWM发生器可以配置为单触发模式或连续输出PWM波形.

当PCR.DZEN01置位, PWM0 与 PWM1形成互补的PWM周期，这一对PWM的周期，占空比 和死区时间由PWM0定时器和死区发生器0决定. 同样, PWM 互补对(PWM2, PWM3)由 PWM2和死区发生器2控制，参考图 5-44到图 5-47查看PWM定时器内部结构.

为防止PWM输出抖动不稳定波形，16位向下计数计数器和16位比较器采用双缓存。当用户向计数器/比较器寄存器写入新的值时，只有当计数器/比较器的值下数到0后，新写入的值才会被重新加载到计数器/比较器。双缓存可以避免PWM输出时产生干扰波形.

当16位向下计数计数器达到0时，中断请求产生。如果PWM定时器被定义为连续模式，当向下计数器达到0时，会自动重新加载PWM计数寄存器的值(CNRx)并重新开始递减。如果定时器设为单次触发模式，向下计数器停止计数，并产生中断请求.

比较器的值用于调制高脉冲的宽度，在下数计数器的值等于比较器的值时，计数器控制逻辑将PWM输出变成高电平

PWM定时器可有输入捕捉功能。当捕捉功能使能时，PWM输出引脚被切为输入捕捉模式。捕捉器0和PWM0共享PWM0的定时器，捕捉器1和PWM1使用PWM1的定时器，以此类推。因此，在使用捕捉功能之前，用户必须先启动PMW定时器。捕捉功能使能后，当输入端口有**上升沿时**转变时，PWM计数器的值被锁存入**CRLR**，输入端口有**下降沿**转变时，PWM计数器的值被锁存入**CFLR**。设定CCR0.CRL_IE0[1] (使能上升沿锁存中断)和CCR0.CFL_IE0[2]] (使能下降沿锁存中断),可以配置捕捉器通道0的中断条件。同样设定CCR0.CRL_IE1[17] 和CCR0.CFL_IE1[18]，可以设定通道1。捕捉通道2和3的中断配置寄存器为CCR2。对每组而言，每当捕捉器触发中断 0/1/2/3 时，PWM计数器 0/1/2/3 的值也会同时被重新加载。

PWM可以支持的最大的捕捉频率由捕捉中断延迟决定. 捕捉中断发生时，软件至少执行以下三步: 第一步，读 PIIR 获取中断源，第二步，读CRLRx/CFLRx(x=0~3) 获取捕捉的值，第三步写1清PIIR为0. 如果中断延迟花T0完成，捕捉信号在（T0）间隔内一定不能改变. 此条件下，最大捕捉频率为1/T0.

例如:

HCLK = 50 MHz, PWM_CLK = 25 MHz, 中断处理时间为 900 ns

因此最大捕捉频率为1/900ns ≈ 1000 kHz

5.9.2.1

5.9.2 特性

PWM功能:

- 两个PWM发生器. 每组PWM有一个8位预分频，一个时钟除频，两个PWM定时器，一个死区发生器和两个PWM输出.

- 最高16位解析度
- PWM 中断与PWM周期同步
- One-shot 或 Auto-reload模式
- 最多1组 PWM(PWMA) 可支持4路PWM通道或2个PWM对

捕捉功能:

5.9.2.2

- 与PWM发生器共用定时器模块
- 支持4个捕捉输入通道，与4个PWM输出通道共享
- 每个通道支持1个上升沿锁存寄存器(CRLR), 一个下降沿锁存寄存器 (CFLR) 和捕捉中断标志(CAPIFx)

5.9.3 框图

图 5-44 到图 5-47描述PWM对的结构(PWM-Timer 0&1为一对， PWM-Timer 2&3 为一对).

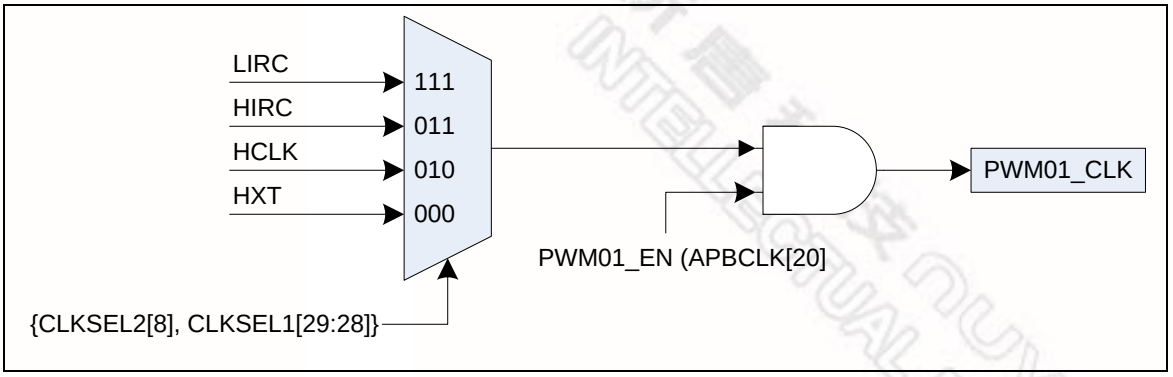


图 5-44 PWM 发生器 0 时钟源控制

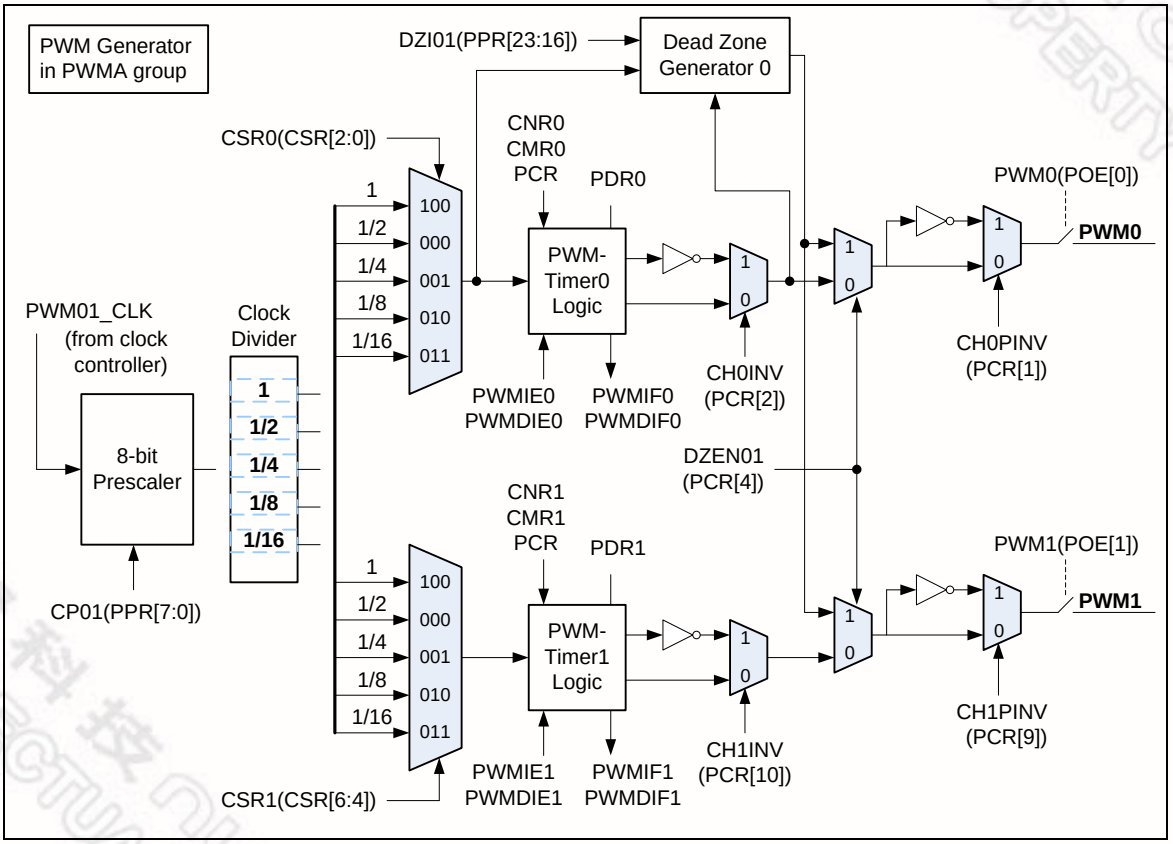


图 5-45 PWM 发生器 0 结构框图

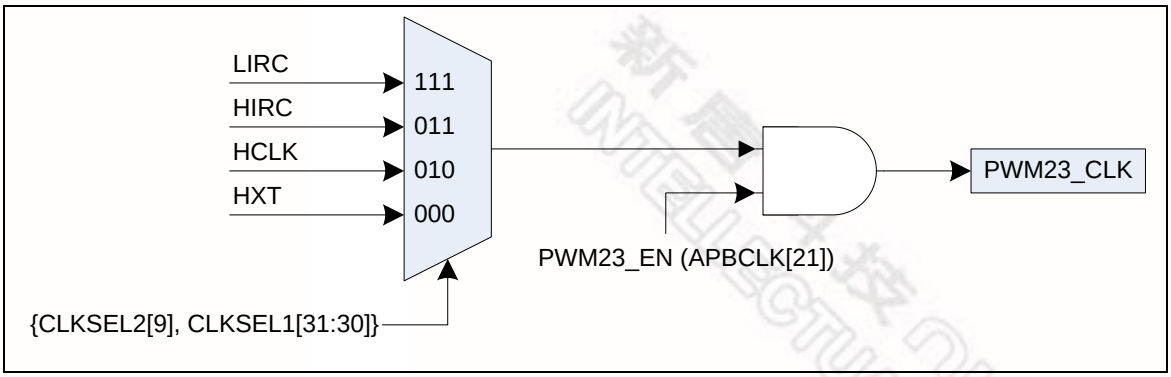


图 5-46 PWM 发生器 2 时钟源控制

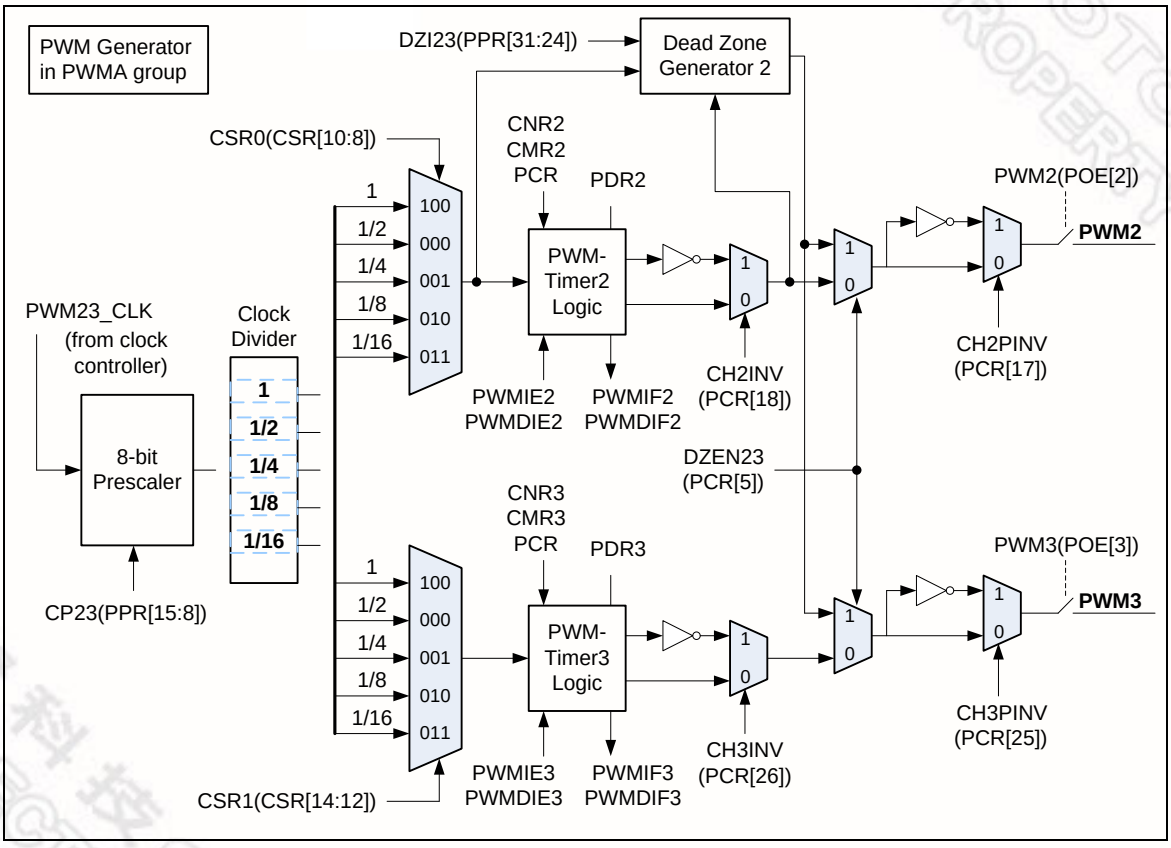


图 5-47 PWM 发生器 2 结构框图

5.9.4 功能描述

PWM-定时器操作

PWM 控制器支持两种操作模式: 边沿对齐和中心对齐.

5.9.4.1 边沿对齐 PWM (下数型计数器)

边沿对齐PWM输出模式下，16位PWM计数器从CNRn开始下数，当等于CMRn(旧)时，PWM发生器将输出反转成高电平。计数器继续下数直到0，同时PWM发生器反转输出成低电平。如果CHnMODE=1，CMRn(新)和CNRn(新)将被更新

PWM 周期 和占空比控制由向下计数的PWM寄存器(CNR)以及PWM比较寄存器(CMR)控制。PWM定时器工作时序如图 5-49. 脉宽宽度调制的公式如下，PWM定时器比较器图示如图 5-48. 注：在PWM功能使能前，相应GPIO管脚应配置成PWM功能(使能 POE 并禁止 CAPENR).

- PWM 频率 = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; xy代表01, 23, 取决于所选择的PWM通道.
- 占空比 = (CMR+1)/(CNR+1)
- CMR >= CNR: PWM 输出为高
- CMR < CNR: PWM低脉宽= (CNR-CMR) unit[1]; PWM高脉宽=(CMR+1) unit
- CMR = 0: PWM低脉宽 = (CNR) unit; PWM高脉宽= 1 unit

Note: [1] Unit = 一个 PWM 时钟周期.

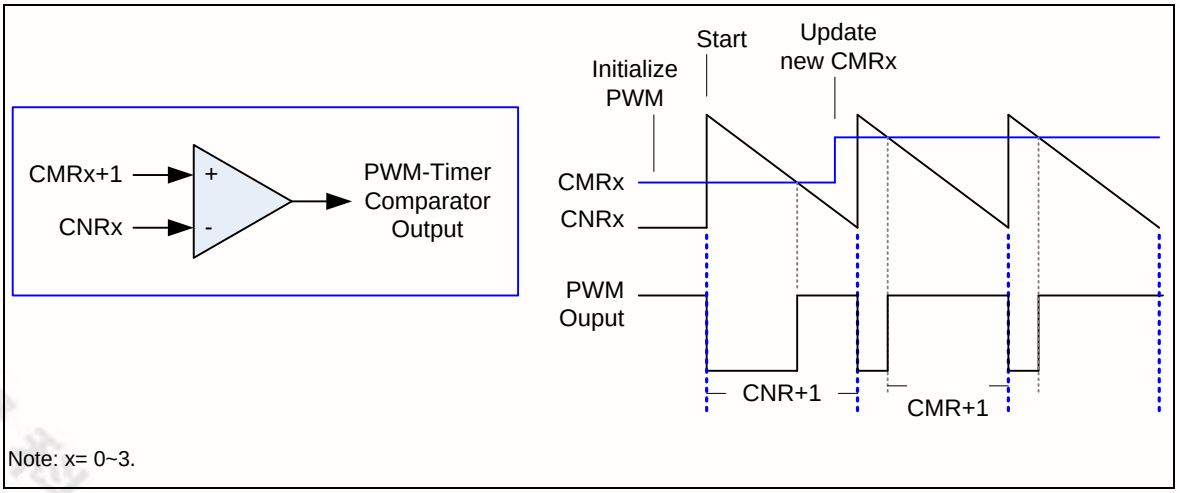


图 5-48 PWM-定时器内部比较器输出图例

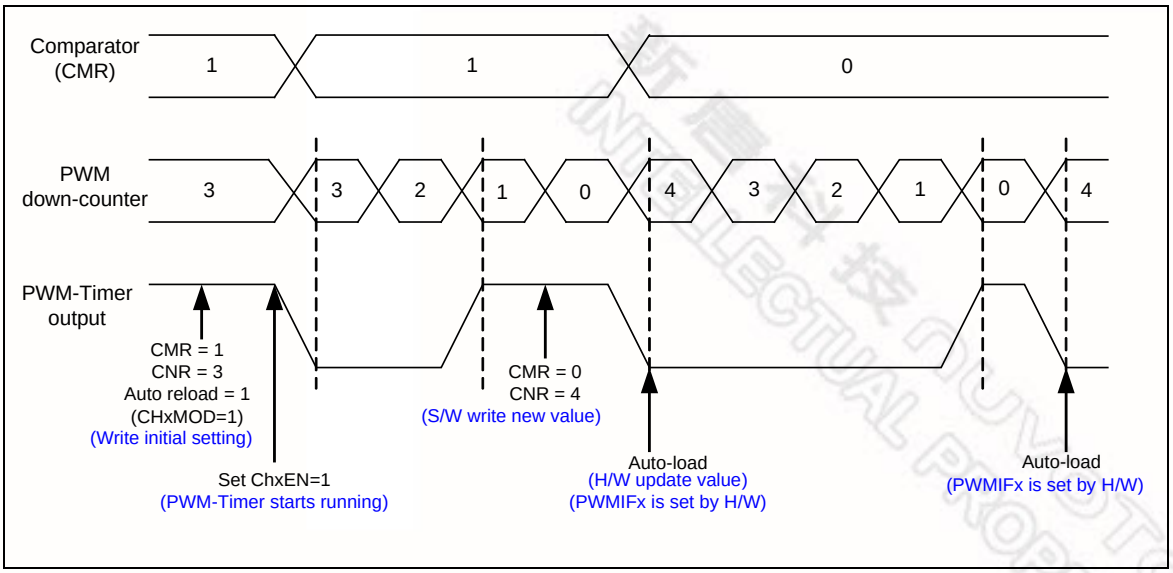


图 5-49 PWM-定时器操作时序

5.9.4.1.2 中心对齐PWM (下数/上数计数器)

当PWM定时器配置为上数/下数计数器模式时，中心对齐 PWM 信号由其产生。PWM 计数器从0开始上数，直到等于CMRn(旧)的值，之后PWM发生器将输出反转成低电平。计数器继续计数直到CNRn(旧)，之后计数器开始自动下数，当再次等于CMRn(旧)时,PWM发生器反转输出为高电平。一旦PWM计数器下溢，如果CHnMODE = 1; PWM周期寄存器CNRn(新)和占空比寄存器CMRn(新)将被更新。

中心对齐模式下, 如果INTTYPExx (PIER[17:16]) =0，下数计数器下溢时，就是在每个PWM周期的开始（结束），PWM 周期中断发生；如果INTTYPExx (PIER[17:16]) =1，上数计数器等于CNRn时，就是在每个PWM周期的中点处，PWM周期中断将发生.

- PWM 频率 = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; 依靠选择的PWM通道，xy 可以是 01, 23.
- 占空比 = [(2 x CMR) + 1]/[2 x (CNR+1)]
- CMR > CNR: PWM 输出总是高电平
- CMR <= CNR: PWM 低电平宽度= 2 x (CNR-CMR) + 1 unit[1]; PWM 高电平宽度 = (2 x CMR) + 1 unit
- CMR = 0: PWM 低电平宽度 = 2 x CNR + 1 unit; PWM 高电平宽度 = 1 unit

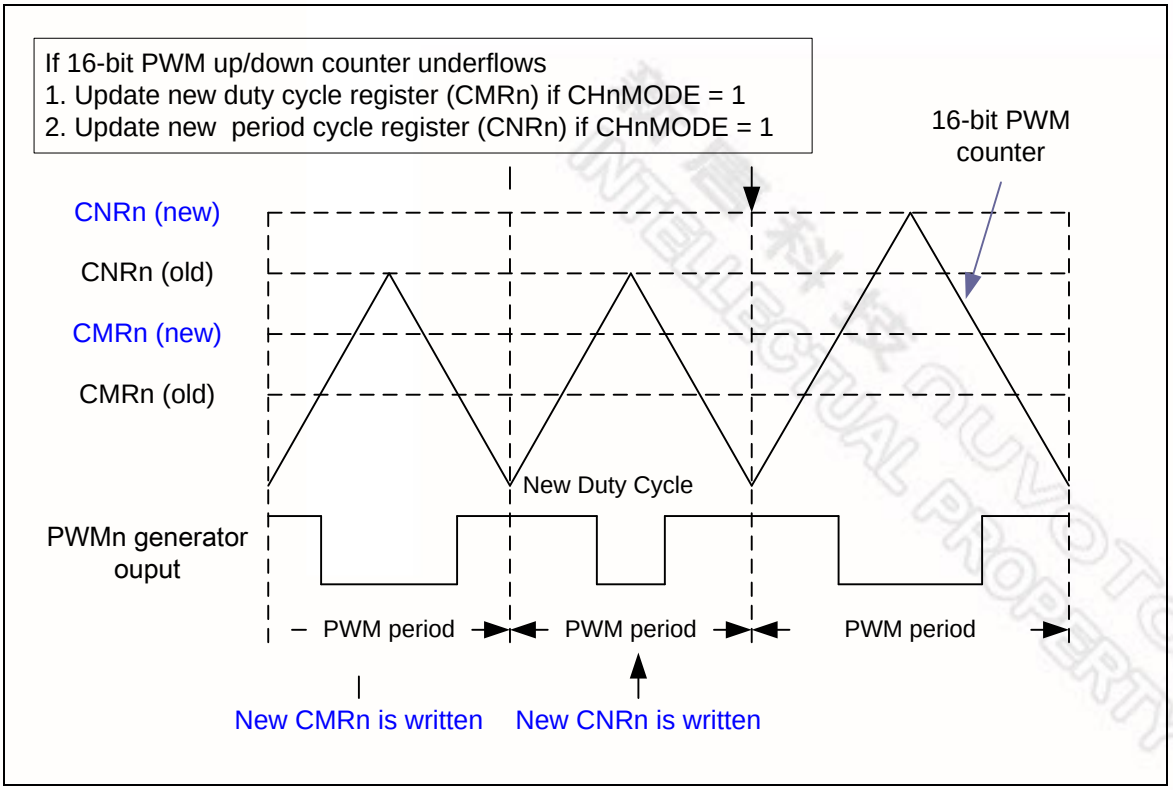


图 5-50 中心对齐模式输出波形

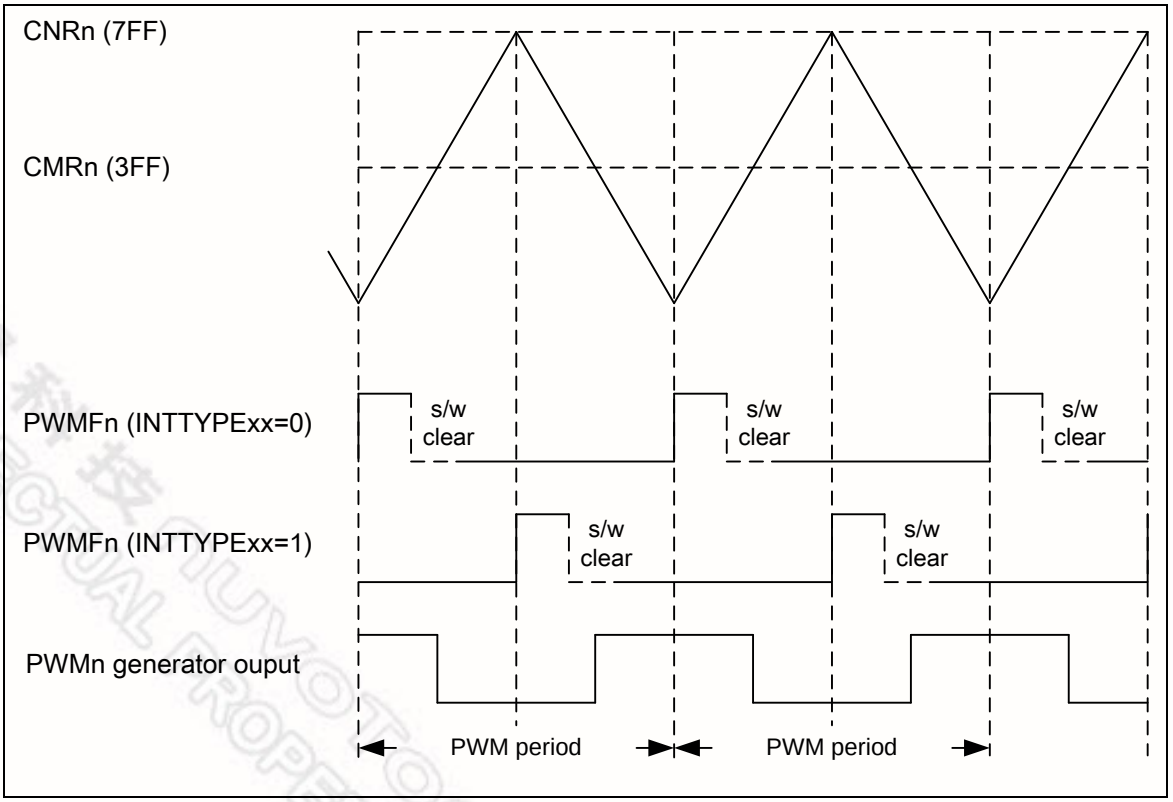


图 5-51 PWM 中心对齐中断发生时序图

PWM 双缓存, 自动重载以及单触发模式

PWM 定时器具有双缓存功能。寄存器新写入的值，在下一个周期加载，不会影响当前的定时器操作。PWM 计数器的值可以写入 CNRx，当前 PWM 计数器的值可从 PDRx 读出.

5.9.4.2

PWM 控制寄存器(PCR) 的CH0MOD 位定义PWM0是自动重载模式或是单触发模式。如果CH0MOD=1为自动重载模式，当PWM计数器计到0，MCU自动重载CNR0 值到PWM 计数器；如果CH0MOD=0为单次模式，PWM计数器到0以后，计数器将立即停止计数。PWM1~PWM3 与PWM0相同.

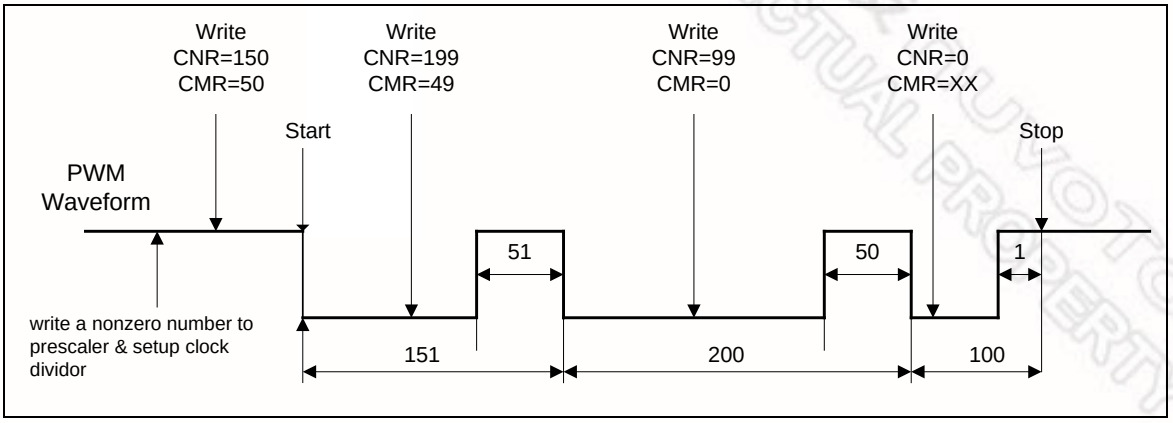


图 5-52 PWM 双缓存图解

5.9.4.3 *占空比调制*

双缓存允许CMRx字可以在任何时间被改写。写入的值在下一个周期才起作用.

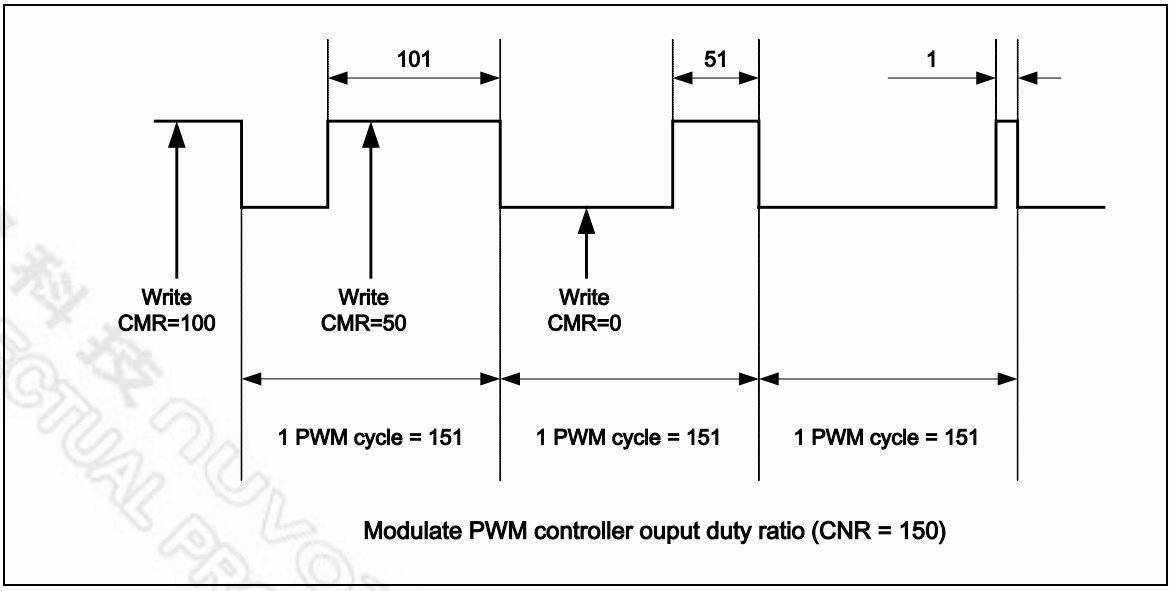


图 5-53 PWM 控制器输出占空比

死区发生器

PWM 控制器提供死区发生器. 用于保护电源器件. 该功能在PWM上升沿产生可编程的延迟时间，用户通过编程PPRx.DZI确定死区间隔.

5.9.4.4

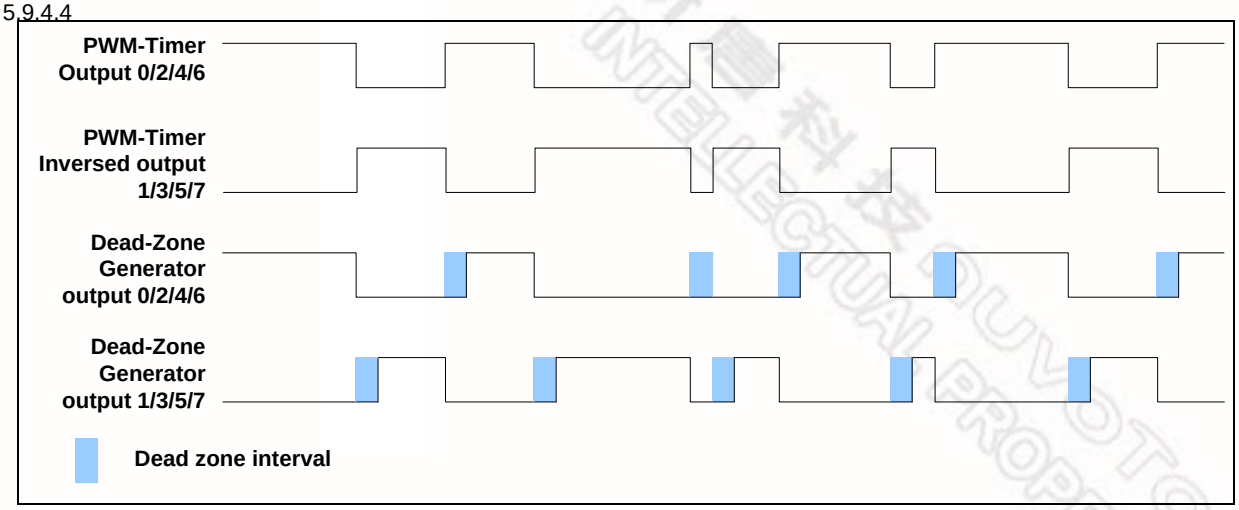


图 5-54 PWM 对输出死区发生器说明

5.9.4.5 *捕捉模式*

捕捉器0和PWM0使用同一个定时器，捕捉器1和PWM1使用另一组定时器，以此类推。在使用捕捉功能之前，必须预先配置PMW定时器。当输入信号有上升沿转变时，PWM计数器的值将存入CRLRx寄存器；当输入信号有下降沿转变时，PWM计数器的值将存入CFLRx寄存器。捕捉器通道0通过设定CCR0[1] (使能上升沿锁存中断)和CCR0[2](使能下降沿锁存中断)来配置中断条件。同样捕捉器通道1通过设定CCR0[17] 和CCR0[18]。依此类推。每当捕捉控制器触发捕捉中断时，相应的PWM 计数器会同时重新装载 CNRX 的值。注: 在捕捉功能使能前，相应的管脚必须配置为捕获模式(禁止POE并使能CAPENR).

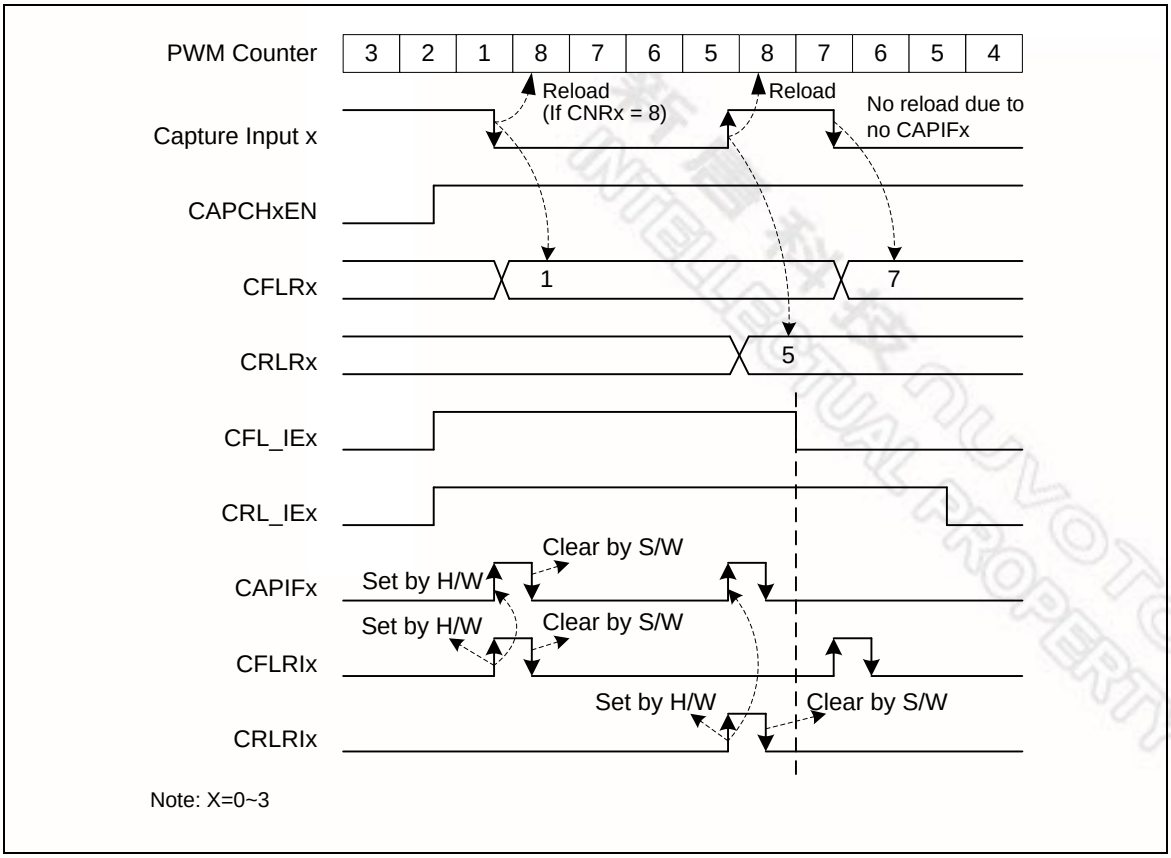


图 5-55 捕捉操作时序

在上述范例中，CNR 为8:

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

5.9.4.6

PWM PDMA 功能

捕捉模式下，每个PWM通道都支持PDMA传输功能. 以通道0为例，当PDMA使能位（在CAPCTL寄存器）被置时，发生捕获事件之前，捕获模块将先发送一个请求给PDMA控制器。PDMA 控制器将回 ACK 给捕获模块并传输 CAP0RFPDMA 的值到内存. 通过设定 CAP0PDMAMOD, PDMA 可以传输上升沿锁存数据或者下降沿锁存数据或者全部数据到内存。

5.9.4.7

当使用PDMA既传输上升沿又传输下降沿数据时，记得设定PWM_CAPCTL 寄存器中的 CAP0RFORDER 来指定数据传输的顺序(下降沿锁存数据先传输还是上升沿锁存数据先传输).

PWM-定时器中断结构

有4个PWM中断，PWM0_INT~PWM3_INT,对NVIC来说，这些中断被组成PWMA_INT。 PWM 0 与捕捉器0共用同一组中断。PWM1 与捕捉器1 共用同一组中断，以此类推。因此，在同一个通道 PWM功能和捕捉功能不能同时发生。 图 5-56 描述了PWM定时器中断的结构。

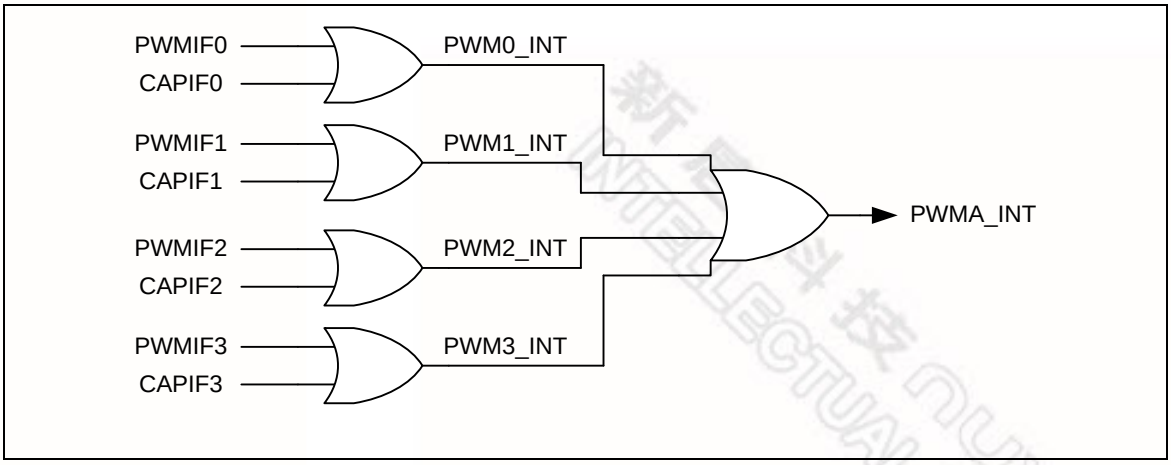


图 5-56 PWM A 组 PWM-定时器中断结构图

5.9.4.8 *PWM-定时器输出步骤*

启动PWM推荐下列步骤：

1. 写CLKSEL1和CLKSEL2寄存器选择PWM的时钟源
2. 写APBCLK寄存器使能PWM的时钟
3. 配置时钟选择寄存器 (CSR)选择时钟除频值
4. 配置预分频寄存器(PPR)
5. 配置反转打开/关闭，死区打开/关闭，自动重载/单触发模式以及停止PWM定时器 (PCR)
6. 配置比较器寄存器(CMR) 设定PWM 占空比.
7. 配置PWM计数器寄存器 (CNR) 设定PWM 周期.
8. 配置中断使能寄存器 (PIER)
9. 配置相应PWM通道GPIO脚为PWM功能（GPx_MFP寄存器和ALT_MFP寄存器）
10. 使能 POE 寄存器相应引脚输出功能，关闭 CAPENR).
11. 使能PWM定时器开始运行 (PCR 中的CHxEN = 1)

5.9.4.9 *PWM-定时器关闭步骤*

方式 1:

设定 16-位计数器(CNR)为0 ，并查看PDR状态(计数器的当前值)。当PDR达到0 ，关闭PWM定时器(PCR的CHxEN位). ***(推荐)***

方式 2:

设定16位计数器(CNR)为0，当中断发生时。在中断内关闭PWM定时器(PCR的CHxEN位). ***(推荐)***

方式 3:

直接关闭PWM定时器(PCR的CHxEN位). (不推荐)

不推荐方式3是因为禁止CHxEN 将立即停止PWM输出信号，会导致PWM输出的占空比改变, 这可能引起电机控制电路的损坏

捕捉开始步骤

1. 写CLKSEL1和CLKSEL2寄存器选择PWM的时钟源

5.9.4.10

2. 写APBCLK寄存器使能PWM的时钟
3. 配置时钟选择寄存器 (CSR)选择时钟除频值
4. 配置预分频寄存器(PPR)
5. 配置通道使能，上升/下降沿中断使能以及输入信号反转打开/关闭(CCR0, CCR1)
6. 配置PWM计数器寄存器(CNR)
7. 配置相应的GPIO为捕捉功能（GPx_MFP寄存器和ALT_MFP寄存器）
8. 禁止POE相应位输出并使能 CAPENR
9. 使能PWM定时器开始运行(设定PCR 中的CHxEN = 1)

5.9.5 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
PWM 基地址: **PWMA_BA = 0x4004_0000**				
PPR	PWMA_BA+0x00	R/W	PWM A组预分频寄存器	0x0000_0000
CSR	PWMA_BA+0x04	R/W	PWM A组时钟选择寄存器	0x0000_0000
PCR	PWMA_BA+0x08	R/W	PWM A组控制寄存器	0x0000_0000
CNR0	PWMA_BA+0x0C	R/W	PWM A组计数寄存器0	0x0000_0000
CMR0	PWMA_BA+0x10	R/W	PWM A组比较寄存器0	0x0000_0000
PDR0	PWMA_BA+0x14	R	PWM A 组数据寄存器0	0x0000_0000
CNR1	PWMA_BA+0x18	R/W	PWM A组计数寄存器1	0x0000_0000
CMR1	PWMA_BA+0x1C	R/W	PWM A组比较寄存器1	0x0000_0000
PDR1	PWMA_BA+0x20	R	PWM A 组数据寄存器1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWM A组计数寄存器2	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWM A组比较寄存器2	0x0000_0000
PDR2	PWMA_BA+0x2C	R	PWM A 组数据寄存器2	0x0000_0000
CNR3	PWMA_BA+0x30	R/W	PWM A组计数寄存器3	0x0000_0000
CMR3	PWMA_BA+0x34	R/W	PWM A组比较寄存器3	0x0000_0000
PDR3	PWMA_BA+0x38	R	PWM A 组数据寄存器3	0x0000_0000
PBCR	PWMA_BA+0x3C	R/W	PWM 向后兼容寄存器	0x0000_0000
PIER	PWMA_BA+0x40	R/W	PWM A组中断使能寄存器	0x0000_0000
PIIR	PWMA_BA+0x44	R/W	PWM A组中断标志寄存器	0x0000_0000
CCR0	PWMA_BA+0x50	R/W	PWM A 组捕捉控制寄存器0	0x0000_0000
CCR2	PWMA_BA+0x54	R/W	PWM A 组捕捉控制寄存器2	0x0000_0000
CRLR0	PWMA_BA+0x58	R	PWM A组捕捉上升沿锁存寄存器(Channel 0)	0x0000_0000
CFLR0	PWMA_BA+0x5C	R	PWM A组捕捉下降沿锁存寄存器(Channel 0)	0x0000_0000
CRLR1	PWMA_BA+0x60	R	PWM A组捕捉上升沿锁存寄存器(Channel 1)	0x0000_0000
CFLR1	PWMA_BA+0x64	R	PWM A组捕捉下降沿锁存寄存器(Channel 1)	0x0000_0000
CRLR2	PWMA_BA+0x68	R	PWM A组捕捉上升沿锁存寄存器(Channel 2)	0x0000_0000

CFLR2	PWMA_BA+0x6C	R	PWM A组捕捉下降沿锁存寄存器(Channel 2)	0x0000_0000
CRLR3	PWMA_BA+0x70	R	PWM A组捕捉上升沿锁存寄存器(Channel 3)	0x0000_0000
CFLR3	PWMA_BA+0x74	R	PWM A组捕捉下降沿锁存寄存器(Channel 3)	0x0000_0000
CAPENR	PWMA_BA+0x78	R/W	PWM A 组捕捉输入 0~3 使能寄存器	0x0000_0000
POE	PWMA_BA+0x7C	R/W	PWM A 组使能通道 0~3输出	0x0000_0000
TCON	PWMA_BA+0x80	R/W	PWM A组触发控制寄存器	0x0000_0000
TSTATUS	PWMA_BA+0x84	R/W	PWM A组触发状态寄存器	0x0000_0000
SYNCBUSY0	PWMA_BA+0x88	R	PWM A组同步忙状态寄存器0	0x0000_0000
SYNCBUSY1	PWMA_BA+0x8C	R	PWM A组同步忙状态寄存器1	0x0000_0000
SYNCBUSY2	PWMA_BA+0x90	R	PWM A组同步忙状态寄存器2	0x0000_0000
SYNCBUSY3	PWMA_BA+0x94	R	PWM A组同步忙状态寄存器3	0x0000_0000
CAPPDMACTL	PWMA_BA+0xC0	R/W	PWM A组 PDMA 控制寄存器	0x0000_0000
CAP0PDMA	PWMA_BA+0xC4	R	PWM A组PDMA 通道0数据寄存器	0x0000_0000
CAP1PDMA	PWMA_BA+0xC8	R	PWM A组PDMA通道1数据寄存器	0x0000_0000
CAP2PDMA	PWMA_BA+0xCC	R	PWM A组PDMA通道2数据寄存器	0x0000_0000
CAP3PDMA	PWMA_BA+0xD0	R	PWM A组PDMA通道3数据寄存器	0x0000_0000

5.9.6 寄存器描述

PWM 预分频寄存器 (PPR)

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

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

Bits	描述	
[31:24]	**DZI23**	**通道2与通道3的死区间隔** (PWM A组的PWM2 和 PWM3 对,) 该8位寄存器决定死区长度. 死区长度时间单位由相关CSR位决定.
[23:16]	**DZI01**	**通道0与通道1的死区间隔**(PWM A组的PWM0和PWM1 对,) 该8位寄存器决定死区长度. 死区长度时间单位由相关CSR位决定.
[15:8]	**CP23**	**时钟预分频 2** (PWM A组的PWM-timer2 & 3) 时钟在输入到相应的PWM定时器之前被(CP23 + 1) 除频 如果CP23=0, 预分频器2输出时钟停止。相应PWM定时器也停止.
[7:0]	**CP01**	**时钟预分频 0** (PWM A组的PWM-timer 0 & 1) 时钟在输入到相应的PWM定时器之前被(CP01 + 1)除频 如果CP01=0, 预分频器0输出时钟停止。相应PWM定时器也停止.

PWM 时钟选择寄存器 (CSR)

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

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

Bits	描述	
[31:15]	保留	保留
[14:12]	CSR3	**PWM 定时器 3 时钟源选择** 选择PWM定时器的时钟输入. CSR3 [14:12] / 输入时钟除频 100 / 1 011 / 16 010 / 8 001 / 4 000 / 2
[11]	保留	保留
[10:8]	CSR2	**PWM定时器 2 时钟源选择** 选择PWM定时器的时钟输入. (表与 CSR3相同)
[7]	保留	保留
[6:4]	CSR1	**PWM 定时器 1 时钟源选择** 选择PWM定时器的时钟输入. (表与 CSR3相同)
[3]	保留	保留
[2:0]	CSR0	**PWM 定时器 0 时钟源选择**

		选择PWM定时器时钟输入. (表与 CSR3相同)

PWM控制寄存器(PCR)

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

31	30	29	28	27	26	25	24
PWMTYPE2	PWMTYPE01	保留		CH3MOD	CH3INV	CH3PINV	CH3EN
23	22	21	20	19	18	17	16
保留				CH2MOD	CH2INV	CH2PINV	CH2EN
15	14	13	12	11	10	9	8
保留				CH1MOD	CH1INV	CH1PINV	CH1EN
7	6	5	4	3	2	1	0
保留		DZEN23	DZEN01	CH0MOD	CH0INV	CH0PINV	CH0EN

Bits	描述	
[31]	PWMTYPE23	**PWM23 对齐类型选择 (PWM2 和 PWM3对)** 0 = 边沿对齐. 1 = 中心对齐.
[30]	PWMTYPE01	**PWM01 对齐类型选择 (PWM0 和 PWM1 对)** 0 = 边沿对齐. 1 = 中心对齐.
[29:28]	保留	保留
[27]	CH3MOD	**PWM-定时器 3 自动加载/单触发模式**(PWM timer 3 for group A) 1 = 自动重载模式 0 = 单触发模式 **注:** 如果该位的值变化，会使CNR3 和CMR3 清0.
[26]	CH3INV	**PWM-定时器 3 输出反转使能**(PWM timer 3 for group A) 1 = 反转打开，PWM输出前一刻反转，就是在加入死区之后反转 0 = 反转关闭
[25]	CH3PINV	**PWM-定时器 3 输出极性反转使能(PWM Timer 3 for Group A)** 1 = PWM3 输出极性反转. 在加死区之前反转 0 = PWM3 输出极性不反转.
[24]	CH3EN	**PWM-定时器 3 使能** (PWM timer 3 for group A) 1 =相应的PWM定时器开始运行

		0 =相应的PWM定时器停止运行
[23:20]	**保留**	保留
[19]	**CH2MOD**	**PWM-定时器 2 自动重载/单触发模式**(PWM timer 2 for group A) 1 = 自动重载模式 0 = 单触发模式 **注:** 如果该位的值变化，会使CNR2和CMR2清0.
[18]	**CH2INV**	**PWM-定时器 2 输出反转使能** (PWM timer 2 for group A) 1 = 反转打开，PWM输出前一刻反转，就是在加入死区之后反转 0 = 反转关闭
[17]	**CH2PINV**	**PWM-定时器 2 输出极性反转使能 (PWM Timer 2 for Group A)** 1 = PWM2 输出极性反转. 在加死区之前反转 0 = PWM2输出极性不反转
[16]	**CH2EN**	**PWM-定时器 2 使能** (PWM timer 2 for group A) 1 = 相应的PWM定时器开始运行 0 =相应的PWM定时器停止运行
[15:12]	**保留**	保留
[11]	**CH1MOD**	**PWM-定时器 1 自动加载/单触发模式** (PWM timer 1 for group A) 1 = 自动重载模式 0 = 单触发模式 **注:** 如果该位的值变化，会使CNR1和CMR1清0.
[10]	**CH1INV**	**PWM-定时器 1 输出反转使能**(PWM timer 1 for group A) 1 = 反转打开，PWM输出前一刻反转，就是在加入死区之后反转 0 = 反转关闭
[9]	**CH1PINV**	**PWM-定时器 1 输出极性反转 (PWM Timer 1 for Group A)** 1 = PWM1输出极性反转. 在加死区之前反转 0 = PWM1输出极性不反转.
[8]	**CH1EN**	**PWM-定时器 1 使能** (PWM timer 1 for group A) 1 =相应的PWM定时器开始运行 0 =相应的PWM定时器停止运行
[7:6]	**保留**	保留
[5]	**DZEN23**	**死区 2 发生器使能** (PWM2 and PWM3 pair for PWM group A) 1 = 使能 0 = 禁用 **注**: 当死区发生器使能，PWM A组的PWM2和PWM3将成为一对，输出互补信号.

[4]	**DZEN01**	**死区 0 发生器使能** (PWM0 and PWM1 pair for PWM group A) 1 = 使能 0 = 禁用 **注**: 当死区发生器使能，PWM A组的PWM0和PWM1将成为一对，输出互补信号.
[3]	**CH0MOD**	**PWM-定时器 0 自动加载/单触发模式** (PWM timer 0 for group A) 1 = 自动重载模式 0 = 单触发模式 **注:** 如果该位的值变化，会使CNR0和CMR0清0.
[2]	**CH0INV**	**PWM-定时器 0 输出反转使能**(PWM timer 0 for group A) 1 = 反转打开，PWM输出前一刻反转，就是在加入死区之后反转 0 = 反转关闭
[1]	**CH0PINV**	**PWM-定时器 0 输出极性反转使能 (PWM Timer 0 for Group A)** 1 = PWM0输出极性反转，在加死区之前反转. 0 = PWM0输出极性反转.
[0]	**CH0EN**	**PWM-定时器 0 使能** (PWM timer 0 for group A) 1 =相应的PWM定时器开始运行 0 =相应的PWM定时器停止运行

PWM计数器寄存器3-0 (CNR3-0)

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

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**CNRx**	**PWM 定时器载入值** CNR 决定PWM的周期. PWM 频率 = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; xy代表01, 23, 取决于所选择的PWM通道. 边沿对齐模式: ● 占空比= (CMR+1)/(CNR+1). ● CMR >= CNR: PWM输出高. ● CMR < CNR: PWM l低脉宽= (CNR-CMR) unit; PWM高脉宽= (CMR+1) unit. ● CMR = 0: PWM低脉宽 = (CNR) unit; PWM 高脉宽 = 1 unit 中心对齐模式: ● 占空比= [(2 x CMR) + 1]/[2 x (CNR+1)]. ● CMR > CNR: PWM 输出高. ● CMR <= CNR: PWM l低脉宽= 2 x (CNR-CMR) + 1 unit; PWM 高脉宽 = (2 x CMR) + 1 unit. ● CMR = 0: PWM 低脉宽 = 2 x CNR + 1 unit; PWM高脉宽= 1 unit

		(Unit = one PWM clock cycle) **注**: CNR写入数据后将在下一个PWM周期生效. **注:** 当PWM 工作在中心对齐模式时，CNR 的值应该被设在0x0000 到 0xFFFE之间。如果f CNR 等于 0xFFFF， PWM 工作将不正常. **注:** 当CNR 等于 0时，PWM 输出总是高电平.

PWM比较寄存器3-0 (CMR3-0)

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

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**CMRx**	**PWM**比较寄存器 CMR 决定 PWM 的占空比. PWM 频率 = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; xy代表 01, 23, 取决于所选择的PWM通道. ● 占空比= (CMR+1)/(CNR+1). ● CMR >= CNR: PWM输出高. ● CMR < CNR: PWM低脉宽= (CNR-CMR) unit; PWM高脉宽= (CMR+1) unit. ● CMR = 0: PWM低脉宽= (CNR) unit; PWM高脉宽= 1 unit 中心对齐模式: ● 占空比= [(2 x CMR) + 1]/[2 x (CNR+1)]. ● CMR > CNR: PWM 输出高. ● CMR <= CNR: PWM l低脉宽= 2 x (CNR-CMR) + 1 unit; PWM 高脉宽 = (2 x CMR) + 1 unit. ● CMR = 0: PWM 低脉宽 = 2 x CNR + 1 unit; PWM高脉宽= 1 unit (Unit = one PWM clock cycle) Note: CMR写入数据后将在下一个PWM周期生效.

PWM数据寄存器3-0 (PDR 3-0)

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

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**PDRx**	**PWM**数据寄存器 用来查询16位计数器的当前值.

PWM 向后兼容寄存器

寄存器	偏移量	R/W	描述	复位后的值
PBCR	PWM_BA0+0x3C	R/W	PWM 向后兼容寄存器	0x0000_0000

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

Bits	描述	
[31:1]	保留	保留
[0]	BCn	**PWM 向后兼容寄存器** 0 = PWM 寄存器兼容Medium Density系列 1 = PWM 寄存器不兼容Medium Density系列 参考寄存器CCR0/CCR2 的第 6, 7, 22, 23位描述

PWM中断使能寄存器(PIER)

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

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留						INTTYPE23	INTTYPE01
15	14	13	12	11	10	9	8
保留				PWMDIE3	PWMDIE2	PWMDIE1	PWMDIE0
7	6	5	4	3	2	1	0
保留				PWMIE3	PWMIE2	PWMIE1	PWMIE0

Bits	描述	
[31:18]	保留	保留
[17]	INTTYPE23	**PWM12 中断类型选择 (PWM2 and PWM3 Pair for PWM Group A)** 1 = 如果PWM计数器的值等于CNRn寄存器的值，PWMIFn 将被置. 0 = 如果PWM计数器的值下溢，PWMIFn将被置. **Note:**这个比特只有在PWM 中心对齐模式才有效.
[16]	INTTYPE01	**PWM01中断类型选择(PWM0 and PWM1 Pair for PWM Group A)** 1 =如果PWM计数器的值等于CNRn寄存器的值，PWMIFn 将被置. 0 =如果PWM计数器的值下溢，PWMIFn将被置. **Note:** 这个比特只有在PWM 中心对齐模式才有效.
[11]	PWMDIE3	**PWM 通道 3 Duty 中断使能** 1 = 使能. 0 = 禁止.
[10]	PWMDIE2	**PWM通道 2 Duty 中断使能** 1 =使能. 0 =禁止.
[9]	PWMDIE1	**PWM通道 1 Duty 中断使能** 1 =使能. 0 =禁止.

[8]	**PWMDIE0**	**PWM通道 0 Duty 中断使能** 1 =使能. 0 =禁止.
[7:4]	**Reserved**	**Reserved**
[3]	**PWMIE3**	**PWM 通道 3 中断使能** 1 = 使能 0 = 禁用
[2]	**PWMIE2**	**PWM 通道 2中断使能** 1 = 使能 0 = 禁用
[1]	**PWMIE1**	**PWM 通道 1中断使能** 1 = 使能 0 = 禁用
[0]	**PWMIE0**	**PWM 通道 0 中断使能** 1 = 使能 0 = 禁用

PWM 中断标志寄存器 (PIIR)

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

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				PWMDIF3	PWMDIF2	PWMDIF1	PWMDIF0
7	6	5	4	3	2	1	0
保留				PWMIF3	PWMIF2	PWMIF1	PWMIF0

Bits	描述	
[31:12]	保留	保留
[11]	PWMDIF3	**PWM 通道 3 Duty 中断标志** 当PWM通道3计数器下数到等于CMR3寄存器的值时，这个标志将由硬件设置，软件可以写1清除.中心对齐和边沿对齐模式都支持该中断，但是中心对齐模式中断发生的时机为计数器下数等于CMR3的时候才触发 **注:** 如果 CMR 寄存器的值等于 CNR, 这个标志不工作
[10]	PWMDIF2	**PWM 通道 2 Duty 中断标志** 当PWM通道2计数器下数到等于CMR2寄存器的值时，这个标志将由硬件设置，软件可以写1清除. 中心对齐和边沿对齐模式都支持该中断，但是中心对齐模式中断发生的时机为计数器下数等于CMR2的时候才触发 **注:** 如果 CMR 寄存器的值等于 CNR, 这个标志不工作
[9]	PWMDIF1	**PWM 通道 1 Duty 中断标志** 当PWM通道1计数器下数到等于CMR1寄存器的值时，这个标志将由硬件设置，软件可以写1清除. 中心对齐和边沿对齐模式都支持该中断，但是中心对齐模式中断发生的时机为计数器下数等于CMR1的时候才触发 **注:** 如果 CMR 寄存器的值等于 CNR, 这个标志不工作
[8]	PWMDIF0	**PWM 通道 0 Duty 中断标志** 当PWM通道0计数器下数到等于CMR0寄存器的值时，这个标志将由硬件设置，软件可以写1清除. 中心对齐和边沿对齐模式都支持该中断，但是中心对齐模式中断发生的时机为计数器下数等于CMR0的时候才触发 **注:** 如果 CMR 寄存器的值等于 CNR, 这个标志不工作
[7:4]	保留	保留
[3]	PWMIF3	**PWM 通道 3 中断状态**

		当PWM3 计数器的值达到中断要求(依靠PIER寄存器INTTYPE23比特的设定，边沿对齐模式下，计数器下溢)，硬件自动将该位置1（如果 PWM 3 中断使能位（PWMIE3）为 1）。软件写1清除该位
[2]	**PWMIF2**	**PWM 通道 2 中断状态** 当PWM2计数器的值达到中断要求(依靠PIER寄存器INTTYPE23比特的设定，边沿对齐模式下，计数器下溢)，硬件自动将该位置1（如果 PWM 2 中断使能位（PWMIE2）为 1）。软件写1清除该位
[1]	**PWMIF1**	**PWM 通道 1 中断状态** 当PWM1计数器的值达到中断要求(依靠PIER寄存器INTTYPE23比特的设定，边沿对齐模式下，计数器下溢)，硬件自动将该位置1（如果 PWM 1 中断使能位（PWMIE1）为 1）。软件写1清除该位
[0]	**PWMIF0**	**PWM 通道 0 中断状态** 当PWM0计数器的值达到中断要求(依靠PIER寄存器INTTYPE23比特的设定，边沿对齐模式下，计数器下溢)，硬件自动将该位置1（如果 PWM 0 中断使能位（PWMIE0）为 1）。软件写1清除该位

注:通过将PIIR寄存器相应位写1，用户可以清除每个中断标志.

捕捉控制寄存器(CCR0)

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

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
CFLRI1	CRLRI1	保留	CAPIF1	CAPCH1EN	CFL_IE1	CRL_IE1	INV1
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
CFLRI0	CRLRI0	保留	CAPIF0	CAPCH0EN	CFL_IE0	CRL_IE0	INV0

Bits	描述	
[31:24]	保留	保留
[23]	CFLRI1	**CFLR1锁定标志位** 当 PWM 组输入通道1发生下降沿变化时, CFLR1 锁存PWM向下计数器的值，该位由硬件置位. 如果BCn为0，软件写0清该位, 如果BCn为1，软件写1清该位.
[22]	CRLRI1	**CRLR1锁定标志位** 当 PWM 组输入通道1发生上升沿变化时, CRLR1 锁存PWM向下计数器的值，该位由硬件置位. 如果BCn为0，软件写0清该位, 如果BCn为1，软件写1清该位.
[5]	保留	保留
[20]	CAPIF1	**通道 1 捕捉中断标志** 如果PWM组通道1上升锁存中断使能(CRL_IE1=1), PWM组通道1发生上升沿转变会使CAPIF1 为高; 同样，如果PWM组通道1下降锁定中断使能(CFL_IE1=1)，下降沿会使CAPIF1置高. 写 1 清该位为0
[19]	CAPCH1EN	**通道 1 捕捉功能使能** 1 = 使能PWM组通道1的捕捉功能 0 = 禁用PWM组通道1的捕捉功能 当使能时，捕捉功能将锁存PWM计数器的值，并存储到CRLR (上升锁存) 或CFLR (下降锁存). 当禁用时，捕捉器不更新CRLR 或CFLR, 并关闭通道1中断.
[18]	CFL_IE1	**通道 1 下降沿锁存中断使能**

		1 = 使能下降沿锁存中断 0 = 禁用下降沿锁存中断 使能时，如果检测到PWM组通道1有下降沿转变，捕捉中断产生.
[17]	**CRL_IE1**	**通道 1 上升沿锁存中断使能** 1 = 使能上升沿锁存中断 0 = 禁用上升沿锁存中断 使能时，如果检测到PWM组通道1有上升沿转变，捕捉中断产生.
[16]	**INV1**	**使能通道 1 反转** 1 =反转打开。来自GPIO引脚的输入信号反转之后再输入到捕捉器 0 =反转关闭
[15:8]	**保留**	保留
[7]	**CFLRI0**	**CFLR0锁存标志位** 当PWM组输入通道0发生下降沿转变时，CFLR0 锁存PWM向下计数值，该位由硬件置位. 如果BCn为0，软件写0清该位, 如果BCn为1，软件写1清该位.
[6]	**CRLRI0**	**CRLR0锁存标志位** 当PWM组输入通道0发生上升沿转变时, CRLR0锁存PWM向下计数值，该位由硬件置位. 如果BCn为0，软件写0清该位, 如果BCn为1，软件写1清该位.
[5]	**保留**	保留
[4]	**CAPIF0**	**通道 0捕捉中断标志** 如果PWM组通道0上升锁存中断使能(CRL_IE0=1), PWM组通道0发生上升沿转变会使CAPIF0 为高; 同样，如果PWM组通道0下降锁存中断使能(CFL_IE0=1)，下降沿转变会使CAPIF0置高. 写 1 清该位为0
[3]	**CAPCH0EN**	**通道0 捕捉功能使能** 1 = 使能PWM组通道0的捕捉功能 0 = 禁用PWM组通道0的捕捉功能 当使能时，捕捉功能锁存PWM计数器的值，并存储到CRLR (上升锁存) 或CFLR (下降锁存). 当禁用时，捕捉器不更新CRLR 或CFLR, 并关闭通道0中断.
[2]	**CFL_IE0**	**通道 0 下降沿锁存中断使能** 1 = 使能下降沿锁存中断 0 = 禁用下降沿锁存中断 使能时，如果检测到PWM组通道0有下降沿转变，捕捉中断产生.
[1]	**CRL_IE0**	**通道 0 上升沿锁存中断使能** 1 = 使能上升沿锁存中断 0 = 禁用上升沿锁存中断

		使能时，如果检测到PWM组通道0有上升沿转变，捕捉中断产生.
[0]	**INV0**	**使能通道 0 反向** 1 =反转打开。来自GPIO引脚的输入信号反转之后再输入到捕捉器 0 =反转关闭

捕捉控制寄存器(CCR2)

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

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
CFLRI3	CRLRI3	保留	CAPIF3	CAPCH3EN	CFL_IE3	CRL_IE3	INV3
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
CFLRI2	CRLRI2	保留	CAPIF2	CAPCH2EN	CFL_IE2	CRL_IE2	INV2

Bits	描述	
[31:24]	保留	保留
[23]	CFLRI3	**CFLR3锁存标志位** 当 PWM 组输入通道3发生下降沿转变时, CFLR3 锁存PWM向下计数器的值，该位由硬件置位. 如果BCn为0，软件写0清该位, 如果BCn为1，软件写1清该位.
[22]	CRLRI3	**CRLR3锁存标志位** 当 PWM 组输入通道3发生上升沿转变时, CRLR3 锁存PWM向下计数器的值，该位由硬件置位. 如果BCn为0，软件写0清该位, 如果BCn为1，软件写1清该位.
[21]	保留	保留
[20]	CAPIF3	**通道 3捕捉中断标志** 如果PWM组通道3上升锁定中断使能(CRL_IE3=1), PWM组通道3发生上升沿转变时会将CAPIF3 置高; 同样，如果PWM组通道3下降锁定中断使能(CFL_IE3=1)，下降沿会使CAPIF3置高. 写 1 清该位为0
[19]	CAPCH3EN	**通道 3 捕捉功能使能** 1 = 使能PWM组通道3的捕捉功能 0 = 禁用PWM组通道3的捕捉功能 当使能时，捕捉功能锁存PWM计数器的值，并存储到CRLR (上升锁存) 或CFLR (下降锁存). 当禁用时，捕捉器不更新CRLR 或CFLR, 并关闭通道3中断
[18]	CFL_IE3	**通道 3 下降沿锁存中断使能**

		1 = 使能下降沿锁存中断 0 = 禁用下降沿锁存中断 使能时，如果检测到PWM组通道3有下降沿转变，捕捉产生中断.
[17]	**CRL_IE3**	**通道 3 上升沿锁定中断使能** 1 = 使能上升沿锁存中断 0 = 禁用上升沿锁存中断 使能时，如果检测到PWM组通道3有上升沿转变，捕捉产生中断.
[16]	**INV3**	**使能通道 3 反向** 1 =反转打开。来自GPIO引脚的输入信号反转之后再输入到捕捉器 0 =反转关闭
[15:8]	**保留**	保留
[7]	**CFLRI2**	**CFLR2锁存标志位** 当PWM组输入通道0发生下降沿转变时，CFLR2 锁存PWM向下计数值，该位由硬件置位. 如果BCn为0，软件写0清该位, 如果BCn为1，软件写1清该位.
[6]	**CRLRI2**	**CRLR2锁存标志位** 当PWM组输入通道0发生上升沿转变时, CRLR2锁存PWM向下计数值，该位由硬件置位. 如果BCn为0，软件写0清该位, 如果BCn为1，软件写1清该位.
[5]	**保留**	保留
[4]	**CAPIF2**	**通道 2捕捉中断标志** 如果PWM组通道2上升沿锁存中断使能(CRL_IE2=1), PWM组通道2发生上升沿转变时会使CAPIF2 为高; 同样，如果PWM组通道2下降沿锁存中断使能(CFL_IE2=1)，下降沿会使CAPIF2置高. 写 1 清该位为0
[3]	**CAPCH2EN**	**通道2 捕捉功能使能** 1 = 使能PWM组通道2的捕捉功能 0 = 禁用PWM组通道2的捕捉功能 当使能时，捕捉功能锁定PWM计数器，并存储到CRLR (上升锁存) 或CFLR (下降锁存). 当禁用时，捕捉器不更新CRLR 或CFLR, 并关闭通道2中断.
[2]	**CFL_IE2**	**通道 2 下降沿锁存中断使能** 1 = 使能向下锁定中断 0 = 禁用向下锁定中断 使能时，如果检测到PWM组通道2有下降沿转变，捕捉中断产生.
[1]	**CRL_IE2**	**通道 2 上升沿锁存中断使能** 1 = 使能上升沿锁存中断 0 = 禁用上升沿锁存中断

		使能时，如果检测到PWM组通道2有上升沿转变，捕捉中断产生.
[0]	**INV2**	**使能通道 2 反向** 1 =反转打开。来自GPIO引脚的输入信号反转之后再输入到捕捉器 0 =反转关闭

捕捉上升沿锁存寄存器 3-0 (CRLR3-0)

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

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**CRLRx**	**捕捉上升沿锁存寄存器** 通道0/1/2/3 有上升沿转变时，锁存PWM计数器的值.

捕捉下降沿锁存寄存器3-0 (CFLR3-0)

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

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**CFLRx**	**捕捉下降沿锁存寄存器** 通道0/1/2/3 有下降沿转变时，锁存PWM计数器的值.

捕捉输入使能寄存器(CAPENR)

寄存器	偏移量	R/W	描述	复位后的值
CAPENR	PWMA_BA+0x78	R/W	PWM A组捕捉输入0~3 使能寄存器	0x0000_0000

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

Bits	描述	
[3:0]	CAPENR	**捕捉输入使能寄存器** 4组捕捉输入。Bit0~Bit3 用于控制每个输入的打开 / 关闭. 0 = 关闭 (PWMx 多功能引脚输入不会影响输入捕捉功能.) 1 = 打开 (PWMx多功能引脚输入影响输入捕捉功能.) CAPENR Bit 3210 对应于 PWM A组 Bit xxx1 ➔ 捕捉通道0 从PA.12输入 Bit xx1x ➔ 捕捉通道1 从PA.13输入 Bit x1xx ➔ 捕捉通道2 从PA.14输入 Bit 1xxx ➔ 捕捉通道3 从PA.15输入

PWM 输出使能寄存器(POE)

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

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

Bits	描述	
[3]	**PWM3**	**通道 3 输出使能寄存器** 1 = 使能PWM通道3输出 0 = 禁止PWM通道3输出 注: :相应的GPIO管脚必须切到PWM功能
[2]	**PWM2**	**通道 2 输出使能寄存器** 1 = 使能PWM通道2输出 0 = 禁止PWM通道2输出 注: : 相应的GPIO管脚必须切到PWM功能
[1]	**PWM1**	**通道 1 输出使能寄存器** 1 = 使能PWM通道1输出 0 = 禁止PWM通道1输出 注: : 相应的GPIO管脚必须切到PWM功能
[0]	**PWM0**	**通道 0 输出使能寄存器** 1 = 使能PWM通道0输出 0 = 禁止PWM通道0输出 注: : 相应的GPIO管脚必须切到PWM功能

PWM 触发控制寄存器 (TCON)

寄存器	偏移量	R/W	描述	复位后的值
TCON	PWMA_BA+0x80	R/W	PWM A组通道0~3触发控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留				PWM3TEN	PWM2TEN	PWM1TEN	PWM0TEN

Bits	描述	
[3]	**PWM3TEN**	**通道3中心对齐触发使能寄存器** 1 = 使能PWM 通道3触发 ADC 功能. 0 = 关闭PWM 通道 3触发 ADC 功能. 如果这个比特为1，当PWM计数器的值等于CNR时，PWM 能触发ADC开始转换. **注:** 这个功能只有当PWM工作在中心对齐模式时才有效.
[2]	**PWM2TEN**	**通道2中心对齐触发使能寄存器** 1 = 使能PWM 通道2触发 ADC 功能. 0 = 关闭PWM 通道 2触发 ADC 功能. 如果这个比特为1，当PWM计数器的值等于CNR时，PWM 能触发ADC开始转换. **注:** 这个功能只有当PWM工作在中心对齐模式时才有效.
[1]	**PWM1TEN**	**通道1中心对齐触发使能寄存器** 1 = 使能PWM 通道1触发 ADC 功能. 0 = 关闭PWM 通道 1触发 ADC 功能. 如果这个比特为1，当PWM计数器的值等于CNR时，PWM 能触发ADC开始转换. **注:** 这个功能只有当PWM工作在中心对齐模式时才有效
[0]	**PWM0TEN**	**通道0中心对齐触发使能寄存器** 1 = 使能PWM 通道0触发 ADC 功能. 0 = 关闭PWM 通道 0触发 ADC 功能. 如果这个比特为1，当PWM计数器的值等于CNR时，PWM 能触发ADC开始转换. **注:** 这个功能只有当PWM工作在中心对齐模式时才有效mode.

PWM 触发状态寄存器 (TSTATUS)

寄存器	偏移量	R/W	描述	复位后的值
TSTATUS	PWMA_BA+0x84	R/W	PWM A组触发状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留				PWM3TF	PWM2TF	PWM1TF	PWM0TF

Bits	描述	
[3]	PWM3TF	**通道3中心对齐触发标志** 对中心对齐操作模式来说，如果PWM3TEN等于1，当PWM计数器的值上数等于CNR时，这个比特由硬件设为1. 在这个比特被设为1之后，如果ADC选择由PWM触发，ADC将开始转换. 软件写1清除.
[2]	PWM2TF	**通道2中心对齐触发标志** 对中心对齐操作模式来说，如果PWM2TEN等于1，当PWM计数器的值上数等于CNR时，这个比特由硬件设为1. 在这个比特被设为1之后，如果ADC选择由PWM触发，ADC将开始转换. 软件写1清除
[1]	PWM1TF	**通道1中心对齐触发标志** 对中心对齐操作模式来说，如果PWM1TEN等于1，当PWM计数器的值上数等于CNR时，这个比特由硬件设为1. 在这个比特被设为1之后，如果ADC选择由PWM触发，ADC将开始转换. 软件写1清除
[0]	PWM0TF	**通道0中心对齐触发标志** 对中心对齐操作模式来说，如果PWM0TEN等于1，当PWM计数器的值上数等于CNR时，这个比特由硬件设为1. 在这个比特被设为1之后，如果ADC选择由PWM触发，ADC将开始转换. 软件写1清除

PWM0 同步忙状态寄存器 (SYNCBUSY0)

寄存器	偏移量	R/W	描述	复位后的值
SYNCBUSY0	PWMA_BA+0x88	R	PWM0 A 组同步忙状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	**保留**	包留.
[0]	**S_BUSY**	**PWM 同步忙** 因为PWM的时钟可能与系统时钟频率不同，当写CNR0/CMR0/PPR寄存器或者切PWM0操作模式(PCR[3])的时候，PWM需要一段时间来完成更新。为了确保上一个设定已经更新完成，在写CNR0/CMR0/PPR或者切PWM0操作模式(PCR[3])之前，软件需要检查这个忙状态 当软件写CNR0/CMR0/PPR 寄存器或者切换PWM0操作模式(PCR[3])的时候，这个比特将被置位，当PWM更新这些值完成之后，硬件将自动清除这个比特.

PWM1同步忙状态寄存器 (SYNCBUSY1)

寄存器	偏移量	R/W	描述	复位后的值
SYNCBUSY1	PWMA_BA+0x8C	R	PWM1 A组同步忙状态寄存器	0x0000_0000

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

Bits	Description	
[31:1]	保留	保留.
[0]	S_BUSY	**PWM 同步忙** 因为PWM的时钟可能与系统时钟频率不同，当写CNR1/CMR1/PPR寄存器或者切换PWM1操作模式(PCR[11])的时候，PWM需要一段时间来完成更新。为了确保上一个设定已经更新完成，在写CNR1/CMR1/PPR或者切PWM1操作模式(PCR[11])之前，软件需要检查这个忙状态 当软件写CNR1/CMR1/PPR 寄存器或者切换PWM1操作模式(PCR[11])的时候，这个比特将被置位，当PWM更新这些值完成之后，硬件将自动清除这个比特.

PWM2同步忙状态寄存器(SYNCBUSY2)

寄存器	偏移量	R/W	描述	复位后的值
SYNCBUSY2	PWMA_BA+0x90	R	PWM2 A组同步忙状态寄存器	0x0000_0000

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

Bits	Description	
[31:1]	保留	保留.
[0]	S_BUSY	**PWM 同步忙** 因为PWM的时钟可能与系统时钟频率不同，当写CNR2/CMR2/PPR寄存器或者切换PWM2操作模式(PCR[19])的时候，PWM需要一段时间来完成更新。为了确保上一个设定已经更新完成，在写CNR2/CMR2/PPR或者切PWM2操作模式(PCR[19])之前，软件需要检查这个忙状态 当软件写CNR2/CMR2/PPR 寄存器或者切换PWM2操作模式(PCR[19])的时候，这个比特将被置位，当PWM更新这些值完成之后，硬件将自动清除这个比特.

PWM3同步忙状态寄存器(SYNCBUSY3)

寄存器	偏移量	R/W	描述	复位后的值
SYNCBUSY3	PWMA_BA+0x94	R	PWM3 A组同步忙状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	保留	保留.
[0]	S_BUSY	**PWM 同步忙** 因为PWM的时钟可能与系统时钟频率不同，当写CNR3/CMR3/PPR寄存器或者切换PWM3操作模式(PCR[27])的时候，PWM需要一段时间来完成更新。为了确保上一个设定已经更新完成，在写CNR3/CMR3/PPR或者切PWM3操作模式(PCR[27])之前，软件需要检查这个忙状态 当软件写CNR3/CMR3/PPR 寄存器或者切换PWM3操作模式(PCR[27])的时候，这个比特将被置位，当PWM更新这些值完成之后，硬件将自动清除这个比特.

PWM PDMA 控制寄存器 (CAPPDMACTL)

寄存器	偏移量	R/W	描述	复位后的值
CAPPDMACTL	PWMA_BA+0xC0	R/W	PWM PDMA控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留				CAP3RFORDER	CAP3PDMAMOD		CAP3PDMAEN
23	22	21	20	19	18	17	16
保留				CAP2RFORDER	CAP2PDMAMOD		CAP2PDMAEN
15	14	13	12	11	10	9	8
保留				CAP1RFORDER	CAP1PDMAMOD		CAP1PDMAEN
7	6	5	4	3	2	1	0
保留				CAP0RFORDER	CAP0PDMAMOD		CAP0PDMAEN

Bits	描述	
[27]	CAP3RFORDER	**捕获通道 3 上升沿/下降沿存放次序** **当CAP3PDMAMOD =11时，这个比特决定PDMA先传输CRLR3还是CFLR3捕获的数据到内存** 1 = CRLR3 先被传输到内存. 0 = CFLR3 先被传输到内存.
[26:25]	CAP3PDMAMOD	**选择PDMA传输 CRLR3 还是 CFLR3** 00 = 保留 01 = CRLR3 10 = CFLR3 11 = CRLR3 和 CFLR3都被传输
[24]	CAP3PDMAEN	**通道 3 PDMA 使能** 1 = 使能通道3的PDMA功能，当通道3捕获到数据时，由PDMA传输到内存. 0 = 关闭通道3的 PDMA功能
[19]	CAP2RFORDER	**捕获通道 2 上升沿/下降沿存放次序** **当CAP3PDMAMOD =11时，这个比特决定PDMA先传输CRLR3还是CFLR3捕获的数据到内存** 1 = CRLR3 先被传输到内存. 0 = CFLR3 先被传输到内存.

[18:17]	**CAP2PDMAMOD**	**选择PDMA传输 CRLR3 还是 CFLR3** 00 = 保留 01 = CRLR3 10 = CFLR3 11 = CRLR3 和 CFLR3都被传输
[16]	**CAP2PDMAEN**	**通道 3 PDMA 使能** 1 = 使能通道3的PDMA功能，当通道3捕获到数据时，由PDMA传输到内存. 0 = 关闭通道3的 PDMA功能
[11]	**CAP1RFORDER**	**捕获通道 3 上升沿/下降沿存放次序** **当CAP3PDMAMOD =11时，这个比特决定PDMA先传输CRLR3还是CFLR3捕获的数据到内存** 1 = CRLR3 先被传输到内存. 0 = CFLR3 先被传输到内存.
[10:9]	**CAP1PDMAMOD**	**选择PDMA传输 CRLR3 还是 CFLR3** 00 = 保留 01 = CRLR3 10 = CFLR3 11 = CRLR3 和 CFLR3都被传输
[8]	**CAP1PDMAEN**	**通道 3 PDMA 使能** 1 = 使能通道3的PDMA功能，当通道3捕获到数据时，由PDMA传输到内存. 0 = 关闭通道3的 PDMA功能
[3]	**CAP0RFORDER**	**捕获通道 3 上升沿/下降沿存放次序** **当CAP3PDMAMOD =11时，这个比特决定PDMA先传输CRLR3还是CFLR3捕获的数据到内存** 1 = CRLR3 先被传输到内存. 0 = CFLR3 先被传输到内存.
[2:1]	**CAP0PDMAMOD**	**选择PDMA传输 CRLR3 还是 CFLR3** 00 = 保留 01 = CRLR3 10 = CFLR3 11 = CRLR3 和 CFLR3都被传输
[0]	**CAP0PDMAEN**	**通道 3 PDMA 使能** 1 = 使能通道3的PDMA功能，当通道3捕获到数据时，由PDMA传输到内存. 0 = 关闭通道3的 PDMA功能

PWM PDMA 数据寄存器0 (CAP0PDMA)

寄存器	偏移量	R/W	描述	复位后的值
CAP0PDMA	PWMA_BA+0xC4	R	PWM A 组PDMA 通道 0 数据寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
CAP0RFPDMA							
7	6	5	4	3	2	1	0
CAP0RFPDMA							

Bits	描述	
[15:0]	CAP0RFPDMA[15:0]	**捕获通道0 PDMA 数据寄存器** 通道0捕获到的值(CFLR0/CRLR0)

PWM PDMA 数据寄存器 1 (CAP1PDMA)

寄存器	偏移量	R/W	描述	复位后的值
CAP1PDMA	PWMA_BA+0xC8	R	PWM A组PDMA 通道 1 数据寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
CAP1RFPDMA							
7	6	5	4	3	2	1	0
CAP1RFPDMA							

Bits	描述	
[15:0]	CAP1RFPDMA[15:0]	**捕获通道1 PDMA 数据寄存器** 通道1捕获到的值(CFLR1/CRLR1)

PWM PDMA 数据寄存器 2 (CAP2PDMA)

寄存器	偏移量	R/W	描述	复位后的值
CAP2PDMA	PWMA_BA+0xCC	R	PWM A组PDMA 通道 2 数据寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
CAP2RFPDMA							
7	6	5	4	3	2	1	0
CAP2RFPDMA							

Bits	描述	
[15:0]	CAP2RFPDMA[15:0]	**捕获通道2 PDMA 数据寄存器** 通道2捕获到的值(CFLR2/CRLR2)

PWM PDMA 数据寄存器 3 (CAP2PDMA)

寄存器	偏移量	R/W	描述	复位后的值
CAP3PDMA	PWMA_BA+0xD0	R	PWM A组PDMA 通道 3 数据寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
CAP3RFPDMA							
7	6	5	4	3	2	1	0
CAP3RFPDMA							

Bits	描述	
[15:0]	CAP3RFPDMA[15:0]	**捕获通道3 PDMA 数据寄存器** 通道3捕获到的值(CFLR3/CRLR3)

5.10 串行外围设备接口 (SPI)

5.10.1 概述

SPI 接口是工作于全双工模式下的同步串行数据传输接口。支持主/从模式，4线，双向传输。最多包含3组SPI控制器，将从外设得到的数据进行串并转换，或将数据进行并串转换，发送到外设。每组SPI控制器可以作为一个主机，也可以被设置为从机，由片外主机控制。

该控制器支持可变串行时钟以适应特殊应用，也可以支持2位传输模式，可同时连接两个片外从机设备，SPI控制器也支持PDMA功能访问数据缓冲，也支持双IO传输模式.

5.10.2 特性

- 最多支持3组SPI控制器
- 支持主/从机模式
- 支持1位或2位传输模式
- 支持双IO传输模式
- 每次传输长度可配置8 ～32位
- 接收和发送有单独的8层FIFO缓冲
- 支持MSB 或 LSB 优先传输
- 主机模式下有2条从机选择线
- 支持字节重排序
- 支持字节或字休眠模式
- 主机模式下，串行输出时钟频率可变
- 支持PDMA传输
- 从机模式下，支持3线模式，没有从机片选，双向接口

5.10.3 框图

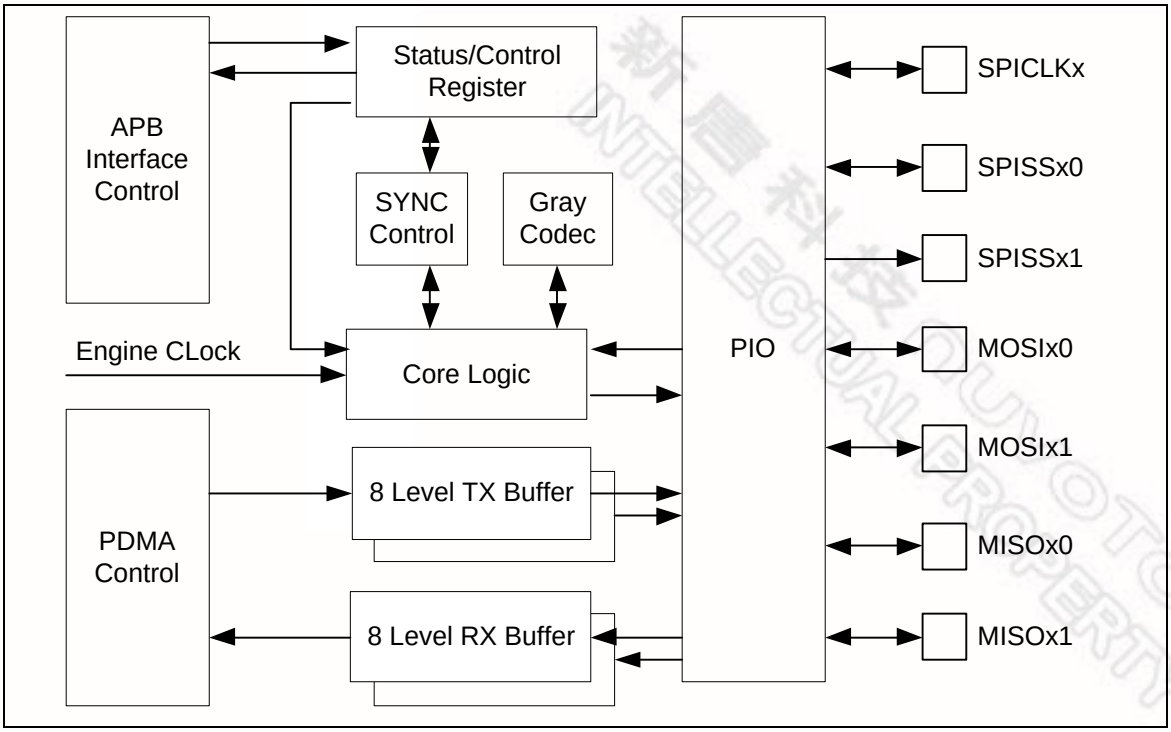


图 5-57 SPI 框图

5.10.4 功能描述

SPI 引擎时钟和串行时钟

SPI 控制器需要SPI引擎时钟驱动SPI逻辑电路实现数据传输。SPI 引擎时钟频率由时钟源选择寄存器BCn和时钟除频器决定。CLKSEL1寄存器的SPIx_S 决定SPI引擎的时钟源。时钟源可以选择HCLK或者PLL输出时钟。设置SPI_CNTRL2 寄存器的BCn为0，SPI时钟频率的计算公式和以前产品兼容。SPI_DIVIDER寄存器的DIVIDER为时钟频率算式的除数。

主模式下，如果可变时钟频率功能关闭，串行时钟的输出脚的输出频率等于SPI引擎的时钟频率。从模式下，SPI串行时钟由片外主机提供。从设备的SPI 引擎时钟频率必须比连接的主设备的串行时钟频率快。无论主还是从模式下，SPI引擎时钟的频率不能比APB时钟频率快。

主/从模式

SPI控制器可通过设置**SLAVE** 位(SPI_CNTRL[18])，配置为主机或从机模式，与片外从机或主机通信。主机模式与从机模式的应用框图如下.

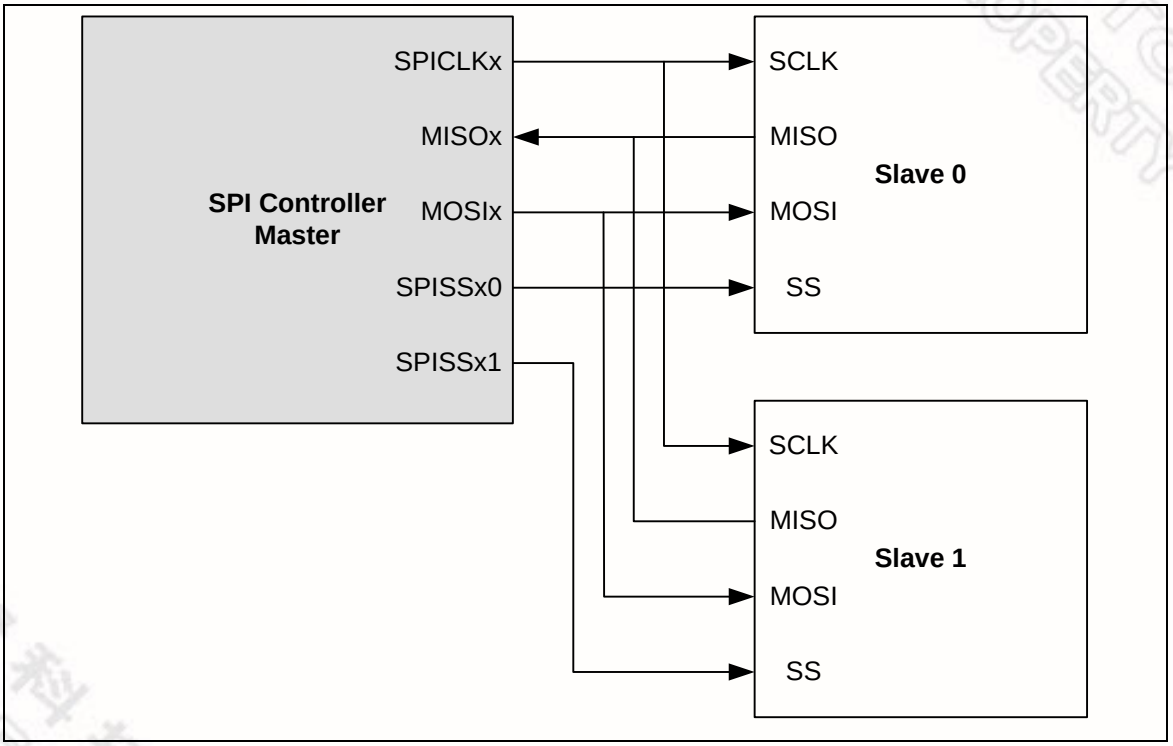


图 5-58 SPI 主机模式应用框图

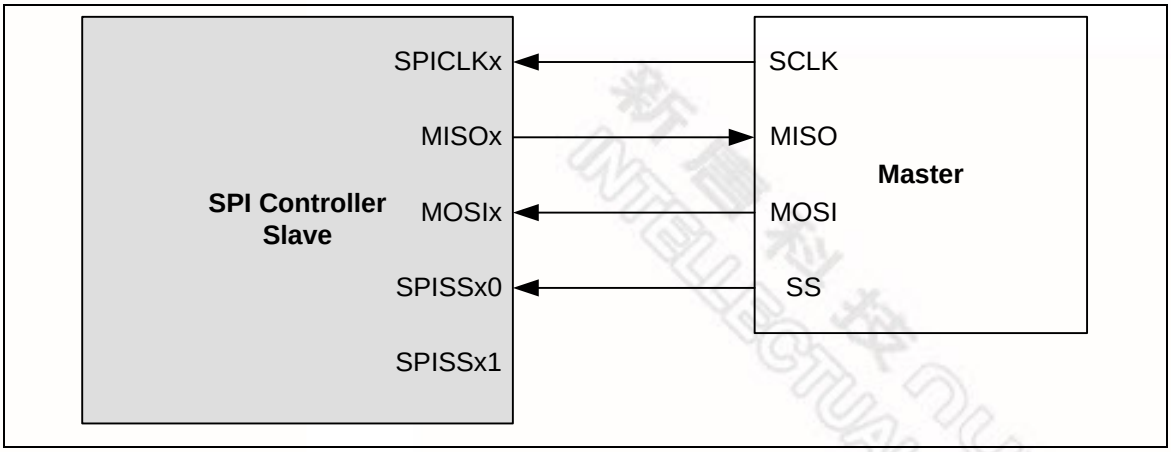


图 5-59 SPI 从机模式应用框图

从机选择

在主机模式下，SPI控制器通过片选输出脚SPISSx0 与 SPISSx1最多能驱动两个片外从机设备。从机模式下，片外的主机设备使用SPISSx0 输入引脚作为片选连接到SPI控制器。在主机/从机模式下，片选信号的有效电平可以通过**SS_LVL** 位 (SPI_SSR[2])被编程为低有效或高有效，通过**SS_LTRIG** 位(SPI_SSR[4])定义片选信号SPISSx0/1 为电平触发或边沿触发。触发条件的选择取决于所连接的片外从机/主机的类型。

从机模式下，如果SS_LTRIG被配置成电平触发，LTRIG_FLAG 位 (SPI_SSR[5]) 用来表示一次传输完成后接收到的位数是否等于TX_BIT_LEN 的设定。（一次传输完成意味着片选信号已无效或SPI 控制器已经完成了一次数据传输。）

电平触发/边沿触发

从机模式下，片选信号可配置成电平触发或边沿触发。当为边沿触发时，数据从检测到有效边沿时刻开始传输，在检测到无效边沿信号时结束。 如果主机没有发送无效边沿信号给从机，传输过程就不会结束，从机的中断标志位也不会置位。当为电平触发时，有两个条件可以中止传输，并置位中断标志位。 第一个条件是如果传输的比特数与TX_BIT_LEN的设置相等，从机中断标志将置位；第二个条件是如果主机设置片选信号为无效电平，无论已经收到多少数据，从机都将中止当前传输，并置位中断标志位。用户可以读LTRIG_FLAG检查是否数据被完全传输..

自动从机选择

在主机模式下，如果置位AUTOSS (SPI_SSR[3])，片选信号将根据**SSR[0]** (SPI_SSR[0]) 与**SSR[1]** (SPI_SSR[1])是否使能而自动产生，并从SPISSx0与SPISSx1引脚输出。这就意味着，设置**GO_BUSY** 位 (SPI_CNTRL[0])开始发送/接收时，被**SSR[1:0]**选择的从机选择信号引脚将由SPI控制器设置为有效状态，而当数据传输结束时，SPI控制器会自动将从机选择信号引脚设置为无效状态。当**ASS**位清零时，可以通过软件自行设置或清零SPI_SSR[1:0]的相关位来决定片选引脚输出为有效状态或无效状态。片选信号的有效状态由**SS_LVL** 位 (SPI_SSR[2])定义。

可变时钟功能

主机模式下，如果使能VARCLK_EN (SPI_CNTRL[23])，串行时钟的输出频率可随着

VARCLK(SPI_VARCLK[31:0])寄存器的配置而动态变化。每个串行时钟的周期都依靠SPI_VARCLK寄存器的设定。当可变时钟功能使能时，TX_BIT_LEN必须设为0x10，将数据传输配置为16-bit传输模式。VARCLK[31]决定第一个时钟的时钟周期。如果VARCLK[31]等于0，第一个时钟周期取决于DIVIDER (SPI_DIVIDER[15:0]) 的设定，如果VARCLK[31] 的值为'1'，第一个时钟周期取决于DIVIDER2 (SPI_DIVIDER[31:16]) 的设定。VARCLK[30:1]每两个连续比特决定一个时钟周期。VARCLK[30:29]定义第二个时钟周期，VARCLK[28:27]定义第三个时钟周期，以此类推。VARCLK[0]没有意义。下图为串行时钟 (SPICLK), VARCLK, DIVIDER 和DIVIDER2之间的时序关系。 注意当使能VARCLK_EN 位时，TX_BIT_LEN 必须设置成0x10 (仅16位模式).

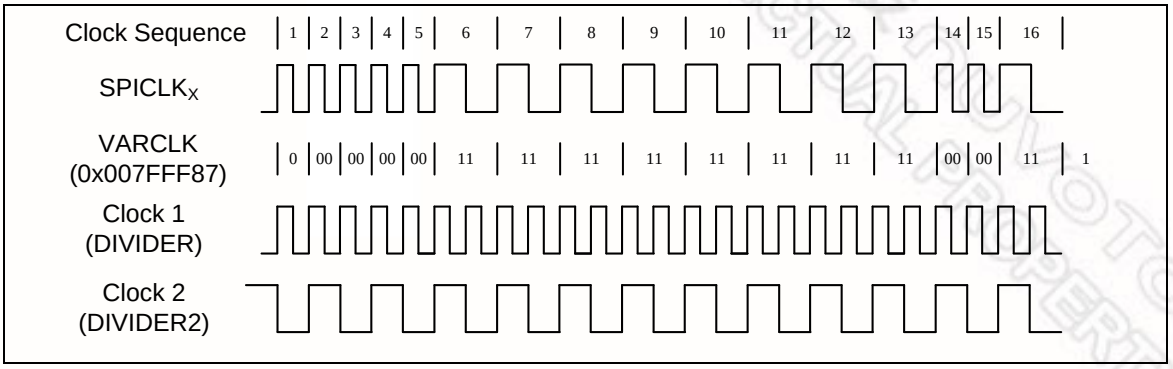


图 5-60 可变串行时钟频率

时钟极性

CLKP位 (SPI_CNTRL[11])定义串行时钟的空闲状态. 当 **CLKP** = 1, 空闲时SPICLK输出为高电平状态，否则**CLKP** = 0时， SPICLK输出为低电平状态.

发送/接收位长度

传输字的长度由**TX_BIT_LEN** 位(SPI_CNTRL[7:3])定义。发送和接收时传输字长度最大可以配置为32位长.

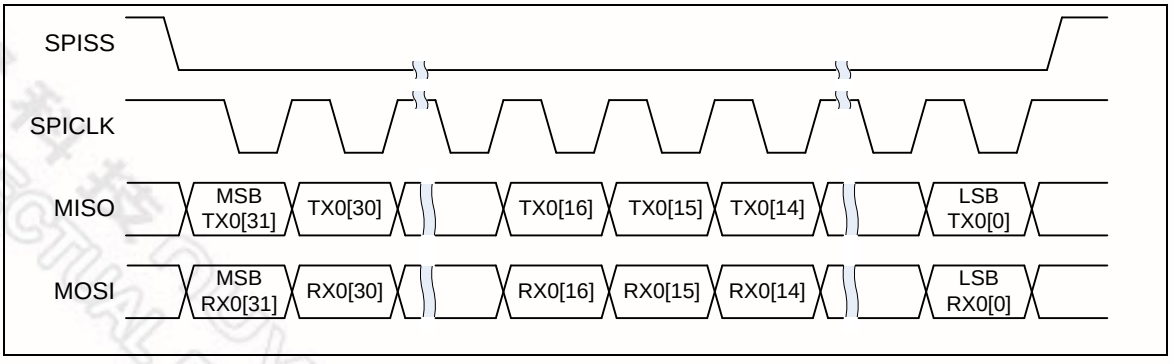


图 5-61 一次传输 32-位

LSB/MSB 优先

LSB位(SPI_CNTRL[10]) 决定是从LSB还是从MSB开始发送/接收数据。LSB等于1表示LSB优先发

送，比特0被第一个发送。如果LSB等于0，表示MSB优先发送

发送沿

TX_NEG 位 (SPI_CNTRL[2]) 定义数据发送是在串行时钟SPICLK的下降沿还是上升沿.

接收沿

Rx_NEG 位 (SPI_CNTRL[1]) 定义数据接收是在串行时钟SPICLK的下降沿还是上升沿。

注: TX_NEG与RX_NEG的设定为互斥的。换句话说就是发送和接收不可设为相同的时钟沿。

字休眠

在主机模式下，SP_CYCLE (SPI_CNTRL[15:12])可配置两个连续传输字之间的休眠间隔为0.5 ~ 15.5个串行时钟周期。休眠间隔定义为从前一次传输字的最后一个时钟沿到下一次传输字的第一个时钟沿。SP_CYCLE的缺省值为0x3 (3.5 个串行时钟周期)。如果软件关闭FIFO模式，SP_CYCLE的设定将不起作用。

主模式下，如果VARCLK_EN, SPI_CNTRL[23], 和 FIFO 位, SPI_CNTRL[21], 都被设为1，最小的字休眠周期为(6.5 + SP_CYCLE)*SPI时钟周期

字节重排序

当设置为MSB优先时（**LSB** = 0），且使能REORDER，在32 位模式时(TX_BIT_LEN = 0)，存储在TX 缓存与RX 缓存的数据将重新按[BYTE0, BYTE1, BYTE2, BYTE3] 的顺序排列, 数据将以BYTE0, BYTE1, BYTE2然后BYTE3的顺序发送/接收。在**TX_BIT_LEN** = 24 位模式时，存储在TX 缓存与RX 缓存的数据将重新按[unknown byte，BYTE0, BYTE1, BYTE2] 的顺序排列, 数据将以BYTE0, BYTE1, BYTE2的顺序发送/接收。每个字节发送/接收都按照MSB优先。16位模式与上面相同。字节重排序功能只有在TX_BIT_LEN配置为16,24,32比特时才有效。

注意：当可变时钟功能使能时，不支持字节重排序功能。

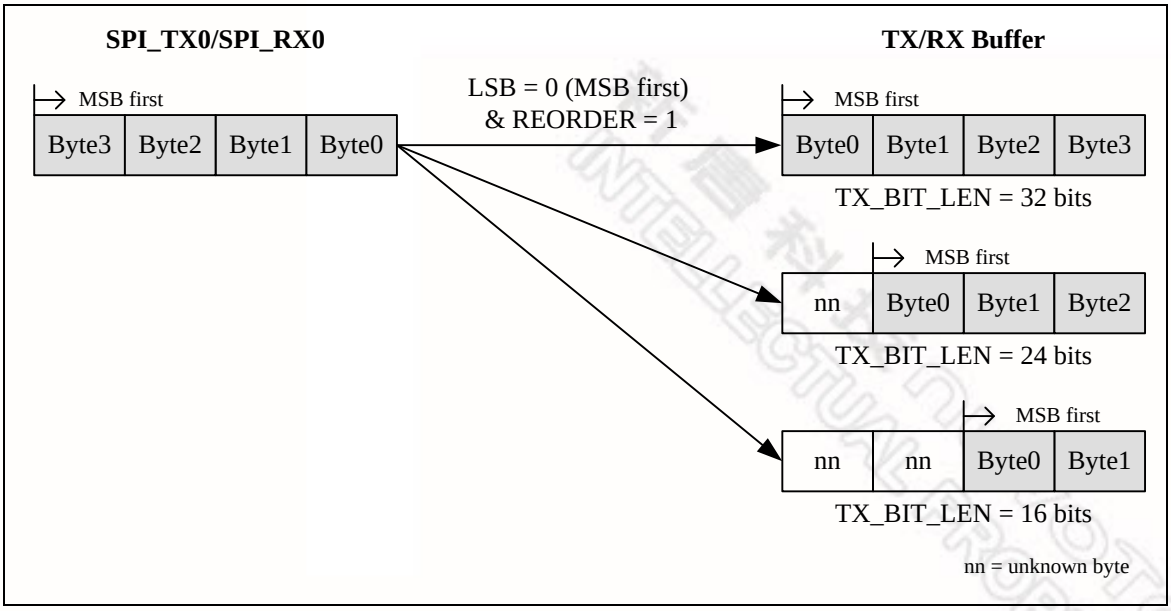


图 5-62 字节重排序功能

字节休眠

主机模式下，如果(SPI_CNTRL[19])为1, 硬件将在两个连续传输字节之间插入0.5 ~ 15.5个串行时钟周期的休眠间隔。字节休眠的设置与字休眠的设置一样都是用SP_CYCLE.

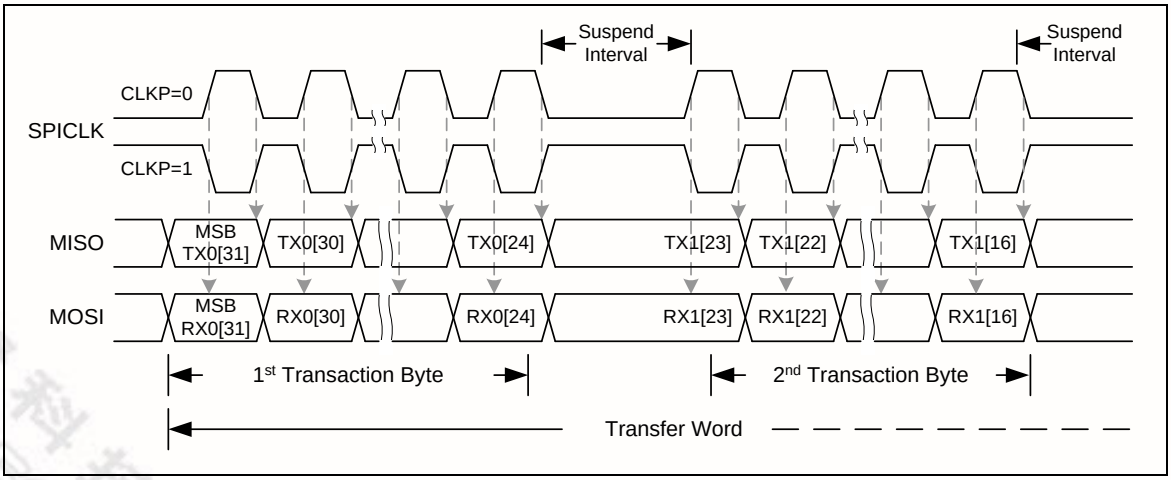


图 5-63 字节休眠时序波形

从3线模式

软件通过设定NOSLVSEL可以使能从3线模式。从模式下，忽略片选信号，SPI控制器可以工作在无片选模式(3线模式)，NOSLVSEL比特只有在从模式下才起作用。接口包括SPICLK, SPI_MISO, 和SPI_MOSI 3根线。SPISS可以配置为GPIO。当NOSLVSEL 等于1时，SPI从设备在GO_BUSY设为1时准备接收/发送数据。无片选模式下，SS_LTRIG, SPI_SSR[4], 都应该设为1.

两位传输模式

当使能TWOB位(SPI_CNTRL[22])时，SPI控制器支持两位(two-bit)传输模式. 此时可以同时发送和接收两位串行数据.

例如，在主机模式下，存储在SPI_TX0寄存器和SPI_TX1寄存器中的数据分别通过MOSIx0和MOSIx1引脚发送；同时，SPI_RX0寄存器和SPI_RX1寄存器分别通过MISOx0和MISOx1引脚接收数据..

在从机模式下，存储在SPI_TX0寄存器和SPI_TX1寄存器中的数据分别通过MISOx0和MISOx1引脚发送；同时，SPI_RX0寄存器和SPI_RX1寄存器分别通过MOSIx0和MOSIx1引脚接收数据.

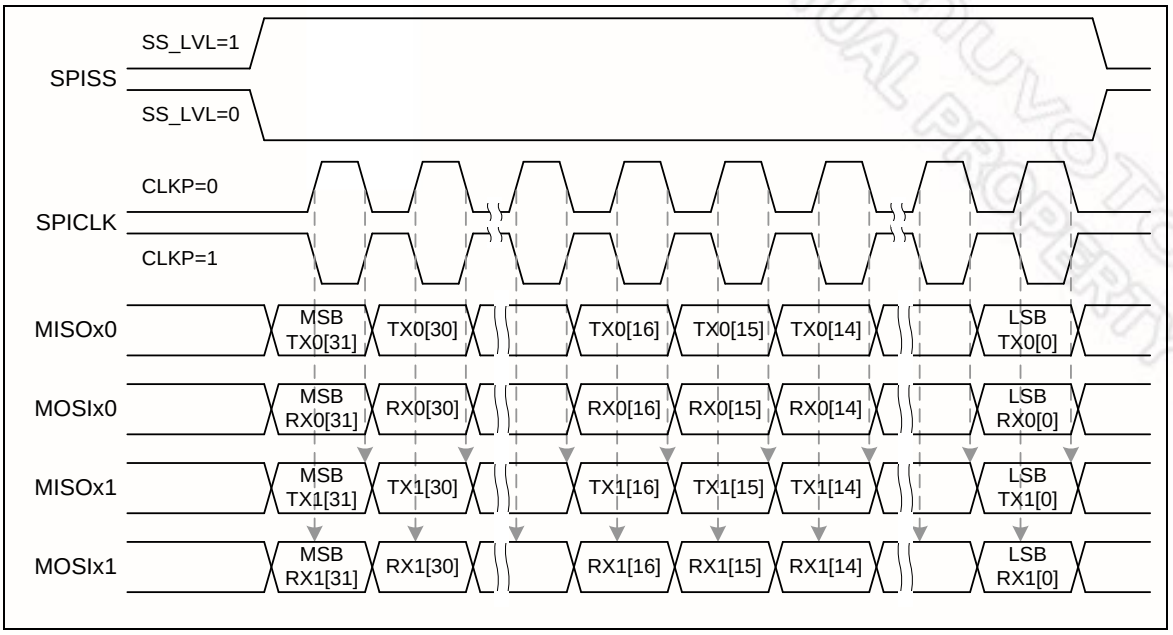


图 5-64 两位传输模式 (从机模式)

双 IO 模式

当设置DUAL_IO_EN 位 (SPI_CNTRL2[13])为1时，SPI控制器也支持双IO传输。许多SPI Flash支持双IO传输。DUAL_IO_DIR (SPI_CNTRL2[12]) 用来定义数据传输的方向。当DUAL_IO_DIR 等于1时，控制器发送数据到外设；当DUAL_IO_DIR 等于0时，控制器从外设读数据。此功能支持8, 16, 24, 和 32位比特长.

3线模式和字节重排序功能使能时，不支持双IO模式.

如果 DUAL_IO_EN 和 DUAL_IO_DIR 都设为1，MISO0 和MOSI0 都作为数据输出端口，MOSI0 输出偶数比特，MISO0输出奇数比特。如果DUAL_IO_EN 等于1，DUAL_IO_DIR 等于0时，MISO0 和MOSI0 都作为数据输入端口，MOSI0 输入偶数比特，MISO0输入奇数比特。

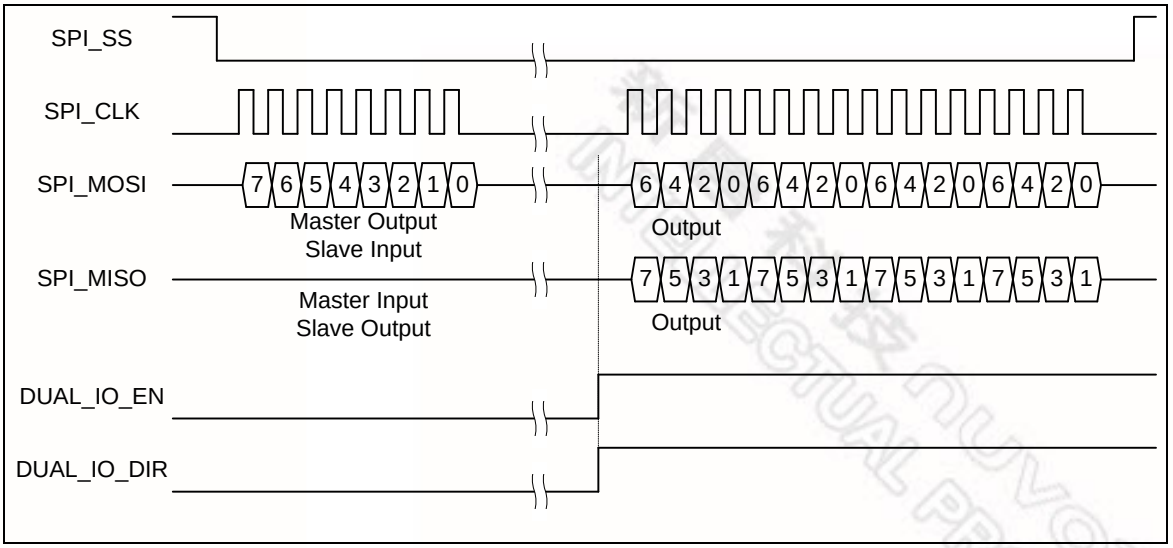


图 5-65 双 IO 输出顺序

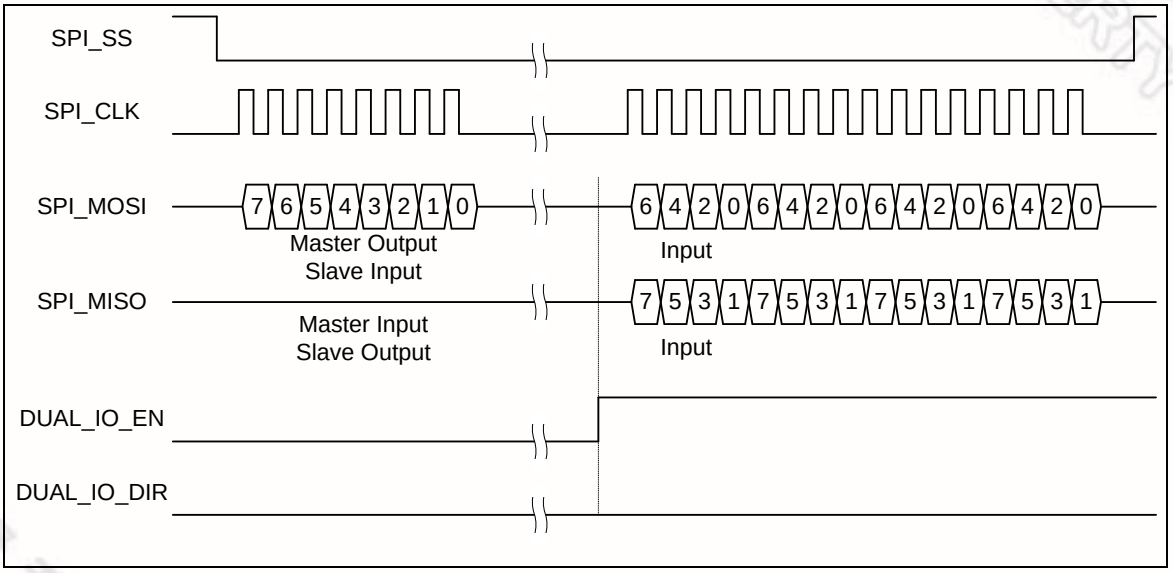


图 5-66 双 IO 输入顺序

FIFO 模式

当SPI_CNTRL[21]寄存器的FIFO 比特为1时，SPI控制器支持FIFO模式。接收和发送FIFO是独立的，都是32bit宽，共8个.

发送FIFO有8个字，先进先出。软件通过写SPI_TX0寄存器可以将数据写到发送FIFO缓冲中。存放在FIFO缓冲中数据，将由发送逻辑读出并发送出去。如果8层FIFO缓冲满，TX_FULL比特(SPIFIFO_STS[27])将被置。如果8层FIFO为空，TXFIFO_EMPTY (SPIFIFO_STS[26])比特将被置。注意，当最后的传输还在进行时，TX_EMPTY标志就会被置为1。主模式下，软件应该检查GO_BUSY位和TX_EMPTY位确保SPI已经在空闲状态。

接收FIFO 也为8个字，先进先出。接收逻辑将数据存到FIFO之后，用户可以通过SPI_RX0寄存器将数据读走。 当前FIFO的状态由RX_EMPTY 和 RX_FULL指示。

FIFO 模式下，发送和接收阈值可以通过TX_THRESHOLD, RX_THRESHOLD来设定。当存储在发送FIFO中的有效数据的数量小于等于TX_THRESHOLD时，TX_INTSTS将被设为1。当存储在接收FIFO中的有效数据的数量大于RX_THRESHOLD时，RX_INTSTS将被设为1。

FIFO模式下，软件可以提前写8个数据到发送FIFO缓冲中。当SPI控制器工作在FIFO模式时，SPI_CNTRL 控制器的GO_BUSY位是只读的，由硬件控制。除非关闭FIFO功能，否则软件不可以修改SPI_CNTRL寄存器的任何内容。

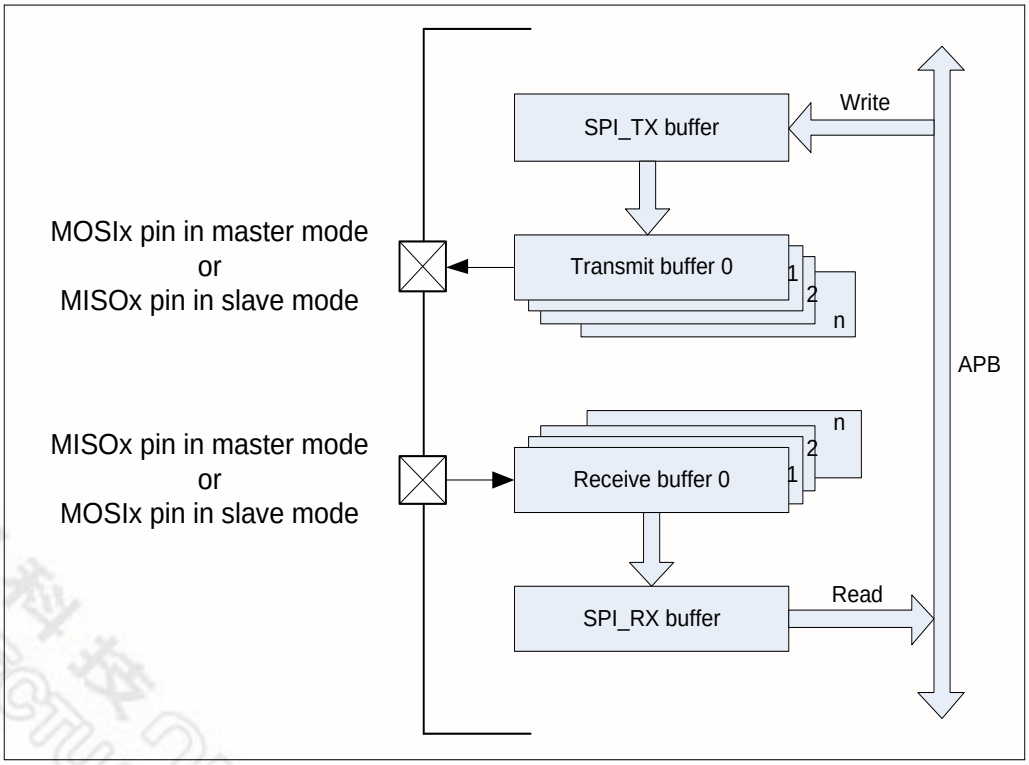


图 5-67 FIFO 模式方块图

主模式传输时，当FIFO比特等于1并且第一笔数据写到SPI_TX0以后，TXFIFO_EMPTY标志将被清成0。只要发送FIFO不为空，传输马上就会开始。用户可以立即将下一个数据写到SPI_TX0中。FIFO模式下，SPI控制器将在连续的两笔传输之间插入休眠间隔，休眠间隔周期由SP_CYCLE (SPI_CNTRL [15:12])的设定决定。无论何时只要TX_FULL为0，用户就可以写数据到SPI_TX0中。

如果下一笔数据及时到达，传输将自动继续进行。如果所有数据都被传输完毕，SPI_TX0仍没有更新，传输将停止。

主模式接收时，串行数据从MISOx引脚接收回来，存放在接收FIFO中。当接收FIFO中包含有未读走的数据时，RX_EMPTY 标志将被清成0。只要RX_EMPTY标志等于0，软件就可以由SPI_RX0寄存器将数据读走。当接收FIFO中有8个数据的时候，RX_FULL标志将被置。此时，SPI控制器将停止接收数据，直到软件读SPI_RX0寄存器之后，接收才将继续。

从模式下，当FIFO设为1时，GO_BUSY比特由硬件自动设为1，并且是只读的。如果用户想停止从模式发送，FIFO比特和GO_BUSY比特必须都由软件清为0.

从模式发送时，当软件写数据到SPI_TX0时，数据将被加载到发送FIFO，TX_EMPTY标志被自动清成0。从设备收到主机的串行时钟信号时自动开始发送。不论何时只要TX_FULL等于0，用户就可以写数据到SPI_TX0寄存器。FIFO中的所有数据都发送完毕后，如果软件没有写数据到SPI_TX0寄存器，TX_EMPTY标志将被设为1。

从模式接收时，串行数据从MOSIx引脚接收回来并存储在接收FIFO中。接收机制和主模式下的相似.

中断

- SPI 传输中断

当SPI控制器结束一次传输时，传输中断标志IF (SPI_CNTRL[16]) 将被设为1。如果传输中断使能IE (SPI_CNTRL[17])，将发生传输中断通知CPU一次传输结束。传输中断标志写1清0.

- SPI 3 线模式开始中断

3线模式下，检测到SPI串行时钟信号时，3线模式开始中断标志，SLV_START_INTSTS, 将被设为1。如果SSTA_INTEN 等于1，SPI控制器将发出中断。如果收到的比特数小于TX_BIT_LEN 的值并且没有串行时钟输入超过用户定义的时间，用户可以设置SLV_ABORT 比特来终止当前传输，传输中断标志, IF, 将被置为1。

- 接收 FIFO 超时中断

FIFO 模式下，有超时功能通知用户。如果在接收FIFO中有数据，主模式下，软件超过64个SPI引擎时钟周期，从模式下软件超过576个SPI引擎时钟周期，还没有读走 ，如果超时中断使能(FIFO_CTL[21]等于 1)，将发生超时中断。

- 发送 FIFO 中断

FIFO 模式下，如果发送FIFO中的数据个数小于或者等于TX_THRESHOLD的设定值，发送FIFO中断标志将被置为1。如果发送FIFO中断使能位，SPI_FIFO_CTL[3]，等于1，SPI 控制器将发生中断通知CPU。

- 接收 FIFO 中断

FIFO 模式下，如果接收FIFO中的有效数据个数大于RX_THRESHOLD的设定值，接收FIFO中断标志将被置为1。如果接收FIFO中断使能位，SPI_FIFO_CTL[2]，等于1，SPI控制器将发生中断通知CPU。

5.10.5 时序图

片选信号(SPISSx) 的有效状态可以由**SS_LVL** 位 (SPI_SSR[2])及**SS_LTRIG** 位 (SPI_SSR[4])配置。串行时钟 (SPICLK)的空闲状态可以通过**CLKP** 位 (SPI_CNTRL[11])配置为高电平或低电平。在**TX_BIT_LEN** (SPI_CNTRL[7:3])中定义传输字的长度，在**LSB** (SPI_CNTRL[10])中定义发送/接收数据是以MSB或LSB优先. **TX_NEG/RX_NEG**(SPI_CNTRL[2:1])可以选择发送/接收数据时串行时钟的边沿。 四个SPI时序图及相关设置如下.

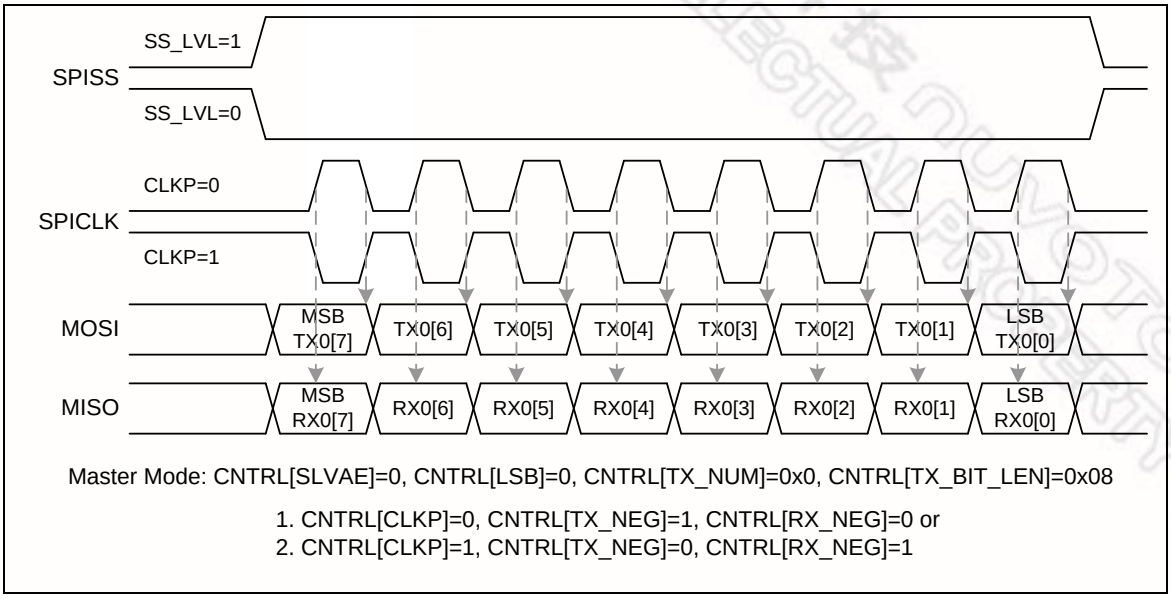


图 5-68 SPI 主机模式下的时序

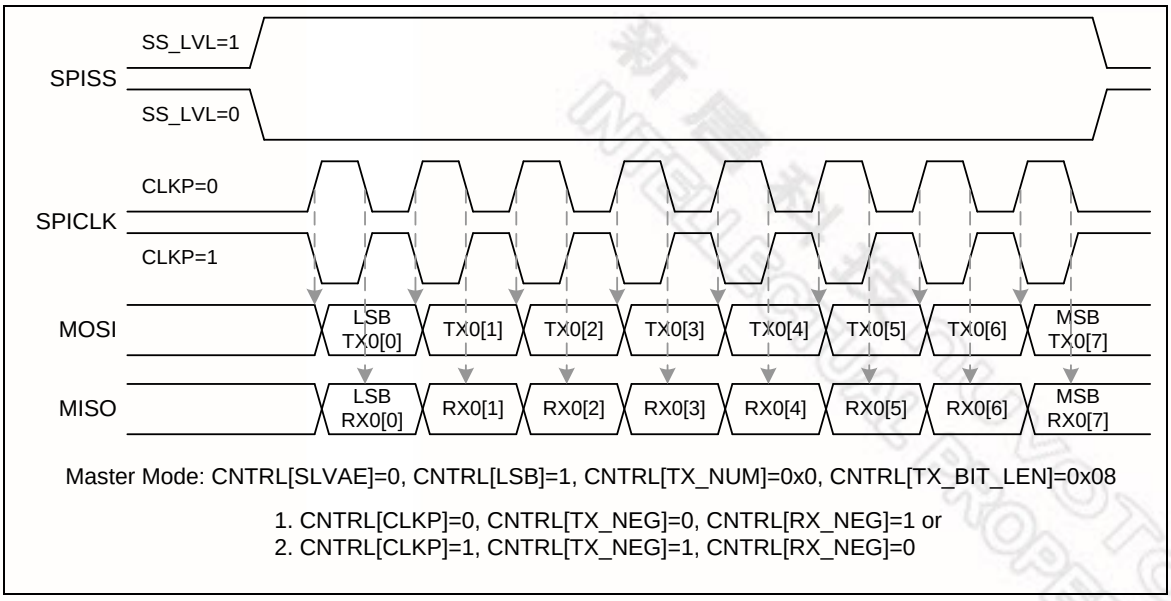


图 5-69 SPI 主机模下的时序 (Alternate Phase of SPICLK)

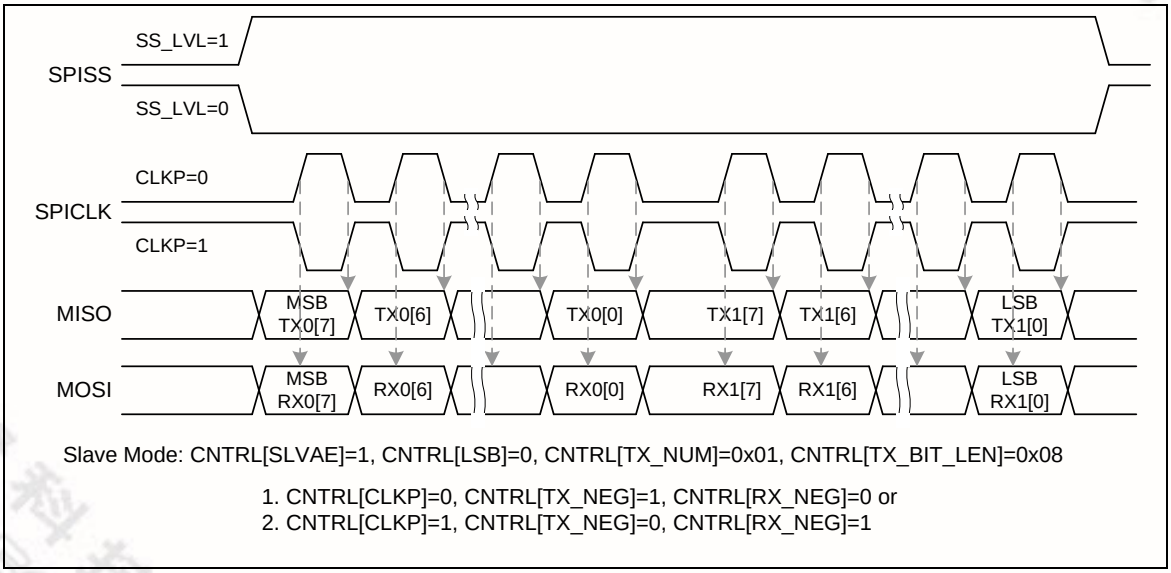


图 5-70 SPI 从机模式下的时序

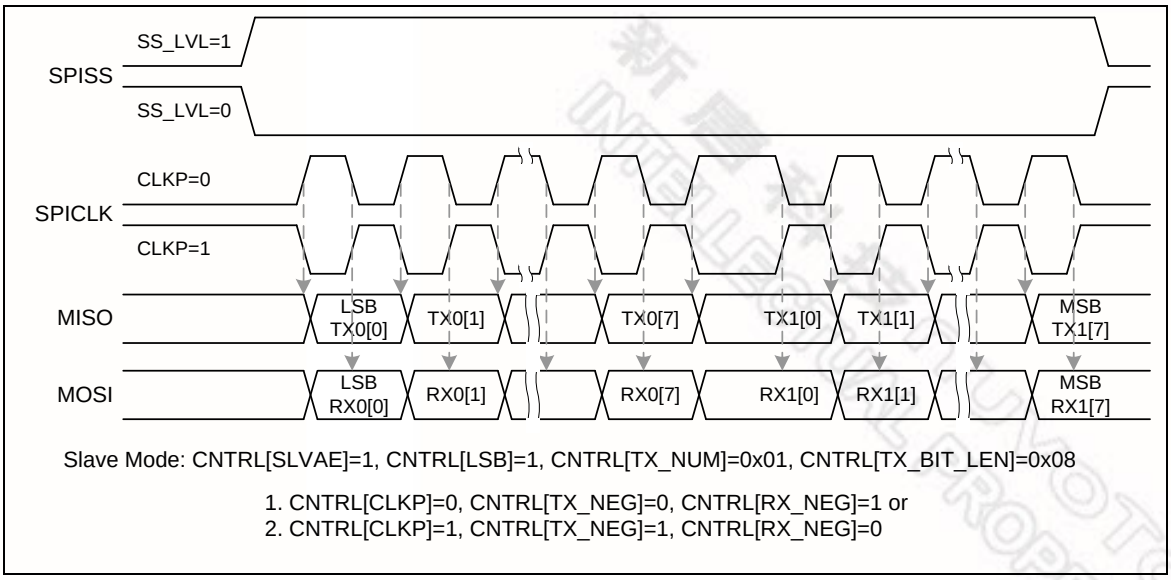


图 5-71 SPI 从机模式下的时序 (Alternate Phase of SPICLK)

5.10.6 编程例程

例 1, SPI作为主机，与 一个从机设备通信，从机规格如下:

- 数据在时钟上升沿锁存
- 数据在时钟下将沿传输
- MSB先传送
- SPICLK 空闲时为低电平状态
- 每次输出/接收一个字节
- 使用第一个 SPI 片选引脚和一个片外从机相连。片选信号的有效状态为低电平

操作流程如下:

1) 设定寄存器DIVIDER 的SPI_DIVIDER[7:0]来决定串行时钟输出频率.
2) 将主机模式的相应设置写入SPI_SSR寄存器,
    1. 本例中禁止自动片选**ASS** (SPI_SSR[3] = 0),
    2. 设置**SS_LVL** (SPI_SSR[2] = 0)和SS_LTRIG (SPI_SSR[4] = 1)，使片选信号为低电平触发输出.
    3. 通过设置片选寄存器**SSR[0]** (SPI_SSR[0])使片选信号有效从而激活片外从机设备
3) 将主机模式的相应设置写入寄存器SPI_CNTRL.
        1. 通过**SLAVE** 位 (SPI_CNTRL[18] = 0)设置SPI控制器为主机设备
        2. 通过**CLKP** 位 (SPI_CNTRL[11] = 0) 设置串行时钟空闲状态为低电平
        3. 通过**TX_NEG** 位 (SPI_CNTRL[2] = 1)选择数据在串行时钟的下降沿传输
        4. 通过**RX_NEG** 位 (SPI_CNTRL[1] = 0)选择数据在串行时钟的上升沿锁存
        5. 通过**TX_BIT_LEN** 位域 (SPI_CNTRL[7:3] = 0x08)设置传输字的长度为8位
        6. 通过**LSB** 位 (SPI_CNTRL[10] = 0)设置为 MSB优先传输，不必理会**SP_CYCLE** 位域 (SPI_CNTRL[15:12])的设置，因为在本例中没有使用FIFO模式
4) 如果SPI主机是要发送一个字节的数据到外设，则将所要发送的数据写入寄存器**TX0[7:0]** (SPI_TX0[7:0]).
5) 如果SPI主机只是要从外设接收一个字节的数据，不必管被传输出去的数据是什么，只需要向寄存器SPI_TX0[7:0]写入0XFF.
6) 使能**GO_BUSY** 位(SPI_CNTRL[0] = 1)，以开始 SPI 接口的数据传输。
7) 等到SPI中断发生（如果中断使能为IE使能），或检测**GO_BUSY** 位直到被硬件自动清零.
8) 从寄存器**RX0[7:0]** (SPI_RX0[7:0])读出所接收到的一个字节的数据.
9) 重复步聚 4) 继续其他数据的传输或设置**SSR[0]** 为0 以停止外设.

例 2, SPI 控制器作为从机设备，由片外主机控制，外设通过SPI接口与片上 SPI 从机通信:

- 数据在时钟上升沿锁存
- 数据在时钟下降沿传输
- LSB先传送
- SPICLK空闲模式为高电平
- 每次输出/接收一个字节
- 片选信号为高电平触发

操作流程如下:

1) 写SPI_SSR寄存器配置从模式相关设定

   通过设定**SS_LVL** (SPI_SSR[2] = 1)和SS_LTRIG (SPI_SSR[4] = 1).选择输入的片选信号为高电平有效.

2) 将从机模式的相应设置写入寄存器SPI_CNTRL.

   1. 通过**SLAVE** 位 (SPI_CNTRL[18] = 1)设置SPI控制器为从机设备
   2. 通过**CLKP** 位 (SPI_CNTRL[11] = 1)选择串行时钟空闲状态
   3. 通过 **TX_NEG** 位 (SPI_CNTRL[2] = 1)选择数据传输发生在串行时钟下降沿
   4. 通过**RX_NEG** 位 (SPI_CNTRL[1] = 0)选择数据在串行时钟的上升沿锁存
   5. 通过**TX_BIT_LEN** 位域 (SPI_CNTRL[7:3] = 0x08) 设置字传输长度为 8 位
   6. 通过 **LSB** 位 (SPI_CNTRL[10] = 1) 设置为LSB传输优先

3) 如果SPI从机是要发送一个字节的数据到外设主机（被读），则将所要发送的数据写入到寄存器**TX0[7:0]** (SPI_Tx0[7:0]).

4) 如果SPI从机仅从外设主机接收一字节数据（被写），用户不必关心什么数据将被传输，只需要向寄存器SPI_TX0[7:0]写入0XFF.

5) 使能**GO_BUSY** bit (SPI_CNTRL[0] = 1)，等到外设的片选触发输入和串行时钟输入，开始数据传输到SPI接口.

6) 等到SPI中断发生（如果IE使能），或检测**GO_BUSY** 位直到被硬件自动清零.

7) 在寄存器**RX[7:0]** (SPI_RX0[7:0])读出所接收到的一个字节的数据.

8) 重复步聚3) 继续其他数据传输或关闭**GO_BUSY** 位停止数据传输.

5.10.7 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
SPI 基地址: **SPI0_BA = 0x4003_0000** **SPI1_BA = 0x4003_4000** **SPI2_BA = 0x4013_0000**				
SPI_CNTRL x=0,1,2	SPIx_BA+0x00	R/W	控制及状态寄存器	0x0500_3004
SPI_DIVIDER x=0,1,2	SPIx_BA+0x04	R/W	时钟除频寄存器	0x0000_0000
SPI_SSR x=0,1,2	SPIx_BA+0x08	R/W	片选寄存器	0x0000_0000
SPI_RX0 x=0,1,2	SPIx_BA+0x10	R	接收数据寄存器0	0x0000_0000
SPI_RX1 x=0,1,2	SPIx_BA+0x14	R	接收数据寄存器1	0x0000_0000
SPI_TX0 x=0,1,2	SPIx_BA+0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1 x=0,1,2	SPIx_BA+0x24	W	数据发送寄存器1	0x0000_0000
SPI_VARCLK x=0,1,2	SPIx_BA+0x34	R/W	可变时钟类型寄存器	0x007F_FF87
SPI_DMA x=0,1,2	SPIx_BA+0x38	R/W	SPI DMA 控制寄存器	0x0000_0000
SPI_CNTRL2 x=0,1,2	SPIx_BA+0x3C	R/W	控制及状态寄存器2	0x0000_0000
SPI_FIFO_CTL x=0,1,2	SPIx_BA+0x40	R/W	FIFO 控制寄存器	0x0000_0000
SPI_STATUS x=0,1,2	SPIx_BA+0x44	R/W	FIFO 状态寄存器	0x0500_0000

注 : 软件编写 CNTRL 寄存器时, GO_BUSY 位必须最后写入.

5.10.8 寄存器描述

SPI控制与状态寄存器 (SPI_CNTRL)

寄存器	偏移量	R/W	描述	复位后的值
SPI_CNTRL	SPIx_BA+0x00	R/W	控制与状态寄存器	0x0500_0004

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

Bits	描述	
[31:28]	保留	保留
[27]	TX_FULL	**发送 FIFO满标志 (只读)** 与SPI_STATUS[27]比特相同 1 = 发送FIFO已满 0 = 发送FIFO没有满.
[26]	TX_EMPTY	**发送 FIFO空标志(只读)** 与SPI_STATUS[26]比特相同 1 = 发送FIFO为空. 0 = 发送FIFO不为空.
[25]	RX_FULL	**接收FIFO满标志 (只读)** 与SPI_STATUS[25]比特相同 1 =接收FIFO已满. 0 =接收FIFO没有满.
[24]	RX_EMPTY	**接收FIFO空标志(只读)** 与SPI_STATUS[24]比特相同 1 =接收FIFO为空. 0 = 接收FIFO不为空.
[23]	VARCLK_EN	**可变时钟使能(只用于主模式)** 1 = 串行时钟输出频率是可变的. 输出频率由VARCLK, DIVIDER, 和 DIVIDER2的值决定.

		0 = 串行时钟输出频率是固定的，只由DIVIDER的值决定. **注**：当使能VARCLK_EN 位时, TX_BIT_LEN 必须设置为0x10 (16 bits mode)
[22]	**TWOB**	**使能两位（Two Bits）传输模式** 1 = 使能两位传输模式. 0 = 禁用两位传输模式. **注**：当使能TWOB, 两位数据从SPI_TX1/0寄存器输出，从SPI_RX1/0寄存器接收.
[21]	**FIFO**	**FIFO 模式** 1 =使能 FIFO 模式. 0 = 关闭FIFO 模式. **注意:** 1. 使能FIFO模式之前，其它相关设定应该提前设好. 2. 主模式下，如果FIFO模式使能，软件将数据写到FIFO以后，GO_BUSY比特由硬件自动置为"1"。当SPI控制器空闲时，GO_BUSY将自动清为0。如果存储在发送FIFO中的所有数据都发送完毕，TX_EMPTY将被置为1，GO_BUSY将被清为0.
[20]	**保留**	保留
[19]	**REORDER**	**重排序模式选择** 1 = 使能字节重排序功能，在每个字节之间插入一个字节的休眠间隔。字节休眠间隔周期由SP_CYCLE定义 0 = 禁用字节重排序功能. 註: 1. 当使能字节重排功能时，TX_BIT_LEN必须设置为 16, 24或32 2. 当运作于从机模式且片选信号设置为电平触发时，主机输出的片选信号在字节休眠间隔期间，必须维持在有效状态 3. 可变串行时钟功能或者双I/O模式使能时，不支持字节重排序
[18]	**SLAVE**	**从机模式** 1 = 从机模式 0 = 主机模式
[17]	**IE**	**传输中断使能** 1 = 使能SPI传输中断 0 = 禁用SPI传输中断
[16]	**IF**	**传输中断标志** 1 = SPI控制器完成一次传输 0 = 自从该位清0之后，没有传输完成. **注**: 该位写1清零.
[15:12]	**SP_CYCLE**	**休眠间隔(只用于主模式)** 这四位用于配置连续两次传输之间的休眠间隔。休眠间隔是指从上一次传输的最后一个时钟沿到下一次传输的第一个时钟沿之间的间隔。默认值为0x3.。间隔周期见如下方程: (SP_CYCLE[3:0] + 0.5) * SPICLK时钟周期 SP_CYCLE = 0x0 … 0.5 SPICLK 时钟周期 SP_CYCLE = 0x1 … 1.5 SPICLK时钟周期

		…… SP_CYCLE = 0xE … 14.5SPICLK 时钟周期 SP_CYCLE = 0xF … 15.5 SPICLK 时钟周期 如果可变时钟功能使能，FIFO模式下发送FIFO不为空，休眠间隔最小周期为(6.5 + SP_CYCLE) * SPICLK 时钟周
[11]	**CLKP**	**时钟极性** 1 = 空闲时SPICLK为高电平 0 = 空闲时SPICLK为低电平.
[10]	**LSB**	**优先传送LSB** 1 =优先发送 LSB (SPI_TX0/1的比特0), 并且接收到的第一个比特放到 Rx 寄存器的LSB位置 (SPI_RX0/1的比特0). 0 =优先发送/接收MSB (根据CNTRL寄存器的Tx_BIT_LEN 决定SPI_TX0/1和SPI_RX0/1的第一个位的编號).
[9:8]	**保留**	保留
[7:3]	**TX_BIT_LEN**	**传输位长度** 该寄存器用于标示一次传输中，传输的比特长度，最多为32位. TX_BIT_LEN = 0x08 … 8 bit TX_BIT_LEN = 0x09 … 9 bits …… TX_BIT_LEN = 0x1F … 31 bits TX_BIT_LEN = 0x00 … 32 bits
[2]	**TX_NEG**	**下降沿发送数据** 1 = 在SPICLK下降沿改变发送数据输出信号 0 =在SPICLK上升沿改变发送数据输出信号
[1]	**RX_NEG**	**下降沿接收数据** 1 = 在SPICLK的下降沿锁定接收数据输入信号 0 =在SPICLK的上升沿锁定接收数据输入信号
[0]	**GO_BUSY**	**SPI传输控制位和忙状态标志** 1 = 在主机模式下, 写1到该位开始SPI数据传输; 在从机模式下, 写1到位表示从机准备好与主机进行通信. 0 = 写0到该位停止数据传输.读该位表示传输已经完成 如果FIFO模式关闭，数据传输过程中，该位值保持为1, 传输完成后，该位自动清零。软件可以读该位查看SPI是否忙。 FIFO模式下，该比特由硬件控制，软件不应该修改该位。从模式下，软件读该位总是返回1。主模式下，该位反应SPI的状态为忙还是空闲 **注**: 1. FIFO模式关闭时，在写1到GO_BUSY之前，所有寄存器都应该设置完成. 2. FIFO模式关闭并且软件使用TX或者RX PDMA功能来传输数据时，在PDMA传输完成之后，这个比特将被清为0

SPI 除频寄存器 (SPI_DIVIDER)

寄存器	偏移量	R/W	描述	复位后的值
SPI_DIVIDER	SPIx_BA+0x04	R/W	时钟除频寄存器(只用于主模式)	0x0000_0000

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

Bits	描述	
[31:24]	**保留**	保留
[23:16]	**DIVIDER2**	**时钟除频2寄存器** (只用于主模式) 这些位用于设置第2个频率除频器的值. 在SPICLK引脚产生的串行时钟, 可根据下列方程获得所期望的频率: $f_{sclk} = \frac{f_{pclk}}{(DIVIDER2+1)*2}$ 当VARCLK_EN置1时，此设置才有意义.
[15:8]	**保留**	保留
[7:0]	**DIVIDER**	**时钟除频寄存器** (只用于主模式) 这些位用于设置频率除频器的值. 在SPICLK引脚所输出的串行时钟频率f_{spi_eclk}. 可根据下列方程获得所期望的频率: 如果BCn，SPI_CNTRL2[31]，等于0 $f_{spi\_eclk} = \frac{f_{system\_clock}}{(DIVIDER+1)*2}$ 否则如果BCn等于1. $f_{spi\_eclk} = \frac{f_{spi\_clock\_src}}{(DIVIDER+1)}$ $f_{spi\_clock\_src}$ 是SPI引擎的时钟源，在CLKSEL1寄存器中选择

SPI片选寄存器(SPI_SSR)

寄存器	偏移量	R/W	描述	复位后的值
SPI_SSR	SPI0_BA+0x08	R/W	片选寄存器	0x0000_0000

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

Bits	描述	
[31:6]	保留	保留
[5]	LTRIG_FLAG	**电平触发完成标志** 从模式下，片选为电平触发时，当前传输完成以后，这个位用来指示收到的比特数是否满足要求. 1 = 收到的比特长度等于TX_BIT_LEN的定义. 0 =收到的比特长度与TX_BIT_LEN的定义不符. **注:** 这个比特是只读的。当GO_BUSY比特被软件置为1时，4个SPI引擎时钟周期+1个系统时钟周期之后，该位自动清0。FIFO模式下，该位无效
[4]	SS_LTRIG	**片选电平触发选择 (只用于从模式)** 1: 片选信号为电平触发。根据 SS_LVL决定是高电平/低电平触发 0: 片选信号为边沿触发。根据 SS_LVL决定是下降沿/上升沿触发.
[3]	AUTOSS	**自动从机选择(只用于主模式)** 1 = 该位置位，SPISSx0/1信号将自动产生。这意味着当发送/接收开始时，被SPI_SSR[1:0]选择的片选信号引脚将由SPI控制器自动设置为有效状态，而当数据传输结束时，SPI控制器会自动将从机选择引脚设置为非有效状态. 0 = 该位清位，软件通过设置或清零SPI_SSR[1:0]的相关位来决定片选引脚输出为有效状态或无效状态.
[2]	SS_LVL	**片选触发电平选择** 定义片选信号(SPISSx0/1)的有效电平. 1 = 片选信号SPISSx0/1 在高电平/上升沿有效 0 = 片选信号SPISSx0/1 在低电平/下降沿有效.

[1:0]	**SSR**	**从机选择寄存器(只用于主模式)** 当AUTOSS位被清除，对SSR[1:0]任何一位写1，将会激活所对应的SPISSx0/1片选线，写0则SPISSx0/1线呈现非有效状态。 当AUTOSS位被置1，对SSR[1:0]任何一位写0，将会使该位所对应的SPISSx0/1引脚输出无效状态；对SSR[1:0]任何一位写1，该位所对应的SPISSx0/1引脚的输出状态将由硬件自动控制。在收发时自动驱动SPISSx0/1引脚输出有效状态。 片选有效状态由SS_LVL决定. **注：**SPISSx0 在从机模式下被定义为片选输入

SPI 数据接收寄存器 (SPI_RX)

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

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

Bits	描述	
[31:0]	**RX**	数据接收寄存器 数据接收寄存器保存从SPI接收引脚收到的数据。如果FIFO等于0，最后收到的数据可以从该寄存器读出。如果FIFO为1，用户可以通过查看RX_EMPTY, SPI_CNTRL[24] 或 SPI_STATUS[24]获知是否FIFO中还有数据。 **注:** 数据接收寄存器为只读寄存器.

SPI 数据发送寄存器(SPI_TX)

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

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

Bits	描述	
[31:0]	TX	数据发送寄存器 数据发送寄存器保存下一次将发送的数据。数据的有效长度根据SPI_CNTRL寄存器内定义的长度决定。 例如，TX_BIT_LEN 设定为 0x08，TX[7:0] 内的数据将被发送。如果 Tx_BIT_LEN 设定为 0x00，SPI 控制器将发送/接收32位数据. **注意**：从模式下，FIFO比特关掉时，如果SPI控制器希望发送数据到主机，应该先写发送寄存器再将GO_BUSY位置为1

SPI 可变时钟类型寄存器 (SPI_VARCLK)

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

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

Bits	描述	
[31:0]	VARCLK	**可变时钟类型** 该寄存器的值决定了SPI串行时钟的频率输出模型。如果可变时钟功能关闭，该寄存器不起作用。细节请参考前面“可变时钟功能”的描述.

SPI_DMA 控制寄存器 (SPI_DMA)

寄存器	偏移量	R/W	描述	复位后的值
SPI_DMA	SPIx_BA+0x38	R/W	SPI DMA 控制寄存器	0x0000_0000

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

Bits	描述	
[31:2]	保留	保留
[2]	PDMA_RST	**PDMA 复位** 用来复位SPI PDMA功能到默认状态. 1= 复位SPI控制器的 PDMA 控制逻辑，该位将自动清0. 0 = 不起作用.
[1]	RX_DMA_GO	**接收DMA 开始** 对该位置1 ，开始PDMA接收模式，当有数据被写到接收缓冲中的时候，SPI控制器将自动向PDMA控制器发出请求信号。PDMA传输完成之后，该位由硬件自动清0。 如果软件使用RX_PDMA功能接收数据，但是TX_PDMA是关闭的，GO_BUSY比特由软件控制 如果软件使用多于1个PDMA通道来传输数据，推荐使能FIFO模式 从模式下，FIFO比特被关闭，如果接收PDMA使能，但是发送PDMA被关闭，边沿触发时，两次连续传输之间的最小间隔必须大于(9个SPI时钟周期 + 4 APB时钟周期)，电平触发时应该大于(9.5 SPI时钟周期+4 APB时钟周期)
[0]	TX_DMA_GO	**发送DMA开始** 对该位置1 ，开始PDMA发送模式，SPI控制器将自动向PDMA控制器发出请求信号。 PDMA传输完成之后，该位自动清0。 如果采用PDMA发送数据，软件不要将SPI_CNTRL寄存器的GO_BUSY位置1. 该位由硬件自动控制。 当需要时SPI控制器内的PDMA控制器会自动将其置1。 从模式下，FIFO模式关闭时，边沿触发时，两次传输之间的最小休眠间隔为(8 SPI串行时钟周期+14 APB时钟周期)，电平触发时两次传输之间

		的最小间隔为(9.5 SPI串行时钟周期+14APB时钟周期)。如果两比特传输模式使能，额外再加18 APB时钟周期

SPI 控制与状态寄存器2 (SPI_CNTRL2)

寄存器	偏移量	R/W	描述	复位后的值
SPI_CNTRL2	SPIx_BA+0x3C	R/W	状态与控制寄存器2	0x0000_0000

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

Bits	描述	
[31]	**BCn**	**SPI 引擎时钟源向后兼容** 1 = 时钟配置不向后兼容。SPI 串行时钟除以 (DIVIDER1+1) (SPI_DIVIDER寄存器) 0 =时钟配置不向后兼容。SPI 串行时钟除以 (DIVIDER1+1)*2 (SPI_DIVIDER寄存器) 细节请参考SPI_DIVIDER寄存器
[30:17]	**保留**	保留
[16]	**SS_INT_OPT**	**片选无效中断选择** 该位只用于从模式下，SPI控制器被配置为电平触发的模式。 1 = 当片选信号无效时，IF位将被置为1。 0 =当片选信号无效时，IF位不会置位.
[15:14]	**保留**	保留
[13]	**DUAL_IO_EN**	**双 I/O 模式使能** 1 = 双 I/O 使能. 0 = 关闭双I/O 功能
[12]	**DUAL_IO_DIR**	**双I/O 模式方向控制** 1 = 双I/O输出模式. 0 =双I/O输入模式.

[11]	**SLV_START_INTSTS**	**3线模式传输开始中断标志** 从模式下，3线模式时，用来指示传输已经开始。与SPI_STATUS[11]相同 1 = 从模式下，3线模式，传输已经开始。当传输结束时，该位自动清0。软件也可以写1清0. 0 = SSTA_INTEN比特使能之后，从设备还没有检测到SPI时钟信号.
[10]	**SSTA_INTEN**	**3线模式传输开始中断使能** 从模式下，3线模式时用来使能传输开始中断。如果传输开始以后，超过用户定义的时间，传输仍没有结束，用户可以设置SLV_ABORT来终止传输. 1 = 传输开始中断使能。当传输结束或者SLV_START_INTSTS 比特被清除（写1清除），该位将被清0. 0 = 禁止传输开始中断.
[9]	**SLV_ABORT**	**3线模式终止传输** 正常情况下，当接收到的数据位数等于TX_BIT_LEN 的定义时将发生中断. 从模式无片选时，如果接收到的数据位数小于TX_BIT_LEN 的定义并且没有串行时钟信号输入超过一次传输时间，用户可以通过设置该位来终止当前传输，然后用户将得到一个传输结束中断. **注:** 软件将该位置位之后，该位由硬件自动清0.
[8]	**NOSLVSEL**	**使能3线模式** 从模式下忽略片选信号。SPI控制器可以工作在3线模式，包括SPICLK, SPI_MISO, 和 SPI_MOSI 3根线. 1 = 控制器为3线双向接口. 0 = 控制器为4线双向接口. **注:** 3线模式下， SS_LTRIG, SPI_SSR[4], 将被自动设为 1.
[7:0]	**保留**	保留

SPI FIFO 控制寄存器 (SPI_FIFO_CTL)

寄存器	偏移量	R/W	描述	复位后的值
SPI_FIFO_CTL	SPIx_BA+0x40	R/W	SPI FIFO 控制寄存器	0x4400_0000

31	30	29	28	27	26	25	24
保留	TX_THRESHOLD			保留	RX_THRESHOLD		
23	22	21	20	19	18	17	16
保留		TIMEOUT_INTEN	保留				
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
保留	RXOV_INTEN	保留		TX_INTEN	RX_INTEN	TX_CLR	RX_CLR

Bits	Description	
[31]	**保留**	保留
[30:28]	**TX_THRESHOLD**	**发送 FIFO 阈值** 3 个比特, 值从 0 ~7. 如果发送FIFO中有效数据个数小于或者等于**TX_THRESHOLD**的值, **TX_INTSTS** 将被设为1, 否则 **TX_INTSTS** 将被清为 0.
[27]	**保留**	保留
[26:24]	**RX_THRESHOLD**	**接收 FIFO 阈值** 3 个比特，值从 0 ~7. 如果接收FIFO中的有效数据个数大于**RX_THRESHOLD**的值, **RX_INTSTS** 将被设为1, 否则 **RX_INTSTS** 将被清为 0
[23:22]	**保留**	保留
[21]	**TIMEOUT_INTEN**	**使能接收FIFO超时中断** 1= 使能超时中断 0 =禁止超时中断
[20:7]	**Reserved**	保留
[6]	**RXOV_INTEN**	**接收 FIFO 溢出中断使能** 1 = 使能接收FIFO 溢出中断.

		0 = 禁止接收FIFO 溢出中断.
[5:4]	**保留**	保留
[3]	**TX_INTEN**	**发送阈值中断使能** 1 = 使能发送阈值中断. 0 = 禁止发送阈值中断.
[2]	**RX_INTEN**	**接收阈值中断使能** 1 = 使能接收阈值中断. 0 = 禁止接收阈值中断.
[1]	**TX_CLR**	**清除发送FIFO缓冲** 1 = 清除发送 FIFO缓冲。TX_FULL 标志将被清成0， TX_EMPTY 标志将被置为 1. 该位由硬件自动清成0 0 = 没有作用.
[0]	**RX_CLR**	**清除接收FIFO缓冲** 1 = 清除接收FIFO缓冲。RX_FULL标志将被清成0， RX_EMPTY标志将被置为 1。该位由硬件自动清成0. 0 =没有作用.

SPI 状态寄存器 (SPI_STATUS)

寄存器	偏移量	R/W	描述	复位后的值
SPI_STATUS	SPIx_BA+0x44	R/W	SPI 状态寄存器	0x0500_0000

31	30	29	28	27	26	25	24
TX_FIFO_COUNT				TX_FULL	TX_EMPTY	RX_FULL	RX_EMPTY
23	22	21	20	19	18	17	16
保留			TIMEOUT				IF
15	14	13	12	11	10	9	8
RX_FIFO_COUNT				SLV_START_INTSTS	保留		
7	6	5	4	3	2	1	0
保留			TX_INTSTS	保留	RX_OVER_RUN	保留	RX_INTSTS

Bits	描述	
[31:28]	TX_FIFO_COUNT	**发送 FIFO中的数据个数 (只读)** 该域指示发送FIFO中有效的数据个数
[27]	TX_FULL	**发送 FIFO缓冲满标志 (只读)** 和SPI_CNTRL[27]含义相同 1 = 发送FIFO已满. 0 = 发送FIFO没有满.
[26]	TX_EMPTY	**发送 FIFO缓冲空标志(只读)** 和 SPI_CNTRL[26] 含义相同 1 = 发送FIFO为空. 0 = 发送FIFO不为空.
[25]	RX_FULL	**接收 FIFO缓冲满标志 (只读)** 和 SPI_CNTRL[25] 含义相同 1 = 接收FIFO已满. 0 = 接收FIFO没有满.
[24]	RX_EMPTY	**接收 FIFO缓冲空标志(只读)** 和 SPI_CNTRL[24] 含义相同 1 = 接收FIFO为空. 0 = 接收FIFO不为空.

[23:21]	**保留**	保留
[20]	**TIMEOUT**	**超时中断标志** 1 = 接收FIFO不为空，主模式下超过64个SPI时钟周期，从模式下超过576个SPI引擎时钟周期没有读操作，该位将被置位。当接收FIFO被用户读过之后，超时状态将被自动清0. 0 = 接收FIFO没有发生超时. **注:** 这个bit可以写1清除.
[19:17]	**保留**	保留
[16]	**IF**	**SPI传输中断标志** 和 SPI_CNTRL[16]含义相同 1 = SPI控制器结束一次传输. 0 = 从上次清0之后，没有传输结束. **注: 这**个bit可以写1清除.
[15:12]	**RX_FIFO_COUNT**	**接收 FIFO中的数据个数 (只读)** 该域用来指示接收FIFO中有效的数据个数
[11]	**SLV_START_INTSTS**	**3线模式开始传输中断状态** 和SPI_CNTRL2[11] 含义相同 从模式3线模式下，用来指示传输已经开始. 1 = 从模式没有片选的情况下，传输已经开始。传输结束或者向该位写1将被清成0. 0 = 自从SSTA_INTEN设成1之后，从模式没有检测到SPI串行时钟.
[10:5]	**保留**	保留
[4]	**TX_INTSTS**	**发送 FIFO 阙值中断标志 (只读)** 1 = 发送FIFO中有效数据个数小于或者等于**TX_THRESHOLD.** 0 =发送FIFO中有效数据个数大于 **TX_THRESHOLD.** **注:** 如果 TX_INTEN = 1 并且 TX_INTSTS = 1，SPI 控制器将产生SPI中断请求
[3]	**保留**	保留
[2]	**RX_OVER_RUN**	**接收 FIFO 溢出标志** 当接收FIFO已满时，下一个接收到的数据将被丢弃，并且该位将被置位 **注:** 这个bit可以写1清除.
[1]	**保留**	保留
[0]	**RX_INTSTS**	**接收 FIFO 阙值中断标志 (只读)** 1 = 接收FIFO中有效数据个数大于 **RX_THRESHOLD.** 0 = 接收FIFO中有效数据个数小于或者等于 **RX_THRESHOLD.**

		注: 如果 RXINT_EN = 1 并且 RX_INTSTS = 1, SPI 将发生中断.

5.11 定时器控制器 (TMR)

5.11.1 概述

定时器模块包含4组32位定时器，TIMER0~TIMER3，提供用户便捷的计数定时功能。定时器模块可支持例如频率测量，事件计数，间隔时间测量，时钟产生，时间延迟等功能。定时器可在计时溢出时产生中断信号，也可在操作过程中提供计数的当前值。

5.11.2 特性

- 4 组 32-位定时器，带24位向上定时器和一个8位的预分频计数器
- 每个定时器都有独立的时钟源
- 提供one-shot, periodic, toggle 和 连续 计数操作模式
- 超时周期 = (输入的定时器时钟周期) * (8-bit预分频计数器 + 1) * (24-bit TCMP)
- 最大计数周期= (1 / T MHz) * (2^8) * (2^24), T为定时器时钟频率
- 通过 TDR（定时器数据寄存器）可读取内部24位向上计数器的值
- 支持事件计数功能可用于计数外部管脚的事件个数
- 支持输入捕获功能，捕获外部管脚高/低电平的时间
- 支持输入捕获功能，复位定时器计数器的值
- 当定时器中断信号产生时（TIF=1），支持将芯片从idle/深度睡眠模式下唤醒

5.11.3 框图

每个通道带一个8位预分频计数器，一个24位向上计数器，一个24位比较寄存器和一个中断请求信号. 参考图 5-72的定时器控制框图。每个通道有4个时钟源可选择, 图 5-73为时钟源控制功能。软件可以编程 8-位 预分频计数器来决定 24-位向上计数器的时钟周期。

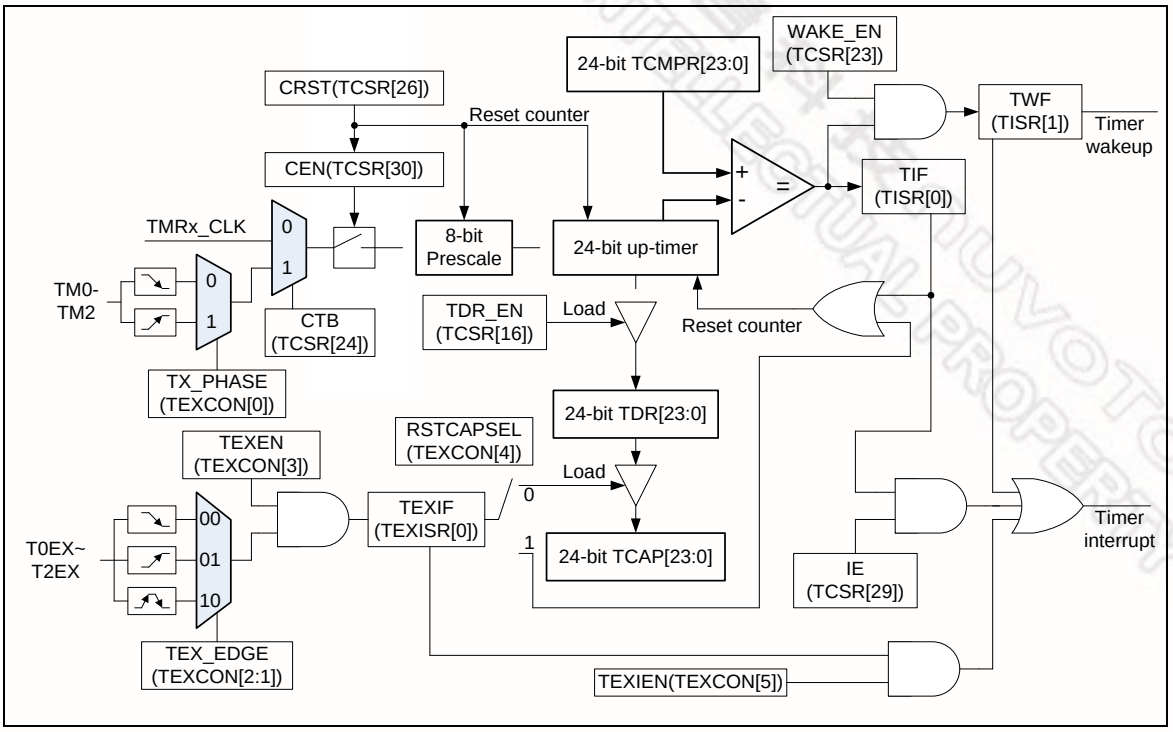


图 5-72 定时器控制器框图

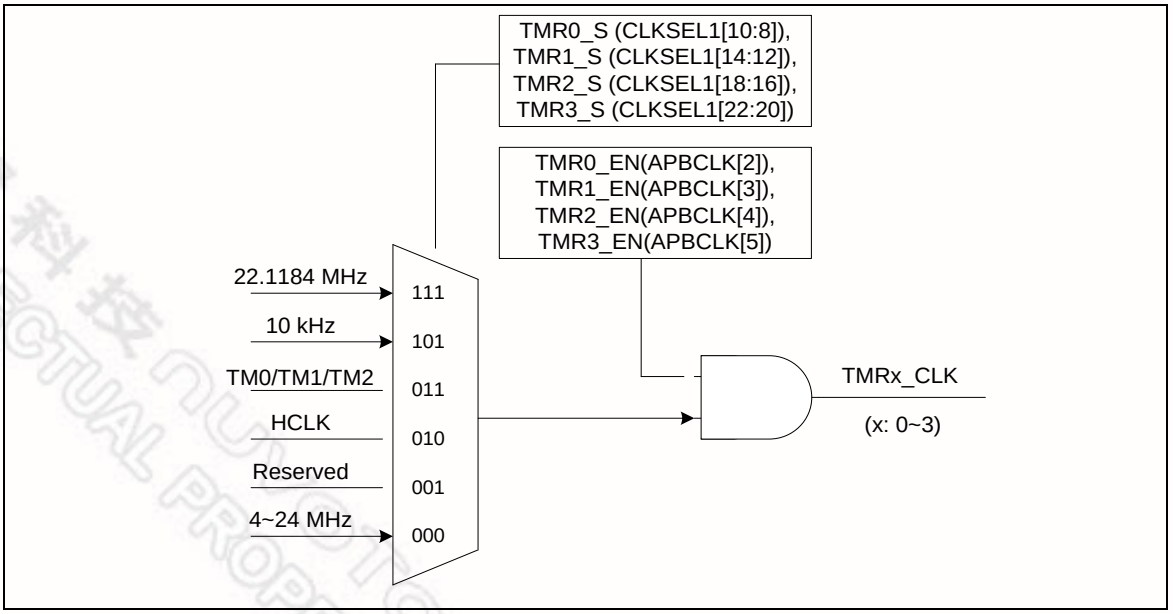


图 5-73 定时器控制器的时钟源

5.11.4 功能描述

定时器控制器提供one-shot, period，toggle和连续计数操作模式。也提供事件计数功能来计数来自外部引脚的事件。还提供输入捕获功能来捕获或者复位计数器的值。每种操作模式如下所述。

One –Shot 模式

5.11.4.1 如果定时器工作在单周期（one-shot） 模式且CEN (TCSR[30]定时器使能位) 置1，定时器的计数器开始上数 。一旦定时器计数器的值达到定时器比较器寄存器 (TCMPR) 的值，且IE (TCSR[29]中断使能位) 置1，则定时器中断标志置位，产生中断信号并送到NVIC通知CPU，表明定时器计数发生溢出。如果IE (TCSR[29] 中断使能位) 置0，无中断信号发生 。

在此工作模式下，一旦定时器计数器的值达到定时器比较寄存器(TCMPR) 的值时，定时器计数器的值回到初始值，并且CEN(定时器使能位)被定时器计数器清成0，定时器计数操作停止.也就是说，写TCMPR并使能CEN之后，定时器计数并和TCMPR的值比较的操作仅执行一次。因此，该操作称为One-Shot 模式。

Periodic 模式

5.11.4.2 如果定时器工作在周期模式且CEN (TCSR[30]定时器使能位)置1，定时器计数器开始上数 。一旦定时器计数器的值达到定时器比较寄存器 (TCMPR)的值，且IE (TCSR[29]中断使能位) 设置为1，则定时器中断标志置位且产生中断信号，并发送到NVIC通知CPU，表示定时器计数溢出发生。如果IE (中断使能位)设置为0，无中断信号发生。在该工作模式下，一旦定时器计数器的值达到定时器比较器寄存器(TCMPR) 的值，定时器计数器的值返回计数初始值且CEN 保持为1 (持续使能计数)。定时器计数器再次计数。如果软件清除中断标志，一旦定时器计数器的值与定时器比较寄存器(TCMPR)的值匹配且IE (中断使能位)设置为1，产生中断信号并再次送到NVIC通知CPU。也就是说，定时器计数并与TCMPR比较的操作是周期性的。直到CEN设置为0，定时器计数操作才会停止，中断信号的产生也是周期性的。因此，这种操作模式称为Periodic 模式。

5.11.4.3

Toggle 模式

如果定时器工作在Toggle (TCSR MODE[1:0]= 0x2)模式且CEN (TCSR[30]定时器使能位)置1，定时器计数器开始计数。一旦定时器计数器的值(TDR)与定时器比较寄存器(TCMPR)的值匹配时，则TIF(TISR(0)定时器中断标志)将被置为1。如果IE (TCSR[29]中断使能位)为 1，并且TIF(TISR(0)定时器中断标志)置为1，则产生中断信号并送到NVIC通知CPU，指示定时器发生计数溢出。如果IE（TCSR[29]中断使能位）为0，将没有中断产生

在这种操作模式，一旦定时器计数器(TDR)的值与定时器比较寄存器(TCMPR)的值匹配，TIF将被置为1(TISR[0]定时器中断标志)，信号输出脚拉高(TMx引脚，x：0～2)。然后定时器计数器的值返回到计数初始值且 CEN 保持为 1 (TCSR[30]定时器使能位)， 定时器计数器重新上数。如果中断标志TIF由软件清除，一旦定时器计数器的值(TDR)与定时器比较寄存器中TCMPR的值匹配，，则定时器中断标志TIF置位，,相应 toggle输出 (TMx引脚)的低电平。定时器计数操作在CEN设置为0之后才
5.11.4.4 停止。因此, toggle输出 (TMx引脚)的信号 以50%的占空比来回切换高/低电平， 所以这种操作模式称为Toggle 模式。

连续计数 模式

如果定时器工作在连续计数模式(TCSR MODE[1:0]=0x3)且 CEN（TCSR[30] 定时器使能位）置1，定时器计数器开始上数。一旦定时器计数器的值(TDR)等于定时器比较寄存器(TCMPR)的值，TIF(TISR[0]定时器中断标志)将被置为1。如果此时 IE（TCSR[29] 中断使能位）的值为 1，则相应的中断信号产生，并送到NVIC通知CPU定时器计数器发生溢出。如果此时IE(TCSR[29]中断使能位)的值为0，则没有中断发生。。

该模式下，一旦定时器计数器的值(TDR)等于定时器比较寄存器（TCMPR）的值，TIF(TISR[0]定时器中断标志)将被置为1，如果CEN的值(TCSR[30]定时器使能位)保持为1，定时器计数器的值(TDR)

保持继续向上计数，而不会从0重新开始计数。用户可以立即改变不同的 TCMPR 值而不需要停止定时器计数器和重新开始计数。

例如，TCMPR 一开始为 80（TCMPR 的值必须大于1且小于2^{24} ）。首先，当 TDR 的值等于 80 则 TIF（TISR[0]定时器中断标志）被置为1。如果此时 CEN 保持为1 (定时器使能位)，则 TDR 的值将不会回到 0，它继续向上计数81, 82, 83,⋯ 到 2^{24} -1,然后 0, 1, 2, 3, ⋯ 到 2^{24} -1 一次又一次。接下来，如果用户设置 TCMPR 为 200 而并且将 TIF 清零，则当 TDR 达到 200 时， TIF 将被再次置为1。最后用户设置 TCMPR 为 500 并且将 TIF 清零，则当 TDR 达到 500 时， TIF 再次被置为1。从应用来看，中断发生取决于 TCMPR。

在此模式下，定时器计数是连续的，即使TIF被置为1。因此，这种操作模式被称为 连续计数模式。

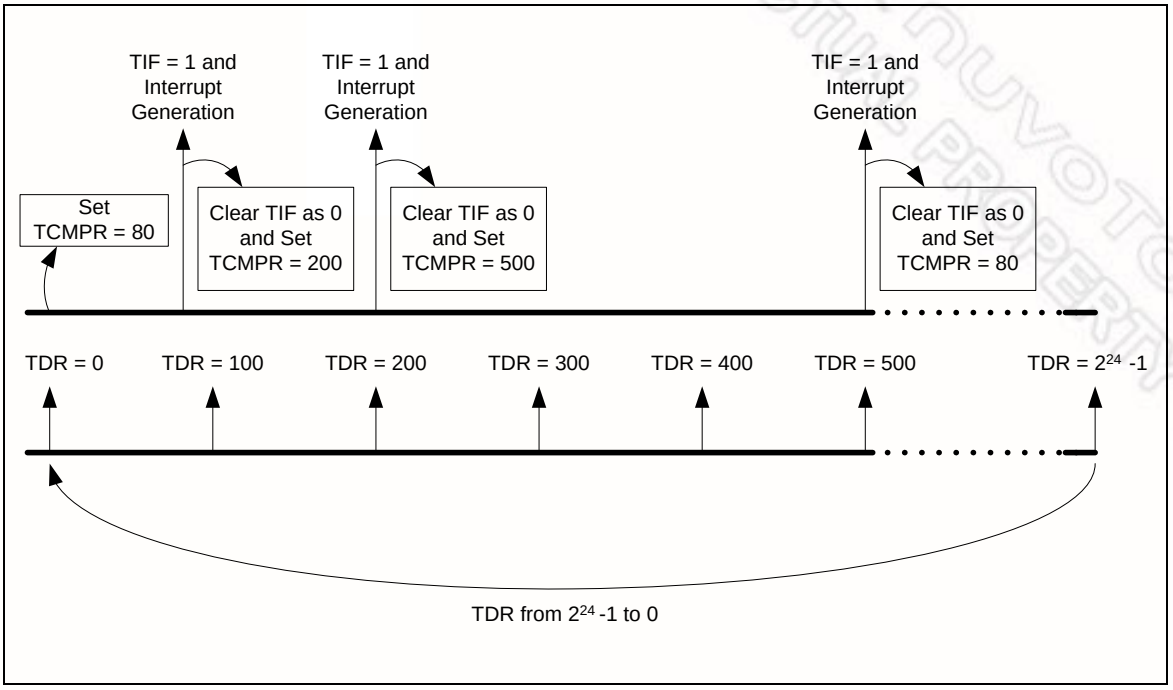


图 5-74 连续计数模式

5.11.4.5

事件计数功能

定时器控制器提供对来自 TMx 引脚（事件技术引脚）输入事件的计数功能，该模式被称为事件计数功能。在事件计数模式下，TCSR MODE[1:0]可以设为单次、周期和连续技术模式。控制寄存器的设定和其他模式一样，除了TCSR MODE[1:0]不能设为0x2（反转输出模式）并需要使能CTB(TCSR[24])。定时器控制器的时钟源 TMRx_CLK, 如图 5-73 所示，必须设为 HCLK。当TMx引脚状态发生变化时，事件计数器的值(TDR的值)将根据TX_PHASE(TEXCON[0])的设定计数。可以通过设置TCDB(TEXCONx[7]) 来使能或禁用 TM0~TM3 边沿检测防抖动功能以及设置

5.11.4.6 TEXCONx[0] 来设定 TM0~TM3 下降沿或上升沿计数。 如果禁用计数引脚防抖动功能，事件计数输入源的频率必须小于 1/3 HCLK；如果使能计数防抖动功能，事件计数源输入频率必须小于 1/8 HCLK的 频率，否则返回的 TDR 值将是不正确的。

输入捕获功能

该模式下，定时器将监控TxEX引脚（定时器外部捕获引脚）的状态变化，来保存定时器计数器的值(TDR的值)到定时器捕获寄存器(TCAP寄存器)或者复位定时器计数器的值(TDR)。定时器控制器的时钟源，TMRx_CLK(0：0～3)，如图 5-73，应该选择HCLK。

如果TEXEN (定时器外部引脚使能)设为1，并且RSTCAPSEL（TEXCON[4]定时器外部复位计数器/捕获模式选择）设为0，定时器工作在捕获模式。当TxEX(x：0，2或者3)引脚触发条件满足TEX_EDGE(TEXCON[2:1]定时器外部引脚检测边沿)的设定时，定时器计数器的值(TDR)将被捕获到TCAP寄存器，并且TEXIF(TEXISR[0]定时器外部中断标志)将被置为1。有3种触发条件可以设置。

如果TEXEN (定时器外部引脚使能)设为1，并且RSTCAPSEL也设为1，定时器工作在计数器复位模式。当TxEX(定时器外部捕获引脚)引脚触发条件满足TEX_EDGE(TEXCON[2:1]外部引脚检测边沿)的设定时，定时器计数器的值(TDR)将被复位成0，同时TEXIF（TEXISR[0]定时器外部中断标志）将被置为1。

TxEX触发边沿由TEX_EDGE(TEXCON[2:1]外部引脚检测边沿)选择。有3种触发条件：下降沿、上升沿或者下降沿和上升沿都触发。

当TxEX触发发生时，TEXIF(定时器外部中断标志)将被设为1，如果TEXIEN(定时器外部中断使能位)也为1，中断信号将产生并送到NVIC通知CPU发生了中断。

TEXDB(TEXCON[6])可以使能或者禁止TxEX引脚捕获防抖动功能。如果禁用TxEX防抖动功能（TEXDB=0），TxEX输入源的频率必须小于 1/3 HCLK的频率；如果使能TxEX防抖动功能(TEXDB=1)，TxEX输入源的频率必须小于 1/8 HCLK的 频率，

5.11.5 初始化范例

5.11.5.1 *定时器初始化*

1. 写CLKSEL1寄存器选择Timer的时钟源
2. 写APBCLK寄存器使能Timer的时钟
3. 如果定时器配置为反转输出模式或者事件计数模式，需要写GPx_MFP和ALT_MFP寄存器将相应引脚配置为TMx功能；如果定时器配置为捕获模式，需要写GPx_MFP和ALT_MFP寄存器将相应引脚配置为TxEX功能
4. 写TCSR寄存器的 PRESCALE[7:0] 配置预分频值
5. 写TCSR寄存器的 MODE[1:0] 配置定时器工作模式：单次、周期、反转、连续计数模式
6. 配置比较寄存器(TCMPR)，决定定时器超时周期
7. 配置TCSR寄存器的IE位使能Timer中断
8. 写TCSR寄存器的CEN位使能定时器开始计数
9. 如果使用中断的方式处理定时器事件，调用NVIC_EnableIRQ(TMRx_IRQn) （x: 0 ~ 3）使能Timer的中断；否则可以轮询TISR寄存器的中断标志位。

5.11.6 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
TIMER 基地址: **TMR01_BA = 0x4001_0000** **TMR23_BA = 0x4011_0000**				
TCSR0	TMR01_BA+0x00	R/W	Timer0 控制和状态寄存器	0x0000_0005
TCMPR0	TMR01_BA+0x04	R/W	Timer0 比较寄存器	0x0000_0000
TISR0	TMR01_BA+0x08	R/W	Timer0 中断状态寄存器	0x0000_0000
TDR0	TMR01_BA+0x0C	R	Timer0 数据寄存器	0x0000_0000
TCAP0	TMR01_BA+0x10	R	Timer0 捕获数据寄存器	0x0000_0000
TEXCON0	TMR01_BA+0x14	R/W	Timer0 外部控制寄存器	0x0000_0000
TEXISR0	TMR01_BA+0x18	R/W	Timer0 外部中断状态寄存器	0x0000_0000
TCSR1	TMR01_BA+0x20	R/W	Timer1 控制和状态寄存器	0x0000_0005
TCMPR1	TMR01_BA+0x24	R/W	Timer1 比较寄存器	0x0000_0000
TISR1	TMR01_BA +0x28	R/W	Timer1 中断状态寄存器	0x0000_0000
TDR1	TMR01_BA+0x2C	R	Timer1 数据寄存器	0x0000_0000
TCAP1	TMR01_BA+0x30	R	Timer1 捕获数据寄存器	0x0000_0000
TEXCON1	TMR01_BA+0x34	R/W	Timer1 外部控制寄存器	0x0000_0000
TEXISR1	TMR01_BA+0x38	R/W	Timer1 外部中断状态寄存器	0x0000_0000
TCSR2	TMR23_BA+0x00	R/W	Timer2 控制和状态寄存器	0x0000_0005
TCMPR2	TMR23_BA+0x04	R/W	Timer2 比较寄存器	0x0000_0000
TISR2	TMR23_BA+0x08	R/W	Timer2 中断状态寄存器	0x0000_0000
TDR2	TMR23_BA+0x0C	R	Timer2 数据寄存器	0x0000_0000
TCAP2	TMR23_BA+0x10	R	Timer2 捕获数据寄存器	0x0000_0000
TEXCON2	TMR23_BA+0x14	R/W	Timer2 外部控制寄存器	0x0000_0000
TEXISR2	TMR23_BA+0x18	R/W	Timer2 外部中断状态寄存器	0x0000_0000
TCSR3	TMR23_BA+0x20	R/W	Timer3 控制和状态寄存器	0x0000_0005
TCMPR3	TMR23_BA+0x24	R/W	Timer3 比较寄存器	0x0000_0000
TISR3	TMR23_BA+0x28	R/W	Timer3 中断状态寄存器	0x0000_0000

TDR3	TMR23_BA+0x2C	R	Timer3 数据寄存器	0x0000_0000
TCAP3	TMR23_BA+0x30	R	Timer3 捕获数据寄存器	0x0000_0000
TEXCON3	TMR23_BA+0x34	R/W	Timer3 外部控制寄存器	0x0000_0000
TEXISR3	TMR23_BA+0x38	R/W	Timer3 外部中断状态寄存器	0x0000_0000

5.11.7 寄存器描述

定时器控制寄存器(TCSR)

寄存器	偏移量	R/W	描述	复位后的值
TCSR0	TMR01_BA+0x00	R/W	Timer0 控制与状态寄存器	0x0000_0005
TCSR1	TMR01_BA+0x20	R/W	Timer1 控制与状态寄存器	0x0000_0005
TCSR2	TMR23_BA+0x00	R/W	Timer2 控制与状态寄存器	0x0000_0005
TCSR3	TMR23_BA+0x20	R/W	Timer3 控制与状态寄存器	0x0000_0005

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

Bits	描述	
[31]	DBGACK_TMR	**仿真器（ICE）调试模式响应禁止**(写保护位) 0 = 仿真器调试模式应答影响定时器计数. 当调试器调试模式响应时（进入调试模式），定时器计数器将被固定住. 1 = 仿真器调试模式响应禁止 无论调试器调试模式响应与否，定时器计数器将持续计数下去.
[30]	CEN	**计数器使能位** 1 = 开始计数 0 = 停止/暂停计数 **注1**: 停止状态下，设置CEN为1，使能24位计数器从上次停止的计数值继续计数. **注 2**: 在 one-shot 模式下 (MODE[28:27]=00b)，当相应的定时中断产生时 (IE[29]=1b)，该位由硬件自动清零.
[29]	IE	**中断使能** 1 = 使能定时器中断

		0 = 禁用定时器中断 当定时器中断使能时，当计数值与TCMPR寄存器内数值相同时，触发中断.
[28:27]	**MODE**	**定时器工作模式** 模式 / 定时器工作模式 00 / 定时器工作在单次触发模式(one-shot)，定时器溢出仅触发一次中断 (IE 使能)，进入中断后CEN自动清除为0. 01 / 定时器工作在周期模式(period)，定时器每次溢出都触发中断(IE 使能). 10 / 定时器工作在反转输出模式(toggle)，中断信号周期性产生(如果IE使能). 且相应的信号 (tout)以占空比为50%周期性反转输出. 11 / 定时器工作在连续计数模式.在TDR =TCMP时， (如果IE使能)相应的中断产生。然而，24位的上数计数器继续计数而不会复位. 细节请参考5.11.4.4 连续操作模式的描述
[26]	**CRST**	**计数器复位** 设置该位将复位24位上数计数器，8位预分频和使CEN为0. 0 = 无动作. 1 = 复位定时器的预分频计数器，内部24位向上计数器和CEN位
[25]	**CACT**	**定时器工作状态(只读)** 该位表示上数计数器的状态。 0 = 定时器未工作 1 = 定时器工作中
[24]	**CTB**	**计数模式使能位** 该位是计数模式使能位. 当定时器用作事件计数器时，该位需要设置成 1 则定时器将作为事件计数器。计数器侦测相位可以通过TX_PHASE域设定为外部引脚的上升/下降沿. 1 = 使能计数器模式 0 = 禁用计数器模式
[23]	**WAKE_EN**	**使能唤醒** 当 WAKE_EN 被置并且TIF也被置时，定时器控制器将产生唤醒信号唤醒CPU. 0 = 禁止唤醒CPU. 1 = 使能唤醒CPU
[22:17]	**保留**	保留
[16]	**TDR_EN**	**数据加载使能** 当置位TDR_EN时，计数器计数时，TDR (Timer数据寄存器) 将被24位上数计数器的值 不断更新。用户读这个寄存器就可以得到当前的计数值

		1 = 定时器数据寄存器更新使能 0 = 定时器数据寄存器禁止更新
[15:8]	**保留**	保留
[7:0]	**PRESCALE**	**预分频计数器** 输入的时钟在喂给计数器之前先除以Prescale +1进行预分频. 如果PRESCALE数值为0，不进行预分频.

定时器比较寄存器(TCMPR)

寄存器	偏移量	R/W	描述	复位后的值
TCMPR0	TMR01_BA+0x04	R/W	Timer0 比较寄存器	0x0000_0000
TCMPR1	TMR01_BA+0x24	R/W	Timer1 比较寄存器	0x0000_0000
TCMPR2	TMR23_BA+0x04	R/W	Timer2 比较寄存器	0x0000_0000
TCMPR3	TMR23_BA+0x24	R/W	Timer3 比较寄存器	0x0000_0000

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

Bits	描述	
[31:24]	**保留**	保留
[23:0]	**TCMP**	**定时器比较值** TCMP是24位比较寄存器. 当内部24位上数计数器的值与TCMP的值相等时，如果TCSR.IE[29]=1，就产生定时器中断请求. TCMP的值决定定时器计数周期. 定时溢出周期= (输入时钟周期) * (8-bit Prescale + 1) * (24-bit TCMP) **注1**: 不能在TCMP里写0x0或0x1，否则内核将运行到未知状态. **注2**: 当定时器工作在连续计数模式时，如果软件向该寄存器写入新的值，24位上数计数器将继续计数不会从0重新开始；当定时器工作在其它模式时，如果软件向该寄存器写入新的值，24位上数计数器将重新开始计数，并与新的值比较.

定时器中断状态寄存器(TISR)

寄存器	偏移量	R/W	描述	复位后的值
TISR0	TMR01_BA+0x08	R/W	Timer0 中断状态寄存器	0x0000_0000
TISR1	TMR01_BA+0x28	R/W	Timer1 中断状态寄存器	0x0000_0000
TISR2	TMR23_BA+0x08	R/W	Timer2 中断状态寄存器	0x0000_0000
TISR3	TMR23_BA+0x28	R/W	Timer3 中断状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	**保留**	保留
[1]	**TWF**	**定时器唤醒标志** 如果定时器导致CPU从掉电/睡眠唤醒，这个比特将被置为高. 由软件写1清除t. 0 = 定时器没有导致CPU唤醒. 1 = 定时器超时导致CPU 从掉电/睡眠唤醒.
[0]	**TIF**	**定时器中断标志** 定时器中断状态位. 当内部24位上数计数器的值与TCMP的值匹配时，TIF由硬件置位，写1清该位

定时器数据寄存器(TDR)

寄存器	偏移量	R/W	描述	复位后的值
TDR0	TMR01_BA+0x0C	R/W	Timer0 数据寄存器	0x0000_0000
TDR1	TMR01_BA+0x2C	R/W	Timer1 数据寄存器	0x0000_0000
TDR2	TMR23_BA+0x0C	R/W	Timer2 数据寄存器	0x0000_0000
TDR3	TMR23_BA+0x2C	R/W	Timer3 数据寄存器	0x0000_0000

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

Bits	描述	
[31:24]	**保留**	保留
[23:0]	**TDR**	**定时器数据寄存器** 1. CTB (TCSR[24]) = 0 : TCSR.TDR_EN 置1时，TDR 为 24 位上数计数器的值. 用户可以读 TDR 取得24位上数计数器的当前值 2. CTB (TCSR[24]) = 1 : TCSR.TDR_EN 置1时，TDR 为 24位上数事件计数器的值. 用户可以读 TDR 取得24位上数事件计数器的当前值

定时器捕获数据寄存器 (TCAP)

寄存器	偏移量	R/W	描述	复位后的值
TCAP0	TMR01_BA+0x10	R/W	Timer0 捕获数据寄存器	0x0000_0000
TCAP1	TMR01_BA+0x30	R/W	Timer1 捕获数据寄存器	0x0000_0000
TCAP2	TMR23_BA+0x10	R/W	Timer2 捕获数据寄存器	0x0000_0000
TCAP3	TMR23_BA+0x30	R/W	Timer3 捕获数据寄存器	0x0000_0000

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

Bits	描述	
[31:24]	**保留**	保留
[23:0]	**TCAP**	**定时器捕获数据寄存器** 当 TEXEN (TEXCON[3]) 等于1, RSTCAPn(TTXCON[4]) 等于 0时, 外部TEX引脚上的信号发生了TEX_EDGE(TEXCON[2:1]) 定义的转变时，内部24位上数计数器的值将被锁存到TCAP中，用户可以读这个寄存器取得计数值.

定时器外部控制寄存器 (TEXCON)

寄存器	偏移量	R/W	描述	复位后的值
TEXCON0	TMR01_BA+0x14	R/W	Timer0 外部控制寄存器	0x0000_0000
TEXCON1	TMR01_BA+0x34	R/W	Timer1外部控制寄存器	0x0000_0000
TEXCON2	TMR23_BA+0x14	R/W	Timer2外部控制寄存器	0x0000_0000
TEXCON3	TMR23_BA+0x34	R/W	Timer3外部控制寄存器	0x0000_0000

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

Bits	描述	
[31:8]	保留	保留
[7]	TCDB	**定时器事件计数引脚防抖功能使能** 1 = 使能防抖功能. 0 = 禁止防抖功能. 如果这个比特使能, TM0~TM3 引脚上的信号边沿将被防抖电路检测
[6]	TEXDB	**定时器外部捕获引脚防抖功能使能** 1 =使能防抖功能. 0 =禁止防抖功能. 如果这个比特使能，T0EX~T3EX引脚上的信号边沿将被防抖电路检测
[5]	TEXIEN	**定时器外部捕获中断使能** 1 = 定时器外部中断使能. 0 = 禁止定时器外部中断. 如果定时器外部中断使能，当TEX引脚上的信号发生了TEX_EDGE(TEXCON[2:1])定义的转变时，定时器发出外部中断信号送到NVIC通知CPU 例如：当TEXIEN = 1, TEXEN = 1, 并且 TEX_EDGE = 00时, TEX引脚上一个 1 到 0 的转变将导致TEXIF(TEXISR[0]) 中断标志被设，然后中断信号会被送到NVIC通知CPU发生了中断.
[4]	RSTCAPn	**定时器外部复位计数器/捕获模式选择**

		1 = TEX 引脚上信号转变导致定时器计数器复位. 0 = TEX 引脚上信号转变导致定时器捕获.
[3]	**TEXEN**	**定时器外部捕获引脚使能** 该位使能TEX引脚的复位/捕获功能. 1 = TEX引脚上的检测到信号转变将导致捕获或者计数器复位. 0 = TEX 引脚被忽略.
[2:1]	**TEX_EDGE**	**定时器外部捕获引脚边沿检测** 00 = 检测TEX引脚上1 到 0 的转变. 01 =检测TEX引脚上0到 1 的转变. 10 = TEX引脚上1 到 0 和0到 1 的转变都将被检测. 11 = 保留.
[0]	**TX_PHASE**	**定时器外部事件计数相位** 该比特指示外部计数引脚的相位. 1 =外部计数引脚上升沿被计数. 0 =外部计数引脚下降沿被计数.

定时器外部中断状态寄存器 (TISR)

寄存器	偏移量	R/W	描述	复位后的值
TEXISR0	TMR01_BA+0x18	R/W	Timer0 外部中断状态寄存器	0x0000_0000
TEXISR1	TMR01_BA+0x38	R/W	Timer1外部中断状态寄存器	0x0000_0000
TEXISR2	TMR23_BA+0x18	R/W	Timer2外部中断状态寄存器	0x0000_0000
TEXISR3	TMR23_BA+0x38	R/W	Timer3外部中断状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	**保留**	保留
[0]	**TEXIF**	**定时器外部中断标志** 这个比特指示定时器外部中断状态. 当TEXEN (TEXCON[3])等于1，并且TEX引脚发生了TEX_EDGE (TEXCON[2:1])定义的信号转变，这个比特由硬件置1。用户可以写1清0. 例如：当 TEXEN = 1, TEX_EDGE = 00是, TEX引脚上 信号1 到 0转变将导致TEXIF 被置.

5.12 看门狗定时器 (WDT)

5.12.1 概述

看门狗定时器的作用是在软件出问题时执行系统复位功能，可以防止系统无限止地挂机，除此之外，看门狗定时器还可将芯片由掉电模式唤醒。

5.12.2 特征

- 18-位计数器，可以在超时之后，指定的复位延迟时间以后，复位系统.
- 可选择的定时超时间隔 (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 / WDT_CLK) * 63
- 复位延迟时间可以选择，包括(1024+2)、(128+2) 、(16+2) 或者 (1+2) 个WDT_CLK周期.
- 如果CWDTEN(Config0[31]看门狗使能)设为0，支持芯片上电或者复位以后，强迫看门狗使能
- 当WDT时钟源选择内部10K时，支持看门狗超时唤醒功能

5.12.3 框图

看门狗定时器时钟控制和框图如下所示.

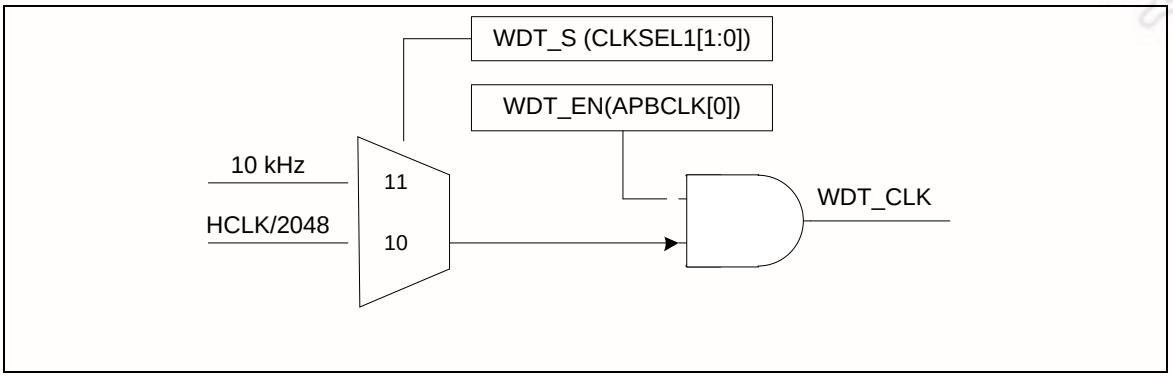


图 5-75 看门狗定时时钟控制

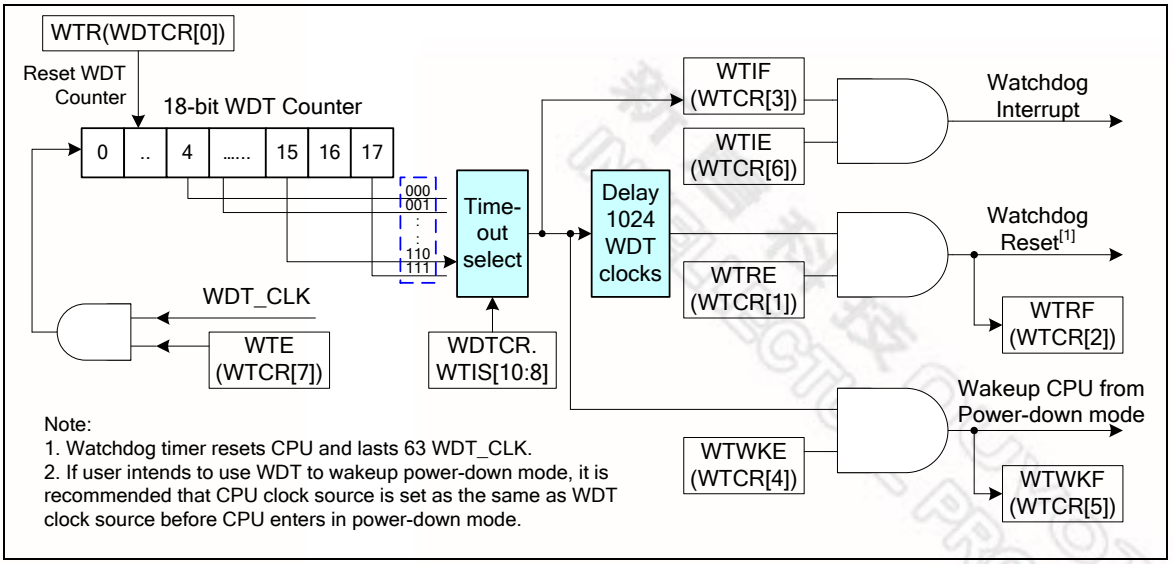


图 5-76 看门狗定时器框图

5.12.4 功能描述

看门狗定时器包含一个18位的自动运行的计数器，超时间隔可编程。**Error! Reference source not found.** 为看门狗超时间隔选择，**Error! Reference source not found.** 为看门狗超时间隔与复位信号的时序

设置 WTE(WDTCR[7]) 为1使能看门狗功能， WDT 计数器开始计数。当计数器达到选择的定时溢出间隔(WTIS设定)时，看门狗定时器中断标志 WTIF（WTCR[3]） 将被立即置位，并请求 WDT 中断（如果看门狗定时器中断使能位 WTIE 置位），同时启动一个指定的WDT复位延迟时间(WTCRALT[1:0]) 。用户必须在指定的复位延迟时间内设置 WTR (WDTCR[0])（看门狗定时器复位）为1，复位18位 WDT 计数器，防止芯片复位。WTR 位在 WDT 计数器复位后自动由硬件清零。

通过设置 WTIS (WDTCR[10:8]) 有8个定时溢出间隔可选择。如果在指定复位延迟时间终止后，WDT 计数器没有被清零，如果WTRE(WTCR[2] WDT复位使能)使能，看门狗定时器将设置看门狗定时器复位标志 (WTRF) 为1，并立即复位芯片。这个复位将持续 63个 WDT 时钟周期 (T_{RST})，然后芯片重启，并从复位向量 (0x0000_0000) 处执行程序。WTRF 不会被看门狗复位清零。用户可用软件检查 WTRF位 识别复位源。

WDT 还提供唤醒功能，当芯片在掉电状态或者空闲状态时，看门狗定时器唤醒功能使能位 (WDTR[4]) 为1，当WTIE位（WTCR[6]WDT中断使能）使能并且WTIF(WTCR[3]WDT中断标志)置为1时，系统就可以被唤醒

WTIS	超时间隔周期 T_{TIS}	延迟复位时间 T_{RSTD}
000	$2^4 * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
001	$2^6 * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
010	$2^8 * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
011	$2^{10} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
100	$2^{12} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
101	$2^{14} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
110	$2^{16} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
111	$2^{18} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}

表5-10 看门狗超时间隔选择

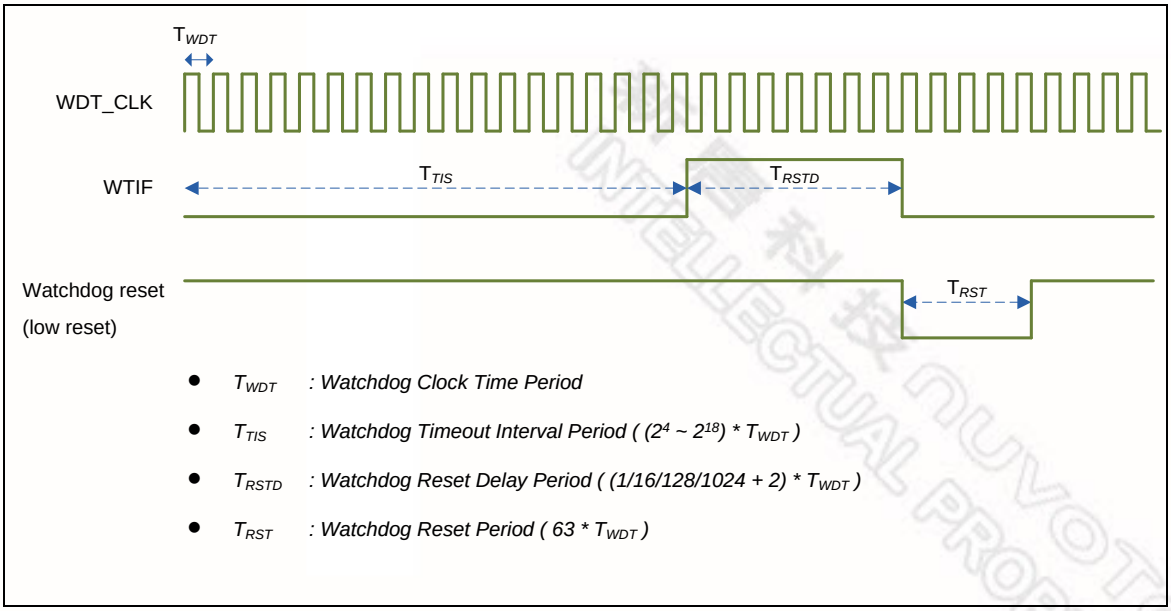


图 5-77 中断和复位信号时序

5.12.5 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
WDT 基地址: **WDT_BA = 0x4000_4000**				
WTCR	WDT_BA+0x00	R/W	看门狗定时器控制寄存器	0x0000_0700
WTCRALT	WDT_BA+0x04	R/W	看门狗定时器控制寄存器2	0x0000_0000

5.12.6 寄存器描述

看门狗定时器控制寄存器(WTCR)

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

注: 该寄存器所有位是写保护的，修改该位时，需要依次向0x5000_0100写入”59h”, “16h”, “88h”, 参考寄存器REGWRPROT,地址GCR_BA + 0x100.

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

Bits	描述	
[31]	DBGACK_WDT	**仿真器（ICE）调试模式响应禁止**(写保护位) 0 = 仿真器调试模式响应影响看门狗定时器计数. 当调试器调试模式响应时，看门狗定时器计数器将被固定住. 1 = 仿真器调试模式响应禁止 无论调试器调试模式响应与否，看门狗定时器计数器将持续计数下去
[30:11]	保留	保留
[10:8]	WTIS	**看门狗定时器间隔选择**(写保护位) 这3位选择看门狗定时器的定时溢出间隔.

WTIS	Timeout Interval Selection	Interrupt Period	WTR Timeout Interval (WDT_CLK=10 KHz)
000	$2^4 * T_{WDT}$	$(2^4 + 1024) * T_{WDT}$	1.6 ms ~ 104 ms
001	$2^6 * T_{WDT}$	$(2^6 + 1024) * T_{WDT}$	6.4 ms ~ 108.8 ms
010	$2^8 * T_{WDT}$	$(2^8 + 1024) * T_{WDT}$	25.6 ms ~ 128 ms
011	$2^{10} * T_{WDT}$	$(2^{10} + 1024) * T_{WDT}$	102.4 ms ~ 204.8 ms
100	$2^{12} * T_{WDT}$	$(2^{12} + 1024) * T_{WDT}$	409.6 ms ~ 512 ms

		101 | 2^{14} * T_{WDT} | (2^{14} + 1024) * T_{WDT} | 1.6384 s ~ 1.7408 s 110 | 2^{16} * T_{WDT} | (2^{16} + 1024) * T_{WDT} | 6.5536 s ~ 6.656 s 111 | 2^{18} * T_{WDT} | (2^{18} + 1024) * T_{WDT} | 26.2144 s ~ 26.3168 s
[7]	**WTE**	**看门狗定时器使能**(写保护位) 0 = 禁用看门狗定时器功能(该动作复位内部计数器) 1 = 使能看门狗定时器
[6]	**WTIE**	**看门狗定时器中断使能**(写保护位) 0 = 禁用看门狗定时器中断 1 = 使能看门狗定时器中断
[5]	**WTWKF**	**看门狗定时器唤醒标志** 如果看门狗定时器导致芯片从掉电模式下唤醒，该位将被置高。必须由软件向该位写1清零 0 = 看门狗定时器没有导致芯片唤醒. 1 = 看门狗超时将芯片由空闲或掉电模式唤醒.
[4]	**WTWKE**	**看门狗定时器唤醒功能使能位**(写保护位) 0 = 禁用看门狗唤醒芯片功能. 1 = 使能看门狗唤醒芯片功能. 注：只有当看门狗时钟源选择内部10K时，才可以将芯片唤醒
[3]	**WTIF**	**看门狗定时器中断标志** 如果看门狗定时器中断使能，该位由硬件置位表示看门狗定时器发生中断，如果没有使能看门狗定时器中断，该位表示看门狗已超时. 0= 没有发生看门狗定时器中断或看门狗没有超时 1= 发生看门狗定时器中断或看门狗已超时 **注**: 写1到该位，清零.
[2]	**WTRF**	**看门狗定时器复位标志** 当看门狗定时器溢出引发复位，该位由硬件置位，通过读取该位可以确认复位是否由看门狗引起。该位通过软件写1清除，如果**WTRE**禁止，看门狗定时器溢出对该位无影响。 0 = 复位不是由看门狗定时器导致的 1 = 看门狗定时器引发复位 **注**: 该位只读，由软件写1清零.
[1]	**WTRE**	**看门狗定时器复位使能**(写保护位) 设定该位使能看门狗定时器复位功能。 0 = 禁用看门狗定时器复位功能 1 = 使能看门狗定时器复位功能
[0]	**WTR**	**清看门狗定时器**(写保护位)

		置位该位清看门狗定时器. 0 = 无动作 1 = 复位看门狗定时器 **注**: 几个时钟周期后该位由硬件自动清零

看门狗定时器控制寄存器2 (WTCRALT)

寄存器	偏移量	R/W	描述	复位后的值
WTCRALT	WDT_BA+0x04	R/W	看门狗定时器控制寄存器2	0x0000_0000

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

Bits	描述	
[31:2]	**保留**	保留
[1:0]	**WTRDSEL**	**看门狗定时器复位延迟选择 (写保护位)** 当看门狗超时发生时，软件有一段时间称为看门狗复位延迟时间来清除看门狗定时器防止看门狗复位发生。针对不同的看门狗超时周期，软件可以选择一个合适的看门狗复位延迟值. 这些比特是写保护的，要写这些位需要向地址0x5000_0100依次写入“59h”, “16h”, “88h”来解锁寄存器保护。参考寄存器 REGWRPROT，在地址 GCR_BA+0x100. 00 = 看门狗超时后延迟 1024 个看门狗时钟周期再复位 01 =看门狗超时后延迟128个看门狗时钟周期再复位 10 =看门狗超时后延迟16个看门狗时钟周期再复位 11 =看门狗超时后延迟1个看门狗时钟周期再复位 如果看门狗复位发生，这个寄存器将被复位

5.13 窗看门狗定时器 (WWDT)

5.13.1 概述

窗 看门狗定时器的目的是在指定的窗周期内实现系统复位，防止软件进入不可控状态.

5.13.2 特性

- 6-bit 下数计数器(WWDTVAL[5:0]) 和 6-bit 比较值(WWDTCR[21:16] – WINCMP)使 窗 周期更有弹性
- WWDT 时钟预分频可以选择，最大11位(WWDTCR[11:8] – PERIODSEL)，使 WWDT 超时间隔可变

5.13.3 方块图

窗 看门狗定时器方块图如下所示.

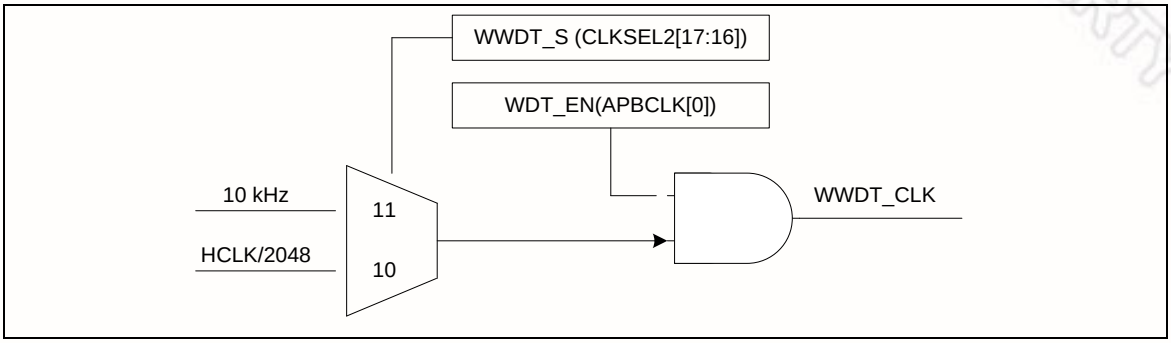


图 5-78 窗 看门狗定时器时钟控制

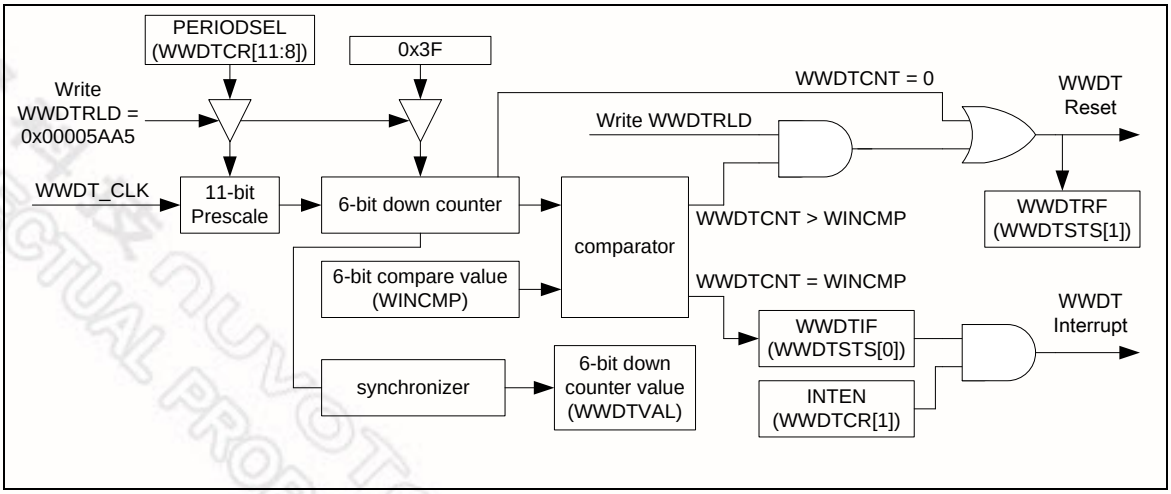


图 5-79 窗 看门狗定时器方块图

5.13.4 功能描述

窗 看门狗定时器包含一个6-bit 下数计数器，预分频可编程以定义不同的超时间隔.

6-bit 窗看门狗定时器的时钟源基于系统时钟除以2048 或者内部10 kHz 振荡器，支持一个11比特的预分频器. 11-bit 预分频器由寄存器 PERIODSEL (WWDTCR[11:8]) 控制。PERIODSEL 和预分频器的关系如表 5-11所示.

PERIODSEL	预分频值	超时周期	超时间隔 (WWDT_CLK=10 kHz)
0000	1	1 * 64 * T_{WWDT}	6.4 ms
0001	2	2 * 64 * T_{WWDT}	12.8 ms
0010	4	4 * 64 * T_{WWDT}	25.6 ms
0011	8	8 * 64 * T_{WWDT}	51.2 ms
0100	16	16 * 64 * T_{WWDT}	102.4 ms
0101	32	32 * 64 * T_{WWDT}	204.8 ms
0110	64	64 * 64 * T_{WWDT}	409.6 ms
0111	128	128 * 64 * T_{WWDT}	819.2 ms
1000	192	192 * 64 * T_{WWDT}	1.2288 s
1001	256	256 * 64 * T_{WWDT}	1.6384 s
1010	384	384 * 64 * T_{WWDT}	2.4576 s
1011	512	512 * 64 * T_{WWDT}	3.2768 s
1100	768	768 * 64 * T_{WWDT}	4.9152 s
1101	1024	1024 * 64 * T_{WWDT}	6.5536 s
1110	1536	1536 * 64 * T_{WWDT}	9.8304 s
1111	2048	2048 * 64 * T_{WWDT}	13.1072 s

表 5-11 窗看门狗预分频选择

芯片上电或者复位以后，窗看门狗定时器由软件通过设定WWDTEN (WWDTCR[0]) 为 1使能. 当窗看门狗定时器被使能时, 下数计数器从0x3F开始计数，并且不能由软件停止.

WWDT 计数器下数期间，如果WWDT计数器的值等于窗看门狗比较寄存器WINCMP (WWDTCR[21:16])的值，WWDTIF(WWDTSR[0] WWDT比较匹配中断标志)将被置为1，WWDT中断将发生（如果WWDTIE(WWDTCR[1]) 等于 1）。

如果WWDT计数器的值等于0，WWDT 复位将发生. WWDT 计数器的值下数到0之前, 软件可以写特定值 (0x00005AA5) 到寄存器WWDTRLD 来重载0x3F 到 WWDT 计数器以防止 WWDT 复位发生。重载行为只有当WWDT计数器的值等于或者小于WINCMP的值才会有效. 如果WWDT计数器的值大于WINCMP，此时软件写0x00005AA5到WWDTRLD，的WWDT复位将产生，从而导致芯片复位。图 5-80 显示了WWDT的复位和重载行为。

为了避免程序意外关闭窗看门狗计数器，芯片上电或者复位以后，控制寄存器WWDTCR只能写1

次。软件使能WWDTEN以后，除非复位芯片或者重新上电，软件不能再关闭窗看门狗计数器（WWDTEN），改变超时预分频(PERIODSEL)，或者改变窗比较值（WINCMP），否则WWDT将导致系统复位。

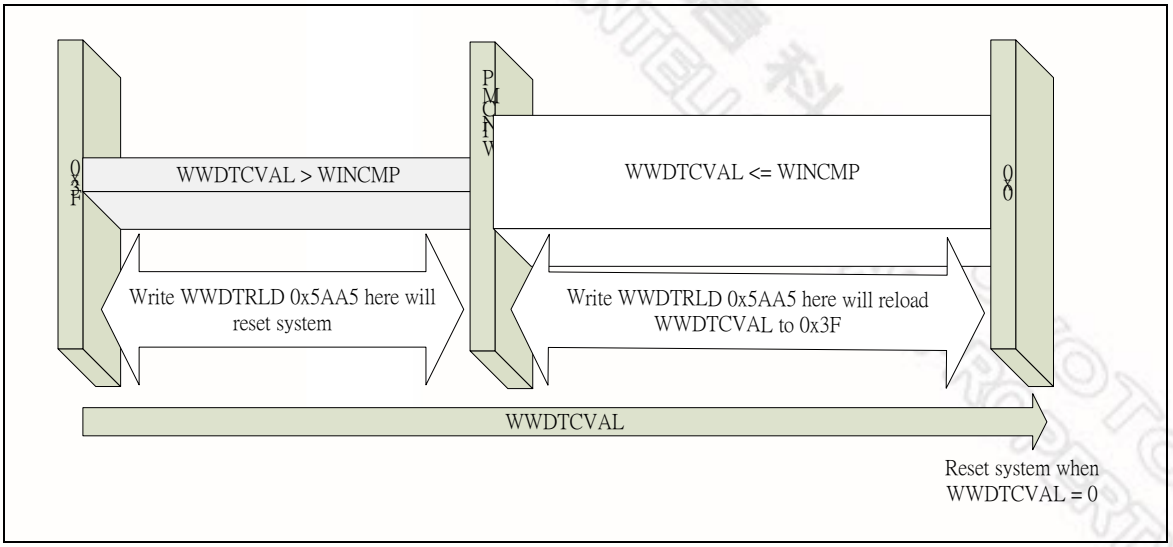


图 5-80 窗看门狗定时器复位和重载行为.

当软件写特定值(0x00005AA5) 到寄存器 WWDTRLD 来重载WWDT 计数器时, 需要 3 个窗看门狗时钟周期来同步重载命令到实际执行。这就意味着，如果软件设置窗看门狗时钟预分频器除以1(WWDTCR[11:8]=0)，比较值WINCMP (WWDTCR[21:16]) 必须大于 2 ，否则在WWDT复位发生之前，软件将来不及重载WWDT 计数器的值.表 5-12 显示了WINCMP的限制.

预分频值	有效的 WINCMP 值
1	0x3 ~ 0x3F
2	0x2 ~ 0x3F
Others	0x0 ~ 0x3F

表 5-12 WINCMP 设定限制

5.13.5 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
WWDT 基地址: **WWDT_BA = 0x4000_4100**				
WWDTRLD	WWDT_BA+0x00	W	窗看门狗定时器重载计数值寄存器	0x0000_0000
WWDTCR	WWDT_BA+0x04	R/W	窗看门狗定时器控制寄存器	0x003F_0800
WWDTSTS	WWDT_BA+0x08	R/W	窗看门狗定时器状态寄存器	0x0000_0000
WWDTCVR	WWDT_BA+0x0C	R	窗看门狗计数器当前值寄存器	0x0000_003F

5.13.6 寄存器描述

窗看门狗定时器重载计数器寄存器 (WWDTRLD)

寄存器	偏移量	R/W	描述	复位后的值
WWDTRLD	WWDT_BA+0x00	W	窗看门狗定时器重载计数器寄存器	0x0000_0000

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

Bits	Description	
[31:0]	**WWDTRLD**	**WWDT 重载计数器寄存器** 写 0x00005AA5 到这个寄存器将导致看门狗定时器计数器的值重新加载成 0x3F. **注:** 只有当WWDT计数器的值在[0, WINCMP]之间时，软件才可以写WWDTRLD. 否则，如果WWDT计数器的值大于WINCMP时，软件写WWDTRLD将导致WWDT 产生 RESET 信号.

窗看门狗定时器控制寄存器 (WWDTCR)

寄存器	偏移量	R/W	描述	复位后的值
WWDTCR	WWDT_BA+0x04	R/W	窗看门狗定时器控制寄存器	0x003F_0800

注: T芯片上电以后这个寄存器只能被写一次.

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

Bits	描述	
[31]	**DBGACK_WWDT**	**ICE 调试模式响应关闭** 0 = 系统进入调试模式时，WWDT 计数器停止计数. 1 = 即使系统进入调试模式，WWDT 仍保持计数.
[30:22]	**保留**	保留
[21:16]	**WINCMP**	**WWDT 窗比较寄存器** 设置这个寄存器来调整有效的重载窗口。 **注:** 当WWDT计数器的值在[0，WINCMP]之间时，软件才可以写WWDTRLD使WWDT计数器重新计数。否则，如果WWDT计数器的值大于WWCMP，此时软件写WWDTRLD将导致WWDT发出RESET 信号.
[15:12]	**保留**	保留
[11:8]	**PERIODSEL**	**WWDT 预分频周期选择** 这4个比特用来选择WWDT计数器预分频的值.

PERIOD SEL	预分频值	超时周期	超时间隔 (WWDT_CLK = 10 kHz)
0000	1	1 * 64 * T_{WWDT}	6.4 ms
0001	2	2 * 64 * T_{WWDT}	12.8 ms
0010	4	4 * 64 * T_{WWDT}	25.6 ms
0011	8	8 * 64 * T_{WWDT}	51.2 ms
0100	16	16 * 64 * T_{WWDT}	102.4 ms
0101	32	32 * 64 * T_{WWDT}	204.8 ms

		0110	64	64 * 64 * T_{WWDT}	409.6 ms
		0111	128	128 * 64 * T_{WWDT}	819.2 ms
		1000	192	192 * 64 * T_{WWDT}	1.2288 s
		1001	256	256 * 64 * T_{WWDT}	1.6384 s
		1010	384	384 * 64 * T_{WWDT}	2.4576 s
		1011	512	512 * 64 * T_{WWDT}	3.2768 s
		1100	768	768 * 64 * T_{WWDT}	4.9152 s
		1101	1024	1024 * 64 * T_{WWDT}	6.5536 s
		1110	1536	1536 * 64 * T_{WWDT}	9.8304 s
		1111	2048	2048 * 64 * T_{WWDT}	13.1072 s
[7:2]	**保留**	保留			
[1]	**WWDTIE**	**WWDT 中断使能** 设定这个比特可以使能窗看门狗定时器中断. 0 = 禁止看门狗定时器中断功能. 1 = 使能看门狗定时器中断功能.			
[0]	**WWDTEN**	**WWDT 使能** 设定这个比特可以使能 窗看门狗定时器. 0 = 禁止窗看门狗定时器. 1 =使能窗看门狗定时器.			

窗看门狗定时器状态寄存器 (WWDTSR)

寄存器	偏移量	R/W	描述	复位后的值
WWDTSR	WWDT_BA+0x08	R/W	窗看门狗定时器状态寄存器	0x0000_0000

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

Bits	描述	
[31:2]	**保留**	保留
[1]	**WWDTRF**	**WWDT 复位标志** 当 WWDT 计数器的值下数到0或者在WWDT计数器的值大于WINCMP的值时写WWDTRLD，芯片将被复位，这个比特将被硬件置为1.软件写1清0.
[0]	**WWDTIF**	**WWDT 比较匹配中断标志** 当 WWCMP 的值与 WWDT 计数器的值匹配时，这个比特将被置1。软件写1清0.

窗看门狗定时器计数器当前值寄存器 (WWDTCVR)

寄存器	偏移量	R/W	描述	复位后的值
WWDTCVR	WWDT_BA+0x0C	R	窗看门狗定时器计数器当前值	0x0000_003F

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

Bits	描述	
[31:6]	**保留**	保留
[5:0]	**WWDTCVAL**	**WWDT 计数器的值** 这个寄存器反应了看门狗计数器的值.

5.14 UART接口控制器 (UART)

该芯片提供2个UART通道。UART控制器UART0与UART1支持流控制.

5.14.1 概述

通用异步收/发器(UART) 在从外设收到数据的时候执行串到并的转换，从CPU发送数据的时候执行并到串的转换. UART控制器也支持IrDA SIR 功能和RS-485模式功能。每个UART通道支持6种中断：包括发送FIFO 空中断(INT_THRE), 接收阈值到达中断(INT_RDA), 线状态中断 (校验错误,帧格式錯誤 或者break 中断) (INT_RLS) ，接收超时中断(INT_TOUT), MODEM/唤醒 状态中断(INT_MODEM) ，和缓冲错误中斷(INT_BUF_ERR). UART0的中断号是12（中断向量为28），UART1的中断号是13（中断向量为29）。参考NVIC章节对系统中断映射的描述.

UART0 和UART1都内嵌一个16-byte 发送FIFO (TX_FIFO) 和 16-byte 接收 FIFO (RX_FIFO) 来降低CPU的中断数量。在操作过程中CPU可以随时读UART的状态. 报告的状态信息包括正在被UART执行的传输操作的类型和条件, 也包括4 种可能接收时发生的错误条件(校验错误, 帧错误, break中断和缓冲错误). UART包括一个可编程的波特率发生器，它可以将输入时钟除以一个除数来得到收发器需要的时钟. 波特率公式是 **Baud Rate = UART_CLK / M * [BRD + 2].** 其中M和BRD在波特率分频寄存器UA_BAUD中定义，见表 5-13.

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 UART 波特率公式

System Clock = 内部 22.1184 MHz						
波特率	Mode 0		Mode 1		Mode 2	
	Parameter	Register	Parameter	Register	Parameter	Register
921600	x	x	A=0,B=11	0x2B00_0000	A=22	0x3000_0016
460800	A=1	0x0000_0001	A=1,B=15 A=2,B=11	0x2F00_0001 0x2B00_0002	A=46	0x3000_002E
230400	A=4	0x0000_0004	A=4,B=15 A=6,B=11	0x2F00_0004 0x2B00_0006	A=94	0x3000_005E
115200	A=10	0x0000_000A	A=10,B=15 A=14,B=11	0x2F00_000A 0x2B00_000E	A=190	0x3000_00BE
57600	A=22	0x0000_0016	A=22,B=15 A=30,B=11	0x2F00_0016 0x2B00_001E	A=382	0x3000_017E
38400	A=34	0x0000_0022	A=62,B=8 A=46,B=11 A=34,B=15	0x2800_003E 0x2B00_002E 0x2F00_0022	A=574	0x3000_023E

19200	A=70	0x0000_0046	A=126,B=8 A=94,B=11 A=70,B=15	0x2800_007E 0x2B00_005E 0x2F00_0046	A=1150	0x3000_047E
9600	A=142	0x0000_008E	A=254,B=8 A=190,B=11 A=142,B=15	0x2800_00FE 0x2B00_00BE 0x2F00_008E	A=2302	0x3000_08FE
4800	A=286	0x0000_011E	A=510,B=8 A=382,B=11 A=286,B=15	0x2800_01FE 0x2B00_017E 0x2F00_011E	A=4606	0x3000_11FE

表 5-14 UART 波特率设置表

UART0与UART1 控制器用2 种low-level 信号(型号的有效电平可以通过UA_MSR和UA_MCR寄存器配置)支持自动流控制功能, /CTS (clear-to-send)和 /RTS (request-to-send)用来控制UART和外设(例如：Modem)之间的数据传输。当使能自动流控时，UART将禁止接收数据直到 UART向外部发送/RTS信号. 当 Rx FIFO 接收到的字节数和RTS_TRI_LEV (UA_FCR [19:16])的值相等时, /RTS信号无效.当UART 控制器侦测到来自外设的 /CTS信号时，向外设发送数据. 如果 /CTS 未被探测到有效, UART 将不向外发送数据.

UART 控制器提供 串行 IrDA (SIR, 串行红外) 功能 (用户需设定 UA_FUN_SEL[IrDA_EN] 使能 IrDA 功能). SIR 定义短程红外异步串行传输模式，该模式有1 个起始位, 8个 数据位, 和1个 停止位. 最大数据速率 为 115.2 Kbps (半双工). IrDA SIR 包括 一个IrDA SIR 协议编码/解码器. IrDA SIR 协议只是半双工协议. 不能同时传输和接收数据. IrDA SIR 物理层规定在传输和接收之间至少10ms 传输延时. **该特性必须由软件执行**.

NuMicro™ NUC123系列，UART控制器的另一个可选的功能是RS-485 9位模式, 由RTS脚控制收/发方向或由软件编程(PB.2 for RTS0 和 PB.6 for RTS1) 执行该功能。RS-485 模式通过设置寄存器UA_FUN_SEL来选择。RS-485驱动器的使能由RTS控制信号实现控制。 在 RS-485 模式, RX和TX的许多特性与UART相同.

5.14.2 特征

- 全双工，异步通信
- 独立接收 / 发送 16 字节FIFO，用于装载数据
- 支持硬件自动流控/流控功能 (CTS, RTS) 和可编程的流控触发电平
- 可编程的接收缓冲触发阈值
- 每个通道独立的可编程的波特率发生器
- 支持CTS唤醒功能
- 支持8位接收缓冲超时检测功能
- UART0/UART1 可以采用DMA 控制器
- 在上一次的停止位与下一次的开始位之间通过设置寄存器UA_TOR [DLY]可编程接收数据延迟时间
- 支持break 错误, 帧错误, 校验错误 和 接收 / 发送缓冲溢出检测功能
- 完全可编程的串口特性
  - 数据位可编程为5, 6, 7, 8位
  - 校验位可编程, 偶校验，奇校验，无校验 或 stick parity 产生和侦测
  - 停止位可编程为 1, 1.5, 或 2 位
- 支持 IrDA SIR功能
  - 普通模式下支持 3/16比特时间
- 支持 RS-485功能
  - 支持 RS-485 9bit 模式
  - RTS引脚支持硬件或软件方向控制

5.14.3 框图

UART时钟控制和框图见图 5-72 和 图 5-73.

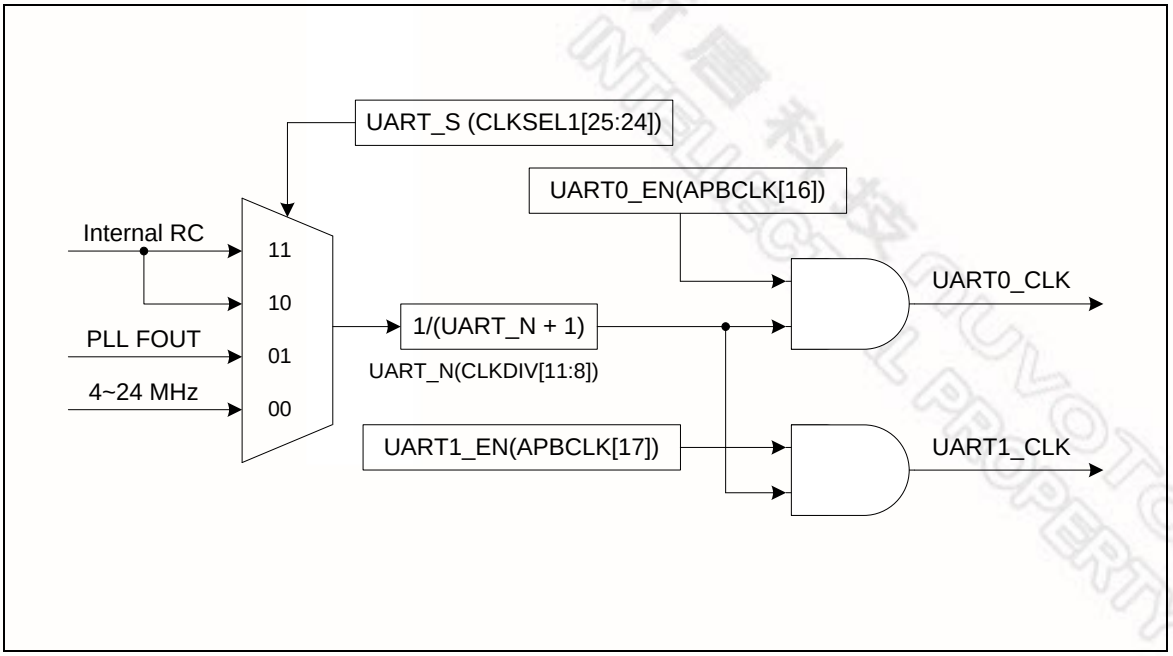


图 5-81 UART 时钟控制图

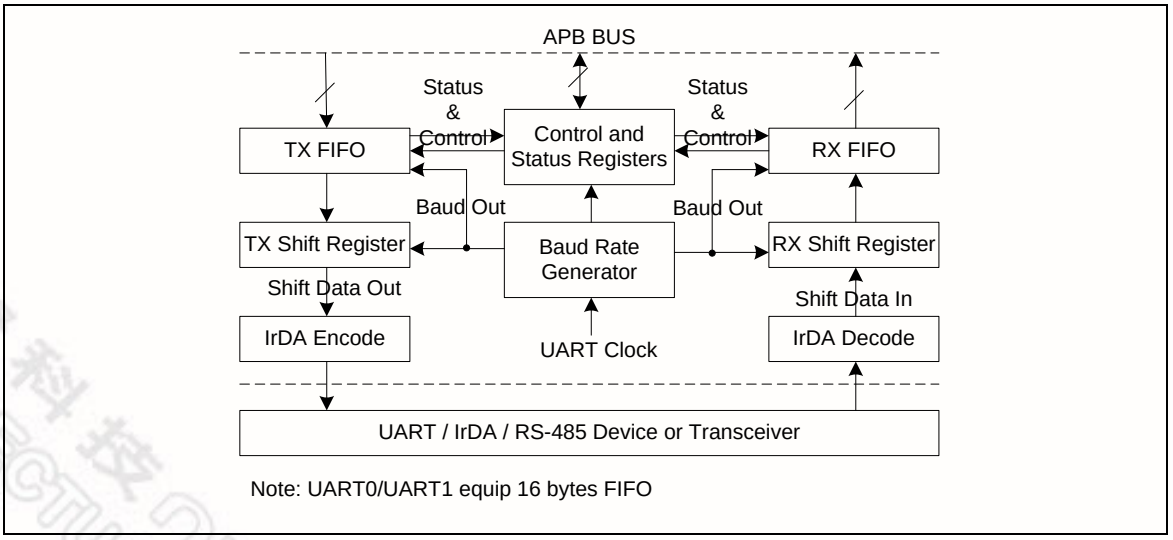


图 5-82 UART 框图

TX_FIFO

发送用一个16个字节的FIFO做缓存来降低CPU的中断数量.

RX_FIFO

接收用一个16个字节(每个字节加3个比特的错误指示比特)的FIFO做缓存来降低CPU的中断数量.

TX shift Register
移位发送的数据串行输出.

RX shift Register
串行移位接收到的数据.

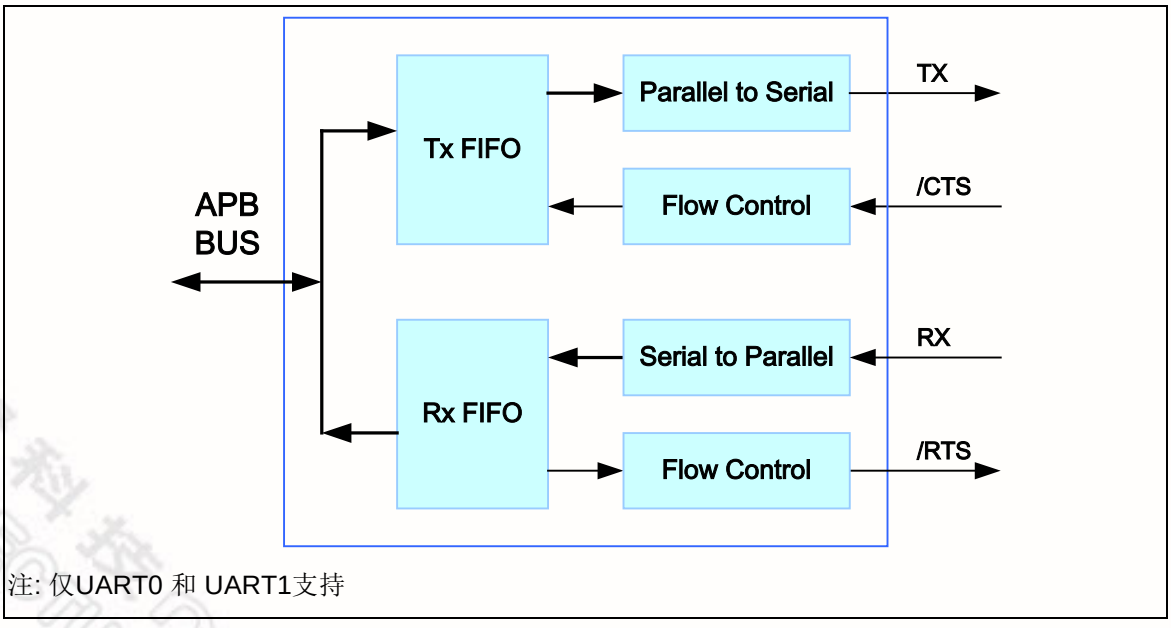
波特率发生器
将外部时钟除以一个除数来产生期望的波特率时钟, 参考波特率方程.

IrDA 编码
IrDA 编码控制模块.

IrDA 解码
IrDA 解码控制模式.

控制和状态寄存器
此域包含一组寄存器, 包括 FIFO 控制寄存器 (UA_FCR), FIFO 状态寄存器(UA_FSR), 和线路控制寄存器 (UA_LCR)应用于传输和接收.超时控制寄存器(UA_TOR) 用于指明超时中断的条件. 该组寄存器也包括中断使能寄存器 (UA_IER) 和中断状态寄存器 (UA_ISR) 来使能或者禁止中断响应并且指明发生的中断. 有六种中断: 发送FIFO为空中断（INT_THRE），接收阈值达到中断（INT_RDA），line 状态中断 (校验错误、帧错误和 break 中断)（INT_RLS）, 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）.

下图为自动流控制框图.



注: 仅UART0 和 UART1支持

图 5-83 自动流控制框图

5.14.4 IrDA 模式

UART支持IrDA SIR (串行红外模式) 传输编码和接收解码, 通过设定**UA_FUN_SEL** 寄存器的**IrDA_EN** 位来选择IrDA 模式.

IrDA 模式下, UA_BAUD[DIV_X_EN] 寄存器是禁止使用的.

波特率 = Clock / (16 * BRD), BRD 为UA_BAUD 寄存器中的波特率分频.

下图为IrDA 控制框图.

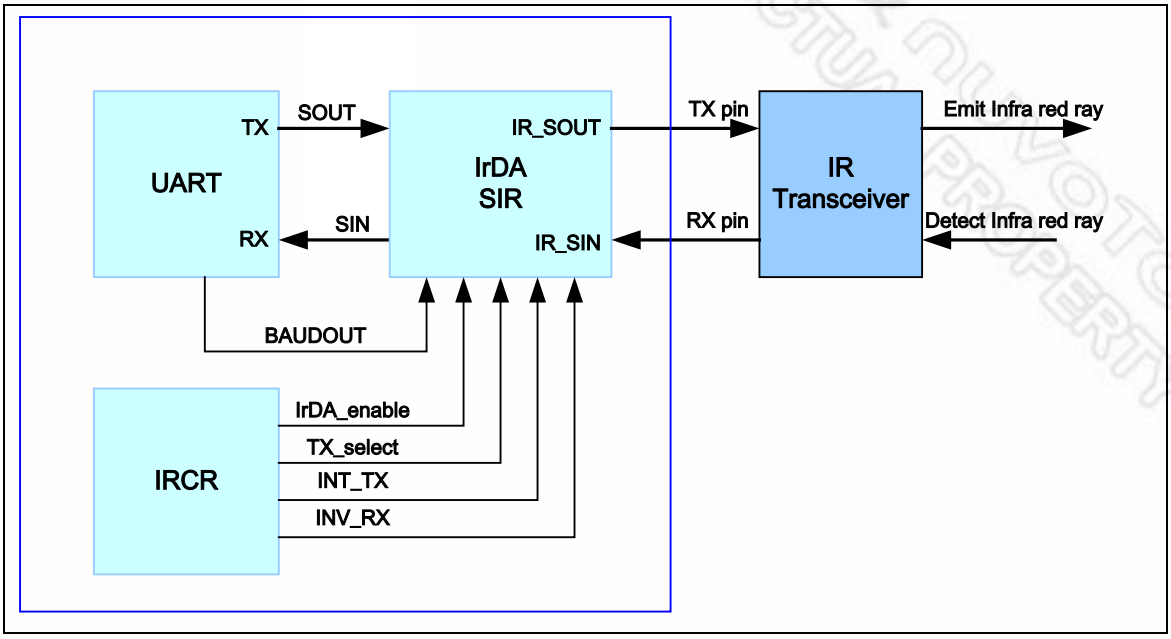


图 5-84 IrDA 框图

5.14.4.1

IrDA SIR 传输编码

IrDA SIR 传输编码调制 Non-Return-to Zero (NRZ) 位流从 UART 输出. IrDA SIR 物理层指定使用归零调制编码，反转（RZI）调制机制用一个红外光脉冲代表逻辑0. 被调制的脉冲输出到外部输出驱动器和红外发射二极管.

5.14.4.2

在正常模式下, 传输脉冲的宽度为 3/16 波特率周期.

IrDA SIR接收解码

SIR接收解码器对来自红外接收器的归零位比特流进行解调，输出NRZ串行比特流到UART。在空闲状态里，解码器输入通常是高. (因为IRCR bit 6 默认应该是1)

5.14.4.3

当解码器输入为低时，表明接收到一个起始位

IrDA SIR运作

IrDA SIR 编码/解码器提供UART数据流和半双工串行SIR接口 之间的转换. IrDA编码/解码波形图如下:

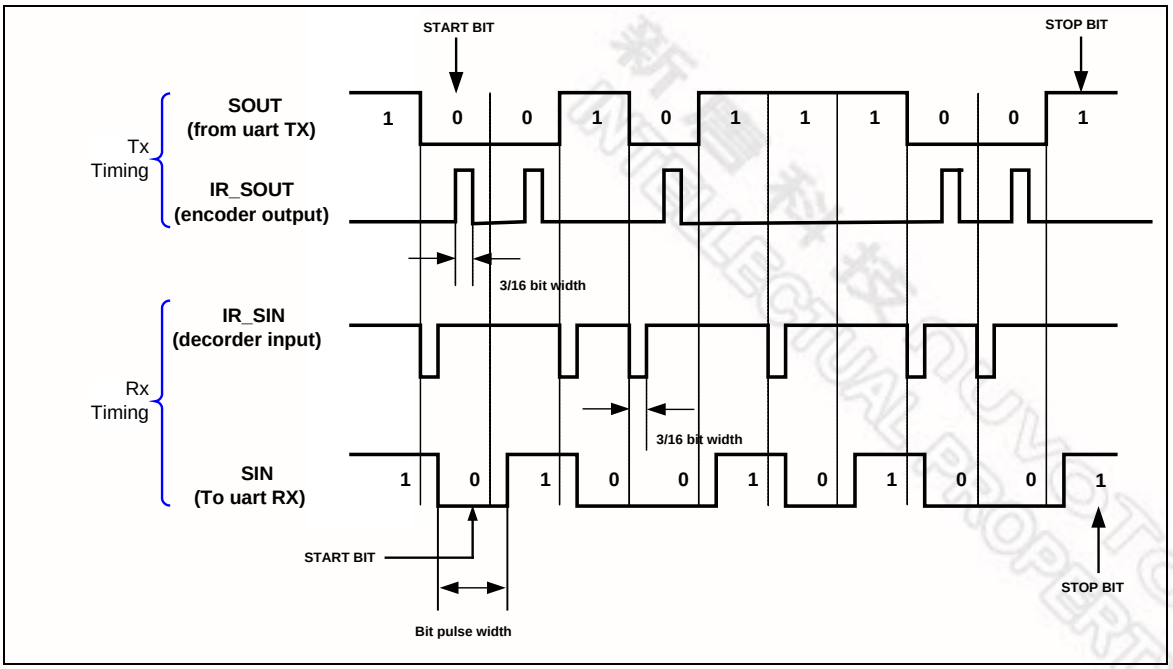


图 5-85 IrDA TX/RX 时序图

5.14.5 RS-485 功能模式

UART支持**RS-485 9 位模式**. 设置UA_FUN_SEL选择RS-485 功能. RS-485 驱动器由异步串口的RTS控制信号提供控制，用来使能RS-485 驱动器。 RS-485 模式下，RX与TX的许多特性与UART一样.

RS-485模式下, 控制器可以配置成RS-485可寻址的从机模式, RS-485主机可通过设置校验位(9^{th}位)为1来标志地址字节。对于数据字节, 校验位为0。设置寄存器UA_LCR 可以控制第9位 (当PBE , EPE 和 SPE都置位时, 第9位发送0 ； 当PBE 和 SPE 置位，EPE为0时, 第9位发送1). 该控制器支持三种操作模式：RS-485 普通模式(NMM), RS-485 自动地址识别模式 (AAD) 和RS-485 自动方向控制模式(AUD), 可通过UA_ALT_CSR寄存器来选择其中一种工作模式, 通过设置UA_TOR [DLY] 可以设置上一个停止与下一个开始位之间的延迟时间..

RS-485 普通多点操作模式(NMM)

RS-485普通多点操作模式, 首先，软件决定在检测到地址字节之前数据是否存储到RX-FIFO，如果软件在检测到地址之前想忽略任何数据，流程为：设置UART_FCR[RX_DIS]，然后使能UA_ALT_CSR[RS485_NMM]，则接收器会忽略数据，直到检测到地址字节(bit9 =1)并且地址字节会被存储到RX-FIFO；如果在检测到地址字节之前，软件想接收任何数据，流程为：关闭UART_FCR [RX_DIS]，使能UA_ALT_CSR[RS485_NMM]，然后接收器将接收任何数据. 如果检测到地址字节(bit9 =1), 会产生一个中断到CPU ，通过设置UA_FCR [RX_DIS]，软件可以决定是否使能或禁止接收器接收后面的数据. 如果使能接收器, 就会接收所有字节数据并存储在RX-FIFO, 如果禁止接收器,所有接收到的字节数据会被忽略，直到检测到下一个地址字节. 如果设置寄存器UA_FCR [RX_DIS]禁止接收器, 当检测到下一个地址字节时, 控制器将清UA_FCR [RX_DIS]且地址字节数据将存储在RX-FIFO中.

RS-485自动地址识别模式(AAD)

RS-485 自动地址识别模式下，接收器在检测到地址字节（bit9=1）并且地址字节数据与UA_ALT_CSR[ADDR_MATCH]的值相匹配之前将忽略所有数据. 地址字节数据将被存储在RX-FIFO中. 其后的数据字节(bit9=0)将被接收并存储于RX-FIFO 直到检测到下一个地址字节(bit9=1)并且其与UA_ALT_CSR[ADDR_MATCH] 的值不匹配.

RS-485自动方向模式(AUD)

RS-485控制器的另一个功能是自动方向控制. RS-485 通过异步串口的RTS控制信号实现驱动控制，使能RS-485 驱动器。RTS 线连接到RS-485 驱动器使能脚，设置RTS线为高（逻辑1并且LEV_RTS=0）,使能RS-485 驱动器；设置RTS为低（逻辑0并且LEV_RTS=1），使驱动器进入tri-state状态. 用户通过设置UA_MCR寄存器中的LEV_RTS位改变 RTS 驱动电平.

编程顺序示例:

1. 设置寄存器UA_FUN_SEL中的FUN_SEL位选择RS-485.
2. 设置寄存器UA_FCR 中的RX_DIS 位使能或禁止RS-485 接收器
3. 设置RS-485_NMM 或 RS-485_AAD 模式.
4. 如果选择RS-485_AAD 模式, ADDR_MATCH应该被编程为自动地址匹配值.
5. 若为自动方向模式，设置RS-485_AUD.

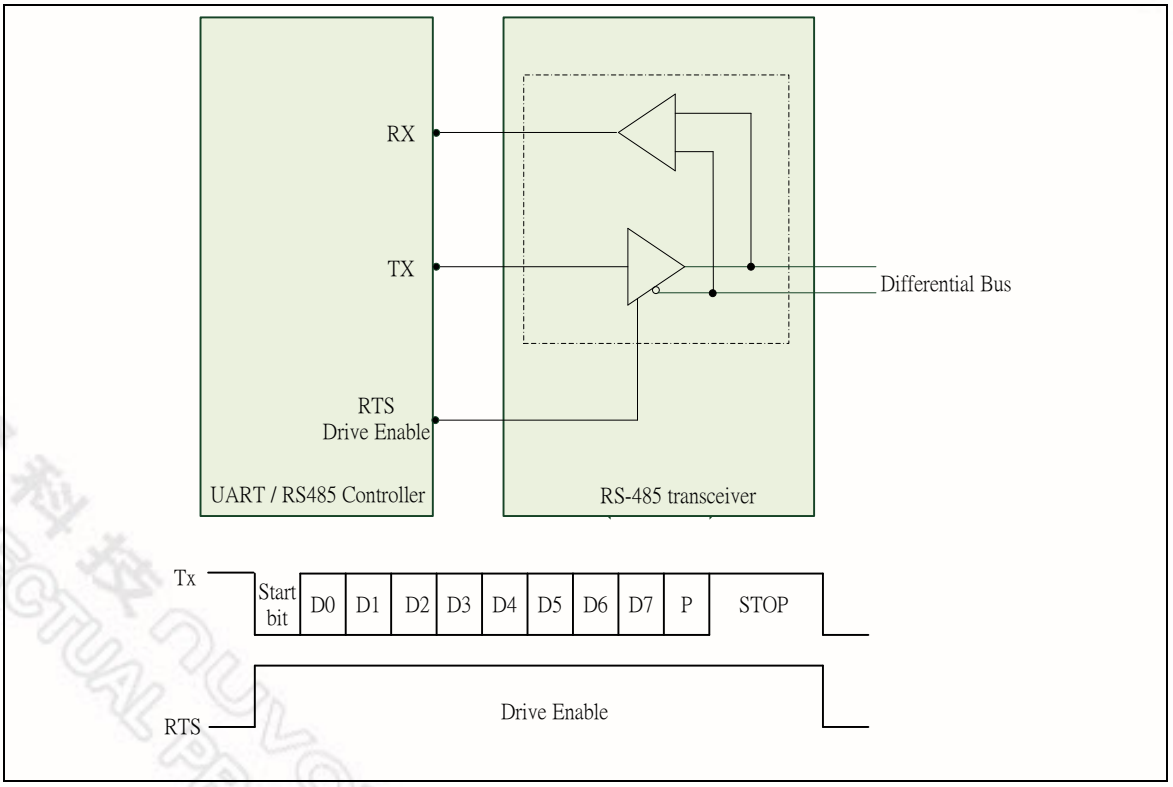


图 5-86 RS-485 帧结构

5.14.6 UART初始化范例

1. 写CLKSEL1寄存器选择UART的时钟源
2. 写APBCLK寄存器使能UART的时钟
3. 写GPx_MFP和ALT_MFP寄存器将相应引脚配置为UART功能
4. 写UA_LCR寄存器配置UART传输数据格式
5. 写UA_FCR寄存器的 RFITL 配置接收FIFO中断极限值
6. 写UA_TOR寄存器配置接收超时
7. 配置UA_BAUD寄存器，配置UART的波特率
8. 写UA_IER寄存器，使能相应UART中断
9. 调用 NVIC_EnableIRQ(UARTx_IRQn) 使能 UART 中断。例如：一般我们会使能RDA_IEN、RTO_IEN和RLS_IEN。这3个中断分别是：接收FIFO达到极限值中断，接收超时中断，和接收线错误中断。而发送缓冲空中断因为只要没有数据要发送就一直会发生，所以一般等有数据要发送的时候才会使能，然后在UART中断处理函数中关闭

5.14.7 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
UART 基地址: **UART0_BA = 0x4005_0000** **UART1_BA = 0x4015_0000**				
UA_RBR x=0,1	UARTx_BA+0x00	R	UART接收数据缓冲寄存器.	Undefined
UA_THR x=0,1	UARTx_BA+0x00	W	UART发送保持寄存器	Undefined
UA_IER x=0,1	UARTx_BA+0x04	R/W	UART中断使能寄存器	0x0000_0000
UA_FCR x=0,1	UARTx_BA+0x08	R/W	UART FIFO控制寄存器	0x0000_0101
UA_LCR x=0,1	UARTx_BA+0x0C	R/W	UART 线控制寄存器	0x0000_0000
UA_MCR x=0,1	UARTx_BA+0x10	R/W	UART Modem控制寄存器	0x0000_0200
UA_MSR x=0,1	UARTx_BA+0x14	R/W	UART Modem 状态寄存器	0x0000_0110
UA_FSR x=0,1	UARTx_BA+0x18	R/W	UART FIFO状态寄存器	0x1040_4000
UA_ISR x=0,1	UARTx_BA+0x1C	R/W	UART 中断状态寄存器	0x0000_0002
UA_TOR x=0,1	UARTx_BA+0x20	R/W	UART 超时寄存器	0x0000_0000
UA_BAUD x=0,1	UARTx_BA+0x24	R/W	UART 波特率分频寄存器	0x0F00_0000
UA_IRCR x=0,1	UARTx_BA+0x28	R/W	UART IrDA 控制寄存器	0x0000_0040

UA_ALT_CSR x=0,1	UARTx_BA+0x2C	R/W	UART 备用控制/状态寄存器	0x0000_0000
UA_FUN_SEL x=0,1	UARTx_BA+0x30	R/W	UART 功能选择寄存器	0x0000_0000

5.14.8 寄存器描述

接收缓冲寄存器 (UA_RBR)

寄存器	偏移量	R/W	描述	复位后的值
UA_RBR x=0,1	UARTx_BA+0x00	R	UART接收数据缓冲寄存器.	Undefined

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

Bits	描述	
[31:8]	**保留**	保留
[7:0]	**RBR**	**接收缓冲寄存器 (只读)** 读此寄存器, UART 将返回一个从 Rx pin接收到的 8-位数据(LSB first).

发送保持寄存器(UA_THR)

寄存器	偏移量	R/W	描述	复位后的值
UA_THR x=0,1	UARTx_BA+0x00	W	UART发送保持寄存器	Undefined

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

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

中断使能寄存器(UA_IER)

寄存器	偏移量	R/W	描述	复位后的值
UA_IER x=0,1	UARTx_BA+0x04	R/W	UART中断使能寄存器	0x0000_0000

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

Bits	描述	
[31:16]	保留	保留
[15]	DMA_RX_EN	**RX DMA 使能** 该位使能或禁止RX DMA 服务. 1 = 使能 RX DMA 0 = 禁用 RX DMA
[14]	DMA_TX_EN	**TX DMA 使能** 该位使能或禁止 TX DMA服务. 1 = 使能 TX DMA 0 = 禁用 TX DMA
[13]	AUTO_CTS_EN	**CTS自动流控制使能** 1 = 使能 CTS **自动流控制**. 0 = 禁用 CTS **自动流控制**. 当 CTS 自动流控使能时，当侦测到CTS信号，UART将向外部设备发送数据(直到CTS 信号有效，UART 才会开始发送数据).
[12]	AUTO_RTS_EN	**RTS自动流控制使能** 1 = 使能 RTS 自动流控制. 0 = 禁用 RTS自动流控制. 当 RTS 自动流控使能时，当 Rx FIFO 中的字节数和

		UA_FCR[RTS_TRI_LEV]相等时, UART 将禁止 RTS 信号.
[11]	**TIME_OUT_EN**	**超时计数器使能** 1 = 使能 超时计数器. 0 = 禁用超时计数器
[10:7]	**保留**	保留
[6]	**WAKE_EN**	**唤醒功能使能** 0 = 禁用UART 唤醒功能 1 =使能唤醒功能, 当系统在掉电模式下, 外部 CTS 的改变将 芯片 从掉电模式下唤醒.
[5]	**BUF_ERR_IEN**	**Buffer Error 中断使能** 1 = 使能INT_BUF_ERR中断 0 = 禁用INT_BUF_ERR中断
[4]	**RTO_IEN**	**RX 超时中断使能** 1 = 使能 INT_TOUT 0 = 屏蔽 INT_TOUT
[3]	**MODEM_IEN**	**Modem中断状态使能** 1 = 使能 INT_MODEM 0 = 屏蔽INT_MODEM
[2]	**RLS_IEN**	**接收线状态中断使能** 1 =使能INT_RLS 0 =屏蔽INT_RLS
[1]	**THRE_IEN**	**发送保持寄存器空中断使能** 1 =使能INT_THRE 0 = 屏蔽INT_THRE
[0]	**RDA_IEN**	**接收数据可得中断使能**. 1 = 使能 INT_RDA 0 = 屏蔽 INT_RDA

FIFO 控制寄存器 (UA_FCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_FCR x=0,1	UARTx_BA+0x08	R/W	UART FIFO控制寄存器	0x0000_0101

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

Bits	描述	
[31:20]	保留	保留
[19:16]	RTS_TRI_LEV	**RTS自动流控触发级别** RTS_TRI_LEV / Trigger Level (Bytes) 0000 / 01 0001 / 04 0010 / 08 0011 / 14 others / 14 **注:** 该寄存器用于自动RTS流控.
[15:9]	保留	保留
[8]	RX_DIS	**接收器禁用寄存器.** 是否禁止接收 1 =禁用接收器，导致停止接收数据 0 = 使能接收器 注: 该位用于RS-485 普通多点模式. 需要在UA_ALT_CSR [RS-485_NMM]之前设置.
[7:4]	RFITL	**RX FIFO 中断 (INT_RDA) 触发级别** 接收FIFO中的字节数等于 RFITL时，RDA_IF 将被置位 (如果UA_IER [RDA_IEN] 使能, 将产生中断).

		RFITL / **INTR_RDA触发级别(Bytes)** 0000 / 01 0001 / 04 0010 / 08 0011 / 14 others / 14
[3]	**保留**	保留
[2]	**TFR**	**发送域软件复位** 当 TX_RST 置位, 发送FIFO中的所有字节和发送内部状态机将被清除. 1 =该位置位将复位发送内部状态机和FIFO指针. 0 = 该位写 0 将无效. **注:** 至少 3 UART时钟周期 自动清 0.
[1]	**RFR**	**接收域软件复位** 当 RX_RST 置位, 接收FIFO中的所有字节和接收内部状态将被清除. 1 = 该位置位将复位接收内部状态机和FIFO指针. 0 = 该位写 0 将无效. **注:**至少 3 UART时钟周期 自动清 0.
[0]	**保留**	保留

Line 控制寄存器(UA_LCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_LCR x=0,1	UARTx_BA+0x0C	R/W	UART Line控制寄存器	0x0000_0000

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

Bits	描述	
[31:7]	保留	保留
[6]	BCB	**钳制控制位** 该位置位時, 串行数据输出 (Tx) 将被迫处于Spacing State (logic 0)状态. 该位仅作用于TX，对传输逻辑不起作用.
[5]	SPE	**Stick 奇偶使能** 1 = 当PBE和EPE都置位時，校验位发送和接收检查都为0，当PBE置位，EPE为0時，校验位发送和接收检查都为1 0 = 禁止 stick 奇偶使能
[4]	EPE	**Even 奇偶使能** 1 = 偶校验，发送和接收检查1的个数为偶数个 0 = 奇校验，发送和接收检查1的个数为奇数个 该位仅当 bit 3 (校验位使能) 位 置位时才有效.
[3]	PBE	**校验位使能** 1 = 每次发送字节时产生校验位，收到字节时检查校验位. 0 = 无校验位.
[2]	NSB	**停止位数目** 1= 当数据位为5位时，产生1.5个停止位；当数据位选择6-, 7- 和 8-位时，.产生2个停止位

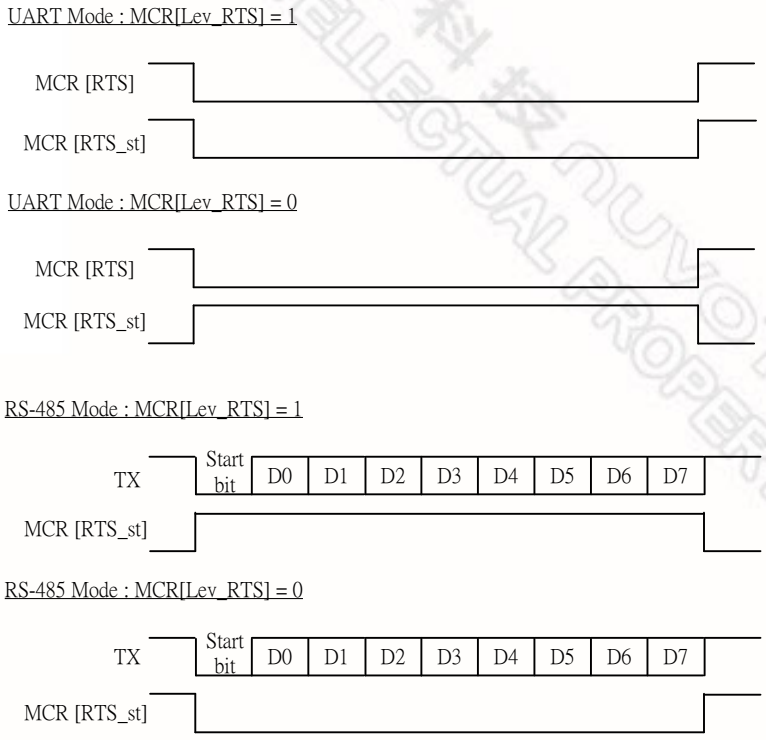
<table>
<tr><td></td><td></td><td>0=传递数据时产生1个停止位1</td></tr>
<tr><td>[1:0]</td><td>WLS</td><td>字长度选择
<table>
<tr><th>WLS[1:0]</th><th>Character length</th></tr>
<tr><td>00</td><td>5 bits</td></tr>
<tr><td>01</td><td>6 bits</td></tr>
<tr><td>10</td><td>7 bits</td></tr>
<tr><td>11</td><td>8 bits</td></tr>
</table></td></tr>
</table>

MODEM 控制寄存器 (UA_MCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_MCR x=0,1	UARTx_BA+0x10	R/W	UART Modem控制寄存器	0x0000_0200

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

Bits	描述	
[31:14]	保留	保留
[13]	RTS_ST	RTS 引脚状态(只读) 该位表示 RTS 引脚的状态.
[12:10]	保留	保留

[9]	**LEV_RTS**	**RTS 触发电平** 该位改变 RTS 触发电平. 1= 高电平触发 0= 低电平触发 
[8:2]	**保留**	保留
[1]	**RTS**	**RTS (Request-To-Send) 信号** 0: 使能 RTS 管脚为 1 (如果 **LEV_RTS** 设为低电平触发). 1: 使能 RTS 管脚为 0 (如果 **LEV_RTS** 设为低电平触发). 0: 使能 RTS 管脚为 0 (如果 **LEV_RTS** 设定高电平触发). 1: 使能 RTS 管脚为 1 (如果 **LEV_RTS** 设定高电平触发).
[0]	**保留**	保留

Modem 状态寄存器(UA_MSR)

寄存器	偏移量	R/W	描述	复位后的值
UA_MSR x=0,1	UARTx_BA+0x14	R/W	UART Modem状态寄存器	0x0000_0000

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

Bits	描述	
[31:9]	保留	保留
[8]	LEV_CTS	**CTS 触发电平** 该位可改变 CTS 触发电平. 1=高电平触发 0=低电平触发
[7:5]	保留	保留
[4]	CTS_ST	**CTS 引脚状况(只读)** 该位表示 CTS 管脚状态
[3:1]	保留	保留
[0]	DCTSF	**侦测 CTS 状态改变标志位(只读)** 只要 CTS 输入状态改变 该位置位, 可向 CPU 产生Modem中断请求(如果使能 UA_IER [MODEM_IEN]). **注:** 该位只读, 可写 '1' 清除.

FIFO 状态寄存器(UA_FSR)

寄存器	偏移量	R/W	描述	复位后的值
UA_FSR x=0,1	UARTx_BA+0x18	R/W	UART FIFO状态寄存器.	0x1040_4000

31	30	29	28	27	26	25	24
保留			TE_FLAG	保留			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
保留	BIF	FEF	PEF	RS485_ADD_DETF	保留		RX_OVER_IF

Bits	描述	
[31:29]	保留	保留
[28]	TE_FLAG	发送空标志位(只读) 当 TX FIFO(UA_THR) 为空并且最后一个字节的STOP 位已经被发送，该位由硬件自动置位. 当 TX FIFO(UA_THR) 不为空或最后一字节的STOP 位未传输，该位由硬件自动清0. 注: 该位只读.
[27:25]	保留	保留
[24]	TX_OVER_IF	发送溢出错误中断标志位(只读) 若 TX FIFO(UA_THR) 满, 再写 UA_THR寄存器 将引起 该位置1. 注: 该位只读, 可通过写'1' 清除该位.
[23]	TX_FULL	发送FIFO满(只读) 该位表示 TX FIFO 是否已满. 当 TX_POINTER 等于16(UART0/UART1)时 该位置位, 反之由硬件清除.
[22]	TX_EMPTY	发送FIFO 为空(只读) 该位表示 TX FIFO 是否为空. 当 TX FIFO 中最后一个字节已经被发送到发送移位寄存器, 硬件置位该位. 当写数据到THR (TX FIFO 非空) 该位清0.

[21:16]	**TX_POINTER**	**发送 FIFO 指针 (只读)** 该位表示TX FIFO 缓冲的指针. 当CPU 写1个字节到 UA_THR时, TX_POINTER 增加 1；当 TX FIFO 中的字节被发送 1 个到 发送移位寄存器时, TX_ POINTER减1.
[15]	**RX_FULL**	**接收FIFO 满 (只读)** 该位表示 接收 FIFO 是否已满. 当 RX_POINTER等于16(UART0/UART1)时 该位置位, 反之由硬件清除.
[14]	**RX_EMPTY**	**接收FIFO 为空(只读)** 该位表示 Rx FIFO是否为空. 当 RX FIFO 中的最后一个字节 被CPU读走, 硬件置位该位. 当 UART 接收到新数据时，该位清0.
[13:8]	**RX_POINTER**	**接收FIFO 指针 (只读)** 该位表示 RX FIFO 缓冲指针. 当UART 从外部设备接收到 1个字节数据时, RX_POINTER增加 1. 当 RX FIFO 被 CPU 读走 1个字节数据时, RX_POINTER减1.
[7]	**保留**	保留
[6]	**BIF**	**钳制中断标志位(只读)** 当RX引脚处于"spacing state" (logic 0)状态的时间比传输一个完整的字还长时(即"start bit" + data bits + parity + stop bits的所有时间)的时间，该位置1。 注：该位只读，软件写“1”清除该位.
[5]	**FEF**	**帧错误标志位(只读)** 若接收字符中无有效"停止位"（也就是说最后的停止位没有检测到，检测到的是0） 该位将置位, CPU 写 1到该位复位. 注: 该位只读，软件写“1”清零
[4]	**PEF**	**校验标志位 (只读)** 若 接收字符中无有效"校验位" 该位将置位, CPU 写 1 到 该位 复位. 注: 该位只读，软件写“1”清零.
[3]	**RS485_ADD_DETF**	**RS-485 地址字节检测标志(只读)** 在RS-485模式下，如果UA_ALT_CSR [RS-485_ADD_EN]使能，接收器检测到任何地址字节(bit9 = ‘1’) ，该位设置为1，CPU写1到该位时复位. 注: 该位用于RS-485功能模式. 注: 该位只读，软件写“1”清零.
[2:1]	**保留**	保留
[0]	**RX_OVER_IF**	**接收溢出错误标志 (只读)** 当 接收 FIFO 溢出时，该位置位. 当接收到的数据大于 RX FIFO(UA_RBR) 的大小16时，该位置位. 注: 该位只读，软件写“1”清零.

中断状态控制寄存器(UA_ISR)

寄存器	偏移量	R/W	描述	复位后的值
UA_ISR x=0,1	UARTx_BA+0x1C	R/W	UART中断状态控制寄存器	0x0000_0002

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

Bits	描述	
[31:30]	**保留**	保留
[29]	**HW_BUF_ERR_INT**	**在DMA 模式下, 缓冲错误中断指示器 (只读)** 当BUF_ERR_IEN 与HW_BUF_ERR_IF都置1时，该位置位. 1 = 在DMA模式下产生buffer error中断 0 = 在DMA模式下没有buffer error 中断
[28]	**HW_TOUT_INT**	**在DMA 模式下, 超时中断指示器(只读)** 当TOUT_IEN 与HW_TOUT_IF都置1时，该位置位. 1 = 在DMA模式下产生超时中断 0 = 在DMA模式下没有超时 中断
[27]	**HW_MODEM_INT**	**在 DMA 模式, MODEM状态中断指示器(只读)** 当MODEM_IEN 和 HW_MODEM_IF都置1时，该位置位. 1 = 在DMA模式下产生Modem 中断 0 = 在DMA模式下没有Modem 中断
[26]	**HW_RLS_INT**	**在DMA 模式, 接收 线状态中断指示器 (只读)** 当RLS_IEN 和 HW_RLS_IF都置1时，该位置位. 1 = 在DMA模式下产生RLS中断 0 = 在DMA模式下没有RLS中断

[25:22]	**保留**	保留
[21]	**HW_BUF_ERR_IF**	**DMA 模式下, 缓冲错误中断标志 (只读)** 当 发送或 接收 FIFO 溢出(TX_OVER_IF 或 RX_OVER_IF 置位) 该位置位. 当 BUF_ERR_IF 置位, 传输可能不正常. 若 UA_IER [BUF_ERR_IEN]使能, 缓冲错误中断产生. **注:** 当TX_OVER_IF和RX_OVER_IF被清除时，该位清空.
[20]	**HW_TOUT_IF**	**DMA 模式下, 超时中断标志 (只读)** 当接收 FIFO不为空，超过TOIC定义的时间，接收 FIFO没有再收到数据，该位置位. 若UA_IER [TOUT_IEN]使能, 中断产生. **注:** 该位只读，用户可读UA_RBR (Rx is in active)清除该位.
[19]	**HW_MODEM_IF**	**DMA 模式下, MODEM中断标志 (只读)** 当 CTS pin 状态(DCTSF=1)改变时 该位置位.若UA_IER [MODEM_IEN]使能, Modem 中断产生. **注:** 该位只读 ，当通过写1将DCTSF 清0 时，该位复位.
[18]	**HW_RLS_IF**	**在 DMA 模式, 接收线 状态标志位(只读)** 当 接收数据有校验错误, 帧错误 或 break error (3位中 BIF, FEF 和 PEF, 至少 1 位置位) 时该位置位. 若UA_IER [RLS_IEN]使能, RLS 中断产生. 注：在RS-485模式，该位包括“接收器检测到地址字节(bit9 = ‘1’) “，就是检测到地址字节时该位也会置位 注**:** 该位只读， BIF, FEF 和 PEF 被清空 该位复位..
[17:15]	**保留**	保留
[13]	**BUF_ERR_INT**	**缓冲错误中断指示器(只读)** 当BUF_ERR_IEN 和 BUF_ERR_IF都置1，该位置位. 1 = 缓冲错误中断已发生 0 = 没有缓冲错误中断
[12]	**TOUT_INT**	**超时中断指示器(只读)** 当TOUT_IEN 和 TOUT_IF都置1，该位置位. 1 = 产生超时中断 0 = 没有超时 中断
[11]	**MODEM_INT**	**MODEM 状态中断指示器(只读).** 当MODEM_IEN 和 MODEM_IF都置1，该位置位. 1 = 产生Modem中断 0 = 没有Modem中断
[10]	**RLS_INT**	**接收线状态中断指示器 (只读).** 当RLS_IEN 和 RLS_IF都设置为1，该位置位. 1 = 产生RLS 中断 0 = 没有RLS 中断

[9]	**THRE_INT**	**发送保持寄存器 空中断指示器(只读).** 当THRE_IEN 和 THRE_IF都置1，该位置位. 1 =产生THRE 中断 0 = 没有THRE 中断
[8]	**RDA_INT**	**接收数据可得中断指示器(只读).** 当RDA_IEN 和RDA_IF都置1，该位置位. 1 = 产生RDA 中断 0 = 没有RDA 中断
[7:6]	**保留**	保留
[5]	**BUF_ERR_IF**	**缓冲错误中断标志位(只读)** 当TX 或RX FIFO溢出 (TX_OVER_IF 或 RX_OVER_IF置位)时 ,该位置位.当BUF_ERR_IF置位, 传输可能不正常. 若UA_IER [BUF_ERR_IEN] 使能，缓存错误中断产生. **注:** 当TX_OVER_IF 和 RX_OVER_IF 被清除时，该位清除.
[4]	**TOUT_IF**	**超时中断标志位 (只读)** 当接收FIFO不为空时，超过TOIC定义的时间没有再接收到数据，该位置位. 若UA_IER [TOUT_IEN] 使能, 超时中断产生. **注:** 用户可读 UA_RBR (Rx is in active) 清该位.
[3]	**MODEM_IF**	**MODEM中断标志位(只读)** 当 CTS 引脚 状态改变(DCTSF=1) 该位置位. 若UA_IER [MODEM_IEN]使能, Modem 中断产生. **注:** 该位只读，当写 1 到DCTSF清除DCTSF比特时 该位清除.
[2]	**RLS_IF**	**接收线中断标志位 (只读).** 当 接收数据有校验错误, 帧错误 或 break error (BIF, FEF 和 PEF, 3 位中至少1位置位)该位置位.若 UA_IER[RLS_IEN]使能, RLS 中断产生. 注: 在RS-485模式，该位包括“接收器检测到地址字节(bit9 = ‘1’)”，就是说检测到地址该位也会置1 注**:** 该位只读， 当 BIF, FEF 和PEF 清除 该位复位.
[1]	**THRE_IF**	**发送保持寄存器 空中断标志位 (只读).** 当发送 FIFO 最后一个数据传输到发送移位寄存器 该位置位. 若 UA_IER [THRE_IEN]置位, THRE 中断产生. **注:** 该位只读 写数据到 THR (Tx FIFO not empty) 该位清空.
[0]	**RDA_IF**	**接收数据可得中断标志位(只读).** 当接收 FIFO 中的数据个数和 RFITL 相等， RDA_IF 将置位. 若UA_IER [RDA_IEN]使能, RDA 中断产生. **注:** 该位只读 。接收 FIFO 中的数据个数跌落到阈值以下， 该位清除 (RFITL).

UART中断源	中断使能位	中断指示 中断控制	中断标志位	标志位清除
Buffer Error Interrupt INT_BUF_ERR	BUF_ERR_IEN	HW_BUF_ERR_INT	**HW_BUF_ERR_IF** = (TX_OVER_IF or RX_OVER_IF)	Write ‘1’ to TX_OVER_IF/ RX_OVER_IF
RX Timeout Interrupt INT_TOUT	RTO_IEN	HW_TOUT_INT	**HW_TOUT_IF**	Read UA_RBR
Modem Status Interrupt INT_MODEM	MODEM_IEN	HW_MODEM_INT	**HW_MODEM_IF** = (DCTSF)	Write ‘1’ to DCTSF
Receive Line Status Interrupt INT_RLS	RLS_IEN	HW_RLS_INT	**HW_RLS_IF** = (BIF or FEF or PEF or RS- 485_ADD_DETF)	Write ‘1’ to BIF/FEF/PEF/ RS- 485_ADD_DETF
Transmit Holding Register Empty Interrupt INT_THRE	THRE_IEN	HW_THRE_INT	**HW_THRE_IF**	Write UA_THR
Receive Data Available Interrupt INT_RDA	RDA_IEN	HW_RDA_INT	**HW_RDA_IF**	Read UA_RBR

表 5-15 在 DMA 模式下 UART 中断源和标志表

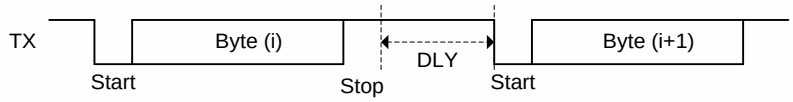
UART 中断源	中断使能位	中断指示 中断控制	中断标志位	标志位清除
Buffer Error Interrupt INT_BUF_ERR	BUF_ERR_IEN	BUF_ERR_INT	**BUF_ERR_IF** = (TX_OVER_IF or RX_OVER_IF)	Write ‘1’ to TX_OVER_IF/ RX_OVER_IF
RX Timeout Interrupt INT_TOUT	RTO_IEN	TOUT_INT	**TOUT_IF**	Read UA_RBR
Modem Status Interrupt INT_MODEM	MODEM_IEN	MODEM_INT	**MODEM_IF** = (DCTSF)	Write ‘1’ to DCTSF
Receive Line Status Interrupt INT_RLS	RLS_IEN	RLS_INT	**RLS_IF** = (BIF or FEF or PEF)	Write ‘1’ to BIF/FEF/PEF
Transmit Holding Register Empty Interrupt INT_THRE	THRE_IEN	THRE_INT	**THRE_IF**	Write UA_THR
Receive Data Available Interrupt INT_RDA	RDA_IEN	RDA_INT	**RDA_IF**	Read UA_RBR

表 5-16 在软件模式下 UART 中断源和标志表

超时寄存器(UA_TOR)

寄存器	偏移量	R/W	描述	复位后的值
UA_TOR x=0,1	UARTx_BA+0x20	R/W	UART时间溢出寄存器	0x0000_0000

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

Bits	描述	
[31:16]	保留	保留
[15:8]	DLY	**发送 延迟时间值** 该位用于编程上一次停止位与下一次起始位之间的传输延迟时间. 
[6:0]	TOIC	**超时中断比较器** 当接收 FIFO 接收到新数据时，超时计数器复位并重新开始计数(计数时钟 = baud rate). 一旦超时计数器 (TOUT_CNT)的值和超时中断比较器(TOIC) 的值相等, 接收超时中断产生 (INT_TOUT)（ 若 UA_IER [RTO_IEN]使能）. 新的数据输入或接收 FIFO为空可清INT_TOUT。为了避免字符接收期间接收超时中断产生，TOIC的值应该设在40 ~ 255之间。例如，如果TOIC等于40，每个UART字符有一个停止位，没有校验位，则4个字符时间之后还没有收到新的数据，超时中断产生

波特率分频寄存器(UA_BAUD)

寄存器	偏移量	R/W	描述	复位后的值
UA_BAUD x=0,1	UARTx_BA+0x24	R/W	UART波特率分频寄存器	0x0F00_0000

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

Bits	描述	
[31:30]	**保留**	保留
[29]	**DIV_X_EN**	**分频 X 使能** BRD = 波特率分频, **波特率= Clock / [M * (BRD + 2)]** ; 默认 M 为 16. 1 = 使能分频 X (公式为 M = X+1, 但是 DIVIDER_X [27:24] must >= 8). 0 = 禁用分频 X (公式为 M = 16) **注**: 在IrDA 模式下 该位禁止.
[28]	**DIV_X_ONE**	**分频X 等于 1** 1 = 除数 M = 1 (公式为 M = 1, 但是BRD [15:0] must >= 3). 0 = 除数 M = X (公式为 M = X+1, 但是 DIVIDER_X[27:24] must >= 8) 参考下表 5-17
[27:24]	**DIVIDER_X**	**分频X** 波特率除数 M = X+1
[23:16]	**保留**	保留
[15:0]	**BRD**	**波特率 分频** 波特率分频位

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

表 5-17 波特率方程表

IrDA 控制寄存器 (IRCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_IRCR x=0,1	UARTx_BA+0x28	R/W	UART IrDA控制寄存器.	0x0000_0040

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

Bits	描述	
[31:7]	**保留**	保留
[6]	**INV_RX**	**INV_RX** 1= Rx 输入信号反转 0= 不反转
[5]	**INV_TX**	**INV_TX** 1= Tx 输出信号反转 0=不反转
[4:2]	**保留**	保留
[1]	**TX_SELECT**	**TX_SELECT** 1: 使能IrDA 发送 0: 使能 IrDA 接收
[0]	**保留**	保留

注：在 IrDA 模式，寄存器 UA_BAUD[DIV_X_EN]必须禁止 (波特率方程必须是Clock / 16 * (BRD)

UART 备用控制/状态寄存器(UA_ALT_CSR)

寄存器	偏移量	R/W	描述	复位后的值
UA_ALT_CSR x=0,1	UARTx_BA+0x2C	R/W	UART备用控制/状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
ADDR_MATCH							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
RS485_ADD_EN	保留				RS485_AUD	RS485_AAD	RS485_NMM
7	6	5	4	3	2	1	0
保留							

Bits	描述	
[31:24]	**ADDR_MATCH**	**地址匹配寄存器** 该位用于RS-485自动地址检测时，存放希望收到的RS-485 地址. 注: 该位用于RS-485自动地址检测模式.
[23:16]	**保留**	保留
[15]	**RS485_ADD_EN**	**RS-485 地址检测使能** 该位用于使能RS-485 地址检测模式. 1 = 使能地址检测模式 0 = 禁用地址检测模式 注: 该位用于RS-485任意 操作模式.
[14:11]	**保留**	保留
[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(AUO) 0 = 禁用RS-485 自动方向操作模式 (AUO) 注: 在RS-485_AAD 或 RS-485_NMM 操作模式有效.
[9]	**RS485_AAD**	**RS-485 自动地址检测操作模式 (AAD)** 1 = 使能 RS-485 自动地址方向操作模式 (AAD) 0 = 禁用 RS-485自动地址方向操作模式 (AAD) 注: 在 RS-485_NMM 操作模式下无效.

[8]	**RS485_NMM**	**RS-485 普通多点操作模式 (NMM)** 1 = 使能RS-485 普通多点操作模式 (NMM) 0 = 禁用 RS-485普通多点操作模式(NMM) 注: 在 RS-485_AAD 操作模式下无效.
[7:0]	**保留**	保留

UART 功能选择寄存器 (UA_FUN_SEL)

寄存器	偏移量	R/W	描述	复位后的值
UA_FUN_SEL x=0,1	UARTx_BA+0x30	R/W	UART功能选择寄存器	0x0000_0000

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

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

5.15 PS/2 设备控制器 (PS2D)

5.15.1 概述

PS/2 设备控制器为PS/2通讯提供基本时序控制. 所有在设备和主机之间的通讯都是通过CLK 和 DATA 引脚控制. 不同于 PS/2 键盘和鼠标设备控制器, 接收/传输代码需要固件进行代码转换成有意义的代码. 在接收到发送请求后启动控制器发送CLK 信号，但是在通信过程中主机拥有最终的控制权。主机发送到设备的数据是在上升沿读取，设备发送到主机的数据在上升沿被读走，设备向主机发送的数据在上升沿之后改变. 16 个字节的 FIFO 用来减少CPU的介入. S/W 可选择 1 ~ 16 字节的连续传输.

5.15.2 特性

- 主机通讯禁止和请求发送侦测
- 接收帧错误侦测
- 可编程1 ~ 16 位传输缓冲 以减少 CPU 干预
- 双数据接收缓冲功能
- S/W 控制总线

5.15.3 系统框图

PS/2 设备驱动器 由APB 接口和用于DATA 和 CLK 的时序逻辑组成.

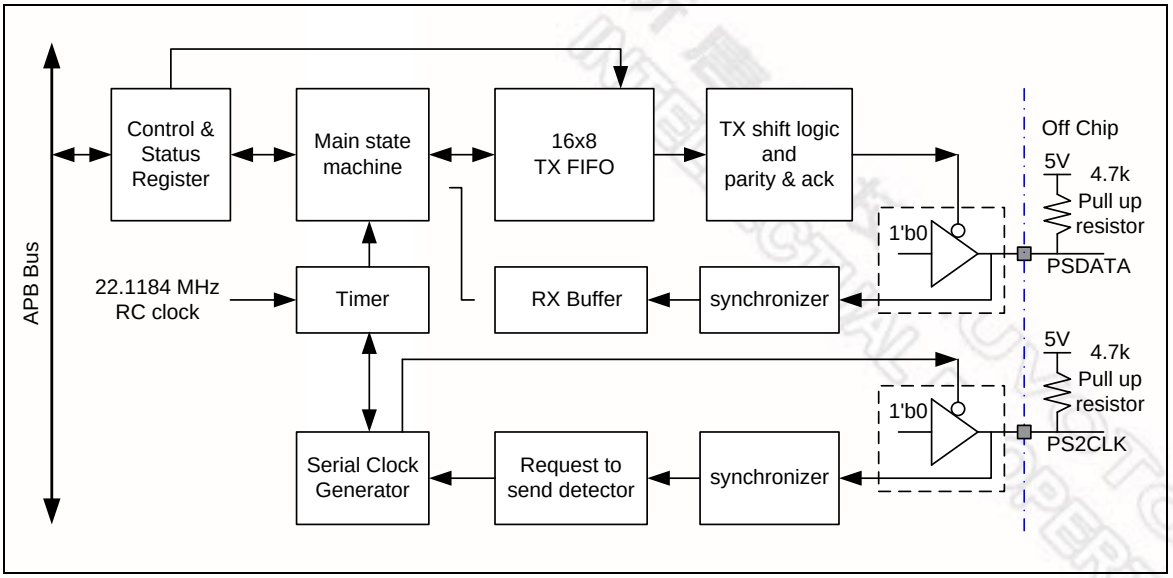


图 5-87 PS/2 设备框图

5.15.4 功能描述

通信

5.15.4.1 PS/2 设备具有双向同步串行协议.当两条线都为高 (open-collector)时，总线为 "空闲" 模式。该状态为设备允许开始 DATA 传输的唯一状态。主机在总线上有最终的控制权 ，并且任何时刻都可以通过下拉CLK 线禁止通讯.

设备负责产生CLK信号。如果主机需要发送 DATA, 必须首先下拉CLK 线为低，禁止从设备进行通讯. 然后主机将DATA 拉低并且释放CLK。这是"请求发送" 状态，通知设备开始发送 CLK 脉冲.

DATA	CLK	总线状态
High	High	空闲
High	Low	禁止通讯
Low	High	主机请求发送

所有数据每次传输 1 字节，每个字节被包含在一个 11-12 位的帧中。如下所示:

- 1个开始位. 总是为 0
- 8个数据位，最低位优先
- 1 个校验位(奇校验)
- 1 个停止位. 该位一直为1
- 1 个应答位 (只是主机~设备通讯)

如果数据位有偶数个1校验位将置1，如果有奇数个1校验位将清0，数据位中的1加上校验位总共1的个数为奇数个，这个可以应用于错误侦测. 设备需要检查该位，如果该位错误，应该返回无效命令的响应.

主机可通过拉底CLK 线 至少100us 来禁止通讯。如果11th 时钟脉冲之前传输被禁止, 设备必须退出当前传输 ，当主机释放CLK时，设备将准备重新传输当前数据。为了有充足的时间让软件解码主机命令，通过设定RXINT位阻塞传输逻辑, S/w 必须清RXINT位来重新开始传输. S/W 可写 CLRFIFO 为 1 来复位 FIFO 指针.

设备向主机传输

设备使用一个 11-位 的帧。 如下:

- 1 个开始位。内容始终为 0
- 8 个数据位. 最低位优先传输
- 1个校验位(奇校验)
- 1个停止位。该位内容总是为1

当CLK为高时，设备写 1 位数据到DATA总线, CLK为低时，数据被主机读走。见图 5-88.

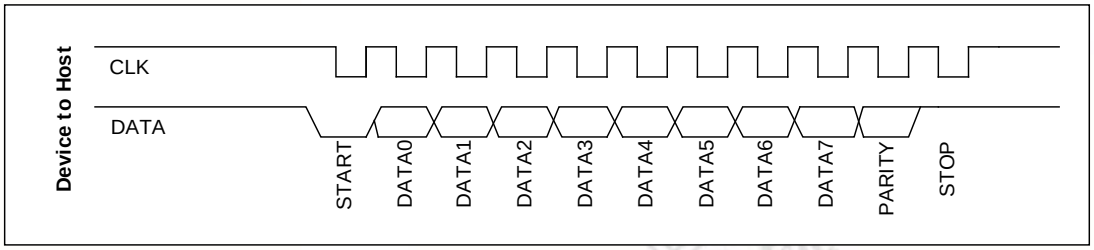


图 5-88 设备向主机传输数据格式

主机向设备传输:

PS/2 设备负责产生 CLK 信号。 如果主机希望发送DATA, 首先需设定CLK 和DATA在 "请求发送" 状态，如下:

- 拉低CLK 至少100 us 禁止通讯
- 应用 "请求发送" ，拉 DATA 为低, 然后释放CLK

设备以不超过10 ms的间隔不间断的监控状态。当设备监控到此状态，将开始产生CLK 信号，将8位DATA位 1 位停止位接收进来。只有当 CLK线为低时，主机才可以改变DATA 线。当 CLK 为高时，设备读取数据.

接收到停止位后，设备发出应答信号，将DATA线拉低并且产生最后一个CLK 脉冲。如果第十一个CLK 脉冲后主机未释放DATA 线, 设备将继续产生CLK脉冲直到 DATA线释放

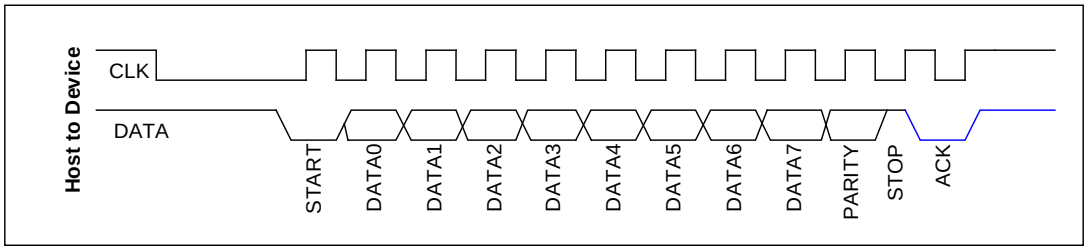


图 5-89 主机向设备传输的数据格式

主机和外设 DATA 和 CLK 详细时序框图如下:

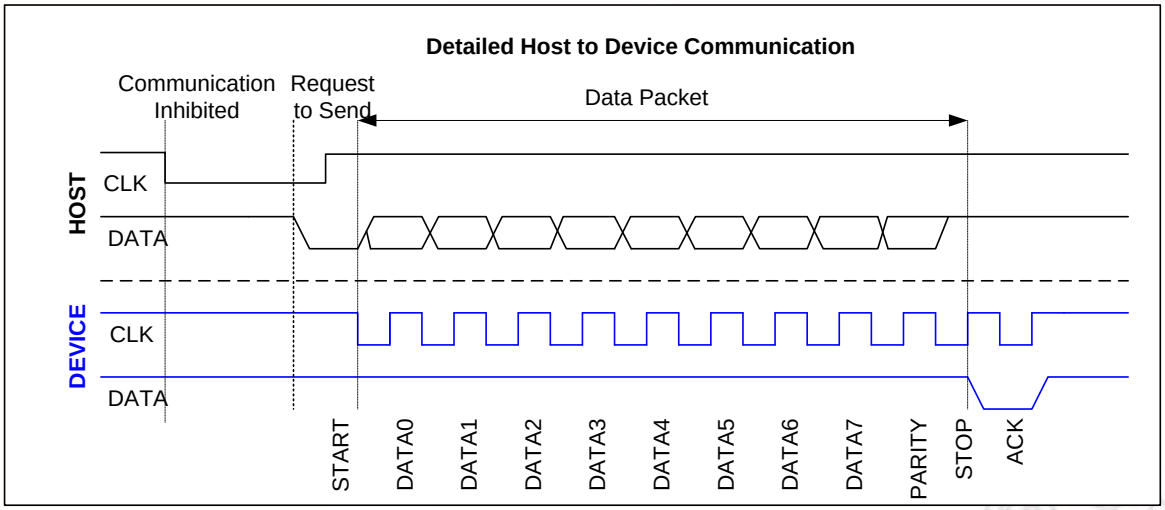


图 5-90 PS/2 Bit 数据格式

5.15.4.2 *PS/2总线时序规范*

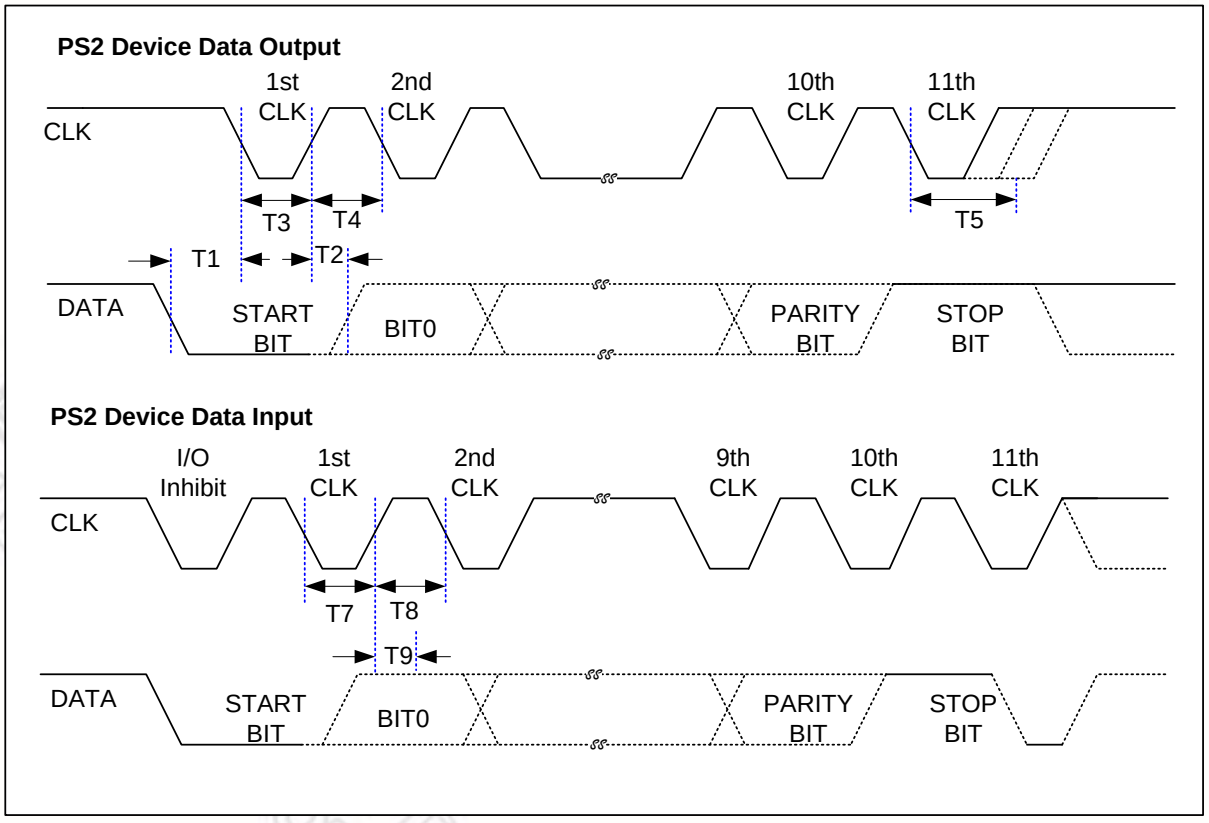


图 5-91 PS/2 总线时序

符号	时序参数	Min	Max
T1	CLK 下降沿 DATA 转换	5us	25us
T2	CLK 上升沿 DATA转换	5us	T4-5us
T3	CLK 持续无效	30us	50us
T4	CLK持续有效	30us	50us
T5	11th clock后 辅助设备禁止其他传输	>0	50us
T7	CLK 持续无效	30us	50us
T8	CLK持续有效	30us	50us
T9	CLK从无效到有效转换时间, 辅助设备取样时间	5us	25us

5.15.4.3 *TX FIFO操作*

写PS2TXDATA0寄存器将触发设备和主机通讯. 在向TX FIFO写发送数据之前，S/W需要定义TXFIFO的长度. 在写PS2TXDATA0寄存器之後100us，第一个字节的start bit才会被传送到总线上。如果要传送的数据超过四个字节，S/W在第四个字节数据传输完成前可写剩余数据到PS2TXDATA1-3。 2个连续的字节之间将延时100us.

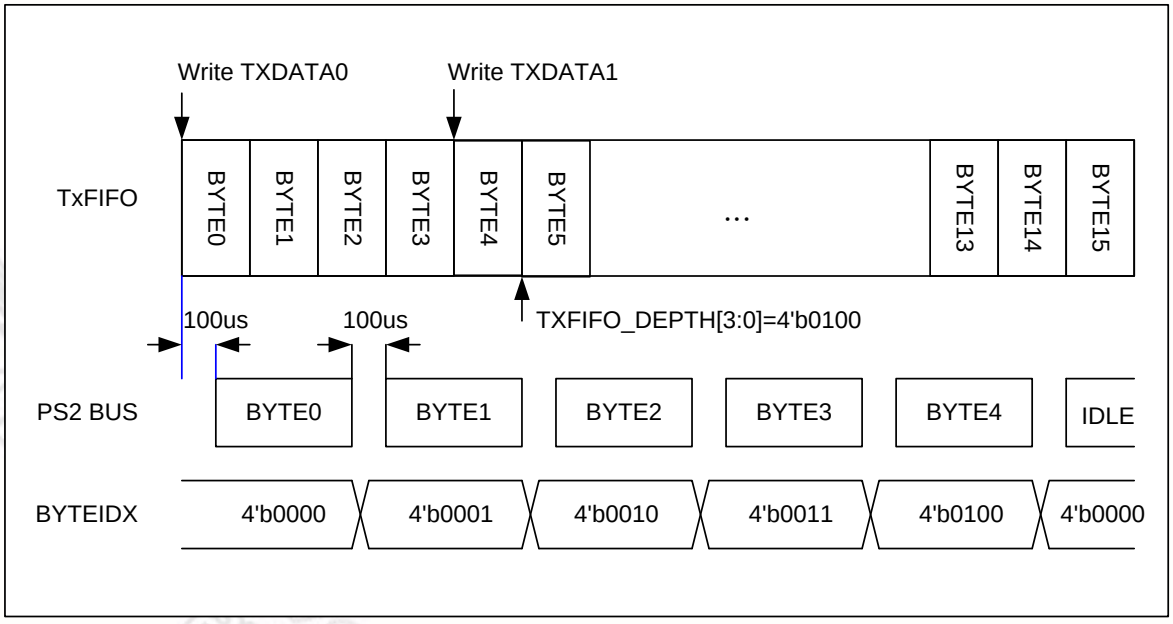


图 5-92 PS/2 数据格式

5.15.5 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
PS2 基地址: **PS2_BA = 0x4010_0000**				
PS2CON	PS2_BA+0x00	R/W	PS/2 控制寄存器	0x0000_0000
PS2TXDATA0	PS2_BA+0x04	R/W	PS/2 发送数据寄存器 0	0x0000_0000
PS2TXDATA1	PS2_BA+0x08	R/W	PS/2 发送数据寄存器 1	0x0000_0000
PS2TXDATA2	PS2_BA+0x0C	R/W	PS/2 发送数据寄存器 2	0x0000_0000
PS2TXDATA3	PS2_BA+0x10	R/W	PS/2 发送数据寄存器 3	0x0000_0000
PS2RXDATA	PS2_BA+0x14	R	PS/2 接收数据寄存器	0x0000_0000
PS2STATUS	PS2_BA+0x18	R/W	PS/2 状态寄存器	0x0000_0083
PS2INTID	PS2_BA+0x1C	R/W	PS/2 中断指示寄存器	0x0000_0000

5.15.6 寄存器描述

PS/2 控制寄存器 (PS2CON)

寄存器	偏移量	R/W	描述	复位后的值
PS2CON	PS2_BA + 0x00	R/W	PS/2控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				FPS2DAT	FPS2CLK	OVERRIDE	CLRFIFO
7	6	5	4	3	2	1	0
ACK	TXFIFO_DEPTH				RXINTEN	TXINTEN	PS2EN

Bits	描述	
[31:12]	保留	保留
[11]	FPS2DAT	**强制 PS2DATA 线** 若OVERRIDE =1，它将强制 PS2DATA低或高，忽略设备控制器的内部状态 1 = 强制 PS2DATA 高 0 = 强制 PS2DATA 低
[10]	FPS2CLK	**强制 PS2CLK 线** 若OVERRIDE =1,它将强制 PS2CLK 线为低或高，忽略设备控制器的内部状态 1 = 强制 PS2CLK 高 0 = 强制 PS2CLK 低
[9]	OVERRIDE	**软件重写PS2 CLK/DATA 引脚 状态** 1 = PS2CLK 和 PS2DATA 引脚由S/W控制 0 = PS2CLK 和 PS2DATA 引脚由内部状态机控制.
[8]	CLRFIFO	**清除 TX FIFO** 向该位写1将终止设备向主机传输. PS2STATUS 寄存器的TXEMPTY 位将置1并且指针 BYTEIDEX 将清0， 无论数据缓冲器中是否有数据. 数据缓冲器不会被清除.

		1 = 清 FIFO 0 = 无效
[7]	**ACK**	**应答使能** 1 = 如果校验错误 或停止位未接收到, 在12th 时钟将不发送应答. 0 = 主机到设备发送时，在12th 时钟总是发送应答位.
[6:3]	**TXFIFODIPTH**	**数据传输FIFO长度** 16 位缓冲器 应用于数据传输. S/W 可定义 FIFO 长度，范围从 1 ~ 16 字节. 0 = 1字节 1 = 2字节 … 14 = 15字节 15 = 16字节
[2]	**RXINTEN**	**使能接收中断** 1 = 使能数据接收完成中断 0 = 禁用数据接收完成中断
[1]	**TXINTEN**	**使能传输中断** 1 = 使能数据传输完成中断 0 = 禁用数据传输完成中断
[0]	**PS2EN**	**使能 PS/2 设备** 使能 PS/2 设备 1 = 使能 0 = 禁用

PS/2 发送数据寄存器0-3 (PS2TXDATA0-3)

寄存器	偏移量	R/W	描述	复位后的值
PS2TXDATA0	PS2_BA + 0x04	R/W	PS/2发送数据寄存器0	0x0000_0000
PS2TXDATA1	PS2_BA + 0x08	R/W	PS/2发送数据寄存器1	0x0000_0000
PS2TXDATA2	PS2_BA + 0x0C	R/W	PS/2发送数据寄存器2	0x0000_0000
PS2TXDATA3	PS2_BA + 0x10	R/W	PS/2发送数据寄存器3	0x0000_0000

31	30	29	28	27	26	25	24
PS2TXDATAx[31:24]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PS2TXDATAx[23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PS2TXDATAx[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PS2TXDATAx[7:0]							

Bits	描述	
[31:0]	**PS2TXDATAx**	**发送数据** 如果总线在空闲状态，向寄存器写数据将启动设备到主机通讯. 软件在写数据到 TX 缓冲器前需使能 PS2EN.

PS/2 接收数据寄存器 (PS2RXDATA)

寄存器	偏移量	R/W	描述	复位后的值
PS2RXDATA	PS2_BA + 0x14	R/W	PS/2接收数据寄存器	0x0000_0000

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

Bits	描述	
[31:8]	保留	保留
[7:0]	PS2RXDATA	**接收数据** 在主机到设备传输中，在应答位发送后，接收到的数据将从移位寄存器复制到 PS2RXDATA 寄存器. CPU 需在下一字节接收完成前读取此寄存器，否则数据将被改写而无效， 并且PS2STATUS[6](RXOVF) 位将置 1

PS/2状态寄存器(PS2STATUS)

寄存器	偏移量	R/W	描述	复位后的值
PS2STATUS	PS2_BA + 0x18	R/W	PS/2状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				BYTEIDX[3:0]			
7	6	5	4	3	2	1	0
TXEMPTY	RXOVF	TXBUSY	RXBUSY	RXPARITY	FRAMERR	PS2DATA	PS2CLK

Bits	描述	
[31:12]	保留	保留
[11:8]	BYTEIDX	**字节索引** 表示哪个数据字节在 发送移位寄存器中 .当 FIFO 所有数据传输完毕，该位清 0..
[7]	TXEMPTY	**TX FIFO 空** 如果PS2EN 使能，当软件写任何数据到 PS2TXDATA0-3时，将置 TXEMPTY 位为 0。当传输 数据的字节数和 FIFODEPTH 相等时，TXEMPTY 将置 1. 1 = FIFO 为空 0 =有数据等待传输 该位只读.

BYTEIDX	DATA Transmit	BYTEIDX	DATA Transmit
0000	TXDATA0[7:0]	1000	TXDATA2[7:0]
0001	TXDATA0[15:8]	1001	TXDATA2[15:8]
0010	TXDATA0[23:16]	1010	TXDATA2[23:16]
0011	TXDATA0[31:24]	1011	TXDATA2[31:24]
0100	TXDATA1[7:0]	1100	TXDATA3[7:0]
0101	TXDATA1[15:8]	1101	TXDATA3[15:8]
0110	TXDATA1[23:16]	1110	TXDATA3[23:16]
0111	TXDATA1[31:24]	1111	TXDATA3[31:24]

[6]	**RXOVF**	**RX Buffer 覆盖** 1 =PS2RXDATA 寄存器中的数据被新收到的数据覆盖. 0 =没有发生覆盖 写 1 清该位
[5]	**TXBUSY**	**发送忙** 该位表示 PS/2 驱动器当前正在发送数据. 1 = 当前正在发送数据 0 = 空闲状态. 该位只读.
[4]	**RXBUSY**	**接收忙** 该位表示 PS/2 驱动器当前正在接收数据. 1 = 当前正在接收数据. 0 =空闲状态. 该位只读.
[3]	**RXPARITY**	**接收校验** 该位反应最后接收字节的校验位. (等于1表示奇校验). 该位只读.
[2]	**FRAMERR**	**帧错误** 对于主机到设备的通信，如果未接收到STOP位则表示帧错误。如果帧错误发生，DATA线在12th时钟时保持低状态. 此时，软件应该驱动PS2CLK一直发送时钟直到PS2DATA 释放成高状态后才停止发送时钟. 然后，设备发送“重传”命令到主机. 1 = 出现帧错误 0 = 未发生帧错误 写 1 清该位.
[1]	**PS2DATA**	**DATA 脚状态** 该位表示同步和取样后PS2DATA线的状态.
[0]	**PS2CLK**	**CLK 脚状态** 该位表示同步后PS2CLK线的状态.

PS/2中断识别寄存器(PS2INTID)

寄存器	偏移量	R/W	描述	复位后的值
PS2INTID	PS2_BA + 0x1C	R/W	PS/2中断识别寄存器	0x0000_0000

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

Bits	描述	
[31:3]	**保留**	保留
[1]	**TXINT**	**传输中断** 当 STOP 位传输后，该位置 1. 如果 TXINTEN 位置 1，中断产生. 1 = 传输中断发生 0 = 无中断 写 1 清该位为 0.
[0]	**RXINT**	**接收中断** 在主机向设备传输时，当应答位发送，该位置位.如果 RXINTEN置 1，中断产生. 1 = 接收中断发生 0 = 无中断 写 1 清该位为 0.

5.16 I²S控制器 (I²S)

5.16.1 概述

I²S控制器由IIS协议与外部音频CODEC接口组成，两个8个字的FIFO分别用于读与写通道，可以处理8～32位字大小。DMA控制器处理数据在FIFO与内存之间的传输.

5.16.2 特征

- I²S可工作于主机模式或从机模式
- 可处理8, 16, 24和32 位字大小
- 支持单声道和立体声的音频数据
- 支持I²S和MSB 校验数据格式
- 提供两个8字的FIFO数据缓冲，一个用于发送，一个用于接收
- 当缓冲超过可编程边界时，产生中断请求
- 两个DMA请求，一个用于发送，一个用于接收

5.16.3 框图

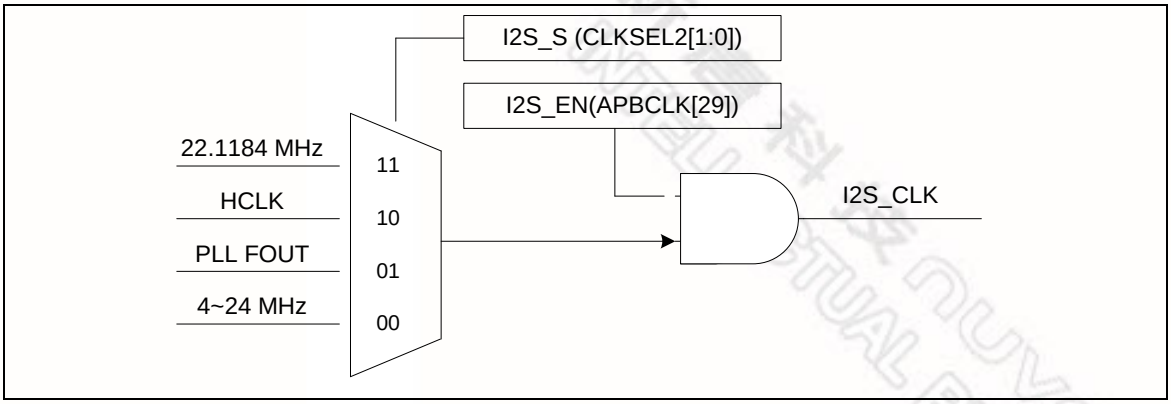


图 5-93 I²S 时钟控制图

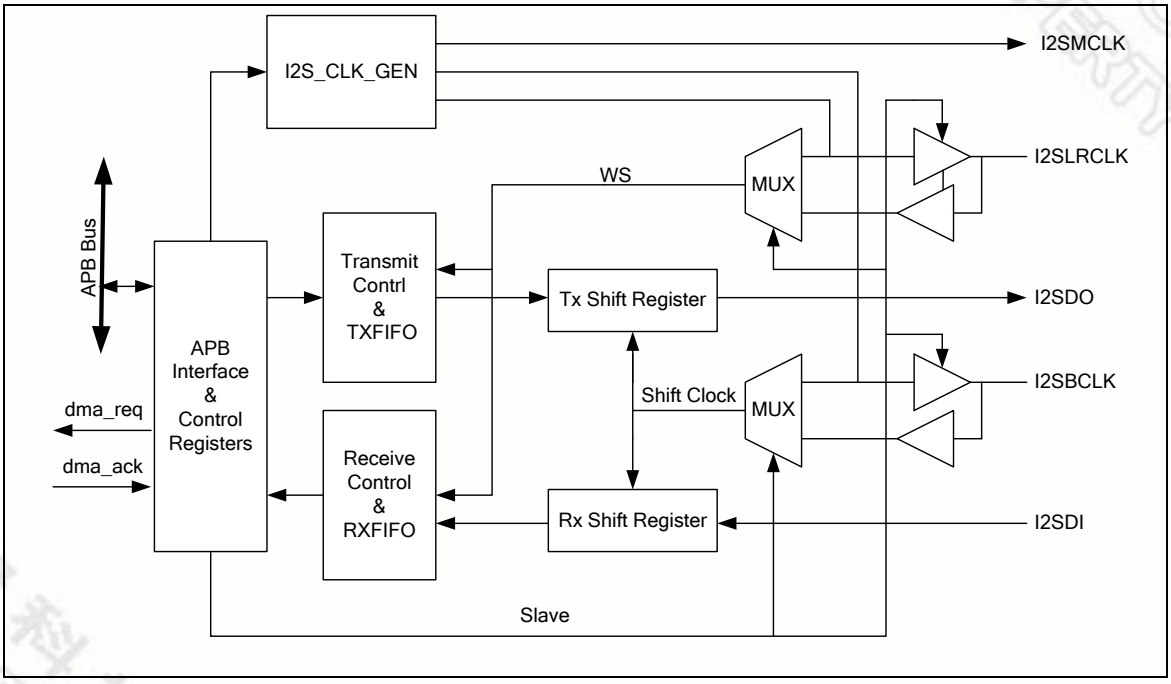


图 5-94 I²S 控制器框图

5.16.4 功能描述

I^2S 操作

5.16.4.1

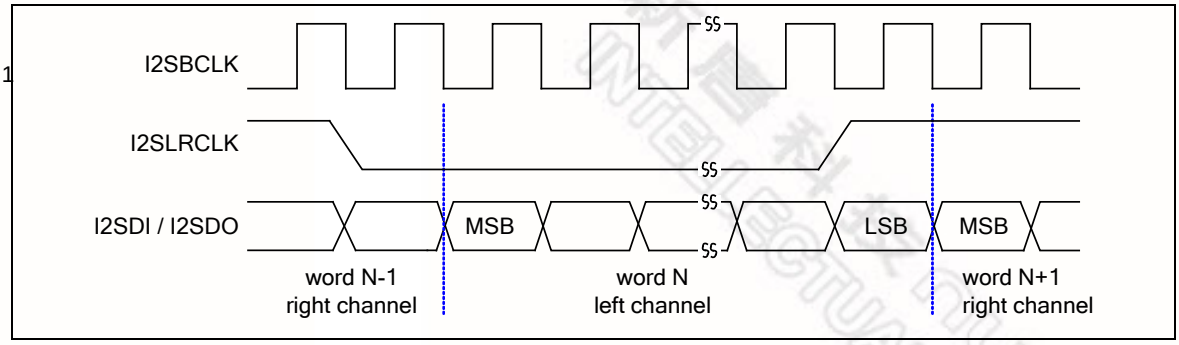


图 5-95 I^2S 总线时序图 (Format = 0)

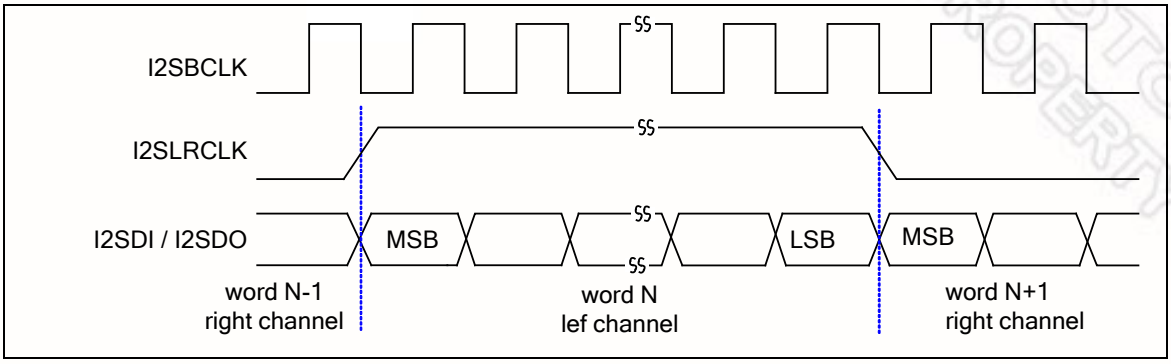


图 5-96 MSB 校正 (MSB Justified) 时序图 (Format = 1)

PCM Operation

5.16.4.2

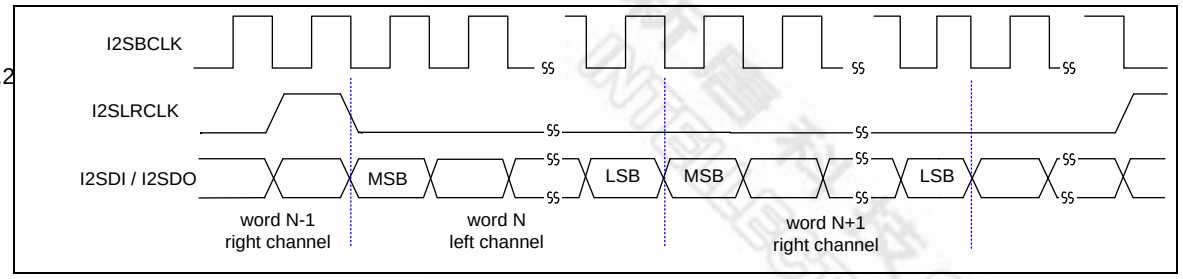


图 5-97 PCM 模式 A 时序图 (Format = 0)

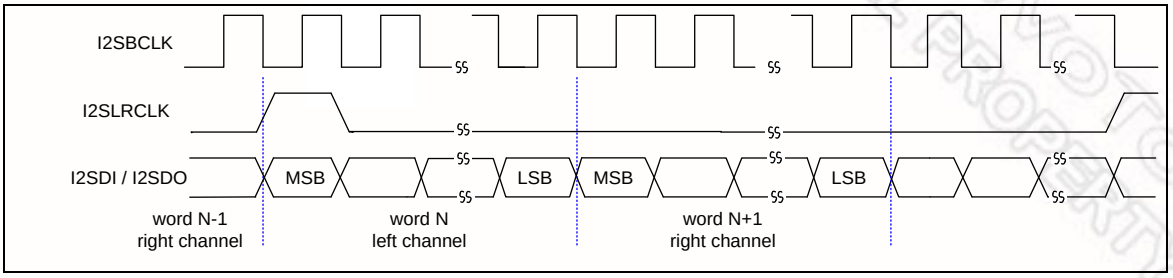


图 5-98 PCM 模式 B 时序图 (Format = 1)

FIFO 操作

5.16.4.3

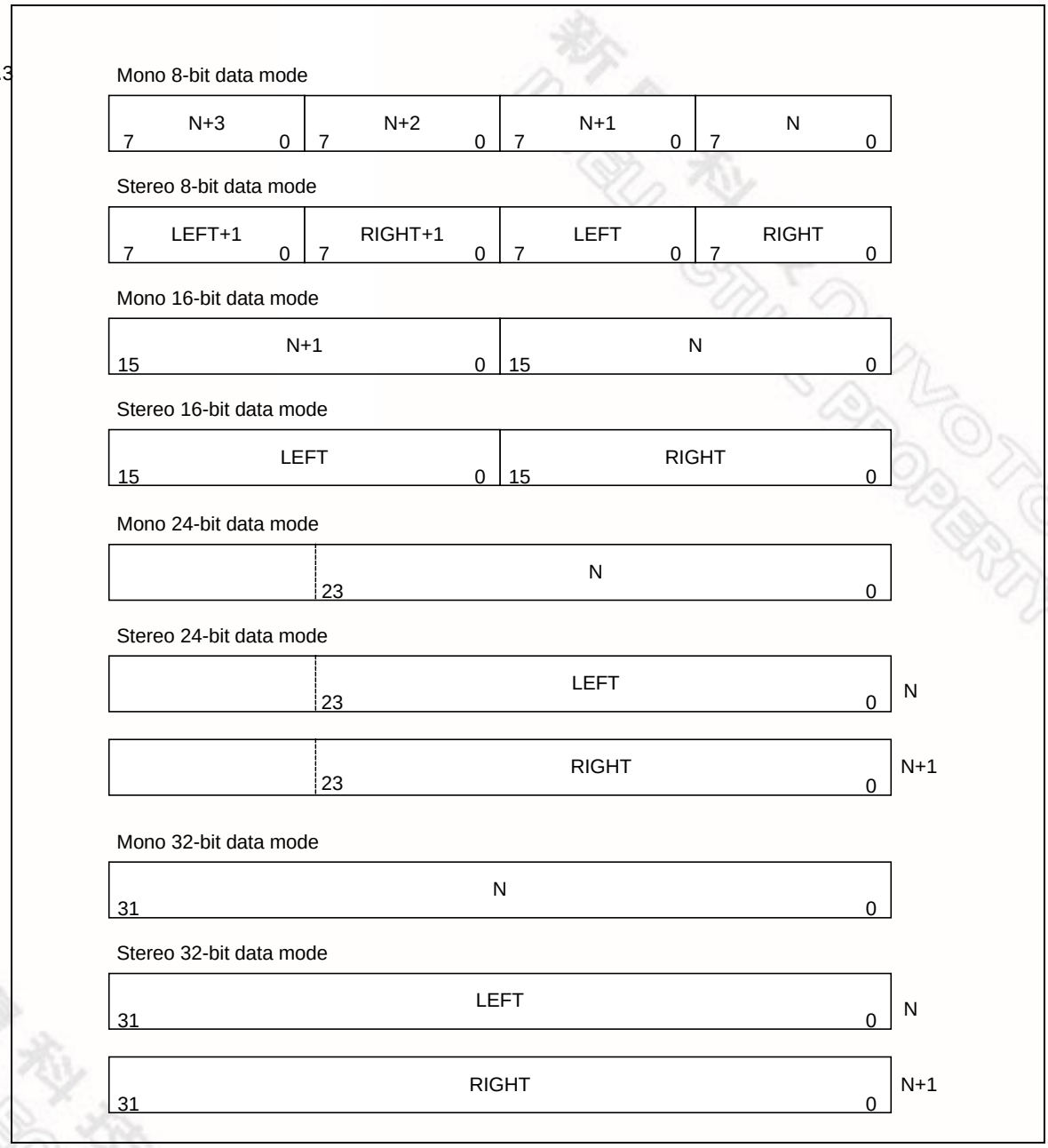


图 5-99 不同 I²S 模式下的 FIFO 内容

5.16.5 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
I2S 基地址: **I2S_BA = 0x401A_0000**				
I2S_CON	I2S_BA+0x00	R/W	I^2S 控制寄存器	0x0000_0000
I2S_CLKDIV	I2S_BA+0x04	R/W	I^2S 时钟分频寄存器	0x0000_0000
I2S_IE	I2S_BA+0x08	R/W	I^2S 中断使能寄存器	0x0000_0000
I2S_STATUS	I2S_BA+0x0C	R/W	I^2S 状态寄存器	0x0014_1000
I2S_TXFIFO	I2S_BA+0x10	R/W	I^2S 发送FIFO寄存器	0x0000_0000
I2S_RXFIFO	I2S_BA+0x14	R/W	I^2S 接收FIFO寄存器	0x0000_0000

5.16.6 寄存器描述

I^2S 控制寄存器(I2S_CON)

寄存器	偏移量	R/W	描述	复位后的值
I2S_CON	I2S_BA+0x00	R/W	I^2S 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							PCM
23	22	21	20	19	18	17	16
RXLCH	保留	RXDMA	TXDMA	CLR_RXFIFO	CLR_TXFIFO	LCHZCEN	RCHZCEN
15	14	13	12	11	10	9	8
MCLKEN	RXTH[2:0]			TXTH[2:0]			SLAVE
7	6	5	4	3	2	1	0
FORMAT	MONO	WORDWIDTH		MUTE	RXEN	TXEN	I2SEN

Bits	描述	
[31:22]	保留	保留
[24]	PCM	**PCM 接口使能** 1 = PCM 接口. 0 = I^2S 接口.
[23]	RXLCH	**接收左声道使能** 当选择单声道时 (MONO = 1)，如果 RXLCH 等于0，I^2S 将接收右声道的数据，如果 RXLCH 等于1，I^2S 将接收左声道的数据. 1 =当选择单声道时，接收左声道的数据. 0 =当选择单声道时，接收右声道的数据.
[22]	保留	保留
[21]	RXDMA	**使能 DMA 接收** 当使能 RX DMA, 如果FIFO不为空，I^2S请求DMA从接收FIFO向SRAM传输数据. 1 = 使能 RX DMA 0 = 禁用 RX DMA
[20]	TXDMA	**使能 DMA 发送** 当使能 TX DMA, 如果FIFO没满，I^2S 请求DMA 从SRAM向发送FIFO传输数据. 1 = 使能TX DMA 0 = 禁用TX DMA

[19]	**CLR_RXFIFO**	**清接收 FIFO** 写“1”清空接收FIFO,内部指针复位到FIFO的起始位置，RXFIFO_LEVEL[3:0] 变成0，接收FIFO为空. 该位由硬件自动清零，读时返回0.
[18]	**CLR_TXFIFO**	**清发送 FIFO** 写“1”清空发送FIFO，内部指针复位到FIFO的起始位置， TXFIFO_LEVEL[3:0] 变成0，发送 FIFO 为空，但在发送FIFO中的数据不变. 该位由硬件自动清零，读时返回0.
[17]	**LCHZCEN**	**使能左声道过零检测** 如果该位置 1，当左声道数据符号位改变或下一个移位数据所有位都为 0，则寄存器 I2S_STATUS 的 LZCF 标志置 1 1 = 使能左声道过零检测 0 = 禁用左声道过零检测
[16]	**RCHZCEN**	**使能右声道过零检测** 如果该位置 1，当右声道数据符号位改变或下一个移位数据所有位全为 0，则寄存器 I2S_STATUS 的 RZCF 标志置 1. 1 = 使能右声道过零检测 0 = 禁用右声道过零检测
[15]	**MCLKEN**	**使能主时钟** 如果NuMicro™ NUC123 外部晶振时钟频率为2*N*256fs，软件可编程寄存器I2S_CLKDIV 的MCLK_DIV[2:0]位，以得到音频CODEC所需的256fs的时钟. 1 = 使能主时钟 0 = 禁用主时钟
[14:12]	**RXTH[2:0]**	**接收 FIFO 阈值** 当接收缓冲里的数据字等于或大于阈值水平时，RXTHF标志置位. 000 = FIFO 接收1个字数据 001 = FIFO接收2个字数据 010 = FIFO接收3个字数据 011 = FIFO接收4个字数据 100 = FIFO接收5个字数据 101 = FIFO接收6个字数据 110 = FIFO接收7个字数据 111 = FIFO接收8个字数据

[11:9]	**TXTH[2:0]**	**发送 FIFO 的阈值水平** 如果发送FIFO中剩下的数据字（32位）等于或小于阈值水平，TXTHF标志置位. 000 = 发送FIFO中有0个字数据 001 = 发送FIFO中有1个字数据 010 = 发送FIFO中有2个字数据 011 = 发送FIFO中有3个字数据 100 = 发送FIFO中有4个字数据 101 = 发送FIFO中有5个字数据 110 = 发送FIFO中有6个字数据 111 = 发送FIFO中有7个字数据
[8]	**SLAVE**	**从机模式** I^2S可以工作于主机模式或从机模式. 主机模式下，I2S_BCLK 与I2S_LRCLK都作为输出模式，从NuMicro™ NUC123发送时钟到外部音频芯片。在从机模式，I2S_BCLK 与I2S_LRCLK 都为输入模式，接收来自于外部音频CODEC芯片的I2S_BCLK 与I2S_LRCLK信号. 1 = 从机模式 0 = 主机模式
[7]	**FORMAT**	**数据格式选择** 如果PCM=0： 1 = MSB 校正数据格式 0 = I^2S 数据格式 如果PCM=1： 1 = PCM模式B 0 = PCM模式A
[6]	**MONO**	**单声道数据** 1 = 单声道数据格式 0 = 立体声数据格式
[5:4]	**WORDWIDTH**	**字宽度** 00 = 数据为8位 11 = 数据为16-位 10 = 数据为24-位 11 = 数据为32-位
[3]	**MUTE**	**使能发送静音** 1= 发送数据为0 0 = 发送数据由缓冲移出
[2]	**RXEN**	**接收使能** 1 = 使能数据接收 0 = 禁止数据接收

[1]	**TXEN**	**发送使能** 1 = 使能数据发送 0 = 禁止数据发送
[0]	**I2SEN**	**使能 I^2S 控制器** 1 = 使能 0 = 禁用

I²S时钟分频(I2S_CLKDIV)

寄存器	偏移量	R/W	描述	复位后的值
I2S_CLKDIV	I2S_BA+0x04	R/W	I²S时钟分频控制寄存器	0x0000_0000

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

Bits	描述	
[31:16]	**保留**	保留
[15:8]	**BCLK_DIV [7:0]**	**位时钟分频** 如果I²S工作于主机模式，NuMicro™ NUC123 负责提供位时钟。软件可以编程这些位产生采样时钟频率. *F_BCLK = F_I2SCLK /(2x(BCLK_DIV + 1))*
[7:3]	**保留**	保留
[2:0]	**MCLK_DIV[2:0]**	**主时钟分频** 如果外部晶振频率等于(2xMCLK_DIV)x256fs ，则软件可编程这些位产生256fs 的时钟频率到外部音频CODEC芯片. 如果MCLK_DIV设置为0， MCLK 与外部输入时钟相同. 如：采样率为24 KHz，芯片外部晶振时钟频率为12.288 MHz, 设置 MCLK_DIV=1. F_MCLK = F_I2SCLK/(2x(MCLK_DIV)) (当 MCLK_DIV >= 1) F_MCLK = F_I2SCLK (当 MCLK_DIV = 0)

I²S中断使能寄存器(I2S_IE)

寄存器	偏移量	R/W	描述	复位后的值
I2S_IE	I2S_BA+0x08	R/W	I²S中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留			LZCIE	RZCIE	TXTHIE	TXOVFIE	TXUDFIE
7	6	5	4	3	2	1	0
保留					RXTHIE	RXOVFIE	RXUDFIE

Bits	描述	
[31:13]	保留	保留
[12]	LZCIE	**左声道过零检测中断使能** 如果该位设为1，左声道有过零信号，产生中断 1 = 使能中断 0 = 禁用中断
[11]	RZCIE	**右声道过零检测中断使能** 1 = 使能中断 0 = 禁用中断
[10]	TXTHIE	**发送 FIFO 阈值水平中断使能** 如果该位设置为1，发送FIFO中的数据少于TXTH[2:0]时，产生中断. 1 = 使能中断 0 = 禁用中断
[9]	TXOVFIE	**发送 FIFO 溢出中断使能** 如果该位设置为1，发送FIFO溢出标志设置为1时，发生中断 1 = 使能中断 0 = 禁用中断
[8]	TXUDFIE	**发送 FIFO 下溢中断使能** 如果该位设置为1，发送FIFO下溢标志设置为1时，中断发生. 1 = 使能中断 0 = 禁用中断

[7:3]	**保留**	保留
[2]	**RXTHIE**	**接收 FIFO 阈值水平中断使能** 当接收FIFO中的数据字等于或大于RXTH[2:0] 时，RXTHF 位设置为1。如果RXTHIE位使能，中断发生. 1 = 使能中断 0 = 禁用中断
[1]	**RXOVFIE**	**接收 FIFO 溢出中断使能** 1 = 使能中断 0 = 禁用中断
[0]	**RXUDFIE**	**接收 FIFO 下溢中断使能** 当接收FIFO为空时，如果软件读接收FIFO，I2SSTATUS寄存器的RXUDF 标志将被设置为1. 1 = 使能中断 0 = 禁用中断

I²S状态寄存器(I2S_STATUS)

寄存器	偏移量	R/W	描述	复位后的值
I2S_STATUS	I2S_BA+0x0C	R/W	I²S状态寄存器	0x0014_1000

31	30	29	28	27	26	25	24
TX_LEVEL[3:0]				RX_LEVEL[3:0]			
23	22	21	20	19	18	17	16
LZCF	RZCF	TXBUSY	TXEMPTY	TXFULL	TXTHF	TXOVF	TXUDF
15	14	13	12	11	10	9	8
保留			RXEMPTY	RXFULL	RXTHF	RXOVF	RXUDF
7	6	5	4	3	2	1	0
保留				RIGHT	I2STXINT	I2SRXINT	I2SINT

Bits	描述	
[31:28]	TX_LEVEL	**发送 FIFO 水平** 这些位表示在发送FIFO中数据字的数目 0000 = 无数据 0001 = 在发送 FIFO有1个字 …. 1000 = 在发送 FIFO有8个
[27:24]	RX_LEVEL	**接收 FIFO 水平** 这些位表示在接收 FIFO 中数据字的数目 0000 = 无数据 0001 = 在接收FIFO中有1个字 …. 1000 =在接收FIFO中有8个字
[23]	LZCF	**左声道过零标志** 表示左声道下一个采样数据符号位发生改变或所有数据位为0. 1 = 左声道检测到过零 0 = 没有过零 该位只读

[22]	**RZCF**	**右声道过零标志** 表示右声道下一个采样数据符号位发生改变或所有数据位为0. 1 = 右声道过零检测 0 = 没有过零 该位只读
[21]	**TXBUSY**	**发送忙** 当所有发送FIFO中的数据和移位缓存中的数据都被移出时，该位清零，当第一个数据加载到移位缓冲时，该位置1 1 = 发送移位缓冲忙 0 = 发送移位缓冲空 该位只读.
[20]	**TXEMPTY**	**发送 FIFO 空** 该位反映发送FIFO中的数据字数是0 0 = 空 1 = 非空 该位只读.
[19]	**TXFULL**	**发送 FIFO 满** 该位反映发送FIFO中数据字数是8 0 = 满. 1 = 没满. 该位只读
[18]	**TXTHF**	**发送 FIFO 阙值标志** 当发送FIFO中的数据字等于或低于TXTH[2:0]中所设的阙值， TXTHF 变成1. 并保持为1直到软件写TXFIFO寄存器使TXFIFO_LEVEL[3:0] 高于TXTH[1:0]. 1 = FIFO中的数据字等于或低于阈值水平 0 = FIFO中的数据字高于阈值水平 该位只读
[17]	**TXOVF**	**发 送 FIFO 溢出标志** 当发送FIFO已满时，再向发送FIFO中写数据，该位将置1 1 = 溢出 0 = 没有溢出 软件写1清该位
[16]	**TXUDF**	**发送 FIFO 下溢标志** 发送FIFO为空，且移位逻辑硬件从数据FIFO读数据，引起该位置1. 1 = 下溢 0 = 无下溢 软件写1清该位

[15:13]	**保留**	保留
[12]	**RXEMPTY**	**接收 FIFO 空** 该位反映接收FIFO中的数据字数为0 1 = 空 0 = 非空 该位只读.
[11]	**RXFULL**	**接收 FIFO 满** 该位反映接收FIFO中的数据字数为8 1 = 满 0 = 没满 该位只读.
[10]	**RXTHF**	**接收 FIFO 阙值标志** 接收FIFO中的数据字数等于或高于RXTH[2:0]中所设定的阙值时，RXTHF 位变为1，并保持为1，直到软件读RXFIFO 寄存器，导致RXFIFO_LEVEL[3:0] 低于 RXTH[1:0] 1 = FIFO中的数据字等于或高于阈值水平 0 = FIFO中的数据字低于阈值水平 该位只读
[9]	**RXOVF**	**接收 FIFO 溢出标志** 当接收FIFO已满，并且接收硬件试图向接收FIFO写数据，该位置1，并且第一个缓冲数据被覆盖. 1 = 发生溢出 0 = 没有发生溢出 该位写1清零
[8]	**RXUDF**	**接收 FIFO 下溢标志** 当接收FIFO为空时，读接收FIFO，该位置1，表示发生下溢. 1 = 发生下溢 0 = 没有 发生下溢 写1清该位
[7:4]	**保留**	保留
[3]	**RIGHT**	**右声道** 该位表示当前发送的数据属于右声道 1 = 右声道 0 = 左声道 该位只读

[2]	**I2STXINT**	**I²S 发送中断** 1 = 发送中断 0 = 没有发送中断 该位只读
[1]	**I2SRXINT**	**I²S 接收中断** 1 = 接收中断 0 = 没 有接收中断 该位只读
[0]	**I2SINT**	**I²S 中断标志** 1 = I²S中断 0 =没有I²S中断 该位是I2STXINT 与I2SRXINT 的“或” 该位只读.

I²S发送FIFO (I2S_TXFIFO)

寄存器	偏移量	R/W	描述	复位后的值
I2S_TXFIFO	I2S_BA+0x10	R/W	I²S发送FIFO	0x0000_0000

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

Bits	描述	
[31:0]	**TXFIFO**	**发送 FIFO 寄存器** I²S包括8个字（8x32 位）的数据缓冲用于数据发送，向该寄存器写数据以准备发送数据，余下的字数由寄存器I2S_STATUS 中的TX_LEVEL[3:0]指示

I²S 接收FIFO (I2S_RXFIFO)

寄存器	偏移量	R/W	描述	复位后的值
I2S_RXFIFO	I2S_BA+0x14	R/W	I²S 接收FIFO	0x0000_0000

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

Bits	描述	
[31:0]	**RXFIFO**	**接收 FIFO 寄存器** I²S包括8个字 (8x32 位)的数据缓冲用于数据接收. 读该寄存器得到FIFO中的数据，余下的数字数由寄存器I2S_STATUS中的 RX_LEVEL[3:0]指示.

5.17 模拟数字转换 (ADC)

5.17.1 概述

NuMicro™ NUC123 包含 一个10位 8通道逐次逼近式 模拟 – 数字转换器 (SAR A/D converter). A/D 转换器支持 三种操作模式: 单次（single），单周期扫描(single-cycle scan) 和连续扫描模式 (continuous scan). A/D 转换可由软件、外部STADC引脚和PWM中心对齐触发.

5.17.2 特性

- 模拟输入电压范围: 0~ V_{DDA} AVDD引脚的电压
- 10位分辨率和8位精确度保证
- 多达 8 路单端模拟输入通道
- 最大 ADC 时钟频率 6 MHz
- 转换速度为150KSPS
- 三种操作模式
  - 单次模式: A/D转换在指定通道完成一次
  - 单周期扫描模式:A/D转换在所有指定通道完成一个循环，转换顺序从最小通道号到最大通道号
  - 连续扫描模式: A/D连续执行单周期扫描模式直到软件停止A/D转换
- A/D转换开始条件
  - 软件向ADST 位写1
  - 外部STADC引脚
  - PWM输出触发
- 每个通道转换结果存储在数据寄存器内，并带有有效/覆盖标志
- 转换结果可以和指定的值相比较，当转换结果和比较寄存器中的设定值匹配时，用户可以选择是否产生一个中断请求
- 通道 7 支持 2个输入源选择：外部模拟电压, 内部带隙(bandgap)基准电压

5.17.3 框图

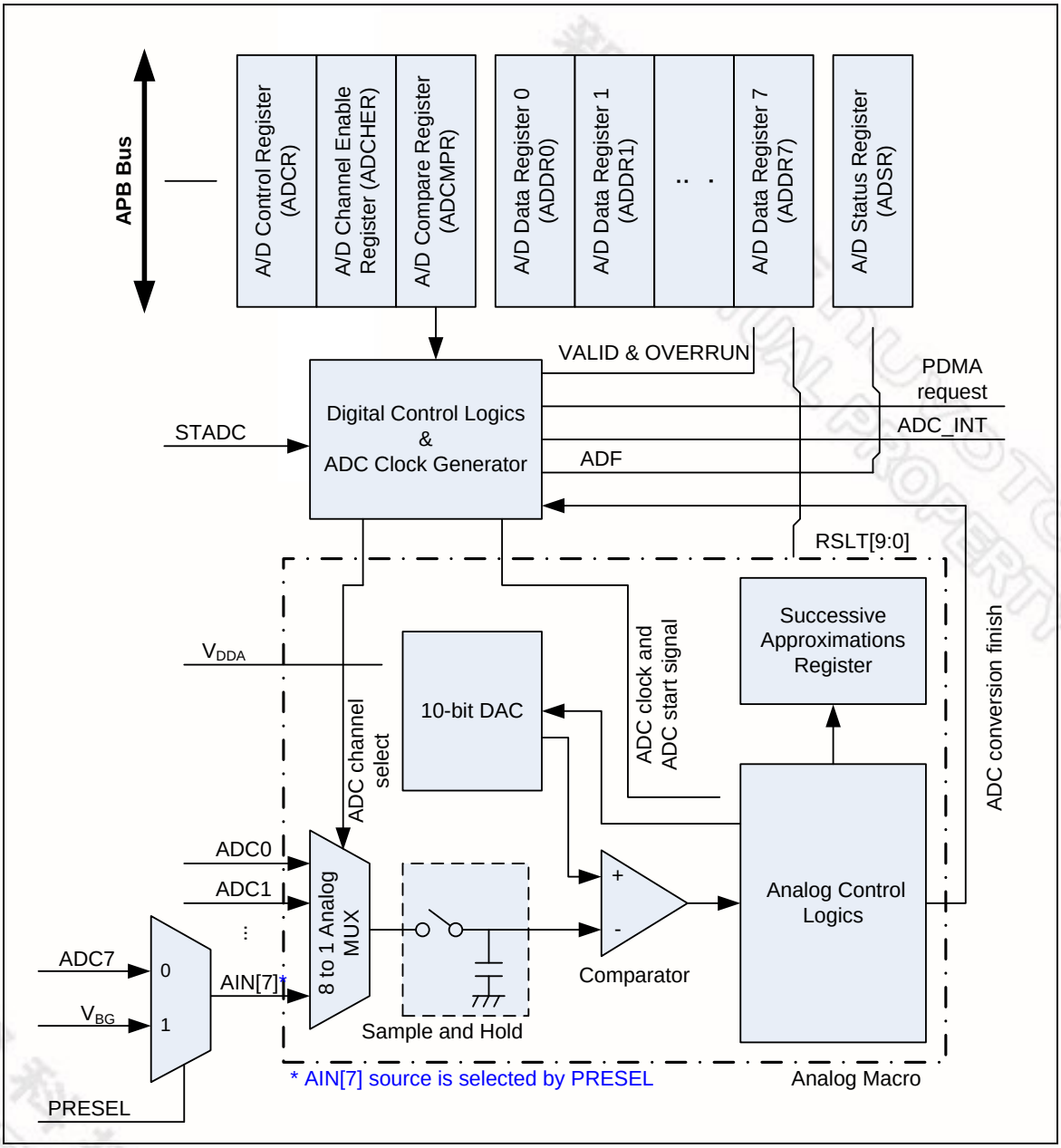


图 5-100 ADC 控制器框图

5.17.4 功能描述

A/D 转换器为 10位逐次逼近式.A/D。ADC有3 种操作模式: 单次触发模式、单周期扫描模式和连续扫描模式. 当改变操作模式或模拟输入通道时, 为了防止错误的操作, 软件需先清 ADST 位为 0 (ADCR 寄存器).

ADC 时钟发生器

5.17.4.1 最大采样率达150 KHz. ADC有四个时钟源，可由两位ADC_S(CLKSEL[3:2])选择, ADC 时钟频率按如下公式进行8位预分频:

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]中设置.

如果时钟源来自HCLK，ADC_N不能为0。一般情况下，软件通过设置ADC_S 与 ADC_N 可以获得 6 MHz 或稍低于6 MHz的频率.

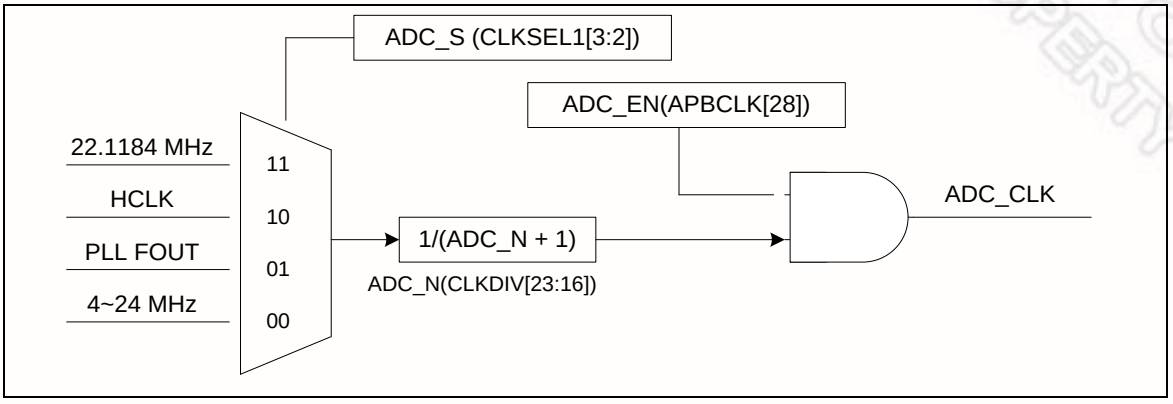


图 5-101 ADC 时钟控制

5.17.4.2

单次触发模式(Single Mode)

在单次触发模式下, A/D 转换仅仅在指定的单一通道中执行一次，运作流程如下:

1. 当 ADCR寄存器 的ADST被软件或者外部触发输入置为1时，A/D转换开启
2. 当 A/D 转换完成, 转换值将存储在与通道相对应的A/D数据寄存器中.
3. A/D 转换完成, ADSR 的ADF 位置为1. 若此时ADCR寄存器的ADIE 位也为1, 将产生ADC中断请求.
4. A/D转换期间，ADST 位维持为1. A/D 转换结束, ADST 位自动清 0 ，A/D 转换器进入空闲状态。注意，ADST位被清为0后，至少要等一个ADC时钟周期，才能再将ADST位置1，否则A/D 转换器可能不工作.

注: 在单次触发模式，如果软件使能多于一个通道，编号最小的通道将会被转换，其他通道被忽略.

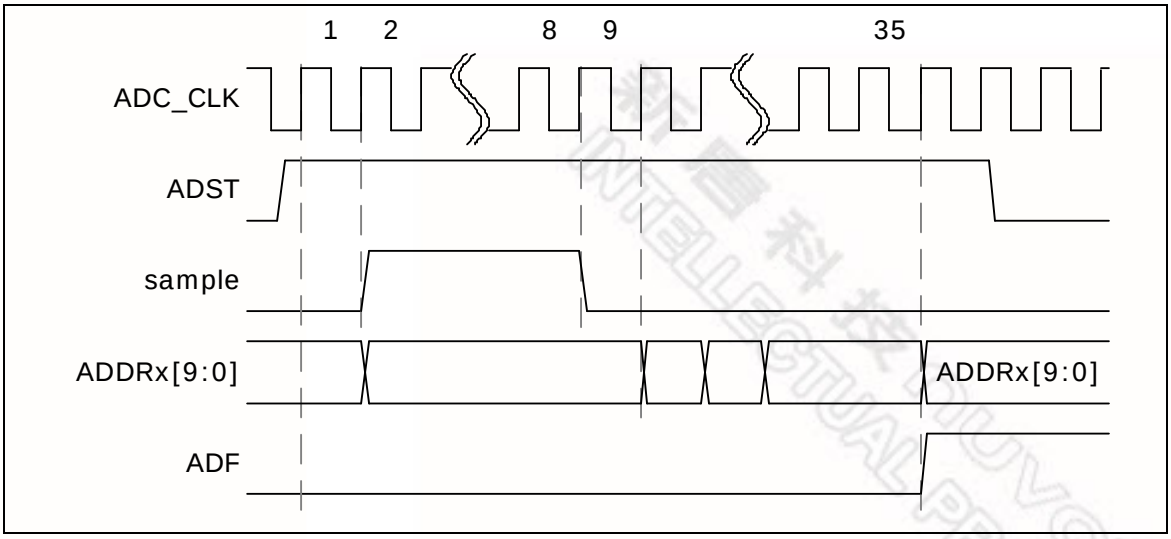


图 5-102 单次转换模式时序图

5.17.4.3 *单周期扫描模式(Single-Cycle Scan Mode)*

在单周期扫描模式下, ADC会对所有使能的通道依次进行一次采样转换,且从编号最小的通道开始，直到编号最高的使能通道。操作流程如下:

1. 当ADCR寄存器的ADST被软件或者外部触发输入置位时，A/D转换从最小编号的通道开始.
2. 每路 A/D 转换完成后, A/D 转换结果将依次放到相应的数据寄存器中.
3. 当所有使能的通道全部转换完成后， ADF 位 （ADSR 寄存器）置1. 若此时 ADIE 置1，将产生 ADC中断请求.
4. AD转换结束后，ADST位自动清零，A/D转换器进入空闲状态。若在转换过程中，ADST被软件清为零，ADC转换器将完成当前转换，并保存结果到当前转换通道的ADDRx寄存器。但是ADIF并不会置位，ADC中断也不会发生。注意，ADST位被清为0后，至少要等一个ADC时钟，才能再将ADST位置1，否则A/D 转换器可能不工作.

例：使能通道(0, 2, 3 和 7) 的单周期扫描模式时序图如下:

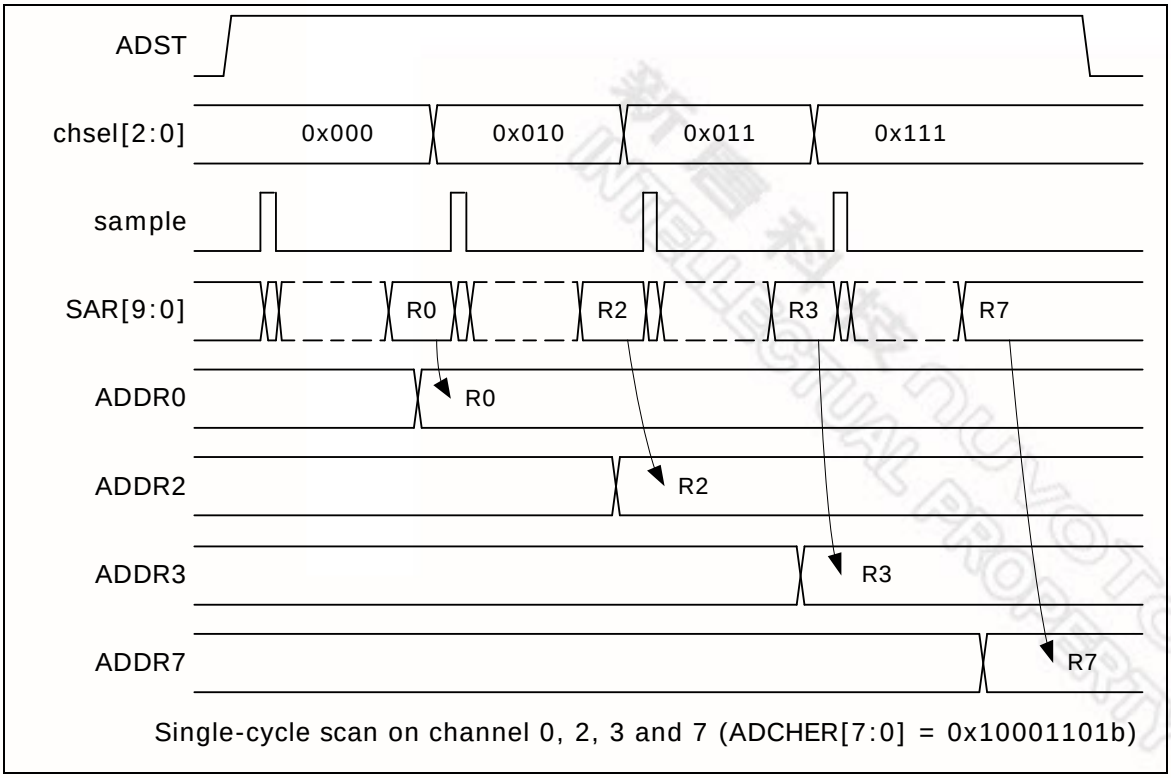


图 5-103 使能通道上单周期扫描模式时序图

连续扫描模式(Continuous Scan Mode)

5.17.4.4 连续扫描模式下， A/D转换器会重复依次对ADCHER寄存器中CHEN位(8个比特)使能的通道执行转换. 操作如下:

1. 当ADCR寄存器的ADST被软件或者外部触发输入置位时，A/D转换从最小编号的通道开始.
2. 每路使能的通道 A/D 转换完成后，转换的结果将放到相应的数据寄存器中.
3. 当所有被使能的通道依序完成一次A/D转换时， ADF 位 （ADSR[0]）置位. 若此时 ADIE 置位, 将产生 ADC中断请求。如果软件没有清除ADST位，使能的最小编号通道将再次开始新的转换.
4. 只要ADST保持为1，步骤2到步骤3就被重复执行。若在转换过程中，ADST被软件清为零，ADC转换器将完成当前转换，并保存结果到当前转换通道的ADDRx寄存器。但是ADIF并不会置位，ADC中断也不会发生。注意，ADST位被清为0后，至少要等一个ADC时钟，才能再将ADST位置1，否则A/D 转换器可能不工作

例：使能通道（0, 2, 3 和 7）连续扫描的时序如下图:

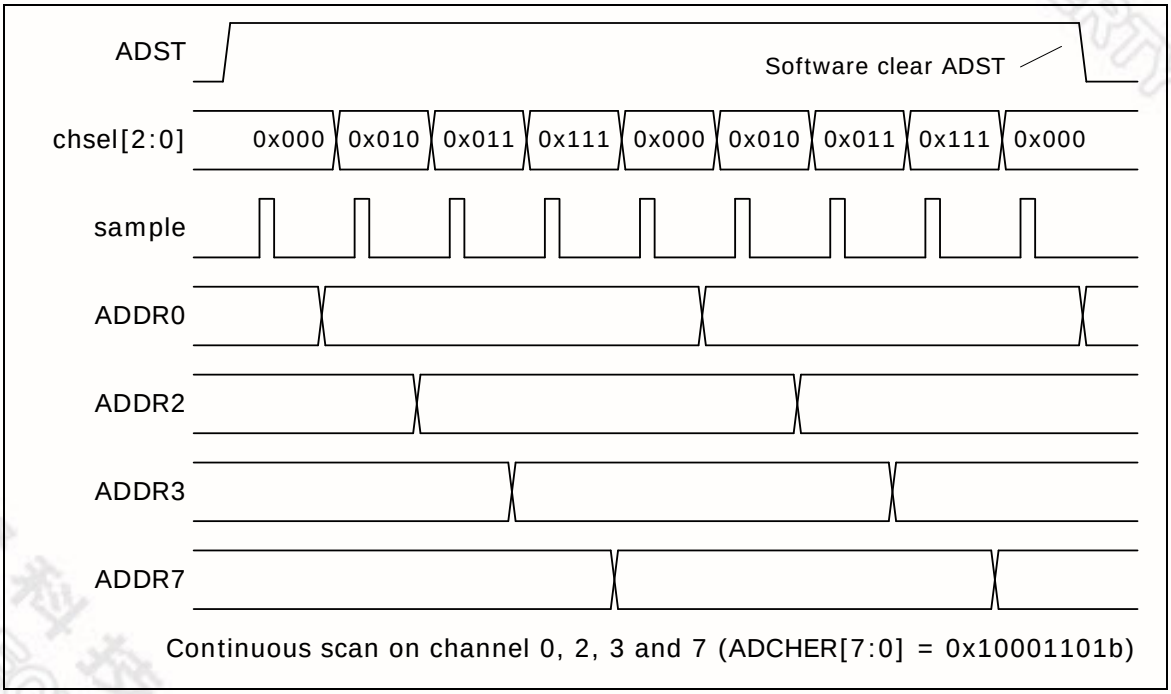


图 5-104 使能通道上连续扫描时序图

外部触发输入采样和A/D 转换时间

5.17.4.5 在单周期扫描模式下,可通过外部引脚触发A/D转换。当ADCR的TRGEN 置为1时，使能ADC外部触发功能，通过设定 TRGS[1:0] 位为 00b 选择STADC 引脚外部触发输入。并可由软件设定TRGCOND[1:0] 选择上升/下降沿或低/高电平触发。若选择电平触发，STADC引脚必须保持在选择的触发状态至少 8个PCLK。在第九个PCLK时，ADST将被置1，开始转换。在电平触发模式下，如果外部触发输入维持在有效状态，转换将会持续进行，仅当外部触发条件消失才停止。若选择边缘触发模式， 高和低状态至少需保持 4 个PLCK，低于该值的脉冲将被忽略。并且ADST一旦置为1，ADC的工作模式就跟单周期扫描模式一样。意思就是，边沿触发时，一旦将ADST置为1，将进行一次单周期扫描

比较模式下AD转换结果监控

5.17.4.6 ADC控制器提供两组比较寄存器 ADCMPR0 和 ADCMPR1，用于监控来自A/D 控制器的 最多2个通道的转换结果, 可参考 图 5-105。软件可通过设定CMPCH(ADCMPRx[5:0]) 来选择监控哪路通道，而CMPCOND 位被用来检查转换结果小于或大于等于CMPD[9:0]中指定的值. 当CMPCH指定的通道转换完成时，比较行为将会被自动的触发一次. 当比较结果和设定值相匹配, 比较匹配计数器将加1, 否则，比较匹配计数器将会被清0。当计数器的值和设定值（CMPMATCNT+1）相等时，CMPF 位将置 1, 如果CMPIE 置位，将产生 ADC_INT中断请求. 在扫描模式下无需软件介入就可用其监控外部模拟输入引脚电压变化。具体逻辑框图如下:

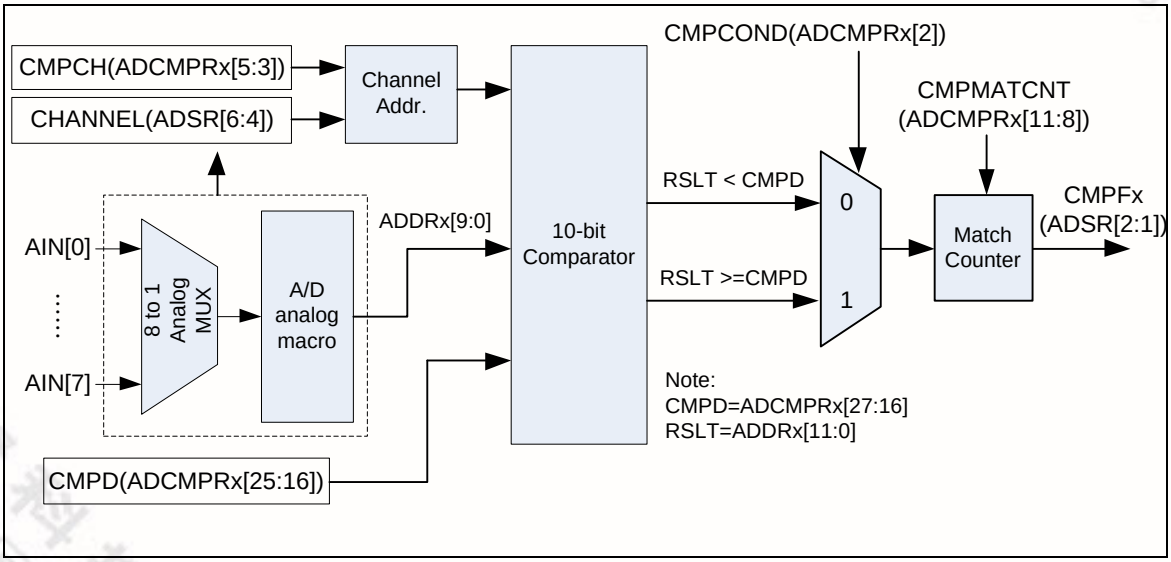


图 5-105 A/D 转换结果监控逻辑图

中断源

5.17.4.7 A/D转换器有三个中断源. 当一次ADC转换结束时，ADSR寄存器的A/D转换结束标志ADF位会被置1.。当A/D转换结果同 ADCMPR0/1 寄存器设定值相匹配时, CMPF0/1会被置1。当ADF、CMPF0和CMPF1 其中一个标志被置1，且其相应的中断使能位 ADIE（ADCR 寄存器）及CMPIE（ADCMPR0/1寄存器）置1时，将产生 ADC中断.。软件可以清除标志位以撤销中断请求.

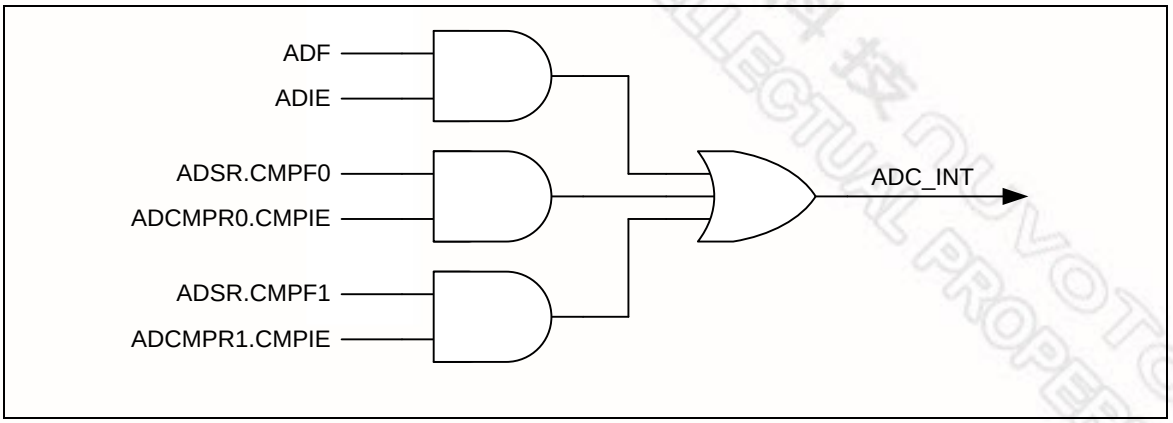


图 5-106 A/D 控制器中断

5.17.4.8 *外设DMA 请求*

当 A/D 转换完成时，转换结果被放到 ADDR 寄存器且VALID置1. 如果 ADCR 寄存器 PTEN 位置为1, ADC 控制器将产生请求到PDMA，以便用户能使用PDMA将数据传输到用户指定的内存空间，而无需CPU参与。不管选择哪个通道，PDMA操作的源地址都为ADPDMA。如果ADC工作于单周期或连续扫描模式，当PDMA正在传输转换结果时，ADC将继续转换下一个使能的通道。用户可以通过读寄存器ADPDMA监控PDMA当前正在传输的数据，如果ADC完成指定通道的转换，但是相同通道上次的转换结果还没有被PDMA传输，相应通道的OVERUN位将置位，上次ADC转换结果将被新的ADC转换结果覆盖. PDMA将传输指定通道的最新数据到用户指定的目的地址.

5.17.5 ADC初始化范例

1. 写CLKSEL1寄存器选择ADC的时钟源
2. 写APBCLK寄存器使能ADC的时钟
3. 写GPx_MFP和ALT_MFP寄存器将相应引脚配置为ADC功能
4. 写GPIOx_OFFD寄存器，关闭相应引脚的数字通路，防止漏电
5. 写ADCHER寄存器使能相应的ADC通道
6. ADCR寄存器配置ADC工作模式，并使能ADC，如果需要写ADIE使能ADC中断
7. 调用NVIC_EnableIRQ(ADC_IRQn)使能ADC中断。
8. 转换结束之后ADSR寄存器的ADF将被置为1
9. 读相应的ADDRx寄存器得到转换的结果

5.17.6 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
ADC 基地址: **ADC_BA = 0x400E_0000**				
ADDR0	ADC_BA+0x00	R	A/D数据寄存器 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D数据寄存器 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D数据寄存器 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D数据寄存器 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D数据寄存器 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D数据寄存器 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D数据寄存器 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D数据寄存器 7	0x0000_0000
ADCR	ADC_BA+0x20	R/W	A/D控制寄存器	0x0000_0000
ADCHER	ADC_BA+0x24	R/W	A/D 通道使能寄存器	0x0000_0000
ADCMPR0	ADC_BA+0x28	R/W	A/D 比较寄存器0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	A/D比较寄存器1	0x0000_0000
ADSR	ADC_BA+0x30	R/W	A/D 状态寄存器	0x0000_0000
ADPDMA	ADC_BA+0x40	R	ADC PDMA 当前传输数据寄存器	0x0000_0000

5.17.7 寄存器描述

A/D 数据寄存器(ADDR0 ~ ADDR7)

寄存器	偏移量	R/W	描述	复位后的值
ADDR0	ADC_BA+0x00	R	A/D数据寄存器 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D数据寄存器 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D数据寄存器 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D数据寄存器 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D数据寄存器 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D数据寄存器 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D数据寄存器 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D数据寄存器 7	0x0000_0000

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

Bits	描述	
[31:18]	**保留**	保留
[17]	**VALID**	**有效标志位** 1 = RSLT[9:0] 中的数据 有效. 0 = RSLT[9:0] 中的数据 无效. 相应通道模拟转换完成后，该位被置位，读ADDR 寄存器后，该位由硬件清除 该位只读
[16]	**OVERRUN**	**溢出标志位** 1 = RSLT[9:0] 数据被覆盖. 0 = RSLT[9:0] 数据没有被覆盖，为当前转换结果. 新的转换结果存到 该寄存器时，若 RSLT[9:0] 的数据没有被读取, OVERRUN 将置 1，前一个转换结果将被覆盖。读ADDR 寄存器后，该位由硬件清除 该位只读

[15:10]	**保留**	保留
[9:0]	**RSLT**	**A/D 转换结果** 包含了10位ADC的转换结果.

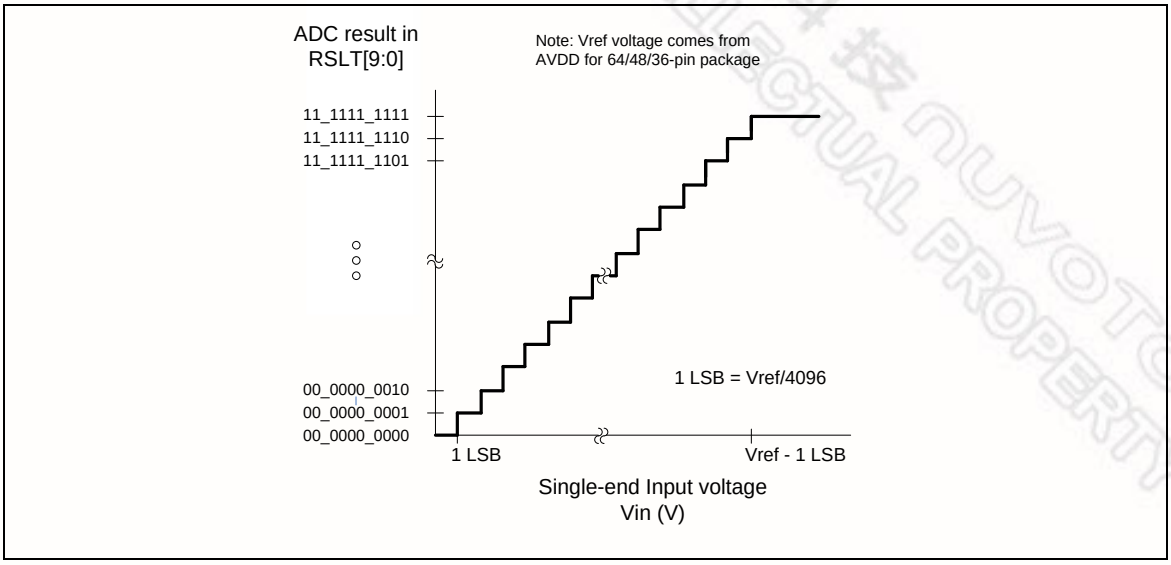


图 5-107 ADC 单端输入转换电压和转换结果映射图

A/D 控制寄存器(ADCR)

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

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

Bits	描述	
[31:12]	保留	保留
[11]	ADST	**A/D 转换开始** 1 = 转换开始. 0 = 转换结束，A/D转换器进入空闲状态. ADST位置1有下列 3 种方式：软件设定和外部引脚 STADC.触发和PWM输出。单周期模式和单次模式下，在转换结束后，ADST将被硬件自动清0。在连续扫描模式下, A/D 转换将一直进行直到软件向该位写0或芯片复位.
[9]	PTEN	**PDMA 传输使能** 1 = 使能 PDMA 传输ADDR 0~7中的数据 0 = 禁用 PDMA 数据传输. 当 A/D 转换完成时, 转换的结果被存放到 ADDR 0~7, 软件可使能该位以便产生PDMA 数据传送请求. 当 PTEN=1时, 软件需设定 ADIE=0 禁止中断.
[8]	TRGEN	**外部触发使能** 使能或禁止外部STADC 引脚触发A/D 转换 1= 使能 0= 禁用 ADC外部触发功能只有在单周期扫描模式下才可以使用 如果使能硬件触发，ADST位由选中的硬件触发源设为1. TRGEN 设为1之前, 确保STADC 引脚为无效状态.否则当TRGEN被设为1时，ADC 可能会额外产生一次触发

[7:6]	**TRGCOND**	**外部触发条件** 该 2 位决定外部引脚 STADC 触发条件为电平 或 边沿触发。该信号必须保持至少8 个PCLK的稳定状态用于电平触发，4个 PCLK 的高和低状态用于边沿触发. 00 = 低电平 01 = 高电平 10 = 下降沿 11 = 上升沿
[5:4]	**TRGS**	**硬件触发源** 00 = A/D转换由外部STADC引脚触发 01 = A/D转换由PWM中心对齐触发 其它配置 =保留 改变 TRGS 前，软件应该关闭TRGE 和 ADST.
[3:2]	**ADMD**	**A/D 转换操作模式** 00 = 单次转换 01 = 保留 10 = 单周期扫描 11 = 连续扫描 当改变操作模式时, 软件应首先禁止 ADST 位.
[1]	**ADIE**	**A/D 中断使能** 1 = 使能 A/D 中断功能 0 = 禁用 A/D 中断功能 当 ADIE 置位，A/D 转换结束将产生中断.
[0]	**ADEN**	**A/D 转换使能** 1 = 使能 0 = 禁止 开始 A/D 转换功能时,该位需置位。 该位为 0 将关闭 A/D 转换器模拟电路的电源.

A/D通道使能寄存器(ADCHER)

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

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

Bits	描述	
[31:9]	保留	保留
[9:8]	PRESEL	**模拟输入通道 7 选择** 0 = 外部模拟输入 1 = 内部bandgap电压 **注：**当选择内部bandgap电压作为通道7的模拟输入时，建议ADC时钟频率低于300 KHz
[7:0]	CHEN	**模拟输入通道7使能位** 设置CHEN[7:0]相应位为1，使能相应的模拟通道7 ~ 0. 1 = 使能 0 = 禁用

A/D 比较寄存器 0/1 (ADCMPR0/1)

寄存器	偏移量	R/W	描述	复位后的值
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
保留						CMPD[9:8]	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPD[7:0]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
保留				CMPMATCNT			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
保留		CMPCH			CMPCOND	CMPIE	CMPEN

Bits	描述	
[31:26]	**保留**	保留
[25:16]	**CMPD**	**比较数值** 此10位数值将和指定通道的转换结果相比较. 当DMOF置0, ADC比较器比较CMPD 与无符号格式的转换结果. CMPD 也要以无符号格式存储. 当DMOF置1, ADC比较器比较CMPD与2'complement格式的转换结果. CMPD也要以2'complement 格式存储.
[15:12]	**保留**	保留
[11:8]	**CMPMATCNT**	**比较匹配次数** 当指定A/D通道的转换结果和比较条件CMPCOND[2]相匹配时，内部匹配计数器将加1，当内部计数器的值达到设定值(CMPMATCNT +1)时，CMPFx 位将被置位.
[7:6]	**保留**	保留

[5:3]	**CMPCH**	**比较通道选择** 000 = 选择 通道 0 的转换结果进行比较. 001 = 选择 通道 1 的转换结果进行比较. 010 = 选择 通道 2 的转换结果进行比较. 011 = 选择 通道 3 的转换结果进行比较. 100 = 选择 通道 4 的转换结果进行比较. 101 = 选择 通道 5 的转换结果进行比较. 110 = 选择 通道 6 的转换结果进行比较. 111 = 选择 通道 7 的转换结果进行比较
[2]	**CMPCOND**	**比较条件** 1= 设置比较条件为当10位A/D转换结果大于或等于10位CMPD(ADCMPRx[25:16])，内部匹配计数器加1. 0=设置比较条件为当10位A/D转换结果小于10位CMPD(ADCMPRx[25:16])，内部匹配计数器加1. **注:** 当内部匹配计数器的值达到(CMPMATCNT +1), CMPFx置位.
[1]	**CMPIE**	**比较中断使能** 1 = 使能 0 = 禁用 如果使能比较功能且比较条件满足CMPCOND 和 CMPMATCNT的设定, CMPF 位置位,同时，如果CMPIE置1, 将产生中断.
[0]	**CMPEN**	**比较使能** 1 = 比较功能使能. 0 = 比较功能禁止. 若该位置位，当A/D转换器完成指定通道的转换并将数据存到ADDR 寄存器.时，将自动进行CMPD[9:0] 与该转换值的比较

A/D 状态寄存器 (ADSR)

寄存器	偏移量	R/W	描述	复位后的值
ADSR	ADC_BA+0x30	R/W	ADC 状态寄存器	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:16]	OVERRUN	溢出标志位 该位是ADDRx寄存器的OVERRUN位的镜像 该位只读.
[15:8]	VALID	数据有效标志位 该位是ADDRx寄存器的VALID位的镜像 该位只读.
[7]	保留	保留
[6:4]	CHANNEL	当前转换通道 当BUSY=1时，此栏表示当前正在转换中的通道编号. 当BUSY=0时, 此栏表示下一个将进行转换的通道编号. 只读位.
[3]	BUSY	BUSY/IDLE 1 = A/D 转换器忙碌 0 = A/D 转换器空闲 该位是ADCR寄存器的ADST 位的镜像. 只读位

[2]	**CMPF1**	**比较标志位** 当所选择的通道转换结果和ADCMPR1寄存器所设定的比较条件相匹配，该位置位. 1 = ADDR 转换结果和 ADCMPR1寄存器所设定的比较条件相匹配 0 = ADDR 转换结果和 ADCMPR1寄存器所设定的比较条件不匹配
[1]	**CMPF0**	**比较标志位** 当所选择的通道转换结果和ADCMPR0寄存器所设定的比较条件相匹配，该位置位. 写 1 清该位. 1 = ADDR 转换结果和 ADCMPR0寄存器所设定的比较条件相匹配 0 = ADDR 转换结果和 ADCMPR0寄存器所设定的比较条件不匹配
[0]	**ADF**	**A/D 转换结束标志位** 用以指示A/D转换已結束的状态标志位 以下两种条件下ADF置1. 1. 单次模式下，A/D 转换结束. 2. 扫描模式下，在所有指定通道的A/D转换结束. 写1可清除该标志

A/D PDMA当前数据传输寄存器(ADPDMA)

寄存器	偏移量	R/W	描述	复位后的值
ADPDMA	ADC_BA+0x40	R	A/D PDMA当前数据传输寄存器	0x0000_0000

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

Bits	描述	
[31:12]	**保留**	保留
[11:0]	**AD_PDMA**	**ADC PDMA 当前数据传输寄存器** 当PDMA传输时，读该寄存器可以监测当前PDMA传输数据. 该位只读.

5.18 PDMA 控制器 (PDMA)

5.18.1 概述

NuMicro™ NUC123 DMA包括6个通道的直接存储器访问(PDMA) 控制器和一个循环冗余检查发生器。

PDMA可以用于和内存之间的数据传输或者和APB设备之间的数据传输。PDMA通道(PDMA CH0~CH5)有一个字的缓冲用于APB外设和内存之间的数据传输。 软件可以通过关闭PDMA [PDMACEN]来停止PDMA操作。当CPU收到PDMA中断或者通过软件轮询可以知道一次PDMA操作完成。PDMA控制器可以支持增加源和目标地址，也支持源和目标地址固定.

PDMA控制器包含一个循环冗余检查发生器，可以实现CRC算法，生成多项式可以编程。CRC发生器支持CPU PIO模式和DMA传输模式

5.18.2 特性

- 支持6个PDMA通道和一个CRC通道。每个PDMA通道支持一个单向的传输
- AMBA AHB 主/从接口兼容, 用于数据传输和寄存器读/写
- 硬件循环优先级机制，DMA 通道0有最高优先级
- PDMA
  - ✧ 外设到内存, 内存到外设, 内存到内存传输
  - ✧ 支持字/半字/字节传输数据宽度，从/到外设
  - ✧ 支持地址方向：增加，固定
- 循环冗余检查(CRC)
  - ✧ 支持4种常用多项式CRC-CCITT, CRC-8, CRC-16, 和 CRC-32
    -- CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$
    -- CRC-8: $X^8 + X^2 + X + 1$
    -- CRC-16: $X^{16} + X^{15} + X^2 + 1$
    -- CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
  - ✧ 种子值可编程
  - ✧ 输入数据和CRC校验支持顺序反向设置可编程
  - ✧ 输入数据和CRC校验支持1的补码设定(1's complement setting) 可编程
  - ✧ 支持 CPU PIO 模式或者 DMA 传输模式
  - ✧ CPU PIO模式下，支持8/16/32-bit 数据宽度
    -- 8-bit 写模式: 1-AHB 时钟周期
    -- 16-bit写模式: 2-AHB时钟周期
    -- 32-bit写模式: 4-AHB时钟周期
  - ✧ CRC DMA模式下，支持字节对齐的传输长度，就是说传输长度以字节为单位

5.18.3 框图

DMA 方块图，DMA 时钟控制方块图和 CRC 方块图如下所示

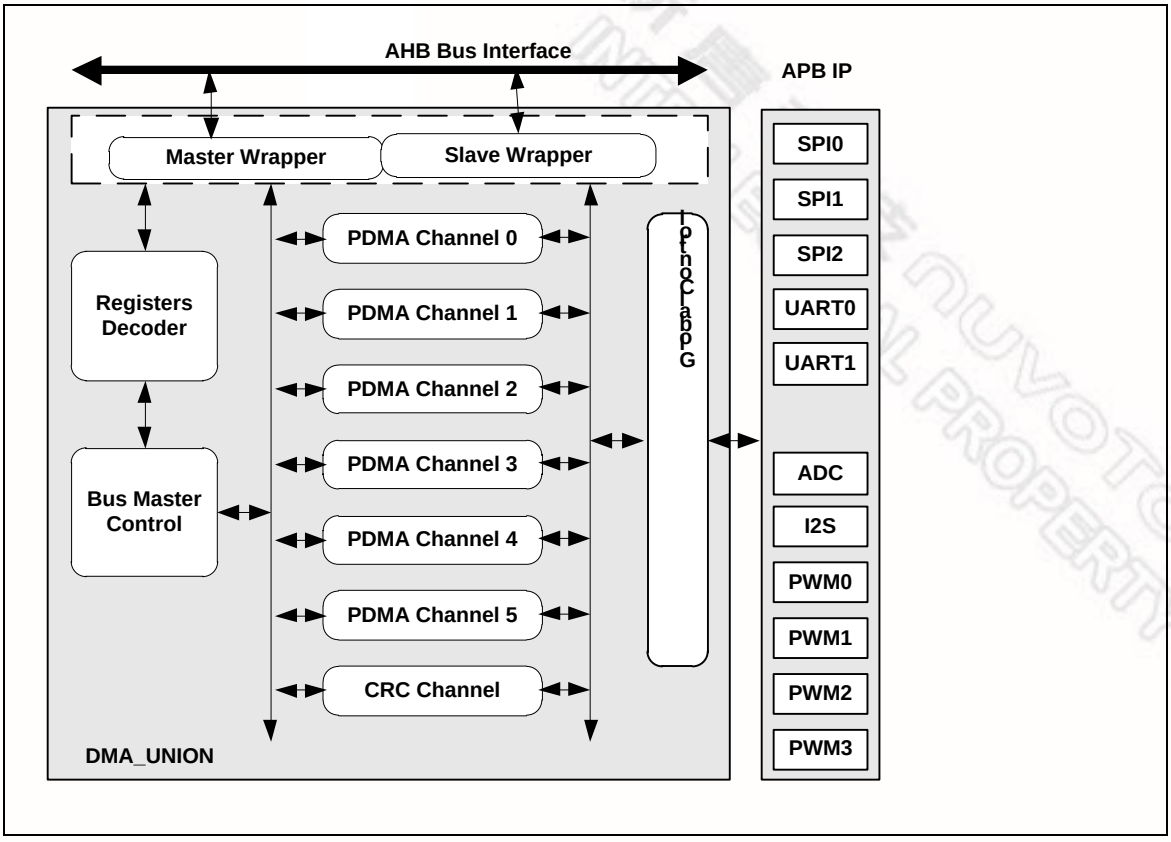


图 5-108 PDMA 控制器框图

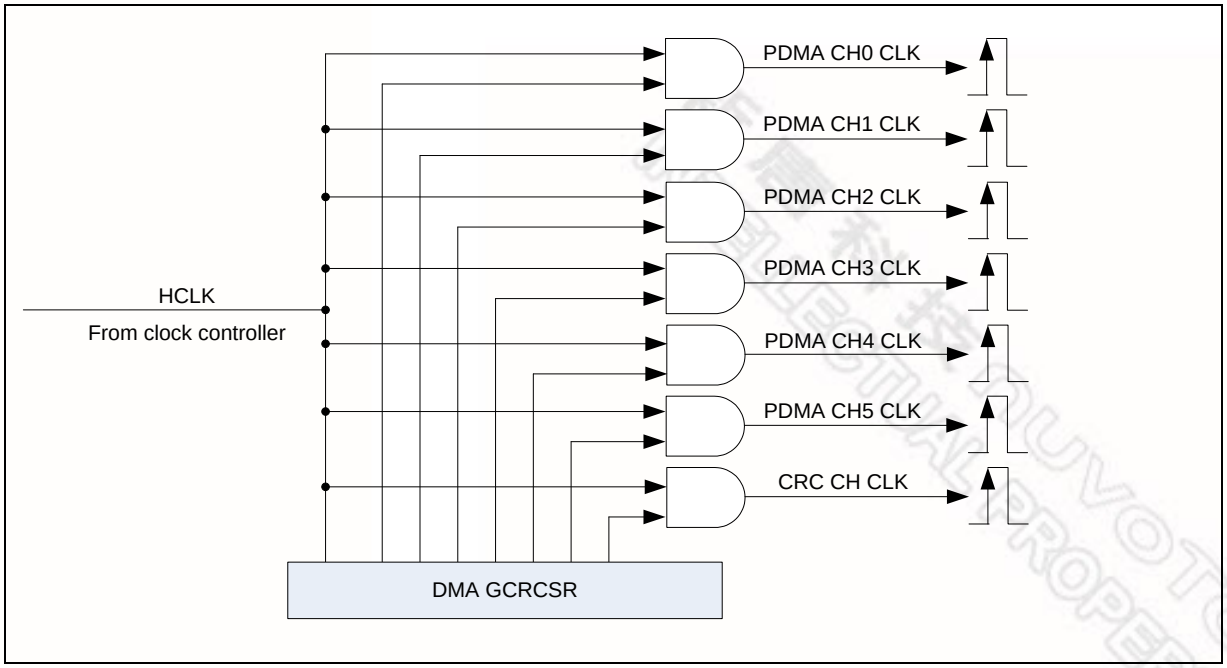


图 5-109 DMA 时钟控制器方块图

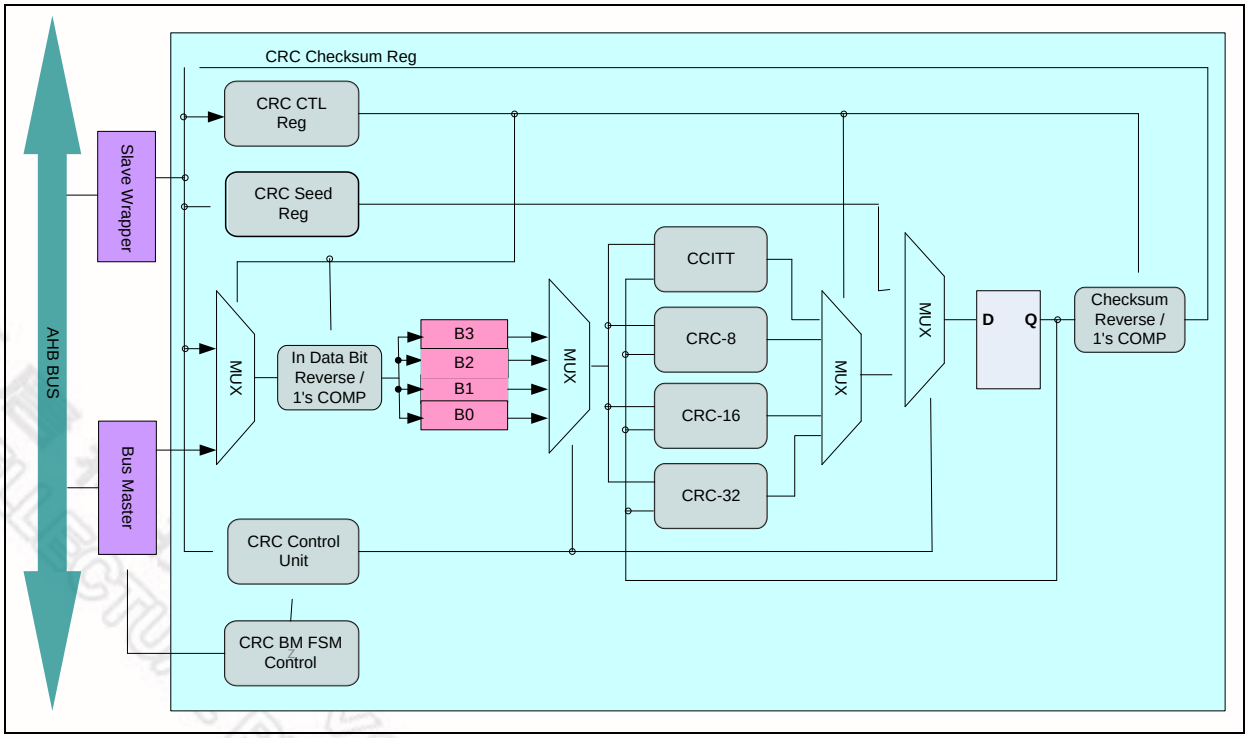


图 5-110 CRC 发生器方块图

5.18.4 功能描述

直接内存访问(DMA)控制器模块从一个地址到另一个地址传输数据，不需要CPU的参与。DMA 控制器包含6 路PDMA(外设-到-内存、内存-到-外设 或 内存-到-内存)通道和1个CRC发生器通道。

CPU 通过软件轮流或 接收到内部 PDMA 中断，可侦测到一次PDMA 操作完成。

每个PDMA 模拟通道没有先设定, 因此用户需通过设定PDMA_PDSSR0 和 PDMA_PDSSR1先行配置每路PDMA 通道.

✧ PDMA

DMA控制器有6路PDMA(外设-到-内存、内存-到-外设 或 内存-到-内存)通道。对于源和目标地址，PDMA控制器支持两种模式：增加和固定

每个PDMA通道没有默认的行为设定，所以用户在启动相应的PDMA通道之前，必须先配置通道的PDMA_PDSSR0, PDMA_PDSSR1 和 PDMA_PDSSR2寄存器

软件需通过设定PDMACEN位使能 DMA 通道，然后写有效的源地址到PDMA_SARx 寄存器，写目的地址到PDMA_DSABx 寄存器，传输数量到 PDMA_BCRx 寄存器。最后触发PDMA_CSRx [TRIG_EN]。PDMA将连续传输直到PDMA_CBCRx减到0，如果在PDMA操作期间产生错误，通道停止直到软件清除错误条件并设置PDMA_CSRx [SW_RST]复位PDMA通道，并且设置PDMA_CSRx [PDMACEN]和[TRIG_EN]重新开始.

在PDMA (外设-到-内存、内存-到-外设) 模式下, DMA 可以在外设 APB IP (ex: UART, SPI, ADC….) 和 内存间传递数据.

✧ CRC

DMA控制器包含一个循环冗余检查(CRC) 发生器，可以实现CRC算法，生成多项式可以编程。生成多项式包括CRC-CCITT, CRC-8, CRC-16 和 CRC-32。软件可以通过设定CRC_CTL寄存器的CRC_MODE域来选择生成多项式模式 。

CRC 引擎支持CPU PIO 模式 (CRC_CTL [CRCCEN] = 1, CRC_CTL [TRIG_EN] = 0) 和 DMA 传输模式 (CRC_CTL [CRCCEN] = 1, CRC_CTL [TRIG_EN] = 1). 下面是一个编程顺序的示例。

CPU PIO 操作模式：

- 通过设定CRC_CTL寄存器的CRCCEN位来使能CRC 引擎.
- 初始化设定。设定数据格式(通过设定CRC_CTL 寄存器的WDATA_RVS, CHECKSUM_RVS, WDATA_COM 和 CHECKSUM_COM)，初始化种子值(CRC_SEED)，通过设定CRC_CTL[CPU_WDLEN]寄存器选择数据长度.
- 通过设定CRC_CTL寄存器的CRC_RST位让CRC电路复位来载入种子值到CRC电路.
- 写数据到 CRC_WDATA 计算CRC.
- 通过读CRC_CHECKSUM寄存器取得CRC校验值

CRC DMA 操作模式:

- 通过设定CRC_CTL寄存器的CRCCEN位来使能CRC 引擎.
- 初始化设定。设定数据格式(通过设定CRC_CTL 寄存器的WDATA_RVS, CHECKSUM_RVS, WDATA_COM 和 CHECKSUM_COM)，初始化种子值 (CRC_SEED)

- 将有效的源地址和传输长度写到CRC_DMASAR 和CRC_DMABCR寄存器.
- 能 CRC_CTL [TRIG_EN] ，然后硬件将复位种子值并读内存数据做CRC校验
- 等 CRC DMA 传输和CRC 校验完成，读CRC_CHECKSUM 寄存器可以得到CRC校验值.

5.18.5 PDMA初始化范例

1. PDMA的时钟源只能选择HCLK，所以不用写CLKSELx寄存器选择时钟源
2. 写AHBCLK寄存器使能PDMA的时钟
3. 写DMA_GCRCSR寄存器使能相应通道的时钟
4. 写DMA_PDSSRx寄存器，选择相应功能使用的通道
5. 写PDMA_SARx寄存器设定源地址
6. 写PDMA_DARx寄存器设定目的地址
7. 写PDMA_BCRx寄存器设定传输字节数
8. 写PDMA_CSR寄存器，选择源地址和目的地址的模式，以及传输宽度，并使能PDMA功能
9. 写PDMA_IER寄存器使能PDMA中断
10. 调用NVIC_EnableIRQ(PDMA_IRQn)使能PDMA中断
11. 初始化需要PDMA传数据的IP，例如：UART并使能UART的PDMA功能
12. 写PDMA_CSR寄存器的TRIG_EN触发PDMA开始搬数据
13. 搬数据结束将发生中断PDMA中断

5.18.6 寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
PDMA 基地址: **PDMA_chx_BA = 0x5000_8000 + (0x100 * x)** x = 0, 1 .. 5 **CRC_BA = 0x5000_8E00** **DMA_GCR_BA = 0x5000_8F00**				
PDMA_CSRx	PDMA_BA_chx+0x00	R/W	PDMA 控制寄存器	0x0000_0000
PDMA_SARx	PDMA_BA_chx+0x04	R/W	PDMA 源地址寄存器	0x0000_0000
PDMA_DARx	PDMA_BA_chx+0x08	R/W	PDMA 目的地址寄存器	0x0000_0000
PDMA_BCRx	PDMA_BA_chx+0x0C	R/W	PDMA 发送字节数寄存器	0x0000_0000
PDMA_POINTx	PDMA_BA_chx+0x10	R	PDMA 内部缓冲指针	0x0404_0000
PDMA_CSARx	PDMA_BA_chx+0x14	R	PDMA 当前源地址寄存器	0x0000_0000
PDMA_CDARx	PDMA_BA_chx+0x18	R	PDMA 当前目的地址寄存器	0x0000_0000
PDMA_CBCRx	PDMA_BA_chx+0x1C	R	PDMA 当前传输字节计数寄存器	0x0000_0000
PDMA_IERx	PDMA_BA_chx+0x20	R/W	PDMA 中断使能寄存器	0x0000_0001
PDMA_ISRx	PDMA_BA_chx+0x24	R/W	PDMA 中断状态寄存器	0x0000_0000
PDMA_SBUF0_cx	PDMA_BA_chx+0x80	R	PDMA 共享缓冲 FIFO	0x0000_0000
CRC_CTL	CRC_BA+0x00	R/W	CRC 控制寄存器	0x2000_0000
CRC_DMASAR	CRC_BA+0x04	R/W	CRC DMA 源地址寄存器	0x0000_0000
CRC_DMABCR	CRC_BA+0x0C	R/W	CRC 传输字节数寄存器	0x0000_0000
CRC_DMACSAR	CRC_BA+0x14	R/W	CRC 当前源地址寄存器	0x0000_0000
CRC_DMACBCR	CRC_BA+0x1C	R/W	CRC 当前传输字节计数寄存器	0x0000_0000
CRC_DMAIER	CRC_BA+0x20	R/W	CRC 中断使能寄存器	0x0000_0001
CRC_DMAISR	CRC_BA+0x24	R/W	CRC 中断状态寄存器	0x0000_0000
CRC_WDATA	CRC_BA+0x80	R/W	CRC 写数据寄存器	0x0000_0000
CRC_SEED	CRC_BA+0x84	R/W	CRC 种子寄存器	0xFFFF_FFFF
CRC_CHECKSUM	CRC_BA+0X88	R	CRC 校验值寄存器	0x0000_0000
DMA_GCRCSR	PDMA_BA_GCR+0x00	R/W	DMA全局控制寄存器	0x0000_0000
DMA_PDSSR0	PDMA_BA_GCR+0x04	R/W	DMA 服务选择控制寄存器 0	0X00FF_FFFF

DMA_PDSSR1	PDMA_BA_GCR+0x08	R/W	DMA 服务选择控制寄存器 1	0x0FFF_FFFF
DMA_GCRISR	PDMA_BA_GCR+0x0C	R/W	DMA全局中断寄存器	0x0000_0000
DMA_PDSSR2	PDMA_BA_GCR+0x10	R/W	DMA 服务选择控制寄存器2	0x00FF_FFFF

5.18.7 寄存器描述

PDMA 控制和状态寄存器 (PDMA_CSRx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_CSR0	PDMA_BA_ch0+0x00	R/W	PDMA 控制和状态寄存器 CH0	0x0000_0000
PDMA_CSR1	PDMA_BA_ch1+0x00	R/W	PDMA 控制和状态寄存器 CH1	0x0000_0000
PDMA_CSR2	PDMA_BA_ch2+0x00	R/W	PDMA 控制和状态寄存器 CH2	0x0000_0000
PDMA_CSR3	PDMA_BA_ch3+0x00	R/W	PDMA 控制和状态寄存器 CH3	0x0000_0000
PDMA_CSR4	PDMA_BA_ch4+0x00	R/W	PDMA 控制和状态寄存器 CH4	0x0000_0000
PDMA_CSR5	PDMA_BA_ch5+0x00	R/W	PDMA 控制和状态寄存器 CH5	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TRIG_EN	保留		**APB_TWS**		保留		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
保留							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DAD_SEL		**SAD_SEL**		**MODE_SEL**		**SW_RST**	**PDMACEN**

Bits	描述	
[31:24]	**保留**	保留
[23]	**TRIG_EN**	**TRIG_EN** 1 = 使能 PDMA 数据读写传输. 0 = 无效. **注:**当 PDMA 传输完成, 该位自动清除. 如果总线 错误 发生, 所有 PDMA 传输将停止. 软件必须复位所有 PDMA 通道, 然后重新触发.
[22:21]	**保留**	保留
[20:19]	**APB_TWS**	**外设传输宽度选择** 00 = PDMA传输以一个字(32位)为单位. 01 = PDMA传输以一个字节(8位)为单位. 10 = PDMA传输以一个半字 (16位) 为单位. 11 = 保留. **注:** 该域只有当**MODE_SEL**选择外设到内存模式 (Peripheral-to-Memory)或者内存到 外设 模式 (Memory-to-Peripheral)才有效.

[18:8]	**保留**	保留
[7:6]	**DAD_SEL**	**传输目的地址方向选择** 00 =传输目的地址 持续增加. 01 = 保留. 10 =传输目的地址固定(适用于传输多个源地址数据到单一目的地址). 11 = 保留.
[5:4]	**SAD_SEL**	**传输源地址方向选择** 00 =传输源地址 持续增加. 01 =保留. 10 = 传输源地址固定(适用于单源地址传递数据到多个目的地址). 11 = 保留.
[3:2]	**MODE_SEL**	**PDMA 模式选择** 00 =内存到内存模式 (Memory-to-Memory). 01 = 外设到内存模式 (Peripheral -to-Memory) 10 = 内存到外设模式 (Memory-to- Peripheral).
[1]	**SW_RST**	**软件复位** 0 = 该位写 0 无效. 1 = 写 1 将复位内部状态机、指针以及内部缓冲. 控制寄存器的内容将不被清除. 该位在几个时钟周期后自动清除.
[0]	**PDMACEN**	**PDMA 通道使能** 该位置位 使能 PDMA 操作. 如果该位被清除, PDMA 将忽略所有 PDMA 请求并且强制主机总线进入空闲状态. **注:** SW_RST(PDMA_CSRx[1], x= 0~8) 将清该位

PDMA 传输源地址寄存器 (PDMA_SARx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_SAR0	PDMA_BA_ch0+0x04	R/W	PDMA 传输源地址寄存器 CH0	0x0000_0000
PDMA_SAR1	PDMA_BA_ch1+0x04	R/W	PDMA 传输源地址寄存器 CH1	0x0000_0000
PDMA_SAR2	PDMA_BA_ch2+0x04	R/W	PDMA 传输源地址寄存器 CH2	0x0000_0000
PDMA_SAR3	PDMA_BA_ch3+0x04	R/W	PDMA 传输源地址寄存器 CH3	0x0000_0000
PDMA_SAR4	PDMA_BA_ch4+0x04	R/W	PDMA 传输源地址寄存器 CH4	0x0000_0000
PDMA_SAR5	PDMA_BA_ch5+0x04	R/W	PDMA 传输源地址寄存器 CH5	0x0000_0000

注: Low Density系列仅支持PDMA 通道 0.

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

Bits	描述	
[31:0]	**PDMA_SAR**	**PDMA 传输源地址寄存器** 32位 PDMA源地址. **注：**源地址必须字对齐

PDMA 传输目的地址寄存器 (PDMA_DARx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_DAR0	PDMA_BA_ch0+0x08	R/W	PDMA 传输目的地址寄存器 CH0	0x0000_0000
PDMA_DAR1	PDMA_BA_ch1+0x08	R/W	PDMA 传输目的地址寄存器 CH1	0x0000_0000
PDMA_DAR2	PDMA_BA_ch2+0x08	R/W	PDMA 传输目的地址寄存器 CH2	0x0000_0000
PDMA_DAR3	PDMA_BA_ch3+0x08	R/W	PDMA 传输目的地址寄存器 CH3	0x0000_0000
PDMA_DAR4	PDMA_BA_ch40x08	R/W	PDMA 传输目的地址寄存器 CH4	0x0000_0000
PDMA_DAR5	PDMA_BA_ch5+0x08	R/W	PDMA 传输目的地址寄存器 CH5	0x0000_0000

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

Bits	描述	
[31:0]	**PDMA_DAR**	**PDMA 传输目的地址寄存器** 32位 PDMA目的地址. **注：**目的地址必须字对齐

PDMA 传输字节数寄存器 (PDMA_BCRx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_BCR0	PDMA_BA_ch0+0x0C	R/W	PDMA 传输字节数寄存器 CH0	0x0000_0000
PDMA_BCR1	PDMA_BA_ch1+0x0C	R/W	PDMA 传输字节数寄存器 CH1	0x0000_0000
PDMA_BCR2	PDMA_BA_ch2+0x0C	R/W	PDMA 传输字节数寄存器 CH2	0x0000_0000
PDMA_BCR3	PDMA_BA_ch3+0x0C	R/W	PDMA 传输字节数寄存器 CH3	0x0000_0000
PDMA_BCR4	PDMA_BA_ch4+0x0C	R/W	PDMA 传输字节数寄存器 CH4	0x0000_0000
PDMA_BCR5	PDMA_BA_ch5+0x0C	R/W	PDMA 传输字节数寄存器 CH5	0x0000_0000

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**PDMA_BCR**	**PDMA 传输字节数寄存器** 16位PDMA 传输字节数寄存器.

PDMA 内部缓冲指针寄存器 (PDMA_POINTx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_POINT0	PDMA_BA_ch0+0x10	R	PDMA 内部缓冲指示寄存器 CH0	0xXXXX_0000
PDMA_POINT1	PDMA_BA_ch1+0x10	R	PDMA 内部缓冲指示寄存器 CH1	0xXXXX_0000
PDMA_POINT2	PDMA_BA_ch2+0x10	R	PDMA 内部缓冲指示寄存器 CH2	0xXXXX_0000
PDMA_POINT3	PDMA_BA_ch3+0x10	R	PDMA 内部缓冲指示寄存器 CH3	0xXXXX_0000
PDMA_POINT4	PDMA_BA_ch4+0x10	R	PDMA 内部缓冲指示寄存器 CH4	0xXXXX_0000
PDMA_POINT5	PDMA_BA_ch5+0x10	R	PDMA 内部缓冲指示寄存器 CH5	0xXXXX_0000

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

Bits	描述	
[31:2]	**保留**	保留
[1:0]	**PDMA_POINT**	**PDMA 内部缓冲指针寄存器 (只读)** 指示内部缓冲指针.

PDMA当前源地址寄存器(PDMA_CSARx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_CSAR0	PDMA_BA_ch0+0x14	R	PDMA 当前源地址寄存器 CH0	0x0000_0000
PDMA_CSAR1	PDMA_BA_ch1+0x14	R	PDMA 当前源地址寄存器 CH1	0x0000_0000
PDMA_CSAR2	PDMA_BA_ch2+0x14	R	PDMA 当前源地址寄存器 CH2	0x0000_0000
PDMA_CSAR3	PDMA_BA_ch3+0x14	R	PDMA 当前源地址寄存器 CH3	0x0000_0000
PDMA_CSAR4	PDMA_BA_ch4+0x14	R	PDMA 当前源地址寄存器 CH4	0x0000_0000
PDMA_CSAR5	PDMA_BA_ch5+0x14	R	PDMA 当前源地址寄存器 CH5	0x0000_0000

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

Bits	描述	
[31:0]	**PDMA_CSAR**	**PDMA 当前源地址寄存器 (只读)** 这个域表示PDMA 正在传递的源地址.

PDMA 当前目的地址寄存器((PDMA_CDARx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_CDAR0	PDMA_BA_ch0+0x18	R	PDMA当前目的地址寄存器CH0	0x0000_0000
PDMA_CDAR1	PDMA_BA_ch1+0x18	R	PDMA当前目的地址寄存器CH1	0x0000_0000
PDMA_CDAR2	PDMA_BA_ch2+0x18	R	PDMA当前目的地址寄存器CH2	0x0000_0000
PDMA_CDAR3	PDMA_BA_ch3+0x18	R	PDMA当前目的地址寄存器CH3	0x0000_0000
PDMA_CDAR4	PDMA_BA_ch4+0x18	R	PDMA当前目的地址寄存器CH4	0x0000_0000
PDMA_CDAR5	PDMA_BA_ch5+0x18	R	PDMA当前目的地址寄存器CH5	0x0000_0000

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

Bits	描述	
[31:0]	**PDMA_CDAR**	**PDMA 当前目的地址寄存器(只读)** 这个域表示PDMA 正在传递的目的地址.

PDMA 当前字节计数寄存器 (PDMA_CBCRx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_CBCR0	PDMA_BA_ch0+0x1C	R	PDMA当前字节计数寄存器CH0	0x0000_0000
PDMA_CBCR1	PDMA_BA_ch1+0x1C	R	PDMA当前字节计数寄存器CH1	0x0000_0000
PDMA_CBCR2	PDMA_BA_ch2+0x1C	R	PDMA当前字节计数寄存器CH2	0x0000_0000
PDMA_CBCR3	PDMA_BA_ch3+0x1C	R	PDMA当前字节计数寄存器CH3	0x0000_0000
PDMA_CBCR4	PDMA_BA_ch4+0x1C	R	PDMA当前字节计数寄存器CH4	0x0000_0000
PDMA_CBCR5	PDMA_BA_ch5+0x1C	R	PDMA当前字节计数寄存器CH5	0x0000_0000

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**PDMA_CBCR**	**PDMA 当前字节计数寄存器 (只读)** PDMA当前剩下的字节数. **注**: SW_RST 将清除该寄存器的值.

PDMA 中断使能控制寄存 (PDMA_IERx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_IER0	PDMA_BA_ch0+0x20	R/W	PDMA 中断使能控制寄存器 CH0	0x0000_0001
PDMA_IER1	PDMA_BA_ch1+0x20	R/W	PDMA 中断使能控制寄存器 CH1	0x0000_0001
PDMA_IER2	PDMA_BA_ch2+0x20	R/W	PDMA 中断使能控制寄存器 CH2	0x0000_0001
PDMA_IER3	PDMA_BA_ch3+0x20	R/W	PDMA 中断使能控制寄存器 CH3	0x0000_0001
PDMA_IER4	PDMA_BA_ch4+0x20	R/W	PDMA 中断使能控制寄存器 CH4	0x0000_0001
PDMA_IER5	PDMA_BA_ch5+0x20	R/W	PDMA 中断使能控制寄存器 CH5	0x0000_0001

.

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

Bits	描述	
[31:2]	**保留**	保留
[1]	**BLKD_IE**	**PDMA 传输完成中断使能** 1 = 使能 PDMA 传递完成中断 0 = 禁止 PDMA 传递完成中断
[0]	**TABORT_IE**	**PDMA 读/写 异常中断使能** 1 = 使能 PDMA 传输异常中断 0 = 禁止 PDMA 传输异常中断.

PDMA中断状态寄存器 (PDMA_ISRx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_ISR0	PDMA_BA_ch0+0x24	R/W	PDMA 中断状态寄存器 CH0	0x0000_0000
PDMA_ISR1	PDMA_BA_ch1+0x24	R/W	PDMA 中断状态寄存器 CH1	0x0000_0000
PDMA_ISR2	PDMA_BA_ch2+0x24	R/W	PDMA 中断状态寄存器 CH2	0x0000_0000
PDMA_ISR3	PDMA_BA_ch3+0x24	R/W	PDMA 中断状态寄存器 CH3	0x0000_0000
PDMA_ISR4	PDMA_BA_ch4+0x24	R/W	PDMA 中断状态寄存器 CH4	0x0000_0000
PDMA_ISR5	PDMA_BA_ch5+0x24	R/W	PDMA 中断状态寄存器 CH5	0x0000_0000

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

Bits	描述	
[31:2]	**保留**	保留
[1]	**BLKD_IF**	**块传输完成 中断标志位** 该位表示PDMA完成所有传输. 1 = 完成. 0 = 未完成. **注:** 软件可写1清该位为0
[0]	**TABORT_IF**	**PDMA 读/写 目标异常中断标志位** 1 = 收到总线 错误应答. 0 = 没有收到总线错误应答. **注:** 软件可写1清该位为0

注: PDMA_ISR [TABORT_IF] 表示总线主控是否接收到错误响应。如果总线主控接收到错误响应，意味着目标异常发生，PDMA 将停止传输，响应该事件，并进入空闲状态。当目标异常发生，软件必须复位PDMA, 并重传这些数据.

PDMA共享缓冲FIFO (PDMA_SBUF0_cx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_SBUF0_c0	PDMA_BA_ch0+0x080	R	PDMA 共享缓冲 FIFO 0 寄存器 CH0	0x0000_0000
PDMA_SBUF0_c1	PDMA_BA_ch1+0x180	R	PDMA 共享缓冲 FIFO 0 寄存器 CH1	0x0000_0000
PDMA_SBUF0_c2	PDMA_BA_ch2+0x280	R	PDMA 共享缓冲 FIFO 0 寄存器 CH2	0x0000_0000
PDMA_SBUF0_c3	PDMA_BA_ch3+0x380	R	PDMA 共享缓冲 FIFO 0 寄存器 CH3	0x0000_0000
PDMA_SBUF0_c4	PDMA_BA_ch4+0x480	R	PDMA 共享缓冲 FIFO 0 寄存器 CH4	0x0000_0000
PDMA_SBUF0_c5	PDMA_BA_ch5+0x580	R	PDMA 共享缓冲 FIFO 0 寄存器 CH5	0x0000_0000

31	30	29	28	27	26	25	24
PDMA_SBUF0 [31:24]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PDMA_SBUF0 [23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PDMA_SBUF0 [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDMA_SBUF0 [7:0]							

Bits	描述	
[31:0]	**PDMA_SBUF0**	**PDMA共享缓冲 FIFO (只读)** 每个通道拥有专属的1个字的内部缓存.

CRC 控制寄存器 (CRC_CTL)

寄存器	偏移量	R/W	描述	复位后的值
CRC_CTL	CRC_BA+0x00	R/W	CRC 控制寄存器	0x2000_0000

31	30	29	28	27	26	25	24
CRC_MODE		CPU_WDLEN		CHECKSUM_COM	WDATA_COM	CHECKSUM_RVS	WDATA_RVS
23	22	21	20	19	18	17	16
TRIG_EN	保留						
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留						CRC_RST	CRCCEN

Bits	描述	
[31:30]	CRC_MODE	**CRC 多项式模式** 00 = CRC-CCITT 多项式模式 01 = CRC-8 多项式模式 10 = CRC-16 多项式模式 11 = CRC-32 多项式模式e
[29:28]	CPU_WDLEN	**CPU 写数据长度** 当CPU在PIO模式时(CRCCEN= 1, TRIG_EN = 0), 该域指示写数据长度. 00 = 数据长度为 8-bit 模式 01 = 数据长度为16-bit 模式 1x = 数据长度为 32-bit 模式 **注1:** 该域只用于 CPU PIO 模式. **注2:**当数据长度为 8-bit模式时，有效数据为 CRC_WDATA [7:0]；如果数据长度为 16-bit 模式时，有效数据为 CRC_WDATA [15:0].
[27]	CHECKSUM_COM	**校验补码** 1 = CRC 校验使用1的补码. 0 = CRC 校验不使用按位取反.
[26]	WDATA_COM	**写数据补码** 1 = CRC 写入的数据使用1的补码. 0 = CRC 写入的数据不使用按位取反.

[25]	**CHECKSUM_RVS**	**校验值反转** 1 = CRC 校验结果位顺序反向. 0 = CRC 校验结果不使用位顺序反向. **注:** 如果校验值是 0XDD7B0F2E, CRC校验值位顺序反向就是0x74F0DEBB.
[24]	**WDATA_RVS**	**写数据顺序反转** 1 = CRC 写入数据位顺序反向 (按字节反向) 0 = CRC 写入数据不使用位顺序反向. **注:** 如果写输入的数据是 0xAABBCCDD, CRC 写入数据位顺序反向就是 0x55DD33BB
[23]	**TRIG_EN**	**TRIG_EN** 1 = CRC DMA 数据读或者写传输使能. 0 = 无作用. **注1:** 如果改位置位，表示CRC引擎工作在CRC DMA模式，CRC_WDATA寄存器不要写入任何数据. **注2:** 当 CRC DMA 传输完成时, 该位将被自动清0 **注3:** 如果总线错误发生, 所有 CRC DMA 传输将停止。软件必须复位所有的 DMA 通达，然后再次触发.
[22:2]	**保留**	保留
[1]	**CRC_RST**	**CRC 引擎复位** 0 = 无作用. 1 = 复位内部 CRC 状态机和内部缓存。控制寄存器的内如不会被清除。几个时钟周期之后该位自动清0. **注:** CPU PIO 模式下, 设置该位将导致初始种子值被重新加载
[0]	**CRCCEN**	**CRC 通道使能** 设置该位为 1 使能 CRC's 操作. CRC DMA 模式下 (TRIG_EN = 1)，如果用户清除该位，DMA 操作将继续，直到CRC DMA操作完成，TRIG_EN 位也将有效直到所有 CRC DMA 操作完成 。但是这种情况下，CRC_DMAISR [BLKD_IF] 标志将不会置位。当TRIG_EN等于0时，用户可以读CRC_CHECKSUM寄存器得到CRC校验结果 CRC DMA 模式下 (TRIG_EN = 1)，如果用户想立即停止传输，可以写1到 CRC_RST 位来停止传输

CRC DMA 传输源地址寄存器 (CRC_DMASAR)

寄存器	偏移量	R/W	描述	复位后的值
CRC_DMASAR	CRC_BA+0x04	R/W	CRC DMA 传输源地址寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CRC_DMASAR	**CRC DMA 传输源地址寄存器** CRC DMA模式下， 指示32-bit 源地址. **注:**源地址必须是字对齐的.

CRC DMA 传输字节数寄存器(CRC_DMABCR)

寄存器	偏移量	R/W	描述	复位后的值
CRC_DMABCR	CRC_BA+0x0C	R/W	CRC DMA 传输字节数寄存器	0x0000_0000

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**CRC_DMABCR**	**CRC DMA 传输字节数寄存器** CRC DMA模式下，此16比特用来指示传输字节数.

CRC DMA 当前源地址寄存器 (CRC_DMACSAR)

寄存器	偏移量	R/W	描述	复位后的值
CRC_DMACSAR	CRC_BA+0x14	R	CRC DMA 当前源地址寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CRC_DMACSAR	**CRC DMA 当前源地址寄存器 (只读)** CRC DMA 当前传输的源地址.

CRC DMA 当前字节计数寄存器 (CRC_DMACBCR)

寄存器	偏移量	R/W	描述	复位后的值
CRC_DMACBCR	CRC_BA+0x1C	R	CRC DMA 当前字节计数寄存器	0x0000_0000

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

Bits	描述	
[31:16]	**保留**	保留
[15:0]	**CRC_DMACBCR**	**CRC DMA 当前字节计数寄存器 (只读)** CRC_DMA模式下当前剩下的未传输的字节数. **注:** CRC_RST 将清除该寄存器的值.

CRC DMA 中断使能控制寄存器 (CRC_DMAIER)

寄存器	偏移量	R/W	描述	复位后的值
CRC_DMAIER	CRC_BA+0x20	R/W	CRC DMA 中断使能控制寄存器	0x0000_0001

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

Bits	描述	
[31:2]	保留	保留
[1]	BLKD_IE	**CRC DMA 传输完成中断使能** 1 = 使能CRC DMA传输完成中断. 0 = 禁止CRC DMA传输完成中断.
[0]	TABORT_IE	**CRC DMA 读/写 目标异常中断使能** 1 = 使能CRC DMA 传输目标异常中断. 0 = 禁止CRC DMA 传输目标异常中断.

CRC DMA 中断状态寄存器 (CRC_DMAISR)

寄存器	偏移量	R/W	描述	复位后的值
CRC_DMAISR	CRC_BA+0x24	R/W	CRC DMA 中断状态寄存器	0x0000_0000

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

Bits	描述	
[31:2]	**保留**	保留
[1]	**BLKD_IF**	**块传输完成中断标志** CRC DMA 所有传输已经完成. 1 = 完成 0 = 没有完成. 软件可以写1清该位为0.
[0]	**TABORT_IF**	**CRC DMA 读/写 目标异常中断标志** 1 = 收到总线错误响应. 0 = 没有收到总线错误响应. 软件可以写1清该位为0.

注: CRC_DMAISR [TABORT_IF] 表示总线主控是否接收到错误响应。如果总线主控接收到错误响应，意味着目标异常发生， DMA 将停止传输，响应该事件，并进入空闲状态。当目标异常发生，软件必须复位DMA, 并重传这些数据.

CRC 写数据寄存器 (CRC_WDATA)

寄存器	偏移量	R/W	描述	复位后的值
CRC_WDATA	CRC_BA+0x80	R/W	CRC 写数据寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CRC_WDATA	**CRC 写数据寄存器** CPU PIO (CRC_CTL [CRCCEN] = 1, CRC_CTL [TRIG_EN] = 0) 模式下，软件可以写数据到这个域来计算CRC; CRC DMA 模式下 (CRC_CTL [CRCCEN] = 1, CRC_CTL [TRIG_EN] = 0)，该域将被用作DMA内部缓存. **注1:** CRC DMA 模式下, 不要写任何数据到这个寄存器 **注2:** CRC_CTL [WDATA_COM] 和 CRC_CTL [WDATA_RVS] 位设定将影响这个域。例如：如果WDATA_RVS = 1，写数据0xAABBCCDD 到 CRC_WDATA 寄存器，读该寄存器将得到0x55DD33BB.

CRC 种子寄存器 (CRC_SEED)

寄存器	偏移量	R/W	描述	复位后的值
CRC_SEED	CRC_BA+0x84	R/W	CRC 种子寄存器	0xFFFF_FFFF

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

Bits	描述	
[31:0]	CRC_SEED	CRC 种子寄存器 该寄存器为CRC种子值，就是CRC的初始值.

CRC 校验值寄存器 (CRC_CHECKSUM)

寄存器	偏移量	R/W	描述	复位后的值
CRC_CHECKSUM	CRC_BA+0x88	R	CRC 校验值寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CRC_CHECKSUM	**CRC 校验值寄存器** 该寄存器存放计算得到的CRC校验值.

DMA 全局控制和状态寄存器 (PDMA_GCRCSR)

寄存器	偏移量	R/W	描述	复位后的值
DMA_GCRCSR	DMA_BA_GCR+0x00	R/W	DMA全局控制和状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							CRC_CLK_EN
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留		CLK5_EN	CLK4_EN	CLK3_EN	CLK2_EN	CLK1_EN	CLK0_EN
7	6	5	4	3	2	1	0
保留							

Bits	描述	
[31:25]	保留	保留
[24]	CRC_CLK_EN	**CRC 控制器时钟使能控制** 0 = 禁止. 1 = 使能.
[23:14]	保留	保留
[13]	CLK5_EN	**PDMA 控制器通道 5 时钟使能控制** 0 = 禁用 1 = 使能
[12]	CLK4_EN	**PDMA 控制器通道 4 时钟使能控制** 0 = 禁用 1 = 使能
[11	CLK3_EN	**PDMA 控制器通道 3 时钟使能控制** 0 = 禁用 1 = 使能
[10	CLK2_EN	**PDMA 控制器通道 2 时钟使能控制** 0 = 禁用 1 = 使能

[9]	**CLK1_EN**	**PDMA 控制器通道 1 时钟使能控制** 0 = 禁用 1 = 使能
[8]	**CLK0_EN**	**PDMA 控制器通道 0 时钟使能控制** 0 = 禁用 1 = 使能
[7:0]	**保留**	保留

DMA 服务选择控制寄存器 0 (DMA_PDSSR0)

寄存器	偏移量	R/W	描述	复位后的值
DMA_PDSSR0	DMA_BA_GCR+0x04	R/W	DMA 服务选择控制寄存器 0	0x00FF_FFFF

31	30	29	28	27	26	25	24
保留							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SPI2_TXSEL				**SPI2_RXSEL**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPI1_TXSEL				**SPI1_RXSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI0_TXSEL				**SPI0_RXSEL**			

Bits	描述	
[31:24]	**保留**	保留
[23:20]	**SPI2_TXSEL**	**PDMA SPI2 TX 选择** 该域定义片上外设SPI2 TX连接到哪一路PDMA 通道. 软件通过设定SPI2_TXSEL配置 TX 信道. 该通道配置与SPI0_RXSEL相同. 可以参考SPI0_RXSEL的说明.
[19:16]	**SPI2_RXSEL**	**PDMA SPI2 RX 选择** 该域定义片上外设SPI2 RX连接到哪一路PDMA 通道. 软件通过设定SPI2_RXSEL配置RX 通道. 该通道配置与SPI0_RXSEL相同. 可以参考SPI0_RXSEL的说明.
[15:12]	**SPI1_TXSEL**	**PDMA SPI1 TX 选择** 该域定义片上外设SPI1 TX连接到哪一路PDMA 通道. 软件通过设定SPI1_TXSEL配置TX 通道. 该通道配置与SPI0_RXSEL相同. 可以参考SPI0_RXSEL的说明.
[11:8]	**SPI1_RXSEL**	**PDMA SPI1 RX 选择** 该域定义片上外设SPI1 RX连接到哪一路PDMA 通道. 软件通过设定SPI1_RXSEL配置RX 通道. 该通道配置与SPI0_RXSEL相同. 可以参考SPI0_RXSEL的说明.
[7:4]	**SPI0_TXSEL**	**PDMA SPI0 TX 选择** 该域定义片上外设SPI0 TX连接到哪一路PDMA 通道. 软件通过设定SPI0_TXSEL配置 TX 通道. 该通道配置与 SPI0_RXSEL相同. 可以参考SPI0_RXSEL的说明.

[3:0]	**SPI0_RXSEL**	**PDMA SPI0 RX 选择** 该域定义片上外设SPI0 RX连接到哪一路PDMA 通道. 通过设定（SPI0_RXSEL）软件可以改变RX 通道 4’b0000: CH0 4’b0001: CH1 4’b0010: CH2 4’b0011: CH3 4’b0100: CH4 4’b0101: CH5 Others : 保留 注: Ex : SPI0_RXSEL = 4’b0100, 意味 SPI0_RX 和 PDMA_CH4相连接

DMA 服务选择控制寄存器 1 (DMA_PDSSR1)

寄存器	偏移量	R/W	描述	复位后的值
DMA_PDSSR1	DMA_BA_GCR+0x08	R/W	DMA 服务选择控制寄存器1	0x0FFF_FFFF

31	30	29	28	27	26	25	24
保留				**ADC_RXSEL**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
保留							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
UART1_TXSEL				**UART1_RXSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UART0_TXSEL				**UART0_RXSEL**			

Bits	描述	
[31:28]	**保留**	保留
[27:24]	**ADC_RXSEL**	**PDMA ADC RX 选择** 该域定义外设ADC RX与哪一路PDMA 通道相连. 软件通过设定 ADC_RXSEL可配置 RX通道. 该通道配置与 UART0_RXSEL相同. 可参考UART0_RXSEL的说明
[23:16]	**保留**	保留
[15:12]	**UART1_TXSEL**	**PDMA UART1 TX 选择** 该域定义外设UART1 TX与哪一路PDMA 通道相连. 软件通过设定UART1_TXSEL可配置 TX通道. 该通道配置与 UART0_RXSEL相同 . 可参考UART0_RXSEL的说明
[11:8]	**UART1_RXSEL**	**PDMA UART1 RX 选择** 该域定义外设UART1 RX与哪一路PDMA 通道相连. 软件通过设定UART1_RXSEL可配置 RX通道. 该通道配置与UART0_RXSEL相同. 可参考UART0_RXSEL的说明
[7:4]	**UART0_TXSEL**	**PDMA UART0 TX 选择** 该域定义外设UART0 TX与哪一路PDMA 通道相连. 软件通过设定UART0_TXSEL可配置TX通道. 该通道配置与 UART0_RXSEL相同 . 可参考UART0_RXSEL的说明

[3:0]	**UART0_RXSEL**	该域定义外设UART0 RX与哪一路PDMA 通道相连. 软件通过设定UART0_RXSEL可配置RX通道 4’b0000: CH0 4’b0001: CH1 4’b0010: CH2 4’b0011: CH3 4’b0100: CH4 4’b0101: CH5 Others : 保留 **注**: Ex : UART0_RXSEL = 4’b0100, 意味 UART0_RX和PDMA_CH4连接

DMA全局中断状态寄存器 (DMA_GCRISR)

寄存器	偏移量	R/W	描述	复位后的值
DMA_GCRISR	DMA_BA_GCR+0x0C	R	DMA 全局中断状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
INTR	保留						
23	22	21	20	19	18	17	16
保留							CRC_INTR
15	14	13	12	11	10	9	8
保留							INTR8
7	6	5	4	3	2	1	0
保留		INTR5	INTR4	INTR3	INTR2	INTR1	INTR0

Bits	描述	
[31]	INTR	**中断状态** 该位为 PDMA 控制器中断状态. **注:** 该位只读
[30:17]	保留	保留
[16]	CRC_INTR	**CRC 控制器中断状态** 该位为CRC 控制器中断状态. **注:** 该位只读.
[15:6]	保留	保留
[5]	INTR5	**通道5的中断状态** 该位是PDMA通道5的中断状态. 注: 该位只读
[4]	INTR4	**通道4的中断状态** 该位是PDMA通道4的中断状态. 注: 该位只读
[3]	INTR3	**通道3的中断状态** 该位是PDMA通道3的中断状态. 注: 该位只读

[2]	**INTR2**	**通道2的中断状态** 该位是PDMA通道2的中断状态. 注: 该位只读
[1]	**INTR1**	**通道1的中断状态** 该位是PDMA通道1的中断状态. 注: 该位只读
[0]	**INTR0**	**通道0的中断状态** 该位是PDMA通道0的中断状态. 注: 该位只读

DMA服务选择控制寄存器2 (DMA_PDSSR2)

寄存器	偏移量	R/W	描述	复位后的值
DMA_PDSSR2	DMA_BA_GCR+0x10	R/W	DMA服务选择控制寄存器2	0x00FF_FFFF

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
PWM3_RXSEL				PWM2_RXSEL			
15	14	13	12	11	10	9	8
PWM1_RXSEL				PWM0_RXSEL			
7	6	5	4	3	2	1	0
I2S_TXSEL				I2S_RXSEL			

Bits	描述	
[31:24]	保留	保留
[23:20]	PWM3_RXSEL	**PDMA PWM3 RX 选择** 该域定义片上外设PWM3 RX与哪一路PDMA 通道相连接. 软件通过设定PWM3_RXSEL可配置 RX通道. 该通道配置与 I2S_RXSEL相同. 可参考I2S_RXSEL的说明.
[19:16]	PWM2_RXSEL	**PDMA PWM2 RX 选择** 该域定义片上外设PWM2 RX与哪一路PDMA 通道相连接. 软件通过设定PWM2_RXSEL可配置 RX通道. 该通道配置与 I2S_RXSEL相同. 可参考I2S_RXSEL的说明.
[15:12]	PWM1_RXSEL	**PDMA PWM1 RX 选择** 该域定义片上外设PWM1 RX与哪一路PDMA 通道相连接. 软件通过设定PWM1_RXSEL可配置 RX通道. 该通道配置与 I2S_RXSEL相同. 可参考I2S_RXSEL的说明..
[11:8]	PWM0_RXSEL	**PDMA PWM0 RX 选择** 该域定义片上外设PWM0 RX与哪一路PDMA 通道相连接. 软件通过设定PWM0_RXSEL可配置 RX通道. 该通道配置与 I2S_RXSEL相同. 可参考I2S_RXSEL的说明.
[7:4]	I2S_TXSEL	**PDMA I^2S Tx选择** 该域定义片上外设**I^2S** TX与哪一路PDMA 通道相连接. 软件通过设定I2S_TXSEL可配置 TX通道. 该通道配置与 I2S_RXSEL相同. 可参考I2S_RXSEL的说明.

[3:0]	**I2S_RXSEL**	**PDMA I²S Rx 选择** 该域定义片上外设**I²S** RX与哪一路PDMA相连接. 可通过软件设定I2S_RXSEL改变 RX 通道 4'b0000: CH0 4'b0001: CH1 4'b0010: CH2 4'b0011: CH3 4'b0100: CH4 4'b0101: CH5 Others : 保留 注: Ex : I2S_RXSEL = 4'b0100, 意味 I2S_RX 和 PDMA_CH4相连接

6 ARM® CORTEX®-M0 内核

6.1 概述

Cortex®-M0处理器是32位可配置的多级流水线RISC处理器. 它有AMBA AHB-Lite接口和嵌套向量中断控制器（NVIC）. 具有可选的硬件调试功能. 可以执行Thumb指令，并与其它Cortex-M系列兼容。支持两种模式-Thread 模式 与 Handler 模式. 异常时系统进入Handler 模式。从Handler模式返回时，执行异常返回. 复位时系统进入Thread 模式。Thread 模式也可由异常返回时进入。 图 6-1 为处理器的功能图。

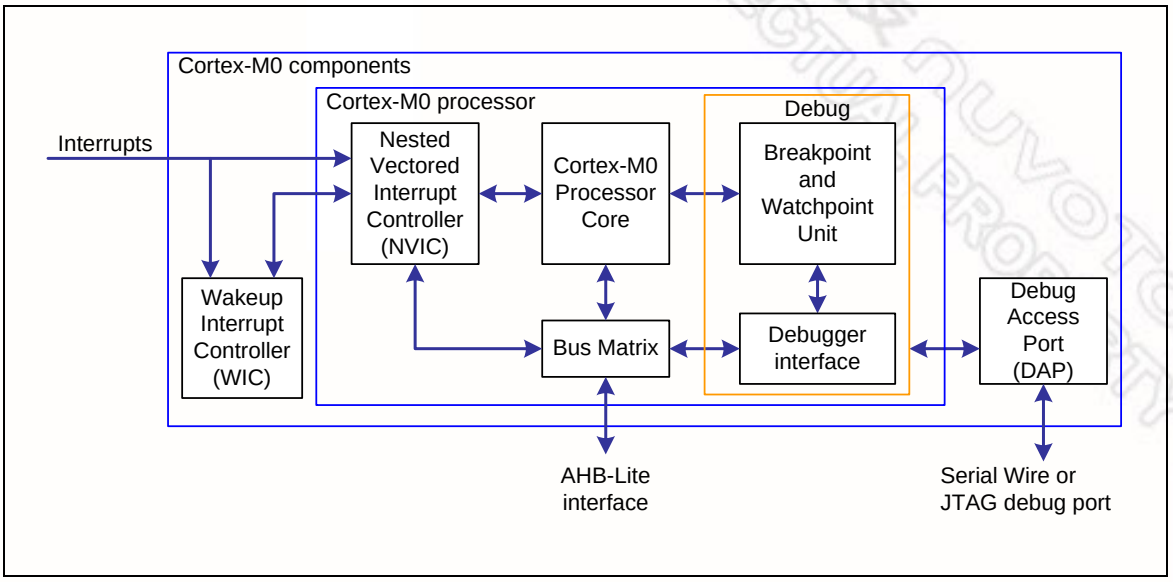


图 6-1 功能框图

6.2 特性

- 低门数处理器
  - ARMv6-M Thumb® 指令集
  - Thumb-2 技术
  - ARMv6-M 兼容 24-位 SysTick定时器
  - 32-位 硬件乘法器
  - 系统接口支持小端（little-endian）数据访问
  - 准确而及时的中断处理能力
  - 加载、存储多个数据和多周期乘法指令可被终止然后重新开始从而实现快速中断处理
  - C应用程序二进制接口兼容的异常模式（C-ABI）。这个ARMv6-M的模式允许用户使用纯C函数实现中断处理
  - 使用等待中断（WFI）与等待事件（WFE）指令进入低功耗的休眠模式，或者从中断退出时返回休眠模式

- NVIC
  - 32 个外部中断，每个中断具有4级优先级
  - 专用的不可屏蔽中断（NMI）.
  - 同时支持电平和脉冲中断触发
  - 中断唤醒控制器(WIC), 支持极低功耗休眠模式.
- 调试支持
  - 四个硬件断点.
  - 两个观察点.
  - 用于非侵入式代码分析的程序计数采样寄存器（PCSR）.
  - 单步和向量捕获能力.
- 总线接口:
  - 单个32位AMBA-3 ABH-Lite系统接口，提供简单方式集成所有系统外设和存储器.
  - 支持DAP(Debug Access Port)的单一32位的从机端口.

6.3 系统定时器 (SysTick)

Cortex-M0 包含一个系统定时器,：SysTick. SysTick 提供一种简单的24位写清零、递减、自装载同时具有可灵活控制机制的计数器。 该计数器可用作实时系统(RTOS) 的节拍定时器或一个简单的计数器.

使能后，定时器从SysTick 当前值寄存器(SYST_CVR)的值向下计数到0，并在下一个时钟周期，重新加载寄存器(SYST_RVR) 的值。当计数器减到0时，标志位COUNTFLAG置位，读COUNTFLAG位使其清零。

复位后，SYST_CVR 的值未知。使能定时器前，软件应该写该寄存器以使其清0. 这样确保定时器从SYST_RVR开始计数，而非任意值.

若SYST_RVR 是0 ，在重新加载后，定时器将保持当前值0。这个功能可以在定时器使能后除了定时器使能位之外用来禁止定时器的又一个方式。

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

6.3.1 系统定时器控制寄存器映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
SCS基地址： **SCS_BA = 0xE000_E000**				
SYST_CSR	SCS_BA+0x10	R/W	SysTick 控制与状态寄存器	0x0000_0000
SYST_RVR	SCS_BA+0x14	R/W	SysTick 重新加载值寄存器	0xXXXX_XXXX
SYST_CVR	SCS_BA+0x18	R/W	SysTick 当前值寄存器	0xXXXX_XXXX

6.3.2 系统定时器控制寄存器描述

SysTick控制与状态寄存器（SYST_CSR）

寄存器	偏移量	R/W	描述	复位后的值
SYST_CSR	SCS_BA+0x10	R/W	SysTick 控制与状态寄存器	0x0000_0000

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

Bits	描述	
[31:17]	保留	保留
[16]	**COUNTFLAG**	从上次该寄存器被读之後，如果定时器计数到0，则返回1. 计数由1到0时，COUNTFLAG 置位. 在读或写当前值寄存器（**SYST_CVR**）时，COUNTFLAG 被清零.
[15:3]	保留	保留
[2]	**CLKSRC**	1 =SysTick使用内核时钟. 0 = 时钟源为外部参考时钟
[1]	**TICKINT**	1 = 向下计数到0将引起SysTick 异常而挂起. 软件通过写动作清SysTick 当前值寄存器(SYST_CVR)将不会导致SysTick 挂起. 0 = 向下计数到0不会引起SysTick异常而挂起. 软件根据COUNTFLAG 来确定是否已经发生计数到0.
[0]	**ENABLE**	1 : 计数器运行于周期模式. 0 : 禁止计数器

SysTick重新加载值寄存器（SYST_RVR）

寄存器	偏移量	R/W	描述	复位后的值
SYST_RVR	SCS_BA+0x14	R/W	SysTick重新加载值寄存器	0xXXXX_XXXX

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

Bits	描述	
[31:24]	保留	保留
[23:0]	RELOAD	当计数器达到0时，这个值将加载到当前值寄存器

SysTick 当前值寄存器（SYST_CVR）

寄存器	偏移量	R/W	描述	复位后的值
SYST_CVR	SCS _BA+0x18	R/W	SysTick当前值寄存器	0xXXXX_XXXX

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

Bits	描述	
[31:24]	保留	保留
[23:0]	CURRENT	当前计数值，为被采样时刻的计数器的值，计数器不提供读-修改-写保护功能，该寄存器为write-clear.软件写入任何值将清该寄存器为0. 不支持RAZ(见SysTick重新加载寄存器)

6.4 系统控制寄存器

系统控制寄存器控制了Cortex®-M0的状态和操作模式，包括CPUID，Cortex-M0中断优先级和Cortex-M0电源管理

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

6.4.1 系统控制寄存器内存映射

R: read only, **W**: write only, **R/W**: both read and write

寄存器	偏移量	R/W	描述	复位后的值
SCS 基地址: **SCS_BA = 0xE000_E000**				
CPUID	SCS_BA+0xD00	R	CPUID寄存器	0x410C_C200
ICSR	SCS_BA+0xD04	R/W	中断控制和状态寄存器	0x0000_0000
AIRCR	SCS_BA+0xD0C	R/W	应用程序中断与复位控制寄存器	0xFA05_0000
SCR	SCS_BA+0xD10	R/W	系统控制寄存器	0x0000_0000
SHPR2	SCS_BA+0xD1C	R/W	系统处理程序优先级寄存器2	0x0000_0000
SHPR3	SCS_BA+0xD20	R/W	系统处理程序优先级寄存器3	0x0000_0000

6.4.2 系统控制寄存器

CPUID寄存器(CPUID)

寄存器	偏移量	R/W	描述	复位后的值
CPUID	SCS_BA+0xD00	R	CPUID寄存器	0x410C_C200

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

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

中断控制和状态寄存器(ICSR)

寄存器	偏移量	R/W	描述	复位后的值
ICSR	SCS_BA+0xD04	R/W	中断控制和状态寄存器	0x0000_0000

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

Bits	描述	
[31]	NMIPENDSET	NMI 设置未处理位 写： 0 = 该位写0无效 1 = 更改 NMI 异常状态为未处理，也就是触发一个NMI中断 读： 0 = 没有NMI 异常等待处理 1 = 有NMI 异常等待处理 因为 NMI 异常是最高优先级的异常，正常情况下处理器一旦检测到这一位被置1，就会立刻进入 NMI 异常处理程序。进入处理程序后处理器会将该位清为零。这意味着只有当处理器正在执行 NMI 异常处理程序时， NMI 信号再次产生，NMI 异常处理程序读取这一位的值才会返回1。
[30:29]	保留	保留
[28]	PENDSVSET	PendSV 设置未处理位 写： 0 = 该位写0无效 1 = 更改 PendSV 异常状态为未处理，也就是触发一个PendSV中断 读： 0 = 没有PendSV 异常等待处理 1 = 有PendSV 异常等待处理 向该位写1是将 PendSV 异常状态设为未处理的唯一方法
[27]	PENDSVCLR	PendSV 清除未处理位 写：

		0 = 该位写0无效 1 = 移除 PendSV 异常的未处理状态 只写位。当想清除 PENDSV 位，必须同时“向 PENDSVSET 写 0 和向 PENDSVCLR写 1”。
[26]	**PENDSTSET**	SysTick 异常设置未处理位 写： 0 = 该位写0无效 1 = 更改SysTick 异常状态为未处理 读： 0 = 没有SysTick 异常等待处理 1 = 有SysTick 异常等待处理
[25]	**PENDSTCLR**	SysTick 异常清除未处理位 写： 0 = 该位写0无效 1 = 移除 SysTick 异常的未处理状态 只写位，当想清除 PENDST 位，必须同时“向 PENDSTSET 写 0 和向 PENDSTCLR写 1”。
[24]	**保留**	保留
[23]	**ISRPREEMPT**	如果置位，由调试停止状态退出时，未处理的异常将被处理. 只读位
[22]	**ISRPENDING**	中断未处理标志，不包括 NMI 和 Faults 0 = 没有中断等待处理 1 = 有中断等待处理 只读位
[21:18]	**保留**	保留
[17:12]	**VECTPENDING**	表示最高优先级等待处理的异常的异常号 0 = 没有异常等待处理。 非0 =最高优先级等待处理的异常的异常号 只读位
[11:6]	**保留**	保留
[5:0]	**VECTACTIVE**	当前正在处理的异常的异常号 0 = 线程模式 非0 = 当前正在处理的异常的异常号. 只读位

应用程序中断和复位控制寄存器(AIRCR)

寄存器	偏移量	R/W	描述	复位后的值
AIRCR	SCS_BA+0xD0C	R/W	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[15:8]							
23	22	21	20	19	18	17	16
VECTORKEY[7:0]							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留					SYSRESETREQ	VECTCLKACTIVE	保留

Bits	描述	
[31:16]	VECTORKEY	写该寄存器时，该域应该写0x05FA, 否则写动作将产生不可预测的结果.
[15:3]	保留	保留
[2]	SYSRESETREQ	该位写1，产生系统复位信号给芯片表示有复位请求. 该位只写，在复位时自动清零.
[1]	VECTCLRACTIVE	该位置1，清除所有固定的和可配置异常的活动状态. 该位只写，只有在内核挂起时可写. 注: 调试器负责重新初始化堆栈.
[0]	保留	保留

系统控制寄存器(SCR)

寄存器	偏移量	R/W	描述	复位后的值
SCR	SCS_BA+0xD10	R/W	系统控制寄存器	0x0000_0000

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

Bits	描述	
[31:5]	保留	保留
[4]	SEVONPEND	未处理状态时的发送事件 0 = 只有使能的中断或者事件可以唤醒处理器，不包括禁用中断在内 1 = 使能的事件和所有中断，包括禁用中断，都可以唤醒处理器 当一个事件或中断进入未处理状态时，事件信号将处理器从 WFE 唤醒. 如果处理器没有在等待事件，那么这个事件会被注册并影响下一个 WFE。 处理器也会在执行一个 SEV 指令或者一个外部事件时被唤醒
[3]	保留	保留
[2]	SLEEPDEEP	控制处理器使用休眠或者深度休眠作为它的低功耗模式 0 = 休眠 1 = 深度休眠
[1]	SLEEPONEXIT	表示当从 Handler 模式切换到 Thread 模式时，是否使用 sleep-on-exit 0 = 当切换到 Thread 模式时不休眠 1 = 当从 某个ISR 切换到 Thread 模式时，进入休眠或者深度休眠 设置该位为1 使能一个中断驱动的应用程序，避免返回到一个空的主函数应用
[0]	保留	保留

系统处理函数优先级寄存器2 (SHPR2)

寄存器	偏移量	R/W	描述	复位后的值
SHPR2	SCS_BA+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		保留					
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留							

Bits	描述	
[31:30]	PRI_11	系统处理器函数11 – SVCall的优先级 “0” 表示最高优先级& “3” 表示最低优先级
[29:0]	保留	保留

系统处理函数优先级寄存器3 (SHPR3)

寄存器	偏移量	R/W	描述	复位后的值
SHPR3	SCS_BA+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		保留					
23	22	21	20	19	18	17	16
PRI_14		保留					
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留							

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

7 电气特性

7.1 绝对最大额定值

符号	参数	最小值	最大值	单位
直流电源电压	VDD–VSS	-0.3	+7.0	V
输入电压	VIN	VSS-0.3	VDD+0.3	V
晶振频率	$1/t_{CLCL}$	4	24	MHz
工作温度	TA	-40	+85	°C
贮存温度	TST	-55	+150	°C
V_{DD}最大流入电流		-	120	mA
V_{SS}最大流出电流			120	mA
单一管脚最大灌电流			35	mA
单一管脚最大源电流			35	mA
所有管脚最大灌电流总合			100	mA
所有管脚最大源电流总合			100	mA

注: 超过上表所列的绝对最大额定值可能影响器件的生命周期和稳定性.

7.2 DC电气特性

7.2.1 NuMicro™ NUC123 DC 电气特性

(在无特别说明的情况下VDD-VSS=3.3V, TA = 25°C, FOSC = 72 MHz)

参数	符号.	明细表				测试条件
		最小值	典型值	最大值	单位	
工作电压	V_{DD}	2.5		5.5	V	V_{DD} =2.5V ~ 5.5V up to 72MHz
VDD 电压上升速率，确保内部操作正常	V_{RISE}	0.05			V/ms	
电源地	V_{SS} AV_{SS}	-0.3			V	
LDO输出电压	V_{LDO}	1.62	1.8	1.98	V	V_{DD} > 2.5V
模拟工作电压	AV_{DD}	0		V_{DD}	V	
模拟参考电压	Vref	0		AV_{DD}	V	
普通模式下的工作电流(72 MHz)	I_{DD1}		36		mA	V_{DD} = 5.5V@72 MHz, All IP and PLL Enabled, XTAL = 12 MHz
	I_{DD2}		21		mA	V_{DD} = 5.5V@72 MHz, All IP Disabled and PLL Enabled, XTAL = 12 MHz
	I_{DD3}		35		mA	V_{DD} = 3V@72 MHz, All IP and PLL enabled, XTAL = 12 MHz
	I_{DD4}		20		mA	V_{DD} = 3V@72 MHz, All IP Disabled and PLL Enabled, XTAL = 12 MHz
普通模式下的工作电流(12 MHz)	I_{DD5}		7		mA	V_{DD} = 5.5V@12 MHz, All IP Enabled and PLL Disabled, XTAL = 12 MHz
	I_{DD6}		4		mA	V_{DD} = 5.5V@12 MHz, All IP and PLL Disabled, XTAL = 12 MHz

参数	符号.	明细表				测试条件
		最小值	典型值	最大值	单位	
	I_{DD7}		6		mA	V_{DD} = 3V@12 MHz, All IP Enabled and PLL Disabled, XTAL = 12 MHz
	I_{DD8}		3		mA	V_{DD} = 3V@12 MHz, All IP and PLL Disabled, XTAL = 12 MHz
普通模式下的工作电流(4 MHz)	I_{DD9}		4		mA	V_{DD} = 5V@4 MHz, All IP Enabled and PLL Disabled, XTAL = 4 MHz
	I_{DD10}		3		mA	V_{DD} = 5V@4 MHz, All IP and PLL Disabled, XTAL = 4 MHz
	I_{DD11}		4		mA	V_{DD} = 3V@4 MHz, All IP Enabled and PLL Disabled, XTAL = 4 MHz
	I_{DD12}		2		mA	V_{DD} = 3V@4 MHz, All IP and PLL Disabled, XTAL = 4 MHz
空闲模式下的工作电流 (72 MHz)	I_{IDLE1}		29		mA	V_{DD}= 5.5V@72 MHz, All IP and PLL Enabled, XTAL = 12 MHz
	I_{IDLE2}		14		mA	V_{DD}=5.5V@72 MHz, All IP Disabled and PLL Enabled, XTAL = 12 MHz
	I_{IDLE3}		28		mA	V_{DD} = 3V@72 MHz, All IP and PLL Enabled, XTAL = 12 MHz
	I_{IDLE4}		13		mA	V_{DD} = 3V@72 MHz, All IP Disabled and PLL Enabled, XTAL=12 MHz
空闲模式下的工作电流 (12 MHz)	I_{IDLE5}		6		mA	V_{DD} = 5.5V@12 MHz, All IP Enabled and PLL Disabled, XTAL = 12 MHz
	I_{IDLE6}		3		mA	V_{DD} = 5.5V@12 MHz, All IP and PLL Disabled, XTAL = 12 MHz

参数	符号.	明细表				测试条件
		最小值	典型值	最大值	单位	
	I_{IDLE7}		5		mA	V_{DD} = 3V@12 MHz, All IP Enabled and PLL Disabled, XTAL = 12 MHz
	I_{IDLE8}		2		mA	V_{DD} = 3V@12 MHz, All IP and PLL Disabled, XTAL = 12 MHz
空闲模式下的工作电流 (4 MHz)	I_{IDLE9}		3		mA	V_{DD} = 5V@4 MHz, All IP Enabled and PLL Disabled, XTAL = 4 MHz
	I_{IDLE10}		2		mA	V_{DD} = 5V@4 MHz, All IP and PLL Disabled, XTAL = 4 MHz
	I_{IDLE11}		2		mA	V_{DD} = 3V@4 MHz, All IP Enabled and PLL Disabled, XTAL = 4 MHz
	I_{IDLE12}		1		mA	V_{DD} = 3V@4 MHz, All IP and PLL Disabled, XTAL = 4 MHz
空闲模式下的工作电流 (10 kHz)	I_{IDLE5}		131		uA	V_{DD} = 5.5V at 10 kHz, All IP Enabled and PLL Disabled, LIRC 10 kHz Enabled
	I_{IDLE6}		129		uA	V_{DD} = 5.5V at 10 kHz, All IP and PLL Disabled, LIRC 10 kHz Enabled
	I_{IDLE7}		125		uA	V_{DD} = 3V at 10 kHz, All IP Enabled and PLL Disabled, LIRC 10 kHz Enabled
	I_{IDLE8}		124		uA	V_{DD} = 3 V at 10 kHz, All IP and PLL Disabled, LIRC 10 kHz Enabled
掉电模式下工作电流 (掉电模式)	I_{PWD1}		12		μA	V_{DD} = 5.5V, No load when BOV function Disabled
	I_{PWD2}		9		μA	V_{DD} = 3.3V, No load when BOV function Disabled
PA, PB, PC, PD, PE, PF输入电流 (准双向模式)	I_{IN1}		-64		μA	V_{DD} = 5.5V, V_{IN} = 0V or V_{IN}=V_{DD}
RESET[1]脚输入电流	I_{IN2}	-55	-45	-30	μA	V_{DD} = 3.3V, V_{IN} = 0.45V

参数	符号.	明细表				测试条件
		最小值	典型值	最大值	单位	
PA, PB, PC, PD, PE, PF输入漏电流	I_{LK}	-2	-	+2	μA	V_{DD} = 5.5V, 0<V_{IN}<V_{DD}
PA~PF逻辑1至0转换电流(准双向模式)	I_{TL} [3]	-650	-	-200	μA	V_{DD} = 5.5V, V_{IN}<2.0V
PA, PB, PC, PD, PE, PF输入低电压 (TTL 输入)	V_{IL1}	-0.3	-	0.8	V	V_{DD} = 4.5V
		-0.3	-	0.6		V_{DD} = 2.5V
PA, PB, PC, PD, PE, PF输入高电压(TTL 输入)	V_{IH1}	2.0	-	V_{DD} +0.2	V	V_{DD} = 5.5V
		1.5	-	V_{DD} +0.2		V_{DD} =3.0V
PA, PB, PC, PD, PE, PF 输入低电压(Schmitt 输入)	V_{IL2}	-0.5		0.35 V_{DD}	V	
PA, PB, PC, PD, PE, PF 输入高电压(Schmitt 输入)	V_{IH2}	0.65 V_{DD}		V_{DD}+0.5	V	
PA~PE迟滞电压 (Schmitt 输入)	V_{HY}		0.2V_{DD}		V	
XT1[*2] 脚 输入低电平	V_{IL3}	0	-	0.8	V	V_{DD} = 4.5V
		0	-	0.4		V_{DD} = 3.0V
XT1[*2] 脚 输入高电平	V_{IH3}	3.5	-	V_{DD} +0.2	V	V_{DD} = 5.5V
		2.4	-	V_{DD} +0.2		V_{DD} = 3.0V
RESET脚 负向门槛电压(Schmitt输入)	V_{ILS}	-0.5	-	0.2V_{DD}	V	
RESET脚 正向门槛电压(Schmitt输入)	V_{IHS}	0.6V_{DD}	-	V_{DD}+0.5	V	
PA, PB, PC, PD, PE, PF源电流(准双向模式)	I_{SR11}	-300	-370	-450	μA	V_{DD} = 4.5V, V_S = 2.4V
	I_{SR12}	-50	-70	-90	μA	V_{DD} = 2.7V, V_S = 2.2V
	I_{SR12}	-40	-60	-80	μA	V_{DD} = 2.5V, V_S = 2.0V
PA, PB, PC, PD, PE, PF源电流(推挽模式)	I_{SR21}	-20	-24	-28	mA	V_{DD} = 4.5V, V_S = 2.4V
	I_{SR22}	-4	-6	-8	mA	V_{DD} = 2.7V, V_S = 2.2V
	I_{SR22}	-3	-5	-7	mA	V_{DD} = 2.5V, V_S = 2.0V
PA, PB, PC, PD, PE, PF灌电流(准双向及推挽模式)	I_{SK1}	10	16	20	mA	V_{DD} = 4.5V, V_S = 0.45V
	I_{SK1}	7	10	13	mA	V_{DD} = 2.7V, V_S = 0.45V
	I_{SK1}	6	9	12	mA	V_{DD} = 2.5V, V_S = 0.45V

参数	符号.	明细表				测试条件
		最小值	典型值	最大值	单位	
BOV_VL [1:0] =00b 时欠压电压	$V_{BO2.2}$	2.1	2.2	2.3	V	
BOV_VL [1:0] =01b 时欠压电压	$V_{BO2.7}$	2.6	2.7	2.8	V	
BOV_VL [1:0] =10b 时欠压电压	$V_{BO3.8}$	3.7	3.8	3.9	V	
BOV_VL [1:0] =11b 时欠压电压	$V_{BO4.5}$	4.4	4.5	4.6	V	
BOD电压迟滞范围	V_{BH}	30	-	150	mV	V_{DD} = 2.5V~5.5V

注:

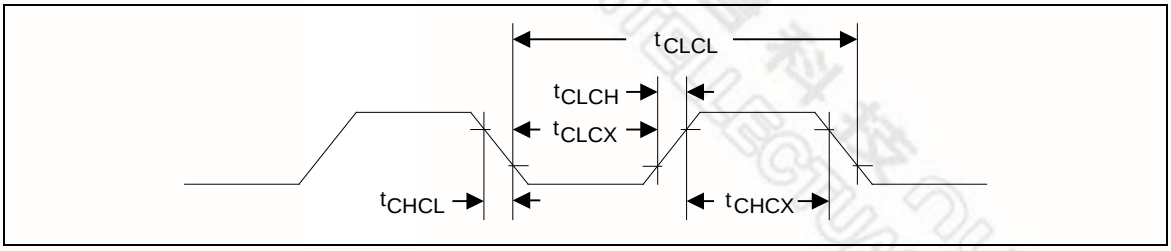
1. /RESET管脚为 Schmitt 触发 输入模式.

2. 晶振输入为CMOS 输入.

3. 当PA, PB, PC, PD 及 PE 管脚被外部由1驱动到0时， 可作为输出电流的源端，在VDD=5.5V时, 输出电流达到最大值， Vin 接近2V.

7.3 AC 电气特性

7.3.1 外部 4~24 MHz 高速振荡器 (Oscillator)



注: 占空比为50%.

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

7.3.2 外部 4~24 MHz 高速晶振 (Crystal)

参数	条件	最小值	典型值	最大值	单位
输入时钟频率	外部晶振	4	12	24	MHz
温度	-	-40	-	85	℃
VDD	-	2.5	5	5.5	V

7.3.2.1

典型晶振应用电路

晶振	C1	C2	R
4 MHz ~ 24 MHz	不需要	不需要	不需要

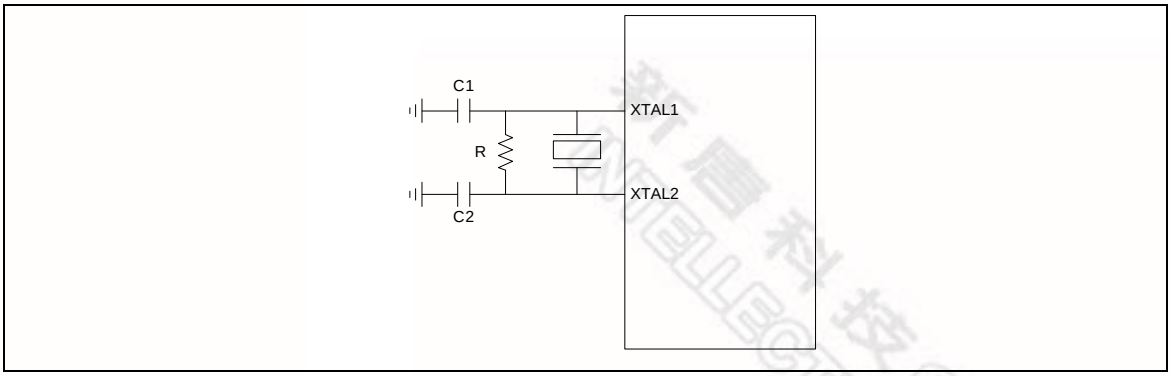


图 7-1 典型晶振应用电路

7.3.3 内部 22.1184 MHz 高速振荡器

参数	条件	最小值	典型值	最大值	单位
电压[1]	-	2.5	-	5.5	V
中心频率	-	-	22.1184	-	MHz
校验内部振荡器频率	+25□C; V_{DD} =5V	-1	-	+1	%
	-40□C~+85□C; VDD=2.5V~5.5V	-3	-	+3	%
工作电流	V_{DD} =5V	-	500	-	uA

7.3.4 内部 10 KHz 低速振荡器

参数	条件	最小值	典型值	最大值	单位
电压[1]	-	2.5	-	5.5	V
中心频率	-	-	10	-	kHz
校验内部振荡器频率	+25□C; V_{DD} =5V	-30	-	+30	%
	-40□C~+85□C; V_{DD}=2.5V~5.5V	-50	-	+50	%

注: 内部的工作电压来自LDO

7.4 模拟量特性

7.4.1 10 位 SARADC 特性

PARAMETER	SYM	SPECIFICATIONS				TEST CONDITIONS
		MIN	TYP	MAX	UNIT	
工作电压	AV_{DD}	2.7		5.5	V	$AV_{DD} = V_{DD}$
工作电流	I_{ADC}			1.5	mA	$AV_{DD} = V_{DD} = 5V$, $F_{SPS} = 150K$
分辨率	R_{ADC}			10	bit	
参考电压	V_{REF}		A_{VDD}		V	V_{REF} Connected to A_{VDD} in Chip
ADC 输入电压	V_{IN}	0		A_{VDD}	V	
采样率	F_{SPS}	150K			Hz	V_{DD} = 5V, ADC Clock = 6MHz Free Running Conversion
积分非线性误差(INL)	INL			±1	LSB	
差分非线性误差(DNL)	DNL			±1	LSB	
增益误差 (传输增益)	E_G			±2	LSB	
偏移误差	E_{OFFSET}		3		LSB	
绝对误差	E_{ABS}		4		LSB	
ADC 时钟频率	F_{ADC}	100K		6M	Hz	V_{DD} = 5V
时钟周期	AD_{CYC}	36			Cycle	
Bang-gap 电压	V_{BG}	1.27	1.35	1.44	V	-40°C ~ +85°C

7.4.2 LDO 规格 与 Power 管理

参数	最小值	典型值	最大值	单位	备注
输入电压	2.5	5	5.5	V	V_{DD}输入电压
输出电压	1.62	1.8	1.98	V	V_{DD} > 2.5V
温度	-40	25	85	℃	
Cbp	-	1	-	uF	Resr=1ohm

注:

1. 建议接一颗10uF或更大的电容和一颗100nF旁路电容在VDD与VSS之间.

2. 为保证电源稳定，要在LDO与VSS之间接一颗1uF或更大的电容.在LDO与最近的VSS之间有助于抑制输出噪声.

7.4.3 低压复位说明

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

7.4.4 欠压检测说明

参数	条件	最小值	典型值	最大值	单位
操作电压	-	2.5	-	5.5	V
静态电流	AVDD=5.5V	-	-	125	μA
温度	-	-40	25	85	℃
欠压电压	BOV_VL[1:0]=11	4.3	4.5	4.7	V
	BOV_VL [1:0]=10	3.6	3.8	4.0	V
	BOV_VL [1:0]=01	2.6	2.7	2.8	V
	BOV_VL [1:0]=00	2.1	2.2	2.3	V
迟滞	-	30	-	150	mV

7.4.5 上电复位说明 (5V)

参数	条件	最小值	典型值	最大值	单位
温度	-	-40	25	85	℃
复位电压	V+	-	2	-	V
静态电流	Vin>复位电压	-	1	-	nA

7.4.6 USB PHY 说明

USB DC 电气特性

7.4.6.1

符号	参数	条件	最小值	典型值	最大值	单位
V_{IH}	输入高 (driven)		2.0			V
V_{IL}	输入低				0.8	V
V_{DI}	差分输入	|PADP-PADM|	0.2			V
V_{CM}	差分同模范围	Includes V_{DI} range	0.8		2.5	V
V_{SE}	单端接收器极限		0.8		2.0	V
	接收器滞后			200		mV
V_{OL}	输出低 (driven)		0		0.3	V
V_{OH}	输出高 (driven)		2.8		3.6	V
V_{CRS}	输出信号串扰电压		1.3		2.0	V
R_{PU}	上拉电阻		1.425		1.575	kΩ
R_{PD}	下拉电阻		14.25		15.75	kΩ
V_{TRM}	上行端口上的上拉电阻的极限电压 (R_{PU})		3.0		3.6	V
Z_{DRV}	驱动输出阻抗*	稳态驱动*		10		Ω
C_{IN}	发射器容值	Pin to GND			20	pF

*驱动输出阻抗不包括串联电阻阻抗.

7.4.6.2

USB 全速驱动器电气特性

符号	参数	条件	最小值	典型值	最大值	单位
T_{FR}	上升时间	C_L=50p	4		20	ns
T_{FF}	下降时间	C_L=50p	4		20	ns
T_{FRFF}	上升与下降时间比值	$T_{FRFF}=T_{FR}/T_{FF}$	90		111.11	%

7.4.6.3

USB 功耗

符号	参数	条件	最小值	典型值	最大值	单位
I_{VDDREG} (全速)	VDDD 和 VDDREG 供给电流(稳态)	待机		50		uA
		输入模式				uA
		输出模式				uA

7.5 SPI 动态特性

符号	参数	最小值	典型值	最大值	单位
SPI 主机模式 (VDD = 4.5V ~ 5.5V, 30pF 负载电容)					
t_{DS}	数据准备时间	TBD	TBD	-	ns
t_{DH}	数据保持时间	TBD	-	-	ns
t_V	数据输出有效时间	-	TBD	TBD	ns
SPI 主机模式 (VDD = 3.0V ~ 3.6V, 30pF 负载电容)					
t_{DS}	数据准备时间	TBD	TBD	-	ns
t_{DH}	数据保持时间	TBD	-	-	ns
t_V	数据输出有效时间	-	TBD	TBD	ns
SPI 从机模式 (VDD = 4.5V ~ 5.5V, 30pF 负载电容)					
t_{DS}	数据准备时间	TBD	-	-	ns
t_{DH}	数据保持时间	TBD	-	-	ns
t_V	数据输出有效时间	-	TBD	TBD	ns
SPI 从机模式 (VDD = 3.0V ~ 3.6V, 30pF 负载电容)					
t_{DS}	数据准备时间	TBD	-	-	ns
t_{DH}	数据保持时间	TBD	-	-	ns
t_V	数据输出有效时间	-	TBD	TBD	ns

TBD:需再定义

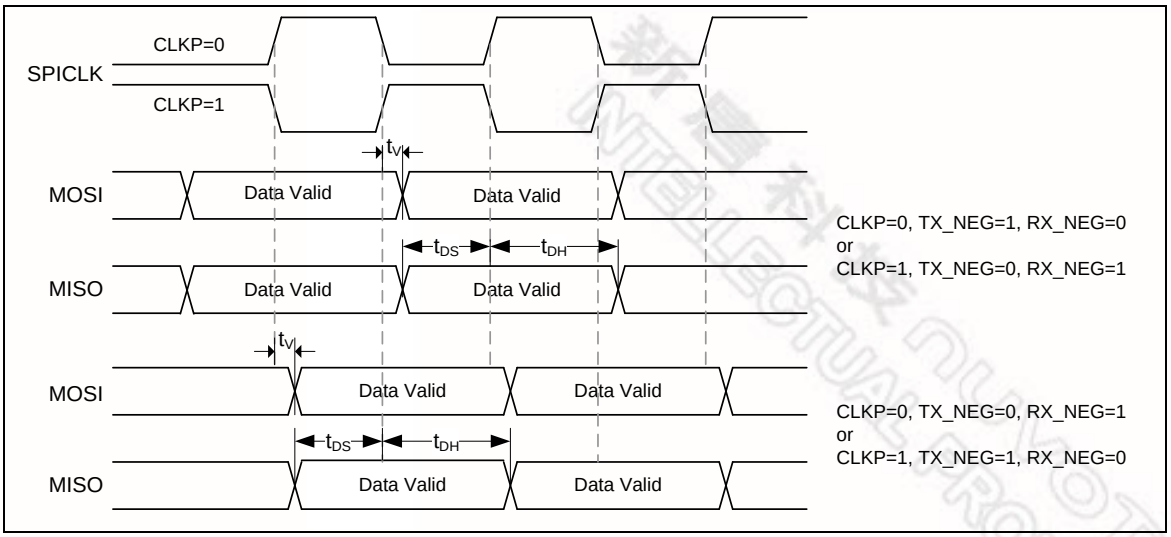


图 7-2 SPI 主机动态特性时序图

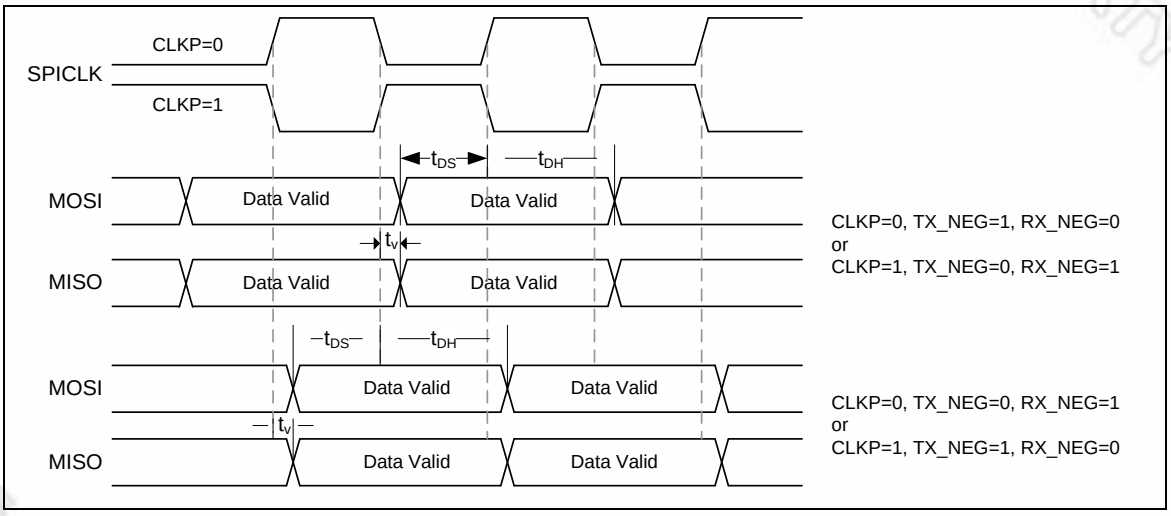


图 7-3 SPI 从机动态特性时序图

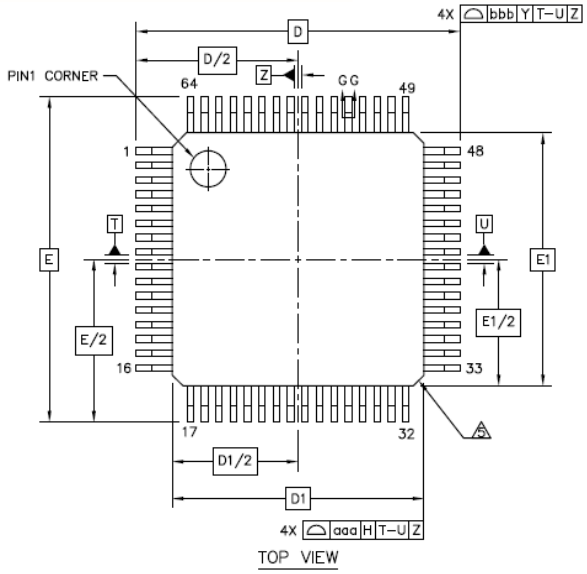
7.6 Flash DC 电气特性

符号	参数	条件	最小值	典型值	最大值	单位
T_{ret}	保存时间	Temp=85 ℃	10			year
T_{erase}	页擦除时间		19	20	21	ms
T_{mass}	Mass erase time		30	40	50	ms
T_{prog}	编程时间		38	40	42	us
V_{DD}	工作电压		1.62	1.8	1.98	V[1]
I_{dd1}	读电流				0.25	mA
I_{dd2}	编程/擦除电流				7	mA
I_{pd}	掉电模式电流			1	20	uA

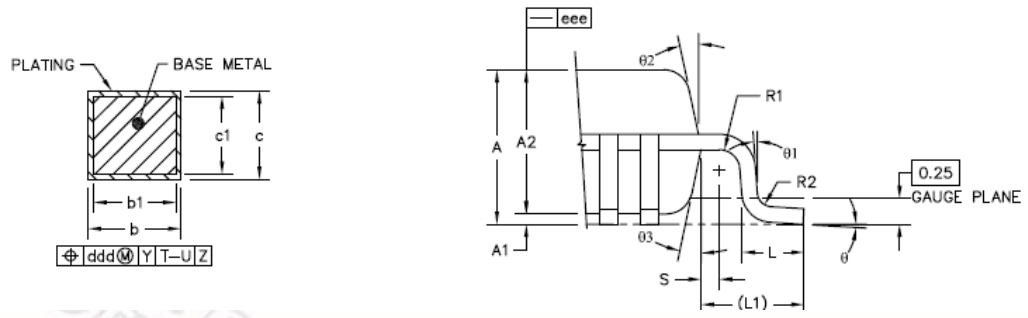
1. V_{DD} 是芯片 LDO 的输出电压源.

8 封装定义

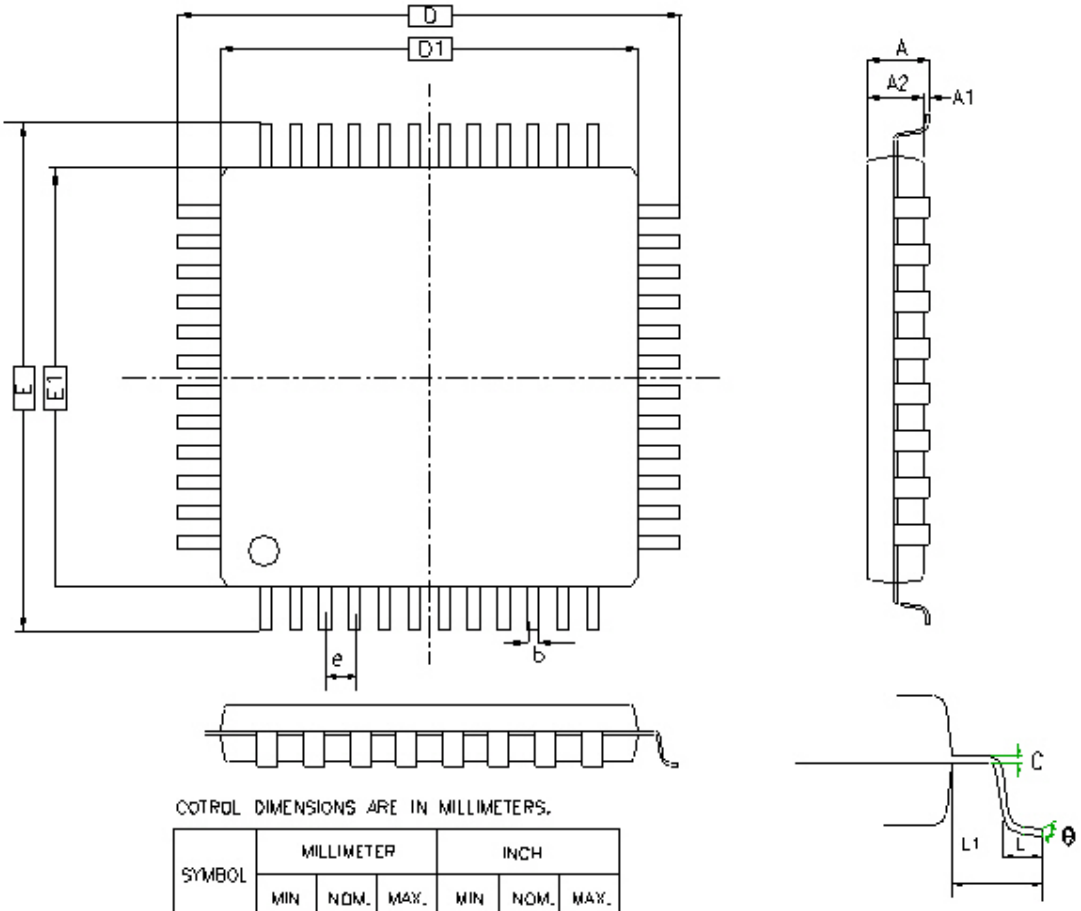
8.1 64L LQFP (7x7x1.4mm footprint 2.0 mm)



		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.13	0.18	0.23
LEAD WIDTH		b1	0.13	0.16	0.19
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D	9 BSC		
	Y	E	9 BSC		
	X	D1	7 BSC		
	Y	E1	7 BSC		
LEAD PITCH		e	0.4 BSC		
		L	0.45	0.6	0.75
FOOTPRINT		L1	1 REF		
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2	---	---
PACKAGE EDGE TOLERANCE		aaa	0.2		
LEAD EDGE TOLERANCE		bbb	0.2		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.07		
MOLD FLATNESS		eee	0.05		



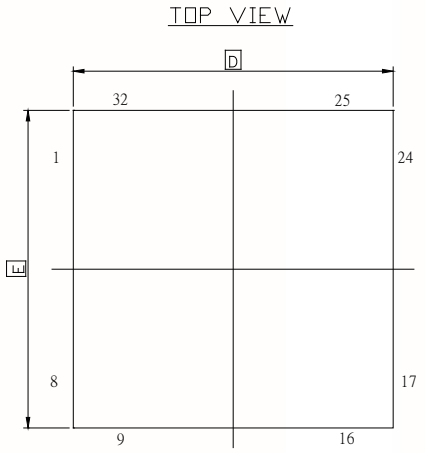
8.2 48L LQFP (7x7x1.4mm footprint 2.0mm)

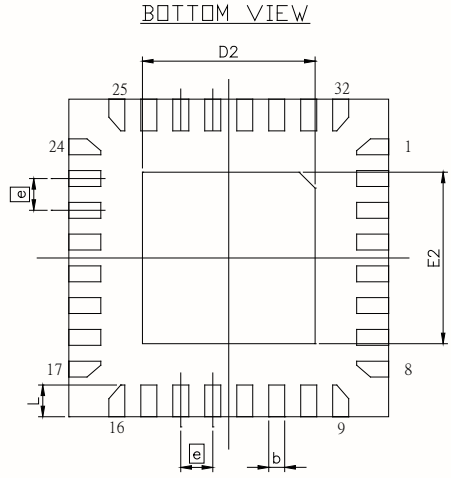


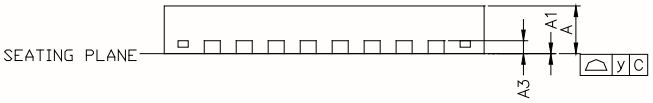
CONTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN	NOM.	MAX.	MIN	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A1	0.05	0.10	0.15	0.002	0.004	0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
D1	6.90	7.00	7.10	0.272	0.276	0.280
E1	6.90	7.00	7.10	0.272	0.276	0.280
e	0.35	0.50	0.65	0.014	0.020	0.280
D	8.9	9.00	9.10	0.350	0.354	0.358
E	8.9	9.00	9.10	0.350	0.354	0.358
L	0.45	0.60	0.75	0.018	0.024	0.030
L1	—	1.00	—	—	0.039	—
C	0.09	—	0.20	0.0035	—	0.0079
θ	0°	—	7°	0°	—	7°
b	0.17	0.22	0.27	0.007	0.0087	0.011

8.3 33L QFN (5x5x0.8 mm)







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		
D2	3.20	3.40	3.60	0.124	0.138	0.142
E2	3.20	3.40	3.60	0.124	0.138	0.142

9 版本历史

版本	日期	页码/章节	描述
V1.00	Mar 27, 2012	-	初次发行版本
V1.01	May 2, 2012	第7章	更新7.2节DC 电器特性
V1.02	May 7, 2012	第3章	更新图 3-1 NuMicro™ NUC123 系列编号规则
V1.03	May 14, 2012	第3章	修改 LQFP 64-pin 和 LQFP 48-pin 引脚图及引脚说明
V1.04	May 30, 2012	第3章 第5章	修改PC.12 引脚的引脚图及引脚说明. 修改 表 5-8 ISP 模式命令.
V1.05	July 20, 2012	第5章 第7章	修正GPC_MFP[12] 寄存器中的GPC_MFP12描述. 移除APBCLK[1] 寄存器中的RTC_EN描述, 修改PWRCON[6] 寄存器中的PD_WU_STS描述和表 5-5 掉电模式控制表中相关的RTC功能 移除7.3节 AC 电气特性中的32.768kHz振荡器特性, 修改PWRCON[7] 寄存器中的PWR_DOWN_EN描述和和表 5-5 掉电模式控制表中相关的32.768kHz振荡器功能. 将CLKSEL1[6:4] 寄存器中的SPI_S分为SPI2_S, SPI1_S和SPI_S. 修改5.1节中有关串行外围设备接口 (SPI)的描述 修改图 5-1 NuMicro™ NUC123 电源分配图和7.4.2和LDO规格 与 Power 管理中的电容值从10uF 改为1uF.

V1.06	Aug. 21, 2012	第2章 第4章 第5章 第7章	更新2.1节NuMicro™ NUC123 特性中SPI描述“主机速率高达32 Mbps，从机高达16Mbps” 更新2.1节NuMicro™ NUC123 特性中封装描述新增“LQFP 64-pin” 更新图 4-1 NuMicro™ NUC123 框图 修正PDID[31:0] 寄存器的描述和复位值 修正GFP_MFP[3:0] 寄存器的描述和复位值 修正FATCON[6] 寄存器的MFOM描述 修正DMA_PDSSR1[23:16]为保留位 修正5.17节ADC的单周期扫描模式和连续扫描模式功能描述 更新5.17节模拟数字转换 (ADC)特性的ADC时钟最大6 MHz和转换率到150K SPS 更新7.4节模拟量特性的10-bit SARADC 说明 新增5.5.6节IAP (In Application Programming) 新增两个批注在PWM Counter (CNR) 寄存器的描述

V1.07	March 29, 2013	第3章 第5章 第7章	移除3.2.1節NuMicro™ NUC123 管脚图中PD.0引腳的SPISS11功能 修正3.2.2管脚功能描述中QFN33封裝的AVDD引腳編號為25 修正LDO的輸出電壓為1.8V在7.2.1節NuMicro™ NUC123 DC 电气特性和7.4.2節LDO 规格 与 Power 管理 修正TRGS[1:0], TRGEN, CHEN[7:0] 和 PERIODSEL[3:0] 寄存器的描述 從使用者設定Config0移除CKF寄存器 移除DFP_CSR寄存器 更改5.8節，5.11節，5.12節和 5.13節中有關I2C, Timer, Watchdog Timer and Window Watchdog Timer 的全部描述 新增7.6節Flash DC 修改第5章寄存器地址空间分配表格式 在5.2.7節中断源控制寄存器分開描述IRQ0~IRQ31全部中断源识别寄存器 在5.7.5寄存器描述中分開描述全部與GPIO相關的寄存器 改善5.5.10節ISP过程的描述 新增ISP命令“Vector Page Re-Map” 在表 5-8 ISP 模式命令和新增ISPSTA 控制寄存器在5.5.12節Flash控制寄存器描述 從5.17.1節概述修正一些ADC描述到5.17.4節功能描述 修正ADDR0~7寄存器中的RSLT[9:0]和ADCMPR0~1寄存器中的CMPD[9:0]為10 bits以及图 5-104 使能通道上连续扫描时序图 從图 5-107 ADC 单端输入转换电压和转换结果映射图移除USB 在5.16.4.2節新增图 5-96 MSB 校正 (MSB Justified) 时序图 (Format = 1)和图 5-97 PCM 模式 A 时序图 (Format = 0)
V1.08	July 9, 2014	第5章	修正表5-10 看门狗超时间隔选择 修正5.17.2節ADC特性 修正ISPFF寄存器的描述 修正图 5‑17 ISP 操作流程 (继续) 的圖表編號 修正TMRx_CLK (x: 0~3) 和 TxEX pin (x: 0, 2 or 3)描述在5.11.4節Timer功能描述

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.