PT32Z192 系列

用户手册

目录

1 概述

PT32Z192 系列产品主要用于智能门锁的指纹识别。芯片基于 Cortex-M3 CPU 内核，512/256KB FLASH 指令存储器，128/64KB SRAM 数据存储器，具有算法加速运算，外加标准外设模块。

2 主要特性

本芯片基本特性如下：

2.1 内核系统

- ◆Cortex-M3 内核
- ◆最高工作频率 160Mhz
- ◆硬件单周期乘法器
- ◆支持 Thumb-2 指令集
- ◆三级流水线，指令和数据总线独立
- ◆NVIC 中断控制器
- ◆SysTick 定时器

2.2 存储系统

- ◆512KB 指令 FLASH
- ◆128KB 数据 SRAM
- ◆4KB Cache

2.3 时钟系统

- ◆片内 24Mhz+/-2%-RC 时钟，复位默认系统时钟
- ◆片内振荡电路驱动外部 24MHz 晶体产生稳定时钟
- ◆系统工作主时钟源可选项
  - ➢ 内部 24Mhz-RC 时钟
  - ➢ 外部 24Mhz 晶振时钟
  - ➢ PLL 最高 160Mhz 时钟
- ◆片内 32Khz+/-20%-RC 独立时钟，供 WDT

2.4 电源系统

- ◆工作电压 2.7~3.6V

2.5 芯片功耗

◆深度睡眠模式 5μA

2.6 GPIO

- 具有 51 个 GPIO(包括复用管脚)
- 单个引脚可编程内部上拉或下拉，输出开漏控制
- 16 引脚支持状态变化唤醒

2.7 标准外设

2.7.1 UART

- 支持 3 个 UART
- 可编程波特率最高至 115200 BPS，
- 支持 8 或 9 位数据通讯， 1 位奇偶校验， 1 位起始位， 0.5~2 位停止位.
- 支持 DMA 数据传输.

2.7.2 SPI

- 支持 2 个标准 SPI
- 支持 Mode0/1/2/3 传输协议，支持主/从模式
- 可编程时钟速率最高至 8MHz
- 可编程 4-16 位宽度数据传输
- 8 级 FIFO 并支持 DMA

2.7.3 QSPI

- 支持 1 个 QSPI
- 只支持主模式
- 支持并行模式
- 具有 16 级深度 FIFO

2.7.4 I2C

- 支持 1 个标准 I2C
- 主/从模式串行数据总线通讯接口
- 可编程时钟速率最高至 400KHz
- 8 位数据传输

2.7.5 USB1.1

- 支持 1 个 USB1.1

- 支持高速/全速模式

2.8 TIMER

- 3 个 16 位通用定时器 TMR
- 1 个 32 位通用定时器 TMR
- 16 位通用定时器 TMR，带 3 路输入捕捉
- 16 位通用定时器 TMR，带 PWM 输出
- 1 个 WTD

2.9 ADC

- 支持 1 个 12-bit SAR ADC
- 支持 6 路片外模拟量输入
- 最高采样率 1Mhz/SPS
- 支持 Timer 触发
- 支持 DMA

2.10 定制外设

2.10.1 硬件加解密引擎 AES

- 128/192/256b 密钥
- 支持 ECB/CBC 模式

2.10.2 计算算法(DSP)

- 开方
- 坐标旋转
- 三角函数 tan

2.10.3 卷积算法 CNN

- 最大支持 13X13 卷积矩阵计算

2.11 温度

◆工作环境：-40℃~+85℃

◆存储环境：-55℃~+125℃

2.12 封装

◆LQFP64(7*7)

3 引脚

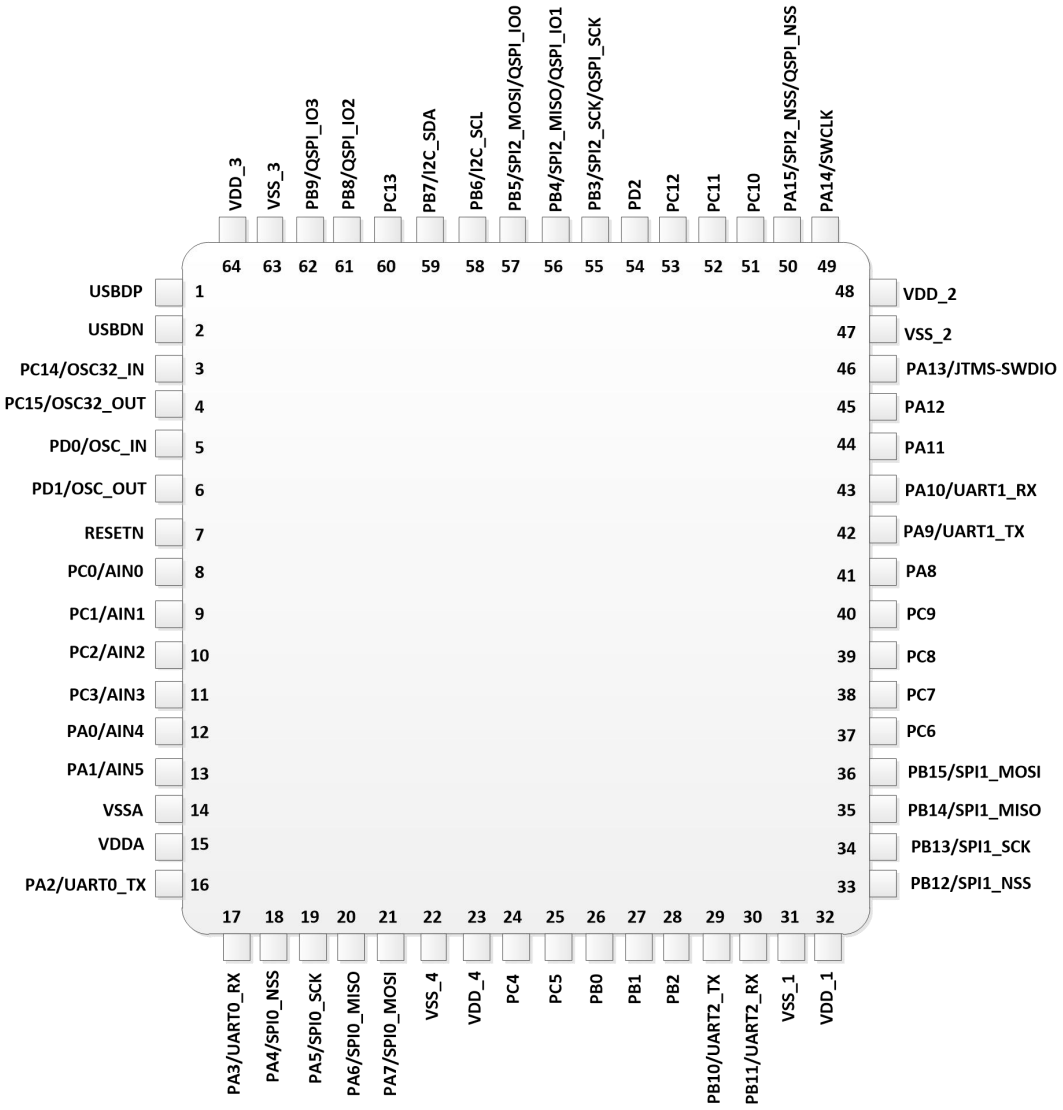


图 3-1：LQFP64 封装管脚分布(TopView)

4 管脚定义

64 脚	引脚名	数字功能				模拟功能
		主功能	替换功能 0	替换功能 1	替换功能 2	
1	USBDP	USBDP	-	-	-	-
2	USBDN	USBDN	-	-	-	-
3	PC14/OSC32_IN	PC14	OSC32_IN	EXT_32K_IN	-	-
4	PC15/OSC32_OUT	PC15	OSC32_OUT	-	-	-
5	PD0/OSC_IN	PD0	OSC_IN	EXT_24M_IN	-	-
6	PD1/OSC_OUT	PD1	OSC_OUT	-	-	-
7	RESETN	RESETN	-	-	-	-
8	PC0/AIN0	PC0	-	-	-	AIN0
9	PC1/AIN1	PC1	-	-	-	AIN1
10	PC2/AIN2	PC2	-	-	-	AIN2
11	PC3/AIN3	PC3	-	-	-	AIN3
12	PA0/AIN4	PA0	-	-	-	AIN4
13	PA1/AIN5	PA1	-	-	-	AIN5
14	VSSA	VSSA	-	-	-	-
15	VDDA	VDDA	-	-	-	-
16	PA2/UART0_TX	PA2	UART0_TX	-	-	-
17	PA3/UART0_RX	PA3	UART0_RX	-	-	-
18	PA4/SPI0_NSS	PA4	SPI0_NSS	SPI0_NSS	-	-
19	PA5/SPI0_SCK	PA5	SPI0_SCK	SPI0_SCK	-	-
20	PA6/SPI0_MISO	PA6	SPI0_MISO	SPI0_SOMI	-	-
21	PA7/SPI0_MOSI	PA7	SPI0_MOSI	SPI0_SIMO	-	-
22	VSS_4	VSS_4	-	-	-	-
23	VDD_4	VDD_4	-	-	-	-
24	PC4	PC4	-	-	-	-
25	PC5	PC5	-	-	-	-
26	PB0	PB0	-	-	-	-
27	PB1	PB1	-	-	-	-
28	PB2	PB2	-	-	-	-
29	PB10/UART2_TX	PB10	UART2_TX	-	-	-
30	PB11/UART2_RX	PB11	UART2_RX	-	-	-
31	VSS_1	VSS_1	-	-	-	-
32	VDD_1	VDD_1	-	-	-	-
33	PB12/SPI1_NSS	PB12	SPI1_NSS	SPI1_NSS	-	-
34	PB13/SPI1_SCK	PB13	SPI1_SCK	SPI1_SCK	-	-
35	PB14/SPI1_MISO	PB14	SPI1_MISO	SPI1_SOMI	-	-
36	PB15/SPI1_MOSI	PB15	SPI1_MOSI	SPI1_SIMO	-	-

37	PC6	PC6	-	PWMn1	-	-
38	PC7	PC7	-	PWMn2	-	-
39	PC8	PC8	-	PWMn3	-	-
40	PC9	PC9	-	PWMn4	-	-
41	PA8	PA8	-	PWM1	-	-
42	PA9/UART1_TX	PA9	UART1_TX	PWM2	-	-
43	PA10/UART1_RX	PA10	UART1_RX	PWM3	-	-
44	PA11	PA11	-	PWM4	-	-
45	PA12	PA12	-	BKI	-	-
46	PA13/JTMS-SWDIO	PA13	JTMS-SWDIO	-	-	-
47	VSS_2	VSS_2	-	-	-	-
48	VDD_2	VDD_2	-	-	-	-
49	PA14/SWCLK	PA14	SWCLK	-	-	-
50	PA15/SPI2_NSS/ QSPI_NSS	PA15	QSPI_NSS	-	-	-
51	PC10	PC10	-	-	-	-
52	PC11	PC11	-	-	-	-
53	PC12	PC12	-	-	-	-
54	PD2	PD2	-	-	-	-
55	PB3/SPI2_SCK/ QSPI_SCK	PB3	QSPI_SCK	-	-	-
56	PB4/SPI2_MISO/ QSPI_IO1	PB4	QSPI_IO1	-	-	-
57	PB5/SPI2_MOSI/ QSPI_IO0	PB5	QSPI_IO0	-	-	-
58	PB6/I2C_SCL	PB6	I2C_SCL	-	-	-
59	PB7/I2C_SDA	PB7	I2C_SDA	-	-	-
60	PC13	PC13	-	-	-	-
61	PB8/QSPI_IO2	PB8	QSPI_IO2	-	-	-
62	PB9/QSPI_IO3	PB9	QSPI_IO3	-	-	-
63	VSS_3	VSS_3	-	-	-	-
64	VDD_3	VDD_3	-	-	-	-

5 系统架构

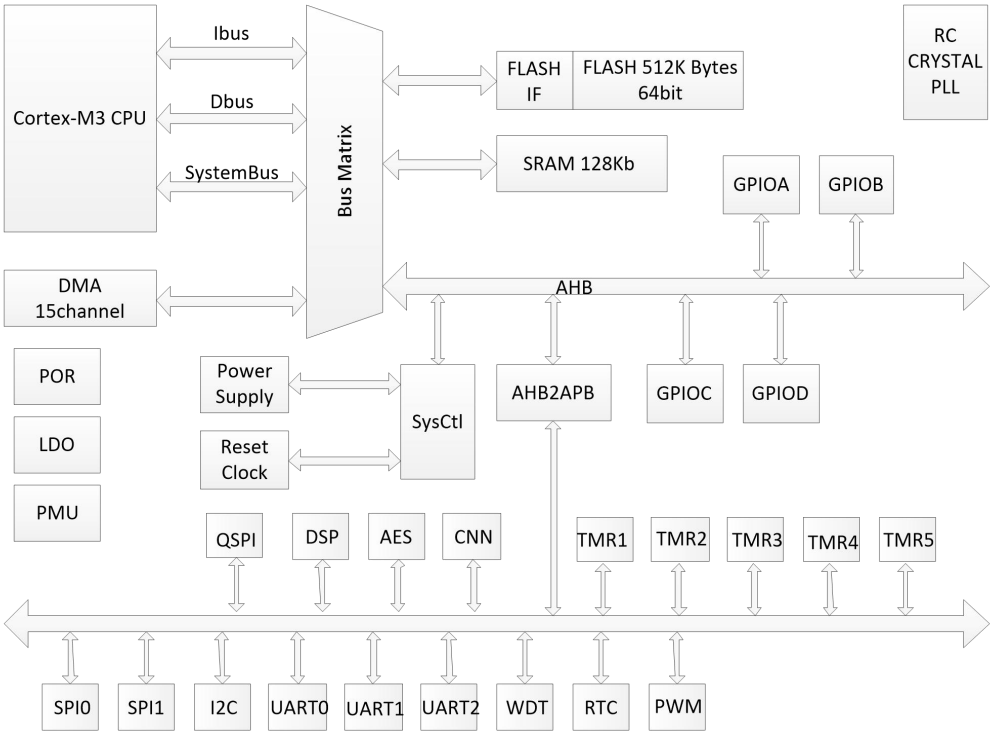


图 5-1：系统结构框图

5.1 地址分配

5.1.1 AHB 地址分配

AHB 访问地址分配见下表：

AHB Memory Address	Size	Peripherals	Description
0x0000 0000 – 0x0007 FFFF	512KB	指令 FLASH 主代码空间	
0x0008 0000 – 0x001F FFFF		Reserved	
0x2000 0000 – 0x2001 FFFF	128KB	数据 SRAM 数据区	
0x4000 0000 – 0x4000 FFFF	64KB	外设功能模块寄存器区 (APB)	详见 6.2
0x4001 0000 – 0x4001 0FFF	4KB	GPIOA	
0x4001 1000 – 0x4001 1FFF	4KB	GPIOB	
0x4001 2000 – 0x4001 2FFF	4KB	GPIOC	
0x4001 2000 – 0x4001 2FFF	1KB	QSPI	
0x4001 3000 – 0x4001 5FFF	4KB	GPIOD	
0x4001 4000 – 0x4001 FFFF		Reserved	
0x4002 1000 – 0x4002 13FF	1KB	SYSTEM_CTL 系统控制	

5.1.2 APB 地址分配

APB 访问地址分配见下表：

APB Memory Address	Size	Peripherals	Description
0x4000 0000 – 0x4000 03FF	1KB	TIMER2	
0x4000 0400 – 0x4000 07FF	1KB	TIMER3	
0x4000 0800 – 0x4000 0BFF	1KB	TIMER4	
0x4000 0C00 – 0x4000 0FFF	1KB	TIMER5	
0x4000 1000 – 0x4000 13FF	1KB	PWM	
0x4000 1400 – 0x4000 17FF	1KB	TIMER1	
0x4000 1800 – 0x4000 27FF	4KB	DSP	
0x4000 2800 – 0x4000 2BFF	1KB	RTC	
0x4000 2C00 – 0x4000 2FFF	1KB	WDG	
0x4000 3000 – 0x4000 37FF	1KB	保留	
0x4000 3800 – 0x4000 3BFF	1KB	SPI0	
0x4000 3C00 – 0x4000 3FFF	1KB	SPI1	
0x4000 4000 – 0x4000 43FF	1KB	保留	
0x4000 4400 – 0x4000 47FF	1KB	UART0	
0x4000 4800 – 0x4000 4BFF	1KB	UART1	
0x4000 4C00 – 0x4000 4FFF	1KB	UART2	
0x4000 5000 – 0x4000 53FF	1KB	保留	
0x4000 5400 – 0x4000 57FF	1KB	I2C	
0x4000 5800 – 0x4000 5BFF	1KB	ANA_MIX	
0x4000 5C00 – 0x4000 5FFF	1KB	保留	
0x4000 6000 – 0x4000 63FF	1KB	ADC	
0x4000 6400 – 0x4000 E7FF	8KB	保留	
0x4000 E800 – 0x4000 EBFF	1KB	AES	
0x4000 F000 – 0x4000 F3FF	1KB	DMA	
0x4000 F400 – 0x4000 F7FF	1KB	保留	
0x4000 F800 – 0x4000 FBFF	1KB	FLASH Controller	

6 复位

芯片有如下几个复位源：

1. 外部引脚复位
2. 上电复位（POR）
3. 低电压掉电复位
4. FLASH 低电压检测复位
5. 看门狗复位
6. 软件复位

6.1 复位系统框图

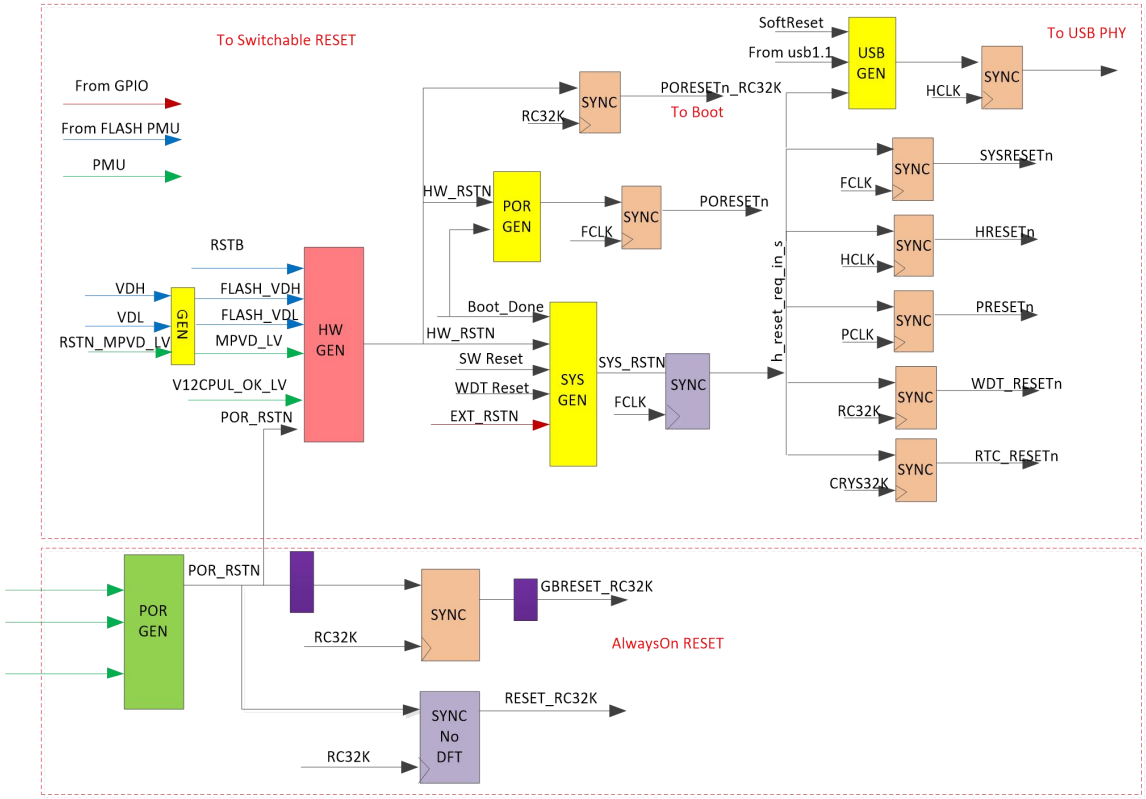


图 7-1：复位系统

6.2 外部引脚复位

在芯片工作时，当外部复位引脚 RSTN 上输入一个有效低电平，并且持续最少 4 个系统时钟周期后，将发生外部复位，芯片进入系统复位状态。外部复位的引脚是一个带有施密特触发的输入引脚。

6.3 POR 复位

当芯片上电之后，芯片处于上电复位状态，芯片的电压会逐渐上升到芯片可以工作的电压值。当这个工作电压持续一段时间后，上电复位被释放，芯片进入上电后的初始化状态。芯片会对各个数字、模拟模块进行相应的初始化工作，然后 CPU 开始工作，运行程序。

6.4 低电压掉电复位

两个电压监测模块，一个在低电压域，一个在高电压域，分别监测和保护两个电压域。如果两个低电压监测模块使能工作，且选择低电压复位功能，只要芯片高电压域电压低于高电压域 LVD 的阈值，或者低电压域电压低于低电压阈值，芯片进入系统复位状态。

6.5 FLASH 电压检测复位

当 FLASH PMU 供电 VDD_2 大于 1.33V 且小于 1.54V，VDL 输出高，系统处于复位状态，或者 VDD_2 大于 6.0V，VDH 输出高，系统处于复位状态。

6.6 看门狗复位

当看门狗被使能工作，且看门狗计数器发生溢出时，发生看门狗复位，芯片进入系统复位状态

6.7 软件复位

执行 Cortex-M3 复位指令，芯片进入系统复位状态

7 时钟

芯片具有以下几组时钟源

1. 内部高速 RC 24M
2. 外部晶振 24M 输入
3. PLL 高速时钟
4. 外部输入时钟
5. 外部晶振 32K 输入
6. 内部 RC 32K

7.1 时钟结构

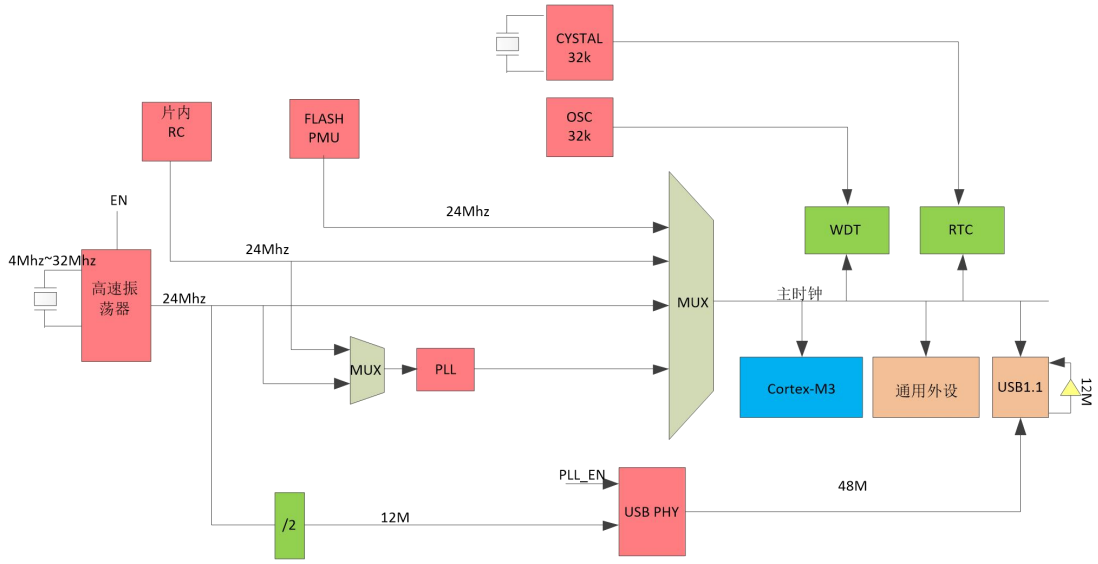


图 9-1：时钟结构图

7.2 晶振 24M 时钟

晶振 24M 时钟由片外 24Mh 晶体和片内负载电容构成的振荡电路产生。片内 PLL 电路可将 24M 倍频至 160M,供内核使用。

7.3 晶振 32K 时钟

晶振 32K 时钟由片外 32KHz 晶体和片内负载电容构成的振荡电路产生。独立供给 RTC 使用。

7.4 片内 24M 时钟

片内 24Mhz 为 RC 性质高可靠性时钟。

- 任何形式的复位后，芯片系统以此 24Mhz 工作，系统进行初始化
- 系统启动外部 24Mh 晶振电路，系统时钟可切换至晶振时钟
- 24MHz 时钟频率经修调后，全电压、全温度范围精度：<±2%
- 时钟频率可软件再修调，范围±20%
- 可作为 PLL 输入时钟

7.5 片内 32K 时钟

片内 32Khz 为 WDT 专用时钟

- 只要上电，永远工作
- 低功耗设计
- 全电压范围，全温度范围精度：±20%

7.6 PLL 时钟

片内 PLL 可产生 160Mhz 高速时钟

- 输入时钟可选片外 24Mhz 晶振时钟和片内 24M RC 时钟
- 可编程 PLL 倍频系数
- 可编程预分频
- 最高输出 160Mhz

8 电源管理

8.1 供电示意图

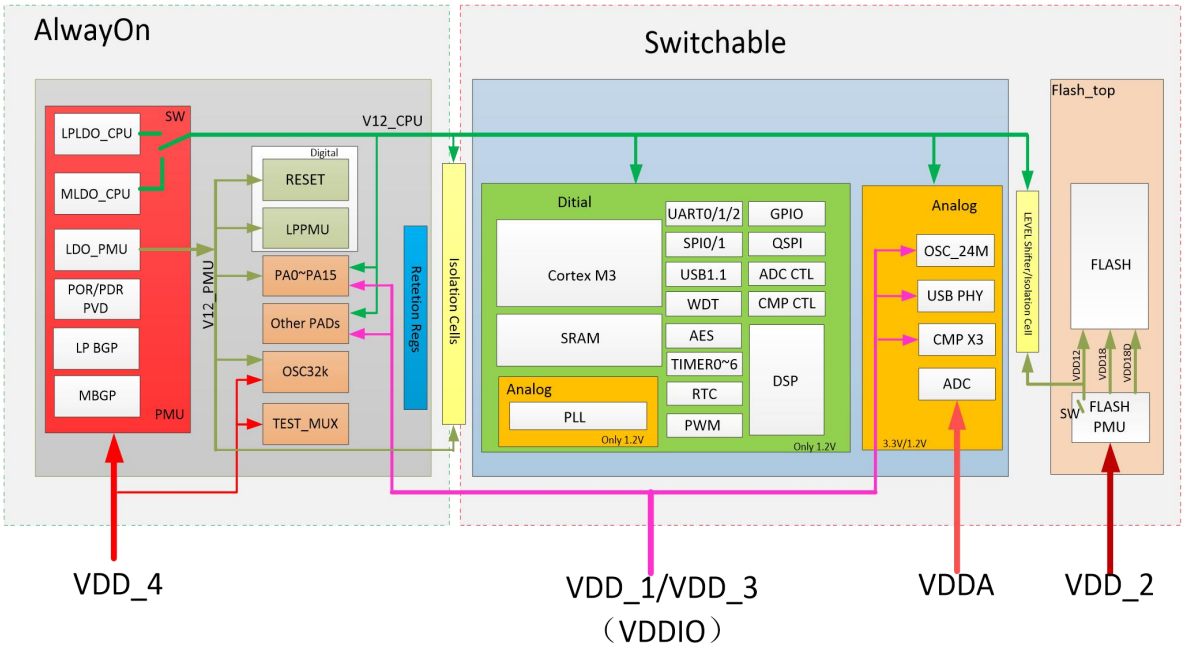


图 9-1: 芯片供电示意图

8.2 芯片上电时序图

9 工作模式

芯片有如下 4 种工作模式，正常工作模式，休眠模式，深度休眠模式和超低功耗模式。其中休眠模式，深度休眠模式为低功耗模式。

1. 工作模式
2. 睡眠模式
3. 深度睡眠模式
4. 超低功耗模式

使用 ARM Cortex-M3 的 WFI 或 WFE 两条指令可以使芯片进入休眠模式或深度睡眠模式。当执行 WFI 或 WFE 指令后，芯片进入哪种低功耗模式，由 Cortex-M3 系统控制寄存器（System Control Register）的 SLEEPDEEP 位决定。请参考 Cortex-M3 系统控制寄存器的说明。

软件配置 PMU 控制寄存器 PMU_CTRL，对 LPPMUENABLE 置 1，使芯片进入超低功耗模式

9.1 模式唤醒

芯片进入休眠模式后，可以通过以下方式唤醒：

1. 外部复位
2. 调试模式请求
3. 所有使能的中断源

芯片进入深度睡眠模式后，可以通过以下方式唤醒：

1. 外部复位
2. 外部中断（GPIO 的电平中断）
3. WDG 中断
4. 调试模式请求

芯片进入超低功耗模式后，可以通过以下方式唤醒：

1. 只有 GPIOA 唤醒

10矢量中断控制

编号	优先级	名称	说明	地址
-16	-	-	保留	0x0000_0000
-15	-3	Reset	复位	0x0000_0004
-14	-2	NMIN	看门狗定时器中断	0x0000_0008
-13	-1	HardFault	硬件失效	0x0000_000C
-12	0	MemManage	存储器管理	0x0000_0010
-11	1	BusFault	总线错误，预取指失败，存储器访问失败	0x0000_0014
-10	2	UsageFault	未定义的指令或非法状态	0x0000_0018
-9	-	-	保留	0x0000_001C
-8	-	-	保留	0x0000_0020
-7	-	-	保留	0x0000_0024
-5	-	-	保留	0x0000_0028

-5	3	SVCall	通过 SWI 指令的系统服务调用	0x0000_002C
-4	4	DebugMon	调试监控器	0x0000_0030
-3	-	-	保留	0x0000_0034
-2	5	PendSV	可挂起的系统服务	0x0000_0038
-1	6	SysTick	系统嘀嗒定时器	0x0000_003C
0	7	--	保留	0x0000_0040
1	8	--	保留	0x0000_0044
2	9	--	保留	0x0000_0048
3	10	RTC	实时时钟(RTC)全局中断	0x0000_004C
4	11	FLASH	FLASH 全局中断	0x0000_0050
5	12	--	保留	0x0000_0054
6	13	P0	GPIOA 中断	0x0000_0058
7	14	P1	GPIOB 中断	0x0000_005C
8	15	P2	GPIOC 中断	0x0000_0060
9	16	P3	GPIOD 中断	0x0000_0064
10	17	--	保留	0x0000_0068
11	18	DMA	DMA 中断	0x0000_006C
12	19	--	保留	0x0000_0070
13	20	--	保留	0x0000_0074
14	21	--	保留	0x0000_0078
15	22	--	保留	0x0000_007C
16	23	--	保留	0x0000_0080
17	24	--	保留	0x0000_0084
18	25	ADC	ADC 中断	0x0000_0088
19	26	--	保留	0x0000_008C
20	27	--	保留	0x0000_0090
21	28	--	保留	0x0000_0094
22	29	--	保留	0x0000_0098
23	30	--	保留	0x0000_009C
24	31	PWM	TIMER0 全局中断	0x0000_00A0
25	32	TMR1	TIMER1 全局中断	0x0000_00A4
26	33	--	保留	0x0000_00A8
27	34	--	保留	0x0000_00AC
28	35	TMR2	TIMER2 全局中断	0x0000_00B0
29	36	TMR3	TIMER3 全局中断	0x0000_00B4
30	37	TMR4	TIMER4 全局中断	0x0000_00B8
31	38	I2C	I2C 全局中断	0x0000_00BC
32	39	--	保留	0x0000_00C0
33	40	--	保留	0x0000_00C4
34	41	--	保留	0x0000_00C8
35	42	SPI0	SPI0 全局中断	0x0000_00CC
36	43	SPI1	SPI1 全局中断	0x0000_00D0
37	44	UART0	UART0 全局中断	0x0000_00D4

38	45	UART1	UART1 全局中断	0x0000_00D8
39	46	UART2	UART2 全局中断	0x0000_00DC
40	47	--	保留	0x0000_00E0
41	48	--	保留	0x0000_00E4
42	49	--	保留	0x0000_00E8
43	50	--	保留	0x0000_00EC
44	51	--	保留	0x0000_00F0
45	52	--	保留	0x0000_00F4
46	53	--	保留	0x0000_00F8
47	54	--	保留	0x0000_00FC
48	55	--	保留	0x0000_0100
49	56	USB1.1	USB 全局中断	0x0000_0104
50	57	--	保留	0x0000_0108
51	58	QSPI	QSPI 全局中断	0x0000_010C
52	59	--	保留	0x0000_0110
53	60	--	保留	0x0000_0114
54	61	--	保留	0x0000_0118
55	62	--	保留	0x0000_011C
56	63	--	保留	0x0000_0120
57	64	TMR5	TIMER5 全局中断	0x0000_0124
58	65	--	保留	0x0000_0128
59	66	AES	AES 加密中断	0x0000_012C

11 系统控制

系统控制主要包括配置系统时钟，系统工作模式

11.1 寄存器列表

地址	寄存器	描述	备注
0x4002_1000	--	--	--
0x4002_1004	PMU_CTRL	PMU 控制寄存器	
0x4002_1008	LOCKUP_RESET	处理器锁死控制寄存器	
0x4002_100C	SYSCLK_CON	系统时钟控制寄存器	
0x4002_1010	RSTINFO	复位状态查询寄存器	
0x4002_1014	RETRIMING	系统重配置控制寄存器	
0x4002_1018	RETRIMING KEY	系统重配置密钥寄存器	
0x4002_101C	GPIO_WAKEUP_ST	GPIO 唤醒信号状态查询	
0x4002_1020	CHIPID	芯片 ID 查询	
0x4002_1024	UUID0	UUID0 查询	
0x4002_1028	UUID1	UUID1 查询	
0x4002_102C	UUID2	UUID2 查询	

0x4002_1030	UUID3	UUID3 查询	

11.2 寄存器描述

11.2.1 PMU 控制寄存器 PMU_CTRL

位域	类型	复位值	名称	描述
[31:15]	W	0x00	LPPMU KEY	低功耗使能密钥
[2]	--	--	--	--
[1]	R/W	0	LPPMUENABLE	进入低功耗模式使能
[0]	R/W	0	PMUENABLE	进入睡眠模式或深度睡眠模式使能

Note:

1.当密钥配置为 0x5049 时，低功耗使能才有效

11.2.2 锁死控制寄存器 LOCKUP_RESET

位域	类型	复位值	名称	描述
[31:1]	--	--	--	--
[0]	R/W	0	LOCKUP_RESET	处理器锁死使能复位

11.2.3 系统时钟选择寄存器 SYSCLK_CON

位域	类型	复位	名称	描述
[31:6]	--	--	--	--
[10]	R/W	0x0	pclk_div_en	pclk 除频使能 0x0: pclk 除频禁止 0x1: pclk 除频使能
[9:7]	R/W	0x0	pclk_div	pclk 除频器 $pclk = \frac{fclk}{pclk\_div + 1}$
[6]	R/W	0x0	pll_src_sel	pll 时钟源选择 0b：RC24M 1b: CRYSTAL24M
[5:2]	R/W	0x0	CLKSEL_ST	系统时钟选择状态查询：(Note4) 0x1：选择内部 RC24M 高速时钟 0x2：选择外部晶振/外部输入 0x4：选择 PLL 高速时钟 0x8：选择 FLASHPMU 的 RC 时钟
[1:0]	R/W	0x0	SYSCLK_SEL	系统时钟选择： 0x0：内部 RC24M 0x1：外部高速晶振/外部输入(Note1) 0x2：PLL 高速时钟(Note2) 0x3：FLASHPMU 的 RC 时钟(Note3)

Note

1. 当外部晶振使能有效的情况下，系统时钟才能选择外部晶振；当配置外部晶振 BYPASS,系统时钟才能选择外部输入。
2. 当 PLL 使能有效的情况下，系统时钟才能选择 PLL 高速时钟。
3. 当 FLASHPMU 的 RC 时钟使能有效的情况下，系统时钟才能选择 FLASHPMU 的 RC 时钟

4. 配置 SYSCLK_SEL 之后，可查询 CLKSEL_ST 寄存器，查看系统时钟是否选择相应的时钟源

11.2.4 复位状态查询寄存器 RSTINFO

位域	类型	复位	名称	描述
[31:6]	--	--	--	--
[5]	R/WC	0	FLASH_VDH	FLASH 高电压检测复位
[4]	R/WC	0	FLASH_VDL	FLASH 低电压检测复位
[3]	R/WC	0	EXT_RSTB	外部引脚复位
[2]	R/WC	0	LOCKUPRESET	锁死对系统复位
[1]	R/WC	0	WDGRESETREQ	看门狗对系统复位
[0]	R/WC	0	SYSRESETREQ	软件执行命令对系统复位

Note:

1. 此寄存器只是查询引起系统复位的信号查询
2. 当寄存器执行清零操作时，只是对寄存器清零，不会对复位信号有任何影响

11.2.5 系统重配置控制寄存器 RETRIMING

位域	类型	复位	名称	描述
[31:3]	--	--	--	--
[2]	R/W	0	Retriming	系统重配置使能(Note1)
[1]	R/W	0	FLASH_ENVDLB	使能 FLASH VDL 低电压检测复位 0：当 FLASH VDL 检测试低电压有效，系统不复位 1：当 FLASH VDL 检测试低电压有效，系统复位
[0]	R/W	0	FLASH_ENVDHB	使能 FLASH VDH 高电压检测复位 0：当 FLASH VDH 检测试低电压有效，系统不复位 1：当 FLASH VDH 检测试低电压有效，系统复位

Note:

1. 需要先配置正确的 RETRIMING 密钥，系统重配置才有效，见寄存器 RETIMING KEY

11.2.6 系统重配置密钥寄存器 RETRIMING KEY

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0	Retriming Key	系统重配置密钥(Note1)

Note:

1. 当密钥配置为 0xAB56 时，系统重配置使能才有效

11.2.7 GPIO 唤醒信号查询寄存器 GPIO_WAKE_ST

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15]	R	0	GPIOA[15]	使用 GPIOA[15]唤醒
[14]	R	0	GPIOA[14]	使用 GPIOA[14]唤醒
[13]	R	0	GPIOA[13]	使用 GPIOA[13]唤醒
[12]	R	0	GPIOA[12]	使用 GPIOA[12]唤醒
[11]	R	0	GPIOA[11]	使用 GPIOA[11]唤醒
[10]	R	0	GPIOA[10]	使用 GPIOA[10]唤醒

[9]	R	0	GPIOA[9]	使用 GPIOA[9]唤醒
[8]	R	0	GPIOA[8]	使用 GPIOA[8]唤醒
[7]	R	0	GPIOA[7]	使用 GPIOA[7]唤醒
[6]	R	0	GPIOA[6]	使用 GPIOA[6]唤醒
[5]	R	0	GPIOA[5]	使用 GPIOA[5]唤醒
[4]	R	0	GPIOA[4]	使用 GPIOA[4]唤醒
[3]	R	0	GPIOA[3]	使用 GPIOA[3]唤醒
[2]	R	0	GPIOA[2]	使用 GPIOA[2]唤醒
[1]	R	0	GPIOA[1]	使用 GPIOA[1]唤醒
[0]	R	0	GPIOA[0]	使用 GPIOA[0]唤醒

11.2.8 芯片 ID 寄存器查询 CHIPID

位域	类型	复位	名称	描述
[31:0]	R	0x54484F52	CHIPID	芯片 ID

11.2.9 芯片 UUID0 寄存器查询

位域	类型	复位	名称	描述
[31:0]	R	--	UUID0	见 FLASH NVR 的 UUID0 配置值

11.2.10 芯片 UUID1 寄存器查询

位域	类型	复位	名称	描述
[31:0]	R	--	UUID1	见 FLASH NVR 的 UUID1 配置值

11.2.11 芯片 UUID2 寄存器查询

位域	类型	复位	名称	描述
[31:0]	R	--	UUID2	见 FLASH NVR 的 UUID2 配置值

11.2.12 芯片 UUID3 寄存器查询

位域	类型	复位	名称	描述
[31:0]	R	--	UUID3	见 FLASH NVR 的 UUID3 配置值

12GPIO

片上有 4 组 GPIO，分别为 PORTA、PORTB，PORTC 和 PORTD。

12.1 特征

- 全双向输入输出，引脚单个可配置
- 输入模式下引脚呈高阻抗
- 可选的内部弱上拉或弱下拉
- 可选的施密特功能
- 支持OpenDrain功能
- 支持1.2V关断
- 支持外部输入中断

- 可选驱动电流4mA、16mA、20mA、40mA

12.2 输入输出方向控制

IO操作支持全双向的输入或输出，其工作模式可分为下列两种之一并可由软件在运行过程中任意切换：

- 高或低电平
- 高阻抗输入，可选择启用内部上拉或下拉

在应用中对于多余引脚的处理，可设为输出模式。若设为输入模式，则必须考虑上拉或下拉（使用内部上拉/下拉，或外加上拉/下拉电阻），避免引脚浮空而造成额外漏电流功耗。

12.3 端口输入中断

PORT 端口支持输入中断。PORT 任意引脚可通过四对寄存器，实现不同模式的输入中断响应。需要注意的是 PORT 引脚在任何时候（包括设为端口输出模式或外设使能状态下）都具备引脚中断触发和响应功能。应用软件必须鉴别所需引脚的中断需求，对 PTnIEC 寄存器相关位写 1（中断使能清零），以禁止无关的引脚产生中断。

控制外部输入端口中断的四对寄存器分别是 PTnIES, PTnIEC, PTnITS0, PTnITC0，PTnITS1, PTnITC1 和 PTnPLS, PTnPLC。其中 PTnIES, PTnIEC 控制端口中断的使能和禁止；PTnITS0, PTnITC0，PTnITS1, PTnITC1 控制端口中断的类型；PTnPLS, PTnPLC 控制端口中断的极性。

端口中断使能位	端口中断极性位	端口中断类型位0	端口中断类型位1	中断响应模式
0	-	-	-	中断禁止
1	0	0	0	低电平
1	0	1	0	下降沿
1	1	0	0	高电平
1	1	1	0	上升沿
1	X	X	1	电平变化

任一引脚所选择的中断被触发后，状态寄存器PT0IST中的相应位被置1。所有引脚的中断请求都由同一个中断服务程序得到响应，用户软件在中断服务程序中需要查询PTnIST和引脚状态确定具体的引脚中断源，并对PTnIST的对应位写1来清除中断标志。

12.4 端口输入内部上拉或下拉

当引脚被配置成端口输入模式时，每个引脚都可以独立控制启用片内的弱上拉或下拉电阻（等效阻值约为10K）。在引脚设为端口输出模式时，该上拉或下拉将被自动禁止。

12.5 端口复用功能

IO 口通常默认功能为 GPIO（Debug 口等除外）。通过对寄存器 PTnPES, PTnPEC 寄存器操作，可以开启或关闭 IO 口的复用功能，如 SPI 口，I2C 口等等。

一个 PORT 口的第 i 位端口复用功能，由 PTnPES 寄存器的第[2i+1:2i]位域值控制。当[2i+1:2i]为不同值时，分别对应不同的复用功能，如下：

- 00：主功能
- 01：复用功能 1
- 10：复用功能 2

- 11：复用功能 3
  复用定义可查看第 4 章节管脚定义

12.6 寄存器列表

地址	寄存器	描述	备注
0x4001_0000	PT0DAT	PORTA 端口数据寄存器	
0x4001_0004	PT0LAT	PORTA 端口数据输出锁存寄存器	
--	--	--	--
0x4001_0010	PT0OES	PORTA 引脚输出使能寄存器	
0x4001_0014	PT0OEC	PORTA 引脚输出禁止寄存器	
0x4001_0018	PT0PES	PORTA 引脚外设复用使能寄存器	
0x4001_001C	PT0PEC	PORTA 引脚外设复用禁止寄存器	
0x4001_0020	PT0IES	PORTA 引脚输入中断使能寄存器	
0x4001_0024	PT0IEC	PORTA 引脚输入中断禁止寄存器	
0x4001_0028	PT0ITS	PORTA 引脚中断类别设置寄存器	
0x4001_002C	PT0ITC	PORTA 引脚中断类别清除寄存器	
0x4001_0030	PT0PLS	PORTA 引脚中断极性设置寄存器	
0x4001_0034	PT0PLC	PORTA 引脚中断极性清除寄存器	
0x4001_0038	PT0IST	PORTA 引脚中断标志寄存器	
0x4001_003C	PT0PUS	PORTA 引脚内部上拉使能寄存器	
0x4001_0040	PT0PUC	PORTA 引脚内部上拉禁止寄存器	
0x4001_0044	PT0ODS	PORTA 引脚开漏输出使能寄存器	
0x4001_0048	PT0ODC	PORTA 引脚开漏输出禁止寄存器	
--	--	--	
0x4001_0054	PT0DRVS	PORTA 引脚输出驱动选择寄存器	
0x4001_0058	PT0DRVC	PORTA 引脚输出驱动禁止寄存器	
0x4001_005C	PT0SES	PORTA 引脚输入施密特使能寄存器	
0x4001_0060	PT0SEC	PORTA 引脚输入施密特禁止寄存器	
0x4001_1000	PT1DAT	PORTB 端口数据寄存器	
0x4001_1004	PT1LAT	PORTB 端口数据输出锁存寄存器	
--	--	--	
0x4001_1010	PT1OES	PORTB 引脚输出使能寄存器	
0x4001_1014	PT1OEC	PORTB 引脚输出禁止寄存器	
0x4001_1018	PT1PES	PORTB 引脚外设复用使能寄存器	
0x4001_101C	PT1PEC	PORTB 引脚外设复用禁止寄存器	
0x4001_1020	PT1IES	PORTB 引脚输入中断使能寄存器	
0x4001_1024	PT1IEC	PORTB 引脚输入中断禁止寄存器	
0x4001_1028	PT1ITS	PORTB 引脚中断类别设置寄存器	

0x4001_102C	PT1ITC	PORTB 引脚中断类别清除寄存器	
0x4001_1030	PT1PLS	PORTB 引脚中断极性设置寄存器	
0x4001_1034	PT1PLC	PORTB 引脚中断极性清除寄存器	
0x4001_1038	PT1IST	PORTB 引脚中断标志寄存器	
0x4001_103C	PT1PUS	PORTB 引脚内部上拉使能寄存器	
0x4001_1040	PT1PUC	PORTB 引脚内部上拉禁止寄存器	
0x4001_1044	PT1ODS	PORTB 引脚开漏输出使能寄存器	
0x4001_1048	PT1ODC	PORTB 引脚开漏输出禁止寄存器	
--	--	--	
0x4001_1054	PT1DRVS	PORTB 引脚输出驱动选择寄存器	
0x4001_1058	PT1DRVC	PORTB 引脚输出驱动禁止寄存器	
0x4001_105C	PT1SES	PORTB 引脚输入施密特使能寄存器	
0x4001_1060	PT1SEC	PORTB 引脚输入施密特禁止寄存器	
0x4001_2000	PT2DAT	PORTC 端口数据寄存器	
0x4001_2004	PT2LAT	PORTC 端口数据输出锁存寄存器	
--	--	--	
0x4001_2010	PT2OES	PORTC 引脚输出使能寄存器	
0x4001_2014	PT2OEC	PORTC 引脚输出禁止寄存器	
0x4001_2018	PT2PES	PORTC 引脚外设复用使能寄存器	
0x4001_201C	PT2PEC	PORTC 引脚外设复用禁止寄存器	
0x4001_2020	PT2IES	PORTC 引脚输入中断使能寄存器	
0x4001_2024	PT2IEC	PORTC 引脚输入中断禁止寄存器	
0x4001_2028	PT2ITS	PORTC 引脚中断类别设置寄存器	
0x4001_202C	PT2ITC	PORTC 引脚中断类别清除寄存器	
0x4001_2030	PT2PLS	PORTC 引脚中断极性设置寄存器	
0x4001_2034	PT2PLC	PORTC 引脚中断极性清除寄存器	
0x4001_2038	PT2IST	PORTC 引脚中断标志寄存器	
0x4001_203C	PT2PUS	PORTC 引脚内部上拉使能寄存器	
0x4001_2040	PT2PUC	PORTC 引脚内部上拉禁止寄存器	
0x4001_2044	PT2ODS	PORTC 引脚开漏输出使能寄存器	
0x4001_2048	PT2ODC	PORTC 引脚开漏输出禁止寄存器	
--	--	--	
0x4001_2054	PT2DRVS	PORTC 引脚输出驱动选择寄存器	
0x4001_2058	PT2DRVC	PORTC 引脚输出驱动禁止寄存器	
0x4001_205C	PT2SES	PORTC 引脚输入施密特使能寄存器	
0x4001_2060	PT2SEC	PORTC 引脚输入施密特禁止寄存器	
0x4001_5000	PT3DAT	PORTD 端口数据寄存器	

0x4001_5004	PT3LAT	PORTD 端口数据输出锁存寄存器	
--	--	--	
0x4001_5010	PT3OES	PORTD 引脚输出使能寄存器	
0x4001_5014	PT3OEC	PORTD 引脚输出禁止寄存器	
0x4001_5018	PT3PES	PORTD 引脚外设复用使能寄存器	
0x4001_501C	PT3PEC	PORTD 引脚外设复用禁止寄存器	
0x4001_5020	PT3IES	PORTD 引脚输入中断使能寄存器	
0x4001_5024	PT3IEC	PORTD 引脚输入中断禁止寄存器	
0x4001_5028	PT3ITS	PORTD 引脚中断类别设置寄存器	
0x4001_502C	PT3ITC	PORTD 引脚中断类别清除寄存器	
0x4001_5030	PT3PLS	PORTD 引脚中断极性设置寄存器	
0x4001_5034	PT3PLC	PORTD 引脚中断极性清除寄存器	
0x4001_5038	PT3IST	PORTD 引脚中断标志寄存器	
0x4001_503C	PT3PUS	PORTD 引脚内部上拉使能寄存器	
0x4001_5040	PT3PUC	PORTD 引脚内部上拉禁止寄存器	
0x4001_5044	PT3ODS	PORTD 引脚开漏输出使能寄存器	
0x4001_5048	PT3ODC	PORTD 引脚开漏输出禁止寄存器	
--	--	--	
0x4001_5054	PT3DRVS	PORTD 引脚输出驱动选择寄存器	
0x4001_5058	PT3DRVC	PORTD 引脚输出驱动禁止寄存器	
0x4001_505C	PT3SES	PORTD 引脚输入施密特使能寄存器	
0x4001_5060	PT3SEC	PORTD 引脚输入施密特禁止寄存器	

12.7 寄存器描述

12.7.1 PORT 端口数据捕获寄存器 PTnDAT

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnDAT	捕获 PORT 端口数据(Note1)

Note:

1. 读操作时，即是端口引脚实际电平逻辑
2. 写操作时，实际写的是 PTnLAT 寄存器

12.7.2 PORT 端口数据输出寄存器 PTnDAT

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnLAT	PORT 端口输出电平(Note1)

Note:

1. PORT 端口需要配置为输出
2. PORT 端口配置主功能 GPIO

12.7.3 PORT 端口输出使能寄存器 PTnOES

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnOES	PORT 端口输出使能(Note1)

Note:

1. 寄存器只能写 1，不能写 0
2. 当需要寄存器清零时，需要配置寄存器 PTnOEC

12.7.4 PORT 端口输出禁止寄存器 PTnOEC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	W	0x0000	PTnOEC	PORT 端口输出禁止(Note1)

Note:

1. 寄存只能写 1，写 0 无效
2. 只用于对寄存器 PTnOES 清零

12.7.5 PORT 端口功能复用寄存器 PTnPES

位域	类型	复位	名称	描述
[31:30]	R/W	0x0	PTnPES[31:30]	PORTn[15]端口复用
[29:28]	R/W	0x0	PTnPES[29:28]	PORTn[14]端口复用
[27:26]	R/W	0x0	PTnPES[27:26]	PORTn[13]端口复用
[25:24]	R/W	0x0	PTnPES[25:24]	PORTn[12]端口复用
[23:22]	R/W	0x0	PTnPES[23:22]	PORTn[11]端口复用
[21:20]	R/W	0x0	PTnPES[21:20]	PORTn[10]端口复用
[19:18]	R/W	0x0	PTnPES[19:18]	PORTn[9]端口复用
[17:16]	R/W	0x0	PTnPES[17:16]	PORTn[8]端口复用
[15:14]	R/W	0x0	PTnPES[15:14]	PORTn[7]端口复用
[13:12]	R/W	0x0	PTnPES[13:12]	PORTn[6]端口复用
[11:10]	R/W	0x0	PTnPES[11:10]	PORTn[5]端口复用
[9:8]	R/W	0x0	PTnPES[9:8]	PORTn[4]端口复用
[7:6]	R/W	0x0	PTnPES[7:6]	PORTn[3]端口复用
[5:4]	R/W	0x0	PTnPES[5:4]	PORTn[2]端口复用
[3:2]	R/W	0x0	PTnPES[3:2]	PORTn[1]端口复用
[1:0]	R/W	0x0	PTnPES[1:0]	PORTn[0]端口复用

Note:

1. 寄存只能写 1，写 0 无效
2. 只用于对寄存器 PTnPEC 清零
3. 相应的端口复用参考第 4 章节管脚定义
4. PTnPES[n+1:n]

   00:主功能（GPIO）

   01:复用功能 1

   10:复用功能 2

   11:复用功能 3

12.7.6 PORT 端口功能复用禁止寄存器 PTnPEC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	W	0x0000	PTnPEC	PORT 端口输出禁止(Note1)

Note:

1. 寄存只能写 1，写 0 无效
2. 只用于对寄存器 PTnPES 清零

12.7.7 PORT 端口输入中断使能寄存器 PTnIES

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnIES	PORT 输入中断使能(Note1)

Note:

1. 寄存只能写 1，写 0 无效
2. 只能用寄存器 PTnIEC 清零

12.7.8 PORT 端口输入中断禁止寄存器 PTnIEC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	W	0x0000	PTnIEC	PORT 输入中断禁止(Note1)

Note:

1. 寄存只能写 1，写 0 无效
2. 用于对寄存器 PTnIES 清零

12.7.9 PORT 端口中断类别寄存器 PTnITS

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnITS	PORT 输入沿跳变产生中断(Note1)

Note:

1. 寄存只能写 1，写 0 无效
2. 只能用寄存器 PTnITC 清零
3. 读取值为 1 时，中断类别选择为沿跳变，读取值为 0 时，中断类别为电平

12.7.10 PORT 端口中断类别清零寄存器 PTnITC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnITC	

Note:

1. 寄存只能写 1，写 0 无效
2. 用于对寄存器 PTnITS 清零

12.7.11 PORT 端口中断极性寄存器 PTnPLS

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnITS	PORT 输入极性选择中断(Note1)

Note:

1. 寄存只能写 1，写 0 无效
2. 只能用寄存器 PTnPLC 清零
3. 读取值为 1 时，中断为上升沿跳变或高电平，读取值为 0 时，中断类别为下降沿或低电平

12.7.12 PORT 端口中断极性清零寄存器 PTnIST

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/WC	0x0000	PTnIST	

Note:

1. 读取 1，端口输入中断产生
2. 写入 1，中断标志位清零

12.7.13 PORT 端口上拉使能寄存器 PTnPUS

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnPUS	

Note:

1. 寄存只能写 1，写 0 无效
2. 只能用寄存器 PTnPUC 清零

12.7.14 PORT 端口中断类别清零寄存器 PTnPUC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	W	0x0000	PTnPUC	

Note:

1. 寄存只能写 1，写 0 无效
2. 用于对寄存器 PTnPUS 清零

12.7.15 PORT 端口上拉使能寄存器 PTnPDS

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnPDS	

Note:

1. 寄存只能写 1，写 0 无效
2. 只能用寄存器 PTnPDC 清零

12.7.16 PORT 端口中断类别清零寄存器 PTnPDC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	W	0x0000	PTnPDC	

Note:

1. 寄存只能写 1，写 0 无效
2. 用于对寄存器 PTnPDS 清零

12.7.17 PORT 端口输入施密特使能寄存器 PTnSES

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0000	PTnSES	

Note:

1. 寄存只能写 1，写 0 无效
2. 只能用寄存器 PTnSEC 清零

12.7.18 PORT 端口输入施密特禁止寄存器 PTnSEC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	W	0x0000	PTnSEC	

Note:

1. 寄存只能写 1，写 0 无效
2. 用于对寄存器 PTnSES 清零

12.7.19 PORT 端口输出驱动选择寄存器 PTnDRVS

位域	类型	复位	名称	描述
[31:30]	R/W	0x0	PTnDRVS[31:30]	PORTn[15]端口驱动选择
[29:28]	R/W	0x0	PTnDRVS[29:28]	PORTn[14]端口驱动选择
[27:26]	R/W	0x0	PTnDRVS[27:26]	PORTn[13]端口驱动选择
[25:24]	R/W	0x0	PTnDRVS[25:24]	PORTn[12]端口驱动选择
[23:22]	R/W	0x0	PTnDRVS[23:22]	PORTn[11]端口驱动选择
[21:20]	R/W	0x0	PTnDRVS[21:20]	PORTn[10]端口驱动选择
[19:18]	R/W	0x0	PTnDRVS[19:18]	PORTn[9]端口驱动选择
[17:16]	R/W	0x0	PTnDRVS[17:16]	PORTn[8]端口驱动选择
[15:14]	R/W	0x0	PTnDRVS[15:14]	PORTn[7]端口驱动选择
[13:12]	R/W	0x0	PTnDRVS[13:12]	PORTn[6]端口驱动选择
[11:10]	R/W	0x0	PTnDRVS[11:10]	PORTn[5]端口驱动选择
[9:8]	R/W	0x0	PTnDRVS[9:8]	PORTn[4]端口驱动选择
[7:6]	R/W	0x0	PTnDRVS[7:6]	PORTn[3]端口驱动选择
[5:4]	R/W	0x0	PTnDRVS[5:4]	PORTn[2]端口驱动选择
[3:2]	R/W	0x0	PTnDRVS[3:2]	PORTn[1]端口驱动选择

[1:0]	R/W	0x0	PTnDRVS[1:0]	PORTn[0]端口驱动选择

Note:

1. 寄存只能写 1，写 0 无效
2. 只用于对寄存器 PTnDRVC 清零
3. PTnDRVS[n+1:n]

   00:4mA

   01:16mA

   10:20mA

   11:40mA

12.7.20 PORT 端口输出驱动清零寄存器 PTnDRVC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	W	0x0000	PTnDRVC	

Note:

1. 寄存只能写 1，写 0 无效
2. 用于对寄存器 PTnDRVS 清零

13 UART 通信

13.1 UART 模块综述

PT32Z192 系列内集成了 3 个功能完全一致但相互独立的 UART 通讯模块，分别为 UART0、UART1、UART2。下文叙述中将其统一描述成 UART*n*（n=0~2）。应用系统可以利用这 3 个 UART 模块实现和外界的异步数据通讯（例如 RS232、RS485 等）。UART 模块主要特性包括：

- 灵活可编程的波特率设定，支持业内标准 9600bps、19200bps、28800bps、38400bps、57600bps、 115.2Kbps 等，或其它特殊应用的波特率
- 可编程的数据传输格式：8 位数据、7 位数据+1 位奇偶校验、8 位数据+1 位奇偶校验可编程奇或偶校验方式
- 可编程设定 0.5 位、1 位、1.5 位或 2 位停止位
- 带 FIFO 缓冲的数据发送：8 或 9 位数据模式下，缓冲深度 16 级，最大宽度 9 位；16 位数据模式下，缓冲深度 8 级；32 位数据模式下，缓冲深度 4 级
- 带 FIFO 缓冲的数据接收：8 或 9 位数据模式下，缓冲深度 16 级，最大宽度 9 位；16 位数据模式下，缓冲深度 8 级；
- 数据收发全双工
- 硬件错误检测
- 可选 UART 正反向数据同步接收以支持 RS485 无极性连接；可选发送数据极性
- 支持 DMA 数据传输

13.2 UART 基本功能

UART 模块的功能和工作方式通过配置控制寄存器 UART*n*_CTL 来实现，该寄存器包含若干用于模式选择和错误检测的控制位，外加一些状态标志位。

往发送缓冲寄存器 UART*n*_TXB（物理地址为 UART*n*_DAT0，只写）内写入一个数据

后，立即启动数据的串行发送过程。UART*n*_TXB 实际上是一个含 16 级 FIFO 缓冲的寄存器组，允许应用软件一次可以连续写入多个发送数据，直至 FIFO 队列满，这样可实现多个数据帧的连续不停顿发送，提高发送效率，降低 CPU 的干预度。

数据接收包含逻辑功能完全相同的两路，可通过使能控制位（UART*n*_CTL.RXEN0 和 UART*n*_CTL.RXEN1）选择接收正相和反相数据接收。一次数据接收过程完成后，接收到的数据及其附加的校验位信息被存入两路独立的接收缓冲寄存器 UART*n*_RXB0（物理地址为 UART*n*_DAT0，只读）和 UART*n*_RXB1（物理地址为 UART*n*_DAT1，只读），并可通过应用软件读出。UART*n*_RXB0/UART*n*_RXB1 实际为 16 级 FIFO 缓冲的寄存器组，无论应用软件是否立即读取刚接收到的数据，UART 模块可继续接收后续的数据并将它们逐个存入接收缓存，直至接收 FIFO 队列满。在接收队列满后如果又有新的数据被收到，则将发生队列溢出的错误，并将状态标志位 OVERR 置 1，此时最前收到的一个数据被最新的数据顶出队列而丢弃，UART*n*_ST 寄存器被相应更新以反映该次新数据的接收。利用硬件的 FIFO 缓冲，应用软件可以在一次从接收队列中连续读取多个数据（只要队列中有数据），提高了 CPU 的运行效率。

UART 模块提供片内数据发送端至接收端的回绕功能，无需外部硬件电路和实际线路连接，即可方便地进行软件自诊断。

UART 的数据收发全双工模式使用相同的数据帧结构和传输波特率。发送的数据位信息被送至 TXD 引脚输出，接收的数据位信息则应接入 RXD 引脚．同时需注意：

- 只有当波特率发生器的控制位（RUN）被置 1 后，才能进行串行数据的收发．如果 RUN 位被清 0，则数据收发过程将立即终止，TXD 引脚输出始终保持空闲态（正常极性为高电平 1，反相极性为低电平 0）。除非应用程序确定 UART 通讯处于空闲状态，不然 RUN 位应始终被置为 1。
- 如果将控制模式（UART*n*_CTL.MODE）配置成某一个未定义的保留模式，UART 模块的工作将不可预测
- 8 位数据加唤醒位模式时，必须屏蔽奇偶校验错误中断以防止假的奇偶校验错误．因为在这两种配置模式下数据帧内没有奇偶校验位信息。

13.3 UART 工作模式

8 位数据帧有两种组成形式：

- 8 位有效数据位（UART*n*_CTL.MODE=001b）
- 7 位有效数据位加上 1 位奇偶校验位（UART*n*_CTL.MODE=011b）

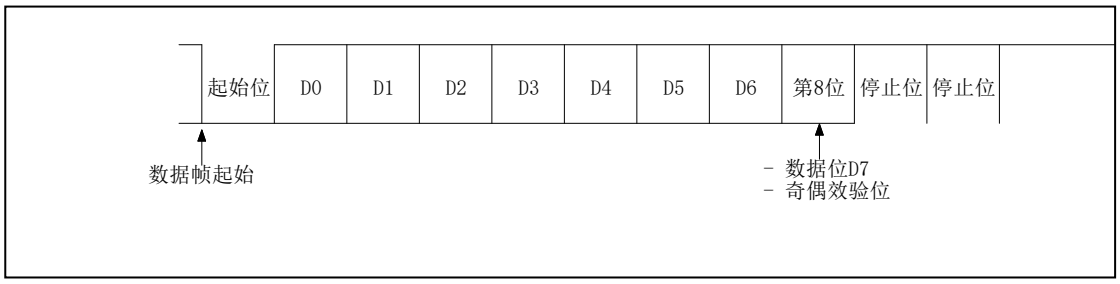


图 13-1：UART 8 位数据帧格式

9 位数据帧有三种组成形式：

- 9 位有效数据位（UART*n*_CTL.MODE=100b）
- 8 位有效数据位加上 1 位奇偶校验位（UART*n*_CTL.MODE=111b）
- 8 位有效数据位加上 1 位唤醒位（UART*n*_CTL.MODE=101b）

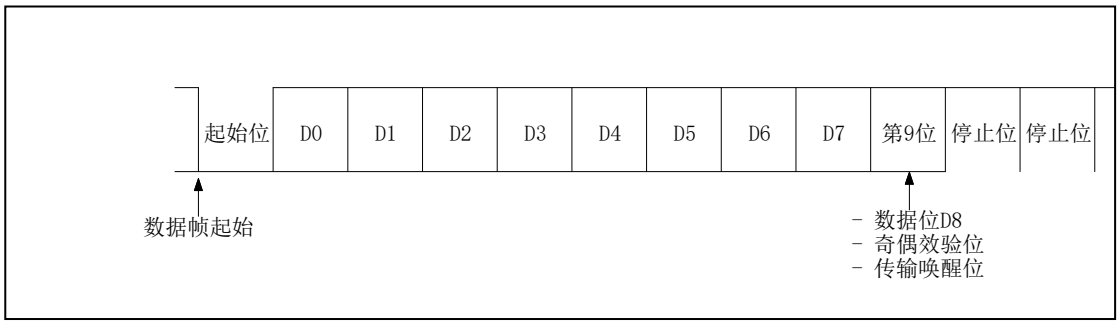


图 13-2 UART 9 位数据帧格式

发送校验位的产生可以是奇校验或偶校验，取决于 UART*n*_CTL.PAR 位的设定。设为奇校验时，如果有效数据位中的 1 为偶数个，则校验位被置 1；在偶校验时此校验位则被置 0。接收时，校验位和有效数据位一起被存入接收缓冲 UART*n*_RXB 队列中。

在 UART 处于传输唤醒模式时，只有当 RXD 引脚送入的数据帧其第 9 位为 1 时，接收结果才会被存入 UART*n*_RXB 缓冲队列并产生接收中断请求；如果此第 9 位为 0，则无任何中断请求，也不会往 UART*n*_RXB 存入任何数据。这一特性非常适用于主-从结构的多机通讯应用。当主机需要发送一串数据至某一指定地址的从机时，它先发送 8 位有效地址外加第 9 位为 1 的一个 9 位寻址帧，所有从机都能收到这个特殊的寻址帧（因其第 9 位为 1）并和本机设定地址作比对，如果地址相符，则立即将本机的 UART 工作模式转为 9 位数据模式，继续接收后面的数据帧（数据帧第 9 位为 0），地址不符的从机则不会响应后续的数据帧，避免过多无谓的中断响应，提高系统运行效率。

13.4 UART 波特率计算和设定

UART 模块含自动重载的波特率发生器，由 RUN 位控制其工作或停止，定时控制寄存器 UART*n*_BR 分为两个区域：BR 区域的宽度为 16 位，作为系统时钟分频；SR 区域为 2 位宽，用于控制每位码元的采样率. 软件通过设置 UART*n*_BR 的值设定 UART 通讯的波特率，具体计算公式如下：

$$BPS_{UART} = f_{SYS} / (16/(2^{SR})*BR) \quad 或$$

$$BR = f_{SYS} / (16/(2^{SR})*BPS_{UART})$$

其中：

- BPS_{UART} 为期望的通讯波特率，同时用于接收和发送
- f_{SYS} 为系统时钟频率
- BR 为寄存器 UART*n*_BR 内的 BR 域设定值
- SR 为寄存器 UART*n*_BR 内的 SR 域设定值

以下列出了在不同系统时钟频率下常用的波特率设定值和误差（SR=00b），可供应用时快速参考。

常用的波特率设定系数

	f_{CLK} = 20MHz		f_{CLK} = 16MHz		f_{CLK} = 10MHz		f_{CLK} = 8MHz	
波特率	UART*n*_BR	误差	UART*n*_BR	误差	UART*n*_BR	误差	UART*n*_BR	误差
38400	-	-	0x001A	-0.16%	-	-	0x000D	-0.16%
28800	0x002B	-0.94%	0x0023	0.79%	-	-	-	-
19200	0x0041	-0.16%	0x0034	-0.16%	-	-	0x001A	-0.16%
9600	0x0082	-0.16%	0x0068	-0.16%	0x0041	-0.16%	0x0034	-0.16%

4800	0x0104	-0.16%	0x00D0	-0.16%	0x0082	-0.16%	0x0068	-0.16%
2400	0x0209	0.03%	0x01A1	0.08%	0x0104	-0.16%	0x00D0	-0.16%
1200	0x0412	0.03%	0x0341	-0.04%	0x0209	0.03%	0x01A1	0.08%

如果需要设定特殊波特率，可按上述公式计算得到 UART*n*_BR 的设定值并确认实际波特率误差不超过 1%，不然数据收发将不能可靠进行。有时为了得到较高的波特率精度，可以考虑使用特殊频率的系统时钟。

13.5 UART 数据发送

在 RUN 位置 1 且有数据写入 UART*n*_TXB（亦即 UART*n*_DAT0）寄存器后，即启动数据帧的串行发送过程。发送的数据帧包含 3 个部分：

- 1 个起始位
- 若干数据位（8 或 9 位），低位先发
- 可选的 1 位校验位
- 停止位（0.5、1、1.5 或 2 位）

发送队列有 16 级的 FIFO 缓存，在先前写入的数据正忙于移位发送时，软件可以继续往 UART*n*_TXB 队列中写入发送数据，直至队列满．当最后一个数据帧的最后 1 位被实际发送出去后，发送器空闲状态标志 TXE 被置 1，告知所有数据的串行发送已完成。

应用软件可以设定串行数据发送的极性，普通极性时，无数据发送时 TXD 引脚保持高电平，发送数据时，起始位为低电平，停止位为高电平，数据位表达 0 为低电平，1 为高电平；反相极性时，所有电平和普通极性相反。提供反相发送模式可以方便不同电压系统间通讯接口的设计，亦可方便实现开漏驱动型通讯。

注意发送起始位延时(以 UART*n*_BR[SR]=00b 标准采样为例)：

- 在发送缓冲队列为非空时：从 UARTn_CTL[RUN]置 1 开始，到发送起始位延时为 $\frac{15\times t_{1\,码元}}{16}+t_{\Delta}$.
- 在 UARTn_CTL[RUN]为 1 时：从发送缓冲队列写入数据开始，到发送起始位延时为 $\frac{15\times t_{1\,码元}}{16}+t_{\Delta}$.
- 1UART 模块时钟$\leq t_{\Delta} \leq \frac{t_{1\,码元}}{16}$

13.6 UART 数据接收

在 RUN 位置 1，RXEN0/RXEN1 位之一或同时置 1 时，UART 模块准备接收串行数据．RXD 引脚上检测到起始位开始时（正相数据的下降沿，反相数据的上升沿），即引发一次数据帧的接收过程，UART 模块按既设波特率对每位数据进行 16 次采样并通过多数表决的方式判断数据位 0 或 1，避免偶发干扰造成接收数据误判。

当确认接收到最后一位停止位后，UART 模块将移位收到的数据转送入接收缓冲队列 UART*n*_RXB*m*（亦即 UART*n*_DAT*m*）（m=0/1），置队列非空标志 RXNE 为 1，若此时接收队列已满，则置队列满标志 RXF 为 1，但不会更新校验错误标志 PERR 和帧错误标志 FERR（接收数据校验标志将在数据被从接收队列中读出时更新）。然后模块准备接收新数据，等待下一个起始位的出现。

如果在数据帧接收过程中软件清除 RXEN*m* 位，当前正在接收的数据帧依然可以被完整的接收，并更新各状态标志。但后续的数据将不能被接收到。在接收唤醒模式下，只有第

9 位为 1 的数据帧才能被接收存入接收缓冲队列并更新各状态标志，第 9 位为 0 的数据帧被丢弃。

13.7 UART 错误检测

为确保串行数据通讯的可靠性，UART 模块提供了必要的通讯错误检测机制，并通过状态寄存器 UART*n*_ST*m* 内若干状态位反映不同的错误信息：

- 当 UART*n*_RXB*m* 队列接收数据溢出时，UART*n*_ST*m*.OVERR 被立即置 1。
- 当应用软件从接收队列中读取暂存的接收数据时，每读取一个数据后，其对应的校验信息将同步反映在 UART*n*_ST*m* 状态寄存器中：若发生奇偶校验错误，UART*n*_ST*m*.PERR 位被置 1；若出现帧接收错误（在停止位处没有检测到 1），UART*n*_ST*m*.FERR 被置 1。

所有的错误标志位，如果对应的中断使能位置 1，在相“或”后可触发 UART 模块的错误中断，软件必须分别查询判断并应对处理。

13.8 UART 数据 DMA 传输

片内的 3 个 UART 模块均支持数据收发的 DMA 传输。使用 DMA 时应用软件需事先在内存中开辟定义发送和接收的缓冲区，并设定特定 DMA 通道关联 UART 模块的外设地址和内存地址。

13.9 UART 中断响应

UART 模块内有两组寄存器，状态寄存器 UART*n*_ST*m* 和中断控制寄存器 UART*n*_IE*m*，用于控制数据通讯的中断响应。状态寄存器内的相关标志位反映发生的是何种中断，中断控制寄存器内的对应位决定了该中断是否被允许响应。多个被允许的中断标志信息通过逻辑“或”后，产生一个统一的串行通讯中断请求送入内核中断控制器，请求系统响应中断服务。

中断标志的清除方式按不同的标志而不同．发送相关的中断标志 TXE 和 TXHE 视写入发送缓冲队列数据的多少由硬件自动清 0；接收相关的中断标志 RXF 在软件从接收缓冲队列内读取一个数据且队列未满时被硬件自动清 0，RXNE 则必须等到软件读完接收缓冲队列内的全部数据后（接收缓冲队列为空）由硬件自动清 0；错误中断标志 PERR、FERR 和 OVERR 则只能通过由软件对中断状态寄存器 UART*n*_ST*m* 的对应位写 1 来清 0。

针对一般通讯过程的中断标志产生条件及含义可归纳如下。数据发送和接收有三个可用的中断标志。

数据发送标志：

- TXE：当队列中的最后一个数据被转入移位发送器后，队列被清空，该标志位被置 1，但有可能数据的发送移位正在进行过程中。空闲状态下没有数据发送时，队列一定是全空的，故软件使能该中断后，将立即进入中断服务程序。
- TXHE：当队列中的一个数据被转入移位发送器，且对列可用空间超过总长度的一半或以上，该标志位被置 1。此时发送缓冲队列为半空。
- TXEND：当最后一个数据的最后一位被实际发送完毕后，该标志位被置 1。此时发送缓冲队列一定为全空。

数据接收标志：

- RXNE：当接收到的一个数据从接收移位器转入接收缓冲队列 UART*n*_RXB 后，该标志位被置 1，指示接收缓冲队列为非空（队列中至少有一个数据）。
- RXHF：当接收到的一个数据从接收移位器转入接收缓冲队列 UART*n*_RXB，且队列

超过半满时，该标志位被置 1，指示接收缓冲队列已被使用超过一半以上，但尚有空间。

- RXF：当接收到的一个数据从接收移位器转入接收缓冲队列 UART*n*_RXB，且队列全满时，该标志位被置 1，指示接收缓冲队列已经全满。如果软件没有及时将队列中的数据读出，下一个接收的数据即会造成接收队列溢出错误。

正反相接收的上述两组标志位，如果对应的中断使能位置 1，在相“或”后触发接收中断，软件必须分别查询判断并应对处理。

内核系统为每个 UART 模块提供了一个独立的中断请求源，分别用于响应数据发送、数据接收和错误处理，归类如下：

- 发送中断：若 UART*n*_ST*m* 状态寄存器中的 TXEND、TXHE 或 TXE 位中任意位为 1 且 UART*n*_IE*m* 中断控制寄存器中的对应位也为 1，则 UART 模块向系统发出发送中断请求。在中断服务程序中，应用软件必须同时相关判别标志位和控制位，辨析是由哪几个位产生的中断。
- 接收中断：若 UART*n*_ST*m* 状态寄存器中的 RXF、RXHF 或 RXNE 位中任意位为 1 且 UART*n*_IE*m* 中断控制寄存器中的对应位也为 1，则 UART 模块向系统发出接收中断请求。在中断服务程序中，应用软件必须同时相关判别标志位和控制位，辨析是由哪几个位产生的中断。
- 错误中断：若 UART*n*_ST*m* 状态寄存器中的 TOIDLE、TONE、OVERR、FERR 或 PERR 位中任意位为 1 且 UART*n*_IE*m* 中断控制寄存器中的对应位也为 1，则 UART 模块向系统发出错误中断请求。在中断服务程序中，应用软件必须同时相关判别标志位和控制位，辨析是由哪几个位产生的中断。

13.10 寄存器列表

地址	寄存器	描述	备注
0x4000_4400	UART0_DAT0	UART0 收发数据寄存器缓冲队列 0	
0x4000_4404	UART0_CTL	UART0 模块控制寄存器	
0x4000_4408	UART0_BR	UART0 波特率控制寄存器	
0x4000_440C	UART0_IE0	UART0 接收队列 0 中断使能控制寄存器	
0x4000_4410	UART0_ST0	UART0 接收队列 0 状态寄存器	
0x4000_4414	UART0_GT	UART0 帧间隔时间寄存器	
0x4000_4418	UART0_TO	UART0 超时控制寄存器	
0x4000_441C	UART0_TXFR	UART0 发送队列复位寄存器	
0x4000_4420	UART0_RXFR	UART0 接收队列复位寄存器	
0x4000_4424	UART0_DAT1	UART0 接收数据寄存器缓冲队列 1	
0x4000_4428	UART0_IE1	UART0 接收队列 1 中断使能控制寄存器	
0x4000_442C	UART0_ST1	UART0 接收队列 1 状态寄存器	
0x4000_4800	UART1_DAT0	UART1 收发数据寄存器缓冲队列 0	
0x4000_4804	UART1_CTL	UART1 模块控制寄存器	
0x4000_4808	UART1_BR	UART1 波特率控制寄存器	
0x4000_480C	UART1_IE0	UART1 接收队列 0 中断使能控制寄存器	
0x4000_4810	UART1_ST0	UART1 接收队列 0 状态寄存器	
0x4000_4814	UART1_GT	UART1 帧间隔时间寄存器	

0x4000_4818	UART1_TO	UART1 超时控制寄存器	
0x4000_481C	UART1_TXFR	UART1 发送队列复位寄存器	
0x4000_4820	UART1_RXFR	UART1 接收队列复位寄存器	
0x4000_4824	UART1_DAT1	UART1 接收数据寄存器缓冲队列 1	
0x4000_4828	UART1_IE1	UART1 接收队列 1 中断使能控制寄存器	
0x4000_482C	UART1_ST1	UART1 接收队列 1 状态寄存器	
0x4000_4C00	UART2_DAT0	UART2 收发数据寄存器缓冲队列 0	
0x4000_4C04	UART2_CTL	UART2 模块控制寄存器	
0x4000_4C08	UART2_BR	UART2 波特率控制寄存器	
0x4000_4C0C	UART2_IE0	UART2 接收队列 0 中断使能控制寄存器	
0x4000_4C10	UART2_ST0	UART2 接收队列 0 状态寄存器	
0x4000_4C14	UART2_GT	UART2 帧间隔时间寄存器	
0x4000_4C18	UART2_TO	UART2 超时控制寄存器	
0x4000_4C1C	UART2_TXFR	UART2 发送队列复位寄存器	
0x4000_4C20	UART2_RXFR	UART2 接收队列复位寄存器	
0x4000_4C24	UART2_DAT1	UART2 接收数据寄存器缓冲队列 1	
0x4000_4C28	UART2_IE1	UART2 接收队列 1 中断使能控制寄存器	
0x4000_4C2C	UART2_ST1	UART2 接收队列 1 状态寄存器	

13.11 寄存器详解

13.11.1 收发数据 FIFO 缓冲寄存器 UARTn_DATm

位域	类型	复位	名称	描述
[31:9]	--	--	--	--
[8]	W			数据位或奇偶校验位，这取决于控制寄存器中的 MODE 设定
[7]	W			数据位或奇偶校验位，这取决于控制寄存器中的 MODE 设定
[6:0]	W	0	DAT	

Note:

1.[8:7]当 UART 的发送 MODE 选择含有校验位的模式时，校验位不需要用户填入，硬件自动生成并组成数据流。

13.11.2 模块控制寄存器 UARTn_CTL

位域	类型	复位	名称	描述
[31:9]	--	--	--	--
[24:23]	R/W	0x0	IRSPD	**红外发送通信速率** 0x0:600bps 0x1:1200bps 0x2:2400bps 0x3:保留
[22:20]	--	--	--	保留
[19]	R/W	0	IRTXE	红外发送使能

				0：红外发送禁止 1：红外发送使能
[18]	--	--	--	保留
[17]	R/W	0	TXPOL	**TX 发送数据极性控制** 0：标准数据发送 1：标准数据反相发送
[16]	R/W	0	RXPOL	**RX 接收数据极性控制** 0：标准数据接收 1：标准数据反相接收
[15:14]	--	--	--	保留
[13:12]	R/W	0x0	BSIZE	**DMA 一次连续传输字节数** 0x0:一个字节 0x1:4 个字节(传输字节数必须是 4 的整数倍) 0x2:8 个字节(传输字节数必须是 8 的整数倍) 0x3:16 个字节(传输字节数必须是 16 的整数倍)
[11]	R/W	0	DMA	**数据 DMA 传输控制** 0：数据禁止 DMA 传输 1：数据允许 DMA 传输
[10:9]	--	--	--	保留
[8]	R/W	0	RXEN0	**队列 0 数据接收使能** 0：禁止队列 0 数据接收 1：允许队列 0 数据接收
[7]	R/W	0	RUN	**波特率发生器控制** 0：波特率发生器被禁止 1：波特率发生器正常工作
[6]	R/W	0	LPB	**回绕模式控制** 0：正常数据接收和发送模式 1：内部自发自收模(可用于自检)
[5]	R/W	0	PAR	**奇偶校验形式** 0：偶校验 1：奇校验
[4:3]	R/W	0x0	STOPB	**停止位长度** 0x0:0.5 位停止位 0x1:1 位停止位 0x2:1.5 位停止位 0x3:2 位停止位
[2:0]	R/W	0x0	MODE	**工作模式** 0x0:保留 0x1:8 位数据格式 0x2:保留 0x3:7 位数据+1 位奇偶校验 0x4: 0x5: 0x6:

				0x7:8 位数据+1 位奇偶校验

13.11.3 波特率控制寄存器 UARTn_BR

位域	类型	复位	名称	描述
[31:18]	--	--	--	--
[17:16]	R/W	0x0	SR	**串行通讯码元采样率设定** 0x0：标准速率采样，采样频率 f_{SYS}/16 0x1：8 倍速率采样，采样频率 f_{SYS}/8 0x2：4 倍速率采样，采样频率 f_{SYS}/4 0x3：4 倍速率采样，采样频率 f_{SYS}/4
[15:0]	R/W	0x1	BR	波特率分频系数

Note:

设定 UART 通讯的波特率 BPS_{UART}，具体计算公式如下：

$BPS_{UART} = f_{SYS} / (16/(2^{SR})*BR)$ 或 $BR = f_{SYS} / (16/(2^{SR})*BPS_{UART})$

13.11.4 中断控制寄存器 UARTn_IEm

位域	类型	复位	名称	描述
[31:11]	--	--	--	--
[10]	R/W	0	TXENDE	**发送全部完成中断使能** 0：禁止中断 1：允许中断
[9]	R/W	0	RXFE	**接收缓冲队列全满中断使能** 0：禁止中断 1：允许中断
[8]	R/W	0	RXHFE	**接收缓冲队列半满中断** 0：禁止中断 1：允许中断
[7]	R/W	0	TOIDLEE	**空闲超时中断使能** 0：禁止中断 1：允许中断
[6]	R/W	0	TONEE	**接收缓冲队列清空超时中断使能** 0：禁止中断 1：允许中断
[5]	R/W	0	OVERRE	**接收缓冲队列溢出中断使能** 0：禁止中断 1：允许中断
[4]	R/W	0	FERRE	**帧错误中断使能** 0：禁止中断 1：允许中断
[3]	R/W	0	PERRE	**奇偶校验错误中断使能** 0：禁止中断 1：允许中断

[2]	R/W	0	TXHEE	**发送缓冲队列半空中断使能** 0：禁止中断 1：允许中断
[1]	R/W	0	TXEE	**发送缓冲队列全空中断使能** 0：禁止中断 1：允许中断
[0]	R/W	0	RXNEE	**接收缓冲队列非空中断使能** 0：禁止中断 1：允许中断

13.11.5 状态寄存器 UARTn_STm

位域	类型	复位	名称	描述
[31:12]	--	--	--	--
[11]	R	0	TXF	发送队列全满标志位
[10]	R	0	TXENDE	发送全部完成标志位
[9]	R	0	RXFE	接收缓冲队列全满标志位
[8]	R	0	RXHFE	接收缓冲队列半满标志位
[7]	R	0	TOIDLEE	空闲超时标志位
[6]	R	0	TONEE	接收缓冲队列清空超时标志位
[5]	R	0	OVERRE	接收缓冲队列溢出标志位
[4]	R	0	FERR	帧错误标志位
[3]	R	0	PERR	奇偶校验错误标志位
[2]	R	0	TXHE	发送缓冲队列半空标志位
[1]	R	0	TXE	发送缓冲队列全空标志位
[0]	R	0	RXNE	接收缓冲队列非空标志位

13.11.6 帧间隔时间寄存器 UARTn_GT

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:0]	R/W	0	GT	帧间隔时间

Note:

1.寄存器定义了发送两个连续数据帧的间隔时间，以位元的时间宽度为单位.

13.11.7 超时控制寄存器 UARTn_TO

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:0]	R/W	0	TO	数据接收过程的超时判断

Note:

1.寄存器定义了接收两个数据帧的最大的间隔时间

13.11.8 发送队列复位寄存器 UARTn_TXFR

位域	类型	复位	名称	描述
[31:0]	W	--	TXFR	寄存器写入操作，不管写什么值，UARTn_DAT0 发送缓冲队列被复位清空

13.11.9 发送队列复位寄存器 UARTn_RXFR

位域	类型	复位	名称	描述
[31:0]	W	--	RXFR	寄存器写入操作，不管写什么值，UARTn_DAT0 接收缓冲队列被复位清空

14I2C 总线

14.1 I2C 模块综述

I2C是嵌入式系统设计中经常被用到的一种串行通讯总线。它基于SCL（串行时钟）和SDA（串行数据）双线联机，以主从方式实现多个互联设备之间的双向数据通讯。

PT32Z192片内I2C模块支持主模式和从模式通讯方式，其基本特性如下：

- 内含并行数据/串行 I2C 协议转换器
- 支持多个主机并存，提供总线仲裁机制，使总线串行数据不损坏
- 支持 7 位从机寻址模式
- 支持广播呼叫
- 提供数据发送和接收状态标识
- 提供字节传输结束标识
- 通讯错误检测
- 支持标准（100Kbps）或快速（400Kbps）I2C 通讯速率模式

作为 I2C 主机时，其特性包括：

- 产生总线通讯时钟
- 实现总线仲裁
- 7 位地址寻址从机，并控制数据读写方向
- 产生总线通讯起始位、重复起始位和停止位
- 检测通讯错误

作为 I2C 从机时，其特性包括：

- 检测起始位、重复起始位和停止位
- 可编程 I2C7 位地址匹配寻址
- 传输过程中检测起始和停止条件
- 检测通讯错误

启用 I2C 模块前，需要将 SDA、SCL 的 IO 复用功能使能，使 SDA、SCL 端口的 GPIO 功能禁止，IO 口作为 SDA、SCL 使用。具体参考 GPIO 模块端口复用的说明。启用 I2C 模块后，SDA 和 SCL 线被配置成高阻开漏状态，使能弱上拉，上拉电阻为 50K 左右。外部硬件也可以用上拉电阻将其拉至高电平，上拉电阻阻值视工作电压、通讯速度和总线负载而定，一般可选 5~10K。SCL 和 SDA 线是双向的，当 I²C 总线处于自由或空闲状态时，两个引脚都处于高电平状态，适用于多个互连设备所要求的线与功能。

I2C模块在开始工作前，首先必须由软件按芯片的系统时钟频率设定I2C_CTLSET中

CR2/CR1/CR0分频系数，选择合适的SCL通讯时钟频率，详见I2C时钟速度计算和设定；然后配置I2C_CTLSET寄存器EN位使能I2C模块，通讯过程的状态信息则在I2C_STAT状态寄存器中反映。如果作为I2C从机使用，则还需在I2C_ADDR地址寄存器中设定本机的地址码。

在发送数据前，SCL 线先被发送方拉低，软件写入一个字节到 I2C_DATA 数据寄存器后，SDA 和 SCL 开始逐位输出数据和时钟信号进行数据发送；在接收到数据后，SCL 线被接收方持续拉低，直到接收方从 I2C_DATA 数据寄存器中读出数据。

14.2 I2C 协议简述

I2C 总线协议要求所有挂在总线上的设备，其 SDA 和 SCL 线必须是开漏输出，换句话说，任何设备可以输出低电平将 SDA 或 SCL 线拉低，但不能输出高电平驱动。总线上的高电平只能依靠硬件上拉电阻获得。

一般的 I2C 总线通讯包含四部分：

- 起始位发送，表示一次通讯过程开始
- 从机地址发送
- 数据传输
- 停止位发送，表示一次通讯过程结束

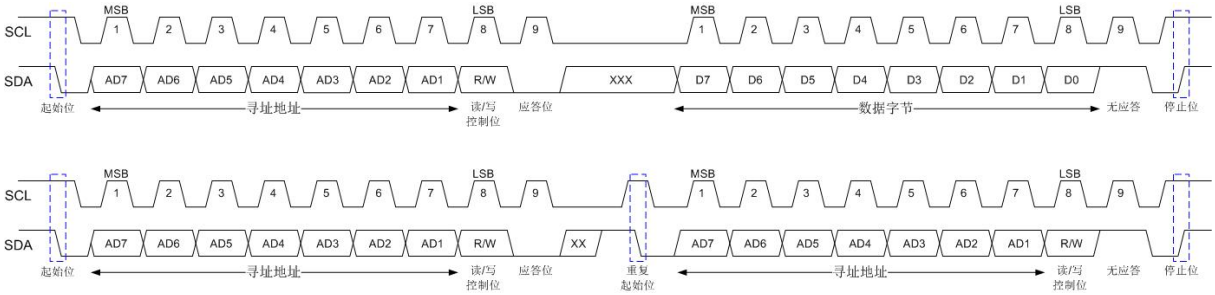


图 14-1 I2C 总线通讯波形示意图

14.2.1 起始位

总线空闲时，没有任何器件占用总线（SCL 和 SDA 线处于逻辑高电平），主机可以通过发送起始位发起通信。如图 15-1 所示，起始位定义为，当 SCL 在高电平时，SDA 从高到低的跳变。该信号表示开始新的数据传输（每次数据传输可能包含几个字节的数据），并使所有从机退出空闲状态。

14.2.2 从机寻址

起始位发出后传输的第一个字节数据是主机发送的从机寻址地址。其中的高 7 位是从机的地址码，最低位 R/W 决定总线上数据传输的方向：

R/W =1 ：读取传输，从机向主机传输数据

R/W =0 ：写入传输，主机向从机传输数据

从机地址可以配置 I2C_ADDR 的 ADDR 栏位，当主机发送的地址与某从机地址匹配时，该从机在第 9 个时钟周期时将 SDA 线拉低（见图 15-1），回送应答位进行响应。

I2C 总线上不能有两个地址相同的从机。如果 I2C 模块是主机，它就不应该发送与其自身从机地址相同的地址。I2C 器件不能同时既是主机又是从机，但是如果在寻址过程中出现仲裁丢失，器件就重新返回到从机模式并正确运行，并可被另一个主机寻址。

14.2.3 数据传输

成功实现从机寻址后，就可以按照主机发送的R/W位指定的方向逐字节地进行数据传输。主机发送地址后的所有传输都被称为数据传输，即使它们包含从机的子地址报文。

每个数据字节的长度均为8位。在传输数据时，在SCL时钟的高电平期间，SDA的数据必须保持稳定。只有当SCL为低时，SDA的数据才可以更改， SCL的一个时钟脉冲传输一个数据位，最高位被首先传输，如图16-1所示。每个数据字节后面都有一个应答位（第9位），该位由从机（发送时）或主机（读取时）回送信号，通过在第9个时钟周期时把SDA 线拉低来实现。这样，一个完整数据字节的传输需要9个时钟脉冲。如果对方在第9个时钟周期时无应答（NACK），则其必须保留SDA线在高电平。主机发生数据，从机返回无应答（NACK），主机将接收到的无应答信号解释为不成功的数据传输；如果主机在接收一个数据字节传输后未应答，返回一个无应答（NACK）给从机，从机将其理解为数据传输结束，并释放总线给主机，因此，为了在收到最后一个字节后产生一个NACK脉冲，在读倒数第二个数据字节之后必须清除ACK位。对于以上两种情况，数据传输都被中止，主机会进行以下两种操作之一：

-发送停止位，终止数据传输，并释放总线

-发送重复起始位，开始新呼叫

14.2.4 停止位

主机可以通过发送停止位终止通信，以释放总线。然而，主机可以直接发送起始位和呼叫命令，而无需首先发送停止位，这被称为重复启动。停止位的定义是SCL在逻辑1位置时的从低到高的SDA跳变（见图15-1）。即便从机发出一个应答，主机也可以发送停止位，此时从机必须释放总线。

14.2.5 重复起始位

如图15-1所示，重复起始位是在无需首先生成停止位以终止通信的情况下，再发送一个开始（START）的信号。该信号由主机用来与另外一个从机进行通信，或者在不同模式（发送/接收模式）中与同一从机进行通信，而不需要释放总线。

14.2.6 总线仲裁

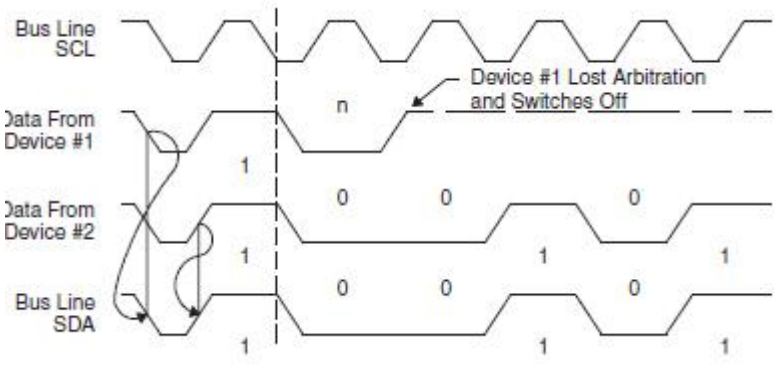


图 14-2 I²C 两个主机仲裁过程示意图

只有在I2C总线处于空闲或者自由的状态时，主机才能开始传输数据。I2C总线是真正的多主控总线，允许一个以上的主机连接到I2C总线上。如果两个甚至多个主机试图同时控制总线，仲裁机制赋予发送二进制低位(逻辑低)串行数据的设备更高的优先级。竞争主机的相对优先级由数据仲裁过程确定，如果一个总线主机发出逻辑1，而另一个发出逻辑0，那

么前者就丢失仲裁（见图15-2）。失败的一方立即切换到从机接收模式，停止驱动SDA输出。在这种情况下，从主模式到从模式的转换不会生成停止条件。与此同时，丢失仲裁的设备状态位I2C_STAT=0x38，表示仲裁丢失。

14.2.7 时钟同步

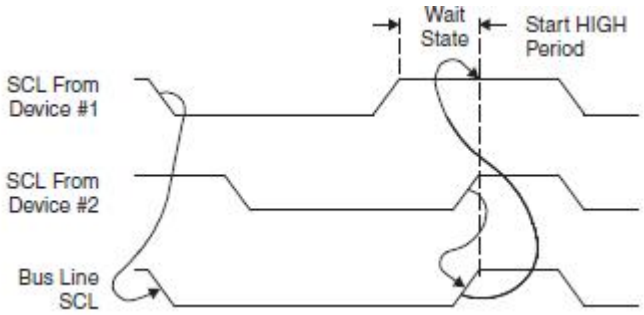


图 14-3 I²C 总线仲裁过程中 SCL 波形示意图

在仲裁过程中，不同主机发送的时钟需要被同步。由于SCL线上的线与逻辑，所以SCL上的电平跳变受连接到总线上的所有设备影响。图15-3显示了时钟同步的仲裁过程中，SCL线上的电平变化。当主机的时钟进入低电平周期后，它将一直保持SCL线的低电平，直到时钟到达高电平。然而，如果另一个主机时钟依旧是低电平，这时主机时钟从低到高的变化就不会改变SCL线的状态。因此，经过同步的时钟SCL拉低的时间由具有最长的低电平周期的主机决定。低电平周期较短的主机在该时段进入高电平等待状态（wait-state）（见图16-3）。当所有主机均结束低电平周期时，同步时钟SCL线才被释放和拉高。这时主机时钟和SCL线的状态一致，所有主机开始计数它们的高电平周期。第一个结束高电平周期的主机再次拉低SCL线。

所以，经过同步的时钟SCL的低电平周期由所有主机中具有最长的低电平周期的主机决定，高电平周期由所有主机中具有最短高电平周期的主机决定。

14.2.8 通讯握手

时钟同步机制可以用作数据传输中的通讯握手。在完成一个字节的传输（9个位）后，从机可以保持SCL低位。在这种情况下，从机会暂停总线时钟，强迫主机时钟进入等待状态，直到从机释放SCL线。

14.2.9 时钟延展

时钟同步机制可以被从机用于减缓传输的比特速率。在主机已经拉低SCL后，从机可以继续拉低SCL一定时间，然后再释放它。如果从机SCL低电平周期长于主机SCL低电平周期，那么就会将SCL总线信号低电平周期延展。

14.2.10 广播呼叫寻址

广播呼叫寻址功能可以用来寻址所有连接到 I²C 总线的设备。在寻址上，主机可通过向地址寄存器 I2C_ADDR 中的 ADDR 写入 0x00 启动广播呼叫寻址功能。

从机设备可通过设置相应的使能 I2C_ADDR 中的从机广播寻址控制位 GC 为 1，以使其支持广播呼叫寻址功能。如果 GC 位已经置 1 以支持广播呼叫寻址功能，I2C_CTLSET 寄存器中的 AA 位也应置 1，当设备接收到值为 00H 的地址时将返回一个确认信号(ACK)。当此条件发生时，从机状态寄存器更新，I2C_STAT=0x0E。

14.3 I2C 主模式

I2C 模块在初始化后默认处于从模式状态。在总线处于空闲状态下（I2C_STAT=0xF8），软件设定 I2C_CTLCLR_STA=1，模块即进入主模式并在总线上产生一个起始位。当 I2C_STAT 更新状态为 0x08，表示产生一个起始位完毕，并在中断允许时产生中断请求。此时主机将 SCL 线持续拉低，直到软件往 I2C_DATA 数据寄存器内写入首个字节（从机地址和传输方向控制位）进行发送。

14.3.1 I2C 主模式寻址方式

I2C 主模式下在起始位后的首个发送字节一定是从机的地址码，支持 7 位寻址模式。

7 位寻址:

- 主机发送地址码字节，其中 7 位地址加最低位读写控制 R/W，决定主设备是发送器模式或接收器模式
- 查询等待或中断响应字节发送完毕
- 若从机地址匹配，并应答，则状态位 I2C_STAT=0x18；若无应答，则 I2C_STAT=0x20

地址发送完毕且从机正确应答后，主机将 SCL 线持续拉低，下一步主机将进入数据发送或数据接收过程。

14.3.2 I2C 主模式数据发送

主机发送完从机的寻址地址（其中的读写控制位为0）后，主机为发送器模式，可以继续往I2C_DATA寄存器内写入数据进行发送，在数据被写入I2C_DATA寄存器前主机将SCL线持续拉低。主机每次发送完一个字节且收到从机的应答，则状态位寄存器I2C_STAT=0x28；若从机无应答，则状态位寄存器I2C_STAT=0x30，模块可以发送停止位终止I2C通讯。当最后一个字节发送完毕后，主机发送一个停止位，模块随即转为从模式。

14.3.3 I2C 主模式数据接收

主机发送完从机的寻址地址（其中的读写控制位为1）后，主机即开始从从机处读取数据。每读取一个字节后主机视ACK位的设定自动产生应答（I2C_CTLSET_AA=1时）或非应答（I2C_CTLSET_AA=0时），主机软件可查询I2C_STAT状态或中断响应，此时主机持续将SCL线拉低，直到主机软件把接收的数据从I2C_DATA寄存器中读走。在读取最新收到的数据前，如果主机确定下一个字节将是最后一个需读取的字节时，应将ACK位清0（I2C_CTLSET_AA=0），确保在读取下一个字节时主机会向从机回送非应答位；读取最后一个字节后，发送一个停止位，模块随即转为从模式。

14.3.4 I2C 主模式错误信息

I2C模块在主模式下如果检测到总线通讯发生错误，I2C_STAT状态将被置为0x00，同时模块将释放对总线的控制。

在一个字节传输过程在，当I2C侦测到总线产生一个停止位或起始位时，认为通讯发生错误。当检测到停止位导致的总线发生错误时，本机会释放总线，状态位I2C_STAT=0x00，产生中断标志位（I2C_CTLSET_SI=1），清除此标志位之后总线恢复空闲状态，I2C_STAT=0xF8；当检测到起始位导致的总线发生错误时，本机状态位I2C_STAT=0x00，

产生中断标志位（I2C_CTLSET_SI=1），清除此标志位之后，主机需要发送停止位，使总线恢复空闲状态，I2C_STAT=0xF8。

14.4 I2C 从模式

I2C模块在初始化后默认处于从模式状态，等待总线上产生起始位，随后接收主机发出的寻址地址码并和本机设定地址进行比对匹配。

14.4.1 I2C 从模式地址匹配

I2C从模式地址寻址为7位地址:

- 从机接收的地址码字节高7位和地址寄存器I2C_ADDR [7:1]设定的本机地址作匹配比对，接收的地址码字节最低位为读写控制位，不参与地址比对。
- 主机发送7位地址与本机地址比对匹配成功，从机回送应答位；否则回送非应答位，从机等待下一个起始位。
- 地址匹配后从机将SCL线持续拉低，直到从机软件清除中断标志位后将其释放（I2C_CTLCLR_SI写1，清除I2C_CTLSET_SI中断标志位）

14.4.2 I2C 从模式数据接收

在地址匹配流程后（其中的读写控制位为 0）从机即进入数据接收器模式。从机每接收到一个字节后即回送一个应答信息（应答或非应答，取决于 I2C_CTLSET_ACK 位的设定），状态位寄存器 I2C_STAT 更新状态（详见表 16-2），并在中断允许时产生中断请求。此时从机将 SCL 线持续拉低，直到从机软件从 I2C_DR 寄存器内将数据读出。

14.4.3 I2C 从模式数据发送

在地址匹配流程后（其中的读写控制位为 1）从机即进入数据发送状态，将 SCL 线持续拉低，从机软件往 I2C_DATA 寄存器内写入一个字节后 SCL 线被释放，随即开始发送。在收到对方的应答信息后更新 I2C_STAT 状态（详见表 16-2），中断允许时产生中断请求。

14.4.4 I2C 从模式通讯终止

在最后一个字节传输完成后如果主机发出停止位，从机也随即终止本次通讯过程，并在中断允许时产生中断请求。

14.4.5 I2C 从模式错误信息

I2C 模块在从模式下如果检测到总线通讯发生错误，I2C_STAT 状态将被置为 0x00，同时模块将释放对总线的控制。

在一个字节传输过程在，当 I2C 侦测到总线产生一个停止位或起始位时，认为通讯发生错误。当检测到停止位导致的总线发生错误时，本机会释放总线，状态位 I2C_STAT=0x00，产生中断标志位（I2C_CTLSET_SI=1），清除此标志位之后总线恢复空闲状态，I2C_STAT=0xF8；当检测到起始位导致的总线发生错误时，本机状态位 I2C_STAT=0x00，产生中断标志位（I2C_CTLSET_SI=1），清除此标志位之后，主机需要发送停止位，使总线恢复空闲状态，I2C_STAT=0xF8。

14.5 I2C 时钟速度计算和设定

I2C 通讯的时钟速度取决于系统时钟频率和分频系数。分频系数通过 I2C_CTLSET 寄存器中 CR2/CR1/CR0 三位选择，用户要修改这三位，需要先对这三位的相关位清除（对 I2C_CTLCLR 中 CR2/CR1/CR0 置 1），再对 I2C_CTLSET 中 CR2/CR1/CR0 写值，具体计算方法如下：

$f_{SCL} = f_{SYS} / DIV$

其中：

- f_{SCL} 为期望的 I2C 通讯 SCL 时钟频率
- f_{SYS} 为系统时钟频率
- DIV 为系统时钟分频系数，按表 15-1 选择

应用软件在已知系统时钟频率的前提下，选择合适的分频系数来设定 I2C 通讯的时钟频率。I2C 最高时钟频率不应超过 400Kbps

CR2	CR1	CR0	DIV
0	0	0	256
0	0	1	224
0	1	0	192
0	1	1	160
1	0	0	960
1	0	1	120
1	1	0	60
1	1	1	保留

表 14-1 产生 I²C 时钟频率时系统时钟分频系数

14.6 I2C 状态信息和中断响应

I2C 模块有多个事件可产生同一个中断标识（I2C_CTLSET_SI）。各事件由 I2C_STAT 寄存器中的位[7:3]状态信息表述，见表 15-2 说明。

I2C_STAT[7:3]	I²C 总线状态	I2C_CTLSET相关位信息				I²C模块的下一步应对
		STA	STO	SI	AA	
00000	主模式或从模式被寻址下I²C通讯错误	0	1	0	X	I²C总线释放
00001	主模式发送起始位(START) 完成	X	0	0	X	发送从机地址+写控制位，接收ACK
00010	主模式发送重复起始位 (Repeated START) 完成	X	0	0	X	同发送起始位（00001） 发送从机地址+读控制位，主机转入接收模式
00011	主模式发送从机地址+写控制位完成，收到ACK响应	0	0	0	X	发送数据字节并接收ACK
		1	0	0	X	发送重复起始位
		0	1	0	X	发送停止位，STO标志置位
		1	1	0	X	停止位后紧接着发送起始位，STO标志清零
00100	主模式发送从机地址+写控制位完成，无ACK响应	0	0	0	X	发送数据字节并接收ACK
		1	0	0	X	发送重复起始位
		0	1	0	X	发送停止位，STO标志置位
		1	1	0	X	停止位后紧接着发送起始位，STO标志清零
00101	主模式发送数据字节完成，收到ACK响应	0	0	0	X	发送数据字节并接收ACK
		1	0	0	X	发送重复起始位
		0	1	0	X	发送停止位，STO标志置位
		1	1	0	X	停止位后紧接着发送起始位，STO标志清零
00110	主模式发送数据字节完成，无ACK响应	0	0	0	X	发送数据字节并接收ACK
		0	0	0	X	发送重复起始位
		1	0	0	X	发送停止位，STO标志置位
		1	0	0	X	停止位后紧接着发送起始位，STO标志清零
00111	主模式在发送地址或数据时总线仲裁丢失	0	0	0	X	I²C总线被释放
		1	0	0	X	I²C总线空闲时发送起始位
01000	主模式发送从机地址+读控制位完成，收到ACK响应	0	0	0	0	接收数据，无ACK响应
		1	0	0	1	接收数据，有ACK响应
01001	从机地址+读控制位发送完成，无ACK响应	1	0	0	X	发送重复起始位
		0	1	0	X	发送停止位，STO标志置位
		1	1	0	X	停止位后紧接着发送起始位，STO标志清零
01010	数据字节接收完成，回送ACK响应	0	0	0	0	接收数据，无ACK响应
		0	0	0	1	接收数据，有ACK响应
01011	数据字节接收完成，回送NACK响应	1	0	0	X	发送重复起始位
		0	1	0	X	发送停止位，STO标志置位
		1	1	0	X	停止位后紧接着发送起始位，STO标志清零
01100	从模式地址+写控制位接收完	X	0	0	0	接收数据，回应NACK

	成，回送ACK响应	X	0	0	1	接收数据，回应ACK
01101	主模式下总线仲裁丢失，作为从模式收到地址+写控制位，并回送ACK响应	X	0	0	0	接收数据，回应NACK
		X	0	0	1	接收数据，回应ACK
01110	收到广播寻址（0x00），并回送ACK响应	X	0	0	0	接收数据，回应NACK
		X	0	0	1	接收数据，回应ACK
01111	主模式下总线仲裁丢失，作为从模式收到广播寻址，并回送ACK响应	X	0	0	0	接收数据，回应NACK
		X	0	0	1	接收数据，回应ACK
10000	从模式被寻址下收到一个数据字节，并回送ACK响应	X	0	0	0	接收数据，回应NACK
		X	0	0	1	接收数据，回应ACK
0001	从模式被寻址下收到一个数据字节，并回送NACK响应	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
		0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
		1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
		1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
10010	广播被寻址下收到一个数据字节，并回送ACK响应	X	0	0	0	接收数据，回应NACK
		X	0	0	1	接收数据，回应ACK
10011	广播被寻址下收到一个数据字节，并回送NACK响应	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
		0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
		1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
		1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
10100	从模式被寻址下收到停止位（STOP）或重复起始位（Repeated START）	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
		0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
		1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
		1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
10101	从模式地址+读控制位接收完成，回送ACK响应	X	0	0	0	发送最后一个字节，接收ACK应答
		X	0	0	1	发送一个字节，接收ACK应答
10110	主模式下总线仲裁丢失，作为从模式收到地址+读控制位，并回送ACK响应	X	0	0	0	发送最后一个字节，接收ACK应答
		X	0	0	1	发送一个字节，接收ACK应答

10111	从模式发送数据字节完成，收到ACK响应	X X	0 0	0 0	0 1	发送最后一个字节，接收ACK应答 发送一个字节，接收ACK应答
11000	从模式发送数据字节完成，无ACK响应	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
		0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
		1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
		1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
11001	从模式发送最后一个数据字节完成，收到ACK响应	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
		0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
		1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
		1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
11111	总线空闲	0	0	0	0	-

表 14-2 I²C 通讯状态信息

当除了总线空闲状态外的任一状态发生时，模块置 I2C_CTLSET_SI 位为 1，应用软件可以设定 I2C 模块的系统中断使能位（NVIC_ISER_I2C=1），响应中断请求并进入中断服务程序。

14.7 寄存器列表

地址	寄存器	描述	备注
0x4000_5400	I2C_CTLSET	控制寄存器	
0x4000_5404	I2C_STAT	I2C状态寄存器	
0x4000_5408	I2C_DATA	I2C数据寄存器	
0x4000_540C	I2C_ADDR	I2C地址寄存器	
0x4000_5418	I2C_CTLCLR	I2C控制清除寄存器	

14.8 寄存器描述

14.8.1 控制寄存器 I2C_CTLSET

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7]	R/W		CR[2]	系统时钟分频系数[2]
[6]	R/W		EN	使能控制
[5]	R/W		STA	发送超始位 该位置 1 后，模块检测总线处于空闲状态时，发送一个起始位，起

				始位结束后，该位被硬件自动清 0
[4]	R/W	0	STO	发送停止位 主模式下对此位写 1 后，总线上发出停止位．在停止位完成后，硬件自动将 STO 清 0
[3]	R/W	0	SI	**中断标志位** 0：I2C 模块有中断 1：I2C 模块无中断
[2]	R/W	0	AA	**应答位控制** 0：不发送应答位 1：地址字节或数据字节后发送应答位
[1:0]	R/W	0	CR[1]	系统时钟分频系数[1]
[0]	R/W	0	CR[0]	系统时钟分频系数[0]

Note:

1. CR[2:0]配置见表 15-1 说明
2. 全部位只能置 1，清零需要使用寄存器 I2C_CTLCLR

14.8.2 状态寄存器 I2C_STAT

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:3]	R	0x1F	STAT	I2C 总线状态(Note1)
[2:0]	--	--	--	--

Note:

1.状态说明见表 15-2 说明

14.8.3 收发数据寄存器 I2C_DATA

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:0]	R/W	0x0	DATA	收发数据寄存器

Note:

1. 发送时，软件写入 8 位数据后随即启动发送流程
2. 接收时，收到的第一个字节为地址，软件必须从寄存器读取刚接收的数据，才能接收其它数据字节

14.8.4 从机地址寄存器 I2C_ADDR

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:1]	R/W	0x0	ADDR	匹配从机地址寄存器
[0]	R/W	0x0	GC	广播寻址使能 0：禁止从机广播寻址 1：使能从机广播寻址

14.8.5 控制清除寄存器 I2C_CTLCLR

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7]	W		CR[2]	系统时钟分频系数[2]清零
[6]	W		EN	使能控制清零
[5]	W		STA	发送超始位清零
[4]	W	0	STO	发送停止位清零
[3]	W	0	SI	中断标志位清零
[2]	W	0	AA	应答位控制清零
[1]	W	0	CR[1]	系统时钟分频系数[1]清零
[0]	W	0	CR[0]	系统时钟分频系数[0]清零

15 SPI 总线

15.1 概述

SPI 是一种同步串行外设接口，对数据进行串并转换，通过主从方式的方式实现 MCU 和外围器件的数据通讯。

SPI 通讯是全双工的，发送或接收数据都是高位在先，在数据发送的同时进行数据接收。按通讯时钟的不同提供方式，SPI 通讯分主机和从机两种。SPI 主机的时钟由本地产生，通讯的发起和结束完全由自己主动控制；SPI 从机的时钟为外部输入（来自 SPI 主机），被动响应主机的通讯。

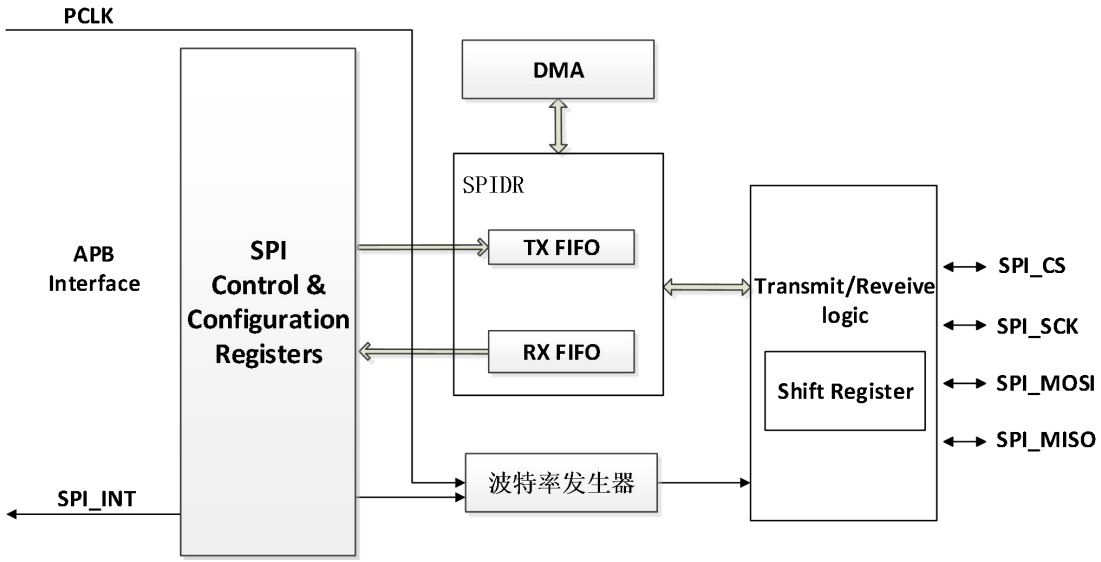


图 15-1. SPI 方框图

PT32Z192 的 SPI 模块支持主/从模式工作．基本特性如下：

- 一帧数据传输 4~16 位可编程
- 主机最高传输速率为系统时钟（PCLK）二分频
- 最大 8 级接收 FIFO 缓冲队列
- 最大 8 级发送 FIFO 缓冲队列
- 主机模式下可编程传输速率

15.2 SPI 通讯信号

SPI 通讯所用引脚描述如下：

CS：

SPI 串行通讯从机片选控制。每一个从机都有一个独立的片选信号控制。

SCK：

SPI 串行通讯时钟，时钟信号只有 SPI 主机才能发出，所有从机的时钟只能输入来自主机的时钟信号。

MISO：

SPI 串行通讯数据输入。该数据为从机发出，在主机侧为输入。

MOSI：

SPI 串行通讯数据输出。该数据为主机发出，在从机侧为输入。

15.3 SPI 工作模式

PT32Z192 的 SPI 模块支持三种模式，Motorola SPI 格式，TI 同步串行格式以及 National Microwire 模式

15.3.1 TI 同步串行模式

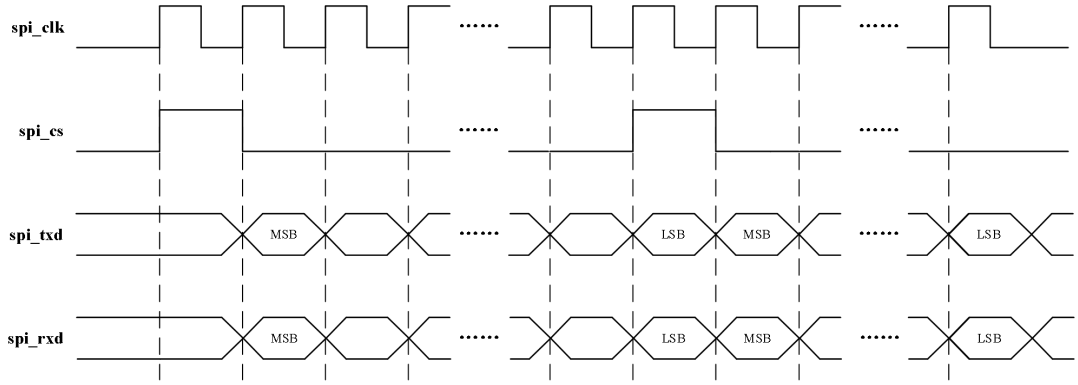


图 15-2 TI 同步串行模式数据传输时序图

当 SPI 处于 TI 同步串行模式时，SPI 处于全双工的工作状态，在空闲状态下 SPI 时钟和片选信号 CS 处于低电平，一旦发送 FIFO 内被写入了数据，则 CS 立即拉高一个 SPI 时钟周期，之后数据开始串行传输，数据帧长短可通过 SPIn_CR0 寄存器的 DSS 位配置。

数据传输时，高位先发，数据在 SPI 时钟上升沿打出，在下降沿保持稳定。

15.3.2 National Microwire 模式

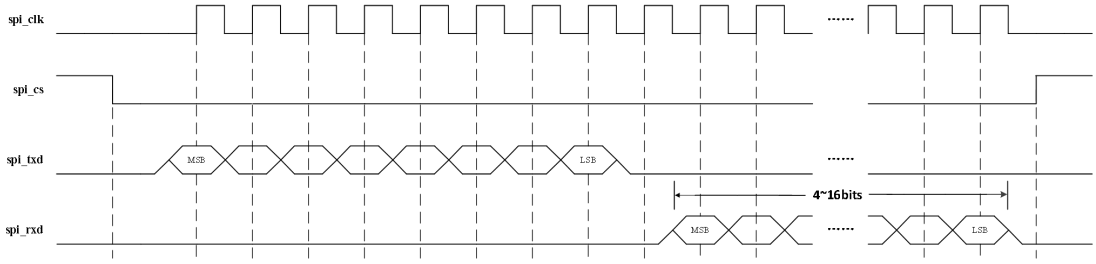


图 15-3 National Microwire 模式单帧数据传输时序图

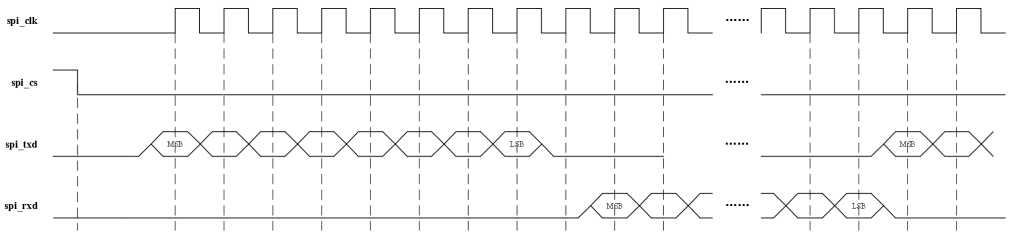


图 15-4 National Microwire 模式连续传输时序图

当 SPI 处于 National Microwire 模式时，SPI 处于半双工的工作状态，在空闲状态下 SPI 时钟处于低电平，片选信号 CS 处于高电平，一旦发送 FIFO 内被写入了数据，则主机拉低 CS 信号，之后数据开始串行传输，先固定发送 8bit 控制数据，之后隔一个 SPI 时钟

周期后开始接收从机发送的数据，接收的数据帧长短可通过 SPIn_CR0 寄存器的 DSS 位配置。

数据传输时，高位先发，数据在 SPI 时钟下降沿打出，在上升沿保持稳定。

15.3.3 Motorola 模式

当 SPI 处于 Motorola 模式时，SPI 处于全双工的工作状态，并且根据 SPIn_CR0 寄存器的 SCLKP 位的设定，分为 4 种不同的工作模式，分别为模式 00、模式 01、模式 10 和模式 11。

15.3.3.1 模式 00

当设为模式 00 时，在空闲状态下 SPI 时钟保持低电平。主机把从机的片选信号 CS 拉低后，从机发往主机的第一位数据即出现在主机的 MISO 引脚上就绪；主机往 SPI 数据发送寄存器内写入数据后，发往从机的第一位数据也即刻出现在 MOSI 引脚上，之后在每一个时钟上升沿主机和从机各自同步采样接收对方发送过来的数据位并送入内部移位寄存器，在其后的时钟下降沿各自打出下一位数据。当一个 SPI 数据帧（4~16 位）传输结束后，接收到的数据从内部移位寄存器转存入 SPI 接收 FIFO，SPI 时钟恢复到低电平状态。主机可继续往发送寄存器内写入数据进行连续发送，或者通过软件把从机片选信号拉高，结束 SPI 通讯。

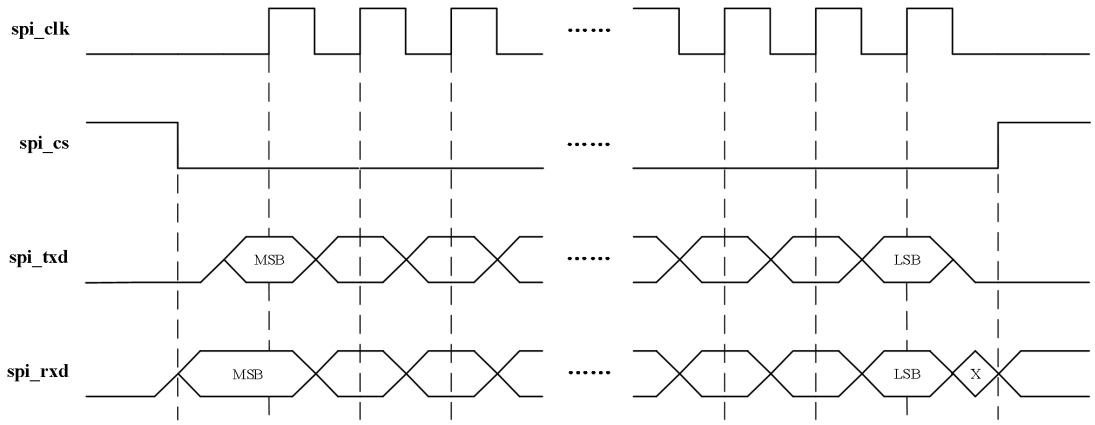


图 15-5 模式 00 时序图（SCLKP=00）

图 15-6 显示了此格式下连续数据传输的时序图。注意，在每个数据帧之间 CS 信号必须转变为无效电平。

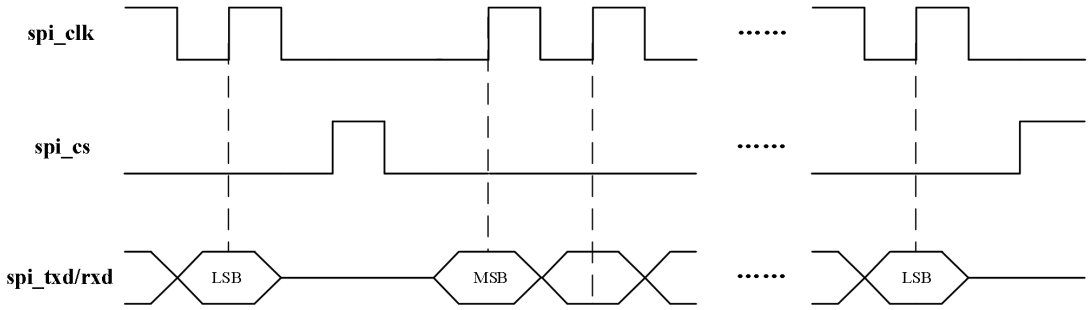


图 15-6 模式 00 连续数据传输时序图（SCLKP=00）

15.3.3.2 模式 01

当设为模式 01 时，在空闲状态下 SPI 时钟保持高电平。主机把从机的片选信号 CS 拉低后，从机发往主机的第一位数据即出现在主机的 MISO 引脚上就绪；主机往 SPI 数据发送寄存器内写入数据后，发往从机的第一位数据也即刻出现在 MOSI 引脚上，之后在每一个时钟下降沿主机和从机各自同步采样接收对方发送过来的数据位并送入内部移位寄存器，在其后的时钟上升沿各自打出下一位数据。当一个 SPI 数据帧（4~16 位）传输结束后，接收到的数据从内部移位寄存器转存入 SPI 接收 FIFO，SPI 时钟恢复到低电平状态。主机可继续往发送寄存器内写入数据进行连续发送，或者通过软件把从机片选信号拉高，结束 SPI 通讯。

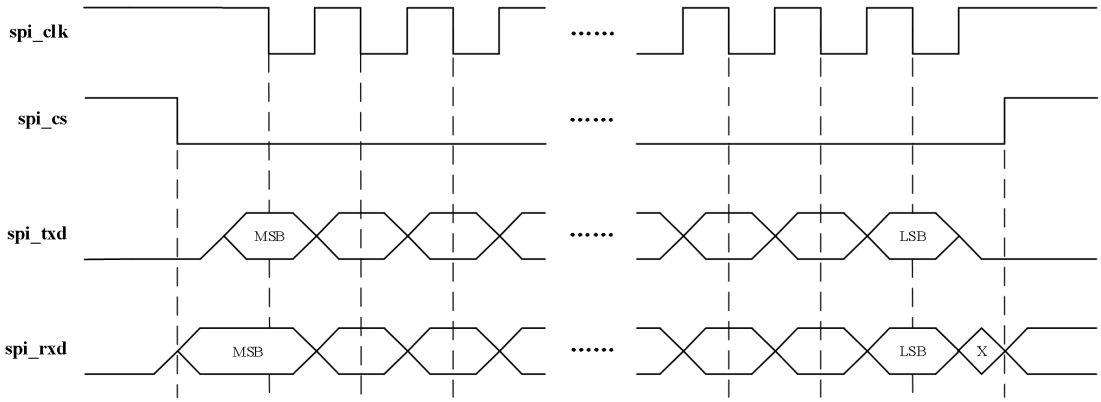


图 15-7 模式 01 时序图（SCLKP=01）

图 15-8 示了此格式下连续数据传输的时序图。注意，在每个数据帧之间 CS 信号必须转变为无效电平。

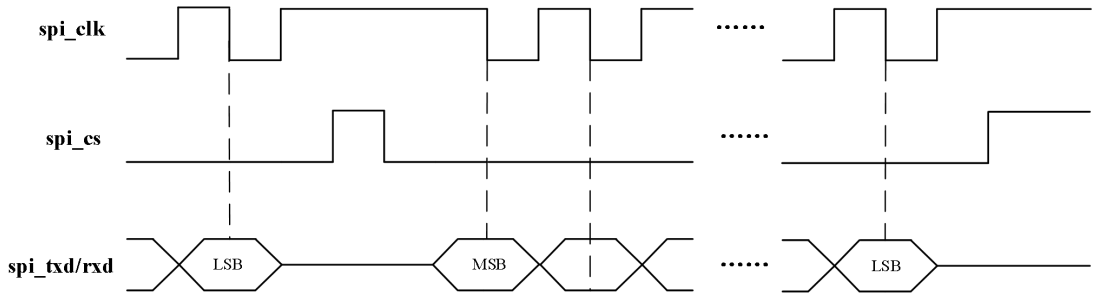


图 15-8 模式 01 连续数据传输时序图（SCLKP=01）

15.3.3.3 模式 10

当设为模式 10 时，在空闲状态下 SPI 时钟保持低电平。主机把从机的片选信号 CS 拉低后，从机发往主机的第一位数据即出现在主机的 MISO 引脚上就绪；主机往 SPI 数据发送寄存器内写入数据后，发往从机的第一位数据也即刻出现在 MOSI 引脚上，之后在每一个时钟下降沿主机和从机各自同步采样接收对方发送过来的数据位并送入内部移位寄存器，在其后的时钟上升沿各自打出下一位数据。当一个 SPI 数据帧（4~16 位）传输结束后，接收到的数据从内部移位寄存器转存入 SPI 接收 FIFO，SPI 时钟恢复到低电平状态。主机可继续往发送寄存器内写入数据进行连续发送，或者通过软件把从机片选信号拉高，结束 SPI 通讯。

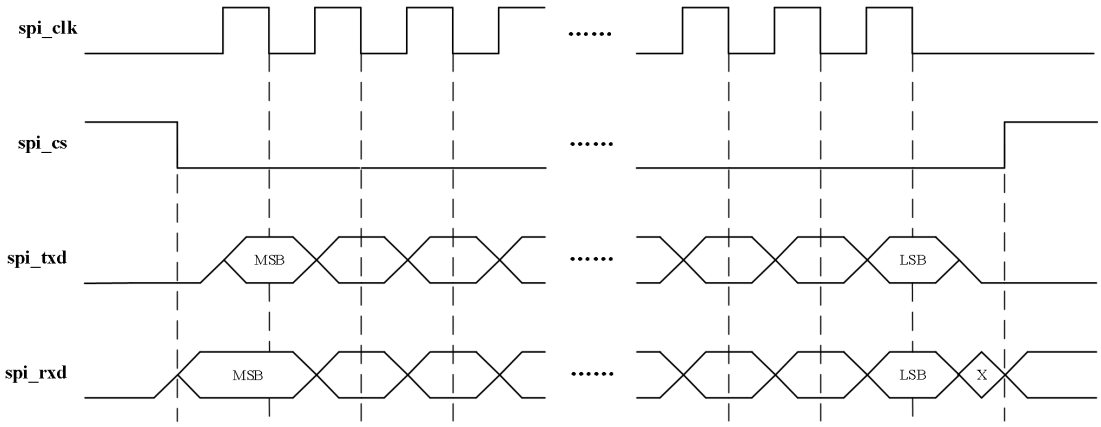


图 15-9：模式 10 时序图（SCLKP=10）

图 15-10 显示了模式 10 连续数据传输时序图。注意，CS 信号必须保持在有效电平直到最后一个数据传输结束。

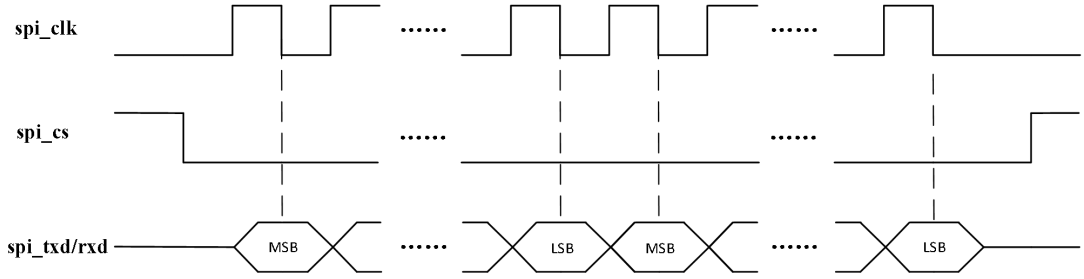


图 15-10：模式 10 连续数据传输时序图（SCLKP=10）

15.3.3.4 模式 11

当设为模式 10 时，在空闲状态下 SPI 时钟保持高电平。主机把从机的片选信号 CS 拉低后，从机发往主机的第一位数据即出现在主机的 MISO 引脚上就绪；主机往 SPI 数据发送寄存器内写入数据后，发往从机的第一位数据也即刻出现在 MOSI 引脚上，之后在每一个时钟上升沿主机和从机各自同步采样接收对方发送过来的数据位并送入内部移位寄存器，在其后的时钟下降沿各自打出下一位数据。当一个 SPI 数据帧（4~16 位）传输结束后，接收到的数据从内部移位寄存器转存入 SPI 接收 FIFO，SPI 时钟恢复到低电平状态。主机可继续往发送寄存器内写入数据进行连续发送，或者通过软件把从机片选信号拉高，结束 SPI 通讯。

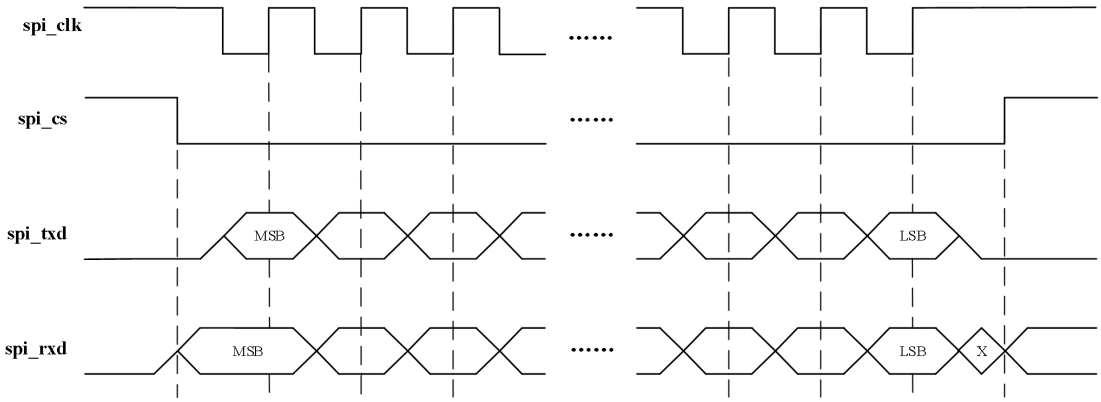


图 15-11：模式 11 时序图（SCLKP=11）

图 15-12 显示了模式 11 连续数据传输时序图。注意，CS 信号必须保持在有效电平直到最后一个数据传输结束。

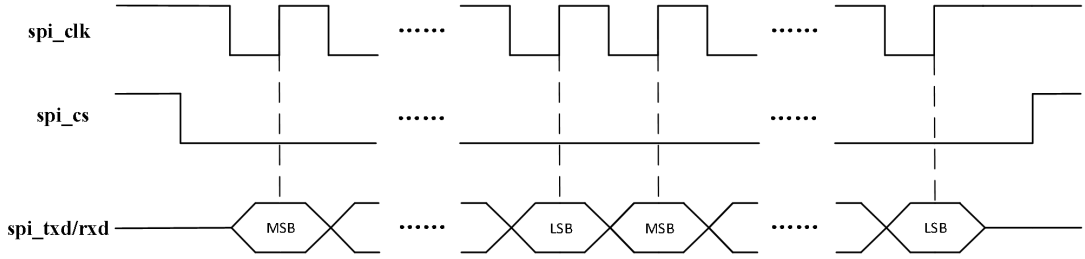


图 15-12：模式 11 连续数据传输时序图（SCLKP=11）

15.4 SPI 通讯波特率计算和设定

当 SPI 被设置为主机模式（SPIn_CR1 寄存器的 MS 位为 0）时，SPI 通讯时钟由硬件主动产生并发送至从机，其通讯速率计算公式如下：

$f_{SPI} = f_{SYS} / (CPSDVSR * (1+SCR))$

其中：

- f_{SPI} 为期望的 SPI 通讯速率，同时用于接收和发送
- f_{SYS} 为 PCLK 时钟频率
- CPSDVSR 为 PCLK 预分频寄存器的设定值，取值范围为 2~254 间的偶数（其第 0 位必须为 0），软件可通过 SPIn 时钟预分频寄存器 SPIn_CPSR 的 CPSDVSR 位配置所需预分频系数
- SCR 为经过预分频后的时钟的分频系数，取值范围为 0~255，分别对应于 1~256 分频，软件可通过 SPIn_CR0 控制寄存器 1 的 SCR 位配置所需分频系数，

当芯片被设置为 SPI 从机（SPIn_CR1 寄存器的 MS 位为 1）时，SPI 通讯时钟为外部输入。为能保证从机模式下的数据接收，必须满足芯片系统时钟（PCLK）频率大于等于 SPI 数据时钟频率的 12 倍。

15.5 SPI 中断

PT32Z192 的 SPI 模块的中断源一共有四个，分别对应数据发送 FIFO 和数据接收 FIFO 的不同状态：

- **接收数据 FIFO 溢出**

当软件没有及时读取接收到的数据，导致接收 FIFO 中已经接收到 8 个数据处于 FIFO 为满的状态时，这时又收到第 9 个数据时，接收 FIFO 就会发生溢出，最早收到的数据被顶出队列而丢弃，接收 FIFO 溢出标志位（OV）被置 1

- **接收 FIFO 非空并且软件读取数据超时**

当接收 FIFO 非空（FIFO 中有一个或多个数据）但软件没有在一定时间内（32 个 SPI 通讯时钟）进行任何的读取数据操作，接收 FIFO 非空超时标志位（TO）被置 1

- **接收 FIFO 数据过半**

当接收 FIFO 中的数据过半（数据个数≥4）时，接收 FIFO 数据半满或过半标志位（RX）被置 1

- **发送 FIFO 数据过半**

当发送 FIFO 中的数据过半（数据个数≥4）时，发送 FIFO 数据半满或过半标志位（TX）被置 1

上述 4 个状态标志位软件可通过原始中断标志寄存器 SPIn_RIS 查看。软件可通过配置中断使能控制寄存器 SPIn_IE 决定上述 4 个状态标志位中哪些中断源被使能从而发出中断请求，经 SPIn_IE 使能控制后的中断标志反映在使能中断标志寄存器 SPIn_MIS 中。

被使能后的中断标志通过或的逻辑，最终向 CPU 发出同一个中断请求，软件可通过查询使能中断标志寄存器 SPIn_MIS 寄存器查询具体的中断源并做出相应的处理。

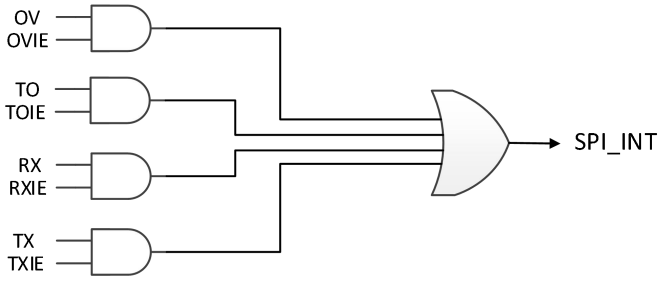


图 15-13 SPI 中断映射图

15.6 寄存器列表

地址	寄存器	描述	备注
0x4000_3800	SPI0_CR0	SPI0 控制寄存器 0	SPI0_CR0 说明
0x4000_3804	SPI0_CR1	SPI0 控制寄存器 1	SPI0_CR1 说明
0x4000_3808	SPI0_DR	SPI0 数据寄存器	SPI0_DR 说明
0x4000_380C	SPI0_SR	SPI0 状态寄存器	SPI0_SR 说明
0x4000_3810	SPI0_CPSR	SPI0 时钟预分频寄存器	SPI0_CPSR 说明
0x4000_3814	SPI0_IE	SPI0 中断使能寄存器	SPI0_IE 说明
0x4000_3818	SPI0_RIS	SPI0 原始中断标志寄存器	SPI0_RIS 说明
0x4000_381C	SPI0_MIS	SPI0 使能中断标志寄存器	SPI0_MIS 说明
0x4000_3820	SPI0_ICR	SPI0 中断标志清除寄存器	SPI0_ICR 说明
0x4000_3828	SPI0_CSCR	SPI0 片选信号控制寄存器	SPI0_CSCR 说明
0x4000_3C00	SPI1_CR0	SPI1 控制寄存器 0	SPI1_CR0 说明
0x4000_3C04	SPI1_CR1	SPI1 控制寄存器 1	SPI1_CR1 说明
0x4000_3C08	SPI1_DR	SPI1 数据寄存器	SPI1_DR 说明
0x4000_3C0C	SPI1_SR	SPI1 状态寄存器	SPI1_SR 说明
0x4000_3C10	SPI1_CPSR	SPI1 时钟预分频寄存器	SPI1_CPSR 说明
0x4000_3C14	SPI1_IE	SPI1 中断使能寄存器	SPI1_IE 说明
0x4000_3C18	SPI1_RIS	SPI1 原始中断标志寄存器	SPI1_RIS 说明
0x4000_3C1C	SPI1_MIS	SPI1 使能中断标志寄存器	SPI1_MIS 说明
0x4000_3C20	SPI1_ICR	SPI1 中断标志清除寄存器	SPI1_ICR 说明
0x4000_3C28	SPI1_CSCR	SPI1 片选信号控制寄存器	SPI1_CSCR 说明

15.7 寄存器描述

15.7.1 SPIn 控制寄存器 0 SPIn_CR0

（地址： SPI0：0x4000_3800；SPI1：0x4000_3C00）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	SCR								SCLKP		FRF		DSS			
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[3:0] DSS：　　帧数据长度选择**
  - 0000b：　保留
  - 0001b：　保留
  - 0010b：　保留
  - 0011b：　数据帧长度为 4 位
  - 0100b：　数据帧长度为 5 位
  - 0101b：　数据帧长度为 6 位
  - 0110b：　数据帧长度为 7 位
  - 0111b：　数据帧长度为 8 位
  - 1000b：　数据帧长度为 9 位
  - 1001b：　数据帧长度为 10 位
  - 1010b：　数据帧长度为 11 位
  - 1011b：　数据帧长度为 12 位
  - 1100b：　数据帧长度为 13 位
  - 1101b：　数据帧长度为 14 位
  - 1110b：　数据帧长度为 15 位
  - 1111b：　数据帧长度为 16 位
- **位[5:4] FRF：　　帧格式选择**
  - 00b：　Motorola SPI 格式
  - 01b：　TI 同步串行格式
  - 10b：　Microwire 格式
  - 11b：　保留
- **位[7:6] SCLKP：　　SPI 时钟极性相位控制**
  - 00b：　空闲状态下时钟保持低电平，时钟上升沿采样数据，时钟下降沿出数据
  - 01b：　空闲状态下时钟保持高电平，时钟下降沿采样数据，时钟上升沿出数据
  - 10b：　空闲状态下时钟保持低电平，时钟下降沿采样数据，时钟上升沿出数据
  - 11b：　空闲状态下时钟保持高电平，时钟上升沿采样数据，时钟下降沿出数据
- **位[15:8] SCR：　　SPI 时钟分频系数**

  详见 163 SPI 通讯波特率计算和设定章节介绍

- **位[31:16] ：** **保留**

15.7.2 SPIn 控制寄存器 1 SPIn_CR1

（地址： SPI0：0x4000_3804；SPI1：0x4000_3C04）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	MS	EN	LBM
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] LBM：** **回绕模式控制**
  0b： 正常工作模式
  1b： 回绕传输模式（自发自收）
- **位[1] EN：** **SPI 使能控制**
  00b： SPI 使能
  01b： SPI 禁止
- **位[2] MS：** **主从模式选择**
  0b： 主机模式
  1b： 从机模式
- **位[31:3] ：** **保留**

15.7.3 SPIn 数据寄存器 SPIn_DR

（地址： SPI0：0x4000_3808；SPI1：0x4000_3C08）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	DATA															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[15:0] DATA：** **传输数据**
  数据的格式均为右对齐，若数据长度小于 16 位，向发送 FIFO 写入数据时则需保持右对齐格式，否则无效的高位数据将被忽略，不被发送；接收 FIFO 的数据格式自动为右对齐格式
  写： 将数据写入发送 FIFO
  读： 从接收 FIFO 读出数据
- **位[31:16] ：** **保留**

15.7.4 SPIn 状态寄存器 SPIn_SR

（地址： SPI0：0x4000_380C；SPI1：0x4000_3C0C）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	-	-	BSY	REF	RNE	TNF	TFE
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] TFE：** **发送 FIFO 全空标志位**
  0b： 发送 FIFO 非空
  1b： 发送 FIFO 为空
- **位[1] TNF：** **发送 FIFO 未满标志位**
  0b： 发送 FIFO 全满
  1b： 发送 FIFO 未满

- **位[2] RNE：** **接收 FIFO 非空标志位**
  0b： 接收 FIFO 为空

1b： 接收 FIFO 非空

- **位[3] RNE： 接收 FIFO 全满标志位**

0b： 接收 FIFO 未满

1b： 接收 FIFO 全满

- **位[4] BSY： 数据传输状态标志位**

0b： SPI 处于空闲状态

1b： SPI 处于传输数据状态或者发送 FIFO 非空

- **位[31:5] ： 保留**

15.7.5 SPIn 时钟预分频寄存器 SPIn_CPSR

（地址： SPI0：0x4000_3810；SPI1：0x4000_3C10）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	CPSDVSR							
R/W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[7:0] CPSDVSR： SPI 时钟预分频系数**

CPSDVSR 必须是 2~254 之间的偶数（bit0 只读，恒为 0）

具体使用详见 18.3 章节描述

- **位[31:8] ： 保留**

15.7.6 SPIn 中断使能寄存器 SPIn_IE

（地址： SPI0：0x4000_3814；SPI1：0x4000_3C14）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	-	-	-	TXIE	RXIE	TOIE	OVIE
R/W	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] OVIE：** **接收 FIFO 溢出中断使能控制**
  - 0b： 禁止中断
  - 1b： 允许中断
- **位[1] TOIE：** **接收 FIFO 非空并超时中断使能控制**
  - 0b： 禁止中断
  - 1b： 允许中断
- **位[2] RXIE：** **接收 FIFO 数据半满或过半中断使能控制**
  - 0b： 禁止中断
  - 1b： 允许中断
- **位[3] TXIE：** **发送 FIFO 数据半满或过半中断使能控制**
  - 0b： 禁止中断
  - 1b： 允许中断
- **位[31:4] ：** **保留**

15.7.7 SPIn 原始中断标志寄存器 SPIn_RIS

（地址： SPI0：0x4000_3818；SPI1：0x4000_3C18）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	-	-	-	TX	RX	TO	OV
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] OV：** **接收 FIFO 溢出标志位**
  0b： 接收 FIFO 未发生溢出
  1b： 接收 FIFO 溢出
- **位[1] TO：** **接收 FIFO 非空并超时标志位**
  0b： 接收 FIFO 为空
  1b： 接收 FIFO 非空并超时
- **位[2] RX：** **接收 FIFO 数据半满或过半标志位**
  0b： 接收 FIFO 数据未过半
  1b： 接收 FIFO 数据已过半
- **位[3] TX：** **发送 FIFO 数据半满或过半标志位**
  0b： 发送 FIFO 数据未过半
  1b： 发送 FIFO 数据已过半
- **位[31:4] ：** **保留**

15.7.8 SPIn 使能中断标志寄存器 SPIn_MIS

（地址： SPI0：0x4000_381C；SPI1：0x4000_3C1C）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	-	-	-	TX	RX	TO	OV
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] OV：** **接收 FIFO 溢出中断标志位**
  0b： 接收 FIFO 未发生溢出或中断未使能
  1b： 接收 FIFO 溢出并产生中断
- **位[1] TO：** **接收 FIFO 非空并超时中断标志位**
  0b： 接收 FIFO 为空或中断未使能
  1b： 接收 FIFO 非空并超时并产生中断
- **位[2] RX：** **接收 FIFO 数据半满或过半中断标志位**
  0b： 接收 FIFO 数据未过半或中断未使能
  1b： 接收 FIFO 数据已过半并产生中断
- **位[3] TX：** **发送 FIFO 数据半满或过半中断标志位**
  0b： 发送 FIFO 数据未过半或中断未使能
  1b： 发送 FIFO 数据已过半并产生中断
- **位[31:4] ：** **保留**

15.7.9 SPIn 中断标志清除寄存器 SPIn_ICR

（地址： SPI0：0x4000_3820；SPI1：0x4000_3C20）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TOCLR	OVCLR
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] OVCLR：** **接收 FIFO 溢出中断清除**
  0b： 无作用
  1b： 清除接收 FIFO 溢出中断
- **位[1] TOCLR：** **接收 FIFO 非空并超时中断标志位**
  0b： 无作用
  1b： 清除接收 FIFO 非空并超时中断
- **位[31:2] ：** **保留**

15.7.10 SPIn 片选信号控制寄存器 SPIn_CSCR

（地址： SPI0：0x4000_3824；SPI1：0x4000_3C24）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	-	-	-	SWCS	SWSEL	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[1:0] ：** **保留**
- **位[2] SWSEL：** **片选信号控制方式选择**
  0b： SPI 模块硬件自动控制
  1b： 软件通过配置 SWCS 位控制
- **位[3] SWCS：** **软件片选信号控制**
  0b： 片选信号为 0
  1b： 片选信号为 1

16 定时器

16.1 系统定时器 SYSTICK

系统定时器是芯片自带的一个模块，它主要用于计时。系统定时器提供了一个简单易用的 24 位循环递减的计数器，当系统定时器使能时，计数器开始工作。当计数器递减到 0 时，会向矢量中断控制器发起中断请求，申请获得处理器响应并处理系统定时器的事务。

16.1.1 寄存器列表

地址	寄存器	描述	备注
0xE000_E010	CSR	控制寄存器	
0xE000_E014	RVR	重载寄存器	
0xE000_E018	CVR	当前值寄存器	
0xE000_E01C	CALIB	校准寄存器	

16.1.2 寄存器描述

16.1.2.1 控制寄存器 CSR

位域	类型	复位	名称	描述
[31:17]	--	--	--	--
[16]	RC/WC		ZF	计数归零标志(Note1)
[15:3]	--	--	--	--
[2]	R/W	1	CLKS	计数器时钟选择 0:外部参考时钟 1:系统时钟
[1]	R/W		CVR	中断使能 0:计数归零不产生中断 1:计数归零产生中断
[0]	R/W	0	EN	计数使能

Note:

1. 读或写之后，ZF 都会清零

16.1.2.2 重载寄存 RVR

位域	类型	复位	名称	描述
[31:24]	--	--	--	--
[23:0]	R/W		RELOAD	计数重载值(Note1)

Note:

1. RELOAD 的值在 0x0~0xFF_FFFF 之间任意值
2. 如果设置为 0，计时器停止工作
3. 产生中断的间隔为 RELOAD+1 个周期

16.1.2.3 当前值寄存器 CVR

位域	类型	复位	名称	描述
[31:24]	--	--	--	--
[23:0]	R		CURRENT	计数当前值(Note1)

16.1.2.4 校准寄存器 CALIB

位域	类型	复位	名称	描述
[31]	R	0	NOREF	独立参考时钟
[30:0]	R		TENMS	10ms 标定值

16.2 捕获定时器 TMR1

捕获定时器 TMR1 为 16 位宽通用定时器，递增计数模式。配有三路外部引脚信号的沿跳变的时间捕捉和四路比较定时的功能，可利用同一个定时器资源，灵活实现外部信号变化时间点的精确捕捉和特定的软件定时控制。

定时器 TMR1 可产生两种中断信号，比较定时中断和捕捉中断，但所有中断信号将使用同一个信号输出端口，应用软件需通过 TMR1_IR 寄存器查询标志位以判断产生中断的具体事件。

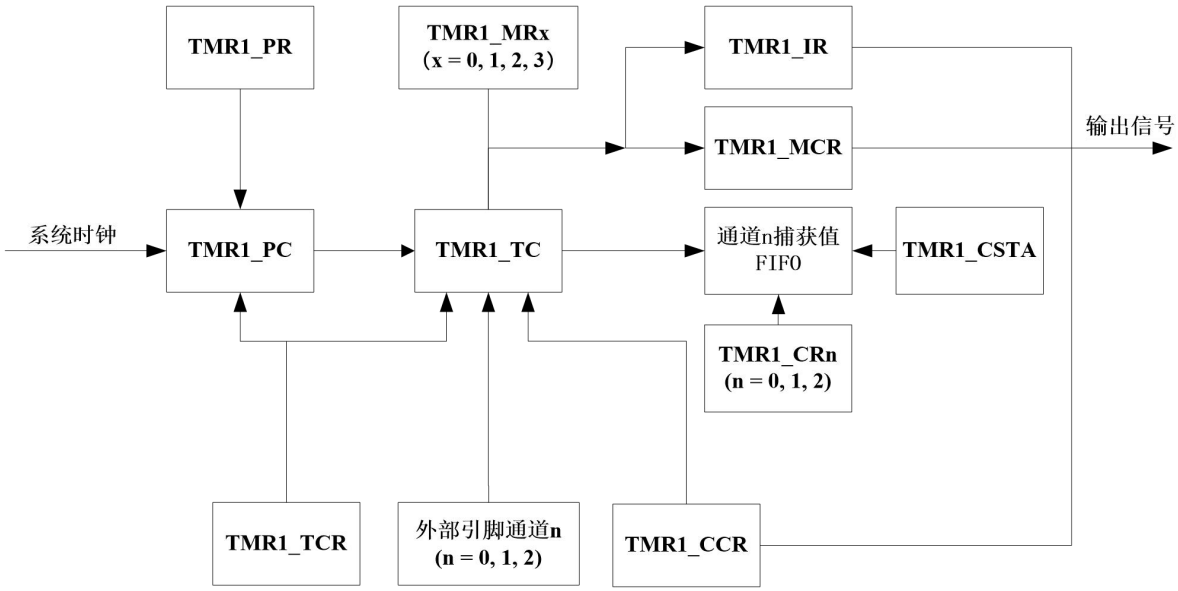


图 16-1：捕获定时器结构图

16.2.1 TMR1 捕获

TMR1 配有三路捕捉通道。当任意一个外部引脚通道信号出现 TMR1_CCR 寄存器设定的电平沿（上升沿或下降沿）跳变时，TMR1_IR 寄存器相关标志位会被更新（CRn_R/F 置 1）。同时，出现该电平沿跳变时所对应的 TMR1_TC 寄存器的值会被保存。

若 TMR1_TCR 寄存器的 FIFO_En 为 0，TMR1_TC 寄存器的值会被存入对应通道的 TMR1_CRn 寄存器，同时根据 TMR1_CCR 寄存器内的参数设置产生输出信号（输出捕捉中断信号）。

若 TMR1_TCR 寄存器的 FIFO_En 为 1，TMR1_TC 寄存器的值会被存入对应通道的捕获值 FIFO，并更新 TMR1_CSTA 寄存器的相关标志位（CnN 加一或者 CAPnOV 置 1），

同时根据 TMR1_CCR 寄存器内的参数设置产生输出信号（输出捕捉中断信号）。软件可通过 TMR1_CRn（n=0,1,2）寄存器读取通道 n（n=0,1,2）的捕获值 FIFO 的数据。

每个捕捉通道的捕获值 FIFO 的深度为 8 级。TMR1_CSTA 寄存器用于描述三路捕捉通道的捕获值 FIFO 的状态。当 TMR1 不使能时，三路捕捉通道的捕获值 FIFO 均为空。

当捕捉通道 n（n=0,1,2）每发生一次符合 TMR1_CCR 寄存器设定的电平沿（上升沿或下降沿）跳变时，TMR1_CSTA 寄存器内对应的 CnN（n=0,1,2）会加一。当软件通过 TMR1_CRn（n=0,1,2）寄存器读走捕获通道 n（n=0,1,2）的捕获值 FIFO 内的一个数据后，TMR1_CSTA 内对应的 CnN（n=0，1，2）会减一。

当捕获值 FIFO 内已经填入 8 个捕获值（FIFO 为满但并未溢出）后，若此时捕获通道 n（n=0,1,2）又发生了一次符合 TMR1_CCR 寄存器设定的电平沿（上升沿或下降沿）跳变，捕捉通道 n（n=0,1,2）的捕获值 FIFO 将产生溢出标志（TMR1_CSTA 寄存器的 CAPnOV 置 1），出现该电平沿跳变时所对应的 TMR1_TC 寄存器的值不会存入捕捉通道 n 的捕获值 FIFO，TMR1_CSTA 寄存器的 CnN 不会加一，TMR1_IR 寄存器的相关标志位会被更新（CRn_R/F 置 1）。

当捕获值 FIFO 溢出（TMR1_CSTA 寄存器的 CAPnOV 为 1），若此时捕获通道 n（n=0,1,2）发生符合 TMR1_CCR 寄存器设定的电平沿（上升沿或下降沿）跳变，出现该电平沿跳变时所对应的 TMR1_TC 寄存器的值不会存入捕捉通道 n 的捕获值 FIFO，TMR1_CSTA 寄存器的 CnN 不会加一，只有 TMR1_IR 寄存器的相关标志位会被更新（CRn_R/F 置 1）。

捕获值 FIFO 的溢出标志位需通过清空 FIFO 来清除。软件可通过以下方法清空捕获值 FIFO：

1. 清零 TMR1_TCR 寄存器的 CEN 位，不使能 TMR1
2. 通过 TMR1_CRn 寄存器读取捕捉通道 n 的捕获值 FIFO 内的数据，直到 FIFO 为空

注：当捕获值 FIFO 为空（TMR1_CSTA 寄存器的 CnN 为 0）时，读取 FIFO 内的数据不可预料，CnN 的值不变，仍旧为 0

16.2.2 寄存器列表

地址	寄存器	描述	备注
0x4000_1400	IR	中断寄存器	
0x4000_1404	TCR	控制寄存器	
0x4000_1408	TC	计数值寄存器	
0x4000_140C	PR	预分频寄存器	
0x4000_1410	PC	预分频值寄存器	
0x4000_1414	MCR	匹配控制寄存器	
0x4000_1418	MR0	匹配 0 寄存器	
0x4000_141C	MR1	匹配 1 寄存器	
0x4000_1420	MR2	匹配 2 寄存器	
0x4000_1424	MR3	匹配 3 寄存器	
0x4000_1428	CCR	捕获控制寄存器	
0x4000_142C	CR0	通道 0 捕捉值寄存器	
0x4000_1430	CR1	通道 1 捕捉值寄存器	
0x4000_1434	CR2	通道 2 捕捉值寄存器	
0x4000_1438	CSTA	捕获缓冲区状态寄存器	

16.2.3 寄存器描述

16.2.3.1 中断寄存器 IR

位域	类型	复位	名称	描述
[31:10]	--	--	--	--
[9]	R/WC	0	CRR2	捕获 2 上升沿中断
[8]	R/WC	0	CRR1	捕获 1 上升沿中断
[7]	R/WC	0	CRR0	捕获 0 上升沿中断
[6]	R/WC	0	CRF2	捕获 2 下降沿中断
[5]	R/WC	0	CRF1	捕获 1 下降沿中断
[4]	R/WC	0	CRF0	捕获 0 下降沿中断
[3]	R/WC	0	MR3	匹配 3 中断
[2]	R/WC	0	MR2	匹配 2 中断
[1]	R/WC	0	MR1	匹配 1 中断
[0]	R/WC	0	MR0	匹配 0 中断

Note:

1. 全部中断不需要使能
2. 全部中断写 1 清零

16.2.3.2 控制寄存器 TCR

位域	类型	复位	名称	描述
[31:3]	--	--	--	--
[2]	R/W	0	FIFO_EN	FIFO 工作使能
[1]	R/W	0	CRST	软件复位定时器
[0]	R/W	0	CEN	定时器使能

16.2.3.3 当前值寄存器 TC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0	TC	定时器当前值

16.2.3.4 时钟预分频寄存器 PR

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:0]	R/W	0	PR	时钟预分频(Note1)

Note:

1. $f_{TMR}=f_{SYS}/PR$

16.2.3.5 分频计数当前值寄存器 PC

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:0]	R/W	0	PR	时钟分频计数当前值

16.2.3.6 匹配控制寄存器 MCR

位域	类型	复位	名称	描述
[31:12]	--	--	--	--
[11]	R/W	0	MR3S	TC 和 MR3 匹配计数器停止使能
[10]	R/W	0	MR3R	TC 和 MR3 匹配产生计数器复位使能
[9]	R/W	0	MR3I	TC 和 MR3 匹配产生中断使能
[8]	R/W	0	MR2S	TC 和 MR2 匹配计数器停止使能
[7]	R/W	0	MR2R	TC 和 MR2 匹配产生计数器复位使能
[6]	R/W	0	MR2I	TC 和 MR2 匹配产生中断使能
[5]	R/W	0	MR1S	TC 和 MR1 匹配计数器停止使能
[4]	R/W	0	MR1R	TC 和 MR1 匹配产生计数器复位使能
[3]	R/W	0	MR1I	TC 和 MR1 匹配产生中断使能
[2]	R/W	0	MR0S	TC 和 MR0 匹配计数器停止使能
[1]	R/W	0	MR0R	TC 和 MR0 匹配产生计数器复位使能
[0]	R/W	0	MR0I	TC 和 MR0 匹配产生中断使能

16.2.3.7 匹配值寄存器 MRn

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0	MRn	定时器匹配值配置

Note:

1. n 可表示为 0/1/2/3

16.2.3.8 捕获控制寄存器 CCR

位域	类型	复位	名称	描述
[31:9]	--	--	--	--
[8]	R/W	0	CAP2I	通道 2 边沿捕获中断使能
[7]	R/W	0	CAP2FE	通道 2 下降沿捕获使能
[6]	R/W	0	CAP2RE	通道 2 上升沿捕获使能
[5]	R/W	0	CAP1I	通道 1 边沿捕获中断使能
[4]	R/W	0	CAP1FE	通道 1 下降沿捕获使能
[3]	R/W	0	CAP1RE	通道 1 上升沿捕获使能
[2]	R/W	0	CAP0I	通道 0 边沿捕获中断使能
[1]	R/W	0	CAP0FE	通道 0 下降沿捕获使能
[0]	R/W	0	CAP0RE	通道 0 上升沿捕获使能

16.2.3.9 捕获控制寄存器 CRn

位域	类型	复位	名称	描述
[31:17]	--	--	--	--
[16]	R/W	0	CAPPR	通道 n 边沿触发标志 0：当前捕获为下降沿触发 1：当前捕获为上升沿触发
[15:0]	R	0	CAPn	通道 n 电平化时间值

16.2.3.10 捕获状态寄存器 CSTA

位域	类型	复位	名称	描述
[31:17]	--	--	--	--
[23]	R	0	CAP2OV	通道 2 捕获值 FIFO 溢出状态
[22:19]	--	--	--	--
[18:16]	R/W	0	C2N	通道 2 捕获值 FIFO 数据个数
[15]	R	0	CAP1OV	通道 1 捕获值 FIFO 溢出状态
[14:12]	--	--	--	--
[11:8]	R/W	0	C1N	通道 1 捕获值 FIFO 数据个数
[7]	R	0	CAP0OV	通道 0 捕获值 FIFO 溢出状态
[6:4]	--	--	--	--
[3:0]	R/W	0	C0N	通道 0 捕获值 FIFO 数据个数

16.3 通用定时器 TMR2~TMR5

定时器为 16 位宽通用定时器，使用递增计数模式。

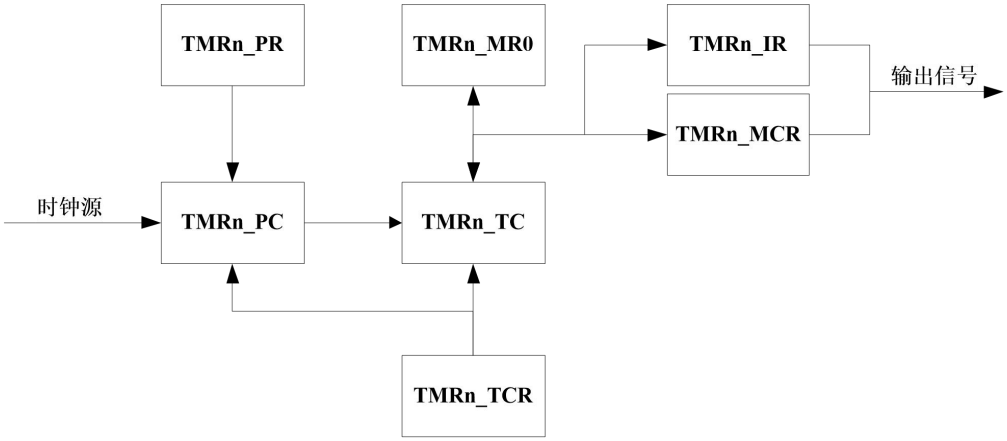


图 16-2 通用定时器结构图

TMRn_PC 寄存器根据 TMRn_PR 寄存器内设置的分频系数对时钟源进行预分频，TMRn_TC 寄存器在 TMRn_PC 寄存器产生的时钟下为定时器 TMRn 计数，当 TMRn_TC 寄存器与 TMRn_MR0 寄存器内的值相同时，TMRn_IR 寄存器会产生比较定时中断信号（TMRn_IR 寄存器的 MR0 置 1），根据 TMRn_MCR 寄存器内的参数设置，定时器 TMRn 产生相应的输出信号（输出比较定时中断信号，复位定时器或停止定时器）。

TMRn_TCR 用于使能和复位定时器 TMRn。

TMR2，TMR3，TMR4，TMR5 都是通用定时器

16.3.1 寄存器列表

地址	寄存器	描述	备注
0x4000_0000	TMR2_IR	TMR2 中断寄存器	
0x4000_0004	TMR2_TCR	TMR2 控制寄存器	
0x4000_0008	TMR2_TC	TMR2 计数值寄存器	
0x4000_000C	TMR2_PR	TMR2 预分频系数寄存器	
0x4000_0010	TMR2_PC	TMR2 预分频计数值寄存器	
0x4000_0014	TMR2_MCR	TMR2 匹配控制寄存器	
0x4000_0018	TMR2_MR0	TMR2 匹配值寄存器	
0x4000_0400	TMR3_IR	TMR3 中断寄存器	
0x4000_0404	TMR3_TCR	TMR3 控制寄存器	
0x4000_0408	TMR3_TC	TMR3 计数值寄存器	
0x4000_040C	TMR3_PR	TMR3 预分频系数寄存器	
0x4000_0410	TMR3_PC	TMR3 预分频计数值寄存器	
0x4000_0414	TMR3_MCR	TMR3 匹配控制寄存器	
0x4000_0418	TMR3_MR0	TMR3 匹配值寄存器	
0x4000_0800	TMR4_IR	TMR4 中断寄存器	
0x4000_0804	TMR4_TCR	TMR4 控制寄存器	
0x4000_0808	TMR4_TC	TMR4 计数值寄存器	
0x4000_080C	TMR4_PR	TMR4 预分频系数寄存器	
0x4000_0810	TMR4_PC	TMR4 预分频计数值寄存器	
0x4000_0814	TMR4_MCR	TMR4 匹配控制寄存器	
0x4000_0818	TMR4_MR0	TMR4 匹配值寄存器	
0x4000_0C00	TMR5_IR	TMR5 中断寄存器	
0x4000_0C04	TMR5_TCR	TMR5 控制寄存器	
0x4000_0C08	TMR5_TC	TMR5 计数值寄存器	
0x4000_0C0C	TMR5_PR	TMR5 预分频系数寄存器	
0x4000_0C10	TMR5_PC	TMR5 预分频计数值寄存器	
0x4000_0C14	TMR5_MCR	TMR5 匹配控制寄存器	
0x4000_0C18	TMR5_MR0	TMR5 匹配值寄存器	

16.3.2 寄存器描述

16.3.2.1 中断寄存器 IR

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R/WC	0	MR0	匹配 0 中断

Note:

1. 写 1 清零

16.3.2.2 控制寄存器 TCR

位域	类型	复位	名称	描述
[31:3]	--	--	--	--
--	--	--	--	--
[1]	R/W	0	CRST	软件复位定时器
[0]	R/W	0	CEN	定时器使能

16.3.2.3 当前值寄存器 TC

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0	TC	定时器当前值

16.3.2.4 时钟预分频寄存器 PR

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:0]	R/W	0	PR	时钟预分频(Note1)

Note:

1. $f_{TMR}=f_{SYS}/PR$

16.3.2.5 分频计数当前值寄存器 PC

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7:0]	R/W	0	PR	时钟分频计数当前值

16.3.2.6 匹配控制寄存器 MCR

位域	类型	复位	名称	描述
[31:12]	--	--	--	--
[11]	R/W	0	MR3S	TC 和 MR3 匹配计数器停止使能
[10]	R/W	0	MR3R	TC 和 MR3 匹配产生计数器复位使能
[9]	R/W	0	MR3I	TC 和 MR3 匹配产生中断使能
[8]	R/W	0	MR2S	TC 和 MR2 匹配计数器停止使能
[7]	R/W	0	MR2R	TC 和 MR2 匹配产生计数器复位使能
[6]	R/W	0	MR2I	TC 和 MR2 匹配产生中断使能
[5]	R/W	0	MR1S	TC 和 MR1 匹配计数器停止使能
[4]	R/W	0	MR1R	TC 和 MR1 匹配产生计数器复位使能
[3]	R/W	0	MR1I	TC 和 MR1 匹配产生中断使能
[2]	R/W	0	MR0S	TC 和 MR0 匹配计数器停止使能
[1]	R/W	0	MR0R	TC 和 MR0 匹配产生计数器复位使能
[0]	R/W	0	MR0I	TC 和 MR0 匹配产生中断使能

17 看门狗

17.1 功能

一个 32 位宽的递减计数器

17.1.1 启动 WDT

上电复位后WDT模块默认处于禁止状态．应用软件必须设置WDTCTL_INTE位启动WDT开始工作．同时，该位的置1也意味着系统将WDT计数器归零

17.1.2 WDT 中断

WDT计数器启动后，递减计数值第一次归零时，将触发WDT中断响应。一旦触发了WDT中断，在又经过一个WDT定时时间后系统即被复位．应用软件可以编写自己的中断服务程序来应对此WDT中断事件．如果发生WDT中断，一般意味着应用软件已有过长的间隔时间没有执行清除看门狗工作，可用于软件调试阶段的可靠性诊断．正式发布的应用程序不能在自己的中断服务程序中执行清除看门狗指令，否则看门狗将永远不会引发系统复位，失去了其系统监控的作用

17.1.3 WDT 复位

在WDT计数器一次归零后，WDT重新载入初值，继续递减计数．当计数值第二次归零，且WDTCTL_RSTE位设为1时，将立即引发一次系统硬件复位，即看门狗复位

17.1.4 WDT 定时时间设定

计数重载寄存器WDTLOAD的设定值直接决定了WDT的定时时间．按应用所期望设定的看门狗定时时间，计算WDTLOAD重载值的公式为：

$$WDTLOAD = T_{WDT} * f_{CLK} - 1$$

17.1.5 软件清除 WDT

应用软件在正常运行时，必须在设定的 WDT 定时时间范围内对 WDTCLR 寄存器写一次任意数，以重载复位 WDT 计数器．正常情况下不应出现 WDT 中断，更不能出现 WDT 复位

17.2 寄存器列表

地址	寄存器	描述	备注
0x4000_2C00	WDTLOAD	WDT 计数重载寄存器	
0x4000_2C04	WDTVALUE	WDT 计数值寄存器	
0x4000_2C08	WDTCONTROL	WDT 控制寄存器	
0x4000_2C0C	WDTINTCLR	WDT 中断清除寄存器	
0x4000_2C10	WDTRIS	WDT 原始中断标志寄存器	
0x4000_2C14	WDTMIS	WDT 掩蔽中断标志寄存器	
0x4000_2FFC	WDTLOCK	WDT 锁定寄存器	

17.3 寄存器描述

17.3.1 重载寄存器 WDTLOAD

位域	类型	复位	名称	描述
[31:0]	R/W	0	WDTLOAD	计数重载

17.3.2 当前值寄存器 WDTVALUE

位域	类型	复位	名称	描述
[31:0]	R/W	0	WDTVALUE	工作时，计数器当前值

17.3.3 控制寄存器 WDTCONTROL

位域	类型	复位	名称	描述
[31:2]	--	--	--	--
[1]	R/W	0	RESEN	使能复位输出
[0]	R/W	0	INTEN	使能中断

17.3.4 中断清除寄存器 WDTINTCLR

位域	类型	复位	名称	描述
[31:0]	W	0	INTCLR	--

Note:

1.对此寄存器写任意值，清除中断

17.3.5 中断标志寄存器 WDTRIS

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	RIF	中断标志位

17.3.6 中断寄存器 WDTMIS

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	IF	中断

17.3.7 锁定寄存器 WDTLOCK

位域	类型	复位	名称	描述
[31:1]	W	0	--	--
[0]	R/W	0	LOCKED	中断

Note:

1. 寄存器写入 0x1ACCE551 后，模块解锁
2. 寄存器写入其它值，模块锁定，模块所有寄存器不可改写

18 DMA

18.1 概述

DMA 模块用于实现外设和内存间或内存和内存间或外设和外设间的数据传输，而不需要 CPU 干预。DMA 模块支持 4 条通道用于外设和存储器之间的数据传输。

DMA 通道传输时共用一个数据缓冲区，以循环优先级 Round-Robin 机制进行传输仲裁。该缓冲区内含一个 FIFO，大小为 4 x 32-bit。

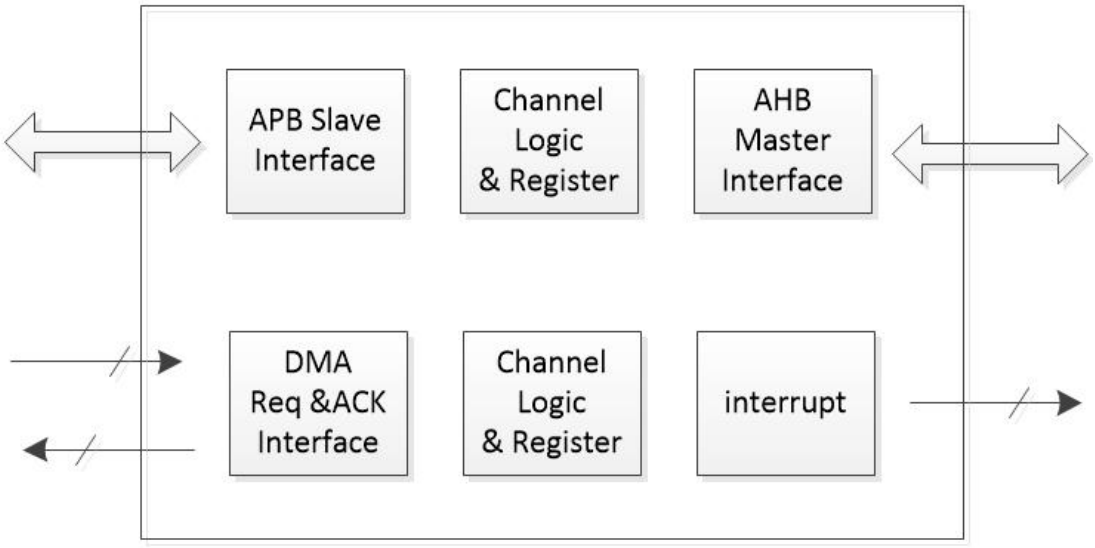


图 18-1 DMA 框图

18.2 特性

- 提供4个独立的可配置的硬件DMA请求传输通道，实现外设和内存间的直接数据传输，每个通道同样支持软件触发。
- 每个通道之间的仲裁使用“Round-Robin”机制
- 所有通道均可配置内存、外设作为访问的源和目标的直接数据传输
- 8-bit、16-bit、和32-bit宽度的数据传输
- 按固定优先级自动实现通道传输仲裁
- DMA通过系统AHB总线完成传输数据
- DMA模块寄存器通过外设总线APB进行配置
- 每个通道都有3个状态标志位（DMA传输完成和DMA当前传输通道），还有一个通道配置错误状态标志位。
- 所有通道均支持自动重复传输

18.3 DMA 通道配置

有 4 个 DMA 通道用于外设和内存之间的数据传输，通道是相互独立的。每个通道可支持多个外设，即一个通道同一时间只能处理一个外设请求。

当需要 DMA 模块在外设和内存间传输数据时，需要将指定 DMA 通道关联至某一特定

外设（详见通道关联外设选择寄存器 DMA_CHCFG），例如 UART_RX、SPI_TX 等；当需要 DMA 模块在外设和外设间传输数据时，需要将指定 DMA 通道关联至源外设，同时使能对应外设的 DMA 请求控制位；当需要 DMA 模块在内存和内存之间传输数据时，需要将指定 DMA 通道的内存和内存之间的数据传输使能（控制寄存器 DMA_CTRL 的 M2M 置 1）。

18.4 DMA 通道传输

18.4.1 通道配置

DMA 请求可以来自多个外设，在发生一个事件后，外设发送一个请求信号到 DMA。DMA 根据通道优先级仲裁处理请求。当 DMA 开始响应外设的请求时，DMA 会发送一个 ACK 信号给外设，外设收到 ACK 信号后，释放它的请求，等待下一个请求的发生。

数据传输由一系列的数据包实现。数据包共有两种：一是从数据源至 DMA 控制器；另一种则是从 DMA 控制器至数据目的地。此 DMA 也支持“外设至外设”访问，此时需要把源地址使能 DMA 模式，而目的地址所对应的外设要当作内存来访问并禁止该外设的 DMA 模式。

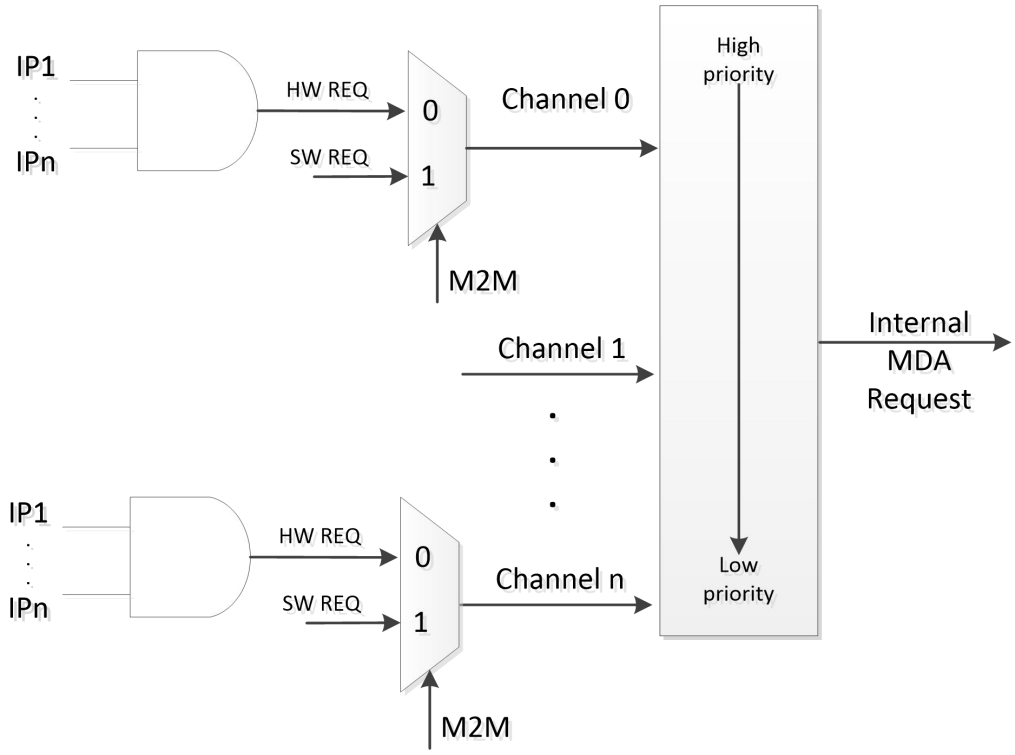


图 18-2 DMA 通道请求图

18.4.2 数据大小

DMA 传输的数据位宽可配置为 8-bit、16-bit 或 32bit，配置控制寄存器 DMACHn_CTL 的 SSIZE/DSIZE。DMA 传输最大数据量是可配置的（配置数据最大长度配置寄存器 DMACHn_MAX），最大可达到 65535，DMA 完成一次请求的数据传输，数据量进行一次递减。

DMA 通道总传输大小=传输数据量 × 数据位宽

18.4.3 地址模式

DMA 的源地址和目的地址分别可以配置为固定地址或者地址递增，具体配置见控制寄存器 DMACHn_CTL 的 SINC 和 DINC。

地址固定表示，每次数据传输后，当前地址不变；地址递增表示，每次数据传输后当前地址以 1、2 或 4 为间隔递增。

18.4.4 自动重载

当 PDMA 控制寄存器 PDMACHn_CTL 中的自动重载控制位 RT 被置位时，当 PDMA 通道数据传输完全完成时，通道 n 当前地址和通道 n 当前传输大小将分别自动载入对应的初始值。通道 n 仍然有效且下一个相关的 PDMA 请求在不需要软件重新配置。

18.4.5 中断状态管理

4 个通道的任意两个通道配置了相同的关联外设设备，配置错误标志位将会立起，如果使能中断，会产生中断。当 DMA 所有数据传输完成，相应通道的传输完成标志位将会立起（IFDn），可以通过配置中断使能寄存器 DMA_MASK 打开对应通道的中断。通过读取状态寄存器 DMA_STAT 的 ACTn 知道当前正在传输数据的通道。

18.5 寄存器列表

地址	寄存器	描述	备注
0x4000_F000	DMACH0_SRCB	DMA 通道 0 数据源基地址	
0x4000_F004	DMACH0_DSTB	DMA 通道 0 数据目的基地址	
0x4000_F008	DMACH0_MAX	DMA 通道 0 数据块最大长度	
0x4000_F00C	DMACH0_CTL	DMA 通道 0 控制寄存器	
0x4000_F010	DMACH0_SCUR	DMA 通道 0 当前数据源地址	
0x4000_F014	DMACH0_DCUR	DMA 通道 0 当前数据目的地址	
0x4000_F018	DMACH0_TCNT	DMA 通道 0 终止传输计数器	
0x4000_F020	DMACH1_SRCB	DMA 通道 1 数据源基地址	
0x4000_F024	DMACH1_DSTB	DMA 通道 1 数据目的基地址	
0x4000_F028	DMACH1_MAX	DMA 通道 1 数据块最大长度	
0x4000_F02C	DMACH1_CTL	DMA 通道 1 控制寄存器	
0x4000_F030	DMACH1_SCUR	DMA 通道 1 当前数据源地址	
0x4000_F034	DMACH1_DCUR	DMA 通道 1 当前数据目的地址	
0x4000_F038	DMACH1_TCNT	DMA 通道 1 终止传输计数器	
0x4000_F040	DMACH2_SRCB	DMA 通道 2 数据源基地址	
0x4000_F044	DMACH2_DSTB	DMA 通道 2 数据目的基地址	
0x4000_F048	DMACH2_MAX	DMA 通道 2 数据块最大长度	
0x4000_F04C	DMACH2_CTL	DMA 通道 2 控制寄存器	
0x4000_F050	DMACH2_SCUR	DMA 通道 2 当前数据源地址	
0x4000_F054	DMACH2_DCUR	DMA 通道 2 当前数据目的地址	
0x4000_F058	DMACH2_TCNT	DMA 通道 2 终止传输计数器	

0x4000_F060	DMACH3_SRCB	DMA 通道 3 数据源基地址	
0x4000_F064	DMACH3_DSTB	DMA 通道 3 数据目的基地址	
0x4000_F068	DMACH3_MAX	DMA 通道 3 数据块最大长度	
0x4000_F06C	DMACH3_CTL	DMA 通道 3 控制寄存器	
0x4000_F070	DMACH3_SCUR	DMA 通道 3 当前数据源地址	
0x4000_F074	DMACH3_DCUR	DMA 通道 3 当前数据目的地址	
0x4000_F078	DMACH3_TCNT	DMA 通道 3 终止传输计数器	
0x4000_F080	DMA_MASK	DMA 中断使能寄存器	
0x4000_F084	DMA_CLR	DMA 中断清除寄存器	
0x4000_F088	DMA_STAT	DMA 状态寄存器	
0x4000_F08C	DMA_CHCFG	DMA 通道关联外设选择寄存器	

18.6 寄存器描述

18.6.1 数据源基地址寄存器 DMACHn_SRCB

位域	类型	复位	名称	描述
[31:0]	R/W	0x0	SOURCE	数据源基地址

18.6.2 数据目的基地址 DMACHn_DSTB

位域	类型	复位	名称	描述
[31:0]	R/W	0x0	DEST	数据目的基地址

18.6.3 数据最大长度配置寄存器 DMACHn_MAX

位域	类型	复位	名称	描述
[31:16]	--	--	--	
[15:0]	R/W	0	MAX_CNT	数据块最大长度配置

Note:

1.数据长度的单位为指定 DMA 传输的数据宽度：8 位/16 位/32 位

18.6.4 控制寄存器 DMACHn_CTL

位域	类型	复位	名称	描述
[31:15]	--	--	--	
[14]	R/W	0	RT	自动重复传输控制 0：在设定数量的数据传输完成后，DMA 停止 1：在设定数量的数据传输完成后，DMA 自动重复工作
[13]	R/W	0	DIR	传输方向控制 0：外设至内存直接数据传输，外设为数据源 1：内存至外设直接数据传输，外设为数据目的
[12]	--	--	--	--
[11]	R/W	0	M2M	内存至内存传输
[10:9]	--	--	--	--
[8:7]	R/W	0	DSIZE	控制器至目的地址的传输数据宽度 00：每次传 1 个字节 01：每次传 2 个字节 10：每次传 4 个字节 11：保留
[6:5]	R/W	0	SBURST	数据源至控制器的传输数据突发长度 00：每次传 1 个单位 01/10/11：保留
[4:3]	R/W	0	SSIZE	数据源至 DMA 控制器的传输数据宽度 00：每次传 1 个字节 01：每次传 2 个字节 10：每次传 4 个字节 11：保留
[2]	R/W	0	DINC	目的地址自动递增使能
[1]	R/W	0	SINC	源地址自动递增使能
[0]	R/W	0	EN	DMA 使能

18.6.5 当前源地址寄存器 DMACHn_SCUR

位域	类型	复位	名称	描述
[31:0]	R	0	SCUR	当前源地址

18.6.6 当前目的地址寄存器 DMACHn_DCUR

位域	类型	复位	名称	描述
[31:0]	R	0	DCUR	当前目的地址

18.6.7 终止传输计数器 DMACHn_TCNT

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R	0	TCNT	一次 DMA 传输过程中有多少数据单元尚未完成

18.6.8 中断使能寄存器 DMA_MASK

位域	类型	复位	名称	描述
[31]	R/W	0	MME	配置错误中断使能
[30:4]	--	--	--	--
[3]	R/W	0	MD3	通道 3 数据传输中断使能
[2]	R/W	0	MD2	通道 2 数据传输中断使能
[1]	R/W	0	MD1	通道 1 数据传输中断使能
[0]	R/W	0	MD0	通道 0 数据传输中断使能

18.6.9 中断清除寄存器 DMA_CLR

位域	类型	复位	名称	描述
[31]	W	0	CME	配置错误中断清除
[30:8]	--	--	--	--
[7:4]	--	--	--	--
[3]	W	0	CD3	通道 3 数据传输中断清除
[2]	W	0	CD2	通道 2 数据传输中断清除
[1]	W	0	CD1	通道 1 数据传输中断清除
[0]	W	0	CD0	通道 0 数据传输中断清除

18.6.10 中断状态寄存器 DMA_STAT

位域	类型	复位	名称	描述
[31]	R	0	IFME	配置错误中断标志
[30:12]	--	--	--	--
[11]	R	0	ACT3	通道 3 数据传输标志
[10]	R	0	ACT2	通道 2 数据传输标志
[9]	R	0	ACT1	通道 1 数据传输标志
[8]	R	0	ACT0	通道 0 数据传输标志
[7:4]	--	--	--	--
[3]	R	0	IFD3	通道 3 数据传输中断标志
[2]	R	0	IFD2	通道 2 数据传输中断标志
[1]	R	0	IFD1	通道 1 数据传输中断标志
[0]	R	0	IFD0	通道 0 数据传输中断标志

18.6.11 通道关联外设选择寄存器 DMA_CHCFG

DMA 通道关联外设选择寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	CH3SEL					-	-	-	*CH2SEL*				
R/W	R	R	R	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	CH1SEL					-	-	-	CH0SEL				

R/W	R	R	R	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

- **位[4:0] CH0SEL： DMA 通道 0 源设备选择**
  按表 19-1 选择 DMA 通道 0 关联的外设功能
- **位[7:5] 保留**
- **位[12:8] CH1SEL： DMA 通道 1 源设备选择**
  按表 19-1 选择 DMA 通道 1 关联的外设功能
- **位[15:13]保留**
- **位[20:16]CH2SEL： DMA 通道 2 源设备选择**
  按表 19-1 选择 DMA 通道 2 关联的外设功能
- **位[23:21]保留**
- **位[28:24]CH3SEL： DMA 通道 3 源设备选择**
  按表 19-1 选择 DMA 通道 3 关联的外设功能
- **位[31:29]保留**

CHnSEL	关联外设
00000b	UART0 TX
00001b	UART0 RX
00010b	UART1 TX
00011b	UART1 RX
00100b	UART2 TX
00101b	UART2 RX
00110b	SPI0 TX
00111b	SPI0 RX
01000b	SPI1 TX
01001b	SPI1 RX
01010b	QSPI TX
01011b	QSPI RX
01100b	TIMER3
01101b	TIMER4
01110b	ADC
01111b ~ 11111b	保留

表 18-1 DMA 通道关联外设

19FLASH 控制器

19.1 Flash 特性

片上提供 512KB 的 Flash 存储空间用于存放程序代码
Flash 基本特性如下：

- Flash 程序区有 32KB 的可用空间
- Flash 快速擦除和编程：
  512 字节块擦: 4-6ms

单字编程 6-7.5us

- >100,000 擦写次数
- >100 年数据保存（25℃）

19.2 Flash 操作控制

Flash 控制模块封装了对 Flash 程序区进行擦除和编程的接口，使得应用程序无需关注 Flash 区在擦除和数据编程时的物理时序和逻辑控制，只需对相关寄存器进行读写操作，设定 Flash 操作命令、地址和数据，即可实现整个 Flash 区的擦除或数据编程工作。

Flash 操作可为：

任意逻辑地址处 Flash 数据读取

任意逻辑地址处 Flash 数据编程写入，写入前应确保单元数据为 0xFFFFFFF（擦除状态）

以 512 字节为单位进行块擦除

整片一次擦除（仅在量产测试模式下，用户模式下仅支持块擦除）

19.3 Flash 擦写时钟选择

Flash 擦写时钟为通过 FLDIV 寄存器的 FLDIV 来对系统时钟进行分频产生 1MHz 的擦写时钟。

19.4 寄存器列表

地址	寄存器	描述	备注
0x4000_0000	FLCMD	FLASH 命令寄存器	FLCMD 说明
0x4000_0004	FLISR	FLASH 中断标志寄存器	FLISR 说明
0x4000_0008	FLIER	FLASH 中断使能寄存器	FLIER 说明
0x4000_000C	FLAR	FLASH 地址寄存器	FLAR 说明
0x4000_0010	FLDR	FLASH 编程数据寄存器	FLDR 说明
0x4000_0028	FLDIV	FLASH 擦写时钟分频寄存器	FLDIV 说明

19.5 寄存器描述

19.5.1 FLASH 命令寄存器 FLCMD

（地址： 0x4000_0000）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	KEYCODE															
R/W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	AINC	-	DPSTB	-	-	MODE	NWS			-	CMD			START
R/W	R	R	RW	R	RW	R	R	RW	RW	RW	RW	R	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] START：　　Flash 操作启动**

  写操作

  0b：　无效

  1b：　启动 Flash 操作命令，对该位写 1 时，务必对

  读操作

  0b：　Flash 操作命令执行完毕

  1b：　Flash 操作命令执行中（执行完毕后硬件自动清 0）

- **位[3:1] CMD：　　Flash 操作命令**

  000b：　单字（32 位）写入编程

  001b：　块擦除

  010b：　保留

  011b：　保留

  1xxb：　保留

- **位[7:5] NWS：　　Flash 操作等待周期**

  设定 Flash 命令操作时的等待周期

  000b：　0 周期等待

  001b：　1 周期等待

  010b：　2 周期等待

  011b：　3 周期等待

  100b：　4 周期等待

  101b：　5 周期等待

  110b：　6 周期等待

  111b：　7 周期等待

- **位[8] MODE：　　Flash 操作模式**

  0b：　Flash 读模式

  1b：　Flash 写模式

  注：当该位为 0 时，任何对 Flash 的写操作无效并且无任何的警告或标志位产生；当该位为 1 时，任何对 Flash 的读操作得到的数据均无效

- **位[10:9] ：　　保留**
- **位[11] DPSTB：　　Flash 深度休眠控制**

  0b：　Flash 深度休眠禁止

  1b：　Flash 深度休眠使能

  注：当软件执行的程序不在 Flash 区而是在别的存储区（比如 SRAM）时，才能对该位进行置 1 操作，不然 Flash 将无法再通过软件唤醒。

- **位[12] ：　　保留**
- **位[13] AINC：　　Flash 地址自动递增**

  0b：　Flash 读写一次后 FLAR 寄存器值保持不变

  1b：　Flash 读写一次后 FLAR 寄存器值自动加 4

- **位[15:14] ：　　保留**
- **位[31:16] ：　　寄存器写操作安全密码输入**

  针对此寄存器低 16 位区域的任何写入操作，必须在此高 16 位区域根据 FLAR 寄存器内的地址同步写入操作的安全密码，对应 MAIN CODE 区域的安全密码为 0xADEB，对应用户信息配置区域的安全密码为 0xC5AE

19.5.2 FLASH 中断状态寄存器 FLISR

（地址： 0x4000_0004）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	ADDR_ERR	-	CMD_ERR	-	PROTECT_ERASE_END	ERASE_END	WRITE_END
R/W	R	R	R	R	R	R	R	R	R	RW1c	R	RW1c	R	RW1c	RW1c	RW1c
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] WRITE_END：　　Flash 写操作命令完成标志位**
  0b：　写命令未完成
  1b：　写命令完成，软件可对该位写 1 将其清 0
- **位[1] ERASE_END：　　Flash 块擦除操作命令完成标志位**
  0b：　块擦除命令未完成
  1b：　块擦除命令完成，软件可对该位写 1 将其清 0

- **位[2] PROTECT_ERASE_END：　　Flash 解除保护擦除主程序操作完成标志位**
  0b：　Flash 解除保护时擦除主程序未完成
  1b：　Flash 解除保护时擦除主程序完成，软件可对该位写 1 将其清 0
- **位[3]：　　保留**
- **位[4] CMD_ERR：　　Flash 操作命令错误标志位**
  0b：　命令正确
  1b：　命令错误，软件可对该位写 1 将其清 0
- **位[5] ：　　保留**
- **位[6] ADDR_ERR：　　Flash 地址错误标志位**
  0b：　地址正确
  1b：　地址错误（Flash 操作地址超限），软件可对该位写 1 将其清 0
- **位[31:7] ：　　保留**

19.5.3 FLASH 中断使能寄存器 FLIER

（地址： 0x4000_0008）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	ADDR_ERRE	-	CMD_ERRE	-	PROTECT_ERASE_ENDE	ERASE_ENDE	WRITE_ENDE
R/W	R	R	R	R	R	R	R	R	R	RW	R	RW	R	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[0] WRITE_ENDE：** **Flash 写操作命令完成中断使能**
  设定 FLISR 寄存器的 WRITE_END 标志位是否引发中断请求
  0b： 中断禁止
  1b： 中断使能
- **位[1] ERASE_ENDE：** **Flash 擦除操作命令完成中断使能**
  设定 FLISR 寄存器的 ERASE_END 标志位是否引发中断请求
  0b： 中断禁止
  1b： 中断使能
- **位[2] PROTECT_ERASE_ENDE：** **Flash 解除保护擦除主程序操作完成中断使能**
  设定 FLISR 寄存器的 PROTECT_ERASE_END 标志位是否引发中断请求
  0b： 中断禁止
  1b： 中断使能
- **位[3]：** **保留**
- **位[4] CMD_ERRE：** **Flash 操作命令错误标志位**
  设定 FLISR 寄存器的 CMD_ERR 标志位是否引发中断请求
  0b： 中断禁止
  1b： 中断使能
- **位[5] ：** **保留**
- **位[6] ADDR_ERRE：** **Flash 地址错误标志位**
  设定 FLISR 寄存器的 ADDR_ERR 标志位是否引发中断请求
  0b： 中断禁止
  1b： 中断使能
- **位[31:7] ：** **保留**

19.5.4 FLASH 地址寄存器 FLAR

（地址： 0x4000_000C）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	BADDR	
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[1:0] BADDR：** **Flash 程序区字节寻址地址**
  Flash 操作总是以 4 字节的字为单位进行，故此两位无效（软件写该位时无效）
- **位[14:2] ADDR：** **Flash 程序区字寻址逻辑地址**
  总计 32KB 的 Flash 程序区主代码空间等价于 8K 的逻辑字空间（每字 4 字节），每个逻辑地址处对应将寻址 4 字节的 Flash 数据
- **位[20:15] ：** **保留**
- **位[21] NVR：** **Flash 程序区操作区域选择**
  FLASH 程序区还有信息配置区，此位决定 Flash 操作针对的具体区域.
  0b： 针对主代码空间进行寻址操作
  1b： 针对信息配置区空间进行寻址操作
- **位[31:22] ：** **保留**

19.5.5 FLASH 编程数据寄存器 FLDR

（地址： 0x4000_0010）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	FLDR															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	FLDR															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[31:0] FLDR： Flash 编程数据**
  内容为 0 时的数据位将被写入 Flash，1 的数据位在编程时被忽略
  编程前确保 Flash 单元中的数据已被擦除为 0xFFFF_FFFF

19.5.6 FLASH 编程数据寄存器 FLDIV

（地址： 0x4000_0028）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	FLDIV							
R/W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

- **位[7:0] FLDIV： Flash 擦写时钟分频系数**
  DIV 用于对系统时钟进行分频，产生 1MHz 的 Flash 擦写时钟。DIV 的配置公式

$$freq_{TICK} = \frac{freq_{SYS}}{FLDIV}$$

  注： $freq_{TICK}$ 为分频后的 1MHz， $freq_{SYS}$ 为系统时钟频率。

20 AES

20.1 概述

AES 模块遵循 NIST AES 标准，实现 Rijndael 密码编码和解码。核心模块提供有 32bits 位宽接口并集成密钥扩展，支持 AES-128、AES-192、AES-256 密钥模式。

支持 CBC/ECB 模式的加解密。在操作方面，对 128bit 密钥 128bit 块输入的 AES 操作需要 11 个时钟周期。而对于 256bit 密钥需要消耗 15 个时钟周期。

特性：

- 符合 NIST FIPS PUB 197 标准
- 动态支持不同长度密钥（128,192 和 256bits）
- 支持 AES-ECB、AES-CBC 模式
- 基于 APB2.0 接口
- 低功耗优化

20.2 结构框图

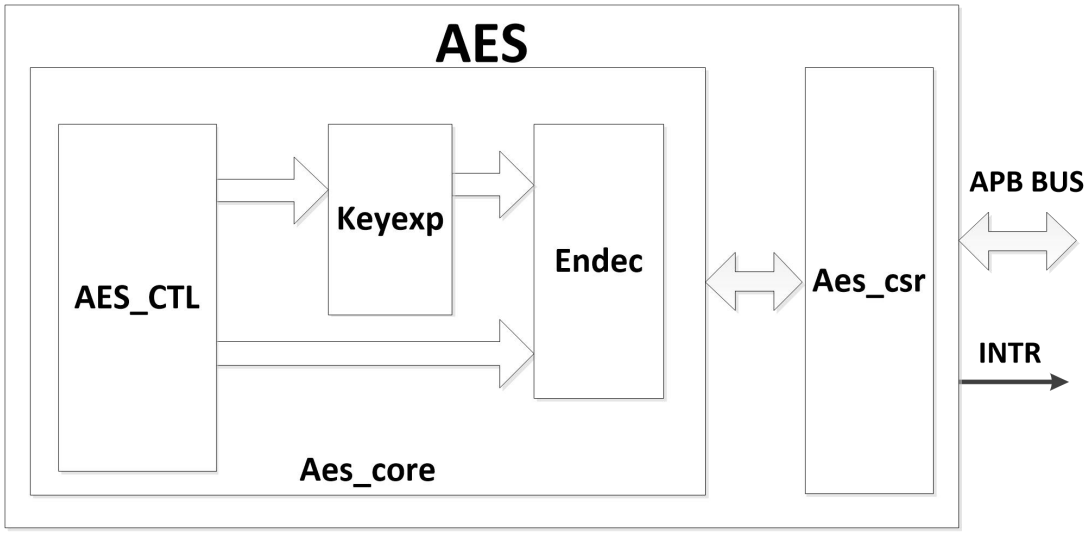


图20-1 AES模块结构框图

20.3 接口

下表为 AES 模块的端口信号：

No	Pin Name	Width	I/O	Description
1	INTR	1	O	中断，操作完成
2	PADDR[7:0]	8	I	PADDR [7:0]
3	PCLK	1	I	PCLK 工作时钟 [APB_BUS]
4	PENABLE	1	I	PENABLE [APB_BUS]
5	PRDATA	32	O	PRDATA [APB_BUS]
6	PRESETn	1	I	PRESETn 复位信号 [APB_BUS]
7	PSEL	1	I	PSEL [APB_BUS]
8	PWDATA	32	I	PWDATA [APB_BUS]
9	PWRITE	1	I	PWRITE [APB_BUS]

20.4 寄存器列表

表格 29-1

地址	寄存器	描述	备注
0x4000_E800	AES_CTRL_STR	AES 控制寄存器	AES_CTRL_STR 说明
0x4000_E804	AES_STA_STR	AES 状态寄存器	AES_STA_STR 说明
0x4000_E810	AES_KEY_REG	ASE 加密密钥配置寄存器（大端模式）	AES_KEY_REG 说明
0x4000_E830	AES_IV_REG	AES 初始矢量配置寄存器（大端模式）	AES_IV_REG 说明
0x4000_E840	AES_DIN_REG	AES 输入数据寄存器（大端模式）	AES_DIN_REG 说明
0x4000_E860	AES_DOUT_REG	AES 输出数据寄存器（大端模式）	AES_DOUT_REG 说明

20.5 寄存器

20.5.1 AES 控制寄存器（AES_CTRL_STR）

Bit	位域	R/W	复位值	描述
0	START	RW	0	开始加/解密操作，自清除 0：空闲 1：AES操作中
1	ENC	RW	0	操作类型： 0:解密 1:加密
2	MODE	RW	0	AES 操作模式： 0:ECB 1:CBC
4:3	KEY_SZ	RW	0	Key size密钥长度: 00:128 01:192 10:256
5	-	RW	0	保留
6	-	RW	0	保留
7	DKEY_GEN	RW	0	生成扩展密钥使能位 当该位与START位同时置一时候，硬件将对密钥进行扩展。 该位仅在加密模式（ENC置1）有效。该位自清除。
8	INT_EN	RW	0	中断使能位 0：不使能 1：使能

20.5.2 AES 状态寄存器（AES_STA_STR）

Bit	位域	R/W	复位值	描述
0	DONE	R	0	操作状态标志寄存器，读取后自清除

				0：空闲或正在操作 1：操作完成

20.5.3 AES 密钥配置寄存器（AES_KEY_REG）

AES 密钥配置寄存器 Keyn，n=0…7 共 8 组密钥配置寄存器（地址：4000E810+4*n H）
对不同密钥长度占用不同数目的 Key，配置从 Key[0]开始.

Bit	位域	R/W	复位值	描述
31:0	Keyn	R/W	0	密钥配置寄存器

20.5.4 AES 初始矢量配置寄存器（AES_IV_REG）

AES 初始矢量配置寄存器 IVn,n=0…3，共 4 组（地址：4000E830+4*n H）

Bit	位域	R/W	复位值	描述
31:0	IVn	R/W	0	初始输入矢量寄存器（仅用于CBC模式）

20.5.5 AES 数据输入寄存器（AES_DIN_REG）

AES 数据输入寄存器 DINn, n=0..3（地址：4000E840+4*n H）

Bit	位域	R/W	复位值	描述
31~0	DINn	RW	0	操作前，需要把数据写入该寄存器

20.5.6 AES 数据输出寄存器（AES_DOUT_REG）

AES 数据输出寄存器 DOUTn ,n=0..3（地址：4000E60+4*n H）

Bit	位域	R/W	复位值	描述
31~0	DOUTn	R	0	状态位done被置起后，再读Dout寄存器值

20.6 功能描述

用户在配置控制寄存器前要配置好 AES_KEY_REG、AES_IV_REG 、AES_DIN_REG. 输入的数据块必须 128bit 对齐，在实际的加解密过程中 AES_DIN_KEY 保持待处理的数据。

状态位 DONE 被置起时候，可以通过 AES_DOUT_REG 读出加/解密的数据。如果中断被使能，那么可以读出中断信号。

该模块包含 DKEY 生成模式，该模式用来把提供的初始密钥生成轮密钥。轮密钥用来解密相应的密文，该轮密钥也可由软件提供。

在 DKEY 生成模式下生成 DKEY 是很简单的操作，用户需要首先把 初始 key 填充到 Key 寄存器，且要设“ENC”位为 1,即处于 ENCODE。控制寄存器中的 KEY_SZ 相应设置、DKEY_GEN 置 1。等待 DONE 信号置起，软件可以从 KEY 寄存器中读出 DKEY.

20.6.1 加密过程

加密过程操作流程如下：

1. 加载密钥到 AES_KEY_REG 寄存器；

2. 加载初始矢量到 AES_IV_REG 寄存器（对于 ECB 操作模式，该步骤可以忽略）；
3. 加载待加密明文到 AES_DIN_REG 数据输入寄存器；
4. 设置相应控制寄存器位。ENC 位相应置 1，DKEY_GEN 置 0.（其他位：根据不同的密钥长度相应设置位对应值，START 置 1 等）；
5. 等待操作状态标志寄存器置起；
6. 读取 AES_DOUT_REG 数据输出寄存器中的密文。

20.6.2 解密过程

解密过程软件操作步骤如下：

1） 加载密钥到 AES_KEY_REG 寄存器中；
2） 加载初始矢量 IV 到 AES_IV_REG 中（如果是 ECB 模式，可以忽略此步）；
3） 加载 128 位密文到 AES_DIN_REG 数据集输入寄存器中；
4） 设置好控制寄存器的相应位，ENC=0,DKEY_GEN=0；
5） 等到 DONE 状态标志位置起，或者等待中断信号输出高；
6） 从 AES_DOUT_REG 输出寄存器中读出明文。

注意：

解密所用的密钥需要先做密钥扩展！参考生成解密密钥生成解密密钥章节。

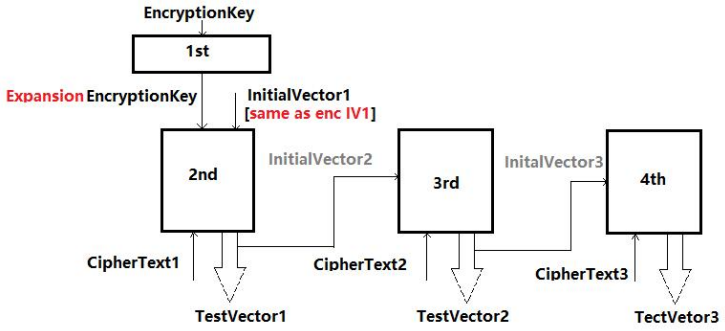


图 20-2CBC 模式解密流程

20.6.3 生成解密密钥

对于解密所需要的密钥(轮密钥)，可以使用 AES 模块生成。操作步骤如下：

1）加载密钥到 AES_KEY_REG,此密钥同加密所用密钥；
2）设置控制寄存器相应位：**ENC=1**,DKEY_GEN=1；
3) 等待 DONE 状态位置起好或者等待中断信号输出高电平；
4）从 AES_KEY_REG 寄存器中读出解密密钥。

20.6.4 AES 操作模式

AES 可以在有不同的工作模式，该模式可以通过控制寄存器配置。他支持 ECB,CBC(分状态操作)。

20.6.4.1 ECB 模式

ECB 模式是最基本的操作模式，输入的数据 block 必须为 128 位。相应输出也是 128 位的。对于 ECB 模式不存在分状态操作。

20.6.4.2 CBC 模式

AES 模块支持 CBC 模式下的分状态操作。因 CBC 加密过程是通过本次输出密文作为下一次加密所用加密初始矢量，所以会有个加密的先后顺序。

注意：在一个完整的 CBC 操作过程中，ENC 位、MODE 位、KEY_SZ 位、KEY 寄存器、IV 寄存器不能被改变。也就是说当分状态操作的最后一个 block 被加密完成之后，这些寄存器才能被改变。

在 CBC 模式下，第一个 block 与 AES_IV_REG 所存入的矢量相互作用。第一个 block 加密生成的密文作为下一个 block 的加密矢量。所有的加密 block 长度必须为 128 位对齐。最后不足 128 位的用户要做补位填充。

在整个加密流程中，IV 初始矢量是有硬件自动加载上次加密结果的（除了第一个 block）的，所以对于中间的 block 或者最后的一个 block 用户无需加载 IV 初始矢量到 AES_IV_REG 中去。

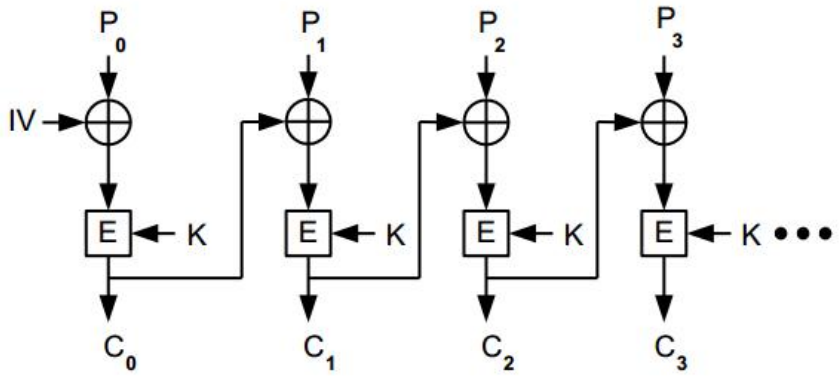


图 20-3 AES-CBC 加密过程示意图

对于解密过程，就是上面逆操作。每次解密后，前面的密文都要作为后面 block 的解密所需的初始矢量。

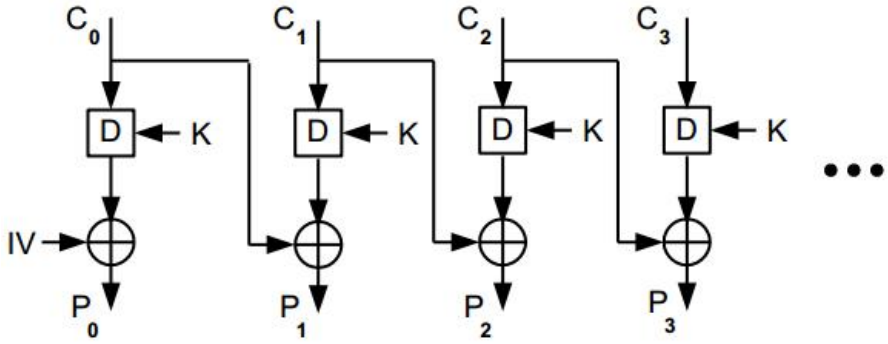
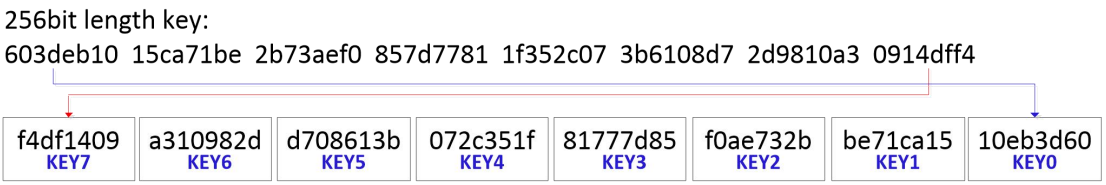


图 20-4 AES-CBC 解密操作

20.7 AES 的应用

AES 遵循 apb2.0 总线协议，总线位宽为 32 位，对于 128bit 的数据则要分次传输。下面实例演示 256 位宽密钥的寄存器配置方式

示例如下：



若 key 长度为 128 位，那么相应填充 KEY0/KEY1/KEY2/KEY3 寄存器即可。数据的配置读取和 key 类似。

20.7.1 ECB 模式下的加密流程

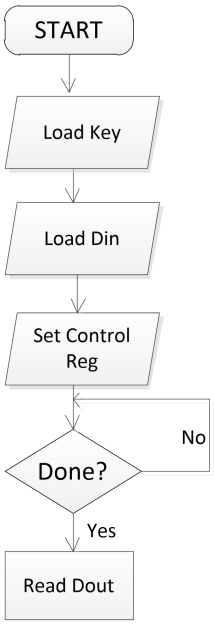


图 20-5ECB 模式下加密流程示意图

ECB 工作模式下，加密应用相对较简单，把数据送到 Key，Din 寄存器里，然后设置好控制寄存器的相应位，等待 DONE 信号置起就可以去 Dout 寄存器读加密后的密文。

20.7.2 CBC 模式下的加密流程

CBC 模式下加密工作流程示意图如下图所示：

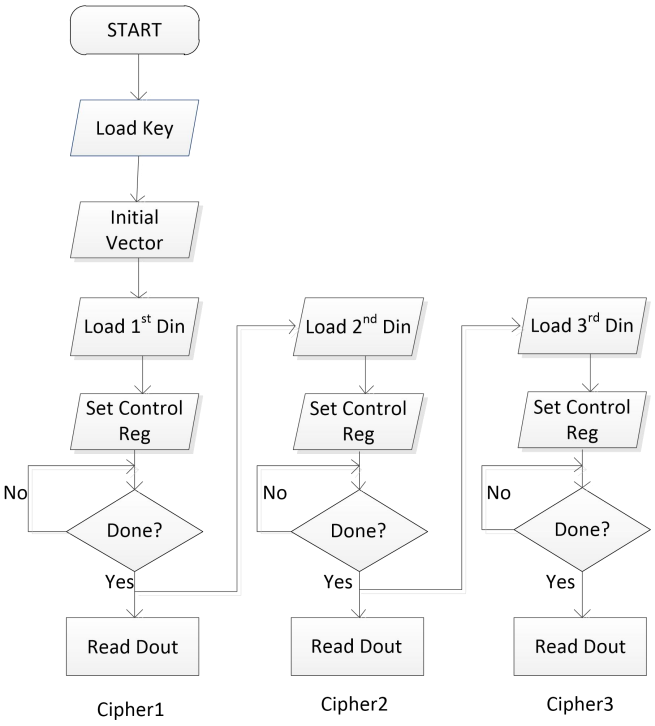


图 20-6CBC 模式下的加密流程

这里演示对 3 组明文进行加密。初始矢量开始时候加载之后，后续 BLOCK 的加密无需手动指定初始矢量。

20.7.3 ECB 模式下的解密流程

ECB 模式下的解密流程如下图所示：

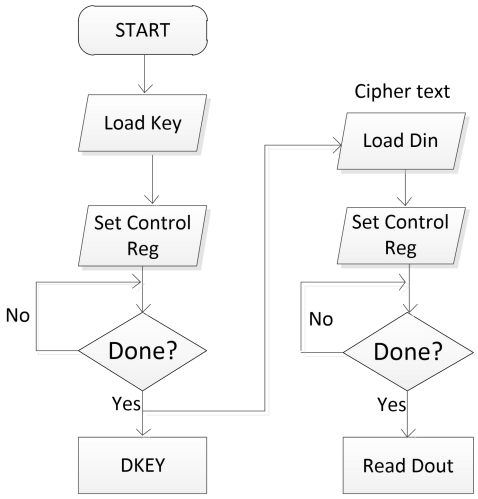


图 20-7ECB 模式下的解密流程示意图

上图解释了 ECB 模式下的解密流程，解密所需密钥同加密密钥。我们先对密钥首先进行一个密钥的扩展处理，生成轮密钥 Dkey。这个密钥既可以由 AES 硬件模块产生，又可以由软件事先产生。

20.7.4 CBC 模式下的解密流程

CBC 模式下的解密流程示意图如下所示：

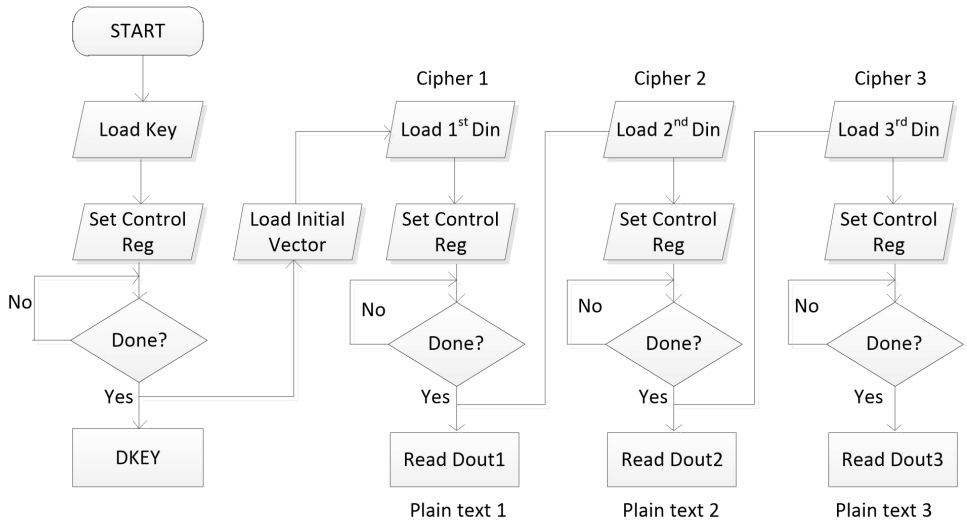


图 20-8CBC 模式下的解密流程

CBC 模式下的解密过程，同样首先要进行一个密钥扩展生成 DKEY（轮密钥）,该 DKEY 也可以由软件产生。CBC 模式解密所需要密钥和初始矢量同加密。而中间所需的初始矢量无需指定加载，AES 模块会自动调用。

20.8 AES 工作时序

20.8.1 ECB 模式下的加解密

ECB 加密时序示意图如下：

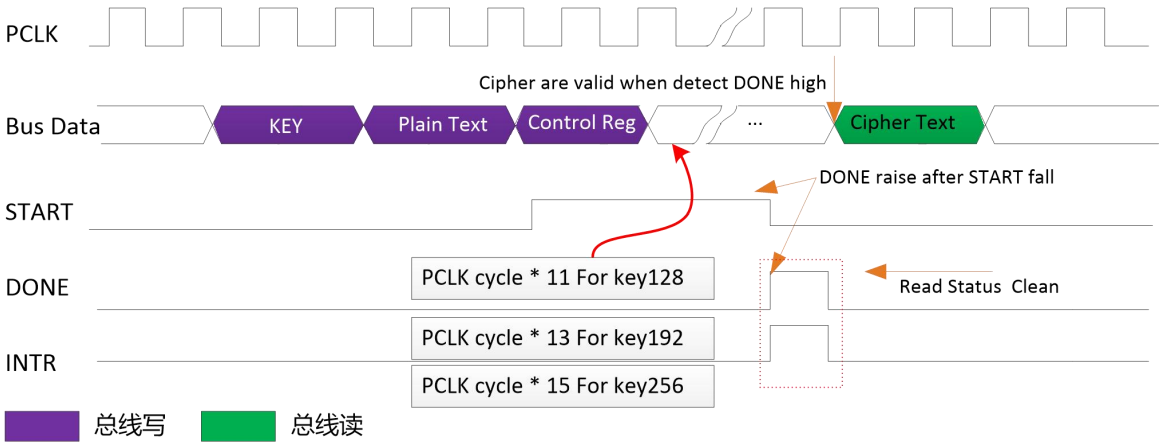


图 20-9ECB 模式下加密时序

对于中断标志信号 INTR，当中断使能位被使能即 INT_EN=1 时，INTR 信号与 DONE 信号同步变换（其他模式下一样）。控制寄存器的 START=1 时候，START 信号相应被置起，当操作完成 DONE 信号置起，START 信号被拉低。

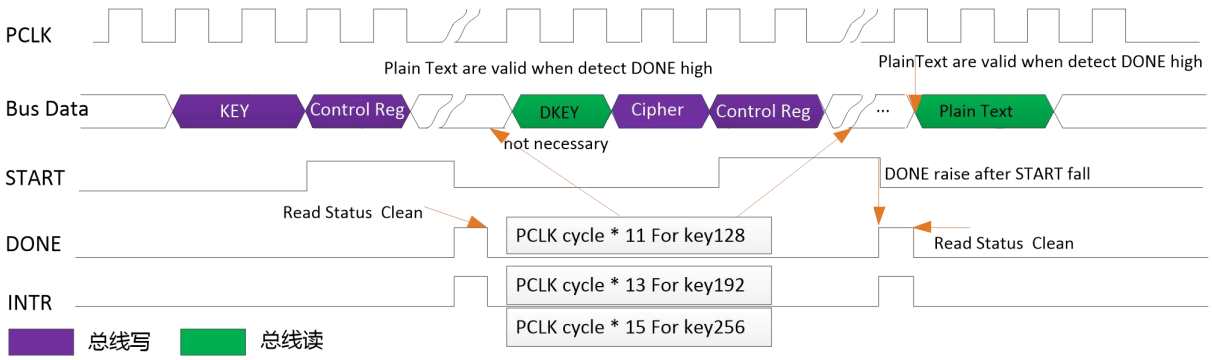


图 20-10ECB 模式下解密时序

解密时序基本和加密过程类似，只是多了一步的预先 DKEN 生成的过程。ECB 模式下的解密流程章节有轮密钥生成配置过程。将密钥（同加密所用密钥）写入 KEY 寄存器，设置控制寄存器为生成 DKEY 模式。等待 DONE 置起后，写入密文到 DIN 寄存器，设置控制寄存器。等待 DONE 信号置起后，读出明文。对于中断信号 INTR，当中断使能位被使能即 INT_EN=1 时，INTR 信号与 DONE 信号同步变换。

20.8.2 CBC 模式下的加解密

CBC 模式下的加密工作时序如下图：

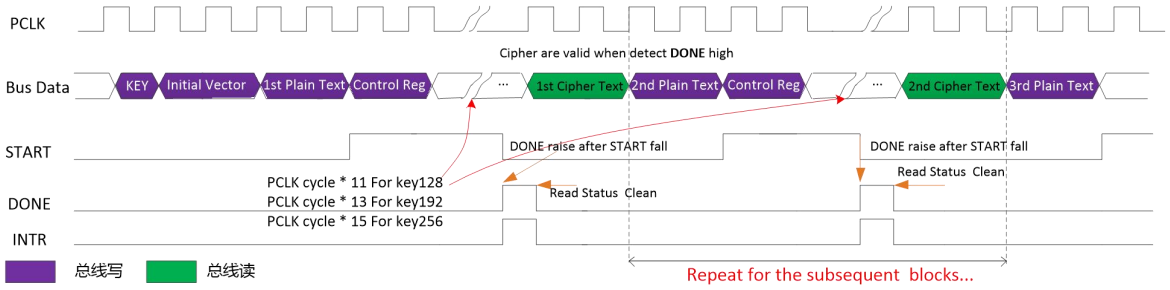


图 20-11 CBC 加密时序示意图

在 CBC 模式下，需要指定初始矢量，对于第一个 block 的加密 IV 有我们配置 IV 寄存器，而后续的 block 加密无需我们再次设置 IV 寄存器。对于中断信号 INTR，当中断使能位被使能即 INT_EN=1 时，INTR 信号与 DONE 信号同步变换。

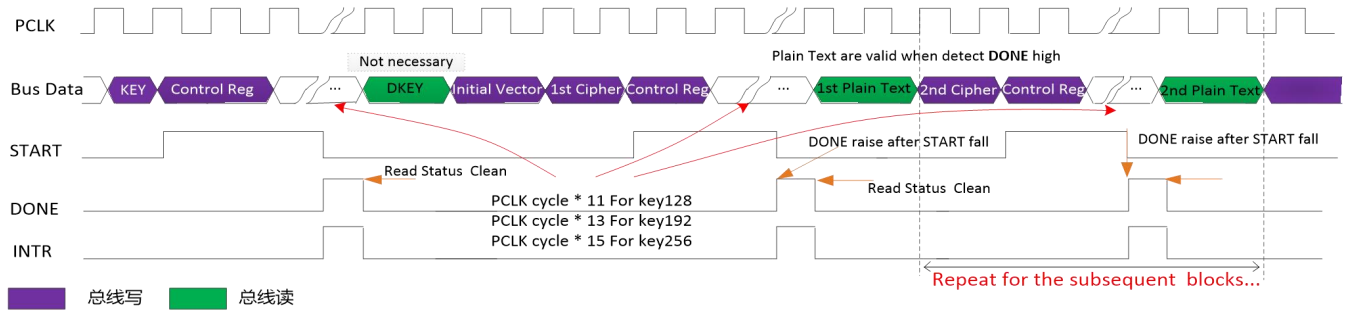


图 20-12 CBC 解密示意图

在该模式下，需首先生成解密所需密钥（轮密钥）。该轮密钥也可以由软件产生。生成解密密钥章节有轮密钥生成配置过程。后续操作和配置过程可参考解密过程章节和 CBC 模式下的解密流程章节。对于中断信号 INTR，当 INT_EN=1 时，INTR 与 DONE 信号同步变化。

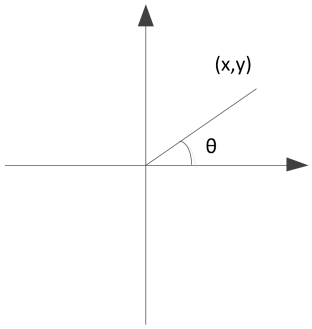
21 DSP 硬件加速器

21.1 sqrt（开方计算）

输入一个有符号数，对此数值进行开方。

21.2 atan（反正切计算）

已知 X，Y（32 位，有符号数），求θ的角度值(8 位数据，范围 0~239)

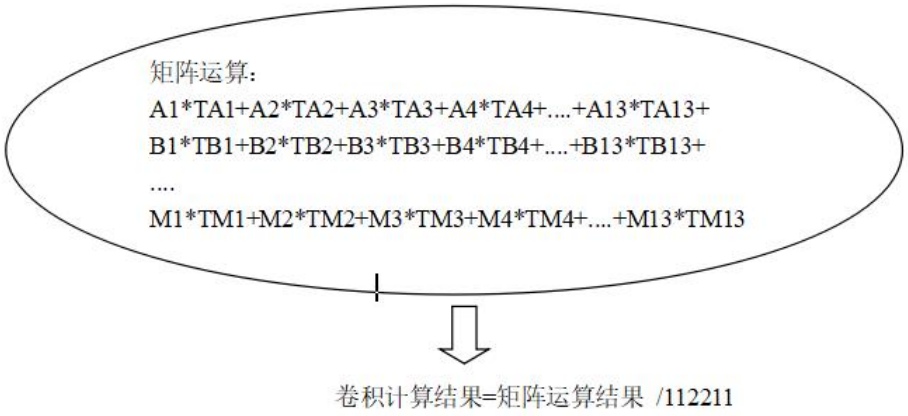


21.3 卷积矩阵计算

卷积计算时，数据输入与卷积核都为一个 13x13 的矩阵，如下图所示：

M1 →	M1	L1	K1	J1	I1	H1	G1	F1	E1	D1	C1	B1	A1 →
M2 →	M2	L2	K2	J2	I2	H2	G2	F2	E2	D2	C2	B2	A2 →
M3 →	M3	L3	K3	J3	I3	H3	G3	F3	E3	D3	C3	B3	A3 →
M4 →	M4	L4	K4	J4	I4	H4	G4	F4	E4	D4	C4	B4	A4 →
	...	...	...	数据移位寄存器阵列									
	...	...	...										
M12 →	M12	L12	K12	J12	I12	H12	G12	F12	E12	D12	C12	B12	A12 →
M13 →	M13	L13	K13	J13	I13	H13	G13	F13	E13	D13	C13	B13	A13 →

TM1	TL1	TK1	TJ1	TI1	TH1	TG1	TF1	TE1	TD1	TC1	TB1	TA1
TM2	TL2	TK2	TJ2	TI2	TH2	TG2	TF2	TE2	TD2	TC2	TB2	TA2
TM3	TL3	TK3	TJ3	TI3	TH3	TG3	TF3	TE3	TD3	TC3	TB3	TA3
TM4	TL4	TK4	TJ4	TI4	TH4	TG4	TF4	TE4	TD4	TC4	TB4	TA4
...	...	...	卷积核寄存器阵列									
...	...	...										
TM12	TL12	TK12	TJ12	TI12	TH12	TG12	TF12	TE12	TD12	TC12	TB12	TA12
TM13	TL13	TK13	TJ13	TI13	TH13	TG13	TF13	TE13	TD13	TC13	TB13	TA13

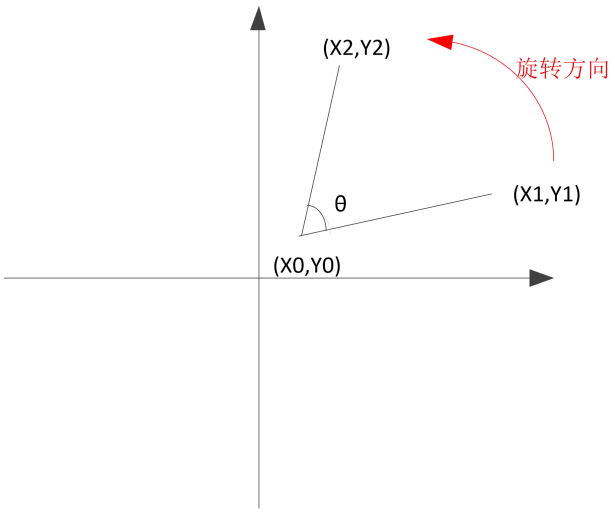


第一次卷积计算，数据移位寄存器阵列需填入 13x13 共 169 笔数据，卷积核寄存器阵列同样是 13x13 共 169 笔数据，启动卷积计算得到结果后，数据移位寄存器阵列会向右平移一列将 A 列数据移出，此时 A~L 列数据为对应的后一列(B~M)数据；所以在第一次卷积计算后仅需填充 M 列共 13 笔数据，即可得到 13x13 的卷积矩阵计算结果。

21.4 点坐标旋转

以某点为基准，逆时针旋转某角度后，某坐标的新坐标值

21.4.1 算法描述



21.5 寄存器列表

地址	寄存器	描述	备注
0x4000_1800	DSP_STATUS		
0x4000_1804	ABSCALC	绝对值计算	
0x4000_1808	SQRTCALC	开方计算	
0x4000_180	ATAN_X	反正切 X 输入值	

C			
0x4000_1810	ATAN_Y	反正切 Y 输入值	
0x4000_1814	ATAN_RESULT	反正切计算结果	
0x4000_1850	ROTATE_ANGLE	坐标旋转角度	
0x4000_1854	ROTATE_XY0	坐标旋转原点坐标 X0,Y0	
0x4000_1858	ROTATE_XY1	坐标旋转原点坐标 Y1,Y1	
0x4000_185C	ROTATE_XY	坐标旋转后新坐标 X,Y	
0x4000_18B0	CONV_A1_A4	卷积数据阵列第 1 列(元素 1~4)	
0x4000_18B4	CONV_A5_A8	卷积数据阵列第 1 列(元素 5~8)	
0x4000_18B8	CONV_A9_A12	卷积数据阵列第 1 列(元素 9~12)	
0x4000_18BC	CONV_A13	卷积数据阵列第 1 列(元素 13)	
0x4000_18C0	CONV_B1_B4	卷积数据阵列第 2 列(元素 1~4)	
0x4000_18C4	CONV_B5_B8	卷积数据阵列第 2 列(元素 5~8)	
0x4000_18C8	CONV_B9_B12	卷积数据阵列第 2 列(元素 9~12)	
0x4000_18CC	CONV_B13	卷积数据阵列第 2 列(元素 13)	
0x4000_18D0	CONV_C1_C4	卷积数据阵列第 3 列(元素 1~4)	
0x4000_18D4	CONV_C5_C8	卷积数据阵列第 3 列(元素 5~8)	
0x4000_18D8	CONV_C9_C12	卷积数据阵列第 3 列(元素 9~12)	
0x4000_18DC	CONV_C13	卷积数据阵列第 3 列(元素 13)	
0x4000_18E0	CONV_D1_D4	卷积数据阵列第 4 列(元素 1~4)	
0x4000_18E4	CONV_D5_D8	卷积数据阵列第 4 列(元素 5~8)	
0x4000_18E8	CONV_D9_D12	卷积数据阵列第 4 列(元素 9~12)	
0x4000_18EC	CONV_D13	卷积数据阵列第 4 列(元素 13)	
0x4000_18F0	CONV_E1_E4	卷积数据阵列第 5 列(元素 1~4)	
0x4000_18F4	CONV_E5_E8	卷积数据阵列第 5 列(元素 5~8)	
0x4000_18F8	CONV_E9_E12	卷积数据阵列第 5 列(元素 9~12)	
0x4000_18FC	CONV_E13	卷积数据阵列第 5 列(元素 13)	
0x4000_1900	CONV_F1_F4	卷积数据阵列第 6 列(元素 1~4)	
0x4000_1904	CONV_F5_F8	卷积数据阵列第 6 列(元素 5~8)	
0x4000_1908	CONV_F9_F12	卷积数据阵列第 6 列(元素 9~12)	
0x4000_190	CONV_F13	卷积数据阵列第 6 列(元素 13)	

C			
0x4000_1910	CONV_G1_G4	卷积数据阵列第 7 列(元素 1~4)	
0x4000_1914	CONV_G5_G8	卷积数据阵列第 7 列(元素 5~8)	
0x4000_1918	CONV_G9_G12	卷积数据阵列第 7 列(元素 9~12)	
0x4000_191C	CONV_G13	卷积数据阵列第 7 列(元素 13)	
0x4000_1920	CONV_H1_H4	卷积数据阵列第 8 列(元素 1~4)	
0x4000_1924	CONV_H5_H8	卷积数据阵列第 8 列(元素 5~8)	
0x4000_1928	CONV_H9_H12	卷积数据阵列第 8 列(元素 9~12)	
0x4000_192C	CONV_H13	卷积数据阵列第 8 列(元素 13)	
0x4000_1930	CONV_I1_I4	卷积数据阵列第 9 列(元素 1~4)	
0x4000_1934	CONV_I5_I8	卷积数据阵列第 9 列(元素 5~8)	
0x4000_1938	CONV_I9_I12	卷积数据阵列第 9 列(元素 9~12)	
0x4000_193C	CONV_I13	卷积数据阵列第 9 列(元素 13)	
0x4000_1940	CONV_J1_J4	卷积数据阵列第 10 列(元素 1~4)	
0x4000_1944	CONV_J5_J8	卷积数据阵列第 10 列(元素 5~8)	
0x4000_1948	CONV_J9_J12	卷积数据阵列第 10 列(元素 9~12)	
0x4000_194C	CONV_J13	卷积数据阵列第 10 列(元素 13)	
0x4000_1950	CONV_K1_K4	卷积数据阵列第 11 列(元素 1~4)	
0x4000_1954	CONV_K5_K8	卷积数据阵列第 11 列(元素 5~8)	
0x4000_1958	CONV_K9_K12	卷积数据阵列第 11 列(元素 9~12)	
0x4000_195C	CONV_K13	卷积数据阵列第 11 列(元素 13)	
0x4000_1960	CONV_L1_L4	卷积数据阵列第 12 列(元素 1~4)	
0x4000_1964	CONV_L5_L8	卷积数据阵列第 12 列(元素 5~8)	
0x4000_1968	CONV_L9_L12	卷积数据阵列第 12 列(元素 9~12)	
0x4000_196C	CONV_L13	卷积数据阵列第 12 列(元素 13)	
0x4000_1970	CONV_M1_M4	卷积数据阵列第 13 列(元素 1~4)	
0x4000_1974	CONV_M5_M8	卷积数据阵列第 13 列(元素 5~8)	
0x4000_1978	CONV_M9_M12	卷积数据阵列第 13 列(元素 9~12)	
0x4000_197C	CONV_M13	卷积数据阵列第 13 列(元素 13)	
0x4000_1980	CONV_CTL	卷积控制寄存器	
0x4000_1984	CONV_CORE_SEL	卷积核选择寄存器	
0x4000_1988	CONV_RESULT	卷积结果寄存器	

21.6 寄存器描述

21.6.1 ABSCALC

绝对值计算寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	ABS_DATA/RESULT															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	ABS_DATA/RESULT															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[31:0] ABS_DATA/RESULT：　　绝对值计算输入&输出寄存器**

  写操作：　写入 32 位待绝对值计算数值，最高位为符号位。

  读操作：　读出上一次写入的数值的绝对值结果，为 32 位无符号数。

21.6.2 SQRTCALC

该寄存器用于设定开方输入/保存计算结果

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	sqrt_data															
R/W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	sqrt_data/sqrt_result															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[31:0]　sqrt_data：**

写：发送 32bit 数据，sqrt 开始计算，取值范围 0x0~0xFFFFFFFF。

- **位[15:0]　sqrt_data：**

读：返回 16bit 开方结果

21.6.3 ATAN_X

该寄存器用于设定待计算反正切 X 坐标

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	atan_x															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	atan_x															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[31:0] atan_x： 反正切 X 坐标值**
  取值范围 0x0000~0xFFFFFFFF。

21.6.4 ATAN_Y

该寄存器用于设定待计算反正切 Y 坐标

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	atan_y															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	atan_y															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[31:0] atan_y： 反正切 y 坐标值**
  取值范围 0x0000~0xFFFFFFFF。

21.6.5 ATAN_RESULT

该寄存器用于保存反正切计算结果

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	atan_result							
R/W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[7:0] atan_result： 反正切计算结果**

读：返回 8bit[7:0]反正切计算结果

写：写任意值复位除输入转换坐标外复位 ATAN 模块

21.6.6 ROTATE_ANGLE_IN

该寄存器用于设定坐标旋转角度

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	rotate_angle							
R/W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[7:0] rotate_angle： 坐标旋转角度**

21.6.7 ROTATE_XY0_IN

该寄存器用于设定原点坐标(x0,y0)

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	rotate_x0															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	rotate_y0															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[15:0] rotate_y0： 旋转原点坐标 y0**
- **位[31:16] rotate_x0： 旋转原点坐标 x0**

21.6.8 ROTATE_XY1_IN

该寄存器用于设定原点坐标(x1,y1)

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	rotate_x1															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	rotate_y1															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[15:0] rotate_y1： 旋转原点坐标 y1**

- **位[31:16] rotate_x1： 旋转原点坐标 x1**

21.6.9 ROTATE_XY_OUT

该寄存器用于保存旋转后的坐标(x,y)

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	rotate_x1															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	rotate_y1															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

读：

- **位[15:0] rotate_y： 旋转坐标 y**

- **位[31:16] rotate_x： 旋转坐标 x**

写：写任意值复位除输入坐标/角度外的坐标旋转模块

21.6.10 CONV_X1_X4 (X=A,B,C,D....L,M)

该寄存器用于保存卷积据数组第 x 列元素 X1~X4

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	X4								X3							
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	X2								X1							
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位 [31:24] X4 ： 卷 积 第 X 列 元 素 X4(x=A,B,C..L,M)**

- **位 [23:16] X3 ： 卷 积 第 X 列 元 素 X3(x=A,B,C..L,M)**

- 位 [15:8] X2 ： 卷积第 X 列元素
  X2(x=A,B,C..L,M)

- 位 [7:0] X1 ： 卷积第 X 列元素
  X1(x=A,B,C..L,M)

21.6.11 CONV_X5_X8 (X=A,B,C,D....L,M)

该寄存器用于保存卷积据数组第 x 列元素 X5~X6

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	X8								X7							
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	X6								X5							
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- 位 [31:24] X8 ： 卷积第 X 列元素
  X8(x=A,B,C..L,M)

- 位 [23:16] X7 ： 卷积第 X 列元素
  X7(x=A,B,C..L,M)

- 位 [15:8] X6 ： 卷积第 X 列元素
  X6(x=A,B,C..L,M)

- 位 [7:0] X5 ： 卷积第 X 列元素
  X5(x=A,B,C..L,M)

21.6.12 CONV_X9_X12 (X=A,B,C,D....L,M)

该寄存器用于保存卷积据数组第 x 列元素 X9~X12

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	X12								X11							
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	X10								X9							
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

-位[31:24] X12：卷积第 X 列元素 X12(x=A,B,C..L,M)

-位[23:16] X11：卷积第 X 列元素 X11(x=A,B,C..L,M)

-位[15:8] X10：卷积第 X 列元素 X10(x=A,B,C..L,M)

-位[7:0] X9： 卷积第 X 列元素 X9(x=A,B,C..L,M)

21.6.13 CONV_X13 (X=A,B,C,D....L,M)

该寄存器用于保存卷积据数组第 x 列元素 X13

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	X13							
R/W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

-位[7:0] X13： 卷积第 X 列元素 X13(x=A,B,C..L,M)

21.6.14 CONV_ CTL

该寄存器设定卷积控制寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	conv_div	conv_start
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

-位[0] conv_start： 启动卷积计算(对 conv_result 寄存
器写 0 将此为清除)

-位[0] conv_div： 写 0：卷积结果除以 112211

写 1：卷积结果不除以 112211

21.6.15 CONV_CORE_SEL

该寄存器用于选择卷积核矩阵

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	-	-	-	-	-	-	-	-	core_sel							
R/W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

-位[7:0] core_sel:选择卷积核矩阵(0~239)

21.6.16 CONV_RESULT

该寄存器用于保存卷积计算结果

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
位域	conv_result															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位域	conv_result															
R/W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **位[31:0] conv_result： 卷积计算结果(写 0 将产**
- **生卷积计算复位信号)**

22 RTC 实时时钟控制

22.1 RTC 模块综述

实时时钟（RTC）包含32位宽向上计数器，一个控制寄存器、一个比较寄存器和一个状态寄存器。

RTC模块主要特性包括：

- 32-bit向上计数器，用于计算所用时间
- 32-bit比较寄存器用于闹钟功能
- RTC时钟源
  - 外部32KHz晶振
  - 外部24KHz晶振除频后得到的32KHz时钟
- 中断设置，可关闭的比较匹配中断

22.2 RTC 框图

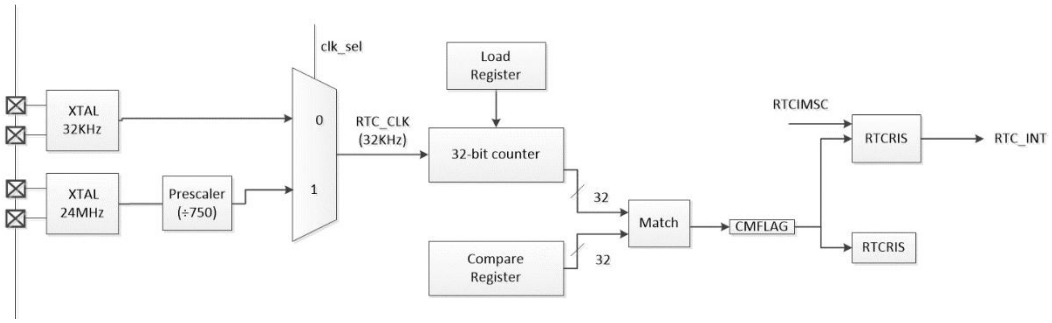


图 22-1 实时定时器 RTC 工作原理图

图 22-1 显示了实时定时器 RTC 的工作原理。RTC 所有寄存器的读写均在 PCLK 下可通过软件进行读写操作，定时器时钟来源有两个，可通过配置寄存器 RTCCR 的 CLK_SEL 选择外部 32KHz 晶振或由外部 24KHz 晶振除频后得到的 32KHz 时钟作为 RTC 时钟。

配置 RTC_CR 寄存器 start，使能 RTC，RTC 开始计数，一旦使能，除非发生复位，否则 RTC 将一直计数。对 RTC_LR 寄存器写值，将载入 RTC 计数值，两个 pclk 时钟后，更新 RTC 当前计数值，RTC Counter 从载入 RTC 计数值开始计数。当 RTC Counter 等于比较匹配值寄存器 RTC_MR 时，比较匹配状态位立起，也可以通过配置中断状态寄存器 RTC_IR 使能中断，当中断状态位立起，即发生比较匹配，产生中断。对中断清除寄存器 RTC_ICR 写 1，可清除状态位。

下图是 RTC 计数示意图，配置比较匹配值寄存器 RTC_MR=0x51，使能中断，在 RTC 计数到 0x51 时，发生中断 INT，在计数到 0x7e 时，写载入寄存器 RTC_LR =0xc0，经过更新逻辑电路，RTC 从 0xc0 计算，这时也配置比较匹配值寄存器 RTC_MR=0x1b0，在计数到 0x101 时，写载入寄存器 RTC_LR =0x153，同理，在 RTC 计数到 0x1b0 时，发生中断 INT。

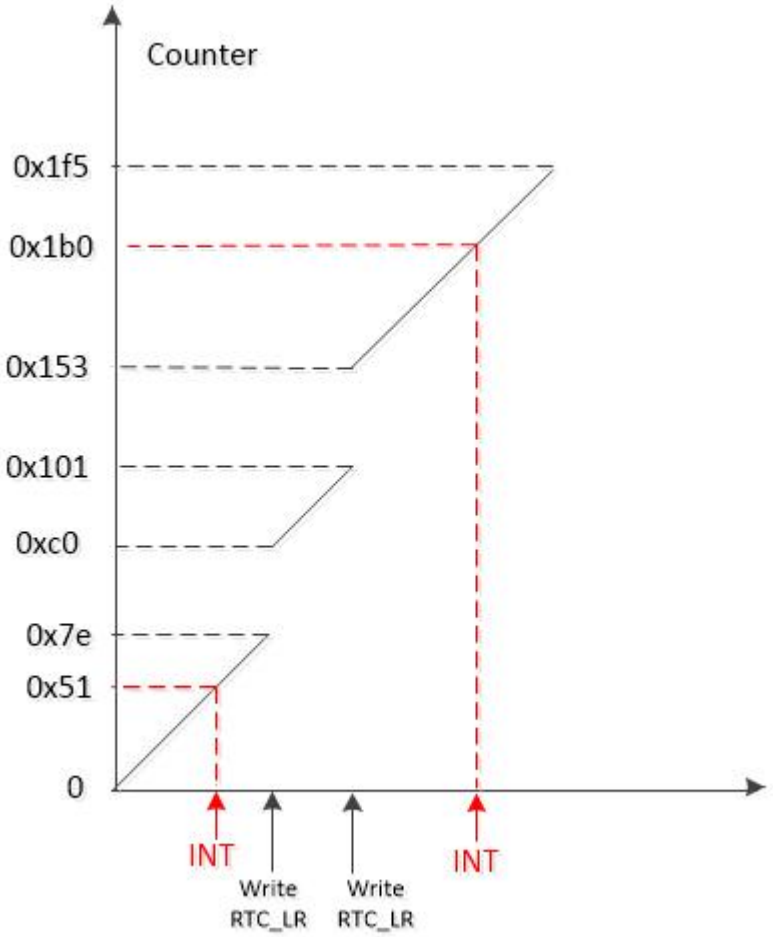


图 22-2 RTC 计数示意图

22.3 寄存器列表

地址	寄存器	描述	备注
0x4000_2800	RTC_DR	RTC当前计数值寄存器	
0x4000_2804	RTC_MR	RTC比较匹配值寄存器	
0x4000_2808	RTC_LR	RTC载入值寄存器	
0x4000_280C	RTC_CR	RTC控制寄存器	
0x4000_2810	RTC_IR	RTC中断寄存器	
0x4000_2814	RTC_RISR	RTC比较匹配状态寄存器	
0x4000_2818	RTC_MISR	RTC中断状态寄存器	
0x4000_281C	RTC_ICR	RTC中断清除寄存器	

22.4 寄存器描述

22.4.1 当前计数值寄存器 RTC_DR

位域	类型	复位	名称	描述
[31：0]	R	0	rtc_cnt	只读，当使能 RTC 后，表示 RTC 当前计数值，取值范围 0x0000_0000~0Xffff_ffff。当 start=0 时，rtc_cnt=0.

22.4.2 比较匹配值寄存器 RTC_MR

位域	类型	复位	名称	描述
[31：0]	R/W	0	Match_cnt	比较匹配值

22.4.3 载入值寄存器 RTC_LR

位域	类型	复位	名称	描述
[31：0]	R/W	0	load_cnt	载入一个计数值，更新到 RTC 计数值

22.4.4 控制寄存器 RTC_CR

位域	类型	复位	名称	描述
[31：2]	--	--	--	--
[1]	R/W	0	clk_sel	RTC 时钟源选择 0：外部 32KHz 晶振 1：外部 24KHz 晶振除频后得到的 32KHz 时钟
[0]	R/W	0	start	使能 RTC，开始计数。自使能后，除非发生复位，才能 stop RTC。 写 0 无效。

22.4.5 中断寄存器 RTC_IR

位域	类型	复位	名称	描述
[31：1]	--	--	--	--
[0]	R/W	0	int_en	使能 RTC 中断。

22.4.6 比较匹配状态寄存器 RTC_RISR

位域	类型	复位	名称	描述
[31：1]	--	--	--	--
[0]	R	0	RIS	RTC 比较匹配状态位，RTC 发生比较匹配时，该状态位立起来。

22.4.7 中断状态寄存器 RTC_MISR

位域	类型	复位	名称	描述
[31：1]	--	--	--	--
[0]	R	0	MIS	RTC 比较匹配中断状态位，只读，当中断使能时，发生比较匹配时，该状态位立起来。

22.4.8 中断清除寄存器 RTC_ICR

位域	类型	复位	名称	描述
[31：1]	--	--	--	--
[0]	R	0	CMCL	清除 RTC 比较匹配中断状态位。

23 ADC

23.1 概述

该 ADC 有 9 个通道，允许 ADC 测量 6 个外部输入引脚电压以及内部 Bandgap 电压等。

23.2 特点

- 12 位的模数转换分辨率
- 最多13路单端模拟信号输入，6路差分模拟信号输入
- 片内通道包括：
  - BandGap电压
  - 1.2v参考电压
- 输入电压范围：0~Vdd
- 参考电压：Vdd
- AD 转换完成可触发中断
- 单次 AD 转换模式、连续 AD 转换模式、定时器溢出信号触发 AD 转换模式。

23.3 ADC 控制时序

23.3.1 ADC 启动时序

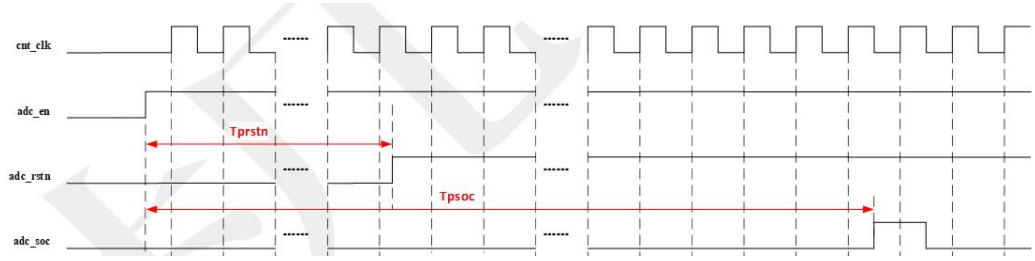


图 23-1 ADC 启动时序

上图为 ADC 启动时序，其中 Tprstn 需要大于 1us，Tpsoc 需要大于 2us，其中 Tprstn 软件可通过配置 ADC_CON1 寄存器中 ADC_RSTN_CNT 来配置时间长短，如果在 adc_rstn 未立起来时，置位 SOC，adc_soc 会等 adc_rstn 为高后立起来。adc_soc 立起来后可启动 ADC 开始转换。

23.3.2 ADC 转换时序

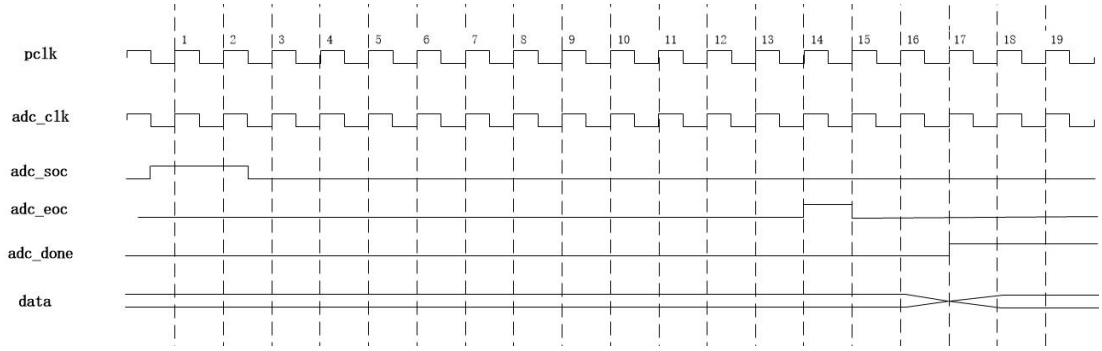


图 23-2 ADC 单次转换时序

上图为 ADC 时钟不分频单次转换时序图，每次由软件配置 ADC_CON 寄存器的 SOC 位为 1 来启动一次 ADC 转换。

由于 EOC 信号来自于 ADC 模块，需要同步，所以 ADC_DONE 将在 EOC 之后两个 pclk 时钟周期后才置起。

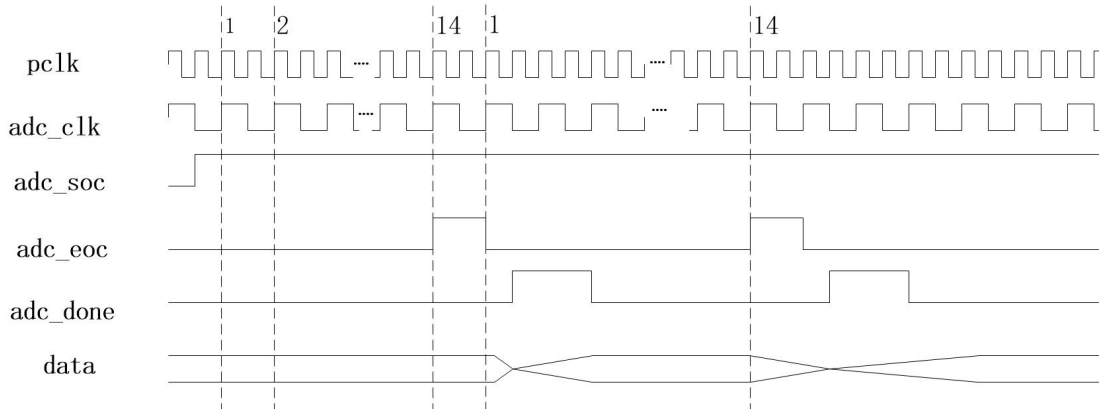


图 23-3 ADC 连续转换时序

上图为 ADC 时钟为 pclk 二分频时的连续转换时序图，软件通过 ADC_CON 寄存器配置 ADC 转换模式为连续转换模式后，启动一次 SOC 之后，ADC 模块会自动开始连续转换，软件无需配置 ADC_CON 寄存器的 SOC 位为 1 去启动 ADC 转换，在每次转换完成之后会自动开始下一次的转换。

23.4 用户操作

从以上时序图看出，无论是单次转换或是连续转换，每次使能 ADC 后（ADCON0 寄存器的 AD_EN 置 1），都需要软件延时 2us，以等待 ADC 内部参考源稳定。然后再启动 ADC 转换。

ADC 有三种转换模式：

23.4.1 ADC 单次转换模式

1）写 ADCIOE 寄存器，使能对应输入通道。

2）软件写 ADCON1 寄存器内的 ADC_MODE 为 00 选择单次转换模式,配置 RST_CNT

（默认为 1）以及 ADC 时钟分频 ADC_PRESCALE 配置 ADC。

3）软件写 ADCON0 寄存器内的控制位（SAIN、ADC_EN）配置 ADC。

3）软件写 ADCON0 寄存器的 ADC_START 位为 1，启动一次转换，转换完成后 ADC 产生 A/D 转换完成标志（ADSTAT 寄存器的 DONE 置 1）。软件读 ADC_DAT 寄存器后，ADSTAT 寄存器的 DONE 会被自动清 0。

23.4.2 TMR2 定时触发 A/D 转换模式

1）软件写 ADCON1 寄存器内的 AD_MODE 为 01 选择 TMR2 定时触发 AD 转换模式，配置 RST_CNT（默认为 1）以及 ADC 时钟分频 ADC_PRESCALE 配置 ADC。

2）软件写 ADCON0 寄存器内的控制位（SAIN、ADC_EN）配置 ADC。

3）软件配置 TM20 并使能 TMR2，TMR2 每次比较定时匹配都后会自动触发一次 A/D 转换，转换完成后 ADC 产生 A/D 转换完成标志（ADSTAT 寄存器的 DONE 置 1）。软件读 AD_DAT 寄存器后，ADSTAT 寄存器的 DONE 会被自动清 0。

23.4.3 连续转换模式

1）写 ADCIOE 寄存器，使能对应输入通道。

2）软件写 ADCON1 寄存器内的 ADC_MODE 为 00 选择单次转换模式,配置 RST_CNT（默认为 1）以及 ADC 时钟分频 ADC_PRESCALE 配置 ADC。

3）软件写 ADCON0 寄存器内的控制位（SAIN、ADC_EN）配置 ADC。

4）软件写 ADCON0 寄存器的 ADC_START 位为 1，启动连续转换，若要停止连续转换，只要选择其他转换模式或清除 ADC_START 即可。

注：1. ADCON0 的 AD_DMA 置 1 使能 DMA 传输时，无论是哪种转换模式，每次转换完成后就会触发一次 DMA 传输。

2. 若 ADCIE 寄存器中 INT_EN 使能，且中断使能打开（通过中断使能设置寄存器 VIC_ISER 配置），每次转换完成时还会产生中断信号。

23.5 寄存器列表

地址	寄存器	描述	备注
0x4000_6000	ADC_CON0	ADC 控制寄存器 0	ADC_CON0 说明
0x4000_6004	ADC_CON1	ADC 控制寄存器 1	ADC_CON1 说明
0x4000_6008	ADC_STAT	ADC 状态寄存器	ADC_STAT 说明
0x4000_600C	ADC_DATA	ADC 结果寄存器	ADC_DATA 说明
0x4000_6010	ADC_IE	ADC 中断使能寄存器	ADC_IE 说明
0x4000_6014	ADC_IOE	ADC 通道输入使能寄存器	ADC_IOE 说明

23.6 寄存器描述

23.6.1 ADC 控制寄存器 0 ADC_CON0

（地址： 0x4000_6000）

位域	类型	复位	名称	描述
[31:22]	--	--	--	--
[21：16]	R/W	0x0	OFFSET	**ADC 补偿码** [21] :符号位 [20:16] 5bit 补码 只读：当 ADC_EN 和 DISH 都置位时，读到 5bit 补偿码。负数取反加一。
[15:14]	--	--	--	--
[13]	R/W	0	GCMP	**ADC 内部延时选择，控制比较的增益** 0b: 更低的 adc 比较增益 1b: 更高的 adc 比较增益
[12]	R/W	0	DISH	**ADC 短接控制** 0b: 正常模式 1b: VIP 和 VIN 内部短接，用于 offset 校正
[11:9]	--	--	--	--
[8：4]	R/W	0xIF	SAIN	**ADC 定时触发源选择** 00000b: AIN0 00001b: AIN1 … … 00101b: AIN5
[3:2]	R/W	0	RESO	**ADC 分辨率配置** 00b: 12bit 01b: 10bit 10b: 8bit 11b: 6bit
[1]	R/W	0	ADC_START	**ADC 开始启动** 0b: adc 不启动 1b: 当 adc 为单次转换模式时，写 1 开始启动 ADC，当 SOC 立起来后，ADC_START 自动清零；当 adc 连续转换模式时，写 1 开始启动 ADC，SOC 立起来，不自动清零。

[0]	R/W	0	ADC_EN	**ADC 使能控制** 0b: ADC 禁止 1b: ADC 使能

23.6.2 ADC 控制寄存器 1 ADC_CON1

（地址： 0x4000_6004）

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:8]	R/W	0x02	ADC_PRESCALE	**adc_clk 分频控制** 取值范围在 1~0xff，不能为 0。 adc_clk = pclk/（ADC_PRESCALE + 1）
[7:4]	R/W	0x1	ADC_RSTN_CNT	**adc RSTN 信号控制** Tprstn 时间配置，建议范围在 1~f。 Tprstn = （ADC_RST_CNT+1）* adc_clk
[3]	--	--	--	--
[2:1]	R/W	0	ADC_MODE	- **adc 模式选择** 00b：adc one shot 01b：adc timer2 MR0 match triggle 10b：adc continue 11b：保留
[0]	R/W	0	DMA_MODE	**adc DMA_MODE 控制** 0b：DMA_MODE 禁止 1b：DMA_MODE 使能

23.6.3 ADC 状态寄存器 ADC_STAT

（地址： 0x4000_6008）

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	DONE	**ADC 转换完成标志位** 0b: ADC 转换未完成 1b: ADC 转换完成

23.6.4 ADC 中断使能寄存器 ADC_IE

（地址： 0x4000_6010）

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R/W	0	INT_EN	**ADC 中断使能控制** 0b: ADC 中断禁止 1b: ADC 中断使能

23.6.5 ADC 通道输入使能寄存器 ADC_IOE

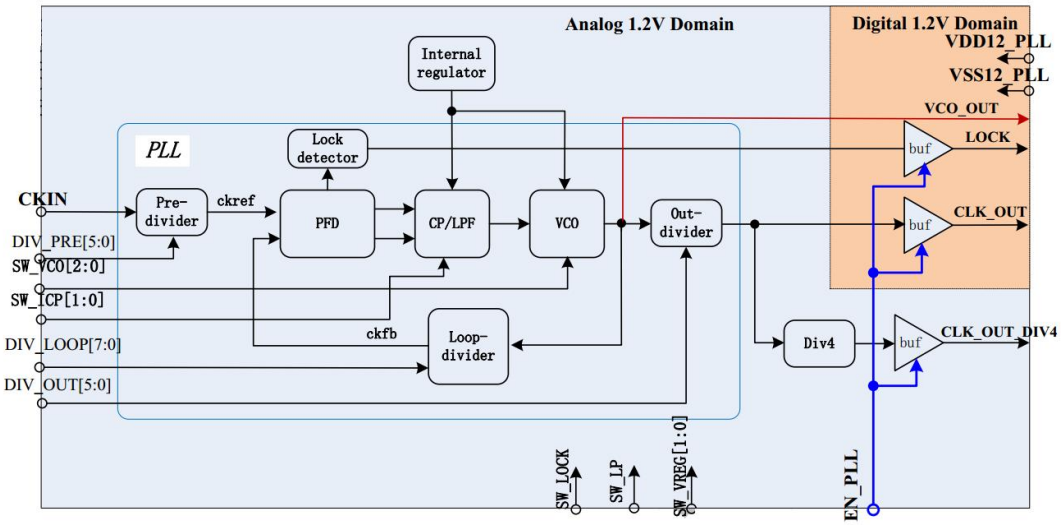
（地址： 0x4000_6014）

位域	类型	复位	名称	描述
[31:6]	--	--	--	--
[5:0]	R/W	0	AIN_EN	**ADC 输入通道使能控制** 6bit 分别对应六个通道 ain0/ ain1/ ain2/ ain3/ ain4/ ain5 AIN_EN [0]: 0b: ADC 输入通道 ain0 禁止 1b: ADC 输入通道 ain0 使能

24 模拟杂项控制

24.1 锁相环 PLL

24.1.1 PLL 框图



24.1.2 PLL 时钟公式

$$f_{CLK\_OUT} = \frac{f_{CKIN} \times N_{DIV\_LOOP}}{N_{DIV\_PRE} \times N_{DIV\_OUT}}$$

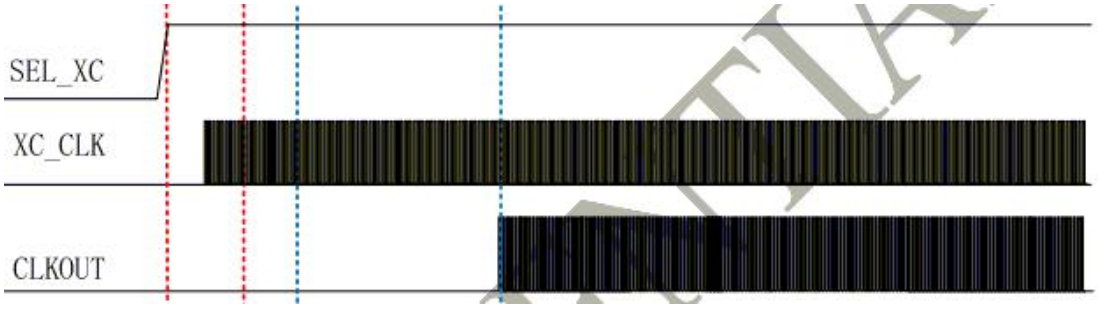
24.1.3 PLL 时钟作为系统时钟应用例子

输入 24MHz 时钟，输出 160MHz PLL 时钟：

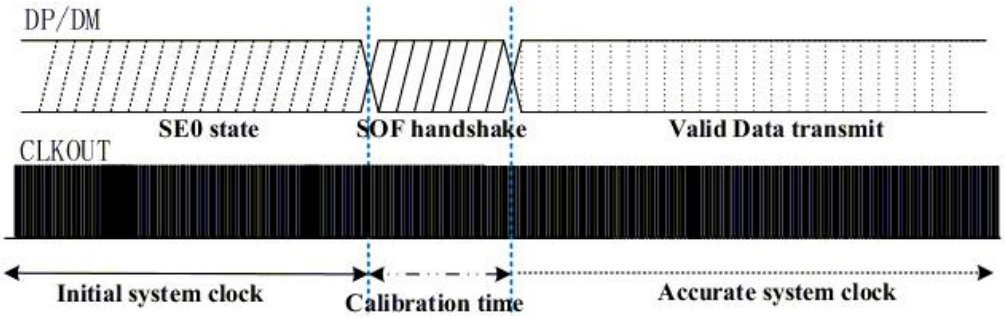
1. 配置 PLL_Config 寄存器的 DIV_PRE=0x12，DIV_LOOP=0xA0，DIV_OUT=0x2，配置输出 PLL 频率 160MHz。
2. 配置 PLL_CON 寄存器的 SW_VCO=0x5，SW_LOCK=0x0，其它默认值即可。
3. 配置 PLL_CON 寄存器的 EN=0x1，使能 PLL,可以通过判断 PLL_CON 寄存器的 bit31 LOCK 来确认 PLL 时钟是否稳定。
4. 等 PLL 时钟稳定，配置系统时钟控制寄存器 SYSCON 的 SYSCLK_SEL=0x2，系统时钟选择 PLL_CLK。

24.2 USB 波形图

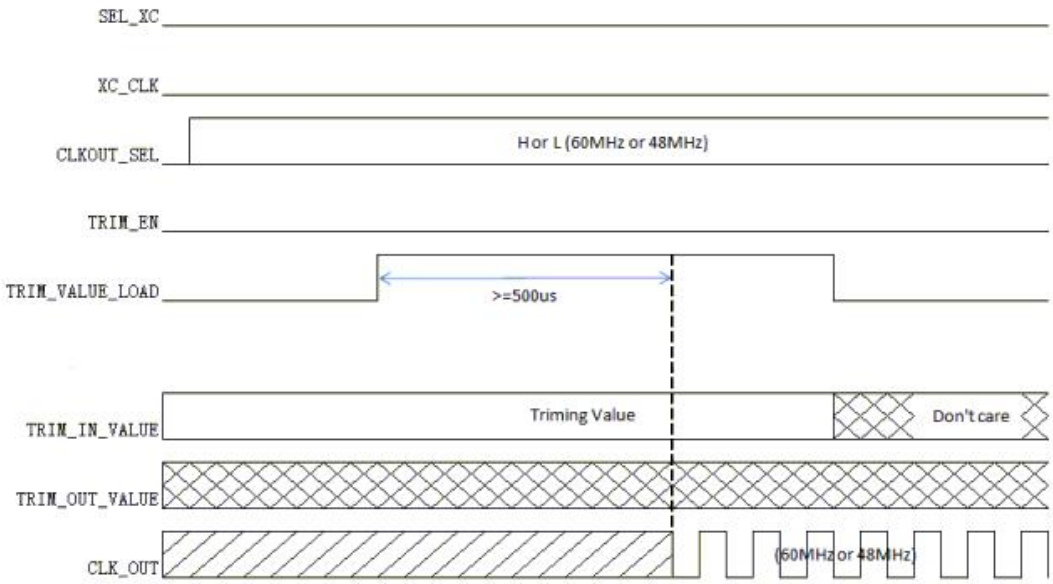
1、SEL_XC=1，外部12MH作为时钟参考源



2、SEL_XC=0，选择由主机发出的 1KHz SOF handshake 信号作为参考时钟



3、软件写 trim 值，USB CLK_OUT



24.3 寄存器列表

地址	寄存器	描述	备注
0x4000_5800	IOSC_32K_CON	内部 32K RC 震荡控制寄存器	IOSC_32K_CON 说明
0x4000_5804	IOSC_32K_STATE	内部 32K RC 震荡状态寄存器	IOSC_32K_STATE 说明
0x4000_5808	IOSC_24M_CON	内部 24M RC 震荡控制寄存器	IOSC_24M_CON 说明
0x4000_580C	OSC24M_TRIM	内部 24M RC 震荡校验寄存器	OSC24M_TRIM 说明
0x4000_5810	PLL_CON	PLL 控制寄存器	PLL_CON 说明
0x4000_5814	PLL_Config	PLL 配置寄存器	PLL_Config 说明
0x4000_5818	USBCON	USB 控制寄存器	USBCON 说明
0x4000_581C	USBTRIM	USB 校准寄存器	USBTRIM 说明
0x4000_5820	USBDEBUG	USB 调试寄存器	USBDEBUG 说明
0x4000_5824	FPMUCON	FLASH PMU 控制寄存器	FPMUCON 说明
0x4000_5828	FPMUSTAT	FLASH PMU 状态寄存器	FPMUSTAT 说明
0x4000_582C	PMUCON0	PMU 控制寄存器 0（不开放）	PMUCON0 说明
0x4000_5830	PMUCON1	PMU 控制寄存器 1（不开放）	PMUCON1 说明
0x4000_5834	CRYST32K_CON	外部 32K 晶振控制寄存器	CRYST32K_CON 说明
0x4000_5838	CRYST32M_CON	外部 32M 晶振控制寄存器	CRYST32M_CON 说明
0x4000_583C	TRIM_CTR	校准值控制寄存器(不开放）	TRIM_CTR 说明

24.4 模拟寄存器描述

24.4.1 内部 32K RC 震荡控制寄存器 IOSC_32K_CON

（地址： 0x4000_5800）

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:8]	R/W	0x02	SW_OSC	**RC32K 振荡器频率校准值** 默认值：20h，该值越大，输出频率越大。
[7:4]	--	--	--	--
[3:2]	R/W	0x3	SW_FREQ	**RC32K 振荡器频率选择** 默认值：11b 00b：输出 4KHz 的时钟频率 01b：输出 16KHz 的时钟频率 10b：输出 8KHz 的时钟频率 11b：输出 32KHz 的时钟频率
[1:0]	R/W	0x1	SW_IOP	**RC32K 振荡器静态电流控制** 默认值：01b，该值越大，静态电流越大。

24.4.2 内部 32K RC 震荡状态寄存器 IOSC_32K_STATE

（地址： 0x4000_5804）

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	STATE_OK	**IOSC_32K 时钟状态位** 只读 0b：32k 时钟未就绪 1b：32k 时钟就绪

24.4.3 内部 24M RC 震荡控制寄存器 IOSC_24M_CON

（地址： 0x4000_5808）

位域	类型	复位	名称	描述
[31:6]	--	--	--	--
[5:4]	R/W	0x01	CSEL	**RC oscillator 电容选择** **00b: 80%*C. frequency of oscillator +20%** **01b/10b: 100%*C. default** **11b: 120%*C. frequency of oscillator -20%**
[3:1]	R/W	0x6	REG_FSEL	频率选择 000b：4MHz 001b：6Mhz 010b：8Mhz 011b： 9.6Mhz 100b： 12Mhz 101b： 16Mhz 110b： 24Mhz 111b： 48Mhz
[0]	R/W	0x1	EN	**IOSC_24K 时钟使能位**

24.4.4 内部 24M RC 震荡校验寄存器 OSC24M_TRIM

（地址： 0x4000_580C）

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15]	R	0x0	TRIM_DONE	**AUTO TRIM 完成标志位** 只读，自动校准完成标志。
[14:10]	--	--	--	--
[9:5]	R/W	0x10	RCI	**时钟频率粗调校准值** 写: 当写 OSC24M_TRIM [31 :24]=0xac，写该位才有效。 读: 读回用户写入的值，不一定是有效的值。 每阶粗调+1.66%: 00000b: 更低的频率 … …

				11111b: 更高的频率
[4:1]	R/W	0x8	RCV	**时钟频率微调校准值** 写： 当写 OSC24M_TRIM [31 :24]=0xac，写该位才有效。 读： 读回用户写入的值，不一定是有效的值。 每阶微调-0.13%： 0000b: 更高的频率 … … 1111b: 更低的频率
[0]	R/W	0x0	TRIM_EN	**OSC24 时钟 trim 使能位** 0b：正常模式 1b：自动校准启动

24.4.5 PLL 控制寄存器 PLL_CON

（地址： 0x4000_5810）

位域	类型	复位	名称	描述
[31]	R	0x0	LOCK	**PLL 就绪状态位** 只读，且只有在 SW_LOOK=0 时，该状态位有效。 0b：PLL 时钟未稳定 1b：PLL 时钟已稳定
[30:12]	--	--	--	--
[11:10]	R/W	0x3	SW_VREG	**内部电压输出控制** SW_VREG[1：0] / VCC12_PLL (V) 00b / 1.00±0.100V 00b~01b / 1.05±0.105V 01b~10b / 1.10±0.110V 10b~11b / 1.20±0.120V
[9:8]	R/W	0x1	SW_ICP	**PLL charge pump 电流控制** 00b：1uA 01b：2uA （默认值） 10b：3uA 11b：4uA
[7]	--	--	--	--
[6：4]	R/W	0x2	SW_VCO	**PLL VCO 输出频率调整** SW_VCO[2：0] / 内部 VCO 输出频率（MHz） 001b / 60~120 010b / 100~200 011b / 160~280 100b / 220~360 101b / 300~440

				110b 380~520 111b 460~600
[3]	--	--	--	--
[2]	R/W	0x1	SWLP	**PLL 低功耗模式控制** 0b：正常模式 1b：PLL 低功耗模式
[1]	R/W	0x1	SW_LOCK	**PLL LOCK 侦测控制** 0b：当 PLL 稳定时，LOCK 标志位会立起 1b：bypass PLL lock 侦测，当 PLL 稳定时，LOCK 标志位不会立起
[0]	R/W	0x0	EN	**PLL 时钟使能位**

24.4.6 PLL 配置寄存器 PLL_Config

（地址： 0x4000_5814）

位域	类型	复位	名称	描述
[31:22]	--	--	--	--
[21:16]	R/W	0x1	DIV_OUT	**PLL output divider** 取值范围 1~63，修改配置前要先除能 PLL。
[15:8]	R/W	0x1	DIV_LOOP	**PLL LOOP divider** 取值范围 1~255，修改配置前要先除能 PLL。
[7:6]	--	--	--	--
[5:0]	R/W	0x1	DIV_PRE	**PLL 输出频率预分频** 取值范围 1~63，修改配置前要先除能 PLL

$$f_{CLK\_OUT} = \frac{f_{CKIN} \times N_{DIV\_LOOP}}{N_{DIV\_PRE} \times N_{DIV\_OUT}}$$

24.4.7 USB 控制寄存器 USBCON

（地址： 0x4000_5818）

位域	类型	复位	名称	描述
[31:22]	--	--	--	--
[21：16]	R/W	0x28	REG_OSC_ADJ	**UBS 内部 OSC 频率控制** 默认值 REG_OSC_ADJ[5：0]=0x28，调整 usb 内部 OSC 频率
[15:9]	--	--	--	--
[8]	R/W	0x1	DMPU_LO	**DM 较低电阻拉高控制** 0b：较低的上拉电阻不与 DM 连接 ，表示 USB 总线在忙状态且使用较高的上拉电阻。 1b：较低的上拉电阻与 DM 连接，表示如果 USB 在 idle 状态，使用较低上拉电阻。
[7]	R/W	0	DPPU_LO	**DP 较低电阻拉高控制** 0b：较低的上拉电阻不与 DP 连接 ，表示 USB 总线在忙状态且使用较高的上拉

				电阻。 1b：较低的上拉电阻与 DP 连接，表示如果 USB 在 idle 状态，使用较低上拉电阻。
[6]	R/W	0	DMPU	**DM pull up 控制** 0b：DM 不拉高 1b：DM 拉高
[5]	R/W	0	DPPU	**DP pull up 控制** 0b：DP 不拉高 1b：DP 拉高
[4]	R/W	0	PULLDN	**USB 管脚 pull down 控制** 0b：管脚不拉低 1b：管脚拉低（r=15k）
[3]	R/W	0	PLL_EN	**USB 内部 OSC 和 PLL 控制** 0b：禁止 USB 内部 OSC 和 PLL 1b：使能 USB 内部 OSC 和 PLL
[2]	R/W	0	CLKOUT_SEL	**USB CLKOUT 输出频率选择** 0b：输出 48MHz 1b：输出 60MHz
[1]	R/W	0	SEL_XC	**USB CLKOUT 参考时钟选择** 0b：选择由主机发出的 1KHz SOF handshake 信号作为参考时钟 1b：选择外部 12MHz 时钟作为参考时钟，使用前需要使能外部 crystal 24MHz 时钟。
[0]	R/W	0	SPEED	**USB 数据速率控制** 0b：低速模式 1b：全速模式

24.4.8 USB 校准寄存器 USBTRIM

（地址： 0x4000_581C）

位域	类型	复位	名称	描述
[31:29]	--	--	--	--
[28]	R	0	TRIM_NEAR_LOCK	**USB_CLK 就绪标志位** 只有当 SEL_XC=0，即选择由主机发出的 1KHz SOF handshake 信号作为参考时钟时，标志位才有机会立起来，表示 USB_CLK 就绪状态。
[15:4]	R/W	0x0	TRIM_IN_VALUE	**载入 trim 值** **当 TRIM_VALUE_LOAD=1 时，写入才有效。参考值 TRIM_IN_VALUE[11:0] = 0x47C。**
[3:2]	--	--	--	--
[1]	R/W	0x0	TRIM_VALUE_LOAD	**m 值载入控制** **0b：trim 值载入禁止** **1b：trim 值载入使能，可写入 TRIM_IN_VALUE 值，使 USB_CLK 输出 48MHz 时钟，参考值为 0x47C。**
[0]	--	--	--	--

24.4.9 USB 调试寄存器 USBDEBUG

（地址： 0x4000_5820）

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0	DEBUG	**调试控制** USB 调试值配置。

24.4.10 FLASH PMU 控制寄存器 FPMUCON

（地址： 0x4000_5824）

位域	类型	复位	名称	描述
[31:8]	--	--	--	--
[7]	R/W	1	ENOSCB	**FLASH PMU 内部 OSC 控制** 0b：使能 OSC 1b：禁止 OSC
[6]	R/W	1	ENVDHB	**高电压检测控制信号** 0b：高电压检测使能 1b：高电压检测禁止
[5]	R/W	1	ENVDLB	**低电压检测控制信号** 0b：低电压检测使能 1b：低电压检测禁止
[4]	R/W	0	PSB3	**REG3 standby mode 控制**
[3]	R/W	0	PSB2	**REG2 standby mode 控制**
[2]	R/W	0	PSB1	**REG1 standby mode 控制**
[1]	R/W	0	ENREG3B	**REG3 使能控制** 0b：REG3 使能 1b：REG3 禁止
[0]	R/W	0	ENREG2B	**REG2 使能控制** 0b：REG2 使能 1b：REG2 禁止

24.4.11 FLASH PMU 状态寄存器 FPMUSTAT

（地址： 0x4000_5828）

位域	类型	复位	名称	描述
[31:2]	--	--	--	--
[1]	R	0	VDH	**高电压检测标志位** 当 ENVDHB=0，检测到 VCC>6.0V 时，高电压检测标志位立起，即 VDH=1。
[0]	R	0	VDL	**低电压检测标志位** 当 ENVDLB=0，检测到 1.33V<VCC<1.54V 时，低电压检测标志位立起，即 VDL=1。

24.4.12 外部 32K 晶振控制寄存器 CRYST32K_CON

（地址： 0x4000_5834）

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:8]	R/W	0x34	REG_XTAL32K_AON	**32K 晶振工作电流及增益配置控制** REG_XTAL32K_AON[2:0]：晶振工作电流控制 100b 为默认值 REG_XTAL32K_AON[3]： 晶振 AGC 功能 bypass 控制 0b：BYPASS 禁止 1b：BYPASS 使能 REG_XTAL32K_AON[5：4]：晶振内部 RF 电阻选择 10b 为默认值 REG_XTAL32K_AON[7：6]：晶振 AGC 增益控制 01b 为默认值
[1]	R/W	0x0	XTAL_BYP	**32K 晶振 BYPASS 控制** 0b：BYPASS 禁止 1b：BYPASS 使能
[0]	R/W	0x0	XTAL_EN	**32K 晶振使能控制** 0b：32K 晶振禁止 1b：32K 晶振使能

24.4.13 外部 32M 晶振控制寄存器 CRYST32M_CON

（地址： 0x4000_5838）

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:8]	R/W	0x64	REG_XTAL32M_AON	**32M 晶振工作电流及增益配置控制** REG_XTAL32M_AON[2:0]：晶振工作电流控制 100b 为默认值 **REG_XTAL32M [2:0]** / Typical operating current (uA) / Recommended frequency range of the crystal (MHz) 3b'011 / 50 / 4M 3b'101 / 100 / 12M 3b'110 / 300 / 32M REG_XTAL32M_AON[3]： 晶振 AGC 功能 bypass 控制 0b：BYPASS 禁止 1b：BYPASS 使能 REG_XTAL32M_AON[5：4]：晶振内部 RF 电阻选择 10b 为默认值 REG_XTAL32M_AON[7：6]：晶振 AGC 增益控制 01b 为默认值
[1]	R/W	0x0	XTAL_BYP	**32M 晶振 BYPASS 控制** 0b：BYPASS 禁止

				1b：BYPASS 使能
[0]	R/W	0x0	XTAL_EN	**32M 晶振使能控制** 0b：32K 晶振禁止 1b：32K 晶振使能

25 QSPI

25.1 概述

QSPI 是一种同步串行外设接口，对数据进行串并转换，通过主从方式的方式实现 MCU 和外围器件的数据通讯。QSPI 接口使用 6 个引脚，其中串行数据输入输出线 MOSI / SIO0 和 MISO / SIO1, SIO2, SI03，时钟线 SCK，从选择线 CS。QSPI 支持主机模式，它使用 CS 和 SCK 信号控制数据流，以控制数据通信的开始/结束和数据采样率。

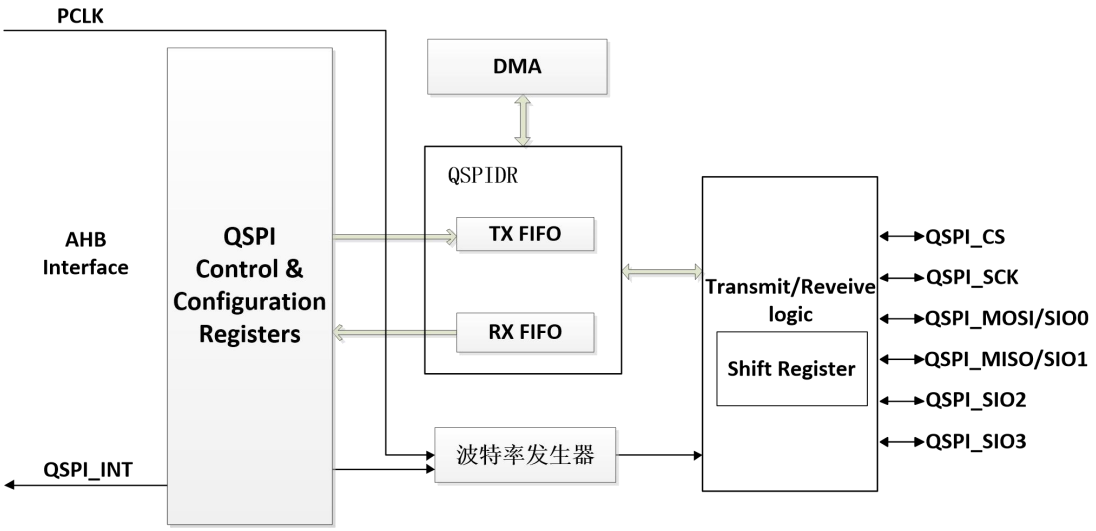


图 25-1. QSPI 方框图

基本特性如下：

- 一帧数据传输 4~32 位可编程
- 主机最高传输速率为系统时钟（PCLK）二分频
- 最大 16 级接收 FIFO 缓冲队列
- 最大 16 级发送 FIFO 缓冲队列
- 可编程传输速率
- MSB first shift
- 支持 DMA 接口
- 可编程串行接口操作模式

QSPI 支持两种串行接口操作模式:

- Motorola 模式 - 全双工串行协议，串行时钟相位和极性有四种可能的组合，QSPI 空闲或关闭时，CS 信号保持为高。
- National Microwire 模式 – 半双工串行协议，主机通过控制字开始一次传输。

QSPI 支持基于 Motorola 模式的 SPI 增强协议：

- 数据阶段支持 Dual、Quad 模式操作
- 可编程指令，地址长度，等待周期和数据帧长
- 可编程跳过地址和指令阶段

25.2 通讯信号

QSPI 通讯所用引脚描述如下：

CS：

SPI 串行通讯从机片选控制。每一个从机都有一个独立的片选信号控制。

SCK：

SPI 串行通讯时钟，时钟信号只有 SPI 主机才能发出，所有从机的时钟只能输入来自主机的时钟信号。

MISO/SIO0：

SPI 串行通讯数据输入(数据输入输出 0)。

MOSI/SIO1：

SPI 串行通讯数据输出(数据输入输出 1)。

SIO2：

数据输入输出 2。

SIO3：

数据输入输出 3。

注 1: SIO0 & SIO1 用于标准单线/双线传输

注 2: SIO0 ~ SIO3 用于四线传输

25.3 数据传输

当满足一下条件，主机开始启动数据传输：

- QSPI 使能(QSPIEN set)
- 使能了从机选择信号(SER set)
- TX FIFO 中的数据量大于 TXFTLR.TFTHR 设置级别

在开始传输数据时，状态寄存器(SR)中的忙碌标志(BUSY)被置位。在新的串行传输之前，需要等待 BUSY 标志被清除。当数据写入传输 FIFO 时，不设置 BUSY 标志位，只有在选择了从机并且正在进行传输时才置位。在轮询忙状态之前，应该首先轮询 TFE 状态位置位。

TX/RX FIFO 的宽度是固定在 32 位，它表明串行传输可以在长度为 1 到 32 位。当写入 TX FIFO 缓冲区时，小于 32 位的数据写入时，数据帧必须右对齐；接收移位寄存器按照自动右对齐移动到 RX FIFO 中。

FIFO 缓冲区中的每个数据条目都包含一个数据帧。它是不可能存储多个数据帧在一个单一的 FIFO 位置，例如，不能在一个 FIFO 位置存储两个 8 位数据帧。如果需要一个 8 位数据帧，FIFO 条目的高 8 位在串行移位器传输数据时将会被忽略。

25.4 工作模式

25.4.1 Motorola 模式

当 SPI 处于 Motorola 模式时，SPI 处于全双工的工作状态，并且根据时钟极性(SCPOL)/相位(SCPH)的设定，分为 4 种不同的帧格式。

- 时钟极性 — SCPOL

当时钟极性为零，SCK 线空闲状态为低电平。当时钟极性位为高，SCK 线空闲状态为高电平。

- 时钟相位 — SCPH

当时钟相位为零，数据会在第一个 SCK 时钟转换信号跳变时被采样。当时钟相位为高，数据会在第二个 SCK 时钟信号跳变时被采样。

25.4.1.1 模式 00/10（SCPH=0）

当设为模式 00 时(SCPOL=0，SCPH=0)，在空闲状态下 QSPI 时钟保持低电平。主机把从机的片选信号 CS 拉低后，从机发往主机的第一位数据即出现在主机的 MISO 引脚上就绪；主机往 SPI 数据发送寄存器内写入数据后，发往从机的第一位数据也即刻出现在 MOSI 引脚上，之后在每一个时钟上升沿主机和从机各自同步采样接收对方发送过来的数据位并送入内部移位寄存器，在其后的时钟下降沿各自打出下一位数据。当一个 QSPI 数据帧（4~32 位）传输结束后，接收到的数据从内部移位寄存器转存入 QSPI 接收 FIFO，QSPI 时钟恢复到低电平状态。主机可继续往发送寄存器内写入数据进行连续发送，或者通过软件把从机片选信号拉高，结束 QSPI 通讯。

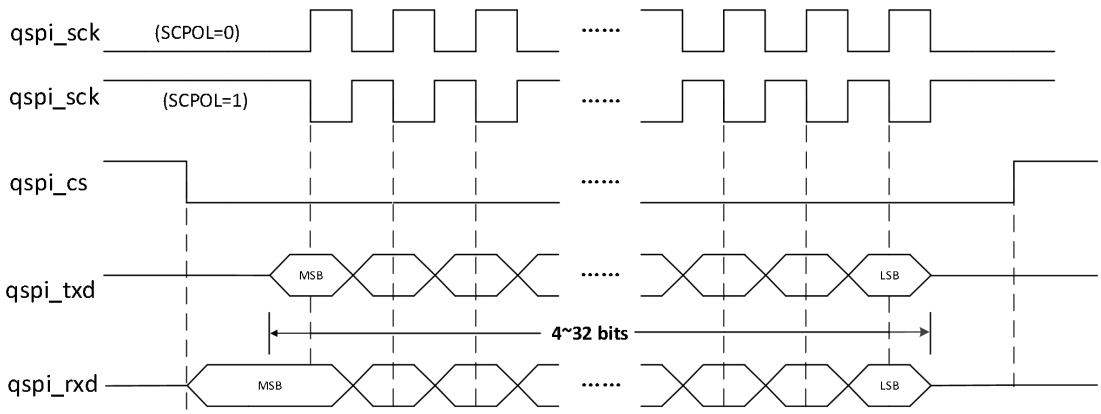


图 25-2 模式 00/01 时序图(SCPH=0)

当 SCPH = 0 时，支持两种不同的连续数据传输模式，通过 CTRLR0.SSTE 的配置选择所需的操作模式。

当 CTRLR0.SSTE 设为 1，QSPI 在连续数据帧传输时，切换 CS 信号，该工作模式如图 25-3 所示。

当 CTRLR0.SSTE 设为 0，QSPI 在连续数据帧传输时保持 CS 为低有效电平，这种工作模式如图 25-4 所示。

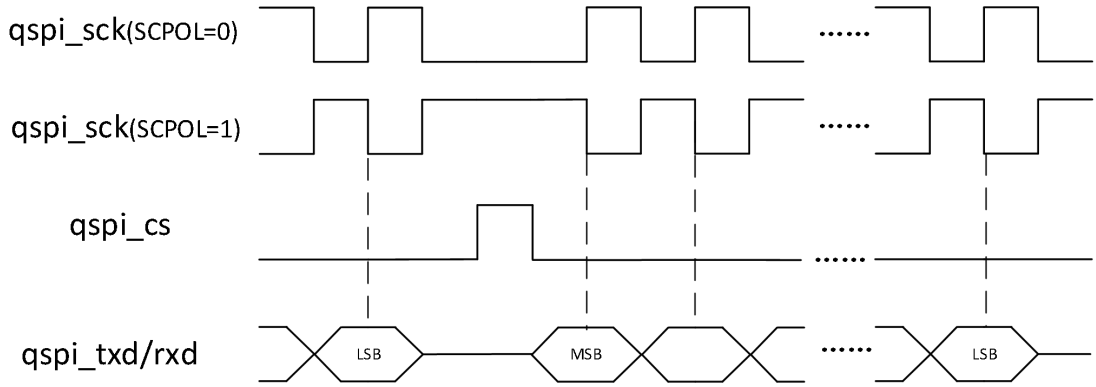


图 25-3 模式 00/01 连续数传输时序图(SSTE=1)

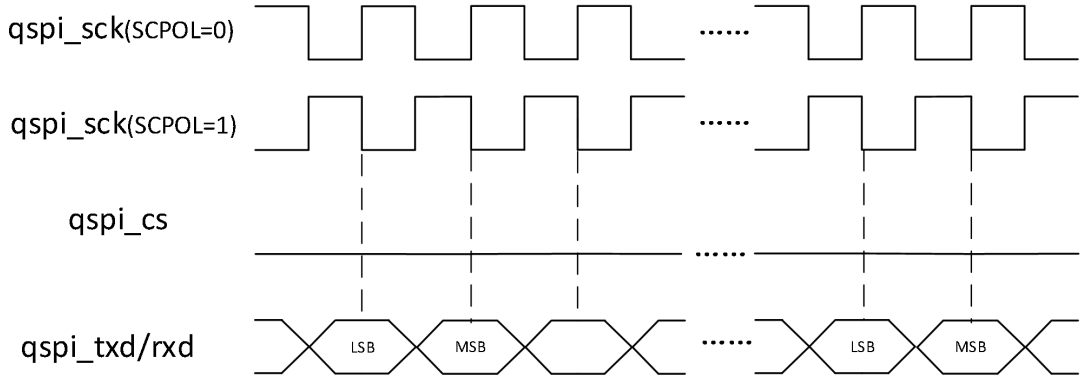


图 25-4 模式 00/01 连续数传输时序图(SSTE=0)

25.4.1.2 模式 01/11（SCPH=1）

当设为模式 01 时(SCPOL=0，SCPH=0)，在空闲状态下 SPI 时钟保持低电平。主机把从机的片选信号 CS 拉低后，从机发往主机的第一位数据即出现在主机的 MISO 引脚上就绪；主机往 SPI 数据发送寄存器内写入数据后，发往从机的第一位数据也即刻出现在 MOSI 引脚上，之后在每一个时钟下降沿主机和从机各自同步采样接收对方发送过来的数据位并送入内部移位寄存器，在其后的时钟上升沿各自打出下一位数据。当一个 SPI 数据帧（4~16 位）传输结束后，接收到的数据从内部移位寄存器转存入 SPI 接收 FIFO，SPI 时钟恢复到低电平状态。主机可继续往发送寄存器内写入数据进行连续发送，或者通过软件把从机片选信号拉高，结束 SPI 通讯。

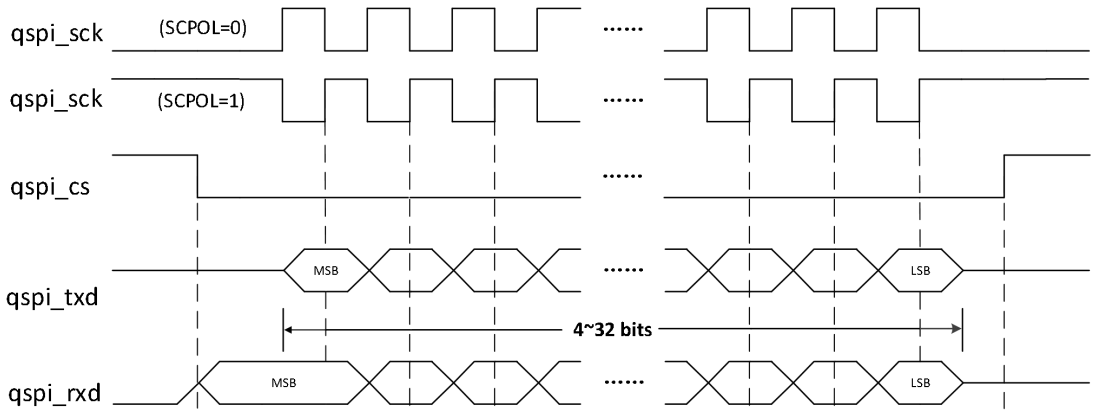


图 25-5：模式 01/11 时序图(SCPH=1)

连续数据帧的传输方式与单帧相同，下一帧的 MSB 直接在当前帧的 LSB 之后。从选择信号 CS 被保持低有效电平的持续时间，如下图 25-6 所示。

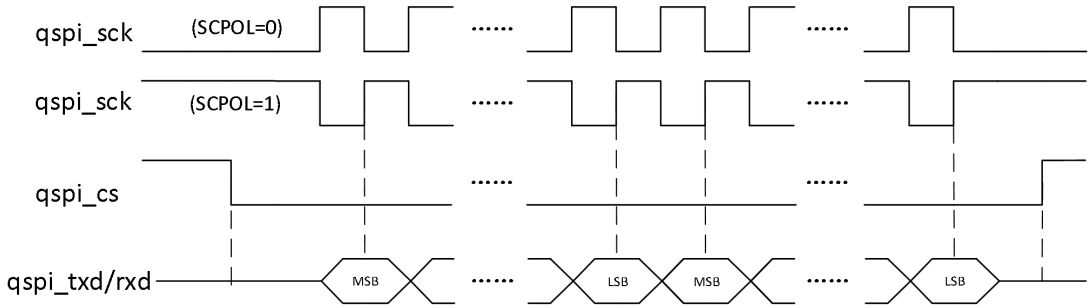


图 25-6：模式 01/11 连续数传输时序图

25.4.2 National Microwire 模式

当 QSPI 被配置为 Microwire，SPI 处于半双工的工作状态，在空闲状态下 SPI 时钟处于低电平，片选信号 CS 处于高电平，一旦发送 FIFO 内被写入了数据，则主机拉低 CS 信号，半个串行时钟周期之后，控制帧的第一个比特被发送到 MOSI 线。控制帧的长度可选 1~16 位的范围内。在控制帧传输期间，MISO 为高阻抗没有数据。

数据字的方向由 MWCR 中的 MDD 位控制。当 MDD=0 时，这表明 QSPI 从外部从接收数据。控制字的 LSB 被传输后的一个时钟周期，从外设响应一位 dummy，然后是数据帧，它的长度可以是 4~32 位。从选择信号在传输期间保持有效低电平，并在数据传输后一个半时钟周期后变为无效电平。图 25-7 显示了从外部串行从器读取单个数据的时序图。

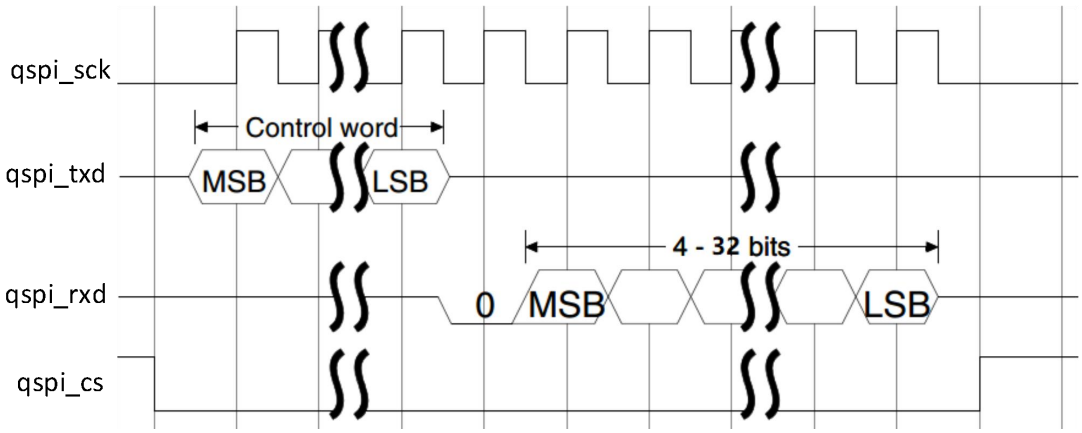


图 25-7 National Microwire 模式单帧数据接收时序图

Microwire 的连续传输可以是序列或非序列，可通过 MWMOD 控制。
图 25-8 显示了采用非序列方式读取多个数据的时序图，下一个传输的控制字紧接在当前数据字的 LSB 之后。

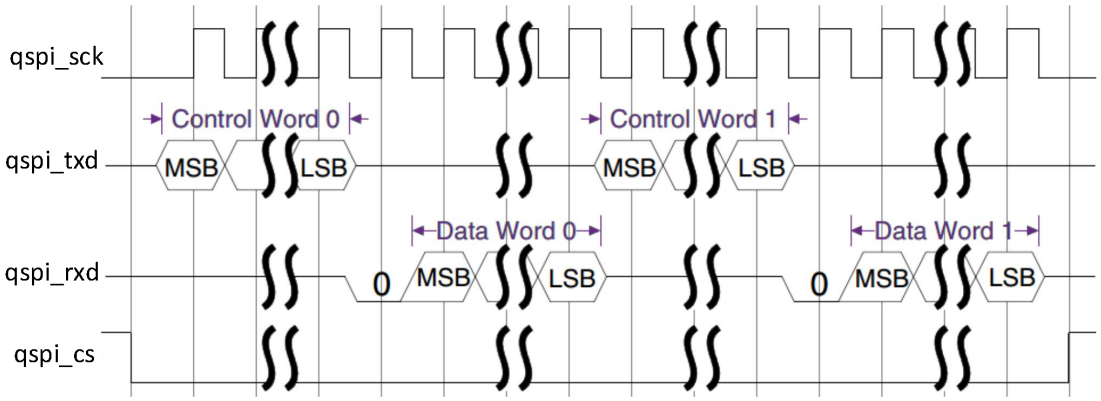


图 25-8 National Microwire 模式非序列连续接收时序图

在序列方式传输期间，只需要发送一个控制字就可以连续读取多个数据，直到当前接收到的数据等于 CTRLR1+1，QSPI 终止传输。图 25-9 显示了采用序列方式读取两个个数据的时序图。

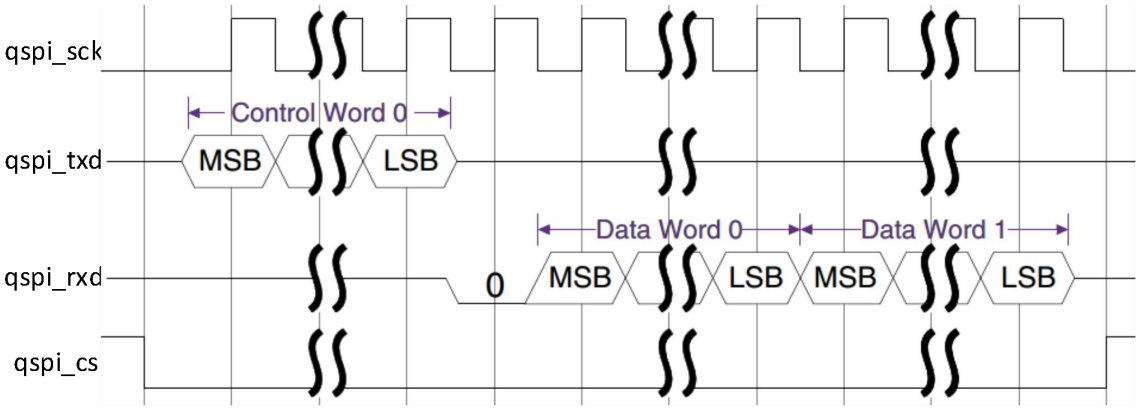


图 25-9 National Microwire 模式序列连续接收时序图

当 MDD =1 时，这表明 QSPI 将数据传输到外部串行从机。在控制字的 LSB 被传输之后，QSPI 主机立即开始传输数据帧。图 25-10 显示了采用非序列方式连续发送多个数据的时序图，下一个传输的控制字紧接在当前数据字的 LSB 之后。

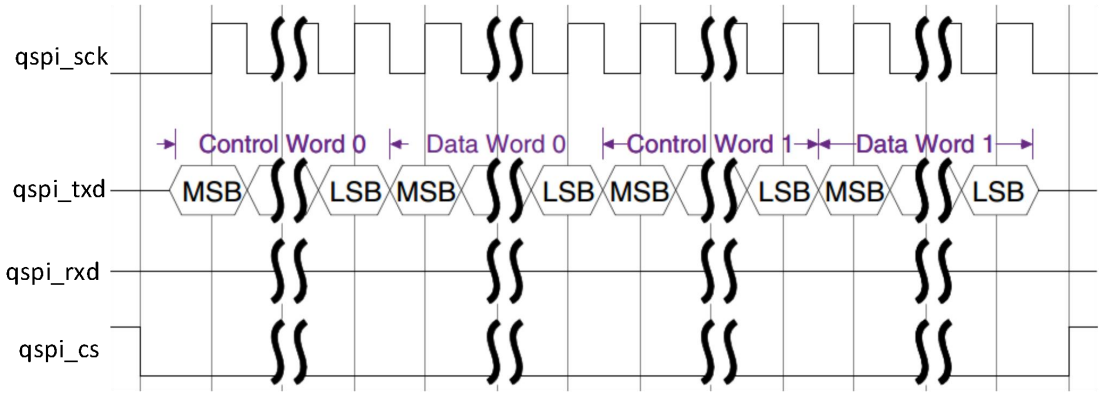


图 25-10 National Microwire 模式非序列连续发送时序图

在序列方式发送期间，只需要发送一个控制字就可以连续读取多个数据，直到当前 FIFO 为空，QSPI 终止传输。

在将数据传输到外部从机，Microwire 接口可以启用握手接口。当 MHS 设置为 1 时，QSPI 串行主机在完成传输或传输下一个控制字以进行连续传输之前检查从设备的就绪状态。图 25-11 显示了一个例子，一个连续的 Microwire 传输与握手接口启用。

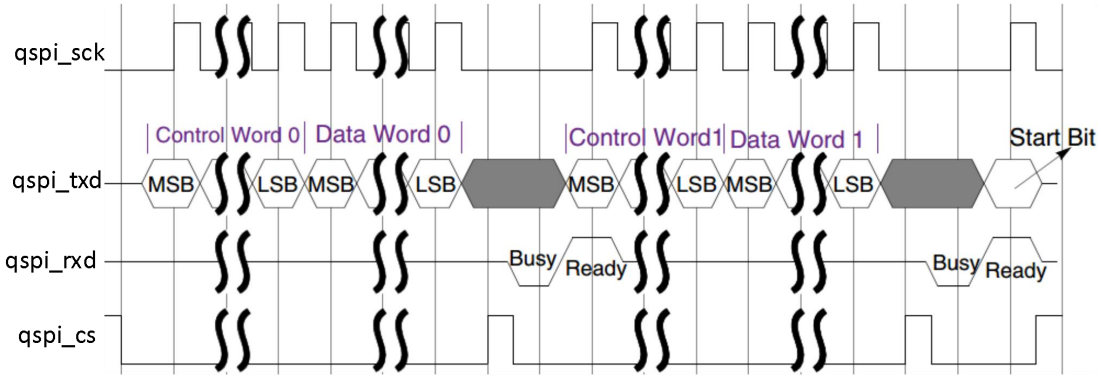


图 25-11 National Microwire 模式非序列连续发送时序图(MHS=1)

在第一个数据字被传输到从机之后，QSPI 轮询 rxd 输入，等待从设备的就绪状态。接收到就绪状态后，QSPI 开始传输下一个控制字。在最后一个数据帧传输完成后，QSPI 在完成传输之前发送一个 start 位来清除从设备的就绪状态。

在序列传输模式，当 MHS 设置为 1 时，QSPI 串行主机在完成传输或传输下一个数据之前检查从设备的就绪状态。

主机只向从机传输控制字 (不后跟数据)，必须设置 MDD 为 1，在发送器 FIFO 中必须只有一笔。不可能连续传输两个控制字，因为 QSPI 将会把第二个控制字视为一个数据字。图 25-12 为只传输一个控制字且使能握手的时序图。

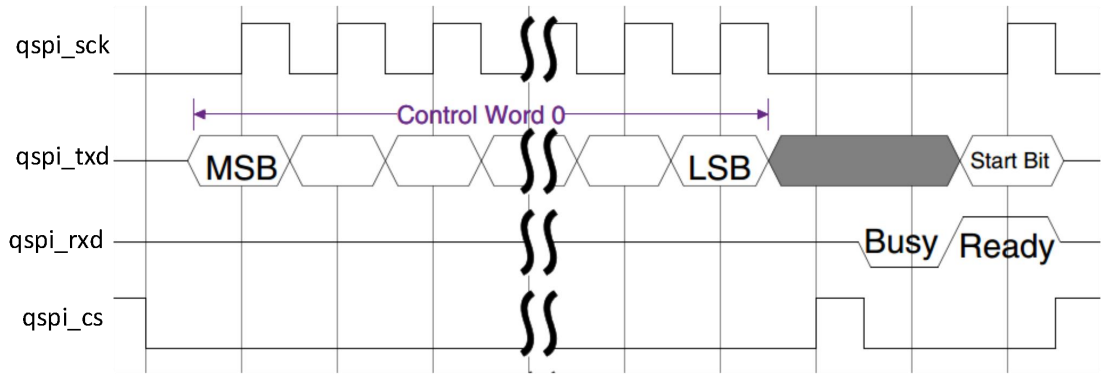


图 26-12 National Microwire 模式仅传输控制字时序图(MHS=1)

25.5 传输模式

QSPI 数据传输可通过 CTRLR0.TMOD 配置不同的模式。对于 Microwire 模式传输，TMOD 配置无效，它是由 MWCR 寄存器控制的。

25.5.1 收发模式 TMOD =00

在收发模式下，收发逻辑均有效。数据传输按照选定的帧格式进行。TX FIFO 中的数据通过 txd 行发送到从机设备，从机通过 rxd 进行应答数据；来自从机设备的数据从接收移位寄存器移动到 RX FIFO 中。

25.5.2 发送模式 TMOD =01

在发送模式下，接收数据无效，不会存入 RX FIFO 中。数据传输按照选定的帧格式进行。TX FIFO 中的数据通过 txd 行发送到从机设备，从机通过 rxd 进行应答数据，接收移位寄存器不将其接收的数据加载到 RX FIFO 中。当进入此模式时，应该屏蔽来自接收逻辑的中断。

25.5.3 接收模式 TMOD =10

在接收模式，传输数据无效。txd 输出在传输期间保持一个恒定的逻辑水平，数据传输按照所选帧格式正常进行。来自从机设备的接收数据从接收移位寄存器移动到 RX FIFO 中。当进入此模式时，您应该屏蔽源自发送逻辑的中断。

25.5.4 EEPROM 读模式 TMOD =11

在此模式下，发送得数据被用来传输一个操作码和/或一个地址到 EEPROM 设备。通常，这需要三个数据帧(8 位操作码后跟 8 位高字节地址和 8 位低字节地址)。在传输操作码和地址期间，接收逻辑不会捕获任何数据(只要 QSPI 在其 txd 上传输数据，rxd 上的数据将被忽略)。QSPI 主机将继续传输数据，直到 TX FIFO 为空。因此，在 TX FIFO 中应该只有足够的数据帧来提供操作码和地址给 EEPROM。如果 TX FIFO 中的数据帧比需要的多，那么读取的数据就会丢失。

当 TX FIFO 变为空时(所有的控制信息已经发送)，接收线上的数据(rxd)是有效的，并存

储在 RX FIFO 中；txd 输出保持在一个恒定的逻辑水平。串行传输继续，直到 QSPI 主机接收到的数据帧数与 CTRLR1.NDF+1 匹配。

25.6 增强型 SPI 模式

QSPI 在 Motorola 模式下，可通过 SPI_FRF 选择模式标准单线 SPI、双线 SPI、四线 SPI 模式。当为该参数选择对双线、四线模式时，txd、rxd 信号的宽度分别更改为 2、4。因此，数据传输不只是单线，从而增加了总体吞吐量。

Motorola 模式串行时钟的极性和相位在这种模式下是有效的，它的工作原理与 Motorola 单线模式下一样，除了 txd、rxd 信号的宽度。

写/读操作模式可以使用 CTRLRO.TMOD 选择，下面详细描述了双线 SPI 和四线 SPI 模式下的读写操作。

25.6.1 写操作

双线 SPI 和四线 SPI 模式写入操作可分为三部分:

- 指令阶段，SPI_CTRLR0.INST_L - 指令长度的可能值为 0、4、8 或 16 位
- 地址阶段，SPI_CTRLR0.ADDR_L – 地址长度
- 数据阶段，CTRLR0.DFS – 数据长度

一条指令采用一个 FIFO 位置，地址可以采用多个(地址长度高于 32 位)FIFO 位置。指令和地址都通过数据寄存器(DR)写入。QSPI 会等待，直到两者都被准备好，以启动写操作指令。对于一次写操作，指令和地址只发送一次，随后是由 DR 编程的数据帧，直到发送 FIFO 变为空。地址和数据可以选择双线/四线模式模式，通过 SPI_CTRLR.TRANS_TYPE 和 CTRLR0.SPI_FRF 选择；数据帧按照 CTRLR0.SPI_FRF 指定格式发送。

以下是在增强的 SPI 模式中可能出现的写操作情况：

➢ A:指令和地址都以标准 SPI 格式传送(SPI_CTRLR0.TRANS_TYPE=00)

图 25-15 显示了当指令和地址都以标准 SPI 格式传输时的时序图。如果 CTRLR0.SPI_FRF=1, N 的值为:1，如果 CTRLR0. SPI_FRF=10,N 的值为 3。

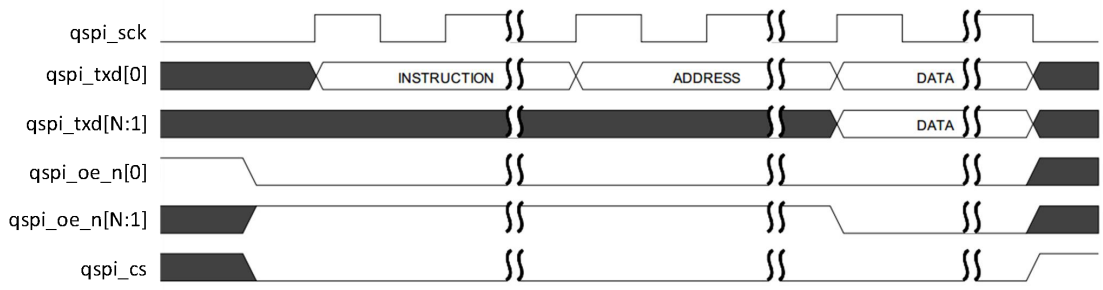


图 25-15 增强 SPI 模式的 TT0 写操作

➢ B:以标准格式传送指令和以增强 SPI 格式传送地址(SPI_CTRLR0.TRANS_TYPE=01)

图 25-16 显示了当指令以标准格式传输和地址以 CTRLR0.SPI_FRF 格式传输的时序图。

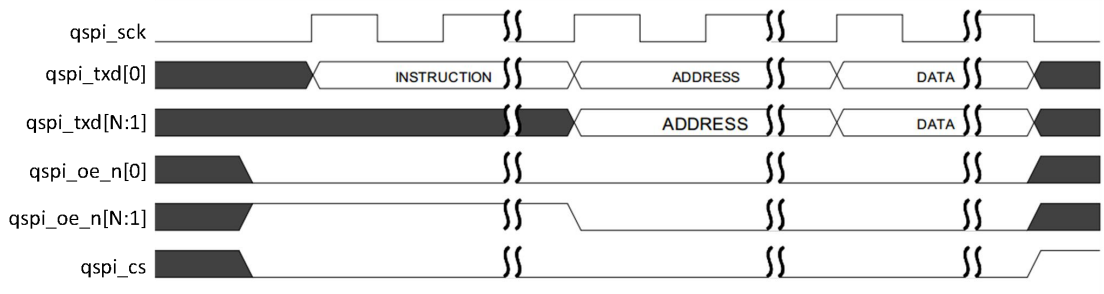


图 25-16 增强 SPI 模式的 TT1 写操作

➢ C: 指令和地址都以增强 SPI 格式传输(SPI_CTRLR0.TRANS_TYPE=10)

图 25-17 显示了指令和地址都以 CTRLR0.SPI_FRF 格式传输的时序图。

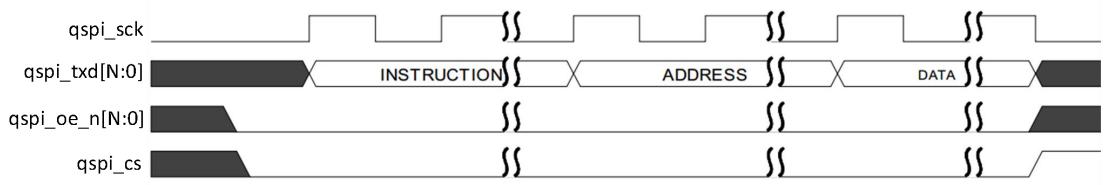


图 25-17 增强 SPI 模式的 TT2 写操作

➢ D: 仅指令传输增强的 SPI 格式 (SPI_CTRLR0.TRANS_TYPE=10)

图 25-18 显示了仅指令传输的增强 SPI 模式时序图。

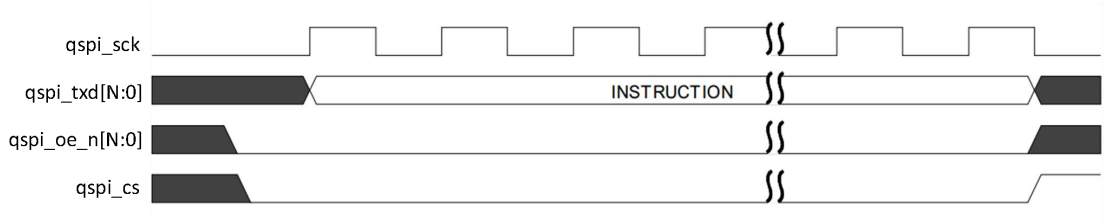


图 25-18 增强 SPI 模式只发送指令 TT2 写操作

一些设备只需要指令和地址转移(没有数据阶段)，这可以通过跳过地址阶段来实现，并将地址作为数据传输到设备。为此，SPI_CTRLR0.ADDR_L= 0,CTRLRO，CTRLR0.DFS 应该按照所需的长度进行编程。

25.6.2 读操作

双线 SPI 和四线 SPI 模式读操作可分为四部分:

- 指令阶段，SPI_CTRLR0.INST_L - 指令长度的可能值为 0、4、8 或 16 位
- 地址阶段，SPI_CTRLR0.ADDR_L – 地址长度
- 等待周期，SPI_CTRLR0.WAIT_CYCLES – 等待时间
- 数据阶段，CTRLR0.DFS – 数据长度

WAIT_CYCLES 字段引入了等待周期，主机把输出改为输入，从机从输入改为输出，不同从机等待周期不同。对于一个读操作，QSPI 仅发送一次指令和地址，并等待，直到它收到

CTRLR1.NDF 的数据帧数，然后拉高 CS 选择信号。

以下是在增强 SPI 模式下写操作的可能情况:

➢ A:指令和地址都以标准 SPI 格式发送读操作(SPI_CTRLR0.TRANS_TYPE=00)

图 25-19 显示了当指令和地址都以标准 SPI 格式传输时的时序图。如果 CTRLR0.SPI_FRF=1, N 的值为:1，如果 CTRLR0. SPI_FRF=10,N 的值为 3。

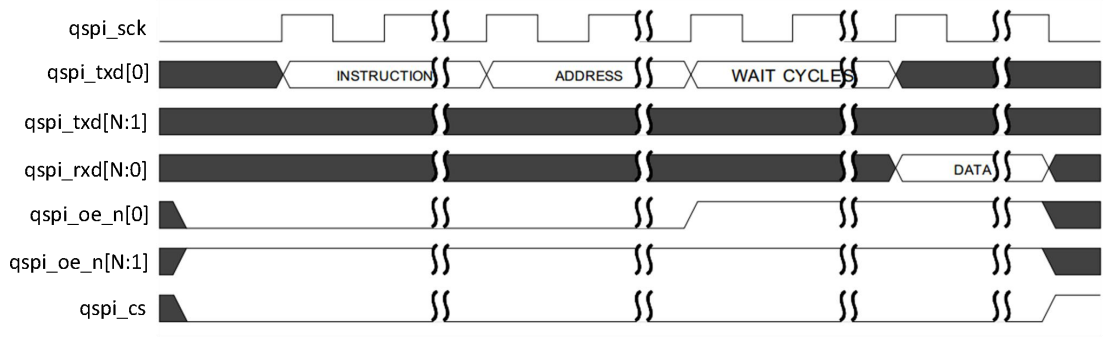


图 25-19 增强 SPI 模式的 TT0 读操作

➢ B:以标准格式传送指令和以增强 SPI 格式传送地址读操(SPI_CTRLR0.TRANS_TYPE=01)

图 25-16 显示了当指令以标准格式传输和地址以 CTRLR0.SPI_FRF 格式传输的时序图。

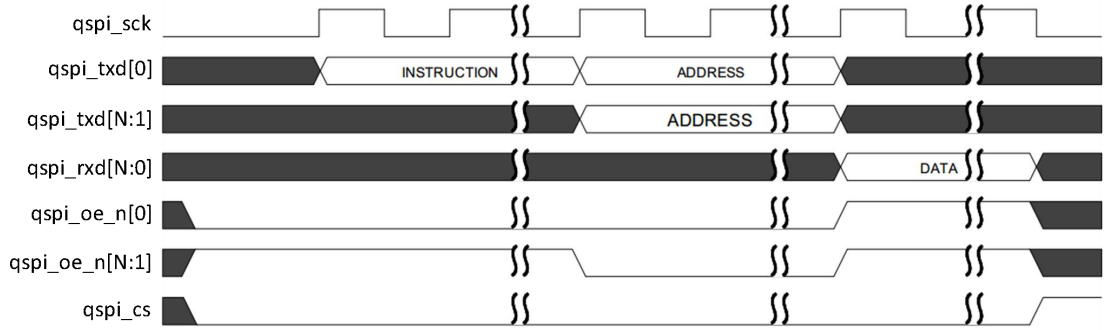


图 25-20 增强 SPI 模式的 TT1 读操作

➢ C: 指令和地址都以增强 SPI 格式传输(SPI_CTRLR0.TRANS_TYPE=10)

图 26-17 显示了指令和地址都以 CTRLR0.SPI_FRF 格式传输的时序图。

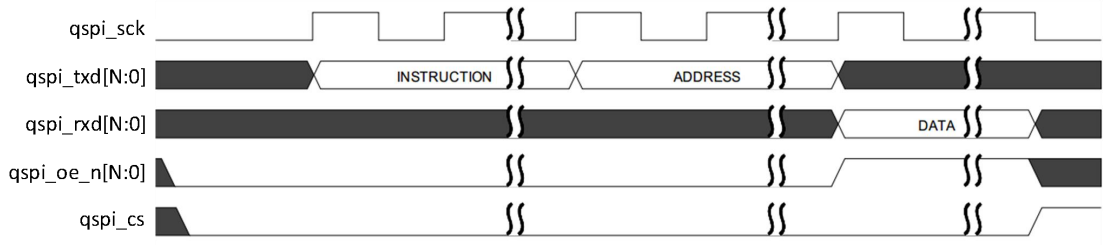


图 25-21 增强 SPI 模式的 TT2 读操作

➢ D: 没有指令、地址方式读操作(SPI_CTRLR0.TRANS_TYPE=10)

这种情况下，SPI_CTRLR0.INST_L 和 SPI_CTRLR0.ADDR_L 都必须设置为 0，且 SPI_CTRLR0.WAIT_CYCLES 必须设置为非零值。图 26-21 显示了这种传输类型的时序图。为了开始这个读操作，软件必须先对数据寄存器写一个任意数据，QSPI 在等待周期后，可连续读取在 NDF 指定的数据量。

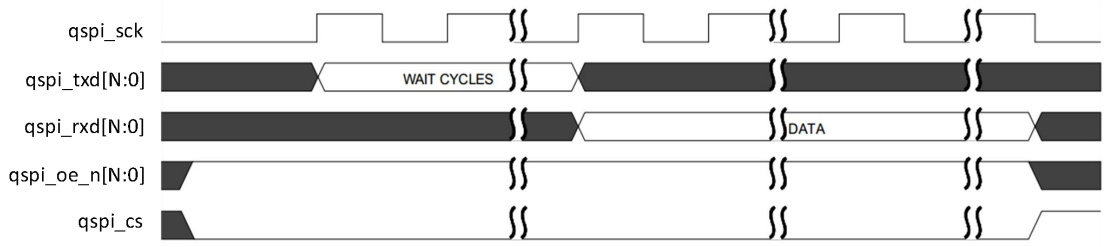


图 25-21 增强 SPI 模式无指令地址 TT2 读操作

25.7 通讯波特率计算和设定

QSPI 作为主机，通讯时钟由硬件主动产生并发送至从机，其通讯速率计算公式如下：

$F_{qspi\_sck} = F_{FCLK} / (SCKDIV)$

其中：

- F_{qspi_sck} 为期望的 QSPI 通讯速率，同时用于接收和发送
- F_{FCLK} 为 PCLK 时钟频率
- SCKDIV 为预分频寄存器的设定值，取值范围为 0~65534 间的偶数（其第 0 位必须为 0），如果 SCKDIV=0，F_{qspi_sck} 无输出。

F_{qspi_sck} 只在数据传输才会翻转，在所有其他时间，保持在无效状态，由它操作的串行协议定义。

25.8 QSPI 中断

QSPI 模块的中断源一共有 5 个，分别对应数据发送 FIFO 和数据接收 FIFO 的不同状态：

- **发送 FIFO 空中断**

  当 TX FIFO 的数据量小于等于 TXFTLR.TFT 设置的阈值，将触发发送 FIFO 空中断。当 TX FIFO 的数据超过设置阈值，硬件会清除这个中断。

- **发送 FIFO 溢出中断**

  当发送 FIFO 中的数据已经满了，再往 DR 写入数据，最新写入的数据将会被丢弃，将触发发送 FIFO 溢出中断。这个中断保持设置，直到读取 TXOICR/ICR 清除。

- **接收 FIFO 下溢中断**

  当尝试通过 DR 读取接收 FIFO 数据，且此时接收 FIFO 为空，读回 0，将触发接收 FIFO 下溢出中断。这个中断保持设置，直到读取 RXUICR/ICR 清除。

- **接收 FIFO 溢出中断**

  当接收 FIFO 已经满了，又接收到一笔数据，新接收的数据将被丢弃，将触发接收 FIFO 溢出中断。这个中断保持设置，直到 RXOICR/ICR 清除。

- **接收 FIFO 满中断**

  当接收 FIFO 的数据量大于等于 RXFTLR.RFT+1，将触发接收 FIFO 满中断。当从接收 FIFO 读取数据时，硬件会清除这个中断，使其低于阈值水平。

上述 5 个状态标志位软件可通过原始中断标志寄存器 RISR 查看。软件可通过配置中断使能控制寄存器 IMR 决定上述 5 个状态标志位中哪些中断源被使能从而发出中断请求，经 IMR 使能控制后的中断标志反映在中断状态寄存器 ISR 中。被使能后的中断标志通过或的逻辑，最终向 CPU 发出同一个中断请求，软件可通过查询具体的中断源并做出相应的处理。

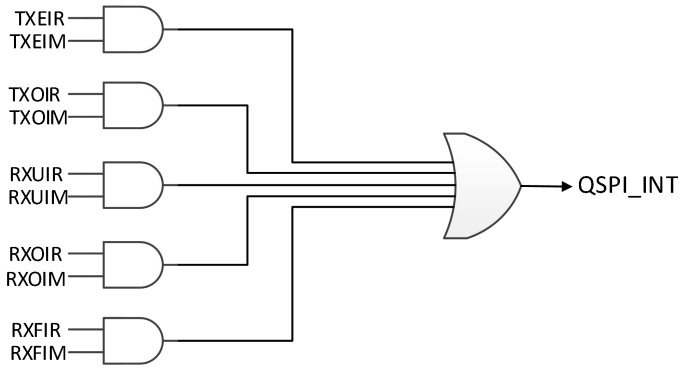


图 25-22 QSPI 中断映射图

25.9 DMA 传输

QSPI 模块支持数据收发缓存队列和系统内存间的 DMA 传输。SPI_DMA 寄存器可分别控制数据接收和发送的 DMA 功能。使用 DMA 时应用软件需事先在内存中开辟定义发送和接收的缓冲区，并设定特定 DMA 通道关联 SPI 模块的外设地址和内存地址。QSPI 使用两个 DMA 通道，一个用于传输数据，一个用于接收数据。

QSPI 有这些 DMA 寄存器:

- DMACR - 使能 DMA 操作
- DMATDLR - 当发送 FIFO 数据量小于等于 DMATDLR，QSPI 发送 DMA 请求被发出
- DMARDLR - 当接收 FIFO 数据量大于等于 DMARDLR+1，QSPI 接收 DMA 请求被发出

25.10 寄存器列表

地址	寄存器	描述	备注
0x4001_3000	CTRLR0	QSPI控制寄存器0	CTRLR0 说明
0x4001_3004	CTRLR1	QSPI控制寄存器1	CTRLR1 说明
0x4001_3008	QSPIENR	QSPI使能寄存器	QSPIENR 说明
0x4001_300C	MWCR	Microwire控制寄存器	MWCR 说明
0x4001_3010	SER	从机选择寄存器	SER 说明
0x4001_3014	BAUDR	波特率控制寄存器	BAUDR 说明
0x4001_3018	TXFTLR	发送FIFO阈值寄存器	TXFTLR 说明
0x4001_301C	RXFTLR	接收FIFO阈值寄存器	RXFTLR 说明
0x4001_3020	TXFLR	发送FIFO状态寄存器	TXFLR 说明
0x4001_3024	RXFLR	接收FIFO状态寄存器	RXFLR 说明
0x4001_3028	SR	状态寄存器	SR 说明
0x4001_302C	IMR	中断使能寄存器	IMR 说明
0x4001_3030	ISR	中断状态寄存器	ISR 说明
0x4001_3034	RISR	原始中断寄存器	RISR 说明
0x4001_3038	TXOICR	发送FIFO溢出中断清除寄存器	TXOICR 说明
0x4001_303C	RXOICR	接收FIFO溢出中断清除寄存器	RXOICR 说明
0x4001_3040	RXUICR	接收FIFO下溢中断清除寄存器	RXUICR 说明
0x4001_3044	MSTICR	多主机中断清除寄存器	MSTICR 说明
0x4001_3048	ICR	中断清除寄存器	ICR 说明
0x4001_304C	DMACR	DMA控制寄存器	DMACR 说明
0x4001_3050	MDATDLR	DMA发送触发级别寄存器	MDATDLR 说明
0x4001_3054	MDARDLR	DMA接收触发级别寄存器	MDARDLR 说明
0x4001_3058	IDR	ID寄存器	IDR 说明
0x4001_305C	VERSION_ID	版本寄存器	VERSION_ID 说明
0x4001_3060+ i*0x4	DRx (for i=0; i<+35)	数据寄存器	DRx说明
0x4001_30F4	SPI_CTRLR0	SPI控制寄存器	SPI_CTRLR0 说明

25.11 寄存器描述

25.11.1 控制寄存器 CTRLR0

位域	类型	复位	名称	描述
[31:24]	--	--	--	--
[23:22]	R/W	0x2	SPI_FRF	**帧格式** 当 FRF=0 时有效。 0x0 (SPI_STANDARD): Standard SPI Format 0x1 (SPI_DUAL): Dual SPI Format 0x2 (SPI_QUAD): Quad SPI Format 0x3 (SPI_OCTAL): Octal SPI Format（无效）
[21：20]	--	--	--	--
[19：16]	R/W	0x0	CFS	**控制帧长度** 用于 Microwire 传输格式的控制帧长度选择 0x0:1 位 0x1:2 位 0x2:3 位 …… 0xF:16 位
[15]	--	--	--	--
[14]	R/W	1	SSTE	**CS 信号翻转使能** 当在 SPI mode，且 SCPH 为 0，用于控制连续数据传输的 CS 行为 0：CS 信号将会保持为低，且 SCLK 在数据传输之间保持正常输出 1：CS 信号将在连续的数据帧之间切换，当 CS 为高，SCLK 保持默认值
[13]	R/W	0	SRL	**回绕模式使能控制** 0：正常工作模式 1：输出回绕至输入，用于功能诊断
[12]	R/W	0	SLV_OE	**从机输出禁止**（仅适用于从机模式） 0：从机正常输出 1：从机禁止输出
[11：10]	R/W	0x1	TMOD	**传输模式** 用于指示接收/发送是否有效； 在发送模式下，从外部设备接收的数据无效，且未存储在接收 FIFO 存储器中;它在下一次传输时被覆盖； 在接收模式下，发送的数据无效； 在收发模式中，收发数据都是有效的。 0x0：收发模式，不支持多线 SPI 模式 0x1：发送模式，在单线模式仅支持发送，或多线 SPI 写操作中 0x2：接收模式，仅支持接收，或多线 SPI 读操作中 0x3：EEPROM 读模式， 支持多线 SPI 模式
[9]	R/W	0	SCPOL	**时钟极性选择** 0：SCLK 空闲状态为低 1：SCLK 空闲状态为高

[8]	R/W	0	SCPH	**时钟相位选择** 0：数据在第一个 SCK 时钟沿被捕捉 1：数据在第二个 SCK 时钟沿被捕捉
[7:6]	R/W	0x0	FRF	**定义帧格式** 0x0: SPI 格式 0x2: Microwire 格式
[5]	--	--	--	--
[4:0]	R/W	0x7	DFS	**定义数据帧长** 选择数据帧长度。当数据帧长小于 32 位，接收数据自动右对齐的接收逻辑，接收 FIFO 的高位零填充。在将数据写入传输 FIFO 之前，必须对传输数据进行右对齐，在传输数据时忽略未使用的高位。 0x0:保留位 0x1:保留位 0x2:保留位 0x3:帧长 4 位 0x4:帧长 5 位 0x5:帧长 6 位 …… 0x1F:帧长 32 位

Note：当 QSPI 使能，该寄存器写无效

25.11.2 控制寄存器 CTRLR1

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:0]	R/W	0x0	NDF	**数据帧的数目** 当 TMOD= 10 或 TMOD= 11 时，这个寄存器设置 QSPI 连续接收的数据帧数。QSPI 继续接收串行数据，直到接收到的数据帧数等于这个寄存器值加上 1，可连续传输中接收最多 64KB 的数据。仅在主机模式可用。

Note：当 QSPI 使能，该寄存器写无效

25.11.3 使能寄存器 QSPIENR

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R/W	0x0	QSPIEN	**QSPI 使能** 0：QSPI 接口除能 1：QSPI 接口使能

25.11.4 Microwire 控制寄存器 MWCR

位域	类型	复位	名称	描述
[31:3]	--	--	--	--
[2]	R/W	0	MHS	**Microwire 握手** 用于在 microwire 主机模式下启用和禁止握手。发送模式，在传输一个数据之后，检查从机的状态位；接收模式，在接收完最后一个数据之后，清除 SR 寄存器中的忙碌状态之前，检查从机的状态位。 0：握手除能

				1：握手使能
[1]	R/W	0	MDD	**Microwire 方向设置** 0：接收 1：发送
[0]	R/W	0	MWMOD	**Microwire 传输模式** 当采用序列方式时，只需要一个控制字来发送或接收一串数据。当使用非序列模式时，每传输或接收的一个数据都需要先发送一个控制字。 0：非序列 1：序列

Note：当 QSPI 使能，该寄存器写无效

25.11.5 从机选择寄存器 SER

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R/W	0x1	SER	**CS 使能** 在开始传输之前，需要使能 CS 0：除能 1：使能

Note：当 QSPI 使能或者 BUSY，不可以修改该寄存器

25.11.6 波特率控制寄存器 BAUDR

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[15：1]	R/W	0x0	SCKDIV	**QSPI 时钟分频** 如果值为 0，则串行输出时钟 SCLK 被禁用，频率由下式导出: SCLK = PCLK/SCKDV 其中 SCKDV 是 2 到 65534 之间的任意偶数
[0]	R	0	SCKDIV_0	只读，固定为 0，确保分频值是偶数

25.11.7 发送 FIFO 阈值寄存器 TXFTLR

位域	类型	复位	名称	描述
[31:20]	-	--	--	--
[23:16]	R/W	0x0	TXFTHR	**开始发送 FIFO 级别设定** 用于控制开始传输的的级别，当 TX FIFO 数据高于设置的级别开始串行发送； 如果尝试写入大于或等于 TX FIFO 深度的值，将无法写入，且保留当前值。
[15:4]	--	--	--	--
[7:0]	R/W	0x0	TFT	**发送 FIFO 中断触发阈值设定** 当 TX FIFO 中数据小于或等于 TFT，将触发发送器 FIFO 空中断。 如果尝试写入大于或等于 TX FIFO 深度的值，将无法写入，且保留当前值。

25.11.8 接收 FIFO 阈值寄存器 RXFTLR

位域	类型	复位	名称	描述
[31:4]	--	--	--	--
[7:0]	R/W	0x0	RFT	**接收 FIFO 中断触发阈值设定**

				当 RX FIFO 中数据大于或等于 RFT+1，将触发接收器 FIFO 满中断。 如果尝试写入大于或等于 RX FIFO 深度的值，将无法写入，且保留当前值。

25.11.9 发送 FIFO 状态寄存器 TXFLR

位域	类型	复位	名称	描述
[31:5]	--	--	--	--
[4:0]	R/W	0x0	TXTFL	**发送 FIFO 状态** TX FIFO 的当前数据总数

25.11.10 接收 FIFO 状态寄存器 RXFLR

位域	类型	复位	名称	描述
[31:5]	--	--	--	--
[4:0]	R/W	0x0	RXTFL	**接收 FIFO 状态** RX FIFO 的当前数据总数

25.11.11 状态寄存器 SR

这是一个只读寄存器，用于指示当前传输状态、FIFO 状态。状态寄存器可以在任何时候读取，这个寄存器不会产生中断请求。

位域	类型	复位	名称	描述
[31:5]	--	--	--	--
[4]	R	0	RFF	接收 FIFO 满 当接收 FIFO 完全满时，这个位置位，当接收 FIFO 包含一个或多个空位置时，这 bit 被硬件清除。 0：接收 FIFO 是满的 1：接收 FIFO 未满
[3]	R	0	RNE	接收 FIFO 非空 当接收 FIFO 包含一个或多个数据时，这个位置位，当接收 FIFO 为空时硬件清除。这个位可以被软件轮询，以完全清空接收 FIFO。 0：接收 FIFO 为空 1：接收 FIFO 非空
[2]	R	1	TFE	发送 FIFO 全空 当发送 FIFO 完全为空时，这个位被置位。当发送 FIFO 包含一个或多个数据时，将硬件清除此位。 0：发送 FIFO 非空 1：发送 FIFO 全空
[1]	R	1	TFNF	发送 FIFO 未满 当发送 FIFO 包含一个或多个空位置时，这个位被置位，当 FIFO 满时硬件清除。 0：发送 FIFO 全满 1：发送 FIFO 未满
[0]	R	0	BUSY	收发忙状态 当该位置位时，表示串行传输正在进行；当清除时，表示 QSPI 为空闲或关闭。 0：空闲或者关闭 1：正在数据传输

25.11.12 中断使能寄存器 IMR

位	类	复	名称	描述
[31:7]	--	--	--	--
[5]	R/W	1	MSTIM	仅适用于从机模式
[4]	R/W	1	RXFIM	**接收 FIFO 满中断使能控制** 0：接收 FIFO 满不产生中断 1：接收 FIFO 满产生中断
[3]	R/W	1	RXOIM	**接收器 FIFO 溢出中断使能控制** 0：接收 FIFO 溢出不产生中断 1：接收 FIFO 溢出产生中断
[2]	R/W	1	RXUIM	**接收 FIFO 下溢中断使能控制** 0：接收 FIFO 下溢不产生中断 1：接收 FIFO 下溢产生中断
[1]	R/W	1	TXOIM	**发送 FIFO 溢出中断使能控制** 0：发送 FIFO 溢出不产生中断 1：发送 FIFO 溢出产生中断
[0]	R/W	1	TXEIM	**发送 FIFO 空中断使能控制** 0：发送 FIFO 空不产生中断 1：发送 FIFO 空产生中断

25.11.13 中断状态寄存器 ISR

位域	类型	复位	名称	描述
[31:5]	--	--	--	--
[4]	R	0	RXFIS	**接收 FIFO 满中断状态位** 0：接收 FIFO 未满或中断未使能 1：接收 FIFO 满并产生中断
[3]	R	0	RXOIS	**接收 FIFO 溢出中断状态位** 0：接收 FIFO 未发生溢出或中断未使能 1：接收 FIFO 溢出并产生中断
[2]	R	0	RXUIS	**接收 FIFO 下溢中断状态位** 0：接收 FIFO 未发生下溢或中断未使能 1：接收 FIFO 下溢并产生中断
[1]	R	0	TXOIS	**发送 FIFO 溢出中断状态位** 0：发送 FIFO 未发生溢出或中断未使能 1：发送 FIFO 溢出并产生中断
[0]	R	0	TXEIS	**发送 FIFO 空中断状态位** 0：发送 FIFO 非空或中断未使能 1：发送 FIFO 空并产生中断

25.11.14 原始中断状态寄存器 RISR

位域	类型	复位	名称	描述
[31:5]	--	--	--	--

[4]	R	0	RXFIR	**接收 FIFO 满状态位** 0：接收 FIFO 未满 1：接收 FIFO 满
[3]	R	0	RXOIR	**接收器 FIFO 溢出状态位** 0：接收 FIFO 未发生溢出 1：接收 FIFO 溢出
[2]	R	0	RXUIR	**接收器 FIFO 下溢状态位** 0：接收 FIFO 未发生下溢 1：接收 FIFO 下溢
[1]	R	0	TXOIR	**发送 FIFO 溢出状态位** 0：发送 FIFO 未发生溢出 1：发送 FIFO 溢出
[0]	R	0	TXEIR	**发送 FIFO 空状态位** 0：发送 FIFO 非空 1：发送 FIFO 空

25.11.15 发送 FIFO 溢出中断清除寄存器 TXOICR

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	BUSY	**清除发送 FIFO 溢出中断** 这个寄存器反应发送器 FIFO 溢出中断的状态。读取该寄存器清除发送器 FIFO 溢出中断，写无效。

25.11.16 接收 FIFO 溢出中断清除寄存器 RXOICR

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	RXOICR	**清除接收 FIFO 溢出中断** 这个寄存器反应接收 FIFO 溢出中断的状态。读取该寄存器清除接收 FIFO 溢出中断，写无效。

25.11.17 接收 FIFO 下溢中断清除寄存器 RXUICR

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	RXUICR	**清除接收 IFO 下溢中断** 这个寄存器反应接收器 FIFO 下溢中断的状态。读取该寄存器清除接收 FIFO 下溢中断，写无效。

25.11.18 多主机中断清除寄存器 MSTICR

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	MSTICR	**清除多主机中断** 这个寄存器反应多主机中断的状态。读取该寄存器清除多主机中断，写无效。

25.11.19 中断清除寄存器 ICR

位域	类型	复位	名称	描述
[31:1]	--	--	--	--
[0]	R	0	ICR	**中断清除** 如果 TXOIS/RXOIS/RXUIS/MSTIS 中的任何中断有效，则置位该位。读取该寄存器清除以上中断，写无效。

25.11.20 DMA 控制寄存器 DMACR

位域	类型	复位	名称	描述
[31:2]	--	--	--	--
[1]	R	0	TDMAE	发送 DMA 使能控制 0：发送 DMA 关闭 1：发送 DMA 使能
[0]	R	0	RDMAE	接收 DMA 使能控制 0：接收 DMA 关闭 1：接收 DMA 使能

25.11.21 DMA 发送触发级别寄存器 DMATDLR

位域	类型	复位	名称	描述
[31:4]	--	--	--	--
[3:0]	R/W	0x0	DMATDL	DMA 发送数据触发级别 当发送器 FIFO 中的数据小于或等于 DMATDL，并且 TDMAE= 1 时，就会触发 DMA 发送信号。

25.11.22 DMA 接收触发级别寄存器 DMARDLR

位域	类型	复位	名称	描述
[31:4]	--	--	--	--
[3:0]	R/W	0x0	DMARDL	DMA 发送数据触发级别 当接收器 FIFO 中的数据大于或等于 DMARDL+1，并且 RDMAE= 1 时，就会触发 DMA 接收信号。

25.11.23 ID 寄存器 IDR

位域	类型	复位	名称	描述
[31:0]	R	0xFFFF FFFF	IDCODE	QSPI ID

25.11.24 版本寄存器 VERSION_ID

位域	类型	复位	名称	描述
[31:0]	R	0x3130 322A	VERSION	QSPI VERSION

25.11.25 数据寄存器 DRx(for 1=0; i<=35)

数据寄存器是一个 32 位的读/写缓冲区，用于发送/接收 FIFO。当 SSIC EN=0 时，FIFO reset。

注意，QSPI 中的 DR 寄存器占用 36 个 32 位地址，以方便 AHB 突发传输。往这里任意地址写入的效果相同；从任意地址读取数据的效果相同。

位域	类型	复位	名称	描述
[31:0]	R/W	0	DR	数据寄存器 当写入到这个寄存器时，数据需要右对齐。读取的数据自动右对齐。 读取：读取接收 FIFO 数据 写入：将数据写入发送 FIFO，QSPIEN=1 时有效

25.11.26 SPI 控制寄存器 SPI_CTRLR0

这个寄存器是用来控制串行数据传输的增强 SPI 模式的操作。只有当 SPI_FRF(在 CTRLRO 中)被设置为 01 或 10 或 11 时，寄存器才相关。当 QSPI 使能，该寄存器写无效。

位域	类型	复位	名称	描述
[31:16]	--	--	--	--
[15:11]	R/W	0x0	WAIT_CYCLES	**等待周期** 设定在 DUAL/QUAD/OCTAL 模式控制帧传输和数据接收之间的等待周期。 指定为 SPI 时钟周期的数目
[10]	--	--	--	--
[9:8]	R/W	0x2	INST_L	**指令帧长度** 用于在 DUAL/QUAD/OCTAL 模式的指令帧长度 0x0: 无指令 0x1: 4bit 指令 0x2: 8bit 指令 0x4: 16 bit 指令
[7:6]	--	--	--	--
[5:2]	R/W	0x0	ADDR_L	**地址帧长度** 这个位定义要传输的地址长度。只有在将地址写入 FIFO 后，传输才能开始。 0x0: 无地址 0x1: 4bit 地址 0x2: 8bit 地址 0x3: 12bit 地址 0x4: 16bit 地址 0x5: 20bit 地址 0x6: 24bit 地址 …… 0xF: 60bit 地址
[1：0]	R/W	0	TRANS_TYPE	**指令和地址发送格式** 选择指令/地址将以标准 SPI 模式或 SPI_FRF 指定的模式传输。 0x0:(TTO):指令和地址将以标准 SPI 模式发送 0x1 (TT1):指令将以标准 SPI 模式发送，地址将以指定的模式发送 0x2 (TT2):指令和地址将以 SPI_FRF 指定的模式发送

26 USB

26.1 概述

该 USB 设备控制器符合 USB 1.1 全速设备的技术规范。其具有一个控制端点(即端点 0)，和 2 个可配置的端点 (EP1~EP2)。一个 512-byte 的 SRAM 用作端点数据发送接收缓冲器。每个端点缓冲器的大小可通过设置相应寄存器来得到，为各种应用提供最大的灵活性。USB 包含了挂起和恢复功能，满足了低功耗产品的需求。

USB 的 FIFO 接口是可配置的，端点 0 的 FIFO 可配置为 8、16、32 或 64 字节的单缓冲数据包，其他端点的 FIFO 可配置从 8~512 字节大小，并可以单/双缓冲数据包。

单独的 FIFOs 可以与每个端点相关联：或者，具有相同端点编号的 IN 端点和 OUT 端点可以配置为使用相同的 FIFO，例如减少所需的 RAM 的大小。

提供了所有的 USB 包编码、解码和检查，当端点数据已成功转移时产生中断。

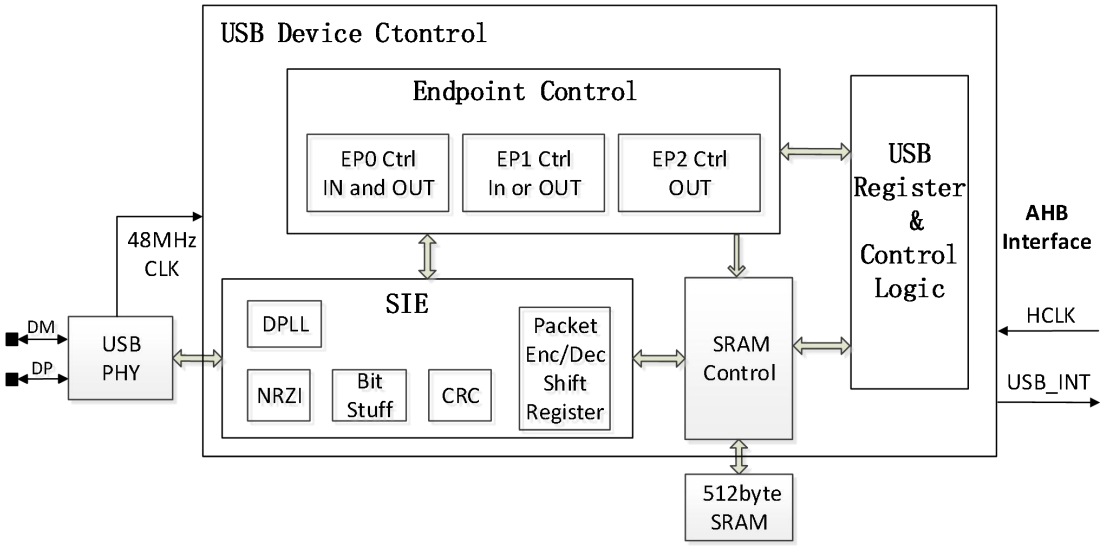


图 26-1. USB 方框图

26.2 特性

- 符合 USB 1.1 全速设备技术规范
- 完全集成的 USB 全速收发器
- 1 个控制端点 (EP0) 用于控制传输
- 2 个双缓冲端点 (EP1~EP2) 用于批量、中断和等时传输
- 512 字节 SRAM 用作端点数据缓冲器

26.3 功能描述

26.3.1 端点

USB 端点 0 是唯一一个用于 USB 控制传输的双向端点。该设备也包含 7 个单向的端点，用于其它 USB 传输类型。EP1 支持双缓冲功能用于批量和中断和等时 IN 或 OUT 数据传输。EP2 支持双缓冲功能用于批量和中断和等时 OUT 数据传输。下表列出了端点特性。

表 26-1. 端点特性

端点地址	传输类型	方向	缓冲器类型
0	控制	IN 和 OUT	单缓冲
1	中断/批量等时	IN 或 OUT	单或双缓冲
2	中断/批量/等时	OUT	单或双缓冲

26.3.2 串行接口引擎 - SIE

SIE 处理 NRZI 编码/解码、位填充/解填充和 CRC 生成/检查。它从输入的 48MHz 时钟产生一个 12MHz 的 USB 时钟，并同步到从 USB 接收数据流。它为要传输的信息包生成报头，并对接收的信息包的报头进行解码。

26.3.3IN 事务处理

IN 端点 1 的 FIFO 深度为 128-byte。

IN 传输的最大数据包大小是可编程的，由每个端点写入 InMaxP 寄存器的值决定。如果将设置值小于或等于 FIFO 大小的一半，则 IN 事务启用双缓冲。如果最大数据包大于 FIFO 大小的一半，则启用单缓冲。当启用双缓冲时，FIFO 可以缓冲两个数据包；启用单缓冲时，即使数据包小于 FIFO 大小的一半，也只能缓冲一个数据包。

单缓存：当每个要发送的数据包被装载到 FIFO 中时，需要将 InCSR1 中的 InPktRdy 置 1，如果设置了 InCSR2 中的 AutoSet 位，那么当 InMaxP 对应的数据包被装载到 FIFO 中时，InPktRdy 位会自动被设置。对于小于最大数据包的数据载入 FIFO，InPktRdy 需要手动设置。当 InPktRdy 位被手动或自动置 1 时，InCSR1 中的 FIFONotEmpty 位也被设置，包就准备好发送了。当包被成功发送时，InPktRdy 和 FIFONotEmpty 都被清除，并生成 IN 端点中断。

双缓冲：当每个要发送的数据包被装载到 FIFO 中时，需要将 InCSR1 中的 InPktRdy 置 1，如果设置了 InCSR2 中的 AutoSet 位，那么当 InMaxP 对应的数据包被装载到 FIFO 中时，InPktRdy 位会自动被设置。对于小于最大数据包的数据载入 FIFO，InPktRdy 需要手动设置。当手动或自动置位 InPktRdy 位时，InCSR1 中的 FIFONotEmpty 将会自动置 1，且 InPktRdy 将自动清除。第二个包现在可以加载到 FIFO，InPktRdy 再次置 1(手动或自动如果载入数据包是最大数据包)。当第一个包被成功发送，InPktRdy 被清除，产生 IN 端点中断产生。此时可以还可以再载入一个数据包。FIFONotEmpty 与 InPktRdy 表示 FIFO 中有多少个包。

26.3.4 OUT 事务处理

OUT 端点 1~2 的 FIFO 深度都为 128-byte。

OUT 传输的最大数据包大小是可编程的，由每个端点写入 OutMaxP 寄存器的值决定。如果设置值小于或等于 FIFO 大小的一半，则 OUT 事务启用双缓冲。当最大数据包大于 FIFO 大小的一半时，启用单缓冲。当启用双缓冲时，FIFO 可以缓冲两个数据包；启用单缓冲时，即使数据包小于 FIFO 大小的一半，也只能缓冲一个数据包。

单缓存：当一个包被接收并放置在 FIFO, OutPktRdy 位和在 OutCSR1 的 FIFOFull 位被置 1，且生成对应的 OUT 端点中断来表示数据包现在可以从 FIFO 读走。在包被读走后，OutPktRdy 位需要被清除，以便下个包被接收。如果设置了 OutCSR2 中的 AutoClear 位并且从 FIFO 中读走了一个最大数据包大小，则自动清除 OutPktRdy 位，FIFOFull 也被清除。对于小于最大数

据包，OutPktRdy 需要手动清除。

双缓冲：当接收到的第一个包装入 OUT FIFO 时，OutPktRdy 位在 OutCSR1 被置位，并生成对应的 OUT 端点中断，来表示数据包现在可以从 FIFO 卸载。此时 OutCSR1 中的 FIFOFull 位还没有置 1，只有在接收到第二个数据包并加载到 OUT FIFO 中时才会置 1。在第一个包被读走后，OutPktRdy 需要被清除，以便允许下一个的包被接收。如果设置了 OutCSR2 中的 AutoClear 位并且从 FIFO 中卸载了一个最大数据包大小的包，则 OutPktRdy 位将被自动清除。对于小于最大数据包大小的包，OutPktRdy 需要手动清除。当 OutPktRdy 被清除时，如果 FIFOFull 位为 1, USB 将首先清除 FIFOFull 位，并再次置位 OutPktRdy，以表明有另一个包在 FIFO 中。

26.3.5 挂起和唤醒

USB 设备在经过 3ms 内总线空闲，并且 EnableSuspend 位置 1，USB 进入挂起模式，产生挂起中断(如果使能)。

USB 进入挂起模式后，48MHz 时钟停止，当发生 bus activity，USB PHY 检测到恢复信号，48MHz 时钟恢复，产生唤醒中断。

26.4 USB 中断

当发生 USB 中断，需要读中断状态寄存器，以确定哪个端点导致中断，并跳转到适当的例程。如果多个端点导致了中断，那么应该首先服务端点 0，然后服务其他端点，挂起中断应该最后处理。

- **以下情况将触发端点 0 中断:**

  在接收到有效令牌并将数据写入 FIFO 后设置 OutPktRdy 位时。

  当 FIFO 中的数据包已经成功传输到主机后，清除 InPktRdy 位时。

  当一个控制事务由于协议违反而终止时，置位 SentStall 时。

  当置位 SetupEnd 位时，因为控制传输在置位 DataEnd 之前就已经结束了。

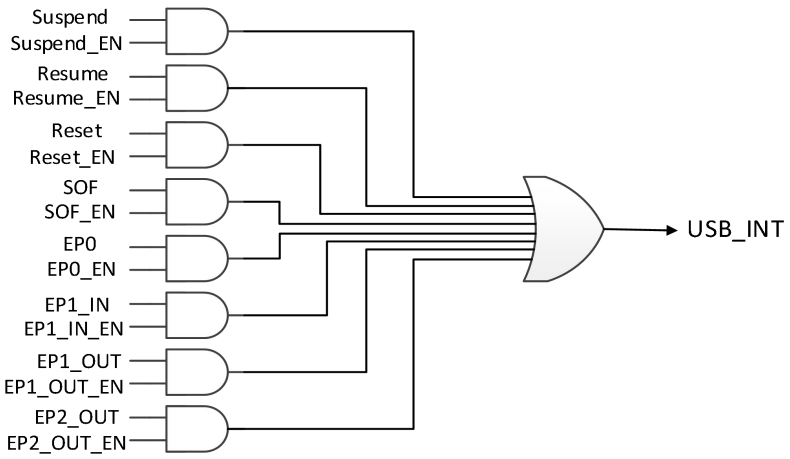


图 26-2. USB 中断映射图

26.5 寄存器描述

26.6 寄存器列表

USB 寄存器分为三个部分:

- 通用 USB 寄存器(offset: 0x0 ~ 0xF)

  这些寄存器为 USB 控制器提供控制和状态。

- 索引寄存器(offset: 0x10 ~0x1F)

  这些寄存器为不同的端点提供控制和状态信息：对于每个 IN 端点，有一个 InMaxP 和两个 InCSR 寄存器；对于每个 OUT 端点，有一个 OutMaxP、两个 OutCSR 和两个 OutCount 寄存器(除了端点 0)。在任何时候，只有一个 IN 端点和对应的 OUT 端点(即具有相同端点编号的端点)的寄存器出现在寄存器映射中。通过将端点号写入索引寄存器来选择端点。因此，要访问 IN 端点 1 中的寄存器和 OUT 端点 1 中的寄存器，必须首先将 1 写入索引寄存器，然后该端点的控制和状态寄存器出现在内存映射中。

- FIFOs

  IN 端点的 FIFOs 以单个字节的形式出现，从地址 20h 开始在内存映射中连续出现。OUT 端点的 FIFOs 也连续出现在同一组地址上。写地址 0x24 会加载的一个字节到 IN 端点 1 的 FIFO 中；读取地址 0x24，将会读回 OUT 端点 1FIFO 的数据。

地址	寄存器	描述	备注
0x4001_4000	FADDR	地址寄存器	FADDR 说明
0x4001_4001	POWER	电源控制寄存器	POWER 说明
0x4001_4002	INTRIN1	IN端点中断寄存器1	INTRIN1 说明
0x4001_4003	INTRIN2		
0x4001_4004	INTROUT1	OUT端点中断寄存器1	INTROUT1 说明
0x4001_4005	INTROUT2		
0x4001_4006	INTRUSB	USB中断寄存器	INTRUSB 说明
0x4001_4007	INTRIN1E	IN端点中断使能寄存器1	INTRIN1E 说明
0x4001_4008	INTRIN2E		
0x4001_4009	INTROUT1E	OUT端点中断使能寄存器1	INTROUT1E 说明
0x4001_400A	INTROUT2E		
0x4001_400B	INTRUSBE	USB中断使能寄存器	INTRUSBE 说明
0x4001_400C	FRAME1	帧号寄存器1	FRAME1 说明
0x4001_400D	FRAME2	帧号寄存器2	FRAME2 说明
0x4001_400E	INDEX	索引寄存器	INDEX 说明
0x4001_400F			
0x4001_4010	EP_INMAXP	IN端点(1)最大数据包寄存器	EP_INMAXP 说明
0x4001_4011	EP0_CSR/ EP_INCSR1	IN(0~1)端点控制状态寄存器	EP0_CSR说明 EP_INCSR1说明
0x4001_4012	EP_INCSR2	IN(1)端点控制状态寄存器2	EP_INCSR2 说明

0x4001_4013	EP_OUTMAXP	OUT端点最大数据包寄存器	EP_OUTMAXP 说明
0x4001_4014	EP_OUTCSR1	OUT端点(1~2)控制状态寄存器1	EP_OUTCSR1 说明
0x4001_4015	EP_OUTCSR2	OUT端点(1~2)控制状态寄存器2	EP_OUTCSR2 说明
0x4001_4016	EP0_COUNT/ EP_COUNT1	接收计数寄存器	EP0_COUNT 说明 EP_COUNT1 说明
0x4001_4017	EP_COUNT2	接收计数寄存器2	EP_COUNT2说明
0x4001_4020	EP0_FIFO	端点0 FIFO寄存器	EP0_FIFO 说明
0x4001_4024	EP1_FIFO	端点1 FIFO寄存器	EP1_FIFO 说明
0x4001_4028	EP2_FIFO	端点2 FIFO寄存器	EP2_FIFO 说明

26.7 寄存器描述

26.7.1 地址寄存器 FADDR

FAddr 是一个 8 位寄存器，包含 7 位地址。它被用于解码后续令牌包中的地址。

主机往地址为 0 的设备的端点 0 发送一个设置地址的请求，设备收到这个请求后，这个寄存器应该被写入包含在设置地址请求中的地址值，新地址将不会立即生效，因为主机将仍然使用旧地址作为设备请求的状态阶段。USB 将继续使用旧地址进行解码包，直到设备请求完成。设备请求的状态可以通过读取这个寄存器的第 7 位来确定。当一个新的地址被写入这个寄存器时，第 7 位将被自动设置。它将保持高位直到设备请求完成，并将在新地址生效时被清除。

位域	类型	复位	名称	描述
[7]	R	0x0	Update	当 FAddr 被写入时设置。在新地址生效时清除(在当前传输结束时)。
[6:0]	R/W	0x0	FAddr	USB 设备地址

26.7.2 电源控制寄存器 POWER

用于控制暂停和恢复信号。

位域	类型	复位	名称	描述
[7]	R/W	0	ISO_Update	当该位置 1，USB 将等待一个 SOF 包收到之后， InPktRdy 在数据包发送之前才会置 1。如果一个 IN 令牌包在一个 SOF 令牌之前被接收，那么将发送一个零长度的数据包。此位仅供执行等时传输的端点使用
[6:4]	--	--	--	--
[3]	R	0	Reset	当总线上存在复位信号时，这个位被置位。
[2]	R/W	0	--	--
[1]	R	0	Suspend_Mode	进入挂起模式时被置位。当 CPU 读取中断寄存器或置位该寄存器的 Resume 时发生远程唤醒清除。
[0]	R/W	0	Suspend_EN	由 CPU 设置，当总线上接收到挂起信号时，进入挂起模式。

26.7.3 IN 端点中断寄存器 INTRIN1

读寄存器，会清除有效中断

位域	类型	复位	名称	描述
[7:2]	--	--	--	--
[1]	R	0	EP1	IN 端点 1 中断
[0]	R	0	EP0	端点 0 中断

26.7.4 OUT 端点中断寄存器 INTROUT1

读取整这个寄存器，会清除有效中断

位域	类型	复位	名称	描述
[7:3]	--	--	--	--
[2]	R	0	EP2	OUT 端点 2 中断
[1]	R	0	EP1	OUT 端点 1 中断

[0]	--	--	--	--

26.7.5 USB 中断寄存器 INTRUSB

读取整这个寄存器，会清除有效中断

位域	类型	复位	名称	描述
[7:4]	--	--	--	--
[3]	R	0	SOF	帧开始时置 1
[2]	R	0	Reset	当总线上检测到复位信号时置 1
[1]	R	0	Resume	当 USB 处于 Suspend 模式时，在总线上检测到唤醒信号时置 1
[0]	R	0	Suspend	当总线上检测到挂起信号时置 1

26.7.6 IN 端点中断使能寄存器 INTRIN1E

位域	类型	复位	名称	描述
[7:2]	--	--	--	--
[1]	R/W	1	EP1	IN 端点 1 中断使能
[0]	R/W	1	EP0	端点 0 中断使能

26.7.7 OUT 端点中断使能寄存器 INTROUT1E

位域	类型	复位	名称	描述
[7:3]	-	--	--	--
[2]	R/W	1	EP2	OUT 端点 2 中断使能
[1]	R/W	1	EP1	OUT 端点 2 中断使能
[0]	-	--	--	-

26.7.8 USB 中断使能寄存器 INTRUSBE

位域	类型	复位	名称	描述
[7:4]	--	--	--	--
[3]	R	0	SOF	SOF 中断使能
[2]	R	0	Reset	Reset 中断使能
[1]	R	0	Resume	Resume 中断使能
[0]	R	0	Suspend	Suspend 中断使能

26.7.9 帧号寄存器 FRAME1

USB 应该每 1ms 从主机接收一次帧开始数据包。当接收到 SOF 包时，将数据包中包含的 11 位帧号写入两个寄存器 Frame1 和 Frame2。一旦 USB 开始接收 SOF 包，它期望每 1ms 接收一个。如果在 1.00358 ms 之后没有收到 SOF 包，则假设该数据包已经丢失，尽管帧寄存器没有更新，但仍生成一个 SOF 脉冲。USB 将继续每 1ms 生成一个 SOF 脉冲。当再次收

到 SOF 包，它将重新同步这些脉冲。

位域	类型	复位	名称	描述
[7:0]	R	0x0	FRAME	帧号：保存接收到的低八位帧号

26.7.10 帧号寄存器 FRAME2

位域	类型	复位	名称	描述
[7:3]	--	--	--	--
[2:0]	R	0x0	FRAME	帧号：保存接收到的高三位帧号

26.7.11索引寄存器 INDEX

决定偏移地址 0x10 ~ 0x17 访问哪个端点控制/状态寄存器。

位域	类型	复位	名称	描述
[7:4]	--	--	--	--
[3:0]	R/W	0x0	INDEX	**索引端点** 每个 IN 端点和 OUT 端点都有自己的一组控制/状态寄存器。在任何时候，只有一个 IN 端点和对应的 OUT 端点的控制/状态寄存器出现在内存映射中。在访问端点的控制/状态寄存器之前，应该将端点编号写入索引寄存器，以确保正确的控制/状态寄存器出现在内存映射中。

26.7.12 端点 0 控制状态寄存器 CSR0

位域	类型	复位	名称	描述
[7]	R/W	0	Serviced SetupEnd	这位写 1 用于清除 SetupEnd，该位自动清零
[6]	R/W	0	Serviced OuPktRdy	这位写 1 用于清除 OuPktRdy，该位自动清零
[5]	R/W	0	SendStall	CPU 向这个位写入一个 1 以终止当前事务。STALL 握手将被传送，然后这个位将被自动清除。
[4]	R	0	SetupEnd	在 DataEnd 位设置之前，结束控制传输，将置 1 此位，此时将生成一个中断并刷新 FIFO。写 1 到 ServicedSetupEnd 位清除此位。
[3]	R/W	0	DataEnd	CPU 置 1 这个位: 1. 当加载最后的数据包到 FIFO； 2. 当读取最后的数据包，清除 OutPktRdy 后； 3. 零长度的数据包，置 1 InPktRdy 时。 它被自动清除。
[2]	R	0	SentStall	此位在传输 STALL 握手时置 1。CPU 应该清除这个位。
[1]	R/W	0	InPktRdy	CPU 在将数据包加载到 FIFO 后将此位置 1。当数据包被传输后，它会被自动清除。当该位被清除时，会产生一个中断。
[0]	R	0	OutPktRdy	在收到数据包时被置 1。当置位这个位时，会产生一个中断。CPU 通过写 1 到 ServicedOutPktRdy 来清除这个位。

如果端点 0 处于写状态，中断表明已经接收到 IN 令牌，并且来自 FIFO 的数据已经发

送。固件必须响应此请求，如果主机仍然需要更多的数据，则通过在 FIFO 中放置更多的数据，或者通过置位 DataEnd 位来表示数据阶段已经完成。一旦事务的数据阶段完成，端点 0 应该被返回到空闲状态以等待下一个控制事务。

如果端点处于读状态，中断表示已经接收到数据包，固件必须通过从 FIFO 卸载接收到的数据来响应，并确定是否已接收到所有预期的数据。如果有，固件应该置位 DataEnd 位，端点 0 到空闲状态。如果需要更多的数据，固件应该置位 ServicedOutPktRdy 位，以表示它已经在 FIFO 中读取了数据，并将端点保持在读状态。

26.7.13 端点 0 接收计数值寄存器 COUNT0

位域	类型	复位	名称	描述
[7]	--	--	--	--
[6:0]	R	0	EP0_COUNT	**端点 0 计数值** 指示端点 0 FIFO 中接收到的数据字节数。当 OutPktRdy 被置 1 时，返回的值是有效的。

26.7.14 IN 端点(1)最大数据包长度寄存器 INMAXP

保存当前选中的 IN 端点传输的最大数据包大小——以 8 字节为单位，除了一个值为 128 的最大数据包大小设置为 1023(全速下的等时传输最大数据包)，而不是 1024。在设置这个值，应该注意 USB 对包大小在全速操作中的批量、中断和等时事务的约束。每个 IN 端点都有一个 InMaxP 寄存器(端点 0 除外)

位域	类型	复位	名称	描述
[7:0]	R	0	INMAXP	IN 端点最大数据包长度

写入到这个寄存器的值所表示的数据总量应该与该端点 IN FIFO 大小(128)匹配，不匹配可能会导致意外的结果。如果将一个大于 16(128/8)值写入该寄存器，则该值将自动调整为 16。如果写入到这个寄存器的值小于或等于 IN FIFO 大小的一半(128/2/8=8)，则可以缓冲两个 IN 包。

寄存器被复位值为 0。如果这个寄存器在端点发送了数据包之后发生了更改，那么在将新值写入这个寄存器之后，FIFO 中的端点应该被完全刷新(使用 InCSR1 中的 FlushFIFO 位(D3))

26.7.15 IN 端点(1)控制状态寄存器 INCSR1

通过当前选择的 IN 端点提供控制和状态位。每个端点(不包括端点 0)都有一个 InCSR1 寄存器。

位域	类型	复位	名称	描述
[7]	--	--	--	--
[6]	R/W	0	ClrDataTog	写 1 到这个位重置该 IN 端点在 DATA 包 PID 为 DATA0
[5]	R	0	SentStall	此位在传输 STALL 握手时置 1。CPU 应该清除这个位。 此位在传输 STALL 握手时设置。FIFO 被刷新，InPktRdy 位被清除。CPU 应该清除这个位。当位被设置时，会产生一个中断。
[4]	R/W	0	SendStall	CPU 向这个位写入一个 1 以终止当前事务。CPU 应该清除这个位。如果 IN 端点在 ISO 模式下，此位没有影响。
[3]	R/W	0	FlushFIFO	在 FIFO 中，CPU 写 1 到这个位以刷新要从端点传输的下一个数据包。FIFO 指

				针被重置，InPktRdy 位被清除。注意:如果 FIFO 包含两个数据包，则需要设置两次 FlushFIFO 才能完全清除 FIFO。会产生一个中断(InPktRdy 被清除)。
[2]	R	0	UnderRun	在 ISO 模式下，在收到 IN 令牌包后，当 InPktRdy 位未置 1，零长度的数据包发送后，这个位被设置。在 Bulk/Interrupt 模式下，在收到 IN 令牌包后，返回 NAK 包，这个位被设置。CPU 应该清除这个位。
[1]	R	0	FIFONE	当 FIFO 中至少有一个数据包时此位置 1
[0]	R/W	0	InPktRdy	CPU 在将数据包加载到 FIFO 后设置此位。当数据包被传输时，它会被自动清除。当位被清除时，会产生一个中断(如果启用)。

26.7.16 IN 端点(1)控制状态寄存器 INCSR2

它通过当前选择的 IN 端点提供进一步的控制位。每个端点(不包括端点 0)都有一个 InCSR2 寄存器。

位域	类型	复位	名称	描述
[7]	R/W	0	AutoSet	如果 CPU 设置这个位，InPktRdy 将自动设置时，最大数据包大小的数据(InMaxP 中的值)加载到 in FIFO。如果加载到 FIFO 的数据包小于最大数据包大小，InPktRdy 必须手动设置。
[6]	R/W	0	ISO	CPU 置 1 这个位来启用 IN 端点进行等时传输(ISO 模式)，清除它以启用 IN 端点进行批量/中断传输。
[5]	R/W	1	Mode	CPU 置 1 此位以启用端点方向为 IN，清除此位以启用端点方向为 OUT。只有在输入和输出事务使用相同端点 FIFO 时才有效。
[4]	R/W	0	--	--
[3]	R/W	0	FrcDataTog	CPU 置 1 此位，以强制 IN 端点的 DATA PID 在每个数据包发送后切换，且 InPktRdy 被清除，而不管是否收到 ACK。这可以在 IN 端点中断传输使用，被用来反馈等时传输通信速率。
[2:0]	--	--	--	--

26.7.17 OUT 端点(1~2)最大数据包长度寄存器 OUTMAXP

保存当前选中的 OUT 端点传输的最大数据包大小——以 8 字节为单位，除了一个值为 128 的最大数据包大小设置为 1023(全速下的等时传输最大数据包)，而不是 1024。在设置这个值，应该注意 USB 对包大小在全速操作中的批量、中断和等时事务的约束。每个 OUT 端点都有一个 OutMaxP 寄存器(端点 0 除外)

位域	类型	复位	名称	描述
[7:0]	R/W	0x0	OUTMAXP	IN 端点最大数据包长度

写入到这个寄存器的值所表示的数据总量不能超过该端点 OUT FIFO 大小(128)，如果需要使用双缓冲，也不能超过 FIFO 大小的一半。如果将大于端点 OUT FIFO 大小的值写入该寄存器，则该值将自动更改为 OUT FIFO 大小。如果写入到这个寄存器的值小于或等于 OUT FIFO 大小的一半，两个输出包可以被缓冲。

26.7.18 OUT 端点(1~2)控制状态寄存器 OUTCSR1

位域	类型	复位	名称	描述

[7]	R/W	0	ClrDataTog	写 1 到这个位重置该 OUT 端点在 DATA 包 PID 为 DATA0
[6]	R	0	SentStall	此位在传输 STALL 握手时置 1。CPU 应该清除这个位。当位被设置时，会产生一个中断。
[5]	R/W	0	SendStall	CPU 向这个位写 1 以终止当前事务。CPU 应清除此位。如果 OUT 端点在 ISO 模式下，此位没有影响。
[4]	R/W	0	FlushFIFO	CPU 在这个位上写 1，以清除要从端点读取的下一个数据包 FIFO。注意:如果 FIFO 包含两个数据包，FlushFIFO 需要置 1 两次以完全清除。
[3]	R	0	DataError	如果数据包有 CRC 或位错误，当 OutPktRdy 被设置时，这个位被置 1。当 OutPktRdy 被清除时，它被清除。该位仅在 ISO 模式下有效。
[2]	R/W	0	OverRun	如果 OUT 包不能加载到输出 FIFO 中，则此位置 1。CPU 应该清除这个位，该位仅在 ISO 模式下有效。
[1]	R	0	FIFOFull	当没有更多的数据包可以加载到 OUT FIFO 时此位置 1。
[0]	R/W	0	OutPktRdy	在收到数据包时被置 1。当数据包从 OUT FIFO 中读走时，CPU 应该清除这个位。当位被设置时，会产生一个中断。

26.7.19 OUT 端点(1~2)控制状态寄存器 OUTCSR2

位域	类型	复位	名称	描述
[7]	R/W	0	AutoClear	如果 CPU 设置这个位，OutPktRdy 位将在 OutMaxP 所表示的数据包大小从 OUT FIFO 读走时被自动清除。当数据包小于 OutMaxP 所表示的数据大小被读走时，CPU 必须手动清除。
[6]	R/W	0	ISO	CPU 置 1 这个位来启用 IN 端点进行等时传输(ISO 模式)，清除它以启用 IN 端点进行批量/中断传输。
[5:0]	R/W	0	--	--
[4]	R/W	0	--	--
[2:0]	--	--	--	--

26.7.20 接收计数寄存器 OUTCOUNT1

它保存当前输出端点 FIFO 接收到的数据字节数的较低的 8 位。当 OutPktRdy 置 1 时，返回的值是有效的。

位域	类型	复位	名称	描述
[7:0]	R	0	OUTCOUNT1	接收字节数低八位

26.7.21 接收计数寄存器 OUTCOUNT2

它保存当前输出端点 FIFO 接收到的数据字节数的较高的 3 位。当 OutPktRdy 置 1 时，返回的值是有效的。

位域	类型	复位	名称	描述
[7:3]	--	--	--	--
[2:0]	R	0	OUTCOUNT2	接收字节数高三位

26.7.22 FIFO 寄存器 FIFOx

这个地址范围为 CPU 访问每个端点的 FIFOs 提供地址。写入这些地址将相应端点的数据加载到 IN FIFO 中；从这些地址读取相应端点 OUT FIFO 的数据。

位域	类型	复位	名称	描述
[7:0]	R	0	FIFOx	端点数据寄存器 读取：返回接收到 FIFO 的数据 写入：将数据写入 FIFO

27 修改历史

版本	修改日期	修改内容
V1.0	2020/8/14	初版
V1.1	2021/12/2	1) 修改已知文字错误