AD24N 用户手册

珠海市杰理科技股份有限公司

Zhuhai Jieli Technologyco.,LTD

2024年12月

目录

1 序言

AD24N系列芯片是低成本低功耗高性能通用语音MCU，本芯片主要面向智能语音玩具、智能家居设备控制以及通用型智能控制等场景。芯片集成了32位CPU自带ICACHE，内置52k byte数据空间，并可外接FLASH扩展存储空间；同时芯片内建低温票高精度时钟振荡器，无须外接晶振；集成高性能音频ADC与DAC，并自带音频功放模块可直接驱动小功率喇叭发声；PMU电源管理模块可灵活提供多种低功耗工作模式，支持各种超低功耗工作场景；此外还提供了FUSB、ADC、IIC、SPI、UART、SD、IIS等丰富的外设接口。

2 总体介绍

2.1 概述

SYSTEM

- 32bit Dual-Issue DSP 240MHz
- I-cache
- Support SDTAP/EMU
- On-chip SRAM 52kbyte(share cache ram 20k)
- Built-In Flash
- Internal RC oscillator,PLL

Audio

- 1 x 16bit DAC
  SNR 96dB
  Noise 9uVrms
  Sampling rate 8~96kHz
- 1 x 16bit Class-D Speaker Driver
  SNR 97dB
  Sampling rate 8~96kHz
  Drive speaker directly 500mW@4Ω
- 1 x 16bit ADC
  SNR 97dB
  Sampling rate 8~48kHz
  Support Speaker for microphone
- I^2S AUDIO Master/Slave interface

Peripherals

- 1 x Full speed USB
- 1 x SD host controller
- 3 x Multi-function 16bit timer
- 2 x UART interface
- 1 x I^2C Master/Slave interface
- 3 x SPI Master/Slave interface
- 4 x MCPWM
- 1 x GPCRC
- 1 x 10bit ADC(16 Channels)
- 22 x GPIO Support function remapping

PMU

- VPWR range 1.8V to 5.5V
- IOVDD range 2.7V to 3.6V

Packages

- QSOP24

Temperature

- Operating temperature
  TC = -20°C to +85°C(standard range)

TC = -40°C to +105°C(extended range)
- Storage temperature -65°C to +150°C

Applications

- Sound Toy
- Audio player

2.2 总体框图

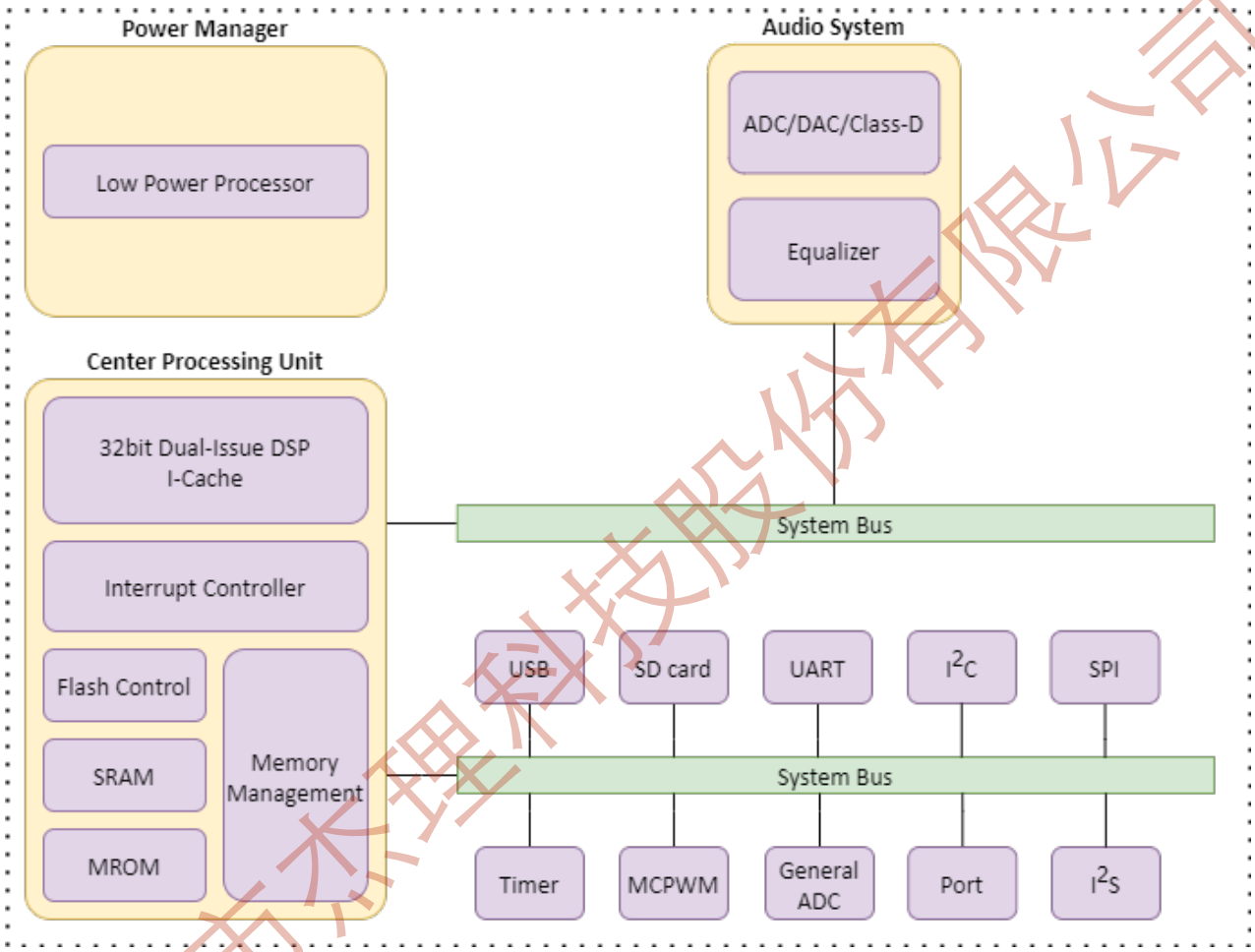


图1 AD24N总体框图

3 IO_CONTROL

3.1 概述

IO状态控制寄存器

3.2 数字模块控制寄存器

3.2.1 PORTX_IN：portx input control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_IN	r	-	-	获取portx组的输入值，只有在对应的GPIO设置为输入使能（DIE） 有效时数值才正确 0：输入为0 1：输入为1

3.2.2 PORTX_OUT：portx output control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_OUT	RW	0x0	-	控制portx组的输出值，只有在对应的GPIO设置为输出方向（DIR） 有效时输出数值才正确 0：输出为0 1：输出为1

3.2.3 PORTX_DIR：portx direction control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_DIR	RW	0x0	-	控制portx组的输入输出方向 0：设置为输出方向 1：设置为输入方向

3.2.4 PORTX_DIE：portx input enable control register：

Bit	Name	RW	Default	RV	Description

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_DIE	RW	0x0	-	控制portx组的输入使能 0：禁止输入使能 1：允许输入使能

3.2.5 PORTX_DIEH：portx input enable p33 control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_DIEH	RW	0x0	-	控制portx组的P33输入使能 0：禁止输入使能 1：允许输入使能

3.2.6 PORTX_PU0：portx pull up0 control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_PU0	RW	0x0	-	控制portx组的上拉电阻值，与PORTX_PU1组成4个档位

3.2.7 PORTX_PU1：portx pull up1 control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_PU0	RW	0x0	-	控制portx组的上拉电阻值，与PORTX_PU0组成4个档位{pu1,pu0} 00：无上拉电阻 01：10KΩ 10：100KΩ 11：1MΩ （mclr上拉电阻不受该位控制，由p33控制）

3.2.8 PORTX_PD0：portx pull down0 control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留

Bit	Name	RW	Default	RV	Description
15-0	PORTX_PD0	RW	0x0	-	控制portx组的下拉电阻值，与PORTX_PD1组成4个档位

3.2.9 PORTX_PD1：portx pull down1 control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_PD1	RW	0x0	-	控制portx组的下拉电阻值，与PORTX_PD0组成4个档位{pd1,pd0} 00：无下拉电阻 01：10KΩ 10：100KΩ 11：1MΩ

3.2.10 PORTX_HD0：portx high driver0 control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_HD0	RW	0x0	-	控制portx组的强驱动电流档位，与PORTX_HD1组成4个档位

3.2.11 PORTX_HD1：portx high driver1 control register：

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	0x0	-	预留
15-0	PORTX_HD1	RW	0x0	-	控制portx组的强驱动电流档位， 与PORTX_HD0组成4个档位{hd1,hd0}

3.2.12 PORTX_SPL：portx pull-down resistor enable in output control register:

Bit	Name	RW	Default	RV	Description
31-1	RESERVED	-	0x0	-	预留
0	SPL	RW	0x0	-	0：在IO为输出状态时，内部上下拉电阻无效，读取IO输入寄存器值总为0； 1：在IO为输出状态时，内部上下拉电阻功能有效，读取IO 输入寄存器值跟随IO实际电平逻辑；

3.2.13 PORTX_OUT_BSR：portx bit set resistor in output control register:

Bit	Name	RW	Default	RV	Description
31-16	OUT_BSR	W	0x0	-	设置portx组的输出0，只有在对应的GPIO设置为输出方向（DIR）有效时输出数值才正确 0：无效 1：有效
15-0	OUT_BSR	W	0x0	-	设置portx组的输出1，只有在对应的GPIO设置为输出方向（DIR）有效时输出数值才正确 0：无效 1：有效

3.2.14 PORTX_DIR_BSR： portx bit set resistor in dir control register:

Bit	Name	RW	Default	RV	Description
31-16	DIR_BSR	W	0x0	-	设置portx组对应GPIO的方向为输出 0：无效 1：有效
15-0	DIR_BSR	W	0x0	-	设置portx组对应GPIO的方向为输入 0：无效 1：有效

3.2.15 PORTX_DIE_BSR： portx bit set resistor in die control register:

Bit	Name	RW	Default	RV	Description
31-16	DIE_BSR	W	0x0	-	设置portx组对应GPIO的die为1 0：无效 1：有效
15-0	DIE_BSR	W	0x0	-	设置portx组对应GPIO的die为0 0：无效 1：有效

3.2.16 PORTX_DIEH_BSR： portx bit set resistor in dieh control register:

Bit	Name	RW	Default	RV	Description
31-16	DIEH_BSR	W	0x0	-	设置portx组对应GPIO的dieh为1 0：无效 1：有效
15-0	DIEH_BSR	W	0x0	-	设置portx组对应GPIO的dieh为0 0：无效 1：有效

3.2.17 PORTX_PU0_BSR： portx bit set resistor in pu0 control register:

Bit	Name	RW	Default	RV	Description

Bit	Name	RW	Default	RV	Description
31-16	PU0_BSR	W	0x0	-	设置portx组对应GPIO的pu0为1 0：无效 1：有效
15-0	PU0_BSR	W	0x0	-	设置portx组对应GPIO的pu0为0 0：无效 1：有效

3.2.18 PORTX_PU1_BSR： portx bit set resistor in pu1 control register:

Bit	Name	RW	Default	RV	Description
31-16	PU1_BSR	W	0x0	-	设置portx组对应GPIO的pu1为1 0：无效 1：有效
15-0	PU1_BSR	W	0x0	-	设置portx组对应GPIO的pu1为0 0：无效 1：有效

3.2.19 PORTX_PD0_BSR： portx bit set resistor in pd0 control register:

Bit	Name	RW	Default	RV	Description
31-16	PD0_BSR	W	0x0	-	设置portx组对应GPIO的pd0为1 0：无效 1：有效
15-0	PD0_BSR	W	0x0	-	设置portx组对应GPIO的pd0为0 0：无效 1：有效

3.2.20 PORTX_PD1_BSR： portx bit set resistor in pd1 control register:

Bit	Name	RW	Default	RV	Description
31-16	PD0_BSR	W	0x0	-	设置portx组对应GPIO的pd0为1 0：无效 1：有效
15-0	PD0_BSR	W	0x0	-	设置portx组对应GPIO的pd0为0 0：无效 1：有效

3.2.21 PORTX_HD0_BSR： portx bit set resistor in hd0 control register:

Bit	Name	RW	Default	RV	Description

Bit	Name	RW	Default	RV	Description
31-16	HD0_BSR	W	0x0	-	设置portx组对应GPIO的hd0为1 0：无效 1：有效
15-0	HD0_BSR	W	0x0	-	设置portx组对应GPIO的hd0为0 0：无效 1：有效

3.2.22 PORTX_HD1_BSR：portx bit set resistor in hd0 control register:

Bit	Name	RW	Default	RV	Description
31-16	HD1_BSR	W	0x0	-	设置portx组对应GPIO的hd1为1 0：无效 1：有效
15-0	HD1_BSR	W	0x0	-	设置portx组对应GPIO的hd1为0 0：无效 1：有效

3.2.23 PORTX_SPL_BSR：portx bit set resistor in spl control register:

Bit	Name	RW	Default	RV	Description
31-16	SPL_BSR	W	0x0	-	设置portx组对应GPIO的spl为1 0：无效 1：有效
15-0	SPL_BSR	W	0x0	-	设置portx组对应GPIO的spl为0 0：无效 1：有效

3.2.24 PORTX_CON_BSR：portx bit set resistor in con control register(USB_IO only):

Bit	Name	RW	Default	RV	Description
31-16	SPL_CON	W	0x0	-	设置USB组IO的CON对应的BIT为1 0：无效 1：有效
15-0	SPL_CON	W	0x0	-	设置USB组IO的CON对应的BIT为0 0：无效 1：有效

3.3 USB_IO CONTROL

1. JL_PORTUSB->DIE：

2. JL_PORTUSB->DIEH：

3. JL_PORTUSB->DIR：

4. JL_PORTUSB->PU0：DP上拉1.5K，DM上拉180K(PU1没有用到)

5. JL_PORTUSB->PD0：DP 下拉15K，DM下拉15K(PD1没有用到)

6. JL_PORTUSB->IN：

7. JL_PORTUSB->OUT：

8. JL_PORTUSB->SPL：
   上述USB IO寄存器跟普通IO寄存器功能一样，不过USB 只有两个IO,BIT(0)设置DP，BIT(1)设置DM

9. JL_PORTUSB->CON：usb io control register 0

Bit	Name	RW	Default	RV	Description
31-13	RESERVED	-	-	-	预留
12-11	DBG_SEL	rw	0x0	-	测试信号选择： 00：1'b0 0 1：usb_diff 10：usb_dp 11：usb_dm
10	USB_CP_MODE	r'w	-	-	测试信号使能
9	USB_DF_MODE	rw	-	-	测试信号使能
8	IO_MODE	rw	0x1	-	IO模式使能 0：USB模式 1：普通IO模式
7	CHKDPO	r	0x1	-	CHKDPO经系统时钟采样后的输入
6	DIDF	r	0	-	差分输入DIDF经lsb_clk采样后的输入
5	HDEN	-	-	-	驱动能力选择 0:low drive mode 1:high drive mode
4	PDCHKDP	rw	0	-	DP 外接下拉检查使能 0：disable 1：enable
3	DM_ADCEN	-	-	-	dm to adc en
2	SR0	rw	0x0	-	输出驱动能力选择
1	DP_ADCEN	rw	0	-	dp to adc en
0	RCVEN	rw	0	-	USB_PHY使能（差分输入使能，使用USB功能时需打开这一BIT）

3.4 IO模拟状态设置

当IO用于模拟功能时，需要正确设置IO寄存器，确保模拟功能不受IO影响。正确配置如下：

如将PA0设置为模拟功能。

```
JL_PORTA->DIR |= BIT(0);
JL_PORTA->DIE &= ~BIT(0);
JL_PORTA->DIEH &= ~BIT(0);
JL_PORTA->PU0 &= ~BIT(0);
JL_PORTA->PU1 &= ~BIT(0);
JL_PORTA->PD0 &= ~BIT(0);
JL_PORTA->PD1 &= ~BIT(0);
```

4 IO_WAKEUP

4.1 概述

IO Wakeup是一个异步事件唤醒模块，它可以捕获IO 的上升沿/下降沿，将处于standby/sleep状态下的系统唤醒，进入正常工作模式。当系统处于正常模式时，Wakeup唤醒源可作为中断触发源，最多可有32个唤醒来源(具体数目以下述Wakeup唤醒源为准);

4.2 Wakeup唤醒源

- 事件0：PA0

  事件1： PA1

  事件2：PA2

  事件3：PA3

  事件4：PA4

  事件5：PA5

  事件6：PA6

  事件7：PA7

  事件8：PA8

  事件9：PA9

  事件10：PA10

  事件11：PA11

  事件12：PA12

  事件13：PA13

  事件14：PA14

  事件15：PA15

  事件16：PB0

  事件17：PB1

  事件18：PB2

  事件19：USBDP

  事件20：USBDM

  事件21：无

  事件22：FSPG(PP1)

  事件23：GP_ICH0

事件24：GP_ICH1

事件25：GP_ICH2

事件26：GP_ICH3

事件27：无

事件28：无

事件29：无

事件30：无

事件31：无

唤醒源数量：32

4.3 寄存器

4.3.1 WKUP_CON0: wakeup enable control register

偏移地址：0x00

Bit	Name	RW	Default	Description
31-n	RESERVED	-	-	-
n-0	WKUP_EN	rw	0x0	对应唤醒源使能 0: 关闭唤醒功能 1: 打开唤醒功能

n 为唤醒源数量

4.3.2 WKUP_CON1：wakeup edge control register

偏移地址：0x04

Bit	Name	RW	Default	Description
31-n	RESERVED	-	-	-
n-0	WKUP_EDGE	rw	0x0	对应唤醒源边沿选择 0: 上升沿唤醒; 1: 下降沿唤醒

n 为唤醒源数量

4.3.3 WKUP_CON2：wakeup pending clear control

偏移地址：0x08

Bit	Name	RW	Default	Description

Bit	Name	RW	Default	Description
31-n	RESERVED	-	-	-
n-0	WKUP_CPND	w	0x0	写1清除对应唤醒源 pending

n 为唤醒源数量

4.3.4 WKUP_CON3： wakeup pending register

偏移地址：0x0c

Bit	Name	RW	Default	Description
31-n	RESERVED	-	-	-
n-0	WKUP_PND	r	0x0	对应唤醒源 pending

n 为唤醒源数量

5 IOMC

5.1 概述

IOMC（IO remapping control）控制寄存器主要用于控制IO除crossbar之外的一些拓展功能的配置，每个外设类型都具有相应的独立IOMC控制寄存器；

项目存在两个基地址不同的IOMC，分别为JL_IOMC和JL_SFC_IOMC，前者用于处理通用外设的拓展功能，后者只用于SFC/spi0处于flash相关的功能。

5.2 JL_IOMC控制寄存器

5.2.1 SPI_IOMC0: spi iomc control register 0

偏移地址: 0x14

Bit	Name	RW	Default	Description
31	Reserved	-	-	Reserved
30-27	Reserved	-	-	Reserved
26-24	Reserved	-	-	Reserved
23	SPI2_DUPLEX	rw	0	SPI2从机输入时钟选择 0: spi2_clki; 1: spi1_clko
22-20	SPI2_ICK_SEL	rw	0	SPI2输入时钟延迟级数选择
18-16	Reserved	-	-	Reserved
15-13	Reserved	-	-	Reserved
12	Reserved	-	-	Reserved
11-8	Reserved	-	-	Reserved
7	Reserved	-	-	Reserved
6-3	Reserved	-	-	Reserved
2-0	SPI0_IOS	rw	0	SPI0 固定IO 组选择

5.2.2 SDC_IOMC0: sdc iomc control register 0

偏移地址: 0x18

Bit	Name	RW	Default	Description
31	Reserved	-	-	Reserved
30-27	Reserved	-	-	Reserved
26-24	Reserved	-	-	Reserved
23	Reserved	-	-	Reserved
22-19	Reserved	-	-	Reserved
18-16	Reserved	-	-	Reserved
15	Reserved	-	-	Reserved
14-10	Reserved	-	-	Reserved
11	Reserved	-	-	Reserved
10-8	Reserved	-	-	Reserved
7-2	Reserved	-	-	Reserved
3	Reserved	-	-	Reserved
2-0	Reserved	-	-	Reserved

5.2.3 sys iomc control register 0

偏移地址: 0x1c

Bit	Name	RW	Default	Description
31-12	Reserved	-	-	Reserved
11	Reserved	-	-	Reserved
10	Reserved	-	-	Reserved
9	Reserved	-	-	Reserved
8	Reserved	-	-	Reserved
7	Reserved	-	-	Reserved
6	Reserved	-	-	Reserved
5	Reserved	-	-	Reserved
4	Reserved	-	-	Reserved
3	Reserved	-	-	Reserved

Bit	Name	RW	Default	Description
2	Reserved	-	-	Reserved
1-0	Reserved	-	-	Reserved

5.2.4 sys iomc control register 1

偏移地址: 0x20

Bit	Name	RW	Default	Description
31	Reserved	-	-	Reserved
30-27	Reserved	-	-	Reserved
26	Reserved	-	-	Reserved
25-24	Reserved	-	-	Reserved
23-1	Reserved	-	-	Reserved
0	SFC_IOS	rw	0	SFC 固定IO 组别选择

5.2.5 ASS_IOMC0: audio iomc control register 0

偏移地址: 0x38

Bit	Name	RW	Default	Description
31	Reserved	-	-	Reserved
31-26	ASS_OCH1_sSEL	rw	0	
25	ASS_OCH1_REG	rw	0	
24	ASS_OCH1_EN	rw	0	
23-18	ASS_OCH0_SEL	rw	0	
17	ASS_OCH0_REG	rw	0	
16	ASS_OCH0_EN	rw	0	
15	Reserved	-	-	Reserved
14-11	Reserved	-	-	Reserved
10-8	Reserved	-	-	Reserved
7	Reserved	-	-	Reserved

Bit	Name	RW	Default	Description
6-3	Reserved	-	-	Reserved
2-0	Reserved	-	-	Reserved

5.2.6 WL_IOMC0: wireless iomc control register 0

偏移地址: 0x3c

Bit	Name	RW	Default	Description
31	Reserved	-	-	Reserved
30-27	Reserved	-	-	Reserved
26-24	Reserved	-	-	Reserved
23	Reserved	-	-	Reserved
22-19	Reserved	-	-	Reserved
18-16	Reserved	-	-	Reserved
15	Reserved	-	-	Reserved
14-11	Reserved	-	-	Reserved
10-8	Reserved	-	-	Reserved
7-6	Reserved	-	-	Reserved
5	Reserved	-	-	Reserved
4	Reserved	-	-	Reserved
3	Reserved	-	-	Reserved
2-0	Reserved	-	-	Reserved

5.2.7 ICH_IOMC0: input channel control register0

偏移地址: 0x40

Bit	Name	RW	Default	Description
31-30	Reserved	-	-	
29-25	ICH5_SEL (TMR2_CAP)	rw	0	同ICH0选项

Bit	Name	RW	Default	Description
24-20	ICH4_SEL (TMR1_CAP)	rw	0	同ICH0选项
19-15	ICH3_SEL (TMR0_CAP)	rw	0	同ICH0选项
14-10	ICH2_SEL (TMR2_CIN)	rw	0	同ICH0选项
9-5	ICH1_SEL (TMR1_CIN)	rw	0	同ICH0选项
4-0	ICH0_SEL (TMR0_CIN)	rw	0	功能输入通道选择 0-4: gp_ich0-gp_ich3 6: tmr1_pwm 7: tmr2_pwm

5.2.8 ICH_ IOMC1: input channel control register1

偏移地址: 0x44

Bit	Name	RW	Default	Description
31-30	Reserved	-	-	
29-25	ICH11_SEL (PLNK_IDAT0)	rw	0	同ICH0选项
24-20	ICH10_SEL (UART1_CTS)	rw	0	同ICH0选项
19-15	ICH9_SEL (MC_PWM1_FP)	rw	0	同ICH0选项
14-10	ICH8_SEL (MC_PWM0_FP)	rw	0	同ICH0选项
9-5	ICH7_SEL (MC_PWM1_CK)	rw	0	同ICH0选项
4-0	ICH6_SEL (MC_PWM0_CK)	rw	0	同ICH0选项

5.2.9 ICH_ IOMC2: input channel control register2

偏移地址: 0x48

Bit	Name	RW	Default	Description
31-30	Reserved	-	-	
29-25	ICH17_SEL (SPI0_CSI)	rw	0	同ICH0选项
24-20	ICH16_SEL (AUD_DBG_DATI)	rw	0	同ICH0选项
19-15	ICH15_SEL (EXT_CLK)	rw	0	同ICH0选项

Bit	Name	RW	Default	Description
14-10	ICH14_SEL (CLK_MUX_IN)	rw	0	同ICH0选项
9-5	ICH13_SEL (CAP_MUX_OUT)	rw	0	同ICH0选项
4-0	ICH12_SEL (PLNK_IDAT1)	rw	0	同ICH0选项

5.2.10 ICH_ IOMC3: input channel control register3

偏移地址: 0x4c

Bit	Name	RW	Default	Description
31-30	Reserved	-	-	
29-25	ICH23_SEL (Reserve)	rw	0	同ICH0选项
24-20	ICH22_SEL (Reserve)	rw	0	同ICH0选项
19-15	ICH21_SEL (Reserve)	rw	0	同ICH0选项
14-10	ICH20_SEL (Reserve)	rw	0	同ICH0选项
9-5	ICH19_SEL (SPI2_CSI)	rw	0	同ICH0选项
4-0	ICH18_SEL (SPI1_CSI)	rw	0	同ICH0选项

5.2.11 ICH_ IOMC4: input channel control register4

偏移地址: 0x50

Bit	Name	RW	Default	Description
31-30	Reserve	-	-	
29-25	ICH29_SEL (Reserve)	rw	0	同ICH0选项
24-20	ICH28_SEL (Reserve)	rw	0	同ICH0选项
19-15	ICH27_SEL (Reserve)	rw	0	同ICH0选项
14-10	ICH26_SEL (Reserve)	rw	0	同ICH0选项
9-5	ICH25_SEL (Reserve)	rw	0	同ICH0选项
4-0	ICH24_SEL (Reserve)	rw	0	同ICH0选项

5.2.12 ICH_ IOMC5: input channel control register5

偏移地址: 0x54

Bit	Name	RW	Default	Description
31-20	Reserved	-	-	
19-15	ICH33_SEL (Reserved)	rw	0	同ICH0选项
14-10	ICH32_SEL (Reserved)	rw	0	同ICH0选项
9-5	ICH31_SEL (Reserved)	rw	0	同ICH0选项
4-0	ICH30_SEL (Reserved)	rw	0	同ICH0选项

5.2.13 OCH_IOMC0: output channel control register0

偏移地址: 0x58

Bit	Name	RW	Default	Description
31-24	OCH3_SEL	rw	0	同OCH0选项
23-16	OCH2_SEL	rw	0	同OCH0选项
15-8	OCH1_SEL	rw	0	同OCH0选项
7-0	OCH0_SEL	rw	0	通用输出通道选择 0: tmr0_pwm 1: tmr1_pwm 2: tmr2_pwm 3: gp_ich0 4: gp_ich1 5: uart1_rts 6: plnk_clk 7: aud_dbg_clko 8: aud_dbg_dato[0] 9: aud_dbg_dato[1] 10: aud_dbg_dato[2] 11: aud_dbg_dato[3] 12: aud_dbg_dato[4] 13: clk_out2io[0] 14: 1'b0 15: clk_out2io[2] 16: p33_clk_dbg 17: p33_sig_dbg[0] 18: p33_sig_dbg[1] 19: usb_dbg_out 20: 1'b0

5.2.14 OCH_IOMC1: output channel control register1

偏移地址: 0x5c

Bit	Name	RW	Default	Description
31-24	OCH7_SEL	-	-	Reserved
23-16	OCH6_SEL	-	-	Reserved
15-8	OCH5_SEL	-	-	Reserved
7-0	OCH4_SEL	-	-	Reserved

5.2.15 OCH_IOMC2/3/4/5: output channel control register2/3/4/5

偏移地址: 0x60/0x64/0x68/0x6c

Bit	Name	RW	Default	Description
31-24	OCH11_SEL/OCH15_SEL/OCH19_SEL/OCH23_SEL	-	-	Reserved
23-16	OCH10_SEL/OCH14_SEL/OCH18_SEL/OCH22_SEL	-	-	Reserved
15-8	OCH9_SEL/OCH13_SEL/OCH17_SEL/OCH21_SEL	-	-	Reserved
7-0	OCH8_SEL/OCH12_SEL/OCH16_SEL/OCH20_SEL	-	-	Reserved

5.2.16 FSPG_CON: FSPG control register

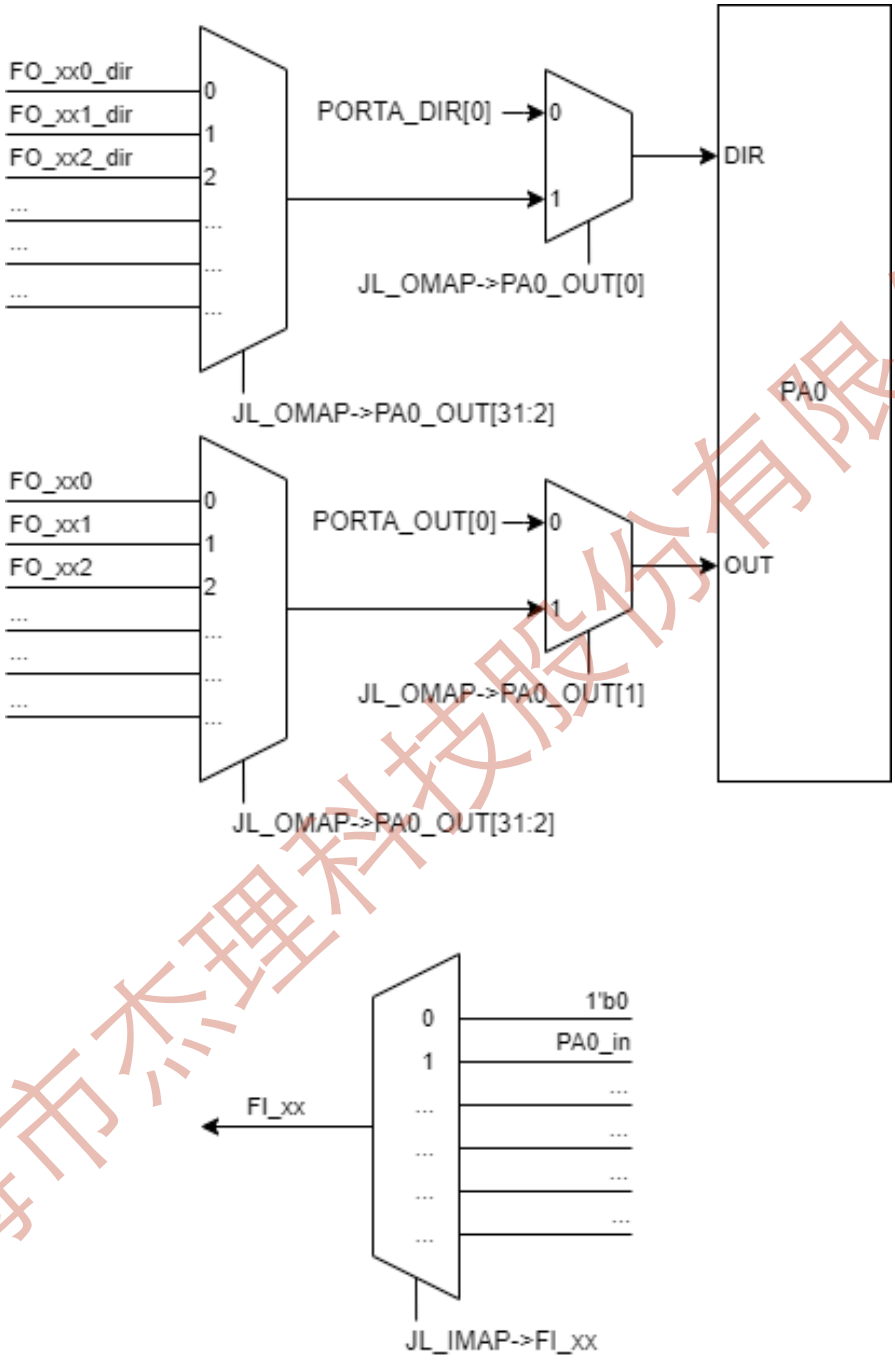
偏移地址: 0x00

Bit	Name	RW	Default	Description
31-3	Reserved	-	-	Reserved
2	SOFT	rw	0	FSPG 驱动强度选择，为flash供电时，上下电速度的控制 00: PAD高阻 0: 快上/下电 (tr/tf≤2us) 1: 慢上/下电 (tr/tf≥20us)
1	CS1_EN	rw	0	是否短接CS1到FSPG 该功能用于FLASH上/下电时避免电流通过CS PIN倒灌

Bit	Name	RW	Default	Description
0	CS0_EN	rw	0	是否短接CS0到FSPG该功能用于FLASH上/下电时避免电流通过CS PIN倒灌

6 PORT_Crossbar

6.1 Crossbar功能结构图



6.2 寄存器功能

6.2.1 JL_OMAP-> POUT

Bit	Name	RW	Description

Bit	Name	RW	Description
[0]	FUN_DIR_EN	rw	Crossbar 方向控制使能： 0：CPU 控制 1：外设控制 （注：若所选外设无方向控制功能，则都由 CPU 控制）
[1]	FUN_OUT_EN	rw	Crossbar 数据控制使能： 0：CPU 数据 1：外设数据
[31:2]	FUN_SELECT	rw	Crossbar 输出功能选择： （见功能表或头文件）
[31:0]	INPUT_SELECT	rw	Crossbar 输入引脚选择： （见功能表或头文件）

6.2.2 JL_IMAP-> FUN_IN

Bit	Name	RW	Description
[31:0]	INPUT_SELECT	rw	Crossbar 输出引脚选择： （见功能表或头文件）

7 CLOCK_SYSTEM

7.1 原生时钟源

本芯片具备的原生时钟源如下表所示：

时钟	概述
rc_250k	内置RC振荡器，用于复位系统和watch dog功能时钟
rc_16m	内置RC振荡器，用于系统启动的初始时钟。
lrc_200k	内置低温度电压漂移RC振荡器，用于低功耗计数和PLL参考时钟。

7.2 衍生时钟源

衍生时钟源于芯片内部的PLL

7.2.1 内部PLL 参考时钟源

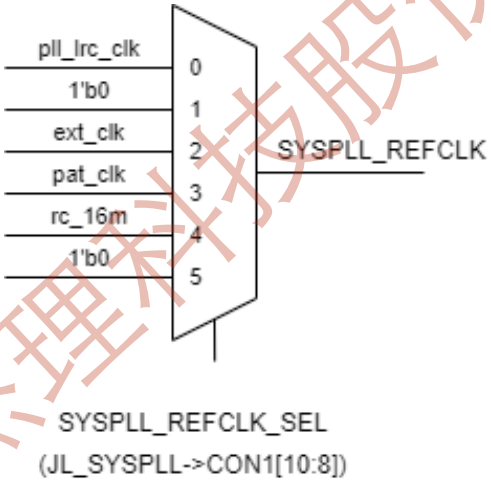
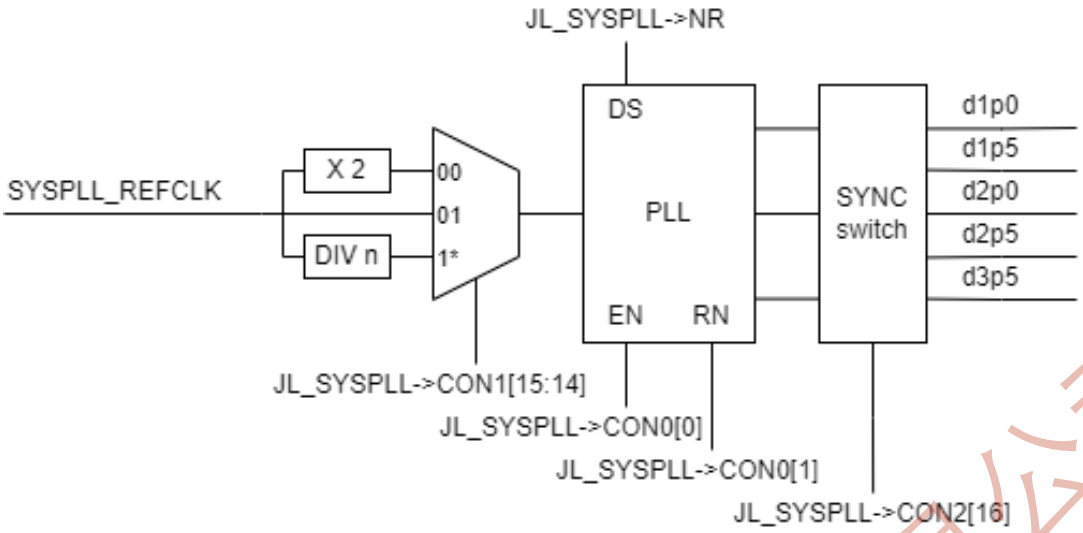


图1 内部PLL 参考时钟源

7.2.2 SYSPLL

SYSPLL工作范围覆盖160-240MHz，同时两者均输出
PLL_D1P0，PLL_D1P5，PLL_D2P0，PLL_D2P5，PLL_D3P5的时钟，如果SYSPLL配置为240MHz，则能输出240MHz、160MHz、120MHz、96MHz和68.5MHz；每个时钟都有独立的使能端。在不使用该时钟时，软件应关闭其使能端以减少额外电源消耗。



Fout = Fref / n * m; （n = SYSPLL_CON1[6:0]+2，m = SYSPLL_NR[11:0]+2）

Fout 可以为 160-240 MHz，建议值为192MHz

图2 SYSPLL时钟结构框图

7.3 标准时钟

除上述的原生时钟外，芯片内还有频率固定的标准时钟，分别是wclk、pll_96m、std_48m、std_24m和std_12m，其时钟源头选择如下：

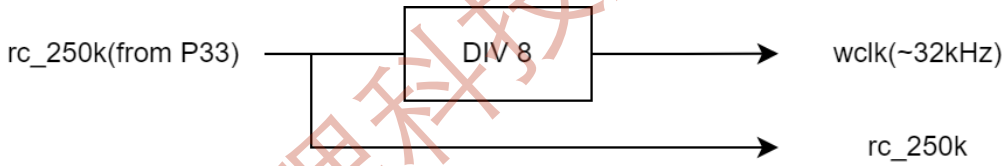


图3 系统 rc_250k 和 wclk 时钟结构

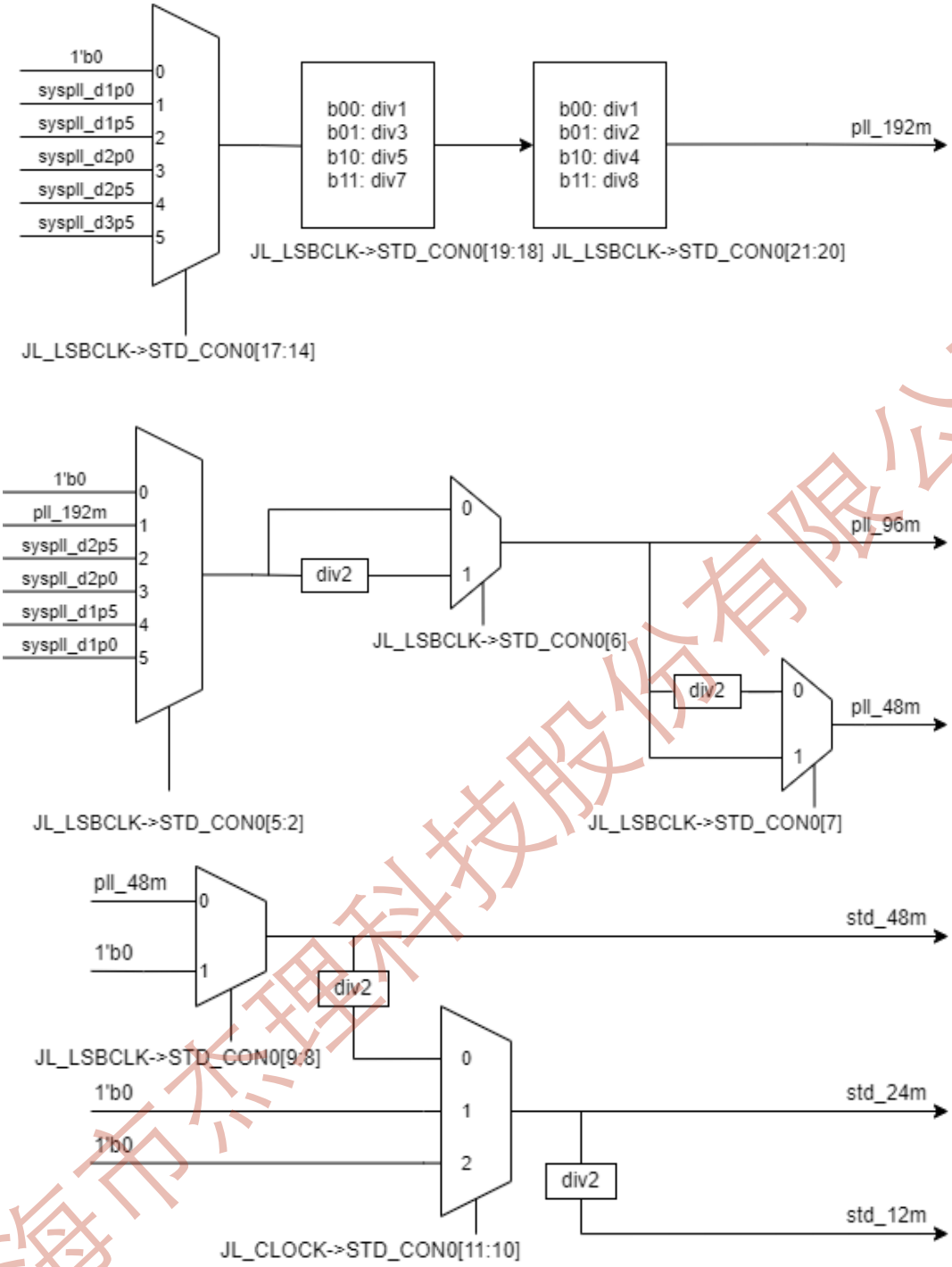


图4 STD_CLK时钟结构图

7.4 系统时钟

系统时钟部分电路框图如下，系统运行过程中可随时更改hsb_clk，lsb_clk的分频值，但需确保在任何时刻，各分频时钟都不会超过其允许的最高运行频率。

7.4.1 HSB时钟源

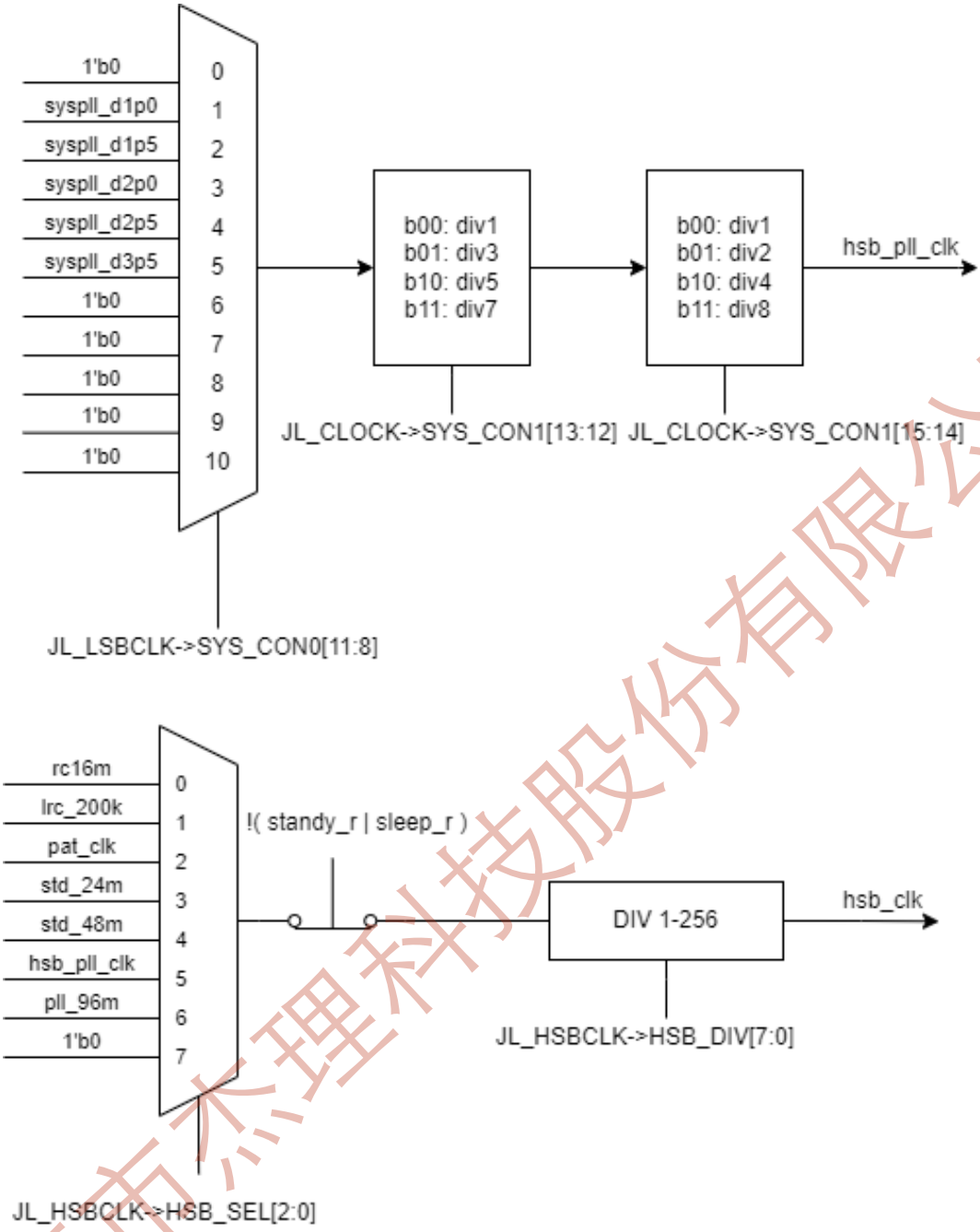


图5 hsb_clk时钟结构图

7.4.2 LSB时钟源

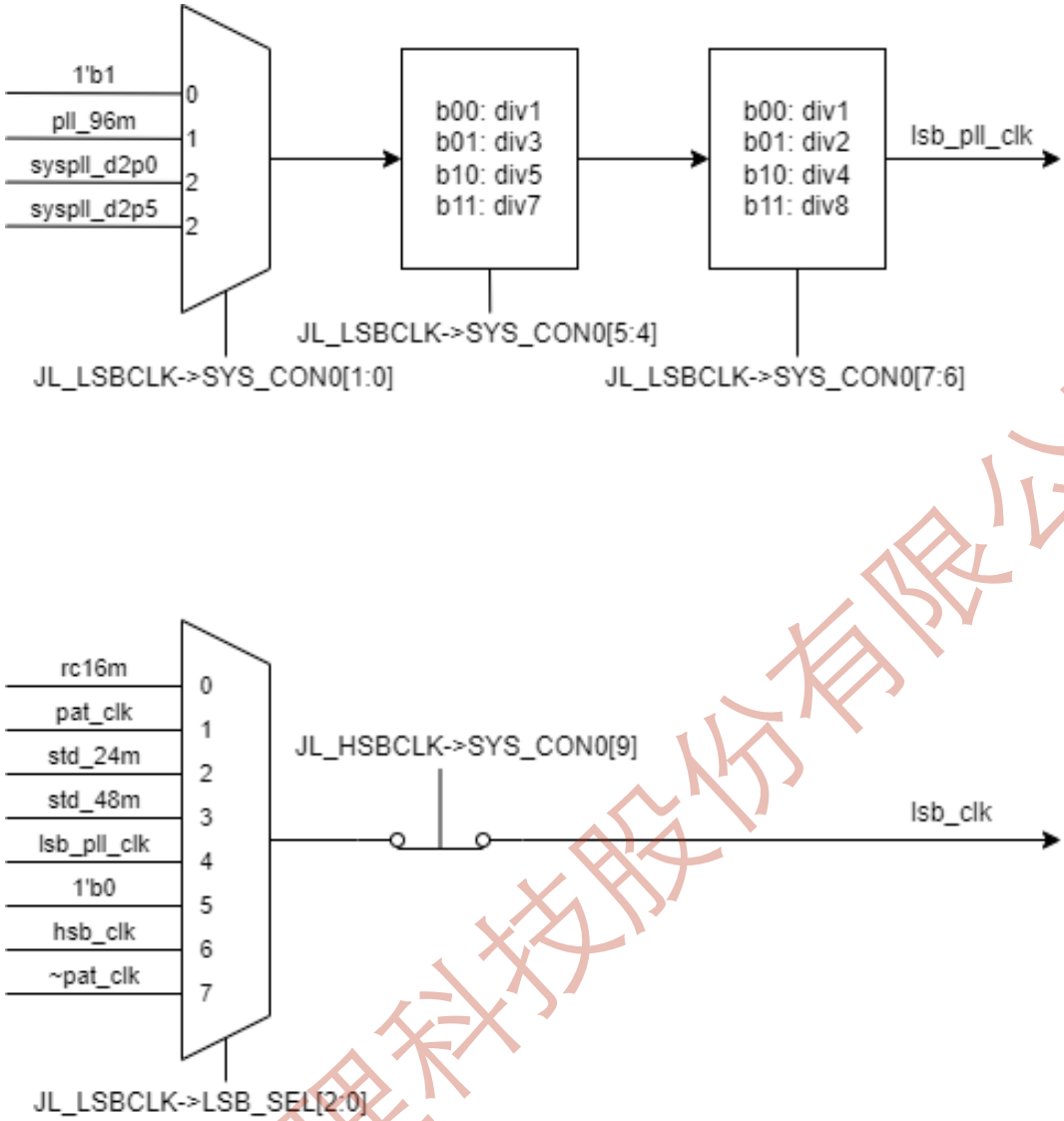


图6 lsb_clk时钟电路

【注意】：LSB_SEL 不能连续切换，两次切换中间需要等待5个LSB_CLK；

7.5 外设时钟

7.5.1 SFC 时钟源

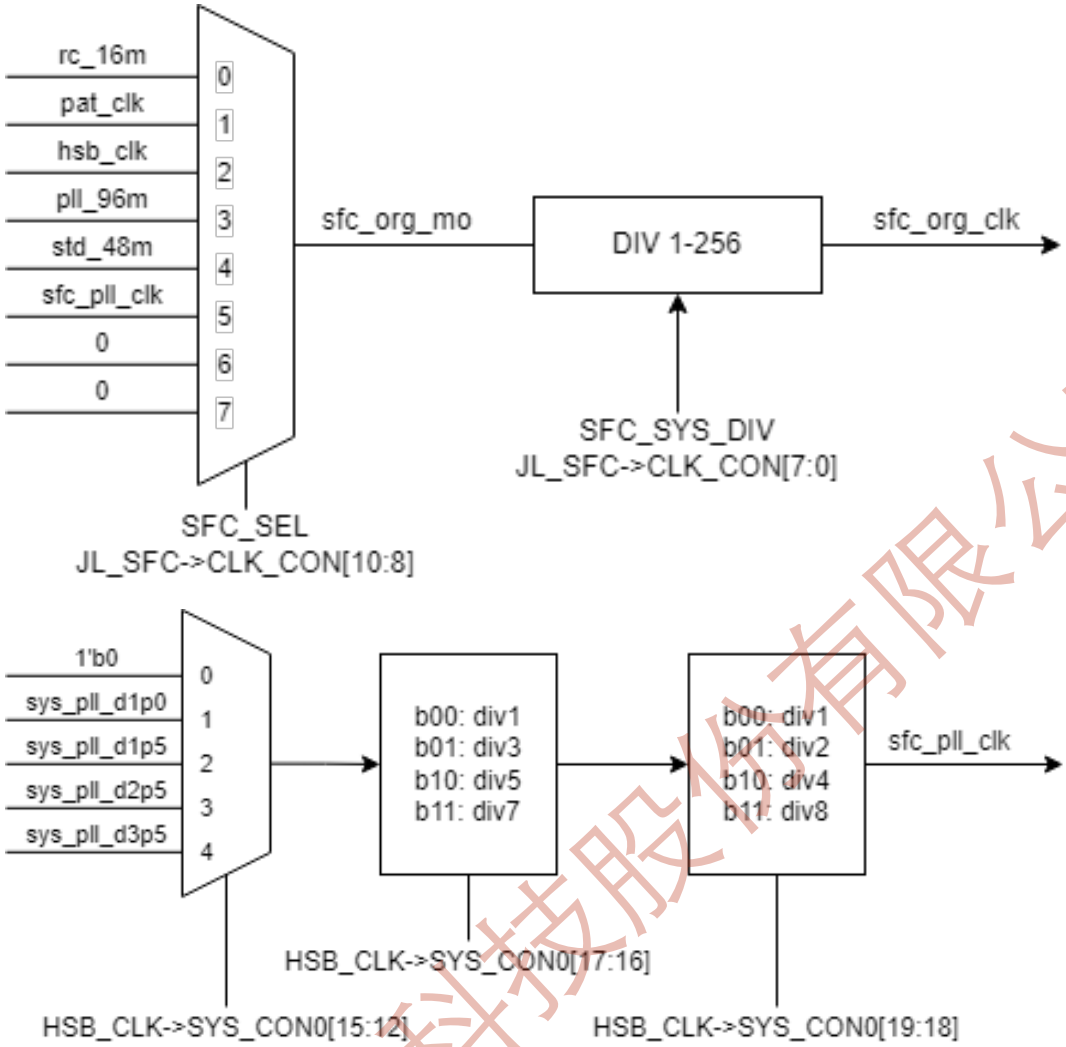
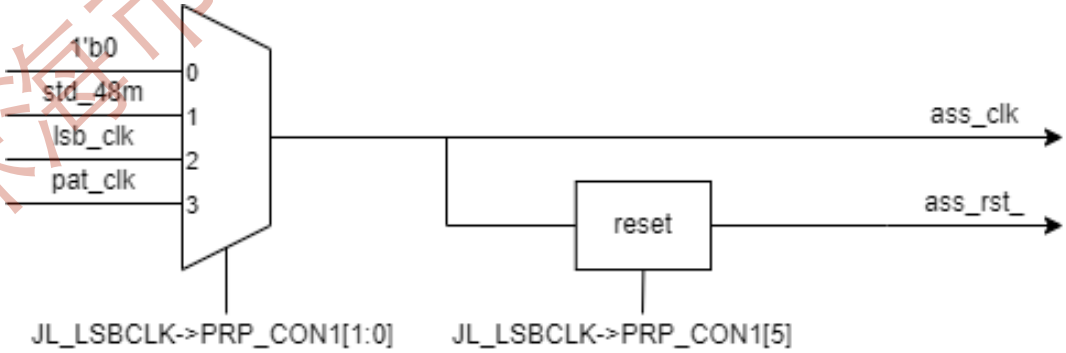


图7 SFC时钟结构图

【注意】：SFC_CLK可动态切换，但不能连续切换，两次切换中间需要等待大于5个SFC_CLK； 【注意2】： sfc_pll_clk的时钟与sfc_org_clk的控制寄存器的基地址不同

7.5.2 音频时钟源



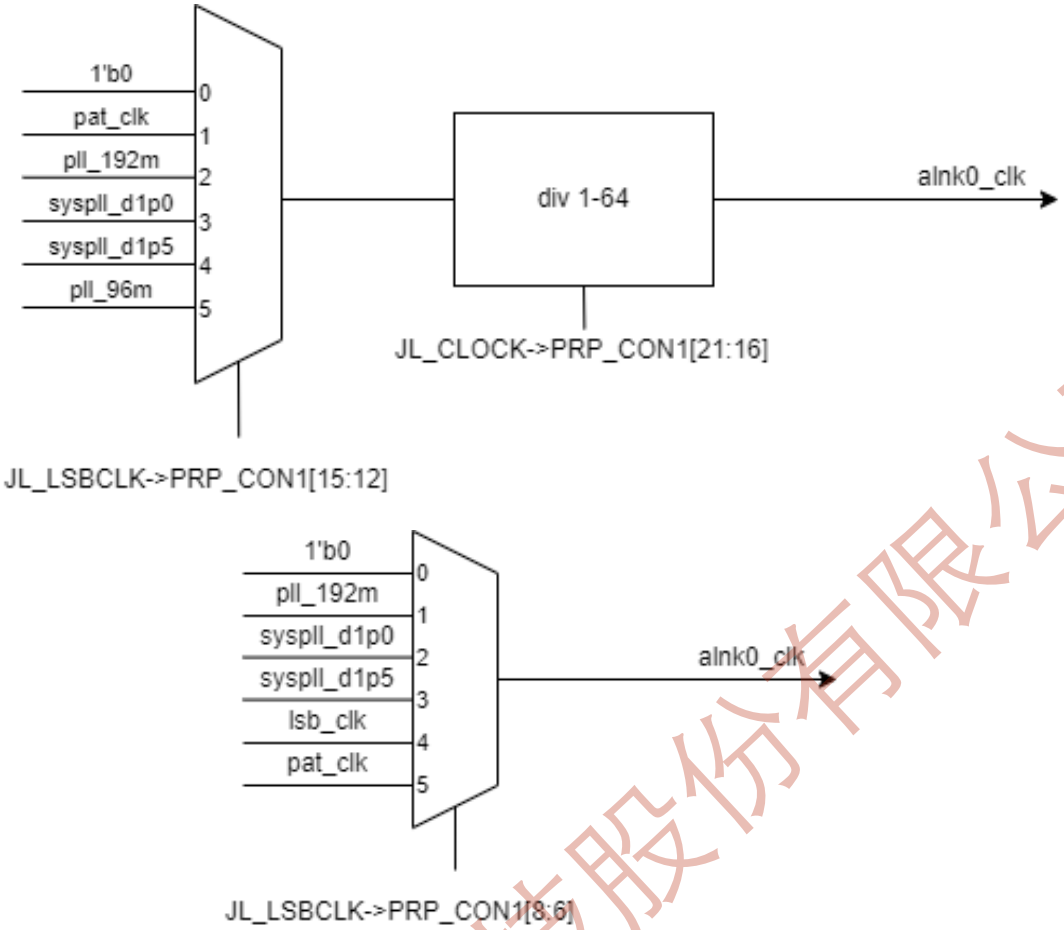
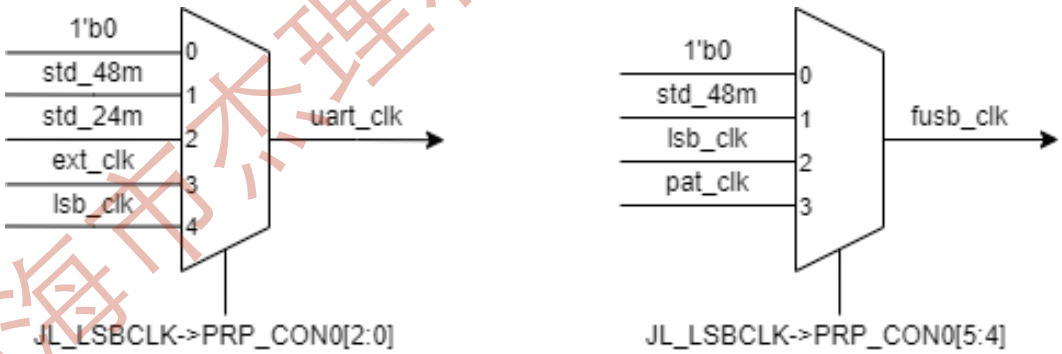


图8 音频时钟结构图

7.5.3 外设时钟源



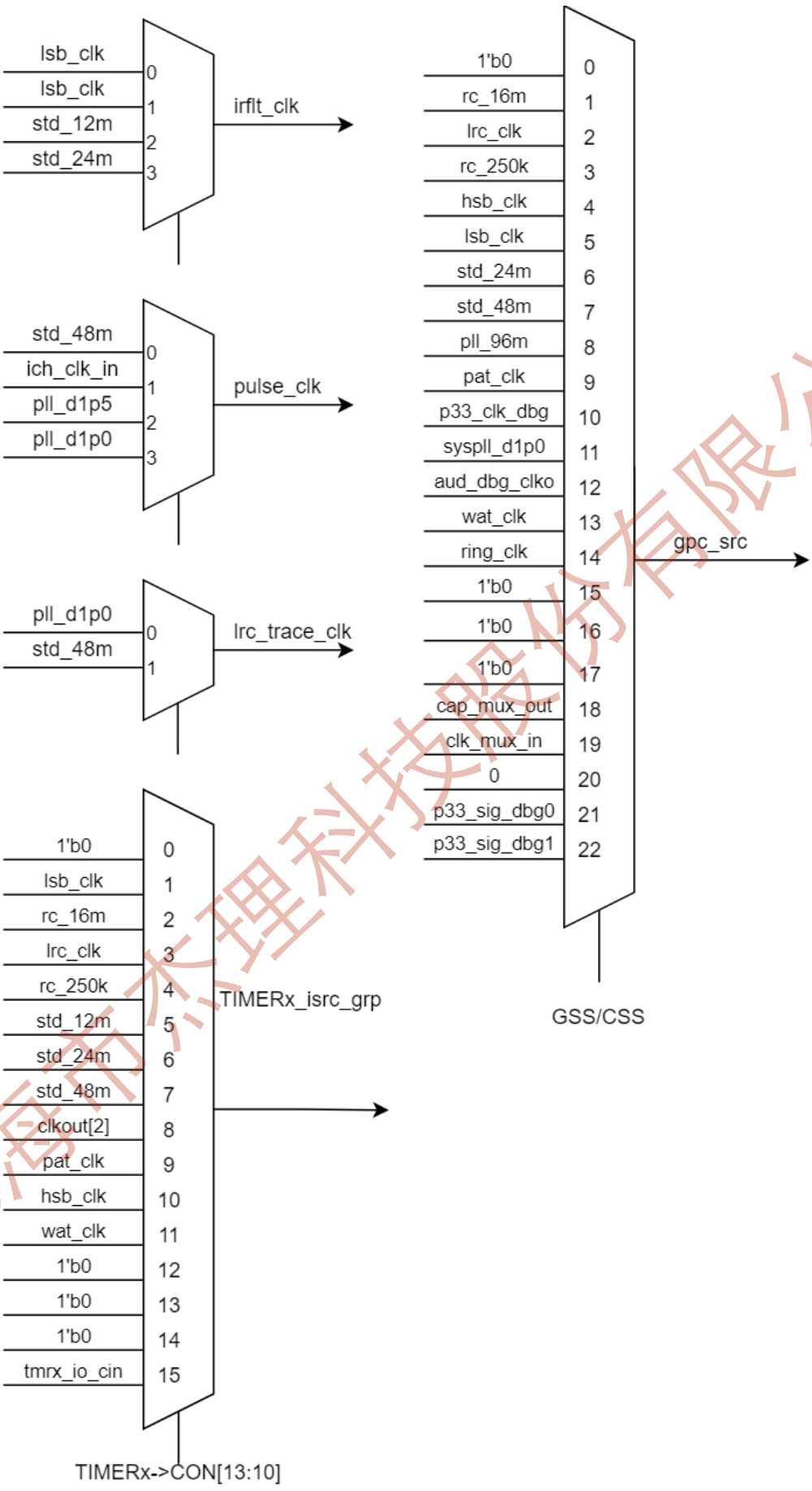


图10 低速外设时钟结构图

- 该芯片有2组时钟输出，可以通过crossbar将内部的时钟信号驱动至片外，用于测试或特殊用途。

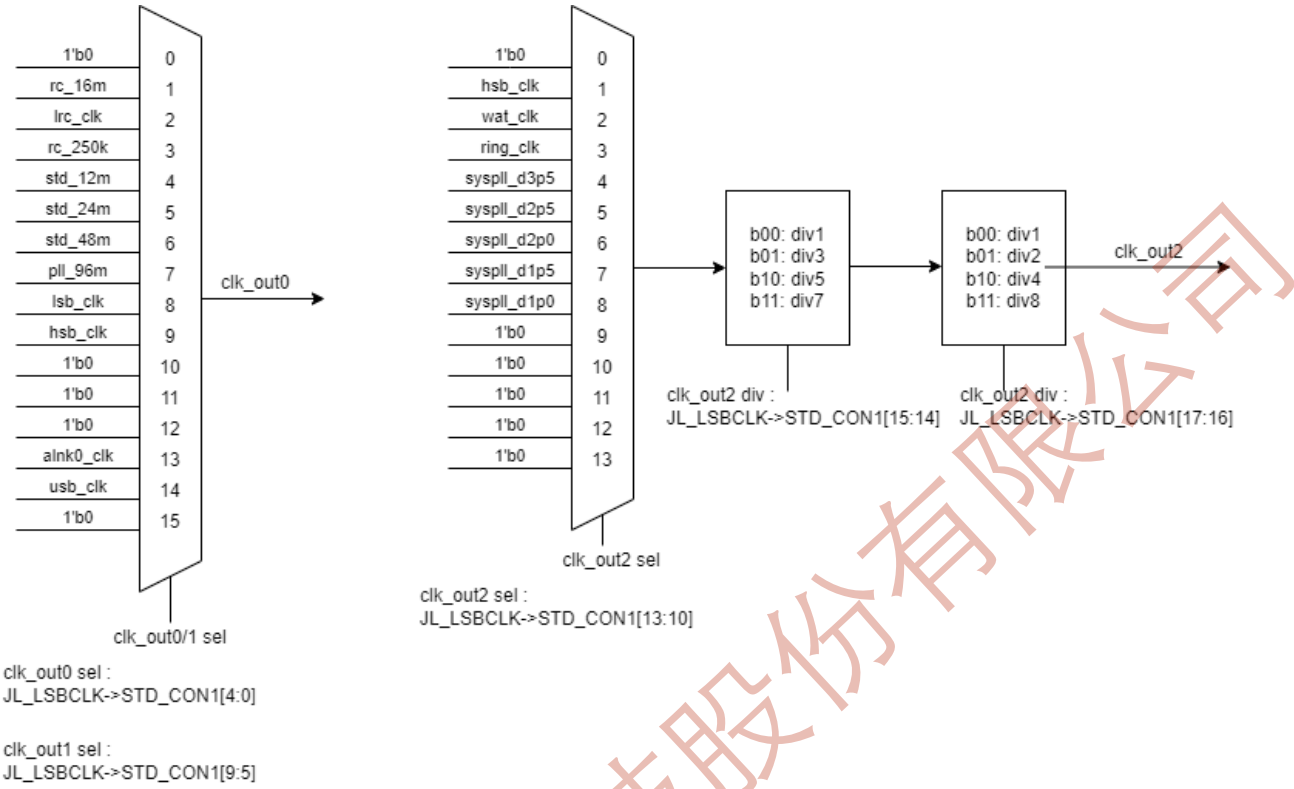


图11 IO输出时钟结构图

7.6 数字模块控制寄存器

7.6.1 JL_LSBCK->SYS_CON0：lsbck system config register 0

Bit	Name	RW	Default	RV	Description
31-11	RESERVED	-	0	-	预留
15-12	HSB_PLL_DIV	rw	0	-	HSB_PLL时钟分频，见图5
11-8	HSB_PLL_SEL	rw	0	-	HSB_PLL时钟选择，见图5
7-4	LSB_PLL_DIV	rw	0	-	LSB_PLL时钟分频，见图6
3-0	LSB_PLL_SEL	rw	0	-	LSB_PLL时钟选择，见图6

7.6.2 LSB_SEL：lsb clock sel register

Bit	Name	RW	Default	RV	Description
31-20	RESERVED	-	0	-	预留
2-0	LSB_SEL	rw	0	-	LSB时钟选择，见图6

7.6.3 JL_LSBCK->STD_CON0： standard clock register 0

Bit	Name	RW	Default	RV	Description
31-9	RESERVED	-	0	-	预留
21-18	PLL_192M_DIV	rw	0	-	PLL_192M分频系数，见图4
17-14	PLL_192M_SEL	rw	0	-	PLL_192M输入时钟源选择，见图4
13-12	RESERVED	-	0	-	预留
11-10	STD_24M_SEL	rw	0	-	STD_24M来源选择，见图4
9-8	STD_48M_SEL	rw	0	-	STD_48M来源选择，见图4
7	PLL_48M_DS	rw	0	-	PLL_48M DIV2使能选择，见图4
6	PLL_96M_DS	rw	0	-	PLL_96M DIV2使能选择，见图4
5-2	PLL_96M_SEL	rw	0	-	PLL_96M输入时钟源选择，见图4
1	RC_250K_EN	rw	1	-	片内RC（250k）振荡器使能（PMU低功耗模式无效） 0: 关闭片内RC振荡器 1: 打开片内RC振荡器
0	RC_16M_EN	rw	1	-	片内RC（16M）振荡器使能（PMU低功耗模式无效） 0: 关闭片内RC振荡器 1: 打开片内RC振荡器

7.6.4 JL_LSBCK->STD_CON1： standard clock register 1

Bit	Name	RW	Default	RV	Description
31-20	RESERVED	-	-	-	预留
17-14	CLKOUT2_DIV	rw	1	-	CLK_OUT2输出时钟分频选择，见图11
13-10	CLKOUT2_SEL	rw	0	-	CLK_OUT2输出时钟源选择，见图11
9-5	RESERVED	-	-	-	预留
4-0	CLKOUT0_SEL	rw	0	-	CLK_OUT0输出时钟选择，见图11

7.6.5 JL_LSBCK->STD_CON2： standard clock register 2

Bit	Name	RW	Default	RV	Description

Bit	Name	RW	Default	RV	Description
31-0	RESERVED	-	-	-	预留

7.6.6 JL_LSBCK->PRP_CON0： peripheral clock config register 0

Bit	Name	RW	Default	RV	Description
31-10	RESERVED	-	0	-	预留
9	RINGOSC_CKEN	rw	0	-	ring_clk使能
8-7	MBIST_CKSEL	rw	0	-	IC内部测试功能，MBIST时钟源
6	USB_CKPOL	rw	0	0	IC内部测试功能，fusb时钟反相
5-4	USB_CKSEL	rw	0	-	fusb时钟源选择，见图10
3	UART_CKPOL	rw	0	0	IC内部测试功能，uart 时钟反相
2-0	UART_CKSEL	rw	0	-	uart 时钟源选择，见图10

7.6.7 JL_LSBCK->PRP_CON1： peripheral clock config register 1

Bit	Name	RW	Default	RV	Description
31-22	RESERVED	-	0	-	预留
21-16	ALNK0_DIV	rw	0	-	alnk0_clk分频选择，见图8
15-12	ALNK0_SEL	rw	0	-	alnk0_clk来源选择，见图8
11-10	RESERVED	-	0	-	预留
9	APA_TEST	rw	0	0	IC内部测试功能，应保持为0
8-6	APA_CKSEL	rw	0	-	apa_clk来源选择，见图8
5	ASS_REG_RSTN	rw	0	-	ass复位
4-2	RESERVED	-	0	-	预留
1-0	ASS_CKSEL	rw	0	-	ass_clk来源选择，见图8

7.6.8 JL_HSBCK->SYS_CON0： hsbck system config register 0

Bit	Name	RW	Default	RV	Description
31-11	RESERVED	-	-	-	预留
19-16	SFC_PLL_DIV	rw	0	-	sfc_pll_mo的时钟分频

Bit	Name	RW	Default	RV	Description
15-12	SFC_PLL_SEL	rw	0	-	sfc_pll_mo的来源选择： 0: 空 1: sys_pll_d1p0 2: sys_pll_d1p5 3: sys_pll_d2p5 4: sys_pll_d3p5
11	RESERVED	-	-	-	预留
10	LSB_SFR_CKMD	rw	0	-	lsb_sfr_clk时钟模式选择： 0：有SFR写的时候，lsb_sfr_clk才使能 1：无论有无SFR写，lsb_sfr_clk都使能
9	LSB_CKEN	rw	1	-	LSB时钟使能，默认开
8	AXI_EN	rw	1	-	axi_clk使能： 0：不使能 1：使能
7-1	RESERVED	-	0	-	预留
0	HSB_SFR_CKMD	rw	0	-	hsb_sfr_clk时钟模式选择： 0：有SFR写的时候，hsb_sfr_clk才使能 1：无论有无SFR写，hsb_sfr_clk都使能

7.6.9 HSB_DIV：hsb clock div register

Bit	Name	RW	Default	RV	Description
31-20	RESERVED	-	0	-	预留
7-0	HSB_DIV	rw	0	-	HSB时钟分频，见图5

7.6.10 HSB_SEL：hsb clock sel register

Bit	Name	RW	Default	RV	Description
31-20	RESERVED	-	0	-	预留
2-0	HSB_SEL	rw	0	-	HSB时钟选择，见图5

7.6.11 SYSPLL_CON0：pll control register 0

Bit	Name	RW	Default	RV	Description
31-30	RESERVED	-	-	-	预留
29	PLL_VCTRL_OE	rw	0	-	VBG TRIM使能位

Bit	Name	RW	Default	RV	Description
28	PLL_TEST_EN	rw	0	-	PLL模拟控制位
27	RESERVED	-	-	-	预留
26-25	PLL_TEST_S	rw	0	-	PLL测试使能
24	PLL_LDO_BP	rw	0	-	PLL测试功能选择
23-20	RESERVED	-	-	-	预留
19-16	PLL_LDO11A_S	rw	0	-	PLL模拟控制位
15-13	PLL_IVCOS	rw	0	-	PLL模拟控制位
12-10	PLL_LPFR2	rw	0	-	PLL模拟控制位
9	RESERVED	-	-	-	预留
8-7	PLL_ICPS	rw	0	-	PLL模拟控制位
5-4	PLL_PFDS	rw	0	-	PLL模拟控制位
2	PLL_MODE	rw	0	-	PLL工作模式 0：整数模式 1：小数模式
1	PLL_RST	rw	0	1	PLL模块复位，需在PLL EN使能10uS之后才能释放 0: 复位 1: 释放
0	PLL_EN	rw	0	1	PLL模块使能 0: 关闭 1: 打开

7.6.12 SYSPLL_CON1：pll control register 1

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	-	-	预留
15-14	PLL_REFDSEN	rw	0	-	PLL参考时钟分频使能（见图1） 00: PLL内参考时钟分频器X2； 01: PLL内参考时钟分频器关闭（DIV1）； 1*: PLL内参考时钟分频器打开（DIV2-129）

Bit	Name	RW	Default	RV	Description
12	PLL_REFSEL	rw	0	-	0:PLL_REFCLKP0_20v 1:PLL_REFCLK_08v
11	RESERVED	-	-	-	预留
10-8	PLL_REF_SEL	rw	0	-	PLL模块参考时钟选择（见图1）
6-0	PLL_REFDS	rw	0	-	PLL参考时钟分频值，n= PLL_REFDS + 2 （见图1）

7.6.13 SYSPLL_NR： pll config register nr

Bit	Name	RW	Default	RV	Description
31-12	RESERVED	-	-	-	预留
11-0	PLL_NR	rw	0	-	PLL参考时钟反馈分频(相当倍频)， m= PLL_NR + 2。见图1 **注意：读出值为写入值+2**

7.6.14 SYSPLL_CON2： pll control register 2

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	-	-	预留
16	PLL_CKEN	rw	0	0	输出时钟使能
15-7	RESERVED	-	-	-	预留
6	PLL_CKSYN_CORE_OE	rw	0	0	同步时钟使能
5	RESERVED	-	-	-	预留
4	PLL_CKOUT_D3P5_OE	rw	0	-	3.5分频时钟输出使能
3	PLL_CKOUT_D2P5_OE	rw	0	-	2.5分频时钟输出使能
2	PLL_CKOUT_D2P0_OE	rw	0	-	2分频时钟输出使能
1	PLL_CKOUT_D1P5_OE	rw	0	-	1.5分频时钟输出使能
0	PLL_CKOUT_D1P0_OE	rw	0	-	1分频时钟输出使能

8 ADC

8.1 概述

10Bit ADC(A/D转换器)，其时钟最大不可超过500KHz。

8.2 寄存器说明

8.2.1 ADC_CON: ADC control register

Bit	Name	RW	Default	RV	Description
31	DONE_PND	r	0	-	只读，采样完成pnd；kst后完成一次该位置1
30	DONE_PND_CLR	w	-	-	只写，写‘1’清除DONE_PND位，写‘0’无效
29	DONE_PND_IE	rw	0	-	DONE_PND中断允许
28-24	RESERVED	-	-	-	预留
23-21	ADC_MUX_SEL	rw	0x0	0	ADC模式选择： 001：ADC采集IO通道电压 010：ADC采集内部模拟通道电压 111：将选通的内部模拟电压通过IO口输出
20-18	ADC_ASEL	rw	0x0	-	内部模拟通道选择： 000: 无 001: AUDIO 010: PMU 011: LRC200K 100: 无 101:SYSPLL 110:WAT 111:CLASSD
17	ADC_CLKEN	rw	0	0	ADC 时钟使能
16	ADCISEL	rw	0		ADC CMP power select
15-12	WAIT_TIME	rw	x	-	启动延时控制，实际启动延时为此数值乘8个ADC时钟

Bit	Name	RW	Default	RV	Description
11-8	CH_SEL	rw	x	-	IO通道选择： 0000：选择PA0 0001：选择PA1 0010：选择PA2 0011：选择PA3 0100：选择DP 0101：选择DM 0110：选择PA4 0111：选择PA5 1000：选择PA6 1001：选择PA10 1010：选择PA11 1011：选择PA12 1100：选择PA13 1101：选择PA14 1110：选择PA15 1111：选择PB0
7	PND	r	1	-	PND：只读，中断请求位，当ADC完成一次转换后，此位会被设置为‘1’，需由软件清‘0’
6	CPND	w	x	-	CPND：只写，写‘1’清除中断请求位，写‘0’无效
5	ADC_IE	rw	0	-	ADC_IE：ADC中断允许
4	ADC_EN	rw	0	-	ADC_EN： ADC控制器使能
3	ADC_AE	rw	0	-	ADC_AE： ADC 模拟模块常使能
2-0	ADC_BAUD	rw	0x0	-	ADC_BAUD： ADC时钟频率选择 000：LSB时钟1分频 001：LSB时钟6分频 010：LSB时钟12分频 011：LSB时钟24分频 100：LSB时钟48分频 101：LSB时钟72分频 110：LSB时钟96分频 111：LSB时钟128分频

8.2.2 ADC_RES：ADC result register

Bit	Name	RW	Default	RV	Description
31-10	RESERVED	-	-	-	预留
9-0	RES	r	x	-	ADC结果

9 CRC

9.1 概述

CRC16计算器，每次运算8bit，多项式为：$X^{16}+X^{12}+X^{5}+X^{1}$。

CRC模块的原理框图如下所示：

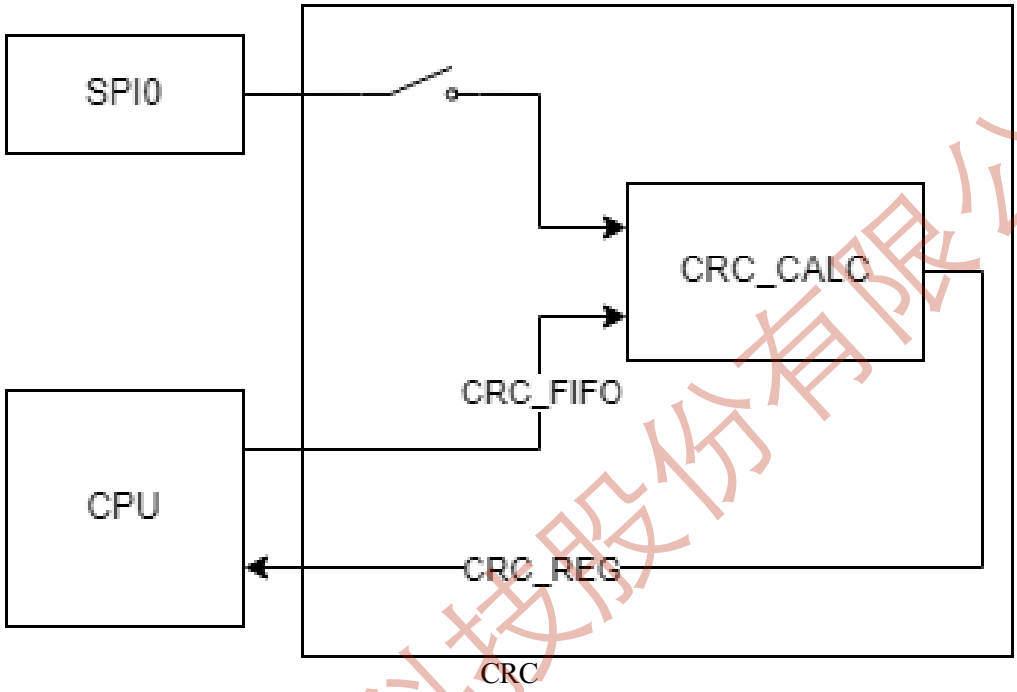


图1、CRC原理框图

当SPI0在DMA状态下计算CRC时，应避免使用CPU操作CRC_FIFO

注意：该项目CRC模块被集成到SPI0中，与SPI0共用基地址

9.2 寄存器说明

9.2.1 REG: CRC16 校验码

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	-	-	预留
15-0	CRC_REG	rw	x	-	写入初始值，CRC计算完毕，读取校验码

9.2.2 FIFO: 运算数据输入

Bit	Name	RW	Default	RV	Description
31-8	RESERVED	-	-	-	预留

Bit	Name	RW	Default	RV	Description
7-0	CRC_FIFO	w	x	-	运算数据输入，HSB先运算，LSB后运算

10 GPCNT

10.1 概述

GPCNT为时钟脉冲计数器，用于计算两个时钟周期的比例，即用一个已知时钟（主时钟）计算另一个时钟（次时钟）的周期。

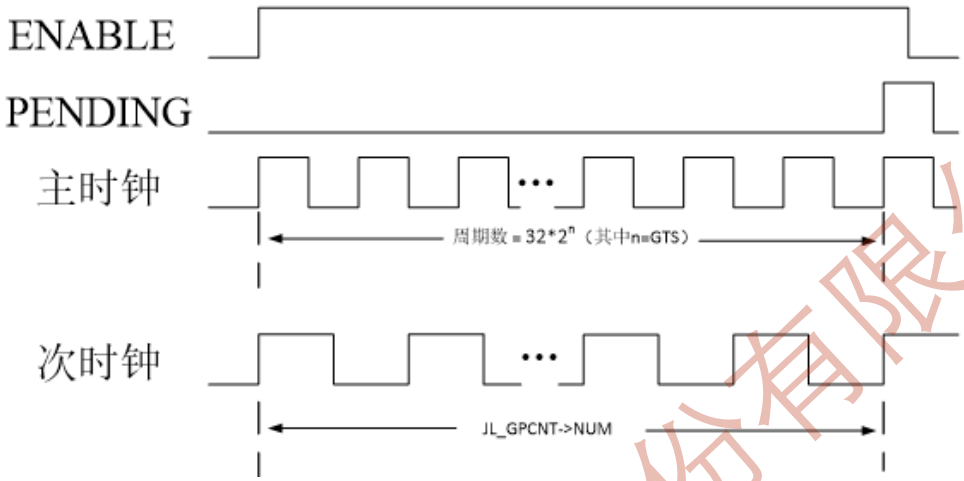


图 1 GPCNT时钟计算示意图

10.2 寄存器说明

10.2.1 JL_GPCNT->CON: GPCNT configuration register

Bit	Name	RW	Default	RV	Description
31	PND	r	0	-	中断请求标志（当JL_GPCNT->CON[0]置1后，主时钟达到JL_GPCNT->CON[11:8]设定的周期数后，PENDING置1，并请求中断）： 0: 无PENDING; 1: 有PENDING
30	CLR_PND	w	x	-	清除中断请求标志位： 0: 无效； 1: 清PENDING
29-20	RESERVED	-	-	-	预留
19-16	GTS	rw	0x0	-	主时钟周期数选择： 主时钟周期数 = 32*2^n （其中n=GTS）
15-13	RESERVED	-	-	-	预留
12-8	GSS	rw	0x0	-	主时钟选择（计数时钟）：参见Clock_System 文档
7-6	RESERVED	-	-	-	预留

Bit	Name	RW	Default	RV	Description
5-1	CSS	rw	0x0	-	次时钟选择（被计数时钟）：参见Clock_System 文档
0	ENABLE	rw	0	-	GPCNT模块使能位： 0: 不使能； 1: 使能

【注意】：需配置好其他位，才将ENABLE置1。

10.2.2 JL_GPCNT->NUM: The number of GPCNT clock cycle register

Bit	Name	RW	Default	RV	Description
31-0	NUM	r	0x0	-	在JL_GPCNT->CON[0]置1到 PENDING置1（中断到来）之间，次时钟跑的周期数。

11 IIC

11.1 概述

IIC接口是一个可兼容大部分IIC总线标准的串行通讯接口。在上面传输的数据以Byte（8bit）为最小单位，且永远是MSB在前。

1. IIC接口主要支持特性： (1). 主模式和从模式，不支持多机功能； (2). 支持标准速度模式（Standard-mode，Sm，高达100kHz），快速模式（Fast-mode，Fm，高达400kHz），快速增强模式（Fast-mode Plus，Fm+，高达1MHz）； (3). 7bit主从机寻址模式，可配置从机地址应答，广播地址响应； (4). 总线时钟延展功能可配置； (5). 总线数据建立时间和保持时间可配置； (6). 基于便捷的事务操作驱动总线； (7). 数据接收发送各独立具有1字节缓冲； (8). 可配置总线信号数字滤波器；

2. 主机： (1). IIC接口SCL时钟由本机产生，提供给片外IIC设备使用； (2). IIC接口SCL时钟可配置，SCL时钟周期为：

```
TIIC_BUS_SCL = TIIC_clk /(2 * BUAD_CNT) + T电阻上拉
```

3. 从机： (1). IIC接口SCL时钟由片外 IIC设备产生，经内部IIC_clk滤波采样后给本机使用； (2). IIC总线上每一个start/restart位后接从机地址，此时当外部IIC设备所发的地址与我们匹配时（开启广播地址响应，在接收到广播地址时也会响应），可配置硬件自动回应（ack位为“0”）；

11.2 特殊注意点

时钟要求：作为主机时，BAUD、TSU 、THD、FLT_SEL值设置应当满足以下关系

$$BAUD\_CNT > (SETUP\_CNT + HOLD\_CNT + FLT\_SEL+5)$$

作为从机时，SCL 时钟低电平时间与TSU、THD、FLT_SEL值设置应当满足以下关系

$$TSCL\_L > (SETUP\_CNT + HOLD\_CNT + FLT\_SEL + 5) * TI2C\_clk$$

IO设置：在选择IO作为IIC接口的SCL、SDA输入输出时，必须配置该两个IO的SPL寄存器位为1；

事务寄存器：写入事务后需等待硬件将写入的事务加载后（tx_task_load/rx_task_load 置位即表示对应事务被硬件加载，task_done pnd置位即表示事务已被硬件加载且执行完成）才可写入新的事务；

【注意】：IIC的通信频率不仅与所配置的波特率有关，还会受到IIC总线上拉时间的影响，请尽可能符合IIC总线硬件电气要求。

11.3 数字模块控制寄存器

11.3.1 IIC_CON0: iic control register 0

Bit	Name	RW	Default	RV	Description
31-10	RESERVED	-	-	-	预留
10	BADDR_RESP_EN	rw	0	-	广播地址响应使能： 1：响应总线上的广播地址； 0：不响应总线上的广播地址

Bit	Name	RW	Default	RV	Description
9	AUTO_SLV_TX	rw	0	-	从机自动发送数据事务使能（主机模式下无效）： 0：不使能； 1：使能 从机接收匹配的地址且为从机接收数据时，接下来硬件会自动填充SEND_DATA事务操作，使得硬件能不间断的发送数据； （作为从机与不支持时钟延展的主机通信时必须使能，其他工作场景依据需求使能）
8	AUTO_SLV_RX	rw	0	-	从机自动接收数据事务使能（主机模式下无效）： 0：不使能； 1：使能 从机接收匹配的地址且为从机接收数据时，接下来硬件会自动填充RECV_DATA事务操作，使得在软件读取数据时，硬件接口能继续不间断接收数据； （作为从机与不支持时钟延展的主机通信时必须使能，其他工作场景依据需求使能）
7	AUTO_ADR_RESP	rw	0	-	从机接收地址硬件自动响应使能： 作为从机时，在接收到地址数据后（接收的第一字节数据） 0：不自动进行响应，即不对接收到的地址数据进行NACK/ACK，需cpu读取接收数据后，再执行NACK/ACK事务； 1：自动进行响应，接收到匹配的地址（含广播地址） 自动响应ACK；接收到不匹配的地址，自动响应NACK （从机与不支持时钟延展的主机通信时必须使能，其他工作场景依据需求使能）
6	IGNORE_NACK	rw	0	-	NACK调试模式使能 0：在发送数据中遇到NACK后会停止接收发送； 1：在发送数据中遇到NACK后不会中断接收发送事务（仅用于调试）；
5	NO_STRETCH	rw	0	-	IIC主从时钟延展功能选择 0：支持时钟延展； 1：不支持时钟延展
4-3	FLT_SEL	rw	0	-	IIC接口数字滤波器选择， 无特殊情况软件配置值为2’b10 2’b11：滤除3 * Tiic_baud_clk 以下尖峰脉宽； 2’b10：滤除2 * Tiic_baud_clk 以下尖峰脉宽； 2’b01：滤除Tiic_baud_clk 以下尖峰脉宽； 2’b00：关闭数字滤波器
2	SLAVE MODE	rw	0	-	IIC接口主从机模式： 0：主机模式； 1：从机模式
1	I2C_RST	rw	0	-	IIC接口复位（使能后再释放复位） 0：IIC接口复位； 1：IIC接口释放

Bit	Name	RW	Default	RV	Description
0	EN	rw	0	-	IIC接口使能。 0：关闭IIC接口 1：打开IIC接口

11.3.2 IIC_TASK: iic task register

Bit	Name	RW	Default	RV	Description
31-4	RESERVED	-	-	-	预留
12	SLAVE_CALL	r	-	-	从机地址匹配（仅debug，用户正常流程不可使用）
11	BC_CALL	r	-	-	广播地址匹配（仅debug，用户正常流程不可使用）
10	SLAVE_RW	r	-	-	从机读写状态（仅debug，用户正常流程不可使用）
9-6	RUNNING_TASK	r	-	-	正在运行事务（仅debug，用户正常流程不可使用）
5	TASK_RUN_RDY	r	-	-	事务运行状态（仅debug，用户正常流程不可使用）
4	TASK_LOAD_RDY	r	-	-	事务加载状态（仅debug，用户正常流程不可使用）
3-0	RUN_TASK	w	0x0	-	事务操作：以下9种事务操作涵盖了主从机接收发送时的总线行为序列（具体解析见下节） 0x0：SEND_RESET 0x1：SEND_ADDR 0x2：SEND_DATA 0x3：SEND_ACK 0x4：SEND_NACK 0x5：SEND_STOP 0x6：SEND_NACK_STOP 0x7：RECV_DATA 0x8：RECV_DATA_WITH_ACK 0x9：RECV_DATA_WITH_NACK

11.3.3 IIC_PND: iic pending register

Bit	Name	RW	Default	RV	Description
31-29	RESERVED	-	-	-	预留
28	RXTASK_LOAD_PND	r	0x0	-	硬件将当前的接收相关的task事务加载完成
27	TXTASK_LOAD_PND	r	0x0	-	硬件将当前的发送相关的task事务加载完成
26	RXBYTE_DONE_PND	r	0x0	-	接收到1byte的数据

Bit	Name	RW	Default	RV	Description
25	ADR_MATCH_PND	r	0x0	-	作为从机接收到与ADDR相符的地址 （开始广播地址响应使能后， 匹配广播地址也会起该PND）
24	RXNACK_PND	r	0x0	-	发送地址/数据后接收到NACK
23	RXACK_PND	r	0x0	-	发送地址/数据后接收到ACK
22	STOP_PND	r	0x0	-	接收到总线STOP信号序列
21	RESTART_PND	r	0x0	-	接收到总线RESTART信号序列
20	TASK_PND	r	0x0	-	事务执行完成PND
19	RESERVED	-	-	-	预留
18	RXTASK_LOAD_CLR	w	-	-	写“1”清除对应的PND
17	TXTASK_LOAD_CLR	w	-	-	写“1”清除对应的PND
16	RXDATA_DONE_CLR	w	-	-	写“1”清除对应的PND
15	ADR_MATCH_CLR	w	-	-	写“1”清除对应的PND
14	RXNACK_CLR	w	-	-	写“1”清除对应的PND
13	RXACK_CLR	w	-	-	写“1”清除对应的PND
12	STOP_CLR	w	-	-	写“1”清除对应的PND
11	RESTART_CLR	w	-	-	写“1”清除对应的PND
10	TASK_CLR	w	-	-	写“1”清除对应的PND
9	RESERVED	-	-	-	预留
8	RXTASK_LOAD_IE	rw	0		对应PND的中断使能
7	TXTASK_LOAD_IE	rw	0		对应PND的中断使能
6	RXDATA_DONE_IE	rw	0		对应PND的中断使能
5	ADR_MATCH_IE	rw	0		对应PND的中断使能
4	RXNACK_IE	rw	0		对应PND的中断使能
3	RXACK_IE	rw	0	-	对应PND的中断使能
2	STOP_IE	rw	0	-	对应PND的中断使能
1	RESTART_IE	rw	0	-	对应PND的中断使能
0	TASK_IE	rw	0	-	对应PND的中断使能

11.3.4 IIC_TXBUF: iic tx buff register

Bit	Name	RW	Default	RV	Description
31-8	RESERVED	-	-	-	-
7-0	TX_BUFF	rw	-	-	待发送数据

11.3.5 IIC_RXBUF: iic tx buff register

Bit	Name	RW	Default	RV	Description
31-8	RESERVED	-	-	-	-
7-0	RX_BUFF	r	-	-	已接收数据

11.3.6 IIC_ADDR: iic device address register

Bit	Name	RW	Default	RV	Description
31-8	RESERVED	-	-	-	-
7-1	IIC DEVICE ADDR	rw	0x00	-	IIC 设备地址（仅支持7bit模式）
0	W/R OPTION	rw	0x0	-	作为主机发送地址时的读写标志： 0：发送数据到从机； 1：读取从机的数据；

11.3.7 IIC_BAUD: iic baud clk register

Bit	Name	RW	Default	RV	Description
31-12	RESERVED	-	-	-	-
11-0	BAUD_CNT	rw	-	-	IIC 总线传输时钟SCL速率配置： FIIC_BUS_SCL = FIIC_clk / BUAD_CNT **注意：不可设置为0**

11.3.8 IIC_TSU: iic setup time register

Bit	Name	RW	Default	RV	Description
31-7	RESERVED	-	-	-	-

Bit	Name	RW	Default	RV	Description
5-0	SETUP_CNT	rw	-	-	IIC 总线SDA信号更新后到SCL时钟上升的最少时间配置： Tsetup = TIIC_clk * SETUP_CNT 无特殊情况软件配置值为2；

11.3.9 IIC_THD: iic hold time register

Bit	Name	RW	Default	RV	Description
31-7	RESERVED	-	-	-	-
5-0	HOLD_CNT	rw	-	-	IIC 总线SCL时钟下降到SDA信号更新前的最少时间配置： Thold = TIIC_clk * HOLD_CNT 无特殊情况软件配置值为2；

11.3.10 IIC_DBG: iic debug register

Bit	Name	RW	Default	RV	Description
31-16	SDA_DBG	r	-	-	SDA相关调试信号
15-0	SCL_DBG	r	-	-	SCL相关调试信号

11.4 基本事务操作

IIC_TASK 事务寄存器允许写入以下9种事务操作值来驱动IIC接口做出相应总线行为序列：

1. 0x0：SEND_RESET

   工作在主机模式下，向IIC总线发送START 和 STOP；

2. 0x1：SEND_ADDR

   工作在主机模式下，向IIC总线发送START 和 ADDR的数据，并接收从机回复的ACK/NACK；

3. 0x2：SEND_DATA

   工作在主机模式下，向IIC总线发送TX BUFF的数据，并接收从机回复的ACK/NACK；

4. 0x3：SEND_ACK

   工作在主/从机模式下，向IIC总线发送ACK；

5. 0x4：SEND_NACK

   工作在主/从机模式下，向IIC总线发送NACK；

6. 0x5：SEND_STOP

   工作在主机模式下，向IIC总线发送STOP；

7. 0x6：SEND_NACK_STOP

   工作在主机模式下，向IIC总线发送NACK和STOP；

8. 0x7：RECV_DATA

   工作在主机/从机模式下，从IIC总线上接收一个字节数据，但不进行ACK/NACK（持续拉低SCL线，总线上的设备需支持时钟延展），直到SEND_ACK/SEND_NACK的事务被填充，SCL才会撤销拉低；

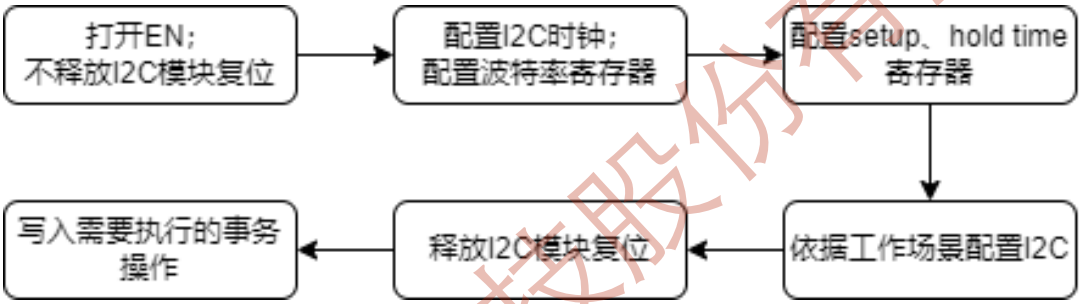
9. 0x8：RECV_DATA_WITH_ACK

   工作在主机/从机模式下，从IIC总线上接收一个字节数据，且随后向IIC总线发送ACK；

10. 0x9：RECV_DATA_WITH_NACK

    工作在主机/从机模式下，从IIC总线上接收一个字节数据，且随后向IIC总线发送NACK；

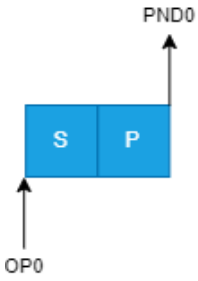
11.5 基本工作场景使用流程



1. 主机发送

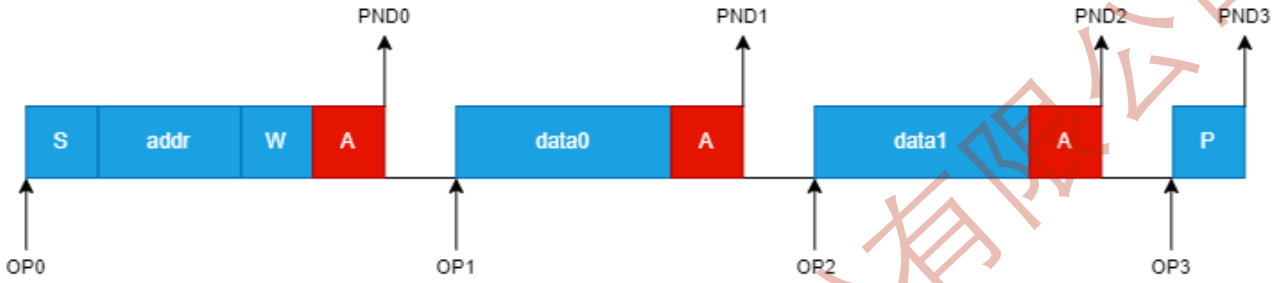
   主机发送具有字节断续和字节连续发送两种工作场景。

   字节连续发送：用于传输数据量较大的（中途无需进行数据处理）场景，主机在发送地址并且匹配后通过txtask_pnd来预先（提前约8个I2C_SCL时钟周期）填充下一事务达到连续发送。



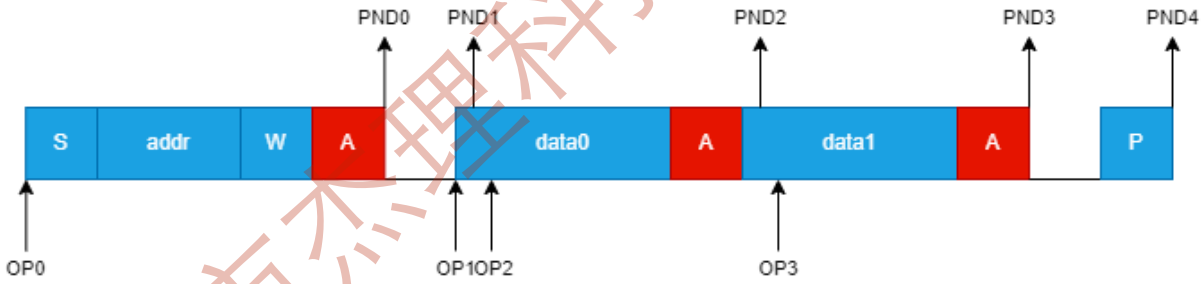
OP0 : TASK = SEND_RESET

PND0 : "start" & "stop" send done



OP0 : ADDR = addr + w
TASK = SEND_ADDR
OP1 : TX_BUF = data0
TASK = SEND_DATA
OP2 : TX_BUF = data1
TASK = SEND_DATA
OP3 : TASK = SEND_STOP

PND0：task_done(发送 addr，接收 ack)
PND1：task_done(发送 data0，接收 ack)
PND2：task_done(发送 data1，接收 ack)
PND3：task_done(发送 stop)

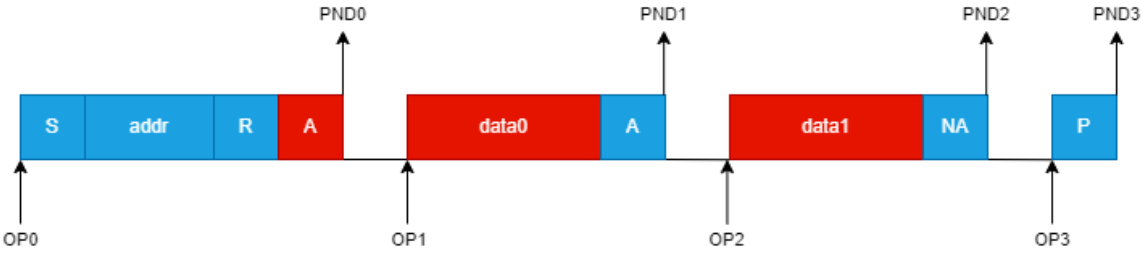


OP0 : ADDR = addr + w
TASK = SEND_ADDR
OP1 : TX_BUF = data0
TASK = SEND_DATA
OP2 : TX_BUF = data1
TASK = SEND_DATA
OP3 : TASK = SEND_STOP

PND0：task_done(发送 addr，接收 ack)
PND1：txtask_load(SEND_DATA task loaded)
PND2：txtask_load(SEND_DATA task loaded)
PND3：task_done(send data1 done, 接收到ack / nack)
PND3：task_done(send stop done)

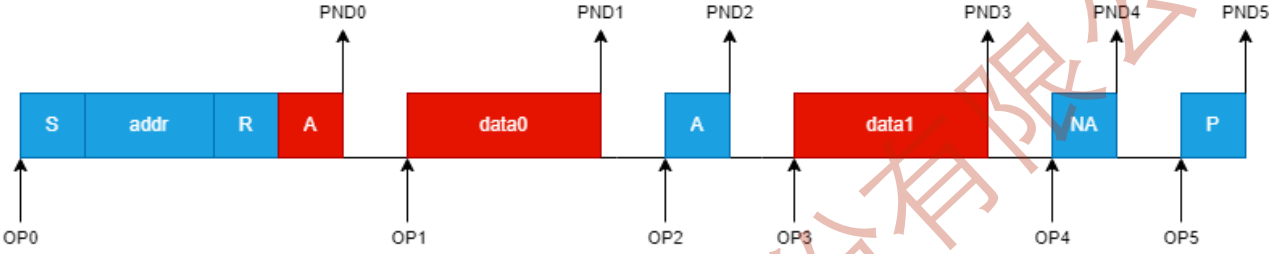
2. 主机接收

与主机发送相似，主机发送同样具有字节断续和字节连续发送两种工作场景；



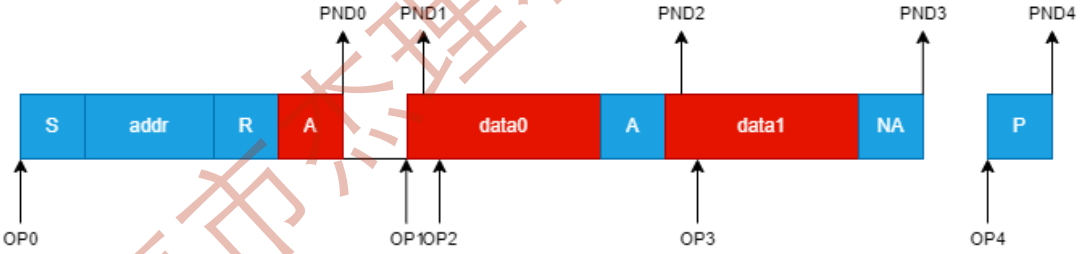
OP0 : ADDR = addr + r
TASK = SEND_ADDR
OP1 : TASK = RECV_DATA_WITH_ACK
OP2 : buf[0] = RX_BUF
TASK = RECV_DATA_WITH_NAK
OP3 : buf[1] = RX_BUF
TASK = SEND_STOP

PND0：task_done(发送 addr，接收 ack)
PND1：task_done(发送 ack，接收 data0)
PND2：task_done(发送 nack，接收 data1)
PND3：task_done(发送 stop)



OP0 : ADDR = addr + r
TASK = SEND_ADDR
OP1 : TASK = RECV_DATA
OP2 : buf[0] = RX_BUF
TASK = SEND_ACK
OP3 : TASK = RECV_DATA
OP4 : buf[1] = RX_BUF
TASK = SEND_NACK
OP5 : TASK = SEND_STOP

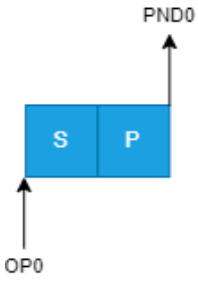
PND0：task_done(发送 addr，接收 ack)
PND1：task_done(接收 data0)
PND2：task_done(发送 ack)
PND3：task_done(接收 data1)
PND4：task_done(发送 nack)
PND5：task_done(发送 stop)



OP0 : ADDR = addr + r
TASK = SEND_ADDR
OP1 : TASK = RECV_DATA_WITH_ACK
OP2 :
TASK = RECV_DATA_WITH_NACK
OP3 : buf[0] = RX_BUF
TASK = SEND_STOP
OP4 : buf[1] = RX_BUF

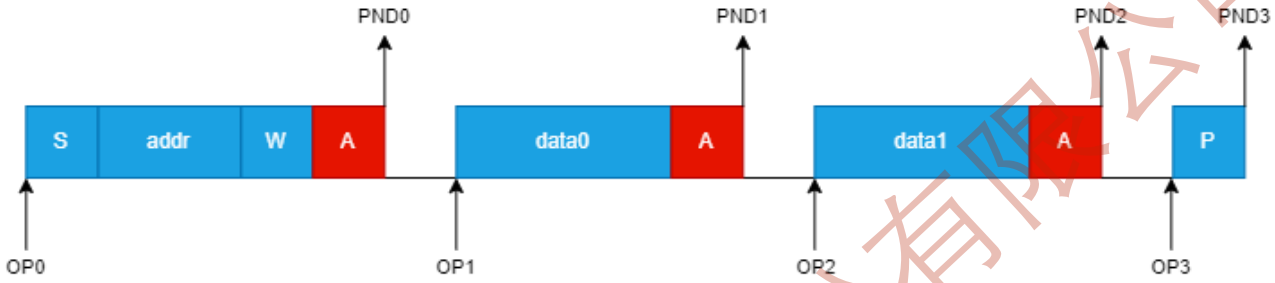
PND0：task_done(发送地址，接收ack)
PND1：rxtask_load(RECV_DATA task loaded)
PND2：rxtask_load(RECV_DATA task loaded)
PND3：task_done(recv data1 done, 发送NACK/ACK)
PND4：task_done(发送 stop)

3. 主机其他操作（用于复位总线上其他从设备）



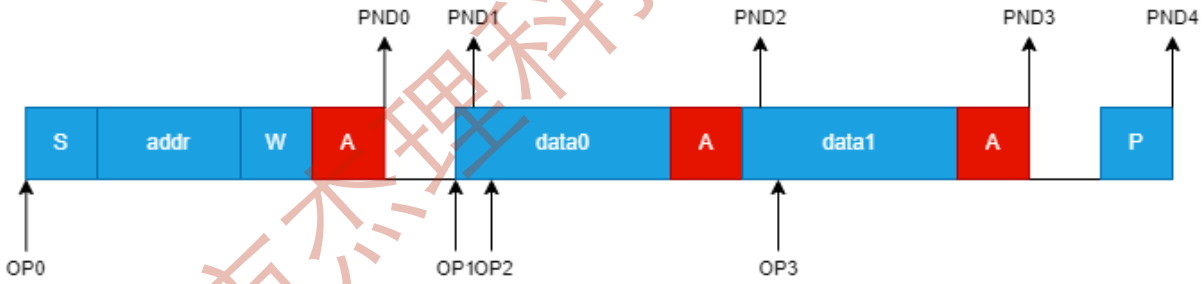
OP0 : TASK = SEND_RESET

PND0 : "start" & "stop" send done



OP0 : ADDR = addr + w
TASK = SEND_ADDR
OP1 : TX_BUF = data0
TASK = SEND_DATA
OP2 : TX_BUF = data1
TASK = SEND_DATA
OP3 : TASK = SEND_STOP

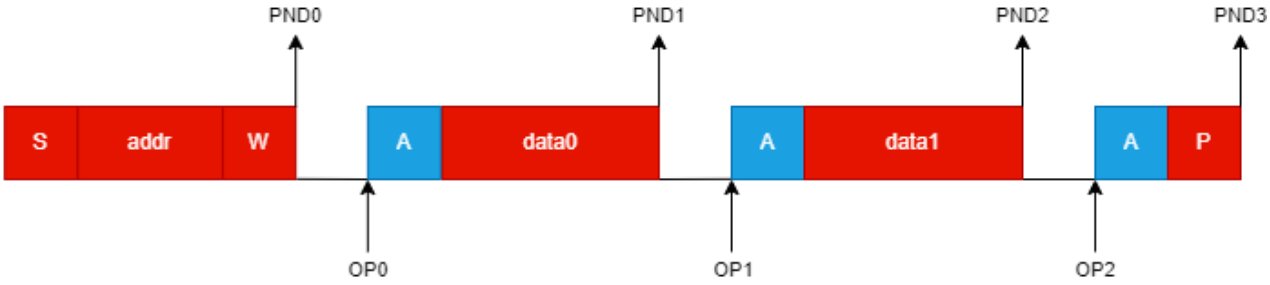
PND0：task_done(发送 addr，接收 ack)
PND1：task_done(发送 data0，接收 ack)
PND2：task_done(发送 data1，接收 ack)
PND3：task_done(发送 stop)



OP0 : ADDR = addr + w
TASK = SEND_ADDR
OP1 : TX_BUF = data0
TASK = SEND_DATA
OP2 : TX_BUF = data1
TASK = SEND_DATA
OP3 : TASK = SEND_STOP

PND0：task_done(发送 addr，接收 ack)
PND1：txtask_load(SEND_DATA task loaded)
PND2：txtask_load(SEND_DATA task loaded)
PND3：task_done(send data1 done, 接收到ack / nack)
PND3：task_done(send stop done)

4. 从机接收



STRETCH_EN = 1
AUTO_ADR_RESP = 0

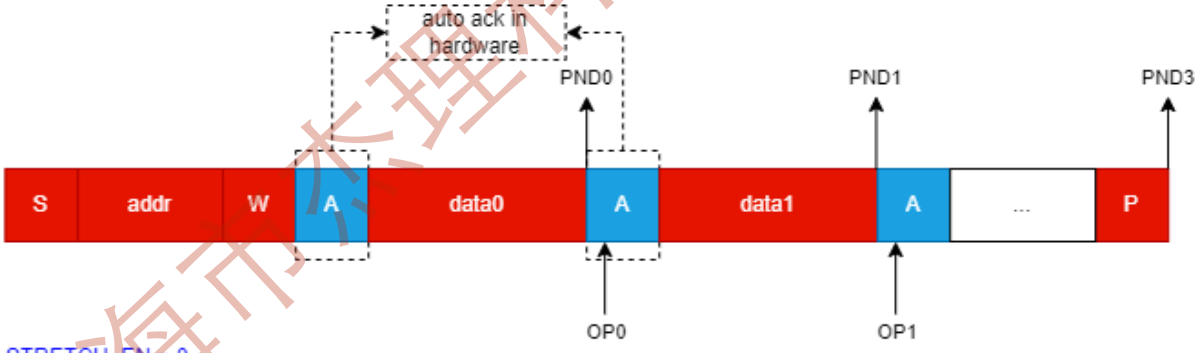
OP0 : TASK = RECV_DATA_WITH_ACK
OP1 : TASK = RECV_DATA_WITH_ACK
OP2 : TASK = RECV_DATA_WITH_ACK
/ SEND_NACK

PND0 : task_done("addr" match)
PND1 : task_done("data0" recieve done)
PND2 : task_done("data1" recieve done)
PND3 : task_done("stop" recieve)

STRETCH_EN = 1
AUTO_ADR_RESP = 1

OP0 : TASK = RECV_DATA_WITH_ACK
OP1 : TASK = RECV_DATA_WITH_ACK
/ RECV_DATA_WITH_NACK

PND0 : task_done("addr" match)
PND1 : task_done("data0" recieve done)
PND2 : task_done("data1" recieve done)
PND3 : task_done("stop" recieve)

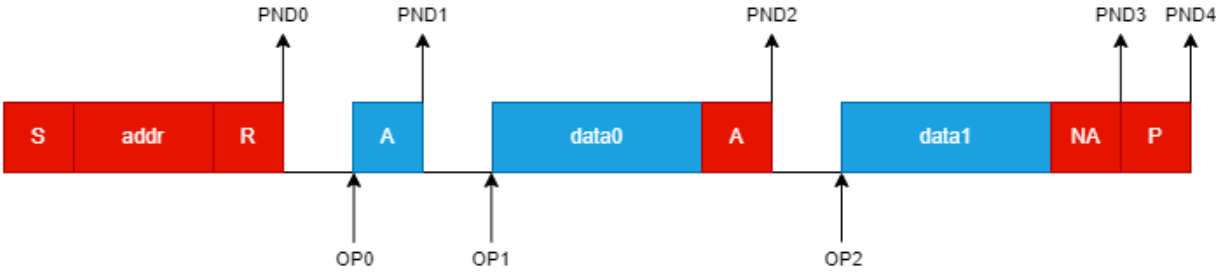


STRETCH_EN = 0
AUTO_ADR_RESP = 1
AUTO_SLV_RX = 1

OP0 : TASK = SEND_ACK(for data1)
OP1 : TASK = SEND_ACK(for data2)
...

PND0 : rxbyte_done("addr" match, "data0" recieve)
PND1 : rxdata_done("data1" recieve)
...
PND3 : stop("stop" recieve)

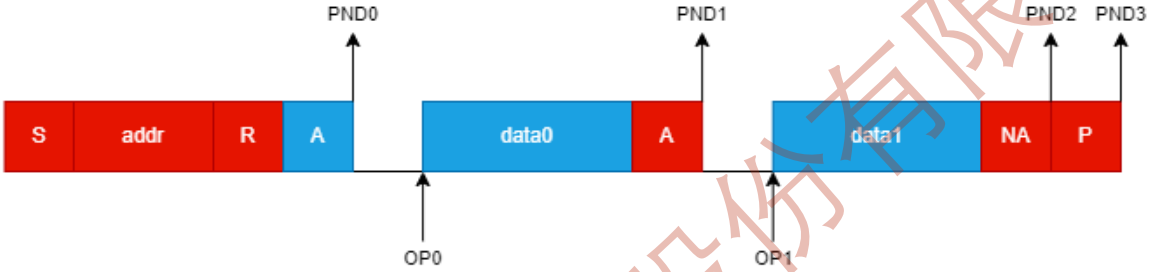
5. 从机发送



STRETCH_EN = 1
AUTO_ADR_RESP = 0

OP0 : TASK = SEND_ACK
OP1 : TX_BUF = data0
TASK = SEND_DATA
OP2 : TX_BUF = data1
TASK = SEND_DATA

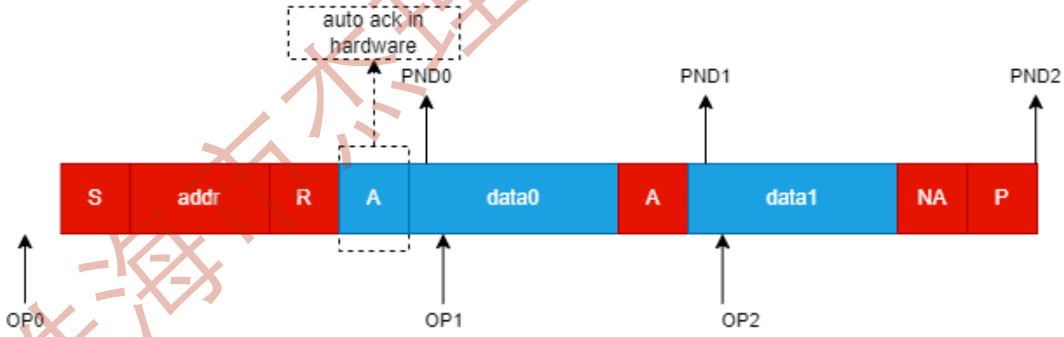
PND0 : task_done("addr" match)
PND1 : task_done("ack" send done)
PND2 : task_done("data0" send done , "ack" receive)
PND3 : task_done("data1" send done , "nack" receive)
PND4 : task_done("stop" recieve)



STRETCH_EN = 1
AUTO_ADR_RESP = 1

OP0 : TX_BUF = data0
TASK = SEND_DATA
OP1 : TX_BUF = data1
TASK = SEND_DATA

PND0 : task_done("addr" match)
PND1 : task_done("data0" send done , "ack" receive)
PND2 : task_done("data1" send done , "nack" receive)
PND3 : task_done("stop" recieve)



STRETCH_EN = 0
AUTO_ADR_RESP = 1
AUTO_SLV_TX = 1

OP0 : TX_BUF = data0
OP1 : TX_BUF = data1
OP2 : TX_BUF = data2

PND0 :tx load done("addr" match ,"data0" tx load done)
PND1 :tx load done("data1" tx load done , "ack" receive)
PND2 :stop ("nack" receive, "stop" receive)

12 GPCRC

12.1 概述

GPCRC（General Purpose Cyclic Redundancy Calculation Unit，即通用循环冗余检验计算单元）使用多项式计算一个8位/16位/32位的数据字产生的CRC码。

支持多项式位数：7位、8位、16位、32位。

支持多项式系数编程，或使用默认多项式（CRC-32/MPEG-2）。

支持输入数据位反转、输出数据位反转、输入数据端序翻转。

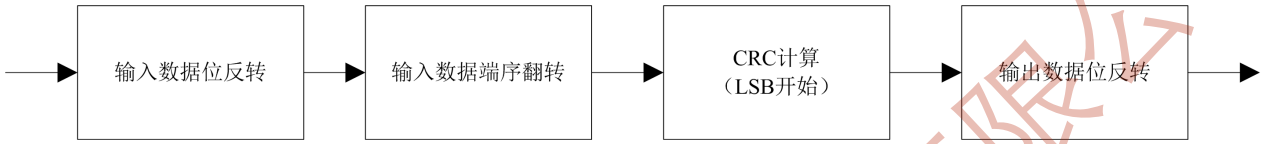


图1-1 GPCRC原理图

12.2 功能描述

12.2.1 数据处理

输入数据端序翻转：

指大小端变换，如数据为{Byte3, Byte2, Byte1, Byte0}，半字翻转为{Byte2, Byte3, Byte0, Byte1}，字翻转为{Byte0, Byte1, Byte 2, Byte3}。

输入数据反转：

指字节按位反转，如数据为{0x01,0x02,0x03,0x04}，反转为{0x80,0x40,0xc0,0x20}。

输出数据反转：

指对应7位、8位、16位、32位多项式按位反转。

12.2.2 多项式配置

多项式名称	多项式公式	REV_IN	REV_OUT	POL	INIT
CRC-7/MMC	x7+x3+1	不反转	不反转	0x00000009	0x00000000
CRC-8	x8+x2+x1+1	不反转	不反转	0x00000007	0x00000000
CRC-8/ITU	x8+x2+x1+1	不反转	不反转	0x00000007	0x00000000
CRC-8/ROHC	x8+x2+x1+1	反转	反转	0x00000007	0x000000ff
CRC-8/MAXIM	x8+x5+x4+1	反转	反转	0x00000031	0x00000000
CRC-16/IBM	x16+x15+x2+1	反转	反转	0x00008005	0x00000000

多项式名称	多项式公式	REV_IN	REV_OUT	POL	INIT
CRC-16/MAXIM	x16+x15+x2+1	反转	反转	0x00008005	0x00000000
CRC-16/USB	x16+x15+x2+1	反转	反转	0x00008005	0x0000ffff
CRC-16/MODBUS	x16+x15+x2+1	反转	反转	0x00008005	0x0000ffff
CRC-16/CCITT	x16+x12+x5+1	反转	反转	0x00001021	0x00000000
CRC-16/CCITT-FALSE	x16+x12+x5+1	不反转	不反转	0x00001021	0x0000ffff
CRC-16/X25	x16+x12+x5+1	反转	反转	0x00001021	0x0000ffff
CRC-16/XMODEM	x16+x12+x5+1	不反转	不反转	0x00001021	0x00000000
CRC-16/DNP	x16+x13+x12+x11+ x10+x8+x6+x5+ x2+1	反转	反转	0x00003d65	0x00000000
CRC-32	x32+x26+x23+x22+ x16+x12+x11+x10+ x8+x7+x5+x4+ x2+x+1	反转	反转	0x04c11db7	0xffffffff
CRC-32/MPEG-32	x32+x26+x23+x22+ x16+x12+x11+x10+ x8+x7+x5+x4+ x2+x+1	不反转	不反转	0x04c11db7	0xffffffff

12.3 数字模块控制寄存器

12.3.1 GPCRC_CON: gpcrc control register

Bit	Name	RW	Default	RV	Description
31:9	RESERVED	-	-	-	预留
8	REV_IN	rw	1	-	输入数据位反转 0：反转 1：不反转
7	REV_OUT	rw	0	-	输出数据位反转 0：不反转 1：反转

Bit	Name	RW	Default	RV	Description
6:5	SWP_IN	rw	0x0	-	输入数据端序翻转 0/3：不翻转 1：半字翻转 2：字翻转
4:3	POL_SIZE	rw	0x0	-	多项式位数 0：32位 1：16位 2：8位 3：7位
2:1	DAT_SIZE	rw	0x0	-	输入数据类型 0/3：32位 1：16位 2：8位
0	RESERVED	-	-	-	预留

12.3.2 GPCRC_POL: gpcrc poly register

Bit	Name	RW	Default	RV	Description
31:0	POL	rw	0x04c11db7	-	CRC多项式系数，第0位恒为1

12.3.3 GPCRC_INIT: gpcrc initial register

Bit	Name	RW	Default	RV	Description
31:0	INIT	w	-	-	写入CRC初始数据

12.3.4 GPCRC_REG: gpcrc data register

Bit	Name	RW	Default	RV	Description
31:0	REG	rw	0xffffffff	-	写入CRC数据或读出CRC结果

12.4 DEMO CODE

```
#define gpcrc_con_set(swp_in, rev_out, rev_in, pol_size, dat_size) \
(                                \
  (rev_in  & 0x1) << 8 | \
  (rev_out & 0x1) << 7 | \
  (swp_in  & 0x3) << 5 | \
  (pol_in  & 0x3) << 3 | \
  (dat_size & 0x3) << 1 | \
)

typedef enum {
  NO_SWP_IN, HWAD_SWP_IN, WORD_SWP_IN
```

```
} SWP_IN_CFG;

typedef enum {
  NO_REV_OUT, REV_OUT
} REV_OUT_CFG;

typedef enum {
  REV_IN, NO_REV_IN
} REV_IN_CFG;

typedef enum {
  POL32, POL16, POL8, POL7
} POL_SIZE;

typedef enum {
  U32, U16, U8
} DAT_SIZE;

u8 crc8_rohc_u8(u8 *data, u32 length)

{
  JL_GPCRC->INIT = 0x000000ff;
  JL_GPCRC->CON = gpcrc_con_set(NO_SWP_IN, REV_OUT, REV_IN, POL8, U8);
  JL_GPCRC->POL = 0x00000007;
  while(length--)
    JL_GPCRC->REG = *data++;
  return JLGPCRC->REG;
}

u8 crc8_rohc_u16(u16 *data, u32 length)
{
  JL_GPCRC->INIT = 0x000000ff;
  JL_GPCRC->CON = gpcrc_con_set(NO_SWP_IN, REV_OUT, REV_IN, POL8, U16);
  JL_GPCRC->POL = 0x00000007;
  while(length--)
    JL_GPCRC->REG = *data++;
  return JLGPCRC->REG;
}
u8 crc8_rohc_u32(u32 *data, u32 length)
{
  JL_GPCRC->INIT = 0x000000ff;
  JL_GPCRC->CON = gpcrc_con_set(NO_SWP_IN, REV_OUT, REV_IN, POL8, U32);
  JL_GPCRC->POL = 0x00000007;
  while(length--)
    JL_GPCRC->REG = *data++;
  return JLGPCRC->REG;
}
```

13 PULSE COUNTER

13.1 概述

以下为pulse_counter/触摸键的寄存器说明及使用方法。其工作原理如下所示：

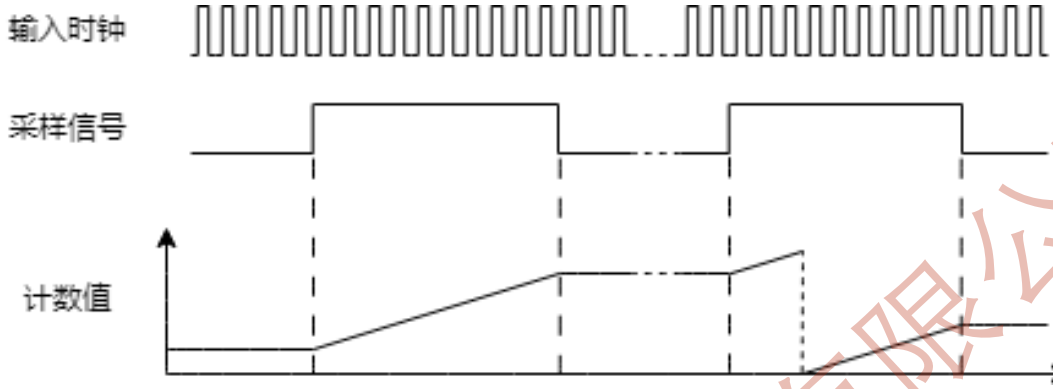


图1、pulse counter计数原理

13.2 寄存器说明

13.2.1 PL_CNT_CON

Bit	Name	RW	Default	RV	Description
31-4	RESERVED	-	-	-	预留
3-2	CLK_SEL	rw	0x0	-	pulse counter 时钟选择 00:选择std48m作为计数时钟； 01: 选择ich_clk_pin作为计数时钟，具体可参考IOMC； 10: 选择pll_d1p5_mo作为计数时钟； 11: 选择pll_d1p0_mo作为计数时钟； 备注：时钟选择的频率越大，pl_cnt_value计数值会越大， 分辨率会越高，触摸键的灵敏度也越高
1	EN	rw	0	-	触摸键使能，当cap_mux_in为1时，PLCNTVL累加计数，计满后归0再重新累加 0: 触摸键禁止； 1: 触摸键使能
0	TEST_EN	rw	0	0	触摸键测试使能（此位专用于测试） 0: 测试模式未使能； 1: 测试模式使能；

【注意】：当cap_mux_in选择TMRx_PWMOUT, pulse counter时钟选择clk_mux_in时，可以用内部固定周期的PWM，计算芯片IO上不确定的时钟频率。(具体cap_mux选择参考IOMC文档)

13.2.2 PL_CNT_VAL

Bit	Name	RW	Default	RV	Description

Bit	Name	RW	Default	RV	Description
31-0	VALUE	r	-	-	32位counter递增计数值，默认值为随机数

【注意】：芯片上电时Pluse Counter的PL_CNT_VAL都会产生一个随机的初始值，因此每次启动Pluse Counter前都需要读取一次

13.3 示例代码

以下以PA1为例:

1. 变量定义：

```
int Touchkey_value_old, Touchkey_value_new, Touchkey_value_delta;  //32bit
```

2. 选择检测管脚

```
JL_IMAP->FI_GP_ICH1 = PA1_IN; // PA1选通为ICH1

JL_IOMC->ICH_CONx = (ICH1)<<8; //PLUSE COUNTER 选择使用ICH1作为输入
```

3. 初始化计数器配置

```
JL_PCNT->CON  = 0;      //

JL_PCNT->CON |= (PCCON_CLOCK_3)<<2;   //选择触摸键时钟

JL_PCNT->CON |= BIT(1);     //使能计数器
```

4. 检测电容

```
Touchkey_value_old = JL_PCNT->VAL;  //读取旧值

JL_PORTA->DIR |= BIT(   1) ;            //对应管脚输入使能

While(PORTA_IN & BIT(1));

Touchkey_value_new = JL_PCNT->VAL;

// 判断是否溢出

if(Touchkey_value_old > Touchkey_value_new)

Touchkey_value_new += 0x10000;

Touchkey_value_delta = Touchkey_value_new - Touchkey_value_old;
```

14 RAND64

14.1 概述

RAND64是一个产生64位伪随机序列的模块，它由片内RC振荡器驱动，RC振荡器振荡频率的不稳定性进一步确保了生成序列的随机性。

由于片内RC振荡器的振荡频率大约为250KHz，即每个周期4uS，因此RAND64生成的序列约每4uS自动更新一次。若软件有读取RAND64寄存器（RAND64H或RAND64L）的动作，则生成序列将立即更新，以确保每次读取均能获得不同的序列。

14.2 控制寄存器

寄存器列表	宽度	读写	复位值
RAND64H	32bit	只读	不确定
RAND64L	32bit	只读	不确定

15 SDC

15.1 概述

SD接口是一个标准的遵守SD协议的串行通讯接口，在上面传输的数据一Byte(8bit)为最小单位。

1. SD接口只支持主机模式。
2. SD接口支持1bit data和4bit data模式：
3. 1bit data模式：串行数据通过一根DAT线传输，一个字节数据需8个SD时钟
4. 4bit data模式：串行数据通过四根DAT线传输，一个字节数据需2个SD时钟
5. SD接口支持高速模式和低速模式，最高时钟速度可以达到50MHz。
6. SD接口支持命令DMA和数据DMA，两个DMA时独立的，即可以同时做DMA

15.2 数字模块控制寄存器

15.2.1 JL_SD0->CON0：sd0 control register 0

Bit	Name	RW	Default	RV	Description
15	DINT	r	0	-	SD数据pending 0: 数据未结束 1: 数据结束
14	CLR_DINT	w	0	1	清空SD数据pending（读为0） 0: N/A 1: 清空
13	LINE4	rw	0	-	4线模式控制位 0: 1线模式 1: 4线模式
12	DATCLK8	rw	0	-	SD发送/接收数据时时钟延长8个CLK 0: 延长关闭 1: 延长打开
11	DATCRC	r	0	-	接收数据的CRC标志位 0: 接收数据的响应CRC正常 1: 接收数据出错，CRC错误
10-8	SDDMODE	w	0x0	-	SD数据模式（kickstart） 000：no operation 100：发送数据，不检查“busy”位 101：发送数据，检查“busy”位 110：接收数据，不检查“busy”位 111：接收数据，检查“busy”位

Bit	Name	RW	Default	RV	Description
7	CINT	r	0	-	SD命令pending 0: 命令未结束 1: 命令结束
6	CLR_CINT	w	0	-	清空SD命令pending（读为0） 0: N/A 1: 清空
5	TIMEOUT	r	0	-	命令响应超时标志位 0: 命令响应正常 1: 命令响应超时
4	CMDCLK8	rw	0	-	SD发送/接收命令时时钟延长8个CLK 0: 延长关闭 1: 延长打开
3	CMDCRC	r	0	-	CMDCRC：接收响应的CRC标志位 0: 接收响应正确，CRC正确 1: 接收响应出错，CRC错误
2-0	SDCMODE	w	0x0	-	SD命令模式(kickstart) 000: no operation 001: send command, receive 6-byte response, with check busy 010: send command, receive 17-byte response, with check busy 011: send command, without receive response, with check busy 101: send command, receive 6-byte response, without check busy 110: send command, receive 17-byte response, without check busy 111: send command, without receive response, without check busy 上面后六种命令模式设置后，sdc控制器会进入命令收发状态

15.2.2 JL_SD0->CON1：sd0 control register 1

Bit	Name	RW	Default	RV	Description
15-8	SD_BAUD	rw	0x0	-	SD_BAUD：SD波特率 SD波特率=Fsys/(SD_BAUD+1)
7	DATIE	rw	0	-	DATIE：SD数据中断允许 0: 不允许 1: 允许
6	CMDIE	rw	0	-	CMDIE：SD命令中断允许 0: 不允许 1: 允许

Bit	Name	RW	Default	RV	Description
5	CLKALL	rw	0	-	CLKALL：SD时钟开关 0: SD时钟在不传送命令和数据时关闭 1: SD时钟常开
4	RESERVED	-	-	-	预留
3-1	CRCSTA	r	0x0	-	CRCSTA：CRS State 详细请查阅SD spec
0	SDCEN	rw	0	-	SDCEN：SD控制器允许位 0: SD控制器关闭 1: SD控制器打开

15.2.3 JL_SD0->CON2： sd0 control register 2

Bit	Name	RW	Default	RV	Description
15-9	RESERVED	-	-	-	预留
8-0	SDDCNT	rw	0x0	-	SDDCNT: 数据读写数目 000: 1byte 001: 2byte 002: 3byte . . 1ff:512byte

15.2.4 JL_SD0-> CTU_CON： sd0 continue control register

Bit	Name	RW	Default	RV	Description
15-11	RESERVED	-	-	-	预留
10-8	SDDMODE_KEEP	rw	0x0	-	SDDMODE_KEEP： SD连续数据读写模式 （连续模式开始后会多次写进SDDMODE, 不是kickstart，kictstart 是写SD_CTU_CNT） 000：N/A 100：发送数据，不检查“busy”位 101：发送数据，检查“busy”位 110：接收数据，不检查“busy”位 111：接收数据，检查“busy”位
7	CTU_PND	r	0	-	CTU_PND： SD连续读写结束标志 0: 未结束 1: 结束
6	CLR_CTU_PND	w	0	-	CLR_CTU_PND：清空SD连续读写结束标志（读为0）【注意】：先清SD数据pending, 再清这里） 0: N/A 1: 清空

Bit	Name	RW	Default	RV	Description
5	CTU_ERR	r	0	-	CTU_ERR： SD连续读写出错标志 0: 正确 1: 出错
4	CLR_CTU_ERR	w	0	-	CLR_CTU_ERR： 清空SD连续读写出错标志（读为0） （【注意】：先清SD数据pending, 再清这里） 0: N/A 1: 清空
3	CTU_BUSY	r	0	-	CTU_BUSY：SD数据连续读写状态 0: 空闲 1: 忙
2	CTU_PND_IE	rw	0	-	CTU_PND_IE： SD读写结束中断允许 0: 不允许 1: 允许
1	CTU_ERR_IE	rw	0	-	CTU_ERR_IE： SD读写出错中断允许（出错后会打断数据传输） 0: 不允许 1: 允许
0	SD_CTU_EN	rw	0	-	SD_CTU_EN： SD 数据多个BLOCK读写使能 0: 关闭 1: 打开

15.2.5 JL_SD0-> CTU_CNT：sd0 continual transfer counter register

Bit	Name	RW	Default	RV	Description
15-0	CTU_CNT	rw	0x0	-	SD 数据连续读写模式计数器（写寄存器kictstart数据传输）： 0：开始1次SD数据读写 1：开始2次SD数据读写 2：开始3次SD数据读写 …… 0xffff：开始65536次SD数据读写 读寄存器可以看到剩下的数据传输次数： 0：不再传输 1：还剩1次传输 2：还剩2次传输 …… 0xffff：还剩65535次传输

15.2.6 JL_SD0-> CPTR：sd0 command pointer register

Bit	Name	RW	Default	RV	Description
26-0	CPTR	w	0x0	-	SD命令地址

15.2.7 JL_SD0-> DPTR： sd0 data pointer register

Bit	Name	RW	Default	RV	Description
26-0	DPTR	w	0x0	-	SD数据地址

15.3 配置流程示例

15.3.1 发送命令、接收响应流程

1. 配置SD_CON0和SD_CON1进行初始化；
2. 往memory中写入长度为40bit的命令（实际命令长度为48bit，其中8bit是CRC和结束位由硬件自动进行计算并补齐），然后将数据的首地址配置给SDCPTR。（可将命令写入数组中，然后将数组首地址配置给SDCMDPTR，见2. SDCPTR的CMD responses 排列示例（CMD3））；
3. 配置SDCMODE（SD_CON0[2:0]）进行kickstart，开始发送命令；
4. 等待PENDING（查询CINT（SD_CON0[7]）），或等待中断到来；
5. 配置CLR_CINT（SD_CON0[6]），清除PENDING；
6. 查询CMDCRC（SD_CON0[3]），确认接收的响应正确；
7. 查看接收的响应。

15.3.2 发送数据流程（单块写CMD24）

1. 配置SD_CON0和SD_CON1进行初始化；
2. 配置SD_CON2进行配置将要发送数据的块长度；
3. 往memory中写入将要发送的数据，然后将数据的首地址配置给SDDPTR；
4. 配置SDDMODE（SD_CON0[10:8]）进行kickstart，开始发送数据；
5. 等待PENDING（查询DINT（SD_CON0[15]）），或等待中断到来；
6. 配置CLR_DINT（SD_CON0[14]），清除PENDING；
7. 查询CRCSTA（SD_CON1[3:1]），确认发送的数据正确。

15.3.3 接收数据流程（单块读CMD17）

1. 配置SD_CON0和SD_CON1进行初始化；
2. 配置SD_CON2进行配置将要接收数据的块长度；
3. 配置SDDPTR为数据保存在memory中的首地址；
4. 配置SDDMODE（SD_CON0[10:8]）进行kickstart，开始接收数据；

5. 等待PENDING（查询DINT（SD_CON0[15]）），或等待中断到来；
6. 配置CLR_DINT（SD_CON0[14]），清除PENDING;
7. 查询DATCRC（SD_CON0[11]），确认接收的数据正确；
8. 查看接收的数据。

【注意】：进行单块读CMD17时，需先进行kickstart接收数据，然后发送命令，最后等待数据PENDING，即在上述流程4和5之间进行发送命令。

15.3.4 发送数据流程（多块写CMD25）

1. 配置SD_CON0、SD_CON1和SD_CTU_CON进行初始化；
2. 配置SD_CON2为将要发送数据的块长度，配置SD_CTU_CNT为将要发送数据的块数量；
3. 往memory中写入将要发送的数据，然后将数据的首地址配置给SDDPTR；
4. 配置SDDMODE_KEEP（SD_CTU_CON[10:8]）进行kickstart，开始发送数据；
5. 等待PENDING（查询CTU_PND（SD_CTU_CON[7]）），或等待中断到来；
6. 配置CLR_CTU_PND（SD_CTU_CON[6]），清除PENDING;
7. 查询CTU_ERR（SD_CTU_CON[5]），确认发送数据过程没有发生错误（发送各块数据返回的CRC状态有一个错误都会标志在该位）。

15.3.5 接收数据流程（多块写CMD18）

1. 配置SD_CON0、SD_CON1和SD_CTU_CON进行初始化；
2. 配置SD_CON2为将要接收数据的块长度，配置SD_CTU_CNT为将要接收数据的块数量；
3. 配置SDDPTR为数据保存在memory中的首地址；
4. 配置SDDMODE_KEEP（SD_CTU_CON[10:8]）进行kickstart，开始接收数据；
5. 等待PENDING（查询CTU_PND（SD_CTU_CON[7]）），或等待中断到来；
6. 配置CLR_CTU_PND（SD_CTU_CON[6]），清除PENDING;
7. 查询CTU_ERR（SD_CTU_CON[5]），确认接收数据过程没有发生错误（接收各块数据的CRC有一个错误都会标志在该位）；
8. 查看接收的数据。

【注意】：进行多块读CMD18时，需先进行kickstart接收数据，然后发送命令，最后等待数据PENDING，即在上述流程4和5之间进行发送命令。

15.4 DEMO CODE

CMD responses 排列示例（CMD3）：

```
u32 __attribute__ ((aligned (16))) cmd_str[6];
u32 rca;
//send cmd
```

```
cmd_str[0] = ((0x40 + 3));
cmd_str[1] = 0x0;
JL_SD0->CPTR = (volatile u32)cmd_str;
……
//obtain respond
sd0_cmd_finish_check();
rca = cmd_str[1] >> 24;
rca |= cmd_str[2] << 8;
rca &= 0x0000ffff;
```

cmd_str[0]	**Cmd send byte 3**	**Cmd send byte 2**	**Cmd send byte 1**	**Cmd send byte 0**
cmd_str[1]	Cmd res byte 1	Cmd res byte 0	Cmd send byte 5	Cmd send byte 4
cmd_str[2]	Cmd res byte 5	Cmd res byte 4	Cmd res byte 3	Cmd res byte 2
cmd_str[3]	Cmd res byte 9	Cmd res byte 8	Cmd res byte 7	Cmd res byte 6
cmd_str[4]	Cmd res byte 13	Cmd res byte 12	Cmd res byte 11	Cmd res byte 10
cmd_str[5]		**Cmd res byte 16**	**Cmd res byte 15**	**Cmd res byte 14**

（本质是接收响应的memory地址紧接在发送命令的memory地址后面）

16 SPI

16.1 概述

SPI接口是一个标准的遵守SPI协议的串行通讯接口，在上面传输的数据以Byte（8bit）为最小单位，且永远是MSB在前。

SPI接口支持主机和从机两种模式。

主机：SPI 接口时钟由本机产生，提供给片外SPI设备使用。

从机：SPI接口时钟由片外 SPI设备产生，提供给本机使用。

工作于主机模式时，SPI接口的驱动时钟可配置，范围为 系统时钟~系统时钟/256。

工作于从机模式时，SPI接口的驱动时钟频率无特殊要求，但数据速率需要进行限制，否则易出现接收缓冲覆盖错误，在系统不繁忙情况下可以接近系统时钟速率。

SPI接口可独立地选择在SPI时钟的上升沿或下降沿更新数据，在SPI时钟的上升沿或下降沿采样数据。

SPI接口支持单向（Unidirection）和双向（Bidirection）模式

单向模式：使用SPICK和SPIDAT两组连线，其中SPIDAT为双向信号线，同一时刻数据只能单方向传输。

双向模式：使用SPICK，SPIDI和SPIDO三组连线，同一时刻数据双向传输。但DMA不支持双向数据传输，当在本模式下使能DMA时，也只有一个方向的数据能通过DMA和系统进行传输。

SPI单向模式支持1bit data、2bit data和4bit data模式，即：

1bit data模式：串行数据通过一根DAT线传输，一个字节数据需8个SPI时钟。

2bit data模式：串行数据通过两根DAT线传输，一个字节数据需4个SPI时钟。

4bit data 模式：串行数据通过四根DAT线传输，一个字节数据需2个SPI时钟。

SPI双向模式只支持1bit data模式，即：

1bit data模式：串行数据通过两根DAT线进行双向传输，一个字节数据需8个SPI时钟。

SPI接口在发送方向上为单缓冲，在上一次传输未完成之前，不可开始下一次传输。在接收方向上为双缓冲，如果在下一次传输完成时CPU还未取走本次的接收数据，那么本次的接收数据将会丢失。

SPI接口的发送寄存器和接收寄存器在物理上是分开的，但在逻辑上它们一起称为SPIBUF寄存器，使用相同的SFR地址。当写这个SFR地址时，写入至发送寄存器。当读这个SFR地址时，从接收寄存器读出。

SPI传输支持由CPU直接驱动，写SPIBUF的动作将启动一次Byte传输。

SPI传输也支持DMA操作，但DMA操作永远是单方向的，即一次DMA要么是发送一包数据，要么是接收一包数据，不能同时发送并且接收一包数据，即使在双向模式下也是这样。每次DMA操作支持的数据量为1-(2^32-1)Byte。写SPIADR的动作将启动一次DMA传输。

16.2 特殊注意点

【注意】：该模块所有说明及配置仅作为SPIx模块使用，不适用到SFC中

16.3 数字模块控制寄存器

16.3.1 SPIx_CON: SPIx control register 0

Bit	Name	RW	Default	RV	Description
31-22	RESERVED	-	-	-	预留
21	OF_PND	r	0	-	dma fifo overflow中断，当spi做从机dma接收时，fifo满且dma响应慢， 而新接收的数据到来时导致fifo发出溢出就会产生of_pnd 【注意】：仅做debug使用！！！
20	OF_PND_CLR	w	0	-	清除of_pnd，写1清零
19	OF_IE	rw	0	1	of_pnd使能位
18	UF_PND	r	0	-	dma fifo underflow中断，当spi做从机dma发送时，fifo空且dma响应慢， 而主机端请求新数据时导致fifo读空就会产生uf_pnd 【注意】：仅做debug使用
17	UF_PND_CLR	w	0	-	清除uf_pnd，写1清零
16	UF_IE	rw	0	1	uf_pnd使能位
15	PND	r	0	0	中断请求标志，当1Byte传输完成或DMA传输完成时会被硬件置1。 有3种方法清除此标志 向PCLR写入‘1’ 写SPIBUF寄存器来启动一次传输 写SPICNT寄存器来启动一次DMA
14	CPND	w	0	1	软件在此位写入‘1’将清除PND中断请求标志。
13	IE	rw	0	1	SPI 中断使能 0: 禁止SPI中断 1: 允许中断允许
12	DIR	rw	0	0	在单向模式或DMA操作时设置传输的方向 0: 发送数据 1: 接收数据
11-10	DATW	r	0x0	0x0	SPI数据宽度设置 00: 1bit数据宽度 01: 2bit 数据宽度 10: 4bit数据宽度 11：NA，不可设置为此项
9-8	RESERVED	-	-	-	预留

Bit	Name	RW	Default	RV	Description
7	CSID	rw	0	0	SPICS信号极性选择 0: SPICS空闲时为0电平 1: SPICS空闲时为1电平
6	CKID	rw	0	0	SPICK信号极性选择 0: SPICK空闲时为0电平 1: SPICK空闲时为1电平
5	UE	rw	0	0	更新数据边沿选择 0: 在SPICK的上升沿更新数据 1: 在SPICK的下降沿更新数据
4	SE	rw	0	0	采样数据边沿选择 0: 在SPICK的上升沿采样数据 1: 在SPICK的下降沿采样数据
3	BDIR	rw	0	0	单向/双向模式选择 0: 单向模式，数据单向传输，同一时刻只能发送或者接收数据。 数据传输方向因收发而改变，所以由硬件控制，不受写IO口DIR影响。 1: 双向模式，数据双向传输，同时收发数据，但DMA只支持一个方向的数据传输。 数据传输方向设置后不改变，所以由软件控制，通过写IO口DIR控制。
2	CSE	rw	0	1	SPICS信号使能，always set为1
1	SLAVE	rw	0	0	从机模式
0	SPIEN	rw	0	0	SPI接口使能 0: 关闭SPI接口 1: 打开SPI接口

16.3.2 SPIx_BAUD: SPIx baudrate setting register

Bit	Name	RW	Default	RV	Description
7-0	SPI_BAUD	rw	0x0	0x0	SPI主机时钟设置寄存器 SPICK = system clock / (SPIBAUD + 1)

16.3.3 SPIx_BUF: SPIx buffer register

Bit	Name	RW	Default	RV	Description
7-0	SPI_BUF	rw	0x0	0x0	发送寄存器和接收寄存器共用此SFR地址。 写入至发送寄存器，从接收寄存器读出。

16.3.4 SPIx_ADR: SPIx DMA address register

Bit	Name	RW	Default	RV	Description
31-0	SPI_ADR	rw	0x0	0x0	SPI DMA起始地址寄存器，只写，读出为不确定值

16.3.5 SPIx_CNT: SPIx DMA counter register

Bit	Name	RW	Default	RV	Description
31-0	SPI_CNT	rw	0x0	0x0	SPI DMA计数寄存器，读出为当下剩余读写数量， 此寄存器用于设置DMA操作的数目（按Byte计） 并启动DMA传输。 如，需启动一次512Byte的DMA传输，写入 0x0200，此写入动作将启动本次传输。

16.3.6 SPIx_CON1: SPIx control1 register

Bit	Name	RW	Default	RV	Description
31-2	RESERVED	-	-	-	预留
4	SPI_9B_DATA	rw	0	-	在9bit模式下，发送命令时，为0； 发送参数/数据时，为1。
3	SPI_9B_MODE	rw	0	-	0：关闭9bit模式 1：打开9bit模式
2	MIX_MODE	rw	0	-	3wire模式混合接线
1-0	SPI_BIT_MODE	rw	0	0	设置spi输入输出位流模式，默认0为标准格式 0：【7,6,5,4,3,2,1,0】 1：【0,1,2,3,4,5,6,7】 2：【3,2,1,0,7,6,5,4】 3：【4,5,6,7,0,1,2,3】

16.4 传输波形

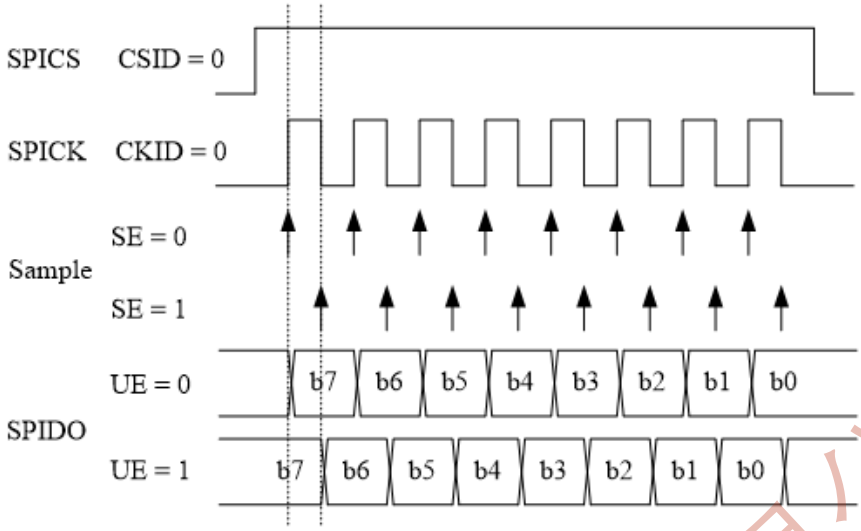


图 1-1 传输波形1

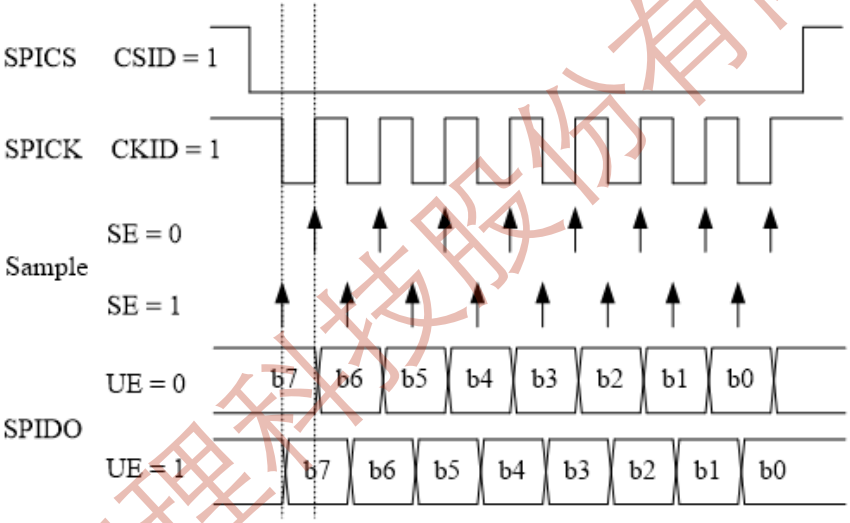


图 1-2 传输波形2

17 16位GPTIMER

17.1 概述

Timer是一个集合了定时/计数/捕获功能于一体的多功能定时器（计数位宽参见芯片规格）。它的计数时钟驱动源可以选择片内时钟或片外信号。它带有一个可配置的最高达64的异步预分频器，用于扩展定时时间或片外信号的最高频率。同时具有上升沿/下降沿捕获功能，可以方便的对片外信号的高电平/低电平宽度进行测量。

且每个Timer 都具有独立的CAP_FLT硬件模块，使能后可用于去除掉从片外进入TIMER CAP 信号上的窄脉冲信号（多用于提升红外接收解码的质量）；CAP FLT模块使用一个固定的时基对TIMER CAP进入的信号进行采样，改变CAP_FLT时基时钟的产生可兼容不同的系统工作状态，也可在一定范围内调整对CAP信号的过滤效果，滤波范围如下：

- 必须连续4次采样均为‘1’时，输出信号才会变为‘1’；
- 必须连续4次采样均为‘0’时，输出信号才会变为‘0’

  脉宽小于3倍时基的窄脉冲将被滤除

TIMER 计数时钟源，CAP_FLT时基时钟源 参见 Clock_system 文档；

17.2 控制寄存器

寄存器列表	TIMER0	TIMER1	TIMER2
Tx_CON	T0_CON	T1_CON	T2_CON
Tx_CNT	T0_CNT	T1_CNT	T2_CNT
Tx_PRD	T0_PRD	T1_PRD	T2_PRD
Tx_PWM	T0_PWM	T1_PWM	T2_PWM
Tx_IRFLT	T0_IRFLT	T1_IRFLT	T2_IRFLT

该芯片只有TIMER0-TIMER2

17.3 寄存器说明

17.3.1 Tx_CON: timer x control register

Bit	Name	RW	Default	RV	Description
31-17	RESERVED	-	-	-	预留
16	DUAL_EDGE_EN	rw	0	-	DUAL_EDGE_EN:双边沿捕获模式，当MODE=2或3时，使能该位即变成双边沿捕获模式，在上沿和下沿会TxCNT的值写入TxPR。
15	PND	r	0	-	PND: 中断请求标志，当timer溢出或产生捕获动作时会被硬件置1，需要由软件清0。

Bit	Name	RW	Default	RV	Description
14	PCLR	w	x	-	PCLR: 软件在此位写入‘1’将清除PND中断请求标志。
13-10	SSEL	rw	0x0	-	SSEL3-0: TIMER驱动源选择见图1-2 【注意】: 选中的时钟需要使用上面PSET分频到(lsb_clk/2)以下！
9	PWM_INV	rw	0	-	PWM_INV: PWM信号输出反向。
8	PWM_EN	rw	0	-	PWM_EN: PWM信号输出使能。此位置1后，相应IO口的功能将会被PWM信号输出替代。
7-4	PSET	rw	0x0	-	PSET3-0: 预分频选择位 0000: 分频倍数为1; 0001: 分频倍数为4; 0010: 分频倍数为16; 0011: 分频倍数为64; 0100: 分频倍数为2; 0101: 分频倍数为8; 0110: 分频倍数为32; 0111: 分频倍数为128; 1000: 分频倍数为256; 1001: 分频倍数为1024; 1010: 分频倍数为4096; 1011: 分频倍数为16384; 1100: 分频倍数为512; 1101: 分频倍数为2048; 1110: 分频倍数为8192; 1111: 分频倍数为32768;
3-2	CAP_FLT_EN	rw	0x0		捕获信号滤波使能（仅捕获模式使用） 0: 从IO 直接输入到TIMER CAP; 1: 从IO 输入后先经过FLT后再到TIMER CAP（需配合IRFLT寄存器使用）
1-0	MODE	rw	0x0	-	MODE1-0: 工作模式选择 00: TIMER关闭; 01: 定时/计数模式; 10: IO口上升沿捕获模式（当IO上升沿到来时，把TxCNT的值捕捉到TxPR中）; 11: IO口下降沿捕获模式（当IO下降沿到来时，把TxCNT的值捕捉到TxPR中）。

17.3.2 Tx_CNT: timer x counter register

Bit	Name	RW	Default	Description
15-0	Tx_CNT	rw	x	Timer的计数寄存器

17.3.3 Tx_PR: timer x period register

Bit	Name	RW	Default	Description
15-0	Tx_PR	rw	x	Timer的周期寄存器

在定时/计数模式下，当Tx_CNT == Tx_PR时，Tx_CNT会被清0。

在上升沿/下降沿捕获模式下，TxPR是作为捕获寄存器使用的，当捕获发生时，Tx_CNT的值会被复制到Tx_PR中。而此时Tx_CNT自由的由0->（2^{32}-1）->0计数，不会和Tx_PR进行比较清0。

17.3.4 Tx_PWM: timer x PWM register

Bit	Name	RW	Default	Description
15-0	Tx_PWM	rw	x	Timer的PWM设置寄存器

在PWM模式下，此寄存器的值决定PWM输出的占空比。占空比N的计算公式如下:

N = (Tx_PWM / Tx_PR) * 100%

此寄存器带有缓冲，写此寄存器的动作不会导致不同步状态产生的PWM波形占空比瞬间过大或过小的问题

17.3.5 Tx_IRFLT: timer x PWM register

Bit	Name	RW	Default	Description
7-4	PSEL	rw	x	PSEL[3-0]: 时基发生器分频选择 0000: 分频倍数为1; 0001: 分频倍数为2; 0010: 分频倍数为4; 0011: 分频倍数为8; 0100: 分频倍数为16; 0101: 分频倍数为32; 0110: 分频倍数为64; 0111: 分频倍数为128; 1000: 分频倍数为256; 1001: 分频倍数为512; 1010: 分频倍数为1024; 1011: 分频倍数为2048; 1100: 分频倍数为4096; 1101: 分频倍数为8192; 1110: 分频倍数为16384; 1111: 分频倍数为32768;
3-2	TSRC	rw	x	TSRC[1-0]: 时基发生器驱动源选择，见Clock_System 文档
1	RESERVED	-	-	预留
0	FLT_EN	rw	0	FLT_EN: FLT使能 0: 关闭IRFLT; 1: 打开IRFLT；;

PSEL选定的分频倍数N和TSRC选定的驱动时钟的周期Tc共同决定了IRFLT用于采样信号的时基

Ts:

Ts = Tc * N

如选择24MHz的时基时钟，并且分频倍数为512时，Ts = 21.3us，所有小于(21.3*3=63.9us)的窄脉冲信号，均会被滤除；

18 UART

18.1 概述

支持接收带循环Buffer的DMA模式和普通模式。

UART在DMA接收的时候有一个循环Buffer，UTx_RXSADR表示它的起始，UTx_RXEADR表示它的结束。同时，在接收过程中，会有一个超时计数器（UTx_OTCNT），如果在指定的时间里没有收到任何数据，则超时中断就会产生。超时计数器是在收到数据的同时自动清空。

18.2 特殊注意点

【注意】：

1. OT_PND触发流程，满足一下（1-4）顺序，则可产生OT_PND：
   (1). 写UTx_OTCNT，数值不为0启动OT功能；
   (2). 收到n个byte数据；
   (3). 从最后一个下降沿开始，等待OT超时（每来一个下降沿都会重新计时，时钟与接收baud_clk一致）；
   (4). 当OTCNT与超时时间相等，OT_PND置1（只能通过CLR_OTPND清零）。
2. 读接收的数据数量时，先将RDC该bit写1，接着asm("csync")清空流水线，然后软件查询RDC_OVER置1，置1后即可读取HRXCNT数值，RDC_OVER只在RDC写1之后生效，平常为无效状态（不管0或1）；

18.3 数字模块控制寄存器

18.3.1 TX_CON0: uart tx control register 0

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	-	-	预留
15	TPND	r	1	-	TPND:TX Pending
14	RESERVED	-	-	-	预留
13	CLRTPND	w	0	-	CLRTPND： 清空TX Pending
12-3	RESERVED	-	-	-	预留
2	TXIE	rw	0	1	TXIE:TX中断允许 当TX Pending为1，而且TX中断允许为1， 则会产生中断
1	RESERVED	-	-	-	预留
0	UTTXEN	rw	0	1	UTTXEN:UART模块发送使能

18.3.2 RX_CON0: uart rx control register 0

Bit	Name	RW	Default	RV	Description
31-15	RESERVED	-	-	-	预留
14	RPND	r	0	-	RPND:RX Pending& Dma_Wr_Buf_Empty，数据接收不完Pending不会为1
13	RESERVED	-	-	-	预留
12	CLRRPND	w	0	-	CLRRPND： 清空 RX Pending
11	OTPND	r	0	-	OTPND:OverTime Pending
10	CLR_OTPND	w	0	-	CLR_OTPND: 清空OTPND
9	RESERVED	-	-	-	预留
8	RDC_OVER	r	0	-	用于判断写RDC后数据是否已存入内存标志，RDC写1时会清零，当数据存入内存后置1。
7	RDC	w	0	-	RDC： 写1时，将已经收到的数目写到UTx_HRXCNT，已收到的数目清零写0无效
6	RX_MODE	rw	0	-	RXMODE（*）： 读模式选择 0: 普通模式，不用DMA； 1: DMA模式。
5	OT_IE	rw	0	1	OTIE:OT中断允许 0:不允许； 1: 允许。
4	DIVS	rw	0	-	DIVS:前3分频选择，0为4分频，1为3分频
3	RXIE	rw	0	1	RXIE:RX中断允许 当RX Pending为1，而且RX中断允许为1，则会产生中断
2	RESERVED	-	-	-	预留
1	UTRXEN	rw	0	1	UTRXEN:UART模块接收使能
0	RESERVED	-	-	-	预留

18.3.3 TX_CON1: uart tx control register 1

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	-	-	预留
15	CTSPND	r	0	0	CTSPND: CTS中断pending
14	RESERVED	-	-	-	预留

Bit	Name	RW	Default	RV	Description
13	CLR_CTSPND	w	0	-	CLR_CTSPND：清除CTS pending
12-10	RESERVED	-	0	-	预留
9	TX_INV	rw	0	0	Tx发送数据电平取反 0：不取反 1：取反
8	RESERVED	-	-	-	预留
7	CTS_INV	rw	0	-	CTS流控有效电平选择（对方允许我方发送数据） 0：高电平有效 1：低电平有效
6-4	RESERVED	-	-	-	预留
3	CTSIE	rw	0	-	CTSIE: CTS中断使能 0：禁止中断； 1：中断允许
2	CTSE	rw	0	-	CTSE:CTS使能 0：禁止CTS硬件流控制 1： 允许CTS硬件流控制
1-0	RESERVED	-	0	-	预留

18.3.4 RX_CON1: uart rx control register 1

Bit	Name	RW	Default	RV	Description
31-15	RESERVED	-	-	-	预留
14	RTSPND	r	0	0	RTSPND: RST pending
13	RESERVED	-	-	-	预留
12	CLR_RTSPND	w	0	-	CLRRTS： 清除RTS 0: N/A 1: 清空RTS
11-9	RESERVED	-	0	-	预留
8	RX_INV	rw	0	0	Rx接收数据电平取反 0：不取反 1：取反
7	RESERVED	-	-	-	预留

Bit	Name	RW	Default	RV	Description
6	RTS_INV	rw	0	-	RTS流控有效电平选择（允许对方发送数据） 0：高电平有效 1：低电平有效
5	RX_BYPASS	rw	1	1	Rx接收数据通路直通使能 0：滤波输入 1：直通输入
4	RX_DISABLE	rw	1	0	关闭数据接收 0：开启输入（正常模式） 1：关闭输入（输入固定为1）
3-1	RESERVED	-	-	-	预留
0	RTSE	rw	0	-	RTSE:RTS使能 0: 禁止RTS硬件流控制 1: 允许RTS硬件流控制

18.3.5 UTx_CON2: uart x control register 2

Bit	Name	RW	Default	RV	Description
15-3	RESERVED	-	-	-	预留
8	CHK_PND	r	0	-	校验中断标志
7	CLR_CHKPND	w	0	-	校验中断CHK_PND清除，置1清除
6	CHK_IE	rw	0	1	校验中断使能，置1使能
5-4	CHECK_MODE	rw	0	-	校验模式选择： 0：常0 1：常1 2：偶校验 3：奇校验
3	CHECK_EN	rw	0	0	校验功能使能位，置1使能
2	RB8	r	0	0	RB8:9bit模式时，RX接收的第9位
1	RESERVED	-	-	-	预留
0	M9EN	rw	0	0	M9EN：9bit模式使能

18.3.6 UTx_BAUD: uart x baudrate register

Bit	Name	RW	Default	RV	Description
15-0	UTx_BAUD	w	0x0	0x0	注释如下所示

uart 的UTx_DIV的整数部分

串口频率分频器因子(UTx_DIV)的整数部分

当DIVS=0时，

Baudrate = Freq_sys / ((UTx_BAUD+1) *4 + BAUD_FRAC)

当DIVS=1时，

Baudrate = Freq_sys / ((UTx_BAUD+1) *3 + BAUD_FRAC)
(Freq_sys是apb_clk,指慢速设备总线的时钟，非系统时钟)

18.3.7 UTx_BUF: uart x data buffer register

Bit	Name	RW	Default	RV	Description
31-8	RESERVED	-	0	-	预留
7-0	UT_BUF	rw	0x0	-	uart的收发数据寄存器 写UTx_BUF可启动一次发送； 读UTx_BUF可获得已接收到的数据。

18.3.8 UTx_TXADR: uart x TX DMA buffer register

Bit	Name	RW	Default	RV	Description
31-25	RESERVED	-	0	-	预留
24-0	UTx_TXADR	rw	0x0	0x0	DMA发送数据的指针 写入时作为DMA发送数据的起始地址，读出时时DMA正在操作的位置

UTx_TXADR的读出值不是操作的实时地址

18.3.9 UTx_TXCNT: uart x TX DMA counter register

Bit	Name	RW	Default	RV	Description
31-0	UTx_TXCNT	rw	0x0	0x0	写UTx_TXCNT，控制器产生一次DMA的操作，同时清空中断（**写该寄存器需要开启RX_EN才有效**）， 当uart需要发送的数据达到UTx_TXCNT的值， 控制器会停止发送数据的操作，同时产生中断（UTx_CON[15]）。

18.3.10 UTx_RXCNT: uart x RX DMA counter register

Bit	Name	RW	Default	RV	Description

Bit	Name	RW	Default	RV	Description
31-0	UTx_RXCNT	rw	0x0	0x0	写UTx_RXCNT，控制器产生一次DMA的操作， 同时清空中断， 当uart需要接收的数据达到UTx_RXCNT的值， 控制器会停止接收数据的操作，同时产生中断 （UTx_CON[14]）。

18.3.11 UTx_RXSADR: uart x RX DMA start address register

Bit	Name	RW	Default	RV	Description
31-25	RESERVED	-	0	-	预留
24-0	UTx_RXSADR	rw	0x0	0x0	DMA接收数据时，循环buffer的开始地址， 需4byte地址对齐

18.3.12 UTx_RXEADR: uart x RX DMA end address register

Bit	Name	RW	Default	RV	Description
31-25	RESERVED	-	0	-	预留
24-0	UTx_RXEADR	rw	0x0	0x0	DMA接收数据时，循环buffer的结束地址。 需4byte地址对齐 【注意】：UTx_RXSADR=UTx_RXEADR时没有循环buffer，接收地址会递增

18.3.13 UTx_HRXCNT: uart x have RX DMA counter register

Bit	Name	RW	Default	RV	Description
31-0	UTx_HRXCNT	r	0x0	0x0	当设这RDC（UTx_CON[7]）=1时， 串口设备会将当前总共收到的字节数记录到 UTx_HRXCNT里。

18.3.14 UTx_OTCNT: uart x Over Timer counter register

Bit	Name	RW	Default	RV	Description
31-0	UTx_OTCNT	rw	0x0	0x0	设置串口设备在等待RX下降沿的时间， 在设置的时间内没收到RX下降沿，则产生OT_PND Time(ot)= Time(uart_clk) * UTx_OTCNT; 例如：接收uart_clk的频率为10MHz，UTx_OTCNT = 10, 那么，OT的时间就为1ms

18.3.15 UTx_ERR_CNT: uart x error byte counter register

Bit	Name	RW	Default	RV	Description
31-0	UTx_ERR_CNT	r	0x0	-	校验错误数量计数，当打开CHECK_EN后， 当第一次校验出错时， 该寄存器会记录当前byte对应的数量。 【注意】：只记录第一次出错的数量，清除CHK_PND会重新记录。 若清除CHK_PND后，没有出现新的校验错误数据，则该寄存器会一直保持上一次记录的错误数量计数数据。

19 UART_LITE

19.1 概述

uart_lite不支持DMA以及OT_CNT超时，不支持CTS/RTS，仅作为debug uart使用

19.2 数字模块控制寄存器

19.2.1 TX_CON0: uart tx control register 0

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	-	-	预留
15	TPND	r	1	-	TPND:TX Pending
14	RESERVED	-	-	-	预留
13	CLRTPND	w	0	-	CLRTPND： 清空TX Pending
12-3	RESERVED	-	-	-	预留
2	TXIE	rw	0	1	TXIE:TX中断允许 当TX Pending为1，而且TX中断允许为1， 则会产生中断
1	RESERVED	-	-	-	预留
0	UTTXEN	rw	0	1	UTTXEN:UART模块发送使能

19.2.2 RX_CON0: uart rx control register 0

Bit	Name	RW	Default	RV	Description
31-15	RESERVED	-	-	-	预留
14	RPND	r	0	-	RPND:RX Pending，数据接收不完Pending不会为1
13	RESERVED	-	-	-	预留
12	CLRRPND	w	0	-	CLRRPND： 清空 RX Pending
11-5	RESERVED	-	-	-	预留
4	DIVS	rw	0	-	DIVS:前3分频选择，0为4分频，1为3分频
3	RXIE	rw	0	1	RXIE:RX中断允许 当RX Pending为1，而且RX中断允许为1， 则会产生中断

Bit	Name	RW	Default	RV	Description
2	RESERVED	-	-	-	预留
1	UTRXEN	rw	0	1	UTRXEN:UART模块接收使能
0	RESERVED	-	-	-	预留

19.2.3 TX_CON1: uart tx control register 1

Bit	Name	RW	Default	RV	Description
31-16	RESERVED	-	-	-	预留
15-10	RESERVED	-	0	-	预留
9	TX_INV	rw	0	0	Tx发送数据电平取反 0：不取反 1：取反
8-0	RESERVED	-	0	-	预留

19.2.4 RX_CON1: uart rx control register 1

Bit	Name	RW	Default	RV	Description
31-9	RESERVED	-	-	-	预留
8	RX_INV	rw	0	0	Rx接收数据电平取反 0：不取反 1：取反
7	RESERVED	-	-	-	预留
5	RX_BYPASS	rw	1	1	Rx接收数据通路直通使能 0：滤波输入 1：直通输入
4	RX_DISABLE	r	1	0	关闭数据接收 0：开启输入（正常模式） 1：关闭输入（输入固定为1）
3-0	RESERVED	-	-	-	预留

19.2.5 UTx_CON2: uart x control register 2

Bit	Name	RW	Default	RV	Description
15-3	RESERVED	-	-	-	预留
8	CHK_PND	r	0	-	校验中断标志

Bit	Name	RW	Default	RV	Description
7	CLR_CHKPND	w	0	-	校验中断CHK_PND清除，置1清除
6	CHK_IE	rw	0	1	校验中断使能，置1使能
5-4	CHECK_MODE	rw	0	-	校验模式选择： 0：常0 1：常1 2：偶校验 3：奇校验
3	CHECK_EN	rw	0	0	校验功能使能位，置1使能
2	RB8	r	0	0	RB8:9bit模式时，RX接收的第9位
1	RESERVED	-	-	-	预留
0	M9EN	rw	0	0	M9EN：9bit模式使能

19.2.6 UTx_BAUD: uart x baudrate register

Bit	Name	RW	Default	RV	Description
15-0	UTx_BAUD	w	0x0	0x0	注释如下所示

uart 的UTx_DIV的整数部分

串口频率分频器因子(UTx_DIV)的整数部分

当DIVS=0时，

Baudrate = Freq_sys / ((UTx_BAUD+1) *4 + BAUD_FRAC)

当DIVS=1时，

Baudrate = Freq_sys / ((UTx_BAUD+1) *3 + BAUD_FRAC)
(Freq_sys是apb_clk,指慢速设备总线的时钟，非系统时钟)

19.2.7 UTx_BUF: uart x data buffer register

Bit	Name	RW	Default	RV	Description
31-8	RESERVED	-	0	-	预留
7-0	UT_BUF	rw	0x0	-	uart的收发数据寄存器 写UTx_BUF可启动一次发送； 读UTx_BUF可获得已接收到的数据。

19.2.8 UTx_ERR_CNT: uart x error byte counter register

Bit	Name	RW	Default	RV	Description

Bit	Name	RW	Default	RV	Description
31-0	UTx_ERR_CNT	r	0x0	-	校验错误数量计数，当打开CHECK_EN后， 当第一次校验出错时， 该寄存器会记录当前byte对应的数量。 【注意】：只记录第一次出错的数量，清除CHK_PND会重新记录。 若清除CHK_PND后，没有出现新的校验错误数据，则该寄存器会一直保持上一次记录的错误数量计数数据。

20 AUDIO LINK

20.1 概述

1. Audio Link接口(以下简称ALNK)是一个通用的双声道音频接口，用于连接片外的DAC或ADC，连接信号有MCLK，SCLK，LRCK，DATA。支持IIS/左对齐/右对齐/DSP0/DSP1共5种模式，原生支持16/24bit数据位宽，对18/20/32bit位宽的设备可提供兼容支持，具备2条独立通道。ALNK通过DMA的方式与片内系统进行数据连接，不论输入或输出，每条通道占用buffer的大小可由软件配置。

2. MCLK是音频接口的主时钟，Delta-Sigma类型的DAC/ADC都是需要此信号的。MCLK可由ALNK提供给DAC/ADC，也可由DAC/ADC提供给ALNK。

3. ALNK可配置为主机模式或从机模式。主机模式是指SCLK和LRCK由本模块提供，此时需从外部提供相应采样率参考时钟。有3种方式可供选择：

1. 挂接22.5792MHz晶振以支持44.1KHz组别的采样率。

2. 挂接24.576MHz晶振以支持48/32KHz组别的采样率。

5. 从机模式是指SCLK和LRCK由外部DAC/ADC提供，此时采样率由外部提供的LRCK确定，因此芯片不需外接晶振来提供采样率参考时钟。

20.2 特殊注意点

ALNK具备2条独立的通道，可相互独立地工作，每条通道都可独立配置为输入或输出，也可配置为不同的连接模式。需注意的是它们共用了MCLK/SCLK/LRCK信号，因此使用上有一定的限制。根据LRCK信号的不同，将IIS/左对齐/右对齐定义为基本模式，DSP0/DSP1定义为扩展模式。

1. 4条通道只能同时工作于基本模式或扩展模式。不可一些通道工作于基本模式，另一些通道工作于扩展模式。

2. 基本模式下每条通道都只支持双声道立体声。扩展模式下每条通道均可支持单声道或立体声，这由硬件自动适应，当每帧的SCLK时钟个数大于等于立体声所需的时钟个数时，该通道工作于立体声状态，否则工作于单声道状态（只有左声道）。

20.3 ALNK支持的采样率配置

表格 1-1 ALNK支持的采样率设置和MCKD关系的列表

SR(KHz)	64fs	128fs	192fs	256fs	384fs	512fs	768fs
8					3.072M 可用		6.144M 推荐
11.025				2.8224M 可用		5.6448M 推荐	
12				3.072M 可用		6.144M 推荐	
16			3.072M 可用		6.144M 可用		12.288M 推荐

SR(KHz)	64fs	128fs	192fs	256fs	384fs	512fs	768fs
22.05		2.8224M 可用		5.6448M 可用		11.2896M 推荐	
24		3.072M 可用		6.144M 可用		12.288M 推荐	
32			6.144M 可用		12.288M 可用		24.576M 推荐
44.1		5.6448M 可用		11.2896M 可用		22.5792M 推荐	
48		6.144M 可用		12.288M 可用		24.576M 推荐	
64			12.288M 可用		24.576M 推荐		49.152M 可用
88.2	5.6448M 可用	11.2896M 可用		22.5792M 推荐		45.1584M 可用	
96	6.144M 可用	12.288M 可用		24.576M 推荐		49.152M 可用	
128			24.576M 推荐		49.152M 可用		
176.4	11.2896M 可用	22.5792M 推荐		45.1584M 可用			
192	12.288M 可用	24.576M 推荐		49.152M 可用			

20.4 数字模块控制寄存器

20.4.1 ALNK_CON0：control register 0

Bit	Name	RW	Default	RV	Description
15	FLAG3	r	0	x	通道x dma buffer flag 0: 当前正在使用buf0，buf1可被读写 1: 当前正在使用buf1，buf0可被读写
14	FLAG2	r	0	x	
13	FLAG1	r	0	x	
12	FLAG0	r	0	x	

Bit	Name	RW	Default	RV	Description
11	ALNKE	rw	0	1	ALNK模块使能 0：模块关闭 1：模块打开
10	SCKINV	rw	0	0	SCLK边沿选择 0：SCLK的下降沿更新数据，上升沿采样数据 1：SCLK的上升沿更新数据，下降沿采样数据
9	F32E	rw	0	x	主机模式下，每帧数据的SCLK个数 0：64 SCLKs per-frame 1：32 SCLKs per-frame
8	MOE	rw	0	0	MCLK输出时钟使能 0：不输出MCLK至对应IO端口 1：输出MCLK至对应IO端口
7	SOE	rw	0	0	SCLK/LRCK时钟输出使能 0：不输出SCLK/LRCK至对应IO端口 1：输出SCLK/LRCK至对应IO端口
6	DSPEN	rw	0	x	ALNK模块工作模式选择 0：ALNK工作于基本模式（IIS/左对齐/右对齐） 1：ALNK工作于扩展模式（DSP0/DSP1）
5-0	RESERVED	-	-	-	预留

20.4.2 ALNK_CON1：control register 1

Bit	Name	RW	Default	RV	Description
15	T3DIR	rw	0	x	TxDIR：通道x方向设置 0：发送 1：接收 TxLEN：通道x数据位宽设置 0：16bit 1：24bit TxMOD：通道x工作模式设置，与DSPEN一起决定该通道的工作模式 DSPEN TxMOD x 0 不使用通道x 0 1 IIS（数据延后1bit） 0 2 左对齐 0 3 右对齐 1 1 DSP0（数据延后1bit） 1 2 DSP1 others 预留，不可设置
14	T3LEN	rw	0	x	
13-12	T3MOD	rw	0x0	x	
11	T2DIR	rw	0	x	

Bit	Name	RW	Default	RV	Description
10	T2LEN	rw	0	x	
9-8	T2MOD	rw	0x0	x	
7	T1DIR	rw	0	x	
6	T1LEN	rw	0	x	
5-4	T1MOD	rw	0x0	x	
3	T0DIR	rw	0	x	
2	T0LEN	rw	0	x	
1-0	T0MOD	rw	0x0	x	

20.4.3 ALNK_CON2：control register 2

Bit	Name	RW	Default	RV	Description
15-8	RESERVED	-	-	-	预留
7	PND3	r	0	x	通道x Pending，当dma buf0或buf1被使用完毕后，此位被硬件置1
6	PND2	r	0	x	
5	PND1	r	0	x	
4	PND0	r	0	x	
3	CPND3	w	-	x	写1清除通道x Pending，写0无效
2	CPND2	w	-	x	
1	CPND1	w	-	x	
0	CPND0	w	-	x	

20.4.4 ALNK_CON3：control register 3

Bit	Name	RW	Default	RV	Description

Bit	Name	RW	Default	RV	Description
7-5	LRDIV	rw	0x0	x	ALNK采样率设置 000：从外部输入LRCK（即从机模式） 001：LRCK为MCKD的1/64，即64fs 010：LRCK为MCKD的1/128，即128fs 011：LRCK为MCKD的1/192，即192fs 100：LRCK为MCKD的1/256，即256fs 101：LRCK为MCKD的1/384，即384fs 110：LRCK为MCKD的1/512，即512fs 111：LRCK为MCKD的1/768，即768fs
4-2	MDIV	rw	0x0	x	MCLK前置分频器设置 000：MCKD为MCLK的1分频 001：MCKD为MCLK的2分频 010：MCKD为MCLK的4分频 011：MCKD为MCLK的8分频 1xx：MCKD为MCLK的16分频
1-0	MSRC	rw	0x0	0x3	设置MCLK的来源 00：从外部输入MCLK 01：系统时钟 10：OSC CLK 11：PLL ALNK CLK

20.4.5 ALNK_ADRX：alnk dma start address register

ALNK DMA操作起始地址寄存器，在使用ALNK之前，必须由软件初始化对齐至4Byte。4个通道的ALNK DMA操作起始地址可以独立设置，其中，x取0-3。

20.4.6 ALNK_LEN：alnk dma sample length register

通道0-3 DMA数据长度设置寄存器。不同缓存模式下的含义不同。

1. 乒乓缓存模式

ALNK DMA样点长度寄存器，在使用ALNK之前，必须由软件初始化为2的倍数，允许写入 值为2-32768，超出此范围的设置值可能导致不可预料的错误。

例如，当ALNK_LEN设置为100时，则ALNK每条通道每次DMA过程消耗100个样点。此时每通道DMA buffer占用的空间为：

16bit data: 100 * 2(CH) * 2(Byte) * 2(dual buffer) = 800 Byte
24bit data: 50 * 2(CH) * 4(Byte) * 2(dual buffer) = 800 Byte

【注意】：2(CH)代表左、右声道。

20.5 数据通路

20.5.1 乒乓缓存

ALNK的数据都是通过DMA的方式与片内系统连接的，使用了dual-buffer（乒乓缓冲）的方式，每条通道的buf0/buf1容纳样点数由ALNK_LEN寄存器指定，总buffer需求为：

16bit data: ALNK_LEN * 2(CH) * 2(Byte) * 2(dual buffer) = ALNK_LEN * 8 Byte
24bit data: ALNK_LEN / 2 * 2(CH) * 4(Byte) * 2(dual buffer) = ALNK_LEN * 8 Byte

当四条通道一起使用时，共需要4倍大小的buffer。

特别的：

1. 上述的ALNK_LEN在此处只能理解为16bit数据宽度下的样点数，当数宽度为24bit时，样点数为ALNK_LEN/2；

2. 在单声道状态下，只有左声道会被使用到，右声道将被停用。

16bit数据宽度时，数据组织如图1-1所示。

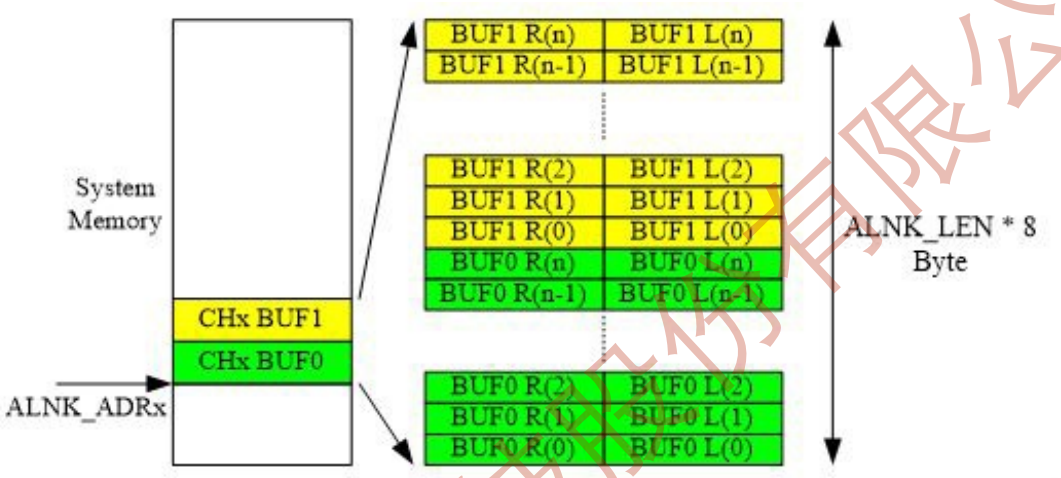


图 1-1 16bit数据宽度乒乓缓冲结构示意图

24bit数据宽度时，数据组织如图1-2所示。

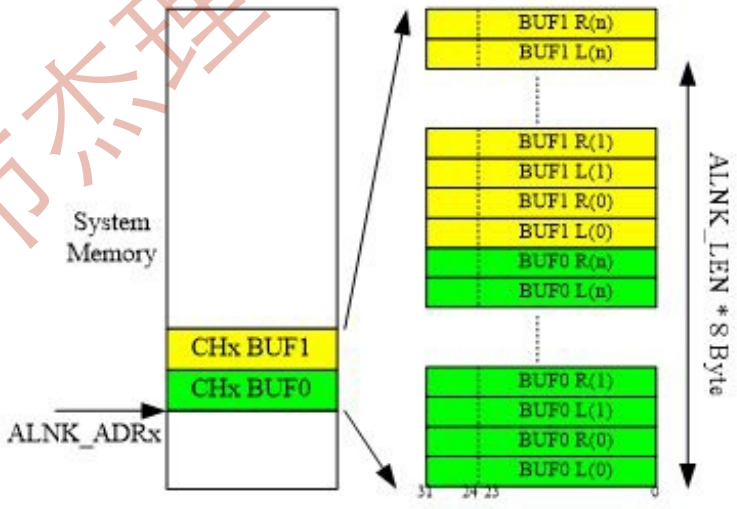


图 1-2 24bit数据宽度乒乓缓冲结构示意图