杰理 AD14N AC104N 芯片介绍 V1.2

珠海市杰理科技股份有限公司
Zhuhai Jieli Technologyco.,LTD

2021 年 08 月 16 日

目录

第 1 章 引言

1.1 编写目的

此说明书主要对杰理 AC140 芯片介绍。

AC140 是一颗具备较强运算能力的 CPU，内部支持 32M 字节 FLASH 的 cache 寻址；内部有 32K+4*4K 的 RAM 空间。

1.2 文档修改日志

2021 年 6 月 21 日 11:24:44

1、第十章和第十一章 timer 时钟源有错，已修改。

2、第十三章 PWM 时钟源说明有错，已修改。

第 2 章 cpu 介绍

2.1 说明

CPU 是一颗具备较强运算能力的 32 位处理器。CPU 有 11 个中断；有 8 级中断优先级。

2.2 cpu 内部 sfr

1) **icfg**: *icfg 是中断配置寄存器，包含一些*

Bit	名称	读/写	默认值	简介
31-10	-	-	-	-
9-8	GIE[1:0]	RW	0b10	总中断 = GIE[1] & GIE[0];同时打开才能进中断
7-0	-	-	-	-

2) **IDEL**: *CPU 进入 IDLE*
CPP 代码里面写：asm violate(“idle”);
ASM 代码里面写：idle;

2.3 中断说明

2.3.1 中断源

CPU 中断源可分为系统中断源和外设中断源。

中断号	中断类型	说明
31-4	外设中断源	优先级可配，外设中断入口，可扩展到 250 个
3	系统中断源	内核定时器 tick_timer
2-0	系统中断源	受 IE, IP 和 GIE 控制

2.3.2 中断控制寄存器 ICFGx

每个外设中断源都分配 4bit 中断控制位 ICFGx[3:0]，bit3-1 对应 IP，bit0 对应 IE。具体见中断表。

优先级 IP 有 3bit，共 8 级，0 代表最低，7 代表最高。进入中断后，硬件会自动屏蔽比其优先级低的中断入

口，例如当优先级为 1 的中断产生后，将被屏蔽优先级小于 1 或等于 1 的中断，直到中断退出。

中断使能 IE，只要将相应 IE 的控制位打开即可。：

2.3.3 总中断

CPU 增设了 GIE0 & GIE1。当 GIE0 于 GIE1 同时为 1 时，总中断才打开。

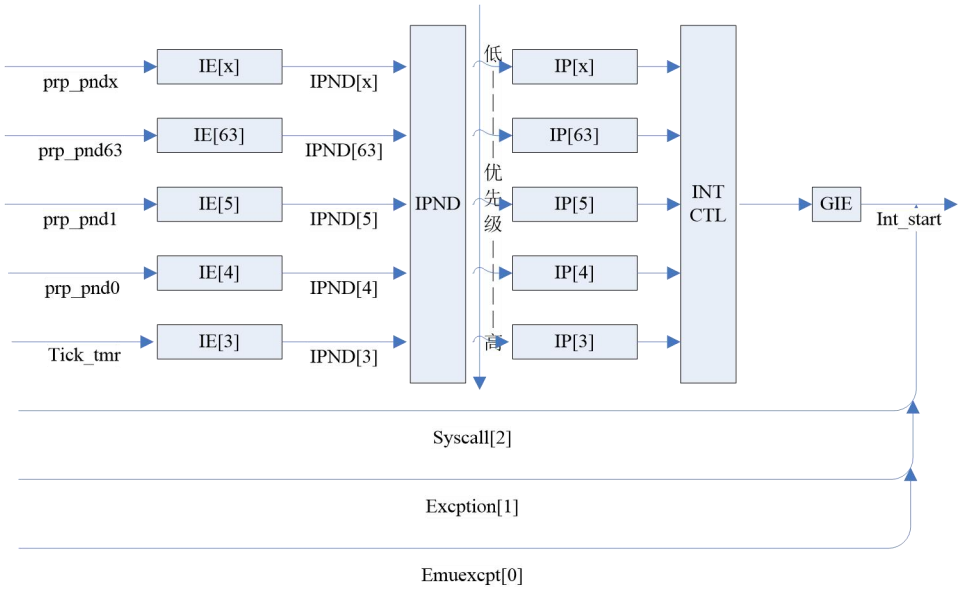


图 2.1 中断示意图

2.3.4 中断入口(中断列表，中断优先级)

中断入口从中断 BASE 地址，每 4 个 byte 对应一个中断，存放相应服务程序的地址。

中断发生时，CPU 会从相应中断入口取中断服务程序的入口地址，跳到该中断服务程序。

中断只会压 PC 入 RETI，中断返回时硬件从 RETI 取出返回地址。

中断号	{IP[2:0],IE}	中断入口地址	中断源
31	ICFG3[31:28]	BASE + 0x7c + 0x80*0	hsoft3 | bank_int
30	ICFG3[27:24]	BASE + 0x78 + 0x80*0	hsoft2
29	ICFG3[23:20]	BASE + 0x74 + 0x80*0	hsoft1
28	ICFG3[19:16]	BASE + 0x70 + 0x80*0	hsoft0
27	ICFG3[15:12]	BASE + 0x6c + 0x80*0	src_int
26	ICFG3[11:08]	BASE + 0x68 + 0x80*0	lrct_int
25	ICFG3[07:04]	BASE + 0x64 + 0x80*0	alnk_int
24	ICFG3[03:00]	BASE + 0x60 + 0x80*0	lptmr1
23	ICFG2[31:28]	BASE + 0x5c + 0x80*0	lptmr0
22	ICFG2[27:24]	BASE + 0x58 + 0x80*0	usb_ctl
21	ICFG2[23:20]	BASE + 0x54 + 0x80*0	usb_sof
20	ICFG2[19:16]	BASE + 0x50 + 0x80*0	mctmrx
19	ICFG2[15:12]	BASE + 0x4c + 0x80*0	
18	ICFG2[11:08]	BASE + 0x48 + 0x80*0	gpc
17	ICFG2[07:04]	BASE + 0x44 + 0x80*0	spi1
16	ICFG2[03:00]	BASE + 0x40 + 0x80*0	spi0
15	ICFG1[31:28]	BASE + 0x3c + 0x80*0	adc
14	ICFG1[27:24]	BASE + 0x38 + 0x80*0	iic
13	ICFG1[23:20]	BASE + 0x34 + 0x80*0	port
12	ICFG1[19:16]	BASE + 0x30 + 0x80*0	sd0
11	ICFG1[15:12]	BASE + 0x2c + 0x80*0	uart1
10	ICFG1[11:08]	BASE + 0x28 + 0x80*0	uart0
09	ICFG1[07:04]	BASE + 0x24 + 0x80*0	audio
08	ICFG1[03:00]	BASE + 0x20 + 0x80*0	P33
07	ICFG0[31:28]	BASE + 0x1c + 0x80*0	
06	ICFG0[27:24]	BASE + 0x18 + 0x80*0	timer2
05	ICFG0[23:20]	BASE + 0x14 + 0x80*0	timer1
04	ICFG0[19:16]	BASE + 0x10 + 0x80*0	timer0
03	ICFG0[15:12]	BASE + 0x0c + 0x80*0	tick_tmr

2.3.5 其他

1、清外设模块内部的中断即可清除外设中断源。

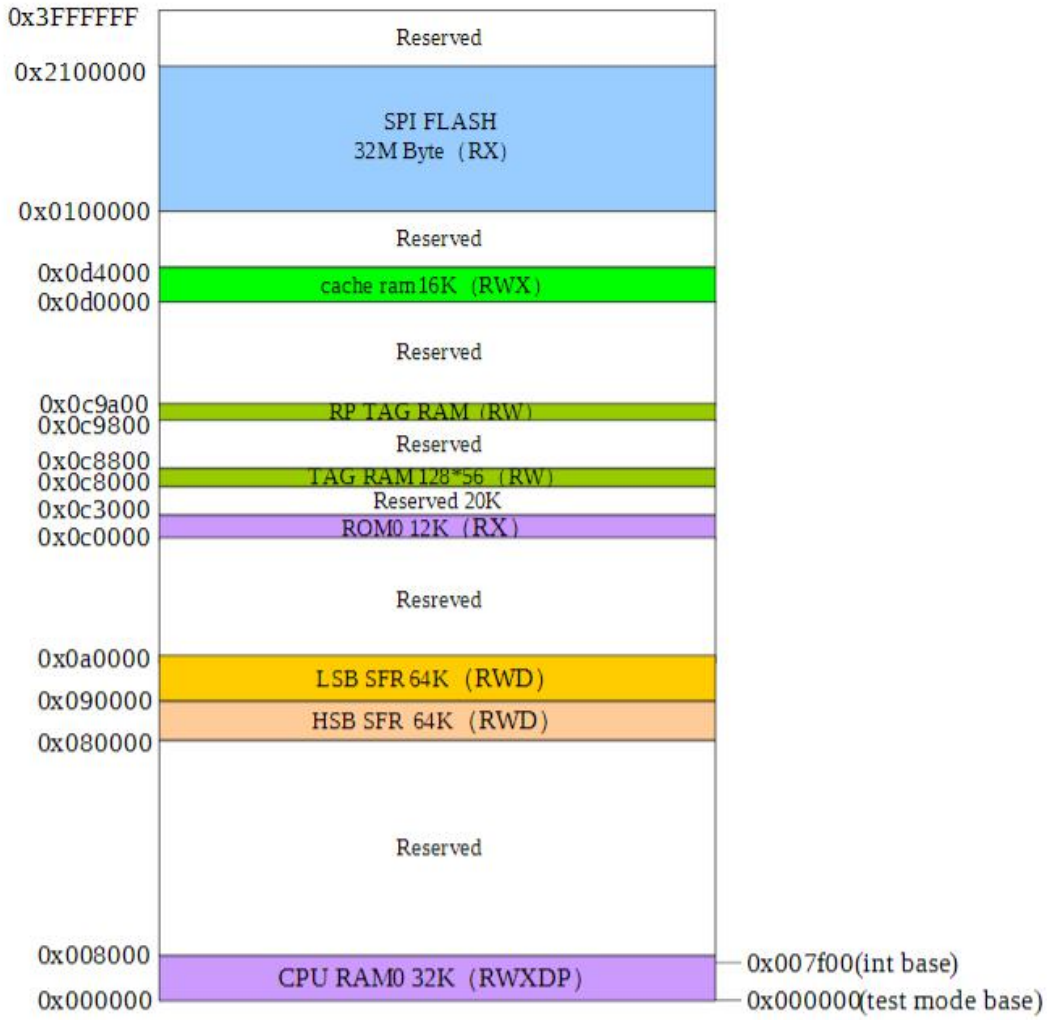
2、置软中断：写 ILAT_SET[7:0]或使用 asm(“swi #u3”)分别对应软中断 7-0

清软中断：写 ILAT_CLR[7:0]清相应软中断

2.4 MEMORY

Memory 主要有对 FLASH 的管理和内部 RAM 的管理：

从 0x0100000 开始映射内置 FLASH，一共有 32M 的地址空间；从 0~0x8000 是系统主要 RAM 的区域。



第 3 章 时钟系统

3.1 模块说明

1. SH54 具备如下 3 个原生时钟源，可直接驱动系统运行：
   a) rc_clk：来自 RC 振荡器，振荡频率约 16MHz，用于系统启动，随电压和温度不同有较大变化。
   b) htc_clk：来自 RC 振荡器，振荡频率约 5.5MHz，作为 PLL 的输入参考时钟。
   c) lrc_clk：来自 PMU 的特殊 RC 振荡器，振荡频率约 32KHz，随电压和温度变化较小。
2. SH54 还具备如下 2 个衍生时钟源：
   a) pll_clk：来自片内 SYS_PLL 的输出，同时输出 480MHz，320MHz，192MHz，137MHz，107MHz 的时钟，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防止额外电源消耗。
   b) dpll_clk：lrc_clk8 倍频而来的，32*8 kHz。
3. SH54 极限运行频率如下表

DVDD 电压	1.0V 电压档	1.1V 电压档	1.2V 电压档
sfc_clk	96MHz	120MHz	160MHz
hsb_clk	96MHz	120MHz	160MHz
lsb_clk	48MHz	60MHz	80MHz

3.2 系统时钟树

1. SYSPLL 只支持整数工作模式，整数模式支持 12M 或 200K 参考频率。总体而言，参考频率越高，PLL 的性能越好。
2. INTE 整数模式下

$$Fout = Fref / DIVn * DIVm$$

（n = PLL_CON[8:2]+2，m = PLL_CON1[11:0]+2）

3. SYSPLL 的输入参考时钟可由 pat_clk 提供。

4. SYSPLL 部分电路框图如下。

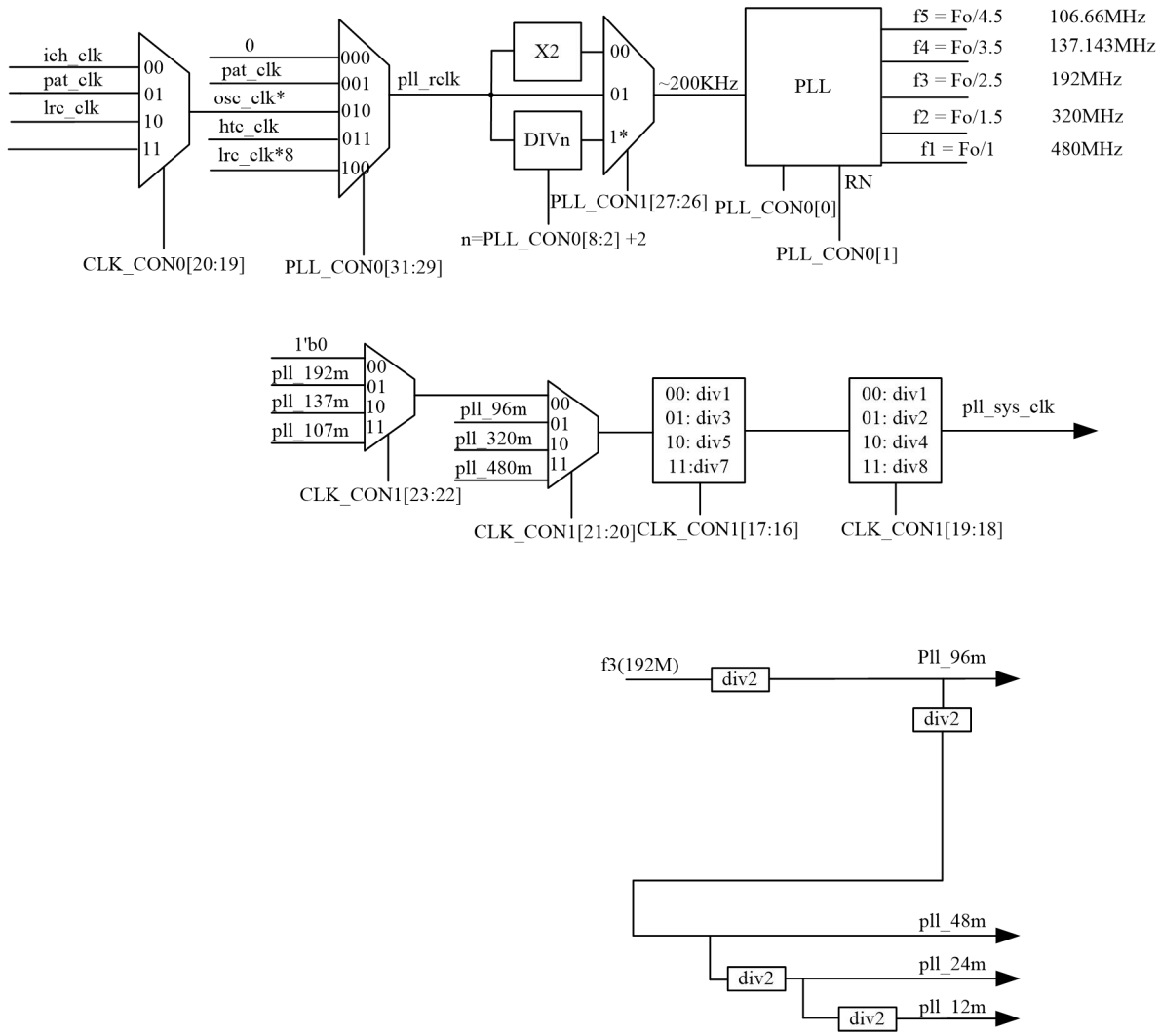


图 1 SYSPLL 部分电路框图

5. SH54 系统可运行于上述 2 个原生时钟源或 1 个衍生时钟源，可随时更改 sfc_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。系统时钟部分电路框图如下。

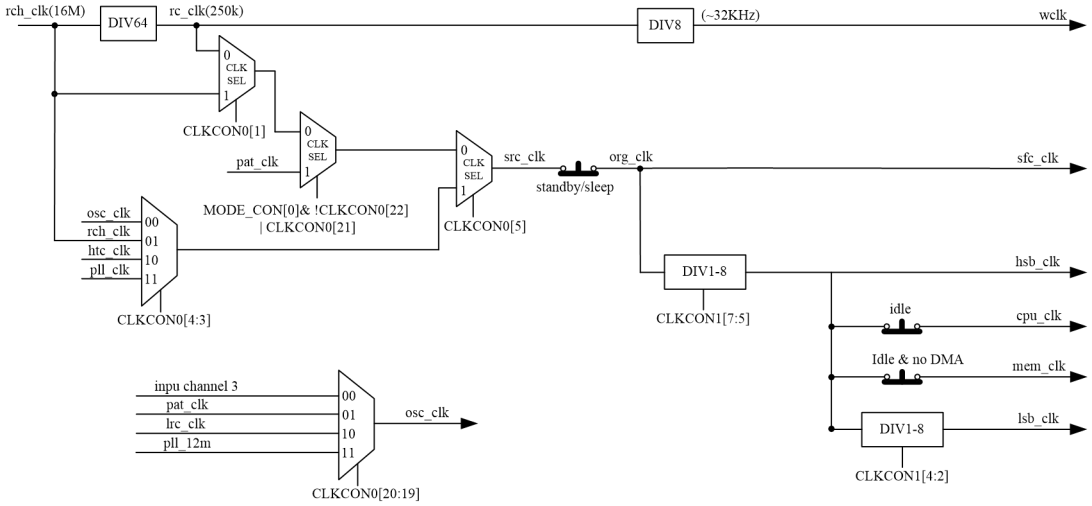


图 2 系统时钟部分电路框图

6. 与系统异步的部分外设具有单独的时钟切换电路，该部分框图如下。

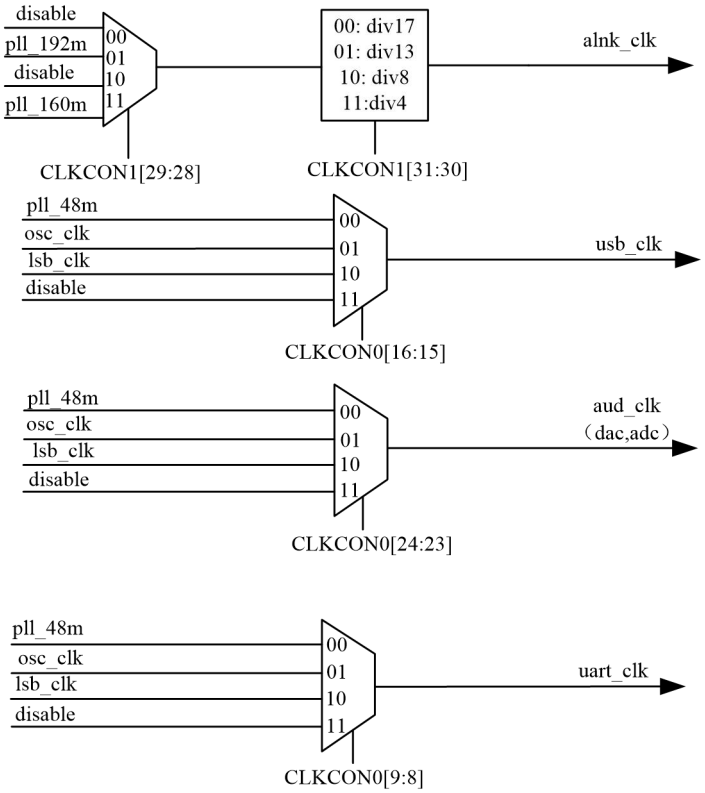


图 3 与系统异步部分外设时钟切换电路框图

7. SH54 的 PA3 引脚为复用测试引脚，该引脚除了普通的 IO 功能之外，还可以将内部时钟信号驱动至片外，用于测试或特殊用途。

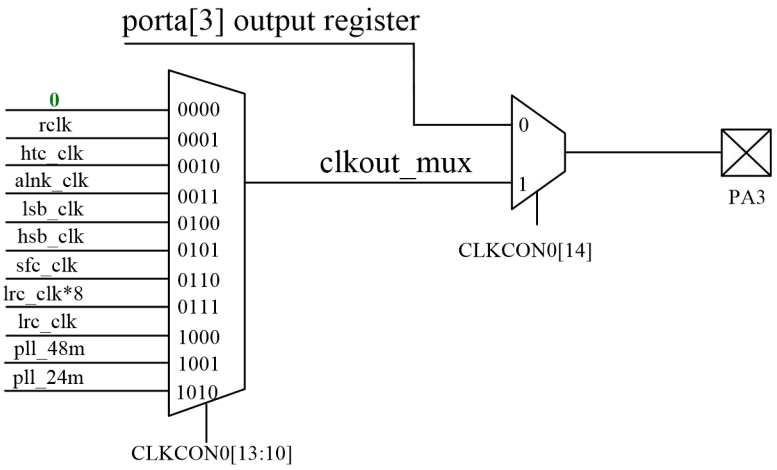


图 4 时钟输出到 IO 电路图

3.3 寄存器 SFR 列表

1．CLK_CON0: Clock control register0

Bit	Name	RW	Description
31~26	-	-	预留
25	SFR_CKMD	rw	SFR 时钟模式选择 0：不写 SFR 的时候，SFR 时钟自动关闭 1：使用 CPU 时钟作为 SFR 时钟
24~23	DAC_CKSEL	rw	DAC 时钟选择，见图 3
22~21	SRC_TSSEL	rw	系统时钟源选择，见图 2
20~19	OSC_SEL	rw	片内 OSC_CLK 来源选择，见图 2 00：input channel3 输入 01：pat_clk 10：lrc_clk 11：pll_12m
18	DPLL_CKSEL	rw	PLL 参考时钟源 pll_ref_32k 时钟选择 0：pll_32k_rclk 1：input channel3 输入
17	DPLL_UDEN	rw	DPLL 的倍频器开关
16~15	USB_CKSEL	rw	USB 时钟选择，见图 3
14~10	CLKOUT_SEL	rw	CLKOUT 时钟选择，见图 4
9~8	UART_CKSEL	rw	UART 时钟选择，见图 3
7~6	SFC_DL_SEL	rw	SFC 采样时钟延迟（只用于 IC 仿真）
5	SRC_CKSEL	rw	系统时钟源选择，见图 2

4~3	SRC_CKMUX	rw	系统时钟源选择，见图 2
2	RING_EN	rw	片内 RINGOSC 振荡器使能 0：关闭 1：使能
1	RCH_EN	rw	片内 RC 振荡器频率选择 0：250k； 1：16M；
0	RC_EN	rw	片内 RC（16M、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；

2．CLK_CON1: Clock control register1

Bit	Name	RW	Description
31~30	PLL_ALNK_DIV	rw	AUDIO LINK 时钟分频，见图 3
29~28	PLL_ALNK_SEL	rw	AUDIO LINK 时钟选择，见图 3
27~26	-	rw	预留
25~24	-	rw	预留
23~20	PLL_SYS_SEL	rw	系统 PLL 选择，见图 1
19~16	PLL_SYS_DIV	rw	系统 PLL 分频选择，见图 1
15~8	-	rw	预留
7~5	HSB_DX_SEL	rw	HSB_CLK 分频，见图 2
4~2	LSB_DX_SEL	rw	LSB_CLK 分频，见图 2
1~0	-	rw	预留

3．PLL_CON0: pll control register 0

Bit	Name	RW	Description
31~29	PLL_RSEL	rw	PLL REF CLK 选择，见图 1
28~26	-	rw	预留
25	PLL_CKDAC_OE	rw	PLL 模拟控制位
24	CK107M_OE	rw	PLL 输出端 107MHz 时钟使能
23	CK137M_OE	rw	PLL 输出端 137MHz 时钟使能
22	CK192M_OE	rw	PLL 输出端 192MHz 时钟使能
21	CK320M_OE	rw	PLL 输出端 320MHz 时钟使能
20	CK480M_OE	rw	PLL 输出端 480MHz 时钟使能
19~17	PLL_LPFR2	rw	PLL 模拟控制位
16~14	PLL_ICP	rw	PLL 模拟控制位

13~12	PLL_PFD	rw	PLL 模拟控制位
11	PLL_MODE	rw	PLL 模拟控制位
10	PLL_TSCK480M_OE	rw	PLL480MHz 时钟输出使能
9	-	-	预留
8~2	PLL_REFDS	rw	PLL 整数模式下，参考时钟分频值，R = PLL_REFDS + 2 （2-129）
1	PLL RST	rw	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	PLL 模块使能 0：关闭； 1：打开；

4．PLL_CON1: pll control register 1

Bit	Name	RW	Description
30~29	-	rw	-
29~28	PLL_REFMOD	rw	PLL 模拟控制位
27~26	PLL_ REFDSEN	rw	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）
25~23	PLL_LDO12D	rw	PLL 模拟控制位
22~20	PLL_LDO12A	rw	PLL 模拟控制位
19	-	rw	-
18	PLL_TSOE	rw	PLL 测试使能，需设置为‘0’
17~16	PLL_TSSEL	rw	PLL 测试选择，需设置为‘00’
15	PLL_LDOBYPAS	rw	PLL 模拟控制位
14~12	PLL_IVCO	rw	PLL 模拟控制位
11~0	PLL_DS	rw	PLL 参考时钟反馈分频(相当倍频)，m= PLL_DS + 2。见图 1

5．CLK_GAT0: clock gate register 0

Bit	Name	RW	Description
31~8	-	-	预留
7	CLK_GAT7	rw	LSB_CLK 输出到 TIMER0 使能
6	CLK_GAT6	rw	LSB_CLK 输出到 IRDA/WMA 使能
5	CLK_GAT5	rw	LSB_CLK 输出到 TIMER2 使能

4	CLK_GAT4	rw	LSB_CLK 输出到 TIMER3/ADC 使能
3	CLK_GAT3	rw	LSB_CLK 输出到 PWM4 使能
2	CLK_GAT2	rw	LSB_CLK 输出到 SPI0 使能
1	CLK_GAT1	rw	LSB_CLK 输出到 CRC/ENCRYPT 使能
0	CLK_GAT0	rw	LSB_CLK 输出到 MODE_DET/ISP/UART 使能

第 4 章 循环冗余校验(CRC16)

4.1 模块说明

CRC(Cyclic Redundancy Check，循环冗余校验)主要用于数据的校验，每次运算 8bits，多项式为： $X^{16}+X^{12}+X^{5}+X^{1}$

4.2 寄存器 SFR 列表

1. CRC_REG: CRC register

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

JL_CRC -> REG: CRC16 校验码

2. CRC_FIFO: CRC FIFO register

Bit	Name	RW	Description
31:8	-	r	预留
7:0	CRC_FIFO	w	运算数据输入，HSB 先运算，LSB 后运算

第 5 章 看门狗

5.1 模块说明

WDT(watch dog timer)看门狗定时器用于防止系统软件进入死循环等不正确的状态。它设定了一个时间间隔，软件必须每在此时间间隔内进行进行一次“清看门狗”的操作，否则看门狗将溢出，并导致系统复位（或引发中断，主要用于程序的调试）。

每次系统复位之后，看门狗默认处于关闭的状态。软件可以随时将其打开。当发生系统复位时看门狗也会被关闭。

WDT 设计在 P33 系统里，读写 WDT_CON 需要用 P33 接口，写 CON 前无需再写 CRC_REG 打 key，并且支持低功耗模式下运行。

低功耗模式下，WDT 支持：

1. 直接复位芯片
2. 唤醒芯片，进入异常
3. 唤醒芯片，进入 RTC 中断（需关闭看门狗异常使能）

5.2 寄存器 SFR 列表

1. P3_WDT_CON: Watchdog control register(8bit addressing)

Bit	Name	RW	Description
7	PND	r	wdt 中断请求标志，当 WDRMD 设置为 1 时，WDT 溢出硬件会将此位置 1，Reset 值为 0 0：without pending 1：with pending
6	CPND	w	Clear pending： 0：invalid 1：clear pending
5	WDRMD	rw	看门狗模式选择： 0：看门狗溢出将导致系统复位，这是看门狗的主要工作模式 1：看门狗溢出将 WINT 置 1，可产生中断或异常，这种模式主要用于调试
4	WDTEN	rw	看门狗定时器使能。 0：看门狗定时器关闭 1：看门狗定时器打开
3~0	TSEL3-0	rw	看门狗溢出时间选择 0000：1mS 0001：2mS

			0010：4mS 0011：8mS 0100：16mS 0101：32mS 0110：64mS 0111：128mS 1000：256mS 1001：512mS 1010：1S 1011：2S 1100：4S 1101：8S 1110：16S 1111：32S Note：上述溢出时间只是参考值。实际上，wdt 由不准确的片内 RC 振荡器驱动，其实际溢出时间可能会有高达 100%的偏差，且不同芯片之间也无法保证一致性。所以在选择溢出时间时必须留有足够余量

2. P3_VLD_KEEP: wdt exception register (8bit addressing)

Bit	Name	RW	Description
7	-	-	其它功能
6	WDT EXPT EN	rw	看门狗异常使能 0: 看门狗异常关闭 1: 看门狗异常打开
5~0	-	-	其它功能

第 6 章 IIC 模块

6.1 模块说明

IIC 通讯模块。该芯片只有一个 IIC 模块。IIC 有 4 组 IO：(请参考另外一章：IO_MAPPING_CONTROL)

6.2 寄存器 SFR 列表

1. IIC_CON: IIC configuration register(16bit addressing)

Bit	Name	RW	Description
15	PND	r	The IIC pending：普通 pending。不论主机还是从机，收或发完一个 Byte，硬件都会 Pending 置 1，Reset 值为 0 0：without pending 1：with pending
14	End_PND	r	结束位 Pending，在做从机时，当收到结束位，硬件会把该位 pending 置 1，Reset 值为 0 0：without pending 1：with pending
13	CLR_PND	w	Clear pending： 0：invalid 1：clear pending
12	CLR_End_PND	w	Clear End pending： 0：invalid 1：clear pending
11	Receive ACK	r	接收到的 ACK 或 NACK 0：接收到应答信号 1：未接收到应答信号
10	Output ACK	rw	发送 ACK 控制位 0：发送应答信号 1：不发送应答信号
9	Start_PND	r	起始位 Pending，在做从机时，当收到起始位，硬件会把该位 pending 置 1，Reset 值为 0（不清除无法检测到下一次起始位） 0：without pending 1：with pending
8	CLR_Start_PND	w	Clear Start pending: 0：invalid

			1：clear pending
7	IIC_IE	rw	IIC 中断允许 0：disable 1：enable
6	END_IE	rw	结束位中断允许 0：disable 1：enable
5~4	Reserved	rw	预留
3	Sending end bit	rw	加结束位。只有在主机模式才有效，是否发送结束位 0：Do not send the end bit 1：Sending end bit
2	Sending start bit	rw	加起始位。只有在主机模式才有效，是否发送起始位 0：Do not send the start bit 1：Sending start bit
1	Slave Mode	rw	IIC 主从模式选择 0：master mode 1：slave mode
0	IICEN	rw	IIC 使能位 0：disable 1：enable

2. IIC_BAUD: IIC baudrate register (16bit addressing, Write Only)

Bit	Name	RW	Description
7~0	IIC_BAUD	w	iic 的波特率寄存器

波特率寄存器，Reset 值为 x，用之前需要初始化。

波特率计算公式：波特率 = Freq_sys / ((IICBAUD+1) *4)

IICBAUD 寄存器在做从机时则作为从机的地址寄存器，高 7 位为 7 位地址，最低 1 位为从机自动响应允许，高有效。自动响应是指检测到起始位后自动回 Ack（低电平）。

3. IIC_BUF: IIC data buffer register (8bit addressing)

Bit	Name	RW	Description
7~0	IIC_BUF	rw	iic 的收发数据寄存器： 写 IIC_BUF 可启动一次发送； 读 UTx_BUF（先写 0xFF）可获得已接收到的数据。

做主机时：

发送：写 IICBUF 会启动一次通信，当普通 pending 出现，则表示发送完毕。此时可检查确认位（ACK， IICCON[11]），以了解从机是否接收到。若在启动通信前选择了“加结束位”，则发送完该字节后会顺便发送结束位。若在启动通信前选择了“加起始位”，则发送该字节前会先发送起始位。若启动通信前时钟和数据线处于空闲状态（上拉），则无论有无

选择“加起始位”，起始位都会自动加上。

接收：则需要往 IICBUF 写 0xFF，等到普通 Pending 出现，则表示接收完毕，CPU 可以读 IICBUF 得到接收的值。若在写 0xFF 前选择了“加结束”，则接收完会顺便发送结束位。

做从机时：

写 IICBUF 会启动一次等待通信，若要发送，则应写入要发送的值，若要接收，则应写入 0xFF。当普通 Pending 出现时，表示通信结束，若刚刚这次通信是发送，则 CPU 应检查确认位，以了解从机是否接收到。若刚刚这次通信是接收，则 CPU 可以通过读 IICBUF，得到接收的内容。

每次通信结束时，CPU 可以通过检查起始位标志，以了解刚刚完成的这次通信，有没有起始位。

做从机时，若出现结束位 Pending，则表示结束位出现。做从机时，IIC 时钟由主机提供，系统时钟频率（IIC 模块的工作时钟）必须是 IIC 外设接口时钟频率的 16 倍以上。

第 7 章 SPI 模块

7.1 模块说明

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。该芯片有 2 个 SPI：SPI0、SPI1

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。

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：串行数据通过一根 DAT 线传输，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 1-65535Byte。写 SPI_CNT 的动作将启动一次 DMA

传输。

7.2 寄存器 SFR 列表

1. SPIx_CON: SPIx control register (16bit addressing)

Bit	Name	RW	Description
15	PND	r	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。 0：without pending 1：with pending 有 3 种方法清除此标志： 1、 向 PCLR 写入‘1’ 2、 写 SPIBUF 寄存器来启动一次传输 3、 写 SPICNT 寄存器来启动一次 DMA
14	PCLR	w	清除 PND 中断请求标志 0：invalid 1：clear pending
13	IE	rw	SPI 中断使能 0：禁止 SPI 中断 1：允许中断
12	DIR	rw	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	QUAD	rw	4BIT 模式（**SPI1** 只有 **1bit** 和 **2bit** 模式） 0：关闭 4BIT 模式 1：开启 4BIT 模式
10	DUAL	rw	2BIT 模式 0：关闭 2BIT 模式 1：开启 2BIT 模式
9~8	reserved	r	预留
7	CSID	rw	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	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	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	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	3wire	rw	单/双向模式 0：2 线模式（单向） 1：3 线模式（双向）
2	CSEN	rw	SPICS 信号使能 0：不使用 SPICS 信号 1：使用 SPICS 信号
1	SLAVE	rw	SPI 主从模式选择 0：master mode 1：slave mode
0	SPIEN	rw	SPI 接口使能 0：关闭 SPI 接口 1：打开 SPI 接口

2. SPI x _BAUD: SPI baud rate setting register (8bit addressing, write only)

Bit	Name	RW	Description
7~0	SPI x _BAUD	wo	SPI 主机时钟设置寄存器

SPI 主机时钟设置寄存器：

SPICK = system clock / (SPIBAUD + 1)

3. SPIx_BUF: SPI buffer register (8bit addressing)

Bit	Name	RW	Description
7~0	SPIx_BUF	rw	SPI 的收发数据寄存器

发送寄存器和接收寄存器共用此 SFR 地址。写入数据至寄存器则发送，从寄存器读出接收数据。

4. SPIx_ADR: SPI DMA start address register (16bit addressing, write only)

Bit	Name	RW	Description
15~0	SPIx_ADR	wo	SPI DMA 起始地址寄存器

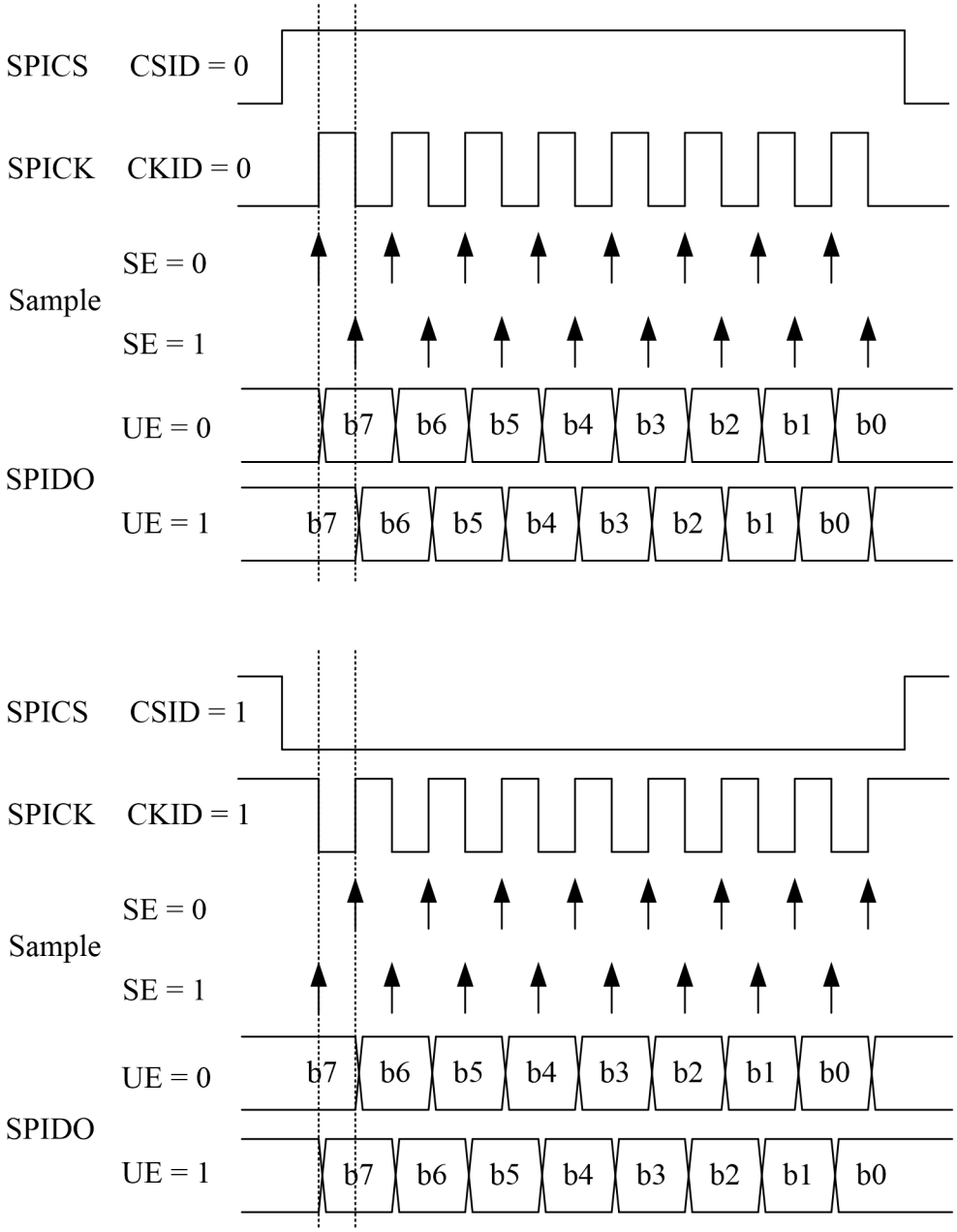
SPI DMA 起始地址寄存器，只写，读出为不确定值。

5. SPIx_CNT: SPI DMA counter register (16bit addressing, write only)

Bit	Name	RW	Description
15~0	SPIx_CNT	wo	SPI DMA 计数寄存器

SPI DMA 计数寄存器，只写，读出为不确定值。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。

传输波形图：


SPICS CSID = 0
SPICK CKID = 0
Sample
SE = 0
SE = 1
SPIDO
UE = 0
UE = 1
b7 b6 b5 b4 b3 b2 b1 b0
SPICS CSID = 1
SPICK CKID = 1
Sample
SE = 0
SE = 1
SPIDO
UE = 0
UE = 1
b7 b6 b5 b4 b3 b2 b1 b0

第 8 章 数模转换器(ADC)

8.1 模块说明

10Bit ADC(A/D 转换器)，其时钟最大不可超过 1MHz。模块输出精度：10Bit。

8.2 寄存器 SFR 列表

1. ADC_CON: ADC configuration register

Bit	Name	RW	Description
31~17	reserved	r	预留
16	adc_isel	rw	ADC CMP 功率选择，写‘1’选择低功率
15~12	WAIT_TIME	rw	WAIT_TIME： 启动延时控制，实际启动延时为此数值乘 8 个 ADC 时钟
11~8	CH_SEL	rw	CH_SEL： 通道选择 0000：选择 PA0 0001：选择 PA1 0010：选择 PA2 0011：选择 PA3 0100：选择 USBDP 0101：选择 USBDM 0110：选择 PA4 0111：选择 PA5 1000：选择 PA10 1001：选择 PA11 1010：选择 PA13 1011：选择 PA14 1100：选择 PA15 1101：选择 PB0 1110：选择 ANA_TEST 1111：选择 P33_TEST
7	PND	r	PND：中断请求位（只读），当 ADC 完成一次转换后，此位会被设置为‘1’，需由软件清‘0’ 0：without pending 1：with pending
6	CPND	w	CPND：清除中断请求位（只写），写‘1’清除，写‘0’无效 0：invalid

			1：clear pending
5	ADC_IE	rw	ADC_IE：ADC 中断允许 0：禁止 ADC 中断 1：允许中断
4	ADC_EN	rw	ADC_EN：ADC 控制器使能 0：关闭 ADC 控制器 1：打开 ADC 控制器
3	ADC_AE	rw	ADC_AE：ADC 模拟模块使能 0：关闭 ADC 模拟模块 1：打开 ADC 模拟模块
2~0	ADC_BAUD	rw	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

2. ADC_RES: ADC result 32-bit register

Bit	Name	RW	Description
31~10	Reserved	r	预留
9~0	ADC_RES	ro	ADC 输出结果

第 9 章 时钟脉冲计数器(GPCNT)

9.1 模块说明

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。

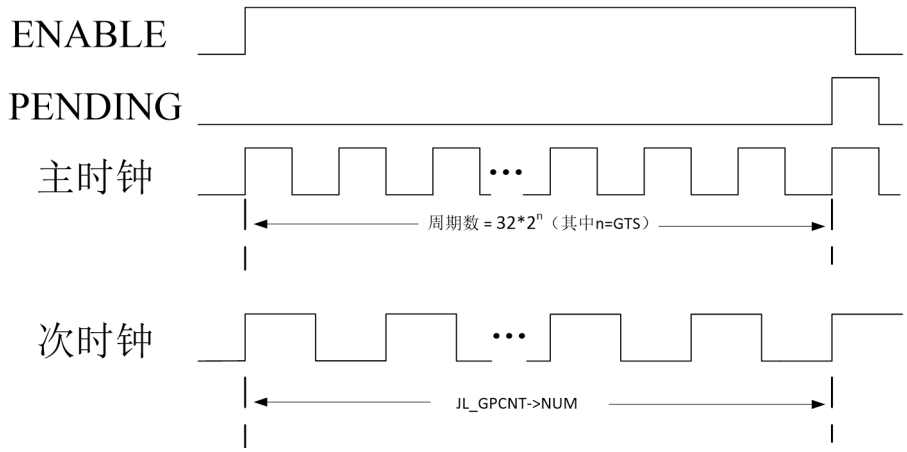


图 1 GPCNT 时钟计算示意图

9.2 寄存器 SFR 列表

1. JL_GPCNT->CON: GPCNT configuration register

Bit	Name	RW	Description
31:19	-	r	预留
18:16	GPC_CKSEL	rw	主时钟选择（当 JL_GPCNT->CON[14:12]==100 时）： 000: 无 001: src_clk 010: sfc_clk 011: sys_clk 100: lsb_clk 101: dac_clk 110: rch_clk 111: usb_clk
15	-	r	预留
14:12	GSS	rw	主时钟选择： 000: lsb_clk 001: osc_clk

			010: input channel0（见 IOMAP_CON2[4:0]） 011: lrc_clk 100: 时钟系统输入（见 JL_GPCNT->CON[18:16]） 101: ring_osc 110: pll_max 111: input channel1（见 IOMAP_CON2[9:5]）
11:8	GTS	rw	主时钟周期数选择： 主时钟周期数 = 32*2n （其中 n=GTS）
7	PND	r	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
6	CLR_PND	w	清除中断请求标志位： 0: 无效 1: 清 PENDING
5:4	-	r	预留
3:1	CSS	rw	次时钟选择： 000: lsb_clk 001: osc_clk 010: input channel0（见 IOMAP_CON2[4:0]） 011: lrc_clk 100: 时钟系统输入（见 JL_GPCNT->CON[18:16]） 101: ring_osc 110: pll_max 111: input channel1（见 IOMAP_CON2[9:5]）
0	ENABLE	rw	GPCNT 模块使能位： 0: 不使能 1: 使能

（注：需配置好其他位，才将 ENABLE 置 1。）

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

Bit	Name	RW	Description
31:0	NUM	r	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。

第 10 章 16 位定时器(Timer0/Timer1)

10.1 模块说明

Timer0/1 是一个集合了定时/计数/捕获功能于一体的多动能 16 位定时器。它的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。它具有上升沿/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。

10.2 寄存器 SFR 列表

1. JL_TMRx->CON: timer x configuration register

Bit	Name	RW	Description
15:8	-	r	预留
7	PND	r	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。
6	PCLR	w	软件在此位写入‘1’将清除 PND 中断请求标志。
5:4	PSET[1:0]	rw	PSET1-0：预分频选择位 00：预分频 1 01：预分频 4 10：预分频 16 11：预分频 64
3:2	SSEL[1:0]	rw	SSEL1-0：timerx 驱动源选择 00: 使用 lsb 时钟作为 timerx 的驱动源 Timer0: 01：使用 IMOC0[13]选择 IO 口 PA8 或 PLL_24M 信号作为 timer0 的驱动源； 10: 使用 RCH_CLK 作为 timer0 的驱动源 Timer1: 01：使用 IMOC0[14]选择 IO 口 PA10 或 PLL_24M 信号作为 timer1 的驱动源； 10: 使用 OSC 时钟作为 timer1 的驱动源 11: 使用 HTC 时钟作为 timerx 的驱动源
1:0	MODE[1:0]	rw	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 CNT 的值捕捉到 TMRxPRD 中）； 11： IO 口下降沿捕获模式（当 IO 下降沿到来时，把 CNT 的值捕捉到 TMRxPRD 中）。

2. JL_TMRx->CNT: timer x counter register

Bit	Name	RW	Description
15:0	CNT	rw	Timerx 的计数寄存器。

Timerx 的计数寄存器

3. JL_TMRx->PRD: timer x period register

Bit	Name	RW	Description
15:0	PRD	rw	Timerx 的计数周期寄存器。

在定时/计数模式下，当 TMRx_CNT == TMRx_PRD 时，TMRx_CNT 会被清 0。

在上升沿/下降沿捕获模式下，TMRx_PRD 是作为捕获寄存器使用的，当捕获发生时，TMRx_CNT 的值会被复制到 TMRx_PRD 中。而此时 TMRx_CNT 自由的由 0-65535-0 计数，不会和 TMRx_PRD 进行比较清 0。

第 11 章 16 位定时器(Timer2)

11.1 模块说明

Timer2 是一个集合了定时/计数/捕获/PWM 功能于一体的多动能 16 位定时器。它的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。它还具有上升沿/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,以及两个 PWM 输出。

11.2 寄存器 SFR 列表

1. JL_TMR2->CON: timer 2 configuration register

Bit	Name	RW	Description
15:14	-	r	预留
13	PWM1_Inv	rw	PWM1_INV： PWM1 信号输出反向。
12	PWM1_En	rw	PWM1_EN： PWM1 信号输出使能。此位置 1 后，相应 IO 口的功能将会被 PWM1 信号输出替代。
11:10	-	r	预留
9	PWM0_Inv	rw	PWM0_INV： PWM0 信号输出反向。
8	PWM0_En	rw	PWM0_EN： PWM0 信号输出使能。此位置 1 后，相应 IO 口的功能将会被 PWM0 信号输出替代。
7	PND	r	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。
6	PCLR	w	软件在此位写入‘1’将清除 PND 中断请求标志。
5:4	PSET[1:0]	rw	PSET1-0：预分频选择位 00：预分频 1 01：预分频 4 10：预分频 16 11：预分频 64
3:2	SSEL[1:0]	rw	SSEL1-0：timer 驱动源选择 00：使用 lsb 时钟作为 timer 的驱动源； 01：使用 IMOC0[15]选择 IO 口 PA4 或 ICH3 信号作为 timer 的驱动源； 10：使用 OSC 时钟作为 timer 的驱动源； 11：使用 HTC 时钟作为 timer 的驱动源。
1:0	MODE[1:0]	rw	MODE1-0：工作模式选择

			00： timer 关闭； 01： 定时/计数/PWM 模式； 10： IO 口上升沿捕获模式（当 IO 上升沿到来时，把 CNT 的值捕捉到 TMR2_PRD 中）； 11： IO 口下降沿捕获模式（当 IO 下降沿到来时，把 CNT 的值捕捉到 TMR2_PRD 中）。

2. JL_TMR2->CNT: timer 2 counter register

Bit	Name	RW	Description
15:0	CNT	rw	Timer2 的计数寄存器

3. JL_TMR2->PRD: timer 2 period register

Bit	Name	RW	Description
15:0	PRD	rw	Timer2 的计数周期寄存器

在定时/计数模式下，当 TMR2_CNT == TMR2_PRD 时，TMR2_CNT 会被清 0。

在上升沿/下降沿捕获模式下，TMR2_PRD 是作为捕获寄存器使用的，当捕获发生时，TMR2_CNT 的值会被复制到 TMR2_PRD 中。而此时 TMR2_CNT 自由的由 0-65535-0 计数，不会和 TMR2_PRD 进行比较清 0。

4. JL_TMR2->PWM0/1: timer 0/1 PWM register

Bit	Name	RW	Description
15:0	TMR2_PWM0/1	rw	Timer2 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 = (TMR2_PWM / TMR2_PRD) * 100%

此寄存器不带有缓冲，写此寄存器的动作将可能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。

第 12 章 红外滤波模块(IRFLT)

12.1 模块说明

IRFLT 是一个专用的硬件模块，用于去除掉红外接收头信号上的窄脉冲信号，提升红外接收解码的质量。

IRFLT 使用一个固定的时基对红外信号进行采样，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’。换言之，脉宽小于 4 倍时基的窄脉冲将被滤除。改变该时基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红外信号的过滤效果。

通过对 IOMC（IO re-mapping）寄存器的配置，可以将 IRFLT 插入到系统 2 个 timer 中某一个的捕获引脚之前。例如通过 IOMC 寄存器选择了 IRFLT 对 timer1 有效，并且 IRFLT_EN 被使能之后，则 IO 口的信号会先经过 IRFLT 进行滤波，然后再送至 timer1 中进行边沿捕获。

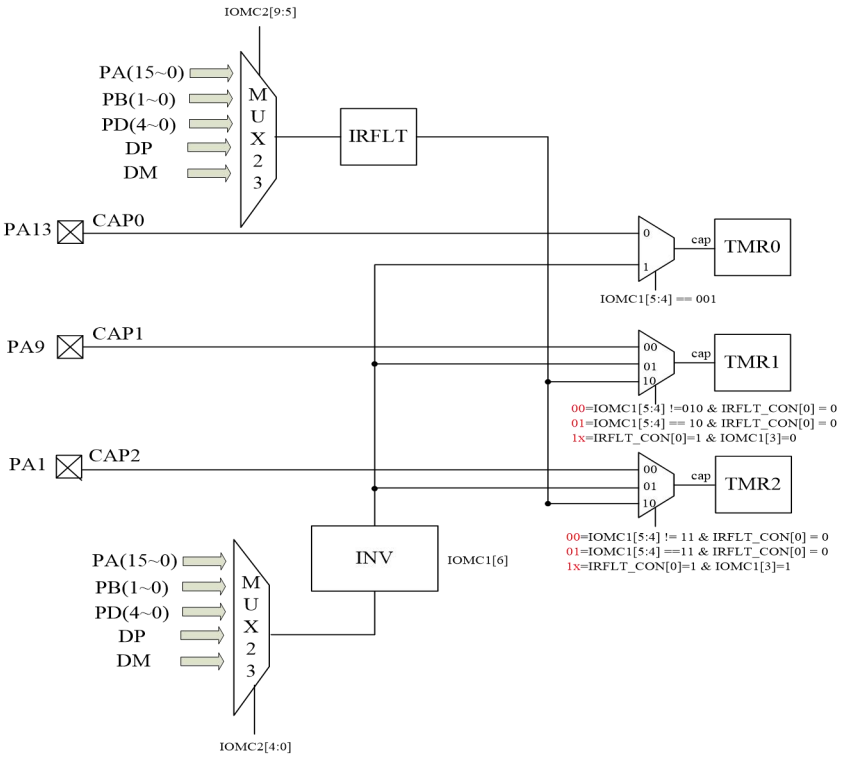


图 1 IRFLT 模块示意图

12.2 寄存器 SFR 列表

1. JL_IRFLT->CON: irda filter configuration register

Bit	Name	RW	Description
7:4	PSEL	rw	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	时基发生器驱动源选择 00: 选择系统时钟来驱动时基发生器 01: 选择 OSC 时钟来驱动时基发生器 1x: 选择 HTC 时钟来驱动时基发生器
1	-	r	预留
0	IRFLT_EN	rw	IRFLT 使能 0: 关闭 IRFLT 1: 打开 IRFLT

12.3 时基选择

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 用于采样红外接收信号的时基 Ts

$$Ts = Tc * N$$

例如，当选择 32KHz 的 OSC 时钟，并且分频倍数为 1 时，Ts = 30.5uS。根据 IRFLT 的工作规则，所有小于(30.5*4=122uS)的窄脉冲信号，均会被滤除。

又如，当选择 48MHz 的系统时钟，并且分频倍数为 1024 时，Ts = 21.3uS。根据 IRFLT 的工作规则，所有小于(21.3*4=85uS)的窄脉冲信号，均会被滤除。

第 13 章 PWM

13.1 模块说明

PWM 功能块包括：

4 个独立的 PWM _TIMER 模块；

4 个独立的 PWM 通道；

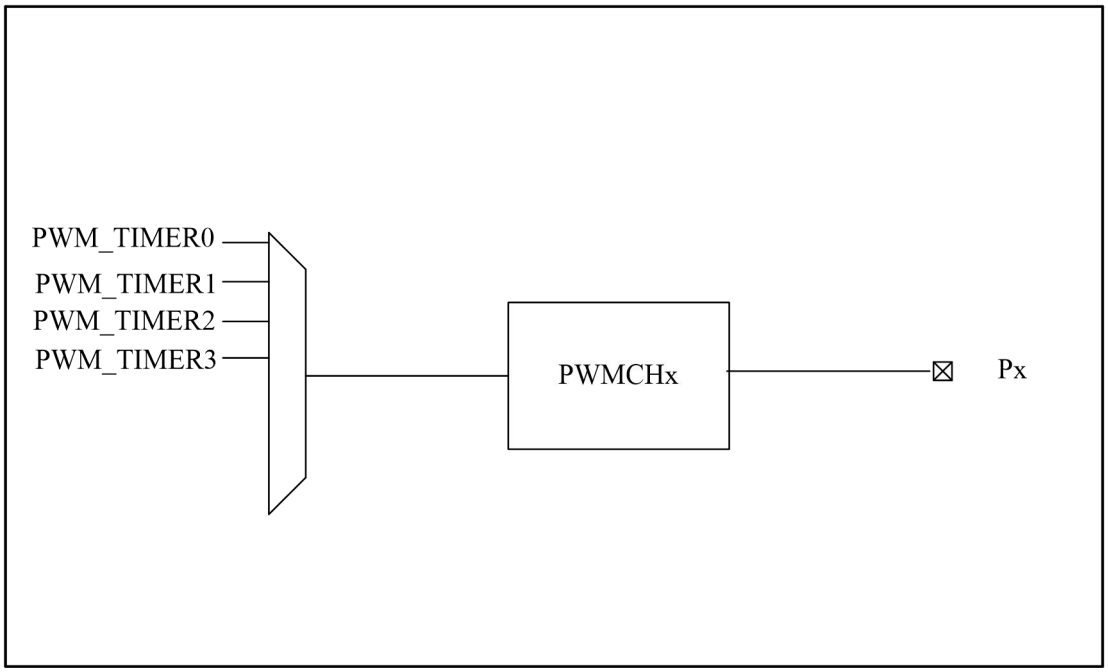


图 1 PWM 模块示意图

13.2 寄存器 SFR 列表

1. JL_PWM->PWMCON0: pwm configuration register0

Bit	Name	RW	Description
15:12	-	r	预留
11	T3EN	rw	T3EN：定时器 3 计数开关控制 0：定时器 3 计数关闭 1：定时器 3 计数开启
10	T2EN	rw	T2EN：定时器 2 计数开关控制 0：定时器 2 计数关闭 1：定时器 2 计数开启

9	T1EN	rw	T1EN：定时器 1 计数开关控制 0：定时器 1 计数关闭 1：定时器 1 计数开启
8	T0EN	rw	T0EN：定时器 0 计数开关控制 0：定时器 0 计数关闭 1：定时器 0 计数开启
7:4	-	r	预留
3	PWM3EN	rw	PWM3EN：PWM3 模块开关控制 0：模块关闭 1：模块开启
2	PWM2EN	rw	PWM2EN：PWM2 模块开关控制 0：模块关闭 1：模块开启
1	PWM1EN	rw	PWM1EN：PWM1 模块开关控制 0：模块关闭 1：模块开启
0	PWM0EN	rw	PWM0EN：PWM0 模块开关控制 0：模块关闭 1：模块开启

2. JL_PWM->PWMCON1: pwm configuration register1

Bit	Name	RW	Description
31:16	-	r	预留
15:13	PWM3_TMRxSEL	rw	PWM3_TMRxSEL：选择 PWM_TMRx 作为 PWM3 时基 000：选择 PWM_TIMER0 001：选择 PWM_TIMER1 010：选择 PWM_TIMER2 011：选择 PWM_TIMER3
12	PWM3_INV	rw	PWM3_INV：PWM3 信号输出反向控制 0：正向 1：反向
11:9	PWM2_TMRxSEL	rw	PWM2_TMRxSEL：选择 PWM_TMRx 作为 PWM2 时基 000：选择 PWM_TIMER0 001：选择 PWM_TIMER1 010：选择 PWM_TIMER2 011：选择 PWM_TIMER3
8	PWM2_INV	rw	PWM2_INV：PWM2 信号输出反向控制 0：正向 1：反向
7:5	PWM1_TMRxSEL	rw	PWM1_TMRxSEL：选择 PWM_TMRx 作为 PWM1 时基

			000：选择 PWM_TIMER0 001：选择 PWM_TIMER1 010：选择 PWM_TIMER2 011：选择 PWM_TIMER3
4	PWM1_INV	rw	PWM1_INV：PWM1 信号输出反向控制 0：正向 1：反向
3:1	PWM0_TMRxSEL	rw	PWM0_TMRxSEL：选择 PWM_TMRx 作为 PWM0 时基 000：选择 PWM_TIMER0 001：选择 PWM_TIMER1 010：选择 PWM_TIMER2 011：选择 PWM_TIMER3
0	PWM0_INV	rw	PWM0_INV：PWM0 信号输出反向控制 0：正向 1：反向

PWMCON1：PWM 控制寄存器 1 ，输出反向和时基选择。

捕获输入引脚 CAPx 和 PWMCHx 输出引脚复用，CAPx 和 PWMTMRx 的捕获输入一一对应。PWM 引脚分配详见芯片 IO mapping 图。

3. JL_PWM->TMRx_CON: pwmtimer x control register(x = 0/1/2/3)

Bit	Name	RW	Description
15:8	reserved	r	预留
7:6	TxMOD	rw	TxMOD:工作模式 0: 计数/定时 1：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） 2：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） 3：IO 口上升下降沿捕获模式（当 IO 上升或下降沿到来时，把 TxCNT 的值捕捉到 TxPR 中）
5	TxPND	rw	TxPND：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。
4	TxPND_CLR	rw	TxPND_CLR: 清除 TxPND 标志位，只写，读为“0” 写 0：无效 写 1：清除 T0PND
3	TxPND_IE	rw	TxPND_IE：TxPND 中断使能 0：禁止 1：允许
2:0	TxCKPS	rw	TxCKPS：时钟预分频设置，TCK/（2^TCKPS）

PWM_TIMER0/1/2/3 是功能相同的 16 位定时器。可作为 PWM0/1/2/3 的时基控制。

4 个定时器的中断合并为 1 个中断请求，具体由软件查询。

定时器时钟源固定为 LSB 时钟。

4. JL_PWM->TMRx_PR: pwmtimer x period register(x = 0/1/2/3)

Bit	Name	RW	Description
15:0	PWMTMRxPR	rw	带缓冲的 16 位周期寄存器

在定时/计数模式下，当 PWM_TMRx_CNT == PWM_TMRx_PR 时，PWM_TMRx_CNT 会被清 0。

在上升沿/下降沿捕获模式下，PWM_TMRx_PR 是作为捕获寄存器使用的，当捕获发生时，PWM_TMRx_CNT 的值会被复制到 PWM_TMRx_PR 中。而此时 PWM_TMRx_CNT 自由的从 0 到 2^(16)-1 再到 0 计数，不会和 PWM_TMRx_PR 进行比较清 0。

5. JL_PWM->TMRx_CNT: pwmtimer x counter register(x = 0/1/2/3)

Bit	Name	RW	Description
15:0	PWMTMRx_CNT	rw	16 位定时/计数器

6. JL_PWM->CHx_CMP: pwm compare x register(x = 0/1/2/3)

Bit	Name	RW	Description
15:0	PWMCMPx	rw	带缓冲的 16 位比较寄存器，对应 PWMCHx 引脚的占空比控制

CMP 重新载入方式固定为：时基 PWM_TMRx_CNT 等于 PWM_TMRx_PR 时候载入。

13.3 使用说明

PWM 输出的占空比 N 的计算公式如下：

N = (PWMCMPx/ (PWM_TMRx_PR+1)) * 100%

PWM 行为：计时器计到 0 时开始翻转为高电平，计时器计数到值 CMP 时翻转为低电平，待计时溢出回归为 0 时又转为高电平，以此往复。所以可以设定好各个 PWM 的定时器 PWM_TMRx_CNT 的初值，然后同时启动各路 PWM_EN 来获得准确的相位差。如 PWM_TMR0_CNT 初值为 0，则 PWM1 和 PWM0 的之间的相位差时间为（PWMTMR1PR-PWM_TMR0_CNT+1）* 时钟(时钟为 TCK1/（2^TCKPS1）)。

第 14 章 UART

14.1 UART0 模块说明

UART0 只支持普通 BUF 传输模式，不支持 DMA 传输模式。不支持小数分频。

14.1.1 UART0 寄存器 SFR 列表

1. UART0_CON: UART0 configuration register

Bit	Name	RW	Description
15	TPND	r	The TX pending: 0：without pending 1：with pending
14	RPND	r	The RX pending: 0：without pending 1：with pending
13	CLR_TPND	w	Clear TX pending: 0：useless 1：clear pending
12	CLR_RPND	w	Clear RX pending: 0：useless 1：clear pending
11	reserved	r	reserved
10	reserved	r	reserved
9	TB8	rw	The ninth bit of data sent when 9-bit mode enable: 0：send 0 1：send 1
8	RB8	r	The ninth bit of data received when 9-bit mode enable: 0：receive 0 1：receive 1
7	reserved	r	reserved
6	reserved	r	reserved
5	RX_DISABLE	rw	The disable of RX: 0：enable RX（正常接收） 1：disable RX（接收固定为 1）
4	DIVS	rw	The pre_division of baud rate selection (more in UARTx_BAUD): 0：DIV4

			1：DIV3
3	RXIE	rw	The interrupt enable of RX: 0：disable 1：enable
2	TXIE	rw	The interrupt enable of TX: 0：disable 1：enable
1	M9EN	rw	The 9-bit mode enable: 0：disable 1：enable
0	UTEN	rw	The UART enable:uart0 接收发送使能 0：disable 1：enable

2. UART0_BAUD: UART0 baud register

Bit	Name	RW	Description
7~0	UART0_BAUD	wo	The UARTx baud

① 配置该 SFR 时，第一次写 UART0_BAUD 是写 UART0_BAUD [15:8]，第二次写 UART0_BAUD 是写 UART0_BAUD [7:0]。（注意每次写该 SFR 都要写两次！）

② UART0_BAUD 和 DIVS（UART0_CON[4]）共同确定最终的波特率，即：

当 **DIVS**=0 时，

Baudrate = Freq_uart / ((**UARTx_BAUD**+1) *4)

当 **DIVS**=1 时，

Baudrate = Freq_uart / ((**UARTx_BAUD**+1) *3)

（其中，Freq_uart 与 uart_clk 选择的时钟有关，详见 Clock_System）

3. UART0_BUF: UARTx buffer register

Bit	Name	RW	Description
7~0	UART0_BUF	rw	The UARTx buffer

该 SFR 既是发送 BUF，也是接收 BUF：

① 发送数据时，往 UART0_BUF 中写入数据，会开始发送数据；

② 接收数据时，可读 UART0_BUF 获得已接收到的数据。

14.2 UART1 模块说明

UART1 支持接收带循环 Buffer 的 DMA 模式和普通模式。

UART1 在 DMA 接收的时候有一个循环 Buffer，UT1_RXSADR 寄存器保存它的起始地址，UT1_RXEADR 寄存器保存它的结束地址。同时，在 DMA 接收过程中，会有一个超时计数器（UT1_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。

14.2.1 UART1 寄存器 SFR 列表

1. UART1_CON(0): UART1 configuration register 0(16bit addressing).

Bit	Name	RW	Description
15	TPND	r	TPND:TX Pending 0：without pending 1：with pending
14	RPND	r	RPND:RX Pending& Dma_Wr_Buf_Empty（数据接收不完 Pending 不会为 1） 0：without pending 1：with pending
13	CLRTPND	r	CLRTPND： 清空 TX Pending 0：useless 1：clear TXpending
12	CLRRPND	r	CLRRPND： 清空 RX Pending 0：useless 1：clear RX pending
11	OTPND	r	（*）OTPND:OverTime Pending 0：without pending 1：with pending
10	CLR_OTPND	r	（*）CLR_OTPND：清空 OTPND 0：useless 1：clear OT pending
9	reserved	r	reserved
8	reserved	r	reserved
7	RDC	w	RDC（*）： 写 1：将已经收到的数目写到 UTx_HRXCNT 寄存器，已收到的数目清零； 写 0：无效。
6	RX_MODE	rw	RXMODE（*）： 读模式选择

			0: 普通模式，不用 DMA ； 1: DMA 模式。
5	OT_IE	rw	OTIE（*）:OT 中断允许 0：disable 1：enable
4	DIVS	rw	DIVS:前 3 分频选择， 0：4 分频， 1：3 分频
3	RXIE	rw	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。 0：disable 1：enable
2	TXIE	rw	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 0：disable 1：enable
1	UTRXEN	rw	UTRXEN:UART 模块接收使能 0：disable 1：enable
0	UTTXEN	rw	UTTXEN:UART 模块发送使能 0：disable 1：enable

（*）：只有 UART1 支持

2. UART1_CON(1): UART1 configuration register 1(16bit addressing).

Bit	Name	RW	Description
15	CTSPND	rw	CTSPND：CTS 中断 pending 0：without pending 1：with pending
14	CLR_CTSPND	wo	CLR_CTSPND： 清楚 CTS pending 0：useless 1：clear CTS pending
13	CLRRTS	wo	CLRRTS：清除 RTS 0: N/A 1: 清空 RTS
12~5	reserved	rw	预留
4	RX_DISABLE	rw	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	CTSIE	rw	CTSIE: CTS 中断使能 0: 禁止中断； 1: 中断允许 注：只有 UART1 有该功能
2	CTSE	rw	CTSE:CTS 使能 0: 禁止 CTS 硬件流控制 1： 允许 CTS 硬件流控制 注：只有 UART1 有该功能
1	RTS_DMAEN	rw	RTS_DMAEN: RTS 接收数据流控制使能 0： 禁止 1: 允许 注：只有 UART1 有该功能
0	RTSE	rw	RTSE:RTS 使能 0: 禁止 RTS 硬件流控制 1: 允许 RTS 硬件流控制 注：只有 UART1 有该功能

3. UART1_CON(2): UART1 configuration register2 (16bit addressing).

Bit	Name	RW	Description
15~3	reserved	r	预留
2	RB8	r	RB8：9Bit 模式时，RX 接收到的第 9 位
1	TB8	rw	TB8：9Bit 模式时，TX 发送的第 9 位
0	M9EN	rw	M9EN：9bit 模式使能

4. UART1_BAUD: UARTx baud register (16bit addressing, Write Only)

Bit	Name	RW	Description
15~0	UTx_BAUD	wo	The UARTx baud

① 配置该 UARTx_BAUD 时，一次写入 16 位数据

② UARTx_BAUD 和 DIVS（UARTx_CON0[4]）共同确定最终的波特率，即：

当 **DIVS**=0 时，

Baudrate = Freq_uart / ((**UARTx_BAUD**+1) *4)

当 **DIVS**=1 时，

Baudrate = Freq_uart / ((**UARTx_BAUD**+1) *3)

（其中，Freq_uart 与 uart_clk 选择的时钟有关，详见 Clock_System）

5. UART1_BUF: UARTx buffer register(8bit addressing)

Bit	Name	RW	Description
7~0	UARTx_BUF	rw	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。

该 SFR 既是发送 BUF，也是接收 BUF：

① 发送数据时，往 UARTx_BUF 中写入数据，会开始发送数据；

② 接收数据时，可读 UARTx_BUF 获得已接收到的数据。

6. UART1_TXADR: uart x TX DMA address(25bit addressing, Write Only)

Bit	Name	RW	Description
24~0	UTx_TXADR	wo	DMA 发送数据的起始地址

7. UART1_TXCNT: uart x TX DMA count (32bit addressing, Write Only)

Bit	Name	RW	Description
31~0	UTx_TXCNT	wo	写 UTx_TXCNT，控制器产生一次 DMA 的操作，同时清空中断，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_CON0[15]）。

8. UART1_RXCNT: uart x receive DMA count(32bit addressing, Write Only)

Bit	Name	RW	Description
31~0	UTx_RXCNT	wo	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_CON0[14]）。

9. UART1_RXSADR: uart x receive DMA address(25bit addressing, Write Only)

Bit	Name	RW	Description
24~0	UTx_RXSADR	wo	DMA 接收数据时，循环 buffer 的起始地址。

10. UART1_RXEADR: uart x receive DMA end address(25bit addressing, Write Only)

Bit	Name	RW	Description
24~0	UTx_RXEADR	wo	MA 接收数据时，循环 buffer 的结束地址。

11. UART1_HRXCNT: uart x have receive DMA count(32bit addressing, Read Only)

Bit	Name	RW	Description
31~0	UTx_HRXCNT	ro	当设 RDC（UTx_CON0[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。

12. UART1_OTCNT: uart x OverTime count(32bit addressing, Write Only)

Bit	Name	RW	Description
31~0	UTx_OTCNT	wo	超时计数器。

该寄存器设置串口设备等待多久 RX 下降沿的时间，如果在所设置的时间里没收到 RX 的下降沿，则产生 OT 中断（OT_PND）

Time(ot)= Time(uart_clk) * UTx_OTCNT

例如：波特率时间为 100ns，UTx_OTCNT = 10,那么，OT 的时间就为 1000ns

注意：

1. OT_PND 触发流程，满足一下（1-4）顺序，则可产生 OT_PND：
   （1） 清 OT_PND；
   （2） 写 UTx_OTCNT；
   （3） 收到 n 个 byte 数据；
   （4） 从最后一个下降沿开始，等待 OT 超时；
   （5） 当 OTCNT 与超时时间相等，OT_PND 置 1。
2. OT_PND 不受 RPND 影响，即 RX_PND 置 1 后，后面如果没有再写一次 UTx_OTCNT， OT_PND 还是会起来；
3. 下降沿包括但不限于 start_bit，数据从 1 到 0 变化；

如果芯片有 lsb_clk，uart_clk 就是 lsb_clk，否则看该芯片的时钟说明；

第 15 章 IO_Mapping_Control

15.1 模块说明

SPI、UART 等外设的端口可支持 re-mapping，即可配置为使用不同的 IO 端口。IOMC0 和 IOMC1 寄存器用于设置此类的 IO re-mapping。

1. 输入：IOMC 寄存器用于配置输入映射，各模块输入映射根据需求有所不同，现以 ICH0_SEL 为例如图 1。

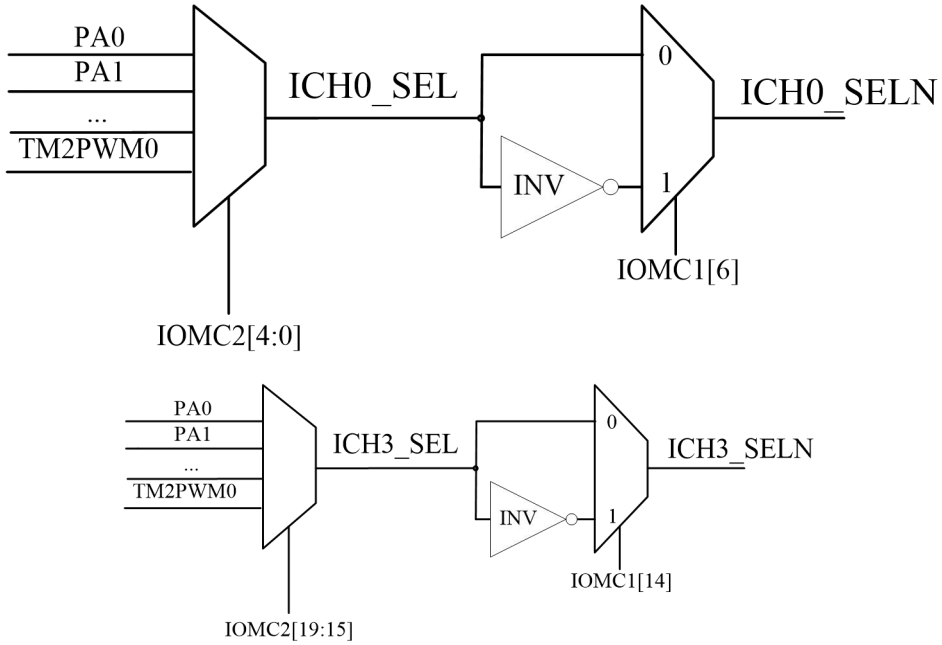


图 1 IO 的输入选通

2. 各引脚功能如下表

引脚	LAT 施放时间	功能								MOTORPWM	FPGA IO
PA0(上拉)	ROM 后	ADC0	APA_DON				支持长按复位	UART0TXB			P00
PA1	ROM 后	ADC1	APA_DOP	spi0_CLKB	IIC_SCL_B		SD0CLK_A	UART0RXB	CAP2		P01
PA2	ROM 后	ADC2	APA_DIN	spi0_DOB(0)	IIC_SDA_B		SD0CMD_A		PWM2H		P02
PA3	ROM 后	ADC3	APA_DIP	spi0_DIB(1)	ISP_DO	SPI1DIA	SD0DAT_A	CLK_OUT	PWM2L	MCAP0	P03
USBDP(下拉)	ROM 中	ADC4		ISP_CLK（mode_det0)	IIC_SCL_A	SPI1CLKA		UART1RXA			USBDP
USBDM(下拉)	ROM 中	ADC5		ISP_DI（mode_det1)	IIC_SDA_A	SPI1DOA	SD0DAT_C	UART1TXA			USBDM
PA4	ROM 后	ADC6		spi0_DAT(2)	IIC_SCL_C	SPI1CLKC	SD0CLK_CD	UART0TXA	TMR2	MPWM0	P04
PA5	ROM 后	ADC7		spi0_DAT(3)	IIC_SDA_C	SPI1DOC	SD0CMD_CD	UART0RXA		MPWM1	P05
PA6	ROM 后		ALNK_MCLK			SPI1DIC	SD0DAT_D				P06
PB2(上拉)(VPP)	ROM 后										P22

第 15 章 IO_Mapping_Control

PA7(VPP)	ROM 后		ALNK_DAT0								P07
PA8(VPP)	ROM 后	OSCI	ALNK_DAT1		IIC_SCL_D				TMR0	MPWM2	P10
PA9(VPP)	ROM 后		ALNK_DAT2		IIC_SDA_D			UART1RXB/UART1TXB	CAP1	MPWM3	P11
PA10	ROM 后	ADC8	ALNK_DAT3			SPI1DIB	SDODAT_B		TMR1	MCAP1	P12
PA11	ROM 后	ADC9	ALNK_SCLK			SPI1CLKB	SDOCLK_B			MCAP2	P13
PA12	ROM 后		ALNK_LRCK			SPI1DOB	SDOCMD_B			MCAP3	P14
PB0	ROM 后	ADC13		DAC	LVD						P20
PB1	ROM 后				MIC_IN						P21
PA13	ROM 后	ADC10		AUX0	MIC_BIAS						P15
PA14	ROM 后	ADC11		AUX1							P16
PA15	ROM 后	ADC12			MIC_LDO						P17
PD0	ROM 中		SFC_CLKA	spi0_CLKA							P86
PD1	ROM 中		SFC_DOA(0)	spi0_DOA(0)							P87
PD2(ROM 上拉)	ROM 中		SFC_CSA	spi0_CSA							P84
PD3	ROM 中		SFC_DIA(1)	spi0_DIA(1)							P85
PD4	ROM 中		Flash Power								P24

注：ADC14 接 AUDIO，ADC15 接 PMU

1.支持两路 OutChannel

2.加简单的触摸键功能

3.记得 PA 的两个脚需要能够通过寄存器配置状态当普通 IO 用。支持 Output_CH。

15.2 寄存器 SFR 列表

1. JL_IOMC->IOMC0:IO Mapping Control register0

Bit	Name	RW	Description
23:20	-	r	预留
19	PWM3IOEN	rw	PWM3IOEN[19]: PWM Channel 3 模块占用 IO 使能 0: PWM 不占用 PA9 1: PWM 占用 PA9
18	PWM2IOEN	rw	PWM2IOEN[18]: PWM Channel 2 模块占用 IO 使能 0: PWM 不占用 PA8 1: PWM 占用 PA8
17	PWM1IOEN	rw	PWM1IOEN[17]: PWM Channel 1 模块占用 IO 使能 0: PWM 不占用 PA5 1: PWM 占用 PA5
16	PWM0IOEN	rw	PWM0IOEN[16]: PWM Channel 0 模块占用 IO 使能 0: PWM 不占用 PA4 1: PWM 占用 PA4

15	TM2IOS	rw	TM2IOS[15]: Timer2 外部时钟输入选择 0: PA4 1: ICH3_SELN
14	TM1IOS	rw	TM1IOS[14]: Timer1 外部时钟输入选择 0: PA10 1: PLL_24M
13	TM0IOS	rw	TM0IOS[13]: Timer0 外部时钟输入选择 0: PA8 1: PLL_24M
12	SPI1IOEN	rw	SPI1IOEN[12]: SPI1 模块占用 IO 使能 0: 不允许 SPI1 模块占用 IO，即使 SPI0 模块已打开 1: 允许 SPI1 模块在打开时占用相应 IO
11:10	SPI1IOS	rw	SPI1IOS[11:10]: SPI1 模块 IO re-mapping 设置 SPI1IOS CLK DO(0) DI(1) 00 USBDP USBDM PA3 01 PA11 PA12 PA10 10 PA4 PA5 PA6
8	UT1IOS	rw	UT1IOS[9:8]: UART1 模块 IO re-mapping 设置 UT1IOS TX RX 00 USBDM USBDP 01 PA9 PA9 10 USBDM ICH0_SEL 11 PA9 ICH0_SEL
7	UT1IOEN	rw	UT1IOEN[7]：允许 UART1 占用 IO 1：占用相应 IO； 0：不占用 IO，该 IO 用于其它功能；
6:5	UT0IOS	rw	UT0IOS[6:5]: UART0 模块 IO re-mapping 设置 UT0IOS TX RX 00 PA4 PA5 01 PA0 PA1 10 PA4 ICH2_SEL 11 PA0 ICH2_SEL
4	UT0IOEN	rw	UT0IOEN[4]：允许 UART0 占用 IO 1：占用相应 IO； 0：不占用 IO，该 IO 用于其它功能；
3	-	rw	预留
2	SPI0IOS	rw	SPI0IOS[2]: SPI0 模块 IO re-mapping 设置 SPI1IOS CLK DO(0) DI(1) DAT2 DAT3 0 PD0 PD1 PD3 PA4 PA5 1 PA1 PA2 PA3 PA4 PA5
1	SPI0MM	rw	SPI0_MM[1]：SPI0 输入结果产生选项

			0：SPI0 内部输入为 DI； 1：SPI0 内部输入为 DI&DO；
0	SPI0IOEN	rw	SPI0IOEN[0]: SPI0 模块占用 IO 使能 0: 不允许 SPI0 模块占用 IO，即使 SPI0 模块已打开 1: 允许 SPI0 模块在打开时占用相应 IO

2. JL_IOMC->IOMC1:IO Mapping Control register1

Bit	Name	RW	Description
23:17	-	r	预留
16	T2PWMHIOEN	rw	T2PWMHIOEN[16]: Timer2 PWMH 占用 IO 允许 0: 不允许 TMR2PWMH 占用 IO（PA2） 1： 允许 TMR2PWMH 占用 IO(PA2)
15	T2PWMLIOEN	rw	T2PWMLIOEN[15]: Timer2 PWML 占用 IO 允许 0: 不允许 TMR2PWML 占用 IO（PA3） 1： 允许 TMR2PWML 占用 IO(PA3)
14	CLKEDGE	rw	CLKEDGE[14]: ICH 边沿选择 0: ICH 原时钟 1: ICH 反向时钟
13	SD0_DTIO_EN	rw	SD0_DTIO_EN[13]：允许 SD0CMD/DAT 占用 IO 0：无论 SD0 有无使能，其 CMD/DAT 不占用相应 IO； 1：当 SD0 使能时，其 CMD/DAT 占用相应 IO；
12	SD0_CKIO_EN	rw	SD0_CKIO_EN[12]：允许 SD0CLK 占用 IO 0：无论 SD0 有无使能，其 CLK 不占用相应 IO； 1：当 SD0 使能时，其 CLK 占用相应 IO；
11:10	SD0IOS	rw	SD0_IOS[11:19]：SD0IO 选择 SD0IOS CLK CMD DAT0 00 PA2 PA1 PA3 01 PA12 PA11 PA10 10 PA5 PA4 USBDM 11 PA5 PA5 PA6
9:8	IICIOS	rw	IIC_IOS[9:8]：IIC 模块 IO 选择 00：选择 USBDP/USBDM 作为 IIC 的 SCLK/SDA； 01：选择 PA1/PA2 作为 IIC 的 SCLK/SDA。 10：选择 PA4/PA5 作为 IIC 的 SCLK/SDA； 11：选择 PA8/PA9 作为 IIC 的 SCLK/SDA；
7	-	r	预留
6	ICH0EDGE	rw	ICH0EDGE[6]: IN_CH0 Mux Edge Select 0: select rising edge 1: select falling edge
5:4	IN_CH0SEL	rw	IN_CH0SEL[5:4]: IN_CH0 Mux to Timers capture select

			0: disable 1: timer0 2: timer1 3: timer2
3	IRFLTOS	rw	IRFLTOS[3]: IRFLT 输出 timer capture 选择，详情请参见 IRFLT 部分 0: IRFLT 的输出至 timer1 的捕获端 1: IRFLT 的输出至 timer2 的捕获端
2:0	-	r	预留

3. JL_IOMC->IOMC2:IO Mapping Control register2

Bit	Name	RW	Description
23:20	-	r	预留
19:15	IN_CH3_SEL	rw	IN_CH3_SEL[19:15]: IO 作为时钟选择 00: PA0 01: PA1 …… 16: PB0 17: PB1 18: PB2 19: PD0 …… 24: USBDP 25: USBDM 26: TM2PWM1 27:TM2PWM0
14:10	IN_CH2_SEL	rw	IN_CH2_SEL[14:10]: IO 作为 WKUP 选择 00: PA0 01: PA1 …… 16: PB0 17: PB1 18: PB2 19: PD0 …… 24: USBDP 25: USBDM 26: TM2PWM1 27:TM2PWM0
9:5	IN_CH1_SEL	rw	IN_CH1_SEL[9:5]:IRFLT 选择 00: PA0

			01: PA1 …… 16: PB0 17: PB1 18: PB2 19: PD0 …… 24: USBDP 25: USBDM 26: TM2PWM1 27:TM2PWM0
4:0	IN_CH0_SEL	rw	IN_CH0_SEL[4:0]:CAPTURE 选择 00: PA0 01: PA1 …… 16: PB0 17: PB1 18: PB2 19: PD0 …… 24: USBDP 25: USBDM 26: TM2PWM1 27:TM2PWM0

4. JL_IOMC->IOMC3:IO Mapping Control register3

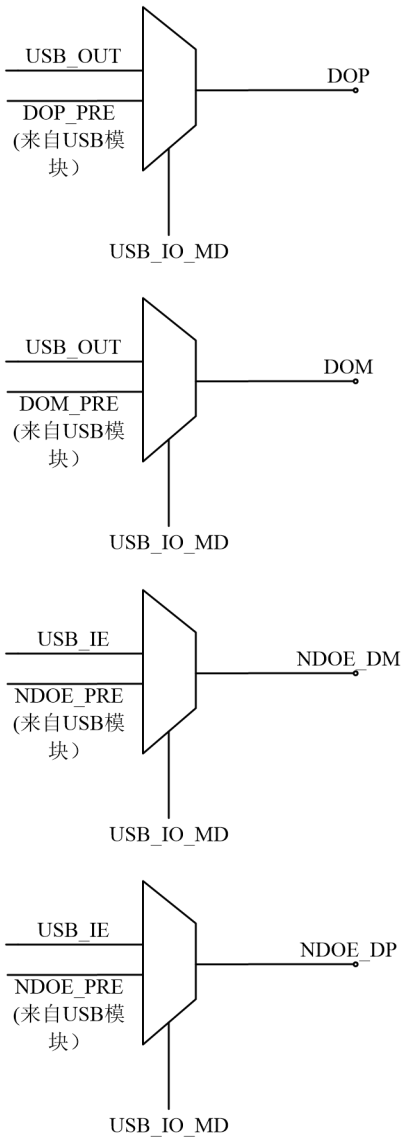
Bit	Name	RW	Description
7:6	-	r	预留
5:3	OUT_CH1_SEL	rw	OUT_CH1S[5:3]：选择输出到 IO 的信号(CHANNEL 1) 000：CH2_PWM; 001：CH3_PWM; 010：选择 TMR2_PWM_OUT0； 011：选择 TMR2_PWM_OUT1； 100：选择 UT0_TX； 101：选择 UT1_TX; 110：选择 CLKOUT； 111：PLL24M；
2:0	OUT_CH0_SEL	rw	OUT_CH0S[2:0]：选择输出到 IO 的信号(CHANNEL 0) 000：CH0_PWM; 001：CH1_PWM; 010：选择 TMR2_PWM_OUT0； 011：选择 TMR2_PWM_OUT1；

			100：选择 SPI0_DO； 101：选择 UT1_TX; 110：选择 ICH3_SELN； 111：PLL24M；

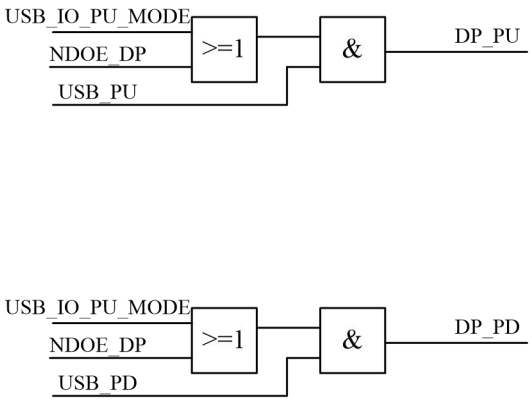
5. JL_IOMC->USB_IO_CON0:USB IO Control register0

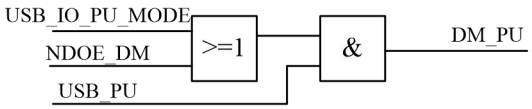
Bit	Name	RW	Description
15	-	r	预留
14	DM_DIEH	rw	DM_DIEH: DM 数字输入允许控制寄存器 1：输入允许 0：输入关闭
13	DP_DIEH	rw	DP_DIEH: DP 数字输入允许控制寄存器 1：输入允许 0：输入关闭
12	USB_SR	rw	
11	USB_IO_MD	rw	USB 输出数据选择
10	DM_DIE	rw	DM_DIE: DM 数字输入允许控制寄存器 1：输入允许 0：输入关闭
9	DP_DIE	rw	DP_DIE: DP 数字输入允许控制寄存器 1：输入允许 0：输入关闭
8	USB_IO_PU_MODE	rw	USB_IO_PU_MODE:USB 上下拉模式选择 1：由 USB_PU 和 USB_PD 控制上下拉 0：由其他和 USB_PU 与 USB_PD 控制上下拉
7:6	USB_PU	rw	USB 端口上拉控制器 BIT7 控制 DM 端，BIT6 控制 DP 端
5:4	USB_PD	rw	USB_PD:USB 端口下拉控制器 BIT5 控制 DM 端，BIT4 控制 DP 端
3:2	USB_DIR	rw	USB_DIR:USB 端口方向寄存器，1：设为输入，0：设为输出 BIT3 控制 DM，BIT2 控制 DP
1:0	USB_OUT	rw	USB_OUT:USB 端口 CPU 输出数据 BIT2 输出到 DM，BIT1 输出到 DP

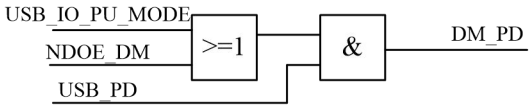
USB_IO_MD:USB 输出数据选择：



USB_IO_PU_MODE:USB 上下拉模式选择：

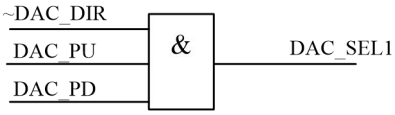






6. JL_IOMC->DAC_IO_CON0:DAC IO Control register0

Bit	Name	RW	Description
15	-	r	预留
14:13	DAC_DIEH	rw	DAC_DIEH: DAC 输入允许控制寄存器 1：输入允许 0：输入关闭
12:11	-	r	预留
10:9	DAC_DIE	rw	DAC_DIE:DAC 输出选择 1：输入允许 0：输入关闭
8	DAC_IO_PU_MOD	rw	DAC_IO_PU_MODE:DAC 上下拉模式选择 1：由 DAC_PU 和 DAC_PD 控制上下拉 0：由其他和 DAC_PU 与 DAC_PD 控制上下拉
7:6	DAC_PU	rw	DAC 端口上拉控制器
5:4	DAC_PD	rw	DAC_PD:DAC 端口下拉控制器
3:2	DAC_DIR	rw	DAC_DIR:DAC 端口方向寄存器，1：设为输入，0：设为输出
1:0	DAC_OUT	rw	DAC_OUT:DAC 端口 CPU 输出数据



第 15 章 IO_Mapping_Control