AC695N 用户手册

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

2019 年 12 月

修改记录

版本	更新日期	描述
V1.0	2019-12-12	初始版本
V1.1	2020-5-5	补充第 23 章 MC_PWM

目 录

序言

AC695N 系列芯片是集成三模蓝牙及 FM 收发的系统级 SOC，支持高性能低功耗的蓝牙及音频应用。芯片内置蓝牙调制解调器、基带及模拟 RF 模块，支持蓝牙 V2.1/V4.2/V5.1 版本及低功耗蓝牙连接待机，通信距离长，待机时间久。芯片集成 FM 立体声收发，性能与单独的外挂 FM 芯片相仿，支持蓝牙后台及蓝牙 FM 同时工作。芯片集成高性能音频 AD/DA，支持 APE、FLAC 等无损音频解码，支持通话回声消除及降噪。AC695N 内置 240Mhz 高性能浮点 DSP，嵌入低延时 cache，方便开发，支持 SDcard、USB、FLASH 等多个外设。

AC695N 主推 BluetoothSmart 和 SmartReady 应用方案，并提供了蓝牙双模的单芯片解决方案，减少设计的工作量让产品更快赢得市场。

AC695N 集成了低成本、超低功耗、2.4G 射频模块，并提供极低的射频活动功耗和 MCU 功耗，支持低功耗模式，出色的电池使用寿命使其适合功耗敏感的应用。

AC695N 提供了完整的 FCC 与 BQB 的认证，使客户在该系列下开发自己的产品时减少了认证的费用。

AC695N 集成了一整套的蓝牙、FM 音频解决方案，为用户开发提供了方便的环境，为产品提供了无限的可能。

第 1 章. 总体介绍

CPU

❖32 位浮点 DSP，最高 240MHz

Memory

❖片上集成了 192K 字节 SRAM

❖16K 4-Way Icache

中断控制器

❖64 个中断源，8 级可编程中断优先级

❖支持 24 个外部 I/O 中断，其中 8 个可低功耗唤醒

❖带软中断（虚拟中断）功能，优先级可以配置

数字 I/O

❖最多 46 个可编程数字 I/O 引脚

❖不同的功能引脚有不同的电流选择，最大拉/灌电流为 64 mA（HD IO）24mA（其它 IO）

❖可配置上拉 10K、下拉 10K 功能

复杂设备

❖Full Speed USB OTG 控制器，除 EP0 外带四个 In EP、四个 Out EP

❖4 路立体声音频 DAC，支持直推耳机

❖3 路立体声音频 ADC

❖带有 1 路 MIC 放大电路

❖FM 调频立体声接收系统，覆盖范围 76-108MHz，步进 50/100KHz（可与蓝牙同时使用）

❖FM 调频立体声发送系统，覆盖范围 76-108MHz，步进 50/100/200KHz（可与蓝牙同时使用）

❖V2.1/ V4.2/ V5.1 版本蓝牙

数字模块

❖4 个 16 位异步分频可重载定时器，可用作计时、捕捉，PWM 模式 (Timer0/1/2/3)

❖硬件看门狗（Watchdog）

❖UART0/1/2 串口控制器,支持 DMA

❖SPI0/1/2 控制器，支持 1/2/4 线（SPI1/2 只支持 1/2 线），支持 DMA

❖SPI FLASH 控制器，只支持跑代码

❖SD0/1 host 控制器， 支持 1/ 4 线，支持 DMA

❖IIC 控制机，支持主从机

❖正交解码器 0/1/2，可同时工作

❖独立供电 RTC，提供闹钟和时基，能够唤醒芯片

❖LCD 控制器，支持最大推 6COM、22SEG 的 LCD 屏

❖AUDIO IIS 0/1 接口，支持片外音频 DAC/ADC，每个接口最大支持 4 路立体声

❖EMI 控制器，支持 DMA 发送

❖SPDIF 音频输入接口（不包含模拟放大）

❖独立推灯模块，可工作在所有 I/O 口，支持呼吸灯，可在正常及低功耗模式下工作

❖INPUT channel 及 OUTPUT channel，为部分模块提供任意的 IO 输入输出选择

❖MCPWM，电机控制模块，3 对 6 路 PWM 输出，支持故障保护及死区控制

❖所有 GPIO 支持普通触摸键接口

模拟模块

❖10 位精度 16 通道 ADC（ADC Key/电压检测等）

❖LVD，支持掉电保护

❖PMU，支持软开关功能，蓝牙 sniff 低功耗，支持电池充电
❖PLL，中心频率为 480MHz
❖RC，内部 RC 时钟，频率在 16MHz 左右
❖LRC，内部低温漂 RC 时钟，频率在 32kHz 左右
❖高精度高速 OSC（12/24/40M）
❖独立 CTMU 高精度触摸按键模块，支持 16 路输入
❖1 个双脚 32K 时钟振荡电路
❖64 位 EFUSE

工作范围

❖工作电压 2.2~5.0V
❖工作温度（-20° C 至+85° C）

应用领域

❖蓝牙音箱
❖蓝牙车机
❖蓝牙声霸
❖蓝牙点烟器

封装

❖LQFP48(7*7)
❖QFN20(3*3)
❖SOP16

第 2 章. 中断系统

2.1. 特征

1. 中断源

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
63-4	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	系统中断源	内核调度入口，不受 IE,IP 和 GIE 控制，使用 syscall 进入
1	系统中断源	内核异常入口，不受 IE,IP 和 GIE 控制
0	系统中断源	仿真调试入口，不受 IE,IP 和 GIE 控制，使用 bkpt 进入

2. 中断控制寄存器 ICFGx

1) 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。

2) 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。

3)中断使能 IE，只要将相应 IE 的控制位打开即可。

3. 总中断

1) 为了方便程序快速打开和关中断。CPU 增设了 GIE0(用户 GIE)，GIE1(操作系统 CLI_GIE)。

2) 当 GIE0 于 GIE1 同时为 1 时，总中断才打开。

4. 中断入口

1) 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。

2) 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。

5. 清外设模块内部的中断即可清除外设中断源。

6. 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

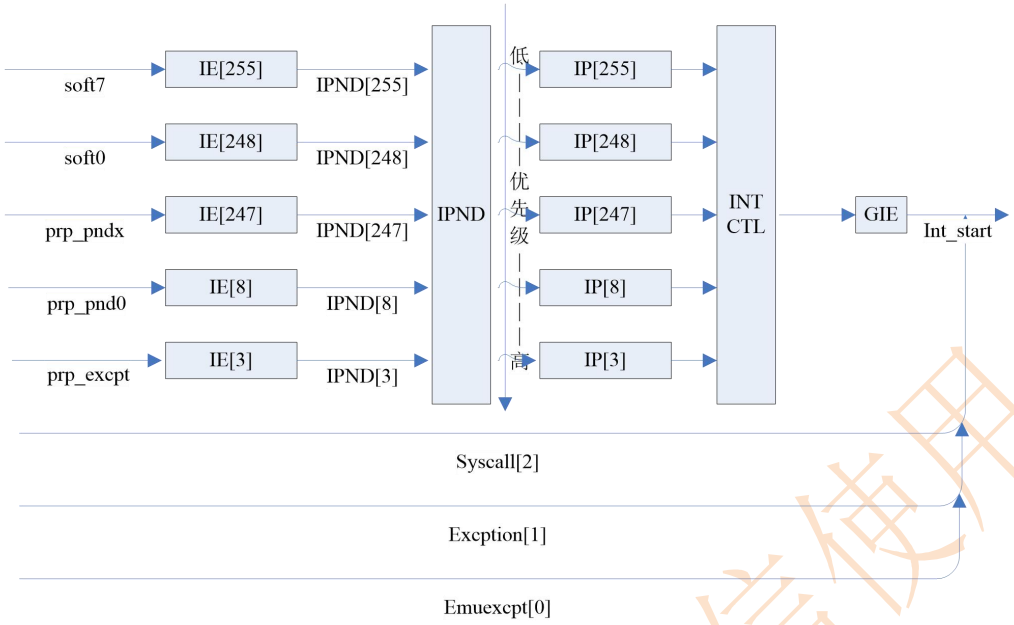


图 2-1 中断系统示意图

2.2. 中断表

中断号	{IP[2:0],IE}	中断入口地址	中断源
63	ICFG7[31:28]	BASE + 中断号×4	soft[3]
62	ICFG7[27:24]	BASE + 中断号×4	soft[2]
61	ICFG7[23:20]	BASE + 中断号×4	soft[1]
60	ICFG7[19:16]	BASE + 中断号×4	soft[0]
59	ICFG7[15:12]	BASE + 中断号×4	
58	ICFG7[11:08]	BASE + 中断号×4	
57	ICFG7[07:04]	BASE + 中断号×4	
56	ICFG7[03:00]	BASE + 中断号×4	
55	ICFG6[31:28]	BASE + 中断号×4	
54	ICFG6[27:24]	BASE + 中断号×4	
53	ICFG6[23:20]	BASE + 中断号×4	CTM
52	ICFG6[19:16]	BASE + 中断号×4	PWM
51	ICFG6[15:12]	BASE + 中断号×4	SPDIF
50	ICFG6[11:08]	BASE + 中断号×4	RDEC2
49	ICFG6[07:04]	BASE + 中断号×4	RDEC1
48	ICFG6[03:00]	BASE + 中断号×4	DCP_INT(DMA_COPY)
47	ICFG5[31:28]	BASE + 中断号×4	FMTX_HWFE
46	ICFG5[27:24]	BASE + 中断号×4	GPC_INT
45	ICFG5[23:20]	BASE + 中断号×4	SBC_INT
44	ICFG5[19:16]	BASE + 中断号×4	SPI2_INT
43	ICFG5[15:12]	BASE + 中断号×4	FMRX_HWFE

42	ICFG5[11:08]	BASE + 中断号×4	CHX_PWM
41	ICFG5[07:04]	BASE + 中断号×4	MCTMRX
40	ICFG5[03:00]	BASE + 中断号×4	AES
39	ICFG4[31:28]	BASE + 中断号×4	BT_EVENT
38	ICFG4[27:24]	BASE + 中断号×4	BT_BLE
37	ICFG4[23:20]	BASE + 中断号×4	OSC_SAFE
36	ICFG4[19:16]	BASE + 中断号×4	ALNK1
35	ICFG4[15:12]	BASE + 中断号×4	PD_TMR1_PND
34	ICFG4[11:08]	BASE + 中断号×4	PD_TMR0_PND
33	ICFG4[07:04]	BASE + 中断号×4	EQ
32	ICFG4[03:00]	BASE + 中断号×4	FFT
31	ICFG3[31:28]	BASE + 中断号×4	SRC_INT
30	ICFG3[27:24]	BASE + 中断号×4	BT_LOFC
29	ICFG3[23:20]	BASE + 中断号×4	BT_DBG
28	ICFG3[19:16]	BASE + 中断号×4	BT_CLKN
27	ICFG3[15:12]	BASE + 中断号×4	BT_BREDR
26	ICFG3[11:08]	BASE + 中断号×4	LRCT
25	ICFG3[07:04]	BASE + 中断号×4	RDEC0
24	ICFG3[03:00]	BASE + 中断号×4	PLNK
23	ICFG2[31:28]	BASE + 中断号×4	SARADC
22	ICFG2[27:24]	BASE + 中断号×4	IIC
21	ICFG2[23:20]	BASE + 中断号×4	PAP
20	ICFG2[19:16]	BASE + 中断号×4	UART2
19	ICFG2[15:12]	BASE + 中断号×4	UART1
18	ICFG2[11:08]	BASE + 中断号×4	UART0
17	ICFG2[07:04]	BASE + 中断号×4	SDC1
16	ICFG2[03:00]	BASE + 中断号×4	SDC0
15	ICFG1[31:28]	BASE + 中断号×4	SPI1
14	ICFG1[27:24]	BASE + 中断号×4	SPI0
13	ICFG1[23:20]	BASE + 中断号×4	PORT
12	ICFG1[19:16]	BASE + 中断号×4	AUDIO
11	ICFG1[15:12]	BASE + 中断号×4	ALNK0
10	ICFG1[11:08]	BASE + 中断号×4	RTC
09	ICFG1[07:04]	BASE + 中断号×4	FUSB_CTL
08	ICFG1[03:00]	BASE + 中断号×4	FUSB_SOF
07	ICFG0[31:28]	BASE + 中断号×4	TMR3
06	ICFG0[27:24]	BASE + 中断号×4	TMR2
05	ICFG0[23:20]	BASE + 中断号×4	TMR1
04	ICFG0[19:16]	BASE + 中断号×4	TMR0
03	ICFG0[15:12]	BASE + 中断号×4	tick_tmr
02	ICFG0[11:08]	BASE + 中断号×4	syscall
01	ICFG0[07:04]	BASE + 中断号×4	exception(misalign/watchdog)
00	ICFG0[03:00]	BASE + 中断号×4	emuexcpt

表 2-1. 中断对应表

2.3. 寄存器说明

1、CPU 内部 SFR：ICFG

icfg 是中断配置寄存器，包括总中断配置及一些中断标志。

Bit	Name	RW/df	Description
31-27	reserved	ro/5'h0	预留,读为 5'h0。
26-24	int_ip	ro/3'h0	上一次中断锁存的优先级
23-16	int_num	ro/8'h0	上一次中断锁存的中断号
15-12	os_flag	rw/4'h0	操作系统标志存储位
11	switch_dis	rw/0	写 1 禁止模式切换时自动切换栈指针。
10	user_mode	ro/0	内部执行模式控制位。
9-8	GIE[1:0]	rw/2'h0	总中断 = GIE[1] & GIE[0]；同时打开才能进入中断。 GIE[1] : CPU 内部总中断 1。（服务于操作系统，是 CLI_GIE） GIE[0] : CPU 内部总中断 0。(服务于普通中断，是 RTI_GIE)
7-0	ibit[7:0]	ro/8'h0	中断阻挡标志位。 ibit[7]=1 时，阻挡所有中断。 ibit[6]=1 时，阻挡中断优先级小于 7 的所有中断。 ibit[5]=1 时，阻挡中断优先级小于 6 的所有中断。 ibit[4]=1 时，阻挡中断优先级小于 5 的所有中断。 ibit[3]=1 时，阻挡中断优先级小于 4 的所有中断。 ibit[2]=1 时，阻挡中断优先级小于 3 的所有中断。 ibit[1]=1 时，阻挡中断优先级小于 2 的所有中断。 ibit[0]=1 时，阻挡中断优先级小于 1 的所有中断。

2、CPU 内部 SFR：IPMASK

IPMASK 是中断优先级屏蔽寄存器，用于快速屏蔽优先级小于 IPMASK 的所有中断。

Bit	Name	RW/df	Description
31-3	reserved	rw/29'h0	预留,读为 29'h0。
2-0	ipmask	rw/3'h0	ipmask=0 时，关闭优先级屏蔽功能。 ipmask=1 时，屏蔽中断优先级小于 1 的所有中断。 ipmask=2 时，屏蔽中断优先级小于 2 的所有中断。 …… ipmask=7 时，屏蔽中断优先级小于 7 的所有中断。

3、CPU 执行模式与指针

第一种情况：switch_dis = 0；

CPU 具有两种执行模式，分别为：用户模式，系统模式（中断时进入系统模式）。

1）sys_mode ：(user_mode==0) → 对应指针 SSP

2）usr_mode ：(user_mode==1) → 对应指针 USP

两种执行模式分别对应两支不同的栈指针，当模式切换时硬件会将系统栈指针切换到当前模式的指针。

在超级模式时：读写 SP 相当于读写 SSP，模式切换时会把 SP 备份到 SSP

在用户模式时：读写 SP 相当于读写 USP，模式切换时会把 SP 备份到 USP

第二种情况：switch_dis = 1；

CPU 具有两种执行模式，分别为：用户模式，系统模式（中断时进入系统模式）。

1）sys_mode ：(user_mode==0) → 对应指针 SP

2）usr_mode ：(user_mode==1) → 对应指针 SP

硬件不自动切指针，SSP/USP 可视为临时寄存器，由程序员控制指针切换。

4、CPU 进入 IDLE

CPP 代码里面写： asm violate(“idle”);

ASM 代码里面写： idle;

第 3 章. 输入/输出（I/O）

3. 1. 概述

3.1.1. 引脚功能复用表

端口	可复用的功能												
PA0	SEG0	AMUX0L							Touch0	CLKOUT0	UART1TXC	PWMCH0H	PO0
PA1	SEG1	AMUX0R						ADC0	Touch1		UART1RXC	PWMCH0L	PO1
PA2	SEG2	PSRAM_D3A	PLNK_SCLK	Q-decoder0_0	spi0_DAT3C(3)	ALNK0_SCLKA			Touch2	CAP3	UART2TXA	TMR0CK	PO2
PA3	SEG3	PSRAM_CKA	PLNK_DAT1	Q-decoder0_1	spi0_CLKC	ALNK0_LRCKA			Touch3		UART2RXA	TMR1CK	PO3
PA4	SEG4	PSRAM_CSA			spi0_CSC	ALNK0_DAT0A			Touch4				PO4
PA5	SEG5		SD0DAT0A	BT_Active		ALNK0_DAT1A	IIC_SCL_D	ADC1	Touch5	PWM0	UART0TXA		PO5
PA6	SEG6	PSRAM_D1A	SD0DAT1A	Wlan_Active	spi0_DIC(1)	ALNK0_DAT2A	IIC_SDA_D	ADC2	Touch6		UART0RXA		PO6
PA7	SEG7	PSRAM_D2A	SD0DAT2A	BT_priority	spi0_DAT2C(2)	ALNK0_DAT3A			Touch7	TMR0			PO7
PA8	SEG8	PSRAM_D0A	SD0DAT3A	BT_Freq	spi0_DOC(0)	ALNK0_MCLKA						FPIN2	PQ0
PA9	SEG9	PA_EN	SD0CMDA	SPDIF_IN_A	PAPD0	ALNK0_SCLKB	UART1_CTS		Touch8		UART2TXB	PWMCH4H	PQ1
PA10	SEG10	LNA_EN	SD0CLKA	SPDIF_IN_B	PAPD1	ALNK0_LRCKB	UART1_RTS	ADC3	Touch9	TMR1	UART2RXB	PWMCH4L	PQ2
PA11	SEG11				PAP_RD	ALNK0_DAT0B					UART0TXD		PQ3
PA12	SEG12				PAP_WR	ALNK0_DAT1B		ADC4		PWM1	UART0RXD	TMR2CK	PQ4
PA13	SEG13		spi0_CSB	SFC_CSB	SPI2DIB	ALNK0_DAT2B						TMR3CK	PQ5
PA14	SEG14		spi0_DIB(1)	SFC_DIB(1)		ALNK0_DAT3B						FPIN0	PQ6
PA15	SEG15		spi0_DAT2B(2)	SFC_DAT2B(2)		ALNK0_MCLKB				CAP2		FPIN1	PQ7
PB0（高压）			SDTAP_CLKC	SPDIF_IN_C	SPI1CLKA	ALNK1_MCLK					UART1TXA	PWMCH1H	PP0
PB1(上拉)		长按 Reset	SDTAP_DATC	SPDIF_IN_D	SPI1DOA			ADC5		TMR2	UART1RXA		PP1
PB2（高压）				Q-decoder1_0	SPI1DIA							PWMCH1L	PP2
PB3				Q-decoder1_1				ADC6		PWM2		TMR4CK	PP3
PB4		Q-decoder2_0	SD1DAT0B	PSRAM_D3B	SD0DAT3B	spi0_DAT3D(3)	IIC_SCL_C	ADC7	LVD		UART0TXB	PWMCH2H	PP4
PB5（高压）			SD1CMDB	PSRAM_CKB	SD0DAT2B	spi0_CLKD				PWM3/ CAP1	UART0TXC/ UART0RXC		PP5

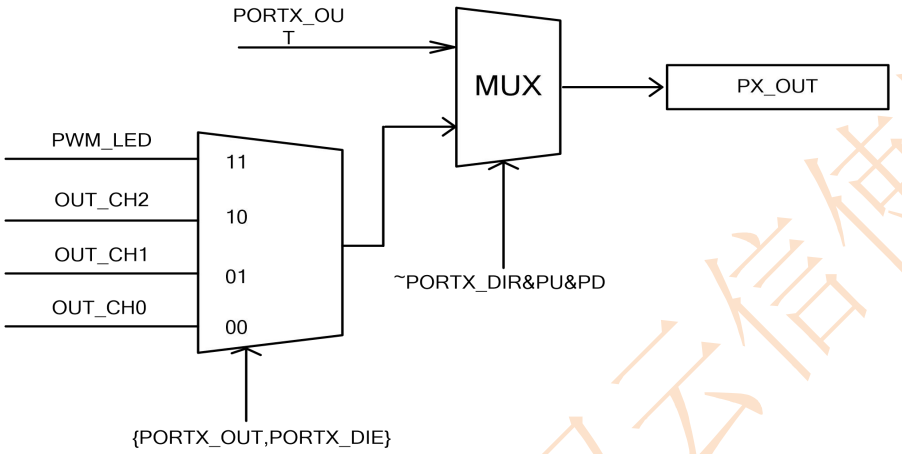
PB6		Q-decoder2_1	SD1CLKB	PSRAM_CSB	SD0DAT1B	spi0_CSD	IIC_SDA_C			TMR3	UART0RXB	PWMCH2L	PP6
PB7		AMUX1L	spi0_DAT2A(2)	SFC_DAT2A(2)									PP7
PB8		AMUX1R	SD1DAT1B	PSRAM_D1B	SD0DAT0B	spi0_DID(1)	SPI2DIA	ADC8		CLKOUT1			PU0
PB9	SDTAP_CLKD	AMUX2L	SD1DAT2B	PSRAM_D2B	SD0CLKB	spi0_DAT2D(2)	SPI2CLKA			CAP0	UART2TXC	PWMCH3H	PU1
PB10	SDTAP_DATD	AMUX2R	SD1DAT3B	PSRAM_D0B	SD0CMDB	spi0_DOD(0)	SPI2DOA	ADC9	FM_TX_A		UART2RXC	PWMCH3L	PU2
PB11 （超强驱动）	SDPG	SD 卡供电脚		SPIDIF_OUT					FM_TX_B				PU3
PC0	SEG16/COM5	SD1DAT3A		ALNK1_SCLK	PAPD2				Touch10		UART1TXB	FPIN3	PE0
PC1	SEG17/COM4	SD1DAT2A		ALNK1_LRCK	PAPD3				Touch11		UART1RXB	FPIN4	PE1
PC2	SEG18/COM3	SD1DAT1A		ALNK1_DAT0	PAPD4				Touch12			FPIN5	PE2
PC3	COM2	SD1DAT0A		ALNK1_DAT1	PAPD5	SPI1DIB			Touch13			TMR5CK	PE3
PC4	COM1	SD1CMDA	SDTAP_CLKA	ALNK1_DAT2	PAPD6	SPI1CLKB	IIC_SCL_B	ADC10	Touch14		UART2TXD	PWMCH5H	PE4
PC5	COM0	SD1CLKA	SDTAP_DATA	ALNK1_DAT3	PAPD7	SPI1DOB	IIC_SDA_B	ADC13	Touch15		UART2RXD	PWMCH5L	PE5
PC6	SEG20	MIC		ISP_DO				ADC11					PE6
PC7 （超强驱动）	SEG21	MIC_BIAS											PE7
PD0			spi0_CLKAB	SFC_CLKAB									
PD1			spi0_DOAB(0)	SFC_DOAB(0)									
PD2			spi0_DIA(1)	SFC_DIA(1)									
PD3			spi0_CSA	SFC_CSA									
PD4 （超强驱动）		Flash 供电脚											
PD5	SEG19		spi0_DAT3AB(3)	SFC_DAT3AB(3)									
PR0 （RTCVDD 供电）		OSCI_32K											
PR1 （RTCVDD 供电）		OSCO_32K											
USBDP(下拉)			SDTAP_CLKB	ISP_CLK （mode_det	SPI2CLKB		IIC_SCL_A	ADC12			UART1TXD		
UDBDM（下拉）			SDTAP_DATB	ISP_DI （mode_det1	SPI2DOB		IIC_SDA_A				UART1RXD		

3.2. 寄存器说明

PAP/SD/SPI/IIC/UART 等外设的端口可支持 remapping，即可配置为使用不同的 IO 端口。IOMAP_CONx 寄存器用于设置此类的 IO remapping。

注意：下文中的 IO 不代表实际 IO 数量，请参照 IO_MAPPING 来选择对应的 IO，如果 CHANNEL 选择到了 IO_MAPPING 上不存在的 IO，结果不可预测。

IO 的输出选通如下图：



3.2.1. IO 功能设置寄存器

1.IOMAP_CON0: IO mapping control register0

Bit	Name	RW	Default	Description
31:18	-	-	-	预留
23	SPDIF_IN_D_IOS	rw	0	SPDIF_IN_D 输入选择 0： input channel 9 1： PB1
22	SPDIF_IN_C_IOS	rw	0	SPDIF_IN_C 输入选择 0： input channel 8 1： PB0
21	SPDIF_IN_B_IOS	rw	0	SPDIF_IN_B 输入选择 0： input channel 10 1： PA10
20	SPDIF_IN_A_IOS	rw	0	SPDIF_IN_A 输入选择 0： input channel 11 1： PA9
19:18	-	-	-	预留
17:16	SPI0_IOS	rw	0	SPI0 IO 选择（见 IO spec） 0： 选择 A 组；

				1： 选择 B 组； 2： 选择 C 组； 3： 选择 D 组
15	SD0_IOS	rw	0	SD0 IO 选择 0: 选 择 PA9/PA10/{PA8,PA7,PA6,PA5} 作 为 SD0 的 CMD/CLK/DAT； 1: 选 择 PB10/PB9/{PB4,PB5,PB6,PB8} 作 为 SD0 的 CMD/CLK/DAT；
14	PAP_DEN	rw	0	允许 PAPDx 占用 IO 0:无论 PAP 有无使能，其 PAPDx 不占用相应 IO； 1:当 PAP 使能时，其 PAPDx 占用相应 IO；
13	PAP_REN	rw	0	允许 PAP_RD 占用 IO 0:无论 PAP 有无使能，其 PAP_RD 不占用相应 IO； 1:当 PAP 使能时，其 PAP_RD 占用相应 IO；
12	PAP_WEN	rw	0	允许 PAP_WR 占用 IO 0:无论 PAP 有无使能，其 PAP_WR 不占用相应 IO； 1:当 PAP 使能时，其 PAP_WR 占用相应 IO；
11	ALNK0_IOS	rw	0	ALNK0 IO 选择 0:选 择 PA2/PA3/PA8/{PA7,PA6,PA5,PA4} 作 为 ALNK0 的 SCLK/LRCK/ MCLK/DAT； 1:选择 PA9/PA10/PA15/{PA14,PA13,PA12,PA11} 作为 ALNK0 的 SCLK/LRCK/MCLK/DAT
10:8	IRFLT_OS	rw	0	与 IRFLT_EN 一起组成 TIMR 捕获端选择 0xx:IRFLT 不输出； 100:IRFLT 输出至 timer0 的捕获端； 101:IRFLT 输出至 timer1 的捕获端； 110:IRFLT 输出至 timer2 的捕获端； 111:IRFLT 输出至 timer3 的捕获端；
7:6	UT0_IOS	rw	0	UART0 模块 IO re-mapping 设置 UT0IOS / TX / RX 0 / PA5 / PA6 1 / PB4 / PB6 2 / PB5 / PB5 3 / PA11 / PA12
5	-	-	-	预留
4	SD1_IOS	rw	0	SD1 IO 选择 0: 选择 PC4/PC5/{PC0，PC1，PC2，PC3}作为 SD1 的 CMD/CLK/DAT； 1: 选择 PB5/PB6/{PB10，PB9，PB8，PB4}作为 SD0 的 CMD/CLK/DAT；
3	SD1_DT_EN	rw	1	允许 SD1 CMD/DAT 占用 IO 0:无论 SD1 有无使能，其 CMD/DAT 不占用相应 IO； 1:当 SD1 使能时，其 CMD/DAT 占用相应 IO；

2	SD1_CK_EN	rw	1	允许 SD1 CLK 占用 IO 0:无论 SD1 有无使能，其 CLK 不占用相应 IO； 1:当 SD1 使能时，其 CLK 占用相应 IO
1	SD0_DT_EN	rw	1	允许 SD0 CMD/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
0	SD0_CK_EN	rw	1	允许 SD0 CLK 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

2.IOMAP_CON1: IO mapping control register1

Bit	Name	RW	Default	Description
31	TMR3_CAP_IOS	rw	0	TIMER3 边沿捕抓 IO 选择 0: PA2 1: input channel 2
30	TMR2_CAP_IOS	rw	0	TIMER2 边沿捕抓 IO 选择 0: PA15 1: input channel 2
29	TMR1_CAP_IOS	rw	0	TIMER1 边沿捕抓 IO 选择 0: PB5 1: input channel 2
28	TMR0_CAP_IOS	rw	0	TIMER0 边沿捕抓 IO 选择 0: PA9 1: input channel 2
27	TMR3_CIN_IOS	rw	0	TIMER3 外部时钟输入 IO 选择 0 :PB6 1: pll_12m
26	TMR2_CIN_IOS	rw	0	TIMER2 外部时钟输入 IO 选择 0 :PB1 1: input channel 4
25	TMR1_CIN_IOS	rw	0	TIMER1 外部时钟输入 IO 选择 0 :PA10 1: pll_24m
24	TMR0_CIN_IOS	rw	0	TIMER0 外部时钟输入 IO 选择 0 :PA7 1: input channel 4
23	-	-	0	预留
22:21	SDTAP_IOS	rw	0	SDTAP 模块 IO re-mapping 设置 UT2_IOS / CLK / DAT 0 / PC4 / PC5 1 / USBDP / USBDM 2 / PB0 / PB1 3 / PB9 / PB10

<table>
<tr><td>20</td><td>PLNK_SCKOE</td><td>rw</td><td>0</td><td>PLNK_SCK 从 PA2 输出使能</td></tr>
<tr><td>19:18</td><td>IIC_IOS</td><td>rw</td><td>-</td><td>IIC 模块 IO 选择<table><tr><td>IIC_IOS</td><td>SCL</td><td>SDA</td></tr><tr><td>0</td><td>USBDP</td><td>USBDM</td></tr><tr><td>1</td><td>PC4</td><td>PC5</td></tr><tr><td>2</td><td>PB4</td><td>PB6</td></tr><tr><td>3</td><td>PA5</td><td>PA6</td></tr></table></td></tr>
<tr><td>17</td><td>-</td><td>-</td><td>0</td><td>预留</td></tr>
<tr><td>16</td><td>SPI2_IOS</td><td>rw</td><td>0</td><td>SPI2 IO 选择
0: PB9/PB10/PB8 => CLK/DO/DI
1: USBDP/USBDM/PA13 => CLK/DO/DI</td></tr>
<tr><td>15:14</td><td>UT2_IOS</td><td>rw</td><td>0</td><td>UART2 模块 IO re-mapping 设置<table><tr><td>UT2_IOS</td><td>TX</td><td>RX</td></tr><tr><td>0</td><td>PA2</td><td>PA3</td></tr><tr><td>1</td><td>PA9</td><td>PA10</td></tr><tr><td>2</td><td>PB9</td><td>PB10</td></tr><tr><td>3</td><td>PC4</td><td>PC5</td></tr></table></td></tr>
<tr><td>13</td><td>RDEC0_SIN1_IOS</td><td>rw</td><td>0</td><td>rdec0_sin1 输入选择
0:input channel 7;
1:PA3;</td></tr>
<tr><td>12</td><td>RDEC0_SIN0_IOS</td><td>rw</td><td>0</td><td>rdec0_sin0 输入选择
0:input channel 6;
1:PA2;</td></tr>
<tr><td>11:8</td><td>OUTPUT_CH0_SEL</td><td>rw</td><td>0</td><td>选择输出到 IO 的信号<table><tr><td>0</td><td>UT0_TX</td></tr><tr><td>1</td><td>UT1_TX</td></tr><tr><td>2</td><td>TMR0_PWM_OUT</td></tr><tr><td>3</td><td>TMR1_PWM_OUT</td></tr><tr><td>4</td><td>RTC_OSCL</td></tr><tr><td>5</td><td>BTOSC_CLK</td></tr><tr><td>6</td><td>PLL_12M</td></tr><tr><td>7</td><td>UT2_TX</td></tr><tr><td>8</td><td>CH0_PWMH</td></tr><tr><td>9</td><td>CH0_PWML</td></tr><tr><td>10</td><td>CH1_PWMH</td></tr><tr><td>11</td><td>CH1_PWML</td></tr><tr><td>12</td><td>CH2_PWMH</td></tr><tr><td>13</td><td>CH2_PWML</td></tr><tr><td>14</td><td>WLC_INT_FREQ</td></tr><tr><td>15</td><td>TMR3_PWM_OUT</td></tr></table></td></tr>
<tr><td>7</td><td>UT1_CTS_IOS</td><td>rw</td><td>0</td><td>UART1 CTS 选择
0:input channel 5;
1:PA5;</td></tr>
<tr><td>6</td><td>CAP_ES</td><td>rw</td><td>0</td><td>input channel2 边沿选择</td></tr>
</table>

				0:上升沿； 1:下降沿；
5	SFC_IOS	rw	0	SFC 模块 IO 选择 0:选择 PD0/PD3/{PD5，PB7，PD2，PD1}作为 SFC 的 CLK/CS/DAT； 1:选择 PD0/PA13/{PD5，PA15，PA14，PD1}作为 SFC 的 CLK/CS/DAT
4	SPI1_IOS	rw	0	SPI1 模块 IO 选择 0:选择 PB0/PB1/PB2 作为 SPI1 的 CLK/DO/DI； 1:选择 PC4/PC5/PC3 作为 SPI1 的 CLK/DO/DI；
3:2	UT1_IOS	rw	0	UART1 模块 IO re-mapping 设置 UT1_IOS / TX / RX 0 / PB0 / PB1 1 / PC0 / PC1 2 / PA0 / PA1 3 / USBDP / USBDM
1	SPI0_DIDO_MIX	rw	0	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	PSR_IOS	rw	0	PSRAM 模块 IO 选择 0: 选择 PA3/PA4/{PA2，PA7，PA6，PA8}作为 PSRAM 的 CK/CS/DAT； 1: 选择 PB5/PB6/{PB4，PB9，PB8，PB10}作为 PSRAM 的 CK/CS/DAT

3.IOMAP_CON2: IO mapping control register2

Bit	Name	RW	Default	Description
31:30	-	-	0	预留
29:24	INPUT CHANNEL 3	rw	0	wkup 输入选择 0-15 :选择 PA0-PA15 16-31:选择 PB0-PB15 32-47:选择 PC0-PC15 48-55:选择 PD0-PD7 56-58:无 59:选择 TMR0_PWM_OUT 60:选择 TMR1_PWM_OUT 61:选择 USBDP 62:选择 USBDM 63:无
23:22	-	rw	0	预留
21:16	INPUT CHANNEL 2	rw	0	IRFLT 功能脚选择，同 INPUT CHANNEL3
15	RDEC2_SIN1_IOS	rw	0	rdec2_sin1 输入选择 0:input channel 7

				1:PB6
14	RDEC2_SIN0_IOS	rw	0	rdec2_sin0 输入选择 0:input channel 6 1:PB4
13:8	INPUT CHANNEL 1	rw	0	CAPTURE 功能脚选择,同 INPUT CHANNEL3
7	RDEC1_SIN1_IOS	rw	0	rdec1_sin1 输入选择 0:input channel 7 1:PB3
6	RDEC1_SIN0_IOS	rw	0	rdec1_sin0 输入选择 0:input channel 6 1:PB2
5:0	INPUT CHANNEL 0	rw	0	UART_RX 脚选择,同 INPUT CHANNEL3

4.IOMAP_CON3: IO mapping control register3

Bit	Name	RW	Default	Description
31:28	-	-	0	预留
27:24	OUTPUT_CH2_SEL	rw	0	0 UT1_RTS 1 UT1_TX 2 WLC_INT_ACTIVE 3 TMR1_PWM_OUT 4 PLNK_SCLK 5 BTOSC_CLK 6 PLL_24M 7 UT2_TX 8 CH0_PWMH 9 CH0_PWML 10 CH1_PWMH 11 CH1_PWML 12 CH2_PWMH 13 CH2_PWML 14 TMR2_PWM_OUT 15 TMR3_PWM_OUT
23:20	OUTPUT_CH1_SEL	rw	0	0 UT0_TX 1 UT1_TX 2 TMR0_PWM_OUT 3 WLC_INT_STATUS

				4 RTC_OSL 5 BTOSC_CLK 6 SPDIF_DO 7 UT2_TX 8 CH0_PWMH 9 CH0_PWML 10 CH1_PWMH 11 CH1_PWML 12 CH2_PWMH 13 CH2-PWML 14 TMR2_PWM_OUT 15 TMR3_PWM_OUT
19:16	-	-	0	预留
15	PLNK_D1_IOS		1	pdm link data1 select 0 ： PA3 1 ： input channel 9
14	-	-	0	预留
13	WLC_EXT_IOS	rw	0	wlc_ext_active IO 选择 0 ： PA6 1 ： input channel 10
12	MC_FPIN_IOS	rw	0	mc_fpin IO 选择 0 ： PA14 1 ： input channel 2
11	UT2_IOEN	rw	1	允许 UART2 占用 IO 1:占用相应 IO； 0:不占用 IO，该 IO 用于其它功能；
10:8	UT2_MXS	rw	0	UART2 输入选择 0XX:选择普通 IO 作为输入（UT2_IOS）； 100:选择由 WKUP_MUX（input channel 0）选择的 IO 作为输入； 101:选择由 IRFT_MUX（input channel 1）选择的 IO 作为输入； 110:选择由 CAP_MUX（input channel 2）选择的 IO 作为输入； 111:选择由 UART_RX_MUX（input channel 3）选择的 IO 作为输入；
7	UT1_IOEN	rw	1	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

6:4	UT1_MXS	rw	0	UART1 输入选择 0XX:选择普通 IO 作为输入（UT1_IOS）； 100:选择由 WKUP_MUX（input channel 0）选择的 IO 作为输入； 101:选择由 IRFT_MUX（input channel 1）选择的 IO 作为输入； 110:选择由 CAP_MUX（input channel 2）选择的 IO 作为输入； 111:选择由 UART_RX_MUX（input channel 3）选择的 IO 作为输入；
3	UT0_IOEN	rw	1	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；
2:0	UT0_MXS	rw	0	UART0 输入选择 0XX:选择普通 IO 作为输入（UT0_IOS）； 100:选择由 WKUP_MUX（input channel 0）选择的 IO 作为输入； 101:选择由 IRFT_MUX（input channel 1）选择的 IO 作为输入； 110:选择由 CAP_MUX（input channel 2）选择的 IO 作为输入； 111:选择由 UART_RX_MUX（input channel 3）选择的 IO 作为输入

5.IOMAP_CON4: IO mapping control register4

Bit	Name	RW	Default	Description
31:30	-	-	0	预留
29:24	INPUT CHANNEL 7		0	RDEC_DI[1] 输入选择 0-15 :选择 PA0-PA15 16-31:选择 PB0-PB15 32-47:选择 PC0-PC15 48-55:选择 PD0-PD7 56-58:无 59:选择 TMR0_PWM_OUT 60:选择 TMR1_PWM_OUT 61:选择 USBDP 62:选择 USBDM 63:无
23:22	-	-	0	预留
21:16	INPUT CHANNEL 6		0	RDEC_DI[0]选择，同 INPUT CHANNEL 7
15:14	-	-	0	预留
13:8	INPUT CHANNEL 5		0	UART1_CTS 选择,同 INPUT CHANNEL 7

7:6	-	-	0	预留
5:0	INPUT CHANNEL 4		0	TIMER CLOCK 选择,同 INPUT CHANNEL 7

6.IOMAP_CON5: IO mapping control register5

Bit	Name	RW	Default	Description
31:30	-	-	0	预留
29:24	INPUT CHANNEL 11	rw	0	SPDIF_DI_A_P 输入选择 0-15 :选择 PA0-PA15 16-31:选择 PB0-PB15 32-47:选择 PC0-PC15 48-55:选择 PD0-PD7 56-58:无 59:选择 TMR0_PWM_OUT 60:选择 TMR1_PWM_OUT 61:选择 USBDP 62:选择 USBDM 63:无
23:22	-	-	0	预留
21:16	INPUT CHANNEL 10	rw	0	WLC_EXT_ACTIVE_P 的输入选择,同 INPUT CHANNEL 11
15:14	-	-	0	预留
13:8	INPUT CHANNEL 9	rw	0	PDLINK_IN1 的输入选择,同 INPUT CHANNEL 11
7:6	-	-	0	预留
5:0	INPUT CHANNEL 8	rw	0	PDLINK_IN0 的输入选择,同 INPUT CHANNEL 11

第 4 章. 时钟系统（Clock_System）

4. 1. 概述

1. AC695N 具备如下 3 个原生时钟源，可直接驱动系统运行：
   1）rc_clk：来自 RC 振荡器，振荡频率约 16MHz，随电压和温度不同会有较大变化。
   2）bt_osc:来自蓝牙模块的振荡器，外部需在 BTXOSCO 和 BTXOSCI 引脚挂载 12-26MHz 晶振（内部已有谐振电容，也可外置）。
   3）rtc_osl：来自 RTC 的晶振，振荡频率约 32KHz，外部需在 PR0 和 PR1 引脚挂载晶振。
2. 额外的，AC695N 还具备如下 1 个原生时钟源，不可直接驱动系统运行，只能驱动 PMU 和 RTC 模块：
   1）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，可用作蓝牙 sniff 或实时时钟计时。
3. AC695N 还具备如下 1 个衍生时钟源：
   1）pll_src:来自片内 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(PLL_CON1[24:28])。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。

4.1.1. 结构框图

1. PLL 只支持整数工作模式，整数模式支持 12 或 24MHz 参考频率。总体而言，参考频率越高，PLL 的性能越好。

1)INTE 整数模式下

$$Fout = Fref / n * m$$

（n = PLL_CON0[8:2]+2，m = PLL_CON1[11:0]+2）

2)PLL 的输入参考时钟可由 rc_clk、btosc_clk 或 pat_clk 提供。

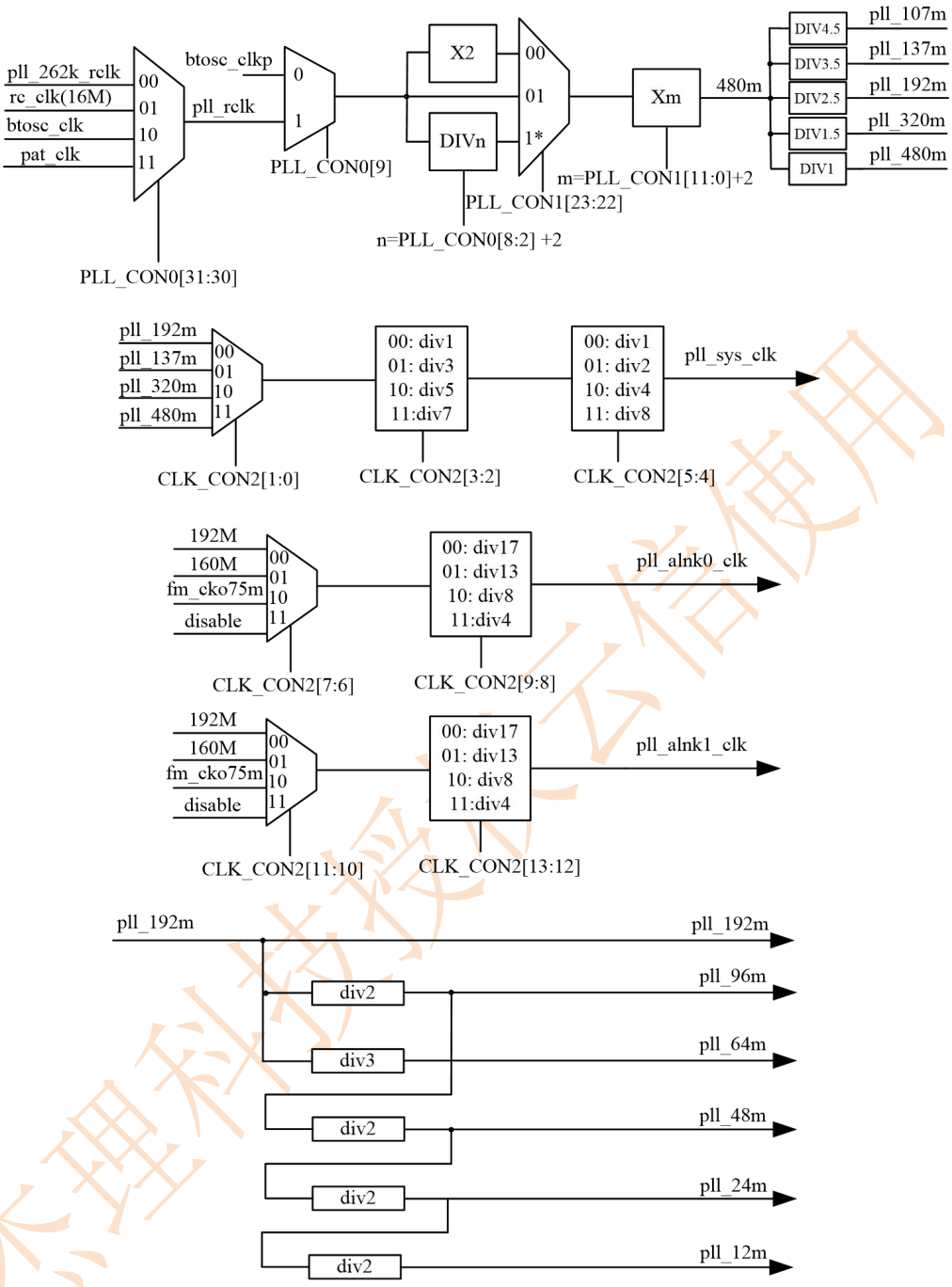


图 4-1. 系统部分时钟结构框图

2．AC695N 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。

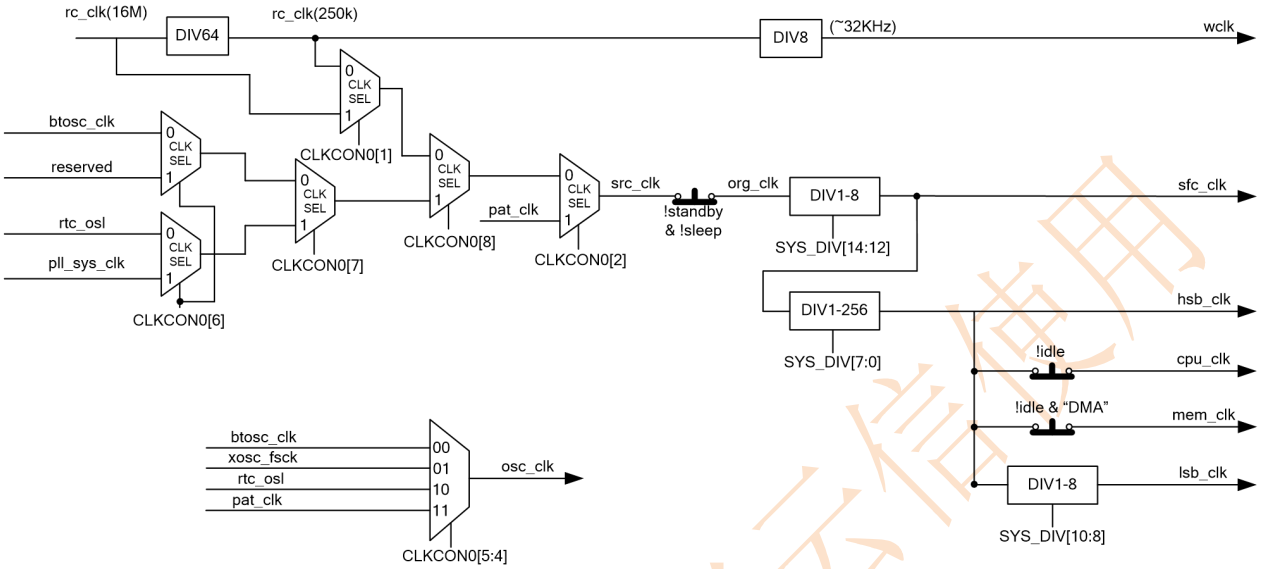


图 4-2. 部分系统时钟电路

3．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。

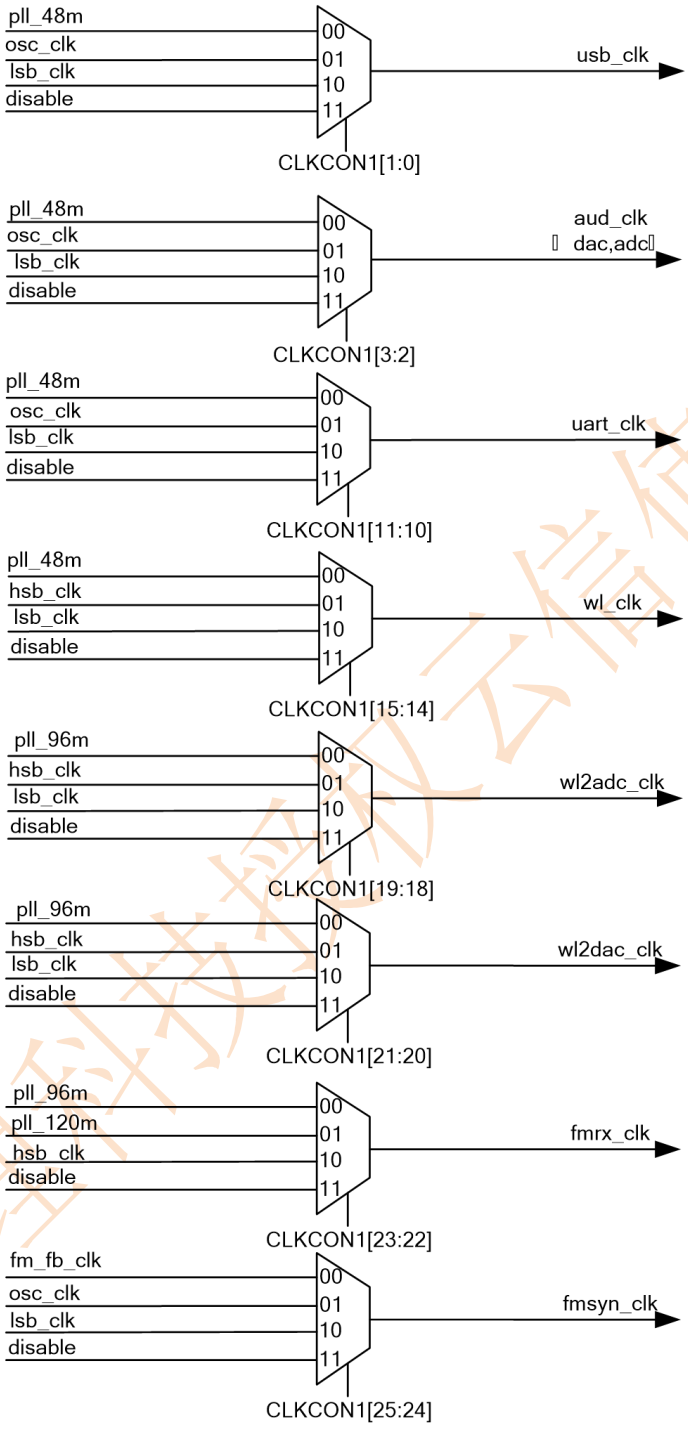


图 4-3. 异步外设时钟电路

4．AC695N 的 PC0/PA4 引脚为复用测试引脚，该引脚除了普通的 PC0/PA4 的 IO 功能之外，还可以将内部的时钟信号驱动至片外，用于测试或特殊用途。

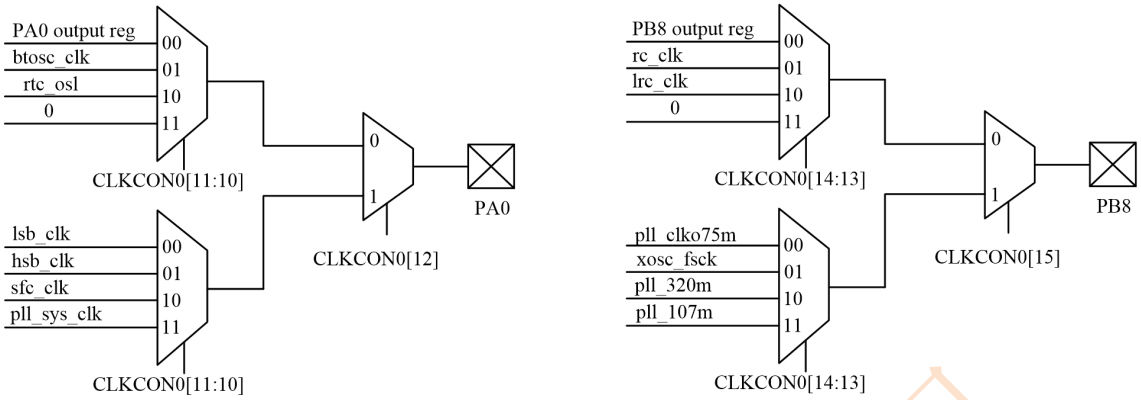


图 4-4. IO 输出时钟电路

4. 2. 寄存器说明

1.CLK_CON0

Bit	Name	R/W	Default	Description
31:16	-	-	-	预留
15:13	PB8_SEL	RW	0	PB8 输出选择
12:10	PA0_SEL	RW	0	PA0 输出选择
9	SFR_CKMD	RW	0	sfr_clk 时钟模式选择： 0:有 SFR 写的时候，sfr_clk 才使能（=hsb_clk） 1:无论有无 SFR 写，sfr_clk 都使能（=hsb_clk）
8:6	CK_SEL	RW	0	主时钟切换，见图 4-2
5:4	OSC_SEL	RW	0	片内 OSC_CLK 来源选择，见图 4-2
3:2	TS_SEL	RW	0	系统时钟源测试选择，见图 4-2
1	RCH_SEL	RW	0	片内 RC 振荡器频率选择 0：250k 1：16M
0	RC_EN	RW	1	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

Bit	Name	R/W	Default	Description
31	AXI_CKEN	RW	1	axi_clk 使能： 0:不使能 1:使能
30	-	-	-	预留

29:28	SFC_DL_SEL	RW	0	SFC 采样时钟延迟（只用于 IC 仿真）
27:26	-	-	-	预留
25:24	FMSYN_CKSEL	RW	0	fmsyn_clk 时钟选择，见图 4-3
23:22	FMRX_CKSEL	RW	0	fmrx_clk 时钟选择，见图 4-3
21:20	WL2DAC_CKSEL	RW	0	wl2dac_clk 时钟选择，见图 4-3
19:18	WL2ADC_CKSEL	RW	0	wl2adc_clk 时钟选择，见图 4-3
17	SFC_SCKE	RW	0	sfc 时钟使能： 0:不使能 1:使能
16	-	-	-	预留
15:14	BT_CKSEL	RW	0	蓝牙时钟选择，见图 4-3
13:12	-	-	-	预留
11:10	UART_CKSEL	RW	0	uart 时钟选择，见图 4-3
9:7	-	-		预留
6:4	GPC_CKSEL	RW	0	GPCNT 要计算的输入时钟选择（GSS=3’b100 时） 000:0 001:src_clk 010:sfc_clk 011:hsb_clk 100:0 101:aud_clk 110:wl_clk 111:usb_clk
3:2	AUD_CKSEL	RW	0	audio 时钟选择，见图 4-3
1:0	USB_CKSEL	RW	0	usb 时钟选择，见图 4-3

3.CLK_CON2

Bit	Name	R/W	Default	Description
31	DPLL_UDEN	RW	0	PLL 参考时钟源 pll_262k_rclk 使能（由 pll_32k_rclk 八倍频生成）： 0:不使能 1:使能
31:12	-	-	-	预留
17:16	PSR_DL_SEL	RW	0	PSRAM 采样时钟延迟（只用于 IC 仿真）
15:14	LCD_CKSEL	RW	0	LCD 时钟选择，见图 4-1
13:12	PLL_ALNK1_DIV	RW	0	AUDIO LINK1 时钟分频，见图 4-1
11:10	PLL_ALNK1_SEL	RW	2’b11	AUDIO LINK1 时钟选择，见图 4-1
9:8	PLL_ALNK0_DIV	RW	0	AUDIO LINK0 时钟分频，见图 4-1
7:6	PLL_ALNK0_SEL	RW	2’b11	AUDIO LINK0 时钟选择，见图 4-1
5:2	PLL_SYS_DIV	RW	0	系统 PLL 分频选择，见图 4-1
1:0	PLL_SYS_SEL	RW	0	系统 PLL 选择，见图 4-1

4.SYS_DIV

Bit	Name	R/W	Default	Description
31:15	-	-	-	预留
14:12	SFC_DIV	RW	0	sfc_clk 分频，见图 4-2
11	-	-	-	预留
10:8	LSB_DIV	RW	0	lsb_clk 分频，见图 4-2
7:0	HSB_DIV	RW	0	hsb_clk 分频，见图 4-2

5.PLL_CON0

Bit	Name	R/W	Default	Description
31:30	PLL_REFSEL	RW	0	pll 参考时钟选择，见图 4-1
29:27	PLL_LPFR2	RW	0	PLL 模拟控制位
26:24	PLL_ICPS	RW	0	PLL 模拟控制位
23:22	PLL_PFDS	RW	0	PLL 模拟控制位
21:20	PLL_DIVS	RW	0	PLL 分频选择： 00/10:模拟整数分频 01:小数分频 11:小数分频、重复采样
19:18	PLL_MSEL	RW	0	PLL 模拟控制位
17:16	PLL_RSEL	RW	0	PLL 模拟控制位
15:12	PLL_TSEL	RW	0	PLL 测试功能选择
11	PLL_DSMS	RW	0	PLL 小数分频选择 0:CIFF 1:MASH111
10	PLL_TEST	RW	0	PLL 测试使能，需设置为‘0’
9	PLL_REFSEL	RW	0	PLL 模块参考时钟选择 0: btosc_clkp 1: pll_rclk（见图 4-1）
8:2	PLL_REFDS	RW	0	PLL 参考时钟分频值，n= PLL_REFDS + 2。见图 4-1。
1	SYSPLL_RST	RW	0	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	SYSPLL_EN	RW	0	PLL 模块使能 0: 关闭 1: 打开

6.PLL_CON1

Bit	Name	R/W	Default	Description
31	-	-	0	预留
30	PLL_CK_DAC_OE	RW	0	输出到 DAC 的使能

29	-	-	0	预留
28	PLL_CKOUT_D4P5_OE	RW	0	4.5 分频(107M)时钟输出使能
27	PLL_CKOUT_D3P5_OE	RW	0	3.5 分频(137M)时钟输出使能
26	PLL_CKOUT_D2P5_OE	RW	0	2.5 分频(192M)时钟输出使能
25	PLL_CKOUT_D1P5_OE	RW	0	1.5 分频(320M)时钟输出使能
24	PLL_CKOUT_D1_OE	RW	0	1 分频(480M)时钟输出使能
23:22	PLL_ REFDSEN	RW	0	PLL 参考时钟分频使能：（见图 4-1） 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）
21:20	PLL_LDO12A_S	RW	0	VCO 电源电压 LDO 输出选择
19	-	-	-	预留
18	PLL_TEST_EN	RW	0	测试使能
17:16	PLL_TEST_SEL	RW	0	测试功能选择
15	PLL_LDO_BYPASS	RW	0	测试所用
14:12	PLL_IVCOS	RW	0	测试所用
11:0	PLL_DS	RW	0	PLL 参考时钟反馈分频(相当倍频)，m= PLL_DS + 2。见图 4-1

7.PLL_INTF

Bit	Name	R/W	Default	Description
31:0	PLL_INTF	RW	0	PLL 模拟 DSM 小数分频

8.PLL_DMAX

Bit	Name	R/W	Default	Description
31:24	-	-	-	预留
23:0	PLL_DMAX	RW	0	PLL 模拟 DSM 小数分频 SSC 误差高位

9.PLL_DMIN

Bit	Name	R/W	Default	Description
31:24	-	-	-	预留
23:0	PLL_DMIN	RW	0	PLL 模拟 DSM 小数分频 SSC 误差低位

10.PLL_DSTP

Bit	Name	R/W	Default	Description
31:24	-	-	-	预留
23:0	PLL_DSTP	RW	0	PLL 模拟 DSM 小数分频 SSC 步数

第 5 章. 16 位定时器（TIMER16）

5.1. 概述

Timer16 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。

5.2. 模块寄存器

寄存器列表	TIMER0	TIMER1	TIMER2	TIMER3
Tx_CON	T0_CON	T1_CON	T2_CON	T3_CON
Tx_CNT	T0_CNT	T1_CNT	T2_CNT	T3_CNT
Tx_PRD	T0_PRD	T1_PRD	T2_PRD	T3_PRD
Tx_PWM	T0_PWM	T1_PWM	T2_PWM	T3_PWM

5.3. 寄存器说明

1.Tx_CON:timer x control register

Bit	Name	RW	Description
15	PND	r	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。
14	PCLR	w	PCLR:软件在此位写入‘1’将清除 PND 中断请求标志。
13-10	reserved	r	保留
9	PWM_INV	rw	PWM_INV:PWM 信号输出反向。
8	PWM_EN	rw	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[3:0]	rw	PSET3-0:预分频选择位 0000:预分频 1 0001:预分频 4 0010:预分频 16 0011:预分频 64 0100:预分频 1*2 0101:预分频 4*2 0110:预分频 16*2 0111:预分频 64*2

			1000:预分频 1*256 1001:预分频 4*256 1010:预分频 16*256 1011:预分频 64*256 1100:预分频 1*2*256 1101:预分频 4*2*256 1110:预分频 16*2*256 1111:预分频 64*2*256
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 IO 口信号作为 timer 的驱动源； 10:使用 OSC 时钟作为 timer 的驱动源； 11:使用 RC 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）。

2.Tx_CNT:timer x control register

Bit	Name	RW	Description
15-0	Tx_CNT	rw	Timer16 的计数寄存器

3.Tx_PR:timer x period register

Bit	Name	RW	Description
15-0	Tx_PR	rw	Timer16 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 0-65535-0 计数，不会和 Tx_PR 进行比较清 0。

4.Tx_PWM:timer x PWM register

Bit	Name	RW	Description
15-0	Tx_PWM	rw	Timer16 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 6 章. PWM LED 灯

6. 1. 概述

PWM LED 是通过脉冲宽度调节控制 LED 的亮度与亮灭的一个模块。放在 1.2V SYSVDD，不可以在软关机下工作。

PWM LED 模块可以配置两个周期大小不同的 PWM0、PWM1，分别控制 LED 的亮度（PWM0）与亮灭（PWM1）。

6. 2. 特性

1. 所有 IO 可用。
2. PDOWN 可用，SOFF 不可用。
3. 目前只放了一个模块，只支持单 IO 单 LED、单 IO 双 LED。
4. 驱动档位特性：

   4.1 PWM 推高电平: 高阻，上拉，输出 1 强驱 0，输出 1 强驱 1，输出 1 强驱 2，输出 1 强驱 3，保持，输出 1 强驱 2，输出 1 强驱 1，输出 1 强驱 0，上拉，高阻， 保持（循环以上步骤）

   4.2 PWM 推低电平: 高阻，下拉，输出 0 强驱 0，输出 0 强驱 1，输出 0 强驱 2，输出 0 强驱 3，保持，输出 0 强驱 2，输出 0 强驱 1，输出 0 强驱 0，下拉，高阻， 保持（循环以上步骤）

   4.3 最高驱动档位可设， 强驱及高阻保持时间可设， 驱动切换等待时间可设（所有等待时间为同样的 n 个 PWM0_CLK）
5. 推灯功能：

   5.1 固定亮度

   5.1.1 （单 LED）单闪 ；双闪（双闪的两个时间长度可调，两个间隔可调）；

   5.1.2 （双 LED）单闪 变色 单闪；

   双闪 变色 双闪（双闪的两个时间长度可调，两个间隔可调）；

   n 闪 变色 n 闪（n<7）；

5.2 呼吸灯

   5.2.1 （单 LED）灭灯时间长度可调，亮灯时间长度可调，亮灯快慢可调，灭灯快慢可调；

   5.2.2 （双 LED）（灭灯时间长度可调，亮灯快慢可调，灭灯快慢可调）呼吸 变色 呼吸；

   无灯---A 灯逐渐变到最亮---最亮变色---B 灯由最亮逐渐变到最暗---变色（无灯）---A 灯逐渐变到最亮...；

   无灯---A 灯呼吸---无灯---A 灯逐渐变到 最亮---最亮变色---B 灯由最亮逐渐变到最暗---无灯---B 灯呼吸---变色（无灯）---A 灯呼吸...；

6.3. 寄存器并接

1. PWM_ PRD_DIV :pwm1_clk 分频比控制器，12bit
   PWM_ PRD_DIV[11:0] = {P3_PWM_CON3[3:0],PWM_ PRD_DIVL[7:0]};
2. PWM_BRI_PRD[9:0]: 亮度周期，10bit
   PWM_BRI_PRD [9:0] = { P3_PWM_BRI_PRDH [1:0], P3_PWM_BRI_PRDL [7:0]};
3. P3_PWM_BRI_DUTY0 [9:0]: 推高电平点亮的灯，亮度占空比，10bit
   P3_PWM_BRI_DUTY0 [9:0] = { P3_PWM_BRI_PRD0H [1:0], P3_PWM_BRI_PRD0L [7:0]};
4. P3_PWM_BRI_DUTY1 [9:0]:推低电平点亮的灯，亮度占空比，10bit
   P3_PWM_BRI_DUTY1[9:0] = { P3_PWM_BRI_PRD1H [1:0], P3_PWM_BRI_PRD1L [7:0]};
5. 呼吸灯，灭灯延时计数器，16bit
   {PWM_DUTY3[7:0] ,PWM_DUTY2[7:0]};

6.4. 时钟控制

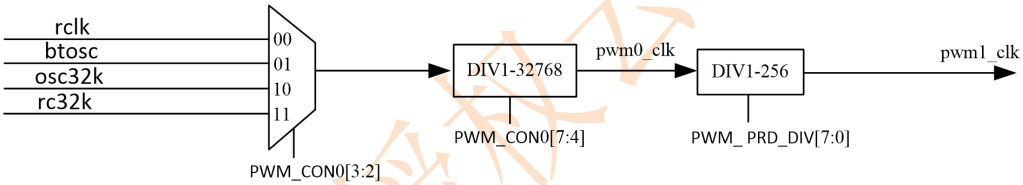


图 6-1. 时钟控制框图

其中 PWM0 由 pwm0_clk 驱动，周期 PRD0 由 PWM_BRI_PRD0/1 10bit 决定，有一个循环计数 bright_cnt 决定显示亮度。

其中 PWM1 由 pwm1_clk 驱动，周期 PRD1 最多 256 个时钟，有一个循环计数 counter_cnt。

特别地，当呼吸灯功能打开，PWM0、PWM1 分别控制 LED 的亮度级数变化及最亮和最暗保持的时间。

6.5. 二级亮灭控制

PWM1 可以控制 LED 在一个 PRD1 周期亮灭一次或两次。

下图以一个 LED 灯 PWM 电平举例：

输出高电平亮

单周期双闪

单灯

非呼吸灯模式

配置：

PWM_DUTY3_EN = 0;

PWM_DUTY2_EN = 1;

PWM_DUTY1_EN = 1;

PWM_DUTY0_EN = 1；
PWM1_PRD_SEL = 0；　　//PWM1 PRD 为 256 pwm1_clk
PWM1_INV = 0;//固定为 0 不变，亮灭不取反，
　　如下图，周期开始时是从灯灭开始
PWM0_INV = 0；//固定为 0 不变，灯亮度控制
OUT_LOGIC = 1；//固定为 1 不变，PWM 输出逻辑控制
BREATHE_EN = 0;
SHIFT_DUTY = 0;

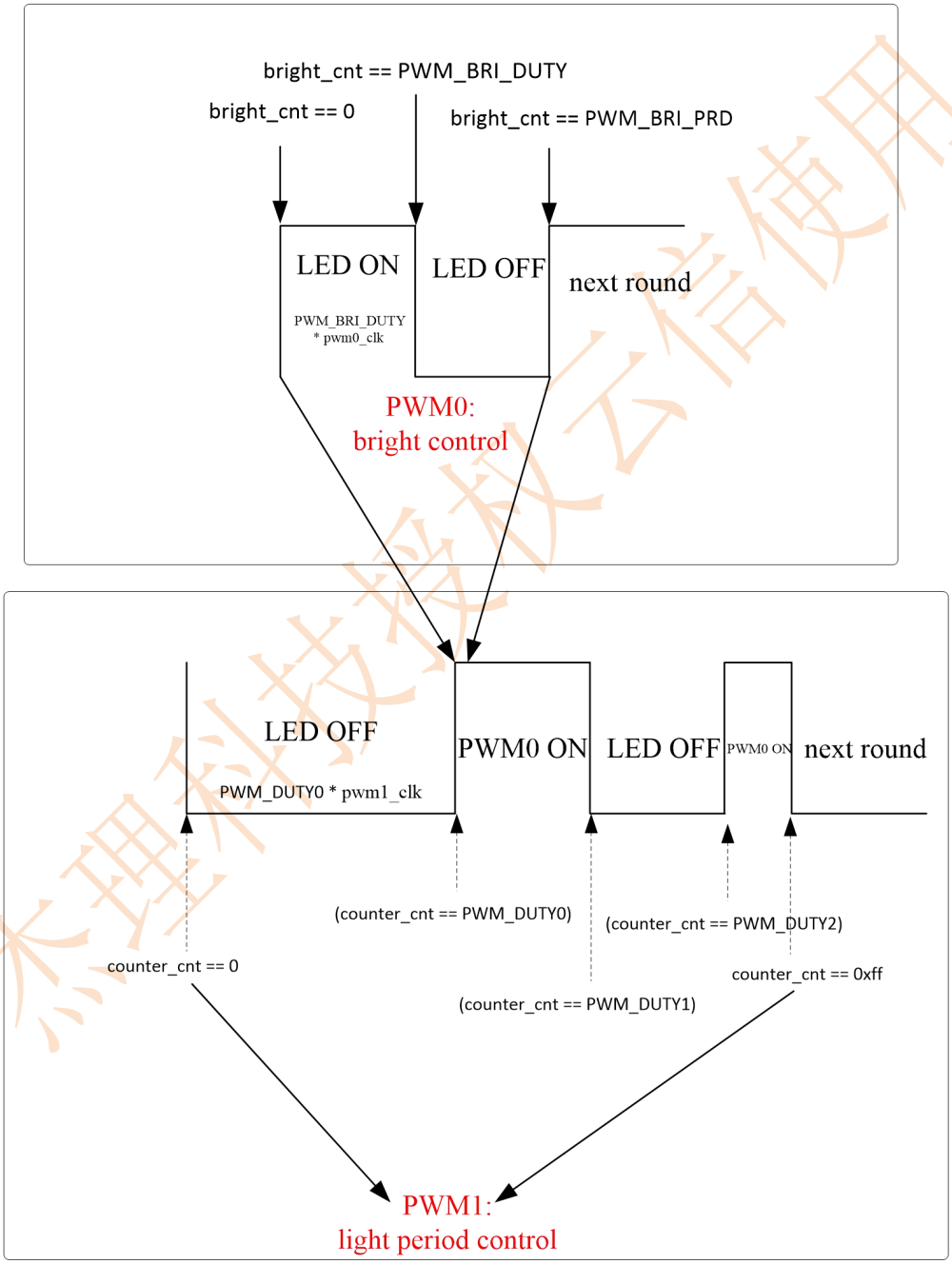
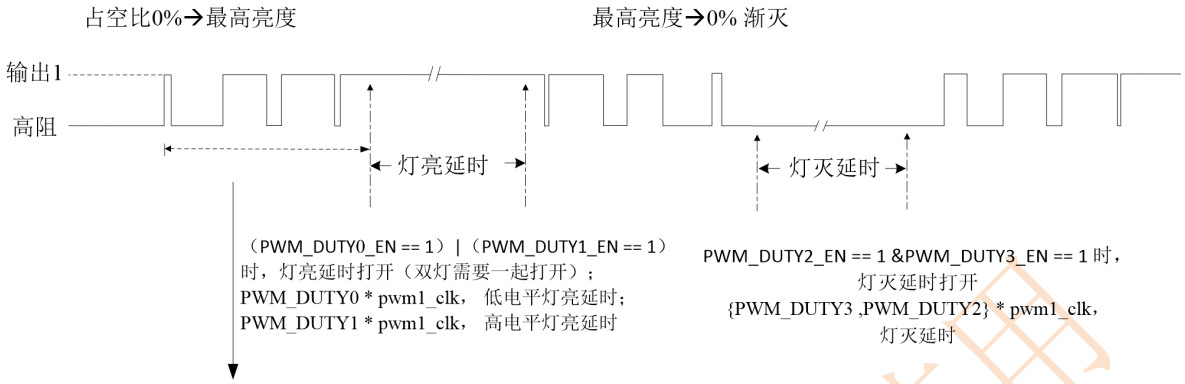


图 6-2. 二级亮灭控制框图

6.6. 呼吸灯控制



灯亮度变化的时间 T = pwm1_clk * PWM_BRI_DUTYx
（PWM_BRI_DUTY0是高电平，PWM_BRI_DUTY1是低电平）。
例如：pwm0_clk 周期25uS， pwm1_clk 周期5mS，

PWM_ PRD_DIV = 200； PWM_BRI_PRD = 200，
PWM_BRI_DUTY0 = 150，
PWM_BRI_DUTY1 = 170；

那么高电平灯从灭到亮度（150/200）时间
T0 = pwm1_clk * PWM_BRI_DUTY0= 750mS；
级数 N0 = T0/pwm0_clk*PWM_BRI_PRD=150
高电平灯灭灯时间一致。

那么低电平灯从灭到亮度（170/200）时间
T1 = pwm1_clk * PWM_BRI_DUTY1= 850mS；
级数 N1 = T1/pwm0_clk*PWM_BRI_PRD=170
低电平灯灭灯时间一致。

以上双灯亮灭转换时间不一致，可以用不同的高电平灯亮的延时、低电平灯亮的延时来平衡：

tmp = PWM_BRI_DUTY0 - PWM_BRI_DUTY1；
PWM_DUTY2 = PWM_DUTY0 + tmp;

6.7. 3.3v 系统控制寄存器说明

1.PWM_CON0: PWM control register 0 (8bit addressing)

Bit	Name	RW	Description
7-4	PWM_PSET	rw	PWM_PSET:PWM pre_scaler setting。设置 PWM0 的分频器除法因子 [1:0]: 00:div101:div4 10:div16 11:div64 [2] :0: x11: x2 [3] :0: x11: x256 计算方式：DIV = [1:0]的 divx 乘以 [2]倍数 乘以 [3]倍数。
3-2	PWM_SSEL	rw	PWM_SSEL:PWM 时钟源的选择。 00:rclk（有偏差 250KHz）

			01:btosc（准确 12/24MHz） 10:osc32k（准确 32KHz） 11:lrc32k（需校准 32KHz）
1	BREATHE_EN	rw	呼吸灯使能。 0:呼吸灯功能关闭 1:呼吸灯功能打开
0	PWM_EN	rw	PWM 使能。 0:模块总使能关闭 1:模块总使能打开

2.PWM_CON1: PWM control register 1 (8bit addressing)

Bit	Name	RW	Description
7	PWM_DUTY3_EN	rw	亮灭 DUTY 控制 3 开关/呼吸灯空周期延时开关高 8bit 0:亮灭 DUTY 控制 3 关闭（BREATHE_EN == 0 & PWM1_PRD_SEL == 0）/ 呼吸灯空周期延时关闭（BREATHE_EN == 1） 1:亮灭 DUTY 控制 3 开启（BREATHE_EN == 0 & PWM1_PRD_SEL == 0）/ 呼吸灯空周期延时开启（BREATHE_EN == 1）
6	PWM_DUTY2_EN	rw	亮灭 DUTY 控制 2 开关/呼吸灯空周期延时开关低 8bit 0:亮灭 DUTY 控制 2 关闭（BREATHE_EN == 0）/ 呼吸灯满周期延时关闭（BREATHE_EN == 1） 1:亮灭 DUTY 控制 2 开启（BREATHE_EN == 0）/ 呼吸灯满周期延时开启（BREATHE_EN == 1）
5	PWM_DUTY1_EN	rw	亮灭 DUTY 控制 1 开关/呼吸灯最高亮度延时开关(高电平灯) 0:亮灭 DUTY 控制 1 关闭（BREATHE_EN == 0）/ 呼吸灯空周期延时关闭（BREATHE_EN == 1） 1:亮灭 DUTY 控制 1 开启（BREATHE_EN == 0）/ 呼吸灯空周期延时开启（BREATHE_EN == 1）
4	PWM_DUTY0_EN	rw	PWM_DUTY0_EN:亮灭 DUTY 控制 0 开关/呼吸灯最高亮度延时开关 0:亮灭 DUTY 控制 0 关闭（BREATHE_EN == 0）/ 呼吸灯满周期延时关闭（BREATHE_EN == 1） 1:亮灭 DUTY 控制 0 开启（BREATHE_EN == 0）/ 呼吸灯满周期延时开启（BREATHE_EN == 1）
3	PWM1_PRD_SEL	rw	PWM1 亮灭周期选择: 0:PWM1 周期为 0xff 1:PWM1 周期为 PWM_DUTY3
2	PWM_OUTPUT_INV	rw	输出电平取反（周期变色 SHIFT_DUTY==0 时可用） 0: 不取反。 1: 取反
1	PWM1_INV	rw	PWM1 输出的 LED 亮灭波形取反（默认由 0 开始），可以控制亮灭周期是从灯亮开始还是灯灭开始，软件常设为 0。

			0:PWM1 输出的 LED 亮度波形不取反（灭灯开始的 PWM1 周期） 1:PWM1 输出的 LED 亮度波形取反（亮灯开始的 PWM1 周期）
0	PWM0_INV	rw	PWM0 输出的 LED 亮度波形取反（默认由 1 开始），软件常设为 0 0:PWM0 输出的 LED 亮度波形不取反 1:PWM0 输出的 LED 亮度波形取反

3.PWM_CON2: PWM control register 2 (8bit addressing)

Bit	Name	RW	Description
7-4	SHIFT_DUTY	rw	LED 周期变色（输出电平取反），非呼吸灯 0000:不变色 0001:1 个周期变色 … 1111:15 个周期变色 LED 周期变色，呼吸灯 PWM_CON2[4]常设为 0 0000:不变色 0010:1 个周期变色（1 个周期为:最暗到最亮，再到最暗） 0100:2 个周期变色 … 1110:7 个周期变色
3-2	IO_DRV_SHIFT_CLK	rw	输出驱动切换时钟个数（PWM0_CLK） 00: 1 01: 2 10: 4 11: 8
1-0	IO_MAX_DRV	rw	IO 最大输出强度 00: 2.4mA（8mA mos + 120Ω res） 01: 8mA（8mA mos ） 10: 18.4mA（24mA mos + 120Ω res） 11: 24mA（24mA mos ）

4.PWM_CON3: PWM control register 3 (8bit addressing)

Bit	Name	RW	Description
7	PENDING	r	亮灭周期结束，呼吸灯周期结束，pengding 置 1
6	CLEAR PENDING	w	清 pending 0:无效 1:清 pending
5	IE	rw	PWM LED pending 中断使能。
4	OUT_LOGIC	rw	输出逻辑， 软件常设为 1

			0:LED 在低电平时为亮（PWM0	PWM1）。 1:LED 在高电平时为亮（PWM0 & PWM1）。
3-0	PWM_PRD_DIVH	rw	pwm1 clk 分频比控制高位 PWM_PRD_DIV[11:8]

5.PWM_BRI_PRDL: PWM0 brightness period register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_BRI_PRD[7:0]	rw	亮度周期低 7bit

6.PWM_BRI_PRDH: PWM0 brightness period register (8bit addressing)

Bit	Name	RW	Description
1-0	PWM_BRI_PRD[9:8]	rw	亮度周期高 2bit

7.PWM_BRI_DUTY0L: PWM0 brightness duty register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_BRI_DUTY0 [7:0]	rw	亮度控制低 7bit，例： 高电平灯亮 PWM_BRI_DUTY0 == 0； 灯最暗 PWM_BRI_DUTY0 == 1； 灯亮度增强一档 … PWM_BRI_DUTY0 == PWM_BRI_PRD； 灯最亮

8.PWM_BRI_DUTY0H: PWM0 brightness duty register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_BRI_DUTY0 [9:8]	rw	亮度控制高 2bit，例： 高电平灯亮

9.PWM_BRI_DUTY1L: PWM0 brightness duty register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_BRI_DUTY1 [7:0]	rw	亮度控制低 7bit，例： 低电平灯亮 PWM_BRI_DUTY0 == 0； 灯最暗 PWM_BRI_DUTY0 == 1； 灯亮度增强一档 … PWM_BRI_DUTY0 == PWM_BRI_PRD； 灯最亮

10.PWM_BRI_DUTY1H: PWM0 brightness duty register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_BRI_DUTY1 [9:8]	rw	亮度控制高 2bit，例： 低电平灯亮

11.PWM_ PRD_DIVL: PWM1 period divider register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_PRD_DIV[7:0]	rw	PWM 时钟的分频因子设置。

PWM_ PRD_DIV :pwm1_clk 分频比控制器，12bit

PWM_ PRD_DIV[11:0] = {P3_PWM_CON3[3:0],PWM_PRD_DIVL[7:0]};

0x000:div = 1;

0x001:div = 2;

… …

0xfff:div = 4096;

12.PWM_DUTY0: PWM duty0 register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_DUTY0	rw	BREATHE_EN == 0 & PWM_DUTY0_EN == 1: 亮灭 DUTY 控制 1 BREATHE_EN == 1 & PWM_DUTY0_EN == 1: 呼吸灯最高亮度延时(低电平灯)

13.PWM_DUTY1: PWM duty1 register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_DUTY1	rw	BREATHE_EN == 0 & PWM_DUTY1_EN == 1: 亮灭 DUTY 控制 1 BREATHE_EN == 1 & PWM_DUTY1_EN == 1: 呼吸灯最高亮度延时(高电平灯)

14.PWM_DUTY2: PWM duty2 register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_DUTY2	rw	BREATHE_EN == 0 & PWM_DUTY2_EN == 1: 亮灭 DUTY 控制 2 BREATHE_EN == 1 & PWM_DUTY2_EN == 1: 呼吸灯灭灯延时低 8bit

15.PWM_DUTY3: PWM duty3 register (8bit addressing)

Bit	Name	RW	Description
7-0	PWM_DUTY3	rw	BREATHE_EN == 0 & PWM1_PRD_SEL == 0 & PWM_DUTY3_EN == 1: 亮灭 DUTY 控制 3 BREATHE_EN == 0 & PWM1_PRD_SEL == 1: 亮灭 PWM1 周期 PWM1_PRD BREATHE_EN == 1 & PWM_DUTY3_EN == 1: 呼吸灯灭灯延时高 8bit

6.8. 例程

```
//-----------------------------------------------//
// led define for breathe demo
//-----------------------------------------------//
#define led_con0_init                                   \
        /* PWM0 PDIV          4bit RW   */ ( 1<<4 )   | \
        /* CLKSEL             2bit RW   */ ( 3<<2 ) /* 0:RC250k 1:BTOSC 2:OSL 3:LRC */ | \
        /* BREATHE            1bit RW   */ ( 1<<1 )   | \
        /* EN                 1bit RW   */ ( 1<<0 )

         //PWM0 PDIV     //pre-scaler setting,     [1:0]: 00:div1     01:div4     10:div16     11:div64
                         //                        [2]:     0: x1          1: x2
                         //                        [3]:     0: x1          1: x256

#define led_con1_init                                   \
        /* PWM_DUTY3_EN       1bit RW   */ ( 1<<7 )   | \
        /* PWM_DUTY2_EN       1bit RW   */ ( 1<<6 )   | \
        /* PWM_DUTY1_EN       1bit RW   */ ( 1<<5 )   | \
        /* PWM_DUTY0_EN       1bit RW   */ ( 1<<4 )   | \
        /* PWM1_PRD_SEL       1bit RW   */ ( 0<<3 ) /* 0 pwm1 round end by 0xff; 1 pwm1 round end by pwm_duty3
*/ | \
        /* PWM_OUT_INV        1bit RW   */ ( 0<<2 ) /* using in shift == 0 */ | \
        /* PWM_EDGE1          1bit RW   */ ( 0<<1 )   | \
        /* PWM_EDGE0          1bit RW   */ ( 0<<0 )

#define led_con2_init                                   \
        /* SHIFT PRD          4bit RW   */ ( 2<<4 )   | \
        /* DRVING CYCLE       2bit RW   */ ( 2<<2 )   | \
        /* DRVING MAX         2bit RW   */ ( 2<<0 )

#define led_con3_init                                   \
        /* PWM PND            1bit RO   */ ( 0<<7 )   | \
```

```
        /* PWM CLR PND          1bit WO    */ ( 1<<6 )   | \
        /* PWM IE               1bit RW    */ ( 1<<5 )   | \
        /* OUT_LOGIC            1bit RW    */ ( 1<<4 )   | \
        /* PWM1 DIVH            4bit RW    */ ( 2<<0 )
void led_init (void)
{
    p33_tx_1byte(P3_LRC_CON0,     0x61);
    p33_tx_1byte(P3_LRC_CON1,     0xaa);

    p33_tx_1byte(P3_PWM_BRI_PRDL,     0x40);
    p33_tx_1byte(P3_PWM_BRI_PRDH,     0x01);
    p33_tx_1byte(P3_PWM_BRI_DUTY0L, 0x71);
    p33_tx_1byte(P3_PWM_BRI_DUTY0H, 0x00);
    p33_tx_1byte(P3_PWM_BRI_DUTY1L, 0x90);
    p33_tx_1byte(P3_PWM_BRI_DUTY1H, 0x00);

    p33_tx_1byte(P3_PWM_PRD_DIVL,     0x40);

    p33_tx_1byte(P3_PWM_DUTY0,     0x01);
    p33_tx_1byte(P3_PWM_DUTY1,     0x09);
    p33_tx_1byte(P3_PWM_DUTY2,     0xb0);
    p33_tx_1byte(P3_PWM_DUTY3,     0x02);

    p33_tx_1byte(P3_PWM_CON3,     led_con3_init);
    p33_tx_1byte(P3_PWM_CON2,     led_con2_init);
    p33_tx_1byte(P3_PWM_CON1,     led_con1_init);
     p33_tx_1byte(P3_PWM_CON0,     led_con0_init);
    p33_or_1byte(P3_PWM_CON3,     BIT(6));//clear pending
}
```

第 7 章. 看门狗定时器（WDT）

7.1. 概述

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）

7.2. 寄存器说明

1.P3_WDT_CON: Watchdog control register

Bit	Name	RW	Description
7	PND	r	wdt 中断请求标志，当 WDRMD 设置为 1 时，WDT 溢出会将此位置 1
6	CPND	w	写 1 清除中断标记位
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 注意：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

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 8 章. 引脚唤醒（Port_Wakeup）

8. 1. 概述

PMU 结构上独立于系统，可以在系统休眠或者掉电的情况下继续工作，提供多种唤醒功能。P33 wkup 示意图：

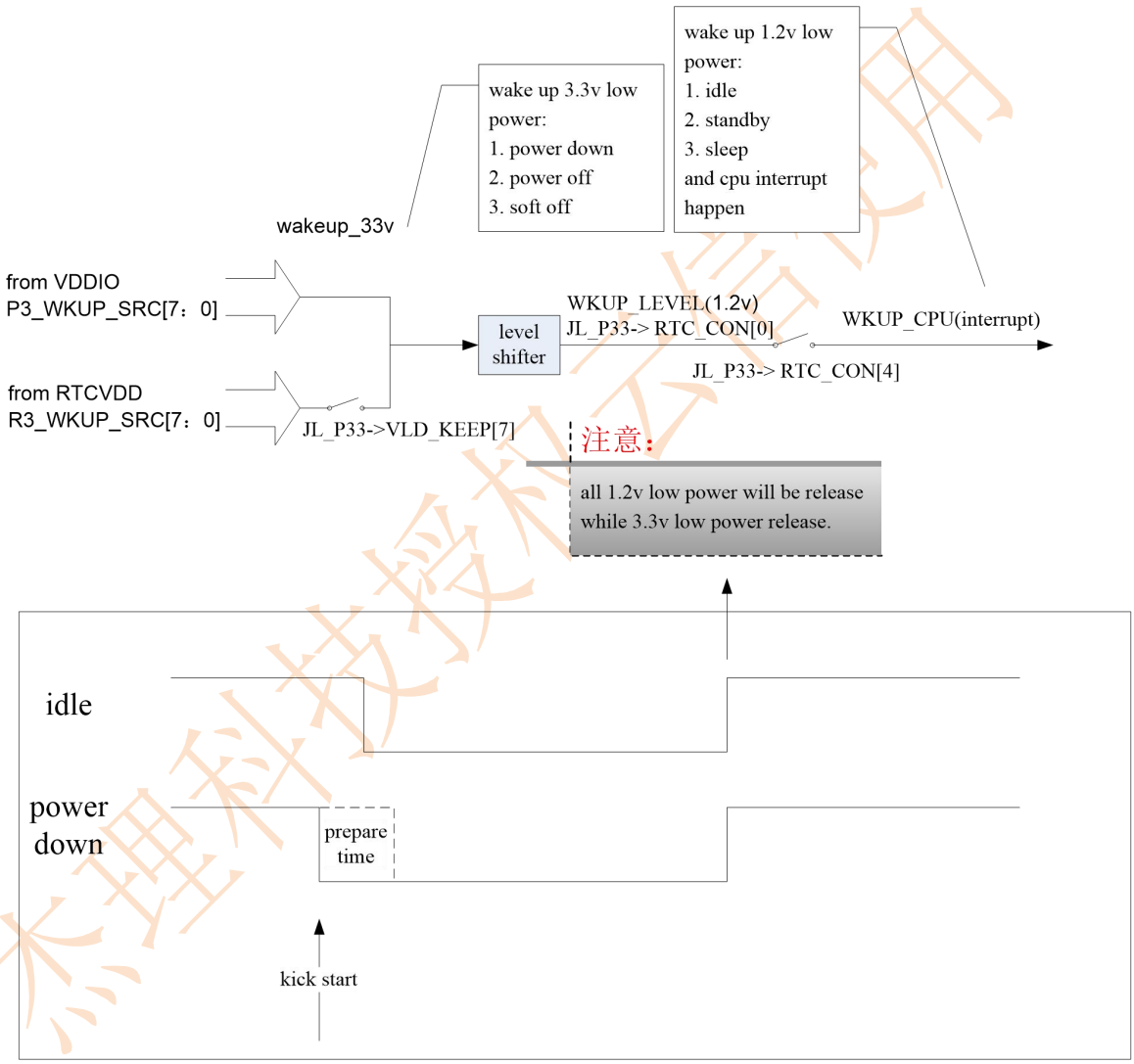


图 8-1. WKUP 示意图

8.2. 寄存器说明

8.2.1. 控制寄存器(1.2V 主系统，普通 SFR 控制)

1.1.JL_P33-> RTC_CON:RTC control registe， 16bit addressing

Bit	Name	RW	Description
15	WKUP_CPND	w	
14	X512_CPND	w	
13	X2_CPND	w	
12	reserved	r	
11-10	reserved	rw	
9	reserved	r	
8	reserved	rw	
7	WKUP_PND	r	
6	X512_PND	r	
5	X2_PND	r	
4	LEVEL_WKUP_EN	r	LEVEL_WKUP_EN，允许 P33 wakeup 唤醒发生 --高电平触发（WKUP_LEVEL）
3	EDGE_WKUP_EN	rw	
2	X512_EN	rw	
1	X2_EN	rw	
0	WKUP_LEVEL	r	1:有 wake UP 事件正在发生 0:无 wake UP 事件发生

8.2.2. 控制寄存器（3.3V 电源部分，P33 接口控制）

1.R3_WKUP_SRC（RTCVDD 电源域）参考文档“RTC_WKUP.doc”

Bit	Name	Default	RW	Description
7	1Hz_WKUP	0	r	1Hz 时基唤醒
6	2Hz_WKUP	0	r	2Hz 秒时基唤醒
5	64Hz_WKUP	0	r	64Hz 时基唤醒
4	512Hz_WKUP	0	r	512Hz 秒时基唤醒
3	PORT_WKUP	0	r	PR 口边沿唤醒（请参考 RTC_IO.doc）

2	-	0	-	预留
1	POR_FLAG_WKUP	0	r	RTCVDD 上电唤醒，与 R3_RST_CON[7]是同 1bit
0	ALM_WKUP	0	r	闹钟唤醒

2.P3_WKUP_SRC（VDDIO 电源域）

Bit	Name	Default	RW	Description
7	WDT_INT	0	r	WDT 中断唤醒
6	CHG_WKUP	0	r	充电信号检测唤醒（P3_CHG_WKUP）
5	reserved	0	r	0
4	VDDIO LVD WKUP	0	r	VDDIO LVD 低电触发芯片唤醒(P3_VLVD_CON)
3	WKUP_SUB	0	r	LDO5V 充电口相关唤醒（P3_WKUP_SUB）
2	reserved	0	r	0
1	EDGE_WKUP	0	r	PORT 口边沿检测唤醒(P3_WKUP_EN) 参考文档“P33_IO.doc”
0	PCNT OVF	0	r	PORT 口边沿计数机溢出唤醒(P3_PCNT_CON) 参考文档“P33_IO.doc”

3.P3_WKUP_SUB（VDDIO 电源域）

Bit	Name	Default	RW	Description
7-2	reserved	0	r	0
1	LDO5V_WKUP	0	r	LDO5V 充电插入唤醒（P3_LDO5V_CON）
0	LVCMP_WKUP	0	r	LDO5V 充电插入唤醒（P3_LVCMP_CON）

4.P3_LDO5V_CON:LDO5V 插入控制（VDDIO 电源域）

Bit	Name	Default	RW	Description
7	LDO5V_PND	x	r	LDO5V 边沿检测 panding
6	LDO5V_PND_CLR	0	w	LDO5V 边沿检测 panding clear， 1:清 LDO5V_PND， 0:无效
5	LDO5V_FLT_DET	x	r	LDO5V 电平(经过 fliter，配合 LDO5V_PND 使用) 0:LDO5V 没有插入 1:LDO5V 插入
4	LDO5V_DET	x	r	LDO5V 电平(原始信号，可配合长按复位使用) 0:LDO5V 没有插入 1:LDO5V 插入

3	LDO5V_EDGE_WKUP_EN	0	rw	LDO5V 检测结果边沿唤醒使能（LDO5V_PND）
2	LDO5V_LEVEL_WKUP_EN	0	rw	LDO5V 检测电平唤醒使能（LDO5V_DET）
1	LDO5V_EDGE	0	rw	LDO5V 检测边沿选择 0:上升沿 (1.0V) 1:下降沿 (0.6V)
0	LDO5V_EN	0	rw	LDO5V 边沿检测使能

5.P3_LVCMP_CON:LDO5V 与 VBAT 比较控制 （VDDIO 电源域）

Bit	Name	Default	RW	Description
7	LVCMP_PND	x	r	LVCMP 边沿检测 panding
6	LVCMP_PND_CLR	0	w	LVCMP 边沿检测 panding clear, 1:清 LVCMP_PND, 0:无效
5	LVCMP_FLT_DET	x	r	LDO5V 电平(经过 fliter，经过 LVCMP_SEL，配合 LDO5V_PND 使用) 0:LDO5V 电压低于 VBAT 1:LDO5V 电压高于 VBAT
4	LVCMP_SEL	x	rw	LDO5V VBAT 斯密特选择 0:LDO5V>VDD50 时，LVCMP_DET 为 1 1:LDO5V>VBAT 时，LVCMP_DET 为 1
3	LVCMP_EDGE_WKUP_EN	0	rw	LVCMP 检测结果边沿唤醒使能（LVCMP_PND）
2	LVCMP_LEVEL_WKUP_EN	0	rw	LVCMP 检测电平唤醒使能（LVCMP_DET）
1	LVCMP_EDGE	0	rw	LDO5V 检测边沿选择 0:上升沿 1:下降沿
0	LVCMP_EN	0	rw	LVCMP 边沿检测使能

6.P3_CHG_READ:模拟充电信号，需要软件滤波后才能获取正确的值（VDDIO 电源域）（AC695N 不可读）

Bit	Name	Default	RW	Description
7-4	reserved	0	r	0
3	reserved	0	r	0
2	CHG_VBGOK	0	r	
1	reserved	0	r	0
0	CHG_FULL	0	r	充电满信号

7.P3_CHG_WKUP:LDO5V 与 VBAT 比较控制 （VDDIO 电源域）（AC695N 不可读）

备注：CHG_DET：CHG 检测电平，将 P3_CHG_READ 中信号通过 CHG_SIG_SEL 选取出来的一个信号

Bit	Name	Default	RW	Description
7	CHG_PND	x	r	CHG 边沿检测 panding
6	CHG_PND_CLR	0	w	CHG 边沿检测 panding clear， 1:清 CHG_PND 0:无效
5	CHG_EDGE	0	rw	CHG 检测边沿选择 0:上升沿 1:下降沿
4	CHG_EN	0	rw	CHG 边沿检测使能
3-2	CHG_SIG_SEL	0	rw	CHG 检测信号选择，选取结果为 CHG_DET 0:P3_CHG_READ[0] 1:P3_CHG_READ[1] 2:P3_CHG_READ[2] 3:P3_CHG_READ[3]
1	CHG_EDGE_WKUP_EN	0	rw	CHG 检测结果边沿唤醒使能（CHG_PND）
0	CHG_LEVEL_WKUP_EN	0	rw	CHG 检测电平唤醒使能（CHG_DET）

第 9 章. 集成电路总线（IIC）

9.1. 概述

IIC 接口是一个标准的遵守 IIC 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。

IIC 接口支持主机和从机两种模式

主机：1）IIC 接口时钟由本机产生，提供给片外 IIC 设备使用；

2）IIC 接口的驱动时钟可配置，频率范围为

F = Fsystem/(（1+IIC_BAUD）*2)2）+ 电阻上拉的时间

上拉电阻越大，频率越低。

从机：1）IIC 接口时钟由片外 IIC 设备产生，提供给本机使用；

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，硬件会自动回应（ack 位上为“0”）；

3）发送时，当前这包的数据传输 ACK 后，不支持接下来直接 restart 或者 end；

4）接收时，当前这包的数据传输 ACK 后，不支持接下来直接 restart；

IIC 接口使用下降沿更新数据，上升沿采集数据。

IIC 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 IIC_BUF 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。

9.2. 寄存器说明

1.IIC_CON0: IIC control register0 (16bit addressing)

Bit	Name	RW	Description
15	PND	ro	PND：中断请求标志，当完成 1Byte 加 1 应答位的传输后会被硬件置“1”；清除此标志用软件方式：向 PCLR 写“1”。
14	PCLR	wo	软件在此位写入‘1’将清除 PND 中断请求标志。
13	END_PND	ro	结束位标志。
12	END_PND_CLR	wo	软件在此位写入‘1’将清除 END_PND 标志
11	reserved	rw	预留
10	END_PND_IE	rw	结束位中断使能
9	IIC_ISEL	rw	iic_cki 和 iic_di 输入选择：

			0:选择 IO 直接输入 1:选择 IO 经过 filter 后再输入
8	IE	rw	IIC 中断使能。 0：禁止 SPI 中断 1：允许中断允许
7	RD_ACK	ro	IIC 中的应答位标志。 0：应答 1：不应答
6	WR_ACK	rw	当作为接收方（主机或者从机模式）对应答位的设置。 0：应答 1：不应答 当作为发送方时，硬件自动置“1”释放总线。
5	M_SET_RSTART	wo	主机重新开始设置位，硬件自动清“0”。 每次传输的第一包数据，只要总线上是上拉的，硬件自动加 start 位，过后的每包数据传输接不接 start 位可配 0：下一包数据传输不接重新开始位 1：下一包数据传输紧接重新开始位
4	M_SET_END	wo	主机结束设置位，硬件自动清“0”。 0：当前这一包数据传输后不接结束位 1：当前这一包数据传输后紧接结束位
3	DAT_DIR	rw	IIC 接口传输方向。 0：发送数据 1：接收数据
2	N_CFG_DONE	wo	每次传输前，传输配置完成标志。（configuration done） 每次传输前，需要对 IIC 接口进行所需的配置（DAT_DIR、M_SET_END、IIC_BUF、WR_ACK 等）之后再将此 bit 置“1”，硬件自动清“0”。 0：配置中 1：配置完成 配置完毕过后才可将此 bit 置“1”，建议配置完所需设置后延时一段时间才将其置“1”。
1	SLAVE	rw	IIC 接口模式选择。 0：主机模式 1：从机模式
0	EN	rw	IIC 接口使能。 0：关闭 IIC 接口 1：打开 IIC 接口

2.IIC_CON1:　IIC control register1 (16bit addressing)

Bit	Name	RW	Description
15	SPND	ro	SPND:起始位标志

14	SPND_CLR	wo	SPND_CLR:起始位清除，写 1 清除
13	SI_MODE	rw	SI_MODE:从机忽略总线活动模式：（1 有效）当 IIC 处于从机模式，访问我们的设备地址与我们设置的地址（IIC_BAUD 高 7 位）不匹配，则硬件不作任何响应，直到访问我们的设备地址匹配，硬件才运作。
12-0	reserved	rw	预留

3.IIC_BAUD:(16bit addressing，write only)

Bit	Name	RW	Description
15-0	IIC_BAUD	rw	

1) IIC 接口为主机时，IIC_BAUD 做为时钟设置寄存器。

2)频率范围为 F = Fsystem/(（1+IIC_BAUD）*2) + 电阻上拉的时间。上拉电阻越大，频率越低。

3) IIC 接口为从机时，IIC_BAUD 做为从机地址(7bits)设置寄存器。

从机地址 ＝IIC_BAUD[7:1]

当地址匹配时硬件自动应答（ACK 位为“0”）

4.IIC_BUF: (8bit addressing，write and read)

Bit	Name	RW	Description
7-0	IIC_BUF	rw	

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。

第 10 章. 串行通信（UART）

10. 1. 概述

UART0、 UART1 支持接收带循环 Buffer 的 DMA 模式和普通模式。

UART2 只支持普通模式，不支持 DMA 模式。

UART0 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。

10. 2. 控制寄存器

寄存器列表	UART0	UART1	UART2
UTx_CON(0)	UT0_CON0	UT1_CON	UT2_CON
UTx_CON1	UT0_CON1	UT1_CON1	
UTx_BAUD	UT0_BAUD	UT1_BAUD	UT2_BAUD
UTx_BUF	UT0_BUF	UT1_BUF	UT2_BUF
UTx_TXADR	UT0_TXADR	UT1_TXADR	
UTx_TXCNT	UT0_TXCNT	UT1_TXCNT	
UTx_RXCNT	UT0_RXCNT	UT1_RXCNT	
UTx_RXSADR	UT0_RXSADR	UT1_RXSADR	
UTx_RXEADR	UT0_RXEADR	UT1_RXEADR	
UTx_HRXCNT	UT0_HRXCNT	UT1_HRXCNT	
UTx_OTCNT	UT0_OTCNT	UT1_OTCNT	

10. 3. 寄存器说明

1.UTx_CON(0): uart x control register (16bit addressing).

Bit	Name	RW	Description
15	TPND	r	TPND:TX Pending
14	RPND	r	RPND:RX Pending& Dma_Wr_Buf_Empty，数据接收不完 Pending 不会为 1
13	CLRTPND	r	CLRTPND:清空 TX Pending
12	CLRRPND	r	CLRRPND:清空 RX Pending

11	OTPND	r	（*）OTPND:OverTime Pending
10	CLR_OTPND	r	（*）CLR_OTPND: 清空 OTPND
9	TB8	rw	TB8:9Bit 模式时，TX 发送的第 9 位
8	RB8	r	RB8:9Bit 模式时。RX 接收到的第 9 位
7	RDC	w	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	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:不允许； 1:允许。
4	DIVS	rw	DIVS:前 3 分频选择，0 为 4 分频，1 为 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
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
1	M9EN	rw	M9EN:9bit 模式使能
0	UTEN	rw	UTEN:UART 模块使能

（*）:只有 UART0/1 支持

2.UTx_CON1: uart x control register1 (16bit addressing).

Bit	Name	RW	Description
15	CTSPND	rw	CTSPND:CTS 中断 pending
14	CLR_CTSPND	wo	CLR_CTSPND:清楚 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.UTx_BAUD: uart x baudrate register (16bit addressing, Write Only).

Bit	Name	RW	Description
15-0	UTx_BAUD	wo	

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)

4.UTx_BUF: uart x data buffer register (8bit addressing).

Bit	Name	RW	Description
15-0	UT0BUF	wo	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。

5.UTx_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

6.UTx_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_CON[15]）。

7.UTx_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_CON[14]）。

8.UTx_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

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

Bit	Name	RW	Description
24-0	UTx_RXEADR	wo	DMA 接收数据时，循环 buffer 的结束地址

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

Bit	Name	RW	Description
31-0	UTx_HRXCNT	ro	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。

11.UTx_OTCNT:uart x OverTime count(32bit addressing, Write Only).

Bit	Name	RW	Description
31-0	UTx_OTCNT	wo	

设置串口设备在等待多久 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

第 11 章. 串行外设接口（SPI）

11.1. 特征

- 传输数据以 Byte（8bit）为最小单位；
- 支持主机模式和从机模式操作。
- 单项传输支持 1bit data、2bit data 和 4bit data 模式，双向传输只支持 1bit data 模式
- 支持 CPU 方式操作和 DMA 方式操作。

11.2. 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。

（1）主/从机模式：

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；在主机模式下，SPI 接口的驱动时钟可配置，范围为 系统时钟~系统时钟/256

从机：SPI 接口时钟由片外 SPI 设备产生，提供给本机使用；在从机模式下，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。

（2）传输模式：

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。

（3）数据模式：

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。

（注：SPI0 支持 1bit，2bit 和 4bit 模式，SPI1/SPI2 只支持 1bit 模式）

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。

（4）数据发送与接收：

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 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。

（5）操作模式：

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 SPIADR 的动作将启动一次 DMA 传输。

11.3. 传输波形

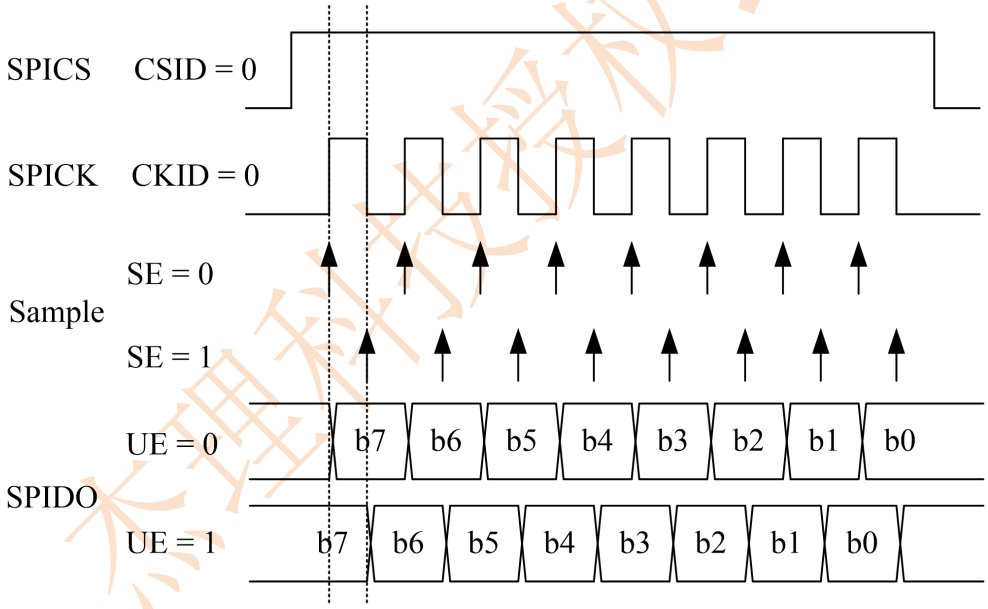


图 11-1. SPICS 空闲时为低电平，SPICLK 空闲时为低电平的波形图

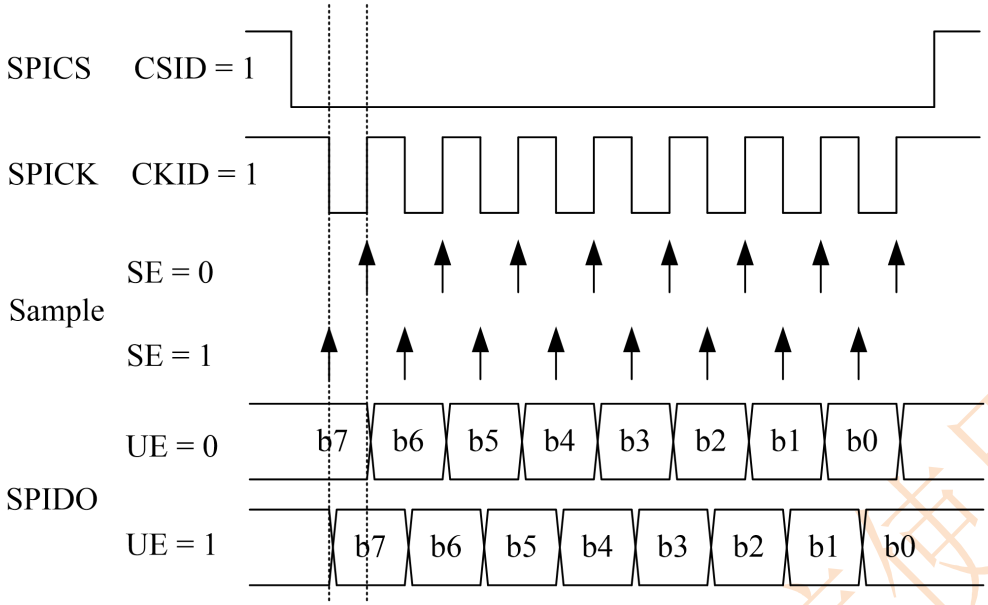


图 11-2. SPI CS 空闲时为高电平，SPI CLK 空闲时为高电平的波形图

11.4. 寄存器说明

寄存器列表	SPI0	SPI1	SPI2
SPIx_CON	SPI0_CON	SPI1_CON	SPI2_CON
SPIx_BUF	SPI0_BUF	SPI1_BUF	SPI2_BUF
SPIx_BAUD	SPI0_BAUD	SPI1_BAUD	SPI2_BAUD
SPIx_ADR	SPI0_ADR	SPI1_ADR	SPI2_ADR
SPIx_CNT	SPI0_CNT	SPI1_CNT	SPI2_CNT

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。 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	软件在此位写入‘1’将清除 PND 中断请求标志。
13	IE	rw	SPI 中断使能 0：禁止 SPI 中断 1：允许中断允许

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
11-10	DATW	r	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 *（注：SPI0 支持 1bit，2bit 和 4bit 模式，SPI1/SPI2 只支持 1bit 模式）*
9	reserved	r	0
8	reserved	r	0
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	BIDIR	rw	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	SPICS 信号使能 0：不使用 SPICS 信号 1：使用 SPICS 信号
1	SLAVE	rw	从机模式
0	SPIE	rw	SPI 接口使能 0：关闭 SPI 接口 1：打开 SPI 接口

2.SPIx_BAUD: SPI baudrate setting register (8bit addressing, write only).

Bit	Name	RW	Description
7-0	SPI_BAUD	rw	

SPI 主机时钟设置寄存器

SPICK = system clock / (SPIBAUD + 1)

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

Bit	Name	RW	Description
7-0	SPI_BUF	rw	

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。

4.SPI_ADR: SPI DMA start address register (25bit addressing, write only).

Bit	Name	RW	Description
24-0	SPI_ADR	rw	

SPI DMA 起始地址寄存器，只写，读出为不确定值

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

Bit	Name	RW	Description
15-0	SPI_CNT	rw	

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。

第 12 章. PAP

12.1. 特征

- 工作于主机模式。
- 支持 8bit 的数据宽带
- 支持 CPU 以及 DMA 操作模式。

12.2. 概述

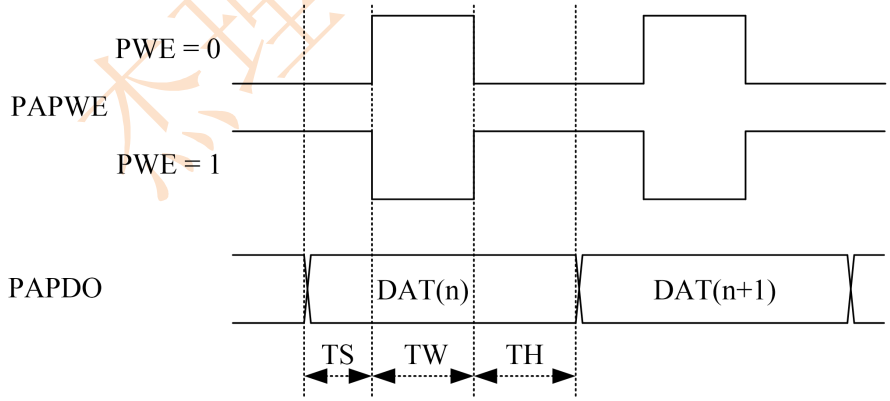
PAP 接口（Parallel Active Port）是一个并行主动数据接口，它工作于主动模式，支持 8bit 数据宽度，具有读/写使能信号，并可支持 DMA 方式发送/接收数据。

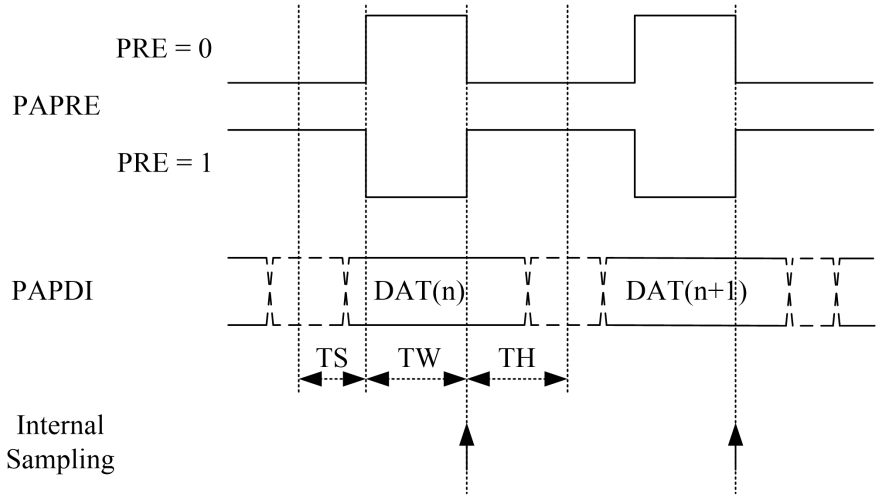
PAP 传输支持由 CPU 直接驱动，写 PAPBUF 的动作将启动一次传输；PAP 传输也支持 DMA 操作，每次 DMA 操作支持的数据量为 1-65535Byte，写 PAPCNT 的动作将启动一次传输。

PAP 在 DMA 发送数据（写）时支持普通模式和数据扩展模式。普通模式是指：PAP 接口直接将原始数据发送出去。数据扩展模式是指，PAP 接口可预设两个 16 位的数值 DAT0 和 DAT1，在发送过程中，硬件依次检查原始数据（1Byte）的每一 bit（从 LSB 到 MSB，或从 MSB 到 LSB），若该 bit 为 1，则发送预设值 DAT1，否则发送预设值 DAT0。此状态下， DAT0/DAT1 先后分两次写出。

注：PAP 在接收数据（读）时只支持普通模式，请勿在读状态下使能数据扩展模式，否则会导致不可预计的错误。

12.3. 传输波形





12.4. 寄存器说明

1.PAPCON: PAP contrl register 0(24bit addressing)

Bit	Name	RW	Description
23-19	reserved	r	预留
18	PAPIE	rw	PAPIE:中断使能 0：关闭，不允许 PAP 引发 CPU 中断 1：打开，允许 PAP 引发 CPU 中断
17	EXTMSB	rw	EXTMSB:数据扩展模式下，数据扩展顺序设置 0：从 LSB 到 MSB 逐位检查原始数据 1：从 MSB 到 LSB 逐位检查原始数据
16	EXTE	rw	EXTE:数据扩展模式使能 0：普通模式 1：数据扩展模式，此模式只支持数据发送（写），勿在数据接收（读）时设置此位
15-14	TS[1:0]	rw	TS1-0:数据建立时间设置 0x0：数据建立时间为 0 0x1：数据建立时间为 1 个系统时钟的宽度 0x2：数据建立时间为 2 个系统时钟的宽度 0x3：数据建立时间为 3 个系统时钟的宽度
13-12	TH[1:0]	rw	TH1-0:数据保持时间设置 0x0：数据保持时间不小于 0 0x1：数据保持时间不小于 1 个系统时钟的宽度 0x2：数据保持时间不小于 2 个系统时钟的宽度 0x3：数据保持时间不小于 3 个系统时钟的宽度
11-8	TW[3:0]	rw	TW3-0：读/写使能信号宽度设置 0x0：读/写使能信号宽度为 16 个系统时钟的宽度 0x1：读/写使能信号宽度为 1 个系统时钟的宽度

			0x2：读/写使能信号宽度为 2 个系统时钟的宽度 0x3：读/写使能信号宽度为 3 个系统时钟的宽度 0x4：读/写使能信号宽度为 4 个系统时钟的宽度 0x5：读/写使能信号宽度为 5 个系统时钟的宽度 0x6：读/写使能信号宽度为 6 个系统时钟的宽度 0x7：读/写使能信号宽度为 7 个系统时钟的宽度 0x8：读/写使能信号宽度为 8 个系统时钟的宽度 0x9：读/写使能信号宽度为 9 个系统时钟的宽度 0xA：读/写使能信号宽度为 10 个系统时钟的宽度 0xB：读/写使能信号宽度为 11 个系统时钟的宽度 0xC：读/写使能信号宽度为 12 个系统时钟的宽度 0xD：读/写使能信号宽度为 13 个系统时钟的宽度 0xE：读/写使能信号宽度为 14 个系统时钟的宽度 0xF：读/写使能信号宽度为 15 个系统时钟的宽度
7	PND	r	PND:中断请求标志，当一次传输完成或 dma 传输完成时会被硬件置 1。 有 3 种方法清除此标志 1.向 PCLR 写入‘1’ 2.写 PAPBUFL 寄存器来启动下一次传输 3.写 PAPCNT 寄存器来启动下一次 dma
6	PCLR	w	PCLR:软件在此位写入‘1’将清除 PND 中断请求标志，写入‘0’无效
5	DW16ED	rw	DWED：数据大小端选择 0:数据至端口低位 1:数据至端口高位
4	DW16EN	rw	DW16EN：8bit/16bit 模式选择 0：8bit 模式 1：N/A
3	PRE	rw	PRE:读使能信号极性选择 0：读使能信号空闲时为 0 电平，有效时为 1 电平 1：读使能信号空闲时为 1 电平，有效时为 0 电平
2	PWE	rw	PWE:写使能信号极性选择 0：写使能信号空闲时为 0 电平，有效时为 1 电平 1：写使能信号空闲时为 1 电平，有效时为 0 电平
1	DIR	rw	DIR:传输方向设置 0：发送数据 1：接收数据
0	PAPE	rw	PAPE:PAP 接口使能 0：关闭 PAP 接口 1：打开 PAP 接口

2.PAPBUF: PAP buffer register (16bit addressing)

Bit	Name	RW	Description
23-19	PAP_BUF	rw	

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。写此寄存器将启动一次发送或接收操作。

3.PAPDAT0: PAP data register 0 (16bit addressing)

Bit	Name	RW	Description
23-19	PAP_DAT0	rw	

用于数据扩展模式的 DAT0 寄存器，只写，读出为不确定值

4.PAPDAT1: PAP data register 1 (16bit addressing)

Bit	Name	RW	Description
23-19	PAP_DAT1	rw	

用于数据扩展模式的 DAT1 寄存器，只写，读出为不确定值

5.PAPADR: PAP dma start address register (32bit addressing)

Bit	Name	RW	Description
23-19	PAP_ADR	rw	

PAP dma 起始地址寄存器，只写，读出为不确定值

6.PAPCNT: PAP dma counter register (16bit addressing)

Bit	Name	RW	Description
23-19	PAP_CNT	rw	

PAP 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。

第 13 章. LCDC

13. 1. 概述

LCD 模块控制器，主要负责控制推 LCD 屏幕。

最大可以推 6COM & nSEG 的屏幕（SEG 数量参考 IO mapping）。

模块可以使用 RTC 晶振时钟 32KHz 或内部 LRC(约 32KHz)作为模块时钟

13. 2. 寄存器说明

1. LCDC_CON0:LCDC control register 0(32bit addressing).

Bit	Name	RW	Description
31	HD_EN1	rw	high drive EN1，强力强驱电源开关； 0:disable 1:enable
31	HD_EN0	rw	high drive EN0，中等强驱电源开关； 0:disable 1:enable
29-27	reserved	ro	预留
26-17	TEST_FLAG[9:0]	rw	测试模式通道使能，IO 直接输出 LCD 对应通道 TEST_FLAG[0]:SEGIOoutputLCDVDD TEST_FLAG[1]:SEGIOoutputLCDVL2 TEST_FLAG[2]:SEGIOoutputLCDVL1 TEST_FLAG[3]:SEGIOoutputLCDVL0 TEST_FLAG[4]:SEGIOoutputPD TEST_FLAG[5]:COMIOoutputLCDVDD TEST_FLAG[6]:COMIOoutputLCDVL2 TEST_FLAG[7]:COMIOoutputLCDVL1 TEST_FLAG[8]:COMIOoutputLCDVL0 TEST_FLAG[9]:COMIOoutputLCDPD
16	TEST_EN	rw	开启测试模式，IO 直接输出 LCD 静态电平；
15-14	COMCNT	rw	选择 COM 的数目(Duty)； 00:3COM 01:4COM 10:5COM 11:6COM 1/Duty=1/(COMCNT+3)
13-12	CHGMOD	rw	充电模式控制 00:一直用弱充电模式 01:一直用强充电模式

			10:交替充电模式 A 11:交替充电模式 B 交替充电模式 A 在状态切换时开始转强驱动，经过 CHGDUTY+1 个 32KHz 时钟后撤销强驱动 交替充电模式 B 在状态切换前半个周期（32KHz 时钟）开始转强驱动，经过 CHGDUTY+1.5 个 32KHz 时钟后撤销强驱动
11-8	CHGDUTY	rw	交替充电模式下强充电占的 Cycle 数（按 32KHz 时钟） 0000:1 0001:2； 0010:3； 0011:4； …… 1111:16
7	FF	rw	帧频率控制 0:FLCD=32KHz/128 1:FLCD=32KHz/64 FLCD 为模块状态切换时钟频率
6-4	VLCDS	rw	VLCD 控制； 000:2.6V 001:2.7V ……. 110:3.2V 111:3.3V
3-2	BIAS	rw	BIAS 选择 00:模拟模块工作禁止 01:1/2bias 10:1/3bias 11:1/4bias
1	DOTEN	rw	COM0-SEG0 输出一个 1Hz 的信号，可作为“跳秒”脚。
0	LCDCEN	rw	LCD enable

2.SEG_IOEN0:SEG IO enable register1(16bit addressing).

Bit	Name	RW	Description
15-0	SEG_IOEN0	rw	写这个寄存器选择 SEG（PA）的允许位

3.SEG_IOEN1:SEG IO enable register1(16bit addressing).

Bit	Name	RW	Description
15-0	SEG_IOEN1	rw	写这个寄存器选择 SEG（PC）的允许位

4. CLK_CON2:clock system control register1(16bit addressing).

Bit	Name	RW	Description
15-14	LCD_CKSEL	rw	选择 LCD 的时钟 00:WCLK 晶振时钟作为 LCD 模块工作时钟 01:RTOSL 时钟作为 LCD 模块工作时钟 10:LSB 时钟作为 LCD 模块工作时钟 11:LRC 时钟作为 LCD 模块工作时钟
13-0			

5.使用说明

首先选择 LCD 模块工作时钟，如果系统有 32K 晶振，则选 32KHz 晶振时钟，否则选内部 RC 时钟。

其次设定 LCD 电压（VLCDS）、偏置（BIAS）、充电模式（CHGMOD、CHGDUTY）等

再次选择 COM 和 SEG 的位置，例如选择需要 4 个 COM，选择 COM0(PC5), COM1(PC4), COM2(PC3), COM3(PH9), 然后选择 SEG8~SEG15(PA8~PA15)，组成 4COMX8SEG。那么控制器选择，如下：

```
LCDCON0 =   (1<<14);//选择 COM
SEG_IOEN0= 0xff00;//选择 SEG
```

然后打开 LCD 模块

```
LCDCCON0 |= 0x01;        //打开 LCD
```

填写数据；共有 6 个 COM，每个 SEG 需要 21bit，填写数据分别是

```
seg0_dat:   ({portc_die_r[7:6], portd_die_r [5]},{portc_die_r [2:0], porta_die_r [15:0]}),
seg1_dat:   ({portc_hd0_r[7:6], portd_ hd0_r [5]},{portc_ hd0_r [2:0], porta_ hd0_r [15:0]}),
seg2_dat:   ({portc_ pd_r [7:6], portd_ pd_r [5]},{portc_ pd_r [2:0], porta_ pd_r [15:0]}),
seg3_dat:   ({portc_ pu_r [7:6], portd_ pu_r [5]},{portc_ pu_r [2:0], porta_ pu_r [15:0]}),
seg4_dat:   ({portc_ out_r [7:6], portd_ out_r [5]},{portc_ out_r [2:0], porta_ out_r [15:0]}),
seg5_dat:   ({portc_ dir_r [7:6], portd_ dir_r [5]},{portc_ dir_r [2:0], porta_ dir_r [15:0]});
```

第 14 章. 实时时钟（RTC）

14. 1. 概述

1.RTC（Real Time Counter）模块用于进行实时的时间计数，它在结构上独立于系统，可以在系统休眠或者掉电的情况下继续工作，提供连续而准确的计时和闹钟功能。

2.RTC 模块带有一个 32.678KHz 的晶振驱动器（OSCI/PR1-OSCO/PR0），可连接外部晶振作为计时参考时钟。下文中称为 X32K2P。默认打开 X32K2P。

3.RTC 模块可连接芯片内部 32kRC 作为计时参考时钟。下文中称为 RC32K。

4.RTC 模块可连接芯片 BTOSC 作为计时参考时钟，硬件支持 12/24M，其它频率需要软件换算时间。下文中称为 BTOSC。

5.RTC 模块的时间和闹钟分辨率为 1/16S，软件可以随时设置或获取该值。RTC 也可以向系统提供 512Hz、64Hz、2Hz 或 1Hz 的时基中断，用于系统唤醒或者普通定时。

6.与 RTC 相关的 SFR 如下：

JL_P33->RTC_CON　RTC 控制寄存器

JL_P33-> SPI_CON　RTC 寄存器读写控制

JL_P33-> SPI_DAT　RTC 寄存器读写内容

14. 2. 寄存器说明

14.2.1. 控制寄存器(1.2V 主系统，普通 SFR 控制)

1.L_P33-> RTC_CON:RTC control registe

Bit	Name	Default	RW	Description
31-16	-	0	-	预留
15	WKUP_CPND	0	w	WKUP_CPND，写 1 清除 WKUP_PND，写 0 无效
14	X512_CPND	0	w	X512_CPND，写 1 清除 X512_PND，写 0 无效
13	X2_CPND	0	w	X2_CPND，写 1 清除 X2_PND，写 0 无效
12-8	-	0	-	预留
7	WKUP_PND	x	r	WKUP_PND,当闹钟唤醒或 IO 口唤醒发生时,此位被硬件置 1,可向 CPU 请求中断
6	X512_PND	x	r	RC32K X512_PND，当 512Hz 事件发生时，此位被硬件置 1，可向 CPU 请求中断

5	X2_PND	x	r	RC32K X2_PND，当 2Hz 事件发生时，此位被硬件置 1，可向 CPU 请求中断
4	LEVEL_WKUP_EN	0	rw	LEVEL_WKUP_EN，允许 P33 weakup 唤醒发生 --高电平触发（WKUP_LEVEL）
3	EDGE_WKUP_EN	0	rw	WKUP_EN，允许 P33 weakup 唤醒发生—上升沿触发（WKUP_CPND）
2	X512_EN	0	rw	RC32K X512_EN，允许 512Hz 事件发生
1	X2_EN	0	rw	RC32K X2_EN，允许 2Hz 事件发生
0	WKUP_LEVEL	0	rw	WKUP_LEVEL 1:有 WEAK UP 事件正在发生 0:无 WEAK UP 事件发生

注意：wake up 功能参考文档：PMU_wkup&detect

14.2.2. RTC 命令和寄存器（3.3V 电源部分，P33 接口控制)

1.R3_ALM_CON

Bit	Name	Default	RW	Description
7	ALMOUT	0	r	当闹钟标记，当闹钟事件发生时，此位被置 1，可通过下述三种方式将此位清 0 1:ALMEN 设置为 0 2:写时间寄存器 3:写闹钟寄存器
6-1	-	-	-	预留
3-2	CLK_SEL	0	rw	RTC 时钟源选择： 0x: 32KHz 10: 12MHz 11: 24MHz
1	-	-	-	预留
0	ALMEN	0	rw	闹钟使能

2.R3_RTC_CON0

Bit	Name	Default	RW	Description
7-6	-	0	-	预留
5	ALM_RDEN	0	w	闹钟数据读出使能
4	RTC_RDEN	0	w	时钟数据读出使能
3-2	-	0	-	预留

1	ALM_WREN	0	w	闹钟数据写入使能
0	RTC_WREN	0	w	时钟数据写入使能

3.R3_RTC_DAT0:
4.R3_RTC_DAT1:
5.R3_RTC_DAT2:
6.R3_RTC_DAT3:
7.R3_RTC_DAT4:RTC 数据交互 SFR，有 3 种用法

Bit	Name	Default	RW	Description
7-0	R3_RTC_DATx	0	-	用法 1:软件寄存器，可读写。 用法 2:OSC 32k RTC 时钟及闹钟信息读写(附录 1) 用法 3:RC 32k RTC 时钟及闹钟信息读写(附录 2)

8.R3_OSL_CON

Bit	Name	Default	RW	Description
7-6	-	0	-	预留
5-4	X32XS	01	rw	X32K2P 振荡器电流选项 00:小电流 10:… 01:… 11:大电流
3	-	0	-	预留
2	X32TS	0	rw	X32K2P 振荡器测试使能
1	X32OE	1	rw	X32K2P 振荡器输出使能
0	X32EN	1	rw	X32K2P 振荡器使能

OSC RTC 走时及闹钟格式

RTC/ALM:

R3_RTC_CON0	0	day[15:8]	day[7:0]	hour[4:0]	minute[5:0]	second[5:0]

RC RTC 走时及闹钟格式

RTC/ALM:

R3_RTC_CON0	0	cnt[39:32]	cnt[31:24]	cnt[23:16]	cnt[15:8]	cnt[7:0]

14. 3. 例程

OSC 32k 走时数据读写

```
void osc_alm_init (u16 day, u8 hour, u8 min, u8 sec);
void osc_rtc_init (u16 day, u8 hour, u8 min, u8 sec);
void osc_alm_read (void);
void osc_rtc_read (void);

void osc_alm_init (u16 day, u8 hour, u8 min, u8 sec){
    u16 addr;
    addr = R3_RTC_CON0;
    p33_cs_h(addr);
    p33_buf((u8)((addr & 0x300) >> 8));
    p33_buf((u8)(addr & 0xff));
    p33_buf(BIT(1));
    p33_buf(0);
    p33_buf(day >> 8);
    p33_buf(day & 0xff);
    p33_buf(hour);
    p33_buf(min);
    p33_buf(sec);
    p33_cs_l;
}

void osc_rtc_init (u16 day, u8 hour, u8 min, u8 sec){
    u16 addr;
    addr = R3_RTC_CON0;
    p33_cs_h(addr);
    p33_buf((u8)((addr & 0x300) >> 8));
    p33_buf((u8)(addr & 0xff));
    p33_buf(BIT(0));
    p33_buf(0);
    p33_buf(day >> 8);
    p33_buf(day & 0xff);
    p33_buf(hour);
    p33_buf(min);
    p33_buf(sec);
    p33_cs_l;
}

u16 alm_day_rd, u8 alm_hour_rd, u8 alm_min_rd, u8 alm_sec_rd;
void osc_alm_read (void){
    u16 addr;
```

```
    addr = R3_RTC_CON0;
    p33_cs_h(addr);
    p33_buf((P33_OR << 5) | (u8)((addr & 0x300) >> 8));
    p33_buf((u8)(addr & 0xff));
    p33_buf(BIT(5));        //rd alm
    p33_buf(0);
     alm_day_rd = p33_buf(0);
     alm_day_rd <<= 8;
    alm_day_rd = p33_buf(0) | alm_day_rd;
    alm_hour_rd = p33_buf(0);
    alm_min_rd = p33_buf(0);
    alm_sec_rd = p33_buf(0);
     p33_cs_l;
}

u16 rtc_day_rd, u8 rtc_hour_rd, u8 rtc_min_rd, u8 rtc_sec_rd;
void osc_rtc_read (void){
     u16 addr;
    addr = R3_RTC_CON0;
    p33_cs_h(addr);
    p33_buf((P33_OR << 5) | (u8)((addr & 0x300) >> 8));
    p33_buf((u8)(addr & 0xff));
    p33_buf(BIT(4));        //rd rtc
    p33_buf(0);
     rtc_day_rd = p33_buf(0);
     rtc_day_rd <<= 8;
    rtc_day_rd = p33_buf(0) | rtc_day_rd;
    rtc_hour_rd = p33_buf(0);
    rtc_min_rd = p33_buf(0);
    rtc_sec_rd = p33_buf(0);
     p33_cs_l;
}
```

第 15 章. ADC

15.1. 概述

10Bit ADC(A/D 转换器)，其时钟最大不可超过 1MHz。

15.2. 寄存器说明

1.ADC_CON: ADC control register

Bit	Name	RW	Description
31-16	-	-	预留
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 PA1 0001:选择 PA5 0010:选择 PA6 0011:选择 PA10 0100:选择 PA12 0101:选择 PB1 0110:选择 PB3 0111:选择 PB4 1000:选择 PB8 1001:选择 PB10 1010:选择 PC4 1011:选择 PC6 1100:选择 USBDP 1101:选择 PC5 1110:选择 RTC to ADC 1111:选择 VDDIO to ADC(PLL,PMU,AUDIO,BT,FM 注意同一时间只能开一个通道，AUDIO 部分 en 是写 1 才是关闭)
7	PND	r	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	CPND:只写，写‘1’清除中断请求位，写‘0’无效
5	ADC_IE	rw	ADC_IE:ADC 中断允许
4	ADC_EN	rw	ADC_EN:ADC 控制器使能
3	ADC_AE	rw	ADC_AE:ADC 模拟模块使能
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.P3_ANA_CON4 ANA control SFR（PMU to ADC）

Bit	Name	RW	Description
7-5	reserved	rw	预留
4	VGB_SEL	rw	0:main VBG 1: weak VBG
3-1	CHANNEL_ADC_S[2:0]	rw	CHANNEL_ADC_S2-0:select channel to ADC 000(default): VBG 001: VDC13 010: SYSVDD 011: VTEMP 100: PROGF 101: 1/4 VBAT 110: 1/4 LDO5v 111: WVDD
0	PMU_DET_EN	r	PMU_DET_EN:PMU voltage detect to ADC output enable

3.ADA_CON3: ADA control register（AUDIO to ADC）

Bit	Name	RW	Description
30	R_VOUTR_TEST_EN_11v	rw	0: R_VOUTR to SARADC（0 是开启，1 是关闭）
29	R_VOUTL_TEST_EN_11v	rw	0: R_VOUTL to SARADC（0 是开启，1 是关闭）
28	DACVDD_TEST_EN_11v	rw	0: DACVDD to SARADC（0 是开启，1 是关闭）
27	CTADCREF_TEST_11v	rw	1: CTADCREF to SARADC
26	MICLDO_TEST_11v	rw	1: MICLDO to SARADC
25	F_VOUTR_TEST_EN_11v	rw	0: F_VOUTR to SARADC（0 是开启，1 是关闭）
24	F_VOUTL_TEST_EN_11v	rw	0: F_VOUTL to SARADC（0 是开启，1 是关闭）

Note:only one TEST channel can set to 1 while others must set to 0.

4.PLL_CON1: pll to adc

Bit	Name	RW	Description
18	PLL_TEST_EN	rw	pll to adc enable
17-16	PLL_TEST_S<1:0>	rw	00bias current 10u（CP 不测）-- 01PLL analog block voltage1.0v~1.2v 10PLL digital block voltage1.0v~1.2v 11Pll output 480meg clock（环路测试实现）

5.WLA_CON25: FM to ADC

BIT	NAME	RW	Description
19	FMTB_ADC_TEST_EN	rw	FM to ADC en
23-21	FMTX_SEL	rw	

6.BT to adc

en: WLA_CON4[6]

第 16 章. USB_bridge

16.1. 概述

负责控制系统与 USB 模块之间的通讯，包括:

1. 接收 CPU 的命令，控制 SIE;
2. 管理 endpoint mapping;
3. 负责 SIE 和 memory 的通讯;
4. 控制 USB PHY。

16.2. 寄存器说明

1.USB_CON0

Bit	Name	RW	Description
31-24	-	-	预留
23	EP4_DISABLE	rw	关闭端点 4，不会返回 ack，nak，stall
22	EP3_DISABLE	rw	关闭端点 3，不会返回 ack，nak，stall
21	EP2_DISABLE	rw	关闭端点 2，不会返回 ack，nak，stall
20	EP1_DISABLE	rw	关闭端点 1，不会返回 ack，nak，stall
19	-	-	预留
18	LOWP_MD_	rw	低有效，默认为使用低功耗模式 0:使用低功耗模式，即系统时钟可跑低于 48m; 1:不使用低功耗模式，即要用 USB 模块时系统得跑 48m 或 48m 以上;
17	SE_DP	r	DP 输入的电平
16	SE_DM	r	DM 输入的电平
15	CHKDPO	r	DP 外接下拉检查结果. 当 PDCHKDP=0, CHKDPO=1
14	SIE_PND	r	SIE 中断请求标志，通过访问 USB 模块清除标志
13	SOF_PND	r	SOF 中断请求标志
12	CLR_SOFP	w	写 1 清除 SOF 中断请求标志，写 0 无效
11	SIEIE	rw	SIE 中断使能
10	SOFIE	rw	SOF 中断使能
9	PDCHKDP	rw	DP 外接下拉检查使能 0:disable 1:enable
8-7	-	-	预留
6	USB_TEST	rw	USB 测试模式
5	VBUS	rw	USB 电源
4	CID	rw	USB 工作模式:

			0:host 1:device
3	TM1	rw	用于缩短检测连接时间(short connect timeout)： 0:disable 1:enable
2	USB_NRST	rw	USB 模块复位： 0:reset 1:release reset
1	LOW_SPEED	rw	低速 USB_DMA 使能： 0:disable 1:enable
0	PHY_ON	rw	USB_PHY 使能: 0:disable 1:enable

2.USB_CON1

Bit	Name	RW	Description
31-16	-	-	预留
15	MC_ACK	r	USB 模块 ACK 信号： 0:busy 1:ready
14	MC_RNW	w	USB 寄存器读写控制： 0:write 1:read
13-8	MC_ADR	w	访问 USB 寄存器地址
7-0	MC_DAT	rw	访问 USB 寄存器的数据

3.EP0_CNT，EP1_CNT，EP2_CNT，EP3_CNT，EP4_CNT

Bit	Name	RW	Description
31-0	EP(n)_CNT	w	usb endpoint 0-4 发送数据个数，单位为 byte

4.EP0_ADR

Bit	Name	RW	Description
31-0	EP0_ADR	w	usb endpoint 0 发送/接收数据的起始地址

5.EP1_TADR，EP2_TADR，EP3_TADR，EP4_TADR

Bit	Name	RW	Description
31-0	EP(n)_TADR	w	usb endpoint 1-4 发送数据的起始地址

6.EP1_RADR，EP2_RADR，EP3_RADR，EP4_RADR

Bit	Name	RW	Description
31-0	EP(n)_RADR	w	usb endpoint 1-4 接收数据的起始地址

7.USB_IO_CON0

Bit	Name	RW	Description
31-15	-	-	预留
14	DMDIEH	rw	3.3V DM 数字输入使能
13	DPDIEH	rw	3.3V DP 数字输入使能
12	SR	rw	输出驱动能力选择 0:slow 1:fast
11	IO_MODE	rw	IO 模式使能 0:USB 模式 1:普通 IO 模式
10	DMDIE	rw	1.2V DM 数字输入使能
9	DPDIE	rw	1.2V DP 数字输入使能
8	IO_PU_MODE	rw	IO 上下拉分两种模式 0:USB 模式 1:普通 IO 模式
7	DMPU	rw	DM 上拉
6	DPPU	rw	DP 上拉
5	DMPD	rw	DM 下拉
4	DPPD	rw	DP 下拉
3	DMIE	rw	DM 方向 0:输出 1:输入
2	DPIE	rw	DP 方向 0:输出 1:输入
1	DMOUT	rw	DM 输出电平
0	DPOUT	rw	DP 输出电平

8.USB_IO_CON1

Bit	Name	RW	Description
31-2	-	-	预留
1	DMIN	r	DM 输入电平
0	DPIN	r	DP 输入电平

第 17 章. AUDIO IIS

17. 1. 概述

Audio IIS 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。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。

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。

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。
3) 从 PA4/PC0/PG8 端口输入相应频率的 MCLK

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。

ALNK 具备 4 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。有：

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）。

下表为 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
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			

下表为 ALNK0 和 ALNK1 使用的 IO 端口列表

	信号名称	占用的 IO 端口	
		ALNK0_IOS=0	ALNK0_IOS=1
ALNK0	MCLK	PA8	PA15
	SCLK	PA2	PA9
	LRCK	PA3	PA10

	CH0DAT	PA4	PA11
	CH1DAT	PA5	PA12
	CH2DAT	PA6	PA13
	CH3DAT	PA7	PA14
ALNK1	信号名称	占用的 IO 端口	
	MCLK	PB0	
	SCLK	PC0	
	LRCK	PC1	
	CH0DAT	PC2	
	CH1DAT	PC3	
	CH2DAT	PC4	
	CH3DAT	PC5	

下表为每路 ALNK 使用的 SFR 列表

SFR 名称	描述
ALNK_CON0	ALNK 控制寄存器 0
ALNK_CON1	ALNK 控制寄存器 1
ALNK_CON2	ALNK 控制寄存器 2
ALNK_CON3	ALNK 控制寄存器 3
ALNK_ADR0	通道 0 DMA 起始地址寄存器
ALNK_ADR1	通道 1 DMA 起始地址寄存器
ALNK_ADR2	通道 2 DMA 起始地址寄存器
ALNK_ADR3	通道 3 DMA 起始地址寄存器
ALNK_LEN	通道 0-3 DMA 数据长度寄存器

AC695N 中包含两个完全相同的 audio link 模块(ALNK0 和 ALNK1)，ALNK0 和 ALNK1 相互独立，因此，AC695N 支持同时 4 声道输入和 4 声道输出。

AC695N 中，ALNK0 的中断向量号为 11，ALNK1 的中断向量号为 36。

由于 ALNK0 和 ALNK1 的内部结构一模一样，这里只对 ALNK0 的相关寄存器配置和数据组织结构进行说明。

17.2. 控制寄存器

1. ALNK_CON0: control register 0 (16bit addressing)

Bit	Name	RW	Description
15	FLAG3	rw	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	rw	
13	FLAG1	rw	
12	FLAG0	rw	
11	ALNKE	rw	ALNK 模块使能 0:模块关闭 1:模块打开
10	SCKINV	rw	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	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	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	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	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	-	-	预留

2. ALNK_CON1: control register 1 (16bit addressing)

Bit	Name	RW	Description
15	T3DIR	rw	通道 x 方向设置 0:发送 1:接收 通道 x 数据位宽设置 0:16bit 1:24bit 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
14	T3LEN	rw	
13~12	T3MOD	rw	
11	T2DIR	rw	
10	T2LEN	rw	
9~8	T2MOD	rw	
7	T1DIR	rw	

6	T1LEN	rw	0	1	IIS（数据延后 1bit）
5~4	T1MOD	rw	0	2	左对齐
3	T0DIR	rw	0	3	右对齐
2	T0LEN	rw	1	1	DSP0（数据延后 1bit）
1~0	T0MOD	rw	1	2	DSP1
			others		预留，不可设置

3. ALNK_CON2: control register 2 (8bit addressing)

Bit	Name	RW	Description
7	PND3	r	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	
5	PND1	r	
4	PND0	r	
3	CPND3	w	写 1 清除 Pending，写 0 无效
2	CPND2	w	
1	CPND1	w	
0	CPND0	w	

4. ALNK_CON3: control register 3 (8bit addressing)

Bit	Name	RW	Description
7~5	LRDIV[2:0]	rw	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[2:0]	rw	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	MDIV[1:0]	rw	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 注意：OSC CLK 和 PLL ALNK CLK 参考 Clock System Spec 的说明。

5. ALNK_ADR: alnk dma start address register (26bit addressing)

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。

6. ALNK_LEN: alnk dma sample length register (16bit addressing)

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】：2(CH)代表左、右声道。

17. 3. 数据组织结构

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：

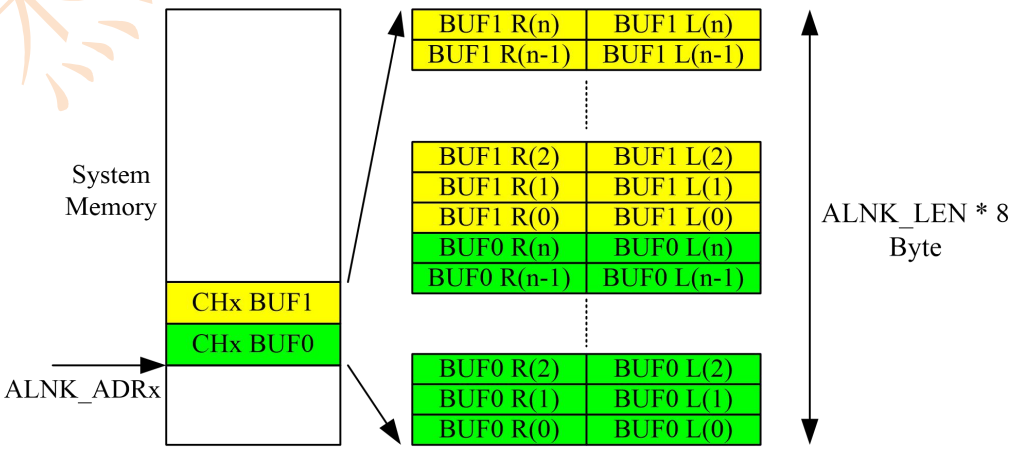
16bit data: ALNK_LEN * 2(CH) * 2(Byte) * 2(dual buffer) = ALNK_LEN * 8 Byte

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

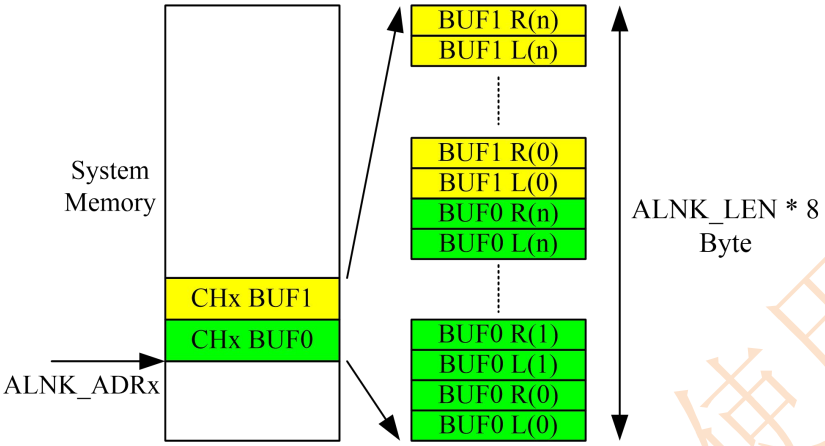
当四条通道一起使用时，共需要 4 倍大小的 buffer。

特别的，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，数据组织如下图。



24bit 数据宽度时，数据组织如下图。



第 18 章. 触摸按键控制器 CTM

18. 1. 模块说明

触摸按键控制器(capacitive-touch module，简称 CTM)通过一个对人体分布电容对触摸按键电容的影响来进行按键检测。该模块通过一个 16 选 1 的模拟开关连接至 16 个 IO 口，当人体触摸外部电容按键时，IO 口外部整体电容增加，因此对电容的充电时间增加，软件通过检测该充电时间差值以进行按键判定。

对于选定的通道，在没有触碰按键的时候，PCB 板的走线和其他寄生的电容组成了触摸按键的固有电容 Cp，此时对应的充电时间是 Tp。当触碰按键时，按键电容增加了 Cs，对应的充电时间增加至 Ts。Ts 和 Tp 的关系，对应于 Cs 和 Cp 的关系：Tp/Ts = Cp/(Cp+Cs)

PCB layout 需要尽可能减少 Cp，增加 Cs，以增加按键灵敏度。

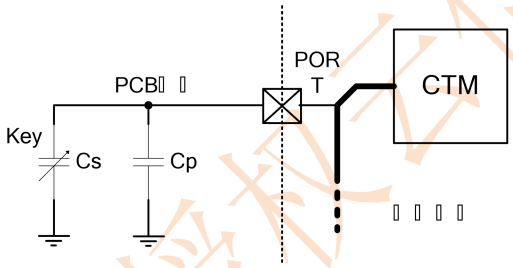


图 18-1 CTM 模块

CTM 模块通过控制电容的充放电，在电容充电过程计数，并将该计数通过 DMA 保存到 SRAM 中。通过该计数和选择的计数时钟，可算出充电时间，进而算出电容量。

触摸按键控制器具有一个可配置的闸门时间发生器，用以产生一个 mS 级的闸门时间，该闸门时间控制每一路通道轮询时间。另有一个计数器对充电时间进行计数。

触摸按键控制器支持 1-16 路按键检测，可通过软件进行配置。所有通道会被自动轮询检测，未使能的通道也会占用一份闸门时间，16 路轮询结束后会产生中断请求。若使能某通道，则需将该通道对应的 IO 口设置为输入，关闭上拉，关闭下拉，关闭数字功能，否则会导致异常。

不论使能与否，所有通道在各自的闸门时间结束后都会产生 DMA 请求，将充电时间计数保存到 SRAM 中。由于不论使能与否，所有通道都占用一份闸门时间，所以每通道的采样间隔时间和闸门时间是相关的，例如闸门时间设置为 2.5mS，则每通道的采样间隔时间为 2.5mS * 16CH = 40mS，即采样率为 25 次/秒。

具体闸门时间的选择需根据实际应用情况而定，若需响应快速点触按键，推荐闸门时间为 1.25mS，即每 20mS 更新一个通道的计数值。若不需响应快速点触按键，推荐闸门时间为 2.5mS，即每 40mS 更新一个通道的计数值。

18.2. 寄存器说明

1. JL_CTM->CON0: CTM configuration register0

Bit	Name	RW	default	Description
31:24	-	r	0	reserved
23	DISC_MD	rw	0	The discharge mode: 0:Don't discharge, after charge full 1:Discharge, after charge full
22:21	reserved	r	0	
20:18	IS	rw	0	The charge current selection: 0: 0.25uA 1: 0.5uA 2: 1 uA 3: 2uA 4: 4uA 5: 8uA 6: 16uA 7: 32uA
17:16	US	rw	0	The comparator reference voltage selection: 0: 0.48V 1: 0.64V 2: 0.8V 3: 0.96V
15:12	DTS	rw	0	The discharge time selection, or the pre_scaler setting: 0: div1 1: div2 2: div4 …… 15: div32768
11:10	CKS	rw	0	The clock source of charge counter : 0: osc_clk 1: lrc_clk 2: pll_240m 3: pll_480m
9	BFLAG	r	0	The buffer flag: 0: using buffer0 1: using buffer1
8	IE	rw	0	The CTM interrupt enable: 0: disable

				1: enable
7	PND	r	0	The CTM pending: 0: without pending 1: with pending
6	CPND	w	0	Clear pending: 0: invalid 1: clear pending
5:4	GTS	rw	0	The clock source of gating : 0: osc_clk 1: lrc_clk 2: pll_240m 3: pll_480m
3:1	PSET	rw	0	The pre_scaler setting: 0: div8 1: div16 2: div32 3: div64 4: div8192 5: div16384 6: div32768 7: div65536
0	EN	rw	0	The CTM enable: 0: disable 1: enable

a)放电时间计算：

放电时间=1/（lsb_clk 频率/预分频）

其中，预分频见 DTS。

（注：放电时间一般需要大于 2us。）

b)闸门时间计算：

闸门时间=1/（闸门时钟源频率/预分频）

其中，预分频见 PSET。

（注：为了获得约 2ms 的闸门时间，一般 PSET 中 0~3 用于闸门时钟源为 LRC_CLK，PSET 中 4~7 用于闸门时钟源为 OSC_CLK。）

2. JL_CTM->CON1: CTM configuration register1

Bit	Name	RW	default	Description
31:16	-	r	0	reserved

15	CH15_EN	rw	0	The channel15 enable.
14	CH14_EN	rw	0	The channel14 enable.
13	CH13_EN	rw	0	The channel13 enable.
12	CH12_EN	rw	0	The channel12 enable.
11	CH11_EN	rw	0	The channel11 enable.
10	CH10_EN	rw	0	The channel10 enable.
9	CH9_EN	rw	0	The channel9 enable.
8	CH8_EN	rw	0	The channel8 enable.
7	CH7_EN	rw	0	The channel7 enable.
6	CH6_EN	rw	0	The channel6 enable.
5	CH5_EN	rw	0	The channel5 enable.
4	CH4_EN	rw	0	The channel4 enable.
3	CH3_EN	rw	0	The channel3 enable.
2	CH2_EN	rw	0	The channel2 enable.
1	CH1_EN	rw	0	The channel1 enable.
0	CH0_EN	rw	0	The channel0 enable.

3. JL_CTM->ADR: CTM address register

Bit	Name	RW	default	Description
31:0	CTM_ADR	rw	0	The CTM address register.

乒乓 BUFFER 缓存机制：

DMA 保存数据到 SRAM，采用了乒乓 BUFFER 的缓存机制，即双缓存机制。每组 BUFFER 大小为 64byte（=16channel*4byte/channel）。BUFFER 的起始地址由 **JL_CTM->ADR** 确定。当软件读到 BFLAG=0 时，说明 DMA 正在使用第 0 组 BUFFER（简称 BUFFER0），当软件读到 BFLAG=1 时，说明 DMA 正在使用第 1 组 BUFFER（简称 BUFFER1）。因此，当软件读到 BFLAG=0 时，应该去读 BUFFER1 中的数据，当软件读到 BFLAG=1 时，应该去读 BUFFER0 中的数据。

一般至少舍弃前 2 次的数据结果。

第 19 章. PDM LINK

19. 1. 概述

Pdm Link(简称 PLNK)是一种数字麦克风(digital mic, DMIC)接口，存在主机和从机两种形式。主机形式的 PLNK 接口的同步时钟(SCLK)由 PLNK 本身产生；从机形式的 PLNK 接口的同步时钟(SCLK)由外部提供。

AC695N 中的 PLNK 为主机形式，该接口用于将外部输入进来的 DSM 形式的信号进行“解调”，恢复成数字码，留作后续处理。

下表为 PLNK 使用的 IO 端口列表

信号名称	占用 IO 端口	
PLNK_SCLK	IOMAP_CON1[20]=1	PA2
	IOMAP_CON1[20]=0	output channel2[4]
PLNK_DAT0	input channel8	
PLNK_DAT1	IOMAP_CON3[15]=0	PA3
	IOMAP_CON3[15]=1	input channel9

19. 2. 寄存器说明

1.PLNK_CON: ADC control register (16bit addressing)

Bit	Name	RW	Description
31-16	-	-	预留
15	PND	r	当 dma buf0 或 buf1 被使用完毕后，此位被硬件置 1
14	CPND	w	写 1 清除 PND
13-10	-	-	预留
9	FLAG	r	通道 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
8	IE	rw	PLNK 中断使能
7-6	CH1MD	rw	通道 1 输入模式选择 0: 输入口 PLNK_DAT1 上升沿采样 1: 输入口 PLNK_DAT1 下降沿采样 2: 输入口 PLNK_DAT0 上升沿采样 3: 输入口 PLNK_DAT0 下降沿采样
5	CH1SC	rw	通道 1 增益控制 0:0dB

			1:-6dB
4	CH1EN	rw	通道 1 使能
3-2	CH0MD	rw	通道 0 输入模式选择 0: 输入口 PLNK_DAT0 上升沿采样 1: 输入口 PLNK_DAT0 下降沿采样 2: 输入口 PLNK_DAT1 上升沿采样 3: 输入口 PLNK_DAT1 下降沿采样
1	CH0SC	rw	通道 0 增益控制 0:0dB 1:-6dB
0	CH0EN	rw	通道 0 使能

2.PLNK_SMR:plnk clock divider register(8bit)

SCLK 分频设置，分频数为 lsb_clk/(PLNK_SMR+1)

3.PLNK_ADR: plnk dma start address register (26bit addressing)

PLNK DMA 操作起始地址寄存器，在使用 PLNK 之前，必须由软件初始化对齐至 4Byte。

4.PLNK_LEN: plnk dma sample length register (16bit addressing)

PLNK DMA 样点长度寄存器，在使用 PLNK 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 PLNK_LEN 设置为 100 时，则 PLNK 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：

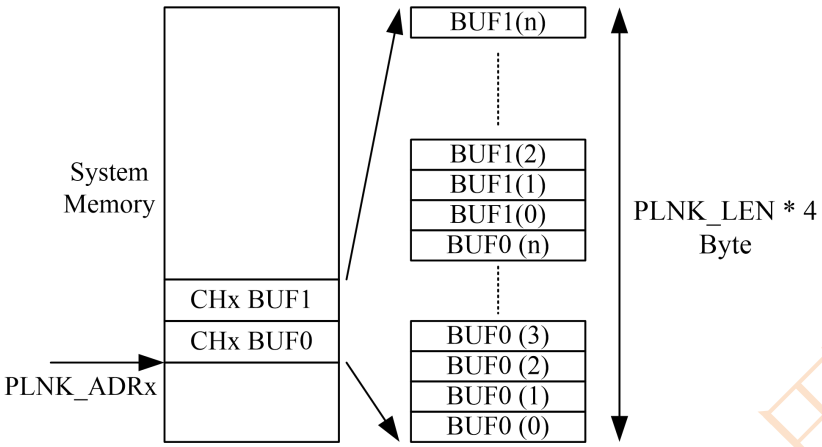
100 * 2(Byte) * 2(dual buffer) = 400 Byte

19.3. 数据组织架构

PLNK 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 PLNK_LEN 寄存器指定，总 buffer 需求为：

PLNK_LEN * 2(Byte) * 2(dual buffer) = PLNK_LEN * 4 Byte

数据组织如下图


System
Memory
CHx BUF1
CHx BUF0
PLNK_ADRx
BUF1(n)
BUF1(2)
BUF1(1)
BUF1(0)
BUF0 (n)
BUF0 (3)
BUF0 (2)
BUF0 (1)
BUF0 (0)
PLNK_LEN * 4
Byte

第 20 章. PULSE COUNTER

20. 1. 概述

以下为 pulse_counter/触摸键的寄存器说明及使用方法。

20. 2. 寄存器说明

1.PL_CNT_CON

Bit	Name	RW	Description
7-4	-	-	预留
3-2	clk_sel	rw	pulse counter 时钟选择 00: 选择 osc_clk 作为计数时钟; 01: 选择 clk_mux_in 作为计数时钟（iomc1[15:12] intput channel 选择）; 10: 选择 pll_192m 作为计数时钟; 11: 选择 pll_240m 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	触摸键使能，当 cap_mux_in（input channel 2）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	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;

NOTE：当 cap_mux_in（input channel 2）选择 TMRx_PWMOUT, pulse counter 时钟选择 clk_mux_in（input channel 4）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。

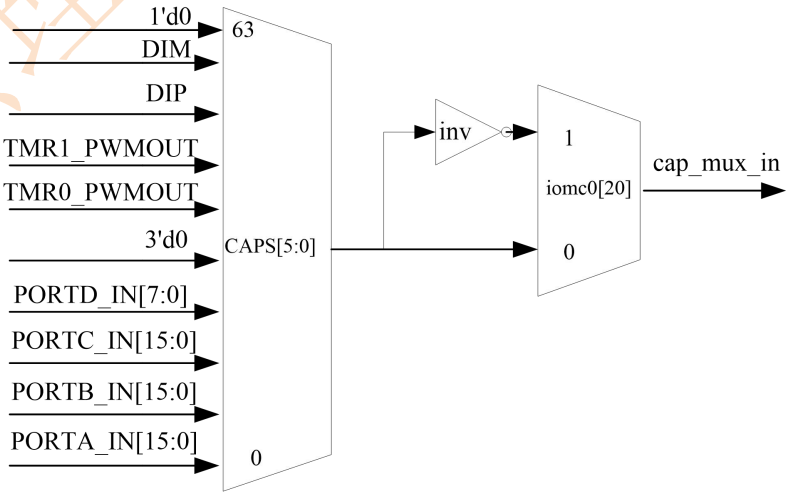


图 20-1 cap_mux_in(input channel 2)电路控制图

20. 3. 例程

以下以 PA1 为例:

1.变量定义：

```
int Touchkey_value_old，Touchkey_value_new，Touchkey_value_delta；      //32bit 有符号整数
```

2.选择检测管脚

```
IOMC2 &= ~(0xff00)；//
IOMC2 |= 0x01 << 16；          //选择 PA1

PORTA_OUT |= BIT(1)；
PORTA_DIR &= ~BIT(1)；         //PA1 输出为 1
PORTA_PD |= BIT(1)；//PA1 下拉打开
```

3.初始化计数器配置

```
PLCNTCON &= ~(0xC)；//
PLCNTCON |= BIT(3)；//选择 pll_192m 作为触摸键时钟
PLCNTCON |= BIT(1)；//使能计数器
```

4.检测电容

```
Touchkey_value_old = PLCNTVL；//读取旧值
PORTA_DIR |= BIT(1)；//对应管脚输入使能
While(PORTA_IN & BIT(1))；
Touchkey_value_new = PLCNTVL；
If(Touchkey_value_old  >  Touchkey_value_new)then
Touchkey_value_new += 0x10000；
Touchkey_value_delta = Touchkey_value_new - Touchkey_value_old；
```

第 21 章. RDEC

21.1. 概述

RDEC（rotate decoder）是一个用于旋转编码器检测的模块，它支持两线输入的旋转编码器，可以检测旋转方向和旋转步数。

AC695N 中，一共支持 3 路 RDEC，每路 RDEC 均可独立工作且互不影响。其中，RDEC0 的中断号为 25，RDEC1 的中断号为 49，RDEC2 的中断号为 50。

下表为各 RDEC 使用的 IO 端口列表

模块	信号名称	IO 选择	
RDEC0	RDEC0_SIN[0]	IOMAP_CON1[12]=0	IOMAP_CON1[12]=1
		PA2	Input channel 6
	RDEC0_SIN[1]	IOMAP_CON1[13]=0	IOMAP_CON1[13]=1
		PA3	Input channel 7
RDEC1	RDEC1_SIN[0]	IOMAP_CON2[6]=0	IOMAP_CON2[6]=1
		PB2	Input channel 6
	RDEC1_SIN[1]	IOMAP_CON2[7]=0	IOMAP_CON2[7]=1
		PB3	Input channel 7
RDEC2	RDEC2_SIN[0]	IOMAP_CON2[14]=0	IOMAP_CON2[14]=1
		PB4	Input channel 6
	RDEC2_SIN[1]	IOMAP_CON2[15]=0	IOMAP_CON2[15]=1
		PB6	Input channel 7

下表为每路 RDEC 使用的 SFR 列表

SFR 名称	描述
RDEC_CON	RDEC 控制寄存器
RDEC_DAT	RDEC 结果读出寄存器

21.2. 控制寄存器

1. RDEC_CON8bit 宽度

Bit	Name	RW	Description
7	PND	r	PND，只读。写入无效。
6	CPND	w	CPND，只写。写入 1 清除 PND，读出永远为 0。
5~2	RDEC_SPD	rw	RDEC_SPD，采样速率设置 Tsr = (2^ RDEC_SPD) / Flsb Flsb 为低速外设总线的频率，软件应当设置 RDEC_SPD 使 Tsr 介于 0.5~2mS 之间
1	RDEC_POL	rw	RDEC_POL 0:输入引脚无信号时处于 1 状态（外部引脚上拉） 1:输入引脚无信号时处于 0 状态（外部引脚下拉）
0	EN	rw	RDEC_EN 0: 模块关闭 1: 模块打开

2. RDEC_DAT8bit 宽度，只读

Bit	Name	RW	Description
7~0	RDEC_DAT	r	此寄存器为 8 位有符号数，表示旋转编码器正反向旋转的步数。 当检测到旋转编码器动作时，PND 会设置为 1，软件可通过中断或查询方式来读取此寄存器。也可以完全不理会 PND，软件隔一段时间来访问此寄存器，但需注意检测时间不能过长，如果此寄存器的数值超过-128~127 的区间，将会产生溢出，无法表示出旋转编码器的正确动作。 【注意】：清 PND 的动作会将此寄存器一同清 0。

第 22 章. SPDif

22.1. 概述

spdif（Sony/Philips Digital Interface）音频接口包含一个 mater 和 slave；master 和 slave 各有一个 IO 和外部进行通讯。

22.2. slave 使用介绍

打开模块使能 ss_en，触发模块开始 ss_str，硬件会在接收的 channel status bit 的数目（以 byte 为单位）等于所设置的数目（csbr_cnt，以 byte 为单位）时，产生中断，通过读取 ss_csb5~0（采集的方式是从高位移位进寄存器）的信息知道接收的音频数据格式等相关信息；

再设置数据位数模式 dat_dma_md（16/24bit），并打开 dat_dma_en(inf_dma_en 根据需要打开)；dma（dat/inf）的模式采用乒乓 buffer 的方式，中断产生后可以通过读取相应的 buffer（dat/inf）使用情况（dat_pha/inf_pha）来判断哪块 buffer 可以用。

22.3. 寄存器说明

1.ss_con ：spdif slave configuration (32bits)

Bit	Name	RW	Description
31:16	rle_cnt	w/r	接收电平标志计数值
15:14	ss_cks	w/r	模块采样时钟选择 0:pll_192m（默认） 1:pll_96m 2:pll_48m 3:1’b1
13	inf_pha	r	0:当前使用为 information buffer0 1:当前使用为 information buffer1
12	dat_pha	r	0:当前使用为 data buffer0 1:当前使用为 data buffer1
11	rl_err	r	电平长度接收错误标志。 当模块在正常收数的过程中，突然接收到的高/低电平（热拔或者是光纤的自激行为）的计数值（以时钟周期为单位） 等于 rle_cnt 计数值时产生的标志； 建议接收到该标志时，将模块复位重启。
10	d_err	r	数据错误：接收到的这包数据中偶校验出错或者有数据的 valid_bit 为 1

9	b_err	r	块错误：块的序文（Z）接收不匹配
8	f_err	r	帧错误：子帧的序文（X/Y）接收不匹配
7	err_clr	w	错误清除，可以同时清除 f_err，b_err，d_err，rl_err
6	i_pnd	r	information 接收数目达到 inf_len 后中断标志
5	i_pnd_clr	w	information pnd clear
4	d_pnd	r	data 接收数目达到 dat_len 后中断标志
3	d_pnd_clr	w	data pnd clear
2	reserve		
1	ss_str	w	start 脉冲，启动工作
0	ss_en	w/r	模块使能

2.SS_IO_CON: spdif slave io configuration (32bits)

Bit	Name	RW	Description
31:28	reserve		
27	Io_doe	w/r	spdif 输出 io 使能控制
26:24	io_selo	w/r	3’b100:io 引脚 a 信号输出 3’b 101:io 引脚 b 信号输出 3’b 110:io 引脚 c 信号输出 3’b 111:io 引脚 d 信号输出 3’b 0XX:无输出
23:20	et_den	w/r	4’b0001: io 引脚 a 拔出检测使能 4’b0010: io 引脚 b 拔出检测使能 4’b0100: io 引脚 c 拔出检测使能 4’b1000: io 引脚 d 拔出检测使能
19	et_pnd	r	设备拔出（extract）pnd
18	et_pnd_clr	w	设备拔出（extract）pnd 清除
17	is_pnd	r	设备插入（inset）pnd
16	is_pnd_clr	w	设备插入（inset）pnd 清除
15:12	online	r	4’b0001: io 引脚 a 有设备在线 4’b0010: io 引脚 b 有设备在线 4’b0100: io 引脚 c 有设备在线 4’b1000: io 引脚 d 有设备在线
11	reserve		

10:8	is_det	r	1:io 引脚 a 有设备插入 2:io 引脚 b 有设备插入 3:io 引脚 c 有设备插入 4:io 引脚 d 有设备插入 others:无
7:4	is_den	w/r	4’b0001: io 引脚 a 插入检测使能 4’b0010: io 引脚 b 插入检测使能 4’b0100: io 引脚 c 插入检测使能 4’b1000: io 引脚 d 插入检测使能
3	reserve		
2:0	io_seli	w/r	3’b100:io 引脚 a 输入 3’b 101:io 引脚 b 输入 3’b 110:io 引脚 c 输入 3’b 111:io 引脚 d 输入 3’b 0XX:无输入

3.SS_DMA_CON: dma configuration (32bits)

Bit	Name	RW	Description
31:22	reserve		
21	b_err_ie	w/r	b_err 中断使能
20	f_err_ie	w/r	f_err 中断使能
19	et_pnd_ie	w/r	et_pnd 中断使能
18	is_pnd_ie	w/r	is_pnd 中断使能
17	i_pnd_ie	w/r	i_pnd 中断使能
16	d_pnd_ie	w/r	d_pnd 中断使能
15	c_pnd_ie	w/r	c_pnd 中断使能
14	csbr_pnd	r	channel status bit 接收到达所要数目标志
13	csbr_pnd_clr	w	csbr_pnd clear
12:8	csbr_cnt	w/r	channel status bit 接收数目（byte 为单位）
7:4	reserve		
3	rle_dect_en	w/r	receive level error detect enable
2	inf_dma_en	w/r	information dma enable
1	dat_dma_md	w/r	0: 24bit

			1: 16bit
0	dat_dma_en	w/r	data dma enable

4.SS_DMA_LEN:inf_len + dat_len (32bits)

Bit	Name	RW	Description
31:16	inf_dma_len	w/r	信息 dma 的样点数
15:0	dat_dma_len	w/r	数据 dma 的样点数

第 23 章. MC_PWM

23. 1. 概述

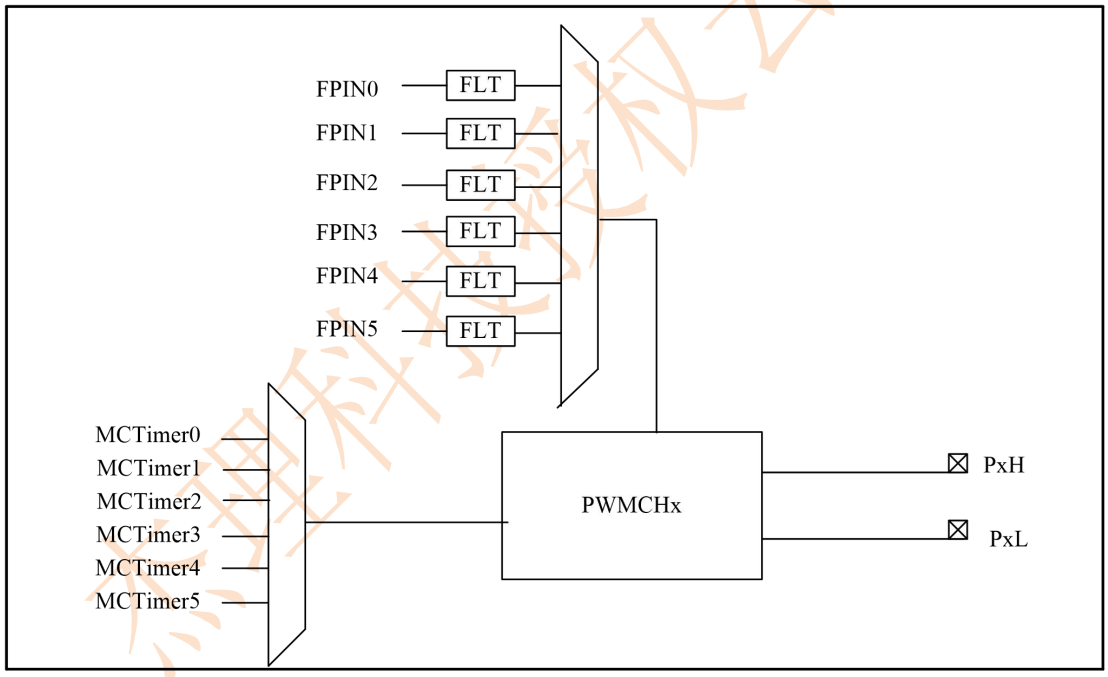
MCPWM 模块可针对性用于电机控制，也可以做普通 PWM 使用。

23. 2. 功能规格

6 对独立 PWM 通道，每对 PWM 有 H 和 L 两路输出，每对 PWM 中 H 和 L 共用同个定时器，H 和 L 输出有两种模式，一是死区功能打开时，H 和 L 是正反相关系带死区时间的两路 PWM；二是死区功能关闭时，H 和 L 可配为占空比不同的两路 PWM。6 对 PWM 共 6 个定时器，每对 PWM 用哪个定时器可配置。

6 路故障保护输入，可分别配置到某对 PWM，关闭该对 PWM 的输出。

框图如下：



23. 3. 定时器 MCTimer 0/1/2/3/4/5

23.3.1. 概述

MCTimer0/1/2/3/4/5 是功能相同的 16 位定时器。
可作为 PWM0/1/2/3/4/5 的时基控制

23.3.2. 模块特性

1)16 位定时功能
2)带缓冲的周期寄存器
3)支持多种工作模式：递增、递增-递减、外部引脚控制递增或递减
4)多种中断模式：上溢出中断、下溢出中断、上下溢出中断

6 个定时器功能相同，以下描述 MCTimer0 的寄存器，MCTimer1/2/3/4/5 的寄存器对应参照 MCTimer0。

23.3.3. 模块寄存器

1.MCTMR0_CON 计数/定时控制寄存器

Bit	Name	RW	Description
15	T0INCF	r	TMR0CNT 递增递减标志位 0：递减 1：递增
14	-	-	-
13	T0UFPND	r	TMR0CNT 递减借位标志 读 0：没借位 读 1：已发生借位
12	T0OFPND	r	TMR0CNT 溢出（TMR0CNT == TMR0PR）标志 读 0：没溢出 读 1：已发生溢出
11	T0UFCLR	w	写“1”清除 T0UFPND 标志位，读为“0”
10	T0OFCLR	w	写“1”清除 T0OFPND 标志位，读为“0”
9	T0UFIE	w/r	TMR0CNT 定时递减借位中断控制 0：禁止 1：允许
8	T0OFIE	w/r	TMR0CNT 定时溢出（TMR0CNT == TMR0PR）中断控制 0：禁止 1：允许
7	T0CKSRC	w/r	定时器时钟源 0：内部时钟 1：外部引脚时钟 TMR0CK
6:3	T0CKPS	w/r	时钟预分频设置，TCK/（2^TCKPS）
2	-	-	-
1:0	T0MODE	w/r	定时器工作模式 00：保持 TMR0CNT 01：递增模式 10：递增-递减循环模式

MCTMR0_PR：带缓冲的 16 位周期寄存器
MCTMR0_CNT：16 位定时/计数器

23. 4. PWM 0/1/2/3/4/5

23.4.1. 概述

内置 6 对 PWM 模块。

23.4.2. 模块引脚

PWM0：PWMCH0H、PWMCH0L
PWM1：PWMCH1H、PWMCH1L
PWM2：PWMCH2H、PWMCH2L
PWM3：PWMCH3H、PWMCH3L
PWM4：PWMCH4H、PWMCH4L
PWM5：PWMCH5H、PWMCH5L

故障输入引脚：FPIN0、FPIN1、FPIN2、FPIN3、FPIN4、FPIN5

23.4.3. 模块特性

1)边沿对齐和中心对齐输出模式
2)可运行过程中更改 PWM 频率、占空比
3)多种更新频率/占空比模式：上溢出重载、下溢出重载、上下溢出重载
4)灵活配置每对通道的有效电平状态
5)可编程死区控制
6)可选硬件故障输入引脚

23.4.4. 模块控制寄存器

PWM0~5 功能相同，以下描述 PWM0 的寄存器，PWM0~5 的寄存器对应参照 PWM0

1.MCCH0_CON0：PWM 控制寄存器 0

Bit	Name	RW	Description
15:12	DTCKPS	w/r	死区时钟预分频，Tsys/（2^ DTCKPS）
11:7	DTPR	w/r	死区时间控制，Tsys/(2^ DTCKPS) × (DTPR+1)
6	DTEN	w/r	死区允许控制，对应 PWMCH0H、PWMCH0L 0：禁止 1：允许

5	L_INV	w/r	对应 PWMCH0L 输出反向控制 0：反向禁止 1：反向允许
4	H_INV	w/r	对应 PWMCH0H 输出反向控制 0：反向禁止 1：反向允许
3	L_EN	w/r	对应 PWMCH0L 输出允许控制 0：禁止 PWMCH0L 1：允许 PWMCH0L
2	H_EN	w/r	对应 PWMCH0H 输出允许控制 0：禁止 PWMCH0H 1：允许 PWMCH0H
1:0	CMP_LD	w/r	MCCH0_CMP 重新载入控制 00：时基 MCTMRx_CNT 等于“0”载入 01：时基 MCTMRx_CNT 等于“0”或者等于 MCTMRx_PR 时候载入 10：时基 MCTMRx_CNT 等于 MCTMRx_PR 时候载入 11：立即载入

2.MCCH0_CON1：PWM 控制寄存器 1

Bit	Name	RW	Description
15	FPND	r	故障保护输入标志 读 0：未发生保护 读 1：已发生保护，模块的 PWM 引脚会变成高阻态
14	FCLR	w	清除 FPND 标志位，只写，读为“0” 写 0：无效 写 1：清除 FPND
13:12	-	-	-
11	INTEN	w/r	FPND 中断允许 0：禁止 1：允许
10:8	TMRSEL	w/r	选择 MCTMRx 作为 PWM 时基 0：选择 MCTimer0 1：选择 MCTimer1 2：选择 MCTimer2 3：选择 MCTimer3 4：选择 MCTimer4 5：选择 MCTimer5 6：- 7：-
7:5	-	-	-
4	FPINEN	w/r	故障保护输入允许控制

			0：禁止保护 1：允许保护
3	FPINAUTO	w/r	故障自动保护控制，当检测到故障引脚有效信号时，自动把 PWM 引脚设成高阻态，直到软件清除 FPND 0：禁止自动保护 1：允许自动保护
2:0	FPINSEL	w/r	故障保护输入引脚选择 0：选择 FPIN0 作为故障保护输入 1：选择 FPIN1 作为故障保护输入 2：选择 FPIN2 作为故障保护输入 3：选择 FPIN3 作为故障保护输入 4：选择 FPIN4 作为故障保护输入 5：选择 FPIN5 作为故障保护输入 6：- 7：-

MCCH0_CMPH：
带缓冲的 16 位比较寄存器，对应 PWMCH0H 引脚的占空比控制

MCCH0_CMPL：
带缓冲的 16 位比较寄存器，对应 PWMCH0L 引脚的占空比控制，死区功能关闭时有效

3.MCFPIN_CON：FPIN0/1/2/3/4/5 滤波控制寄存器，复位值：0x00000

Bit	Name	RW	Description
23	-	-	-
22	-	-	-
21	EDGE_5	w/r	FPIN5 边沿选择。0：下降沿， 1：上升沿
20	EDGE_4	w/r	FPIN4 边沿选择。0：下降沿， 1：上升沿
19	EDGE_3	w/r	FPIN3 边沿选择。0：下降沿， 1：上升沿
18	EDGE_2	w/r	FPIN2 边沿选择。0：下降沿， 1：上升沿
17	EDGE_1	w/r	FPIN1 边沿选择。0：下降沿， 1：上升沿
16	EDGE_0	w/r	FPIN0 边沿选择。0：下降沿， 1：上升沿
15	-	-	-
14	-	-	-
13	FLT_EN5	w/r	FPIN5 滤波使能开关。0：滤波关闭，1：滤波开启
12	FLT_EN4	w/r	FPIN4 滤波使能开关。0：滤波关闭，1：滤波开启
11	FLT_EN3	w/r	FPIN3 滤波使能开关。0：滤波关闭，1：滤波开启
10	FLT_EN2	w/r	FPIN2 滤波使能开关。0：滤波关闭，1：滤波开启
9	FLT_EN1	w/r	FPIN1 滤波使能开关。0：滤波关闭，1：滤波开启
8	FLT_EN0	w/r	FPIN0 滤波使能开关。0：滤波关闭，1：滤波开启

7:6	-	-	-
5:0	FLT_PR	w/r	滤波宽度选择，宽度=16 × FLT_PR × Thsb_clk

4.MCPWM_CON0 启动控制寄存器:

Bit	Name	RW	Description
15	-	-	-
14	-	-	-
13	T5EN	w/r	定时器 5 计数开关控制，0：关闭，1：开启
12	T4EN	w/r	定时器 4 计数开关控制，0：关闭，1：开启
11	T3EN	w/r	定时器 3 计数开关控制，0：关闭，1：开启
10	T2EN	w/r	定时器 2 计数开关控制，0：关闭，1：开启
9	T1EN	w/r	定时器 1 计数开关控制，0：关闭，1：开启
8	T0EN	w/r	定时器 0 计数开关控制，0：关闭，1：开启
7	-	-	-
6	-	-	-
5	PWM5EN	w/r	PWM5 模块开关控制，0：关闭，1：开启
4	PWM4EN	w/r	PWM4 模块开关控制，0：关闭，1：开启
3	PWM3EN	w/r	PWM3 模块开关控制，0：关闭，1：开启
2	PWM2EN	w/r	PWM2 模块开关控制，0：关闭，1：开启
1	PWM1EN	w/r	PWM1 模块开关控制，0：关闭，1：开启
0	PWM0EN	w/r	PWM0 模块开关控制，0：关闭，1：开启

23. 5. 使用说明

定时器递增模式下：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）)。

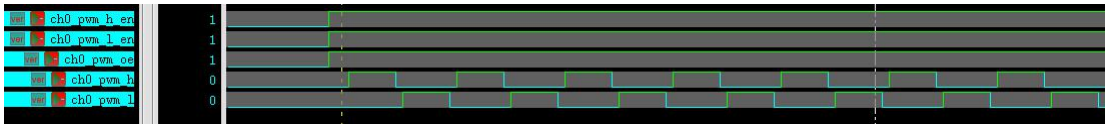
定时器递增递减模式下：PWM 输出的占空比 N 的计算公式如下：

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

PWM 行为：计时器计到 0 时开始翻转为高电平，计时器增大计数到值 CMP 时翻转为低电平，待 计时到 PR 时开始递减，计时器递减计数到值 CMP 时翻转为高电平，计到 0 时后开始增大计数，增大到值 CMP 时翻转为低电平，以此往复

输出 PWM 如下：

1)死区功能打开：chx_pwm_h 和 chx_pwm_l 输出如下，此时可以再配置 L_INV,H_INV 使信号反向。



2)死区功能关闭：chx_pwm_h 和 chx_pwm_l 输出如下，此时可以通过配置 L_INV,H_INV 使信号反向或配置 CMPH 和 CMPL 其占空比不同。