密级状态：绝密(　　)　　秘密(　　)　　内部资料(　　)　　公开(√)

目前平台上 pmu 的相关使用说明

(技术研发部，手机组)

文件状态： ［　］草稿 ［√］正式发布 ［　］正在修改	文件标识：	目前平台上 pmu 的相关使用说明
	当前版本：	1.1
	作　　者：	张晴
	完成日期：	2012-8-16

版 本 历 史

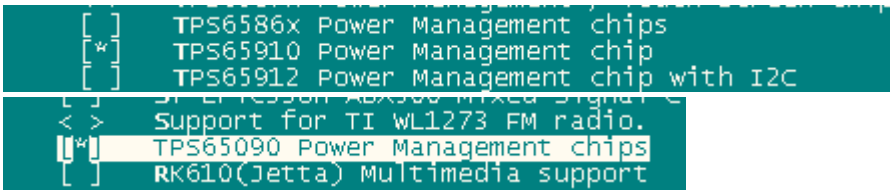
版本号	作者	修改日期	修改说明
1.0	张晴	2012-7-27	初稿
1.1	张晴	2012-8-16	

目录

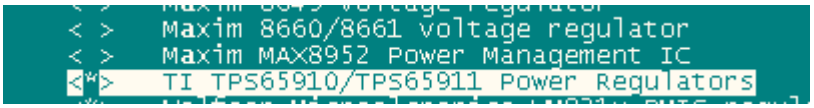
1 Tps65910 使用说明（不兼容其他 pmu）

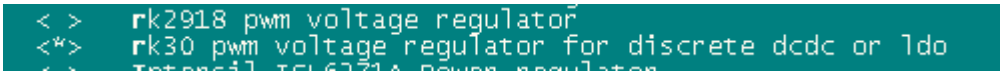
1.1 配置说明

[*] Multifunction device drivers --->



[*] Voltage and Current Regulator Support --->





（使用 pwm 调整电压的需要打开宏 rk30 pwm voltage regulator for discrete dcdc or ldo）

[*] Real Time Clock --->

1.2修改各路初始电压

pmu 在 eprom 中会默认配置一些电压的上电时序和默认输出电压， pmu 一上电先按照 eprom 中的上电时序和电压输出，然后发送 reset 信号给芯片，芯片启动后，在 kernel 中重新配置了 pmu 各路输出电压，如下：

在 board-rk30-sdk-tps65910.c 中的 tps65910_post_init()中：

```
dcdc = regulator_get(NULL, "vdd_cpu");     // vdd_arm
regulator_set_voltage(dcdc, 1100000, 1100000);     //设置运行下电压
regulator_enable(dcdc);
printk("%s set dcdc2 vdd_cpu(vdd_arm)=%dmV end\n", __func__, regulator_get_voltage(dcdc));
regulator_put(dcdc);
```

```
udelay(100);
ldo = regulator_get(NULL, "ldo1");      // vcc18_cif
regulator_set_voltage(ldo, 1800000, 1800000);
regulator_enable(ldo);
//      printk("%s set ldo1  vcc18_cif=%dmV end\n", __func__, regulator_get_voltage(ldo));
regulator_put(ldo);
udelay(100);
```

1.3 休眠唤醒

1.3.1 pmu 进入 sleep 模式

在 board-rk30-sdk.c 中：

```
void __sramfunc board_pmu_suspend(void)   //通用接口，所有的方案都使用，在 pm.c 中被调用
{
        board_pmu_tps65910_suspend();
}
void __sramfunc board_pmu_resume(void)
{
         board_pmu_tps65910_resume();
}
```

Pmu 各自的休眠函数在各自的版级文件实现（board-rk30-sdk-tps65910.c）

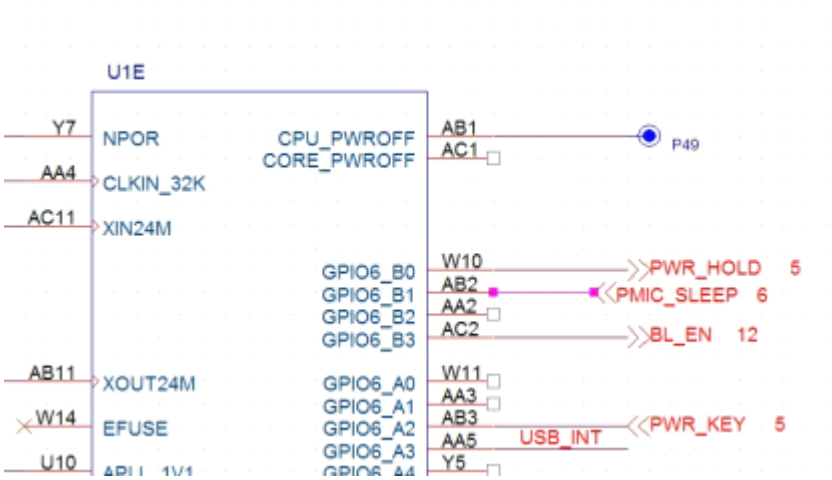
```
void __sramfunc board_pmu_tps65910_suspend(void)
{
    grf_writel(GPIO6_PB1_DIR_OUT, GRF_GPIO6L_DIR_ADDR);
    grf_writel(GPIO6_PB1_DO_HIGH, GRF_GPIO6L_DO_ADDR);   //set gpio6_b1  output high
    grf_writel(GPIO6_PB1_EN_MASK, GRF_GPIO6L_EN_ADDR);
```

```
}
void __sramfunc board_pmu_tps65910_resume(void)
{
    grf_writel(GPIO6_PB1_DIR_OUT, GRF_GPIO6L_DIR_ADDR);
    grf_writel(GPIO6_PB1_DO_LOW, GRF_GPIO6L_DO_ADDR);   //set gpio6_b1 output low
    grf_writel(GPIO6_PB1_EN_MASK, GRF_GPIO6L_EN_ADDR);
    #ifdef CONFIG_CLK_SWITCH_TO_32K                     //switch clk to 24M
    sram_32k_udelay(10000);
    #else
    sram_udelay(2000);
    #endif
}
```

控制 pmu 进入 sleep 的 io 口根据原理图确认。

Pmu 进入 sleep 模式有几种方式：直接通过 i2c 写寄存器的状态位；通过操作 gpio 口。目前使用的方案中都是通过操作 gpio 实现。对应 pmic_sleep(或者 pmu_sleep)

很多方案中 pmu 的 sleep 和 24M 晶振切换是同一个 gpio 口，如果使用同一个必须要保证 24M 切换的功能有打开。（打开宏 CONFIG_CLK_SWITCH_TO_32K）



1.3.2 在休眠状态下关闭一些 ldo 或者降低各路休眠电压

（1）关闭一些 ldo：

在 board-rk30-sdk-tps65910.c 的 tps65910_pre_init（）中：

```
/*close ldo when in  sleep mode */
        val = tps65910_reg_read(tps65910, TPS65910_SLEEP_SET_LDO_OFF);
        if (val<0) {
                printk(KERN_ERR "Unable to read TPS65910_DCDCCTRL reg\n");
                return val;
        }
    val |= 0x9b;
    err = tps65910_reg_write(tps65910, TPS65910_SLEEP_SET_LDO_OFF, val);
    if (err) {
        printk(KERN_ERR "Unable to read TPS65910 Reg at offset 0x%x= \
                \n", TPS65910_VDIG1);
        return err;
    }
```

红色字体代表关闭 ldo，这个需要配合如下 datasheet，具体：

Bits	Field Name	Description	Type	Reset
7	VDAC_SETOFF	When 1, LDO regulator is turned off during device SLEEP state. When 0, No effect	RW	0
6	VPLL_SETOFF	When 1, LDO regulator is turned off during device SLEEP state. When 0, No effect	RW	0
5	VAUX33_SETOFF	When 1, LDO regulator is turned off during device SLEEP state. When 0, No effect	RW	0
4	IVAUX2_SETOFF	When 1, LDO regulator is turned off during device SLEEP state. When 0, No effect	RW	0
3	VAUX1_SETOFF	When 1, LDO regulator is turned off during device SLEEP state. When 0, No effect	RW	0
2	VDIG2_SETOFF	When 1, LDO regulator is turned off during device SLEEP state. When 0, No effect	RW	0
1	VDIG1_SETOFF	When 1, LDO regulator is turned off during device SLEEP state. When 0, No effect	RW	0
0	VMMC_SETOFF	When 1, LDO regulator is turned off during device SLEEP state. When 0, No effect	RW	0

（2）降低各路休眠电压

由于 ti 的没有 sleep voltage 寄存器，所以无法通过 regulator_set_suspend_voltage（）接口实现休眠电压的设置。

目前降压操作使用 i2c-sram.c 在通过 i2c 操作寄存器实现在休眠的时候降低某些路的电压，具体操作如下：（打开配置， System Type ---> Support for RK power manage ---> Support i2c control interface in sram ）

```
void __sramfunc rk30_suspend_voltage_set(unsigned int vol)
{
    uint8 slaveaddr;
    uint16 slavereg;
    uint8 data,ret = 0;
    uint8 rtc_status_reg = 0x11;
    slaveaddr = I2C_SADDR;                  //slave device addr
    slavereg = 0x22;                  // reg addr
    data = 0x23;            //set arm 1.0v
    sram_i2c_init();   //init i2c device
    ret = sram_i2c_read(slaveaddr, rtc_status_reg);
    sram_i2c_write(slaveaddr, rtc_status_reg, ret);
    arm_voltage = sram_i2c_read(slaveaddr, slavereg);
//    sram_printhex(ret);
    sram_i2c_write(slaveaddr, slavereg, data);//
    sram_i2c_deinit();   //deinit i2c device
}
void __sramfunc rk30_suspend_voltage_resume(unsigned int vol)
{
    uint8 slaveaddr;
```

```
    uint16 slavereg;
    uint8 data,ret = 0;
    slaveaddr = I2C_SADDR;                    //slave device addr
    slavereg = 0x22;                    // reg addr
    sram_i2c_init();    //init i2c device
    if (arm_voltage >= 0x3b ){      // set arm <= 1.3v
        data = 0x3b;
    }
    else if(arm_voltage <= 0x1f){
        data = 0x1f;                    // set arm >= 0.95v
    }
    else
        data = arm_voltage;
    sram_i2c_write(slaveaddr, slavereg, data);
    sram_i2c_deinit();    //deinit i2c device
}
```

休眠电压设置见红色字体，唤醒时电压为休眠前的电压。

Table 45. VDD1_OP_REG

Address Offset	0x22
Physical Address	Instance
Description	VDD1 voltage selection register. This register can be accessed by both control and smartreflex I²C interfaces depending on SR_CTL_I2C_SEL register bit value.
Type	RW

7	6	5	4	3	2	1	0
CMD	SEL						

Bits	Field Name	Description	Type	Reset
7	CMD	Smart-Reflex command: when 0: VDD1_OP_REG voltage is applied when 1: VDD1_SR_REG voltage is applied	RW	0
6:0	SEL	Output voltage (EEPROM bits) selection with GAIN_SEL = 00 (G = 1, 12.5 mV per LSB): SEL[6:0] = 1001011 to 1111111 : 1.5 V ... SEL[6:0] = 0111111 : 1.35 V ... SEL[6:0] = 0110011 : 1.2 V ... SEL[6:0] = 0000001 to 0000011 : 0.6 V SEL[6:0] = 0000000 : Off (0.0 V) Note: from SEL[6:0] = 3 to 75 (dec) Vout = (SEL[6:0] × 12.5 mV + 0.5625 mV) × G	RW	See (1)

(1) The reset value for this field varies with boot mode selection and the processor support. Please refer to the corresponding processor user guide to find the correct default value.

1.4 Arm 和 logic 的动态调节

Arm 路采用 pmu 输出，在 pmu 中注册相应 name -"vdd_cpu",而 logic 采用 pwm 调节占空比实现 logic 电压的调整，Logic 路注册 name - "vdd_core"，dvfs 调整 arm 和 logic 会自动通过搜索"vdd_cpu"、"vdd_core" 的 name 实现调压操作,具体：

以 Pwm 调整 logic 为例：

在 board-rk30-sdk.c 中：

```
#if CONFIG_RK30_PWM_REGULATOR
const static int pwm_voltage_map[] = {
    1000000, 1025000, 1050000, 1075000, 1100000, 1125000, 1150000, 1175000, 1200000, 1225000, 1250000, 1275000, 1300000, 1325000, 1350000, 1375000, 1400000
};
static struct regulator_consumer_supply pwm_dcdc1_consumers[] = {
    {
        .supply = "vdd_core",
    }
```

```
};
struct regulator_init_data pwm_regulator_init_dcdc[1] =
{
    {
        .constraints = {
            .name = "PWM_DCDC1",
            .min_uV = 600000,
            .max_uV = 1800000,      //0.6-1.8V
            .apply_uV = true,
            .valid_ops_mask          =          REGULATOR_CHANGE_STATUS          |
REGULATOR_CHANGE_VOLTAGE,
        },
        .num_consumer_supplies = ARRAY_SIZE(pwm_dcdc1_consumers),
        .consumer_supplies = pwm_dcdc1_consumers,
    },
};
static struct pwm_platform_data pwm_regulator_info[1] = {
    {
        .pwm_id = 3,
        .pwm_gpio = RK30_PIN0_PD7,              //根据各自原理图配置
        .pwm_iomux_name = GPIO0D7_PWM3_NAME,
        .pwm_iomux_pwm = GPIO0D_PWM3,
        .pwm_iomux_gpio = GPIO0D_GPIO0D6,
        .pwm_voltage = 1100000,
        .suspend_voltage = 1050000,
        .min_uV = 1000000,
```

```
        .max_uV = 1400000,
        .coefficient = 455,   //45.5%
        .pwm_voltage_map = pwm_voltage_map,
        .init_data = &pwm_regulator_init_dcdc[0],
    },
};

struct platform_device pwm_regulator_device[1] = {
    {
        .name = "pwm-voltage-regulator",
        .id = 0,
        .dev        = {
            .platform_data = &pwm_regulator_info[0],
        }
    },
};
#endif
```

Pwm 设备的添加在 board-rk30-sdk-tps65910.c 的 tps65910_post_init 中：

```
    #ifdef CONFIG_RK30_PWM_REGULATOR
    platform_device_register(&pwm_regulator_device[0]);
    #endif
```

1.5关机流程

Pmu 的关机流程在 board-rk30-sdk.c 中实现，具体如下：

```
#define POWER_ON_PIN RK30_PIN6_PB0      //power_hold
static void rk30_pm_power_off(void)
```

```
{
    printk(KERN_ERR "rk30_pm_power_off start...\n");
    gpio_direction_output(POWER_ON_PIN, GPIO_LOW);
    #if defined(CONFIG_MFD_TPS65910)
        tps65910_device_shutdown();//tps65910 shutdown
    #endif
    while (1);
}
```

2 Wm8326 使用说明（不兼容其他 pmu）

2.1 配置说明

[*] Multifunction device drivers --->

[*] Voltage and Current Regulator Support --->

[*] Real Time Clock --->

2.2 修改各路初始电压

在 board-rk30-sdk-wm8326.c 中的 wm8326_post_init()中：

```
dcdc = regulator_get(NULL, "vdd_cpu");     // vdd_arm
regulator_set_voltage(dcdc, 1100000, 1100000);     //设置运行下电压
```

```
regulator_set_suspend_voltage(dcdc, 1000000);     //设置休眠后电压
regulator_enable(dcdc);
printk("%s set dcdc2 vdd_cpu(vdd_arm)=%dmV end\n", __func__, regulator_get_voltage(dcdc));
regulator_put(dcdc);
udelay(100);
```

2.3 休眠唤醒

2.3.1 pmu 进入 sleep 模式

在 board-rk30-sdk.c 中：

```
void __sramfunc board_pmu_suspend(void)   //通用接口，所有的方案都使用，在 pm.c 中被调用
{
        board_pmu_wm8326_suspend();
}
void __sramfunc board_pmu_resume(void)
{
         board_pmu_wm8326_resume();
}
```

Pmu 各自的休眠函数在各自的版级文件实现（board-rk30-sdk-wm8326.c）

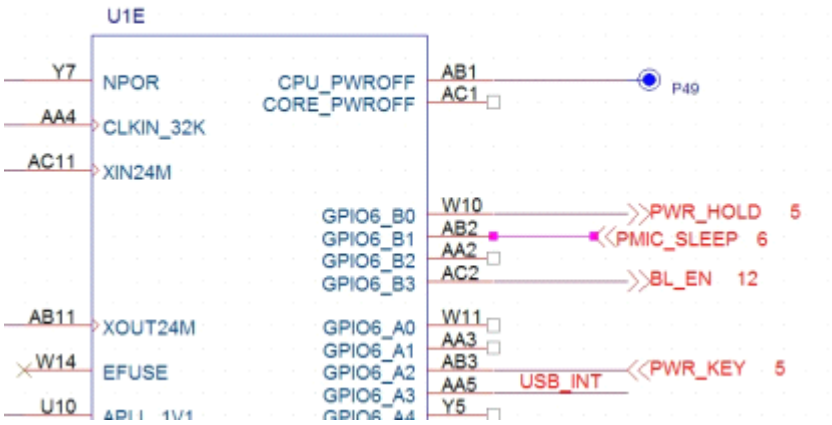
```
void __sramfunc board_pmu_wm8326_suspend(void)
{
    cru_writel(CRU_CLKGATE5_GRFCLK_ON,CRU_CLKGATE5_CON_ADDR); //open grf clk
    grf_writel(GPIO6_PB1_DIR_OUT, GRF_GPIO6L_DIR_ADDR);
    grf_writel(GPIO6_PB1_DO_HIGH, GRF_GPIO6L_DO_ADDR);   //set gpio6_b1  output high
    grf_writel(GPIO6_PB1_EN_MASK, GRF_GPIO6L_EN_ADDR);
}
```

```
void __sramfunc board_pmu_wm8326_resume(void)
{
    grf_writel(GPIO6_PB1_DIR_OUT, GRF_GPIO6L_DIR_ADDR);
    grf_writel(GPIO6_PB1_DO_LOW, GRF_GPIO6L_DO_ADDR);        //set gpio6_b1 output high
    grf_writel(GPIO6_PB1_EN_MASK, GRF_GPIO6L_EN_ADDR);
#ifdef CONFIG_CLK_SWITCH_TO_32K
    sram_32k_udelay(10000);
#else
    sram_udelay(10000);
#endif
}
```

控制 pmu 进入 sleep 的 io 口根据原理图确认。

Pmu 进入 sleep 模式有几种方式：直接通过 i2c 写寄存器的状态位；通过操作 gpio 口。目前使用的方案中都是通过操作 gpio 实现。对应 pmic_sleep(或者 pmu_sleep)

很多方案中 pmu 的 sleep 和 24M 晶振切换是同一个 gpio 口，如果使用同一个必须要保证 24M 切换的功能有打开。（打开宏 CONFIG_CLK_SWITCH_TO_32K）



2.3.2 在休眠状态下关闭一些 ldo 或者降低各路休眠电压

（1）关闭一些 ldo

在 board-rk30-sdk-wm8326.c 的 wm831x_pre_init（）中：

wm831x_set_bits(parm,WM831X_LDO5_SLEEP_CONTROL ,0xe000,0x2000);// set ldo5 is disable in sleep mode

如果关闭 dcdc，必须保证此 dcdc 没有作为其他 ldo 的输入，如果作为其他 ldo 的输入请注意要先关闭那些 ldo 再关闭此路 dcdc，并且上电的时候要保证 dcdc 先上电。

举例，dc4，作为 ldo1、ldo2、ldo3 的输入：

wm831x_set_bits(parm,WM831X_LDO1_SLEEP_CONTROL ,0xe000,0x3000);

wm831x_set_bits(parm,WM831X_LDO3_SLEEP_CONTROL ,0xe000,0x2000);

wm831x_set_bits(parm,WM831X_LDO2_SLEEP_CONTROL ,0xe000,0x3000);

wm831x_set_bits(parm,WM832X_DC4_SLEEP_CONTROL ,0xe000,0x2000);

红色字体代表关闭时序，这个需要配合各路上电时序确定，具体：

15:13	DC1_SLP_SLOT [2:0]	000	DC-DC1 SLEEP Slot select 000 = SLEEP voltage / operating mode transition in Timeslot 5 001 = Disable in Timeslot 5 010 = Disable in Timeslot 4 011 = Disable in Timeslot 3 100 = Disable in Timeslot 2 101 = Disable in Timeslot 1 110 = SLEEP voltage / operating mode transition in Timeslot 3 111 = SLEEP voltage / operating mode transition in Timeslot 1 If DC-DC1 is assigned to a Hardware Enable Input, then codes 001-101 select in which timeslot the converter enters its SLEEP condition.

（2）降低各路休眠电压

在 board-rk30-sdk-wm8326.c 的 post_init 中：

```
    ldo = regulator_get(NULL, "ldo5");      //vcc_25
    regulator_set_voltage(ldo, 2500000, 2500000);
    regulator_set_suspend_voltage(ldo, 2500000);
    regulator_enable(ldo);
//    printk("%s set ldo5 vcc_25=%dmV end\n", __func__, regulator_get_voltage(ldo));
```

```
regulator_put(ldo);
dcdc = regulator_get(NULL, "vdd_cpu");     // vdd_arm
regulator_set_voltage(dcdc, 1100000, 1100000);
regulator_set_suspend_voltage(dcdc, 1000000);
regulator_enable(dcdc);
printk("%s set dcdc2 vdd_cpu(vdd_arm)=%dmV end\n", __func__, regulator_get_voltage(dcdc));
regulator_put(dcdc);
udelay(100);
```

2.4 Arm 和 logic 的动态调节

Arm 和 logic 都采用 pmu 输出，在 pmu 中注册相应 name -"vdd_cpu"、"vdd_core"，dvfs 调整 arm 和 logic 会自动通过搜索"vdd_cpu"、"vdd_core" 的 name 实现调压操作。

2.5低电检测

Wolfson 的低电通过 i2c 配置，在 rk30 项目中使用 wm8326 的低电配置方法如下：

在 board-rk30-sdk-wm8326.c 的 static int wm831x_low_power_detection()中：

打开宏 CONFIG_WM8326_VBAT_LOW_DETECTION

1、单电池

采用判断 pvdd 电压，设置低电阈值，打开宏 CONFIG_BATTERY_RK30_VOL3V8

```
#ifdef CONFIG_BATTERY_RK30_VOL3V8
    wm831x_reg_write(wm831x,WM831X_SYSTEM_INTERRUPTS_MASK,0xbe5c);
    wm831x_set_bits(wm831x,WM831X_INTERRUPT_STATUS_1_MASK,0x8000,0x0000);
    wm831x_set_bits(wm831x,WM831X_SYSVDD_CONTROL ,0xc077,0x0035);          //set  pvdd
low voltage is 3.1v hi voltage is 3.3v
    #else
```

低电电压修改方法（0x0035）：

ADDRESS	BIT	LABEL	DEFAULT	DESCRIPTION
R16385 (4001h) PVDD Control	15:14	SYSLO_ERR_ACT	00	SYSLO Error Action Selects the action taken when SYSLO is asserted 00 = Interrupt 01 = WAKE transition 10 = Reserved 11 = OFF transition
	11	SYSLO_STS	0	SYSLO Status 0 = Normal 1 = PVDD is below SYSLO threshold
	6:4	SYSLO_THR [2:0]	010	SYSLO threshold (falling PVDD) This is the falling PVDD voltage at which SYSLO will be asserted 000 = 2.8V 001 = 2.9V ... 111 = 3.5V
	2:0	SYSOK_THR [2:0]	101	SYSOK threshold (rising PVDD) This is the rising PVDD voltage at which SYSLO will be de-asserted 000 = 2.8V 001 = 2.9V ... 111 = 3.5V

2、双电池

采用从 vbat 分压，通过 pmic 的 adc 采集，设置阈值

```
wm831x_reg_write(wm831x,WM831X_AUXADC_CONTROL,0x803f);        //open adc
wm831x_reg_write(wm831x,WM831X_AUXADC_CONTROL,0xd03f);
wm831x_reg_write(wm831x,WM831X_AUXADC_SOURCE,0x0001);
wm831x_reg_write(wm831x,WM831X_COMPARATOR_CONTROL,0x0001);
wm831x_reg_write(wm831x,WM831X_COMPARATOR_1,0x2844);     //set the low power is 3.1v
wm831x_reg_write(wm831x,WM831X_INTERRUPT_STATUS_1_MASK,0x99ee);
wm831x_set_bits(wm831x,WM831X_SYSTEM_INTERRUPTS_MASK,0x0100,0x0000);
if (wm831x_reg_read(wm831x,WM831X_AUXADC_DATA)< 0x1844){
    printk("The vbat is too low.\n");
    wm831x_device_shutdown(wm831x);
}
#endif
```

低压电压修改方法（0x0844）：

Voltage (mV) = AUX_DATA x 1.465

REGISTER ADDRESS	BIT	LABEL	DEFAULT	DESCRIPTION	REFER T
R16433 4031h) Comparator 1	15:13	DCMP1_SRC [2:0]	000	Digital Comparator 1 source select 0 = Reserved 1 = GPIO10 2 = GPIO11 3 = GPIO12 4 = Reserved 5 = Chip Temperature 6 = Reserved 7 = PVDD voltage	
	12	DCMP1_GT	0	Digital Comparator 1 interrupt control 0 = interrupt when less than threshold 1 = interrupt when greater than or equal to threshold	
	11:0	DCMP1_THR [11:0]	0000_0000 _0000	Digital Comparator 1 threshold (12-bit unsigned binary number; coding is the same as AUX_DATA)	

2.6关机流程

Pmu 的关机流程在 board-rk30-sdk.c 中实现，具体如下：

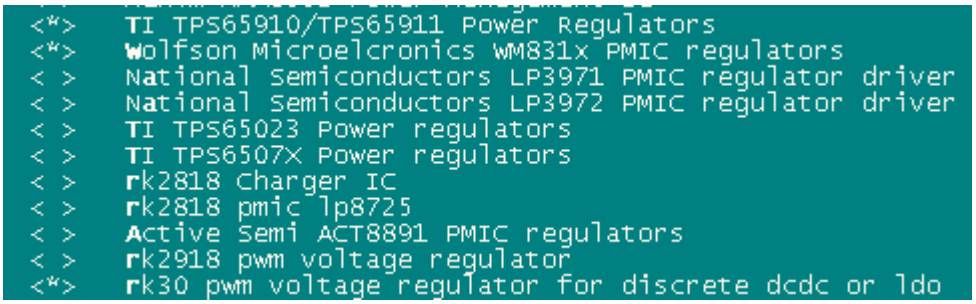
```
#define POWER_ON_PIN RK30_PIN6_PB0      //power_hold
static void rk30_pm_power_off(void)
{
    printk(KERN_ERR "rk30_pm_power_off start...\n");
    gpio_direction_output(POWER_ON_PIN, GPIO_LOW);
    #if defined(CONFIG_MFD_WM831X)
        wm831x_set_bits(Wm831x,WM831X_GPIO_LEVEL,0x0001,0x0000);    //set sys_pwr 0
        wm831x_device_shutdown(Wm831x);//wm8326 shutdown
    #endif
    while (1);
}
```

3 Wm8326 兼容 tps65910 使用说明

3.1 配置说明

[*] Multifunction device drivers --->

[*] Voltage and Current Regulator Support --->



（使用 pwm 调整电压的需要打开宏 rk30 pwm voltage regulator for discrete dcdc or ldo）

[*] Real Time Clock --->

3.2 修改各路初始电压

在 board-rk30-sdk-wm8326.c、board-rk30-sdk-tps65910.c 中的 post_init()中：

```
dcdc = regulator_get(NULL, "vdd_cpu");     // vdd_arm
regulator_set_voltage(dcdc, 1100000, 1100000);     //设置运行下电压
regulator_enable(dcdc);
printk("%s set dcdc2 vdd_cpu(vdd_arm)=%dmV end\n", __func__, regulator_get_voltage(dcdc));
```

```
    regulator_put(dcdc);
    udelay(100);
```

3.3 休眠唤醒

3.3.1 pmu 进入 sleep 模式

在 board-rk30-sdk.c 中：

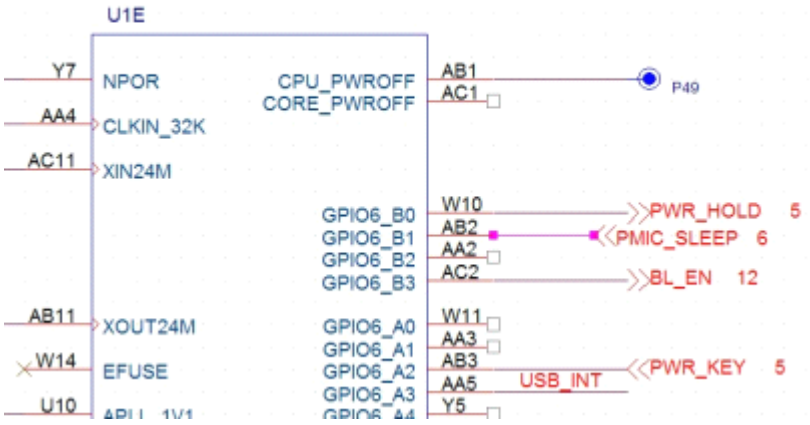
定义可一个全局变量：int __sramdata g_pmic_type = 0;

```
void __sramfunc board_pmu_suspend(void)   //通用接口，所有的方案都使用，在 pm.c 中被调用
{
    #if defined (CONFIG_MFD_WM831X_I2C)
        if(g_pmic_type == PMIC_TYPE_WM8326)
        board_pmu_wm8326_suspend();
    #endif
    #if defined (CONFIG_MFD_TPS65910)
        if(g_pmic_type == PMIC_TYPE_TPS65910)
        board_pmu_tps65910_suspend();
    #endif
}
void __sramfunc board_pmu_resume(void)
{
    #if defined (CONFIG_MFD_WM831X_I2C)
        if(g_pmic_type == PMIC_TYPE_WM8326)
        board_pmu_wm8326_resume();
    #endif
    #if defined (CONFIG_MFD_TPS65910)
```

```
        if(g_pmic_type == PMIC_TYPE_TPS65910)
        board_pmu_tps65910_resume();
    #endif
}
```

Pmu 各自的休眠函数在各自的版级文件实现 board-rk30-sdk-wm8326.c、board-rk30-sdk-tps65910.c（详细见各自的休眠处理中）

很多方案中 pmu 的 sleep 和 24M 晶振切换是同一个 gpio 口，如果使用同一个必须要保证 24M 切换的功能有打开。（打开宏 CONFIG_CLK_SWITCH_TO_32K）



3.3.2 在休眠状态下关闭一些 ldo 或者降低各路休眠电压

见 Pmu 各自的相关处理中。

3.4 Arm 和 logic 的动态调节

Arm 路采用 pmu 输出，在 pmu 中注册相应 name -"vdd_cpu",而 logic 采用 pwm 调节占空比实现 logic 电压的调整，Logic 路注册 name - "vdd_core"，dvfs 调整 arm 和 logic 会自动通过搜索"vdd_cpu"、"vdd_core" 的 name 实现调压操作,

但是 wm8326 不需要使用 pwm 调整 logci，所以此部分有所不同：具体：

在 board-rk30-sdk.c 中仍注册 pwm 接口：

```
#if CONFIG_RK30_PWM_REGULATOR
const static int pwm_voltage_map[] = {
    1000000, 1025000, 1050000, 1075000, 1100000, 1125000, 1150000, 1175000, 1200000, 1225000,
1250000, 1275000, 1300000, 1325000, 1350000, 1375000, 1400000
};
static struct regulator_consumer_supply pwm_dcdc1_consumers[] = {
    {
        .supply = "vdd_core",
    }
};
struct regulator_init_data pwm_regulator_init_dcdc[1] =
{
    {
        .constraints = {
            .name = "PWM_DCDC1",
            .min_uV = 600000,
            .max_uV = 1800000,    //0.6-1.8V
            .apply_uV = true,
            .valid_ops_mask = REGULATOR_CHANGE_STATUS |
REGULATOR_CHANGE_VOLTAGE,
        },
        .num_consumer_supplies = ARRAY_SIZE(pwm_dcdc1_consumers),
        .consumer_supplies = pwm_dcdc1_consumers,
    },
};
static struct pwm_platform_data pwm_regulator_info[1] = {
```

```
    {
        .pwm_id = 3,
        .pwm_gpio = RK30_PIN0_PD7,              //根据各自原理图配置
        .pwm_iomux_name = GPIO0D7_PWM3_NAME,
        .pwm_iomux_pwm = GPIO0D_PWM3,
        .pwm_iomux_gpio = GPIO0D_GPIO0D6,
        .pwm_voltage = 1100000,
        .suspend_voltage = 1050000,
        .min_uV = 1000000,
        .max_uV = 1400000,
        .coefficient = 455,   //45.5%
        .pwm_voltage_map = pwm_voltage_map,
        .init_data = &pwm_regulator_init_dcdc[0],
    },
};

struct platform_device pwm_regulator_device[1] = {
    {
        .name = "pwm-voltage-regulator",
        .id = 0,
        .dev        = {
            .platform_data = &pwm_regulator_info[0],
        }
    },
};
#endif
```

Pwm 设备的添加在 board-rk30-sdk-tps65910.c 的 tps65910_post_init 中：

```
g_pmic_type = PMIC_TYPE_TPS65910;
printk("%s:g_pmic_type=%d\n",__func__,g_pmic_type);
#ifdef CONFIG_RK30_PWM_REGULATOR
platform_device_register(&pwm_regulator_device[0]);
#endif
```

注意：必须保证挂载上的是 tps65910，才可以注册 pwm 的 device,如果挂载设备是 wm8326 即不注册设备，否则会出现 vdd_core 的 name 重名，导致 dvfs 调整的时候 logic 电压调整出错。

3.5 Wm8326 一级休眠接口

一级休眠接口主要用于在系统在进入一级休眠后，通过设置 pmic 的 dcdc 和 ldo 的工作模式，设置为低功耗模式，来提高转换效率，降低功耗，也可以关闭一些在一级休眠下不使用的 ldo。

在 board0-rk30-sdk-wm8326.c 中：

```
#ifdef CONFIG_HAS_EARLYSUSPEND
void wm831x_pmu_early_suspend(struct regulator_dev *rdev)
{   ……
    dcdc = regulator_get(NULL, "dcdc4"); //vcc_io
    regulator_set_voltage(dcdc, 2800000, 2800000);
    regulator_set_mode(dcdc, REGULATOR_MODE_STANDBY);   //设置为低功耗模式
    regulator_enable(dcdc);
    printk("%s set dcdc4 vcc_io=%dmV end\n", __func__, regulator_get_voltage(dcdc));
    regulator_put(dcdc);
    udelay(100);

    ldo = regulator_get(NULL, "ldo1");      //
    regulator_set_mode(ldo, REGULATOR_MODE_IDLE);
```

```
        regulator_enable(ldo);
        regulator_put(ldo);
        udelay(100);
        ……
}
void wm831x_pmu_early_resume(struct regulator_dev *rdev)
{       ……
        dcdc = regulator_get(NULL, "dcdc4");  //vcc_io
        regulator_set_voltage(dcdc, 3000000, 3000000);
        regulator_set_mode(dcdc, REGULATOR_MODE_FAST);   //恢复成 fccm 模式（pwm 模式）
        regulator_enable(dcdc);
        printk("%s set dcdc4 vcc_io=%dmV end\n", __func__, regulator_get_voltage(dcdc));
        regulator_put(dcdc);
        udelay(100);

        ldo = regulator_get(NULL, "ldo1");      //
        regulator_set_mode(ldo, REGULATOR_MODE_NORMAL);
        regulator_enable(ldo);
        regulator_put(ldo);
        udelay(100);
……
}
#else
void wm831x_pmu_early_suspend(struct regulator_dev *rdev)
{
}
```

```
void wm831x_pmu_early_resume(struct regulator_dev *rdev)
{
}
#endif
```

3.6关机流程

Pmu 的关机流程在 board-rk30-sdk.c 中实现，具体如下：

```
static void rk30_pm_power_off(void)
{
    printk(KERN_ERR "rk30_pm_power_off start...\n");
    gpio_direction_output(POWER_ON_PIN, GPIO_LOW);
    #if defined(CONFIG_MFD_WM831X)
    if(g_pmic_type == PMIC_TYPE_WM8326)
    {
        wm831x_set_bits(Wm831x,WM831X_GPIO_LEVEL,0x0001,0x0000);    //set sys_pwr 0
        wm831x_device_shutdown(Wm831x);//wm8326 shutdown
    }
    #endif
    #if defined(CONFIG_MFD_TPS65910)
    if(g_pmic_type == PMIC_TYPE_TPS65910)
    {
        tps65910_device_shutdown();//tps65910 shutdown
    }
    #endif

    while (1);
```

}

4 不使用 pmu 采用分立 dcdc 和 ldo 的使用说明

4.1 配置说明

[*] Voltage and Current Regulator Support --->

```
< >   Active Semi ACT8891 PMIC regulators
< >   rk2918 pwm voltage regulator
<*>   rk30 pwm voltage regulator for discrete dcdc or ldo
```

（使用 pwm 调整电压的需要打开宏 rk30 pwm voltage regulator for discrete dcdc or ldo）

4.2 Arm 和 logic 的动态调节

Arm 和 logic 采用 pwm 调节占空比实现电压的调整，注册 name - "vdd_cpu"、"vdd_core"，dvfs 调整 arm 和 logic 会自动通过搜索"vdd_cpu"、"vdd_core" 的 name 实现调压操作,

在 board-rk30-sdk.c 中注册 pwm 接口：

```
#if CONFIG_RK30_PWM_REGULATOR
struct pwm_platform_data {
    int    pwm_id;
    int    pwm_gpio;
    //char      pwm_iomux_name[50];
    char*      pwm_iomux_name;
    unsigned int    pwm_iomux_pwm;
    int    pwm_iomux_gpio;
    int    pwm_voltage;
    struct regulator_init_data *init_data;
};
```

```
const static int pwm_voltage_map[] = {
    1000000, 1025000, 1050000, 1075000, 1100000, 1125000, 1150000, 1175000, 1200000, 1225000,
1250000, 1275000, 1300000, 1325000, 1350000, 1375000, 1400000
};
static struct regulator_consumer_supply pwm_dcdc1_consumers[] = {
    {
        .supply = "vdd_core",
    }
};
static struct regulator_consumer_supply pwm_dcdc2_consumers[] = {
    {
        .supply = "vdd_cpu",
    }
};
struct regulator_init_data pwm_regulator_init_dcdc[2] =
{
    {
        .constraints = {
            .name = "PWM_DCDC1",
            .min_uV = 600000,
            .max_uV = 1800000,      //0.6-1.8V
            .apply_uV = true,
            .valid_ops_mask          =          REGULATOR_CHANGE_STATUS          |
REGULATOR_CHANGE_VOLTAGE,
        },
        .num_consumer_supplies = ARRAY_SIZE(pwm_dcdc1_consumers),
```

```
        .consumer_supplies = pwm_dcdc1_consumers,
    },
    {
        .constraints = {
            .name = "PWM_DCDC2",
            .min_uV = 600000,
            .max_uV = 1800000,      //0.6-1.8V
            .apply_uV = true,
            .valid_ops_mask = REGULATOR_CHANGE_STATUS |
REGULATOR_CHANGE_VOLTAGE,
        },
        .num_consumer_supplies = ARRAY_SIZE(pwm_dcdc2_consumers),
        .consumer_supplies = pwm_dcdc2_consumers,
    },
};

static struct pwm_platform_data pwm_regulator_info[2] = {
    {
        .pwm_id = 1,
        .pwm_gpio = RK30_PIN0_PA4,
        .pwm_iomux_name = GPIO0A4_PWM1_NAME,
        .pwm_iomux_pwm = GPIO0A_PWM1,
        .pwm_iomux_gpio = GPIO0A_GPIO0A4,
        .pwm_voltage = 1100000,
        .suspend_voltage = 1050000,
        .min_uV = 1000000,
```

```
        .max_uV = 1400000,
        .coefficient = 455,   //45.5%
        .pwm_voltage_map = pwm_voltage_map,
        .init_data = &pwm_regulator_init_dcdc[0],
    },
    {
        .pwm_id = 2,
        .pwm_gpio = RK30_PIN0_PD6,
        .pwm_iomux_name = GPIO0D6_PWM2_NAME,
        .pwm_iomux_pwm = GPIO0D_PWM2,
        .pwm_iomux_gpio = GPIO0D_GPIO0D6,
        .pwm_voltage = 1100000,
        .suspend_voltage = 1050000,
        .min_uV = 1000000,
        .max_uV = 1400000,
        .coefficient = 455,   //45.5%
        .pwm_voltage_map = pwm_voltage_map,
        .init_data = &pwm_regulator_init_dcdc[1],
    },
};
struct platform_device pwm_regulator_device[2] = {
    {
        .name = "pwm-voltage-regulator",
        .id = 0,
        .dev        = {
            .platform_data = &pwm_regulator_info[0],
```

```
        }
    },
    {
        .name = "pwm-voltage-regulator",
        .id = 1,
        .dev        = {
            .platform_data = &pwm_regulator_info[1],
        }
    },
};
#endif
```

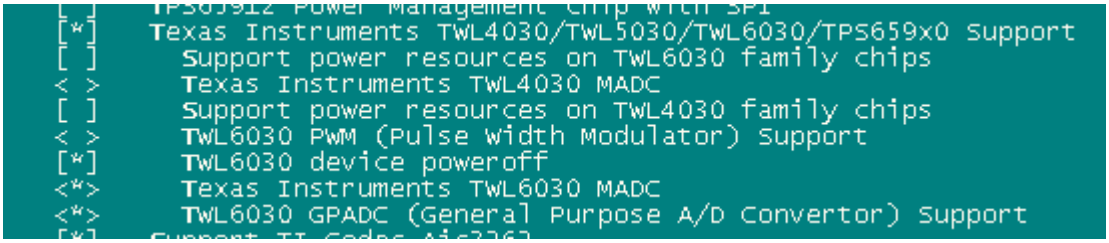
Pwm 设备的添加在 board-rk30-sdk-sdk.c 中：

```
static struct platform_device *devices[] __initdata = {
#ifdef CONFIG_RK30_PWM_REGULATOR
        &pwm_regulator_device[0],
        &pwm_regulator_device[1],
#endif
}
```

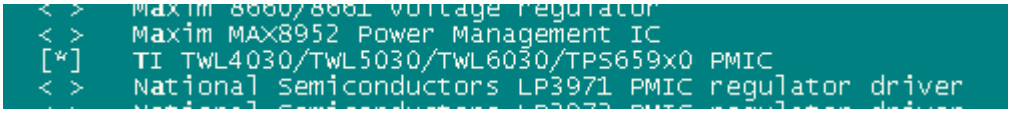
5 Tps80032 使用说明

5.1 配置说明

[*] Multifunction device drivers --->



[*] Voltage and Current Regulator Support --->



[*] Real Time Clock --->

5.2 修改各路初始电压

在 board-rk30-phone-twl60xx.c 中的 post_init()中：

```
dcdc = regulator_get(NULL, "vdd_cpu");     // vdd_arm
regulator_set_voltage(dcdc, 1100000, 1100000);     //设置运行下电压
regulator_enable(dcdc);
printk("%s set dcdc2 vdd_cpu(vdd_arm)=%dmV end\n", __func__, regulator_get_voltage(dcdc));
regulator_put(dcdc);
udelay(100);
ldo = regulator_get(NULL, "ldo1");      // vcc18_cif
regulator_set_voltage(ldo, 1800000, 1800000);
regulator_enable(ldo);
//      printk("%s set ldo1  vcc18_cif=%dmV end\n", __func__, regulator_get_voltage(ldo));
regulator_put(ldo);
udelay(100);
```

5.3 休眠唤醒

5.3.1 pmu 进入 sleep 模式

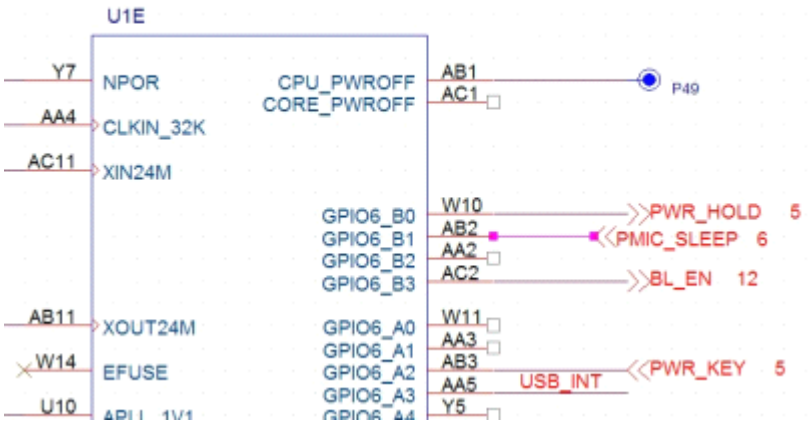
在 board-rk30-phone-twl60xx.c 中：

```
void __sramfunc board_pmu_suspend(void)
{
    #ifdef CONFIG_CLK_SWITCH_TO_32K                    //switch clk to 32k
    grf_writel(GPIO6_PB1_DIR_OUT, GRF_GPIO6L_DIR_ADDR);
    grf_writel(GPIO6_PB1_DO_HIGH, GRF_GPIO6L_DO_ADDR);   //set gpio6_b1 output high
    grf_writel(GPIO6_PB1_EN_MASK, GRF_GPIO6L_EN_ADDR);
    #endif
    grf_writel(GPIO6_PB3_DIR_OUT, GRF_GPIO6L_DIR_ADDR);
    grf_writel(GPIO6_PB3_DO_HIGH, GRF_GPIO6L_DO_ADDR);   //set gpio6_b3 output high
    grf_writel(GPIO6_PB3_EN_MASK, GRF_GPIO6L_EN_ADDR);
}
void __sramfunc board_pmu_resume(void)
{
    grf_writel(GPIO6_PB3_DIR_OUT, GRF_GPIO6L_DIR_ADDR);
    grf_writel(GPIO6_PB3_DO_LOW, GRF_GPIO6L_DO_ADDR);      //set gpio6_b3 output low
    grf_writel(GPIO6_PB3_EN_MASK, GRF_GPIO6L_EN_ADDR);
    #ifdef CONFIG_CLK_SWITCH_TO_32K                    //switch clk to 24M
    grf_writel(GPIO6_PB1_DIR_OUT, GRF_GPIO6L_DIR_ADDR);
    grf_writel(GPIO6_PB1_DO_LOW, GRF_GPIO6L_DO_ADDR);   //set gpio6_b1 output low
    grf_writel(GPIO6_PB1_EN_MASK, GRF_GPIO6L_EN_ADDR);
    sram_32k_udelay(10000);
    #else
```

```
    sram_udelay(2000);
    #endif
}
```

Gpio6_b1 用于 pmu sleep，gpio6_b3 用于切换 24m 晶振（打开宏 CONFIG_CLK_SWITCH_TO_32K）。

不同客户需要核对硬件原理图确认 gpio 口。



5.3.2 在休眠状态下关闭一些 ldo 或者降低各路休眠电压

（2）关闭一些 ldo：

在 board-rk30-sdk-twl60xx.c 的 pre_init（）中：

twl_reg_write(LDO5_CFG_TRANS,TWL_MODULE_PM_RECEIVER,0x03);　　//set ldo5 is disabled when in sleep mode

（3）降低各路休眠电压

由于 ti 的没有 sleep voltage 寄存器，所以无法通过 regulator_set_suspend_voltage（）接口实现休眠电压的设置目前使用 I2c_sram.c 的流程还没有实现。后期会按照 tps65910 的方案实现。

5.4 Arm 和 logic 的动态调节

Arm 和 logic 都采用 pmu 输出，在 pmu 中注册相应 name -"vdd_cpu"、"vdd_core"，dvfs 调整 arm 和 logic 会自动通过搜索"vdd_cpu"、"vdd_core" 的 name 实现调压操作。

5.5 Tps80032 一级休眠接口

一级休眠接口主要用于在系统在进入一级休眠后，通过设置 pmic 的 dcdc 和 ldo 在一级休眠下的电压，或者中断状态等。

在 board-rk30-phone-twl60xx.c 中：

```
#ifdef CONFIG_HAS_EARLYSUSPEND
void twl60xx_pmu_early_suspend(struct regulator_dev *rdev)
{
}
void twl60xx_pmu_early_resume(struct regulator_dev *rdev)
{
}
#else
void twl60xx_pmu_early_suspend(struct regulator_dev *rdev)
{
}
void twl60xx_pmu_early_resume(struct regulator_dev *rdev)
{
}
#endif
```

5.6关机流程

Pmu 的关机流程在 board-rk30-sdk.c 中实现，具体如下：

```
static void rk30_pm_power_off(void)
{
```

```
	printk(KERN_ERR "rk30_pm_power_off start...\n");
#if defined(CONFIG_TWL4030_CORE)
	twl6030_poweroff();
#endif
	while (1);
}
```