Android 10 PMIC

使用说明书

1.1

2020.05.08

文档履历

版本号	日期	制/修订人	内容描述
1.1	2020.05.08		

目录

1. 概述

1.1 编写目的

本文档主要介绍 PMU 使用方法。

1.2 适用范围

硬件平台：AXP2101、AXP803

软件版本：Linux-4.9

1.3 相关人员

本文档可供系统维护人员、驱动开发人员和测试人员参考。

2. 模块介绍

2.1 模块功能介绍

电源管理单元，负责系统各模块供电及电池充放电管理。

2.2 相关术语介绍

术语	解释说明
PMU	电源管理单元。
AXP	全志 PMU 的名称，如 AXP152、AXP22x 等。
LDO	是 low dropout regulator，意为低压差线性稳压器。线性稳压器使用在其线性区域内运行的晶体管或 FET，从应用的输入电压中减去超额的电压，产生经过调节的输出电压。
DC-DC	是直流变直流，即不同直流电源值之间的转换，只要符合这个定义都可以叫 DC-DC 转换器，也包括 LDO。但是一般的说法是把直流变直流由开关方式实现的器件叫 DCDC。
Regulator	linux 内核对 LDO、DC-DC 的控制核心。

2.3 模块配置介绍

2.3.1 menuconfig 配置说明

在 kernel 目录，运行 make menuconfig ARCH=arm 或 make menuconfig ARCH=arm64，进入配置界面。以 AXP2101 的设备举例，其他参考相应的 defconfig 文件。

2.3.1.1 PMU

Device Drivers ---> I2C support ---> <*> I2C support
I2C Hardware Bus support ---> <*> SUNXI I2C controller

Device Drivers ---> Multifunction device drivers ---> <*> X-Powers AXP2101 PMICs with I2C

2.3.1.2 regulator

Device Drivers ---> [*] Voltage and Current Regulator Support ---> <*> X-POWERS AXP2101 PMIC Regulators

2.3.1.3 charger

Device Drivers --->
[*] Power supply class support --->
<*> AXP2101 power supply driver

2.3.1.4 power key

Device Drivers ---> Input device support ---> -*- Generic input layer (needed for keyboard, mouse, ...)
[*] Miscellaneous devices --->
<*> X-Powers AXP2101 power button driver

2.3.1.5 virtual regulator

Device Drivers ---> [*] Voltage and Current Regulator Support ---> <*> Virtual regulator consumer support

2.3.2 dts 配置说明

PMU 包含 regulator，power supply，power key。PMU 的配置是通过设备树进行配置. PMU 在内核中的设备是多个设备同时存在，并存在层次对应关系。

```
  PMU主设备(MFD)
  |
  +------> regulator device
  |
  +------> power key device
  |
  +------> power supply device
  |
  +------> gpio device
|
+------> wdt device
```

PMU 设备树配置，以 AXP2101 的设备举例，其他参考相应的 dts 文件:

```
pmu0: pmu@0{
    compatible = "x-powers,axp2101";
    reg = <0x34>;
    #address-cells = <1>;
    #size-cells = <0>;
    interrupts = <0 IRQ_TYPE_LEVEL_LOW>;
    interrupt-parent = <&nmi_intc>;
    status = "okay";
    wakeup-source;

    charger0: charger@0{
        compatible = "x-powers,axp2101-power-supply";
        param = <&axp2101_parameter>;
        status = "okay";
        pmu_chg_ic_temp = <0>;
```

```
    pmu_battery_cap = <888>;
    pmu_runtime_chgcur = <1000>;
    pmu_suspend_chgcur = <1500>;
    pmu_shutdown_chgcur = <1500>;
    pmu_init_chgvol = <4200>;
    pmu_usbpc_cur = <2000>;
    pmu_battery_warning_level1 = <15>;
    pmu_battery_warning_level2 = <0x0>;
    pmu_chgled_type = <0x0>;

    wakeup_usb_in;
    wakeup_usb_out;
    wakeup_bat_out;
    /* wakeup_bat_in; */
    /* wakeup_bat_charging; */
    /* wakeup_bat_charge_over; */
    /* wakeup_low_warning1; */
    /* wakeup_low_warning2; */
    /* wakeup_bat_untemp_work; */
    /* wakeup_bat_ovtemp_work; */
    /* wakeup_bat_untemp_chg; */
    /* wakeup_bat_ovtemp_chg; */


};

powerkey0: powerkey@0{
    status = "okay";
    compatible = "x-powers,axp2101-pek";
    pmu_powkey_off_time = <6000>;
    pmu_powkey_off_func = <0>;
    pmu_powkey_off_en = <1>;
    pmu_powkey_long_time = <1500>;
    pmu_powkey_on_time = <512>;
    wakeup_rising;
```

```
    /* wakeup_falling; */

};

regulator0: regulators@0{
    reg_dcdc1: dcdc1 {
        regulator-name = "axp2101-dcdc1";
        regulator-min-microvolt = <1500000>;
        regulator-max-microvolt = <3400000>;
        regulator-boot-on;
        regulator-always-on;
    };
    reg_dcdc2: dcdc2 {
        regulator-name = "axp2101-dcdc2";
        regulator-min-microvolt = <500000>;
        regulator-max-microvolt = <1540000>;
        regulator-boot-on;
        regulator-always-on;
    };
    reg_dcdc3: dcdc3 {
        regulator-name = "axp2101-dcdc3";
        regulator-min-microvolt = <500000>;
        regulator-max-microvolt = <3400000>;
        regulator-boot-on;
        regulator-always-on;
    };
    reg_dcdc4: dcdc4 {
        regulator-name = "axp2101-dcdc4";
        regulator-min-microvolt = <500000>;
        regulator-max-microvolt = <1840000>;
        regulator-boot-on;
        regulator-always-on;
    };
    reg_dcdc5: dcdc5 {
        regulator-name = "axp2101-dcdc5";
```

```
        regulator-min-microvolt = <1200000>;
        regulator-max-microvolt = <3700000>;
    };
    reg_rtcldo: rtcldo {
        /* RTC_LDO is a fixed, always-on regulator */
        regulator-name = "axp2101-rtcldo";
        regulator-min-microvolt = <1800000>;
        regulator-max-microvolt = <1800000>;
        regulator-boot-on;
        regulator-always-on;
    };
    reg_rtcldo1: rtcldo1 {
        regulator-name = "axp2101-rtcldo1";
        regulator-min-microvolt = <1800000>;
        regulator-max-microvolt = <1800000>;
    };
    reg_aldo1: aldo1 {
        regulator-name = "axp2101-aldo1";
        regulator-min-microvolt = <500000>;
        regulator-max-microvolt = <3500000>;
        regulator-boot-on;
        regulator-always-on;
    };
    reg_aldo2: aldo2 {
        regulator-name = "axp2101-aldo2";
        regulator-min-microvolt = <500000>;
        regulator-max-microvolt = <3500000>;
        regulator-always-on;
    };
    reg_aldo3: aldo3 {
        regulator-name = "axp2101-aldo3";
        regulator-min-microvolt = <500000>;
        regulator-max-microvolt = <3500000>;
        regulator-boot-on;
    };
```

```
        reg_aldo4: aldo4 {
            regulator-name = "axp2101-aldo4";
            regulator-min-microvolt = <500000>;
            regulator-max-microvolt = <3500000>;
        };
        reg_bldo1: bldo1 {
            regulator-name = "axp2101-bldo1";
            regulator-min-microvolt = <500000>;
            regulator-max-microvolt = <3500000>;
            regulator-boot-on;
            regulator-always-on;
        };
        reg_bldo2: bldo2 {
            regulator-name = "axp2101-bldo2";
            regulator-min-microvolt = <500000>;
            regulator-max-microvolt = <3500000>;
        };
        reg_dldo1: dldo1 {
            regulator-name = "axp2101-dldo1";
            regulator-min-microvolt = <500000>;
            regulator-max-microvolt = <3500000>;
        };
        reg_dldo2: dldo2 {
            regulator-name = "axp2101-dldo2";
            regulator-min-microvolt = <500000>;
            regulator-max-microvolt = <1400000>;
        };
        reg_cpusldo: cpusldo {
            regulator-name = "axp2101-cpusldo";
            regulator-min-microvolt = <500000>;
            regulator-max-microvolt = <1400000>;
        };
    };

    virtual-dcdc1 {
```

```
        compatible = "xpower-vregulator,dcdc1";
        dcdc1-supply = <&reg_dcdc1>;
    };
    virtual-dcdc2 {
        compatible = "xpower-vregulator,dcdc2";
        dcdc2-supply = <&reg_dcdc2>;
    };
    virtual-dcdc3 {
        compatible = "xpower-vregulator,dcdc3";
        dcdc3-supply = <&reg_dcdc3>;
    };

    virtual-dcdc4 {
        compatible = "xpower-vregulator,dcdc4";
        dcdc4-supply = <&reg_dcdc4>;
    };
    virtual-dcdc5 {
        compatible = "xpower-vregulator,dcdc5";
        dcdc5-supply = <&reg_dcdc5>;
    };

    virtual-aldo1 {
        compatible = "xpower-vregulator,aldo1";
        aldo1-supply = <&reg_aldo1>;
    };
    virtual-aldo2 {
        compatible = "xpower-vregulator,aldo2";
        aldo2-supply = <&reg_aldo2>;
    };
    virtual-aldo3 {
        compatible = "xpower-vregulator,aldo3";
        aldo3-supply = <&reg_aldo3>;
    };
    virtual-aldo4 {
        compatible = "xpower-vregulator,aldo4";
```

```
    aldo4-supply = <&reg_aldo4>;
};

virtual-bldo1 {
    compatible = "xpower-vregulator,bldo1";
    bldo1-supply = <&reg_bldo1>;
};
virtual-bldo2 {
    compatible = "xpower-vregulator,bldo2";
    bldo2-supply = <&reg_bldo2>;
};

virtual-dldo1 {
    compatible = "xpower-vregulator,dldo1";
    dldo1-supply = <&reg_dldo1>;
};
virtual-dldo2 {
    compatible = "xpower-vregulator,dldo2";
    dldo2-supply = <&reg_dldo2>;
};

axp_gpio0: axp_gpio@0{
    gpio-controller;
    #size-cells = <0>;
    #gpio-cells = <6>;
    status = "okay";
                };
            };

            axp2101_parameter:axp2101-parameter {
                select = "battery-model";

                battery-model {
                    parameter = /bits/ 8 <0x01 0xF5 0x00 0x00 0xFB 0x00 0x00 0xFB
                        0x00 0x1E 0x32 0x01 0x14 0x04 0xD8 0x04
```

```
                0x74 0xFD 0x58 0x0B 0xB3 0x10 0x3F 0xFB
                0xC8 0x00 0xBE 0x03 0x4E 0x06 0x3F 0x06
                0x02 0x0A 0xD3 0x0F 0x74 0x0F 0x31 0x09
                0xE5 0x0E 0xB9 0x0E 0xC0 0x04 0xBE 0x04
                0xBB 0x09 0xB4 0x0E 0xA0 0x0E 0x92 0x09
                0x79 0x0E 0x4C 0x0E 0x27 0x03 0xFC 0x03
                0xD5 0x08 0xBC 0x0D 0x9C 0x0D 0x55 0x06
                0xB8 0x2E 0x24 0x2E 0x2E 0x24 0x2E 0x24
                0xC5 0x98 0x7E 0x66 0x4E 0x44 0x38 0x1A
                0x12 0x0A 0xF6 0x00 0x00 0xF6 0x00 0xF6
                0x00 0xFB 0x00 0x00 0xFB 0x00 0x00 0xFB
                0x00 0x00 0xF6 0x00 0x00 0xF6 0x00 0xF6
                0x00 0xFB 0x00 0x00 0xFB 0x00 0x00 0xFB
                0x00 0x00 0xF6 0x00 0x00 0xF6 0x00 0xF6>;
        };
    };
```

2.3.2.1 PMU 属性配置

reg <u32>
    i2c寄存器地址

interrupts <args>
    中断配置，参考内核中断配置文档

interrupt-parent <phandler>
    上级中断控制器结点

wakeup-source <bool>
    是否支持唤醒

pmu_powerok_noreset <bool>
    powerok io不支持低电平reset功能

x-powers,drive-vbus-en <bool>

set N_VBUSEN pin as an output pin to control an external regulator to drive VBus

pmu_reset

when power key press longer than 16s, PMU reset or not.

0: not reset

1: reset

pmu_irq_wakeup

press irq wakeup or not when sleep or power down.

0: not wakeup

1: wakeup

pmu_hot_shutdown

when PMU over temperature protect or not.

0: disable

1: enable

2.3.2.2 regulator 属性配置

regulator 为系统 regulator_dev 设备，每个 regulator_dev 代表一路电源，设备通过对 regulator_dev 的引用建立 regulator，用来实现对电源的电压设置等功能。regulator 设备目前暂无设备树属性可以配置。

2.3.2.3 power supply 属性配置

power supply 属性配置，包括 ac-power-supply、usb-power-supply 和 battery-power-supply。

对于 ac-power-supply 属性配置如下。

pmu_ac_vol <u32>

ac输入电压限制值

单位为mV

pmu_ac_cur <u32>
ac输入电流限制值
单位为mA

wakeup_ac_in <bool>
ac插入唤醒使能

wakeup_ac_out <bool>
ac拔出唤醒使能

对于 usb-power-supply 属性配置如下。

pmu_usbpc_vol <u32>
usb pc输入电压限制值
单位为mV

pmu_usbpc_cur <u32>
usb pc输入电流限制值
单位为mA

pmu_usbad_vol <u32>
usb adaptor输入电压限制值
单位为mV

pmu_usbad_cur <u32>
usb adaptor输入电流限制值
单位为mA

wakeup_usb_in <bool>
usb插入唤醒使能

wakeup_usb_out <bool>
usb拔出唤醒使能

对于 battery-power-supply 属性配置如下。pmu_init_chgvol 充电电压限制单位为 mV

pmu_chgled_func <u32>
CHGKED pin control
0: controlled by pmu
1: controlled by Charger

pmu_chgled_type <u32>
CHGLED Type select when pmu_chgled_func is 0
0: display with type A function
1: display with type B function
3: output controlled by the register of chgled_out_ctrl

pmu_chled_enable <u32>
设置CHGLED pin 是否使能
0: disalbe
1: enable

pmu_chg_ic_temp <u32>
1: TS current source always on
0: TS current source off

pmu_battery_warning_level1 <u32>
5-20 5% - 20% warning level1

pmu_battery_warning_level2 <u32>
0-15 0% - 15% warning level2

pmu_pre_chg <u32>
设置预充电电流限制
0 - 200 step is 25单位为mA

pmu_iterm_limit <u32>
设置截至充电电流
0 - 200 step is 25 mA

pmu_runtime_chgcur <u32>

运行时constant充电电流限制
单位为mA

pmu_suspend_chgcur <u32>
休眠时constant充电电流限制
单位为mA

pmu_shutdown_chgcur <u32>
关机时constant充电电流限制
单位为mA

pmu_bat_det <u32>
bat_det此为一个byte写入bat_det寄存器，由三个有效数据或组成
battery detection means select
byte[2]: 0 charge/discharge
1 NTC
battery detection charge/discharge current time
byte[1]: 0 1s
1 128ms
battery detection enable
byte[0]: 0 disable
1 enable

pmu_btn_chg_en <bool>
button battery charge enable

pmu_btn_chg_cfg
2600 - 3300 纽扣电池充电截至电压，单位为mV

param <phandler>
指定电池充电结点
电池参数由多个字节组成,在param指定的phandler结点里面添加电池结点，然后通过select选择参数结点名字用来指定哪个结点。
参考实例：根据前面的pmic参考设备树节点的pmic-parameter结点。
电池参数根据使用的电池不同，通过仪器测量出来。

pmu_bat_para1 <u32>
pmu_bat_para2 <u32>
...
pmu_bat_para32 <u32>
电池曲线参数
电池参数根据使用的电池不同，通过仪器测量出来。

pmu_battery_rdc <u32>
电池内阻
单位为mΩ

pmu_battery_cap <u32>
电池容量
单位为mAh

pmu_bat_temp_para1 <u32>
pmu_bat_temp_para2 <u32>
...
pmu_bat_temp_para16 <u32>
电池温度曲线参数
电池温度参数根据使用的电池不同

pmu_bat_temp_enable <u32>
设置电池温度检测是否使能
0: disalbe
1: enable

pmu_bat_charge_ltf <u32>
电池低温关闭充电阈值

pmu_bat_charge_htf <u32>
电池高温关闭充电阈值

pmu_bat_shutdown_ltf <u32>

电池低温关机阈值

pmu_bat_shutdown_htf <u32>
电池高温关机阈值

wakeup_bat_in <bool>
电池插入唤醒使能

wakeup_bat_out <bool>
电池拔出唤醒使能

wakeup_bat_charging <bool>
电池充电唤醒使能

wakeup_bat_charge_over <bool>
电池充电结束唤醒使能

wakeup_low_warning1 <bool>
电池低电量告警唤醒使能

wakeup_low_warning2 <bool>
电池低电量告警2唤醒使能

wakeup_bat_untemp_chg <bool>
电池低温充电唤醒使能

wakeup_bat_ovtemp_chg <bool>
电池超温充电唤醒使能

wakeup_bat_untemp_work <bool>
电池低温工作唤醒使能

wakeup_bat_ovtemp_work <bool>
电池高温工作唤醒使能

2.3.2.4 power key 属性配置

power key 设备为按键设备，具体的说为电源按键设备。power key 属性配置。

pmu_powkey_off_time <u32>
控制按下多长时间响应poweroff事件
可选的值为：
4000 4s
6000 6s
8000 8s
10000 10s

pmu_powkey_off_func <u32>
控制power_off事件功能,如果不配置，默认为power-off
1 复位系统
0 power_off

pmu_powkey_long_time <u32>
控制ponlevel 寄存器0x27[5:4]
1000 1s
1500 1.5s
2000 2s
2500 2.5s

pmu_powkey_off_en
控制按键关机使能
1 PWRON > OFFLEVEL AS poweroff source enable
0 PWRON > OFFLEVEL as poweroff source disable

pmu_powkey_on_time <u32>
控制按钮按下多长时间开机
128 0.128s
512 0.512s
1000 1s
2000 2s

pmu_pwrok_time <u32>
delay of PWROK after all power output good
8 8ms
16 16ms
32 32ms
64 64ms

wakeup_rising <bool>
控制是否弹起按钮唤醒系统

wakeup_falling <bool>
控制是否按下按钮唤醒系统

2.3.2.5 watchdog 属性配置

暂未添加设备树支持。

2.4 源码结构介绍

```
linux4.9
|-- drivers/mfd/axp2101.c
|-- drivers/mfd/axp2101-i2c.c
|-- drivers/regulator/axp2101-regulator.c
|-- drivers/input/misc/axp2101-pek.c
|-- drivers/power/supply/axp2201_charger.c
|-- drivers/power/supply/axp803_ac_power.c
|-- drivers/power/supply/axp803_usb_power.c
|-- drivers/power/supply/axp803_battery.c
```

mfd 目录下为 PMIC 的 mfd 驱动代码；regulator 目录下为 PMIC 的 regulator 驱动代码；input/misc 目录下为 PMIC 的 power key 驱动代码；drivers/power/supply 目录下为 PMIC 的 charger 驱动代码。

3. 接口设计

3.1 regulator 接口

无。

3.2 gpio 接口

无。

4. demo

4.1 regulator 使用 demo

其他设备对regulator_dev设备的引用通过设备树配置。

<name>-supply = <&reg_dcdc1>;

设备中通过对name的获取，可以获取reg_dcdc1的regulator_dev设备，然后对此路电源进行电源开关，电压设置等功能。

具体设备引用参考内核的regulator使用文档。

4.2 gpio 使用 demo

4.3 charger 使用 demo

4.4 watchdog 使用 demo

PMU 会创建一个 hw_timeout 为 4s 的一个看门狗，默认 timeout 为 5s。使用方法。

创建的 watchdog 会在/dev/watchdog*, 如果使能 sunxi-dev 的 soc 内部 watchdog，pmic 的 watchdog 会抢先使用/dev/watchdog -> /dev/watchdog0, 这样保证直接使用/dev/watchdog 为使用 pmic 的 watchdog。在某些情况下面，soc 内部的 watchdog 会存在不能复位 pmic 的情况，这时需要使用 pmic 的 watchdog。1. pmic 的 watchdog 打开后即开启，关闭文件不能关闭看门狗。2. 如需要关闭看门狗，需要发送'V' 字符即可关闭看门狗。3. 看门狗使用标准的 set_timeout 方法设置看门狗时间。4. 通过写 watchdog 可以喂狗，或者使用标准的 ioctl 方法。

5. 调试方法

在设备进行开发过程中，难免需要对各路电源进行调试，控制电源各路电压等操作，内核中提供了对电源调试的方式.

5.1 shell 命令设置 regulator 电压

对各路电压的控制是常用的调试手段，通过对各路不同电压设置，实现功耗，性能，稳定性等信息.

内核通过对每一路电源创建一个virtual设备，通过导出virtual设备的电源控制结点，用来对每一路电源的电压进行控制。
通过进入不同的virtual设备，来控制不同的电源。

virtual设备存在于axp2101主设备结点下面，因此设备路径为主设备下面的从设备。以AXP2101的设备举例。

/sys/devices/platform/soc/twi4/i2c-4/4-0034/regulator/regulator.1/reg-virt-consomer.1-dcdc1/

通过此路径下面的**max_microvolts**和**min_microvolts**设备结点进行写操作，用来完成对设备电源的控制，
此例为：

echo 3000000 > max_microvolts
echo 3000000 > min_microvolts

设置电压为3000000uv,3000mv,3v

5.2 shell 命令获取 regulator 引用设备

获取有哪几路电源引用了电源，进入需要查看的电源，进入 /sys/class/regulator/regulator.1 , 查看当前目录下的目录，即可确定有哪几路引用设备。另外一种方法就是进入 regulator 的 debugfs 结点，用来查看 regulator 的 map 信息。参考读取各路电源状态

5.3 shell 命令查询 regulator 状态

kernel 提供调试结点供电源进行调试进行，我们可以通过 kernel 的调试结点获取各路电源的各个详细状态。以 AXP2101 的设备举例，首先需要 mount debugfs 文件系统

```
mount -t debugfs none /sys/kernel/debug
cat /sys/kernel/debug/regulator/regulator_summary
```

```
regulator                  use open bypass voltage current     min     max
---------------------------------------------------------------------------
regulator-dummy             0    1    0       0mV     0mA     0mV     0mV
   uart0                                        0mV     0mV
axp2101-dcdc1               0    7    0    3300mV     0mA  1500mV  3400mV
   spi0                                         0mV     0mV
   sdc0                                         0mV     0mV
   sdc0                                         0mV     0mV
   sdc0                                         0mV     0mV
   sdc0                                         0mV     0mV
   sdc0                                         0mV     0mV
   reg-virt-consumer.1                          0mV     0mV
axp2101-dcdc2               0    1    0     900mV     0mA   500mV  1540mV
   reg-virt-consumer.2                          0mV     0mV
axp2101-dcdc3               0    2    0    1000mV     0mA   500mV  3400mV
   cpu0                                      1000mV  1000mV
   reg-virt-consumer.3                          0mV     0mV
axp2101-dcdc4               0    1    0    1500mV     0mA   500mV  1840mV
   reg-virt-consumer.4                          0mV     0mV
axp2101-dcdc5               0    1    0    1400mV     0mA  1400mV  3700mV
   reg-virt-consumer.5                          0mV     0mV
axp2101-rtcldo              0    0    0    1800mV     0mA  1800mV  1800mV
axp2101-rtcldo1             0    0    0    1800mV     0mA  1800mV  1800mV
axp2101-aldo1               0    1    0    1800mV     0mA   500mV  3500mV
   reg-virt-consumer.8                          0mV     0mV
axp2101-aldo2               0    2    0    3300mV     0mA   500mV  3500mV
   spi2                                         0mV     0mV
```

```
  reg-virt-consumer.9                                0mV    0mV
axp2101-aldo3          0   1    0  3300mV    0mA   500mV  3500mV
  reg-virt-consumer.10                               0mV    0mV
axp2101-aldo4          0   1    0  3300mV    0mA   500mV  3500mV
  reg-virt-consumer.11                               0mV    0mV
axp2101-bldo1          0   1    0  1800mV    0mA   500mV  3500mV
  reg-virt-consumer.12                               0mV    0mV
axp2101-bldo2          0   1    0  3300mV    0mA   500mV  3500mV
  reg-virt-consumer.13                               0mV    0mV
axp2101-dldo1          1   1    0  1200mV    0mA   500mV  3500mV
  reg-virt-consumer.14                               0mV    0mV
axp2101-dldo2          0   1    0  1200mV    0mA   500mV  1400mV
  reg-virt-consumer.15                               0mV    0mV
axp2101-cpusldo        0   0    0   900mV    0mA   500mV  1400mV
```

通过上例可以看出各路电源有哪几路设备请求，已经请求的值和目前各路电源的状态

5.4 shell 命令读写寄存器

寄存器调试是指直接对 pmic 的寄存器进行读写操作，此操作应该对寄存器有了解的情况下进行操作，不正确的操作方式将会导致芯片烧毁。在终端中，对抛出的调试结点进行读写操作，即可对寄存器进行读写操作。无论是读还是写寄存器，都应该首先挂载 debugfs 文件系统。

由于 pmic 是通过 regmap 进行读写操作，应该可以使用 regmap 的调试结点进行对 pmic 的读写访问操作。regmap 的调试结点在 debugfs 文件系统下面，通过对 regmap 调试结点的操作可以对 pmic 的寄存器进行读写访问操作。

5.4.1 写操作

寄存器调试挂载在debugfs文件系统。

```
mount -t debugfs none /sys/kernel/debug
echo ${reg} ${value} > /sys/kernel/debug/regmap/${dev-name}/registers
```

实例：

echo 0xff 0x01 > /sys/kernel/debug/regmap/4-0034/registers
写0xff寄存器值为0x01

5.4.2 读操作

寄存器调试挂载在debugfs文件系统。
mount -t debugfs none /sys/kernel/debug
cat /sys/kernel/debug/regmap/${dev-name}/registers

实例：

cat /sys/kernel/debug/regmap/4-0034/registers
读取pmic所有寄存器

另外，还支持 axp 驱动自定义节点 axp_reg 读写寄存器。但是这种用法是不推荐的，因为有标准 regmap 方式来读写寄存器，根本没必要用私有非标的方式。示例如下。

5.4.3 axp_reg 节点读写操作

往axp寄存器0x0f写入值0x55：
echo 0x0f55 > /sys/class/axp/axp_reg
读出axp寄存器0x0f的值：
echo 0x0f > /sys/class/axp/axp_reg
cat /sys/class/axp/axp_reg

5.5 power supply 调试方法

根据电池的各种状态，读取/sys/class/power_supply/{battery,usb,ac} 下面状态，判断一致性。

5.6 power key 调试方法

在用户空间调用 evtest, 通过 evtest 选择 pmic 的 evdev 测试。按下按钮为 1，弹起为 0。

```
available devices:
/dev/input/event0:      axp2101-pek
/dev/input/event1:      sunxi-gpadc0
Select the device event number [0-1]: 0
Input driver version is 1.0.1
Input device ID: bus 0x0 vendor 0x0 product 0x0 version 0x0
Input device name: "axp2101-pek"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 116 (KEY_POWER)
Key repeat handling:
  Repeat type 20 (EV_REP)
  Repeat code 0 (REP_DELAY)
    Value    250
  Repeat code 1 (REP_PERIOD)
    Value     33
Properties:
Testing ... (interrupt to exit)
Event: time 84985.644515, type 1 (EV_KEY), code 116 (KEY_POWER), value 1
Event: time 84985.644515, ---------- SYN_REPORT ------------
Event: time 84985.900628, type 1 (EV_KEY), code 116 (KEY_POWER), value 2
Event: time 84985.900628, ---------- SYN_REPORT ------------
Event: time 84985.934310, type 1 (EV_KEY), code 116 (KEY_POWER), value 0
Event: time 84985.934310, ---------- SYN_REPORT ------------
Event: time 84986.267772, type 1 (EV_KEY), code 116 (KEY_POWER), value 1
Event: time 84986.267772, ---------- SYN_REPORT ------------
Event: time 84986.446532, type 1 (EV_KEY), code 116 (KEY_POWER), value 0
Event: time 84986.446532, ---------- SYN_REPORT ------------
```

6. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This document neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.