Linux PMIC
开发指南

版本号: 2.0
发布日期: 2020.11.10

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.07.22	AWA0863	1. 添加初始版本
2.0	2020.11.10	AWA1442	1. 支持 linux-5.4 版本 2. 新增部分节点功能描述

目　录

1 前言

1.1 文档简介

介绍 PMIC 模块配置和调试方法。

1.2 目标读者

PMIC 模块开发、维护人员。

1.3 适用范围

表 1-1：适用产品列表

产品名称	内核版本	驱动文件
T509	Linux-4.9	drivers/regulator/* drivers/power/supply/*
MR813	Linux-4.9	drivers/regulator/* drivers/power/supply/*
R818	Linux-4.9	drivers/regulator/* drivers/power/supply/*
A133	Linux-4.9/5.4	drivers/regulator/* drivers/power/supply/*
H616	Linux-4.9/5.4	drivers/regulator/* drivers/power/supply/*

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

表 2-1：术语介绍

术语	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 Device Tree 配置说明

设备树中存在的是该类芯片所有平台的模块配置，设备树文件的路径为：kernel/linux-4.9/arch/arm64（32 位平台为 arm）/boot/dts/sunxi/CHIP.dtsi(CHIP 为研发代号，如 sun50iw10p1 等)。PMIC 模块在 dtsi 中无用户可用配置。

2.3.2 board.dts 配置说明

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 默认配置信息。

PMIC 包含 regulator，power supply，power key。PMU 的配置是通过设备树进行配置。PMIC 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
```

PMIC 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>;
            };
    };
};
```

- PMIC 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>
        是否作为唤醒源

pmu_powerok_noreset <bool>
        powerok pin 下拉复位功能使能。
        0: disabe
        1: enable
        **视具体的board.dts属性节点而定，部分型号PMU驱动暂未支持该功能**

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
        **视具体的board.dts属性节点而定，部分型号PMU驱动暂未支持该功能**
```

- 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 属性配置，参考内核原生 regulator 使用文档：Documentation/devicetree/bindings/regulator/regulator.txt。

```
reg_aldo1: aldo1{
    regulator-name = "axp2101-dcdc1";
            为电源设备的名称

    regulator-min-microvolt = <1500000>;
            电源的最小值，单位：uV

    regulator-max-microvolt = <3400000>;
            电源的最大值，单位：uV

    regulator-ramp-delay = <2500>;
            电源的调压延时，单位：us

    regulator-enable-ramp-delay = <1000>;
            电源从关闭到开启的使能延时，单位：us

    regulator-boot-on;
            电源从启动时开启

    regulator-always-on;
            电源保持常开
};
```

- 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 <u32>
        电池满充电压
        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
        **视具体的board.dts属性节点而定，部分型号PMU驱动暂未支持该功能**

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
        **视具体的board.dts属性节点而定，部分型号PMU驱动暂未支持该功能**

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
        **视具体的board.dts属性节点而定，部分型号PMU驱动暂未支持该功能**

pmu_btn_chg_en <bool>
        button battery charge enable
        **视具体的board.dts属性节点而定，部分型号PMU驱动暂未支持该功能**

pmu_btn_chg_cfg
        2600 - 3300 纽扣电池充电截至电压，单位为mV
        **视具体的board.dts属性节点而定，部分型号PMU驱动暂未支持该功能**

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。**
        **视具体的board.dts属性节点而定，部分型号PMU驱动暂未支持该功能**

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_ts_current <u32>
        TS pin的电流大小配置
        20uA/40uA/60uA/80uA 四档可配
        默认80uA，如若修改，下面的pmu_bat_temp_para计算时也要换成对应的TS电流

pmu_bat_temp_para1 <u32>
        电池包-25度对应的TS pin电压，单位：mV

pmu_bat_temp_para2 <u32>
        电池包-15度对应的TS pin电压，单位：mV

pmu_bat_temp_para3 <u32>
        电池包-10度对应的TS pin电压，单位：mV

pmu_bat_temp_para4 <u32>
        电池包-5度对应的TS pin电压，单位：mV

pmu_bat_temp_para5 <u32>
        电池包0度对应的TS pin电压，单位：mV

pmu_bat_temp_para6 <u32>
        电池包5度对应的TS pin电压，单位：mV

pmu_bat_temp_para7 <u32>
        电池包10度对应的TS pin电压，单位：mV

pmu_bat_temp_para8 <u32>
        电池包20度对应的TS pin电压，单位：mV

pmu_bat_temp_para9 <u32>
        电池包30度对应的TS pin电压，单位：mV

pmu_bat_temp_para10 <u32>
        电池包40度对应的TS pin电压，单位：mV

pmu_bat_temp_para11 <u32>
        电池包45度对应的TS pin电压，单位：mV

pmu_bat_temp_para12 <u32>
        电池包50度对应的TS pin电压，单位：mV

pmu_bat_temp_para13 <u32>
        电池包55度对应的TS pin电压，单位：mV

pmu_bat_temp_para14 <u32>
        电池包60度对应的TS pin电压，单位：mV
```

```
pmu_bat_temp_para15 <u32>
        电池包70度对应的TS pin电压，单位：mV

pmu_bat_temp_para16 <u32>
        电池包80度对应的TS pin电压，单位：mV
        **不同电池包的温敏电阻特性不一样，根据电池包的TS温敏电阻手册，找到pmu_bat_temp_para[1-16]对应温
    度点的电阻阻值，将阻值乘以80uA得到的电压数值（单位：mV），将电压数值填进pmu_bat_temp_para[1-16]的节
    点中即可**

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>
        触发电池高温停充的TS pin电压阈值，单位：mV
        默认：2112mV
        范围：0-3264mV

pmu_bat_charge_htf <u32>
        触发电池低温停充的TS pin电压阈值，单位：mV
        默认：397mV
        范围：0-3264mV

pmu_bat_shutdown_ltf <u32>
        非充电模式下，触发电池过温中断的TS pin电压阈值，单位：mV
        默认：3226mV
        范围：0-3264mV

pmu_bat_shutdown_htf <u32>
        非充电模式下，触发电池欠温中断的TS pin电压阈值，单位：mV
        默认：284mV
        范围：0-3264mV

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
```

- 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
```

- watchdog 属性配置

暂未添加设备树支持。

2.3.3 sysconfig 配置说明

在 sysconfig 中定义了 PMU 的 regulator 输出信息及板型 PMU 类型，在 boot0 和 uboot 会通过解析这部分属性来执行调压等操作。

```
;----------------------------------------------------------------------------------
;[target]  system bootup configuration
;boot_clock     = CPU boot frequency, Unit: MHz
;storage_type   = boot medium, 0-nand, 1-card0, 2-card2, -1(defualt)auto scan
;burn_key        1:support burn key; 0:not support burn key
;dragonboard_test 1:support card boot dragonboard; 0:not support card boot dragonboard
;power_mode     = axp_type,   0:axp81X, 1:dummy, 2:axp806, 3:axp2202, 4:axp858
;----------------------------------------------------------------------------------
[target]
boot_clock       = 1008
storage_type     = -1
burn_key         = 1
dragonboard_test = 0
power_mode       = 4

;----------------------------------------------------------------------------------
;   system configuration
;   ?
;dcdc1_vol                                set dcdc1 voltage,mV,500-1200,10mV/step
;                                                          1220-3400,20mV/step
;dcdc2_vol                                set dcdc2 voltage,mV,500-1200,10mV/step
;                                                          1220-1540,20mV/step
;aldo1_vol                                set aldo1 voltage,mV,500-3500,100mV/step
;dldo1_vol                                set dldo1 voltage,mV,500-3500,100mV/step
; dcdcX_mode                               set dcdc mode,  0:pfm-pwm   1:force pwm
; battery_exist                            select non-battery mode, 0: 无电池  1: 有电池
;----------------------------------------------------------------------------------
[power_sply]
dcdc1_vol                = 1003300
dcdc2_vol                = 1000940
dcdc6_vol                = 1003300
aldo1_vol                = 1003300
aldo2_vol                = 1001800
aldo3_vol                = 1003300
aldo4_vol                = 1003300
aldo5_vol                = 1001800
bldo1_vol                = 1001800
bldo2_vol                = 1001000
bldo4_vol                = 1001800
bldo5_vol                = 1001800
cldo1_vol                = 1001800
cldo2_vol                = 1002500
cldo3_vol                = 1001200
cldo4_vol                = 1001200
dc1sw_vol                = 1003300
dcdc5_mode               = 1
battery_exist            = 0
charge_mode              = 1

;----------------------------------------------------------------------------------
;   deylay after change voltage     Unit:us     default:0ms
;
;dcdc1_vol_delay                          ---set delay time after setting dcdc1_vol
```

```
;dcdc2_vol_delay                          ---set delay time after setting dcdc2_vol
;aldo1_vol_delay                          ---set delay time after setting aldo1_vol
;xxxxx_vol_delay                          ---set delay time after setting xxxxx_vol
;-----------------------------------------------------------------------------
[power_delay]
aldo3_vol_delay            = 50000

;-----------------------------------------------------------------------------
;   gpio bias and gpio power supply setting
;
;pl_bias    =1800           ---set gpio-l bias to 1800mV
;pl_supply  ="aldo3_vol"        ---change gpio-l bias to 1800mV after setting aldo3_vol
;
;xx_bias                ---set gpio bias to XXXmV
;xx_supply              ---change gpio bias after setting xxxx_vol
;-----------------------------------------------------------------------------
[gpio_bias]
pl_bias     = 3300
pl_supply   = "aldo3_vol"
```

- target 属性配置

```
power mode <u32>
        属性决定了当前SOC板型使用哪个PMU，需boot0代码支持解析，目前仅A133/T509方案支持解析该节点。后续别
    的平台需支持时应根据boot0代码更新sysconfig中的版型选择说明。
        0:axp81X,
        1: dummy
        2: axp806
        3: axp2202
        4: axp858
```

power mode 这个节点就是为了告诉平台当前使用的是哪个 PMU 的哪种方案（有时在同一 SOC 平台同一 PMU 也可能出现电源树配置不一样的情况），在 boot0 阶段就会解析出该属性并调用调压接口进行调压。以 T509 为例，boot0 阶段的调压过程如下：

对应的代码路径在：brandy/brandy-2.0/spl/board/sun50iw10p1/board.c

```
1  int sunxi_board_init(void)
2  {
3      ...
4      sunxi_board_pll_init();
5      axp_init(0);
6      set_pll_voltage(CONFIG_SUNXI_CORE_VOL);
7
8      switch (get_power_mode()) {
9      /* 0: A100-AXP81X */
10     case 0:
11         set_sys_voltage_ext("dcdc4", CONFIG_SUNXI_SYS_VOL);
12         break;
13     /* 1: Dummy */
14     /* 2: MR813-AXP305, R818-AXP305 */
15     case 2:
16         set_sys_voltage_ext("dcdcc", CONFIG_SUNXI_SYS_VOL);
17         break;
18     /* 3: A100-AXP2202*/
19     case 3:
```

```
        set_sys_voltage_ext("dcdc2", CONFIG_SUNXI_SYS_VOL);
        break;
    /* 4: T509-AXP858*/
    case 4:
        set_sys_voltage_ext("dcdc4", CONFIG_SUNXI_SYS_VOL);
        break;
    default:
        break;
    }
    ...
}
```

- power_sply 属性配置

```
xxxx_vol <u32>
        uboot阶段xxxx这路电是否开关及输出电压配置，其中xxxx为供电输出名。属性由前缀(100/000)和后缀组成。
    未配置的电在uboot阶段不会进行开关电和调压操作。
        前缀:  100，这路电在uboot阶段打开
        前缀:  000，这路电在uboot阶段关闭
        后缀:  3300，这路电输出电压设置为3300 mV

dcdcx_mode <u32>
        uboot阶段强制将dcdcx设置为fpwm开关模式，提高这路抗负载扰动能力。该属性不配置默认为0。目前仅
    AXP806/AXP305/AXP81X/AXP803支持该功能。
        0:  pfm-pwm模式自由切换
        1:  强制pwm模式

battery_exist <u32>
        强制电池存在状态，uboot阶段根据该属性决定是否做电池状态的相关判断。如果该属性不进行配置，默认为1。
    适用于部分无电持方案或factory_mode的无电池场景的调试
        0:  强制认为无电池存在，uboot阶段不做电池相关状态判断
        1:  认为电池存在，uboot阶段正常进行电池状态的相关判断

charge_mode <u32>
        配置充电页面，根据该属性决定是否进入关机充电页面。
        如果该属性不进行配置，默认为1。
        适用于不需要充电页面，或者适配器唤醒直接开机的需求。
        0:  不进入充电页面，适配器唤醒直接进入开机流程
        1:  适配器唤醒进入充电流程
```

- power_delay 属性配置

```
xxxx_vol_delay <u32>
        uboot阶段xxxx这路电调压后的延时时间，单位us。用于部分调压后需等电压稳定才能操作的模块，如：twi等。
        该属性需与**power_sply**中的xxx_vol属性一块使用，当输出需要开关或调压时才需要进行延时。
```

- gpio_bias 属性配置

```
xx_bias <u32>
        GPIOx口的耐压值设置，单位：mV。用于调整GPIOx口的耐压值，使其与GPIOx模块挂载的电压匹配，避免IO口损
    坏，提升信号质量。
```

```
xx_supply <char>
        GPIOx模块挂载的输出电压名，名字格式需与**power_sply**中的xxx_vol一致。该属性如果配上，在GPIOx
    挂载的输出改编后，会将对应的GPIOx bias耐压值修改过来。
```

2.3.4 kernel menuconfig 配置说明

进入内核根目录，执行 make ARCH=arm menuconfig（64 位平台为 make ARCH=arm64 menuconfig），以 AXP2101 的设备举例，其他参考相应的 defconfig 文件。进入配置主界面，并按以下步骤操作：

- PMIC

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

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

- regulator

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

- charger

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

- power key

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

- virtual regulator

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

2.4 源码结构介绍

以 AXP2101、AXP803 等 PMIC 为例。

```
kernel/
|-- 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
...
```

2.5 驱动框架介绍

AXP PMIC 是高度集成的电源系统管理芯片，针对单芯锂电池且需要多路电源转换输出的应用，提供简单易用而又可以灵活配置的完整电源解决方案，充分满足目前日益复杂的应用处理器系统对于电源精确控制的要求。

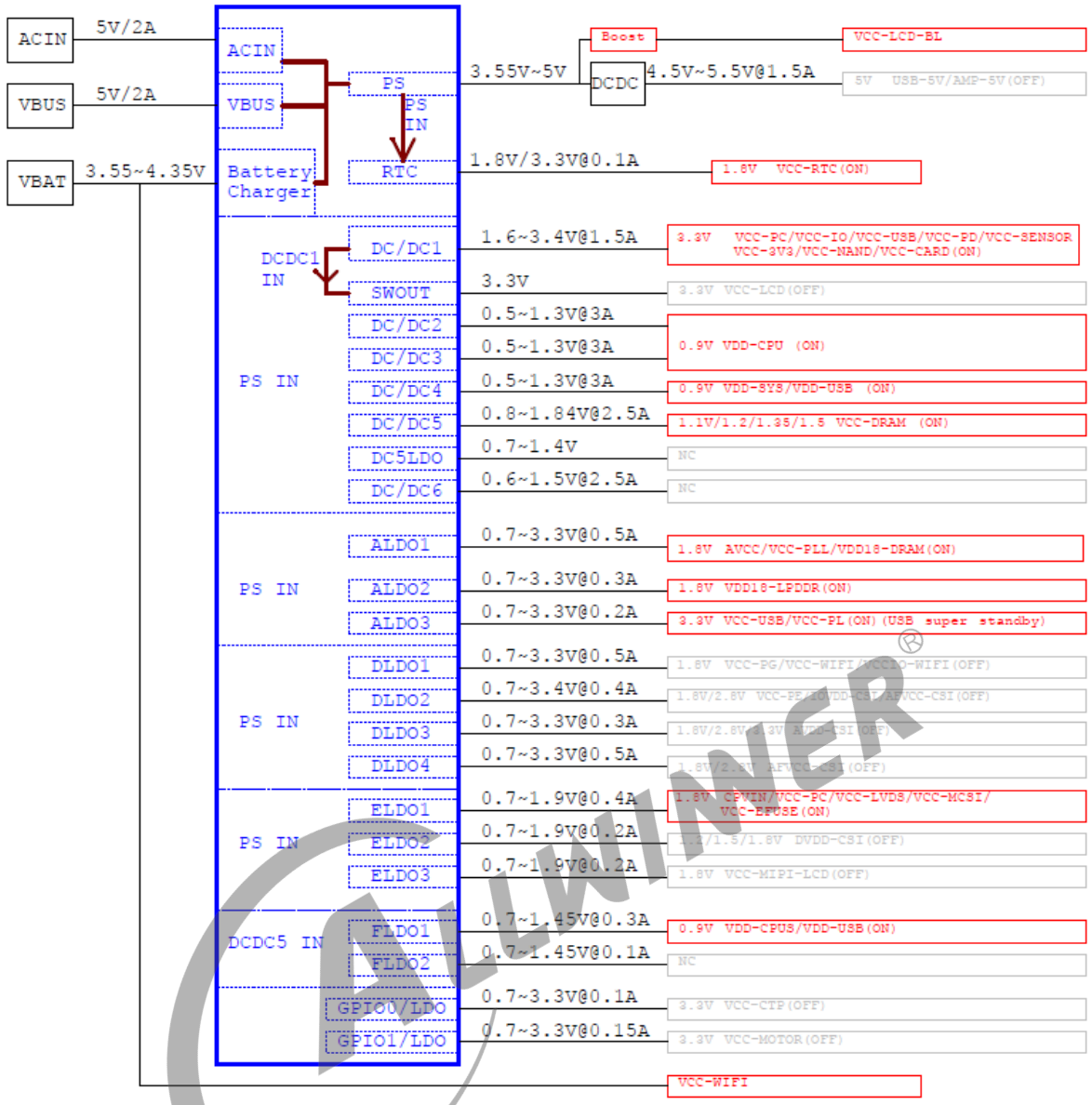

图 2-1: PMIC 原理框图

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。

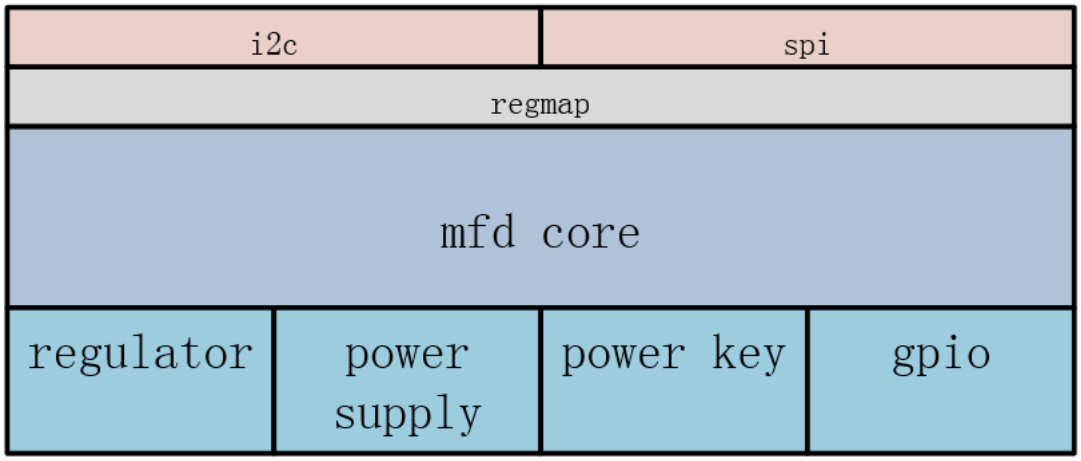

图 2-2: PMIC 驱动总体结构

3 模块使用范例

3.1 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。

3.2 gpio 使用 demo

其他设备对gpio_chip设备的引用通过设备树配置。
    <gpio name> = <&axp_gpio 2 GPIO_ACTIVE_HIGH>;
设备中通过对name的获取，可以获取gpio的gpio_chip设备，然后对此gpio设置输出高低电平等功能。
具体设备引用参考内核的gpio使用文档。

3.3 charger 使用 demo

- dts 配置示例

其他设备对 power supply 设备的引用通过设备树配置。= <&usb_power_supply>; 设备中通过对 name 的获取，可以获取 usb_power_supply 的 power_supply 设备，然后读取或设置此供电状态。
具体设备引用参考内核的 power supply 使用文档。

```
pmu0: pmu@34 {
...
    usb_power_supply: usb_power_supply {
        compatible = "x-powers,axp803-usb-power-supply";
        status = "okay";

    };
...
}

...
udc:udc-controller@0x05100000 {
    det_vbus_supply = <&usb_power_supply>;
```

```
};
...
```

- 驱动代码示例

模块驱动通过 devm_power_supply_get_by_phandle 获取 usb_power_supply 的 power_supply 设备，然后使用 power_supply_set_property 等接口，读取或设置此供电状态。

```
if (of_find_property(g_udc_pdev->dev.of_node, "det_vbus_supply", NULL))
    psy = devm_power_supply_get_by_phandle(&g_udc_pdev->dev,
                        "det_vbus_supply");

if (!psy || IS_ERR(psy)) {
    DMSG_PANIC("%s()%d WARN: get power supply failed\n",
           __func__, __LINE__);
} else {
    temp.intval = 500;

    power_supply_set_property(psy,
                POWER_SUPPLY_PROP_INPUT_CURRENT_LIMIT, &temp);
}
```

3.4 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。使用方法如下。

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。

4 FAQ

4.1 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。

4.1.1 调试工具

4.1.1.1 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 ------------
```

4.1.2 调试节点

4.1.2.1 max_microvolts 和 min_microvolts 节点

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。

4.1.2.2 /sys/class/regulator/regulator.x

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”。

4.1.2.3 /sys/kernel/debug/regulator/regulator_summary 节点

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。

4.1.2.4 regmap registers 节点

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。

- 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
```

- 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
```

4.1.2.5 axp_reg 节点

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。

```
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
```

4.1.2.6 debug_mask 节点

axp 驱动自定义节点 debug_mask 打开和关闭调试信息。相关调试信息参考具体的 PMIC 驱动。示例如下。

```
系统打印等级设置为8：
echo 8 > /proc/sys/kernel/printk
打开所有axp调试信息：
echo 0xf > /sys/class/axp/debug_mask
关闭所有axp调试信息：
echo 0x0 > /sys/class/axp/debug_mask
```

调试信息一般如下。

```
[  712.458412] ic_temp = 45
[  712.461311] vbat = 3977
[  712.464082] ibat = -779
[  712.466280] healthd: battery l=96 v=3977 t=30.0 h=2 st=3 c=-779 fc=5066880 chg=
[  712.475174] charge_ibat = 0
```

```
[  712.478448] dis_ibat = 779
[  712.481545] ocv = 4073
[  712.484239] rest_vol = 96
[  712.487182] rdc = 123
[  712.489862] batt_max_cap = 5066
[  712.493472] coulumb_counter = 4857
[  712.497583] AXP803_COULOMB_CTL = 0xe0
[  712.501803] ocv_percentage = 86
[  712.505436] col_percentage = 96
[  712.509061] bat_current_direction = 0
[  712.513386] ext_valid = 0
```

4.1.2.7 /sys/class/power_supply/

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。

4.2 常见问题

4.2.1 内核阶段

4.2.1.1 电池图标显示异常

电池状态的获取函数在 driver/power/supply 目录下的 xxx-battery.c 或 driver/power/supply/axp 目录下的 xxx-charger.c 中定义。以 axp2101 为例，系统需要获取电池状态时会调用 axp2101_get_property()，根据电池图标异常的类型可以进到对应的 case 路径下进行排查。代码如下：

```
1  static int axp210x_get_property(struct power_supply *psy,   enum power_supply_property psp,
2                                                              union power_supply_propval *val
       )
3  {
4      ...
5      switch (psp) {
6      case POWER_SUPPLY_PROP_CAPACITY_LEVEL: // customer modify
7          if (axp210x_info->regcache.soc == 100)
8              val->intval = POWER_SUPPLY_CAPACITY_LEVEL_FULL;
9          else if (axp210x_info->regcache.soc > 80)
10             val->intval = POWER_SUPPLY_CAPACITY_LEVEL_HIGH;
11         else if (axp210x_info->regcache.soc > axp210x_info->regcache.lowsocth)
12             val->intval = POWER_SUPPLY_CAPACITY_LEVEL_NORMAL;
13         else if (axp210x_info->regcache.soc < axp210x_info->regcache.lowsocth)
14             val->intval = POWER_SUPPLY_CAPACITY_LEVEL_LOW;
15         else if (axp210x_info->regcache.soc <= 1)
16             val->intval = POWER_SUPPLY_CAPACITY_LEVEL_CRITICAL;
17         else
18             val->intval = POWER_SUPPLY_CAPACITY_LEVEL_UNKNOWN;
```

```
            break;
    case POWER_SUPPLY_PROP_STATUS:
        ret = axp210x_get_bat_status(psy, val);
        break;
    case POWER_SUPPLY_PROP_PRESENT:
        val->intval = di->stat.bat_stat;
        break;
    ...
}
```

当前报出该问题的主要平台是使用 AXP803、AXP813、AXP223 的平台，造成电池图标显示异常的原因为 PMU 设计缺陷导致驱动读出的电池有效状态错误，电池状态返回 0 导致。

4.2.2 boot 阶段

4.2.2.1 开机流程错误

当前报出该问题的原因与上述内核电池图标显示异常原因相同，均是 PMU 设计缺陷导致电池有效位无法正常读出，导致驱动读取状态错误，返回错误值。目前 sunxi 平台在 boot 阶段的开机流程如下：

```
                                        |Y->为上次插着适配器关机，走关机充电流程
                        |Y->适配器是否存在|
                        |               |N->上次插着适配器关机，但是后来适配器移除，走关机流程
                        |
    |Y->上次关机时是否往  |                |->适配器接入开机：走关机充电流程
    |  PMU BUFF里面写值  |                |
  电|                    |                |          |->电池低电量：走关机充电流程
  池|                    |N->为正常开机，判断开机源|->按键开机|
  存|                                     |          |->电池非低电量：走正常开机流程
  在|                                     |
  ? |                                     |->无开机源：上次系统重启，走正常开机流程
    |
    |N->走开机流程，因为没有电池但系统能跑，一定是接了适配器
```

- brandy-1.0 中的开机流程开机时系统在 uboot 阶段最终会调用 board/sunxi/common/power check.c 中的 PowerCheck（）函数，通过层层调用最终执行 PMU 驱动的回调函数来获取电池的状态和适配器状态。以 AXP81X 为例，系统在 PowerCheck（）函数中会执行 axp_probe_battery_exist（）和 axp_probe_power_source()，其最终都是会调用到 u-boot-2014.07/driver/power/sunxi/axp81X.c 中的 axp81_probe_battery_exist() 、axp81_probe_power_status() 和 probe_pre_sys_mode() 来判断电池状态和适配器状态，代码如下：

```
int PowerCheck(void)
{
    ...
    //check battery
    BatExist = pmu_bat_unused?0:axp_probe_battery_exist();
```

```
    //check power bus
    PowerBus = axp_probe_power_source();
    pr_msg("PowerBus = %d( %d:vBus %d:acBus other: not exist)\n", PowerBus, AXP_VBUS_EXIST,
     AXP_DCIN_EXIST);

    power_limit_for_vbus(BatExist,PowerBus);

    if(BatExist <= 0)
    {
        printf("no battery exist\n");
        EnterNormalBootMode();
        return 0;
    }

    //if android call shutdown when  charing , then boot should enter android charge mode
    if((PMU_PRE_CHARGE_MODE == ProbePreSystemMode()))
    {
        if(PowerBus)
        {
            EnterAndroidChargeMode();
        }
        else
        {
            printf("pre system is charge mode,but without dc or ac, should be ShowDown\n");
            EnterNormalShutDownMode();
        }
        return 0;
    }
    ...

}
```

- brandy-2.0 中的开机流程开机时系统在 uboot 阶段会调用 board/sunxi/power_manage.c 中的 sunxi_update_axp_info() 获取开机源和适配器状态，其最终会调用对应 AXP 驱动下的.get_poweron_source 更新 chargemode 属性，当 chargemode 属性为 1 时，系统走关机充电流程，当 chargemode 为 0 时，系统走开机流程。代码如下：

```
int sunxi_update_axp_info(void)
{
    ...
    if ((val == -1) && (pmu_get_sys_mode() == SUNXI_CHARGING_FLAG)) {
        val = AXP_BOOT_SOURCE_CHARGER;
        pmu_set_sys_mode(0);
    }
    switch (val) {
    case AXP_BOOT_SOURCE_BUTTON:
        strncpy(bootreason, "button", sizeof("button"));
        break;
    case AXP_BOOT_SOURCE_IRQ_LOW:
        strncpy(bootreason, "irq", sizeof("irq"));
        break;
    case AXP_BOOT_SOURCE_VBUS_USB:
        strncpy(bootreason, "usb", sizeof("usb"));
        break;
    case AXP_BOOT_SOURCE_CHARGER:
```

```
        strncpy(bootreason, "charger", sizeof("charger"));
        gd->chargemode = 1;
        break;
    case AXP_BOOT_SOURCE_BATTERY:
        strncpy(bootreason, "battery", sizeof("battery"));
        break;
    default:
        strncpy(bootreason, "unknow", sizeof("unknow"));
        break;
    }
    env_set("bootreason", bootreason);
    return 0;
    ...
}
```

4.2.2.2 模块电压不对

系统启动到 uboot 阶段，会解析平台的板级配置文件 sysconfig.fex 中的属性来对相应的输出进行配置，sysconfig.fex 中的部分属性字段如下：

```
...
[power_sply]
dcdc1_vol                   = 1003300
aldo1_vol                   = 1001800
; aldo2_vol                  = 1001800
...
```

其中 xxxx_vol 代表需要配置的输出，前缀 100 代表在 uboot 阶段这路电需要被打开，前缀 000 代表在 uboot 阶段这路电会被关闭；后缀 3300、1800 等代表该路的输出电压，单位为 mV；如无须对某路电进行操作，在属性前面增加“；”注释掉，或者不在该文件中定义该路输出信息即可。

如果在系统启动过程中发现某路输出电压不对或者出现电压跳变的情况，可以优先排查 sysconfig.fex 中的各路输出属性。

著作权声明

商标声明

免责声明