



# Linux PMIC 开发指南

版本号: 3.1  
发布日期: 2023.11.13

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.07.22	AWA0863	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。
2.1	2021.10.22	AWA1691	新增部分节点功能描述。
2.2	2022.03.10	AWA1691	新增部分节点功能描述。
2.3	2022.04.27	AWA1691	新增关于 A33 的内容。
2.4	2022.05.25	AWA1691	修改部分格式。
2.5	2022.05.25	AWA1691	新增 T507 的适用内核范围。
2.6	2022.11.02	AWA1691	T3 系列 & A40i 系列的适用。
2.7	2023.03.03	AWA1691	1. 优化结构。 2. 整理和新增相关术语介绍。 3. 新增 A523 的相关内容。
2.8	2023.04.25	AWA1691	新增 mr527 的相关内容。®
2.9	2023.06.21	AWA1691	新增 AI985 的相关内容。
3.0	2023.08.01	AWA1691	新增 T527 的相关内容。
3.1	2023.11.13	AWA1691	1. 更新关于过流、电量更新唤醒等属性配置说明。 2. 修改部分格式。

# 目 录

<b>1 前言</b>	<b>1</b>
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
1.4 相关术语介绍	1
<b>2 模块介绍</b>	<b>3</b>
2.1 模块功能介绍	3
2.2 模块配置介绍	3
2.2.1 sys_config.fex 配置说明	3
2.2.1.1 boot0	3
2.2.2 Device Tree 配置说明	4
2.2.2.1 uboot	4
2.2.2.2 kernel	5
2.2.3 kernel menuconfig 配置说明	8
2.3 驱动框架介绍	9
2.4 源码结构介绍	10
2.4.1 boot0	11
2.4.2 uboot	11
2.4.3 kernel	11
<b>3 模块接口说明</b>	<b>13</b>
<b>4 模块使用范例</b>	<b>14</b>
4.1 regulator 使用 demo	14
4.2 gpio 使用 demo	14
4.3 power_supply 使用 demo	14
4.4 watchdog 使用 demo	15
<b>5 调试方法</b>	<b>16</b>
5.1 调试工具	16
5.1.1 调试 power key	16
5.2 调试节点	17
5.2.1 设置 regulator 电压	17
5.2.2 获取 regulator 引用设备	17
5.2.3 查询 regulator 状态	17
5.2.4 手动读写 PMIC 寄存器	18
5.2.4.1 使用标准 regmap registers 节点	18
5.2.4.2 axp_reg 节点	19

5.2.5	查看供电状态	19
5.2.6	其他 AXP 调试节点	21
5.2.6.1	debug_mask 节点	21
<b>6</b>	<b>FAQ</b>	<b>23</b>
6.1	常见功能配置	23
6.1.1	唤醒/开关机/重启配置	23
6.1.1.1	唤醒源配置	23
6.1.1.2	开关机配置	25
6.1.1.3	重启配置	25
6.1.2	powerkey 自定义属性	26
6.1.3	无电池方案属性	27
6.1.4	uboot 供电调节	27
6.1.5	开机判断配置	28
6.1.6	GPIO 口耐压值配置	29
6.1.7	基础电池属性	29
6.1.7.1	电池满充电压、电池内阻、电池容量	29
6.1.7.2	低电警告电量	30
6.1.7.3	输入限流/限压	31
6.1.8	充电属性配置	32
6.1.8.1	bc1.2 模式配置	32
6.1.8.2	电池充电限流	33
6.1.8.3	预充电电流限制	33
6.1.8.4	截止充电电流	34
6.1.9	充电指示灯配置	34
6.1.10	drivebus 属性配置	35
6.1.11	NTC 温控属性配置	36
6.1.11.1	ntc、jeita 使能、jeita 限流参数	36
6.1.11.2	电池温度参数、ts 电流配置	37
6.1.11.3	开关机温度限制 (uboot)	39
6.1.11.4	触发限流、停充、关机的电压阈值	39
6.1.12	type-c 属性配置	40
6.1.13	ACIN 属性配置	41
6.1.13.1	方案原理图	41
6.1.13.2	menuconfig 配置	42
6.1.13.3	Device Tree 配置	42
6.1.14	外挂 DCDC 配置	43
6.1.14.1	menuconfig 配置	43
6.1.14.2	Device Tree 配置	43
6.1.15	其他配置	45
6.2	常见问题	45
6.2.1	内核阶段-电池图标显示异常	45
6.2.2	uboot 阶段-开机流程错误	47

## 表 格

表 1-1	适用产品列表	1
表 1-2	术语介绍	1
表 1-3	相同驱动 AXP 对照表	2
表 2-1	不同版本的配置文件一览	3
表 5-1	battery 状态信息一览	20
表 5-2	usb 状态信息一览	21
表 6-1	外挂芯片型号、节点简称和基地址对照表	44

ALLWINNER®

# 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
A133	Linux-5.10/5.15	bsp/drivers/power/
T507	Linux-5.10	bsp/drivers/power/
A40i/A40i-C/A40i-H	Linux-5.10	bsp/drivers/power/
T3/T3-C/T3-Pro	Linux-5.10	bsp/drivers/power/
A523	Linux-5.15	bsp/drivers/power/
MR527	Linux-5.15	bsp/drivers/power/
AI985	Linux-5.15	bsp/drivers/power/
T527	Linux-5.15	bsp/drivers/power/

## 1.4 相关术语介绍

表 1-2: 术语介绍

术语	说明
PMU	电源管理单元。
BMU	电池控制器。

术语	说明
PMIC	电源管理芯片。
AXP	全志 PMU 的名称，如 AXP2202 等。
LDO	文档里指低压差线性稳压器经过调节的输出。
DC-DC	文档里把直流变直流由开关方式实现的器件和对应输出叫 DCDC。
Regulator	linux 内核对 LDO、DC-DC 的控制核心。
powerkey	电源按键和其控制核心。
TWI	同等于 I2C，AXP 的使用需要用到 I2C 的通讯进行读写寄存器。
GPIO	通用型输入和输出，其引脚可以作为输入、输出或其他特殊功能，如中断。
WDT	看门狗计时器，一种硬件计时设备，常用于在程序异常时的处理和重启。
VBUS	文档内指从 usb 线输入 pmu 的供电或供电源。
NTC	热敏电阻器，文档内指电池过温停充的策略。
JETIA	一种锂电池使用和充电规范，文档内指电池过温限流的策略。

表 1-3: 相同驱动 AXP 对照表

驱动命名	对应 AXP 编号
AXP803	AXP803、AXP707
AXP2202	AXP2202、AXP717
AXP1530	AXP1530、AXP313、AXP323
AXP22x	AXP221s、AXP223

## 2 模块介绍

如无特殊说明，以下介绍均适用于所有版本的 AXP

### 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 模块配置介绍

这部分将讲述一个正常运行的 PMIC 驱动需要具备的基础配置。

其中，PMIC 模块在 dtsi 中无用户可用配置，device tree 相关的所有配置项几乎都在 device 下，根据配置生效时间分布在不同的配置文件中。

表 2-1: 不同版本的配置文件一览

生效时间	配置文件
boot0	sys_config.fex
uboot	uboot-board.dts
kernel	board.dts

#### 2.2.1 sys\_config.fex 配置说明

##### 2.2.1.1 boot0

power mode 属性决定了当前平台使用哪个 PMU，需 boot0 代码支持解析。

目前仅 A133/T509 方案支持解析该节点。

以 AXP2202 为例

```
;
;power_mode = axp_type, 0:axp81X, 1:dummy, 2:axp806, 3:axp2202, 4:axp858
;
```



```
[target]
power_mode = 3
```

## 2.2.2 Device Tree 配置说明

### 2.2.2.1 uboot

uboot 中需要配置一份和内核 regulator 几乎相同的属性，具体属性含义可参考 kernel 章节内的 regulator。

同时，有些模块如 twi 在调压后需等电压稳定才能操作，因此增加 power\_delay 作为延时配置。

以 AXP2202 为例：

```
&twi6 {
    clock-frequency = <200000>;
    pinctrl-0 = <&s_twi0_pins_a>;
    twi-supply = <&reg_aldo3>;
    no_suspend = <1>;
    twi_drv_used = <1>;
    status = "okay";
    pmu0: pmu@34 {
        compatible = "x-powers,axp2202";
        status = "okay";
        /* interrupts = <0 IRQ_TYPE_LEVEL_LOW>;
        * interrupt-parent = <&gic>; */
        x-powers,drive-vbus-en;

        wakeup-source;

        regulator0: regulators@0 {
            reg_dcdc1: dcdc1 {
                regulator-name = "axp2202-dcdc1";
            };
            reg_dcdc2: dcdc2 {
                regulator-name = "axp2202-dcdc2";
            };
            .....
        }

        &power_delay {
            device_type = "power_delay";
            aldo3_vol_delay = <20000>;
        };
    };
}
```

```
&twi6 {
    twi-supply = <&reg_aldo3>;
    这路i2c使用reg_aldo3供电
}

&power_delay {
    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。  
};
```

### 2.2.2.2 kernel

- mfd

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，包含 regulator、powerkey 和 power\_supply (battery、usb、ac) 三个子系统。

因此，在内核要使用 PMIC 需要先配置、打开主设备，其次才能去配置和使用三个子系统。

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

主设备的节点为 pmu0，放置在 PMIC 所使用的 i2c 节点下。

后面三个子系统的节点均放在 pmu0 下。

以 AXP2202 为例：

```
pmu0: pmu@0{  
    compatible = "x-powers,axp2202";  
    reg = <0x34>;  
    #address-cells = <1>;  
    #size-cells = <0>;  
    interrupts = <0 IRQ_TYPE_LEVEL_LOW>;  
    interrupt-parent = <&nmi_intc>;  
    status = "okay";  
    x-powers,drive-vbus-en;  
    wakeup-source;  
    .....  
}
```

```
compatible <char>  
    设备匹配名，对应AXP的型号  
  
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

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

是否将 N\_VBUSEN 作为输出以对外供电

- regulator

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。

Virtual regulator 则是作为虚拟设备，通过引用对应的 regulator\_dev 以增加一些调试节点。

regulator 属性配置和具体含义可参考内核原生 regulator 使用文档：Documentation/devicetree/bindings/regulator/regulator.yaml

以 AXP2202 为例：

```
regulator0: regulators@0{
    reg_dcdc1: dcdc1 {
        regulator-name = "axp2202-dcdc1";
        regulator-min-microvolt = <1500000>;
        regulator-max-microvolt = <3400000>;
        regulator-boot-on;
        regulator-always-on;
    };
    reg_dcdc2: dcdc2 {
        regulator-name = "axp2202-dcdc2";
        regulator-min-microvolt = <500000>;
        regulator-max-microvolt = <1540000>;
        regulator-boot-on;
        regulator-always-on;
    };
    .....
};
virtual-dcdc1 {
    compatible = "xpower-vregulator,dcdc1";
    dcdc1-supply = <&reg_dcdc1>;
};
.....
```

```
reg_ald01: ald01{
    regulator-name = "axp2202-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
};

virtual-dcdc1 {
    compatible = "xpower-vregulator,dcdc1";
    为虚拟设备的名称
    dcdc1-supply = <&reg_dcdc1>;
    引用对应的regulator_dev
};
```

- powerkey

power key 设备为按键设备，具体的说为电源按键设备。

以 AXP2202 为例：

```
powerkey0: powerkey@0{
    status = "okay";
    compatible = "x-powers,axp2101-pek";
    .....
};
```

```
compatible <char>
    所有版本的AXP的匹配项都是"x-powers,axp2101-pek"
```

- power\_supply

power\_supply 属性配置，根据 ac、usb、battery 三种供电方式分成三个子节点。

其中 battery 为电池节点，ac 和 usb 为充电节点。

:::note

具体节点情况视具体 axp 而定，部分 axp 仅包含 usb、battery 两个子节点。

以下例子以包含三种供电方式的 axp803 为例。

:::

```
ac_power_supply: ac-power-supply {
    compatible = "x-powers,axp803-ac-power-supply";
    status = "okay";
    .....
```

```
};  
usb_power_supply: usb_power_supply {  
    compatible = "x-powers,axp803-usb-power-supply";  
    status = "okay";  
    .....  
};  
battery_power_supply: battery-power-supply {  
    compatible = "x-powers,axp803-battery-power-supply";  
    status = "okay";  
    .....  
};
```

compatible <char>  
匹配不同的供电模式

注1：在一些AXP（例如AXP2202）中，battery的供电节点是以“bat\_power\_supply”来命名

注2：AXP2202下有两个充电节点，分别为usb和acin，其中acin的含义与AXP803的ac的不同，详见《模块功能配置-ACIN属性配置》章节

### 2.2.3 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 AXP2202 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

```
Allwinner BSP ---> Device Drivers --->  
I2C Drivers ---> <*> I2C Support for Allwinner SoCs  
PMIC Drivers ---> <*> X-POWERS AXP2101 PMICs with I2C
```

- regulator

```
Allwinner BSP ---> Device Drivers --->  
PMIC Drivers ---> <*> X-POWERS AXP2101 PMIC Regulators
```

- charger

```
Allwinner BSP ---> Device Drivers --->  
PMIC Drivers ---> <*> AXP2202 Power Supply Driver
```

- power key

```
Allwinner BSP ---> Device Drivers --->
PMIC Drivers ---> <*> X-POWERS AXP2101 Power Button Driver
```

- virtual regulator

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

- acin

```
Allwinner BSP ---> Device Drivers --->
PMIC Drivers ---> <*> AXP2202 Power Virtual ACIN
```

## 2.3 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。

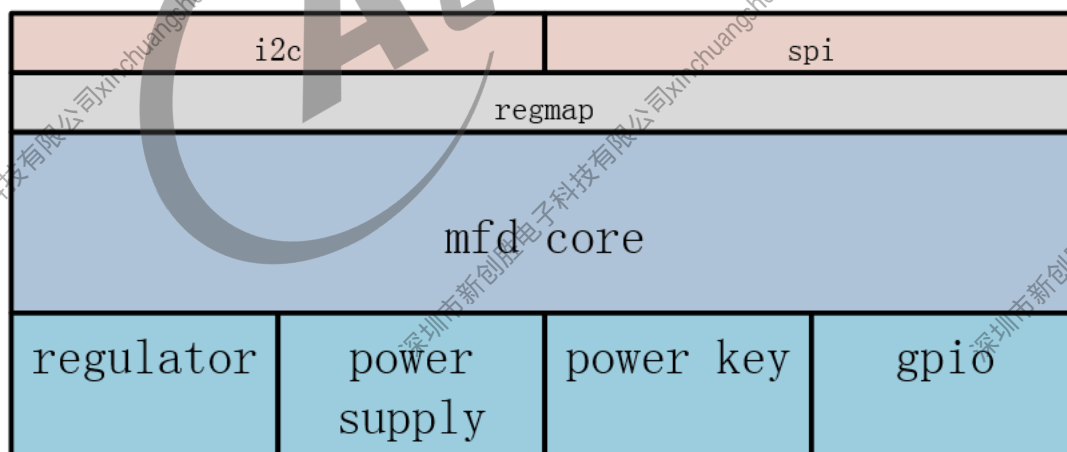


图 2-1: PMIC 驱动总体结构

其中：

1. mfd 目录下为 PMIC 的 mfd 代码，负责管理 PMIC 整体配置；
2. regulator 目录下为 PMIC 的 regulator 驱动代码，负责对 LDO、DC-DC 等的控制；
3. input/misc 目录下为 PMIC 的 power key 驱动代码，负责电源按键的管理；

4. power/supply 目录下为 PMIC 的 charger 驱动代码，负责管理电池和充电等事项。5. PMIC 的 gpio 驱动代码，负责管理 PMIC 芯片上的 gpio 引脚。

以 AXP2202 为例，其在原理图如下所示：

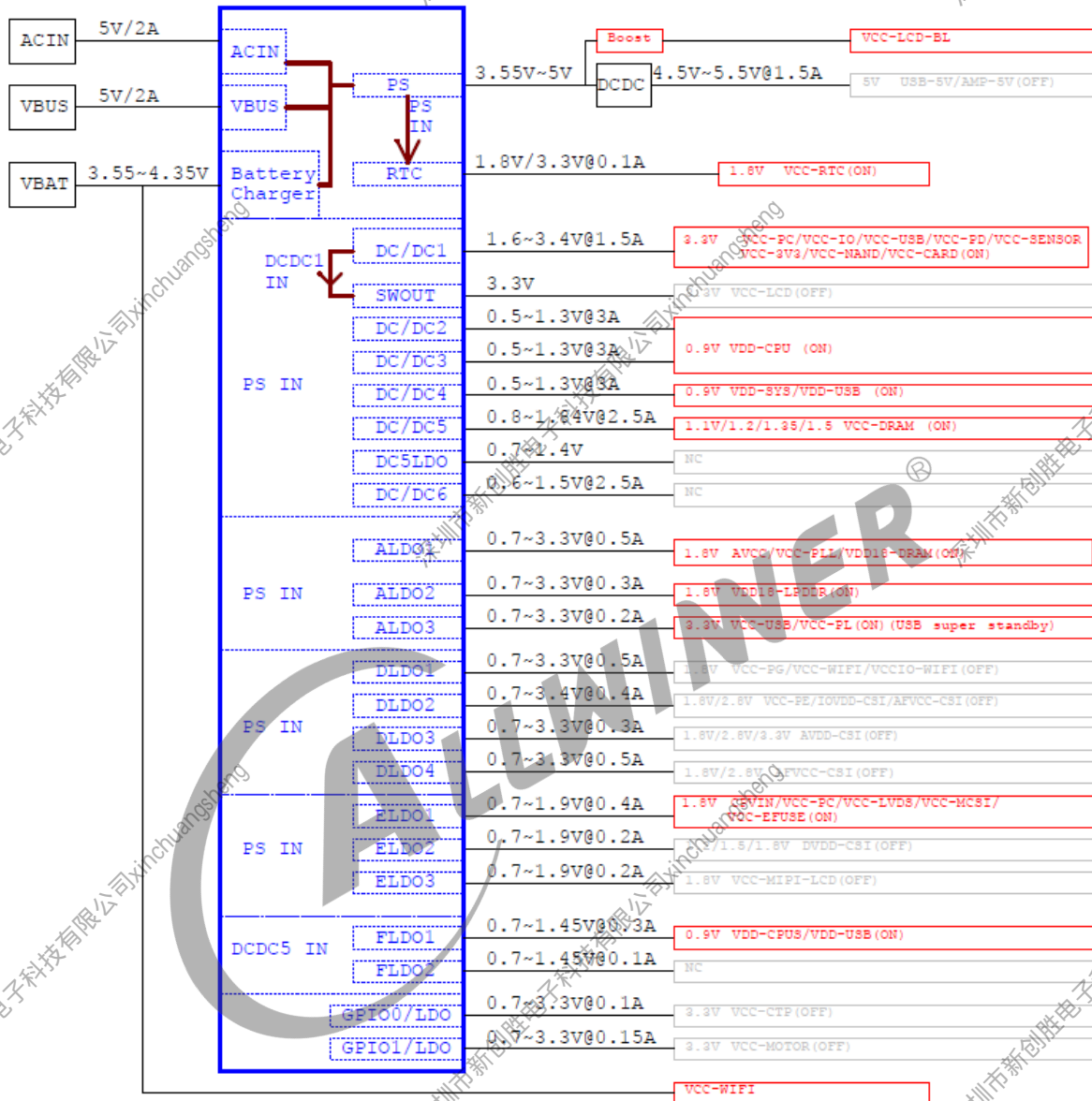


图 2-2: PMIC 原理框图

## 2.4 源码结构介绍

以下源码结构的介绍顺序为开机流程的顺序：

## 2.4.1 boot0

在 boot0 中，AXP PMIC 主要负责以下事项：

- 调节 cpu、sys 的电压
- 给接口让 dram 进行调压

其源码结构以在 sun50iw10p1 上使用 axp2202 为例：

```
brandy/brandy-2.0/spl
|-- board/sun50iw10p1/board.c
|-- drivers/power/
|   |-- axp2202.c
|   |-- axp.c
...
```

## 2.4.2 uboot

在 uboot 中，AXP PMIC 的代码分为 PMU（管理 LDO、DC-DC、识别按键）和 BMU（管理电池、充电）两个部分，主要负责以下事项：

- 开机状态判断：正常开机或进入 Android 充电模式
- 电池低电量检测
- 预充电
- 电池温度状态检测
- 各路电的电压和开关
- 控制各 dc/dc 的 pwm 模式

其源码结构以 axp2202 为例：

```
brandy/brandy-2.0/u-boot-2018
|-- board/sunxi/power_manage.c
|-- drivers/sunxi_power/
|   |-- axp.c
|   |-- bmu.c
|   |-- pmu.c
|   |-- bmu_axp2202.c
|   |-- pmu_axp2202.c
...
```

## 2.4.3 kernel

note



AXP2202 下有两个充电管理单元，分别对应管理 usb 充电和火牛（ACIN）充电，后者与 AXP803 的 ac 不同，会在《模块功能配置-ACIN 属性配置》章节进行详细介绍

...

```
bsp/drivers/power
|-- mfd/
|   |-- axp2101.c
|   |-- axp2101-i2c.c
|-- supply/
|   |-- axp2202_usb_power.c
|   |-- axp2202_battery.c
|   |-- axp2202_gpio_power.c
|-- regulator/axp2101-regulator.c
|-- power_key/axp2101-pek.c
...
```

### 3 模块接口说明

暂无

## 4 模块使用范例

### 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

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。

### 4.3 power\_supply 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。

```
1 if (of_find_property(g_udc_pdev->dev.of_node, "det_vbus_supply", NULL))  
2     psy = devm_power_supply_get_by_phandle(&g_udc_pdev->dev,  
3         "det_vbus_supply");  
4  
5 if (!psy || IS_ERR(psy)) {  
6     DMSG_PANIC("%s() %d WARN: get power supply failed\n",  
7         __func__, __LINE__);  
8 } else {  
9     temp.intval = 500;  
10  
11     power_supply_set_property(psy,  
12         POWER_SUPPLY_PROP_INPUT_CURRENT_LIMIT, &temp);  
13 }
```

## 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。

## 5 调试方法

### 5.1 调试工具

#### 5.1.1 调试 power key

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 -----
```

## 5.2 调试节点

### 5.2.1 设置 regulator 电压



需要确保虚拟设备 Virtual regulator 已编译，详见《模块基础配置-regulator》

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-consumer.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。



可以通过这个节点，将某路电设置成常开。

### 5.2.2 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，

以 AXP2101 为例：

```
进入/sys/class/regulator/regulator.1
通过ls查看当前目录下的目录，即可确定有哪几路引用设备。
```

也可以进入 regulator 的 debugfs 结点，用来查看 regulator 的 map 信息，详见下章。

### 5.2.3 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。

```
要求内核已打开DEBUG_FS的宏
```

以 AXP2101 的设备举例，首先需要 mount debugfs 文件系统。

然后就可以很直观的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。

```
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	0mA		
axp2101-dcdc1	0	7	0	3300mV	0mA	1500mV	3400mV
spi0				0mV	0mA		
sdcard				0mV	0mA		
sdcard				0mV	0mA		
sdcard				0mV	0mA		
sdcard				0mV	0mA		
sdcard				0mV	0mA		
reg-virt-consumer.1				0mV	0mA		
axp2101-dcdc2	0	1	0	900mV	0mA	500mV	1540mV
reg-virt-consumer.2				0mV	0mA		
axp2101-dcdc3	0	2	0	1000mV	0mA	500mV	3400mV
cpu0				1000mV	1000mA		
reg-virt-consumer.3				0mV	0mA		
axp2101-dcdc4	0	1	0	1500mV	0mA	500mV	1840mV
reg-virt-consumer.4				0mV	0mA		
axp2101-dcdc5	0	1	0	1400mV	0mA	1400mV	3700mV
reg-virt-consumer.5				0mV	0mA		
axp2101-rtclldo	0	0	0	1800mV	0mA	1800mV	1800mV
axp2101-rtclldo1	0	0	0	1800mV	0mA	1800mV	1800mV
axp2101-alldo1	0	1	0	1800mV	0mA	500mV	3500mV
reg-virt-consumer.8				0mV	0mA		
axp2101-alldo2	0	2	0	3300mV	0mA	500mV	3500mV
spi2				0mV	0mA		
reg-virt-consumer.9				0mV	0mA		
axp2101-alldo3	0	1	0	3300mV	0mA	500mV	3500mV
reg-virt-consumer.10				0mV	0mA		
axp2101-alldo4	0	1	0	3300mV	0mA	500mV	3500mV
reg-virt-consumer.11				0mV	0mA		
axp2101-blldo1	0	1	0	1800mV	0mA	500mV	3500mV
reg-virt-consumer.12				0mV	0mA		
axp2101-blldo2	0	1	0	3300mV	0mA	500mV	3500mV
reg-virt-consumer.13				0mV	0mA		
axp2101-dldo1	1	1	0	1200mV	0mA	500mV	3500mV
reg-virt-consumer.14				0mV	0mA		
axp2101-dldo2	0	1	0	1200mV	0mA	500mV	1400mV
reg-virt-consumer.15				0mV	0mA		
axp2101-cpusldo	0	0	0	900mV	0mA	500mV	1400mV

## 5.2.4 手动读写 PMIC 寄存器

### 5.2.4.1 使用标准 regmap registers 节点

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。

```
mount -t debugfs none /sys/kernel/debug
```

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。

#### 📖 说明

需要注意，原生内核默认不支持写操作，需要增加宏定义 `REGMAP_ALLOW_WRITE_DEBUGFS`  
对应驱动：drivers/base/regmap/regmap-debugfs.c

#### • 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
```

### 5.2.4.2 axp\_reg 节点

另外，还支持 axp 驱动自定义节点 axp\_reg 读写寄存器。

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
```

### 5.2.5 查看供电状态

根据供电的种类，读取 /sys/class/power\_supply/{battery,usb,ac} 下面状态，判断一致性。



以 AXP2202 为例：

```
/ # cat sys/class/power_supply/axp2202-battery/uevent
POWER_SUPPLY_NAME=axp2202-battery
POWER_SUPPLY_TYPE=Battery
POWER_SUPPLY_PRESENT=1
POWER_SUPPLY_VOLTAGE_NOW=4165000
POWER_SUPPLY_ENERGY_FULL_DESIGN=1771
POWER_SUPPLY_STATUS=Charging
POWER_SUPPLY_CAPACITY_ALERT_MIN=15
POWER_SUPPLY_TEMP=300
POWER_SUPPLY_CAPACITY=90
POWER_SUPPLY_TEMP_ALERT_MIN=-200000
POWER_SUPPLY_TEMP_ALERT_MAX=200000
POWER_SUPPLY_TEMP_AMBIENT_ALERT_MIN=-200000
POWER_SUPPLY_TEMP_AMBIENT_ALERT_MAX=200000
POWER_SUPPLY_TIME_TO_EMPTY_NOW=52440
POWER_SUPPLY_TIME_TO_FULL_NOW=1260
POWER_SUPPLY_MANUFACTURER=xpower,axp2202
POWER_SUPPLY_CAPACITY_LEVEL=High
POWER_SUPPLY_CONSTANT_CHARGE_CURRENT=1044
POWER_SUPPLY_HEALTH=Good
POWER_SUPPLY_CHARGE_COUNTER=1593
POWER_SUPPLY_CHARGE_FULL=1771
```

表 5-1: battery 状态信息一览

参数名称	对应含义
POWER_SUPPLY_NAME	驱动名称
POWER_SUPPLY_TYPE	驱动类型（电池）
POWER_SUPPLY_PRESENT	电池存在状态
POWER_SUPPLY_VOLTAGE_NOW	电池电压
POWER_SUPPLY_ENERGY_FULL_DESIGN	满充电量
POWER_SUPPLY_STATUS	充电状态
POWER_SUPPLY_CAPACITY_ALERT_MIN	警告电量
POWER_SUPPLY_TEMP	电池温度/芯片温度（根据配置变化）
POWER_SUPPLY_CAPACITY	电池电量百分比
POWER_SUPPLY_TEMP_ALERT_MIN	电池欠温关机温度
POWER_SUPPLY_TEMP_ALERT_MAX	电池过温关机温度
POWER_SUPPLY_TEMP_AMBIENT_ALERT_MIN	电池欠温停充温度
POWER_SUPPLY_TEMP_AMBIENT_ALERT_MAX	电池过温停充温度
POWER_SUPPLY_TIME_TO_EMPTY_NOW	放电放光时间
POWER_SUPPLY_TIME_TO_FULL_NOW	电池充满时间
POWER_SUPPLY_MANUFACTURER	电池驱动名字
POWER_SUPPLY_CAPACITY_LEVEL	电池电量情况
POWER_SUPPLY_CONSTANT_CHARGE_CURRENT	充电电流上限
POWER_SUPPLY_HEALTH	电池状态
POWER_SUPPLY_CHARGE_COUNTER	当前电流
POWER_SUPPLY_CHARGE_FULL	满充电量

```
/ # cat sys/class/power_supply/axp2202-usb/uevent  
POWER_SUPPLY_NAME=axp2202-usb  
POWER_SUPPLY_TYPE=USB  
POWER_SUPPLY_PRESENT=1  
POWER_SUPPLY_VOLTAGE_NOW=0  
POWER_SUPPLY_MANUFACTURER=xpower,axp2202  
POWER_SUPPLY_ONLINE=1  
POWER_SUPPLY_INPUT_CURRENT_LIMIT=2500  
POWER_SUPPLY_VOLTAGE_MIN_DESIGN=3960
```

表 5-2: usb 状态信息一览

参数名称	对应含义
POWER_SUPPLY_NAME	驱动名称
POWER_SUPPLY_TYPE	驱动类型 (usb)
POWER_SUPPLY_PRESENT	usb 接入状态
POWER_SUPPLY_VOLTAGE_NOW	vbus 设置电压
POWER_SUPPLY_ENERGY_FULL_DESIGN	usb 驱动名字
POWER_SUPPLY_ONLINE	usb 接入状态
POWER_SUPPLY_INPUT_CURRENT_LIMIT	输入限流
POWER_SUPPLY_VOLTAGE_MIN_DESIGN	vindpm 电压

## 5.2.6 其他 AXP 调试节点

### 5.2.6.1 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。



当前仅 AXP803 支持

以 AXP803 为例：

```
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  
[ 712.475174] 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
```

ic\_temp <u32>

芯片温度或电池温度，单位°C。

rdc <u32>

电池内阻，单位Ω。

vbat、ibat <u32>

电池电压、电流，单位分别为mV和mA。

charge\_ibat、dis\_ibat <u32>

充电、放电电流，单位为mA。

rest\_vol <u32>

电池剩余电量，为百分比。

AXP803\_COULOMB\_CTL <u32>

AXP803\_COULOMB\_CTL (0xB8) 寄存器内的值

batt\_max\_cap <u32>

满充电量，单位mAh。

ocv <u32>

电量计中的值。

coulumb\_counter <u32>

库伦计中的值

ocv\_percentage、col\_percentage <u32>

电量计、库伦计百分比

bat\_current\_direction <u32>

指示USB连接有效且可以选择vbus

0：无效

1：有效

ext\_valid <u32>

判断是充电还是放电状态

0：放电

1：充电

## 6 FAQ

### 6.1 常见功能配置

这部分将根据实现的不同功能进行配置的讲解，可根据功能检索对应配置项。

#### 说明

以下代码演示如无特殊说明均以 linux5.15 及以上使用 axp2202 为例，且为通用配置。  
其余版本可以参考模块配置介绍寻找对应配置文件。

#### 6.1.1 唤醒/开关机/重启配置

##### 6.1.1.1 唤醒源配置

#### 说明

按键部分的唤醒配置归类在 powerkey 自定义属性中

```
device/...../board.dts;
pmu0: pmu@34 {
    .....
    pmu_irq_wakeup = <1>;
    wakeup_source;

    ac_power_supply: ac-power-supply {
        .....
        wakeup_ac_in;
        wakeup_ac_out;
        .....
    };

    usb_power_supply: usb_power_supply {
        .....
        wakeup_usb_in;
        wakeup_usb_out;
        .....
    };

    battery_power_supply: battery-power-supply {
        .....
        wakeup_bat_out;
        wakeup_new_soc;
        /* 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; */
.....
};
.....
};

```

wakeup-source <bool>

是否作为唤醒源，在使用PMU任何中断时必须配置

pmu\_irq\_wakeup

trigger irq wakeup or not when sleep or power down, default 0

0: not wakeup

1: wakeup

wakeup\_ac\_in <bool>

ac插入唤醒使能

wakeup\_ac\_out <bool>

ac拔出唤醒使能

wakeup\_usb\_in <bool>

usb插入唤醒使能

wakeup\_usb\_out <bool>

usb拔出唤醒使能

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\_new\_soc <bool>

电池电量更新唤醒使能

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，欠温关机必须配置

```
wakeup_bat_ovtemp_work <bool>  
    电池高温工作唤醒使能，过温关机必须配置
```

### 6.1.1.2 开关机配置

#### 说明

关机充电电流归类在基础充电属性中

```
device/...../board.dts:  
pmu0: pmu@0{  
    .....  
    pmu_powerok_noreset = <0>;  
    pmu_hot_shutdown = <1>;  
    pmu_hot_shutdown_value = <125>;  
    pmu_over_current = <1>;  
    .....  
};
```

```
pmu_powerok_noreset <bool>  
    powerok pin 下拉复位功能使能，默认disable。  
    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_hot_shutdown  
    配置触发pmu过温保护使能，默认enable。  
    0: disable  
    1: enable  
    **视具体的PMU驱动来定，部分型号PMU驱动暂未支持该功能**  
    **当前支持AXP: AXP2101、AXP2202、AXP803、AXP221/AXP223、AXP809、AXP202/AXP209、AXP152、AXP806、**  
    **AXP1530、AXP858**  
  
pmu_hot_shutdown_value  
    配置触发pmu过温保护时的温度，默认值跟随pmu型号。  
    **视具体的PMU驱动来定，部分型号PMU驱动暂未支持该功能**  
    **当前支持AXP: AXP2101、AXP2202、AXP803、AXP221/AXP223、AXP809、AXP202/AXP209、AXP152、AXP806、**  
    **AXP1530、AXP858**  
  
pmu_over_current  
    配置触发pmu的ldo过流保护使能，默认enable。  
    **视具体的PMU驱动来定，部分型号PMU驱动暂未支持该功能**  
    **当前支持AXP: AXP2101、AXP2202**
```

### 6.1.1.3 重启配置

```
device/...../board.dts:  
pmu0: pmu@0{  
    .....  
    pmu_reset = <1>;  
    .....  
};
```

pmu\_reset  
长按按键16s后，进行关机/重启，默认关机  
0: 关机  
1: 重启

## 6.1.2 powerkey 自定义属性

```
device/...../board.dts:  
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;  
};
```

pmu\_powkey\_off\_time <u32>  
控制按下多长时间响应poweroff事件，默认6s  
可选的值根据不同AXP变化，示例中6000指6s

pmu\_powkey\_off\_func <u32>  
控制power\_off事件功能,如果不配置，默认为关机  
1 复位系统  
0 关机

pmu\_powkey\_off\_en <u32>  
控制按键关机使能，默认使能  
1 使能  
0 不使能

pmu\_powkey\_long\_time <u32>  
控制ponlevel 寄存器，默认1.5s  
可选的值根据不同AXP变化，示例中1500指1.5s

pmu\_powkey\_on\_time <u32>  
控制按钮按下多长时间开机，默认1s  
128 0.128s  
512 0.512s  
1000 1s  
2000 2s

pmu\_pwrok\_time <u32>  
delay of PWROK after all power output good, default 64ms  
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



### 6.1.3 无电池方案属性

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_bat_unused = <0>;
    .....
};

device/...../uboot-board.dts:
&power_sply {
    .....
    battery_exist = <0>;
    .....
};
```

pmu\_bat\_unused <bool>

该属性不配置默认为0，仅内核生效。

1：强制判断为不使用电池

0：根据实际寄存器情况来判断

\*\*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\*\*

\*\*当前支持AXP：AXP221s/AXP223、AXP2585、AXP203/AXP209\*\*

battery\_exist <bool>

该属性不配置默认为0，仅uboot生效。

1：强制判断为不使用电池

0：根据实际寄存器情况来判断

注：在配置无电池方案的同时，建议关闭NTC温控属性功能，详见《模块功能配置-NTC温控属性配置》

### 6.1.4 uboot 供电调节

在 uboot 阶段，供电可调节：

1. 各路电的电压和开关
2. dcdc 的 pwm 模式

```
device/...../uboot-board.dts:
&power_sply {
    aldo1_vol = <1101800>;
    aldo2_vol = <1101800>;
    aldo3_vol = <1003300>;
    aldo4_vol = <1001800>;
    blldo1_vol = <3300>;
    blldo2_vol = <1001800>;
    blldo4_vol = <1101800>;
    clldo1_vol = <1001800>;
    clldo3_vol = <1003300>;
    clldo4_vol = <1103300>;
    cpusldo_vol = <1000900>;
    dcdc1_mode = <1>;
    dcdc2_mode = <1>;
};
```



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：110，这路电在uboot阶段打开，但是烧写时会关闭

前缀：000，这路电在uboot阶段关闭，可省去

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，但同时会增大功耗。

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

## 6.1.5 开机判断配置

以下几项内容跟开机判断相关：

1. 强制电池不存在判断
2. 机器的开机模式
3. 开机最低电压和电量

device/……/uboot-board.dts:

```
&power_sply {  
    charge_mode = <1>;  
    battery_exist = <1>;  
};  
&charger0 {  
    pmu_safe_vol = <3400>;  
    pmu_safe_ratio = <1>;  
};
```

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。

适用于部分无电持方案或factory\_mode的无电池场景的调试。

如果该属性不进行配置，默认为1。

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>

配置开机方式，根据该属性决定接适配器开机的方式：进入充电页面、需等待按键摁下开机和直接开机。

适用于不需要充电页面，或者适配器唤醒直接开机的需求。

又或者是需要等待按键摁下才能开机的需求。

如果该属性不进行配置，默认为1。

0：不进入充电页面，接适配器开机直接进入开机流程

此情况下在接适配器时关机后会重启

1：接适配器开机后进入关机充电页面

此情况下在接适配器时关机后会进入关机充电页面

2：接适配器开机进入等待状态，摁下按键后直接进入开机流程

此情况下在接适配器时关机后会重新进入等待状态，摁下按键后会直接进入开机流程

pmu\_safe\_vol <u32>

uboot阶段允许开机到内核的最低电压，默认为3.4v

pmu\_safe\_ratio <u32>

uboot阶段允许开机到内核的最低电量，默认为1%

## 6.1.6 GPIO 口耐压值配置

该配置在 uboot 中用于调整 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。

```
device/...../uboot-board.dts:
&gpio_bias { /* Avoid dtc compiling warnings. @TODO: Developer should modify this to the actual value */
    device_type = "gpio_bias";
    pl_bias = <3300>;
    pl_supply = "aldo3_vol";
};
```

```
xx_bias <u32>
    GPIOx口的耐压值设置，单位：mV。

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挂载的输出改变后，会将对应的GPIOx bias耐压值修改过来。
```

## 6.1.7 基础电池属性

基础电池属性包含以下几个方面：

1. 电池满充电压、电池内阻、电池容量
2. 低电警告电量
3. 电池曲线参数

### 6.1.7.1 电池满充电压、电池内阻、电池容量

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_init_chgvol = <4200>;
    pmu_battery_rdc = <147>;
    pmu_battery_cap = <1771>;
    .....
};
```

```
pmu_init_chgvol <u32>
    电池满充电压，默认4.2v
    单位为mV

pmu_battery_rdc <u32>
    电池内阻
    单位为mΩ

pmu_battery_cap <u32>
```

电池容量，默认4000mAh  
单位为mAh

### 6.1.7.2 低电警告电量

低电警告电量有两个等级，一般默认为 15% 和 0%

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_battery_warning_level1 = <15>;
    pmu_battery_warning_level2 = <0>;
    .....
};
```

```
pmu_battery_warning_level1 <u32>
5-20 5% - 20% 警告等级1，默认15%

pmu_battery_warning_level2 <u32>
0-15 0% - 15% 警告等级2，默认0%
```

#### • 电池曲线参数

电池曲线参数分为两种版本，以 AXP803 和 AXP2202 为例：

AXP803：

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_runtime_chgcur = <1000>;
    pmu_suspend_chgcur = <2000>;
    pmu_shutdown_chgcur = <2000>;
    .....
};
```

```
pmu_bat_para1 <u32>
pmu_bat_para2 <u32>
...
pmu_bat_para32 <u32>
    电池曲线参数，输入的是电量相关的信息，可通过仪器或APP测量出来
    **电池参数根据使用的电池不同，更换电池后需要重新测量**
    **当前支持AXP：AXP803**
```

AXP2202：

#### 📖 说明

若要更新该参数，可参考 1 号通上的文档《X-POWERS 电池参数测试系统使用说明\_V1.4.pdf》来进行电池参数测试  
当前仅 AXP2202 使用。

```
device/...../board.dts:
/{
    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>;
};
};

```

param <phandler>

指定电池充电结点，节点名称统一为“axp2101\_parameter”

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\*\*

\*\*当前支持AXP：AXP2202\*\*

### 6.1.7.3 输入限流/限压

限流分为输入限流和电池充电限流，前者决定适配器/pc 能输入到机器的电流上限，后者决定充到电池的电流上限。

限压指的是输入到机器的电压上限。

输入电流 = 电池充电电流 + SOC功耗

以 AXP803 为例：

```

device/...../board.dts:
ac_power_supply: ac-power-supply {
    .....
    pmu_ac_vol = <4600>;
    pmu_ac_cur = <3000>;
    .....
};
usb_power_supply: usb-power-supply {
    .....
    pmu_usbpc_vol = <4600>;
    pmu_usbpc_cur = <500>;
    pmu_usbad_vol = <4000>;
    pmu_usbad_cur = <2500>;
    .....
};

```

```
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

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(vimdpn)
    单位为mV

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

## 6.1.8 充电属性配置

充电相关的内容如下所示：

1. bc1.2 模式配置
2. 电池充电限流
3. 预充电电流限制
4. 截至充电电流

### 6.1.8.1 bc1.2 模式配置

#### 说明

当前仅 axp2202/axp803 支持内核配置。  
仅 axp2202 支持 uboot 配置。

```
device/...../uboot-board.dts:
&power_sply {
    bc12_mode = <0>;
};

device/...../board.dts:(axp803)
battery_power_supply: battery-power-supply {
    .....
    pmu_init_bc_en = <0>;
    .....
};
```

```
device/...../board.dts:(axp2202)
usb_power_supply: usb-power-supply {
    .....
    pmu_bc12_en = <0>;
    .....
};
```

bc12\_mode <u32>  
bc1.2模式开启后可根据充电电源（pc或充电器）进行自动调节充电电流。  
该属性不配置默认为0，仅在uboot阶段生效。  
0：不开启bc1.2模式  
1：开启bc1.2模式

pmu\_init\_bc\_en\pmu\_bc12\_en <u32>  
bc1.2模式开启后可根据充电电源（pc或充电器）进行自动调节充电电流。  
该属性不配置默认为0，仅在内核生效。  
0：不开启bc1.2模式  
1：开启bc1.2模式

### 6.1.8.2 电池充电限流

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_runtime_chgcur = <1000>;
    pmu_suspend_chgcur = <2000>;
    pmu_shutdown_chgcur = <2000>;
    .....
};
```

pmu\_runtime\_chgcur <u32>  
运行时constant充电电流限制，默认300mA(axp803)/500mA(axp2202)  
单位为mA

pmu\_suspend\_chgcur <u32>  
休眠时constant充电电流限制，默认1200mA  
单位为mA

pmu\_shutdown\_chgcur <u32>  
关机时constant充电电流限制，默认1200mA  
单位为mA

### 6.1.8.3 预充电电流限制

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_pre_chg = <100>;
    .....
};
```

```
pmu_pre_chg <u32>
    设置预充电电流限制
    0 - 200 step is 25单位为mA
    **部分型号PMU驱动暂未支持该功能**
```

#### 6.1.8.4 截止充电电流

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_itterm_limit = <100>;
    .....
};
```

```
pmu_itterm_limit <u32>
    设置截止充电电流
    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**
    **当前支持AXP: AXP2101**
```

#### 6.1.9 充电指示灯配置

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_chled_enable = <0>;
    pmu_chgled_func = <0>;
    pmu_chgled_type = <0>;
    .....
};
```

```
pmu_chgled_func <u32>
    CHGLED 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
    **对应指示灯的A、B模式视具体PMU型号而定，可参考对应说明书中《LED指示功能》章节**

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**
```

## 6.1.10 drivevbus 属性配置

### 说明

当前支持 drivevbus 功能的 AXP 型号为：AXP803、AXP2202。

AXP803:

```
device/...../board.dts:
pmu0: pmu@0{
    .....
    x-powers,drive-vbus-en;
    .....
    regulator0: regulators@0 {
        .....
        reg_drivevbus: drivevbus {
            regulator-name = "axp803-drivevbus";
            regulator-enable-ramp-delay = <1000>;
        };
    };
    .....
};
```

AXP2202:

```
device/...../board.dts:
pmu0: pmu@0{
    .....
    x-powers,drive-vbus-en;
    .....
    regulator0: regulators@0 {
        .....
        reg_vmid: vmid {
            regulator-name = "axp2202-vmid";
            regulator-enable-ramp-delay = <1000>;
        };
        reg_drivevbus: drivevbus {
            regulator-name = "axp2202-drivevbus";
            regulator-enable-ramp-delay = <1000>;
            drivevbusin-supply = <&reg_vmid>;
        };
    };
    .....
};
```

usb 部分:

```
&ehci0 {
    drvbus-supply = <&reg_drivevbus>;
};

&ohci0 {
    drvbus-supply = <&reg_drivevbus>;
};
```



};

reg\_vmid: vmid  
仅开启boost供电，仅AXP2202-D有实际效果

reg\_drivevbus: drivevbus  
AXP2202-C:同时操作boost供电和rbfet开关  
AXP2202-D：仅操作rbfet开关  
其他AXP:控制drivevbus输出pin脚的模式

### 6.1.11 NTC 温控属性配置

NTC 温控相关属性如下所示：

1. ntc、jeita 使能、jeita 限流参数
2. 电池温度参数、ts 电流配置
3. 开机温度限制 (uboot)
4. 触发限流、停充、关机的电压阈值

#### 说明

当前支持 ntc 功能的 AXP 型号为：AXP803、AXP2202。

当前支持 jeita 功能的 AXP 型号为：AXP2202。

#### 说明

需内核过温关机功能，则必须配置过温关机的唤醒源，相关配置归类于唤醒源配置中

#### 6.1.11.1 ntc、jeita 使能、jeita 限流参数

```
device/...../uboot-board.dts:
&power_sply {
    ntc_status = <1>;
};

device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_bat_temp_enable = <0>;
    pmu_jetia_en    = <0>;
    pmu_jcool_ifall = <1>;
    pmu_jwarm_ifall = <1>;
    .....
};
```

ntc\_status <u32>

配置ts引脚电流开关，根据该属性决定是否开启，该属性不配置默认为0，仅在uboot生效。

如果不使用ntc功能，则配置为0。

如果使用ntc功能，则必须配置成1或2。

0：关闭ts引脚的电，ntc功能失效

1：开启ts引脚的电，ntc功能启用,在过温/欠温时直接关机。

2: 开启ts引脚的电, ntc功能启用, 在过温/欠温时进入等待状态, 如果没接适配器则关机

pmu\_bat\_temp\_enable <u32>

设置电池温度检测、ntc是否使能, 该设置必须与ntc\_status保持一致。

该属性不配置默认为0, 仅在内核生效。

0: disable

1: enable

pmu\_jetia\_en <u32>

设置jeita功能是否使能, 需配置pmu\_bat\_temp\_enable为1才生效。

该属性不配置默认为0, 仅在内核生效。

0: disable

1: enable

pmu\_jcool\_ifall <u32>

低温电流限流降, 需配置pmu\_jetia\_en为1才生效

该属性不配置默认为50%, 仅在内核生效。

00: 100%

01: 50%, 电流降低到一半

10: 75%, 电流降低到75%

00: 0%

pmu\_jwarm\_ifall <u32>

高温电流限流降, 需配置pmu\_jetia\_en为1才生效

该属性不配置默认为50%, 仅在内核生效。

00: 100%

01: 50%, 电流降低到一半

10: 75%, 电流降低到75%

00: 0%

### 6.1.11.2 电池温度参数、ts 电流配置

#### 说明

由于 TS pin 电压寄存器存在上下限, 因而电池温度参数尽量配置在一定范围内。

AXP803: 80 ~ 3,276mV。

AXP2202/AXP2101: 0 ~ 4,095 mV。

device/...../board.dts:

bat\_power\_supply: bat-power-supply {

.....

pmu\_bat\_ts\_current = <40>;

.....

pmu\_bat\_temp\_para1 = <2814>;

pmu\_bat\_temp\_para2 = <2202>;

pmu\_bat\_temp\_para3 = <1737>;

pmu\_bat\_temp\_para4 = <1381>;

pmu\_bat\_temp\_para5 = <1105>;

pmu\_bat\_temp\_para6 = <890>;

pmu\_bat\_temp\_para7 = <722>;

pmu\_bat\_temp\_para8 = <484>;

pmu\_bat\_temp\_para9 = <332>;

pmu\_bat\_temp\_para10 = <233>;

pmu\_bat\_temp\_para11 = <196>;

pmu\_bat\_temp\_para12 = <166>;

pmu\_bat\_temp\_para13 = <141>;

pmu\_bat\_temp\_para14 = <121>;

```
pmu_bat_temp_para15 = <89>;
pmu_bat_temp_para16 = <66>;
.....
};

device/...../uboot-board.dts:
&charger0 {
    ntc_cur = <40>;
    pmu_bat_temp_para1 = <2814>;
    .....
    pmu_bat_temp_para16 = <66>;
};
```

pmu\_bat\_ts\_current <u32>  
TS pin的电流大小配置，电流大小跟随AXP  
AXP803：20uA/40uA/60uA/80uA 四档可配，默认80uA  
AXP2202/AXP2101：20uA/40uA/50uA/60uA 四档可配，默认50uA  
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pmu\_bat\_ts\_current中的值得到的电压数值（单位：mV），将电压数值填进

pmu\_bat\_temp\_para[1-16]的节点中即可\*\*

ntc\_cur <u32>

配置ts引脚电流，默认为0，该配置必须跟内核dts的一致

pmu\_bat\_temp\_para[16] <u32>

配置16个ntc参数，默认为0，该配置必须跟内核dts的一致

### 6.1.11.3 开关机温度限制（uboot）

#### 说明

uboot 阶段中，目前仅 AXP2202 支持了温控配置。

开关机温度限制即只允许一定温度范围内的机器开机：

```
device/...../uboot-board.dts:
```

```
&charger0 {
```

```
    safe_temp_H = <600>;
```

```
    safe_temp_L = <0xFFFFFCE>;
```

```
};
```

```
safe_temp_H\safe_temp_L <u32>
```

配置uboot阶段允许开机的温度范围，该范围为小于safe\_temp\_H，大于safe\_temp\_L

该属性不配置默认为0，仅在uboot生效。

注：此处填入的值为：温度值 \* 10，负数需要转化成32进制负数

例：在5°~60°才能开机

```
safe_temp_H = <600>;
```

```
safe_temp_L = <0xFFFFFCE>;
```

### 6.1.11.4 触发限流、停充、关机的电压阈值

#### 说明

由于寄存器存在上下限，因此 TS pin 电压阈值必须配置在一定范围内。

AXP803 可配范围：0 ~ 3624mV。

AXP2202 可配范围：

停充/关机过温：0 ~ 8160mV。

停充/关机欠温：0 ~ 510mV。

限流过温：0-4080mV。

限流欠温：0-2040mV。

#### 说明

对于限流、停充、关机的阈值

过温情况下：关机 <= 停充 <= 限流欠温情况下：关机 >= 停充 >= 限流

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....

    pmu_bat_charge_ltf = <1105>;
    pmu_bat_charge_htf = <141>;
    pmu_bat_shutdown_ltf = <1381>;
    pmu_bat_shutdown_htf = <89>;
    pmu_jetia_cool = <722>;
    pmu_jetia_warm = <196>;
    .....
};
```

pmu\_bat\_charge\_ltf <u32>  
触发电池低温停充的TS pin电压阈值，单位：mV  
默认：1312mV  
范围：0-8160mV

pmu\_bat\_charge\_htf <u32>  
触发电池高温停充的TS pin电压阈值，单位：mV  
默认：176mV  
范围：0-510mV

pmu\_bat\_shutdown\_ltf <u32>  
非充电模式下，触发电池低温中断的TS pin电压阈值，单位：mV  
默认：1984mV  
范围：0-8160mV

pmu\_bat\_shutdown\_htf <u32>  
非充电模式下，触发电池高温中断的TS pin电压阈值，单位：mV  
默认：152mV  
范围：0-510mV

pmu\_jetia\_cool <u32>  
触发电池低温限流/限压的TS pin电压阈值，单位：mV  
默认：880mV(10°C)  
范围：0-4080mV

pmu\_jetia\_warm <u32>  
触发电池高温限流/限压的TS pin电压阈值，单位：mV  
默认：240mV(45°C)  
范围：0-2040mV

## 6.1.12 type-c 属性配置

### 📖 说明

当前支持 type-c 功能的 AXP 型号为：AXP2202。  
且同时需要配置 pmic 和 usb 的节点。

```
device/...../board.dts:
usb_power_supply: usb_power_supply {
    .....

    pmu_usb_typec_used = <1>;
    .....
};
usb0:usb0@0 {
    .....
```

```
usb_detect_type = <0x2>;
.....
};
```

pmu\_usb\_typec\_used <bool>

usb接口为type-c

1:使用type-c接口

0:不使用type-c接口

usb\_detect\_type <u32>

设置usb的扫描模式

2:使用type-c接口

1:不使用type-c接口

## 6.1.13 ACIN 属性配置

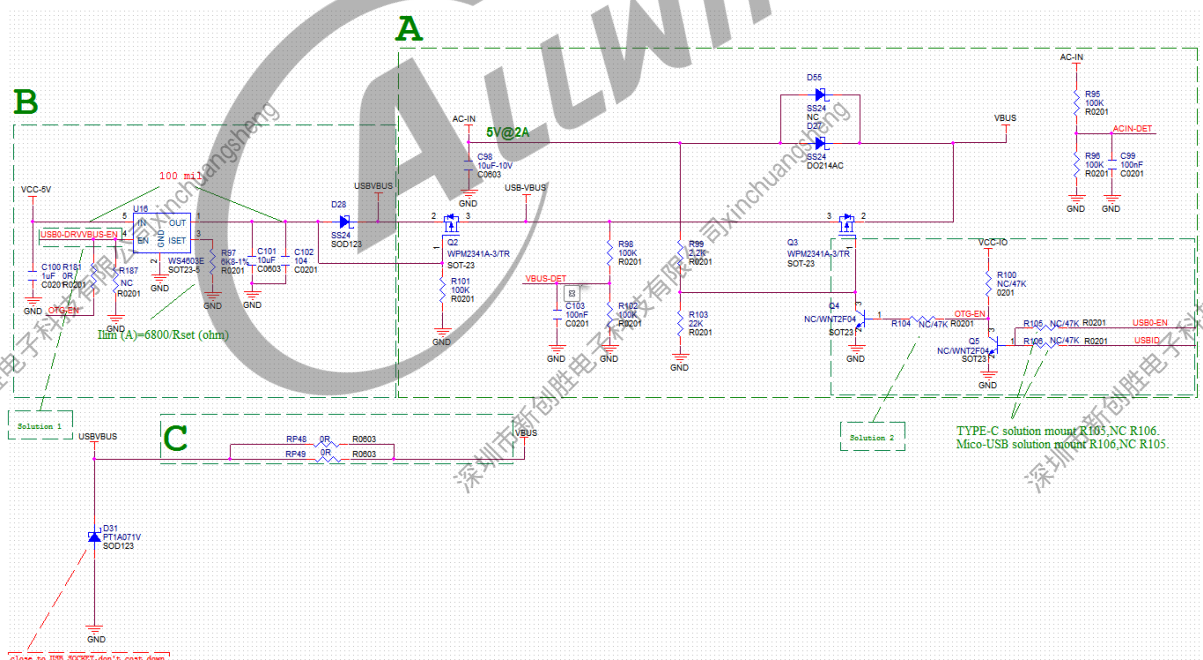


说明

当前支持 type-c 功能的 AXP 型号为：AXP2202。

且同时硬件要满足能使用 acin 的要求。

### 6.1.13.1 方案原理图



2. 当客户方案确认有 ACIN 需求，A、B 两个模块包含元件需贴片，C 模块 NC。

对于 OTG 的供电开关的使能方案，有 solution1（软件 GPIO 控制）和 solution2（纯硬件电平控制）。

1. 若 IO 数量充裕，可选 solution1。
2. 若 IO 数量紧张，建议采用 solution2。

具体方案由客户实际情况而定，对于驱动配置而言，默认按 solution1 配置。

### 6.1.13.2 menuconfig 配置

```
Allwinner BSP ---> Device Drivers --->  
PMIC Drivers ----> <*> AXP2202 Power Virtual ACIN
```

### 6.1.13.3 Device Tree 配置

```
device/...../board.dts:  
usb_power_supply: usb_power_supply {  
    .....  
    det_acin_supply = <&gpio_power_supply>;  
    pmu_acin_usbid_drv = <&pio PH 12 GPIO_ACTIVE_LOW>;  
    pmu_vbus_det_gpio = <&pio PH 13 GPIO_ACTIVE_LOW>;  
    .....  
};  
  
gpio_power_supply: gpio_power_supply {  
    compatible = "x-powers,gpio-supply";  
    status = "disabled";  
    pmu_acin_det_gpio = <&pio PH 14 GPIO_ACTIVE_LOW>;  
    det_usb_supply = <&usb_power_supply>;  
};  
  
&usb0 {  
    usb_drv_vbus_gpio = <&pio PH 7 GPIO_ACTIVE_HIGH>;  
    enable-active-high;  
};  
  
&ehci0 {  
    drvvbus-supply = <&reg_vmid>;  
};  
  
&ohci0 {  
    drvvbus-supply = <&reg_vmid>;  
};
```

```
det_acin_supply = <&gpio_power_supply>;  
引用acin配置，用于检测ac接入状态。  
  
pmu_acin_usbid_drv
```



当usb-id不存在时，用于充当usb-id的gpio口。  
仅在type-c使用的情况下有效。

pmu\_vbus\_det\_gpio

检测vbus接入的gpio口，用于判断usb接入。

status <args>

配置是否加载acin这个驱动，默认为disabled。

pmu\_acin\_det\_gpio

检测ac接入的gpio口，用于判断ac接入。

det\_usb\_supply = <usb\_power\_supply>;

引用usb配置，用于检测usb接入状态，同时进行限流处理。

usb\_drv\_vbus\_gpio = <&pio PH 7 GPIO\_ACTIVE\_HIGH>;

enable-active-high;

需要在usbc0下多加上一个gpio口配置。

用于检测到otg存在时打开drivvbus-en的gpio口。

drivvbus-supply = <&reg\_vmid>;

需要将ehci0和ohci0下引用的reg\_drivevbus修改成reg\_vmid。

## 6.1.14 外挂 DCDC 配置



说明

当前支持外挂 DCDC 功能的平台为：A133、A523。

### 6.1.14.1 menuconfig 配置

menuconfig 配置所有外挂芯片都一致。

```
Allwinner BSP ---> Device Drivers --->  
PMIC Drivers ----> <*> PMU_EXT Power regulator
```

### 6.1.14.2 Device Tree 配置

以外挂芯片 TCS4838 为例。

```
device/...../uboot-board.dts:  
&power_sply {  
    .....  
    tcs_dcdc0_mode = <0>;  
    tcs_dcdc1_mode = <0>;  
    .....  
};
```

```
device/...../board.dts:  
twi6:s_twi@0x07081400{  
    .....  
};
```



```
tcs0: tcs@41 {
    compatible = "ti,tcs4838";
    reg = <0x41>;
    status = "okay";
    tcs4838_delay = <0>;
    regulator1: regulators@1 {
        reg_tcs0: dcdc0 {
            regulator-name = "tcs4838-dcdc0";
            regulator-min-microvolt = <712500>;
            regulator-max-microvolt = <1500000>;
            regulator-always-on;
            regulator-boot-on;
        };
        reg_tcs1: dcdc1 {
            regulator-name = "tcs4838-dcdc1";
            regulator-min-microvolt = <712500>;
            regulator-max-microvolt = <1500000>;
        };
    };
};
.....
};
```

tcs\_dcdc\_mode <u32>

uboot阶段强制将dcdc设置为fpwm开关模式，提高这路抗负载扰动能力，但同时会增大功耗。

该属性不配置默认为0，仅在uboot生效。

0： pfm-pwm模式自由切换

1： 强制pwm模式

regulator基础配置与AXP一致

tcs4838\_delay <u32>

修改外挂芯片的调压速率。

该属性不配置默认为0，仅在内核生效。

0:10mv/0.15us

1:10mv/0.3us

2:10mv/0.6us

3:10mv/1.2us

4:10mv/2.4us

5:10mv/4.8us

6:10mv/9.6us

7:10mv/19.2us

\*\*当前支持外挂芯片型号：TCS4838、SY8827G\*\*

表 6-1: 外挂芯片型号、节点简称和基地址对照表

外挂芯片型号	节点简称	基地址
TCS4838	tcs	0x41
SY8827G	sy	0x60
AXP323	axp1530	0x36

#### 说明

外挂芯片需要用到独立的驱动和独立的配置，这里的 axp323 不使用 axp1530 的驱动。

## 6.1.15 其他配置

### 说明

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，为无效参数。

```
device/...../board.dts:
bat_power_supply: bat-power-supply {
    .....
    pmu_chg_ic_temp = <0>;
    pmu_bat_det = <0>;
    .....
};

&power_sply {
    .....
    axp_bus_num = <0>;
    .....
};
```

```
pmu_chg_ic_temp <u32>
1: 可读PMIC芯片温度。
0: 不可读PMIC芯片温度。

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**
**当前支持AXP: AXP2101**

axp_bus_num <u32>
为pmu所引用的twi的编号。
**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**
**当前支持AXP: AXP221s**
```

## 6.2 常见问题

### 6.2.1 内核阶段-电池图标显示异常

#### 问题现象：

电池图标无法正常显示

#### 问题分析：

电池状态的获取函数在driver/power/supply目录下的xxx-battery.c中定义。以axp2202为例，系统需要获取电池状态时会调用axp2202\_bat\_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 axp2202_bat_get_property(struct power_supply *psy,
2     enum power_supply_property psp,
3     union power_supply_propval *val)
4
5     ...
6     switch (psp) {
7     ...
8     case POWER_SUPPLY_PROP_CAPACITY_LEVEL: // customer modify
9         ret = axp2202_get_bat_present(psy, val);
10        if (ret < 0)
11            return ret;
12
13        if (val->intval) {
14            ret = axp2202_get_soc(psy, val);
15            if (ret < 0)
16                return ret;
17
18            if (val->intval == 100)
19                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_FULL;
20            else if (val->intval > 80)
21                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_HIGH;
22            else if (val->intval > bat_power->dts_info.pmu_battery_warning_level1)
23                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_NORMAL;
24            else if (val->intval > bat_power->dts_info.pmu_battery_warning_level2)
25                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_LOW;
26            else if (val->intval >= 0)
27                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_CRITICAL;
28            else
29                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_UNKNOWN;
30        }
31        else
32            val->intval = POWER_SUPPLY_CAPACITY_LEVEL_UNKNOWN;
33        break;
34    case POWER_SUPPLY_PROP_STATUS:
35        ret = axp2202_get_bat_status(psy, val);
36        if (ret < 0)
37            return ret;
38        break;
39    case POWER_SUPPLY_PROP_PRESENT:
40        ret = axp2202_get_bat_present(psy, val);
41        if (ret < 0)
42            return ret;
43        break;
44    ...
45 }
```

### 问题原因：

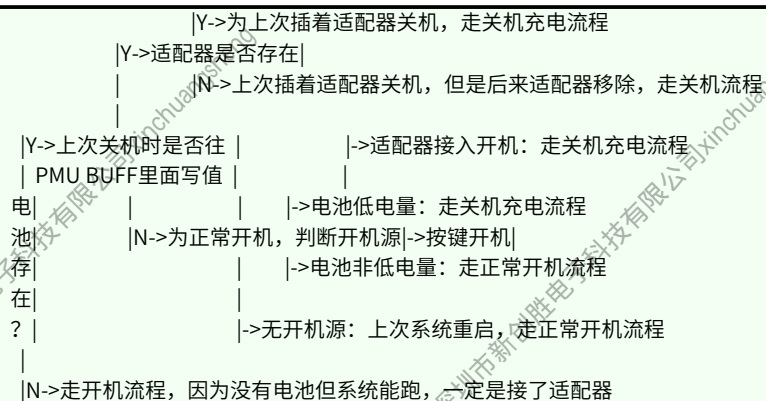
造成电池图标显示异常的原因一般为 PMU 设计缺陷导致驱动读出的电池有效状态错误，电池状态返回 0 导致。

### 问题现象：

无法正常开机，开机卡在 uboot 阶段。

### 问题分析：

目前 sunxi 平台在 boot 阶段的开机流程如下:



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 charge-mode 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：

```

1 int sunxi_update_axp_info(void)
2 {
3     ...
4     if ((val == -1) && (pmu_get_sys_mode() == SUNXI_CHARGING_FLAG)) {
5         val = AXP_BOOT_SOURCE_CHARGER;
6         pmu_set_sys_mode(0);
7     }
8     switch (val) {
9     case AXP_BOOT_SOURCE_BUTTON:
10         strncpy(bootreason, "button", sizeof("button"));
11         break;
12     case AXP_BOOT_SOURCE_IRQ_LOW:
13         strncpy(bootreason, "irq", sizeof("irq"));
14         break;
15     case AXP_BOOT_SOURCE_VBUS_USB:
16         strncpy(bootreason, "usb", sizeof("usb"));
17         break;
18     case AXP_BOOT_SOURCE_CHARGER:
19         strncpy(bootreason, "charger", sizeof("charger"));
20         gd->chargemode = 1;
21         break;
22     case AXP_BOOT_SOURCE_BATTERY:
23         strncpy(bootreason, "battery", sizeof("battery"));
24         break;
25     default:
26         strncpy(bootreason, "unknow", sizeof("unknow"));
27         break;

```

```
28 }  
29 env_set("bootreason", bootreason);  
30 return 0;  
31 ...  
32 }
```

### 问题排查步骤：

顺着 uboot 开机流程进行加打印调试，定位问题地点。




## 著作权声明

版权所有 © 2023 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

## 商标声明

、 **全志科技** （不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标、产品名称，和服务名称，均由其各自所有人拥有。

## 免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。