Linux PMIC
开发指南

版本号: 4.2
发布日期: 2025.06.30

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。
3.2	2024.08.07	AWA1691	新增 pmu-charging-poweroff 节点说明。
3.3	2024.09.11	AWA2152	T536/MR536 系列适用
3.4	2024.11.21	AWA1691	1. 新增 A733 的相关内容。 2. 修改部分描述。
4.0	2025.03.18	AWA1691	1. 更新源码结构变化和版本管理。 2. 更新 dts 属性配置方式。
4.1	2025.06.24	AWA1691	1. 对文档描述的内容进行优化。2. 将快充开发相关内容整合到本文档中。3. 更新输入管理核心的 dts 属性配置方式。4. 删除快充开发指南，快充相关内容整合到本文档中。
4.2	2025.06.30	AWA2152	1.T536/MR536 系列适用

目　录

1 前言 **1**

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

2 模块介绍 **4**

2.1 模块功能介绍 . . . 4
2.1.1 核心功能模块 . . . 5
2.1.1.1 PMU . . . 5
2.1.1.2 BMU . . . 5
2.1.1.3 PMU-ext . . . 6
2.1.1.4 BMU-ext . . . 7
2.1.2 扩展功能模块 . . . 7
2.1.2.1 电源按键管理 . . . 7
2.1.2.2 温度控制 . . . 7
2.1.2.3 电源通知管理核心 . . . 7
2.1.2.4 Type-C 控制 . . . 7
2.1.2.5 PMIC GPIO 引脚管理 . . . 8
2.1.2.6 PMIC WDT 管理 . . . 8
2.2 模块配置介绍 . . . 8
2.2.1 sys_config.fex 配置说明 . . . 8
2.2.1.1 关键配置项 . . . 8
2.2.1.2 配置示例 . . . 9
2.2.2 Device Tree 配置说明 . . . 9
2.2.2.1 uboot . . . 9
2.2.2.2 kernel . . . 10
2.2.3 kernel menuconfig 配置说明 . . . 17
2.2.3.1 配置步骤 . . . 17
2.2.3.2 必要配置 . . . 17
2.2.3.3 基础配置 . . . 18
2.2.3.4 可选配置 . . . 19
2.2.4 Android 配置说明 . . . 20
2.2.4.1 ko 配置 . . . 20
2.2.4.2 selinux 权限配置 . . . 20
2.3 驱动框架介绍 . . . 21
2.3.1 框架特点 . . . 21
2.3.2 功能模块对应关系 . . . 23

2.3.3 硬件原理图 . . . 24
2.4 源码结构介绍 . . . 24
2.4.1 boot0 . . . 25
2.4.2 uboot . . . 25
2.4.2.1 v0.0.x . . . 26
2.4.2.2 v1.0.x 及之后 . . . 26
2.4.3 kernel . . . 27
2.4.3.1 v0.0.x . . . 27
2.4.3.2 v1.0.10 之前 . . . 28
2.4.3.3 v1.0.10 及之后 . . . 29

3 模块接口说明 30
3.1 uboot 调压接口 . . . 30
3.1.1 sunxi_get_voltage_by_phandle . . . 30
3.1.2 sunxi_get_voltage_by_full_name . . . 30
3.1.3 sunxi_set_voltage_by_phandle . . . 31
3.1.4 sunxi_set_voltage_by_full_name . . . 31

4 模块使用范例 32
4.1 regulator 使用 demo . . . 32
4.2 gpio 使用 demo . . . 32
4.3 power_supply 使用 demo . . . 32
4.4 watchdog 使用 demo . . . 33

5 调试方法 35
5.1 调试工具 . . . 35
5.1.1 调试 power key . . . 35
5.2 调试节点 . . . 35
5.2.1 设置 regulator 电压 . . . 35
5.2.2 获取 regulator 引用设备 . . . 36
5.2.3 查询 regulator 状态 . . . 36
5.2.4 寄存器读写操作 . . . 38
5.2.4.1 **标准 regmap 方式：** . . . 38
5.2.4.2 **AXP 专用节点：** . . . 39
5.2.5 查看供电状态 . . . 40
5.2.5.1 battery . . . 41
5.2.5.2 usb . . . 43
5.2.5.3 多口充电 . . . 43
5.2.5.4 TYPE-c . . . 44
5.2.6 其他 AXP 调试节点 . . . 45
5.2.6.1 debug_mask 节点 . . . 45

6 FAQ 47
6.1 常见功能配置 . . . 47

6.1.1 独立 PMIC 通用配置 . . . 48
6.1.1.1 uboot 属性配置 . . . 48
6.1.1.2 无电池方案属性 . . . 51
6.1.1.3 唤醒源配置 . . . 52
6.1.1.4 开关机配置 . . . 53
6.1.1.5 重启配置 . . . 54
6.1.1.6 powerkey 自定义属性 . . . 55
6.1.1.7 输入限流限压配置 . . . 56
6.1.1.8 基础电池属性 . . . 57
6.1.1.9 充电属性配置 . . . 60
6.1.1.10 NTC 温控属性配置 . . . 62
6.1.1.11 充电指示灯配置 . . . 74
6.1.1.12 电源欧标配置 . . . 75
6.1.2 外部驱动关联配置 . . . 76
6.1.2.1 充电温控策略 . . . 76
6.1.2.2 drivevbus 属性配置 . . . 77
6.1.2.3 type-c 功能属性配置 . . . 79
6.1.3 扩展专项配置 . . . 82
6.1.3.1 多口充电属性配置 . . . 82
6.1.3.2 快充系统配置 . . . 87
6.1.3.3 供电扩展配置 . . . 96
6.1.4 平台专用配置 . . . 98
6.1.4.1 gpio 耐压值配置 . . . 98
6.1.4.2 BMU 充电配置 . . . 98
6.1.4.3 其他杂项配置 . . . 99
6.2 常见问题 . . . 101
6.2.1 内核阶段-电池图标显示异常 . . . 101
6.2.2 uboot 阶段-开机流程错误 . . . 102
6.2.3 PMU 异常断电 . . . 103

插　图

图 2-1　PMIC 驱动总体结构 22
图 2-2　PMIC 原理框图 24
图 6-1　普通 NTC 阻值/电压/温度值的对应表 69
图 6-2　AXP519_NTC 温度值对应表 70
图 6-3　软件 NTC 功能流程设计 73
图 6-4　PMIC_ACIN 方案原理图 83
图 6-5　PMIC_ 双 TYPE-C 方案原理图 84
图 6-6　单节快充硬件设计 89
图 6-7　单节快充限流通路框架 90
图 6-8　双节快充硬件设计 90
图 6-9　双节快充限流通路框架 91
图 6-10　AXP2202 关机源 103
图 6-11　AXP8191 关机源 104

表　格

表 1-1 适用产品列表 1
表 1-2 电源管理核心术语 2
表 1-3 电源管理充电术语 2
表 1-4 PMIC 驱动对照表 3
表 2-1 模块功能分类 4
表 2-2 BMU 驱动子类支持情况 6
表 2-3 不同阶段的配置文件 8
表 2-4 Type-C 的支持情况 15
表 2-5 功能模块对应关系 23
表 5-1 regulator 状态说明一览 38
表 5-2 battery 状态动态信息一览 41
表 5-3 battery 状态静态信息一览 42
表 5-4 usb 状态动态信息一览 43
表 5-5 usb 状态静态信息一览 43
表 5-6 多口充电状态动态信息一览 44
表 5-7 多口充电状态静态信息一览 44
表 5-8 TYPE-c 状态动态信息一览 44
表 5-9 TYPE-c 状态静态信息一览 44
表 6-1 PMIC 功能配置分类对照表 47
表 6-2 PMIC 常见功能配置快速索引-PMIC 独立配置 47
表 6-3 PMIC 常见功能配置快速索引-外部驱动关联配置 48
表 6-4 PMIC 常见功能配置快速索引-扩展专项配置 48
表 6-5 PMIC 常见功能配置快速索引-平台专用配置 48
表 6-6 NTC 温控功能支持配置一览 63
表 6-7 jeita 限流参数支持一览 65
表 6-8 TS 引脚电流配置支持一览 66
表 6-9 NTC 温控参数范围 67
表 6-10 NTC 温控电池温度参数对照表 68
表 6-11 温度保护阈值配置范围 70
表 6-12 boost 放电保护阈值配置范围 72
表 6-13 软件 NTC 温控实现方式对比 73
表 6-14 快充方案板型扩展对照表 87
表 6-15 快充方案一览 88
表 6-16 快充方案芯片功能对照表 88
表 6-17 外挂芯片型号、节点简称和基地址对照表 97

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/
MR536	Linux-5.15-origin	bsp/drivers/power/
T536	Linux-5.10-origin	bsp/drivers/power/
A733	Linux-6.6	bsp/drivers/power/
MR153	Linux-5.15-origin	bsp/drivers/power/
T153	Linux-5.10-rt	bsp/drivers/power/

1.4 相关术语介绍

表 1-2: 电源管理核心术语

术语	说明
PMIC	电源管理芯片，常由 PMU 和 BMU 组成，部分芯片可能只包含其中一种功能
AXP	全志 PMIC 系列名称
PMU	电源管理单元，负责系统各模块供电
PMU-ext	扩展电源管理单元，管理外挂供电芯片，用于扩展供电路数
BMU	电池管理单元，负责电池充放电管理，部分方案会分为充电芯片和电量计芯片
BMU-ext	扩展电源管理单元，管理额外电源芯片，用于管理额外充电芯片（常用于快充方案）
TWI	同等于 I2C，PMIC 与主控通讯的接口
Regulator	电源调节器，包含以下子类： 1.DCDC: 开关电源，效率高但纹波较大 2.LDO: 低压差线性稳压器，纹波小但效率较低 3.DRIVEVBUS: 对外的反向 boost，用于给 otg 等外接设备供电，默认输出 0.5A
powerkey	电源按键控制模块
GPIO	通用型输入输出接口，用于电源按键、状态指示等
WDT	看门狗计时器，用于系统异常时复位

表 1-3: 电源管理充电术语

术语	说明
charger 驱动	专指管理输入电源和充电过程的驱动，包含： 1. 输入限流/限压控制 2. 充电曲线管理 (CC/CV 阶段) 3. 充电安全保护 (过压/过流/过温)
battery 驱动	集成充电和电量计功能的驱动，包含： 1. 电池状态监测 (电压/电流/温度) 2. 电量计算 (SOC/SOH) 3. 充放电保护
gauge 驱动	独立电量计芯片驱动，主要功能： 1. 精确电量计算 (库仑计) 2. 电池健康状态评估 3. 电池参数学习
输入电源管理	管理不同输入电源的切换和优先级： 1. 输入源自动切换 (USB/AC/多口充电) 2. 输入过压/反接保护
多口充电	管理多口充电的充电状态，旧式为 acin/gpio_power，新式为 sunxi_multi_charge_power
VBUS	USB 供电电压，从 USB 线输入到 PMIC 的电源

术语	说明
NTC	热敏电阻，用于电池温度监测和保护
JETIA	锂电池充放电温度管理规范
TS	温度传感器电流，用于精确测量电池温度
Type-C	Type-C 功率传输协议控制模块
快充	高于 5V 的充电方案，包含： 1. 单节快充：最高 18W(9V2A) 2. 双节快充：最高 45W(15V3A 或 20V2.25A)

表 1-4: PMIC 驱动对照表

驱动命名	对应芯片型号	芯片类型
AXP22x	AXP221s、AXP223	PMU + BMU
AXP803	AXP803、AXP707	PMU + BMU
AXP2202	AXP2202、AXP717	PMU + BMU + Type-C（简）
AXP515	AXP515	BMU + Type-C（简）
AXP517	AXP517	BMU + Type-C
HUSB311	HUSB311	Type-C
AXP1530	AXP1530、AXP313、AXP323	PMU
AXP8191	AXP8191、AXP318	PMU
ETA6973	ETA6973、ETA6974	CHARGER
AXP519	AXP519	CHARGER
AXP2601	AXP2601	GAUGE
AXP2602	AXP2602	GAUGE

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

电源管理芯片 (PMIC) 是系统电源管理的核心，由电源管理单元 (PMU) 和电池管理单元 (BMU) 两大部分组成。
根据不同的芯片方案，可能还包含电源按键、Type-C 协议控制等扩展模块。
此外，根据实际平台的电源方案类型，在多电源芯片的电源方案中，还会包含扩展电源管理单元 (PMU-ext) 和扩展电池管理单元 (BMU-ext)

表 2-1: 模块功能分类

功能类型	驱动类型	功能描述	支持芯片类型/型号
核心功能			
PMU	regulator	系统供电管理	PMU
BMU	power supply	外部输入和电池充电管理	BMU/GAUGE/CHARGER
扩展功能			
电源按键	power key	按键事件处理	PMU/BMU
温度控制	power temp ctrl	温度监测保护	PMU/BMU/GAUGE
通知机制	notifier	PMIC 驱动间通信	全平台支持
Type-C	typec	Type-C 接口和协议管理	Type-C
GPIO	gpio	引脚状态管理	AXP22x
看门狗	wdt	系统异常复位	/
PMU-ext	regulator	扩展电源管理	PMU
BMU-ext	power supply	扩展输入和充电管理	BMU/GAUGE/CHARGER

说明

BMU 单元的 power supply 子系统下有多个驱动子类。详细支持情况见表《BMU 驱动子类支持情况》。

2.1.1 核心功能模块

2.1.1.1 PMU

- **主要功能**:
- 系统各模块供电管理
- 电源状态监测与保护
- 电源时序控制

- **驱动实现**:
- 通过 regulator 子系统控制各路 DCDC/LDO 输出
- 提供电源状态监测功能

2.1.1.2 BMU

- **主要功能**:
- 电池充放电管理
- 电池状态监测
- 温度/电压/电流保护

- **常见实现方案**:
- 单芯片方案：集成充电和电量计功能
- 双芯片方案：分离为充电芯片 (Charger) 和电量计芯片 (Gauge)

- **驱动分类**:
- 输入电源管理核心类驱动 (Input Power core)
- **功能**：对输入电源统筹管理
- **子类**:
  - sunxi_usb_power: 输入管理核心，统一管理 usb 外部输入的状态
  - sunxi_usb_power_limit: 限流管理核心，统一管理 usb 输入电源的限流值
  - sunxi_power_supply_core: 统一管理定义 power_supply 下的私有化接口
  - sunxi_multi_charge_power: 多口充电管理核心，管理多口充电的充电状态

- 充电管理类驱动 (Charger Drivers)
- **功能**：管理输入电源
- **子类**:
  - usb_power: 获取 USB 输入电源的信息
  - ac_power: 交流适配器输入电源管理
  - acin/gpio_power: 多口充电管理驱动（旧）
  - charger_power: 充电芯片管理驱动，除获取 USB 输入电源的信息外有管理电池充电的能力

- 电池管理类驱动 (Battery Drivers)

- **功能**：电池状态监测，电池充电控制和电量计量
- **子类**：
  - battery：

1. 集成充电和电量计功能（单芯片方案）
2. 独立电量计芯片驱动（双芯片方案），通过 charger 驱动来完成电池充电控制。

- 输入状态识别类驱动 (Input Power detect)
- **功能**：通过其他方式识别端口接入充电
- **子类**：
  - sunxi_ac_virtual_power: 通过 gpio 口识别充电器接入

说明

管理核心是内核 pmic 驱动在 v1.0.10 后才新增的内容，使用时需要注意驱动版本型号。

说明

charger_power 实际常用于 BMU-ext 中。

表 2-2: BMU 驱动子类支持情况

BMU 驱动分类	BMU 驱动子类	支持芯片驱动
管理核心类驱动	sunxi_usb_power	AXP515, AXP517, AXP2202
管理核心类驱动	sunxi_usb_power_limit	AXP515, AXP517, AXP2202
管理核心类驱动	multi_charge_power	AXP515, AXP517
充电管理类驱动	ac_power	AXP22x, AXP803
充电管理类驱动	usb_power	BMU 和 CHARGER 都支持
充电管理类驱动	acin/gpio_power	AXP2202
充电管理类驱动	charger_power	AXP519, ETA6973
电池管理类驱动	battery_power	BMU 和 GAUGE 都支持
输入状态识别类驱动	sunxi_ac_virtual_power	和实现平台硬件相关

说明

acin 的含义与 ac_power 的不同，为旧式多口充电管理驱动。
多口充电管理的详细内容参见《模块功能配置-多口充电属性配置》章节。

2.1.1.3 PMU-ext

- **主要功能**：
- 扩展系统供电路数
- 提供额外电源轨
- 增强供电能力

说明

PMU-ext 的驱动实现和 PMU 的一致。

2.1.1.4 BMU-ext

• **主要功能**:
• 支持更高功率充电
• 实现多节电池管理
• 增强快充能力

说明

BMU-ext 的驱动实现和 BMU 的一致。

2.1.2 扩展功能模块

2.1.2.1 电源按键管理

• **主要功能**:
• 电源按键行为检测
• 长短按键识别
• 系统唤醒/休眠控制

2.1.2.2 温度控制

• **主要功能**:
• 定义通用温控接口
• 使用特定 pmu 的 adc 进行温度采样

说明

notifier 与 power temp ctrl 一般为虚拟驱动。
在当使用特定 pmu 的 adc 进行温度采样时，power temp ctrl 才会有实体驱动。
当前仅 AXP8191 支持 power temp ctrl 的实体驱动。

2.1.2.3 电源通知管理核心

• **主要功能**:
• pmic 驱动内部信息传递

2.1.2.4 Type-C 控制

• **主要功能**:
• Type-C 接口检测 • 功率传输协议处理

2.1.2.5 PMIC GPIO 引脚管理

• **主要功能**:
• 管理 pmic 的 gpio 引脚状态

说明

仅 AXP22x 有相关驱动支持。

2.1.2.6 PMIC WDT 管理

• **主要功能**:
• 管理 pmic 芯片上的看门狗

2.2 模块配置介绍

PMIC 驱动需要以下基础配置才能正常运行:
• 配置生效时间分为三个阶段：boot0、uboot 和 kernel
• 每个阶段使用不同的配置文件
• PMIC 模块在 dtsi 中无用户可用配置，所有配置项几乎都在 device 下

说明

这里只介绍最基础的配置，若想知道具体哪个功能怎样配置，可参考 FAQ 章节内的《常见功能配置》。

表 2-3: 不同阶段的配置文件

生效阶段	配置文件	配置项数量
boot0	sys_config.fex	少
uboot	uboot-board.dts	中
kernel	board.dts	多

2.2.1 sys_config.fex 配置说明

2.2.1.1 关键配置项

- power_mode属性：决定当前平台使用哪个 PMU，需 boot0 代码支持解析

说明

power_mode 属性是否有效与产品平台有关。适用产品平台：A133/T509、A523。

2.2.1.2 配置示例

以在 A133 产品平台上使用 AXP2202 为例，其配置如下：

```
;-----------------------------------------------------------------------------
;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

- 需要配置与内核 regulator 几乎相同的属性
- 增加power_delay作为延时配置（用于 twi 等模块调压后等待电压稳定）

2.2.2.1.1 配置示例

以 twi6 上挂载 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
}
   &twi6代表该pmu挂载在twi6下。

&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

2.2.2.2.1 配置层次

PMIC 在内核中包含多个子系统，存在层次关系：

```
PMU主设备(MFD)
 ├── regulator device
 ├── power key device
 ├── power supply device
 ├── gpio device
 ├── wdt device
 └── power temp ctrl
```

说明

使用 PMIC 必须按顺序配置：
1. 先配置并启用主设备
2. 再配置各子系统
3. 如需关闭特定子系统，必须设置 status = "disabled" 4. 虚拟驱动和管理核心类驱动（除多口充电）不受配置层次限制。

2.2.2.2.2 主设备配置示例

主设备的节点为 pmu0，放置在 PMIC 所使用的 i2c 节点下。
后面其余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
```

2.2.2.2.3 子系统配置示例

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_aldo1: aldo1{
  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

以 AXP2202 为例：

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

```
compatible <char>
    在单pmic方案下所有版本的AXP的匹配项都是"x-powers,axp2101-pek"。
    多pmu方案中，副pmic的匹配项是"x-powers,{axp_name}-pek",其中axp_name为副pmic的名字，且status必定要配置
     成disabled。
```

power supply

说明

注意事项：

- **管理核心是内核 pmic 驱动在 v1.0.10 后才新增的内容。**
- **内核 v1.0.10 及之后的版本，外部驱动不可直接引用特定管理类驱动，只可引用管理核心。**
- **管理核心的配置是直接放在根目录下，固定放在其他管理类之后。**

管理核心驱动

```
/{
  sunxi_usb_power_supply: sunxi_usb_power_supply {
    compatible = "x-powers,sunxi-usb-power-supply";
    status = "okay";
    det_usb_supply = <&usb_power_supply>;
```

```
    };

    sunxi_usb_power_limit: sunxi_usb_power_limit {
        compatible = "x-powers,sunxi-usb-power-limit";
        status = "okay";
        det_battery_supply = <&bat_power_supply>;
        det_usb_supply = <&sunxi_usb_power_supply>;
        det_typc_supply = <&tcpc>;
        det_multi_charge_supply = <&gpio_power_supply>;
        extcon_udc = <&udc>;
        extcon_phy = <&phy_switcher>;
    };
};
```

```
sunxi_usb_power_supply: sunxi_usb_power_supply {
    det_usb_supply = <&usb_power_supply>;
        引用usb_power_supply对应的充电管理类驱动
};

sunxi_usb_power_limit: sunxi_usb_power_limit {
    det_battery_supply = <&bat_power_supply>;
        引用bat_power_supply对应的充电管理类驱动

    det_usb_supply = <&sunxi_usb_power_supply>;
        引用输入管理核心

    det_typc_supply = <&tcpc>;
        引用det_typc_supply对应的Type-C 控制驱动

    det_multi_charge_supply = <&gpio_power_supply>;
        引用gpio_power_supply对应的多口充电管理核心，或多口充电管理驱动（旧）

    extcon_udc = <&udc>;
        引用udc对应的usb_udc驱动

    extcon_phy = <&phy_switcher>;
        引用phy_switcher对应的phy驱动
};
```

其他管理类驱动

以 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>
    匹配不同的电源驱动，其名字按照实际承接BMU的具体功能来命名。

注：在一些AXP（例如AXP2202）中，battery的供电节点是以“bat_power_supply”来命名
```

多口充电驱动/管理核心

说明

注意事项：

- **以下配置以内核电源驱动 v1.0.10 及之后的版本为主。**
- **旧式多口充电驱动和多口充电管理核心的配置略有差异。**

旧式多口充电驱动 (以 AXP2202 为例)：

```
device/……/board.dts:

gpio_power_supply: gpio_power_supply {
  compatible = "x-powers,axp2202-acin-power-supply";
  status = "okay";
};
```

多口充电管理核心 (以 AXP515 为例)：

```
device/……/board.dts:

gpio_power_supply: gpio_power_supply {
  compatible = "x-powers,axp515-multi-charge-supply";
  status = "okay";
};
```

```
compatible <char>
    匹配不同名字的多口充电驱动，其名字按照实际情况来命名。
    旧式多口充电驱动：固定为"x-powers,axp2202-acin-power-supply"
    多口充电管理核心：为"x-powers,axp[xxx]-multi-charge-supply"
            其中axp[xxx]为承接BMU（charge）供电的芯片的驱动名。
```

power temp ctrl

power temp ctrl 为实体驱动时才会需要进行 dts 属性配置。
以 AXP8191 为例：

```
power_temp_ctrl0: power_temp_ctrl@0 {
  compatible = "x-powers,axp8191-temp-ctrl";
  status = "okay";
  det_bat_supply = <&bat_power_supply>;
};
```

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

det_bat_supply
    当temp_ctrl设备用于进行管理电池充电限流时，配置相关的battery驱动。
```

Type-c

说明

注意事项：

- **需要同时配置 PMIC 和 USB 节点。**
- **需要配置 USB 检测模式 2。**

表 2-4: Type-C 的支持情况

芯片驱动	TYPC 功能支持情况
AXP2202	简单 cc 逻辑
AXP515	简单 cc 逻辑
AXP517	type-c 完整功能
HUSB311	type-c 完整功能

简单 cc 逻辑

```
device/……/board.dts:
usb_power_supply: usb_power_supply {
  ……
  pmu_usb_typec_used = <1>;
  ……
};

usbc0:usbc0@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>
    需设置usb的扫描模式为2
    0:none
    1:vbus/id detect，通过id值检测
    2:pmu detect，由pmu检测结果决定
    3:tcpm detect，由typec驱动切换
```

type-c 完整功能

说明

注意事项：

- **有 type-c 完整功能的芯片需要额外配置 typec 驱动。**
- **需要配置 USB 检测模式 3。**
- **确保引用的节点 (如 reg_qc_drivevbus) 已正确定义。**
- **中断引脚需与原理图一致。**
- **在这里只是配置了驱动使能，更详细的配置见章节《type-c 功能属性配置》。**

说明

适用芯片：• AXP517 (集成 Type-C 控制器)

• HUSB311 (独立 Type-C 芯片)

```
&twi5 {
  husb311: husb311@4e {
    compatible = "hynetek,husb311";
    reg = <0x4e>;
    interrupt-parent = <&r_pio>;
```

```
    husb311,intr_gpio = <&r_pio PL 13 GPIO_ACTIVE_LOW>;
    vbus-supply = <&reg_drivevbus>;
    det_usb_supply = <&sunxi_usb_power_supply>;
    /* aw,vbus-wakeup-quirk; */
    /* aw,port-reset-quirk; */
    status = "okay";

    ports {
      #address-cells = <1>;
      #size-cells = <0>;

      port@0 {
        reg = <0>;
        usbc0_role_sw: endpoint@0 {
          remote-endpoint = <&usbc0_role_switch>;
        };
      };
    };

    usb_con: connector {
      compatible = "usb-c-connector";
      label = "USB-C";
      ……
    };
  };
};

&usbc0 {
  usb-role-switch;
  port {
    #address-cells = <1>;
    #size-cells = <0>;
    usbc0_role_switch: endpoint@0 {
      reg = <0>;
      remote-endpoint = <&usbc0_role_sw>;
    };
  };
};

usbc0:usbc0@0 {
  ……
  usb_detect_type = <0x3>;
  ……
};
```

```
husb311: husb311@4e {

compatible <char>
    设备匹配名，对应Type-C PD控制器的型号

reg <u32>
    i2c寄存器地址

interrupt-parent <args>
husb311,intr_gpio <args>
    中断配置，例子中husb311使用PL13作为中断脚

vbus-supply = <&reg_drivevbus>;
    引用boost 5V输出，根据CC状态设置Vbus的开启和关闭
```

```
det_usb_supply = <&sunxi_usb_power_supply>;
    引用充电驱动，通过PowerSupply框架接口回调快充驱动的函数

aw,vbus-wakeup-quirk <bool>
    husb311是否能唤醒soc

aw,port-reset-quirk <bool>
    husb311唤醒时是否重新初始化

ports
    进行usb角色切换所操作的usb端口

connector
    更细节的连接器置，详见章节《type-c功能属性配置》

}

&usbc0 {
  usb端口配置，其名字需要和ports匹配。
};

usbc0:usbc0@0 {
  ……
  usb_detect_type <u32>
    需设置usb的扫描模式为3
    0:none
    1:vbus/id detect，通过id值检测
    2:pmu detect，由pmu检测结果决定
    3:tcpm detect，由typec驱动切换
  ……
};
```

2.2.3 kernel menuconfig 配置说明

2.2.3.1 配置步骤

进入 logan 根目录，执行：

```
./build.sh menuconfig
```

2.2.3.2 必要配置

这些配置是 PMIC 驱动正常运行所必需的：

- I2C 支持

```
Allwinner BSP  --->  Device Drivers  --->
  I2C Drivers  --->  <*> I2C Support for Allwinner SoCs
```

- PMIC 核心驱动

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

2.2.3.3 基础配置

- 电源调节器管理核心

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

- 电源通知管理核心

```
Allwinner BSP  --->  Device Drivers  --->
  PMIC Drivers  --->  {M} Allwinner Power Notifier support
```

- 温度管理核心

```
Allwinner BSP  --->  Device Drivers  --->
  PMIC Drivers  --->  {M} Allwinner Power Temp Ctrl support
```

- 限流管理核心

```
Allwinner BSP  --->  Device Drivers  --->
  PMIC Drivers  --->  {M} Allwinner Usb Power Ctrl support
```

- 输入管理核心

```
Allwinner BSP  --->  Device Drivers  --->
  PMIC Drivers  --->  {M} Allwinner Power Supply Ctrl
```

- 多口充电管理核心

```
Allwinner BSP  --->  Device Drivers  --->
  PMIC Drivers  --->  <M> sunxi multi charge Power Supply Driver
```

说明

部分管理核心驱动的配置会在可选配置选择后自动使能，无需额外配置。

2.2.3.4 可选配置

这些是根据具体需求选择的功能模块：

- 电源管理功能

说明

电源管理驱动名字和具体芯片相关，以下配置以 AXP2202 为例。

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

- Type-C 接口和协议功能

说明

- **有 type-c 完整功能的芯片需要额外配置 typec 驱动。**
- **HUSB311 的配置和 AXP517 的配置不同。**

```
HUSB311：

Allwinner BSP  --->  Device Drivers  --->
  USB Drivers  --->  USB Type-C Support --->
    <*> Allwinner USB Type-C support
    <*> Hynetek HUSB311 Type-C chip driver
```

```
AXP517：

Allwinner BSP  --->  Device Drivers  --->
  USB Drivers  --->  USB Type-C Support --->
    <*> Allwinner Type-C Port Controller Interface driver (Non-Standard)
      <*> Allwinner AXP517 Type-C chip driver
```

- 按键管理功能

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

- 温度管理驱动

```
Allwinner BSP  --->  Device Drivers  --->
  PMIC Drivers  --->  <*> AXP8191 Temp Ctrl Drivers
```

- 多口充电配置（旧）

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

说明

多口充电配置（旧）仅 AXP2202 支持。

2.2.4 Android 配置说明

2.2.4.1 ko 配置

配置原则：1. 核心驱动必须加载 2. 扩展驱动按需加载

配置示例：

以 “主驱动（axp2202）+pmu 扩展驱动 + 温度管理驱动” 为例：

说明

bmu 扩展驱动基本都为快充服务，因此一般放在特定板级下，具体配置详见《快充系统配置》章节。

```
device/softwinner/[平台名]/common/system/vendor_ramdisk.modules:

# 必要核心
axp2101-i2c.ko
axp2101.ko

# 电源通知管理核心
sunxi_power_temp_ctrl.ko

# 温度管理核心
sunxi-power-notifier.ko

# 输入电源管理核心，v1.0.10新增
sunxi_usb_power_limit.ko
sunxi_power_supply_core.ko
sunxi_usb_power.ko

# 按键管理
axp2101-pek.ko

# 电源调节器管理核心
axp2101-regulator.ko
sunxi_virtual_consumer.ko

# 电源管理
axp2202_battery.ko
axp2202_usb_power.ko

# 温度管理
axp8191_temp_ctrl.ko

# 扩展电源管理
pmu-ext-core.ko
pmu-ext-i2c.ko
pmu-ext-regulator.ko
```

2.2.4.2 selinux 权限配置

配置原则：1. 仅 BMU 相关驱动需要特殊权限 2. 按功能分类配置

配置示例：

以 AXP2202 为例：

```
device/softwinner/common/sepolicy/vendor/genfs_contexts：

# 电池信息节点
genfscon sysfs /devices/platform/.../axp2202-bat-power-supply.0 u:object_r:sysfs_batteryinfo:s0

# USB电源节点
genfscon sysfs /devices/platform/.../axp2202-usb-power-supply.0 u:object_r:sysfs_batteryinfo:s0

# 输入电流限制节点
genfscon sysfs /devices/platform/.../axp2202-usb-power-supply.0/power_supply/axp2202-usb/input_current_limit u:
      object_r:battery_input_module:

# 唤醒节点
genfscon sysfs /devices/platform/.../axp2202-bat-power-supply.0/power_supply/axp2202-battery/wakeup u:object_r:
      sysfs_wakeup:s0
genfscon sysfs /devices/platform/.../axp2202-usb-power-supply.0/power_supply/axp2202-usb/wakeup u:object_r:
      sysfs_wakeup:s0
```

说明

1. **路径中的...需替换为实际平台路径**
2. **新平台建议使用device/softwinner/[平台名]/sepolicy下的独立配置**

2.3 驱动框架介绍

全志 PMIC 驱动框架是一个完整的电源管理系统解决方案，支持多种 PMIC 芯片。
该框架采用分层架构设计，提供统一的接口和模块化的功能实现。

2.3.1 框架特点

1. **模块化设计**：将 PMIC 功能划分为核心模块和扩展模块
2. **统一接口**：提供标准化的驱动接口和 API
3. **灵活扩展**：支持多种 PMIC 芯片的接入
4. **完整功能**：覆盖从电源管理到电池充电的全套功能
5. **多阶段支持**：支持 boot0、uboot 和 kernel 全流程

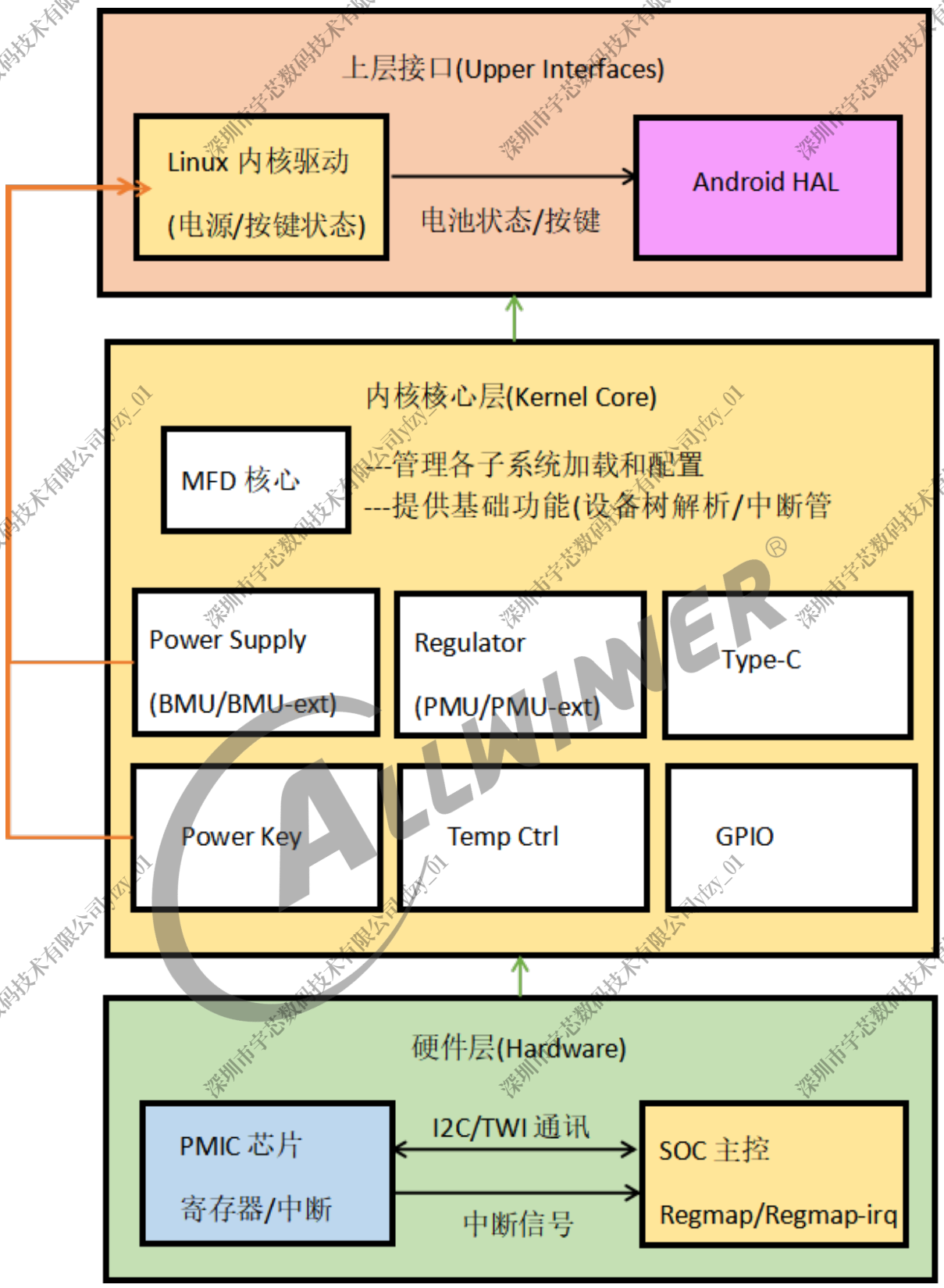

图 2-1: PMIC 驱动总体结构

2.3.2 功能模块对应关系

表 2-5: 功能模块对应关系

功能分类	驱动架构层级	功能描述	典型实现文件
核心功能			
核心	MFD 核心层	芯片基础功能注册和中断管理	axp2101.c
核心	I2C 通讯层	芯片基础功能注册和中断管理	axp2101-i2c.c
PMU	Regulator 驱动层	系统供电管理和电压调节	axp2101-regulator.c
BMU	Power Supply 驱动层	电池充电管理和状态监测	axp2202_battery.c
		外部输入和电池充电管理	axp2202_usb_power.c
		多口充电的充电状态管理(旧)	axp2202_gpio_power.c
		统一管理监测 usb 外部输入的状态	sunxi_usb_power.c
		统一管理 usb 输入限流	sunxi_usb_power_limit.c
		统一管理多口充电的充电状态	sunxi_multi_charge_power.c
扩展功能			
电源按键	Power Key 驱动层	按键事件处理	axp2101-pek.c
温度控制	Temp Ctrl 驱动层	温度监测和保护	axp8191_temp_ctrl.c
通知机制	Notifier 驱动层	PMIC 驱动间通信	sunxi-power-notifier.c
Type-C	Type-C 驱动层	接口检测和协议处理	tcpci_axp517.c
GPIO	GPIO 驱动层	引脚状态管理	pinctrl-axp22x.c
看门狗	WDT 驱动层	系统异常复位	(平台相关实现)
PMU-ext	Regulator 驱动层	扩展电源管理	pmu-ext-regulator.c
BMU-ext	Power Supply 驱动层	扩展输入和充电管理	eta6973_charger_power.c

说明

1. **实际驱动实现可能因芯片型号有所不同**
2. **部分功能模块为可选配置，需根据具体需求启用**
3. **新版本驱动 (v1.0.x) 对架构进行了优化，建议优先使用**

2.3.3 硬件原理图

以 AXP2202 为例，其在原理图如下所示：

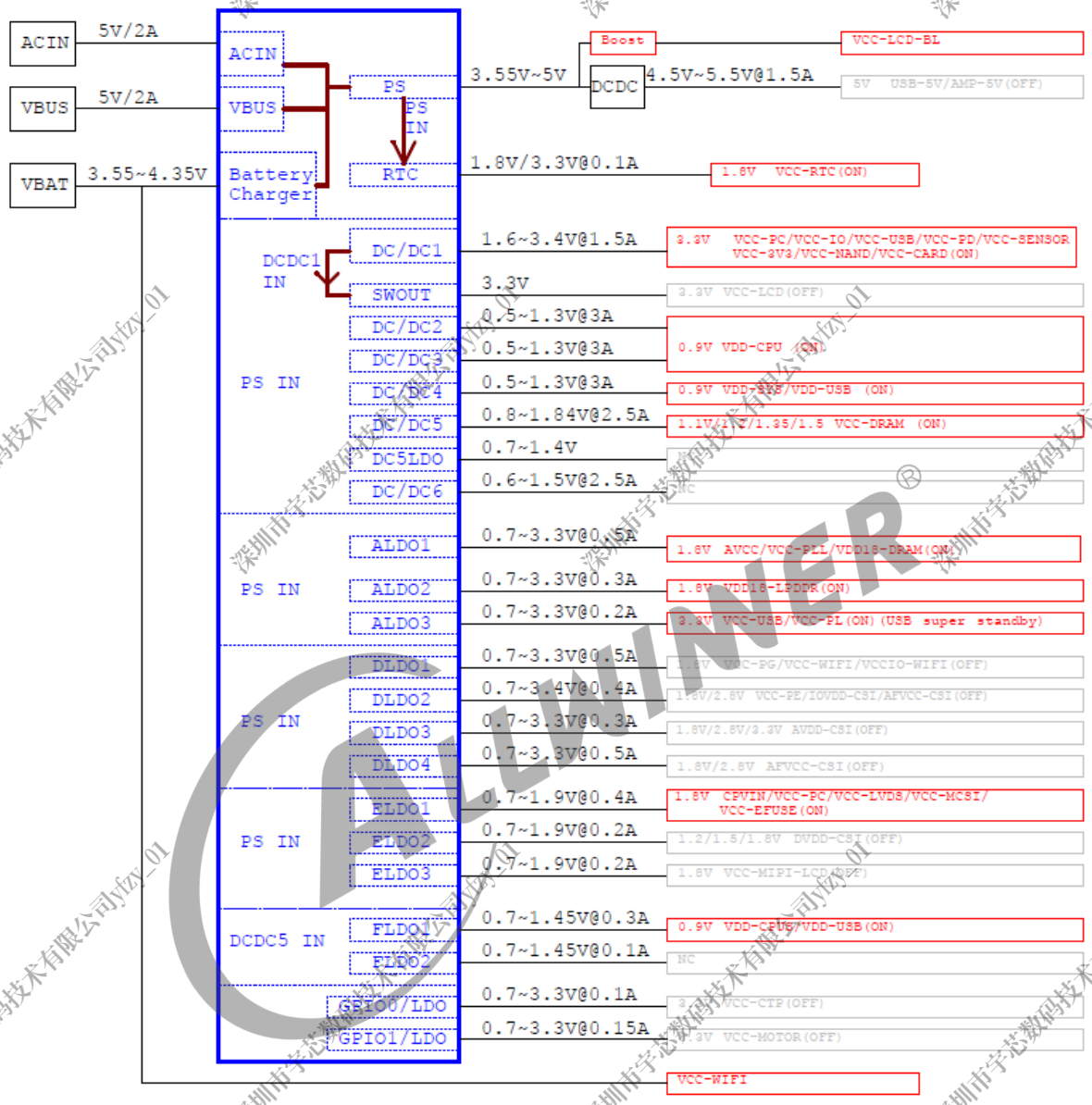

图 2-2: PMIC 原理框图

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 阶段，AXP PMIC 驱动采用模块化设计，将功能划分为两个核心部分：

1. **PMU 模块**：负责
   - 系统电源管理（LDO/DC-DC 调节）
   - 电源按键检测
   - 电压监控
2. **BMU 模块**：负责
   - 电池充放电管理
   - 温度监测
   - 充电状态控制

主要功能特性包括：- 支持多种开机模式检测（正常启动/充电模式）- 提供完整的低电量管理方案 - 精确的电池温度监测机制 - 灵活的 PWM 模式控制

说明

源码结构 v0.0.x 与后面的 v1.0.x 差异较大，使用时注意版本状态。

说明

如何区分 v0.0.x 与 v1.0.x：看是否存在相关版本管理文件 version。
version 路径：longan\brandy\brandy-2.0\u-boot-2018\drivers\sunxi_power\version

其源码结构以 “主驱动（axp2202）+pmu 扩展驱动（tcs4838）+bmu 扩展驱动（eta6973）” 为例：

2.4.2.1 v0.0.x

```
brandy/brandy-2.0/u-boot-2018
|-- board/sunxi/power_manage.c
|-- drivers/sunxi_power/
   |-- axp.c
   |-- bmu.c
   |-- pmu.c
   |-- pmu_ext.c
   |-- bmu_ext.c
   |-- bmu_axp2202.c
   |-- bmu_axp2202.c
   |-- pmu_tcs4838.c
   |-- bmu_eta6973.c
|-- include/sunxi_power/
   |-- axp.h
   |-- pmu_ext.h
   |-- bmu_ext.h
   |-- power_manage.h
   |-- pmu_axp2202.h
   |-- bmu_axp2202.h
   |-- pmu_tcs4838.h
   |-- bmu_eta6973.h
...
```

2.4.2.2 v1.0.x 及之后

```
brandy/brandy-2.0/u-boot-2018
|-- drivers/sunxi_power/
   |-- version
   |-- core/
      |-- axp.c
      |-- pmu.c
      |-- bmu.c
      |-- pmu_ext.c
      |-- bmu_ext.c
      |-- power_manage.c
      |-- pmu_general.c
   |-- pmu/
      |-- pmu_axp2202.c
      |-- pmu_axp1530_ext.c
   |-- bmu/
      |-- bmu_axp2202.c
      |-- bmu_eta6973.c
   |-- include/
      |-- axp.h
      |-- pmu_ext.h
      |-- bmu_ext.h
      |-- power_manage.h
      |-- pmu_axp2202.h
      |-- bmu_axp2202.h
      |-- pmu_tcs4838.h
      |-- bmu_eta6973.h
|-- include/sunxi_power/
   |-- sunxi_power.h
```

```
...
```

对比 v0.0.x 版本，v1.0.x 新增了：

1. sunxi_power.h 头文件专门处理对外接口
2. pmu_general.c 专门用于多 pmu 调压场景。

2.4.3 kernel

内核驱动采用分层架构设计，主要特点包括：

1. **核心功能**：

- 统一的电源管理接口
- 标准化的驱动框架
- 模块化的功能实现

2. **扩展功能**：

- Type-C 协议支持
- 温度控制子系统
- 驱动间通信机制

说明

power 部分的源码结构 v0.0.x 与后面的 v1.0.x 差异较大，使用时注意版本状态。

说明

如何区分 v0.0.x 与 v1.0.x：看是否存在相关版本管理文件 version。
version 路径：longan\bsp\drivers\power\version

其中，Type-C 协议相关的驱动源码结构几乎不变，如下所示：

```
bsp/drivers/usb/typec/tcpm/
  |-- tcpci_husb311.c
  |-- allwinner/
    |-- tcpci.c
    |-- tcpci.h
    |-- tcpci_axp517.c
```

其他部分的源码结构以 axp2202 为例：

2.4.3.1 v0.0.x

```
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
|-- include/power
   |-- axp2101.h
...
```

2.4.3.2 v1.0.10 之前

```
bsp/drivers/power
|-- version
|-- include/
   |-- axp2101.h
   |-- sunxi-power-core.h
   |-- sunxi-power-mfd.h
   |-- sunxi-power-regulator.h
   |-- sunxi-power-powerkey.h
   |-- sunxi-power-supply.h
   |-- sunxi-power-notifier.h
   |-- sunxi-power-temp-ctrl.h
|-- 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
|-- temp_ctrl/
   |-- sunxi_power_temp_ctrl.c
|-- notifier/
   |-- sunxi-power-notifier.c
...
```

对比 v0.0.x 版本，v1.0.x 新增了：

1. include 目录以及相关按功能区分的头文件
2. sunxi-power-temp-ctrl.c 专门用于处理 power 相关温控功能。
3. sunxi-power-notifier.c 专门用于管理 power 驱动之间的信息传递。

2.4.3.3 v1.0.10 及之后

```
bsp/drivers/power
|-- version
|-- include/
   |-- axp2101.h
   |-- sunxi-power-core.h
   |-- sunxi-power-mfd.h
   |-- sunxi-power-regulator.h
   |-- sunxi-power-powerkey.h
   |-- sunxi-power-supply.h
   |-- sunxi-power-notifier.h
   |-- sunxi-power-temp-ctrl.h
|-- mfd/
   |-- axp2101.c
   |-- axp2101-i2c.c
|-- supply/
   |-- axp2202_usb_power.c
   |-- axp2202_battery.c
   |-- axp2202_gpio_power.c
   |-- sunxi_usb_power.c
   |-- sunxi_usb_power_limit.c
   |-- sunxi_power_supply_core.c
   |-- sunxi_multi_charge_power.c
|-- regulator/
   |-- axp2101-regulator.c
|-- power_key/
   |-- axp2101-pek.c
|-- temp_ctrl/
   |-- sunxi_power_temp_ctrl.c
|-- notifier/
   |-- sunxi-power-notifier.c
...
```

对比 v1.0.x 版本，v1.0.10 新增了：

1. 输入电源管理核心类驱动
2. 更改了限流逻辑，因此务必检查 device tree 的配置是否有同步更新。

3 模块接口说明

3.1 uboot 调压接口

当前仅 v1.0.x 后版本支持，注意相关 version 信息

3.1.1 sunxi_get_voltage_by_phandle

- 函数原型：int sunxi_get_voltage_by_phandle(const void *fdt, uint32_t phandle)
- 作用：根据引用供电的句柄，获取对应供电电路的电压值。
- 参数：
  - fdt：从外部传入的 work fdt。
  - phandle：指向引用供电的句柄。
- 返回：
  - 成功，返回电压值。
  - 失败，返回-1。

3.1.2 sunxi_get_voltage_by_full_name

- 函数原型：int sunxi_get_voltage_by_full_name(char *name)
- 作用：根据引用供电的名字，获取对应供电电路的电压值。
- 参数：
  - name：供电的名字。
  - phandle：指向引用供电的句柄。
- 返回：
  - 成功，返回电压值。
  - 失败，返回-1。

3.1.3 sunxi_set_voltage_by_phandle

- 函数原型：int sunxi_set_voltage_by_phandle(const void *fdt, uint32_t phandle, uint vol_value, uint onoff)
- 作用：根据引用供电的句柄，配置对应供电电路的电压值与开关状态。
- 参数：
  - fdt：从外部传入的 work fdt。
  - phandle：指向引用供电的句柄。
  - vol_value：设置目标电压，-1 则不调压。
  - onoff：是否开关供电，-1 则不改动开关状态。
- 返回：
  - 成功，返回 0。
  - 失败，返回-1。

3.1.4 sunxi_set_voltage_by_full_name

- 函数原型：int sunxi_set_voltage_by_full_name(char *name, uint vol_value, uint onoff)
- 作用：根据引用供电的句柄，配置对应供电电路的电压值与开关状态。
- 参数：
  - name：供电的名字。
  - vol_value：设置目标电压，-1 则不调压。
  - onoff：是否开关供电，-1 则不改动开关状态。
- 返回：
  - 成功，返回 0。
  - 失败，返回-1。

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

1. v1.0.10 版本之前其他设备对 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);
}
```

2. v1.0.10 版本及之后

设备树配置中其他设备驱动不可直接引用特定管理类驱动，只可引用管理核心。

devm_power_supply_get_by_phandle和power_supply_set_property等接口无差异。

```
/{
  sunxi_usb_power_supply: sunxi_usb_power_supply {
    compatible = "x-powers,sunxi-usb-power-supply";
    status = "okay";
    det_usb_supply = <&usb_power_supply>;
  };

  sunxi_usb_power_limit: sunxi_usb_power_limit {
    compatible = "x-powers,sunxi-usb-power-limit";
    status = "okay";
    det_battery_supply = <&bat_power_supply>;
    det_usb_supply = <&sunxi_usb_power_supply>;
    det_typc_supply = <&tcpc>;
    det_multi_charge_supply = <&gpio_power_supply>;
    extcon_udc = <&udc>;
    extcon_phy = <&phy_switcher>;
  };
};

udc:udc-controller@0x05100000 {
  det_vbus_supply = <&sunxi_usb_power_supply>;
};
```

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

功能说明:

通过虚拟设备节点控制 PMIC 各路电源电压

操作步骤：

1. 定位虚拟设备节点路径（以 twi4 上挂载 AXP2101 为例）：

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

2. 设置目标电压（示例设置为 3V）：

```
echo 3000000 > max_microvolts
```

```
echo 3000000 > min_microvolts
```

说明

注意事项：

- **确保虚拟设备 Virtual regulator 已编译。**
- **可通过此节点将某路电源设置为常开状态。**

5.2.2 获取 regulator 引用设备

功能说明：

查看哪些设备引用了 PMIC 的各路电源

操作步骤：

1. 进入 regulator 设备节点目录（以 AXP2101 为例）：

```
cd /sys/class/regulator/regulator.1
```

2. 查看引用设备列表：

```
ls -l | grep ^l
```

说明

注意事项：

- **需要 root 权限访问 debugfs。**
- **不同 PMIC 芯片的 regulator 编号可能不同。**
- **也可以进入入 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

功能说明：获取 PMIC 各路电源的详细状态信息

操作步骤：1. 挂载 debugfs 文件系统：

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

2. 查看 regulator 状态摘要：

```
cat /sys/kernel/debug/regulator/regulator_summary
```

3. 查看特定 regulator 的详细状态（以 dcdc1 为例）：

```
cat /sys/kernel/debug/regulator/axp2101-dcdc1/regulator_state
```

案例：

以读取 AXP2101 的挂载供电为例：

```
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
```

表 5-1: regulator 状态说明一览

字段名称	说明
use	引用计数
open	打开状态
bypass	旁路状态
voltage	当前输出电压
current	当前输出电流
min/max	允许的最小/最大电压

说明

注意事项：

- **内核需配置 CONFIG_DEBUG_FS 选项。**
- **需要 root 权限访问 debugfs。**

5.2.4 寄存器读写操作

5.2.4.1 标准 regmap 方式：

操作流程：

1. 挂载 debugfs 文件系统：

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

2. 寄存器操作：

- 写操作：

```
echo [寄存器地址] [写入值] > /sys/kernel/debug/regmap/[设备名称]/registers
```

- 读操作：

```
cat /sys/kernel/debug/regmap/[设备名称]/registers
```

案例：

以读取 twi4 上挂载 AXP2101 的寄存器为例

```
cat /sys/kernel/debug/regmap/4-0034/registers
```

说明

注意事项：
- **错误操作可能导致芯片损坏。**
- **内核需配置 CONFIG_DEBUG_FS 选项。**
- **需要 root 权限访问 debugfs。**
- **原生内核的寄存器读操作能一次性将所有寄存器读出。**
- **如果想操作写寄存器，原生内核需配置 REGMAP_ALLOW_WRITE_DEBUGFS 宏。**

5.2.4.2 AXP 专用节点：

说明

该节点功能在内核电源驱动版本 v0.0.x 与后面的 v1.0.x 存在差异，使用时注意内核电源驱动版本状态。

说明

如何区分 v0.0.x 与 v1.0.x：看是否存在相关版本管理文件 version。
version 路径：longan\bsp\drivers\power\version

5.2.4.2.1 v0.0.x

操作流程：

- 写寄存器：

```
echo [地址][值] > /sys/class/axp/axp_reg
```

- 读寄存器：

```
echo [地址] > /sys/class/axp/axp_reg
```

```
cat /sys/class/axp/axp_reg
```

案例：

```
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
```

说明

注意事项：

- **错误操作可能导致芯片损坏。**
- **该节点只可读主控 pmic 的寄存器，无法读取 pmu-ext 和 bmu-ext 等扩展芯片的寄存器**

5.2.4.2.2 v1.0.x 及之后

操作流程：

- 写寄存器：

```
echo [地址][值] > /sys/class/axp/[芯片驱动名]/axp_reg
```

- 读寄存器：

```
echo [地址] > /sys/class/axp/[芯片驱动名]/axp_reg
```

```
cat /sys/class/axp/[芯片驱动名]/axp_reg
```

案例：

```
往axp515寄存器0x0f写入值0x55：
echo 0x0f55 > /sys/class/axp/axp515/axp_reg

读出axp515寄存器0x0f的值：
echo 0x0f > /sys/class/axp/axp515/axp_reg
cat /sys/class/axp/axp515/axp_reg
```

说明

注意事项：

- **错误操作可能导致芯片损坏。**
- **该节点可以读所有加载的 pmic 驱动的寄存器，读取前注意芯片驱动名拼写正确。具体芯片驱动名见表格《PMIC 驱动对照表》**

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。

电池状态查询：

```
cat /sys/class/power_supply/[电池驱动节点]/uevent
```

USB 输入状态查询：

```
cat /sys/class/power_supply/[USB 驱动节点]/uevent
```

多口充电输入状态查询：

```
cat /sys/class/power_supply/[多口充电驱动节点]/uevent
```

以 axp517 为例，部分节点可能不在其中：

5.2.5.1 battery

```
/ # cat sys/class/power_supply/axp517-battery/uevent
POWER_SUPPLY_NAME=axp517-battery
POWER_SUPPLY_TYPE=Battery
POWER_SUPPLY_PRESENT=1
POWER_SUPPLY_CAPACITY=99
POWER_SUPPLY_CAPACITY_LEVEL=High
POWER_SUPPLY_HEALTH=Good
POWER_SUPPLY_TEMP=209
POWER_SUPPLY_STATUS=Discharging
POWER_SUPPLY_VOLTAGE_NOW=4268000
POWER_SUPPLY_CURRENT_NOW=19000
POWER_SUPPLY_CONSTANT_CHARGE_CURRENT=1984
POWER_SUPPLY_CHARGE_CONTROL_LIMIT=0
POWER_SUPPLY_CHARGE_COUNTER=6930000
POWER_SUPPLY_CHARGE_FULL=7000000
POWER_SUPPLY_CYCLE_COUNT=0
POWER_SUPPLY_MODEL_NAME=axp517-battery
POWER_SUPPLY_MANUFACTURER=xpower,axp517
POWER_SUPPLY_CAPACITY_ALERT_MIN=5
POWER_SUPPLY_VOLTAGE_MAX_DESIGN=4350
POWER_SUPPLY_TEMP_ALERT_MIN=-100
POWER_SUPPLY_TEMP_ALERT_MAX=648
POWER_SUPPLY_TEMP_AMBIENT_ALERT_MIN=-50
POWER_SUPPLY_TEMP_AMBIENT_ALERT_MAX=600
POWER_SUPPLY_ENERGY_FULL_DESIGN=7000000
POWER_SUPPLY_CHARGE_FULL_DESIGN=7000000
POWER_SUPPLY_CHARGE_CONTROL_LIMIT_MAX=2
POWER_SUPPLY_MANUFACTURE_YEAR=2024
POWER_SUPPLY_MANUFACTURE_MONTH=1
POWER_SUPPLY_MANUFACTURE_DAY=1
```

表 5-2: battery 状态动态信息一览

参数名称	对应含义
POWER_SUPPLY_PRESENT	电池存在状态
POWER_SUPPLY_CAPACITY	电池电量百分比
POWER_SUPPLY_CAPACITY_LEVEL	电池电量等级

参数名称	对应含义
POWER_SUPPLY_HEALTH	电池健康状态
POWER_SUPPLY_TEMP	电池温度
POWER_SUPPLY_STATUS	充电状态
POWER_SUPPLY_VOLTAGE_NOW	电池电压
POWER_SUPPLY_CURRENT_NOW	电池充电电流
POWER_SUPPLY_CONSTANT_CHARGE_CURRENT	充电电流上限
POWER_SUPPLY_CHARGE_CONTROL_LIMIT	charger cooling 等级
POWER_SUPPLY_CHARGE_COUNTER	当前电流
POWER_SUPPLY_CHARGE_FULL	当前满充电量
POWER_SUPPLY_CYCLE_COUNT	当前充电循环次数
POWER_SUPPLY_TIME_TO_EMPTY_NOW	放电放光时间
POWER_SUPPLY_TIME_TO_FULL_NOW	电池充满时间

表 5-3: battery 状态静态信息一览

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_MODEL_NAME	驱动名称
POWER_SUPPLY_MANUFACTURER	电池驱动名字
POWER_SUPPLY_CHARGE_FULL_DESIGN	设计的满充电量
POWER_SUPPLY_ENERGY_FULL_DESIGN	设计的满充电量
POWER_SUPPLY_CAPACITY_ALERT_MIN	警告电量
POWER_SUPPLY_VOLTAGE_MAX_DESIGN	满充电压
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_CHARGE_CONTROL_LIMIT_MAX	charger cooling 最大等级
POWER_SUPPLY_MANUFACTURE_YEAR	电池出厂日期-年
POWER_SUPPLY_MANUFACTURE_MONTH	电池出厂日期-月
POWER_SUPPLY_MANUFACTURE_DAY	电池出厂日期-日

5.2.5.2 usb

```
/ # cat sys/class/power_supply/axp517-usb/uevent
POWER_SUPPLY_NAME=axp517-usb
POWER_SUPPLY_TYPE=USB
POWER_SUPPLY_ONLINE=1
POWER_SUPPLY_PRESENT=1
POWER_SUPPLY_VOLTAGE_NOW=5000
POWER_SUPPLY_INPUT_CURRENT_LIMIT=2500
POWER_SUPPLY_TEMP=353
POWER_SUPPLY_SCOPE=Unknown
POWER_SUPPLY_USB_TYPE=SDP [DCP]
POWER_SUPPLY_MANUFACTURER=xpower,axp517
POWER_SUPPLY_VOLTAGE_MIN_DESIGN=4700
```

表 5-4: usb 状态动态信息一览

参数名称	对应含义
POWER_SUPPLY_ONLINE	usb 接入状态
POWER_SUPPLY_PRESENT	usb 接入状态
POWER_SUPPLY_VOLTAGE_NOW	vbus 设置电压
POWER_SUPPLY_INPUT_CURRENT_LIMIT	输入限流
POWER_SUPPLY_TEMP	充电芯片温度
POWER_SUPPLY_SCOPE	充电 cc 状态
POWER_SUPPLY_USB_TYPE	usb 接入充电类型

表 5-5: usb 状态静态信息一览

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_MANUFACTURER	充电驱动名字
POWER_SUPPLY_VOLTAGE_MIN_DESIGN	vindpm 电压

5.2.5.3 多口充电

```
/ # cat sys/class/power_supply/axp517-acin/uevent
POWER_SUPPLY_NAME=axp517-acin
POWER_SUPPLY_TYPE=Mains
POWER_SUPPLY_MODEL_NAME=axp517-acin
POWER_SUPPLY_ONLINE=0
```

表 5-6: 多口充电状态动态信息一览

参数名称	对应含义
POWER_SUPPLY_ONLINE	多口充电接入状态

表 5-7: 多口充电状态静态信息一览

参数名称	对应含义
POWER_SUPPLY_NAME	驱动名称
POWER_SUPPLY_TYPE	驱动类型（Mains）
POWER_SUPPLY_MANUFACTURER	充电驱动名字

5.2.5.4 TYPE-c

```
/ # cat sys/class/power_supply/tcpm-source-psy-axp517-typec-port.0/uevent
POWER_SUPPLY_NAME=tcpm-source-psy-axp517-typec-port.0
POWER_SUPPLY_TYPE=USB
POWER_SUPPLY_USB_TYPE=[C] PD PD_PPS
POWER_SUPPLY_ONLINE=0
POWER_SUPPLY_VOLTAGE_MIN=0
POWER_SUPPLY_VOLTAGE_MAX=0
POWER_SUPPLY_VOLTAGE_NOW=0
POWER_SUPPLY_CURRENT_MAX=0
POWER_SUPPLY_CURRENT_NOW=0
```

表 5-8: TYPE-c 状态动态信息一览

参数名称	对应含义
POWER_SUPPLY_ONLINE	TYPE-c 接入状态
POWER_SUPPLY_USB_TYPE	TYPE-c 握手识别到的接入类型
POWER_SUPPLY_VOLTAGE_MIN	TYPE-c 握手后的最小协议电压
POWER_SUPPLY_VOLTAGE_MAX	TYPE-c 握手后的最大协议电压
POWER_SUPPLY_VOLTAGE_NOW	TYPE-c 握手后的当前协议电压
POWER_SUPPLY_CURRENT_NOW	TYPE-c 握手后的当前协议电流
POWER_SUPPLY_CURRENT_MAX	TYPE-c 握手后的最大协议电流

表 5-9: TYPE-c 状态静态信息一览

参数名称	对应含义
POWER_SUPPLY_NAME	驱动名称
POWER_SUPPLY_TYPE	驱动类型（USB）

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。

说明

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、库仑计百分比

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

1、如无特殊说明，以下代码演示均以 linux5.15 及以上使用 axp2202 为例。
2、如无特殊说明，则该配置为全平台全芯片的通用配置。
3、标注了 (uboot) 后缀的配置项是会在 uboot 阶段生效的配置，无标注默认仅内核生效。

表 6-1: PMIC 功能配置分类对照表

分类名称	核心特点	典型示例
PMIC 独立配置	仅需修改 dts 即可生效	基础电压调节、GPIO 状态配置
驱动关联配置	需要主驱动与子驱动协同工作	Type-C PD 协议控制
硬件依赖配置	涉及硬件设计变更	多口充电属性配置
平台专用配置	非标准历史遗留实现	GPIO 口耐压值配置

表 6-2: PMIC 常见功能配置快速索引-PMIC 独立配置

配置分类	主要功能
uboot 属性配置	uboot 阶段电压调节/开机模式设置
无电池方案	无电池工作模式配置
唤醒源配置	电源事件唤醒系统设置
开关机配置	过温保护/过流保护设置
重启配置	长按按键行为控制
powerkey 配置	自定义电源按键行为
输入限流/限压	适配器/USB 输入电流电压限制
基础电池属性	电池参数/低电警告设置
充电属性	BC1.2/充电限流/预充电设置
NTC 温控	温度监测保护功能配置
充电指示灯	充电指示灯功能配置
电源欧盟标准配置	和电源欧盟标准有关联的配置

表 6-3: PMIC 常见功能配置快速索引-外部驱动关联配置

配置分类	主要功能
充电温控策略	电池温度监测与保护
drivevbus 配置	USB OTG 供电管理
type-c	Type-C 接口协议控制

表 6-4: PMIC 常见功能配置快速索引-扩展专项配置

配置分类	主要功能
多口充电	管理多口充电的充电状态
快充	管理快充充电相关功能
外挂 DCDC	扩展电源管理

表 6-5: PMIC 常见功能配置快速索引-平台专用配置

配置分类	主要功能
GPIO 口耐压值配置	GPIO 引脚电压设置
其他杂项配置	其他历史遗留配置项

6.1.1 独立 PMIC 通用配置

6.1.1.1 uboot 属性配置

说明

无电池属性配置和 NTC 温控配置等跨阶段的属性配置详见后续相应的章节。

6.1.1.1.1 供电调节

功能说明:

1. 在 uboot 阶段可调节各路电的电压和开关状态
2. 可配置 DCDC 的 PWM 模式以提高抗负载扰动能力

配置示例:

```
device/……/uboot-board.dts:
&power_sply {
    aldo1_vol = <1101800>;
    aldo2_vol = <1101800>;
```

```
    aldo3_vol = <1003300>;
    aldo4_vol = <1001800>;
    bldo1_vol = <3300>;
    bldo2_vol = <1001800>;
    bldo4_vol = <1101800>;
    cldo1_vol = <1001800>;
    cldo3_vol = <1003300>;
    cldo4_vol = <1103300>;
    cpusldo_vol = <1000900>;
    dcdc1_mode = <1>;
    dcdc2_mode = <1>;
};
```

```
xxxx_vol <u32>
    uboot阶段电压调节配置，配置名中xxxx为供电输出名，数值格式为前缀+电压值(mV)
    未配置的电在uboot阶段不会进行开关电和调压操作。
    前缀：
      100 - 开启供电
      110 - 开启供电但烧写时关闭
      000 - 关闭供电(可省略)
    示例：1101800表示开启供电，烧录时关闭，配置输出电压1800mV

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。
    0: PFM-PWM自动切换模式(默认)
    1: 强制PWM模式
    **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、AXP803、AXP8191、AXP1530、AXP806、AXP305、AXP81X**
```

6.1.1.1.2 开机判断配置

功能说明：

1. 配置机器的开机模式
2. 设置开机最低电压和电量要求

配置示例：

```
device/……/uboot-board.dts:
&power_sply {
    charge_mode = <1>;
};
&charger0 {
    pmu_safe_vol = <3400>;
    pmu_safe_ratio = <1>;
};
```

```
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_radio <u32>
  uboot阶段允许开机到内核的最低电量，默认为1%
```

6.1.1.1.3 低电图标显示模式配置

功能说明：

1. 控制低电状态下接入充电器时的图标显示行为
2. 适用于需要特殊低电提示的场景

配置示例：

```
device/·····/uboot-board.dts:
pmu0: pmu@0{
  ·····
  bat_bmp_type = <0>;
  ·····
};
```

```
bat_bmp_type <bool>
    配置是否第一时间显示低电图标，即，在低电状态下接入充电器时：
    0：先保持黑屏，直到拔掉充电器或者摁下按键才出现红色图标
    1：直接显示红色低电图标
    默认配置是0。
```

6.1.1.1.4 关机充电模式配置

功能说明：

1. 控制是否允许在充电状态下关机
2. 适用于需要严格电源管理的场景
3. 是否进入关机充电模式虽由 bmu 判断，但关机充电模式配置跟随 pmu

配置示例：

```
device/·····/board.dts:
pmu0: pmu@0{
  ·····
  pmu-charging-poweroff = <1>;
  ·····
};
```

```
pmu-charging-powerof <bool>
    配置是否禁用关机充电模式，即，即使有电池+充电器都会走关机流程
    默认disable。
    0：使用关机充电模式
    1：禁用关机充电模式
    **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、AXP803、AXP8191+AXP515、AXP8191+AXP517

6.1.1.2 无电池方案属性

功能说明：

1. 配置无电池工作模式
2. 适用于调试或特殊硬件方案

配置示例：

6.1.1.2.1 uboot

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

```
battery_exist <bool>
    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，仅uboot生效。
    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

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》
```

6.1.1.2.2 kernel

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

```
status = "disabled";
    该属性配置后不加载电池驱动，强制认为无电池存在，仅内核生效。
    建议优先使用该方法来配置无电池模式。
    **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、AXP2202、AXP515**

pmu_bat_unused <bool>
    适用于部分无电持方案或factory_mode的无电池场景的调试。
    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**
```

6.1.1.3 唤醒源配置

功能说明：

1. 配置 PMIC 是否能作为唤醒源
2. 配置各种唤醒源的使能

说明

按键部分的唤醒配置归类在 powerkey 自定义属性中

配置示例：

```
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
```

6.1.1.4 开关机配置

功能说明:

1. 配置电源管理相关保护功能
2. 包括过温保护、过流保护等

说明

关机充电电流归类在基础充电属性中

配置示例:

```
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：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**
    **当前支持AXP：AXP2101、AXP2202**

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、AXP8191**
```

6.1.1.5 重启配置

功能说明：

1. 配置长按按键行为
2. 控制是关机还是重启

配置示例：

```
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
    **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、
AXP517

6.1.1.6 powerkey 自定义属性

功能说明：

1. 完全自定义电源按键行为
2. 包括按键时长、功能等

配置示例：

```
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
    8 8ms
    16 16ms
    32 32ms
    64 64ms
    **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**

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
```

6.1.1.7 输入限流限压配置

功能说明：

1. **输入限流**：限制从适配器/USB 端口输入到系统的总电流
- 决定外部电源能提供的最大电流
- 影响系统充电速度和供电能力
2. **输入限压**：限制输入电压下限
- 当充电器电压过低时限流，防止拉挂充电器

电流关系：

```
系统总输入电流 = 电池充电电流 + 系统运行电流(SOC功耗)
```

配置示例：

```
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(vimdpm)
    单位为mV

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

6.1.1.8 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

6.1.1.8.1 电池满充电压、电池内阻、电池容量

功能说明：

1. 满充电压：决定电池充电终止电压
2. 电池内阻：影响充电效率和电量计算精度
3. 电池容量：用于显示满充状态下有多少 mAh

配置示例：

```
device/·····/board.dts:
bat_power_supply: bat-power-supply {
  ·····
  pmu_init_chgvol = <4200>;
  pmu_battery_rdc= <147>;
  pmu_battery_cap = <7000>;
  ·····
};
```

```
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
```

6.1.1.8.2 低电警告电量

功能说明：

1. 设置低电量警告阈值，默认为 15% 和 0% 2. 触发系统低电提示

```
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%
```

说明

当前仅存在一个警告等级的情况下，默认值是 15%。
只有一个警告等级的芯片驱动：AXP2601、AXP2602.

6.1.1.8.3 电池曲线参数

功能说明：电流型电量计电池曲线 (AXP803/AXP515)：

1. 基于库仑计原理，通过测量充放电电流积分计算电量
2. 需要配置 32 个电池参数 (para1-para32)
3. 参数通过专用仪器或 APP 测量获得

电压型电量计电池曲线 (AXP2202/AXP2101/AXP517/AXP2601/AXP2602)：

1. 基于电压-容量对应关系计算电量
2. 一般需要配置 128 字节的电池参数数组 (AXP2601 是 80 字节)
3. 参数测试需参考 1 号通上的文档《X-POWERS 电池参数测试系统使用说明 _V1.4.pdf》测量

配置示例：

电流型电量计配置

以 AXP803 为例：

```
device/·····/board.dts:
bat_power_supply: bat-power-supply {
  ·····
  pmu_bat_para1 = <0>;
  pmu_bat_para2 = <0>;
  pmu_bat_para3 = <0>;
  ·····
  pmu_bat_para30 = <98>;
  pmu_bat_para31 = <100>;
  pmu_bat_para32 = <100>;
```

```
    ......
};
```

```
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、AXP515**
```

电压型电量计配置

以 AXP2202 为例：

```
    ......
bat_power_supply: bat-power-supply {
    param = <&axp2202_parameter>;
};
    ......

/{
    axp2202_parameter:axp2202-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>;
        };
    };
};
```

```
xxxx_parameter
    指定电池充电结点，节点名称一般不固定，确保bat_power_supply下param能准确引用即可。
    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、AXP2101、AXP517、AXP2601、AXP2602**
```

6.1.1.8.4 Android 配置节点权限说明

功能说明：

配置电池容量相关节点的权限，使 Android 系统能够正确显示电池信息

影响功能：

设置界面中的电池容量显示

配置示例：

以 AXP2202 为例：

```
device/softwinner/[方案平台名]/common/system/init.sun[芯片平台名].ntc.rc:

on boot
   chmod 666 /sys/class/power_supply/axp2202-battery/energy_full_design
   chown system system /sys/class/power_supply/axp2202-battery/energy_full_design
```

```
energy_full_design
   电池满充电量节点
```

6.1.1.9 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

6.1.1.9.1 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。

功能说明：

1. 自动识别充电源类型 (PC/充电器)
2. 根据电源类型自动调节充电电流

配置示例：

```
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
```

6.1.1.9.2 电池充电限流

功能说明：

1. 分阶段设置充电电流 (运行/休眠/关机)
2. 保护电池寿命和系统稳定性

配置示例：

```
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
```

6.1.1.9.3 预充电电流限制

功能说明：配置低电量时的安全充电电流

配置示例：

```
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**
```

6.1.1.9.4 截止充电电流

功能说明：充电终止条件判断

配置示例：

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

```
pmu_iterm_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**
```

6.1.1.10 NTC 温控属性配置

NTC 温控相关属性如下所示：

公共通用配置：

1. ntc、jeita 使能
2. 开机温度限制 (uboot)
3. Android 配置节点权限

差异化配置：

1. jeita 限流参数
2. ts 电流配置

3. 电池温度参数
4. 触发限流、停充、关机的电压阈值
5. boost 放电过温/欠温配置
6. 软件 NTC 功能的实现

表 6-6: NTC 温控功能支持配置一览

配置类别	AXP803	AXP2202	AXP515	AXP517	ETA6973	AXP519
NTC 停充	✓	✓	✓	✓	✓	✓
JEITA 限流	☒	✓	✓	✓	✓	✓
限流可配	☒	✓	✓	✓	☒	☒
开机限制	☒	✓	✓	✓	✓	✓
温控关机	✓	✓	✓	✓	✓	☒
充电控制	硬件实现	硬件实现	软件实现	硬件实现	软件实现	硬件实现
参数配置	ts 电压值	ts 电压值	ts 电压值	ts 电压值	ts 电压值	NTC 阻值
参数单位	mV	mV	mV	mV	mV	Ω
中断数量	2	4	4 + 4	4	4	1
多 adc 采样	☒	☒	✓	☒	✓	☒
放电温控	☒	☒	☒	☒	☒	✓

说明

1. **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**
2. **NTC 温控功能的充电控制实现跟随 BMU 中的充电芯片（Charger），但其参数配置在 battery 驱动下（对应电量计芯片(Gauge)）**

6.1.1.10.1 ntc、jeita 使能

功能说明:

1. NTC 功能：启用电池温度监测后默认启动 ntc 功能
2. JEITA 功能：启用锂电池温度管理规范

配置示例:

```
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>;
    ……
};
```

```
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功能关闭
    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: disalbe
    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: disalbe
    1: enable
```

6.1.1.10.2 开机温度限制（uboot）

功能说明：设置允许开机的温度阈值

配置示例：

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 = <0xFFFFFFCE>;
};
```

```
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 = <0xFFFFFFCE>;
```

6.1.1.10.3 Android 配置节点权限

功能说明：

配置温度相关节点的权限，实现系统级温度保护功能

影响功能：

1. 过温/欠温自动关机
2. 充电温度保护
3. 温度异常弹窗提醒

配置示例：

以 AXP2202 为例：

```
device/softwinner/[方案平台名]/common/system/init.sun[芯片平台名].ntc.rc:

on boot
   chmod 666 /sys/class/power_supply/axp2202-battery/temp_alert_max
   chown system system /sys/class/power_supply/axp2202-battery/temp_alert_max
   chmod 666 /sys/class/power_supply/axp2202-battery/temp_alert_min
   chown system system /sys/class/power_supply/axp2202-battery/temp_alert_min
   chmod 666 /sys/class/power_supply/axp2202-battery/temp_ambient_alert_max
   chown system system /sys/class/power_supply/axp2202-battery/temp_ambient_alert_max
   chmod 666 /sys/class/power_supply/axp2202-battery/temp_ambient_alert_min
   chown system system /sys/class/power_supply/axp2202-battery/temp_ambient_alert_min
```

```
temp_alert_max
temp_alert_min
  电池过温/关机温度节点

temp_ambient_alert_max
temp_ambient_alert_min
  电池过温/关机停充节点
```

6.1.1.10.4 jeita 限流参数

功能说明：

配置触发 JEITA 限流时的电流限制比例

表 6-7: jeita 限流参数支持一览

Charger 型号	限流电流档位	默认档位
AXP803	/	/
AXP2202	0%/50%/75%/100%	50%
AXP515	0%/50%/75%/100%	50%
AXP517	0%/50%/75%/100%	50%
ETA6973	50%	50%
AXP519	50%~25%	/

说明

1. **需配置 jeita 功能使能为 1 才生效。**
2. **限流电流档位是指触发了 jeita 之后，电流降低到原本限流值的百分之多少。**
3. **AXP803 不支持的 jeita 限流。**
4. **AXP519 的限流值会由硬件动态调配。**
5. **ETA6973 的限流挡位固定为 50%。**

配置示例：

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

```
pmu_jcool_ifall <u32>
    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>
    00：100%
    01：50%，电流降低到一半
    10：75%，电流降低到75%
    00：0%
```

6.1.1.10.5 ts 电流配置

功能说明：

配置温度传感器工作电流

表 6-8: TS 引脚电流配置支持一览

Charger 型号	可配置电流档位	默认电流值
AXP803	20uA/40uA/60uA/80uA	80uA
AXP2202	20uA/40uA/50uA/60uA	50uA
AXP515	20uA/40uA/60uA/80uA	60uA
AXP517	20uA/40uA/50uA/60uA	60uA
ETA6973	/	/
AXP519	/	/

说明

1. **ETA6973 本身无 ts 采样 adc，需借助其他芯片去进行的 ts 采样。**
2. **AXP519 的采样数据和其他的不同，没有相关配置**

配置示例：

```
device/……/board.dts:
bat_power_supply: bat-power-supply {
  ……
  pmu_bat_ts_current = <40>;
  ……
};

device/……/uboot-board.dts:
&charger0 {
  ntc_cur = <40>;
```

```
};
```

```
pmu_bat_ts_current <u32>
    TS pin的电流大小配置，电流大小跟随AXP
    如若修改，下面的电池温度参数在计算时也要换成对应的TS电流

ntc_cur <u32>
    uboot配置ts引脚电流，默认为0，该配置必须跟内核dts的一致
```

6.1.1.10.6 电池温度参数

功能说明：

配置 16 个温度点的 TS 电压值

表 6-9: NTC 温控参数范围

芯片型号	参数范围	单位
AXP803	80 ~ 3,276	mV
AXP2202	0 ~ 4,095	mV
AXP515	80 ~ 3,276	mV
AXP517	0 ~ 8,191	mV
AXP519	0 ~ 112,612	Ω
ETA6973	/	/

配置示例：

```
device/……/board.dts:
bat_power_supply: bat-power-supply {
  ……
  pmu_bat_temp_para1 = <2814>;
  ……
  pmu_bat_temp_para16 = <66>;
  ……
};

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

```
pmu_bat_temp_para[16 <u32>
    配置16个ntc参数，默认为0。
    不同电池包的温敏电阻特性不一样，需根据电池包的TS温敏电阻手册的结果进行配置。

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
```

表 6-10: NTC 温控电池温度参数对照表

参数名称	AXP515/517/2202/AXP803 (mV)	AXP519 (Ω)
pmu_bat_temp_para1	-25 度对应电压	-25 度对应电阻
pmu_bat_temp_para2	-15 度对应电压	-15 度对应电阻
pmu_bat_temp_para3	-10 度对应电压	-10 度对应电阻
pmu_bat_temp_para4	-5 度对应电压	-5 度对应电阻
pmu_bat_temp_para5	0 度对应电压	0 度对应电阻
pmu_bat_temp_para6	5 度对应电压	5 度对应电阻
pmu_bat_temp_para7	10 度对应电压	10 度对应电阻
pmu_bat_temp_para8	20 度对应电压	15 度对应电阻
pmu_bat_temp_para9	30 度对应电压	20 度对应电阻
pmu_bat_temp_para10	40 度对应电压	25 度对应电阻
pmu_bat_temp_para11	45 度对应电压	30 度对应电阻
pmu_bat_temp_para12	50 度对应电压	40 度对应电阻
pmu_bat_temp_para13	55 度对应电压	45 度对应电阻
pmu_bat_temp_para14	60 度对应电压	50 度对应电阻
pmu_bat_temp_para15	70 度对应电压	55 度对应电阻
pmu_bat_temp_para16	80 度对应电压	65 度对应电阻

配置说明：

1. 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
2. 对于 AXP515/517/2202/803：将阻值乘以 ts 电流配置中的值得到的电压数值（单位：mV）
3. 对于 AXP519：直接使用电阻值（单位：Ω）
4. 将计算得到的数值填入 pmu_bat_temp_para[1-16] 的节点中
5. uboot 的配置必须跟内核 dts 的一致

NTC 温度值对应表：

TS 温敏电阻手册上的 NTC 温度值对应表一般如下所示：

Temperature	equivalent resistance	detected voltage	ADC DATA
-20°C	63.00Kohm	3.150V	189Ch
-15°C	50.15Hohm	2.508V	1398h
-10°C	40.26Kohm	2.013V	FBAh
-5°C	32.55Kohm	1.628V	CB8h
0°C	26.49Kohm	1.325V	A5Ah
5°C	21.68Kohm	1.084V	878h
10°C	17.78Kohm	0.889V	6F2h
15°C	14.63Kohm	0.732V	5B8h
20°C	12.07Kohm	0.604V	4B8h
25°C	10.00Kohm	0.500V	3E8h
30°C	8.320Kohm	0.416V	340h
35°C	6.954Kohm	0.348V	2B8h
40°C	5.839Kohm	0.292V	248h
45°C	4.924Kohm	0.246V	1ECh
50°C	4.171Kohm	0.209V	1A2h
55°C	3.549Kohm	0.177V	162h
60°C	3.032Kohm	0.152V	130h

图 6-1: 普通 NTC 阻值/电压/温度值的对应表

AXP519 的 NTC 温度值对应表比较特殊，如下所示：

温度/℃	NTC 阻值/k	NTC 电压/V	NTC 电流/uA
-25	86.43	1.728	20
-20	67.77	1.355	20
-15	53.41	1.068	20
-10	42.47	1.698	40
-5	33.9	1.356	40
0	27.28	1.091	40
5	22.05	1.764	80
10	17.96	1.436	80
15	14.69	1.175	80
20	12.09	0.967	80
25	10.00	0.800	80
30	8.313	0.665	80
35	6.940	0.555	80
40	5.827	0.466	80
45	4.911	0.392	80
50	4.160	0.332	80
55	3.536	0.282	80
60	3.020	0.241	80
65	2.588	0.207	80

图 6-2: AXP519_NTC 温度值对应表

6.1.1.10.7 触发限流、停充、关机的电压阈值

功能说明：设置温度保护触发阈值

表 6-11: 温度保护阈值配置范围

芯片型号	保护类型	参数范围	单位	备注
AXP803	所有温度保护	0 ~ 3624	mV	统一范围
AXP2202	停充/关机过温	0 ~ 8160	mV	
	停充/关机欠温	0 ~ 510	mV	
	限流过温	0 ~ 4080	mV	
	限流欠温	0 ~ 2040	mV	
AXP517	停充/关机过温	0 ~ 8160	mV	
	停充/关机欠温	0 ~ 510	mV	
	限流过温	0 ~ 4080	mV	
	限流欠温	0 ~ 2040	mV	

芯片型号	保护类型	参数范围	单位	备注
AXP515	停充/关机过温	0 ~ 8160	mV	
	停充/关机欠温	0 ~ 510	mV	
	限流过温	0 ~ 8160	mV	
	限流欠温	0 ~ 510	mV	
AXP519	停充欠温	-5°C/0°C/5°C/10°C 换算后的参数值	Ω	电阻阈值
	停充过温	40°C/45°C/50°C/55°C 换算后的参数值	Ω	电阻阈值
ETA6973	/	/	/	/

说明

1. **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。**
2. **对 AXP519，如果其配置不等于对应参数值，则会取到大于该值的最近一个档位上。**
3. **对 ETA6973，本身无 ts 采样 adc，需借助其他芯片去进行的 ts 采样，其参数范围限制和进行 ts 采样的芯片相关。**

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

配置示例：

```
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参数阈值

pmu_bat_charge_htf <u32>
    触发电池高温停充的参数阈值

pmu_bat_shutdown_ltf <u32>
    非充电模式下，触发电池低温中断的参数阈值
    对于AXP519来说是boost放电欠温

pmu_bat_shutdown_htf <u32>
    非充电模式下，触发电池高温中断的参数阈值
    对于AXP519来说是boost放电过温

pmu_jetia_cool <u32>
```

```
    触发电池低温限流/限压的参数阈值

pmu_jetia_warm <u32>
    触发电池高温限流/限压的参数阈值
```

6.1.1.10.8 boost 放电过温/欠温配置

功能说明：在温度过高或者过低时，会关闭 buck-boost

说明

当前支持 AXP：AXP519

表 6-12: boost 放电保护阈值配置范围

芯片型号	保护类型	参数范围	单位	备注
AXP519	boost 放电欠温	-20°C/-10°C/-5°C/0°C 换算后的参数值	Ω	电阻阈值
	boost 放电过温	50°C/55°C/60°C/65°C 换算后的参数值	Ω	电阻阈值

配置示例：

```
device/·····/board.dts:
bat_power_supply: bat-power-supply {
  ·····
  pmu_bat_work_htf = <1381>;
  pmu_bat_work_ltf = <89>;
  ·····
};
```

```
pmu_bat_work_htf <u32>
    boost放电欠温的参数阈值

pmu_bat_work_ltf <u32>
    boost放电过温的参数阈值
```

6.1.1.10.9 软件 NTC 功能的实现

功能说明：当 BMU 中断不足时，使用 PMU ADC 实现完整温控功能。目前支持两种实现方式：

1. **AXP515 方案**：
   - 通过 AXP8191 的 ADC 获取温度数据
   - 在软件层面处理中断和温控逻辑
   - 需要配置power_temp_ctrl节点并绑定电池驱动
   - 支持唤醒功能

2. **ETA6973 方案**：
   - 通过 AXP717 的 ADC 直接采样温度
   - 硬件完成温度转换，软件直接读取结果
   - 温控配置集成在充电驱动中
   - 唤醒能力取决于 AXP717 芯片

说明

当前支持组合：
- **AXP8191+AXP515**
- **ETA6973+AXP2202**

表 6-13: 软件 NTC 温控实现方式对比

特性	AXP8191+AXP515 方案	ETA6973+AXP2202 方案
ADC 来源	AXP8191 ADC	AXP2202 ADC
温度数据处理	软件处理中断和计算	硬件完成转换，软件直接读取
配置节点	power_temp_ctrl	集成在充电驱动配置中
中断源	AXP8191(4 个)+AXP515(4 个)	AXP2202(4 个)
应用场景	AXP8191+AXP515 组合方案	ETA6973+AXP2202 快充方案

说明

- **虽具体实现方式存在差异，但其软件流程框架设计类似。**
- **ETA6973+AXP2202 的恢复迟滞温度值固定为 3°C。**
- **AXP8191+AXP515 的恢复迟滞温度值由电池温度参数决定。**

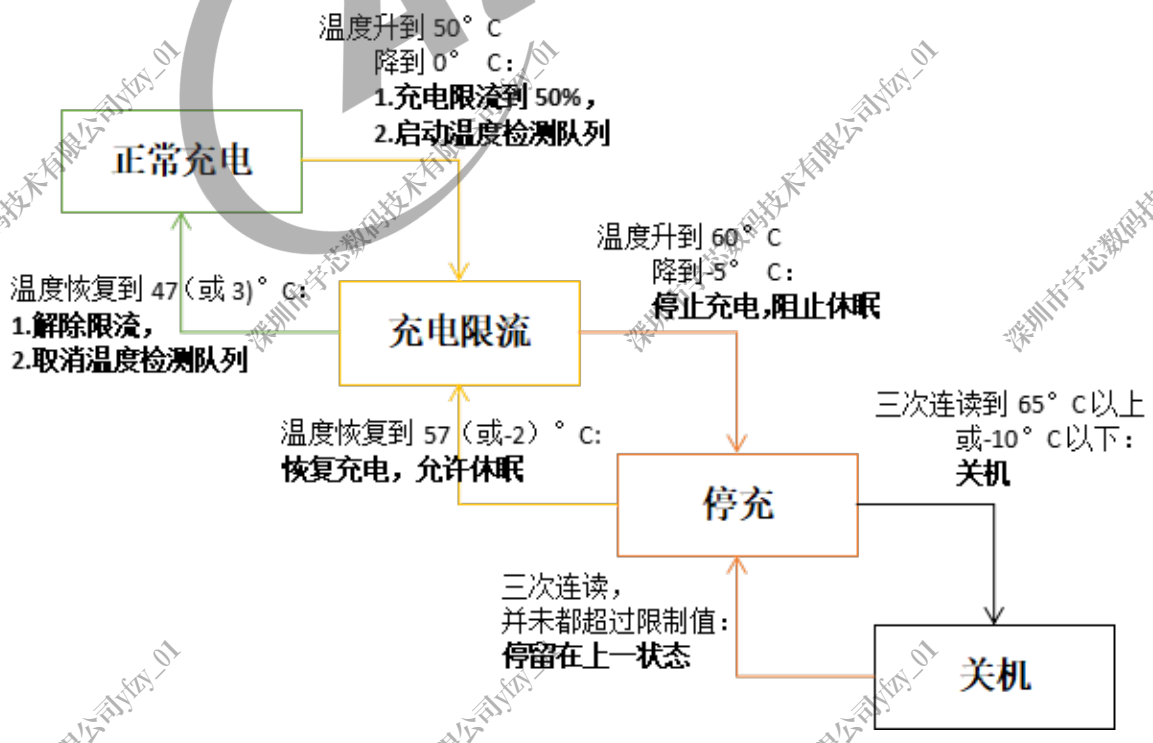

图 6-3: 软件 NTC 功能流程设计

配置示例：

AXP8191+AXP515 方案：

```
power_temp_ctrl0: power_temp_ctrl@0 {
    compatible = "x-powers,axp8191-temp-ctrl";
    status = "okay";
    det_bat_supply = <&bat_power_supply>;
};
```

```
power_temp_ctrl0: power_temp_ctrl@0 {
    status
        在不使用jeita的场景下，status一定要配置disabled

    det_bat_supply = <&bat_power_supply>;
        配置相对应的电池驱动
};
```

ETA6973+AXP2202 方案：

```
charger_power_supply: charger_power_supply {
    det_battery_supply = <&bat_power_supply>;
};
```

```
charger_power_supply: charger_power_supply@0 {
    det_bat_supply = <&bat_power_supply>;
        配置相对应的电池驱动
};
```

6.1.1.11 充电指示灯配置

功能说明：

1. 控制充电状态 LED 指示灯
2. 支持两种显示模式 (A/B)
3. 可配置由 PMU 或充电器控制

配置示例：

```
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>
        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
    **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**
```

6.1.1.12 电源欧标配置

功能说明：

1. 配置电池健康属性
2. 配置电池出厂日期

配置示例：

```
device/……/board.dts:
bat_power_supply: bat-power-supply {
  ……
  pmu_battery_cap = <7000>;
  pmu_bat_cycle_life = <800>;
  pmu_bat_cycle_cap_reduce = <20>;
  ……
  pmu_bat_manufacture_year = <2024>;
  pmu_bat_manufacture_month = <1>;
  pmu_bat_manufacture_day = <1>;
  ……
};
```

```
pmu_battery_cap <u32>
    电池容量，默认4000mAh
    单位为mAh

pmu_bat_cycle_life <u32>
    总循环次数上限

pmu_bat_cycle_cap_reduce <u32>
    达到总循环次数上限后，电池总电量衰减百分比

pmu_bat_manufacture_year <u32>
pmu_bat_manufacture_month <u32>
pmu_bat_manufacture_day <u32>
    电池出厂日期（年/月/日）
```

6.1.2 外部驱动关联配置

6.1.2.1 充电温控策略

功能说明：

1. 基于系统温度动态调整充电电流
2. 支持多传感器温度监测
3. 可配置温度触发点和限流比例

说明

1. **A523 及之后平台支持基础限流功能**
2. **A733 及之后平台支持多传感器温控**

配置参数：

限流值配置：

```
device/……/board.dts:
bat_power_supply: bat-power-supply {
  ……
  pmu_bat_charge_control_lim = <600>;
  ……
};
```

```
pmu_bat_charge_control_lim <u32>
    当板卡其他sensor温度偏高时constant充电电流限制，默认600mA
    单位为mA
    **视具体的board.dts属性节点而定，部分芯片平台暂未支持该功能**
    **当前支持平台：A523及之后的平台**
```

限流温度配置：

```
/{
  battery_supply_ths: battery_supply_ths {
    compatible = "allwinner,sunxi-max-temp-ths";
    ext-ths-rely = <&cpub_thermal_zone>, <&cpul_thermal_zone>, <&gpu_thermal_zone>, <&npu_thermal_zone>, <&
      ddr_thermal_zone>;
    #thermal-sensor-cells = <0>;
    status = "okay";
  };
};

&thermal_zones {
  battery_zone: battery_zone {
    polling-delay = <10000>;
    polling-delay-passive = <10000>;
    thermal-sensors = <&battery_supply_ths>;
    trips {
      battery_trip0: t0 {
        temperature = <60000>;
        hysteresis = <0>;
        type = "passive";
      };
      battery_trip1: t1 {
```

```
            temperature = <70000>;
            hysteresis = <0>;
            type = "passive";
          };
        };
        cooling-maps {
          map0 {
            trip = <&battery_trip0>;
            cooling-device = <&bat_power_supply 1 1>;
          };

          map1 {
            trip = <&battery_trip1>;
            cooling-device = <&bat_power_supply 2 2>;
          };
        };
      };
    };
};
```

```
**视具体的board.dts属性节点而定，部分芯片平台暂未支持该功能**
**当前支持平台：A733及之后的平台**
battery_supply_ths {
  ext-ths-rely
    当前取哪些sensor的温度，并取其中最高值为温控检测温度
}
thermal_zones {
  battery_zone {
    battery_trip0
      过温后恢复限流的温度点
    battery_trip1
      开始限流的温度点
  }
  cooling-maps {
    map0 {
      cooling-device
        配置相对应的电池驱动
    };
  }
}
```

6.1.2.2 drivevbus 属性配置

功能说明：

1. drivevbus 管理 AXP 对外输出供电的功能
2. 在 AXP2202/AXP515/AXP517 中：• vmid：负责 boost 升压供电
   • drivevbus：控制电流方向开关

3. 其他 AXP 型号仅通过 drivevbus 控制输出引脚模式

说明

支持 drivevbus 功能的 AXP 型号：AXP803、AXP2202、AXP515、AXP517、AXP519、ETA6973

配置示例：

AXP803 典型配置：

```
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 典型配置：

```
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 关联配置：

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

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

6.1.2.3 type-c 功能属性配置

功能说明：

1. 实现完整的 Type-C 接口功能，包括：
   - 电源角色切换 (Source/Sink/DRP)
   - 数据角色切换 (DFP/UFP/DRD)
   - PD 协议支持

2. 支持快充协议：
   - USB PD 3.0
   - PPS 可编程电源

3. 提供完整的充电/放电功率管理

适用芯片：

- AXP517 (集成 Type-C 控制器)
- HUSB311 (独立 Type-C 芯片)

配置示例：

Device Tree 配置

说明

当前支持完整 type-c 功能的 AXP 型号为：AXP517、HUSB311。

```
&twi5 {
  husb311: husb311@4e {
    usb_con: connector {
      data-role = "dual";
      power-role = "dual";
      try-power-role = "sink";
      op-sink-microwatt = <1000000>;
      slow-charger-loop;
      sink-pdos = < PDO_FIXED(5000, 500, (0x1<<29)|(0x1<<25))
            PDO_VAR(5000, 5000, 2000)
            PDO_VAR(9000, 9000, 2000)
            /* PDO_VAR(12000, 12000, 3000)
            PDO_VAR(15000, 15000, 3000)
            PDO_VAR(20000, 20000, 2250)*/
            >;
      source-pdos = < PDO_FIXED(5000, 500, (0x1<<29)|(0x1<<25))
              PDO_VAR(5000, 5000, 1200) >;
    };
  };
};
```

```
data-role <char>
    数据传输方向：
    dual：双向
    host/device：主机/从机
```

```
power-role <char>
    充电/供电方向：
    dual：双向
    sink/source：充电/放电
try-power-role <char>
    优先匹配充电/供电方向
    sink/source：充电/放电
op-sink-microwatt <char>
    最小充电功率
slow-charger-loop <bool>
    匹配档位时，是否从最低默认档位开始
sink-pdos <args>
    配置充电档位，例子中配置的档位如下：
    默认档：5V 0.5A
    档1：5V 2A
    档2：9V 2A
    档3：12V 3A
    档4：15V 3A
    档5：20V 2.25A
source-pdos <args>
    配置放电档位，例子中配置的档位如下：
    默认档：5V 0.5A
    档1：5V 1.2A
```

Android 配置

需要配置的文件均在device/softwinner/[方案平台名]/common/下：

config.mk

device/softwinner/[方案平台名]/common/misc/config.mk：

```
PRODUCT_COPY_FILES += \
  # usb配置
  $(PRODUCT_PREBUILT_PATH)/dist/sunxi-hci.ko:recovery/root/sunxi-hci.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/ehci-sunxi.ko:recovery/root/ehci-sunxi.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/ohci-hcd.ko:recovery/root/ohci-hcd.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/ohci-sunxi.ko:recovery/root/ohci-sunxi.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/sunxi_usb_udc.ko:recovery/root/sunxi_usb_udc.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/sunxi_usbc.ko:recovery/root/sunxi_usbc.ko \

  # Type-C配置
  $(PRODUCT_PREBUILT_PATH)/dist/typec_tcpci.ko:recovery/root/typec_tcpci.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/tcpci_axp517.ko:recovery/root/tcpci_axp517.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/tcpci_husb311.ko:recovery/root/tcpci_husb311.ko \

  # phy配置
  $(PRODUCT_PREBUILT_PATH)/dist/sunxi-phy-switcher.ko:recovery/root/sunxi-phy-switcher.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/phy-sunxi-plat.ko:recovery/root/phy-sunxi-plat.ko \
  $(PRODUCT_PREBUILT_PATH)/dist/dwc3-sunxi-plat.ko:recovery/root/dwc3-sunxi-plat.ko \
```

- 在新增Type-C驱动时，需要将新增的ko放到Type-C配置下。
- 需检查usb配置和phy配置是否完整，如不完整会影响Type-C驱动初始化

init.quick_charge.rc

device/softwinner/[方案平台名]/common/quick_charge/init.quick_charge.rc：

```
on early-init
  # Type-C配置
```

```
insmod /typec_tcpci.ko
insmod /tcpci_axp517.ko
insmod /tcpci_husb311.ko

# phy配置
insmod /sunxi-phy-switcher.ko
insmod /phy-sunxi-plat.ko
insmod /dwc3-sunxi-plat.ko

# usb配置
insmod /sunxi-hci.ko
insmod /ehci-sunxi.ko
insmod /ohci-hcd.ko
insmod /ohci-sunxi.ko
insmod /sunxi_usb_udc.ko
insmod /sunxi_usbc.ko
insmod /usbserial.ko
insmod /usb_wwan.ko
insmod /extcon-sunxi-plat.ko
```

- 在新增Type-C驱动时，需要将新增的ko放到Type-C配置下。
- 需检查usb配置和phy配置是否完整，如不完整会影响Type-C驱动初始化

init.secondmodules.rc

device/softwinner/[方案平台名]/common/system/init.secondmodules.rc：

```
on init

# Type-C配置
insmod /vendor/lib/modules/typec_tcpci.ko
insmod /vendor/lib/modules/tcpci_axp517.ko
insmod /vendor/lib/modules/tcpci_husb311.ko

# phy配置
insmod /vendor/lib/modules/ps8743.ko
insmod /vendor/lib/modules/sunxi-phy-switcher.ko
```

- 在新增Type-C驱动时，需要将新增的ko放到Type-C配置下。
- 需检查phy配置是否完整，如不完整会影响Type-C驱动初始化

init.[芯片平台名].rc

device/softwinner/[方案平台名]/common/system/init.[芯片平台名].rc：

```
on charger

# usb配置
insmod /vendor/lib/modules/ohci-hcd.ko
insmod /vendor/lib/modules/sunxi-hci.ko
insmod /vendor/lib/modules/ehci-sunxi.ko
insmod /vendor/lib/modules/ohci-hcd.ko
insmod /vendor/lib/modules/ohci-pci.ko
insmod /vendor/lib/modules/ohci-sunxi.ko
insmod /vendor/lib/modules/sunxi_usb_udc.ko
insmod /vendor/lib/modules/sunxi_usbc.ko
insmod /vendor/lib/modules/sunxi-inno-combophy.ko

# Type-C配置
insmod /vendor/lib/modules/typec_tcpci.ko
```

```
insmod /vendor/lib/modules/tcpci_axp517.ko
insmod /vendor/lib/modules/tcpci_husb311.ko

# phy配置
insmod /vendor/lib/modules/phy-sunxi-plat.ko
insmod /vendor/lib/modules/dwc3-sunxi-plat.ko
insmod /vendor/lib/modules/ps8743.ko
```

- 在新增Type-C驱动时，需要将新增的ko放到Type-C配置下。
- 需检查phy配置是否完整，如不完整会影响Type-C驱动初始化

6.1.3 扩展专项配置

6.1.3.1 多口充电属性配置

功能说明：

1. 支持两种多口充电方案：
   - ACIN 方案：USB+AC 双充电口，仅 USB 口能识别外设
   - 双 Type-C 方案：双 Type-C 充电口，双口都可识别外设

2. 主要特性：
   - 充电口自动切换
   - 充电限流管理
   - 充电状态检测

适用芯片：

- AXP2202
- AXP515
- AXP517

硬件要求：

1. ACIN 方案需满足：
   - 专用 AC 充电接口电路
   - 检测 GPIO 引脚

2. 双 Type-C 方案需满足：
   - 专用双 Type-C 接口电路
   - 检测 GPIO 引脚

说明

ACIN 与双 TYPE-C 的硬件与 usb 配置差异很大，进行开发使用时需要注意。
从外，双 TYPE-C 不可同时接入充电器，因为在同时接入充电器的场景下双 TYPE-C 方案无法充电。

说明

以下配置以内核电源驱动 v1.0.10 及之后的版本为主，使用前注意内核电源驱动版本号。

6.1.3.1.1 ACIN 充电方案设计

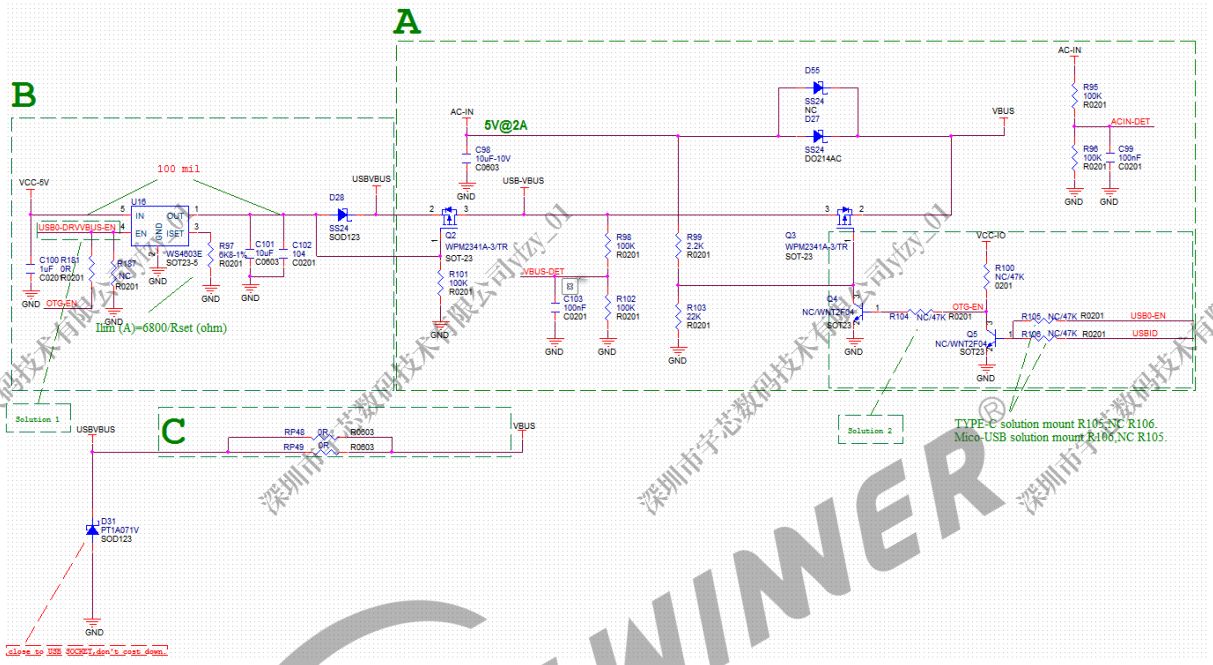

图 6-4: PMIC_ACIN 方案原理图

ACIN 兼容方案电路，整体包含 A、B、C 三个模块：

1. 当客户方案确认没有 ACIN 接口，A、B 两个模块 NC，C 模块电路贴片。
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。

6.1.3.1.2 双 TYPE-C 方案设计

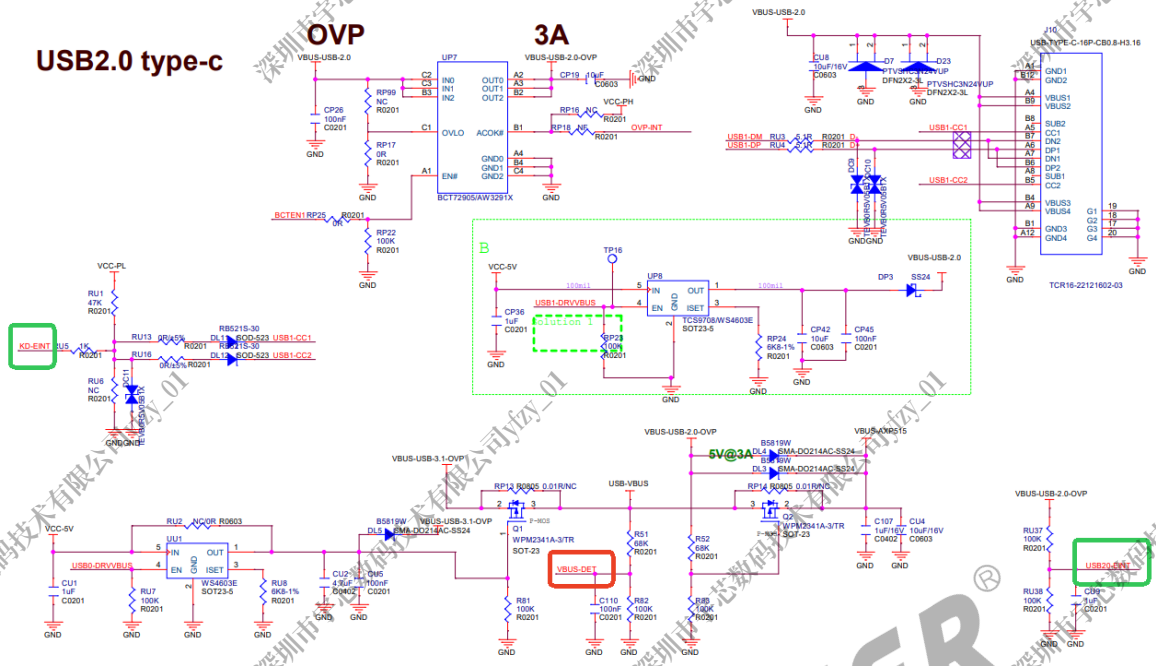

图 6-5: PMIC_ 双 TYPE-C 方案原理图

对于双 TYPE-C 方案：

1. 基于 ACIN 方案设计，额外添加多口的外设识别电路。
2. 红色框内是识别主口充电（USB）的 IO 口。
3. 绿色框为识别多口充电（multi_charge）的 IO 口组合。

说明

• 主口充电（USB）的 IO 口在部分方案下会被省略，转由 Type-C 驱动识别充电接入。• 多口充电（multi_charge）在部分方案下会沿用 ACIN 方案中的识别方式：用一个 IO 口来识别。

6.1.3.1.3 配置示例

因 ACIN 与双 TYPE-C 的电源部分驱动配置差异不大，因此以下内容以 ACIN 应用场景为例介绍电源配置。

menuconfig 配置

AXP2202 使用旧多口充电配置：

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

AXP515、AXP517 使用多口充电管理核心：

```
Allwinner BSP  --->  Device Drivers  --->
   PMIC Drivers  --->  <M> sunxi multi charge Power Supply Driver
```

Device Tree 配置

多口充电部分配置：

1. 使能多口充电驱动/管理核心
2. 配置多口检测功能

```
device/·····/board.dts:
usb_power_supply: usb_power_supply {
   ·····
   pmu_acin_usbid_drv = <&pio PH 12 GPIO_ACTIVE_LOW>;
   pmu_vbus_det_gpio = <&pio PH 13 GPIO_ACTIVE_LOW>;
   ·····
};

旧式多口充电驱动：
gpio_power_supply: gpio_power_supply {
   pmu_acin_det_gpio = <&pio PH 14 GPIO_ACTIVE_LOW>;
   //extcon = <&extcon_usb1>;
};

多口充电管理核心：
gpio_power_supply: gpio_power_supply {
   //multi_charge_det_gpio = <&pio PH 14 GPIO_ACTIVE_LOW>;
   extcon = <&extcon_usb1>;
};
```

```
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。
    仅在有简单cc逻辑的芯片使用type-c功能的情况下有效。
    **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、AXP515**

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/multi_charge_det_gpio
    检测多口接入的gpio口，用于判断ac接入。

extcon = <&extcon_usb1>
    检测多口接入的extcon来源，用于判断ac接入。
    当多口接入时，usb进行接入判断，并将结果通过extcon的方式发给acin驱动。
```

usb 部分配置（ACIN）：

```
&usbc0 {
   usb_drv_vbus_gpio = <&pio PH 7 GPIO_ACTIVE_HIGH>;
   enable-active-high;
};

&ehci0 {
   drvvbus-supply = <&reg_vmid>;
};
```

```
&ohci0 {
    drvvbus-supply = <&reg_vmid>;
};

extcon_usb1: extcon-usb1 {
    compatible = "allwinner,extcon-usb-gpio", "linux,extcon-usb-gpio";
    reg = <0x0 0x10 0x0 0x0>;
    interrupt-parent = <&r_pio>;
    id-gpios = <&r_pio PL 3 GPIO_ACTIVE_HIGH>;
    vbus-gpios = <&pio PJ 27 GPIO_ACTIVE_HIGH>;
    id-set-debounce = <0x1>;
    wakeup-source;
    status = "okay";
};
```

```
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。

drvvbus-supply = <&reg_vmid>;
      需要将ehci0和ohci0下引用的reg_drivevbus修改成reg_vmid。

extcon_usb1: extcon-usb1 {
    id-gpios = <&r_pio PL 3 GPIO_ACTIVE_HIGH>;
      配置识别usb id的gpio口。
    vbus-gpios = <&pio PJ 27 GPIO_ACTIVE_HIGH>;
      配置识别acin接入充电器的gpio口。
};
```

usb 部分配置（双 TYPE-C）：

```
&ehci0 {
    drvvbus-supply = <&reg_usb0_vbus>;
};

&ohci0 {
    drvvbus-supply = <&reg_usb0_vbus>;
};

extcon_usb1: extcon-usb1 {
    compatible = "allwinner,extcon-usb-gpio", "linux,extcon-usb-gpio";
    reg = <0x0 0x10 0x0 0x0>;
    interrupt-parent = <&r_pio>;
    id-gpios = <&r_pio PL 3 GPIO_ACTIVE_HIGH>;
    vbus-gpios = <&pio PJ 27 GPIO_ACTIVE_HIGH>;
    id-set-debounce = <0x1>;
    wakeup-source;
    status = "okay";
};

reg_boost_vbus: boost-vbus {
    compatible = "regulator-fixed";
    regulator-name = "boost-vbus";
    regulator-min-microvolt = <5000000>;
    regulator-max-microvolt = <5000000>;
    regulator-enable-ramp-delay = <1000>;
    gpio = <&r_pio PL 12 GPIO_ACTIVE_HIGH>;
    enable-active-high;
};
```

```
reg_usb0_vbus: usb0-vbus {
    compatible = "regulator-fixed";
    regulator-name = "usb0-vbus";
    regulator-min-microvolt = <5000000>;
    regulator-max-microvolt = <5000000>;
    regulator-enable-ramp-delay = <1000>;
    gpio = <&r_pio PL 2 GPIO_ACTIVE_HIGH>;
    enable-active-high;
    vin-supply = <&reg_boost_vbus>;
};
```

```
drvvbus-supply = <&reg_usb0_vbus>;
    需要修改ehci0和ohci0下引用的对外供电配置，使其和原理图适配。

extcon_usb1: extcon-usb1 {
    id-gpios = <&r_pio PL 3 GPIO_ACTIVE_HIGH>;
      配置识别usb id的gpio口。
    vbus-gpios = <&pio PJ 27 GPIO_ACTIVE_HIGH>;
      配置识别acin接入充电器的gpio口。
};

reg_usb0_vbus: reg_usb0_vbus {
    类似PMIC的drivevbus，控制电流方向开关。
};

reg_boost_vbus: reg_boost_vbus {
    类似PMIC的vmid，负责管理boost供电。
};
```

6.1.3.2 快充系统配置

功能说明：

1. 快充方案基于现有板型平台扩展实现，其板型命名规则为[基础板型]_[快充扩展后缀]
2. 快充系统以下两部分功能组成：

• Type-C：负责 PD 协议握手

• BMU：负责外部输入和电池管理，快充方案下常拆分为充电 (Charger) 和电量计 (Gauge) 两部分。

3. 支持多种快充方案：

• 单节快充：支持最高 18W(9V2A)，单颗电池

• 双节快充：支持最高 45W(15V3A 或 20V2.25A)，两颗或多颗电池

说明

只有支持 type-c 完整功能的 Type-C 芯片驱动才能完成 PD 协议握手，协同 BMU 实现快充功能。

板型扩展方案：

表 6-14: 快充方案板型扩展对照表

平台	快充类型	基础板型	快充扩展板型	快充后缀
A523	单节快充	a523_evb	a523_evb_qc_single	qc_single
A523	双节快充	a523_evb	a523_evb_qc_double	qc_double

平台	快充类型	基础板型	快充扩展板型	快充后缀
A733	单节快充	a733_ag863109vcb	a733_ag863109vcb_axp517	axp517
A733	双节快充	a733_evb1	a733_evb1_qc_double	qc_double

芯片组合方案：

表 6-15: 快充方案一览

平台	快充类型	Type-C	BMU(Charger)	BMU(Gauge)
A523	单节快充	HUSB311	ETA6973/ETA6974	AXP717
A523	双节快充	HUSB311	AXP519	AXP2601
A733	单节快充	AXP517(集成)	AXP517(集成)	AXP517(集成)
A733	双节快充	HUSB311	AXP519	AXP2602

说明

AXP517 是 typec，charger 和 gauge 三合一芯片。
uboot 阶段无 typec 驱动，因此在该阶段内无 pd 握手，无快充功能。

表 6-16: 快充方案芯片功能对照表

芯片驱动	对应功能类型	驱动类型（kernel）	驱动类型（uboot）
HUSB311	Type-C	typec	无驱动
ETA6973	BMU-ext（Charger）	power supply	BMU-ext
AXP519	BMU-ext（Charger）	power supply	BMU-ext
AXP2601	BMU-ext（Gauge）	power supply	BMU-ext
AXP2602	BMU-ext（Gauge）	power supply	BMU-ext
AXP517	Type-C + BMU	power supply + typec	BMU

说明

AXP517 是 typec，charger 和 gauge 三合一芯片。
uboot 阶段无 typec 驱动，因此在该阶段内无 pd 握手，无快充功能。

6.1.3.2.1 单节快充方案设计

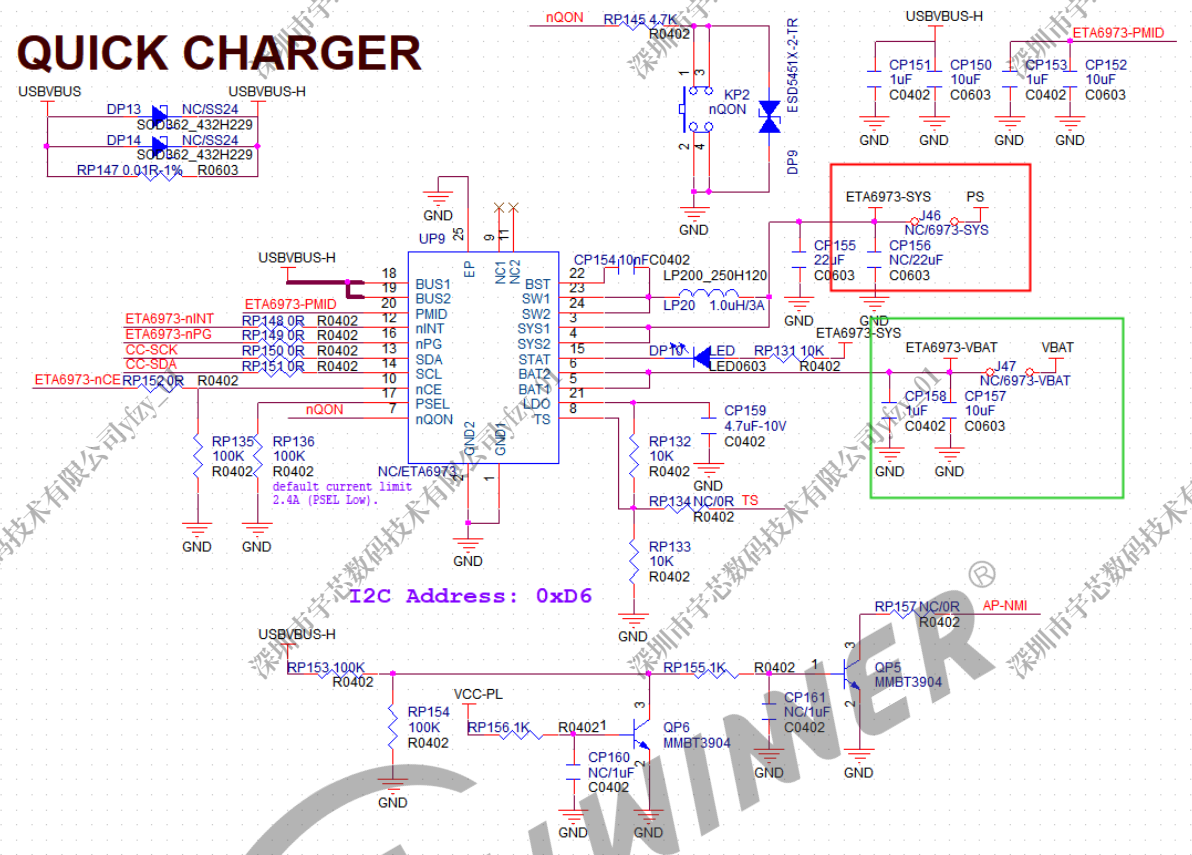

图 6-6: 单节快充硬件设计

说明

绿色框内 ETA6973 SYS 输出作为电源管理芯片（PMU）PS 的输入。

红色框内 ETA6973 VBAT 输出给单节电池充电，与电源管理芯片（PMU）的 VBAT 连接，使用电源管理芯片 (PMU) 的电量计功能。

其输入控制框架如下：

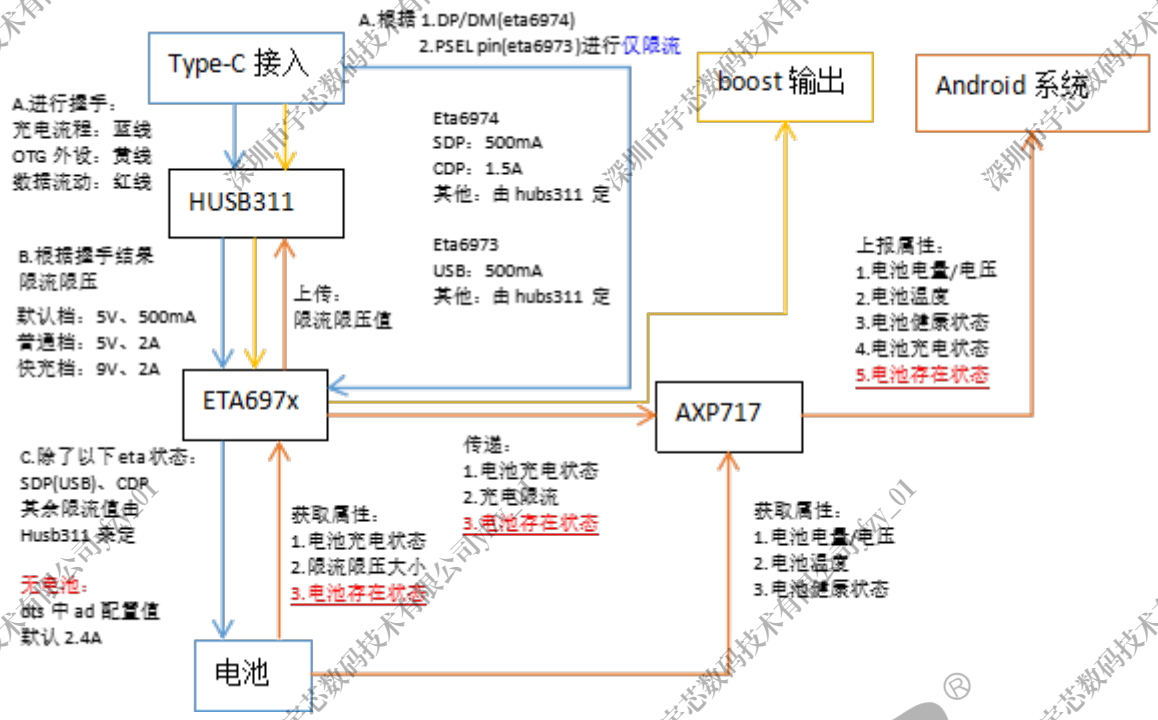

图 6-7: 单节快充限流通路框架

6.1.3.2.2 双节快充方案设计

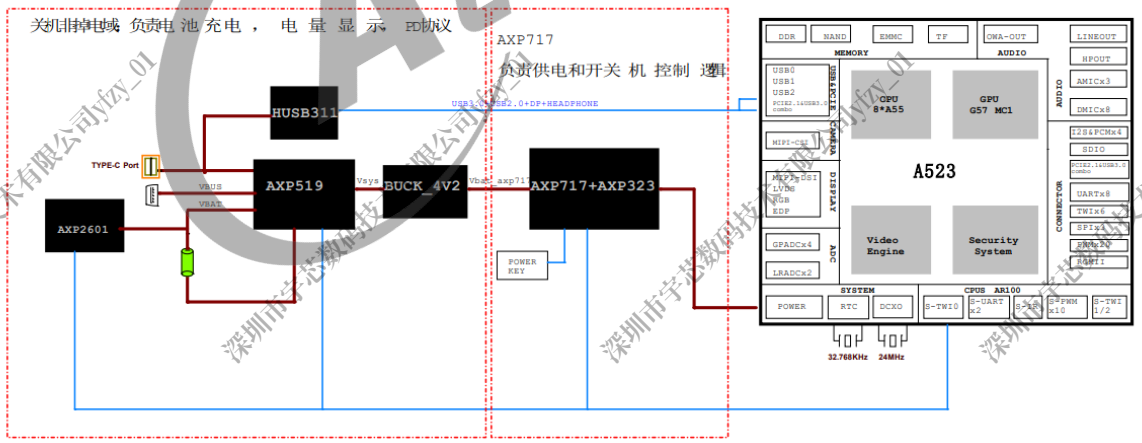

图 6-8: 双节快充硬件设计

说明

当方案上外挂独立电量计，并且外挂电量计由于存在上报电量的功能，其 IRQ 不能和系统的 IRQ 直连，需要单独接 GPIO。

其输入控制框架如下：

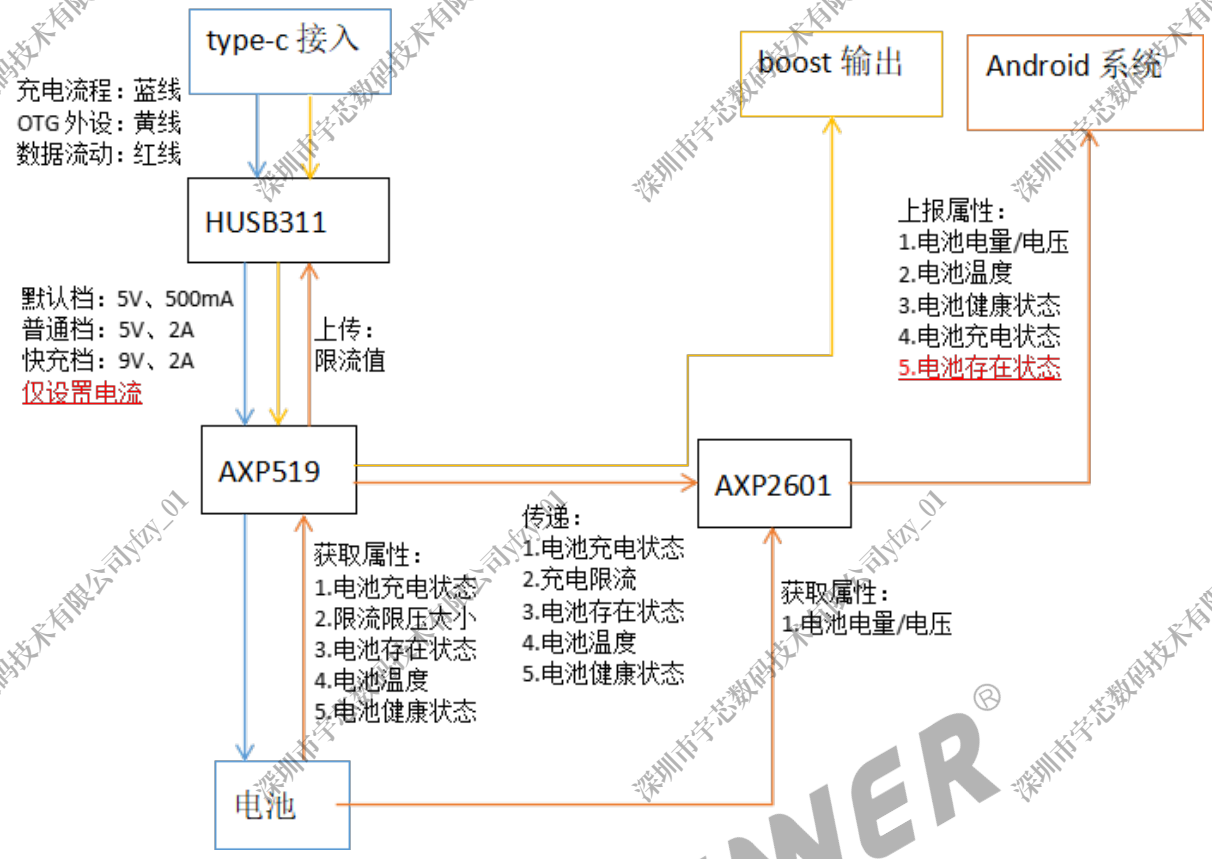

图 6-9: 双节快充限流通路框架

6.1.3.2.3 软件配置

需配置的内容：

1. 使能 type-c 完整驱动, 并完整配置属性，见章节《type-c 完整功能基础配置》和《type-c 功能属性配置》。
2. 配置 BMU 部分的属性，见章节《基础电池属性》等 BMU 功能的配置。

说明

在进行快充功能配置之前，需先检查相关 Type-C 功能和 BMU 功能是否配置完全。

menuconfig 配置

说明

相关属性含义前文已有详细说明，故不再赘述。

完整配置示例

以 HUSB311+ETA6973+AXP2202 的快充组合为例：

```
Allwinner BSP  --->  Device Drivers  --->
   I2C Drivers  --->
     <*> I2C Support for Allwinner SoCs
```

```
  USB Drivers  --->  USB Type-C Support --->
    <*> Allwinner USB Type-C support
    <*> Hynetek HUSB311 Type-C chip driver

  PMIC Drivers  ---> <*> BMU_EXT PMICs with I2C
          ---> <*> ETA6973 Power Supply Driver
```

Device Tree 配置

说明

相关属性含义前文已有详细说明，故不再赘述。

完整配置示例

以 HUSB311+ETA6973+AXP2202 的快充组合为例：

```
&twi5 {
  husb311: husb311@4e {
    compatible = "hynetek,husb311";
    reg = <0x4e>;
    interrupt-parent = <&r_pio>;
    husb311,intr_gpio = <&r_pio PL 13 GPIO_ACTIVE_LOW>;
    vbus-supply = <&reg_qc_drivevbus>;
    det_usb_supply = <&charger_power_supply>;
    /* aw,vbus-wakeup-quirk; */
    /* aw,port-reset-quirk; */
    status = "okay";

    ports {
      #address-cells = <1>;
      #size-cells = <0>;

      port@0 {
        reg = <0>;
        usbc2_role_sw: endpoint@0 {
          remote-endpoint = <&dwc3_0_role_switch>;
        };
      };
    };

    usb_con: connector {
      compatible = "usb-c-connector";
      label = "USB-C";
      data-role = "dual";
      power-role = "dual";
      try-power-role = "source";
      op-sink-microwatt = <1000000>;
      //typec-power-opmode = "default";
      //pd-disable;
      slow-charger-loop;
      sink-pdos = < PDO_FIXED(5000, 500, (0x1<<29)|(0x1<<25))
              PDO_VAR(5000, 5000, 2000)
              PDO_VAR(9000, 9000, 2000)
             /* PDO_VAR(12000, 12000, 3000)
              PDO_VAR(15000, 15000, 3000)
              PDO_VAR(20000, 20000, 2250)*/
            >;
      source-pdos = < PDO_FIXED(5000, 500, (0x1<<29)|(0x1<<25))
                PDO_VAR(5000, 5000, 1200) >;
      /* fixme */
```

```
        sink-vdos = <0xd1002e99 0x00 0x03110000>;
        sink-vdos-v1 = <0xd1002e99 0x00 0x03110000>;

        altmodes {
          #address-cells = <1>;
          #size-cells = <0>;

          altmode@0 {
            reg = <0>;
            svid = <0xff01>;
            vdo = <0xffffffff>;
          };
        };

        ports {
          #address-cells = <1>;
          #size-cells = <0>;
          port@0 {
            reg = <0>;
            usbc2_orien_sw: endpoint {
              remote-endpoint = <&usbdp_orientation_switch>;
            };
          };
        };
      };
    };
  };
};

&twi6 {
  qc0: qc@6b{
    compatible = "eta,eta6973";
    reg = <0x6b>;
    status = "okay";
    quick-charge,drive-vbus-en;
    wakeup-source;
    //watchdog_enable = <1>;

    charger_power_supply: charger_power_supply{
      compatible = "eta,eta6973-charger-power-supply";
      status = "okay";
      wakeup-source;
      det_battery_supply = <&bat_power_supply>;
      pmu_chargerpc_vol = <5000>;
      pmu_chargerpc_cur = <500>;
      pmu_chargerad_vol = <5000>;
      pmu_chargerad_cur = <2400>;
      battery_exist_sets = <0>;
      qc_det_gpio = <&r_pio PL 4 IRQ_TYPE_LEVEL_LOW>;
      qc_nce_gpio = <&r_pio PL 5 GPIO_PULL_DOWN>;
      bat_state_gpio = <&r_pio PL 7 GPIO_ACTIVE_LOW>;
    };
    regulator4: regulators@4 {
      reg_qc_drivevbus: drivevbus {
        regulator-name = "eta6973-drivevbus";
        regulator-enable-ramp-delay = <1000>;
      };
    };
  };

  pmu0: pmu@34 {
```

```
    compatible = "x-powers,axp2202";

    bat_power_supply: bat-power-supply {
      compatible = "x-powers,axp2202-bat-power-supply";
      param = <&axp2202_parameter>;
      status = "okay";
      det_qc_supply = <&charger_power_supply>;

      pmu_chg_ic_temp = <0>;

      pmu_battery_rdc= <170>;
      pmu_battery_cap = <5000>;
      pmu_runtime_chgcur = <2000>;
      pmu_suspend_chgcur = <3000>;
      pmu_shutdown_chgcur = <3000>;
      pmu_init_chgvol = <4350>;
      pmu_battery_warning_level1 = <15>;
      pmu_battery_warning_level2 = <0>;
      pmu_chgled_func = <0>;
      pmu_chgled_type = <0>;

      pmu_bat_temp_enable = <1>;
      pmu_jetia_en       = <1>;
      pmu_bat_charge_ltf = <1695>; //-5
      pmu_bat_charge_htf = <151>; //60
      pmu_bat_shutdown_ltf = <2125>; //-10
      pmu_bat_shutdown_htf = <131>; //65
      pmu_jetia_cool = <1361>; //0
      pmu_jetia_warm = <208>; //50
      pmu_jcool_ifall = <1>;//50%
      pmu_jwarm_ifall = <1>;//50%
      pmu_bat_temp_para1 = <4378>; //Murata -25
      pmu_bat_temp_para2 = <2682>; //-15
      pmu_bat_temp_para3 = <2125>; //-10
      pmu_bat_temp_para4 = <1695>; //-5
      pmu_bat_temp_para5 = <1361>;//0
      pmu_bat_temp_para6 = <1101>; //5
      pmu_bat_temp_para7 = <896>; //10
      pmu_bat_temp_para8 = <604>; //20
      pmu_bat_temp_para9 = <416>; //30
      pmu_bat_temp_para10 = <292>; //40
      pmu_bat_temp_para11 = <246>; //45
      pmu_bat_temp_para12 = <208>; //50
      pmu_bat_temp_para13 = <177>; //55
      pmu_bat_temp_para14 = <151>; //60
      pmu_bat_temp_para15 = <111>; //70
      pmu_bat_temp_para16 = <83>; //80

      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;
    };
```

```
    };
};
```

Android 配置

说明

[基础板型]： logan 下 device 配置
[方案基础板型]： Android 下 device 配置

说明

这里主要是介绍快充功能所需要的 BMU 驱动配置，TYPE-c 的配置详见《type-c 功能属性配置》章节。

配置步骤：

1. 打开快充配置

```
device/softwinner/[方案平台名]/[方案基础板型]/device-common.mk:
.....
CONFIG_AW_QUICK_CHARGER := true
.....
```

```
CONFIG_AW_QUICK_CHARGER <bool>
    打开Android层适配快充的功能
```

2. 添加产品配置

```
device/softwinner/[方案平台名]/AndroidProducts.mk:

PRODUCT_MAKEFILES := \
  $(LOCAL_DIR)/[方案基础板型]_[快充扩展后缀]_arm_go.mk \
.....
COMMON_LUNCH_CHOICES := \
  [方案基础板型]_[快充扩展后缀]_arm_go-user \
  [方案基础板型]_[快充扩展后缀]_arm_go-userdebug \

.....
```

```
PRODUCT_MAKEFILES 与 COMMON_LUNCH_CHOICES
    增加方案配置
```

3. 创建产品配置文件
4. 现根据基础板型复制一份其板型配置文件
5. 将配置文件重命名为 “[方案基础板型]_[快充扩展后缀]” 的配置文件

说明

注意名称需要和产品配置对应上。

```
device/softwinner/saturn/[基础板型]_[快充扩展后缀]_arm_go.mk:
.....
PRODUCT_NAME := [方案基础板型]_[快充扩展后缀]_arm_go
PRODUCT_DEVICE := [方案基础板型]
PRODUCT_BOARD := [基础板型]_[快充扩展后缀]
```

```
……
```

```
[基础板型]_[快充扩展后缀]_arm_go.mk
    复制已有板型配置，修改NAME/DEVICE/BOARD配置信息，以新增方案配置文件

PRODUCT_NAME
    当前方案名称

PRODUCT_DEVICE
    Android-device配置

PRODUCT_BOARD
    linux-device配置
```

4. 添加内核模块配置

考虑到开机时间要求，快充所需新增的 BMU 的 ko 大多都只在固定板型下才会加载，不放入平台下的 common 配置中。

以 BMU 组合为 axp519+axp2601 的快充方案为例：

```
device/softwinner/saturn/[方案基础板型]/system/vendor_ramdisk.modules:
……
+axp519_charger_power.ko
+axp2601_battery.ko
……
```

```
system/vendor_ramdisk.modules:
    加载当前快充方案所需的相关ko
```

6.1.3.3 供电扩展配置

说明

当前支持外挂 DCDC 功能的平台为：A133、A523、A537。

6.1.3.3.1 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
```

6.1.3.3.2 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_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，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 6-17: 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

外挂芯片型号	节点简称	基地址
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。

6.1.4 平台专用配置

6.1.4.1 gpio 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。

说明

该功能在 A523 及之后的平台不适用。

```
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。
```

6.1.4.2 BMU 充电配置

说明

该功能仅在 AXP519 和 ETA6973 上有效。

```
device/·····/board.dts:
charger_power_supply: charger_power_supply{
   ·····
   pmu_chargerpc_vol = <5000>;
   pmu_chargerpc_cur = <500>;
   pmu_chargerad_vol = <5000>;
   pmu_chargerad_cur = <2400>;
   ·····
   ·····
   pmu_runtime_chgcur = <1000>;
   pmu_suspend_chgcur = <2000>;
   pmu_shutdown_chgcur = <2000>;
   ·····
```

```
};
```

```
pmu_chargerpc_vol <u32>
    usb pc输入电压限制值
    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**
    **当前支持AXP：ETA6973、AXP519**

pmu_chargerpc_cur <u32>
    usb pc输入电流限制值
    单位为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：ETA6973、AXP519**

pmu_chargerad_vol <u32>
    usb adaptor输入电压限制值(vimdpm)
    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**
    **当前支持AXP：ETA6973、AXP519**

pmu_chargerad_cur <u32>
    usb adaptor输入电流限制值
    单位为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：ETA6973、AXP519**

pmu_runtime_chgcur <u32>
    运行时constant充电电流限制，默认2000mA
    单位为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：ETA6973、AXP519**

pmu_suspend_chgcur <u32>
    休眠时constant充电电流限制，默认3000mA
    单位为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：ETA6973、AXP519**

pmu_shutdown_chgcur <u32>
    关机时constant充电电流限制，默认3000mA
    单位为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：ETA6973、AXP519**
```

6.1.4.3 其他杂项配置

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>;
  ……
};
```

```
charger_power_supply: charger_power_supply{
    ……
    bmu_ext_save_charge_time = <0>;
    battery_exist_sets = <0>;
    bmu_ext_max_power = <0>;
    ……
};

device/……/uboot-board.dts:
&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。
    **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_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**

bmu_ext_save_charge_time <u32>
    充电多长时间后触发充电保护
    ETA6973/6974：10/5(单位h，是指完整充电流程)
    AXP519：12/24/48/72(单位h，是指恒流充电时间)
    **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：ETA6973、AXP519**

battery_exist_sets <u32>
    强制判断电池存在状态
    0：按照实际情况检测
    1：强制判断无电池
    2：强制判断有电池
    **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：ETA6973、AXP519**

bmu_ext_max_power <u32>
    高压充电芯片最高输入功率
    **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：ETA6973、AXP519**

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：

```
static int axp2202_bat_get_property(struct power_supply *psy,
                enum power_supply_property psp,
                union power_supply_propval *val)
    ...
    switch (psp) {
    ...
    case POWER_SUPPLY_PROP_CAPACITY_LEVEL: // customer modify
        ret = axp2202_get_bat_present(psy, val);
        if (ret < 0)
            return ret;

        if (val->intval) {
            ret = axp2202_get_soc(psy, val);
            if (ret < 0)
                return ret;

            if (val->intval == 100)
                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_FULL;
            else if (val->intval > 80)
                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_HIGH;
            else if (val->intval > bat_power->dts_info.pmu_battery_warning_level1)
                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_NORMAL;
            else if (val->intval > bat_power->dts_info.pmu_battery_warning_level2)
                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_LOW;
            else if (val->intval >= 0)
                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_CRITICAL;
            else
                val->intval = POWER_SUPPLY_CAPACITY_LEVEL_UNKNOWN;
            }
        else
            val->intval = POWER_SUPPLY_CAPACITY_LEVEL_UNKNOWN;
        break;
    case POWER_SUPPLY_PROP_STATUS:
        ret = axp2202_get_bat_status(psy, val);
        if (ret < 0)
            return ret;
        break;
    case POWER_SUPPLY_PROP_PRESENT:
        ret = axp2202_get_bat_present(psy, val);
        if (ret < 0)
            return ret;
```

```
        break;
    ...
}
```

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。

6.2.2 uboot 阶段-开机流程错误

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：

```
                    |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
```

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：

```
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;
    ...
}
```

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。

6.2.3 PMU 异常断电

问题现象：

系统运行过程中突然挂死，VDD-CPU、VDD-SYS 均掉电，疑似 PMIC 异常断电

问题分析：

PMIC 支持过温、过压、欠压等多种关机源，当出现关机源时 PMIC 会自动断电保护。PMIC 支持的关机源说明如下，仅供参考，详见 PMIC 芯片手册。

6.14.2.17 REG 21: PWROFF status

Bit	Description	R/W	Reset	Default
7	Die Over Temperature as POWEROFF Source 0: no 1: yes	RO	POR	0b
6	DCDC Over Voltage as POWEROFF Source 0: no 1: yes	RO	POR	0b
5	DCDC Under Voltage as POWEROFF Source 0: no 1: yes	RO	POR	0b
4	LDO over current as POWEROFF Source 0: no 1: yes	RO	POR	0b
3	VSYS Under Voltage as POWEROFF Source 0: no 1: yes	RO	POR	0b
2	Reserved	RO	/	0
1	Software configuration as POWEROFF Source 0: no 1: yes	RO	POR	0b
0	PWRON pin low for OFFLEVEL as POWEROFF Source 0: no 1: yes	RO	POR	0b

图 6-10: AXP2202 关机源

6.10.2.48 REG 51H：关机源指示

Bit	描述	R/W	复位	默认值
7	DCDC 过压关机	R	POR	0
6	DCDC 欠压关机	R	POR	0
5	DIE 过温关机	R	POR	0
4	DCIN 过压关机	R	POR	0
3	PS 欠压关机	R	POR	0
2	保留	R	/	0
1	PWRON 按键关机	R	POR	0
0	软件关机	R	POR	0

图 6-11: AXP8191 关机源

当出现 PMIC 异常断电时，可以按 RESET 键复位或先长按电源键关机后短按电源键开机，系统重新启动，u-boot 阶段打印出开关机源寄存器信息，可以确定 PMIC 异常断电原因。

以 AXP2202 为例说明，“onoff status: 0x20 = 0x1, 0x21 = 0x80” 即关机源寄存器 0x21 的值为 0x80，所以关机源是 “Die over temperature” 即 PMIC 芯片过温。

```
[00.551]CPU:  Allwinner Family
[00.553]Model: sun55iw3
I2C:  ready
[00.564]DRAM:  2 GiB
[00.568]Relocation Offset is: 75e98000
[00.613]secure enable bit: 0
[00.616]PMU: AXP2202
[00.618]BMU: AXP2202
[00.620][AXP2202] comm status : 0x0 = 0x20, 0x1 = 0x90
[00.625][AXP2202] onoff status: 0x20 = 0x1, 0x21 = 0x80
[00.630][AXP2202] reboot/charge status: 0xf0 = 0x0
```

问题原因：

通过 u-boot 阶段打印出开关机源寄存器信息，对照 PMIC 芯片手册，可以确定 PMIC 异常断电原因。

著作权声明

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





（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。