



Unisoc Confidential For hiar

Kernel 4.14 充电指导手册

文档版本	V1.2
发布日期	2020-10-27

版权所有 © 紫光展锐（上海）科技有限公司。保留一切权利。

本文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

Unisoc Confidential For hiar

紫光展锐（上海）科技有限公司



前言

概述

本文主要介绍充电的代码框架、基本功能、配置和一些常规的 debug 手段。

读者对象


本文档适用于充电驱动开发人员。

缩略语

缩略语	英文全名	中文解释
FGU	Fuel Gauge	电量计
OCV	Open Circuit Voltage	开路电压
UVLO	Under Voltage Lock Out	低电压锁定
PSY	Power Supply	电源

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 说明	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

变更信息

文档版本	发布日期	修改说明
V1.1	2020-01-19	第一次正式发布。
V1.2	2020-10-27	更新模板，优化内容。

关键字

Kernel、充电。

Unisoc Confidential For hiar

目 录

1 充电架构介绍.....	1
1.1 充电软件框架.....	1
1.2 Uboot 充电介绍.....	2
1.2.1 低电量检测.....	4
1.2.2 预充电.....	4
1.2.3 关机充电模式.....	4
1.2.4 POCV 检测.....	4
1.3 Kernel 充电介绍.....	5
2 硬件原理.....	10
2.1 电池在位检测.....	10
2.2 电池温度检测.....	10
2.3 库伦计电量统计.....	11
3 充电细分功能介绍.....	13
3.1 充电器类型识别.....	13
3.2 电池温度检测功能.....	14
3.3 内阻 - 温度补偿算法.....	14
3.4 温控策略.....	15
3.4.1 Jeita 温控策略.....	15
3.4.2 高低温停充策略.....	16
3.4.3 Jeita 温控策略开关.....	16
3.5 充电电流设置.....	17
3.6 关闭充电功能.....	17
3.7 容量 - 温度补偿算法.....	17
3.8 容量自学习功能.....	18
3.9 Fuel Gauge 介绍.....	19
3.9.1 初始容量.....	19
3.9.2 电量软件策略.....	20
4 项目配置介绍.....	24
4.1 bat: battery 节点.....	24
4.2 charger-manager 节点.....	25
4.3 PMIC_FGU 节点.....	27
4.4 Charger IC 节点.....	27
5 常规 debug 介绍.....	28
5.1 Healthd log.....	28
5.2 插拔充电器 log.....	28

5.3 限流信息	29
5.4 第一次开机	29
5.5 battery info.....	29
5.6 debug 节点	30

Unisoc Confidential For hiar

图目录

图 1-1 Kernel 充电架构图	1
图 1-2 Uboot 充电流程图	3
图 1-3 第一次开机 POCV 采集	5
图 1-4 Kernel 驱动关系图	6
图 2-1 电池在位检测	10
图 2-2 电池温度检测电路	11
图 2-3 FGU 模块内部原理	11
图 2-4 库仑计采样硬件电路	12
图 3-1 充电器软件识别流程图	13
图 3-2 NTC 电阻温度与阻值关系图	14
图 3-3 初始容量读取	20
图 3-4 电量更新	20
图 3-5 电量统计	21
图 3-6 低电量校准流程	22

表目录

表 3-1 充电温控策略说明	16
----------------------	----

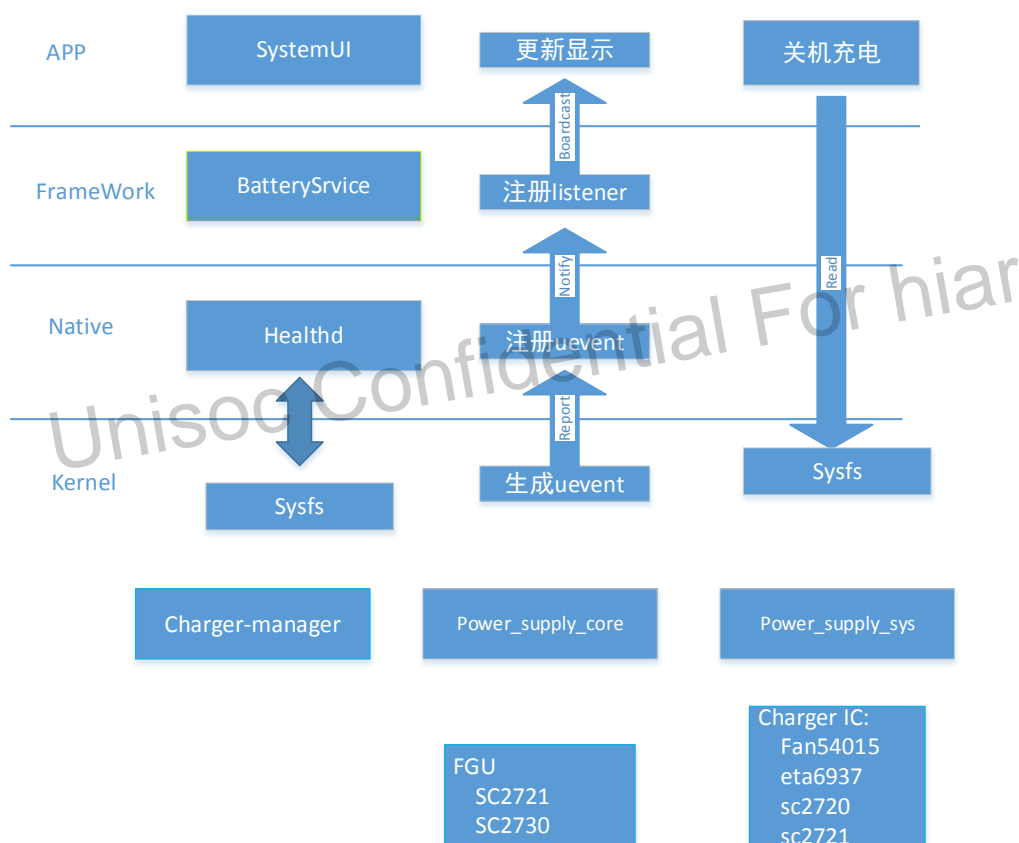
Unisoc Confidential For hiar

1 充电架构介绍

1.1 充电软件框架

Android 层充电架构如图 1-1 所示，充电设备驱动按照标准的 linux power supply 架构设计，通过 ueventd 守护进程 Healthd 上报相关信息，并将消息广播。

图1-1 Kernel 充电架构图



软件充电模块从上至下分为四层：APP 层、FrameWork 层、Native 层、Kernel 层。

- APP 层

该部分属于电量上报的最后的环节。其主要工作是：监听系统广播并对 UI 作出相应更新，包括电池电量百分比，充电状态，低电提醒，led 指示灯，异常提醒等。

- FrameWork 层

本层的 Battery 服务使用 Java 代码写成，运行在 Framework 中的 SystemServer 进程。该系统服务的主要作用是：监听电池信息变化消息，并将该消息以系统广播的形式转发至 Android 系统中各处。

- Native 层

Healthd 守护进程属于 Android Native 层的一个系统服务，负责接受 Kernel Driver 层上报的 uevent 事件，对电池信息和充电状态实时监控。

- Kernel 层

本层属于电池的驱动部分，由 Charger-manager 驱动、充电 IC 驱动、Fuel 驱动构成，负责与硬件交互，注册 Power supply 属性，并生成 uevent 上报 Native 层。包含充电状态管理、电量统计与更新。

Kernel 充电驱动文件：

- kernel4.14/drivers/power/supply/charger-manager.c
- kernel4.14/drivers/power/supply/sc27xx_fuel_gauge.c
- kernel4.14/drivers/power/supply/sc2720-charger.c
- kernel4.14/drivers/power/supply/sc2721-charger.c
- kernel4.14/drivers/power/supply/fan54015-charger.c
- kernel4.14/drivers/power/supply/sc2730_fast_charger.c

1.2 Uboot 充电介绍

Uboot 中主要实现如下几个功能：

- 低电量检测
- 预充电
- 关机充电模式选择
- POCV 检测

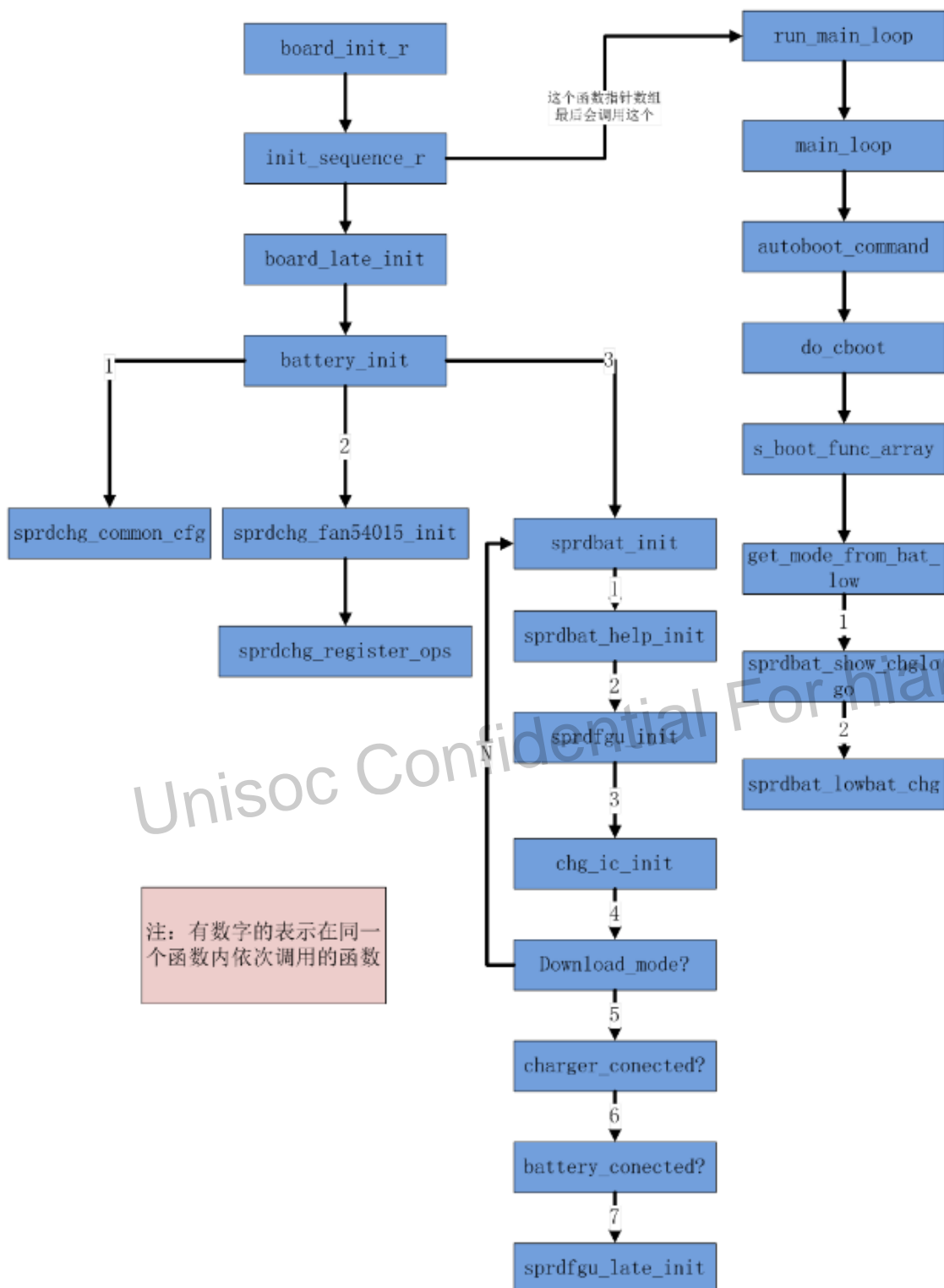
Uboot 充电驱动文件目录为：u-boot15/drivers/power/battery/

主要的驱动文件有：

- u-boot15/drivers/power/battery/sprd_battery.c
- u-boot15/drivers/power/battery/sprd_fgu.c
- u-boot15/drivers/power/battery/fan54015.c
- u-boot15/drivers/power/battery/sc2703_chg.c
- u-boot15/drivers/power/battery/eta6937_chg.c
- u-boot15/drivers/power/battery/sprd_chg_2720.c
- u-boot15/drivers/power/battery/sprd_chg_2721.c
- u-boot15/drivers/power/battery/sprd_chg_2730.c

以 FAN54015 charger ic 为例，开机及充电流程如图 1-2 所示。

图1-2 Uboot 充电流程图



- charger_connected

判断是否连接充电器，若是，则配置充电参数。

- Battery_connected

判断电池是否在位，若否，则关闭充电。

1.2.1 低电量检测

展锐系统启动分为 Uboot 和 Kernel 两个阶段，为了确保系统能安全稳定运行，必须满足如下条件：

- 不插充电器时，电池电压大于 3500mv 时，才允许进入 Kernel 阶段。否则显示当前电池电量低并直接关机。
- 插充电器时，电池电压大于 3300mv 时，才允许进入 Kernel 阶段。否则显示电池电量低且正在充电中，进入预充电阶段。

在此之前都认为电池处于低电量状态，需要在 Uboot 阶段进行预充电。

```
#define LOW_BAT_VOL    3500 /*phone battery voltage low than this value will not boot up*/  
#define LOW_BAT_VOL_CHG    3300    //3.3V charger connect
```

1.2.2 预充电

在预充电阶段，根据充电器的类型设定充电电流，充电电流范围为 450mA ~ 1000mA。

```
#define DCP_CHG_CUR    1000  
#define SDP_CHG_CUR    450
```

在预充电阶段，每 200ms 会轮询一次，若电池仍处于低电量状态，则继续预充电。

在预充电阶段，若电池电压连续 30 分钟低于 2000mV，则认为电池已损坏，停止充电。

1.2.3 关机充电模式

为了实现关机充电功能，每个 board 需要配置开关机充电启动项。

```
CBOOT_FUNC s_boot_func_array[CHECK_BOOTMODE_FUN_NUM] = {  
    /*4 get mode from charger*/  
    get_mode_from_charger,  
}  
}
```

在关机情况下，插入充电器，选择关机充电流程。系统只会启动 Kernel，不会启动 Android。

1.2.4 POCV 检测

POCV 是在系统开机上电之前，在电流极小的情况下，硬件量测电池端电压值，此时电流 < 1mA，电池端电压就是开路电压。

在拔插过电池的情况下开机，手机的初始电量值根据 OCV 电压表格得到。

OCV - CAP，开路电压 - 电池容量表。

系统无法直接获得电池的开路电压，只能通过公式计算：

开路电压 = 电池端电压 - 此时电流 * 此时内阻。

若电流很大，则会影响开路电压的准确性，从而影响初始电量的准确性。若拔插电池后，不插充电器按 power key 走开机流程，同时立马插入充电器，此时会因为插入充电器造成 POCV 电压不准。因此为了避免这种现象，在拔插电池且插着充电器第一次开机的时候会：

1. Power down device
2. 重启系统重新获取一次 POCV

图1-3 第一次开机 POCV 采集

```

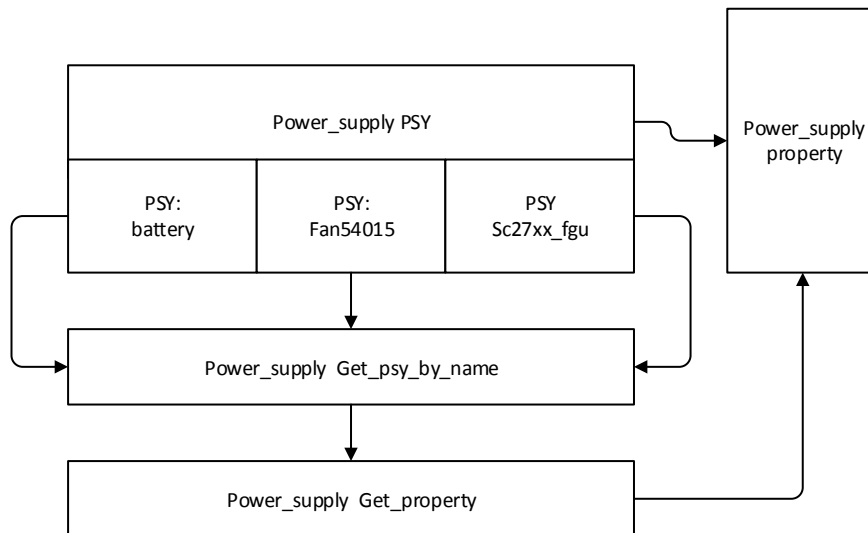
281
282 void sprdfgu_late_init(void)
283 {
284     if (charger_connected() && sprdbat_is_battery_connected()) {
285         if ((FIRST_POWERON == sprdfgu_poweron_type_read())
286             || (sprdfgu_rtc_reg_read() == 0xFF)) {
287             sprdfgu_rtc_reg_write(0xFF);
288             mdelay(1);
289             sprdfgu_poweron_type_write(NORMAL_POWERON);
290             printf("charge first poweron reset!!!!\n");
291             mdelay(3);
292             power_down_devices(0);
293         }
294     }
295
296     if (sprdfgu_rtc_reg_read() == 0xFF) {
297         printf("secend poweron !!!!!\n");
298         sprdfgu_poweron_type_write(FIRST_POWERON);
299         mdelay(1);
300         sprdfgu_rtc_reg_write(0xFF);
301     }
302 }
303

```

1.3 Kernel 充电介绍

Kernel 驱动分为三个部分，分别是 Charge Manger、Fuel Gauge、Charge IC。这三部分作为独立的设备驱动均注册到 Power-supply 中，每一个设备为单独的 PSY。PSY 之间可以通过 power supply 属性相互访问，这三者的关系可以通过图 1-4 来表示。虽然每个 PSY 都是独立的，但实际上 fuel-gauge 跟 charge-ic 是服务于 charge-manger，charge-manger 不需要了解硬件细节，仅通过获取相应功能的 PSY 设备实例，通过这个 PSY 的属性获取相应信息。

图1-4 Kernel 驱动关系图



Charger Manager

Charger Manager 是充电的控制策略层，主要负责：

- 修复并更新电量百分比。
- 充电流程管理 (charging, notcharging, discharging, full 充电状态转换管理)。
- 安全管理 (Ovp, Health, Charge Time out)。
- 温控管理 (Jeita 功能, thermal 限流)。
- 电池电量显示策略 (充放电曲线)。
- 电池容量管理 (容量自学习功能)。

Charger Manager 以 “battery” 名字注册至 Power Supply 架构，会读写 Fuel Gauge 和 Charger IC 的 Power supply 属性。

PSY 描述：

```
1 static const struct power_supply_desc psy_default = {
2     .name = "battery", //psy名称 sys/class/power_supply/battery
3     .type = POWER_SUPPLY_TYPE_BATTERY, //psy的type类型这个就是Battery
4     .properties = default_charger_props, //支持的属性集
5     .num_properties = ARRAY_SIZE(default_charger_props), //属性数量
6     .get_property = charger_get_property, //获取属性接口
7     .set_property = charger_set_property, //设置属性接口
8     .property_is_writeable = charger_property_is_writeable, //可写属性设置
9     .no_thermal = true, //thermal zone支持 (true 不支持)
10 };
```

支持属性：

```
1 static enum power_supply_property default_charger_props[] = {
2     POWER_SUPPLY_PROP_STATUS, //电池状态充电、放电、满电等
3     POWER_SUPPLY_PROP_HEALTH, //电池健康度如过压、过温
4     POWER_SUPPLY_PROP_PRESENT, //电池在位与否
```

```

5 POWER_SUPPLY_PROP_VOLTAGE_NOW, //电池当前工作电压
6 POWER_SUPPLY_PROP_CAPACITY, //当前百分比
7 POWER_SUPPLY_PROP_ONLINE, //充电器在位信息
8 POWER_SUPPLY_PROP_CHARGE_FULL, //显示满电条件
9 POWER_SUPPLY_PROP_CONSTANT_CHARGE_CURRENT, //充电电流设置
10 POWER_SUPPLY_PROP_INPUT_CURRENT_LIMIT, //输入限流
11 POWER_SUPPLY_PROP_CHARGE_COUNTER, //充电循环未使用
12 POWER_SUPPLY_PROP_CHARGE_CONTROL_LIMIT, //thermal 限流节点
13 };

```

可读写属性节点:

- sys/class/power_supply/battery/constant_charge_current
- sys/class/power_supply/battery/input_current_limit
- sys/class/power_supply/battery/charger.0/stop_charge

只读属性节点:

- sys/class/power_supply/battery/status
- sys/class/power_supply/battery/health
- sys/class/power_supply/battery/present
- sys/class/power_supply/battery/voltage_now
- sys/class/power_supply/battery/current_now
- sys/class/power_supply/battery/temp
- sys/class/power_supply/battery/capacity
- sys/class/power_supply/battery/online
- sys/class/power_supply/battery/charger_full
- sys/class/power_supply/battery/constant_charge_current
- sys/class/power_supply/battery/input_current_limit
- sys/class/power_supply/battery/online
- sys/class/power_supply/battery/charger_full
- sys/class/power_supply/battery/constant_charge_current
- sys/class/power_supply/battery/input_current_limit

Fuel Gauge

PMIC 部分主要负责:

- 库伦计电量积分
- 充电器类型获取
- 电池在位检测
- 开机电压管理
- 内阻 - 温度, 容量 - 温度等补偿算法

sc27xx_fuel_gauge 以 “sc27xx-fgu” 名字注册至 Power supply 架构, 提供属性给 Charger Manager 读写。

```

1 static const struct power_supply_desc sc27xx_fgu_desc = {
2     .name      = "sc27xx-fgu", //属性名称
3     .type      = POWER_SUPPLY_TYPE_UNKNOWN, //类型为unknown
4     .properties = sc27xx_fgu_props, //所支持的全部属性

```

```

5 .num_properties = ARRAY_SIZE(sc27xx_fgu_props),
6 .get_property = sc27xx_fgu_get_property,
7 .set_property = sc27xx_fgu_set_property,
8 .external_power_changed = sc27xx_fgu_external_power_changed,
9 .property_is_writeable = sc27xx_fgu_property_is_writeable,
10};

```

支持属性:

```

1 static enum power_supply_property sc27xx_fgu_props[] = {
2     POWER_SUPPLY_PROP_STATUS, //充电状态
3     POWER_SUPPLY_PROP_HEALTH, //电池状态
4     POWER_SUPPLY_PROP_PRESENT, //电池在位
5     POWER_SUPPLY_PROP_TEMP, //电池温度
6     POWER_SUPPLY_PROP_TECHNOLOGY, //电池化学成分默认锂离子电池
7     POWER_SUPPLY_PROP_CAPACITY, //电池容量
8     POWER_SUPPLY_PROP_VOLTAGE_NOW, //当前电池电压
9     POWER_SUPPLY_PROP_VOLTAGE_OCV, //电池开路电压
10    POWER_SUPPLY_PROP_CONSTANT_CHARGE_VOLTAGE, //恒压电压值
11    POWER_SUPPLY_PROP_CURRENT_NOW, //当前充电电流
12    POWER_SUPPLY_PROP_CURRENT_AVG, //平均电池电流
13    POWER_SUPPLY_PROP_ENERGY_FULL_DESIGN, //电池容量
14    POWER_SUPPLY_PROP_ENERGY_NOW, //当前电量积分
15    POWER_SUPPLY_PROP_CALIBRATE //电量校准
16};

```

属性节点:

- sys/class/power_supply/sc27xx-fgu/present
- sys/class/power_supply/sc27xx-fgu/temp
- sys/class/power_supply/sc27xx-fgu/capacity
- sys/class/power_supply/sc27xx-fgu/voltage_now
- sys/class/power_supply/sc27xx-fgu/voltage_ocv
- sys/class/power_supply/sc27xx-fgu/constant_charge_voltage
- sys/class/power_supply/sc27xx-fgu/current_now

Charger IC

Charger IC 主要负责以下具体内容:

- 打开/关闭充电
- 设置充电电流
- 设置截止充电电压点
- 打开/关闭 OTG

以 Fan54015 为例, 以 “fan54015_charger” 名字注册至 Power supply 架构。提供属性给 Charger Manager 读写。

PSY 描述:

```

1 static const struct power_supply_desc fan54015_charger_desc = {
2     .name = "fan54015_charger", //属性名称
3     .type = POWER_SUPPLY_TYPE_USB, //psy的type类型这个就是USB
4     .properties = fan54015_usb_props,

```



```
5 .num_properties = ARRAY_SIZE(fan54015_usb_props),
6 .get_property = fan54015_charger_usb_get_property,
7 .set_property = fan54015_charger_usb_set_property,
8 .property_is_writeable = fan54015_charger_property_is_writeable,
9 .usb_types = fan54015_charger_usb_types,
10 .num_usb_types = ARRAY_SIZE(fan54015_charger_usb_types),
11};
```

支持属性:

```
1 static enum power_supply_property fan54015_usb_props[] = {
2     POWER_SUPPLY_PROP_STATUS,
3     POWER_SUPPLY_PROP_CONSTANT_CHARGE_CURRENT, //CC 电流值
4     POWER_SUPPLY_PROP_INPUT_CURRENT_LIMIT, //输入限流
5     POWER_SUPPLY_PROP_ONLINE, //充电器在位信息
6     POWER_SUPPLY_PROP_HEALTH, //充电器状态
7     POWER_SUPPLY_PROP_USB_TYPE, //充电器类型
8 };
```

属性节点:

- sys/class/power_supply/fan54015_charger/constant_charge_current
- sys/class/power_supply/fan54015_charger/input_current_limit
- sys/class/power_supply/fan54015_charger/online
- sys/class/power_supply/fan54015_charger/status

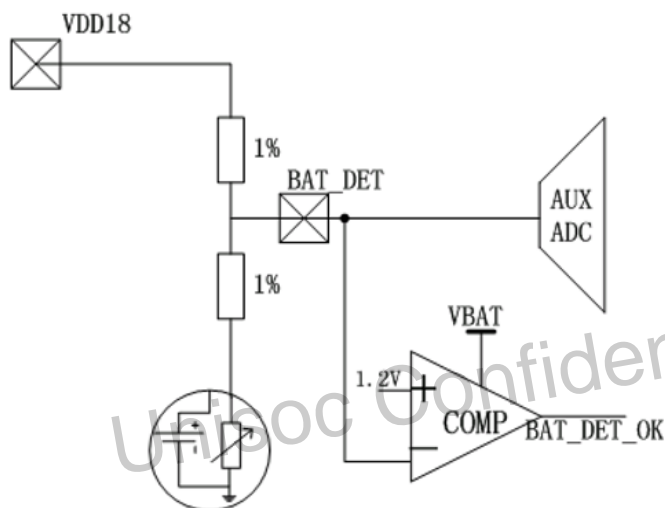
Unisoc Confidential For hiar

2 硬件原理

2.1 电池在位检测

目前只支持电压模式电池在位检测，不支持电流模式的电池在位检测。电压模式的电池在位检测图 2-1 所示。

图2-1 电池在位检测



在电压检测模式中，采用 AUX ADC 上拉 1.8V 电压，通过电阻分压来检测电池是否在位。

- 电池不在位，NTC 电阻在电池内部，如果电池不在位，比较器的负端输入就是 VDD18。 $VDD18 > 1.2V$ （比较值），比较器输出低电平，电池不在位。
- 电池在位，NTC 电阻接入电路，比较器负端输入是两个电阻对 VDD18 的分压值，电路设计时，电阻选择会确保电池在位，比较器负端分压值小于 1.2V，此时比较器输出高电平，电池在位。

2.2 电池温度检测

电池温度检测电路图如图 2-2 所示，图中电池内部 NTC 电阻是一个负反馈电阻。NTC 电阻值随着温度降低而变大，NTC 电阻分压值也就越大。电池温度检测是基于电阻分压原理，通过集成在 PMIC(SC2730, SC2721)中的 AUX ADC 采集 NTC 电阻上的分压值，得到当前温度，然后通过查找 T-V 表得到当前电池温度。

The diagram shows the SC2730 block with two pins. The top pin is connected to a voltage divider network. This network consists of a 47K resistor connected to VDD1V8, and an NTC thermistor connected to ground. The bottom pin of the SC2730 is also connected to ground.

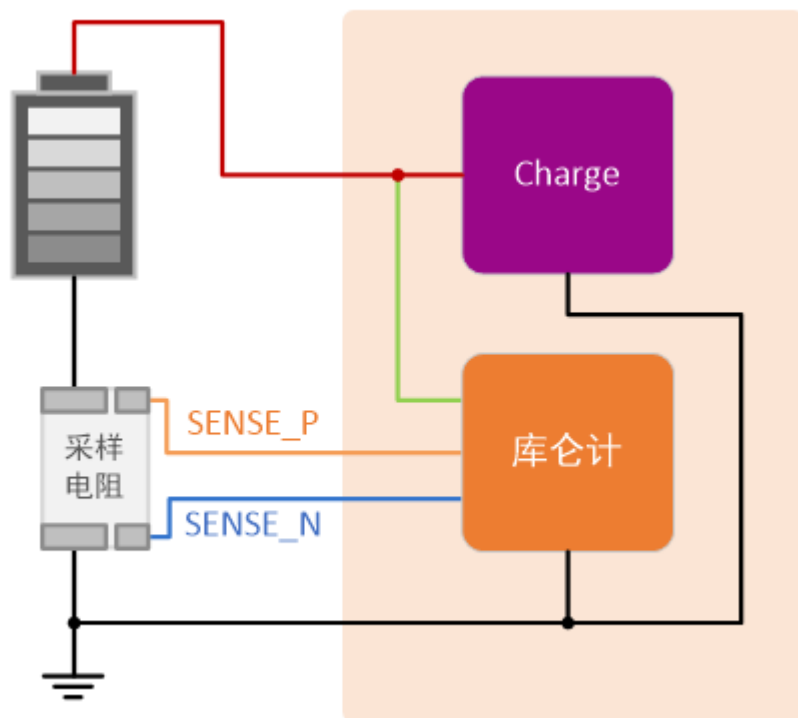
展锐各 PMIC 中集成的 FGU 主要由两个模块组成, FGU 模块和 FGU_ANA 模块, FGU_ANA 模块仅在开机时的 250ms 内运行, FGU 模块在 FGU_ANA 之后持续运行。

The diagram illustrates the internal architecture of the Fuel Gauge Unit (FGU). It shows the connection between the Application Processor (AP) and the FGU through the APB (Advanced Peripheral Bus). The APB is connected to the Registers Control block, which in turn manages the Coulomb Counter, Relax Counter, Inow, Vnow, and OCV blocks. The Coulomb Counter is also connected to the Registers Control. The Inow, Vnow, and OCV blocks are connected to the AD converter blocks. The AD converter blocks are connected to the MLX (Multiplexer) block, which is connected to the I and V inputs.

- APB 总线，用于 AP 通信。
- Relax Count，用于判断当前电池是否处于低功耗状态。
- Coulomb 库伦计，连续进行电量积分，用于电量统计。
- ADC，用于检测电池电压和电流。

文档版本 V1.2 (2020-10-27)

图2-4 库仑计采样硬件电路



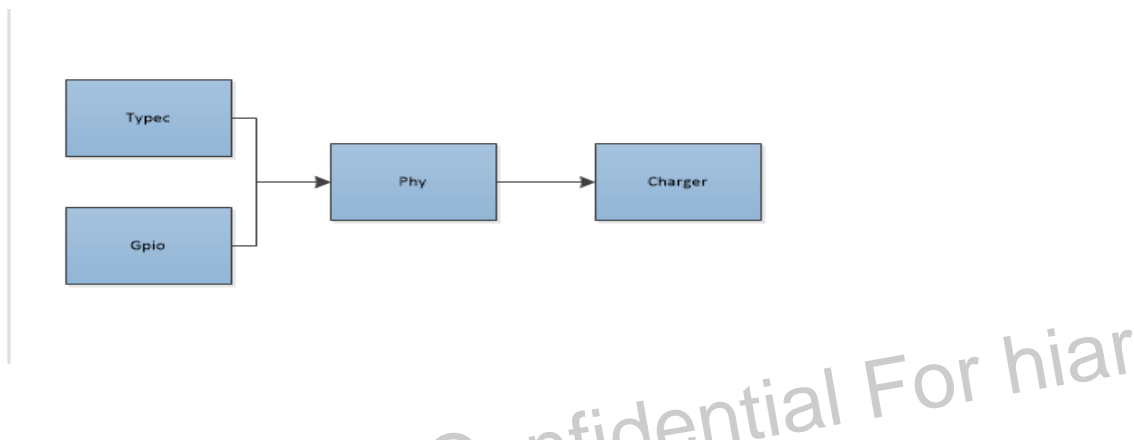
Unisoc Confidential For hiar

3 充电细分功能介绍

3.1 充电器类型识别

充电器软件识别流程图如图 3-1 所示。

图3-1 充电器软件识别流程图



充电器类型检测根据由 BC1.2 硬件自动实现，当插入充电器时，Vbus 由于电平变为高电平时会触发中断，同时触发 BC1.2 检测，Phy 模块去读 PMIC 上的 BC1.2 检测结果（把 BC1.2 的检测结果存在 PMIC 的寄存器上）。充电驱动在接收到 USB notify 通知后去读取检测结果。

在 BC1.2 标准中定义了三种充电器类型，加上不支持的充电器类型，将所有充电器类型分为四类，分别为 DCP、SDP、CDP 和 None-DCP。

- POWER_SUPPLY_USB_TYPE_DCP

标准 AC 充电。DCP 全称为 Dedicated Charging Port，即专用充电接口，该接口仅用于充电，无 USB 通信功能，DP 和 DM 通过 $<200\Omega$ 的电阻连在一起。

- POWER_SUPPLY_USB_TYPE_SDP

标准 USB 充电。SDP 全称为 Standard Download Port，即标准下行接口，是指符合现有 USB 规范的主机（HOST）上的下行 USB 接口，具有通信功能，SDP 内部 DP 和 DM 分别通过 RDP_DWN ($14.25K\Omega \leq RDP_DWN \leq 24.8K\Omega$) 和 RDM_DWN ($14.25K\Omega \leq RDP_DWN \leq 24.8K\Omega$) 下拉到地。根据 USB2.0 规范，当 USB 外设处于未连接 (un-connect) 或挂起 (suspend) 状态时，从 SDP 抽取的平均电流不超过 2.5mA，当外设处于 connect 并且未挂起未配置状态时，从 SDP 抽取的最大电流不超过 100mA。而当外设已经配置 (configured) 并且未挂起时，从 SDP 抽取的最大电流不超过 500mA。

- POWER_SUPPLY_USB_TYPE_CDP

大功率主板如笔记本充电。CDP 全称为 Charging Downstream Port，即充电下行接口，是指兼容 USB 规范，同时又针对 USB 充电进行优化的下行 USB 接口，可以是主机上的 USB 接口，也可以是 USB HUB 上的接口，可以提供最大至 1.5A 的供电能力。

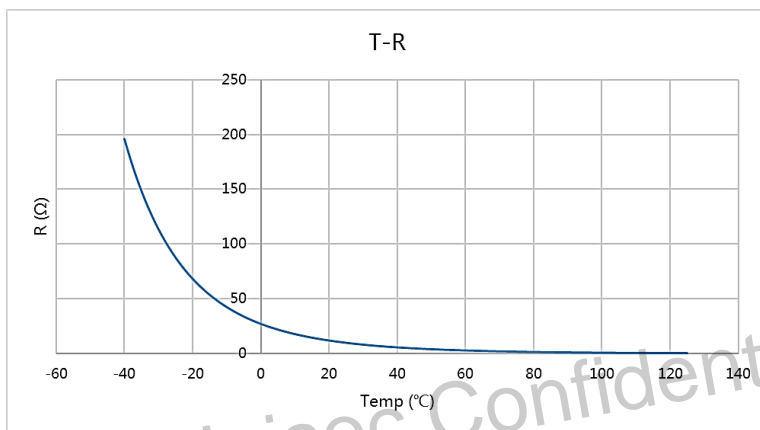
- POWER_SUPPLY_USB_TYPE_UNKOWN

非标准充电。除上述三种标准充电器类型外，其余充电器类型均为 None-DCP，如 DP DM 均悬空的充电器。

3.2 电池温度检测功能

在电池制造时其内部均集成了 NTC 电阻，其 T-R 映射关系图 3-2 所示。

图3-2 NTC 电阻温度与阻值关系图



基于电池温度检测的 T-R 映射关系，结合温度检测电路图，得到该检测电路的 T-V 映射关系，从而可以根据 ADC 采到的 NTC 电阻分压值 V 计算出此时电池内部的温度 T。

```
voltage-temp-table = <1095000 800>, <986000 850>, <878000 900>,
    <775000 950>, <678000 1000>, <590000 1050>,
    <510000 1100>, <440000 1150>, <378000 1200>,
    <324000 1250>, <278000 1300>, <238000 1350>,
    <204000 1400>, <175000 1450>, <150000 1500>,
    <129000 1550>, <111000 1600>, <96000 1650>;
```

3.3 内阻 - 温度补偿算法

电池内阻是根着温度变化而变化的。系统初始会定义常温下电阻值 factory-internal-resistance-micro-ohms。

```
factory-internal-resistance-micro-ohms = <320000>;
resistance-temp-table = <20 100>, <10 243>, <(-10) 468>;
```

其中：

- factory-internal-resistance-micro-ohms: 常温下电阻值, 单位 u 欧。
- resistance-temp-table: 真实电池实测值, 代表不同温度下电池内阻与常温下内阻百分比值, 如:
 - <20 100>: 20 度下电池值是常温下的 100%, 即 320m 欧。
 - <(-10) 468>: -10 度下电池值是常温下的 468%, 即 $320 * 468\% = 1497.6\text{m}$ 欧。

当获取某个温度下的电池内阻时:

1. 根据 T-V 表查得当前电池温度下的电池内阻 internal_resist。
2. 根据当前温度与 resistance-temp-table, 利用线性插值方法计算得到当前温度下电池内阻与常温下电池内阻的百分比 (resistance)。
3. 当前温度下电池内阻 = internal_resist * resistance。

3.4 温控策略

3.4.1 Jeita 温控策略

Jeita 策略是指根据不同的电池温度和不同的充电器类型设置对应的充电电流和充电截止电压, Jeita 温控策略限流的是 CONSTANT_CHARGE_CURRENT 值。具体设置值在 Board 对应的 dts 文件里配置, 如下表所示。

因为 BC1.2 支持 3 种不同类型的充电器和非标准充电器的检测, 因此 JEITA Table 对应也有 4 个。

```
cm-dcp-jeita-temp-table = <1000 1030 0 4400000>, <1150 1180 700000 4400000>,
                           <1450 1420 1150000 4400000>, <1600 1570 700000 4100000>;
cm-sdp-jeita-temp-table = <1000 1030 0 4400000>, <1150 1180 500000 4400000>,
                           <1450 1420 500000 4400000>, <1600 1570 500000 4100000>;
cm-cdp-jeita-temp-table = <1000 1030 0 4400000>, <1150 1180 700000 4400000>,
                           <1450 1420 1150000 4400000>, <1600 1570 700000 4100000>;
cm-unknown-jeita-temp-table = <1000 1030 0 4400000>, <1150 1180 500000 4400000>,
                              <1450 1420 500000 4400000>, <1600 1570 500000 4100000>;
```

例如 Jeita Table 中元素<1150 1180 700000 4400000>解析:

- 1150 1180
代表温度值, 此值减 1000 后再除以 10 就代表摄氏温度值。如 1150 代表 15 度。 $(1150 - 1000) / 10 = 15$ 度。
- 700000
代表最大限流值, 此值不大于 charge-sdp-current-microamp、charge-cdp-current-microamp、charge-dcp-current-microamp、charge-unknown-current-microamp 中的电流值。
- 4400000
代表截止充电电压值, 该温度下的电压值需要根据 constant_charge_voltage_max_microvolt 电池电压来配置。若小于此值, 可能会出现无法充满电的情况。

表3-1 充电温控策略说明

电池温度 T(°C)	充电电流 ICC(mA)	充电截止电压 VEOC(V)
$T \leq 0$	0	4400
$0 < T < 15$	700	4400
$15 \leq T < 45$	2000	4400
$45 \leq T < 60$	700	4100
$T \geq 60$	0	4100

3.4.2 高低温停充策略

1. 根据电池温度停充

根据 Jeita 表策略，当电池温度：

- 小于 Jeita 表最小温度时，停充
<1000 1030 0 4400000>
- 大于 Jeita 表最大温度时，停充
<1600 1570 500000 4100000>

2. 根据板温判断停充

```
cm-battery-cold = <200>; //电池温度 CM_EVENT_BATT_COLD 阈值 temp_min
cm-battery-cold-in-minus; // 负温标志
cm-battery-hot = <800>; //电池温度 CM_EVENT_BATT_OVERHEAT 阈值 temp_max
cm-battery-temp-diff = <100>; // 修正异常时COLD 和 OVERHEAT 阈值 temp_diff
```

若有定义"cm-thermal-zone"则使用 thermal zone 温度来做判断，若没有定义则使用电池温度来做判断，接口为 cm_check_thermal_status。

- 初始时，upper_limit = temp_ma, lower_limit = temp_min
- 若当前温度大于 temp_max 或者小于 temp_min 则停充
- 若当前状态为 Emergency_stop，则用 temp_diff 修改
upper_limit -= temp_diff
lower_limit -= temp_diff

3.4.3 Jeita 温控策略开关

在某些测试场景，如稳定性测试，monkey 测试等，由于场景耗电大，电池温度升高，导致充电被正常限流，最终电池没电关机，但这可能会影响测试。因此增加了关闭 Jeita 功能的节点，需要的时候关闭 Jeita。

```
1 sysfs_attr_init(&charger->attr_jeita_control.attr); //创建属性
2
3 charger->attr_jeita_control.attr.name = "jeita_control"; //属性名称
4
5 charger->attr_jeita_control.attr.mode = 0644; //属性权限
```



```
charger->attr_jeita_control.show = jeita_control_show; //属性读取接口
charger->attr_jeita_control.store = jeita_control_store; //属性设置接口
```

默认情况下 Jeita 功能是开启的，按需求是否将 Jeita 功能关闭。

3.5 充电电流设置

除了 Jeita 会设置充电电流限制等，Thermal 温控策略也会限制充电电流。

Thermal 温控策略在插上充电器之后，通过 POWER_SUPPLY_PROP_INPUT_CURRENT_LIMIT 节点读取最大限流值，而后根据不同的温度，限制不同的电流。Thermal 设置的是 INPUT_CURRENT_LIMIT，如果 INPUT_CURRENT_LIMIT 设置失败才会去设置 CONSTANT_CHARGE_CURRENT。设置 CONSTANT_CHARGE_CURRENT 时需考虑当前 Jeita 的温度值，两者取最小。

3.6 关闭充电功能

充电在必要的时候关闭，充电器不在位情况下充电一定是关闭的，但并不是充电器插着就一定打开充电，一些异常状态下为了保护设备会关闭充电，如高温高压等。还有在一些模式下也需要关闭充电，保证流程正确进行，如某些工厂模式。因此充电就有了 4 个状态，并将状态放在 POWER_SUPPLY_PROP_STATUS 属性里。

- Discharging: 放电
- Charging: 充电
- NotCharging: 充电异常停止
- Full: 满电

charger-manger 提供了应用层主动停止充电的接口，如果某些应用希望主动停止充电可以通过 psy 提供的 stop_charge 属性进行控制。

```
1 sysfs_attr_init(&charger->attr_stop_charge.attr); //创建属性
2 charger->attr_stop_charge.attr.name = "stop_charge"; //属性名称
3 charger->attr_stop_charge.attr.mode = 0644; //属性权限
4 charger->attr_stop_charge.show = charger_stop_show; //属性读取接口
5 charger->attr_stop_charge.store = charger_stop_store; //属性设置接口
```

3.7 容量 - 温度补偿算法

不同温度下电池容量定义如下，此参数为电池实测值：

```
capacity-temp-table = <25 100>, <10 97>, <(-10) 60>;
```

其中：

- 第一个参数表示温度。
- 第二个参数表示当前温度下电池容量与常温下电池容量百分比。

计算某个温度下的电容量算法为：

1. 根据库伦计统计计算得到常温下电池容量 (cap)。
2. 根据当前温度与 capacity-temp-table 表, 利用线性插值方法计算得到当前电池温度下电池容量与常温下电池容量的百分比 (temp_cap)

当前温度下电池容量 = ((*cap + temp_cap - 1000) * 1000, temp_cap)

3.8 容量自学习功能

容量自学习机制发生在电池电压和电流满足极低的情况 (类似已经放空状态就开始自学习功能), 当满足电池满电状态时结束自学习功能, 用这个实际的容量值替代电池标准容量值记录在文件系统里, 后续 Fuel Gauge 计算电量百分比时会用到这个容量值, 使电量百分比显示更加精准。

容量自学习公式:

$$Soc1 = \frac{Q1}{Q_{max}} \quad Soc2 = \frac{Q2}{Q_{max}}$$

$$Q_{max} = \frac{Q1 - Q2}{|Soc1 - Soc2|} = \frac{\Delta Q}{|Soc1 - Soc2|}$$

当电池工作电流足够小时, 电池端电压近似等于电池开路电压, 根据 OCV Table 就能准确的定位电池容量。容量自学习的原理为:

1. 记录 T1 时刻电池 OCV 电压 V1 与库仑计计数值容量 Q1。通过电池 OCV 电压查表得到当前电量值 Soc1。
2. 经过一段时间充放电, 记录 T2 时刻电池 OCV 电压 V2 与库仑计计数值容量 Q2。通过电池 OCV 电压查表得到当前电量值 Soc2。
3. 根据上述公式计算出当前电池的真实容量。

因为 OCV Table 在中间区域电压密度大, 且软件读取计算 OCV 电压在高密度区域误差大, 因此选择在低电量区域电量密度低去记录 Q1。若 delta Q 和 delta SOC 都很小误差会很大, 因此选择在充满电后, 即电池真实容量接近 100% 时, 将库仑计计数值记录为 Q2。根据以上几个数据就可以得到电池真实有效容量。

$$Q_{max} = (Q2 - Q1) * 100 / (100 - C1)$$

一般采用低电量开始作为初始采集点, 并严格规定了环境温度、电流、电压条件, 只有符合规定的容量自学习才认为是有效的自学习。

容量自学习开关在 dts 中配置, 有配置则打开, 没有配置则不打开。

"cm-capacity-track"

容量自学习的结果保存在如下文件中, 只有成功打开文件才能进入追踪流程。

"/mnt/vendor/battery/calibration_data/.battery_file"

初始时, 状态为 CAP_TRACK_INIT, 若能正确打开 "battery_file", 则将状态切换到 CAP_TRACK_IDLE 状态, 并进入追踪流程。否则将状态切换到 CAP_TRACK_ERR。

容量追踪功能开始条件，分为关机和开机两种情况：

- 在关机模式下插入 usb 线时，读到 $pocv < 3500\text{mV}$ 并且 $ocv < 3650\text{mV}$ 。
- 在开机模式下满足 $ocv < 3650\text{mV}$ 并且 cur 绝对值 $< 30\text{mA}$ 。

以上两种情况满足容量追踪开始条件，将 `track state` 设置为 `CAP_TRACK_UPDATING` 状态，以 15s 为周期 `monitor workqueue` 会检测是否达到容量追踪结束条件，不满足容量追踪条件则将 `track state` 设置为 `idle` 状态继续等待满足开始条件再开始。

检测 `battery voltage` 和 `cur`，如果在 30 小时内同时满足以下条件，则认为追踪结束。

- `battery voltage` > 软件截止电压-5mV
- `cur` < 软件截至电流+5mA
- 追踪前后电池电量差异小于 50%

停止容量追踪功能条件为：

- 状态为 `CAP_TRACK_ERR` 时停止追踪
- 开机模式下，若初始电池电量大于 20%，则停止追踪

```
#define CM_TRACK_START_CAP_THRESHOLD    200
```

- 当温度超过范围时也停止追踪

```
#define CM_TRACK_HIGH_TEMP_THRESHOLD    450
```

```
#define CM_TRACK_LOW_TEMP_THRESHOLD     150
```

- 如果超过 30 小时没有追踪完成，则结束此次追踪，将状态设为 `CAP_TRACK_IDLE` 状态。并等待下次满足开始条件时重启电量追踪。

```
#define CM_TRACK_TIMEOUT_THRESHOLD      108000
```

- 达到追踪结束条件时，追踪前后电池电量差异不小于 50%，则追踪失败，将状态设为 `CAP_TRACK_IDLE` 状态。并等待下次满足开始条件时重启电量追踪。

3.9 Fuel Gauge 介绍

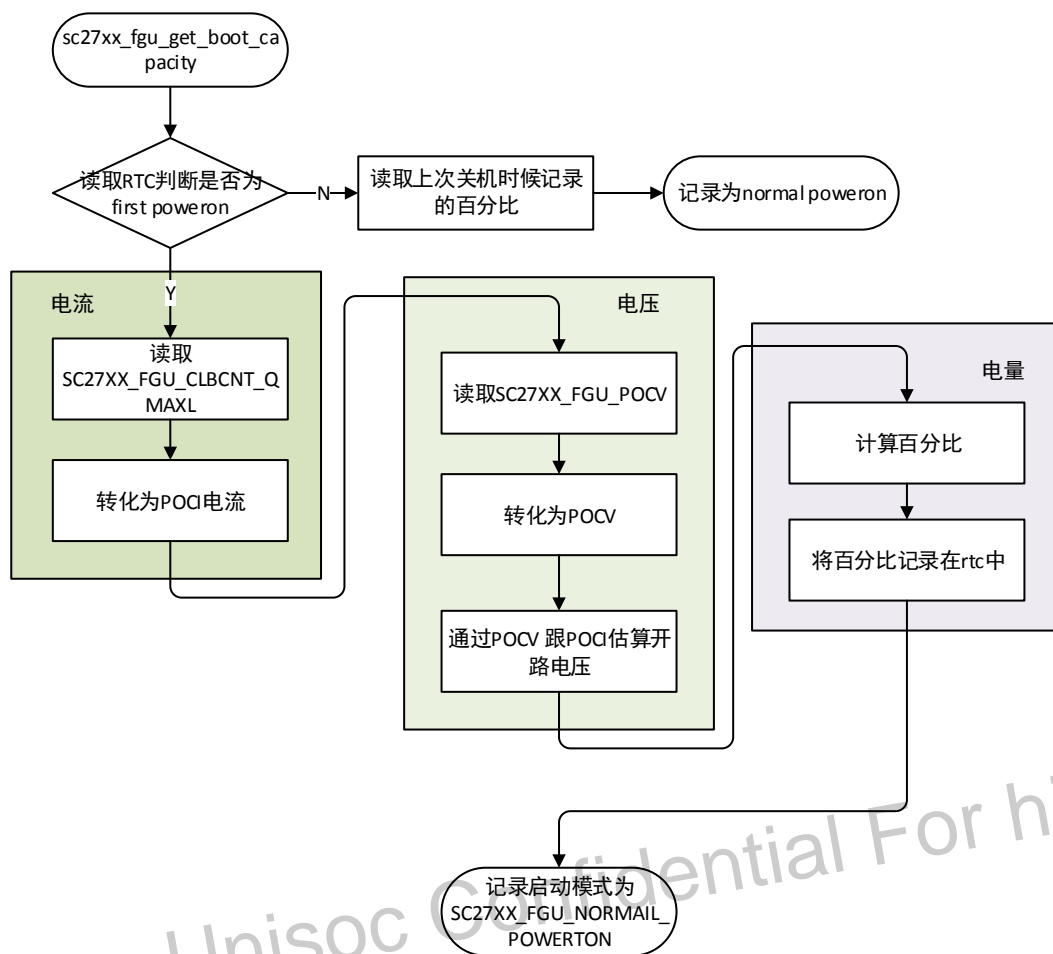
3.9.1 初始容量

库仑计采用电流积分来计算容量变化，但是如果更换电池的情况下是无法获取电池的电量积分，这样就需要用电压模式来辅助定位初始容量。目前采用库仑计加 OCV Table 的混合模式进行电量统计。在开机启动系统上电之前，电源管理芯片可以在低功耗状态下记录电池的电流（POCI）和电压（POCV），FGU 模块会采 POCV 电压（因为系统启动前 250ms 系统功耗较小，采的 POCV 值近似于电池开路电压值），再根据这个电压值查 OCV Table 表得到电池的容量。这个容量作为初始容量，把它做为开机百分比，请参见 1.2.4 POCV 检测。

如果未拔插过电池，开关机前后，UI 显示的电池容量必须是连续的，由于查 POCV 表得到的电量值有较大误差，因此在此过程中使用库仑计电量积分方法可以使得电量显示连续。电流采样周期为 500ms，记录 500ms 的平均值。

由此就有了两种初始容量：FIRST POWERON / NORMAL POWERON，这两个模式是通过 `rtc` 寄存器里面保存的数据来区分，如果掉电 `rtc` 会恢复默认值，开机后启动模式为 First poweron。如果 reboot 重启等非掉电重启，`rtc` 里面会保存 reboot 前容量值跟 Normal Power 标志。初始容量读取流程如图 3-3 所示。

图3-3 初始容量读取



3.9.2 电量软件策略

电量更新过程如图 3-4 所示，电量的统计过程图 3-5 所示。

图3-4 电量更新

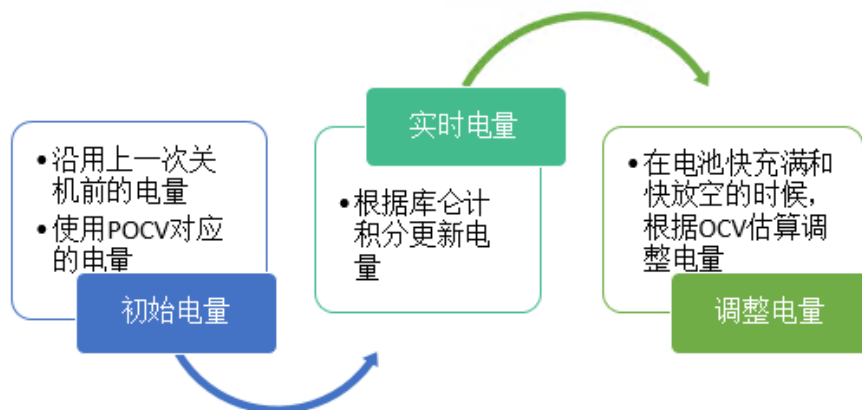
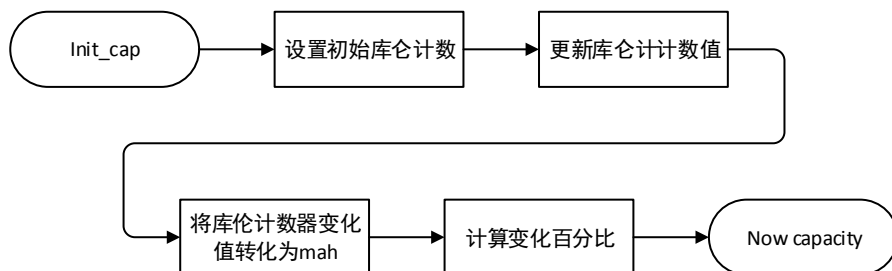


图3-5 电量统计



Charger Manager 设置启动了一个 15s 的周期性 Timer，在充电过程中定时唤醒系统来监视电量百分比和充电状态。

放电过程中允许系统休眠，休眠过程中 FGU 模块会持续统计容量，电量更新仅在系统唤醒的瞬间执行。由于存在 ADC 精度、环境温度、硬件准确性等因素可能影响到电量显示，软件 Charger Manager 加入一些机制来保证电量显示与实际容量匹配。

电量更新有以下几大原则策略：

- 充电时，在充入值小于消耗值的情况下，电量允许下降。
- 放电时，电量不允许上升。
- 充电时，99%维持 25 分钟后电量跳到 100%。
- 电量过低执行关机。
- 95% ~ 100%区间按实际电量消耗更新电量值。
- 满电量时校准，低电量时校准。

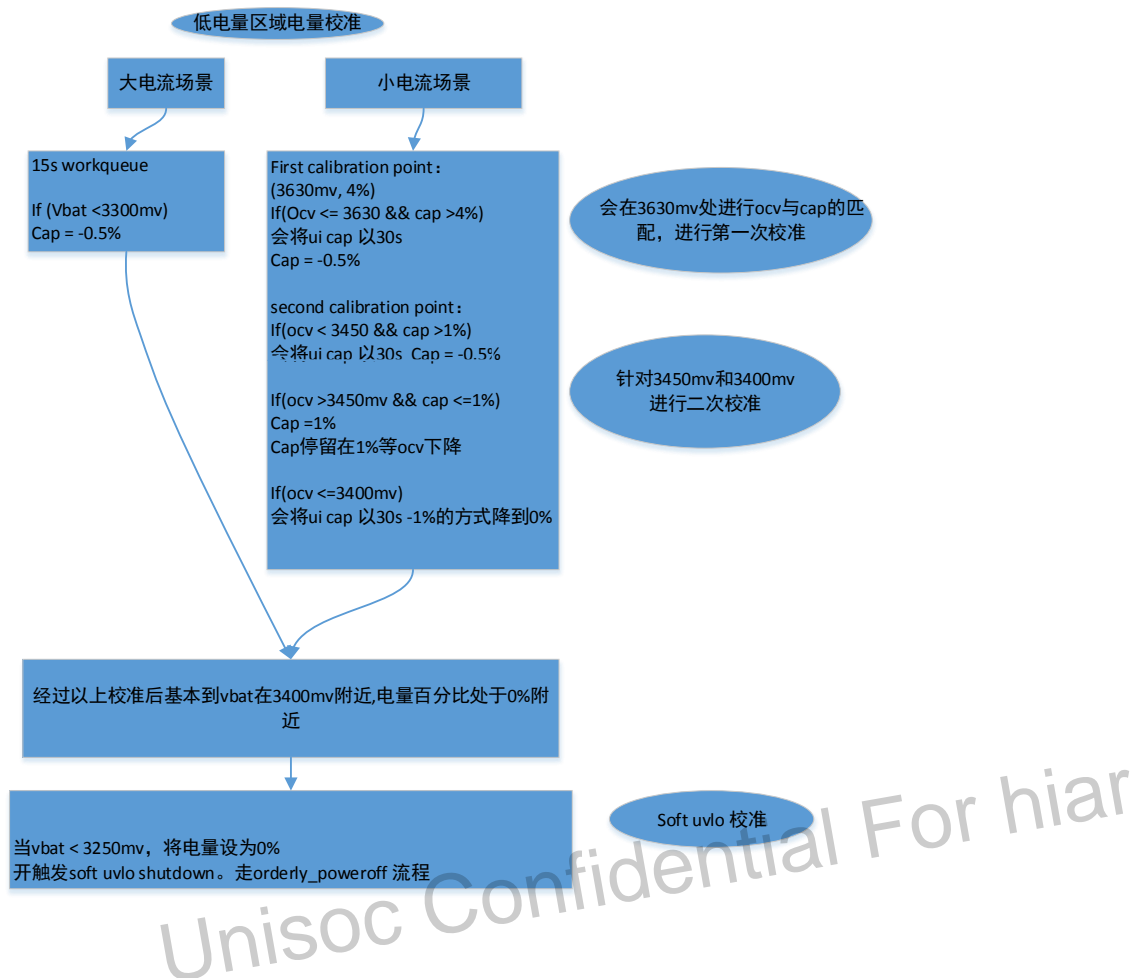
3.9.2.1 满电量及满电量校准策略

软件满电量条件：电池电流小于 `cm-fullbatt-current`、电池端电压高于 `cm-fullbatt-voltage`。此时将 FGU 电量校准为 100%，同时将充电状态设为 `force set full` 状态，将 UI 电量值往 100%更新。

3.9.2.2 低电量校准策略

低电量校准流程如图 3-6 所示。

图3-6 低电量校准流程



根据电池的 OCV 能量密度区域，在低电量区选择两个校准点，选择 4%和 1%两个校准点。

```

first-calib-voltage = <3630000>;
first-calib-capacity = <40>;
voltage-min-design-microvolt = <3450000>;
  
```

4%校准点电压和电量值在 DTS 中配置，1%校准点在 DTS 中只配置电压值，对应电量值通过查表 OCV 得到，OCV 表参见 4.1 bat: battery 节点中 OCV 到电量映射表。

- 当电池真实容量小于等于 4%时，如果 UI 电量值大于 4%，则快速将 UI 显示电量值校准为 4%。
- 当电池真实容量小于等于 1%时，如果 UI 电量值大于 1%，则快速将 UI 显示电量值校准为 1%。
- 当电池真实容量大于 1%时，如果 UI 电量值小于等于 1%，则将 UI 显示电量值维持在 1%，直到电池真实电量降到 1%。

3.9.2.3 UVLO 策略

电池电压太低时，有些模块在低电压情况下无法正常工作，会出现不可预知风险。为了保证系统稳定性，软件如果检测到电池电压低于系统工作门限电压 3050mv，则会触发软件 UVLO 流程，主动通知上层发起关机命令。

当电池端电压小于 UVLO 电压阈值时：

1. 将 UI 电量值设为 0%。
2. 走软件关机流程。

3.9.2.4 高电区放电策略

为了使高电量区，在存在误差的情况下，100%~95%电量不会下降太快。100%~95%电量区间：

- 如果不存在误差，电量实际下降 0.8%时，更新 0.8%电量值。
- 如果存在误差，电量实际下降 0.8%时，更新 1%电量值。
- 直到电量下降到 95%时，恢复到正常的电量显示。

3.9.2.5 电量再分配

如果有需要使某个电量区间电池电量比较耐用，且可在 DTS 中配置：

```
cm-cap-remap-table = <1 2 3>, <2 3 2>, <98 99 2>, <99 100 3>;
```

- <1 2 3>代表 1%~2%电量区间电量是正常情况下的 3 倍。
- <98 99 2>代表 98%~99%电量区间电量是正常情况下的 2 倍。

电量区间与电量值可以根据需要配置：

- 要使得 100%电量比较耐用，是正常电量的 5 倍，只需要配置：<99 100 5>。
- 要使得 50%~60%电量比较耐用，是正常电量的 2 倍，只需要配置：<50 60 2>。

Unisoc Confidential For hiar

4 项目配置介绍

4.1 bat: battery 节点

```
bat: battery {
    compatible = "simple-battery";

    charge-full-design-microamp-hours = <3690000>; //电池容量uAh

    charge-term-current-microamp = <120000>; //截止充电电流

    constant_charge_voltage_max_microvolt = <4400000>; //截止充电电压

    factory-internal-resistance-micro-ohms = <147000>; //电池内阻

    voltage-min-design-microvolt = <3450000>; //电池内阻值

    ocv-capacity-celsius = <20>; // ocv-capacity-table-0代表是在哪个温度下测量

    ocv-capacity-table-0 = <4380000 100>, <4317000 95>, <4258000 90>, //OCV到电量映射表
        <4200000 85>, <4145000 80>, <4092000 75>,
        <4047000 70>, <3990000 65>, <3955000 60>,
        <3900000 55>, <3861000 50>, <3834000 45>,
        <3813000 40>, <3796000 35>, <3783000 30>,
        <3770000 25>, <3752000 20>, <3730000 15>,
        <3698000 10>, <3687000 5>, <3400000 0>;

    voltage-temp-table = <1095000 800>, <986000 850>, <878000 900>, //内阻-电压映射表
        <775000 950>, <678000 1000>, <590000 1050>,
        <510000 1100>, <440000 1150>, <378000 1200>,
        <324000 1250>, <278000 1300>, <238000 1350>,
        <204000 1400>, <175000 1450>, <150000 1500>,
        <129000 1550>, <111000 1600>, <96000 1650>;

    //电池容量 - 温度补偿表
    capacity-temp-table = <45 100>, <25 100>, <10 97>, <0 95>, <(-10) 82>, <(-20) 62>;

    //电池内阻值 - 温度补偿表
    resistance-temp-table = <45 100>, <25 100>, <10 483>, <0 680>, <(-10) 789>, <(-20) 816>;

    //不同充电器类型充电限流值
    charge-sdp-current-microamp = <500000 500000>;
    charge-dcp-current-microamp = <2000000 3000000>;
}
```



```
charge-cdp-current-microamp = <1150000 1150000>;  
charge-unknown-current-microamp = <500000 500000>;  
charge-fchg-current-microamp = <3250000 3000000>;  
};
```

4.2 charger-manager 节点

```
charger-manager {  
    compatible = "charger-manager";  
    cm-name = "battery";  
    cm-poll-mode = <2>; /* "_cm_monitor" 轮询模式 */  
    cm-poll-interval = <15000>; /* "_cm_monitor" 轮询时间间隔 */  
    cm-battery-stat = <2>; /* 电池在位检测方法，电压法 */  
  
    cm-fullbatt-vchkdirp-ms = <30000>; /* 充满电后，检查复充条件的周期 */  
    cm-fullbatt-vchkdirp-volt = <60000>; /* 满电后复充电压条件 */  
    cm-fullbatt-voltage = <4350000>; /* 软件满电电压判断阈值，必须配置 */  
    cm-fullbatt-current = <120000>; /* 软件满电电流判断阈值，必须配置 */  
    cm-fullbatt-capacity = <100>; /* 电池满电时百分比 */  
  
    cm-num-chargers = <1>; /* charger ic 数量 */  
    cm-chargers = "fan54015_charger"; /* charger ic 名字 */  
    cm-fuel-gauge = "sc27xx-fgu"; /* fgu 名字 */  
  
    /* in deci centigrade */  
    cm-battery-cold = <200>; /* 电池温度CM_EVENT_BATT_COLD阈值 */  
    cm-battery-cold-in-minus; /* 负温标志 */  
    cm-battery-hot = <800>; /* 电池温度CM_EVENT_BATT_OVERHEAT阈值 */  
    cm-battery-temp-diff = <100>; /* 修正异常时COLD 和 OVERHEAT阈值 */  
  
    /* Allow charging for 6hr */  
    cm-charging-max = <21600000>; /* 允许连续充电的最长时间 */  
    /* recovery charging after stop charging 45min */  
    cm-discharging-max = <2700000>; /* 停充后，如果插着充电器，允许复充时间 */  
  
    /* the interval to feed charger watchdog */
```

```
cm-wdt-interval = <60>; //feed watchdog周期，同时也是开启feedwatchdog的开关
```

```
/* drop voltage in microVolts to allow shutdown */
```

```
cm-shutdown-voltage = <3100000>; //uvlo电压阈值
```

```
/* when 99% of the time is exceeded, it will be forced to 100% */
```

```
cm-tickle-time-out = <1500>; //电量维持在99% 15分钟后跳到100%
```

```
/* how much time to allow capacity change */
```

```
cm-one-cap-time = <30>; //允许电量增加1%最快时间
```

```
/* when the safe charging voltage is exceeded, stop charging */
```

```
cm-charge-voltage-max = <6500000>; //充电器过压保护电压阈值
```

```
/* drop voltage in microVolts to restart charging */
```

```
cm-charge-voltage-drop = <700000>; //复充电压条件
```

```
cm-dcp-jeita-temp-table = <1000 1030 0 4400000>, <1150 1180 700000 4400000>,
```

```
    <1450 1420 1150000 4400000>, <1600 1570 700000 4100000>;
```

```
cm-sdp-jeita-temp-table = <1000 1030 0 4400000>, <1150 1180 500000 4400000>,
```

```
    <1450 1420 500000 4400000>, <1600 1570 500000 4100000>;
```

```
cm-cdp-jeita-temp-table = <1000 1030 0 4400000>, <1150 1180 700000 4400000>,
```

```
    <1450 1420 1150000 4400000>, <1600 1570 700000 4100000>;
```

```
cm-unknown-jeita-temp-table = <1000 1030 0 4400000>, <1150 1180 500000 4400000>,
```

```
    <1450 1420 500000 4400000>, <1600 1570 500000 4100000>;
```

```
monitored-battery = <&bat>; //电池参数信息，容量自学习必须
```

```
cm-capacity-track; //容量自学习开关
```

```
cm-cap-remap-table = <1 2 3>, <2 3 2>, <98 99 2>, <99 100 3>; //电量再分配
```

```
regulator@0 {
```

```
    cm-regulator-name = "vddgen0";
```

```
    cable@0 {
```

```
        cm-cable-name = "USB";
```

```
        extcon = <&PMIC_typec>;
```

```
    };
```

```
};
```

```
};
```

4.3 PMIC_FGU 节点

```
&PMIC_fgu {  
    monitored-battery = <&bat>;  
    sprd,calib-resistance-real = <10000>;  
    sprd,calib-resistance-spec = <20000>;  
    first-calib-voltage = <3630000>; //低电量第一校准点电压  
    first-calib-capacity = <40>; //低电量第一校准点电量  
};
```

4.4 Charger IC 节点

```
compatible = "sprd,sc2703";  
reg = <0x4c>;  
sc2703-charger {  
    compatible = "sprd,sc2703-charger";  
    phys = <&ssphy>;  
    monitored-battery = <&bat>;  
    extcon = <&PMIC_typec>;  
    sprd,long-key-detection;  
    vddvbus:otg-vbus {  
        regulator-name = "vddvbus";  
    };  
};
```

5

常规 debug 介绍

5.1 Healthd log

```
healthd: battery l=23 v=3329 t=37.0 h=2 st=2 c=-419000 fc=2780000 chg=u
```

- battery l: 当前 UI 显示的电池电量
- v: 电池端电压
- h: 当前 health 状态
- st: 当前充电状态
- c: 当前充电电流 uA
- fc: 电池容量 uAh
- chg: 充电器类型, AC 或者 USB

5.2 插拔充电器 log

插充电器:

```
android_work: sent uevent USB_STATE=DISCONNECTED  
sc2720-charger sc27xx-charger: battery present = 1, charger type = 0  
charger-manager charger-manager: Charging Start/Stop  
musb-sprd 20200000.usb: device disconnect detected from VBUS GPIO.
```

- device connection detected from VBUS GPIO: 充电器插入
- battery present: 电池在位信息
- charger type: 充电器类型
- USB_STATE=CONNECTED: USB connected

拔充电器:

```
android_work: sent uevent USB_STATE=DISCONNECTED  
sc2720-charger sc27xx-charger: battery present = 1, charger type = 0  
charger-manager charger-manager: Charging Start/Stop  
musb-sprd 20200000.usb: device disconnect detected from VBUS GPIO.
```

- device disconnect detected from VBUS GPIO: 充电器拔出
- USB_STATE=DISCONNECTED: USB disconnected

5.3 限流信息

```
charger-manager charger-manager: target terminate voltage = 4350000, target current = 500000
```

```
charger-manager charger-manager: current-last jeita status: 2-2, current temperature: 294
```

- target terminate voltage: 目标截止电压
- target current: 目标充电电流
- current-last jeita status: 2-2: jeita 状态变迁
- current temperature: 当前电池温度

5.4 第一次开机

```
sc27xx-fgu sc27xx-fgu: First_poweron:cur = 8192, volt = 4317, ocv = 4317000, cap = 950\n]
```

```
charge first poweron reset!!!!
```

- First_poweron: 拔插过电池后第一次开机
- charge first poweron reset: 拔插过电池后第一次开机，这是 uboot log

5.5 battery info

```
charger-manager charger-manager: battery voltage = 4367000, OCV = 4327320, current = 320000, capacity = 885, charger status = 1, force set full = 0, charging current = 1150000, charging limit current = 2000000, battery temperature = 294, board temperature = 294, track state = 4, charger type = 2, thm_adjust_cur = 2000000, charger input voltage = 4924000
```

- battery voltage: 电池端电压
- OCV: 当前 OCV 电压值
- Current: 实时充电电流
- Capacity: FGU 统计电量值
- Charger status: 当前充电状态
- Force set full: 是否强制走到满电流程
- charging current: 当前 CC 限流值
- charging limit current: 当前充电芯片输入限流值
- battery temperature: 电池温度
- board temperature: 板温
- track state: 容量自学习状态
- charger type: 充电器类型
- thm_adjust_cur thermal: 限流值
- charger input voltage: 充电器输出电压值

充电状态值:

- charger status = 0: unknown
- charger status = 1: Charging

- charger status = 2: Discharging
- charger status = 3: Not charge
- charger status = 4: Full

5.6 debug 节点

- 查询充电状态（正在充电/未充电）

```
Cat sys/class/power_supply/battery/status
```

- "Charging" : 充电
- "Discharging": 没有插充电器，放电状态
- "Not charging": 插着充电器，但没有在充电
- "Full": 满电状态

- 查询当前手机电量百分比

```
Cat sys/class/power_supply/battery/capacity
```

值为当前 UI 显示的电量百分比

- 查询当前充电电流和电压

```
Cat sys/class/power_supply/battery/current_now
```

值为当前时刻充电电流值

- 正值为充电
- 负值为放电

```
Cat sys/class/power_supply/battery/voltage_now
```

当前时刻电池电压值

- 查询当前充电器类型（DCP，CDP，SDP，未识别...）

```
Cat sys/class/power_supply/sc2721_charger/usb_type
```

Unknown [SDP] DCP CDP C PD PD_DRP BrickID, “[]” 选中的值代表当前的充电器类型

- 过压保护，充电电压

```
Cat sys/class/power_supply/sc27xx-fgu/constant_charge_voltage
```

当这个值大于 6.5V（每个项目各自配置，典型值为 6.5V），停充

- 关 jeita 功能节点

```
Echo 0 > sys/class/power_supply/battery/charger.0/jeita_control
```

- 写 0 关 jeita 功能
- 写 1 打开 jeita 功能