



Unisoc Confidential For hiar

# AP Power 调试指导手册

文档版本  
发布日期

V1.6  
2020-08-14

**版权所有 © 紫光展锐科技有限公司。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。非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

Unisoc Confidential For hiar

# 紫光展锐科技有限公司



# 前言

## 概述

本文档详细介绍了手机 SOC AP 侧的功耗调试，涵盖了 CPU 和设备相关，包含 cpuidle、cpu hotplug、cpu dvfs、system suspend(deepsleep)。

## 读者对象


本文档主要适用于从事 UNISOC 智能机平台 AP 侧省电相关软件研发与测试人员。

## 缩略语

缩略语	英文全名	中文解释
AP	Application Processor	应用处理器
SOC	System On Chip	芯片级系统
CPU	Central Processing Unit	中央处理器
SMP	Symmetrical Multi-Proessing	对称多处理
DVFS	Dynamic Voltage and Frequency Scaling	动态电压频率调整

## 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
V1.0	2019-05-08	初稿。
V1.1	2019-08-01	更新 5.8 节校准模式下关闭外设，防止电流偏高的解决方案。
V1.2	2019-09-15	增加对 Android 10.0 的适配。
V1.3	2019-10-23	更改文档名使文档适用于展锐所有智能机平台。
V1.4	2020-02-11	适用平台信息新增 SL8541E。
V1.5	2020-04-02	更改文件名，优化内容，调整结构，更新格式等。
V1.6	2020-08-14	适用平台的 OS 增加 Android 11.0。

## 关键字

Cpuidle、Cpu Hotplug、Cpufreq、Deepsleep、Suspend/Resume。

Unisoc Confidential For hiar

# 目 录

1 概述 .....	1
2 Cpuidle 调试指导 .....	2
2.1 Cpuidle 介绍 .....	2
2.2 Cpuidle 内核开关 .....	3
2.3 Cpuidle sysfs 调试 .....	3
3 Cpu Hotplug 调试指导 .....	5
3.1 Cpu Hotplug 介绍 .....	5
3.2 Cpu Hotplug 内核开关 .....	6
3.3 Cpu Hotplug sysfs 调试 .....	6
4 Cpufreq 调试指导 .....	8
4.1 Cpufreq 介绍 .....	8
4.2 调频调压介绍 .....	9
4.2.1 调压 .....	9
4.2.2 调频 .....	9
4.3 Cpufreq 驱动概览 .....	9
4.3.1 Config 文件 .....	9
4.3.2 DVFS 驱动 .....	10
4.3.3 OPP 表 .....	10
4.4 Cpufreq sysfs 调试 .....	11
4.5 常见问题处理 .....	12
4.5.1 临时关闭 DVFS .....	12
4.5.2 固定 CPU 频率 .....	12
4.5.3 DVFS 驱动成功加载日志 .....	12
4.5.4 I2C 通信问题 .....	13
4.5.5 I2C 无法获取操作权限 .....	13
5 Suspend/Resume 调试指导 .....	14
5.1 Suspend/Resume 介绍 .....	14
5.2 Deepsleep 前置条件 .....	15
5.3 Deepsleep 功耗问题分析步骤 .....	15
5.4 常用分析工具 .....	15
5.4.1 YLog .....	15
5.4.2 Trace32 debug bus .....	17
5.5 常用 Shell 调试命令 .....	17
5.5.1 打印控制相关命令 .....	17
5.5.2 wake_lock 相关命令: .....	17

5.5.3 手动触发 Suspend 命令 .....	18
5.6 常见问题分析 .....	18
5.6.1 Deepsleep VDDCORE 电源域底电流偏高 .....	18
5.6.2 底电流偏高 .....	18
5.6.3 灭屏待机平均电流偏大分析 .....	19
5.7 基于时间戳获取唤醒时间 .....	19
5.8 校准模式下关闭外设防止电流偏高的解决方案 .....	24
5.8.1 Touch Panel 类解决方案 .....	24
5.8.2 LCD 类解决方案 .....	25
5.8.3 Charge 类解决方案 .....	25
5.8.4 指纹类解决方案 .....	30

Unisoc Confidential For hiar

## 图目录

图 2-1 内核中的 Cpuidle .....	2
图 2-2 Cpuidle 节点信息 .....	3
图 3-1 内核中的 Cpu Hotplug.....	5
图 3-2 SC9863A 四小核 Online 信息.....	6
图 4-1 内核中的 Cpufreq.....	8
图 4-2 SC9863A DCDC 供电示意图 .....	9
图 5-1 Suspend/Resume 流程.....	14
图 5-2 Power Monitor 电流测试图 1.....	21
图 5-3 Power Monitor 电流测试图 2.....	22
图 5-4 Power Monitor 电流测试图 3.....	23
图 5-5 Power Monitor 电流测试图 4.....	23
图 5-6 Power Monitor 电流测试图 5.....	23

Unisoc Confidential For hiar

## 表目录

---

表 4-1 CPU 核运行频率与电压 .....	10
表 5-1 Kernel Log 中确定子系统睡眠状态的关键字 .....	19
表 5-2 Kernel Log 搜索 Times 关键字的搜索结果 .....	20

Unisoc Confidential For hiar



# 1 概述

在整个手机项目的量产节点中，功耗验收是一个关键指标。功耗、性能和稳定性往往是互相牵制的。

从性能角度来看，要想手机有更高的跑分，运行网络游戏不卡顿，则意味着手机需要支持更高的 CPU 负载和更高 CPU 运行频率，需要更多 CPU 核参与调度，这些都会带来更高的功耗。

从功耗角度看，在不同用户应用场景下，手机 SOC 需要动态调整不同子系统的开关、sleep 信号、clock gated 等，使得手机功耗尽可能低。

本文详细介绍了手机 SOC AP 侧的功耗调测，涵盖了 CPU 和相关设备，包含 cpuidle、cpu hotplug、cpu dvfs、deepsleep 场景系统的 suspend/resume。

Unisoc Confidential For hiar

# 2 Cpuidle 调试指导

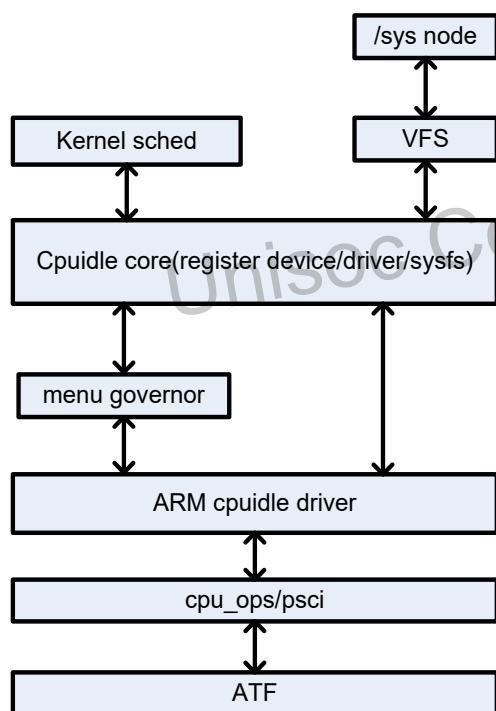
## 2.1 Cpuidle 介绍

CPU 空闲时就会运行 idle 线程，上层 governor 机制根据当前环境，决定 CPU 进入 WFI、Light Sleep、Heavy Sleep、CORE-PD 或 Cluster-PD 等状态中的一个，以达到低功耗的目的。

Linux 原生 governor 有 ladder 和 menu 两种，紫光展锐平台一般配置 menu governor。menu governor 根据当前 cpu 的负载以及各个状态的功耗和唤醒延迟，计算出一个索引值，这个索引值被传递给底层的 idle driver，直接驱动 cpu 进入相应状态。

内核中的 Cpuidle 如图 2-1 所示。

图2-1 内核中的 Cpuidle



Cpuidle 在内核中的详细介绍可参考以下资料：

[http://www.wowotech.net/pm\\_subsystem/cpuidle\\_overview.html](http://www.wowotech.net/pm_subsystem/cpuidle_overview.html)

[http://www.wowotech.net/pm\\_subsystem/cpuidle\\_core.html](http://www.wowotech.net/pm_subsystem/cpuidle_core.html)

[http://www.wowotech.net/pm\\_subsystem/cpuidle\\_arm64.html](http://www.wowotech.net/pm_subsystem/cpuidle_arm64.html)

[http://www.wowotech.net/pm\\_subsystem/cpuidle\\_menu\\_governor.html](http://www.wowotech.net/pm_subsystem/cpuidle_menu_governor.html)

## 2.2 Cpuidle 内核开关

Cpuidle 内核开关如下：

- cpuidle 宏开关

```
CONFIG_CPU_IDLE           //cpuidle总开关
CONFIG_CPU_IDLE_GOV_LADDER //ladder governor选择开关
CONFIG_CPU_IDLE_GOV_MENU  //menu governor选择开关
```

- Idle driver 宏开关

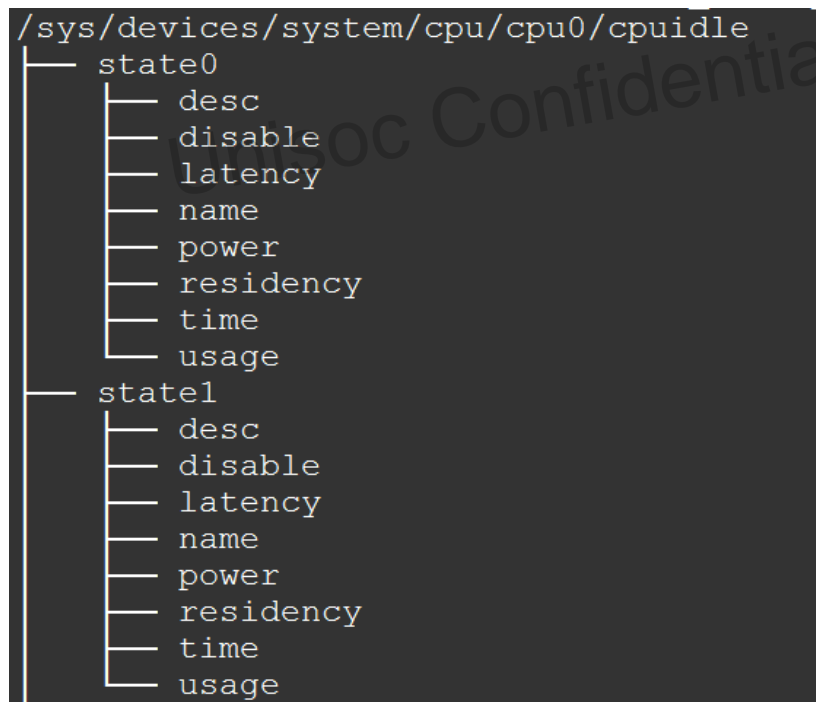
```
CONFIG_ARM_CPUIDLE        //arm策略驱动开关，依赖于CONFIG_CPU_IDLE开关
CONFIG_ARM_SPRD_CPUIDLE   //sprd驱动策略开关，依赖于CONFIG_CPU_IDLE开关
```

## 2.3 Cpuidle sysfs 调试

### 查看 cpuidle 节点信息

通过 adb shell 命令行查看 cpuidle 节点信息，如图 2-2 所示。

图2-2 Cpuidle 节点信息



```
/sys/devices/system/cpu/cpu0/cpuidle
├── state0
│   ├── desc
│   ├── disable
│   ├── latency
│   ├── name
│   ├── power
│   ├── residency
│   ├── time
│   └── usage
└── state1
    ├── desc
    ├── disable
    ├── latency
    ├── name
    ├── power
    ├── residency
    ├── time
    └── usage
```

### 查看当前 CPUIDLE Driver

查看命令如下：

```
cat /sys/devices/system/cpu/cpuidle/current_driver
```

以 SC9863A(kernel4.4)为例, 查看结果如下:

```
arm_idle
```

## 查看当前 CPUIDLE Governor

查看命令如下:

```
cat /sys/devices/system/cpu/cpuidle/current_governor_ro
```

以 SC9863A (kernel4.4)为例, 查看结果如下:

```
sprd
```

以 SC9863A (kernel4.14)为例, 查看结果如下:

```
menu
```

## 查看当前状态

以 CPU4 为例, 查看命令如下:

```
console:/ # cat /sys/devices/system/cpu/cpu4/cpuidle/state1/name
```

查看结果返回当前状态, 例如:

```
core_pd
```

## 查看当前状态累积停留时间

以 CPU4 为例, 查看命令如下:

```
console:/ # cat /sys/devices/system/cpu/cpu4/cpuidle/state1/time
```

查看结果返回当前状态累积停留时间(单位 us), 例如:

```
21653602
```

## 查看当前状态累积进出次数

以 CPU4 为例, 查看命令如下:

```
console:/ # cat /sys/devices/system/cpu/cpu4/cpuidle/state1/usage
```

查看结果返回当前状态累积进出次数, 例如:

```
2510
```

## 关闭当前状态

以 CPU4 为例, 关闭命令如下:

```
console:/ # echo 1 > /sys/devices/system/cpu/cpu4/cpuidle/state1/disable
```

# 3 Cpu Hotplug 调试指导

## 3.1 Cpu Hotplug 介绍

单核系统，只要系统处于开机状态就不能关闭唯一 CPU。

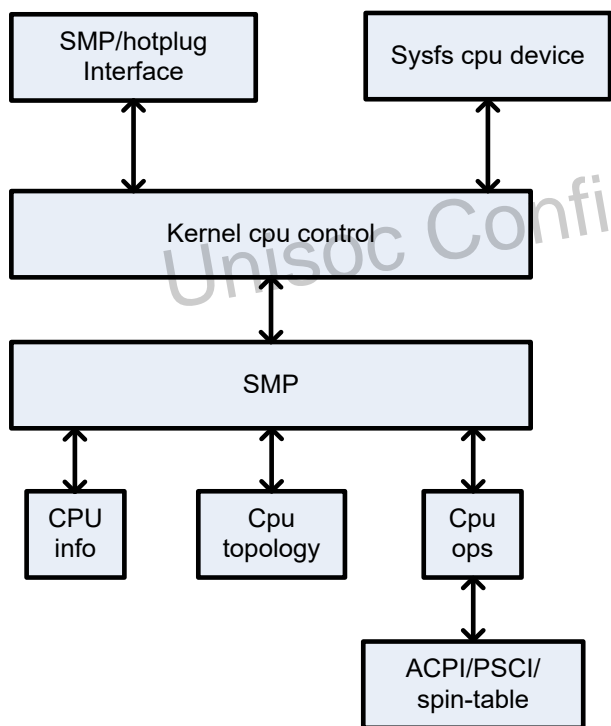
随着 SMP 出现，底层驱动提供了 cpu\_up/cpu\_down 接口，可实现动态 plug in/unplug 某个 CPU 核。

该功能在 Suspend/Resume 的 disable\_nonboot\_cpus()/enable\_nonboot\_cpus()中被动态调用。

该功能亦可用于压力测试脚本，进行 CPU 动态 hotplug 测试。

内核中的 Cpu Hotplug 如图 3-1 所示。

图3-1 内核中的 Cpu Hotplug



Cpu Hotplug 在内核中的详细介绍可参考以下资料：

[http://www.wowotech.net/pm\\_subsystem/cpu\\_ops.html](http://www.wowotech.net/pm_subsystem/cpu_ops.html)

[http://www.wowotech.net/pm\\_subsystem/cpu\\_hotplug.html](http://www.wowotech.net/pm_subsystem/cpu_hotplug.html)

## 3.2 Cpu Hotplug 内核开关

Cpu Hotplug 内核宏开关如下：

```
CONFIG_SMP                //该开关控制开启多核
CONFIG_HOTPLUG_CPU        //该开关控制开启Hotplug，依赖于CONFIG_SMP。
```

## 3.3 Cpu Hotplug sysfs 调试

查看当前 Online 的 CPU 信息

以 SC9863A 为例，查看 Online CPU 信息的命令如下：

```
cat /proc/cpuinfo
```

如图 3-2 所示，截图显示了四小核的查询结果。

图3-2 SC9863A 四小核 Online 信息

```
console:/ # cat /proc/cpuinfo
processor       : 0
BogoMIPS      : 52.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrdm lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x1
CPU part      : 0xd05
CPU revision   : 0

processor       : 1
BogoMIPS      : 52.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrdm lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x1
CPU part      : 0xd05
CPU revision   : 0

processor       : 2
BogoMIPS      : 52.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrdm lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x1
CPU part      : 0xd05
CPU revision   : 0

processor       : 3
BogoMIPS      : 52.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrdm lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x1
CPU part      : 0xd05
CPU revision   : 0
```

查看当前 Online 的 CPU 个数

同样以 SC9863A 为例，查看 Online CPU 个数的命令如下：

```
console:/ # cat /sys/devices/system/cpu/online
```

查看结果如下：

0-7

## Unplug 某个 CPU

以下命令将 CPU7 和 CPU3 Unplug:

```
console:/ # echo 0 > /sys/devices/system/cpu/cpu7/online
console:/ #
console:/ # cat /sys/devices/system/cpu/online
0-6
console:/ #
console:/ # echo 0 > /sys/devices/system/cpu3/online
console:/ #
console:/ # cat /sys/devices/system/cpu/online
0-2,4-6
console:/ #
```

## Plugin 某个 CPU

以下命令将 CPU3 Plugin。

```
console:/ # cat /sys/devices/system/cpu/online
0-2,4-6
console:/ # echo 1 > /sys/devices/system/cpu3/online
console:/ # cat /sys/devices/system/cpu/online
0-6
```

Unisoc Confidential For hiar

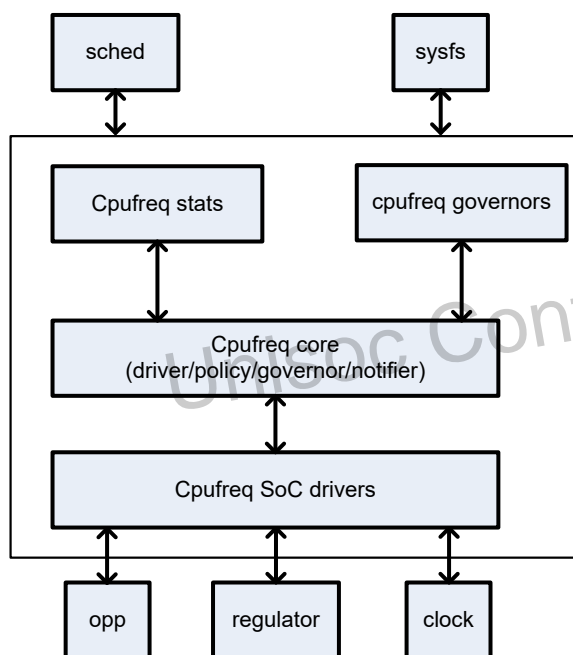
# 4 Cpufreq 调试指导

## 4.1 Cpufreq 介绍

Cpufreq 根据 CPU 负载动态进行调频调压，实现功耗与性能之间的平衡。Cpufreq 的调频调压动作由内核中的 governor 触发，选择当前负载下一个最优的 OPP 索引，通过 SOC 相关的 Cpufreq driver，实现调频调压过程。

内核中的 Cpufreq 如图 4-1 所示。

图4-1 内核中的 Cpufreq



Cpufreq 在内核中的详细介绍可参考如下资料：

[http://www.wowotech.net/pm\\_subsystem/cpufreq\\_overview.html](http://www.wowotech.net/pm_subsystem/cpufreq_overview.html)

[http://www.wowotech.net/pm\\_subsystem/cpufreq\\_driver.html](http://www.wowotech.net/pm_subsystem/cpufreq_driver.html)

[http://www.wowotech.net/pm\\_subsystem/cpufreq\\_core.html](http://www.wowotech.net/pm_subsystem/cpufreq_core.html)

[http://www.wowotech.net/pm\\_subsystem/cpufreq\\_governor.html](http://www.wowotech.net/pm_subsystem/cpufreq_governor.html)

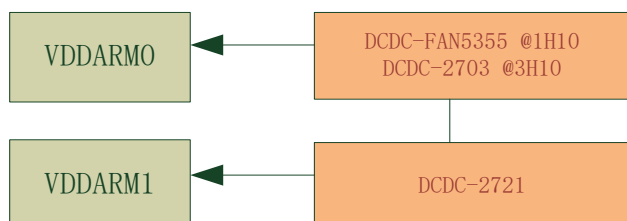


## 4.2 调频调压介绍

### 4.2.1 调压

以 SC9863A 平台为例，通过 DCDC 给大核和小核供电，如图 4-2 所示。

图4-2 SC9863A DCDC 供电示意图



VDDARM0 给小核（CPU0~CPU3）供电，可通过 I2C 控制其电压输出。

VDDARM1 给大核（CPU4~CPU7）供电，可通过 ADI 控制其电压输出。

#### 说明

1h10 的 board 使用 DCDC-FAN5355 这颗 dcdc。

3h10 的 board 使用 DCDC-2703 这颗 dcdc。

以上类型 DCDC 都是固定使用，更换 DCDC 须经过 Unisoc 调试认证。

### 4.2.2 调频

HW DVFS 调频操作是由硬件 HW DVFS IP 自动完成。

SW DVFS 通过 clk driver 接口完成调频操作，调频参数配置在 DTS 中完成。

## 4.3 Cpufreq 驱动概览

### 4.3.1 Config 文件

```
CPU Power Management --->
  CPU Frequency scaling --->
    [ ] CPU Frequency scaling          # 打开/关闭dvfs
    Default CPUFreq governor --->
      ( ) performance                   # 更改默认为performance governor
      ( ) powersave                    # 更改默认为powersave governor
      ( ) userspace                     # 更改默认为userspace governor
      ( ) ondemand                      # 更改默认为ondemand governor
      ( ) conservative                  # 更改默认为conservative governor
      ( ) interactive                   # 更改默认为interactive governor
```

(X) schedutil

# 更改默认为schedutil governor

## 说明

Unisoc 默认使用 schedutil，已对此策略调优，不可随意更改。

## 4.3.2 DVFS 驱动

### 4.3.2.1 DVFS 驱动配置

以 SC9863A 平台为例，DVFS 驱动配置相关的宏如下：

```
CONFIG_ARM_SPRD_HW_CPUFREQ
CONFIG_ARM_SPRD_HW_CPUFREQ_SHARKL3
```

### 4.3.2.2 DVFS 驱动文件

软件硬件 DVFS 驱动文件如下：

```
CPU SW DVFS driver: sprd-cpubreqsw.c
CPU HW DVFS driver: sprd-hwdfvs-sharkl3.c
```

### 4.3.2.3 DVFS 驱动加载

DVFS 驱动加载步骤如下：

步骤 1 sprd-hwdfvs-sharkl3.c 开机初始化 hw dfvs ip，等待上层 sprd-cpubreq.c 调用。

步骤 2 little cluster(CPU0~CPU3)上电后调用 sprd-cpubreq.c 加载 dfvs driver,

big cluster(CPU4~CPU7)上电后调用 sprd-cpubreq.c 加载 dfvs driver。

----结束

## 4.3.3 OPP 表

表4-1 CPU 核运行频率与电压

CPU 核	运行频率与电压
little cluster	[freq-1200000000 & volt-1100000]
	[freq-1100000000 & volt-1050000]
	[freq-1000000000 & volt-1000000]
	[freq-884000000 & volt-950000]
	[freq-768000000 & volt-900000]
big cluster	[freq-1570000000 & volt-1100000]
	[freq-1500000000 & volt-1050000]
	[freq-1400000000 & volt-1000000]

CPU 核	运行频率与电压
	[freq-1225000000 & volt-950000]
	[freq-1050000000 & volt-900000]
	[freq-768000000 & volt-900000]

#### 说明

CPU 核的运行频率与电压是在 sc9863a.dtsi 中定义，不可改动，更改可能导致系统不稳定。

运行频率单位为 Hz，电压单位为微伏。如 freq-1200000000 & volt-1100000 即运行频率 1.2GHz，电压 1.1V。

## 4.4 Cpubreq sysfs 调试

### 查看可用 governor

```
cat /sys/devices/system/cpu/cpufreq/policy*/scaling_available_governors
```

### 设置和查看 governor

```
cat /sys/devices/system/cpu/cpufreq/policy*/scaling governor
echo schedutil >/sys/devices/system/cpu/cpu0/cpufreq/scaling governor
echo schedutil >/sys/devices/system/cpu/cpu4/cpufreq/scaling governor
echo interactive > /sys/devices/system/cpu/cpu0/cpufreq/scaling governor
echo interactive > /sys/devices/system/cpu/cpu4/cpufreq/scaling governor
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling governor
echo userspace > /sys/devices/system/cpu/cpu4/cpufreq/scaling governor
```

### 查看可用频率表

```
cat /sys/devices/system/cpu/cpufreq/policy*/ scaling_available_frequencies
```

### 设置规定频率

```
echo 1200000 >/sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
echo 1100000 >/sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
echo 1000000 >/sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
echo 884000 >/sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
echo 768000 >/sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed

echo 1570000 >/sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed
echo 1500000 >/sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed
echo 1400000 >/sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed
echo 1225000 >/sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed
echo 1050000 >/sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed
echo 768000 >/sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed
```

### 查看当前频率

```
cat /sys/devices/system/cpu/cpufreq/policy*/cpufreq_cur_freq
```

## 查看当前电压

```
cat /sys/kernel/debug/sprd-regulator/vddcpu/voltage
cat /sys/kernel/debug/SC2703_BUCK_CH1/voltage
```

## 固定电压

```
echo 1200000 > /sys/devices/system/cpu/cpubreq/policy0/scaling_fix_freq
```

### 说明

以 9863A 平台为例，其他平台依此类推。

## 4.5 常见问题处理

### 4.5.1 临时关闭 DVFS

关闭后，CPU 频率就不会根据当前任务量动态调整。

```
echo userspace > /sys/devices/system/cpu/cpu0/cpubreq/scaling_governor
echo userspace > /sys/devices/system/cpu/cpu4/cpubreq/scaling_governor
```

### 4.5.2 固定 CPU 频率

下例是固定 CPU 频率

```
echo userspace > /sys/devices/system/cpu/cpu0/cpubreq/scaling_governor
echo userspace > /sys/devices/system/cpu/cpu4/cpubreq/scaling_governor
```

小核：

```
echo 1200000 > /sys/devices/system/cpu/cpubreq/policy0/scaling_setspeed
echo 1100000 > /sys/devices/system/cpu/cpubreq/policy0/scaling_setspeed
echo 1000000 > /sys/devices/system/cpu/cpubreq/policy0/scaling_setspeed
echo 884000 > /sys/devices/system/cpu/cpubreq/policy0/scaling_setspeed
echo 768000 > /sys/devices/system/cpu/cpubreq/policy0/scaling_setspeed
```

大核：

```
echo 1570000 > /sys/devices/system/cpu/cpubreq/policy4/scaling_setspeed
echo 1500000 > /sys/devices/system/cpu/cpubreq/policy4/scaling_setspeed
echo 1400000 > /sys/devices/system/cpu/cpubreq/policy4/scaling_setspeed
echo 1225000 > /sys/devices/system/cpu/cpubreq/policy4/scaling_setspeed
echo 1050000 > /sys/devices/system/cpu/cpubreq/policy4/scaling_setspeed
echo 768000 > /sys/devices/system/cpu/cpubreq/policy4/scaling_setspeed
```

### 4.5.3 DVFS 驱动成功加载日志

小核或大核 DVFS 驱动加载成功会打印如下日志：

```
init cpu0 is ok, freq=1200000000, freq_req=1200000000, volt_req=1100000
init cpu4 is ok, freq=1570000000, freq_req=1570000000, volt_req=1100000
```

HW DVFS 加载成功会打印如下日志：

```
sprd cpubreqhw probed true
```

### 4.5.4 I2C 通信问题

如打印以下日志，可以判定是 I2C 通信出现问题：

```
“sprd_cpufreqhw_isr : CHNL0 TIMEOUT”
```

一般可能是单颗 DCDC 有问题，I2C 没有响应，可以更换 DCDC 试试或者检查 I2C 硬件的 SDA、SCL 是否有问题。

### 4.5.5 I2C 无法获取操作权限

当有“CHNL0 i2c\_lock\_adapter fail”出现，可能是其它设备驱动一直占用 I2C 总线，需要排查其它共用 I2C 总线的设备驱动是否异常。

Unisoc refphone 参考机的 I2C 总线挂载了 charger 和 flash，若客户手机方案在 I2C 总线上还挂载有其它设备，请检查这些设备的驱动是否有问题。

Unisoc Confidential For hiar

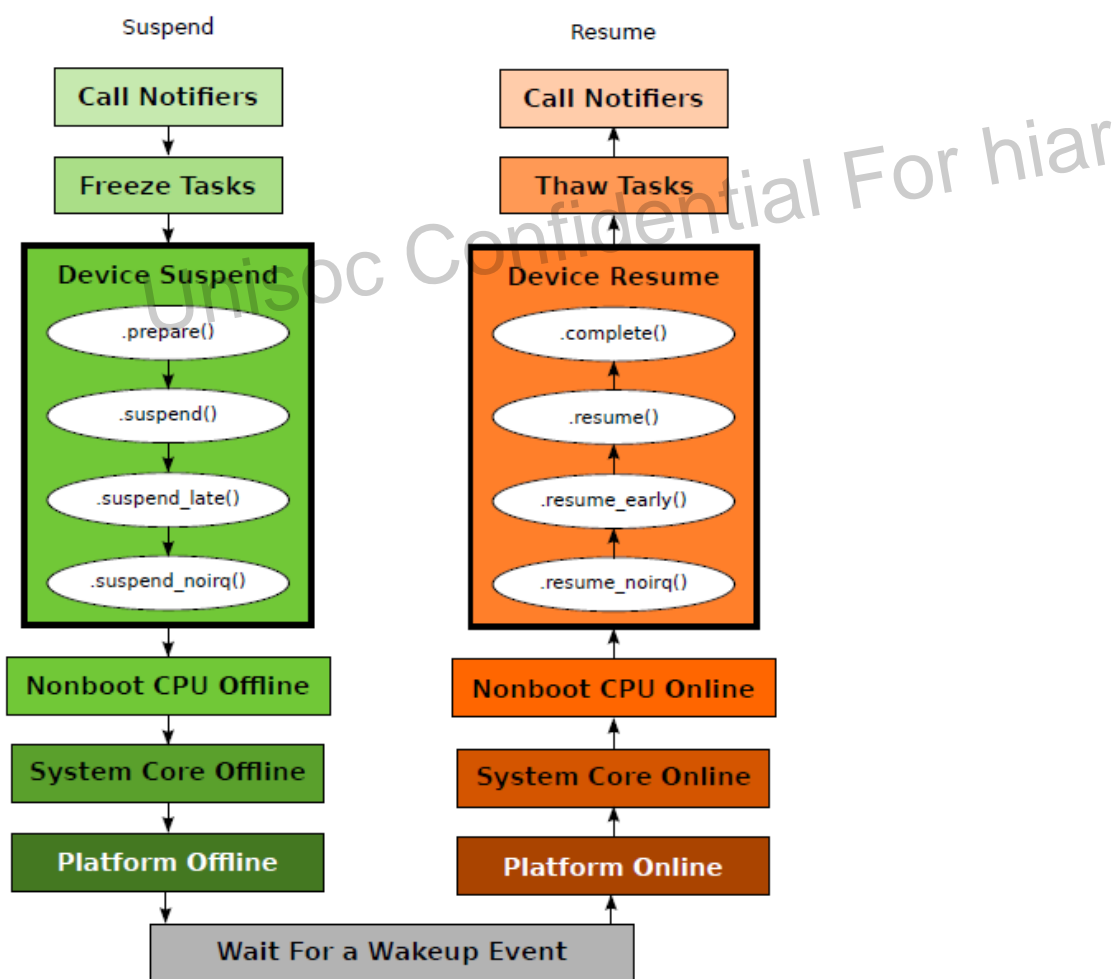
# 5 Suspend/Resume 调试指导

## 5.1 Suspend/Resume 介绍

System Suspend/Resume 是指系统在没有任务运行或者工程师通过 shell 强行写 sysfs 节点时，系统进入睡眠(deepsleep)和唤醒的过程。进入 system suspend 的过程中，应用程序被 freeze，所有外设被挂起，nonboot CPUs 被强行 unplug，CPU0 和 cluster 进入下电状态，DDR 保持上电但进入自刷新状态。

Suspend/Resume 流程如图 5-1 所示。

图5-1 Suspend/Resume 流程



System Suspend/Resume 内核中的详细介绍可参考以下资料：

[http://www.wowotech.net/pm\\_subsystem/suspend\\_and\\_resume.html](http://www.wowotech.net/pm_subsystem/suspend_and_resume.html)

[http://www.wowotech.net/pm\\_subsystem/wakelocks.html](http://www.wowotech.net/pm_subsystem/wakelocks.html)

[http://www.wowotech.net/pm\\_subsystem/237.html](http://www.wowotech.net/pm_subsystem/237.html)

[http://www.wowotech.net/pm\\_subsystem/suspend-irq.html](http://www.wowotech.net/pm_subsystem/suspend-irq.html)

## 5.2 Deepsleep 前置条件

一般场景下测试 Deepsleep 的基础前置条件是：无卡、飞行模式、关 Ylog、关 BT/Wifi。在此基础上再叠加其它衍生场景测试条件，如插卡待机要退出飞行模式，还要考虑要不要打开数据链接，BT/Wifi 场景测试要打开 BT/Wifi，并考虑要不要接入网络，出现问题时要打开 Ylog 抓取 Log，具体测试前置条件要根据测试场景需要配置。

分析 Deepsleep 场景功耗时，对项目 Deepsleep 功耗特性至少要作以下几方面了解：

- 功耗目标值
- 项目硬件架构，如是否有 sensorHub。
- Deepsleep 底电流及长待机电流情况
- 唤醒情况等等。

## 5.3 Deepsleep 功耗问题分析步骤

Deepsleep 功耗问题分析步骤如下：

步骤 1 复现问题，测试底电流，并开启 Ylog 同步抓取 log。

步骤 2 确认 AP 睡眠状态。

步骤 3 确认其它子系统睡眠状态。

步骤 4 长待机场景中分析唤醒情况。

----结束

## 5.4 常用分析工具

### 5.4.1 YLog

Ylog 是分析功耗问题最常用工具，主要用于分析 AP 不睡、AP 睡眠其它子系统不睡、长待机异常唤醒等问题。

抓取 Ylog 前清空手机上的 Ylog，以便抓取的 Ylog 只有一份，方便分析。

如果没有插入 T 卡，Ylog 会保存在系统的 sdcard/ylog，或 data/ylog 路径下，可以 adb pull 出来；如果插入了 T 卡，Ylog 会保存在系统的 storage/sdcard0/ylog，即 T 卡根目录。

分析一些和其它子系统相关的场景功耗问题时，如 WIFI 待机，最好关掉这些子系统的 Ylog 开关，因为这可能会影响问题分析。

以 SC9863A 为例，测试时只开 AP 侧 log 的操作方法如下：

步骤 1 在拨号界面，输入\*##83781#\*，打开工程模式->DEBUG&LOG->YLOG，进入 Ylog 界面。

步骤 2 单击左下角的图标，清空之前手机上的 log。

步骤 3 在 Settings-> Scene Select ->Custom scene 单击+，弹出窗口，单击 ok，进入 Custom Scene 界面。

步骤 4 在 Custom Scene 界面除第一个 Android Log 和第三个 AP Cap Log 外，其它选项都关闭，然后单击 commit 保存。

----结束

一般 Log 中会打印出 SOC 睡眠状态寄存器，详细说明见下例：

```
#####Enter Deepsleep Condition Check!#####
#####Enter AP_AHB Condition Check!##### //判断 device 是否关闭，此处无异常
Enter AP_APB Condition Check!#####
BIT_AP_APB_UART1_EB not 0
BIT_AP_APB_SIM0_EB not 0
#####Enter AON_APB Condition Check!#####
BIT_AON_APB_CA7_DAP_EB not 0
BIT_AON_APB_AP_DAP_EB not 0
#####PWR STATUS0#####
---status of power domain 'MM_TOP' :0x00000007 //7-MM_TOP 处于 power down 状态；0-MM_TOP 处于 power on 状态
---status of power domain 'GPU_TOP' :0x00000007 //7-GPU_TOP 处于 power down 状态；0-GPU_TOP 处于 power on 状态
//如下是 AP deepsleep 的条件，此时应该除了 CA7_C0、CA7_TOP、AP_SYS 在最后会被 power down，其余在当前都已经处于 power down 状态
---status of power domain 'AP_SYS' :0x00000000 // 7-AP_SYS 处于 power down 状态；0-AP_SYS 处于 power on 状态
---status of power domain 'CA7_C3' :0x00000007 // 7-CA7_C3 处于 power down 状态；0-CA7_C3 处于 power on 状态
---status of power domain 'CA7_C2' :0x00000007 // 7-CA7_C2 处于 power down 状态； 0-CA7_C2 处于 power on 状态
---status of power domain 'CA7_C1' :0x00000007 // 7-CA7_C1 处于 power down 状态； 0-CA7_C1 处于 power on 状态
---status of power domain 'CA7_C0' :0x00000000 // 7-CA7_C0 处于 power down 状态； 0-CA7_C0 处于 power on 状态
---status of power domain 'CA7_TOP' :0x00000000 // 7-CA7_TOP 处于 power down 状态；0-CA7_TOP 处于 power on 状态
//如下是 chip deepsleep 的条件，此时除了 PUB_SYS 会在 AP_SYS 掉电后 power down，其余在当前都已经处于 power down 状态
#####PWR STATUS1#####
---status of power domain 'WCN_GNSS' :0x00000007 // 7-WCN_GNSS 处于 power down 状态；0-WCN_GNSS 处于 power on 状态
---status of power domain 'WCN_WIFI' :0x00000007 // 7-WCN_WIFI 处于 power down 状态；0-WCN_WIFI 处于 power on 状态
---status of power domain 'WCN_TOP' :0x00000007 // 7-WCN_TOP 处于 power down 状态；0-WCN_TOP 处于 power on 状态
```



```
##--status of power domain 'PUB_SYS' :0x00000000 // 7-PUB_SYS 处于 power down 状态; 0-PUB_SYS 处于 power on 状态
##--status of power domain 'WTLCP_TGDSP' :0x00000007 // 7-WTLCP_TGDSP 处于 power down 状态; 0-WTLCP_TGDSP 处于 power on 状态
##--status of power domain 'HU3GE_A' :0x00000007 // 7-HU3GE_A 处于 power down 状态; 0-HU3GE_A 处于 power on 状态
##--status of power domain 'CP_SYS' :0x00000007 // 7-CP_SYS 处于 power down 状态; 0-CP_SYS 处于 power on 状态
//如下是各个子系统的 sleep 的状态, 这里重点关注 CM4_SLP 是否 sleep, 因为其他的都可以通过上述的 power 状态确认
#####SLEEP_STATUS#####
##--status of sleep 'CM4_SLP' :0x00000000 // 6-CM4 处于 normal 状态; 0-CM4 处于 sleep 状态
##--status of sleep 'WCN_SLP' :0x00000000 // 6-WCN 处于 normal 状态; 0-WCN 处于 sleep 状态
##--status of sleep 'CP_SLP' :0x00000000 // 6-CP 处于 normal 状态; 0-CP 处于 sleep 状态
##--status of sleep 'AP_SLP' :0x00000006 // 6-AP 处于 normal 状态; 0-AP 处于 sleep 状态
wakeup by INTC1!!mask:0x00000040 raw:0x00000040 en:0xb2bf4ce8 //"mask"为获取到的唤醒源, 这些唤醒源是发送到 ap 的中断, 具体中断源可以参照各个项目的 interrupt list
AON INTC mask:0x00000010 raw:0x00000010 en:0xc0005034//获取 sleep 唤醒源, 这些唤醒源 pending 到 AON 上, 作为唤醒用
```

## 5.4.2 Trace32 debug bus

Trace32 debug bus 主要用于 Deepsleep 信号已经出来但功耗偏高场景下, 分析是否有哪个模块阻塞子系统睡眠、哪个电源域没有正常 power down 等问题。

### 说明

Trace32 是第三方工具, 如有需要, 请找第三方 vendor 支持。

## 5.5 常用 Shell 调试命令

### 5.5.1 打印控制相关命令

```
echo 7 > /proc/sys/kernel/printk
echo 0 > /sys/module/printk/parameters/console_suspend
```

### 说明

- 两个命令都可以在 DTS 配置文件中修改。
- 第一条命令用于 kernel 打印等级。
- 正常 Suspend 流程中 console 会被挂起, Suspend 后打印 log 临时保存在 buffer 中, 直到下次唤醒才打印出来。
- 第二条命令用于打开 no\_console\_suspend, 在 Suspend 流程中不会挂起 console, 控制台会一直输出 log, 直到跳出 kernel (系统调用到 SML)或 Suspend 成功, 可用于打印 Suspend 后面一些状态 log。

### 5.5.2 wake\_lock 相关命令:

```
echo xxx > /sys/power/wake lock
echo xxx > /sys/power/wake unlock
cat /sys/power/wake_lock
```

### 说明

- xxx 为 wake\_lock 名称。
- 第一个命令用于持有 wake\_lock。

- 第二个命令用于释放 `wake_lock`。
- 第三条命令用于查看系统现有的 `wake_lock`。

### 5.5.3 手动触发 Suspend 命令

```
echo mem > /sys/power/state
```

#### 说明

该命令手动触发 Suspend 流程，强制 kernel Suspend。

## 5.6 常见问题分析

### 5.6.1 Deepsleep VDDCORE 电源域底电流偏高

VDDCORE 是给板级供电的，上面有很多模块、外设、电源域和时钟。在 Deepsleep 场景 VDDCORE 会降压以减小功耗，因此可以通过测量 VDDCORE 电压变化来判断是否进入 Deepsleep。

VDDCORE 漏电导致 Deepsleep 底电流偏高的问题从以下两方面分析：

- 软件上检查相关状态寄存器  
根据 log 或 trace32 检查各电源域是否有正常关闭，各 PLL 是否有正常 gated（最好用 trace32 检查，因为在打印 log 时 AP 还在工作）。AP 侧也可以同步查看 AP-APB、AP-AHB 上的外设是否有正常关闭。
- 硬件上分解电流  
VDDCORE 是给板级供电，上面有很多模块，电路设计复杂，功率消耗和具体的硬件电路设计有关，所以需要请硬件同事尽量细化分解电流，找出具体是哪个电源域功耗过大。进一步可以根据 SOC power tree 找出具体是哪个模块功耗过大，最后请模块软件 owner 确认是否配置正确。

### 5.6.2 底电流偏高

底电流偏高可以分为两种情况：

- 有 Deepsleep 信号出来  
这种情况一般是有漏电或者哪个电源域没有正常关闭，这时可以请 HW 协助分解电流找出是哪个电源漏电，同时通过 log 或者 trace32 查看 PMU 状态寄存器确认是什么电源域没有正常关闭。
- 没有 Deepsleep 信号出来  
这种情况很明显是有子系统没有睡眠导致。  
首先通过 log 或者 trace32 确认 AP 睡眠状态，AP 睡眠失败一般可以通过 log 上下文和 PMU 状态寄存器确认问题点，比如是不是有应用拿 wakelock。  
然后找到问题发生时间段内的 log 根据 PMU 状态寄存器确认其它子系统睡眠状态。

如果有应用拿 wakelock 导致 AP 长时间不能睡眠，可以根据 kernel log 判断是什么 wakelock。

log 关键词如下：

PM: Wakeup pending, aborting Suspend

active wakeup source:.....(wake event name)

根据 Kernel log 时间轴，找到对应的时间段的 Android log，找到持锁的具体应用，Log 搜索 acquireWakeLockInternal 关键字查看申请 wakelock 情况，搜索 releaseWakeLockInternal 查看释放情况，lock=后面的证书能降申请和释放对应起来，结合时间轴能得到持锁的时长，根据 tag 能查看申请 wakelock 的具体应用。

Kernel Log 中确定子系统睡眠状态的关键字见表 5-1。

表5-1 Kernel Log 中确定子系统睡眠状态的关键字

序号	关键字	说明
1	PM: suspend entry	AP Deepsleep 入口
2	PM: suspend exit	AP Deepsleep 出口
3	wake up by	AP Resume 唤醒源
4	Enter Deepsleep Condition Check!	检查进入 Deepsleep 前寄存器配置是否符合逻辑
5	REG_PMU_APB_SLEEP_STATUS	子系统睡眠状态
6	REG_PMU_APB_SLEEP_CNTx	子系统进入睡眠的次数
7	REG_PMU_APB_PWR_STATUSx_DBG	电源域开关状态

### 5.6.3 灭屏待机平均电流偏大分析

灭屏待机平均电流偏大原因：底电流偏大、底电流正常但有异常唤醒。

底电流偏大需要按照上述介绍的方法查找不能睡眠的原因。如果确认能睡眠，那就需要硬件电流分解，查看哪一部分耗电异常。底电流正常但有异常唤醒，首先在电流图上能抓到异常唤醒的波形，确认异常唤醒的时间点，然后再 dump ylog，分析相应时间的 log，确认异常唤醒源。

常见正常 RTC 唤醒：

- Healthd: 上报电池信息，目前唤醒周期 10min。
- Doze: Android6.0 以后才有，谷歌默认关闭，有 sensorHub 的手机可以开启该功能。

log 关键字: device\_idle

AP 刚进睡眠会有两次连续的 RTC 唤醒: calendar。

## 5.7 基于时间戳获取唤醒时间

以 SC9863A 平台为例，介绍基于 YLog 中 Kernel log 的时间戳，如何获取 Suspend 持续时间，以及系统 Resume 后工作多长时间后再次 Suspend。

基于 Kernel Log 时间戳获取挂起和唤醒时间的方法如下：

步骤 1 在 Kernel Log 中搜索如下关键字：Deep Sleep、Times 或者 wake up by，获取搜索结果。

例如搜索 Times 关键字得到一个如表 5-2 所示的搜索结果，可以通过这些搜索结果计算 Suspend/Resume 时间。

#### 说明

- 如果没有搜索到类似结果，需要先判断是否 Suspend 成功。
- Kernel4.14 版本中没有 Deep Sleep 和 Times 关键字，可以搜索 wake up by。
- 每一条搜索结果代表一次 Resume，在功耗电流上对应一次较大的电流突起。

表5-2 Kernel Log 搜索 Times 关键字的搜索结果

338[01-01 09:21:54.416] <6>[ 41.979119] c0	668 Deep Sleep 1 Times
709[01-01 09:22:00.414] <6>[ 44.223022] c0	668 Deep Sleep 2 Times
081[01-01 09:27:01.405] <6>[ 46.522937] c0	668 Deep Sleep 3 Times
457[01-01 09:30:01.406] <6>[ 48.822853] c0	668 Deep Sleep 4 Times
829[01-01 09:35:00.406] <6>[ 51.050731] c0	668 Deep Sleep 5 Times
204[01-01 09:40:01.397] <6>[ 53.282655] c0	668 Deep Sleep 6 Times
578[01-01 09:45:00.398] <6>[ 55.578581] c0	668 Deep Sleep 7 Times
062[01-01 09:50:01.427] <6>[ 61.114736] c0	668 Deep Sleep 8 Times
438[01-01 09:55:00.426] <6>[ 63.542594] c0	668 Deep Sleep 9 Times
825[01-01 10:00:00.433] <6>[ 65.862354] c0	668 Deep Sleep 10 Times
200[01-01 10:10:01.423] <6>[ 68.114218] c0	668 Deep Sleep 11 Times
576[01-01 10:15:00.423] <6>[ 70.430342] c0	668 Deep Sleep 12 Times
956[01-01 10:20:01.424] <6>[ 76.034267] c0	668 Deep Sleep 13 Times
337[01-01 10:25:00.430] <6>[ 78.358181] c0	668 Deep Sleep 14 Times
766[01-01 10:25:40.423] <6>[ 89.110065] c0	668 Deep Sleep 15 Times
246[01-01 10:30:01.424] <6>[ 91.417973] c0	668 Deep Sleep 16 Times
625[01-01 10:35:00.418] <6>[ 93.745850] c0	668 Deep Sleep 17 Times
005[01-01 10:40:01.422] <6>[ 96.049792] c0	668 Deep Sleep 18 Times
381[01-01 10:48:35.764] <6>[ 98.369692] c0	668 Deep Sleep 19 Times

步骤 2 计算 Resume 唤醒后 Kernel 工作多长时间。

Deep Sleep X Times Resume 唤醒后 Kernel 工作多长时间的计算公式如下：

$$W_{Deep\ Sleep\ x\ Times} = K_{Deep\ Sleep\ x+1\ Times} - K_{Deep\ Sleep\ x\ Times}$$

## 说明

$W_{Deep\ Sleep\ x\ Times}$ : Deep Sleep X Times resume 唤醒后 kernel 工作多长时间。

$K_{Deep\ Sleep\ x\ Times}$ : Deep Sleep X Times 对应 log 打印的 kernel 时间戳。

以表 5-2 的前四条搜索结果为例，计算从 Deep Sleep 1 Times 到 Deep Sleep 4 Times 之间前三次 Resume 唤醒后 Kernel 工作时间。

Deep Sleep 1 Times Resume 后再次 Kernel 工作时间是：44.223022 - 41.979119 约 2.24s。

Deep Sleep 2 Times Resume 后再次 Suspend 持续时间是：46.522937 - 44.223022 约 2.3s。

Deep Sleep 2 Times Resume 后再次 Suspend 持续时间是：48.822853 - 46.522937 约 2.3s。

三次计算结果分别与 Power Monitor 电流测试结果中三次较大电流突起的持续时间吻合，如图 5-2、图 5-3 所示。

图5-2 Power Monitor 电流测试图 1

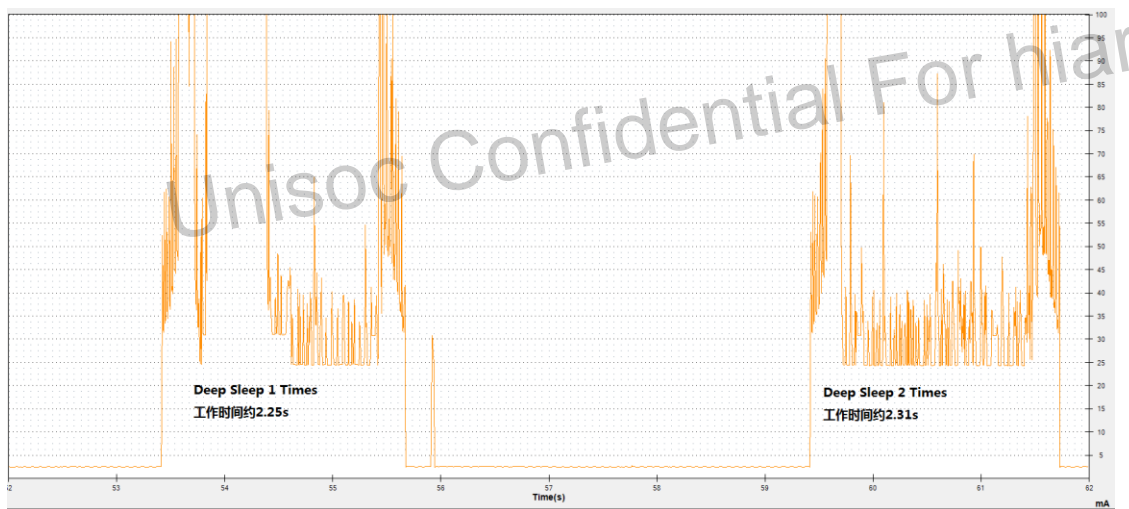
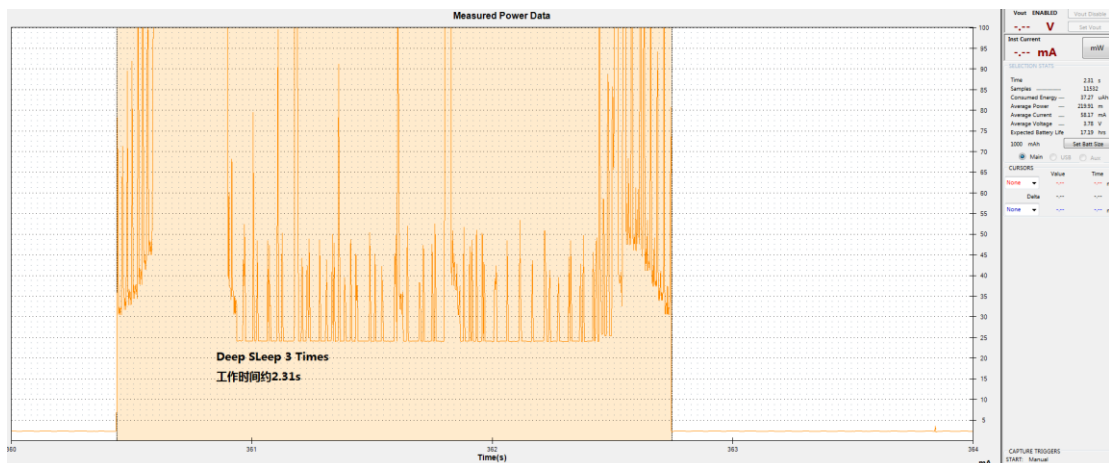


图5-3 Power Monitor 电流测试图 2



### 步骤 3 计算 Kernel Suspend 持续时间。

使用以下公式计算 Deep Sleep X Times Resume 后再次 Suspend 多长时间。

$$T_{Deep\ sleep\ x\ Times} = R_{Deep\ Sleep\ x+1\ Times} - R_{Deep\ Sleep\ x\ Times} - W_{Deep\ Sleep\ x\ Times}$$

#### 说明

$T_{Deep\ sleep\ x\ Times}$ : Deep Sleep X Times Resume 后再次 Suspend 多长时间。

$R_{Deep\ Sleep\ x\ Times}$ : Deep Sleep X Times 对应 log 打印的 RTC 时间戳。

同样以表 5-2 的前四条搜索结果为例，计算从 Deep Sleep 1 Times 到 Deep Sleep 4 Times 之间三次 Suspend 持续时间。

Deep Sleep 1 Times Resume 后再次 Suspend 持续时间：

01-01 09:22:00.414 - 01-01 09:21:54.416 - 2.24 约 3.76s。

Deep Sleep 2 Times Resume 后再次 Suspend 持续时间是：

01-01 09:27:01.405 - 01-01 09:22:00.414 - 2.3 约 298.7s。

Deep Sleep 2 Times Resume 后再次 Suspend 持续时间是：

01-01 09:30:01.406 - 01-01 09:27:01.405 - 2.3 约 177.7s。

三次计算结果分别与 Power Monitor 电流测试结果中三个持续时间吻合，如图 5-2、图 5-3、图 5-3 所示。



图5-4 Power Monitor 电流测试图 3

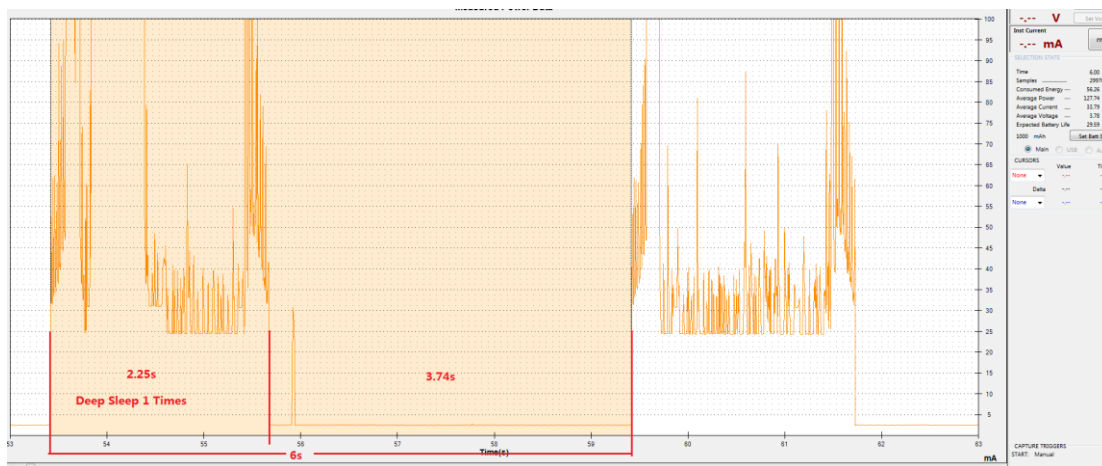


图5-5 Power Monitor 电流测试图 4

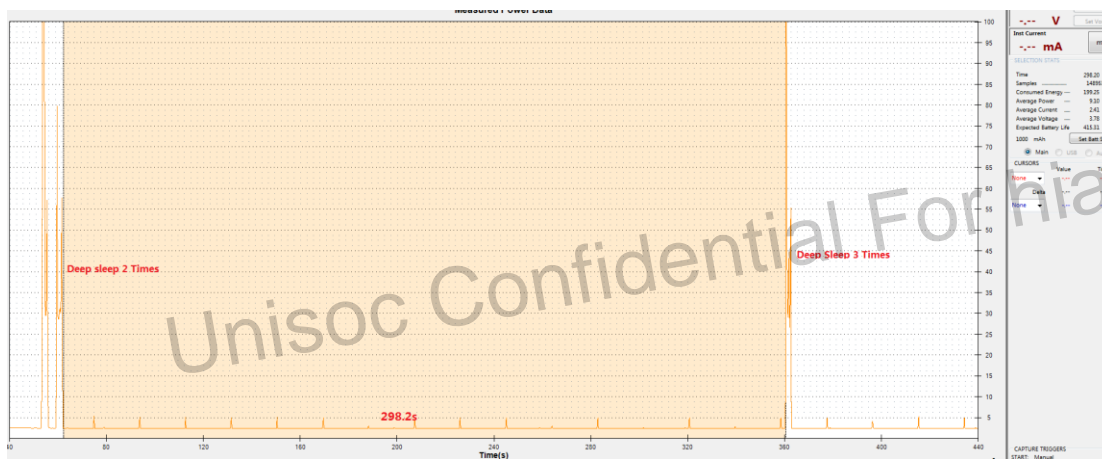
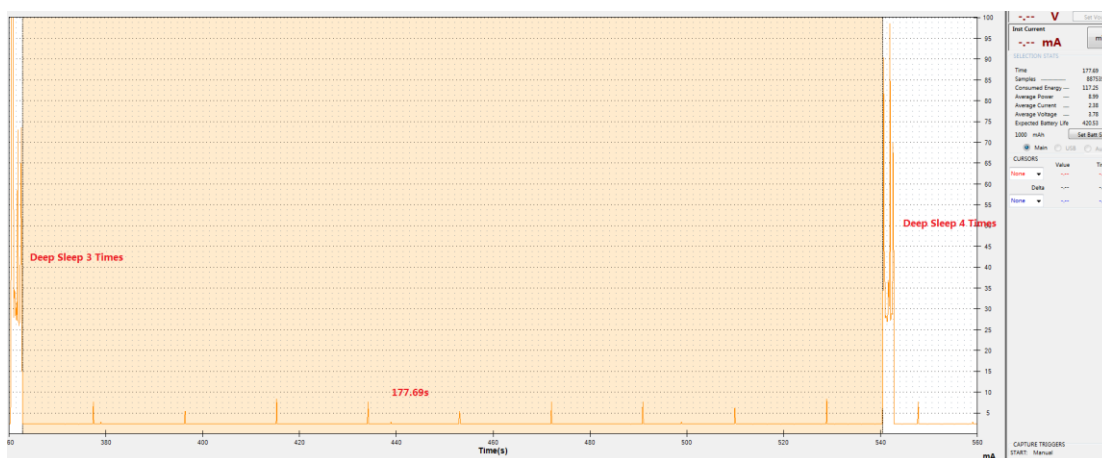


图5-6 Power Monitor 电流测试图 5



----结束

## 5.8 校准模式下关闭外设防止电流偏高的解决方案

校准模式是生产线上用于设备调试的一个特殊模式，可以通过测试软件向测试手机下发各种测试命令，如果接收到的是 Suspend 命令，则系统开始走 Suspend 流程。

校准模式下出现的电流问题，是由第三方驱动代码与平台参考机的差异所导致。对于平台参考机的驱动代码，通常已经按照 Suspend/Resume 流程进行过优化，而且会有 calibration 的条件判断；而客户端的各种驱动代码，一般没有专门适配 calibration，因此会出现问题。

目前在校准模式下出现功耗偏高问题的外设，主要有 Touch Panel、LCD、Charge 和指纹等模块，针对这几种外设，提供以下解决方案。

### 5.8.1 Touch Panel 类解决方案

对于 Kernel4.4 及以前的版本，请结合实际使用的 TP 驱动，参考以下 patch 进行修改：

vendor/sprd/modules/devdrv/input/touchscreen/focaltech/focaltech.c

```
extern bool in_calibration(void);
static int __init fts_init(void)
{
+ if (in_calibration()) {
+ printk("[FST] %s: probe cali_mode tp\n", __func__);
+ return -1;
+ }
return i2c_add_driver(&fts_driver);
}
```

对于 Kernel4.14 以后的版本，由于 bool in\_calibration(void)不可用了，参考以下 patch：

vendor/sprd/modules/devdrv/input/touchscreen/focaltech/focaltech.c

```
+static bool cali_mode;
+static int __init boot_mode_check(char *str)
+{
+    if (str != NULL && !strncmp(str, "cali", strlen("cali")))
+        cali_mode = true;
+    else
+        cali_mode = false;
+    return 0;
+}
+__setup("androidboot.mode=", boot_mode_check);
```



```
static int __init fts_init(void)
{
+   if (cali_mode) {
+       printk("[FST] %s: probe cali_mode_tp\n", __func__);
+       return -1;
+   }

    return i2c_add_driver(&fts_driver);
}
```

## 5.8.2 LCD 类解决方案

在校准模式下，LCD 驱动是不加载的。如果需要在校准模式下，机器启动加载软件的时候，加载 LCD 驱动，让 LCD 正常显示；并在启动完成后不再需要 LCD 显示时，将 LCD 模块掉电关闭。这种情况下可以添加如下的休眠动作。

vendor/sprd/proprieties-source/engmode/eng\_cmd4linuxhdlr.c

```
void *thread_fastsleep(void *para) {
    int count;
    int sleep = 0;

    char cmd[] = {"echo mem > /sys/power/state"};

    char cmd[] = {"echo 0 > /sys/class/adf/sprd-adf-dev-interface0/dpms_state && echo 3 > /sys/class/adf/sprd-adf-dev-interface0/dpms_state && echo mem > /sys/power/state"};

    ENG_LOG("##: please plug out usb within 60s...\n");
    for (count = 0; count < 60*5; count++) {
```

以上使用这个节点的前提是，在手机正常开机模式保持手机不灭屏状态下，通过如下命令能对手机进行显示的唤醒、休眠操作。

```
echo 0 > /sys/class/adf/sprd-adf-dev-interface0/dpms state
echo 3 > /sys/class/adf/sprd-adf-dev-interface0/dpms state
```

增加对 **display** 休眠唤醒节点权限的操作：

```
sprddroid8.1 trunk 18a rls2/device/sprd/sharkle/common/rootdir/root/init.common.rc
    chmod 666 /dev/adf0
    chmod 666 /dev/adf-interface0.0
    chmod 666 /dev/adf-overlay-engine0.0
    chown system system /sys/class/adf
    chmod 0666 /sys/class/adf/sprd-adf-dev-interface0/dpms state
on post-fs
insmod /lib/modules/mali.ko
```

## 5.8.3 Charge 类解决方案

进入 BBAT 或校准模式无法停止充电的原因可能是 charge ic 的控制方式发生了变化，导致 stop charge 没有生效。这里提供了使用 charge\_pd 的 sample code。

# /kernel/drivers/power/fan54015.c

```

@@ -207,8 +207,7 @@ void fan54015_otg_enable(int enable)

void fan54015_stop_charging(void)
{
- fan54015_set_value(FAN5405_REG_CONTROL1,
-                     FAN5405_CE_N, FAN5405_CE_N_SHIFT, DISCHARGER);
+ sprd_charge_pd_control(false);
}

unsigned char fan54015_get_vendor_id(void)
@@ -232,10 +231,7 @@ unsigned char fan54015_get_fault_val(void)

void fan54015_enable_chg(void)
{
- fan54015_set_value(FAN5405_REG_CONTROL1,
-                     FAN5405_CE_N, FAN5405_CE_N_SHIFT, DISCHARGER);
- fan54015_set_value(FAN5405_REG_CONTROL1,
-                     FAN5405_CE_N, FAN5405_CE_N_SHIFT, ENCHARGER);
+ sprd_charge_pd_control(true);
}

```

# /kernel/drivers/power/sprd\_charge\_2720\_helper.c

```

@@ -87,6 +87,14 @@ int glb_adi_write(u32 reg, u32 or_val, u32 clear_msk)
    clear_msk, or_val);
}

+int sprd_charge_pd_control(bool enable)
+{
+    if (enable)
+        return glb_adi_write(ANA_REG_GLB_CHGR_CTRL0, 0, BIT_CHGR_PD);
+
+    return glb_adi_write(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD, BIT_CHGR_PD);
+}
+

```

# /kernel/drivers/power/sprd\_charge\_2721\_helper.c -----

```

@@ -99,6 +99,16 @@ int glb_adi_write(u32 reg, u32 or_val, u32 clear_msk)
    clear_msk, or_val);

```

```

}

+int sprd_charge_pd_control(bool enable)
+{
+    if (enable)
+        return glb_adi_write(ANA_REG_GLB_CHGR_CTRL0, 0,
+                               BIT_CHGR_PD_EXT);
+
+    return glb_adi_write(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD_EXT,
+                          BIT_CHGR_PD_EXT);
+}

```

#### /kernel/drivers/power/sprd\_charge\_2723\_helper.c

```

@@ -97,6 +97,14 @@ int glb_adi_write(u32 reg, u32 or_val, u32 clear_msk)
    clear_msk, or_val);
}

+int sprd_charge_pd_control(bool enable)
+{
+    if (enable)
+        return glb_adi_write(ANA_REG_GLB_CHGR_CTRL0, 0, BIT_CHGR_PD);
+
+    return glb_adi_write(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD, BIT_CHGR_PD);
+}
+

```

#### /kernel/drivers/power/sprd\_charge\_helper.c

```

@@ -35,6 +35,8 @@
#define SAMPLE_NUM (1)
#define TEMP_BUFF_CNT (5)
#define CUR_RESULT_NUM (4)
+#define SC2731_CHG_CFG0 0x00
+#define SC2731_CHG_PD_BIT BIT(0)

static struct regmap *reg_map;
static int temp_buff[TEMP_BUFF_CNT] = { 200, 200, 200, 200, 200 };
@@ -97,6 +99,15 @@ int glb_adi_write(u32 reg, u32 or_val, u32 clear_msk)
    clear_msk, or_val);
}

```

```
+int sprd_charge_pd_control(bool enable)
+{
+    if (enable)
+        return glb_adi_write(SC2731_CHG_CFG0, 0, SC2731_CHG_PD_BIT);
+
+    return glb_adi_write(SC2731_CHG_CFG0, SC2731_CHG_PD_BIT,
+        SC2731_CHG_PD_BIT);
+}
```

/kernel/drivers/power/sprd\_charge\_helper.h

```
@@ -48,7 +48,7 @@ int sprdchg_bclp2_disable(int disable);
extern unsigned int glb_adi_read(u32 reg);
extern int glb_adi_write(u32 reg, u32 or_val, u32 clear_msk);
enum usb_charger_type sprdchg_charger_is_adapter(void);
+extern int sprd_charge_pd_control(bool enable);
#ifdef CONFIG_OTP_SPRD_PMIC_EFUSE
extern u32 sprd_pmic_efuse_block_read(int blk_index);
extern u32 sprd_pmic_efuse_bits_read(int bit_index, int length);
```

/u-boot15/drivers/power/battery/fan54015.c

```
@@ -284,12 +284,12 @@ void fan54015_ta_start_charging(void)

void sprdchg_fan54015_start_chg(int type)
{
-    fan54015_set_value(FAN5405_REG_CONTROL1, FAN5405_CE_N, FAN5405_CE_N_SHIFT, ENCHARGER);
+    sprd_charge_pd_control(true);
}

void sprdchg_fan54015_stop_charging(int value)
{
    printf("stop charge\n");
-    fan54015_set_value(FAN5405_REG_CONTROL1, FAN5405_CE_N, FAN5405_CE_N_SHIFT,
DISCHARGER);
+    sprd_charge_pd_control(false);
}
```

/u-boot15/drivers/power/battery/sprd\_chg.c

```
@@ -24,6 +24,16 @@ void sprdchg_stop_charge(int value)
```

```

    ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD, BIT_CHGR_PD);
}

+int sprd_charge_pd_control(bool enable)
+{
+    if (enable)
+        ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, 0, BIT_CHGR_PD);
+    else
+        ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD, BIT_CHGR_PD);
+
+    return 0;
+}
+
static void sprdchg_set_recharge(void)
{

```

#### /u-boot15/drivers/power/battery/sprd\_chg\_2720.c

```

@@ -24,6 +24,16 @@ void sprdchg_stop_charge(int value)
    ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD, BIT_CHGR_PD);
}

+int sprd_charge_pd_control(bool enable)
+{
+    if (enable)
+        ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, 0, BIT_CHGR_PD);
+    else
+        ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD, BIT_CHGR_PD);
+
+    return 0;
+}

```

#### /u-boot15/drivers/power/battery/sprd\_chg\_2721.c

```

@@ -24,6 +24,17 @@ void sprdchg_stop_charge(int value)
    ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD, BIT_CHGR_PD);
}

+int sprd_charge_pd_control(bool enable)
+{

```

```
+   if (enable)
+       ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, 0, BIT_CHGR_PD_EXT);
+   else
+       ANA_REG_MSK_OR(ANA_REG_GLB_CHGR_CTRL0, BIT_CHGR_PD_EXT,
+           BIT_CHGR_PD_EXT);
+
+   return 0;
+}
+
+static void sprdchg_set_recharge(void)
```

/u-boot15/drivers/power/battery/sprd\_chg\_2731.c

```
@@ -55,6 +55,18 @@
/* CHG_CFG5 */
#define BITS_ICRSET(x)      (((x) & 0x3) << 8)

+/* CHG_PD */
+#define SC2731_CHG_PD      BIT(0)
+
+int sprd_charge_pd_control(bool enable)
+{
+   if (enable)
+       return sci_adi_write(CHG_CFG0, 0, SC2731_CHG_PD);
+
+   return sci_adi_write(CHG_CFG0, SC2731_CHG_PD,
+       SC2731_CHG_PD);
+}
```

## 5.8.4 指纹类解决方案

针对常保供电的指纹模组，其供电不受开关和系统状态控制，只要开机有电就会供电，由于校准模式下不起 Android 系统，上层指纹服务无法启动，无法给指纹模组发进入低功耗指令，导致指纹无法进入低功耗模式。

### 说明

正常 android 模式下无上述问题，不影响最终产品功耗。

若 Android 模式下存在指纹导致的功耗问题，需另行分析处理。

由于修改方案涉及增加硬件成本、需第三方厂商配合提供方案且不同厂商方案不统一、Android 大系统无此问题，即不影响量产，因此软硬件不做调整。推荐放宽 **deepsleep spec**，测试减去指纹模组电流的贡献值。

Unisoc Confidential For hiar