# NV问题

## 1、SIM卡槽反了

### 平台：SC7731C、SC9832

ITEM\_CONTENT = 0x1000203

修改的文件路径例如为：modem\Tshark\TM\_BASE\_PHASE2\_TSHARK\_NV\FS066V20\_G4W15\td\_nv\_type.nvm

修改的内容为： ITEM\_CONTENT = 0x1000203   #其中0x1000203的他是对应4个卡槽，其实他是0x01000203前面的0被省略了，不同的颜色表示对应不同的卡槽，其中这个板子是对应4个卡槽，因此就有8个数字，如果此时SM1和SM2在实际现象中是相反的我们只要把01和00的位置对调就可以了，也就是ITEM\_CONTENT = 0x0001203

## 2、NV 6.0转到7.0使用

### 平台：SC7731C、SC9832

1、从6.0中拷贝相应的NV文件到7.0中。

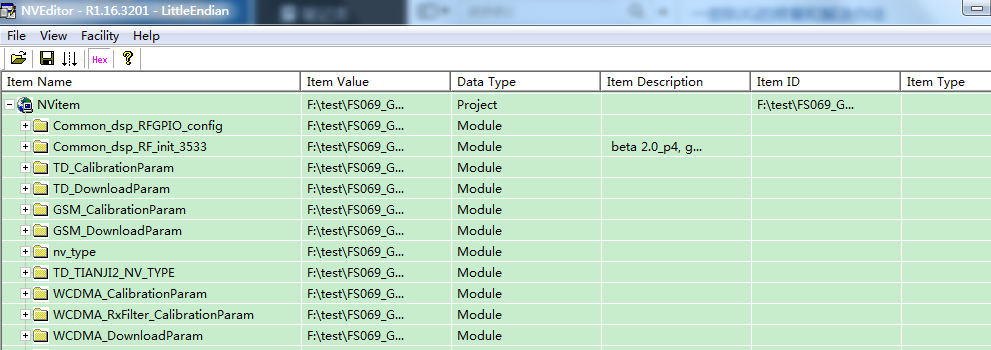
2、把nv外层的Android6.0\_Android7.0\_Changed.nv文件考到刚刚的项目中。

3、并从别的文件拷贝4个audio开头的文件到这个项目中。

4、在NV目录下面执行   patch -p1 <Android6.0\_To\_Android7.0.patch

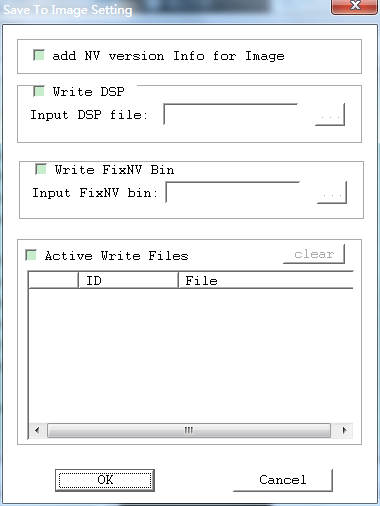
5、使用工具打开，工具路径：modem\SC7731C\Tools\NVEDITOR\Bin\NVEditor.exe

6、File -> Pro... 打开工程。



7、点击File -> save pro..保存工程

8、点击File -> save image 选项用默认的就可以【这一步会在刚才的目录下面生成一个.bin文件】。



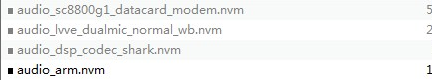
8、注意，新弄一个 新的NV,在base\_nv.list中要记得增加，不然编译过不了。

## 3、 深圳的硬件同事给回来的NV我们要做如下改动

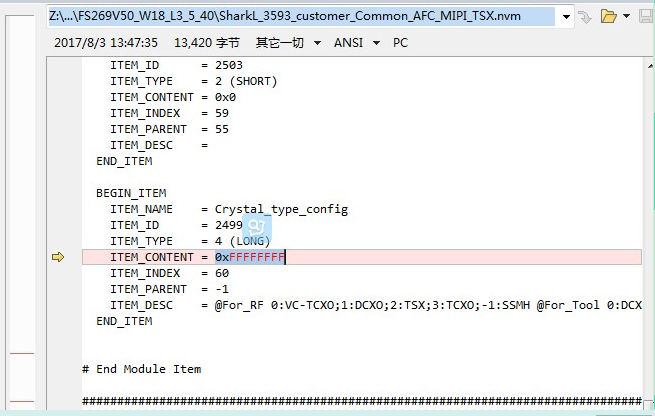
### 平台：SC7731C、SC9832

1、修改命名方式，命名方式以我们这边的为准，其中nv的第一个是板子的名字比如FS090。

2、修改5个文件，拷贝我们相同版型的文件深圳的nv中如下：



3、修改晶振，在下面的文件中的相应地方晶振改为和我们一样的。



## 4、如何制作NV差分文件和如何导入NV差分文件

**SC9832E SC9850**

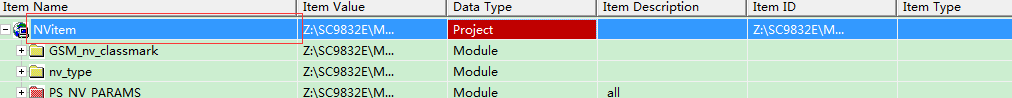
**NV工具路径：modem\Tools\NVEDITOR\Bin**

### （1）、如何制作NV差分包（一般是使用展讯的基础NV和要差分的NV）

1）、使用NV工具打开展讯基础NV（open project）。

2）、点击file->load image 导入另一个要差分的NV的bin文件。

3）、在红色框框处右击export出来相应的差分文件：



### （2）、如导入NV差分包

1）、用工具打开展讯基础NV（open project）。

2）、点击C:\Users\ADMINI~1\AppData\Local\Temp\Image.png选择里面的import导入.NV的差分文件（不用在乎什么名字）。

3）、保存工程C:\Users\ADMINI~1\AppData\Local\Temp\Image.png ---》save project 。

4）、生成bin文件File --- > save image。

## 5、MP3音频的代码位置

### 平台：SC7731C、SC9832、SC9850

路径：idh.code\device\sprd\sharkl2\sp9850ka\_1h10\rootdir\system\etc\audio\_params

## 6、修改音频模式

### 平台： SC9850

修改的地方：位于相应的dts中 sprd,spk-ext-pa-info = <0 0 2 0>; 其中数字2代表输出两个脉冲，处于模式2

## 7、L+L(双4G配置)

### 平台： SC9850

1. 双4G的modem和NV必须对应上，如果没有对应上会出现问题（比如：找不到串口。）。
2. 目前支持双4G的modem是18A ；NV有：

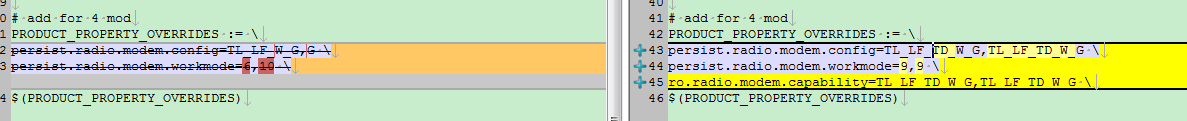
FS098\_W18\_L1\_3\_8\_40\_l558\_50a\_evercoss\_360os\_20180118

FS098\_W18\_L1\_3\_8\_l558\_50a\_evercoss\_360os\_20180117

FS098\_W18\_L1\_3\_8\_40

FS098\_W18\_L1\_3\_8

1. 配置双4G的步骤：
2. 在Config.mk中加入宏 ZCFG\_MODEM\_DIR=FM\_BASE\_18A\_W18.04.1\_9850ka\_CP0\_CP1 （也就是配置使用哪个modem）。
3. NV必须确认是支持双4G的。
4. 还有需要从大版本中拉出文件 idh.code\device\sprd\sharkl2\sp9850ka\_1h10\sp9850ka\_1h10\_oversea.mk 做如下的差分：



# add for 4 mod

PRODUCT\_PROPERTY\_OVERRIDES := \

persist.radio.modem.config=TL\_LF\_TD\_W\_G,TL\_LF\_TD\_W\_G \

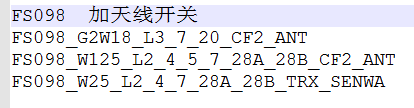
persist.radio.modem.workmode=9,9 \

ro.radio.modem.capability=TL\_LF\_TD\_W\_G,TL\_LF\_TD\_W\_G \

## 8、加了天线开关的NV如何处理

### 平台：SC7731C、SC9832、SC9850

处理方法：目前的处理方法是把加了天线开关的NV当做新的基础NV来处理，也就是在 base\_nv.list 中加入相应的NV。



# TP问题

### TP的引脚：电源、地、SDA、SCL、REST、中断脚 I2C通讯

## 1、三个虚拟按键的顺序反了

### 平台：SC7731C、SC9832

如果原有的触屏按键的顺序和现在的顺序是不一样的，或者说是去区域是不一样的，可以通过驱动中预留出来的宏进行修改，驱动路径：idh.code\kernel\drivers\input\touchscreen，在入口.c文件里面（如：focaltech.c）里面有一个函数virtual\_keys\_show，在这个函数下面就可以看到类似ZCFG\_TP\_ICN8XXX\_KEY\_MAP的宏，把这个宏拷贝到zyt.h中做如下定义#define ZCFG\_TP\_MSG2XXX\_KEY\_MAP {400,1298,60,40,240,1298,60,40,80,1298,60,40} 。脚本会自动解析放在相应项目下面的zyt.h文件，这个文件里面放着的是一些前辈们在改大版本中使用的一些宏，这些宏并不一定全部用到，但是如果用到在这里直接定义就可以了。相应宏的说明放在/zprj的目录下面的“zyt.h宏控列表.txt”。

/\*TP 虚拟键位置\*/

#define ZCFG\_TP\_MSG2XXX\_KEY\_MAP {400,1298,60,40,240,1298,60,40,80,1298,60,40}

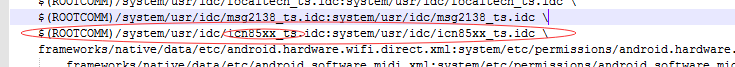
其中 400,1298,60,40,240,1298,60,40,80,1298,60,40 这一串数分别4个一组（x，y，x方向的范围，y方向的范围）；因此这一串数字代表了3个按键从左到右分别为：KEY\_MENU、KEY\_HOMEPAGE、KEY\_BACK。

## 2、调试tp的时候出现一个箭头导致触屏不可以用

### 平台：SC7731C、SC9832

注意手机的目录对应于：idh.code\device\sprd\scx35l\common\rootdir

这个箭头是默认的，因此必须对其进行转换，方法就是在rootdir\system\usr\idc 目录下面建立一个和挂载节点名字一样后缀为.idc的文件如：icn85xx\_ts.idc ，里面的内容是什么没有多大关系。但是为了实现大版本的统一，修改大版本的配置mk文件，让脚本自己创建（如果有修改到大版本的配置文件记得提交。）。修改的文件路径例如是在 idh.code\device\sprd\scx35l\sp9832a\_2h11 文件是sp9832a\_2h11\_volte.mk 修改里面的内容如下：



其中icn85xx\_ts表示节点的名字。

## 3、 厂家调整了TP的分辨率

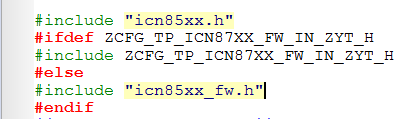
### 平台：SC7731C、SC9832

注意：相同的ic芯片可能可以做出很多分辨率的tp，这些分辨率厂家会调试。

1、厂家通常会提供类似于 icn85xx\_fw.h 文件。

2、将其拷贝到相应的目录下如：idh.code\kernel\drivers\input\touchscreen\icn87xx\，并把它的名字修改为相应的项目名称，比如：icn85xx\_fw\_FS088\_TRX\_L458.h

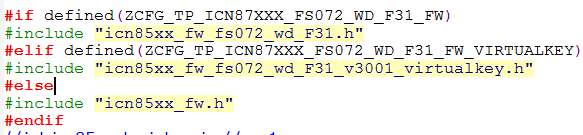
3、对 icn85xx\_fw.h 文件进行修改，这里有两种改法，方法一：在icn85xx.c中像下面这么写，



那么这个时候就要修改相应项目目录下面的zyt.h文件，加上下面这一句就可以了。

C:\Users\ADMINI~1\AppData\Local\Temp\Image.png

方法二：在icn85xx.c中像下面这么写



然后在zyt.h文件加上 #defined ZCFG\_TP\_ICN87XXX\_FS072\_WD\_F31\_FW\_VIRTUALKEY 这样的宏声明就可以了。

4、编译的手段就是编译bootimage,下载boot.img 。

## 4、关于TP手势

### 平台：SC7731C、SC9832

（1）、如何开启TP手势

1）、首先厂家给TP代码中应该有开启手势功能。

2）、在config.mk中打开下面的宏：C:\Users\ADMINI~1\AppData\Local\Temp\Image.png，这个宏是作用在比如idh.code\kernel\drivers\input\touchscreen\icn87xx\icn85xx.c .如果出现手势功能没有什么效果可能出现问题的地方有：icn85xx\_ts\_suspend 就是按电源键要进入低功耗模式的时候，必须开TP的终端，不能关闭。

（2）、MSG2XX手势调试经验

1）、通常每个TP厂给我们驱动的时候里面都会包含手势的代码并且通过宏来控制（关键词：GESTURE），例如这个TP的宏就是 CONFIG\_ENABLE\_GESTURE\_WAKEUP 。我们要做的就是通过我们本身的宏来控制这个宏的开关。例如在这个TP放头文件的文件（ilitek\_drv\_common.h）里面做如下定义 ：

#if defined(ZCFG\_MK\_TP\_GESTURE)

#define CONFIG\_ENABLE\_GESTURE\_WAKEUP

#endif

2）、找到该TP的suspend的函数（休眠时候跑的函数；该项目文件 ilitek\_drv\_main.c 函数： MsDrvInterfaceTouchDeviceFbNotifierCallback）在里面加上上层的开关调用接口如下：

定义：

static int s\_gesture\_switch = 1; // Defaultly, the macro is open

#define GESTURE\_FUNCTION\_SWITCH "/data/system/gesture\_switch"

实现开关：

fp = filp\_open(GESTURE\_FUNCTION\_SWITCH, O\_RDONLY , 0);

if (IS\_ERR(fp))

{

pr\_info("open file %s error!\n", GESTURE\_FUNCTION\_SWITCH);

s\_gesture\_switch = 1;

}

else

{

printk("open file %s success!\n", GESTURE\_FUNCTION\_SWITCH);

s\_gesture\_switch = 0;

filp\_close(fp, NULL);

}

然后通过s\_gesture\_switch这个变量去控制suspend和resume中关于手势的相关函数（相关函数一般给过来就驱动就有，大概为suspend的时候向TP发送指令进入手势模式，然后使能TP中断，resume的时候发送指令推出手势模式。），实现一个可变开关。

3）、实现各种手势上传键值

一、在probe的时候初始化键值如：

input\_set\_capability(input\_dev, EV\_KEY, KEY\_POWER);

二、找到原先TP驱动的键值中断函数（一般TP厂为了验证键值都是报KEY\_POWER）

如函数：\_DrvSelfParsePacket 中

找到读键值的地方实现函数 check\_gesture(int gesture\_id)

## 5、TP接近 光感

### 平台：SC7731C、SC9832

目前公司拥有的接近传感器芯片有3个epl259x\_cdc 、 stk3x1x\_cdc 、epl2182\_cdc 。

注意：如果要同事兼容epl259x\_cdc和stk3x1x\_cdc，需要在zyt.h中开宏#define ZYT\_CDC\_EPL\_COMPATIBLE\_WITH\_259X\_DTS

## 6、TP手势问题的解决方案

### 平台：SC7731C、SC9832

1、使用adb 查看是否有中断 cat /proc/interrupts查看开机的时候有几个中断，然后用手去触碰屏幕的时候看有几个中断。如果没有产生新的中断，那么可能得原因在：一、相应的中断没有打开。二、触发条件，保持低电平在200ms的时间不过，导致没有检查到。三、上层的手势开关没有打开。

2、通过log查看产生的中断函数中的相应键值有没有上报给上层。

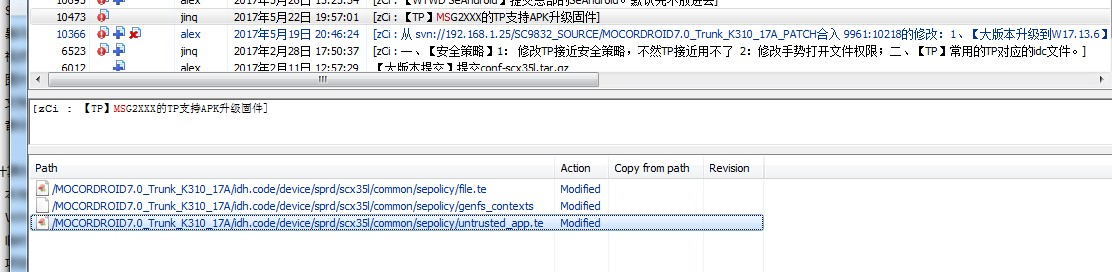
3、如果键值已经上报给上层了，但是没效果就由上层同事来完成。

## 7、  用APK升级TP固件（SEAndroid   /SELinux）

### 平台：SC7731C、SC9832

1、idh.code\device\sprd\scx35l\common\sepolicy 客制化权限路径。

2、idh.code\system\sepolicy 谷歌权限路径。



3、ls -Z 可以查看权限。

4、setenforce 0 关闭权限。

setenforce 1 //打开SEAndroid

## 8、修改传感器接近为TP接近方式（单步编译）

### 平台：SC7731C、SC9832

1、在Config.mk中修改相应的配置为CTP。

2、编译bootimage

3、进入idh.code\vendor\sprd\modules\sensors的目录执行mm，生成.so的路径在idh.code\out\target\product\sp9832a\_2h11\system\lib\hw\sensors.sc8830 （中间层），用adb工具push进去即可。

注意：查看是编译进去CTP还是CDC(光感)的方法进入路径：\idh.code\out\target\product\sp9832a\_2h11\obj\SHARED\_LIBRARIES\sensors.sc8830\_intermediates\pls 路径下的.o文件，如果是Pls\_CDC那么就是光感，如果是Pls\_CTP就TP接近。

## 9、TP手势深度睡眠无法唤醒

### 平台：SC7731C、SC9832、SC9850

配置在深度休眠状态下可被唤醒的中断需要注意如下:

### 1.申请中断时请置上IRQF\_NO\_SUSPEND选项，并且需要在suspend函数中设置中断为 电平触发，在resume函数中再恢复边沿触发。

例如：

申请中断：

int \_gIrq = -1;

static int MS\_TS\_MSG\_IC\_GPIO\_INT = 1;

MS\_TS\_MSG\_IC\_GPIO\_INT = of\_get\_named\_gpio\_flags(pDeviceNode, "mstar,irq-gpio",0, &nFlag);

gpio\_request(MS\_TS\_MSG\_IC\_GPIO\_INT, "C\_TP\_INT");

\_gIrq = gpio\_to\_irq(MS\_TS\_MSG\_IC\_GPIO\_INT);

nRetVal = request\_threaded\_irq(\_gIrq/\*MS\_TS\_MSG\_IC\_GPIO\_INT\*/, NULL, \_DrvFingerTouchInterruptHandler,

/\*IRQF\_TRIGGER\_RISING\*/ IRQF\_TRIGGER\_FALLING | IRQF\_ONESHOT/\* | IRQF\_NO\_SUSPEND \*/,"msg2xxx", NULL);

if (nRetVal != 0)

{

DBG(&g\_I2cClient->dev, "\*\*\* Unable to claim irq %d; error %d \*\*\*\n", \_gIrq, nRetVal);

}

suspend函数中

irq\_set\_irq\_type(\_gIrq,IRQF\_TRIGGER\_LOW | IRQF\_NO\_SUSPEND | IRQF\_ONESHOT);

resume函数中：

irq\_set\_irq\_type(\_gIrq,IRQF\_TRIGGER\_FALLING | IRQF\_NO\_SUSPEND | IRQF\_ONESHOT);

参数解析(详见下面---关于request\_threaded\_irq----中断线程化)：

\_gIrq ：中断号

\_DrvFingerTouchInterruptHandler ：中断处理函数

IRQF\_TRIGGER\_FALLING：下降沿触发中断

IRQF\_TRIGGER\_RISING ：上升沿触发中断

IRQF\_ONESHOT ：在没有执行 \_DrvFingerTouchInterruptHandler 这个线程前不会接收其他中断

IRQF\_NO\_SUSPEND :在系统挂起的时候，这个中断不挂起。

msg2xxx ： 中断的名字

### 2.由于休眠状态下，clock时钟一般为32K，所以需要保证中断的保持时间；比如电平中断建议持续时间要超过200ms （这个需要TP厂商修改固件实现）

### 3.pinmap文件中的TP中断引脚需配置成在睡眠状态的IE（输入时能）

例如：

把pinmap文件中的

{REG\_PIN\_SIMDA2, BIT\_PIN\_SLP\_AP|BIT\_PIN\_NULL|BITS\_PIN\_DS(1)|BITS\_PIN\_AF(3)|BIT\_PIN\_WPU|BIT\_PIN\_SLP\_WPU|BIT\_PIN\_SLP\_Z},//CTP\_INT

改为

{REG\_PIN\_SIMDA2, BIT\_PIN\_SLP\_AP|BIT\_PIN\_NULL|BITS\_PIN\_DS(1)|BITS\_PIN\_AF(3)|BIT\_PIN\_WPU|BIT\_PIN\_SLP\_WPU|BIT\_PIN\_SLP\_IE},//CTP\_INT

### 4、增加超时锁

有一种情况就是，在中断处理函数中，当有一个手势处理到一半的时候，系统突然休眠了。这个时候寄存器的值就有可能被压入栈中，等到下一次中断来时在上传。会出现一种现象就是划手势的时候很容易出现上一次的值。所以需要做的是在进入中断的时候上一把定时锁，具体做法如下：

定义一个变量：

struct wake\_lock suspend\_gestrue\_lock;

Proble中初始化一个锁：

wake\_lock\_init(&suspend\_gestrue\_lock, WAKE\_LOCK\_SUSPEND, "suspend\_gestrue");

Handler中断路口中打开一把定时锁：

wake\_lock\_timeout(&suspend\_gestrue\_lock, msecs\_to\_jiffies(2500));

### 5 请在GTP\_GESTURE\_WAKEUP 进入doze模式后，msleep一下，进行测试

#if GTP\_GESTURE\_WAKEUP

if (DOZE\_ENABLED == doze\_status)

{

GTP\_INFO("guangyao enter \n");

msleep(40);

ret = gtp\_i2c\_read(i2c\_connect\_client, doze\_buf, 3);

GTP\_DEBUG("0x814B = 0x%02X", doze\_buf[2]);

if (ret > 0)

{

### 6、8.1之后把 enable\_irq (this\_client->irq); 替换成 enable\_irq\_wake(this\_client->irq); 同理 把disable\_irq(this\_client->irq); 替换成 disable\_irq\_wake(this\_client->irq);

### 7、在读完数据后发IIC命令给 TP 让TP 释放INT，然后重新把INT中断切回电平触发

static int fts\_read\_Gestruedata(void)

{

。。。

fts\_write\_reg(this\_client, 0xd5, 0x00); // let TP release INT pin

。。。

}

static int touch\_event\_handler(void \*unused)

{

。。。

irq\_set\_irq\_type(this\_client->irq,IRQF\_TRIGGER\_LOW | IRQF\_NO\_SUSPEND | IRQF\_ONESHOT);

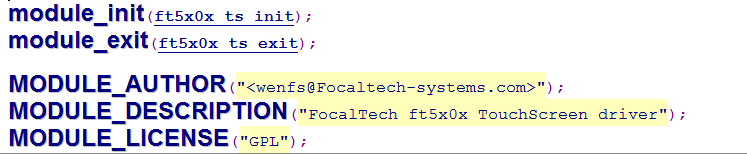
。。。

}

8、i2c进入suspend 模式导致TP通过gpio中断唤醒系统的时候I2C无法被唤醒读取数据失败

## 10、就墩太TP深入解刨TP驱动

**（1）、常规的模块声明和引入init和exit的入口**



**（2）、ft5x0x\_ts\_init 和 ft5x0x\_ts\_exit的实现**

static const struct of\_device\_id focaltech\_of\_match[] = {

{ .compatible = "focaltech,focaltech\_ts", }, //与dts里面的节点的 compatible 值一样

{ }

};

MODULE\_DEVICE\_TABLE(of, focaltech\_of\_match);

static struct i2c\_driver ft5x0x\_ts\_driver = {

.probe = ft5x0x\_ts\_probe,

.remove = ft5x0x\_ts\_remove,

.id\_table = ft5x0x\_ts\_id,

.driver = {

.name = FOCALTECH\_TS\_NAME,

.owner = THIS\_MODULE,

.of\_match\_table = focaltech\_of\_match,

},

static int \_\_init ft5x0x\_ts\_init(void)

{

pr\_info("==ft5x0x\_ts\_init==\n");

if(tp\_device\_id(0)!=0)

{

printk("CTP(0x%x)Exist!", tp\_device\_id(0));

return -ENODEV;

}

return i2c\_add\_driver(&ft5x0x\_ts\_driver); //注册一个i2c设备

}

static void \_\_exit ft5x0x\_ts\_exit(void)

{

i2c\_del\_driver(&ft5x0x\_ts\_driver); //删除已经注册的I2C设备

}

**（3）、跑ft5x0x\_ts\_probe**

static int ft5x0x\_ts\_probe(struct i2c\_client \*client, const struct i2c\_device\_id \*id)

{

1、定义一些保存数据的结构体：

struct ft5x0x\_ts\_data \*ft5x0x\_ts;

struct input\_dev \*input\_dev;

struct ft5x0x\_ts\_platform\_data \*pdata = client->dev.platform\_data;

2、取出dts中的数据存放在 pdata 中。

3、执行硬件初始化

static void ft5x0x\_ts\_hw\_init(struct ft5x0x\_ts\_data \*ft5x0x\_ts)

{

struct regulator \*reg\_vdd;

struct i2c\_client \*client = ft5x0x\_ts->client;

struct ft5x0x\_ts\_platform\_data \*pdata = ft5x0x\_ts->platform\_data;

int ret = 0;

1）、初始化rest脚和inq脚

gpio\_request(pdata->irq\_gpio\_number, "ts\_irq\_pin");

gpio\_request(pdata->reset\_gpio\_number, "ts\_rst\_pin");

gpio\_direction\_output(pdata->reset\_gpio\_number, 1);

gpio\_direction\_input(pdata->irq\_gpio\_number);

2）、上电：

reg\_vdd = regulator\_get(&client->dev, pdata->vdd\_name);

if (!WARN(IS\_ERR(reg\_vdd), "[FST] ft5x0x\_ts\_hw\_init regulator: failed to get %s.\n", pdata->vdd\_name)) {

if(!strcmp(pdata->vdd\_name,"vdd18"))

regulator\_set\_voltage(reg\_vdd,1800000,1800000);

if(!strcmp(pdata->vdd\_name,"vdd28"))

regulator\_set\_voltage(reg\_vdd, 2800000, 2800000);

ret=regulator\_enable(reg\_vdd);

if (ret) {

printk("ft5x0x\_ts\_hw\_init:regulator\_enable fail\n");

}

}

}

3）、做reset动作

）

（4）、如果有TP接近：

# 摄像头问题

## 1、厂家提供了一个新的摄像头驱动

### 平台：SC7731C、SC9832

（andriod 5.1 、6.0）

1、 通常厂家会给两个的文件如：sensor\_hi553\_raw\_param.c 、sensor\_hi553\_raw.c 。（如果FAE拿回去调效果）

2、在zprj/zsensor\_src 目录下面新建一个和本项目名称相关的目录，比如 sensor\_hi553\_mipi\_raw\_FS066\_CF8 。

3、把刚刚的两个文件拷贝到新建的目录中，名字也做相应的修改：sensor\_hi553\_raw\_param.c --> sensor\_hi553\_mipi\_raw\_FS066\_CF8\_param.c sensor\_hi553\_raw.c --> sensor\_hi553\_mipi\_raw\_FS066\_CF8.c

4、在文件中修改两个地方：（1）、#include "sensor\_hi259\_mipi\_raw\_FS066\_CF8\_param.c" （2）、#define g\_hi259\_mipi\_raw\_info g\_sensor\_hi259\_mipi\_raw\_FS066\_CF8\_info

5、编译和下载方法：

adb root

adb remount

adb push /patch/lib /system/vendor/lib

adb reboot

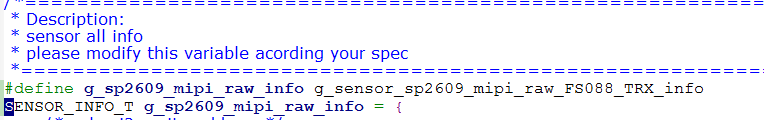
（andriod 7.0）

1、 通常厂家会给一些文件的参数文件如：NR文件夹 、isp\_nr.h 、sensor\_sp2609\_mipi\_raw.c、sensor\_sp2609\_raw\_param\_cap\_0.c 、sensor\_sp2609\_raw\_param\_common.c 、sensor\_sp2609\_raw\_param\_main.c、sensor\_sp2609\_raw\_param\_prv\_0.c、sensor\_sp2609\_raw\_param\_video\_0.c

2、在zprj/zsensor\_src 目录下面新建一个和本项目名称相关的目录，比如 zsensor\_src\sensor\_sp2609\_mipi\_raw\_FS088\_TRX

3、把刚刚客户提供的文件考到这个文件夹下面，然后修改 sensor\_sp2609\_mipi\_raw.c --> sensor\_sp2609\_mipi\_raw\_FS088\_TRX.c

4、在文件 sensor\_sp2609\_mipi\_raw\_FS088\_TRX.c中增加一句：



5、编译和下载方法在“编译与调试中有提到”。

## 2、摄像头无法正常工作的解决方法

### 平台：SC7731C、SC9832

== nanjun.zhong@spreadtrum.com On 2017-09-11T08:44:14Z (json) ==

Hi 张工，如电话沟通，如果其他项目同样的平台和版本是能打开的，这个考虑硬件问题，从log看，确实只是timeout。另外，在一开始就有中间层的如下log。

A001-01 08:24:28.563 291 2407 I SPRDAVCEncoder: H264EncStrmEncode[2874] 7ms, in {0xaf320000-0x20235000, 640x960}, out {0xb0086000-19504, 1}, wh{0, 0}, xy{0, 0}

A001-01 08:24:28.563 291 2407 I SPRDAVCEncoder: onQueueFilled, 1916, out\_stream\_ptr: 0xaef24000

对于0x10004错误：

原因一般是mipi时序配置sensor 和平台没有匹配好，可以做如下修改：

1 可以通过调整bps以及sensor的HS\_prepare和HS\_zero验证（需要fae检查）；

2 可以在stream\_on和stream\_off的写寄存器前和后都加上150mS的延时，在xxxx\_stream\_on函数中将打开mipi开始放数据的寄存器不能在其他地方打开，比如不能在init寄存器里面打开。

3 对于预览preview的丢帧可以多丢两帧，比如可以丢4帧

4 如果以上都不行的话，需要找原厂用示波器测量，看看mipi clk是否匹配正确，mipi clk和bps的计算方法是：

bps = OPCLK \* 10 / lane数

## 3、摄像头修改没有的像素

### 平台：SC7731C、SC9832

CAMERA\_SUPPORT\_SIZE

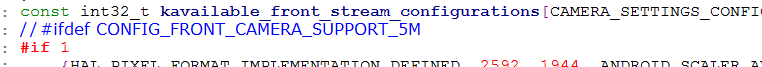
CONFIG\_CAMERA\_SUPPORT\_5M

vendor\sprd\modules\libcamera\hal3\SprdCamera3Setting.cpp

idh.code\vendor\sprd\modules\libcamera\SprdCtrl.mk

idh.code\device\sprd\scx35l\sp9832a\_2h11\BoardConfig.mk

前摄像素：



后摄像素：

C:\Users\ADMINI~1\AppData\Local\Temp\Image.png

## 4、摄像头出现上下左右反

### 平台：SC7731C、SC9832

修改摄像头的驱动文件如 sensor\_gc2385\_mipi\_raw\_FS089\_TRX 里面有以下宏：

下面的四个宏不能够同时打开

//#define IMAGE\_NORMAL\_MIRROR //开这个宏是厂家默认的

//#define IMAGE\_H\_MIRROR //如果是上下反开这个宏

//#define IMAGE\_V\_MIRROR //如果是左右反就开这个宏

#define IMAGE\_HV\_MIRROR //上下左右都反开这个宏

## 5、假双摄相关备注

### 平台：SC7731C、SC9832

代码路径： sys/bus/platform/devices/gc0310-user/

Config.mk中打开 CUSTOM\_MAIN2\_IMGSENSOR= ASSISTANT\_CAM

节点路径：

1：上层操作这两个文件，打开后摄像头的时候open一下 gc0310\_init 这个文件

/sys/bus/platform/devices/gc0310-user/gc0310\_init

2：读取摄像头AE值

/sys/bus/platform/devices/gc0310-user/gc0310\_access

## 6、I2C地址读取不到

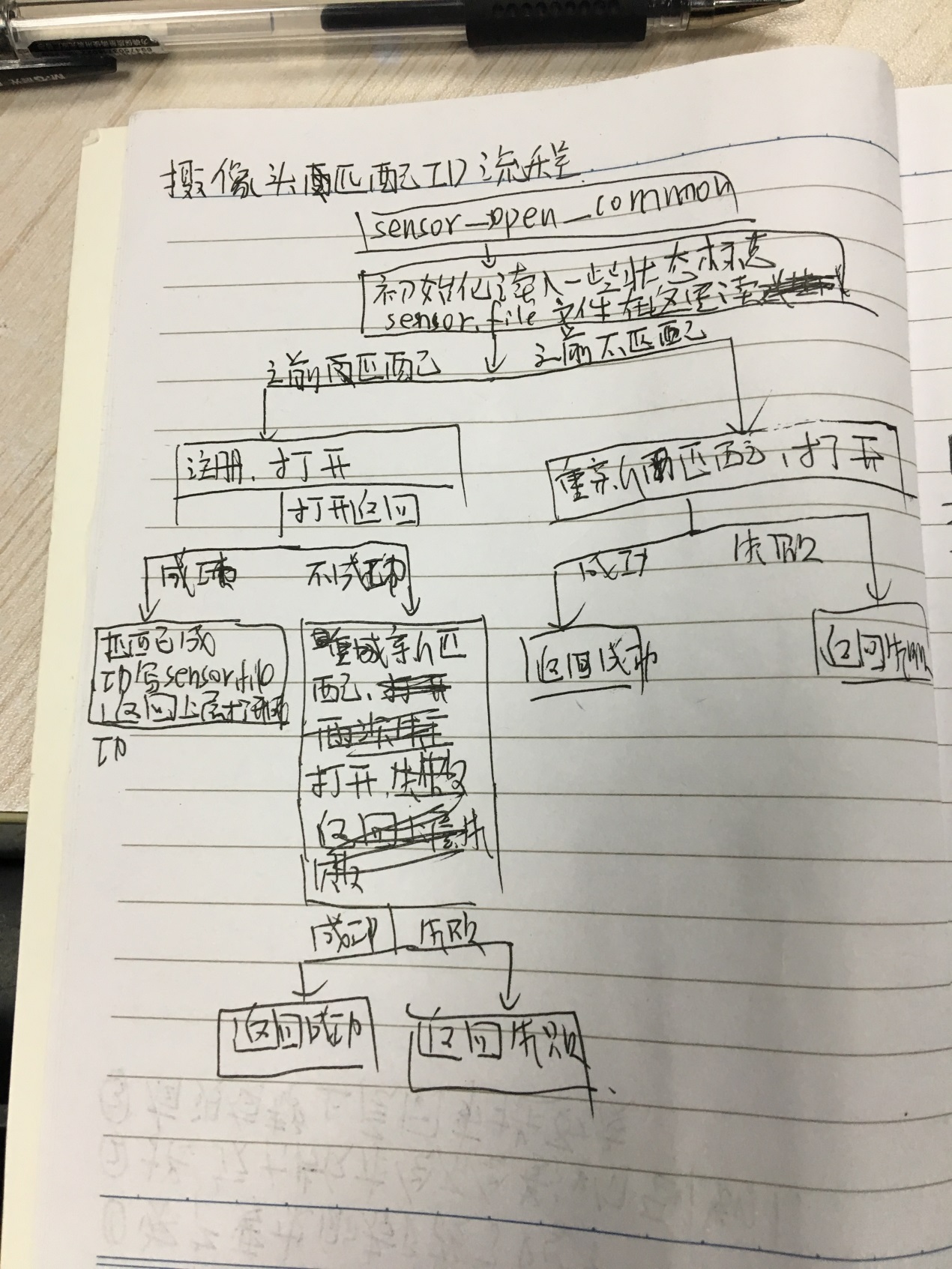
### 平台：SC7731C、SC9832、9850

存在这个问题，应该是由下面的3种情况造成的：

1. 这个摄像头可能存在多个ID（ 比如hi556(sp5509) ）；（尝试解决的方法有：1）、从别的已经点亮的同一个ic的驱动中找到相应的id是否可以对应的上。2）、询问相应厂商的FAE进行确认。）
2. 摄像头的相应上电时序不对，这个时候可以向FAE要相应的上电时序；上电是在函数 xx\_ power\_on 中。
3. 如果前面的两种情况都是正确的话，那么很有可能是摄像头的模组管脚有问题，这个时候可以叫项目进行确认。

## 7、摄像头调用步骤

### 平台：SC7731C、SC9832



## 8、闪光灯

### 平台：SC7731C、SC9832、SC9850

**1、软件路径：**

SC9832 SC7731C 平台：

底层驱动代码路径：kernel\drivers\media\sprd\_dcam\common\

其中的主要实现文件是dcam\_img.c ，其他的文件是真闪假闪的寄存器操作，通过Confg.mk的配置，来决定是否编入工程（各个文件的接口是一样的供给dcam\_img.c调用。）

HAL层代码路径：idh.code\vendor\sprd\modules\libcamera\oem\src 或者 idh.code\vendor\sprd\modules\libcamera\oem2v0\src 里面的 cmr\_oem.c 文件。

SC9850平台：

底层驱动代码路径：idh.code\kernel\drivers\misc\sprd\_camera\flash

**2、闪光灯实现的一些知识：**

（1）、注意：设置为自动拍摄的时候需要对闪光灯做一些处理：

1. 如果是raw的摄像头驱动不需要进行处理，效果参数里面有处理这一个问题。
2. 如果是yue的摄像头，那么闪光灯驱动我们必须自己写，这种芯片摄像头的效果都是在芯片内部完成的。 我们需要在驱动文件中完成函数 \_BF3905\_flash （读取当前摄像头的感光值，然后判断相应的感光值是否需要调用闪光灯。）

（2）、驱动实现的一些接口（目前我司是使用flash\_gpio.c 这种控制方式）：

## 9、sp5509(hi556) 和 sp2509方向调试的寄存器

### 平台：SC7731C、SC9832

背景:这两个IC没有开出相应的宏让我们修改方向，和FAE沟通得知只要在寄存器**{0x000e, 0x0200},** 中写入相应的值既可以，后面的0x0200的第二为分别写入0 1 2 3表示不同的方向。

注意：修改到方向寄存器，这个时候摄像头的颜色可能会有所改变，改变变量.image\_pattern的值分别为 SENSOR\_IMAGE\_PATTERN\_RAWRGB\_GR SENSOR\_IMAGE\_PATTERN\_RAWRGB\_GB SENSOR\_IMAGE\_PATTERN\_RAWRGB\_R SENSOR\_IMAGE\_PATTERN\_RAWRGB\_R这几个宏来调整。

## 10、使用ISP tool工具导出来的摄像头驱动和实际编译进去的名字不一样

### 平台：SC7731C、SC9832、SC9850

原因：就是编译进去的摄像头驱动的效果参数有问题。

展讯建议的解决方法：这个问题产生的原因是可能是原来的效果参数中有些参数不正确引起的。以后遇到这个问题，请FAE使用ISP tool工具的auto tuning工具重新生成一套参数，基于工具生成的参数来调试可以确保参数没有问题（我发你的参数是基于auto tuning工具生成的参数）。我的参数并不是基于原来的参数修改的，而是重新生成的效果参数。原来的参数有点问题，sensor size和line time配置的不正确，和原来的参数不一样。效果可能会差距比较大。请FAE在camera调试之前，仔细检查效果参数的配置是否正确，以免后期发现问题，问题反复。

谢谢！

## 11、单摄和真双摄的配置

### 平台：SC9850

单摄：

ARCH\_NAME=arm

VENDOR\_NAME=sprd

BRANCH\_NAME=sharkl2

BOARD\_NAME=sp9850ka\_1h10

BASE\_MK\_NAME=sp9850ka\_1h10\_native

MK\_FILE\_NAME=sp9850ka\_1h10\_oversea.mk

DTS\_FILE\_NAME=sp9850ka-1h10-native

真双摄：

ARCH\_NAME=arm

VENDOR\_NAME=sprd

BRANCH\_NAME=sharkl2

BOARD\_NAME=sp9850ka\_2c30

BASE\_MK\_NAME=sp9850ka\_2c30\_native

MK\_FILE\_NAME=sp9850ka\_2c30\_sbs.mk

DTS\_FILE\_NAME=sp9850ka-2c30-sbs

## 12、9850的摄像头配置

### 平台： SC9850

摄像头配置笔记：zcommon\docs\FAQ\Camera

## 13、调试摄像头效果的时候使用adb 拍摄jpg 图和抓到raw数据

（1）、开启抓取图片命令（设 置 后 相 机 拍 照 会 同 时 拍 下 raw 图 与 jpg 图。拍下的 raw 图 可 以 在 手 机 的 /data/misc/cameraserver目录下找到）

**adb shell setprop persist.sys.camera.raw.mode raw**

（2）关闭抓取图片命令

**adb shell setprop persist.sys.camera.raw.mode jpeg**

## 14、真双摄调试

### （1）、如何从手机中导出OTP

adb shell setprop debug.camera.save.otp.raw.data 1

adb shell stop cameraserver

adb shell start cameraserver

adb pull /data/vendor/cameraserver/xxx\_mipi\_raw\_xxx\_otp\_dump.bin //(名字和具体摄像头有关)

### （2）、临时验证input\_parameters\_values.txt是否有效

adb root

adb remount

adb push input\_parameters\_values.txt system/etc/otpdata

adb reboot

## 15、9863后双摄切换到副摄的方法

切换副摄的命令和方法：

1.先进入主摄，设置分辨率，该分辨率要小于副摄的分辨率，以SC9863为例，可以设置为（4：3） 1.9M

2.切换camera id，9.0命令：adb shell setprop persist.vendor.cam.id 2（0：后主摄，2：后副摄）

3.退出camera，再进入camera，就是副摄了（可以用遮挡camera的方法确认下）

4.恢复主摄命令：adb shell setprop persist.vendor.cam.id 0，重启手机。

## 16、9863A双摄项目如何确认当前版本设置RMS的门限值

经过verify test pass之后抓取log可以再log中得到关键字rms 0.51660 is less than rms threshold 1.50000.

## 17、9863A如何确认慢录是不是90帧

在以下函数添加log：

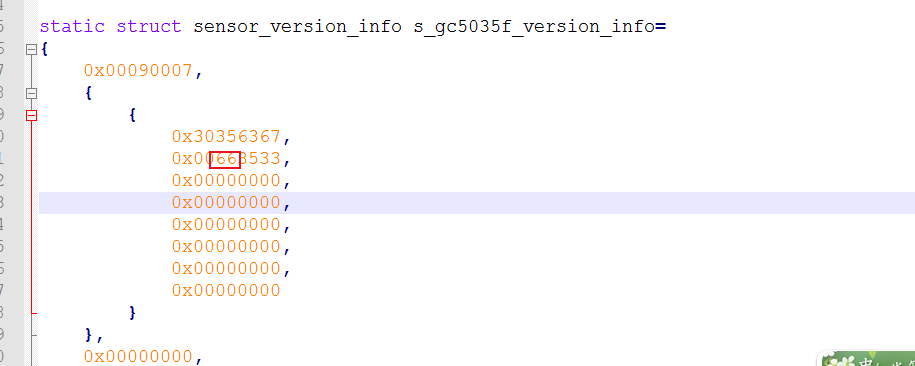
\*\kernel4.4\drivers\misc\sprd\_camera\dcam\_if\_r5p0\_isp\_r6p12\src\dcam\_int.c  
  
static void sprd\_dcamint\_cap\_sof(enum dcam\_id idx,  
    enum dcam\_irq\_id irq\_id, void \*param)  
{  
    dcam\_isr\_func user\_func;  
    void \*data;  
    struct dcam\_module \*module = (struct dcam\_module \*)param;  
    struct camera\_frame frame;  
    struct dcam\_cap\_desc \*cap\_desc = sprd\_dcam\_drv\_cap\_get(idx);  
  
    if (DCAM\_ADDR\_INVALID(module))  
        return;  
+   printk("sprd\_dcamint\_cap\_sof+++++\n");  
.  
.  
.  
}

其中每一次取画面都会进入到这个函数，所以可以通过1S钟打印出了多少log来大致推算出目前的帧率。

## 18、前后兼容相同的IC的摄像头

一般的处理方法是把前摄的摄像头进行改名在原有名字的基础上面后面加一个f，当然不管是文件名还是文件里面的内容都必须改，这里以gc5025的IC为例。首先需要把所有的gc5025改成gc5025f。

有了上面还不够，工具导出来的效果参数还是会显示成gc5025，这个时候需要改到 sensor\_gc5025f\_raw\_param\_main.c 里面的内容。



其中的66就是新加的f，默认是00.

## 19、如何查看录像模式的帧率是动态帧率还是固定帧率以及动态帧率应该如何修改范围

（1）如何判断动态帧率还是固定帧率

1）、adb 输入

adb root

adb remount

adb shell

logcat | grep “setCameraPreviewMode”

2）、打开摄像头并切换进入录像模式（前后摄像头都可以）从adb中会得到以下结果



----------->由于min\_fps == max\_fps 所以他的帧率是被固定在30了。



----------->设置为浮动帧率fps 15-30之间

（2）、如何修改动态帧率的值

diff --git a/hal3\_2v1/SprdCamera3OEMIf.cpp b/hal3\_2v1/SprdCamera3OEMIf.cpp

index 1e452a2..45332fa 100644

--- a/hal3\_2v1/SprdCamera3OEMIf.cpp

+++ b/hal3\_2v1/SprdCamera3OEMIf.cpp

@@ -1974,6 +1974,20 @@ void SprdCamera3OEMIf::setCameraPreviewMode(bool isRecordMode) {

fps\_param.max\_fps = 30;

#endif

}

+#else

+ bool changeFPS = false;

+ if ((mSprdAppmodeId != -1) && (mSprdAppmodeId != CAMERA\_MODE\_TIMELAPSE)) {

+ /\*dream camera2\*/

+ changeFPS = true;

+ }

+

+ if (changeFPS && (controlInfo.ae\_mode != ANDROID\_CONTROL\_AE\_MODE\_OFF)) {

+ /\* change the min fps as you want.\*/

+ fps\_param.min\_fps = 15; //这个就是 min\_fps =15 max\_fps =30

+ /\* use the max fps from APP.\*/

+ fps\_param.max\_fps = MAX(fps\_param.max\_fps,fps\_param.min\_fps);

+ }

+ HAL\_LOGI("changeFPS=%d,min:%d,max:%d",changeFPS,fps\_param.min\_fps,fps\_param.max\_fps);

#endif

// when 3D video recording with face beautify, fix frame rate at 25fps.

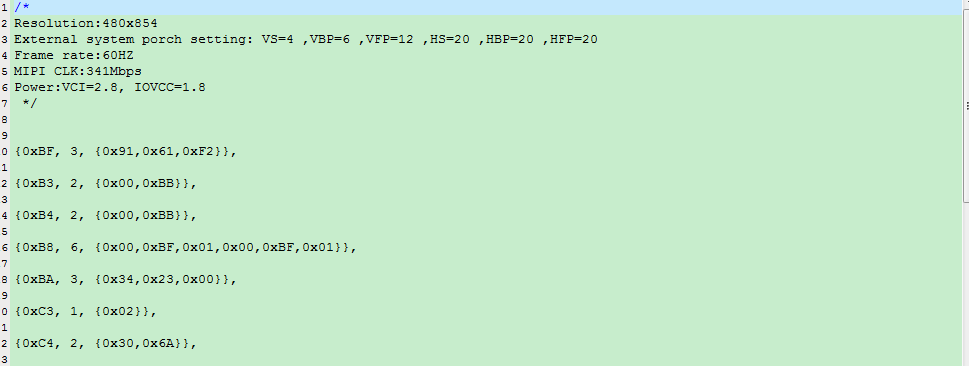
# LCM相关问题

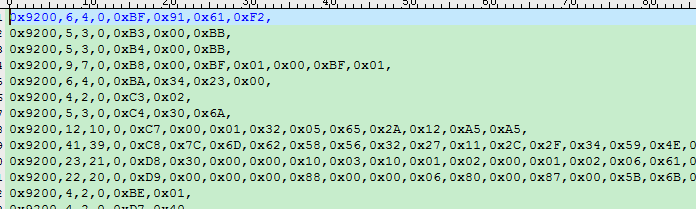
## 厂家给资料来配置UDC

### 平台：SC7731C、SC9832 SC9850

路径：UDC路径  idh.code\kernel\drivers\udc  屏路径   idh.code\kernel\drivers\video\sprdfb\lcd

1、厂家会给类似于C\_JD9161\_CTC4.5\_TN(PH045NA)\_480X854\_2dot\_G2.2\_160418\_AN\_MTK.txt这样的文件，里面的内容是：

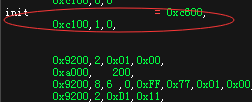
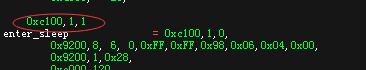


然后我们要对这些数据进行处理为下面这样子：

其中 0x9200后面跟的是数据，0xa000后面跟的是延时时间单位，0x9200后面跟着的有两种情况，如果后面的数据大于2那么他就是如 0x9200，6,4，跟4个数据；如果是小于等于2的话就是0x9200 1 跟一个数据 。

2、在UdcConfig目录下面新建一个自己 的项目名称比如路径 UdcConfig\lcd\FS088\TRX\L458

3、从隔壁项目考来一个mipi文件，最好是芯片名字是一样的，比如 lcd\_st7701s\_mipi

4、打开这个文件，里面的数据 这两句是数据头，这一句是数据尾中间的是数据，但是其中还有两句很重要的启动语句：

，也就是说应该有0x11和

0x29，当然前面的设置指令也是必不可少的,如果客户没有给，最好去参照别的项目拷贝一个过来。

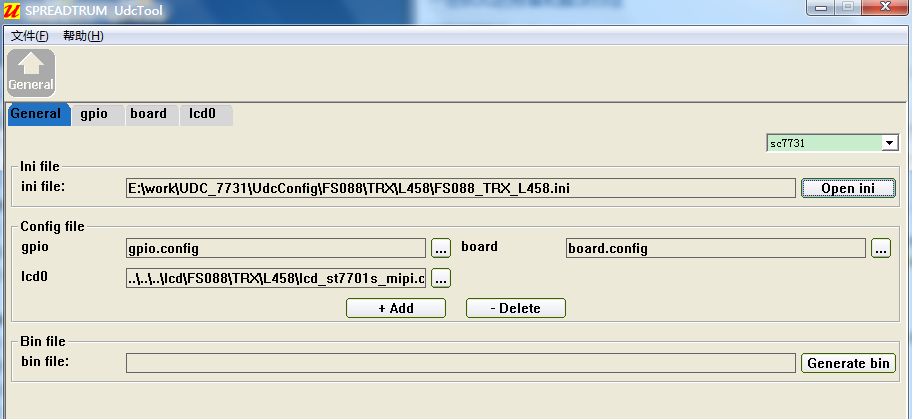
6、把厂家给的数据转换好了，然后拷贝到刚刚的文件的数据区，加上 11 29的命令，然后保存

7、这时候我们就使用UDC的生成工具，打开生成工具。

8、在UdcConfig目录下面新建一个自己 的项目名称比如路径 UdcConfig\FS088\TRX\L458

9、去兄弟目录哪里考来三个文件board、gpio、和一个配置文件。

10、通过UDC的生成目录打开配置文件，如下图（其中lc0一开始是没有）



11、点击add，在path中选择刚刚我们保存的文件。

12、点击Generate bin就可以在目录 UdcConfig\FS088\TRX\L458 看到生成一个.bin的文件。

13、把这个文件拷贝到udc\_bin的目录下面。

14、提交的时候记得提交3个地方的改动。

注意：如果修改分辨率的时候，不仅要修改config文件中的C:\Users\ADMINI~1\AppData\Local\Temp\Image.png，下面的初始化数据也应该按照厂家的来，还有config.mk也必须改，这三个缺一不可。UDC 屏

## 屏幕出现漏电现象

### 平台：SC7731C、SC9832

1、一般出现的原因是，进入休眠状态的时候屏幕没有真正的进入休眠，所以现在应该对屏的reset脚进行拉高或者拉低，让他进入休眠状态。已经在zy.h中做出一个宏ZCFG\_LCD\_SUSPEND\_WITH\_ENTER\_SLEEP

2、屏代码的路径是在idh.code\kernel\drivers\video\sprdfb\sprdfb\_panel.c 中

## timing对应值

### 平台：SC7731C、SC9832

static struct timing\_rgb lcd\_ili9881c\_mipi\_timing = {

.hfp = /\* unit: pixel \*/

.hbp =,

.hsync =,

.vfp =, /\*unit: line\*/

.vbp =,

.vsync =,

};

## 4、屏幕效果不好，一边正常一边会斗

### 平台：SC7731C、SC9832、9850

引起这种现象一般是要改 和这两个地方进行调大调小至于 这个是怎么算的目前还没有看懂有待研究。

## 5、多个屏幕兼容之后就不能用

### 平台：SC9850

出现这种情况，一般是由于ID匹配的问题； 查看log文件sprdfb\_panel.c 函数 adapt\_panel\_from\_readid 关键字 sprdfb\_udc\_lcd

1. 系统如果都匹配不到id的时候会默认下载最后一个id的初始化，所以有时候屏ID错了也是可以点亮的。
2. 由于1中的原因，所以之前有些屏能点亮并不一定是有雨匹配到了id，而是刚好处于最后一个屏。如果处这种情况的话，继续兼容屏的时候就很有可能造成之前的屏点不亮的情况。

3、udc匹配是在uboot15 中完成的，匹配完之后把得到的ID值和屏宽、屏高等一些信息传递到内核中（怎么传还待深入。）

## 6、esd check 使用te信号检测

### 平台： SC9850

背景：如果使用发送指令查询esd（静电干扰）功能的时候，容易出现干扰（闪屏）。这个时候就可以尝试使用te esd check这一个功能。

原理：目前读展讯给的代码的理解是：屏有一个te脚连接我们板子上面的LCM\_FMARK脚上面，这个脚会定时的像我们主板发送信号（除了受到静电干扰的时候。），如果系统在相应的时间里面（可以再电池配置表里面配置，但是目前这一个功能还没有做进去，只是在udc.c里面把这个值写成固定的1s。）没有触发中断，那么系统就判定这个屏存在静电干扰，然后就重新复位下载初始化。

开启方法：在UDC配置中 （1）、power\_mode = 0x03 设个模式设为0x03,下面的read\_powermode配置成什么已经无所谓了。（2）、在初始化配置中，在发送0X11前应该加 0x9200,2,0x35,0x00, 向0x35寄存器发送00让屏开启TE功能。

## 7、展频开关方法



## 8、如何录制屏幕

（1）、adb shell screenrecord /sdcard/demo.mp4

（2）、操作你要录制的视频。

（3）、导出视频：adb pull /sdcard/demo.mp4

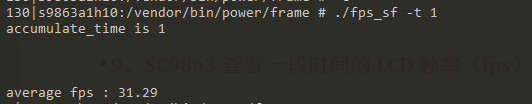
## 9、SC9863查看一段时间的LCD帧率（fps）

### 一、查看一段时间内的平均屏帧率（fps）

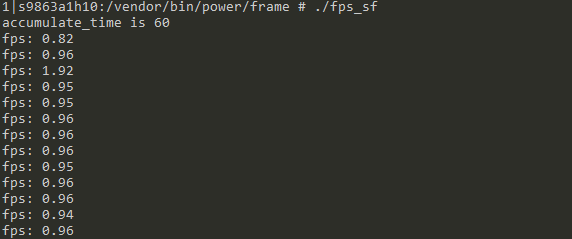
adb shell

./vendor/bin/power/frame/fps\_sf -t 1

-t 时长(默认1min)以分钟为单位



### 二、实时查看屏的帧率（fps）



### 三、SC9863 统计一段时间内ddr频点变换表

（1）、用法

连接adb切换root进入shell

adb root

adb remount

adb shell

./vendor/bin/power/ddr/trans\_table/ddr\_trans\_table -t 60

-t 测试时间(默认为60s)

（2）、示例

s9863a1h10:/vendor/bin/power/ddr/trans\_table # ./ddr\_trans\_table -t 60

accumulate\_time is 60

counts is 6000

160 233 311 400 533 622 800 933

160 0 0 0 0 0 0 0 0

233 0 23 4 0 0 0 0 0

311 0 4 98 40 0 7 1 0

400 0 0 27 324 0 93 2 0

533 0 0 0 0 0 0 0 0

622 0 0 17 67 0 1307 129 0

800 0 0 4 12 0 106 2401 65

933 0 0 1 3 0 6 55 1203

### 四、如何固定ddr的频率

**方法一：**

（1）、用法:

连接adb切换root进入shell

adb root

adb remount

adb shell

./vendor/bin/power/ddr/fix\_freq/fix\_ddr\_freq

（2）、示例

s9863a1h10:/vendor/bin/power/ddr/fix\_freq # ./fix\_ddr\_freq

./fix\_ddr\_freq

ddr avalible freq:

160 233 311 400 533 622 800 933

input fix freq

311

311

cur\_freq:311

**方法二：**

固定频点：

echo 933 > sys/class/devfreq/scene-frequency/sprd\_governor/scaling\_force\_ddr\_freq

实时查看频点：

cat sys/class/devfreq/scene-frequency/sprd\_governor/ddrinfo\_cur\_freq

# Config.mk问题

## SC9832的板子从16+1 改为 16+2

### 平台：SC7731C、SC9832

#---内存配置 ( 512M:9832 7.0不允许;2G:2GByte;默认不定义就是1GByte )--------------------

ZCFG\_PRODUCT\_RAM=2G

## 上下音量键反了

### 平台：SC7731C、SC9832

解决方法：修改mk中的一个物理键的位子：

#---物理按键配置区，顺序Volumedown Volumeup

ZCFG\_MK\_KEYMAP\_GPIO=124 125

## 3、9863 9.0摄像头配置

1：ZCFG\_SINGLE\_CAMERA\_BLUR //前柔光自拍，后副摄支持遮挡提示 ，展讯平台不支持了，我们自己维护

2：ZCFG\_FRONT\_SLIGHT\_REAR\_APERTURE //前:单摄柔光自拍, 后:大光圈

3：ZCFG\_FRONT\_SLIGHT\_REAR\_SCAM //前:单摄柔光自拍, 后:普通单摄

4：ZCFG\_FRONT\_SCAM\_REAR\_APERTURE //前:双摄柔光自拍, 后:大光圈

5：ZCFG\_FRONT\_SCAM\_REAR\_SCAM //前:普通单摄, 后:大光圈

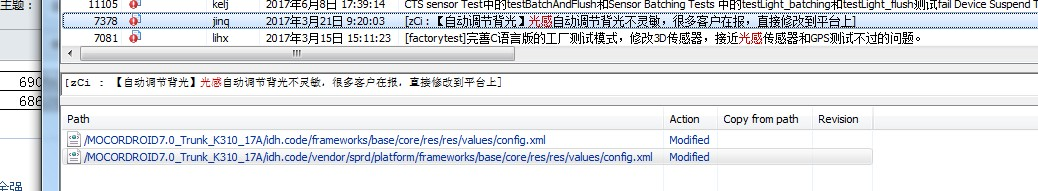
SC9863 9.0 CAMERA配置开关 zprj/camera配置列表.txt

# 传感器问题

## 光感不灵敏要修改的地方

### 平台：SC7731C、SC9832

上层修改



## 2、开机震动时间调短

路径：u-boot15\common\loader\loader\_nvm.c 里面的关键词：set\_vibrator

解决方法：收短振荡器 开set\_vibrator（1） 和 关 set\_vibrator（0）的时间间隔。

## 3、sensorhub如何抓取log

### （1）、使用工程模式抓取

如果想要抓取sensorhub的log？

需要插入SD卡，然后用展讯的抓log工具导出就可以看到一个sensorhub目录的log了。

如何抓取更多的sensorhub的log呢？

adb shell

echo 4 2 > /sys/class/sprd\_sensorhub/sensor\_hub/logctl

### （2）、使用adb抓取实时数据

adb root

adb remount

adb shell

echo log on > /dev/sctl\_pm

cat /dev/slog\_pm > /data/sensorhub.log

adb pull /data/sensorhub.log 本地路径

### （3）、如何开启更加详细的log

adb shell

echo 4 2 > /sys/class/sprd\_sensorhub/sensor\_hub/logctl

## 4、如何快速编译sensorhub

（1）、进入路径：idh.code/vendor/sprd/modules/sensors/libsensorhub

（2）、执行命令： mm

（3）、生成so文件路径:

\idh.code\out\target\product\s9863a1h10\obj\SHARED\_LIBRARIES\libsensorsdrvcfg\_intermediates\libsensorsdrvcfg.so

（4）、adb push操作

adb push libsensorsdrvcfg.so /vendor/lib64/ libsensorsdrvcfg.so

# Pinmap 配置相关问题

## pinmap文件的目录及配置规则

（1）、pinmap的目录：

u-boot64/board/spreadtrum/sp9830aed\_5m\_h100/pinmap-sp9630.c

（sp9830aed\_5m\_h100对应到实际项目）

（2）、配置规则：

例如：

{REG\_PIN\_IIS0DI, BIT\_PIN\_SLP\_AP|BIT\_PIN\_NULL|BITS\_PIN\_DS(1)|BITS\_PIN\_AF(2)|BIT\_PIN\_NUL|BIT\_PIN\_SLP\_WPD|BIT\_PIN\_SLP\_IE},

|  |  |
| --- | --- |
| REG\_PIN\_IIS0DI | 需要配置的pin脚 |
| BIT\_PIN\_SLP\_AP | AP或CP的sleep控制 |
| BIT\_PIN\_NULL | 强上下拉的设置 |
| BITS\_PIN\_DS(1) | 驱动能力设置 |
| BITS\_PIN\_AF(2) | 功能选择 |
| BIT\_PIN\_NUL | 弱上下拉设置 |
| BIT\_PIN\_SLP\_WPD | Sleep时的上下拉设置 |
| BIT\_PIN\_SLP\_IE | Sleep时的IO输入输出设置 |

## 1、SC9832E不使用SIM卡热插拔功能时的PINMAP配置

SC9832E不使用SIM卡热插能拔功能时，PINMAP中的SIM卡热插拔检测管脚需要对应修改为GPIO功能，否则会造成无法识别SIM卡的问题。不使用SIM卡热插能拔功能时正确的配置如下：

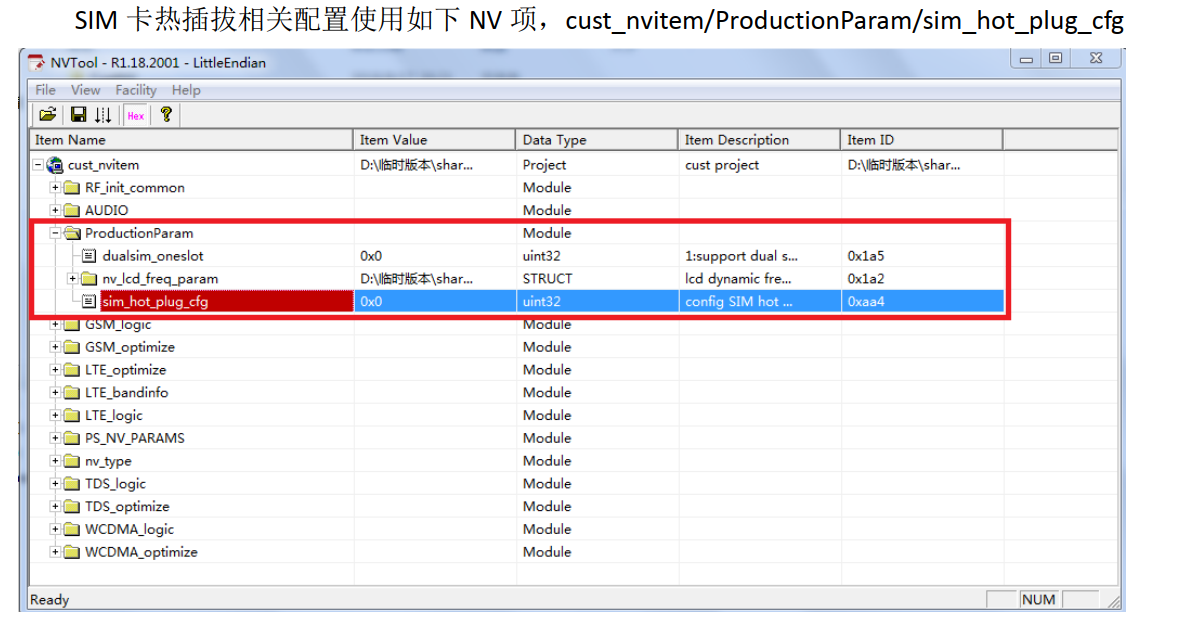
{REG\_PIN\_GPIO31, BITS\_PIN\_AF(3)},

{REG\_MISC\_PIN\_GPIO31, BITS\_PIN\_DS(1)|BIT\_PIN\_NULL|BIT\_PIN\_WPD|BIT\_PIN\_SLP\_NONE|BIT\_PIN\_SLP\_WPD|BIT\_PIN\_SLP\_Z},//SIM0\_DETSIM0

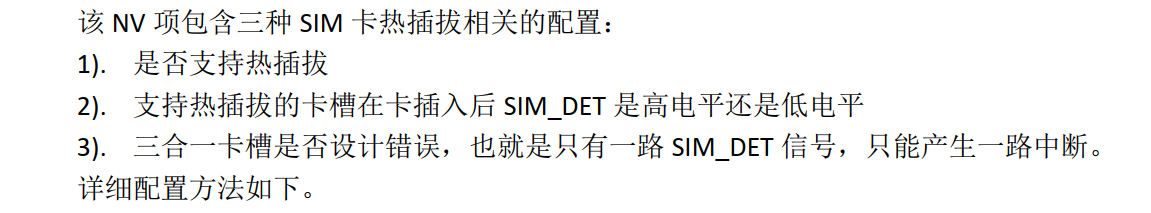
{REG\_PIN\_GPIO9, BITS\_PIN\_AF(3)},

{REG\_MISC\_PIN\_GPIO9, BITS\_PIN\_DS(1)|BIT\_PIN\_NULL|BIT\_PIN\_WPD|BIT\_PIN\_SLP\_NONE|BIT\_PIN\_SLP\_WPD|BIT\_PIN\_SLP\_Z},//SIM1\_DETSIM1

**注意：**没有热插拔的板子出现不识卡一般就是配置了热插拔功能，修改方法一：如上修改pinmap 。方法二修改



里面的这一项为0，其他值的定义：





## 2、TP增加手势唤醒功能后，底电流过高,从2.2MA增加到5.3MA

请在pinmap.c文件中，

将TP中断脚配置中的BIT\_PIN\_SLP\_Z

改为BIT\_PIN\_SLP\_IE

## 3、sc9832内部上拉电阻，软件如何设置

在pinmap中配置：

{REG\_PIN\_xxx, BIT\_PIN\_SLP\_AP|BIT\_PIN\_XXA|BITS\_PIN\_DS(1)|BITS\_PIN\_AF(0)|BIT\_PIN\_XXB|BIT\_PIN\_SLP\_WPU|BIT\_PIN\_SLP\_Z},

20K: BIT\_PIN\_XXA 配置为BIT\_PIN\_NULL BIT\_PIN\_XXB配置为BIT\_PIN\_WPU

4.7K: BIT\_PIN\_XXA 配置为BIT\_PIN\_WPUS BIT\_PIN\_XXB配置为BIT\_PIN\_NULL

1.8K: BIT\_PIN\_XXA 配置为BIT\_PIN\_WPUS BIT\_PIN\_XXB配置为BIT\_PIN\_WPU

相关pin脚支持的上拉电阻值请查看spec

## 4、如何在pinmap中把GPIO一开机就拉高

以GPIO236为例，在spec中可以对应查到其名字以及地址，

首先，对其配置成gpio功能，具体如下：

BIT\_PIN\_SLP\_AP|BIT\_PIN\_NULL|BITS\_PIN\_DS(1)|BITS\_PIN\_AF(3)|BIT\_PIN\_WPD|BIT\_PIN\_SLP\_WPD|BIT\_PIN\_SLP\_Z},然后，可在pinmap中替换如下代码从而拉高GPIO236：

int pin\_init(void)

{

int i;

++ unsigned int gpiodata;

for (i = 0; i < sizeof(pinmap)/sizeof(pinmap[0]); i++) {

\_\_raw\_writel(pinmap[i].val, CTL\_PIN\_BASE + pinmap[i].reg);

}

++ gpiodata = \_\_raw\_readl(0x40280704);

++ \_\_raw\_writel(gpiodata|0x1000, 0x40280704);//打开gpio的mask，运行读写

++ gpiodata = \_\_raw\_readl(0x40280708);

++ \_\_raw\_writel(gpiodata|0x1000, 0x40280708);//设置gpio为输出状态

++ gpiodata = \_\_raw\_readl(0x40280700);

++ \_\_raw\_writel(gpiodata|0x1000, 0x40280700);//将gpio设置为高

return 0;

}

## 5、如何打印出系统休眠时不被屏蔽的所有中断源

一般情况下，如果AP被某个中断源唤醒而进行resume流程，在kernel.log中是可以通过搜索关键字wake up by来确定中断源的。

但是在遇到毛刺电平等特殊情况下，AP被唤醒后却又无法从中断控制器中读取到中断源，也就无法打印出wake up by这句log。

如果AP不断被毛刺中断唤醒引起功耗问题，同时又无法通过关键字wake up by来确定中断源，

就需要通过以下方法把AP休眠时不被屏蔽的中断源全部打印出来，然后逐个排查这些可疑的中断源。

该方法的原理是：

如果驱动程序认为在AP休眠时不要屏蔽某个中断源，

那么该驱动程序在通过request\_irq()函数注册中断处理函数时需要在参数中指定IRQF\_NO\_SUSPEND标志位。

在AP休眠时，suspend流程中会轮询每一个中断源，

如果中断源没有指定IRQF\_NO\_SUSPEND标志位，则调用irq\_disable()函数将该中断源的使能控制寄存器disable。

如果中断源指定了IRQF\_NO\_SUSPEND标志位，kernel原生的逻辑是什么事情都不做直接return，

而该方法就是在return之前把中断源所对应的中断控制器名和中断号打印出来。

kernel/kernel/irq/manage.c

void \_\_disable\_irq(struct irq\_desc \*desc, unsigned int irq, bool suspend)

{

if (suspend) {

if (!desc->action || (desc->action->flags & IRQF\_NO\_SUSPEND)) {

+ if (desc->action && (desc->action->flags & IRQF\_NO\_SUSPEND)) {

+ printk(KERN\_ERR "desc->name=%s, irq\_data.irq=%d, irq\_data.hwirq=%d irq\_chip->name=%s\n",

+ desc->name, desc->irq\_data.irq, desc->irq\_data.hwirq, (desc->irq\_data.chip)->name);

+ }

return;

}

desc->istate |= IRQS\_SUSPENDED;

}

if (!desc->depth++)

irq\_disable(desc);

}

例如log打印出下面的语句，则证明GPIO200在系统休眠时没有被屏蔽，

可以重点排查该GPIO引脚的电平信号以及pinmap配置。

c0 desc->name=(null), irq\_data.irq=361, irq\_data.hwirq=200 irq\_chip->name=irq-d-gpio

## 6、pin脚状态软件如何动态修改

软件上使用如下函数：

drivers/platform/sprd/pin\_switch.c

int pinmap\_set(u32 offset, u32 value)

offset表示pin脚寄存器地址偏移量，value表示地址写入的值。

## 7、如何保证GPIO口的输出状态维持到深睡眠

在pinmap.c里面设置GPIO口在sleep时的状态为SLP\_OE即可解决：

{REG\_PIN\_TRACEDAT3, PIN\_NULL, FUNC3, PIN\_NULL, SLP\_WPD, DS\_L1, PIN\_OE,

PIN\_NULL, SLP\_AP},

## 8、SC7731 GPIO应该如何配置和使用

1.首先要将要使用的PIN脚配置成GPIO功能

在u-boot/board/spreadtrum/sp7731gea/pinmap.c找到要用的PIN将Function配置FUNC3,FUNC3表示GPIO功能（以CCIRRST为例）。

{REG\_PIN\_CCIRRST, BIT\_PIN\_SLP\_AP|BIT\_PIN\_NULL|BITS\_PIN\_DS(1)|BITS\_PIN\_AF(3)|BIT\_PIN\_NUL|BIT\_PIN\_SLP\_NUL|BIT\_PIN\_SLP\_OE},

2.定义GPIO号

在/kernel/arch/arm/mach-sc/include/mach/\_\_board-sp7731gea.h

#define GPIO\_SENSOR\_RESET 186 （定义GPIO号）

3.配置GPIO的方向

在 kernel/drivers/media/sprd\_sensor/sensor\_drv\_k.c

int sensor\_k\_probe(struct platform\_device \*pdev)

{

......

//申请GPIO资源

ret = gpio\_request(s\_p\_sensor\_mod->pin\_main\_reset, "main camera rst");

if (ret) {

tmp = s\_p\_sensor\_mod->pin\_main\_reset;

goto gpio\_err\_exit;

}

......

}

LOCAL int \_sensor\_k\_reset(uint32\_t level, uint32\_t width)

{

......

//配置GPIO方向：输出

gpio\_direction\_output(s\_p\_sensor\_mod->pin\_sub\_reset, level);

// 设置GPIO输出值

gpio\_set\_value(s\_p\_sensor\_mod->pin\_sub\_reset, level);

......

}

# 编译问题

## 使用zy.h的宏编译不进去

### 平台：SC7731C、SC9832

原因：使用zy.h的宏最终会拷贝到Z:\SC9832\_Android7.0\MOCORDROID7.0\_Trunk\_K310\_17A\idh.code\kernel\include\soc\sprd下面对应的文件中比如 \_\_board-sp9832a\_2h11\_volte.h ，然后这个头文件又包含在board.h中。所以使用的文件应该引用相应的头文件 #include <soc/sprd/board.

## 2、9863A-9.0对于服务器的编译要求

（1）、Ubuntu系统需要升级到14.04以上。

（2）、装以下3个工具

sudo apt-get install libssl-dev

sudo apt-get install libswitch-perl

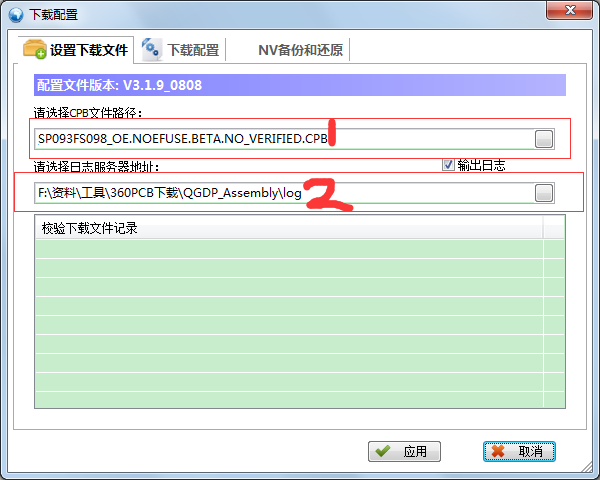
sudo apt-get install libxml2-utils

# 下载问题

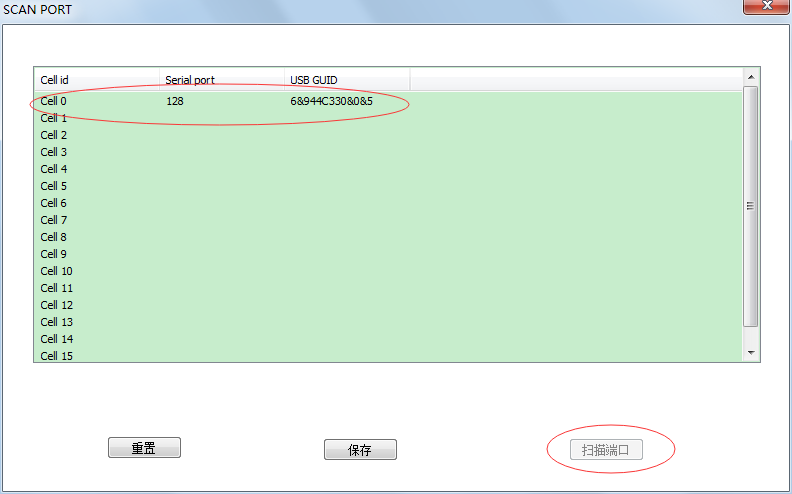
## 2、3600S的CPB下载

### 平台：SC7731C、SC9832 、SC9850

（1）、先设置好程序路径和log的路径如下图，其中1是选择程序路径，2是log路径（Tool里面有一个log文件夹）



（2）、手机插入USB但是不要按开机键或者音量键，点击扫描，出现如下图表示扫描成功，然后点击保存并拔出USB。



（3）、插入USB按住音量键上键点击开始 ，如果按住上键没有效果就三个按键同时按。

（4）、下载完成重启即可。

# 电池问题

## 电池曲线好坏的判定标准

### 平台：SC7731C、SC9832

1、充电或者放电的时候不能出现跳变的现象。

2、从百分九十九到百分之百时间不能太长，允许30分钟以内。

3、必须当电量达到百分之一（3.4V到3.5V之间）才能关机。

## 2、充电最高只能达到99达不到100

### 平台：SC7731C、SC9832

分析可能原因：

1、电流没有办法比设定的门限小（这个目前理解是芯片控制的 还没有看到配置的选项）。

2、电压没有办法比设定的门限大，如果电压一直达不到，可以看下以下的电池曲线的配置选项：

chg-end-vol-pure = <4380>; //末端电压 末端电压配置的值要比实际大20mv也就是现在配置的4360mv 注意这个末端电压如果开宏CHG\_CUR\_JEITA 会被下面数组jeita-cccv-tab里面的值替换掉。

结论：这次BUG经过调试log发现，电流是已经比电流门限小，但是电压最大只能是4330mv，而满电电压的条件是要大于4335，因此永远都达不到这个值，也就出现永远99的状态。查询电池曲线配置发现 chg-end-vol-pure = <4380> 这个末端电压显然是正常的，但是由于开了宏CHG\_CUR\_JEITA 而且jeita-cccv-tab = <4200 4200 4350 4350 4350 4350> 也就是常温的时候最大只能4330mv这样就对应上了，所以只要把这里修改为jeita-cccv-tab = <4200 4200 4370 4370 4370 4370> 这样最大就可以达到4350了，为什么不直接改到最大为4335呢，由于考虑到有写客户要求满电的时候必须在电池断可以测试到较为准确的满电电压，这里可以适当的配置高一点。

JEITA小知识：

目前JEITA功能只支持外部充电IC，开启该功能步骤如下：

1./kernel/drivers/power/sprd\_charge\_helper.h中定义宏CHG\_CUR\_JEITA

2.配置/kernel/arch/arm/boot/dts/sprd\_battery.dtsi中如下参数

a） jeita-tab-size = <6>;

目前只能配置6

b）jeita-temp-tab = <900 1010 1160 1460 1510 3000>;//实际温度=（配置值-1000)/10

配置不同的温度区间，此配置包含6个区间-10度以下，-10到1度，1度到16度，16度到46度，46度到51度，51度到300度

升温的过程中，不同的温度区间将设定不同的充电电流以及CCCV点，与配置jeita-cur-tab以及 jeita-cccv-tab一一对应，也就是说16度到46度，将设置充电电流为500mA，CCCV点将设置为4200V。

c）jeita-temp-recovery-tab = <870 1000 1140 1440 1480 3000>;

降温的过程中，当温度从区间-10到1度跌落回-10时，并不会立马将充电电流以及CCCV点调整到-10对应的温度区间的设定值，此时需要与jeita-temp-recovery-tab中对应的值比较，区间-10到1度跌落回区间-10以下，设置并不会发生在电池温度-10度的时候，温度必须跌落到-13度以下，设置才会触发，充电电流设置为0，CCCV设置到4200V，该项设置主要防止在一个温度左右起伏，来回设置充电电流以及CCCV点。

d）jeita-cur-tab = <0 0 500 1000 500 0>;

对应jeita-temp-tab中不同温度区间需要设置的充电电流。

e）jeita-cccv-tab = <4200 4200 4200 4200 4200 4200>;

对应jeita-temp-tab中不同温度区间需要设置的CCCV点。

## 3、高压电池低压电池

### 平台：SC7731C、SC9832

末端电压如果是4.35V的表示高压电池，4.2V普通电池。

## 4、快充阅读

### 平台：SC7731C、SC9832

pdata->jeita\_tab[i].x jeita-temp-tab

pdata->jeita\_tab[i].w jeita-temp-recovery-tab

pdata->jeita\_tab[i].y jeita-cur-tab

pdata->jeita\_tab[i].z eita-cccv-tab

## 5、vbat电压与电池ocv区别

### 平台：SC7731C、SC9832 、SC9850

电池OCV电压，即为电池的开路电压，可以简单理解为电池闲置搁置时候的电压。Vbat点电压，即为电池的闭路电压，可以理解为电池的工作电压。他们之间的关系如下：

Vbat=Vocv+Icurrent\*Rint

其中：

Vocv：电池的OCV电压

Icurrent：电池当前的充放电电流（负值放电，正值充电）

Rint：电池的内阻

当电池正常工作的过程中，由于电池内阻的存在，放电的时候，会使得电池正极的电势比实际的OCV偏低，充电的过程正好相反，电池的正极电势会随着电流方向的改变而比实际的电池OCV高。

## 6、快充和线充、艾维分别调用到的文件

### 平台：SC7731C、SC9832 、SC9850

快充：

sprd\_charge\_helper.c    sprd\_charge\_helper.h  sprd\_ext\_ic\_power.c   sprd\_battery.h   sprd\_2713\_fgu.c   sprd\_2713\_fgu.h

线充、艾维：

sprd\_battery.h   sprd\_2713\_fgu.c   sprd\_2713\_fgu.h sprd\_2713\_charge.c  sprd\_2713\_charge.h sprd\_2713\_power.c

## 7、使用FAN快充芯片的时候NTC功能失效

### 平台： SC9832

背景：FAN的快充芯片处理充电部分和线充、快充艾维并不是同一个文件。

解决方案：对应FAN的快充芯片如果想要支持NTC功能，不仅要在电池曲线配置表中配置支持NTC，同时还必须在zyt.h中开宏  **ZCFG\_FAN54XXX\_CHARGE\_HIGHORLOW\_STOP** 。

## 8、库仑计量电池容量

### 平台： SC9832 SC9850 SC7731C

库仑是国际标准单位为电流乘于时间的安培秒.; 1mAh=0.001安培\*3600秒=3.6安培秒=3.6库仑

锂离子电池容量计量之"库仑计"法　;

本文共为4部分：

　　1.基本概念介绍,就是引出为什么要用库仑计来计量电池容

量.

　　2.库仑计简单说明,讲得比较浅适合一般的用户.

　　3.库仑计进阶说明,详细补充库仑计的工作原理.

　　4.有关库仑计在手机上使用的各种现象解释,特别是拥有爱

　第一部分.基本概念介绍

　mAh:电池容量的计量单位,实际就是电池中可以释放为外部使用的电子的总数.;折合物理上的标准的单位就是大家熟悉的库仑.;库仑的国际标准单位为电流乘于时间的安培秒.; 1mAh=0.001安培\*3600秒=3.6安培秒=3.6库仑; mAh不是标准单位,但是这个单位可以很方便的用于计量和计算.;比如一颗900mAh的电池可以提供300mA恒流的持续3小时的供电能力.

　　在"锂离子电池容量计量之电压法"一文中,我详细介绍了用电池电压近似获得电池容量的方面.但是这个方法是比较粗糙的.要想获得锂离子电池的电量使用的正确情况,只有用库仑计.就象大家家里面的水量计量用的水表的作用原理.要计算流经的电荷的多少才能获得锂离子电池的电量使用情况.；如何得到流经的电荷量呢,就只有通过测试单位时间内回路流经的电流大小.；如果测到的电流是恒定不变的,把测到的电流乘于时间就可以简单的得到了电荷量了.这钟情况一般只有实验室里面才能发生.通常用一个恒流来放电计算电池的容量,而且测到的容量也比较精确.。如果测到的电流是随时间变化的,那么就需要对这个变化的电流进行积分才能得到电荷量,手机上的电池就是这种情况,电流

随时都在变化。

　　电流要如何测到呢?

简单的讲就是通过测量一个特定的电阻两端的电压(单位伏特V或毫伏mV),把电压除于这个电阻的阻值(单位欧姆ohm或毫欧mohm)就得到即时的电流值(单位是安培A或毫安mA)了.

　　第二部分,库仑计的工作原理及简单应用

　　■库仑计工作原理■

　　目前手机电池的容量计量(象爱立信手机电池),是在电池的保护线路上串联了一个电量计量芯片,其中串联的是一个集成的电阻,电阻值一般是20~30毫欧之间.

　　其基本原理是,芯片上集成了一个取样电阻,当流过不同电流后产生不同的压差,芯片就对这个电压(实际转换为电流)和时间进行积分,得到用户使用时的正确电量(注意,电量的单位是mAh). ;很多芯片可以实现这个功能,TI的,MAXIM的几个半导体芯片公司均有类似产品出品.电池增加这个芯片后,成本提高,对手机也提出了"通讯"的要求.所以很多手机没有这个功能.

　　那么该芯片通过实时积分得到容量以后,把容量(单位是mAh)数据存储在芯片的EEPROM中,并根据手机的需求,通过通讯线传递给手机.那么手机就得到了这块电池的准确容量.;库仑计的芯片,即电量计量芯片是装在手机电池里面的,通常是与锂离子电池的保护线路设计在一起.

　　在库仑计芯片的存储器里面通常有如下的基本电池信息

　　▲电池的初始容量(mAh),即额定容量,一个电池完全充放后得到的容量

　　▲电池的当前容量(mAh),处于使用状态是的电池容量

　　▲当前流经的电流(mA),即手机的电流损耗

　　■库仑计简单应用■

　　但是一般手机不会显示容量(比如"还有550mAh"或"剩下345mAh"这么一个让用户不好理解的数值.

　　这里就有两种容量显示方式,通过计算把容量mAh转换成可以很好理解的另外一种单位.

　　1.容量百分比(单位是%)

　　每个电池都有一个初始容量或称为额定容量.把当前的电池容量除于额定容量就可以得到当前电池的容量百分比一块额定容量为600mAh的电池,如果当前电池容量只有456mAh,手机就显示456/600=76%.

　　2.待机时间和通话时间(单位是时间:小时h)

　　手机本身根据自身的待机或通话电流损耗,会相应的内置一个计算公式,见这个容量转换成用户可以理解的待机时间和通话

时间.

　　待机时间的计算公式就是:

　　电池的当前容量/当前流经的电流=\*\*\*小时

　　通话时间的计算公式就是

　　电池的当前容量/通话时平均电流=\*小时

第三部分 库仑计进阶说明

　　1.电路

　　有两种类型的设计电路来获得电池的容量一种是A/D模数转换,通常是12位精度,将电压值转换为12位精度的16进制数字后乘于取样的间隔时间.一种是集成模式,利用RC积分电路来获得电压对时间的积分值.两者最终得到的都是电池的容量.

　　2.容量的归一

　　电压法那样测一个电压的时间非常短.不需要容量的归一.由于库仑计的容量获得必须是对时间的积分才能获得.那么要获得该电池的额定容量,就必须对电池进行一次完全的充分才能得到正确的电池容量信息.在库仑计的使用之初,厂家会赋予库仑计一个电池的初始容量.但是每次使用不可能都是完全用光或完全充饱.所以经常的浅充浅放会积累容量的误差.所以必须定期对使用库仑计的锂离子电池进行完全充分的操作,使其重新获得正确的电池额定容量.

　　3.额定容量的调整

　　锂离子电池的实际容量会随着循环的进行,可用容量会逐渐的衰减.如果库仑计里面保存的初始容量一成不变的话,肯定会导致容量显示的错误.比如初始容量是600mAh的电池,在100次循环以后,其实际容

量已经变成500mAh了,如果在进行容量显示是仍然按照600mAh这个数来计算的话,势必造成电池永远充不饱这个问题.所以在库仑计的里面还有一个容量对使用次数调整的算法,会根据电池循环次数调整其实际容量.锂离子电池的容量还会随使用温度的变化得到不同的可用容量.比如在25度时,电池可以得到几乎100%的可用容量而在0度是,电池只能放出80%的可用容量.如果不对容量进行温度调整的话,势必造成低温环境使用时,明明显示剩余20%的容量,却提前关机这个现象.所以在库仑计里面还有一个容量对温度调整的算法,会根据电池的实际温度进行可用容量调整.

　　4.电池标识码

　　许多库仑计芯片都顺便集成了一个电池标识码记录区域.它的作用是给手机电池标一个序列号,这个序列号是保存在芯片里面的,利用与手机的通讯口传递给手机.手机验证这个序列号.如果是正确的就判断为正品电池.如果序列号验证错误,手机就提示是非正品电池甚至拒绝工作.不要小看库仑计芯片,小心的一个芯片,不但要进行高精度的电量计量,还要进行复杂的运算调整.而且 还要与主机(手机)进行通讯联系.

第四部分.有关库仑计在手机上使用的现象解释

　　上面详细的介绍了库仑计芯片本身的工作原理.但是作为一个面向用户的应用,得到容量(mAh)这个数据是远远不够的.那么如何把容量转换成为待机时间就是手机的软件的工作了.

　　手机通过通讯,获得如下信息

　　▲电池的初始容量(mAh),即额定容量,一个电池完全充放后得到的容量

　　▲电池的当前容量(mAh),处于使用状态是的电池容量

　　▲当前流经的电流(mA),即手机的电流损耗

　　注意,当前流经的电流是个非常没有规律的值.显而易见,许多因素影响了这个电流.当地的信号强度,会影响手机的待机电流手机附加功能,比如蓝牙功能,红外功能会增加或改变待机电流手机背光(操作手机时),会大大的增大待机电流.这些影响因素会导致以下这个待机时间计算公式出现偏差

　　▲电池的当前容量/当前流经的电流=\*\*\*小时

　　如果待机电流稳定不变或变化很小的话,这个待机时间就会起到很好的计算作用这就是为什么手机固定放置在一个地方,待机时间很准的原因当地信号的好坏也直接的影响手机的待机时间显示.本来在A地显示180个小时待机的同一个手机拿到B地马上就变成了240个小时待机.这是因为计算公式的分母变化了.

　　但是待机时间的变化和强烈反差的出现并不是说电池出现了问题,实际其电池的容量还是没有变化的.这个也并不能用来说明是电池不好的原因.

　　虽然锂离子电池没有记忆效应,不需要专门的放电处理,不需要充饱电池就可以使用,这并不影响锂离子电池的性能发挥.但是,对于正在使用配备库仑计电池的手机用户,有一点必须

强调.

　　★ ★ ★请您定期的对你所用的电池进行一次完全的充分,

以便库仑计进行容量调整和归一化的进行.确保库仑计一直工作

在最佳状态.★ ★ ★

## 9、4.2的电池apk却显示充到了4.35

### 平台：SC7731C、SC9832 SC9850

出现问题的原因：由于配置表中这个表格原本写的是4350；这一项是末端电压（也就是横流到恒压的转换点）。如果这个值设置的偏高会出现一种现象，在充电的时候电压一直往上升，比如这个4.2的电池有可能升到4.3多；过晚的进入恒压状态，由于功率是一定的，电压一直往上升，电流也就不会下降（一直处于恒流状态）。当电压升到4.3多的时候电池也基本接近饱和了。当电压在4.3多停止的时候，电流会往下降；但是由于电池已经饱和了，会导致电流瞬间从600mA多降到100mA多。这样充电充出来的电池会有一种虚高的假象。

## 10、库仑计的一些相关函数（sprdfgu\_read\_soc）解析

### 平台：SC7731C、SC9832 SC9850

int sprdfgu\_read\_soc(void)

{

int cur\_cc, cc\_delta, capcity\_delta, temp;

uint32\_t cur\_ocv;

...................

cur\_cc = sprdfgu\_clbcnt\_get();

cc\_delta = cur\_cc - sprdfgu\_data.init\_clbcnt;

temp = DIV\_ROUND\_CLOSEST(cc\_delta, (3600 \* FGU\_CUR\_SAMPLE\_HZ));

................

}

这几行代码，具体什么意思？

[ANSWER]

FGU FGU\_CLBCNT\_VAL由两个寄存器组成，一个保存高16位数据，一个保存低16位数据，两个的组合将得到最终电流的累加和，我们以Ia表示。

假设第一次采样电流值为I1，第二次采样值为I2，第三次值为I3。。。第n此采样值为In，那么从第一次到第n次，寄存器 FGU\_CLBCNT\_VAL对应的值如下：

Ia=I1+I2+I3+。。。+In

理解了这个上述代码中的cc\_delta也不难理解，Ia的值就是cc\_delta，下面首先解释下下面三个语句的意思：

cur\_cc = sprdfgu\_clbcnt\_get(); //获取寄存器FGU FGU\_CLBCNT\_VAL的当前值

sprdfgu\_data.init\_clbcnt;//该变量中保存了寄存器FGU FGU\_CLBCNT\_VAL的一个初始值，或者理解该值为一个校准值，每次电量的计算都是拿当值与该值计算差值获取。

cc\_delta = cur\_cc - sprdfgu\_data.init\_clbcnt;//该计算就能获取从校准点到当前时刻电流的一个累加。

temp = DIV\_ROUND\_CLOSEST(cc\_delta, (3600 \* FGU\_CUR\_SAMPLE\_HZ));//计算库仑量Q

前面两句理解不难，最后一句再说明如下：

库仑量是电流对时间的积分，我们以Q表示，假设第一次采样电流值为I1，时间为t1，第二次采样值为I2，时间为t2，第三次值为I3，时间为t3。。。第n此采样值为In，时间为tn，那么从第一次到第n次统计的库仑量计算如下：：

Q=I1\*t1+I2\*t2+I3\*t3+。。。In\*tn

目前电流积分的采样频率500ms，则上述公式可以转化如下：

Q=I1\*500ms+I2\*500ms+I3\*500ms+。。。In\*500ms=(I1+I2+...+In)\*500ms

而Ia=I1+I2+I3+。。。+In

所以Q=Ia\*500ms

Q与temp都代表同一事件，不过他们之间的单位不同，Q单位是mAms，而temp单位是mAH，所以他们之间的值存在如下的转化关系：

temp=Q/(1000\*3600)=(Ia\*500)/(1000\*3600)=Ia/(2\*3600)

代码中的宏FGU\_CUR\_SAMPLE\_HZ值为2。

## 11、电池曲线的一些相关知识

### 平台：SC7731C、SC9832 SC9850

（1）、电池曲线入口：sprd\_battery.c

（2）、一些变量备注：

chg-end-vol = <4350>; // jeita-cccv-tab 现在 jeita-cccv-tab 表格的最大值如果里面有大于chg-end-vol的值就赋值为chg-end-vol。

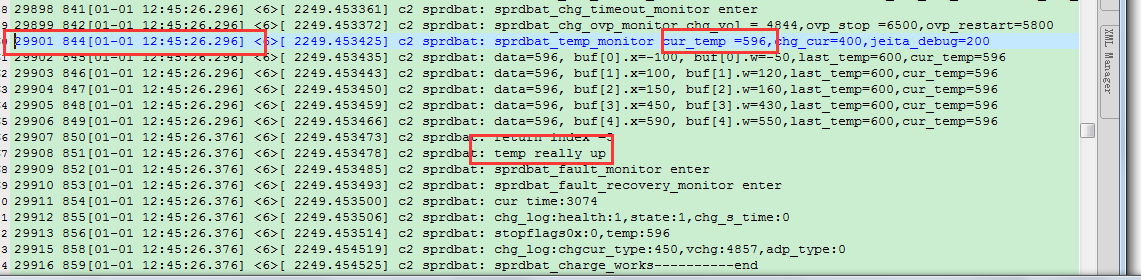
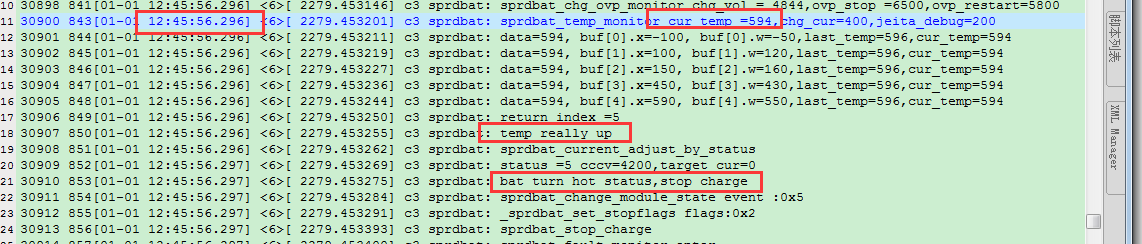
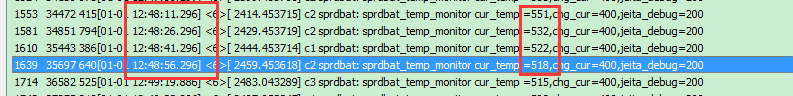
chg-end-vol-check = <4330>; //充电停止电压

chg-bat-safety-vol = <4430>; //安全电压

/\*0 vol and cur,1 status and vol,2 ext ic\*/

chg-full-condition = <0>; //充满电的判断条件

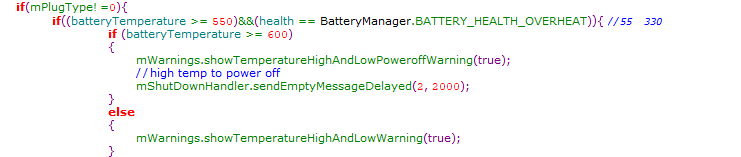
12、关于疑问

展讯的这一套NTC的控制算法有以下几点需要和你说明下：1、为了考虑到CPU的负荷，温度侦测是每15秒进行一次，所以温度更新本身并不是时时的。2、为了防止是由于侦测短时间内不准等造成的温度虚假上升，所以采用多次判断的做法，目前是当软件侦测到3次高温的时候（至少需要持续45秒），才判定为高温进而停止充电。3、为了减少计算量只有当温度持续上升或者持续下降的时候才会进行温度的判断。                从你提供的log中可以看出  当12:45：26的时候系统已经检测到了温度过高并进行第一次的高温预警，当12:45:56的时候 温度还是持续在59.4此时系统报了第三次高温预警从而也停止了充电，这边可以看出高温停充是正常的。 至于恢复充电从log中可以看出 55.1度到51.8度才用了45s不到，所以当第51.8度的时候系统才第三次检测到了回温，所以恢复充电的时候是51.8度。                       所以建议测试的时候如果想得到比较刚好的停充和恢复充电的温度，外界温度需要在59-60度保持至少1分钟左右才能看到停充效果，降温的时候需要控制缓慢的下降，最好能在54-55度之间保持一会。   而且使用的battery log的工具并不是时时的会导致有误差，比如59停充的时候battery log还没有更新，当温度缓慢下降到56度的时候，此时还没有到达回充状态还是停止充电，如果此时battery log更新显示的话就会看到56度停充的现象。

## 12、高温自动关机

文件路径：frameworks\base\packages\SystemUI\src\com\android\systemui\power\PowerUI.java

修改代码：



## 13、SC9863A 电池自学习机制解析

### 一、SC9863A 9.0电池自学习机制解析

**（1）、1个重要宏（在电池曲线的dts里面配置）**

sprdfgu\_data.is\_track 当等于1时代表开启自学习机制，当等于0时代表关闭自学习机制

**（2）3个重要函数**

static void sprdfgu\_qmax\_update\_monitor(void)

{

}

此函数的功能：判断是否进入自学习机制的条件

static void sprdfgu\_qmax\_works(struct work\_struct \*work)

{

}

此函数的功能：存储学习到的cnom.

static void sprdfgu\_track(void)

{

}

c此函数的功能：通过adc读取当前的电压、电流、温度等参数供自学习机制使用。

**（3）、自学习实现简单介绍**

**一、当满足以下条件时触发自学习机制（sprdfgu\_qmax\_update\_monitor函数中）：**

1）、当sprdbat\_data结构体为空时代表没有初始化因而不进入自学习机制。

if (!sprdbat\_is\_bat\_present())

return;

2）、当前温度小于15度时影响时不进入自学习机制。

if (track.c\_temp < 150)

return;

3）、当电压为0的时候不进入自学习机制。

if (track.relax\_vol == 0)

return;

4）、当处于一种错误状态的时候不进入自学习机制。

if (\*pstate == QMAX\_ERR)

return;

5）、当前电流大于300mA的时候不进入自学习机制

if (abs(track.relax\_cur) > sprdfgu\_data.qmax.s\_cur)

return;

6）、当前电压（vbat）大于3.6V的时候不进入自学习机制

if (track.relax\_vol > sprdfgu\_data.qmax.s\_vol)

return;

7）、当前电压在dts的电压表中换算成百分比如果大于50%就不进入自学习机制

ocv = track.relax\_vol -

(track.relax\_cur \* sprdfgu\_data.cur\_rint)

/ 1000;

sprdfgu\_data.qmax.s\_soc = sprdfgu\_vol2capacity(ocv);

if (sprdfgu\_data.qmax.s\_soc > 500) {

sprdfgu\_data.qmax.s\_soc = 0;

return;

}

8）、记录当前时间，并记录当前的电流积分作为初始积分。

cur\_time = ktime\_to\_timespec64(ktime\_get\_boottime());

sprdfgu\_data.qmax.s\_time = cur\_time.tv\_sec; //得到当前时间

sprdfgu\_data.qmax.s\_clbcnt = sprdfgu\_clbcnt\_get(); //记录当前的电流积分作为初始的电流积分

**二、当满足以下条件时得到有效的自学习到总容量cnom（sprdfgu\_qmax\_update\_monitor函数中）：**

1）、计算离触发自学习时间差是否大于10800s，如果大于就报超时自学习失败，同时重新判断自学习条件。

cur\_time = ktime\_to\_timespec64(ktime\_get\_boottime());

if (cur\_time.tv\_sec - sprdfgu\_data.qmax.s\_time >

QMAX\_TIMEOUT) {

SPRD\_FGU\_DEBUG("qmax time out...\n");

\*pstate = QMAX\_IDLE;

return;

}

2）、当前电压是否大于截止电压并且当前电流小于截止电流，如果满足进入下一个条件。

if ((track.relax\_vol > sprdfgu\_data.qmax.e\_vol) &&

(track.relax\_cur < sprdfgu\_data.qmax.e\_cur))

{

}

3）、计算学习到的容量和原先容量得差值的平方是否小于原先容量得50%，如果是满足条件。

if (abs(cnom - sprdfgu\_data.cur\_cnom)

< (50 \* sprdfgu\_data.cur\_cnom) / 100) {

SPRD\_FGU\_DEBUG("update cnom...\n");

sprdfgu\_data.cur\_cnom = cnom;

\*pstate = QMAX\_DONE;

queue\_delayed\_work(sprdfgu\_data.fgu\_wqueue,

&sprdfgu\_data.fgu\_qmax\_work, 0);

} else {

\*pstate = QMAX\_IDLE;

}

**三、存入学习到的容量（sprdfgu\_qmax\_works ）。**

if (\*p\_state == QMAX\_DONE) {

\*p\_state = QMAX\_IDLE;

SPRD\_FGU\_DEBUG("save battery capacity: %d\n",

pdrv->cur\_cnom);

file\_buf[0] = (unsigned int)(pdrv->cur\_cnom) ^ CAP\_KEY0;

file\_buf[1] = (unsigned int)(pdrv->cur\_cnom) ^ CAP\_KEY1; //这两部主要是用于后面的校验第四部有用到

vfs\_write(filep, (char \*)&file\_buf, sizeof(file\_buf), &pos);

}

**四、开机的时候读出自学习得到的容量（sprdfgu\_qmax\_works ）。**

static void sprdfgu\_qmax\_works(struct work\_struct \*work)

{

.

.

.

if (\*p\_state == QMAX\_INIT) { //开机刚初始化的时候

\*p\_state = QMAX\_IDLE;

if (vfs\_read(filep, (char \*)&file\_buf,sizeof(file\_buf), &pos) < 0) { //读取存入的数据

SPRD\_FGU\_DEBUG("battery file is empty or read error\n");

goto out;

}

SPRD\_FGU\_DEBUG("read buff: 0x%x,0x%x\n",

file\_buf[0], file\_buf[1]);

update\_cnom = (int)(file\_buf[0] ^ CAP\_KEY0); //解码

check\_cnom = (int)(file\_buf[1] ^ CAP\_KEY1); //解码

if (update\_cnom != check\_cnom) { //校验存入的数据是否被改变过（原理：用同样的数去异或两个不一样的值存入，取出来的时候在以后或出原先的值对比下两个值是不是一样，不一样的话就是改变了）

SPRD\_FGU\_DEBUG("buf data err %d,%d!\n",

update\_cnom, check\_cnom);

goto out;

}

if (abs(pdrv->cur\_cnom - update\_cnom)

< (50 \* pdrv->cur\_cnom) / 100) {

SPRD\_FGU\_DEBUG("update cnom old:%d, new:%d\n",

pdrv->cur\_cnom, update\_cnom);

pdrv->cur\_cnom = update\_cnom;

}

SPRD\_FGU\_DEBUG("read cap: %d\n", update\_cnom);

}

.

.

.

}

### 二、SC9863A 10.0 电池自学习机制解析

**dts宏开关：**cm-capacity-track;

**流程：**

1、probe init

static int charger\_manager\_probe(struct platform\_device \*pdev)

{

….

/\*获取相应的dts宏来判断是否使用自学习机制\*/

cm->track.cap\_tracking =

device\_property\_read\_bool(&pdev->dev, "cm-capacity-track");

if (cm->track.cap\_tracking) {

s\_cap\_tracking\_en = cm->track.cap\_tracking;

ret = cm\_get\_bat\_info(cm);

if (ret) {

dev\_err(&pdev->dev, "Failed to get battery information\n");

goto err\_reg\_sysfs;

}

cm\_track\_capacity\_init(cm);

}

…

}

static void cm\_track\_capacity\_init(struct charger\_manager \*cm)

{

INIT\_DELAYED\_WORK(&cm->track.track\_capacity\_work,

cm\_track\_capacity\_work);

cm->track.end\_vol =

cm->desc->fullbatt\_uV - CM\_TRACK\_CAPACITY\_VOLTAGE\_OFFSET;

cm->track.end\_cur =

cm->desc->fullbatt\_uA + CM\_TRACK\_CAPACITY\_CURRENT\_OFFSET;

**cm->track.state = CAP\_TRACK\_INIT; //此时状态是CAP\_TRACK\_INIT**

queue\_delayed\_work(system\_power\_efficient\_wq,

&cm->track.track\_capacity\_work,

5 \* HZ);//调用工作队列 cm\_track\_capacity\_work

}

static void cm\_track\_capacity\_work(struct work\_struct \*work)

{

struct charger\_manager \*cm = container\_of(work,

struct charger\_manager,

track.track\_capacity\_work.work);

u32 total\_cap, capacity, check\_capacity, file\_buf[2];

struct file \*filep;

loff\_t pos = 0;

static int retry\_cnt = 5;

int ret;

filep = filp\_open(CM\_TRACK\_FILE\_PATH,

O\_RDWR | O\_CREAT,

S\_IRUGO | S\_IWUSR);

if (IS\_ERR(filep)) {

dev\_warn(cm->dev, "failed to open track file.\n");

if (cm->track.state == CAP\_TRACK\_INIT && retry\_cnt > 0) {

dev\_err(cm->dev, "track file not ready.\n");

retry\_cnt--;

queue\_delayed\_work(system\_power\_efficient\_wq,

&cm->track.track\_capacity\_work,

5 \* HZ);

} else {

cm->track.state = CAP\_TRACK\_ERR; **//报错状态就改为CAP\_TRACK\_ERR**

}

return;

}

ret = get\_batt\_total\_cap(cm, &total\_cap);

if (ret) {

dev\_err(cm->dev, "failed to get total cap.\n");

goto out;

}

total\_cap = total\_cap / 1000;

switch (cm->track.state) {

case CAP\_TRACK\_INIT: //probe的时候会跑初始化

/\*

\* When the capacity tracking function starts to work,

\* need to read the last saved capacity value from the

\* file system, for security reasons we need to decrypt,

\* in contrast, when writing data to the file system,

\* we need to encrypt it.

\*/

cm->track.state = CAP\_TRACK\_IDLE;

if (kernel\_read(filep, (char \*)&file\_buf, sizeof(file\_buf), &pos) < 0) { //从文件中读取相应的数据

dev\_err(cm->dev, "track file is empty or read error\n");

goto out;

}

capacity = file\_buf[0] ^ CM\_TRACK\_CAPACITY\_KEY0;

check\_capacity = file\_buf[1] ^ CM\_TRACK\_CAPACITY\_KEY1;

if (capacity != check\_capacity) {

dev\_err(cm->dev, "track file data error.\n");

goto out;

}

if (abs(total\_cap - capacity) < total\_cap / 2)//文件中的容量如果满足条件就设置为容量

set\_batt\_total\_cap(cm, capacity);

break;

**case CAP\_TRACK\_DONE://这个由后面自学习完成之后才跑**

cm->track.state = CAP\_TRACK\_IDLE;

file\_buf[0] = total\_cap ^ CM\_TRACK\_CAPACITY\_KEY0;

file\_buf[1] = total\_cap ^ CM\_TRACK\_CAPACITY\_KEY1;

ret = kernel\_write(filep, &file\_buf, sizeof(file\_buf), &pos);

if (ret < 0) {

dev\_err(cm->dev, "write file\_buf data error\n");

goto out;

}

break;

default:

cm->track.state = CAP\_TRACK\_IDLE;

break;

}

out:

if (!IS\_ERR(filep))

filp\_close(filep, NULL);

}

**2、cm\_batt\_works 15s查询一次**

static void cm\_batt\_works(struct work\_struct \*work)

{

。。。

cm\_track\_capacity\_monitor(cm);//15s调用自学习状态更新函数

。。。。

}

static void cm\_track\_capacity\_monitor(struct charger\_manager \*cm)

{

int cur\_temp, cur\_now, vol\_now, ret;

int capacity, clbcnt, ocv;

u32 total\_cap;

if (!cm->track.cap\_tracking) //检查是否开启自学习

return;

if (!is\_batt\_present(cm)) { //检查电池是否在位

dev\_err(cm->dev, "battery is not present, cancel monitor.\n");

return;

}

ret = cm\_get\_battery\_temperature\_by\_psy(cm, &cur\_temp);//获取温度

if (ret) {

dev\_err(cm->dev, "failed to get battery temperature\n");

return;

}

if (cur\_temp > CM\_TRACK\_HIGH\_TEMP\_THRESHOLD ||

cur\_temp < CM\_TRACK\_LOW\_TEMP\_THRESHOLD) {//当温度大于45度或者小于15都的时候不触发自学习

dev\_err(cm->dev, "exceed temperature range, cancel monitor.\n");

return;

}

ret = get\_batt\_cur\_now(cm, &cur\_now); //获取当前电流

if (ret) {

dev\_err(cm->dev, "failed to get relax current.\n");

return;

}

ret = get\_batt\_vol\_now(cm, &vol\_now); //获取当前电压

if (ret) {

dev\_err(cm->dev, "failed to get relax voltage.\n");

return;

}

/\*

\* If the capacity tracking monitor in idle state, we will

\* record the start battery coulomb. when the capacity

\* tracking monitor meet end condition, also will record

\* the end battery coulomb, we can calculate the actual

\* battery capacity by delta coulomb.

\* if the following formula , we will replace the standard

\* capacity with the calculated actual capacity.

\* formula:

\* abs(current\_capacity -capacity) < capacity / 2

\*/

switch (cm->track.state) {

case CAP\_TRACK\_ERR:

dev\_err(cm->dev, "track status error, cancel monitor.\n");

return;

case CAP\_TRACK\_IDLE: //init之后就进入IDLE模式

/\*

\* The capacity tracking monitor start condition is

\* the current less than 30000ua and the voltage

\* less than 3750000uv. When meet the above two

\* conditions, the battery is almost empty, which is

\* the result of multiple test data, so this point

\* suitable as a starting condition.

\*/

if (abs(cur\_now) > CM\_TRACK\_CAPACITY\_START\_CURRENT ||

vol\_now > CM\_TRACK\_CAPACITY\_START\_VOLTAGE) { //电流大于30mA 电压大于3.75V的话不进入自学习

dev\_info(cm->dev,

"does not satisfy start condition, cancel monitor.\n");

return;

}

ret = get\_batt\_ocv(cm, &ocv);

if (ret) {

dev\_err(cm->dev, "get ocv error\n");

return;

}

/\*

\* Parse the capacity table to look up the correct capacity percent

\* according to current battery's corresponding OCV values.

\*/

cm->track.start\_cap = power\_supply\_ocv2cap\_simple(cm->desc->cap\_table,

cm->desc->cap\_table\_len,

ocv); //获取当前的OCV电压应该对应在哪个百分比

/\*

\* When the capacity tracking start condition is met,

\* the battery is almost empty,so we set a starting

\* threshold, if it is greater than it will not enable

\* the capacity tracking function, now we set the capacity

\* tracking monitor initial percentage threshold to 20%.

\*/

if (cm->track.start\_cap > 20) { //百分比大于20%不进入自学习

cm->track.start\_cap = 0;

dev\_info(cm->dev,

"does not satisfy the track start condition, start\_cap = %d\n",

cm->track.start\_cap);

return;

}

ret = get\_batt\_energy\_now(cm, &clbcnt);

if (ret) {

dev\_err(cm->dev, "failed to get energy now.\n");

return;

}

cm->track.start\_time =

ktime\_divns(ktime\_get\_boottime(), NSEC\_PER\_SEC);

cm->track.start\_clbcnt = clbcnt;

cm->track.state = CAP\_TRACK\_UPDATING; //满足以上条件开始进入自学习

break;

case CAP\_TRACK\_UPDATING: //自学习中

if ((ktime\_divns(ktime\_get\_boottime(), NSEC\_PER\_SEC)

- cm->track.start\_time) > CM\_TRACK\_TIMEOUT\_THRESHOLD) {

cm->track.state = CAP\_TRACK\_IDLE;

dev\_err(cm->dev, "track capacity time out.\n");

return;

}

/\*

\* When the capacity tracking end condition is met,

\* the battery voltage is almost full, so we use full

\* stop charging condition as the the capacity

\* tracking end condition.

\*/

if (vol\_now > cm->track.end\_vol &&

cur\_now < cm->track.end\_cur) { //当电流和电压满足充满条件的时候停止自学习

ret = get\_batt\_energy\_now(cm, &clbcnt);

if (ret) {

dev\_err(cm->dev, "failed to get energy now.\n");

return;

}

ret = get\_batt\_total\_cap(cm, &total\_cap);

if (ret) {

dev\_err(cm->dev, "failed to get relax voltage.\n");

return;

}

total\_cap = total\_cap / 1000;

/\*

\* Due to the capacity tracking function started, the

\* coulomb amount corresponding to the initial

\* percentage was not counted, so we need to

\* compensate initial coulomb with following

\* formula, we assume that coulomb and capacity

\* are directly proportional.

\*

\* For example:

\* if capacity tracking function started, the battery

\* percentage is 3%, we will count the capacity from

\* 3% to 100%, it will discard capacity from 0% to 3%

\* so we use "capacity \* (100 / (100 -3))" to

\* compensate.

\*

\* formula:

\* capacity = capacity \* (100 / (100 - start\_cap) )

\*/

capacity = (clbcnt - cm->track.start\_clbcnt) / 1000;

capacity = (capacity \* 100) / (100 - cm->track.start\_cap);

if (abs(capacity - total\_cap) < total\_cap / 2) {

set\_batt\_total\_cap(cm, capacity);

cm->track.state = CAP\_TRACK\_DONE; //改变状态写相应的自学习容量到文件中

queue\_delayed\_work(system\_power\_efficient\_wq,

&cm->track.track\_capacity\_work,

0);

dev\_info(cm->dev,

"track capacity is done capacity = %d\n",

capacity);

} else {

cm->track.state = CAP\_TRACK\_IDLE;

dev\_info(cm->dev,

"less than half standard capacity.\n");

}

}

break;

default:

break;

}

}

# 指纹问题

### 注意：9850上展讯平台自带集创指纹，\*#9980#中也自带了指纹测试，相对还是比较方便。

## 如何调试或者更新新的指纹

### 平台：SC7731C、SC9832 SC9850

**指纹各个文件介绍：**

\*.so文件：是一个库文件，是指纹的一些上层操作。

\*.tos文件：是指纹的一些底层操作。

\*.ta.elf 文件：是用来合入\*.tos的补丁。

\*.ta.syms.elf 文件：指纹ta对应的debug符号表文件（也就是log）。

操作步骤（比如新增或者优化一个集创的指纹）：

1. \*.tos文件需要通过 \*.ta.elf 文件 来生成，具体的生成步骤如下：

**注意：**以下命令的源文件和目标文件的命名和固定并不是固定的可以依情况而定。

a. 查看tos.bin文件里面的全部ta，确认是否已经包含集创的指纹TA

执行命令： ./mktosimg --tos tos.bin –l

显示如下：

All TAs:

UUID Start End Built

{fee67f9f-e1b1-4e3d-84-55-04-7f-60-01-af-ef} 0x2b000 0x3330b 22:37:49 Jul 22 2017

{eca48f94-00aa-560e-8f-8c-d9-4b-50-d4-84-f3} 0x3330c 0x39617 22:37:49 Jul 22 2017

{f7fc6e07-78d8-5efa-b4-a9-ec-f5-e0-77-fd-63} 0x39618 0x3f923 22:37:49 Jul 22 2017

{5f902ace-5e5c-4cd8-ae-54-87-b8-8c-22-dd-af} 0x3f924 0x91c2f 22:37:49 Jul 22 2017

{38ba0cdc-df0e-11e4-98-69-23-3f-b6-ae-47-95} 0x91c30 0x9ef3b 22:37:49 Jul 22 2017

{cea8706d-6cb4-49f3-b9-94-29-e0-e4-78-bd-29} 0x9ef3c 0xc2247 22:37:49 Jul 22 2017

{48c32cae-974d-4597-88-d7-bc-fc-cd-b2-c8-03} 0xc2248 0xc8553 22:37:36 Jul 22 2017

{4304bef6-36e5-4d91-94-b0-1e-a4-cd-51-d4-0c} 0xc8554 0x1338ff 04:26:00 Aug 30 2017

其中UUID {4304bef6-36e5-4d91-94-b0-1e-a4-cd-51-d4-0c} 为集创的指纹TA文件，如果tos.bin 里面没有集创的指纹ta, 就直接重命名成tos.org.bin ,直接执行步骤c，如果有相应的集创指纹要把它先删除，也就是执行步骤吧。

b. 删除tos.bin中已有的集创指纹ta，

执行命令: ./mktosimg --tos tos.bin -d -u 4304bef6-36e5-4d91-94-b0-1e-a4-cd-51-d4-0c -o tos.org.bin

c. 通过fp.ta.elf在tos.bin里面正价集创指纹：

执行命令： ./mktosimg --tos tos.org.bin --ta fp\_ta.elf -o tos.bin （这里可以得到已合入指纹ta的 tos.bin文件）

2、指纹CA库更新（fingerprint.default.so）

需要放到系统的某个目录下（目录应该已经有了，替换原文件即可），在编译时会被直接copy到out目录下的/system/lib/hw/fingerprint.default.so

3、\*.ta.syms.elf 文件只是一个记录而已，实际没有什么功能，因此找一个固定的地方提交即可，也可以不提交不影响功能。

**SC9850 调试指纹需要注意的地方：**

0、有指纹的项目应该在Config.mk里面配置 ZCFG\_MK\_FP\_SUPPORT=chipone

1、需要用到的文档和SDK我放到代码目录下：zcommon\docs\FAQ\Fingerprint\；

2、指纹厂提供TA/CA后，我们要通过工具生成对应的tos.bin文件，请放到 zprj\zfingerprint\_src\下。生成工具和原始tos文件也都放在这个目录；

3、调试新指纹我们驱动同事需要修改的地方 以及 为了让厂家顺利开发 CA/TA/配合我们工厂测试模式的so文件 三个东西，需要我们提供的东西。等等说明请见 zcommon\docs\FAQ\Fingerprint\Fingerprint.txt。 4、目前大版本的脚本还没有做好：a.\*.so文件是放在 idh.code\vendor\sprd\partner\chipone 路径下，当有客制化的指纹调试的\*.so，应该把相应的\*.so拉倒project下面。b、\*.elf 文件\*.tos文件是放在路径 zprj\zfingerprint\_src 路径下，当有客制化的\*.elf 文件\*.tos文件更新的时候，应该在这个目录下面新建相应project的文件夹注意命名，和9832不一样的地方是这个名字不用在Config.mk配置，只需要在从大版本中拉出文件 idh.code\vendor\sprd\build\tasks\sprdbuildidh.mk\修改

ifeq ($(strip $(BOARD\_FINGERPRINT\_CONFIG)), chipone)

tos\_bin := \

$(shell cp ../zprj/zfingerprint\_src/chipone\_trx\_l558/tos.bin ${PRODUCT\_OUT}/tos.bin)

endif

其中标红的地方就是刚刚的文件夹的名字。

## 2、指纹位于的路径

### 平台：SC9850

\idh.code\vendor\sprd\partner\chipone

## 3、9863指纹调试

### 平台：SC9863

驱动路径：idh.code\kernel\drivers\input\fingerprint\

Hal层路径：idh.code\vendor\sprd\partner\

节点路径：/dev/xxx （比如：/dev/fpsensor）

Apk运行服务路径：/vendor/bin/hw/xxx

（比如 ：[android.hardware.biometrics.fingerprint@2.1-service](mailto:android.hardware.biometrics.fingerprint@2.1-service)）

**调试步骤（以下以调试集创指纹为例）：**

（1）、在文件 idh.code\device\sprd\sharkl3\common\ AndroidKernel.mk 中增加

例如：

ifeq ($(strip $(BOARD\_FINGERPRINT\_CONFIG)),chipone)

$(call sprd\_create\_user\_config, $(KERNEL\_CONFIG), $(TARGET\_DEVICE\_CHIPONE\_CONFIG))

Endif

……

ifeq ($(strip $(BOARD\_FINGERPRINT\_CONFIG)),chipone)

TARGET\_DEVICE\_CHIPONE\_CONFIG := $(KERNEL\_DIFF\_CONFIG\_COMMON)/chipone\_diff\_config

endif

(2)在 idh.code\kernel\sprd-diffconfig\sharkl3\common\ 路径下增加文件 chipone\_diff\_config (和上一步的文件名一样)；内容配上**驱动用到的宏**：ADD:CONFIG\_CHIPONE\_FINGERPRINT

（3）、在文件idh.code\device\sprd\sharkl3\common\ security\_feature.mk中增加

ifeq ($(strip $(BOARD\_FINGERPRINT\_CONFIG)), chipone)

include vendor/sprd/partner/chipone/chipone.mk （其中chipone.mk对应的是hal层的mk）

endif

（4）、在hal层的相应指纹路径 idh.code\vendor\sprd\partner\chipone 中创建chipone.mk（名字同上）

内容：

# add for fpsensor fingerprint

PRODUCT\_COPY\_FILES += vendor/sprd/partner/chipone/fingerprint.default.so:vendor/lib64/hw/fingerprint.default.so

PRODUCT\_COPY\_FILES += vendor/sprd/partner/chipone/lib/libfp\_ext\_svc2.so:vendor/lib/libfp\_ext\_svc2.so

PRODUCT\_COPY\_FILES += vendor/sprd/partner/chipone/lib/libchiponeic\_fingerprint\_factory.so:vendor/lib/libfactorylib.so

PRODUCT\_COPY\_FILES += vendor/sprd/partner/chipone/lib/vendor.fpsensor.hardware.fpsensorhidlsvc@2.0.so:vendor/lib/vendor.fpsensor.hardware.fpsensorhidlsvc@2.0.so

PRODUCT\_COPY\_FILES += vendor/sprd/partner/chipone/uinput-fpsensor.kl:system/usr/keylayout/uinput-fpsensor.kl

PRODUCT\_COPY\_FILES += vendor/sprd/partner/chipone/uinput-fpsensor.idc:system/usr/idc/uinput-fpsensor.idc

#support dynamic TA

PRODUCT\_COPY\_FILES += vendor/sprd/partner/chipone/fp\_ta.elf:/vendor/firmware/fpsensor.elf

PRODUCT\_COPY\_FILES += vendor/sprd/partner/chipone/init.chipone.rc:vendor/etc/init/init.chipone.rc

BOARD\_SEPOLICY\_DIRS += vendor/sprd/partner/chipone/sepolicy\_androido （这句话很关键它是指定设置selinux的时候用到\*te 和 \*conetxts的文件）

**调试小技巧：**

如果看log是打开节点失败，如果怀疑是权限问题但是此时上层有跑起来了，如果上层跑完发现节点打不开就很容易remove掉底层驱动创建的节点；导致开机之后想要改节点的权限的时候节点已经不存在了。所以可以先更改一下 /vendor/bin/hw/[android.hardware.biometrics.fingerprint@2.1-service](mailto:android.hardware.biometrics.fingerprint@2.1-service) 名字，比如

/vendor/bin/hw/[android.hardware.biometrics.fingerprint@2.1-service](mailto:android.hardware.biometrics.fingerprint@2.1-service)1 让开机之后不会自动炮这个服务。等我们改完我们要改的权限之后通过运行 ./vendor/bin/hw/[android.hardware.biometrics.fingerprint@2.1-service](mailto:android.hardware.biometrics.fingerprint@2.1-service)1让指纹的上层再跑起来省的多次编译了。

# 暗码问题

上层路径：idh.code/vendor/sprd/platform/packages/apps/ValidationTools/src/com/sprd/validationtools/

## 关于按码\*#9988#

### 平台：SC7731C、SC9832

1、驱动的按码代码路径：idh.code\kernel\drivers\zyt\_info

上层的按码代码路径：idh.code/vendor/sprd/platform/packages/apps/ValidationTools/src/com/sprd/validationtools/HardwareInfo.java

2、可以使用adb命令读取按码：cat /proc/zyt\_info

1. 通过adb中可以cat data/misc/cameraserver读取底层上传的信息。

## 2、关于\*#37\*#（\*#\*#83789#\*#\*）

### 平台：SC9850

1、代码路径：

入口：idcode\vendor\sprd\platform\packages\apps\ValidationTools\src\com\sprd\validationtools\ Const.java

测试项：idcode\vendor\sprd\platform\packages\apps\ValidationTools\src\com\sprd\validationtools\ itemstest

## 3、关于\*#9981\*#（\*#\*#83781#\*#\*）

# 温升问题

## 1、主板温升如何处理

### 平台： SC9850

相关笔记：zcommon\tools\thermal

软件路径：zprj\FS098\TRX\DRV\_ONLY\_L558\_50A\idh.code\device\sprd\sharkl2\common

## 2、功耗

后续9.0平台凡是设计到联网待机功耗大的，请统一按照我这个要求来做，如果测试数据准确的情况下，可以准确定位到哪个app，不要再用去掉gms包的版本来做排除法

一：测试要求：

1：debug版本，关闭log

二：测试方法如下：

1).灭屏待机前请进入adb执行以下命令，拔掉usb，灭屏待机。

adb shell cat /proc/net/xt\_qtaguid/stats >old\_status.txt ////请确保这个文件内容不为空

2)测试一个小时以上

3).插上usb，亮屏后执行如下命令

adb shell cat /proc/net/xt\_qtaguid/stats >new\_status.txt ////请确保这个文件内容不为空

adb shell ps -A >ps.txt ////请确保这个文件内容不为空

adb shell cat data/system/packages.list >packages.txt ////请确保这个文件内容不为空

提供old\_status.txt, new\_status.txt,ps.txt,packages.txt 这个四个文件给我

# LDO配置

### 平台： SC9850

Uboot：idh.code\u-boot15\drivers\power\\_\_sc2721\_regulator\_map.h

Kernel：idh.code\kernel\arch\arm\boot\dts\ sc2721.dtsi

# DTV问题

### 平台： SC7731C

配置（详见SC7731-FS259）：

1、在Config.mk中开宏：

#----高清DTV-FCI8300

ZCFG\_MK\_FCI8300\_SUPPORT=yes

2、信道有不同（其中8MHZ意思就是调频的时候是8 16这样一直往上加）：6MHZ 带宽 ：日本 巴西 南美 菲律宾 都是 8MHZ 带宽：斯里兰卡 。如果是8M带宽应该在zyt.h中开宏：#define ZCFG\_DTV\_FC8300\_BAND\_WIDTH\_8\_ENABLE

3、目前有apk ： fs286\_myphone\_myldigitalTV4.17F

测试方法（由于我国目前并不支持DTV，我们只能做一些简单的测试，实际测试应该让客户自己验证）：

1、apk必须打得开。

2、自动化测试工具的dtv测试ok。

# DDR问题

## 1、相关的文件路径

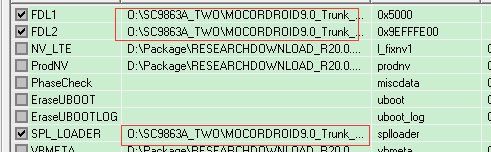
idh.code\chipram\include\configs

idh.code\chipram\ddr\ddr\_init\init\ddrc\r1p0

## 2、编译和下载

make chipram

下载



## 3、9863 DDR频率的相关知识

### （1）、查询DDR频率

**1）、查询频率**

方法一（实时查询频率）

cat sys/class/devfreq/scene-frequency/sprd\_governor/ddrinfo\_cur\_freq



方法二（统计一段时间内频率转换）

# 自动化测试（BBAT）

## 1、如何抓取想要的log（关键字 BBAT）

（1）、清空log数据并打开ylog开关。

（2）、按power键关机进行自动化测试。

（3）、等测试完毕拔掉USB（经过测试这步还是蛮有必要的不然有时候log没有办法生成）等待1-2分钟。

（4）、长按power键开机或者扣掉电池开机等待1-2分钟。

（5）、关闭ylog开关，到处log。

（6）、在\internal\_storage\ylog\目录下回多出一个cache的目录，这个目录就是自动化测试生成的log。

**10.0log路径：**Z:\20200115120519\data\ylog\ylog\ap\001-0115\_112456\_poweron 0-android.log 0-kernel.log

## 2．如何快速编译

# SD卡问题

## 1、SD卡不识别

**问题分析**：正常情况下是由于dts文件配置中断脚出错，比如sp9832e-1h10-gofu.dts:

&sdio0 {

cd-gpios= <&ap\_gpio 9 1>;

};

基本上就是改其中的9 和 1,9的意思是哪个gpio口 1的意思是高电平触发还是低电平触发。

# 其他

## 1、adb 指令操作

### （1）、用adb查看寄存器的值

lookat 0x20E0302C

## 2、提CQ注意内容

### （1）、9863 9.0提交涉及假双摄得CQ时，或者邮件，统一以单摄虚化来替代，展讯默认不支持假双摄。

## 3、把Android引导到串口log中

1.在init.xx.rc中靠前的位置增加如下启动项

-----------------------------------------------------------------

@bjand03:~/sprdroid9.0\_trunk/device/sprd/sharkl3/common$ git diff ./

diff --git a/common/rootdir/root/init.common.rc b/common/rootdir/root/init.common.rc

index 95fec1e..2730e33 100755

--- a/common/rootdir/root/init.common.rc

+++ b/common/rootdir/root/init.common.rc

@@ -141,6 +141,12 @@

on post-fs-data

.......

mkdir /data/vendor/thermald 0770 system system

chown system system /data/vendor/thermald

mkdir /data/misc/hprofs 0777

#start aprd

+service logcat /system/bin/logcat -f /dev/kmsg \*:w

+ class main

+ user root

+ group log

+ oneshot

+

2. 关闭selinux

idh.code/system/core/init/selinux.cpp

EnforcingStatus StatusFromCmdline() {

}

bool IsEnforcing() {

+ return false;

if (ALLOW\_PERMISSIVE\_SELINUX) {

return StatusFromCmdline() == SELINUX\_ENFORCING;

}

4、如何配置samba

https://blog.csdn.net/wssxy/article/details/

## 4、如何设置展讯9.0版本时间会显示秒

（1）、首先adb执行以下命令：

adb shell pm enable com.android.systemui/com.android.systemui.tuner.TunerActivity

adb shell am start -n com.android.systemui/com.android.systemui.tuner.TunerActivity

（2）、进入设置--系统—系统界面调节工具--状态栏--时间--显示时分秒

## 5、SC9863统计一段时间内CPU平均使用率前N的进程

一、用法

正常连接adb进入shell

./vendor/bin/power/tops/tops -t 10 -n 10

-t 后面接的是测试时长(默认是10s)以s为单位

-n 后面接的是显示的CPU平均使用率前N的进程(默认是前10的进程)

二、示例

s9863a1h10:/vendor/bin/power/tops # ./tops -t 10 -n 10

%[CPU] ARGS

44.1 com.android.lau+

43.4 top -n 1 -d 3

39.4 system\_server

16.2 surfaceflinger

8.1 android.hardwar+

7.9 com.android.sys+

4.6 logd

3.4 ylog

2.3 [kworker/u16:1]

2.2 [irq/75-adaptiv+

# 项目配置问题

## 1、6.0的如何升级到7.0

步骤：

（1）、确定一份已经从6.0升到7.0的项目作为参考项目。

（2）、拷贝7.0中的刚刚确认的项目，然后对比对于board\_cfg 目录里面的文件的修改只要以下文件：

\_\_board-sp7731cea.h //uboot的头文件

\_\_board-sp7731ceb.h

sp7731cea.h //kerne的头文件

sp7731ceb.h

sprd-scx20\_sp7731cea.dts //dts

sprd-scx20\_sp7731ceb.dts

其他的原封不动的烤过来即可一般有以下文件：

connectivity\_calibration.ini //这个应该是蓝牙wifi的一些东西

connectivity\_configure.ini

GetHardwareConfig.sh

pin\_switch.h

pinmap-sp7731.c ////管脚配置

项目说明

还有其他的一些调试说明书

## 2、关于16+2和8+1

### 平台：SC7731C SC9832 SC9850

SC7731C SC9832 中16+2 和 8+1 是有区别的，需要在config.mk中配置宏：

#---内存配置 ( 512M:9832 7.0不允许;2G:2GByte;默认不定义就是1GByte )--------------------

ZCFG\_PRODUCT\_RAM=

SC9850 中软件的配置上面并不需要区分16+2 和8+1

注意：经过验证8+1的板子烧写16+2的程序和16+2的板子烧写8+1的程序都是可以成功的；但是SC7731 和 SC9832 应该是开不了机。