密级状态：绝密（　　）　秘密（　　）　内部（ ）　公开（ √ ）

Sofia-3gr 项目 PCBA 测试工具说明

（技术部，系统产品二部）

文件状态： [ ] 正在修改 [√] 正式发布	当前版本：	V1.0
	作　　者：	王剑辉
	完成日期：	2015-05-18
	审　　核：	
	完成日期：	

福州瑞芯微电子有限公司

Fuzhou Rockchips Semiconductor Co . , Ltd

版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.0	Wjh	2015-5-18	初始版本	

目 录

1 概述.....2

2 PCBA 编译打包和运行环境介绍.....3

2.1 PCBA 测试工程编译介绍.....3

2.2 PCBA 测试工具打包介绍.....3

2.3 PCBA 测试工具使用介绍.....4

3 测试项.....7

3.1 测试项分类说明：.....7

3.2 测试项详细说明：.....7

3.2.1 重力感应(gsensor)测试.....7

3.2.2 无线网络（wifi 测试）.....7

3.2.3 sd 卡（sdcard）测试.....7

3.2.4 屏幕（LCD）测试.....8

3.2.5 按键（KEY）测试.....8

3.2.6 TP 测试.....8

3.2.7 SIM 卡测试.....8

4 配置文件.....9

5 字体.....13

6 测试样例扩展.....13

1 概述

PCBA 测试工具用于帮助在量产的过程中快速的甄别 PCBA 的好坏，提高生产效率。目前包括屏幕（LCD）、触摸屏测试（TP）、重力感应（gsensor）、无线（wifi）、SD 卡（sdcard）、按键（KEY）测试项目、sim 卡测试（支持双 sim 卡的测试）。

该工具支持通过配置文件 test_config.cfg 对测试项进行配置，具体的配置说明请参第 4 部分“配置文件”

2 PCBA 固件编译打包和运行环境介绍

2.1 编译

PCBA 测试程序位于 Android 源码/extenal/rk-pcba-test 目录下，编译会生成 pcba_core 可执行文件，pcba_core 和 rk-pcab-test/res 下的相关文件在编译的时候会被自动拷贝到 system 的 bin 目录下。

编译说明：

一.进入sofia-3gr项目代码的跟目录下面，执行source build/envsetup.sh；然后执行lunch，会显示以下几个配置项：1. mini_emulator_x86_64-userdebug

2. mini_emulator_x86-userdebug

3. sofia3gr-user

4. sofia3gr-userdebug

5. sofia3gr-eng

这里我们选择5

二．进入/external/rk-pcba-test/目录下面，执行mm -B编译生成pcba_core可执行文件。

2.2 打包

由于 PCBA 运行于 ptest 模式中，因此我们打包固件时必须把 pcba 工具里面使用到的 sh 脚本文件和 .cfg 文件等打包到固件里面，生成一个 system.fls 文件

2.3 使用说明介绍

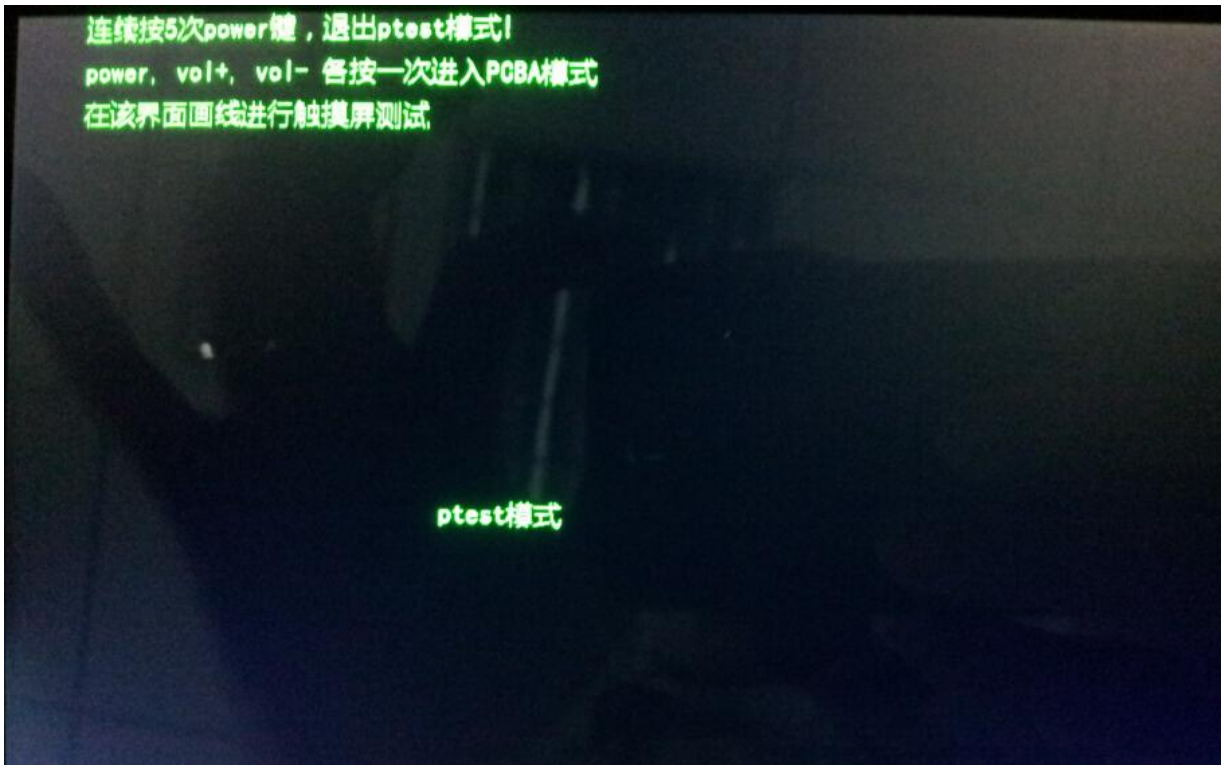
Sofia-3gr 量产流程介绍：

1. 升级烧写固件（成功后，设置下次启动进入 ptest 模式）
2. 写 pcb_sn 号、wifi 校准、2G/3G 校准、写默认的 IMEI 号（成功后，根据下一

步功能测试工具的需要来确定进 ptest 模式还是进 Android 模式。假如客户选择用 PCBA 测试工具测试，那么校准成功后应该继续保持进入 ptest 模式；假如客户选择用 devicetest.apk 来测试功能，那么校准成功后必须选择设置下次启动模式为 Android 模式）

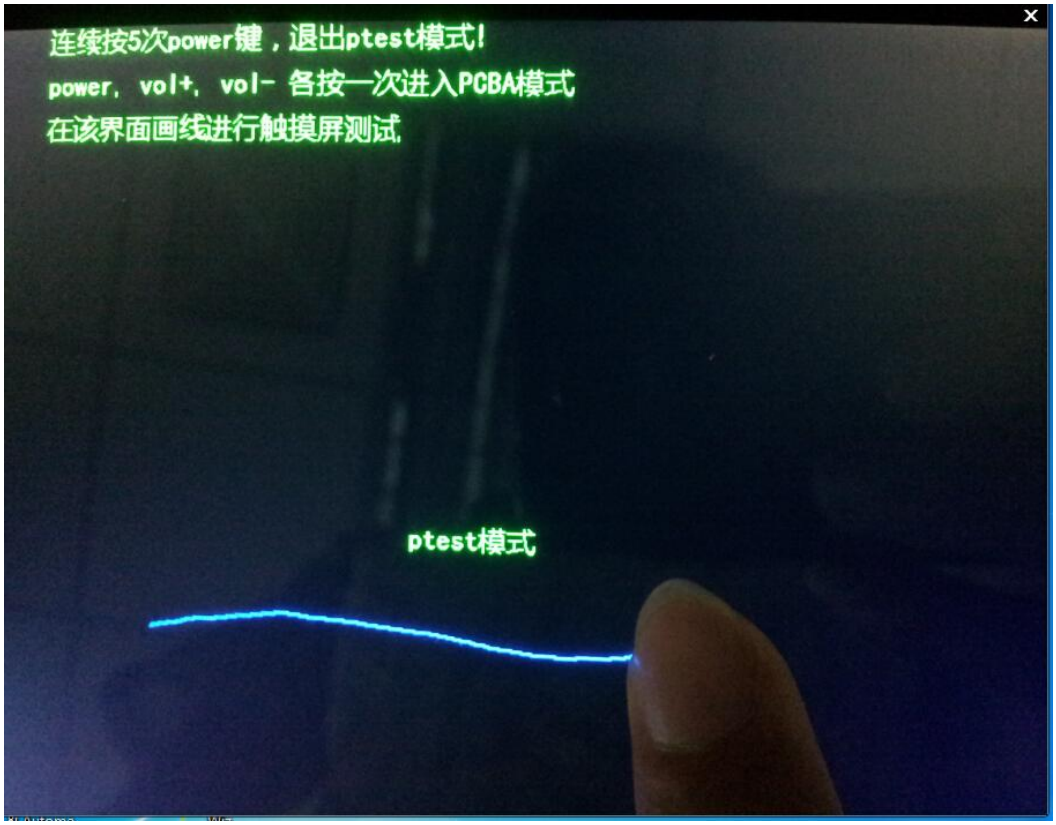
3. pcba 功能测试：可选择 pcba 测试工具或者用 devicetest.apk 测试工具。注意：pcba 测试工具是工作在 ptest 模式的，devicetest.apk 是工作在 android 模式的（测试成功后，设置下次启动模式为 Android 模式）
4. 组装整机，并使用 devicetest.apk 测试工具进行整机测试（测试通过后，设置下次启动模式为 ptest 模式）
5. 写 DEV_SN、写 MAC 地址、写 IMEI 号（写号成功后设置启动模式为 Android 模式）

PCBA 程序运行于 ptest 模式中，具体测试流程为：开机进入 Ptest 模式，启动 PCBA 工具，进入 PCBA 首界面（RF 校准就是在此界面下进行）。首界面如下：



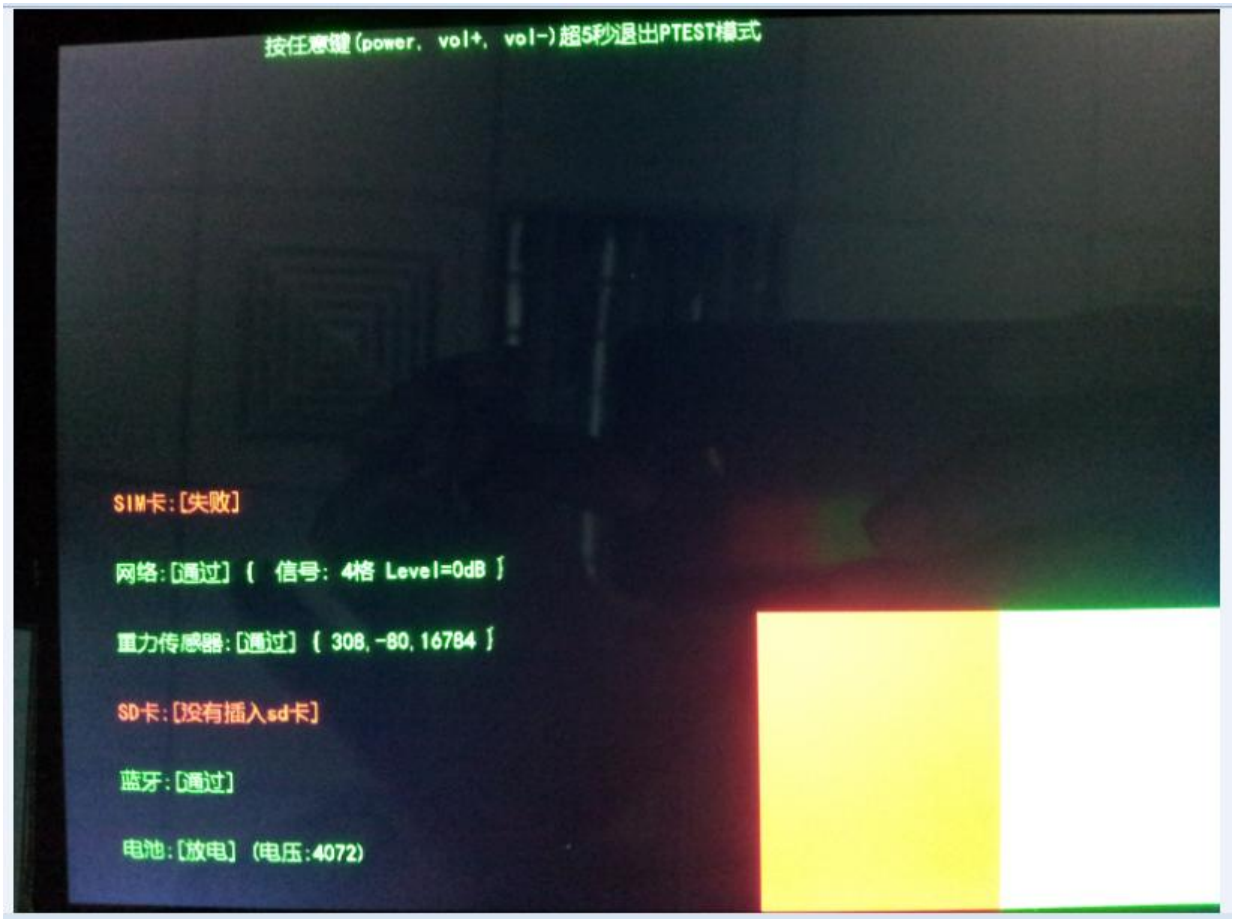
首界面里面有以下几个功能：

1. 连续按 5 次 power 键，可以退出 ptest 模式，进入 Android 模式。
2. 连续按 power、vol+、vol-键，进入 pcba 功能测试界面，进行 pcba 功能测试（这个功能也进行了 key 按键的测试）
3. 在该界面进行 tp 测试，手指可以随意在屏幕上划动。如下图：



PCBA 功能测试界面：

当在首界面中，连续按 power、vol+、vol- 三个按键时，pcba 测试工具会自动进入功能测试界面。功能测试界面如下图：



全部测试项通过之后，可以长按 power、vol+、vol-其中的任意一个按钮 5s，系统会退出 ptest 模式，进入 Android 模式。

3 测试项

3.1 测试项分类说明：

测试项分别有 “红”,“黄”,“绿” 三种颜色表示不同的测试状态

黄色：未测试项或者正在测试的项

绿色：测试通过项

红色：测试未通过项

3.2 测试项详细说明：

3.2.1 重力感应(gsensor)测试

Gsensor 为自动测试项，实时显示读取的 Gsensor 坐标。

3.2.2 无线网络（wifi 测试）

Wifi 为自动测试项，会自动扫描周边的 AP，显示第一个扫描到的的那个 AP 名字和信号强度。

3.2.3 sd 卡（sdcard）测试

Sdcard 为自动测试项，插入 sdcard，如果 SD 卡正常识别到，则会提示测试成功。SD card 必须为 FAT32 格式，不支持其他格式！整个卡只能包含一个分区。如果不符合要求，请通过格式化来格式成标准格式。

3.2.4 屏幕（LCD）测试

LCD 为自动测试项，测试的时候会在屏幕的右下方显示红、绿、蓝三原色的方块，需要测试人员自动判断这三种颜色的方块显示是否正常。

3.2.5 按键（KEY）测试

按键为手动测试项目，PCBA 测试工具从首界面连续按 power、vol+、vol-按键进入到功能测试界面，这个过程中就实现了 KEY 的测试。

3.2.6 TP 测试

TP 为手动测试，直接在 TP 上画线就可。

3.2.7 SIM 卡测试

SIM 卡为自动动测试，插入 SIM 卡。PCBA 会自动读取 SIM 卡里面的 IMSI 信息，并把 IMSI 信息显示到界面上。（这个测试项前提必须要在校准工位上写一个默认的 IMEI 号，）

所有项测试完成后，请长按任意一个按键 5s 后松开，则停止测试，移除 sdcard 和 sim 卡，然后系统会进入 Android 模式。

4 配置文件

PCBA 所有的测试项目通过一个配置脚本 test_config.cfg 来配置，位于 Androidsrc/external/rk-pcba-test/res/test_config.cfg，用户可以根据项目的硬件配置来配置 test_config.cfg 文件，决定要对哪些模块进行测试，以及给自己的测试程序传递相关的参数。

该脚本使用 ini 文件格式，由段、键和值三者组成，通常一个段表示一个模块配置。目前要求该配置文件使用 UTF-8 编码，其他编译格式可能会导致未知错误。

模块配置示例：

测试模块配置模板

```
[example]
display_name= "Example"
activated = 1
program = "example.sh"
category = 0
```

（1）[example]

Example 表示一个配置模块的名称，如果是cfg文件中自带的模块名称，则 不能改动，否则会导致某个测试项不被测试系统启动。

（2）display_name

display_name表示该测试模块在屏幕上显示的名称，可以根据自己的需要修改。该名称最长为64字节，如果为空，则测试程序不会运行。

（3）activated

activated表示是否测试该模块

0：不测试该模块

1：测试该模块

（4）program

该键值目前没用到，可以不用配置

（5）category

category 表示测试方式

0：自动测试

1：手动测试

屏幕测试

```
[Lcd]
display_name= "lcd"
activated     = 1                    //测试该项
program       = "lcdtester.sh"
category      = 0                    //自动测试
run_type      = 1
```

实时时钟测试

```
[rtc]
display_name= "rtc"
activated     = 1                    //测试该项
program       = "rtctester.sh"
category      = 0                    //自动测试
run_type      = 1
module_args = "20121113.160145" //测试rtc的时候  设置的时间
```

无线测试

```
[wifi]
```

display_name= "wlan"

activated = 1 //测试该项

program = "wifitester.sh"

category = 0 //自动测试

run_type = 1

module_path = "/system/vendor/modules/8192cu.ko"

module_args =

WiFi测试，测试结果测试如下：

“网络：[通过] {“testap” 信号强度 4 格 }”

信号强度为实际扫描到的AP的信号强度，与Android上一样，分为0到4格。

重力感应测试

[gsensor]

display_name= "gsensor"

activated = 1 //测试该项目

program = "gsensortester.sh"

category = 0 //自动测试

run_type = 1

蓝牙测试

[bluetooth]

display_name= "bluetooth"

activated = 1

program =

category =

run_type = 1

chip_type = "sofia-3gr" ; rk903, mt6622, rda587x, rda5990,rtk8723as

SD卡测试

[sdcard]

display_name= "SDcard"

activated = 1 //测试该项目

program = "mmctester_sofia.sh"

category = 0 //自动测试

run_type = 1

该配置脚本可以扩展，如果某个模块需要通过配置脚本传递相关参数，可以扩展相关的键值，比如RTC配置项如下

实时时钟测试

[rtc]

display_name= "rtc"

activated = 1 //测试该项

program = "rtctester.sh"

category = 0 //自动测试

run_type = 1

module_args = "20121113.160145" //测试rtc的时候 设置的时间

在具体的测试程序中，可以通过script_fetch api获得设置的相关键值：

int script_fetch(char *main_name, char *sub_name, int value[], int count)

main_name：测试模块的名称，在test_config.cfg文件中[xxxx]

sub_name:键值，比如activated、display_name、module_args等等。

if(script_fetch("rtc", "module_args", (int *)dt, 8) == 0)

```
{
    trncpy(s, dt, 32);
}
```

这里，可获取在配置文件中设置的rtc测试时module_args设置的值。

测试程序中可以通过ui_print_xy_rgba()接口，打印测试结果到屏幕上，由于屏幕空间有限，原则上，尽量打印简单的结果，一个测试项打印一行，成功用蓝色打印，失败用红色打印。

5 字体

说明：PCBA 2.0以后的版本增加了对中文的支持，并可以支持多种字体大小的配置，包括18*18, 20*20, 24*24, 28*28, 32*32, 36*36,可以通过修改minuitwrp/graphics.c 的头文件来包含修改使用不同大小的字库

(输出到屏幕的中文必须是UTF-8编码格式)

6 测试样例扩展

该测试程序允许用户扩展自己的测试样例。如果因为项目需要，用到了该测试程序中目前还未支持到的模块，可以自己添加测试程序，然后集成到测试框架中。

集成方法如下：

（1）先写好自己的测试程序和头文件。测试程序要封装成 void * xxxx_test(void *argv)格式的接口。

（2）确定该测试项为手动测试项或者是自动测试项，并在 test_config.cfg 里面加入想要的配置。

（3）如果是手动测试，在 pretest.c 的 init_manual_test_item()函数中注册自

己的测试代码：

```
int init_manual_test_item(struct testcase_info *tc_info)
{
        printf("%s\n",tc_info->base_info->name);
        if(!strcmp(tc_info->base_info->name, "Codec"))
        {
                tc_info->func = codec_test;
        }
        else if(!strcmp(tc_info->base_info->name, "Key"))
        {
                tc_info->func = key_test;
        }
        else if(!strcmp(tc_info->base_info->name, "Camera_1"))
        {
                tc_info->func = camera_test;
                tc_info->dev_id = 1;
        }
        else if(!strcmp(tc_info->base_info->name, "xxx")) //test item name,defined int test_config.
        {
                tc_info->func = xxxx_test;                //item test function,defined in your test
        }
```

strcmp函数中的“xxx”为在test_config.cfg中定义的测试模块名称[xxxx]

xxx_test是在测试代码中定义的测试函数。

（4）如果是自动测试代码，在pcba测试程序启动的时候，会作为一个线程去启动所有的测试代码，需要在pretest.c的start_auto_test_item（）函数中注册自己的测试函数：

```
int start_auto_test_item(struct testcase_info *tc_info)
{
        int err;
        printf("%s\n",tc_info->base_info->name);

        if(!strcmp(tc_info->base_info->name, "Lcd"))
        {
                err = pthread_create(&screen_tid, NULL, screen_test,screen_msg); //
                if(err != 0)
                {
                        printf("create screen test thread error: %s/n",strerror(er
                        return -1;

                }
        }
        else if(!strcmp(tc_info->base_info->name, "xxx"))
        {
                err = pthread_create(&xxx_tid, NULL, xxx_test,xxx_msg); //
                if(err != 0)
                {
                   printf("create xxx test thread error: %s/n",strerror(err));
                   return -1;

                }
        }
```