RockChip PCBA

测试工具

V2.3

2013-09-04

版本历史

Version	Date	Author	Update note
V1.0	2012-10-23	YXJ	

一、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)、相机(Camera)、实时时钟(RTC)、重力感应(gsensor)、无线（wifi)、SD 卡（sdcard)、按键（KEY)，喇叭耳机（Codec）测试项目。

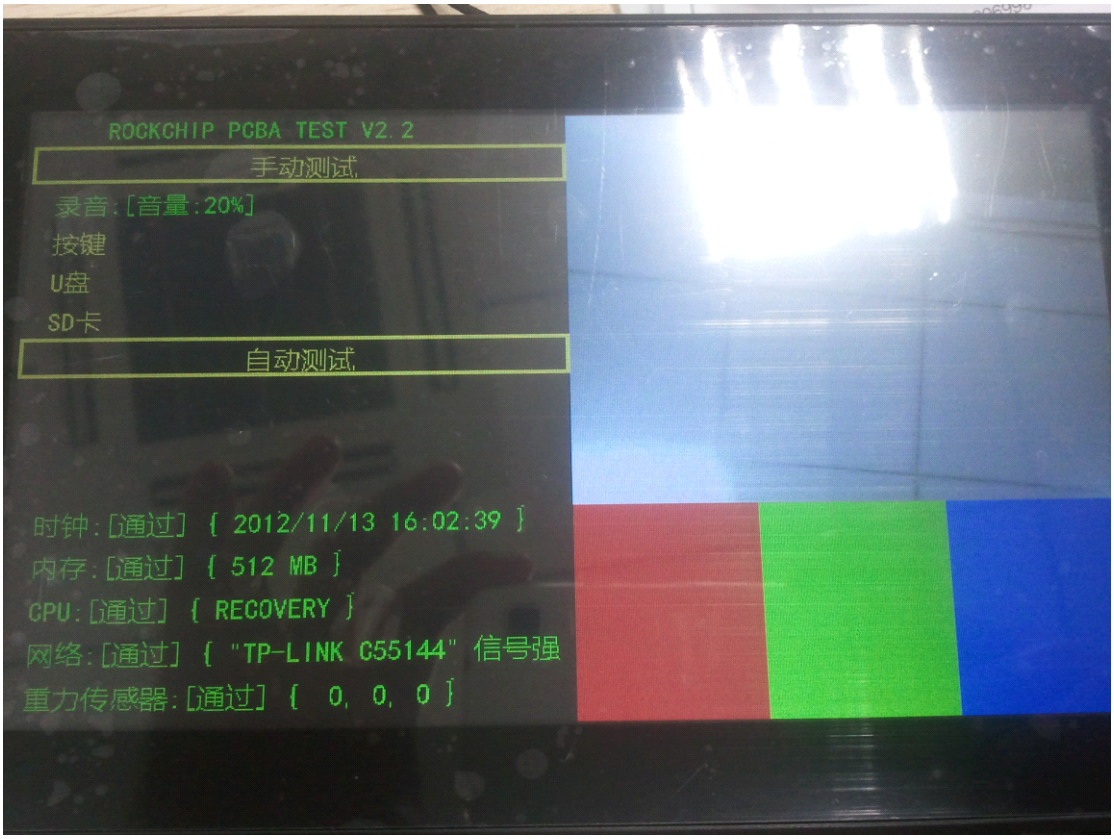
这些测试项目包括自动测试项和手动测试项，LCD、Camera、RTC、Gsensor、wifi、sdcard 为自动测试项，KEY、Codec、Camera_front(前置摄像头)为手动测试项目。

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四部分“配置文件”

二、PCBA 测试固件的生成

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 recovery 的 sbin 目录下。

编译好完整的 Android 固件后，用”瑞芯微创建升级磁盘工具”生成可以从 sdcard 启动的固件（要在工具的功能选择中勾选 PCBA 测试项)，然后用 sdcard 启动 PCBA 板或者是普通的固件，在 parameter 的 CMDLINE 里面加入“bootmode=sdfwupdate”，烧写由 parameter、misc.img.、recovery.img,打包生成的 update.img 系统启动后，会自动进入 PCBA 测试功能，测试界面如图一所示：



图（1）PCBA 测试界面

三、测试项

测试项分类说明：

测试项分为 “自动测试项” 和 “手动测试项”

自动测试项：由系统自动进行测试并判断测试结果,如：网络,内存,时钟，重力传感器等。

手动测试项：需要由人工配合完成或者配合判断测试结果。如：录音，按键，U 盘，SD 卡等。

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

测试项详细说明：

（1）实时时钟（RTC)测试

RTC 为自动测试项，测试的时候会向 RTC 里面设置一个时间，然后读取，判断读取的时间是否和设置的时间相等，如果相等则测试成功，并用蓝色字体打印 rtc test success：年-月-日 时：分：秒 ，失败用红色字体打印 rtc test fail。RTC 测试的时候使用的时间可以在 test_config.cfg 中设置。

（2）重力感应(gsensor)测试

Gsensor 为自动测试项，测试成功会在屏幕上用蓝色字体打印“gesensor test success”以及 x、y、z 方向采集到的数据。如果测试失败，会用红色字体打印“gsensor test fail”。

（3）无线网络（wifi 测试）

Wifi 为自动测试项，测试成功会在屏幕上打印 wlan test success 和搜索到的第一个 AP，测试失败会用红色字体打印 wlan test fail。

（4）sd 卡（sdcard）测试

Sdcard 为自动测试项，插入 sdcard，如果测试成功会在屏幕上打印 sdcard test success 和卡的容量，测试失败会打印 sdcard test fail。注意，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。

（5）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。

（6）相机（Camera）测试

后置 Camera 为自动测试项，测试成功会在屏幕的右上方实时显示采集到的图像，如果没有正常的图像显示，则为测试失败。前置摄像头为手动测试，在测试的时候需要点击屏幕右上方的摄像头区域，摄像头将自动切换到前置摄像头。

（7）按键（KEY）测试

按键为手动测试项目，测试之前请在屏幕左上方点击 KEY（点中后颜色会变化），然后按需要测试的按键，按键松开的时候，会用黄色字体显示检测到的按键。注意，最后测试 POWER 键，POWER 键作为按键测试结束的标志。

（8）耳机喇叭（codec）测试

Codec 为手动测试项目，测试之前请在屏幕左上方点击 Codec，前 3 秒放一段提示声音提示用户在滴声后开始进行录音测试，中间 3 秒存储 MIC 的输入音，此时外放是关的，最后 3 秒播放中间 3 秒所 MIC 的输入音。这个流程是一直循环的。

（9）TP 测试

由于按键和 Codec 两项是通过触摸启动的，如果点击 Codec 和 KEY 能正常的启动测试项，说明触摸正常工作，因而不最专门的触摸测试项。

（10）USB HOST 测试

USB HOST 测试通过测试挂载 u 盘来实现，为自动测试项，测试的时候，请接上 U 盘，如果测试成功，会用红色字体显示 udisk test success 一级检测到的容量。U 盘必须为 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。

（11）DDR 测试

DDR 检测测试：默认开启，系统软件通过对 DDR 内存进行不断读写判断 DDR 地址线是否正常。

DDR 变频测试：默认关闭，开启后测试过程中将对 DDR 进行不断变频测试 DDR 的稳定性，用户需要开启 DDR 变频测试，设置变频范围并且需要配置内核才可支持，详见“配置说明”。

（12）CPU 测试

CPU 负载测试：默认开启，系统不断进行复杂运算，保持系统处于满负载或过载状态，来测试系统稳定性。

CPU 变频测试：默认开启，系统软件通过对多个 CPU 进行不断变频，来测试 CPU 在变频过程中的稳定性。

（由于系统持续处于过载状态，一些处理命令可能延迟，如重力感应器的夹角数值的跳变将会有延迟）

所有项测试完成后，请长按任意一个按键 3s 后松开，则停止测试，移除 sdcard，然后系统才会继续升级。

四、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      = "" ; rk903, mt6622, rda587x, rda5990,rtk8723as //  选择相应的BT芯片型号，默认为空，也就是不测试BT
```

SD卡测试
[sdcard]
display_name= "SDcard"
activated = 1 //测试该项目
program = "mmctester.sh"
category = 0 //自动测试
run_type = 1

USB HOST测试
[udisk]
display_name= "Udisk"
activated = 1 //测试该项目
program = "udisktester.sh"
category = 0 //自动测试
run_type = 1

按键测试
[Key]
display_name= "Key"
activated = 1 //测试该项目
program = "keytester"
category = 1 //手动测试
run_type = 1

音频测试
[Codec]
display_name= "Codec"
activated = 1 //测试该项目
program = "case1" ; case1, case2
category = 1 //手动测试
run_type = 1
delay = 5
volume = 40

case1 :
先放后录模式，测试效率相对低，使用喇叭时不会有啸叫，可在使用喇叭时选择此模式
case2 :
边录边放模式，测试效率高，使用喇叭时会有啸叫，可在使用耳机时选择此模式

录音音量测试，测试结果显示如下，音量根据实际输入变化，范围从0-100%：
“录音音量：[25%]”

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。

内存测试

```
[ddr]
display_name= "ddr"
activated      = 1       //1：开启内存测试，0：关闭内存测试
program         = "memtester.sh" //预留
category       = 0       //自动测试项
freq_test      = 0       //1: 允许变频，  0：禁止变频
min_freq       = 0       //变频范围-最小值
max_freq       = 0       //变频方位-最大值
```

需要内核开启如下配置：

```
CONFIG_DDR_TEST = y
CONFIG_RK_CLOCK_PROC = y
```

五、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)

六、测试样例的扩展

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 p
        }
```

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：

```c
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;

                }
        }
```