目录

[第1章概述 2](#_Toc414377672)

[1.1 软件产生的背景 2](#_Toc414377673)

[1.2 软件功能的要求 2](#_Toc414377674)

[第2章流程图 3](#_Toc414377675)

[2.1 蓝牙测试程序流程 3](#_Toc414377676)

[2.2 蓝牙测试界面显示 4](#_Toc414377677)

[第3章 实现过程 6](#_Toc414377678)

[3.1 蓝牙设备接接口获取 6](#_Toc414377679)

[3.2 蓝牙设备接口实例化 6](#_Toc414377680)

[3.3 蓝牙打开并搜索周围设备 7](#_Toc414377681)

[3.4 搜索到的设备动态显示 8](#_Toc414377682)

[第4章 总结 10](#_Toc414377683)

# 第1章 概述

## 1.1 软件产生的背景

Android系统在开启多个应用后，性能表现有所下降，为维持长时间的Android系统处于健康状态，于是对后台操作进行限制，控制后台应用数量及限制后台拉起应用，采用白名单的方式来避免因清理导致的功能异常和满足特定需求。本文中的应用主要是为了满足用户个性化的设置，来提高用户体验。

## 1.2 软件功能的要求

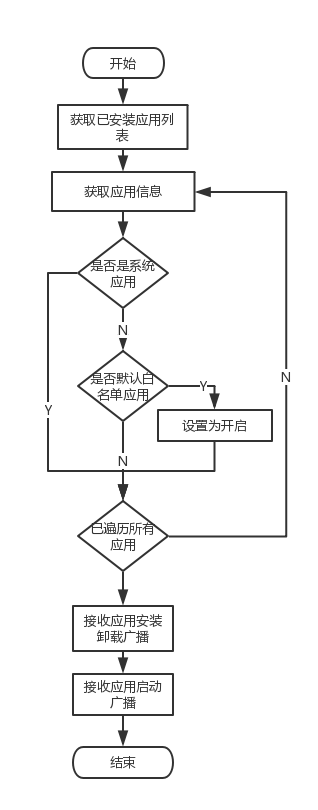
天珑MTK平台性能优化白名单配置应用软件设计要求如下:

1. 能够对用户设置生效，提供友好界面管理及设置名单。
2. 能够检测应用安装后的第一次启动，发送通知提示用户是否将应用配置进白名单。

# 第2章 流程图

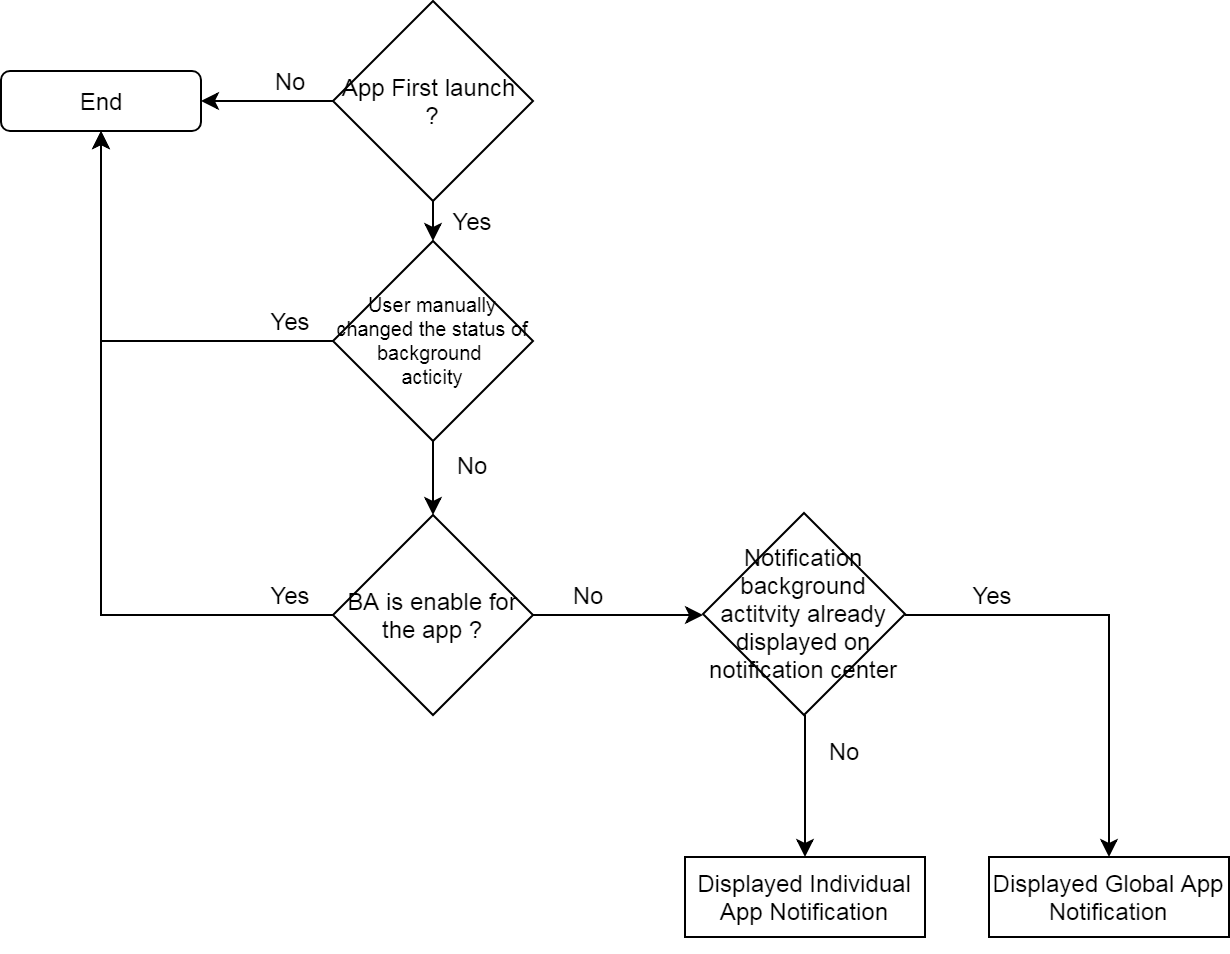
## 2.1 MTK平台性能优化白名单配置应用软件

下图是程序初始化流程图：



本程序流程图描叙了初始化工作流程图。通过获取系统过已安装的应用来初始化数据，并通过数据库保存初始话后的状态。通过接收系统广播，来对用户安装新应用做出相应的操作。

下面是提示用户在第一次启动应用时使用Notification通知用户进行设置的流程图：



## 2.2 用户界面及通知显示

（1）因该应用做为设置中的一项，功能入口如图1所示，界面风格仿照系统界面，如图2所示。

（2）在新安装应用第一次启动的过程时，通过Notification通知用户进行是否加入白名单进行配置。通知形式及内容如图3 及图4所示。

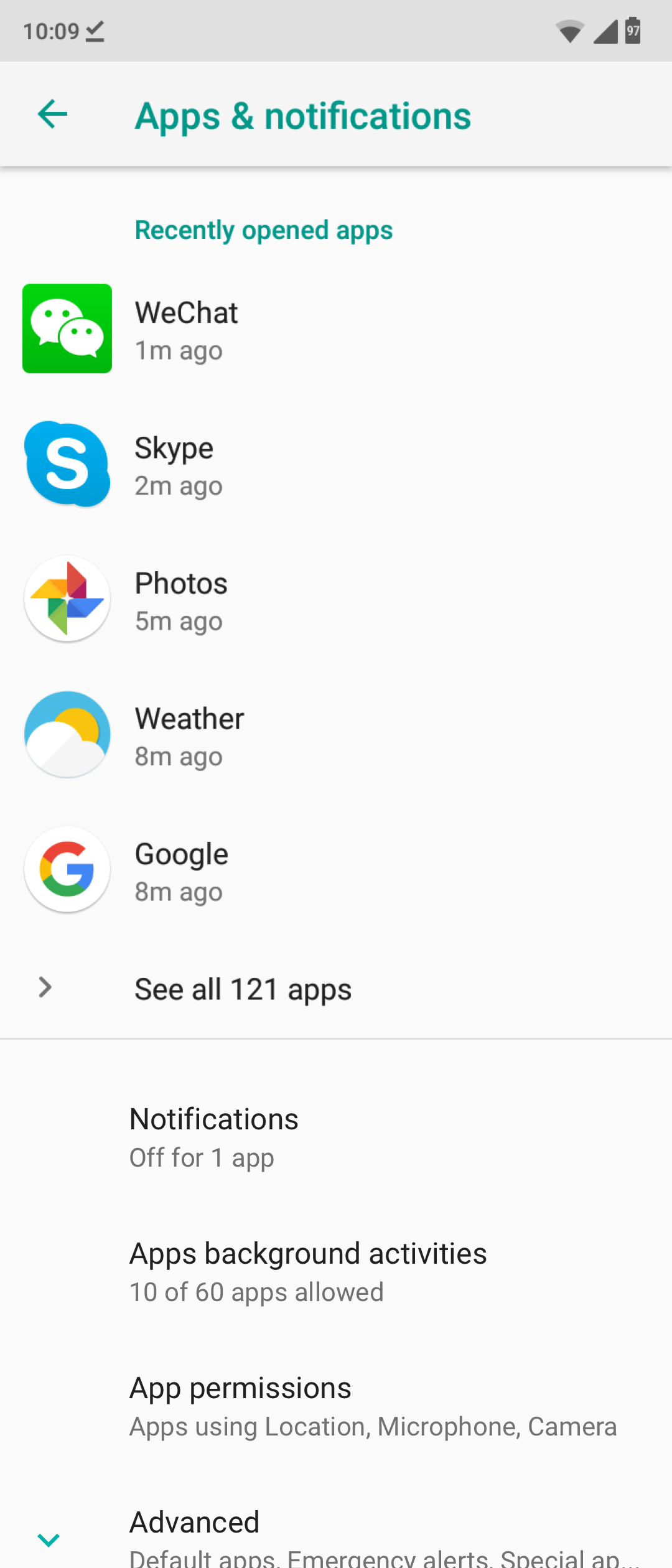
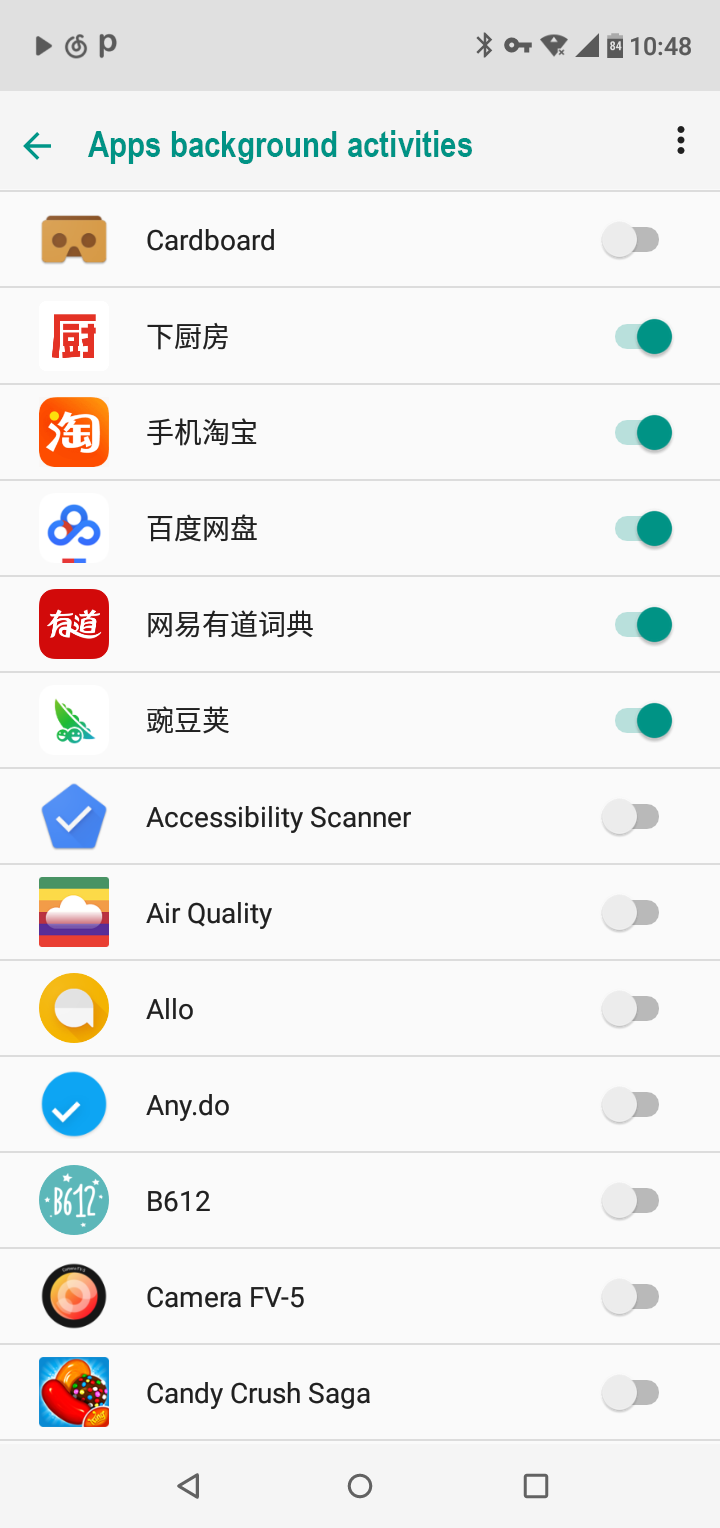


图 1 图 2



图 3 图 4

# 实现过程

## 3.1 蓝牙设备接接口获取

本软件使用了android标准的HAL层接口来获取蓝牙接口并进行操作。HAL可以获取所有硬件设备的接口，获取时需要定义hw\_module\_t和hw\_device\_t，然后通过MODULE ID来进行匹配module，匹配完成后打开device，而device本身有get\_interface来获取接口。本软件实现方法如下：

int HAL\_load(void)

{

int err = 0;

hw\_module\_t\* module;

hw\_device\_t\* device;

err = hw\_get\_module(BT\_HARDWARE\_MODULE\_ID, (hw\_module\_t const\*\*)&module);

if (err == 0)

{

err = module->methods->open(module, BT\_HARDWARE\_MODULE\_ID, &device);

if (err == 0) {

bt\_device = (bluetooth\_device\_t \*)device;

sBtInterface = bt\_device->get\_bluetooth\_interface();

}

}

return err;

}

bt\_device是定义的全局设备，sBtInterface是定义的全局设备接口。此后的所有针对设备的操作都需要用到这两个变量。

## 3.2 蓝牙设备接口实例化

蓝牙接口获取到之后，要对接口中的功能函数进行实例化，这样才能使接口成功调用并从回调函数得到运行的结果。本软件中实现了三个功能函数的实例化，分别为adapter\_state\_changed，devices\_found\_callback，discovery\_state\_callback。

adapter\_state\_changed是监测蓝牙设备打开是否成功，代码如下：

static void adapter\_state\_changed(bt\_state\_t state)

{

if (state == BT\_STATE\_ON) {

bt\_enabled = 1;

} else {

bt\_enabled = 0;

}

}

devices\_found\_callback是当搜索到一个新设备之后将所有搜到的设备属性返回，其中包括设备名称和设备MAC ADDR。代码如下：

static void devices\_found\_callback(int num\_properties, bt\_property\_t \*properties){

……

pTmpBtInfo = (BT\_Info \*)malloc(sizeof(BT\_Info));

memset(pTmpBtInfo, 0x0, sizeof(BT\_Info));

for(i = 0; i < num\_properties; i++){

if (properties[i].type == BT\_PROPERTY\_BDADDR){

memcpy(pTmpBtInfo->bd\_addr, (char \*)properties[i].val, properties[i].len);

}

if(properties[i].type == BT\_PROPERTY\_BDNAME){

strcpy(pTmpBtInfo->bd\_name,(char \*)properties[i].val);

}

}

……

}

discovery\_state\_callback是监测本机的蓝牙是否可被其它设备发现，代码如下：

static void discovery\_state\_callback(bt\_discovery\_state\_t state) {

discovery\_state = (state == BT\_DISCOVERY\_STARTED) ? true : false;

……

}

然后使用interface的init方法，将这些实例化函数与接口联系起来。

static bt\_callbacks\_t bt\_callbacks = {

sizeof(bt\_callbacks\_t),

adapter\_state\_changed,

devices\_found\_callback, /\* device\_found\_cb \*/

discovery\_state\_callback, /\* discovery\_state\_changed\_cb \*/

……

};

status = (bt\_status\_t)sBtInterface->init(&bt\_callbacks);

check\_return\_status(status);

## 3.3 蓝牙打开并搜索周围设备

调用BT interface 接口enable来打开蓝牙并下载firmware，代码如下：

status = (bt\_status\_t)sBtInterface->enable();

之后通过check\_return\_status(status);来判断其返回值。如果返回值是fail，则打开蓝牙失败，此时通过text\_waiting->set\_text("Status: INIT FAILED");在屏幕上显示“INIT FAILED”来提示测试人员。如果返回值是success，则通过BT interface来控制蓝牙设备开始搜索周围设备，代码如下：

static bool startDiscovery(void) {

unsigned char result = false;

if (!sBtInterface) return result;

status = (bt\_status\_t)sBtInterface->start\_discovery();

result = (status == BT\_STATUS\_SUCCESS) ? true : false;

return result;

}

此后如果看搜到蓝牙设备，BT interface会通过之前注册的回调函数devices\_found\_callback来通知本软件有新设备发现，从而触发屏幕显示更新。

## 3.4 搜索到的设备动态显示

当搜索到蓝牙设备，devices\_found\_callback执行时，会更新设备列表：

mTmpBtInfoForList = g\_pBtListHear;

while((mTmpBtInfoForList != NULL) && (mTmpBtInfoForList->pNext != NULL))

mTmpBtInfoForList = mTmpBtInfoForList->pNext;

mTmpBtInfoForList->pNext = pTmpBtInfo;

g\_pBtListHear是定义的全局链表，新搜到的设备会添加到此链表中。然后在updateTextInfo中遍历链表中所有设备，如果该设备已获取到名称，则显示其名称，如果未获取名称，则显示其MAC ADDR。代码如下：

void updateTextInfo(BT\_Info \*pTmpBtInfoForList)

{

……

while(pTmpBtInfoForList){

memset(cBuf,0x00,sizeof(cBuf));

//判断是否获得设备的名称

if(strlen((const char\*)pTmpBtInfoForList->bd\_name)){ char\*)pTmpBtInfoForList->bd\_name);

}else{

//如未获得名称,把ADDR按12位标准格式打印在屏幕

sprintf(cBuf, "%02x:%02x:%02x:%02x:%02x:%02x",

pTmpBtInfoForList->bd\_addr[0], pTmpBtInfoForList->bd\_addr[1], pTmpBtInfoForList->bd\_addr[2],

pTmpBtInfoForList->bd\_addr[3], pTmpBtInfoForList->bd\_addr[4], pTmpBtInfoForList->bd\_addr[5]);

}

pTmpBtInfoForList = pTmpBtInfoForList->pNext;

if(i<10){

text[i]->set\_text(cBuf);

i++;

}

}

……

}

最后所有的数据通过FTM的显示控件mmi\_text打印在屏幕上。搜索结束时会触发discovery\_state\_callback，此接口会修改Scanning\_state，通知到updateTextInfo。然后在屏幕上显示“Status: Scanning Completed”。到此蓝牙完整的测试流程就结束了。

# 总结

天珑高通平台FTM模式蓝牙测试功能软件测试效率高，能实现工厂需要的所有功能，且简洁明了。它可以实时显示搜到的蓝牙设备，从而让测试人员快速判断测试结果，并且能根据不同的情况输出测试状态，如出问题也可以文件返修人员定位问题所在。此外本软件使用android的标准HAL接口实现，兼容性高，方便移植到其它平台。本软件的设计为工厂生产节约了大量时间，对生产的成本控制做出很大贡献，也为以后基它手机平台生产提供了可借鉴的方法。