超显科技
工业安卓串口屏
开发指导手册 V5.3

Android LCMs Development Guide
V5.3

超显安卓屏模组使用须知

获取规格说明书与开发指导手册

在购买或使用安卓屏模组前，可以先了解安卓屏模组的特性与接口。访问超显科技-产品中心（http://hyperlcd.com/product/），通过点击对应型号可以找到指定型号的安卓产品数据书与超显安卓屏开发指导手册。

1.安卓产品数据书

文档介绍安卓模组的性能与功能特性，提供售后与技术支持的说明。

2．超显安卓屏开发指导手册

详细介绍开发环境的搭建，提供串口开发、蜂鸣器使用等实例 ，助力您快速上手安卓模组。文档提供了安卓模组烧录系统、系统自定义的操作步骤 。

⚠ 注意：镜像文件您可以洽询与您对接的销售。镜像工具、镜像烧录工具您可至超显科技-技术支持（http://hyperlcd.com/downloads/）下载，对应工具请参考开发指导手册中的表格。使用安卓屏模组时遇到无法解决问题，可查阅超显安卓屏开发指导手册第四章注意事项章节 。若无法解决您遇到的问题，请与您对接的销售人员寻求技术支持。

获取安卓屏模组配套软件与源码

为助力您更快上手安卓屏模组软件的开发，我们为您准备了多款示例程序。您可至超显科技-示例程序 （http://hyperlcd.com/downloads/）下载示例。

1. 超显科技安卓屏项目演示示例

   项目内涵盖了串口开发、人脸识别、居家生活的应用的代码参考示例。

2. 串口实例程序参考示例

   串口实例程序帮助您快速上手串口开发，安卓串口使用的代码参考示例。

3. 隐藏导航栏示例

   使用系统级接口，隐藏导航栏，全屏展示应用的代码参考示例。

4. 蜂鸣器示例

调用蜂鸣器，提供使用蜂鸣器与禁用蜂鸣器的代码参考示例。

我们可以提供的帮助

在您购买安卓屏模组后，我们将会为您提供安卓屏模组软/硬件上有限的帮助。

- 硬件方面

支持客户安卓产品硬件方面疑问解答

安卓屏模组接口方面疑问解答

- 软件方面

安卓屏模组软件开发环境方面疑问解答

安卓屏模组系统 API 接口方面疑问解答与提供

安卓屏模组软件资料支持

⚠ 注意：在您购买安卓屏模组后，您可能需要在安卓屏模组上运行您的项目，安卓系统支持的可执行文件后缀名为.apk，只有后缀名为.apk 的项目才可以运行在安卓屏模组上。使用的开发工具为 Android Studio 或 Eclipse，目前 Android Studio 为主流的安卓集成开发工具。

📌 提示：开发安卓应用所支持的编程语言有 JAVA、Kotlin、Flutter 等，若您掌握以上任意一种编程语言即可自行开发安卓应用项目。若无法自行开发安卓应用，我们可以提供安卓应用开发的服务，具体收费报价这需要您与销售进行对接。您也可以寻求可靠、稳定的安卓应用开发团队，我们会提供有限的技术资料支持。

常见问题 Q&A

1. 我该怎么选择适合我的安卓屏模组?

通过超显科技官网的产品中心界面通过配置进行选择安卓模组，或通过超显科技淘宝店铺与客服沟通选择产品。

2. 你们的安卓屏模组使用稳定吗?

我们的安卓屏模组针对行业能做到 24 小时不死机、不断电工作，软件后台用 4G、WIFI 也能长时间稳定工作。

3. 你们的安卓屏模组进行了哪些测试?

我们对产品进行了断电测试、温度测试、掉电测试、接口压力测试、高低温测试，出厂前会进行老化测试，保证产品的稳定性。

4. 安卓屏模组可以选配哪些版本?

安卓屏模组在标准版的基础上，可以添加 WIFI 模块（不含蓝牙）、WIFI 模块（含蓝牙）、4G 模块 （不含 GPS）、4G 模块（含 GPS）。

5. 安卓屏模组不停的断电会有问题吗?

我们进行断电测试 10 万次以上，产品完全正常，不会有问题。

6. 安卓屏模组可以做防水吗?

屏幕防水需求：红外触摸屏都是防水的，您可选用红外触摸，也可定制电容触摸防水屏（默认电容 触摸不防水）。

整机防水需求：客户需要自己做防水结构，我公司 K 系列、R 系列外壳只能做到局部防水，接口均未 做处理

7. 我需要中性的包装可以吗?

可以的，开机界面及机器上面都没有超显的 logo，只有包装盒、返修卡和合格证上有超显的 logo，如您需要中性包装，与销售人员说明，我们可以抹掉超显 logo 发货。

8. 安卓屏模组可以进行蓄电池供电吗?

我们安卓屏模组没有蓄电池供电，不支持定时开关机。上电开机断电关机，不会对系统造成损害。

9. 我购买了产品，你们后续会有技术支持么？支持范围是哪些?

购买超显产品之日起，12 个月内，一线工程师在工作日、工作时及时提供支持。

支持范围：

- 支持用户能运安卓系统及相关接口测试程序
- 支持安卓系统的常见配置

◆ 支持客户安卓产品硬件方面支持

10. 我购买了产品，你们可以保修吗?

我们提供 7 天无理由服务，一年保修，终身维修（有偿）的服务。CPU 和配件不在保修范围内，由于不正确人为使用因素或不可抗力引起的损坏不在保修范围。

11. 你们有没有技术支持的文档?

您可在超显科技官网顶部的技术支持 - 程序下载 中找到您需要的文档。

12. 什么是烧录镜像，为什么要烧录系统镜像?

烧录镜像通俗的说就是给模组刷入所支持的系统，可以给模组刷入 Android 或者 Ubuntu 系统。如您有其他特殊设置，系统功能更新、 添加新的外接模块等需二次烧录安卓系统。

13. 我该怎么烧录系统，MAC 电脑可以烧录系统镜像吗?

超显科技官网下载超显安卓屏使用手册，查看 3.6 固件烧录程序章节，本章节详细介绍了烧录系统的过程，烧录镜像前需要准备好与安卓模组对应版本的安卓系统镜像。MAC 电脑端暂不支持烧录系统镜像，请使用 Windows 系统进行烧录镜像。

14. 我知道怎么烧录，我要去哪里下载你们的系统镜像?

暂不提供镜像下载链接，您可联系与您对接的销售获取最新系统镜像。

15. 你们的镜像系统有那些可以配置修改的?

镜像支持替换开机 logo、替换开机动画、修改默认壁纸并支持安卓应用 APK 添加到镜像内。详细操作步骤请查看超显安卓屏使用手册 3.5 开机配置章节。

16. 我需要在安卓模组上做安卓的项目，你们可以做么？需要提供什么?

可以，我们支持软件开发定制，需要您提供软件开发需求与已有的资料，我们需要评估项目的周期与报价，评估完成后会与您对接。

17. 我需要在安卓模组上做安卓的项目，我找了别的团队，需要给他们提供什么?

需要提供安卓产品数据书与超显安卓屏使用手册，产品数据书提供安卓模组详细的接口与安卓模组使用规范。如果以上资料无法解决问题，也可以与我们的技术进行对接。

18. 以上都解决不了我遇到的问题，我该怎么找到你们的技术来寻求支持呢?

您可以与您对接的销售沟通您遇到的问题，我们会分配技术人员与您对接。

版本记录

文档版本	更新日期	说明
V5.3	20230919	优化内容

目 录
catalogue

第一章 硬件与接口介绍

1.1 产品接口

超显安卓模组，是基于瑞芯微 CPU 核心板，搭载不同底板接口的整体液晶显示模组方案。在不同规格的 CPU 芯片支持下，模组通过串口、WIFI、USB 等和主控机进行连接。其中串口 I/O 是产品的主要连接通讯方式。

- 串口通讯

以 7 寸搭载 RK3288 的产品为例，超显安卓模组支持 3 路 RS232 或 TTL 电平串口，1 路 RS485 串口/RS232/TTL 电平串口，其中 ttyCOM0、ttyS1、ttyS3 是普通串口，可以用于串口通讯，最高支持 115200 波特率,ttyS4 是默认 RS232 可以修改为 RS485。

提示：在其他规格下的产品，ttyCOM0 口也可支持 USB 转串口进行扩展。

- USB 接口

以 7 寸搭载 RK3288 的产品为例，产品带三路 USB HOST 和一路 USB OTG 支持 USB 鼠标，U 盘，摄像头等外设；USB OTG 用于安卓开发工程师进行安卓应用开发。

超显 RK3288 安卓屏的产品接口如下图 1-1-1 所示：

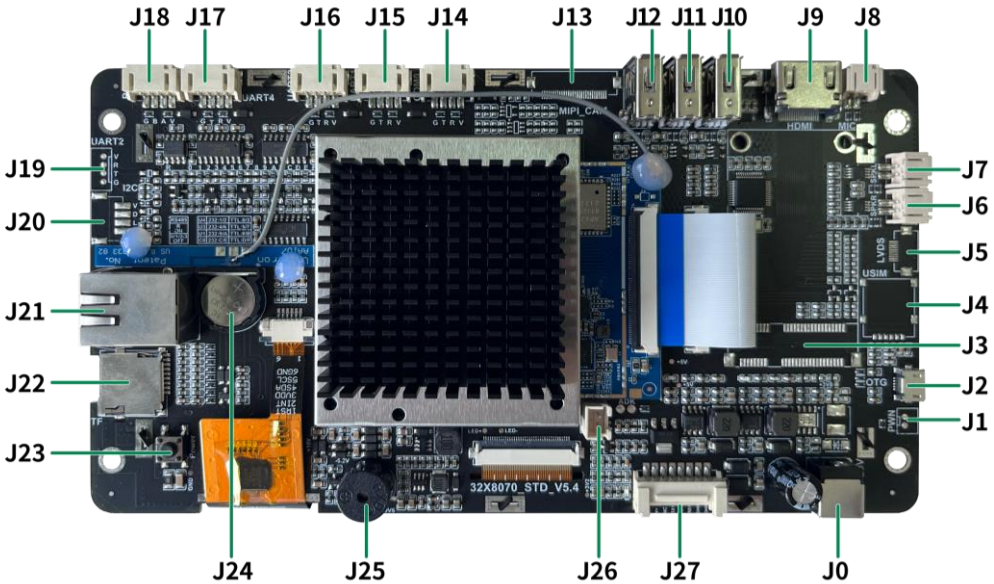


图 1-1-1 7 寸 RK3288 标准接口说明

标号	接口名称	说明
J0	电源接口	推荐 12V/2A 电源供电（最大电压支持 DC 6V-16V）
J1	休眠唤醒接口	控制屏幕系统开关机
J2	USB_Micro 接口	OTG 功能/APP 调试接口/固件升级接口
J3	MINI PCIE 4G 接口	4G 网口接口/GPS 定位接口（选配）
J4	SIM 卡 接口	当前支持 NANO 模式卡座（选配）
J5	LVDS 接口	预留
J6	SPK_R 接口	音频右声道输出接口
J7	SPK_L 接口	音频左声道输出接口
J8	MIC 接口	麦克风接口
J9	HDMI 接口	HDMI 视频输出接口
J10	USB_HOST3 接口	支持 USB 外接输入输出设备
J11	USB_HOST2 接口	支持 USB 外接输入输出设备
J12	USB_HOST1 接口	支持 USB 外接输入输出设备/支持双 USB 并行使用
J13	MIPI 摄像头 接口	预留
J14	COM0 通讯接口	设备名 COM0 接口定义：5V、RXD、TXD、GND
J15	UART1 通讯接口	设备名 TTYS1 接口定义：5V、RXD、TXD、GND
J16	UART3 通讯接口	设备名 TTYS3 接口定义：5V、RXD、TXD、GND
J17	UART4 通讯接口	设备名 TTYS4 接口定义：5V、RXD、TXD、GND/与 J18 定义相同
J18	RS485 通讯接口	设备名 TTYS4 接口定义：5V、A、B、GND/与 J18 定义相同
J19	调试接口	预留
J20	IIC 通讯接口	接口定义：5V、SDA、SCL、GND （预留）
J21	RJ45 网口接口	支持 10M/100M/1000M 网络
J22	TF 卡 接口	可做内存扩展

标号	接口名称	说明
J23	休眠唤醒开关	控制屏幕系统开关机
J24	RTC 电池座	采用 RC1220 纽扣电池，提供系统时钟
J25	蜂鸣器	报警提示
J26	散热器接口	可外接散热风扇
J27	用户接口	接口定义：12V、12V、NC、TXD、RXD、RXD、GND、GND

1.2 硬件修改 RS232/RS485/TTL

超显安卓模组，串口默认为 RS232 模式，如果您需要使用 RS485 或者是 TTL 功能，请参考下方方法进行修改,以超显 RK3288 安卓屏演示。

1. 检查安卓模组是否支持 RS485/TTL

查看超显安卓屏产品规格书中的串口接口参数，如果支持 RS485/TTL 则代表产品可以修改串口模式。如下图 1-2-1 所示。

接口参数					
名称	测试环境	最小值	典型值	最大值	单位
波特率	标准	1200	9600	115200	bps
串行模式	用户自定义	1200	—	115200	bps
串口模式	4 路串口 (3*RS232/TTL, 1*RS232/TTL/RS485)				
用户接口方式	标准通讯协议. 8Pin_2.0mm 串口线/座.				
USB	USB 调试*1. USB HOST*3.				
以太网	支持 10m/100m/1000m 以太网				
WIFI/蓝牙	支持 802.11b/g/n Wi-Fi 无线网络.				

图 1-2-1 串口模式

提示：超显安卓屏产品规格书可以前往超显科技官网（产品中心 – 北京超显科技有限公司 (hyperlcd.com)）进行下载。若无法解决您遇到的问题，请与您对接的销售人员寻求技

术支持。

2. 检查模组丝印并修改串口模式

查看安卓模组背部左上角丝印，如下图 1-2-2、1-2-3。

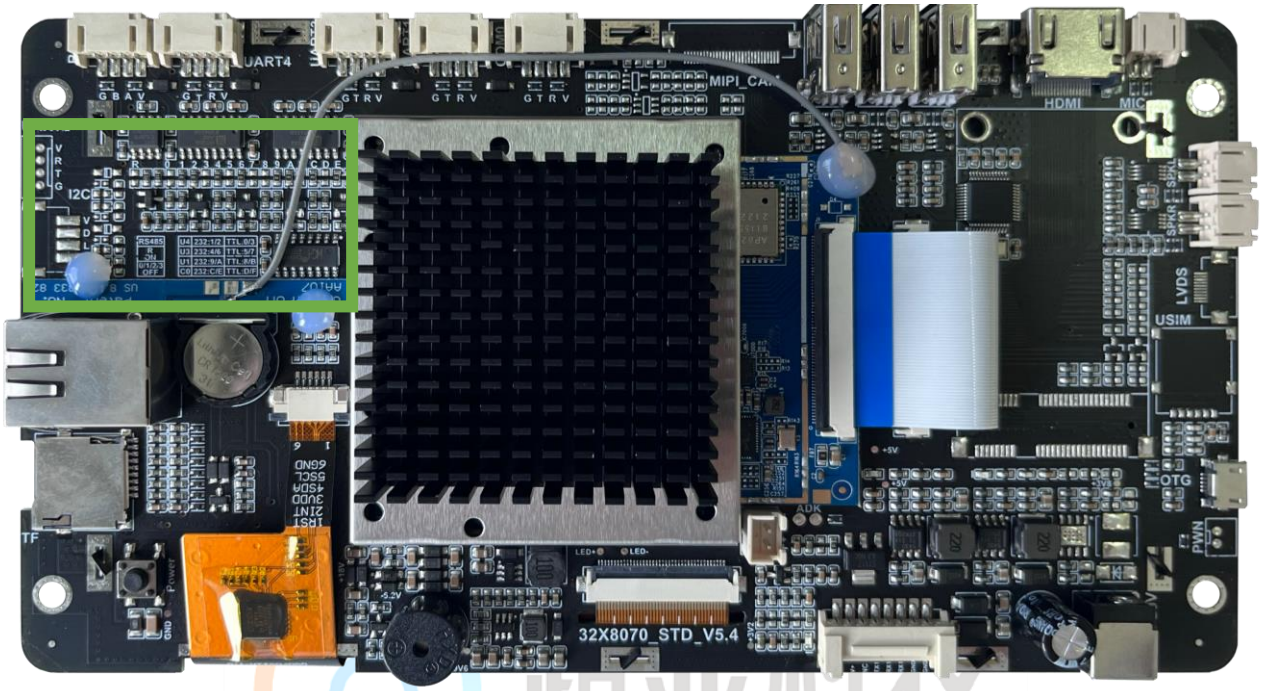


图 1-2-2 整体图

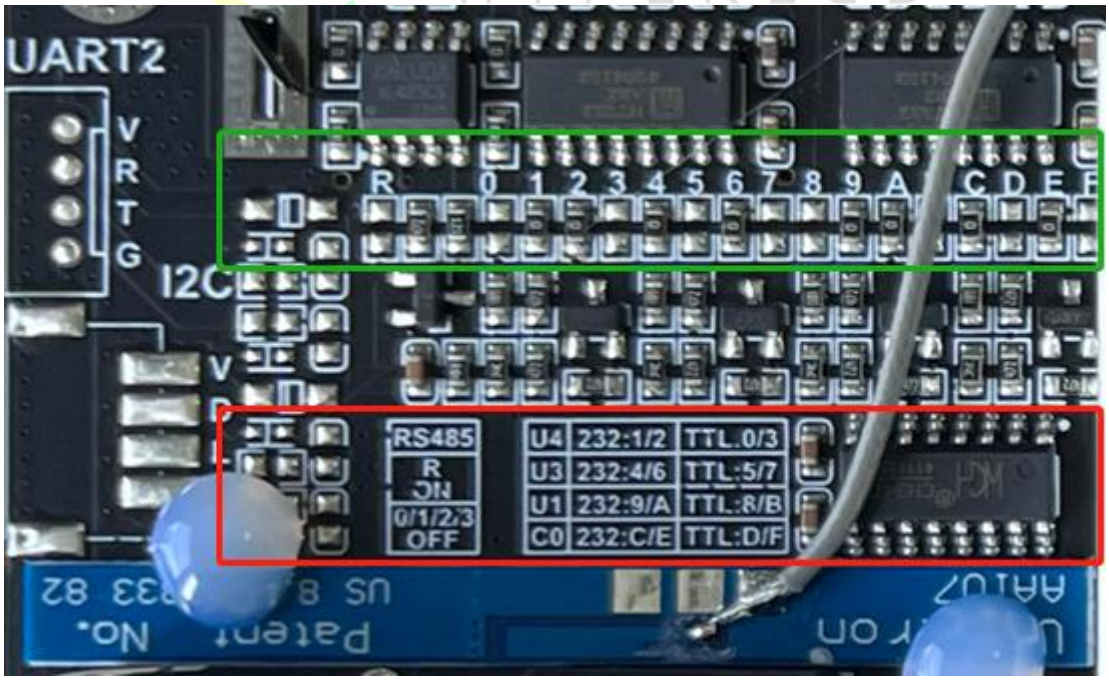


图 1-2-3 修改区域

上图 1-2-3 分别标注了绿框与红框，绿框中是改动区域，红框中是参考区域。

以更改 RS485 为例，请参考下图 1-2-4.

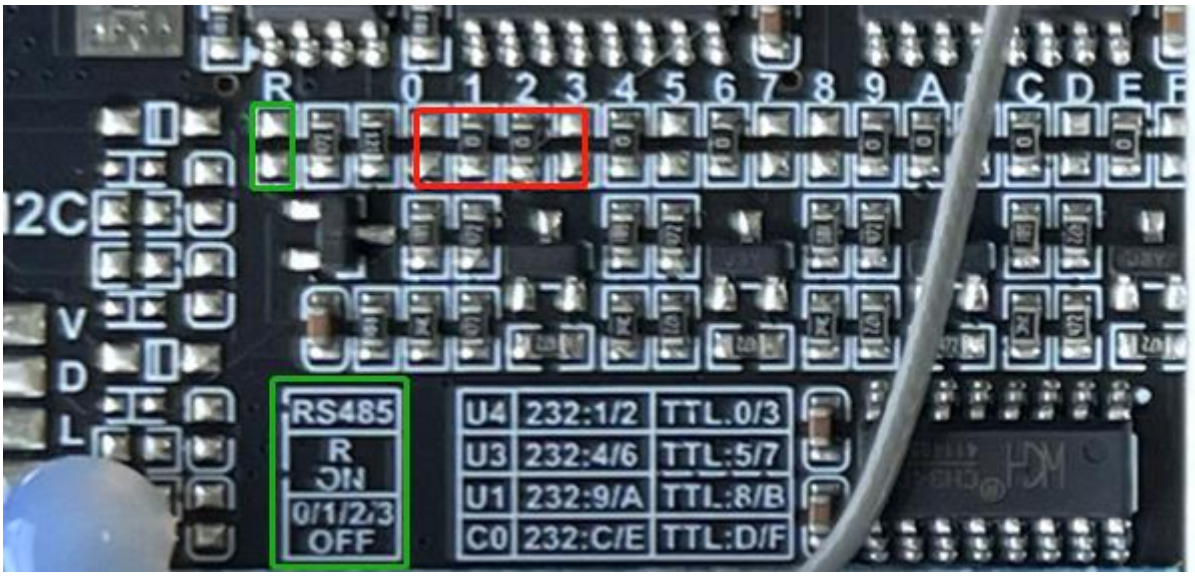


图 1-2-4 修改使用 RS485

上图 1-2-4 分别标注了绿框与红框，绿框中是改动区域，添加上 0Ω电阻，红框中是移除区域，移除其中的 4 个电阻。修改完成后，此时 RS485 接口可以正常通讯。

⚠ 注意：如果您还需要使用其他串口，只需要在 R 区域补上一颗 0Ω电阻，0/1/2/3 区域无需改动。若无法解决您遇到的问题，请与您对接的销售人员寻求技术支持，我们会分配硬件工程师帮助您解决问题。

以更改 UART4 为 TTL 为例，请参考下图 1-2-5。

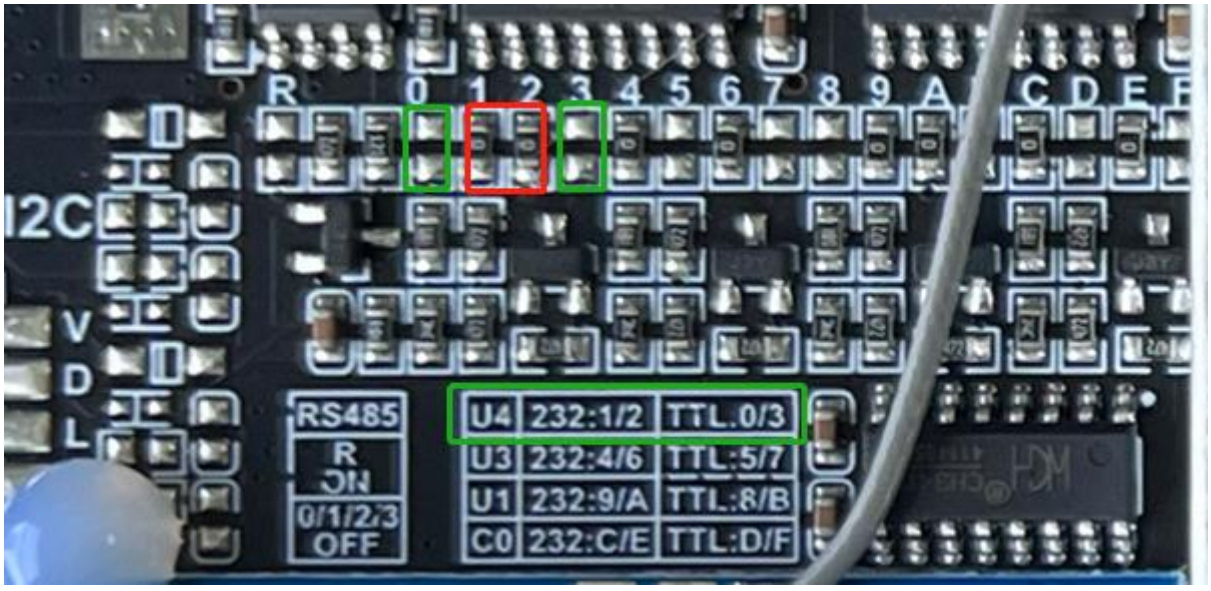


图 1-2-5 修改使用 UART4 TTL

上图 1-2-5 分别标注了绿框与红框，绿框中是改动区域，添加上 0Ω电阻，红框中是移除区域，移除其中的 2 个电阻。修改完成后，此时 UART4 接口可以正常使用 TTL 通讯。

⚠ 注意：若无法解决您遇到的问题，请与您对接的销售人员寻求技术支持，我们会分配硬件工程师帮助您解决问题。

3. 检查串口通讯功能

在安卓模组上安装 ComAssistant 串口调试软件，如下图 1-2-6，在电脑上打开任意一款串口调试软件，如下图 1-2-7。

图 1-2-6 ComAssistant 串口调试软件

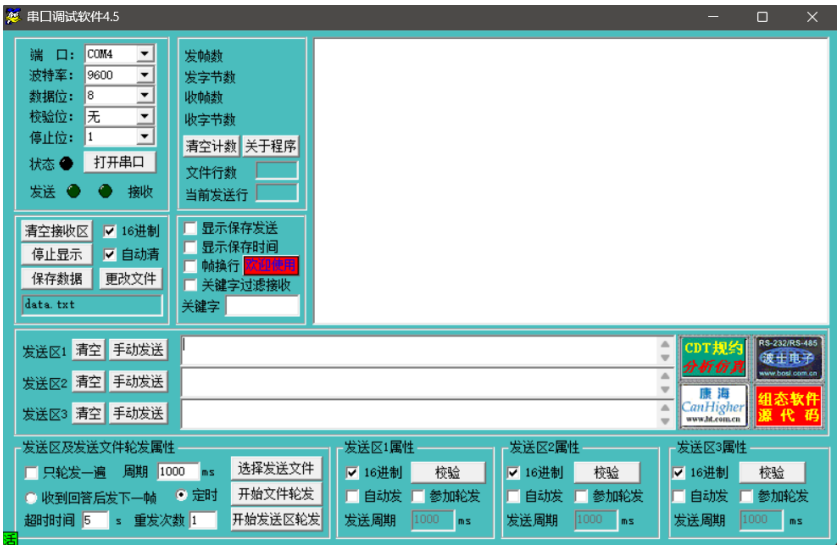


图 1-2-7 Windows 串口调试软件

将波特率、校验位、停止位设置为一致，打开串口进行收发数据，可以测试一段时间查看其稳定性。

⚠ 注意：ComAssistant 串口调试软件下载地址：点我下载。Windows 版本串口调试软件暂不提供。若无法解决您遇到的问题，请与您对接的销售人员寻求技术支持，我们会分配硬件工程师帮助您解决问题。

1.3 进入系统烧录模式

超显安卓模组在出厂前均会预装相应版本的安卓系统，如您有其他特殊设置，例如修改开机动画、修改开机 logo、定制系统功能需要烧录系统，请按照以下方法进入系统烧录模式,以超显 RK3288 安卓屏演示。

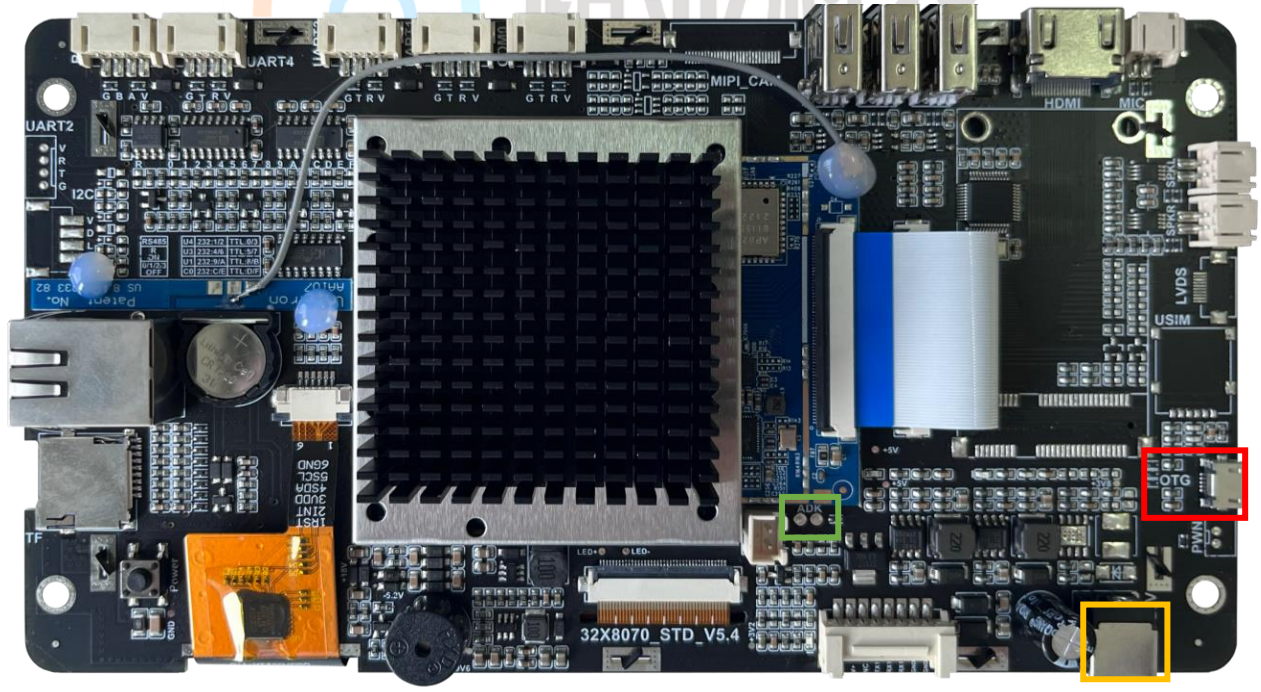


图 1-3-1 ADK

上图 1-3-2 绿框标注了进入系统烧录模式的短接点（ADK），绿框中包含了两个触点。请严格按照以下顺序执行

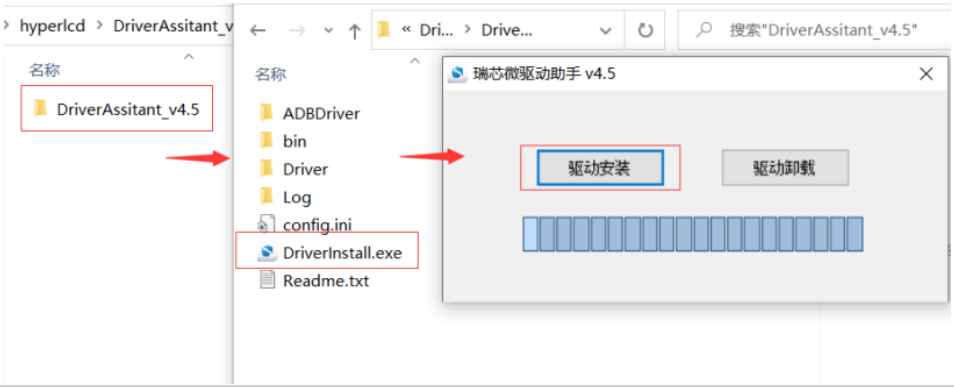
1. 将 USB-MicroUSB 数据线接到红框内 OTG 接口并连上电脑。
2. 使用镊子短接 ADK，将电源线插到黄框内电源接口，通电，此时不要放开 ADK 的短接。
3. 当瑞芯微 Android Tool 提示发现一个 LOADER 设备后断开镊子短接，系统烧录模式进入完毕。

警告：ADK 短接点部分老款产品烧录管脚位于产品核心板上方，无标示。请勿随意短接未标示的触点，如要改动硬件请在硬件工程师的指导下对设备进行修改、短接，详情请接洽您的销售人员寻求相应的技术支持。

注意：USB-MicroUSB 数据线一定要是可以进行数据传输的数据线，一般为 4 芯，请注意 2 芯数据线并不能传输数据，无法与模组、电脑进行通信。若遇到模组接到电脑上没反应请尝试更换数据线。

若数据线确定是可以通信的，请安装瑞芯微安卓驱动，然后重启电脑。

瑞芯微驱动下载链接：http://hyperlcd.com/download/driverassitant_v4-5/



提示：硬件相关详细参数请查看超显安卓屏产品规格书，规格书可以前往超显科技官网（产品中心 – 北京超显科技有限公司 (hyperlcd.com)）进行下载。若无法解决您遇到的问题，请与您对接的销售人员寻求技术支持。

通过工具直接进入系统烧录模式

将安卓模组连上电脑，点击切换后模组会自动进入系统烧录模式，无需短接 ADK.

通过 ADB 进入系统烧录模式

在模组开机状态下，进入系统，执行 adb 命令将模组切换到烧录模式

```
adb reboot loader
```

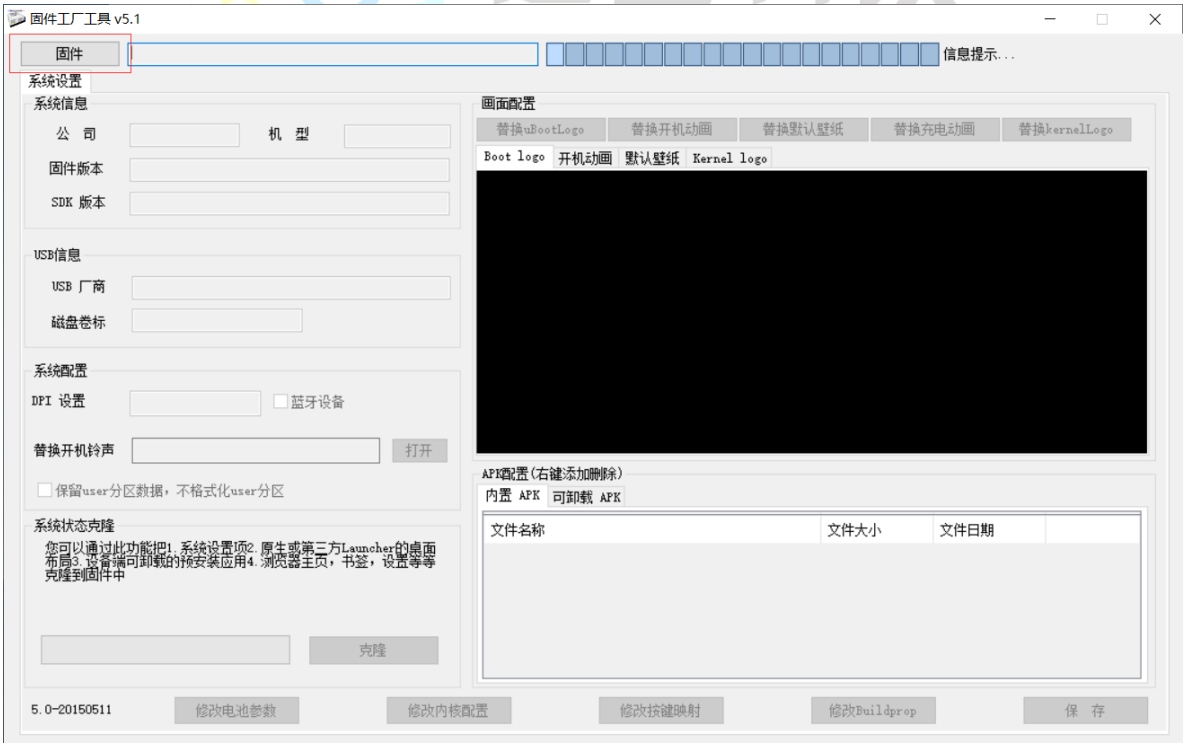
第二章 系统的定制与烧录

2.1 系统定制

超显安卓模组内加载的安卓开源系统，拥有开放的 root 权限，支持修改开机动画、默认壁纸时区、屏幕方向等自定义设置。这些设置包括但不仅限于开机图片或动画、墙纸、启动项等。系统定制将会为您的项目美观度添砖加瓦。

若您需要进行系统的开机配置，请在超显官网选择对应开发板版本的固件工厂工具。对应固件工厂工具请参考下表。

固件工厂工具	适用产品
FWFactoryTools_v5.1_RK3128_5.1	用于 RK3128 开发板的安卓 5.1 系统自定义
FWFactoryTools_v5.3_RK3288_5.1	用于 RK3288 开发板的安卓 5.1 系统自定义。
FWFactoryTools_v5.51_RK3128_RK3288_7.1	用于 RK3128、RK3288 开发板的安卓 7.1 系统自定义。

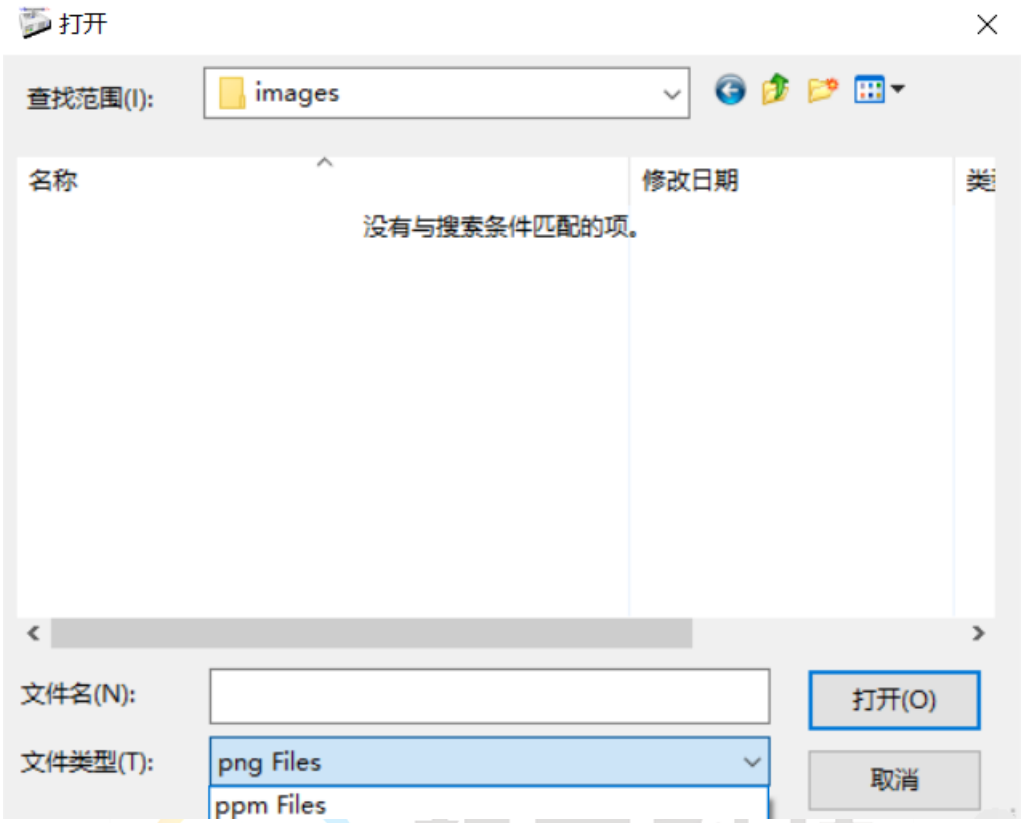


⚠ 注意：固件工厂工具依据安卓版本的不同具有不同的版本，各版本相互不通用。在下方链接中选择开发板对应的固件工厂工具下载，请至官网下载对应版本。

📌 提示：固件工厂工具下载地址：http://hyperlcd.com/downloads/

2.1.1 修改开机 LOGO

此功能仅 Android 5.1 系统支持，Android7.1 修改开机 Logo 请查看下方注意提示框。在固件工厂工具界面，点选相应固件后（固件资料请向您的销售垂询）请点击替换 Boot Logo 按钮，选择需要替换的开机 logo。您可以根据文件类型选择符合要求的图片， logo 格式支持 ppm, bmp, jpg,png。



注意：

1.内核配置为 Bmp Logo 的固件可支持多分辨率的替换，支持多种图片格式的替换。分辨率不做设定，但最好是以开发板分辨率为准，否则图片可能会因为分辨率而被压缩。

2.Android 7.1 因开机 logo 文件在内核中，暂不支持自行定义修改，请洽询与您对接的销售人员，我们会在源码中修改。

2.1.2 修改开机动画

Android 系统默认开机动画是显示 ANDROID 字样的图片，想要替换掉 ANDROID，需要修改开机动画，请参考下方提供的方法。

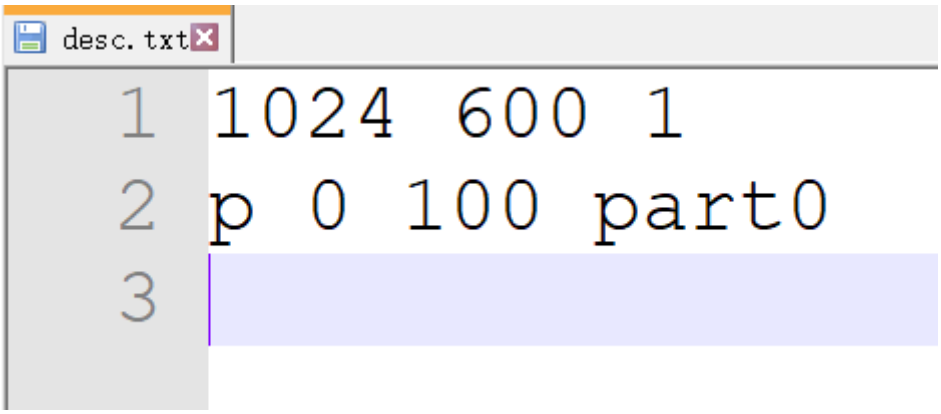
1. 制作开机动画

Android 默认开机动画是显示 ANDROID 字样的图片，想要替换掉 ANDROID，需要用到 bootanimation.zip。请将开机动画命名为 bootanimation.zip。该文件中包

含 part0 文件夹 和 desc.txt 两部分。

part0：该文件夹下为开机显示的动画图片，图片分辨率必须和产品分辨率一致。文件名以 0001.png~0099.png 格式或者 001.png~099.png，图片格式支持 PNG,JPG,文件名中不可添加特殊字符，如：*（），—_等；

desc.txt:此文件要在 Windows 或 Linux 系统下生成。文件参数如下（以分辨率 1024*600 的屏为例）：



Windows 下打开 desc.txt

```
1024 600 1
p 0 100 part0
```

在此可以复制参数（注意：每个参数之间都有一个空格，请勿删除空格，第三行空行也要复制）；

参数	说明
1024 600 1	1024 600 -- 屏幕分辨率（宽*高）； 1 -- 每秒播放的图片张数；
p 0 100 part0	p –表示 播放； 0 –表示循环播放；1 表示单次播放； 此处的 p 可替换为 c 100 – 图片播放延时时间； part0 – 存储动画图片文件的文件夹。

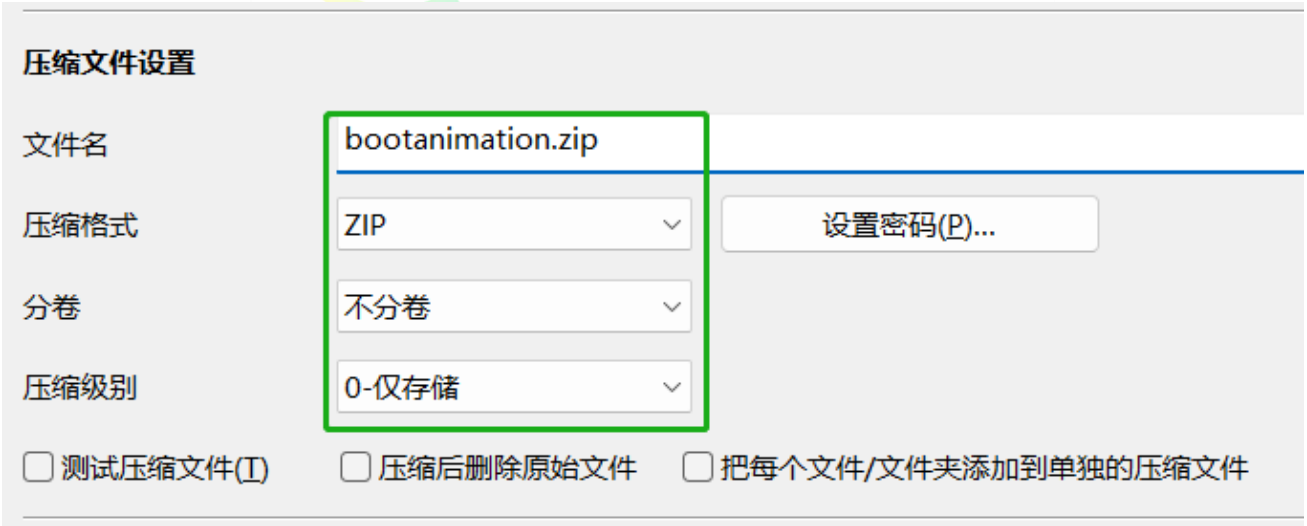
开机动画也可以添加多个文件夹通过设置参数以达到最完美的展示效果。示例：

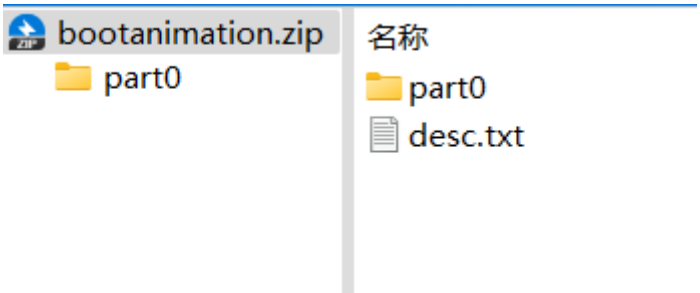
```
1024 600 5
p 1 0 part0
p 0 0 part1
```

上面展示了在 1024*600 分辨率的安卓模组上，以每秒 5 张图片的速度进行播放开机动画，首先播放 part0 中的图片，每张图片之间延迟 0ms，在 part0 图片播放完成后，播放 part1 文件夹中的图片，无限循环直到系统启动完成。

⚠ 注意：c 和 p 是两种动画播放方式,p 表示动画可能会中断,c 表示即便是 android 启动进程完成,也会继续等待动画播放结束后进入界面

在完成 part0 文件夹和 desc.txt 文件设置后，请按如下方式压缩文件夹 bootanimation (必须选择 zip 和存储的压缩方式为仅存储)。压缩完成后，请双击打开压缩文件查看是否有多余文件，若有则将其删掉，否则开机没有动画。

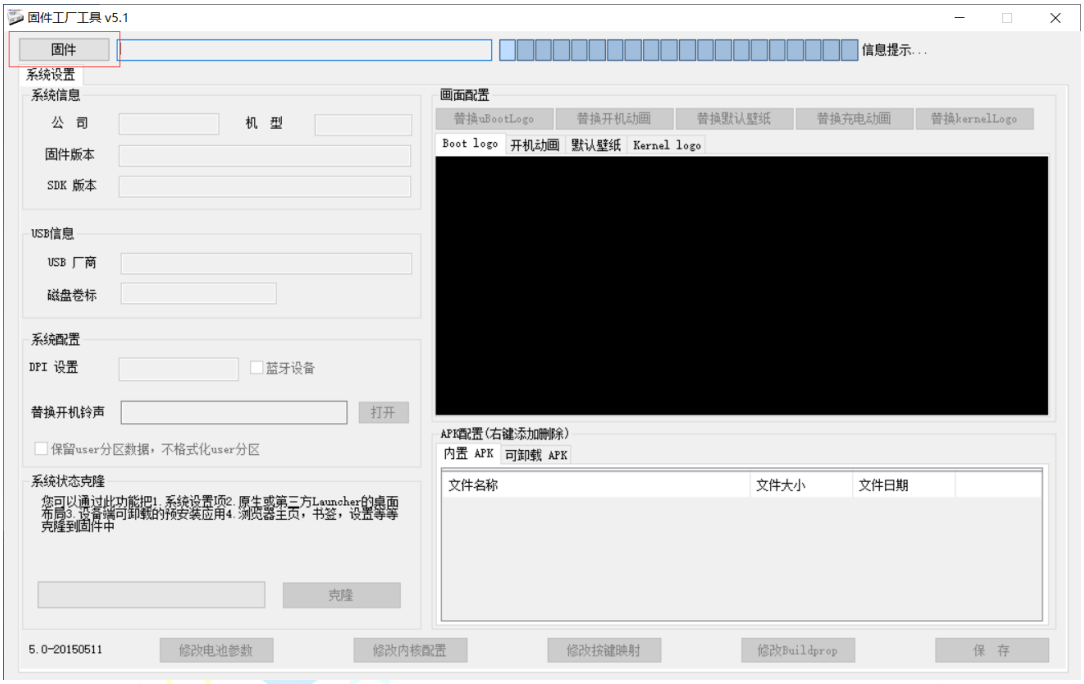




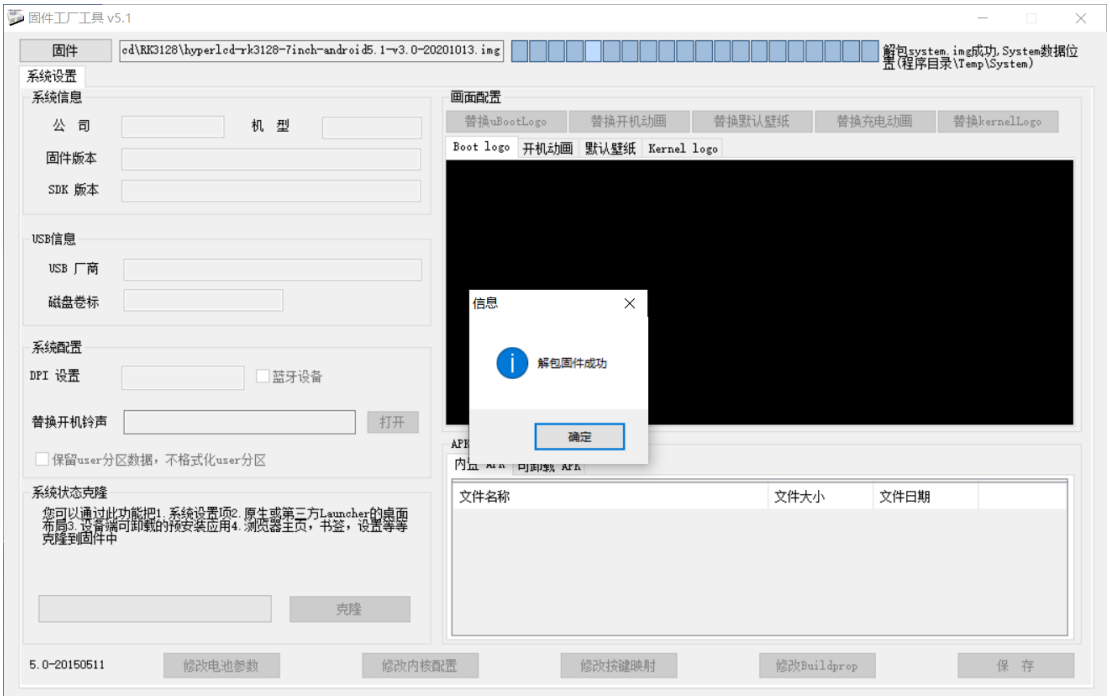
压缩包压缩后不可以有多余的文件，如上图所示，压缩包打开后应只有 desc.txt 与 part0 文件夹。

2. 加载开机动画

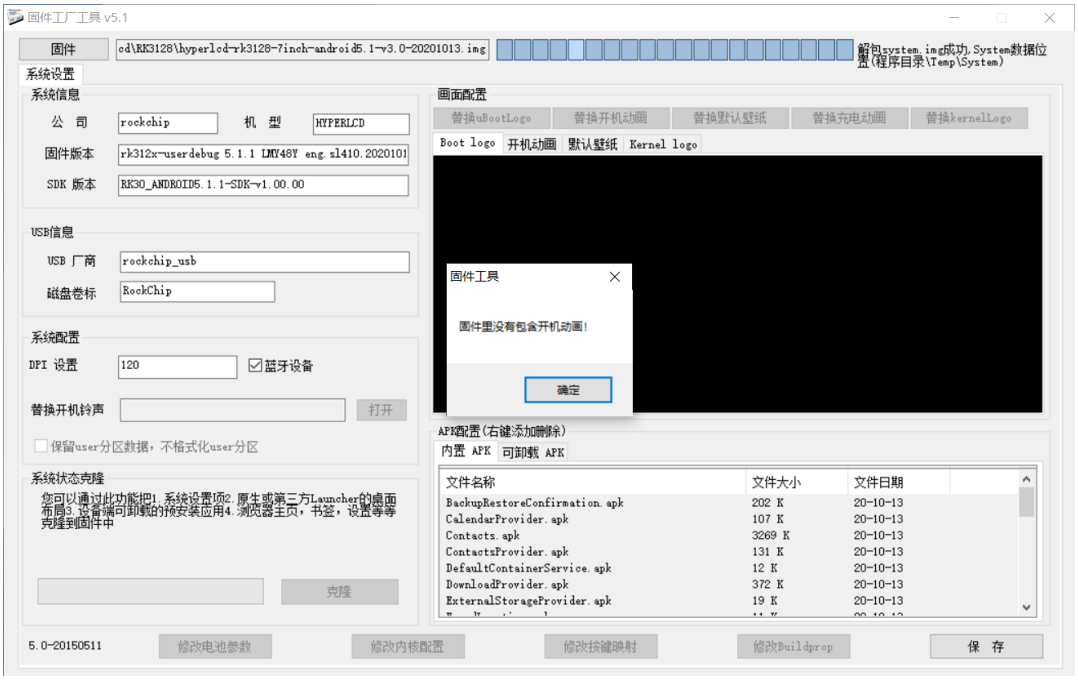
点击固件，打开需要修改的固件，选择后软件自动解包。



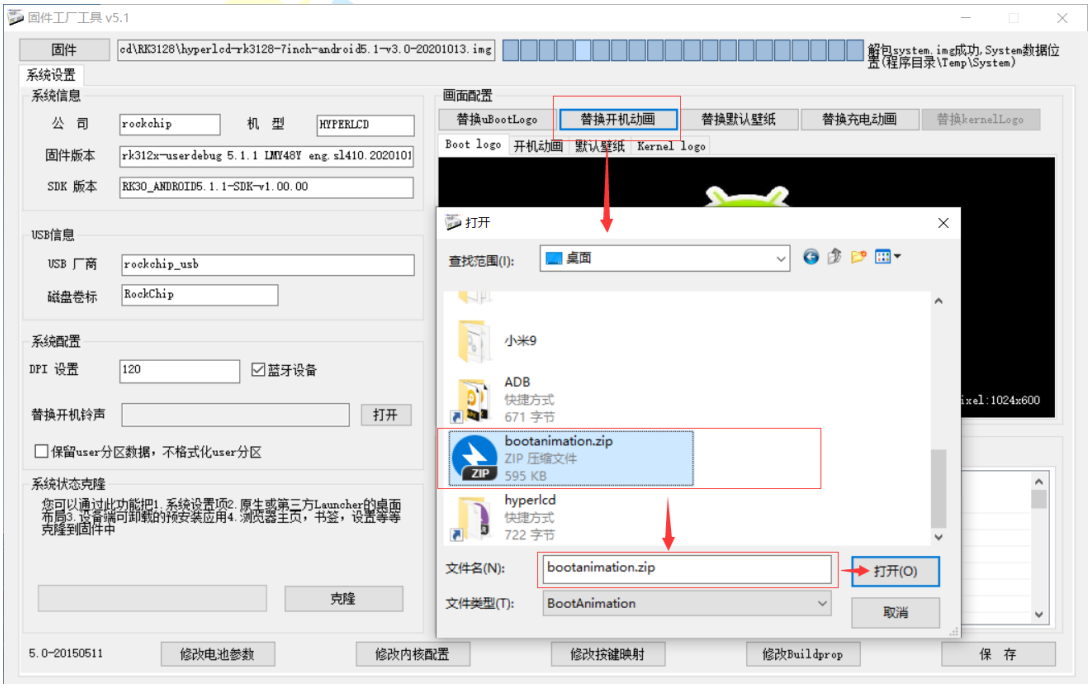
提示：默认 IMG 固件，请向您的销售人员垂询。



解包固件，固件里没有包含开机动画，请点击确定。

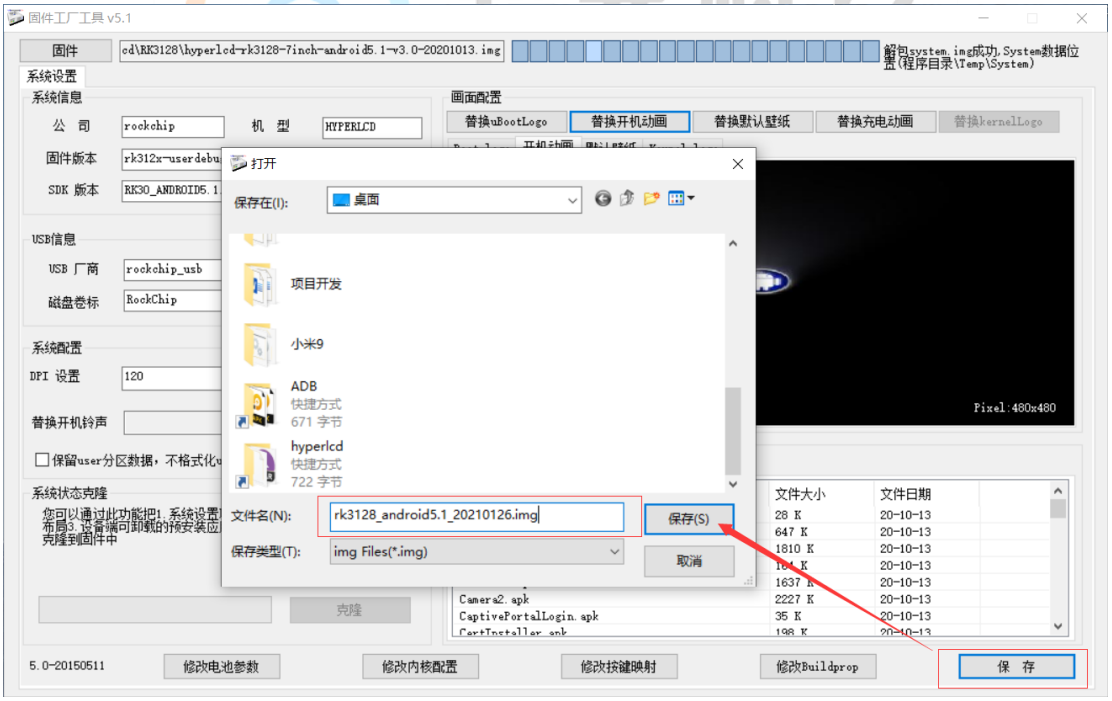


点击替换开机动画按钮，制作好的开机动画导入。



预览开机动画的效果。

修改完成后点击保存，请保存 img 文件。



2.1.3 添加开机音乐

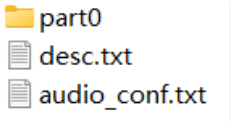
开机时如果需要播放开机音乐，请按照下方方法进行操作。

· Android5.1 添加开机音乐

按照 2.1.2 章节制作开机动画，并新建文件 audio_conf.txt ，txt 文档内容如下：

```
card=0
device=0
period_size=1024
period_count=4

```

共 5 行（包含含一行空行）



添加 WAV 音频文件到 part0 中，WAV 音频命名为 audio.wav

按照 2.1.2 章节进行打包系统。

· Android7.1 添加开机音乐

按照 2.1.2 章节制作开机动画，添加 WAV 音频文件到 part0 中，WAV 音频命名为 audio.wav

2.1.4 烧录系统固件程序

超显安卓模组，在出厂前均会预装相应版本的安卓系统。如您有其他特殊设置，需二次烧录安卓系统文件（IMG 文件）。请按如下步骤进行固件程序烧录。

第一步：打开安卓固件烧录工具

开发板对应安卓固件烧录工具请参考下表。

固件烧录程序	适用产品
AndroidTool_V2.54	用于 RK3128/3188/3288 开发板的 img 文件烧录，支持 Android、linux 的系统烧录.

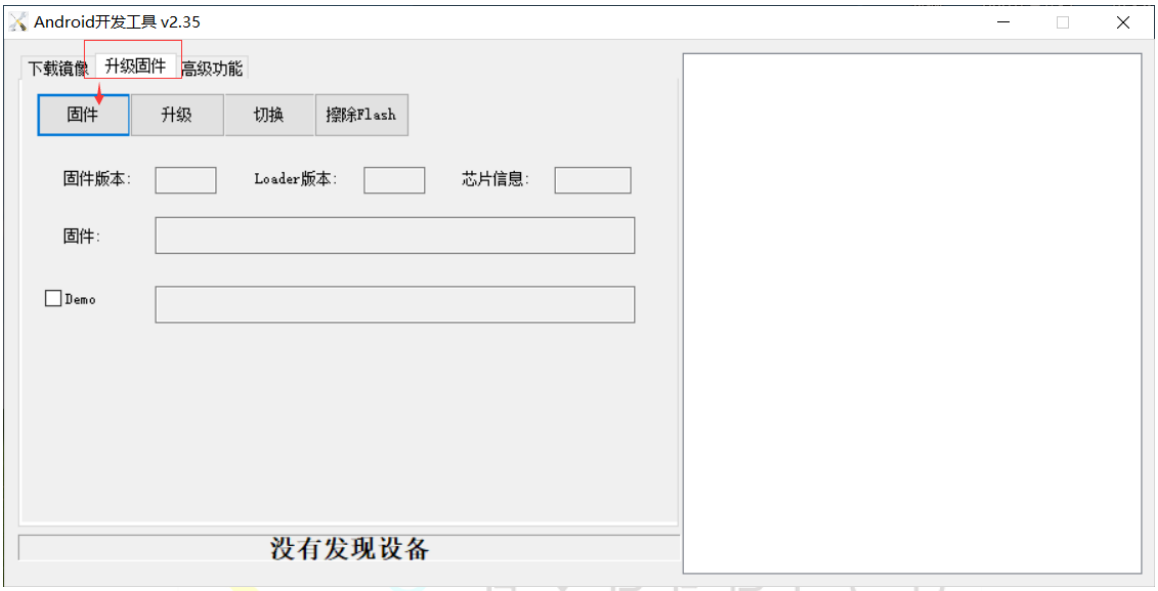
⚠ 注意：烧录固件前必须先安装驱动，驱动下载地址:

http://hyperlcd.com/download/driverassitant_v4-5/

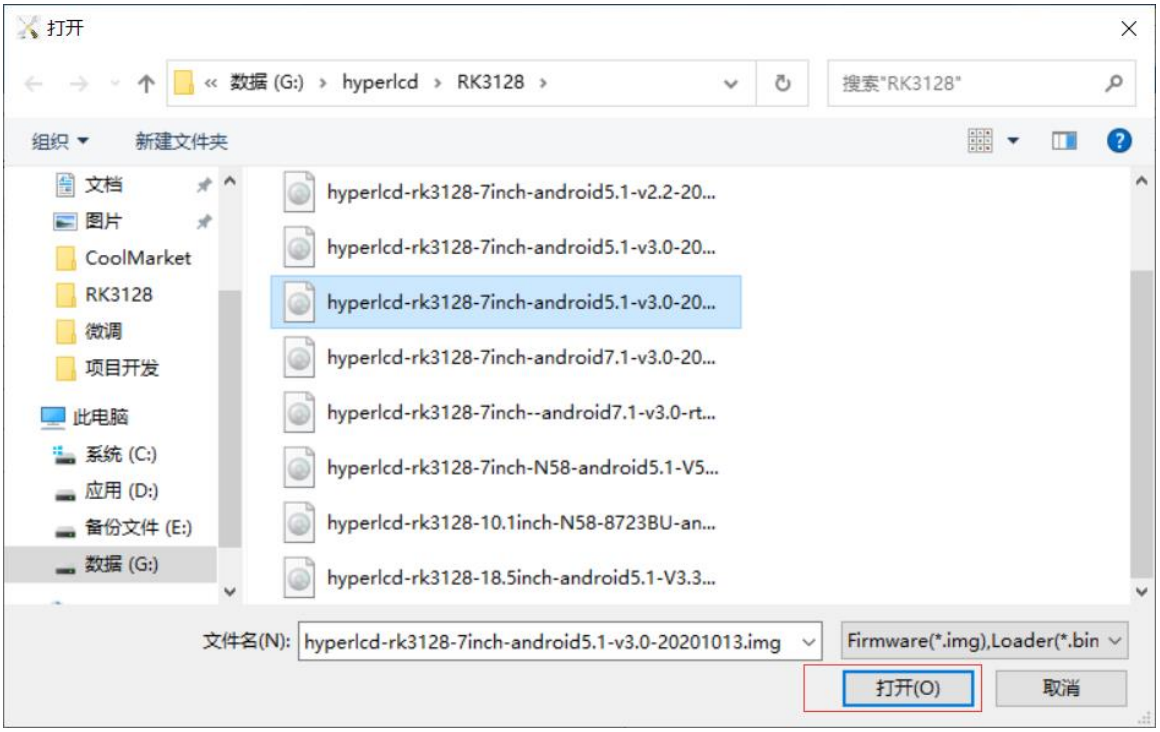
Android Tool 工具下载地址：

http://hyperlcd.com/downloads/

第二步：点击升级固件 ，进入固件选择界面



第三步：连接设备，选择需要烧录的固件



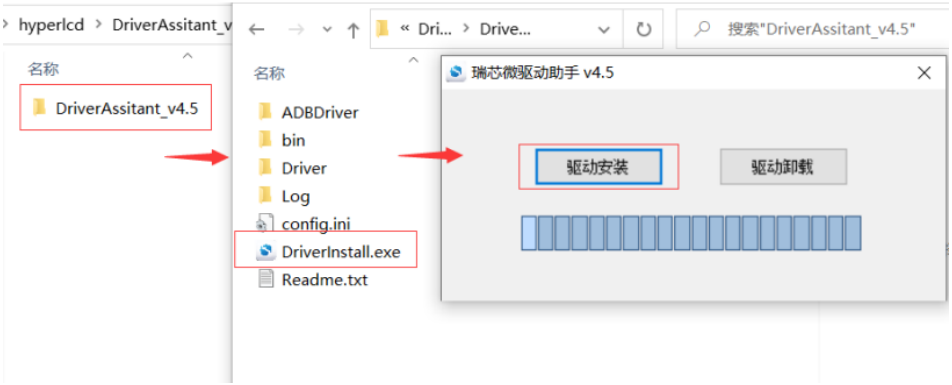
第四步：进入系统烧录模式

⚠ 注意：请参考本文档 1.3 章节进入系统烧录模式。若无法解决您遇到的问题，请与您对接的销售人员寻求技术支持。

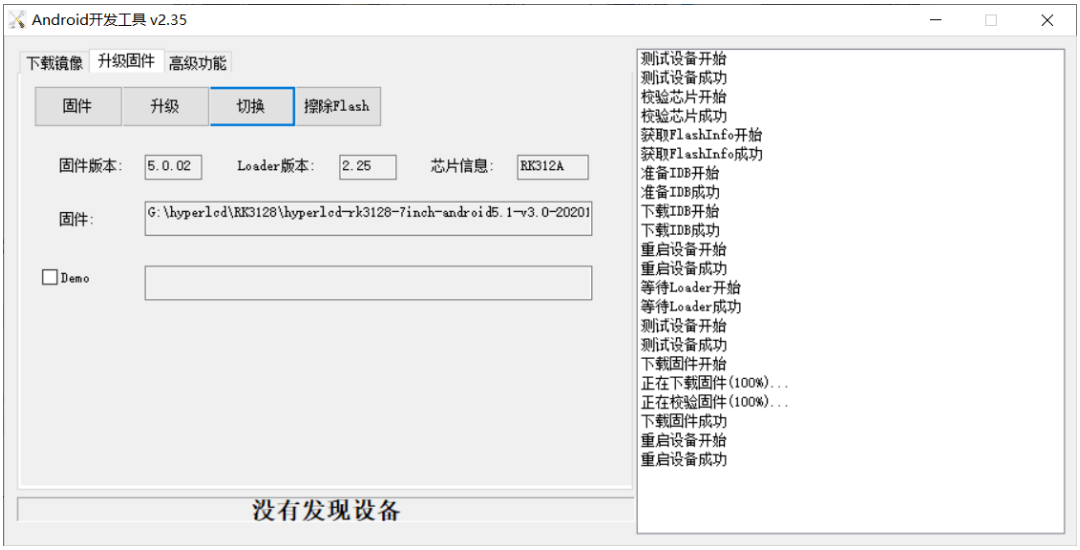
第五步：连接后软件显示（发现一个 LOADER 设备）表示已经连接识别，此时放开镊子的短接。

⚠ 注意：此处如无法正常加载固件，先安装驱动，驱动下载地址:

http://hyperlcd.com/download/driverassitant_v4-5/



第六步：点击升级，软件开始自动加载并刷 img



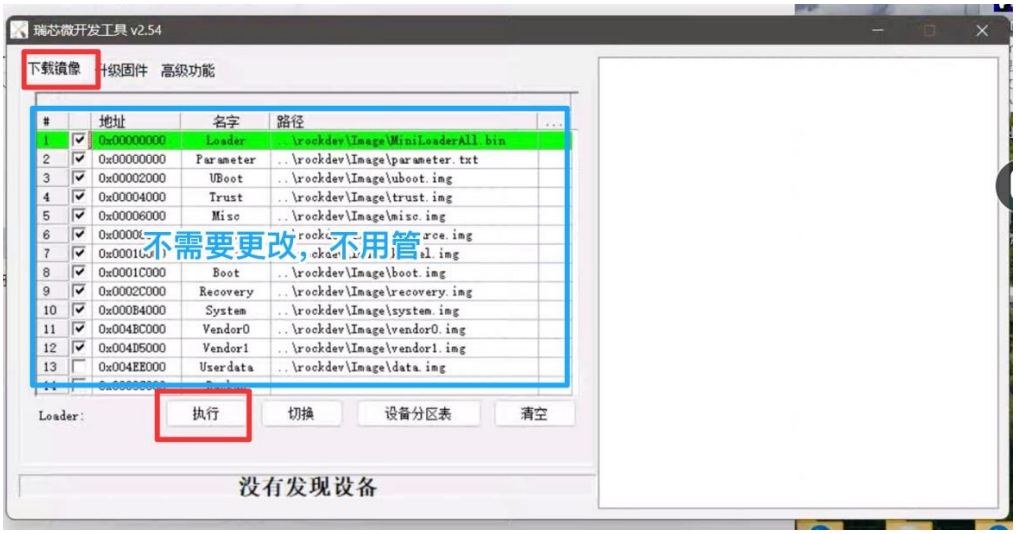
右侧提示下载固件成功，重启设备成功。至此，您的设备 img 系统文件更新成功。

⚠ 注意：如果提示烧录失败，请检查固件工厂工具，烧录工具，目录以字母、不带空格等特殊符号，文件夹出现特殊符号或中文可能会导致烧录失败。

2.1.5 烧录补丁包

超显安卓模组，支持烧录指定分区，这一般是由系统工程师单独提供的补丁。用于更新系统功能、修复问题。一般是以压缩包形式展现。

右键解压压缩包后打开 Android Tool，将安卓模组进入烧录模式，点击下载固件 -> 执行，等待重启完成此时补丁烧录完成。



第三章 软件开发环境搭建与配置

超显科技提供完整的硬件配置和安卓开源系统环境。根据不同的接口定义，超显提供客制化的硬件接口设计服务（详情请接洽您的销售人员）。软件开发部分，您可以提供开发需求与技术要求给我司，请接洽您的销售人员协商软件开发的费用，您也可以自行完成或委外第三方协助完成。

本章节提供安卓开发环境的基础搭建说明。这些开发环境搭建说明能够协助您快速上手安卓屏的使用和测试。我司提供用户端示例软件 APK 的源码支持或相关技术问题解决。详情请接洽您的技术人员。

⚠ 注意：在产品开发过程中，如您的软件开发设置含相关系统底层设置，请接洽我司技术人员更改安卓系统 IMG 文件。

📌 提示：配置 IDE 是 Android 程序开发的重要部分，优秀的 IDE 可以提高程序开发效率。

3.1 Java 环境搭建与安装

3.1.1 工具清单

- JDK 工具包

  JDK 是 Java 语言的软件开发工具包，它包含了 Java 的运行环境、工具集合、基础类库等内容。

- Android SDK

  Android SDK 是谷歌提供的 Android 开发工具包。在开发 Android 程序时，需要引入该工具包来使用 Android 相关的 API。

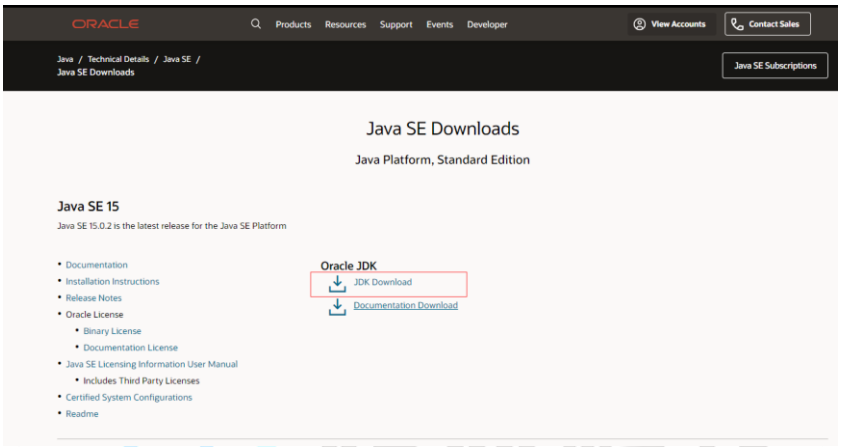
- Android Studio

  Android Studio 是谷歌推出一款官方的 IDE 工具，由于不再是以插件的形式存在，Android Studio 在开发 Android 程序时比 Eclipse 更加强大和方便。

提示：您可以选择不同的 IDE 环境，Android Studio 为我司推荐安卓开发环境。如您使用其他安卓开发环境，可跳过章节.

3.1.2 下载 JAVA SDK

请下载 Java 开发工具包 JDK SE ,如网络加载缓慢，请寻求代理 proxy 服务器或 VPN 支持。下载地址：http://www.oracle.com/technetwork/java/javase/downloads/index.html



注意：请根据您的计算机操作系统选择对应的软件版本。本文档我们以 Windows 64 位系统为例。

请根据 JDK 安装提示完成您的下载。在下载过程中同时会安装 JRE。安装过程中可以自定义安装目录等信息，例如我们选择安装目录为 C:\Program Files\Java\jdk-14\

3.1.3 配置环境变量

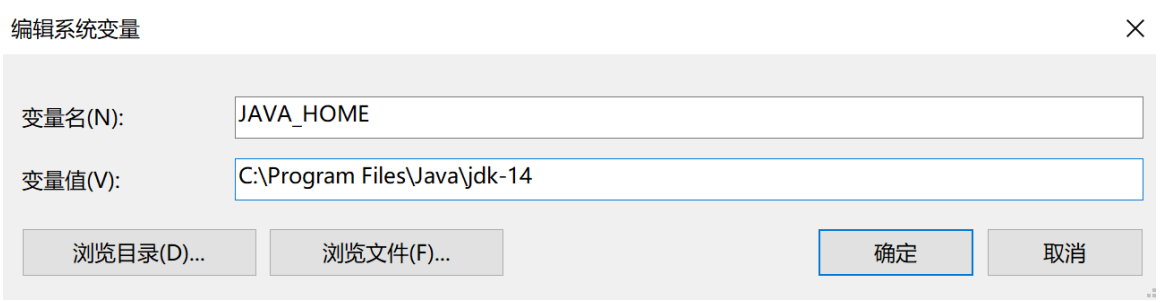
1. 安装完成后，在 Windows 当中右击开始菜单，选择系统 ，下拉选择系统信息 ，点击左栏高级系统设置 ；

2. 选择高级选项卡，点击环境变量；

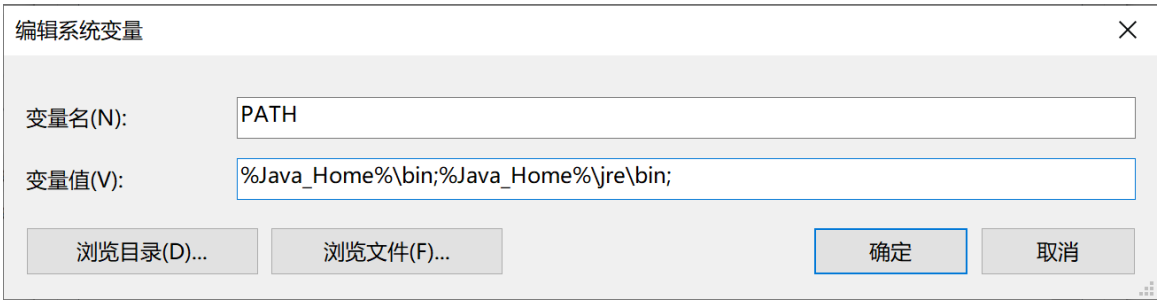
3. 请在系统变量中设置 3 项属性： JAVA_HOME ， PATH 以及 CLASSPATH 。若已存在则点击 编辑 ，不存在则点击 新建 。

变量名	变量值
JAVA_HOME	C:\Program Files \Java\jdk-14
PATH	%Java_Home%\bin;%Java_Home%\jre\bin;
CLASSPATH	.;%Java_Home%\bin;%Java_Home%\lib\dt.jar;%Java_Home%\lib\tools.jar

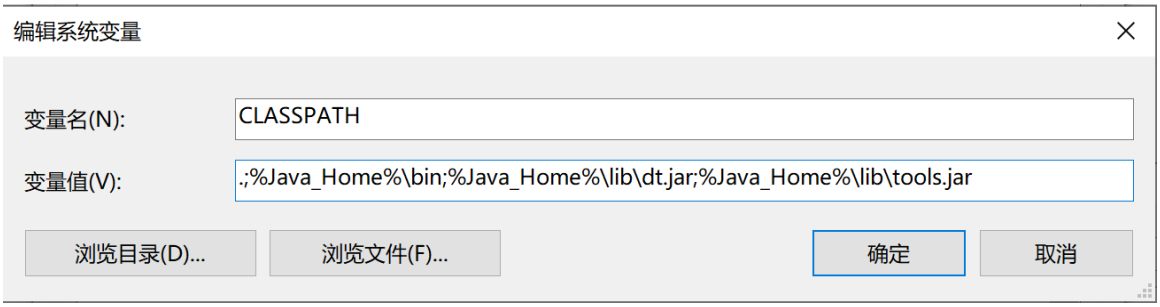
- JAVA_HOME 设置



- PATH 设置



- CLASSPATH 设置



3.1.4 测试 JDK 安装状态

1. 开始 > 运行 ，键入 cmd ；
2. 键入命令： java -version 命令，如出现以下类似信息，说明环境变量配置成功；

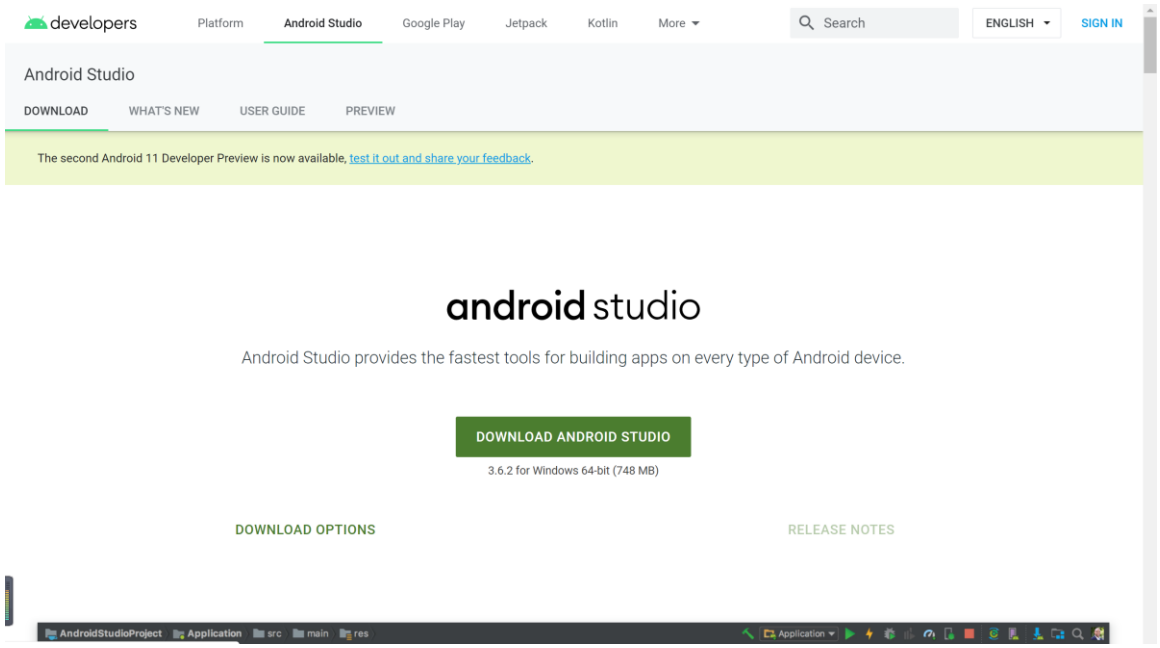
```
java version "14.0.1" 2020-04-14
Java(TM) SE Runtime Environment (build 14.0.1+7)
Java HotSpot(TM) 64-Bit Server VM (build 14.0.1+7, mixed mode, sharing)
```

3.2 Android Studio 的安装和环境搭建

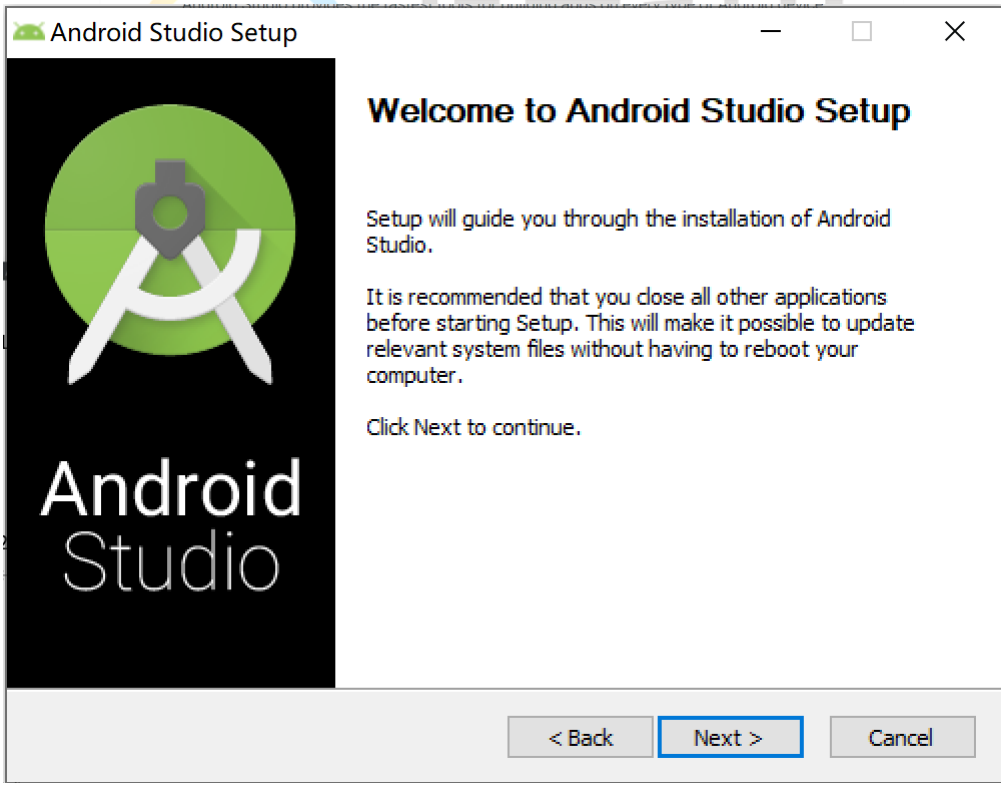
3.2.1 安装 Android Studio

下载地址:https://developer.android.google.cn/studio/，获取 Android Studio 最新版本。

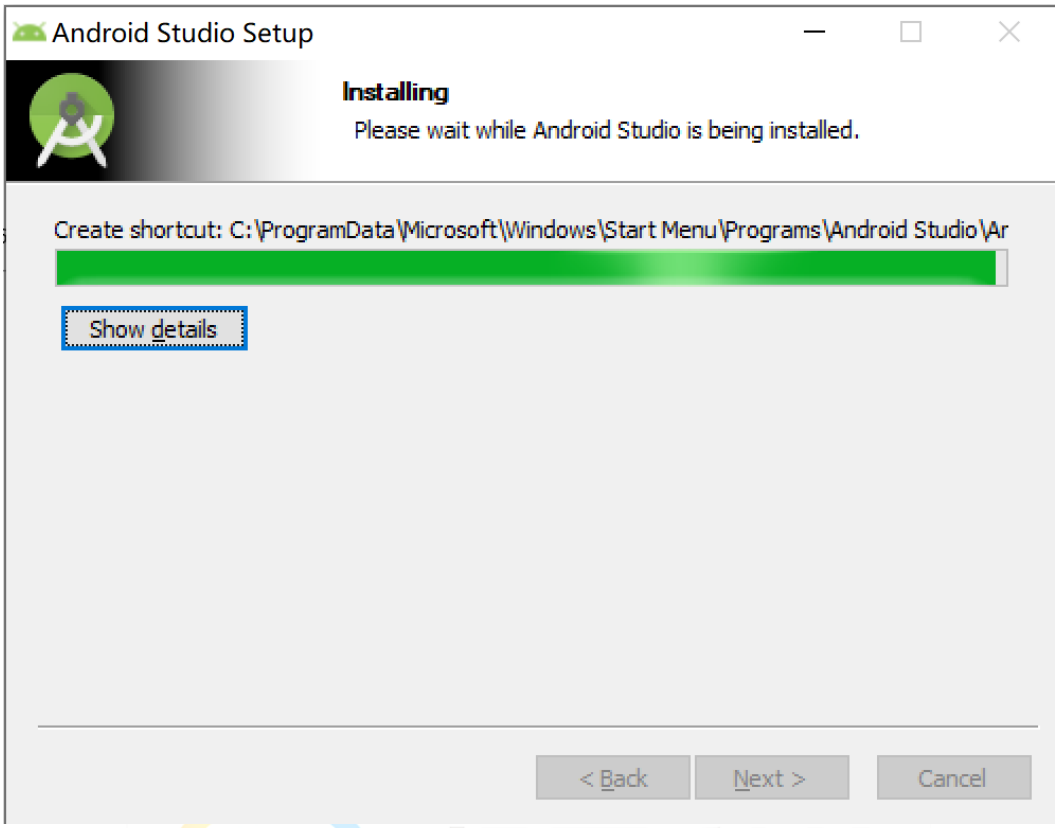
⚠ 注意：安卓版本的发布，取决于 Google Developers 的更新。您不必随时更新产品版本，只需确保符合开发需要即可。 Android studio 产品的更新和发布此文档不做另行通知 。



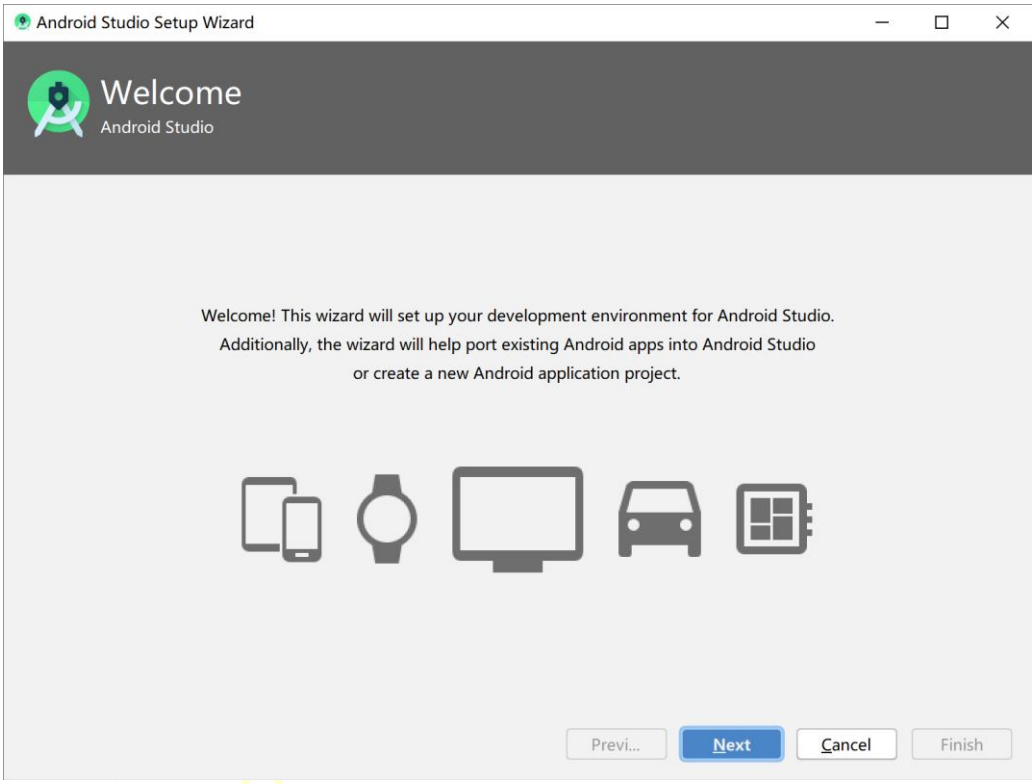
下载完成后双击 SETUP 安装包进行安装，然后点击 Next ，如下图所示：



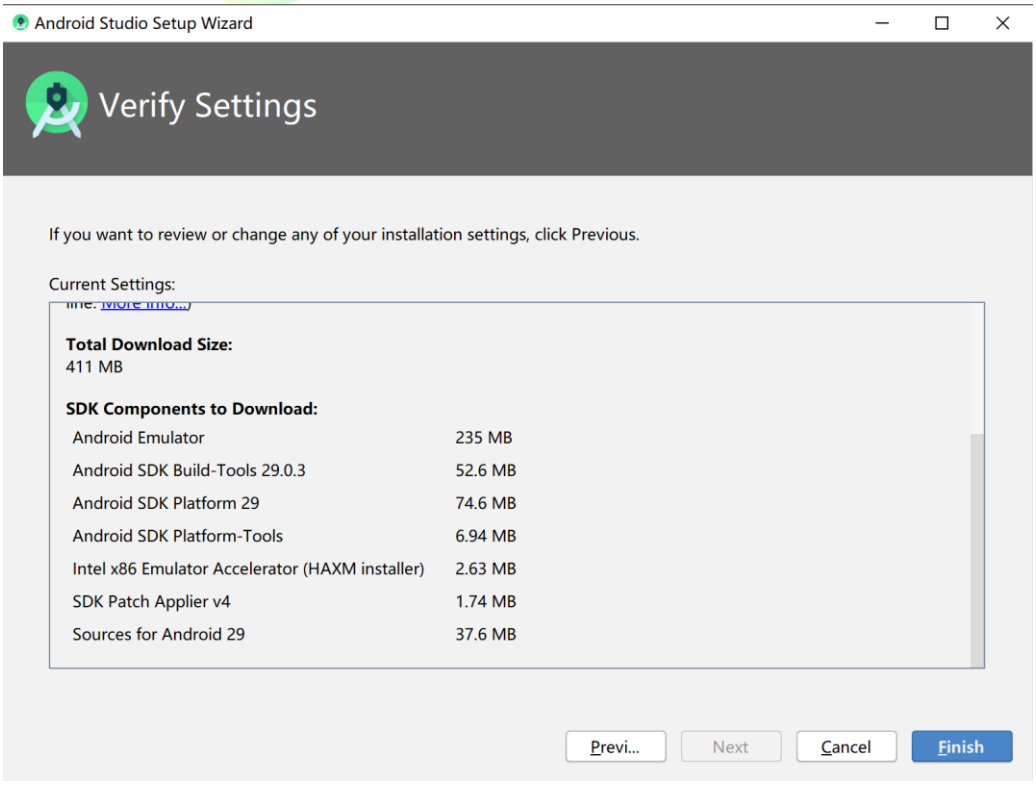
点击“ I Agree ”按钮，如下图所示；
选择 Android Studio 安装位置和 Android SDK 安装位置，然后点击 Next ；
点击“ Install ”按钮，安装过程如下图所示：

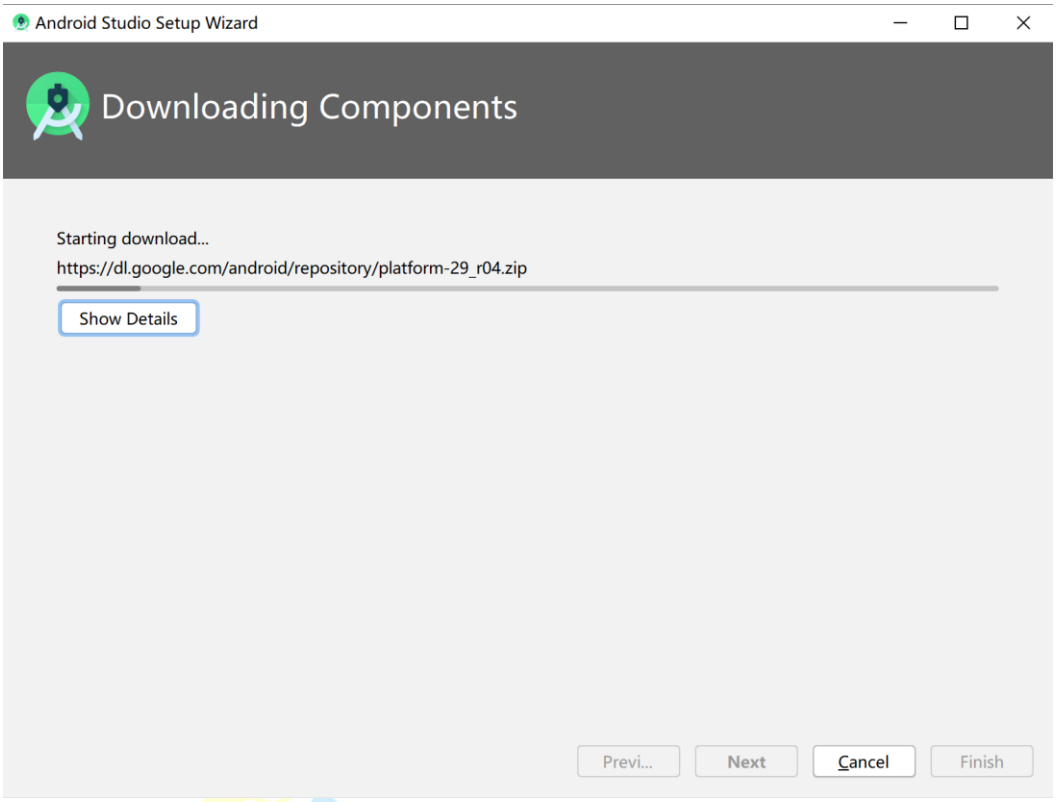


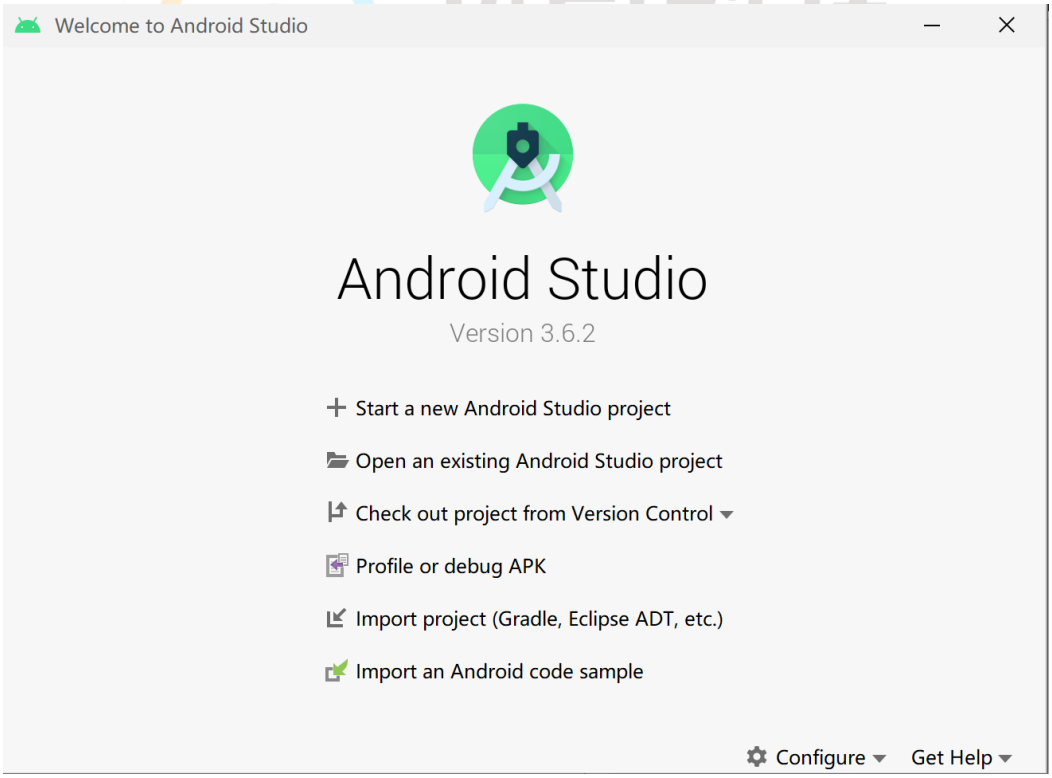
直至安装完成，如下图所示：



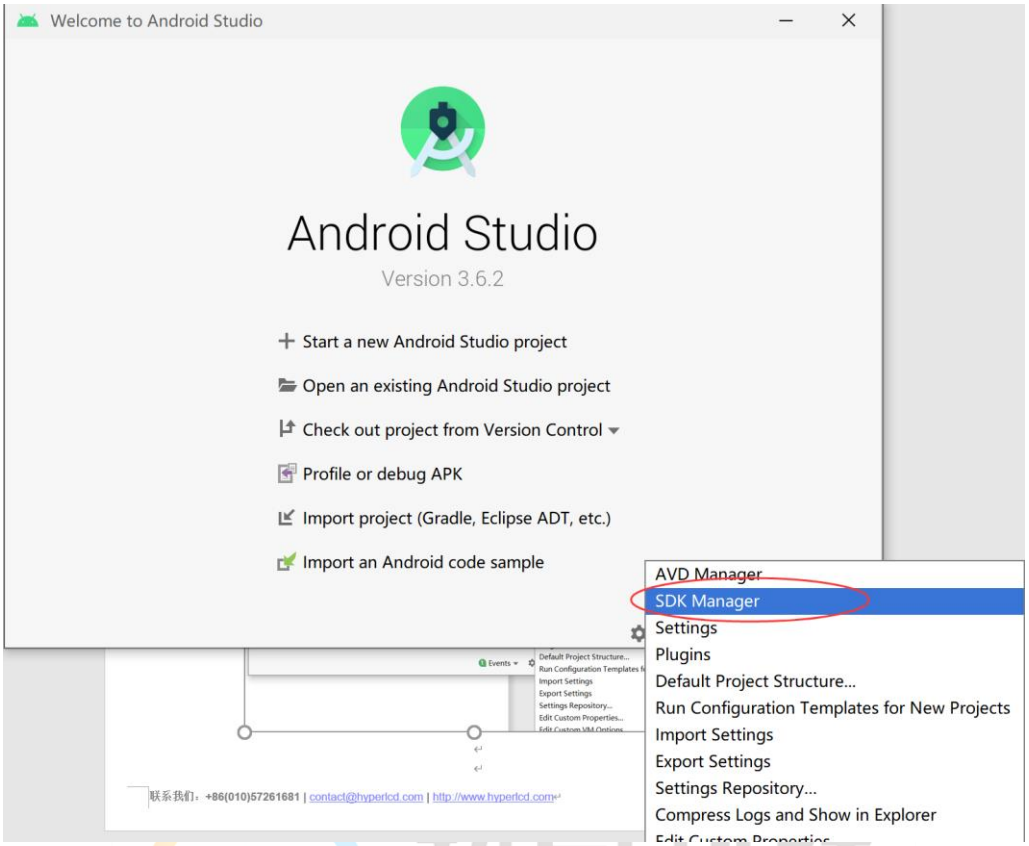
Android Studio 的相关 SDK 工具，可在进入 Android Studio 欢迎界面后继续完成相关配置和下载。这些配置包含但是不仅限于 Android SDK 相关资源的下载和使用。



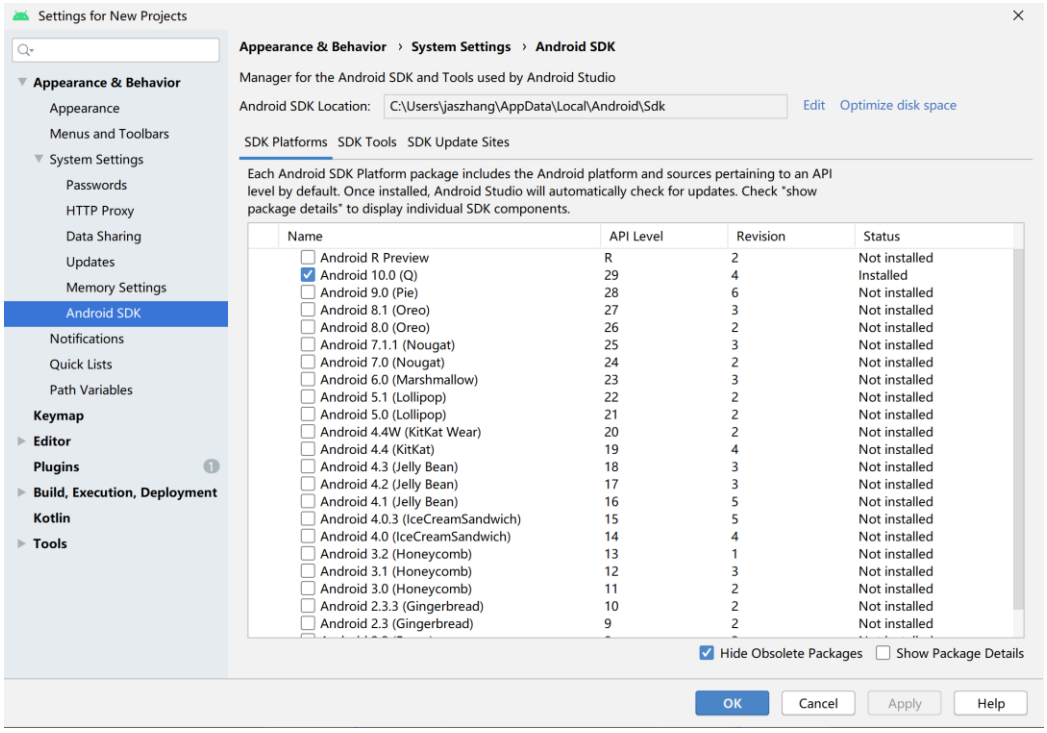


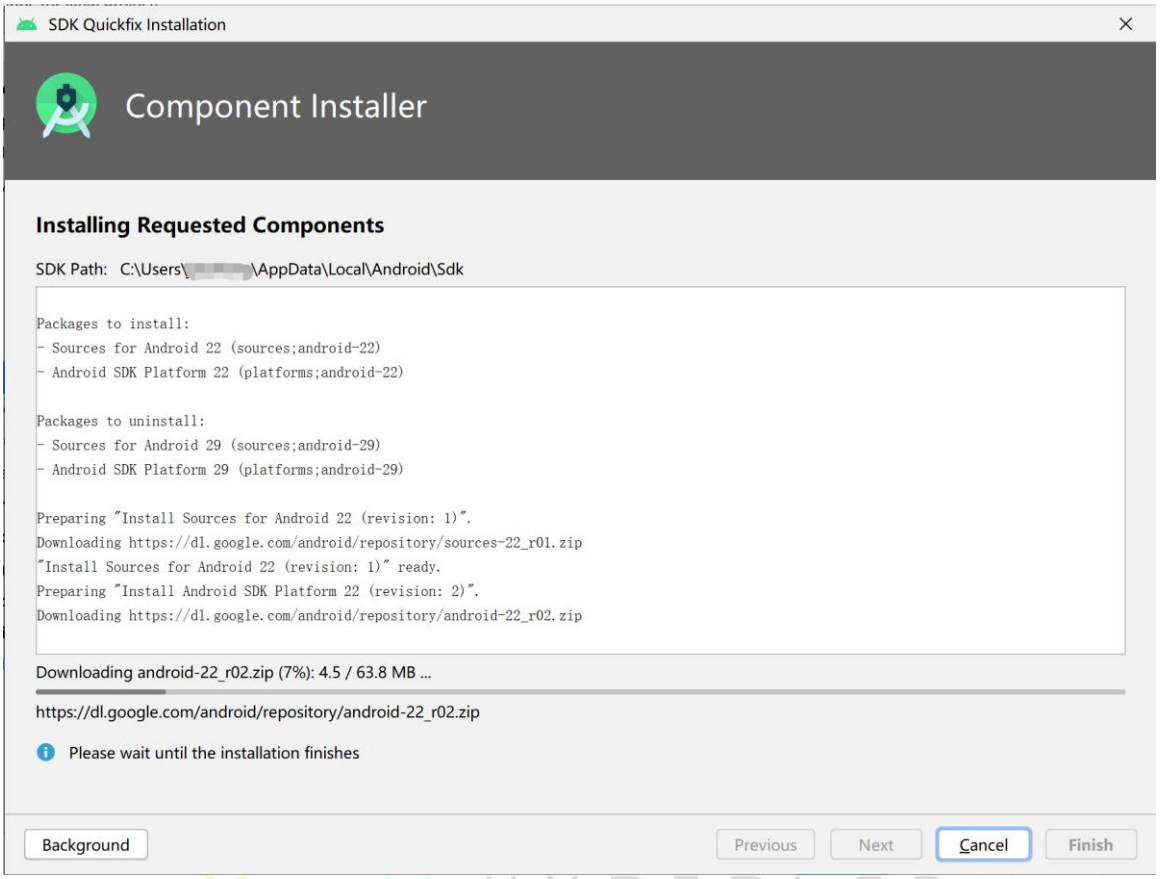


选择 SDK Manager ，如下图所示：



在 System Settings 中，点击 Android SDK ，如下图所示：



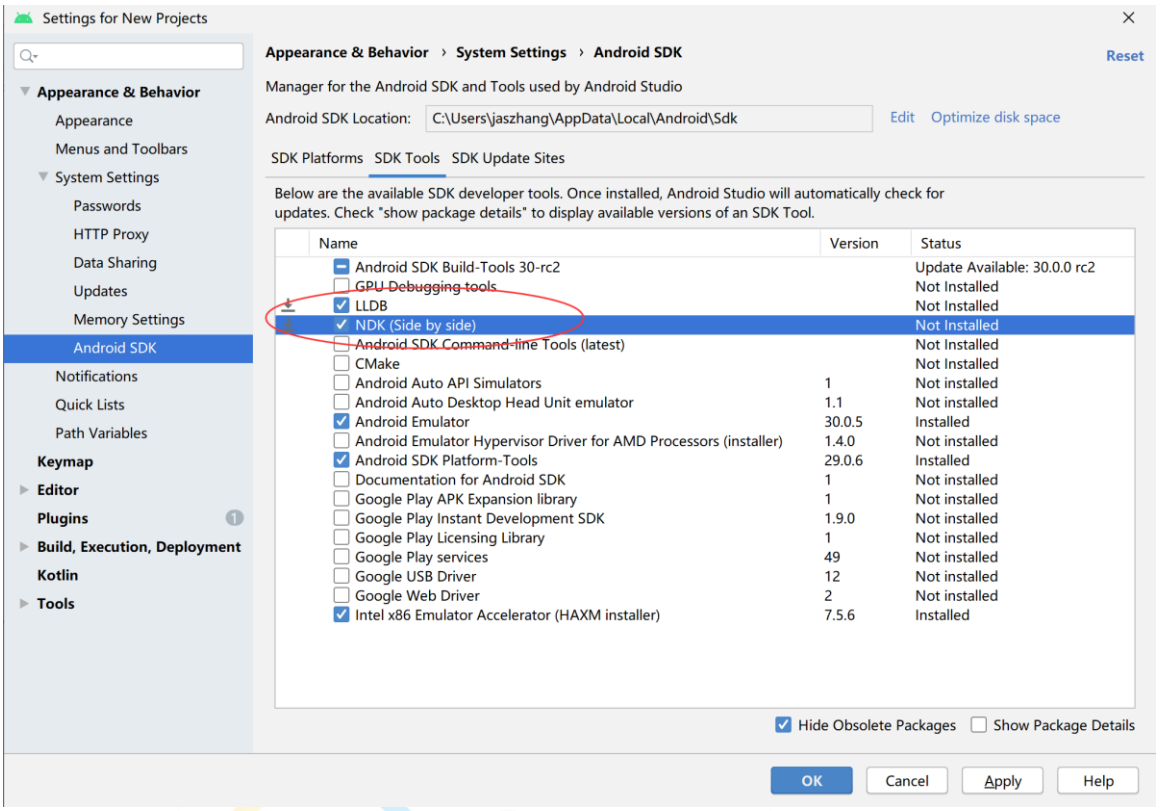


3.2.2 配置 NDK 开发环境

NDK，Native Development Kit，原生开发工具包是一组可以让您在 Android 应用中利用 C 和 C++代码的工具，可用以从您自己的源代码构建，或者利用现有的预构建库。

1.安装 NDK

打开 Tools->Android->SDK Manager->SDK Tools 选中 LLDB 和 NDK ，点击确认，软件会自动安装 NDK。



2. 将 so 库导入 libs 目录下（project 结构）

so 文件是 Linux 下的程序函数库，即编译好的可以供其他程序使用的代码和数据。Android 应用支持的 ABI 取决于 APK 中位于 lib/ABI 目录中的.so 文件

第四章 安卓开发实例

4.1 串口开发实例

串口通信（Serial Communications），是指外设和计算机间通过数据信号线、地线、控制线等，按位进行传输数据的一种通讯方式。串口按位（bit）发送和接收字节。尽管比按字节 (byte) 的并行通信慢，但是串口可以在使用一根线发送数据的同时用另一根线接收数据。它很简单并且能够实现远距离通信。

串口通信最重要的参数是波特率、数据位、停止位和奇偶校验。对于进行通信的端口，这些参数必须匹配。

我们提供串口开发示例程序，您可以在项目开发前，参考我司相关示例程序源码，从而顺利完成您的应用软件应用端串口部分的通讯设计。

下载链接：http://hyperlcd.com/download/串口实例程序 v5-1-参考 1/

串口实例程序参考 1_v5.1，是 ComAssistant_signed.1.1 串口示例程序，提供串口调试 apk 及源码；(本项目兼容安卓 5.0-安卓 7.1)

⚠ 注意：串口通信的源码设计，由于产品的多样性程序在实际应用中可能会所有不同。我们所提供的程序参考实例，仅供您学习参考代码原理之用。产品的串口可用性，视您的下控机在实际使用过程中可能有所调整，相关请咨询研发人员。

4.1.1 串口通讯步骤

实现步骤:

1. 串口初始化——创建串口实例、设置串口参数（定义路径和波特率）;
2. 获取输入流——读取串口数据;
3. 获取输出流——向串口发送数据;
4. 数据处理与显示;
5. 关闭串口;

4.1.2 串口 Demo 代码说明

以串口实例程序参考 2_v5.1 为例，我司提供一个串口示例程序的代码参考。相关原始文件请至如下地址下载：http://hyperlcd.com/download/串口实例程序参考 2_v5-1/

第一步：导入 SO

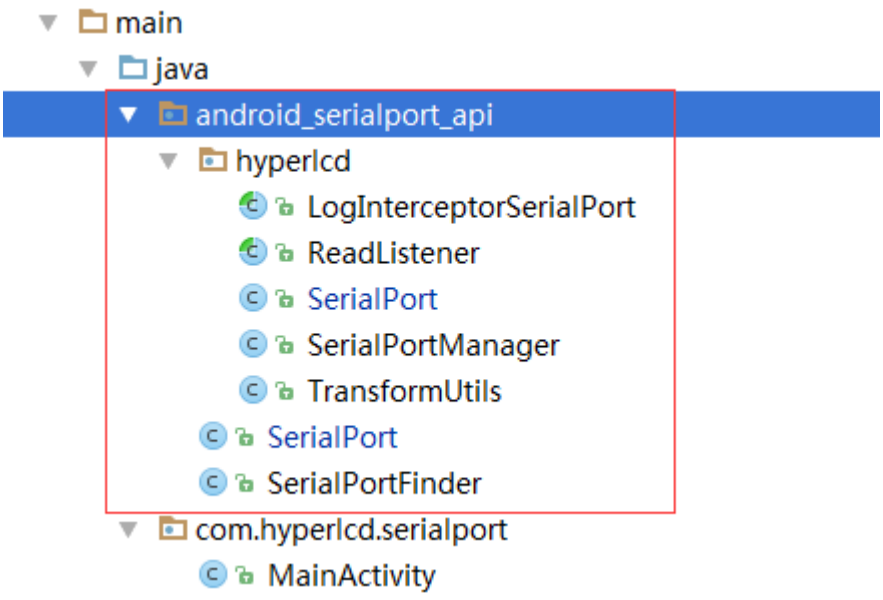
1. 拷贝相应架构下的 so 文件到该项目的 jniLibs 文件夹 (安卓 5.1-安卓 7.0 系统操作步骤相同)

2. 在 build.gradle 的 Android 标签下添加配置

```
sourceSets {
    main {
        jni.srcDirs = []
        jniLibs.srcDirs = ['src/main/jniLibs', 'libs']
    }
}
```

第二步：拷贝文件

拷贝 android_sericalport_api 所有文件到当前处理项目的 java 文件夹下，注意路径不可更换。



第三步：使用串口

1. 定义全局变量

```
private BaseReader baseReader; //读取回调
private SerialPortManager spManager; //串口控制

private String checkPort = "dev/ttyS1"; //这里端口号按照需要测试的改
private boolean isAscii = true; //true 时以 ascii 形式发送
```

2. 初始化操作 SerialPortManager

```
spManager = SerialPortManager.getInstances().setLogInterceptor(new LogInterceptorSerialPort() {
    @Override
    public void log(@SerialPortManager.Type final String type, final String port, final boolean isAscii, final
String log) {
        Log.d("SerialPortLog", new StringBuffer()
                .append("串口号：").append(port)
                .append("\n 数据格式：").append(isAscii ? "ascii" : "hexString")
                .append("\n 操作类型：").append(type)
                .append("操作消息：").append(log).toString());
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                logTV.append(new StringBuffer()
                        .append(" ").append(port)
                        .append(" ").append(isAscii ? "ascii" : "hexString")
                        .append(" ").append(type)
                        .append("： ").append(log)
                        .append("\n").toString());
            }
        });
    }
});
baseReader = new BaseReader() {
    @Override
    protected void onParse(final String port, final boolean isAscii, final String read) {
        Log.d("SerialPortRead", new StringBuffer()
                .append(port).append("/").append(isAscii ? "ascii" : "hex")
                .append(" read：").append(read).append("\n").toString());
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                readTV.append(new StringBuffer()
                        .append(port).append("/").append(isAscii ? "ascii" : "hex")
```

```
                .append(" read： ").append(read).append("\n").toString());
            }
        });
    }
};
```

3. 打开串口

```
//checkport 串口号
//isAscii 数据格式
//baseReader 读取回调
spManager.startSerialPort(checkPort, isAscii, baseReader);
```

4. 修改读数据格式

```
//currentPort 串口
//isAscii 数据格式
spManager.setReadCode(currentPort, isAscii);
```

5. 发送数据（需先打开串口）

```
spManager.send(currentPort,“send”);
// 向串口 currentPort 写数据，,数据内容为：send
```

6. 关闭串口

```
spManager.stopSerialPort(currentPort);
// 关闭串口 currentPort
```

7. 销毁 SerialPortManager，销毁后如需再次使用串口，需要重新执行初始化操作

```
spManager.destroy();
```

第四步：串口运行注意事项

1. 运行时出现 java.lang.UnsatisfiedLinkError: … couldn't find "libserial_port.so"错误

出现此问题代表"libserial_port.so"没有找到，请检查 jniLibs 文件夹名称是否正确无误，请检查 jniLibs 文件夹位置是否正确，在 Project 目录结构下，jniLibs 文件夹应存在 main 文件夹中，与 java 文件夹所在目录一致。

2. 访问某些串口安卓屏卡死

请检查软件是否访问了/dev/ttyS2 口，此串口作为单独调试系统使用。用户无法使用，请在代码中屏蔽此串口。RK3288 开发板 ttyS0 被无线蓝牙二合一模块占用，用户不可用。

3. 串口发送数据接收不成功

使用串口发送数据时，出现了提示发送成功，但是接收端没有接收到信息的情况，请优先检查您的连接线材可以正常收发数据，可以尝试换一条连接线，确认连接线无误。检查代码，优先查看接收端与发送端的串口是否一致，波特率（两端波特率要一致）。请查看 Logcat 排除代码无异常。

4. ANR 异常

常见的有如下两种情况会产生 ANR：

输入事件（例如按键或屏幕轻触事件等）在 5 秒内没有响应；

BroadcastReceiver 在 10 秒内没有执行完成。

出现 ANR 异常时，请定位到出现此情况的代码，进行修改。

4.2 开机自启动介绍

开机自启动一般有两种方式：

1. 接收开机广播后启动，开机会进入系统桌面后启动 APK

2. 设置自定义 APK 为 Launcher ，开机跳过系统桌面直接进入 APK。

选择一种即可，推荐第二种。

4.2.1 接收开机广播后启动

当 Android 启动时，会发出一个系统广播，内容为 ACTION_BOOT_COMPLETED 。

它的字符串常量表示为 android.intent.action.BOOT_COMPLETED 。只要在程序中“捕捉”到这个消息，再启动之即可。所以实现的手段就是实现一个 Broadcast Receiver。

第一步：自定义广播类 BootReceiver

```
public class BootReceiver extends BroadcastReceiver {
    private SharedPreferences pref;
    private boolean autoStarts = false;
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {
            // boot
            pref = context.getSharedPreferences("BOOT_COMPLETED", Context.MODE_PRIVATE);
            autoStarts = pref.getBoolean("Autostarts",false);
            if (autoStarts){
                Log.e("Autostarts","开机自动启动");
```

```
                Intent intent2 = new Intent(context, MainActivity.class);
                intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(intent2);
            }else {
                Log.e("Autostarts","开机不自动启动");
            }

        }
        if (intent.getAction().equals("android.intent.action.ACTION_SHUTDOWN")){
            Log.e("shutdown","关机了");
        }
    }
}
```

第二步：清单文件配置

在 AndroidManifest.xml 中 Application 节点内，添加自定义的 receiver

```
<receiver
    android:name=".BootReceiver"
    tools:ignore="Instantiatable">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.ACTION_SHUTDOWN"/>
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</receiver>
```

第三步：添加权限

在 AndroidManifest.xml 中 manifest 节点内，添加开机启动权限

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

重启安卓模组，测试 app 有没有自动启动。

安装 app 到模组上，然后启动一次程序（安卓 4.0 以后，必须先启动一次程序才能接收到开机完成的广播，目的是防止恶意程序）。

检查一下是不是安装了 360 等安全助手之类的软件，如果有，请在软件的自启动软件管理中将 app 设置为允许，app 安装到内部存储，不可以安装在 SD 卡上。

4.2.2 APK 设置为系统桌面

替换配置文件中的第一个启动的 activity 的 <intent-filter>

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.HOME" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

程序运行后点击 Home ，弹出 Launcher 选择框 ，选中自己开发的 APK 后点“ 始终 ”即可。

4.3 APK 加密

4.3.1 防止二次打包

1. Java 代码中加入签名校验(直接修改 smali 文件)
2. NDK 中加入签名校验(ida 查看直接 hex 修改)
3. 利用二次打包工具本身的缺陷阻止打包(manifest 欺骗，图片修改后缀等等)

4.3.2 第三方加密工具的使用

如需要进行应用程序的加密，请导出 apk 资源，通过第三方加密平台完成。

提示：第三方加密平台有多个服务商可选，参考如下：

http://dev.360.cn/protect/welcome

http://bangcle.com

4.4 蜂鸣器使用

蜂鸣器是一种一体化结构的电子讯响器，它采用直流电压供电，在电子产品中作发

声器件。蜂鸣器由 GPIO 控制实现发声。其代码如下：

第一步：创建 execShell 方法

```
public void execShell(String cmd) {
    try {
        //权限设置
        Process p = Runtime.getRuntime().exec("su");
        //获取输出流
        OutputStream outputStream = p.getOutputStream();
        DataOutputStream dataOutputStream = new DataOutputStream(outputStream);
        //将命令写入
        dataOutputStream.writeBytes(cmd);
        //提交命令
        dataOutputStream.flush();
        //关闭流操作
        dataOutputStream.close();
        outputStream.close();
    } catch (Throwable t) {
        t.printStackTrace();
    }
}
```

第二步：按照模组版本选择对应方法

```
//3128、3288——10inch 产品使用此方法
public void newBuzzer(boolean flag,int port) throws IOException,
        InterruptedException {
    execShell("echo chmod 0777 /sys/class/gpio/export");
    execShell("echo "+port+"> /sys/class/gpio/export");
    execShell("echo out > /sys/class/gpio/gpio"+port+"/direction");
    if (flag){
        execShell("echo 1 > /sys/class/gpio/gpio"+port+"/value");
    }else {
        execShell("echo 0 > /sys/class/gpio/gpio"+port+"/value");
    }
}


//3288——7inch 产品使用此方法
public void newBuzzer(boolean flag) throws IOException, InterruptedException {
    if (flag){
        execShell("echo 1 > /sys/class/leds/beep/brightness");
    }else {
        execShell("echo 0 > /sys/class/leds/beep/brightness");
```

```
    }
}


//3128——7inch 产品使用此方法
public void newBuzzer(boolean flag) throws IOException, InterruptedException {
    if (flag){
        execShell("echo 0 > /sys/class/leds/beep/brightness");
    }else {
        execShell("echo 1 > /sys/class/leds/beep/brightness");
    }
}
```

4.5 GPIO 操作

⚠ 注意：GPIO 功能仅部分屏支持，在使用前请洽询与您对接的销售人员确定模组是否支持。

新建 GPIO 类

```
public class Gpio {
    public Gpio() {
    }
    /**
     * 创建 GPIO 操作实例
     * @param gpio gpio 引脚编号
     * **/
    public String getGpio(String gpio) {
        String path = "echo " + gpio + "> /sys/class/gpio/export";
        return execRootCmd(path);
    }
    /**
     * GPIO 设置 out
     * @param gpio gpio 引脚编号
     * **/
    public void setGpioOut(String gpio) {
        execRootCmd("echo out > /sys/class/gpio/gpio" + gpio + "/direction");
    }
    /**
     * GPIO 设置 in
```

```
     * @param gpio gpio 引脚编号
     * **/
    public void setGpioIn(String gpio) {
        execRootCmd("echo in > /sys/class/gpio/gpio" + gpio + "/direction");
    }
    /**
     * GPIO 设置 高电位
     * @param gpio gpio 引脚编号
     * **/
    public void setGpioHigh(String gpio) {
        execRootCmd("echo 1 > /sys/class/gpio/gpio" + gpio + "/value");
    }
    /**
     * GPIO 设置 低电位
     * @param gpio gpio 引脚编号
     * **/
    public void setGpioLow(String gpio) {
        execRootCmd("echo 0 > /sys/class/gpio/gpio" + gpio + "/value");
    }
    /**
     * GPIO 获取当前电位
     * @param gpio gpio 引脚编号
     * **/
    public String getGpioValue(String gpio) {
        return this.getGpioString("/sys/class/gpio/gpio" + gpio + "/value");
    }
     public static String execRootCmd(String cmd) {
        String result = "";
        DataOutputStream dos = null;
        DataInputStream dis = null;
        try {
            Process p = Runtime.getRuntime().exec("su");
            dos = new DataOutputStream(p.getOutputStream());
            dis = new DataInputStream(p.getInputStream());
            dos.writeBytes(cmd + "\n");
            dos.flush();
            dos.writeBytes("exit\n");
            dos.flush();
            for(String line = null; (line = dis.readLine()) != null; result =
                    result + line) {
            }
            p.waitFor();
        } catch (Exception var18) {
            var18.printStackTrace();
```

```
            } finally {
                if (dos != null) {
                    try {
                        dos.close();
                    } catch (IOException var17) {
                        var17.printStackTrace();
                    }
                }
                if (dis != null) {
                    try {
                        dis.close();
                    } catch (IOException var16) {
                        var16.printStackTrace();
                    }
                }
            }
            return result;
        }
        private String getGpioString(String path) {
            String defString = "0";
            try {
                BufferedReader reader = new BufferedReader(new FileReader(path));
                defString = reader.readLine();
            } catch (IOException var4) {
                var4.printStackTrace();
            }
            return defString;
        }
}
```

使用方法

```
//获取 GPIO
Gpio gpio = new Gpio();
//GPIO 位号，可以咨询与您对接的销售
gpio.getGpio("165");
//设置 GPIO165 为输入状态
gpio.setGpioIn("165");
//获取 GPIO165 为的状态返回 0 或 1
gpio.getGpioValue("165");
//设置 GPIO165 为输出状态
gpio.setGpioOut("165");
//设置 GPIO165 为高电位
gpio.setGpioHigh("165");
```

```
//设置 GPIO165 为低电位
gpio.setGpioLow("165");
```

⚠ 注意：GPIO 位号需要进行换算，在使用前请洽询与您对接的销售人员确定 GPIO 位号。以下是换算方法：

每个 GPIO 的端口值通过查找表和计算来计算。

我们知道，gpio 通常分为几组，如 GPIO1, GPIO2, GPIO3…

```
RK3128 系列计算公式是：num = 32 * GPIO_X + PORT

GPIO_X 为 GPIO 的组号，对应关系如下

GPIO0 --> 0

GPIO1 --> 1

GPIO2 --> 2

...

PORT 的值如下表所示

#define GPIO_A0 0

#define GPIO_A1 1

#define GPIO_A2 2

#define GPIO_A3 3

#define GPIO_A4 4

#define GPIO_A5 5

#define GPIO_A6 6

#define GPIO_A7 7

#define GPIO_B0 8

#define GPIO_B1 9

#define GPIO_B2 10

#define GPIO_B3 11

#define GPIO_B4 12

#define GPIO_B5 13

#define GPIO_B6 14
```

#define GPIO_B7 15

#define GPIO_C0 16

#define GPIO_C1 17

#define GPIO_C2 18

#define GPIO_C3 19

#define GPIO_C4 20

#define GPIO_C5 21

#define GPIO_C6 22

#define GPIO_C7 23

#define GPIO_D0 24

#define GPIO_D1 25

#define GPIO_D2 26

#define GPIO_D3 27

#define GPIO_D4 28

#define GPIO_D5 29

#define GPIO_D6 30

#define GPIO_D7 31

例如，如果我们想控制 gpio2_d4 引脚，它的端口号是

2 * 32 + 28 = 92

如果我们想控制 gpio0_a1 引脚，它的端口号是

0 * 32 + 1 = 1

RK3288 系列 Android5.1 请按照以下方法计算

将引脚定义转换为以下形式

GPIOX_YZ

我们得到 3 个参数 X Y Z;引脚定义中的 Y 以字母的形式存在，并且需要存在

用数字代替对应关系如下

A = 0

B = 1

C = 2

D = 3

公式：(X*32)+(Y*8)+Z

示例：

GPIO1_A7 (1*32)+(0*8)+7 = 39

GPIO1_B1 (1*32)+(1*8)+1 = 41

GPIO2_C4 (2*32)+(2*8)+4 = 84

GPIO3_D0 (3*32)+(3*8)+1 = 120

GPIO3_D6 (3*32)+(3*8)+6 = 126

RK3288 系列 Android7.1 请按照以下方法计算

当 X=0 时(引脚定义从 GPIO0 开始的 gpio)

公式：(Y*8)+Z

示例：

GPIO0_A7 (0*8)+7 = 7

GPIO0_B4 (1*8)+4 = 12

GPIO0_C2 (2*8)+2 = 18

当 X>0 时(gpio 以非 GPIO0 的形式开始)

公式：24+((X-1)*32)+(Y*8)+Z

示例：

GPIO1_A7 24+((1-1)*32)+(0*8)+7 = 31

GPIO1_B1 24+((1-1)*32)+(1*8)+1 = 33

GPIO2_C4 24+((2-1)*32)+(2*8)+4 = 76

GPIO3_D0 24+((3-1)*32)+(3*8)+0 = 112

GPIO3_D6 24+((3-1)*32)+(3*8)+6 = 118

4.6 功能代码示例

4.6.1 4G 问题解答

1. 提示无 SIM 卡

进入 设置-关于设备 ，查看基带版本，若无基带版本请在断电后，尝试将 4G 模块拧下，擦拭金手指后重新插入 MINI-PCIE 接口。若依旧无基带，请将系统版本与模组背部拍照发给与您对接的销售。

进入 设置-关于设备 ，查看基带版本，若有基带版本，请在断电后将 SIM 卡擦拭，轻轻擦拭 SIM 卡坐金手指，将 SIM 放入卡座轻轻按压几下，开机查看是否有 SIM 卡，若依旧无 SIM 卡，请将系统版本与模组背部拍照发给与您对接的销售。

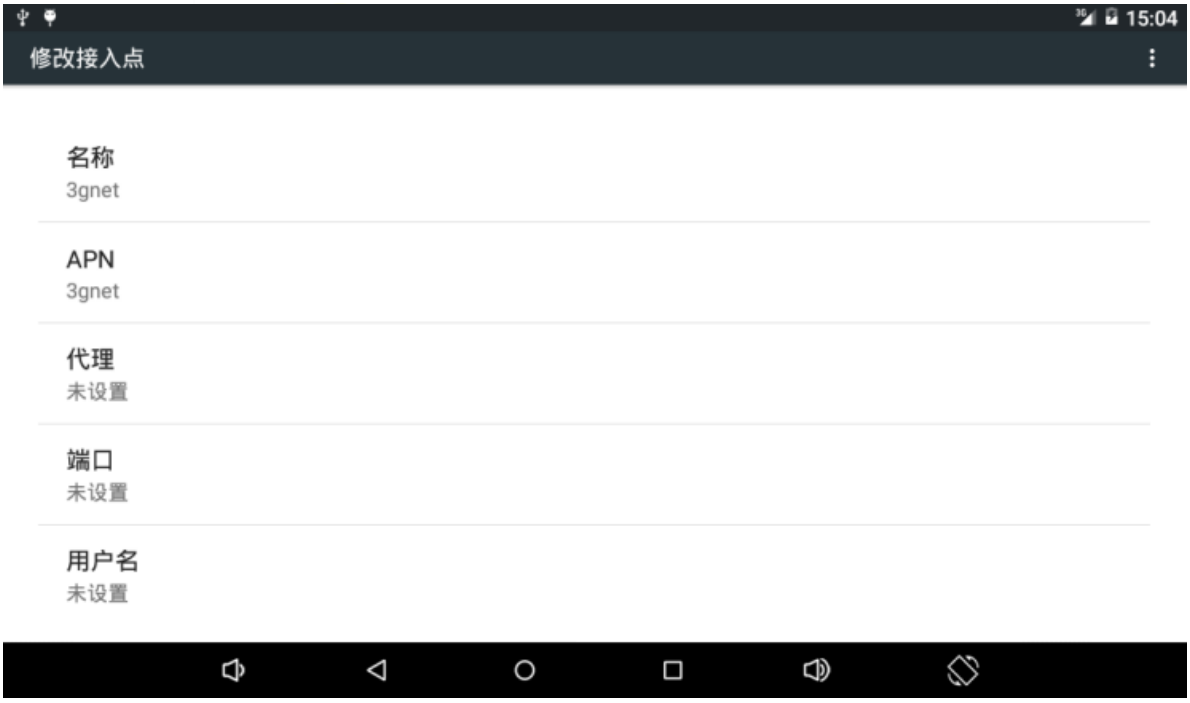
2. 有信号但是无法上网

进入 设置-更多-移动网络-接入点名称（APN） ，点击右上角+号按钮，添加 APN。APN 进行拨号时需要加上运营商的 APN，不同运营商的 APN 不同。各 APN 如下：

中国移动：cmnet

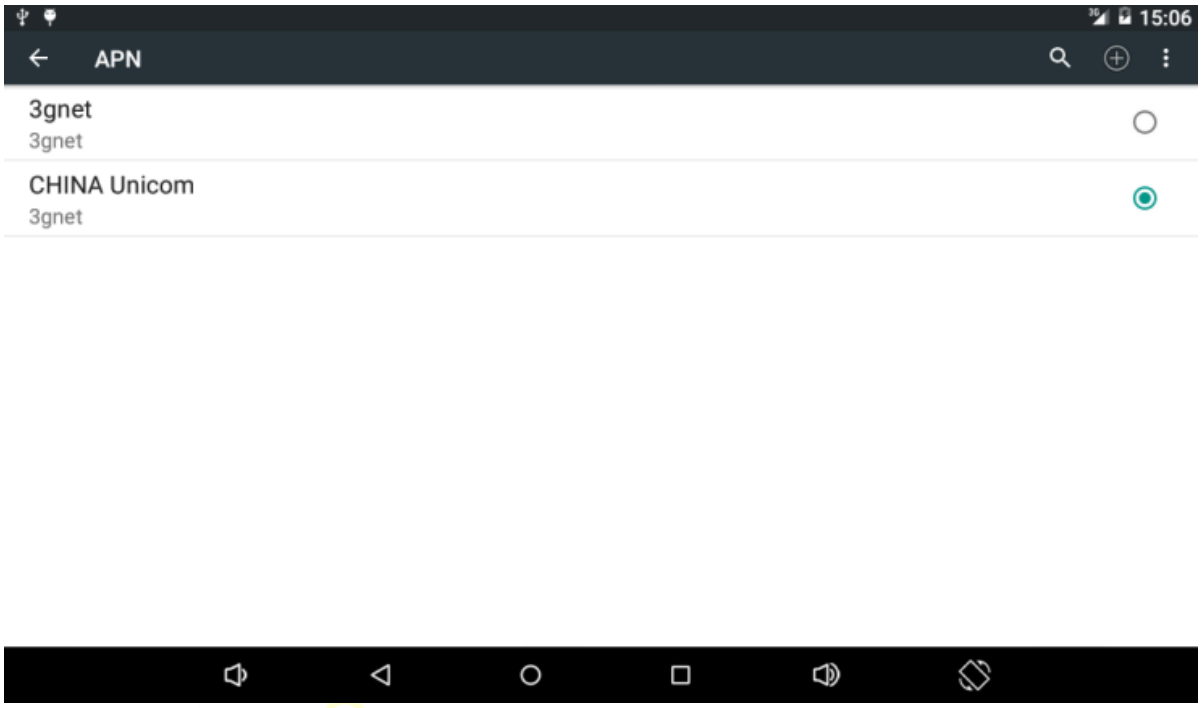
中国联通：3gnet

中国电信：ctnet



以联通卡为例，名称填写 3gnet，apn 填写 3gnet。点击右上角的按钮并保存。

选择刚刚新建好的 APN，稍等片刻状态栏信号会出现 4g 或 3g 字样（根据信号而定）



4.6.2 调用 shell 命令

```
//这个方法用来执行命令
public void execShell(String cmd) {
    try {
        //权限设置
        Process p = Runtime.getRuntime().exec("su");
        //获取输出流
        OutputStream outputStream = p.getOutputStream();
        DataOutputStream dataOutputStream = new DataOutputStream(outputStream);
        //将命令写入
        dataOutputStream.writeBytes(cmd);
        //提交命令
        dataOutputStream.flush();
        //关闭流操作
        dataOutputStream.close();
        outputStream.close();
    } catch (Throwable t) {
        t.printStackTrace();
    }
}
```

使用示例

```
添加 execShell 方法到需要的 JAVA 类中，调用以下方法实现以太网的关和开

execShell("busybox ifconfig eth0 down");//关闭以太网

execShell("busybox ifconfig eth0 up");//打开以太网

修改 I2C0 设备权限

execShell("chmod 777 /dev/i2c-0");
```

4.6.3 HDMI 双屏异显

调用 4.6.2 的 execShell 方法可以实现控制 HDMI 的输出功能

```
execShell("on > /sys/class/drm/card0-HDMI-A-1/status");//打开 hdmiout 显示
execShell("off > /sys/class/drm/card0-HDMI-A-1/status");//关闭 hdmiout 显示
execShell("cat /sys/class/drm/card0-HDMI-A-1/status");//打开是 connected，关闭是 disconnected

/**
 * Android 7.1 HDMI 双屏异显
 * 如果不打开双屏异显，hdmi 显示的界面会在左上角一小块，而不是全屏显示。
 * Set HDMI Dual Screen Mode
 ***/
public void setHDMIDualScreenMode() {
    if (Build.VERSION.SDK_INT >= 25) {
        boolean state = Settings.System.getInt(this.getContentResolver(), "dual_screen_mode", 0) == 1;
        if (state) {
            Log.d("HDMI", "Android7.1 双屏异显已打开");
        } else {
            Settings.System.putInt(this.getContentResolver(), "dual_screen_mode", 1);
            Log.d("HDMI", "Android7.1 双屏异显打开中");
            Toast.makeText(this, "检测到双屏异显未打开，已帮您打开，需要重启应用。",
Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}
```

4.6.4 配置安装 ADB(安卓调试桥)

ADB 全称为 Android Debug Bridge，起到调试桥的作用，是一个客户端-服务器端程序。其中客户端是用来操作的电脑，服务端是 Android 设备。

ADB 也是 Android SDK 中的一个工具，可以直接操作管理 Android 模拟器或者真实

的 Android 设备。

1.下载 ADB

Windows 版本：https://dl.google.com/android/repository/platform-tools-latest-windows.zip

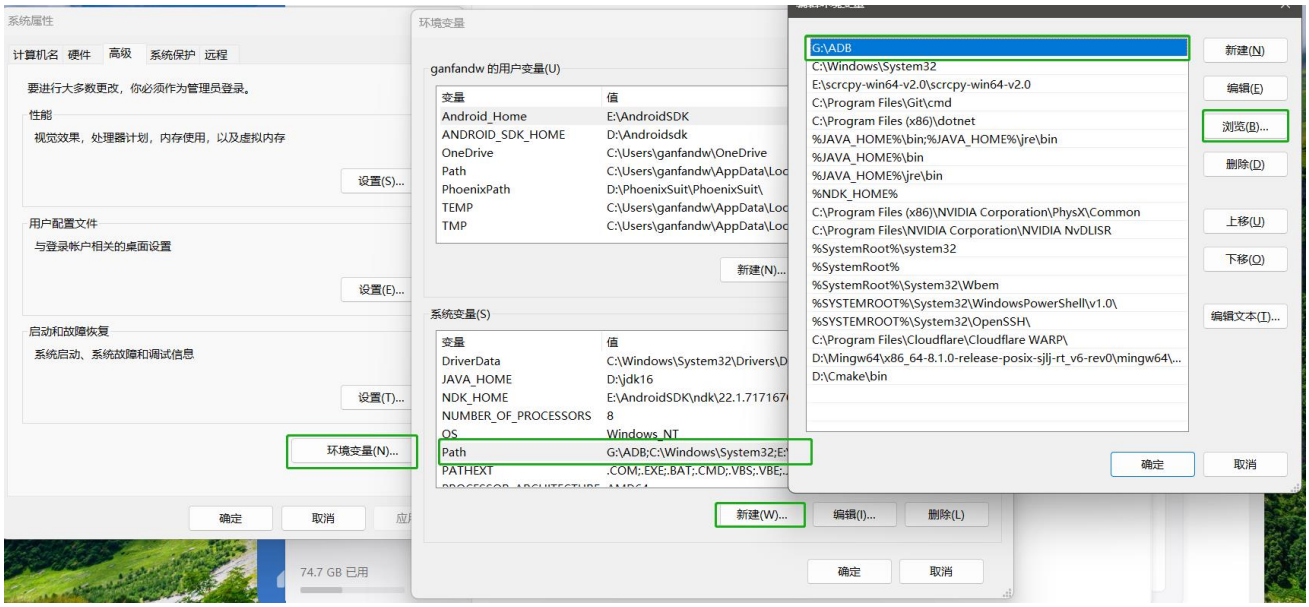
Mac 版本：https://dl.google.com/android/repository/platform-tools-latest-windows.zip

Linux 版本：https://dl.google.com/android/repository/platform-tools-latest-linux.zip

2.配置环境变量

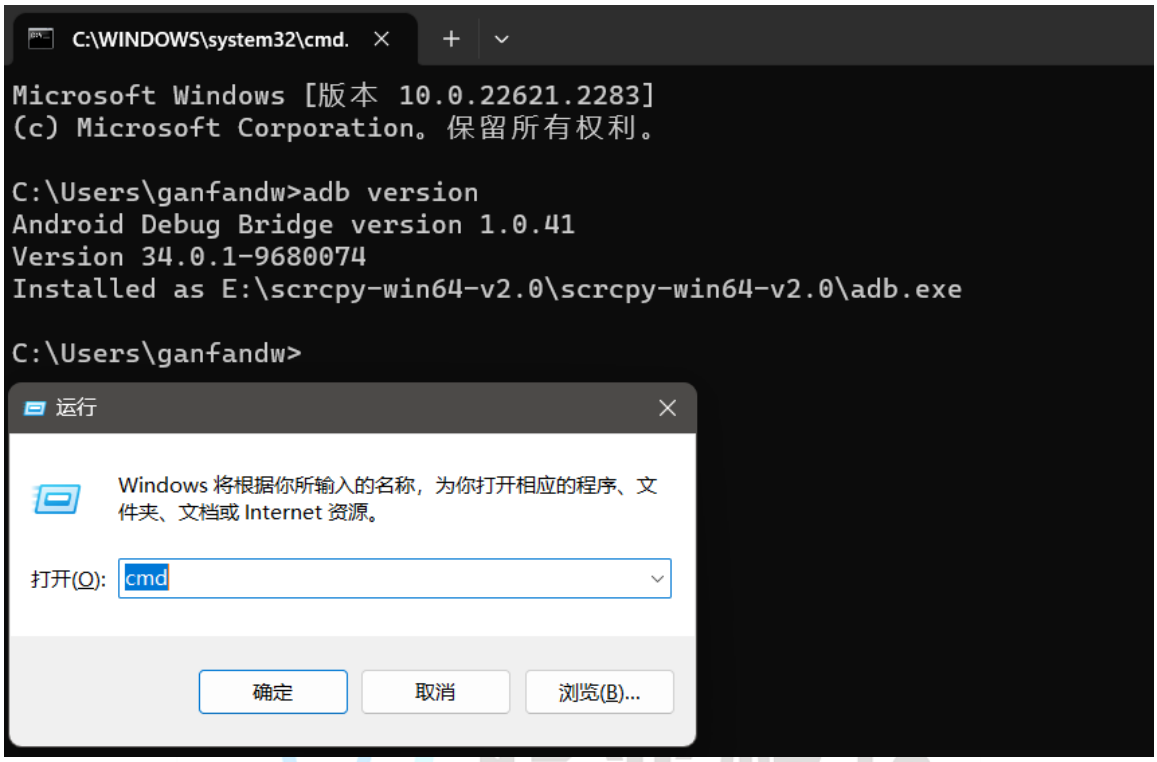
将下载的文件解压，把解压的路径放到系统变量中。

系统属性->环境变量->选中 Path 变量->编辑->添加或者浏览选中 ADB 解压的路径。



3.检查 ADB 安装状态

Windows 快捷键 WIN+R 调出运行，输入 cmd 打开命令行。输入 adb version 显示 adb 版本表示安装成功。



4.连接设备

通过 USB 数据线与安卓模组连接，打开开发者选项，进入开发者选项后打开 USB 调试功能。打开 CMD 命令行，输入 adb devices 查看是否有设备。

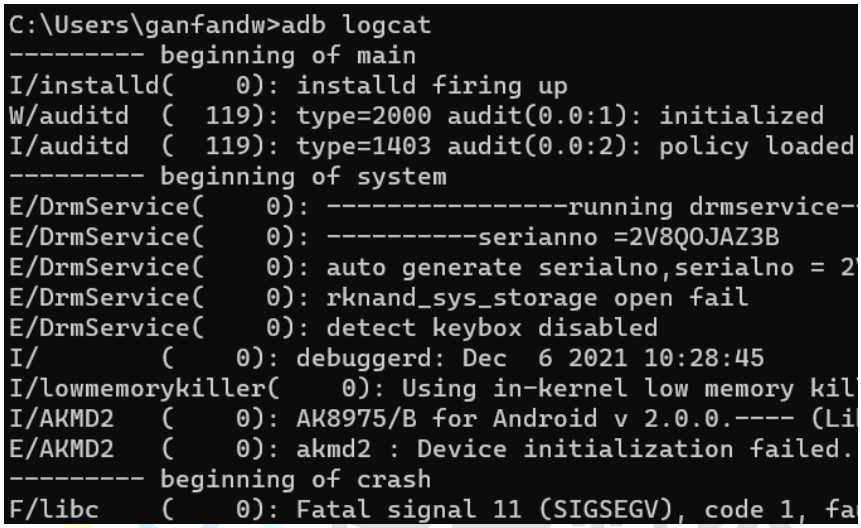
⚠ 注意：打开开发者选项方法：设置->关于本机->系统版本号点击十次-提示开发者选项

已打开->返回->开发者选项->usb 调试打开

4.6.5 通过 ADB(安卓调试桥)抓取日志

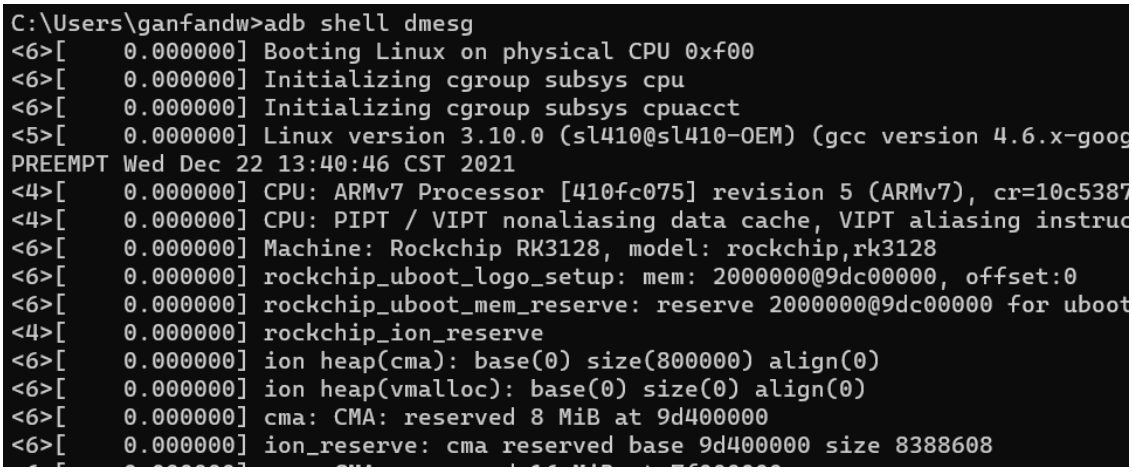
1.抓取通用日志

```
adb logcat
```



2.抓取内核日志(Dmesg/kernel logs)

```
adb shell dmesg 或者 adb shell " cat /proc/kmsg "
```



3. 抓取 ANR 日志

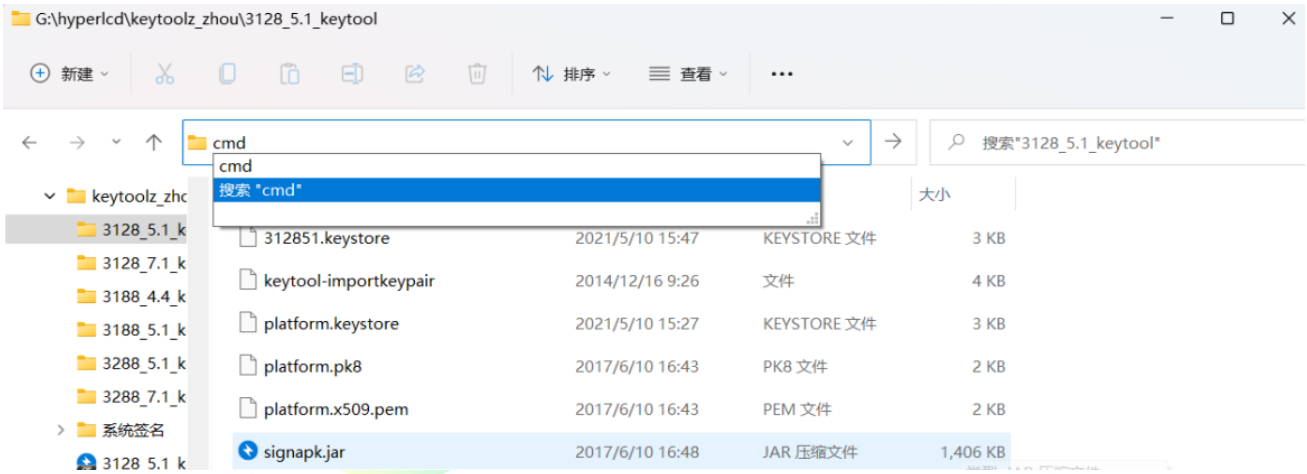
```
adb shell cat /data/anr/traces.txt
```

4.6.6 给应用添加系统签名

我们的应用程序有时需要系统级签名才能访问某些方法，超显科技已经准备好签名所需的所有文件。请联系与您对接的销售人员获取文件。手动系统签名与 Android 的 APK 重签名不同，重签名是之前 APK 已经签名完成，在实际使用时，需要更改签名文件，例如一些特殊的算法处理对于系统的包名和签名都有绑定操作，如更换则无法使用。 手动系统签名是对于通过 AndroidStudio 的 build 生成的无签名的 APK 文件，进行手动系统签名操作。

具体执行过程如下：

1.下载打包好的签名工具与 security 文件



2.执行系统签名操作，在文件夹链接处键入 CMD，打开 CMD 命令行

3.输入下方命令行

```
java -jar signapk.jar platform.x509.pem platform.pk8 in.apk out.apk
```

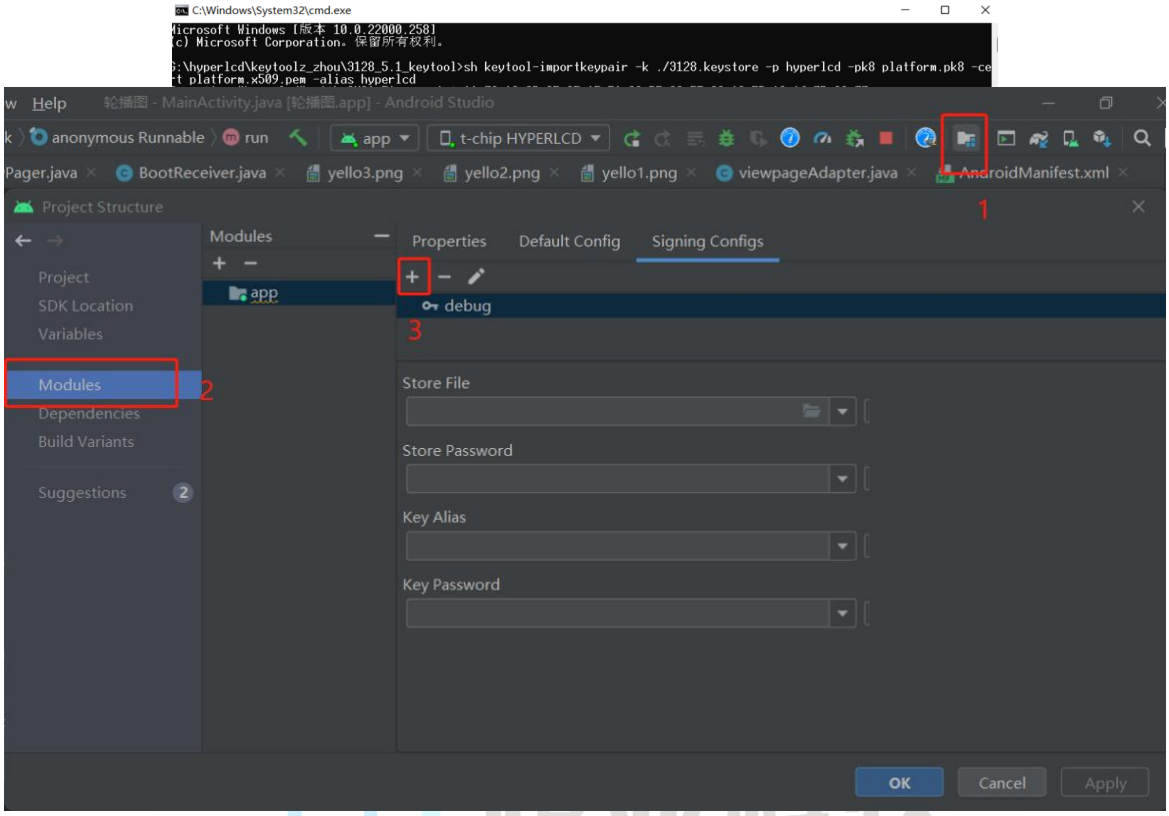
in.apk 是 Android Studio 构建出来的未签名应用，需要放在签名的文件夹中，out.apk 就是签名后的应用。

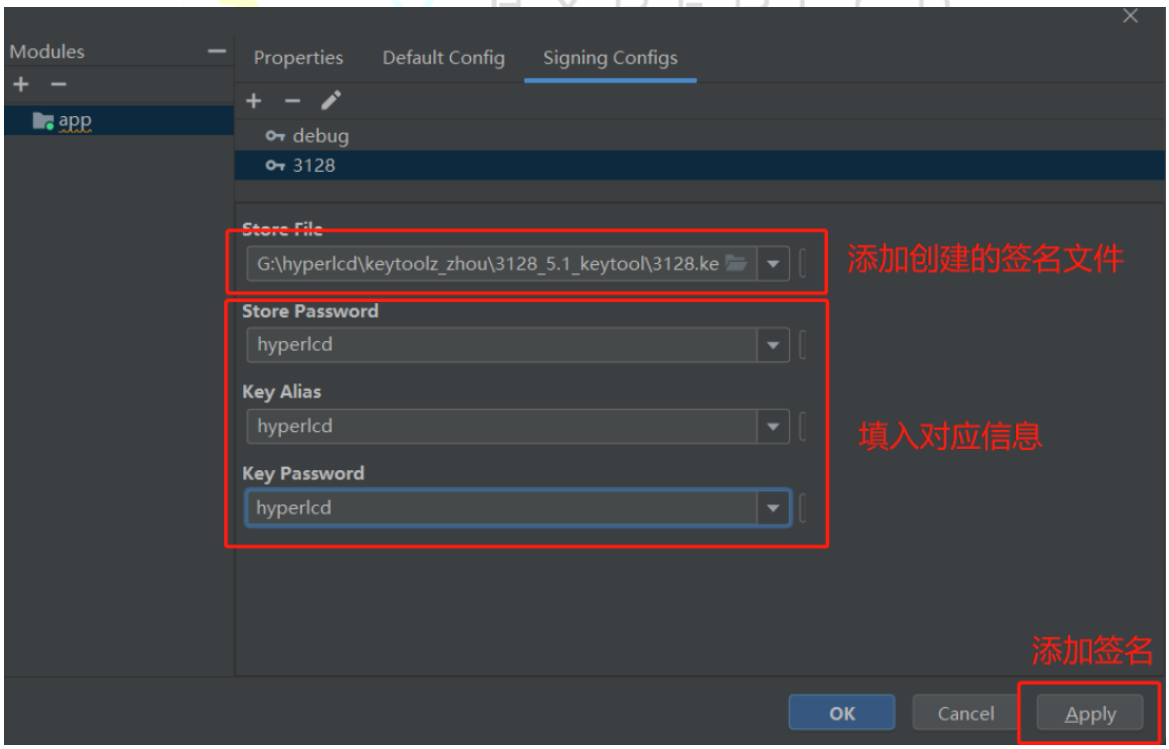
更推荐另一种方式，输入下方命令行

```
sh keytool-importkeypair -k ./3128.keystore -p hyperlcd -pk8 platform.pk8 -cert

platform.x509.pem -alias hyperlcd

_____

k: 生成的 keystore 文件

pk8: 要导入的 platform.pk8 文件

cert: 要导入的 platform.x509.pem 文件
```

alias 生成的 keystore 的 alias

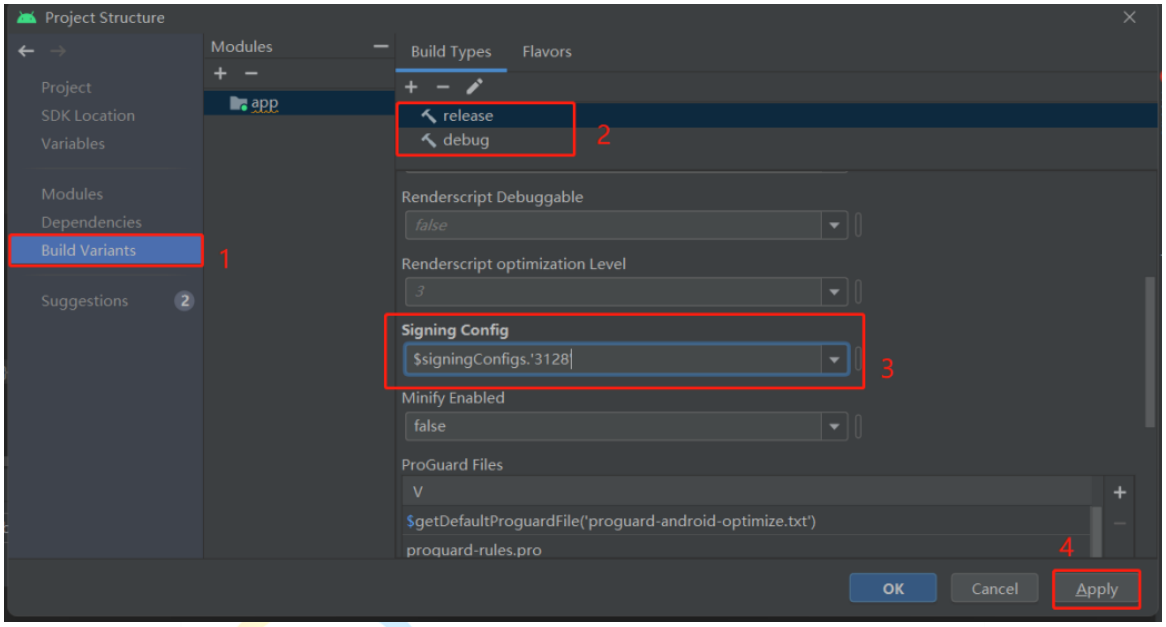
p: 生成的 keystore 文件的密码





可以看到系统签名已经导出，这时在 Android Studio 中选择签名即可。

添加完成后，选择 debug 与 release 时的默认签名



app 的 AndroidManifest.xml 里添加

```
android:sharedUserId="android.uid.system"
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:sharedUserId="android.uid.system"
    package="com.hyperlcd.usertest">
    <meta-data android:name="android.webkit.WebView.EnableSafeBrowsing"
        android:value="true"/>
```

⚠ 注意：此方法可以解决掉在 android studio 中编译出来 apk 就是安装不上,报如下错误的问题 INSTALL_FAILED_SHARED_USER_INCOMPATIBLE，若未解决，请尝试重新下载对应系统版本的系统签名 .

4.6.7 隐藏系统导航栏

创建 HideMenu 类，通过 HideMenus 方法，传入 true 为隐藏，传入 false 为显示。可

搭配生命周期实现退出显示导航栏，进入应用隐藏导航栏。

```
public class HideMenu {
    //安卓 7.1 隐藏导航栏的广播
    private final static String show = "am broadcast -a action.SHOW_STATUSBAR";
    //安卓 7.1 显示导航栏的广播
    private final static String hide = "am broadcast -a action.HIDE_STATUSBAR";
    //这个方法用来执行命令
    public void execShell(String cmd){
        try{
            //权限设置
            Process p = Runtime.getRuntime().exec("su");
            //获取输出流
            OutputStream outputStream = p.getOutputStream();
            DataOutputStream dataOutputStream=new DataOutputStream(outputStream);
            //将命令写入
            dataOutputStream.writeBytes(cmd);
            //提交命令
            dataOutputStream.flush();
            //关闭流操作
            dataOutputStream.close();
            outputStream.close();
        }
        catch(Throwable t)
        {
            t.printStackTrace();
        }
    }
    public void HideMenus(boolean flag, Context context){
        //安卓 7.0 以下使用的方法
        if (Build.VERSION.SDK_INT > 11 && Build.VERSION.SDK_INT <= 23) {
            if (flag){
                Intent intent = new Intent();
                intent.setAction("marvsmart_bar");
                intent.putExtra("marvsmart_swich", false);
                intent.putExtra("marvsmart_toast", "");
                context.sendBroadcast(intent);
            }else {
                Intent intent = new Intent();
                intent.setAction("marvsmart_bar");
                intent.putExtra("marvsmart_swich", true);
                intent.putExtra("marvsmart_toast", "");
                context.sendBroadcast(intent);
            }
```

```
//安卓 7.0 以上使用的方法
        } else if (Build.VERSION.SDK_INT > 23) {
            Log.d("menu",">21");
            if(flag){
                execShell(hide);
            }else {
                execShell(show);
            }
        }
    }
}
```

使用方法

```
//隐藏状态栏
HideMenus(true,context);
//显示状态栏
HideMenus(false,context);
```

也可以对应用进行系统签名后使用以下方法

```
Settings.System.putInt(this.getContentResolver(),"always_hide_bar",0);//0 显示，1 隐藏
sendBroadcast(new Intent("action.ALWAYS_HIDE_STATUSBAR_CHENAGE"));
```

4.6.8 修改系统时间

修改系统时间需要添加系统签名

在 AndroidManifest.xml 的根节点中设置 android:sharedUserId="android.uid.system"属性，编译完成后拥有修改系统功能的权限

```
android:sharedUserId="android.uid.system"

<!--//网络和时间相关权限-->
<uses-permission android:name="android.permission.SET_TIME"
    tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.WRITE_SETTINGS"
    tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.WRITE_SECURE_SETTINGS"
    tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.SET_TIME_ZONE"
    tools:ignore="ProtectedPermissions" />
```

添加修改系统时间的方法

```
/**
 * 设置系统时间
```

```
 * @param year 年 示例“2023”
 * @param month 月 示例“3”
 * @param day 日 示例”1“
 * @param hour 时 示例“1”
 * @param minute 分 示例“1”
 * @param second 秒 示例“1”
 * @throws Settings.SettingNotFoundException
 */
public void setNowTime(int year ,int month ,int day,int hour ,int minute,int second) throws
        Settings.SettingNotFoundException {
    //关闭自动确定日期和时间
    int autoTime = Settings.Global.getInt(mContext.getContentResolver(),
            Settings.Global.AUTO_TIME);
    if (autoTime == 1) {
        Settings.Global.putInt(mContext.getContentResolver(),
                Settings.Global.AUTO_TIME, 0);
    }
    //关闭自动确定时区
    int autoZoneEnable = Settings.Global.getInt(mContext.getContentResolver(),
            Settings.Global.AUTO_TIME_ZONE);
    if (autoZoneEnable == 1) {
        Settings.Global.putInt(mContext.getContentResolver(),
                Settings.Global.AUTO_TIME_ZONE, 0);
    }
    //设置 24 小时制
    if (!DateFormat.is24HourFormat(mContext)) {
        Settings.System.putString(mContext.getContentResolver(),
                Settings.System.TIME_12_24, "24");
    }
    Calendar c = Calendar.getInstance();
    c.set(Calendar.YEAR, year);
    c.set(Calendar.MONTH, month);
    c.set(Calendar.DAY_OF_MONTH, day);
    c.set(Calendar.HOUR_OF_DAY, hour);
    c.set(Calendar.MINUTE, minute);
    c.set(Calendar.SECOND, second);
    long when = c.getTimeInMillis();
    ((AlarmManager) mContext.getSystemService(Context.ALARM_SERVICE)).setTime(when);
}
```

使用方法：

```
setNowTime(Year,Month,Day,Hour,Minute,0)
```

4.6.9 修改屏幕方向

本节将介绍两种修改屏幕方向的方法，由代码控制屏幕方向与修改系统配置定义屏幕方向。

一．代码修改屏幕方向

```
//这个方法用来执行命令
public void execShell(String cmd){
    try{
        //权限设置
        Process p = Runtime.getRuntime().exec("su");
        //获取输出流
        OutputStream outputStream = p.getOutputStream();
        DataOutputStream dataOutputStream=new DataOutputStream(outputStream);
        //将命令写入
        dataOutputStream.writeBytes(cmd);
        //提交命令
        dataOutputStream.flush();
        //关闭流操作
        dataOutputStream.close();
        outputStream.close();
    }
    catch(Throwable t)
    {
        t.printStackTrace();
    }
}
```

添加四个 Button 并设置监听器

定义四个按钮用于点击测试 按钮 0 正常 按钮 1 横屏 按钮 2 反转竖屏 按钮 3 反转横屏

```
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.bt_set_0:
            execShell("settings put system accelerometer_rotation 0");
            execShell("settings put system user_rotation "+0);
            break;
        case R.id.bt_set_1:
            execShell("settings put system accelerometer_rotation 0");
            execShell("settings put system user_rotation "+1);
            break;
```

```
            case R.id.bt_set_2:
                execShell("settings put system accelerometer_rotation 0");
                execShell("settings put system user_rotation "+2);
                break;
            case R.id.bt_set_3:
                execShell("settings put system accelerometer_rotation 0");
                execShell("settings put system user_rotation "+3);
                break;
        }
}
```

二．ES 文件管理器修改屏幕方向

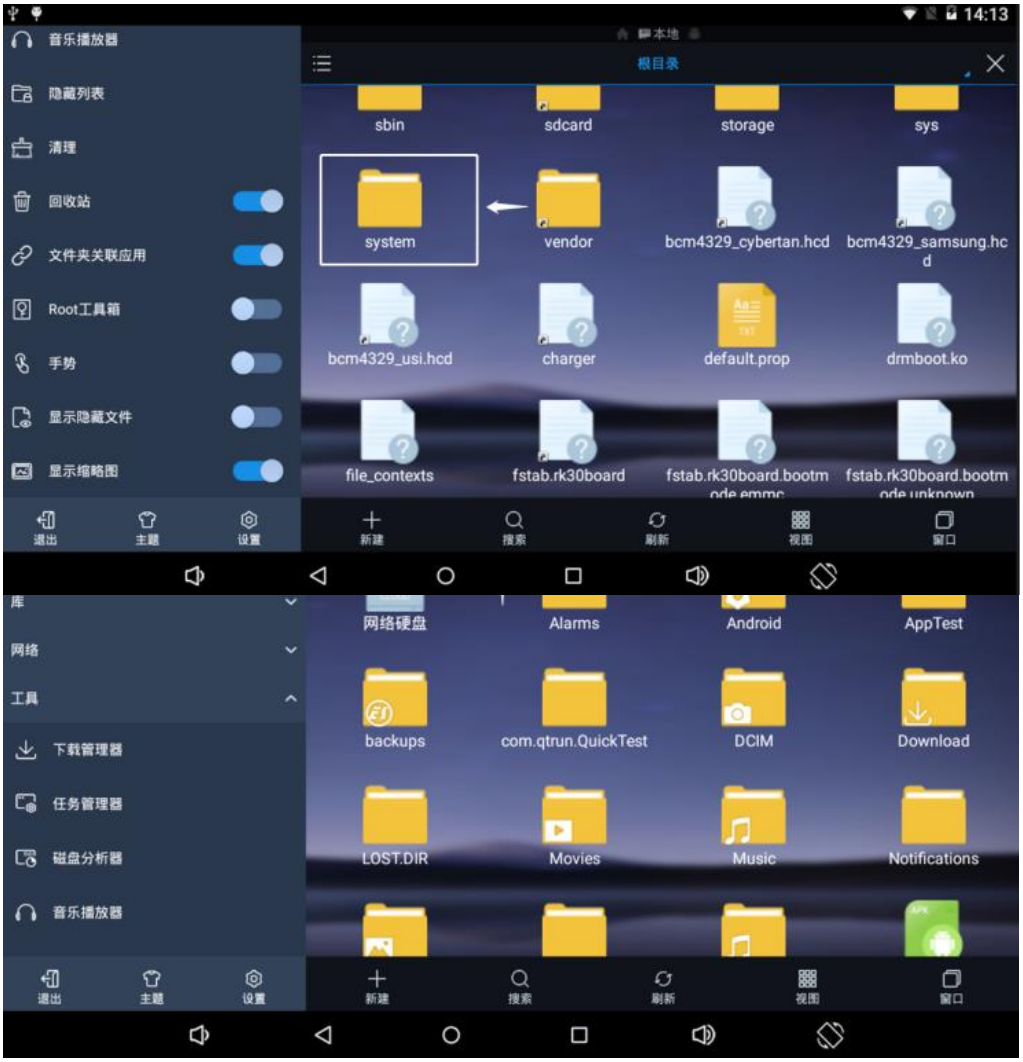
⚠ 注意：此方法有风险!！请勿修改其他数据!！可能会导致系统崩溃!！

1.下载 ES 文件管理器

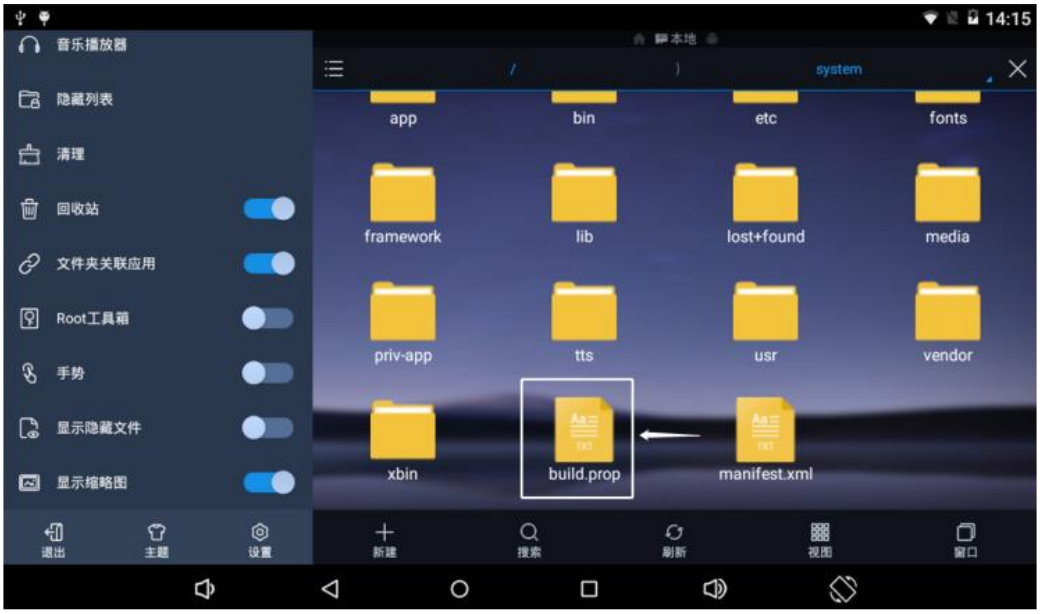
打开 ROOT 工具箱

点击进入根目录

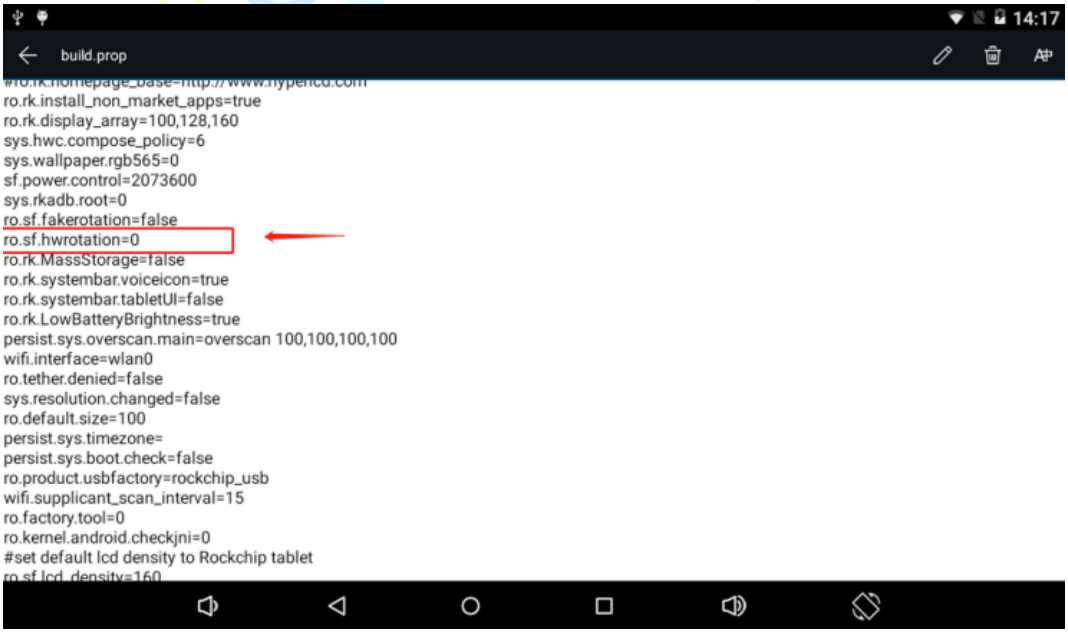
找到 System 文件夹



找到 build.prop 文件，打开后选择编辑



找到 ro.sf.hwrotation=0 修改为 ro.sf.hwrotation=90 保存后重启



4.6.10 设置应用为桌面

设置应用为主桌面，在 AndroidManifest.xml 文件中将应用首页的意图过滤添加以下几

项参数，重启应用选择为 Launcher。

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
    <category android:name="android.intent.category.HOME" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.MONKEY" />
</intent-filter>
```

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.HOME" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.MONKEY" />
    </intent-filter>
</activity>
```


4.6.11 设置静态以太网

此方法适用于 Android5.1-Android7.1 设置静态以太网，首先导入文件夹内的 EthernetManager.jar，然后创建 NetUtils 类。

```
public class NetUtils {
    /*
    * convert subMask string to prefix length
    */
    public static int maskStr2InetMask(String maskStr) {
        StringBuffer sb ;
        String str;
        int inetmask = 0;
        int count = 0;
       /*
        * check the subMask format
        */
        Pattern pattern = Pattern.compile("(^((\\d|[01]?\\d\\d|2[0-4]\\d|25[0-5])\\.){3}(\\d|[01]?\\d\\d|2[0-
4]\\d|25[0-5])$)|^(\\d|[1-2]\\d|3[0-2])$");
        if (pattern.matcher(maskStr).matches() == false) {
```

```
                Log.e("33333","subMask is error");
                return 0;
            }

            String[] ipSegment = maskStr.split("\\.");
            for(int n =0; n<ipSegment.length;n++) {
                sb = new StringBuffer(Integer.toBinaryString(Integer.parseInt(ipSegment[n])));
                str = sb.reverse().toString();
                count=0;
                for(int i=0; i<str.length();i++) {
                    i=str.indexOf("1",i);
                    if(i==-1)
                        break;
                    count++;
                }
                inetmask+=count;
            }
            return inetmask;
        }


        public static Inet4Address getIPv4Address(String text) {
            try {
                return (Inet4Address) NetworkUtils.numericToInetAddress(text);
            } catch (IllegalArgumentException|ClassCastException e) {
                return null;
            }
        }

}
```

以下是操作静态以太网的变量

```
private StaticIpConfiguration mStaticIpConfiguration;
private IpConfiguration mIpConfiguration;
private EthernetManager mEthManager;
private static String mEthIpAddress = "192.168.88.154"; //IP
private static String mEthNetmask = "255.255.255.0"; //  子网掩码
private static String mEthGateway = "192.168.88.1"; //网关
private static String mEthdns1 = "8.8.8.8"; // DNS1
private static String mEthdns2 = "8.8.4.4"; // DNS2

/**
 * 设置静态以太网
 *
```

```
 * @param ip          ip 地址
 * @param gateway  网关
 * @param netmask  子网掩码
 * @param dns1       dns
 * @param dns2       dns
 */
public void setEthernetStaticIp(String ip, String gateway, String netmask, String dns1, String dns2) {
    mStaticIpConfiguration = new StaticIpConfiguration();
    /*
     * get ip address, netmask,dns ,gw etc.
     */
    Inet4Address inetAddr = getIPv4Address(ip);
    int prefixLength = maskStr2InetMask(netmask);
    InetAddress gatewayAddr = getIPv4Address(gateway);
    InetAddress dnsAddr1 = getIPv4Address(dns1);
    InetAddress dnsAddr2 = getIPv4Address(dns2);
    if (inetAddr.getAddress().toString().isEmpty() || prefixLength == 0 ||
            gatewayAddr.toString().isEmpty()
            || dnsAddr1.toString().isEmpty() || dnsAddr2.toString().isEmpty()) {
        return;
    }
    Class<?> clazz = null;
    try {
        clazz = Class.forName("android.net.LinkAddress");
        Class[] cl = new Class[]{InetAddress.class, int.class};
        Constructor cons = null;
        //取得所有构造函数
        cons = clazz.getConstructor(cl);
        //给传入参数赋初值
        Object[] x = {inetAddr, prefixLength};
        mStaticIpConfiguration.ipAddress = (LinkAddress) cons.newInstance(x);
        mStaticIpConfiguration.gateway = gatewayAddr;
        mStaticIpConfiguration.dnsServers.add(dnsAddr1);
        mStaticIpConfiguration.dnsServers.add(dnsAddr2);
        mIpConfiguration = new
                IpConfiguration(IpConfiguration.IpAssignment.STATIC,
                IpConfiguration.ProxySettings.NONE, mStaticIpConfiguration,
                null);
        mEthManager.setConfiguration(mIpConfiguration);
    } catch (Exception e) {
        // TODO: handle exception
        ILogUtils.e(e.toString());
    }
```

```
}

/**
 * 设置以太网为 DHCP
 */
public void setEthernetDhcp() {
    mIpConfiguration = new
            IpConfiguration(IpConfiguration.IpAssignment.DHCP,
            IpConfiguration.ProxySettings.NONE, null,
            null);
    mEthManager.setConfiguration(mIpConfiguration);
}
```

在 AndroidManifest.xml 文件中添加权限

```
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CONNECTIVITY_INTERNAL" />
```

最后给应用添加系统签名，请参考本文档的 4.6.6 章节。

安装软件，连接以太网查看以太网是否为静态 ip 地址是否为以下数据：

```
"192.168.88.154"; //IP

"255.255.255.0"; // 子网掩码

"192.168.88.1"; //网关

"8.8.8.8"; // DNS1

"8.8.4.4"; // DNS2
```

4.6.12 设置静态 WIFI

此方法适用于 Android5.1-Android7.1 设置静态 WIFI，首先导入文件夹内的 EthernetManager.jar，然后创建 NetUtils 类。

```
public class NetUtils {
    /*
    * convert subMask string to prefix length
    */
    public static int maskStr2InetMask(String maskStr) {
        StringBuffer sb ;
        String str;
```

```
            int inetmask = 0;
            int count = 0;
           /*
            * check the subMask format
            */
            Pattern pattern = Pattern.compile("(^((\\d|[01]?\\d\\d|2[0-4]\\d|25[0-5])\\.){3}(\\d|[01]?\\d\\d|2[0-
4]\\d|25[0-5])$)|^(\\d|[1-2]\\d|3[0-2])$");
            if (pattern.matcher(maskStr).matches() == false) {
                Log.e("33333","subMask is error");
                return 0;
            }

            String[] ipSegment = maskStr.split("\\.");
            for(int n =0; n<ipSegment.length;n++) {
                sb = new StringBuffer(Integer.toBinaryString(Integer.parseInt(ipSegment[n])));
                str = sb.reverse().toString();
                count=0;
                for(int i=0; i<str.length();i++) {
                    i=str.indexOf("1",i);
                    if(i==-1)
                        break;
                    count++;
                }
                inetmask+=count;
            }
            return inetmask;
        }


        public static Inet4Address getIPv4Address(String text) {
            try {
                return (Inet4Address) NetworkUtils.numericToInetAddress(text);
            } catch (IllegalArgumentException|ClassCastException e) {
                return null;
            }
        }

}
```

以下是操作静态 WIFI 的变量

```
//定义需要用到的变量，修改这些变量就行
public String STATIC_IP = "192.8.8.8";
public String STATIC_NETMASK = "255.0.0.0";
public String STATIC_GATEWAY = "255.255.255.225";
```

```
public String STATIC_DNS1 = "192.8.8.8";
public String STATIC_DNS2 = "192.8.8.8";

private static android.net.wifi.WifiManager mWifiManager;

private static Context mContext;
```

以下是设置静态 WIFI 的方法

```
/**
 * wifi 控制构造方法，不需要回调监听
 */
public IWifiManager(Context context) {
    mContext = context;
    mWifiManager = (android.net.wifi.WifiManager) mContext.getSystemService(Context.WIFI_SERVICE);
}


/**
 * 已连接 wifi 通过 ip 地址、网关、dns 配置 STATIC Wifi
 * @param ip 设备 ip 地址
 * @param gateway 网关
 * @param netmask 子网掩码
 * @param dns1 DNS1
 * @param dns2 DNS2
 * @return 是否设置成功
 */
public boolean setStaticWifi(String ip, String gateway, String netmask, String dns1, String dns2) {
    List<WifiConfiguration> configuredNetworks = getConfiguredNetworks();
    WifiConfiguration wifiConfig = null;
    WifiInfo connectionInfo = mWifiManager.getConnectionInfo();   //得到连接的 wifi 网络
    for (WifiConfiguration conf : configuredNetworks) {
        if (conf.networkId == connectionInfo.getNetworkId()) {
            wifiConfig = conf;
            break;
        }
    }
    return setStaticWifi(wifiConfig, ip, gateway, netmask, dns1, dns2);
}


private boolean setStaticWifi(WifiConfiguration config, String ip, String gateway, String netmask, String dns1,
String dns2) {
    try {
        WifiConfiguration wifiConfig = config;
        Inet4Address inetAddr = getIPv4Address(ip);
```

```
            int prefixLength = maskStr2InetMask(netmask);
            InetAddress gatewayAddr = getIPv4Address(gateway);
            InetAddress dnsAddr1 = getIPv4Address(dns1);
            InetAddress dnsAddr2 = getIPv4Address(dns2);
            Class[] cl = new Class[]{InetAddress.class, int.class};
            Constructor cons = null;

            Class<?> clazz = Class.forName("android.net.LinkAddress");

            //取得所有构造函数
            try {
                cons = clazz.getConstructor(cl);
            } catch (NoSuchMethodException e) {
                e.printStackTrace();
            }

            if (cons == null) {
                return false;
            }
            //给传入参数赋初值
            Object[] x = {inetAddr, prefixLength};
            //构造 StaticIpConfiguration 对象
            Class<?> staticIpConfigurationCls = Class.forName("android.net.StaticIpConfiguration");
            //实例化 StaticIpConfiguration
            Object staticIpConfiguration = null;

            staticIpConfiguration = staticIpConfigurationCls.newInstance();
            Field ipAddress = staticIpConfigurationCls.getField("ipAddress");
            Field gateWay = staticIpConfigurationCls.getField("gateway");
            Field dnsServers = staticIpConfigurationCls.getField("dnsServers");
            //设置 ipAddress
            ipAddress.set(staticIpConfiguration, (LinkAddress) cons.newInstance(x));
            //设置网关
            gateWay.set(staticIpConfiguration, gatewayAddr);
            //设置 dns
            ArrayList<InetAddress> dnsList = (ArrayList<InetAddress>) dnsServers.get(staticIpConfiguration);
            dnsList.add(dnsAddr1);
            dnsList.add(dnsAddr2);
            @SuppressLint("PrivateApi") Class ipAssignmentCls =
Class.forName("android.net.IpConfiguration$IpAssignment");
            Object ipAssignment = null;
            ipAssignment = Enum.valueOf(ipAssignmentCls, "STATIC");
            Method setIpAssignmentMethod = wifiConfig.getClass().getDeclaredMethod("setIpAssignment",
ipAssignmentCls);
```

```
            setIpAssignmentMethod.invoke(wifiConfig, ipAssignment);
            Method setStaticIpConfigurationMethod =
wifiConfig.getClass().getDeclaredMethod("setStaticIpConfiguration", staticIpConfiguration.getClass());
            //设置静态 IP，将 StaticIpConfiguration 设置给 WifiConfiguration
            setStaticIpConfigurationMethod.invoke(wifiConfig, staticIpConfiguration);
            //WifiConfiguration 重新添加到 WifiManager
            int netId = mWifiManager.addNetwork(wifiConfig);
            mWifiManager.disableNetwork(netId);
            return mWifiManager.enableNetwork(netId, true);
        } catch (NoSuchFieldException | IllegalAccessException | InstantiationException |
InvocationTargetException | ClassNotFoundException | NoSuchMethodException e) {
            e.printStackTrace();
        }
        return false;
    }

    /**
     * 已连接 wifi 配置 DHCP Wifi
     * @return 是否设置成功
     */
    public boolean setDhcpWifi() throws Exception{
        List<WifiConfiguration> configuredNetworks = getConfiguredNetworks();
        WifiConfiguration wifiConfig = null;
        WifiInfo connectionInfo = mWifiManager.getConnectionInfo();   //得到连接的 wifi 网络
        for (WifiConfiguration conf : configuredNetworks) {
            if (conf.networkId == connectionInfo.getNetworkId()) {
                wifiConfig = conf;
                break;
            }
        }
        Class ipAssignmentCls = Class.forName("android.net.IpConfiguration$IpAssignment");
        Object ipAssignment = null;
        ipAssignment = Enum.valueOf(ipAssignmentCls, "DHCP");
        Method setIpAssignmentMethod = wifiConfig.getClass().getDeclaredMethod("setIpAssignment",
ipAssignmentCls);
        setIpAssignmentMethod.invoke(wifiConfig, ipAssignment);
        int netId = mWifiManager.addNetwork(wifiConfig);
        mWifiManager.disableNetwork(netId);
        return mWifiManager.enableNetwork(netId, true);
    }
```

4.6.13 设置屏保遮罩

在产品实际使用过程中，如果液晶屏长时间以高亮的状态运行，将会减少液晶屏的寿命，有时甚至会出现残影，建议为应用添加黑色保护遮罩，在应用空闲时降低屏幕亮度，可以延长液晶屏使用的寿命。

以下是操作显示遮罩的变量

```
//需要使用的变量
private WindowManager mWindowManager = null;
private WindowManager.LayoutParams mNightViewParam;
private View mNightView = null;
private boolean mIsAddedView;
```

添加以下方法

```
//需要使用的方法
    /**
     * 设置黑色遮罩
     */
    private void changeToNight() {
        if (mIsAddedView == true)
            return;
        mNightViewParam = new WindowManager.LayoutParams(
                WindowManager.LayoutParams.TYPE_APPLICATION,
                WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE |
                        WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE,
                PixelFormat.TRANSPARENT);
        mWindowManager = getWindowManager();
        mNightView = new View(this);
        mNightView.setBackgroundResource(R.color.black);
        mWindowManager.addView(mNightView, mNightViewParam);
        mIsAddedView = true;
    }
    /**
     * 取消屏幕黑色遮罩
     */
    public void changeToDay() {
        if (mIsAddedView && mNightView != null) {
            mWindowManager.removeViewImmediate(mNightView);
            mWindowManager = null;
            mNightView = null;
            mIsAddedView = false;
        }

    }
```

使用方法方法

```
//切换到黑色遮罩
changeToNight();
//切换到正常显示
changeToDay();
```

4.6.14 设置应用开机自启动

应用开机自启动可以提高用户体验满意度，以下是应用开机自启动的使用方法。

在 AndroidManifest.xml 文件中配置监听启动权限，建立一个监听广播接收者 BootReceiver

添加以下权限

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

添加 Receiver

```
<receiver
    android:name=".BootReceiver"
    tools:ignore="Instantiatable">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.ACTION_SHUTDOWN"/>
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</receiver>
```

创建监听启动完成的广播接收者

```
public class BootReceiver extends BroadcastReceiver {

    private SharedPreferences pref;
    private boolean autoStarts = false;
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {
            // boot
            pref = context.getSharedPreferences("BOOT_COMPLETED", Context.MODE_PRIVATE);
            autoStarts = pref.getBoolean("Autostarts",false);
            if (autoStarts){
                Log.e("Autostarts","开机自动启动");
```

```
                Intent intent2 = new Intent(context, MainActivity.class);
                intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(intent2);
            }else {
                Log.e("Autostarts","开机不自动启动");
            }

        }
        if (intent.getAction().equals("android.intent.action.ACTION_SHUTDOWN")){
            Log.e("shutdown","关机了");
        }
    }
}
```

重启安卓模组，测试 APP 是否自启动了

⚠ 注意：安装 app 到模组上，然后启动一次程序（安卓 4.0 以后，必须先启动一次程序才能接收到开机完成的广播，目的是防止恶意程序）

检查一下是不是安装了 360 等安全助手之类的软件，如果有，请在软件的自启动软件管理中将 app 设置为允许

app 安装到内部存储，不可以安装在 SD 卡上

4.6.15 设置 WIFI ADB 调试

以命令行方式执行

```
//设置端口 5555

setprop service.adb.tcp.port 5555

//打开 WIFI 调试

start adbd

//关闭 WIFI 调试

stop adbd


//adb 连接设备

adb connect ip:5555
```

以代码方式执行

```
//这个方法用来执行命令
public void execShell(String cmd){
    try{
        //权限设置
        Process p = Runtime.getRuntime().exec("su");
        //获取输出流
        OutputStream outputStream = p.getOutputStream();
        DataOutputStream dataOutputStream=new DataOutputStream(outputStream);
        //将命令写入
        dataOutputStream.writeBytes(cmd);
        //提交命令
        dataOutputStream.flush();
        //关闭流操作
        dataOutputStream.close();
        outputStream.close();
    }
    catch(Throwable t)
    {
        t.printStackTrace();
    }
}

_____

开启 WIFI ADB 调试

execShell("setprop service.adb.tcp.port 5555");

execShell("start adbd");
```

提示：附上三种查看 ip 地址的方法：

1.设置－关于手机－状态信息－IP 地址中查看

2.设置－WLAN-点击当前链接上的 Wi-Fi 查看 IP 例如：设置—〉无线或网络---〉WLAN 设置—〉查看当前连接 Wi-Fi 的 IP 地址

3.通过 adb 命令查看设备 IP 地址： adb shell netcfg

4.6.16 禁用 USB 根节点

查询 USB 根节点名称

```
adb shell

su

cd /sys/bus/usb/devices

ls
```

禁用对应 USB 根节点,重启失效

```
adb shell

su

//禁用 usb1

echo 'usb1' > /sys/bus/usb/drivers/usb/unbind

//使用 usb1

echo 'usb1' > /sys/bus/usb/drivers/usb/bind
```

4.6.17 静默安装与自启动

静默安装是在用户无感知的情况下进行安装，无需用户手动点击。原理是调用 pm install -r 来安装的，-r 是保留原来 APP 的数据。

添加方法

```
protected void executeInstall(String path) {
    Process process = null;
```

```
    OutputStream out = null;
    InputStream in = null;
    try {
        // 请求 root
        process = Runtime.getRuntime().exec("su");
        out = process.getOutputStream();
        // 调用安装
        out.write(("pm install -r " + path + "\n").getBytes());
        in = process.getInputStream();
        int len = 0;
        byte[] bs = new byte[256];
        while (-1 != (len = in.read(bs))) {
            String state = new String(bs, 0, len);
            if (state.equals("success\n")) {
                Log.d("SUC","success");
                //安装成功后的操作
                //静态注册自启动广播
                Intent intent=new Intent();
                //与清单文件的 receiver 的 anction 对应
                intent.setAction("android.intent.action.PACKAGE_REPLACED");
                //发送广播
                sendBroadcast(intent);

            }
        }
    }catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (out != null) {
                out.flush();
                out.close();
            }
            if (in != null) {
                in.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

添加 Receiver

```
<receiver android:name=".Utils.UpdateRestartReceiver"
    >
    <intent-filter>
        <action android:name="android.intent.action.PACKAGE_REPLACED"/>
        <data android:scheme="package"/>
    </intent-filter>

</receiver>
```

创建监听启动完成的广播接收者

```
public class UpdateRestartReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("android.intent.action.PACKAGE_REPLACED")){
            //Toast.makeText(context,"已升级到新版本",Toast.LENGTH_SHORT).show();
            Intent intent2 = new Intent(context, LoginActivity.class);
            intent2.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(intent2);

        }
    }
}
```

使用方法

```
executeInstall(文件路径);
```

4.6.18 获取模组系统信息

```
/**
 * </br>
 * <p>使用该类中的接口方法必须先调用 {@link ISystemInfo#getInstance(Context)} (String)} 获取实例。
</p>
 * <p>结束使用调用 {@link ISystemInfo#destroyInstance()} 销毁单例。</p>
 *  接口功能：
 * </br>
 * <ol>
 * <li>{@link ISystemInfo#getBrand()} 获取手机厂商名称</li>
 * <li>{@link ISystemInfo#getProduct()} 获取到对用户有意义的手机厂商名称</li>
 * <li>{@link ISystemInfo#getBoard()} 主板型号</li>
 * <li>{@link ISystemInfo#getCpuABI()} 通过这个字段可以获取设备指令集名称（CPU 的类型）</li>
 * <li>{@link ISystemInfo#getModel()}  通过这个字段可以获取到型号</li>
```

```
 * <li>{@link ISystemInfo#getSerial()}     通过这个字段可以获取到序列号</li>
 * <li>{@link ISystemInfo#getTelephonyState()} 获取移动数据连接状态</li>
 * <li>{@link ISystemInfo#getTelephonyDeviceId()} 获取移动设备 id</li>
 * <li>{@link ISystemInfo#getTelephonyNumber()}   获取手机号码</li>
 * <li>{@link ISystemInfo#getTelephonyNetworkOperatorName()} 获取运营商名称</li>
 * <li>{@link ISystemInfo#getTelephonyNetworkType()} 获取网络类型</li>
 * <li>{@link ISystemInfo#getLocalIpAddress()}   获取本地 ip 地址</li>
 * <li>{@link ISystemInfo#getAvailMemory(Context)} 获取 android 当前可用运行内存大小</li>
 * <li>{@link ISystemInfo#getTotalMemory(Context)}   获取 android 当前总运行内存大小</li>
 * <li>{@link ISystemInfo#getInternalStorageTotal()}   获取内部存储总量</li>
 * <li>{@link ISystemInfo#getInternalStorageRemain()} ()}     获取内部存储剩余</li>
 * <li>{@link ISystemInfo#getExternalStorageTotal()}       获取外部存储总量</li>
 * <li>{@link ISystemInfo#getExternalStorageRemain()} ()}       获取外部存储余量</li>
 * </ol>
 *
 */
public class ISystemInfo {
    private static Context mContext;
    private static ISystemInfo mInstance;
    private static TelephonyManager telephonyManager;

    public ISystemInfo(Context context) {
        mContext = context;
        telephonyManager = (TelephonyManager)
mContext.getSystemService(Context.TELEPHONY_SERVICE);
    }

    /**
     * 获取系统相关信息单例
     *
     * @param context 上下文对象
     * @return ISystemInfo 单例
     */
    public static ISystemInfo getInstance(Context context) {
        if (mInstance == null) {
            synchronized (ISystemInfo.class) {
                if (mInstance == null) {
                    mInstance = new ISystemInfo(context);
                }
            }
        }
        return mInstance;
    }
    /**
```

```
     * 销毁控制单例
     */
    public static void destroyInstance() {
        if (mInstance != null) {
            mInstance = null;
        }
    }
    /**
     * 通过这个字段可以获取到对用户有意义的手机厂商名称，例如 Xiaomi，Meizu，Huawei 等。
     *
     * @return brand
     */
    public String getBrand() {
        return Build.BRAND;
    }

    /**
     * 通过这个字段可以获取到对用户有意义的手机厂商名称，例如 Xiaomi，Meizu，Huawei 等。
     *
     * @return brand
     */
    public String getProduct() {
        return Build.PRODUCT;
    }

    /**
     * 通过这个字段可以获取到主板型号
     *
     * @return brand
     */
    public String getBoard() {
        return Build.BOARD;
    }

    /**
     * 通过这个字段可以获取设备指令集名称（CPU 的类型）
     *
     * @return brand
     */
    public String getCpuABI() {
        return Build.CPU_ABI;
    }

    /**
```

```
     * 通过这个字段可以获取到型号
     *
     * @return brand
     */
    public String getModel() {
        return Build.MODEL;
    }

    /**
     * 通过这个字段可以获取到序列号
     *
     * @return brand
     */
    public String getSerial() {
        return Build.SERIAL;
    }

    /**
     * 获取移动数据连接状态
     * <p>
     * DATA_CONNECTED = 2  数据连接状态：已连接
     * DATA_CONNECTING = 1  数据连接状态：正在连接
     * DATA_DISCONNECTED = 0  数据连接状态：断开
     * DATA_SUSPENDED = 3  数据连接状态：暂停
     */
    public int getTelephonyState() {
        if (telephonyManager != null) {
            telephonyManager.getDataState();
        }
        return -1;
    }

    /**
     * 返回唯一的设备 ID
     * 如果是 GSM 网络，返回 IMEI；如果是 CDMA 网络，返回 MEID；如果设备 ID 是不可用的返回 null
     */
    public String getTelephonyDeviceId() {
        if (telephonyManager != null) {
            telephonyManager.getDeviceId();
        }
        return "";
    }

    /**
```

```
     * 返回手机号码
     * 对于 GSM 网络来说即 MSISDN，如果不可用返回 null
     */
    @SuppressLint("MissingPermission")
    public String getTelephonyNumber() {
        if (telephonyManager != null) {
            telephonyManager.getLine1Number();
        }
        return "";
    }

    /**
     * 返回移动网络运营商的名字(SPN)
     */
    public String getTelephonyNetworkOperatorName() {
        if (telephonyManager != null) {
            telephonyManager.getNetworkOperatorName();
        }
        return "";
    }

    /**
     * 获取网络类型
     * <p>
     * NETWORK_TYPE_CDMA = 4  网络类型为 CDMA
     * NETWORK_TYPE_EDGE = 2  网络类型为 EDGE
     * NETWORK_TYPE_EVDO_0 = 5  网络类型为 EVDO0
     * NETWORK_TYPE_EVDO_A = 6  网络类型为 EVDOA
     * NETWORK_TYPE_GPRS = 1  网络类型为 GPRS
     * NETWORK_TYPE_HSDPA = 8  网络类型为 HSDPA
     * NETWORK_TYPE_HSPA = 10  网络类型为 HSPA
     * NETWORK_TYPE_HSUPA = 9  网络类型为 HSUPA
     * NETWORK_TYPE_UMTS = 3  网络类型为 UMTS
     * <p>
     * 在中国，联通的 3G 为 UMTS 或 HSDPA，移动和联通的 2G 为 GPRS 或 EGDE，电信的 2G 为
CDMA，电信的 3G 为 EVDO
     */

    @SuppressLint("MissingPermission")
    public int getTelephonyNetworkType() {
        if (telephonyManager != null) {
            telephonyManager.getNetworkType();
        }
        return -1;
```

```
}

/**
 * 获取系统版本
 *
 * @return 系统版本
 */
public String getSystemVersion() {
    return Build.VERSION.RELEASE;
}

/**
 * 获取当前系统语言和地区
 *
 * @return 系统语言和地区 如：“zh_rCN”
 */
public String getSystemLanguageAndCountry() {
    return Locale.getDefault().toString();
}

/**
 * 获取当前系统语言
 *
 * @return 系统语言 如：“zh”
 */
public static String getSystemLanguage() {
    return Locale.getDefault().getLanguage();
}

/**
 * 获取当前系统语言地区
 *
 * @return 系统语言地区 如：“CN”
 */
public String getSystemCountry() {
    return Locale.getDefault().getCountry();
}

/**
 * 获取当前 CPU 温度
 *
 * @return float 温度 如：“52.1” 为 52.1°C
 */
public float getCurrentCPUTemperature() {
```

```
        String file = readFile("/sys/devices/virtual/thermal/thermal_zone0/temp", '\n');
        if (file != null) {
            if(Build.VERSION.SDK_INT > Build.VERSION_CODES.M) {
                return Long.parseLong(file)/1000;
            }
            return Long.parseLong(file)/100;
        } else {
            return Long.parseLong("0");

        }
    }

    private byte[] mBuffer = new byte[4096];
    @SuppressLint("NewApi")
    private String readFile(String file, char endChar) {
        StrictMode.ThreadPolicy savedPolicy = StrictMode.allowThreadDiskReads();
        FileInputStream is = null;
        try {
            is = new FileInputStream(file);
            int len = is.read(mBuffer);
            is.close();

            if (len > 0) {
                int i;
                for (i = 0; i < len; i++) {
                    if (mBuffer[i] == endChar) {
                        break;
                    }
                }
                return new String(mBuffer, 0, i);
            }
        } catch (java.io.FileNotFoundException e) {
        } catch (java.io.IOException e) {
        } finally {
            if (is != null) {
                try {
                    is.close();
                } catch (java.io.IOException e) {
                }
            }
            StrictMode.setThreadPolicy(savedPolicy);
        }
        return null;
    }
```

```
/**
 * 获取外部存储总量
 *
 * @return float “1024” 代表 1024MB
 */
public float getExternalStorageTotal() {
    String state = Environment.getExternalStorageState();
    if(Environment.MEDIA_MOUNTED.equals(state)) {
        File sdcardDir = Environment.getExternalStorageDirectory();
        StatFs sf = new StatFs(sdcardDir.getPath());
        long blockSize = sf.getBlockSize();
        long blockCount = sf.getBlockCount();
        return blockSize*blockCount/1024/1024;
    }
    return 0;
}

/**
 * 获取外部存储剩余量
 *
 * @return float “1024” 代表剩余 1024MB
 */
public float getExternalStorageRemain() {
    String state = Environment.getExternalStorageState();
    if(Environment.MEDIA_MOUNTED.equals(state)) {
        File sdcardDir = Environment.getExternalStorageDirectory();
        StatFs sf = new StatFs(sdcardDir.getPath());
        long blockSize = sf.getBlockSize();
        long blockCount = sf.getBlockCount();
        long availCount = sf.getAvailableBlocks();
        return  availCount*blockSize/1024/1024;
    }
    return 0;
}

/**
 * 获取内部存储剩总量
 *
 * @return float “1024” 内存总量 1024MB
 */
public float getInternalStorageTotal() {
    File root = Environment.getRootDirectory();
    StatFs sf = new StatFs(root.getPath());
```

```
        long blockSize = sf.getBlockSize();
        long blockCount = sf.getBlockCount();
        long availCount = sf.getAvailableBlocks();
        return blockSize*blockCount/1024/1024;
    }

    /**
     * 获取内部存储剩余量
     *
     * @return float “1024” 内存余量 1024MB
     */
    public float getInternalStorageRemain() {
        File root = Environment.getRootDirectory();
        StatFs sf = new StatFs(root.getPath());
        long blockSize = sf.getBlockSize();
        long blockCount = sf.getBlockCount();
        long availCount = sf.getAvailableBlocks();
        return availCount*blockSize/1024/1024;
    }

    /***
     * 获取本地 IP 地址
     * @return 示例 “192.168.1.135”
     */
    public String getLocalIpAddress() {
        String ip = "null";
        ConnectivityManager conMann = (ConnectivityManager)
                mContext.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo mobileNetworkInfo = conMann.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
        NetworkInfo wifiNetworkInfo = conMann.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
        if (mobileNetworkInfo.isConnected()) {
            ip = getIpAddress();
        }else if(wifiNetworkInfo.isConnected())
        {
            WifiManager wifiManager = (WifiManager)
mContext.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
            WifiInfo wifiInfo = wifiManager.getConnectionInfo();
            int ipAddress = wifiInfo.getIpAddress();
            ip = intToIp(ipAddress);
        }
        return ip;
    }
    private static String intToIp(int ipInt) {
        StringBuilder sb = new StringBuilder();
```

```
        sb.append(ipInt & 0xFF).append(".");
        sb.append((ipInt >> 8) & 0xFF).append(".");
        sb.append((ipInt >> 16) & 0xFF).append(".");
        sb.append((ipInt >> 24) & 0xFF);
        return sb.toString();
    }
    /**
     * 取得 IP 地址
     * @return
     */
    private String getIpAddress() {
        try {
            for (Enumeration<NetworkInterface> en = NetworkInterface
                    .getNetworkInterfaces(); en.hasMoreElements();) {
                NetworkInterface intf = en.nextElement();
                for (Enumeration<InetAddress> enumIpAddr = intf
                        .getInetAddresses(); enumIpAddr.hasMoreElements();) {
                    InetAddress inetAddress = enumIpAddr.nextElement();
                    if (!inetAddress.isLoopbackAddress()
                            && inetAddress instanceof Inet4Address) {
                        return inetAddress.getHostAddress().toString();
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    /**
     * 获取 android 当前可用运行内存大小
     * @param context
     *
     */
    public String getAvailMemory(Context context) {
        ActivityManager am = (ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE);
        ActivityManager.MemoryInfo mi = new ActivityManager.MemoryInfo();
        am.getMemoryInfo(mi);
        // mi.availMem; 当前系统的可用内存
        return String.format("%.2f", (float) mi.availMem/1024/1024);
    }

    /**
```

```
     *     * 获取 android 总运行内存大小
     *     * @param context
     *     *
     */
    public String getTotalMemory(Context context) {
        String str1 = "/proc/meminfo";// 系统内存信息文件
        String str2;
        String[] arrayOfString;
        float initial_memory = 0;
        try {
            FileReader localFileReader = new FileReader(str1);
            BufferedReader localBufferedReader = new BufferedReader(localFileReader, 8192);
            str2 = localBufferedReader.readLine();// 读取 meminfo 第一行，系统总内存大小
            arrayOfString = str2.split("\\s+");
            // 获得系统总内存，单位是 KB
            initial_memory = Integer.valueOf(arrayOfString[1]).intValue();
            localBufferedReader.close();
        } catch (IOException e) {
        }
        return String.format("%.2f", initial_memory/1024);
    }
    /**
     * 获取手机序列号
     *
     * @return 手机序列号
     */
    @SuppressLint({"NewApi", "MissingPermission"})
    public String getSerialNumber() {
        String serial = "";
        try {
            if (Build.VERSION.SDK_INT > Build.VERSION_CODES.N) {//8.0+
                serial = Build.SERIAL;
            } else {//8.0-
                Class<?> c = Class.forName("android.os.SystemProperties");
                Method get = c.getMethod("get", String.class);
                serial = (String) get.invoke(c, "ro.serialno");
            }
        } catch (Exception e) {
            e.printStackTrace();
            Log.e("e", "读取设备序列号异常：" + e.toString());
        }
        return serial;
    }
}
```

第五章 外设选择与功能支持

5.1 外设配件

超显模组可支持丰富的外设配件，您可以在超显科技官方淘宝店配件栏目中选择您需要的配件。

功能	配置	详情图
GPS	若需要使用 GPS 定位功能，需要购买模组时和与您对接的销售人员沟通选购 4G 功能，增加 4G 功能后只需要购买 GPS 模块与 ipex 转 sma 母头连接线端子接通后即可使用 GPS 定位。	
音频输出	7 寸模组使用 4Ω3W 双输出音频接口。端头为 PH2.0 MM 端头。 10.1 寸模组使用 4Ω3W 双输出音频接口。端头为 XH2.54 MM 端头。	
HDMI	超显模组支持 HDMI 2.0，若需使用 HDMI 请购买 HDMI 2.0 适配线	
电源	推荐使用 12V 2A 工作电源适配器电（详细最大电压支持请参阅安卓产品数据书）	

外部存储	软件长时间读写运行，非正规内存卡可能会导致数据丢失或损毁，推荐使用品牌内存卡。	
摄像头	USB2.0 免驱，高清摄像头，适合人脸识别开发	
摄像头	USB2.0 免驱，高清红外摄像头，适合人脸识别开发	
4G	支持 4g 上网功能，下列模块可选择： 1- N58：有方通信模块，支持 GPS，低成本 CAT1 方案 2- EC20：移远通信模块，信号更稳定，CAT4. 3- EC25：移远通信模块，支持海外全频段，CAT4。	
USBOTG	超显模组支持 Micro USB 接口，使用 Micro USB 数据线进行调试。若使用数据线电脑识别不到模组，请尝试安装 Android 驱动。	
打印机	超显模组的安卓系统支持市面主流打印机，可以在安卓市场安装打印机厂商开发的打印应用，也可安装趣打印 PrinterShare 应用进行打印操作。	无图片

第六章 模组使用注意事项

6.1 注意事项

1.勿带电插拔核心板以及外围模块

2.请遵循所有标注在产品上的警示和指引信息

3.请保持本产品干燥。如果不慎被任何液体泼溅或浸润，请立刻断电并充分晾干

4.使用中注意本产品的通风散热，避免温度过高造成元器件损坏

5.请勿在多尘、脏乱的环境中使用或存放本产品

6.请勿将本产品应用在冷热交替环境中，避免损坏元器件

7.请勿粗暴对待本产品，跌落、敲打或剧烈晃动都可能损坏线路及元器件

8.请勿使用有机溶剂或腐蚀性液体清洗本产品

9.请勿自行修理、拆卸本公司产品，如产品出现故障请及时联系本公司进行维修

10. 擅自修改或使用未经授权的配件可能损坏本产品，由此造成的损坏将不予以保修

11. 若液晶屏以最高亮度连续工作，LCD 背光生命周期将会减半；如果超过 30 分钟及以上时间高对比度静止显示可能会引起液晶屏残影，建议增加屏保避免该问题

6.2 串口使用注意事项

注意事项一：路径

因超显安卓屏有多个串口，所以务必要确定连接串口的路径。

注意事项二：波特率

作为传输频率，波特率非常重要，如果不一致会出现乱码。在测试中如果可以收到消息但无法正常工作，大部分是由于串口波特率不正确所致。

注意事项三：乱码

乱码通常是因为收发不一致导致的，即请检查发送和接收端的波特率、校验位、停止位等是否一致，数据格式是否一致（HEX 或 ASCII）。

注意事项四：数据收发不正常

1. 优先排除串口接线问题，排除因产品接线不良导致无法通讯。

2. 在排除 1 的情况下，通过串口测试软件检测串口的工作性能。

3. 在排除 1 的情况下，若使用 usb 转串口，建议检测工具是否损坏。

6.3 USB Device 常见问题

USB device 一般为联网自动识别，无网情况下需要确认。PC 的 USB 接口连接安卓屏的 DEBUG/OTG 接口后，安卓屏可自动识别。

常见问题一：当前电脑上插入 USB 不能过多，如果插入过多，会导致连接异常。

常见问题二：上电后电脑的设备管理器会自动刷新弹出新的设备进行安装，如果设备管理器没有刷新，请检查 USB 连接的问题。

常见问题三：如果弹出提示框，选择安装驱动时，直接自动搜索驱动即可。

常见问题四：安装完成后，设备管理器中会多出一个 Android Tablet。如显示感叹号，请卸掉该驱动重新插入。

6.4 产品连接故障

解决措施一：驱动错误，需重新安装驱动；

解决措施二：重新拔插所有接口；

解决措施三：重启安卓屏、电脑、豌豆荚、手机助手等；

解决措施四：PC 机上至多只能连一台 android 设备；

6.5 其他问题

1. 当您发现产品出现黑屏、白屏的故障，或触摸屏反应迟钝的现象，以及无法进入主界面的故障，请参照章节 2.1.4 固件烧录程序，重新烧录安卓产品系统。如故障无法解决，请联系与您对接的销售工程师进行返厂返修处理。

2. 如产品出现花屏的故障，请重新插拔一下液晶屏连接底板排线，或联系与您对接的销售工程师进行返厂返修处理。花屏通常由于排线松动或老化所致。

3. 如您发现触摸屏有断点现象（部分列或行无法点击），请进入设置-开发者选项-选择指针位置，确认触点是否正常。如果断点现象出现，请联系与您对接的销售工程师进行返厂返

修处理。