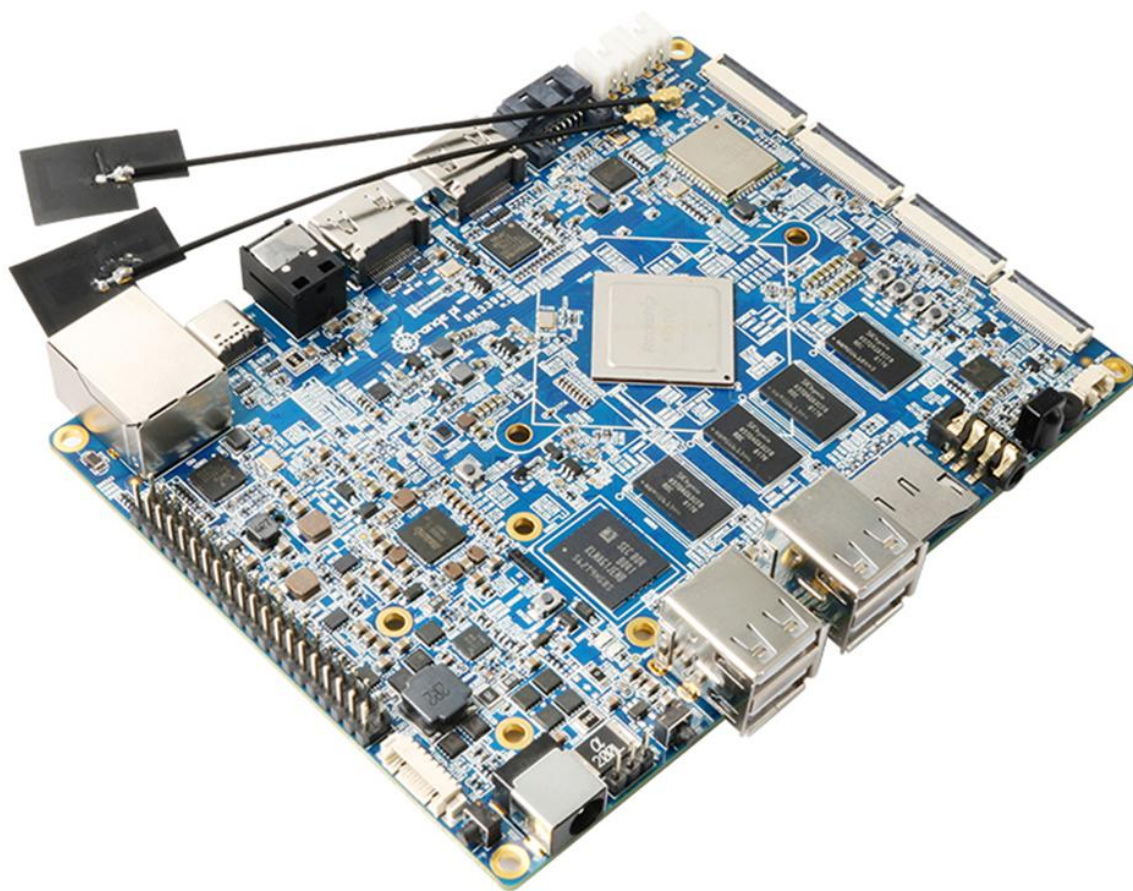


Orange Pi RK3399 用户手册



History

| Ver | Data | Author | Brief | Publish | Mem |
|-----|------------|--------|--------------------------|------------|-----|
| 1.1 | 2017-07-14 | Younix | 创建文档 | 2018-1-31 | |
| 1.2 | 2018-02-01 | Younix | 更新 Android 编译环境搭建三（3）3 | 2018-3-6 | |
| 1.3 | 2018-03-19 | Younix | 更新固件烧录问题汇总四（3） | 2018-3-19 | |
| 1.4 | 2018-06-05 | pmm | 更新 Linux 固件烧录方式和串口调试工具介绍 | 2018-6-5 | |
| 1.5 | 2018-10-30 | pmm | 更新 HDMI IN 的使用 | 2018-10-30 | |
| 1.6 | 2019-04-28 | baiywt | 更新第五章，添加第八章 | 2019-05-10 | |
| | | | | | |
| | | | | | |
| | | | | | |

目录

| | |
|-----------------------------------|-----------|
| 一、OrangePi RK3399 的基本特性 | 4 |
| 1、什么是 OrangePi RK3399 | 4 |
| 2、OrangePi RK3399 的用途 | 4 |
| 3、OrangePi RK3399 是为谁设计的 | 4 |
| 4、OrangePi RK3399 的硬件特征 | 4 |
| 5、GPIO 规格 | 7 |
| 二、开发板使用说明 | 9 |
| 1、准备硬件与软件工具 | 9 |
| 2、开发板的供电方式 | 9 |
| 三、Android 编译环境搭建 | 10 |
| 1、获取 SDK 源码压缩包 | 10 |
| 2、搭建编译环境 | 10 |
| 3、编译 SDK 源码 | 11 |
| 4、生成固件 | 11 |
| 四、Android 固件烧录 | 13 |
| 1、Windows 下进行镜像烧录 | 14 |
| 2、Linux 下镜像烧录 | 17 |
| 3、固件烧录问题汇总 | 20 |
| 五、Linux 编译环境搭建及 Linux 固件烧录 | 23 |
| 1、获取 linux 源码 | 23 |
| 2、编译 Linux 源码 | 25 |
| 3、linux 固件烧录 | 27 |
| 六、搭建编译环境并制作 Rootfs 镜像 | 30 |
| 1、搭建编译环境 | 30 |
| 2、获取 Linux Rootfs 源码包 | 30 |
| 3、修改 Rootfs 并添加定制软件 | 30 |
| 4、制作 Rootfs 镜像 | 31 |
| 七、串口调试工具介绍 | 32 |
| 1、基于 Windows 平台的使用 | 33 |
| 2、基于 Linux 平台的使用 | 36 |
| 八、Linux 系统使用说明 | 37 |
| 1、登录账号和密码 | 37 |
| 2、扩展 rootfs 分区 | 37 |
| 3、GPU 性能测试 | 37 |
| 4、编解码 | 37 |
| 5、使用 HDMI IN 功能 | 38 |
| 6、ov13850 摄像头的使用 | 38 |
| 7、GPIO 的使用 | 38 |

一、OrangePi RK3399 的基本特性

1、什么是 Orange Pi RK3399

香橙派是一款开源的单板卡片电脑，新一代的 ARM 开发板，它可以运行 Android6.0、Ubuntu、Debian 等操作系统。香橙派开发板（RK3399）使用瑞芯微 RK3399 服务器级芯片，同时拥有 2GB DDR3 内存。

2、OrangePi RK3399 的用途

我们可以用它搭建：

- 一台计算机
- 一个无线网络服务器
- 游戏机
- 音乐播放器
- 高清视频播放器
- 扬声器
- Android
- Scratch
-

还有更多的其他功能，因为 Orange Pi RK3399 是开源的。

3、OrangePi RK3399 是为谁设计的

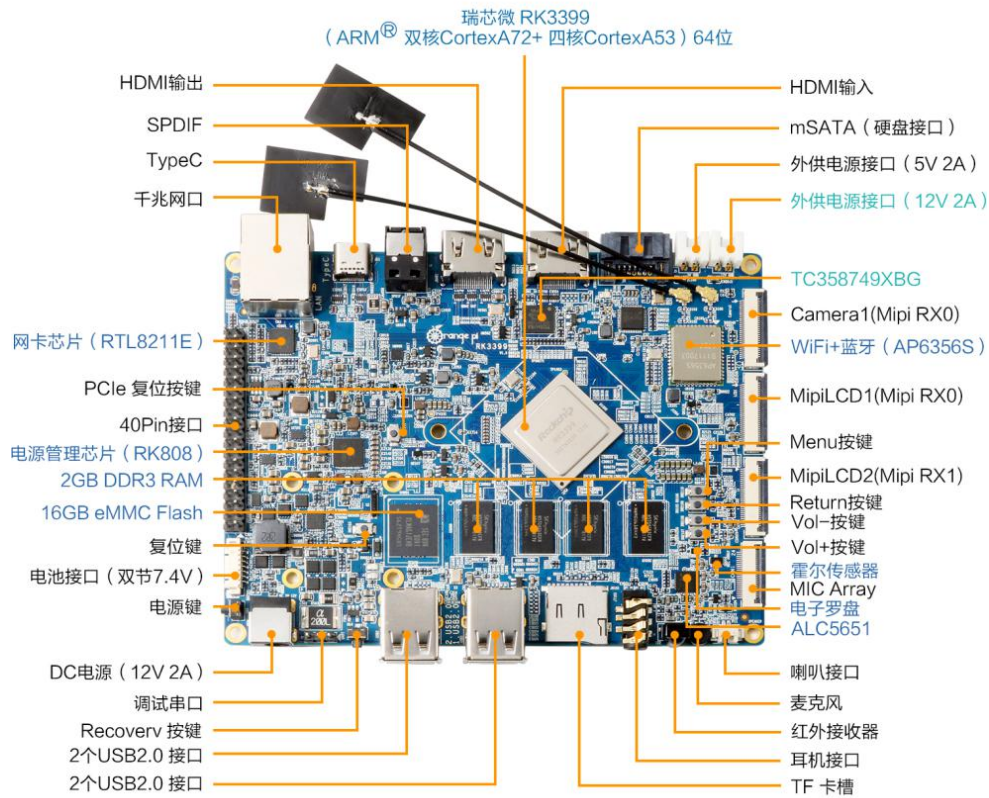
Orange Pi RK3399 不仅仅是一款消费品，同时也是给任何想用技术进行创作创新的人设计的。它是一款非常简单、有趣、实用的工具，你可以用它去打造你身边的世界。

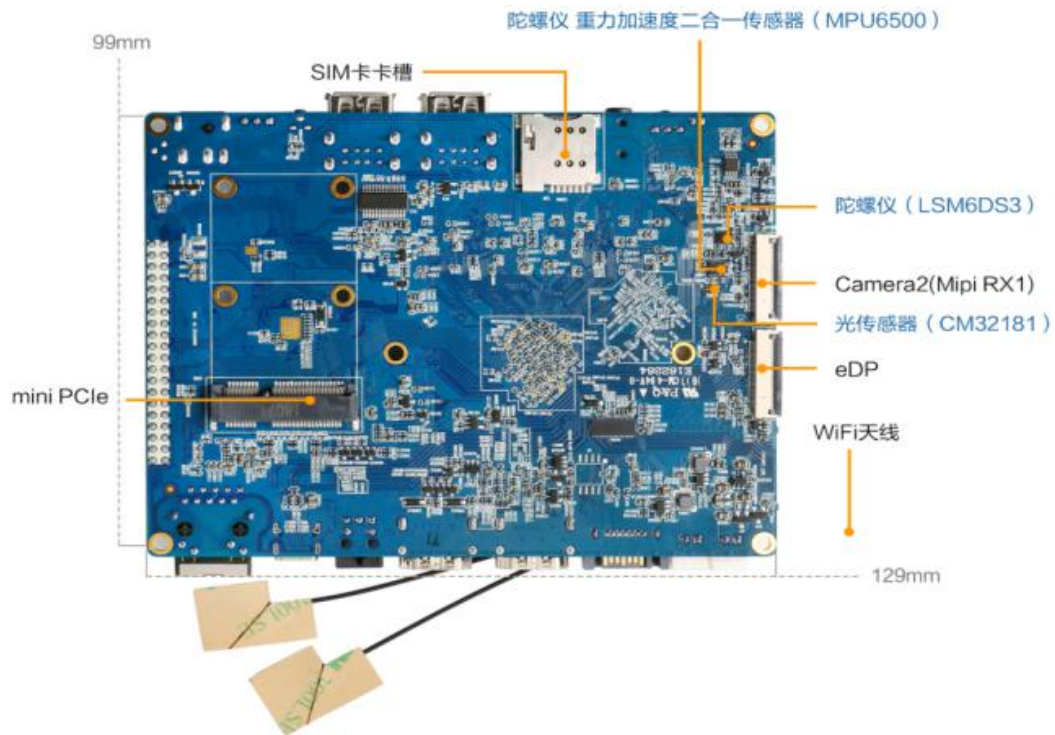
4、OrangePi RK3399 的硬件特征

| 硬件特性介绍 | |
|--------|--|
| 主控芯片 | Rockchip RK3399（28 纳米 HKMG 制程） |
| 处理器 | 6 核 ARM® 64 位处理器，主频高达 2.0GHz 双核 Cortex-A72(大核)+四核 Cortex-A53(小核) |
| 图形处理器 | ARM Mali-T860 MP4 四核 GPU 支持 OpenGL ES1.1/2.0/3.0/3.1, OpenVG1.1, OpenCL, DX11 支持 AFBC（帧缓冲压缩） |

| | |
|-------|---|
| 视频处理器 | 支持 4K VP9 and 4K 10bits H265/H264 视频解码，高达 60fps 1080P 多格式视频解码（WMV，MPEG-1/2/4，VP8） 1080P 视频编码，支持 H.264，VP8 格式 视频后期处理器：反交错、去噪、边缘/细节/色彩优化 |
| 电源管理 | RK808 PMU 芯片 BQ25700 Charger IC 充电管理 IC CW2015 Fuel Gas 电量计 |
| 内存 | 2GB DDR3 |
| 存储器 | 16GB 高速 eMMC MicroSD (TF)卡槽 miniPCIe 接口（兼容 USB2.0 以及 mSATA 硬盘） mSATA 接口 |
| 无线网络 | 板载 WiFi 模块（AP6356S）： 2.4GHz/5GHz 双频 WiFi，支持 802.11a/b/g/n/ac 协议，2x2 MIMO 技术 Bluetooth 4.1（支持 BLE） |
| 以太网 | 10/100/1000Mbps 以太网（Realtek RTL8211E） |
| 显示 | 1 x HDMI 2.0（Type-A），支持 4K@60 帧输出 1 x DP 1.2（DisplayPort），支持 4K@60 帧输出 2 x MIPI，支持双通道 2560x1600@60 帧输出 1 x eDP 1.3（4 lanes with 10.8Gbps） 1 x HDMI IN 功能 |
| 音频 | 1 x HDMI 或 1 x DP（DispalyPort），音频输出 1 x 耳麦，用于音频输入输出 1 x SPEAKER，喇叭输出（1.5W 8Ω/2.5W 4Ω） 1 x SPDIF 数字音频接口，用于音频输出 1 x 麦克风，板载音频输入 1 x I2S，支持 8 通道 1 x MIC 阵列接口 |
| 摄像头 | 2 x MIPI-CSI 摄像头接口（每个接口最高支持 13Mpixel） 支持 USB Camera |
| 传感器 | 1 x 陀螺仪+重力加速度计，1 x 陀螺仪，1 x 霍尔元件，1 x 光传感器， 1 x 电子罗盘 |
| PCIe | 1 x mini PCIe，支持 LTE，兼容 USB 用于拓展 TF Card，兼容 mSATA 用于拓展 SATA 硬盘或者 SSD |
| SIM | 1 x SIM 卡座，用于配合 miniPCIe 扩展的 LTE 模块 |
| USB | 4 x USB2.0 HOST，1 x USB3.0 Type-C |
| 红外 | 1 x 红外接收头，支持红外遥控功能 |
| LED | 2 x 电源状态 LED（红色和绿色） 1 x mSATA 电源状态 LED（绿色） |
| 按键 | 1 x 复位键，1 x 电源键，1 x 升级键，1 x 菜单键，1 x 返回键，1 x 音量+，1 x 音量- |
| 调试 | 1 x 调试串口，用于开发调试 |
| 预留接口 | 40pin 2.54mm 排针（4 x I2C、1 x SPI、2 x UART、5 x GPIO（在全功能实现后还有 5 个 Gpio 富余）） |

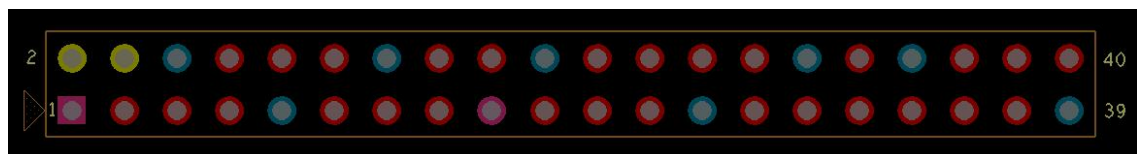
| | |
|----------------------------|---|
| 供电座 | DC12V - 2A 两 pin 插座 DC5V - 2A 两 pin 插座 |
| 电源 | DC 12V 2A 供电 TypeC 供电 Battery 供电 |
| 软件特性介绍 | |
| 系统 | Android 6.0、Debian、Ubuntu |
| 编程语言支持 | C、C++、Kotlin、Java、Shell、Pyhon 等 |
| 外观规格介绍 | |
| 产品尺寸 | 129 mm × 99 mm |
| 重量 | 99g |
| OrangePi™是深圳市迅龙软件有限公司的注册商标 | |





5、GPIO 规格

下图是香橙派 RK3399 的 GPIO Pin 脚功能图：



| | | | | | |
|-------|--------|------------|-------------|--|--------------|
| PIN1 | 3V3-1 | Power 3.3V | | | |
| PIN2 | 5V-1 | Power 5V | | | |
| PIN3 | SDA | GPI01_B3 | I2C4_SDA | | 默认用作 I2C |
| PIN4 | 5V-2 | Power 5V | | | |
| PIN5 | SCL | GPI01_B4 | I2C4_SCL | | 默认用作 I2C |
| PIN6 | GND4 | 地 | | | |
| PIN7 | GPI04 | GPI02_A0 | I2C2_SDA | | |
| PIN8 | TX | GPI04_C4 | UART2DBG_TX | | 默认用作串口 Debug |
| PIN9 | GND1 | 地 | | | |
| PIN10 | RX | GPI04_C3 | UART2DBG_RX | | 默认用作串口 Debug |
| PIN11 | GPI017 | GPI02_C0 | UART0_RXD | | |
| PIN12 | GPI018 | GPI02_A1 | I2C2_SCL | | |
| PIN13 | GPI027 | GPI02_C1 | UART0_TXD | | |
| PIN14 | GND5 | 地 | | | |
| PIN15 | GPI022 | GPI02_C2 | UART0_CTS | | |
| PIN16 | GPI023 | GPI02_A2 | | | |

| | | | | | |
|-------|--------|------------|-----------|----------|---|
| PIN17 | 3V3-2 | Power 3.3V | | | |
| PIN18 | GPI024 | GPI02_A3 | | | |
| PIN19 | MOSI | GPI01_A7 | SPI1_RXD | | |
| PIN20 | GND6 | 地 | | | |
| PIN21 | MISO | GPI01_B0 | SPI1_TXD | | |
| PIN22 | GPI025 | GPI02_C3 | UART0_RTS | | |
| PIN23 | SCLK | GPI01_B1 | SPI1_CLK | | |
| PIN24 | CS0 | GPI01_B2 | | | |
| PIN25 | GND2 | 地 | | | |
| PIN26 | CS1 | GPI02_D4 | | | 当用到 J90005 接 Camera 时，此管脚被占用 DVP_PDNO_H |
| PIN27 | DNP1 | GPI04_D2 | | | |
| PIN28 | DNP2 | GPI01_C2 | | | |
| PIN29 | GPI05 | GPI02_A4 | | | 默认用作 PCIE_PERST |
| PIN30 | GND7 | 地 | | | |
| PIN31 | GPI06 | GPI02_A5 | | | 默认用作 HDMIIN_PWREN33 |
| PIN32 | GPI012 | GPI02_B4 | SPI2_CSN | | 当用到 J4601 接 Camera 时，此管脚被占用 DVP_PDNO_H |
| PIN33 | GPI013 | GPI02_A6 | | | 默认用作 HDMIIN_PWREN |
| PIN34 | GND8 | 地 | | | |
| PIN35 | GPI019 | GPI02_A7 | I2C7_SDA | | |
| PIN36 | GPI016 | GPI02_B1 | SPI2_RXD | I2C6_SDA | 默认用作 HDMIIN_PWREN18 |
| PIN37 | GPI026 | GPI02_B0 | I2C7_SCL | | |
| PIN38 | GPI020 | GPI02_B2 | SPI2_TXD | I2C6_SCL | |
| PIN39 | GND3 | 地 | | | |
| PIN40 | GPI021 | GPI02_B3 | SPI2_CLK | | |

二、开发板使用说明

1. 准备硬件与软件工具

硬件需求：

- Orange Pi RK3399 开发板
 - 一台编译用的主机，配置最好满足以下条件：
 - 64 位 CPU
 - 16GB 内存
 - 40GB 以上的空闲磁盘空间
 - 操作系统为 Ubuntu12.04 以上，最好为 Ubuntu16.04
- 更 详 细 的 内 容 可 以 参 考 [Google 文 档](https://source.android.com/source/building)
<https://source.android.com/source/building>

软件需求：

- Orange Pi RK3399 SDK
- Orange Pi RK3399 固件
- Android 烧录工具

以上软件可以通过 Mega、百度云盘 的方式获取，详情参见中英文官网：

<http://www.orangepi.org/downloadresources/>

<http://www.orangepi.cn/downloadresourcescn/>

2. 开发板的供电方式

开发板的供电方式有三种：

- DC （12V 2A）供电：
插入 DC 适配器后即可开机。
- TypeC （5V 3A）供电：
插入 TypeC 适配器后即可开机。
- Battery （7.4V 双节）供电：

根据我们的 BAT 接口接好电池即可。现在支持 TypeC 充电、DC 充电、TypeC 与DC同时插上时采用DC充电。一般用户不推荐使用电池供电，因为不同的电池需要根据其各自的参数和 BQ25700 电池管理 IC 进行匹配。

三、Android 编译环境搭建

1、获取 SDK 源码压缩包

为方便客户快速进行开发，OrangePi 有提供不同版本的 SDK 初始压缩包。
以 OrangePi-RK3399_Android6.0_V1.0_2017_0720.tgz 为例。获取到该初始压缩包后：

```
mkdir OrangePi-rk3399
tar xvf OrangePi-RK3399_Linux4.4_V1.0_2018_1030.tar.gz -C
OrangePi-rk3399
cd OrangePi-rk3399
```

2、搭建编译环境

可以参考 Google 官方文档：<http://source.android.com/source/initializing.html>

● 安装 JDK

Android6.0 系统编译依赖于 JAVA7。编译前需要安装 OpenJDK。

安装命令如下：

```
sudo apt-get install openjdk-7-jdk
```

配置 JAVA 环境变量，比如安装路径为 /usr/lib/jvm/java-7-openjdk-amd64

可以在终端执行如下命令配置环境变量：

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

● 安装软件包

对于 Ubuntu12.04:

```
sudo apt-get update
sudo apt-get install git gnupg flex bison gperf build-essential \
zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
g++-multilib mingw32 tofrodos gcc-multilib ia32-libs \
python-markdown libxml2-utils xsltproc zlib1g-dev:i386
```

对于 Ubuntu14.04:

```
sudo apt-get update
sudo apt-get install git-core gnupg flex bison gperf libssl1.2-dev \
libbsd0-dev libxgtk2.8-dev squashfs-tools build-essential zip curl \
libncurses5-dev zlib1g-dev pngcrush schedtool libxml2 libxml2-utils \
```

```
xsltproc lzop libc6-dev schedtool g++-multilib lib32z1-dev  
lib32ncurses5-dev \  
lib32readline-gplv2-dev gcc-multilib libswitch-perl
```

安装 ARM 交叉编译工具链和内核相关的软件包：

```
sudo apt-get install gcc-arm-linux-gnueabihf \  
lzop libncurses5-dev \  
libssl1.0.0 libssl-dev
```

至此，我们可以开始进行 SDK 源码的编译。

3、编译 SDK 源码

U-boot 编译：

```
cd u-boot  
make rk3399_defconfig  
make ARCH=aarch64 -j4
```

Kernel 编译：

```
cd kernel  
make ARCH=arm64 orangepi_defconfig  
make ARCH=arm64 rk3399-orangepi.img -j4
```

Android 编译：

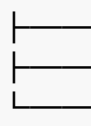
```
source build/envsetup.sh  
lunch rk3399_mid_orangepi-userdebug  
make -j4
```

4、生成固件

```
./mkimage.sh
```

执行完 ./mkimages.sh 后，在 rockdev/Image-rk3399_mid 目录下会生成完整的固件包：

```
rockdev/Image-xxx/  
├── boot.img  
├── kernel.img  
├── misc.img  
├── parameter.txt  
├── recovery.img  
├── resource.img  
└── RK3399MiniLoaderAll_V1.05.bin
```

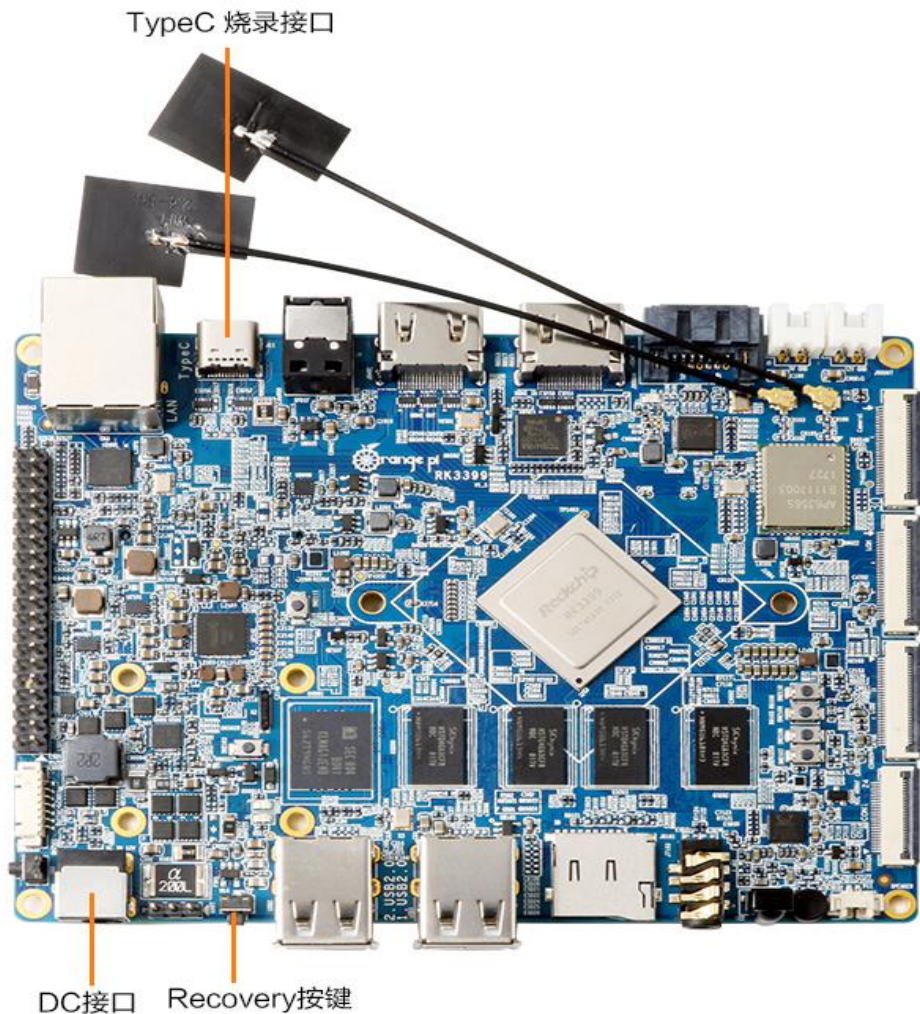


- system.img
- trust.img
- uboot.img

除此之外，开发者也可通过 `Linux_Pack_Firmware` 对以上固件进行打包，生成统一固件 `update.img`。

四、 Android 固件烧录

Orange Pi RK3399 开发板 烧录相关按键与接口如下：



固件文件有两种

- 多分区镜像文件：编译生成的 `uboot.img`、`recovery.img`、`trust.img`、`kernel.img`、`resource.img`、`system.img` 等，一般用于调试阶段。
- 单个统一镜像文件：由打包工具将多分区镜像文件打包生成的 `update.img`，一般用于固件发布。

我们提供已经制作好的 统一镜像文件，请下载后解压：

您也可以参考Orange Pi RK3399 环境搭建及固件编译这一章节 自行进行编译。

主机 操作系统支持：

- Windows XP （32/64 位）
- Windows 7 (32/64 位)
- Windows 8 (32/64 位)

- Linux (32/64 位)

在 Windows 下我们采用的工具是 AndroidTool

下载路径: RKTool_Windows.tar.gz

在 Linux 下我们采用的工具是 upgrade_tool

下载路径: RKTool_Linux.tar.gz里的Linux_Upgrade_Tool_v1.24.zip

请根据自己的主机环境选取合适的工具。

1、Windows 下进行镜像烧录

Windows 所采用的工具是 AndroidTool, 他可以用来烧写多分区镜像文件, 也可以用来烧录统一镜像文件 update.img。

在烧录前, 我们需要安装 RK 设备在 Windows 下的 USB 驱动。

- 安装 RK USB 驱动

下载路径: DriverAssitant

解压后运行 DriverInstall.exe

为了使所有设备都使用最新版本的驱动, 请先点击“驱动卸载”, 再点击“驱动安装”。如下图



在安装完 USB 驱动后将 TypeC 数据线连接 主机 和 Orange Pi 开发板, 右下角会显示“Rockusb Device 成功安装”。如下图



进入烧录模式

1. Type-A 连接电脑。
2. 按住开发板的 Recovery 键。
3. Type-C 连接开发板，此时工具下方会显示，识别到 Loader 设备。如下图：

发现一个LOADER设备

如果有连接串口，可以观察到此时串口信息如下：

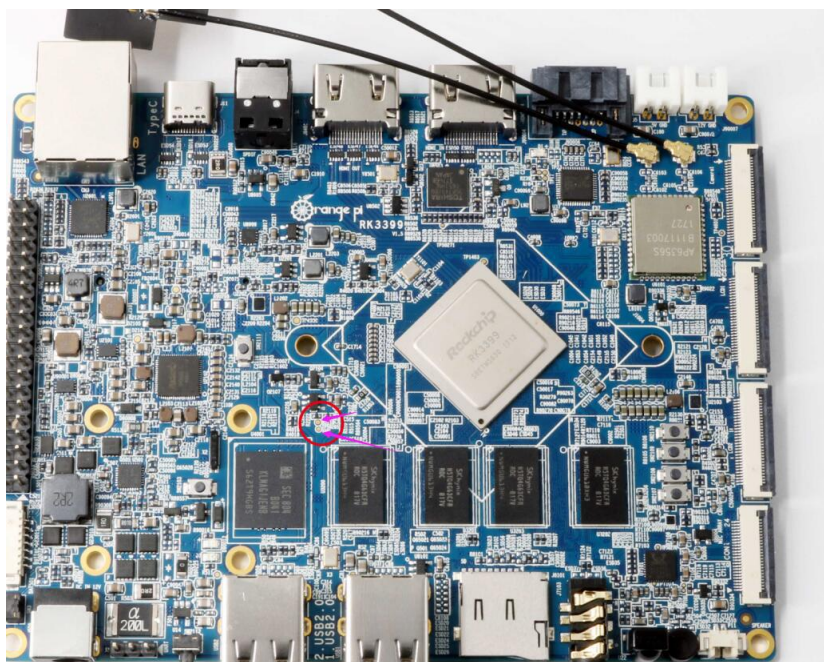
```
#Boot ver: 2017-07-12#1.05
empty serial no.
normal boot.
checkKey
vbus = 1
rockusb key pressed.
```

如果识别不到 Loader 设备，则需要短接烧录，方法如下：

- 1) 在短接 EMMC 的 Clk 和 GND 的同时，连接 DC 电源上电。进入 Maskrom 模式。

EMMC Clk 在板子正面 EMMC 的右上角 如下图:



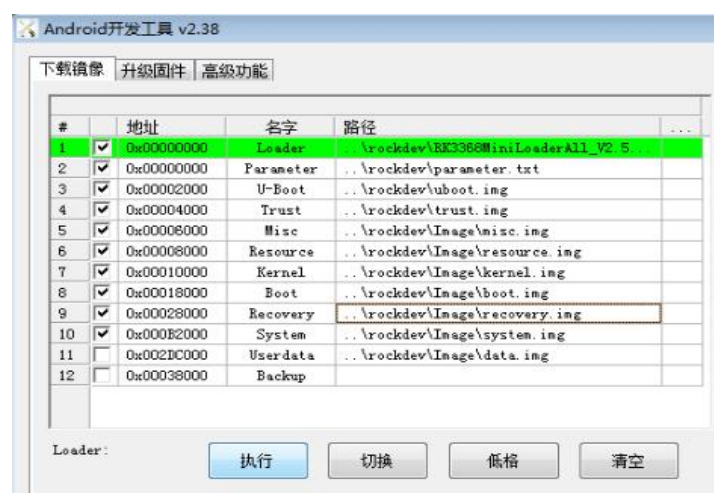


2) 连接 TypeC 线。

4. 插入 DC 电源（适配器）。

因为 OrangePi RK3399 支持 TypeC 供电，一般来说，主机的 USB 端口电压足以支撑烧录，但是不足以支持板子开机启动。所以我们仍然需要插入 DC 电源，以保证烧录完成后板子能够正常启动。

5. 如果需要单独烧录各个分区 *.img，在“下载镜像”这个子页中，点击右侧“...”选择好对应要烧录的分区的固件路径。如下：



点击“执行”，即开始进行烧录。右边会出现烧录进度条。

6. 如果需要整体烧录统一固件 update.img，在“升级固件”这个子页中，点击“固件”选择 update.img 的固件路径。同样，在识别到 loader 设备后，点击“升级”，即可开始升级固件。右侧将会显示烧录进度条。

2、Linux 下镜像烧录

● 烧录工具 upgrade_tool

Linux 系统下，我们所采用的工具叫做 upgrade_tool。工具下载链接：

和 Windows 相同，请按照如下方式进入 Loader 烧录模式：

1. Type-A 连接电脑。
2. 按住开发板的 Recovery 键。
3. Type-C 连接开发板，此时可以观察到电源灯亮。
4. 插入 DC 电源（适配器）。

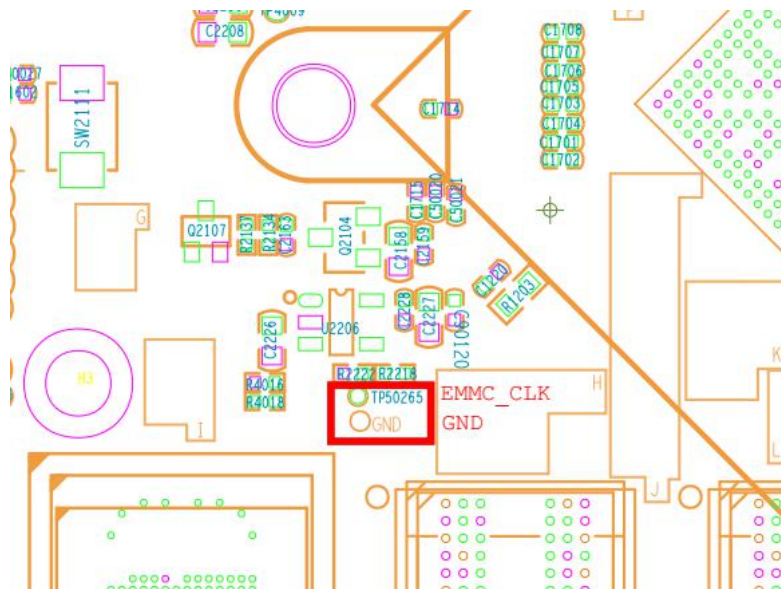
虽然 OrangePi RK3399 支持 TypeC 供电，一般来说，主机的 USB 端口电压足以支撑烧录，但是不足以支持板子开机启动。所以我们仍然需要插入 DC 电源，以保证烧录完成后板子能够正常启动。

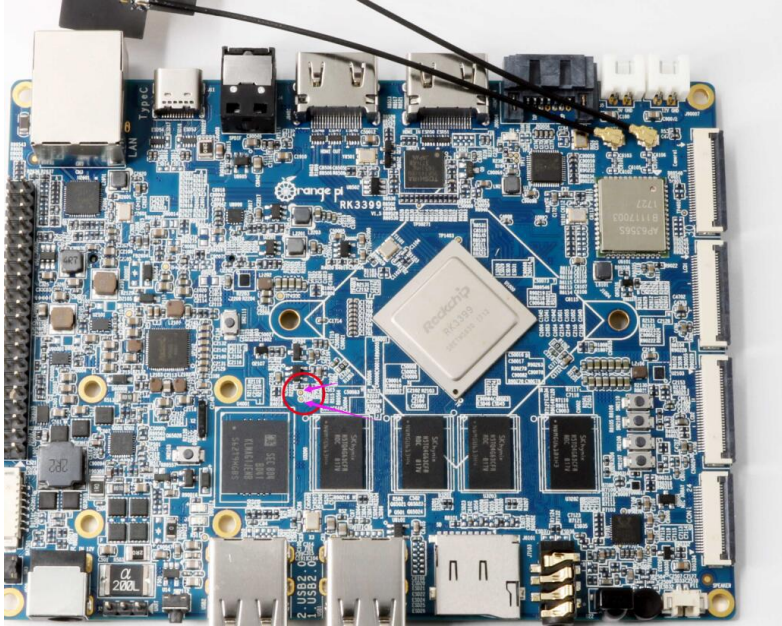
如果有接串口，可以观察到如下信息：

```
#Boot ver: 2017-07-12#1.05
empty serial no.
normal boot.
checkKey
vbus = 1
rockusb key pressed.
```

如果识别到不设备，则需要短接烧录，方法如下：

- 1) 在短接 EMMC 的 Clk 和 GND 的同时，连接 DC 电源上电。进入 Maskrom 模式。
EMMC Clk 在板子正面 EMMC 的右上角 如下图：





2) 连接 TypeC 线。

5. 在 Linux 终端中执行 upgrade_tool 工具:

```
$ sudo ./upgrade_tool
List of rockusb connected
DevNo=1 Vid=0x2207,Pid=0x330c,LocationID=201 Loader
Found 1 rockusb,Select input DevNo,Rescan press <R>,Quit press <Q>:
```

输入 R 是重新扫描 USB 口以发现设备;
输入 Q 是退出烧录工具;
输入 DevNo 号 是选取相应要操作的设备。
我们这里输入 1:

```
Found 1 rockusb,Select input DevNo,Rescan press <R>,Quit press <Q>:1
```

进入的时候会显示帮助菜单，并且出现 Rockusb> 提示符


```

-----Tool Usage -----
Help:          H
Quit:          Q
Version:       V
Clear Screen:  CS
-----Upgrade Command -----
ChooseDevice:  CD
SwitchDevice:  SD
UpgradeFirmware:  UF <Firmware>
UpgradeLoader:  UL <Loader>
DownloadImage:  DI <-p|-b|-k|-s|-r|-m image> [parameter file]
DownloadBoot:   DB <Loader>
EraseFlash:     EF <Loader|firmware>
LowerFormat:    LF
-----Professional Command -----
TestDevice:     TD
ResetDevice:    RD [subcode]
ResetPipe:      RP [pipe]
ReadFlashID:    RID
ReadFlashInfo:  RFI
ReadChipInfo:   RCI
ReadSector:     RS <BeginSec> <SectorLen> [-decode] [File]
WriteSector:    WS <BeginSec> <File>
ReadLBA:        RL <BeginSec> <SectorLen> [File]
WriteLBA:       WL <BeginSec> <File>
EraseBlock:     EB <CS> <BeginBlock> <BlockLen> [--Force]
-----
Rockusb>

```

6. 我们通过在 Rockusb> 提示符后输入相应指令进行操作。不区分大小写。
首先介绍 **TD 命令**，TD 命令是用来测试设备状态是否正常。

```

Rockusb>td
Test Device OK.

```

如果需要单独烧录各个分区 *.img 可以使用 **DI 命令**：

DI <-p|-b|-k|-s|-r|-m image> [parameter file]

DI 的第一个参数用来指定需要烧录的分区名

DI 的第二个参数用来指定所烧镜像的路径

比如我现在需要烧录 kernel.img，有两种做法：

```
Rockusb>di -k ./kernel.img
```

```
Rockusb>di kernel ./kernel.img
```

```

Rockusb>di kernel Image-rk3399_mid/kernel.img
Download kernel start...
Download_image ok.

```

如果需要整体烧录统一固件 update.img 可以使用 **UF 命令**：

UF <Firmware>

UF 的唯一一个参数用来指定需要烧录固件路径

比如我的固件路径为 RK3399_IMAGE/Image_Android6.0_20171228.img，则执行命令如下：

```
Rockusb>uf RK3399_IMAGE/Image_Android6.0_20171228.img
```

```

Rockusb>uf ALL_IMAGE/Image_20171228_释放给客户的固件_DDR800MHZ/update.img
Loading firmware...
Support Type:RK330C      FW Ver:6.0.01   FW Time:2017-12-28 16:56:01
Loader ver:1.05 Loader Time:2017-07-12 16:56:34
Download Image Total(937159K),Current(446067K)

```

烧录完成会显示如下内容并自动重启系统。（UF 命令会自动重启，DI 命令不会）

```

Loading firmware...
Support Type:RK330C      FW Ver:6.0.01   FW Time:2017-12-28 16:56:01
Loader ver:1.05 Loader Time:2017-07-12 16:56:34
Upgrade firmware ok.

```

● 编写脚本实现自定义烧录

除了上述进入 `upgrade_tool` 工具，并完成烧录。

`upgrade_tool` 也支持作为 Linux 命令实行烧录，只需要将工具所在路径添加到环境变量中即可。

这也意味着我们可以通过在脚本中添加烧录命令完成一些便捷的调试，实例如下。

在调试 Kernel 的时候，我希望快速实现 修改—编译—烧录。那么可以这样做：

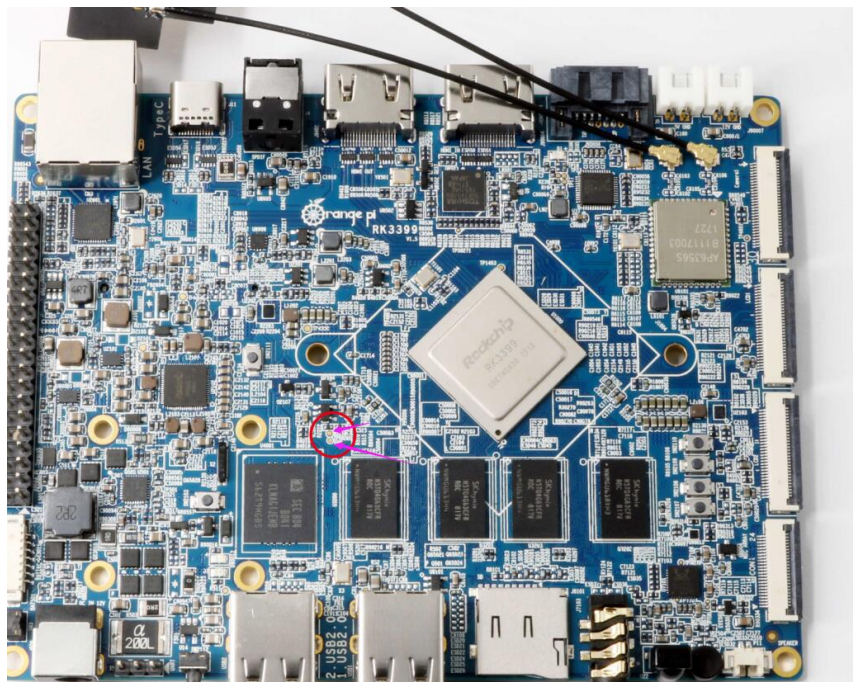
```
# 编译 kernel 部分的固件，产生有 kernel.img 和 resource.img
make -j2 rk3399-orangepi.img
# 通过 adb 命令进入 loader 烧录模式
adb shell rebot bootloader
# 利用 di 命令完成烧录
sudo upgrade_tool di resource resource.img
sudo upgrade_tool di kernel kernel.img
# 利用 rd 命令完成重启
sudo upgrade_tool rd
```

3、固件烧录问题汇总

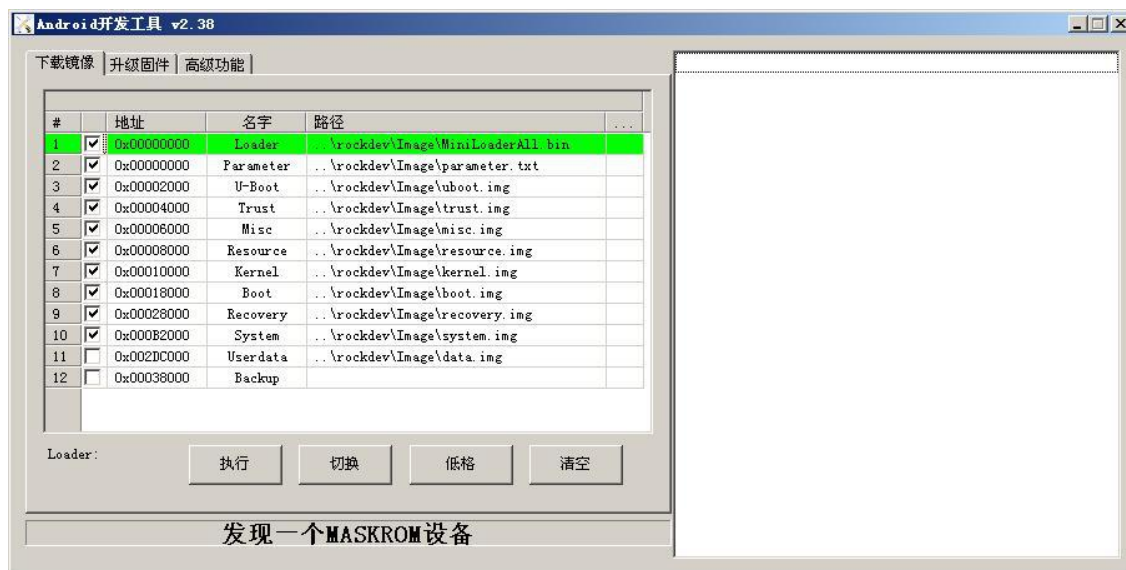
● 因固件有误而导致板子无法开机

如 3.2 中所述，我们一般可以通过进入 LOADER 模式来进行固件烧录。但是如果因为制作的固件有误而不慎无法开机，我们可以进入 MASKROM 模式来烧录开机引导程序。方法如下：

1. 设备断开所有的电源
2. 使用金属镊子短接 EMMC CLK （TP50265 焊点）和 GND
3. 插上 DC 电源
4. 稍等片刻（3 秒），松开镊子
5. 插上 USB TypeC 数据线连接好设备和主机
6. 设备已经进入 MASKROM 模式



使用 Windows 下的 AndroidTools 工具，现象如下：



此后即可按照正常的方法进行烧录。（注：解决变砖问题请重新烧录官方固件）

具体操作步骤参看第三章《Windows 下固件烧录》。

使用 Linux 下的 upgrade_tool 工具，现象如下：

```
→ rockdev sudo ./upgrade_tool
List of rockusb connected
DevNo=1 Vid=0x2207,Pid=0x330c,LocationID=203 Maskrom
Found 1 rockusb,Select input DevNo,Rescan press <R>,Quit press <Q>:1
```

利用 uf 命令即可重新烧录完整固件，具体操作步骤参看 第四章《Linux 下镜像烧录》。

原理：将 Flash 的 pin 脚和地线进行短接，系统会认为 Flash 数据出错，从而清除 Flash 数据。

五、Linux 编译环境搭建及 Linux 固件烧录

1、获取 linux 源码

● OrangePi 源码下载器

Orange Pi RK3399 的 Linux 源码已经上传到 GitHub, 内核版本为 Linux 4.4, 我们可以使用

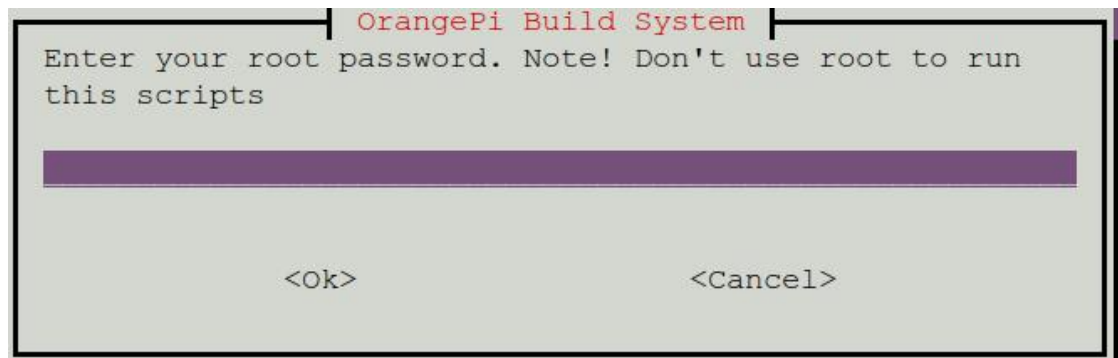
OrangePi Linux 源码专用的下载器进行下载, 获取下载器源码的方式如下所示:

```
$ sudo apt-get install git
$ git clone https://github.com/orangepi-xunlong/OrangePi_Build.git
$ cd OrangePi_Build
$ ls
Build_OrangePi.sh lib README.md
```

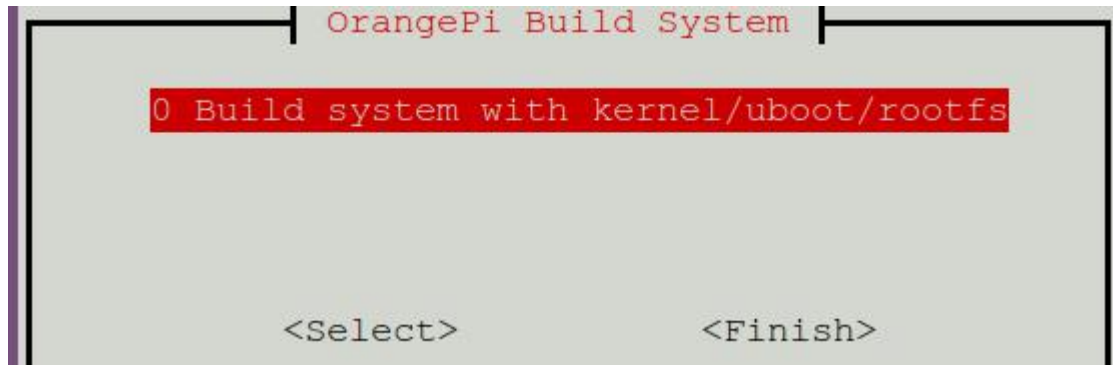
● 运行下载器

```
$ ./Build_OrangePi.sh
```

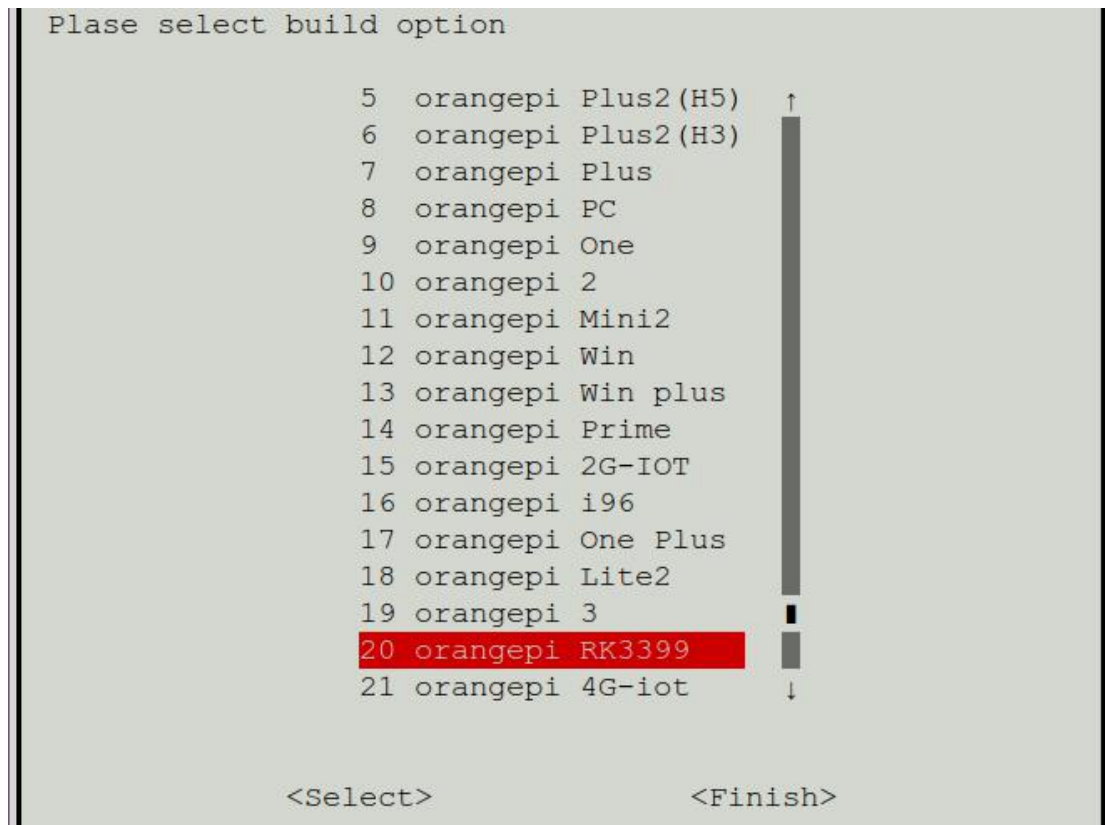
输入 root 密码, 然后回车



选择 0 Build system with kernel/uboot/rootfs, 进入开发板型号选择界面。



选择 20 orangepi RK3399, 回车后将会开始下载 Orange Pi RK3399 的 Linux 源码



下载的源码会存放在 OrangePi_Build 的同级目录下。

```
$ ls ../
OrangePi_Build OrangePiRK3399
```

OrangePi RK3399 的 Linux 源码目录结构如下所示

```
$ cd OrangePiRK3399
$ tree -L 1
.
├── build.sh -> scripts/build.sh
├── external
├── kernel
├── output
├── scripts
├── toolchain
└── uboot
```

编译启动脚本
存放额外的配置文件
Linux 内核源码
存放输出文件
编译脚本
交叉编译工具
uboot 的源码

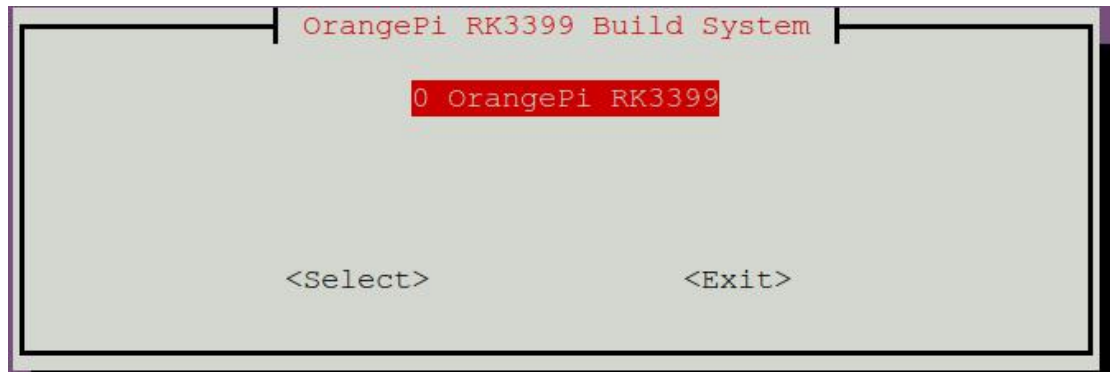
6 directories, 1 file

2、编译 Linux 源码

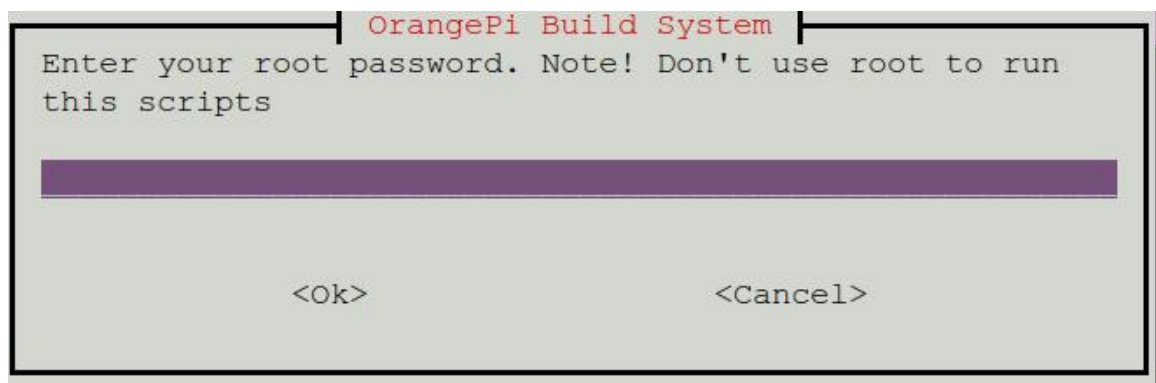
● 执行编译启动脚本

```
$ cd OrangePiRK3399
$ ./build.sh
```

选择回车

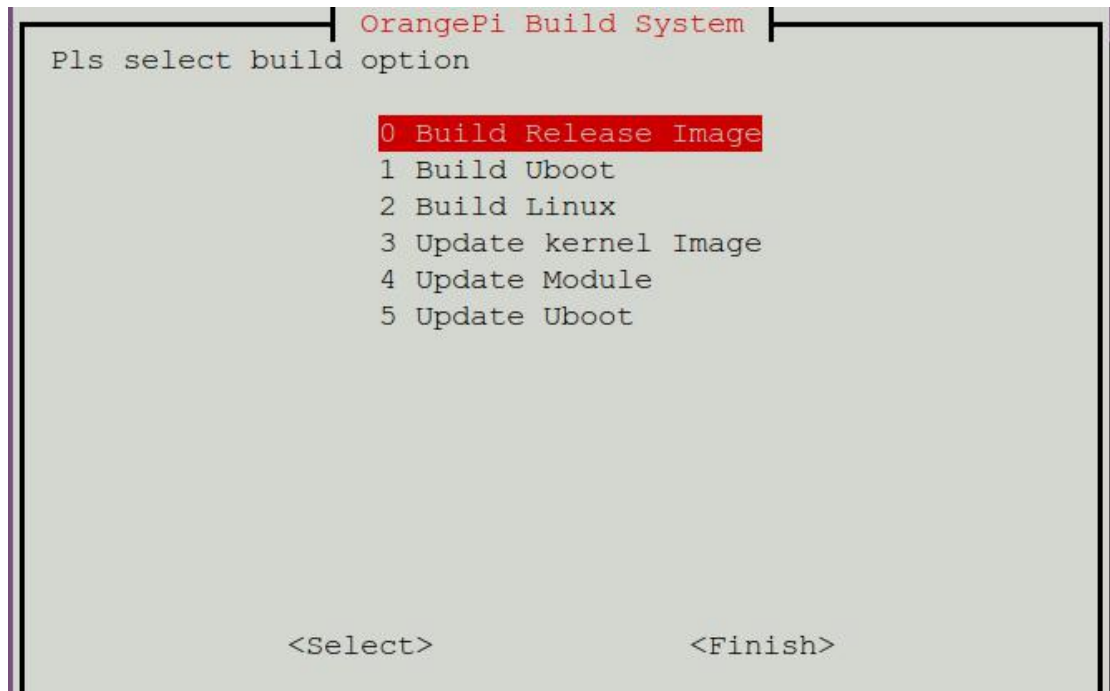


输入 root 密码并回车,然后选择需要执行的功能



其中各选项的功能如下:

- | | |
|-------------------------|---------------------------|
| ● 0 Build Release Image | 编译 Ubuntu 或 Debian 的发行版镜像 |
| ● 1 Build Uboot | 编译 uboot 源码 |
| ● 2 Build Linux | 编译 Linux 内核源码 |
| ● 3 Update kernel Image | 更新 内核 |
| 0 SD Card | 更新 SD 卡中 linux 系统的内核 |
| 1 EMMC | 更新 EMMC 中 Linux 系统的内核 |
| ● 4 Update Module | 更新 SD 卡 Linux 系统中的内核模块 |
| ● 5 Update Uboot | |
| 0 SD Card | 更新 SD 卡中 linux 系统的 uboot |
| 1 EMMC | 更新 EMMC 中 Linux 系统的 uboot |



编译完 u-boot 和内核源码后生成的最终文件会保存在 output 目录下

Uboot 镜像

```
$ tree -L 1 output/u-boot/  
├── idbloader.img  
├── rk3399_loader_v1.08.106.bin  
├── trust.img  
└── uboot.img
```

0 directories, 4 files

Kernel 镜像

```
$ tree -L 1 output/kernel/  
output/kernel/  
├── Image  
└── rk3399-orangepi.dtb
```

0 directories, 2 files

Rootfs 编译

参考《搭建编译环境并制作 Rootfs 镜像》一章。

执行以下步骤将所有分区的镜像打包为一个完整的固件：

```
$ cd scripts
```

打包 Image 和 extlinux 配置文件生成 boot.img ，路径在 output 文件夹下

```
$ ./mk-image.sh -c rk3399 -t boot -f emmc
```

打包 uboot、boot、rootfs 生成完整镜像 system.img，路径在 output 文件夹下

```
$ ./mk-image.sh -c rk3399 -t system -s 4000 -r ../output/rootfs.img
```

c for chip，代表芯片

t for target，代表生成的 image 的名字

s for size，代表制作固件时预分配的大小（并不代表实际最后固件的大小，最后的固件会重新动态调整），单位 Mbyte。

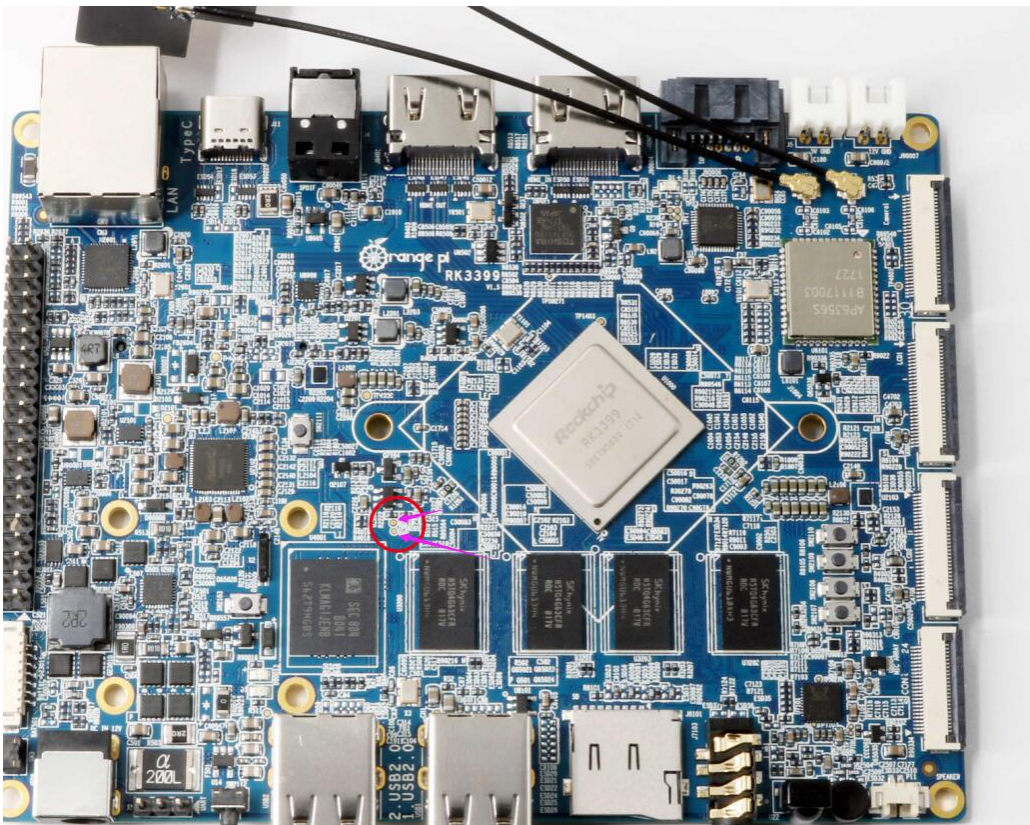
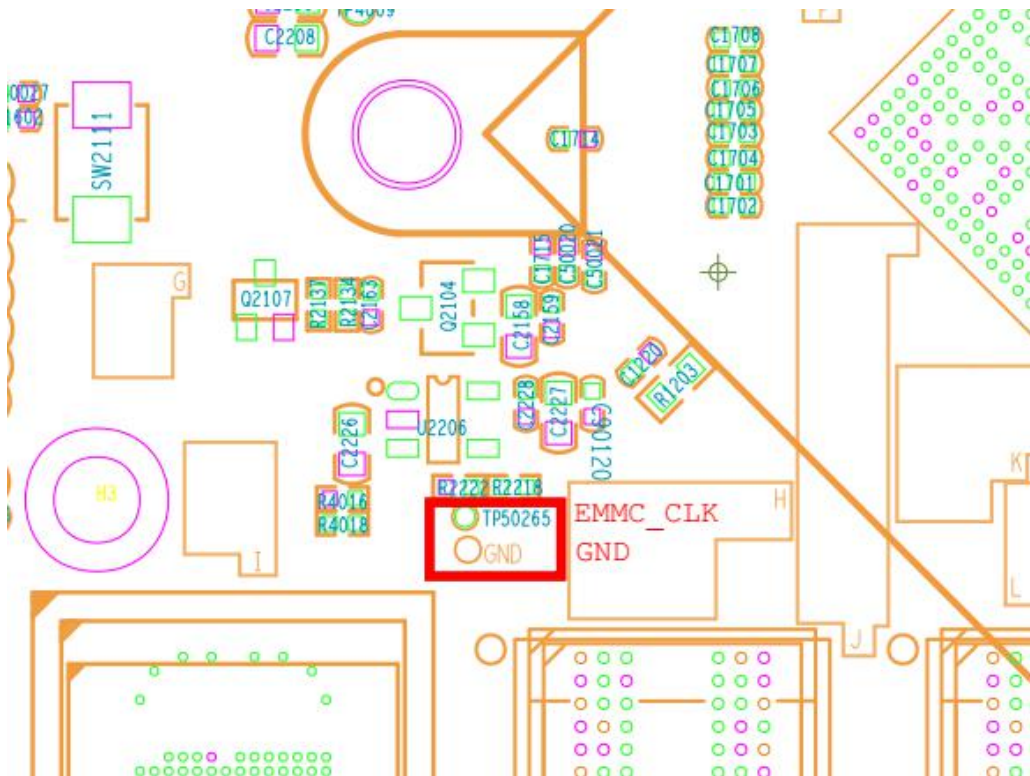
r for rootfs，代表所使用的 rootfs 的路径。

执行完成后，在 out 目录下生成了 system.img 即为最终的完整固件。

3、linux 固件烧录

烧录完整的 Linux 固件到 eMMC：

1. 在短接 EMMC 的 Clk 和 GND 的同时，连接 DC 电源上电。进入 Maskrom 模式。EMMC Clk 在板子背面 miniPCIe 的左下角 如下图：



2. 连接 TypeC 线。
3. 执行如下命令进行烧录

```
./flash_tool.sh -c rk3399 -p system -i ../output/system.img
```

c for chip, 代表芯片型号

p for partation, 代表分区, 比如 boot loader1 system

i for image, 代表镜像的路径

您也可以单独烧录不同的分区, 比如

单独烧录 boot:

```
./flash_tool.sh -c rk3399 -p boot -i ../output//boot.img
```

单独烧录 uboot:

```
./flash_tool.sh -p loader1 -i ../output/u-boot/idbloader.img -c rk3399  
./flash_tool.sh -p loader2 -i ../output/u-boot/uboot.img -c rk3399
```

烧录官网提供的 **linux** 镜像

```
./flash_tool.sh -c rk3399 -p system -i \  
OrangePi_rk3399_xenial_desktop_v1.0.img
```


六、 搭建编译环境并制作 Rootfs 镜像

1、 搭建编译环境

```
sudo apt-get update
sudo apt-get install qemu-user-static
```

2、 获取 Linux Rootfs 源码包

客户可以前往不同发行版各自的官网获取其源码包。这里以 Ubuntu16.04 为例。我们在 Ubuntu cdimage 上获取本指南中所用的源码：

<http://cdimage.ubuntu.com/ubuntu-base/releases/16.04/release/>

下载 ubuntu-base-16.04.1-base-arm64.tar.gz 并解压

```
mkdir rootfs
sudo tar -xpf ubuntu-base-16.04.1-base-arm64.tar.gz -C rootfs
```

3、 修改 Rootfs 并添加定制软件

```
sudo cp -b /etc/resolv.conf rootfs/etc/resolv.conf
sudo cp /usr/bin/qemu-aarch64-static rootfs/usr/bin/
# 进入根文件系统
sudo chroot rootfs /bin/bash

# 更新软件仓库并安装软件
apt update
apt upgrade
# 根据自己需求安装功能
apt install build-essential vim ping ssh 等
# 安装桌面版，请保持网络顺畅，耗时比较长
# 如果不需要也可以不执行，那制作出来的就是 Server 版
apt install ubuntu-desktop

# 添加用户及设置密码
useradd -s '/bin/bash' -m -G adm,sudo orangepi
# 给用户 orangepi 设置密码
passwd orangepi
# 给用户 root 设置密码
passwd root

# 退出 Rootfs
exit
```

4、制作 Rootfs 镜像

```
# 生成空白 image 文件
dd if=/dev/zero of=ubuntu-desktop.img bs=1M count=2048

# 格式化 image 文件为 ext4 格式
sudo mkfs.ext4 ubuntu-desktop.img

# 挂载 image 文件到 ubuntu-desktop 文件夹
mkdir ubuntu-desktop
sudo mount ubuntu-desktop.img ubuntu-desktop/

# 将刚才制作的 rootfs 的内容拷贝到 image 挂载的文件夹下
sudo cp -rfp rootfs/* ubuntu-desktop/

# 取消挂载
sudo umount ubuntu-desktop/

# 检查文件系统的正确性
e2fsck -p -f ubuntu-desktop.img

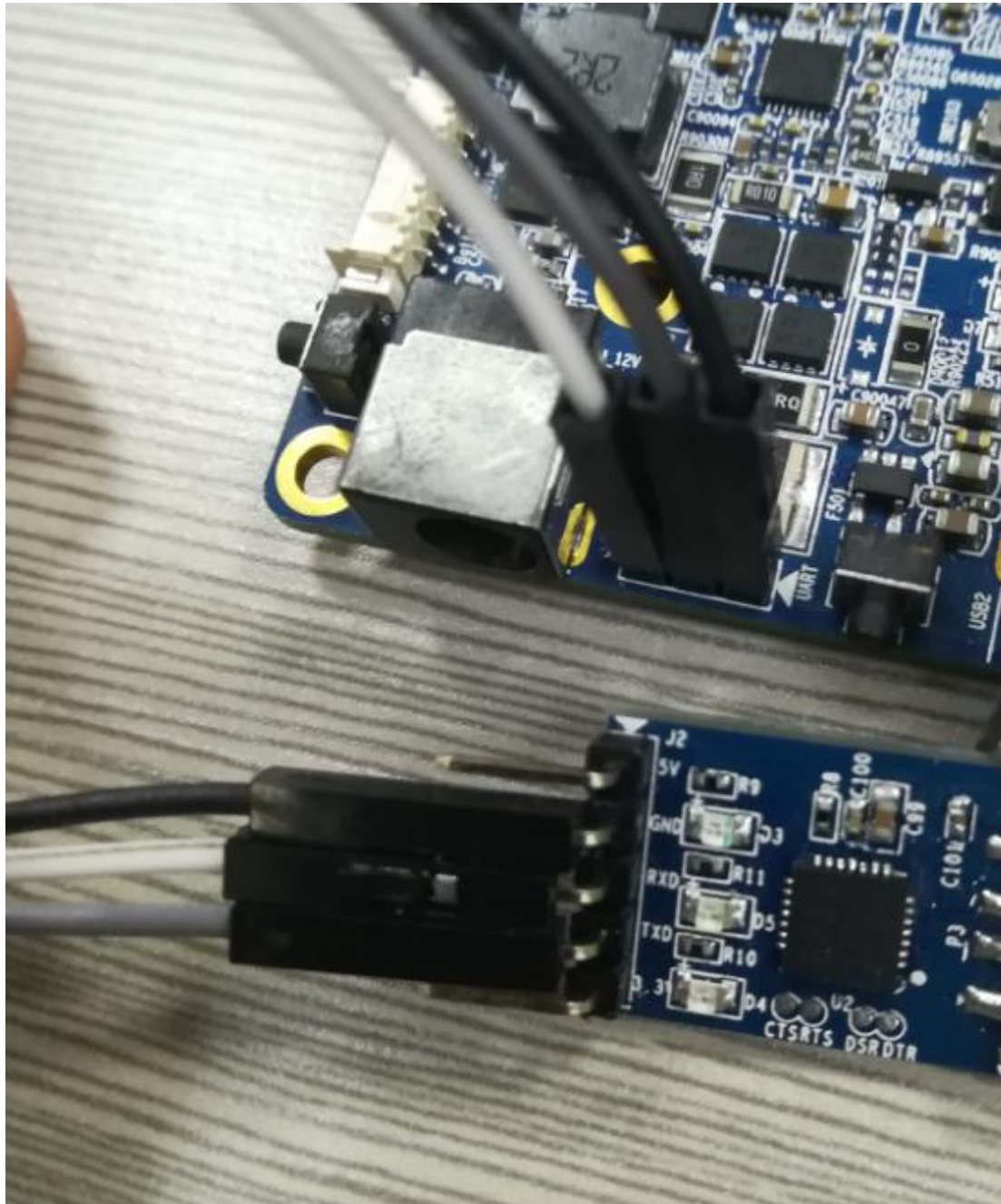
# 自动调整分区的大小
resize2fs -M ubuntu-desktop.img
```

七、串口调试工具介绍

首先需要准备一根 USB 转 TTL 串口线，需要支持 1500000 波特率

按下图接好串口线，不同颜色的线对应的功能如下：

- 黑色——GND
- 绿色——RX
- 白色——TX



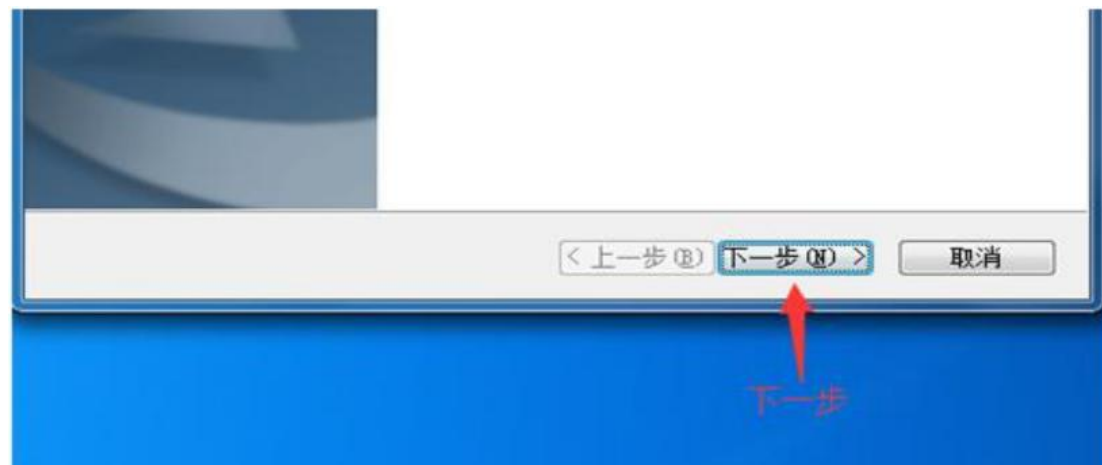
1、基于 Windows 平台的使用

● 安装 USB 驱动

下载最新版的驱动 PL2303_Prolific_DriverInstaller_v130.zip, 下载解压

| | | | | |
|--------------------------------------|-----------------|------------------|----------|-------------|
| PL2303_Prolific_DriverInstaller_v130 | 2010/7/15 10:41 | 应用程序 | 3,099 KB | ← 解压之后的应用程序 |
| PL2303_Prolific_DriverInstaller_v130 | 2016/8/3 9:20 | WinRAR ZIP 压缩... | 2,316 KB | ← 下载的压缩包 |
| releasenote | 2010/7/22 10:14 | 文本文档 | 2 KB | |

以管理员身份选择应用程序安装



等待安装完成

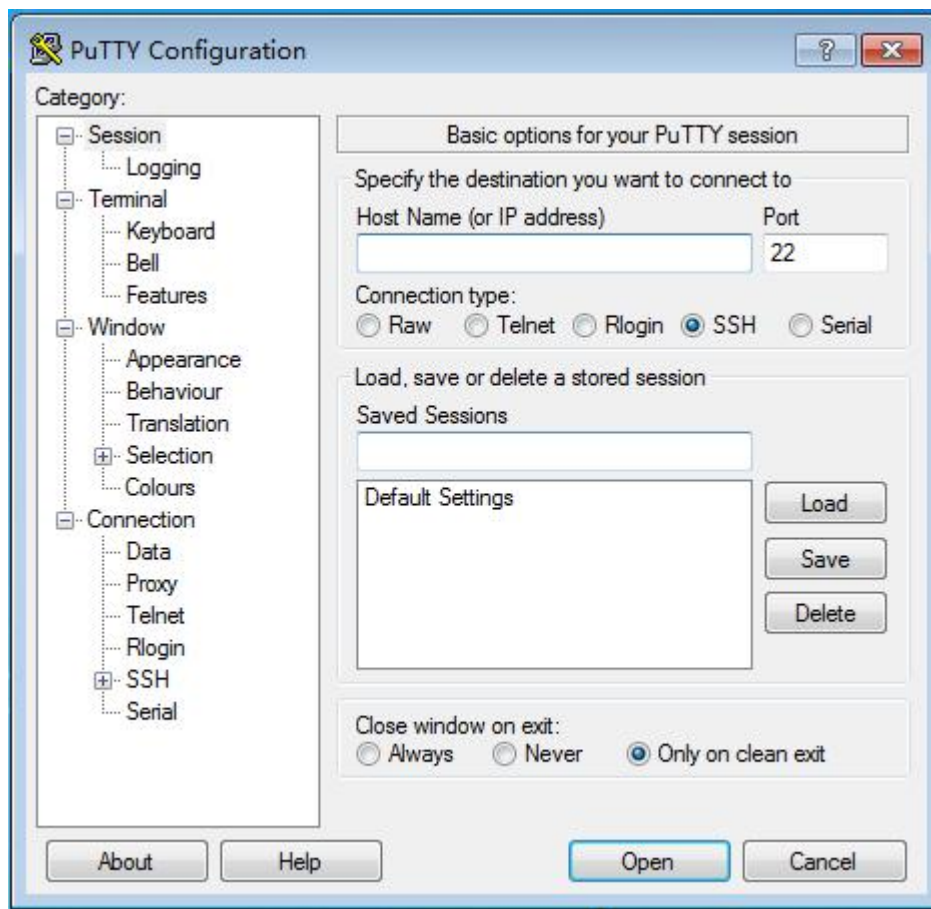


● 下载安装 Putty

Putty 可从下面的地址下载, 请选择适合自己开发环境的版本。

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

直接双击下载的 putty.exe 即可打开 putty, 软件界面如下图所示。



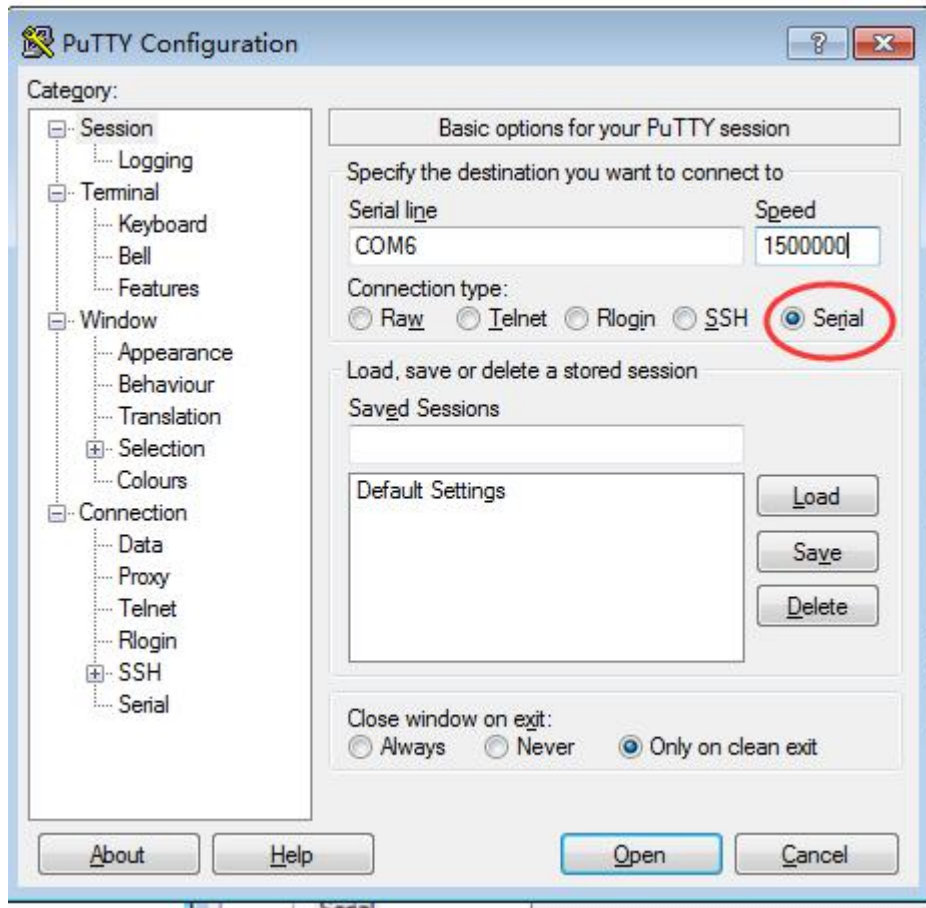
● 设备信息的获取

在 Windows7 中, 我们可以通过设备管理器查看串口连接是否正常以及串口的设备号。如果设备没有正常识别, 请检查驱动是否安装成功。如果驱动安装有问题, 可以尝试使用 360 驱动大师扫描安装驱动。



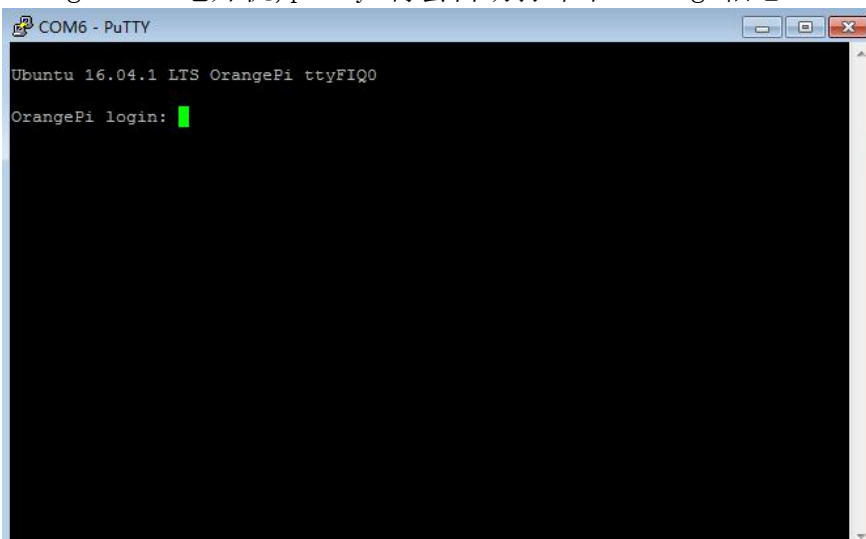
● Putty 配置

串行口设置成相应的端口号 (COM6), , 速度设置成 1500000



● 启动调试串口输出

OrangePi 上电开机, putty 将会自动打印串口 log 信息



2、基于 Linux 平台的使用

在 Linux 平台使用 putty 和 Windows 平台区别不太, 下面主要说明有差异地方的

操作步骤。所有操作都是基于 Ubuntu 14.04 系统。

● 安装并启动 Putty

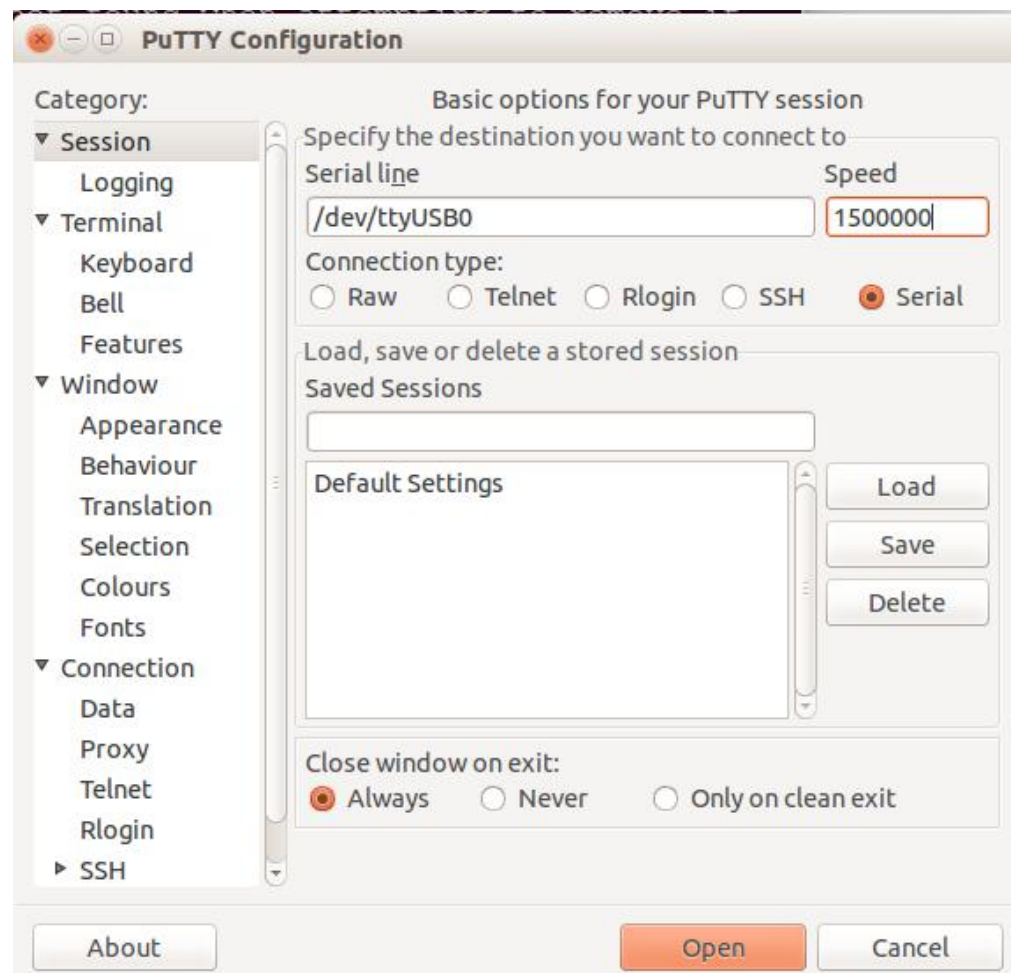
```
$ sudo apt-get install putty  
$ sudo putty
```

● 配置 Putty

串口号可以通过 `ls /dev/ttyUSB*` 查看

波特率需要设置为 1500000

并且关闭流控



八、Linux 系统使用说明

1、登录账号和密码

用户名: root 密码: orangepi

用户名: orangepi, 密码: orangepi

2、扩展 rootfs 分区

```
root@OrangePi:~# parted /dev/mmcblk1
GNU Parted 3.2
Using /dev/mmcblk1
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) resizepart
Error: The backup GPT table is corrupt, but the primary appears OK, so
that will
be used.
OK/Cancel? ok
Warning: Not all of the space available to /dev/mmcblk1 appears to be used,
you
can fix the GPT to use all of the space (an extra 22343680 blocks) or
continue
with the current setting?
Fix/Ignore? fix
Partition number? 7
End? [4193MB]? 15.6G
(parted) q
Information: You may need to update /etc/fstab.

root@OrangePi:~# resize2fs /dev/mmcblk1p7
```

3、GPU 性能测试

启动系统，在系统菜单 System Tools 中点击 LXTerminal 打开命令行终端，并运行命令：

```
$ glmark2-es2
```

可以看到 Mali-T860 的测试结果和跑分情况

4、编解码测试

● 播放 4k 视频

在系统菜单 System Tools 中点击 LXTerminal 打开命令行终端，并运行命令

```
$ test_dec-gst.sh
```

可用 `which test_dec_gst.sh` 找到这个脚本的位置。
默认音频输出到 3.5mm 耳机孔，可修改 `card` 变量为 2 输出至 `hdmi`。

● 测试编码

执行

```
$ test_enc.sh
```

5、使用 HDMI IN 功能

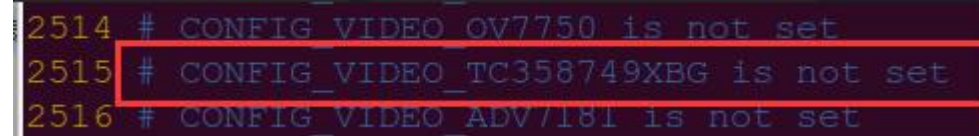
`hdmi in` 只支持接收 1080p60 的输入信号，将其他设备的 `hdmi` 输出口接到 RK3399 的 `hdmi in` 接口上。

启动系统，打开命令行终端，执行

```
$ test_hdmiin.sh
```

6、ov13850 摄像头的使用

`camera1` 接口可使用 `ov13850` 摄像头，需要在 `.config` 文件中关闭 `hdmi in` 的配置。



```
2514 # CONFIG_VIDEO_OV7750 is not set
2515 # CONFIG_VIDEO_TC358749XBG is not set
2516 # CONFIG_VIDEO_ADV7181 is not set
```

重新编译并替换内核

启动系统后，打开命令行终端，执行

```
$ test_camera.sh
```

7、GPIO 的使用

在应用层使用 `wiringOP` 操作 `gpio`，安装步骤如下。

```
$ git clone https://github.com/orangepi-xunlong/wiringOP.git
$ cd wiringOP
$ export PLATFORM=OrangePi_RK3399
$ ./build
```

命令行输入

```
gpio readall
```

| OrangePi RK3399 | | | | | | | | | | | | |
|-----------------|-----|--------|------|---|----------|----|------|--------|--------|------|-----|--|
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO | | |
| | | 3.3v | | | 1 | 2 | | 5v | | | | |
| 43 | 0 | SDA.0 | ALT2 | 0 | 3 | 4 | | 5V | | | | |
| 44 | 1 | SCL.0 | ALT2 | 0 | 5 | 6 | | 0v | | | | |
| 64 | 2 | GPIO4 | OUT | 1 | 7 | 8 | 0 | ALT2 | Tx | 3 | 148 | |
| | | 0v | | | 9 | 10 | 0 | ALT2 | Rx | 4 | 147 | |
| 80 | 5 | GPIO17 | ALT2 | 0 | 11 | 12 | 0 | IN | GPIO18 | 6 | 65 | |
| 81 | 7 | GPIO27 | ALT2 | 0 | 13 | 14 | | 0v | | | | |
| 82 | 8 | GPIO22 | ALT2 | 0 | 15 | 16 | 0 | IN | GPIO23 | 9 | 66 | |
| | | 3.3v | | | 17 | 18 | 0 | IN | GPIO24 | 10 | 67 | |
| 39 | 11 | MOSI | ALT3 | 0 | 19 | 20 | | 0v | | | | |
| 40 | 12 | MISO | ALT3 | 0 | 21 | 22 | 1 | ALT2 | GPIO25 | 13 | 83 | |
| 41 | 14 | SCLK | ALT3 | 0 | 23 | 24 | 0 | ALT3 | CS0 | 15 | 42 | |
| | | 0v | | | 25 | 26 | 0 | IN | CS1 | 16 | 133 | |
| 154 | 17 | DNP1 | IN | 0 | 27 | 28 | 0 | IN | DNP2 | 18 | 50 | |
| 68 | 19 | GPIO5 | OUT | 1 | 29 | 30 | | 0v | | | | |
| 69 | 20 | GPIO6 | OUT | 1 | 31 | 32 | 1 | OUT | GPIO12 | 21 | 76 | |
| 70 | 22 | GPIO13 | OUT | 1 | 33 | 34 | | 0v | | | | |
| 71 | 23 | GPIO19 | OUT | 1 | 35 | 36 | 1 | OUT | GPIO16 | 24 | 73 | |
| 72 | 25 | GPIO26 | OUT | 1 | 37 | 38 | 1 | OUT | GPIO20 | 26 | 74 | |
| | | 0v | | | 39 | 40 | 0 | ALT4 | GPIO21 | 27 | 75 | |
| | | (null) | | | 41 | 42 | | (null) | | | | |
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO | | |