ROC-RK3328-CC

发布 *1.0.0*

Firefly Team

2018 年 11 月 22 日

上手指南

CHAPTER 1

简介

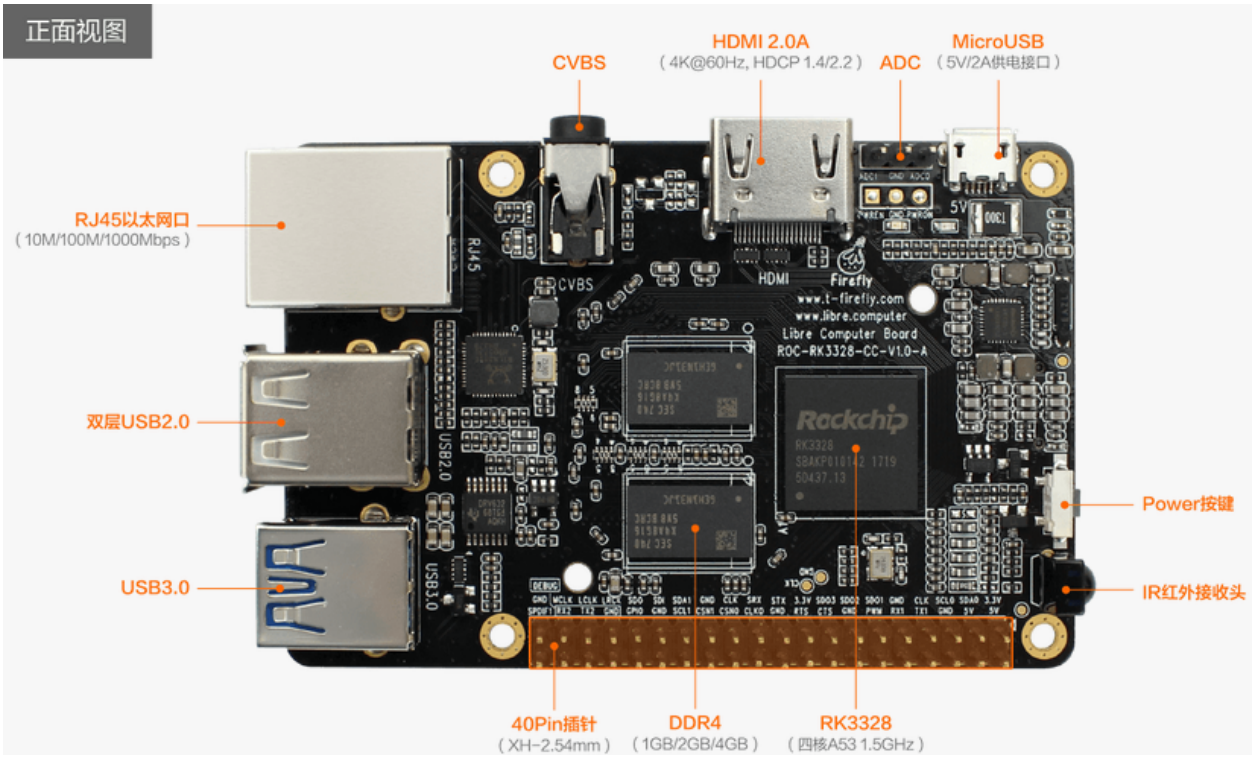
恭喜你入手 ROC-RK3328-CC 开发板！

对刚接触的人来说，《上手指南》 提供了烧写固件并让开发板跑起来的一切指引。

如果仔细阅读 《上手指南》后仍有疑问，请先查看《常见问题解答》 。如果问题仍未解决，请参照《串口调试》一章，获取系统日志信息，并联系我们（联系方式），以便共同完善这份文档。

1.1 产品规格

ROC-RK3328-CC，是 Firefly 荣誉出品的第一块信用卡大小、性能强、功耗小的开源主板。



主要功能:

- 核心
  - Quad-Core ARM® Cortex-A53 64-bit 处理器，高达 1.5GHz 的主频
  - ARM Mali-450 MP2 Quad-Core GPU，支持 OpenGL ES1.1/2.0，OpenVG1.1
  - DDR4 RAM (1GB/2GB/4GB)
- 连接
  - 2 x USB 2.0 Host 口，1 x USB 3.0 Host 口
  - 10/100/1000Mb 网口
  - 1 x IR 红外模组，支持自定义 IR 远程控制
  - 40 引脚拓展接口，包含 GPIO，I2S，SPI，I2C，UART，PWM SPDIF，DEBUG
- 显示
  - HDMI 2.0 (Type-A)，支持最大 4K@60Hz 显示
  - TV out，CVBS 显示，参照 480i，576i 标准
- 音频
  - I2S，支持 8 通道
  - SPDIF，音频输出
- 视频
  - 4K VP9 和 4K 10位 H265 / H264 视频解码，可达 60fps
  - 1080P 多格式视频解码(WMV，MPEG-1/2/4，VP9，H.264，H.265)
  - 1080P 视频解码，支持 H.264/H.265

  - 视频后处理器：去隔行，去噪，边缘/细节/颜色优化
- 存储
  - 高速 eMMC 扩展接口
  - MicroSD (TF) 卡槽

(请参考完整的《完整规格说明》)

这款令人难以置信的超小型主板，凭借其低功耗和高性能，能够安静流畅地运行 Android 7.1 或 Ubuntu 16.04 系统。

1.2 包装配件

ROC-RK3328-CC 标准套件的包装清单：

- ROC-RK3328-CC 主板
- Micro USB 线

对开发者来说，建议购买以下配件：

- 5V2A 电源适配器，一个好的电源是必须的
- USB 转串口适配器，用于串口调试
- eMMC 闪存，提供更高的系统性能和可靠性

CHAPTER 2

上手指南

ROC-RK3328-CC 支持从以下存储设备启动：

- SD 卡
- eMMC

我们需要将系统固件烧写到 SD 卡或 eMMC 里，这样开发板上电后才能正常启动进入操作系统。

SDCard Installer 是官方推荐的 SD 卡烧写工具，它基于 Etcher / Rock64 Installer 定制，实现了一站式的固件选择和烧录操作，让烧写工作变成轻松简单。

2.1 固件格式

固件有两种格式：

- 原始固件(raw firmware)
- RK固件(Rockchip firmware)

原始固件，是一种能以逐位复制的方式烧写到存储设备的固件，是存储设备的原始映像。原始固件一般烧写到 SD 卡中，但也可以烧写到 eMMC 中。 烧写原始固件有许多工具可以选用：

- 烧写 SD 卡
  - 图形界面烧写工具：
    - SDCard Installer (Linux/Windows/Mac)
    - Etcher (Linux/Windows/Mac)
  - 命令行烧写工具
    - dd (Linux)
- 烧写 eMMC
  - 图形界面烧写工具：

        * AndroidTool (Windows)
    - 命令行烧写工具：
        * upgrade_tool (Linux)
        * rkdeveloptool (Linux)

RK 固件，是以 Rockchip专有格式打包的固件，使用 Rockchip 提供的 upgrade_tool (Linux) 或 AndroidTool (Windows) 工具烧写到eMMC 闪存中。RK 固件是 Rockchip 的传统固件打包格式，常用于 Android 设备上。另外，Android 的 RK 固件也可以使用 SD Firmware Tool 工具烧写到 SD 卡中。

原始固件的通用性更好，因此，为了简单起见，官方**不再提供** RK 固件下载。

分区映像，是分区的映像数据，用于存储设备对应分区的烧写。例如，编译 Android SDK会构建出 `boot.img`、`kernel.img`和`system.img`等分区映像文件，`kernel.img` 会被写到eMMC 或 SD 卡的 “kernel” 分区。

2.2 下载和烧写固件

推荐使用 SDCard Installer 来烧写固件到 SD 卡。

如果使用 SDCard Installer 以外的工具，需要到下载页面去下载固件，然后再选择工具去烧写。

以下是支持的系统列表：

- Android 7.1.2
- Ubuntu 16.04
- Debian 9
- LibreELEC 9.0

注意：下载页面提供的全是原始固件，官方**不再提供**RK 固件。

根据所使用的操作系统来选择合适的工具去烧写原始固件：

- 烧写 SD 卡
    - 图形界面烧写工具：
        * SDCard Installer (Linux/Windows/Mac)
        * Etcher (Linux/Windows/Mac)
    - 命令行烧写工具
        * dd (Linux)
- 烧写 eMMC
    - 图形界面烧写工具：
        * AndroidTool (Windows)
    - 命令行烧写工具：
        * upgrade_tool (Linux)
        * rkdeveloptool (Linux)

2.3 开发板上电启动

在开发板上电启动前，确认以下事项：

- 可启动的 SD 卡或eMMC
- 5V2A 电源适配器
- Micro USB 线

然后按照以下步骤操作：

1. 将电源适配器拔出电源插座。
2. 使用 micro USB 线连接电源适配器和主板。
3. 插入可启动的 SD 卡或eMMC 之一（不能同时插入）。
4. 插入 HDMI 线、USB 鼠标或键盘（可选）。
5. 检查一切连接正常后，电源适配器上电。

CHAPTER 3

烧写 SD 卡

下面我们将介绍如何烧写固件到 SD 卡。如果没有特别说明，以下的固件都是指原始固件。关于固件的类型说明可以看这里。

推荐使用 SDCard Installer 来烧写固件到 SD 卡。

如果使用 SDCard Installer 以外的工具，需要到下载页面去下载固件，然后再选择工具去烧写。

以下是支持的系统列表：

- Android 7.1.2
- Ubuntu 16.04
- Debian 9
- LibreELEC 9.0

注意：下载页面提供的全是原始固件，官方**不再提供**RK 固件。

3.1 准备 SD 卡

作为启动盘，一张**优质可靠**、**高速读写**的 SD 卡，对于系统的稳定性来说是非常关键。现将 《如何准备一张 SD 卡(英文)》一文的要点摘录如下：

- 遇到启动或稳定性问题，有超过 95% 的可能是电源供应不足或 SD 卡问题（坏卡，坏读卡器，烧写SD卡时出错，卡读写太慢）。
- 推荐使用 Class 10 以上的 SD 卡，并使用专业工具去测试是否真卡。
- 尽量使用带校验的烧写工具。
- 如需将 SD 卡恢复至出厂设置，使用 SD Formatter 工具做格式化。
- 选择以下优质 SD 卡：

3.2 烧写工具

请根据所用主机的操作系统选择相应的烧写 SD 卡工具：

- 烧写 SD 卡
  - 图形界面烧写工具：
    - SDCard Installer (Linux/Windows/Mac)
    - Etcher (Linux/Windows/Mac)
  - 命令行烧写工具：
    - dd (Linux)

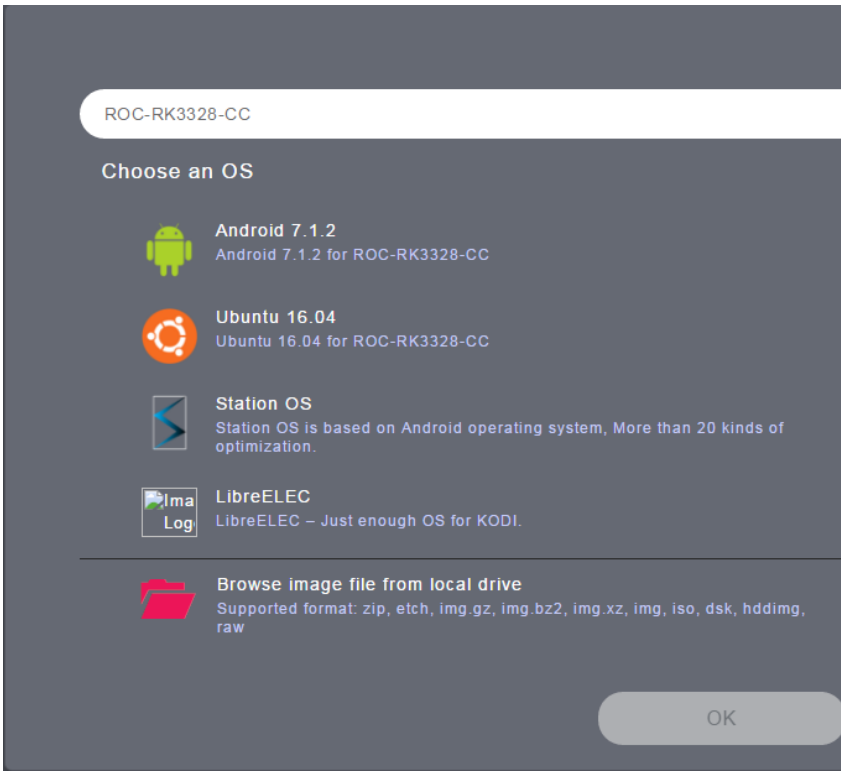
3.3 SDCard Installer

烧写原始固件，最轻松的方式就是使用官方的 SDCard Installer，它基于 Etcher / Rock64 Installer 定制，实现了一站式的固件选择和烧录操作，让烧写工作变成轻松简单。

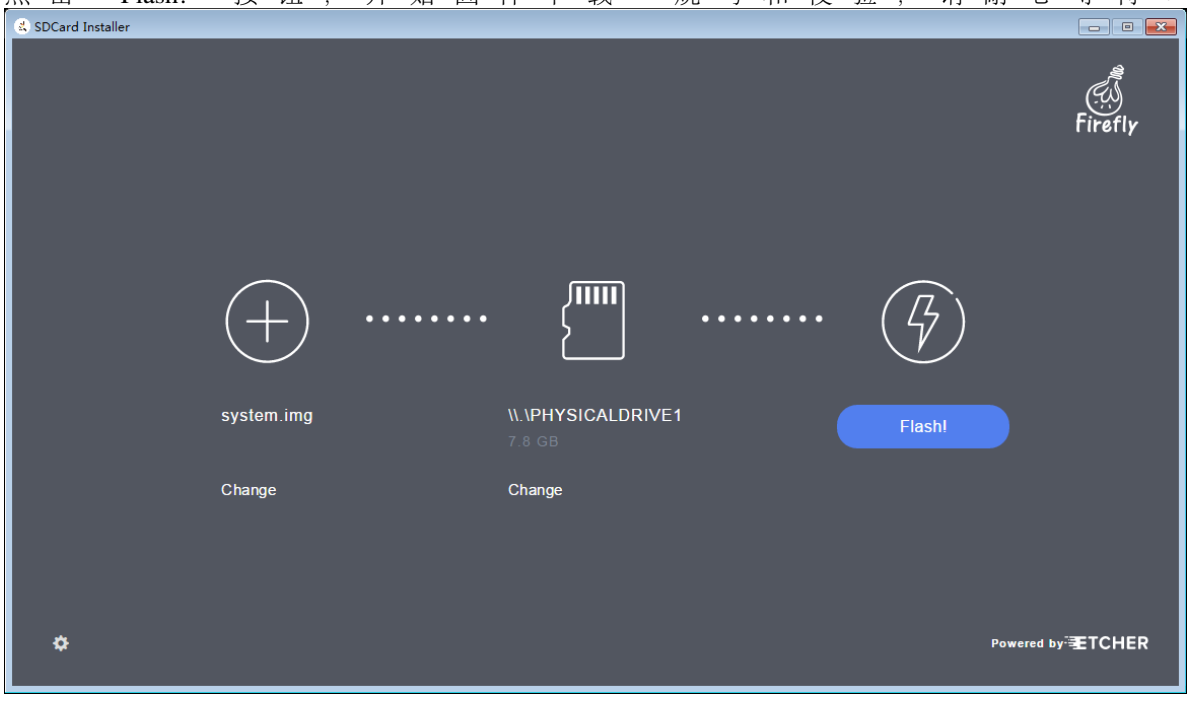
SDCard Installer 节省了搜索开发板可用固件的时间。你只需要选择开发板、操作系统，插入 SD 卡，点击烧写按钮即可完成整个写卡工作，实在简单方便。

安装使用说明：

1. 到 下载页面去下载SDCard Installer。
2. 安装运行：
   - Windows： 解压后运行安装程序，按照提示安装到系统，之后在开始菜单里找到 SDCard Installer，并**以管理员身份**打开。
   - Linux： 解压后运行其中的 `.AppImage` 文件即可。
   - Mac： 直接双击 `.dwg` 文件，拖动安装到系统或直接运行。
3. 点击 “Choose an OS” 按钮， 在 “Please select your device” 组合框中选择 “ROC-RK3328-CC”。



4. 可用的固件列表将从网络更新，如下图所示：
5. 选择所需的操作系统和版本，并点击 “OK” 按钮确认。另外也可以从文件管理器中选择本地的一个固件文件，拖放到 `SDCard Installer`。
6. 插入 SD 卡，工具应该会自动选中该卡；如果插有多张 SD 卡，可以点击 “Change” 按钮进行选择。
7. 点击 “Flash!” 按钮，开始固件下载、烧写和校验，请耐心等待。

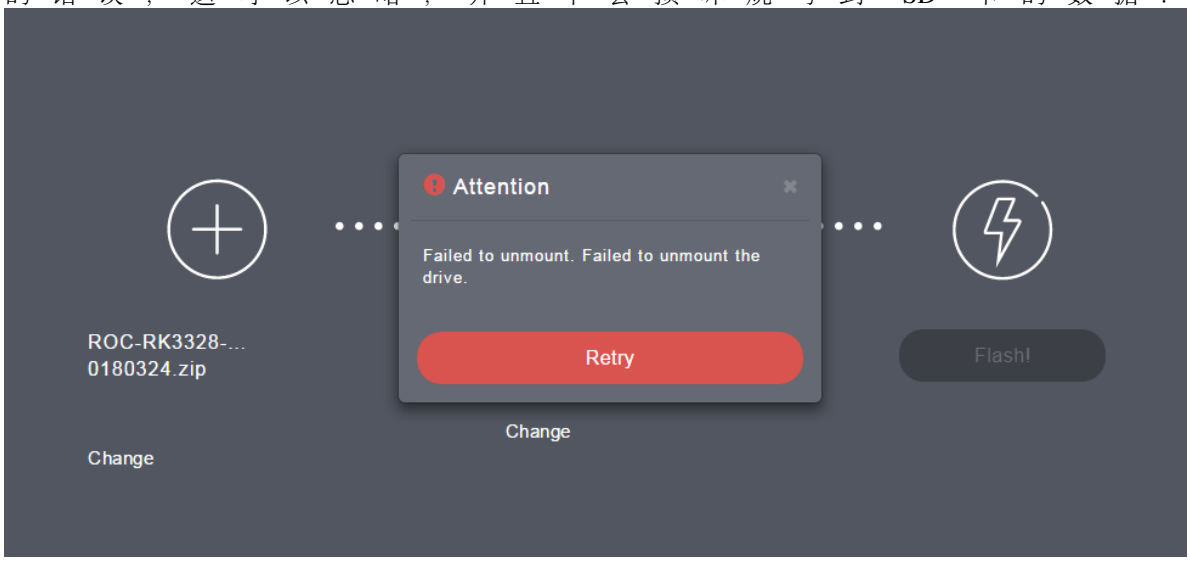


注意事项：

- SDCard Installer 在 Windows 下运行需要管理员权限，请用鼠标右键点击应用图标，在弹出菜单中选择

以管理员身份运行。

- 有时，当进度达到 99% 或 100% 时，可能会出现卸载 SD 卡的错误，这可以忽略，并且不会损坏烧写到 SD 卡的数据：



- 在线下载的固件会缓存到本地目录，下次烧写时不用重新下载。缓存目录可以点击左下角的设置按钮，在 “Download Location:” 处设置。

3.4 Etcher

Etcher 与 SDCard Installer 相比，少了固件选择的集成，但代码比较新。如果 SDCard Installer 在烧写 SD 卡中出错，或有什么问题，可以尝试使用 Etcher 去烧写，此时直接使用 SDCard Installer 缓存目录里的固件即可。

Etcher 可以到Etcher 官网去下载，安装和使用过程与 SDCard Installer 比较类似，这里就不再重复。

3.5 dd

dd 是 Linux 下常用的命令行工具。

首先，插入SD卡，如果被文件管理器自动挂载，则先将其卸载。

然后通过检查内核的日志查找 SD 卡的设备文件：

```
dmesg | tail
```

如果设备文件为 `/dev/mmcblk0`，使用 `dd` 命令去烧录：

```
sudo dd if=/path/to/your/raw/firmware of=/dev/mmcblk0 conv=notrunc
```

烧写一般所需时间较长，但上面的命令不会显示烧写进度，只能一直等待命令的完成。此时，我们可以使用另一个工具 `pv` 去实现进度条的显示。

安装 `pv`：

```
sudo apt-get install pv
```

然后利用管道操作显示烧写进度：

```
pv -tpreb /path/to/your/raw/firmware | sudo dd of=/dev/mmcblk0 conv=notrunc
```

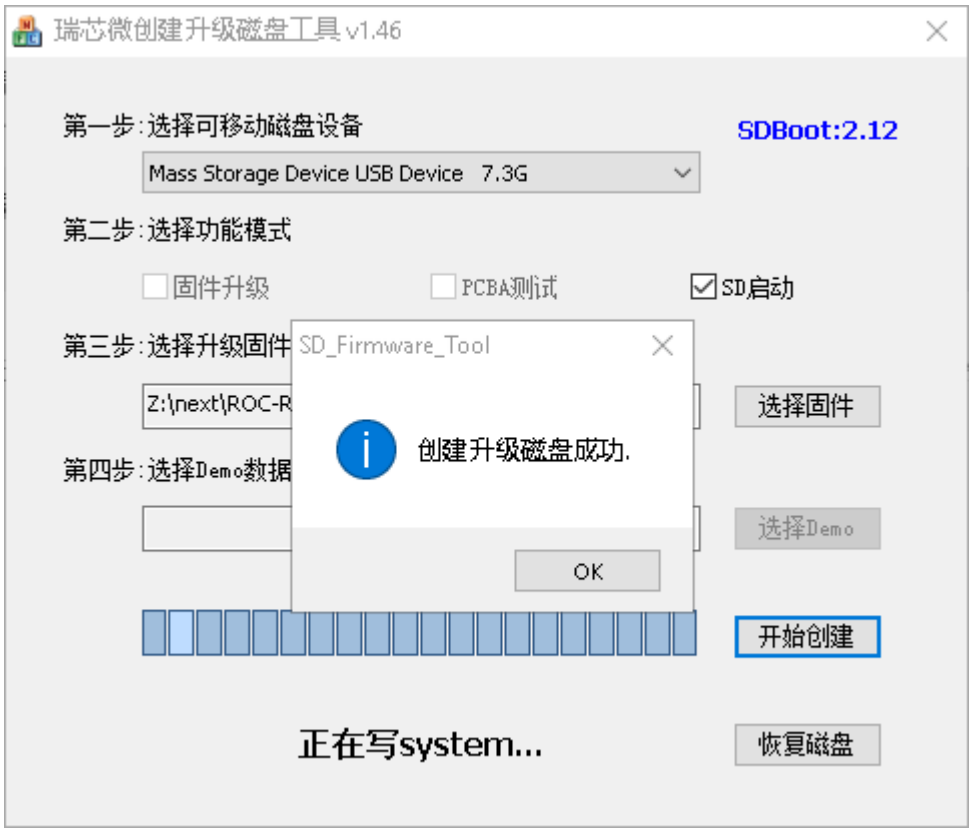
3.6 SD Firmware Tool

注意：以下介绍的是如何将RK 固件烧写到 SD 卡。

首先，到SD Firmware Tool 下载页面去下载 `SD_Firmware_Tool`，并解压。

运行 `SD_Firmware_Tool.exe`:

1. 插入 SD 卡。
2. 从组合框中选择 SD 卡对应的设备。
3. 勾选 “SD启动” 选项。
4. 点击 “选择固件” 按钮，在文件对话框中选择固件。
5. 点击 “开始创建” 按钮。
6. 然后会显示警告对话框，选择 “是” 来确保选择了正确的SD卡设备。
7. 等待操作完成，直到提示成功对话框出现:


瑞芯微创建升级磁盘工具 v1.46
第一步:选择可移动磁盘设备
SDBoot:2.12
Mass Storage Device USB Device 7.3G
第二步:选择功能模式
固件升级
PCBA测试
SD启动
第三步:选择升级固件
SD_Firmware_Tool
Z:\next\ROC-R
选择固件
创建升级磁盘成功.
第四步:选择Demo数据
选择Demo
OK
开始创建
正在写system...
恢复磁盘

8. 拔出 SD 卡。

CHAPTER 4

烧写 eMMC

4.1 启动模式

eMMC 一般都是直接焊在主板上，有些虽然是可插拔的，但没有专用的读卡器，因此需要使用板载烧写的方式来更新固件，即板上跑一个小系统，负责从主机或其它存储介质读取固件，再烧写到 eMMC 上。

取决于 eMMC 现存的内容，开发板有两种特殊的启动模式： Rockusb 模式 和 Maskrom 模式。

通常只需要进入 Rockusb 模式 即可升级现有的 Android 或 Ubuntu 系统。这种方式升级方式一般适用于 RK 固件或分区映像。

Maskrom 模式 则是系统未能识别到合法的启动设备而进入的模式。烧写原始固件到 eMMC 必须要进入该模式。

4.1.1 Rockusb 模式

开发板启动后，CPU 如果在 eMMC 中找到有效的 IDB (IDentity Block)，它将从 eMMC 读取并加载 bootloader，并将执行控制权交给它。

如果 bootloader 检测到 Recovery 按钮按下并且 USB 已连接，它就会进入 Rockusb 模式，等待来自主机的命令。

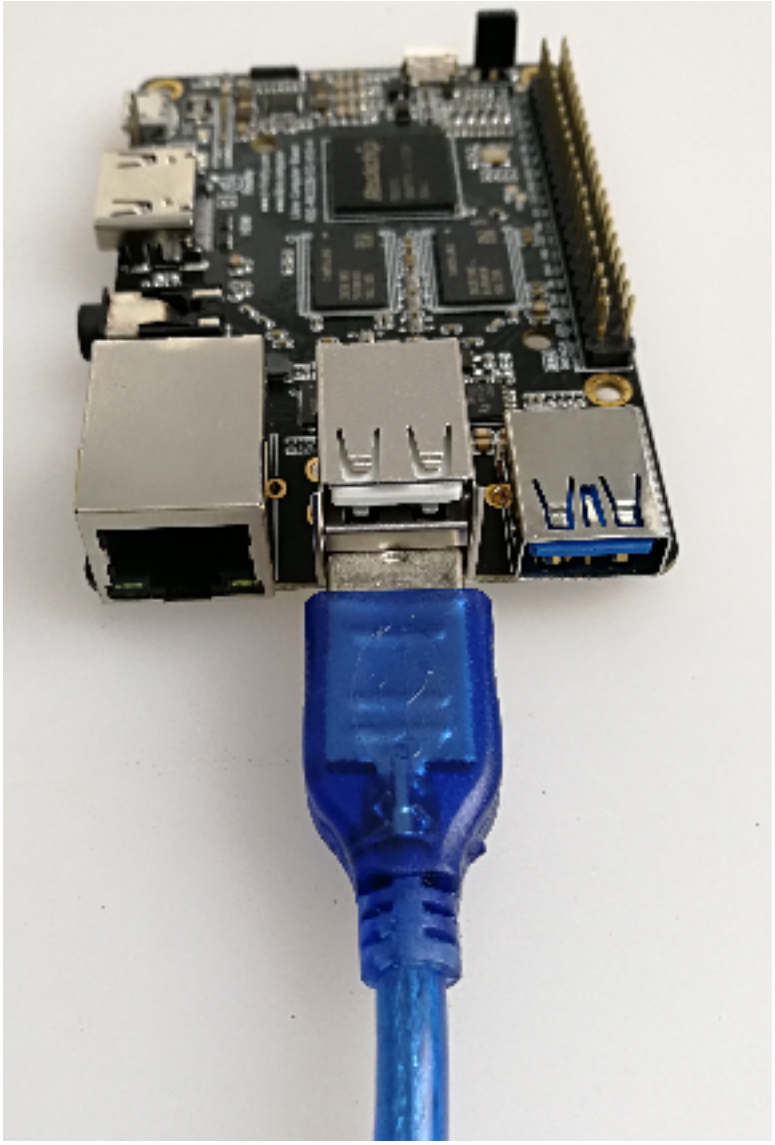
进入 Rockusb 模式 的准备工作：

- 5V2A 电源适配器。
- 连接电源适配器和开发板的 Micro USB 线。
- 用来连接电脑 PC 以及开发板的公对公 USB 线。
- eMMC 。

操作步骤:

1. 将所有 USB 线（包括 Micro USB 线和公对公 USB 线）拔出开发板，以保持开发板断电。
2. 安装好 eMMC，拔出 SD 卡。

3. 使用公对公 USB 线将主机的 USB 端口与开发板的双层 USB 端口中靠近电路板的 OTG 端口相连:



4. 按住开发板上的 RECOVERY 按键。
5. 将 Micro USB 线插入到开发板中，让开发板上电。
6. 等待大概 3 秒左右松开 RECOVERY 按键。

4.1.2 Maskrom 模式

如果开发板上电后遇到以下情况之一:

- eMMC 内容为空。
- eMMC 上的 bootloader 损坏。
- 将 eMMC 数据/时钟引脚接地，eMMC读取数据失败。

CPU 在 eMMC 中就会找不到有效的 IDB (IDentity Block)，转而执行一段小型的 ROM 代码，等待主机通过 USB 上传 bootloader 来初始化 DDR 内存并进入升级状态。这种模式称为 Maskrom 模式 。

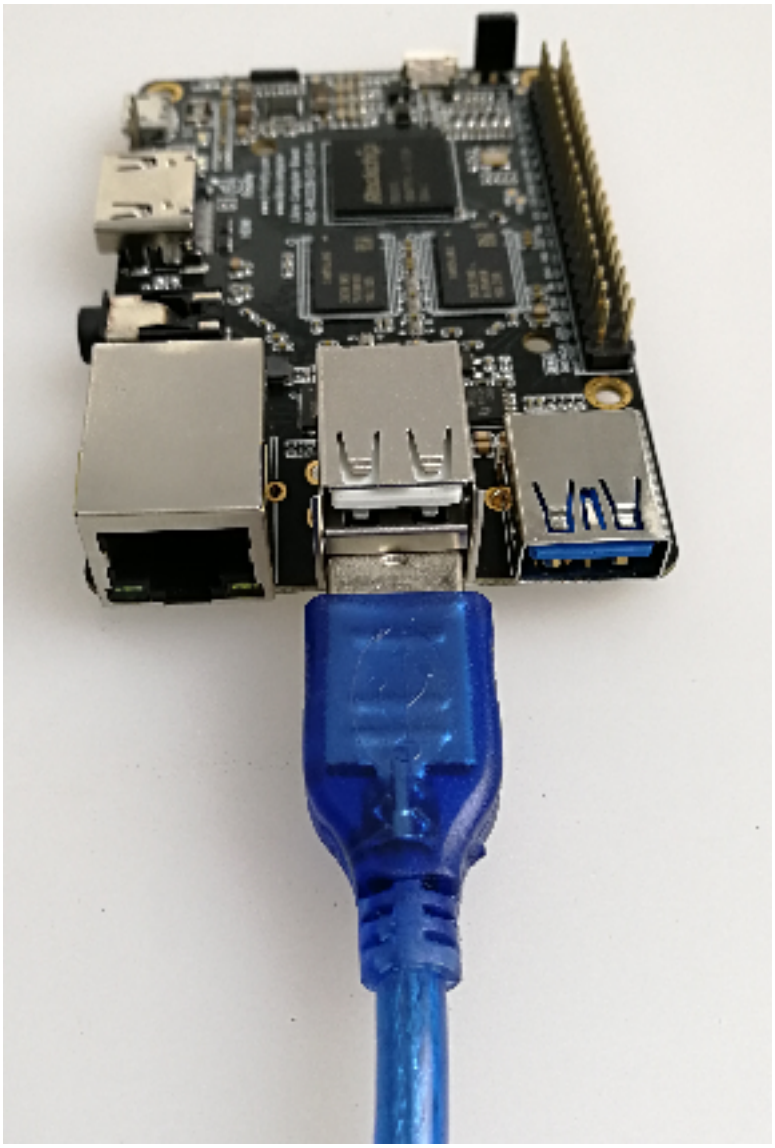
强制进入 MaskRom 模式 涉及到硬件操作，具有一定的风险，因此操作上需要 **非常谨慎** 。

进入 Maskrom 模式 的准备工作:

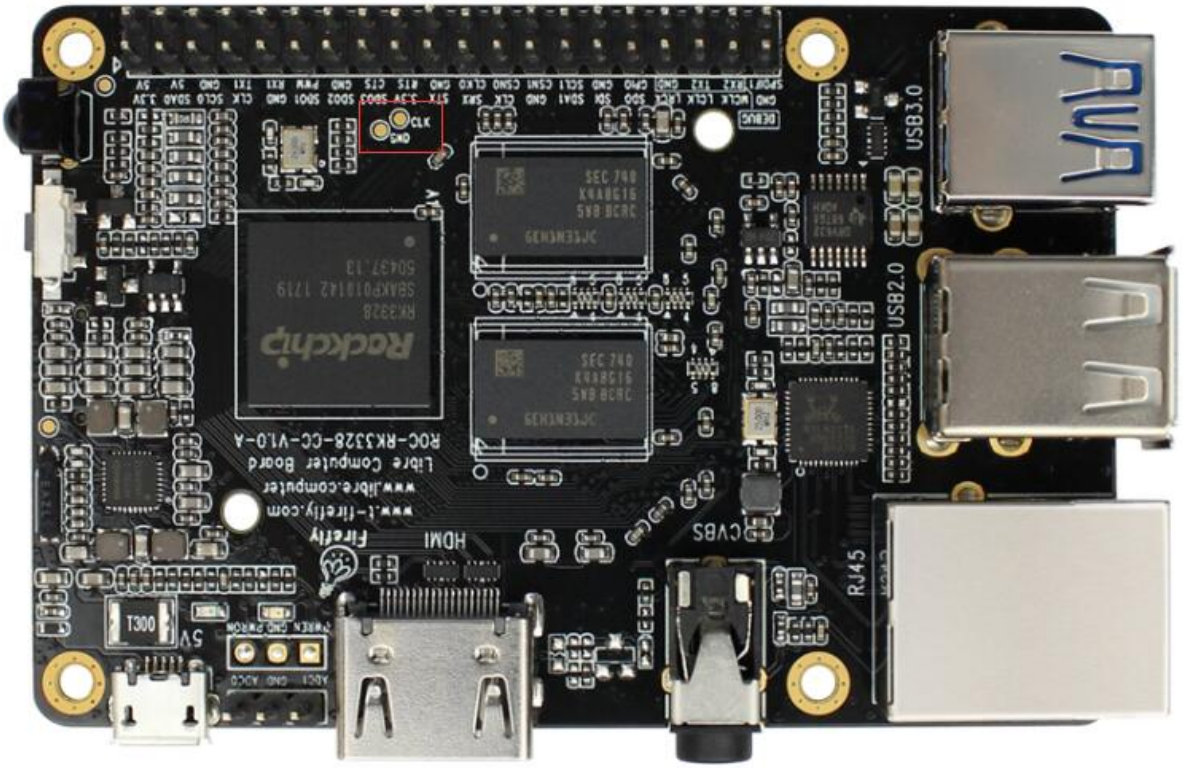
- 5V2A 电源适配器。
- 用来连接电源适配器和开发板的 Micro USB 线。
- 连接电脑 PC 和开发板的公对公 USB 线。
- 用于将 eMMC 时钟引脚短接到地的金属镊子。
- eMMC 。

操作步骤:

1. 将所有 USB 线（包括 Micro USB 线和公对公 USB 线）拔出开发板，以保持开发板断电。
2. 安装好 eMMC，拔出 SD 卡。
3. 使用公对公 USB 线将主机的 USB 端口与开发板的双层 USB 端口中靠近电路板的 OTG 端口相连:



4. 找到开发板上预留的 eMMC 的 CLK 引脚和 GND 脚，见下图:



5. 用金属镊子短接 eMMC 的 CLK 和 GND 焊盘，并保持短接良好。
6. 将 Micro USB 线插入到开发板中，让开发板上电。
7. 保持住一秒后松开镊子。

4.2 烧写工具

请根据所用主机的操作系统选择相应的烧写 eMMC 工具：

- 烧写 eMMC
  - 图形界面烧写工具：
    - AndroidTool (Windows)
  - 命令行烧写工具：
    - upgrade_tool (Linux)
    - rkdeveloptool (Linux)

4.3 AndroidTool

AndroidTool 是 Windows 下用来烧写原始固件、RK 固件和分区映像到 eMMC 的工具。

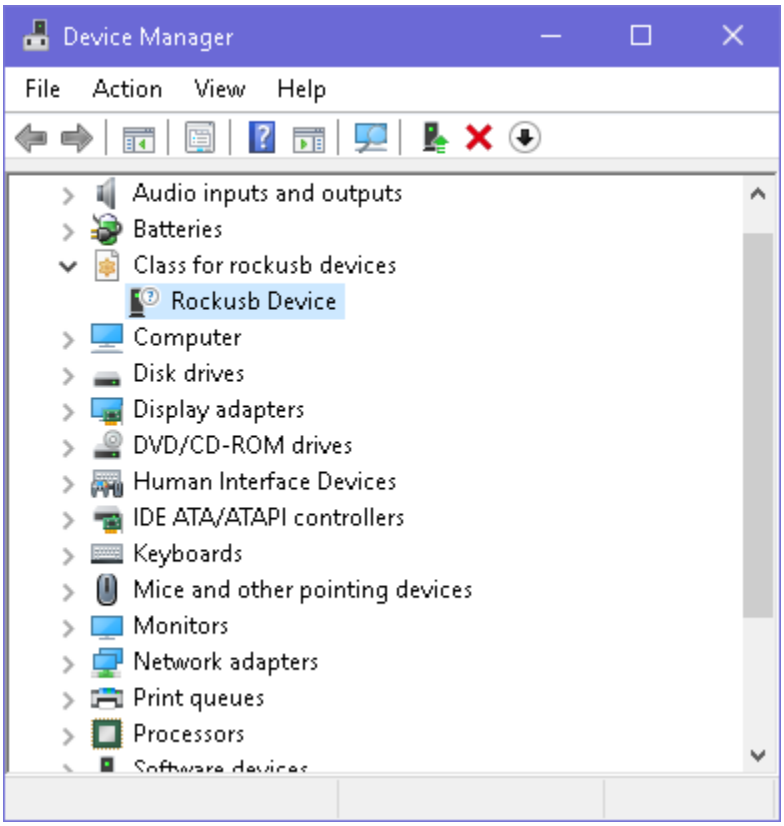
使用 AndroidTool 之前， 应先安装 Rockusb 驱动，然后再安装运行 AndroidTool。

4.3.1 安装 Rockusb 驱动

下载 DriverAssistant，解压后运行里面的 `DriverInstall.exe`：

点击 “驱动安装” 按钮安装驱动；如果想卸载驱动，则点击 “驱动卸载” 按钮。

若设备处于 Rockusb 模式 或 Maskrom 模式，在设备管理器中会出现 “Rockusb Device”：

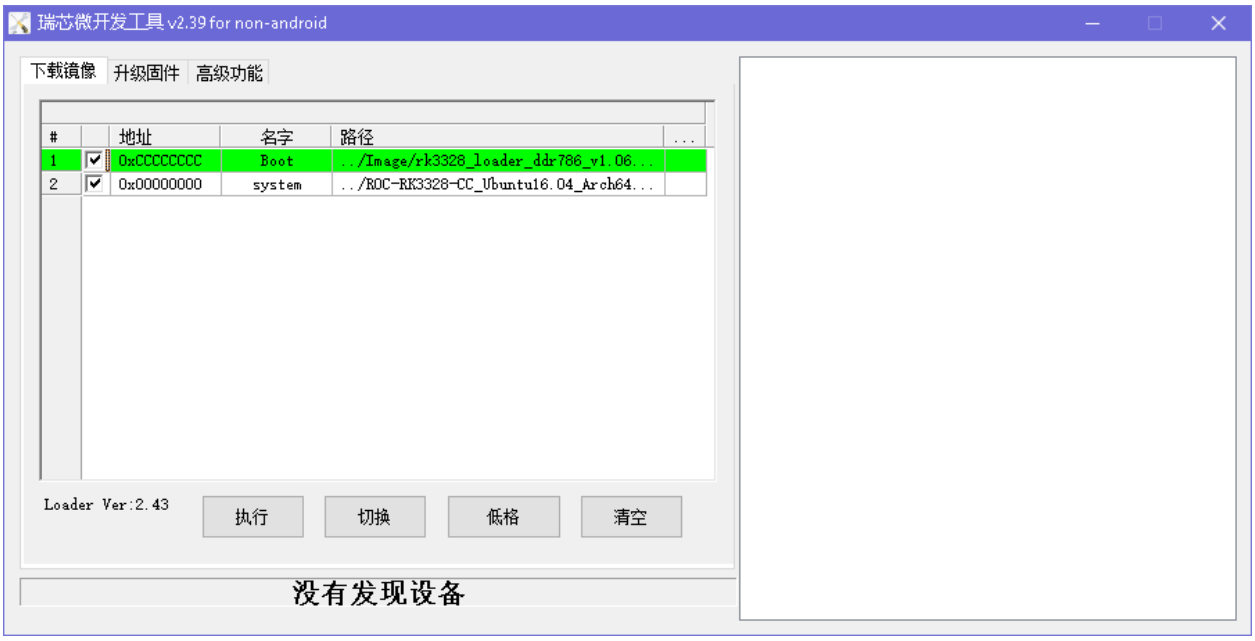


这表示驱动安装成功。

4.3.2 安装 AndroidTool

AndroidTool 下载地址

下载并解压后运行里面的 `AndroidTool.exe`：



若设备处于 Rockusb 模式，状态行将显示 “发现一个LOADER设备”。

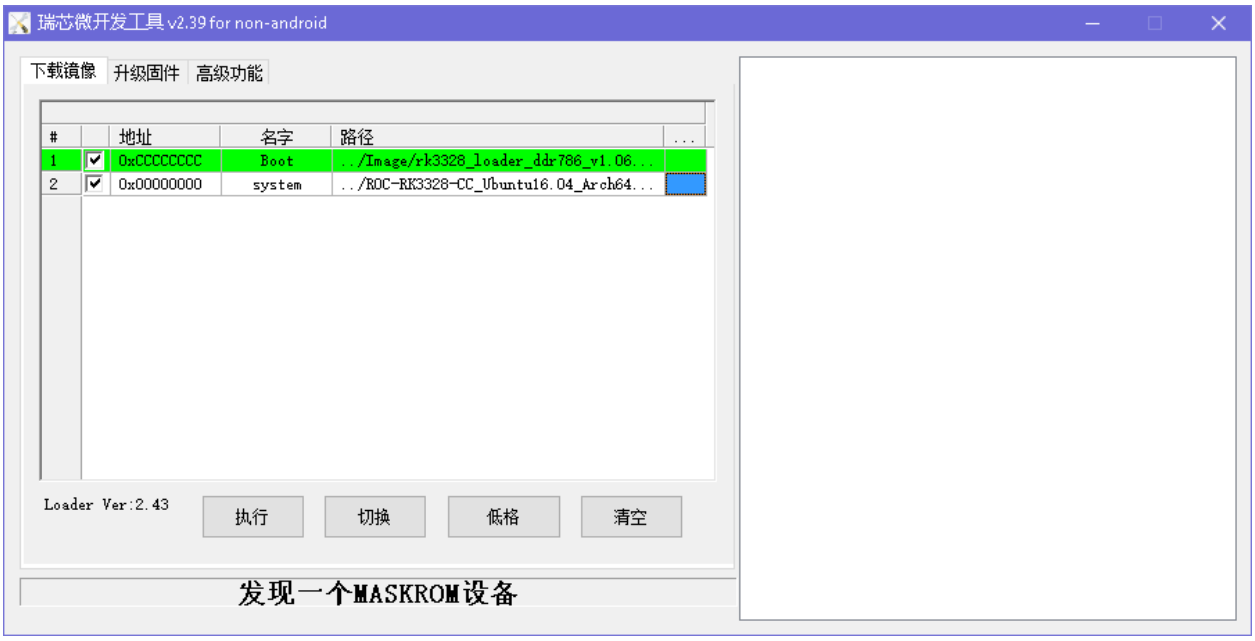
若设备处于 Maskrom 模式，状态行将显示 “发现一个MASKROM设备”。

4.3.3 烧写原始固件

原始固件需要从 eMMC 的偏移地址为 0 的位置开始烧写。但在 Rockusb 模式 下无法做到这点，因为所有 LBA 写入操作会偏移 0x2000 个扇区（即 LBA0 对应于存储设备上第 0x2000 个扇区）。因此，开发板必须强制进入 Maskrom 模式 才能烧写原始固件。

使用 AndroidTool 烧写原始固件到 eMMC 的步骤如下:

1. 强制设备进入 Maskrom 模式。
2. 运行 AndroidTool。
3. 打开 “下载镜像” 制表页。
4. 保持表格的第一行不变， 使用默认的 “Loader” 文件。
5. 点击第二行右侧的空白单元格，在弹出的文件对话框里打开原始固件文件。
6. 点击 “执行” 按钮开始烧写。



4.3.4 烧写 RK 固件

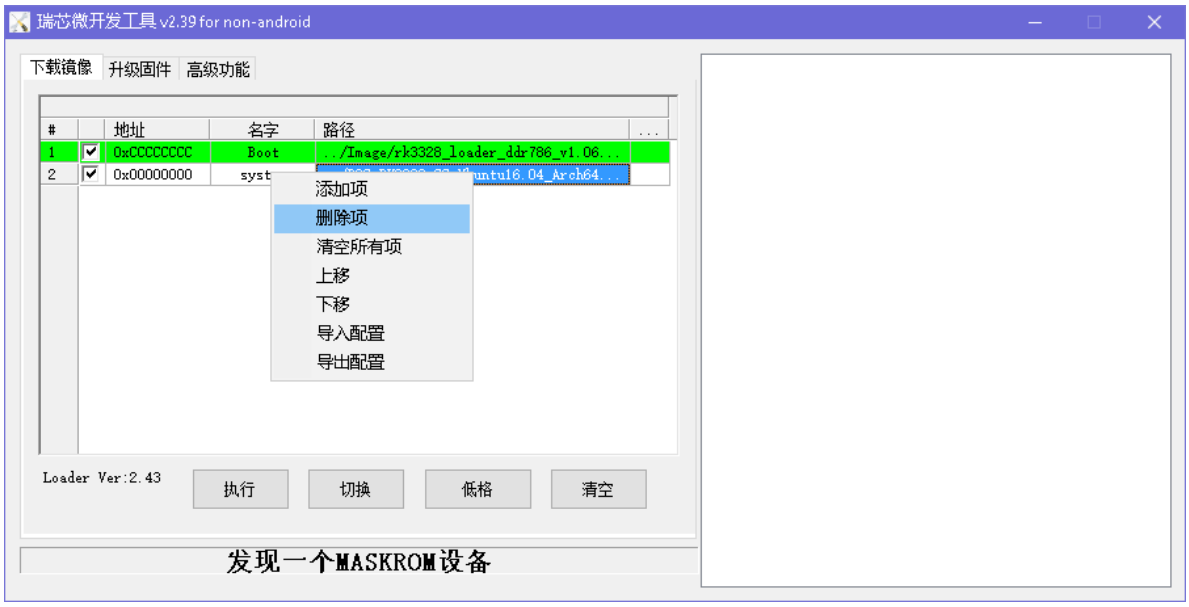
使用 AndroidTool 烧写 RK 固件到 eMMC 的步骤如下:

1. 强制设备进入 Rockusb 模式 或 Maskrom 模式。
2. 运行 AndroidTool。
3. 打开 “升级固件” 制表页。
4. 点击 “固件” 按钮， 在弹出的文件对话框里打开RK 固件文件。
5. 固件版本号、loader 版本号和芯片信息会从固件中读取并显示。
6. 点击 “升级” 按钮烧录。

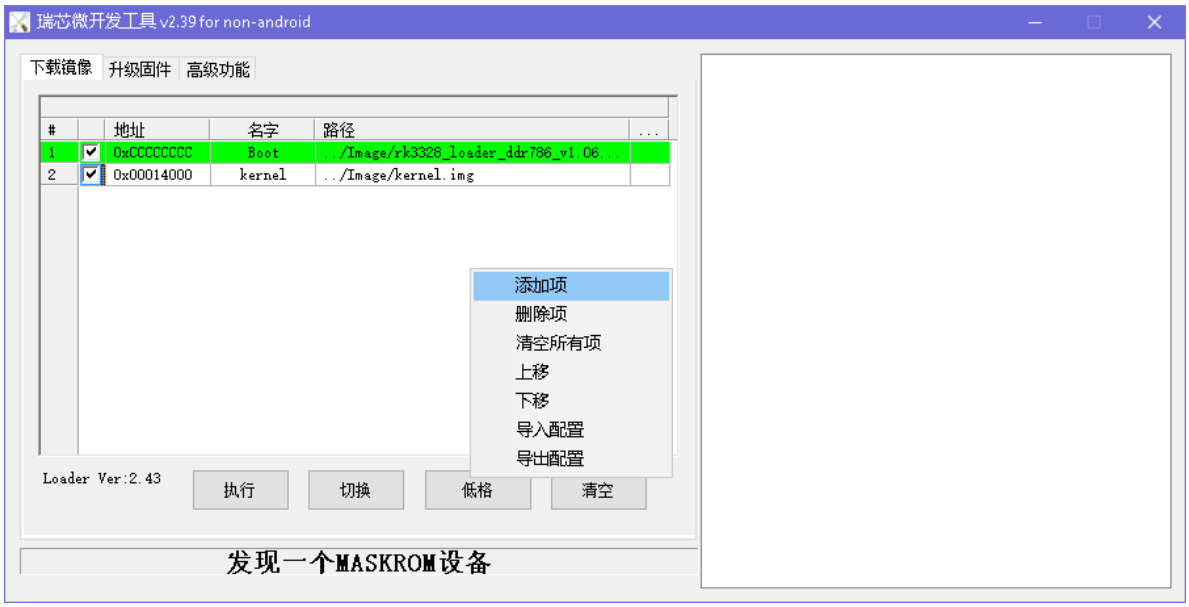
4.3.5 烧写分区映像

使用 AndroidTool 烧写分区映像到 eMMC 的步骤如下:

1. 强制设备进入 Rockusb 模式 或 Maskrom 模式。
2. 运行 `AndroidTool`。
3. 打开 “下载镜像” 制表页。
4. 保持表格第一行不变。
5. 鼠标右键点击其它行，在弹出菜单中选择 “删除项”， 重复直至删除第一行除外的所有行。



6. 鼠标右键点击表格，在弹出菜单中选择 “添加项” 以便添加分区映像：
   - 选中第一个单元格上的复选框。
   - 填入 `parameter.txt` 中该分区的起始扇区作为烧写地址（如果是 Maskrom 模式 则须再加上 `0x2000`）。
   - 单击右侧空白单元格，在弹出的文件对话框里打开对应的分区映像文件。



7. 点击 “执行” 按钮烧录。

注意：

- 您可以通过重复步骤 6 将多个分区映像烧写到闪存。
- 通过取消选中地址单元格前面的复选框，可以跳过此分区的烧写。
- 在 Maskrom 模式 中， `parameter.txt` 中分区的起始扇区必须再加上 `0x2000` 作为烧写地址。参见《分区偏移量》一章以便使用脚本获取该地址。

4.4 upgrade_tool

upgrade_tool 是 Linux 下用来烧写原始固件、RK 固件和分区映像到 eMMC 的工具，是 Rockchip 提供的闭源命令行工具。

4.4.1 安装 upgrade_tool

upgrade_tool 下载链接

下载 upgrade_tool，并安装到 Linux 系统上：

```
unzip Linux_Upgrade_Tool_v1.24.zip
cd Linux_UpgradeTool_v1.24
sudo mv upgrade_tool /usr/local/bin
sudo chown root:root /usr/local/bin/upgrade_tool
sudo chmod 0755 /usr/local/bin/upgrade_tool
```

然后根据此处的说明去添加 `udev` 规则。这是为了让普通用户有权限烧写 Rockchip 设备。如果跳过这步，那么所有的烧写命令均需在前面加 `sudo` 才能成功执行。

4.4.2 烧写原始固件

原始固件需要从 eMMC 的偏移地址为 0 的位置开始烧写。但在 Rockusb 模式 下所有 LBA 写入操作会偏移 0x2000 个扇区（即 LBA0 对应于存储设备上的第 0x2000 个扇区）。因此，开发板必须强制进入 Maskrom 模式 才能烧写原始固件。

使用 upgrade_tool 烧写原始固件到 eMMC 的步骤如下：

1. 强制设备进入 Maskrom 模式。
2. 运行以下命令：

```
upgrade_tool db     out/u-boot/rk3328_loader_ddr786_v1.06.243.bin
upgrade_tool wl 0x0 out/system.img
upgrade_tool rd     # 重置并启动设备
```

其中：

- `rk3328_loader_ddr786_v1.06.243.bin` 是编译 `U-Boot` 时复制进去的 loader 文件，也直接到此处下载 `rk3328_loader_xxx.bin` 文件。
- `system.img` 是打包后的原始固件，也可以是官网上下载并解压后的原始固件文件。

4.4.3 烧写 RK 固件

使用 upgrade_tool 烧写 RK 固件到 eMMC 的步骤如下：

1. 强制设备进入 Rockusb 模式 或 Maskrom 模式。
2. 使用 `upgrade_tool` 烧写 RK 固件：

```
upgrade_tool uf update.img
```

4.4.4 烧写分区映像

取决开发板原有的固件，烧写分区映像到 eMMC 的指令会有所不同。

原始固件

如果开发板原有系统是原始固件，那么很可能使用了 GPT 分区方案。每个分区的预定义偏移量和大小可以在 SDK 里的 build/partitions.sh 中找到，可以参考《分区偏移量》一章。

使用 upgrade_tool 烧写 分区映像的步骤如下：

1. 强制设备进入 Maskrom 模式
2. 使用 upgrade_tool 烧写分区映像：

```
upgrade_tool db         out/u-boot/rk3328_loader_ddr786_v1.06.243.bin
upgrade_tool wl 0x40    out/u-boot/idbloader.img
upgrade_tool wl 0x4000  out/u-boot/uboot.img
upgrade_tool wl 0x6000  out/u-boot/trust.img
upgrade_tool wl 0x8000  out/boot.img
upgrade_tool wl 0x40000 out/linaro-rootfs.img
upgrade_tool rd         # 重置并启动设备
```

RK 固件

如果开发板原有系统是 RK 固件，那么它使用 parameter 文件作为分区方案，这样就可直接使用分区名称来烧写分区映像：

1. 强制设备进入 Rockusb 模式。
2. 使用 upgrade_tool 烧写分区映像：

```
upgrade_tool di -b /path/to/boot.img
upgrade_tool di -k /path/to/kernel.img
upgrade_tool di -s /path/to/system.img
upgrade_tool di -r /path/to/recovery.img
upgrade_tool di -m /path/to/misc.img
upgrade_tool di resource /path/to/resource.img
upgrade_tool di -p parameter    # 烧写 parameter
upgrade_tool ul bootloader.bin # 烧写 bootloader
```

注意：

- -b 是 boot 分区的预定义缩写。如果没有缩写可用，请改为分区名称，例如上述例子中的 resource。
- 可根据 《参数文件格式》 这份文档的说明自定义内核参数和分区布局。分区布局更改后，必须先重新烧写该 parameter 文件，方可重新烧写受影响的相应分区。

4.4.5 常见问题

如果由于闪存问题而出现错误，可以尝试使用低格或擦除闪存：

```
upgrade_tool lf    # 低格闪存
upgrade_tool ef    # 擦除闪存
```

4.5 rkdeveloptool

rkdeveloptool j是 Linux 下用来烧写原始固件和分区映像到 eMMC 的工具。它**不支持**烧写 RK 固件。

rkdeveloptool 是由 Rockchip 开发的命令行烧写工具，是闭源工具 upgrade_tool 的开源替代品。在日常使用中，一般都能取代 upgrade_tool。

4.5.1 安装 rkdeveloptool

首先是下载、编译和安装 rkdeveloptool:

```
#install libusb and libudev
sudo apt-get install pkg-config libusb-1.0 libudev-dev libusb-1.0-0-dev dh-autoreconf
# clone source and make
git clone https://github.com/rockchip-linux/rkdeveloptool
cd rkdeveloptool
autoreconf -i
./configure
make
sudo make install
```

然后根据此处的说明去添加 udev 规则。这是为了让普通用户有权限烧写 Rockchip 设备。如果跳过这步，则所有的烧写命令均需在前面加 sudo 才能成功执行。

4.5.2 烧写原始固件

原始固件需要从 eMMC 的偏移地址为 0 的位置开始烧写。但在 Rockusb 模式 下所有 LBA 写入操作会偏移 0x2000 个扇区（即 LBA0 对应于存储设备上的第 0x2000 个扇区）。因此，开发板必须强制进入 Maskrom 模式 才能烧写原始固件。

使用 rkdeveloptool 烧写原始固件到 eMMC 的步骤如下:

1. 强制设备进入 Maskrom 模式。
2. 运行以下命令:

```
rkdeveloptool db            out/u-boot/rk3328_loader_ddr786_v1.06.243.bin
rkdeveloptool wl 0x0        out/system.img
rkdeveloptool rd            # 重置并启动设备
```

其中:

- rk3328_loader_ddr786_v1.06.243.bin 是编译 U-boot 时复制进去的 loader 文件，也直接到此处下载 rk3328_loader_xxx.bin 文件。
- system.img 是打包后的原始固件，也可以是官网上下载并解压后的原始固件文件。

4.5.3 烧写分区映像

以下的说明**仅适用**于开发板原有系统是原始固件时的分区映像烧写。每个分区的预定义偏移量和大小可以在 SDK 里的 build/partitions.sh 中找到，可以参考《分区偏移量》一章。

使用 rkdeveloptool 烧写分区映像的步骤如下:

1. 强制设备进入 Maskrom 模式。

2. 运行以下命令：

```
rkdeveloptool db            out/u-boot/rk3328_loader_ddr786_v1.06.243.bin
rkdeveloptool wl 0x40       out/u-boot/idbloader.img
rkdeveloptool wl 0x4000     out/u-boot/uboot.img
rkdeveloptool wl 0x6000     out/u-boot/trust.img
rkdeveloptool wl 0x8000     out/boot.img
rkdeveloptool wl 0x40000    out/linaro-rootfs.img
rkdeveloptool rd            # 重置并启动设备
```

4.6 udev

创建 `/etc/udev/rules.d/99-rk-rockusb.rules`，并插入以下内容[1]。 如有必要，用实际 Linux 组替换 `users` 组：

```
SUBSYSTEM!="usb", GOTO="end_rules"

# RK3036
ATTRS{idVendor}=="2207", ATTRS{idProduct}=="301a", MODE="0666", GROUP="users"
# RK3229
ATTRS{idVendor}=="2207", ATTRS{idProduct}=="320b", MODE="0666", GROUP="users"
# RK3288
ATTRS{idVendor}=="2207", ATTRS{idProduct}=="320a", MODE="0666", GROUP="users"
# RK3328
ATTRS{idVendor}=="2207", ATTRS{idProduct}=="320c", MODE="0666", GROUP="users"
# RK3368
ATTRS{idVendor}=="2207", ATTRS{idProduct}=="330a", MODE="0666", GROUP="users"
# RK3399
ATTRS{idVendor}=="2207", ATTRS{idProduct}=="330c", MODE="0666", GROUP="users"

LABEL="end_rules"
```

重新加载 udev 规则：

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

4.7 分区偏移量

4.7.1 GPT 分区

如果使用原始固件，那么系统很可能使用了 GPT 分区方案。每个分区的预定义偏移量和大小可以在 SDK 里的 `build/partitions.sh` 中找到。

分区映像的偏移量可以通过以下命令获得（假设位于 Firefly Linux SDK 的目录中）：

```
(. build/partitions.sh ; set | grep _START | \
while read line; do start=${line%=*}; \
printf "%-10s 0x%08x\n" ${start%_START*} ${!start}; done )
```

会得到：

```
ATF         0x00006000
BOOT        0x00008000
LOADER1     0x00000040
LOADER2     0x00004000
RESERVED1   0x00001f80
RESERVED2   0x00002000
ROOTFS      0x00040000
SYSTEM      0x00000000
```

4.7.2 parameter

如果使用 RK 固件，那么系统是使用 parameter.txt 文件来定义分区方案，该文件的格式参见文档《参数文件格式》 。

这里有一个 Linux 下的 Bash 脚本能列出 parameter.txt 中的分区偏移量:

```
#!/bin/sh

PARAMETER_FILE="$1"
[ -f "$PARAMETER_FILE" ] || { echo "Usage: $0 <parameter_file>"; exit 1; }

show_table() {
    echo "$1"
    echo "--------"
    printf "%-20s %-10s %s\n" "NAME" "OFFSET" "LENGTH"
    for PARTITION in `cat ${PARAMETER_FILE} | grep '^CMDLINE' | sed 's/ //g' | sed 's/
↪.*:\(0x.*[^)])\).*/\1/' | sed 's/,/ /g'`; do
        NAME=`echo ${PARTITION} | sed 's/\(.*\)(\(.*\))/\2/'`
        START=`echo ${PARTITION} | sed 's/.*@\(.*\)(.*)/\1/'`
        LENGTH=`echo ${PARTITION} | sed 's/\(.*\)@.*/\1/'`
        START=$((START + $2))
        printf "%-20s 0x%08x %s\n" $NAME $START $LENGTH
    done
}

show_table "Rockusb Mode" 0
echo
show_table "Maskrom Mode" 0x2000
```

将这个脚本保存为 /usr/local/bin/show_rk_parameter.sh ，然后添加脚本的执行权限。

下面是一个显示 RK3328 Android SDK 中定义的分区偏移量的例子:

```
$ show_rk_parameter.sh device/rockchip/rk3328/parameter.txt
Rockusb Mode
--------
NAME                 OFFSET     LENGTH
uboot                0x00002000 0x00002000
trust                0x00004000 0x00004000
misc                 0x00008000 0x00002000
baseparamer          0x0000a000 0x00000800
resource             0x0000a800 0x00007800
kernel               0x00012000 0x00010000
boot                 0x00022000 0x00010000
recovery             0x00032000 0x00010000
backup               0x00042000 0x00020000
```

(continues on next page)

(续上页)

```
cache                0x00062000 0x00040000
metadata             0x000a2000 0x00008000
kpanic               0x000aa000 0x00002000
system               0x000ac000 0x00300000
userdata             0x003ac000 -

Maskrom Mode
--------
NAME                 OFFSET     LENGTH
uboot                0x00004000 0x00002000
trust                0x00006000 0x00004000
misc                 0x0000a000 0x00002000
baseparamer          0x0000c000 0x00000800
resource             0x0000c800 0x00007800
kernel               0x00014000 0x00010000
boot                 0x00024000 0x00010000
recovery             0x00034000 0x00010000
backup               0x00044000 0x00020000
cache                0x00064000 0x00040000
metadata             0x000a4000 0x00008000
kpanic               0x000ac000 0x00002000
system               0x000ae000 0x00300000
userdata             0x003ae000 -
```

CHAPTER 5

串口调试

如果正在进行 U-Boot 或内核开发，USB串口适配器（USB 转串口TTL 适配器的简称）对于检查系统启动日志非常有用，特别是在没有图形桌面显示的情况下。

5.1 准备 USB 串口适配器

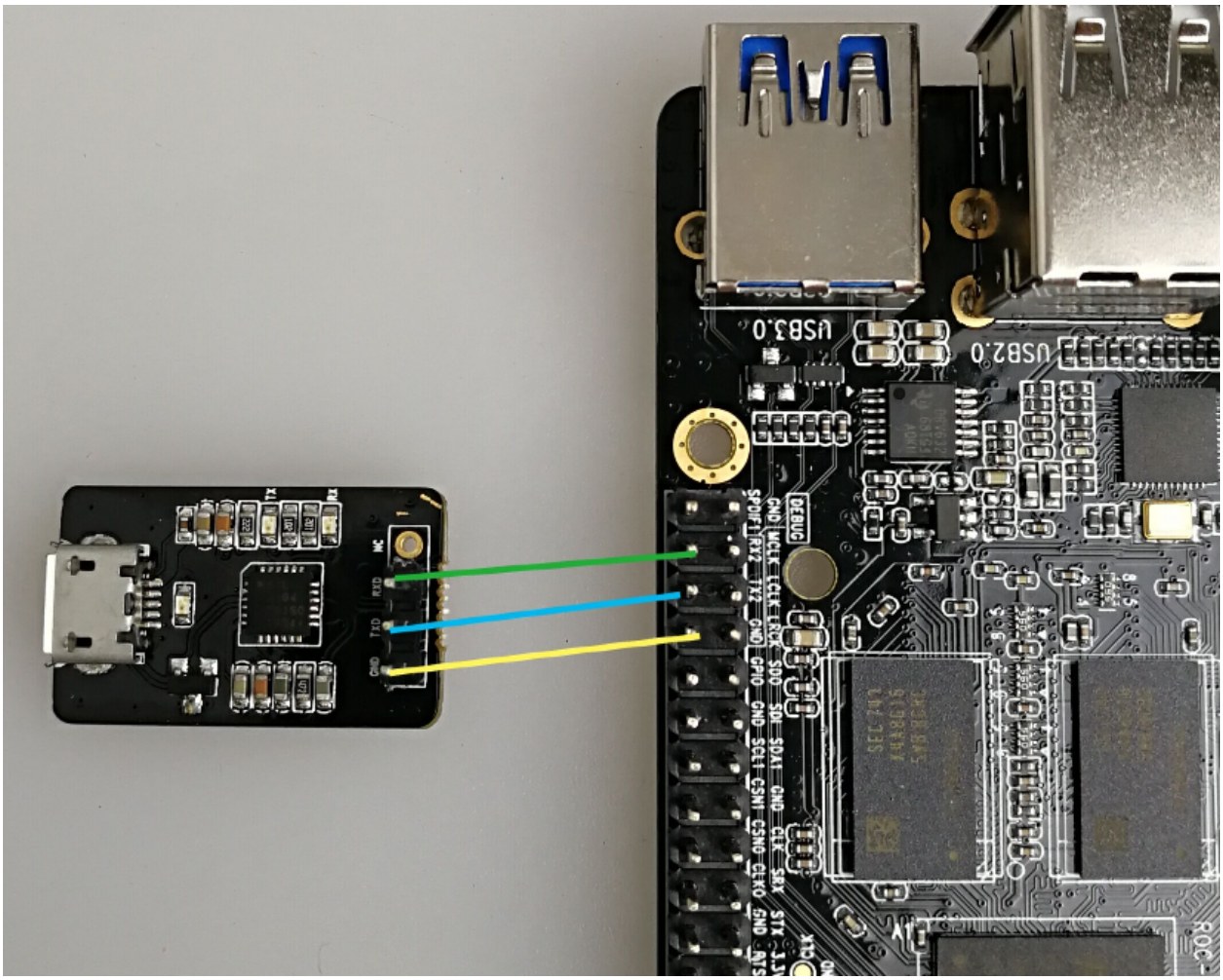
5.1.1 选择 USB 串口适配器

ROC-RK3328-CC 的 UART 调试口默认使用**1,500,000**波特率和 TTL 电平。

一些串口适配器不能支持如此高的波特率。因此在购买之前，请确保它符合要求并有可用的驱动。参考在线商城上带 CP2104 芯片的 USB 转串口适配器。

5.1.2 连接适配器和调试口

用三根线将 TX/RX/GND 引脚分别连接在一起：



如果在使用某些适配器时串口控制台没有输出，可以尝试将适配器的 TX 引脚连接到开发板的 RX 引脚，适配器的 RX 引脚连接到开发板的 TX 引脚，即交换一下 TX、RX 引脚。

5.1.3 串口参数配置

ROC-RK3328-CC 使用如下配置:

- 波特率：1,500,000
- 数据位：8
- 停止位：1
- 奇偶检查：无
- 流控：无

接下来，根据操作系统的不同，为你介绍详细的操作说明。

5.2 Windows 下的串口调试

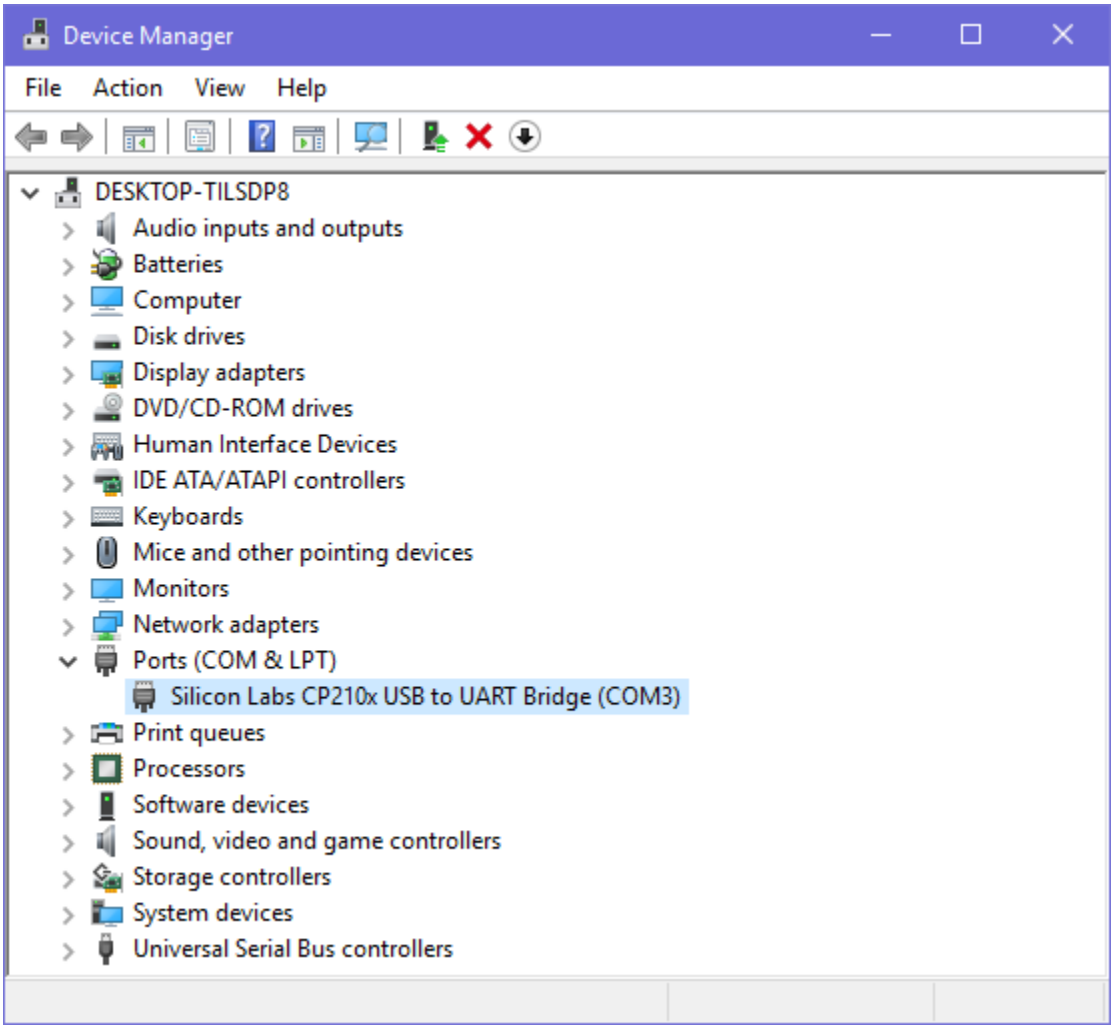
5.2.1 安装驱动

安装卖家推荐的 USB 串口适配器驱动。如果没有，可以检查芯片组并尝试以下驱动

- CH340
- PL2303
- CP210X

  提示：如果 PL2303 在 Win8 下无法工作，则可以尝试将驱动程序降级到版本 3.3.5.122 或之前。

安装驱动后，将适配器连接到主机的 USB 端口。操作系统将提示检测到新硬件。完成后，可以在设备管理器中找到新的COM端口：



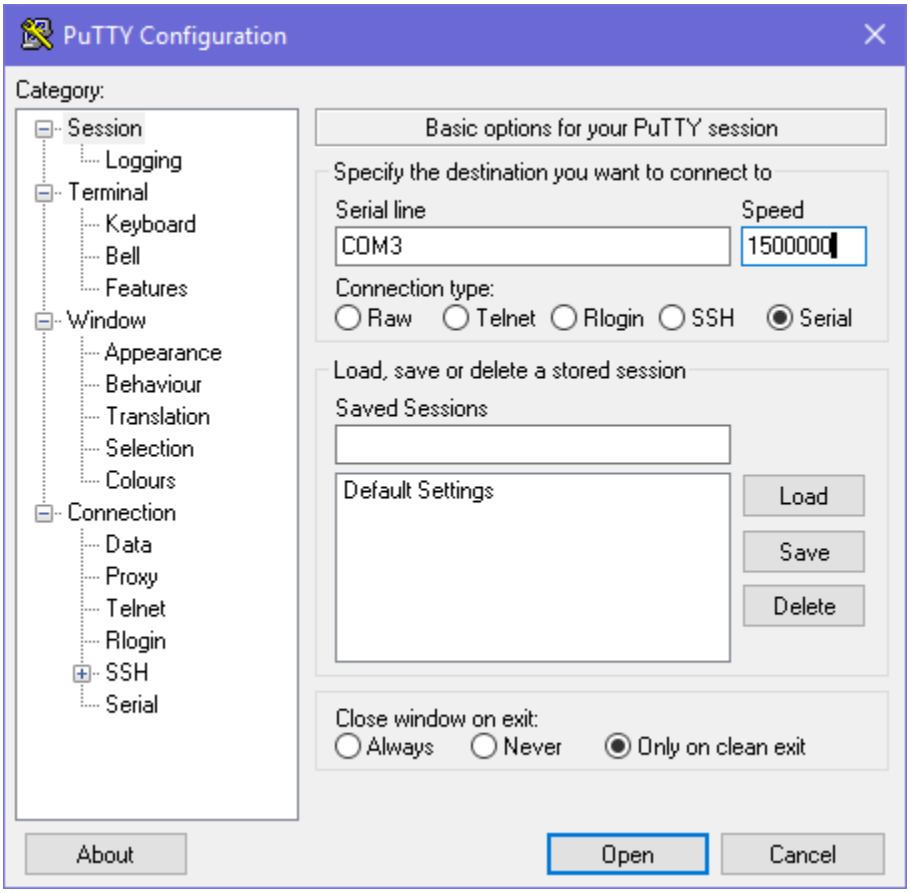
5.2.2 安装工具

Windows 中有很多串口终端工具，例如 putty 和 SecureCRT。以下介绍Putty这款流行的开源软件的使用。

Putty 下载地址

下载 putty.zip，解压并运行 PUTTY.exe：

1. 选择 “Connection type” 为 “Serial”。
2. 将 “Serial line” 修改为在设备管理器中找到的 COM 端口。
3. 设置 “Speed” 为 1500000 。
4. 点击 “Open” 按钮。



5.3 Linux 下的串口调试

如果 USB 串口适配器的芯片组受 Linux 内核支持，驱动程序将自动加载。

连接串口适配器，并通过如下命令检查相应的串口设备文件：

```
$ ls -l /dev/ttyUSB*
crw-rw---- 1 root uucp 188, 0 Apr 10 16:44 /dev/ttyUSB0
```

将你的 Linux 用户添加到 uucp 组中，以便获得访问此设备的权限（否则每次需要在命令前加sudo来运行相应命令）：

```
sudo gpasswd -a $(whoami) uucp
```

用户组的更改将在注销并重新登录 Linux 后生效，或使用 newgrp 命令进入带有新组的 shell:

```
newgrp uucp
```

然后根据偏好，使用自己喜欢的串口控制台工具。以下介绍 picocom 和 minicom 。

5.3.1 picocom

picocom 轻便小巧， 容易使用。

安装 picocom 命令:

```
sudo apt-get install picocom
```

启动 picocom:

```
$ picocom -b 1500000 /dev/ttyUSB0
picocom v3.1

port is        : /dev/ttyUSB0
flowcontrol    : none
baudrate is    : 1500000
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
hangup is      : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-q] [C-h] to see available commands
Terminal ready
```

上面的信息显示 Ctrl-a 是转义键。 按下 Ctrl-a Ctrl-q 将退出 picocom 并返回到 shell 。

5.3.2 minicom

安装 minicom 命令:

sudo apt-get install minicom

启动 minicom:

```
$ minicom
Welcome to minicom 2.7
```

(continues on next page)

(续上页)

```
OPTIONS: I18n
Compiled on Jan  1 2014, 17:13:19.
Port /dev/ttyUSB0, 15:57:00

Press CTRL-A Z for help on special keys
```

根据以上提示: 按 Ctrl-a，然后按 z（而不是 Ctrl-z ）调出帮助菜单:

```
+-----------------------------------------------------------------------+
|                      Minicom Command Summary                          |
|                                                                       |
|               Commands can be called by CTRL-A <key>                  |
|                                                                       |
|                Main Functions                   Other Functions       |
|                                                                       |
| Dialing directory..D  run script (Go)....G | Clear Screen.......C |
| Send files.........S  Receive files......R | cOnfigure Minicom..O |
| comm Parameters....P  Add linefeed.......A | Suspend minicom....J |
| Capture on/off.....L  Hangup.............H | eXit and reset.....X |
| send break.........F  initialize Modem...M | Quit with no reset.Q |
| Terminal settings..T  run Kermit.........K | Cursor key mode....I |
| lineWrap on/off....W  local Echo on/off..E | Help screen........Z |
| Paste file.........Y  Timestamp toggle...N | scroll Back........B |
| Add Carriage Ret...U                                                  |
|                                                                       |
|              Select function or press Enter for none.                 |
+-----------------------------------------------------------------------+
```

按提示按 O 进入设置屏幕:

```
           +-----[configuration]------+
           | Filenames and paths      |
           | File transfer protocols  |
           | Serial port setup        |
           | Modem and dialing        |
           | Screen and keyboard      |
           | Save setup as dfl        |
           | Save setup as..          |
           | Exit                     |
           +--------------------------+
```

选择 Serial port setup，然后按选项前面的大写字母并设置为如下所示的值:

```
   +-----------------------------------------------------------------------+
   | A -    Serial Device      : /dev/ttyUSB0                              |
   | B - Lockfile Location     : /var/lock                                 |
   | C -   Callin Program      :                                           |
   | D -  Callout Program      :                                           |
   | E -    Bps/Par/Bits       : 1500000 8N1                               |
   | F - Hardware Flow Control : No                                        |
   | G - Software Flow Control : No                                        |
   |                                                                       |
   |    Change which setting?                                              |
   +-----------------------------------------------------------------------+
```

注意:

- Hardware Flow Control 和 Software Flow Control 应该设置为 No 。

- 结束设置之后，按 ESC 键回到之前的菜单，选择 Save setup as dfl 保存覆盖掉默认配置。

CHAPTER 6

编译 Linux 固件

这一章将介绍编译 ROC-RK3328-CC Linux 固件的整个流程。

6.1 准备工作

Linux 固件在如下的环境中编译:

- Ubuntu 16.04 amd64

安装以下包:

```
sudo apt-get install bc bison build-essential curl \
     device-tree-compiler dosfstools flex gcc-aarch64-linux-gnu \
     gcc-arm-linux-gnueabihf gdisk git gnupg gperf libc6-dev \
     libncurses5-dev libpython-dev libssl-dev libssl1.0.0 \
     lzop mtools parted repo swig tar zip
```

6.2 下载 Linux SDK

创建工程目录:

```
# create project dir
mkdir ~/proj/roc-rk3328-cc
cd ~/proj/roc-rk3328-cc
```

下载 Linux SDK:

```
# U-Boot
git clone -b roc-rk3328-cc https://github.com/FireflyTeam/u-boot
# Kernel
git clone -b roc-rk3328-cc https://github.com/FireflyTeam/kernel --depth=1
```

(continues on next page)

(续上页)

```
# Build
git clone -b debian https://github.com/FireflyTeam/build
# Rkbin
git clone -b master https://github.com/FireflyTeam/rkbin
```

提示：通过以上链接可以在线浏览源码。

开发板编译配置在:

```
build/board_configs.sh
```

6.3 编译 U-Boot

编译 U-Boot:

```
./build/mk-uboot.sh roc-rk3328-cc
```

输出:

```
out/u-boot/
├── idbloader.img
├── rk3328_loader_ddr786_v1.06.243.bin
├── trust.img
└── uboot.img
```

- `rk3328_loader_ddr786_v1.06.243.bin`: DDR 初始化文件。
- `idbloader.img`: DDR 初始化与 miniloader 结合的文件。
- `trust.img`: ARM trusted 固件。
- `uboot.img`: U-Boot映像文件。

相关文件:

- `configs/roc-rk3328-cc_defconfig`: 默认 U-Boot 配置

6.4 编译 Kernel

编译 kernel:

```
./build/mk-kernel.sh roc-rk3328-cc
```

输出:

```
out/
├── boot.img
└── kernel
    ├── Image
    └── rk3328-roc-cc.dtb
```

- `boot.img`: 包含 `Image` and `rk3328-roc-cc.dtb` 的映像文件, 为 fat32 文件系统格式。
- `Image`: 内核映像。

- `rk3328-roc-cc.dtb`: 设备树 blob。

相关文件:

- `arch/arm64/configs/fireflyrk3328_linux_defconfig`: 默认内核配置。
- `arch/arm64/boot/dts/rockchip/rk3328-roc-cc.dts`: 开发板设备树描述。
- `arch/arm64/boot/dts/rockchip/rk3328.dtsi`: CPU 设备树描述。

自定义内核配置和更新默认配置:

```
# 这非常重要!
export ARCH=arm64

cd kernel

# 首先使用默认配置
make fireflyrk3328_linux_defconfig

# 自定义你的 kernel 配置
make menuconfig

# 保存为默认配置
make savedefconfig
cp defconfig arch/arm64/configs/fireflyrk3328_linux_defconfig
```

注意: 构建脚本不会将内核模块复制到根文件系统。

6.5 编译根文件系统

可以下载预编译的根文件系统，或参考《编译 *Linux* 根文件系统》自己编译一个。

6.6 打包原始固件

把你的 Linux 根文件系统映像文件放在 `out/rootfs.img`

`out` 目录将包含以下文件:

```
$ tree out
out
├── boot.img
├── kernel
│   ├── Image
│   └── rk3328-roc-cc.dtb
├── rootfs.img
└── u-boot
    ├── idbloader.img
    ├── rk3328_loader_ddr786_v1.06.243.bin
    ├── trust.img
    └── uboot.img

2 directories, 8 files
```

打包原始固件:

```
./build/mk-image.sh -c rk3328 -t system -r out/rootfs.img
```

这条命令根据《存储映射》所描述的布局，将分区映像文件写到指定位置，最终打包成 `out/system.img`，

烧写原始固件的步骤，请参考 《上手指南》 一章。

CHAPTER 7

编译 Debian 根文件系统

7.1 准备编译系统

```
git clone https://github.com/FireflyTeam/rk-rootfs-build.git
cd rk-rootfs-build
sudo apt-get install binfmt-support qemu-user-static
sudo dpkg -i ubuntu-build-service/packages/*
sudo apt-get install -f
```

7.2 编译根文件系统

1. 编译基础 Debian 系统：

   ```
   VERSION=stretch TARGET=desktop ARCH=armhf ./mk-base-debian.sh
   ```

   上述命令会调用 `live-build` 去生成一个基本的 Debian Stretch 桌面系统，并打包到文件 `linaro-stretch-alip-*.tar.gz` 中。此操作比较耗时，除非修改了 `live-build` 的配置文件，一般只需要运行一次。

2. 编译 rk-debian 系统：

   ```
   VERSION=stretch TARGET=desktop ARCH=armhf ./mk-rootfs.sh
   ```

   该命令会先解压基础 Debian 系统文件 `linaro-stretch-alip-*.tar.gz`，在此基础上安装和设置 Rockchip 的相关组件，例如 mali、视频解码等，最终生成完整的 rk-debian 系统。

3. 创建 ext4 映像文件 `linaro-rootfs.img`：

   ```
   ./mk-image.sh
   ```

注意：默认的用户和密码均为 ”linaro“。

CHAPTER 8

编译 Ubuntu 根文件系统

环境：

- Ubuntu 16.04 amd64

安装依赖包：

```
sudo apt-get install qemu qemu-user-static binfmt-support debootstrap
```

下载 Ubuntu core：

```
wget -c http://cdimage.ubuntu.com/ubuntu-base/releases/16.04.1/release/ubuntu-base-16.
↪04.1-base-arm64.tar.gz
```

创建一个大小为 1000M 的根文件系统映像文件，并使用 Ubuntu 的基础包去初始化：

```
fallocate -l 1000M rootfs.img
sudo mkfs.ext4 -F -L ROOTFS rootfs.img
mkdir mnt
sudo mount rootfs.img mnt
sudo tar -xzvf ubuntu-base-16.04.1-base-arm64.tar.gz -C mnt/
sudo cp -a /usr/bin/qemu-aarch64-static mnt/usr/bin/
```

`qemu-aarch64-static`是其中的关键，能在 x86_64 主机系统下 chroot 到 arm64 文件系统：

Chroot 到新的文件系统中去并初始化：

```
sudo chroot mnt/

# 这里可以修改设置
USER=firefly
HOST=firefly

# 创建用户
useradd -G sudo -m -s /bin/bash $USER
passwd $USER
```

(continues on next page)

(续上页)

```
# 输入密码

# 设置主机名和以太网
echo $HOST > /etc/hostname
echo "127.0.0.1    localhost.localdomain localhost" > /etc/hosts
echo "127.0.0.1    $HOST" >> /etc/hosts
echo "auto eth0" > /etc/network/interfaces.d/eth0
echo "iface eth0 inet dhcp" >> /etc/network/interfaces.d/eth0
echo "nameserver 127.0.1.1" > /etc/resolv.conf

# 使能串口
ln -s /lib/systemd/system/serial-getty\@.service /etc/systemd/system/getty.target.
↪wants/serial-getty@ttyS0.service

# 安装包
apt-get update
apt-get upgrade
apt-get install ifupdown net-tools network-manager
apt-get install udev sudo ssh
apt-get install vim-tiny
```

卸载文件系统:

```
sudo umount rootfs/
```

Credit: bholland

8.1 参考

- http://opensource.rock-chips.com/wiki_Distribution

CHAPTER 9

LibreELEC 娱乐影音中心

9.1 前言

本文简单介绍了 LibreELEC 的使用流程。

9.2 制作 LibreELEC 启动盘

准备工作:

- ROC-RK3328-CC 开发板
- 下载 LibreELEC 镜像
- 下载并安装 LibreELEC 启动盘制作工具

操作步骤:

1. 运行 LibreELEC 介质启动制作工具:

2. 点击“选择文件”按钮，选择所下载的映像文件并确认。
3. 插入 SD 卡。
4. 在标号 3 处选择对应的 SD 卡设备。
5. 点击“写入”按钮，等待写入完成。

LibreELEC 第一次启动，会有两次重启，请耐心等待。

9.3 编译 LibreELEC 系统

9.3.1 准备工作

以下操作的环境是 Ubuntu 16.04 。

首先安装所需的软件包:

```
sudo apt install gcc make git unzip wget xz-utils
```

然后设置 bash 为系统默认的 shell:

```
sudo dpkg-reconfigure dash
# 选择 No
```

系统原来默认的 dash 会产生脚本的兼容性问题。

详细环境搭建，可参照官网 compile。

9.3.2 源码获取和编译

从 github 上获取适配过 ROC-RK3328-CC 的 LibreELEC 源码:

```
git clone https://github.com/T-Firefly/LibreELEC.tv.git
```

编译:

```
cd LibreELEC.tv/
PROJECT=Rockchip DEVICE=RK3328 ARCH=arm UBOOT_SYSTEM=roc-cc make image
```

第一次构建需要等待较长的时间。编译完成后，在当前 target 目录下，将生成对应文件:

- `.img.gz` 可写入介质创建一个新的安装文件
- `.tar` 用于更新现有的安装文件

9.4 使用提示

- SSH 登录，用户为 “root“，密码为”libreelec“。
- LibreELEC 升级系统的方式有多种，参考 update_libreelec 。
- 外部可通过 SSH/Samba 访问到 ROC-RK3328-CC 开发板，参考 accessing_libreelec 。
- 如果切换中文？先设置皮肤字体(Skin –> Fonts –> 选择 ‘Arial based’)，再设置语言（Regional –> Language –> 选择 ‘Chinese(Simple)’）。

CHAPTER 10

编译 Android 7.1

10.1 准备

10.1.1 硬件配置

编译 Android 7.1 开发环境硬件配置建议：

- 64 位 CPU
- 16GB 内存 + 交换内存
- 30GB 空闲空间用来编译， 源码树另占 8GB

另外可参考 Google 官方文档硬件和软件配置：

- https://source.android.com/setup/build/requirements
- https://source.android.com/setup/initializing

10.1.2 软件配置

安装 JDK 8

```
sudo add-apt-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install openjdk-8-jdk
```

安装环境包

```
sudo apt-get install git-core gnupg flex bison gperf libsdl1.2-dev \
  libesd0-dev libwxgtk2.8-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
```

(continues on next page)

(续上页)

```
sudo apt-get install gcc-arm-linux-gnueabihf \
  libssl1.0.0 libssl-dev \
  p7zip-full
```

10.2 下载 Android SDK

由于 SDK 较大，请选择以下云盘之一下载 ROC-RK3328-CC_Android7.1.2_git_20171204.7z:

- Baiduyun
- Google Drive

下载完成后，在解压前先校验下 MD5 码:

```
$ md5sum /path/to/ROC-RK3328-CC_Android7.1.2_git_20171204.7z
6d34e51fd7d26e9e141e91b0c564cd1f ROC-RK3328-CC_Android7.1.2_git_20171204.7z
```

然后解压:

```
mkdir -p ~/proj/roc-rk3328-cc
cd ~/proj/roc-rk3328-cc
7z x /path/to/ROC-RK3328-CC_Android7.1.2_git_20171204.7z
git reset --hard
```

更新远程仓库:

```
git remote rm origin
git remote add gitlab  https://gitlab.com/TeeFirefly/RK3328-Nougat.git
```

从 gitlab 处同步源码:

```
git pull gitlab roc-rk3328-cc:roc-rk3328-cc
```

也可以到如下地址查看源码： https://gitlab.com/TeeFirefly/RK3328-Nougat/tree/roc-rk3328-cc

10.3 使用 Firefly 脚本编译

编译内核

```
./FFTools/make.sh -k -j8
```

编译 U-Boot

```
./FFTools/make.sh -u -j8
```

编译 Android

```
./FFTools/make.sh -a -j8
```

编译全部

如下指令会编译出内核, U-Boot 以及 Android:

```
./FFTools/make.sh -j8
```

10.4 不使用脚本编译

编译之前请先执行如下命令配置好环境变量:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

编译内核

```
make ARCH=arm64 firefly_defconfig
make -j8 ARCH=arm64 rk3328-roc-cc.img
```

编译 U-Boot

```
make rk3328_box_defconfig
make ARCHV=aarch64 -j8
```

编译 Android

```
source build/envsetup.sh
lunch roc_rk3328_cc_box-userdebug
make installclean
make -j8
./mkimage.sh
```

10.5 打包 RK 固件

在 Linux 下打包固件

编译完成后使用 Firefly 官方脚本即可打包所有的分区映像成 RK固件或 原始固件:

```
rk固件:
./FFTools/mkupdate/mkupdate.sh update


原始固件:
./FFTools/mkupdate/sd_mkupdate.sh update
```

最终生成的文件是 `rockdev/Image-rk3328_firefly_box/update.img`.

RK固件需使用 `SD_Firmware_Tool` 工具，功能模式选择SD启动，来制作启动卡， 而原始固件可使用 `SDCard-installer` 制作启动卡

在 Windows 下打包固件

在 Windows 下打包 RK 固件 `update.img` 也是很简单的:

1. 拷贝所有在 `rockdev/Image-rk3328_firefly_box/` 目录下编译好的文件到 AndroidTool 的 `rockdev\Image` 目录下。
2. 运行在 AndroidTool 的 `rockdev` 目录下的 `mkupdate.bat` 文件。

3. 在 rockdev\Image 目录将会生成 update.img。

10.6 分区映像

update.img 是发布给最终用户的固件，方便升级开发板。而在实际开发中，更多的时候是修改并烧写单个分区映像文件，这样做大大节省开发时间。

下表总结了在各个编译阶段所生成的分区映像文件：

Stage	Product	Partition
Compiling Kernel	kernel/kernel.img	kernel
	kernel/resource.img	resource
Compiling U-Boot	u-boot/uboot.img	uboot
./mkimage.sh	boot.img	boot
	system.img	system

注意，执行 ./mkimage.sh 后， boot.img 和 system.img 将会被重新编译并打包到目录 out/target/product/rk3328_firefly_box/ 下，所有生成的映像文件将会拷贝到目录 rockdev/Image-rk3328_firefly_box/ 下。

如下是映像文件列表：

- boot.img： Android 的 initramfs 映像，包含Android根目录的基础文件系统，它负责初始化和加载系统分区。
- system.img： ext4 文件系统格式的 Android 文件系统分区映像。
- kernel.img： 内核映像。
- resource.img： Resource 映像， 包含启动图片和内核设备树。
- misc.img： misc 分区映像， 负责启动模式的切换和急救模式参数的传递。
- recovery.img： Recovery 模式映像。
- rk3328_loader_v1.08.244.bin： Loader 文件。
- uboot.img： U-Boot 映像文件。
- trust.img： Arm trusted file (ATF) 映像文件。
- parameter.txt： 分区布局和内核命令行。

CHAPTER 11

解包/打包 RK 固件

11.1 RK 固件格式

RK 固件 `release_update.img` 包含引导加载程序 `loader.img` 和实际的固件数据`update.img`:

release_update.img

```
|- loader.img
`- update.img
```

`update.img` 是打包工具读取 `package-file` 索引文件从而将多个映像文件打包而成固件 。一个典型的 `package-file` 文件的内容为:

```
# NAME Relative path
package-file    package-file
bootloader      Image/MiniLoaderAll.bin
parameter       Image/parameter.txt
trust           Image/trust.img
uboot           Image/uboot.img
misc            Image/misc.img
resource        Image/resource.img
kernel          Image/kernel.img
boot            Image/boot.img
recovery        Image/recovery.img
system          Image/system.img
backup          RESERVED
#update-script  update-script
#recover-script recover-script
```

- `package-file`: `update.img` 的索引文件，也包含在 `update.img` 中。
- `Image/MiniLoaderAll.bin`: 通过 CPU ROM 代码加载的第一个 bootloader。
- `Image/parameter.txt`: 参数文件，可以在其中设置内核启动参数和分区布局。
- `Image/trust.img`: Arm trust file (ATF) 映像，用于安全启动。

- Image/misc.img：misc 分区映像，用于控制 Android 的启动模式。
- Image/kernel.img：Android 内核映像。
- Image/resource.img：具有启动图片和内核设备树的资源映像。
- Image/boot.img：Android initramfs，一个在正常启动时加载的根文件系统，包含重要的初始化和服务描述。
- Image/recovery.img：Recovery 模式映像。
- Image/system.img：Android 系统分区映像。

解包是从 release_update.img 中提取 update.img，然后解开里面的所有映像文件。 重新打包时，则是相反的过程。它将由 package-file 描述的所有映像文件合成到 update.img 中，该文件将与 bootloader 一起打包以创建最终的 release_update.img。

11.2 安装工具

```
git clone https://github.com/TeeFirefly/rk2918_tools.git
cd rk2918_tools
make
sudo cp afptool img_unpack img_maker mkkrnlimg /usr/local/bin
```

11.3 解包 RK 固件

- 解包 release_update.img:

```
$ cd /path/to/your/firmware/dir
$ img_unpack Firefly-RK3399_20161027.img img
rom version: 6.0.1
build time: 2016-10-27 14:58:18
chip: 33333043
checking md5sum....OK
```

- 解包 update.img:

```
$ cd img
$ afptool -unpack update.img update
Check file...OK
------- UNPACK -------
package-file              0x00000800    0x00000280
Image/MiniLoaderAll.bin   0x00001000    0x0003E94E
Image/parameter.txt       0x00040000    0x00000350
Image/trust.img           0x00040800    0x00400000
Image/uboot.img           0x00440800    0x00400000
Image/misc.img            0x00840800    0x0000C000
Image/resource.img        0x0084C800    0x0003FE00
Image/kernel.img          0x0088C800    0x00F5D00C
Image/boot.img            0x017EA000    0x0014AD24
Image/recovery.img        0x01935000    0x013C0000
Image/system.img          0x02CF5000    0x2622A000
RESERVED                  0x00000000    0x00000000
UnPack OK!
```

- 在 update 目录检查目录树:

```
$ cd update/
$ tree
.
├── Image
│   ├── boot.img
│   ├── kernel.img
│   ├── MiniLoaderAll.bin
│   ├── misc.img
│   ├── parameter.txt
│   ├── recovery.img
│   ├── resource.img
│   ├── system.img
│   ├── trust.img
│   └── uboot.img
├── package-file
└── RESERVED

1 directory, 12 files
```

11.4 打包 RK 固件

首先, 确保在 parameter.txt 文件中的 system 分区足以容纳 system.img. 参考 《Parameter 文件格式》 了解分区布局。

例如，在 parameter.txt 的前缀为 “CMDLINE” 的行中，可以找到类似于以下内容的 system 分区的描述:

```
0x00200000@0x000B0000(system)
```

“@”之前的十六进制字符串是分区的大小（以扇区为单位，此处为 1 扇区= 512 字节），因此系统分区的大小为:

```
$ echo $(( 0x00200000 * 512 / 1024 / 1024 ))M
1024M
```

创建 release_update_new.img:

```
# 当前目录仍然是 update/，package-file 文件
# 以及所描述的所有映像文件均在此目录下。
# 将参数文件复制为 "parameter" 文件，afptool 默认使用此名。

$ afptool -pack . ../update_new.img
------ PACKAGE ------
Add file: ./package-file
Add file: ./Image/MiniLoaderAll.bin
Add file: ./Image/parameter.txt
Add file: ./Image/trust.img
Add file: ./Image/uboot.img
Add file: ./Image/misc.img
Add file: ./Image/resource.img
Add file: ./Image/kernel.img
Add file: ./Image/boot.img
Add file: ./Image/recovery.img
```

(continues on next page)

(续上页)

```
Add file: ./Image/system.img
Add file: ./RESERVED
Add CRC...
------ OK ------
Pack OK!

$ img_maker -rk33 loader.img update_new.img release_update_new.img
generate image...
append md5sum...
success!
```

11.5 自定义

11.5.1 自定义 system.img

`system.img` 是 ext4 文件系统格式的映像文件，可以直接挂载到系统进行修改:

```
sudo mkdir -p /mnt/system
sudo mount Image/system.img /mnt/system
cd /mnt/system
# Modify the contents of the inside.
# Pay attention to the free space,
# You can not add too many APKs

# 结束时，需要卸载
cd /
sudo umount /mnt/system
```

请注意，`system.img` 的可用空间几乎为 0，如果需要扩展映像文件，请相应地调整 `parameter.txt` 中的分区布局。

以下是如何将映像文件的大小增加 128MB 的示例。

扩展之前先确保 `system.img` 没有被系统挂载上:

```
mount | grep system
```

改变映像文件及其内在文件系统的大小:

```
dd if=/dev/zero bs=1M count=128 >> Image/system.img
# Expand file system information
e2fsck -f Image/system.img
resize2fs Image/system.img
```

CHAPTER 12

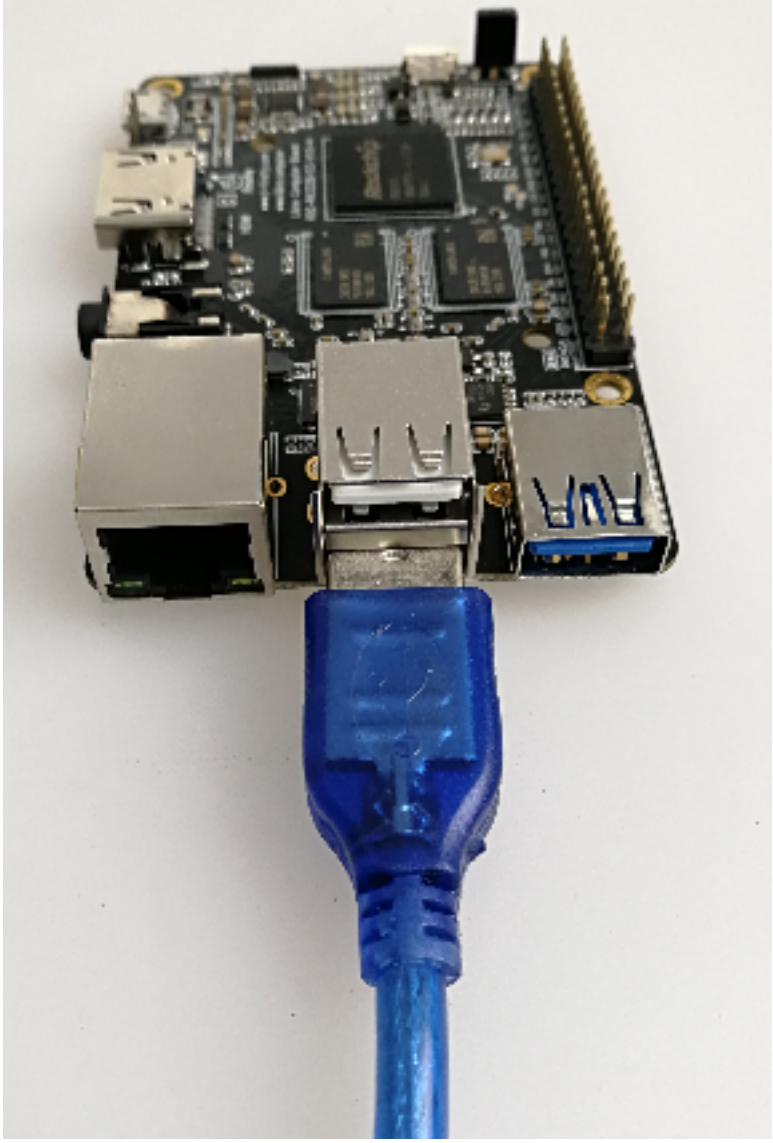
Adb 介绍

Adb 是 Android Debug Bridge 的简称，是 Android 的命令行调试工具，可以完成多种功能，如跟踪系统日志、上传下载文件、安装应用等。

12.1 准备工作

使用 adb时，你需要：

1. 使用公对公 USB 线连接电脑和板子的 USB OTG 口：



2. 在跑 Android 的开发板上，选择 `Settings` -> `USB`，然后勾选 `Connect to PC` 选项。
3. 基于你的系统安装 adb 驱动和命令。

12.1.1 Adb 在 Windows 下的安装

1. 安装 Rockusb 驱动。
2. 下载 adb.zip，然后解压到 `C:\adb`。

打开 `cmd` 窗口然后运行:

```
C:\adb\adb shell
```

若成功就会进入 adb shell 。

12.1.2 Adb 在 Ubuntu 下的安装

1. 安装 adb 工具:

```
sudo apt-get install android-tools-adb
```

2. 添加设备 ID:

```
mkdir -p ~/.android
vi ~/.android/adb_usb.ini
# add the following line:
0x2207
```

3. 为非 root 用户添加 udev 规则:

```
sudo vi /etc/udev/rules.d/51-android.rules
# add the following line:
SUBSYSTEM=="usb", ATTR{idVendor}=="2207", MODE="0666"
```

4. 重载 udev 规则:

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

5. 普通用户下重启 adb:

```
sudo adb kill-server
adb start-server
```

然后就可以直接使用 adb 了, 如:

```
adb shell
```

12.2 常用 Adb 命令

12.2.1 连接管理

列出所有连接设备以及它们的序列号:

```
adb devices
```

若没有多连接设备，就必须用序列号来区分:

```
export ANDROID_SERIAL=<device serial number>
adb shell ls
```

也可以用 TCP/IP 网络连接 Adb :

```
adb tcpip 5555
```

Adb 会在设备上重启并监听 5555 TCP 端口， 这个时候就可以拔出 USB 线了。

如果设备的 IP 地址为 192.168.1.100，执行以下命令连接:

```
adb connect 192.168.1.100:5555
```

一旦连接，就可以执行 adb 命令了：

```
adb shell ps
adb logcat
```

直到断开 adb 连接：

```
adb disconnect 192.168.1.100:5555
```

12.2.2 调试

查询系统日志

用法:

```
adb logcat [option] [Application label]
```

示例:

```
# 查看所有日志
adb logcat

# 仅查看部分日志
adb logcat -s WifiStateMachine StateMachine
```

收集 Bug 报告

`adb bugreport` 用来收集错误报告和一些系统信息。

```
adb bugreport

# 保存到本地，易于编辑和查看
adb bugreport >bugreport.txt
```

12.2.3 运行 shell

打开一个交互的 shell:

```
adb shell
```

执行 shell 命令:

```
adb shell ps
```

12.2.4 Apk 管理

安装 Apk

```
adb install [option] example.apk

选项：
-l 转发锁定
-r 重新安装应用程序以保留原始数据
-s 安装到SD卡而不是内部存储
```

示例:

```
# 安装 facebook.apk
adb install facebook.apk

# 升级 twitter.apk
adb install -r twitter.apk
```

若安装失败，检查下常见原因:

- INSTALL_FAILED_ALREADY_EXISTS: 尝试添加 -r 参数再次安装。
- INSTALL_FAILED_SIGNATURE_ERROR: APK 签名不一致，这可能是由于签名和调试版本的不同导致的。如果确认APK文件签名是正常的，可以使用 adb uninstall 命令卸载旧的应用程序，然后重新安装。
- INSTALL_FAILED_INSUFFICIENT_STORAGE: 存储空间不够。

卸载 Apk

```
adb uninstall apk_name
```

示例:

```
adb uninstall com.android.chrome
```

apk 包的名称可以用下面的命令列出:

```
adb shell pm list packages -f
...
package:/system/app/Bluetooth.apk=com.android.bluetooth
...
```

Apk文件路径和软件包名称用 = 分隔。

CHAPTER 13

FAQ

13.1 如何写入 MAC address?

可以自己改变 ROC-RK3328-CC MAC 地址，进入 RockUSB 模式，然后使用 WNpctool 工具写入 MAC 地址。

13.2 接入耳机没有声音

Ubuntu 系统下，在多媒体菜单下运行 PluseAudio Volume Control，在配置项中，选择正在工作中的声卡并屏蔽掉其他。

CHAPTER 14

固件和工具

- 固件下载页面

固件烧写工具：

- 烧写 SD 卡
  - 图形界面烧写工具：
    - SDCard Installer (Linux/Windows/Mac)
    - Etcher (Linux/Windows/Mac)
  - 命令行烧写工具
    - dd (Linux)
- 烧写 eMMC
  - 图形界面烧写工具：
    - AndroidTool (Windows)
  - 命令行烧写工具：
    - upgrade_tool (Linux)
    - rkdeveloptool (Linux)

CHAPTER 15

文档和参考

- Partition and Storage Map

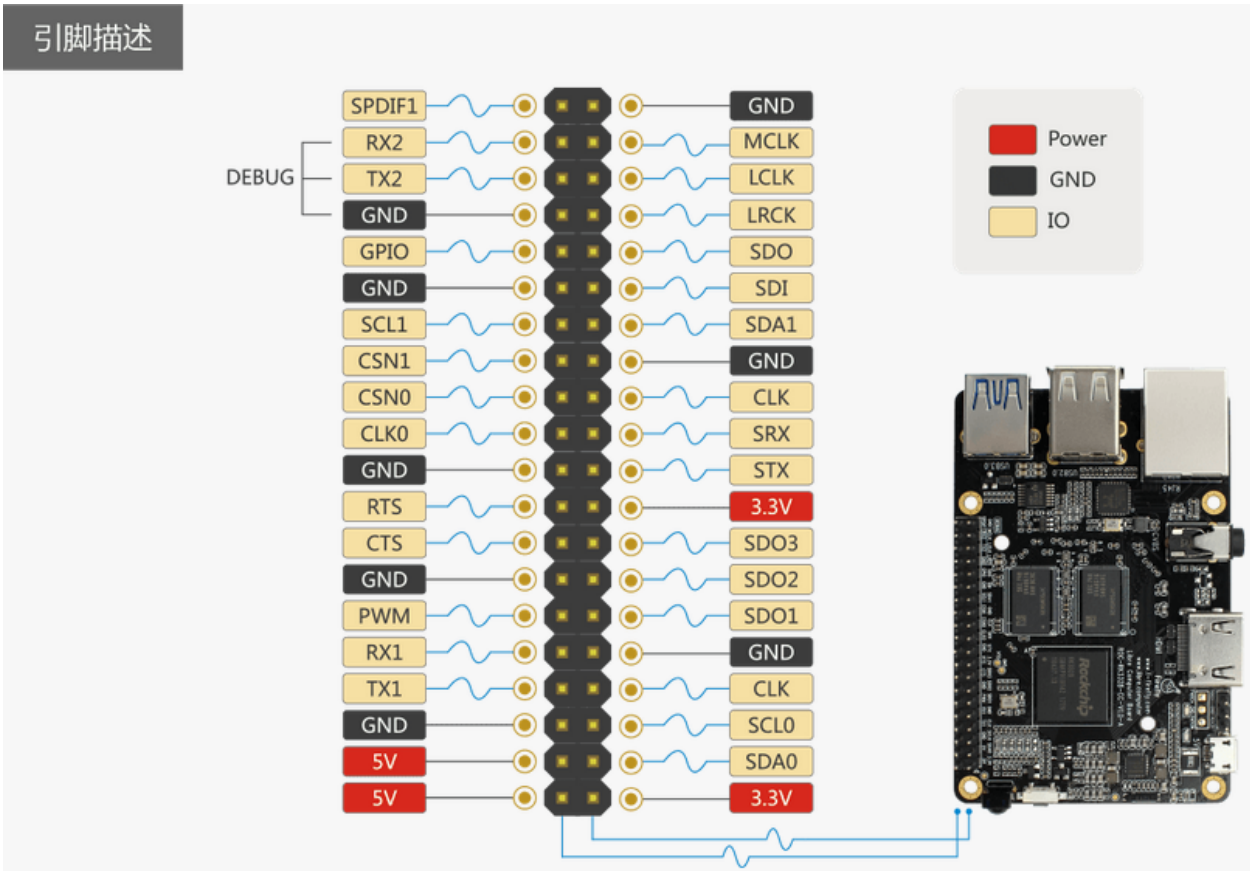
CHAPTER 16

硬件规格书和接口

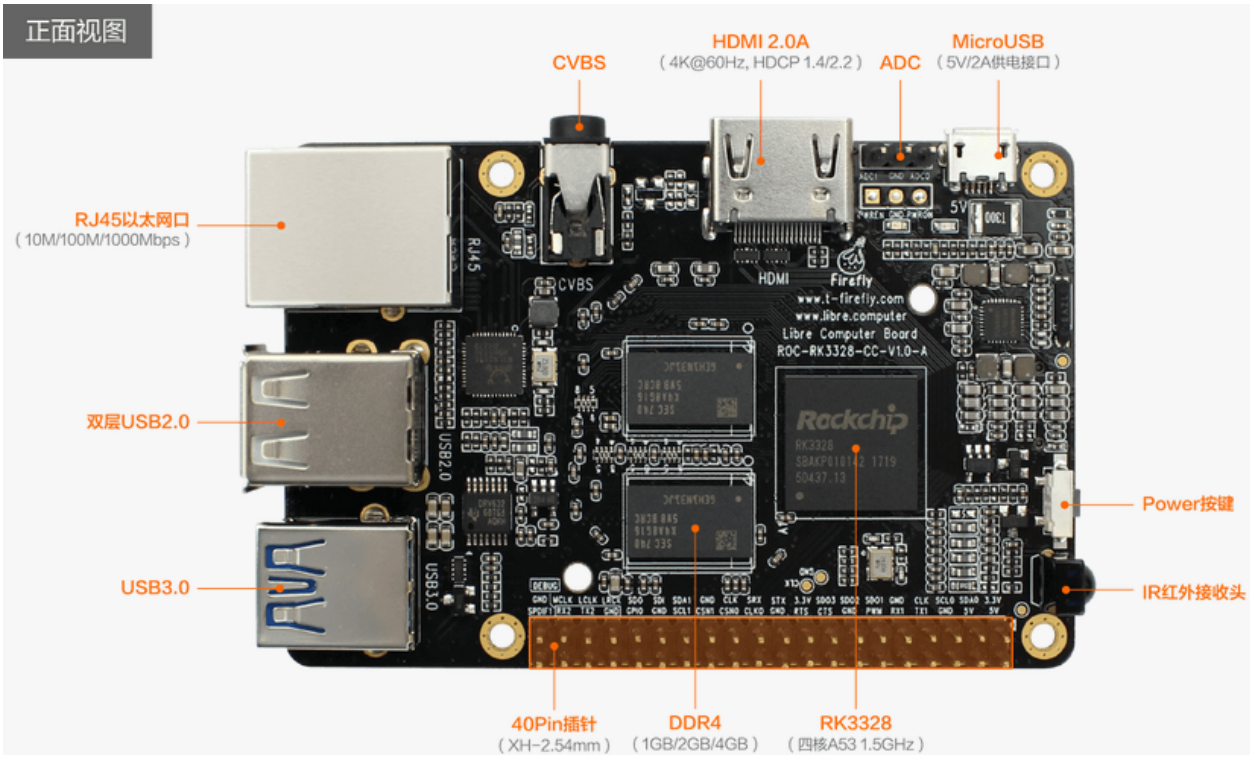
规格书：

- ROC-RK3328-CC 原理图
- ROC-RK3328-CC 贴片图
- Rockchip RK3328 规格书
- Rockchip RK805 规格书 V1.1

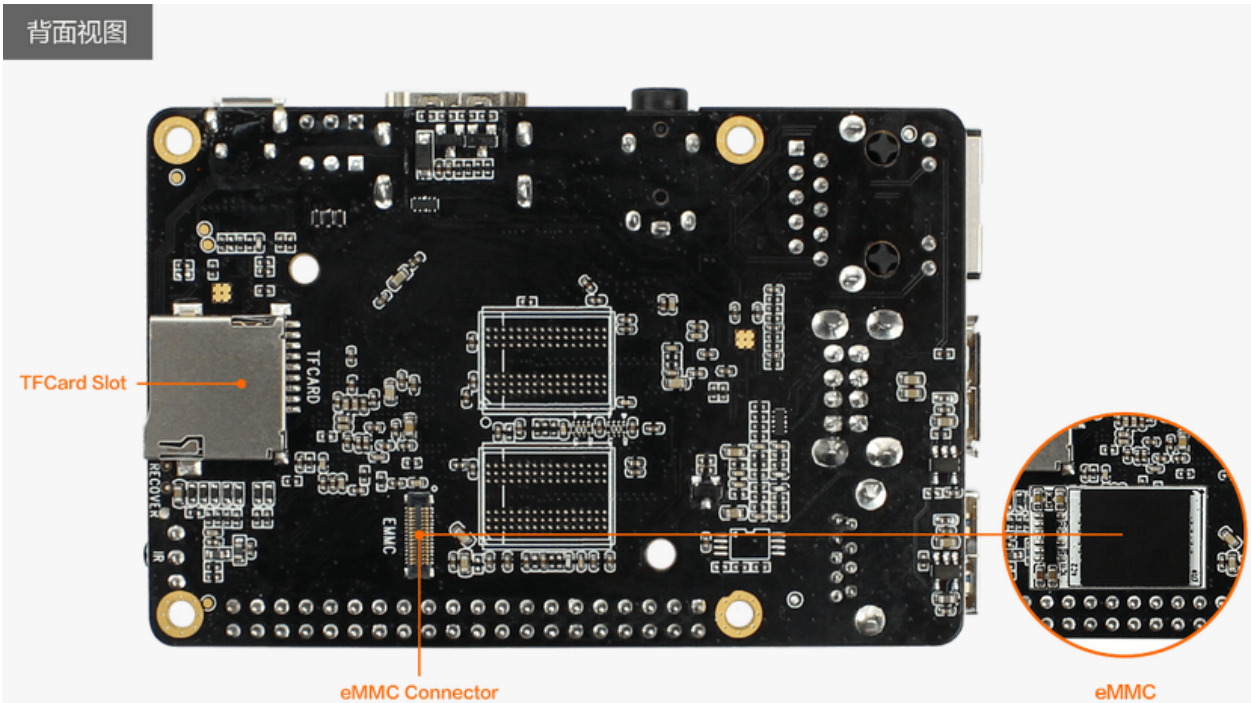
扩展口：


引脚描述
DEBUG
SPDIF1
RX2
TX2
GND
GPIO
GND
SCL1
CSN1
CSN0
CLK0
GND
RTS
CTS
GND
PWM
RX1
TX1
GND
5V
5V
GND
MCLK
LCLK
LRCK
SDO
SDI
SDA1
GND
CLK
SRX
STX
3.3V
SDO3
SDO2
SDO1
GND
CLK
SCL0
SDA0
3.3V
Power
GND
IO

开发板正面：


正面视图
CVBS
HDMI 2.0A
（4K@60Hz, HDCP 1.4/2.2）
ADC
MicroUSB
（5V/2A供电接口）
RJ45以太网口
（10M/100M/1000Mbps）
双层USB2.0
USB3.0
Power按键
IR红外接收头
40Pin插针
（XH-2.54mm）
DDR4
（1GB/2GB/4GB）
RK3328
（四核A53 1.5GHz）

开发板背面:



CHAPTER 17

社区

Firefly：

- 论坛
- 脸书
- Google+
- 油管
- 推特
- 在线商城