ROC-RK3328-CC

Release 1.0.0

Firefly Team

Nov 22, 2018

Getting Started

CHAPTER 1

Introduction

1.1 Welcome

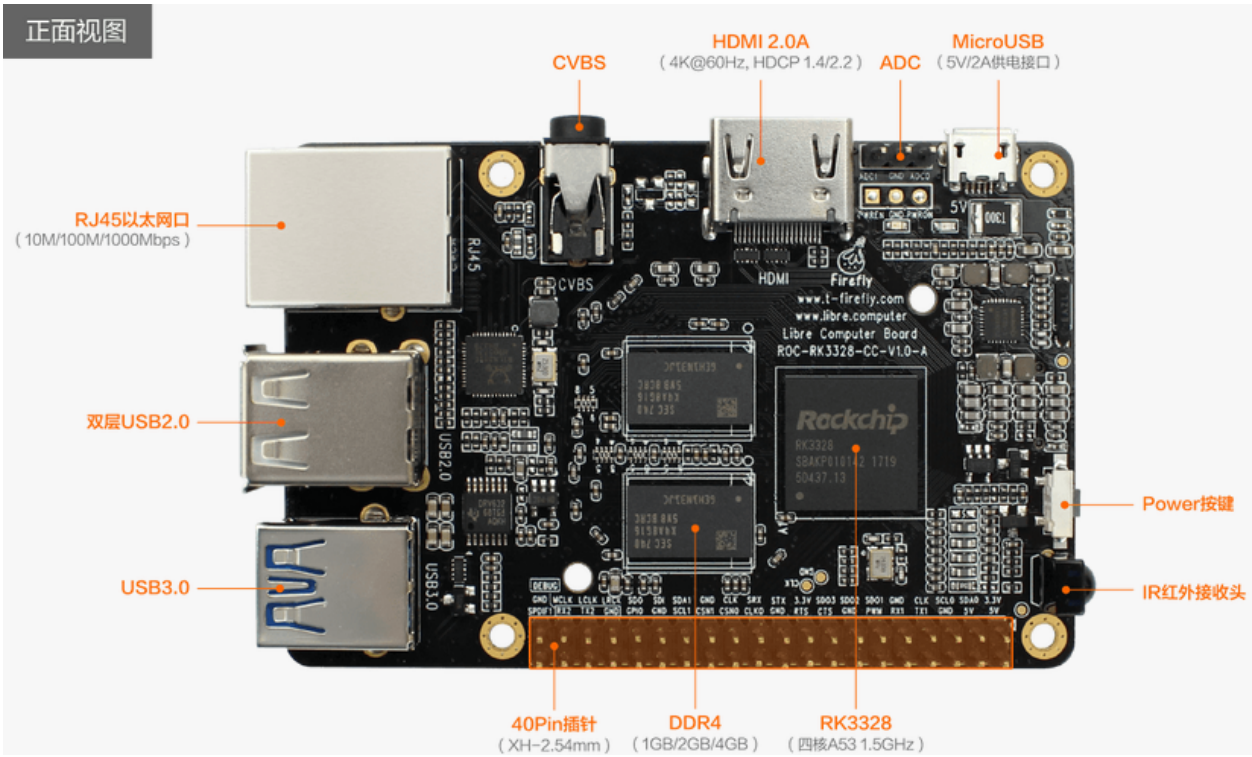
If you're new to ROC-RK3328-CC, the *Getting Started* section provides a guide for everything you need to flash the firmware and get the board running.

If you need help and have read through *Getting Started*, check out *FAQ*.

If you still can't find what you need here, read *Serial Debug* section, get the log, contact us, and help improve this documentation.

1.2 Specification

ROC-RK3328-CC, the first credit card sized and affordable open source main board honored by Firefly, features:



- Core
  - Quad-Core ARM® Cortex-A53 64-bit processor, with frequency up to 1.5GHz
  - ARM Mali-450 MP2 Quad-Core GPU, supports OpenGL ES1.1/2.0, OpenVG1.1
  - DDR4 RAM (1GB/2GB/4GB)
- Connectivity
  - 2 x USB 2.0 Host, 1 x USB 3.0 Host
  - 10/100/1000Mb Ethernet
  - 1 x IR Receiver Module, supports self-defined IR remote
  - 40 pin expansion interface, contains GPIO, I2S, SPI, I2C, UART, PWM SPDIF, DEBUG
- Display
  - HDMI 2.0 (Type-A), supports maximum 4K@60Hz display
  - TV out, CVBS display, in accordance with 480i, 576i standard
- Audio
  - I2S, supports 8 channels
  - SPDIF, for audio output
- Video
  - 4K VP9 and 4K 10bits H265 / H264 video decoding, up to 60fps
  - 1080P multi-format video decoding(WMV, MPEG-1/2/4, VP9, H.264, H.265)
  - 1080P video coding, supports H.264/H.265
  - Video postprocessor: de-interlacing, denoising, edge / detail / color optimization

- Storage
  - High-Speed eMMC extension interface
  - MicroSD (TF) Card Slot

(Full Specification)

This incredible ultra small board runs Android 7.1 or Ubuntu 16.04 smoothly and quietly, thanks to its low power consumption.

1.3 Package & Accessories

ROC-RK3328-CC standard kit contains the following items:

- ROC-RK3328-CC main board
- Micro USB cable

The following accessories are highly recommended, especially when you are doing developer's work:

- 5V2A US Adapter, a good power source is a must have.
- USB Serial Adapter, for serial console debugging.
- eMMC Flash, provides more system performance and reliability.

CHAPTER 2

Getting Started

ROC-RK3328-CC supports booting from the following storage devices:

- SD card
- eMMC

You need to flash the firmware file to the SD card or eMMC, in order to make the board bootable.

SDCard Installer is the officially recommended SD card flashing tool, which is derived from Etcher / Rock64 Installer. It implements one-stop downloading and flashing of firmware file, which makes life easy.

2.1 Firmware Format

There are two firmware file formats:

- Raw Firmware
- RK Firmware

Raw Firmware, when flashing, is copied to the storage device bit by bit. It is the raw image of the storage device. Raw Firmware is flashed to the SD card in common cases, but it can also be flashed to the eMMC. There are lots of flashing tools available:

- To flash to the SD card:
  - GUI:
    - SDCard Installer (Linux/Windows/Mac)
    - Etcher (Linux/Windows/Mac)
  - CLI:
    - dd (Linux)
- To flash to the eMMC:
  - GUI:

        * AndroidTool (Windows)
    - CLI:
        * upgrade_tool (Linux)
        * rkdeveloptool (Linux)

RK Firmware, is packed in Rockchip's proprietary format, which is flashed to the eMMC via Rockchip's upgrade_tool (Linux) or AndroidTool (Windows). It is Rockchip's traditional packing format, commonly used in Rockchip Android firmware. RK Firmware of Android can also be flashed into SD card using SD Firmware Tool.

Since Raw Firmware has wider audience, for simplicity, we **DO NOT PROVIDE** RK Firmware to download any more.

Partition Image, is the raw image of the partition. When you build the Android SDK, you'll get a list of `boot.img`, `kernel.img`, `system.img`, etc, which is called Partition Image and will be flashed to the corresponding partition. For example, `kernel.img` is to be flashed to `kernel` partition in the eMMC or SD card.

2.2 Download & Flash

We recommend to use SDCard Installer to flash Raw Firmware to SD card.

If you are using tools other than SDCard Installer, please download the Raw Firmware in the Download Page first.

Here's the available OS list of firmware:

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

WARNING: Only Raw Firmwares are available in the download page. We **DO NOT PROVIDE** RK Firmware any more.

Then choose the flashing tool according to your host PC's OS:

- To flash to the SD card:
    - GUI:
        * SDCard Installer (Linux/Windows/Mac)
        * Etcher (Linux/Windows/Mac)
    - CLI:
        * dd (Linux)
- To flash to the eMMC:
    - GUI:
        * AndroidTool (Windows)
    - CLI:
        * upgrade_tool (Linux)
        * rkdeveloptool (Linux)

2.3 System Boot Up

Before system boots up, make sure you have:

- A bootable SD card or eMMC
- 5V2A power adapter
- Micro USB cable

Then follow the procedures below:

1. Pull the power adapter out of the power socket.
2. Use the micro USB cable to connect the power adapter and the board.
3. Plug in the bootable SD card or eMMC (NOT BOTH).
4. Plug in the optional HDMI cable, USB mouse or keyboard.
5. Check everything is okay, then plug the power adapter into the power socket to power on the board.

CHAPTER 3

Flashing to the SD Card

We will introduce how to flash the firmware to the SD card. If not explicitly stated, the following firmware is referred to the Raw Firmware. Read about firmware format if of any doubt.

We recommend to use SDCard Installer to flash the Raw Firmware to the SD card.

If you are using tools other than SDCard Installer, please download the Raw Firmware in the Download Page first.

Here's the available OS list of firmware:

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

WARNING: Only Raw Firmwares are available in the download page. We **DO NOT PROVIDE** RK Firmware any more.

3.1 Preparing the SD Card

Please read this good article about how to prepare a SD card first, to make sure that you have a **good, reliable and fast** SD card, which is of essential importance for system stability.

3.2 Flashing Tools

Please choose the flashing tool according to your host PC OS:

- To flash to the SD card
  - GUI:
    - SDCard Installer (Linux/Windows/Mac)

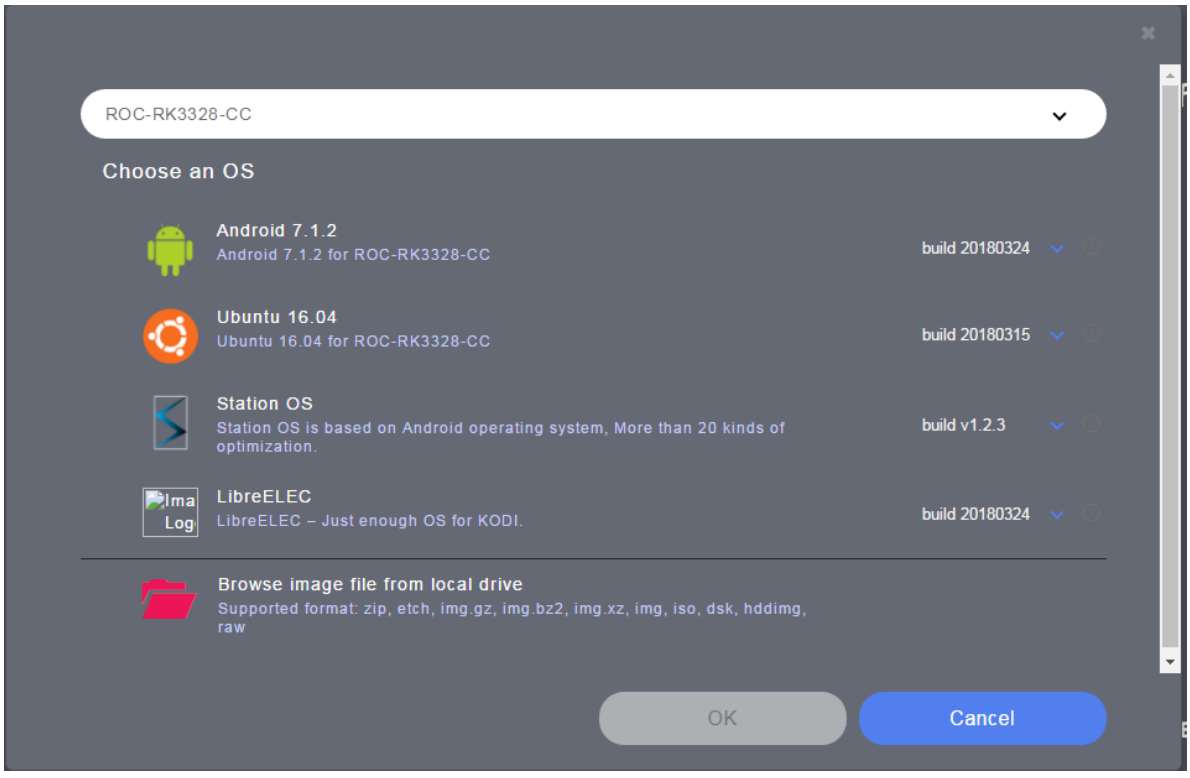
    * Etcher (Linux/Windows/Mac)
  - CLI:
    * dd (Linux)

3.3 SDCard Installer

The easiest way to flash the Raw Firmware is to use the official SDCard Installer, a handy firmware flashing tool derived from Etcher / Rock64 Installer. It saves time to search for available firmwares for your device. You just need to select the board, choose firmware, plugin in the SD card, and finally click the flash button, which is simple and straightforward.

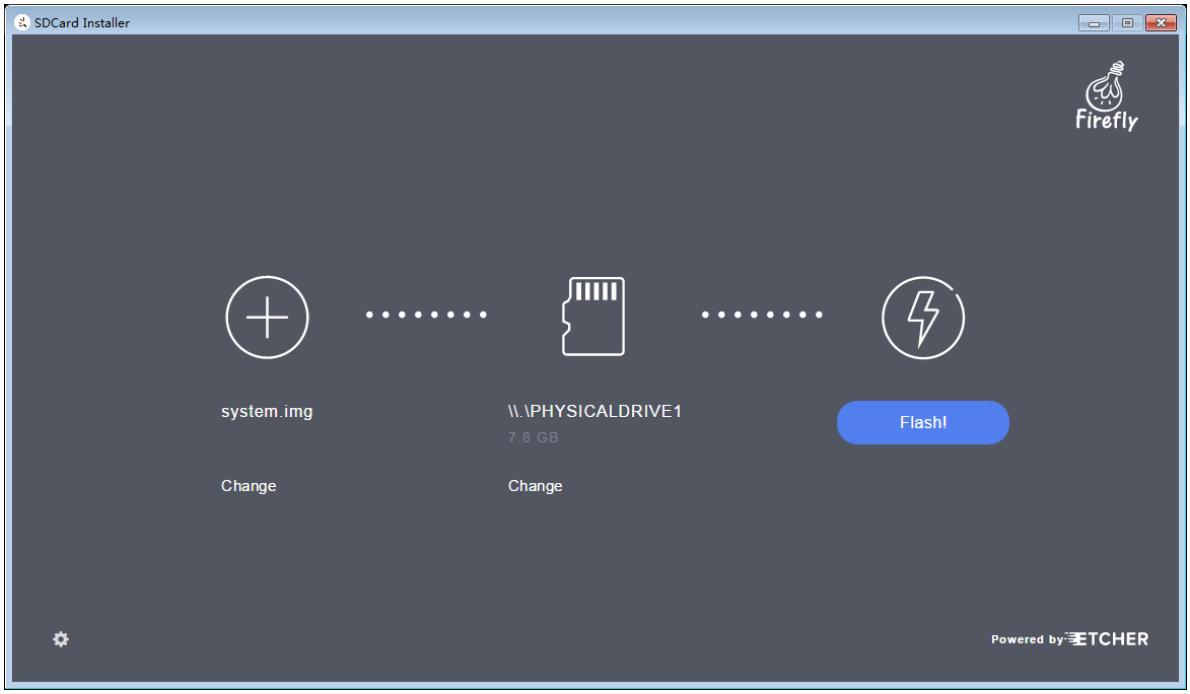
Instructions:

1. Download SDCard Installer from the Download Page.
2. Install and run:
   - Windows: Extract the archive file and run the setup executable inside. After installation, run SDCard Installer **as administrator** from the start menu.
   - Linux: Extract the archive file and run the `.AppImage` file inside.
   - Mac: Double click the `.dwg` file, install to the system or run directly.
3. Click the "Choose an OS" button, and select "ROC-RK3328-CC" in the "Please select your device" combobox.
4. A list of available firmware is updated from the network and presented to you, as illustrated below:



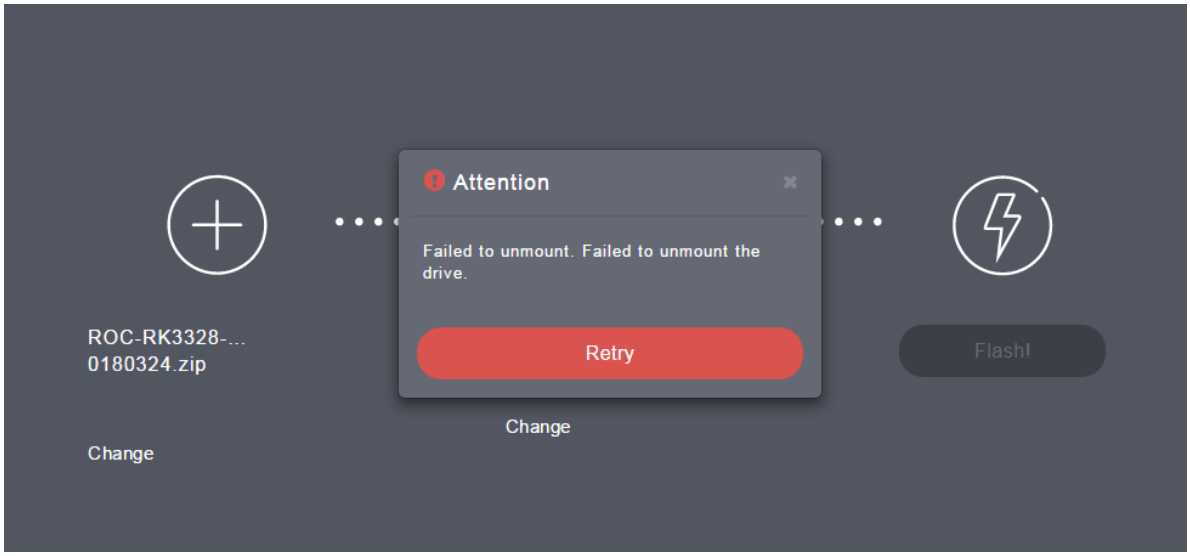
5. Choose an firmware OS, and click "OK" button. To flash local firmware, drag it from your local drive and drop to SDCard Installer.

6. Plug in the SD card. It should be automatically selected. If there are multiple SD cards, click the "Change" button to choose one.
7. Click the "Flash!" button. SDCard Installer will start to download the firmware, flash to the SD card, and verify the content. Please wait patiently.



Note:

- To run SDCard Installer with proper permission in Windows, you need to right click the shortcut and select **Run as administrator**.
- Sometimes, when the progress reaches to 99% or 100%, an error of unmounting the SD card may occur, which can be ignored and does no harm to the data flashed to the SD card.



- The downloaded firmware will be saved to the local directory, which will be reused the next time you flash the same firmware again. The download directory can be set by clicking the setting icon in the bottom left of the

main window and changing the "Download Location:" field.

3.4 Etcher

Compared with SDCard Installer, Etcher lacks of firmware list and download. But its code is newer. If you have any flashing problem with the SDCard Installer, you can try Etcher, reusing the firmware file in the download directory of SDCard Installer.

Etcher can be downloaded from the Etcher official site. Installation and usage are similiar with SDCard Installer.

3.5 dd

dd is a commonly used command line tool in Linux.

First, plug in the SD card, and unmount it if it is automatically mounted by the file manager.

Then find the device file of the SD card by checking kernel log:

```
dmesg | tail
```

If the device file is `/dev/mmcblk0`, use the following command to flash:

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

Flashing takes lots time and the command above does not show the progress. We can use another tool `pv` to do this job.

First install `pv`:

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

Then add `pv` to the pipe to report progress:

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

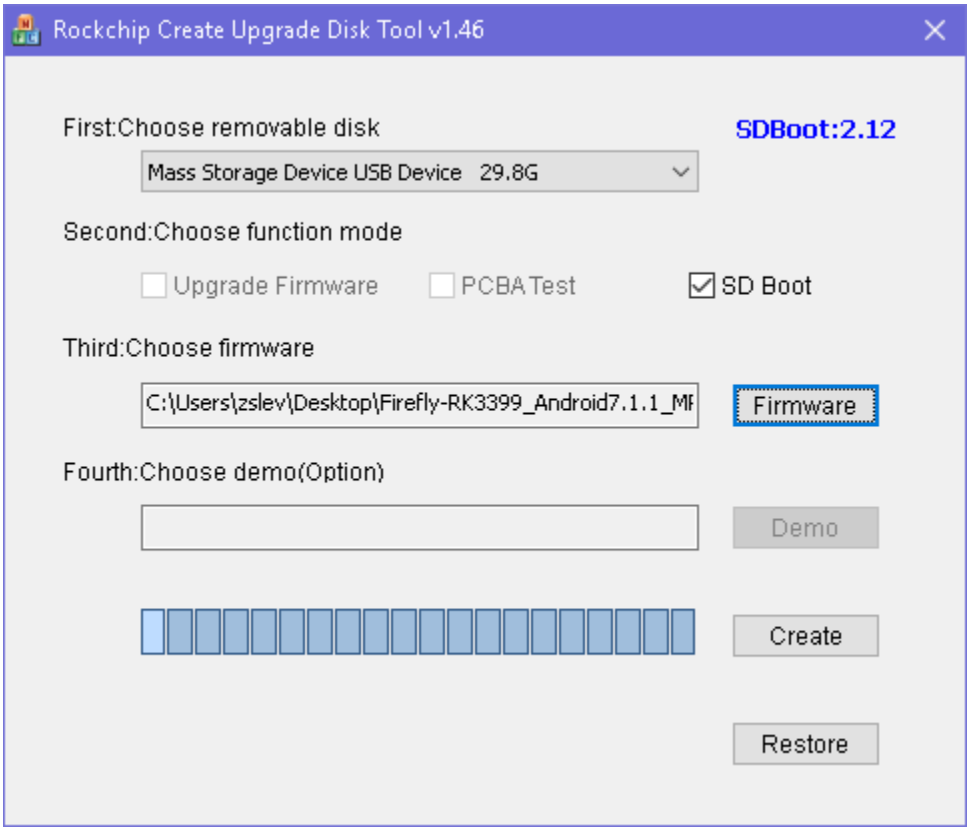
3.6 SD Firmware Tool

NOTE: This section is about how to flash RK Firmware of Android to the SD card.

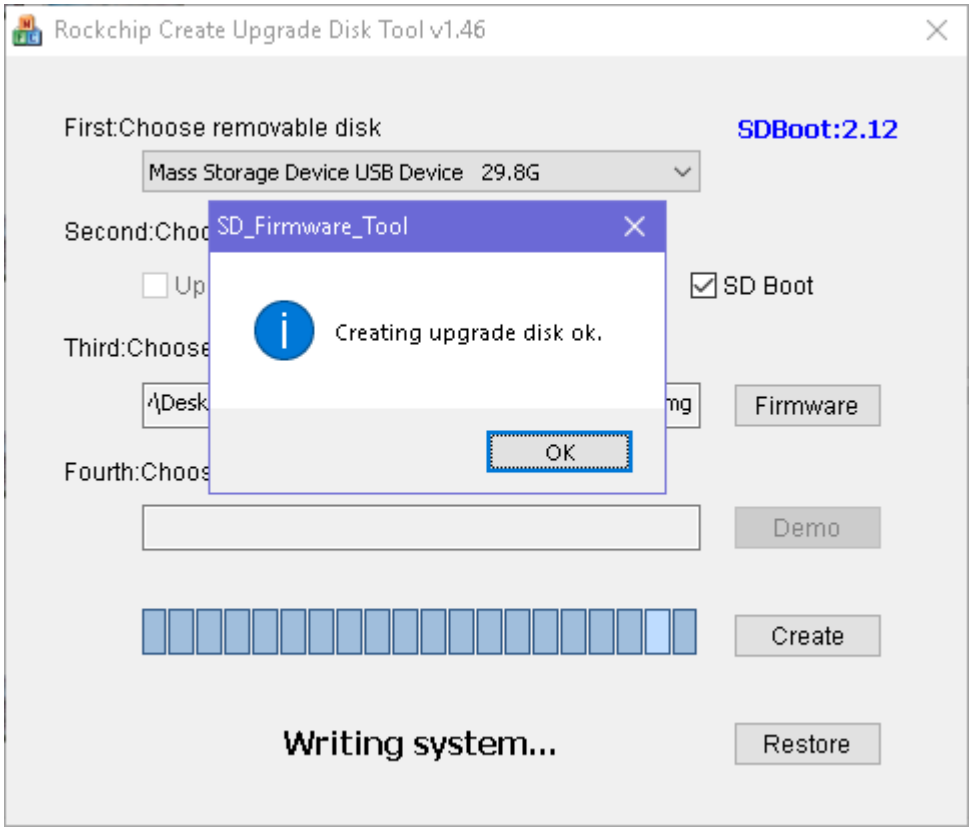
First, you will need to download SD Firmware Tool from the SD Firmwware Tool Download Page and extract it.

After extraction, in the directory of SD Firmware Tool, edit `config.ini` by changing the 4th line from `Selected=1` to `Selected=2`, in order to select English as the default user interface language.

Run `SD_Firmware_Tool.exe`:



1. Plug in the SD card.
2. Select the SD card from the combo box.
3. Check on the “SD Boot” option.
4. Click “Firmware” button, and select the firmware in the file dialog.
5. Click “Create” button.
6. A warning dialog will show up. By making sure you have the right SD card device selected, select “Yes” to continue.
7. Wait for the operation to complete, until the info dialog shows up.



8. Plug out the SD card.

CHAPTER 4

Flashing to the eMMC

4.1 Boot Mode

eMMC flash is commonly soldered directly to the board. Some eMMC flash are pluggable, but it is hard to find a reader to use on PC. Therefore, eMMC is generally flashed onboard, that is, running to tiny system on the obard, which reads firmware data from PC and flashes to eMMC.

Depending on the existing data on the eMMC flash, there are two special boot modes: Rockusb Mode and Maskrom Mode.

You usually just need to enter Rockusb Mode for upgrading an existing Android OS or Firefly Ubuntu OS, which is packed with RK Firmware format.

Maskrom Mode is the last resort when Rockusb Mode is not available due to bootloader damage, or you need to flash Raw Firmware to eMMC.

4.1.1 Rockusb Mode

If the board powers on and finds a valid IDB (IDentity Block) in the eMMC, it will continue to load the bootloader from the eMMC and pass execution control to it. If the bootloader checks that the Recovery button is pressed and USB connection is made, then it enters the so-called Rockusb Mode, waiting for further instructions from the host.

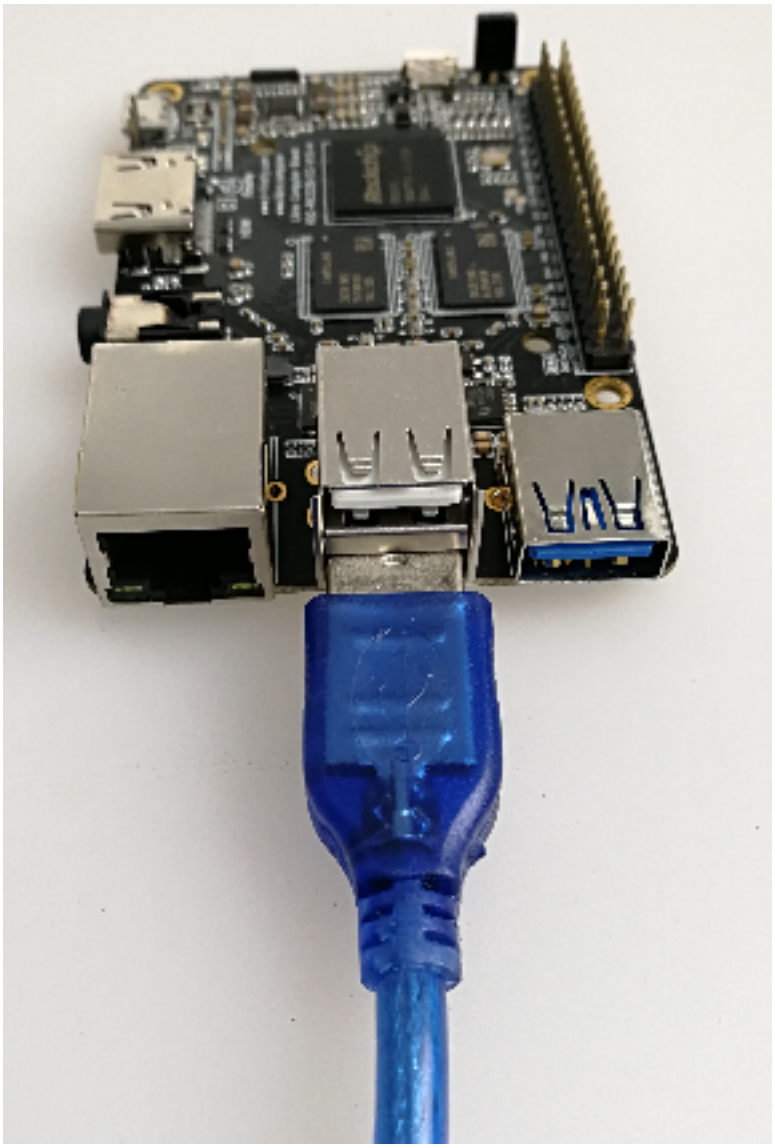
Requirement:

- 5V2A power adapter.
- Micro USB cable to connect power adapter and board.
- Male to male USB cable to connect host PC and board.
- eMMC.

Steps:

1. Pull all the USB cables (including micro USB cable and male to male USB cable) out of the board, to keep the board powering off.

2. Install the eMMC and pull out the SD card.
3. Use the male to male USB cable to connect the host PC with the USB 2.0 OTG port (the lower one in the double-decker ports) of the board.



4. Keep the RECOVERY button on the board pressed.
5. Plug in the micro USB cable to the board to power up.
6. Wait about 3 seconds before releasing the RECOVERY button.

4.1.2 Maskrom Mode

If anyone of the following conditions is met when the board powers on:

- eMMC is empty.
- The bootloader on eMMC is damaged.
- eMMC read data failed by connecting eMMC data/clock pin to ground.

then no valid IDB (IDentity Block) will be found in the eMMC. The CPU will execute a small ROM code, waiting for the host to upload via USB a small DDR blob to initialize the DDR memory, and later a full bootloader to handle further firmware upgrading. This mode is called Maskrom Mode.

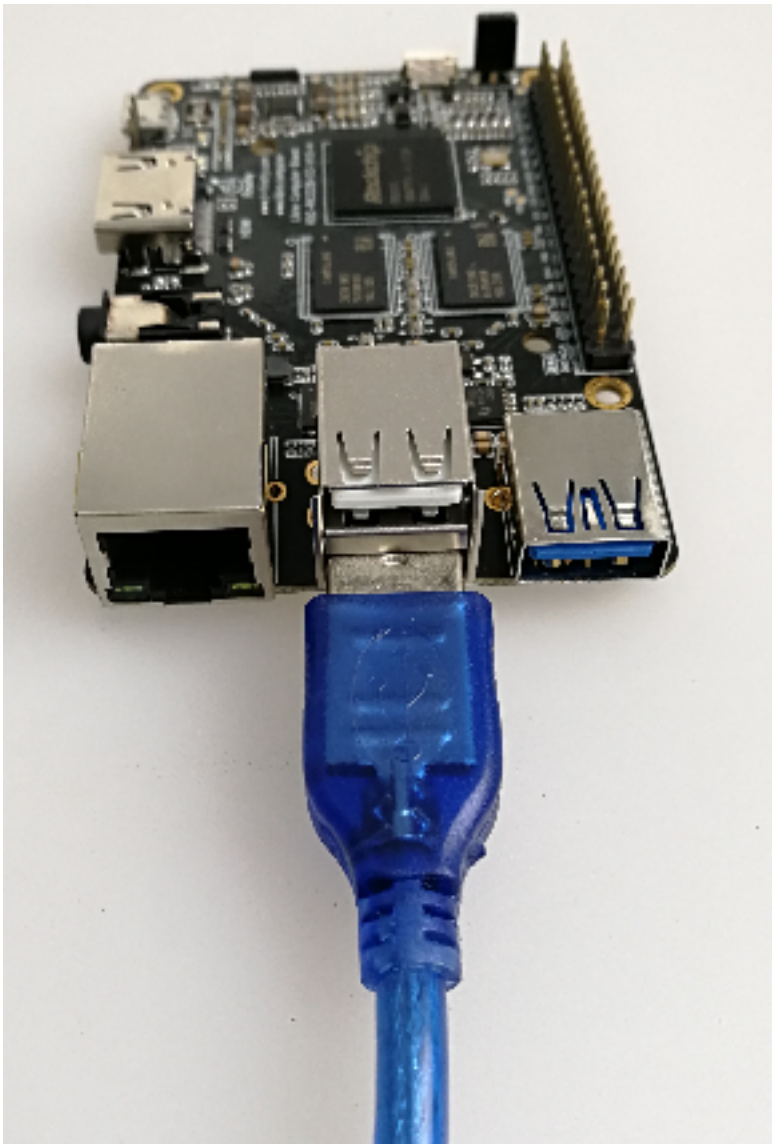
It involves hardware operation to force into MaskRom Mode, which has certain risk and should be carried out **VERY CAREFULLY**.

Requirement:

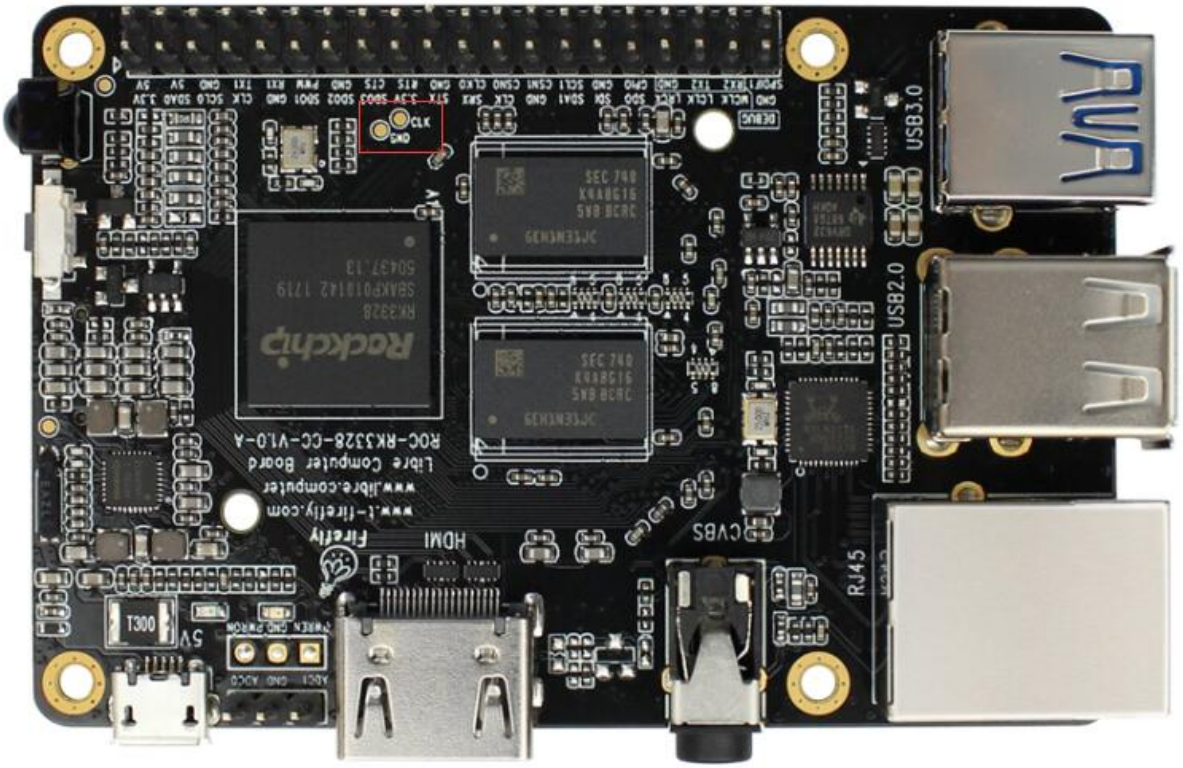
- 5V2A power adapter.
- Micro USB cable to connect power adapter and board.
- Male to male USB cable to connect host PC and board.
- Metal tweezers to connect eMMC clock pin to ground.
- eMMC.

Steps:

1. Pull all the USB cables (including micro USB cable and male to male USB cable) out of the board, to keep the board power off.
2. Install the eMMC and pull out the SD card.
3. Use a male to male USB cable to connect your host PC and USB OTG port of the board:



4. Find the reserved eMMC CLK and GND pads on the board, as shown below:



5. Connect the eMMC CLK and GND pads with metal tweezers and keep holding steadily.
6. Plug in the micro USB cable to the board to power on.
7. Wait about 1 seconds before releasing the metal tweezers.

4.2 Flashing Tools

Please use the eMMC flashing tools according to your OS:

- To flash to the eMMC:
  - GUI
    - AndroidTool (Windows)
  - CLI
    - upgrade_tool (Linux)
    - rkdeveloptool (Linux)

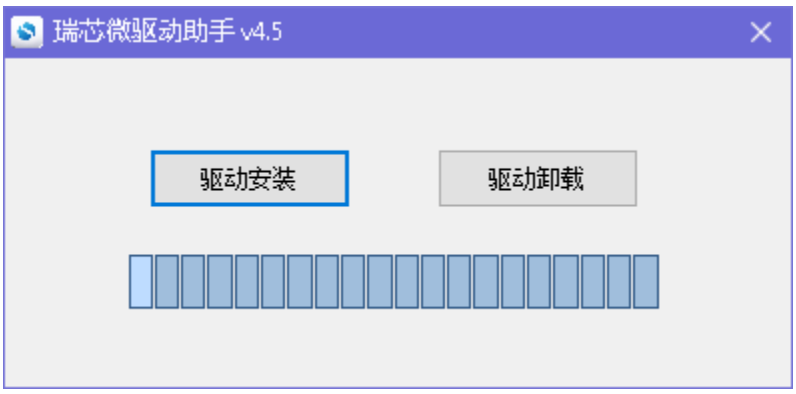
4.3 AndroidTool

AndroidTool is used to flash Raw Firmware, RK Firmware and Partition Image to eMMC.

To use AndroidTool, you need to install Rockusb Driver first.

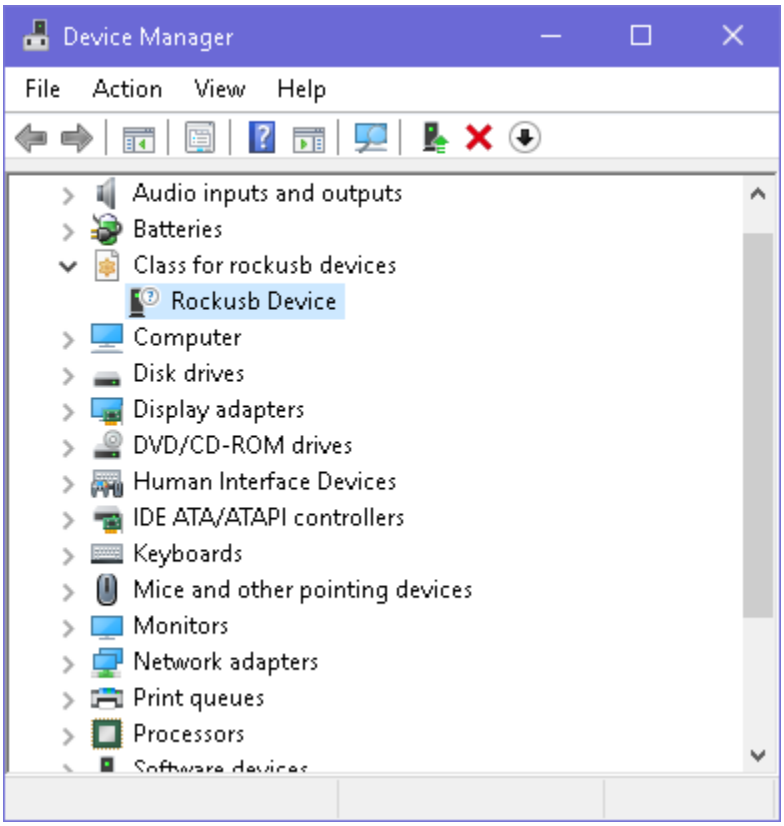
4.3.1 Installing Rockusb Driver

Download DriverAssistant, extract the archive and run `DriverInstall.exe` inside.



Click the "驱动安装" button to install the driver. If you want to uninstall the driver, click the "驱动卸载" button.

If your device is in Rockusb Mode or Maskrom Mode, you'll find a `Rockusb Device` in the device manager:

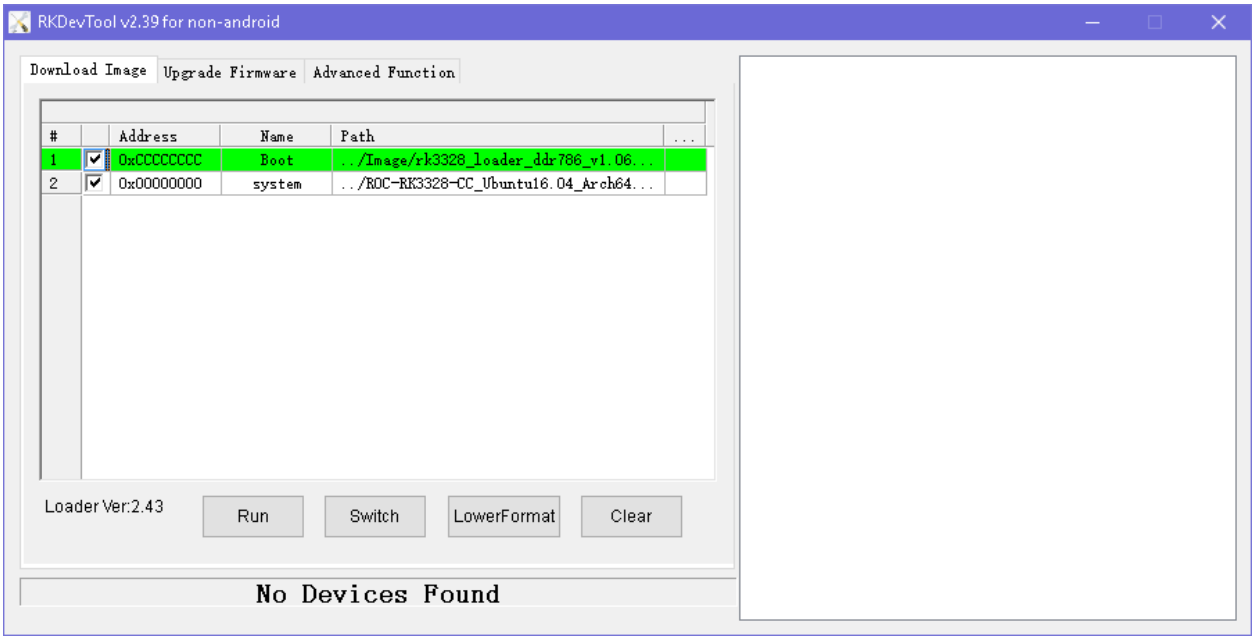


4.3.2 Installing AndroidTool

AndroidTool Download Page

Download AndroidTool, extract it. Locate the file named `config.ini`, and edit it by changing the 4th line from `Selected=1` to `Selected=2`, in order to select English as the default user interface language.

Launch `AndroidTool.exe`:



If your device is in Rockusb Mode, the status line will be "Found One LOADER Device".

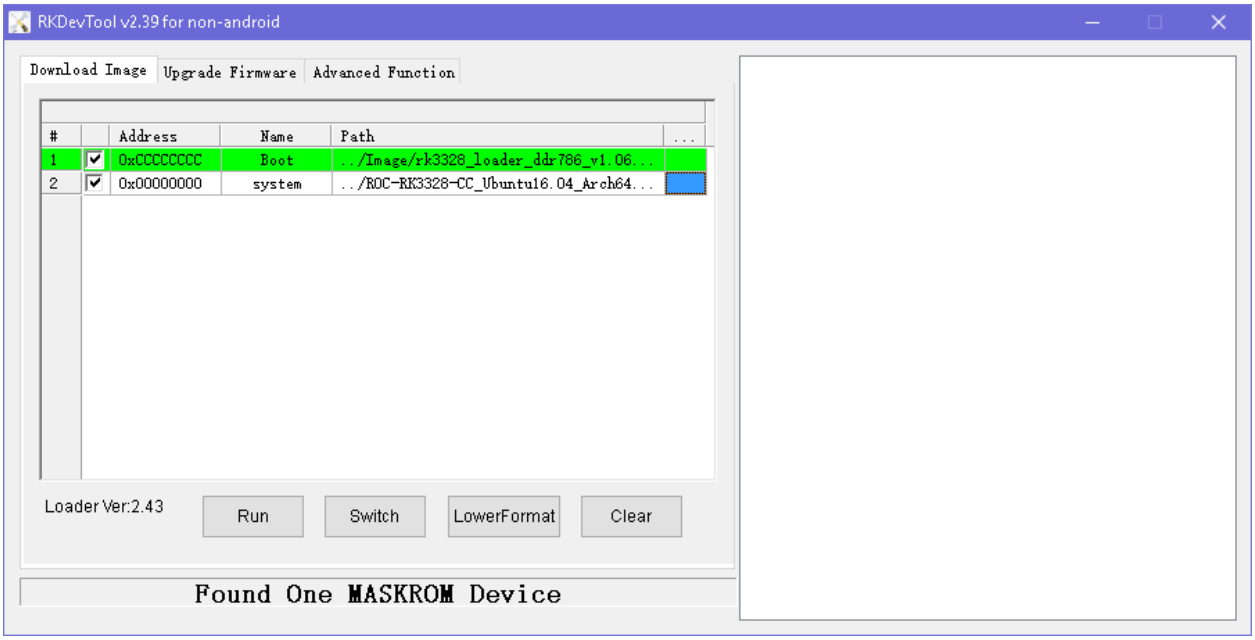
If your device is in Maskrom Mode, the status line will be "Found One MASKROM Device".

4.3.3 Flashing Raw Firmware

Raw Firmware needs to be flashed to offset 0 of eMMC storage. However, in Rockusb Mode, all LBA writes are offset by 0x2000 sectors. Therefore, the device has to be forced into Maskrom Mode.

To flash Raw Firmware to the eMMC using AndroidTool, follow the steps below:

1. Force the device into Maskrom Mode.
2. Run AndroidTool.
3. Switch to the "Download Image" tab page.
4. Keep the first line of the table unchanged, using the default loader file.
5. Click the right blank cell on the second line, which will pop up a file dialog to open the Raw Firmware file.
6. Click the "Run" button to flash.



4.3.4 Flashing RK Firmware

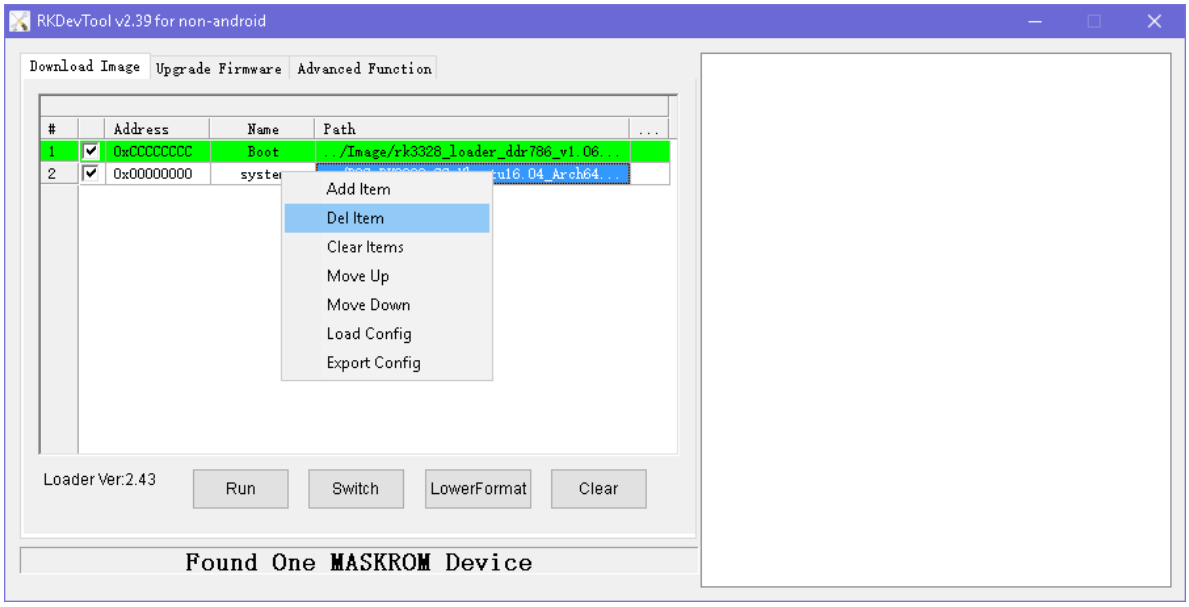
To flash RK Firmware to the eMMC using AndroidTool, follow the steps below:

1. Force the device into Rockusb Mode or Maskrom Mode.
2. Run AndroidTool.
3. Switch to the "Upgrade Firmware" tab page.
4. Click the "Firmware" button, which will pop up a file dialog to open the RK Firmware file.
5. The firmware version, loader version and chip info will be read and displayed.
6. Click the "Upgrade" button to flash.

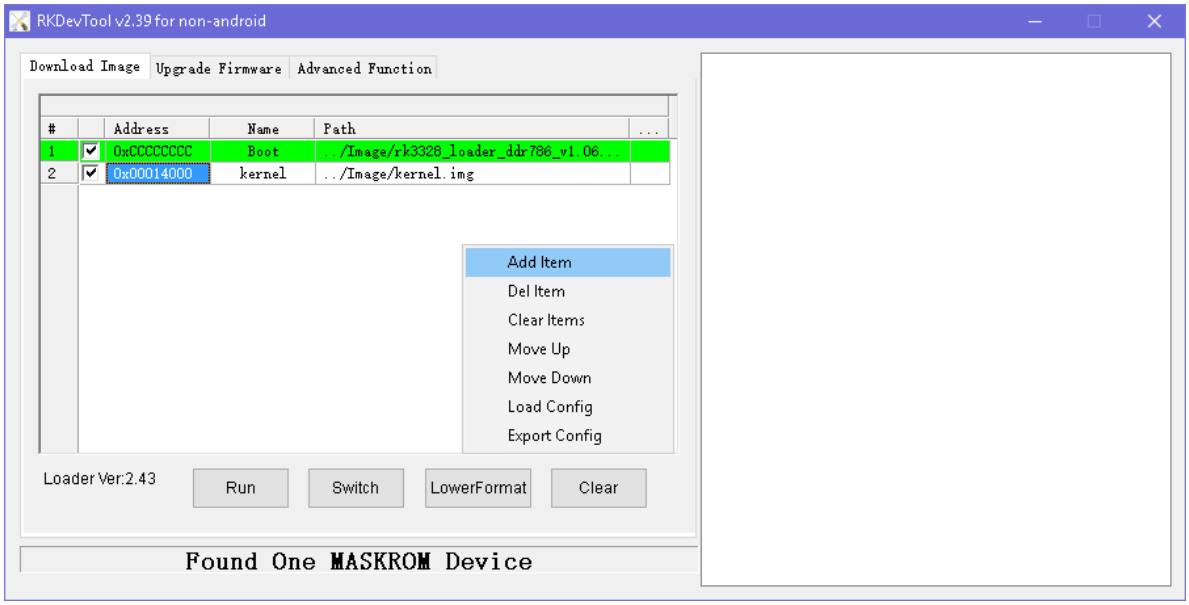
4.3.5 Flashing Partition Image

To flash Partition Image to the eMMC using AndroidTool, follow the steps below:

1. Force the device into Rockusb Mode or Maskrom Mode.
2. Run AndroidTool.
3. Switch to the "Download Image" tab page.
4. Keep the first line of the table unchanged.
5. Delete all others unused rows by selecting "Delete Item" from the right-click popup menu.



6. Add partition image to flash by selection “Add Item” from the right-click popup menu.
   - Check on the checkbox on the first cell.
   - Fill in the address with the sector offset (plus `0x2000` if in Maskrom Mode) of partition in `parameter.txt` file.
   - Click the right blank cell to browse to the Partition Image file.



7. Click the “Run” button to flash.

Note:

- You can add multiple partitions to flash by repeating step 6.
- You can skip the partition flashing by checking off the checkbox in front of the address cell.
- In Maskrom Mode, you must add `0x2000` to the sector offset of the partition in `parameter.txt`. See *Partition Offset* for more detail.

4.4 upgrade_tool

upgrade_tool is a close-sourced command line tool provided by Rockchip, which supports flashing Raw Firmware, RK Firmware and Partition Image to the eMMC.

4.4.1 Installing upgrade_tool

upgrade_tool Download Link

Download upgrade_tool, and install it to your Linux host:

```
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
```

Then add `udev` rules by instructions *here*, in order to have permission for the normal user to flash Rockchip devices. If you skip this, you must prefix the following commands with `sudo` to have the right permission.

4.4.2 Flashing Raw Firmware

Raw Firmware needs to be flashed to offset 0 of eMMC storage. However, in Rockusb Mode, all LBA writes are offset by 0x2000 sectors. Therefore, the device has to be forced into Maskrom Mode.

To flash Raw Firmware to the eMMC using upgrade_tool, follow the steps below:

1. Force the device into Maskrom Mode.
2. Run:

```
upgrade_tool db           out/u-boot/rk3328_loader_ddr786_v1.06.243.bin
upgrade_tool wl 0x0       out/system.img
upgrade_tool rd           # reset device to boot
```

Note:

- `rk3328_loader_ddr786_v1.06.243.bin` is the copied loader file after compiling `U-Boot`. It can also be downloaded from here (choose `rk3328_loader_xxx.bin` file).
- `system.img` is Raw Firmware after packing, which can also be Raw Firmware downloaded from official site (decompress first).

4.4.3 Flashing RK Firmware

To flash RK Firmware to the eMMC using upgrade_tool, follow the steps below:

1. Force the device into Rockusb Mode or Maskrom Mode.
2. Run:

```
upgrade_tool uf update.img
```

4.4.4 Flashing Partition Image

You can write individual Partition Image to the eMMC. Depending on the original content of the eMMC, the instructions can be somewhat different.

Raw Firmware

If the original firmware format is raw, chances are that it is using the `GPT` partition scheme, and the predefined offset and size of each partition can be found in `build/partitions.sh` in the SDK. See *Partition Offset* for more detail.

To flash Partition Image to the eMMC using upgrade_tool, follow the steps below:

1. Force the device into Maskrom Mode.
2. Use upgrade_tool to flash the Partition Image:

```
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         # reset device to boot
```

RK Firmware

If the original firmware format is Rockchip, it is using the `parameter` file for partition scheme, and you can use the partition name to flash Partition Image directly.

To flash the Partition Image to the eMMC using upgrade_tool, follow the steps below:

1. Force the device into Rockusb Mode.
2. Use upgrade_tool to flash the Partition Image:

```
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   # flash parameter
upgrade_tool ul bootloader.bin # flash bootloader
```

Note:

- `-b` is a predefined shortcut for `boot` partition. If no shortcuts are available, use partition name instead (`resource` in above example).
- You can customize kernel parameters and partition layout according to Parameter file format. Once the partition layout is changed, you must flash the `parameter` file first, before reflashing other changed partitions.

4.4.5 FAQ

If errors occur due to flash storage problem, you can try to low format or erase the flash by:

```
upgrade_tool lf   # low format flash
upgrade_tool ef   # erase flash
```

4.5 rkdeveloptool

rkdeveloptool is an open-source command line flashing tool developed by Rockchip, which is an alternative to the close-source upgrade_tool.

rkdeveloptool **DOES NOT** support proprietary RK Firmware.

4.5.1 Installing rkdeveloptool

First, download, compile and install 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
```

Then add `udev` rules by instructions *here*, in order to have permission for the normal user to flash Rockchip devices. If you skip this, you must prefix the following commands with `sudo` to have the right permission.

4.5.2 Flashing Raw Firmware

Raw Firmware needs to be flashed to offset 0 of eMMC storage. However, in Rockusb Mode, all LBA writes are offset by 0x2000 sectors. Therefore, the device has to be forced into Maskrom Mode.

To flash Raw Firmware to the eMMC using rkdeveloptool, follow the steps below:

1. Force the device into Maskrom Mode.
2. Run:

```
rkdeveloptool db            out/u-boot/rk3328_loader_ddr786_v1.06.243.bin
rkdeveloptool wl 0x0        out/system.img
rkdeveloptool rd            # reset device to boot
```

Note:

- `rk3328_loader_ddr786_v1.06.243.bin` is the copied loader file after compiling `U-Boot`. It can also be downloaded from here (choose `rk3328_loader_xxx.bin` file).
- `system.img` is Raw Firmware after packing, which can also be the Raw Firmware downloaded from official site (decompress it first).

4.5.3 Flashing Partition Image

The following instructions **ONLY APPLIY** to boards which are flashed with Raw Firmware and use `GPT` partition scheme. The predefined offset and size of each partition can be found in `build/partitions.sh` in the SDK. See *Partition Offset* for more detail.

To flash Partition Image to the eMMC using rkdeveloptool, follow the steps below:

1. Force the device into Maskrom Mode.

2. Run:

```
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            # reset device to boot
```

4.6 udev

Create `/etc/udev/rules.d/99-rk-rockusb.rules` with following content[1]. Replace the group `users` with your actual Linux group if neccessary:

```
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"
```

Reload the udev rules to take effect without reboot:

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

4.7 Partition Offset

4.7.1 GPT Partition

The offset of partition image can be obained by following command(assuming you are in the directory of Firefly Linux SDK):

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

which gives result of:

```
ATF        0x00006000
BOOT       0x00008000
```

(continues on next page)

(continued from previous page)

```
LOADER1     0x00000040
LOADER2     0x00004000
RESERVED1   0x00001f80
RESERVED2   0x00002000
ROOTFS      0x00040000
SYSTEM      0x00000000
```

4.7.2 parameter

If RK Firmware is used, `parameter.txt` is used to define partition layout.

Here's a handy script to list the partition offsets in `parameter.txt`:

```
#!/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
```

Save it as a script in `/usr/local/bin/show_rk_parameter.sh` and give the script executing permission.

Here's an example of showing partition offsets defined in `RK3328 Android SDK`:

```
$ 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
cache                0x00062000 0x00040000
metadata             0x000a2000 0x00008000
kpanic               0x000aa000 0x00002000
```

(continues on next page)

(continued from previous page)

```
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

Serial Debug

If you're doing serious U-Boot or kernel development, a USB serial adapter (short for the USB serial TTL bridge adapter) is very useful to check the system startup log, especially in case of no graphics desktop shows up.

5.1 Preparing a USB Serial Adapter

5.1.1 Choosing a USB Serial Adapter

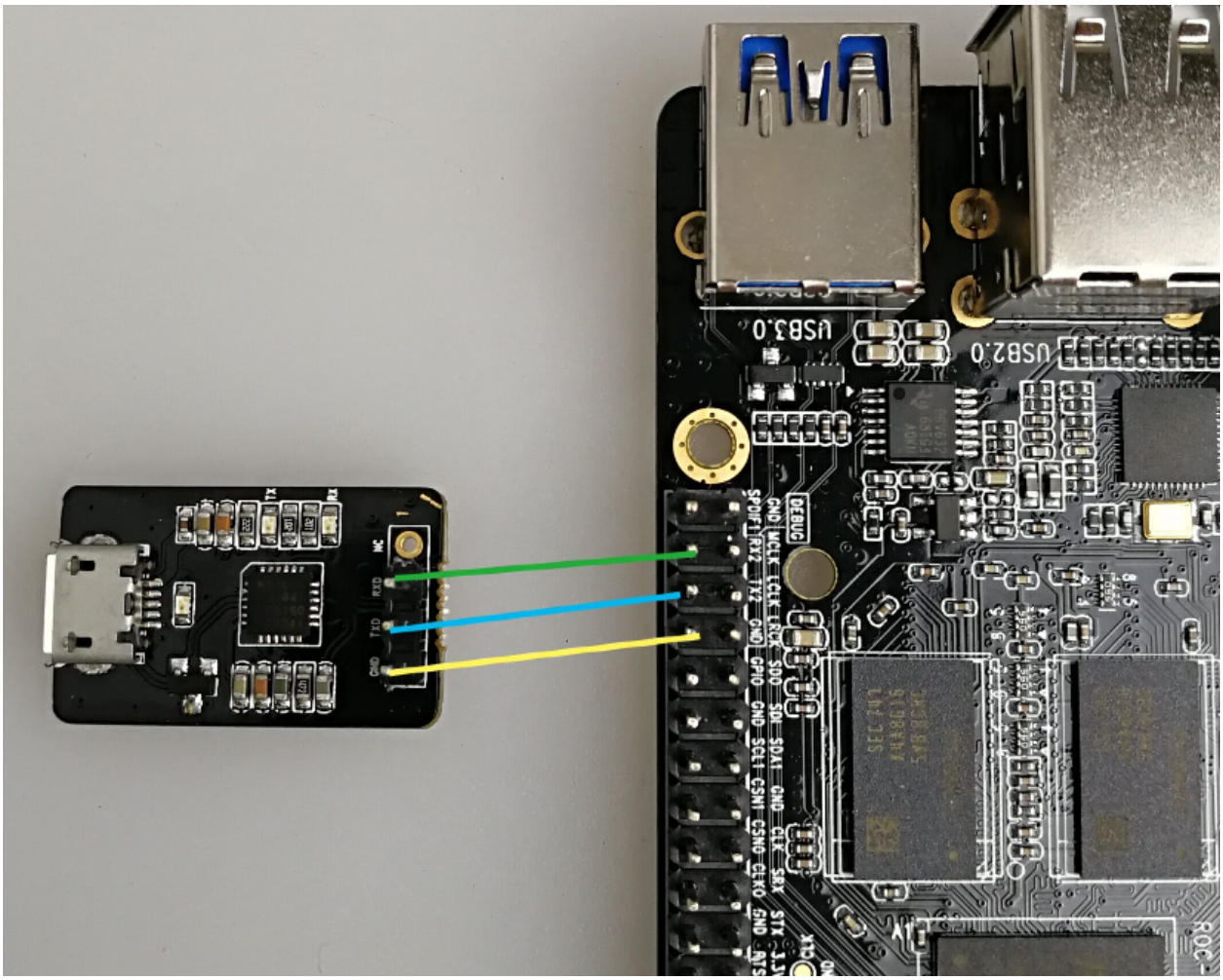
ROC-RK3328-CC defaults to use **1,500,000** baud rate for its UART debug port, with TTL voltage.

Some adapters cannot support such high baud rate. So before purchase, make sure it meets the requirements and has the proper driver for your OS.

You can refer to the USB Serial Adapter with `CP2104` chip in the Firefly Online Shop.

5.1.2 Connecting to the Debug Port

Three wires are required to connect the TX/RX/GND pins together:



With some adapters, you may try to connect TX pin of the adapter to RX pin of the board, and RX pin of adapter to TX pin of the board, if you do not get the serial console working.

5.1.3 Serial Parameters

ROC-RK3328-CC uses the following serial parameters:

- Baud rate: 1,500,000
- Data bit: 8
- Stop bit: 1
- Parity check: none
- Flow control: none

Next, depending on the OS you are using, we'll guide you through the detailed steps.

5.2 Serial Debugging in Windows

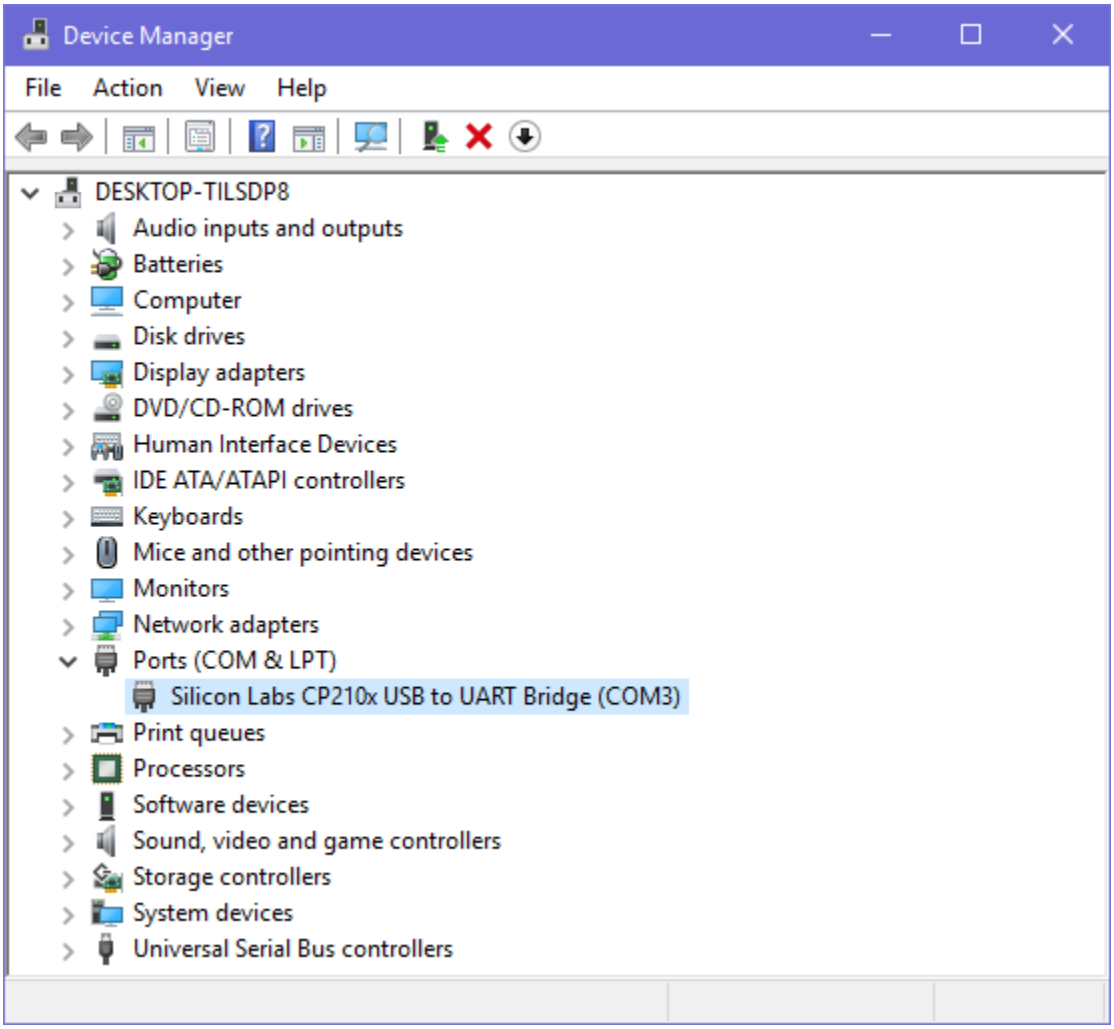
5.2.1 Install Driver

Use the recommended driver from your USB serial adapter vendor. If not available, you can check the chipset and try the following drivers:

- CH340
- PL2303
- CP210X

  Tips: If PL2303 does not work under Win8, you can try to downgrade the driver to version 3.3.5.122 or before.

After installing the driver, connect the adapter to the USB port. The OS will prompt that a new hardware is detected. When it finishes, you can find the new COM port in the device manager.



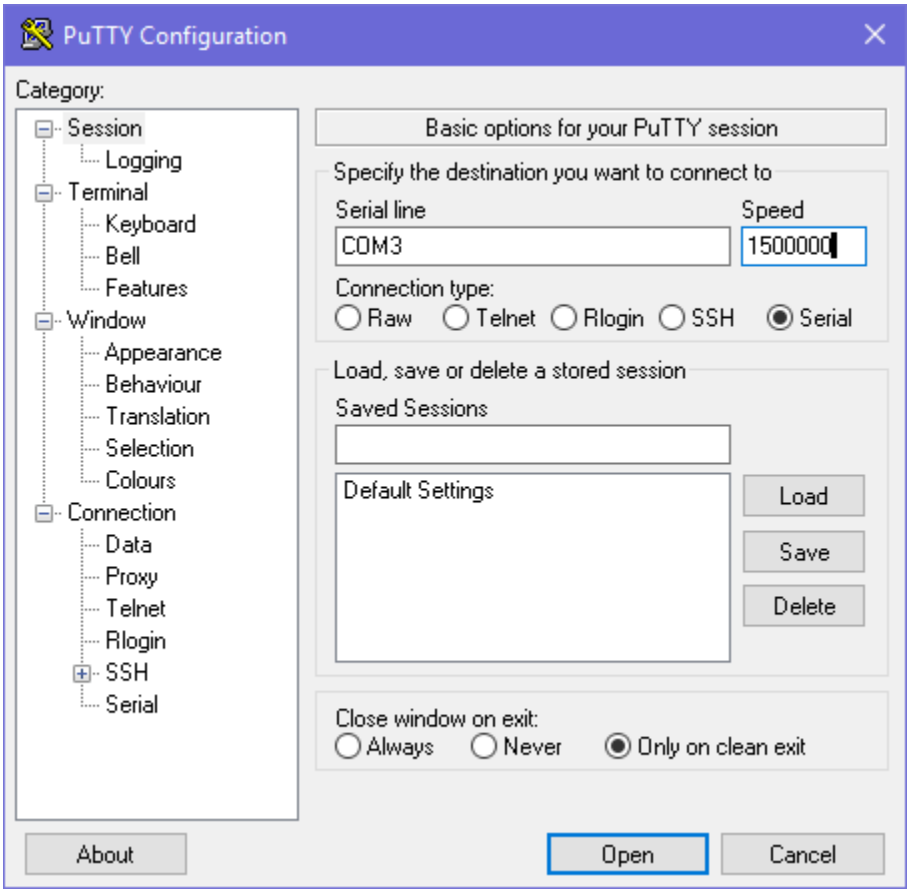
5.2.2 Putty

There are many serial console utilities in Windows, such as putty and SecureCRT. Putty, a popular opensource software, is taken as an example here.

> Putty Download Page

Download `putty.zip`, extract it and run `PUTTY.exe`.

1. Select "Connection type" to "Serial".
2. Modify "Serial line" to the COM port found in the device manager.
3. Set "Speed" to 1500000.
4. Click "Open" button.



5.3 Serial Debugging in Linux

If the chipset of the USB serial adapter is supported by Linux kernel, the driver will be loaded automatically.

Connect the serial adapter, and check the corresponding serial device file by checking:

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

Add your Linux user to the `uucp` group, in order to have permission to access this device (or you may use `sudo` everytime):

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

The group change will take effect after logout/login Linux, or use the `newgrp` command to enter a shell with the new group:

```
newgrp uucp
```

Then proceed the following steps depending on your favor of `picocom` or `minicom`.

5.3.1 picocom

`picocom` is light weight, easy to use.

Install it with:

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

Start `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
```

The messages above show that `Ctrl-a` is the escape key. Pressing `Ctrl-a Ctrl-q` will quit `picocom` and return to the shell.

5.3.2 minicom

Install `minicom` with:

```
sudo apt-get install minicom
```

Start `minicom`:

```
$ minicom
Welcome to minicom 2.7

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
```

Based on the above tips, press `Ctrl-a` then `z` (not `Ctrl-z`) to bring up the Help menu.

```
+-----------------------------------------------------------------------+
|                       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.                  |
+-----------------------------------------------------------------------+
```

Press `O` as prompted to enter setup screen:

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

Choose “Serial port setup”, then press the upper-case letter in front of the option to set to values as illustrated below:

```
   +-----------------------------------------------------------------------+
   | 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                                        |
```

(continues on next page)

(continued from previous page)

```
| G - Software Flow Control : No                                              |
|                                                                             |
|    Change which setting?                                                    |
+-----------------------------------------------------------------------------+
```

Note:

- `Hardware Flow Control` and `Software Flow Control` should be set to `No`.
- After finishing the setting, go back to the previous menu by pressing `ESC`, and select `Save setup as dfl` to save as the default configuration.

CHAPTER 6

Compiling Linux Firmware

In this chapter, we'll walk through the steps of compiling Linux firmware for ROC-RK3328-CC.

6.1 Preparation

The Linux firmware is built under the following environment:

- Ubuntu 16.04 amd64

Install following packages:

```
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 Download the Linux SDK

Create the project directory:

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

Download the 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)

(continued from previous page)

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

You can also browse the source code online using the github links above.

The board building config is within:

```
build/board_configs.sh
```

6.3 Compiling U-Boot

Compile U-Boot:

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

Ouput:

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

- `rk3328_loader_ddr786_v1.06.243.bin`: A DDR init bin.
- `idbloader.img`: Image combined with DDR init bin and miniloader bin.
- `trust.img`: ARM trusted firmware.
- `uboot.img`: U-Boot image.

Related files:

- `configs/roc-rk3328-cc_defconfig`: default U-Boot config

6.4 Compiling Kernel

Compile kernel:

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

Ouput:

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

- `boot.img`: A image file containing `Image` and `rk3328-roc-cc.dtb`, in fat32 filesystem format.
- `Image`: Kernel image.

- `rk3328-roc-cc.dtb`: Device tree blob.

Related files:

- `arch/arm64/configs/fireflyrk3328_linux_defconfig`: default kernel config
- `arch/arm64/boot/dts/rockchip/rk3328-roc-cc.dts`: board dts
- `arch/arm64/boot/dts/rockchip/rk3328.dtsi`: soc dts

To customize the kernel config and update the default config:

```
# this is important!
export ARCH=arm64

cd kernel

# first use default config
make fireflyrk3328_linux_defconfig

# customize your kernel
make menuconfig

# save as default config
make savedefconfig
cp defconfig arch/arm64/configs/fireflyrk3328_linux_defconfig
```

NOTE: The build script does not copy kernel modules to the root filesystem. You have to do it yourself.

6.5 Building Root Filesystem

You can download the prebuilt root filesystem or build one yourself by following *Building Linux Root Filesystem*.

6.6 Packing Raw Format Firmware

Place your Linux root filesystem image file as `out/rootfs.img`.

The `out` directory should contain the following files:

```
$ 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
```

To create the [Raw Firmare]:

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

The command above will pack the neccessary image files into `out/system.img`, according to this Storage Map.

To flash this Raw Firmware, please follow the *Getting Started* chapter.

CHAPTER 7

Building Debian Root Filesystem

7.1 Preparing Build System

```
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 Compile the Root File System

1. Build the Debian basic system:

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

   This will use `live-build` to create a basic Debian Stretch desktop system, which is later packed as file `linaro-stretch-alip-*.tar.gz`. This operation takes time and only needs to run once unless you have modified the `live-build` configuration.

2. Build the rk-debian system:

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

   This will install/setup Rockchip components such as mali and video encode/decode, based on the `linaro-stretch-alip-*.tar.gz` file from previous step, to create the full rk-debian system.

3. Create the ext4 image file `linaro-rootfs.img`:

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

Note: default user and password is “linaro”.

CHAPTER 8

Building Ubuntu Root Filesystem

Environment:

- Ubuntu 16.04 amd64

Install required packages:

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

Download Ubuntu core:

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

Create a root filesystem image file sized 1000M and populate it with the ubuntu base tar file:

```
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` is the magic cure here, which make possible chrooting into an Arm64 filesystem under x86_64 host system.

Chroot to the new filesystem and initialize:

```
sudo chroot mnt/

# Change the setting here
USER=firefly
HOST=firefly

# Create User
useradd -G sudo -m -s /bin/bash $USER
```

(continues on next page)

(continued from previous page)

```
passwd $USER
# enter user password

# Hostname & Network
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

# Enable serial console
ln -s /lib/systemd/system/serial-getty\@.service /etc/systemd/system/getty.target.
↪wants/serial-getty@ttyS0.service

# Install packages
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
```

Unmount filesystem:

```
sudo umount rootfs/
```

Credit: bholland

8.1 Reference

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

CHAPTER 9

Compiling Android 7.1

9.1 Preparation

9.1.1 Hardware Requiremnts

Recommended hardware requirement of development workstation compiling Android 7.1:

- 64 bit CPU
- 16GB Physical memory + Swap memory
- 30GB Free disk space is used for building, and the source tree takes about 8GB

See also the hardware and software configuration stated in Google official document:

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

9.1.2 Software Requiements

Installing JDK 8

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

Installing required packages

```
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)

(continued from previous page)

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

9.2 Downloading Android SDK

Due to the huge size of the Android SDK, please select one of the following clouds to download `ROC-RK3328-CC_Android7.1.2_git_20171204.7z`:

- Baiduyun
- Google Drive

After the download completes, verify the MD5 checksum before extraction:

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

Then extract it:

```
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
```

Update the correct git remote:

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

Synchronize source code from gitlab:

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

You can also view the source code online at: https://gitlab.com/TeeFirefly/RK3328-Nougat/tree/roc-rk3328-cc

9.3 Compiling with Firefly Scripts

Compiling Kernel

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

Compiling U-Boot

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

Compiling Android

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

Compiling Everying

This will compile kernel, U-Boot and Android with a single command:

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

9.4 Compiling Without Script

Before compilation, execute the following commands to configure environment variables:

```
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
```

Compiling Kernel

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

Compiling U-Boot

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

Compiling Android

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

9.5 Packing Rockchp Firmware

Packing Fimware in Linux

After compiling you can use Firefly official script to pack all partition image files into the one true Rockchip firmware or the one true original firmware, by executing the following command:

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

original firmware:
./FFTools/mkupdate/sd_mkupdate.sh update
```

The resulting file is `rockdev/Image-rk3328_firefly_box/update.img`.

The RK firmware needs to use the `SD_Firmware_Tool` tool and the function mode to select `SD Startup` to create the boot card, and original firmware can use the `SDCard-installer` to make the boot card.

Packing Fimware in Windows

It is also very simple in packaging Rockchip firmware `update.img` under Windows:

1. Copy all the compiled files in `rockdev/Image-rk3328_firefly_box/` to the `rockdev\Image` directory of AndroidTool
2. Run the `mkupdate.bat` batch file in the `rockdev` directory of AndroidTool.

3. update.img will be created in rockdev\Image directory.

9.6 Partition Images

update.img is the firmware released to end users, which is convenient to upgrade the system of the deveopment board.

During development cycle, it is a great time saving to only flash modified partition images.

Here's a table summarising the partition image in various stage:

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

Note that by excuting ./mkimage.sh, boot.img and system.img will be repacked wth the compiled results of Android in out/target/product/rk3328_firefly_box/ directory, and all related image files will be copied to the directory rockdev/Image-rk3328_firefly_box/.

The following is a list of the image files:

- boot.img: Android initramfs image, contains base filesystem of Android root directory, whcih is responsible for initializing and loading the system partition.
- system.img: Android system partition image in ext4 filesystem format.
- kernel.img: kernel image.
- resource.img: Resource image, containing boot logo and kernel device tree blob.
- misc.img: misc partition image, responsible for starting the mode switch and first aid mode parameter transfer.
- recovery.img: Recovery mode image.
- rk3328_loader_v1.08.244.bin: Loader files.
- uboot.img: U-Boot image file.
- trust.img: Arm trusted file (ATF) image file.
- parameter.txt: Partition layout and kernel command line.

CHAPTER 10

Unpack/Packing Rockchip Firmware

10.1 Rockchip Firmware Format

The rockchip firmware `release_update.img`, contains the boot loader `loader.img` and the real firmware data `update.img`:

release_update.img

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

`update.img` is packed with multiple image files, described by a control file named `package-file`. A typical `package-file` is:

```
# 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`: packing description of `update.img`, which is also included by `update.img`.
- `Image/MiniLoaderAll.bin`: The first bootloader loaded by cpu rom code.
- `Image/parameter.txt`: Parameter file where you can set the kernel boot parameters and partition layout.

- `Image/trust.img`: The Arm Trusted Image.
- `Image/misc.img`: misc partition image, used to control boot mode of Android.
- `Image/kernel.img`: Android kernel image.
- `Image/resource.img`: Resource image with boot logo and kernel device tree blob.
- `Image/boot.img`: Android initramfs, a root filesystem loaded in normal boot, contains important initialization and services description.
- `Image/recovery.img`: Recovery mode image.
- `Image/system.img`: Android system partition image.

Unpacking is extracting `update.img` from `release_update.img`, and then unpacking all the image files inside.

While repacking, it is the inverse process. It synthesizes the image files described by the `package-file`, into `update.img`, which will be further packed together with the bootloader to create the final `release_update.img`.

10.2 Installation of Tools

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

10.3 Unpacking Rockchip Firmware

- Unpacking `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
```

- Unpacking `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
```

(continues on next page)

(continued from previous page)

```
RESERVED                0x00000000    0x00000000
UnPack OK!
```

- Check the file tree in the update directory:

```
$ 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
```

10.4 Packing Rockchip Firmware

First of all, make sure `system` partition in `parameter.txt` file is larger enough to hold `system.img`. You can reference Parameter file format to understand the partition layout.

For example, in the line prefixed with "CMDLINE" in `parameter.txt`, you will find the description of `system` partition similiar to the following content:

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

The heximal string before the "@" is the partiton size in sectors (1 sector = 512 bytes here), therefore the size of the system partition is:

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

To create `release_update_new.img`:

```
# The current directory is still update/, which contains package-file,
# and files that package-file lists still exist
# Copy the parameter file to paramter, because afptool is used by default

$ 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
```

(continues on next page)

(continued from previous page)

```
Add file: ./Image/resource.img
Add file: ./Image/kernel.img
Add file: ./Image/boot.img
Add file: ./Image/recovery.img
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!
```

10.5 Customization

10.5.1 Customizing system.img

system.img is an ext4 file system format image file which can be mounted directly to the system for modification:

```
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

# When finished, you need to unmount it
cd /
sudo umount /mnt/system
```

Note that the free space of `system.img` is almost 0. If you need to expand the image file, do adjust the partition layout in `parameter.txt` accordingly.

The following is an example of how to increase the size of the image file by 128MB.

Before expanding, make sure `system.img` is not mounted by running:

```
mount | grep system
```

Resize the image file:

```
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 11

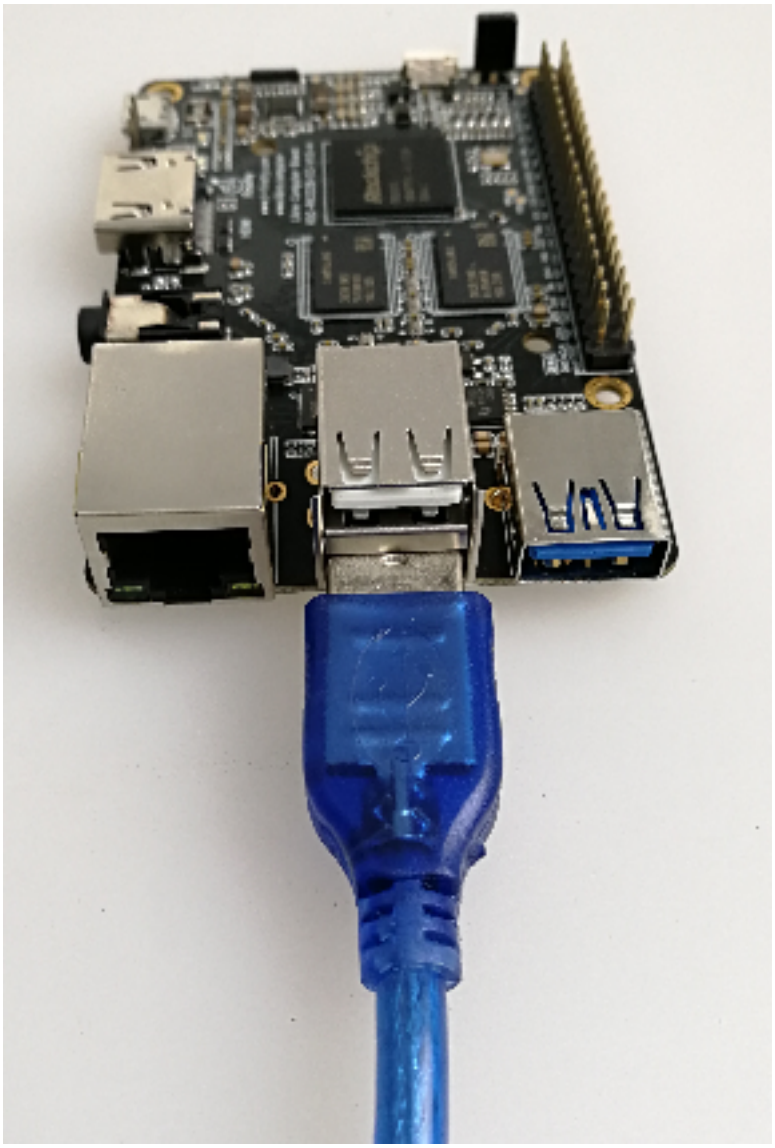
Adb Instructions

`Adb`, short for Android Debug Bridge, is a versatile command line debugging tools, which has a variety of functions, such as tracking system logs, upload and download files, install and uninstall applications.

11.1 Preparation

In order to use `adb`, you need to:

1. Use the male to male USB cable to connect host PC with the lower USB OTG port of the board:



2. In Android running on the board, select `Settings` -> `USB`, and check on `Connect to PC` option.
3. Install adb driver and command, depending on your OS.

11.1.1 Adb Installation in Windows

1. Install Rockusb Driver.
2. Download adb.zip, then unzip it to `C:\adb` for convenience.

Open a `cmd` window and run:

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

You are entering the adb shell if everything goes well.

11.1.2 Adb Installation in Ubuntu

1. Install adb tool:

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

2. Add device ID:

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

3. Add udev rules for non-root user use:

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

4. Reload udev rules:

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

5. Restart adb with normal user:

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

The next you can invoke adb directly, for example:

```
adb shell
```

11.2 Frequently Used Adb Commands

11.2.1 Connection Management

List all connected devices and their serial numbers:

```
adb devices
```

If there are multiple connected devices, you need to use the serial number to distinguish them:

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

Also adb can be connected via the tcp/ip network.

```
adb tcpip 5555
```

Adb will restart on the device side and listen on TCP port 5555. The USB cable can be disconnected from now on.

If the IP address of the device is 192.168.1.100, you the following command to connect:

```
adb connect 192.168.1.100:5555
```

Once connected, you can run adb command as usual:

```
adb shell ps
adb logcat
```

Until you explictly disconnect adb:

```
adb disconnect 192.168.1.100:5555
```

11.2.2 Debug

Getting System Log

Usage:

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

For example:

```
# View all logs
adb logcat

# View only part of the log
adb logcat -s WifiStateMachine StateMachine
```

Gathering Bug Report

`adb bugreport` is used for error reporting, which gathers useful system information.

```
adb bugreport

# Save to local, make it easy to use editor view
adb bugreport >bugreport.txt
```

11.2.3 Running shell

Open an interactive shell:

```
adb shell
```

Run shell command:

```
adb shell ps
```

11.2.4 Apk Management

Install Apk

```
adb install [option] example.apk

options:
-l forward-lock
-r Reinstall the application to retain the original data
-s Install to SD card instead of internal storage
```

For example:

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

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

If install fails, check the common reasons below:

- `INSTALL_FAILED_ALREADY_EXISTS`: Try to add `-r` parameter to install again.
- `INSTALL_FAILED_SIGNATURE_ERROR`: APK signature is inconsistent, and it may be due to the different version of the signature and debug version. If you confirm the APK file signature is normal, you can use the adb uninstall command to uninstall the old application, and then install again.
- `INSTALL_FAILED_INSUFFICIENT_STORAGE`: There is not enough storage space.

Uninstall Apk

```
adb uninstall apk_name
```

For example:

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

The name of the apk package can be listed with the following command:

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

The apk file path and package name are separated by =.

CHAPTER 12

FAQ

12.1 How to write MAC address?

You can change ROC-RK3328-CC MAC address by yourself. Please enter RockUSB mode, then write MAC address with WNpctool.

12.2 No sound in the headset

In Ubuntu system, run PluseAudio Volume Control in Multimedia menu. Then in Configuration, select the working sound card and close others.

CHAPTER 13

Firmware and Tools

- Firmware Download Page

Firmware Flashing Tools:

- To flash to the SD card:
  - GUI:
    - SDCard Installer (Linux/Windows/Mac)
    - Etcher (Linux/Windows/Mac)
  - CLI:
    - dd (Linux)
- To flash to the eMMC:
  - GUI:
    - AndroidTool (Windows)
  - CLI:
    - upgrade_tool (Linux)
    - rkdeveloptool (Linux)

CHAPTER 14

Documents and Reference

- Partition and Storage Map

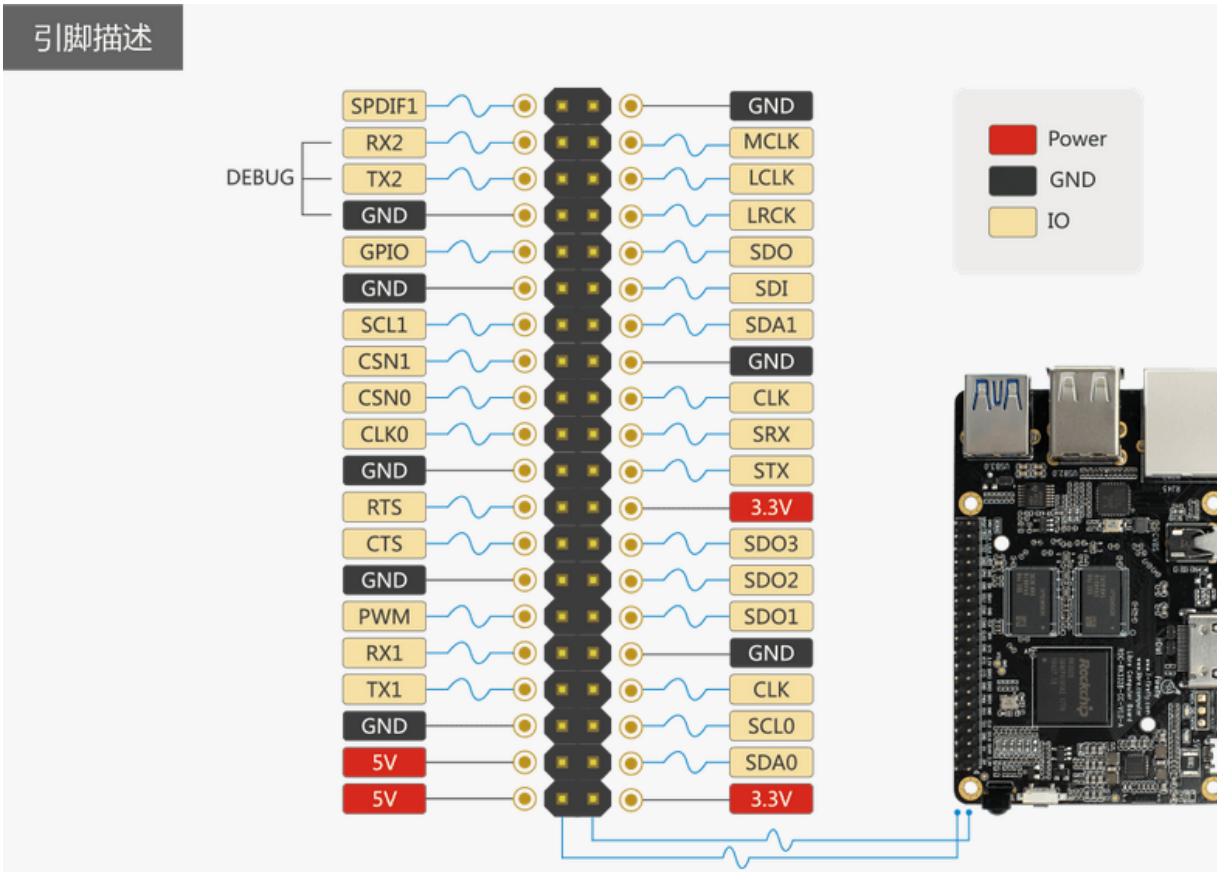
CHAPTER 15

Hardware Datasheets and Interfaces

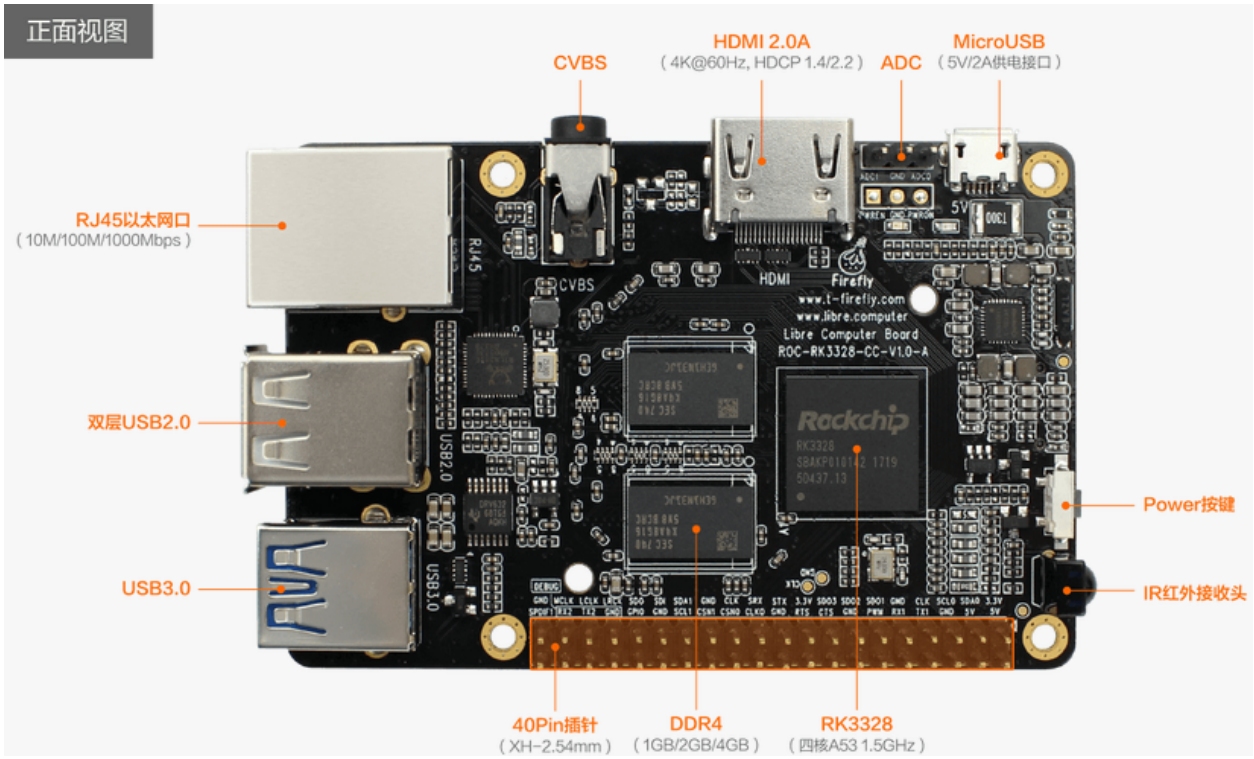
Datasheets:

- ROC-RK3328-CC Schematic
- ROC-RK3328-CC Components Position Reference
- Rockchip RK3328 Datasheet
- Rockchip RK805 Datasheet V1.1

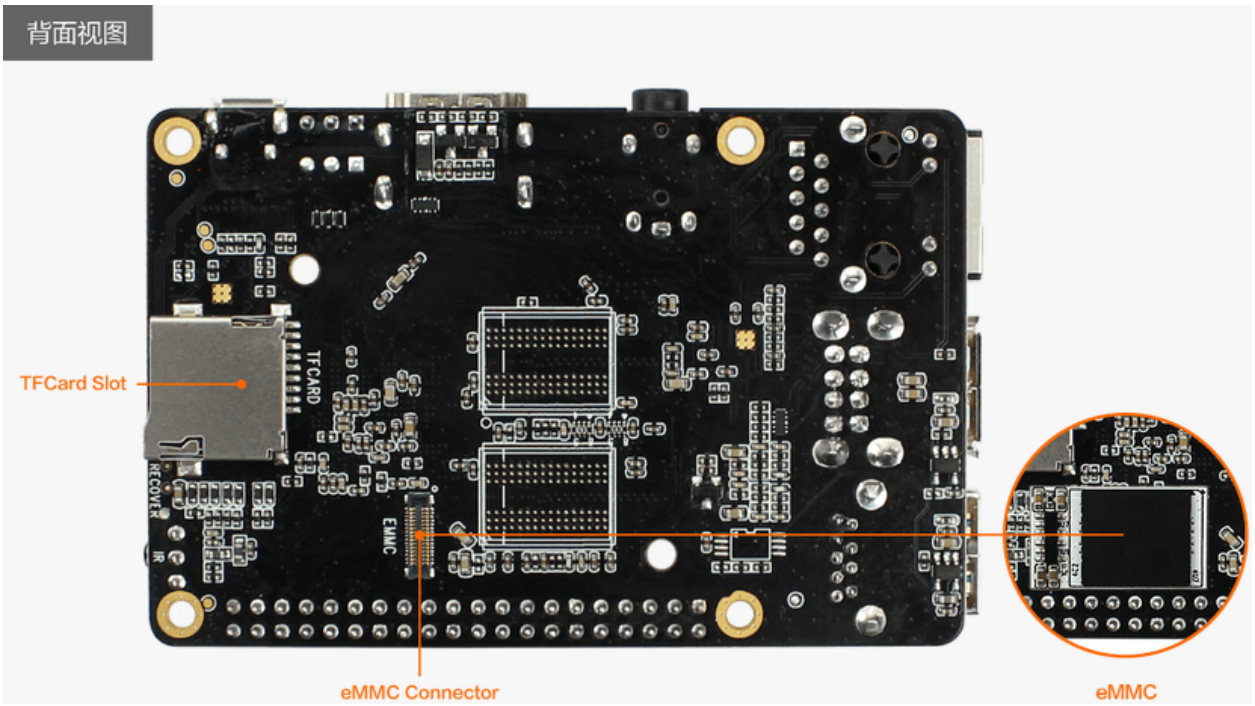
Expansion Interface:



Board Top:



Board Bottom:



CHAPTER 16

Community

Firefly:

- Forum
- Facebook
- Google+
- Youtube
- Twitter
- Shop