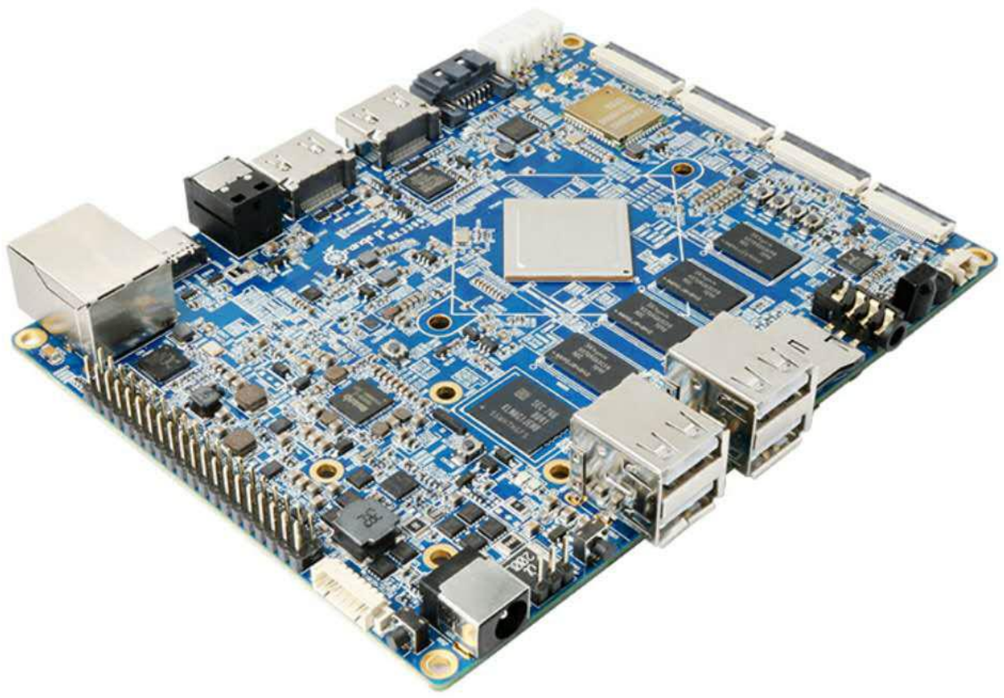
Orange Pi RK3399 User Manual



History

Ver	Data	Author	Brief	Publish	Memo
1.1	2017-07-14	Younix	Create Document	2018-1-31	
1.2	2018-02-01	Younix	Update Compilation of Android SDK in chapter Ⅲ(3)	2018-3-6	
1.3	2018-03-19	Younix	Update Frequency FAQⅤ（3）	2018-3-19	

Contents

I. Orange Pi RK3399 Introduction

1. What is Orange Pi RK3399?

It's an open-source single-board computer. It can run Android 6.0, Ubuntu, Debian, it uses the RK3399 SoC, and has 2GB DDR3 SDRAM

2. What can I do with Orange Pi RK3399?

You can use it to build…

- A computer
- A wireless server
- Games
- Music and sounds
- HD video
- A speaker
- Android
- Scratch

……

Pretty much anything else, because Orange Pi RK3399 is open source.

3. Whom is it for?

Orange Pi RK3399is for anyone who wants to create with technology– not just consuming. It's a simple, fun, useful tool and you can use it to take control of the world around you.

4. Orange Pi RK3399 Hardware specification

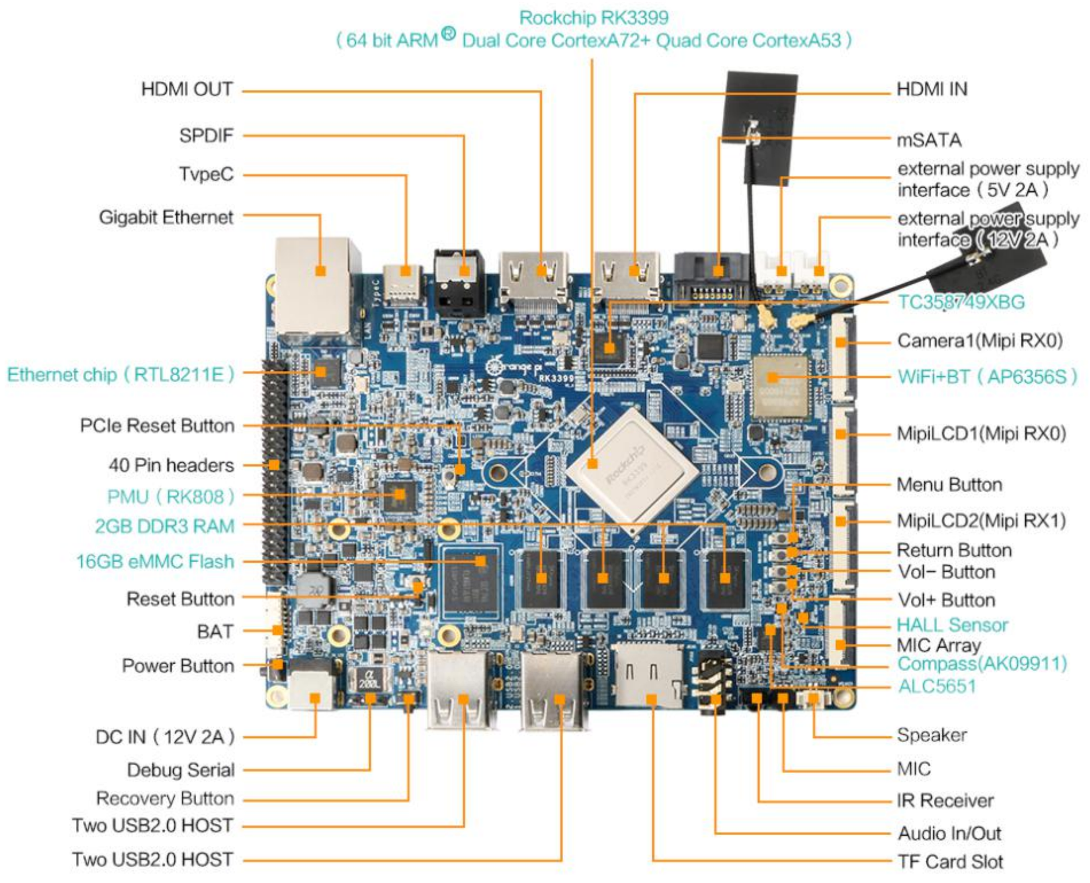
Hardware Specification	
Soc	Rockchip RK3399 (28nm HKMG Process）
CPU	Six-CoreARM® 64-bit processor, up to 2.0GHz frequency

	Dual-Core Cortex-A72 and Quad-Core Cortex-A53
GPU	ARM Mali-T860 MP4 Quad-Core GPU Completely compatible with OpenGL ES1.1/2.0/3.0/3.1, OpenVG1.1, OpenCL, DX11 Support AFBC
VPU	Supports multi-format video decoders including H.264/H.265/VP9 4Kx2K@60fps 1080P Multi format video decoding (WMV, MPEG-1/2/4, VP8) 1080PVideo encoding, supporting H.264, VP8 format Video post processor: anti interleaving, denoising, edge/ detail / color optimization
PMU	RK808 PMU BQ25700 Charger IC CW2015 Fuel Gas
Memory	2GB DDR3
Storage	16GB High-Speed eMMC MicroSD (TF) Card Slot miniPCIe (for LTE / mSATA) mSATA interface
Wireless	Integrated WiFi Combo Module(AP6356S): 2.4GHz/5GHz Dual-Band WiFi Support 802.11a/b/g/n/ac 2x2 MIMO standard Bluetooth 4.1(Support BLE)
Ethernet	10/100/1000Mbps Ethernet (Realtek RTL8211E)
Display	1 x HDMI 2.0 (Type-A), Support maximum 4K@60Hz display 1 x DP 1.2 (DisplayPort) , Support maximum 4K@60Hz display 2 x MIPI , support 2560x1600@60fps output with dual channel 1 x eDP 1.3 (4 lanes with 10.8Gbps) 1 x HDMI IN
Audio	1 x HDMI or 1 x DP (DispalyPort) for audio output 1 x Analog audio (via 3.5mm Combo Audio Jack for

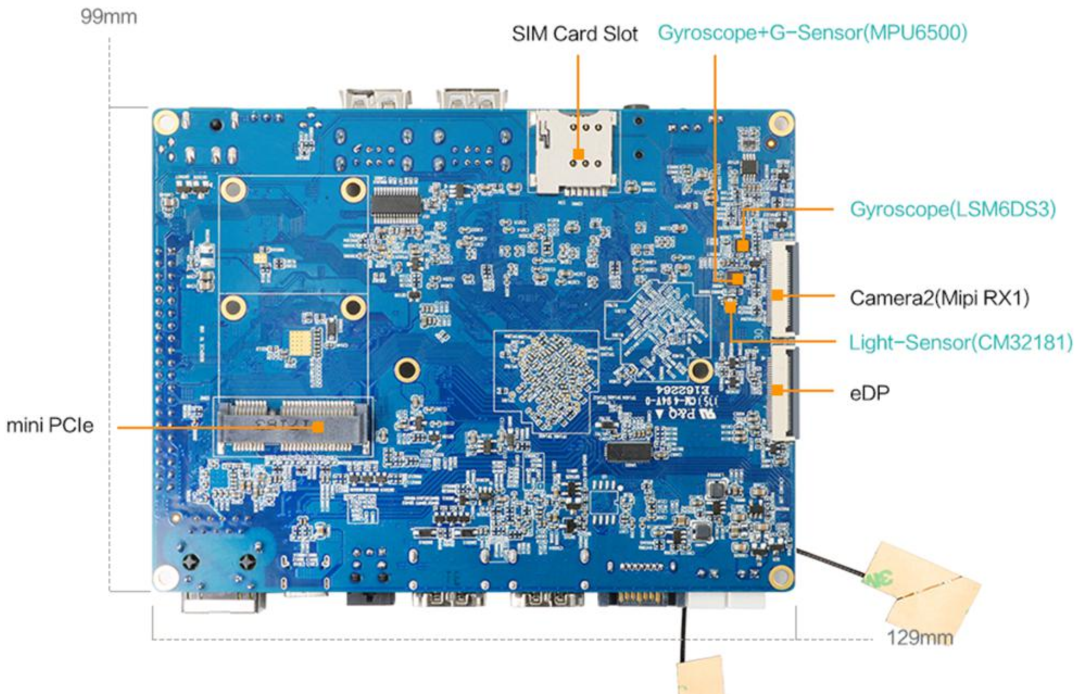
	audio input and ouput) 1 x Speaker for audio output (1.5W 8Ω or 2.5W 4Ω) 1 x SPDIF 1 x On-board Microphone 1 x I2S (up to 8 channels) for audio input and output 1 x Mic Array Interface
Camera	2 x MIPI-CSI (13Mpixel Max for each port) (OV13850(13M)) Support USB Camera
Sensor	1 x Gyroscope+G-Sensor(MPU6500) 1 x Gyroscope(LSM6DS3) 1 x HALL Sensor(HAL248TWCL) 1 x Light Sensor(CM32181) 1 x Compass(AK09911)
PCIe	1 x Mini PCIe Compatible USB, used for LTE or TF Card Compatible mSATA, used for expand SATA harddisk or SSD
SIM	1 x SIM slot, use as LTE module for miniPCIe extension
USB	4 x USB2.0 HOST, 1 x USB3.0 Type-C
IR	1 x IR, Support IR control function
LED	2 x Power Status LED (Red and Green) 1 x SATA Power Status LED(Green)
Key	1x Reset Button, 1 x Power Button, 1 x Recovery Button, 1 x Menu Button, 1 x Return Button, 1 x Vol+ Button, 1 x Vol- Button
Debugging	1 x Serial Console
Reserved Interface	40pin 2.54mm header 4 x I2C , 1 x SPI, 2 x UART, 5 x GPIO
External Power supply interface	DC12V - 2A (2 pins) DC5V - 2A (2 pins)
Power	DC12V-2A (via DC 5.5*2.1mm Jack) TypeC 5V-3A Battery (Dual Battery 7.4V)

OS / Software	
OS	Android 6.0, Debian 9
Programming support	C, C++, Kotlin, Java, Shell, Pyhon
Interface definition	
Size	129 mm × 99 mm
Weight	100g

Top view

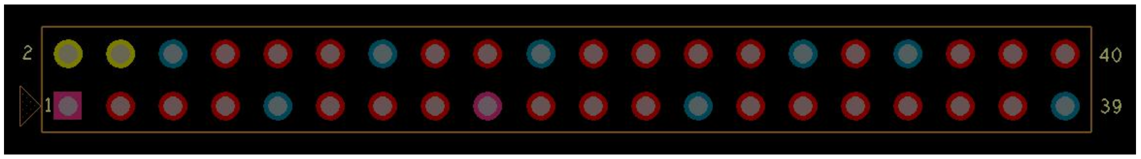


Bottm view



5. GPIO Specifications

The following is GPIO Pin of Orange Pi RK3399:



PIN1	3V3-1	Power 3.3V			
PIN2	5V-1	Power 5V			
PIN3	SDA	GPIO1_B3	I2C4_SDA		Default used as I2C
PIN4	5V-2	Power 5V			
PIN5	SCL	GPIO1_B4	I2C4_SCL		Default used as I2C
PIN6	GND4	Ground			
PIN7	GPIO4	GPIO2_A0	I2C2_SDA		

PIN8	TX	GPIO4_C4	UART2DBG_TX		Default used as Debug
PIN9	GND1	Ground			
PIN10	RX	GPIO4_C3	UART2DBG_RX		Default used as Debug
PIN11	GPIO17	GPIO2_C0	UART0_RXD		
PIN12	GPIO18	GPIO2_A1	I2C2_SCL		
PIN13	GPIO27	GPIO2_C1	UART0_TXD		
PIN14	GND5	Ground			
PIN15	GPIO22	GPIO2_C2	UART0_CTS		
PIN16	GPIO23	GPIO2_A2			
PIN17	3V3-2	Power 3.3V			
PIN18	GPIO24	GPIO2_A3			
PIN19	MOSI	GPIO1_A7	SPI1_RXD		
PIN20	GND6	Ground			
PIN21	MISO	GPIO1_B0	SPI1_TXD		
PIN22	GPIO25	GPIO2_C3	UART0_RTS		
PIN23	SCLK	GPIO1_B1	SPI1_CLK		
PIN24	CS0	GPIO1_B2			
PIN25	GND2	Ground			
PIN26	CS1	GPIO2_D4			When use J90005 to connect Camera, this pin has been occupied, DVP_PDN0_H
PIN27	DNP1	GPIO4_D2			
PIN28	DNP2	GPIO1_C2			
PIN29	GPIO5	GPIO2_A4			Default used as PCIE_PERST
PIN30	GND7	Ground			
PIN31	GPIO6	GPIO2_A5			Default used as HDMIIN_PWREN33
PIN32	GPIO12	GPIO2_B4	SPI2_CSN		When use J4601 to connect Camera, this pin has been occupied, DVP_PDN0_H

PIN33	GPIO13	GPIO2_A6			Default used as HDMIIN_PWREN
PIN34	GND8	Ground			
PIN35	GPIO19	GPIO2_A7	I2C7_SDA		
PIN36	GPIO16	GPIO2_B1	SPI2_RXD	I2C6_SDA	Default used as HDMIIN_PWREN18
PIN37	GPIO26	GPIO2_B0	I2C7_SCL		
PIN38	GPIO20	GPIO2_B2	SPI2_TXD	I2C6_SCL	
PIN39	GND3	Ground			
PIN40	GPIO21	GPIO2_B3	SPI2_CLK		

II. Using Method Introduction

1. Hardware Requirement

- Orange Pi RK3399 Development Board
- A PC for compilation with following specs:

  64bit CPU

  Up to 16GB RAM

  UP to 40GB spare disk space

  Operation system should up to Ubuntu12.04, it would be better if it is Ubuntu16.04

You could refer to Google file for more details: https://source.android.com/source/building

2. Software Requirement

- Orange Pi RK3399 SDK
- Orange Pi RK3399 Firmware
- Android-image-flash-tool

3. Power Supply Requirement

There are three methods for power supply:

- DC （12V 2A） in for power
- TypeC （5V 3A） in for power
- Battery （Dual Battery 7.4V） in for power, connect the battery with our BAT interface

If insert TypeC and DC connector in the same time, the system would default charge by DC in. It would not recommend to power by the battery, because different batteries need matching with BQ25700 battery management IC according to their respective parameters.

III. Android Compilation Environment Construction

1. Download SDK compression package

Take OrangePi-RK3399_Android6.0_V1.0_2017_0720.tgz as am example, after get the original compression package:

```
mkdir OrangePi-rk3399
tar xvf OrangePi-rk3399_Android6.0_V1.0_2017_0720.tgz -C OrangePi-rk3399
cd OrangePi-rk3399
```

2. Construct Compilation Environment

It could also refer to Google file: http://source.android.com/source/initializing.html

● Install JDK

Compilation of Android6.0 is based on JAVA7, it needs to first install OpenJDK before compilation.

Command for installing:

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

Configure environment variable of JAVA, here is the path for installation:

/usr/lib/jvm/java-7-openjdk-amd64

It could configure on the terminal with the following command:

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

● Install Software Package

For Ubuntu12.04:

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

For Ubuntu14.04:

```
sudo apt-get update
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
```

The relevant software package for installing ARM cross compilation tool chain and kernel:

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

3. Compilation of SDK Source Code

● Compilation with auto-building shell scripts

We can make use of the RKTool/make.sh script in the SDK root directory for automatic compilation, using the following methods(Please ensure that it runs in the root directory):

U-boot Compilation:

```
./RKTools/make.sh -u -j4
```

Kernel Compilation:

```
./RKTools/make.sh -k -j4
```

Android Compilation:

```
./RKTools/make.sh -a -j4
```

Compile u-boot、Kernel、Android in the same time:

```
cd SDK_ROOT/
./RKTools/make.sh -j4
```

- **Manual Compilation with Different Module**

U-boot Compilation:

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

Kernel Compilation:

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

Android Compilation:

```
source build/envsetup.sh
lunch rk3399_mid-userdebug
make -j4
```

4. Generated Firmware

```
./mkimages.sh
```

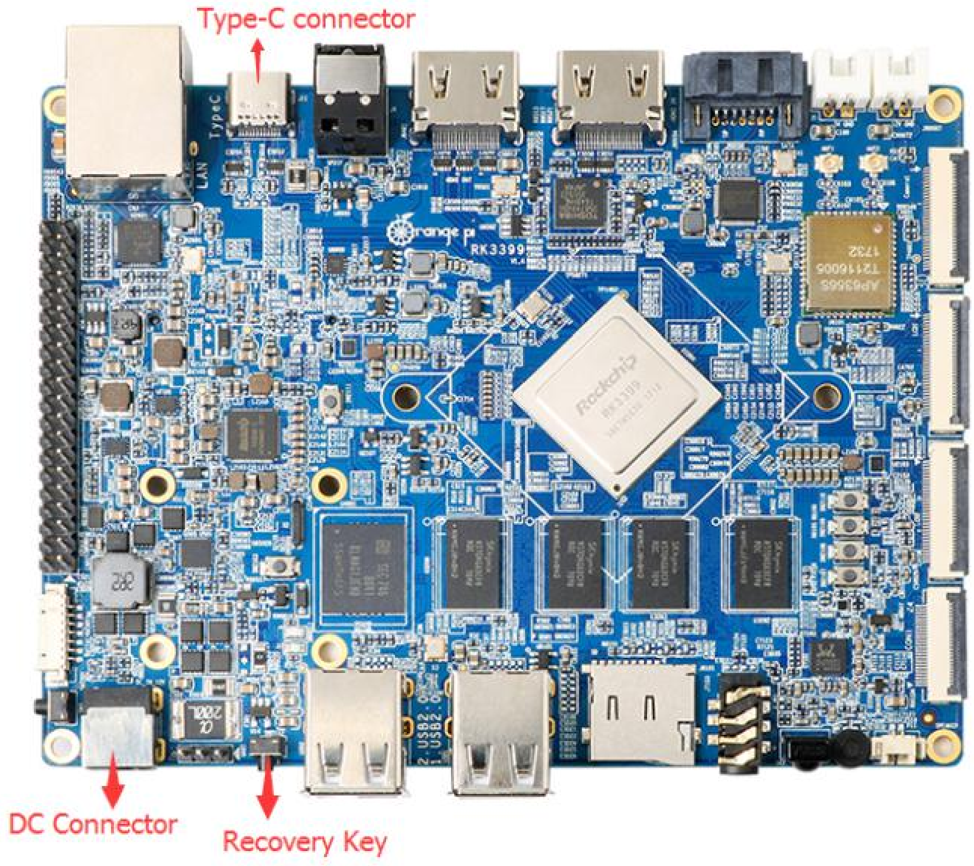
After execute ./mkimages.sh, it will generate a full firmware package on the directory of rockdev/Image-rk3399_mid.

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

Except the above method, the unity image(update.img) could also be generated via Linux_Pack_Firmware.

IV. Android Firmware Flashing

Relevant keys and connectors for firmware flashing of **Orange Pi RK3399 :**



There are two types of Firmware file:

- Multi-partition images: generated uboot.img, recovery.img, trust.img, kernel.img, resource.img, system.img, usually used for debug.
- One image: generated into update.img with packing tool from several partition image files, usually used for Firmware release.

(The official has already made Android image, also you could try to compile you own image with reference of our manual.)

Supporting OS of PC:

- Windows XP (32/64bit)
- Windows 7 (32/64bit)
- Windows 8 (32/64bit)
- Linux (32/64bit)

We use AndroidTool on Windows

Download path: AndroidTool

We use upgrade_tool on Linux:

Download path: upgrade_tool

Please select the corresponding tool according to your PC environment.。

1. Flashing image on Windows

The tool we should use on Windows is AndroidTool, you could use it for multi-partition image flashing or one image file: update.img.

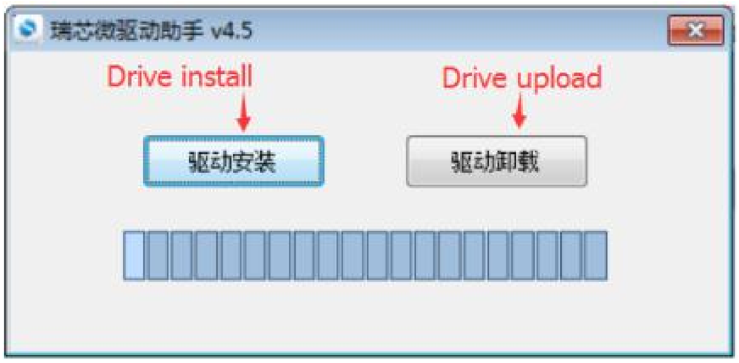
Before image writing, we need to first install RK USB driver on Windows.

1) Install RK USB Driver

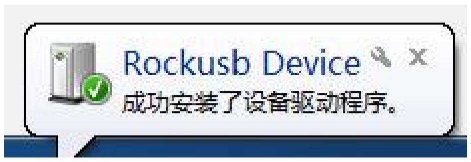
Download path: DriverAssitant

Run this after unzip: DriverInstall.exe

In order to use the latest driver of all device, please first click Drive unload and then click Drive install:

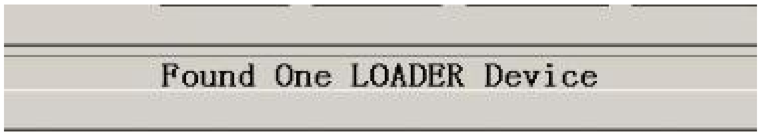


Connect with Type-C data cable and PC with Orange Pi RK3399 after installed USB driver, there would be show the status of USB driver on lower right corner like the following:

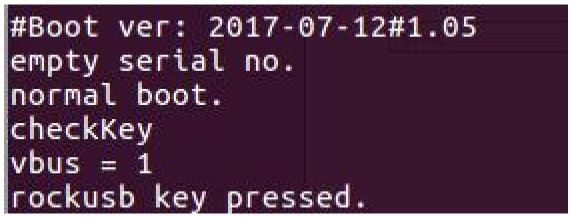

Rockusb Device
成功安装了设备驱动程序。

2) Enter into Flashing Mode

- Type-A Connect to PC
- Press on Recovery key of Orange Pi RK3399
- Type-C Connect to Orange Pi RK3399, there should be notice on the following :


Found One LOADER Device

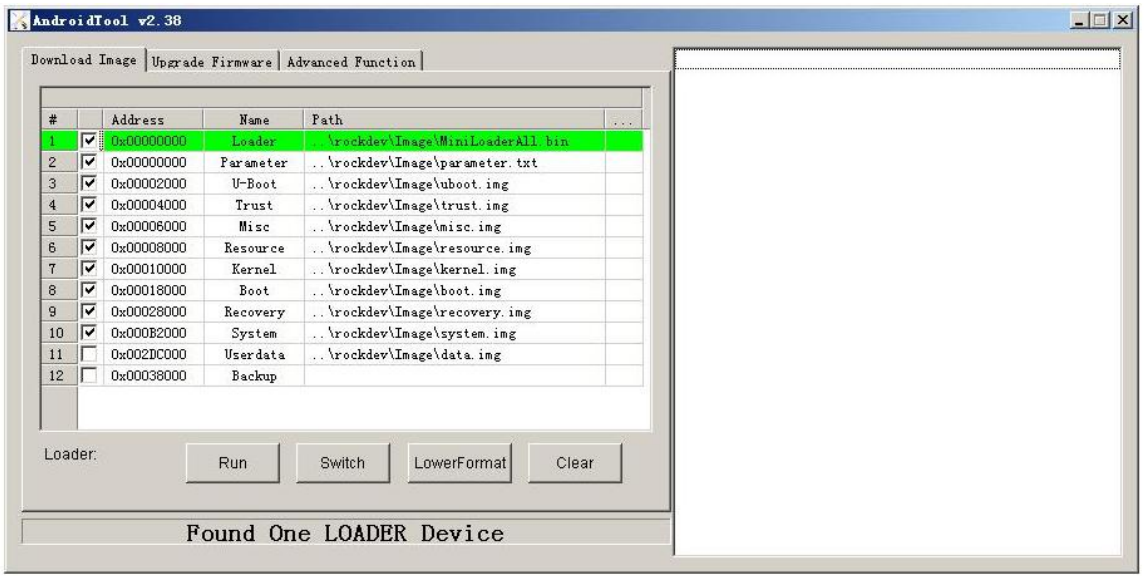
There would be log information if you connect debug pin.


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

- Insert DC power supply

Since OrangePi RK3399 supports powered by TypeC, power voltage on USB of PC is enough for flashing image. However, it may not enough to support Orange Pi booting. In that case, we need to insert DC power supply to make sure the Orange Pi could boot successfully after flashed image.

- If need to flash every partition image(***.img**) separately, right click Download Image then click corresponding firmware path. After that click Run, and the right side would show the status of running.


AndroidTool v2.38

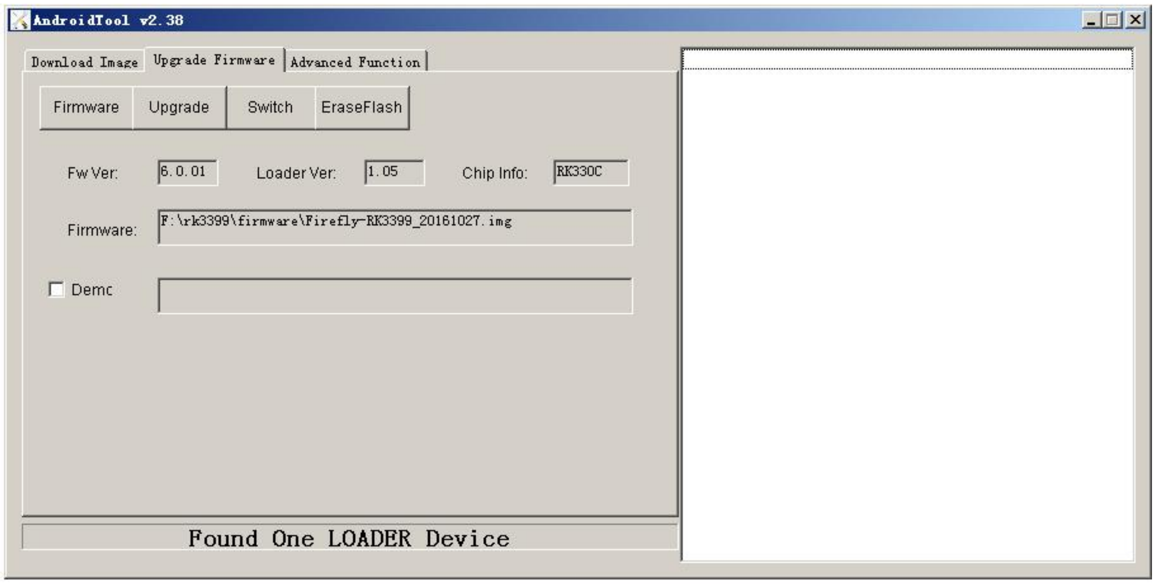
Download Image | Upgrade Firmware | Advanced Function

#		Address	Name	Path	...
1	✓	0x00000000	Loader	..\rockdev\Image\MiniLoaderAll.bin	
2	✓	0x00000000	Parameter	..\rockdev\Image\parameter.txt	
3	✓	0x00002000	U-Boot	..\rockdev\Image\uboot.img	
4	✓	0x00004000	Trust	..\rockdev\Image\trust.img	
5	✓	0x00006000	Misc	..\rockdev\Image\misc.img	
6	✓	0x00008000	Resource	..\rockdev\Image\resource.img	
7	✓	0x00010000	Kernel	..\rockdev\Image\kernel.img	
8	✓	0x00018000	Boot	..\rockdev\Image\boot.img	
9	✓	0x00028000	Recovery	..\rockdev\Image\recovery.img	
10	✓	0x000B2000	System	..\rockdev\Image\system.img	
11		0x002DC000	Userdata	..\rockdev\Image\data.img	
12		0x00038000	Backup		

Loader:

Found One LOADER Device

- If need to flash unity firmware(update.img), click Firmware on Upgrade Firmware, select path of update.img. After it recognized LOADER device, click Upgrade and it will begin to upgrade. The right side would show the status of flashing.


AndroidTool v2.38

Download Image | Upgrade Firmware | Advanced Function

Fw Ver: 6.0.01 Loader Ver: 1.05 Chip Info: RK330C

Firmware: F:\rk3399\firmware\Firefly-RK3399_20161027.img

☐ Demc

Found One LOADER Device

2. Flashing image on Linux

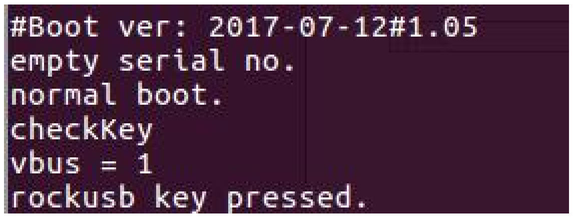
1) Flash tool: upgrade_tool

On Linux, the tool we use is upgrade_tool which is same with working on Windows. Enter into Loader flashing mode like the following:

- Type-A Connect to PC
- Press on Recovery key of Orange Pi RK3399
- Type-C Connect to Orange Pi RK3399
- Insert DC power supply

Since OrangePi RK3399 supports powered by TypeC, power voltage on USB of PC is enough for flashing image. However, it may not enough to support Orange Pi booting. In that case, we need to insert DC power supply to make sure the Orange Pi could boot successfully after flashed image.

There would be log information if you connect debug pin.



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

Run upgrade_tool on terminal of Linux:

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

Enter R: Re-scan the USB port to find the device

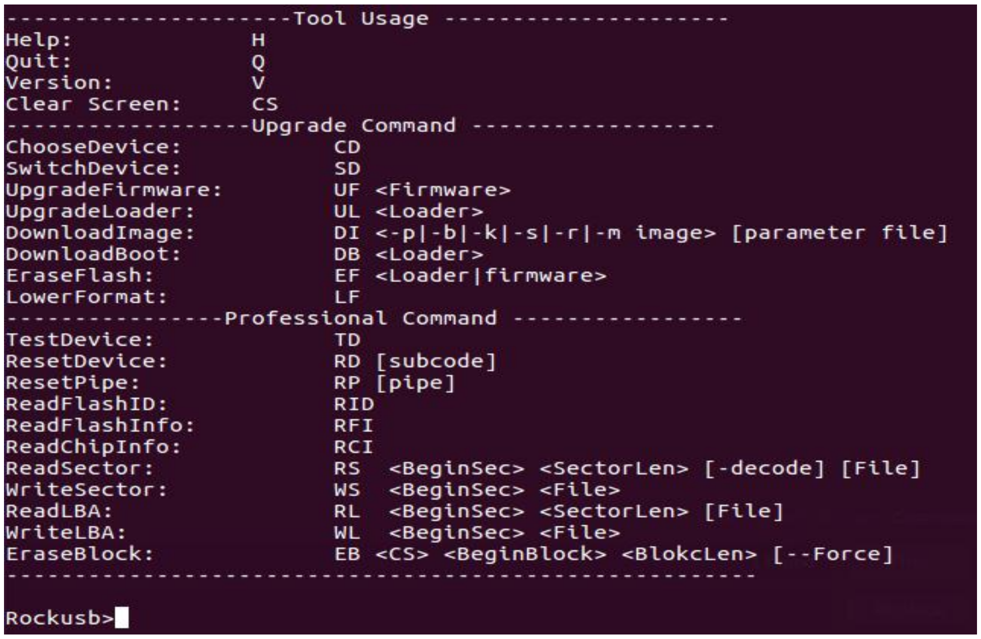
Enter Q: Exit flashing tool

Enter DevNo: Select the corresponding operation device

Here I would enter 1:

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

The help menu is displayed when you enter, and the Rockusb> prompt appears

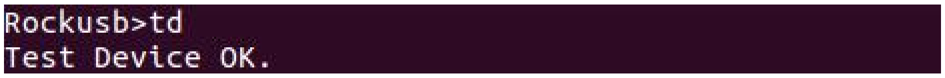


```
---------------------Tool Usage --------------------
Help:              H
Quit:              Q
Version:           V
Clear Screen:      CS
------------------Upgrade Command -----------------
ChooseDevice:            CD
SwitchDevice:            SD
UpgradeFirmware:         UF <Firmware>
UpgradeLoader: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

Rockusb>
```

- It could be operated by entering the corresponding instructions after the Rockusb> prompt. No distinguish between capital and lowercase.

TD Command: used to test whether the device status is normal



```
Rockusb>td
Test Device OK.
```

DI Command: used for flashing separate partitions *.img ：

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

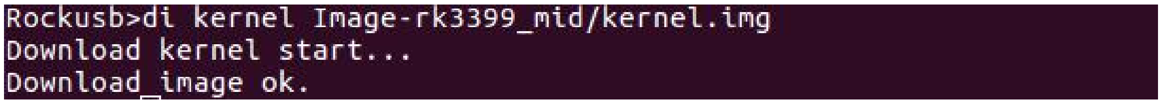
The first parameter is used to specify the partition name that needs to be flashed.

The second parameter is used to specify the path of the flashed image

For example, there would be two method to flash kernel.img:

Rockusb>di -k ./kernel.img

Rockusb>di kernel ./kernel.img



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

UF Command: used for unity flashed image update.img

UF <Firmware>

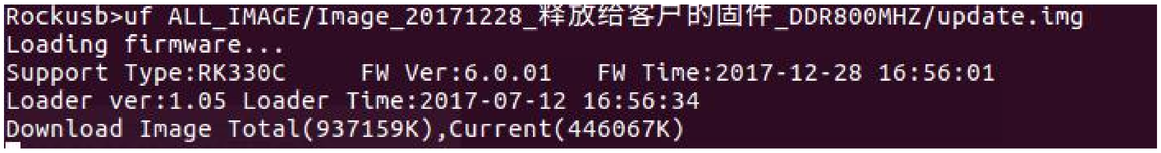
The only parameter is used to specify the need to burn the firmware path.

For example, if my firmware path is:

RK3399_IMAGE/Image_Android6.0_20171228.img

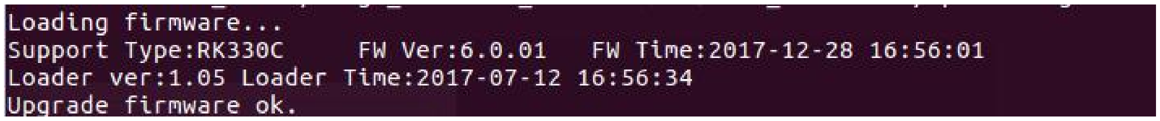
Then the command should be:

Rockusb>uf RK3399_IMAGE/Image_Android6.0_20171228.img



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

After flashed, there would show the following information and the Orange Pi would reboot. (UF command would reboot, but not DI command.)



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

2) Write Scripts to Implement User-Defined Flash

upgrade_tool also supports used as Linux command for flashing, you only need to add the path of tool into environment variable.

For example, when debug Kernel, if you want to make it realize modify-compile-flash, you could also try the following:

```
# Compile part of the firmware of kernel it will generated kernel.img and resource.img
make -j2 rk3399-orangepi.img
# enter into Loader mode with adb command
adb shell rebot bootloader
# finished flashing with di Command
sudo upgrade_tool di resource resource.img
sudo upgrade_tool di kernel kernel.img
# reboot with rd Command
sudo upgrade_tool rd
```

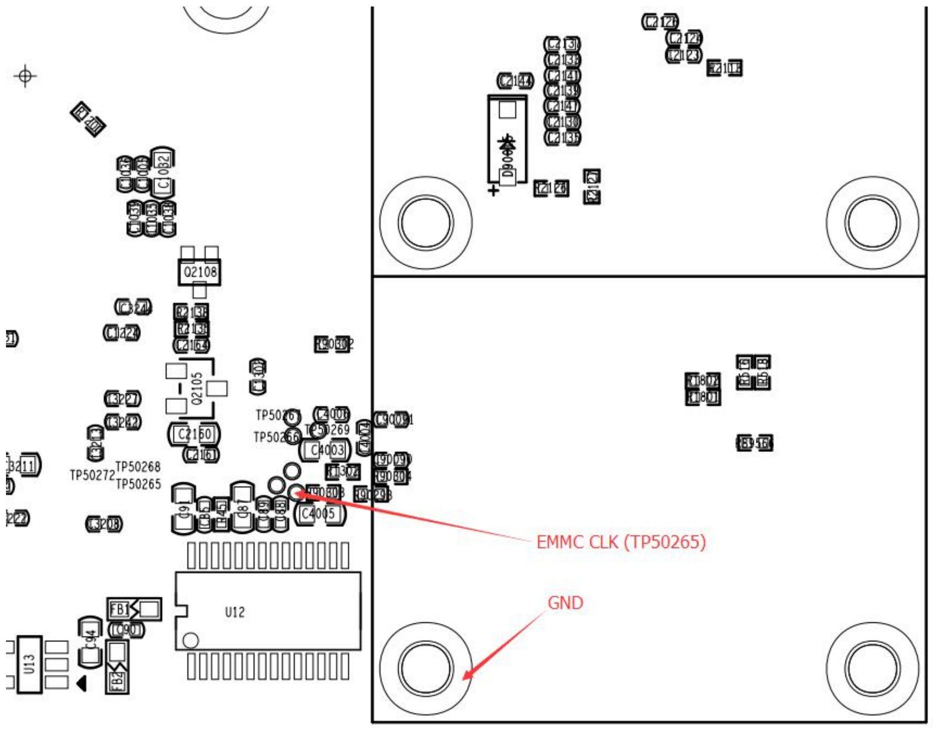
3. FAQ of Image Flashing

● Cannot power on because of error firmware

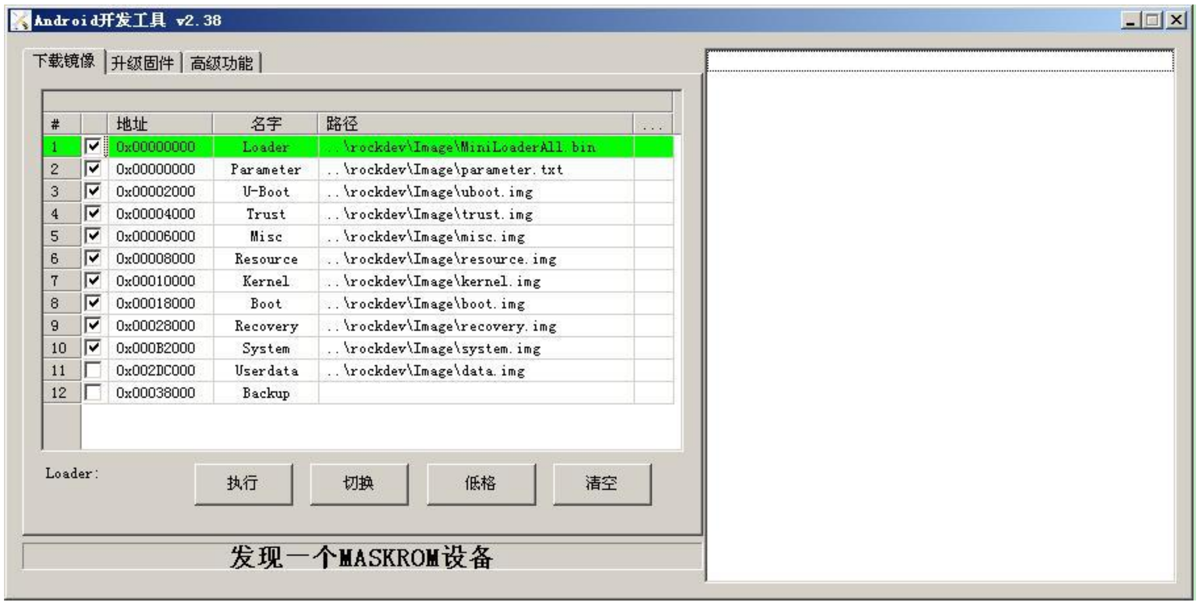
Usually we could enter into LOADER mode to flashing hardware, however, if it cannot be powered on because of error firmware, we could try to use MASKROM mode to write image and boot.

1. Power off the device
2. Use metal tweezers to keep TP50265 and GND connected
3. Power on the board with DC

4. Wait a moment then release the metal tweezers
5. Use a micro USB Type-C cable to connect device and host PC
6. Device should enter MASKROM mode



It would show the following with AndroidTools on Windows:



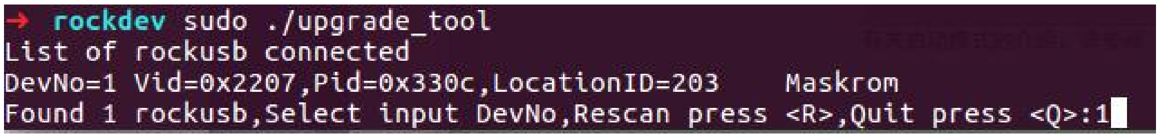
#		地址	名字	路径
1	✓	0x00000000	Loader	..\rockdev\Image\MiniLoaderAll.bin
2	✓	0x00000000	Parameter	..\rockdev\Image\parameter.txt
3	✓	0x00002000	U-Boot	..\rockdev\Image\uboot.img
4	✓	0x00004000	Trust	..\rockdev\Image\trust.img
5	✓	0x00006000	Misc	..\rockdev\Image\misc.img
6	✓	0x00008000	Resource	..\rockdev\Image\resource.img
7	✓	0x00010000	Kernel	..\rockdev\Image\kernel.img
8	✓	0x00018000	Boot	..\rockdev\Image\boot.img
9	✓	0x00028000	Recovery	..\rockdev\Image\recovery.img
10	✓	0x000B2000	System	..\rockdev\Image\system.img
11		0x002DC000	Userdata	..\rockdev\Image\data.img
12		0x00038000	Backup	

Loader:

发现一个MASKROM设备

After this could flash the image in the normal way.

It would show the following if use upgrade_tool on Linux:



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

It would be re-flash the image with uf command.

Principle:

It would clean data on flash if short connect pin and GND because the system would consider data error of flash.

V. Linux Environment Construction and Firmware Compilation

1. Download SDK compression package

Take OrangePi_Android6.0_V1.0_2017_0720.tgz as an example, after get the original compression package:

```
mkdir OrangePi-rk3399-Linux
tar xvf OrangePi_Linux_V1.0_2018_0110.tgz -C OrangePi-rk3399-Linux
cd OrangePi-rk3399-Linux
```

2. Construct Compilation Environment

It could also refer to Google file: http://source.android.com/source/initializing.html

3. Install Software Package

For Ubuntu16.04:

```
sudo apt-get update
sudo apt-get install git-core gitk git-gui gcc-arm-linux-gnueabihf u-boot-tools
device-tree-compiler gcc-aarch64-linux-gnu mtools parted libudev-dev libusb-1.0-0-dev
libssl-dev pv e2fsprogs build-essential fakeroot devscripts
```

If your development environment is not Ubuntu16.04, please make sure the compile tool train version of gcc-arm-linux-gnueabihf and gcc-aarch64-linux-gnu is less than 6.0.

4. Compile Linux SDK and Make Firmware

It is defaulted compilation under userdebug mode.

U-boot Compilation:

```
./build/mk-u-boot.sh rk3399-orangepi
```

There will be generated the following file on out/u-boot directory:

```
➜  tree ./out/u-boot
u-boot
├── idbloader.img
├── rk3399_loader_v1.08.106.bin
├── trust.img
└── uboot.img
```

Kernel Compilation:

```
./build/mk-kernel.sh rk3399-orangepi
```

There will be generated the following file on out directory

```
➜  tree ./out/kernel
├── boot.img
└── kernel
    ├── Image
    └── rk3399-orangepi.dtb
```

Rootfs Compilation:

You could use different Rootfs, what have already compiled just like the following which could be used directory:

Ubuntu16.04 Desktop version: ubuntu-desktop.img

Ubuntu16.04 Server version: ubuntu-server.img

Ubuntu16.04 LXDE version: ubuntu-lxde.img

Debian9 Desktop version: debian-desktop.img

You could also make your own Rootfs with reference of OrangePi RK3399 Rootfs prepare.

Pack every partitions' image into a unity full firmware:

```
./build/mk-image.sh -c rk3399 -t system -s 4000 -r out/ubuntu-server.img
```

c for chip, represents the chip model

t for target, represents the generated image name

s for size, represents the predistribution size(but it does not means the final real size. The firmware will Redynamic adjustment the size), unit: Mbyte

r for rootfs, represents the path used by rootfs

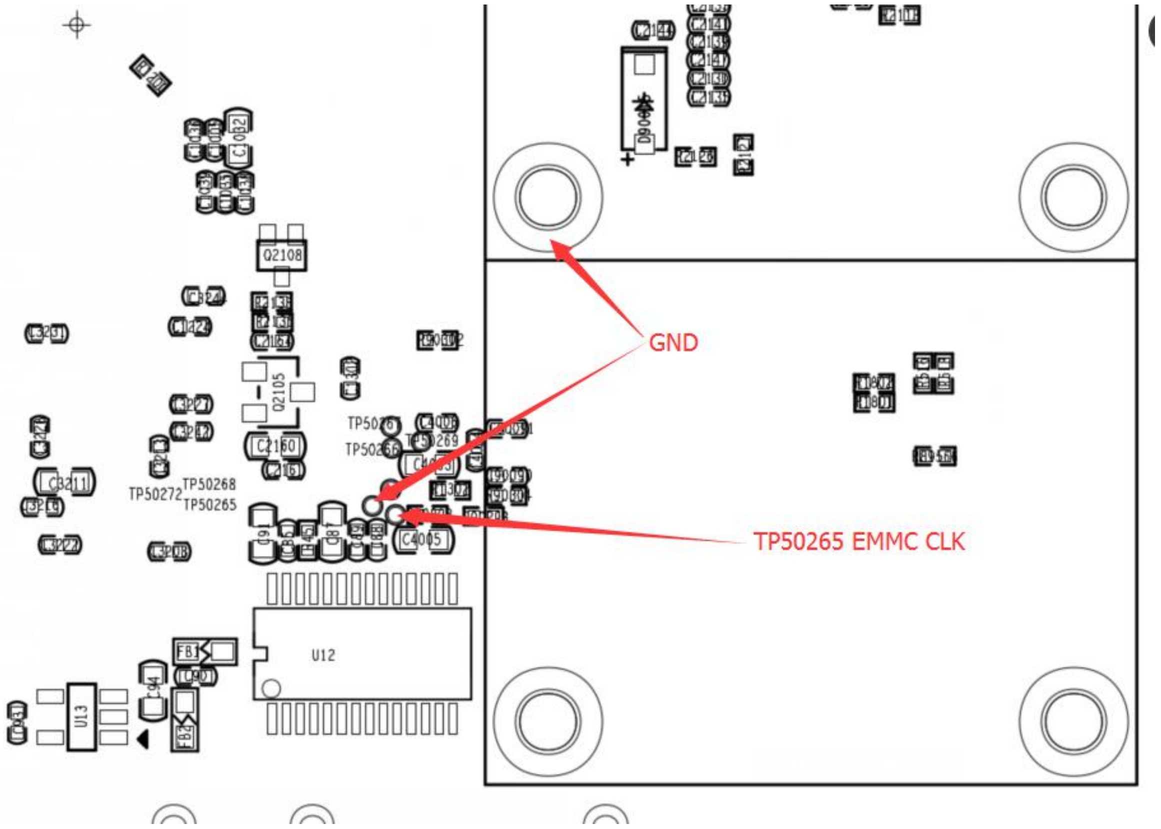
After execute the above command, there will be generate the unity full firmware(system.img) on the directory of out.

5. Flash Linux Firmware

Flash the full Linux firmware into eMMC:

1. Short circuit connecting with Clk and GND of EMMC, connect DC power supply, enter into Maskrom mode.

EMMC Clk is on behind of miniPCIe just like the following shows:



2. Connect TypeC cable
3. Execute the following command to flash

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

c for chip, represents chip model

p for partition, represents partition, such as boot loader1 system

i for image, represents path of image

You could also flash different partitions separately, for example:

Separate flashing boot:

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

Separate flashing uboot:

```
./build/flash_tool.sh -p loader1 -i out/u-boot/idbloader.img -c rk3399
./build/flash_tool.sh -p loader2 -i out/u-boot/uboot.img -c rk3399
./build/flash_tool.sh -p atf -i out/u-boot/trust.img -c rk3399
```

VI. Construct Compilation Environment and Make Rootfs Image

1. Construct Compilation Environment

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

If your development environment is not Ubuntu16.04, please make sure the gcc-arm-linux-gnueabihf and gcc-aarch64-linux-gnu compile tool train less than 6.0.

2. Download Linux Rootfs Source Code Package

Take example for Ubuntu16.04:

We could get SDK from Ubuntu cdimage:

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

Dowload ubuntu-base-16.04.1-base-arm64.tar.gz and unzip it:

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

3. Modify Rootfs and Add Customize Software

```
sudo cp -b /etc/resolv.conf rootfs/etc/resolv.conf
sudo cp /usr/bin/qemu-aarch64-static rootfs/usr/bin/
# Enter into root system
sudo chroot rootfs /bin/bash

# Update software library and install software
apt update
apt upgrade
# according to you installation need
apt install build-essential vim ping ssh, etc.
```

```
# Install desktop version, it would take a little long time, please keep the network smooth.
# It would be Server version if do not execute
apt install ubuntu-desktop

# Add user and set password
useradd -s '/bin/bash' -m -G adm,sudo orangepi
# Set password for user orangepi
passwd orangepi
# Set password for rootpasswd root

# exit Rootfs
exit
```

4. Make Rootfs Image

```
# Generate spare image file
dd if=/dev/zero of=ubuntu-desktop.img bs=1M count=2048

# Format image file into ext4 format
sudo   mkfs.ext4   ubuntu-desktop.img

# Load image file to ubuntu-desktop folder
mkdir   ubuntu-desktop
sudo mount ubuntu-desktop.img ubuntu-desktop/

# copy the generated rootfs contents into folder which image loaded
sudo cp -rfp rootfs/*   ubuntu-desktop/

# Unmount
sudo umount ubuntu-desktop/

# Check the correctness of the file system
e2fsck -p -f ubuntu-desktop.img

# Automatically adjust the size of the partition
resize2fs   -M ubuntu-desktop.img
```