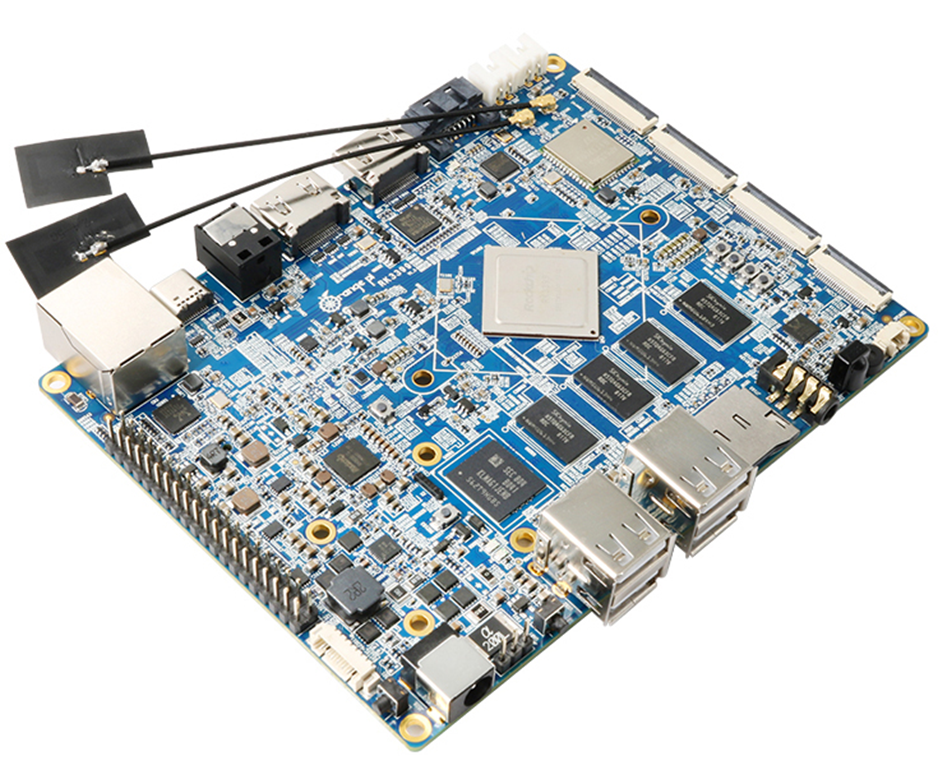
**Orange Pi RK3399  
User Manual**



**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 | |  |
| 1.4 | 2018-06-05 | pmm | Update Linux firmware flashing and Serial port debug introduction | | 2018-6-5 |  |
| 1.5 | 2018-10-30 | pmm | Update the use of HDMI IN | 2018-10-30 | |  |
| 1.6 | 2019-04-28 | baiywt | Update the Chapter V and add the Chapter VIII | 2019-05-10 | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | |  |

**Contents**

[I. Orange Pi RK3399 Introduction 1](#_Toc18793)

[1. What is Orange Pi RK3399? 1](#_Toc15560)

[2. What can I do with Orange Pi RK3399? 1](#_Toc6241)

[3. Whom is it for? 1](#_Toc10787)

[4. Orange Pi RK3399 Hardware specification 1](#_Toc22148)

[5. GPIO Specifications 5](#_Toc28475)

[II. Using Method Introduction 8](#_Toc27839)

[1． Hardware Requirement 8](#_Toc10185)

[2． Software Requirement 8](#_Toc3286)

[3． Power Supply Requirement 8](#_Toc18530)

[III. Android Compilation Environment Construction 9](#_Toc3603)

[1． Download SDK compression package 9](#_Toc27311)

[2． Construct Compilation Environment 9](#_Toc18368)

[3． Compilation of SDK Source Code 10](#_Toc9229)

[4． Generated Firmware 11](#_Toc4120)

[IV. Android Firmware Flashing 12](#_Toc30424)

[1． Flashing image on Windows 13](#_Toc30948)

[2． Flashing image on Linux 1](#_Toc28835)6

[3． FAQ of Image Flashing 2](#_Toc9074)0

[V. Linux Environment Construction and Firmware Compilation 2](#_Toc28344)3

[1． Download Linux Source Code 2](#_Toc30882)3

[2． Compile Linux Source Code 2](#_Toc28419)5

[3． Flash Linux Firmware 2](#_Toc28495)8

[VI. Construct Compilation Environment and Make Rootfs Image 2](#_Toc11391)7

[1． Construct Compilation Environment 2](#_Toc8953)7

[2． Download Linux Rootfs Source Code Package 2](#_Toc31962)7

[3． Modify Rootfs and Add Customize Software 2](#_Toc8442)7

[4． Make Rootfs Image 2](#_Toc8766)8

[VII. Using Debug tools on OrangePi 2](#_Toc3255)9

[1. Operation Steps on Windows 2](#_Toc7409)9

[2. Operation Steps on Linux 3](#_Toc28166)1

[VIII.](#_Toc3255) [Linux Using Method 41](#_Toc11391)

[1.Login Account and Password 4](#_Toc8953)[1](#_Toc3255)

[2.Extend rootfs partition 4](#_Toc31962)[1](#_Toc3255)

[3.GPU Performance Test](#_Toc8442) [41](#_Toc3255)

[4. Compile Decoding Source 42](#_Toc8766)

[5.Using HDMI IN 42](#_Toc8442)

[6. Using Ov13850 Camera 42](#_Toc8766)

[7. Using GPIO](#_Toc8766) [43](#_Toc3255)

1. **Orange Pi RK3399 Introduction**

## 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

## 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.

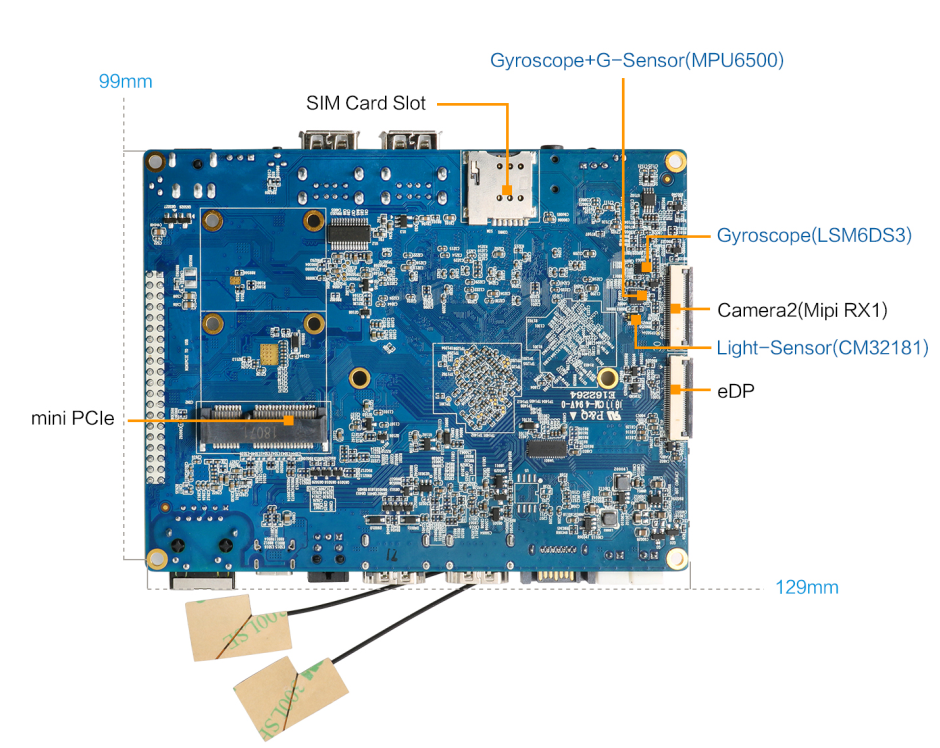
## Whom is it for?

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.

## Orange Pi RK3399 Hardware specification

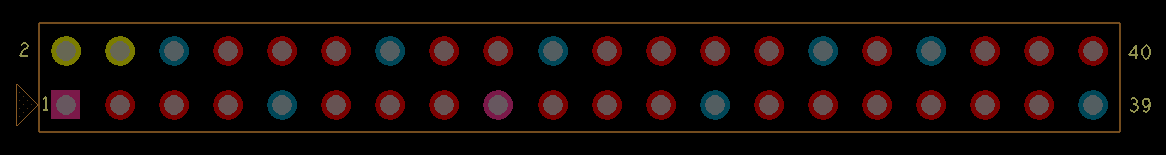
|  |  |  |
| --- | --- | --- |
| Hardware Specification | | |
| Soc | Rockchip RK3399 (28nm HKMG Process） | |
| CPU | Six-CoreARM® 64-bit processor, up to 2.0GHz frequency  Dual-Core Cortex-A72 and Quad-Core Cortex-A53 | |
| GPU | ARM Mali-T860 MP4 Quad-Core GPU  Completely compatible with OpenGL ES1.1/2.0/3.0/3.1, 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 2. x DP 1.2 (DisplayPort) , Support maximum 4K@60Hz display 3. x MIPI , support 2560x1600@60fps output with dual channel 4. 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 2. 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 | 1. 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 |





## GPIO Specifications

The following is GPIO definition 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 | |  |  |

1. **Using Method Introduction**
2. **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

1. **Software Requirement**

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

1. **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.

1. **Android** **Compilation Environment Construction**

## Download SDK compression package

Take OrangePi-RK3399\_Android6.0\_V1.0\_2017\_0720.tgz as an 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

## **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

## **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

## **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

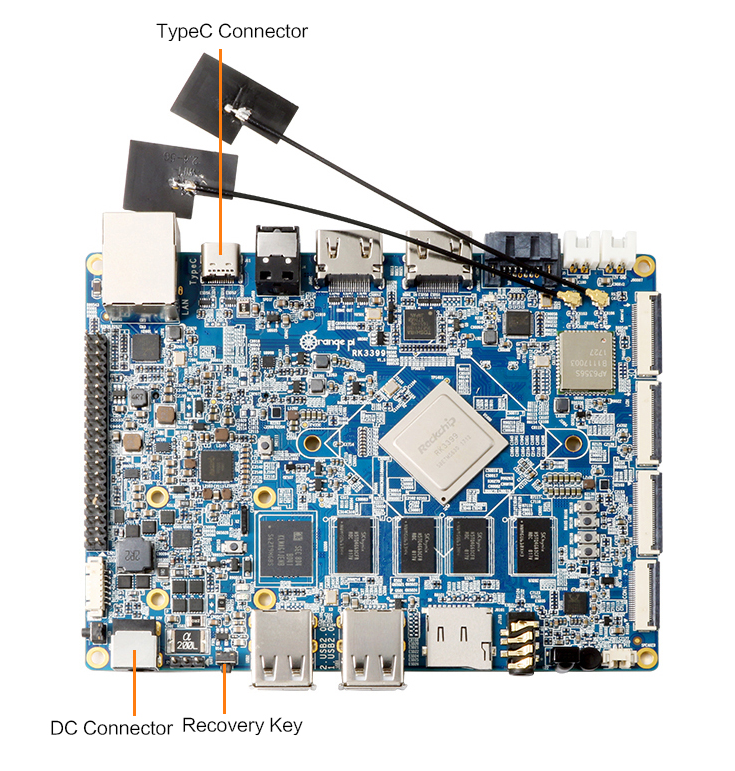
├── trust.img

└── uboot.img

Except the above method, the unity image(update.img) could also be generated via Linux\_Pack\_Firmware.

1. **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.。

## **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.

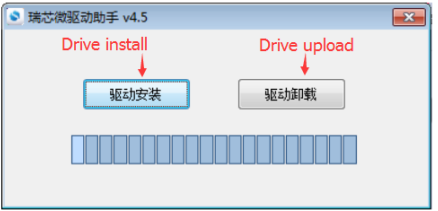
Before image writing, we need to first install RK USB driver on Windows.

### 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:

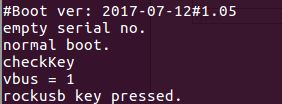


Connect with Type-C data cable and PC with Orange Pi RK3399 after installed USB driver, there would be show the status of USB driver on lower right corner like the following:

### RockChip USB DeviceEnter 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 :

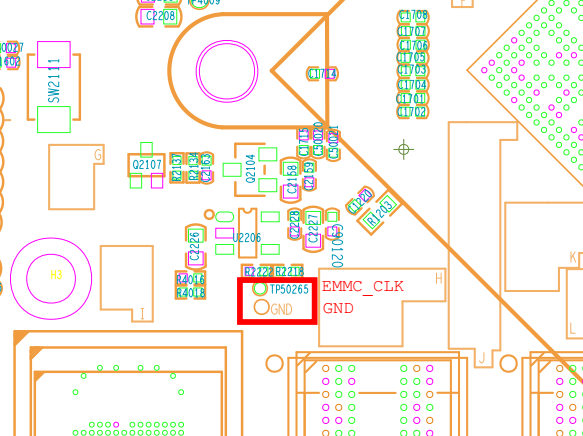


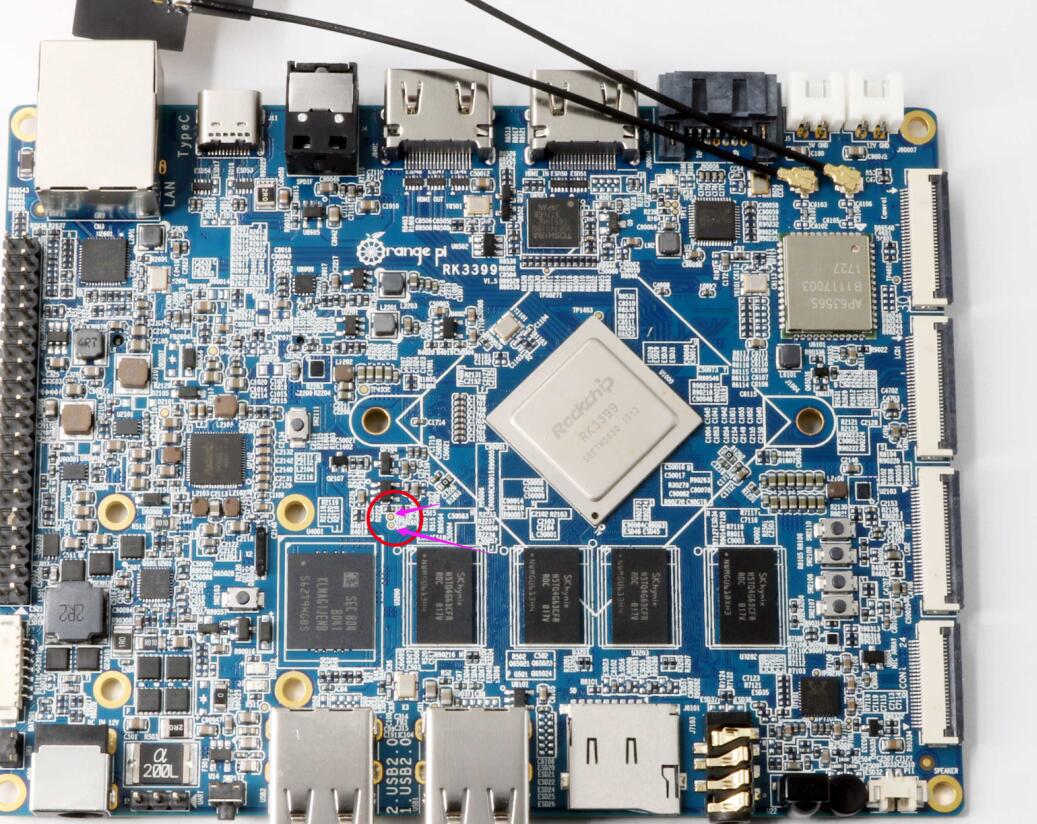
There would be log information if you connect debug pin.

If there is no Loader device, then you need to flash image with short connect like the following:

When short connect EMMC Clk and GND, insert DC power supply, enter into Maskrom mode. And insert Type-C cable to connect both PC and RK3399.

Location of EMMC Clk is show on the following:

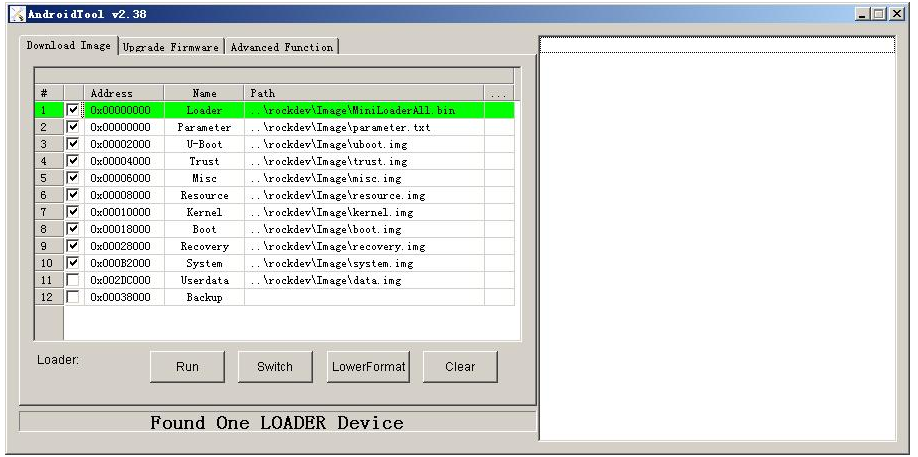




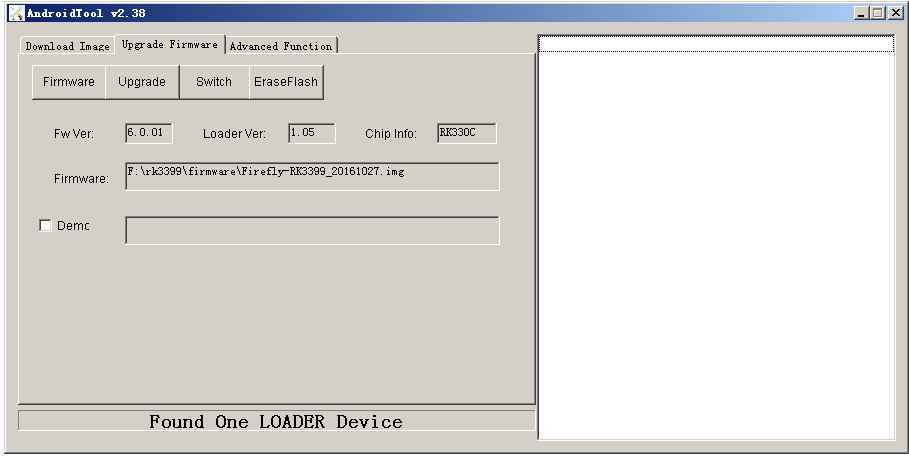
* Insert DC power supply

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.



* 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.

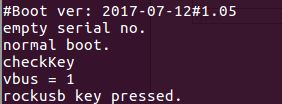


## **Flashing image on Linux**

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

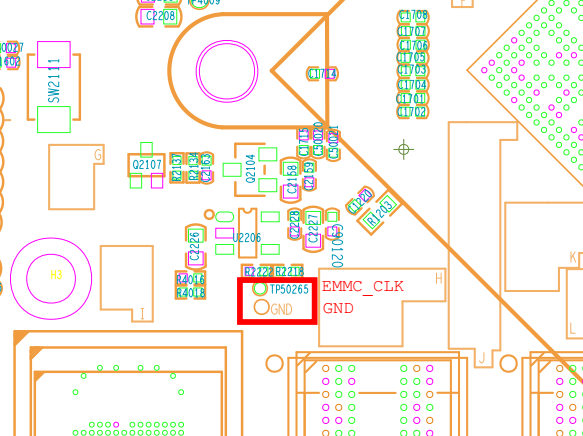
Since OrangePi RK3399 supports powered by TypeC, power voltage on USB of PC is enough for flashing image. However, it may not enough to support Orange Pi booting. In that case, we need to insert DC power supply to make sure the Orange Pi could boot successfully after flashed image.

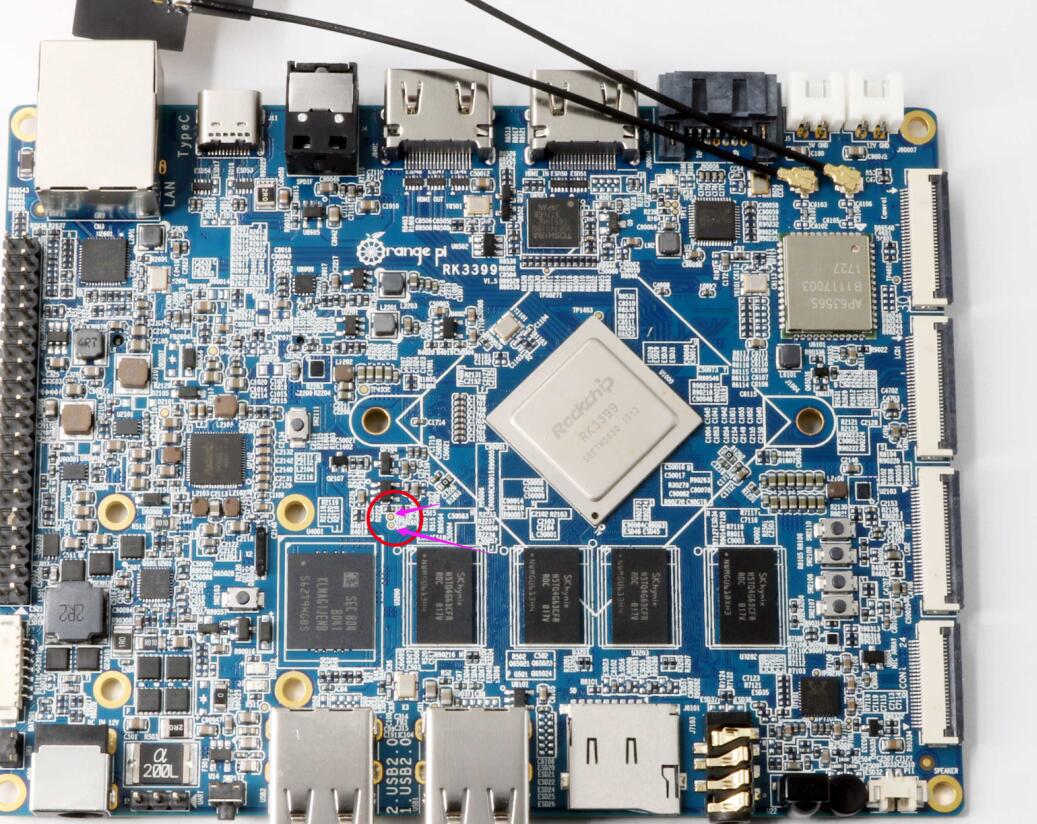
There would be log information if you connect debug pin.

If there is no Loader device, then you need to flash image with short connect like the following:

When short connect EMMC Clk and GND, insert DC power supply, enter into Maskrom mode. And insert Type-C cable to connect both PC and RK3399.

Location of EMMC Clk is show on the following:





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

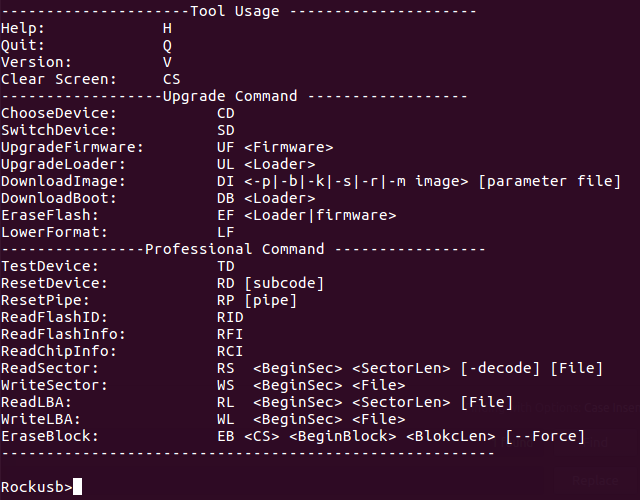
Enter Q: Exit flashing tool

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



* 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



**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



**UF Command:** used for unity flashed image update.img

**UF <Firmware>**

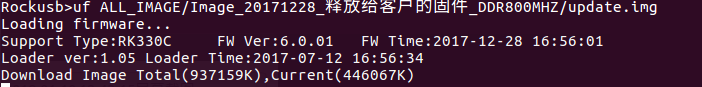
The only parameter is used to specify the need to burn the firmware path.

For example, if my firmware path is:

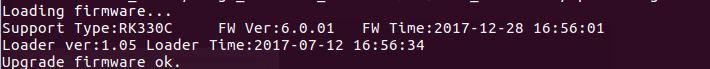
RK3399\_IMAGE/Image\_Android6.0\_20171228.img

Then the command should be:

Rockusb>uf RK3399\_IMAGE/Image\_Android6.0\_20171228.img



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



1. **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

## **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

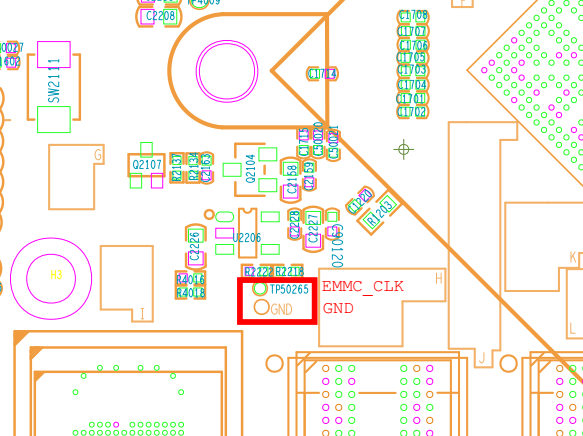
2. Use metal tweezers to keep TP50265 and GND connected

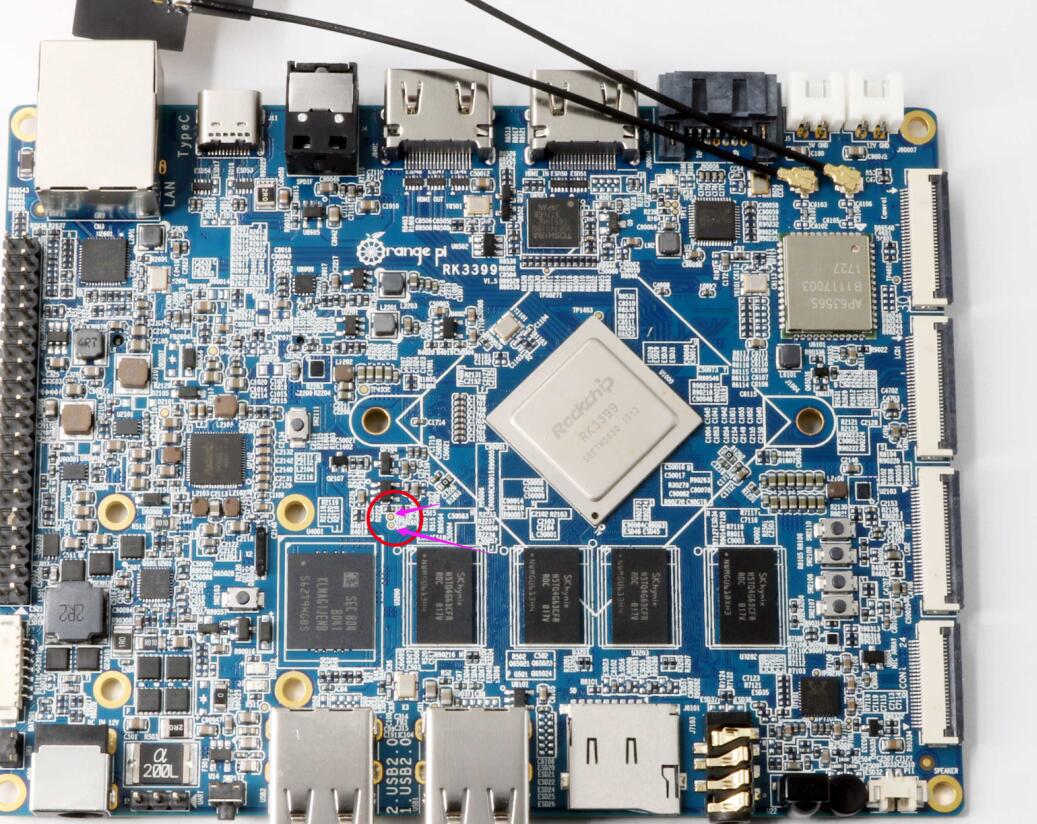
3. Power on the board with DC

4. Wait a moment then release the metal tweezers

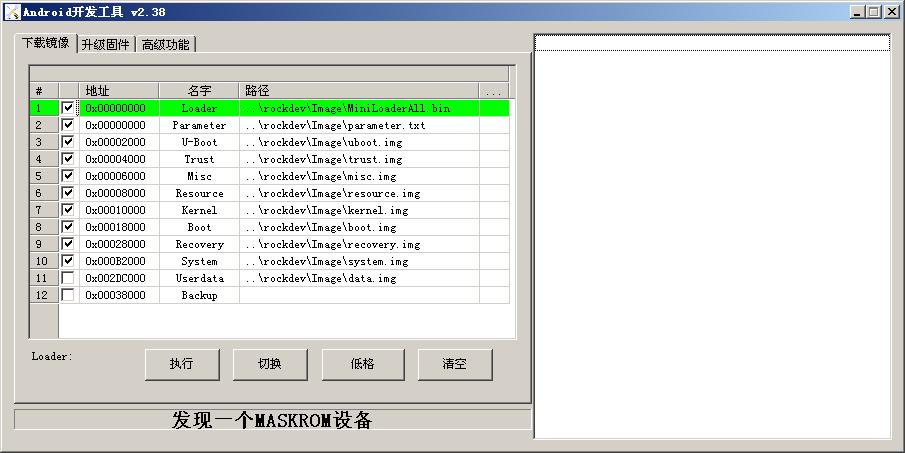
5. Use a micro USB Type-C cable to connect device and host PC

6. Device should enter MASKROM mode



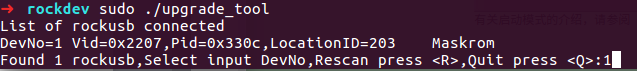


It would show the following with AndroidTools on Windows:

[](http://www.t-firefly.com/index.php/%E6%96%87%E4%BB%B6:Rk3399_maskrom_AndroidTool.jpg" \o "maskrom_AndroidTool)

After this could flash the image in the normal way.

It would show the following if use upgrade\_tool on Linux:



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.

1. **Linux Environment Construction and Firmware Compilation**
2. **Download Linux Source Code**

* OrangePi Souce Code Downloader

The Linux source for the Orange Pi RK3399 has been uploaded to GitHub, the kernel version is Linux 4.4, we can use it

Download with the downloader for OrangePi Linux source code, the method to get the downloader source code is as follows:

$ sudo apt-get install git

$ git clone https://github.com/orangepi-xunlong/OrangePi\_Build.git

$ cd OrangePi\_Build

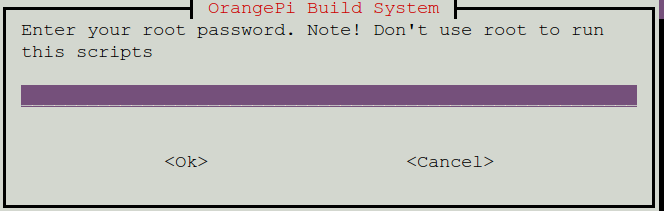
$ ls

Build\_OrangePi.sh lib README.md

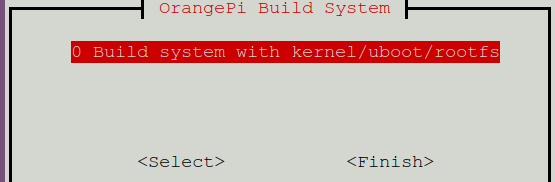
* Run the downloader

$ ./Build\_OrangePi.sh

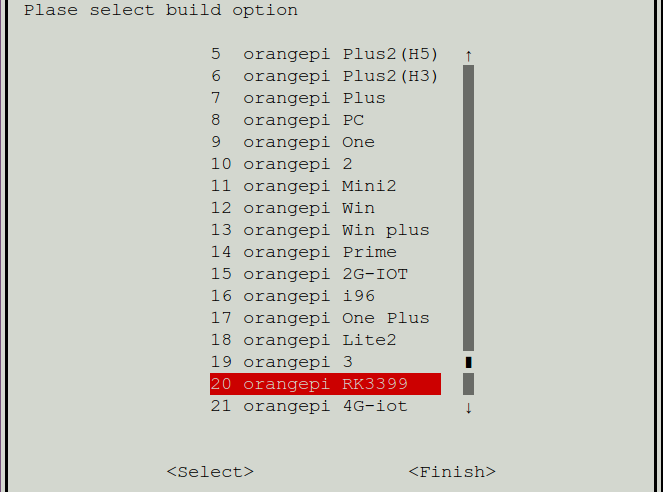
Enter the root password and press Enter



Select 0 Build system with kernel/uboot/rootfs, enter the development board model selection interface.



Select 20 orangepi RK3399 and press Enter to start downloading the Linux source code for Orange Pi RK3399



The downloaded source code will be stored in the same directory of OrangePi\_Build

$ ls ../

OrangePi\_Build OrangePiRK3399

The Linux source code directory structure of OrangePi RK3399 is as follows

$ cd OrangePiRK3399

$ tree -L 1

.

├── build.sh -> scripts/build.sh Compile the startup script

├── external Store extra configuration files

├── kernel Linux kernel source Linux 内核源码

├── output Store output files 存放输出文件

├── scripts Compile script 编译过程使用的脚本文件

├── toolchain Cross compilation tool 交叉编译工具链

└── uboot uboot source code uboot源码

6 directories, 1 file

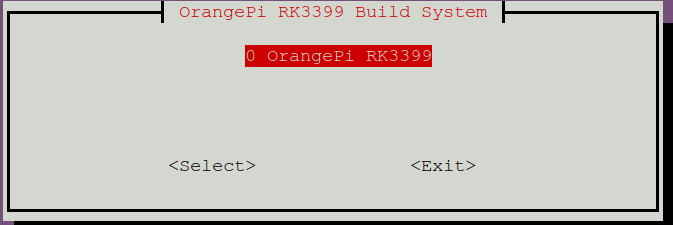
1. **Compile Linux source code**

* Execute the compile startup script

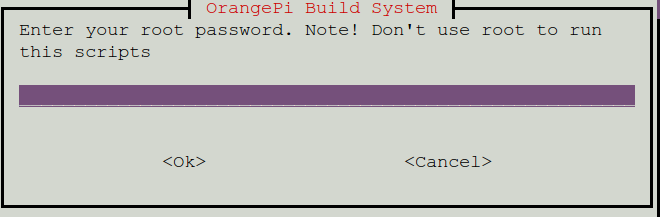
$ cd OrangePiRK3399

$ ./build.sh

**Select Enter**



Enter the root password and press Enter, then select the function needed to be executed



The functions of each option are as follows:

* 0 Build Release Image Compile Ubuntu or Debian Release image
* 1 Build Uboot Compile uboot source code
* 2 Build Linux Compile Linux kernel source code
* 3 Update kernel Image Update kernel

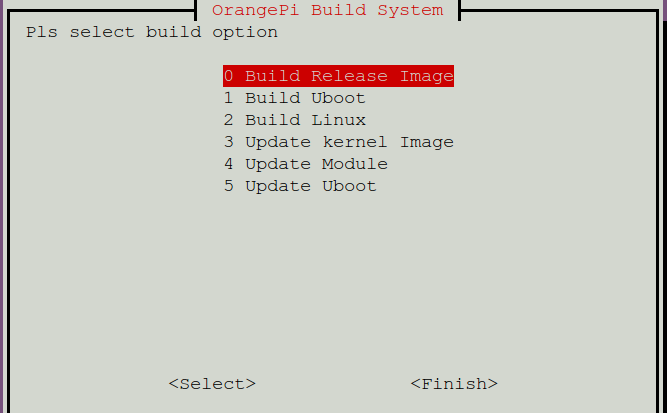
0 SD Card Update the kernel of Linux in SD card

1 EMMC Update the kernel of Linux in EMMC

* 4 Update Module Update the kernel module of Linux in SD card
* 5 Update Uboot

0 SD Card Update the uboot of Linux in SD card

1 EMMC Update the uboot of Linux in EMMC



After compiling u-boot and kernel source code,the final generated file will be stored in the output directory

Uboot image

$ tree -L 1 output/u-boot/

├── idbloader.img

├── rk3399\_loader\_v1.08.106.bin

├── trust.img

└── uboot.img

0 directories, 4 files

Kernel image

$ tree -L 1 output/kernel/

output/kernel/

├── Image

└── rk3399-orangepi.dtb

0 directories, 2 files

**Rootfs Compilation**

Refer to the chapter "Building a Compilation Environment and Making a Rootfs Image"

Execute the following steps to package the images of all partitions into one complete firmware:

$ cd scripts

Package Image and extlinux configuration files to generate boot.img, path in the output folder

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

c for chip,represent chip

t for target,represent the generated image name

s for size,represent the pre-allocated size when making the firmware (does not represent the actual final firmware size, the final firmware will be dynamically re-adjusted), unit Mbyte

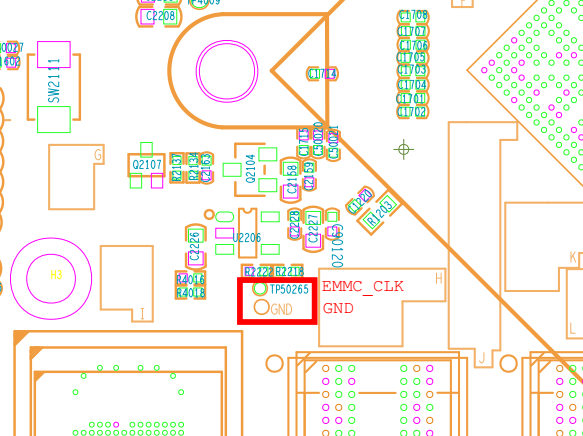
r for rootfs，represent the used rootfs path

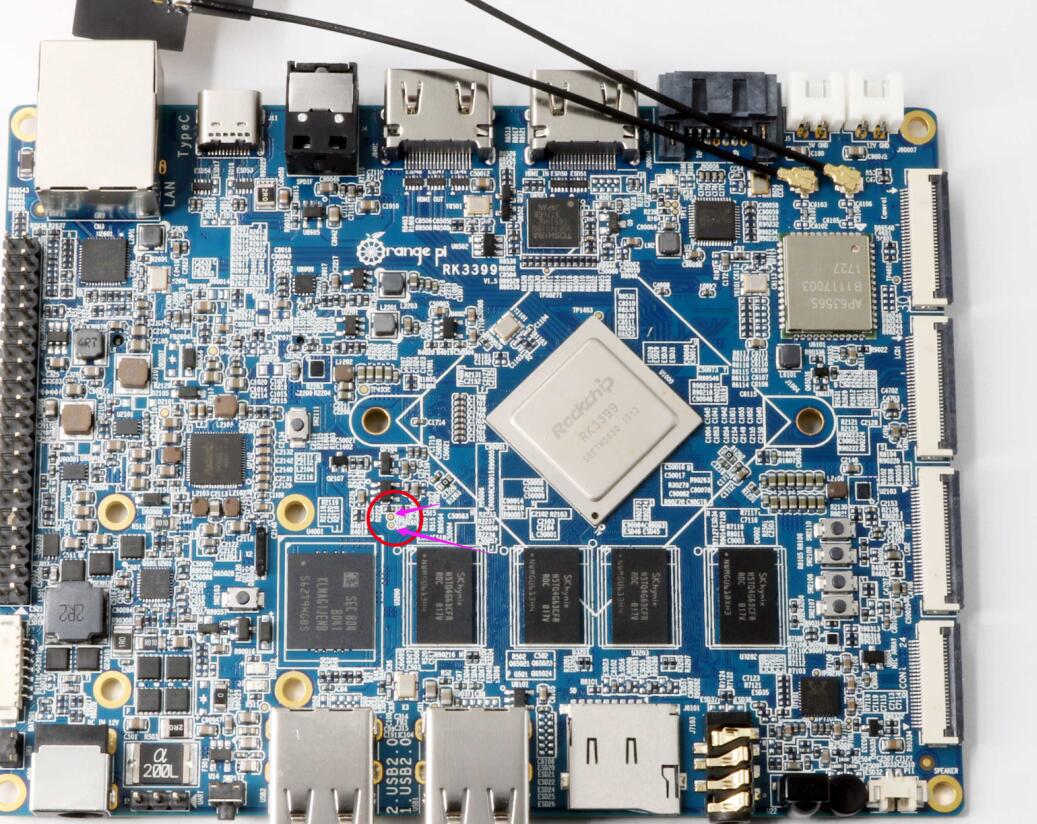
After the execution is completed, system.img is generated in the out directory as the final complete firmware.

1. **Linux firmware flashing**

Flash the complete linux firmware to the EMMC

1. Connect the DC power supply while short-connecting the Clk and GND of the EMMC. Enter the Maskrom mode. EMMC Clk is on the lower left corner of the miniPCIe on the back of the board shown as following picture





1. Connect the TypeC line
2. Execute the following command to flash

./flash\_tool.sh -c rk3399 -p system -i ../output/system.img

c for chip，represent the chip model   
p for partation，represent Partition，such as boot loader1 system   
i for image，represent the image path

You can also flash different partitions separately, such as

Separately flash boot：

./flash\_tool.sh -c rk3399 -p boot -i ../output//boot.img

Separately flash uboot:

./flash\_tool.sh -p loader1 -i ../output/u-boot/idbloader.img -c rk3399

./flash\_tool.sh -p loader2 -i ../output/u-boot/uboot.img -c rk3399

Flash the linux image published in the official website

./flash\_tool.sh -c rk3399 -p system -i \

OrangePi\_rk3399\_xenial\_desktop\_v1.0.img

1. **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 Construction.

**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

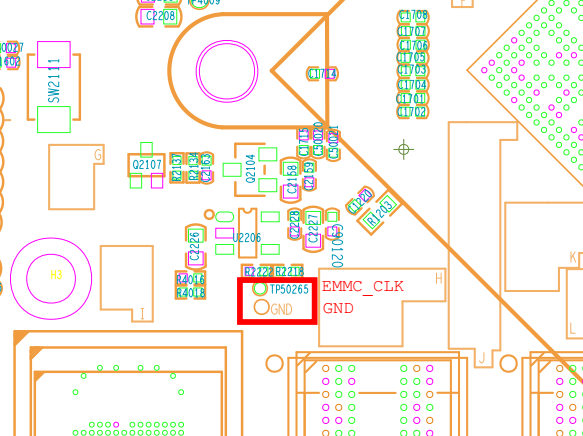
After execute the above command, there will be generate the unity full firmware(system.img ) on the directory of out.

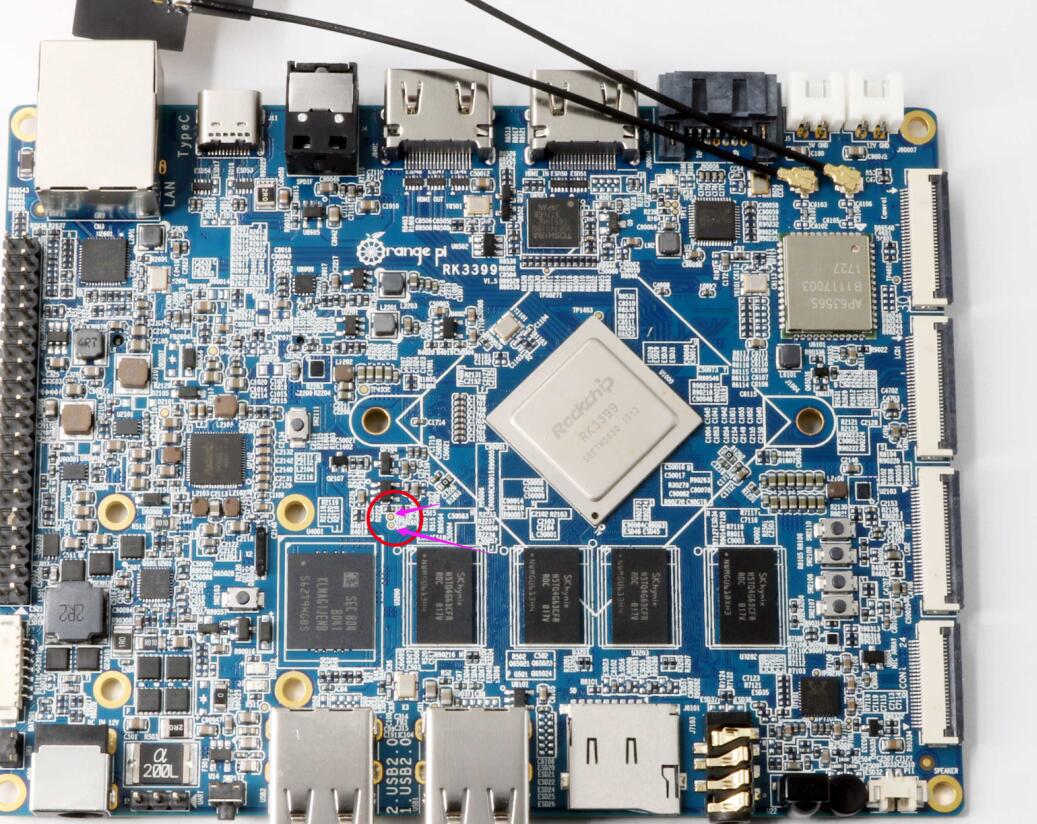
1. **Flash Linux Firmware**

**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 like the following shows:





1. Connect TypeC cable with PC and RK3399
2. 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

1. **Construct Compilation Environment and Make Rootfs Image**
2. **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.

1. **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

1. **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

1. **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

1. **Using Debug tools on OrangePi**

**Hardware: RK3399 and TTL to USB cable**

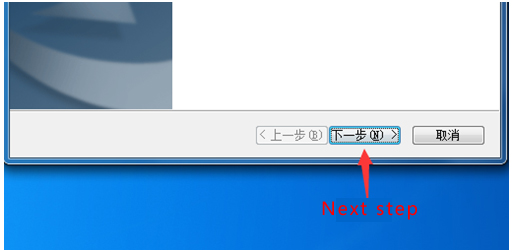
1. **Operation Steps on Windows**

**1) Install USB driver on Windows**

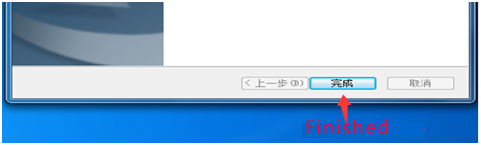
* Download and unzip the latest version of driver PL2303\_Prolific\_DriverInstaller\_v130.zip

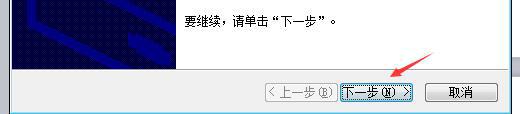
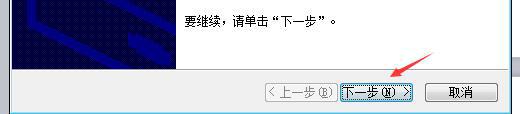
C:\Users\Administrator\Desktop\@D}{C@3CWCBD7O}YYO4I2`M副本.jpg

* Choose application installation as Administrator



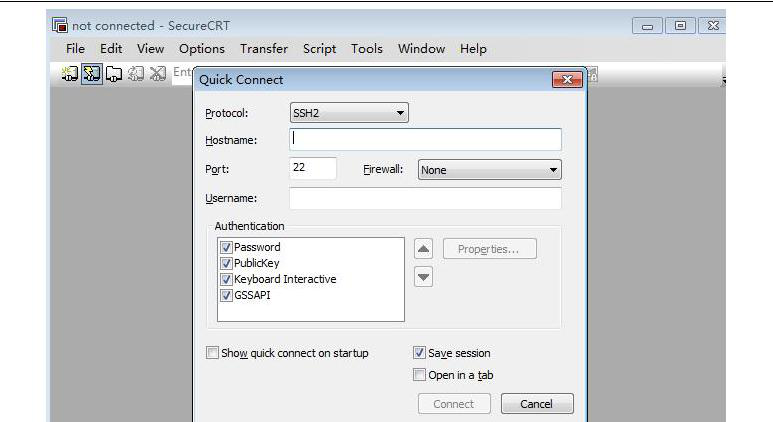
* Wait for completing installation



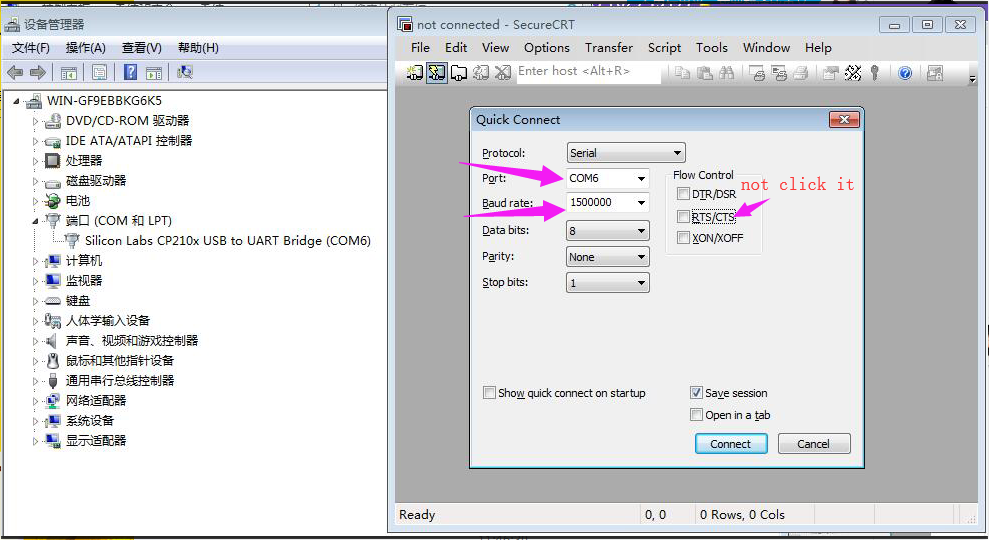
1. **Secure CRT Usage on Windows**

Use TTL to USB cable to connect RK3399 and PC, you could download securecrs\_33lc tool of RK3399 from official website on the download section.

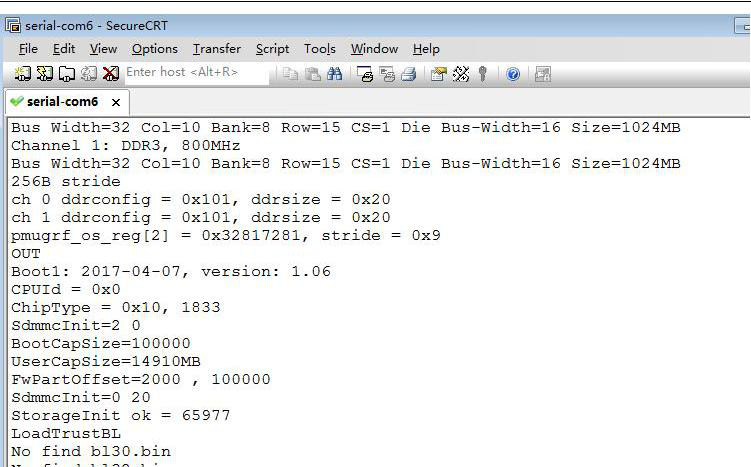
Click SecureCRTPortable after unzip the package, it would show like the following:

****

Select serial on Protocol, select corresponding port of your PC on port(you could use device manager to check the port number). On my PC it is COM6, input baud rate of 1500000, not click RTS/CTS.



Click connect and boot the board, there will print the debug log:

****

1. **Operation Steps on Linux**

There are minicom and kermit for debug on Linux, here we will make exampleofminicom since kermit do not support baud rate at 1500000. Connect RK3399 and PC with TTL cable.

* **Install and configure minicom**

Device node of serial port:

ls /dev/ttyUSB\*

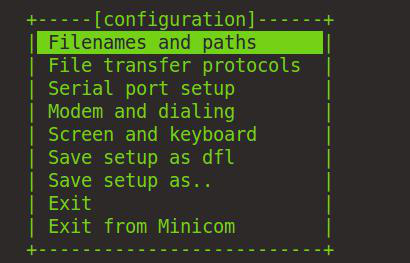


Device node is ttyUSB0

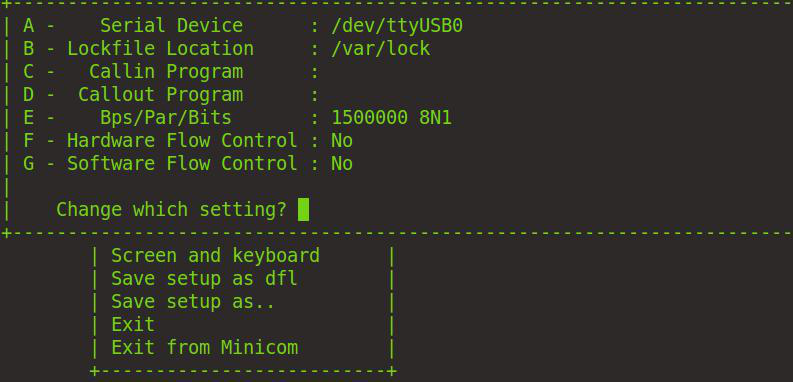
Execute command on the terminal and install minicom

sudo apt-get install minicom

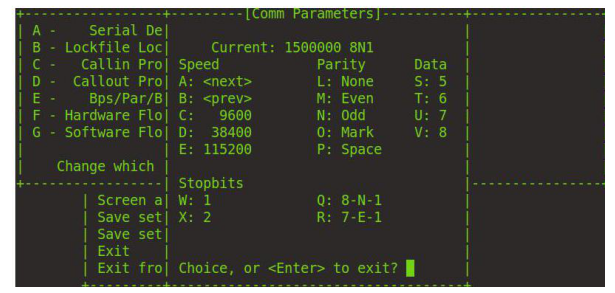
After installation, configure minicom, input sudo minicom -s on the terminal



Select Serial port setup and press Enter, it will show like the following:

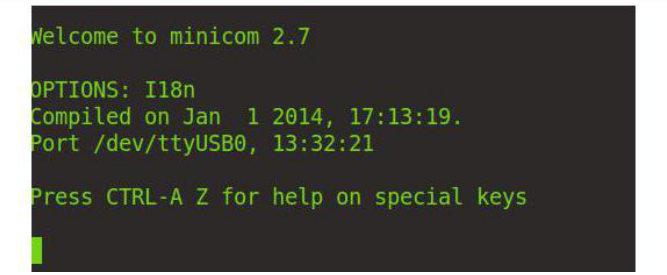


Press ***a*** to configure Serial Device as serial port device node, what I am working at is ttyUSB0, then press Enter. Press f to not click RTS/CTS. Press e will show the configure window like the following:

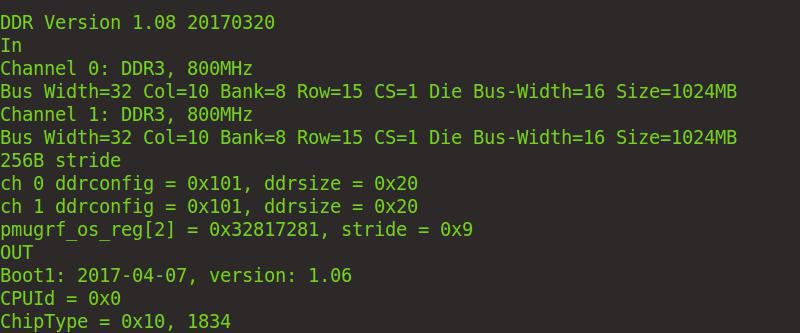




Select Save setup as dfl, press Enter and save it, then press ESC to exit.



Boot RK3399 it will print the log:



1. **Linux Using Method**
2. **Login account and passport**

Account：root, Passport：orangepi

Account：orangepi，Passport：orangepi

1. Extend rootfs partition

root@OrangePi:~# parted /dev/mmcblk1

GNU Parted 3.2

Using /dev/mmcblk1

Welcome to GNU Parted! Type 'help' to view a list of commands.

(parted) resizepart

Error: The backup GPT table is corrupt, but the primary appears OK, so that will

be used.

OK/Cancel? ok

Warning: Not all of the space available to /dev/mmcblk1 appears to be used, you

can fix the GPT to use all of the space (an extra 22343680 blocks) or continue

with the current setting?

Fix/Ignore? fix

Partition number? 7

End? [4193MB]? 15.6G

(parted) q

Information: You may need to update /etc/fstab.

root@OrangePi:~# resize2fs /dev/mmcblk1p7

1. GPU Performance Test

Start the system, click LXTerminal in the system menu System Tools to open the command line terminal, and run the command:

$ glmark2-es2

The test results and running scores of the Mali-T860 can be seen.

1. Compile decoding source test

* Play 4k vedio

Click LXTerminal in the system menu System Tools to open the command line terminal, and run the command:

$ test\_dec-gst.sh

Use which test\_dec\_gst.sh to find the location of this script

The default audio output is to the 3.5mm headphone jack, and the card variable can be modified to 2 output to hdmi

* Test compilation code

Execute

$ test\_enc.sh

1. Using HDMI IN function

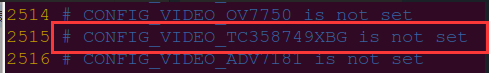
Hdmi in only supports receiving input signals of 1080p60, and connects the hdmi output port of other devices to the hdmi in interface of RK3399

Boot the system, open the command line terminal, run

$ test\_hdmiin.sh

1. Using ov13850 Camera

The camera1 interface can use the ov13850 camera, you need to disable the hdmi in configuration in the .config file



Recompile and replace the kernel

Boot the system, open the command line terminal, run

$ test\_camera.sh

7.Using GPIO

Use the wiringOP to operate gpio at the application layer, the installation steps are as follows

$ git clone <https://github.com/orangepi-xunlong/wiringOP>.git

$ cd wiringOP

$ export PLATFORM=OrangePi\_RK3399

$ ./build

the command line enter

gpio readall

