KIT1126B Linux6.1 IPC User Manual

V1.0

Boardcon Embedded Design

Overview

This document applies only to the KIT1126B development board. It is intended to help users quickly understand the hardware interfaces of the board and provides guidance for environment setup, source code compilation, firmware flashing, and functional testing of onboard hardware interfaces.

System Support

Development Board	IPC
LGA1126B-V1 + KIT1126B-V1	Y

Revision History

Version	Date	Author	Revision History
V1.0	2026-08-03	Liu Yuan	Initial version

Disclaimer

The information in this manual is provided for reference only. While Boardcon has made every effort to ensure the accuracy and reliability of the content, no guarantee is given regarding its completeness or correctness. All information in this manual is subject to change without prior notice.

Boardcon reserves the right to revise or update the contents of this manual at any time without prior notification. Boardcon shall not be held liable for any direct or indirect loss or damage arising from the use of this document or the products described herein.

Boardcon embedded design limited

2007-11 Haofang Tianji Plaza, 11008 Beihuan Avenue, Nanshan District, Shenzhen, Guangdong, China. 518051

URL: www.armdesigner.com | www.boardcon.com

Email: market@armdesigner.com

Technical Support Inquiries: support@armdesigner.com

Tel: +86-755-26481393 | +86-755-27571591

Content

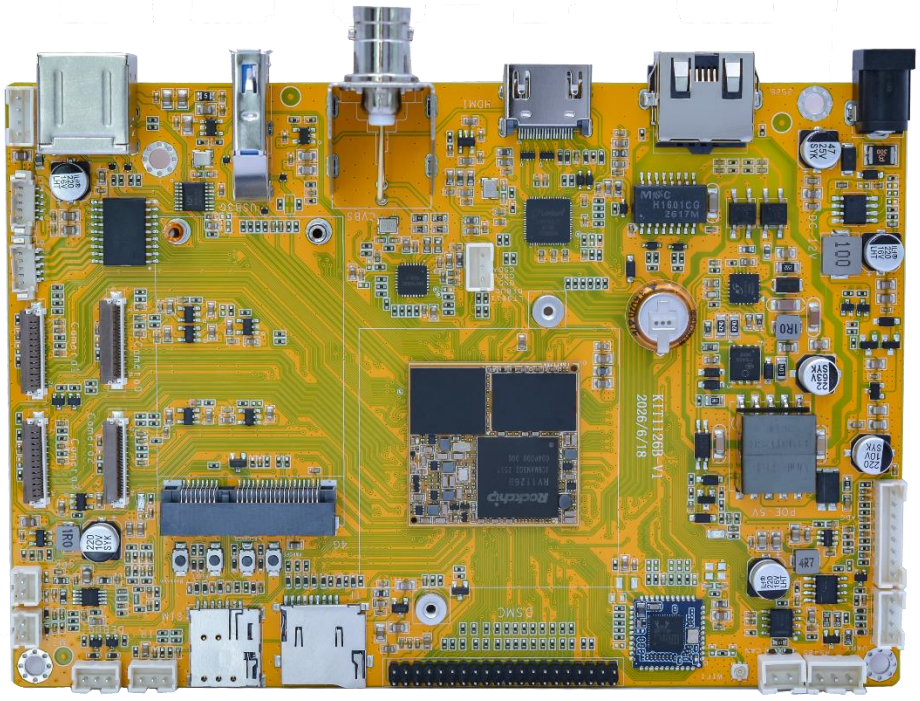
1. Introduction

1.1 Overview

KIT1126B is an embedded AI vision development board based on the Rockchip RV1126B processor. It adopts a modular design with a removable LGA1126B core board, providing a flexible platform for AI vision, intelligent recognition, security monitoring, industrial applications, and edge computing solutions.

The board provides rich peripheral interfaces for development and system integration, including four camera inputs, HDMI output, CVBS video input, USB3.0 DRD, three USB2.0 host interfaces, Ethernet with PoE support, Wi-Fi/BT, CAN, RS485, UART, ADC, MicroSD, 4G module expansion, IR, MIC input, speaker output, and motor control interface.

With its soldered core-board and carrier-board design, KIT1126B provides higher reliability, better thermal performance, and improved system stability. This design makes KIT1126B suitable for AI application development, product evaluation, and mass-production deployment.

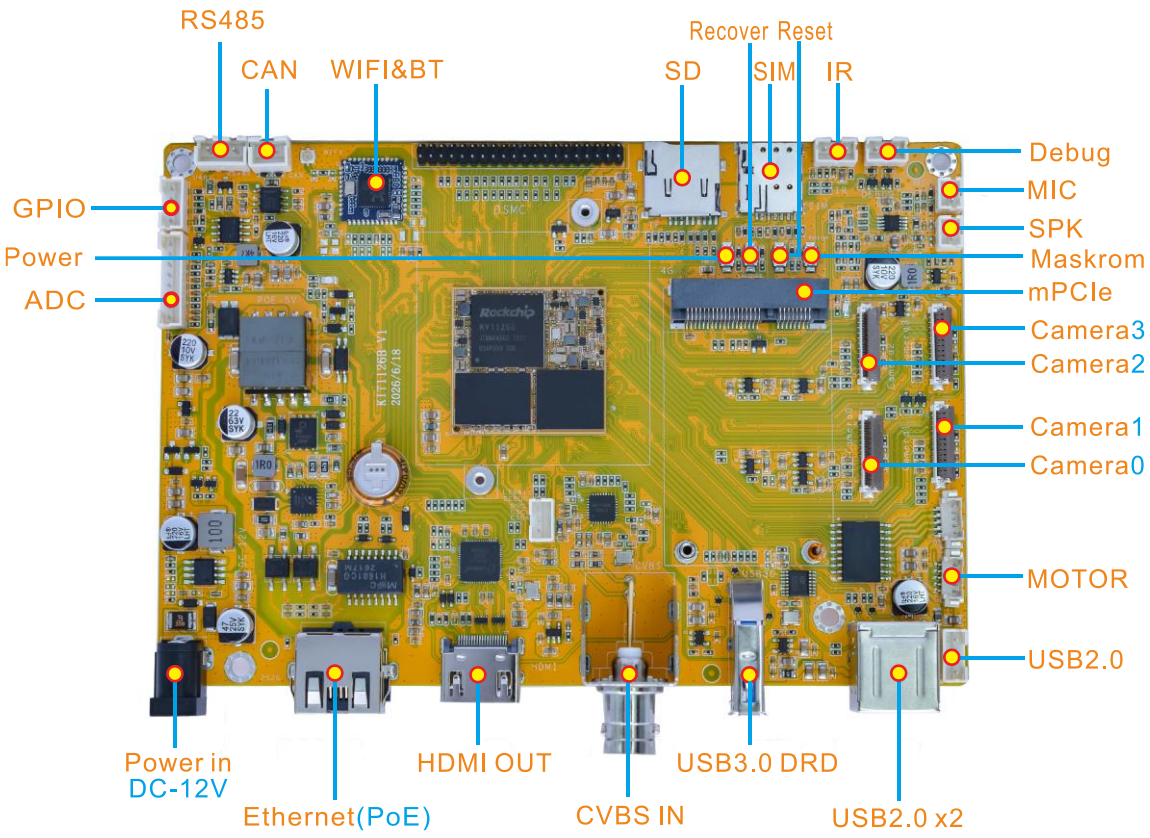


1.2 Product Parameters

Basic Parameters		
SOC		RV1126B
CPU		• Quad-core Arm Cortex-A53 64-bit processor • Neon SIMD and FPU support • 32 KB I-cache and 32 KB D-cache per core • 512 KB unified L2 cache
NPU		• Up to 3 TOPS AI computing power • Supports INT4, INT8, INT16, and FP16 operations • Supports mainstream AI frameworks, including TensorFlow, Caffe, TFLite, PyTorch, ONNX NN, and Android NN
Video	Decoder	• H.265/H.264 video decoding up to 3840×2160@30fps • JPEG decoding
	Encoder	• H.265/H.264 video encoding up to 3840×2160@30fps • JPEG encoding
AI-ISP		• Built-in AI-ISP engine • Supports up to 8MP@30fps • Supports AI-assisted image enhancement and spatial noise reduction
RAM		2GB LPDDR4x (up to 4GB)
ROM		8GB eMMC (up to 256GB)
Support system		Debian, Buildroot
Hardware Parameters		
Extended Storage		• Support 1x MicroSD Card
Display		• Support 1x HDMI
Audio		• Support 1x Speaker

	• Support 1x MIC
USB	• Support 3x USB2.0 • Support 1x USB3.0 DRD
Network	• Support 1x 100M Ethernet(PoE) • Support 1x WIFI/BT module
Camera	• Support 4x MIPI Camera • Support 2x CVBS Camera
Peripheral communication	• Support 1x RS485 • Support 2x CAN
Other parameters	Support 1xDebug UART, 1xADC, 1xIR, 1xMotor
Electrical Parameters	
Power supply input voltage	12V/3A
RTC input voltage	3V/0.6uA
Operating temperature	0°C to 70°C
Storage temperature	-40°C to 85°C
Structural Parameters	
Core board dimensions	28.75mm x 30.0mm
Motherboard dimensions	155.0mm x 105.0mm

1.3 Hardware Interface Introduction



Interface parameters	
Power in DC 12V	12V DC power input interface
Ethernet(PoE)	100M Ethernet RJ45 interface with PoE support
HDMI OUT	HDMI OUT interface
CVBS IN	CVBS video input interface
USB3.0 DRD	USB 3.0 DRD interface, supporting Host and Device modes
USB2.0 x2	Dual-layer USB2.0 HOST interface
USB2.0	USB2.0 Host interface via GPIO header
MOTOR	Stepper motor control interface
Camera0	Camera0 interface
Camera1	Camera1 interface
Camera2	Camera2 interface

Camera3	Camera3 interface
mPCIe	4G module interface
SPK	Speaker output interface
MIC	Differential microphone input interface
Debug	Debug interface
IR	IR interface
SIM	SIM card interface
SD	MicroSD card slot
WIFI&BT	WIFI and Bluetooth module
Maskrom	Maskrom key
Reset	Reset key
Recovery	Recovery key
Power	Power key
CAN	CAN communication interface
RS485	RS485 communication interface
ADC	ADC interface

2. Install Drivers and Tools

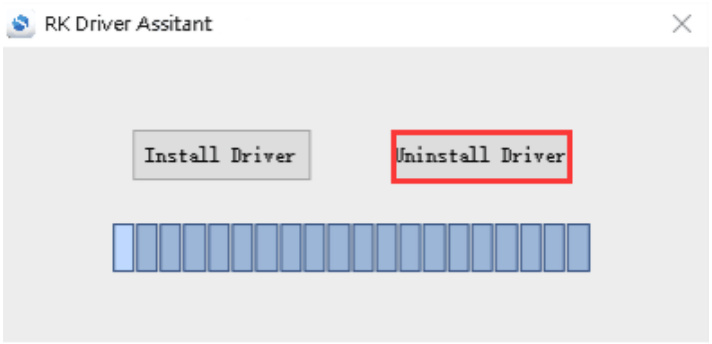
To download firmware and debug in the terminal, the following drivers and software need to be installed (for Windows computers):

Number	Driver name	Driver	Use
1	RK Driver Assistant	DriverInstall.exe	OTG USB driver installation assistant
2	CH9102x	SETUP.EXE	Serial port debugging driver
3	Serial Terminal Tool	SecureCRT.exe	Debugging tool

2.1 Install RK Driver Assistant

Step 1: Open *DriverAssistant/DriverInstall.exe*.

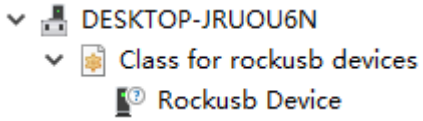
Step 2: To avoid driver conflicts, click **"Uninstall Driver"** to uninstall the driver.



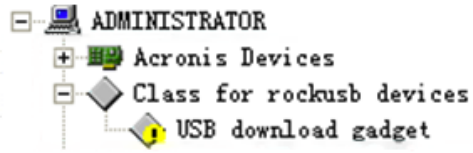
Step 3: Click button **"Install Driver"** to install.

Step 4: After the installation is complete, connect the board and PC with USB_A cable and press the **Recovery** key and hold then power the board, the following information is displayed in the Computer **Device Manager**, indicating that the USB driver was

successfully installed.

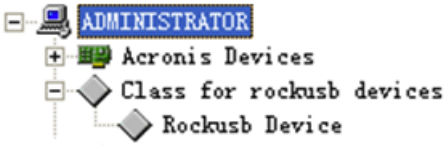


Step 5: If the following device information appears in the **Device Manager** after the operation in Step 4, user need to proceed to the next step.



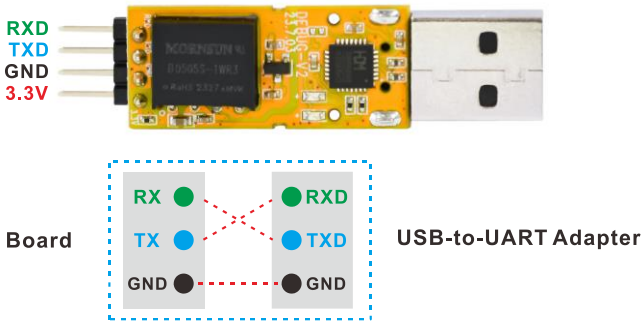
Step 6: The WINDOW will pop up found New Hardware Wizard dialog box, choose to install from the specified location, and then select: *DriverAssistant/ADBDriver*.

Step 7: After the installation is completed, the following device information can be seen in the Computer **Device Manager**.



2.2 Install CH9102X Driver

2.2.1 How to Connect the Serial Port Tool



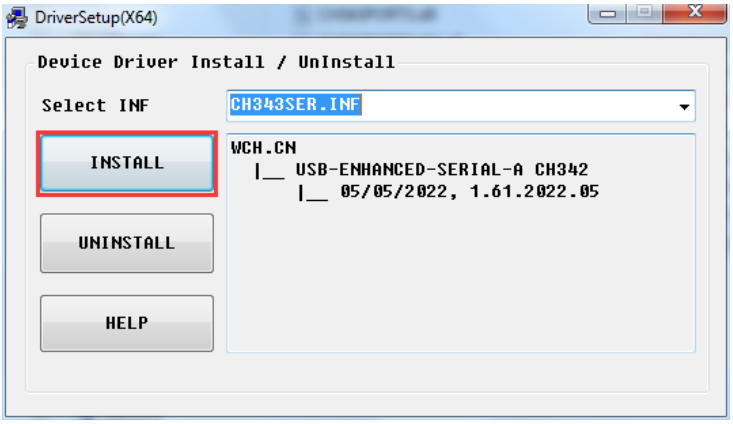
Pin	Connection Description
RXD	Receive, connect to TX pin of the board.
TXD	Transmit, connect to RX pin of the board.
GND	Ground, connect to GND pin of the board.
3V3	No need to connect.

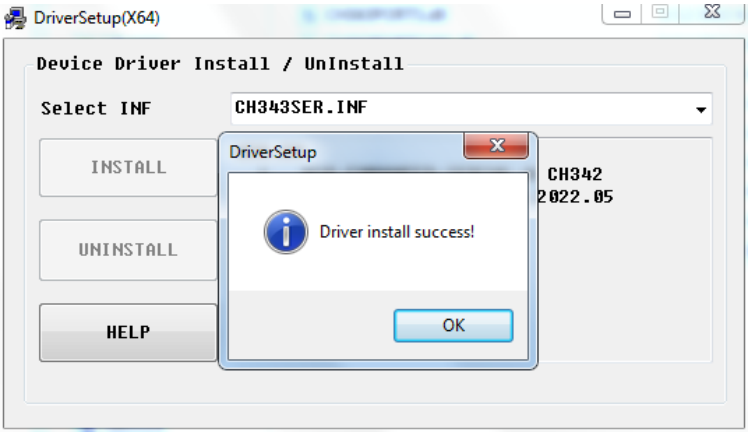
2.2.2 Install Driver

Step 1: Connect the CH9102X module to the PC.

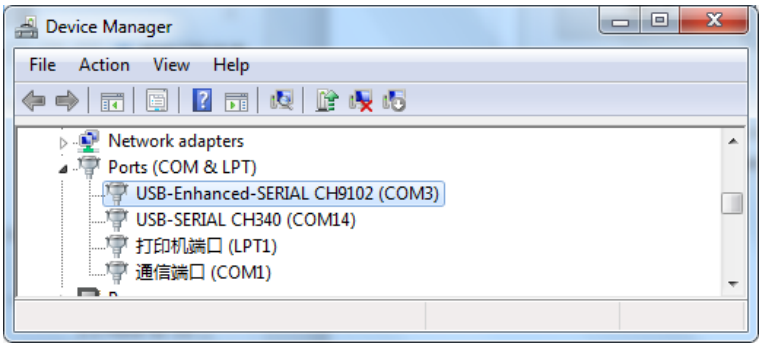
Step 2: Unzip *CH343SER.ZIP* on Windows.

Step 3: Select and install the corresponding *SETUP.EXE* according to the computer properties.





Step 4: After the installation is completed, the device will be listed under **Device Manager** ports with unique serial port assigned.

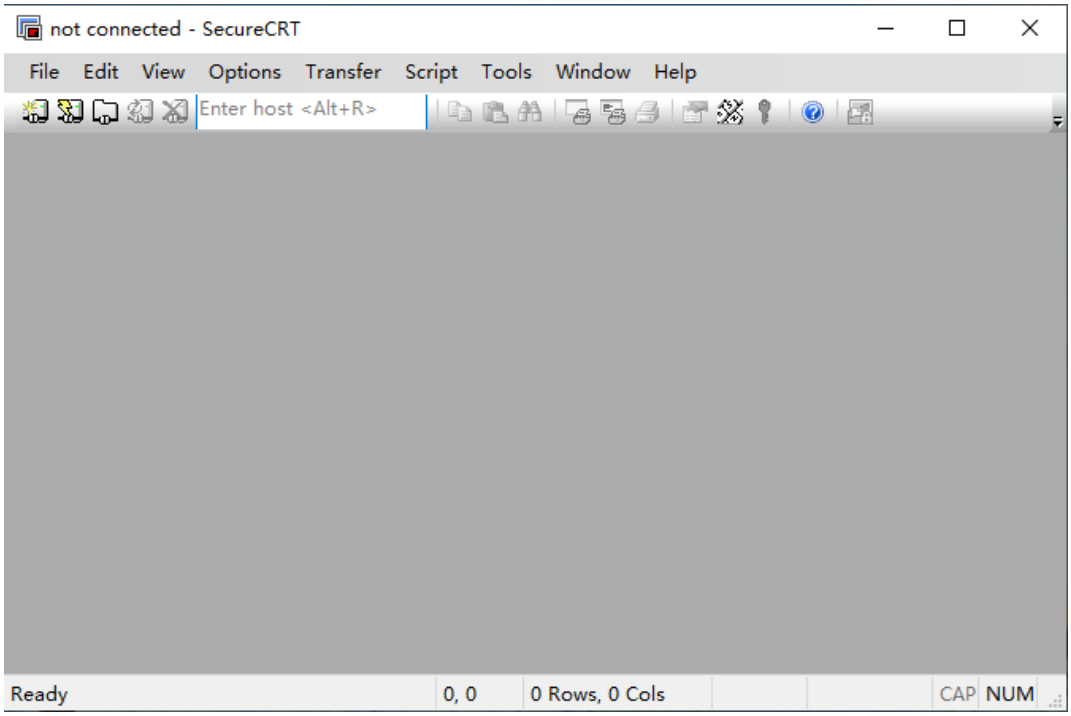


2.3 Install Serial Terminal Tool

The serial terminal SecureCRT is used for debugging in Windows. It can be used directly after decompression.

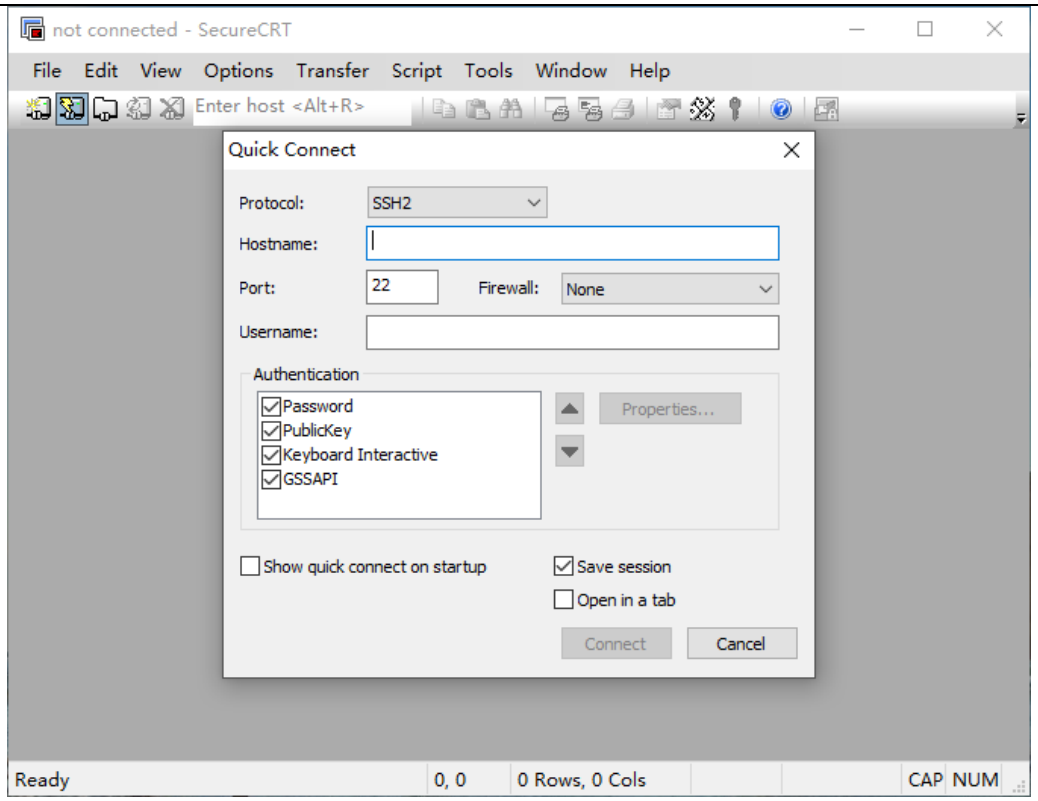
Step 1: Unzip *Platform/SecureCRT.rar* on PC.

Step 2: Click *SecureCRT/SecureCRT.exe* open to the SecureCRT.

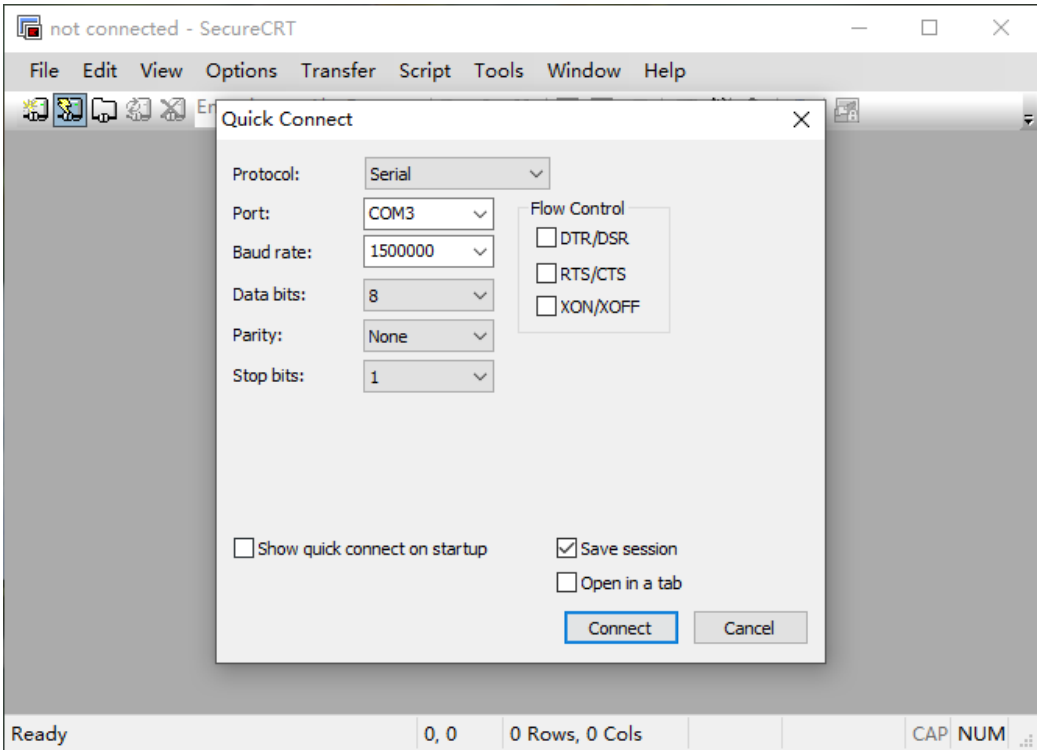


Step 3: Confirm that the CH9102X driver has been installed and the CH9102X module is connected to the PC.

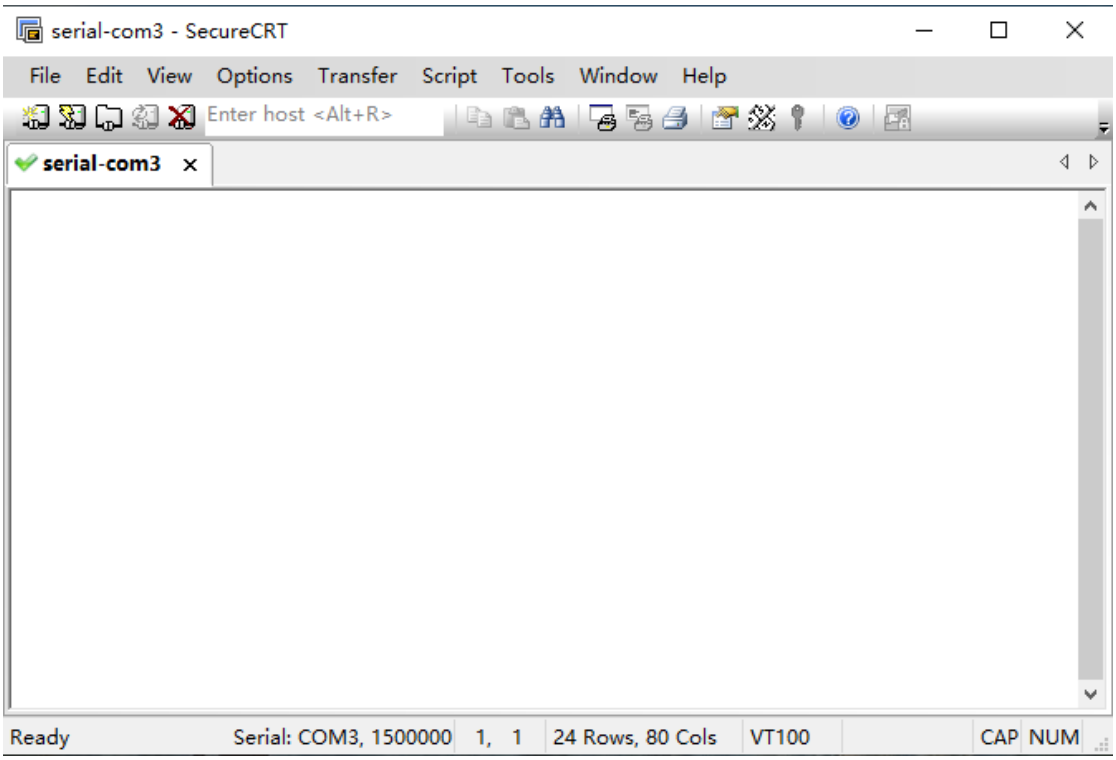
Step 4: Click the **"Quick Connect"** button to go to the Quick Connect configuration screen.



Step 5: Configure as shown in the following figure.



Step 6: After clicking the **"Connect"** button, the terminal serial interface will be successfully accessed.



3. Upgrade Introduction

3.1 Upgrade Mode

The firmware of this board must be upgraded in MaskRom Mode via USB cable.

Before upgrading, make sure the required USB driver has been installed on the PC. For driver installation instructions, refer to 2.1 Install RK Driver Assistant.

3.1.1 How to Enter MaskRom Mode

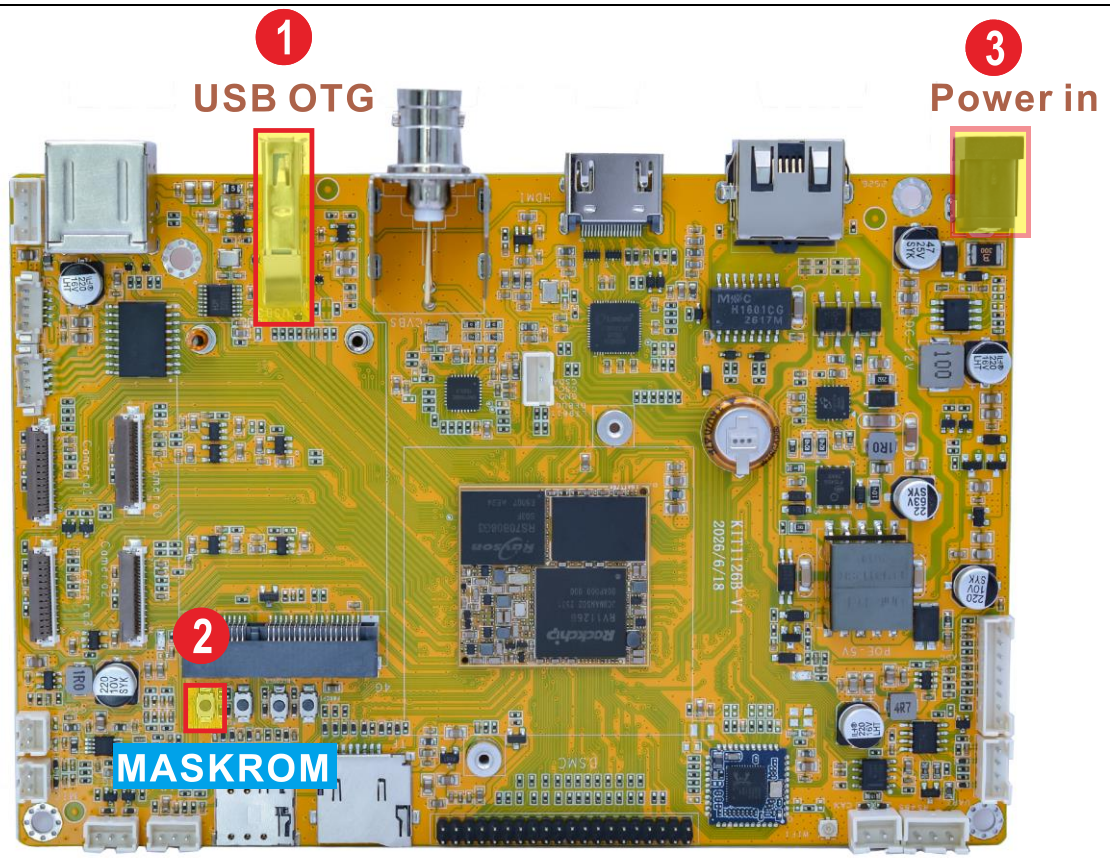
3.1.2.1 Hardware Method

Step 1: Disconnect the power adapter.

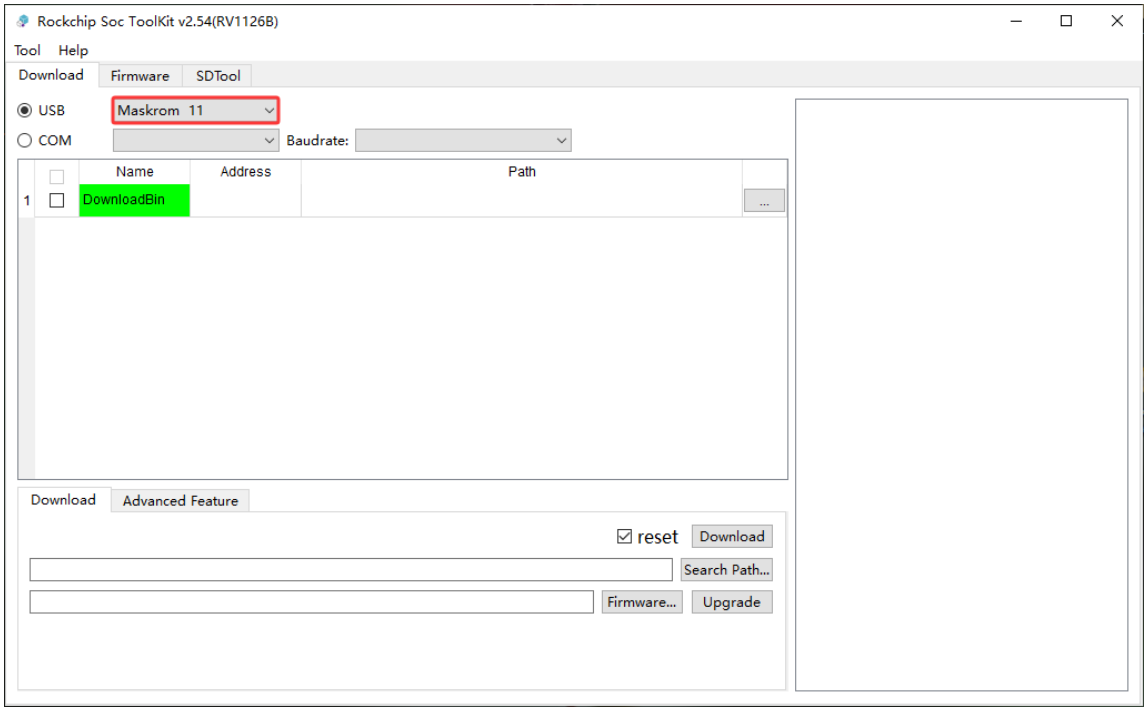
Step 2: Connect one end of the USB-A cable to the host PC and the other end to the development board.

Step 3: Press and hold the **Maskrom** button on the board.

Step 4: Connect the power supply.



Step 5: After a few seconds, release the MaskRom button when the flashing tool detects the device and displays "**MASKROM**" mode.

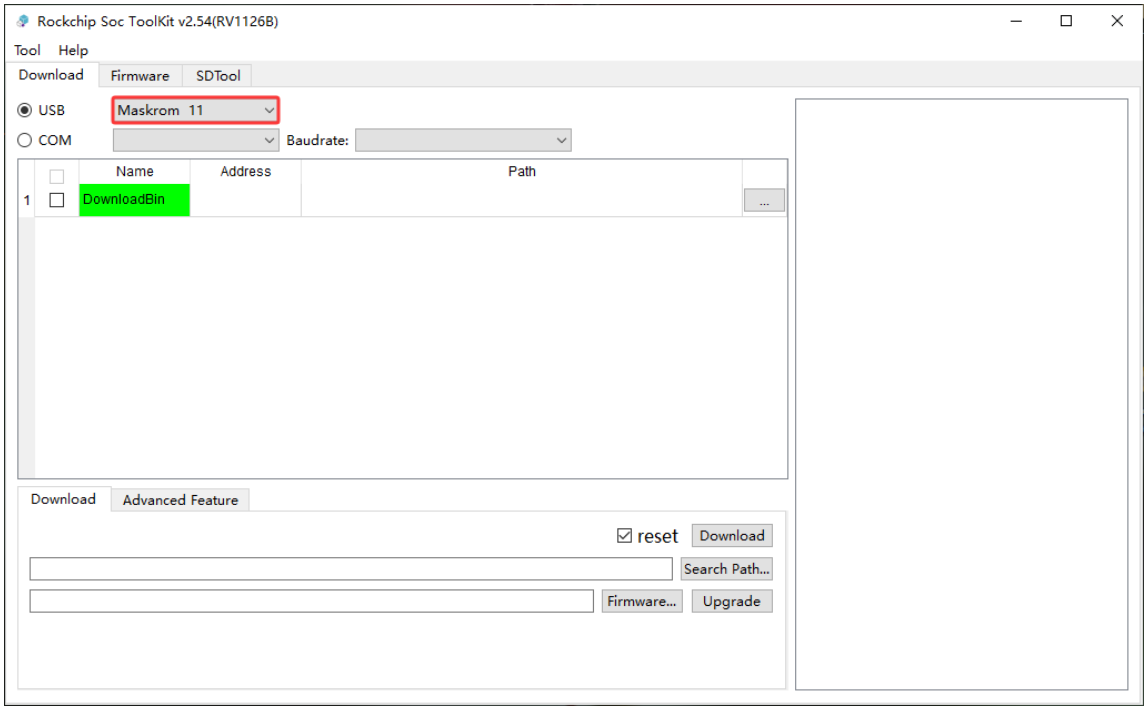


3.1.2.2 Serial Terminal Method

Step 1: Connect one end of the USB_A cable to the host PC and the other end to the

development board, then open the serial debug terminal.

Step 2: Press and hold **Ctrl + B** in the serial terminal, then power on the board. Release the keys when the flashing tool detects "**MASKROM**".

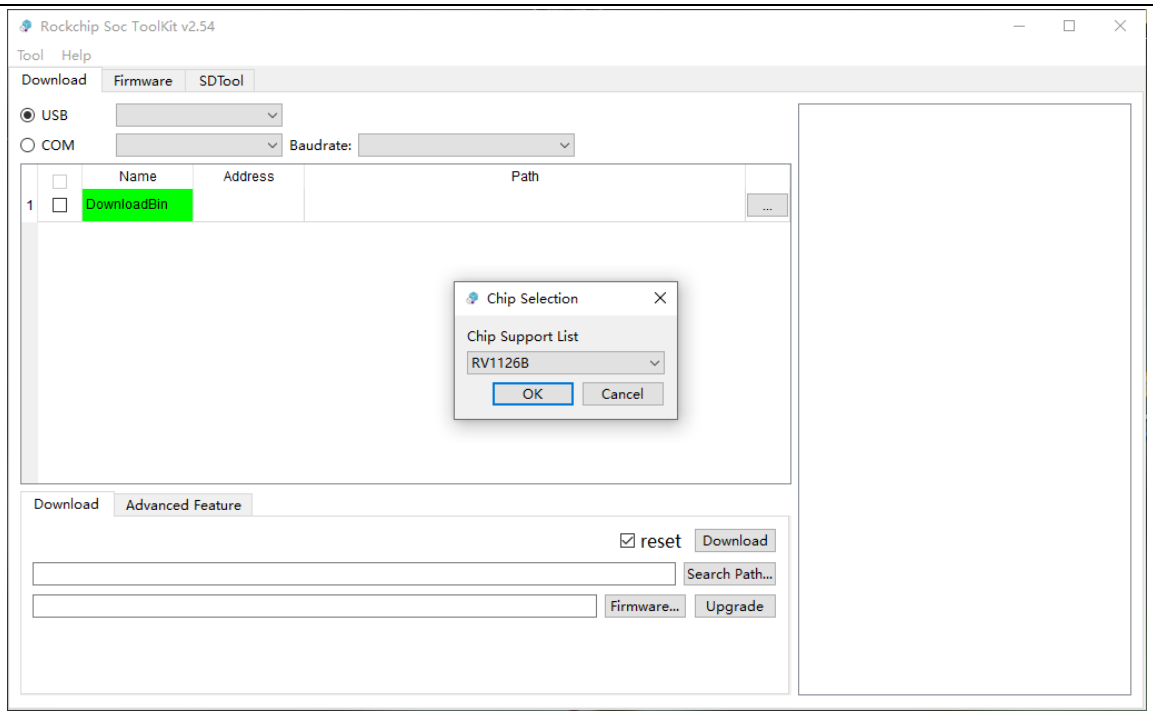


3.2 Firmware Flashing

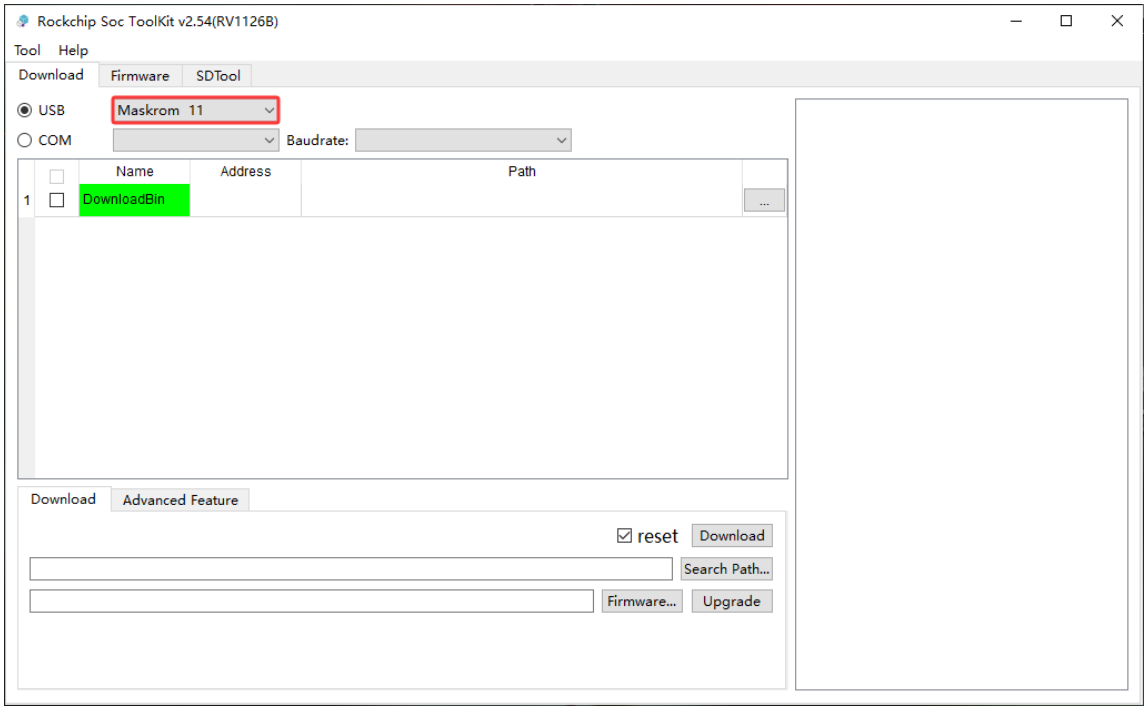
3.2.1 Flash Full Firmware Image

Step 1: Open *SocToolKit\SocToolKit.exe*.

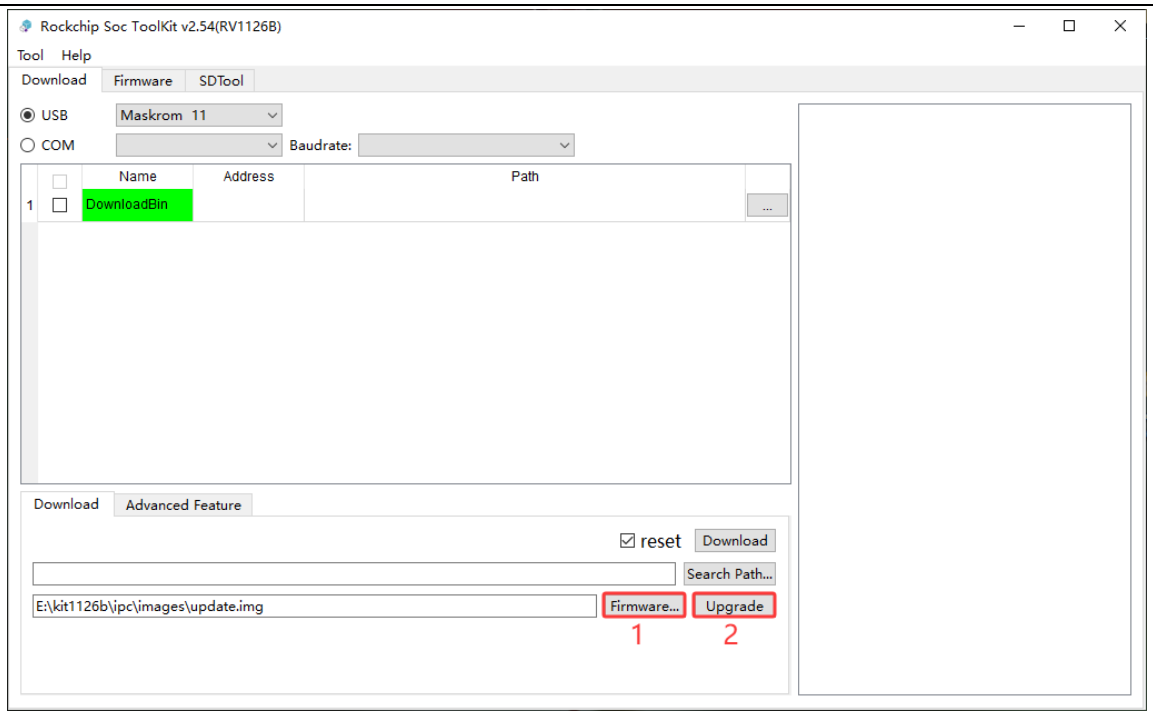
In the Chip Selection dialog box, keep the default chip option "**RV1126B**", and then click "**OK**".



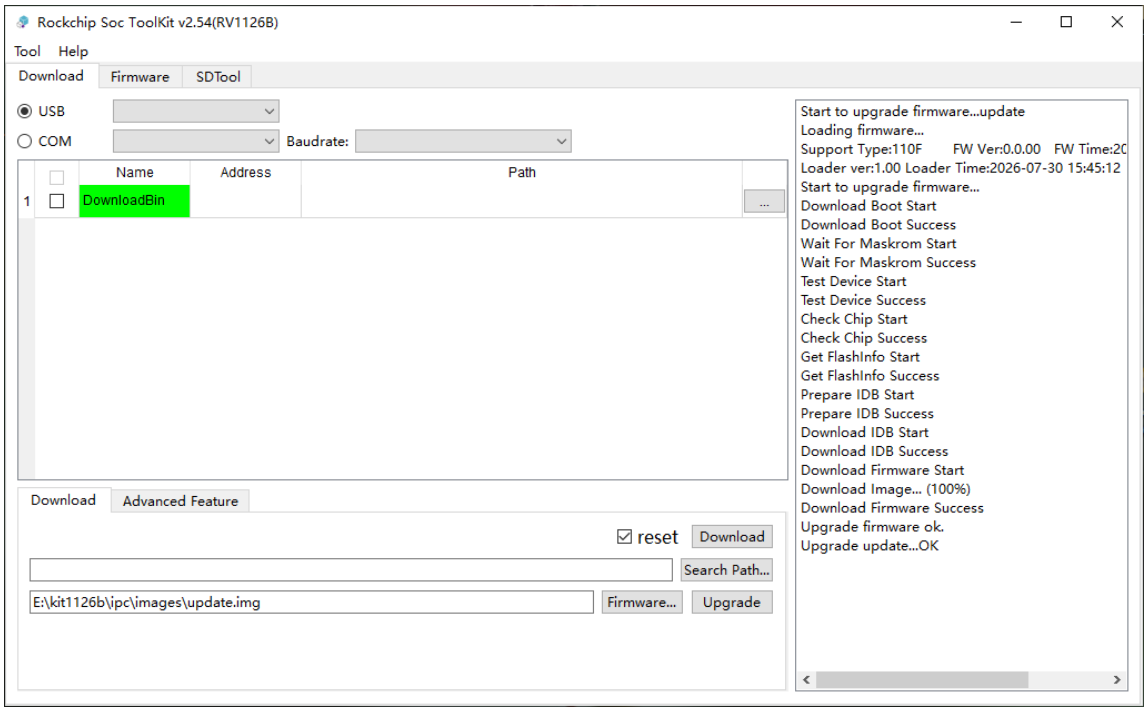
Step 2: Put the board into MaskRom mode. For details, refer to Section 3.1.1 How to Enter MaskRom Mode.



Step 3: Click **Firmware**, select **update.img**, and then click **Upgrade** to start flashing.

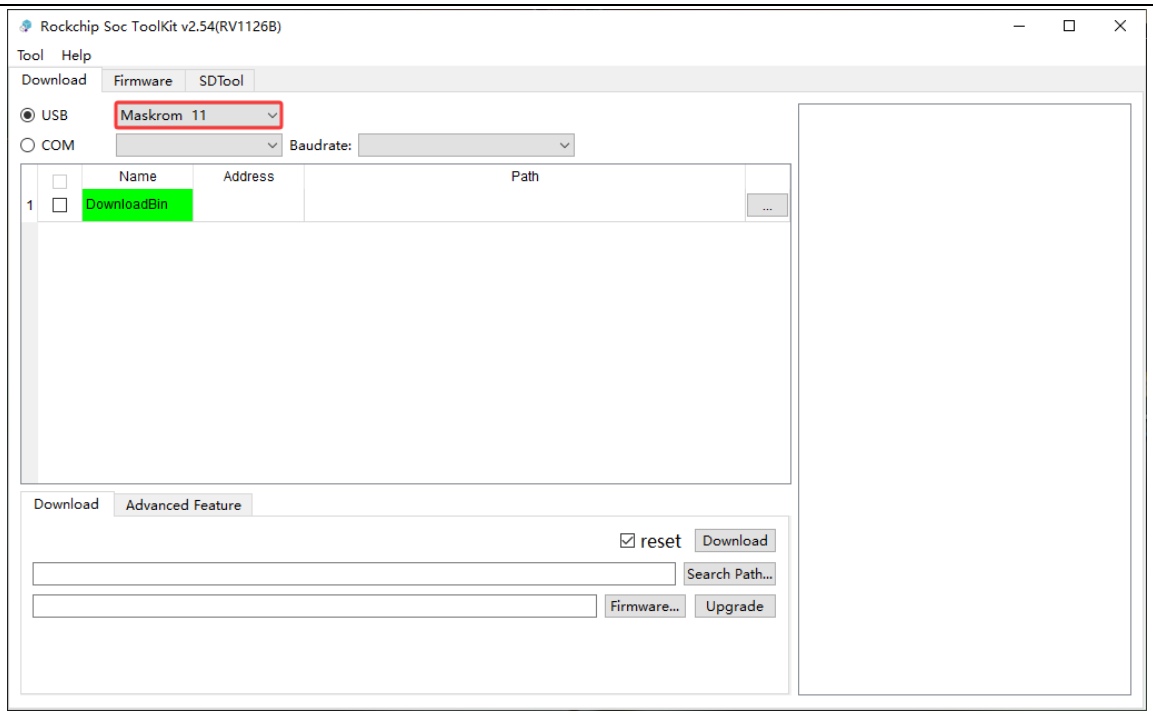


After the flashing is complete, the board will automatically reboot.

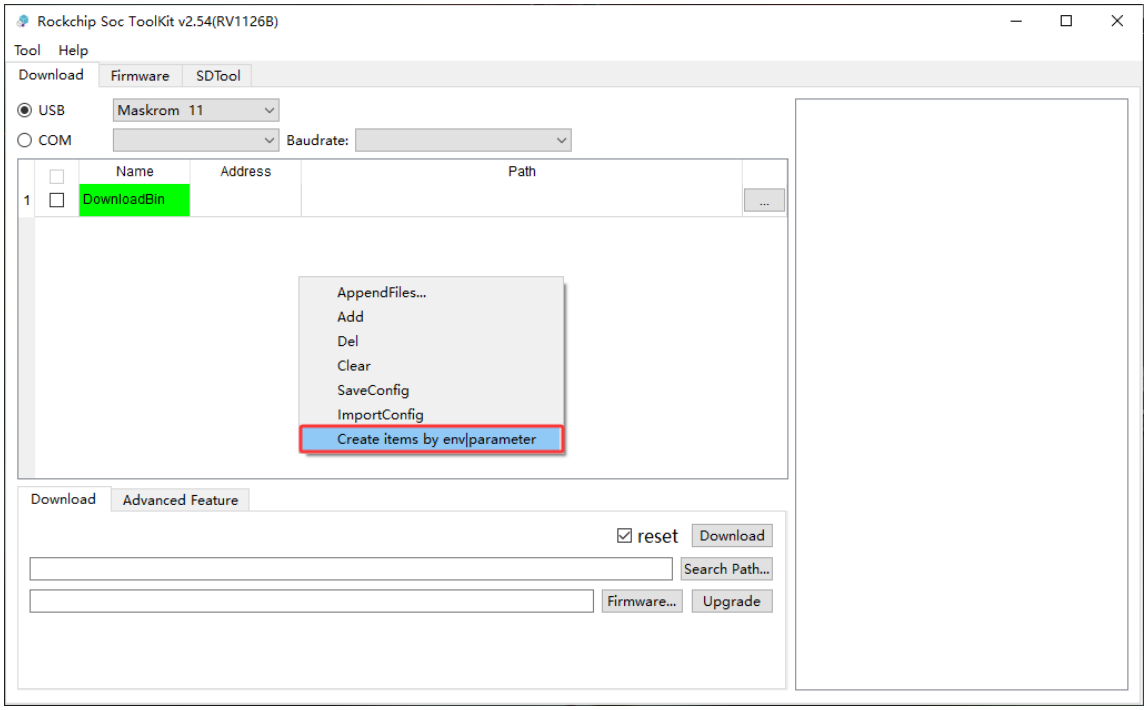


3.2.2 Flash Separate Partition Images

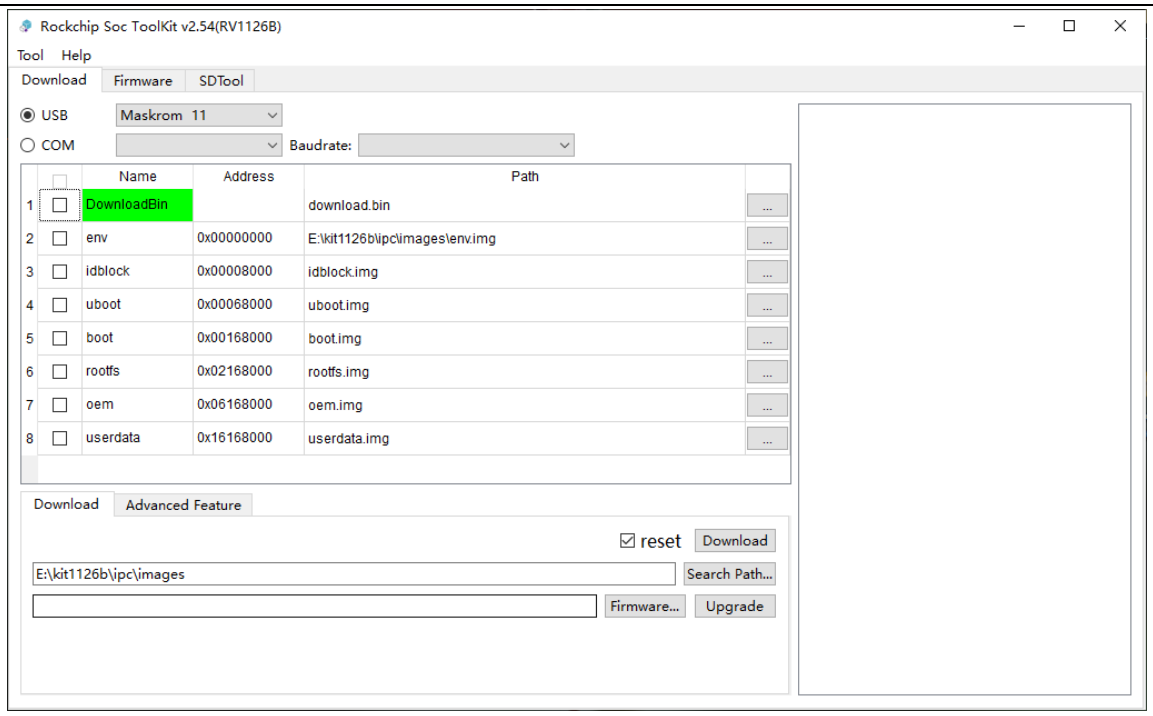
Step 1: Put the board into MaskRom mode. For details, refer to Section 3.1.1 How to Enter MaskRom Mode.



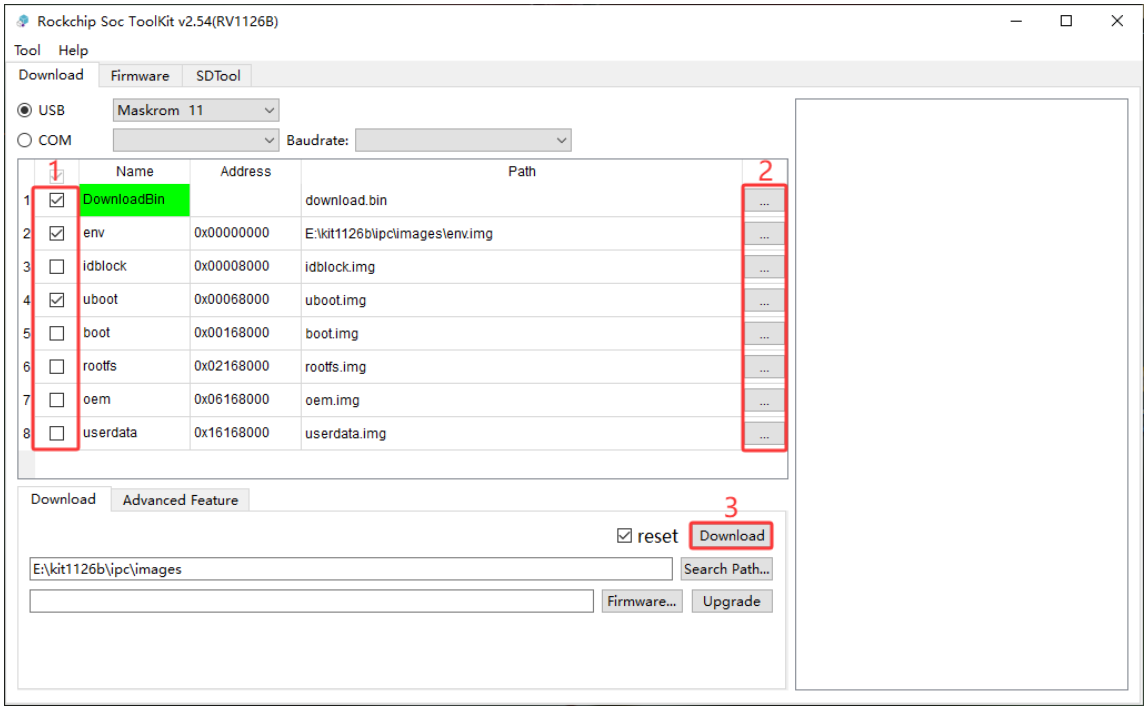
Step 2: Right-click the blank area in the image list and select **Create items by env|parameter**.



In the file selection window, select the **env.img** file from the separate image firmware directory. SocToolKit will automatically generate the flashing items and load the corresponding partition names, addresses, and image paths.

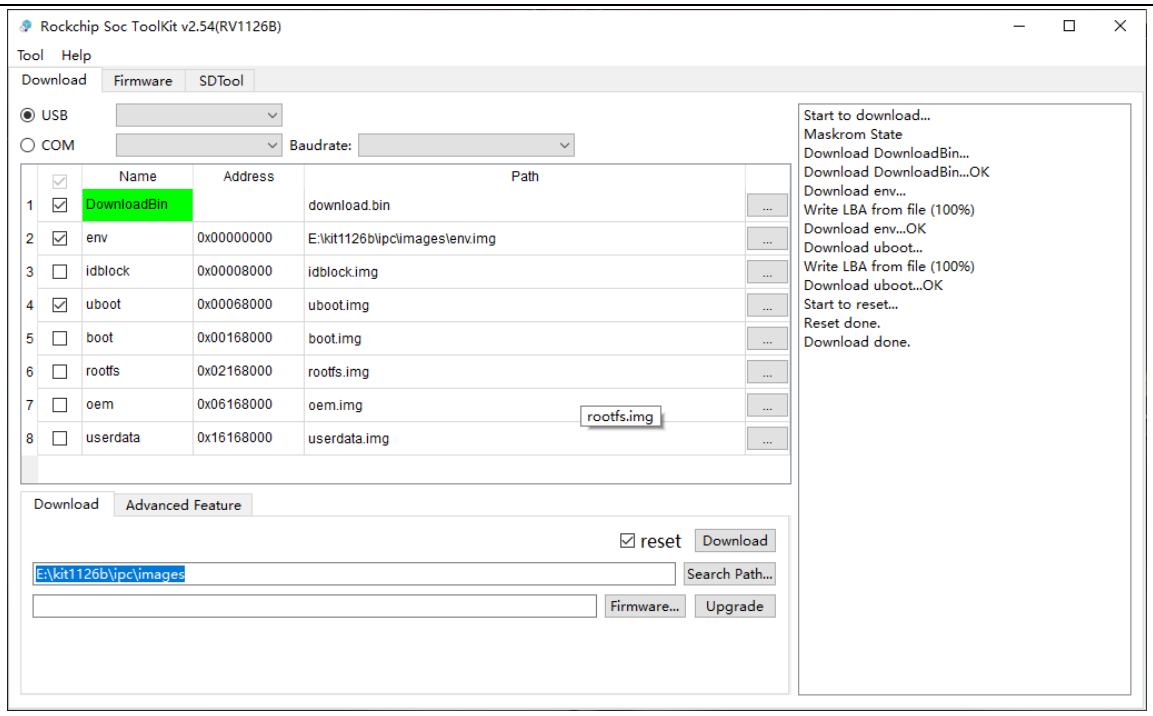


Step 3: Select the required image items in the list, click "**...**" to load the corresponding image files, and then click "**Download**" to start flashing.



Note: When flashing separate images, make sure that "**DownloadBin**" is selected. Otherwise, the flashing process may fail.

After flashing is complete, the board will automatically reboot.


Rockchip Soc ToolKit v2.54(RV1126B)
Tool
Help
Download
Firmware
SDTool
USB
COM
Baudrate:
Name
Address
Path
1 DownloadBin download.bin
2 env 0x00000000 E:\kit1126b\ipc\images\env.img
3 idblock 0x00008000 idblock.img
4 uboot 0x00068000 uboot.img
5 boot 0x00168000 boot.img
6 rootfs 0x02168000 rootfs.img
7 oem 0x06168000 oem.img
8 userdata 0x16168000 userdata.img
rootfs.img
Download
Advanced Feature
reset
Download
E:\kit1126b\ipc\images
Search Path...
Firmware...
Upgrade
Start to download...
Maskrom State
Download DownloadBin...
Download DownloadBin...OK
Download env...
Write LBA from file (100%)
Download env...OK
Download uboot...
Write LBA from file (100%)
Download uboot...OK
Start to reset...
Reset done.
Download done.

4. Development Environment

4.1 Preparing the Development Environment

Ubuntu 22.04 or later is recommended for SDK compilation. If an error occurs during compilation, check the error message and install the required software packages accordingly. For other Linux distributions, the package names or installation commands may need to be adjusted.

In addition to the operating system requirements, the following hardware and software requirements should be met:

Hardware requirements	Software requirements
64-bit system, hard disk space should be greater than 200G. If you do multiple builds, you will need more hard drive space.	Ubuntu 22.04

4.2 Installing Libraries and Toolkits

This section provides the commands for installing the software packages required to build the SDK compilation environment. Other tools, such as Samba and SSH, should be installed as needed.

PC OS	Network	Permission
Ubuntu 22.04	online	root

To install the required tools, execute the following commands:

```
$ sudo apt-get install git ssh make gcc libssl-dev liblz4-tool libmpc-dev
$ sudo apt-get install expect g++ patchelf chrpath gawk texinfo chrpath diffstat
$ sudo apt-get install binfmt-support live-build bison flex fakeroot libgmp-dev
$ sudo apt-get install cmake gcc-multilib g++-multilib unzip device-tree-compiler
$ sudo apt-get install ncurses-dev libgucharmap-2-90-dev bzip2 expat gpgv2
$ sudo apt-get install cpp-aarch64-linux-gnu g++-aarch64-linux-gnu
$ sudo apt install python2 python-is-python3
```

5. Compile Source Code

5.1 Extract Source Code

To extract the source code package, execute the following commands:

```
$ tar xvf RV1126B_Linux_IPC*.tar.bz2
$ cd RV1126B_Linux_IPC_SDK
```

5.2 Configure the Target Board

To configure the target board, execute the following command:

```
$ ./build.sh lunch
```

When the BoardConfig list is displayed, select option **3**:

3. BoardConfig_IPC/BoardConfig-EMMC-NONE-Boardcon-KIT1126B_FASTBOOT.mk

```
You're building on Linux
Lunch menu...pick a combo:
BoardConfig-*.mk naming rules:
BoardConfig-"启动介质"-"电源方案"-"硬件版本"-"应用场景".mk
BoardConfig-"boot medium"-"power solution"-"hardware version"-"application".mk
------------------------------------------------------------
0. BoardConfig_BatteryIPC/BoardConfig-EMMC-NONE-RV1126B_EVB2_V12-BAT_IPC.mk
                           boot medium(启动介质): EMMC
                         power solution(电源方案): NONE
                       hardware version(硬件版本): RV1126B_EVB2_V12
                           application(应用场景): BAT_IPC
------------------------------------------------------------
------------------------------------------------------------
1. BoardConfig_CVR/BoardConfig-EMMC-RK801-RV1126B_EVB1_V10-CVR.mk
                           boot medium(启动介质): EMMC
                         power solution(电源方案): RK801
                       hardware version(硬件版本): RV1126B_EVB1_V10
                           application(应用场景): CVR
------------------------------------------------------------
------------------------------------------------------------
2. BoardConfig_DV/BoardConfig-EMMC-RK801-RV1126B_EVB1_V12-DV_64bit.mk
                           boot medium(启动介质): EMMC
                         power solution(电源方案): RK801
                       hardware version(硬件版本): RV1126B_EVB1_V12
                           application(应用场景): DV_64bit
------------------------------------------------------------
------------------------------------------------------------
3. BoardConfig_IPC/BoardConfig-EMMC-NONE-Boardcon-KIT1126B_FASTBOOT.mk
                           boot medium(启动介质): EMMC
                         power solution(电源方案): NONE
                       hardware version(硬件版本): Boardcon
                           application(应用场景): KIT1126B_FASTBOOT
------------------------------------------------------------
------------------------------------------------------------
4. BoardConfig_IPC/BoardConfig-EMMC-RK801-Boardcon-SBC1126B_FASTBOOT.mk
                           boot medium(启动介质): EMMC
                         power solution(电源方案): RK801
                       hardware version(硬件版本): Boardcon
                           application(应用场景): SBC1126B_FASTBOOT
------------------------------------------------------------
Which would you like? [0]: 3
[build.sh:info] switching to board:
```

5.3 Build the Firmware

The firmware can be built in either of the following ways:

- Build all components with one command.
- Build each component step by step.

5.3.1 Build All Components

To build all components with one command, execute:

```
$ ./build.sh
```

After the command is completed, the image files will be generated in the *output/image/* directory.

5.3.2 Build Components Step by Step

To build each component separately, execute the following commands as needed.

Step 1: Compile U-Boot

To compile U-Boot, execute the following command:

```
$ ./build.sh uboot
```

Step 2: Compile the Kernel

To compile the kernel, execute the following command:

```
$ ./build.sh kernel
```

Step 3: Compile the Sysdrv

To compile the sysdrv, execute the following command:

```
$ ./build.sh sysdrv
```

Step 4: Compile the Root Filesystem

To compile the root filesystem, execute the following command:

```
$ ./build.sh rootfs
```

Step 5: Compile Media

To compile media, execute the following command:

```
$ ./build.sh media
```

Step 6: Compile App

To compile app, execute the following command:

```
$ ./build.sh app
```

Step 7: Generate Firmware Images

After all required components are compiled, execute the following command to generate the firmware images:

```
$ ./build.sh firmware
```

After the command is completed, the image files will be generated in the *output/image/* directory.

5.3.3 Package Firmware Images

To package the firmware images into update.img, execute:

```
$ ./build.sh updateimg
```

After the command is completed, **update.img** will be generated in the *output/image/* directory.

6. Test

In the IPC version, the rkipc service starts automatically after system boot.

To avoid conflicts during function tests, stop the rkipc service before testing other functions.

Stop the rkipc service:

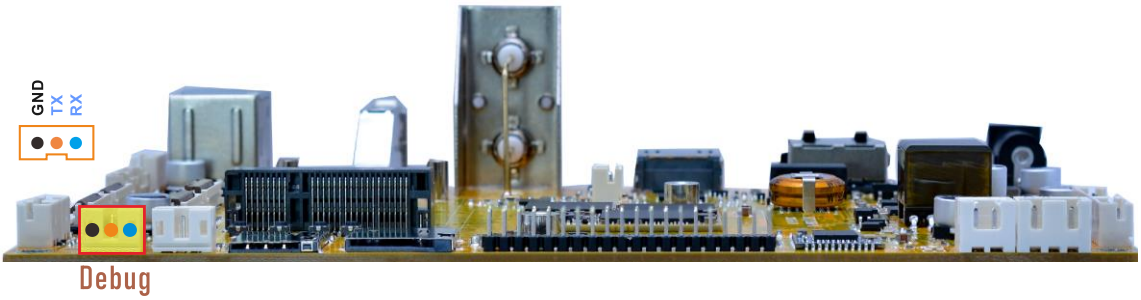
```
# RkLunch-stop.sh
```

Restart the rkipc service after testing:

```
# RkLunch.sh
```

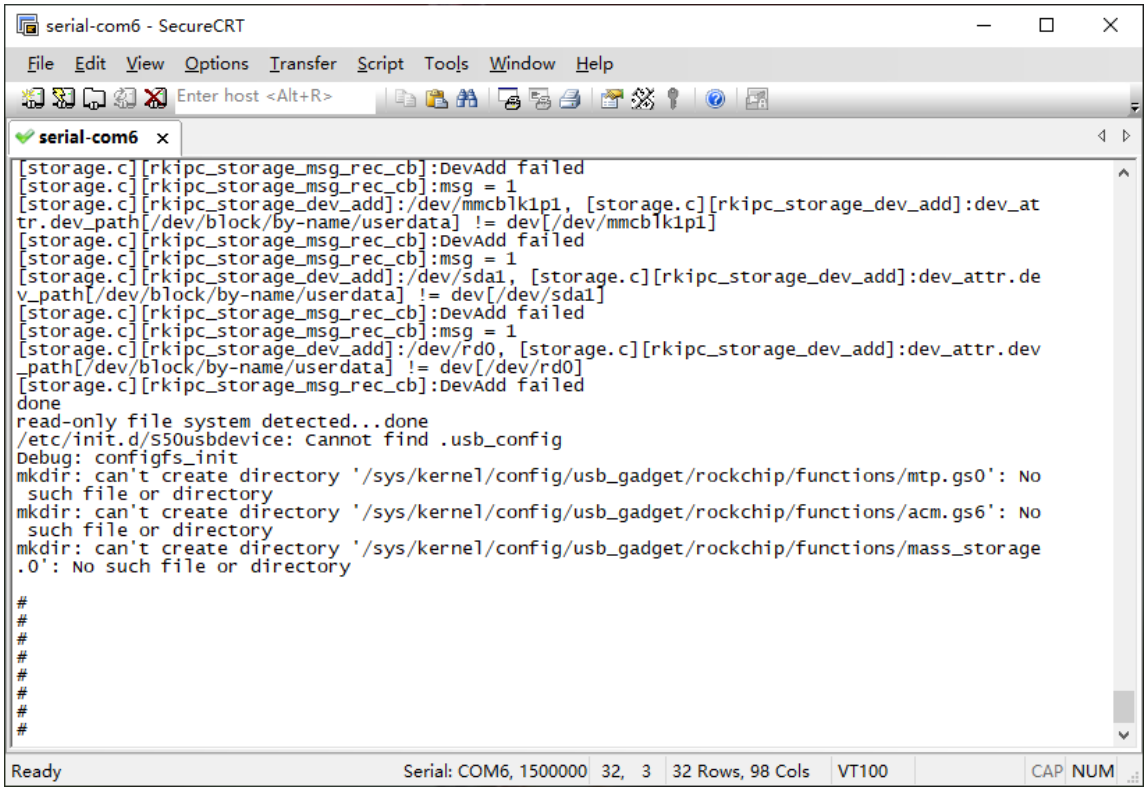
6.1 Serial Terminal

Step 1: Connect the USB-to-serial module to the PC and connect the other end to the Debug UART interface on the board.



Step 2: Open a serial terminal tool on the PC, select the corresponding serial port, and set the baud rate to 1500000.

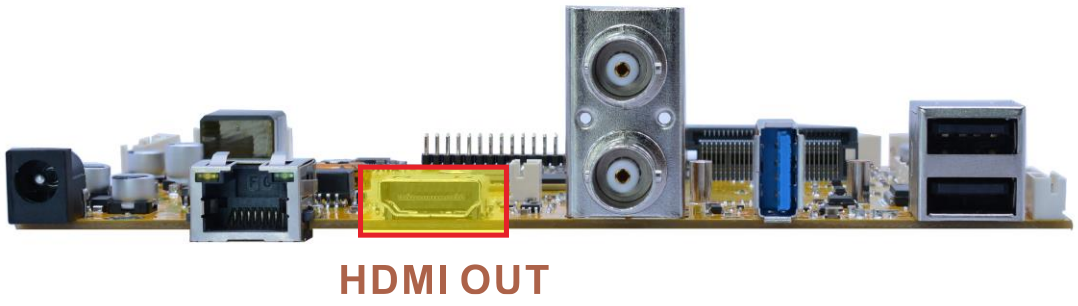
Step 3: Power on the board and check whether the boot log is displayed in the serial terminal.



For details about installing and using the serial terminal tool, please refer to Section 2.2 Install CH9102X Driver and 2.3 Install Serial Terminal Tool.

6.2 Display

The KIT1126B uses an LT9611 MIPI-to-HDMI bridge chip to convert the MIPI DSI display signal to HDMI output. The HDMI interface supports 1920×1080@60Hz video output.

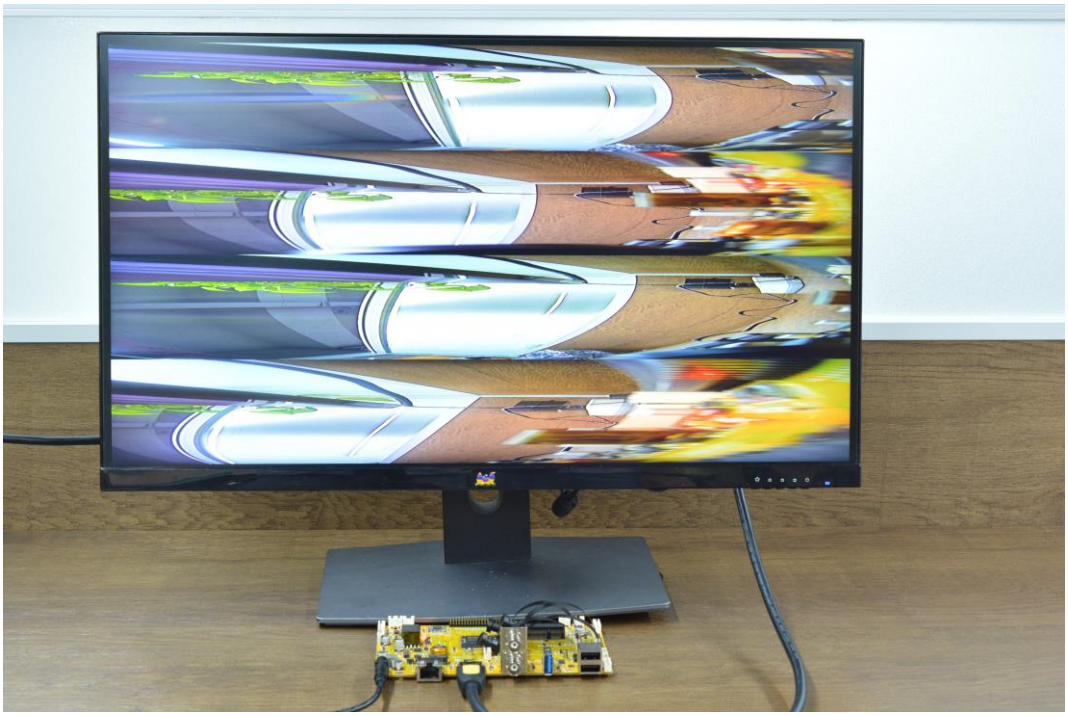


The HDMI status can be checked using the following command:

```
# cat /sys/class/lt9611/hdmi/status
```

After the IPC version system boots, if four MIPI cameras are connected before power-

on, the system automatically displays the combined preview image from the four MIPI camera channels.



If the four MIPI cameras are not connected, the desktop UI will not be displayed after boot-up.

6.3 USB

6.3.1 USB OTG

The USB OTG port of the KIT1126B is configured as Device mode by default after system boot.

• To switch the USB OTG port to Host mode, execute the following command:

```
# echo host > /sys/kernel/debug/usb/21500000.usb/mode
```

```
# echo host > /sys/kernel/debug/usb/21500000.usb/mode
# cat /sys/kernel/debug/usb/21500000.usb/mode
host
```

• To switch the USB OTG port back to Device mode, execute the following command:

```
# echo device > /sys/kernel/debug/usb/21500000.usb/mode
# usbdevice start
# /etc/init.d/S50usbdevice restart
```

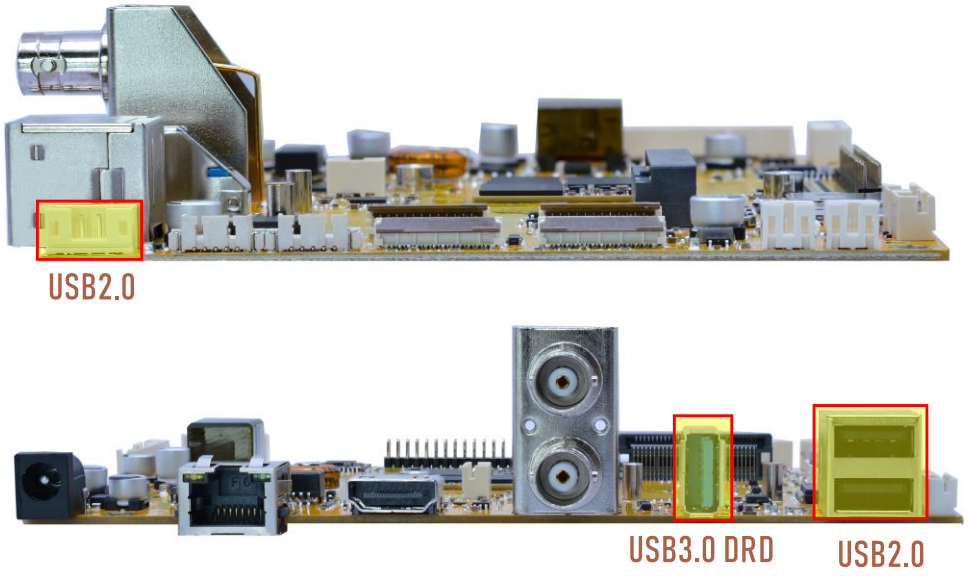
```
# echo device > /sys/kernel/debug/usb/21500000.usb/mode
# usbdevice start
# /etc/init.d/S50usbdevice restart
sh: write error: No such device
mkdir: can't create directory '/dev/usb-ffs/adb': File exists
mount: mounting adb on /dev/usb-ffs/adb failed: Device or resource busy
#
# cat /sys/kernel/debug/usb/21500000.usb/mode
device
```

6.3.2 USB HOST

The KIT1126B provides USB 2.0 Host interfaces and one USB 3.0 DRD interface. The USB Host interfaces can be used to connect USB peripherals, such as a USB mouse, USB keyboard, USB flash drive, and other USB devices.

Note:

The USB 3.0 interface is an OTG interface. To use it as a Host interface, switch it to Host mode first. For details, refer to Section 6.3.1 USB OTG.



The current USB connection speed can be checked from the debug log.

• When the USB device operates in USB 2.0 mode, the log usually shows “high-speed”:

```
# dmesg | grep -Ei "high-speed|usb"
[    3.110105] ohci-platform 214c0000.usb: Generic Platform OHCI controller
[    3.110178] ohci-platform 214c0000.usb: new USB bus registered, assigned bus number 1
[    3.110358] ohci-platform 214c0000.usb: irq 116, io mem 0x214c0000
[    3.174602] hub 1-0:1.0: USB hub found
[    3.392445] usbcore: registered new interface driver usbhid
[    3.392466] usbhid: USB HID core driver
[    3.425906] ehci-platform 21480000.usb: EHCI Host Controller
[    3.425942] ehci-platform 21480000.usb: new USB bus registered, assigned bus number 2
[    3.426058] ehci-platform 21480000.usb: irq 118, io mem 0x21480000
[    3.437521] phy phy-21400000.usb2-phy.3: charger = USB_SDP_CHARGER
[    3.473537] usb 1-1: new full-speed USB device number 2 using ohci-platform
[    3.489543] ehci-platform 21480000.usb: USB 2.0 started, EHCI 1.00
[    3.490869] hub 2-0:1.0: USB hub found
[    3.595577] usbcore: registered new interface driver usb-storage
[    3.745575] usb 2-1: new high-speed USB device number 2 using ehci-platform
[    3.903353] hub 2-1:1.0: USB hub found
[    4.197594] usb 2-1.1: new low-speed USB device number 3 using ehci-platform
[    4.317694] input: USB Optical Mouse  as /devices/platform/21480000.usb/usb2/2-1/2-1.1/2-
1.1:1.0/0003:30FA:0400.0001/input/input3
[    4.377610] hid-generic 0003:30FA:0400.0001: input: USB HID v1.11 Mouse [USB Optical Mouse ] on usb-
21480000.usb-1.1/input0
[   14.443334] dwc3 21500000.usb: device reset
[   14.530268] android_work: sent uevent USB_STATE=CONNECTED
[   14.563064] android_work: sent uevent USB_STATE=CONFIGURED
[  557.901703] usb 2-1.2: new high-speed USB device number 4 using ehci-platform
[  558.045996] usb-storage 2-1.2:1.0: USB Mass Storage device detected
[  558.047432] scsi host0: usb-storage 2-1.2:1.0
```

• When the USB device operates in USB 3.0 mode, the log usually shows “SuperSpeed”:

```
# echo host > /sys/kernel/debug/usb/21500000.usb/mode
# dmesg | grep -Ei "SuperSpeed|xhci"
[   35.591003] xhci-hcd xhci-hcd.2.auto: xHCI Host Controller
[   35.591037] xhci-hcd xhci-hcd.2.auto: new USB bus registered, assigned bus number 3
[   35.591221] xhci-hcd xhci-hcd.2.auto: hcc params 0x0220fe64 hci version 0x110 quirks 0x0004008022010010
[   35.591284] xhci-hcd xhci-hcd.2.auto: irq 116, io mem 0x21500000
[   35.591482] xhci-hcd xhci-hcd.2.auto: xHCI Host Controller
[   35.591506] xhci-hcd xhci-hcd.2.auto: new USB bus registered, assigned bus number 4
[   35.591531] xhci-hcd xhci-hcd.2.auto: Host supports USB 3.0 SuperSpeed
[   35.994819] usb 4-1: new SuperSpeed USB device number 2 using xhci-hcd
```

After a USB flash drive is connected, it will be mounted automatically. Run the

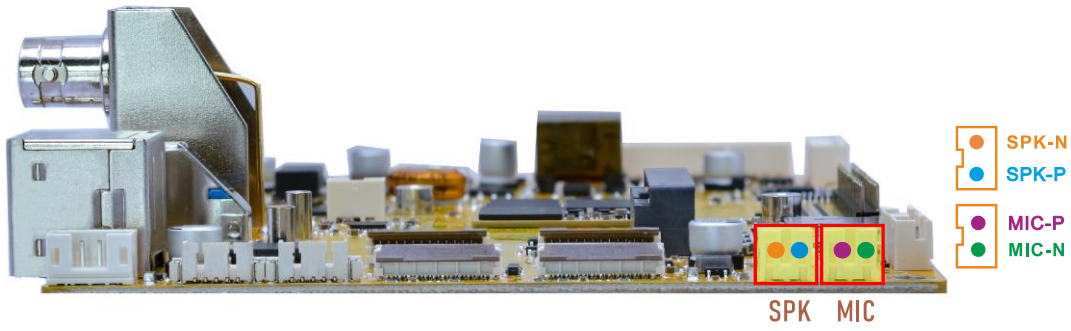
following command to check the mount path of the USB flash drive:

```
# df -h
```

```
# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                48.0M     48.0M         0 100% /
devtmpfs                 60.4M         0     60.4M   0% /dev
tmpfs                   966.9M         0    966.9M   0% /dev/shm
tmpfs                   966.9M     16.0K    966.9M   0% /tmp
tmpfs                   966.9M    732.0K    966.2M   0% /run
/dev/block/by-name/oem
                        237.9M     54.7M    169.3M  24% /oem
/dev/block/by-name/userdata
                          5.8G    320.0K      5.7G   0% /userdata
/dev/mmcblk1p1           59.5G     12.1G     47.4G  20% /mnt/sdcard
/dev/sda1                58.0G     45.2G     12.8G  78% /tmp/sda1
```

6.4 Audio I/O

Connect the speaker and microphone to the corresponding interfaces on the KIT1126B.



6.4.2 Audio Input Test

Step 1: Stop the rkipc service.

Execute the following command:

```
# RkLunch-stop.sh
```

```
# RkLunch-stop.sh
Stop Application ...
[rkipc.c][sig_proc]:received signo 15
  344 root     2776m S<   rkipc -a /oem/usr/share/iqfiles
rkipc active
[server.c][rkipc_server_deinit]:rkipc_server_deinit failed
[osd.c][osd_time_server]:exit
[video.c][rkipc_osd_bmp_destroy]:RK_MPI_RGN_Destroy [2] failed with 0xa0038005
[video.c][rkipc_osd_bmp_destroy]:RK_MPI_RGN_Destroy [3] failed with 0xa0038005
[video.c][rkipc_osd_cover_destroy]:RK_MPI_RGN_Destroy [4] failed with 0xa0038005
[video.c][rkipc osd cover destroy]:RK MPI RGN Destroy [11] failed with 0xa0038005
```

Step 2: Configure Audio Input Gain.

Set the audio input gain using the following command:

```
# rk_mpi_amix_test -C 0 --control 'ACodec Digital Gain Volume' --value '100'
```

```
# rk_mpi_amix_test -C 0 --control 'ACodec Digital Gain Volume' --value '100'
cmd parse result:
sound control id     : 0
control name         : ACodec Digital Gain Volume
control value        : 100
list controls        : 0
list contents        : 0
```

Step 3: Record Audio.

Execute the following command to start recording:

```
# rk_mpi_ai_test --device_rate 44100 --device_ch 2 --out_rate 44100 --out_ch 2 -o
/oem --sound_card_name hw:0,0
```

```
# rk_mpi_ai_test --device_rate 44100 --device_ch 2 --out_rate 44100 --out_ch 2 -
o /oem --sound_card_name hw:0,0
cmd parse result:
input  file name     : (null)
output file name     : /oem
loop count          : 1
channel number       : 1
open sound rate      : 44100
record data rate     : 44100
sound card channel    : 2
output channel       : 2
bit_width           : 16
frame_number         : 4
frame_length         : 1024
sound card name      : hw:0,0
device id           : 0
set volume curve     : 0
set volume          : 100
set mute            : 0
set track_mode       : 0
get volume          : 0
get mute            : 0
get track_mode       : 0
data read enable     : 0
aed enable          : 0
bcd enable          : 0
buz enable          : 0
vqe gap duration (ms) : 16
vqe enable          : 0
get vqe result       : 0
bcd NN model path    : (null)
vqe config file      : (null)
dump algo pcm data    : 0
dev queue len        : 0
sed queue len        : 0
fd get enable        : 1
rockit_load start
v4l2_tx probe successv4l2_rx_probe success
rockit_load end
rockit log path (null), log_size = 0
log_file = (nil)
(null)          15:36:07-526 {log_level_init    :209}

 please use echo name=level > /tmp/rt_log_level set log level
       name: all cmpi mb sys vdec venc rgn vpss vgs tde avs wbc vo vi ai ao aenc adec
       log_level: 0 1 2 3 4 5 6

rockit default level 4, can use export rt_log_level=x, x=0,1,2,3,4,5,6 change
```

6.4.1 Audio Output Test

Step 1: Configure Audio Output Volume.

Configure the speaker output volume using the following command:

```
# rk_mpi_amix_test -C 0 --control 'DAC Digital Volume' --value '250,250'
```

```
# rk_mpi_amix_test -C 0 --control 'DAC Digital Volume' --value '250,250'
cmd parse result:
sound control id     : 0
control name         : DAC Digital Volume
control value        : 250,250
list controls        : 0
list contents        : 0
```

Step 2: Play the audio file.

Execute the following command to play the audio file:

```
# rk_mpi_ao_test -i /oem/cap_out.pcm --input_ch 2 --input_rate 44100 --
device_rate 44100 --device_ch 2 --sound_card_name hw:0,0
```

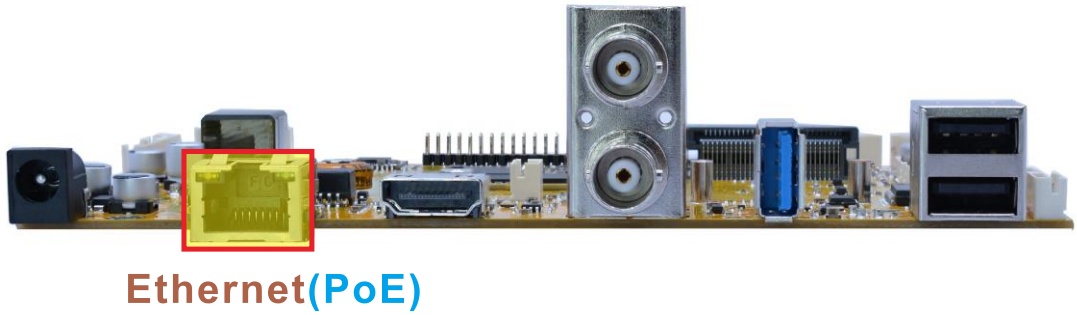
```
# rk_mpi_ao_test -i /oem/cap_out.pcm --input_ch 2 --input_rate 44100 --device_ra
te 44100 --device_ch 2 --sound_card_name hw:0,0
cmd parse result:
input  file name     : /oem/cap_out.pcm
output file name     : (null)
loop count           : 1
channel number       : 1
open sound rate      : 44100
open sound channel    : 2
input stream rate    : 44100
input channel        : 2
bit_width            : 16
period_count         : 4
period_size          : 4096
sound card name      : hw:0,0
device id            : 0
set volume curve     : 0
set volume           : 100
set mute             : 0
set track_mode       : 0
get volume           : 0
get mute             : 0
get track_mode       : 0
query stat           : 0
pause and resume chn  : 0
save file            : 0
query save file stat  : 0
clear buf            : 0
get attribute        : 0
clear attribute      : 0
set loopback mode    : 0
vqe enable           : 0
rockit_load start
v4l2_tx probe successv4l2_rx_probe success
rockit_load end
rockit log path (null), log_size = 0
log_file = (nil)
(null)         15:41:17-011 {log_level_init    :209}

 please use echo name=level > /tmp/rt_log_level set log level
       name: all cmpi mb sys vdec venc rgn vpss vgs tde avs wbc vo vi ai ao aenc adec
       log_level: 0 1 2 3 4 5 6
```

Note: The following commands can be used to select different audio output devices.

```
• --sound_card_name hw:0,0  # Speaker audio output
• --sound_card_name hw:1,0  # HDMI audio output
```

6.5 Ethernet

The KIT1126B provides a 100M Ethernet RJ45 interface with PoE support, allowing both data transmission and power supply through a single Ethernet cable.



Step 1: Connect the network cable to the Ethernet port.

According to the log, it can be seen that the 100M Ethernet recognition is successful.

```
# dmesg | grep -Ei "link|100Mbps"
[    0.284307] NET: Registered PF_NETLINK/PF_ROUTE protocol family
[    1.413451] mpp_rkvdec2 22140100.rkvdec: link mode probe finish
[    1.420173] can: netlink gateway - max_hops=1
[    3.188508] rk_gmac-dwmac 21c70000.ethernet eth0: configuring for phy/rmii link mode
[    5.061192] dw-mipi-dsi-rockchip 22120000.dsi: [drm:dw_mipi_dsi_bridge_atomic_enable] final DSI-Link
bandwidth: 996 x 4 Mbps
[    5.244812] rk_gmac-dwmac 21c70000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
[    5.245030] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

Step 2: Check the network interface information.

Execute the following command:

```
# ifconfig eth0
```

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 0E:2A:C5:09:62:B0
          inet addr:192.168.0.188  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::c2a:c5ff:fe09:62b0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:448 errors:0 dropped:60 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:38540 (37.6 KiB)  TX bytes:2296 (2.2 KiB)
          Interrupt:79
```

Step 3: Test the network connection.

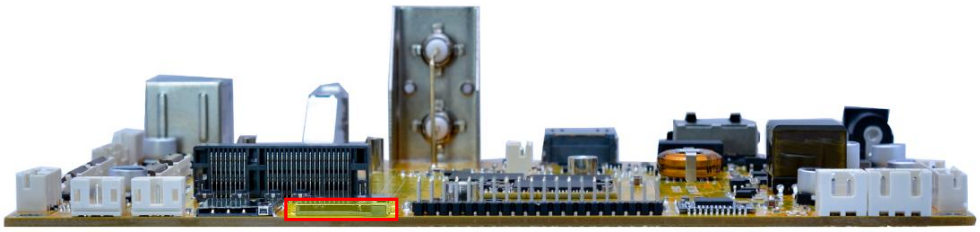
Execute the following command:

```
# ping -I eth0 www.armdesigner.com
```

```
# ping -I eth0 www.armdesigner.com
PING www.armdesigner.com (13.32.99.68): 56 data bytes
64 bytes from 13.32.99.68: seq=0 ttl=246 time=183.090 ms
64 bytes from 13.32.99.68: seq=1 ttl=246 time=182.210 ms
64 bytes from 13.32.99.68: seq=2 ttl=246 time=181.958 ms
64 bytes from 13.32.99.68: seq=3 ttl=246 time=182.285 ms
64 bytes from 13.32.99.68: seq=4 ttl=246 time=181.792 ms
^C
--- www.armdesigner.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 181.792/182.267/183.090 ms
```

6.6 SD Card

Step 1: Insert the micro SD card into the card slot.



Micro SD

After the SD card is recognized, it will be mounted automatically.

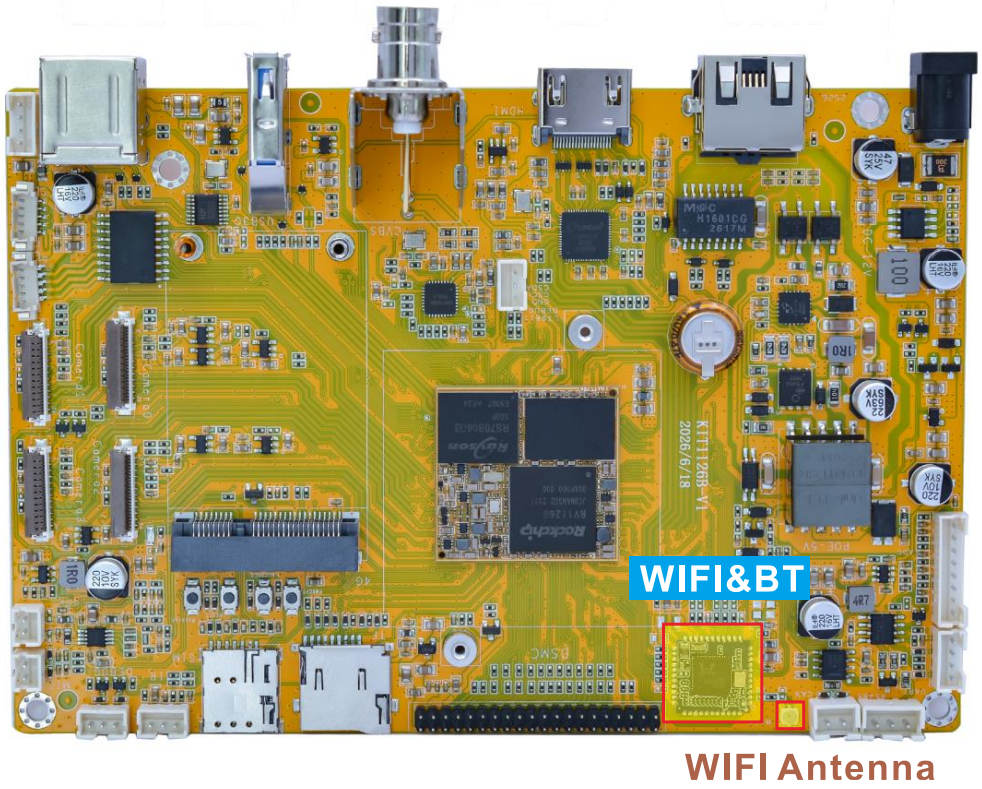
Step 2: Execute the following command to check the mount path of the SD card.

```
# df -h
```

```
# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                48.0M     48.0M         0 100% /
devtmpfs                 60.4M         0     60.4M   0% /dev
tmpfs                   966.9M         0    966.9M   0% /dev/shm
tmpfs                   966.9M     16.0K    966.9M   0% /tmp
tmpfs                   966.9M    728.0K    966.2M   0% /run
/dev/block/by-name/oem
                        237.9M     55.7M    166.8M  25% /oem
/dev/block/by-name/userdata
                          5.8G    328.0K      5.7G   0% /userdata
/dev/mmcblk1p1           59.5G     12.1G     47.4G  20% /mnt/sdcard
/dev/sda1                58.0G     45.2G     12.8G  78% /tmp/sda1
```

6.7 WiFi & Bluetooth

To use the WiFi and Bluetooth functions properly, make sure the antenna is connected.



6.7.1 WiFi

Step 1: Check the Wi-Fi device information.

Execute the following command:

```
# ifconfig wlan0
```

```
# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 14:0A:02:C9:0B:71
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Step 2: Scan for available WiFi hotspots.

Execute the following command:

```
# wpa_cli -i wlan0 scan
# wpa_cli -i wlan0 scan_results
```

```
# wpa_cli -i wlan0 scan
OK
# [network.c][rk_network_get_cable_state]:read Cable state
[network.c][rk_network_get_cable_state]:
[wlan0] link down
# wpa_cli -i wlan0 scan_results
bssid / frequency / signal level / flags / ssid
c8:3a:35:51:1e:98       2442    -47     [WPA-PSK-CCMP][ESS]     Tenda_511E98
b4:f1:8c:6d:d1:24       2437    -49     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    Boardcon
b4:f1:8c:fd:d1:29       2437    -49     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    Boardcon_Wi-Fi5
b2:22:7a:5a:b6:4a       2437    -53     [WPA2-PSK-CCMP][ESS]    DIRECT-4A-HP Laser 136w
38:20:28:50:1e:a1       2437    -65     [WPA2-PSK-CCMP][ESS]    Casvi
9c:a6:15:10:e5:57       2437    -71     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    TP
9c:a6:15:10:da:c5       2462    -73     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    TP
38:20:28:50:0f:81       2412    -70     [WPA2-PSK-CCMP][ESS]    Casvi
b4:f1:8c:6d:d1:25       2437    -49     [ESS]
```

Step 3: Connect to the WiFi hotspot.

Execute the following command:

```
# rkwifi_server connect ssid password
```

```
# rkwifi_server connect Boardcon Boardcon43435656
rkwifi cmd: connect Boardcon Boardcon43435656
cmd_line: connect Boardcon Boardcon43435656 [33]
rk_cmd_parse connect Boardcon Boardcon43435656
rk_tokenize_cmd connect Boardcon Boardcon43435656
OK
[connect 3]: connect Boardcon
rk_wifi_connect: ssid: Boardcon, psk: Boardcon43435656
[1785773185:252666][src/Rk_wifi.c][wpa_wifi_connect1][l:1851],[wpa_wifi_connect1] ssid: Boardcon, psk:
Boardcon43435656

[1785773185:252709][src/Rk_wifi.c][_RK_wifi_running_getConnectionInfo][l:710],_RK_wifi_running_getConnecti
onInfo: enter
```

Step 4: Check the network interface status.

Execute the following command:

```
# ifconfig wlan0
```

```
# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 14:0A:02:C9:0B:71
          inet addr:192.168.0.137  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::160a:2ff:fec9:b71/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:237 errors:0 dropped:35 overruns:0 frame:0
          TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:29090 (28.4 KiB)  TX bytes:4802 (4.6 KiB)
```

Step 5: Test the WiFi network connection.

Execute the following command:

```
# ping -I wlan0 www.armdesigner.com
```

```
# ping -I wlan0 www.armdesigner.com
PING www.armdesigner.com (108.138.246.87): 56 data bytes
64 bytes from 108.138.246.87: seq=0 ttl=246 time=320.862 ms
64 bytes from 108.138.246.87: seq=1 ttl=246 time=188.781 ms
64 bytes from 108.138.246.87: seq=2 ttl=246 time=180.639 ms
64 bytes from 108.138.246.87: seq=3 ttl=246 time=200.371 ms
64 bytes from 108.138.246.87: seq=4 ttl=246 time=232.152 ms
^C
--- www.armdesigner.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 180.639/224.561/320.862 ms
```

6.7.2 Bluetooth

Bluetooth is configured to work as a Bluetooth speaker by default.

Step 1: Start the BlueALSA A2DP sink service.

Execute the following command:

```
# bluealsa -p a2dp-sink &
```

```
# bluealsa -p a2dp-sink &
# bluealsa: W: Couldn't create storage directory: No such file or directory
bluealsa: E: Couldn't acquire D-Bus name. Please check D-Bus configuration. Requested name: org.bluealsa

[1]+  Done(1)                 bluealsa -p a2dp-sink
```

Step 2: Enter the Bluetooth control tool.

Execute the following command:

```
# bluetoothctl
```

```
# bluetoothctl
Waiting to connect to bluetoothd...Waiting to connect to bluetoothd...[bluetooth]#
hci0 new_settings: powered bondable ssp br/edr le secure-conn
[bluetooth]# [bluetooth]#                     Agent registered
[bluetooth]# [bluetooth]#                     [CHG] Controller 14:0A:02:C9:0B:72 Pairable: yes
[bluetooth]# [bluetooth]#
```

Step 3: Enable Bluetooth and set it to discoverable mode.

Execute the following commands in bluetoothctl:

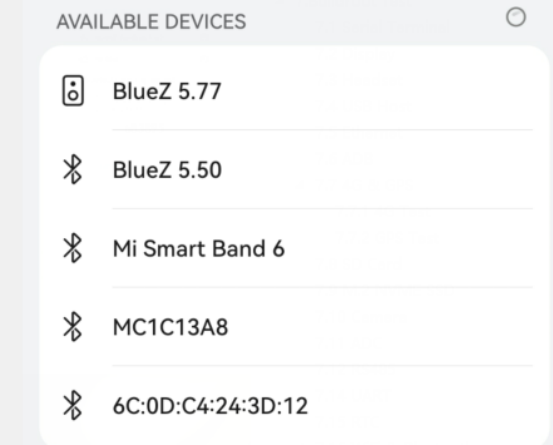
```
[bluetooth]# power on
[bluetooth]# discoverable on
```

```
[bluetooth]# [bluetooth]# power on
[bluetooth]# [bluetooth]#                     Changing power on succeeded
[bluetooth]# [bluetooth]# discoverable on
[bluetooth]# [bluetooth]#                     hci0 new_settings: powered connectable bondable ssp br/edr
le secure-conn
[bluetooth]# [bluetooth]#                     hci0 new_settings: powered connectable discoverable
bondable ssp br/edr le secure-conn
[bluetooth]# [bluetooth]#                     Changing discoverable on succeeded
[bluetooth]# [bluetooth]#                     [CHG] Controller 14:0A:02:C9:0B:72 Discoverable: yes
[bluetooth]# [bluetooth]#
```

After this step, the Bluetooth device can be discovered by the phone.

Step 4: Pair and connect the Bluetooth device.

On the phone, find the Bluetooth device named "**BlueZ 5.77**" in the Bluetooth device list, then tap it to pair and connect.



If a pairing confirmation message appears, confirm it on both the phone and the board.

```
[bluetooth]# [bluetooth]# [liuy]#                    Request confirmation
[liuy]# [liuy]# [agent] Confirm passkey 781208 (yes/no): yes
[liuy]# [liuy]#                  hci0 new_link_key A8:35:12:9A:EB:4D type 0x05 pin_len 0 store_hint 1
[liuy]# [liuy]#                  hci0 device_flags_changed: A8:35:12:9A:EB:4D (BR/EDR)
[liuy]# [liuy]#                       supp: 0x00000000  curr: 0x00000000
[liuy]# [liuy]#                  [CHG] Device A8:35:12:9A:EB:4D Bonded: yes
[liuy]# [liuy]#                  [CHG] Device A8:35:12:9A:EB:4D Modalias: bluetooth:v010Fp107Ed1436
[liuy]# [liuy]#                  [CHG] Device A8:35:12:9A:EB:4D ServicesResolved: yes
[liuy]# [liuy]#                  [CHG] Device A8:35:12:9A:EB:4D Paired: yes
[liuy]# [liuy]#                  hci0 A8:35:12:9A:EB:4D type BR/EDR disconnected with reason 3
[liuy]# [liuy]#                  [CHG] Device A8:35:12:9A:EB:4D ServicesResolved: no
[liuy]# [liuy]# [bluetooth]#                     [CHG] Device A8:35:12:9A:EB:4D Connected: no
[bluetooth]# [bluetooth]#                   hci0 A8:35:12:9A:EB:4D type BR/EDR connected eir_len 11
[bluetooth]# [bluetooth]# [liuy]#                    [CHG] Device A8:35:12:9A:EB:4D Connected: yes
[liuy]# [liuy]#                 Authorize service
[liuy]# [liuy]# [agent] Authorize service 0000110d-0000-1000-8000-00805f9b34fb (yes/no): yes
 [liuy]# [liuy]#                  [CHG] Device A8:35:12:9A:EB:4D UUIDs: 9664aa26-d76c-43ad-9775-
d310f253a408
 [liuy]# [liuy]#                  [CHG] Player /org/bluez/hci0/dev_A8_35_12_9A_EB_4D/player0 Track.Genre:
[liuy]# [liuy]#                  [CHG] Player /org/bluez/hci0/dev_A8_35_12_9A_EB_4D/player0 Position:
0x0001b781 (112513)
[liuy]# [liuy]#                  [CHG] Player /org/bluez/hci0/dev_A8_35_12_9A_EB_4D/player0 Position:
0x0001b781 (112513)
[liuy]# [liuy]#
```

Check the log output to confirm the MAC address. In this example, the MAC address of the mobile phone is **A8:35:12:9A:EB:4D**.

Step 5: Exit bluetoothctl.

Execute the following command:

```
[bluetooth]# exit
```

Step 6: Start Bluetooth audio playback.

Execute the following command:

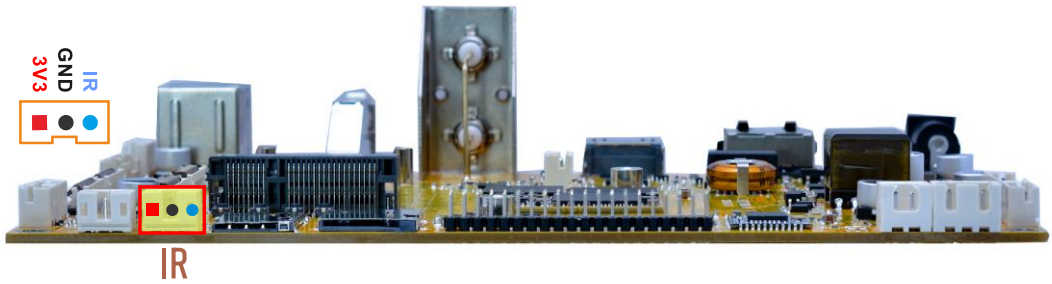
```
# bluealsa-aplay A8:35:12:9A:EB:4D &
```

```
[liuy]# [liuy]# exit
[liuy]# [liuy]#                 #
# bluealsa-aplay A8:35:12:9A:EB:4D &
```

After the Bluetooth connection is established, play audio on the phone. The audio should be output from the board.

6.8 IR

Step 1: Connect the IR receiver.



Step 2: Enable IR debug logs.

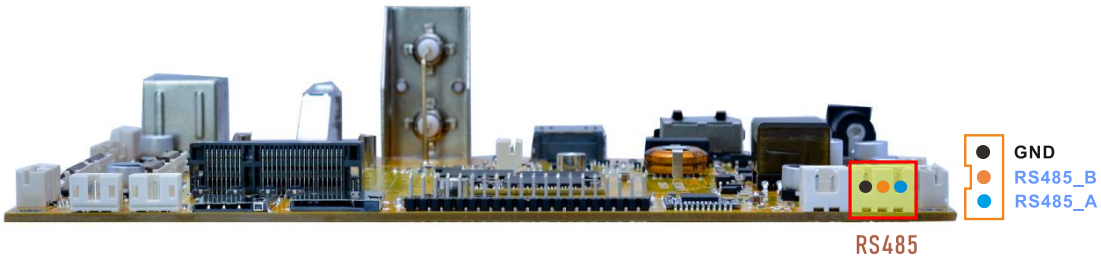
Execute the following command:

```
# echo 1 > /sys/module/rockchip_pwm_remotectl/parameters/code_print
```

Step 3: Point the remote control at the IR receiver and press a button. The corresponding key value will be printed in the log.

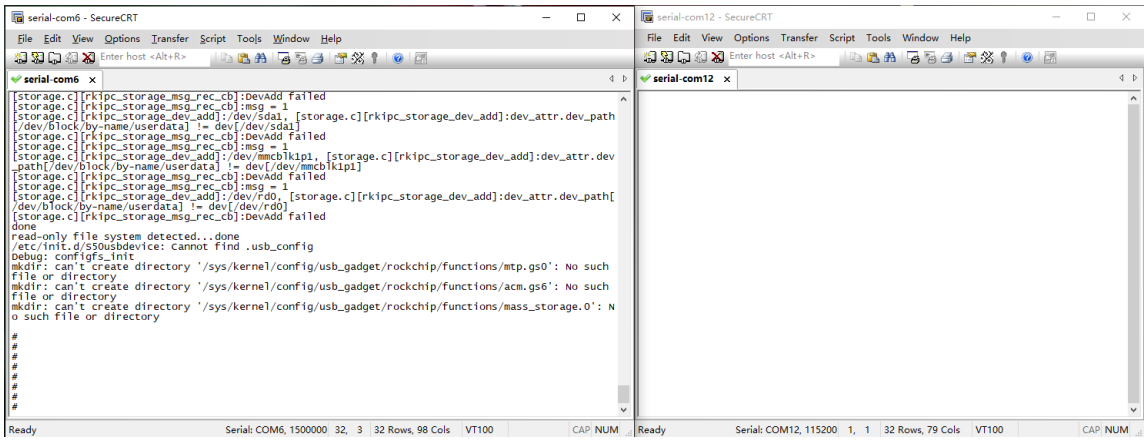
```
# echo 1 > /sys/module/rockchip_pwm_remotectl/parameters/code_print
# dmesg | grep -Ei "USERCODE|RMC_GETDATA"
[ 2086.646835] USERCODE=0x1818
[ 2086.674028] RMC_GETDATA=e4
[ 2086.956856] USERCODE=0x1818
[ 2086.984036] RMC_GETDATA=e6
[ 2087.258583] USERCODE=0x1818
[ 2087.285717] RMC_GETDATA=e5
[ 2087.597475] USERCODE=0x1818
[ 2087.624648] RMC_GETDATA=e7
[ 2087.906789] USERCODE=0x1818
[ 2087.933899] RMC_GETDATA=99
[ 2088.208530] USERCODE=0x1818
[ 2088.235625] RMC_GETDATA=99
[ 2088.531438] USERCODE=0x1818
[ 2088.556328] RMC_GETDATA=34
```

6.9 RS485



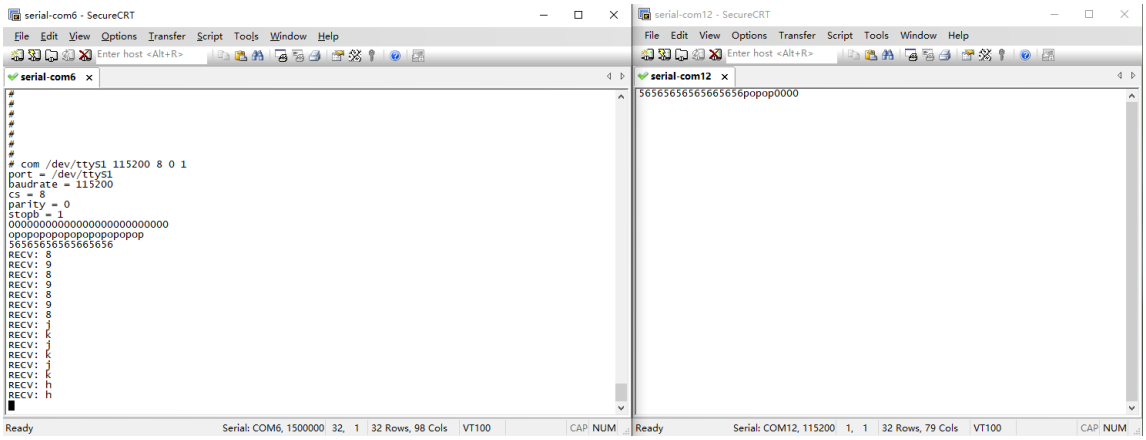
Step 1: Connect the RS485 test tool to the development board as shown in the diagram.

Step 2: Open two serial terminal windows. Set the debug serial port baud rate to 1500000, and set the RS485 serial port baud rate to 115200.



Step 3: Execute the following command on the board to test the RS485 data transmission and reception.

```
# com /dev/ttyS1 115200 8 0 1
```

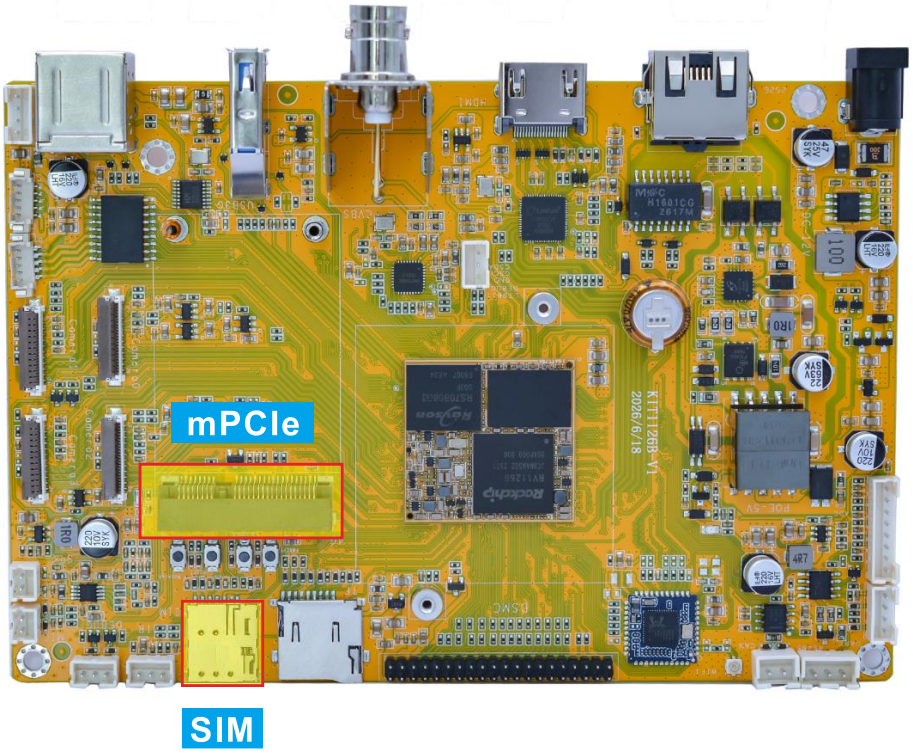


6.10 4G & GPS

Step 1: Install the EC20 or EC25 4G module into the mPCIe socket.

Step 2: Connect the antenna and insert the SIM card.

Step 3: Power on the board.



6.10.1 4G Test

Step 1: Initiate the PPP connection.

Execute the following command:

```
# pppd call quectel-ppp &
```

```
# pppd call quectel-ppp &
# pppd options in effect:
debug          # (from /etc/ppp/peers/quectel-ppp)
nodetach               # (from /etc/ppp/peers/quectel-ppp)
dump           # (from /etc/ppp/peers/quectel-ppp)
noauth         # (from /etc/ppp/peers/quectel-ppp)
user test              # (from /etc/ppp/peers/quectel-ppp)
password ??????        # (from /etc/ppp/peers/quectel-ppp)
remotename 3gppp               # (from /etc/ppp/peers/quectel-ppp)
/dev/ttyUSB3           # (from /etc/ppp/peers/quectel-ppp)
115200         # (from /etc/ppp/peers/quectel-ppp)
lock           # (from /etc/ppp/peers/quectel-ppp)
connect chat -s -v -f /etc/ppp/peers/quectel-chat-connect              # (from /etc/ppp/peers/quectel-ppp)
disconnect chat -s -v -f /etc/ppp/peers/quectel-chat-disconnect        # (from /etc/ppp/peers/quectel-ppp)
nocrtscts              # (from /etc/ppp/peers/quectel-ppp)
modem          # (from /etc/ppp/peers/quectel-ppp)
hide-password          # (from /etc/ppp/peers/quectel-ppp)
novj           # (from /etc/ppp/peers/quectel-ppp)
novjccomp              # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-local              # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-remote             # (from /etc/ppp/peers/quectel-ppp)
ipparam 3gppp          # (from /etc/ppp/peers/quectel-ppp)
noipdefault            # (from /etc/ppp/peers/quectel-ppp)
ipcp-max-failure 30            # (from /etc/ppp/peers/quectel-ppp)
defaultroute           # (from /etc/ppp/peers/quectel-ppp)
usepeerdns             # (from /etc/ppp/peers/quectel-ppp)
noccp          # (from /etc/ppp/peers/quectel-ppp)
abort on (BUSY)
abort on (NO CARRIER)
abort on (NO DIALTONE)
abort on (ERROR)
abort on (NO ANSWER)
timeout set to 30 seconds
send (AT^M)
expect (OK)
AT^M^M
OK
 -- got it

send (ATE0^M)
expect (OK)
^M
ATE0^M^M
OK
 -- got it
```

Step 2: Check the network interface status.

Execute the following command:

```
# ifconfig ppp0
```

```
# ifconfig ppp0
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.3.226.122  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:52 (52.0 B)  TX bytes:198 (198.0 B)
```

Step 3: Test the PPP connection.

Execute the following command:

```
# ping -I ppp0 www.armdesigner.com
```

```
# ping -I ppp0 www.armdesigner.com
PING www.armdesigner.com (108.138.246.80): 56 data bytes
64 bytes from 108.138.246.80: seq=0 ttl=245 time=466.868 ms
64 bytes from 108.138.246.80: seq=1 ttl=245 time=431.883 ms
64 bytes from 108.138.246.80: seq=2 ttl=245 time=375.755 ms
64 bytes from 108.138.246.80: seq=3 ttl=245 time=344.079 ms
64 bytes from 108.138.246.80: seq=4 ttl=245 time=295.851 ms
64 bytes from 108.138.246.80: seq=5 ttl=245 time=264.560 ms
^C
--- www.armdesigner.com ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 264.560/363.166/466.868 ms
```

6.10.2 GPS Test

Step 1: Enable the GPS function.

Execute the following command:

```
# echo -e "AT+QGPS=1\r\n" > /dev/ttyUSB2
```

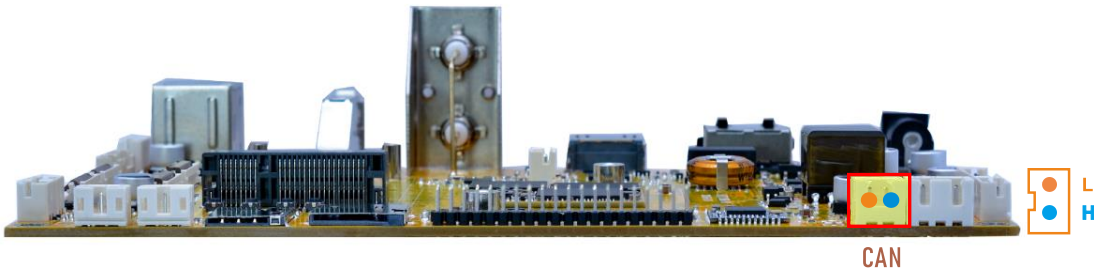
Step 2: Read the GPS data.

Execute the following command:

```
# cat /dev/ttyUSB1
```

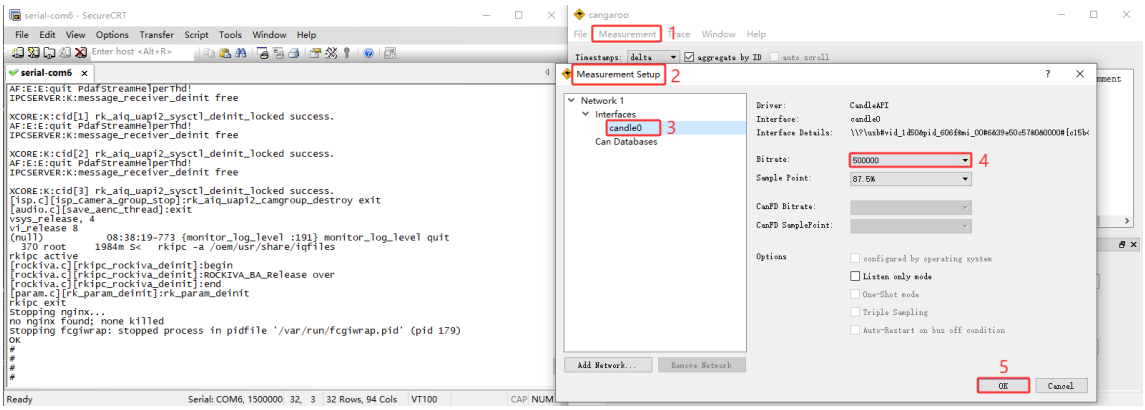
```
# echo -e "AT+QGPS=1\r\n" > /dev/ttyUSB2
# cat /dev/ttyUSB1
$GPVTG,,T,,M,,N,,K,N*2C

$GPGSA,A,1,,,,,,,,,,,,,,*1E

$GPGGA,,,,,,0,,,,,,,,*66

$GPRMC,,V,,,,,,,,,,N*53

$GPVTG,,T,,M,,N,,K,N*2C

$GPGSA,A,1,,,,,,,,,,,,,,*1E

$GPGGA,,,,,,0,,,,,,,,*66

$GPRMC,,V,,,,,,,,,,N*53
```

6.11 CAN



Step 1: Connect the CAN test tool to the board as shown in the diagram below.

Step 2: Open the CAN test software and set the baud rate to 500000.



Step 3: Configure and enable the CAN interface on the board with a bitrate of 500000.

Execute the following command:

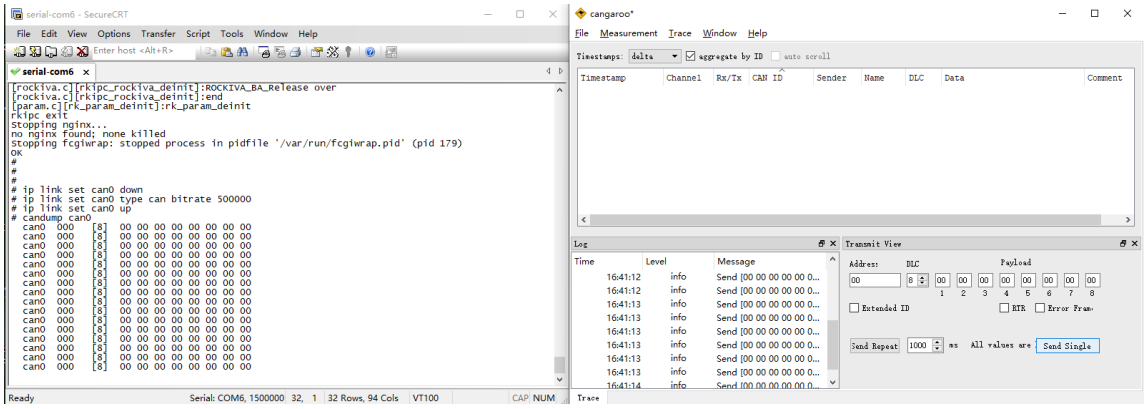
```
# ip link set can0 down
# ip link set can0 type can bitrate 500000
# ip link set can0 up
```

Step 4: Test CAN receiving.

Configure the CAN test tool as the sender, and configure CAN on the board as the receiver.

Execute the following command:

```
# candump can0
```

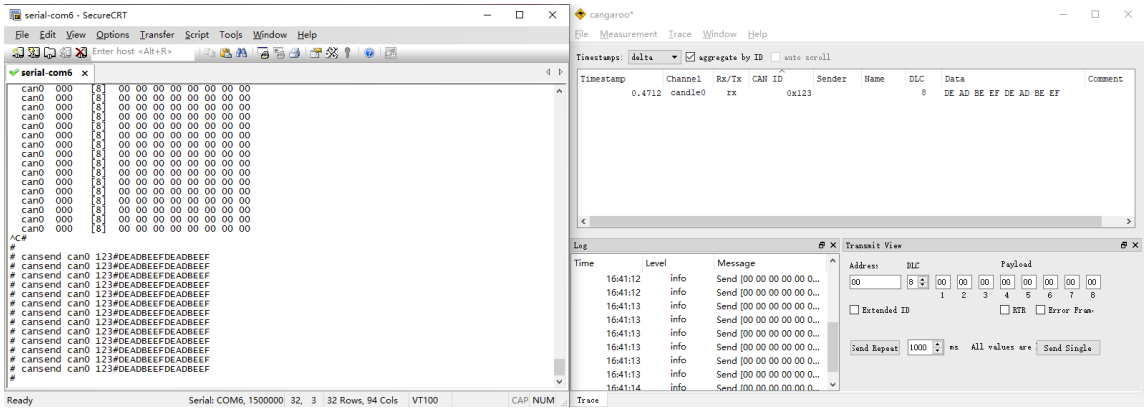


Step 5: Test CAN sending.

Configure the CAN test tool as the receiver, and configure CAN on the board as the sender.

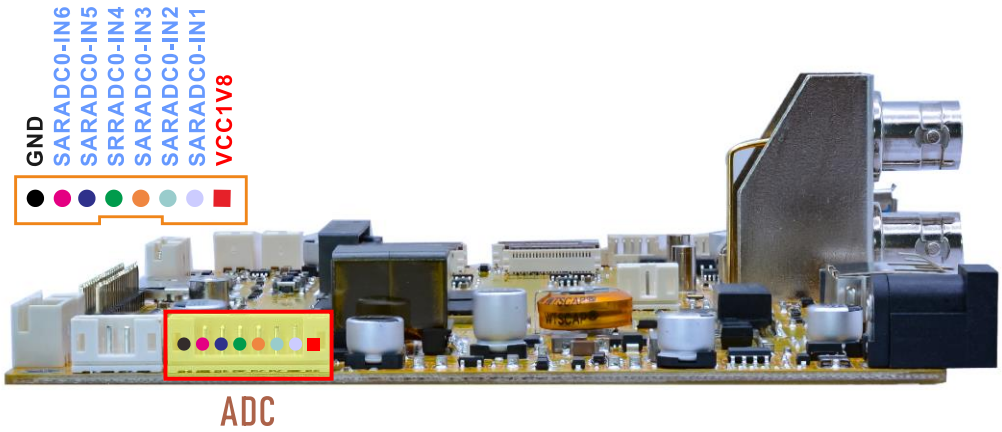
Execute the following command:

```
# cansend can0 123#DEADBEEFDEADBEEF
```



6.12 ADC

The board provides six ADC input interfaces: ADC1, ADC2, ADC3, ADC4, ADC5, and ADC6.



This test is used to check whether the ADC raw value changes according to the input voltage.

Step 1: Apply a voltage level to the ADC input pin.

Connect the ADC pin to GND for a low level, or connect it to 1.8 V for a high level.

Step 2: Read the ADC raw value.

Example for **ADC1**:

```
# cat /sys/bus/iio/devices/iio:\device0/in_voltage1_raw
```

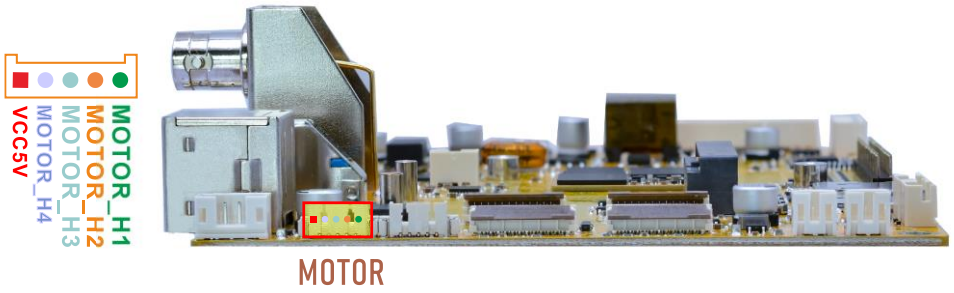
```
# cat /sys/bus/iio/devices/iio:\device0/in_voltage1_raw
8187
# cat /sys/bus/iio/devices/iio:\device0/in_voltage1_raw
12
```

For other ADC read commands, refer to the ADC read command description below:

```
Single-channel read commands:
ADC1 : cat /sys/bus/iio/devices/iio:\device0/in_voltage1_raw
ADC2 : cat /sys/bus/iio/devices/iio:\device0/in_voltage2_raw
ADC3 : cat /sys/bus/iio/devices/iio:\device0/in_voltage3_raw
ADC4 : cat /sys/bus/iio/devices/iio:\device0/in_voltage4_raw
ADC5 : cat /sys/bus/iio/devices/iio:\device0/in_voltage5_raw
ADC6 : cat /sys/bus/iio/devices/iio:\device0/in_voltage6_raw

To read all ADC1-ADC6 raw values, execute the following command:
for i in 1 2 3 4 5 6; do
    echo -n "ADC$i: "
    cat /sys/bus/iio/devices/iio:\device0/in_voltage${i}_raw
done
```

6.13 MOTOR

The KIT1126B provides a stepper motor control interface.



The motor can be controlled using the **motor_test** tool.

Test Command:

```
motor_test /dev/motorX <steps>
```

Note:

One complete motor rotation is equivalent to 4076 steps.

Test Examples:

Rotate the motor forward by 4076 steps:

```
# motor_test /dev/motor0 4076
```

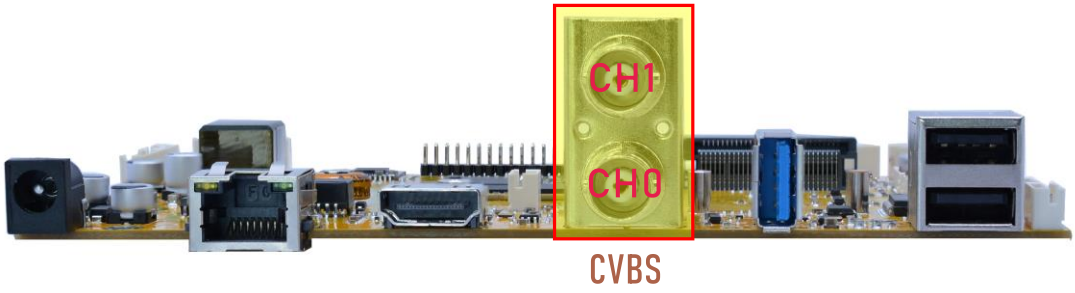
Rotate the motor backward by 1000 steps:

```
# motor_test /dev/motor0 -1000
```

6.14 Camera

6.14.1 CVBS Camera

The KIT1126B provides two CVBS input channels (CH0 and CH1). Only one channel can be used for video capture at a time.



Step 1: Stop the default rkipc service.

Execute the following command:

```
# RkLunch-stop.sh
```

```
# RkLunch-stop.sh
Stop Application ...
[rkipc.c][sig_proc]:received signo 15
  375 root     2720m S<   rkipc -a /oem/usr/share/iqfiles
rkipc active
[server.c][rkipc_server_deinit]:rkipc_server_deinit failed
[osd.c][osd_time_server]:exit
[video.c][rkipc_osd_bmp_destroy]:RK_MPI_RGN_Destroy [2] failed with 0xa0038005
[video.c][rkipc_osd_bmp_destroy]:RK_MPI_RGN_Destroy [3] failed with 0xa0038005
[video.c][rkipc_osd_cover_destroy]:RK_MPI_RGN_Destroy [4] failed with 0xa0038005
[video.c][rkipc_osd_cover_destroy]:RK_MPI_RGN_Destroy [11] failed with 0xa0038005
[video.c][rkipc_osd_cover_destroy]:RK_MPI_RGN_Destroy [5] failed with 0xa0038005
```

Step 2: Obtain the Ethernet IP address and check the network interface status.

Execute the following command:

```
# udhcpc
# ifconfig eth0
```

```
# udhcpc
udhcpc: started, v1.36.1
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.0.188, server 192.168.0.1
udhcpc: lease of 192.168.0.188 obtained from 192.168.0.1, lease time 86400
deleting routers
adding dns 192.168.0.1
#
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 0E:2A:C5:09:62:B0
          inet addr:192.168.0.188  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::c2a:c5ff:fe09:62b0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:712 errors:0 dropped:85 overruns:0 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:100521 (98.1 KiB)  TX bytes:4218 (4.1 KiB)
          Interrupt:79
```

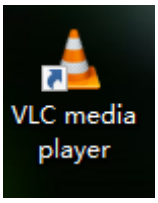
Step 3: Start the cvbs camera demo.

Execute the following command:

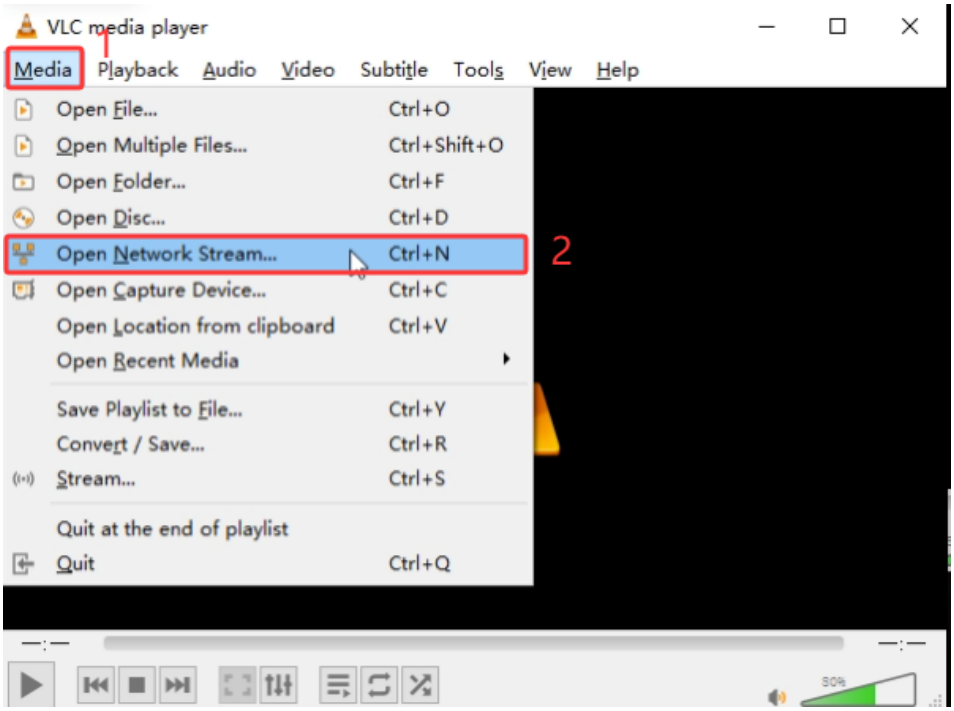
```
# sample_demo_cvbs_camera &
```

```
# sample_demo_cvbs_camera &
# === V4L2 Single-Camera RTSP Streamer ===
Config: /dev/video0 720x576 @ 30fps -> H265 -> rtsp://*:554/live/0

[INFO  rtsp_demo.c:280:rtsp_new_demo] rtsp server demo starting on port 554
[DEBUG rtsp_demo.c:480:rtsp_new_session] add session path: /live/0
rockit_load start
v4l2_tx probe successv4l2_rx_probe success
rockit_load end
rockit log path (null), log_size = 0
log_file = (nil)
(null)          17:05:25-963 {log_level_init    :209}

 please use echo name=level > /tmp/rt_log_level set log level
       name: all cmpi mb sys vdec venc rgn vpss vgs tde avs wbc vo vi ai ao aenc adec
       log_level: 0 1 2 3 4 5 6

rockit default level 4, can use export rt_log_level=x, x=0,1,2,3,4,5,6 change
```

Step 4: Open VLC media player on the PC.

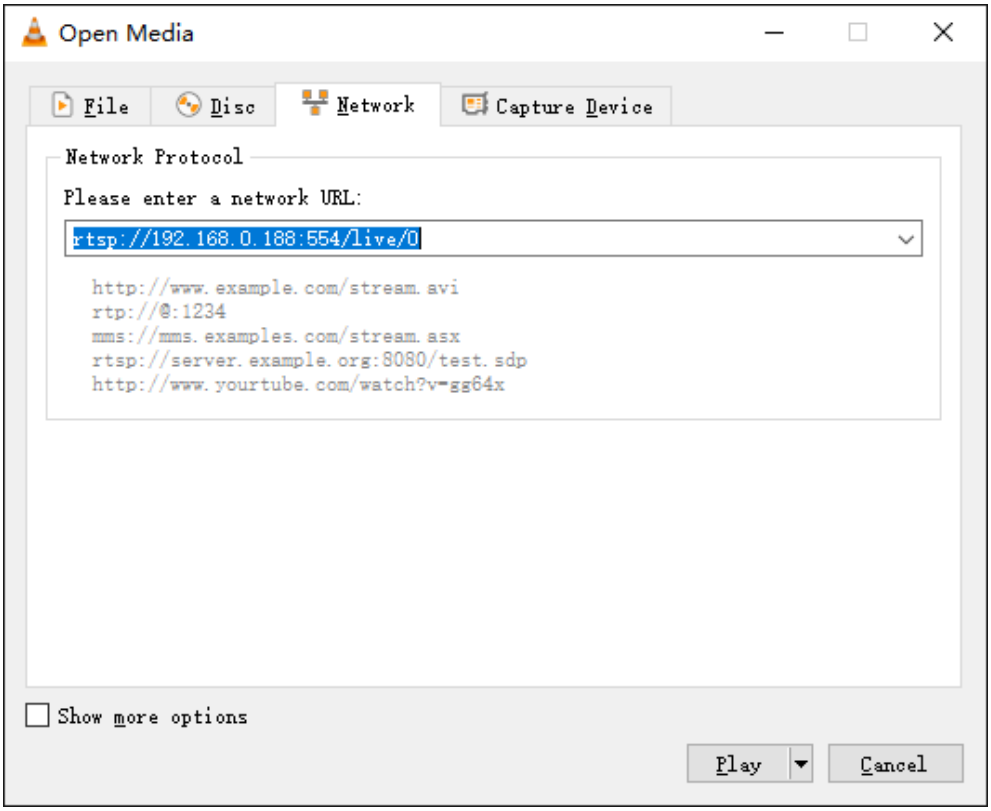


Step 5: Select **Media** -> **Open Network Stream...**.



Step 6: Enter the following RTSP URL to preview the camera stream:

```
rtsp://192.168.0.188:554/live/0
```



Note:

Replace 192.168.0.188 with the actual Ethernet IP address of the board.

Step 7: Switch to the desired CVBS input channel.

After starting the stream, switch the CVBS input channel using the following commands.

Switch to CH0:

```
# echo 0 > /sys/class/gm7150/gm7150_4/input_channel
```

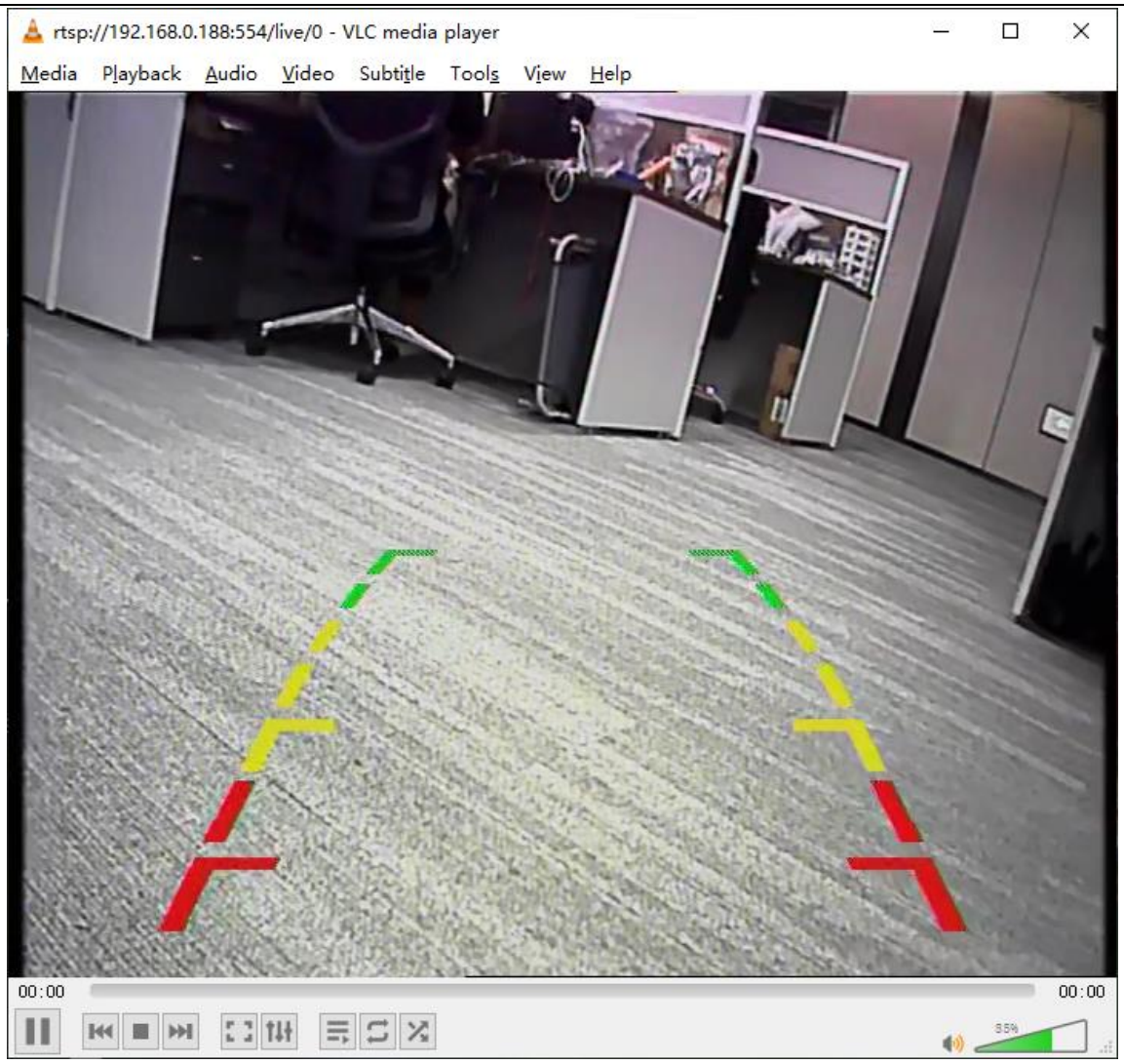
```
# echo 0 > /sys/class/gm7150/gm7150_4/input_channel
#
```

Switch to CH1:

```
# echo 1 > /sys/class/gm7150/gm7150_4/input_channel
```

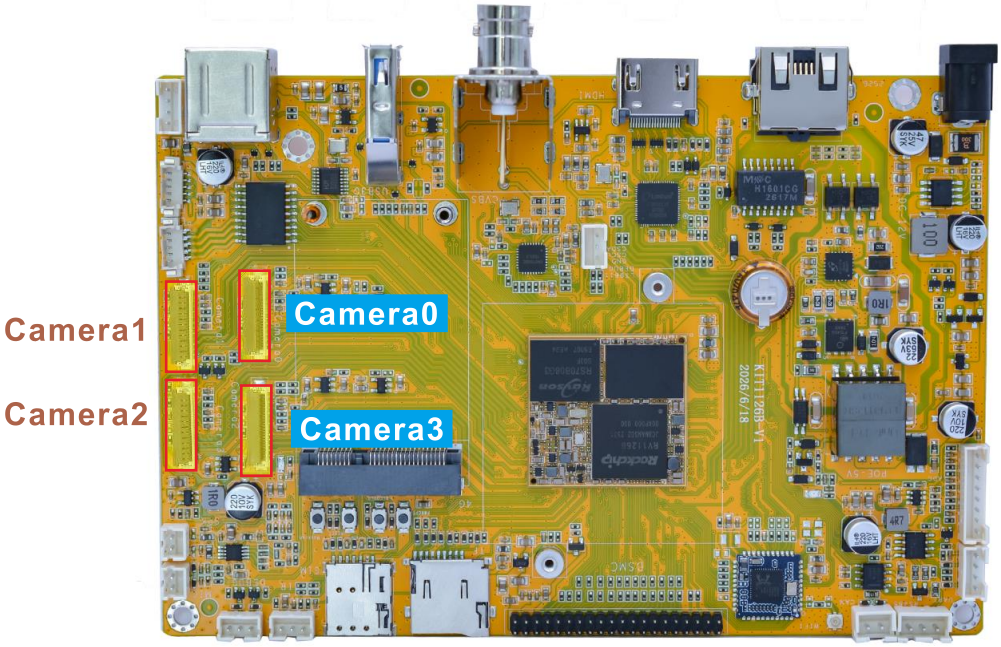
```
# echo 1 > /sys/class/gm7150/gm7150_4/input_channel
#
```

Step 8: Verify that the corresponding CVBS camera image is displayed correctly after switching channels.



6.14.2 MIPI Camera

Connect the camera module, then power on the board.



After the system boots, the stitched preview image from all four camera channels is displayed by default.

6.14.2.1 Four-Channel Camera Preview

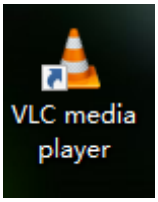
Step 1: Check the network status and obtain the Ethernet IP address of the board.

Execute the following command:

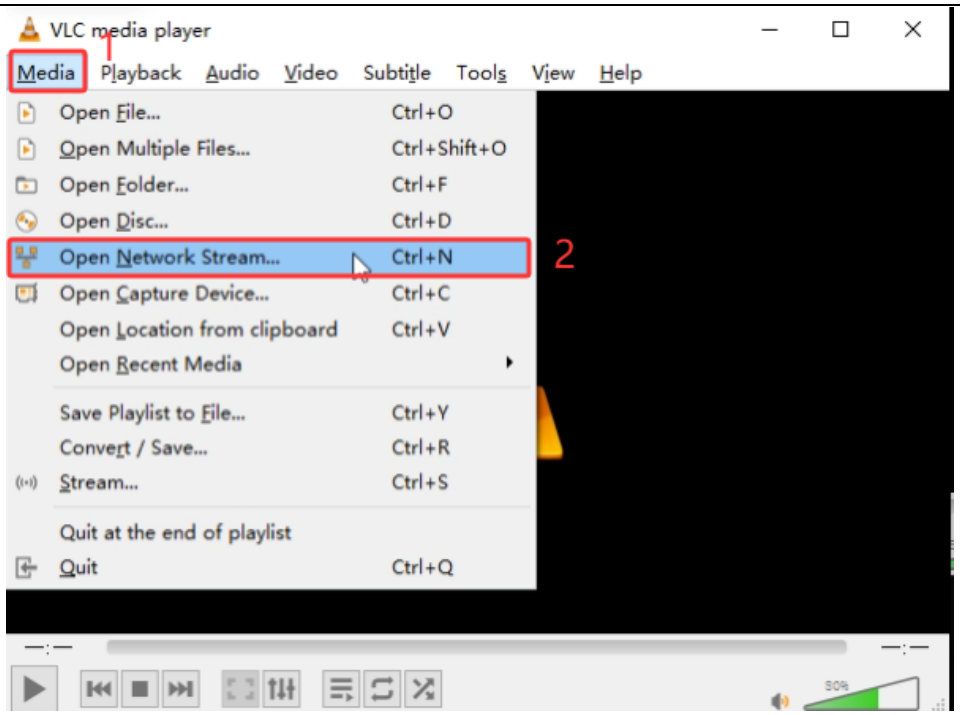
```
# ifconfig eth0
```

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 0E:2A:C5:09:62:B0
          inet addr:192.168.0.188  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::c2a:c5ff:fe09:62b0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2462 errors:0 dropped:280 overruns:0 frame:0
          TX packets:76802 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:282377 (275.7 KiB)  TX bytes:94820146 (90.4 MiB)
          Interrupt:79
```

Step 2: Open VLC media player on the PC.

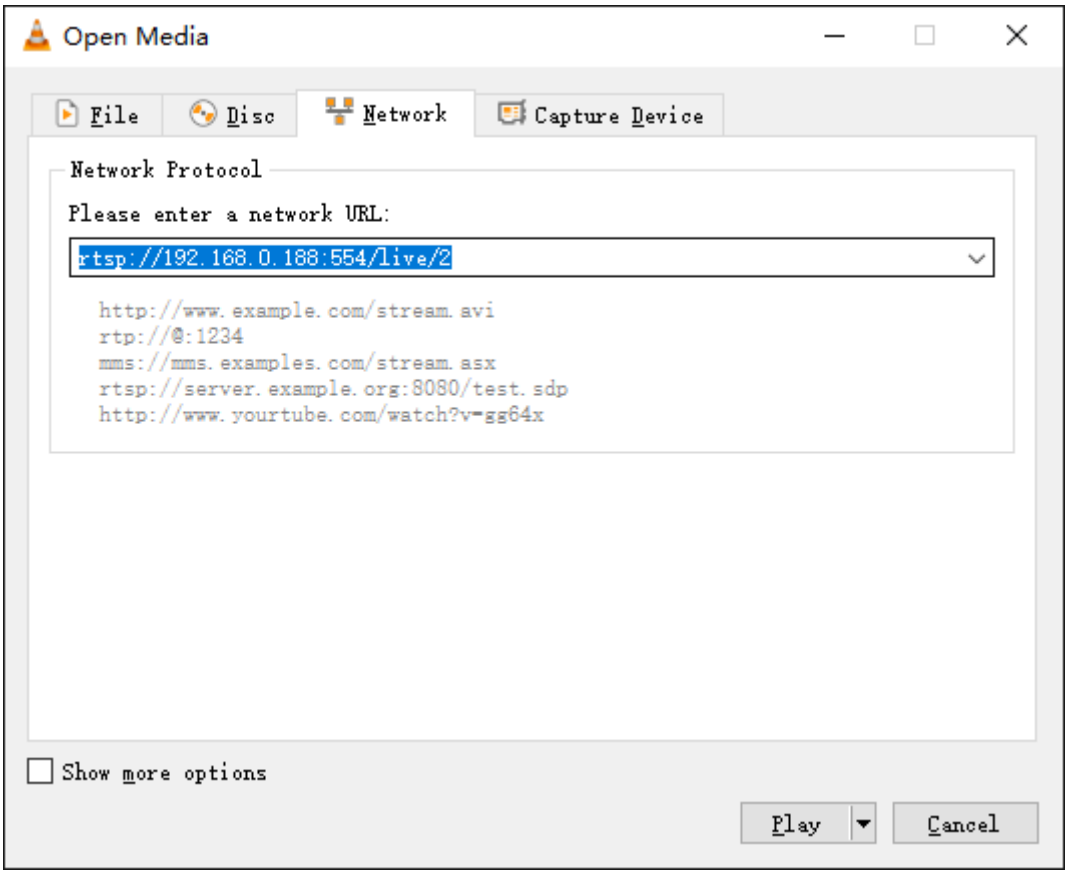


Step 3: Select **Media** -> **Open Network Stream...**.



Step 4: Enter the following RTSP URL to preview the camera stream:

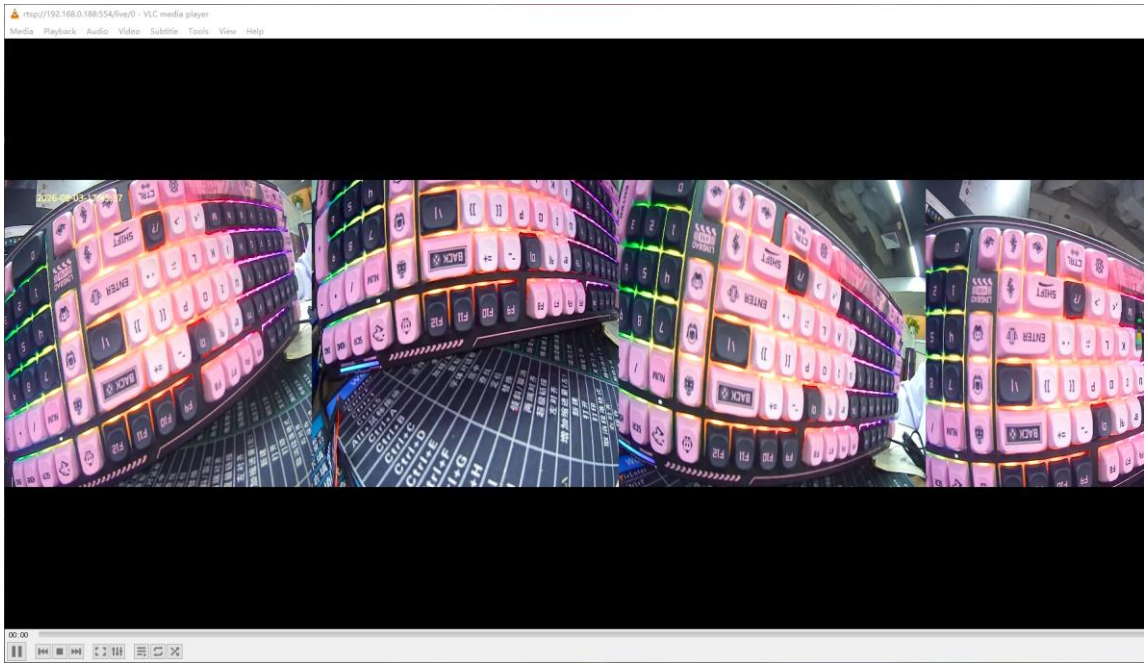
```
rtsp://192.168.0.188:554/live/0
```



Note:

Replace 192.168.0.188 with the actual Ethernet IP address of the board.

Step 5: Verify that the stitched preview image from the four MIPI camera channels is displayed correctly.



6.14.2.2 Single-Channel Camera Preview

Step 1: Stop the default rkipc service.

Execute the following command:

```
# RkLunch-stop.sh
```

```
# RkLunch-stop.sh
Stop Application ...
[rkipc.c][sig_proc]:received signo 15
  385 root     2720m S<   rkipc -a /oem/usr/share/iqfiles
rkipc active
[server.c][rkipc_server_deinit]:rkipc_server_deinit failed
[osd.c][osd_time_server]:exit
[video.c][rkipc_osd_bmp_destroy]:RK_MPI_RGN_Destroy [2] failed with 0xa0038005
[video.c][rkipc_osd_bmp_destroy]:RK_MPI_RGN_Destroy [3] failed with 0xa0038005
[video.c][rkipc_osd_cover_destroy]:RK_MPI_RGN_Destroy [4] failed with 0xa0038005
[video.c][rkipc_osd_cover_destroy]:RK_MPI_RGN_Destroy [11] failed with 0xa0038005
[video.c][rkipc_osd_cover_destroy]:RK_MPI_RGN_Destroy [5] failed with 0xa0038005
[video.c][rkipc_osd_cover_destroy]:RK_MPI_RGN_Destroy [12] failed with 0xa0038005
[video.c][rkipc_osd_bmp_destroy]:RK_MPI_RGN_Destroy [6] failed with 0xa0038005
[video.c][rkipc_bind_deinit]:start
  385 root     2667m S<   rkipc -a /oem/usr/share/iqfiles
rkipc active
```

Step 2: Obtain the Ethernet IP address and check the network interface status.

Execute the following command:

```
# udhcpc
# ifconfig eth0
```

```
# udhcpc
udhcpc: started, v1.36.1
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.0.188, server 192.168.0.1
udhcpc: lease of 192.168.0.188 obtained from 192.168.0.1, lease time 86400
deleting routers
adding dns 192.168.0.1
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 0E:2A:C5:09:62:B0
          inet addr:192.168.0.188  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::c2a:c5ff:fe09:62b0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:863 errors:0 dropped:122 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:79750 (77.8 KiB)  TX bytes:3302 (3.2 KiB)
          Interrupt:79
```

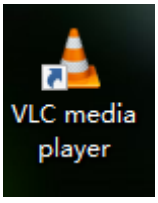
Step 3: Start the camera demo.

Execute the following command:

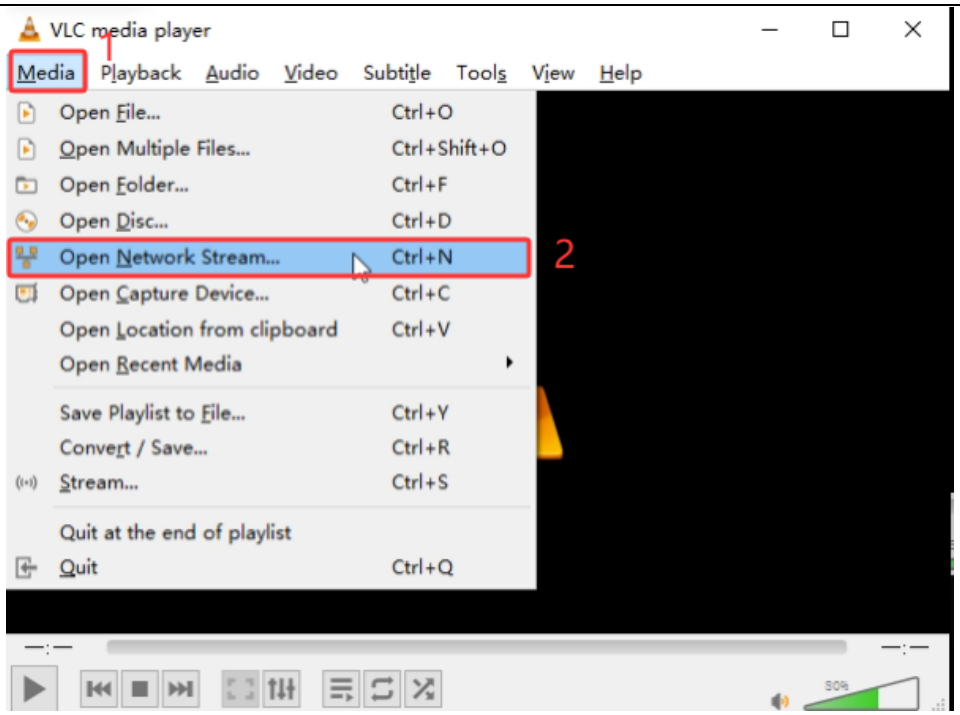
```
# sample_demo_four_camera
```

```
# sample_demo_four_camera
No arguments provided, using default settings for 4 cameras.
usage example:
       sample_demo_four_camera -s 0 -W 1920 -H 1080 -w 720 -h 576 -f 30 -r 0 -s 1 -W 1920 -H 1080 -w 720
-h 576 -f 30 -r 0 -s 2 -W 1920 -H 1080 -w 720 -h 576 -f 30 -r 0 -s 3 -W 1920 -H 1080 -w 720 -h 576 -f 30
-r 0 -n 0 -b 1
       rtsp://xx.xx.xx.xx/live/0, camera 0 main-stream
       rtsp://xx.xx.xx.xx/live/1, camera 0 sub-stream
       rtsp://xx.xx.xx.xx/live/2, camera 1 main-stream
       rtsp://xx.xx.xx.xx/live/3, camera 1 sub-stream
       rtsp://xx.xx.xx.xx/live/4, camera 2 main-stream
       rtsp://xx.xx.xx.xx/live/5, camera 2 sub-stream
       rtsp://xx.xx.xx.xx/live/6, camera 3 main-stream
       rtsp://xx.xx.xx.xx/live/7, camera 3 sub-stream
       -s | --sensor id (0~3)
       -f | --fps: frame per second, Default 30
       -r | --hdr: high dynamic range, Default 0
       -W | --main_width: main-stream width, Default 1920
       -H | --main_height: main-stream height, Default 1080
       -w | --sub_width: sub-stream width, Default 720
       -h | --sub_height: sub-stream height, Default 576
       -n | --enable_npu: enable npu, Default 0
       -b | --buf_share: enable buf share, Default 1
Starting with default config...
```

Step 4: Open VLC media player on the PC.



Step 5: Select **Media** -> **Open Network Stream...**.



Step 6: Enter the following RTSP URL to preview the camera stream:

Camera0:

```
rtsp://192.168.0.188:554/live/0
```

Camera1:

```
rtsp://192.168.0.188:554/live/2
```

Camera2:

```
rtsp://192.168.0.188:554/live/4
```

Camera3:

```
rtsp://192.168.0.188:554/live/6
```

Note:

Replace 192.168.0.240 with the actual Ethernet IP address of the board.

Step 7: Verify that each MIPI camera stream can be previewed correctly.

6.15 Video Playback

Video playback can be tested using the simple_vdec_bind_vo command.

Execute the following command:

```
# simple_vdec_bind_vo -i /tmp/sda1/testh264.h264
```

```
# simple_vdec_bind_vo -i /tmp/sda1/testh264.h264
param list: w = 1920, h = 1080, i = /tmp/sda1/testh264.h264, b = 1
rockit_load start
v4l2_tx probe successv4l2_rx_probe success
rockit_load end
rockit log path (null), log_size = 0
log_file = (nil)
(null)          10:25:13-692 {log_level_init    :209}

 please use echo name=level > /tmp/rt_log_level set log level
       name: all cmpi mb sys vdec venc rgn vpss vgs tde avs wbc vo vi ai ao aenc adec
       log_level: 0 1 2 3 4 5 6

rockit default level 4, can use export rt_log_level=x, x=0,1,2,3,4,5,6 change
(null)          10:25:13-692 {read_log_level    :083} text is all=4
(null)          10:25:13-692 {read_log_level    :085} module is all, log_level is 4
(null)          10:25:13-692 {dump_version      :060} ----------------------------------------------------
-----
(null)            10:25:13-692 {dump_version      :061} rockit version: git-b85f3c523 Mon Jan 26 09:47:22
2026 +0800
(null)          10:25:13-692 {dump_version      :062} rockit building: built-Chu 2026-01-27 09:45:41
(null)          10:25:13-692 {dump_version      :063} ----------------------------------------------------
-----
(null)          10:25:13-693 {monitor_log_level :141} #Start monitor_log_level thread, arg:(nil)
vsys dev open 4
load library(librga.so) in releative path
mpp[3484]: mpp_info: mpp version: d7dd00e4f author: Herman Chen  2026-02-13 fix[h265d]: Fix invalid hvcc
data crash on seek
mpp[3484]: hal_264d_com: control info: fmt 0, w 1920, h 1080
mpp[3484]: mpp_dec: mpp_dec_proc_cfg found MPP_DEC_SET_FRAME_INFO fmt 0
mpp[3484]: mpp_buf_slot: mismatch h_stride_by_pixel 1984 - 1920
```

Command explanation:

- simple_vdec_bind_vo: Video decoding and display test tool.
- -i: Specifies the input video file.
- /tmp/sda1/testh264.h264: Input H.264 file path.

Note:

This command supports H.264 elementary stream files, such as .h264 files. Standard MP4 files are not supported.