EM1103B Linux5.10 User Manual

V1.0

Boardcon Embedded Designer

Overview

The content of this document is intended solely for the EM1103B development board, aiming to help users quickly understand, apply, and test the EM1103B development board.

System Support

Development Board	Linux
mini1103b_V1 em1103b_V1	5.10.209

Revision History

Version	Date	Author	Revision History
V1.0	2025-07-15	Xue Junchao	Initial version

Disclaimer

The information in this manual is for reference only. While Boardcon strives to ensure its accuracy, no guarantees are made regarding its completeness or correctness. All content is subject to change without prior notice. Boardcon reserves the right to revise the content of this manual without prior notification.

Boardcon embedded design limited
2508 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

The Boardcon EM1103B is a compact single-board computer (SBC) built around Rockchip's RV1103B, a highly integrated vision processor SoC optimized for AI-driven applications such as smart IP cameras, intelligent doorbells, and Battery IPC. Combining robust processing power, advanced imaging capabilities, and versatile connectivity, the EM1103B delivers a streamlined solution for next-generation vision-based IoT systems.

Powerful AI & Vision Processing

At its core, the RV1103B features a single-core ARM Cortex-A7 CPU with NEON and FPU support, paired with a built-in NPU (Neural Processing Unit) capable of INT8 operations. This enables efficient execution of AI models converted from popular frameworks like TensorFlow, PyTorch, and Caffe, making the EM1103B ideal for real-time object detection, facial recognition, and other machine learning tasks.

Advanced Imaging Capabilities

The EM1103B leverages Rockchip's 8MP hardware ISP with dual MIPI CSI interfaces, supporting simultaneous video input from two camera sensors. Integrated HDR, 3A (AE/AF/AWB), 3DNR, and other algorithm accelerators ensure superior image quality even in challenging lighting conditions. The SoC's multi-stream H.265/H.264 video encoding allows concurrent high-resolution local storage and lower-resolution cloud streaming, optimizing bandwidth and storage efficiency.

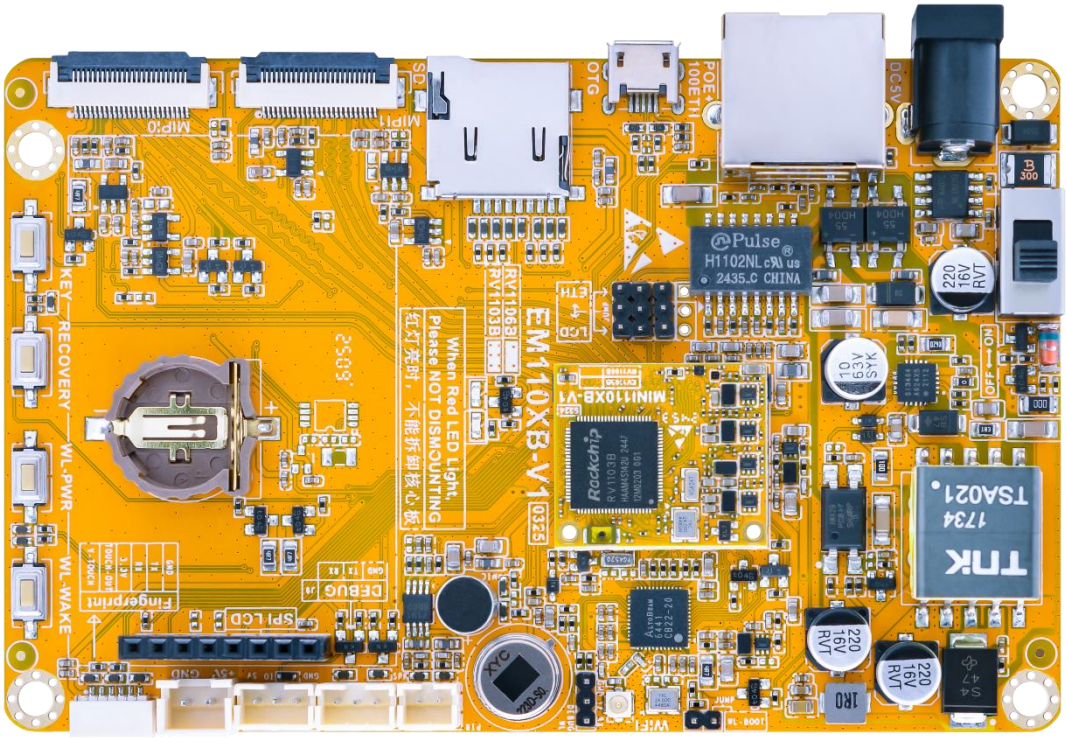
Versatile Connectivity & Expandability

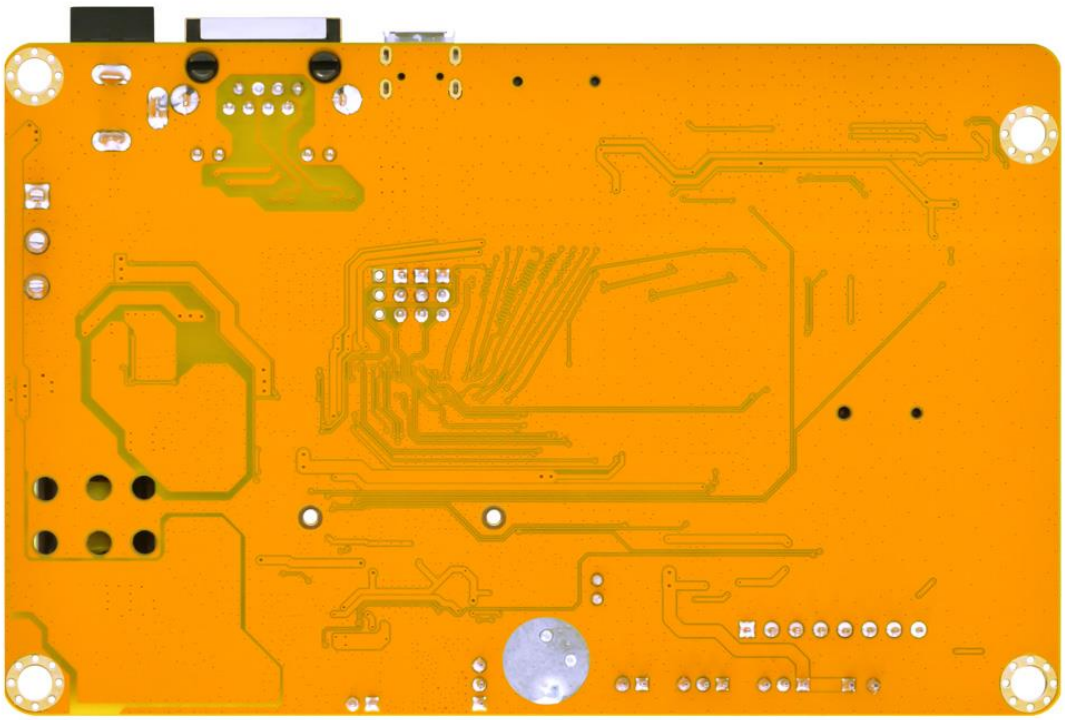
Designed for flexibility, the EM1103B offers dual 2-lane MIPI camera support, a micro SD slot, 100M Ethernet, Wi-Fi, speaker, and mic interfaces. Unique features include a fingerprint sensor interface (compatible with FPM383C modules) and an onboard PIR sensor for motion detection. Power-over-Ethernet (PoE) support (5V) simplifies deployment in surveillance and access control systems.

Compact Design & Low Power Consumption

The EM1103B adopts a modular design with a 22mm x 19mm core board and a 110mm x 70mm baseboard, connected via a 100-pin board-to-board connector. Its fast boot-up time and ultra-low power consumption make it ideal for always-on devices like smart doorbells and IP cameras, while the integrated 16-bit DRAM and RTC ensure reliable operation.

Combining Rockchip's cutting-edge vision processing with Boardcon's hardware expertise, the EM1103B empowers developers to build intelligent, responsive, and energy-efficient AIoT solutions with ease.



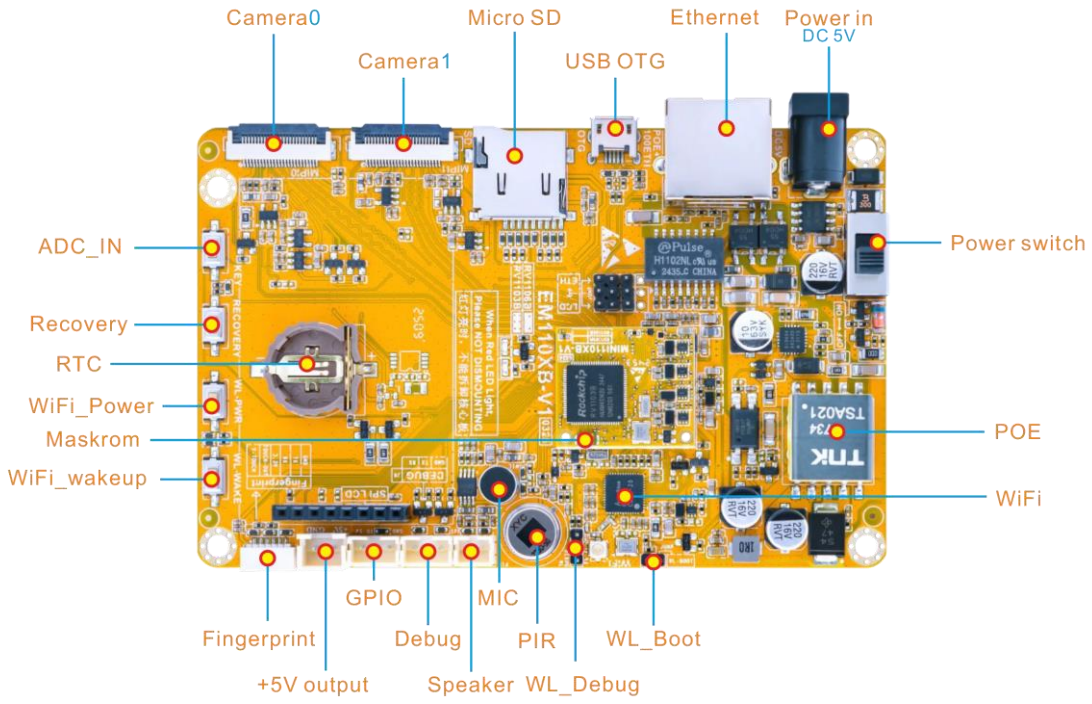


1.2 Product Parameters

Basic Parameters		
SOC		• RV1103B • Single-core 32-bit ARM Cortex-A7@ up to 1.2GHz • RISC-V
NPU		• 0.5 TOPS AI computing power • Supports int8 operations
Video	Encoder	• Support H.265/H.264 video encoding • Support the multi-stream encoding
Memory		• 128MB DDR3 (up to 256MB)
Storage		• 8GB eMMC (up to 32GB)
Support system		• Linux5.10.209 • U-Boot 2017.09
Hardware Parameters		
Extended Storage		• Support MicroSD Card
Audio		• Support 1x differential MIC • Support 1x Speaker, 2-pin connector
USB		• Support 1x USB2.0 OTG, Micro USB
Network		• Support 1x 10/100 Mbps Ethernet RJ45 port • Support 2.4G WiFi (802.11b/g/n) with 512KB SRAM and 2MB SPI Flash
Camera		• Support 2x 2-lane MIPI CSI, 24-pin FPC connector
Fingerprint		• Support Requires separate FPM383C module. 6-pin connector
Serial Port		• Support 1x Debug, 3-pin connector • Support 1x WiFi_Debug, 3-pin header

Keys & Switch	• Support Maskrom(on CPU module), WiFi_Power, Power switch
Other parameters	• Support RTC with battery connector, GPIO, WiFi boot/download select, PIR, +5V output.
Electrical Parameters	
Power supply input voltage	5V/3A DC input jack POE
RTC input voltage	3V/0.6uA
Operating temperature	0 ~ 70°
Storage temperature	-40 ~ 85°
Structural Parameters	
Core board dimensions	22.0mm x 19.0mm
Motherboard dimensions	110.0mm x 70.0mm

1.3 Hardware Interface Introduction



Interface parameters	
Power in DC 5V	5V DC power input interface
MIC	Audio input
Speaker	Audio output
RTC	RTC coin cell connector
Gigabit Ethernet	Gigabit Ethernet RJ45 interface
USB OTG	OTG download interface, ADB
Micro SD	MicroSD card slot
Marskrom	Marskrom key
WL_Boot	WiFi download switch
WL_Debug	WiFi Debug
PIR	Passive Infrared Sensor
Fingerprint	Fingerprint interface
Power switch	Power switch
Debug	UART0, debug the serial port
POE	Power over Ethernet
ADC_IN	Not used
Recovery	Not used
WiFi_Power	Reset WiFi
WiFi_wakeup	Not used
GPIO	GPIO interface
WIFI	ALTOBEAM ATBM6441 module
Camera0/1	SC500AI camera interface

2.Install Drivers and Tool

To download firmware and debug in the terminal, the following drivers and software need

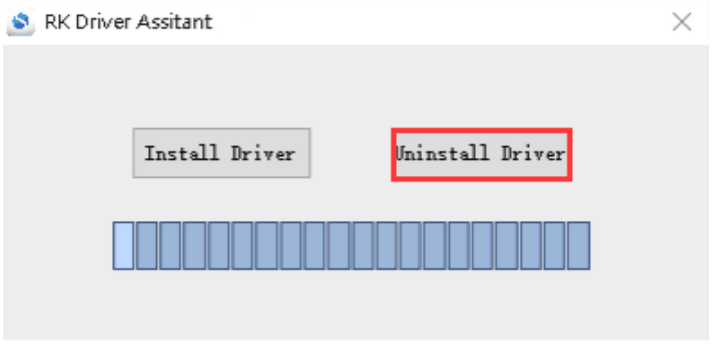
to be installed (for Windows computers):

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

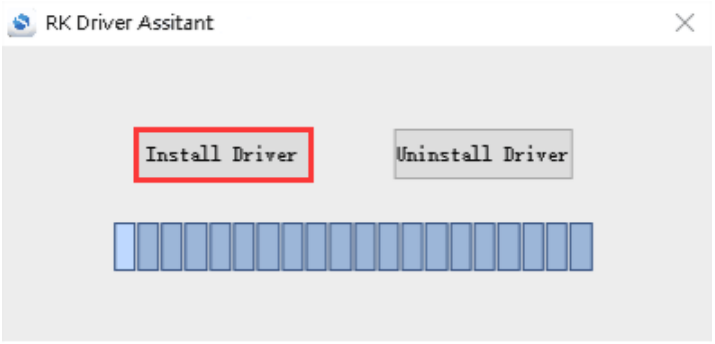
2.1 Install RK Driver Assitant

Step 1: Open *DriverAssitant_v5.1.1/DriverInstall.exe*.

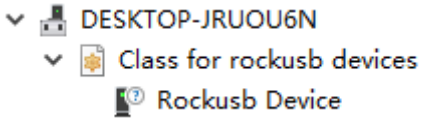
Step 2: To avoid driver conflicts, click **"Uninstall Driver"** to uninstall the driver.



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

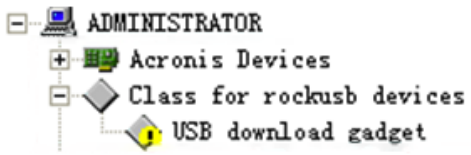


Step 4: After the installation is complete, connect the board and PC with Micro USB cable and press the **Recovery** key and hold then power the board, the following information is displayed in the Computer **Device Manager**, indicating that the USB driver was successfully installed.



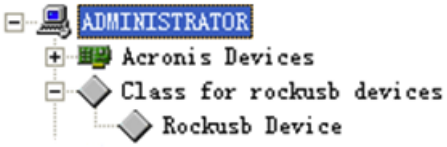
Step 5: If the following device information appears in the **Device Manager** after the

operation in Step 4, user need to proceed to the next step.



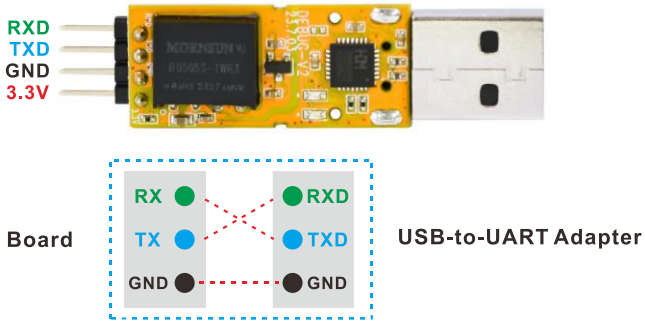
Step 6: The WINDOW will pop up found New Hardware Wizard dialog box, choose to install from the specified location, and then select: *DriverAssitant_v5.1.1/ADBDriver.*

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.

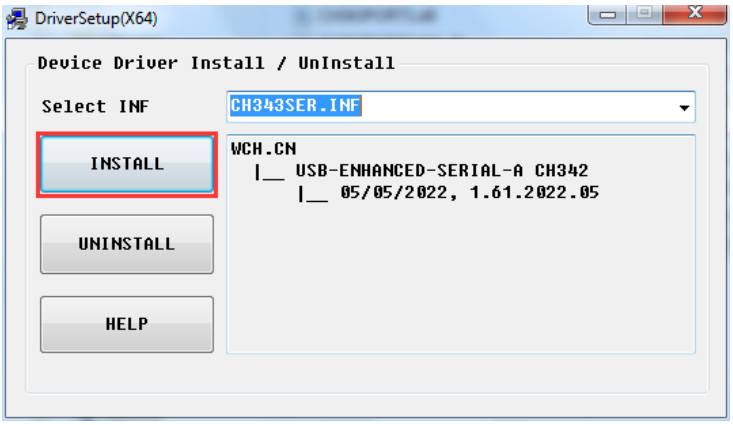
2.2.2 Install Driver

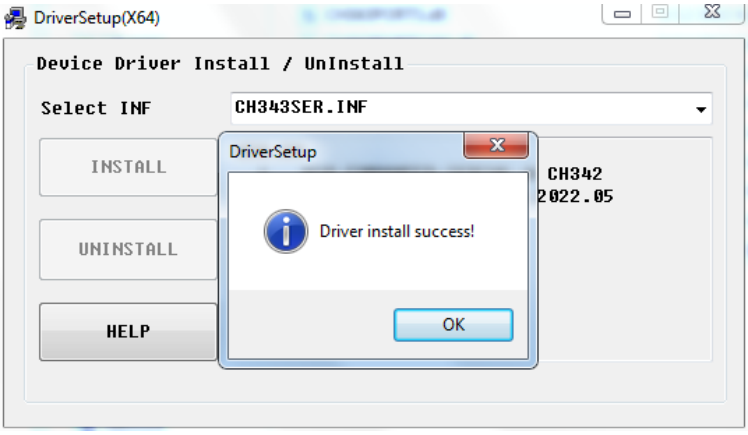
Step 1: Plug the CH9102X Module to the PC

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

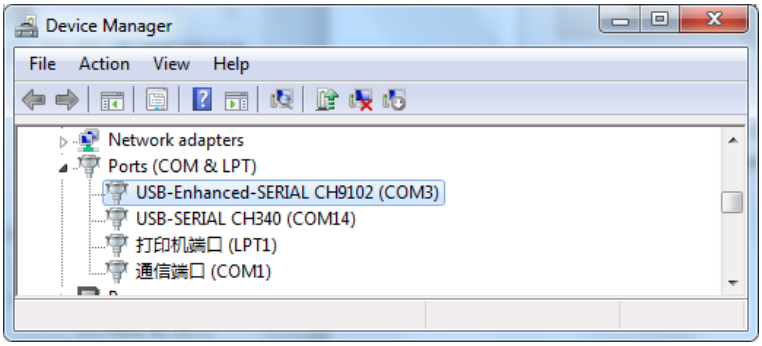
Step 3: Select and install the corresponding *SETUP.EXE* according to the computer

properties.





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

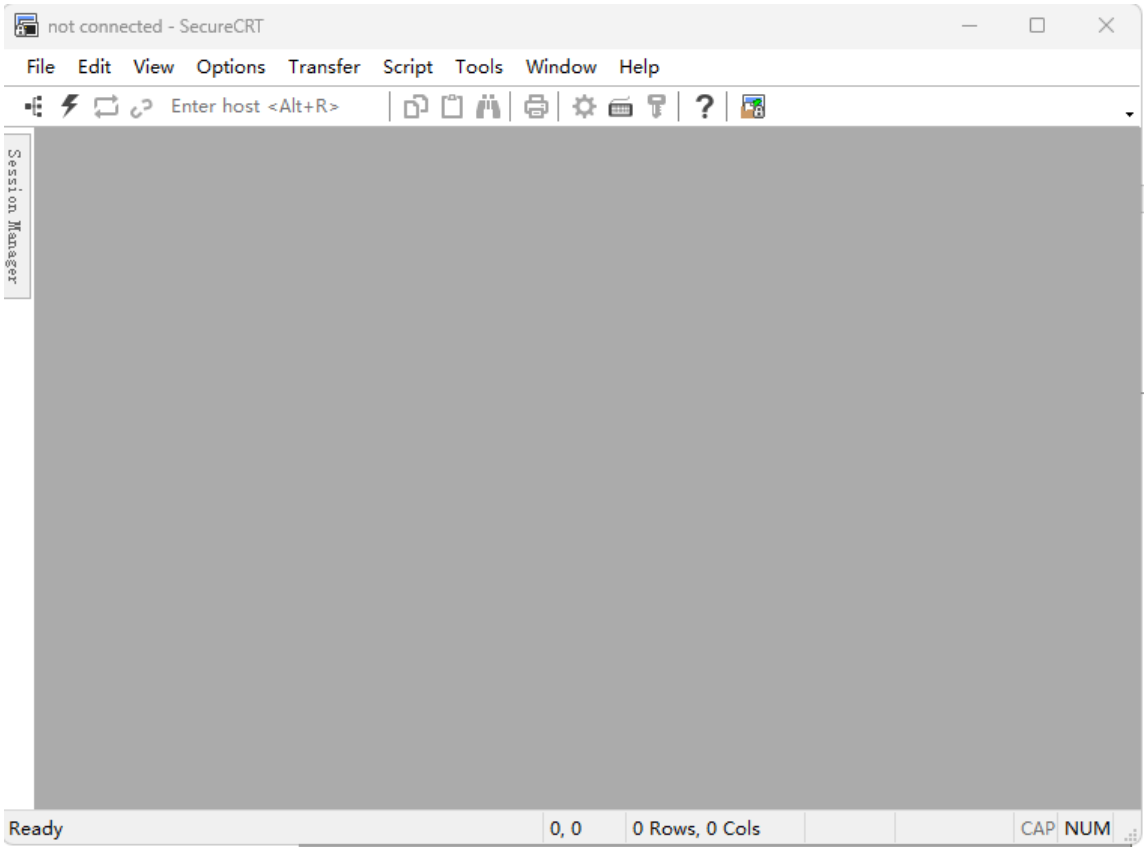


2.3 Install Serial Terminal Tool

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

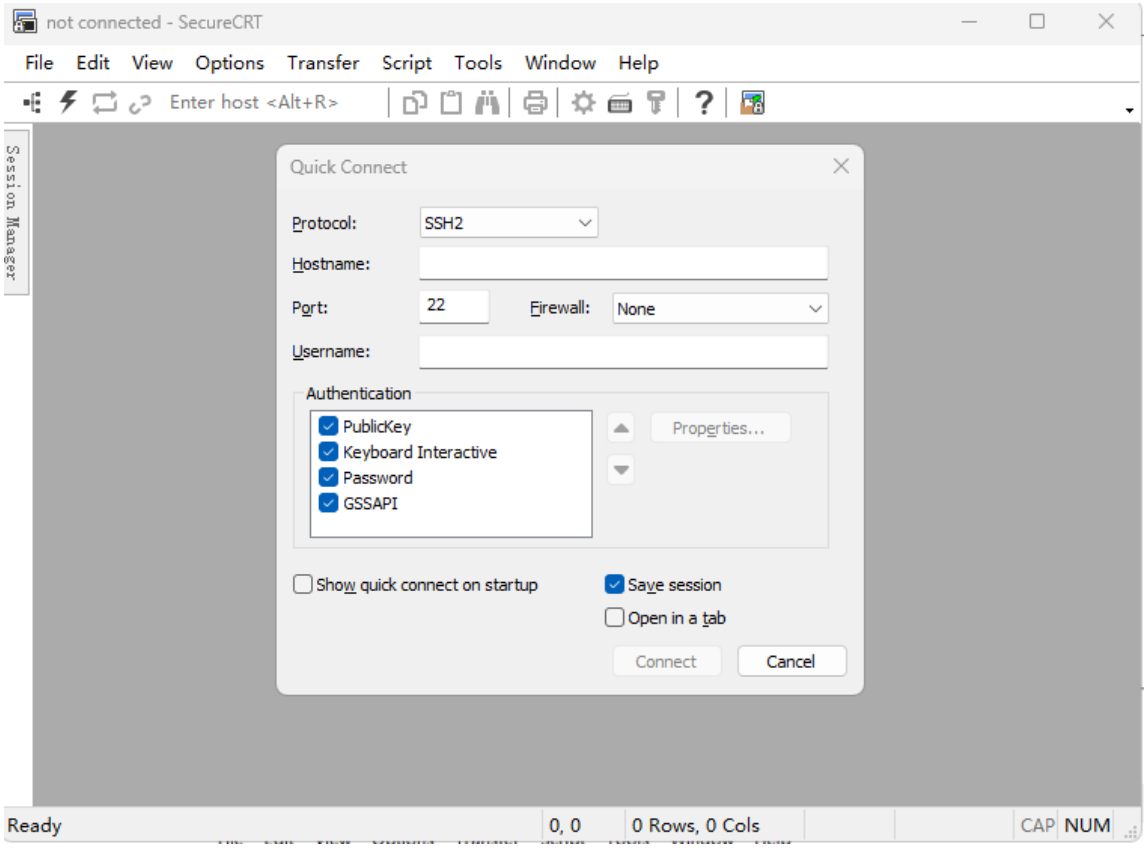
Step 1: Unzip *Platform/SecureCRT.rar* on PC.

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

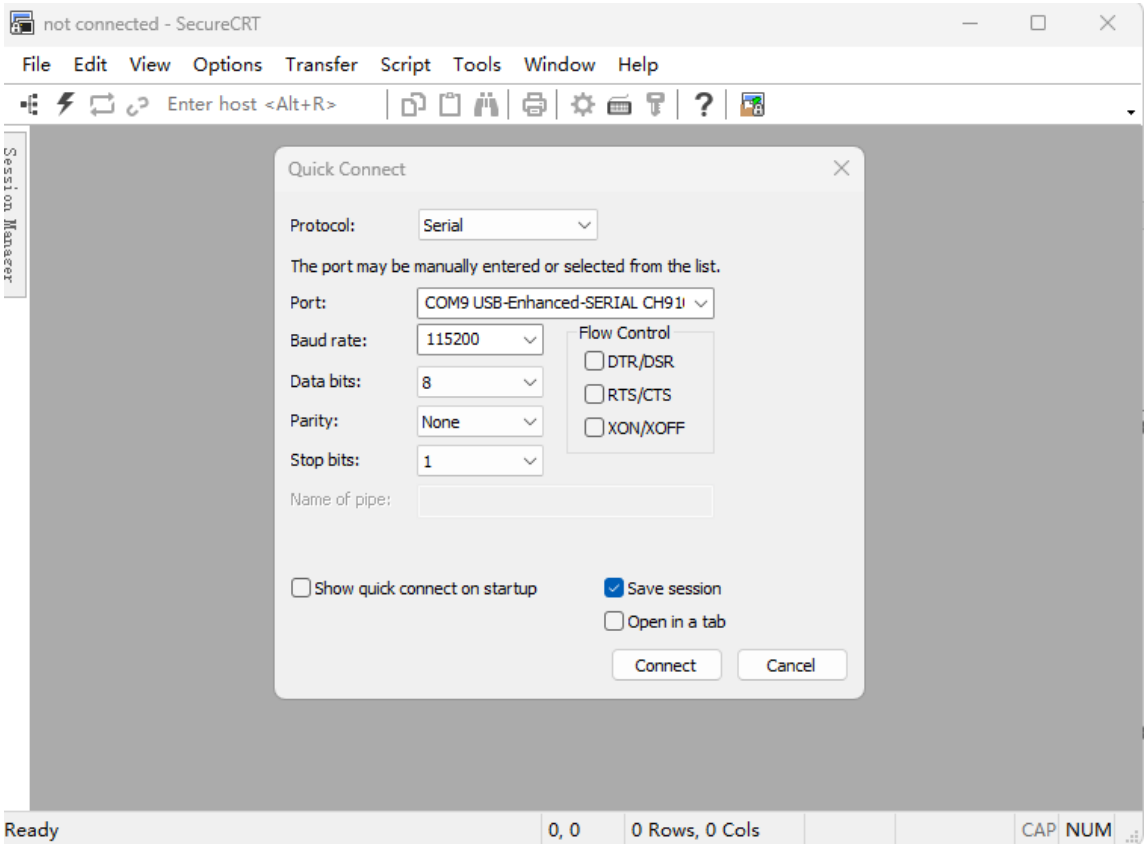


Step 3: Confirm that the CH9102X driver has been installed and the CH9102X module is connecting to the PC.

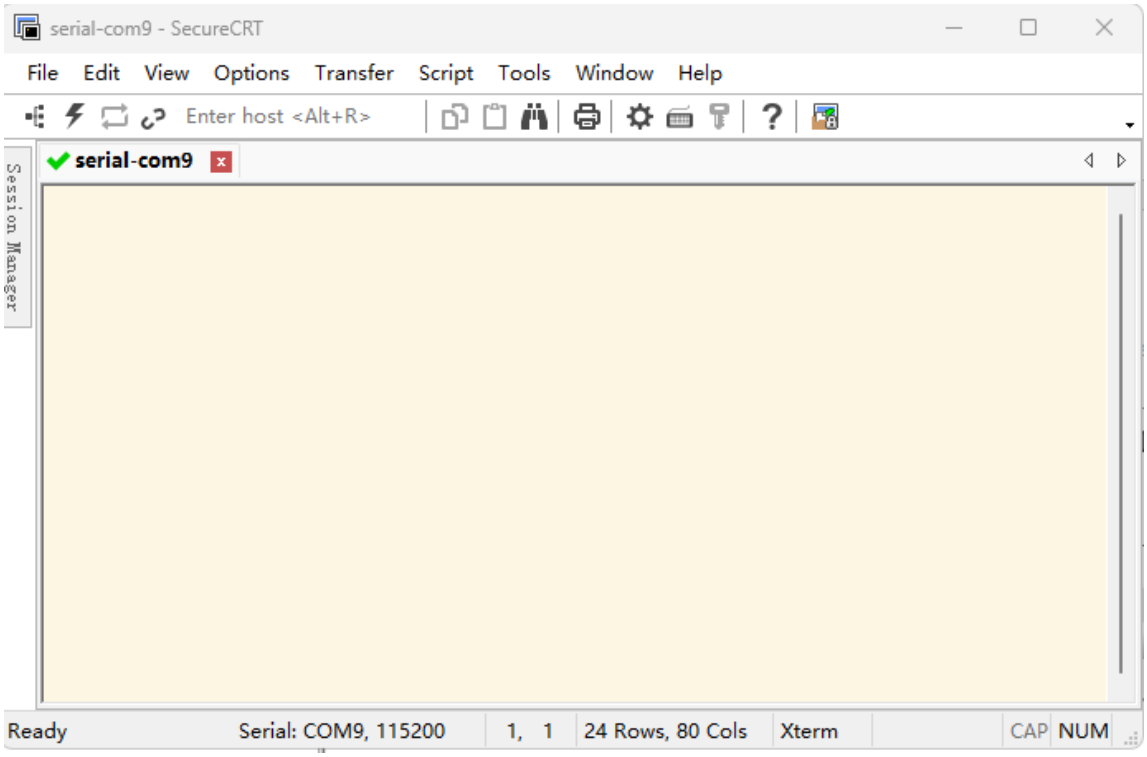
Step 4: Click the **"Quick Connect"** button to go to the Quick Connect configuration screen.



Step 5: Configure as shown in the following figure.



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



3.Upgrade Introduction

3.1 Burn RV1103B Firmware

The RV1103B can upgrade its firmware in MaskRom Mode via a USB cable.

MaskRom Mode:

Entering the Maskrom mode enables the firmware to be programmed.

• Prerequisite

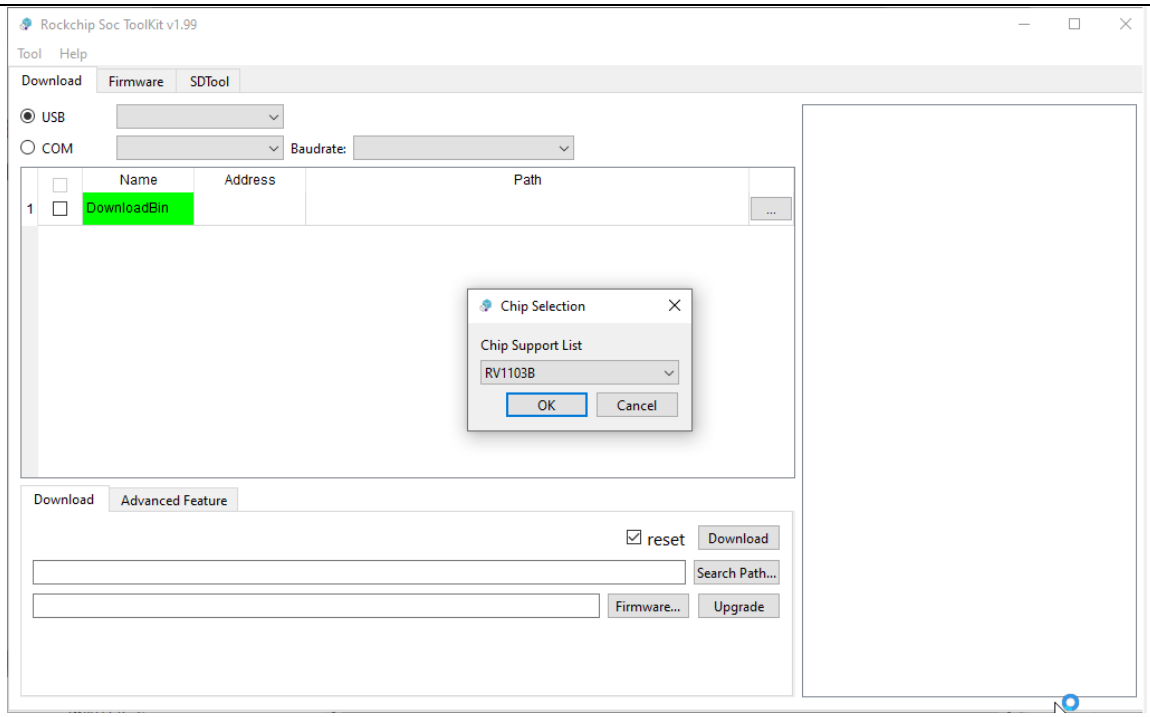
Before upgrading the firmware via USB cable, ensure that the necessary drivers are installed. For installation instructions, refer to the section Install RK Driver Assistant.

3.1.1 How to Enter Maskrom Mode

3.1.1.1 Hardware

Step 1: Unzip SocToolKit.rar on Windows.

Step 2: Open SocToolKit.exe, select RV1103B and click OK.

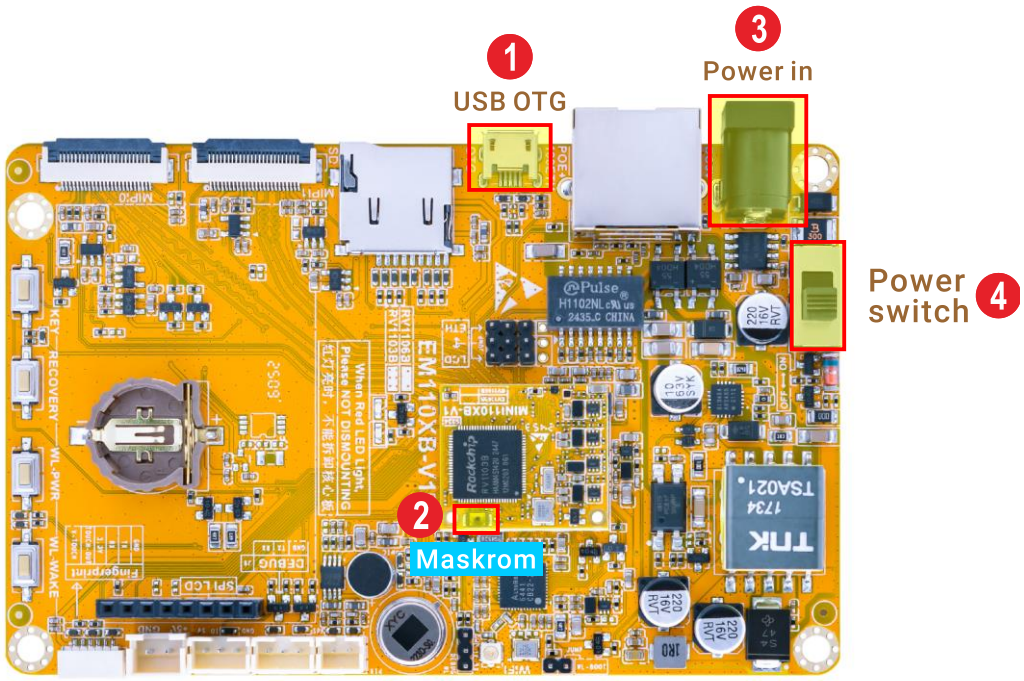


Step 3: Disconnect the power adapter.

Step 4: Connect one end of the Micro data cable to the host, and the other end to the USB OTG interface of the development board.

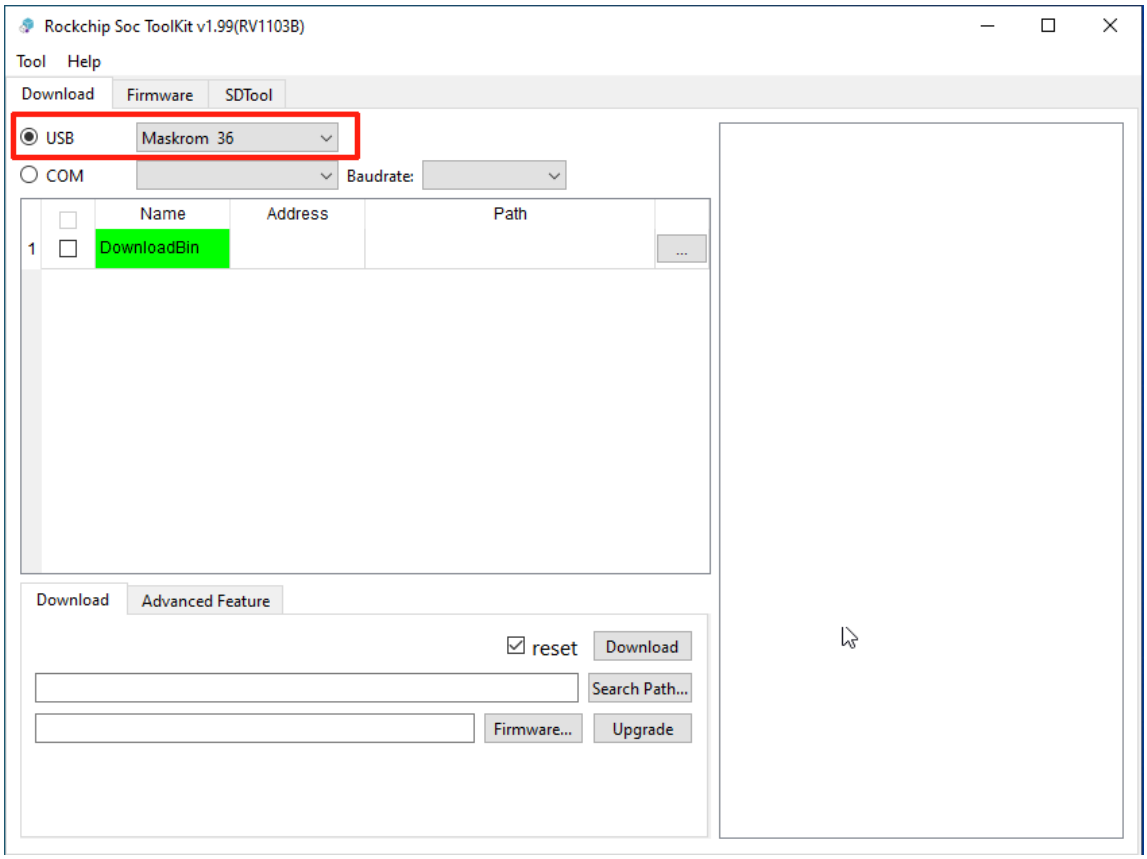
Step 5: Press and hold the **Maskrom** button on the board

Step 6: Connect the power supply.



Step 7: After a few seconds, release the **Maskrom** button when the tool shows **"Maskrom**

36”.



3.1.1.2 Software

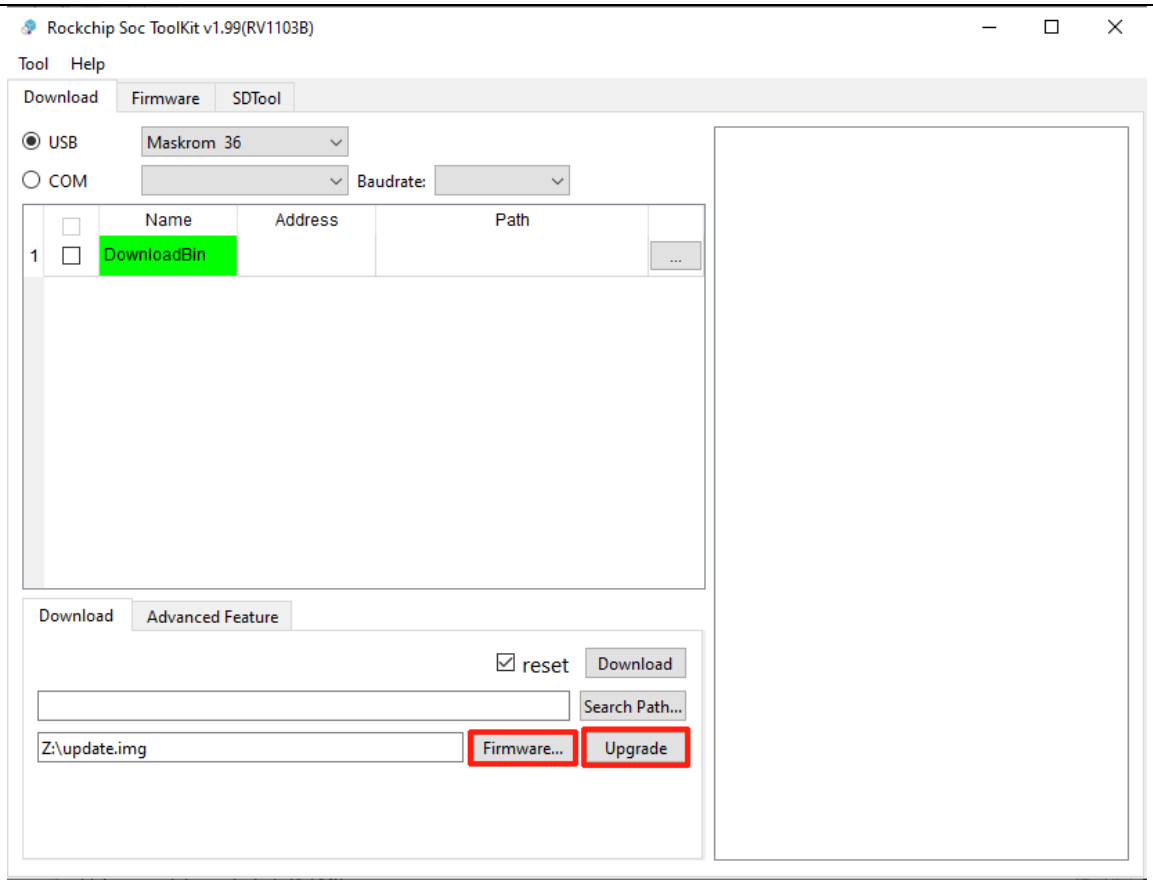
After connecting the Micro cable, execute the following command in the serial debug terminal or adb shell.

```
# reboot loader
```

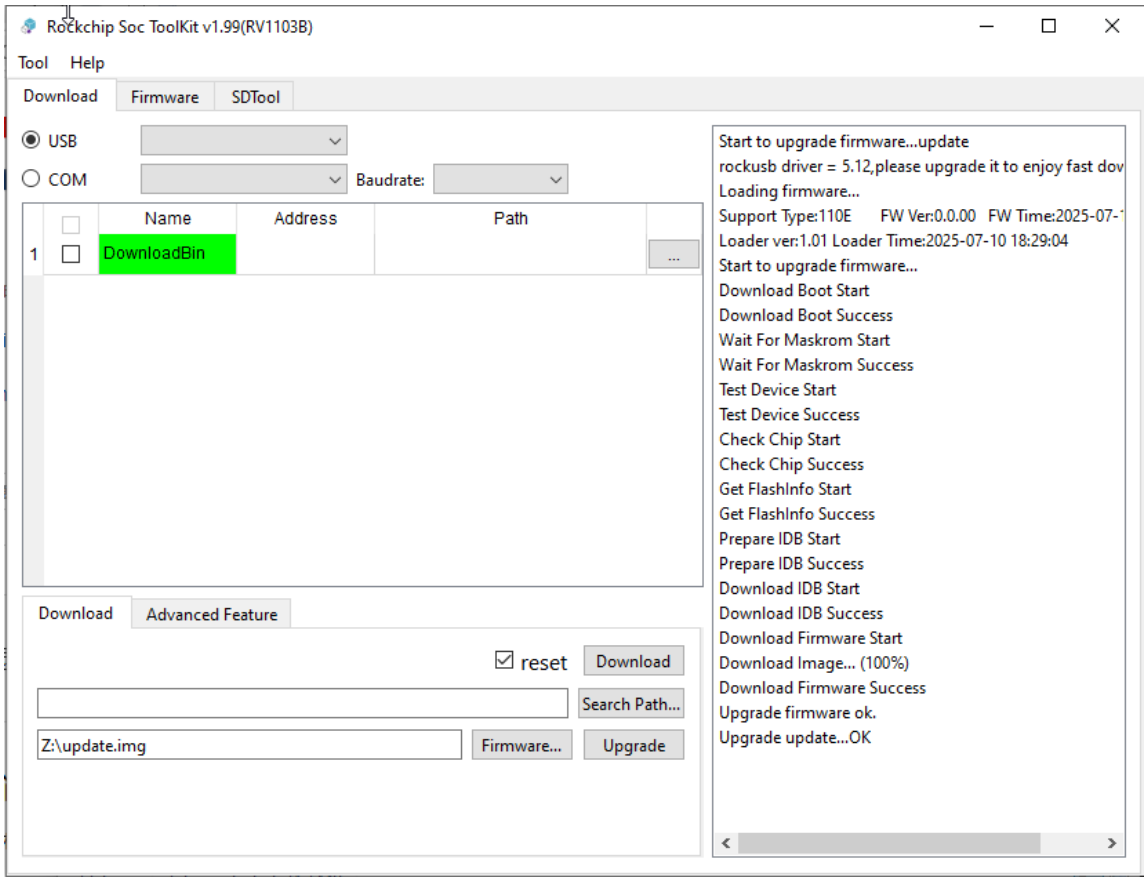
3.1.2 Burn Update.img Firmware

Environment: Windows OS (Operating System).

Click **Firmware**, select **update.img**, then click **Upgrade** to flash.

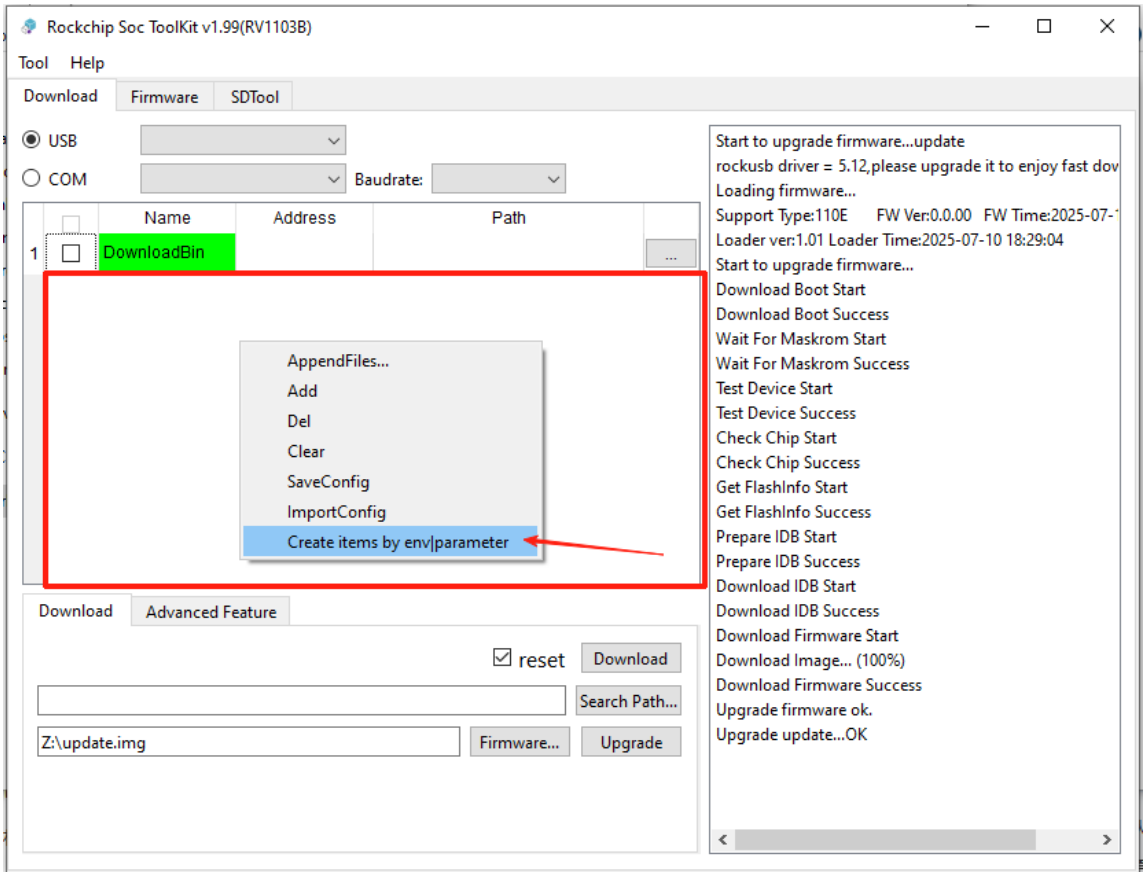


After the flashing is complete, the board will automatically reboot.

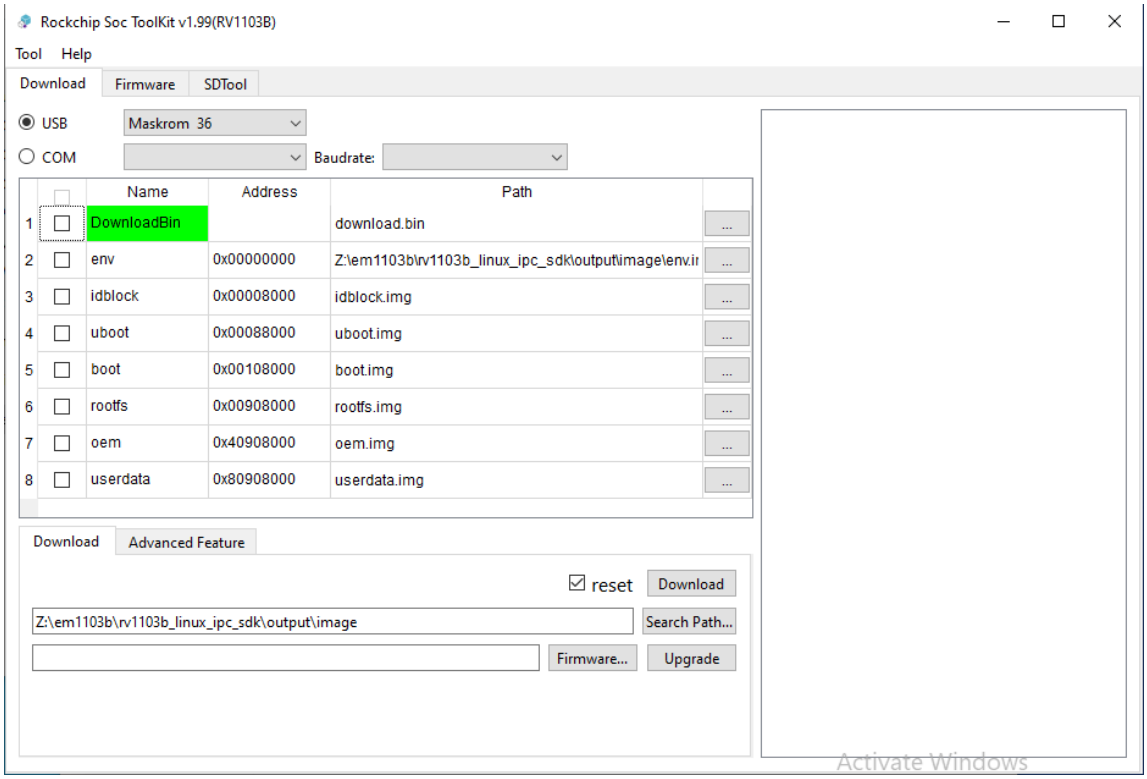


3.1.3 Burn Split Firmware

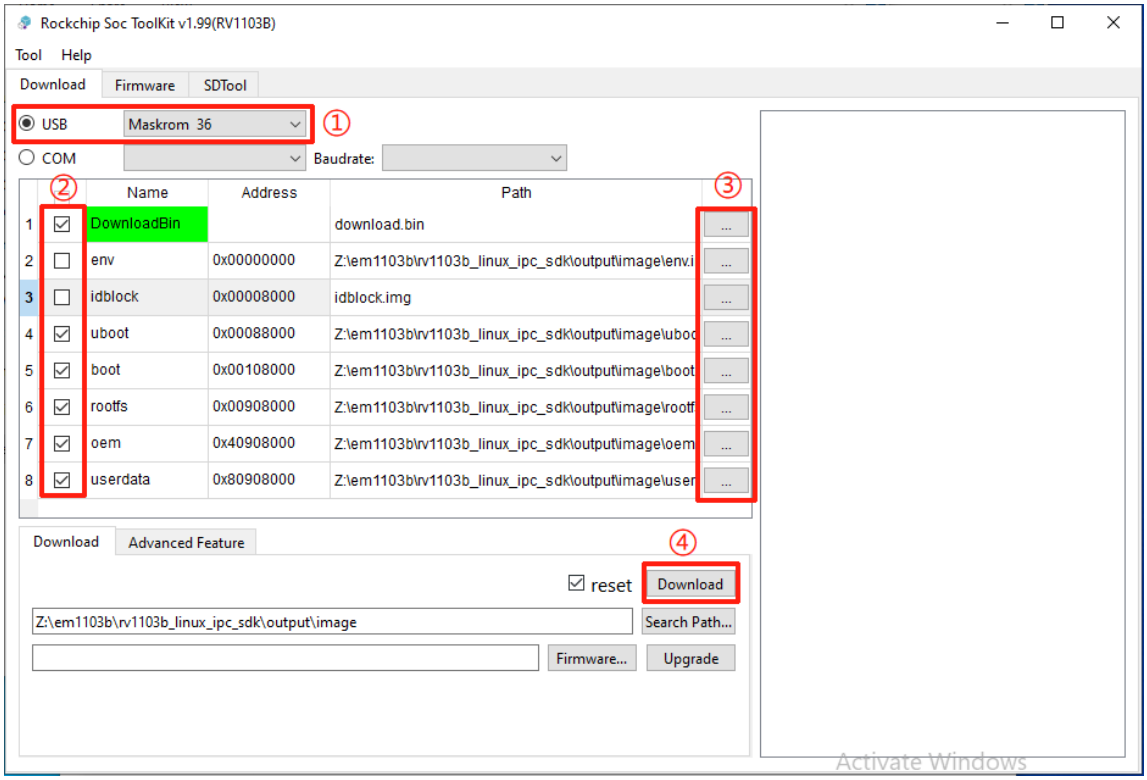
Step 1: Right-click within the red box, and then click "Create items by env|parameter".



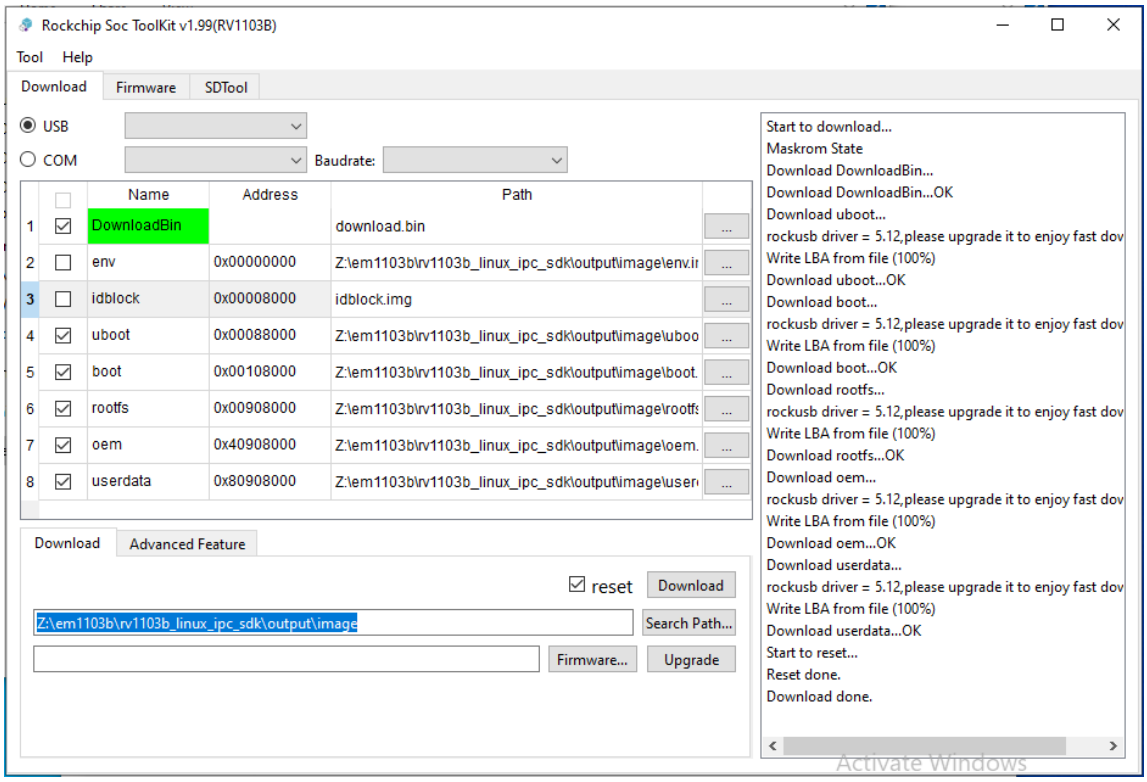
Step 2: Open the env.img file located in the **sdk/output/image/** directory.



Step 3: Click the **Run** button to flash the image.

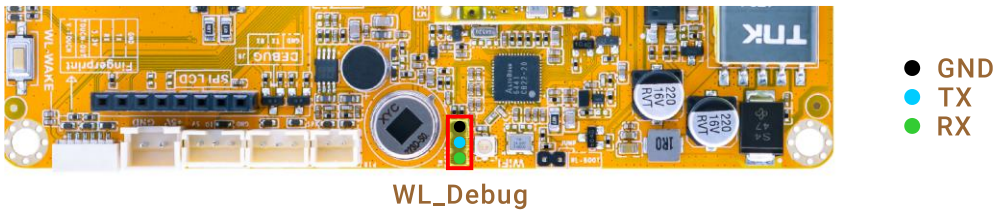


After the flashing is complete, the board will automatically reboot.

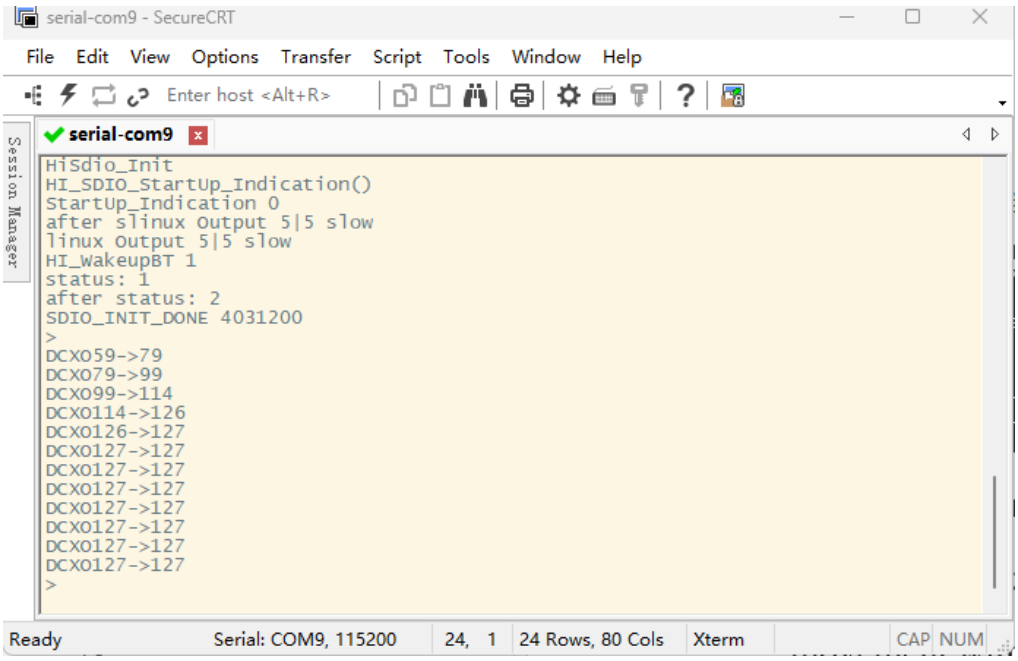


4.2 Burn ATBM6441 Firmware

The ATBM6441 chip uses UART to update the firmware or debug, The default UART programming baud rate is always 115200, use the serial port tool to connect WL_DEBUG and the PC.



WL_Debug



The ATBM6441 has two programming modes: Boot mode or ROM Code mode.

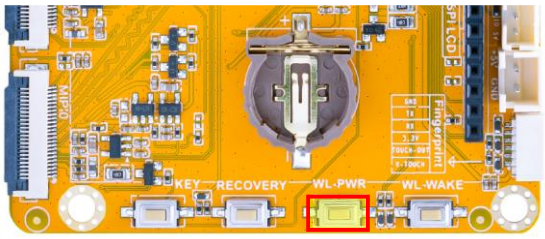
4.2.1 Boot mode

Step 1: BOOT_SEL pin is pulled high.

WL_Boot

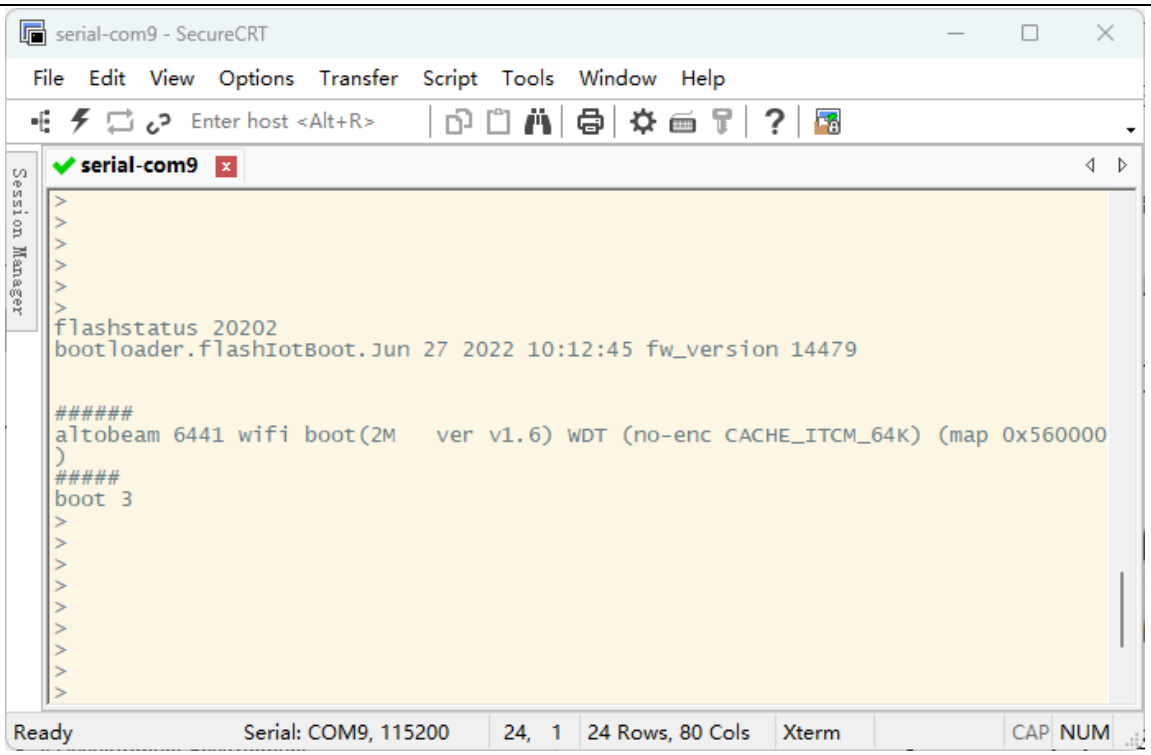
Step 2: Keep pressing the Enter key on the serial port tool.

Step 3: Press the WL-Power button.



WL-Power

Step 4: ATBM6441 enters Boot mode.

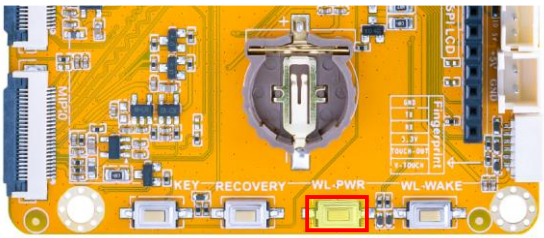


4.2.2 ROM Code mode

Step 1: Insert the jumper cap, BOOT_SEL pin is pulled low.

WL_Boot

Step 2: Press the WL-Power button.



WL-Power

Step 3: ATBM6441 enters ROM Code mode.

Step 4: Remove the jumper cap.

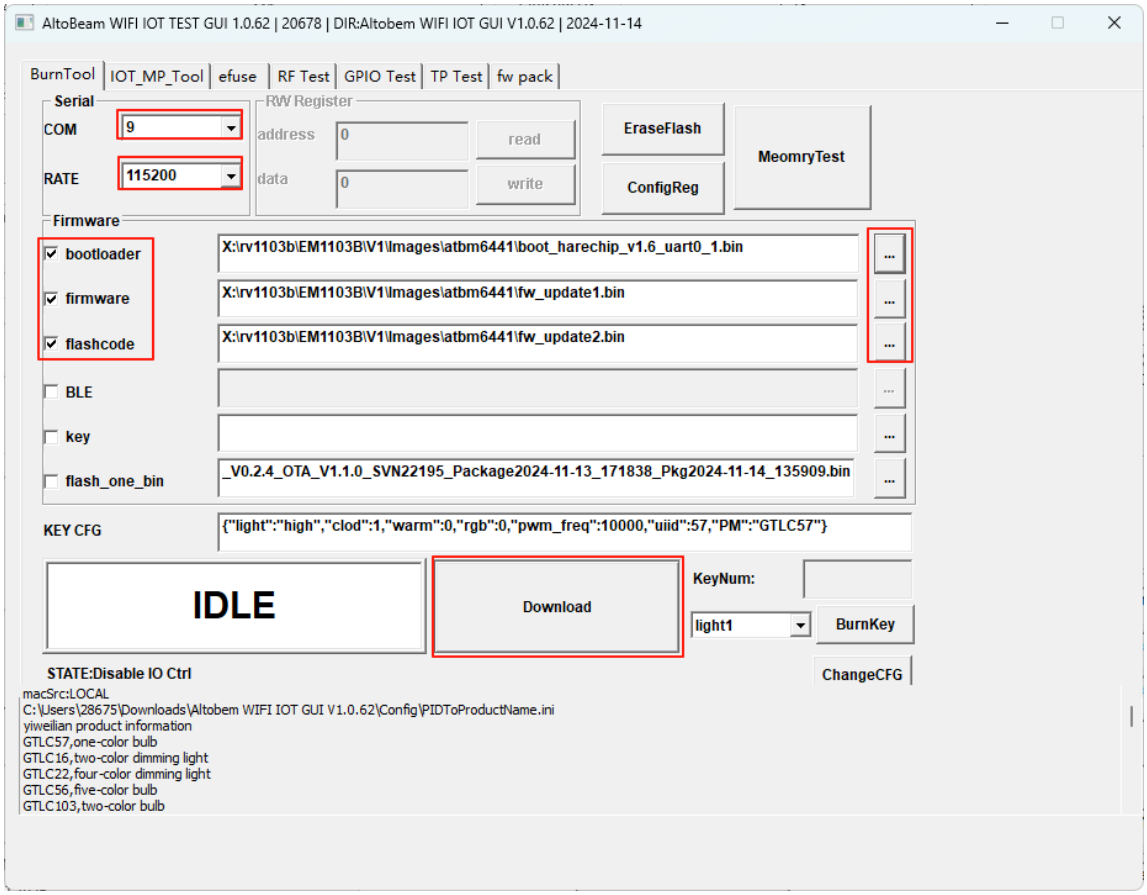
4.2.3 bootloader_GUI Tool

When performing the burning operation, please close the COM port that the serial port software has opened.

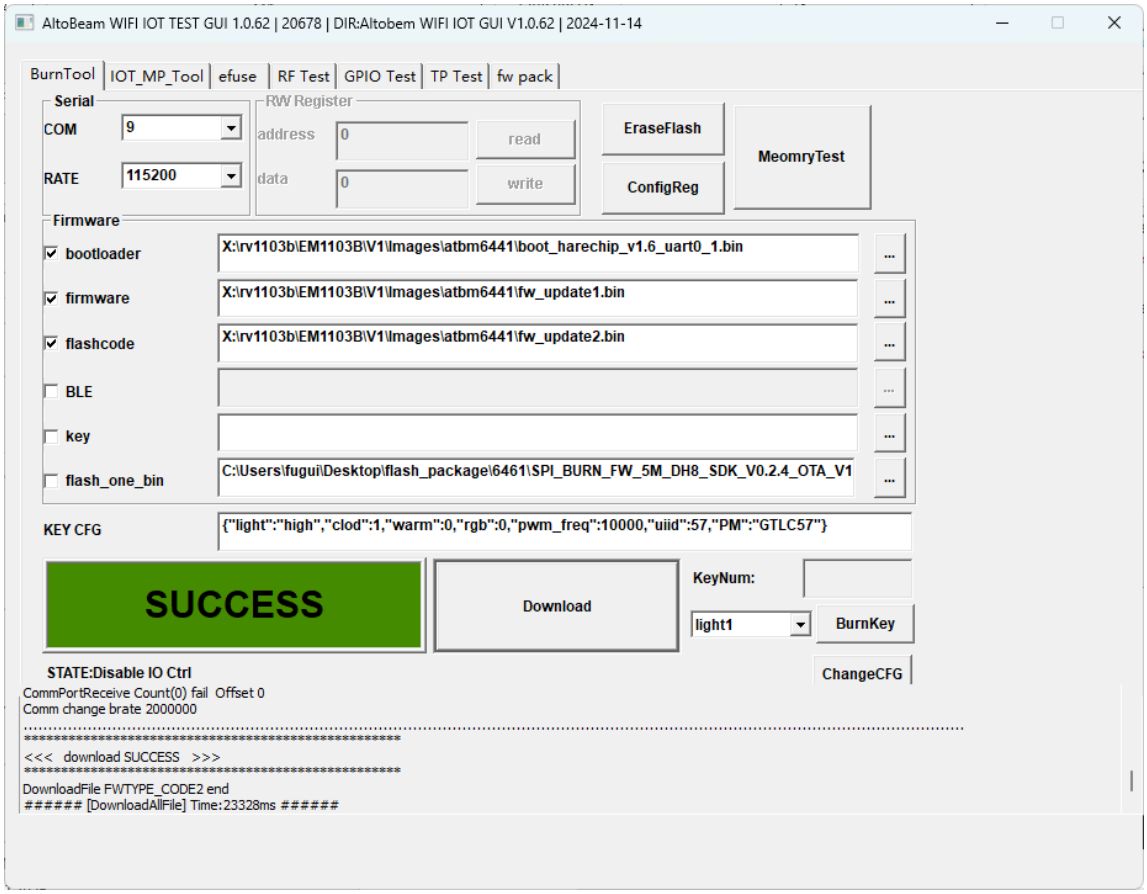
Step 1: Unzip Altobem WIFI IOT GUI V1.0.62.7z on Windows.

Step 2: Open Altobem WIFI IOT GUI V1.0.62/bootloader_GUI.exe.

Step 3: Set the COM port number and baud rate, select Firmware, click the "Download" button to start the burning process.



Step 4: Burn completed.



4.Development Environment

4.1 Preparing the Development Environment

It is recommended to use Ubuntu 18.04 or higher version for compilation. If you encounter an error during compilation, user can check the error message and install the corresponding software packages accordingly. Other Linux versions may need to adjust the software package accordingly. In addition to the system requirements, there are other hardware and software requirements.

Hardware requirements	Software requirements
64-bit system, hard disk space should be greater than 20G. If you do multiple builds, you will need more hard drive space.	Ubuntu 18.04

4.2 Installing Libraries and Toolkits

The contents of this directory only provide the software package installation commands that are needed to build the compiled SDK environment. Please install other tools such as samba and ssh yourself.

PC OS	Network	Permission
Ubuntu 18.04	online	root

To install the required tools, execute the following commands:

```
$ sudo apt-get install repo git ssh make gcc
$ sudo apt-get install gcc-multilib g++-multilib module-assistant
$ sudo apt-get install expect g++ gawk texinfo libssl-dev
$ sudo apt-get install bison flex fakeroot cmake unzip gperf autoconf
$ sudo apt-get install device-tree-compiler libncurses5-dev
```

5.Compile Source

5.1 EM1103B

Step 1: Unzip the Source

To extract the source files, execute the following commands:

```
$ tar xvf em1106b_em1103b_linux5.10_ipc_20250721.tar.bz2
$ cd em1106b_em1103b_linux5.10_ipc_20250721/
```

Step 2: Configure the Compiled Board

The board-level configuration of the SDK is located in the **"project/cfg/"** directory. The "BoardConfig.mk" file is an important file for SDK compilation. To configure the board, execute:

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

After running **./build.sh lunch**, the system will list available .mk files. Select the 17th option:

17.BoardConfig_IPC/BoardConfig-EMMC-NONE-EM1103B_EVB1_V10_V11-IPC_SINGLE_CAMERA.mk, this ensures compatibility with the EM1103B development board. Using other options may cause kernel mismatches and prevent proper booting.

```
xuejunchao@boardcon: ~/em1103b/em1106b_em1103b_linux5.10_ipc_20250721$ ./build.sh lunch
ls: cannot access 'BoardConfig*.mk': No such file or directory
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
----------------------------------------------------------------
17. BoardConfig_IPC/BoardConfig-EMMC-NONE-EM1103B_EVB1_V10_V11-IPC_SINGLE_CAMERA.mk
                            boot medium(启动介质): EMMC
                        power solution(电源方案): NONE
                    hardware version(硬件版本): EM1103B_EVB1_V10_V11
                            application(应用场景): IPC_SINGLE_CAMERA
----------------------------------------------------------------
Which would you like? [0]: 17
[build.sh:info] switching to board:
/home/xuejunchao/em1103b/em1106b_em1103b_linux5.10_ipc_20250721/project/cfg/BoardConfig_IPC/BoardConfig-
EMMC-NONE-EM1103B_EVB1_V10_V11-IPC_SINGLE_CAMERA.mk
[build.sh:info] Running build_select_board succeeded.
```

Step 3: One-click automatic compilation

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

Step 4: Compile U-Boot

To compile uboot, execute the following command:

```
$ ./build.sh clean uboot

$ ./build.sh uboot
```

Generate image files: **output/image/download.bin, output/image/idblock.img** and **output/image/uboot.img**.

Step 5: Compile the Kernel

To compile the kernel, execute the following command:

```
$ ./build.sh clean kernel

$ ./build.sh kernel
```

Generate image file: **output/image/boot.img**.

Step 6: Compile rootfs

To compile rootfs, execute the following command:

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

Generate image file: **output/image/rootfs.img**.

Step 7: Generate and Check Firmwares

To generate firmware, execute the following command:

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

Images and **update.img** are generated in **output/image/directory**.

Step 8: Compile media

To compile media, execute the following command:

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

Directory for storing generated files: **output/out/media_out/**.

Step 9: Compilation reference application

To compile reference application, execute the following command:

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

Directory for storing generated files: **output/out/app_out**.

Step 10…….: Compilation kernel driver

To compile kernel driver, execute the following command:

```
$ ./build.sh clean driver
$ ./build.sh driver
```

Directory for storing generated files:**output/out/sysdrv_out/kernel_drv_ko/**.

5.2 ATBM6441

Step 1: Unzip the Source

To extract the source files, execute the following commands:

```
$ tar xvf atbm6441_sdk_20250714.tar.bz2
$ cd atbm6441_sdk_20250714/
```

Step 2: Compile the Source

```
$ make clean
$ make all
```

After the compilation is completed, files **fw_update1.bin**, **fw_update2.bin**, **iot_6441.bin**, **OTA_6441.bin** and **OTA_6441_zip.bin** will appear in the **bin/** directory, indicating that the compilation was successful.

Note: If the following log appears during the compilation process, it will cause the compilation to fail.

```
error while loading shared libraries: libz.so.1: cannot open shared object file:
No such file or directory.
```

Please execute the following command to fix this issue.

```
$ sudo apt-get install lib32z1
```

6.EM1103B Test

6.1 ADB

USB OTG

Step 1: Connect the board and PC with USB OTG cable.

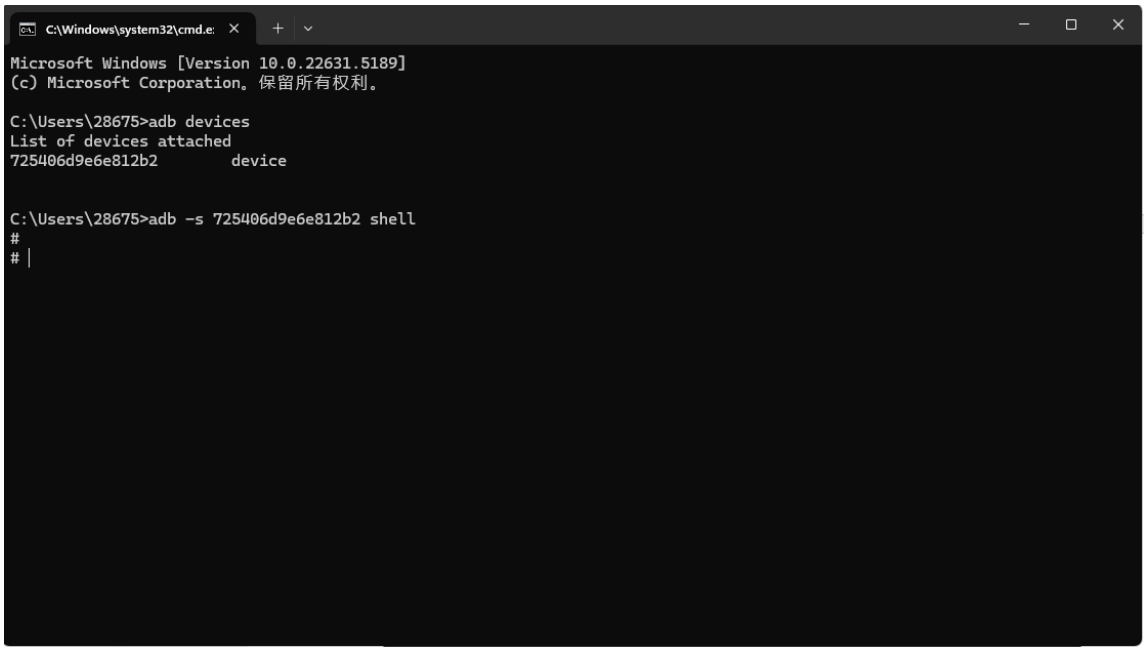
Step 2: Install ADB driver on Windows system.

Step 3: Press **Windows + R** to open the Run program. Type "cmd" and press Enter.

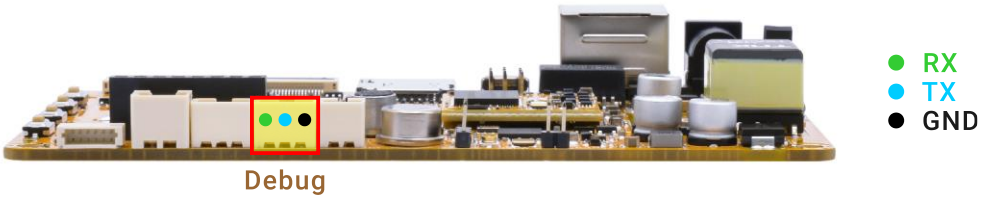
Step 4: Execute the following commands to view and connect adb device.

```
# adb devices
```

```
# adb -s <device number> shell
```



6.2 Serial Terminal



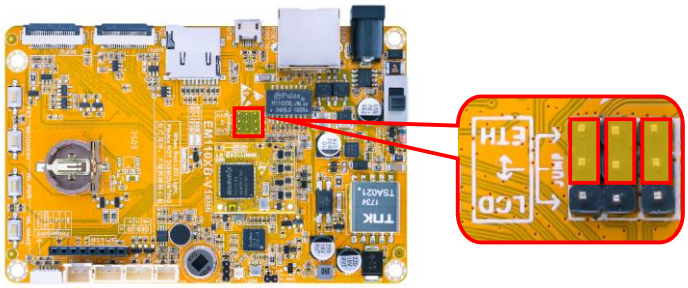
PC Serial Port Configuration Instructions: Baud rate: 115200, Data bits: 8, Parity: None, Stop bits: 1, Flow control: None.

Executing the following command can block WiFi-related information.

```
# echo 0 > /sys/module/atbm6041_wifi_sdio/parameters/atbm_printk_mask
```

6.3 Ethernet

Step 1: Connect the network cable to the Ethernet interface and insert these three jumper caps.



According to the log, it can be seen that the Gigabit Ethernet was successfully identified and an IP address was automatically obtained.

```
# [   24.777819] rk_gmac-dwmac 20800000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.0.9, server 192.168.0.1
udhcpc: lease of 192.168.0.9 obtained from 192.168.0.1, lease time 86400
deleting routers
adding dns 192.168.0.1
```

Step 2: View network interface information.

```
# ifconfig eth0
```

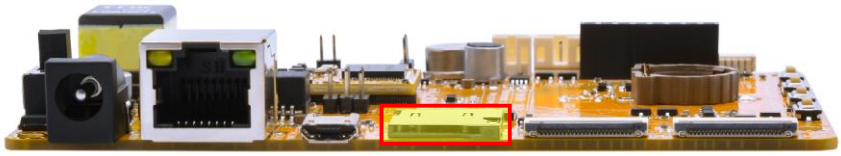
```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 26:7E:EF:44:02:76
          inet addr:192.168.0.9  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:513 errors:0 dropped:159 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:64325 (62.8 KiB)  TX bytes:684 (684.0 B)
          Interrupt:29
```

Step 3: Network connection test.

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

```
# ping -I eth0 www.baidu.com
PING www.baidu.com (183.2.172.17): 56 data bytes
64 bytes from 183.2.172.17: seq=0 ttl=54 time=6.973 ms
64 bytes from 183.2.172.17: seq=1 ttl=54 time=6.821 ms
64 bytes from 183.2.172.17: seq=2 ttl=54 time=6.867 ms
64 bytes from 183.2.172.17: seq=3 ttl=54 time=6.812 ms
64 bytes from 183.2.172.17: seq=4 ttl=54 time=6.681 ms
64 bytes from 183.2.172.17: seq=5 ttl=54 time=6.714 ms
^C
--- www.baidu.com ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 6.681/6.811/6.973 ms
```

6.4 TF Card



Micro SD

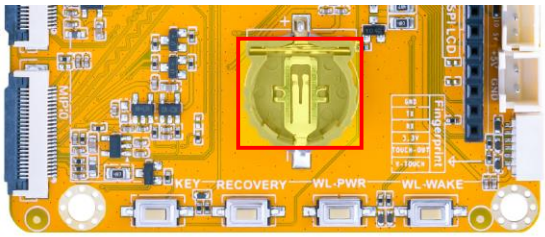
After inserting the TF card and turning on the machine, the TF card will be automatically mounted to Under /mnt/sdcard, execute the following command check TF card device.

```
# df -h
```

```
# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root               974.7M     15.7M    931.5M   2% /
devtmpfs                 23.1M         0     23.1M   0% /dev
tmpfs                    23.2M         0     23.2M   0% /dev/shm
tmpfs                    23.2M     12.0K     23.2M   0% /tmp
tmpfs                    23.2M    224.0K     23.0M   1% /run
/dev/mmcblk1p1           29.4G    223.0M     29.2G   1% /mnt/sdcard
```

6.5 RTC

RTC



Step 1: Set the system time.

```
# date -s "2025-01-08 11:00:00"
```

Step 3: Write the system time to the hardware clock.

```
# hwclock -w
```

Step 4: Display the current hardware clock time.

```
# hwclock
```

```
# date -s "2025-01-08 11:00:00"
Wed Jan  8 11:00:00 UTC 2025
# date
Wed Jan  8 11:00:02 UTC 2025
# hwclock -w
# hwclock
Wed Jan  8 11:00:12 2025  0.000000 seconds
# hwclock
Wed Jan  8 11:00:17 2025  0.000000 seconds
# hwclock
Wed Jan  8 11:00:21 2025  0.000000 seconds
```

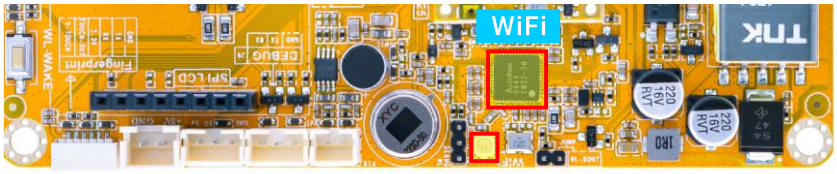
Step 5: Turn off the development board, and turn on after a period of time to check whether the time is saved.

```
# hwclock
Wed Jan  8 11:00:47 2025  0.000000 seconds
# hwclock
Wed Jan  8 11:00:57 2025  0.000000 seconds
# hwclock
Wed Jan  8 11:01:03 2025  0.000000 seconds
```

When the system has access to the network, the ntpd command can be used for time synchronization.

```
# ntpd -p 203.107.6.88 -qNn
```

6.6 WiFi



WiFi_ANT

Step 1: Check whether the wifi device **wlan0** exists.

```
# ifconfig wlan0
# ifconfig wlan0 up
```

```
# ifconfig wlan0 up
# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 38:54:F5:72:E8:DB
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:326 (326.0 B)  TX bytes:2172 (2.1 KiB)
```

Step 2: Scan the available Wi-Fi networks.

```
# atbm_iot_cli scan
```

```
# atbm_iot_cli scan
[  641.313298] [atbm_log]:atbm_sdio_tx_bh:cmd free(14),used(1)
cmd_line: scan
# OK
[  641.329076] [atbm_log]:atbm_sdio_tx_bh:cmd free(13),used(1)
[  641.839033] [atbm_log]:atbm_sdio_tx_bh:cmd free(12),used(1)
[  642.349009] [atbm_log]:atbm_sdio_tx_bh:cmd free(11),used(1)
[  642.859066] [atbm_log]:atbm_sdio_tx_bh:cmd free(10),used(1)
>> --- ioctl_msg_func  called  3060
Scan Completed...
CTRL-EVENT-SCAN-RESULTS
>> --- ioctl_msg_func  exit  3217
```

Step 3: View the scanning results.

```
# atbm_iot_cli scan_results
```

```
# atbm_iot_cli scan_results
[  658.200203] [atbm_log]:atbm_sdio_tx_bh:cmd free(9),used(1)
cmd_line: scan_results

get scan cnt: 13 (OVER)
bssid   /      level   /      channel /      ssid
b4:f1:8c:6d:d1:29      -33     1
d8:32:14:25:b7:a8      -53     1       LYB-2.4G
b2:22:7a:5a:b6:4a      -44     1       DIRECT-4A-HP Laser 136w
d8:fe:42:ce:f2:58      -68     1       DIRECT-Pl-HUAWEI CV81-WDMa
b4:f1:8c:6d:d1:24      -33     1       Boardcon
b4:f1:8c:fd:d1:29      -33     1       Boardcon_Wi-Fi5
b4:f1:8c:6d:d1:25      -34     1
e2:5d:54:de:58:86      -80     1
50:fa:84:cb:e4:35      -78     6       HYLS_2.4G
66:6c:80:30:bc:68      -81     6       DIRECT-68-HP M148 LaserJet
42:f5:20:c8:0f:bf      -87     10      LEDnet0006C80FBF
4c:10:d5:2a:d3:14      -71     11      yiqi8888
a0:10:77:09:db:9f      -72     11      ChinaNet-T74G
OK
```

Step 4: Set the SSID of the wireless network.

```
# atbm_iot_cli set_network ssid <SSID>
```

```
# atbm_iot_cli set_network ssid Boardcon
cmd_line: set_network ssid Boardcon
OK
```

Step 5: Set the password for the wireless network.

```
# atbm_iot_cli set_network key <PASSWORD>
```

```
# atbm_iot_cli set_network key Boardcon43435656
cmd_line: set_network key Boardcon43435656
OK
```

Step 6: Set the security type of the wireless network.

```
# atbm_iot_cli set_network key_mgmt WPA2
```

```
# atbm_iot_cli set_network key_mgmt WPA2
cmd_line: set_network key_mgmt WPA2
OK
```

Step 7: Enable wireless network connection.

```
# atbm_iot_cli enable_network
```

```
[   36.878085] [atbm_ioctl_connect_async]:(171) called
[   36.878085]
>> --- ioctl_msg_func  called  3060
[   40.340017] hostevent->bssid b4:f1:8c:6d:d1:24 ipaddr 4200a8c0
CTRL-EVENT-STATE-CONNECTED - Connection to b4:f1:8c:6d:d1:24 completed
[   40.340051] [atbm_log]:atbm_ioctl_notify_add enter
ip addr:192.168.0.66
[   40.340060] [atbm_log]:atbm_ioctl_notify_add need async notify usr layer
>> --- ioctl_msg_func  exit  3217
[   40.340068] [atbm_ioctl_connect_async]:(171) called
ANALYZER:E:cid:1, fid:431, no free aiq params buffer!
```

Step 8: Check the status of the wireless network connection.

```
# atbm_iot_cli status
```

```
cmd_line: status
[   41.291389] [atbm_log]:atbm_sdio_tx_bh:cmd free(18),used(1)
wifi_mode=STA
wpa_state=ACTIVE
ssid=Boardcon
ip_address=192.168.0.66
mac_address=b4:f1:8c:6d:d1:24
ip_mask=255.255.255.0
gate_way=192.168.0.1
OK
```

Step 9: Add the default gateway.

```
# route add default gw <Default Gateway>
```

Step 10: Configure the DNS server.

```
# echo 'nameserver <DNS>' > /etc/resolv.conf
```

Step 11: View **wlan0** IP address.

```
# ifconfig wlan0
```

```
# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 38:54:F5:72:E8:DB
          inet addr:192.168.0.66  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:538 (538.0 B)  TX bytes:1448 (1.4 KiB)
```

Step 12: Internet test.

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

```
# ping -I wlan0 www.baidu.com
PING www.baidu.com (183.2.172.177): 56 data bytes
64 bytes from 183.2.172.177: seq=0 ttl=54 time=17.720 ms
64 bytes from 183.2.172.177: seq=1 ttl=54 time=11.943 ms
64 bytes from 183.2.172.177: seq=2 ttl=54 time=8.811 ms
64 bytes from 183.2.172.177: seq=3 ttl=54 time=10.622 ms
64 bytes from 183.2.172.177: seq=4 ttl=54 time=10.815 ms
64 bytes from 183.2.172.177: seq=5 ttl=54 time=9.343 ms
^C
--- www.baidu.com ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 8.811/11.542/17.720 ms
```

6.7 Fingerprint

Step 1: Connect the fingerprint module to the fingerprint interface.

Step 2: Execute the command to conduct the fingerprint test.

```
# Fingerprint
```

```
# Fingerprint
port = /dev/ttyS2 nbaudrate = 57600
Please enter command (0-12), or type 100 to show.
```

Enter the number 100 to display the help information, listing all the commands and their functions.

```
100
command 0: get finger device ID
command 1: enroll finger
command 2: match finger
command 3: delete finger
command 4: deivce enter sleep
command 5: updateFeature
command 6: autoEnroll
command 7: matchSyn
command 8: deleteSyn
command 9: uploadFTR
command 10: downloadFTR
command 11: set green led
command 12: setSysPolicy
command 100: show help
Please enter command (0-12), or type 100 to show.
```

Enter the number 1 for fingerprint registration.

```
1
Lift your finger and touch the sensor again.
Lift your finger and touch the sensor again.
. . . . . .
Lift your finger and touch the sensor again.
Lift your finger and touch the sensor again.
Fingerprint saved successfully. The saved ID is: [0]
Please enter command (0-12), or type 100 to show.
```

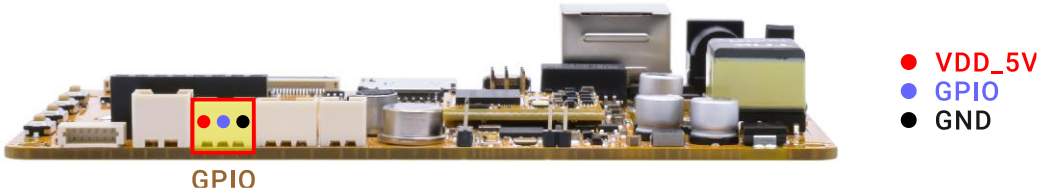
The program has completed the fingerprint entry process and successfully saved the fingerprint. The assigned ID is [0].

Enter the number 2 for fingerprint matching.

```
2
Match ID 0
Please enter command (0-12), or type 100 to show.
```

The program has completed the fingerprint matching process, and the matched fingerprint ID is [0].

6.8 GPIO



GPIO10 is actually directly controlled by ATBM6441. AT commands can be sent to ATBM6441 to use GPIO10.

Set GPIO10 to output mode, DIR 1: output, 0: input.

```
# atbm_iot_cli fw_cmd "AT+GPIO_SET_DIR PIN 10 DIR 1"
```

```
# atbm_iot_cli fw_cmd "AT+GPIO_SET_DIR PIN 10 DIR 1"
[ 2198.666122] [atbm_log]:atbm_sdio_tx_bh:cmd free(25),used(1)
cmd_line: fw_cmd AT+GPIO_SET_DIR PIN 10 DIR 1
# OK
```

Set GPIO10 to output 0.

```
# atbm_iot_cli fw_cmd "AT+GPIO_WRITE PIN 10 VALUE 0"
```

```
# atbm_iot_cli fw_cmd "AT+GPIO_WRITE PIN 10 VALUE 0"
[ 2275.051563] [atbm_ioctl_conn_err_async]:(229) called
cmd_line: fw_cmd AT+GPIO_WRITE PIN 10 VALUE 0
[ 2275.051563]
# OK
```

Set GPIO10 to output 1.

```
# atbm_iot_cli fw_cmd "AT+GPIO_WRITE PIN 10 VALUE 1"
```

```
# atbm_iot_cli fw_cmd "AT+GPIO_WRITE PIN 10 VALUE 1"
[ 2387.051522] [atbm_ioctl_conn_err_async]:(229) called
cmd_line: fw_cmd AT+GPIO_WRITE PIN 10 VALUE 1
[ 2387.051522]
# OK
```

Set GPIO10 to input mode.

```
# atbm_iot_cli fw_cmd "AT+GPIO_SET_DIR PIN 10 DIR 0"
```

```
# atbm_iot_cli fw_cmd "AT+GPIO_SET_DIR PIN 10 DIR 0"
[   41.450624] [atbm_ioctl_conn_err_async]:(229) called
cmd_line: fw_cmd AT+GPIO_SET_DIR PIN 10 DIR 0
[   41.450624]
# OK
[   44.469626] [atbm_log]:atbm_sdio_tx_bh:cmd free(28),used(1)
```

ATBM6441 is required to return information to EM1103B; otherwise, the status information of GPIO10 will only be viewed on the serial port of ATBM6441.

```
# atbm_iot_cli fw_print 1
```

```
# atbm_iot_cli fw_print 1
[  258.488841] [atbm_ioctl_conn_err_async]:(229) called
cmd_line: fw_print 0
[  258.488841]
OK
[  288.414296] [atbm_log]:atbm_sdio_tx_bh:cmd free(27),used(1)
```

Input a 5V high level to GPIO10, and then read the status of GPIO10.

```
# atbm_iot_cli fw_cmd "AT+GPIO_READ PIN 10"
```

```
# atbm_iot_cli fw_cmd "AT+GPIO_READ PIN 10"
[  359.062844] [atbm_log]:atbm_sdio_tx_bh:cmd free(23),used(1)
cmd_line: fw_cmd AT+GPIO_READ PIN 10
# OK
[  361.752997] pin:10 level:1
```

At this point, the pin data of GPIO10 read is 1. If GPIO10 is grounded, the pin data of GPIO is 0. This will not be elaborated here.

6.9 RKIPC

To run RKIPC successfully, please be sure to prepare the following environment. Otherwise, RKIPC cannot start up normally.

Step 1: Power off.

Step 2: Connect the two SC500AI camera modules to the camera0 and camera1 interfaces respectively.

Step 3: Power on.

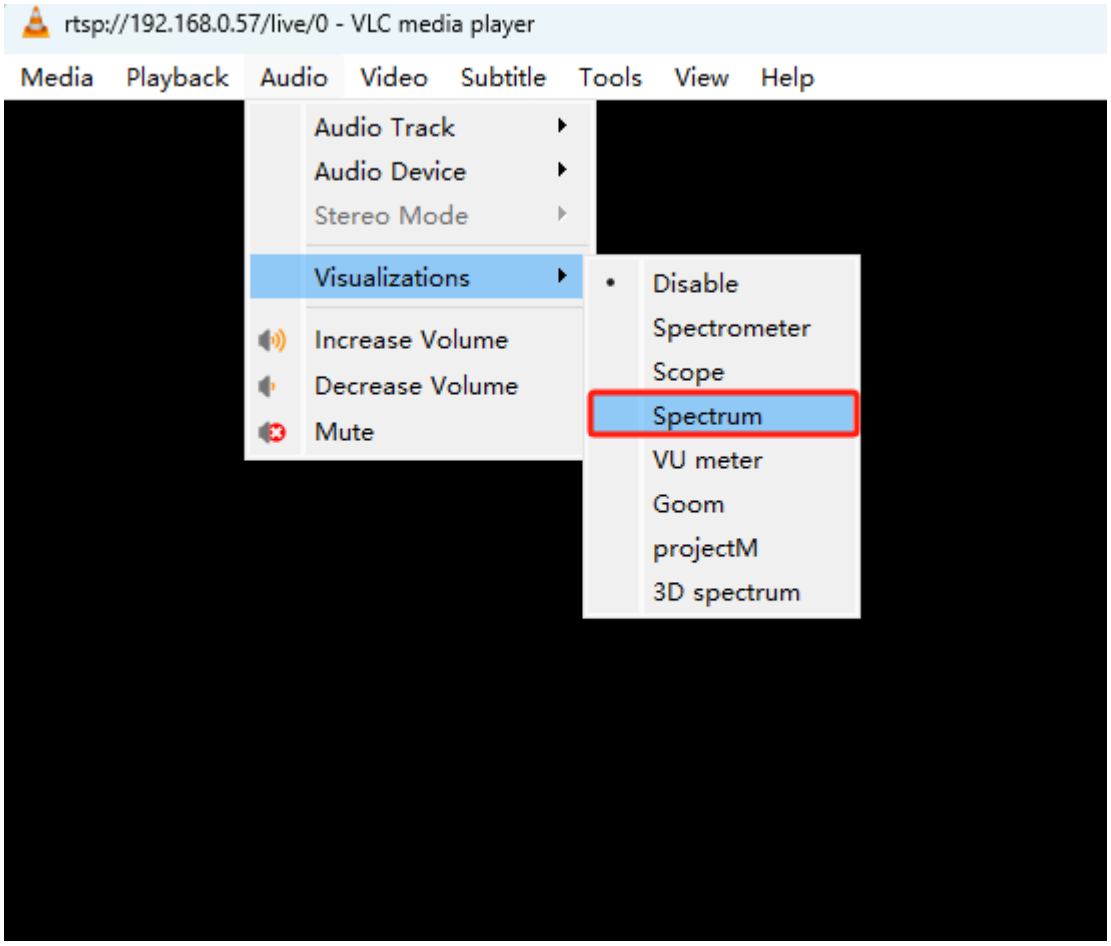
Executing the following instructions to ensure that the RKIPC process is running

```
# ps | grep rkipc
```

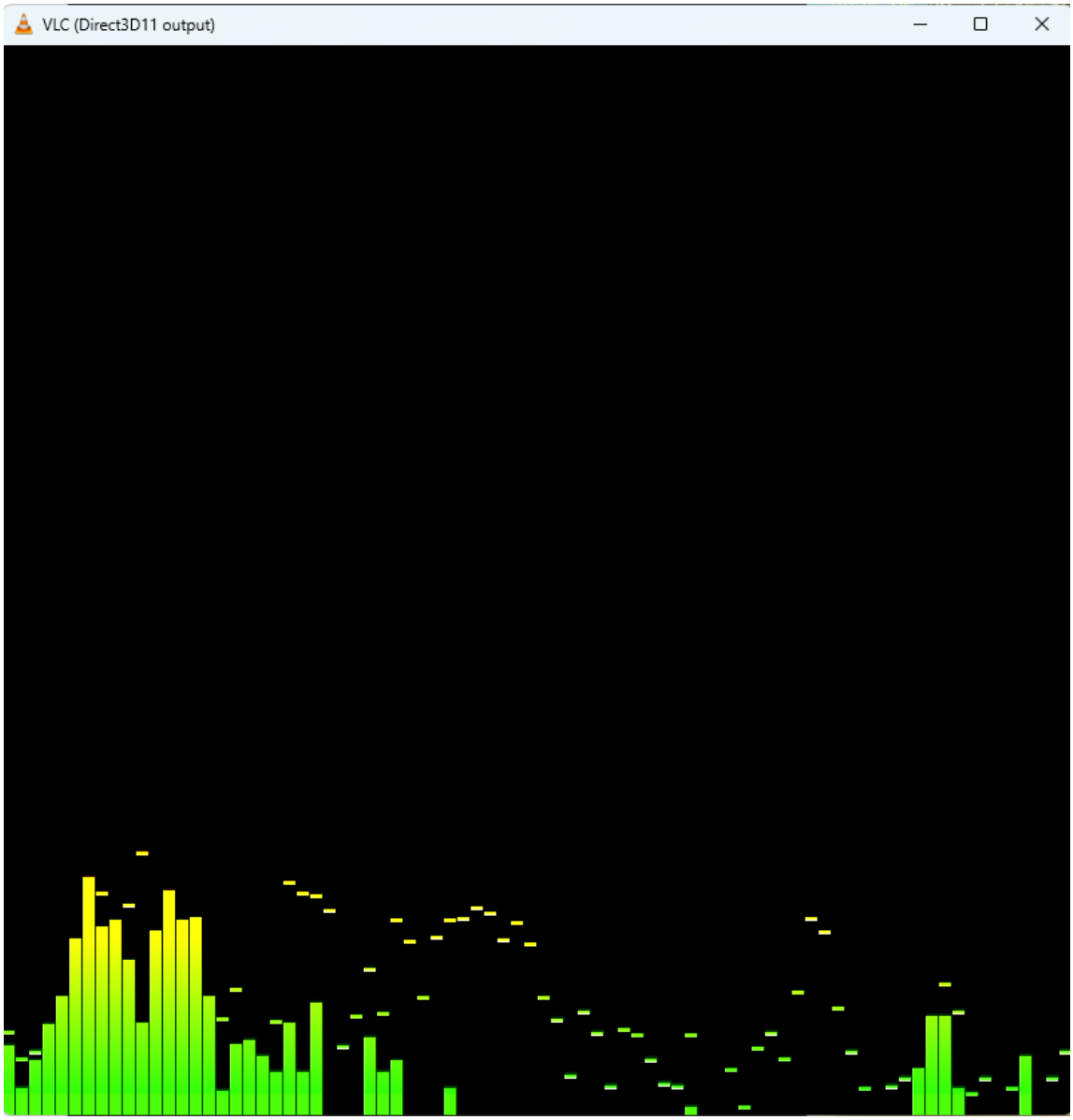
```
# ps | grep rkipc
  433 root      1234 S    rkipc -a /oem/usr/share/iqfiles
  574 root      1176 S    grep rkipc
```

6.9.1 Audio (MIC) test

The RTSP bitstream preview can be accessed through the PC end, and the audio can be obtained simultaneously. The audio can be played directly from the speaker or detected using the visual audio provided by the software. The method for viewing the audio spectrum is as follows:

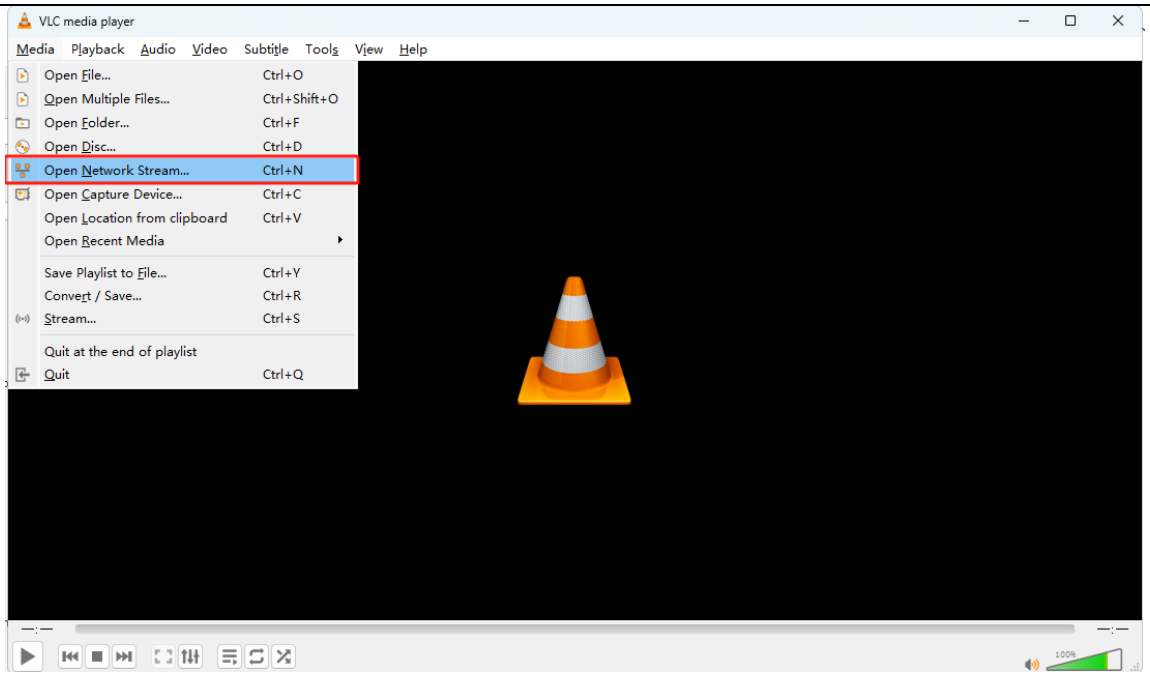


At this point, when speaking to the MIC, the spectrum effect is as follows:



6.9.2 RTSP

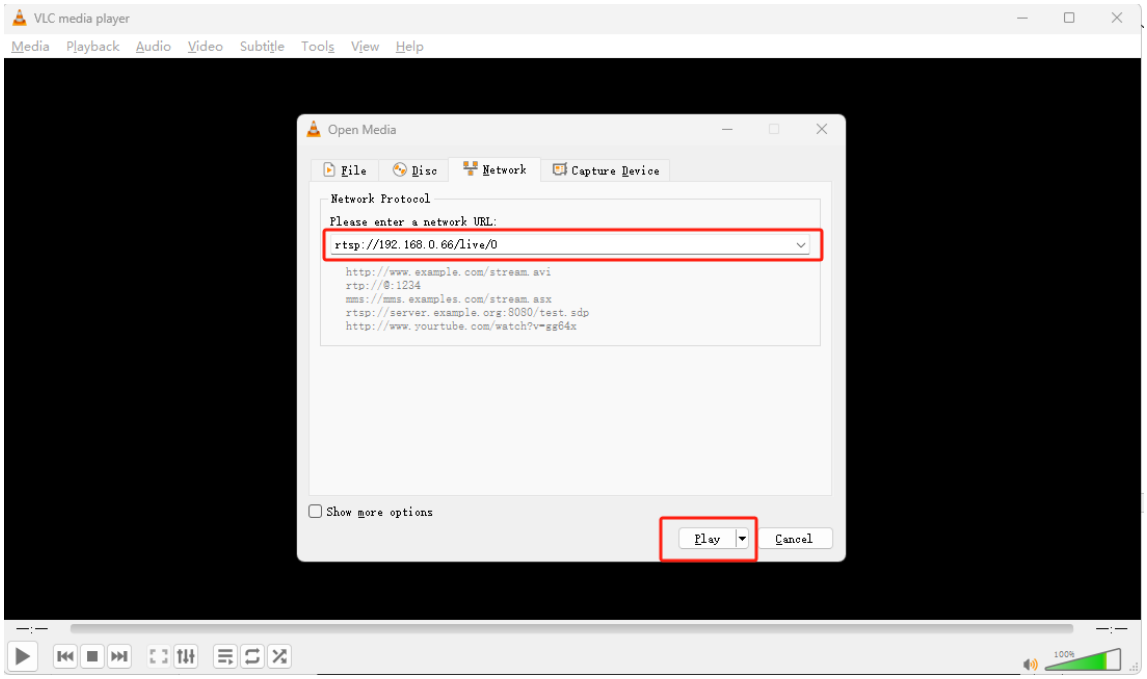
The video footage from the camera can be previewed within the same local area network. After the device is connected to the network, the RTSP software on the PC (such as VLC) can be used to enable network streaming.

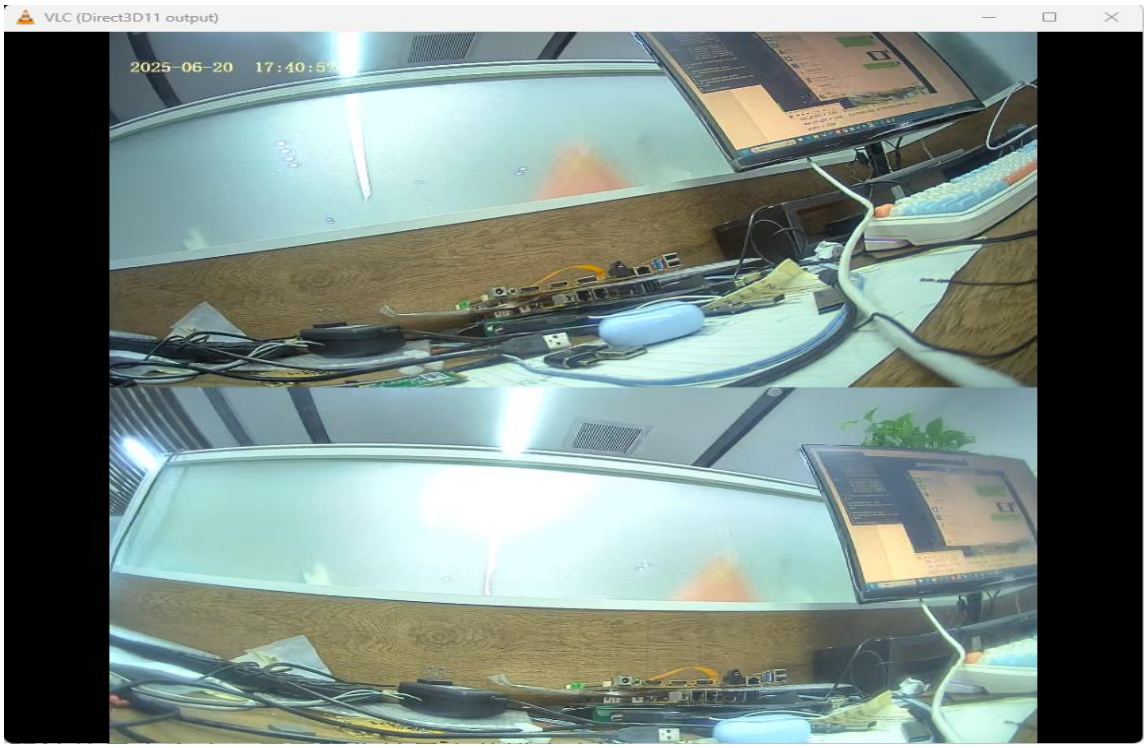


Enter the following address:

```
# rtsp:// (IP address of the device) /live/0
```

In section 6.2, for example, access to the IP address of the Ethernet as 192.168.0.66, then input rtsp://192.168.0.66/live/0, click the play button to preview the images of two cameras splicing video, push through the WiFi module flow in the same way.



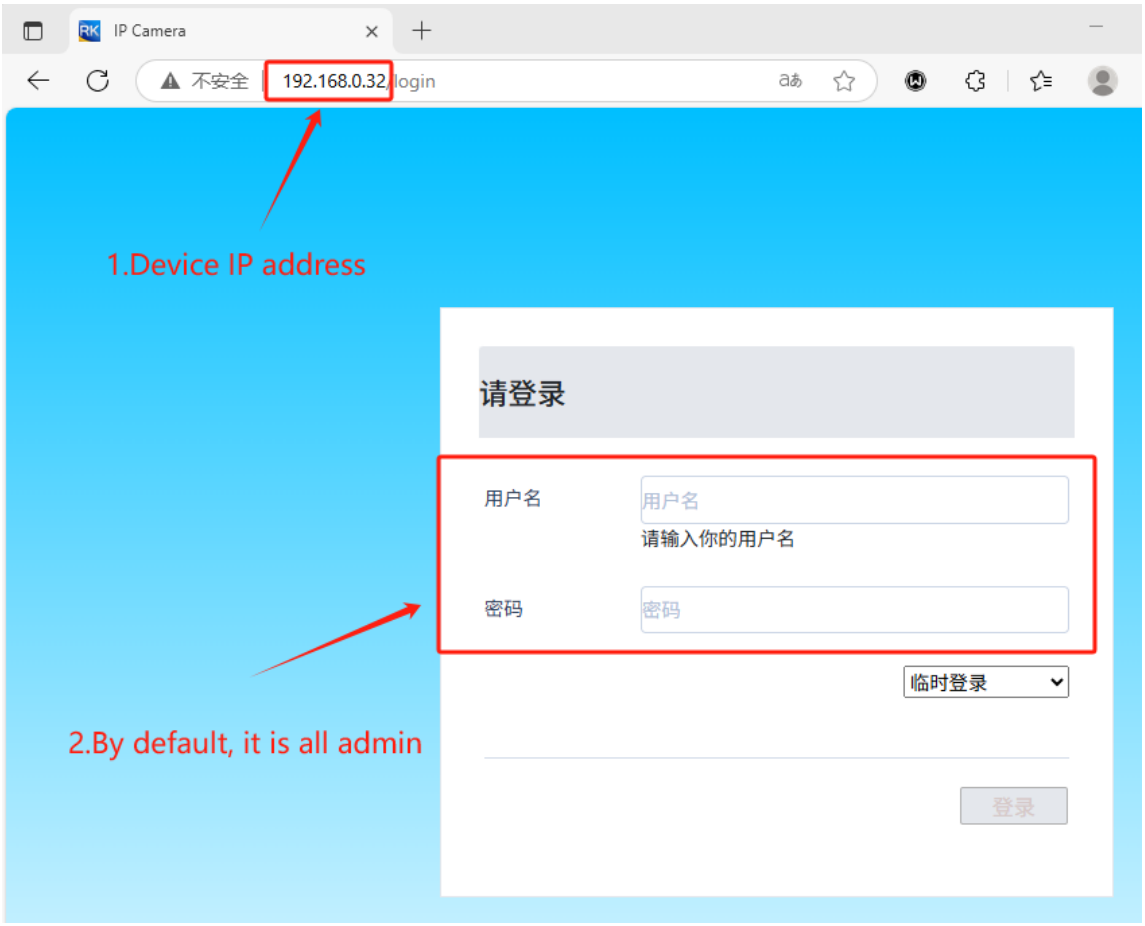


The video captured by the camera will be saved in the form of MP4 video files under /mnt/sdcard/video0/.

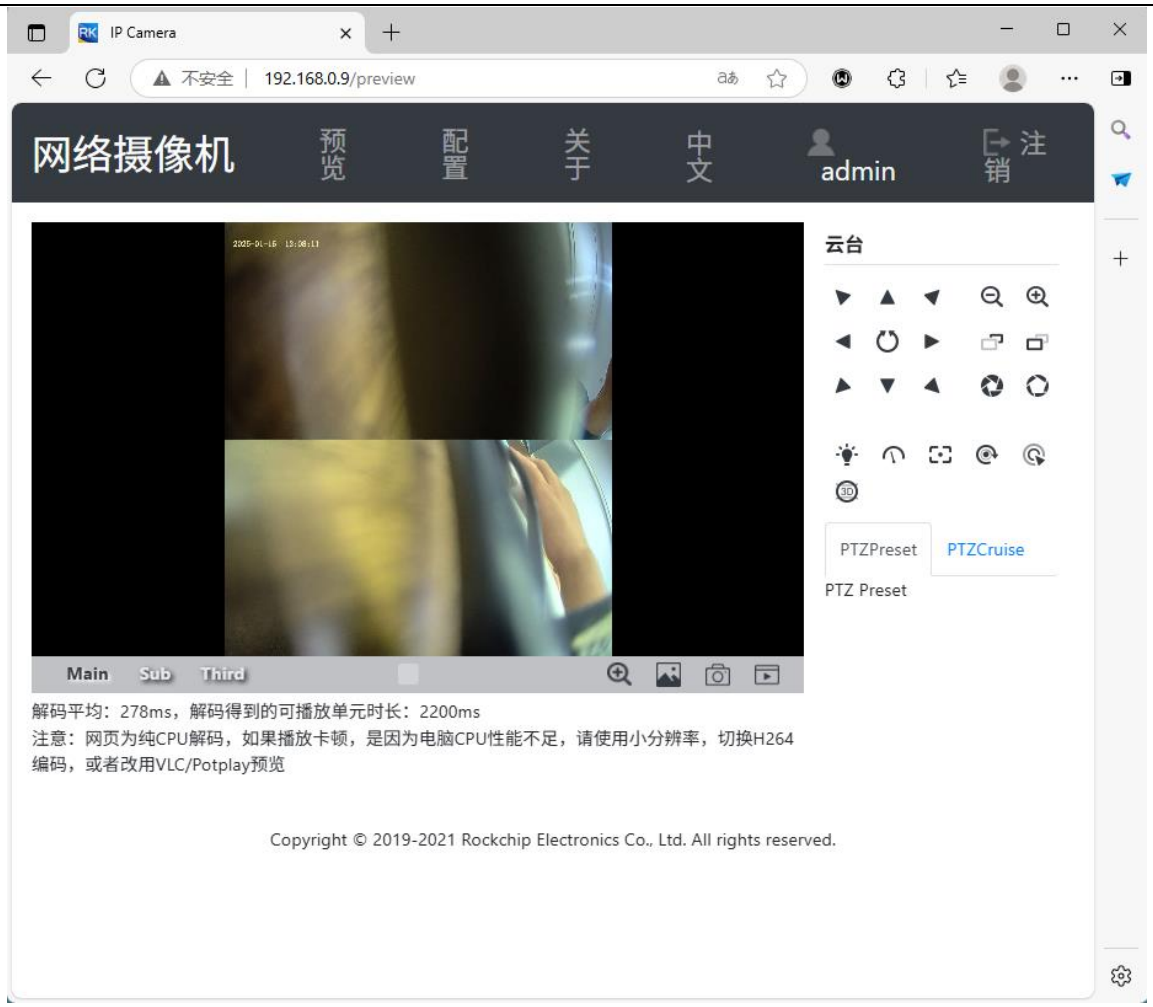
```
# ls /mnt/sdcard/video0/
20250111064442.mp4  20250111061042.mp4  20250111053641.mp4  20250111050241.mp4
20250111064342.mp4  20250111060942.mp4  20250111053541.mp4  20250111032207.mp4
20250111064242.mp4  20250111060842.mp4  20250111053441.mp4  20250111032107.mp4
20250111064142.mp4  20250111060742.mp4  20250111053341.mp4  20250111032007.mp4
20250111064042.mp4  20250111060642.mp4  20250111053241.mp4  20250111031907.mp4
20250111063942.mp4  20250111060542.mp4  20250111053141.mp4  20250111031807.mp4
20250111063842.mp4  20250111060442.mp4  20250111053041.mp4  20250111031707.mp4
20250111063742.mp4  20250111060342.mp4  20250111052941.mp4  20250111031607.mp4
```

6.9.3 Web Live Streaming

The Web end supports main stream, sub-stream, and H264/H265 live streaming. Access the device's IP address through a browser, enter the username and password (both default to admin), and enter the live preview system.



The preview effect is as follows:



6.9.2 IVS

Motion Detection: When an object moves within the camera lens, the terminal will print out a log.

```
[video.c][rkipc_ivs_get_results]:MD: md_area is 512000, md_area_threshold is 155520
```

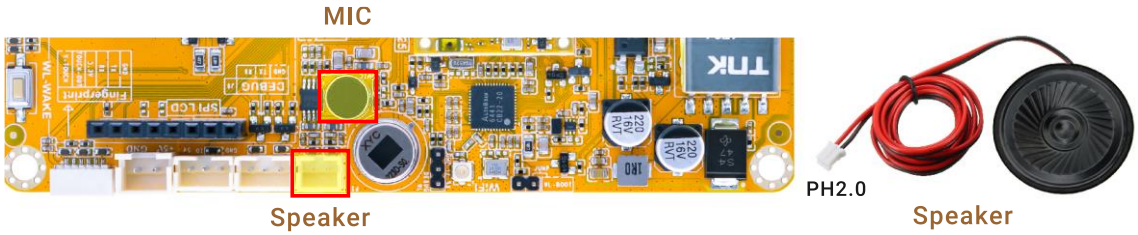
The currently detected motion area size (md_area) is 512,000, which is much larger than the preset threshold (md_area_threshold) of 155,520. This indicates that the detected motion area is extremely large.

Obstruction Detection: When the entire lens of the camera is blocked by an object, the terminal will print out a log.

```
[video.c][rkipc_ivs_get_results]:OD flag:1
```

The flag indicating object detection is set to 1.

6.10 Audio



Step 1: Connect the speaker and the Speaker interface.

Step 2: Since RKIPC occupies the driver devices of the speaker and microphone, it is necessary to kill the RKIPC process first before customizing the use of the speaker and microphone.

```
# killall rkipc
```

Step 3: MIC recording.

```
# rk_mpi_amix_test --control "ADC ALC Left Volume" --value "31"
# rk_mpi_ai_test --device_rate 44100 --device_ch 2 --out_rate 44100 --out_ch 2 -
o ./ --sound_card_name hw:0,0 --set_volume 100 & sleep 10&&killall rk_mpi_ai_test
```

```
# rk_mpi_amix_test --control "ADC ALC Left Volume" --value "31"
cmd parse result:
sound control id      : 0
control name          : ADC ALC Left Volume
control value         : 31
list controls         : 0
list contents         : 0
# rk_mpi_ai_test --device_rate 44100 --device_ch 2 --out_rate 44100 --out_ch 2 -
o ./ --sound_card_name hw:0,0 --set_volume 100 & sleep 10&&killall rk_mpi_ai_tes
t
cmd parse result:
[ 1015.851345] venc_release 6
input  file name      : (null)
output file name      : ./
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, can use export rt_log_path=, export rt_log_size= change
log_file = (nil)
(null)          03:05:22-906 {log_level_init    :190}

 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)          03:05:22-907 {read_log_level    :077} text is all=4
(null)          03:05:22-907 {read_log_level    :079} module is all, log_level is 4
(null)          03:05:22-907 {dump_version      :055} ------------------------------------------------------
----
(null)          03:05:22-907 {dump_version      :056} rockit version: git-82723ff48 Fri Feb 7 10:14:47 2025
+0800
(null)          03:05:22-907 {dump_version      :057} rockit building: built-Chu 2025-02-07 14:34:36
(null)          03:05:22-908 {dump_version      :058} ------------------------------------------------------
----
vsys dev open 4
[00000.56722] (null)(908): vrgn_thread_fn       [102] dev(vrgn-13) register ok(null)          03:05:22-909
{monitor_log_level :131} #Start monitor_log_level thread, arg:(nil)
cmpi              03:05:22-909 {sys_runtime_librar:180} Failed to open /dev/mpi/vdec,please check insmod
success or fail
cmpi            03:05:22-909 {sys_runtime_librar:180} Failed to open /dev/mpi/vo,please check insmod success
or fail
cmpi            03:05:22-914 {RKIsp_GetIspSubdev:131} not find rkisp* model in this media device
cmpi            03:05:22-914 {RKIsp_IspInitDevic:513} fail to getIspSubdevs, not found isp topology
cmpi            03:05:22-915 {RKIsp_GetIspSubdev:131} not find rkisp* model in this media device
cmpi            03:05:22-915 {RKIsp_IspInitDevic:513} fail to getIspSubdevs, not found isp topology
load library(librga.so) in releative path
cmpi            03:05:22-919 {rk_mpi_ai_test_ent:1900} start running loop count  = 0
cmpi            03:05:22-919 {RKAIPrepare       :294} HAVE_MODULE_SKV && HAVE_MODULE_SED
cmpi             03:05:22-919 {RKAIPrepare       :304} check enabled - file_read:0 vqe:0 aed:0 bcd:0 buz:0
gbs:0
cmpi            03:05:22-919 {RKAISetQueueCnt   :1922} sed queue len = 0
cmpi            03:05:22-919 {RKAIPrepare       :350} mb pool creat success, MBCnt:8
card(hw:0,0): ts_type = BOOTTIME
cmpi            03:05:23-039 {test_init_mpi_ai  :889} Set volume curve type: 0
cmpi            03:05:23-039 {commandThread     :1576} test info : mute = 0, volume = 100
cmpi            03:05:23-041 {RKAISaveFile      :1057} create file(.//cap_out.pcm) succeed
cmpi            03:05:23-041 {getDataThread     :1288} getDataThread - fd get enable
[ 1040.286901] vsys dev open 4
[1]+  Terminated              rk_mpi_ai_test --device_rate 44100 --device_ch 2 --out_rate 44100 --out_ch
2 -o ./ --sound_card_name hw:0,0 --set_volume 100
[ 1050.301169] vi_release 12
# [ 1050.301333] venc_release 6
```

The first command is used to increase the ADC gain, and the second command is used for recording for up to 10 seconds.

Step 4: Speaker play audio.

```
# rk_mpi_ao_test -i cap_out.pcm --input_ch 2 --input_rate 44100 --device_rate
44100 --device_ch 2 --sound_card_name hw:0,0 --set_volume 100
```

```
# rk_mpi_ao_test -i cap_out.pcm --input_ch 2 --input_rate 44100 --device_rate 44
100 --device_ch 2 --sound_card_name hw:0,0 --set_volume 100
cmd parse result:
input  file name      : 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, can use export rt_log_path=, export rt_log_size= change
log_file = (nil)
(null)          05:29:07-342 {log_level_init    :190}

 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
[00000.547] (null)(344): vrgn_thread_fn [102] dev(vrgn-13) register ok(null)              05:29:07-344
{read_log_level    :077} text is all=4
(null)          05:29:07-345 {read_log_level    :079} module is all, log_level is 4
(null)          05:29:07-346 {dump_version      :055} ----------------------------------------------------
------
(null)           05:29:07-346 {dump_version      :056} rockit version: git-82723ff48 Fri Feb 7 10:14:47
2025 +0800
(null)          05:29:07-346 {dump_version      :057} rockit building: built-Chu 2025-02-07 14:34:36
(null)          05:29:07-347 {dump_version      :058} ----------------------------------------------------
------
vsys dev open 4
cmpi             05:29:07-348 {sys_runtime_librar:180} Failed to open /dev/mpi/vdec,please check insmod
success or fail
cmpi              05:29:07-348 {sys_runtime_librar:180} Failed to open /dev/mpi/vo,please check insmod
success or fail
load library(librga.so) in releative path
(null)          05:29:07-355 {monitor_log_level :131} #Start monitor_log_level thread, arg:(nil)
cmpi            05:29:07-356 {rk_mpi_ao_test_ent:717} start running loop count  = 0
mb               05:29:07-389 {mb_create_buffer  :485} allocated buffer(this=0xa6f3da30, data=0xa54d3000,
size=16384, phy=(nil), id = 1047)
mb               05:29:07-389 {mb_create_buffer  :485} allocated buffer(this=0xa6f3da90, data=0xa54cb000,
size=16384, phy=(nil), id = 1048)
mb               05:29:07-390 {mb_create_buffer  :485} allocated buffer(this=0xa6f3daf0, data=0xa54c3000,
size=16384, phy=(nil), id = 1049)
mb               05:29:07-391 {mb_create_buffer  :485} allocated buffer(this=0xa6f3db50, data=0xa54bb000,
size=16384, phy=(nil), id = 1050)
cmpi            05:29:07-391 {test_init_mpi_ao  :235} Set volume curve type: 0
cmpi            05:29:07-392 {commandThread     :397} test info : mute = 0, volume = 100
cmpi            05:29:07-393 {sendDataThread    :318} params->s32ChnIndex : 0
cmpi            05:29:13-229 {sendDataThread    :371} eof
cmpi            05:29:13-349 {rk_mpi_ao_test_ent:722} end running loop count  = 0
vsys_release, 4
vi_release 7
```

Play the user's audio file.

```
# rk_mpi_ao_test -i /oem/usr/share/speaker_test.wav --sound_card_name=hw:0,0 --
device_ch=2 --device_rate=8000 --input_rate=8000 --input_ch=2 --set_volume 80
```

6.11 Camera

Step 1: Connect the speaker and the Speaker interface.

Step 2: Since RKIPC occupies the driver devices of the camera0 and camera1, it is necessary to kill the RKIPC process first before customizing the use of the camera0 and camera1.

```
# killall rkipc
```

Step 3: Execute the following commands to view the device channel.

```
# grep "" /sys/class/video4linux/v*/name | grep mainpath
# grep "" /sys/class/video4linux/v*/name | grep selfpath
```

```
# grep "" /sys/class/video4linux/v*/name | grep mainpath
/sys/class/video4linux/video14/name:rkisp_mainpath
/sys/class/video4linux/video22/name:rkisp_mainpath
# grep "" /sys/class/video4linux/v*/name | grep selfpath
/sys/class/video4linux/video15/name:rkisp_selfpath
/sys/class/video4linux/video23/name:rkisp_selfpath
```

Step 4: Execute the following commands to start 3A service.

```
# rkaiq_3A_server &
```

Step 5: Camera0 video recording.

(1) This command captures a 360-frame video stream from the /dev/video14 device using the v4l2-ctl tool, saves it as a video50.yuv file, and sets the video format to 640×480 resolution and the pixel format to NV12.

```
# v4l2-ctl --device=/dev/video14 --set-fmt-video=width=640,height=480,pixelformat=
NV12 --stream-mmap --stream-to=video50.yuv --stream-count=360
```

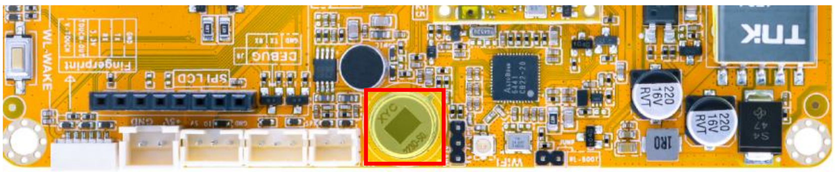
```
# v4l2-ctl --device=/dev/video14 --set-fmt-video=width=640,height=480,pixelforma
t=NV12 --stream-mmap --stream-to=video50.yuv --stream-count=360
DBG: /dev/media2: wait stream start event success ...
DBG: device manager start
CAMHW:K:cid[0] AiqCamHw_start success. isGroup:0, isOnline:1, isMultiIsp:0, init_ens:0x71fec47fc472b
XCORE:K:cid[0] rk_aiq_uapi2_sysctl_start success.
DBG: rkisp_init engine succeed
DBG: /dev/media2: wait stream stop event...
CAMHW:K:blc get bay3dbuf: ds_fd is 45 ds_size is 327680 ds_address is 0xa5beb000
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 30.85 fps
<<<<<<<<<<<<<<<<<<<DBG: /dev/media2: wait stream stop event success ...
CAMHW:K:munmap success
XCORE:K:cid[0] rk_aiq_uapi2_sysctl_stop success.
DBG: /dev/media2: wait stream start event...
```

Step 6: Upload the recorded video to the Ubuntu virtual machine via the adb pull command, and then play it using ffplay.

```
$ ffplay -f rawvideo -pixel_format nv12 -video_size 640x480 video50.yuv
```

Change /dev/video14 to /dev/video15 and use camera1 for recording.

6.12 PIR



PIR

PIR is actually directly controlled by ATBM6441. AT commands can be sent to ATBM6441 to use PIR.

Step 1: ATBM6441 is required to return information to EM1103B; otherwise, the status information of GPIO10 will only be viewed on the serial port of ATBM6441.

```
# atbm_iot_cli fw_print 1
```

```
# atbm_iot_cli fw_print 1
cmd_line: fw_print 1
[ 3618.682408] [atbm_ioctl_conn_err_async]:(229) called
# OK
[ 3618.682408]
[ 3690.888886] [atbm_log]:atbm_sdio_tx_bh:cmd free(28),used(1)
```

Step 2: Execute the following command to enable the PIR function.

```
# atbm_iot_cli fw_cmd "AT+PIR_READ 1"
```

```
# atbm_iot_cli fw_cmd "AT+PIR_READ 1"
cmd_line: fw_cmd AT+PIR_READ 1
[ 4359.705237] [atbm_log]:atbm_sdio_tx_bh:cmd free(27),used(1)
OK
# [ 4362.706247] Start read pir
[ 4643.100819] --------------PIR: Somebody is near! gpio_value= 1
[ 4666.112466] --------------PIR: Nobody is near! gpio_value= 0
```

The PIR sensor will detect the movement of a living body within a certain range. If there is movement of a living body, it will emit "Somebody is near!" If there is no live movement within a certain period of time, send out "Nobody is near!".

Step 3: Execute the following command to disable the PIR function.

```
# atbm_iot_cli fw_cmd "AT+PIR_READ 0"
```