

广东龙芯2K0300先锋派快速使用指南

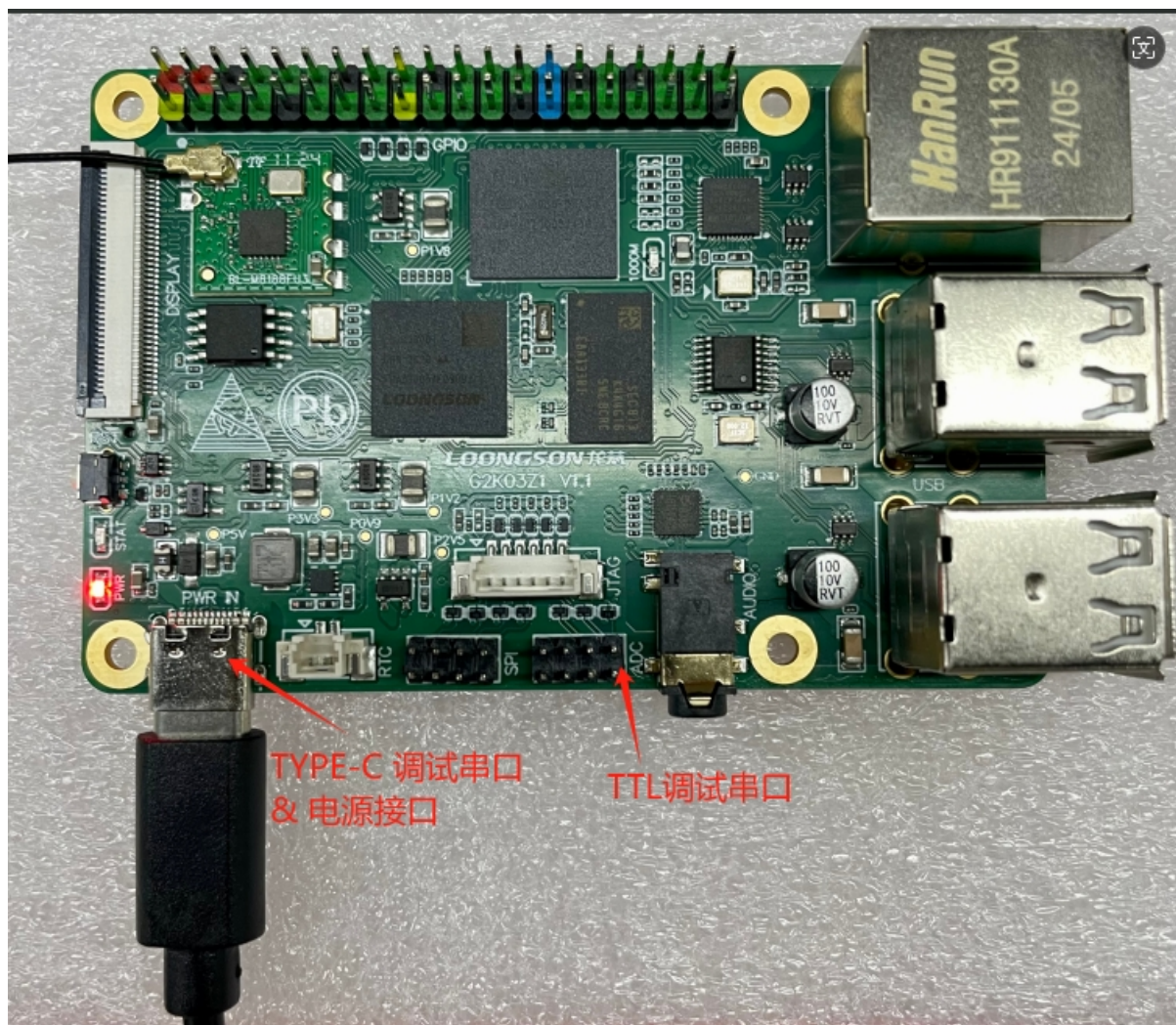
一、开发板使用

1.1 先显示出来

1.1.1 使用串口调试

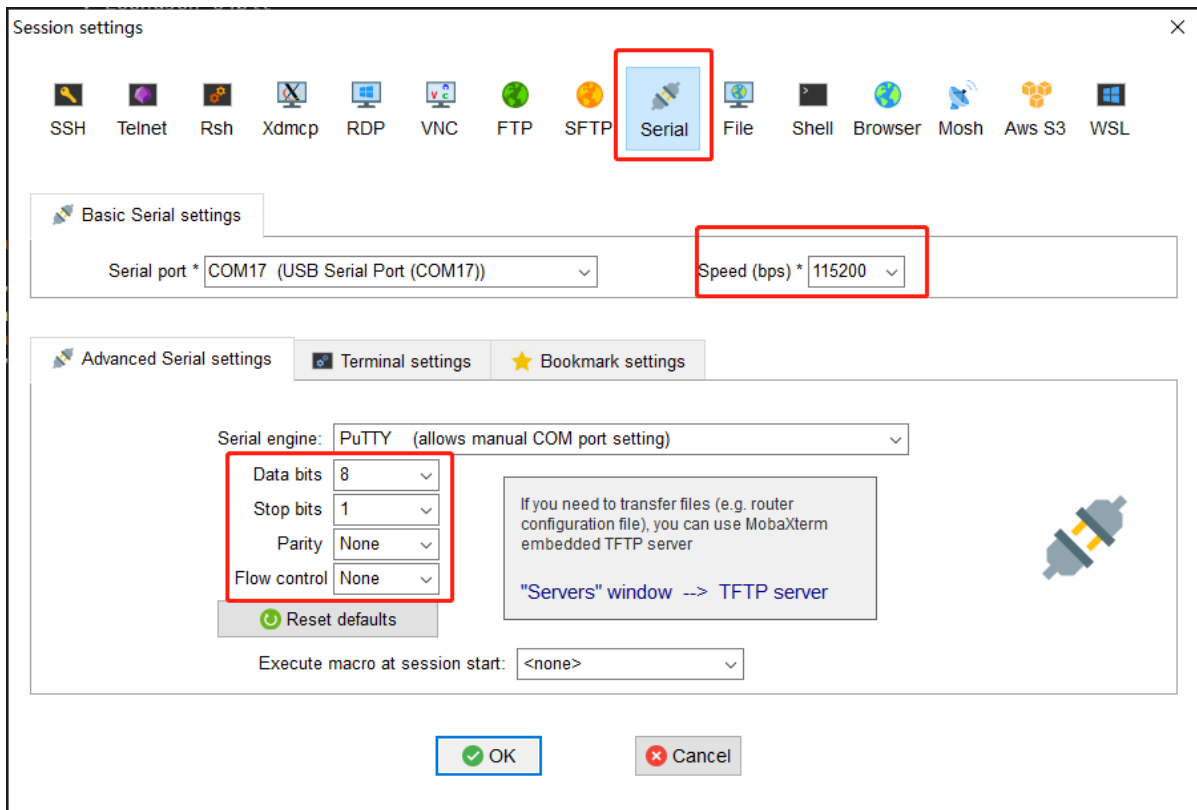
1. **UART0** 做为 Debug 串口，配套 TYPE-C 线一端接板卡，另一端接PC。**TYPE-C**线同时可以做为电源供电，如果板卡上的负载较大时，建议**TYPE-C**的**USB-A**接口接在5v电源适配器上，然后使用网络或TTL调试串口进行调试。

Debug 串口接线如下图：



2. PC上打开串口工具（比如 MobaXterm, minicom 等），配置好串口（**选择好串口号，设置波特率：115200，数据位：8，停止位：1，硬件流控：无**），等待上电。

MobaXterm的串口设置图



3.按下开发板上POWER键启动，并进入预置Busybox系统，默认自动登录；系统默认账户为 `root`，默认密码为 `123`，默认IP为 `192.168.1.10`。

```
Welcome to Loongson-gd
LS-GD login: root (automatic login)

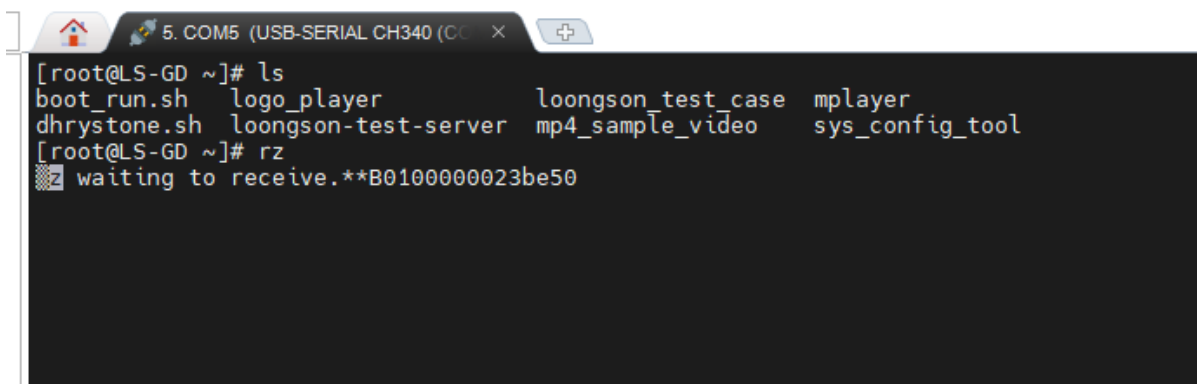
[root@LS-GD ~]#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB0
```

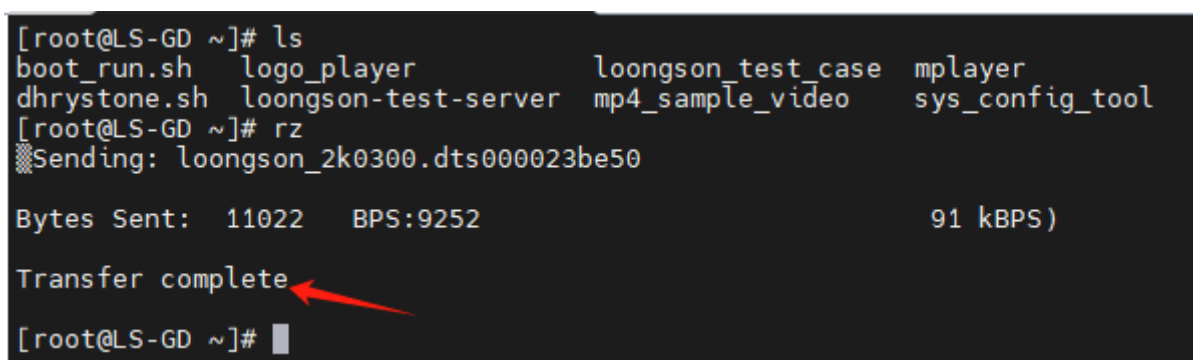
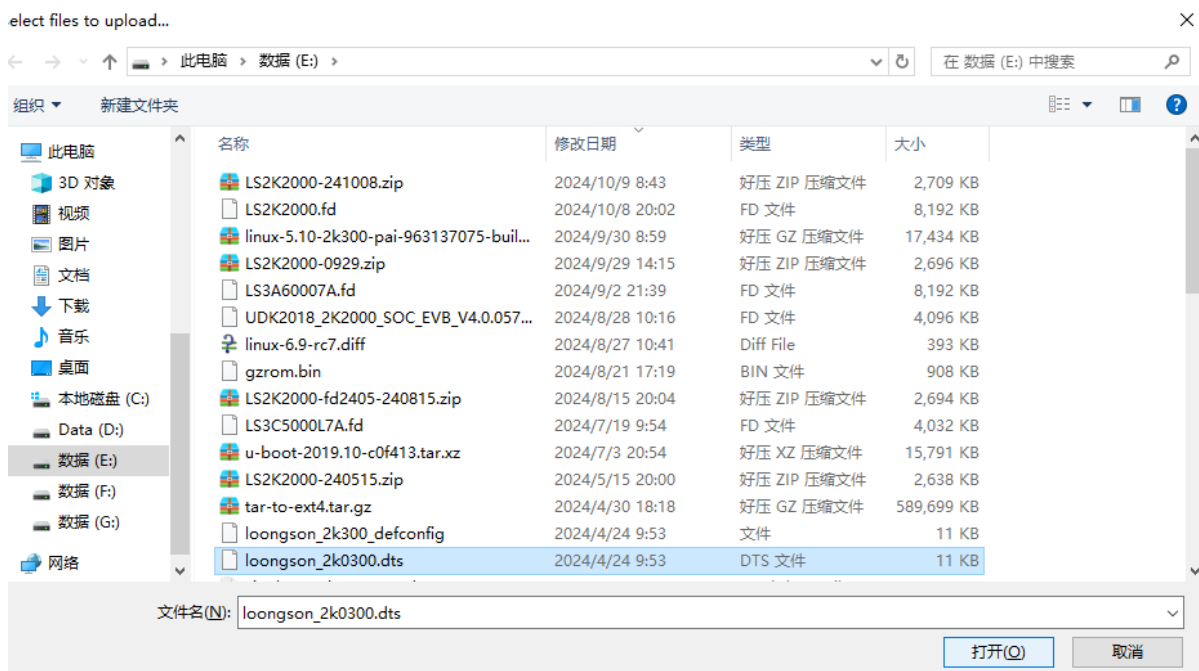
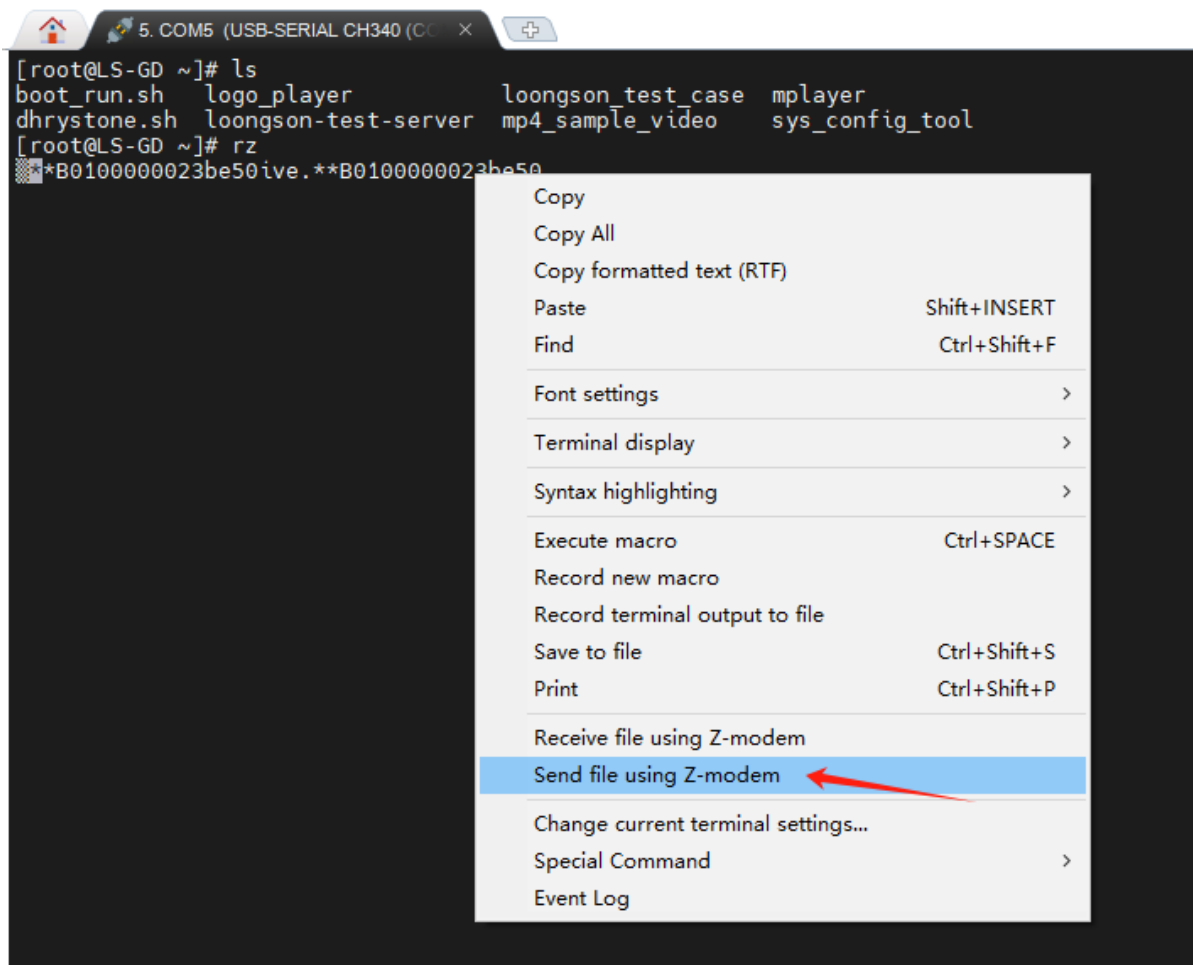
1.1.1.1 通过串口传输文件

串口支持使用XMODE, YMODE, ZMODE协议进行文件传输，这里使用ZMODE协议进行文件传输。

将文件传到板卡的步骤如下：

- 1.在串口终端输入 `rz` 命令
- 2.右键在弹出的菜单中选择"Send file using Z-modem"
- 3.在弹出的对话框中选择要下发的文件
- 4.待串口终端显示“Transfer complete”，表示文件传输结束。按回车键，即可回到命令行。





```
5. COM5 (USB-SERIAL CH340 (CC X +
[root@LS-GD ~]# ls
boot_run.sh      logo_player      loongson_test_case  mplayer
dhrystone.sh     loongson-test-server  mp4_sample_video    sys_config_tool
[root@LS-GD ~]# rz
Sending: loongson_2k0300.dts000023be50

Bytes Sent:  11022   BPS:9252                91 kbps)

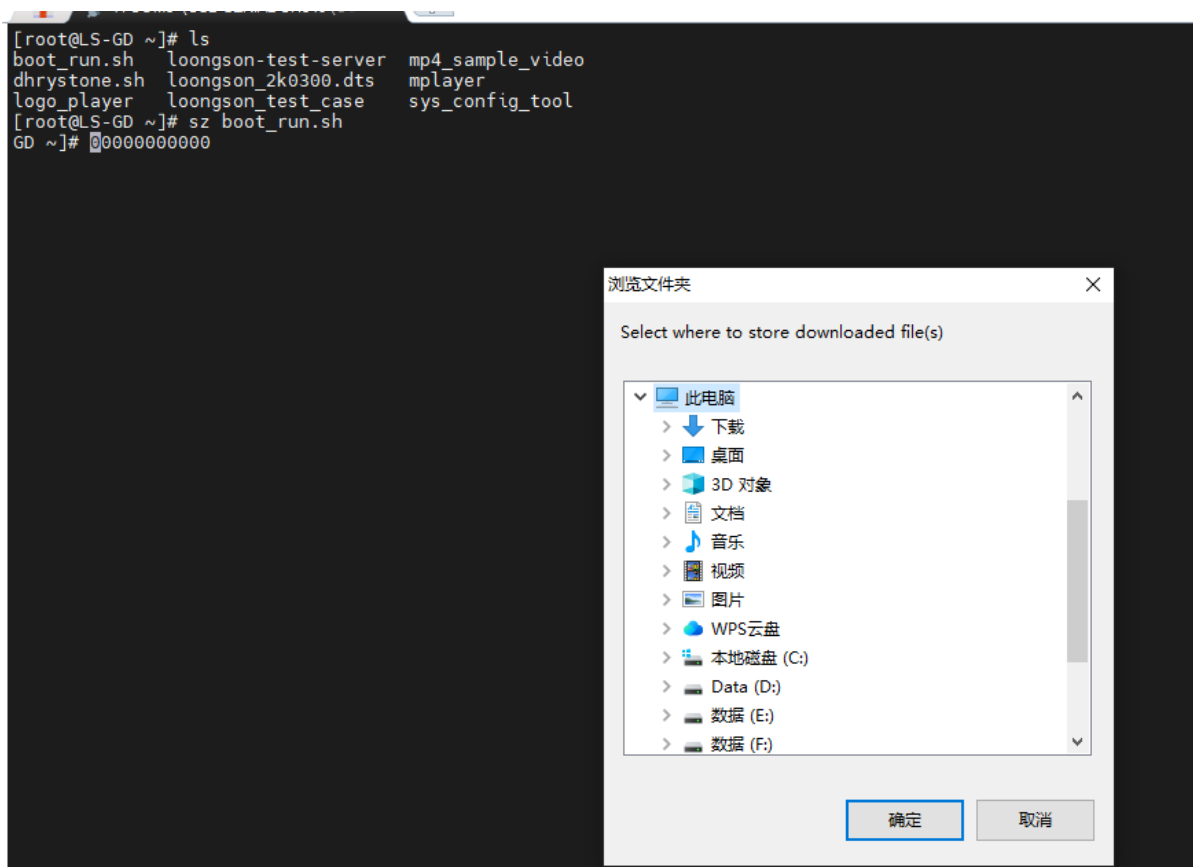
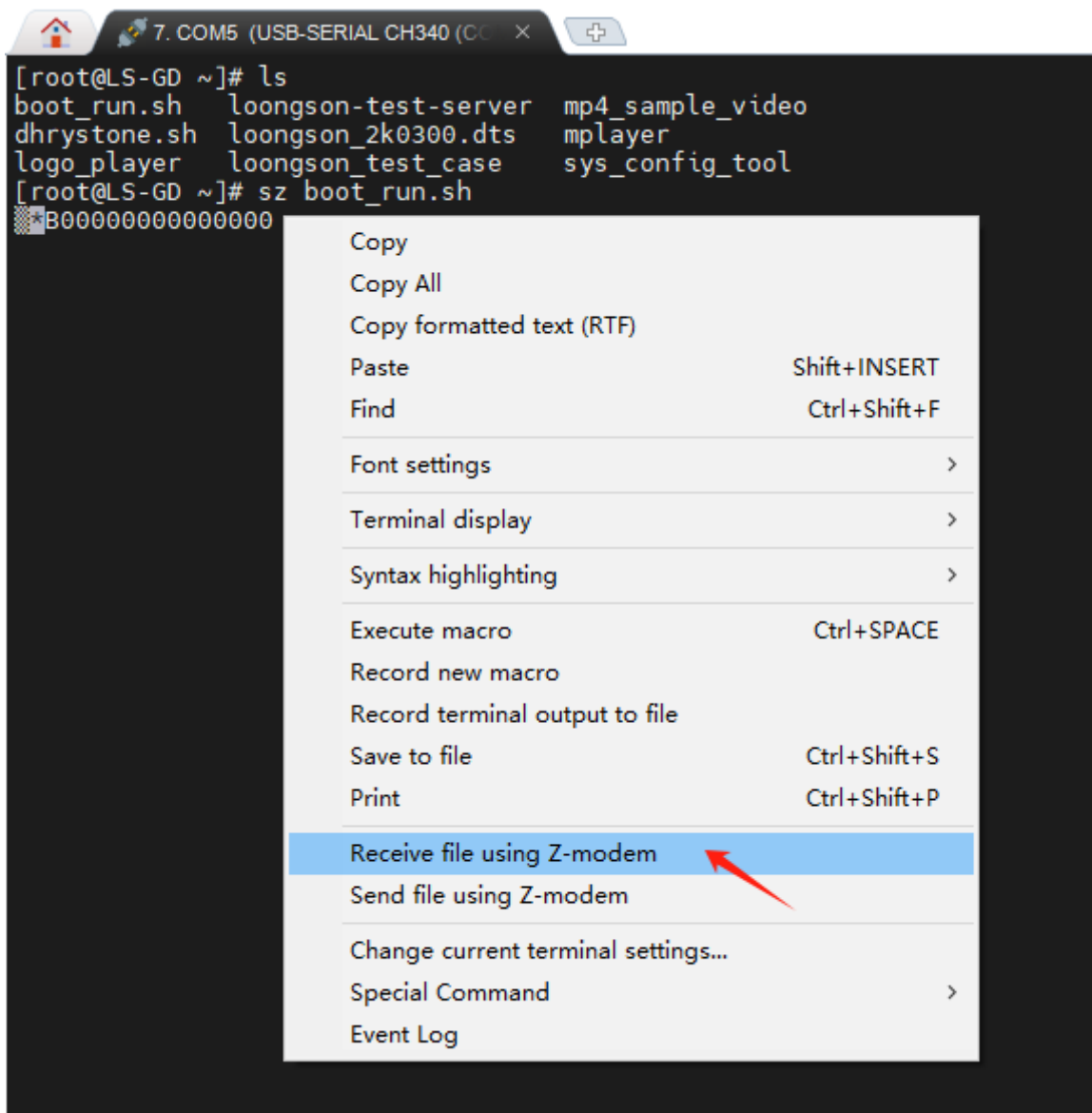
Transfer complete

[root@LS-GD ~]# ls
boot_run.sh      loongson-test-server  mp4_sample_video
dhrystone.sh     loongson_2k0300.dts  mplayer
logo_player      loongson_test_case   sys_config_tool
[root@LS-GD ~]#
```

将文件从板卡传到PC机的步骤如下:

- 1.在串口终端输入 `sz xxx` 命令, xxx 为要传输的文件。
- 2.右键在弹出的菜单中选择"Recevice file using Z-modem".
- 3.在弹出的对话框中选择文件存放到哪到个位置。
- 4.待串口终端显示"Transfer complete", 表示文件传输结束。按回车键, 即可回到命令行。

```
7. COM5 (USB-SERIAL CH340 (CC X +
[root@LS-GD ~]# ls
boot_run.sh      loongson-test-server  mp4_sample_video
dhrystone.sh     loongson_2k0300.dts  mplayer
logo_player      loongson_test_case   sys_config_tool
[root@LS-GD ~]# sz boot_run.sh
*B000000000000000
```



```
[root@LS-GD ~]# sz boot_run.sh
lrz waiting to receive.

Receiving: boot_run.sh
Downloading: 100% kBytes received: 0/ 0 kBPS:4

Transfer complete

[root@LS-GD ~]#
```

1.1.2. 使用网络调试

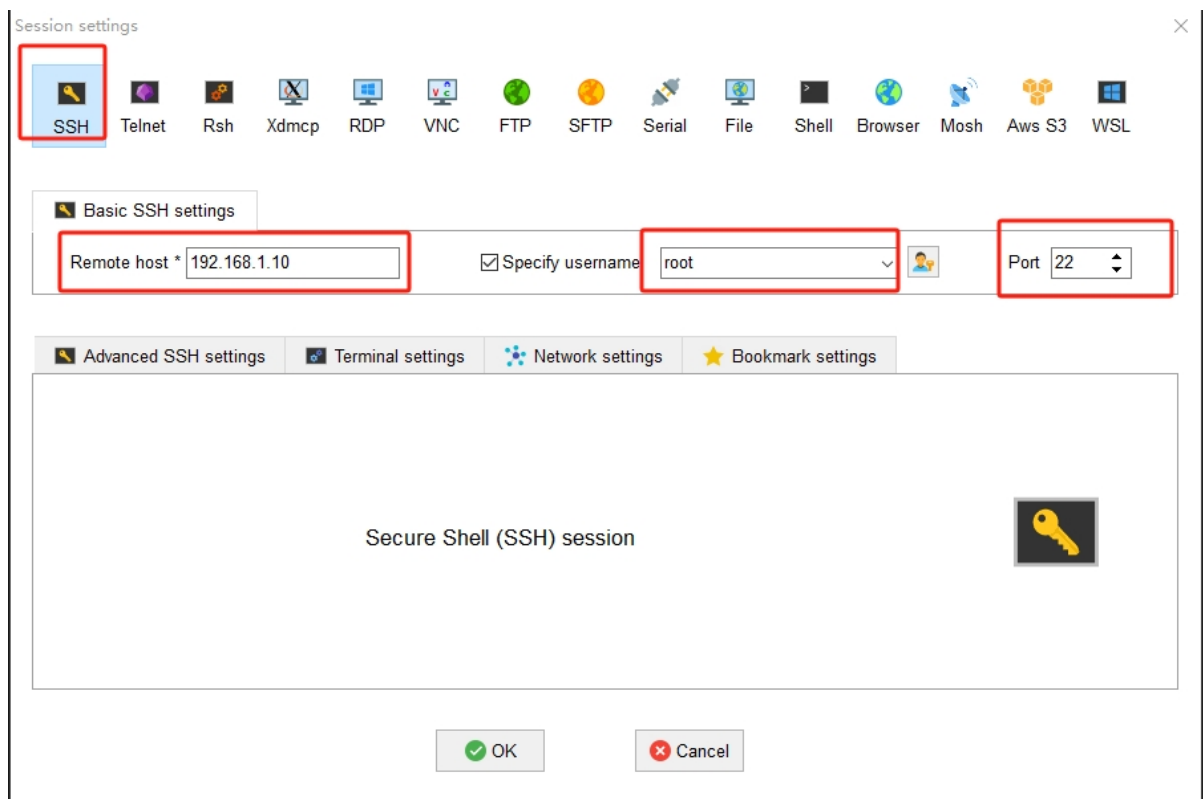
2K0300先锋派集成了有线网口和WIFI 模块。

1.1.2.1 使用有线网络

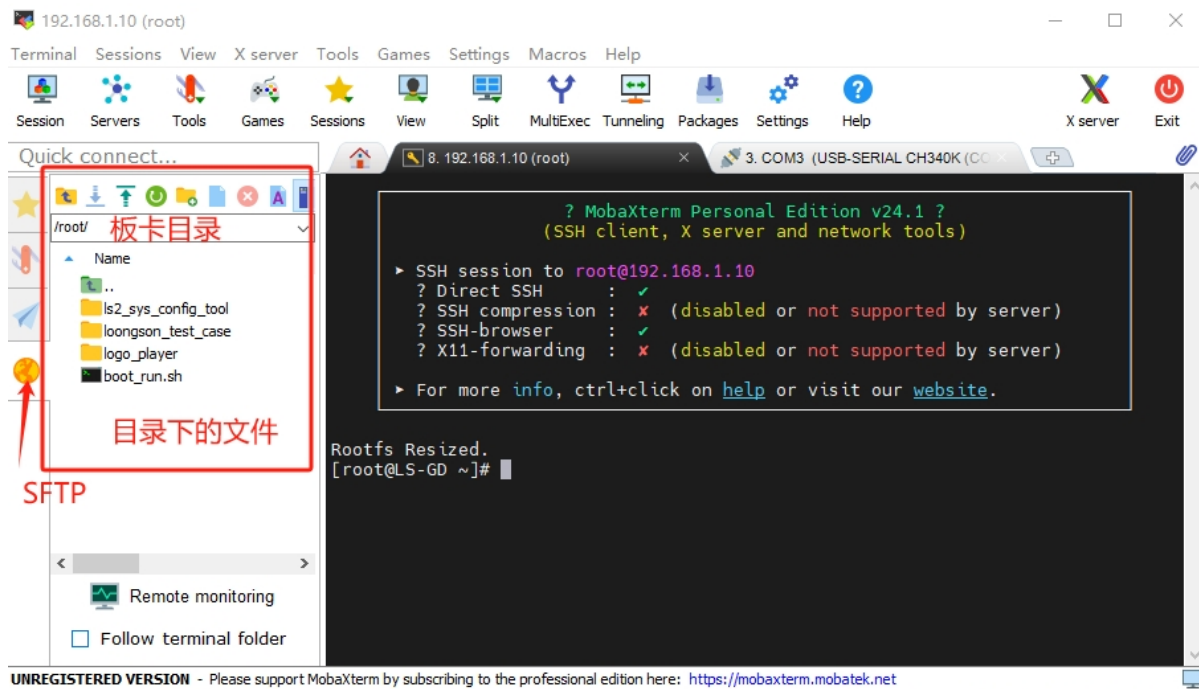
网盘里面的文件系统均带有ssh服务，可以通过ssh 登录到板卡上进行调试。将板卡与电脑通过网线直连，同时配置电脑的IP 为 192.168.1.2，也可以是同一网段的其他IP，建议配成 192.168.1.2，因为 u-boot 中默认配的serverip 是 192.168.1.2。系统默认账户为 root，默认密码为 123，默认IP为 192.168.1.10，SSH默认端口为：22。

如出现板卡Ping 不通电脑，但电脑可以Ping 通板卡的现象，请检查电脑的防火墙状态，将其关闭后再试。

MobaXterm的SSH设置图：



ssh 连接的后图如下：



ssh 连接后可以通过SFTP 传输文件，支持拖拽式进行文件上下传。

1.1.2.2 使用无线网络

在支持 WIFI 模块驱动、固件的情况下，ifconfig 能看到 wlan 设备：

```
# ifconfig -a
...

wlan0  Link encap:Ethernet  HWaddr D6:EF:AB:03:06:69
        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)
```

1.1.2.2.1.NetworkManager链接WIFI

A.nmcli命令行工具

1.扫描 WIFI

```
$ nmcli dev wifi
IN-USE  BSSID              SSID              MODE  CHAN  RATE
      50:88:11:AE:48:79  wifi-test        Infra  1     260
Mbit/s  84             ****  WPA2
...
```

2.连接WIFI

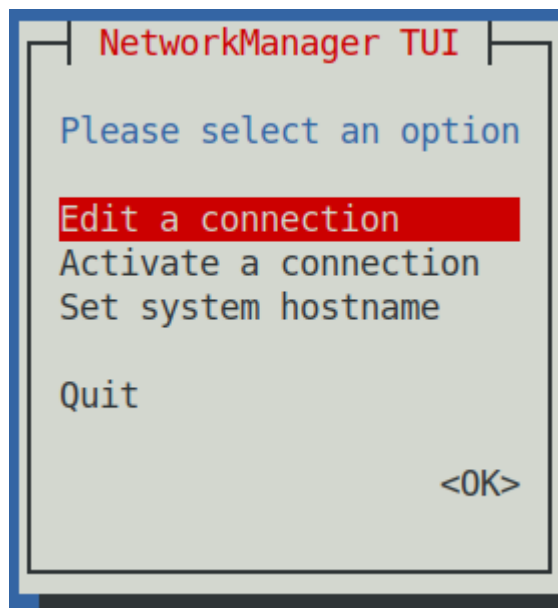
```
$ nmcli dev wifi connect "wifi-test" password "wifi-test123"
[ 1635.357063] usb 1-1: request firmware rtlwifi/rtl8188fufw.bin
[ 1635.366481] usb 1-1: request firmware rtlwifi/rtl8188fufw.bin loaded
[ 1636.074489] RTL871X: nolinked power save leave
[ 1636.988127] RTL871X: nolinked power save enter
```

```
[ 1637.075790] usb 1-1: request firmware rtlwifi/rtl8188fufw.bin
[ 1637.087725] usb 1-1: request firmware rtlwifi/rtl8188fufw.bin loaded
[ 1637.704456] RTL871X: nolinked power save leave
[ 1637.709590] RTL871X: rtw_set_802_11_connect(wlan0) fw_state=0x00000008
[ 1637.763436] RTL871X: start auth
[ 1637.771009] RTL871X: auth success, start assoc
[ 1637.782256] RTL871X: assoc success
[ 1637.795474] RTL871X: recv eapol packet
[ 1637.806561] RTL871X: send eapol packet
[ 1637.817420] RTL871X: recv eapol packet
[ 1637.834301] RTL871X: send eapol packet
[ 1637.855694] RTL871X: set pairwise key camid:4, addr:50:88:11:ae:48:79, kid:0,
type:AES
[ 1637.870885] IPV6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 1637.897575] RTL871X: set group key camid:5, addr:50:88:11:ae:48:79, kid:1,
type:AES
Device 'wlan0' successfully activated with '1d8f7241-1fb0-43c8-bfb6-
875bce6561a3'.
```

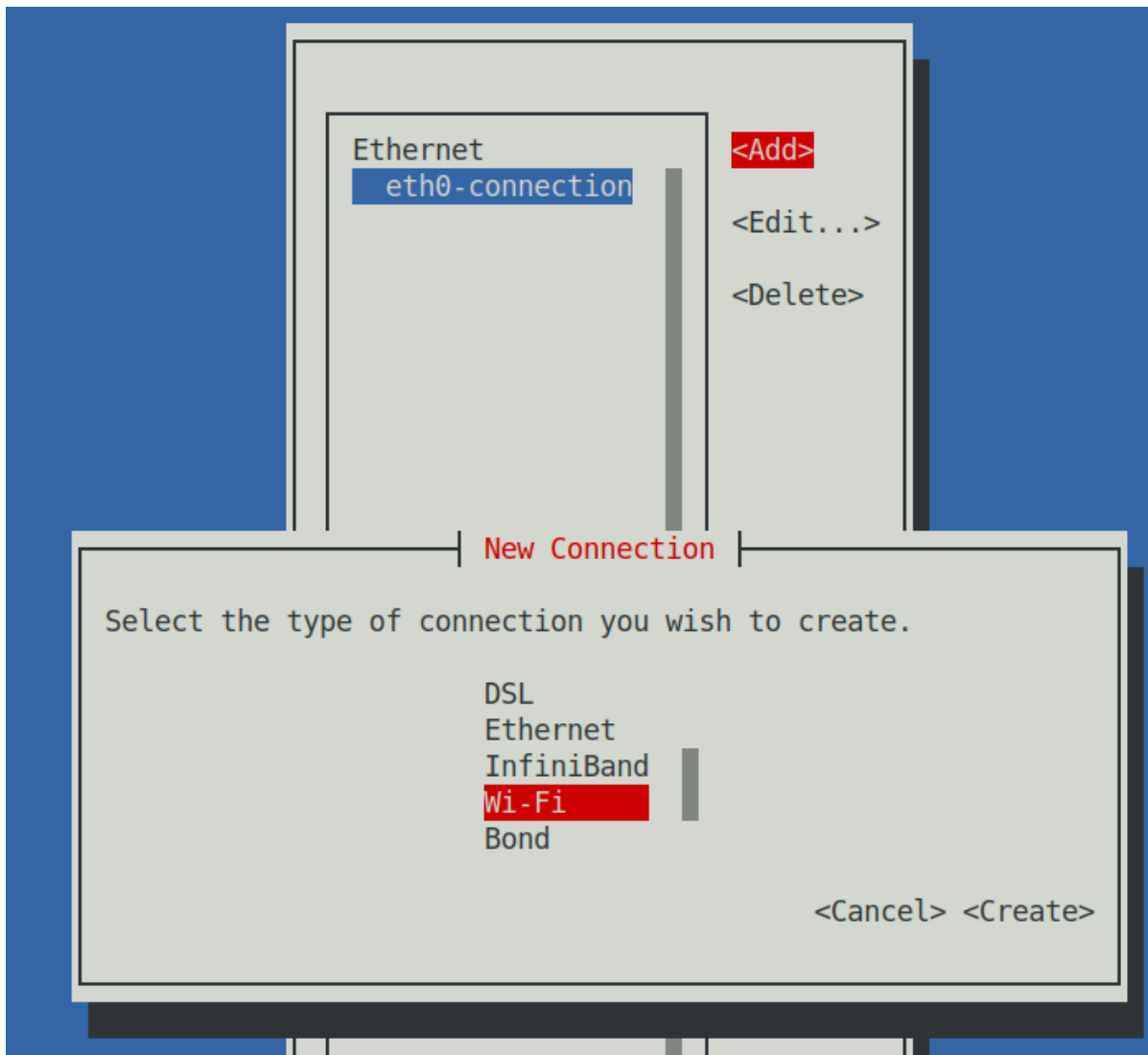
B.nmtui图形化工具

建议在 ssh 终端调用 nmtui

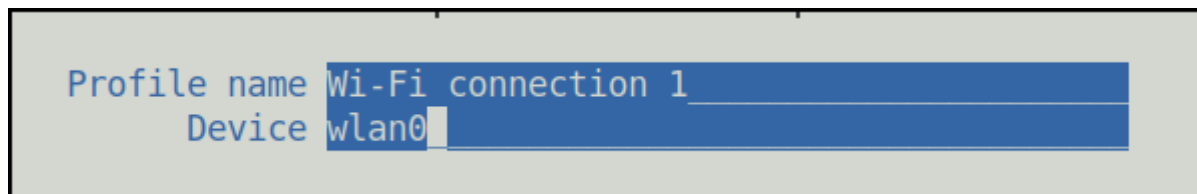
1.选择"Edit a connection".



2.选择右栏添加"Add", 并选中"Wi-Fi".



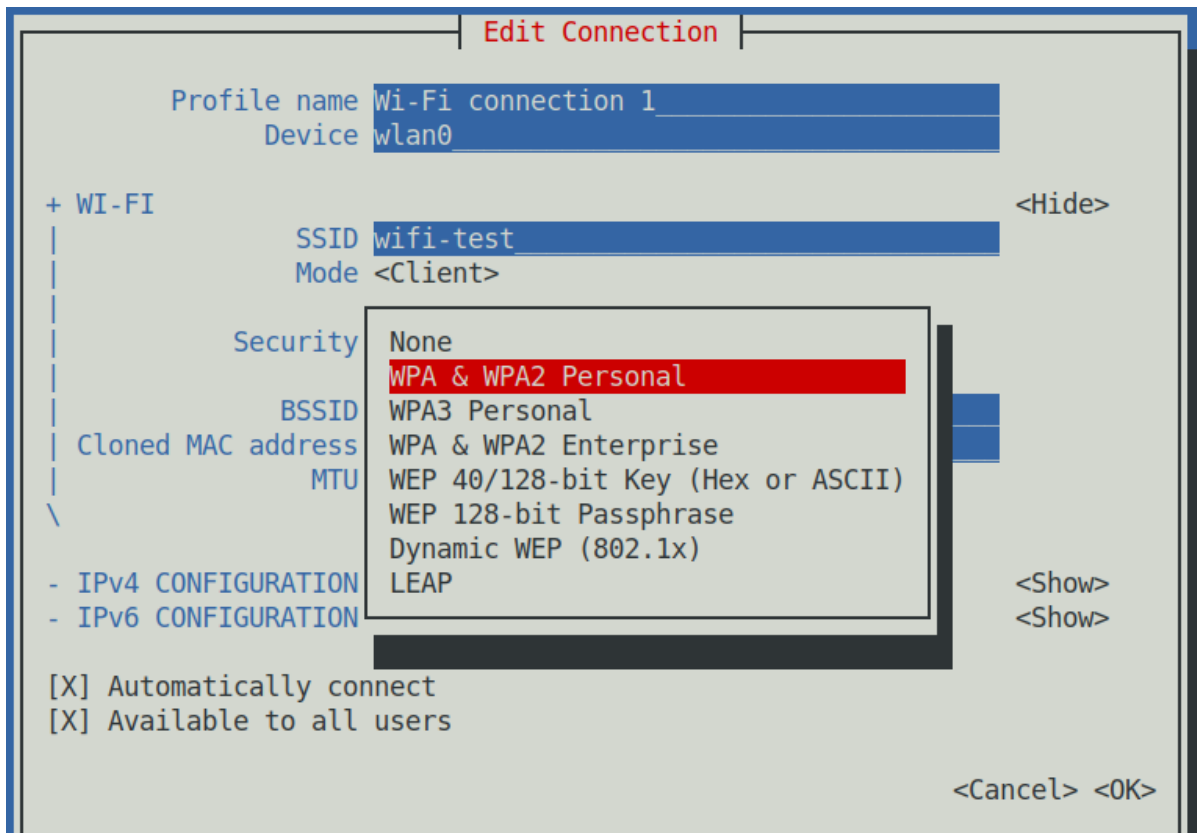
3.填写"Device"为无线网卡名称"wlan0".



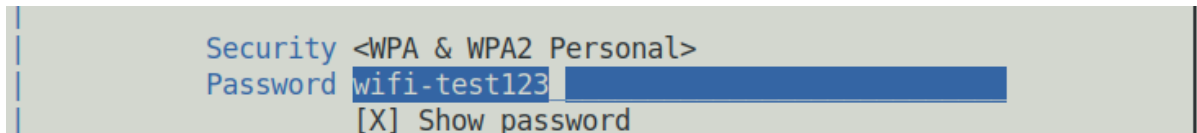
4.填写"SSID"为要接入的无线网络"wifi-test".



5.选中"Security", 改为无线网络所使用的加密方式, 通常为 "WPA & WPA2 Personal".



6.填写Wifi密码"Password".



1.1.2.2.2.iw与wpa工具配置WIFI

注意：NetworkManager会抢占WIFI控制，因此建议在没有NetworkManager运行的系统下使用wpa工具

1.iw工具扫描 WIFI SSID

```
$ iw dev wlan0 scan | grep SSID
    SSID: wifi-test
    ...
```

2.wpa_passphrase设置 WIFI密码

```
$ wpa_passphrase "wifi-test" "wifi-test123" > /etc/wpa_supplicant.config
```

3.wpa_supplicant连接 WIFI

```
$ wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.config
Successfully initialized wpa_supplicant
```

4.dhccpd 协商 IP

```
$ dhccpd wlan0
DUID 00:01:00:01:27:ac:2c:96:00:0e:02:00:21:db
wlan0: IAID 02:00:21:db
wlan0: soliciting a DHCP lease
```

```
wlan0: soliciting an IPv6 router
wlan0: offered 192.168.10.252 from 192.168.10.1
wlan0: probing address 192.168.10.252/24
wlan0: leased 192.168.10.252 for 43200 seconds
wlan0: adding route to 192.168.10.0/24
wlan0: adding default route via 192.168.10.1
forked to background, child pid 240

$ ifconfig wlan0
wlan0      Link encap:Ethernet  HWaddr 00:0E:02:00:21:DB
           inet addr:192.168.10.252  Bcast:192.168.10.255  Mask:255.255.255.0
           inet6 addr: fe80::20e:2ff:fe00:21db/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:112 errors:0 dropped:0 overruns:0 frame:0
           TX packets:78 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:14951 (14.6 KiB)  TX bytes:9470 (9.2 KiB)
```

配置好无线网络之后，ssh的设置方法同[使用有线网络](#)一致。

1.1.2.2.3.WIFI使用问题答疑

1.1.2.2.3.1.使用NetworkManager无法连接WIFI

排查思路如下：

1.先关闭 NetworkManager

```
$ systemctl stop NetworkManager
```

2.使用 wpa 工具连接 WIFI

```
$ wpa_passphrase "wifi-test" "wifi-test123" > /etc/wpa_supplicant.config
$ wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.config
$ dhcpcd wlan0
```

3.如果 wpa 工具连接 WIFI 成功，那么有可能是 WIFI 芯片不支持 Random Mac 导致的，我们就要禁用 NetworkManager 中的 Random Mac，重启后再使用 NetworkManager 连接

```
$ echo -e "[device]\nwifi.scan-rand-mac-address=no" | tee
/etc/NetworkManager/conf.d/disable-random-mac.conf
$ reboot
```

如果以上思路无法解决问题请反馈。

1.1.2.2.3.2.使用传统的 iw 工具无法连接 WIFI

iw 工具只支持 WEP 加密，现在常见的 WIFI 都是 WPA 加密，建议使用 wpa 工具或 NetworkManager 连接。

1.1.3. 使用LCD

我们适配了多种LCD屏的分辨率(800x480, 1024x600, 1280x800), 覆盖了4.3', 7', 10.1'等主流尺寸的LCD屏。默认分辨率为1024x600, 分辨率切换参考 [1.3切换LCD分辨率](#)。

LCD 接口接线如下图:



1.2.首次进入 uboot

蜂鸟板的文件系统是典型的linux系统，一些操作事项参考 [五、文件系统](#)。本章之后的内容我们将着重说明在 uboot 中的操作办法，让我们先进入 uboot 菜单：

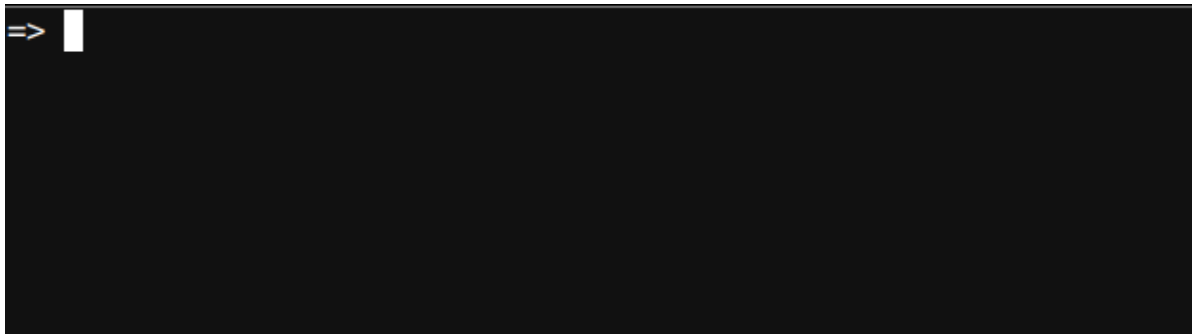
开机按住键盘上的字母“m”键进入u-boot菜单

```
*** U-Boot Boot Menu ***

[1] System boot select
[2] Update kernel
[3] Update rootfs
[4] Update u-boot
[5] Update ALL
[6] System install or recover
[7] Board product
[8] Video resolution select
[9] Video rotation select
[a] U-Boot console

Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

在u-boot 菜单中选择 “[a] U-Boot console”进入u-boot 命令终端
或者开机按住键盘上的字母“c”进入 u-boot console



在u-boot 命令终端输入 `bootmenu` 命令也可以进入u-boot 菜单

```
=> bootmenu
*** U-Boot Boot Menu ***

[1] System boot select
[2] Update kernel
[3] Update rootfs
[4] Update u-boot
[5] Update ALL
[6] System install or recover
[7] Board product
[8] Video resolution select
[9] Video rotation select
[a] U-Boot console

Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

在命令终端中键入 `boot` , 即可进入正常启动流程

```
=> boot
--if mmc0 --fmt ext4 --sym /boot/uImage
--of ram --fmt --sym
--extra 0x0
loading...
loaded&burning 8812660 bytes ...
load&burn 8812660 finished
## Booting kernel from Legacy Image at 9000000003000000 ...
Image Name:   Linux-5.10.0.lsgd-ga653fe8758c4
Image Type:   LoongArch Linux Kernel Image (gzip compressed)
Data Size:    8812596 Bytes = 8.4 MiB
Load Address: 00200000
Entry Point:  00e20250
Verifying Checksum ... OK
Uncompressing Kernel Image
warning: invalid device tree. Used linux default dtb
512 MiB
[ 0.000000] Linux version 5.10.0.lsgd- ...
```

在命令终端中键入 `reboot` / `reset` , 即可重启

```

=> reboot
LoongArch Initializing ...

RAM(Cache AS RAM) Initializing ...

Lock Scache Done.

Copy spl code to locked scache...

Jump to board_init_f...
Enter board_init_f...


-   -   -   -   -   -   -   -   -   -   /   -   -   \
|   |   |   |   | \ |   |   |   |   |   |   |   |   |   |
|___|___|___|___| \ |___|___|___|___|___|___|___|___|___|

=====ddr4 init and training done!=====
Trying to boot from BootSpace
U-boot start ...
Jump to board_init_f...
U-Boot 2022.04-v2.1.0-00502-gcadd162f (Jul 25 2024 - 11:32:31 +0800)
CPU: LA264
Speed: Cpu @ 997 MHz/ Mem @ 800 MHz/ Bus @ 200 MHz
Model: loongson-2k300

```

下面我们介绍几个比较简单的菜单项

1.2.1.启动模式选择

蜂鸟板支持从多种介质启动系统，前提是该介质安装有系统，默认从EMMC启动。可以按以下流程更改启动介质。

1.uboot菜单选择“[1] System boot select”

```

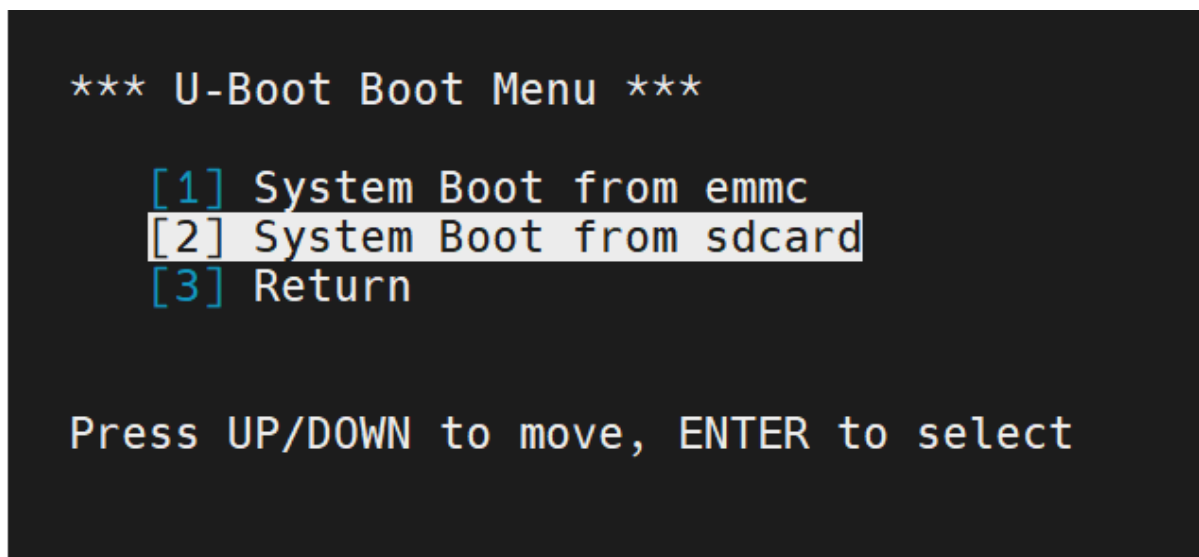
*** U-Boot Boot Menu ***

[1] System boot select
[2] Update kernel
[3] Update rootfs
[4] Update u-boot
[5] Update ALL
[6] System install or recover
[7] Board product
[8] Video resolution select
[9] Video rotation select
[a] U-Boot console

Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit

```

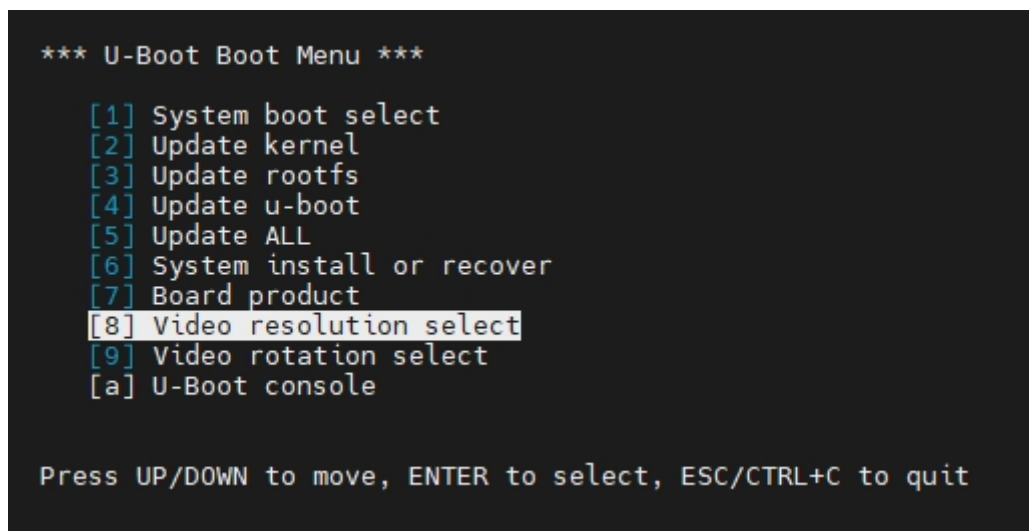
2.选择“[1] System Boot from emmc”或“[2] System Boot from sdcard”



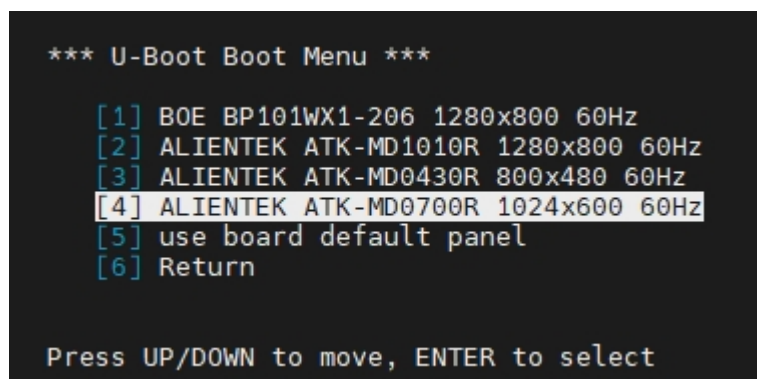
1.2.2.切换LCD分辨率

如果切换之后，显示花屏，可检查是否使用了新的u-boot 和 kernel，以及板卡供电是否满足。

1.uboot菜单选择"[8] Video resolution select"



2.选择相应的分辨率



1.3.uboot下常用外设与网络服务

1.3.1.U盘-FAT32

考虑到linux与Windows的兼容性，我们选择将U盘格式设为 FAT32。

1.3.1.1.U盘准备

1.U盘格式化为FAT32，并在U盘目录下创建update文件夹

U盘格式化可参考 [第八章第二节 如何格式化U盘](#)，从而避免诸如分区表之类的问题

```
loongson@loongson-PC:~$ sudo parted /dev/sdb1
请输入密码：
验证成功
GNU Parted 3.2
Using /dev/sdb1
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Model: Unknown (unknown)
Disk /dev/sdb1: 31.0GB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:

Number  Start  End      Size    File system  Flags
  1      0.00B  31.0GB  31.0GB  fat32
(parted) quit
loongson@loongson-PC:~$ df
文件系统          1K-块      已用      可用  已用% 挂载点
udev              8216464          0  8216464    0% /dev
tmpfs             1653440      5472   1647968    1% /run
/dev/sda3         15416304  9810420  4803068   68% /
tmpfs             8267184   203152   8064032    3% /dev/shm
tmpfs              5120         32     5088    1% /run/lock
tmpfs             8267184          0   8267184    0% /sys/fs/cgroup
/dev/sda2         1515376   123176   1297176    9% /boot
/dev/sda6         14384176  7594060   6039732   56% /recovery
/dev/sda5         181306564 72480528 99546500   43% /data
tmpfs             1653424        144   1653280    1% /run/user/1000
/dev/sdb1         30252592 7722928 22529664   26% /media/loongson/vfat
loongson@loongson-PC:~$ ls -ld /media/loongson/vfat/update
/media/loongson/vfat/update
loongson@loongson-PC:~$
```

2.在U盘update目录下放入要更新的内核（文件名为：uImage）、固件（文件名为：u-boot-with-spl.bin 或 u-boot.bin）、文件系统(rootfs.img)。

```
H:\update
λ ls
rootfs.img  u-boot-with-spl.bin  uImage

H:\update
λ |
```

3.将U盘插在开发板上

1.3.1.2.uboot下尝试读写U盘

1.重新扫描USB设备

```
=> usb reset
resetting USB...
Bus ehci@0x16080000: USB EHCI 1.00
Bus ohci@0x16088000: USB OHCI 1.0
Bus otg@0x16040000: dwc2_usb otg@0x16040000: Core Release: 2.93a
USB DWC2
scanning bus ehci@0x16080000 for devices... 1 USB Device(s) found
scanning bus ohci@0x16088000 for devices... 1 USB Device(s) found
scanning bus otg@0x16040000 for devices... 2 USB Device(s) found
    scanning usb for storage devices... 1 Storage Device(s) found
```

2.我们已经在U盘上放置了一些文件（u-boot-with-spl.bin、ulmage、rootfs.img），尝试列出文件：

```
=> fatls usb 0:1 /update
    ./
    ../
52168418  rootfs.img
8830630   uImage
906162    u-boot-with-spl.bin

3 file(s), 2 dir(s)
```

3.尝试读取一个文件到内存：

```
=> fatload usb 0:1 ${loadaddr} /update/uImage
8830630 bytes read in 206 ms (40.9 MiB/s)
=> printenv loadaddr
loadaddr=0x9000000003000000
```

4.这样 U 盘中的文件就被写入了 loadaddr 这一块地址

若出现U盘无法读写等情况，建议检查U盘分区状态，建议用MBR分区表、单分区格式

1.3.2.tftp 服务

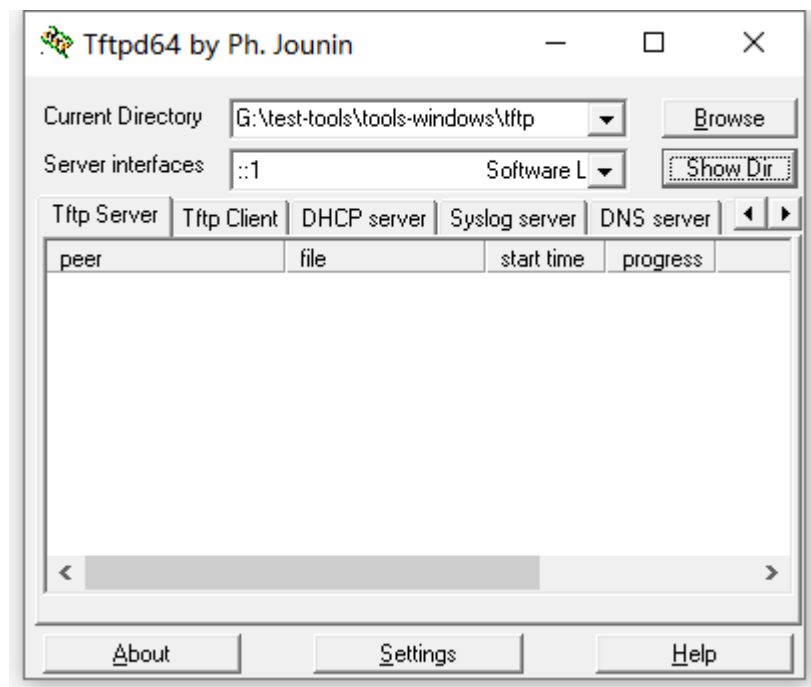
1.3.2.1.安装tftp服务

a.windows平台tftp服务器

windows平台使用tftpd.exe(32位或64位版本都可以，此处用的是64位的版本)，**要更新的文件(内核（文件名为：ulmage）、固件（文件名为：u-boot-with-spl.bin）、文件系统(rootfs.img))和tftpd.exe放在同一目录下**，然后打开tftp.exe 即可。

Windows tftp 服务器可参考“5.2.2.2.tftp方式安装所需条件”，有详细步骤说明解决防火墙问题与Windows IP问题

 rootfs.img	2024/05/15 15:57
 tftpd64.exe	2019/02/27 22:07
 u-boot-with-spl.bin	2024/06/03 14:42
 ulmage	2024/05/10 9:29



b. linux平台TFTP服务器

1. linux平台TFTP服务器（示例为tftpd-hpa服务），此处设置的tftp文件根目录为 `/home/loongson/tftpboot/`，要更新的文件(内核（文件名为：ulmage）、固件（文件名为：u-boot-with-spl.bin 或 u-boot.bin）、文件系统(rootfs.img))放在此目录下即可

```
loongson@loongson-PC:~$ sudo service tftpd-hpa status
• tftpd-hpa.service - LSB: HPA's tftp server
   Loaded: loaded (/etc/init.d/tftpd-hpa; generated)
   Active: active (running) since Tue 2022-09-06 09:04:35 CST; 6h ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1645 ExecStart=/etc/init.d/tftpd-hpa start (code=exited, status=0/SUCCESS)
    Tasks: 1 (limit: 4915)
   Memory: 9.0M
    CGroup: /system.slice/tftpd-hpa.service
            └─1663 /usr/sbin/in.tftpd --listen --user loongson --address :69 -l -c -s /home/loongson/tftpboot
```

tftp 文件根目录
↓

2. 查看服务器IP地址，比如 192.168.1.2

```
loongson@loongson-PC:~$ ifconfig
enol: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.120.1.31 netmask 255.255.255.0 broadcast 10.120.1.255
    inet6 fe80::2e45:c24d:125d:e7de prefixlen 64 scopeid 0x20<link>
    ether 00:23:9e:25:92:70 txqueuelen 1000 (Ethernet)
    RX packets 173648 bytes 205104052 (195.6 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 70729 bytes 7969553 (7.6 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 46

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::9d42:4cd8:dd99:7028 prefixlen 64 scopeid 0x20<link>
    ether f8:e4:3b:1a:13:66 txqueuelen 1000 (Ethernet)
    RX packets 625827 bytes 37306871 (35.5 MiB)
    RX errors 617731 dropped 0 overruns 0 frame 617731
    TX packets 623206 bytes 940294439 (896.7 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4208 bytes 310267 (302.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4208 bytes 310267 (302.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

loongson@loongson-PC:~$
```

3. 网线连接服务器与开发板，准备更新或传输文件

1.3.2.2.uboot网络设置（可选）

1.查看“tftp服务器ip”与“本机ip”

“tftp服务器ip”与“本机ip”都存储在环境变量中，其中，变量 **serverip** 是“tftp服务器ip”，变量 **ipaddr** 是“本机ip”，可在 `uboot console` 下执行 `printenv` 查看环境变量。

```
=> printenv
...
ipaddr=192.168.1.20
...
serverip=192.168.1.2
...

Environment size: 1315/16380 bytes
=>
```

2.更改服务器ip与本机ip

```
=> setenv serverip 192.168.2.2
=>
```

3.更改本机ip

```
=> setenv ipaddr 192.168.2.20
=>
```

4.网关配置

如果服务器与设备ip处在不同网段，则需设置网关ip。网关ip对应变量 **gatewayip**。

```
=> setenv gatewayip 192.168.1.1
=>
```

5.保存更改后的配置

ip更改后，可以执行 **saveenv** 长期保存

```
=> saveenv
Saving Environment to SPIFlash... Erasing SPI flash...Writing to SPI flash...done
OK
=>
```

1.3.2.3.uboot下尝试tftp

我们已经在tftp服务器上放置了一些文件（u-boot-with-spl.bin、ulmage、rootfs.img），尝试以下操作：

```
=> tftp uImage
Speed: 1000, full duplex
Using ethernet@0x16020000 device
TFTP from server 192.168.1.2; our IP address is 192.168.1.20
Filename 'uImage'.
```

```
Load address: 0x9000000003000000
Loading: #####
...
#####
4.9 MiB/s
done
Bytes transferred = 8826725 (86af65 hex)
=>
```

注:

若出现网络下载或更新不了的情况，建议检查电脑的防火墙是否为打开状态，如果是打开状态则将其关闭之后再试。

1.4.EMMC使用方法

将安装系统到EMMC提供有两种方法，当因系统镜像过大（大于 152M）导致安装失败时可使用方法2。

1.4.1.EMMC安装系统方法1

1.将文件系统 **文件系统(rootfs.img)** 放在tftp服务器**根目录**下，或者U盘**update**目录下。板卡插上网线或U盘

2.uboot菜单选择“[3] Update rootfs”

```
*** U-Boot Boot Menu ***

[1] System boot select
[2] Update kernel
[3] Update rootfs
[4] Update u-boot
[5] Update ALL
[6] System install or recover
[7] Board product
[8] Video resolution select
[9] Video rotation select
[a] U-Boot console

Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

3.选择“[1] Update rootfs (rootfs.img) (by usb)”或“[2] Update rootfs (rootfs.img) (by tftp)”，开始升级：

```
*** U-Boot Boot Menu ***

[1] Install System (rootfs.img) (by usb)
[2] Install System (rootfs.img) (by tftp)
[3] Install System (rootfs.img) (by mmc)
[4] Return

Press UP/DOWN to move, ENTER to select
```

```
*** U-Boot Boot Menu ***

[1] Install System (rootfs.img) (by usb)
[2] Install System (rootfs.img) (by tftp)
[3] Install System (rootfs.img) (by mmc)
[4] Return

Press UP/DOWN to move, ENTER to select
```

4.升级完成，重启输入reboot或reset

[illegible]

1.4.2.EMMC安装系统方法2

1.将以下三个文件 **内核 (ulmage)**、**文件系统压缩包(rootfs.tar.gz)** 以及 **ramdisk.gz** 放在tftp服务器根目录下, 或者U盘 **install** 目录下。

注:

放在U盘 **install** 目录下, 不再是 **update** 目录。

2.uboot菜单选择"[6] System install or recover"

```
*** U-Boot Boot Menu ***
```

```
[1] System boot select
[2] Update kernel
[3] Update rootfs
[4] Update u-boot
[5] Update dtb
[6] Update ALL
[7] System install or recover
[8] Video resolution select
[9] Video rotation select
[a] U-Boot console
```

```
Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

3.选择"[1] System install to mmc (by usb)"或"[3] Update u-boot to mmc (by tftp)", 开始安装:

```
*** U-Boot Boot Menu ***
```

```
[1] System install to mmc device (by usb)
[2] System install to mmc device (by tftp)
[3] System install to mmc device (by mmc)
[4] System recover from mmc disk
[5] System recover last time boot
[6] Return
```

```
Press UP/DOWN to move, ENTER to select
```

```

*****
*****STARG 1*****
*****this stage for format SSD and copy file to backup*****
*****

[ 8.923993] FAT-fs (sdb1): Volume was not properly unmounted. Some data may be corrupt. Ple.
-----> stage1 mount /dev/sdb1 success <-----
-----> stage1 check_file_for_safe <-----

-----> stage1.1 fdisk disk <-----

would split sda to 2 partition! /dev/sda1 and /dev/sda3
[ 11.965852] sda: sda1 sda3

Disk /dev/sda: 15 GB, 16013942784 bytes, 31277232 sectors
/dev/sda1 261,22,17 1023,254,63 4194367 31277231 27082865 12.9G 83 Linux
/dev/sda3 0,1,1 261,22,16 63 4194366 4194304 2048M 82 Linux swap

-----> stage1.2 format / partition --- start <-----
mkfs2fs 1.45.6 (20-Mar-2020)
-----> stage1.2 format / partition --- success <-----
-----> stage1.2 format swap partition --- start <-----
-----> stage1.2 format swap partition --- success <-----

-----> stage1.3: copy file to SSD <-----
[ 23.410476] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null)
-----> copy uImage
'/mnt/usb0/install/uImage' -> '/mnt/usb1/uImage'
-----> copy rootfs.tar.gz (wait a few minutes)
rootfs.tar.gz
32,768 0% 0.00kB/s 0:00:00

```

安装成功之后系统会自动重启。

1.4.3. EMMC更新内核

1.将 **内核 (uImage)** 放在tftp服务器**根目录**下，或者U盘**update**目录下。

2.uboot菜单选择“[2] Update kernel”

```

*** U-Boot Boot Menu ***

[1] System boot select
[2] Update kernel
[3] Update rootfs
[4] Update u-boot
[5] Update ALL
[6] System install or recover
[7] Board product
[8] Video resolution select
[9] Video rotation select
[a] U-Boot console

Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit

```

3.选择“[1] Update kernel (ulmage) (by usb)”或“[2] Update kernel (ulmage) (by tftp)”，开始升级：

```
[1] Update kernel (uImage) (by usb)
[2] Update kernel (uImage) (by tftp)
[3] Return
```

```
*** U-Boot Boot Menu ***
```

```
[1] Update kernel (uImage) (by usb)
[2] Update kernel (uImage) (by tftp)
[3] Return
```

Press UP/DOWN to move, ENTER to select

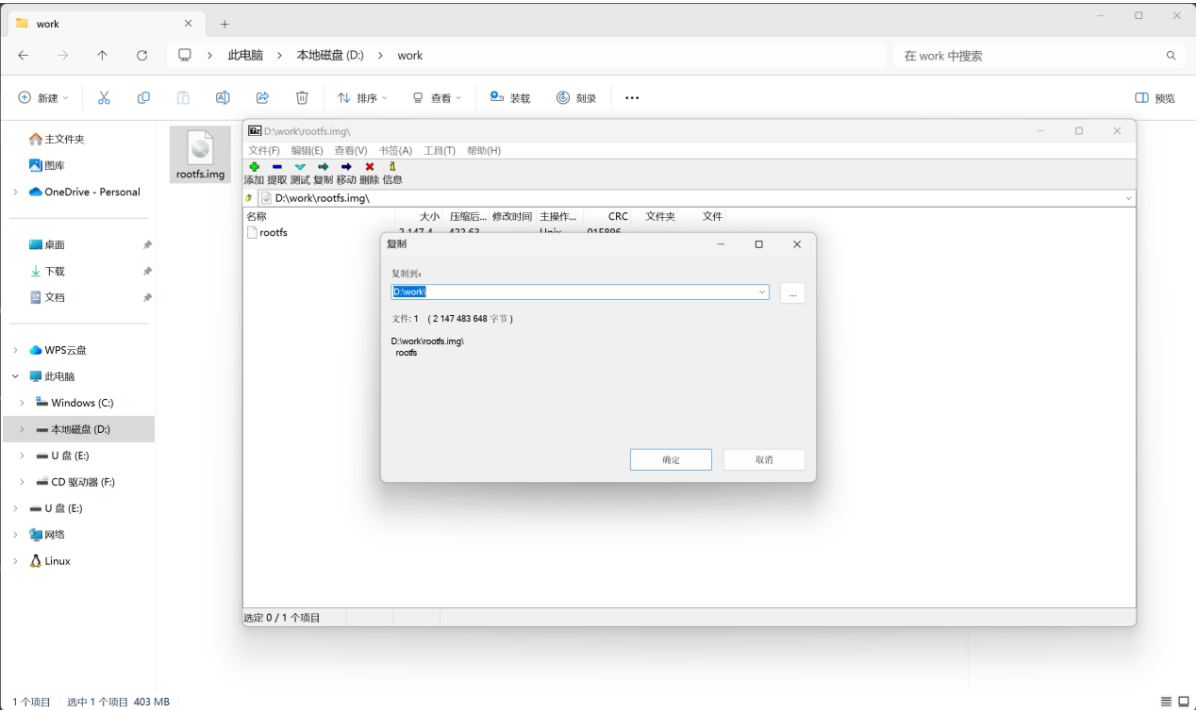
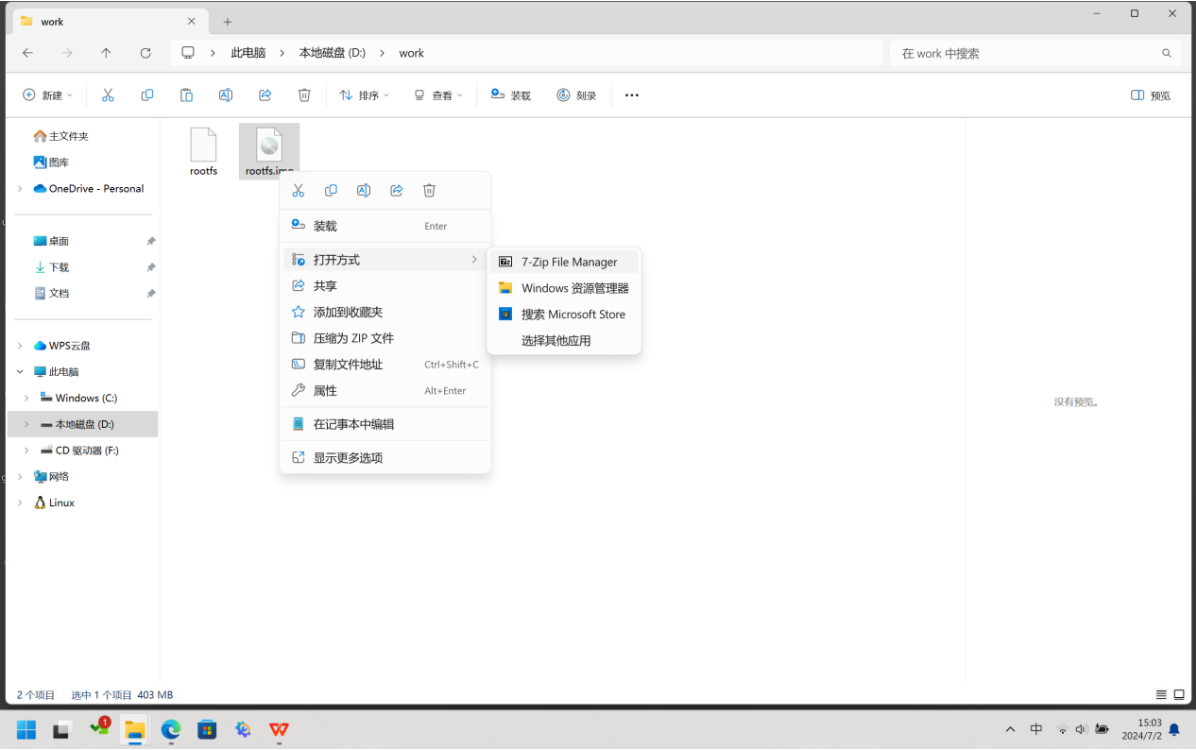
```
--if net:192.168.1.2 --fmt tftp --sym uImage  
--of mmc0 --fmt ext4 --sym /boot/uImage  
loading...  
ethernet@0x16020000 Waiting for PHY auto negotiation to complete..... done  
Speed: 1000, full duplex  
Using ethernet@0x16020000 device  
TFTP from server 192.168.1.2; our IP address is 192.168.1.20  
Filename 'uImage'.  
Load address: 0x9000000003000000  
Loading: #####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####!  
1.3 MiB/s  
done  
Bytes transferred = 8667379 (8440f3 hex)  
loaded&burning 8667379 bytes ...  
File System is consistent  
file found, deleting  
update journal finished  
File System is consistent  
update journal finished  
load&burn 8667379 finished  
=>
```

2K300 蜂鸟板支持从TF卡启动系统，TF卡方便拆下用读卡器连接到PC上进行各种定制修改，且不像EMMC系统需要关心镜像大小，只需要一个rootfs.img镜像文件即可。

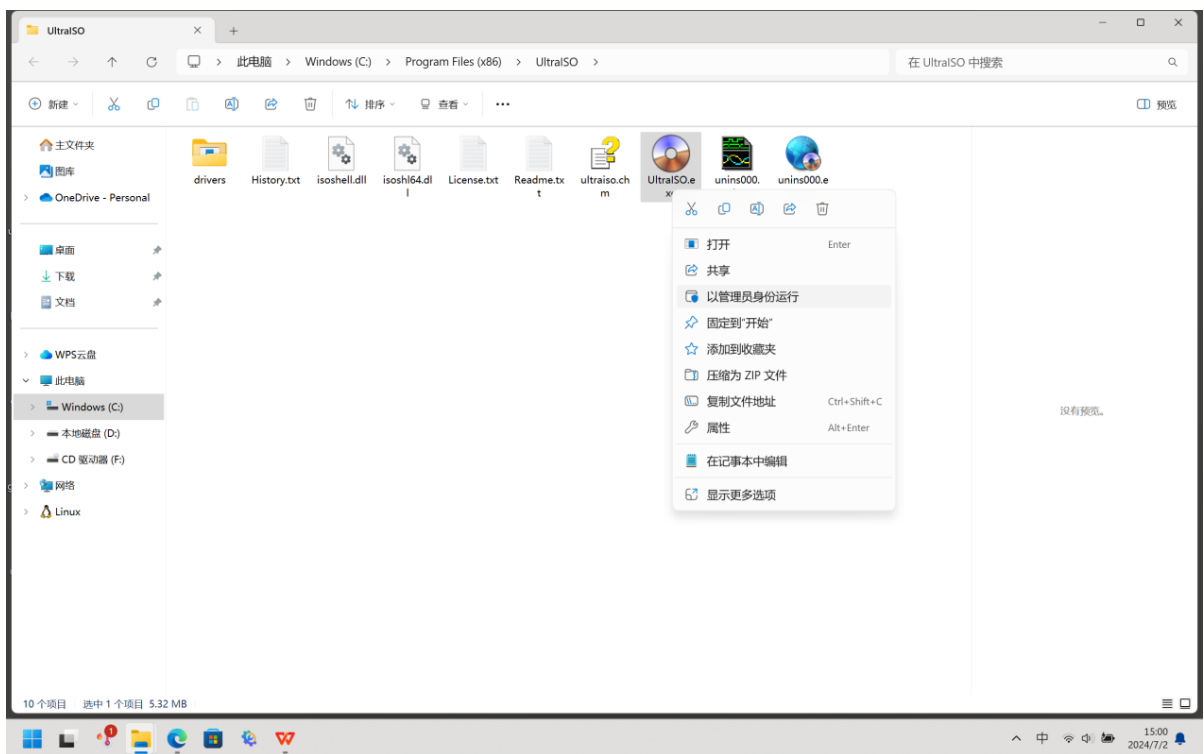
1.5.1. TF卡系统安装

a.Windows下安装TF卡系统

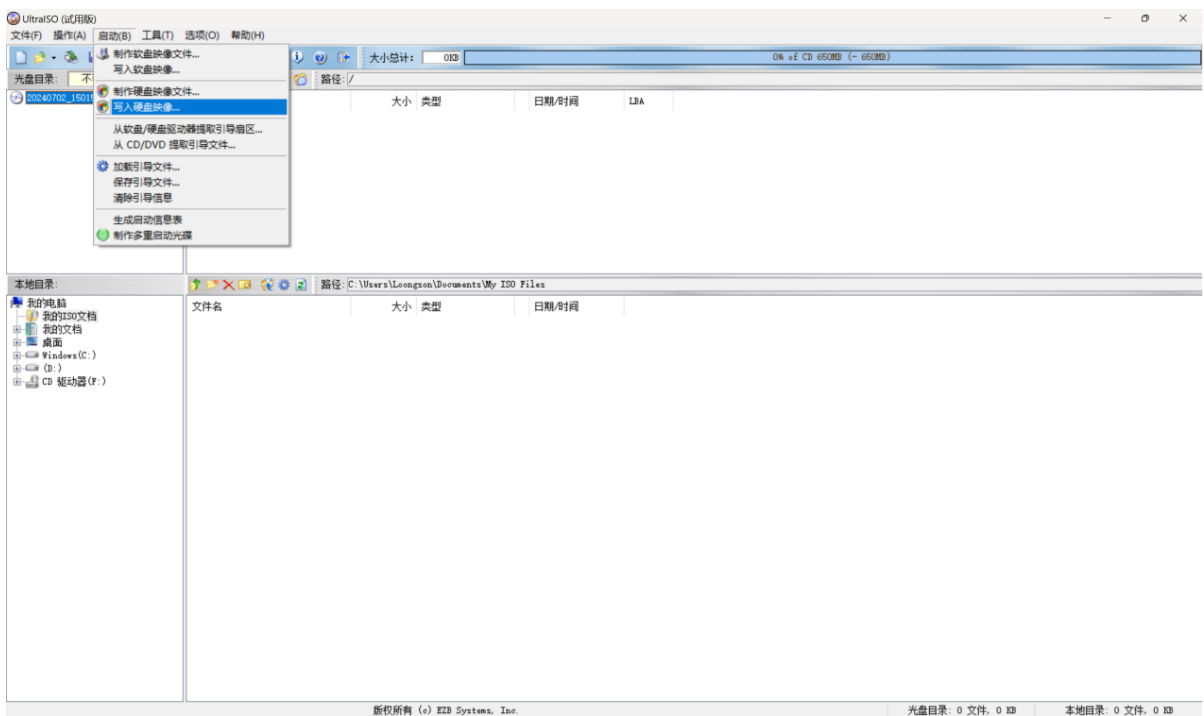
- 1.提前安装7Z（或其他解压缩工具）、UltraISO工具
- 2.通过读卡器将TF卡插在PC上
- 3.用 7Z 或其他解压工具打开 rootfs.img，解压后生成 rootfs。



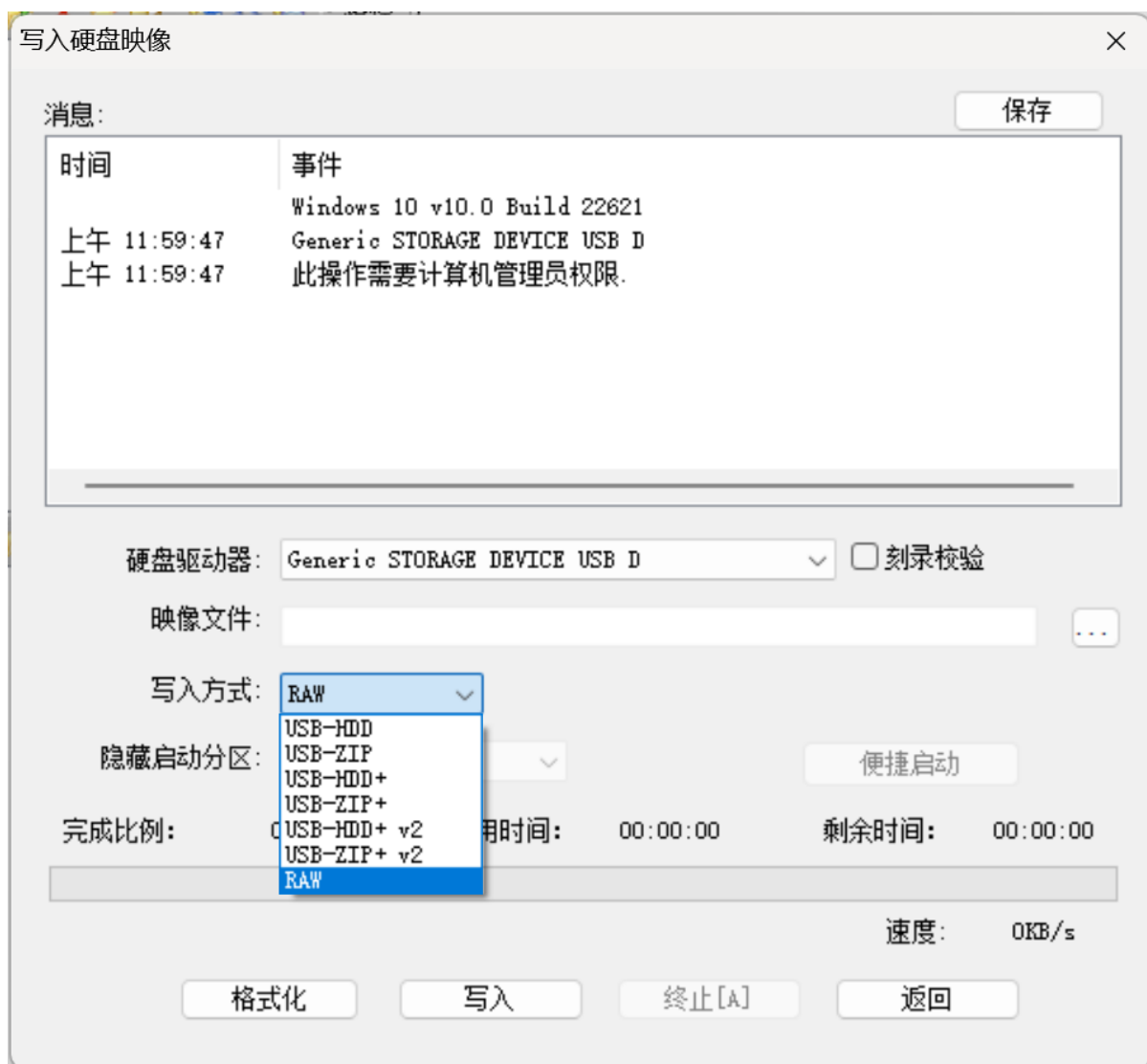
4.以管理员身份运行ultraiso工具



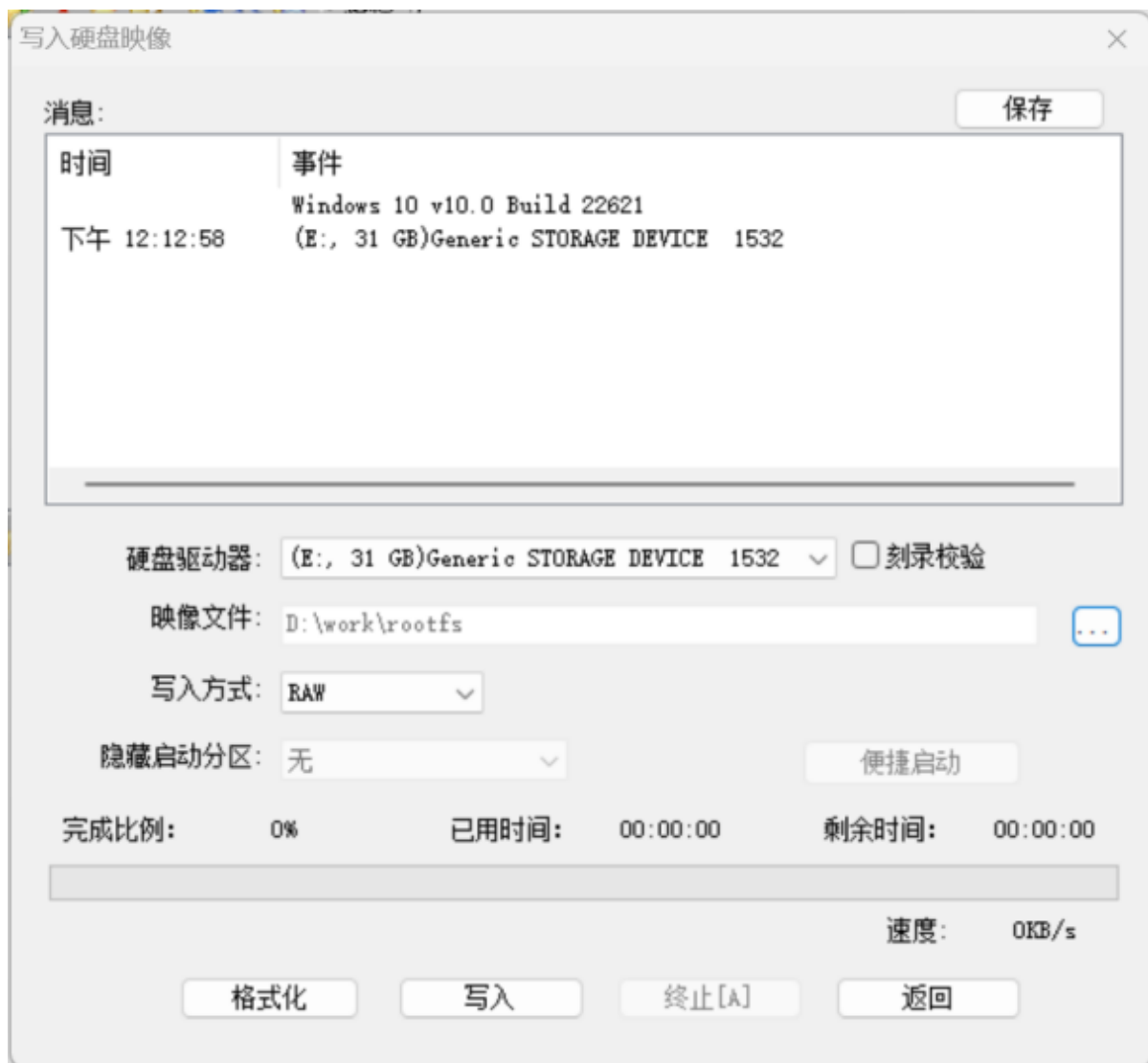
5.选择“启动”->“写入硬盘镜像”



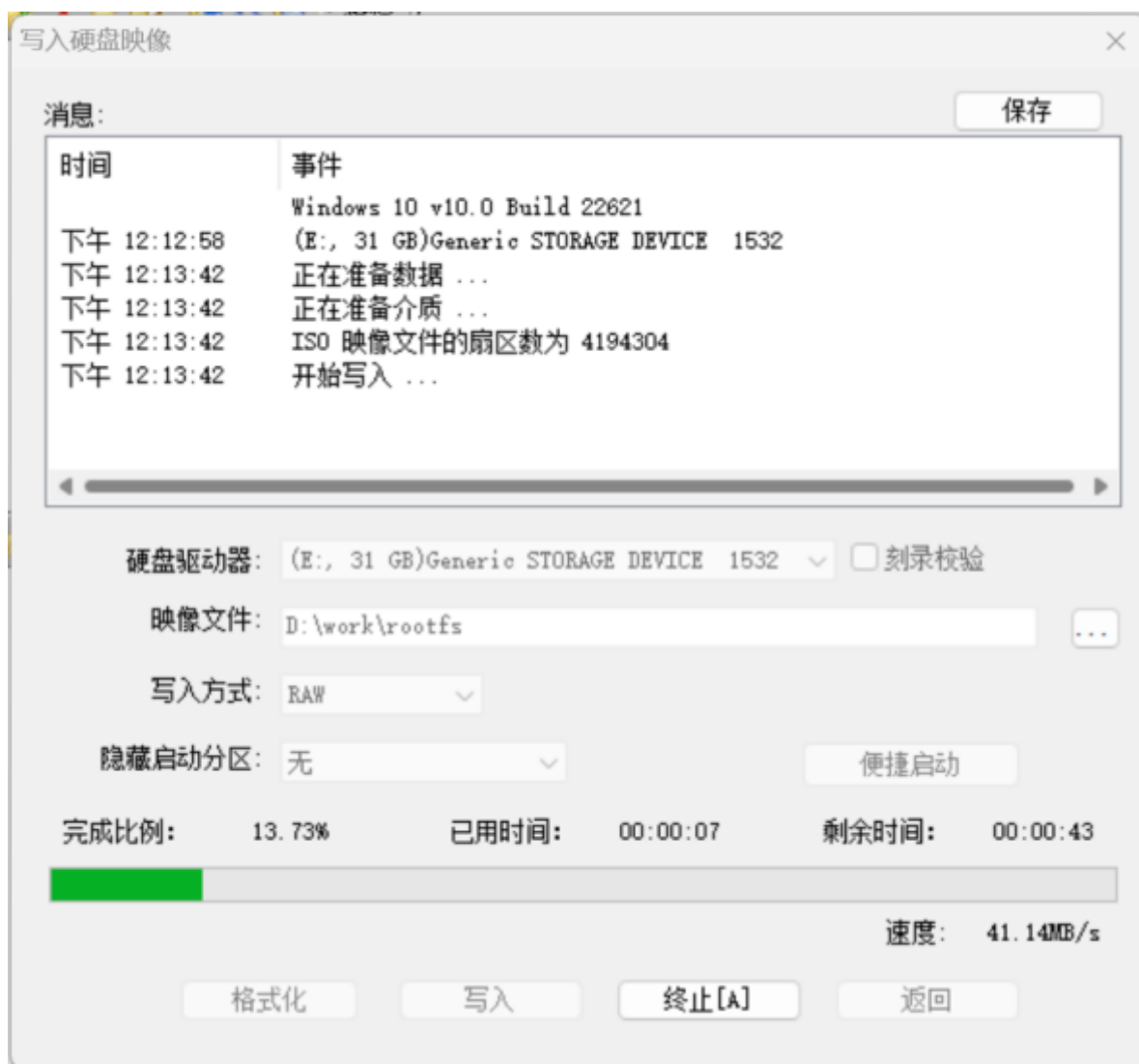
6.写入方式改成“RAW”



7.选择映象文件“rootfs”



8.点击“写入”，等待完成



b.Linux下安装TF卡系统

1.通过读卡器将TF卡插在PC上，得到设备符，比如 `/dev/sdX`，并确保没有挂载

```
$ sudo umount /dev/sdX
```

2.接下来进行解压缩与写入操作

```
$ gunzip -S .img rootfs.img
$ sudo dd if=rootfs of=/dev/sdX
```

1.5.2.TF卡系统更新内核

TF卡可以灵活拆下，所以更新TF卡系统内核只需要用读卡器连接到PC上，替换 `boot/uImage` 即可。或者通过U盘，TFTP, SCP 等方式更新亦可。

这些操作在 linux-PC 上是非常简单的，通过读卡器将TF卡插在PC上，得到设备符，比如 `/dev/sdX`。

```
$ sudo mount /dev/sdX1 /media
$ sudo cp uImage /media/boot/uImage
$ sudo umount /dev/sdX1
```

1.6.设备树 (DTB) 更新

可以在linux内核中执行 `make dtbs` 编译新的dtb, 生成需要的 `ls2k300_mini_dp.dtb`

```
make dtbs
```

1.将编译好的 `ls2k300_mini_dp.dtb` 改名为 `dtb.bin` 放在U盘 `update` 目录, 或者 tftp 根目录下。

2.uboot菜单选择“Update dtb”

```
*** U-Boot Boot Menu ***
```

```
[1] System boot select
[2] Update kernel
[3] Update rootfs
[4] Update u-boot
[5] Update dtb
[6] Update ALL
[7] System install or recover
[8] Video resolution select
[9] Video rotation select
[a] U-Boot console
```

```
Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

3.选择“Update DTB (dtb.bin) (by usb)”或“Update DTB (dtb.bin) (by tftp)”, 开始升级。 (也可以选择“Clean DTB parts” 使用默认 DTB)

```
*** U-Boot Boot Menu ***
```

```
[1] Update DTB (dtb.bin) to spi flash (by usb)
[2] Update DTB (dtb.bin) to spi flash (by mmc)
[3] Update DTB (dtb.bin) to spi flash (by tftp)
[4] Clean DTB parts
[5] Return
```

```
Press UP/DOWN to move, ENTER to select
```

```
*** U-Boot Boot Menu ***
```

```
[1] Update DTB (dtb.bin) to spi flash (by usb)
[2] Update DTB (dtb.bin) to spi flash (by mmc)
[3] Update DTB (dtb.bin) to spi flash (by tftp)
[4] Clean DTB parts
[5] Return
```

```
Press UP/DOWN to move, ENTER to select
```

```
*** U-Boot Boot Menu ***
```

```
[1] Update DTB (dtb.bin) to spi flash (by usb)
[2] Update DTB (dtb.bin) to spi flash (by mmc)
[3] Update DTB (dtb.bin) to spi flash (by tftp)
[4] Clean DTB parts
[5] Return
```

```
Press UP/DOWN to move, ENTER to select
```

4.升级完成，重启输入 `reboot` 或 `reset`

1.7.固件更新

注意：烧录固件需谨慎，可能导致板卡无法启动，

另外更新固件后第一次启动有个训练内存的过程，在输出 LOONGSON(GD) 的logo后会有大约15秒左右无输出，这个是正常现象。

1.7.1.uboot菜单更新固件

1.将 固件 (`u-boot-with-spl.bin`) 放在tftp服务器根目录下，或者U盘update目录下。

1.uboot菜单选择"[4] Update u-boot"

```
*** U-Boot Boot Menu ***
```

- [1] System boot select
- [2] Update kernel
- [3] Update rootfs
- [4] Update u-boot
- [5] Update ALL
- [6] System install or recover
- [7] Video resolution select
- [8] U-Boot console

```
Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

2.选择“[1] Update u-boot to spi flash (by usb)”或“[2] Update u-boot to spi flash (by tftp)”，开始升级：

```
*** U-Boot Boot Menu ***
```

- [1] Update u-boot to spi flash (by usb)
- [2] Update u-boot to spi flash (by tftp)
- [3] Return

```
Press UP/DOWN to move, ENTER to select
```

```
*** U-Boot Boot Menu ***
```

- [1] Update u-boot to spi flash (by usb)
- [2] Update u-boot to spi flash (by tftp)
- [3] Return

```
Press UP/DOWN to move, ENTER to select
```

3.升级完成，重启输入 `reboot` 或 `reset`

```

update u-boot.....
try to get u-boot-with-spl.bin .....
Speed: 1000, full duplex
Using ethernet@0x1f020000 device
TFTP from server 192.168.1.2; our IP address is 192.168.1.20
Filename 'u-boot-with-spl.bin'.
Load address: 0x9000000003000000
Loading: #####
2.1 MiB/s
done
Bytes transferred = 847078 (cece6 hex)
SF: Detected w25q80bl with page size 256 Bytes, erase size 4 KiB, total 1 MiB
Erase uboot partition ... SF: 942080 bytes @ 0x0 Erased: OK
device 0 offset 0x0, size 0xcece6
847078 bytes written, 0 bytes skipped in 8.488s, speed 102144 B/s
save bdinfo environment
Erasing 0x00000000 ... 0x00001fff (2 eraseblock(s))
Writing 1024 byte(s) at offset 0x00000000

#####
### update target: uboot
### update way : tftp
### update result: success
#####

```

1.7.2.使用系统工具更新固件

板卡自带的文件系统里面含有更新uboot的软件，存放在 `/root/sys_config_tool` 目录下，更新步骤如下：

- 1.上电后等待板卡进入系统
- 2.通过u 盘拷贝或网络传输的方式(scp, tftp)将 **固件 (u-boot-with-spl.bin)** 及 **md5校验文件(u-boot-with-spl.bin.md5)** 放到板卡的`/root/sys_config_tool/file`目录下。
- 3.校验 `/root/sys_config_tool/file`下的固件，确保固件的完整性。

```

# cd sys_config_tool/file
# ls u-boot-with-spl.bin*
u-boot-with-spl.bin  u-boot-with-spl.bin.md5
# md5sum -c u-boot-with-spl.bin.md5
u-boot-with-spl.bin: OK

```

- 4.回到`sys_config_tool` 目录，运行 `update_uboot` 程序进行更新，更新完成后重启即可。

```

# cd /root/sys_config_tool/file
# ./update_uboot
start burn uboot file!
burn uboot file success!
Syncing ...
# reboot

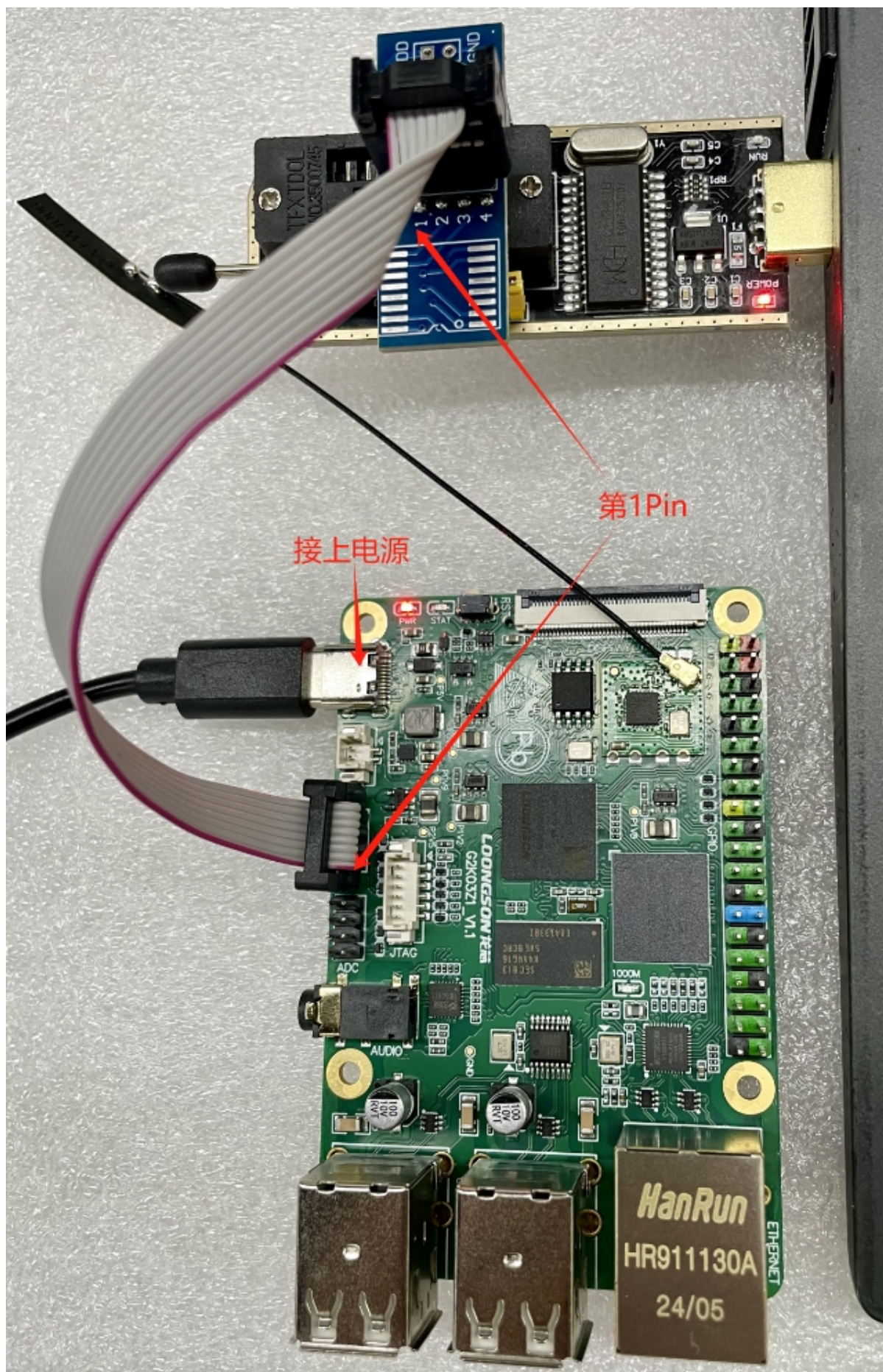
```

1.7.3.烧录器更新固件

1.7.3.1.烧录器接线

先将先锋板接上电，再将SPI 烧录器的一头接在2K0300蜂鸟板的SPI烧录口上（注意线序），一头接在PC机上。连接方式如图：

注意：烧录时建议2K0300蜂鸟板**先接电源**，以避免USB供电不足从而引起烧录失败。

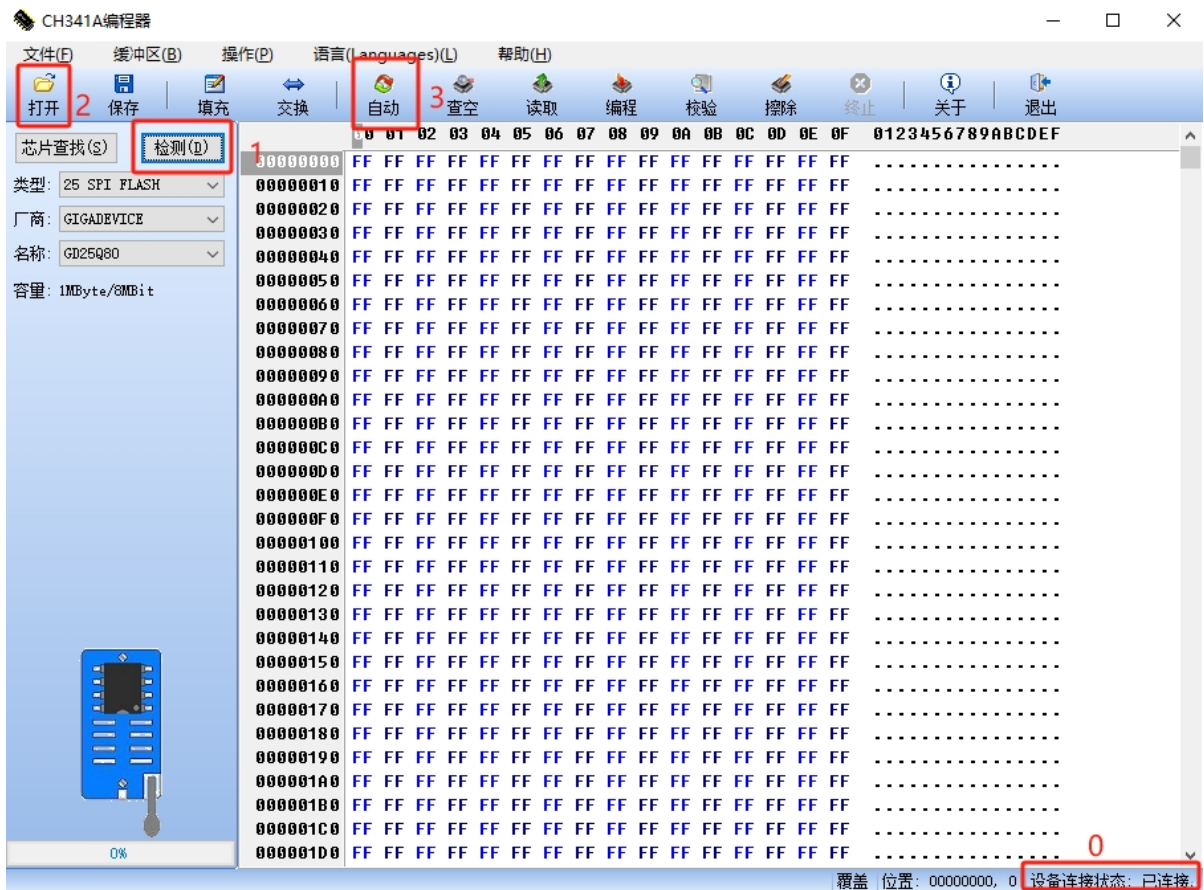


1.7.3.2.烧录器烧录

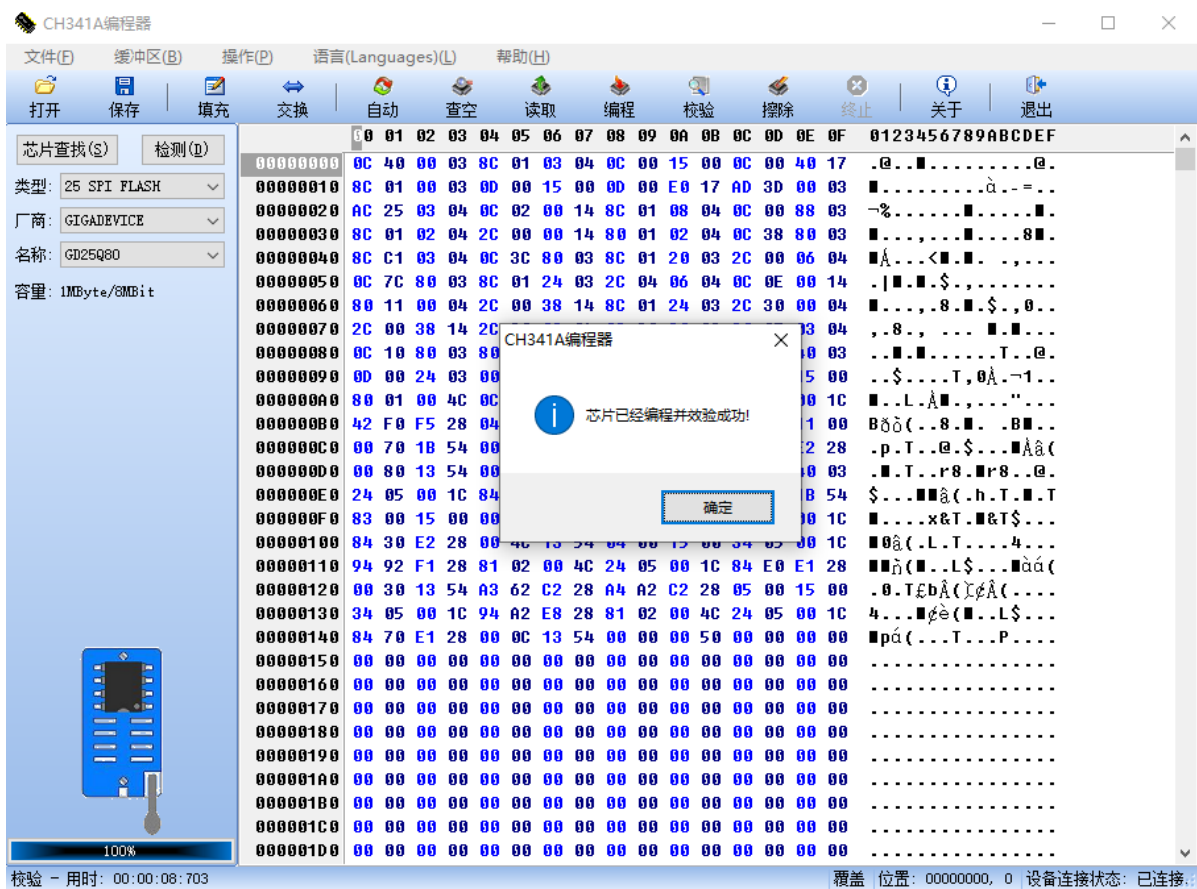
烧录有自动与手动两种模式，如自动模式烧录失败，建议采用手动模式烧录。

1.7.3.3.自动模式烧录

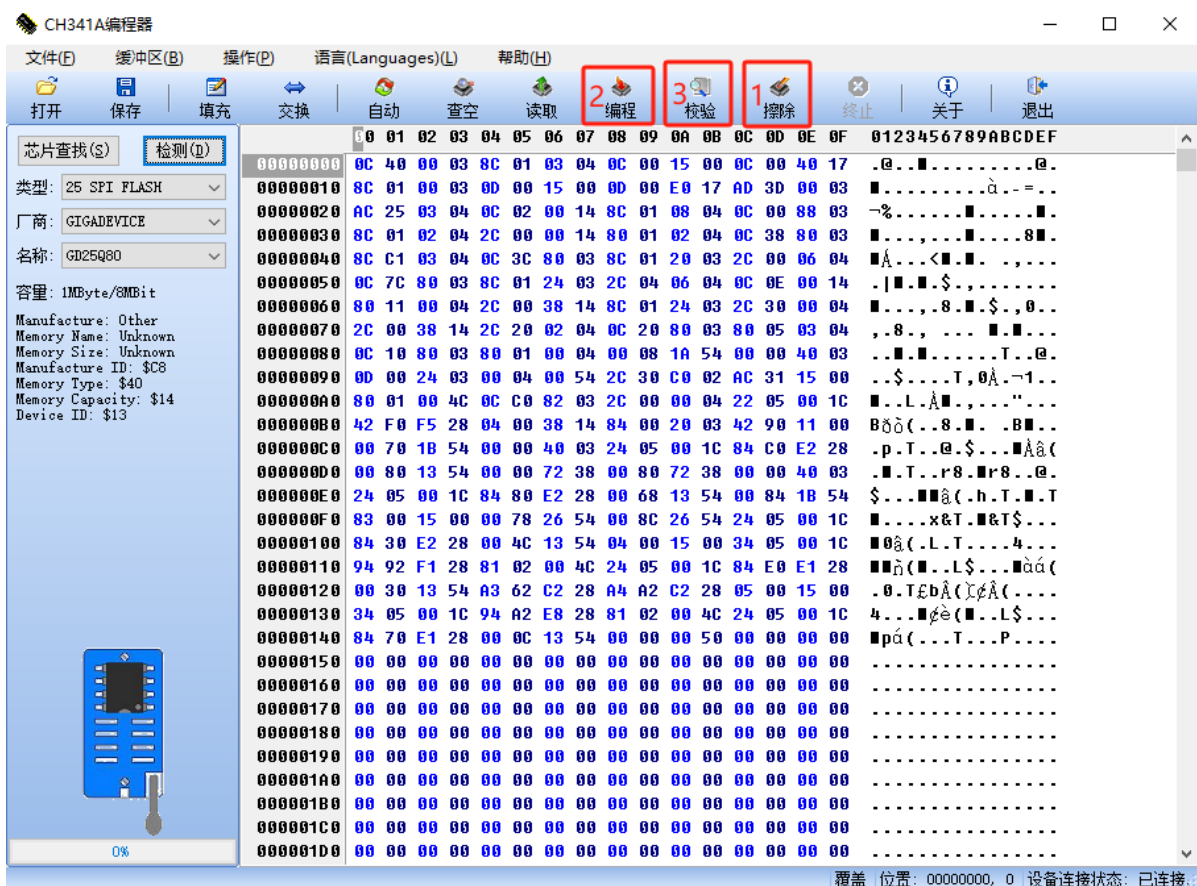
- 1.先检查烧录器是否正常被烧录软件识别
- 2.点击"检测" 以识别SPI 芯片
- 3.点击"打开" 导入要烧录的固件
- 4.点击"自动" 开始烧录，烧录器软件会显示烧录进度



- 5.烧录成功，烧录器软件会有弹框提示

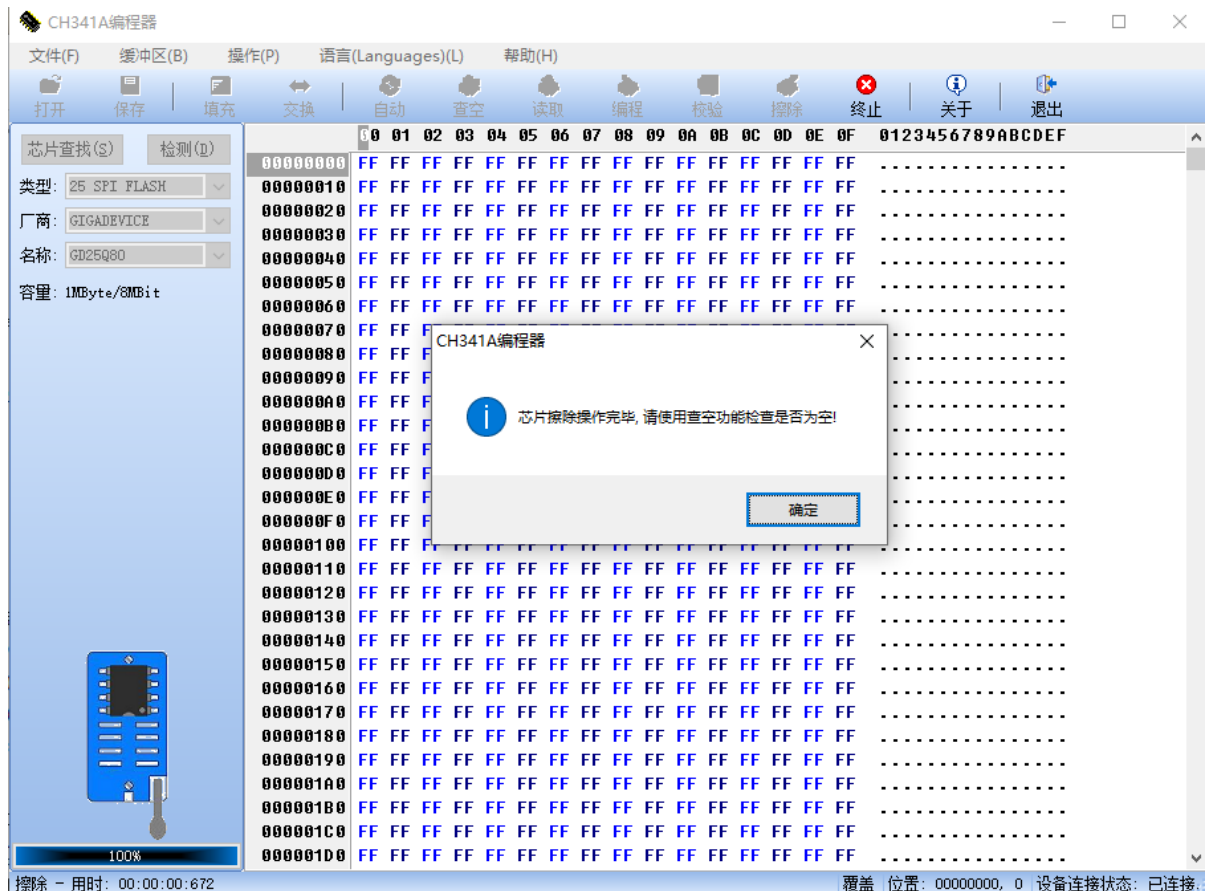


1.7.3.4 手动模式烧录



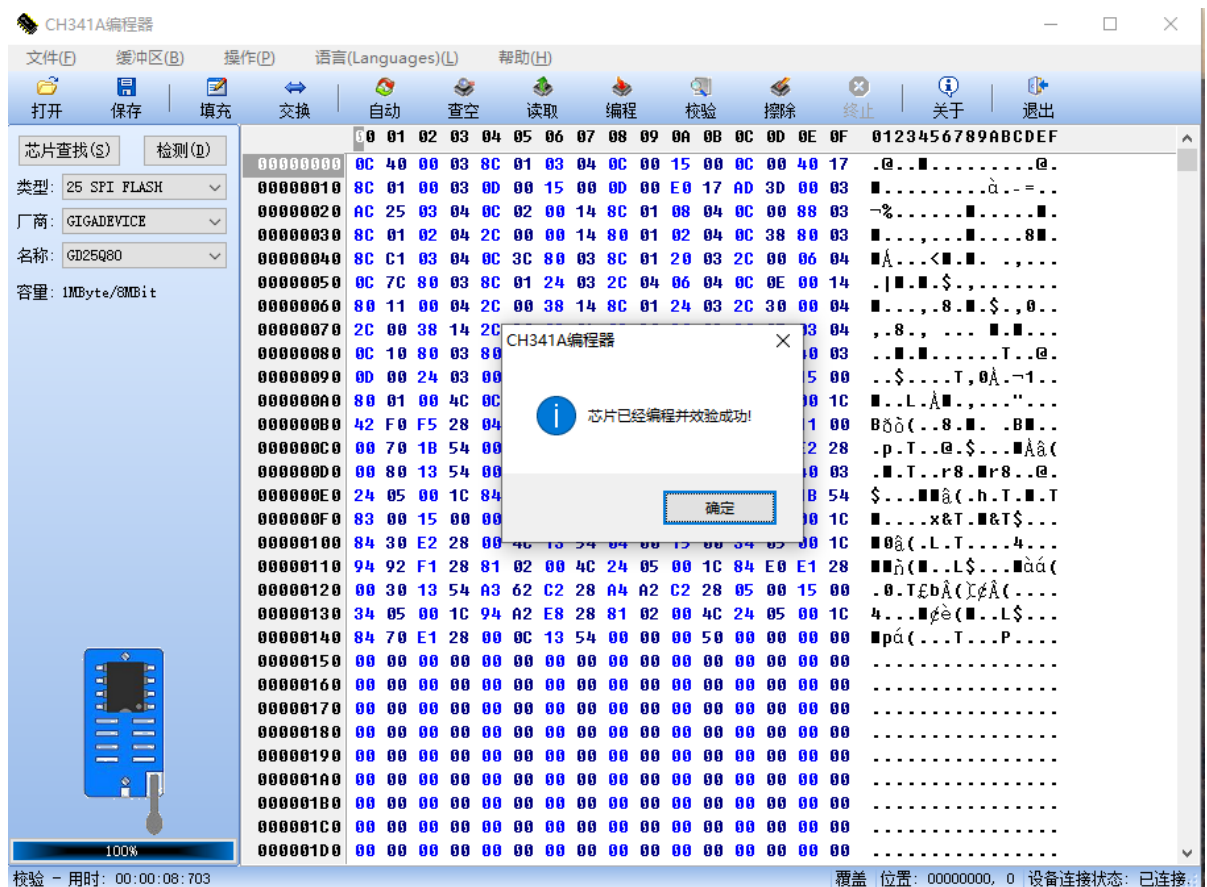
- 1.先检查烧录器是否正常被烧录软件识别
- 2.点击"检测" 以识别SPI 芯片
- 3.点击"打开" 导入要烧录的固件

4.点击"擦除" 开始擦除芯片，之后会弹框提示查空，可以忽略。



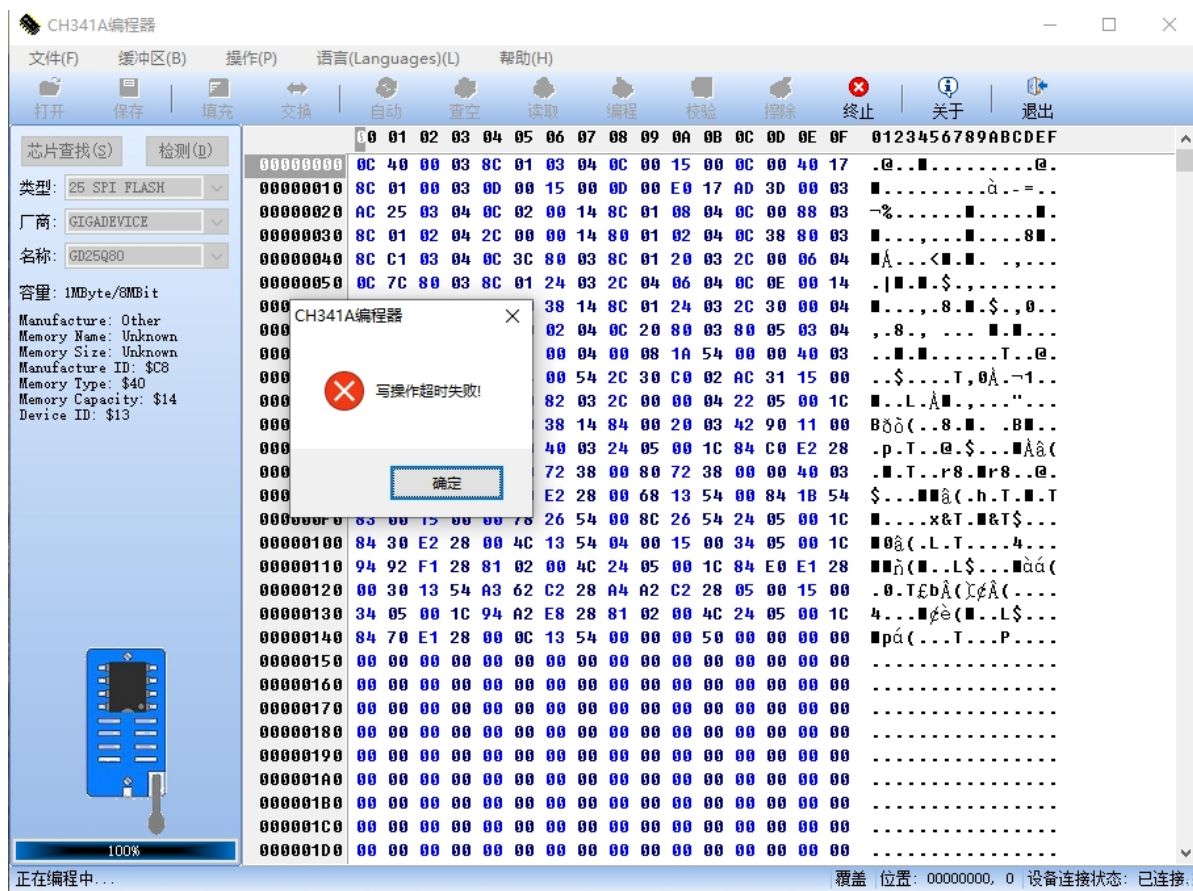
5.点击"编程" 开始烧录

6.待"编程"完成到 100%后，点击"校验" 以确保烧录的正确性。



注:

有时点击"自动"开始烧录时，会弹出"写超时失败"的弹框，可以忽略此警告，再次点击"自动" 进烧录即可。



1.7.4 JTAG 烧录固件

龙芯JTAG 仿真器支持在线调试和烧录功能，JTAG 仿真器软件的安装说明见[仿真器软件--->lsdebugger.pdf](#) 文件。

JTAG 仿真器软件支持多种平台，这里以window 系统为例进行使用说明。

解压loongson-debugger-mingw-20240324.tar.xz 到任意目录（建议路径无中文）。进入目录双击 `1a32_dbg_tool_usb.exe` 打开软件，会弹出个终端。

在终端输入输入相应的命令即可。具体流程如下：

- 1.将要烧录的文件（u-boot-with-spl.bin）先负责到JTAG仿真器软件软件目录，后面烧录需要使用到
- 2.连接jtag与板卡，再将jtag 仿真器的usb 接到PC上
- 3.打开 `1a32_dbg_tool_usb.exe` ， 会弹出命令交互终端（命令支持Tab 键自动补全）。
- 4.指定芯片配置，在命令交互终端输入 `source configs/config.ls2k300`
- 5.开发板上电
- 6.命令交互终端输入 `set` 指令，检查是否有寄存器的值返回，有表示连接正常。如果卡住或无返回可以尝试通过复位按钮对板卡进行复位。然后重复步骤2~6
- 7.命令交互终端输入 `program_cachelock_spi uboot-with-spl.bin` 进行烧录

```
cpu0 -source configs/config.ls2k300 #先指定芯片配置文件
##jtag_clk 1 2
#jtag_clk 8 2
#expr1 match "/03-soft/loongson-debugger/1a32_dbg_tool_usb" .*gpio
```

```
#if 0x0 ret
#let1 clk 8
#expr 0x10000|8
#usblooptest 0x81000070 0x10008
#expr 3>2
#do if 0x1
#let1 phase 2
#expr 0x20000|2
#usblooptest 0x81000070 0x20002
#end
#ret
#setconfig core.cpucount 1
#setconfig core.cpuwidth 64
#setconfig core.abisize 64
#setconfig helpaddr 0x900000000000f000
#setconfig putelf.uncached 0
#setconfig usb_ejtag.put_speed 0
#setconfig usb_ejtag.get_speed 0
#setconfig core.nocache 1
#setconfig jtag.pcswidth 83
#setconfig jtag.synci 1
#setconfig jtag.jrhb 1
#setconfig jtag.jalrhb 1
#setconfig put.fastdata 0
#setenv ENV_memsize 256
##setenv ENV_highmemsize 1792
#setenv ENV_cpuclock 800000000
#let spibase 0x8000000016010000
#setconfig spi.iobase 0x8000000016010000
#let1 ejtag_spibase 0xdb00000000ffff00
##setconfig flash.type byte
#setconfig flash.type page
#usblooptest 0x81000070 0x20001
#fix_ejtag_la
#expr1 ( match "/03-soft/loongson-debugger/la32_dbg_tool_usb" .*usb ) == 0
#if 0x0 ret
#usbver
#expr 0x20210129==0x20150105
#do if 0x0
#usblooptest 0x4c
#expr 0x00001920!=0
#elseif 0x1
#usblooptest 0x40
#usblooptest 0x00001914 4
#usblooptest 0x44
#usblooptest 0x00001918 0
#usblooptest 0x48
#usblooptest 0x0000191c 5
#usblooptest 0x4c
#usblooptest 0x00001920 4
#usblooptest 0x50
#usblooptest 0x00001924 3
#usblooptest 0x54
#usblooptest 0x00001928 8
#end
#ret
```

```

#expr "(0x1 << 16)"
#letl PLL_L1_LOCKED      0x10000
#expr "(0x1 << 2)"
#letl PLL_L1_ENA          0x4
#expr "(0x1 << 2)"
#letl PLL_MEM_ENA        0x4
#expr "(01 << 16)"
#letl PLL_MEM_LOCKED     0x10000
#letl PLL_CHANG_COMMIT 0x1
#letl DDR_REFC           4
#letl DDR_DIV            1
#letl DDR_DIV_L2         4
#letl GPU_DIV_L2         5
#letl PLL_IN 100000000
#expr 333000000/10*1*4*4/(100000000/10)
#letl DDR_LOOPC 0x35
##/* CPU @ 1000Mhz */
#letl L1_LOOPC      80
#letl L1_REFC        4
#letl L1_DIV          1
#letl L2_DIV          2
#expr 1000000000/10*1*4*2/(100000000/10)
#letl L1_LOOPC 0x50
##dc 200M
##gmac 125M
#letl DC_LOOPC      80
#letl DC_REFC        4
#letl DC_DIV          1
#letl DC_DIV_L2      8
#letl GMAC_DIV       16
##letl DC_LOOPC {250000000/10*${DC_DIV}*${DC_REFC}*${DC_DIV_L2}/(${PLL_IN}/10)}
##letl PIX0_LOOPC      109
#letl PIX0_REFC 5
#letl PIX0_DIV 1
#letl PIX0_DIV_L2      20
#expr 64000000/10*1*5*20/(100000000/10)
#letl PIX0_LOOPC 0x40
##letl PIX1_LOOPC      109
#letl PIX1_REFC 5
#letl PIX1_DIV 1
#letl PIX1_DIV_L2      20
#expr 64000000/10*1*5*20/(100000000/10)
#letl PIX0_LOOPC 0x40
#let rtc_reg 0xffffffffbfff6c100
#letl pcicfbase 0x800000fe00000000
#letl pcicfbase1 0x800000fe00000000
##letl pcicfbase 0x8000000016800000
##letl pcicfbase1 0x8000000016800000
#letl i2creg 0xffffffffbffe01000
#dellabel cache_init
#source "scripts/gmac.cmd"
#letl phy 16
#letl base 0xffffffffbffe10000
#letl coherent_base 0x9800000000000000
#letl base 0x800000001f020000
#letl phy 24

```

```

##letl base {($ (pci_config_read 0 0x3 0 0x10)&~0xf)|0x9000000000000000}
#source scripts/nand.cmd
#letl nand_msize nand_oseize nand_cap nand_esize 8192 256 7 0x20000
#letl ncmd 0xfffffffffbfe06000
#letl orderreg 0xfffffffffbfe10c00
#letl ncmd 0xfffffffffbfe06000
#letl orderreg 0xfffffffffbfe10c00
#source scripts/spi.cmd
#let spibase 0x8000000016010000
#let spi_cs 0
#let spi_speed 4
#let spibase 0x8000000016010000
#let spi_cs 0
#dellabel gdb_module_setup
##acpi_gmac_suspend()
##{
##devmem 0x4004002c 32 $((($ (devmem 0x4004002c)|0x203))
##devmem 0x4005002c 32 $((($ (devmem 0x4005002c)|0x203))
##devmem 0x1fe0702c 32 0x70
##devmem 0x1fe07028 32 0x0000ffff
##devmem 0x1fe0700c 32 0x0000ffff
##devmem 0x1fe07008 32 $((($ (devmem 0x1fe07008)|0x80))
##devmem 0x1fe07004 32 $((($ (devmem 0x1fe07004)|0x80))
##devmem 0x1fe07014 32 $(((1<<13)|($ {1:-5}<<10)))
##}
#source scripts/i2c.cmd
#let i2c_noack 0
#source scripts/spi.cmd
#let spibase 0x8000000016010000
#let spi_cs 0
#let spi_speed 4
#let spi_cs 0
#source scripts/dumpserial.cmd
#let kargs 'g console=ttyS0,115200 log_buf_len=10M initcall_debug=1 loglevel=20
nousb'
#let kernel /tmp/vmlinuz
#let rd /tmp/rootfs.cpio.gz
#let iobase 0x8000000000000000
#info f

```

name	type	line	contents
gdbaccess	f	48	gdbmap \$1 \$2 \$3 \$2 \$4
gdb	f	70	letl m \$(setconfig gdbserver.cpubitmap)
gdb_remote	f	71	letl m \$(setconfig gdbserver.cpubitmap)
gdbmod_remote	f	72	letl m \$(setconfig gdbserver.cpubitmap)
gdbmod	f	73	letl m \$(setconfig gdbserver.cpubitmap)
gdbmod0_remote	f	74	letl m \$(setconfig gdbserver.cpubitmap)
gdbmod0	f	75	letl m \$(setconfig gdbserver.cpubitmap)
eclipse	f	76	letl m \$(setconfig gdbserver.cpubitmap)
eclipse_remote	f	77	letl m \$(setconfig gdbserver.cpubitmap)
eclipsemod	f	78	letl m \$(setconfig gdbserver.cpubitmap)
eclipsemod_remote	f	79	letl m \$(setconfig gdbserver.cpubitmap)
ddd	f	80	letl m \$(setconfig gdbserver.cpubitmap)
ddd_remote	f	81	letl m \$(setconfig gdbserver.cpubitmap)
dddmod	f	82	letl m \$(setconfig gdbserver.cpubitmap)
dddmod_remote	f	83	letl m \$(setconfig gdbserver.cpubitmap)
fls	f	189	letl num i "\$1" 31

cache_config	f	198	setconfig core.nocache 1
check_ejtag	f	214	letl clkdiv \${1:1}
ejtag_check	f	215	letl clkdiv \${1:1}
gdbserver	f	254	letl m \$(setconfig gdbserver.cpubitmap)
localmem	f	291	newfunc f_d "devmem \{\$\$2&0xffffffff\} \{%d \$1*8\} \$\$3 \$\$4"
devmem	f	307	newfunc f_d "echo_2 devmem \{\$\$2&0xffffffff\} \{%d \$1*8\} \$\$3 \$\$4"
dummy	f	323	newfunc f_dq "echo_2 \$1 \$2; echo \$\$RANDOM;"
myput64	f	338	letl f \${1:/srv/tftp/vmlinux}
jtag_clk	f	352	if \$(expr1 match "\${EJTAGEXE}" .*gpio) ret
rm_jtag_init	f	361	devmem 0x1e6e4008 32 {\$(devmem 0x1e6e4008) (3<<30)}
la64_jtag_init	f	371	devmem 0x1fe10444 32 {\$(devmem 0x1fe10444)&~ (0xb (5<<25))) 4}
ls1c_ejtag_init	f	381	devmem 0x1fd011c4 32 {\$(devmem 0x1fd011c4)&~ ((0xf<<14) (1<<21) (1<<27))}
ls1c_cjtag_init	f	409	devmem 0x1fd011c4 32 {\$(devmem 0x1fd011c4)&~ ((0xf<<14) (1<<21) (1<<27))}
cjtag_wait_reset	f	434	do while {\$(jtagregs d8q 1 1)!=0x20010819}
mysi	f	440	hb 0 -1
mySi	f	444	Hb 0 -1
fix_ejtag_la	f	448	if \$(expr1 (match "\${EJTAGEXE}" .*usb) == 0) ret
configserial	f	565	let serialdiv \${1:{125000000/115200/16}}
testserial	f	583	let serialdiv \${1:{125000000/115200/16}}
set_cpu_clk	f	649	letl t0 0xffffffffbfe10480
set_ddr_clk	f	659	letl t0 0xffffffffbfe10490
set_dc_clk	f	669	echo dc
set_pix0_clk	f	680	echo pix0
set_pix1_clk	f	691	letl t0 0xffffffffbfe104c0
rtc_set	f	702	letl y m d h M s \${1:2016} \${2:1} \${3:1} \${4:12} \${5:0} \${6:0}
rtc_alarmset	f	709	letl y m d h M s \${1:2016} \${2:1} \${3:1} \${4:12} \${5:0} \${6:0}
rtc_read	f	715	letl c \$(d4q \${rtc_reg+0x40} 1)
dm8	f	723	letl i \${1&0xfff}
configddr	f	730	cp0s 0 m8 12 {\$(cp0s 0 d8q 12 1)&~4} 0xe0}
dumpddr	f	853	letl ddrwidth \${1:0}
ddrtest	f	914	do if \$(test \$0 == ddrtest_init)
ddrtest_init	f	915	do if \$(test \$0 == ddrtest_init)
ddrtest_uc	f	916	do if \$(test \$0 == ddrtest_init)
erase1	f	935	#let spi_speed 4
erase	f	941	setconfig spi.iobase \$spibase
program	f	946	letl file \${1:/tmp/gzrom.bin}
program_bin	f	947	letl file \${1:/tmp/gzrom.bin}
pci_config_read_byte	f	964	letl bus dev func reg cnt "\$1" "\$2" "\$3" "\$4" "\$5"
pci_config_write_byte	f	969	let bus dev func reg data "\$1" "\$2" "\$3" "\$4" "\$5"
pci_config_read	f	979	local bus dev func reg cnt addr addrp q
pci_config_readv	f	980	local bus dev func reg cnt addr addrp q
pci_config_write	f	993	local bus dev func reg cnt addr type
pci_config_read_pci	f	1005	local bus dev func reg cnt addr addrp q
pci_config_read_pciv	f	1006	local bus dev func reg cnt addr addrp q
pci_config_write_pci	f	1019	local bus dev func reg cnt addr type

select_pci	f	1031	newcmd pci_config_read pci_config_read_pci
pci_find_cap	f	1047	letl bus dev func type \${1:0} \${2:0} \${3:0} \${4:0x10}
pcie_max_dev	f	1058	letl bus dev func \${1:0} \${2:0} \${3:0}
pci_list_bus	f	1072	letl show \$(setconfig sys.showcmd)
pci_payload	f	1104	letl bus dev func \${1:0} \${2:0} \${3:0}
bus2phys	f	1116	echo \$1
bus2virt	f	1119	echo \${1 0x9000000000000000}
dumpe1000e	f	1122	letl bus dev func \${1:0} \${2:0} \${3:0}
reboot_tos	f	1153	set_gpio_out 46 1
reboot1	f	1158	memset4 0x800000001fff6c030 1 4
reboot2	f	1161	letl v \$(d4q 0x8000000016010204 1)
reboot	f	1166	m4 0x800000001fff6c030 2
reboot_off	f	1171	m4 0x800000001fff6c014 0x3c00
poweroff	f	1174	m4 0x800000001fff6c00c \$(d4q 0x800000001fff6c00c 1)
i2c0_init	f	1179	let i2creg {0xfffffffffbfe01000+\${1:0}*0x800};
i2c0_read	f	1186	letl cr dr adr reg count \$(expr \$i2creg+0x4) \$(expr \$i2creg+0x3) "\$1" "\$2" \$(expr (\$#>3)*(\$3-1)+1)
i2c0_write	f	1209	letl cr dr adr reg val \$(expr \$i2creg+0x4) \$(expr \$i2creg+0x3) \${1:0} \${2:0} \${3:0}
i2c0_read2a	f	1224	letl cr dr adr reg count \$(expr \$i2creg+0x4) \$(expr \$i2creg+0x3) "\$1" "\$2" \$(expr (\$#>3)*(\$3-1)+1)
i2c0_write2a	f	1250	letl cr dr adr reg val \$(expr \$i2creg+0x4) \$(expr \$i2creg+0x3) \${1:0} \${2:0} \${3:0}
i2c0_read2	f	1268	letl cr dr adr reg count \$(expr \$i2creg+0x4) \$(expr \$i2creg+0x3) "\$1" "\$2" \$(expr (\$#>3)*(\$3-1)+1)
i2c0_write2	f	1296	letl cr dr adr reg val \$(expr \$i2creg+0x4) \$(expr \$i2creg+0x3) \${1:0} \${2:0} \${3:0}
net	f	1314	let gmacreg \$(pci_config_read 0 3 0 0x10 1)
exectest	f	1322	letl p 0xfffffffff90000000
exectest	f	3436	letl p 0xfffffffff90000000
cache_init	f	1333	setconfig core.nocache 0
cache_lock	f	1341	m8 0x8000000016000200 0x800000001c000000
cache_init_and_lock	f	1345	call cache_init
cache_init_and_lock1	f	1346	call cache_init
memtest_cachelock	f	1357	echo_on
program_cachelock_spi	f	1369	spi_memen
program_cachelock	f	1370	spi_memen
fprogram_cachelock_spi	f	1372	spi_memen
program_cachelock_bin	f	1373	spi_memen
get_cachelock	f	1415	setconfig helpaddr 0x900000001c000000
get_cachelock	f	3104	setconfig helpaddr 0x900000001c000000
reset_lpc	f	1427	letl v \$(d4q 0xfffffffffbfd00200 1)
enable_pcieclk	f	1433	m4 0xfffffffffbfe10430 \${\$(d4q 0xfffffffffbfe10430 1) 0x30000}
disable_pcieclk	f	1436	m4 0xfffffffffbfe10430 \${\$(d4q 0xfffffffffbfe10430 1)&~0x30000}
testpcie	f	1439	letl portnum \${1:0}
init_pcieport	f	1456	letl port \${1:0xe}
trainen_pcieport	f	1471	letl port \${1:0xe}
forcesta_pcieport	f	1488	letl port \${1:0xe}
init_pciephy	f	1498	m4 0xfffffffffbfe10580 0xc2492331
pcie_train	f	1514	letl port \${1:0xe}
pcie_lookback	f	1528	m1 0xb8110011 0x2
read_phy	f	1536	@letl show \$(setconfig sys.showcmd)

write_phy	f	1552	@letl show \$(@setconfig sys.showcmd)
write_phy_reg_val	f	1563	let phy \${1:0}
read_phy_reg_cnt	f	1567	let phy \${1:0}
print_speed	f	1571	letl stat \$1
print_speed1	f	1598	letl s \${(d4q \${base+0xd8} 1)}
find_phy	f	1620	letl phy0 -1
switch_mv88e6070_phy_read	f	1644	@echo_off
switch_mv88e6070_phy_write	f	1656	@echo_off
vbus2phys	f	1664	expr (\$1>0x90000000)?\$1:(\$1&0xffffffff)
gmacdesc	f	1667	letl bar0 \$base
read_switch	f	1711	letl oldphy \$phy
write_switch	f	1723	letl oldphy \$phy
config_switch	f	1738	write_switch 0x8 0x1c
set_gpio_out	f	1748	letl pin val \${1:0} \${2:0}
physaddr	f	1765	echo
			{(\$1>0xffffffff80000000&&\$1<0xffffffff90000000)?(\$1&0xffffffff):(\$1&0xffffffff)}
nand_init	f	1768	m4 0xffffffffbfe10420 {\$(d4q 0xffffffffbfe10420 1) (1<<9)}
nand_reset	f	1771	letl naddr \${ncmd+8}
nand_readid	f	1791	letl naddr \${ncmd+8}
nand_erase	f	1828	letl naddr \${ncmd+8}
nand_Erase	f	1854	letl naddr \${ncmd+8}
nand_read	f	1874	letl naddc \${ncmd+4}
nand_write	f	1915	letl naddc \${ncmd+4}
spi_set_cs	f	1962	do if \$1
spi_set_cs	f	2910	do if \$1
spi_send_data	f	1969	m1 \${spibase+0x2} \$1
spi_send_data	f	2917	m1 \${spibase+0x2} \$1
spi_qsend_data	f	1970	m1 \${spibase+0x2} \$1
spi_qsend_data	f	2918	m1 \${spibase+0x2} \$1
spi_readid	f	1976	call spi_init \$spi_speed
spi_readid	f	2924	call spi_init \$spi_speed
spi_read_area	f	1986	let addr cnt \${1:0} \${2:16}
spi_read_area	f	2934	let addr cnt \${1:0} \${2:16}
spi_wait_sr	f	2005	spi_set_cs 0
spi_wait_sr	f	2953	spi_set_cs 0
spi_set_wren	f	2011	spi_wait_sr
spi_set_wren	f	2959	spi_wait_sr
spi_write_sr	f	2021	let sr \${1:0}
spi_write_sr	f	2969	let sr \${1:0}
spi_read_sr	f	2030	spi_set_cs 0
spi_read_sr	f	2978	spi_set_cs 0
spi_write_area	f	2036	let addr \${1:0}
spi_write_area	f	2984	let addr \${1:0}
spi_init	f	2053	letl d \${1:4}
spi_init	f	3001	letl d \${1:4}
spi_mytest	f	2063	call spi_init \$spi_speed
spi_mytest	f	3011	call spi_init \$spi_speed
spi_memen	f	2073	letl en \${1:1}
spi_memen	f	3021	letl en \${1:1}
spi_sst_wen	f	2082	letl en {\${1:1}?0:0xff}
spi_sst_wen	f	3030	letl en {\${1:1}?0:0xff}
spi_winbond_wen	f	2098	call spi_init \$spi_speed
spi_winbond_wen	f	3046	call spi_init \$spi_speed
spi_erase_all	f	2106	call spi_init \$spi_speed
spi_erase_all	f	3054	call spi_init \$spi_speed

spi_erase_area	f	2119	let addr \${1:0}
spi_erase_area	f	3067	let addr \${1:0}
spi_gd25_diesel	f	2132	call spi_init \$spi_speed
spi_gd25_diesel	f	3080	call spi_init \$spi_speed
spi_cmd	f	2139	spi_set_cs 0
spi_cmd	f	3087	spi_set_cs 0
spi_fifoclear	f	2150	do while {\$(d1q \${spibase+0x1} 1)&1==0}
spi_fifoclear	f	3098	do while {\$(d1q \${spibase+0x1} 1)&1==0}
vxworks0	f	2157	setconfig core.abisize 32
vxworks	f	2164	setconfig core.abisize 32
vxworks1	f	2173	setconfig core.abisize 32
stfillbuffer_disable	f	2180	letl v \${cp0s 6 d4q 16 1}&~0x100}
config_ls2k_xbar	f	2184	m8 0xfffffffffbfe10020 0;m8 0xfffffffffbfe10060
0;m8 0xfffffffffbfe100a0			0xf2
gdb_module_setup	f	2188	>> gdb.cmd echo monitor python pgdorder=0
acpi_suspend	f	2203	letl acpibase 0xffffffffbfff6c000
rtc_wake	f	2218	letl t \${1:10}
wdt_close	f	2224	set_gpio_out 3 0
cachedump	f	2227	setconfig putelf.uncached 2
cachedump1	f	2233	setconfig putelf.uncached 2
lio_config	f	2241	letl width8 speed \${1:0} \${2:0}
lio_cmd	f	2247	do while {\${#>1}
lio_id	f	2253	lio_cmd 0x555 0xaa 0x2aa 0x55 0x555 0x90
pr2_func	f	2258	#lsu-clk 25MHz cfg
pr4_func	f	2331	#lsu-clk 25MHz cfg
lsu_st	f	2437	m4 0xffffffffbf0c0240 0x100
lsu_end	f	2443	m4 0xffffffffbf0c0240 0x1000
jbig_func	f	2448	m4 0xffffffffbf0c201c 0x01
runlinux	f	2457	inputserial "c\b" 0xffffffffbfff40805
0xffffffffbfff40800			
test_pciaccess	f	2466	call cache_init
test_pcidma	f	2472	letl cfg \$(d4q 0x8000000016000110 1)
erase_spi	f	2515	setconfig spi.iobase \$spibase
program_spi	f	2522	letl file \${1:/tmp/gzrom.bin}
mmc_probe	f	2534	letl file \${1:/tmp/gzrom-mmc.bin}
mmc_vmlinux	f	2553	letl file \${1:/tmp/vm.2p.jpeg}
fixup_window	f	2572	m4 0x8000000016000100 \${d4q
0x8000000016000100 1} 0x20}			
i2c_delay	f	2577	#msleep \$1
i2c_gpio_start	f	2585	set_gpio_out \$gpio_d 1
i2c_gpio_send	f	2592	let d \$1
i2c_gpio_rec	f	2605	letl d \$(get_gpio_in \$gpio_d)
i2c_gpio_stop	f	2621	set_gpio_out \$gpio_c 1
i2c_gpio_rack	f	2627	letl d \$(get_gpio_in \$gpio_d)
i2c_gpio_wack	f	2635	set_gpio_out \$gpio_d \$1
i2c_gpio_read	f	2640	letl addr reg \${1:0} \${2:0}
i2c_gpio_write	f	2661	letl addr reg dat \${1:0} \${2:0} \${3:0}
i2c_via_read	f	2679	#i2c_init
i2c_via_read_block	f	2699	#i2c_init
i2c_via_read_byte	f	2720	letl device data \${1:0} \${2:0}
i2c_via_write_byte	f	2733	#i2c_init
_i2c_ls2k_stop	f	2747	letl cr dr \$(expr \$i2creg+0x4) \$(expr
\$i2creg+0x3)			
_i2c_ls2k_read	f	2752	letl cr dr adr count \$(expr \$i2creg+0x4)
\$(expr \$i2creg+0x3) "\$1" "\${2:1}"			

```

_i2c_ls2k_write      f      2765      letl cr dr adr $(expr $i2creg+0x4) $(expr
$i2creg+0x3) ${1:0}
_i2c_ls2k_write      f      2784      _i2c_ls2k_write $*
_i2c_ls2k_read       f      2788      letl cr dr adr reg count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
_i2c_ls2k_scan       f      2798      local r
pci8619_i2c_read     f      2809      letl adr reg port ${3:0x38}<<1} ${1:0x78}
${2:4}
_i2c_ls2k_read2a     f      2816      letl cr dr adr reg count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
_i2c_ls2k_write2a    f      2842      letl cr dr adr reg val $(expr $i2creg+0x4)
$(expr $i2creg+0x3) ${1:0} ${2:0} ${3:0}
_i2c_ls2k_read2      f      2860      letl cr dr adr reg count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
_i2c_ls2k_write2     f      2888      letl cr dr adr reg val $(expr $i2creg+0x4)
$(expr $i2creg+0x3) ${1:0} ${2:0} ${3:0}
dumpserial          f      3118      mems
dumpserial.hb       f      3130      letl saddr ${1:0xfffffffffbfe001e0}
watchprint          f      3150      echo_off
awatchprint         f      3169      echo_off
mywatchset          f      3195      echo_off
mywatch             f      3229      echo_off
mywait              f      3268      do while 1
watchgpu            f      3274      echo_off
rwatchpcicfg        f      3301      echo_off
watchddrparam       f      3324      echo_off
inputserial         f      3346      setconfig log.level 0
waitnmi             f      3378      do while 1
myput               f      3391      #myput serialaddr #use let to set kernel, rd,
kargs to boot kernel
waitserial          f      3404      letl str ${1:PMON>}
#echo_on
cpu0 -
cpu0 -set           #通过set 命令可以查看寄存器的状态
#set

```

```

zero:0x0             ra:0x9000000000e319cc  tp:0x90000000011dc000
sp:0x90000000011dfdc0
a0:0x90000000011fea80 a1:0x9000000001202100  a2:0x1             a3:0xa
a4:0x3               a5:0xfffffffffffffffff  a6:0x8080808080808080 a7:0x16
t0:0x0               t1:0x4             t2:0x14547ee
t3:0xf5c28f
t4:0xfffffffffffffffff t5:0x9ab0b8e2c90    t6:0x90000000011ee290
t7:0x90000000011ee290
t8:0x53c17e1705106d7a u0:0x0             fp:0x200000        s0:0x4
s1:0x0               s2:0x90000000011ee414 s3:0x4
s4:0x90000000012aca5f
s5:0x9000000001088828 s6:0x1             s7:0x900000000eca00a8 s8:0x0
pc:0x9000000002018c0

```

```

csr0-crmd:0xb4        csr1-prmd:0x4          csr2-cu   :0x0
                        csr3-cfg :0x0
csr4-excfg:0x71c1c    csr5-exst:0x800          csr6-epc
:0x90000000002018c0  csr7-badv:0x7ffbe3b0ca
csr8-badi:0x0         csrc-ebase:0x9000000001700000
cpu0 -

```

```

cpu0 -program_cachelock_spi u-boot-with-spl.bin      #烧录固件
#program_cachelock_spi u-boot-with-spl.bin
#spi_memen
#letl en 1
#expr 0x8000000016010000+0x4
#d1q 0x8000000016010004 1
#letl o 0x47
#do if 1
#expr 0x8000000016010000+0x4
#expr 0x47|1
#m1 0x8000000016010004 0x47
#loop_break
#ret
#letl file u-boot-with-spl.bin
#echo_on
#stop
#setconfig usb_ejtag.put_speed 0x400
#setconfig callbin.stacksize 0x1000
#setconfig helpaddr 0x900000001c000000
#setconfig put.pack_size 0x10000
#call cache_init_and_lock
#call cache_init
#setconfig core.nocache 0
#setconfig cacheflush.nohelp_size 0x1000000
#set csr0 0xb0
#set csr180 0x800000000000000f
#set csr181 0x900000000000001f
#setconfig core.nocache 1
#ret
#call fixup_window
#d4q 0x8000000016000100 1
#expr 0x18020010|0x20
#m4 0x8000000016000100 0x18020030
#m8 0x8000000016000090 0x000000001c0000f2 0x000000001c0000f2
#ret
#call cache_lock
#m8 0x8000000016000200 0x800000001c000000
#m8 0x8000000016000240 0xffffffffffffc0000
#ret
#memset8 0x900000001c000000 0 0x800
...

#do while 0x1#put u-boot-with-spl.bin 0x900000001c010000 0x20000 0xc0000
pack:1,time :5,dmload size :0x1d332,dmload rate=23920 B/S
#setconfig log.disas1
#log /tmp/log-$i.txt
#test program cachelock spi ==program cachelock bin
#do if 0
#Sprogram 0x900000001c0100000xc0000
1196020x00000000000c0000(786432)0x0000000000d0000(851968)
#end
##letl s s
##setconfig log.disas 0
##log
#end
#expr0xc0000+0x20000

```

```
#letl i 0xe0000
#expr 0xe0000<906034
#do while 0x0
#setconfig helpaddr 0x900000000000f000
#setconfig put.pack size 0x100000
#ret
cpu0-
```

二、搭建交叉编译环境

系统环境要求: ubuntu 22.04 或以下的版本, 推荐使用 **ubuntu 18.04**。

从网盘下载交叉工具链解压到/opt 目录

```
sudo tar xf loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu-rc1.3-1.tar.xz -C /opt
```

或者从 <http://www.loongnix.cn/zh/toolchain/GNU/> 下载, 选择 LoongArch GCC 交叉工具链, 下面的xxx 为具体版本号

```
sudo tar xf loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu-xxx.tar.xz -C /opt
```

建立软链接 (方便统一设置环境变量)

```
sudo cd /opt
ln -s loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu-xxx loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu
```

源码目录里面有设置环境变量的脚本 `set_env.sh`, 内容如下:

```
function setup_loongarch_env()
{
    echo "====>setup env for LoongArch..."
    CC_PREFIX=/opt/loongson-gnu-toolchain-x86_64-loongarch64-linux-gnu
    export PATH=$CC_PREFIX/bin:$PATH
    export LD_LIBRARY_PATH=$CC_PREFIX/lib:$LD_LIBRARY_PATH
    export LD_LIBRARY_PATH=$CC_PREFIX/loongarch64-linux-gnu/lib64:$LD_LIBRARY_PATH

    export ARCH=loongarch
    export CROSS_COMPILE=loongarch64-linux-gnu-
}

setup_loongarch_env
```

三、u-boot 编译

源码可从网盘或<https://gitee.com/open-loongarch>下载

1.设置交叉工具链等环境

```
$ source ./set_env.sh
====>setup env for LoongArch...
```

2.指定板卡配置

```
$ make loongson_2k300_pai_defconfig
make[1]: Entering directory '/home/vm/u-boot-2022.04'
  GEN      Makefile
  HOSTCC   scripts/kconfig/conf.o
  YACC     scripts/kconfig/zconf.tab.c
  LEX      scripts/kconfig/zconf.lex.c
  HOSTCC   scripts/kconfig/zconf.tab.o
  HOSTLD   scripts/kconfig/conf
#
# configuration written to .config
#
make[1]: Leaving directory '/home/vm/u-boot-2022.04'
```

3.开始编译

```
$ make
make[1]: Entering directory '/home/vm/u-boot-2022.04'
  GEN      Makefile
scripts/kconfig/conf  --syncconfig kconfig
  CFG      u-boot.cfg
  GEN      include/autoconf.mk.dep
  CFG      spl/u-boot.cfg
  GEN      include/autoconf.mk
  GEN      spl/include/autoconf.mk
  GEN      Makefile
  CFGCHK   u-boot.cfg
...
  CC       spl/lib/rtc-lib.o
  CC       spl/lib/elf.o
  AR       spl/lib/built-in.o
  LD       spl/u-boot-spl
  OBJCOPY  spl/u-boot-spl-nodtb.bin
  SYM      spl/u-boot-spl.sym
  CAT      spl/u-boot-spl-dtb.bin
  COPY     spl/u-boot-spl.bin
  CAT      u-boot-with-spl.bin
make[1]: Leaving directory '/home/vm/u-boot-2022.04'
```

编译成功之后会生成 `u-boot-with-spl.bin`

四、kernel 编译

源码可从网盘或<https://gitee.com/open-loongarch>下载

注：编译前先更新mkimage 工具，否则编译uImage 的时候可能会遇到 Invalid architecture 的错误。

mkimage更新方法

在编译后的u-boot源码目录里面执行命令：`sudo cp tools/mkimage /usr/bin/`，如果u-boot 源码没有编译，默认是无mkimage工具的。

1.设置交叉工具链等环境

```
$ source ./set_env.sh
====>setup env for LoongArch...
```

2.指定板卡配置

```
$ make loongson_2k300_defconfig
make[1]: Entering directory '/home/vm/kernel-5.10-LoongArch'
  GEN      Makefile
#
# No change to .config
#
make[1]: Leaving directory '/home/vm/kernel-5.10-LoongArch'
```

3.开始编译

```
$ make uImage
  GEN      .version
  CHK      include/generated/compile.h
  UPD      include/generated/compile.h
  CC      init/version.o
  AR      init/built-in.a
  LD      vmlinux.o
  MODPOST vmlinux.symvers
  MODINFO modules.builtin.modinfo
  GEN      modules.builtin
  LD      .tmp_vmlinux.kallsyms1
  KSYMS   .tmp_vmlinux.kallsyms1.s
  AS      .tmp_vmlinux.kallsyms1.s
  LD      .tmp_vmlinux.kallsyms2
  KSYMS   .tmp_vmlinux.kallsyms2.s
  AS      .tmp_vmlinux.kallsyms2.s
  LD      vmlinux
  SORTTAB vmlinux
  SYMAP    System.map
  OBJCOPY arch/loongarch/boot/vmlinux.bin
  GZIP     arch/loongarch/boot/vmlinux.bin.gz
  UIMAGE  arch/loongarch/boot/uImage.gz
Image Name:   Linux-5.10.0.lsgd-g217c34b2e4e3
Created:      Sat Sep 17 14:20:28 2022
Image Type:   LoongArch Linux Kernel Image (gzip compressed)
Data Size:    7824190 Bytes = 7640.81 KiB = 7.46 MiB
```

```
Load Address: 00200000
Entry Point: 00ca1a44
Image arch/loongarch/boot/uImage is ready
make[1]: Leaving directory '/home/vm/work/kernel-5.10-LoongArch'
```

编译成功之后会在 `arch/loongarch/boot/` 生成 `uImage` 文件。

五、buildroot 编译

源码可从网盘或<https://gitee.com/open-loongarch>下载

1.指定板卡配置

```
make loongson2k300_mini_dp_defconfig
```

2.编译

```
make
```

编译成功之后在 `output/images` 下会生成 `rootfs.img` 文件，将其即可烧录。

六、第三方开源软件编译

为了简化可移植构建的难度，早期有一套AutoTools的工具帮助程序员构建软件。c/c++开源项目大多由Autotools 工具管理，源码编译常见三步曲。

```
./configure
make
make install
```

在LoongArch 下编译常见的问题如下：

```
./configure
configure: error: cannot guess build type; you must specify one
```

在 `./configure` 阶段，Autotools 会扫描宿主系统（即当前正在运行的计算机）以进行各种测试并完成默认设置。

在测试过程中，有一项是系统架构测试，当在陌生的新架构中运行时，`./configure` 将报错并退出，导致无法完成软件编译。

其中最关键的2个架构相关文件为：`config.guess` 和 `config.sub`。

GNU Autotools 套件中 `config.guess` 和 `config.sub`，被独立维护在为 `config.git` 仓库。

其他软件如gnulib、automake将不定期从这里获取，并合并入自己的代码树中，最终上游开发者通过GNU Autotools打包后，这2个文件将进入tarballs中被发布。

LoongArch架构相关的补丁，于2020年12月22日被合并到 `config.git` 主分支。

<https://git.savannah.gnu.org/gitweb/?p=config.git;a=commitdiff;h=c8ddc8472f8efcadafc1ef53ca1d863415fddd5f>

最新的config.git 已合并了对Loongarch 架构的支持。而对早期的config.git 没有添加 Loongarch 架构支持的，需要自己手动添加。

方法一：编辑 config.guess 文件，在ia64:Linux:~) 下面添加Loongarch 架构的支持。

```
loongarch*:Linux:~*)
    echo ${UNAME_MACHINE}-unknown-linux-gnu
    exit ;;
```

方法二：从网上获取最新的config.guess 和 config.sub 文件然后进行替换

```
$ sudo wget -O /usr/share/misc/config.sub "git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD"

$ sudo wget -O /usr/share/misc/config.guess "git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD"
```

在源码目录执行下面的命令会 /usr/share/misc/config.sub 和 config.guess 替换掉工程目录下面的 config.sub 和 config.guess 文件

```
$ libtoolize -f -i -c
```

c/c++ 代码编译选项、编译宏移植

功能	x86编译选项	龙芯（LoongArch）编译选项
定义编译生成的程序为64 位程序	-m64	-mabi=lp64d
定义指令集类型,由X86修改为LoongArch	-march=skylake	-march=loongarch64
原有x86版编译宏替换为LoongArch宏	__X86_64__	__loongarch__
查看编译器自定义编译宏	echo gcc -E -dM -	

更多参考：[交叉编译成LoongArch\(Makefile,CMake,AutoTool,Qt等方式\)](https://blog.csdn.net/mxcai2005/article/details/130977477?spm=1001.2014.3001.5502)
<https://blog.csdn.net/mxcai2005/article/details/130977477?spm=1001.2014.3001.5502>

七、Python 编程

默认的buildroot文件系统里面集成了python3 环境和常用的python包。

内置的python 包列表如下：

```
[root@LS-GD ~]# pip list
Package            Version
-----
aenum              2.2.3
canopen            1.0.0
cbor2              5.2.0
chardet            4.0.0
Click              7.0
DataProperty       0.48.1
Flask              1.1.2
itsdangerous       1.1.0
Jinja2             2.11.3
luma.core          2.4.2
luma.oled          3.13.0
MarkupSafe         1.1.1
mbstrdecoder       1.0.1
modbus-tk          1.1.2
pathvalidate       0.29.0
Pillow             9.4.0
pip               20.0.2
pyftdi             0.55.4
pygame            1.9.2a0
pymodbus           2.2.0
PyMySQL           0.9.3
PyQRCode           1.2.1
PyQt5             5.15.6
pyserial           3.5
python-can         3.3.4
python-dateutil    2.8.1
python-engineio    3.12.1
python-socketio    4.3.1
pytz               2019.3
pyusb              1.0.0
RPi.GPIO           0.7.0
setuptools         44.0.0
SimpleSQLite       0.45.2
six                1.15.0
smbus              1.1
smbus2             0.4.3
spidev             3.5
sqliteschema       0.16.2
tabledata          1.1.2
typepy             0.0.20
urllib3            1.26.6
w1thermsensor      2.3.0
werkzeug           1.0.1
wrapit             1.11.2
```

里面包含有适配好的 `RPi.GPIO`, `luma.core`, `luma.oled`, `w1thermsensor` 等包。

```
[root@LS-GD ~]# pip list | grep -i rpi
RPi.GPIO          0.7.0
[root@LS-GD ~]# pip list | grep -i luma
luma.core          2.4.2
luma.oled          3.13.0
[root@LS-GD ~]# pip list | grep -i sensor
withersensor      2.3.0
[root@LS-GD ~]# python3 --version
Python 3.9.6
[root@LS-GD ~]#
```

如果使用 `RPI.GPIO` 等包，扩展针上的 `GPIO` 引脚号和功能定义请查看原理图与芯片手册中的引脚复用关系，`GPIO` 号默认从 0 开始。

运行python 程序

```
python3 leds.py
```

八、FAQ

1.如何设置开机启动

板卡自带的buildroot 系统里面，开机会运行 `/root/boot_run.sh`，所以如果需要设置开机启动，把要启动的程序加入到 `boot_run.sh` 中即可。

板卡开机默认启动的Qt程序也是在里面设置的，如果想禁用此程序修改 `boot_run.sh` 即可。

`boot_run.sh` 源码如下：

```
# cat boot_run.sh
#!/bin/sh

#you can custom your action after boot in this script
#if you want to run your application after boot

psplash-write "MSG Welcome to Loongson-gd"
psplash-write "PROGRESS 100"
psplash-write "QUIT"
if [ -z "$(cat /proc/cmdline | grep ubifs)" ]; then
    sleep 5
fi
export QT_QPA_FB_TSLIB=1

cd /root/logo_player && ./logo_player &
```

2.如何设置网络

板卡自带的buildroot 系统里面，网络由 `NetworkManager` 管理。默认配置的IP是 `192.168.1.10`

默认添加的网络配置如下：

```
# nmcli c show
NAME                UUID                                TYPE      DEVICE
eth0-connection    1bbd9bed-6870-4098-a4f3-06a5bb11ad3d  ethernet  eth0
```

```

[root@LS-GD ~]# cd /etc/NetworkManager/system-connections
[root@LS-GD system-connections]# ls
eth0-connection.nmconnection
[root@LS-GD system-connections]# cat eth0-connection.nmconnection
[connection]
id=eth0-connection
uuid=1bbd9bed-6870-4098-a4f3-06a5bb11ad3d
type=ethernet
interface-name=eth0
permissions=

[ethernet]
mac-address-blacklist=

[ipv4]
address1=192.168.1.10/24
dns-search=
method=manual

[ipv6]
addr-gen-mode=stable-privacy
dns-search=
method=auto

[proxy]

```

如果想修改IP, 可以运行 `nmtui` 进行修改, 也可以直接编辑 `/etc/NetworkManager/system-connections/eth0-connection.nmconnection`, 也可运行 `systemctl stop NetworkManager` 命令关闭 `NetworkManager` 服务, 然后通过 `ifconfig` 或 `ip` 命令进行修改。

3.如何查看uart 与tty 的对应关系

方法1:

```

# cd /sys/class/tty/
# ls -l tty*
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS0 -> ../../devices/platform/2k300-soc/16100000.serial/tty/ttyS0
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS1 -> ../../devices/platform/2k300-soc/16100400.serial/tty/ttyS1
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS10 -> ../../devices/platform/serial8250/tty/ttyS10
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS11 -> ../../devices/platform/serial8250/tty/ttyS11
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS12 -> ../../devices/platform/serial8250/tty/ttyS12
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS13 -> ../../devices/platform/serial8250/tty/ttyS13
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS14 -> ../../devices/platform/serial8250/tty/ttyS14

```

```
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS15 ->
../../devices/platform/serial8250/tty/ttyS15
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS2 -> ../../devices/platform/2k300-
soc/16100800.serial/tty/ttyS2
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS3 -> ../../devices/platform/2k300-
soc/16100c00.serial/tty/ttyS3
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS4 ->
../../devices/platform/serial8250/tty/ttyS4
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS5 ->
../../devices/platform/serial8250/tty/ttyS5
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS6 ->
../../devices/platform/serial8250/tty/ttyS6
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS7 ->
../../devices/platform/serial8250/tty/ttyS7
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS8 ->
../../devices/platform/serial8250/tty/ttyS8
lrwxrwxrwx 1 root root 0 Feb  3 23:21 ttyS9 ->
../../devices/platform/serial8250/tty/ttyS9
```

方法2:

```
# cd /sys/devices/platform/2k300-soc
# find . -name "ttyS*"
./16100800.serial/tty/ttyS2
./16100c00.serial/tty/ttyS3
./16100400.serial/tty/ttyS1
./16100000.serial/tty/ttyS0
```

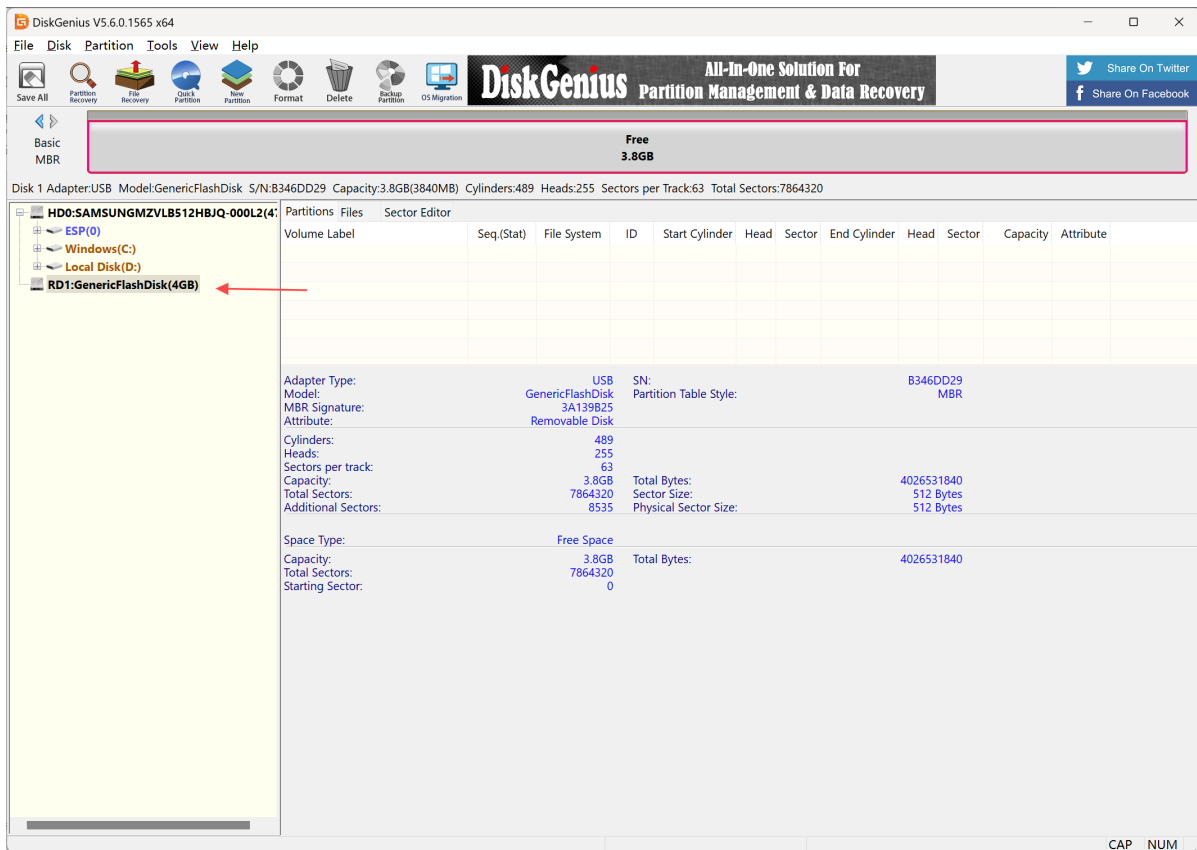
得到的结果中【数字.serail】部分中的数字为uart 的寄存器地址，末尾为tty的名字。查看芯片手册的寄存器描述或u-boot, kernel 中的dts 描述，便可知对应关系。

4.如何格式化U盘

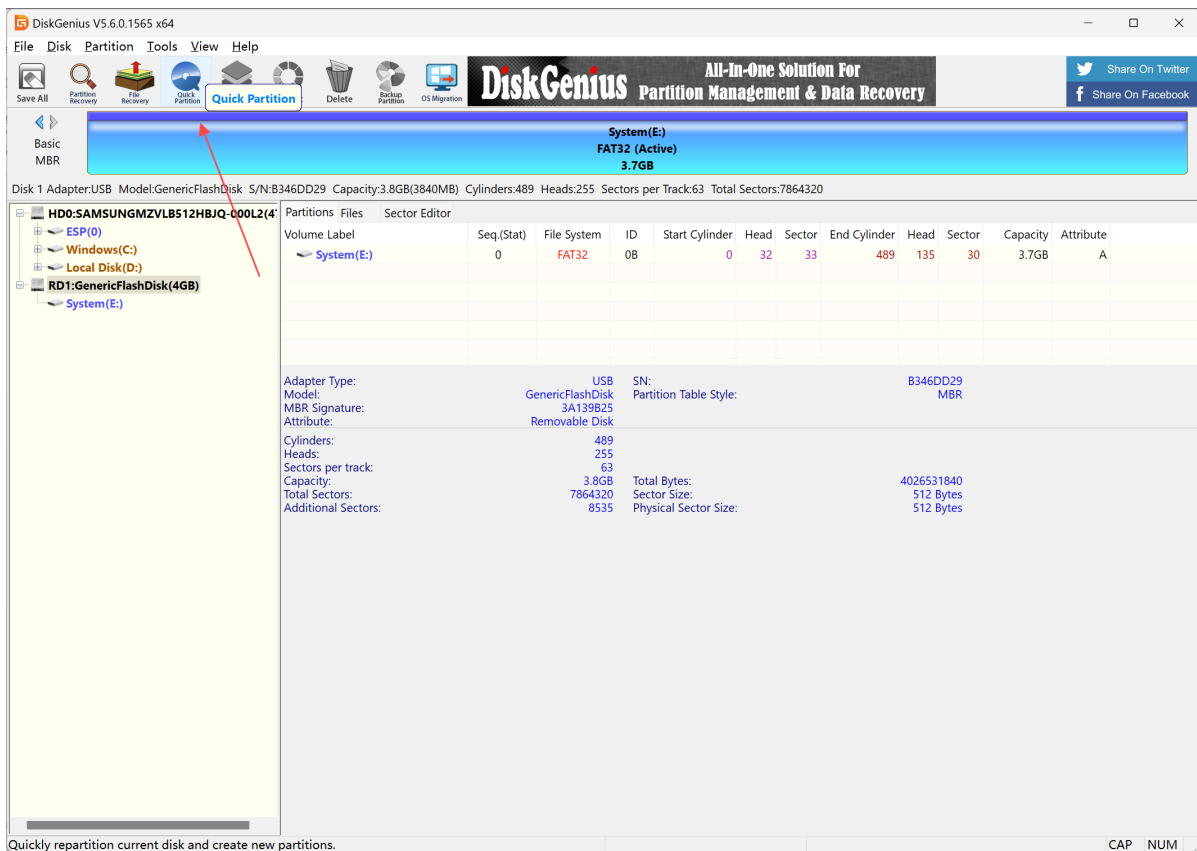
a.Windows 格式化U盘

推荐使用**DiskGenius**避免分区表错误。**注意，请事先备份好U盘的数据。**演示请看下图：

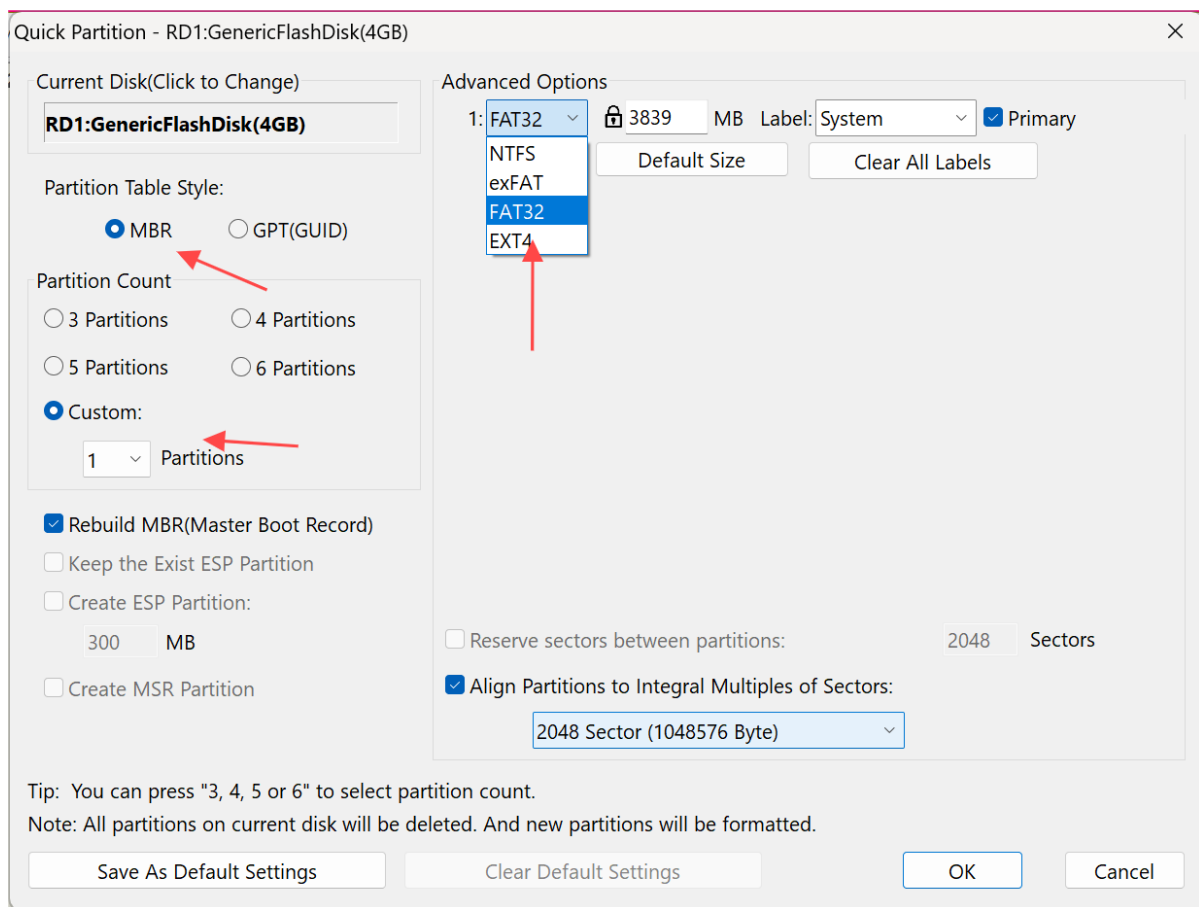
插入U盘，打开DiskGenius，选中U盘设备



点击“Quick Partition”



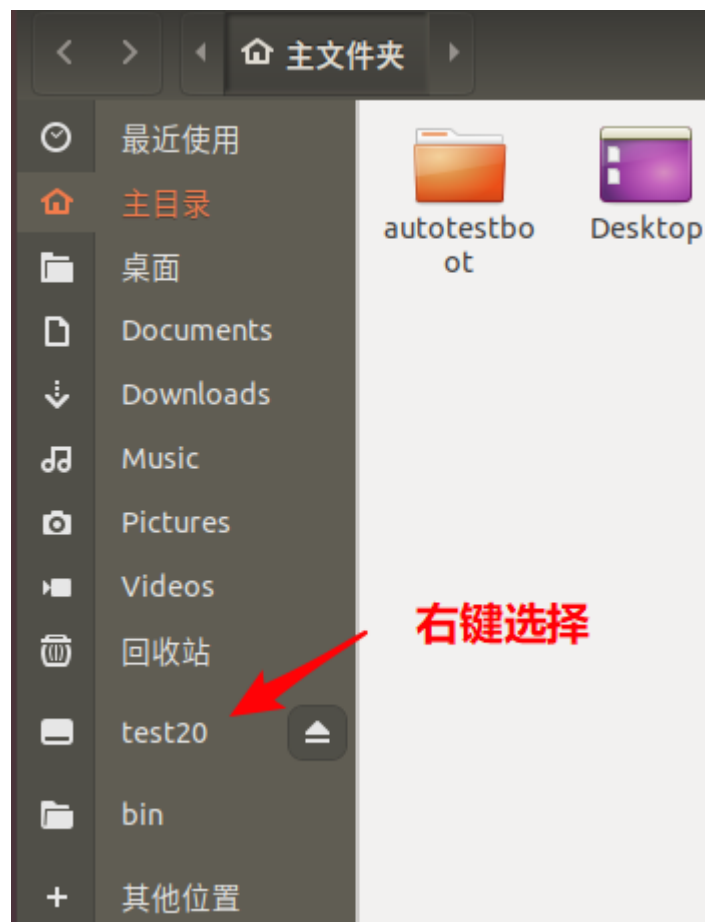
依次选MBR分区表、创建一个分区，格式fat32，Lable 随便填。点击“OK”开始分区



b.Linux 格式化U盘

推荐使用ubuntu18.04（也就是推荐用于交叉编译开发的文件系统）的自带U盘格式化工具。注意，请事先备份好U盘的数据。演示请看下图：

PC机插入U盘后，打开任意文件夹。





然后点击“下一个”按钮。



点击“格式化”按钮，即可等待完成格式化。



九、2K0300先锋派开发板资料

更多资料从网盘下载

网盘链接: <https://pan.baidu.com/s/1l1-X7BOmLP0jimjz4qHeg?pwd=1234> 提取码: 1234

龙芯技术论坛: https://bbs.elecfans.com/group_1650 已有不少试用报告, 欢迎前去查看和灌水。

