IDO-EVB2708-V1 Linux系统使用手册

IDO-EVB2708-V1

Linux系统使用手册

深圳触觉智能科技有限公司

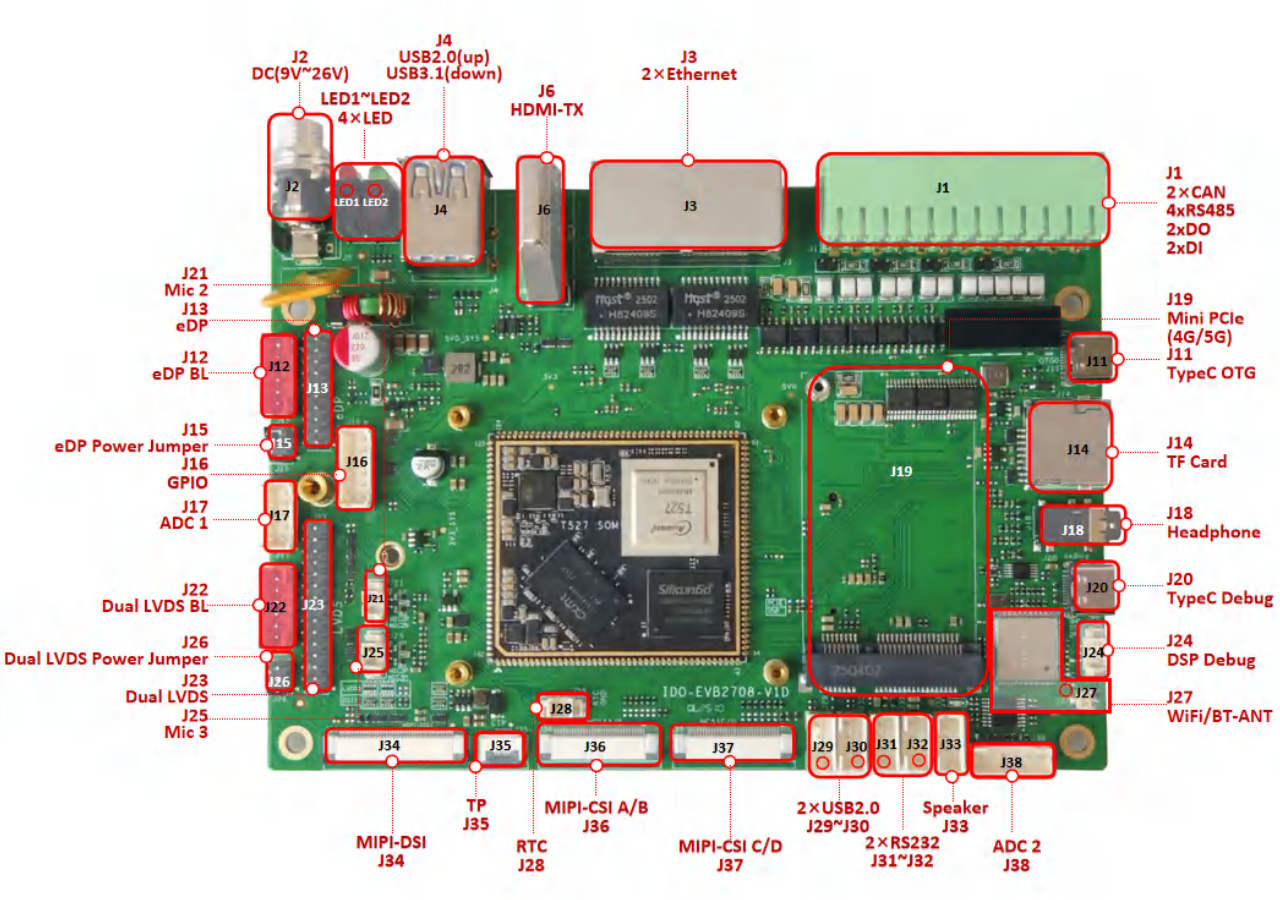
www.industio.cn

文档修订历史

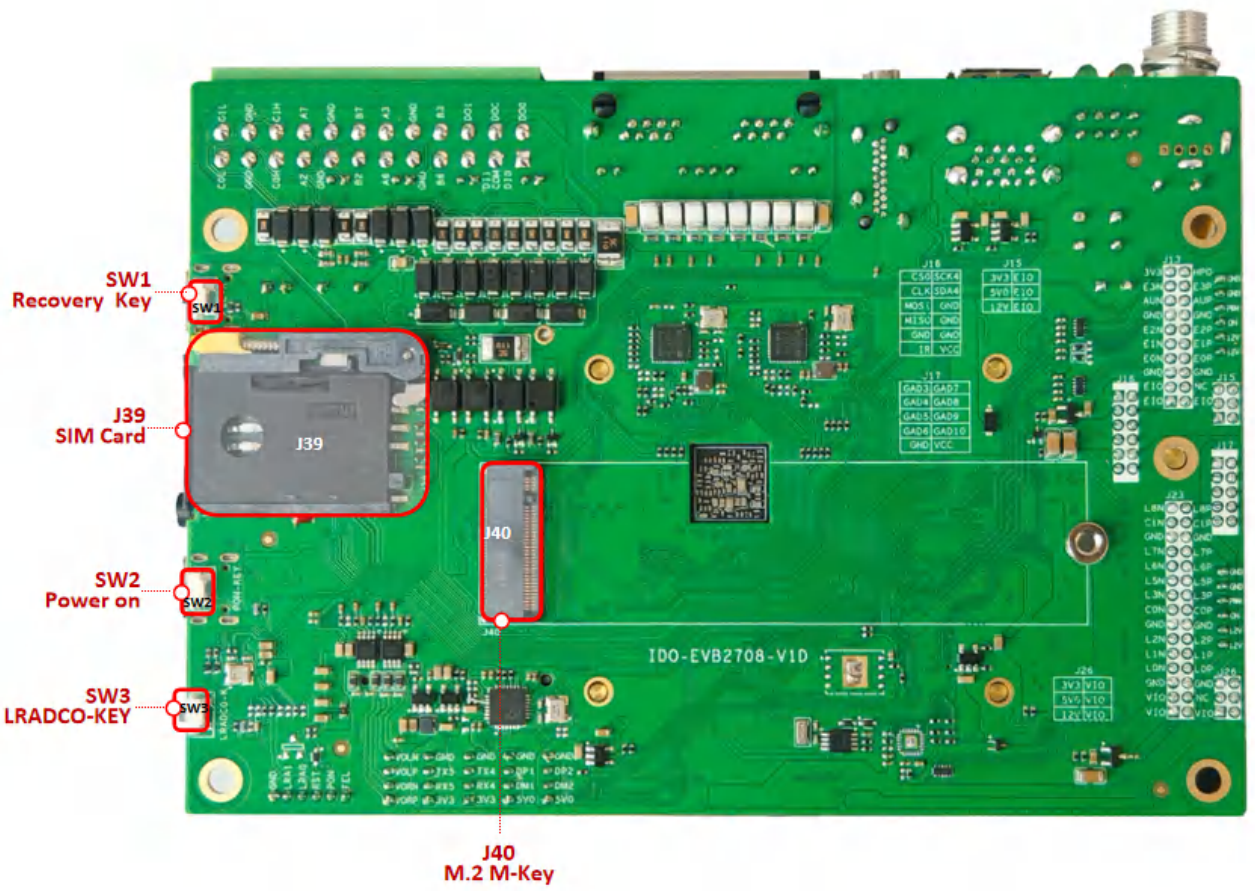
版本	PCBA版本	修订内容	修订	审核
V1.0	V1D	创建文档	LJH	IDO

1、硬件资源概况

1.1 主板照片



IDO-EVB2708-V1 正面实物图



IDO-EVB2708-V1 背面实物图

1.2 硬件资源

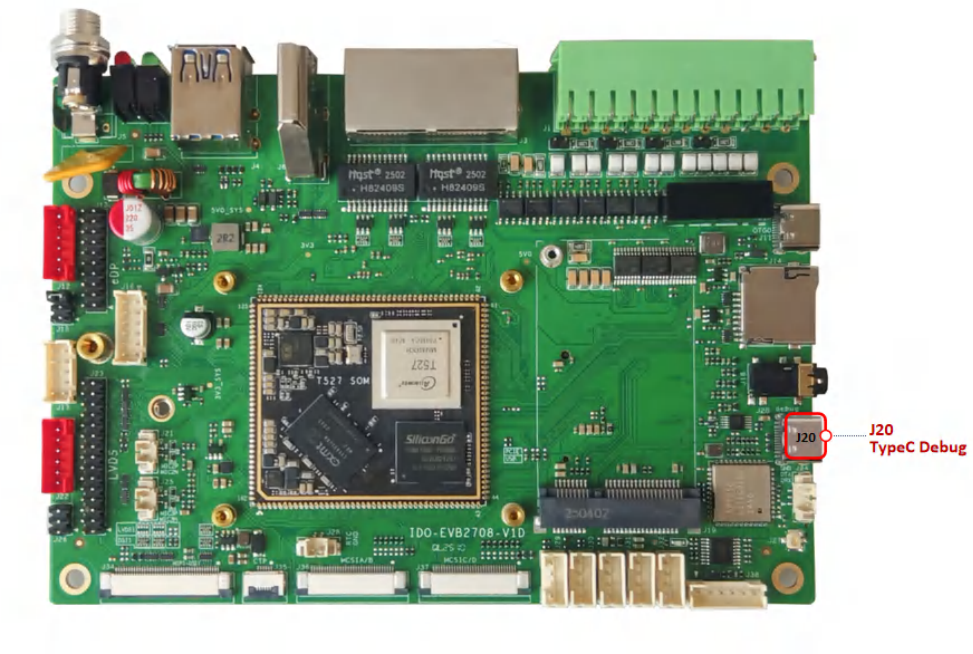
序号	名称	描述
1	内核版本	Linux 5.15.147
2	系统版本	Debian
3	内存	LPDDR4（2G/4G/8GB选配）
4	存储	eMMC5.1（64GB / 128GB）
5	供电	DC接口12V@2A

6	显示	HDMI LVDS Dual-LVDS eDP MIPI
7	USB OTG	USB OTG Type-C
8	USB HOST	USB2.0 HOST(Type-A) X 2 USB2.0 HOST(PH2.0) X 2
9	TF Card	TF Card x 1
10	以太网	千兆以太网 x 1
11	WIFI/BT	AP6256
12	扬声器	4Ω 10W
13	耳机	3.5mm 美标
14	Camera	OV13850
15	串口	RS232 x 2 RS485 x 4
16	调试串口	TTL x 2
17	RTC	HYM8563 x 1
18	系统指示灯	x3
19	4G/5G	1路支持USB2.0 和USB2.0 MIPI PCIE 接口 EC20和RG200U模块
20	POWER ON	x1

2、功能测试及接口使用方法

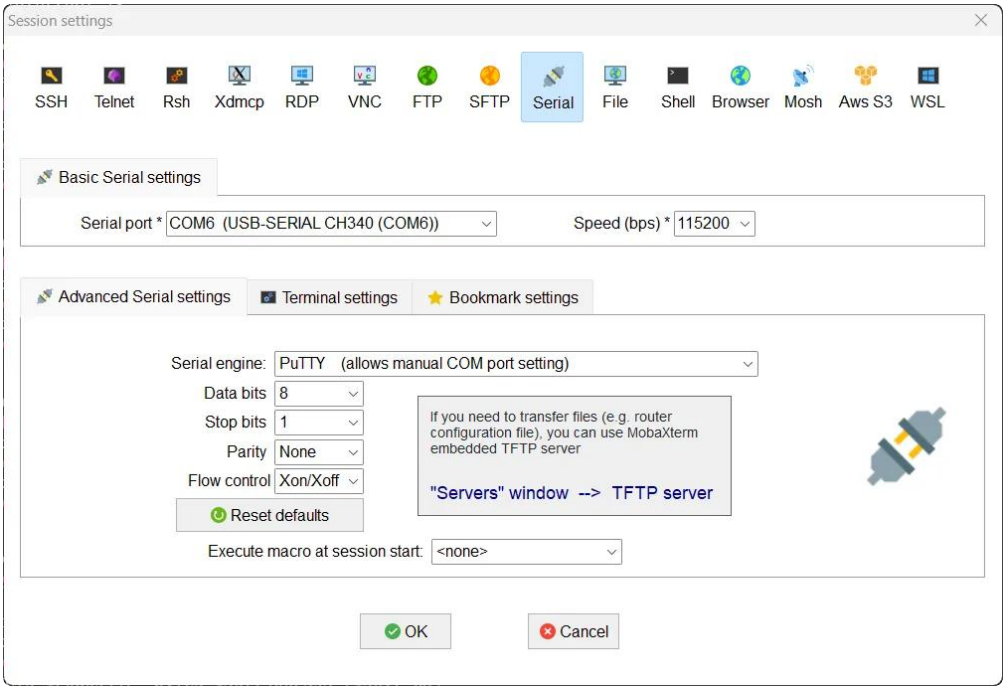
2.1 调试

2.1.1 串口调试



调试串口位置J20如上图所示，使用USB Type-C数据线，连接PC端的USB接口。系统会识别到一个“USB-SERIAL CH340”端口设备。

使用调试软件（MobaXterm、putty），以MobaXterm为例子，设置参数如下：

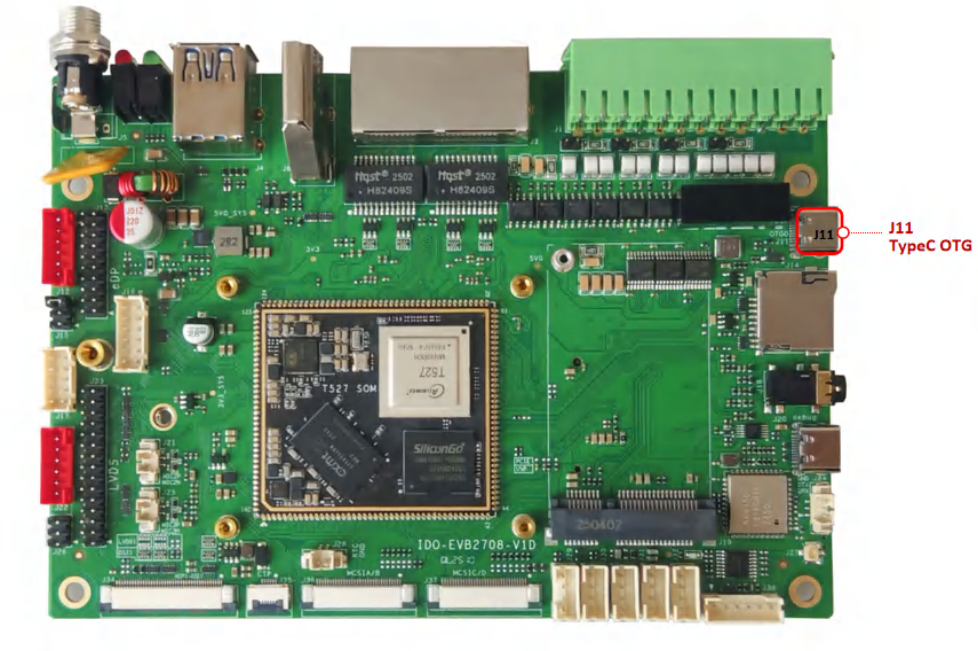


串口调试默认登录账号密码为：industio/123456

root用户密码为123456。

2.1.2 adb调试

主板adb接口位于J11处，与烧录接口为同一接口，如下图所示：

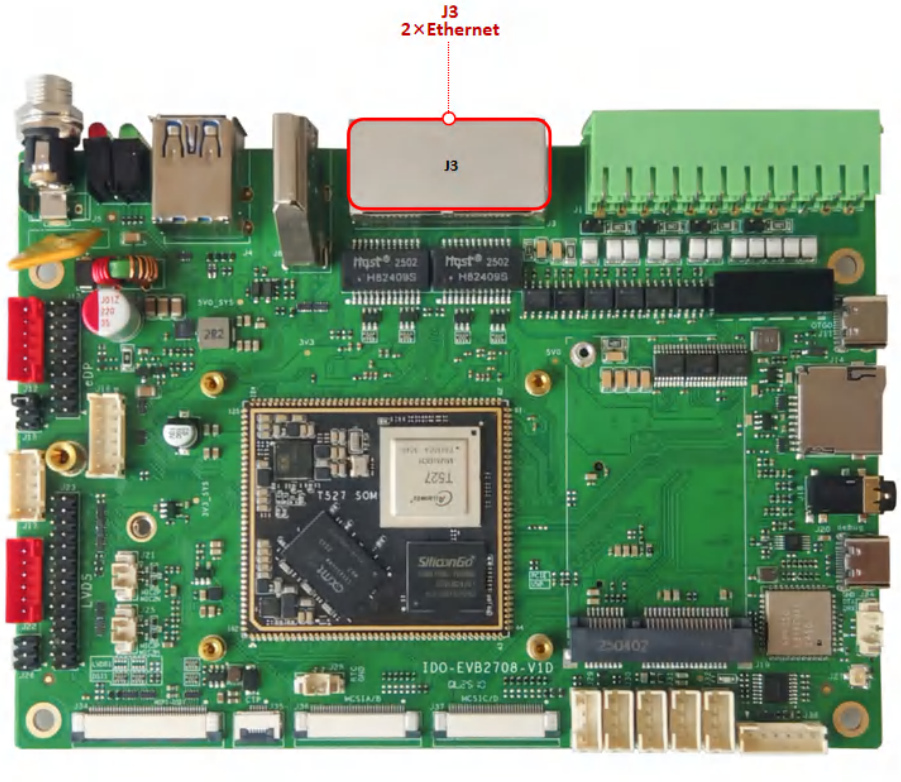


2.1.3 ssh调试

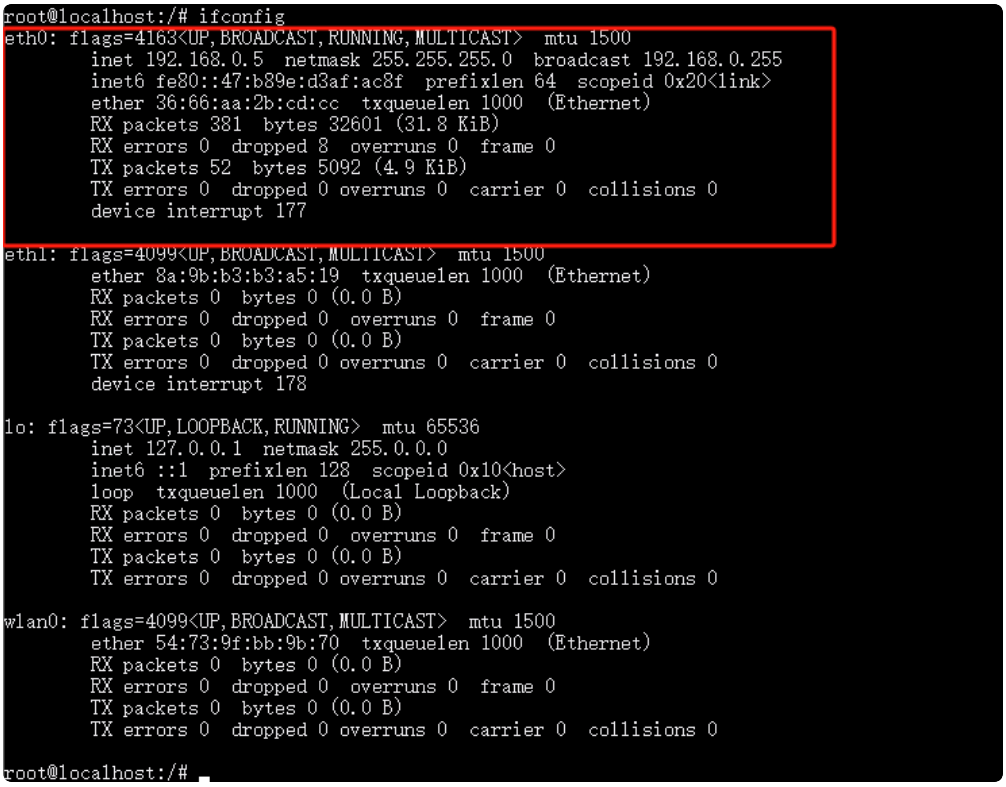
主板支持ssh调试，默认登录账号密码为：

1. 账号：industio
2. 密码：123456
3. 端口：22

2.2 以太网



主板有两路千兆以太网接口，设备节点分别为eth0和eth1(上图红色标注处，左为eth0，右为eth1)，以太网接口默认支持DHCP，只需要将以太网接口连接路由器即可为主板动态分配 IP 地址。如下图所示即为成功分配到ip



#问题#增加网络管理器用法

启动关闭指定网卡 Shell

```
$ ifconfig eth0 down
$ ifconfig eth0 up
```

为网卡配置和删除IPv6地址 Shell

```
$ ifconfig eth0 add 33ffe:3240:800:1005::2/ 64 //为网卡设置IPv6地址
$ ifconfig eth0 del 33ffe:3240:800:1005::2/ 64 //为网卡删除IPv6地址
```

用ifconfig修改MAC地址 Shell

```
$ ifconfig eth0 down //关闭网卡
$ ifconfig eth0 hw ether 00:AA:BB:CC:DD:EE //修改MAC地址
$ ifconfig eth0 up //启动网卡
$ ifconfig eth1 hw ether 00:1D:1C:1D:1E //关闭网卡并修改MAC地址
$ ifconfig eth1 up //启动网卡
```

配置IP地址 Shell

```
$ ifconfig eth0 192.168.1.56
//给eth0网卡配置IP地址
$ ifconfig eth0 192.168.1.56 netmask 255.255.255.0
// 给eth0网卡配置IP地址,并加上子掩码
$ ifconfig eth0 192.168.1.56 netmask 255.255.255.0 broadcast 192.168.1.255
// 给eth0网卡配置IP地址,加上子掩码,加上个广播地址
```

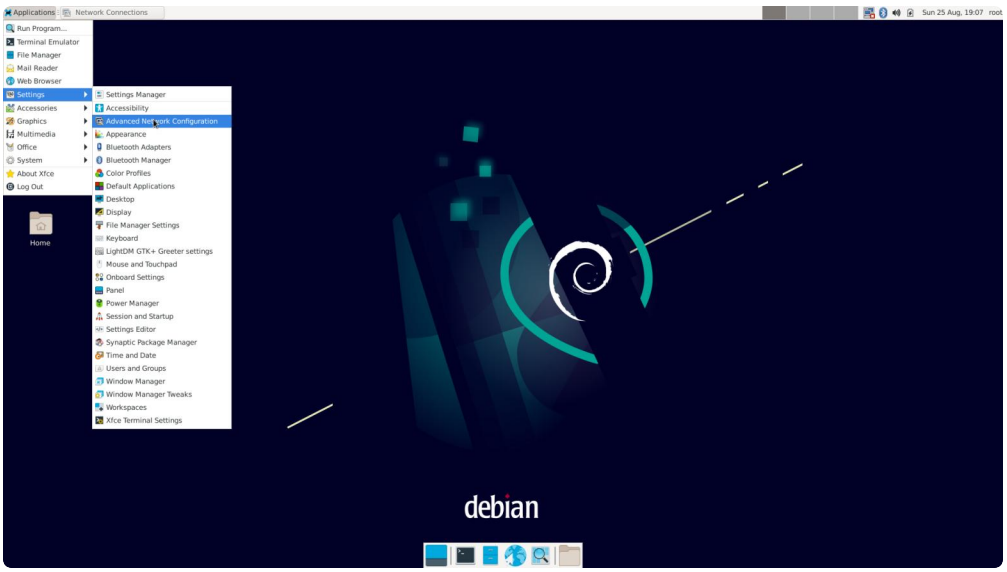
启用和关闭ARP协议 Shell

```
$ ifconfig eth0 arp  //开启
$ ifconfig eth0 -arp  //关闭
```

设置最大传输单元 Shell

```
$ ifconfig eth0 mtu 1500
//设置能通过的最大数据包大小为 1500 bytes
```

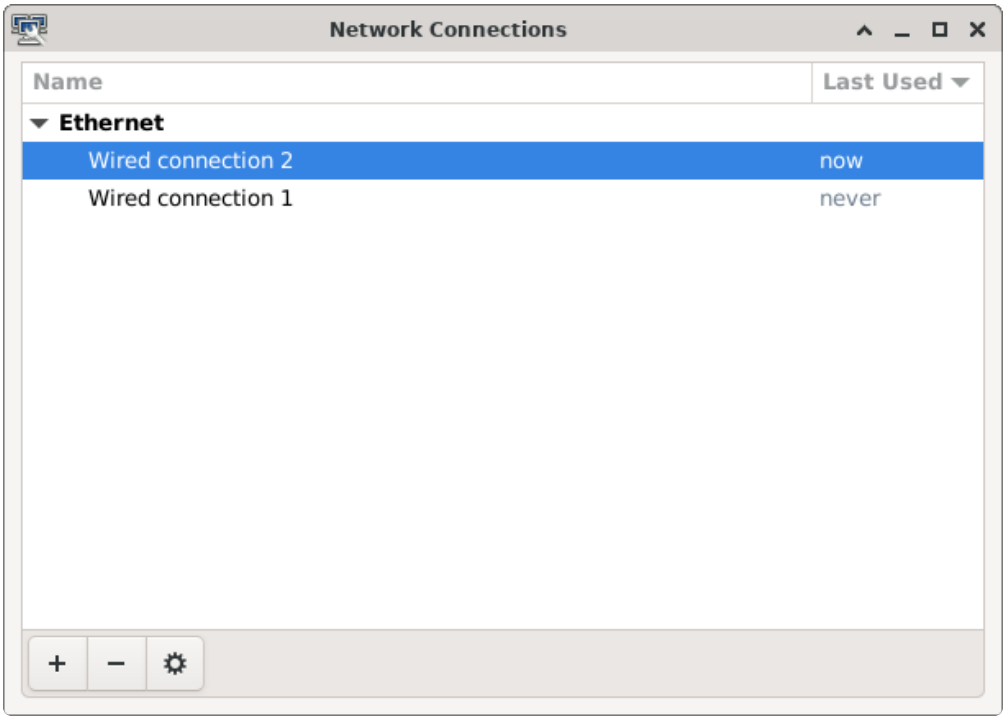
通过桌面管理器配置固定IP



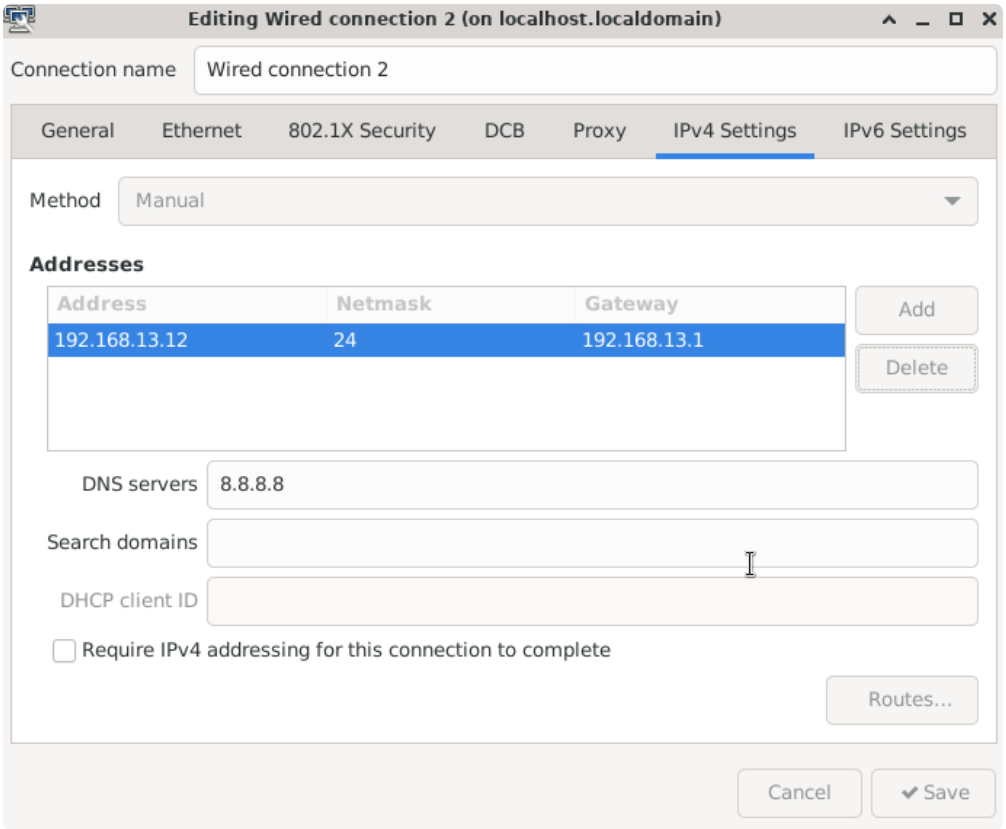
打开 Applications –>

Setting –>

Advanced Network Configuration;



以 `Wired connection 2` 节点为例，点击 `Wired connection 2` 节点进入配置界面。

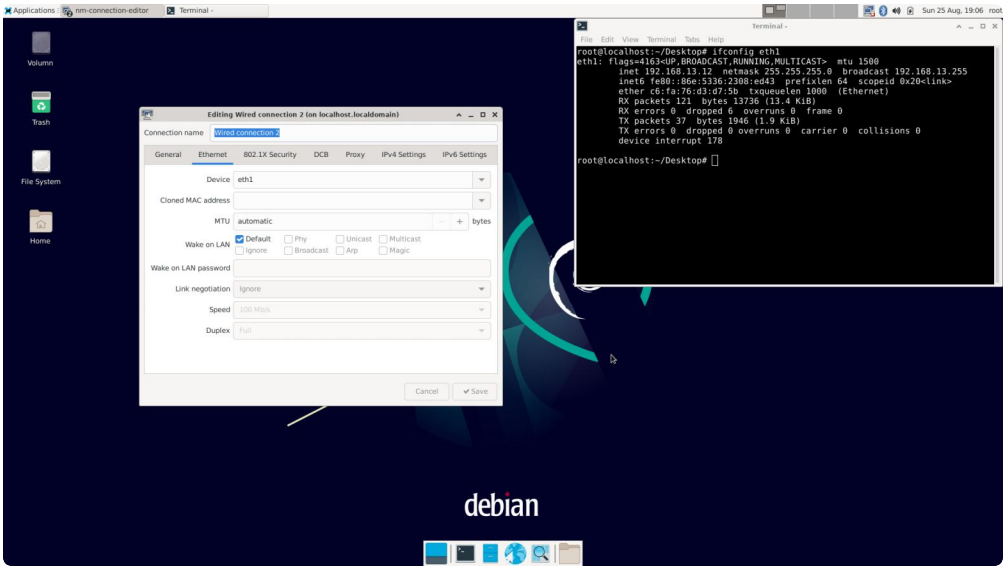


在 `IPv4 Setting` 中，这里以将ip配置为 `192.168.13.12` 为例。

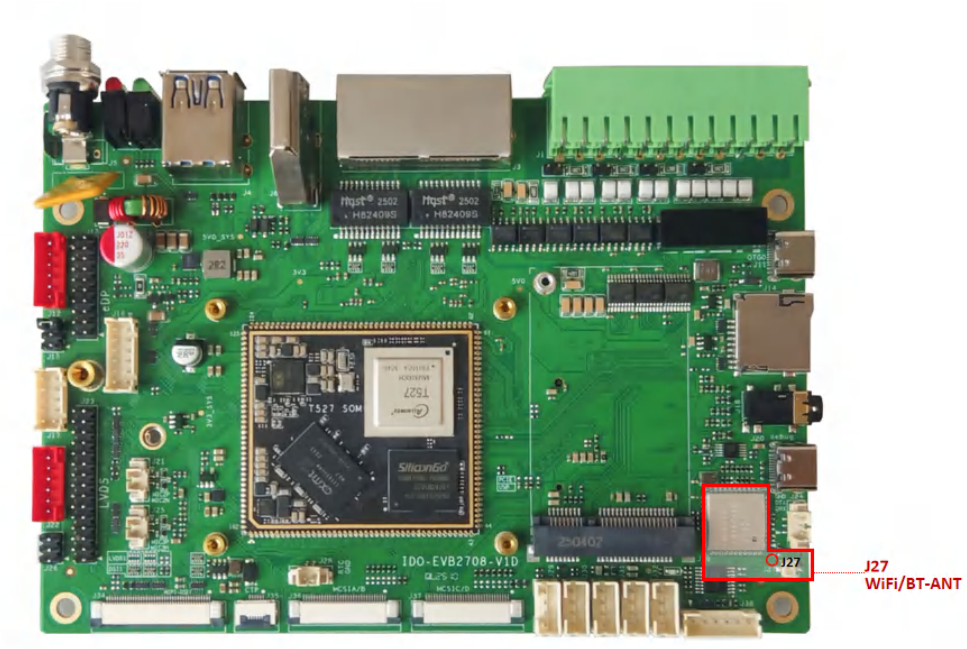
Method选择 `Manual`

点击Addresses右侧的Add按钮添加配置，分辨填入IP地址，子网掩码。

点击底部 `Save` ，接上网线重启开发板。

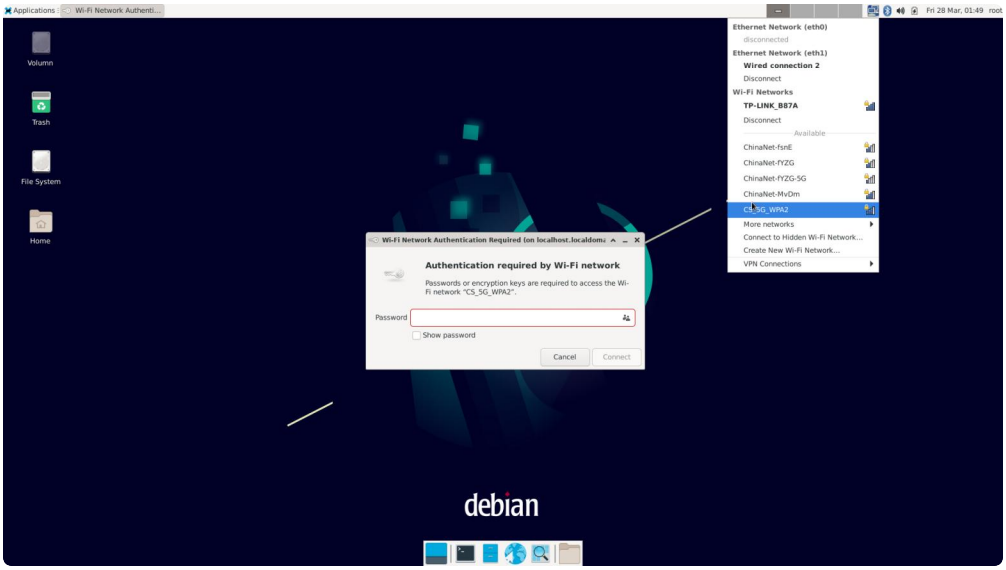


2.3 WiFi



使用WiFi/蓝牙时，需要连接天线以获得良好的信号，上图WiFi/蓝牙模块，J27为天线接口

通过桌面应用连接WiFi设备：



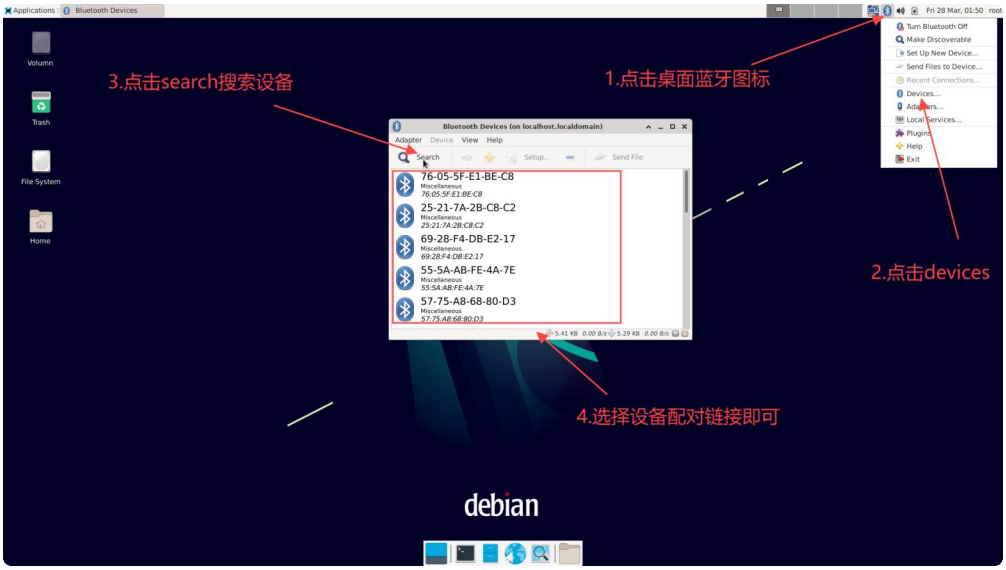
通过命令行连接WiFi设备：

Shell

```
$ wpa_cli -i wlan0 add_network
// 输入wifi账号
$ wpa_cli -i wlan0 set_network 0 ssid '"TP-LINK_B87A"'
// 输入WiFi密码
$ wpa_cli -i wlan0 set_network 0 psk '"12345678"'
// 连接WiFi
$ wpa_cli -i wlan0 select_network 0
```

2.4 Bluetooth

通过桌面应用链接蓝牙设备：

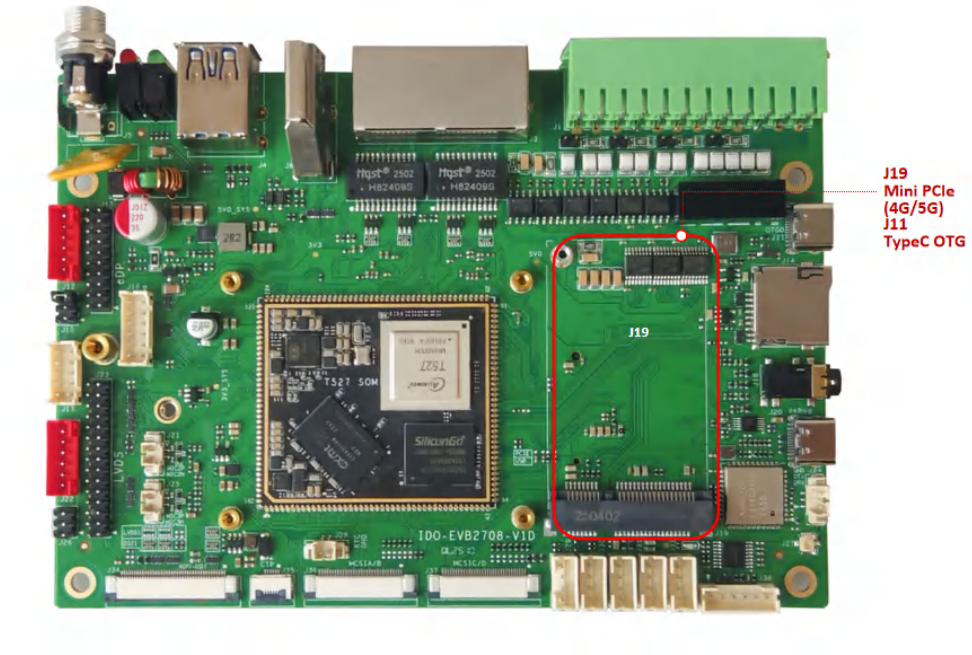


通过命令行连接蓝牙设备：

```shell
root@ido:/# bluetoothctl
[bluetooth]# scan on
...
[bluetooth]# trust 7C:C1:80:09:DD:6C
[bluetooth]# pair 7C:C1:80:09:DD:6C
[bluetooth]# connect 7C:C1:80:09:DD:6C
[cainiaocl]# exit
```

2.5 4G/5G

已适配支持EC20模组，主板接入EC20 模组，并给模块连接好天线



插入 `4G/5G` 模块和SIM卡后自动拨号；测试是否可上网如下图

```
root@localhost:/root# ifconfig wwan0
wwan0: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet [redacted]4  netmask 255.255.255.248
        inet6 fe[redacted]:1[redacted]:c1ff:feda:9049  prefixlen 64  scopeid 0x20<link>
        ether 12[redacted]:c1:c[redacted]  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 612 (612.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5  bytes 824 (824.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

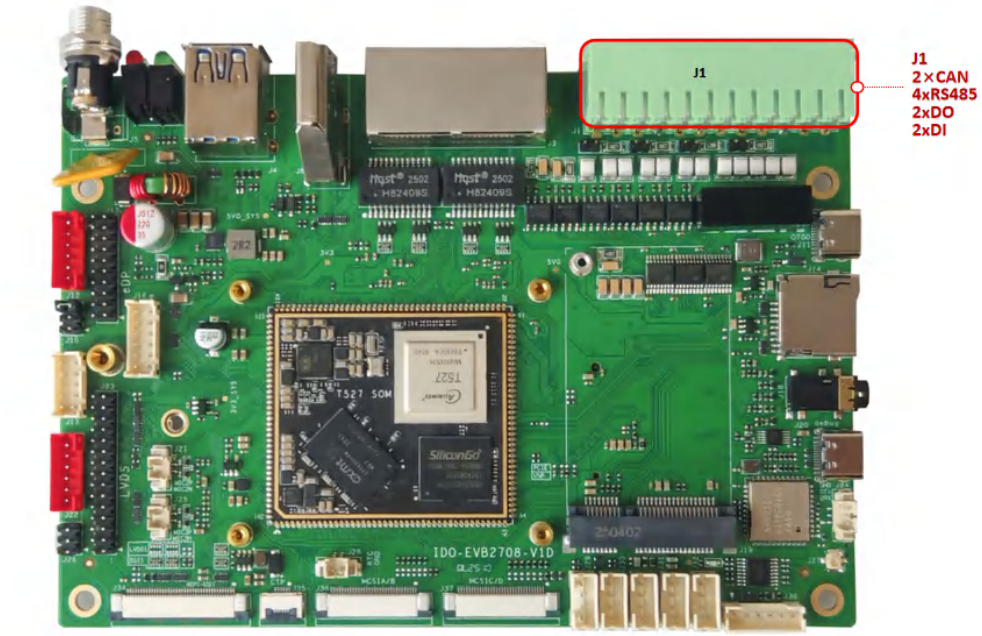
4G设备为 `wwan0` 节点，5G设备为 `USB0` 节点，拨号成功后为自动获取ip

通过 `ping www.baidu.com -I wwan0` 命令，验证 `4G/5G` 是否能ping通外网

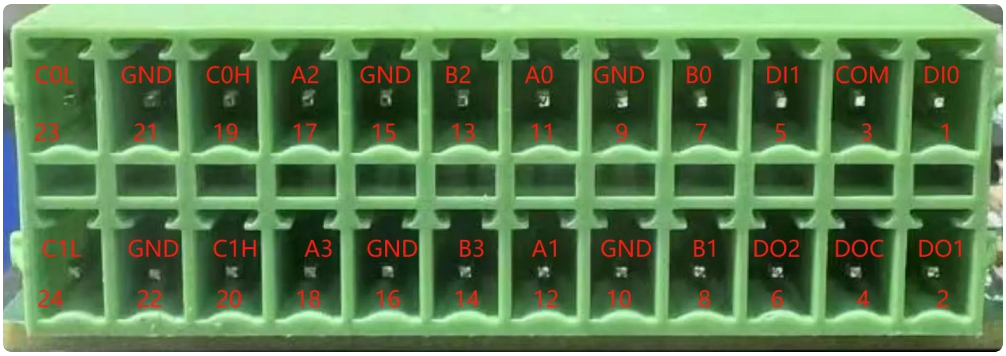
EC20模组 自动需要输入AT指令：

2.6 工业接线端子

工业接线端子位于开发板 `J1` 位置



引脚定义表如下：

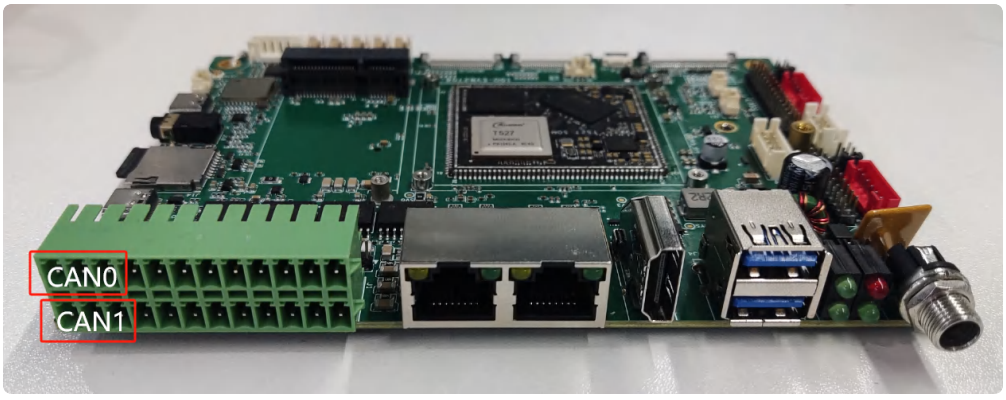


序号	定义	电平/V	说明
1	DI0	/	DI信号对
3	M	/	
5	DI1	/	
7	B0	/	RS485-0信号对（/dev/ttyS6）
9	ISO0_GND	/	
11	A0	/	
13	B2	/	RS485-2信号对（/dev/ttyS2）
15	ISO2_GND	/	

17	A2	/	
19	CAN0H	/	CAN0信号对
21	ISO5_GND	/	
23	CAN0L	/	
2	DO1	/	DO信号对
4	DOL	/	
6	DO2	/	
8	B1	/	RS485-1信号对（/dev/ttyS3）
10	ISO1_GND	/	
12	A1	/	
14	B3	/	RS485-3信号对（/dev/ttyS7）
16	ISO3_GND	/	
18	A3	/	
20	CAN1H	/	CAN1信号对
22	ISO4_GND	/	
24	CAN1L	/	

3.6.1 CAN

主板共有2路CAN接口，位置如下图所示。



两个节点名称分别为 awlink0和 awlink1

测试方法：

1. 设置CAN参数

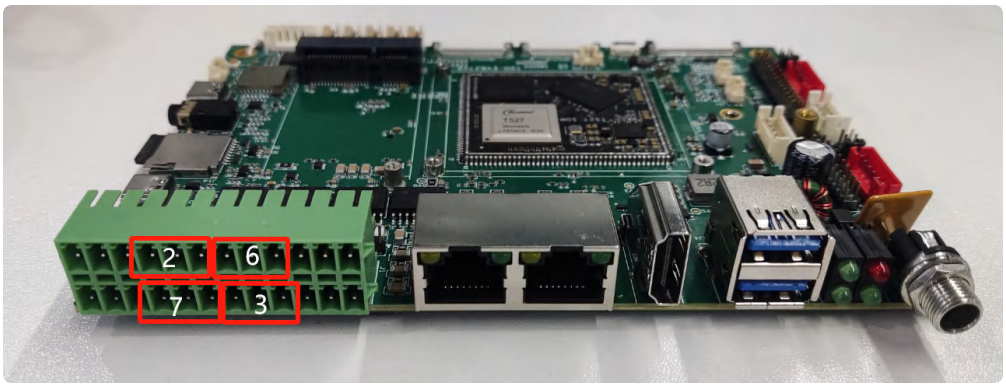
```shell
# 查看awlink0配置信息，可以看到awlink节点的波特率、位时序设置等
# ip -d -s -s link show awlink0

# 配置awlink之前需要先把awlink节点关闭
# ip link set awlink0 down

# 设置awlink0异常10ms重启
# ip link set awlink0 type can restart-ms 10

# 设置awlink0的比特率为1 Mbps/s，且开启回环
# ip link set awlink0 type can bitrate 1000000 loopback on

# 设置队列深度
# ip link set awlink0 qlen 300

# 打开awlink0节点
# ip link set awlink0 up
```

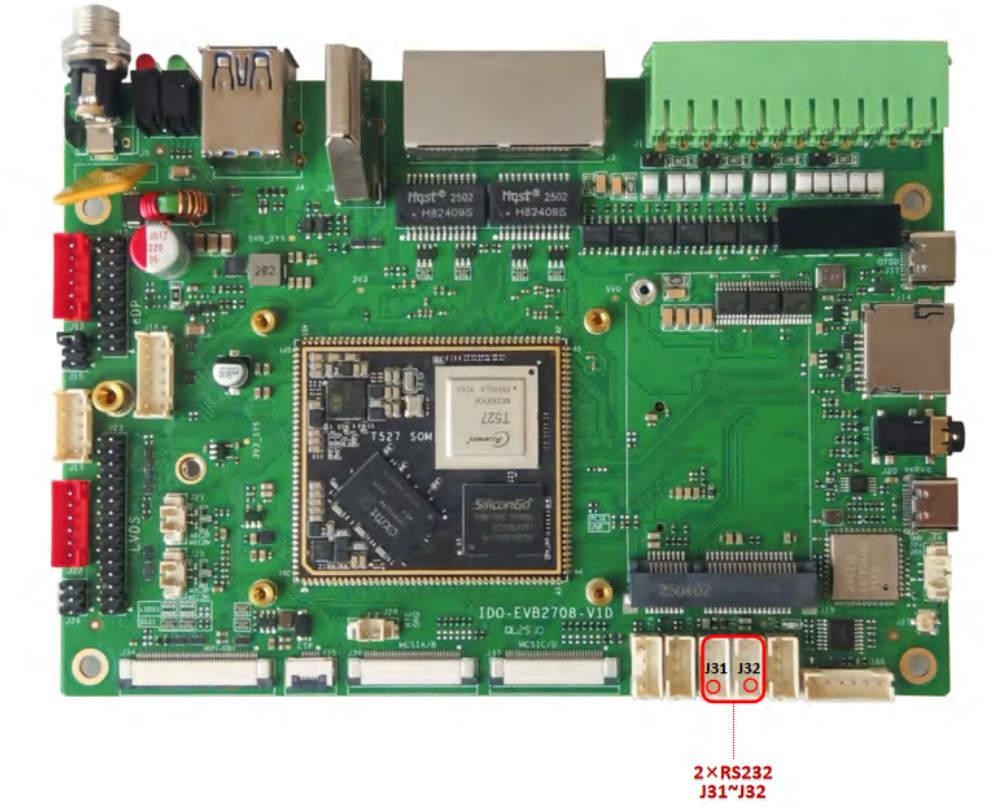
2. 测试收发

两台机器通过收发器连接好，发送方的 AWLINK_H 连接接收方的 AWLINK_H, 发送方的AWLINK_L 连接接收方的 AWLINK_L；

```shell
# candump工具转储总线上的发送接收信息
# candump -x -d awlink0 &

# awlink0节点发送一帧数据
# cansend awlink0 123#1122334455667788
awlink0 TX - - 123 [8] 11 22 33 44 55 66 77 88
awlink0 RX - - 123 [8] 11 22 33 44 55 66 77 88

# awlink0节点发送n帧数据
# cangen awlink0 -n 3
awlink0 TX - - 24E [5] 5D 31 F1 2F 96
awlink0 RX - - 24E [5] 5D 31 F1 2F 96
awlink0 TX - - 5FF [1] 7D
awlink0 RX - - 5FF [1] 7D
```

3.6.2 RS485





uart接口位置及引脚定义如上图所示，设备节点列表如下：

串口号	电平类型	设备节点
2	RS485	/dev/ttyAS2
3	RS485	/dev/ttyAS3
4	RS232	/dev/ttyAS4
5	RS232	/dev/ttyAS5
6	RS485	/dev/ttyAS6

7	RS485	/dev/ttyAS7

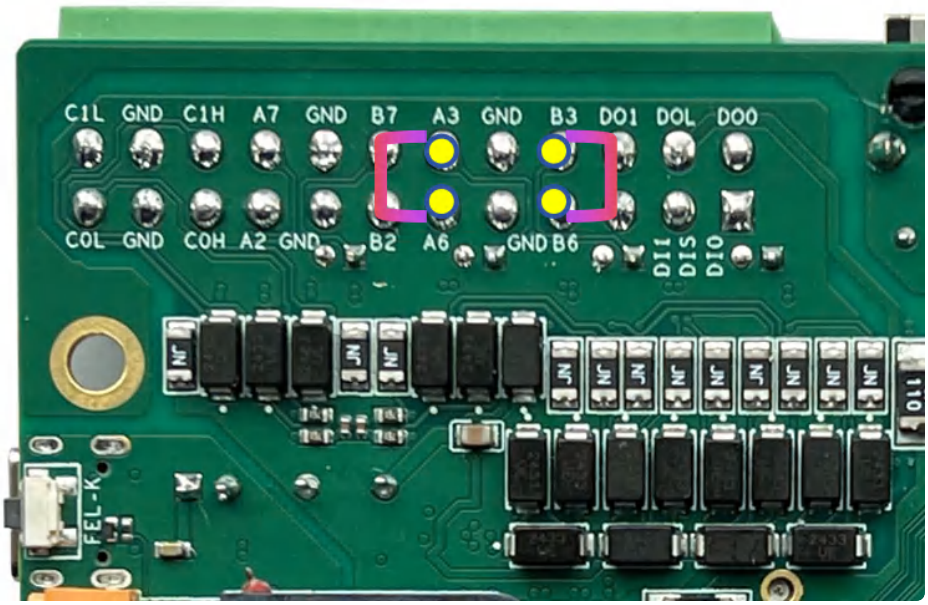
RS485测试

```shell
# 首先安装gcc所需的各种库
$ sudo apt update
$ sudo apt install gcc
```

📎uart.zip

将 `rs485_test.zip` 解压至开发板上任意目录下

短接任意两路485串口

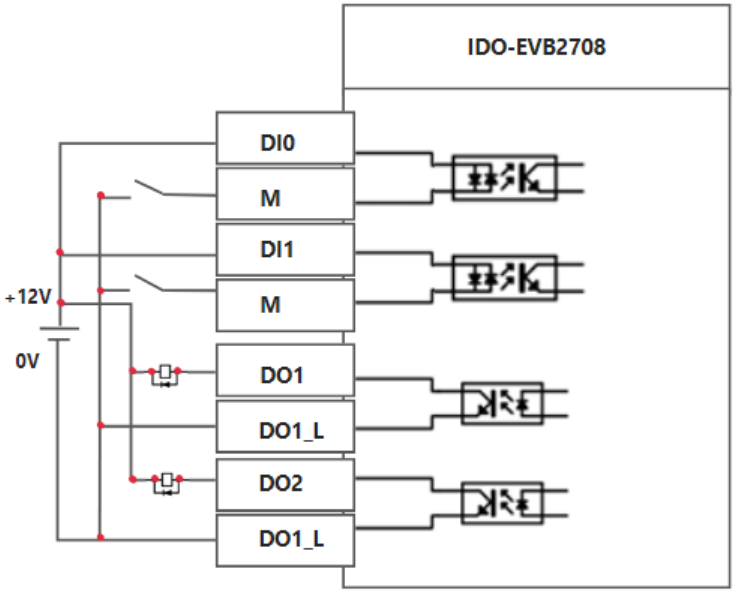


这里以短接 `/dev/ttyAS3` 和 `/dev/ttyAS6` 为例（硬件连接如上），对测成功会有如下打印，若只有timeout，则说明RS485通讯失败。

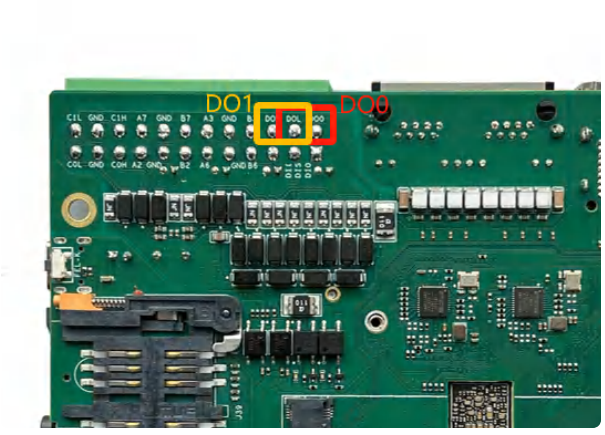
```shell
$ ./rs485_test /dev/ttyAS3 /dev/ttyAS6
------------------------------------------------------
68
------------------------------------------------------
------------------------------------------------------
01 00 00 99 14 59 68 11 04 33 33 34 33 b9 16
------------------------------------------------------
timeout
```

3.6.3 DIDO

DIDO逻辑接线图如下



3.6.3.1 DO



主板共有两路DO接口，位置如上图所示：

DOL为公共端口，当使能DO后，DO0和DO1分别和DOL短接。

控制节点为：/dev/ido_do

序号	设备节点	控制方法

DO0	/dev/ido_do	写0H，开启；写0L，断开；
DO1		写1H，开启；写1L，断开；

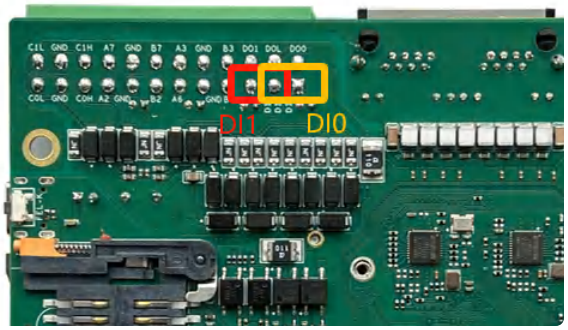
DO0 Shell

```
# DO0短接到DOL
echo 0H > /dev/ido_do
# DO0 和DOL 断开
echo 0L > /dev/ido_do
```

DO1 Shell

```
# DO1短接到DOL
echo 1H > /dev/ido_do
# DO1 和DOL 断开
echo 1L > /dev/ido_do
```

3.6.3.2 DI



主板共有两路DI接口，位置如上图所示：

DI0到DIS的电流大于1mA，DI0的软件接口检测到输入；

DI1到DIS的电流大于1mA，DI1的软件接口检测到输入；

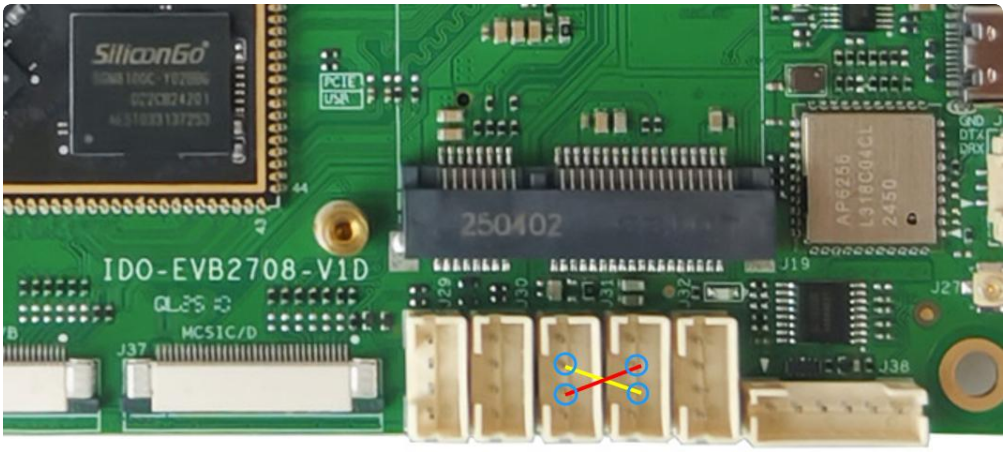
接口	设备节点	各个引脚单独接入时的状态
DI1	/dev/ido_di	01
DI2		10

命令行中获取DI状态

```shell
cat /dev/ido_di
```

2.7 RS232

RS232串口可将TX、RX短接测试



```shell
root@localhost:/# cat /dev/ttyAS4 &
[1] 26129
root@localhost:/# echo 123123 > /dev/ttyAS5
123123

root@localhost:/# echo 123123 > /dev/ttyAS5
123123

root@localhost:/# kill 26129
root@localhost:/#
[1]+  Terminated              cat /dev/ttyAS4
root@localhost:/# cat /dev/ttyAS5 &
[1] 26284
root@localhost:/# echo 123123 > /dev/ttyAS4
123123

root@localhost:/# echo 123123 > /dev/ttyAS4
123123
```

RS232也可以通过串口工具连接电脑进行收发测试

首先使用USB转RS232工具连接PC和开发板：

开发板平台下载串口收发工具，例如micrcom

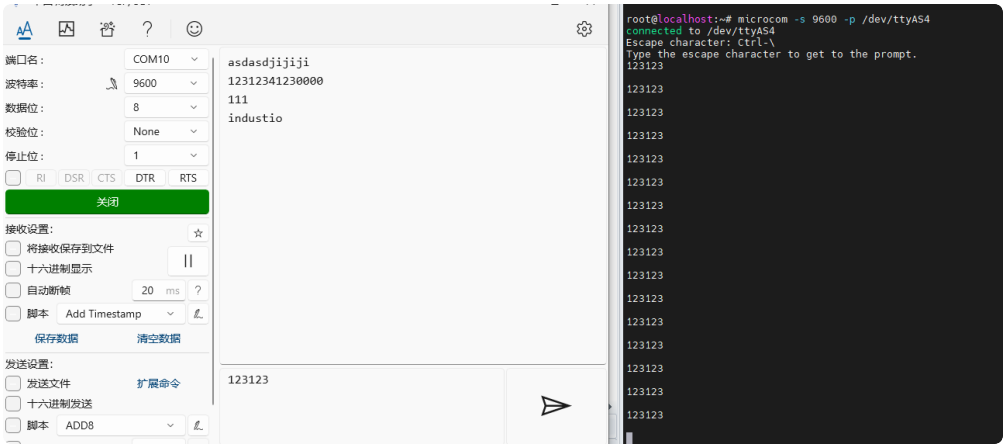
```shell
$ sudo apt-get update
$ sudo apt-get install microcom
```

通过microcom打开设备节点（以设置 `9600` 波特率打开 `/dev/ttyAS4` 为例）

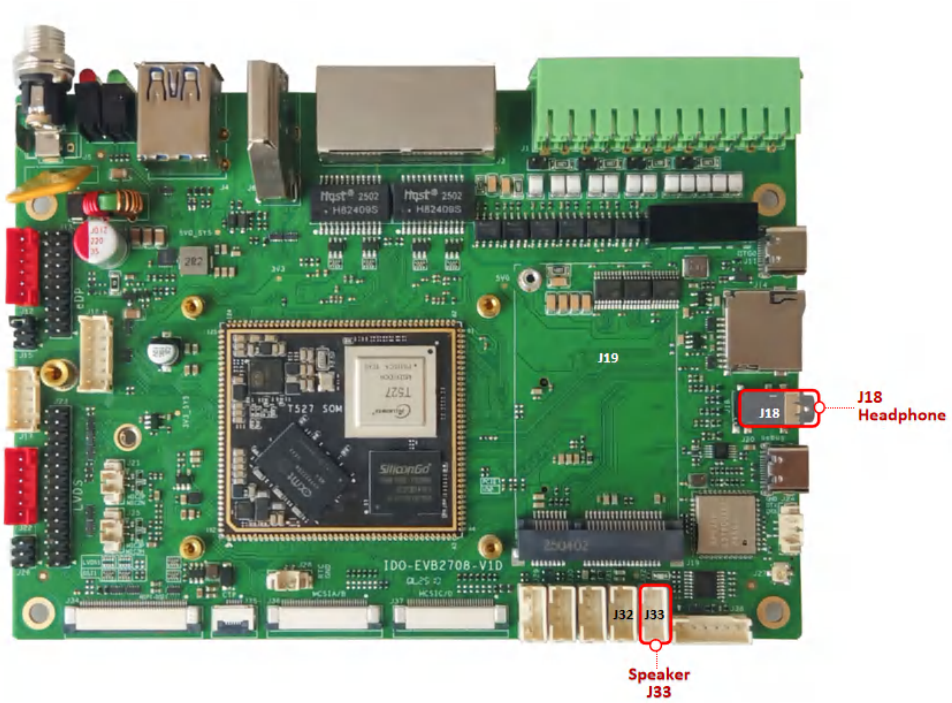
```shell
$ microcom -s 9600 -p /dev/ttyAS4
```

pc端通过串口调试工具选择对应串口，且设置对应波特率

测试收发：

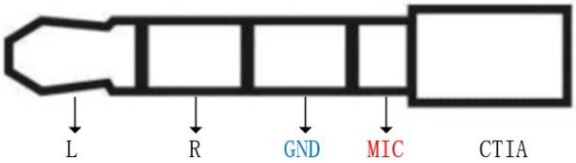


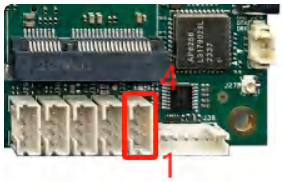
2.8 Audio



注意：

美标耳机的插头示意图如下所示。国标（OMTP）和美标（CTIA）的区别在于MIC和GND，两者相反， IDO-EVB2708-V1 CTIA接口示意图，如下图所示：





序号	定义	电平/V	说明
1	VORP	/	右声道喇叭驱动输出
2	VORN	/	
3	VOLP	/	左声道喇叭驱动输出
4	VOLN	/	

喇叭为PH2.0 4pin接口，最大支持8Ω@5W；耳机为一路美标四节耳机座。

耳机开关

```shell
$ root@industio:~# amixer -c 0 controls
……
numid=29,iface=MIXER,name='HPOUT Switch'
numid=19,iface=MIXER,name='LINEOUT Gain'
numid=27,iface=MIXER,name='LINEOUTL Switch'
numid=28,iface=MIXER,name='LINEOUTR Switch'
numid=24,iface=MIXER,name='MIC1 Switch'
numid=25,iface=MIXER,name='MIC2 Switch'
numid=26,iface=MIXER,name='MIC3 Switch'
numid=30,iface=MIXER,name='SPK Switch'
numid=2,iface=MIXER,name='rx sync mode'
numid=1,iface=MIXER,name='tx hub mode'

$ root@industio:~# amixer cset numid=29,iface=MIXER,name='HPOUT Switch' on
numid=29,iface=MIXER,name='HPOUT Switch'
  ; type=BOOLEAN,access=rw------,values=1
  : values=on
```

喇叭开关 Shell

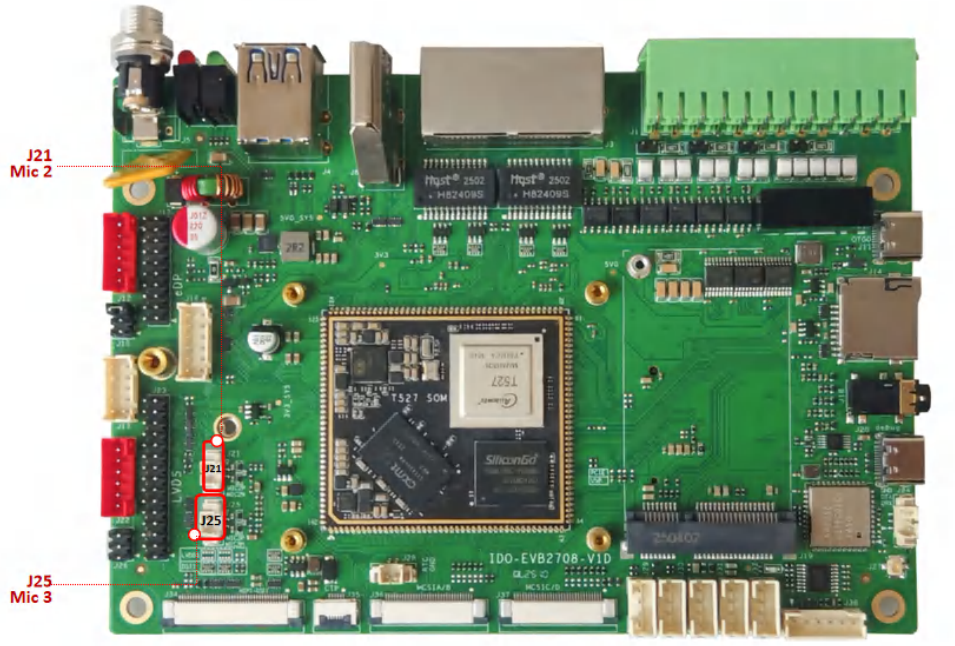
```
$ root@industio:~# amixer -c 0 controls
……
numid=29,iface=MIXER,name='HPOUT Switch'
numid=19,iface=MIXER,name='LINEOUT Gain'
numid=27,iface=MIXER,name='LINEOUTL Switch'
numid=28,iface=MIXER,name='LINEOUTR Switch'
numid=24,iface=MIXER,name='MIC1 Switch'
numid=25,iface=MIXER,name='MIC2 Switch'
numid=26,iface=MIXER,name='MIC3 Switch'
numid=30,iface=MIXER,name='SPK Switch'
numid=2,iface=MIXER,name='rx sync mode'
numid=1,iface=MIXER,name='tx hub mode'

####  通过amixer命令操作LINEOUT节点即可控制喇叭音量
####  LINEOUTL与LINEOUTR则单独对应左/右声道
####  values值为喇叭音量，最大值为31，最小值为0
root@industio:/# amixer -c 0 cset numid=19,iface=MIXER,name='LINEOUT Gain'
numid=19,iface=MIXER,name='LINEOUT Gain'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=31,step=0
  : values=31
  | dBrange-
    rangemin=0,,rangemax=1
      | dBscale-min=0.00dB,step=0.00dB,mute=1
    rangemin=2,,rangemax=31
      | dBscale-min=-43.50dB,step=1.50dB,mute=1

####  关闭喇叭
root@industio:/# amixer -c 0 cset numid=19,iface=MIXER,name='LINEOUT Gai
n' 0
numid=19,iface=MIXER,name='LINEOUT Gain'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=31,step=0
  : values=0
  | dBrange-
    rangemin=0,,rangemax=1
      | dBscale-min=0.00dB,step=0.00dB,mute=1
    rangemin=2,,rangemax=31
      | dBscale-min=-43.50dB,step=1.50dB,mute=1

####  打开喇叭
root@industio:/# amixer -c 0 cset numid=27,iface=MIXER,name='LINEOUTL Swit
ch' on
root@industio:/# amixer -c 0 cset numid=28,iface=MIXER,name='LINEOUTR Swit
ch' on
root@industio:/# amixer -c 0 cset numid=19,iface=MIXER,name='LINEOUT Gai
n' 31
```

```
numid=19,iface=MIXER,name='LINEOUT Gain'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=31,step=0
  : values=0
  | dBrange-
    rangemin=0,,rangemax=1
      | dBscale-min=0.00dB,step=0.00dB,mute=1
    rangemin=2,,rangemax=31
      | dBscale-min=-43.50dB,step=1.50dB,mute=1

####  将喇叭音量调节为“20”
root@industio:/# amixer -c 0 cset numid=19,iface=MIXER,name='LINEOUT Gain' 20
numid=19,iface=MIXER,name='LINEOUT Gain'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=31,step=0
  : values=20
  | dBrange-
    rangemin=0,,rangemax=1
      | dBscale-min=0.00dB,step=0.00dB,mute=1
    rangemin=2,,rangemax=31
      | dBscale-min=-43.50dB,step=1.50dB,mute=1

####  同样的方法可用在操作喇叭左/右声道
```

播放音频 Shell

```
$ aplay /usr/share/sounds/alsa/Rear_Right.wav
```

2.9 Mic录音



两路mic，于J21、J25（暂不支持多路mic同时录音）

MIC1为耳机MIC，需先确定耳机是否支持MIC功能

打开MIC1 Shell

```
root@industio:/# amixer -c 0 cset numid=24,iface=MIXER,name='MIC1 Switch' on
```

打开MIC2 Shell

```
root@industio:/# amixer -c 0 cset numid=25,iface=MIXER,name='MIC2 Switch' on
```

打开MIC3 Shell

```
root@industio:/# amixer -c 0 cset numid=26,iface=MIXER,name='MIC3 Switch' on
```

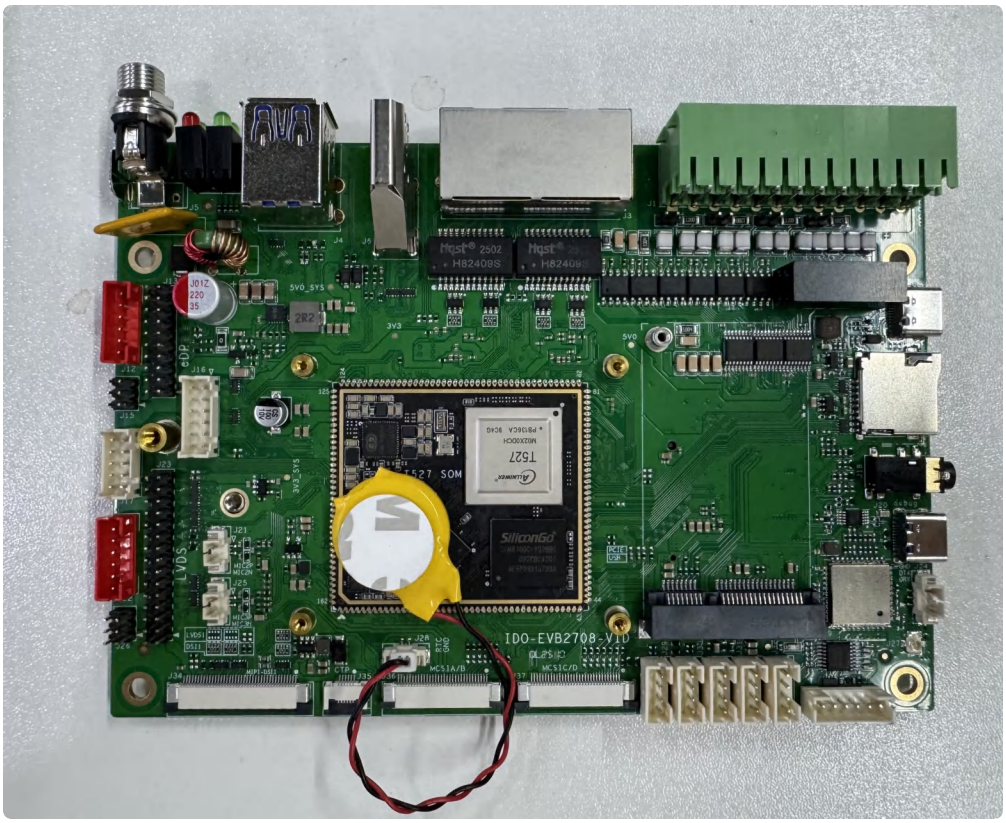
mic录音测试，MIC2对应开发板J21，MIC3对应J25

```shell
root@industio:/# arecord -D hw:0,0 -f S16_LE test.wav
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 8000 Hz, Mono
root@industio:/#
```

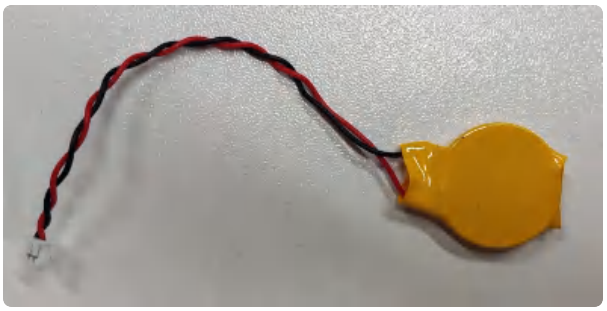
录音完毕后会在所在目录生成 `test.wav` 文件，接上喇叭通过 `aplay ./test.wav` 命令播放测试即可。

录音测试时，暂不允许多个MIC同时打开。

2.10 RTC



外部RTC HYM8563 电池座位于J28，规格为 MX1.25-2P 立式，可连接3V 纽扣电池，RTC电池参考如下



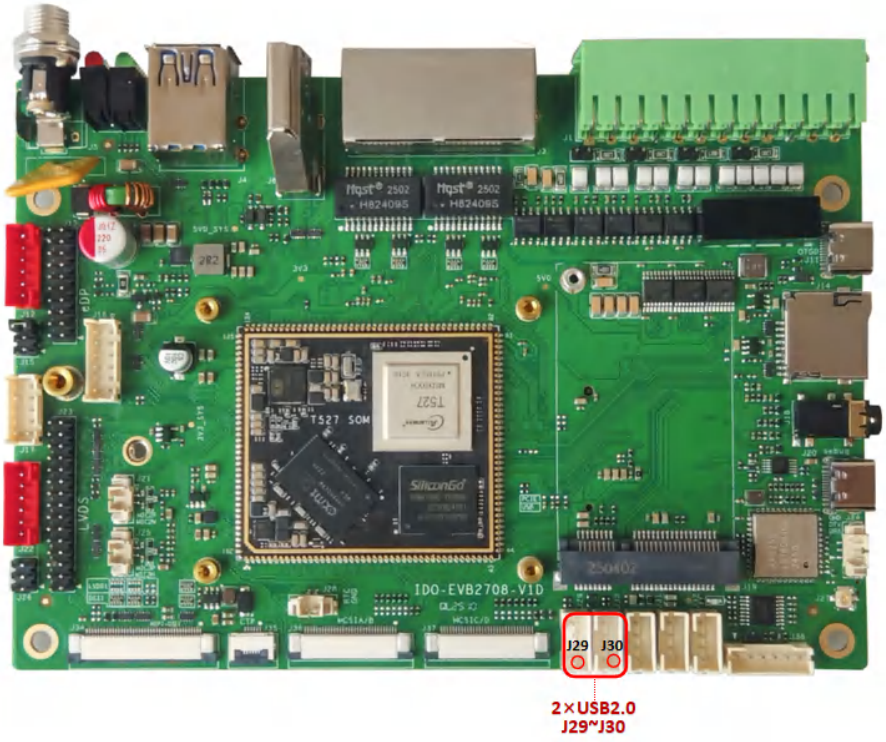
设备节点：/dev/rtc1

系统默认使用HYM8563作为系统时钟，时间设置方法：

```shell
#手动设置时间
$ date -s "2024-03-04 14:00:00"

#将rtc时钟调整为与目前的系统时钟一致
$ hwclock -f /dev/rtc1 -w

#获取硬件rtc当前时间
$ hwclock -f /dev/rtc1 -r
Fri Nov 24 14:01:31 2023  0.000000 seconds
```

2.11 USB

主板共有4路USB口，两个USB Type-A和两个PH2.0 4pin接口。



序号	功能	控电设备节点
1	USB 2.0 HOST	/sys/class/leds/host_fe1_pwr/brightness
2	USB 2.0 HOST	/sys/class/leds/host_fe2_pwr/brightness
3	USB 2.0 HOST	/sys/class/leds/host_fe3_pwr/brightness
4	USB 3.0 HOST	/sys/class/leds/host1_pwr/brightness

供电控制说明：设备节点写"0"关闭电源，写"1"开启电源

命令行控制方法如下，以序号1为例

Shell

```
#关闭
echo 0 > /sys/class/leds/host_fe1_pwr/brightness
#开启（默认状态）
echo 1 > /sys/class/leds/host_fe1_pwr/brightness
```

U盘自动挂载：

新建 `/lib/udev/rules.d/99-udisk-hotplug.rules` ：

```shell
KERNEL!="sd[a-z][0-9]", GOTO="udisk_hotplug_end"
SUBSYSTEM!="block",GOTO="udisk_hotplug_end"

#获取快设备信息
IMPORT{program}="/sbin/blkid -o udev -p %N"
#仅文件系统类型是非空时执行
ENV{ID_FS_TYPE}=="", GOTO="udisk_hotplug_end"
#当块设备Label非空时将label赋值值给label_uuid_name,否则用uuid赋值
ENV{ID_FS_LABEL}!="", ENV{label_uuid_name}="%E{ID_FS_LABEL}"
ENV{ID_FS_LABEL}=="", ENV{label_uuid_name}="%E{ID_FS_UUID}"

#检测到插入/移除设备时，执行脚本文件，并传递设备名称和label_uuid_name
ACTION=="add", RUN+="/etc/init.d/udisk-hotplug.sh %k %E{label_uuid_name}"
ACTION=="remove",RUN+="/etc/init.d/udisk-hotplug.sh %k %E{label_uuid_name}"

LABEL="udisk_hotplug_end"
```

新建 `/etc/init.d/udisk-hotplug.sh`：

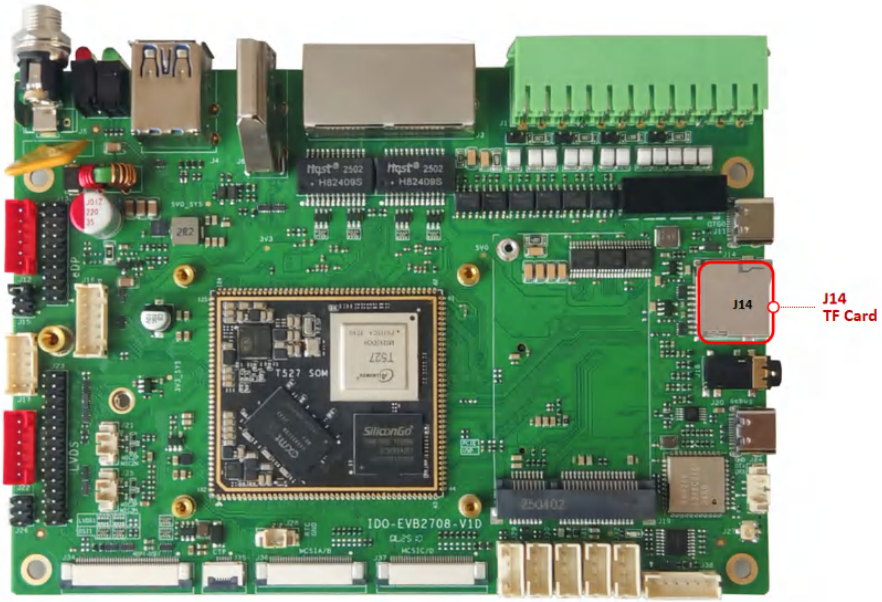
```shell
#!/bin/bash

if [ $# -ne 2 ]; then
        exit 1
fi
#获取设备名称
DEV_NAME=$1

#挂载主目录设置为/udisk
MAJOR_PATH=/mnt/usb
#设置二级目录为获取到的label或uuid名称
MINOR_PATH=$2

if [ $ACTION == "add" ]; then
        mkdir -p $MAJOR_PATH/$MINOR_PATH
        /usr/bin/systemd-mount -o relatime,sync,uid=1000,gid=1000 --no-block --collect  /dev/$DEV_NAME $MAJOR_PATH/$MINOR_PATH

        if [[ $? -ne 0 ]]; then
                rmdir $MAJOR_PATH/$MINOR_PATH
        fi
elif [ $ACTION == "remove" ];then
        if [[ -e $MAJOR_PATH/$MINOR_PATH ]]; then
                /usr/bin/systemd-mount --umount $MAJOR_PATH/$MINOR_PATH
                rmdir $MAJOR_PATH/$MINOR_PATH
        fi

fi
```

并赋予可执行权限：

```shell
industio@industio:~$ sudo chmod a+x /etc/init.d/udisk-hotplug.sh
industio@industio:~$ sudo chown root:root /etc/init.d/udisk-hotplug.sh
```

重启系统生效。

2.12 TF Card



插上tf卡（接入时tf卡金手指朝向开发板），通过 `fdisk -l` 可以查看是否识别到tf卡分区，如下。

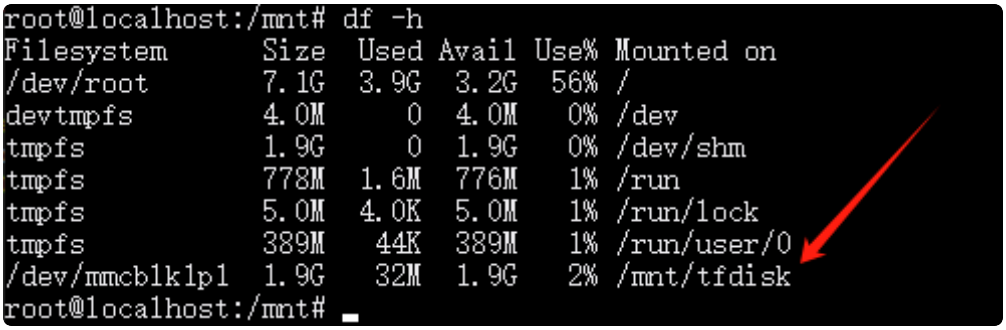
```
root@localhost:/mnt# fdisk -l
Disk /dev/mmcblk0: 29.12 GiB, 31268536320 bytes, 61071360 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: AB6F3888-569A-4926-9668-80941DCB40BC

Device           Start      End  Sectors  Size Type
/dev/mmcblk0p1    73728   139263    65536   32M Microsoft basic data
/dev/mmcblk0p2   139264   172031    32768   16M Microsoft basic data
/dev/mmcblk0p3   172032   368639   196608   96M Microsoft basic data
/dev/mmcblk0p4   368640 15608319 15239680  7.3G Microsoft basic data
/dev/mmcblk0p5 15608320 61071326 45463007 21.7G Microsoft basic data


Disk /dev/mmcblk1: 1.86 GiB, 1999110144 bytes, 3904512 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device         Boot Start     End Sectors  Size Id Type
/dev/mmcblk1p1        129 3904511 3904383  1.9G  6 FAT16
root@localhost:/mnt#
```

可手动挂载 `/dev/mmcblk1p1` 分区

Shell

```
root@localhost:/# cd /mnt/
root@localhost:/mnt# mkdir tfdisk
root@localhost:/mnt# mount /dev/mmcblk1p1 ./tfdisk/
```

挂在成功后可以通过 `df -h` ，可识别到如下分区节点



SD卡自动挂载：

新建 `/lib/udev/rules.d/99-sdcard-hotplug.rules` ：

Shell

```
KERNEL!="mmcblk1p1", GOTO="sdcard_hotplug_end"
SUBSYSTEM!="block",GOTO="sdcard_hotplug_end"

ACTION=="add", RUN+="/etc/init.d/sdcard-hotplug.sh %k"
ACTION=="remove",RUN+="/etc/init.d/sdcard-hotplug.sh %k"
LABEL="sdcard_hotplug_end"
```

新建 `/etc/init.d/sdcard-hotplug.sh` ：

```shell
#!/bin/bash

if [ $# -ne 1 ]; then
        exit 1
fi

MOUNT_PATH=/mnt/sdcard
DEV_NAME=$1

if [ $ACTION == "add" ]; then
        mkdir -p $MOUNT_PATH
        /usr/bin/systemd-mount -o relatime,sync,uid=1000,gid=1000 --no-block --collect  /dev/$DEV_NAME $MOUNT_PATH/

        if [[ $? -ne 0 ]]; then
                rmdir $MOUNT_PATH/
        fi
elif [ $ACTION == "remove" ];then
        if [[ -e $MOUNT_PATH ]]; then
                /usr/bin/systemd-mount --umount $MOUNT_PATH
                rmdir $MOUNT_PATH
        fi

fi
```

并赋予可执行权限：

```shell
industio@industio:~$ sudo chmod a+x /etc/init.d/sdcard-hotplug.sh
industio@industio:~$ sudo chown root:root /etc/init.d/sdcard-hotplug.sh
```

重启系统生效。

2.13 LED指示灯

主板共有4个LED指示灯，位置如上图所示，接口说明如下表

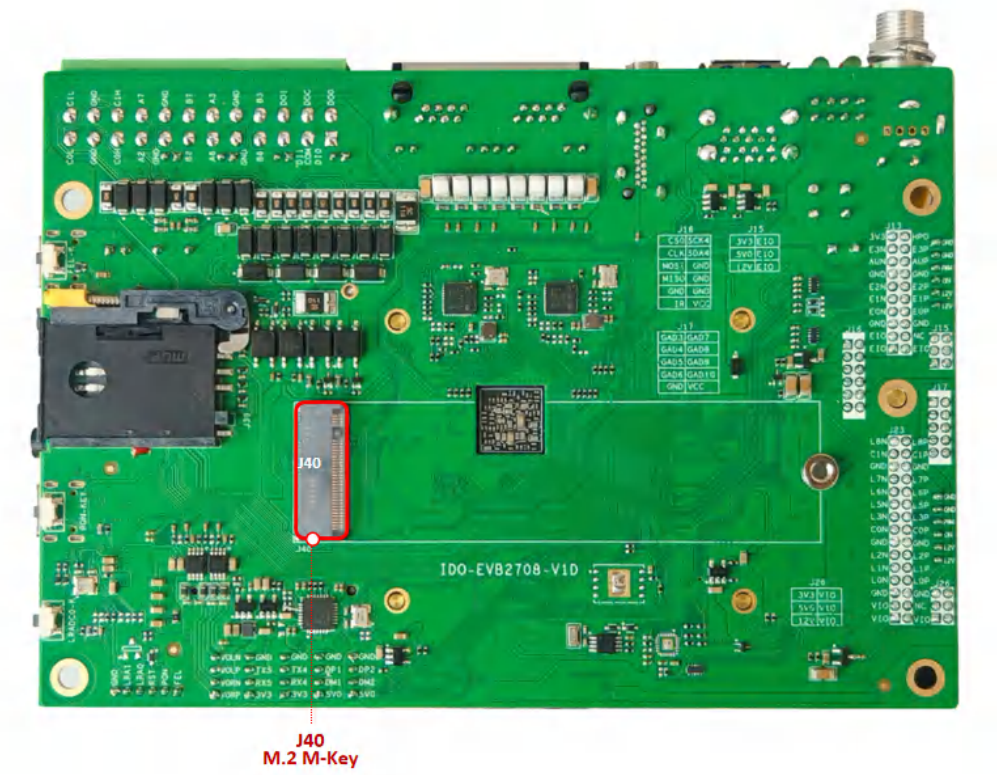
序号	说明	控制节点	说明
1	电源指示灯	无	系统上电后，亮灯
2	系统工作心跳灯	/sys/class/leds/work/brightness	系统工作后，心跳灯闪烁
3	用户指示灯1	/sys/class/leds/user1/brightness	默认常亮
4	用户指示灯2	/sys/class/leds/user2/brightness	默认常亮

用户指示灯控制方法，以指示灯1为例：

Shell

```
# 开启user1灯
echo 1 > /sys/class/leds/user1/brightness
# 关闭user1灯（默认状态）
echo 0 > /sys/class/leds/user1/brightness
```

2.14 M.2接口



与USB3.0 复用，默认配置为USB3.0，如果需要使用PCIe M.2 接口，则需要修改硬件，软件上也需要烧录相应的固件。

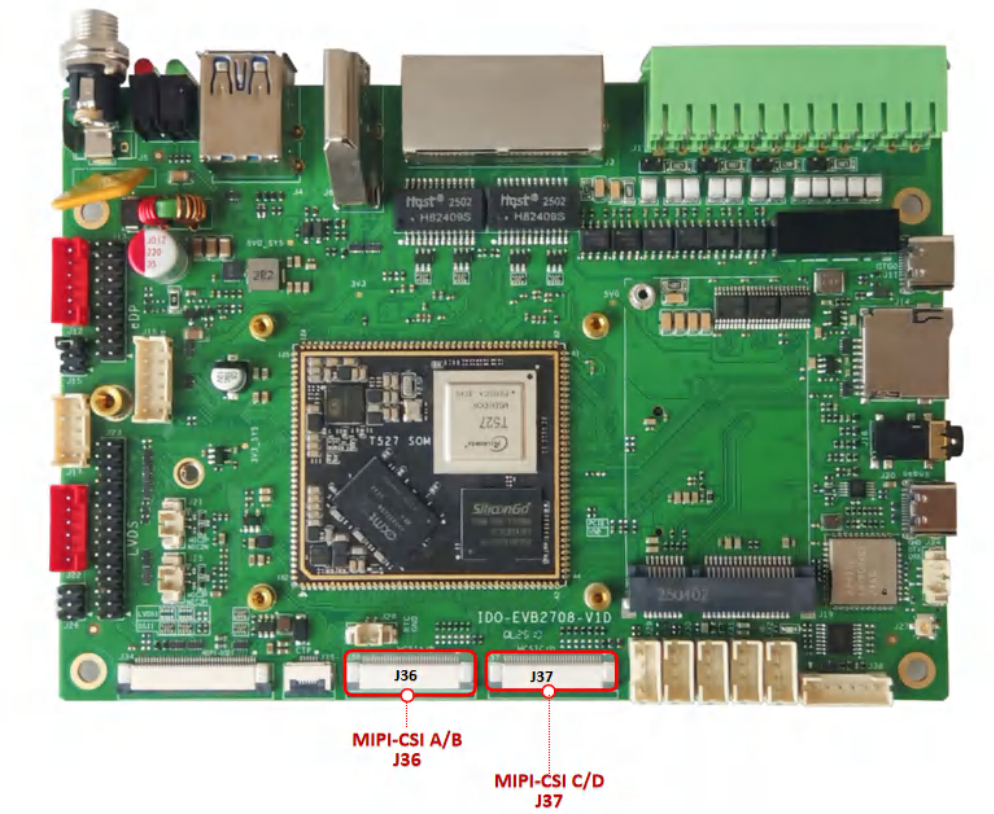
挂载m.2固态 Shell

```
root@industio:/# fdisk -l
Disk /dev/mmcblk0: 29.12 GiB, 31268536320 bytes, 61071360 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: AB6F3888-569A-4926-9668-80941DCB40BC

Device            Start      End  Sectors  Size Type
/dev/mmcblk0p1    73728   139263    65536   32M Microsoft basic data
/dev/mmcblk0p2   139264   172031    32768   16M Microsoft basic data
/dev/mmcblk0p3   172032   368639   196608   96M Microsoft basic data
/dev/mmcblk0p4   368640 15608319 15239680  7.3G Microsoft basic data
/dev/mmcblk0p5 15608320 61071326 45463007 21.7G Microsoft basic data


Disk /dev/nvme0n1: 119.24 GiB, 128035676160 bytes, 250069680 sectors
Disk model: Colorful CN600 128GB
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc9ab632a

Device         Boot Start       End   Sectors   Size Id Type
/dev/nvme0n1p1       2048 250069646 250067599 119.2G  7 HPFS/NTFS/exFAT
root@industio:/# cd /mnt
root@industio:/mnt# ls
rootfs  usb
root@industio:/mnt# mkdir nvme
root@industio:/mnt# mount /dev/nvme0n1p1 ./nvme/
root@industio:/mnt# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.1G  3.5G  3.6G  50% /
devtmpfs        4.0M     0  4.0M   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           776M  1.6M  775M   1% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           388M   44K  388M   1% /run/user/0
/dev/nvme0n1p1  120G  2.1G  118G   2% /mnt/nvme

############ 取消挂载 ############
root@industio:/mnt# umount ./nvme
root@industio:/mnt#
```

若需要自行编译出支持pcie的固件，可在dts钟修改 `&combophy` 节点

```Shell
&combophy {
        resets = <&ccu RST_BUS_PCIE_USB3>;
        phy_use_sel = <2>; /* 0:PCIE; 1:USB3; 2:PCIE_USB3_U2 */
        status = "okay";
};

```

2.15 Camera



主板共有两路mipi摄像头，如上图所示

接口	设备节点
MIPI-CSI A/B	/dev/vido0，/dev/video1
MIPI-CSI C/D	/dev/vido4，/dev/video5

摄像头测试方法如下：

```shell
$ csi_test_mplane 0 0 1920 1080 ./ 1 20
参数依次为：
• 【参数 0】./csi_test_mplane：测试用例可执行文件
• 【参数 1】0：/dev/video0
• 【参数 2】0：set_input 时传入的 index 参数（默认设置为 0 即可）
• 【参数 3、4】1920 1080：抓图目标图像分辨率
• 【参数 5】./：抓取到的图像 bin 文件所保存路径，可自行修改
• 【参数6】1：图像格式，1表示抓取YUV420M，其他分辨率可根据测试用例csi_test_mplane.c代码中 camera_fmt_set 函数来修改相应的图像格式。
• 【参数 7】20：采集帧数（若想要持续不间断抓图，将此帧数参数设置为 10000 及以上即可，常用于调试阶段使用）
• 【参数 8，可选】目标帧率（需根据实际所接模组或驱动配置进行合理选择）
• 【参数 9，可选】是否开启 WDR 功能
• 最后会在 bat 指令的文件夹生成 result 文件夹里面保存二进制的图像数据 *.bin 文件
• 可用 RawViewer 等软件查看图像数据，注意看图时，看图软件中所设置的分辨率需注意按 16 字节进行对齐（向上取整），如使用 RawViewer 查看 1920x1080 图像时，需将看图尺寸设置为 1920x1088，否则可能会看到花屏或抖动的图像。
```

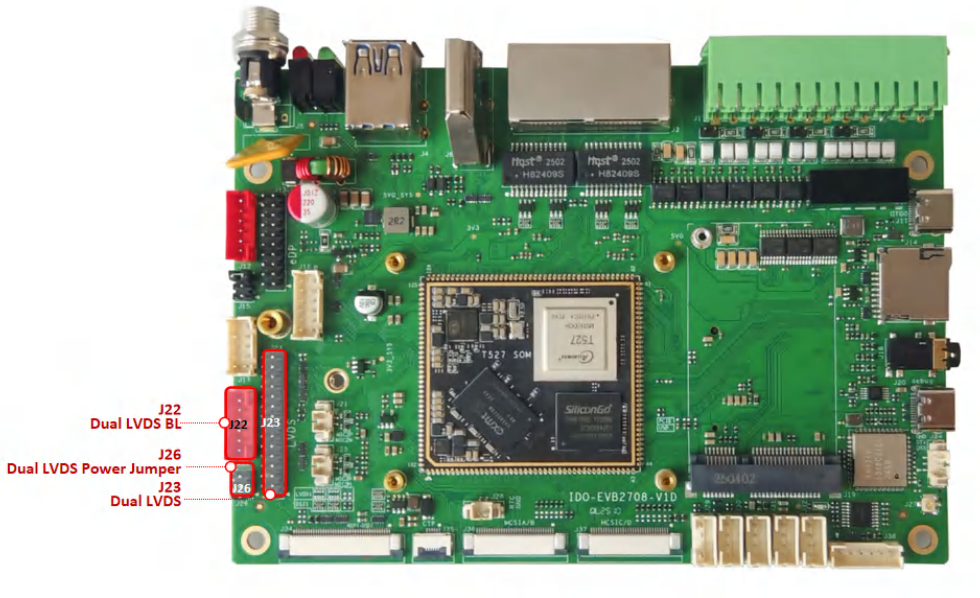
📎RawViewer.exe.zip

2.16 LCD显示屏

2.16.1 LVDS屏

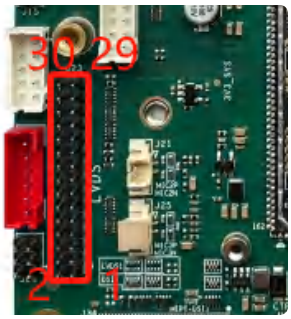
单LVDS固件使用的屏幕型号为 `TD101R1040H01`，分辨率为 `1280x800`。

双LVDS固件使用的屏幕型号为 `LR060FHD-L01`，分辨率为 `1920x1080`。



接屏前，需要根据具体的屏幕规格书来确认J26的供电跳线帽短接到3.3V、5V还是12V。

背光接 J22，屏幕排线接J23。



序号	定义	电平/V	说明
1	LVDS_VIO	3.3V/5V/12V	• LVDS屏幕供电 • 3.3V/5V/12V可通过J21用2mm跳线帽选择 • 主板默认通过跳线帽配置成3.3V
2	LVDS_VIO	3.3V/5V/12V	
3	LVDS_VIO	3.3V/5V/12V	
4	NC	/	NC
5	GND	GND	电源地
6	GND	GND	电源地
7	LVDS0_D0N	/	LVDS0_D0信号对
8	LVDS0_D0P	/	

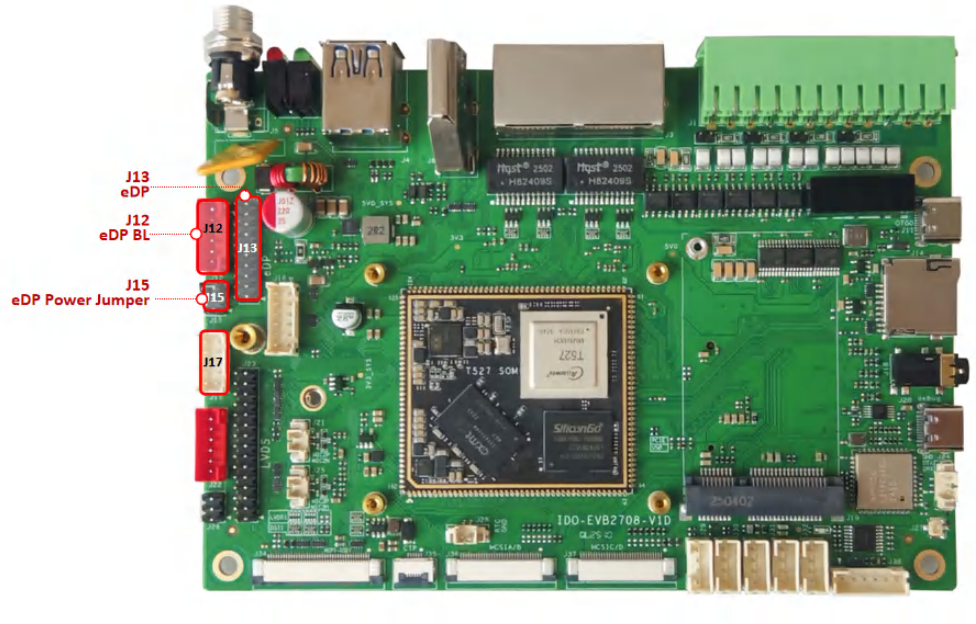
9	LVDS0_D1N	/	LVDS0_D1信号对
10	LVDS0_D1P	/	
11	LVDS0_D2N	/	LVDS0_D2信号对
12	LVDS0_D2P	/	
13	GND	GND	电源地
14	GND	GND	电源地
15	LVDS0_CLKN	/	LVDS0_CLK信号对
16	LVDS0_CLKP	/	
17	LVDS0_D3N	/	LVDS0_D3信号对
18	LVDS0_D3P	/	
19	LVDS1_D0N	/	LVDS1_D0信号对
20	LVDS1_D0P	/	
21	LVDS1_D1N	/	LVDS1_D1信号对
22	LVDS1_D1P	/	
23	LVDS1_D2N	/	LVDS1_D2信号对
24	LVDS1_D2P	/	
25	GND	GND	电源地
26	GND	GND	电源地
27	LVDS1_CLKN	/	LVDS1_CLK信号对
28	LVDS1_CLKP	/	
29	LVDS1_D3N	/	LVDS1_D3信号对
30	LVDS1_D3P	/	

2.16.2 EDP屏

接屏前，需要根据具体的屏幕规格书来确认J26的供电跳线帽短接到3.3V、5V还是12V。

背光接 J22，屏幕排线接J23。

eDP固件使用的屏幕型号为 `NV133FHM-N42` ，分辨率为 `1920x1080` 。



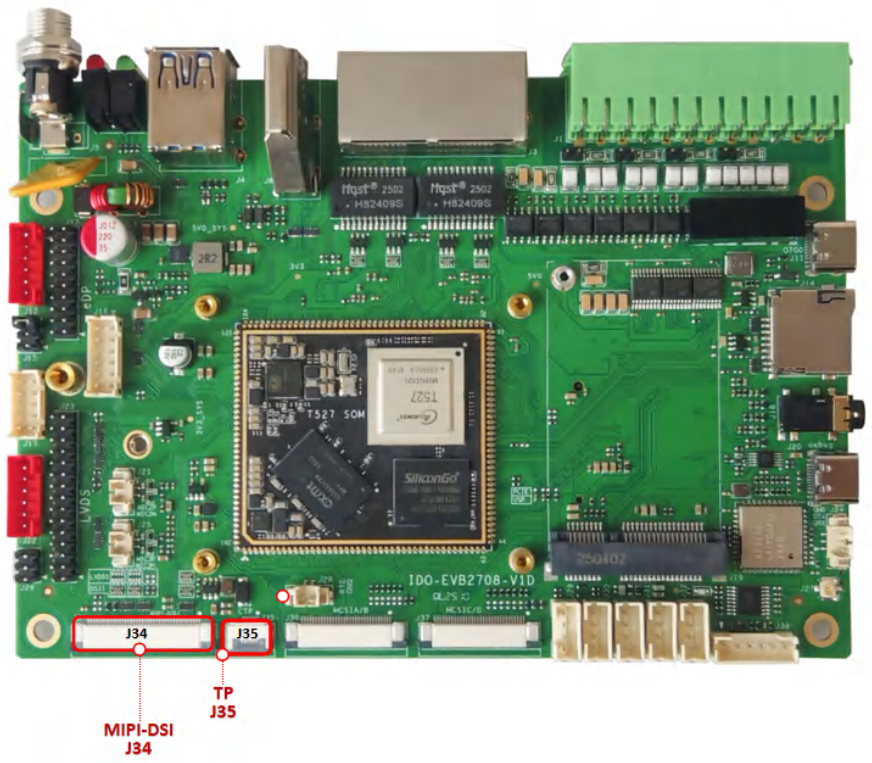
序号	定义	电平/V	说明
1	VCC_eDP_VIO	/	• eDP屏幕驱动电压 • 3.3V/5V/12V可通过J25用2mm跳线帽选择 • 主板默认通过跳线帽配置成3.3V
2	VCC_eDP_VIO	/	
3	NC	/	NC
4	GND	GND	电源地
5	GND	GND	电源地
6	GND	GND	电源地
7	eDP_TX_D0N	/	eDP_TX_D0信号对

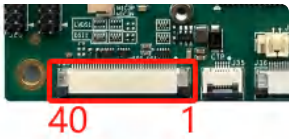
8	eDP_TX_D0P	/	
9	eDP_TX_D1N	/	eDP_TX_D1信号对
10	eDP_TX_D1P	/	
11	eDP_TX_D2N	/	eDP_TX_D2信号对
12	eDP_TX_D2P	/	
13	GND	GND	电源地
14	GND	GND	电源地
15	eDP_TX_AUXN	/	eDP_TX_AUX信号对
16	eDP_TX_AUXP	/	
17	eDP_TX_D3N	/	eDP_TX_D3信号对
18	eDP_TX_D3P	/	
19	VCC3V3	3.3V	3.3V供电
20	eDP_HPD	/	eDP热拔插检测

2.16.3 MIPI屏

屏幕屏线接J34，I2C接口触摸屏接J35口。

mipi固件使用的屏幕型号为 `RY10-6127GC` ，分辨率为 `1920x1200` 。





序号	定义	电平/V	说明
1	VCC_LEDA_TX1	/	屏幕背光源输出正极
2	VCC_LEDA_TX1	/	
3	NC	/	悬空
4	TP_RST	3.3V	复位信号
5	TP_INT	3.3V	中断信号
6	TP_SCL	3.3V	I2C时钟信号
7	TP_SDA	3.3V	I2C数据信号
8	NC	/	悬空
9	VCC_LEDK_TX1	/	屏幕背光源输出负极
10	VCC_LEDK_TX	/	
11	GND	GND	电源地

12	NC	/	悬空
13	NC	/	
14	NC	/	
15	NC	/	
16	GND	GND	电源地
17	VCC3V3_SYS	3.3V	触摸屏供电输出3.3V
18	GND	GND	电源地
19	GND	GND	电源地
20	MIPI_DPHY1_TX_D3P	/	MIPI_DPHY1_TX_D3信号对
21	MIPI_DPHY1_TX_D3N	/	
22	GND	GND	电源地
23	MIPI_DPHY1_TX_D2P	/	MIPI_DSI_TX_D2信号对
24	MIPI_DPHY1_TX_D2N	/	
25	GND	GND	电源地
26	MIPI_DPHY1_TX_CLKP	/	MIPI_DSI_TX_CLK信号对
27	MIPI_DPHY1_TX_CLKN	/	
28	GND	GND	电源地
29	MIPI_DPHY1_TX_D1P	/	MIPI_DPHY1_TX_D1信号对
30	MIPI_DPHY1_TX_D1N	/	
31	GND	GND	电源地
32	MIPI_DPHY1_TX_D0P	/	MIPI_DPHY1_TX_D0信号对
33	MIPI_DPHY1_TX_D0N	/	
34	GND	GND	电源地
35	NC	/	悬空
36	MIPI_DPHY_TX1_RST	3.3V	LCD复位信号

37	GND	GND	电源地
38	VCC3V3	3.3V	屏幕供电输出3.3V
39	VCC3V3	3.3V	
40	NC	/	默认悬空

注意：MIPI_DPHY1_TX背光电流可通过更改物料调节，默认100mA。