

IDO-EVB5301-V1 Linux使用手册



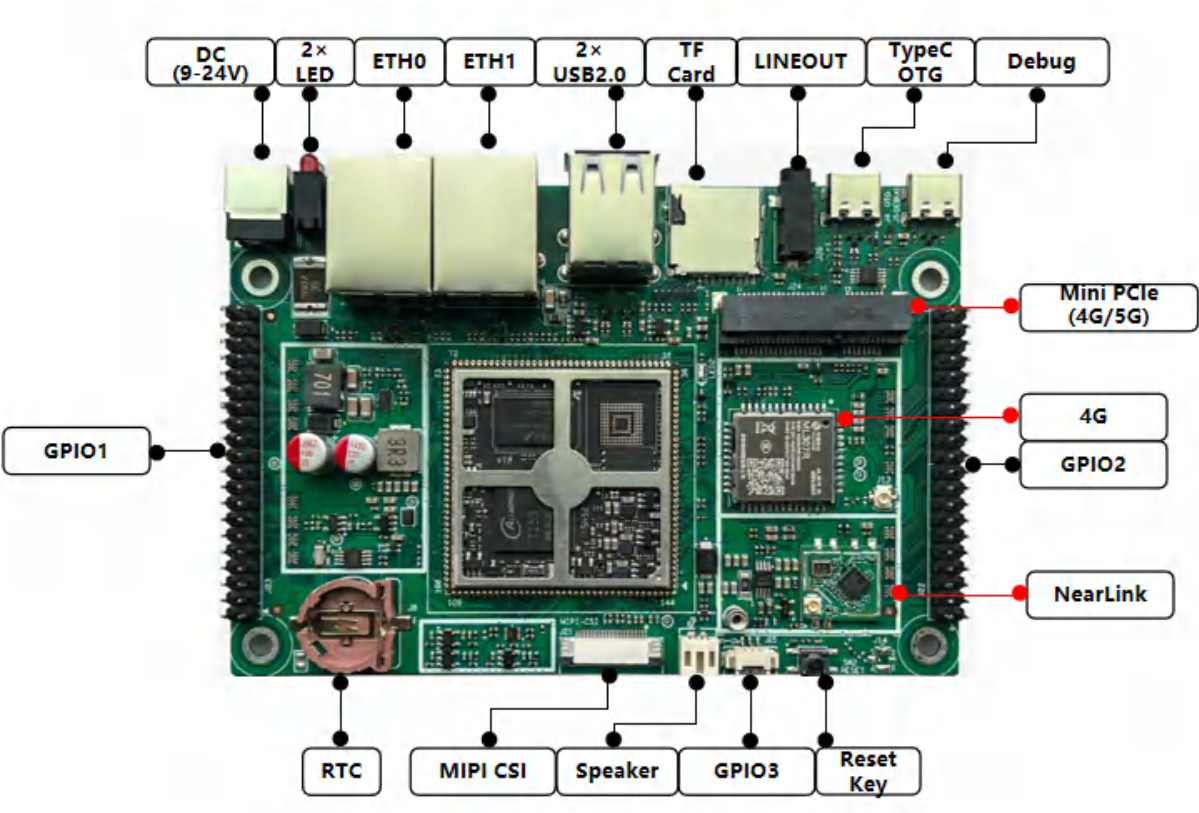
IDO-EVB5301-V1 Linux使用手册

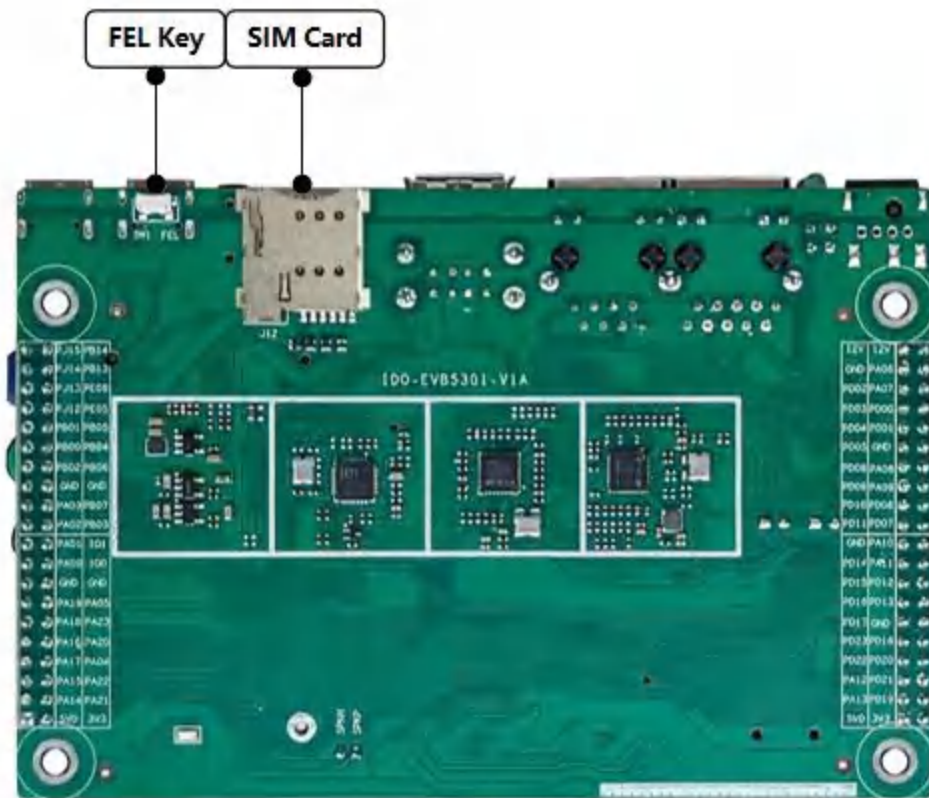
深圳触觉智能科技有限公司

www.industio.cn

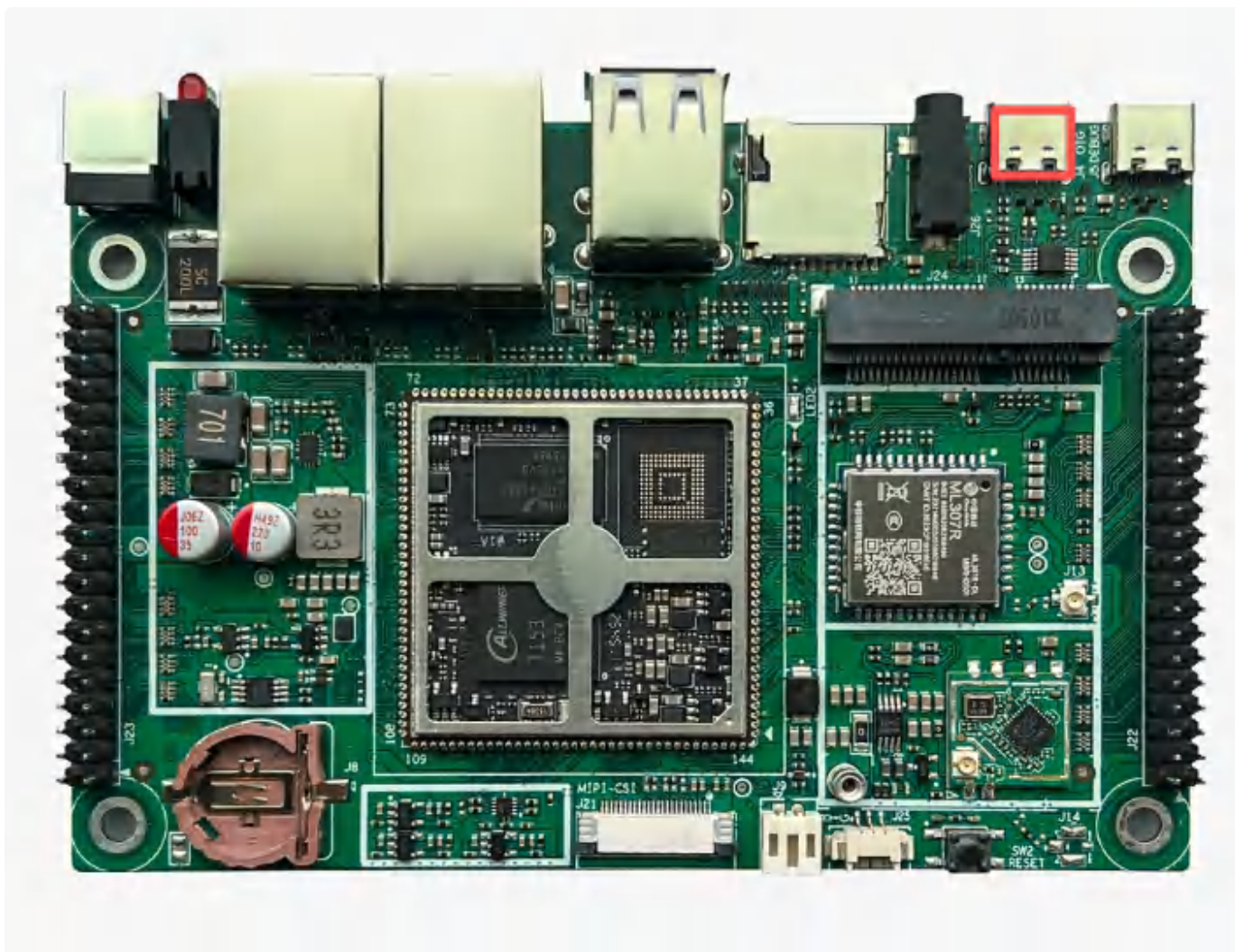
文档修订历史

版本	PCBA版本	修订内容	修订	审核	日期
V1.0	V1B	创建文档	LJH	IDO	2025/09/28



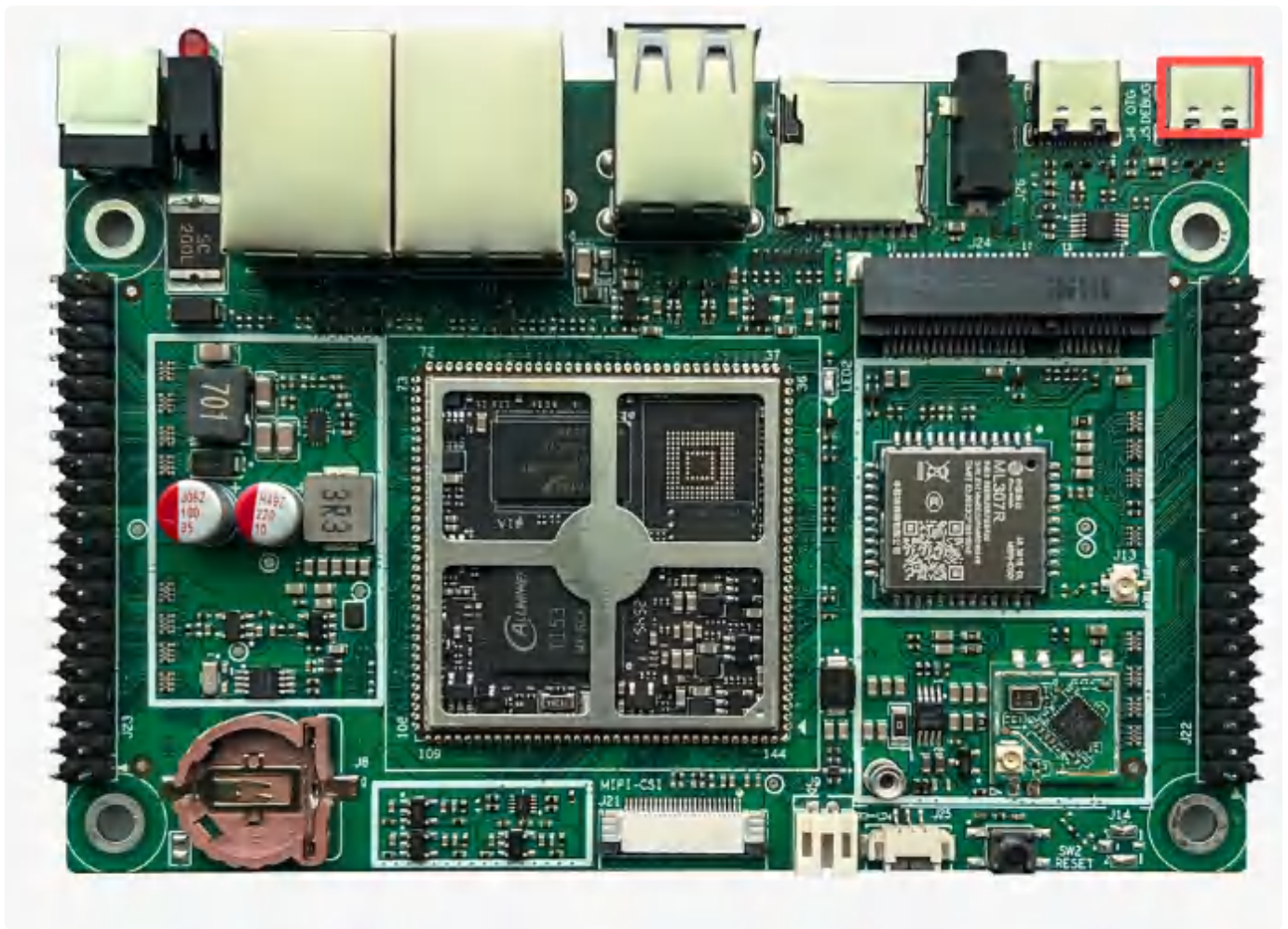


底板（背面）



1.2. 调试串口

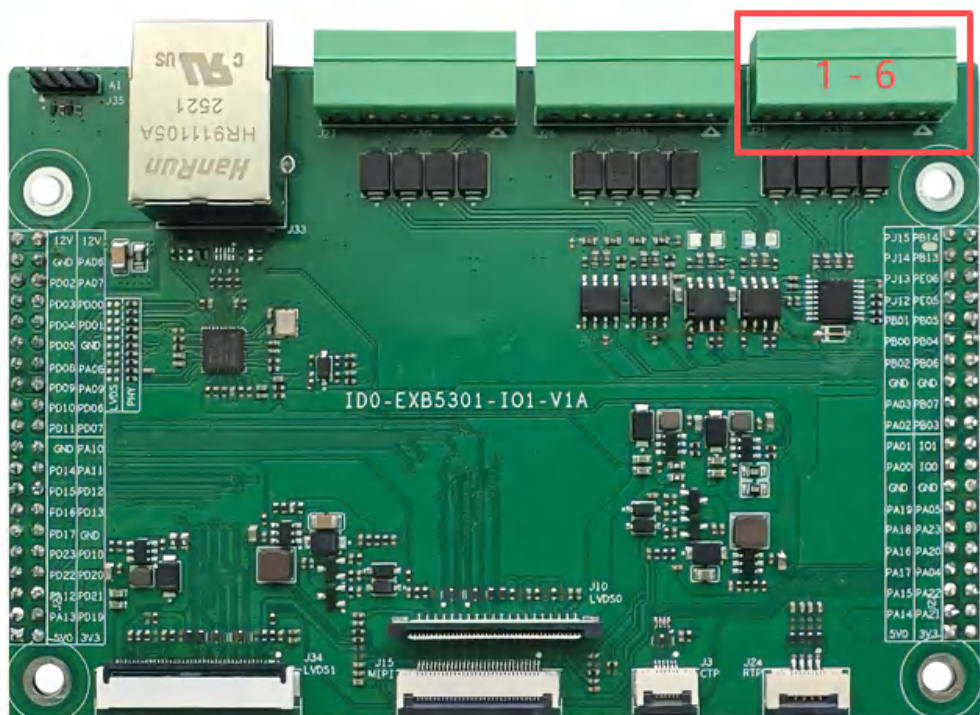
开发板debug串口位于J5处，波特率：115200



2. UART

主板共配置4路串口，包括2路RS232和2路RS485（除调试串口），如下所示。

2.1. RS232



序号	接口位置	电平类型	串口设备节点	备注
1	J25	RS232	/dev/ttyAS5	/
2		RS232	/dev/ttyAS7	/

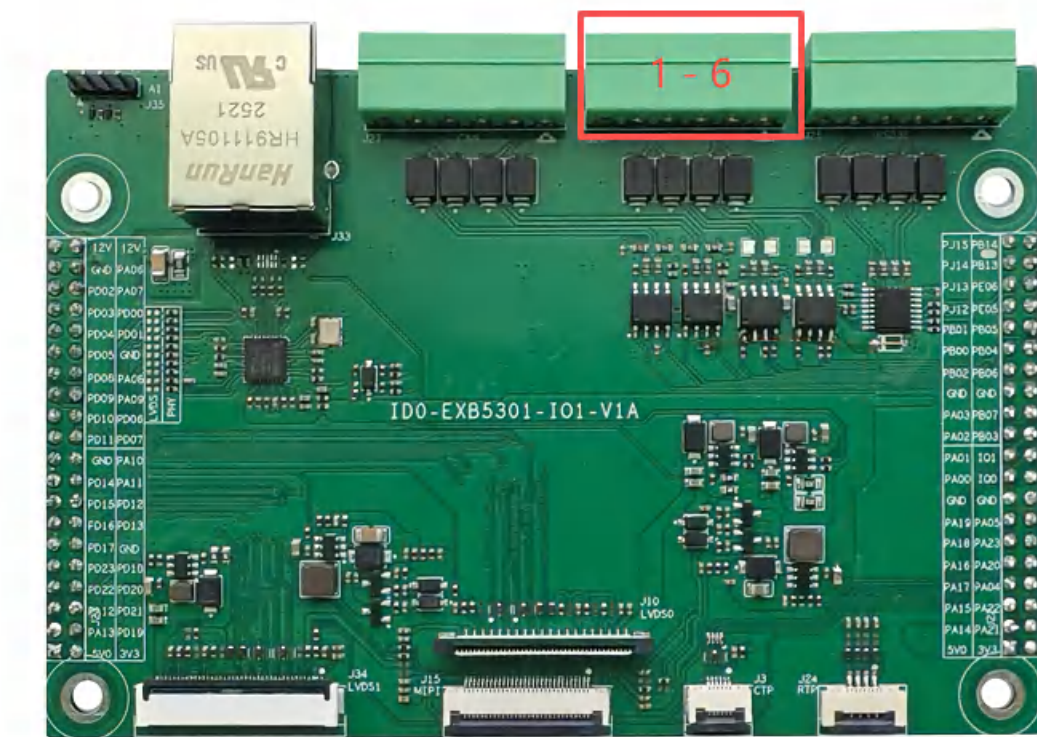
引脚定义：

序号	定义	电平/V	说明
1	GND	GND	电源地
2	TX7	/	RS232_TX
3	RX7	/	RS232_RX
4	TX5	/	RS232_TX
5	RX5	/	RS232_RX
6	5V	5V	电源5V

收发测试

```
▼ Bash |
1 # cat /dev/ttyAS5 &
2 ▾ [1] 453
3 # echo 123123 > /dev/ttyAS7
4 123123
5
6 # echo 123123 > /dev/ttyAS7
7 123123
8
9 # kill 453
10 # cat /dev/ttyAS7 &
11 ▾ [2] 454
12 ▾ [1] Terminated cat /dev/ttyAS5
13 # echo 123123 > /dev/ttyAS5
14 123123
15
16 # echo 123123 > /dev/ttyAS5
17 123123
18
19 # echo 123123 > /dev/ttyAS5
20 123123
21
22 #
```

2.2. RS485



序号	接口位置	电平类型	串口设备节点	备注
1	J26	RS485	/dev/ttyAS2	
2		RS485	/dev/ttyAS8	

引脚定义：

序号	定义	电平/V	说明
1	GND	GND	电源地
2	RS485_B8	/	RS485_B
3	RS485_A8	/	RS485_A
4	RS485_B2	/	RS485_B
5	RS485_A2	/	RS485_A

6	5V	5V	电源5V
---	----	----	------

rs485可以通过内置 `rs485_test` 测试demo进行短接验证，短接 `ttyAS2` 与 `ttyAS8`

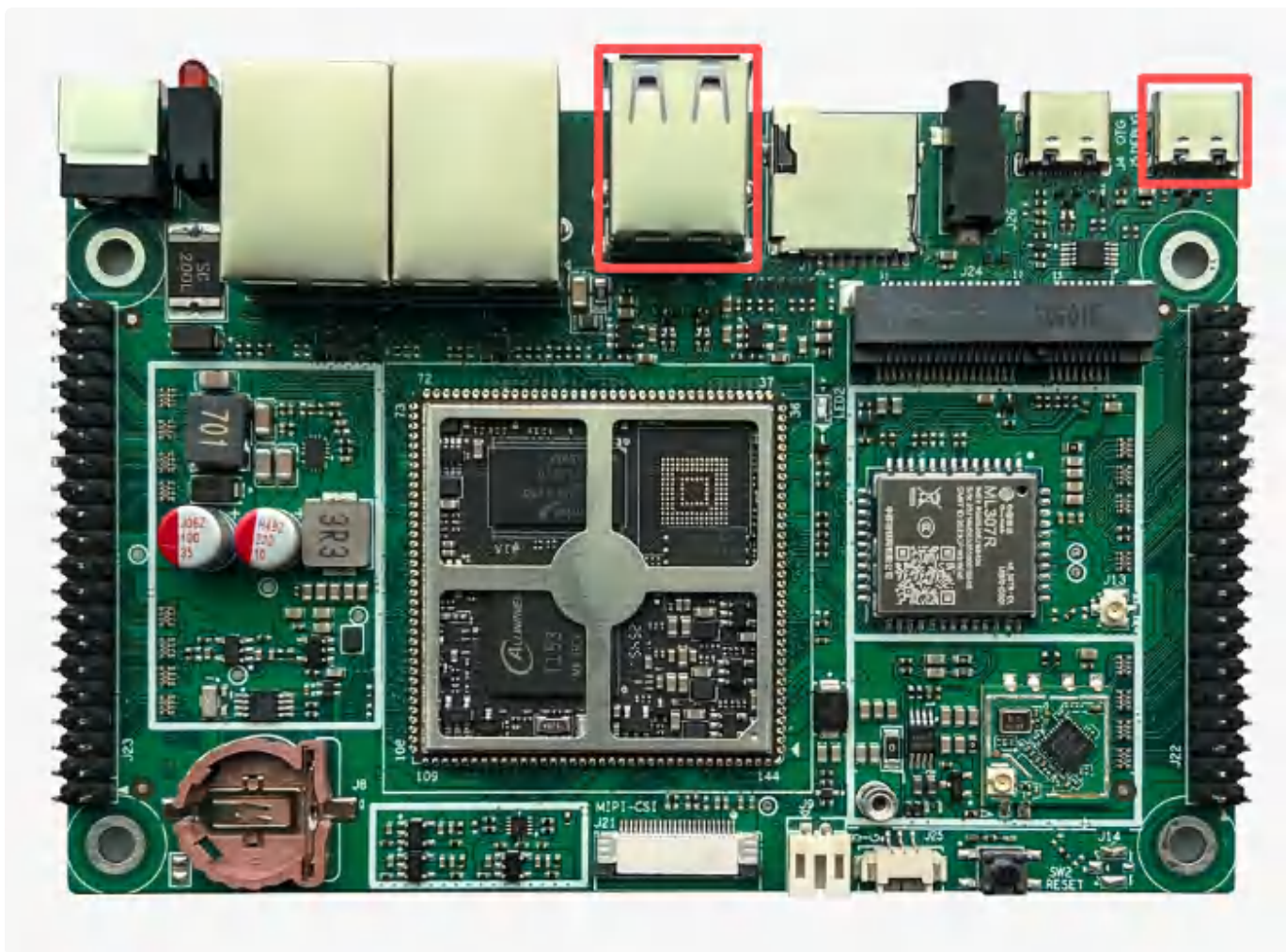
```

1  # rs485_test
2  usage: ./rs485_test /dev/ttyS2 /dev/ttyAS4
3  # 若出现 check okay 打印则说明测试验证通过
4  # rs485_test /dev/ttyAS2 /dev/ttyAS8
5  len=1
6  len=3
7  timeout
8  recv:31
9  32
10 33
11
12 check okay
13 # rs485_test /dev/ttyAS8 /dev/ttyAS2
14 len=1
15 len=3
16 timeout
17 recv:31
18 32
19 33
20
21 check okay
22
23 # 若出现以下 check fail 打印，则需检查硬件连接
24 # rs485_test /dev/ttyAS8 /dev/ttyAS2
25 timeout
26 recv:
27 check fail
28 #

```

3. USB

主板共配置2路USB座子，如下图所示：



序号	座子位置	类型
USB1	J4	OTG
USB2	J6-上	USB2.0 HOST
USB3	J6-下	USB2.0 HOST

接上U盘后，系统出现如下打印

```
▼ Bash |
1 [ 2487.378996] usb 1-1.2: new high-speed USB device number 9 using sunxi-ehci
2 [ 2487.533172] usb 1-1.2: New USB device found, idVendor=0951, idProduct=1666, bcdDevice= 2.00
3 [ 2487.533186] usb 1-1.2: New USB device strings: Mfr=2, Product=3, SerialNumber=4
4 [ 2487.533195] usb 1-1.2: Product: DataTraveler 3.0
5 [ 2487.533203] usb 1-1.2: Manufacturer: Kingston
6 [ 2487.533210] usb 1-1.2: SerialNumber: E0D55E6C0EC11691B8C1065B
7 [ 2487.534261] usb-storage 1-1.2:1.0: USB Mass Storage device detected
8 [ 2487.536615] scsi host0: usb-storage 1-1.2:1.0
```

通过 `fdisk -l` 命令可以查看识别出的usb设备

```
▼ Bash |
1 # fdisk -l
2 Found valid GPT with protective MBR; using GPT
3
4 .....
5
6 Disk /dev/sda: 29 GB, 30995907072 bytes, 60538881 sectors
7 3768 cylinders, 255 heads, 63 sectors/track
8 Units: sectors of 1 * 512 = 512 bytes
9
10 Device Boot StartCHS EndCHS StartLBA EndLBA Sectors Size
11 /dev/sda1 * 122,32,39 1023,254,63 1961984 59617280 57655297 27.4
12 /dev/sda2 1023,254,63 1023,254,63 59619328 60534783 915456 447
13 M 1b Hidden Win95 FAT32
```

通过 `mount` 手动将U盘挂载至rootfs

```
▼ Bash |
1 # mkdir /mnt/usb && mount /dev/sda1 /mnt/usb
```

Linux 内核自带的 ntfs 驱动 只能读，不能写（这是历史遗留问题，该驱动自 2004 年起基本停止开发）。

若需要挂载 `ntfs` 格式的U盘或sd卡，需要通过 `ntfs-3g` 命令进行挂载

查看NTFS格式U盘

Bash

```

1 # fdisk -l
2 Found valid GPT with protective MBR; using GPT
3 .....
4
5 Device  Boot StartCHS      EndCHS          StartLBA      EndLBA        Sectors  Siz
   e Id Type
6 /dev/sda1 *  0,1,2          1023,254,63      64  121077759  121077696  57.7
   G  7 HPFS/NTFS
7 Found valid GPT with protective MBR; using GPT
8
9 Disk /dev/mmcblk1: 249737216 sectors, 3158M
10 Logical sector size: 512
11 Disk identifier (GUID): 6c411a4e-ad3e-4594-8831-15292a650574
12 Partition table holds up to 128 entries
13 First usable sector is 34, last usable sector is 249737182
14

```

挂载NTFS格式U盘

Bash

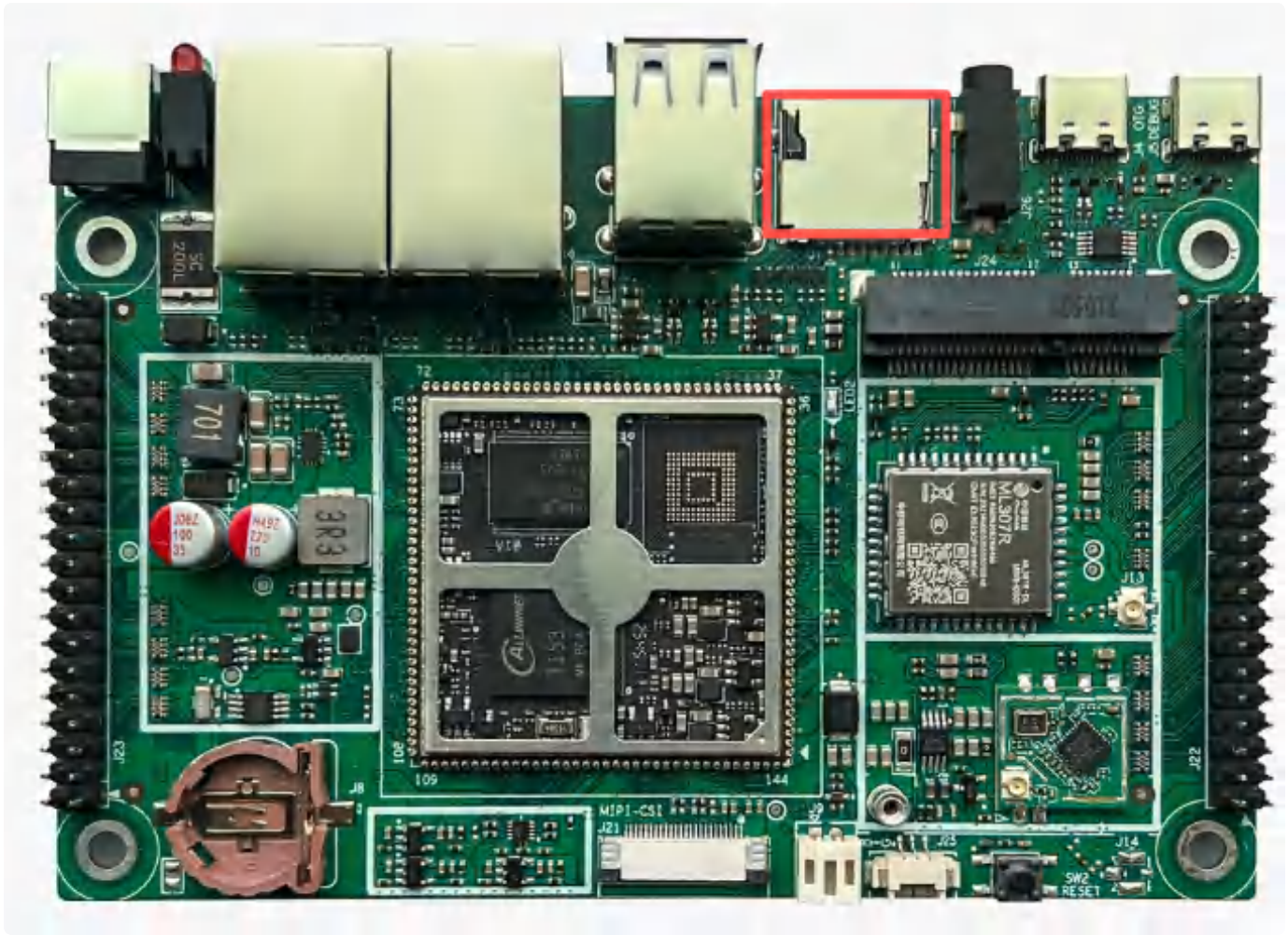
```

1 ## 通过mount命令挂载可以看到，mount尝试将其挂在为exFAT失败后，以ntfs格式进行挂载
2 $ mount /dev/sda1 /mnt/usb/
3 [ 210.471276] exFAT-fs (sda1): invalid fs_name
4 [ 210.476039] exFAT-fs (sda1): failed to read boot sector
5 [ 210.481914] exFAT-fs (sda1): failed to recognize exfat type
6 [ 210.546830] ntfs: volume version 3.1.
7 ## 写入失败
8 $ touch /mnt/usb/123123.txt
9 touch: /mnt/usb/123123.txt: Permission denied
10 #
11
12 ## 通过ntfs-3g命令进行挂载
13 $ ntfs-3g /dev/sda1 /mnt/usb/
14 $ touch /mnt/usb/123123.txt
15 $ echo 123123 > /mnt/usb/123123.txt
16 $ cat /mnt/usb/123123.txt
17 123123

```

4. Micro SD

主板共配置一路Micro SD接口，如下图所示：



插入SD卡后，默认挂载到/mnt/sdcard目录，如下所示：

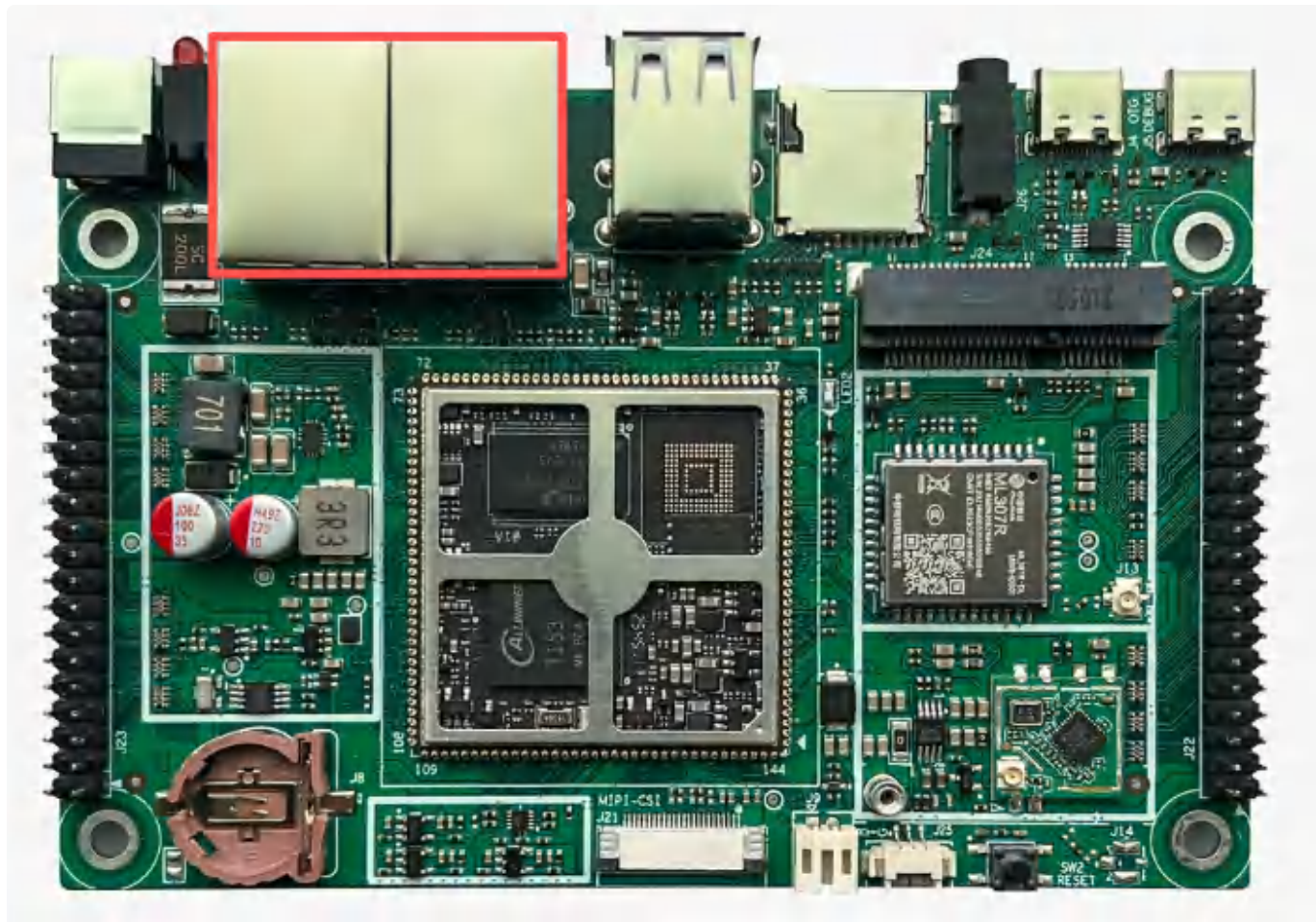
▼ Shell

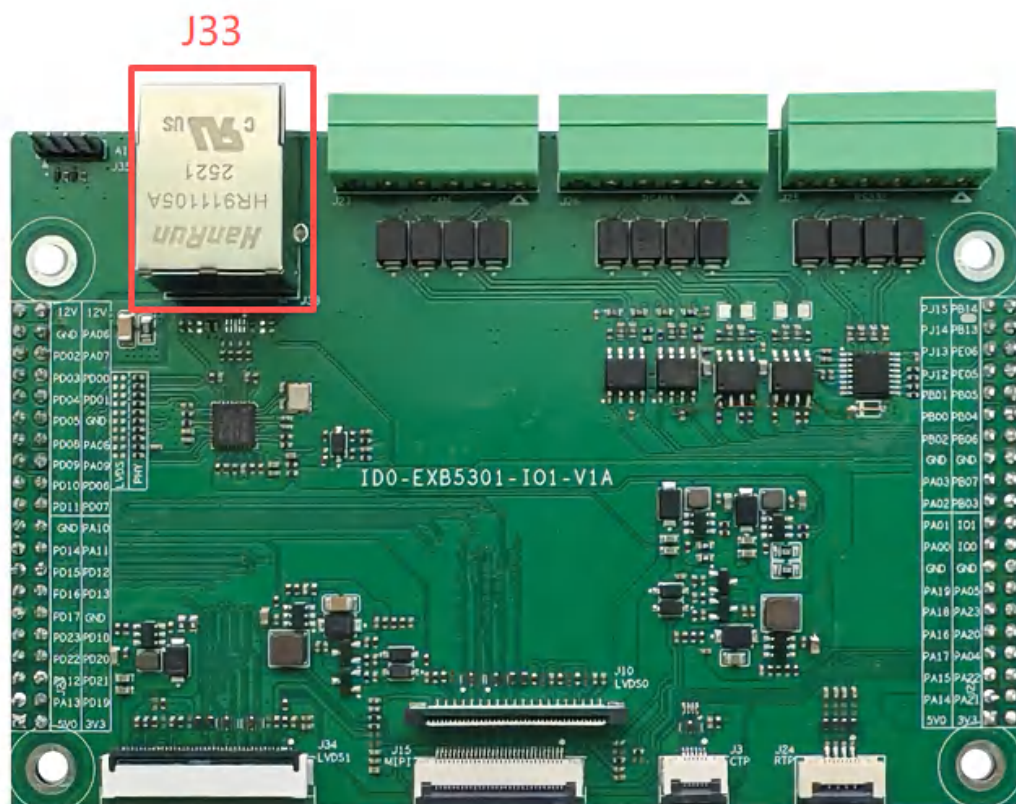
```
1 # mount
2 ...
3 /dev/mmcblk0p1 on /mnt/sdcard type vfat (rw,nodev,noexec,noatime,nodiratime
,fmask=0022,dmask=0022,codepage=437,iocharset=iso8859-1,shortname=mixed,err
ors=remount-ro)
```

测试项目	要求	测试结果
挂载SD	可以自动挂载fat32格式sd卡	✓
	可以手动挂载NTFS的sd卡	✓

5. Ethernet

主板共配置1路千兆以太网接口和2路百兆以太网接口，如下图所示：





序号	接口位置	速率	网络节点
1	J11	百兆	eth0
2	J10	千兆	eth1
3	J33	百兆	eth2

系统默认开启DHCP服务，动态获取IP。

▼ Shell

```
1 # ifconfig eth0
2 eth0      Link encap:Ethernet  HWaddr 7A:1F:F2:40:0F:F0
3            inet addr:192.168.0.12  Bcast:192.168.0.255  Mask:255.255.255.0
4            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
5            RX packets:25 errors:0 dropped:0 overruns:0 frame:0
6            TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
7            collisions:0 txqueuelen:1000
8            RX bytes:3358 (3.2 KiB)  TX bytes:1368 (1.3 KiB)
9            Interrupt:57 Base address:0xe000
```

可通过ifconfig指令设置临时静态IP，命令如下：

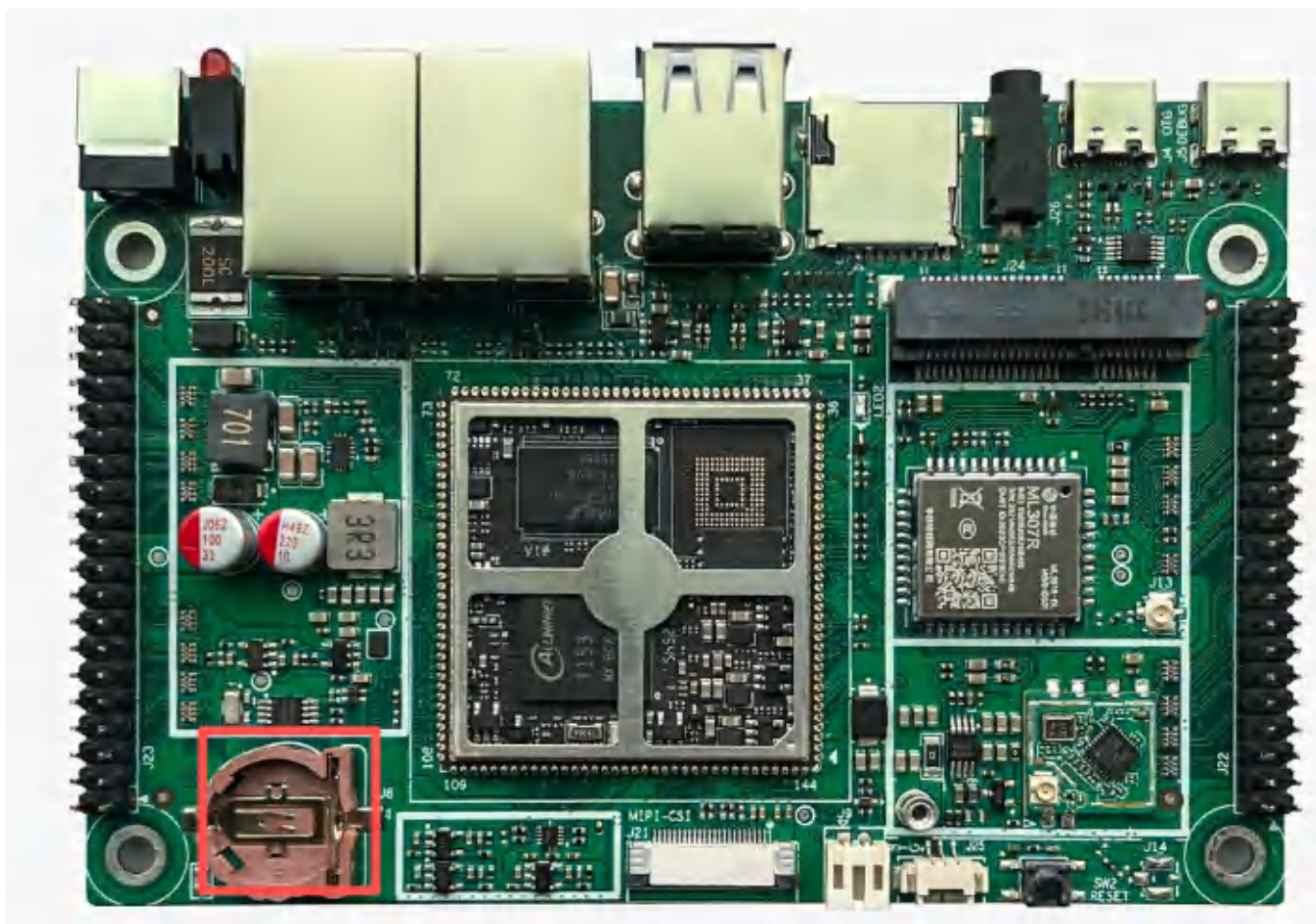
▼ Shell

```
1 # ifconfig eth0 192.168.3.123
```

注：eth2与 mipi/lvds0 硬件通路复用，使用前请先确定硬件是否支持。

6. RTC

主板配置了1路rtc接口，如下图所示：



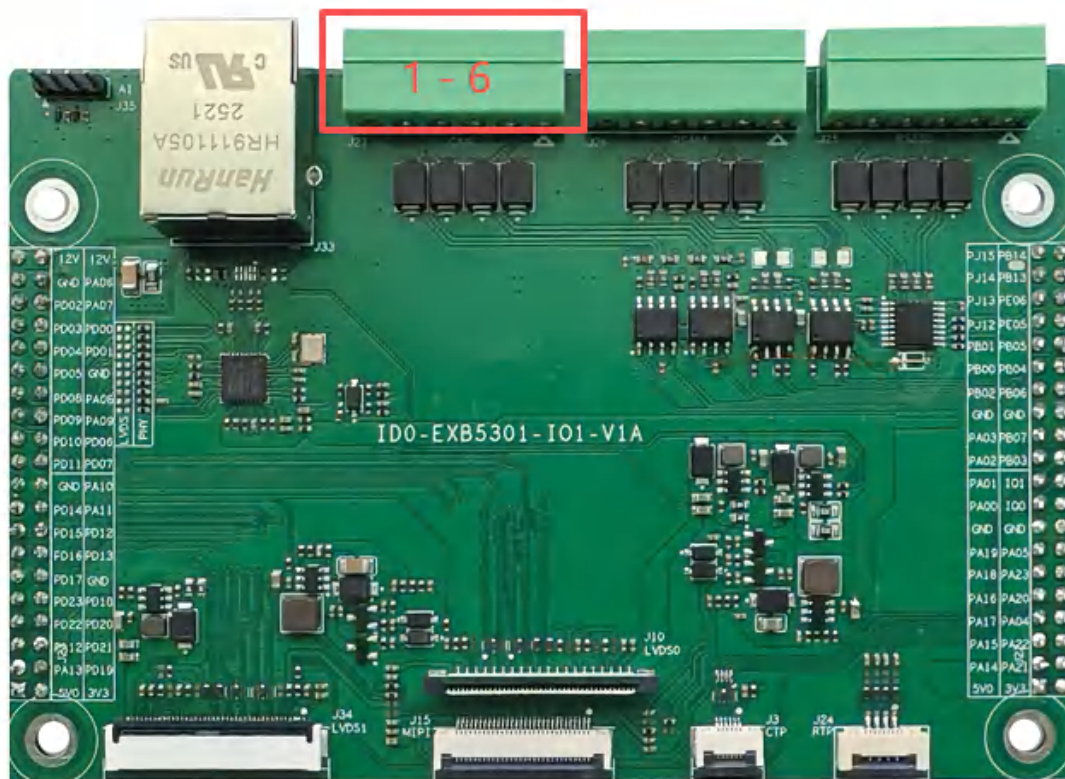
rtc时间设置方法：

Bash

```
1  #设置系统时间
2  $ date -s "2024-10-09 14:02:30"
3
4  #将rtc时间调整为与目前的系统时间一致
5  $ hwclock -w -f /dev/rtc1
6
7  #获取硬件rtc当前时间（断电重启读取时间没有太大偏差）
8  $ hwclock -r -f /dev/rtc1
9  2024-10-09 14:02:35.945604+00:00
```

7. CAN

主板配置了2路can接口，如下图所示：



序号	接口位置	电平类型	串口设备节点	备注
1	J27	CAN	can1	/
2		CAN	can0	/

引脚定义:

序号	定义	电平/V	说明
1	GND	GND	电源地
2	CAN0_L	/	CAN_L
3	CAN0_H	/	CAN_H
4	CAN1_L	/	CAN_L
5	CAN1_H	/	CAN_H
6	5V	5V	电源5V

查看can节点，以can1为例：

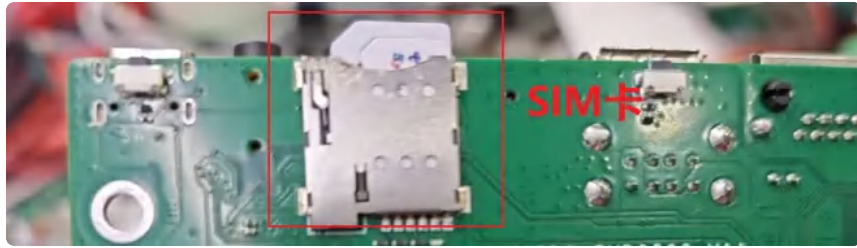
```
▼ Shell |
1 # ifconfig can0
2 can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
3           NOARP  MTU:16  Metric:1
4           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
5           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
6           collisions:0 txqueuelen:10
7           RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
8           Interrupt:49
```

数据收发，以can1为例：

```
▼ Shell |
1 #检查can设备
2 $ ip link show
3
4 #在收发端关闭can0设备
5 $ ip link set can0 down
6
7 #设置仲裁段1M波特率，数据段1M波特率
8 $ ip link set can0 type can bitrate 1000000 loopback off
9 [ 8846.889995] sunxi-can 453c800.can0: PM runtime resume.
10 [ 8846.895843] sunxi-can 453c800.can0: PM runtime suspend.
11
12 #在收发端打开can0设备
13 $ ip link set can0 up
14
15 #在接收端执行candump,阻塞等待报文
16 $ candump can0
17
18 #在发送端执行cansend,发送报文
19 $ cansend can0 123#1122334455667788
20
21 #检查是否启用成功
22 $ ip link show can0
```

8. 4G模块

主板配置了1路中移4G模块ML307R，如下图所示：



拨号上网：

```
▼ Shell |
1  cat /dev/ttyUSB2 &
2
3  #设置拨号协议为RNDIS
4  echo -e "AT+MDIALUPCFG="mode",0" > /dev/ttyUSB2
5
6  #设置拨号协议为ECM
7  echo -e "AT+MDIALUPCFG="mode",1" > /dev/ttyUSB2
8
9  #关闭回显
10 echo -e 'ATE0\r\n' > /dev/ttyUSB2
11
12 #查询PDP上下文配置
13 echo -e 'AT+CGDCONT?\r\n' > /dev/ttyUSB2
14
15 #查询驻网状态
16 echo -e 'AT+COPS?\r\n' > /dev/ttyUSB2
17
18 #设置自动拨号命令
19 echo -e 'AT+MDIALUP=1,1\r\n' > /dev/ttyUSB2
20
21 #查询拨号状态
22 echo -e 'AT+MDIALUP?\r\n' > /dev/ttyUSB2
```

查看网络节点：

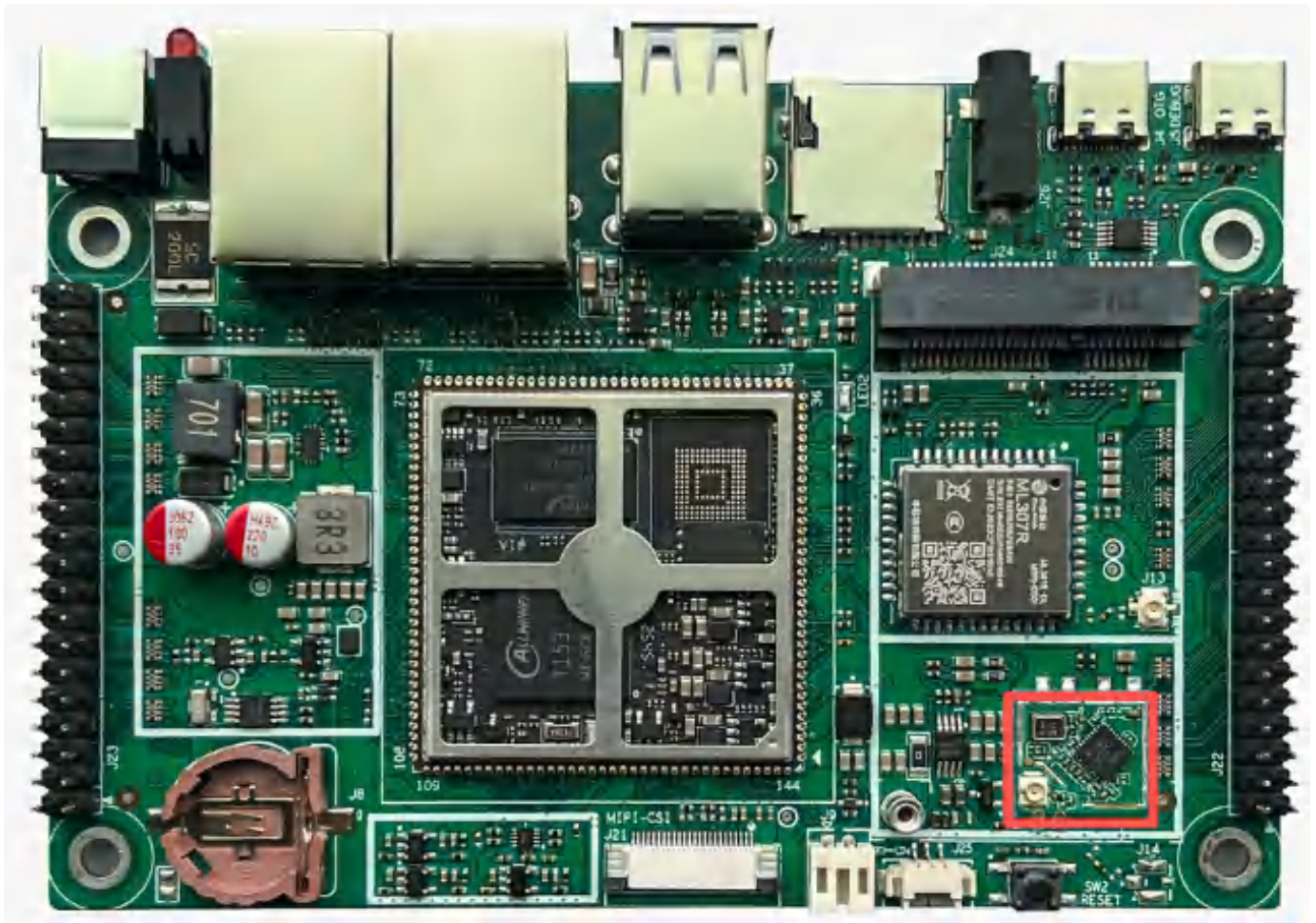
```
▼ Shell |
1  # 如果是mipi屏幕的固件，4g节点则为eth2
2  $ ifconfig eth3
3  eth3      Link encap:Ethernet  HWaddr AC:0C:29:A3:9B:6D
4            inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
5            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
6            RX packets:4 errors:0 dropped:0 overruns:0 frame:0
7            TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
8            collisions:0 txqueuelen:1000
9            RX bytes:928 (928.0 B)  TX bytes:1026 (1.0 KiB)
```

注意：

- 1.ML307R 4G模块支持RNDIS和ECM模块，两种模式选择其一即可。
- 2.ML307R 4G模块拨号上网后默认获取局域网内IP地址。

9. 星闪模块

主板配置了1路星闪模块，如下图所示：



9.1. WIFI

STA连接方法：

星闪wifi.so加载起来后，会打开Featureid0和wlan0两个网络节点，Featureid0会影响wlan0的ip获取，需要关闭，命令如下。

```
▼ Bash |  
1  ifconfig Featureid0 down
```

vi /etc/wpa_supplicant.conf

账号密码配置：

```
1  ctrl_interface=/var/run/wpa_supplicant
2  ap_scan=1
3  update_config=1
4
5  network={
6      ssid="SSID"
7      psk="PASSWORD"
8      key_mgmt=WPA-PSK
9  }
```

ssid: 账号。

psk: 密码。

若需要连接WPA3加密的WiFi，可以参考如下配置

```
1  ctrl_interface=/var/run/wpa_supplicant
2  ap_scan=1
3  update_config=1
4
5  network={
6      ssid="CS_2.4G_WPA3"
7      sae_password="CS123123"
8      pairwise=CCMP TKIP
9      group=CCMP TKIP
10     key_mgmt=SAE WPA-PSK
11     scan_ssid=1
12     ieee80211w=2
13 }
```

连接方法：

```
1  killall wpa_supplicant
2  wpa_supplicant -Dnl80211 -c /etc/wpa_supplicant.conf -i wlan0 &
```

获取IP:

```
▼ Bash |
1 dhclient wlan0
2 udhcpc: started, v1.36.1
3 udhcpc: broadcasting discover
4 udhcpc: broadcasting select for 192.168.74.160, server 192.168.74.123
5 udhcpc: lease of 192.168.74.160 obtained from 192.168.74.123, lease time 3
  599
6 deleting routers
7 adding dns 192.168.74.123
8
9 ifconfig wlan0
10 wlan0      Link encap:Ethernet  HWaddr 78:22:88:85:63:96
11            inet addr:192.168.74.160  Bcast:192.168.74.255  Mask:255.255.25
  5.0
12            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
13            RX packets:77 errors:0 dropped:0 overruns:0 frame:0
14            TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
15            collisions:0 txqueuelen:1000
16            RX bytes:6190 (6.0 KiB)  TX bytes:11915 (11.6 KiB)
```

测试:

```
▼ Bash |
1 ping www.baidu.com -I wlan0
2 PING www.baidu.com (103.235.46.102): 56 data bytes
3 64 bytes from 103.235.46.102: seq=0 ttl=48 time=266.668 ms
4 64 bytes from 103.235.46.102: seq=1 ttl=48 time=233.898 ms
5 64 bytes from 103.235.46.102: seq=2 ttl=48 time=276.997 ms
6 64 bytes from 103.235.46.102: seq=3 ttl=48 time=229.841 ms
```

9.2. Bluetooth

打开蓝牙设备:

```
▼ Shell |
1 $ hciconfig hci0 up
2 $ bluetoothd &
```

扫描蓝牙设备：

```
▼ Shell |
1  $ hciconfig hci0 iscan
2
3  $ bluetoothctl
4  Agent registered
5  ▾ [CHG] Controller 10:BB:F3:56:44:55 Pairable: yes
6  ▾ [bluetooth]# scan on
7  Discovery started
8  ▾ [CHG] Controller 10:BB:F3:56:44:55 Discovering: yes
9  ▾ [CHG] Device 24:4C:AB:09:98:A6 RSSI: -92
10 ...
11 ▾ [NEW] Device 7C:C1:80:09:DD:6C AirPods - Find My
12 ...
13
```

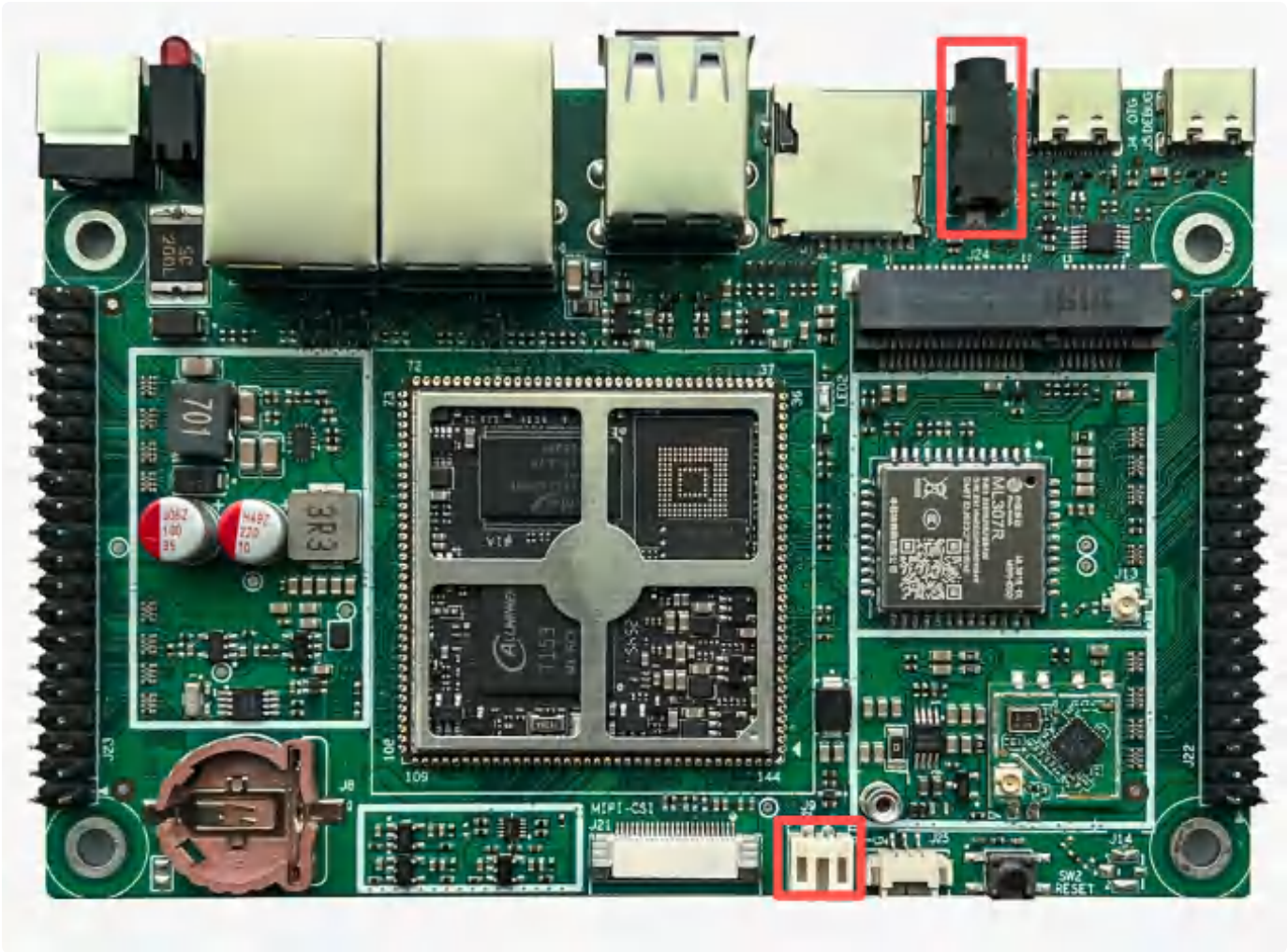
通过mac配对蓝牙设备：

```
▼ Shell |
1  ▾ [bluetooth]# trust 7C:C1:80:09:DD:6C
2  ▾ [CHG] Device 7C:C1:80:09:DD:6C Trusted: yes
3  Changing 7C:C1:80:09:DD:6C trust succeeded
4  ▾ [bluetooth]# pair 7C:C1:80:09:DD:6C
5  Attempting to pair with 7C:C1:80:09:DD:6C
6  ▾ [CHG] Device 7C:C1:80:09:DD:6C Connected: yes
7  ▾ [CHG] Device 7C:C1:80:09:DD:6C Name: AirPods
8  ▾ [CHG] Device 7C:C1:80:09:DD:6C Alias: AirPods
9  ▾ [CHG] Device 7C:C1:80:09:DD:6C Modalias: bluetooth:v004Cp2013dB087
10 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 00001000-0000-1000-8000-00805f9b34fb
11 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 0000110b-0000-1000-8000-00805f9b34fb
12 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
13 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
14 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 0000111e-0000-1000-8000-00805f9b34fb
15 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 00001200-0000-1000-8000-00805f9b34fb
16 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 74ec2172-0bad-4d01-8f77-997b2be0722a
17 ▾ [CHG] Device 7C:C1:80:09:DD:6C ServicesResolved: yes
18 ▾ [CHG] Device 7C:C1:80:09:DD:6C Paired: yes
19 Pairing successful
20 ▾ [AirPods - Find My]# exit
```

9.3. SLE

10. 声卡

主板配置的声卡接口，如下图所示：



序号	接口位置	功能
1	J9	Speaker
2	J26	Line Out

声卡节点

```

1  #声卡节点
2  # aplay -l
3  **** List of PLAYBACK Hardware Devices ****
4  card 0: audiocodec [audiocodec], device 0: sunxi-snd-plat-audio-sunxi-snd-
   codec 2030000.codec-0 []
5      Subdevices: 1/1
6      Subdevice #0: subdevice #0
7
8

```

声卡开关

```

1  #声卡控制节点
2  # amixer -c 0 controls
3  numid=2,iface=MIXER,name='DAC DRC Mode'
4  numid=3,iface=MIXER,name='DAC HPF Mode'
5  numid=6,iface=MIXER,name='DAC Output Select'
6  numid=4,iface=MIXER,name='DAC Volume'
7  numid=5,iface=MIXER,name='DACL Volume'
8  numid=7,iface=MIXER,name='LINEOUTL Gain'
9  numid=8,iface=MIXER,name='LINEOUTL Switch'
10 numid=9,iface=MIXER,name='SPK Switch'
11 numid=1,iface=MIXER,name='tx hub mode'
12
13 #打开耳机开关
14 # amixer -c 0 cset numid=8,iface=MIXER,name='LINEOUTL Switch' on
15 numid=8,iface=MIXER,name='LINEOUTL Switch'
16     ; type=BOOLEAN,access=rw-----,values=1
17     : values=on
18
19 #打开功放开关
20 # amixer -c 0 cset numid=9,iface=MIXER,name='SPK Switch' on
21 numid=9,iface=MIXER,name='SPK Switch'
22     ; type=BOOLEAN,access=rw-----,values=1
23     : values=on
24 #
25

```

注：由于喇叭是由lineout音频信号经过功放转换而来，所以在测试喇叭音频时，需先打开耳机开关。

音频播放

```
▼ Shell |
1  #音频播放
2  # aplay -Dhw:0,0 ./test_mono.wav
3  Playing WAVE './test_mono.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
```

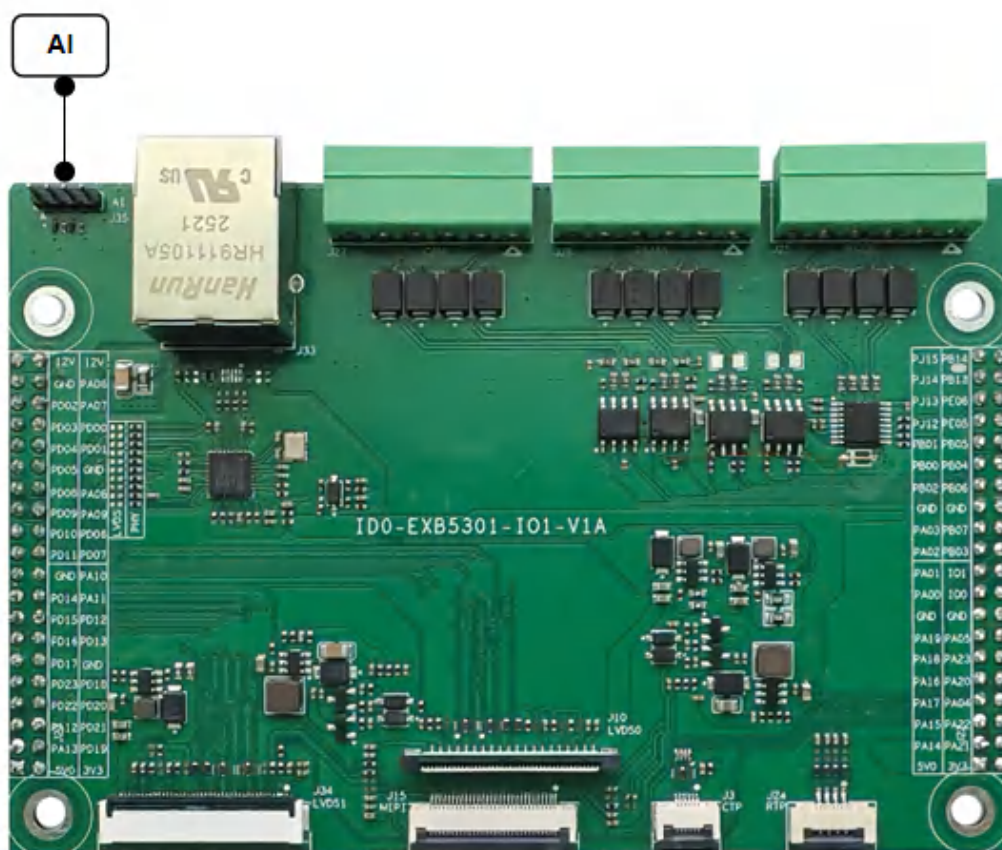
注：由于T153默认声卡只支持单声道音频，播放音频若出现 `aplay:set_params:1358:channels count non available` 等相关报错信息，可通过 `ffmpeg` 将其转换成默认支持的音频。

```
▼ Shell |
1  $ ffmpeg -i ./8k16bpsStereo.wav -ac 1 8k16bpsStereo_mono.wav
```

`8k16bpsStereo.wav` 为音频源文件，`8k16bpsStereo_mono.wav` 为转换后输出的音频文件（开发板系统默认不带ffmpeg工具，需在其他平台上转换）

11. ADC

拓展板带有两路adc捕获，位于拓展板J35处，如下图



四路引脚从左往右依次为 AI1+, AI1-, AI2+, AI2-

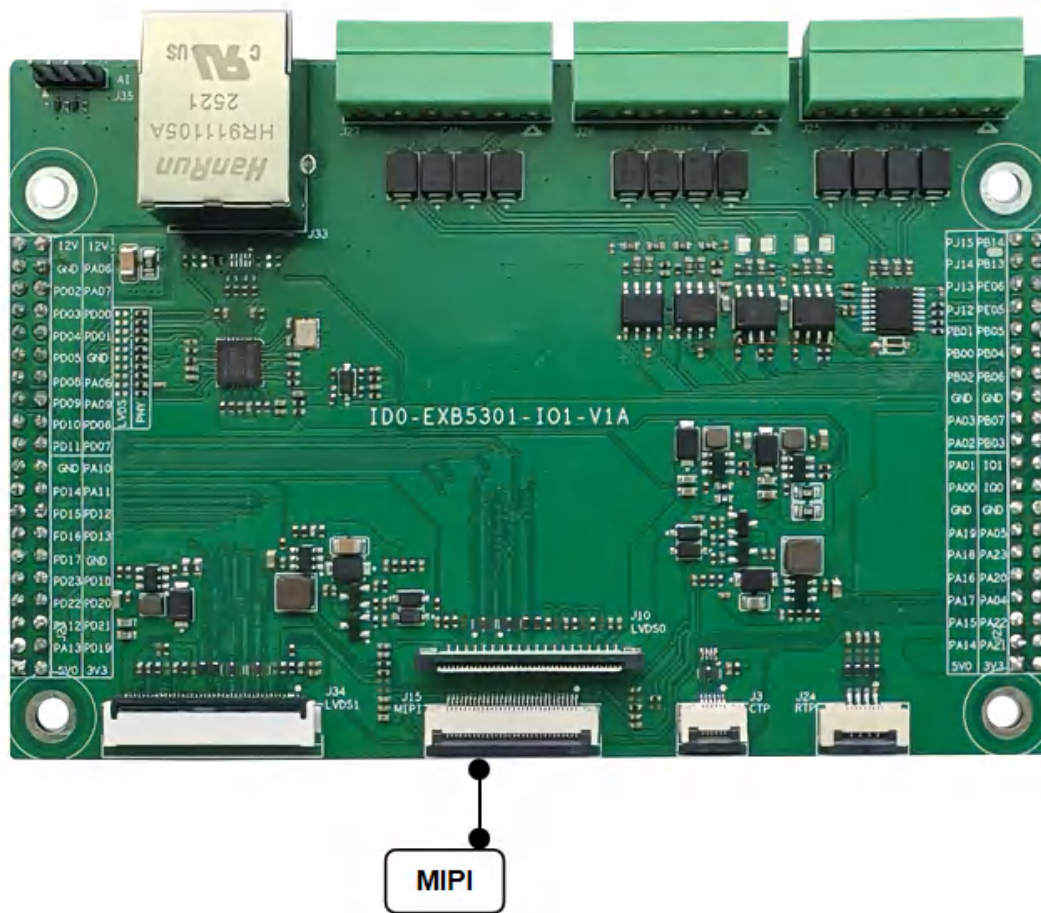
▼ AI捕获 (单位为mv)

Shell

```
1  $ cd /sys/class/gpadc/gpadc_chip0
2  $ echo gpadc6,1 > status
3  $ echo gpadc7,1 > status
4
5  # 捕获AI1的值
6  $ echo 6 > data ;cat data
7  gpadc0-channel6 voltage data is 1792
8  # 捕获AI2的值
9  $ echo 7 > data ;cat data
10 gpadc0-channel7 voltage data is 1791
```

12. 显示接口

12.1. MIPI

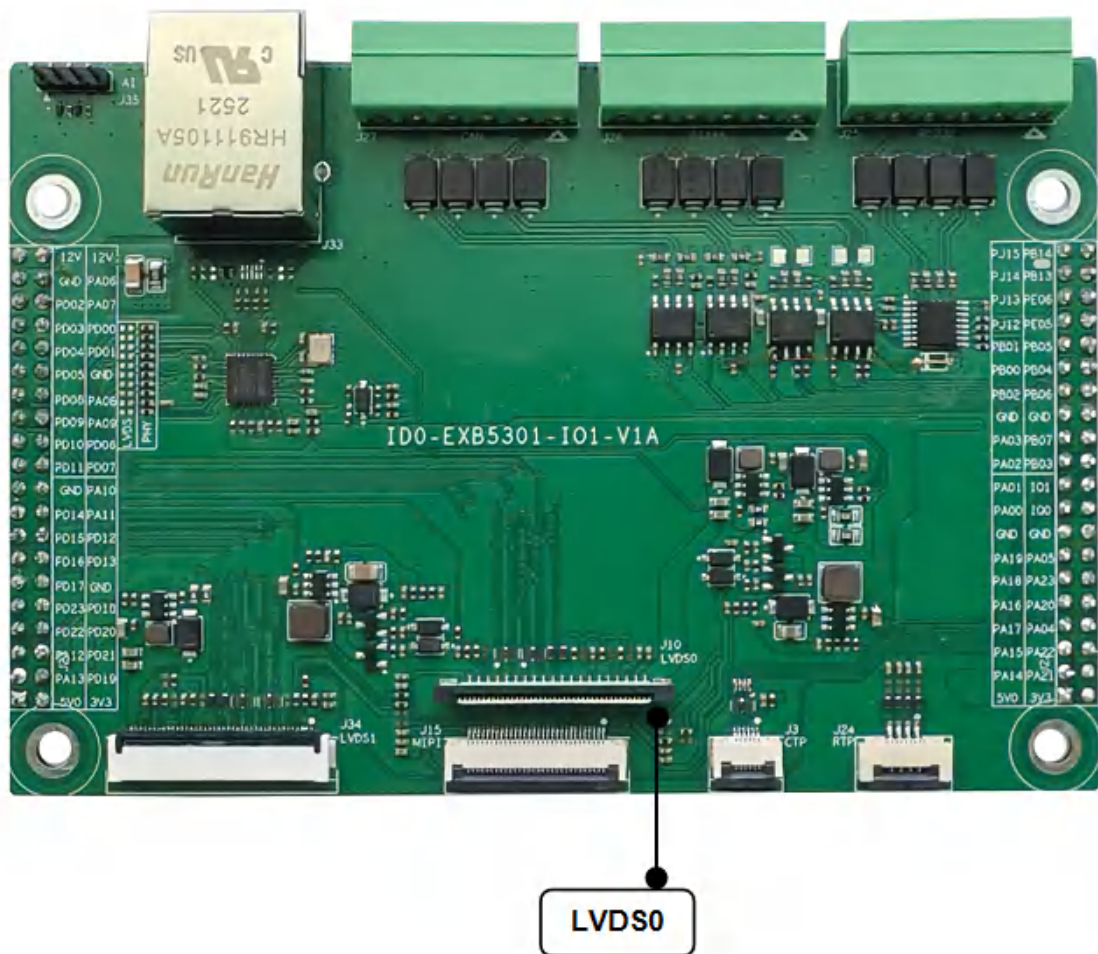


分辨率：1024x600

默认屏幕是否支持触摸：是

注：eth2与 mipi/lvds0 硬件通路复用，使用前请先确定硬件是否支持。

12.2. LVDS0屏幕

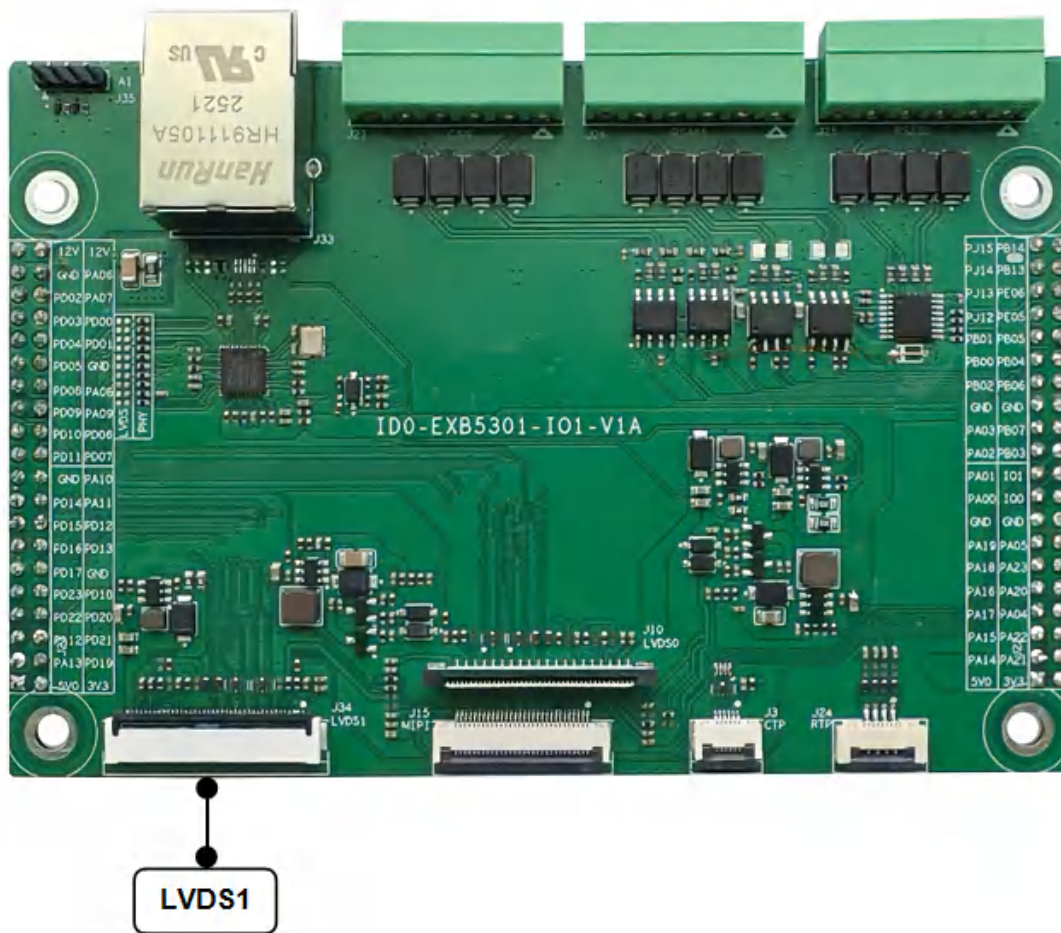


分辨率：1024x600

默认屏幕是否支持触摸：否

注：eth2与 mipi/lvds0 硬件通路复用，使用前请先确定硬件是否支持。

12.3. LVDS1屏幕



分辨率：1024x600

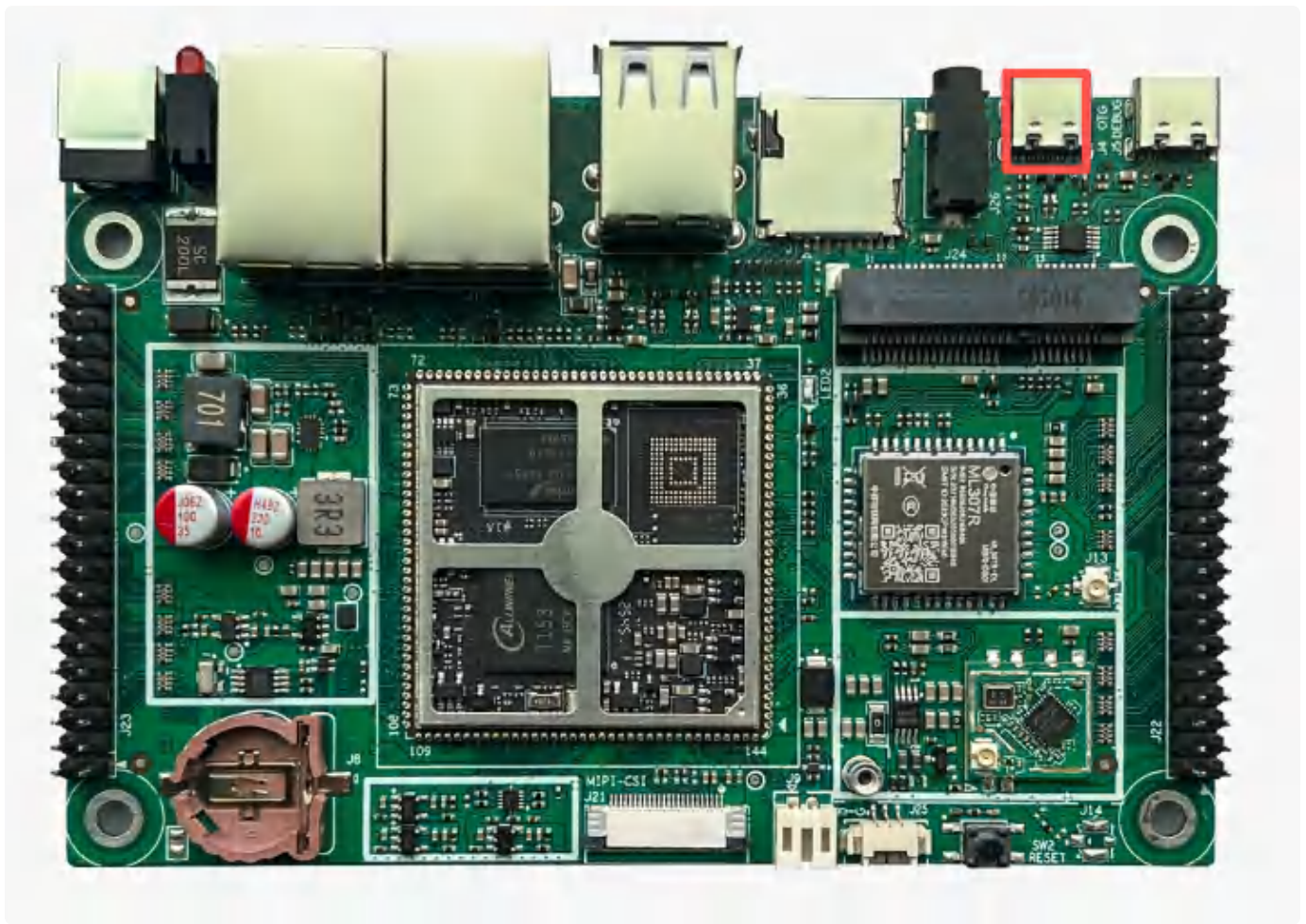
默认屏幕是否支持触摸：否

Ubuntu使用手册

1. 调试接口

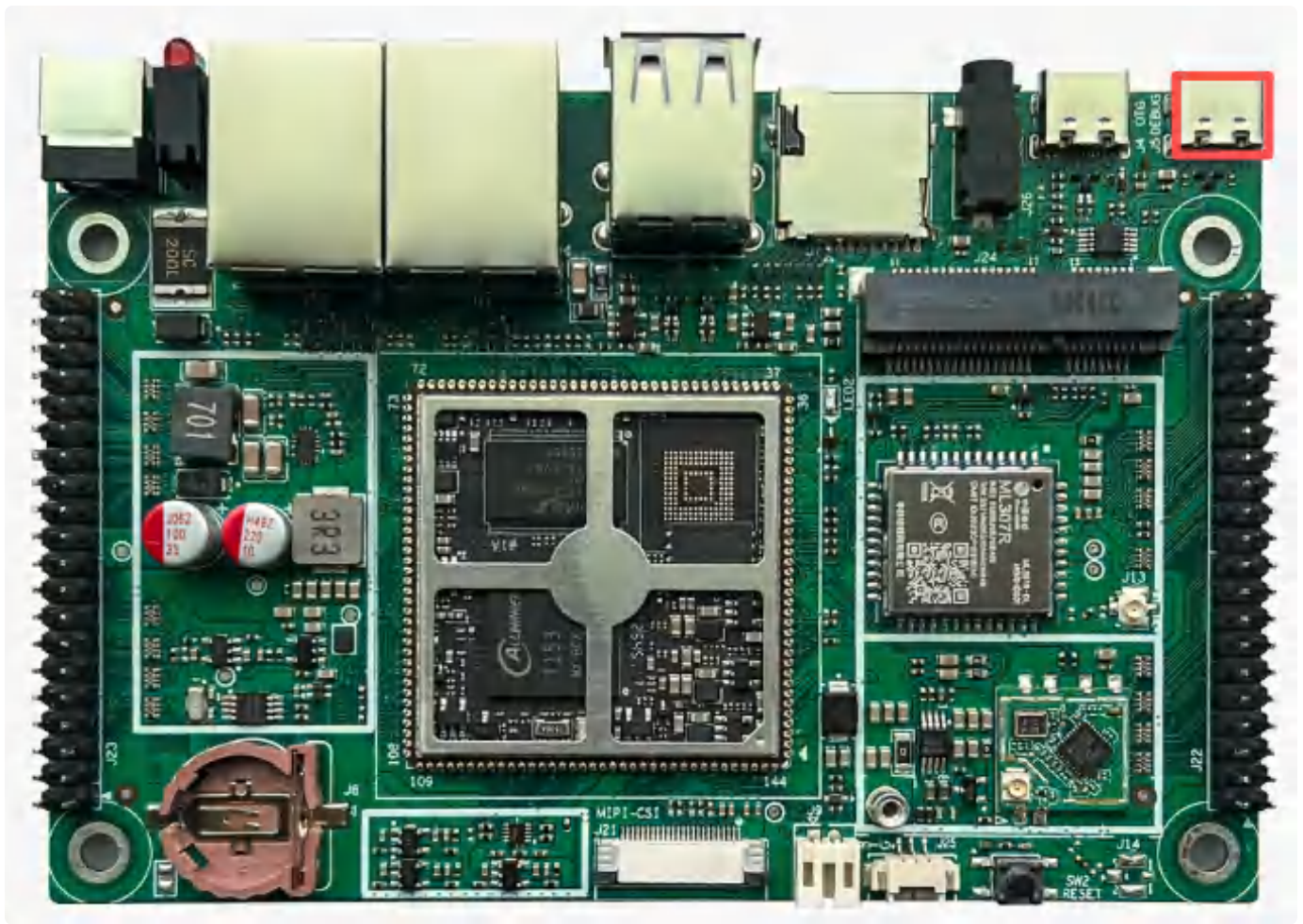
1.1. adb

开发板adb调试口位于主板J4处



1.2. 调试串口

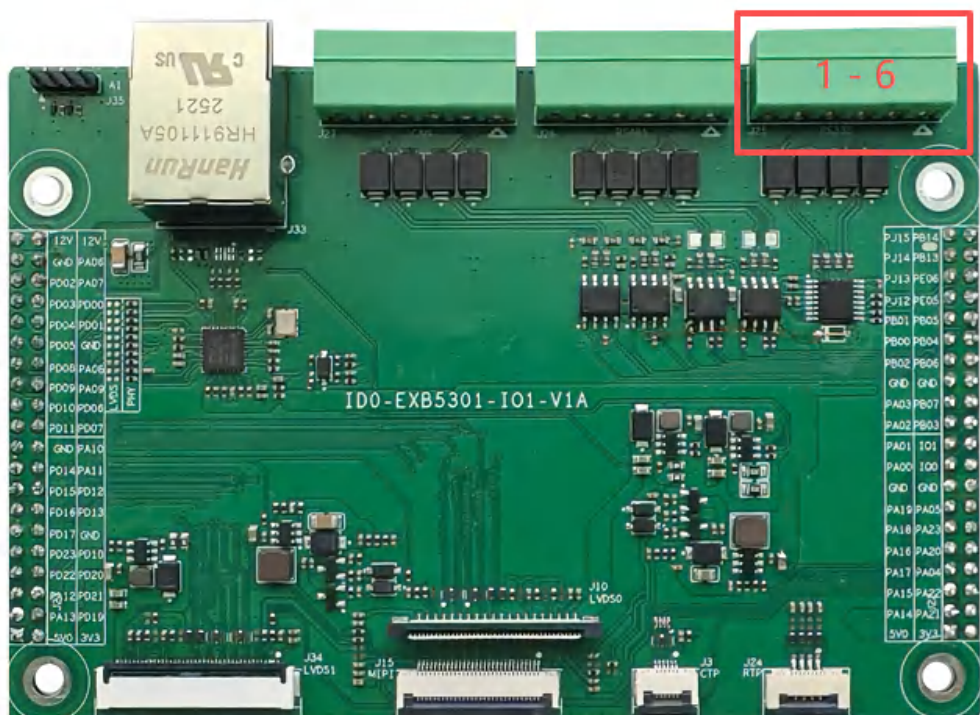
开发板debug串口位于J5处，波特率：115200



2. UART

主板共配置4路串口，包括2路RS232和2路RS485（除调试串口），如下所示。

2.1. RS232



序号	接口位置	电平类型	串口设备节点	备注
1	J25	RS232	/dev/ttyAS5	/
2		RS232	/dev/ttyAS7	/

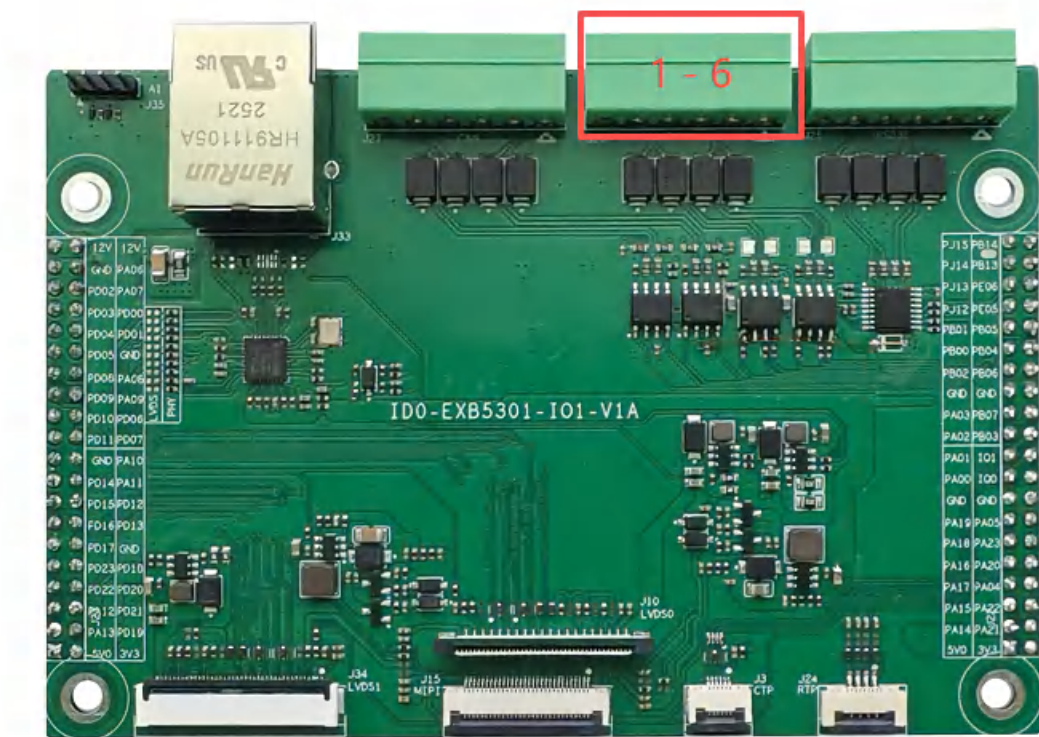
引脚定义：

序号	定义	电平/V	说明
1	GND	GND	电源地
2	TX7	/	RS232_TX
3	RX7	/	RS232_RX
4	TX5	/	RS232_TX
5	RX5	/	RS232_RX
6	5V	5V	电源5V

收发测试

```
▼ Bash |
1  # cat /dev/ttyAS5 &
2  [1] 453
3  # echo 123123 > /dev/ttyAS7
4  123123
5
6  # echo 123123 > /dev/ttyAS7
7  123123
8
9  # kill 453
10 # cat /dev/ttyAS7 &
11 [2] 454
12 [1] Terminated          cat /dev/ttyAS5
13 # echo 123123 > /dev/ttyAS5
14 123123
15
16 # echo 123123 > /dev/ttyAS5
17 123123
18
19 # echo 123123 > /dev/ttyAS5
20 123123
21
22 #
```

2.2. RS485



序号	接口位置	电平类型	串口设备节点	备注
1	J26	RS485	/dev/ttyAS2	
2		RS485	/dev/ttyAS8	

引脚定义：

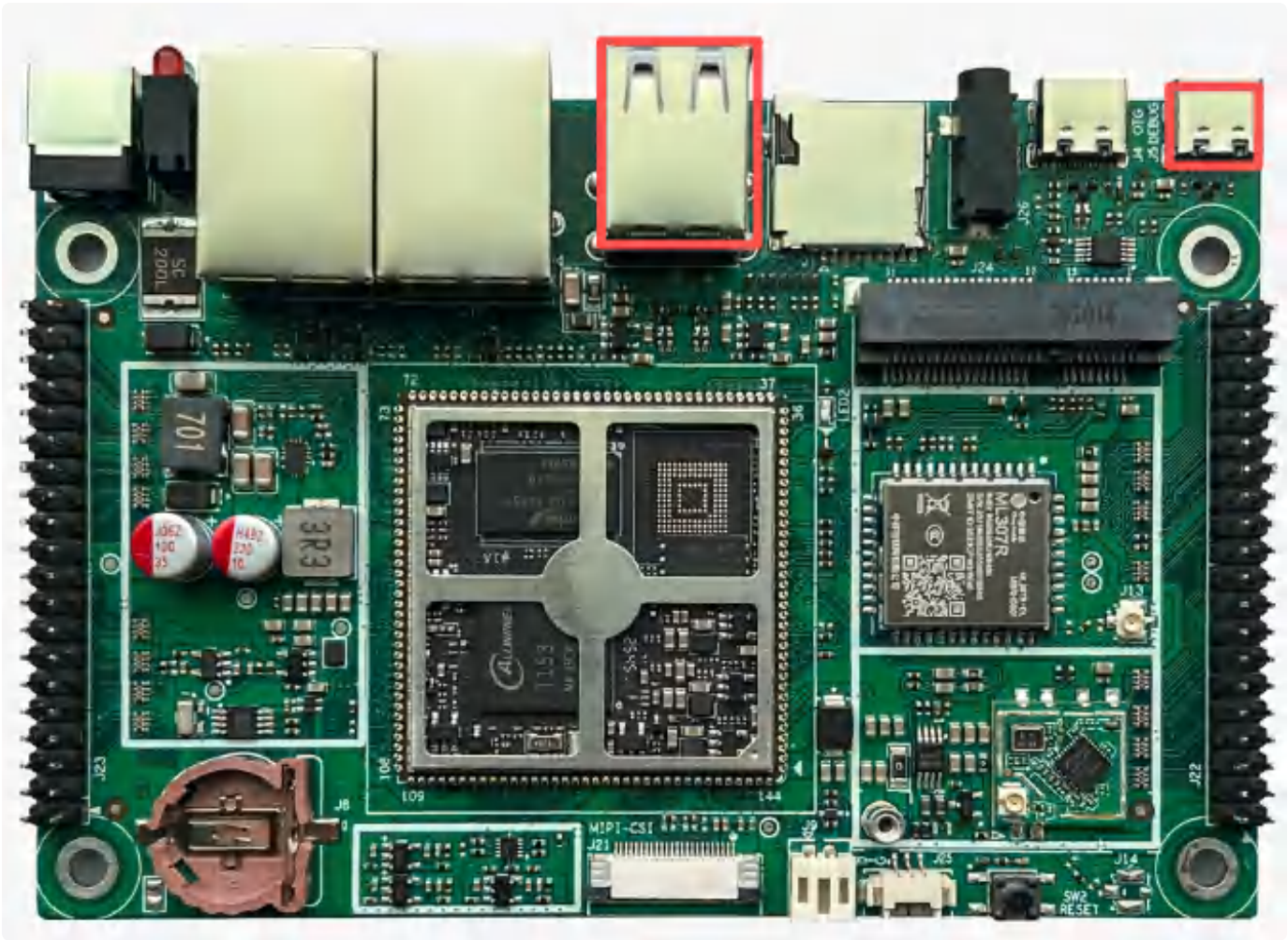
序号	定义	电平/V	说明
1	GND	GND	电源地
2	RS485_B8	/	RS485_B
3	RS485_A8	/	RS485_A
4	RS485_B2	/	RS485_B
5	RS485_A2	/	RS485_A

6	5V	5V	电源5V
---	----	----	------

rs485可以通过《IDO-EVB5301-V1 串口开发手册》进行验证

3. USB

主板共配置2路USB座子，如下图所示：



序号	座子位置	类型
USB1	J4	OTG
USB2	J6-上	USB2.0 HOST
USB3	J6-下	USB2.0 HOST

接上U盘后，系统出现如下打印

```
1 [ 2487.378996] usb 1-1.2: new high-speed USB device number 9 using sunxi-ehci
2 [ 2487.533172] usb 1-1.2: New USB device found, idVendor=0951, idProduct=1666, bcdDevice= 2.00
3 [ 2487.533186] usb 1-1.2: New USB device strings: Mfr=2, Product=3, SerialNumber=4
4 [ 2487.533195] usb 1-1.2: Product: DataTraveler 3.0
5 [ 2487.533203] usb 1-1.2: Manufacturer: Kingston
6 [ 2487.533210] usb 1-1.2: SerialNumber: E0D55E6C0EC11691B8C1065B
7 [ 2487.534261] usb-storage 1-1.2:1.0: USB Mass Storage device detected
8 [ 2487.536615] scsi host0: usb-storage 1-1.2:1.0
```

通过 `fdisk -l` 命令可以查看识别出的usb设备

```
1 # fdisk -l
2 Found valid GPT with protective MBR; using GPT
3
4 .....
5
6 Disk /dev/sda: 29 GB, 30995907072 bytes, 60538881 sectors
7 3768 cylinders, 255 heads, 63 sectors/track
8 Units: sectors of 1 * 512 = 512 bytes
9
10 Device Boot StartCHS EndCHS StartLBA EndLBA Sectors Size
11 /dev/sda1 * 122,32,39 1023,254,63 1961984 59617280 57655297 27.4
12 /dev/sda2 1023,254,63 1023,254,63 59619328 60534783 915456 447
13 M 1b Hidden Win95 FAT32
```

通过 `mount` 手动将U盘挂载至rootfs

```
1 # mkdir /mnt/usb && mount /dev/sda1 /mnt/usb
```

Linux 内核自带的 ntfs 驱动 只能读，不能写（这是历史遗留问题，该驱动自 2004 年起基本停止开发）。

若需要挂载 `ntfs` 格式的U盘或sd卡，需要通过 `ntfs-3g` 命令进行挂载

查看NTFS格式U盘

Bash

```

1 # fdisk -l
2 Found valid GPT with protective MBR; using GPT
3 .....
4
5 Device  Boot StartCHS      EndCHS          StartLBA      EndLBA        Sectors  Siz
   e Id Type
6 /dev/sda1 *  0,1,2          1023,254,63      64  121077759  121077696  57.7
   G  7 HPFS/NTFS
7 Found valid GPT with protective MBR; using GPT
8
9 Disk /dev/mmcblk1: 249737216 sectors, 3158M
10 Logical sector size: 512
11 Disk identifier (GUID): 6c411a4e-ad3e-4594-8831-15292a650574
12 Partition table holds up to 128 entries
13 First usable sector is 34, last usable sector is 249737182
14

```

挂载NTFS格式U盘



Bash

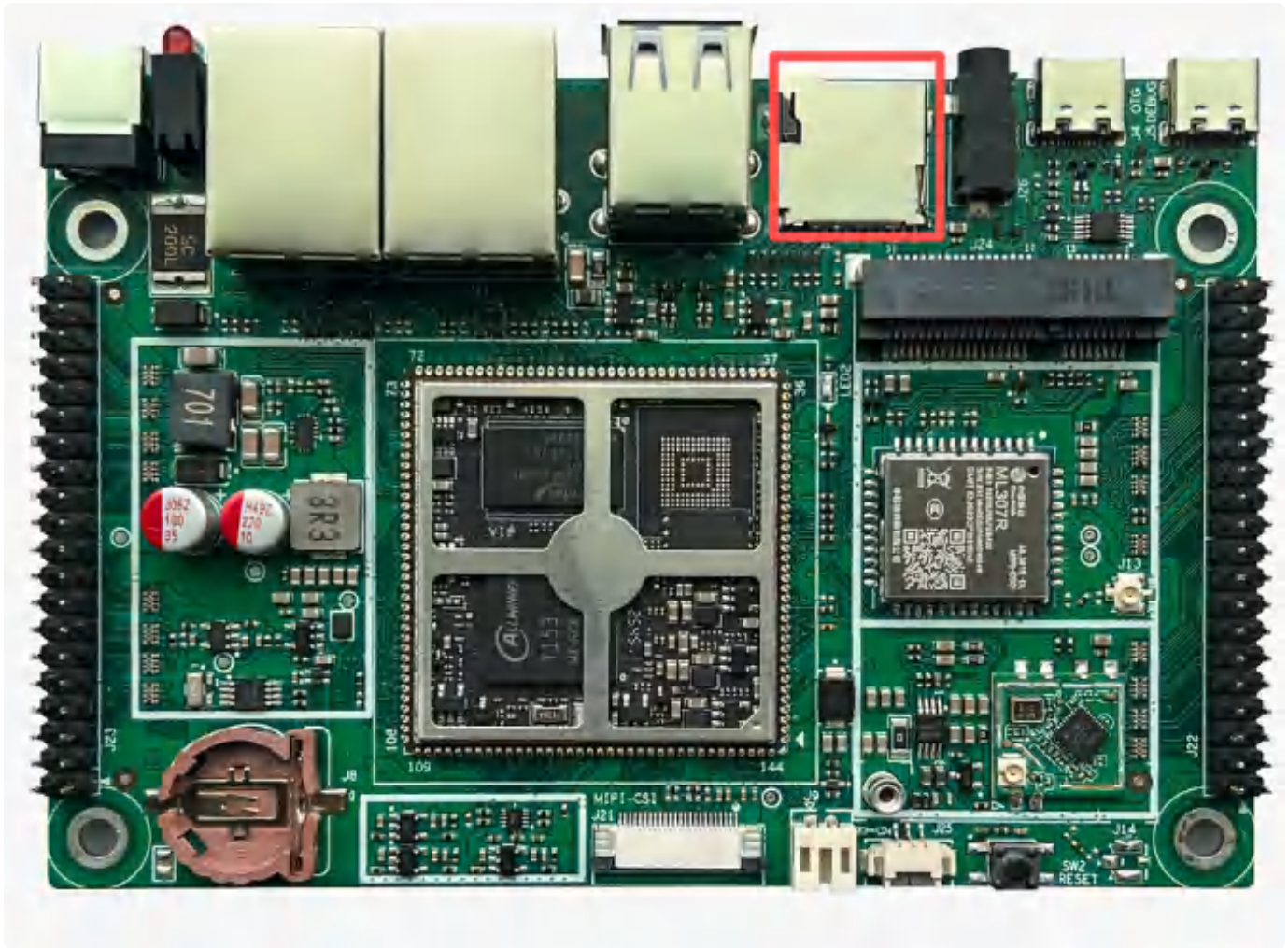
```

1 ## 通过mount命令挂载可以看到，mount尝试将其挂在为exFAT失败后，以ntfs格式进行挂载
2 $ mount /dev/sda1 /mnt/usb/
3 [ 210.471276] exFAT-fs (sda1): invalid fs_name
4 [ 210.476039] exFAT-fs (sda1): failed to read boot sector
5 [ 210.481914] exFAT-fs (sda1): failed to recognize exfat type
6 [ 210.546830] ntfs: volume version 3.1.
7 ## 写入失败
8 $ touch /mnt/usb/123123.txt
9 touch: /mnt/usb/123123.txt: Permission denied
10 #
11
12 ## 通过ntfs-3g命令进行挂载
13 $ ntfs-3g /dev/sda1 /mnt/usb/
14 $ touch /mnt/usb/123123.txt
15 $ echo 123123 > /mnt/usb/123123.txt
16 $ cat /mnt/usb/123123.txt
17 123123

```

4. Micro SD

主板共配置一路Micro SD接口，如下图所示：



插入SD卡后，默认挂载到/mnt/sdcard目录，如下所示：

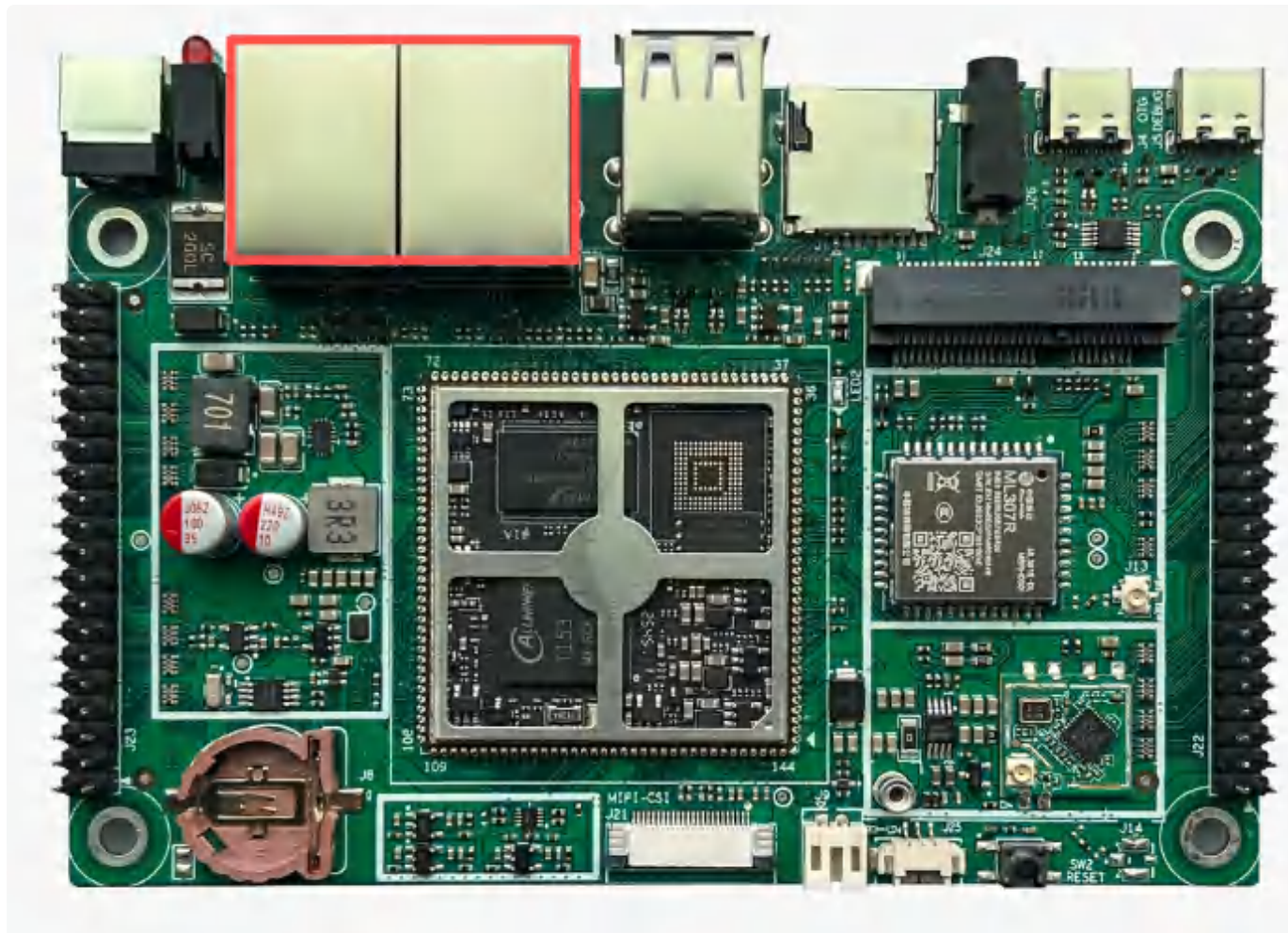
▼ Shell

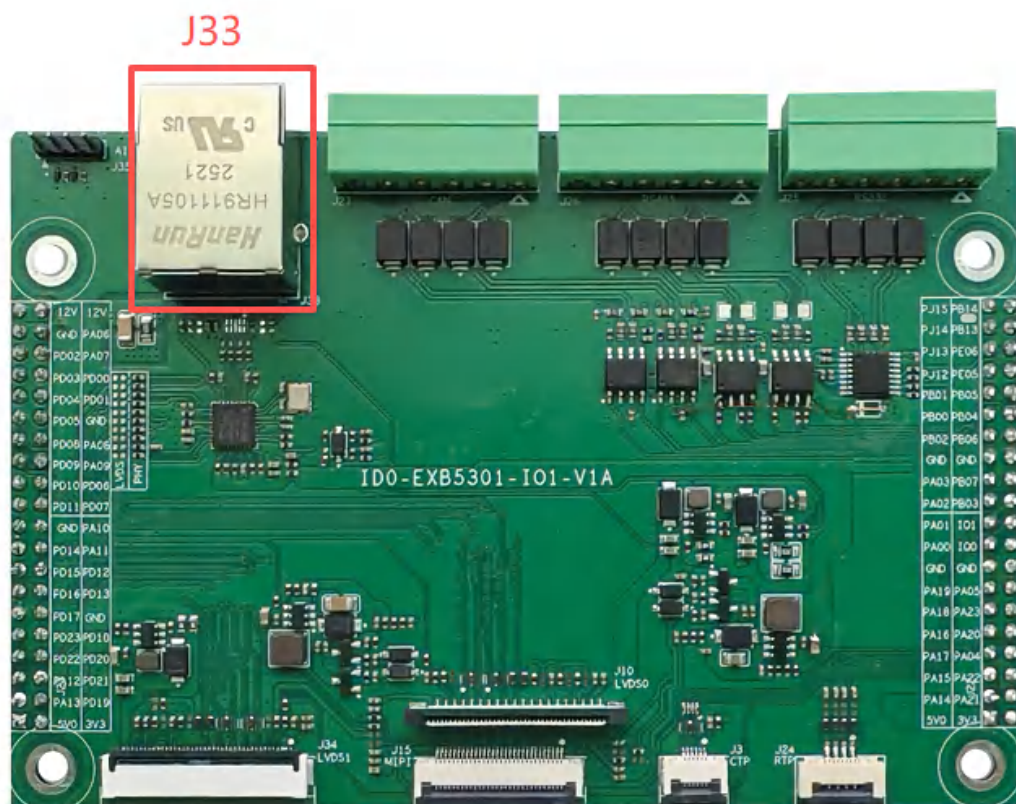
```
1 # mount
2 ...
3 /dev/mmcblk0p1 on /mnt/sdcard type vfat (rw,nodev,noexec,noatime,nodiratime
,fmask=0022,dmask=0022,codepage=437,ioccharset=iso8859-1,shortname=mixed,err
ors=remount-ro)
```

测试项目	要求	测试结果
挂载SD	可以自动挂载fat32格式sd卡	✓
	可以手动挂载NTFS的sd卡	✓

5. Ethernet

主板共配置1路千兆以太网接口和2路百兆以太网接口，如下图所示：





序号	接口位置	速率	网络节点
1	J11	百兆	eth0
2	J10	千兆	eth1
3	J33	百兆	eth2

系统默认开启DHCP服务，动态获取IP。

▼ Shell

```
1 # ifconfig eth0
2 eth0      Link encap:Ethernet  HWaddr 7A:1F:F2:40:0F:F0
3           inet addr:192.168.0.12  Bcast:192.168.0.255  Mask:255.255.255.0
4           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
5           RX packets:25 errors:0 dropped:0 overruns:0 frame:0
6           TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
7           collisions:0 txqueuelen:1000
8           RX bytes:3358 (3.2 KiB)  TX bytes:1368 (1.3 KiB)
9           Interrupt:57 Base address:0xe000
```

可通过ifconfig指令设置临时静态IP，命令如下：

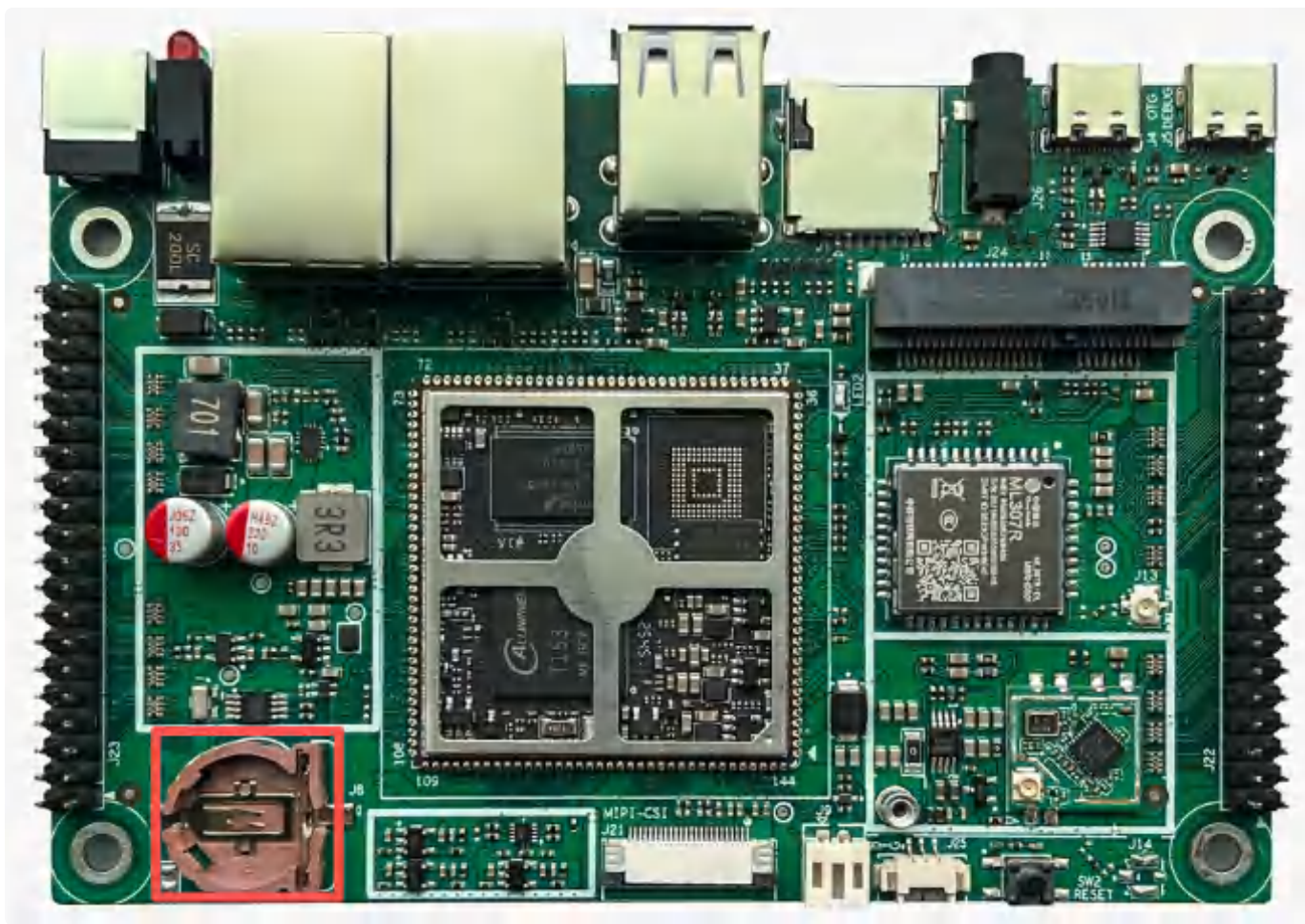
▼ Shell

```
1 # ifconfig eth0 192.168.3.123
```

注：eth2与 mipi/lvds0 硬件通路复用，使用前请先确定硬件是否支持。

6. RTC

主板配置了1路rtc接口，如下图所示：



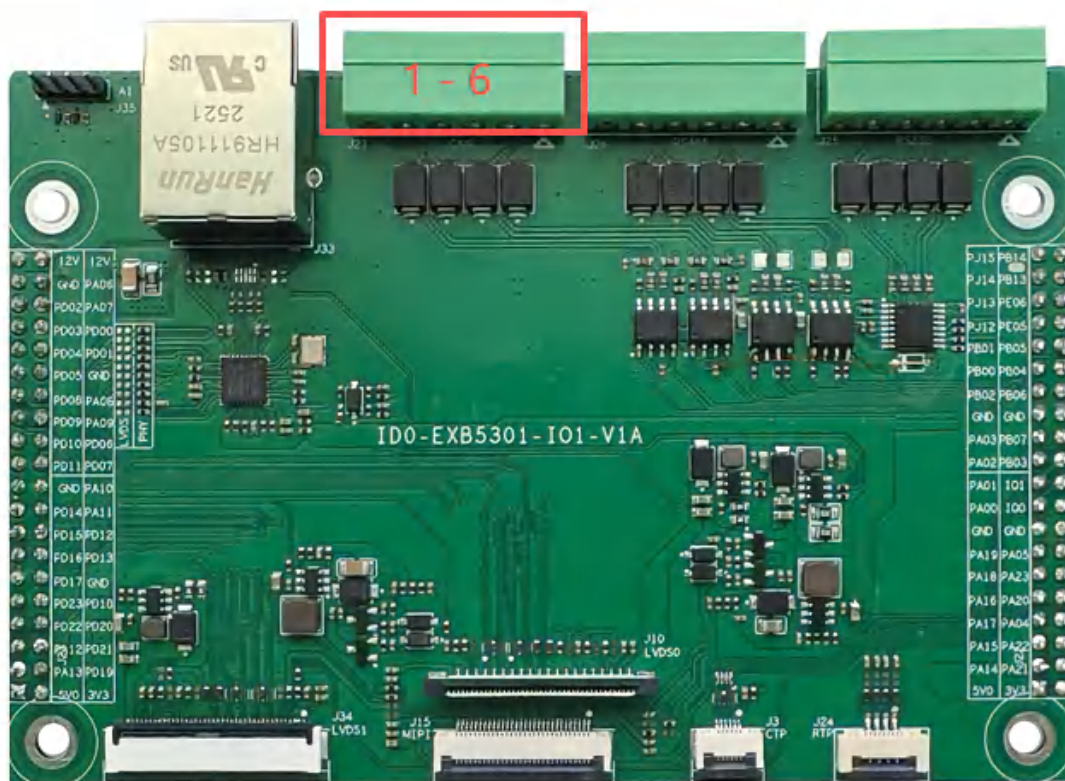
rtc时间设置方法：

Bash

```
1  #设置系统时间
2  $ date -s "2024-10-09 14:02:30"
3
4  #将rtc时间调整为与目前的系统时间一致
5  $ hwclock -w -f /dev/rtc1
6
7  #获取硬件rtc当前时间（断电重启读取时间没有太大偏差）
8  $ hwclock -r -f /dev/rtc1
9  2024-10-09 14:02:35.945604+00:00
```

7. CAN

主板配置了2路can接口，如下图所示：



序号	接口位置	电平类型	串口设备节点	备注
1	J27	CAN	can1	/
2		CAN	can0	/

引脚定义:

序号	定义	电平/V	说明
1	GND	GND	电源地
2	CAN0_L	/	CAN_L
3	CAN0_H	/	CAN_H
4	CAN1_L	/	CAN_L
5	CAN1_H	/	CAN_H
6	5V	5V	电源5V

查看can节点，以can1为例：

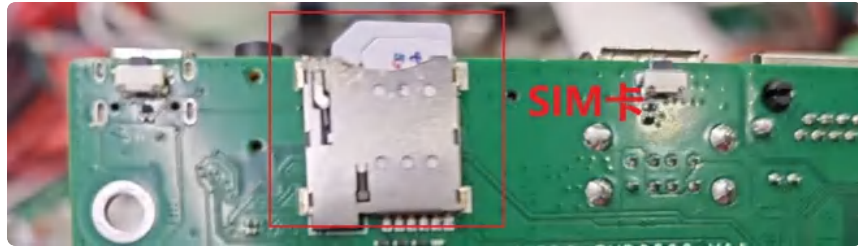
```
▼ Shell |
1 # ifconfig can0
2 can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
3           NOARP  MTU:16  Metric:1
4           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
5           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
6           collisions:0 txqueuelen:10
7           RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
8           Interrupt:49
```

数据收发，以can1为例：

```
▼ Shell |
1 #检查can设备
2 $ ip link show
3
4 #在收发端关闭can0设备
5 $ ip link set can0 down
6
7 #设置仲裁段1M波特率，数据段1M波特率
8 $ ip link set can0 type can bitrate 1000000 loopback off
9 [ 8846.889995] sunxi-can 453c800.can0: PM runtime resume.
10 [ 8846.895843] sunxi-can 453c800.can0: PM runtime suspend.
11
12 #在收发端打开can0设备
13 $ ip link set can0 up
14
15 #在接收端执行candump,阻塞等待报文
16 $ candump can0
17
18 #在发送端执行cansend,发送报文
19 $ cansend can0 123#1122334455667788
20
21 #检查是否启用成功
22 $ ip link show can0
```

8. 4G模块

主板配置了1路中移4G模块ML307R，如下图所示：



拨号上网：

```
▼ Shell |
1  cat /dev/ttyUSB2 &
2
3  #设置拨号协议为RNDIS
4  echo -e "AT+MDIALUPCFG="mode",0" > /dev/ttyUSB2
5
6  #设置拨号协议为ECM
7  echo -e "AT+MDIALUPCFG="mode",1" > /dev/ttyUSB2
8
9  #关闭回显
10 echo -e 'ATE0\r\n' > /dev/ttyUSB2
11
12 #查询PDP上下文配置
13 echo -e 'AT+CGDCONT?\r\n' > /dev/ttyUSB2
14
15 #查询驻网状态
16 echo -e 'AT+COPS?\r\n' > /dev/ttyUSB2
17
18 #设置自动拨号命令
19 echo -e 'AT+MDIALUP=1,1\r\n' > /dev/ttyUSB2
20
21 #查询拨号状态
22 echo -e 'AT+MDIALUP?\r\n' > /dev/ttyUSB2
```

查看网络节点：

```

1 # 如果是mipi屏幕的固件，4g节点则为eth2
2 $ ifconfig eth3
3 eth3      Link encap:Ethernet  HWaddr AC:0C:29:A3:9B:6D
4           inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
5           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
6           RX packets:4 errors:0 dropped:0 overruns:0 frame:0
7           TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
8           collisions:0 txqueuelen:1000
9           RX bytes:928 (928.0 B)  TX bytes:1026 (1.0 KiB)

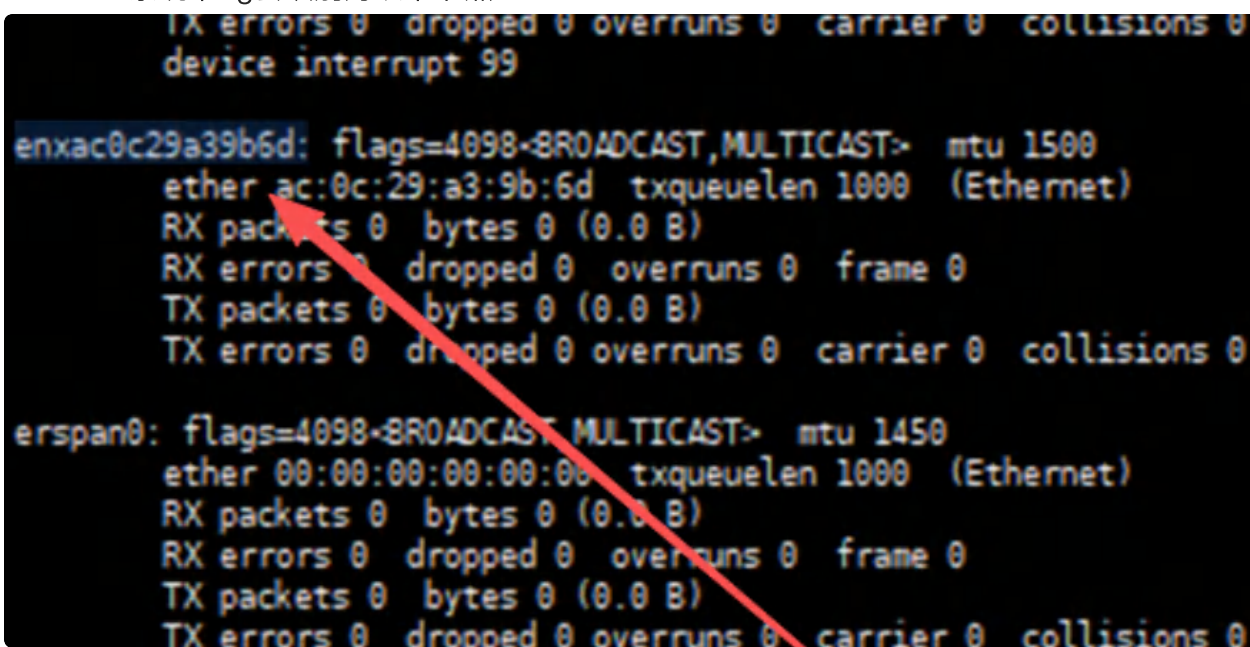
```

注意：

1.ML307R 4G模块支持RNDIS和ECM模块，两种模式选择其一即可。

2.ML307R 4G模块拨号上网后默认获取局域网内IP地址。

3.Ubuntu系统下4g会识别为以下节点



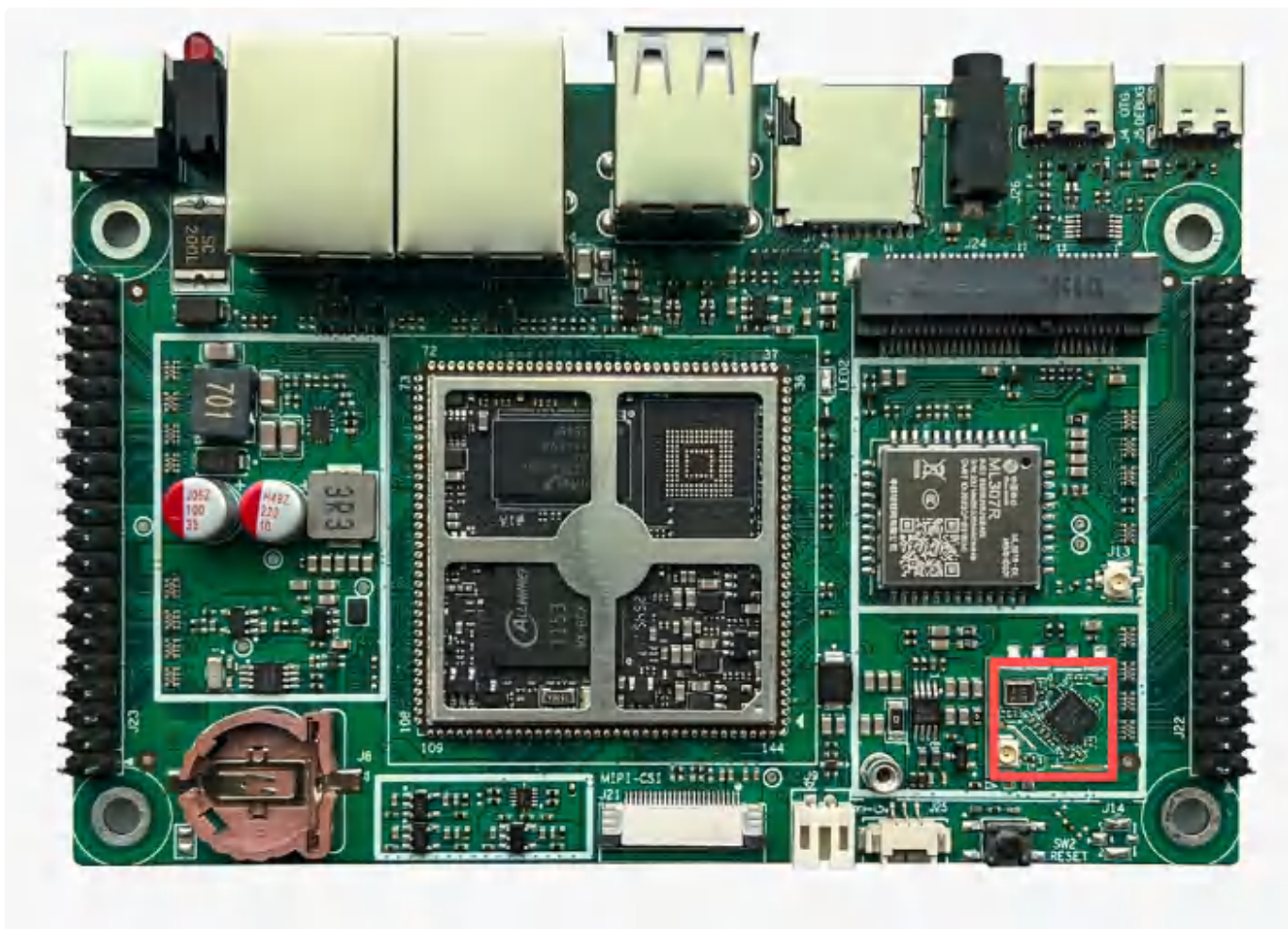
```

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 99
enxac0c29a39b6d: flags=4098<BROADCAST,MULTICAST> mtu 1500
  ether ac:0c:29:a3:9b:6d txqueuelen 1000 (Ethernet)
  RX packets 0 bytes 0 (0.0 B)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 0 bytes 0 (0.0 B)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
erspan0: flags=4098<BROADCAST,MULTICAST> mtu 1450
  ether 00:00:00:00:00:00 txqueuelen 1000 (Ethernet)
  RX packets 0 bytes 0 (0.0 B)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 0 bytes 0 (0.0 B)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

9. 星闪模块

主板配置了1路星闪模块，如下图所示：



9.1. WIFI

STA连接方法：

星闪wifi.so加载起来后，会打开Featureid0和wlan0两个网络节点，Featureid0会影响wlan0的ip获取，需要关闭，命令如下。

```
1 # ifconfig Featureid0 down
```

账号密码配置：

```
▼ Bash |
1 # cat /etc/wpa_supplicant.conf
2 ctrl_interface=/var/run/wpa_supplicant
3 ap_scan=1
4 update_config=1
5
6 network={
7     ssid="SSID"                #账号
8     psk="PASSWORD"            #密码
9     key_mgmt=WPA-PSK
10 }
```

ssid: 账号。

psk: 密码。

若需要连接WPA3加密的WiFi, 可以参考如下配置

```
▼ Bash |
1 ctrl_interface=/var/run/wpa_supplicant
2 ap_scan=1
3 update_config=1
4
5 network={
6     ssid="CS_2.4G_WPA3"
7     sae_password="CS123123"
8     pairwise=CCMP TKIP
9     group=CCMP TKIP
10    key_mgmt=SAE WPA-PSK
11    scan_ssid=1
12    ieee80211w=2
13 }
```

连接方法:

```
▼ Bash |
1 # killall wpa_supplicant
2 # wpa_supplicant -Dnl80211 -c /etc/wpa_supplicant.conf -i wlan0 &
```

获取IP:

```

1 # dhclient wlan0
2 udhcpd: started, v1.36.1
3 udhcpd: broadcasting discover
4 udhcpd: broadcasting select for 192.168.74.160, server 192.168.74.123
5 udhcpd: lease of 192.168.74.160 obtained from 192.168.74.123, lease time 3
  599
6 deleting routers
7 adding dns 192.168.74.123
8
9 # ifconfig wlan0
10 wlan0      Link encap:Ethernet  HWaddr 78:22:88:85:63:96
11            inet addr:192.168.74.160  Bcast:192.168.74.255  Mask:255.255.25
  5.0
12            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
13            RX packets:77 errors:0 dropped:0 overruns:0 frame:0
14            TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
15            collisions:0 txqueuelen:1000
16            RX bytes:6190 (6.0 KiB)  TX bytes:11915 (11.6 KiB)

```

测试:

```

1 # ping www.baidu.com -I wlan0
2 PING www.baidu.com (103.235.46.102): 56 data bytes
3 64 bytes from 103.235.46.102: seq=0 ttl=48 time=266.668 ms
4 64 bytes from 103.235.46.102: seq=1 ttl=48 time=233.898 ms
5 64 bytes from 103.235.46.102: seq=2 ttl=48 time=276.997 ms
6 64 bytes from 103.235.46.102: seq=3 ttl=48 time=229.841 ms

```

9.2. Bluetooth

打开蓝牙设备:

```

1 $ hciconfig hci0 up
2 $ bluetoothd &

```

扫描蓝牙设备:

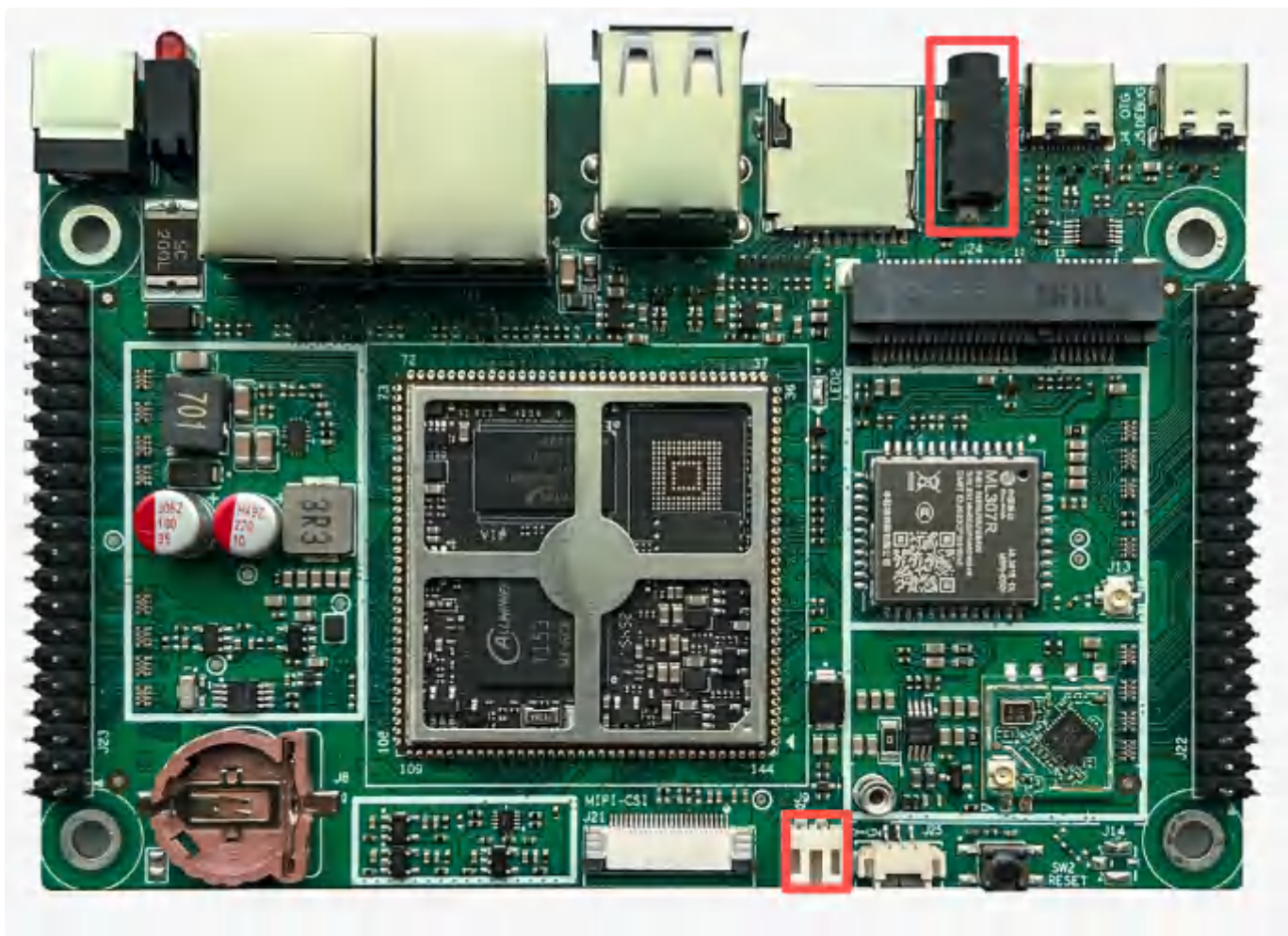
```
▼ Shell |
1 $ hciconfig hci0 iscan
2
3 $ bluetoothctl
4 Agent registered
5 ▾ [CHG] Controller 10:BB:F3:56:44:55 Pairable: yes
6 ▾ [bluetooth]# scan on
7 Discovery started
8 ▾ [CHG] Controller 10:BB:F3:56:44:55 Discovering: yes
9 ▾ [CHG] Device 24:4C:AB:09:98:A6 RSSI: -92
10 ...
11 ▾ [NEW] Device 7C:C1:80:09:DD:6C AirPods - Find My
12 ...
13
```

通过mac配对蓝牙设备：

```
▼ Shell |
1 ▾ [bluetooth]# trust 7C:C1:80:09:DD:6C
2 ▾ [CHG] Device 7C:C1:80:09:DD:6C Trusted: yes
3 Changing 7C:C1:80:09:DD:6C trust succeeded
4 ▾ [bluetooth]# pair 7C:C1:80:09:DD:6C
5 Attempting to pair with 7C:C1:80:09:DD:6C
6 ▾ [CHG] Device 7C:C1:80:09:DD:6C Connected: yes
7 ▾ [CHG] Device 7C:C1:80:09:DD:6C Name: AirPods
8 ▾ [CHG] Device 7C:C1:80:09:DD:6C Alias: AirPods
9 ▾ [CHG] Device 7C:C1:80:09:DD:6C Modalias: bluetooth:v004Cp2013dB087
10 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 00001000-0000-1000-8000-00805f9b34fb
11 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 0000110b-0000-1000-8000-00805f9b34fb
12 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
13 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
14 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 0000111e-0000-1000-8000-00805f9b34fb
15 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 00001200-0000-1000-8000-00805f9b34fb
16 ▾ [CHG] Device 7C:C1:80:09:DD:6C UUIDs: 74ec2172-0bad-4d01-8f77-997b2be0722a
17 ▾ [CHG] Device 7C:C1:80:09:DD:6C ServicesResolved: yes
18 ▾ [CHG] Device 7C:C1:80:09:DD:6C Paired: yes
19 Pairing successful
20 ▾ [AirPods - Find My]# exit
```

10. 声卡

主板配置的声卡接口，如下图所示：



序号	接口位置	功能
1	J9	Speaker
2	J26	Line Out

10.1. 声卡节点

```

1  #声卡节点
2  # aplay -l
3  **** List of PLAYBACK Hardware Devices ****
4  card 0: audiocodec [audiocodec], device 0: sunxi-snd-plat-audio-sunxi-snd-
    codec 2030000.codec-0 []
5      Subdevices: 1/1
6      Subdevice #0: subdevice #0
7
8

```

10.2. 声卡开关

```

1  #声卡控制节点
2  # amixer -c 0 controls
3  numid=2,iface=MIXER,name='DAC DRC Mode'
4  numid=3,iface=MIXER,name='DAC HPF Mode'
5  numid=6,iface=MIXER,name='DAC Output Select'
6  numid=4,iface=MIXER,name='DAC Volume'
7  numid=5,iface=MIXER,name='DACL Volume'
8  numid=7,iface=MIXER,name='LINEOUTL Gain'
9  numid=8,iface=MIXER,name='LINEOUTL Switch'
10 numid=9,iface=MIXER,name='SPK Switch'
11 numid=1,iface=MIXER,name='tx hub mode'
12
13 #打开耳机开关
14 # amixer -c 0 cset numid=8,iface=MIXER,name='LINEOUTL Switch' on
15 numid=8,iface=MIXER,name='LINEOUTL Switch'
16     ; type=BOOLEAN,access=rw-----,values=1
17     : values=on
18
19 #打开功放开关
20 # amixer -c 0 cset numid=9,iface=MIXER,name='SPK Switch' on
21 numid=9,iface=MIXER,name='SPK Switch'
22     ; type=BOOLEAN,access=rw-----,values=1
23     : values=on
24 #
25

```

注：由于喇叭是由lineout音频信号经过功放转换而来，所以在测试喇叭音频时，需先打开耳机开关。

10.3. 音频播放

```
1  #音频播放
2  # aplay -Dhw:0,0 ./test_mono.wav
3  Playing WAVE './test_mono.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
```

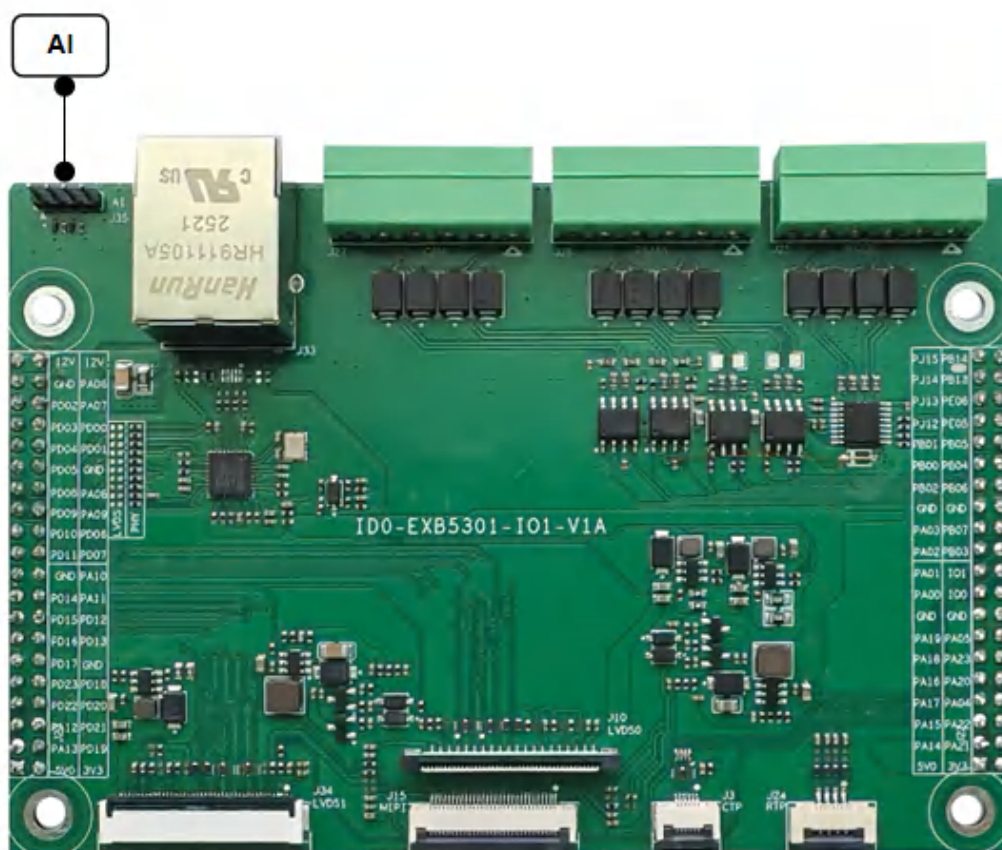
注：由于T153默认声卡只支持单声道音频，播放音频若出现 `aplay:set_params:1358:channels count non available` 等相关报错信息，可通过 `ffmpeg` 将其转换成默认支持的音频。

```
1  $ ffmpeg -i ./8k16bpsStereo.wav -ac 1 8k16bpsStereo_mono.wav
```

`8k16bpsStereo.wav` 为音频源文件，`8k16bpsStereo_mono.wav` 为转换后输出的音频文件（开发板系统默认不带ffmpeg工具，需在其他平台上转换）

11. ADC

拓展板带有两路adc捕获，位于拓展板J35处，如下图



四路引脚从左往右依次为 AI1+, AI1-, AI2+, AI2-

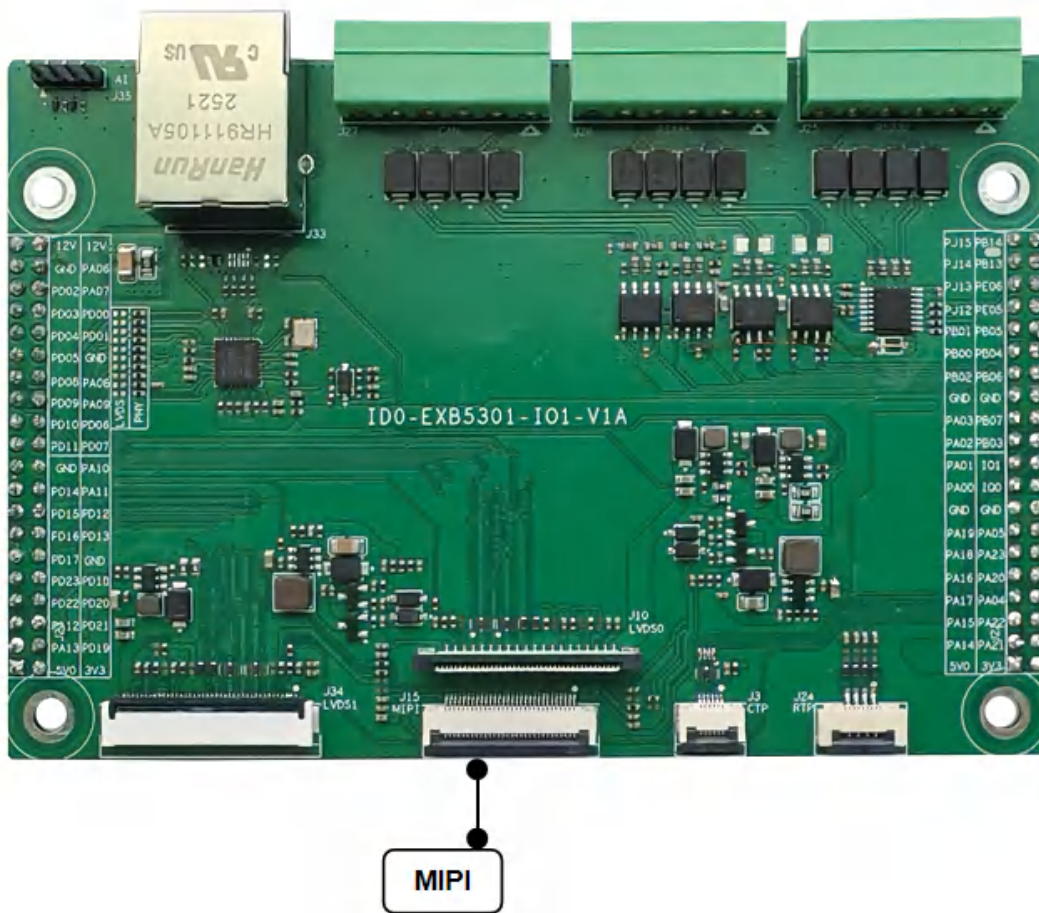
▼ AI捕获 (单位为mv)

Shell

```
1  $ cd /sys/class/gpadc/gpadc_chip0
2  $ echo gpadc6,1 > status
3  $ echo gpadc7,1 > status
4
5  # 捕获AI1的值
6  $ echo 6 > data ;cat data
7  gpadc0-channel6 voltage data is 1792
8  # 捕获AI2的值
9  $ echo 7 > data ;cat data
10 gpadc0-channel7 voltage data is 1791
```

12. 显示接口

12.1. MIPI

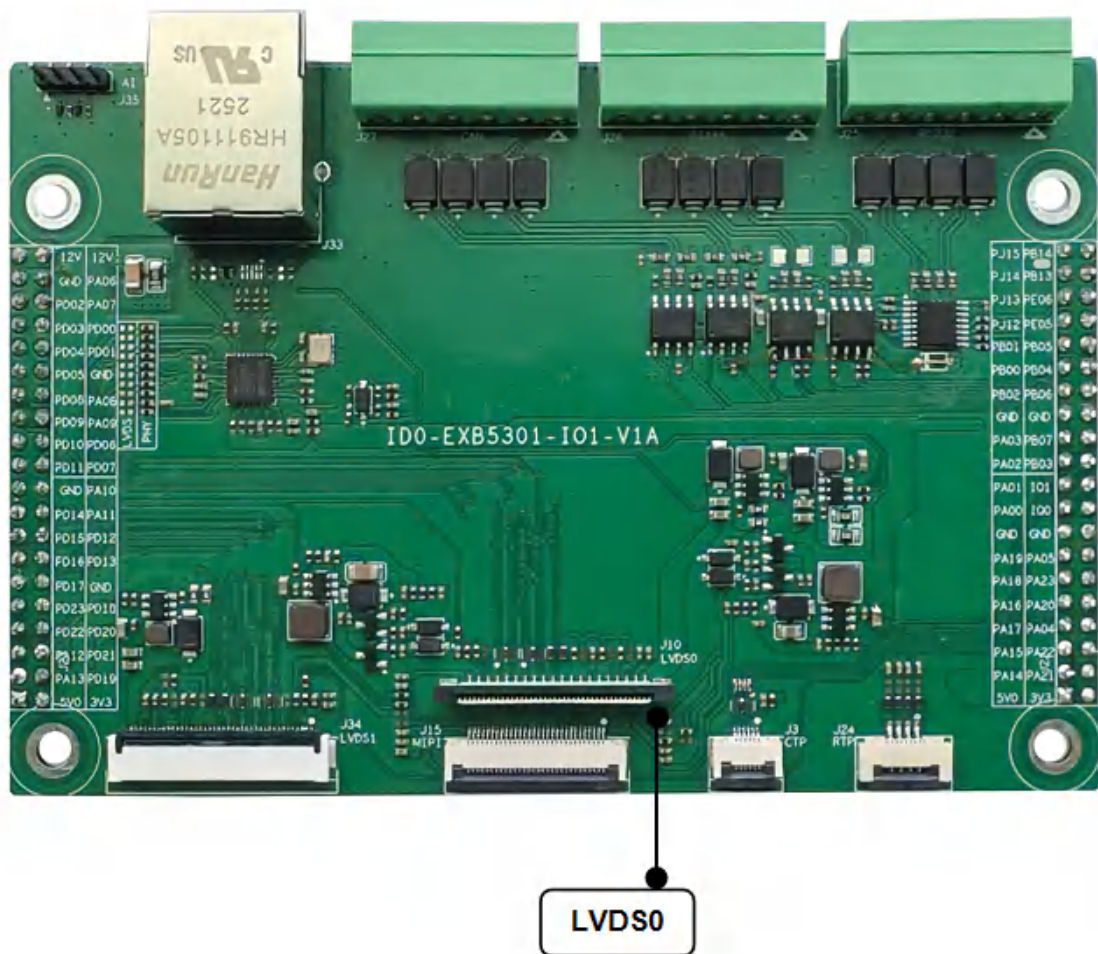


分辨率：1024x600

默认屏幕是否支持触摸：是

注：eth2与 mipi/lvds0 硬件通路复用，使用前请先确定硬件是否支持。

12.2. LVDS0屏幕

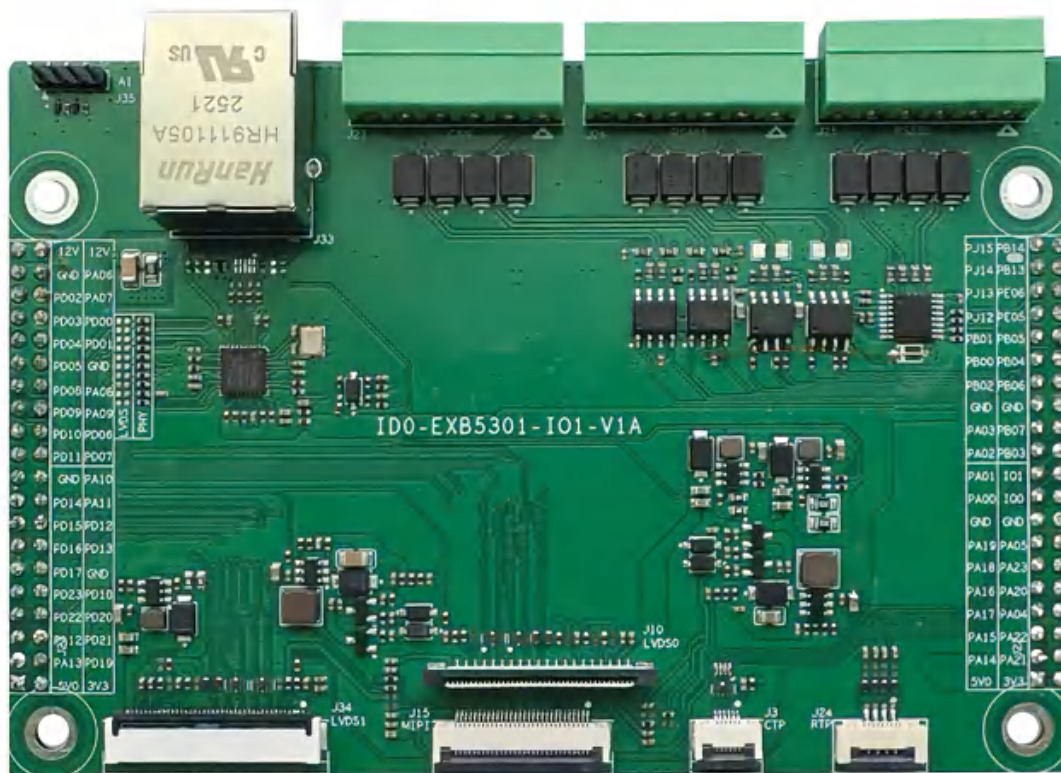


分辨率：1024x600

默认屏幕是否支持触摸：否

注：eth2与 mipi/lvds0 硬件通路复用，使用前请先确定硬件是否支持。

12.3. LVDS1屏幕



LVDS1

分辨率：1024x600

默认屏幕是否支持触摸：否

