

IDO-EVB3568-V2 - Buildroot系统使用说明

1、调试

1.1 串口调试

1.2 ADB调试

1.3 ssh调试

2、串口

2.1 测试方法

3、USB

3.1 电源控制

4、TF CARD

5、以太网

5.1 查看以太网IP地址

5.1.1 使用命令查看

5.2 设置IP地址

5.2.1 临时设置IP

5.2.2 设置静态IP

6、WiFi

6.1 连接热点

7、连接蓝牙设备

7.1 扫描设备

8、4G

9、音频

9.1 查看声卡设备

9.2 播放音频

9.4 录音

10、摄像头

10.1 查看摄像头是否存在

10.2 抓取视频

11、RTC

方法一

11.1 获取RTC时间

11.2 设置RTC时间

方法二

12、开机自启动

13、PWM功能

13.1 测试

14、屏幕控制

14.1 背光调节

15、按键

16、ADC

16.1 ADC转换方法

16.2 测试

17、网络优先级设置

17.1 查看路由表

17.2 设置默认路由

17.2.1 设置WiFi为默认路由

17.2.2 设置以太网为默认路由

18、CAN

18.1测试



IDO-EVB3568-V2

Buildroot 系统使用说明

深圳触觉智能科技有限公司

www.industio.cn

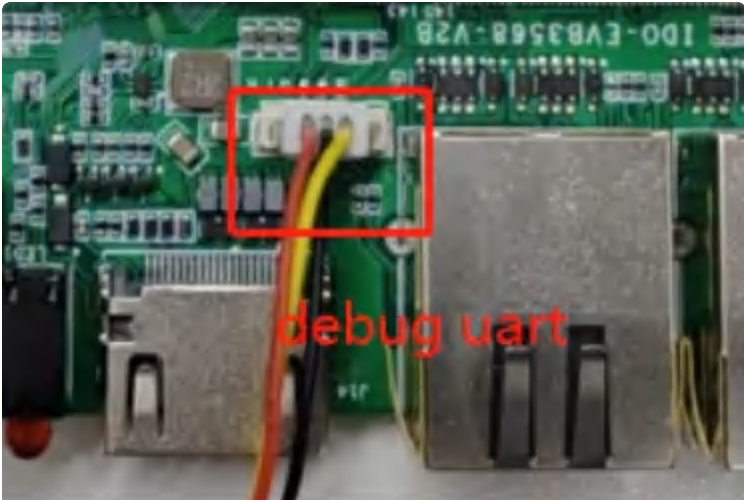
文档修订历史

版本	修订内容	修订	审核	日期
V1.0	创建文档	何伟聪		2024/1/4
	问题： 1. otg 切换到host后无法切换回device模式 2. j20 底座，摄像头接入，无节点			

1、调试

1.1 串口调试

串口调试端口位于J4，通信参数为1500000 8 N 1，电平状态为TTL电平。



串口调试默认无登录账号密码为：

▼

Bash

```
1 root@rk3568-buildroot:/# ls
2 S36load_wifi_modules  etc      lost+found  opt        sbins   udisk
3 bin                   info     media       proc       sdcard  userdata
4 busybox.fragment      lib      misc        rockchip-test sys      usr
5 data                  lib64    mnt         root       system  var
6 dev                   linuxrc  oem         run        tmp     vendor
```

1.2 ADB调试

ADB调试端口位于J5，使用TYPE-C线，连接主板的TYPE-C端口和电脑，即可在电脑上使用adb调试。



```
▼ Bash |
1 C:\Users\industio_1>adb shell
2 root@rk3568-buildroot:/# ls
3 S36load_wifi_modules  etc          lost+found  opt          sbin         udisk
4 bin                   info         media       proc         sdcard       userdata
5 busybox.fragment     lib         misc       rockchip-test sys          usr
6 data                 lib64       mnt        root         system       var
7 dev                  linuxrc     oem        run          tmp          vendor
8 root@rk3568-buildroot:/#
```

1.3 ssh调试

系统默认ssh账号和密码为 root@ rockchip。

```
? MobaXterm Personal Edition v21.5 ?
(SSH client, X server and network tools)

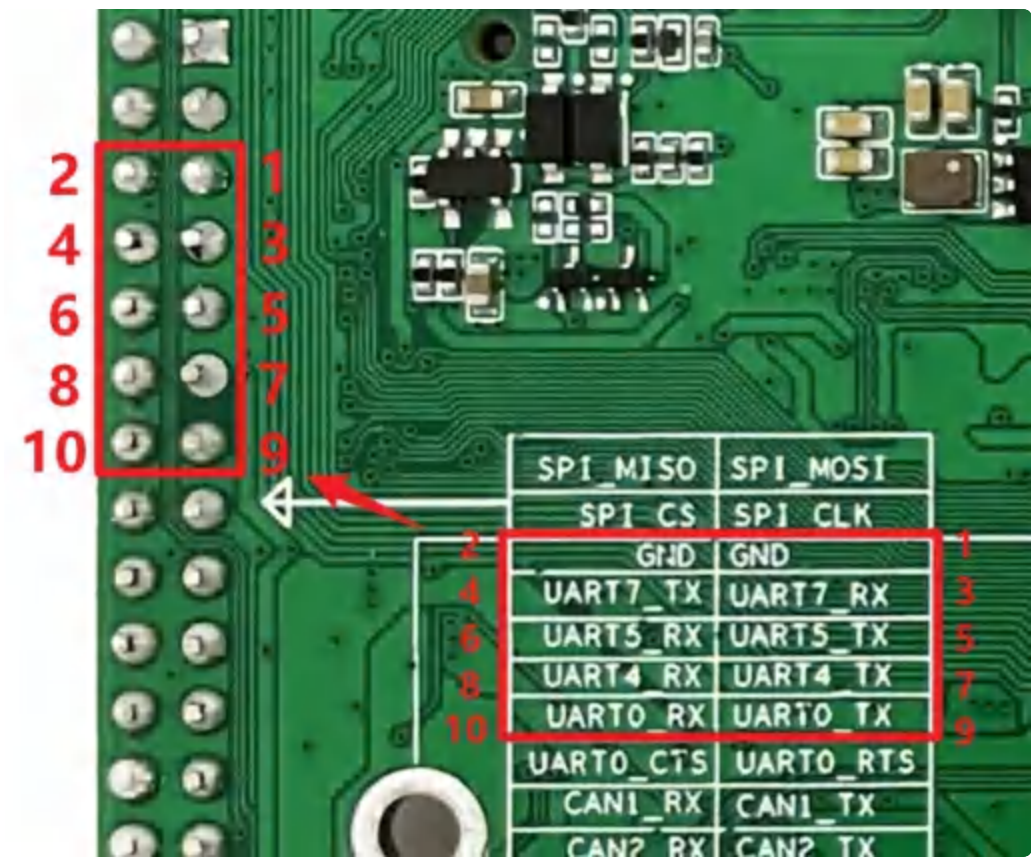
> SSH session to root@192.168.0.121
? Direct SSH      : ✓
? SSH compression : ✗ (disabled or not supported by server)
? SSH-browser     : ✓
? X11-forwarding  : ✗ (disabled or not supported by server)

> For more info, ctrl+click on help or visit our website.

root@rk3568-buildroot:~#
root@rk3568-buildroot:~# ls
root@rk3568-buildroot:~# cd /
root@rk3568-buildroot:/# ls
S36load_wifi_modules  data  info  linuxrc  misc  opt  root  sdcard  tmp  usr
bin                   dev   lib   lost+found  mnt  proc  run  sys  udisk  var
busybox.fragment     etc   lib64  media    oem  rockchip-test  sbin  system  userdata  vendor
root@rk3568-buildroot:/#
```

2、串口

主板共配置了4路串口（不包括调试串口），其中1路支持流控。



序号	设备节点	默认电平类型	位置	备注
1	/dev/ttyS0	TTL	J24	4线，支持流控
2	/dev/ttyS4	TTL	J24	2线，不支持流控
3	/dev/ttyS5	TTL	J24	2线，不支持流控
4	/dev/ttyS7	TTL	J24	2线，不支持流控

2.1 测试方法

4路使用microcom工具进行简单的收发测试。

以测试/dev/ttyS0为例：

```

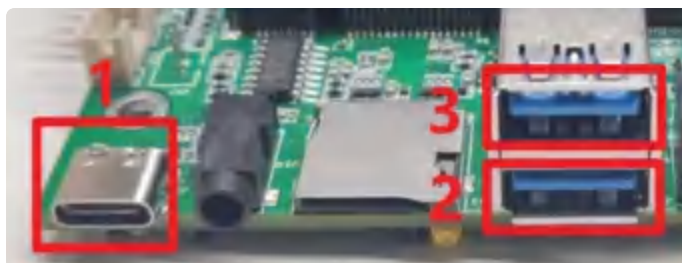
1 root@rk3568-buildroot:/# microcom -s 115200 -X /dev/ttyS
2 ttyS0 ttyS4 ttyS5 ttyS7 ttyS8
3 root@rk3568-buildroot:/# microcom -s 115200 -X /dev/ttyS0
4 [ 9094.238336] of_dma_request_slave_channel: dma-names property of node '/
serial@fdd50000' missing or empty
5 [ 9094.238476] dw-apb-uart fdd50000.serial: failed to request DMA, use int
errupt mode
6 22222222
7 22222222
8 22222222
9 22222222
10 22222222
11 22222222

```

buildroot系统如果使用了microcom无法使用ctr+\退出，只能上下电重启

3、USB

主板共配置了5路USB接口，分别为USB0-5，均为USB-HOST。



编号	名称	类型	位置
1	usb0	USB OTG	J5
2	usb1	host-2.0	J6
3	usb2	host-3.0	J6

4	usb3	host-2.0	J9
5	usb4	host-2.0	J8
6	usb5	host-2.0	J7

USB OTG 支持host 和device 模式的切换，软件切换方法如下

```

1  ## host
2  echo host > /sys/devices/platform/fe8a0000.usb2-phy/otg_mode
3  ## device
4  echo peripheral > /sys/devices/platform/fe8a0000.usb2-phy/otg_mode

```

当USB-HOST插入U盘后，会自动挂载/media/ido/目录下：

```

1  root@rk3568-buildroot:/# ls /media/ido/
2  KINGSTON

```

3.1 电源控制

默认所有USB-HOST的电源都是开启的，其中USB3-5我们提供了开启/关闭电源的方法。

编写	名称	电源控制节点	位置
4	usb3	/sys/class/leds/usb_fed3_pwr/brightness	J9
5	usb4	/sys/class/leds/usb_fed2_pwr/brightness	J8
6	usb5	/sys/class/leds/usb_fed1_pwr/brightness	J7

打开USB3的电源：

```

1  root@rk3568-buildroot:/# echo 255 > /sys/class/leds/usb_fed1_pwr/brightness

```

关闭USB3的电源：

```
1 root@rk3568-buildroot:/# echo 0 > /sys/class/leds/usb_fed1_pwr/brightness
```

USB4-5的电源控制方法类似。

4、TF CARD

主板配置了一个TF CARD接口，当TF CARD接口插入TF卡后，会自动挂载到/media/ido/目录下。

```
1 root@rk3568-buildroot:/# [ 494.235146] mmc_host mmc1: Bus speed (slot 0)
= 375000Hz (slot req 400000Hz, actual 375000HZ div = 0)
2 [ 494.400868] mmc_host mmc1: Bus speed (slot 0) = 50000000Hz (slot req 50
000000Hz, actual 50000000HZ div = 0)
3 [ 494.401109] mmc1: new high speed SDHC card at address e624
4 [ 494.403724] mmcblk1: mmc1:e624 SU04G 3.69 GiB
5 [ 494.420352] mmcblk1: p1
6 [ 494.870755] FAT-fs (mmcblk1p1): utf8 is not a recommended IO charset fo
r FAT filesystems, filesystem will be case sensitive!
7 [ 494.876127] FAT-fs (mmcblk1p1): Volume was not properly unmounted. Som
e data may be corrupt. Please run fsck.
8
9 root@rk3568-buildroot:/#
10 root@rk3568-buildroot:/# df -h
11 Filesystem      Size  Used Avail Use% Mounted on
12 /dev/root        1.1G  608M  421M   60% /
13 devtmpfs         1.9G   8.0K   1.9G    1% /dev
14 tmpfs            2.0G  252K   2.0G    1% /tmp
15 tmpfs            2.0G  312K   2.0G    1% /run
16 tmpfs            2.0G     0   2.0G    0% /dev/shm
17 /dev/mmcblk0p7   121M   12M  101M   11% /oem
18 /dev/mmcblk0p8    23G  240K   22G    1% /userdata
19 /dev/mmcblk1p1   3.7G   1.9M   3.7G    1% /mnt/sdcard
20
```

5、以太网

5.1 查看以太网IP地址

5.1.1 使用命令查看

系统默认以太网为动态获取IP，当以太网接口插入网线时，会自动获取IP。

▼

Bash

```
1 root@rk3568-buildroot:/# ifconfig eth0
2 eth0      Link encap:Ethernet  HWaddr 42:DC:16:A0:6C:1F
3           inet addr:192.168.0.121  Bcast:0.0.0.0  Mask:255.255.255.0
4           UP BROADCAST MULTICAST  MTU:1500  Metric:1
5           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
6           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
7           collisions:0 txqueuelen:1000
8           RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
9           Interrupt:49
10
```

5.2 设置IP地址

5.2.1 临时设置IP

▼

Plain Text

```
1 ifconfig eth1 192.168.1.123
```

5.2.2 设置静态IP

▼

Bash

```
1 vi /etc/network/interfaces
2 -----
3 # interface file auto-generated by buildroot
4 auto lo
5 iface lo inet loopback
6 auto eth0
7 iface eth0 inet static
8 address 192.168.0.121
9 netmask 255.255.255.0
10 gateway 192.168.0.1
11 dns-nameservers 114.114.114.114
```

6、WiFi

系统上电默认会打开WiFi，对应的网络设备节点为wlan0。

```
1 root@rk3568-buildroot:/# ifconfig wlan0
2 wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
3         ether 2c:d2:6b:10:ea:4d txqueuelen 1000 (Ethernet)
4         RX packets 0 bytes 0 (0.0 B)
5         RX errors 0 dropped 0 overruns 0 frame 0
6         TX packets 0 bytes 0 (0.0 B)
7         TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
8
9 root@rk3568-buildroot:/#
```

6.1 连接热点

修改/userdata/cfg/wpa_supplicant.conf，填写正确的热点账号和密码：

```
1 [root@RK356X:/]# cat /userdata/cfg/wpa_supplicant.conf
2 ctrl_interface=/var/run/wpa_supplicant
3 ap_scan=1
4 update_config=1
5
6 network={
7     ssid="TP-LINK_B87A"
8     psk="12345678"
9     key_mgmt=WPA-PSK
10 }
11 [root@RK356X:/]#
```

执行命令，连接wifi

```
1 wpa_supplicant -D nl80211 -i wlan0 -c /userdata/cfg/wpa_supplicant.conf -B
```

测试

```
1 root@rk3568-buildroot:/# ifconfig wlan0
2 wlan0      Link encap:Ethernet  HWaddr 10:BB:F3:15:5A:25
3             inet addr:192.168.1.113  Bcast:192.168.1.255  Mask:255.255.255.0
4             inet6 addr: fe80::723e:b72c:ccf:3282/64 Scope:Link
5             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
6             RX packets:198 errors:0 dropped:0 overruns:0 frame:0
7             TX packets:110 errors:0 dropped:0 overruns:0 carrier:0
8             collisions:0 txqueuelen:1000
9             RX bytes:25864 (25.2 KiB)  TX bytes:12986 (12.6 KiB)
10
11
12 root@rk3568-buildroot:/# ping baidu.com -I wlan0
13 64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=1 ttl=49 time=47.2 m
14 s
15 64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=2 ttl=49 time=48.2 m
16 s
17 64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=3 ttl=49 time=47.3 m
18 s
```

7、连接蓝牙设备

设备节点为hci0，我们已经做了开机自启动开启蓝牙功能

蓝牙功能开启后，将产生hci0节点

```
1 root@rk3568-buildroot:/# hciconfig
2 hci0:      Type: Primary  Bus: UART
3             BD Address: 10:BB:F3:15:CF:4B  ACL MTU: 1021:8  SCO MTU: 255:12
4             UP RUNNING
5             RX bytes:1486 acl:0 sco:0 events:49 errors:0
6             TX bytes:3990 acl:0 sco:0 commands:49 errors:0
```

7.1 扫描设备

使用hcitool工具，扫描蓝牙设备

```

▼ Bash |
1 root@rk3568-buildroot:/# hcitool -i hci0 scan
2 Scanning ...
3
4         BC:64:D9:BF:92:84      Ace 2
5         AC:D6:18:5C:0D:4D      cainiaocl

```

8、4G

主板默认适配EC20模块（4G）和 RG200U-CN（5G），上电前，正确按照模块和SIM卡，上电后，系统会自动进行拨号上网。

序号	模块名称	说明
1	EC20	4G LTE
2	RG200U-CN	支持 5G NSA 和 SA 模式，支持 TDD 和 FDD 两种模式，向下兼容 4G/3G。

拨号成功会产生wwan0网络节点：

```

▼ Bash |
1
2 root@rk3568-buildroot:/# ifconfig wwan0
3 wwan0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
4           inet addr:10.2.155.147  P-t-P:10.2.155.147  Mask:255.255.255.248
5           UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
6           RX packets:45 errors:0 dropped:0 overruns:0 frame:0
7           TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
8           collisions:0 txqueuelen:1000
9           RX bytes:4829 (4.7 KiB)  TX bytes:9410 (9.1 KiB)
10

```

使用ping命令测试4G/5G上网功能是否正常：

```

1 root@rk3568-buildroot:/# ping baidu.com -I wwan0
2 PING baidu.com (110.242.68.66) from 10.2.155.147 wwan0: 56(84) bytes of dat
  a.
3 64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=1 ttl=49 time=107 ms
4 64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=2 ttl=49 time=79.4 ms

```

9、音频

9.1 查看声卡设备

```

1 root@rk3568-buildroot:/# aplay -l
2 **** List of PLAYBACK Hardware Devices ****
3 card 0: rockchipdmi [rockchip,hdmi], device 0: fe400000.i2s-i2s-hifi i2s-h
  ifi-0 [fe400000.i2s-i2s-hifi i2s-hifi-0]
4   Subdevices: 1/1
5   Subdevice #0: subdevice #0
6 card 1: rockchiprk809co [rockchip,rk809-codec], device 0: fe410000.i2s-rk81
  7-hifi rk817-hifi-0 [fe410000.i2s-rk817-hifi rk817-hifi-0]
7   Subdevices: 1/1
8   Subdevice #0: subdevice #0
9

```

9.2 播放音频

注意：如果接入的是其他显示器只有一个声卡，这时候spk_c0就是播放到耳机或者喇叭上
播放到HDMI：

```

1 rk3568-buildroot:/root# aplay -D plug:spk_c0 Rear_Center.wav
2 Playing WAVE 'Rear_Center.wav' : Signed 16 bit Little Endian, Rate 48000 H
  z, Mono
3

```

播放到Lineout：



Bash

```
1 rk3568-buildroot:/root# aplay -D plug:spk_c1 Rear_Center.wav
2 Playing WAVE 'Rear_Center.wav' : Signed 16 bit Little Endian, Rate 48000 H
  z, Mono
3
```

播放到耳机（需要插入耳机）：



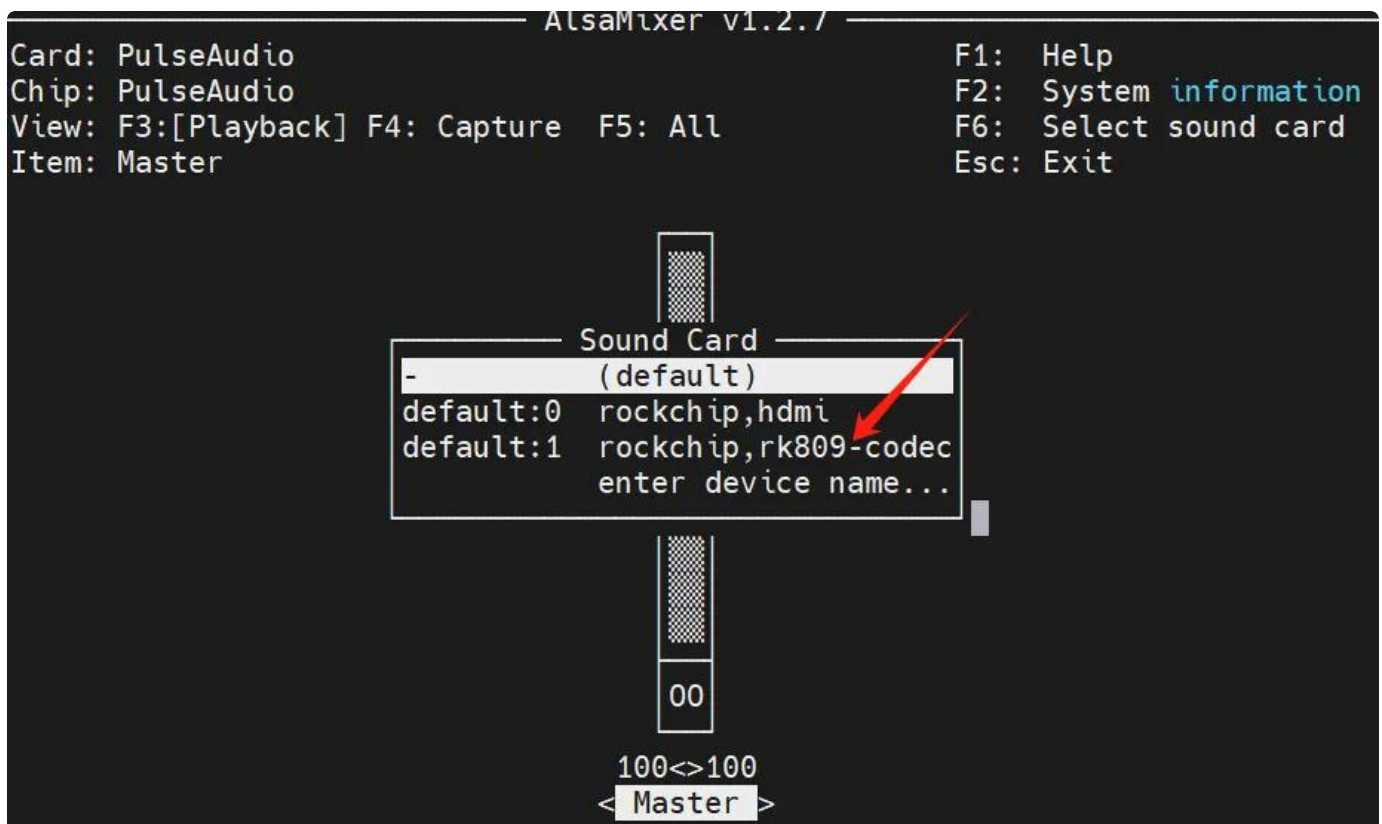
Bash

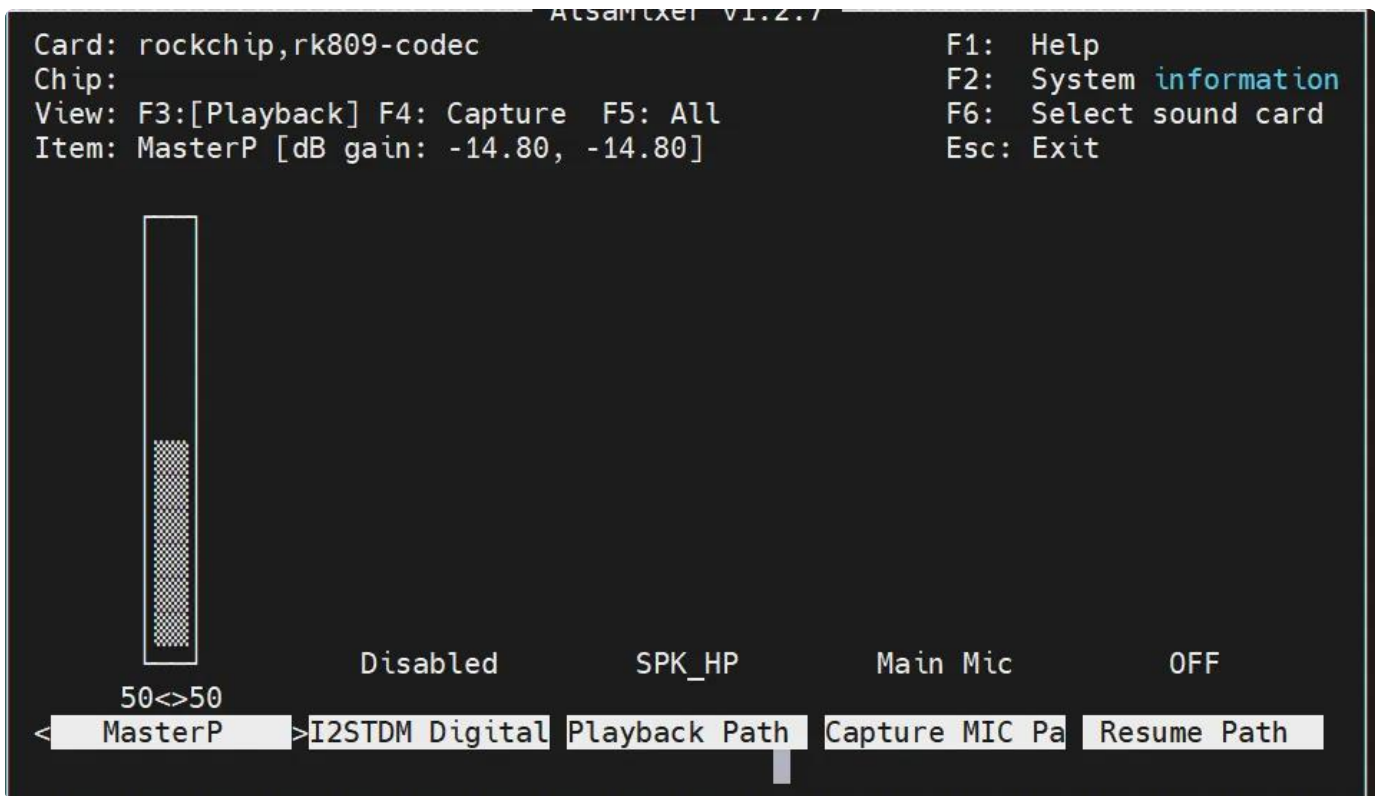
```
1 rk3568-buildroot:/root# aplay -D plug:spk_c1 Rear_Center.wav
2 Playing WAVE 'Rear_Center.wav' : Signed 16 bit Little Endian, Rate 48000 H
  z, Mono
```

注：如果播放到耳机或者喇叭没有声音，那么需要在alsamixer配置一下

```
1 rk3568-buildroot:/root# alsamixer
2
```

键盘按下"s",选择声卡





如图所示，最左边是控制播放音量大小的，hdmi也是一样，选择声卡进入就可以控制音量大小，playback path 何capture mic pa 必须选择，否则会播放没有声音

9.4 录音

将麦克风连接到J11。



使用arecord工具可以进行录音测试：

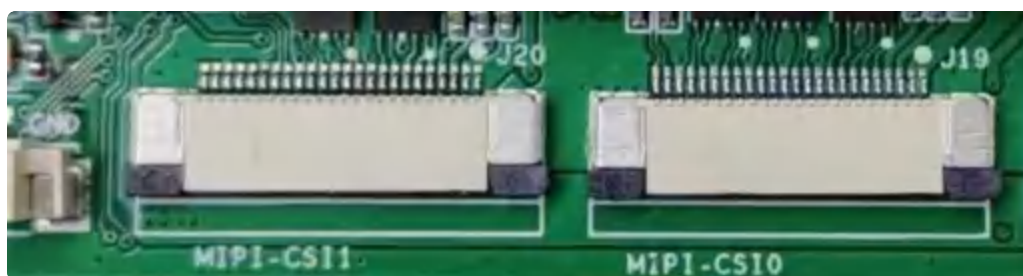
```
▼ Bash |
1 root@rk3568-buildroot:/root# arecord -D hw:1,0 -r 48000 -c 2 -f S16_LE tes
t.wav
2 Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Ste
reo
3 ^CAborted by signal Interrupt...
4 root@rk3568-buildroot:/root#
5
```

注：因为我们的card 1，rk809集成了mic所以，我们在选择录音的时候选择的是声卡1即hw:1,0
录音完后播放到喇叭测试：

```
▼ Bash |
1 root@rk3568-buildroot:/root# aplay -D plug:spk_c1 test.wav
2 Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stere
o
3 ^CAborted by signal Interrupt...
4
```

10、摄像头

主板默认适配OV5648+OV8858摄像头。



10.1 查看摄像头是否存在

```

1  root@rk3568-buildroot:/# media-ctl -pd /dev/media0
2  Media controller API version 5.10.160
3
4  Media device information
5  -----
6  driver          rkisp-vir0
7  model           rkisp0
8  serial
9  bus info
10 hw revision     0x0
11 driver version  5.10.160
12
13 Device topology
14 - entity 1: rkisp-isp-subdev (4 pads, 7 links)
15     type V4L2 subdev subtype Unknown flags 0
16     device node name /dev/v4l-subdev0
17     pad0: Sink
18         [fmt:SBGGR10_1X10/2592x1944 field:none
19         crop.bounds:(0,0)/2592x1944
20         crop:(0,0)/2592x1944]
21         <- "rkisp-csi-subdev":1 [ENABLED]
22         <- "rkisp_rawrd0_m":0 []
23         <- "rkisp_rawrd2_s":0 []
24     pad1: Sink
25         <- "rkisp-input-params":0 [ENABLED]
26     pad2: Source
27         [fmt:YUYV8_2X8/2592x1944 field:none colorspace:smpte170m q
28 quantization:full-range
29         crop.bounds:(0,0)/2592x1944
30         crop:(0,0)/2592x1944]
31         -> "rkisp_mainpath":0 [ENABLED]
32         -> "rkisp_selfpath":0 [ENABLED]
33     pad3: Source
34         -> "rkisp-statistics":0 [ENABLED]
35     .....
36     .....
37     .....
38
39 - entity 70: m00_b_ov5648 4-0036 (1 pad, 1 link)
40     type V4L2 subdev subtype Sensor flags 0
41     device node name /dev/v4l-subdev3
42     pad0: Source
43         [fmt:SBGGR10_1X10/2592x1944@10000/150000 field:none]
44         -> "rockchip-csi2-dphy0":0 [ENABLED]

```

10.2 抓取视频

```

1  root@rk3568-buildroot:/# v4l2-ctl --verbose -d /dev/video0 --set-fmt-video
   =width=1920,height=1080,pixelformat='NV12' --stream-mmap=4 --set-selection
   =target=crop,flags=0,top=0,left=0,width=1920,height=1080 --stream-to=./ou
   t.yuv
2  VIDIOC_QUERYCAP: ok
3  VIDIOC_G_FMT: ok
4  VIDIOC_S_FMT: ok
5  Format Video Capture Multiplanar:
6      Width/Height      : 1920/1080
7      Pixel Format       : 'NV12' (Y/CbCr 4:2:0)
8      Field              : None
9      Number of planes   : 1
10     Flags              :
11     Colospace           : Default
12     Transfer Function   : Default
13     YCbCr/HSV Encoding : Default
14     Quantization        : Full Range
15     Plane 0             :
16         Bytes per Line : 1920
17         Size Image      : 3110400
18  VIDIOC_G_SELECTION: ok
19  VIDIOC_S_SELECTION: ok
20      VIDIOC_REQBUFS returned 0 (Success)
21      VIDIOC_QUERYBUF returned 0 (Success)
22      VIDIOC_QUERYBUF returned 0 (Success)
23      VIDIOC_QUERYBUF returned 0 (Success)
24      VIDIOC_QUERYBUF returned 0 (Success)
25      VIDIOC_QBUF returned 0 (Success)
26      VIDIOC_QBUF returned 0 (Success)
27      VIDIOC_QBUF returned 0 (Success)
28      VIDIOC_QBUF returned 0 (Success)
29  [ 113.761757] rkisp_hw fdff0000.rkisp: set isp clk = 297000000Hz
30  [ 113.774118] rockchip-csi2-dphy0: dphy0, data_rate_mbps 420
31  [ 113.774173] rockchip-csi2-dphy csi2-dphy0: csi2_dphy_s_stream stream o
   n:1, dphy0
32  [ 113.774186] ov5648 4-0036: ov5648_s_stream(878) enter!
33  [ 113.774193] ov5648 4-0036: stream on!!!
34  [ 113.790858] ov5648 4-0036: ov5648_set_ctrl Unhandled id:0x9f0905, val:0
   x400
35      VIDIOC_STREAMON returned 0 (Success)
36  cap dqbuf: 0 seq:      0 bytesused: 3110400 ts: 166.994238 (ts-monotonic,
   ts-src-eof)
37  cap dqbuf: 1 seq:      1 bytesused: 3110400 ts: 167.060730 delta: 66.492 m
   s (ts-monotonic, ts-src-eof)
38

```

```

39 cap dqbuf: 2 seq:      2 bytesused: 3110400 ts: 167.127250 delta: 66.520 m
   s (ts-monotonic, ts-src-eof)
40 cap dqbuf: 3 seq:      3 bytesused: 3110400 ts: 167.193752 delta: 66.502 m
   s (ts-monotonic, ts-src-eof)
41 cap dqbuf: 0 seq:      4 bytesused: 3110400 ts: 167.260267 delta: 66.515 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
42 cap dqbuf: 1 seq:      5 bytesused: 3110400 ts: 167.326780 delta: 66.513 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
43 cap dqbuf: 2 seq:      6 bytesused: 3110400 ts: 167.393286 delta: 66.506 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
44 cap dqbuf: 3 seq:      7 bytesused: 3110400 ts: 167.459802 delta: 66.516 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
45 cap dqbuf: 0 seq:      8 bytesused: 3110400 ts: 167.526312 delta: 66.510 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
46 cap dqbuf: 1 seq:      9 bytesused: 3110400 ts: 167.592825 delta: 66.513 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
47 cap dqbuf: 2 seq:     10 bytesused: 3110400 ts: 167.659336 delta: 66.511 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
48 cap dqbuf: 3 seq:     11 bytesused: 3110400 ts: 167.725846 delta: 66.510 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
49 cap dqbuf: 0 seq:     12 bytesused: 3110400 ts: 167.792353 delta: 66.507 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
50 cap dqbuf: 1 seq:     13 bytesused: 3110400 ts: 167.858870 delta: 66.517 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
51 cap dqbuf: 2 seq:     14 bytesused: 3110400 ts: 167.925377 delta: 66.507 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
52 cap dqbuf: 3 seq:     15 bytesused: 3110400 ts: 167.991892 delta: 66.515 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
53 cap dqbuf: 0 seq:     16 bytesused: 3110400 ts: 168.058402 delta: 66.510 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
54 cap dqbuf: 1 seq:     17 bytesused: 3110400 ts: 168.124914 delta: 66.512 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
55 cap dqbuf: 2 seq:     18 bytesused: 3110400 ts: 168.191427 delta: 66.513 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
56 cap dqbuf: 3 seq:     19 bytesused: 3110400 ts: 168.257938 delta: 66.511 m
   s fps: 15.04 (ts-monotonic, ts-src-eof)
57 cap dqbuf: 0 seq:     20 bytesused: 3110400 ts: 168.324442 delta: 66.504 m
58 s fps: 15.04 (ts-monotonic, ts-src-eof)

```

11、RTC

主板包含2个RTC，其中/dev/rtc1为外部RTC（HYM8563），/dev/rtc0为CPU内部的RTC（RK808）。系统默认使用rtc0的时间。所以这里有两种解决方法，如果rtc-hym8563是rtc0，则直接

设置即可

```
▼ Bash |
1 root@rk3568-buildroot:/# dmesg | grep rtc
2 [ 1.583028] rk808-rtc rk808-rtc: registered as rtc0
3 [ 1.584797] rk808-rtc rk808-rtc: setting system clock to 2017-08-04T09:00:03 UTC (1501837203)
4 [ 1.601112] rtc-hym8563 5-0051: registered as rtc1
5
```

方法一

11.1 获取RTC时间

```
▼ Bash |
1 root@rk3568-buildroot:/# hwclock
2 Fri Aug 4 09:00:53 2017 0.000000 seconds
```

11.2 设置RTC时间

```
▼ Bash |
1 root@rk3568-buildroot:/# date -s '2000-01-30 1:1:1'
2 Sun Jan 30 01:01:01 UTC 2000
3 root@rk3568-buildroot:/# hwclock -w -f /dev/rtc1
4 root@rk3568-buildroot:/# hwclock -r -f /dev/rtc1
5 Sun Jan 30 01:01:11 2000 0.000000 seconds
```

断电重新上电，我们可以看到时间又被复原，我们直接

```

1 root@rk3568-buildroot:/# date
2 Fri Aug  4 09:00:17 UTC 2017
3
4 //写入系统时间
5 root@rk3568-buildroot:/# hwclock --hctosys --rtc=/dev/rtc1
6 root@rk3568-buildroot:/# date
7 Sun Jan 30 01:03:58 UTC 2000
8 root@rk3568-buildroot:/# hwclock -r -f /dev/rtc1
9 Sun Jan 30 01:04:17 2000 0.000000 seconds
10

```

方法二

如果不想这么麻烦的话，在内核中找到CONFIG_RTC_DRV_RK808把他关掉就行

```

1 #CONFIG_RTC_DRV_RK808

```

这时，外部rtc的节点就是外部RTC（HYM8563），也是系统默认使用的rtc，我们常规设置就可以

```

1 root@rk3568-buildroot:/# hwclock
2 Fri Aug  4 09:07:26 2017 0.000000 seconds
3
4 //设置系统时间
5 root@rk3568-buildroot:/# date -s '2023-12-19 16:15:20'
6 Tue Dec 19 16:15:20 UTC 2023
7
8 //把系统时间写入rtc
9 root@rk3568-buildroot:/# hwclock -w
10 root@rk3568-buildroot:/# hwclock -r
11 Tue Dec 19 16:15:25 2023 0.000000 seconds
12
13 //断电重启后，直接hwclock就会把rtc时间写入系统
14 root@rk3568-buildroot:/# hwclock
15 Tue Dec 19 16:16:55 2023 0.000000 seconds
16
17

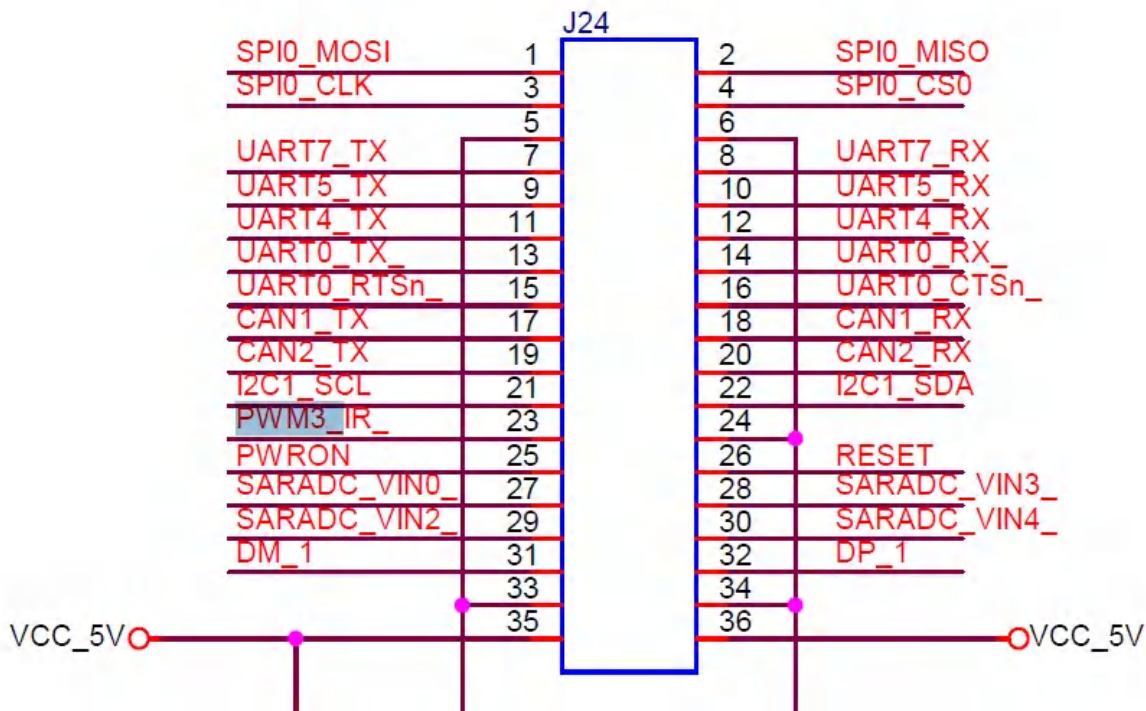
```

12、开机自启动

默认系统开机会运行/etc/init.d/Sxxx脚本，默认是执行S开头的脚本，将要开机执行的程序放到该脚本中即可。

13、PWM功能

PWM位于J24双排针。



《双排针图》

序号	定义	电平	说明
23	PWM3_IR_	3.3V	

13.1 测试

▼

Bash

```
1 cd /sys/class/pwm/pwmchip0/
```

开启PWM:

▼ Bash |

```
1 echo 0 > export
2 cd pwm0
```

设置周期:

▼ Bash |

```
1 echo 330000 > period
```

设置占空比:

▼ Bash |

```
1 echo 150000 > duty_cycle
```

使能PWM:

▼ Bash |

```
1 echo 1 > enable
```

失能PWM:

▼ Bash |

```
1 echo 0 > enable
```

14、屏幕控制

14.1 背光调节

通过修改/sys/class/backlight/backlight/brightness的值，实现背光的调节，范围取0–255，值越大，亮度越高。（/backlight/brightness没有这个）（backlight/brightness没找到）

设置亮度为100:

```
1 root@rk3568-buildroot:/# echo 100 > /sys/class/backlight/backlight/brightness
```

15、按键

主板配置了一个按键SW2，作为电源键使用，对应的设备节点为/dev/input/event0。

系统运行时，短按该按键上报KEY_POWER，并且进入待机状态。

系统待机时，短按该按键，系统恢复正常运行。

系统运行时，长按该按键5秒关机。

系统关机时，短按该按键开机。

16、ADC

主板配置了4路ADC，位于J24的第7.8.9,10引脚，分别记作ADC0、ADC2、ADC3、ADC4，精度为10位。

16.1 ADC转换方法

$$V = (raw/1024)*1.8v$$

其中raw为对应设备节点读取的值，范围为0-1023。

序号	编号	设备节点
9	SARADC VINU0_	/sys/bus/iio/devices/iio:device0/in_voltage0_raw
7	SARADC VINU2_	/sys/bus/iio/devices/iio:device0/in_voltage2_raw
10	SARADC VINU3_	/sys/bus/iio/devices/iio:device0/in_voltage3_raw
8	SARADC VINU4_	/sys/bus/iio/devices/iio:device0/in_voltage4_raw

16.2 测试

以测试ADC2为例，其余ADC测试方法类似。

Bash

```
1 root@rk3568-buildroot:/# cat /sys/bus/iio/devices/iio:device0/in_voltage3_raw
2 997
```

设备节点读取的raw值为997，代入到公式计算：

$$V=(997/1024)*1.8v=1.75v$$

即ADC1输入的电压为1.75v。

17、网络优先级设置

主板支持以太网、WiFi和4G/5G三种网络，通过路由表来设置它们的网络优先级。

17.1 查看路由表

Bash

```
1 root@rk3568-buildroot:/# route
2 Kernel IP routing table
3 Destination      Gateway            Genmask           Flags Metric Ref    Use Ifa
4 default          _gateway          0.0.0.0           UG    100   0      0 eth
5 default          _gateway          0.0.0.0           UG    600   0      0 wla
6 192.168.1.0      0.0.0.0           255.255.255.0    U     100   0      0 eth
7 192.168.1.0      0.0.0.0           255.255.255.0    U     600   0      0 wla
```

17.2 设置默认路由

17.2.1 设置WiFi为默认路由

当前以太网和WiFi同时使用，设置WiFi优先：

▼

Bash

```
1 root@rk3568-buildroot:/# route
2 Kernel IP routing table
3 Destination      Gateway            Genmask           Flags Metric Ref    Use If
4 default          _gateway          0.0.0.0           UG      100    0      0 et
5 default          _gateway          0.0.0.0           UG      600    0      0 wl
6 192.168.1.0      0.0.0.0           255.255.255.0    U       100    0      0 et
7 192.168.1.0      0.0.0.0           255.255.255.0    U       600    0      0 wl
8 root@rk3568-buildroot:/# route del default dev eth0
9 root@rk3568-buildroot:/# route
10 Kernel IP routing table
11 Destination      Gateway            Genmask           Flags Metric Ref    Use If
12 default          _gateway          0.0.0.0           UG      600    0      0 wl
13 192.168.1.0      0.0.0.0           255.255.255.0    U       100    0      0 et
14 192.168.1.0      0.0.0.0           255.255.255.0    U       600    0      0 wl
```

这样默认路由就是wlan0了，即优先使用WiFi进行数据通信。

17.2.2 设置以太网为默认路由

当前以太网和WiFi同时使用，且WiFi优先：

▼

Bash

```
1 root@rk3568-buildroot:/# route
2 Kernel IP routing table
3 Destination      Gateway            Genmask           Flags Metric Ref    Use Ifa
4 default          _gateway          0.0.0.0           UG      600    0      0 wla
5 192.168.1.0      0.0.0.0           255.255.255.0    U       100    0      0 eth
6 192.168.1.0      0.0.0.0           255.255.255.0    U       600    0      0 wla
7 root@rk3568-buildroot:/#
```

设置为以太网优先：

Bash

```

1 root@rk3568-buildroot:/# route del default dev wlan0
2 root@rk3568-buildroot:/# route add default dev eth0
3 root@rk3568-buildroot:/# route add default gw 192.168.1.1
4 root@rk3568-buildroot:/# route
5 Kernel IP routing table
6 Destination      Gateway            Genmask           Flags Metric Ref    Use If
7 default          0.0.0.0           0.0.0.0           U        0      0      0 et
8 192.168.1.0      0.0.0.0           255.255.255.0     U        100    0      0 et
9 192.168.1.0      0.0.0.0           255.255.255.0     U        600    0      0 wl
10 root@rk3568-buildroot:/#

```

其他情况按照类似的方法进行处理即可。

18、CAN

CAN位于J24的双排针。共有3路CAN可供使用。

序号	编号	描述
19	CAN1_TX	CAN1
20	CAN1_RX	
17	CAN2_TX	CAN2
18	CAN2_RX	
15	CAN0_TX (I2C_SCL)	CAN0 当作为CAN0时需要关闭I2C1
16	CAN0_RX (I2C_SDA)	

注意：由于开发板未带有CAN芯片，所以这里需要用到转接板进行测试。

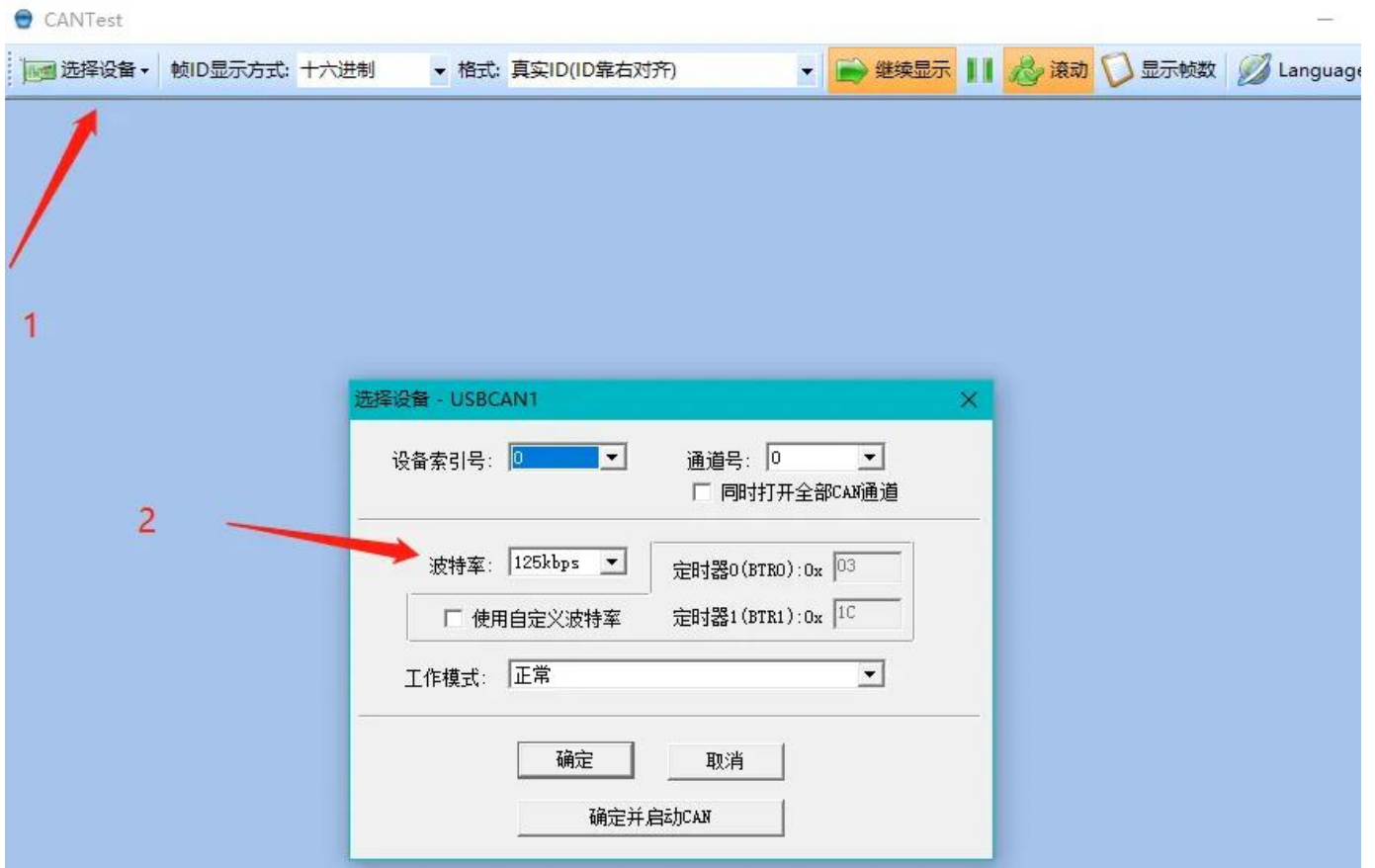
18.1测试

测试需要用USB转CAN工具，通过PC与板上CAN通信。

这里以CAN0为例，其余节点测试方法相同：

```
1 //关掉can
2 ifconfig can0 down
3
4 //配置can通信的波特率
5 ip link set can0 type can bitrate 125000 triple-sampling on
6
7 //开启can通信
8 ifconfig can0 up
9
10 //作为接收端接收数据
11 candump can0
12
13 //作为发送端发送数据
14 cansend can0 5A1#1122334455667788
15
```

(1) 选择USBCAN1



(2) 启动CAN测试

The screenshot displays the CANtest application window. The main area shows a list of CAN frames with columns for sequence number, transmission direction, time, frame ID, frame format, frame type, data length, and data (HEX). A red arrow points to the '停止' (Stop) button in the top toolbar. The bottom panel, titled '基本操作' (Basic Operation), contains settings for transmission mode, frame type, frame ID, data, and a '发送' (Send) button, which is also highlighted with a red arrow.

序号	传输方向	时间戳	帧ID	帧格式	帧类型	数据长度	数据(HEX)
00000073	接收	17:54:33.4...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000074	接收	17:54:33.7...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000075	接收	17:54:34.1...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000076	接收	17:54:34.5...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000077	发送	19:15:27.3...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000078	发送	19:15:27.8...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000079	发送	19:15:28.0...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000080	发送	19:15:28.2...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000081	发送	19:15:28.4...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000082	接收	19:15:34.1...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000083	接收	19:15:34.4...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000084	接收	19:15:35.2...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000085	接收	19:15:35.7...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000086	发送	19:16:07.6...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000087	发送	19:16:08.6...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000088	发送	19:16:08.9...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000089	发送	19:16:09.5...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000090	发送	19:16:16.0...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000091	发送	19:16:16.2...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000092	发送	19:16:16.4...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000093	发送	19:16:16.6...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000094	发送	19:16:16.8...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000095	发送	19:16:17.0...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000096	发送	19:16:40.3...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000097	发送	19:16:41.0...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000098	发送	19:16:41.2...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000099	发送	19:16:41.7...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000100	发送	19:16:47.5...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000101	发送	19:16:47.7...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000102	发送	19:16:47.9...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000103	发送	19:16:48.1...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000104	发送	19:16:48.3...	0x00000000	数据帧	标准帧	0x08	00 01 02 03 04 05 06 07
00000105	接收	19:16:54.9...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000106	接收	19:16:55.4...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000107	接收	19:16:56.1...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
00000108	接收	19:16:56.7...	0x000005a1	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88

基本操作

发送方式: 正常发送 ☒ 每次发送单帧 ☐ 每次发送 10 帧 ☐ 帧ID每发送一帧递增

帧类型: 标准帧 帧ID(HEX): 00000000 数据(HEX): 00 01 02 03 04 05 06 07 发送

帧格式: 数据帧 发送次数: 1 每次发送间隔(ms): 0 停止

板端发送过来的数据可以在CANtest上打印出来。