百问网 Linux 开发板硬件测试说明

——100ASK_IMX6ULL_Mini

深圳百问网科技有限公司

2022-9-13

[1.0]

更新记录

日期	更改内容	版本号	更新者	审核
2022/9/13	初稿	V1.0	百问网研发团队	

目录

更新记录 1
目录 1
第 1 章 有线网卡接口测试 1
1.1 ifconfig 查看 IP 1
1.2 udhcpc 获取 IP 2
1.3 ifconfig 强制指定 IP 2
1.4 网络连通性测试 3
1.5 使能网卡一接口 4
第 2 章 USB Host 接口测试 6
第 3 章 耳机接口测试方法 8
3.1 录制音频： 8
3.2 播放音频： 8
第 4 章 LCD 显示测试 11
4.1 lcd 显示红色 11
4.2 lcd 显示多种颜色 11
第 5 章 触摸屏测试 12
第 6 章 屏幕背光调节 13
第 7 章 RTC 测试 13
第 8 章 RS485 测试 15
第 9 章 CAN 功能测试 16
第 10 章 Key(按键)测试 17
第 11 章 查看 CPU 温度 17

第1章 有线网卡接口测试

此节演示在串口终端下如何设置开发板的 ip 地址，测试网络的连通性。

既然是在开发板和电脑之间测试网络，那双方需要有网络连接。可以使用一个路由器，开发板通过网线与路由器连接，而电脑与路由器之间，可以使用网线连接，也可以使用 WIFI 连接。

注意：如果要测试全功能版的 2 个网卡，先测一个网卡，然后把它的网线取下来，再接网线到第 2 个网卡并测试。

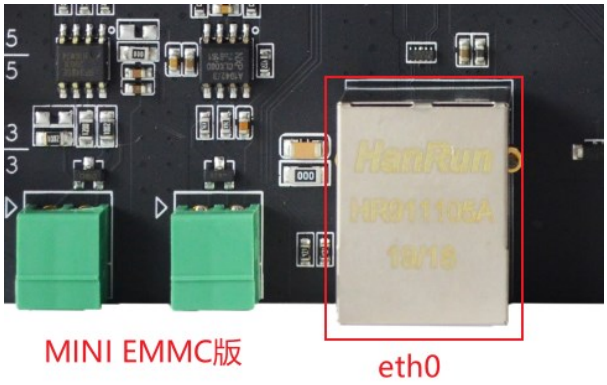


图 1.1 IMX6ULL Mini 开发板的网口

1.1 ifconfig 查看 IP

使用串口登录开发板后，在串口助手命令行输入 ifconfig 查看 IP：

```
[root@100ask:~]# ifconfig
```

会得到类似如下结果：

```
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 169.254.118.140  netmask 255.255.0.0  broadcast 169.254.255.255
        inet6 fe80::63cc:c589:dc93:2607  prefixlen 64  scopeid 0x20<link>
        ether 76:05:eb:9b:8f:c4  txqueuelen 1000  (Ethernet)
        RX packets 72  bytes 4062 (3.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 27  bytes 2849 (2.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 38  bytes 3436 (3.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 38  bytes 3436 (3.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 169.254.108.15  netmask 255.255.0.0  broadcast 169.254.255.255
        inet6 fe80::2989:b098:bde9:7eb8  prefixlen 64  scopeid 0x20<link>
        ether b0:02:47:38:57:6e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 87  bytes 6119 (5.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

通过上图可知，开发板已经自动获得 IP 地址 169.254.118.140(你的开发板自动获取的 IP 可能不一样)。

1.2 udhcpc 获取 IP

如果开发板未能获取 IP，则可以使用 udhcpc 命令尝试获取 IP：

```
[root@100ask:~]# udhcpc -i eth0
```

会得到类似如下的结果：

```
udhcpc: started, v1.31.1
[  365.415081] stm32-dwmac 5800a000.ethernet eth0: PHY [stmmac-0:06] driver [Generi
c PHY]
[  365.477538] dwmac4: Master AXI performs any burst length
[  365.481505] stm32-dwmac 5800a000.ethernet eth0: No Safety Features support found
[  365.488844] stm32-dwmac 5800a000.ethernet eth0: IEEE 1588-2008 Advanced Timestam
p supported
[  365.502941] stm32-dwmac 5800a000.ethernet eth0: registered PTP clock
[  365.507899] stm32-dwmac 5800a000.ethernet eth0: configuring for phy/rgmii link m
ode
[  365.533801] 8021q: adding VLAN 0 to HW filter on device eth0
udhcpc: sending discover
[  368.651556] stm32-dwmac 5800a000.ethernet eth0: Link is Up - 1Gbps/Full - flow c
ontrol rx/tx
[  368.658864] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.50.125
udhcpc: received DHCP NAK
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.50.126
udhcpc: lease of 192.168.50.126 obtained, lease time 86400
deleting routers
SIOCADDRT: Network is unreachable
adding dns 192.168.50.1
```

那么开发板分配到的 ip 就是 192.168.50.126。

1.3 ifconfig 强制指定 IP

如果通过 udhcpc 命令无法获得 IP，也可以使用 ifconfig 命令强制设置 IP（一般情况下不用），如下图使用 ifconfig 命令强制指定 IP 地址为 192.168.5.10：

```
[root@100ask:~]# ifconfig eth1 192.168.5.10
```

再用 ifconfig 查询 ip 的时候就会得到如下结果：

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.126  netmask 255.255.255.0  broadcast 192.168.50.255
        inet6 fe80::ae54:676e:dd7a:c583  prefixlen 64  scopeid 0x20<link>
        ether 00:01:1f:2d:3e:4d  txqueuelen 1000  (Ethernet)
        RX packets 781  bytes 55612 (54.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 170  bytes 16934 (16.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 51

eth1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
```

```
        inet 192.168.5.10  netmask 255.255.255.0  broadcast 192.168.5.255
        ether 76:05:eb:9b:8f:c4  txqueuelen 1000  (Ethernet)
        RX packets 600  bytes 29553 (28.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 31  bytes 4381 (4.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 230  bytes 21877 (21.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 230  bytes 21877 (21.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 169.254.108.15  netmask 255.255.0.0  broadcast 169.254.255.255
        inet6 fe80::2989:b098:bde9:7eb8  prefixlen 64  scopeid 0x20<link>
        ether b0:02:47:38:57:6e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 354  bytes 20402 (19.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

可以看到 eth0 是自动获取的 IP，而 eth1 则是我们手动指定的 ip: 192.168.5.10。

1.4 网络连通性测试

在开发板上执行如下命令，如果有数据返回则表示开发板跟互联网是连通的(前提是你的路由器是可以上网的)：

```
[root@100ask:~]# ping www.baidu.com
```

可以得到如下结果：

```
PING www.baidu.com (183.232.231.174): 56 data bytes
64 bytes from 183.232.231.174: seq=0 ttl=54 time=32.148 ms
64 bytes from 183.232.231.174: seq=1 ttl=54 time=31.771 ms
64 bytes from 183.232.231.174: seq=2 ttl=54 time=31.274 ms
64 bytes from 183.232.231.174: seq=3 ttl=54 time=30.633 ms
64 bytes from 183.232.231.174: seq=4 ttl=54 time=30.147 ms
```

这就表示能联通外网。

当然，很多时候开发板不能 ping 通互联网，这也没关系，只要能 ping 通 Windows 或是 Windows 能 ping 通开发板就可以(Windows 开了防火墙时，开发板无法 ping 通 windows)。

比如我们的 Windows IP 地址为 192.168.50.240，此时可以通过 ping 命令测试两者是否可以相互通信：

- **开发板 pingWindows**

```
[root@100ask:~]# ping 192.168.50.240
```

能通信情况下会得到类似如下的结果：

```
PING 192.168.50.240 (192.168.50.240): 56 data bytes
64 bytes from 192.168.50.240: seq=0 ttl=128 time=138.926 ms
64 bytes from 192.168.50.240: seq=1 ttl=128 time=2.236 ms
64 bytes from 192.168.50.240: seq=2 ttl=128 time=1.937 ms
64 bytes from 192.168.50.240: seq=3 ttl=128 time=1.846 ms
```

- Windows ping 开发板

```
PS C:\Users\Administrator> ping 192.168.50.126
```

能通信情况下会得到类似如下的情况：

```
正在 Ping 192.168.50.126 具有 32 字节的数据:
来自 192.168.50.126 的回复: 字节=32 时间=3ms TTL=64
来自 192.168.50.126 的回复: 字节=32 时间=1ms TTL=64
来自 192.168.50.126 的回复: 字节=32 时间=8ms TTL=64
来自 192.168.50.126 的回复: 字节=32 时间=1ms TTL=64

192.168.50.126 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 1ms, 最长 = 8ms, 平均 = 3ms
```

1.5 使能网卡一接口

开发板有 2 个网口：eth0、eth1。使用 ifconfig -a 查看都有那些网卡设备：

```
[root@100ask:~]# ifconfig -a
```

会看到下面这些信息：

```
can0: flags=128<NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (UNS
PEC)
        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
        device interrupt 55

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.125  netmask 255.255.255.0  broadcast 192.168.50.255
        inet6 fe80::ae54:676e:dd7a:c583  prefixlen 64  scopeid 0x20<link>
        ether 00:01:1f:2d:3e:4d  txqueuelen 1000  (Ethernet)
        RX packets 73  bytes 4790 (4.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 27  bytes 2236 (2.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 51

eth1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 76:05:eb:9b:8f:c4  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

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2  bytes 196 (196.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 196 (196.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

```
        inet 169.254.108.15  netmask 255.255.0.0  broadcast 169.254.255.255
        inet6 fe80::2989:b098:bde9:7eb8  prefixlen 64  scopeid 0x20<link>
        ether b0:02:47:38:57:6e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28  bytes 2959 (2.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图 1.2 ifconfig -a

把网线接入 eth1 网口，最好把 eth0 网口的网线取下。执行以下命令启用 eth1 网卡设备：

```
[root@100ask:~]# ifconfig eth1 up
```

如果网线另一端接入的也是路由器，可能会有如下的 log：

```
[root@100ask:~]# [  499.936310] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
[  499.943995] r8152 2-1.2:1.0 eth1: carrier on
```

并使用 udhcpc 自动获取 ip 地址：

```
[root@100ask:~]# udhcpc -i eth1
```

如果连接的是路由器，能正常分配 ip 的情况下会得到如下的 log 信息：

```
udhcpc: started, v1.31.1
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.50.156
udhcpc: lease of 192.168.50.156 obtained, lease time 86400
deleting routers
adding dns 192.168.50.1
```

第2章 USB Host 接口测试

此节演示在终端下如何在 USB Host 接口上使用 usb 存储设备。

注意：需要准备一个 USB 设备，比如 U 盘、USB 蓝牙模块、usb 网卡或者 usb 摄像头等。

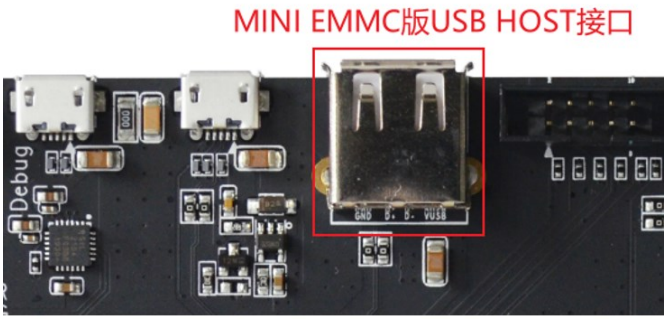


图 2.1

下面使用一个 U 盘作为例子，插到任意一个 USB Host 接口，会打印出如下设备信息：

```
[root@100ask:~]# [ 2724.499743] usb 2-1.3: new high-speed USB device number 6 using
 ehci-platform
[ 2724.551550] usb 2-1.3: New USB device found, idVendor=14cd, idProduct=1212, bcdD
evice= 1.00
[ 2724.558501] usb 2-1.3: New USB device strings: Mfr=1, Product=3, SerialNumber=2
[ 2724.565972] usb 2-1.3: Product: Mass Storage Device
[ 2724.570848] usb 2-1.3: Manufacturer: Generic
[ 2724.575043] usb 2-1.3: SerialNumber: 121220160204
[ 2724.587424] usb-storage 2-1.3:1.0: USB Mass Storage device detected
[ 2724.596225] scsi host0: usb-storage 2-1.3:1.0
[ 2725.621803] scsi 0:0:0:0: Direct-Access     Mass     Storage Device   1.00 PQ: 0
 ANSI: 0 CCS
[ 2725.636235] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 2725.856558] sd 0:0:0:0: [sda] 61069312 512-byte logical blocks: (31.3 GB/29.1 Gi
B)
[ 2725.864506] sd 0:0:0:0: [sda] Write Protect is off
[ 2725.869095] sd 0:0:0:0: [sda] No Caching mode page found
[ 2725.873449] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 2725.929946]  sda: sda1
[ 2725.938116] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 2726.003471] systemd-journald[199]: /dev/kmsg buffer overrun, some messages lost.
[ 2726.009544] systemd-journald[199]: /dev/kmsg buffer overrun, some messages lost.
……//后面还有很多，这里省略
```

通过打印的设备信息可知，系统为该 usb 存储设备创建的设备节点为/dev/sda。一般来说/dev/sda 对应整个 U 盘，/dev/sda1 对应该 U 盘的第 1 个分区，/dev/sda2 对应第 2 个分区。

有些 U 盘没有划分分区，它只有一个设备节点/dev/sda，而没有/dev/sda1 等节点。对于这种情况，/dev/sda 既代表整个 U 盘，也代表第 1 个分区。

我们可以挂载某个分区，挂载之前要先通过 fdisk 命令获取分区类型,如下所示：

```
[root@100ask:~]# fdisk -l /dev/sda
```

会得到如下的信息：

```
Disk /dev/sda: 29.12 GiB, 31267487744 bytes, 61069312 sectors
```

```
Disk model: Storage Device
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: 0x6ba2086e

Device     Boot Start      End  Sectors  Size Id Type
/dev/sda1        8192 61069311 61061120 29.1G  c W95 FAT32 (LBA
```

从上图可知/dev/sda1 是 FAT32，挂载时可以指定类型为“vfat”：

```
[root@100ask:~]# mount -t vfat /dev/sda1 /mnt
[root@100ask:~]# ls /mnt/
OBS-Studio-25.0.8-Full-Installer-x64 (1).exe  System Volume Information  network
[root@100ask:~]# [ 2484.098433] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.109435] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.119960] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.130282] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.138322] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.150005] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.157570] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.169365] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.176891] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
[ 2484.200190] systemd-journald[782]: /dev/kmsg buffer overrun, some messages lost.
```

图 2.2 挂载分区

注意：暂不支持分区类型为 NTFS 的 U 盘。

测试完以后，通过 umount 卸载/mnt，才可拔下 usb 设备：

```
[root@100ask:~]# umount  /mnt
```

第3章 耳机接口测试方法

此节演示使用三段式耳机在 100ask_stm32mp157_pro 开发板上录制声音、播放音频。

注意:需要准备一个带麦克风的三段式耳机，如下图所示：

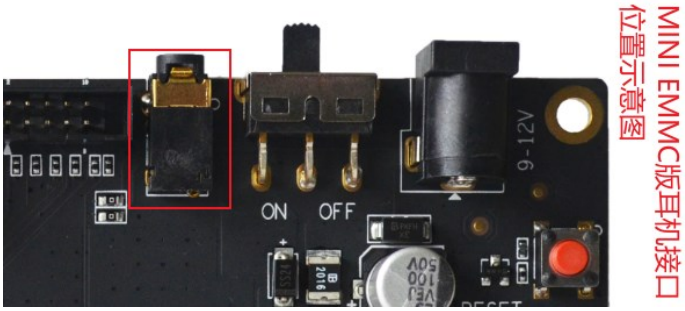


图 3.1

3.1 录制音频：

将耳机插入开发板耳机孔，使用如下命令进行录制(执行命令后，对着麦克风说话)：

```
[root@100ask:~]# arecord -v  --format=cd --device=plughw:0,1  test.wav
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'WM8960-100ASK' device 1 subdevice 0
Its setup is:
  stream       : CAPTURE
  access       : RW_INTERLEAVED
  format       : S16_LE
  subformat    : STD
  channels     : 2
  rate         : 44100
  exact rate   : 44100 (44100/1)
  msbits       : 16
  buffer_size  : 8192
  period_size  : 1024
  period_time  : 23219
  tstamp_mode  : NONE
  tstamp_type  : MONOTONIC
  period_step  : 1
  avail_min    : 1024
  period_event : 0
  start_threshold  : 1
  stop_threshold   : 8192
  silence_threshold: 0
  silence_size : 0
  boundary     : 1073741824
  appl_ptr     : 0
  hw_ptr       : 0
Aborted by signal Interrupt...
```

3.2 播放音频：

将耳机插入开发板耳机孔，使用 aplay 进行播放音频文件：

```
[root@100ask:~]# aplay -v --format=cd  --device=plughw:0,0  test.wav
```

```
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'WM8960-100ASK' device 0 subdevice 0
Its setup is:
  stream       : PLAYBACK
  access       : RW_INTERLEAVED
  format       : S16_LE
  subformat    : STD
  channels     : 2
  rate         : 44100
  exact rate   : 44100 (44100/1)
  msbits       : 16
  buffer_size  : 8192
  period_size  : 1024
  period_time  : 23219
  tstamp_mode  : NONE
  tstamp_type  : MONOTONIC
  period_step  : 1
  avail_min    : 1024
  period_event : 0
  start_threshold  : 8192
  stop_threshold   : 8192
  silence_threshold: 0
  silence_size : 0
  boundary     : 1073741824
  appl_ptr     : 0
  hw_ptr       : 0
```

如果戴上耳机没有听到声音，执行 alsamixer 将最左边的音量调到最大：

```
[root@100ask:~]# alsamixer
```

如下图：

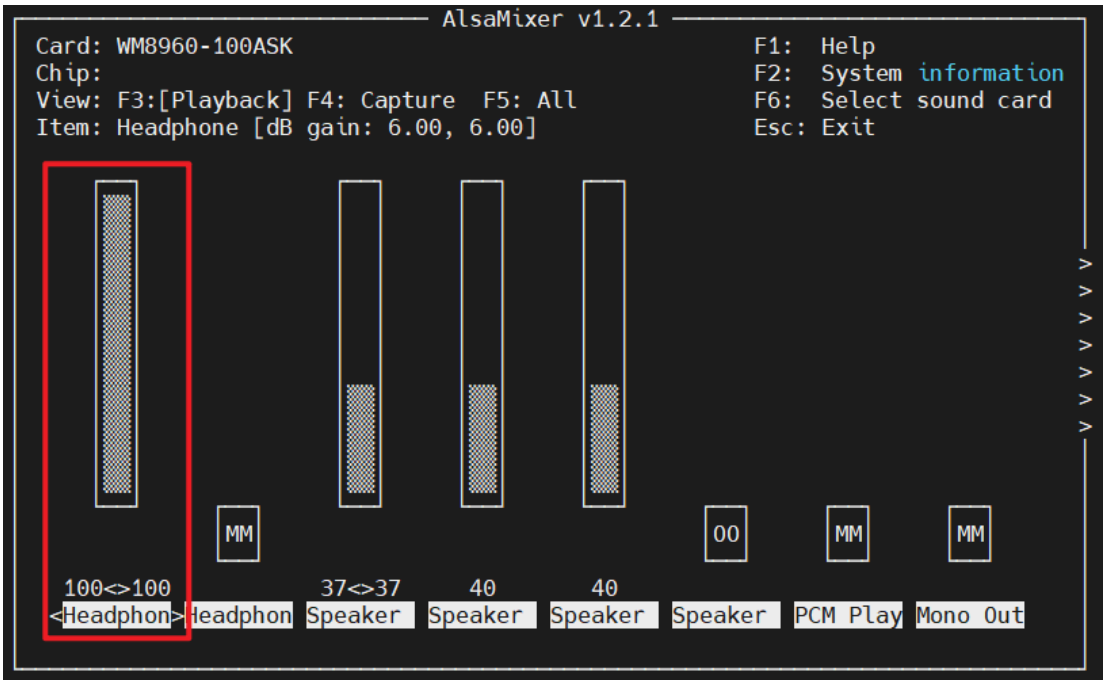


图 3.2

注意：录音后再播放所录得的音频文件，只有一边耳朵有声音，因为只有一个麦克采集单声道数据。还可以通过 ssh 登录开发板，将电脑中的 wav 格式的音频上传到开发板，再用 aplay 进行播放。也可以执行以下命令，让 2 只耳朵都能听到声音：

```
[root@100ask:~]# speaker-test  -t  wav  -c  2  -D  plughw:0,0
```

会得到这些信息，并且听到声音：

```
speaker-test 1.2.1

Playback device is plughw:0,0
Stream parameters are 48000Hz, S16_LE, 2 channels
WAV file(s)
Rate set to 48000Hz (requested 48000Hz)
Buffer size range from 512 to 8192
Period size range from 256 to 1024
Using max buffer size 8192
Periods = 4
was set period_size = 1024
was set buffer_size = 8192
 0 - Front Left
 1 - Front Right
```

第4章 LCD 显示测试

注意：此章节测试需要外接 LCD 屏幕才可以进行测试验证，LCD 模块介绍请参考页面
http://download.100ask.org/modules/Lcd/100ask_imx6ull_7-inch_LCD/
此节演示通过 fb-test 测试程序让 lcd 显示红绿蓝白 4 中颜色，用以观察 lcd 的显示效果。

在测试前最好先关闭 UI：

- **LVGL GUI**

```
[root@100ask:~]# mv /etc/init.d/S05lvgl /root
```

- **米尔 GUI**

```
[root@100ask:~]# mv  /etc/init.d/S99myirhmi2 /root
```

然后重启开发板

```
[root@100ask:~]# reboot
```

4.1 lcd 显示红色

```
[root@100ask:~]# fb-test  -r
```

log:

```
fb-test 1.1.0 (rosetta)
fb res 1024x600 virtual 1024x600, line_len 4096, bpp 32
```

现象：屏幕显示纯红色

4.2 lcd 显示多种颜色

```
[root@100ask:~]# fb-test
```

log:

```
fb-test 1.1.0 (rosetta)
fb res 1024x600 virtual 1024x600, line_len 4096, bpp 32
```

现象：开发板显示红绿蓝三色等多种彩色。

第5章 触摸屏测试

注意：此章节测试需要外接 LCD 屏幕才可以进行测试验证，LCD 模块介绍请参考页面
http://download.100ask.org/modules/Lcd/100ask_imx6ull_7-inch_LCD/
触摸屏能点击的话，就表示它没问题。另外，电容屏不需要较准。如果你就是想走一遍，请按下面方法：

最好先关闭 gui：

- LVGL GUI

```
[root@100ask:~]# mv /etc/init.d/S05lvgl /root
```

- 米尔 GUI

```
[root@100ask:~]# mv  /etc/init.d/S99myirhmi2 /root
```

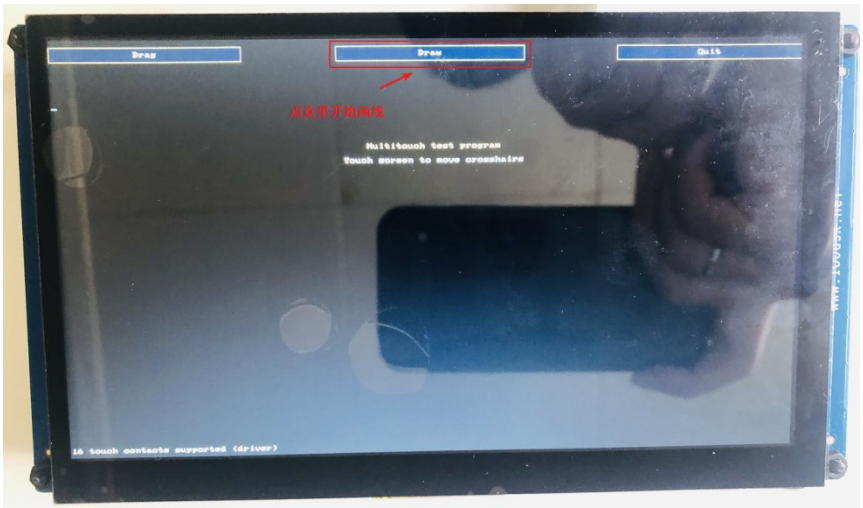
然后重启开发板

```
[root@100ask:~]# reboot
```

之后在终端执行以下命令：

```
[root@100ask:~]# export TSLIB_TSDEVICE=/dev/input/event1
[root@100ask:~]# export TSLIB_CONFFILE=/etc/ts.conf
[root@100ask:~]# export TSLIB_CALIBFILE=/etc/pointercal
[root@100ask:~]# export TSLIB_PLUGINDIR=/usr/lib/ts
[root@100ask:~]# export TSLIB_CONSOLEDEVICE=none
[root@100ask:~]# export QT_QPA_FB_TSLIB=1
[root@100ask:~]# export QT_QPA_GENERIC_PLUGINS=tslib:/dev/input/event1
[root@100ask:~]# ts_test_mt
```

ts_test_mt 用来测试电容触摸屏，可以在屏幕上点击、画线，如下图所示：
注意：暂时不支持多点触摸，以后录到电容屏驱动时再改进驱动程序。



恢复 GUI，在终端上执行如下命令，即可启动 qt gui 界面：

- LVGL GUI

```
[root@100ask:~]# mv /root/S05lvgl /etc/init.d
```

- 米尔 GUI

```
[root@100ask:~]# mv /root/S99myirhmi2 /etc/init.d
```

然后重启开发板

```
[root@100ask:~]# reboot
```

第6章 屏幕背光调节

注意：此章节测试需要外接 LCD 屏幕才可以进行测试验证，LCD 模块介绍请参考页面
http://download.100ask.org/modules/Lcd/100ask_imx6ull_7-inch_LCD/
此节演示通过操作 LCD 在/sys 目录下的对应文件，以实现查询、调节背光亮度。
目前背光亮度的设置范围有 0～8，0 表示关，其他值表示亮度值。
先通过 cat 命令查看当前背光亮度等级：

```
[root@100ask:~]# cat /sys/class/backlight/backlight/brightness
```

最后设置背光亮度值为 8，可以看到 LCD 亮了：

- 关闭背光

```
[root@100ask:~]# echo 0 > /sys/class/backlight/backlight/brightness
```

- 开启背光

```
[root@100ask:~]# echo 8 > /sys/class/backlight/backlight/brightness
```

第7章 RTC 测试

此节演示通过使用 date 和 hwclock 命令设置系统时间、硬件时间，并测试当操作系统重启后，系统时钟与硬件时间是否同步。

一般的板子都会有一个名为 RTC(实时时钟)的硬件，RTC 使用电池模块来供电，在系统关闭时用来维持时钟。RTC 维持的时间，被称为硬件时间。下图显示的是电池模块的安装位置：

MINI EMMC版rtc接口位置示意图

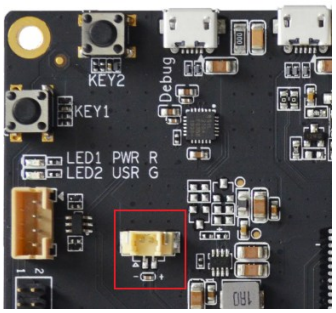


图 7.1

注意：我们没有提供 RTC 模块，需要的话自行购买(淘宝搜“1.25 bios 电池”)；不装 RTC 电池也可以做实验，但是断电重启后时间无法保存。

Linux 系统启动之后，它会自己维持时间，这个时间被称为系统时间。系统时间的初始值来源有二：

① 如果没有 RTC，系统时间初始值为 1970 年 1 月 1 日 0 点 0 分 0 秒
② 如果有 RTC，Linux 启动后，系统时间初始值从 RTC 读取

在实际使用过程中，要注意系统时间、硬件时间的同步问题：

① 使用 date 命令查看、设置系统时间，在设置系统时间后，要使用“hwclock -w” 命令同步到 RTC；
② 使用 hwclock 查看、设置硬件时间，在设置硬件时间后，要使用“hwclock -s”命令同步到系统时间。

以下命令是设置系统时间、并同步到 RTC：

```
[root@100ask:~]# date -s "2020-08-15 12:00:00"
```

log：

```
Sat Aug 15 12:00:00 UTC 2020
[root@100ask:~]# hwclock -w
```

一般不直接设置硬件时间，要设置硬件时间时，先使用 date 设置系统时间，再使用“hwclock -w”同步到 RTC 硬件。

你使用 date、hwclock 命令设置好时间后，可以关闭开发板并等待一会后重启，再用 date 命令查看时间是否正常。

对 RTC 硬件的操作使用 hwclock 命令，常见用法如下：

- hwclock -r：显示硬件时钟与日期

```
Sat Aug 15 12:02:12 2020  0.000000 seconds
Sat Aug 15 12:02:37 2020  0.000000 seconds
Sat Aug 15 12:02:42 2020  0.000000 seconds
```

- hwclock -s：将系统时钟调整为与目前的硬件时钟一致。
- hwclock -w：将硬件时钟调整为与目前的系统时钟一致。

第8章 RS485 测试

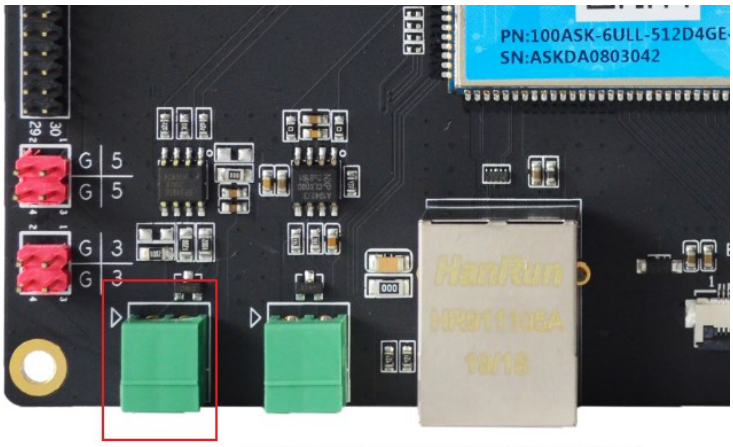
此节演示使用 rs485_test 程序测试 rs485 接口。

注意：rs485 通信时需要 2 个 rs485 设备，使用“3.5 15EDG 5P”插拔式接线端子进行连接测试。

以下测试是通过两块开发板进行测试，板子背面标有 RS485 引脚丝印“A B”，A 接 A、B 接 B。发货清单中不含接线端子，请自行购买连接测试，或者使用我们的 RS485 转 CAN 模块来测试学习。

模块地址：

http://download.100ask.org/modules/CommunicationModule/RS485-CAN/



MINI EMMC版RS485位置示意图

图 8.1

① 接收端：

```
[root@100ask:~]# echo 128 > /sys/class/gpio/export
[root@100ask:~]# echo out > /sys/class/gpio/gpio128/direction
[root@100ask:~]# echo 0 > /sys/class/gpio/gpio128/value
[root@100ask:~]# rs485_test  -d  /dev/ttySTM1 -b 115200
```

② 发送端：

```
[root@100ask:~]# echo 128 > /sys/class/gpio/export
[root@100ask:~]# echo out > /sys/class/gpio/gpio128/direction
[root@100ask:~]# echo 1 > /sys/class/gpio/gpio128/value
[root@100ask:~]# rs485_test  -d  /dev/ttymxc2 -b 115200
```

第9章 CAN 功能测试

此节主要演示使用两块开发板通过 ip 和 can-utils 命令测试 can0 的通信。

注意：CAN 通信时需要 2 个 CAN 设备，使用“3.5 15EDG 5P”插拔式接线端子进行连接测试。

以下测试是通过两块开发板进行测试，板子背面标有 CAN 引脚丝印“H L”，H 接 H、L 接 L。发货清单中不含接线端子，请自行购买连接测试，或者使用我们的 RS485 转 CAN 模块来测试学习。

模块地址：

http://download.100ask.org/modules/CommunicationModule/RS485-CAN/

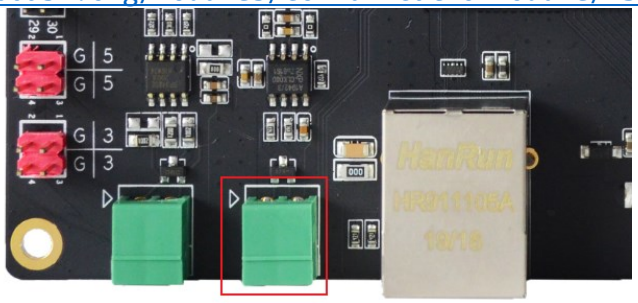


图 9.1

① 发送端：

■ 关闭 can0 接口：

```
[root@100ask:~]# ip link set can0 down
```

■ 设置 can0 传输速率为 50Kbits/sec：

```
[root@100ask:~]# ip link set can0 type can bitrate 50000
```

■ 打印 can0 的信息：

```
[root@100ask:~]# ip -details link show can0
```

■ 打开 can0 接口：

```
[root@100ask:~]# ip link set can0 up
```

■ 使用 cansend 命令向另一端发送数据：

```
[root@100ask:~]# cansend can0 123#DEADBEEF
[root@100ask:~]# cansend can0 123#DEADBEEF
[root@100ask:~]# cansend can0 123#DEADBE23
```

2)接收端

建议等接收端设置好并执行 candump 命令之后，发送端再执行 cansend 命令

■ 关闭 can0 接口：

```
[root@100ask:~]# ip link set can0 down
```

■ 设置 can0 传输速率为 50Kbits/sec：

```
[root@100ask:~]# ip link set can0 type can bitrate 50000
```

■ 打印 can0 的信息：

```
[root@100ask:~]# ip -details link show can0
```

■ 打开 can0 接口：

```
[root@100ask:~]# ip link set can0 up
```

■ 打印 can0 的接收到的数据：

```
[root@100ask:~]# candump can0
```

第10章 Key(按键)测试

此节演示通过 hexdump 命令以及 dmesg 命令来查看按键是否有反应。

执行下面命令之后，操作按键，如果一切正常会有打印信息：

```
[root@100ask:~]# hexdump  /dev/input/event0
```

按下按键的 log：

```
0000000 d39a 5f37 70f2 0000 0001 001e 0001 0000
0000010 d39a 5f37 70f2 0000 0000 0000 0000 0000
0000020 d39a 5f37 b963 0000 0001 001e 0000 0000
0000030 d39a 5f37 b963 0000 0000 0000 0000 0000
```

退出测试按 CTRL+C。

第11章 查看 CPU 温度

查看 CPU 温度，可以使用以下命令，这里表示当前 CPU 温度是 52°C。可以用手触摸 CPU 芯片几秒钟后离开，再 cat 查看，可以看到温度短暂下降后又上升。

在开发板终端下执行

```
[root@100ask:~]# cat /sys/class/hwmon/hwmon0/temp1_input
```

会得到一个数值 log：

```
58661
```

隔几秒钟后再次执行得到一个新的温度数值 log：

```
58892
```