WiFi

使用指南

文档版本　09

发布日期　2019-02-14

商标声明

 、HISILICON、海思和其他海思商标均为海思技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

上海海思技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://www.hisilicon.com/cn/

客户服务邮箱： support@hisilicon.com

前言

概述

本文档主要是介绍WiFi需要使用到的配置，基本操作、调试方法，使用注意事项和常见问题处理。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3798C	V1XX
Hi3798C	V2XX
Hi3796C	V1XX
Hi3798M	V1XX
Hi3796M	V1XX
Hi3798M	V2XX(H)
Hi3796M	V2XX
Hi3798M	V3XX(H)
Hi3716M	V43X

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

作者信息

章节号	章节名称	作者信息
全文	全文	W00212401/L66197

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2019-02-14	09	删除3.2章节起始段落的一句话。
2018-10-10	08	新增支持Hi3716MV430、Hi3798MV310、Hi3798MV300H、Hi3798MV200H芯片。
2018-06-06	07	更新1.3.2章节。
2017-12-30	06	更新WiFi相关内核配置截图。
2017-08-31	05	新增支持Hi3798MV300芯片。
2017-03-07	04	新增支持Hi3798MV200芯片。新增6.2.15、6.2.16章节。
2016-11-04	03	新增支持Hi3798MV200芯片。
2015-04-21	02	新增支持Hi3798CV200芯片。
2014-10-30	01	新增支持Hi3796MV100芯片。
2014-06-18	00B01	第1次临时版本发布。

目 录

1 配置说明

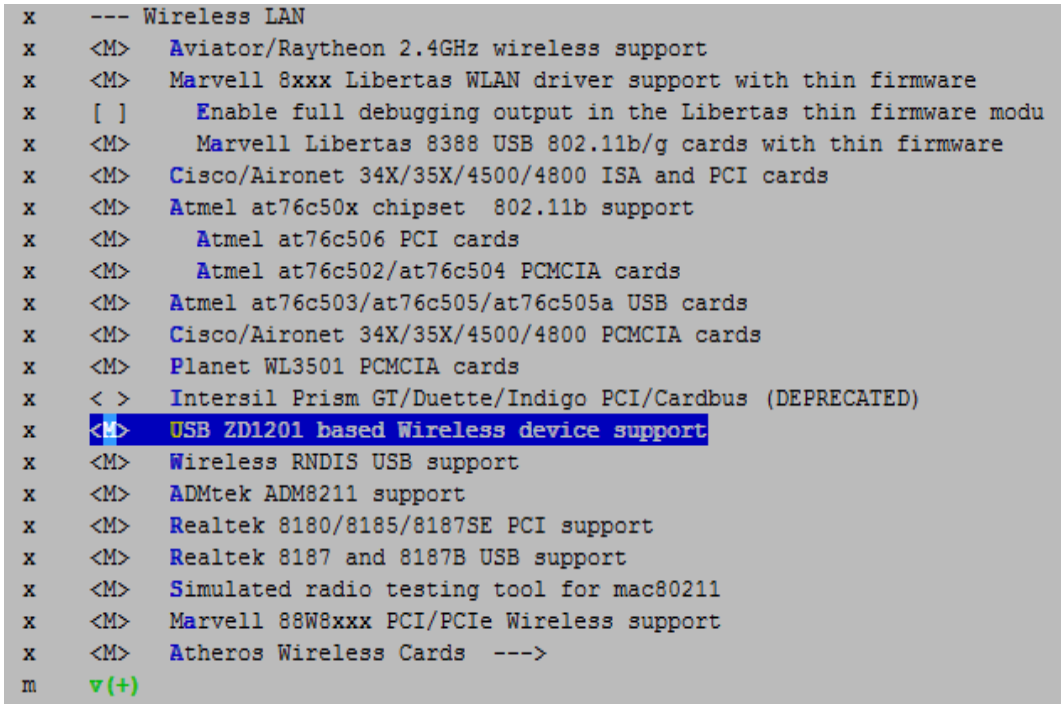
1.1 内核配置

1.1.1 配置 WEXT

WEXT即Wireless Extension，是内核中WiFi驱动和用户态进程的标准接口。

有的内核版本中，WEXT没有直接的配置选项，需要在Device Drivers->Network device support->Wireless LAN中将依赖于CONFIG_WIRELESS_EXT=y的驱动打开，这样CONFIG_WIRELESS_EXT会自动打开。如图1-1所示，将ZD1201设置成M。

图 1-1 WEXT 配置



如图1-1所示，内核中已经附带了多款WiFi的驱动，但实际上并没有使用内核中的驱动程序，而是直接从厂家获取驱动程序，集成到SDK中，因此内核中的相同型号WiFi的驱动程序不要打开，否则会出现编译冲突。

1.1.2 配置 CFG80211

CFG80211是内核中WiFi驱动和用户态进程的标准接口，在CFG80211出现之前是WEXT，现在越来越多的使用CFG80211，WiFi Direct功能只有CFG80211才支持。

进入Network support->Wireless，设置cfg80211和mac80211为y，如图1-2所示。

图 1-2 CFG80211 配置

```
x    --- Wireless
x    <M>   cfg80211 - wireless configuration API
x    [ ]     nl80211 testmode command
x    [ ]     enable developer warnings
x    [ ]     cfg80211 regulatory debugging
x    [ ]     cfg80211 certification onus
x    [*]     enable powersave by default
x    [*]     cfg80211 DebugFS entries
x    [ ]     use statically compiled regulatory rules database
x    [ ]   lib80211 debugging messages
x    <M>   Generic IEEE 802.11 Networking Stack (mac80211)
x    [*]   Minstrel
x    [*]     Minstrel 802.11n support
x    [ ]       Minstrel 802.11ac support (NEW)
x          Default rate control algorithm (Minstrel)  --->
x    [*]   Enable mac80211 mesh networking (pre-802.11s) support
x    -*-   Enable LED triggers
x    -*-   Export mac80211 internals in DebugFS
x    [*]   Trace all mac80211 debug messages
x    [ ]   Select mac80211 debugging features  ----
x
m
```

注意

Atheros的WiFi驱动中带有他们修改过的CFG80211程序，只能使用驱动中的CFG80211，如果内核中的CFG80211设置为y会引起WiFi驱动中的CFG80211程序编译冲突，因此内核中的cfg80211和mac80211要设置成M。

1.1.3 配置 USB

请参考《外设 使用指南》中的USB操作指南，另外需要开启对WiFi的支持，进入Device Drivers->USB support，开启USB Wireless Device Management support。

图 1-3 USB 配置

```
       *** USB Device Class drivers ***
< >    USB Modem (CDC ACM) support
< >    USB Printer support
<*>    USB Wireless Device Management support
< >    USB Test and Measurement Class support
```

1.1.4 配置 Netlink

说明

wpa_supplicant、hostapd模块和内核通信采用了Netlink，所以需要配置Netlink。

进入Network support->Networking options，设置Network packet filtering framework (Netfilter)为y。再进入Network packet filtering framework (Netfilter)，设置Advanced netfilter Configuration为y，再进入Core Netfilter Configuration，按下图进行设置。

图 1-4 Netlink 配置

```
x     [*] Netfilter ingress support (NEW)
x     {M} Netfilter NFACCT over NFNETLINK interface
x     {M} Netfilter NFQUEUE over NFNETLINK interface
x     {M} Netfilter LOG over NFNETLINK interface
x     <M> Netfilter connection tracking support
x     -*- Connection mark tracking support
x     [*] Connection tracking security mark support
x     [*] Connection tracking zones
x     [ ] Supply CT list in procfs (OBSOLETE)
x     [*] Connection tracking events
x     [*] Connection tracking timeout
x     [*] Connection tracking timestamping
x     <M> DCCP protocol connection tracking support
x     <M> SCTP protocol connection tracking support
x     <M> UDP-Lite protocol connection tracking support
x     <M> Amanda backup protocol support
x     <M> FTP protocol support
x     <M> H.323 protocol support
x     <M> IRC protocol support
x     <M> NetBIOS name service protocol support
x     <M> SNMP service protocol support
m     v(+)
```

上述的内核配置在平台中都已经默认设置好，不需要再次进行配置。

1.1.5 配置 NAT 转发

如果需要SoftAP的网络共享功能，进入Network support->Networking options-> Network packet filtering framework (Netfilter)->Core Netfilter Configuration，设置Netfilter connection tracking support为y。再进入Network support->Networking options-> Network packet filtering framework (Netfilter)->IP: Netfilter Configuration，按下图进行配置：

图 1-5 NAT 配置

```
<*> IPv4 connection tracking support (required for NAT)
[*]   proc/sysctl compatibility with old connection tracking (NEW)
<*> IP tables support (required for filtering/masq/NAT)
< >   "ah" match support
< >   "ecn" match support
< >   "ttl" match support
< >   Packet filtering
< >   ULOG target support
<*>   IPv4 NAT
<*>     MASQUERADE target support
<*>     NETMAP target support
<*>     REDIRECT target support
< >   Packet mangling
< >   raw table support (required for NOTRACK/TRACE)
<*> ARP tables support
< >   ARP packet filtering
< >   ARP payload mangling
```

Android平台默认需要网络共享功能，因此SDK中已经配好，Linux平台默认没有配置。

如果需要支持IPv6，请参考IPv4来配置Network support->Networking options-> Network packet filtering framework (Netfilter)->IPv6: Netfilter Configuration。

1.2 Bootargs 配置

1.2.1 配置 ATOMIC 内存大小

RT3070、RT5370、RT5372、RT5572、MT7601U驱动使用了较大的ATOMIC原子内存，kernel中默认的大小是256KB，不够使用，在加载驱动时可能申请不到内存而失败，驱动打印“Failed to allocate memory”。使用这几款WiFi，需要设置ATOMIC内存为1MB字节。MT7632U需要2MB ATOMIC内存，请在bootargs中设置“coherent_pool=2M”。

操作过程如下：

步骤1 编辑bootargs输入文件，在原有bootargs的配置中增加“coherent_pool=1M”。如下所示：

```
bootargs=mem=1G console=ttyAMA0,115200 root=/dev/mtdblock3
rootfstype=yaffs2 mtdparts=hi_sfc:512K(boot),64K(bootargs);hinand:
6M(kernel),96M(rootfs),20M(test),-(other) coherent_pool=1M
```

设置内容与前面的内容中间要有一个空格。

步骤2 执行mkbootargs生成bootargs文件，然后烧入单板。

----结束

1.3 编译配置

1.3.1 Linux 平台编译配置

操作过程如下：

步骤1 在SDK根目录下执行：make menuconfig，进入component，开启“WiFi Support”。

步骤2 进入“WiFi Support”->“WiFi Device Type”， 配置采用的WiFi芯片类型；

注意

这里列出所有支持的wifi芯片。可支持同时打开多个WiFi，但不建议打开不使用的WiFi，打开越多会引起编译变慢和文件系统过大的问题。

步骤3 进入“WiFi Support”->“WiFi Working Mode”， 配置支持的WiFi模式。WiFi模式包括STA和SoftAP模式，可多选。

步骤4 配置文件系统大小。

如果选择的WiFi较多，文件系统可能超过默认的96MB，导致生成文件系统时失败，这时需要将文件系统大小设置成更大的值，如果生成文件系统时没有报错，这一步可以跳过。修改方法：进入“Rootfs”->“File System Config”->“eMMC Rootfs Size”，将96改成更大的数值。

----结束

1.3.2 Android 平台编译配置

说明

Android平台由于需要编译一些相关的用户态程序，因此WiFi配置没有放在SDK中，而是在Android平台通用的BoardConfig.mk中。

BoardConfig.mk中列出所有Android平台支持的WiFi芯片，每一款WiFi都以“BOARD_WLAN_DEVICE_芯片名称”来配置，如果是WiFi+BT的combo芯片，则以“BOARD_WIFI_BLUETOOTH_DEVICE_芯片名称”来配置，设置为“y”表示编译后的镜像支持该款WiFi，设置为“n”或其它值，表示编译后的镜像不支持该款WiFi。

例如：Hi3798CV200平台支持RTL8822BU，修改device/hisilicon/Hi3798CV200/BoardConfig.mk：

```
BOARD_WIFI_BLUETOOTH_DEVICE_RTL8822BU := y
```

2 WiFi 基本操作

2.1 STA 模式基本操作示例

2.1.1 检查 WiFi 设备

开启WiFi前，检查一下WiFi设备是否正常。

执行shell命令：

```
lsusb
```

或者

```
busybox lsusb
```

会打印出USB设备ID：

```
Bus 001 Device 004: ID 0bda:8179
Bus 001 Device 001: ID 1d6b:0002
Bus 002 Device 001: ID 1d6b:0001
```

0bda:8179即RTL8188EUS的USB设备ID，看到了ID说明设备已经识别到。

如果不支持lsusb和busybox lsusb命令，那么可以采用以下方法检查：进入/sys/bus/usb/devices目录，可以看到有多个子目录，每个子目录下都保存着一个USB设备的信息，其中有一个uevent文件，里面保存着设备类型、设备ID等信息。依次查看每个子目录下的uevent文件中的PRODUCT=xx/xx/xx，看是否有采用的WiFi的设备ID，如果找到，那说明已经识别到WiFi设备，如果没有则WiFi未插入、未上电或者已经损坏。

图 2-1 RTL8188EUS USB 设备 ID

2.1.2 加载驱动

步骤1 加载CFG80211驱动。

如果驱动使用了cfg80211，cfg80211是编译成ko文件，需要先加载cfg80211.ko。

进入ko存放目录，执行shell命令：

```
insmod cfg80211.ko
```

步骤2 加载WiFi驱动。

进入ko存放目录，执行shell命令：

```
insmod rtl8188eu.ko
```

一般来说，WiFi芯片不同，驱动也不同，也有不同的WiFi芯片采用了相同的驱动，如RTL8188ETV和RTL8188EUS的驱动是相同的，RTL8188CUS和RTL8192CU的驱动是相同的。

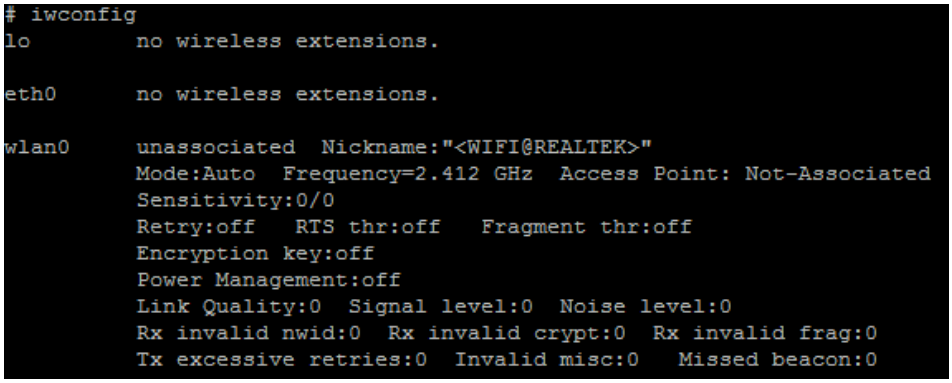
步骤3 查看驱动是否加载成功。

执行shell命令：

```
iwconfig
```

如果看到有一个wlan0网口，那说明驱动已经初始化成功，WiFi设备可用。

图 2-2 iwconfig 执行结果



步骤4 启动WiFi网口。

执行shell命令：

```
ifconfig wlan0 up
```

执行完后，WiFi是可用状态，可以进行扫描和连接操作了。

----结束

2.1.3 扫描 AP

执行shell命令：

```
iwlist wlan0 scan
```

图 2-3 扫描 AP 执行结果

```
# iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: F4:EC:38:22:30:60
                    ESSID:"HiMMI"
                    Protocol:IEEE 802.11bg
                    Mode:Master
                    Frequency:2.412 GHz (Channel 1)
                    Encryption key:on
                    Bit Rates:54 Mb/s
                    Extra:wpa_ie=dd160050f20101000050f20401000050f20401000050f202
                    IE: WPA Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    Extra:rsn_ie=30140100000fac040100000fac040100000fac020100
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                       Preauthentication Supported
                    Quality=0/100  Signal level=42/100
```

扫描到的AP会以“Cell xx”的形式显示，一个AP对应一个“Cell xx”。

每个AP的信息包括：

- Address：MAC地址。
- ESSID：AP的名称，即SSID。
- Protocol：IEEE80211协议，11b/g/n。
- Frequency：信道。
- 认证加密信息：WEP、WPA-PSK、WPA2-PSK、WPA、WPA2。
- Quality：信号质量，该数据有些WiFi显示得不准确，可以忽略。
- Singal Level：信号强度，数字越大，信号强度越高，WiFi芯片不同，显示的方式有些区别，有的是以xx/100类型显示，有的是以xx dBm显示。

上述信息并不是所有WiFi都是以这种格式显示，WiFi不同显示的格式也不一样。

注意

使用iwlist进行扫描时，iwlist不会等驱动扫描完所有信道才返回扫描结果，所以经常会出现有些AP没有搜出来的情况，尤其是MT7601U，由于在每个信道上停留的时间较长，所以第一次扫描时，只能搜到1~2个信道里的AP。

2.1.4 连接 AP

连接AP是通过wpa_supplicant进程进行的。wpa_supplicant是开源代码，Linux、Android都是采用它负责WiFi的连接过程，它包含了WEP、WPA/WPA2、WPA-SPK/WPA2-PSK、WAPI、WPS、P2P、EAP等协议。

步骤1 启动wpa_supplicant进程。

执行shell命令：

```
wpa_supplicant -iwlan0 -Dnl80211 -c/etc/Wireless/wpa_supplicant.conf&
```

- -iwlan0表示使用wlan0网口；
- -Dnl80211表示使用cfg80211接口（用户态的接口是libnl，内核中是cfg80211），另外一个可选的是-iwext，表示使用wext接口；
- -c/xxx/wpa_supplicant.conf是wpa_supplicant的配置文件，要保证该文件已经存在。

执行完后，用ps命令查看一下wpa_supplicant进程是否存在，存在表示工作正常。如果没有wpa_supplicant进程，可以增加wpa_supplicant的打印级别，从log看出现什么问题，如：

```
wpa_supplicant -iwlan0 -Dnl80211 -c/etc/Wireless/wpa_supplicant.conf -ddd &
```

步骤2 启动wpa_cli进程。

执行shell命令：

```
wpa_cli -iwlan0
```

执行成功会出现“>”符号。

如果出现“Could not connect to wpa_supplicant - re-trying”，那表示wpa_cli不能和wpa_supplicant建立socket连接，这时要检查wpa_supplicant进程是否还在，再看是否有/var/run/wpa_supplicant/wlan0，然后检查wpa_supplicant.conf文件中是否是ctrl_interface=/var/run/wpa_supplicant。

步骤3 扫描。

在“>”后执行“scan”命令，收到“CTRL-EVENT-SCAN-RESULTS”后再执行“scan_results”，会获得扫描结果。

图 2-4 wpa_cli 扫描 AP 结果

```
> scan
OK
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE

> > scan_results
bssid / frequency / signal level / flags / ssid
78:a1:06:48:e2:e8       2472    -65     [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] B21-1
40:4d:8e:81:08:f1       2462    -69     [WPA-PSK-TKIP][ESS]     B25_chenxie
f4:ec:38:22:30:60       2412    -74     [WPA-PSK-CCMP][WPA2-PSK-CCMP-preauth][ESS]      HiMMI
8c:21:0a:a5:cd:b2       2437    -48     [WEP][ESS]      B21
```

步骤4 连接。

1. 以连接OPEN方式的AP为例，在“>”后执行“add_network”，假如返回网络ID为0。
2. 配置网络的SSID，执行set_network 0 ssid AP的SSID。
3. 配置网络的加密方式，执行“set_network 0 key_mgmt NONE”。
4. 启动网络，执行“enable_network 0”。
5. 收到“CTRL-EVENT-CONNECTED”表示连接成功。

图 2-5 连接 AP

```
> add_network
0
> set_network 0 ssid "WINDSKY_WLAN"
OK
> set_network 0 key_mgmt NONE
OK
> enable_network 0
OK
> wlan0: Trying to associate with ac:f7:f3:e5:d7:33 (SSID='WINDSKY_WLAN' freq=2437 MHz)
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>Trying to associate with ac:f7:f3:e5:d7:33 (SSID='WINDSKY_WLAN' freq=2437 MHz)
wlan0: Associated with ac:f7:f3:e5:d7:33
<3>Associated with ac:f7:f3:e5:d7
wlan0: CTRL-EVENT-CONNECTED - Connection to ac:f7:f3:e5:d7:33 completed (auth) [id=0 id_st
<3>CTRL-EVENT-CONNECTED - Connection to ac:f7:f3:e5:d7:33 completed (auth) [id=0 id_str=]
```

步骤5 获取IP地址。

输入q退出wpa_cli，执行shell命令：udhcpc －i wlan0

获取了IP地址后，可以ping网关看是否能ping通。

----结束

2.2 SoftAP 模式基本操作示例

RT3070、RT5370、RT5372、RT5572、MT7601U操作SoftAP是通过iwpriv命令进行的。其它的WiFi是通过hostapd进程进行的。hostapd和wpa_supplicant类似，它包含了AP端的各种认证协议、连接流程，wpa_supplicant是STA端的。

2.2.1 检查 WiFi 设备、加载驱动

检查WiFi设备、加载驱动的操作和STA模式一样的。

2.2.2 iwpriv 配置和启动 SoftAP

步骤1 开启SoftAP。

执行shell命令：

```
ifconfig wlan0 up
```

驱动会读取/etc/Wireless/RT2870AP/RT2870AP.dat文件中的参数，初始化并开启SoftAP。

步骤2 配置信道。

执行shell命令：

```
iwpriv wlan0 set Channel=6
```

信道号为1~11。

步骤3 配置加密方式。

- OPEN
  ```
  iwpriv wlan0 set AuthMode=OPEN
  iwpriv wlan0 set EncrypType=NONE
  ```
- WEP
  ```
  iwpriv wlan0 set AuthMode=WEPAUTO
  iwpriv wlan0 set EncrypType=WEP
  iwpriv wlan0 set DefaultKeyID=1
  iwpriv wlan0 set Key1=xxxxx
  ```
- WPA2-PSK
  ```
  iwpriv wlan0 set AuthMode=WPA2PSK
  iwpriv wlan0 set EncrypType=AES
  iwpriv wlan0 set WPAPSK=xxxxxxxx
  ```

步骤4 配置SSID。

执行shell命令：

```
iwpriv wlan0 set SSID=XXX
```

----结束

2.2.3 hostapd 配置和启动 SoftAP

步骤1 修改hostapd.conf文件。

hostapd进程需要使用hostapd.conf配置文件，在配置文件里设置SSID、信道、加密方式等。配置文件的内容举例如下：

- OPEN
  ```
  interface=wlan0
  driver=nl80211
  ctrl_interface=/var/run/hostapd
  ssid=HisiAP
  channel=6
  hw_mode=g
  ieee80211n=1
  ht_capab=[SHORT-GI-20][SHORT-GI-40][HT40-]
  ```
- WEP
  ```
  interface=wlan0
  driver=nl80211
  ctrl_interface=/var/run/hostapd
  ssid=HisiAP
  channel=6
  hw_mode=g
  wep_default_key=0
  wep_key0="12345"
  ```
- WPA2-PSK
  ```
  interface=wlan0
  driver=nl80211
  ctrl_interface=/var/run/hostapd
  ssid=HisiAP
  ```

```
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[SHORT-GI-20][SHORT-GI-40][HT40-]
wpa=3
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
wpa_passphrase=12345678
```

hostapd是开源代码，配置文件中的参数可以参考网络资源。

步骤2 启动hostapd进程。

执行Shell命令：

```
hostapd /etc/Wireless/hostapd.conf &
```

执行完后，用ps命令查看一下hostapd进程是否存在，存在表示工作正常，用STA设备可以搜索到SoftAP。如果没有，可以增加hostapd的打印级别，从log看出现什么问题，如：

```
hostapd /etc/Wireless/hostapd.conf -ddd &
```

----结束

2.2.4 开启 udhcpd

执行Shell命令：

```
ifconfig wlan0 192.168.1.1
udhcpd -fS /etc/udhcpd.conf
```

请确保/etc/udhcpd.conf文件存在，并且配置的网段为192.168.1.x。执行完后，用STA设备可以连接该SoftAP。

2.2.5 网络共享

网络共享的应用场景为单板上行通过以太网连接外网，下行通过SoftAP将网络分享给STA设备，让连接上SoftAP的STA设备也可以访问外网。

执行Shell命令：

```
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j ASQUERADE
```

2.3 配置适用的国家区域

不同的国家或区域，采用的频率范围有些不同，比如2.4GHz频段，美国支持1~11信道，中国和欧洲支持1~13信道，日本支持1~14信道。5GHz频段也类似。WiFi需要根据产品上市的国家或区域做相应的配置，以适用于改国家的频率范围。

不同的WiFi配置方法不一样。

- RTL8188EUS配置成美国

  在加载驱动时带上rtw_channel_plan=0x22参数：

  ```
  insmod rtl8188eu.ko rtw_channel_plan=0x22
  ```

- MT7601U配置成美国

  修改驱动配置文件，如/etc/Wireless/RT2870STA/RT2870STA.dat，设置参数为：

```
CountryRegion=0
CountryCode=US
```

本文档不详细列出所有配置方法，有需要可以咨询WiFi芯片厂家。

3 测试

3.1 功能测试

3.1.1 STA 模式

包含但不局限于以下测试：

- 修改AP的SSID，单板去扫描、连接AP；
- 修改AP的信道，单板去扫描、连接AP；
- 修改AP的无线协议，单板去扫描、连接AP；
- 修改AP的加密方式，单板去扫描、连接AP；
- 连接不同类型的AP，测试兼容性；
- 连接AP，连续ping 12小时以上；
- 逐渐增加干扰，测试连接的稳定性。

3.1.2 SoftAP 模式

包含但不局限于以下测试：

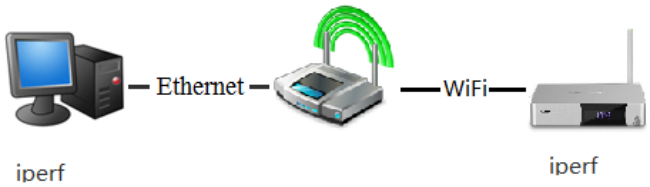
- 修改SSID，用手机搜索并连接；
- 修改信道，用手机搜索并连接；
- 修改加密方式，用手机搜索并连接；
- 采用不同类型的手机连接，测试兼容性；
- 用手机连接，连续ping 12小时以上；
- 逐渐增加干扰，测试连接的稳定性。

3.2 吞吐量测试

吞吐量测试可以反映WiFi的性能，是目前芯片厂家、模组厂家、设备厂家普遍使用的测试方法，具有很高的认同度。吞吐量测试最常使用的工具是iperf。

测试环境为PC机通过有线和AP连接，单板通过WiFi和AP连接，单板和PC机可以互相ping通。在PC机和单板上都有iperf工具。假设PC机的IP地址为192.168.1.100，单板的IP地址为192.168.1.101。

图 3-1 吞吐量测试组网环境



3.2.1 TCP 发送吞吐量测试

发送吞吐量测试操作如下：

步骤1 PC机上命令行进入iperf工具目录，执行：

```
iperf -s
```

步骤2 单板上通过Shell进入iperf工具目录，执行：

```
iperf -c 192.168.1.100 -t 10 -i 1
```

图 3-2 发送吞吐量测试示例

```
# iperf -c 192.168.1.100 -t 10 -i 1
Client connecting to 192.168.1.100, TCP port 5001
TCP window size:  512 KByte (default)
[  3] local 192.168.1.101 port 44753 connected with 192.168.1.100 port 5001
[  3]  0.0- 1.0 sec  8.40 MBytes  70.5 Mbits/sec
[  3]  1.0- 2.0 sec  8.57 MBytes  71.9 Mbits/sec
[  3]  2.0- 3.0 sec  8.65 MBytes  72.5 Mbits/sec
[  3]  3.0- 4.0 sec  8.52 MBytes  71.4 Mbits/sec
[  3]  4.0- 5.0 sec  8.57 MBytes  71.9 Mbits/sec
[  3]  5.0- 6.0 sec  8.52 MBytes  71.4 Mbits/sec
[  3]  6.0- 7.0 sec  8.59 MBytes  72.1 Mbits/sec
[  3]  7.0- 8.0 sec  8.52 MBytes  71.5 Mbits/sec
[  3]  8.0- 9.0 sec  8.72 MBytes  73.1 Mbits/sec
[  3]  9.0-10.0 sec  8.62 MBytes  72.4 Mbits/sec
[  3]  0.0-10.0 sec  85.7 MBytes  71.6 Mbits/sec
```

其中，iperf －s表示启动服务端，iperf －c 192.168.1.100表示启动客户端，连接192.168.1.100，-t 10表示测试10秒钟，-i 1表示每隔1秒钟打印一次结果。

最后打印的“0.0-10.0 sec 85.7 MBytes 71.6 Mbits/sec”表示这10秒钟的平均吞吐量为71.6Mbps。

----结束

3.2.2 TCP 接收吞吐量测试

接收吞吐量测试操作如下：

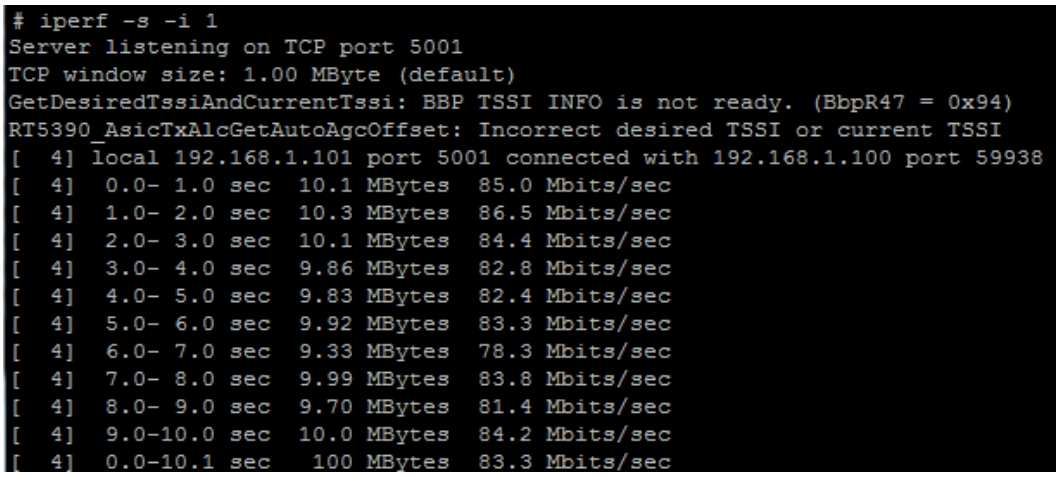
步骤1 单板上通过Shell进入iperf工具目录，执行：

```
iperf -s
```

步骤2 PC机上命令行进入iperf工具目录，执行：

```
iperf -c 192.168.1.101 -t 10 -i 1 -w 1M
```

图 3-3 接收吞吐量测试示例



iperf也可以进行UDP测试，在有些PC机上单个UDP线程进行了限速，因此需要开启多个线程。

SoftAP的吞吐量测试类似。

注意

有些PC机，由于安装了一些软件，对速率会有影响，一定要确保PC机没有速率的瓶颈。WEP安全模式不能使用802.11n协议，因此速率比较低，一般只有20+Mbps。

----结束

3.2.3 UDP 发送吞吐量测试

发送吞吐量测试操作如下：

步骤1 PC机上命令行进入iperf工具目录，执行：

```
iperf -s -u -l 32k
```

步骤2 单板上通过Shell进入iperf工具目录，执行：

```
iperf -c 192.168.1.100 -u -t 10 -i 1 -l 32k -b 100M
```

----结束

3.2.4 UDP 接收吞吐量测试

接收吞吐量测试操作如下：

步骤1 单板上通过Shell进入iperf工具目录，执行：

```
iperf -s -u
```

步骤2 PC机上命令行进入iperf工具目录，执行：

```
iperf -c 192.168.1.101 -u -t 10 -i 1 -l 32k -b 100M
```

----结束

3.3 射频指标测试

吞吐量测试可以反映WiFi的性能，在产品开发中必须要做的工作。有条件的公司还可以进行射频指标测试，它可以准确的验证WiFi模组射频是否达标。因为模组厂家在生产模组时是必须做的工作，所以如果采用的是模组，这项工作是可选的。但由于硬件设计时，有可能地线不干净、板上干扰等原因，影响WiFi射频性能，因此建议有条件的公司要进行该项测试。

射频指标包括：接收灵敏度、领道功率抑制、发送功率、发送载频容差、丢包率、EVM、接收杂散、发送杂散等。

测试仪器包含：频谱分析仪、功率测量仪、网络分析仪等。

测试方法较复杂，可以参考测量仪器说明书。

3.4 天线测试

天线是影响WiFi性能的另一大原因，天线指标测试是产品开发过程中必不可少的。天线性能测试必须使用整机，可以使用天线厂家的测试环境。

天线指标包括但不限于以下几条：

- 效率：指天线辐射出去的功率（即有效的转换电磁波部分的功率）和输入到天线的有功功率之比，为天线的主要指标
- 增益：在输入功率相等的条件下，实际天线与理想的辐射单元在空间同一点处所产生的信号功率密度比。天线的另一项关键指标，结合方向性一起综合判断天线好坏。
- 驻波比：反映天线的匹配情况，确保信号能量进入天线。是天线效率的重要保证指标，同种指标描述还有回波损耗、反射系数、输入阻抗等。

4 硬件设计注意事项

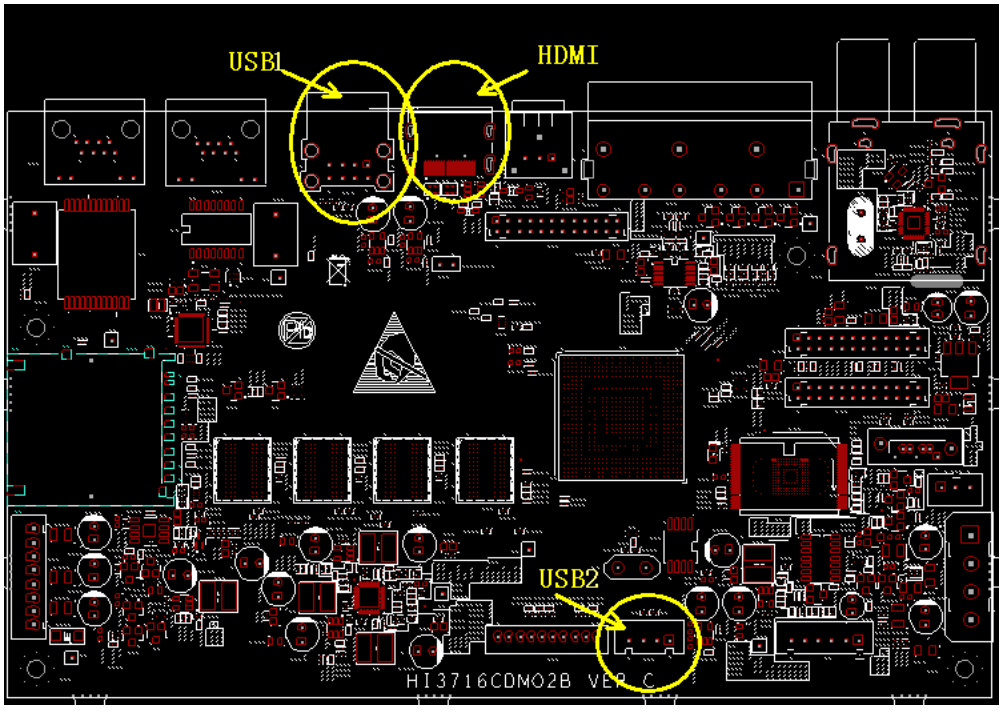
4.1 HDMI 接口干扰 WiFi 信号

HDMI使用74.2MHz频率时，其33倍频正好处于WiFi的2.4G频段内，会严重的干扰WiFi信号。如果使用148.5MHz频率，虽然其16倍频没有处于WiFi频段内，但是由于频率的隔离度也不好，也会在一定程度上干扰WiFi的信号。

由于HDMI连接器焊接处也无法屏蔽，如果PCB板上HDMI接口和WiFi模块距离小于5cm，HDMI的输出显示都会干扰WiFi信号，导致WiFi无法连接、吞吐量下降等问题。

因此，在硬件布局上要将WiFi模块的位置远离HDMI端口，避免干扰。如图4-1所示，WiFi模块建议不要使用USB1，建议使用USB2。

图 4-1 HDMI 与 WiFi 硬件设计示例



4.2 系统时钟对 WiFi 的干扰

如果系统时钟工作在600MHz左右，其倍频有可能会处于2.4GHz～2.5GHz范围内，从而干扰WiFi的信号，导致WiFi无法连接、吞吐量下降等问题。

解决方法如下：

- 定位干扰是否从PCB通过电源或者GND影响WiFi模块。用频谱分析仪，将接触探头（不能接收空间辐射信号）点在wifi模块的电源和GND附近，对比附近非倍频的区域的噪声强度，如果相差在5dB以内，则干扰不大，反则可视为干扰。如果是这种干扰，那么在电源和GND上串联磁珠隔离噪声干扰。
- 定位干扰是否从空间辐射影响WiFi模块和天线。用频谱分析仪连接近场探头（用于探测近场辐射），扫描WiFi模块附近，如果倍频上有噪声，则基本可认为噪声存在。如果是WiFi模块收到辐射干扰影响，建议WiFi模块加屏蔽。如果是WiFi天线受到辐射干扰，可配置WiFi天线到盒子内部干扰较少的位置，如果产品外壳较小，无法满足，可以使用外置天线。
- 可以调整系统时钟，如把600MHz调整到590MHz，避开WiFi频段达到减少干扰的目的。

5 软件设计注意事项

5.1 搜索发现

5.1.1 对搜索发现的影响

对比有线网络，WiFi网络的报文丢失的概率更高，其丢包率根据WiFi的信号强度、WiFi模组的性能、环境干扰情况的不同而不同。在邻频（相邻信道）有大量数据业务的情况下，干扰最为严重，丢包率非常高。在设备发现时，会出现组播报文丢失而无法发现的问题。

5.1.2 应对策略

- 增强对WiFi性能的分析和提示；
- 减少发送组播报文的间隔，同时增长超时的时间；
- 上下线的通知消息尽量以TCP短链接方式进行，并考虑TCP握手包丢包的情况，超时时间要增长到20～30秒；
- 尽量缩小WiFi的使用范围。在进行搜索发现时，考虑到IP层丢包对于业务的影响，所以在实现软件方案时，可以考虑优先使用有线网络。

5.2 UDP 业务

5.2.1 UDP 的主要业务

UDP协议主要应用于实时流的业务，如mirror、mircast、Video Phone。这些业务对数据的实时性有很高的要求，延时过大会影响业务体验，而对于延时和缓冲并不敏感的媒体播放，则不适合采用UDP进行数据传输。

5.2.2 对 UDP 业务的影响

WiFi网络下由于干扰易出现丢包，并且丢包不可预测，UDP协议无重传机制，数据包丢失后在发送端不会重发，另外在接收端接收到的UDP报文乱序情况也比有线网络严重，这两个因素会影响UDP业务的用户体验。

以Miracast业务为例，根据统计数据，丢包的情况总结如下：

- 网卡的丢包率要比应用层RTP包统计的丢包率要多，可以表明在进行Miracast业务时，不仅有承载RTP包的UDP报文会被丢弃，还会有其他报文丢包，如P2P连接保活等；
- 网卡统计的丢包率与应用层RTP的丢包率并不成线性关系，由此得出的结论是不能用网卡的丢包率来表示RTP包的丢包率，即不能用网卡的丢包率来表示Miracast的性能；
- 在统计RTP包丢包率在低于1%时，效果最好，在介于1%～2%之间显示效果是可以接受的。丢包率超过6%，表现就会比较差。

5.2.3 应对策略

对于丢包的处理策略，归纳如下：

- 在帧数据的处理上，“卡屏”的效果会优于“花屏”。这一策略可以这么解释，首先我们规定一个数据帧可以完整显示一个图像，一个数据帧通过UDP传送到对端可以是很多个UDP报文。当一个数据帧出现了大量丢包时，完全的丢弃这个数据帧要优于将这个数据帧显示出来，通过丢弃数据帧而降低帧率的处理方式是优于数据报文被解码而产生大量错包或者马赛克的体验的；
- 对丢包率的统计几乎是没有意义的，因为WiFi在受到干扰时，有可能报文没有发送出去或者没有接收到，所以出现丢包时，尽量保证最后一个媒体的数据帧能够显示出来，是最可靠的方法；
- 降低帧率、图像质量、码率等降低视频清晰度的方法来减少IP报文的负载，这种方式在一定程度上是减少WiFi的丢包。

总之，在使用UDP进行数据传输时，一定要充分考虑到数据报文丢失后的策略，无论是重传，还是降低图像质量，根据不同的软件应用场景，都需要进行综合的考虑。

5.3 TCP 业务

5.3.1 TCP 的主要业务

TCP协议主要应用于多媒体播放、网络浏览等业务，其主要的应用层协议是HTTP，业务包括NFS、FTP、SAMBA、DLNA等。

5.3.2 对 TCP 业务的影响

由于TCP有重传机制，报文丢失后会继续重发。在WiFi受到干扰时，丢包率较大时会出现TCP握手时间增长、数据发送或者接收时间增长，另外还有可能出现TCP单通的情况（只能进行读或者写）。根据Mirror和Miracast的验证情况，TCP报文可能出现长达5秒的单通，在干扰严重时，可能出现20～30秒的超时时间。长时间的超时会导致应用层处理策略改变，无法区分网络是否已经不通了。

总结在WIFI情况下TCP业务的特点如下：

- 在WiFi工作环境复杂时，报文握手时间和确认的时间可能会很长；
- 和报文大小关系不大；
- 控制信令容易同步；
- 依靠超时来进行退出时，没办法直接判断对端的网络是否不可用了。

5.3.3 应对策略

- 尽量使用短链接，在使用短链接时，不会出现状态机的不同步问题；
- 超时和软件其他模块的状态需要分开，否则其他模块可能会让用户无法接受；
- TCP报文禁止使用在实时性要求非常高的业务上；
- 尽量使用标准协议和标准的软件和组件；
- 所有以超时方式进行退出的模块，都需要审视设计，因为超时时可能网络并没有真正断开，可以适当延长超时时间。

5.4 WiFi Direct 和 Station 模式比较

说明

WiFi网络有多种形式，WiFi Direct、Station-AP、Ad-hoc。WiFi Direct是连接的双方是对等的，直接可以连接；Station-AP是常用的网络形式，必须有一方是AP；Ad-hoc现在使用得很少。

在业务设计上，数据通路可以采用不同类型的WiFi网络，需要根据业务的功能、性能要求而选择更为合适的网络类型。

WiFi Direct和Station-AP网络的受干扰情况是一样的，所以在干扰环境下，WiFi的稳定性都不是很好。连接时间上，WiFi Direct比Station-AP网络要多花3秒钟左右。Station-AP网络连接断开时会尝试重新连接，如果重连成功，业务会继续，WiFi Direct网络断开后不重连。业务体验要求较高的场景，最好避免使用WiFi，如果非要使用WiFi，那么尽量减少邻频干扰。

5.5 5G 频段和 2.4G 频段比较

2.4G频段有13个信道（中国），信道中心频率间隔为5MHz，5G的频段信道有二十多个，信道中心频率间隔有20MHz，2.4G频段的WiFi干扰比5G频段严重。ZigBee、蓝牙、微波炉等都是工作在2.4G频段上，非WiFi的干扰比5G频段多。所以在干扰的问题上，5G频段要比2.4G频段好，业务可以尽量选择5G频段。在穿墙能力上，2.4G频段优于5G频段，如果考虑穿墙，业务优先选择2.4G频段。

5.6 开启/关闭 WiFi

WiFi开启和关闭都需要较长时间，开启一般需要1~3秒钟，API调用返回时并不表示开启和关闭已经完成，而是直到收到状态变化通知后才表示完成，如Android平台，在调用WifiManager.setWifiEnabled()后等待WifiManager.WIFI_STATE_CHANGED_ACTION广播：

```
    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (WifiManager.WIFI_STATE_CHANGED_ACTION.equals(action)) {
                handleWifiStateChanged(intent.getIntExtra(
                        WifiManager.EXTRA_WIFI_STATE,
WifiManager.WIFI_STATE_UNKNOWN));
            }
        }
    }
    private void handleWifiStateChanged(int state) {
        switch (state) {
```

```
            case WifiManager.WIFI_STATE_ENABLING:
                break;
            case WifiManager.WIFI STATE ENABLED:
                break;
            case WifiManager.WIFI_STATE_DISABLING:
                break;
            case WifiManager.WIFI STATE DISABLED:
                break;
            default:
                break;
        }
```

状态为WIFI_STATE_ENABLED和WIFI_STATE_DISABLED后才表示开启和关闭完成，在这之间不能对WiFi进行操作。

6 常见问题及解决办法

6.1 定位工具

6.1.1 iw tools

包括iwconfig、iwlist、iwpriv，是Linux用来配置无线网络，查看网络状态的工具。

- iwconfig
  - 看WiFi驱动是否初始化成功

```
# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

eth1      no wireless extensions.

wlan0     unassociated  Nickname:"<WIFI@REALTEK>"
          Mode:Auto  Frequency=2.412 GHz  Access Point: Not-
Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/100  Signal level=0 dBm  Noise level=0
dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:
0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:
0
```

  有“wlan0”表示驱动初始化成功，否则失败。

  - 查看无线网络状态

```
# iwconfig wlan0
wlan0     IEEE 802.11bgn  ESSID:"B21"  Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 8C:
21:0A:A5:CD:B2
          Bit Rate:150 Mb/s   Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:****-****-****-****-****-****-****-
****   Security mode:open
          Power Management:off
          Link Quality=88/100  Signal level=-45 dBm  Noise
```

```
level=0 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:
0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:
0
```

其中包含的信息有支持的80211协议、AP的SSID、STA/Ad-hoc/AP模式、AP的频率、AP的MAC地址、当前的速率、加密方式、信号质量。

- iwlist

  扫描AP

```
# iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 -  Address: 8C:21:0A:A5:CD:B2
                     ESSID:"B21"
                     Protocol:IEEE 802.11bgn
                     Mode:Master
                     Frequency:2.437 GHz (Channel 6)
                     Encryption key:on
                     Bit Rates:300 Mb/s

Extra:wpa_ie=dd1a0050f20101000050f20202000050f2040050f20201000050f202
                     IE: WPA Version 1
                         Group Cipher : TKIP
                         Pairwise Ciphers (2) : CCMP TKIP
                         Authentication Suites (1) : PSK

Extra:rsn_ie=30180100000fac020200000fac04000fac020100000fac020000
                     IE: IEEE 802.11i/WPA2 Version 1
                         Group Cipher : TKIP
                         Pairwise Ciphers (2) : CCMP TKIP
                         Authentication Suites (1) : PSK
                     Quality=0/100  Signal level=-47 dBm
```

  扫描结果说明见“2.1.3 扫描AP”。

- iwpriv

  iwpriv是直接从驱动中读取或者配置参数。不同的WiFi支持的参数是不一样的，每个参数代表的意义需要咨询厂家。

  查看WiFi支持的参数：

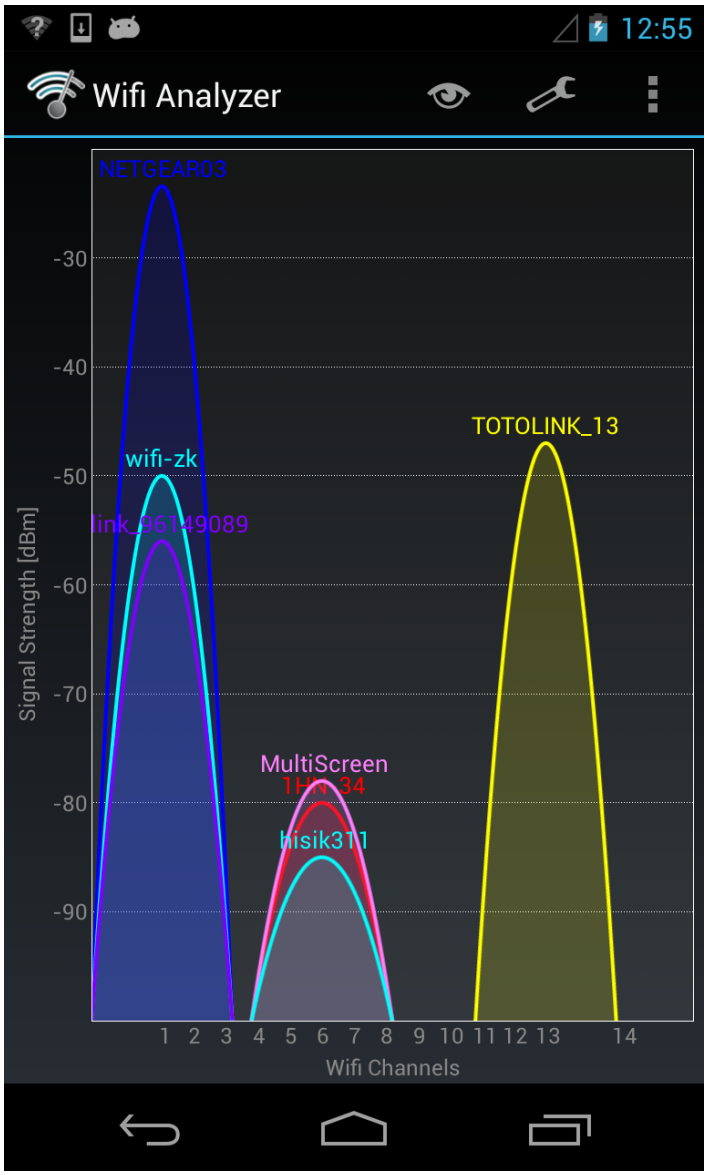
```
# iwpriv wlan0
wlan0     Available private ioctls :
          write            (8BE0) : set 2047 char  & get   0
          read             (8BE1) : set 2047 char  & get  16 char
          driver_ext       (8BE2) : set   0       & get   0
          mp_ioctl         (8BE3) : set   0       & get   0
          apinfo           (8BE4) : set   1 int   & get   0
          setpid           (8BE5) : set   2 int   & get   0
          wps_start        (8BE6) : set   1 int   & get   0
          get_sensitivity  (8BE7) : set   1 int   & get   0
          wps_prob_req_ie  (8BE8) : set   1 int   & get   0
          wps_assoc_req_ie (8BE9) : set   1 int   & get   0
          channel_plan     (8BEA) : set   1 int   & get   0
          dbg              (8BEB) : set   2 int   & get   0
          rfw              (8BEC) : set   3 int   & get   0
          rfr              (8BED) : set   2 int   & get  16 char
          p2p_set          (8BF0) : set  64 char  & get   0
          p2p_get          (8BF1) : set  64 char  & get  64 char
          p2p_get2         (8BF2) : set  64 char  & get  16 char
          NULL             (8BF3) : set 128 char  & get   0
          tdls             (8BF4) : set  64 char  & get   0
```

```
tdls_get          (8BF5) : set   64 char   & get   64 char
pm_set            (8BF6) : set   64 char   & get    0
rereg_nd_name     (8BF8) : set   16 char   & get    0
efuse_set         (8BFA) : set 1024 char   & get    0
efuse_get         (8BFB) : set 128 char   & get 2047 char
```

6.1.2 WiFi 分析仪

WiFi分析仪是一款Android应用，可以统计周围的AP数量、SSID、采用的信道、信号强度。可以用来分析当前的干扰情况，如图6-1所示。

图 6-1 WiFi 分析仪示例



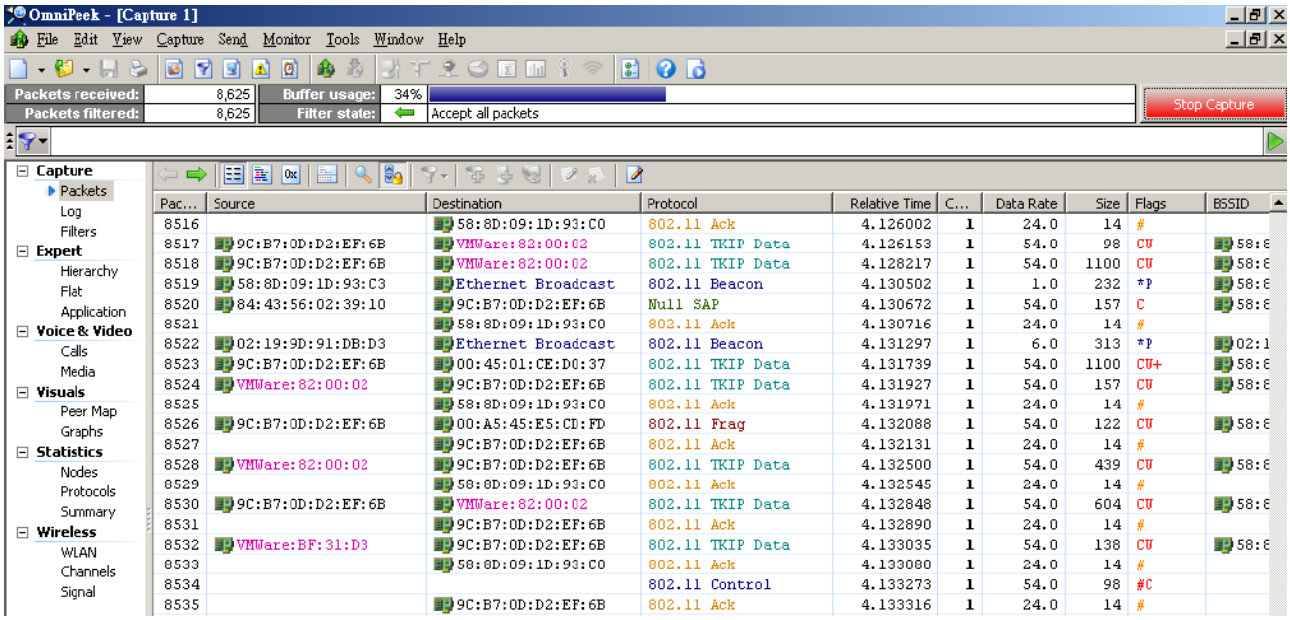
图中表示在信道1上有3个AP，信道6上有三个AP，信道13上有一个AP，AP的峰值越高表示信号越强，跨度表示频率范围，比如信道1的AP对信道2和3也会有干扰。

WiFi分析仪不能知道每个信道中传输的数据量，所以不能体现实际的干扰情况，只能从中按照概率来判断，AP多的信道干扰更大。

6.1.3 OmniPeek

OmniPeek是一款网络抓包工具，搭配DWA160无线网卡使用，可以抓到WiFi的空口包，用来分析WiFi协议非常有用。

图 6-2 OmniPeek 抓包示例



从抓取的数据中可以看到每个80211包的具体内容。

6.1.4 logcat

logcat是Android平台的获取log的工具。WiFi设置、Framework层、HAL、wpa_supplicant、hostapd都会通过logcat打印log，当出现问题的时候可以看到相应的log信息。

6.2 常见问题及解决办法

6.2.1 加载 WiFi 驱动失败

问题描述

加载驱动时，提示文件格式不对，或者魔术字不对。

问题分析

原因是驱动中Makefile的交叉编译环境配置得不对，编译成PC平台的模块，或者Linux内核版本不对。

解决办法

修改驱动的Makefile，设置正确的ARCH、CROSS_COMPILE、linux内核路径。然后再编译驱动。

如RTL8188EUS：

```
ifeq ($(CONFIG_PLATFORM_HISILICON), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_PLATFORM_ANDROID -
DCONFIG_PLATFORM_SHUTTLE
ARCH := arm
ifeq ($(CROSS_COMPILE),)
CROSS_COMPILE = arm-hisiv200-linux-
endif
MODULE_NAME := rtl8188eu
ifeq ($(KSRC),)
KSRC := ../../../../../../kernel/linux-3.4.y
endif
endif
```

6.2.2 WiFi 使用过程中出现 USB disconnect

问题描述

在WiFi使用的过程中，在串口中出现类似“usb 1-2.1: USB disconnect, address 3”打印，然后WiFi就不能再使用了。

问题分析

原因是USB口已经识别不到WiFi设备了，这句log是USB驱动打印的。原因有两个：一是WiFi设备已经损坏；二是USB口供电不足。

解决办法

首先替换WiFi设备，看是否是模组有问题，再检查WiFi使用过程中USB的电压和电流是否达到模组要求。

6.2.3 WiFi 吞吐量低

问题描述

单板和AP的距离在1米以内，测试WiFi的吞吐量比较低。

解决办法

造成吞吐量低的原因有很多，可以按照以下方法排查：

步骤1 打开AP的设置页面，看是否设置成11n模式。

步骤2 打开AP的设置页面，看是否设置成WEP加密方式，WEP加密不支持11n。

步骤3 打开AP的设置页面，看是否设置成WPA-PSK/WPA2-PSK，加密加密算法设置为TKIP。

步骤4 在测试吞吐量的同时，单板执行Shell命令：iwconfig wlan0，看Bit Rate是否大于或等于150Mbps。如果采用的是MTK的WiFi，Bit Rate为72Mbps，那么修改RT2870STA.dat文件中的HT_BW=1。

步骤5 AP和单板是否不在一个平面上，不在一个平面上会影响WiFi性能。

步骤6 如果AP是两根天线，这两根天线和单板的天线是否成一条直线，它们在一条直线上也会影响WiFi性能。

步骤7 用WiFi分析仪看看周围的AP是否很多，对WiFi造成干扰。

步骤8 检查WiFi天线是否插好。

步骤9 检查WiFi设备是否和HDMI接口在5cm范围内，HDMI会对WiFi造成很大的干扰。

步骤10 检查硬件上WiFi模组的地线是否干净。

步骤11 单板和PC机采用有线网络连接，测试吞吐量，看是否PC机有速率限制。

步骤12 将AP恢复出厂设置。

----结束

6.2.4 扫描不到 AP

问题描述

单板和AP的距离很近，但是单板扫描不到AP。

问题分析

原因可能是：

- 驱动没有设置好国家码，不支持AP的信道；
- WiFi模组的地线不干净或HDMI干扰。

解决办法

对于原因1，按照厂家的指导设置好国家码，重新编译并烧入。

对于原因2，让模组厂家检查WiFi硬件设计。

6.2.5 连接不上 AP

问题描述

可以扫描到AP，但是连接不上。

解决办法

原因也有很多，可以按照以下步骤排查：

步骤1 检查密码设置得是否正确；

步骤2 单板的天线是否接好；

步骤3 WiFi是否和HDMI接口很近；

步骤4 使用OmniPeek抓取空口包，分析数据包中是否不遵从协议。

----结束

6.2.6 打开不了 WiFi

问题描述

Android平台，在UI上点击开启WiFi，无法打开。

解决办法

可以开启logcat，分析logcat打印信息和驱动log：

- logcat打印“Cannot find supported device”，说明WiFi模块损坏或者版本中不支持该WiFi。
- logcat打印“Failed to load driver！”，说明没有找到驱动、驱动初始化失败、驱动已经加载了没有卸载掉。
- 驱动已经加载了，初始化时打印错误信息（具体打印的出错信息根据WiFi芯片不同而不同），执行iwconfig无wlan0网口。这种情况可能是由于驱动不匹配；分配内存失败；WiFi模组有问题。
- logcat打印“Supplicant not running, cannot connect”，是由于wpa_supplicant进程启动失败，执行“ps”命令确认一下wpa_supplicant进程是否在运行，如果没有运行，那么init.rc中的wpa_supplicant服务参数错误，修改正确后编译kernel，将kernel镜像烧入单板。
- logcat打印“Unable to open connection to supplicant on xxx”，是由于HAL和wpa_supplicant进程建立socket连接失败。可能的原因是init.rc中的wpa_supplicant服务参数错误或者wpa_supplicant.conf中的ctrl_interface参数设置错误，应该设置为“ctrl_interface=wlan0”。
- 在UI上看到WiFi反复自动打开关闭，这种情况是由于驱动不支持wpa_supplicant的某些命令导致的，请检查驱动是否支持：SCAN-ACTIVE、SCAN-PASSIVE、RSSI、LINKSPEED、BTCOEXSCAN-START、BTCOEXSCAN-STOP、BTCOEXMODE、MACADDR、GETBAND、SETBAND等。

6.2.7 连接不上某个 AP，连接其它的 AP 没有问题

问题描述

连接某个AP总是连不上、很难连上或者连上后容易断开。连接其它的AP都没有问题，其它的WiFi和单板连接这个AP也没有问题。

问题分析

出现这个问题有可能是WiFi模组出现问题，因为AP使用频繁会出现一些频偏，如果WiFi模组也出现一些频偏或者有些模组的射频质量不好，就会出现这种现象。

解决办法

将AP恢复出厂设置，如果问题还在替换单板上的模组。

6.2.8 待机唤醒问题

问题描述

开启WiFi后，待机唤醒可能会出现的问题：待机时内核崩溃；待不下去；唤醒后出现内核崩溃；唤醒后WiFi无法使用；连接上AP后不能立刻进入待机。

问题分析

Android平台是为手机、平板而设计，WiFi待机机制不适用于机顶盒，另外，Android的原始设计是基于SDIO接口的WiFi，待机时不断电，USB接口的WiFi在待机时是会断电的。

解决办法

不管是Linux平台还是Andrid平台，建议WiFi的待机策略改为待机前关闭WiFi，唤醒后再打开。

注意

Android平台，如果WiFi连接上AP，关闭WiFi时，Android会申请一个60秒的wakelock，让设备有时间连接上移动数据网，这样可以保证待机时仍可以收到网络数据包，所以可能会出现关闭WiFi后60秒后才进入待机状态。这种情况下需要注释掉WifiService中的mCm.requestNetworkTransitionWakelock(TAG)。

6.2.9 MU 连接过 WiFi Direct 后换其它的 WiFi 就连不上了

问题描述

单板上插入MT7601U模块，和手机进行过WiFi Direct或Miracast业务后，将MT761U换成其它WiFi模块，WiFi Direct和Miracast总是连不上，就算重启单板和手机都没用。

问题分析

MT7601U开启WiFi Direct（Miracast也是使用了WiFi Direct）时，连接后新建了一个网口，并保存在单板内，后续的连接都需要使用这个网口，而其它WiFi不支持该网口。

解决办法

避免使用MT7601U连接WiFi Direct后再使用其它WiFi模块。

6.2.10 打开 WiFi 的编译配置后编译失败

问题描述

打开WiFi的编译配置后，完整编译整个工程，出现编译失败，显示错误信息：

```
cannot find -lnl-genl
```

或者以下错误信息：

```
bison -y -d  -o route/pktloc_syntax.c route/pktloc_syntax.y
route/pktloc_syntax.y:11.9-16: syntax error, unexpected identifier,
expecting string
```

问题分析

WiFi模块需要使用libnl，编译libnl需要服务器安装bison和flex。出现第二个错误信息是由于bison的版本太旧引起的。

解决办法

在编译服务器上安装bison 2.4.1或更高的版本，flex 2.5.35或更高的版本。

6.2.11 Android 平台，在“设置”中打开 WiFi，WiFi 反复开启关闭

问题描述

Android平台，用命令行方式加载WiFi驱动，扫描都是正常的，但是在“设置”界面上打开WiFi，出现刚打开又自动关闭，然后反复打开关闭。

问题分析

原因是framework和wpa_supplicant之间的socket连接失败，framework连接不上wpa_supplicant时会将wpa_supplicant进程重启，然后再建立socket连接，这样反复多次。造成这种情况的原因也有两个：

- 第一种是由于init.xxx.rc中启动wpa_supplicant的参数错误，尤其是-D参数，参数错误或者参数次序不对会造成socket节点创建失败或者wpa_supplicant初始化失败；
- 第二种是由于WiFi驱动不支持Android的几个命令，从而wpa_supplicant初始化时失败，Android的命令包括但不限于以下几个：
  - SCAN-ACTIVE
  - SCAN-PASSIVE
  - RSSI、LINKSPEED
  - BTCOEXMODE
  - MACADDR。

解决办法

- 第一种原因需要修改init.xxx.rc中的wpa_supplicant参数，具体修改方法还需要看具体的问题。
- 第二种原因检查驱动是否支持Android的那些命令，与厂家联系在驱动中添加对这些命令的支持。

6.2.12 iperf 或 ping 测试时，测试一段时间后数据中断了

问题描述

单板通过WiFi连接上AP后，单板和AP进行长时间iperf或ping测试，测试几个小时后出现iperf数据中断或者ping不通了的问题。

问题分析

AP的DHCP更新时，WifiStateMachine收到更新事件，发送BTCOEXMODE给驱动，但驱动不支持该命令，于是wpa_supplicant返回CTRL-EVENT-DRIVER-STATE HUANG，WifiStateMachine收到这个事件后会关闭WiFi然后再开启，因此数据中断。

解决办法

在驱动中添加对BTCOEXMODE命令的支持，可以不进行处理，直接返回成功。

6.2.13 SoftAP 吞吐量低

问题描述

周围环境AP的数量很少，测试SoftAP吞吐量时，吞吐量只有5~6Mbps，或者只有20多Mbps。

问题分析

出现上述几种吞吐量的原因是发送速率没有使用11n的速率。吞吐量只有5~6Mbps是使用的11b速率，只有20多Mbps是使用的11g速率。11n速率在20MHz带宽时能够达到40多Mbps，40MHz带宽时能够达到80Mbps。Realtek和Atheros的WiFi SoftAP是使用hostapd配置的，hostapd.conf文件中没有正确配置好11n。

解决办法

修改hostapd.conf文件添加以下内容：

- 信道设置小于等于7时

```
hw_mode=g
ieee80211n=1
ht_capab=[SHORT-GT-20][SHORT-GI-40][HT40+]
```

- 信道设置大于7时

```
hw_mode=g
ieee80211n=1
ht_capab=[SHORT-GT-20][SHORT-GI-40][HT40-]
```

6.2.14 Android 版本 WiFi 高级设置中没有切换频带的选项

问题描述

Android版本，WiFi芯片是支持2.4G和5G的，打开WiFi设置里面的“高级”菜单，没有“WLAN频带”选项，不能切换2.4G或5G频带。类似的手机是有这样的选项的。

问题分析

是否有“WLAN频带”选项是通过frameworks/base/core/res/res/values/config.xml文件中的"config_wifi_dual_band_support"参数来配置的，当采用的WiFi芯片支持2.4G和5G频带，要将这个参数配置成true。由于版本要支持多款只支持2.4G的WiFi芯片，所以默认设置这个参数为false，因此在“高级”设置里没有“WLAN频带”选项。

解决办法

frameworks/base/core/res/res/values/config.xml文件中的"config_wifi_dual_band_support"参数为true，如下：

<bool translatable="false" name="config_wifi_dual_band_support">true</bool>

6.2.15 Linux 版本加载 WiFi 驱动时提示内存申请失败

问题描述

Linux版本，打开WiFi加载驱动时提示“Cannot allocate memory”。

问题分析

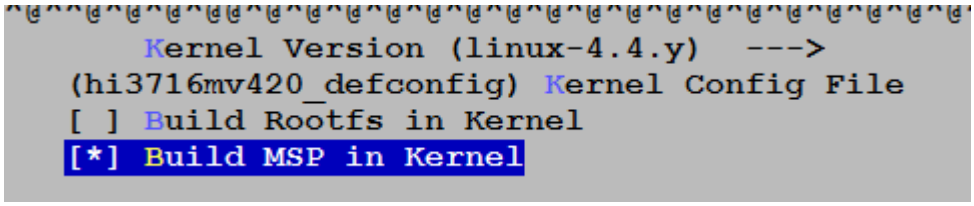
Linux版本的MSP模块默认编为module，将Linux系统的module空间几乎占满，导致WiFi加载驱动无法申请内存。

解决办法

如需使用WiFi模块，请将MSP模块build in内核，方法如下：

Make menuconfig进入Kernel配置，选中Build MSP in Kernel

图 6-3 将 MSP 模块 build in 内核



注意

将MSP Build in内核后，hi_kernel.bin镜像大小会增加。

6.2.16 USB、PCI-e、SDIO 接口 WiFi 打开优先级

问题描述

WiFi芯片有USB、PCI-e和SDIO接口，当这三种接口同时接了WiFi芯片时，WiFi打开的优先级如何？

解决办法

一套镜像可以自适应WiFi芯片，SDK中目前是先查找PCI-e接口，然后查找USB接口，然后SDIO接口；如果找到了支持的PCI-e接口WiFi，则加载对应驱动，不会再检测其他接口；USB接口同理；如果PCI-e和USB节点下都没有找到WiFi设备，默认加载cfg.mak

中配置的SDIO接口WiFi芯片，如果cfg.mk中没有选择SDIO接口WiFi，则打开WiFi失败。