Rockchip

WIFI/BT 开发指南

发布版本**:4.0**

日期**:2018.11**

免责声明

商标声明

版权所有 © 2018 福州瑞芯微电子股份有限公司

福州瑞芯微电子股份有限公司
Fuzhou Rockchip Electronics Co., Ltd.

地址：	福建省福州市铜盘路软件园 A 区 18 号
网址：	www.rock-chips.com
客户服务电话：	+86-591-83991906
客户服务传真：	+86-591-83951833
客户服务邮箱：	www.rock-chips.com

前言

概述

本文档主要介绍基于 Rockchip 平台的 WIFI、BT 的内核配置、相关功能的开发等等；

产品版本

芯片名称	内核版本
RK3308	4.4

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

修订记录

日期	版本	作者	修改说明
2018/05/02	0.01	XY	初始版本
2018/05/16	1.0	XY	正式版本
2018/06/22	2.0	CTF	正式版本
2018/07/21	3.0	CTF	正式版本
2018/11/20	4.0	XY	正式版本
2018/11/22	5.0	CTF	正式版本

目录

目录

前言

此文档主要介绍 RK linux 平台下 WiFiBT 的开发，作为该文档的补充：RTL 系列芯片参见:\docs\Linux reference documents: RK3308_RTL8723DS_WIFI_BT_说明文档_V1.20.pdf

1 WIFI/BT 内核配置

1.1 DTS

WIFI 硬件管脚的配置主要有以下几点：

WIFI_REG_ON: WiFi 的电源 PIN 脚

```
sdio_pwrseq: sdio-pwrseq {
     compatible = "mmc-pwrseq-simple";
     pinctrl-names = "default";
     pinctrl-0 = <&wifi_enable_h>;
     reset-gpios = <&gpio0 RK_PA2 GPIO_ACTIVE_LOW>; //有个注意要点是：这里的电平状态恰
好跟使能状态相反，比如 REG_ON 高有效，则这里为 LOW；如果 REG_ON 低有效，则填 HIGH
};
&pinctrl {
    sdio-pwrseq {
        wifi_enable_h: wifi-enable-h {
            rockchip,pins =
            <0 RK_PA2 RK_FUNC_GPIO &pcfg_pull_none>; // WIFI_REG_ON
        };
    };
};
```

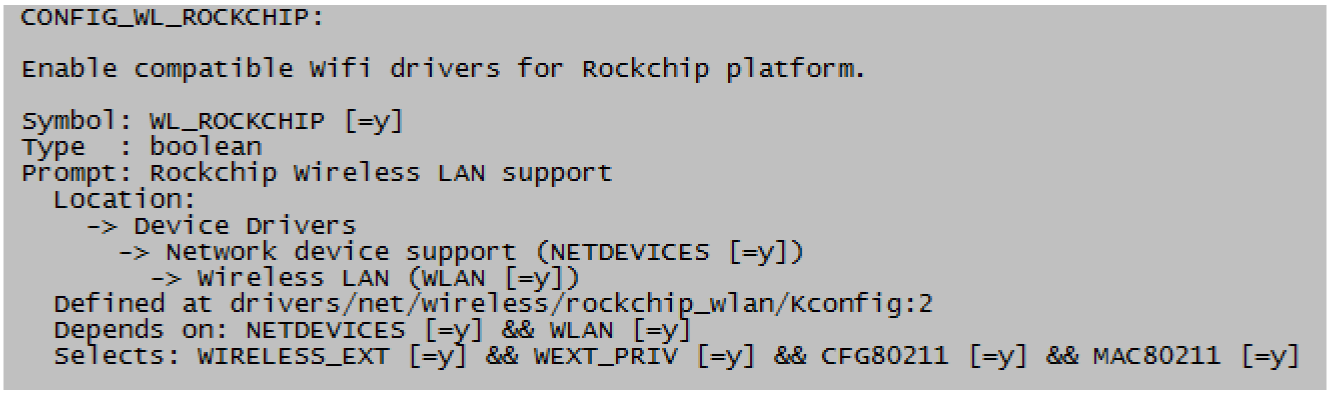
WIFI_WAKE_HOST: WIFI 唤醒主控的 PIN 脚

```
wireless-wlan {
    compatible = "wlan-platdata";
    rockchip,grf = <&grf>;
    wifi_chip_type = "ap6255"; //海华/正基模组可以不用修改此名称，realtek 需要按实际填写
    WIFI,host_wake_irq = <&gpio0 RK_PA0 GPIO_ACTIVE_HIGH>; //WIFI_WAKE_HOST
GPIO_ACTIVE_HIGH 特别注意：确认下这个 wifi pin 脚跟主控的连接关系，如果中间加了一个反向管就要
改成低电平触发
    status = "okay";
};
```

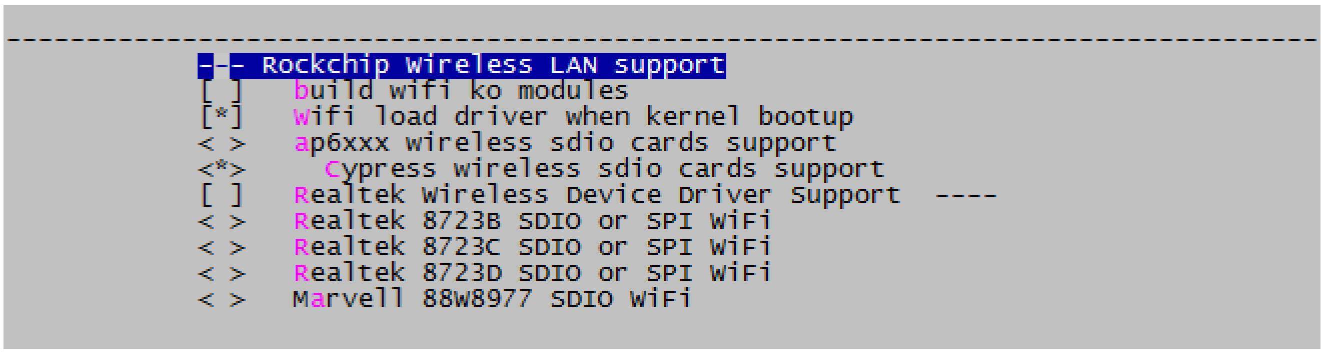
```
wireless-bluetooth {
    compatible = "bluetooth-platdata";
    uart_rts_gpios = <&gpio4 RK_PA7 GPIO_ACTIVE_LOW>;
    pinctrl-names = "default", "rts_gpio";
    pinctrl-0 = <&uart4_rts>;
    pinctrl-1 = <&uart4_rts_gpio>;
    BT,power_gpio     = <&gpio4 RK_PB3 GPIO_ACTIVE_HIGH>; // BT_REG_ON
    BT,wake_host_irq = <&gpio4 RK_PB4 GPIO_ACTIVE_HIGH>; // BT_WAKE_HOST
    status = "okay";
};
```

1.2 内核

根据实际 WiFi 选择对应配置



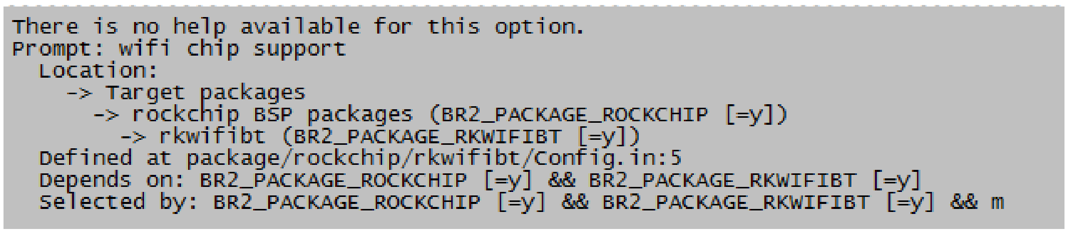
```
CONFIG_WL_ROCKCHIP:

Enable compatible Wifi drivers for Rockchip platform.

Symbol: WL_ROCKCHIP [=y]
Type  : boolean
Prompt: Rockchip Wireless LAN support
  Location:
    -> Device Drivers
      -> Network device support (NETDEVICES [=y])
        -> Wireless LAN (WLAN [=y])
  Defined at drivers/net/wireless/rockchip_wlan/Kconfig:2
  Depends on: NETDEVICES [=y] && WLAN [=y]
  Selects: WIRELESS_EXT [=y] && WEXT_PRIV [=y] && CFG80211 [=y] && MAC80211 [=y]
```



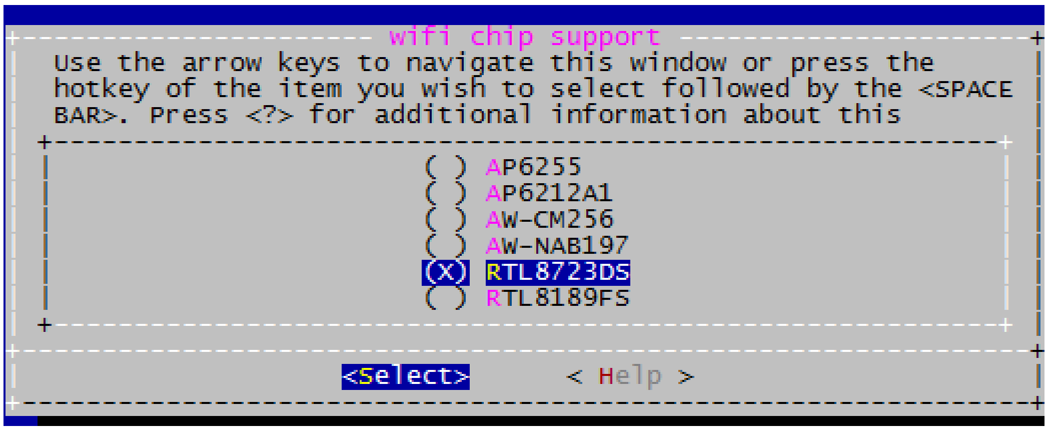
```
----------------------------------------------------------------------------------
            --- Rockchip Wireless LAN support
            [ ]   build wifi ko modules
            [*]   Wifi load driver when kernel bootup
            < >   ap6xxx wireless sdio cards support
            <*>     Cypress wireless sdio cards support
            [ ]   Realtek Wireless Device Driver Support  ----
            < >   Realtek 8723B SDIO or SPI WiFi
            < >   Realtek 8723C SDIO or SPI WiFi
            < >   Realtek 8723D SDIO or SPI WiFi
            < >   Marvell 88W8977 SDIO WiFi
```

Buildroot 配置：

根据实际 WiFi 选择对应配置，要跟内核配置一致：



```
There is no help available for this option.
Prompt: wifi chip support
  Location:
    -> Target packages
      -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
        -> rkwifibt (BR2_PACKAGE_RKWIFIBT [=y])
  Defined at package/rockchip/rkwifibt/Config.in:5
  Depends on: BR2_PACKAGE_ROCKCHIP [=y] && BR2_PACKAGE_RKWIFIBT [=y]
  Selected by: BR2_PACKAGE_ROCKCHIP [=y] && BR2_PACKAGE_RKWIFIBT [=y] && m
```



```
+---------------------- wifi chip support ----------------------+
|  Use the arrow keys to navigate this window or press the      |
|  hotkey of the item you wish to select followed by the <SPACE |
|  BAR>. Press <?> for additional information about this        |
| +-----------------------------------------------------------+ |
| |                        ( ) AP6255                         | |
| |                        ( ) AP6212A1                       | |
| |                        ( ) AW-CM256                       | |
| |                        ( ) AW-NAB197                      | |
| |                        (X) RTL8723DS                      | |
| |                        ( ) RTL8189FS                      | |
| +-----------------------------------------------------------+ |
+---------------------------------------------------------------+
|                <Select>      < Help >                         |
+---------------------------------------------------------------+
```

2 配网开发

2.1 命令行配网：

首先确保 WiFi 的服务进程启动： ps | grep wpa_supplicant，如果没启动请手动启动：

```
wpa_supplicant -B -i wlan0 -c /data/cfg/wpa_supplicant.conf
```

修改如下文件：

```
/ # vi /data/cfg/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1
```

#添加如下配置项

```
network={
          ssid="WiFi-AP"             // WiFi 名字
          psk="12345678"             // WiFi 密码
          key_mgmt=WPA-PSK    // 选填加密方式，不填的话可以自动识别
          # key_mgmt=NONE       // 不加密
}
```

重新读取上述配置：wpa_cli reconfigure

并重新连接：wpa_cli reconnect

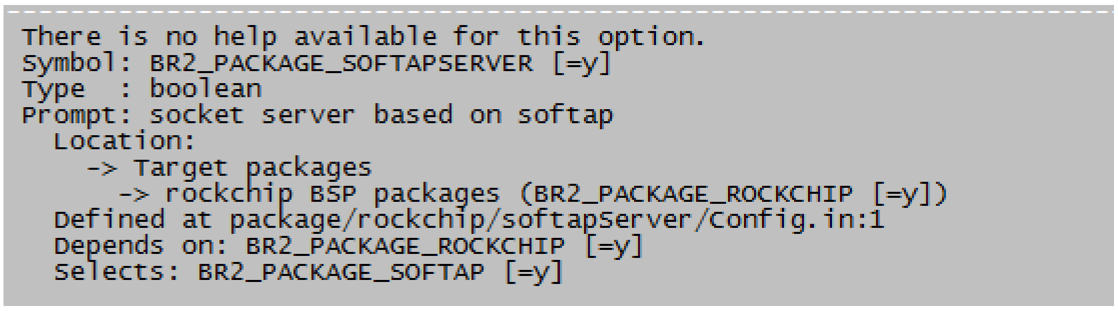
2.2 手机配网：

2.2.1 Softap 配网

APP: /external/app/RkEcho.apk

简介：首先，用 SDK 板的 WiFi 创建一个 AP 热点，在手机端连接该 AP 热点；其次，通过手机端 apk 获取 SDK 板的当前扫描到的热点列表，在手机端填入要连接 AP 的密码，apk 会把 AP 的 ssid 和密码发到 SDK 端；最后，SDK 端会根据收到的信息连接 WiFi。

Buildroot 配置：



```
There is no help available for this option.
Symbol: BR2_PACKAGE_SOFTAPSERVER [=y]
Type  : boolean
Prompt: socket server based on softap
  Location:
    -> Target packages
      -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/softapServer/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y]
  Selects: BR2_PACKAGE_SOFTAP [=y]
```

源码开发目录：

```
/external/softapServer/ -- WIFI 与 APK 端相关操作
/external/softapDemo/ -- WiFi 相关操作
```

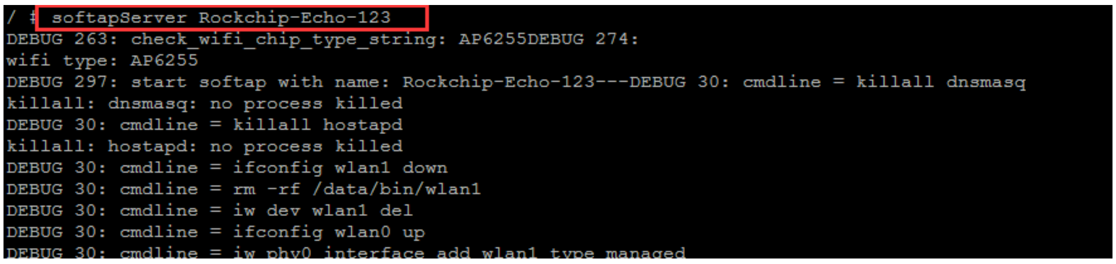
准备手机安装 apk：

确保 wifi server 进程启动

# wpa_supplicant -B -i wlan0 -c /data/cfg/wpa_supplicant.conf

第一步：板子的命令行执行：

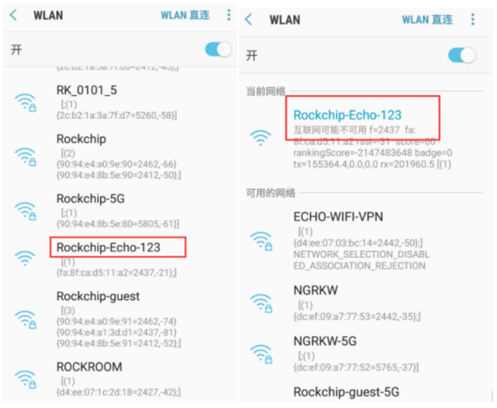
softapServer Rockchip-Echo-123 (wifi 热点的名字，前缀必须为 Rockchip-Echo-xxx)



```
/ # softapServer Rockchip-Echo-123
DEBUG 263: check_wifi_chip_type_string: AP6255DEBUG 274:
wifi type: AP6255
DEBUG 297: start softap with name: Rockchip-Echo-123---DEBUG 30: cmdline = killall dnsmasq
killall: dnsmasq: no process killed
DEBUG 30: cmdline = killall hostapd
killall: hostapd: no process killed
DEBUG 30: cmdline = ifconfig wlan1 down
DEBUG 30: cmdline = rm -rf /data/bin/wlan1
DEBUG 30: cmdline = iw dev wlan1 del
DEBUG 30: cmdline = ifconfig wlan0 up
DEBUG 30: cmdline = iw phy0 interface add wlan1 type managed
```

第二步：打开手机的 wifi setting 界面：

找到 Rockchip-Echo-123，点击连接；

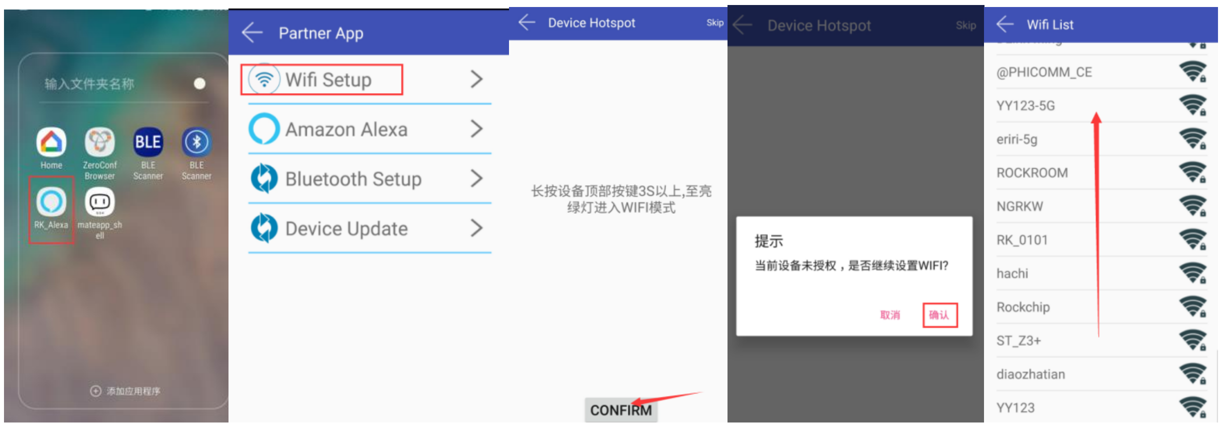


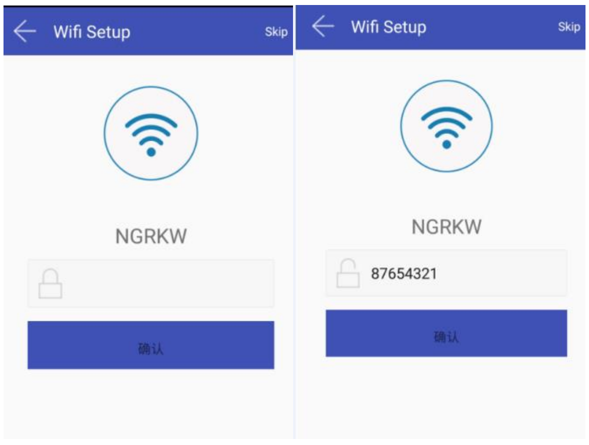
```
WLAN                                   WLAN 直连
开
RK_0101_5
[:(1)
{2c:b2:1a:3a:7f:d7=5260,-58}]
Rockchip
[(2)
{90:94:e4:a0:9e:90=2462,-66}
{90:94:e4:8b:5e:90=2412,-50};]
Rockchip-5G
[:(1)
{90:94:e4:8b:5e:80=5805,-61}]
Rockchip-Echo-123
[(1)
{fa:8f:ca:d5:11:a2=2437,-21};]
Rockchip-guest
[(3)
{90:94:e4:a0:9e:91=2462,-74}
{90:94:e4:a1:3d:d1=2437,-81}
{90:94:e4:8b:5e:91=2412,-52};]
ROCKROOM
[(1)
{d4:ee:07:1c:2d:18=2427,-42};]
```

```
WLAN                                   WLAN 直连
开
当前网络
Rockchip-Echo-123
互联网可能不可用 f=2437 fa:
8f:ca:d5:11:a2 rssi=-31 score=60
rankingScore=-2147483648 badge=0
tx=155364.4,0.0,0.0 rx=201960.5 [(1)
可用的网络
ECHO-WIFI-VPN
[(1)
{d4:ee:07:03:bc:14=2442,-50};]
NETWORK_SELECTION_DISABL
ED_ASSOCIATION_REJECTION
NGRKW
[(1)
{dc:ef:09:a7:77:53=2442,-35};]
NGRKW-5G
[:(1)
{dc:ef:09:a7:77:52=5765,-37}]
Rockchip-guest-5G
```

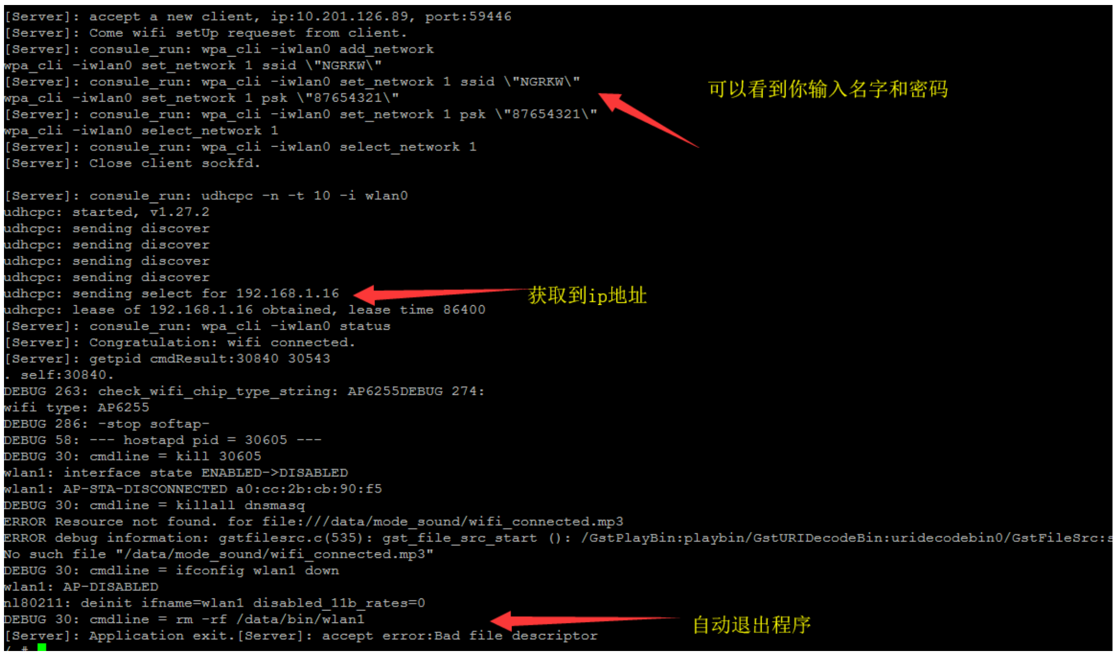
第三步：打开手机 apk：

打开 apk，点击 wifi setup->CONFIRM->确认->wifi 列表->点击你要连接的网络名字->输入密码->点击确认





板子串口端显示：



```
[Server]: accept a new client, ip:10.201.126.89, port:59446
[Server]: Come wifi setUp requeset from client.
[Server]: consule_run: wpa_cli -iwlan0 add_network
wpa_cli -iwlan0 set_network 1 ssid \"NGRKW\"
[Server]: consule_run: wpa_cli -iwlan0 set_network 1 ssid \"NGRKW\"
wpa_cli -iwlan0 set_network 1 psk \"87654321\"
[Server]: consule_run: wpa_cli -iwlan0 set_network 1 psk \"87654321\"
wpa_cli -iwlan0 select_network 1
[Server]: consule_run: wpa_cli -iwlan0 select_network 1
[Server]: Close client sockfd.

[Server]: consule_run: udhcpc -n -t 10 -i wlan0
udhcpc: started, v1.27.2
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.1.16
udhcpc: lease of 192.168.1.16 obtained, lease time 86400
[Server]: consule_run: wpa_cli -iwlan0 status
[Server]: Congratulation: wifi connected.
[Server]: getpid cmdResult:30840 30543
. self:30840.
DEBUG 263: check_wifi_chip_type_string: AP6255DEBUG 274:
wifi type: AP6255
DEBUG 286: -stop softap-
DEBUG 58: --- hostapd pid = 30605 ---
DEBUG 30: cmdline = kill 30605
wlan1: interface state ENABLED->DISABLED
wlan1: AP-STA-DISCONNECTED a0:cc:2b:cb:90:f5
DEBUG 30: cmdline = killall dnsmasq
ERROR Resource not found. for file:///data/mode_sound/wifi_connected.mp3
ERROR debug information: gstfilesrc.c(535): gst_file_src_start (): /GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0/GstFileSrc:s
No such file "/data/mode_sound/wifi_connected.mp3"
DEBUG 30: cmdline = ifconfig wlan1 down
wlan1: AP-DISABLED
nl80211: deinit ifname=wlan1 disabled_11b_rates=0
DEBUG 30: cmdline = rm -rf /data/bin/wlan1
[Server]: Application exit.[Server]: accept error:Bad file descriptor
```

检查网络是否连通：

```
/ # echo nameserver 8.8.8.8 > etc/resolv.conf //  添加 dns 域名解析
/ # ping www.baidu.com //看下是否 ping 通
```

注意要点：

1、softspServer Rockchip-Echo-123 执行后命令行是无法退出的，直到配网完成；

2、名字千万不要写错，否则 apk 无法进入确认界面（Rockchip-Echo-xxx）

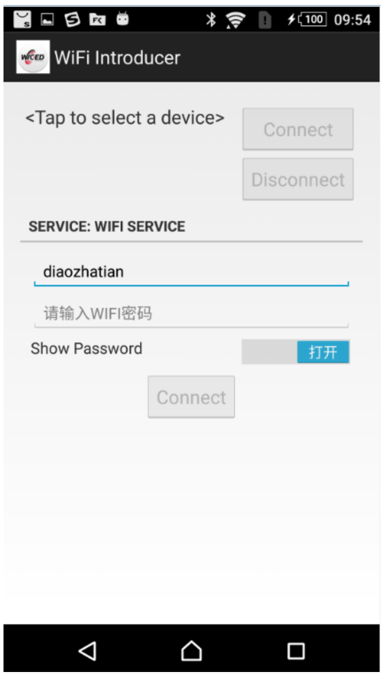
2.2.2 蓝牙配网

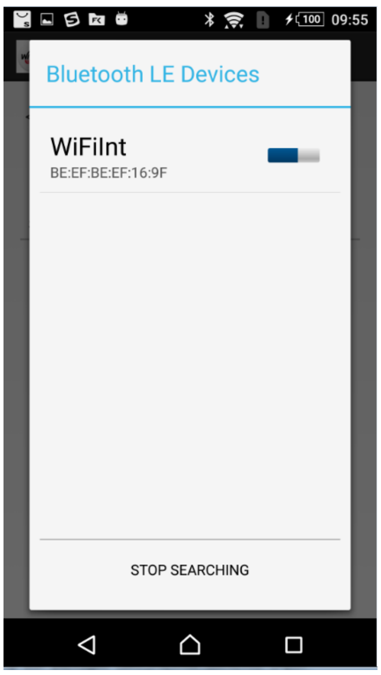
1、Realtek 请参考：\docs\Linux reference documents: RK3308_RTL8723DS_WIFI_BT_说明文档_V1.20.pdf。

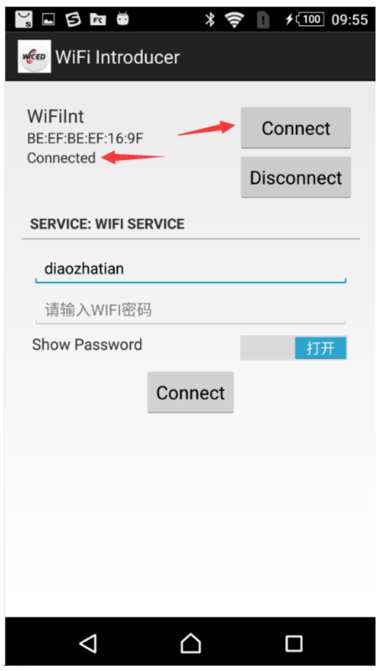
2、apk 路径：external/app/ WiFiIntroducer.apk

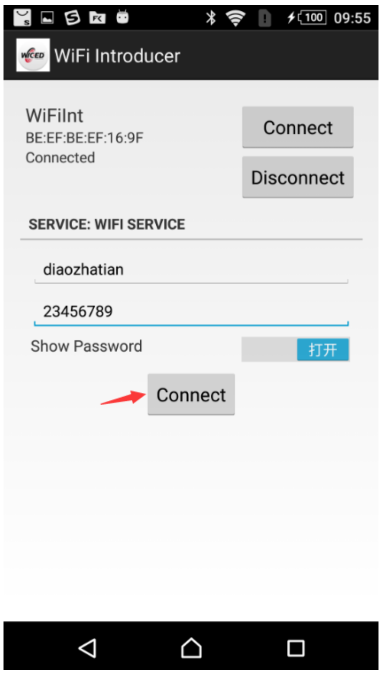
3、AP 蓝牙配网，板端运行 bsa_ble_wifi_introducer.sh start

4、手机端 app：点击 Tap to select a device -> 选择带有配网服务的设备 -> Connect 连接设备 -> device 选项框显示连接状态 ->如果没有进行过配对，会弹窗或通知栏提示配对，不配对会导致配对超时、蓝牙连接断开 -> 输入 WiFi 名称和密码，Connect



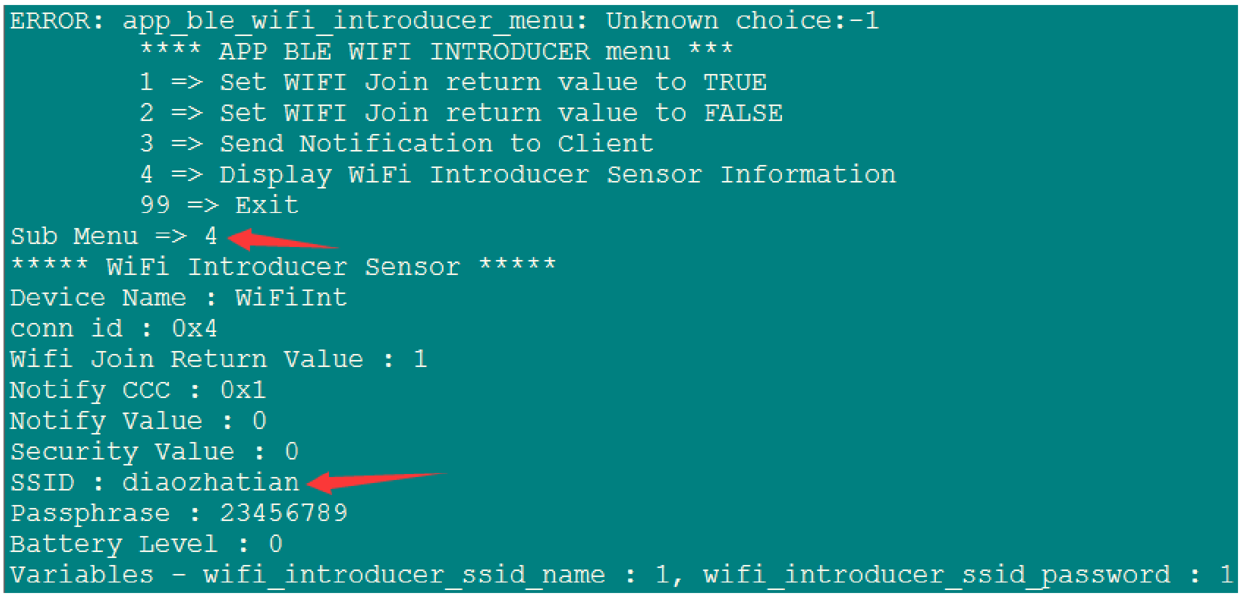






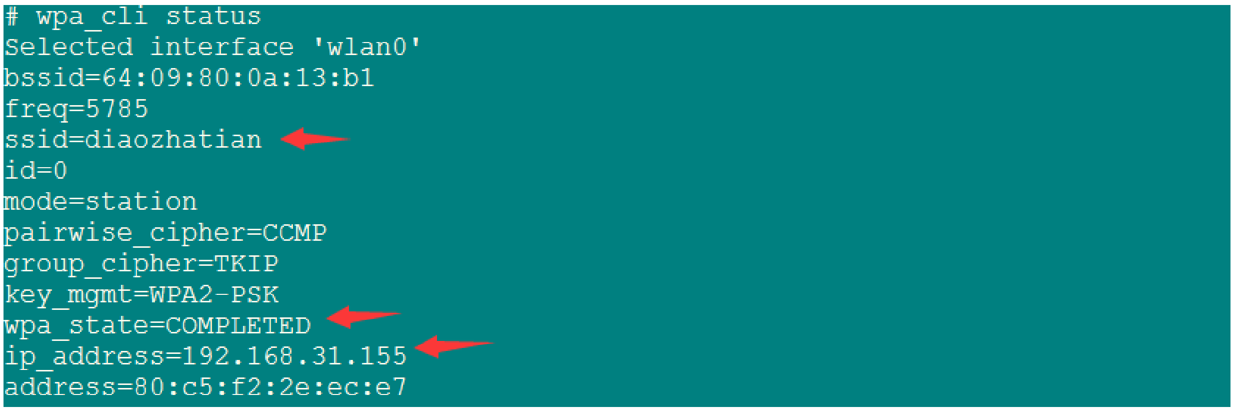
5、板端收到 SSID 和 Passphrase 后，存储在 data/bsa/wpa_supplicant.conf；命令行 menu 选择 4 => Display WiFi Introducer Sensor Information，可以查看配置的 SSID 和 Passphrase

```
# cat data/bsa/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1
network={
ssid="diaozhatian"
psk="7788123456"
key_mgmt=WPA-PSK
}
```



```
ERROR: app_ble_wifi_introducer_menu: Unknown choice:-1
        **** APP BLE WIFI INTRODUCER menu ***
        1 => Set WIFI Join return value to TRUE
        2 => Set WIFI Join return value to FALSE
        3 => Send Notification to Client
        4 => Display WiFi Introducer Sensor Information
        99 => Exit
Sub Menu => 4
***** WiFi Introducer Sensor *****
Device Name : WiFiInt
conn id : 0x4
Wifi Join Return Value : 1
Notify CCC : 0x1
Notify Value : 0
Security Value : 0
SSID : diaozhatian
Passphrase : 23456789
Battery Level : 0
Variables - wifi_introducer_ssid_name : 1, wifi_introducer_ssid_password : 1
```

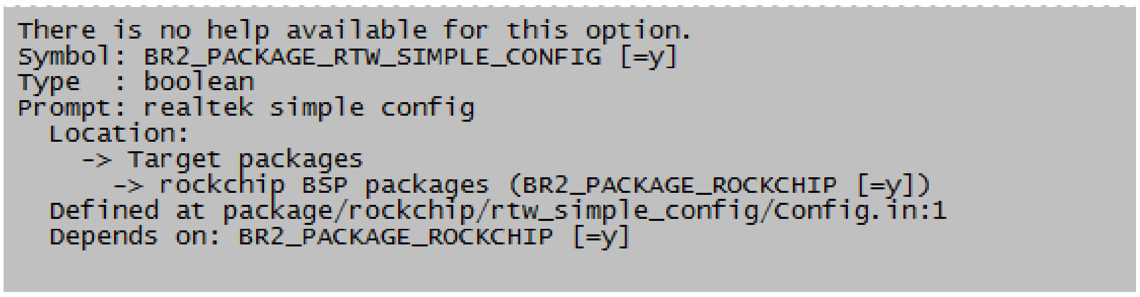
6、板端启动 wpa_supplicant、dhcpcd 开始配网，配网完成后，命令行运行 wpa_cli status 查看网络连接状态和 ip 地址



```
# wpa_cli status
Selected interface 'wlan0'
bssid=64:09:80:0a:13:b1
freq=5785
ssid=diaozhatian
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=TKIP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.31.155
address=80:c5:f2:2e:ec:e7
```

2.2.3 Simple config 配网

Realtek 模组：



```
There is no help available for this option.
Symbol: BR2_PACKAGE_RTW_SIMPLE_CONFIG [=y]
Type  : boolean
Prompt: realtek simple config
  Location:
    -> Target packages
      -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/rtw_simple_config/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y]
```

内核修改：

```
--- a/drivers/net/wireless/rockchip_wlan/rtl8xxx/Makefile
+++ b/drivers/net/wireless/rockchip_wlan/rtl8xxx /Makefile
@@ -68,7 +68,7 @@ CONFIG_80211W = n
 CONFIG_REDUCE_TX_CPU_LOADING = n
 CONFIG_BR_EXT = y
 CONFIG_TDLS = n
-CONFIG_WIFI_MONITOR = n
```

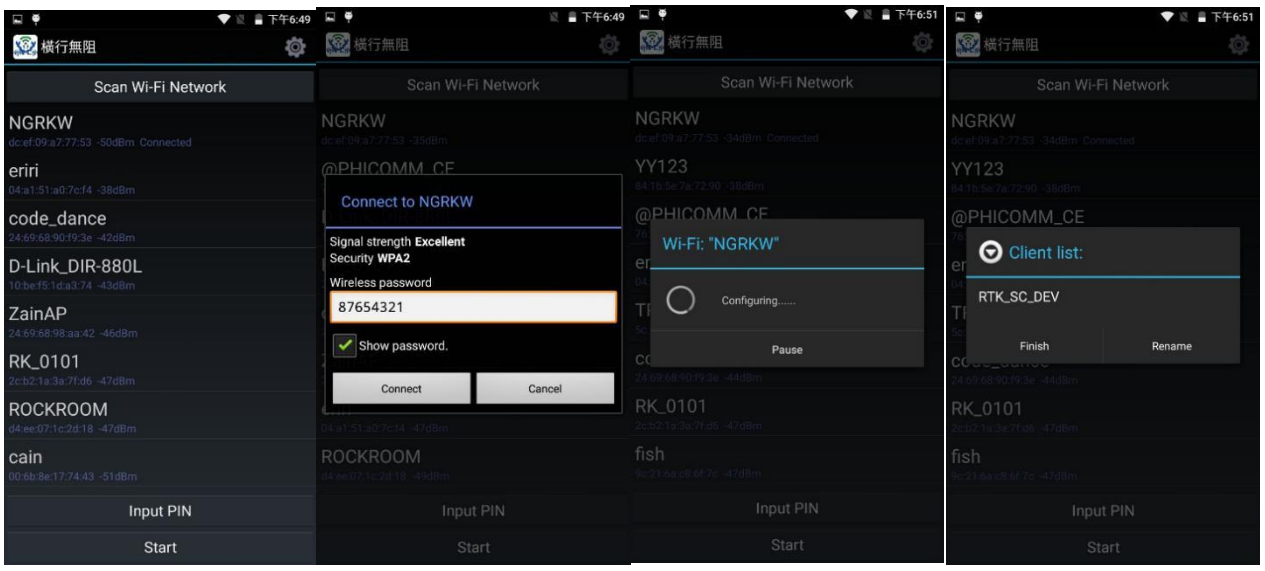
+CONFIG_WIFI_MONITOR = y

仅支持 realtek 模组

external/app/SimpleConfigApp.apk

命令行执行 rtw_simple_config –D & (rtw_simple_config –h 查看帮助)

手机端按照 app：选择网络->输入密码->点击 start 发送->配置完成



Connect to NGRKW

Signal strength **Excellent**
Security **WPA2**
Wireless password
87654321
✓ Show password.

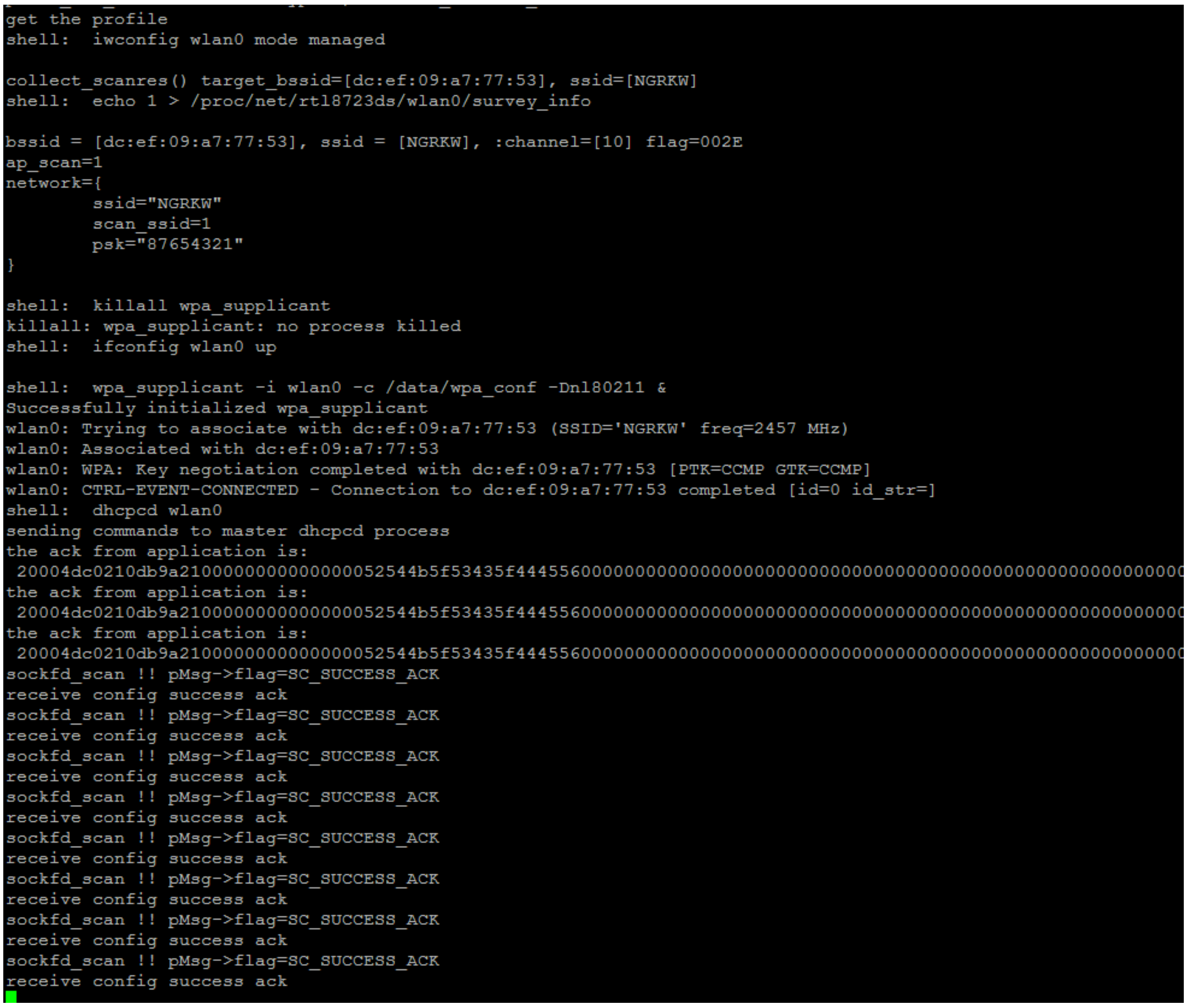
Wi-Fi: "NGRKW"

Configuring......

Client list:

RTK_SC_DEV

板端显示如下：



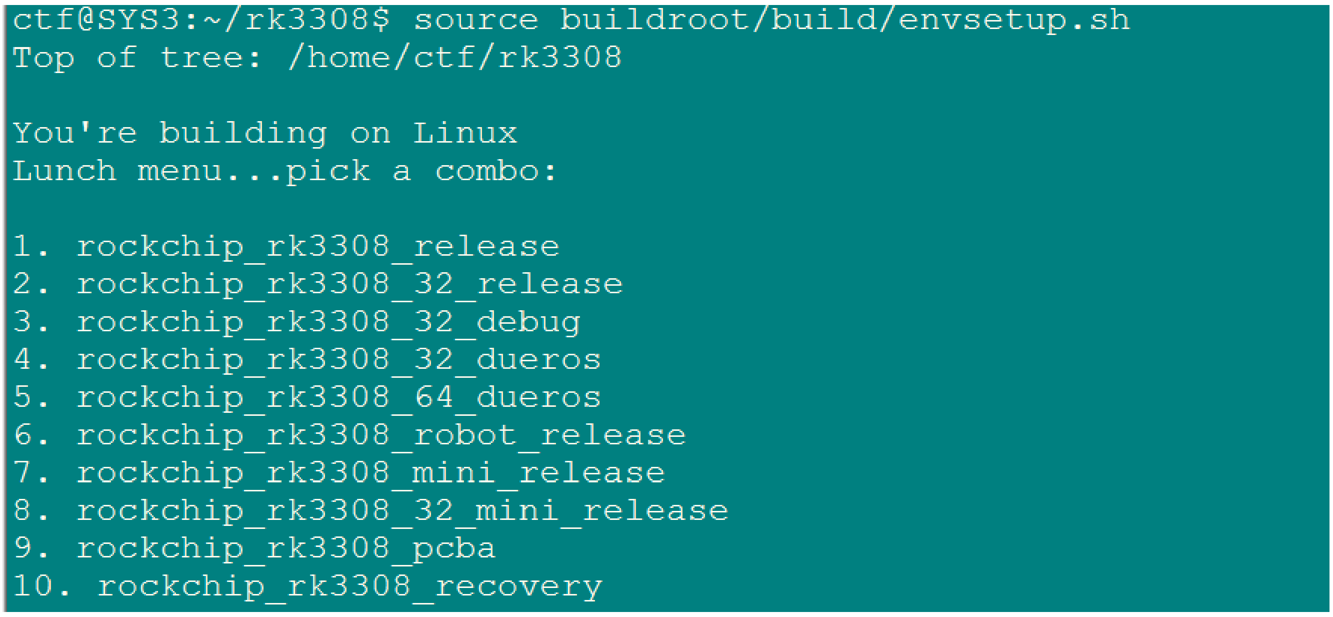
```
get the profile
shell:  iwconfig wlan0 mode managed

collect_scanres() target_bssid=[dc:ef:09:a7:77:53], ssid=[NGRKW]
shell:  echo 1 > /proc/net/rtl8723ds/wlan0/survey_info

bssid = [dc:ef:09:a7:77:53], ssid = [NGRKW], :channel=[10] flag=002E
ap_scan=1
network={
        ssid="NGRKW"
        scan_ssid=1
        psk="87654321"
}

shell:  killall wpa_supplicant
killall: wpa_supplicant: no process killed
shell:  ifconfig wlan0 up

shell:  wpa_supplicant -i wlan0 -c /data/wpa_conf -Dnl80211 &
Successfully initialized wpa_supplicant
wlan0: Trying to associate with dc:ef:09:a7:77:53 (SSID='NGRKW' freq=2457 MHz)
wlan0: Associated with dc:ef:09:a7:77:53
wlan0: WPA: Key negotiation completed with dc:ef:09:a7:77:53 [PTK=CCMP GTK=CCMP]
wlan0: CTRL-EVENT-CONNECTED - Connection to dc:ef:09:a7:77:53 completed [id=0 id_str=]
shell:  dhcpcd wlan0
sending commands to master dhcpcd process
the ack from application is:
 20004dc0210db9a21000000000000000052544b5f53435f4444556000000000000000000000000000000000000000000000000000000
the ack from application is:
 20004dc0210db9a21000000000000000052544b5f53435f4444556000000000000000000000000000000000000000000000000000000
the ack from application is:
 20004dc0210db9a21000000000000000052544b5f53435f4444556000000000000000000000000000000000000000000000000000000
sockfd_scan !! pMsg->flag=SC_SUCCESS_ACK
receive config success ack
sockfd_scan !! pMsg->flag=SC_SUCCESS_ACK
receive config success ack
sockfd_scan !! pMsg->flag=SC_SUCCESS_ACK
receive config success ack
sockfd_scan !! pMsg->flag=SC_SUCCESS_ACK
receive config success ack
sockfd_scan !! pMsg->flag=SC_SUCCESS_ACK
receive config success ack
sockfd_scan !! pMsg->flag=SC_SUCCESS_ACK
receive config success ack
sockfd_scan !! pMsg->flag=SC_SUCCESS_ACK
receive config success ack
sockfd_scan !! pMsg->flag=SC_SUCCESS_ACK
receive config success ack
```

3 蓝牙应用开发

3.1 海华/正基模组

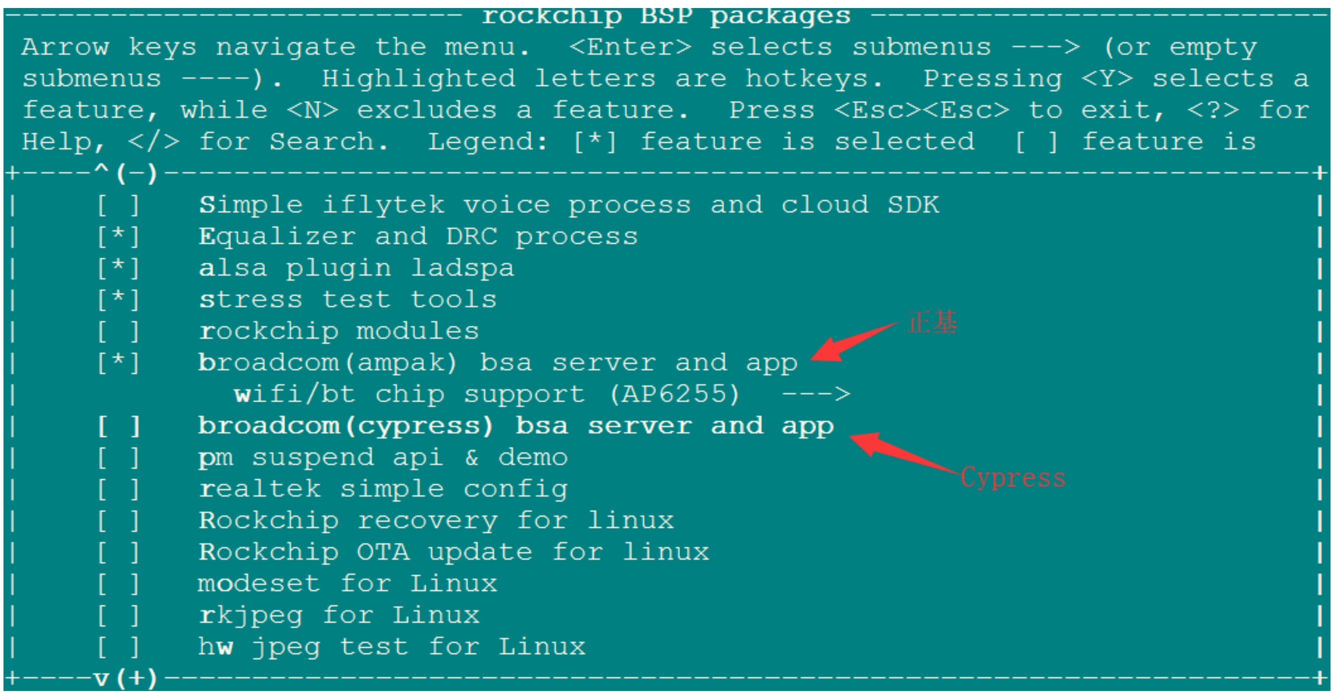
3.1.1 配置

1、source buildroot/build/envsetup.sh，选择想要编译的版本，一般选择 rockchip_rk3308_release



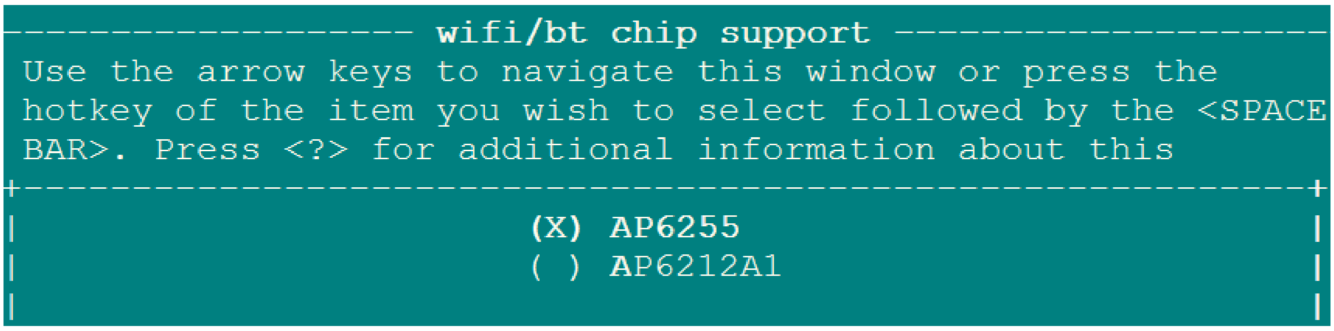
```
ctf@SYS3:~/rk3308$ source buildroot/build/envsetup.sh
Top of tree: /home/ctf/rk3308

You're building on Linux
Lunch menu...pick a combo:

1. rockchip_rk3308_release
2. rockchip_rk3308_32_release
3. rockchip_rk3308_32_debug
4. rockchip_rk3308_32_dueros
5. rockchip_rk3308_64_dueros
6. rockchip_rk3308_robot_release
7. rockchip_rk3308_mini_release
8. rockchip_rk3308_32_mini_release
9. rockchip_rk3308_pcba
10. rockchip_rk3308_recovery
```

2、make menuconfig，



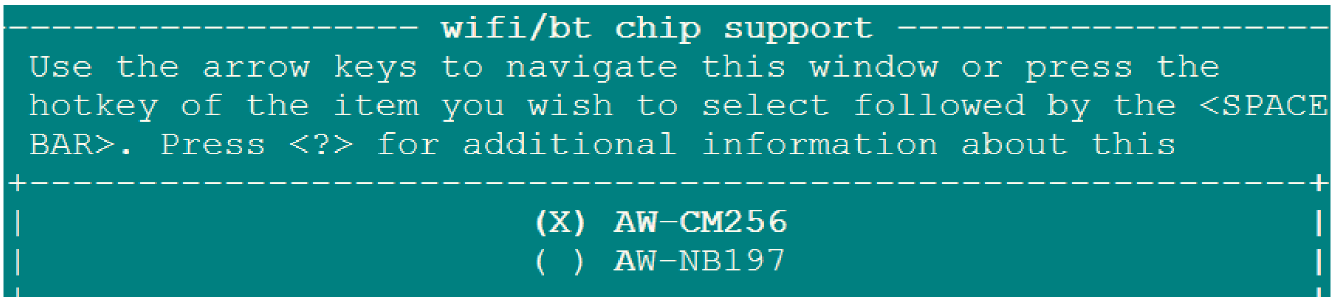
```
-------------------------- rockchip BSP packages --------------------------
 Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
 submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> selects a
 feature, while <N> excludes a feature.  Press <Esc><Esc> to exit, <?> for
 Help, </> for Search.  Legend: [*] feature is selected  [ ] feature is
+----^(-)-----------------------------------------------------------------+
|     [ ]   Simple iflytek voice process and cloud SDK                    |
|     [*]   Equalizer and DRC process                                     |
|     [*]   alsa plugin ladspa                                            |
|     [*]   stress test tools                                             |
|     [ ]   rockchip modules                                              |
|     [*]   broadcom(ampak) bsa server and app                            |
|             wifi/bt chip support (AP6255)  --->                         |
|     [ ]   broadcom(cypress) bsa server and app                          |
|     [ ]   pm suspend api & demo                                         |
|     [ ]   realtek simple config                                         |
|     [ ]   Rockchip recovery for linux                                   |
|     [ ]   Rockchip OTA update for linux                                 |
|     [ ]   modeset for Linux                                             |
|     [ ]   rkjpeg for Linux                                              |
|     [ ]   hw jpeg test for Linux                                        |
+----v(+)-----------------------------------------------------------------+
```

2.1、正基模组
选择‘broadcom(ampak) bsa server and app’
进入‘wifi/bt chip support (AP6255) --->’选择芯片型号



```
-------------------- wifi/bt chip support --------------------
 Use the arrow keys to navigate this window or press the
 hotkey of the item you wish to select followed by the <SPACE
 BAR>. Press <?> for additional information about this
+------------------------------------------------------------+
|                        (X) AP6255                          |
|                        ( ) AP6212A1                        |
|                                                            |
```

2.2、海华模组
选择‘broadcom(cypress) bsa server and app’
进入‘wifi/bt chip support (AW-CM256) --->’选择芯片型号



```
-------------------- wifi/bt chip support --------------------
 Use the arrow keys to navigate this window or press the
 hotkey of the item you wish to select followed by the <SPACE
 BAR>. Press <?> for additional information about this
+------------------------------------------------------------+
|                        (X) AW-CM256                        |
|                        ( ) AW-NB197                        |
```

2.3、退出选项框，选择 Yes，make savedefconfig 保存配置；

2.4、编译、打包

正基模组：make broadcom_bsa-rebuild
海华模组：make cypress_bsa-rebuild

```
make rkwifibt-rebuild
make
./mkfirmware.sh
```

3.1.2 源码目录

基于 broadcom 的海华/正基模组支持 BSA 协议栈，而 BSA 协议栈是 broadcom 公司开发的蓝牙协议栈，类似 BLUEZ，开发人员可以基于它开发各种蓝牙 APP，并且提供丰富的 app demo：
注：此部分仅简单介绍，由于 BSA 协议栈是不开源的，我们也没有代码，所以如果需要详细开发文档，请咨询模组厂，他们会提供完善的技术支持；如果联系不上，可以联系我们；当然如果不想使用 BSA，也可以使用开源的 blueZ，BlueZ 是通用的所以可以直接参考：RK3308_RTL8723DS_WIFI_BT_说明文档_V1.20.pdf。

1、正基模组：

脚本、编译 mk 文件：buildroot/package/rockchip/broadcom_bsa
源码：external/broadcom_bsa

App demo：external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux

2、海华模组：

脚本、编译 mk 文件：buildroot/package/rockchip/cypress_bsa

源码：external/bluetooth_bsa

App demo：external/ bluetooth_bsa /3rdparty/embedded/bsa_examples/linux

*Bsa Application Demo List

app_3d -- This application is used to connect 3D stereo glasses
app_ag -- Audio Gateway (AG) application. AG applications (usually located in cellular phones) are used to connect to mono headsets.
app_av -- This application is used to stream music from an AV Source (AVS) profile.
app_avk -- This application is used to receive a music steam from an AV sink (AVK) profile.
app_ble -- This application shows how to use BLE client/server module BSAs.
app_ble_blp -- This application is sample code for the BLE blood-pressure collector.
app_ble_cscc -- This application is sample code for the BLE cycling speed and cadence collector.
app_ble_hrc -- This application is sample code for the BLE heart rate controller.
app_ble_htp -- This application is sample code for the BLE health thermometer collector.
app_ble_pm -- This application is sample code for the BLE proximity monitor.
app_ble_rscc -- This application is sample code for the BLE running speed and cadence collector.
app_ble_wifi -- This sample application makes it easier to set up a Wi-Fi Direct connection between peers by exchanging Wi-Fi Direct information over BLE.
app_dg -- Data Gateway (DG). Two Bluetooth devices can connect using this application to exchange data via serial communications port emulation.
app_fm -- This application shows how to use the FM radio APIs.
app_ftc -- FTP client (FTC). FTP clients can connect and access (browse/read/write) files located on the FTP server.
app_fts -- File Transfer Server (FTS). The FTS can connect and access (browse/read/write) files located on the FTP server.
app_hd -- Demonstrates Bluetooth HID device and remote control functionality.
app_headless -- Shows how the Headless mode is controlled.
app_hd -- Connects Human Interface Devices (HIDs) such as a keyboard, mouse, or remote control.
app_hh -- Connects Human Interface Devices (HIDs) such as a keyboards, mice, or remote controls.
app_hl -- This application is used to connect the health devices.
app_hs -- Headset (HS) application. HS applications are used to connect to cellular phones which run AG applications.
app_manager -- The main regular application. Governs security and device management. It must always be running.
app_mce -- This application provides the Bluetooth MAP client (Message Client Equipment [MCE]) function.

app_opc -- OP client (OPC). OP clients can exchange (send/receive) files with the OPS.
app_ops -- Object Push (OP) server (OPS). OP clients can exchange (send/receive) files with the OP server.
app_pan -- This application provides the test framework for Bluetooth PAN functionality.
app_pbc -- This application provides the Bluetooth PBAP client (PBC) function.
app_pbs -- Phone Book (PB) server (PBS). PB clients can connect to the PBS to download/upload phone books.
app_sac -- This application provides the Bluetooth SIM Access Profile (SAP) client function.
app_sc -- This application provides the Bluetooth SAP client (SC) function.
app_switch -- This application demonstrates the sequence of operations necessary to implement role switching between AV/AG and AVK/HF.
app_tm -- This application describes Test Module (TM) APIs.

3.1.3 Application Demo 运行示例

1、上电：

```
echo 0 > /sys/class/rfkill/rfkill0/state
sleep 1
echo 1 > /sys/class/rfkill/rfkill0/state
sleep 1
```

2、所有的蓝牙相关的进程都要在/data/bsa/config 目录执行

```
mkdir -p /data/bsa/config
cd /data/bsa/config
```

3、启动 server 进程

```
bsa_server -r 12 -p /system/etc/firmware/XXXX.hcd -d /dev/ttyS4 -b /data/btsnoop.log > /data/bsa_log &
```

XXXX.hcd：各芯片对应的 firmware

AP6255：	BCM4345C0.hcd
AP6212A1 ：	bcm43438a1.hcd
AWCM256：	BCM4345C0.hcd
AWNB197：	BCM4343A1.hcd

4、启动管理进程：（运行到此步骤，才能在手机蓝牙列表发现并连接设备）

```
app_manager &
```

5、启动你想要运行的客户端（app_av、app_avk 等）：

```
app_xxxx &
```

3.1.4 设置蓝牙设备名和 mac 地址

1、文件：app_manager.c

2、接口：app_mgr_config

3、现有例子：

```
#ifdef DUEROS
        app_mgr_get_bt_config((char *)app_xml_config.bd_addr, BD_ADDR_LEN);
        sprintf((char *)app_xml_config.name, "%s%02X%02X", "DuerOS_",
            app_xml_config.bd_addr[4], app_xml_config.bd_addr[5]);
        APP_DEBUG1("device name: %s", app_xml_config.name);
#else
        bdcpy(app_xml_config.bd_addr, local_bd_addr);
        /* let's use a random number for the last two bytes of the BdAddr */
        gettimeofday(&tv, NULL);
        rand_seed = tv.tv_sec * tv.tv_usec * getpid();
        app_xml_config.bd_addr[4] = rand_r(&rand_seed);
        app_xml_config.bd_addr[5] = rand_r(&rand_seed);
        sprintf((char *)app_xml_config.name, "My BSA Bluetooth Device %02x%02x",
            app_xml_config.bd_addr[4], app_xml_config.bd_addr[5]);
#endif
```

说明：

1）、app_xml_config.name：存储蓝牙名称

2）、app_xml_config.bd_addr：存储 mac 地址

3）、app_mgr_get_bt_config：从蓝牙芯片获取 mac 地址

4）、bdcpy(app_xml_config.bd_addr, local_bd_addr)：写一个固定的 mac 地址，后两位使用随机值

4、设置方式：

在 app_xml_config.name 和 app_xml_config.bd_addr 中写入自己想要的蓝牙名称和 mac 地址

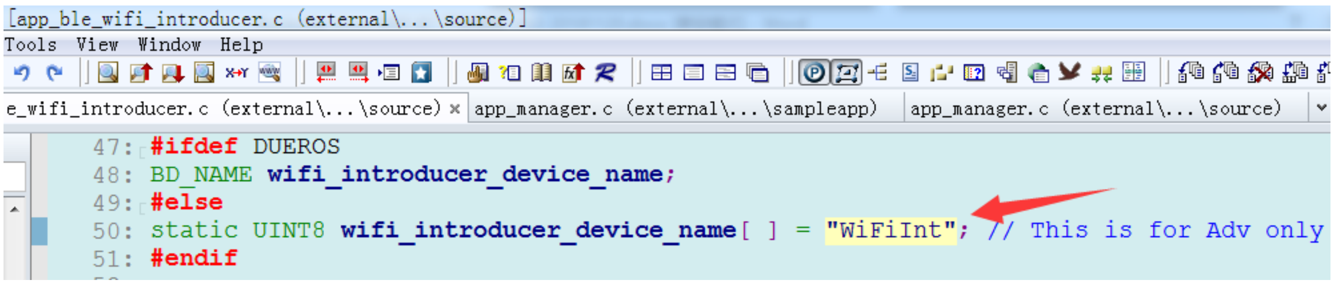
3.1.5 Ble 配网

1、执行 bsa_ble_wifi_introducer.sh start

2、参考本文档中的章节 2.2.2 蓝牙配网，配合 apk 进行配网

3、代码路径：external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux/app_ble_wifi_introducer

4、修改配网时的设备名：


[app_ble_wifi_introducer.c (external\...\source)]
Tools View Window Help
e_wifi_introducer.c (external\...\source)
app_manager.c (external\...\sampleapp)
app_manager.c (external\...\source)

```
#ifdef DUEROS
BD_NAME wifi_introducer_device_name;
#else
static UINT8 wifi_introducer_device_name[ ] = "WiFiInt"; // This is for Adv only
#endif
```

3.1.6 A2DP SINK

1、执行：bsa_bt_sink.sh start

2、打开手机蓝牙会显示出 My BSA Bluetooth Device XXXX，点击连接即可实现播放音乐的功能；

注意：如果连接之后没有声音，请检查声卡的配置。

3、代码路径：external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux/app_avk

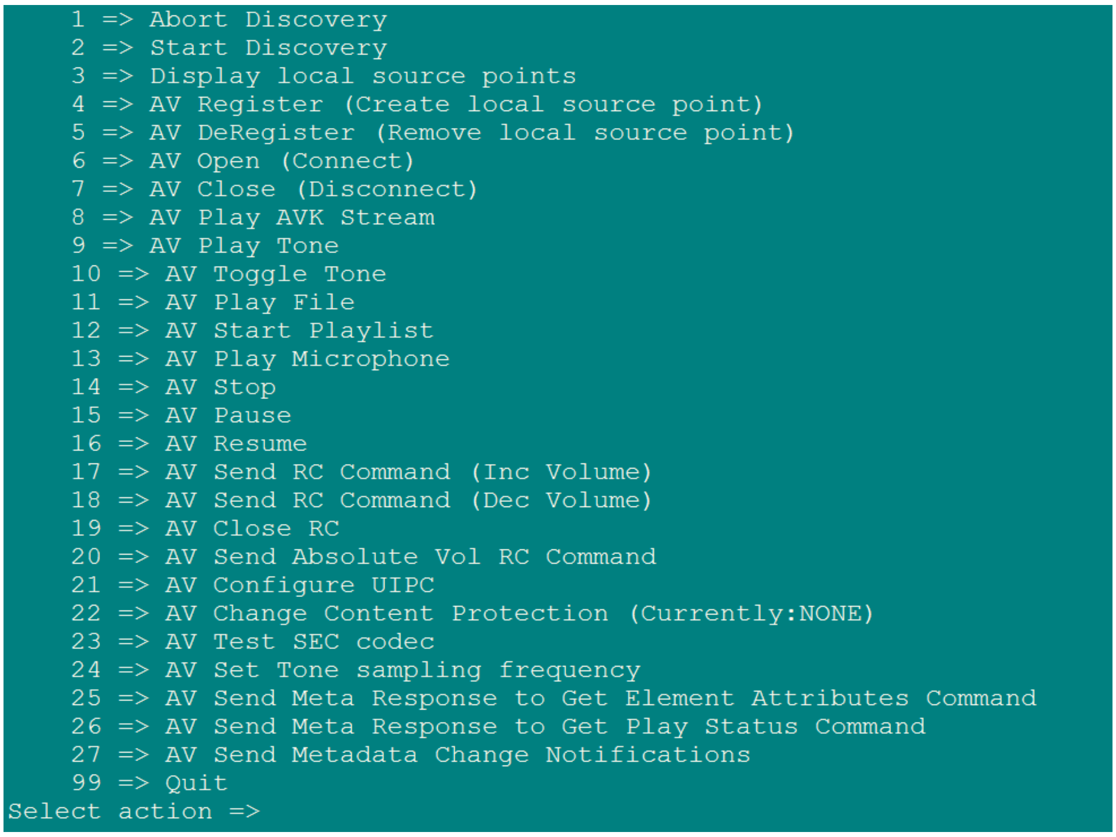
3.1.7 A2DP SRC

1、代码路径：external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux/app_av

2、运行 bsa_bt_source.sh start

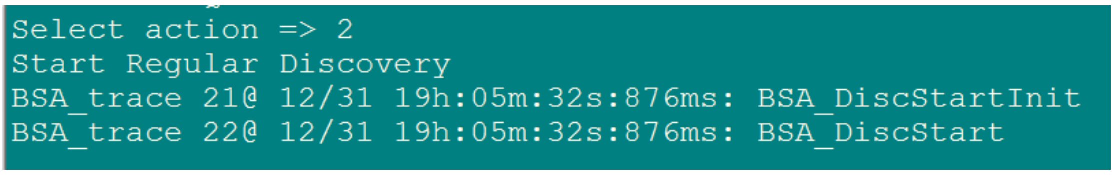
3、cd data/bsa/config/

4、运行 app_av，显示如下 app_av menu

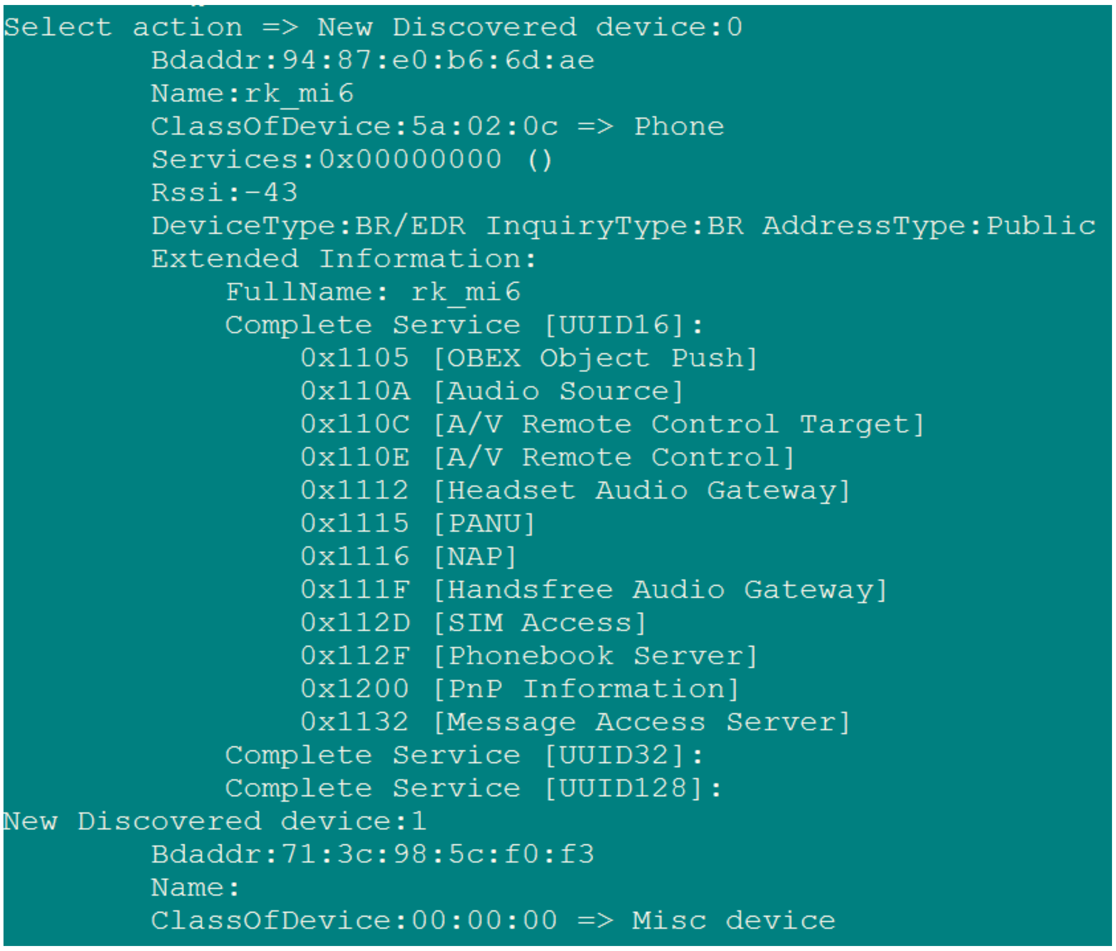


```
    1 => Abort Discovery
    2 => Start Discovery
    3 => Display local source points
    4 => AV Register (Create local source point)
    5 => AV DeRegister (Remove local source point)
    6 => AV Open (Connect)
    7 => AV Close (Disconnect)
    8 => AV Play AVK Stream
    9 => AV Play Tone
    10 => AV Toggle Tone
    11 => AV Play File
    12 => AV Start Playlist
    13 => AV Play Microphone
    14 => AV Stop
    15 => AV Pause
    16 => AV Resume
    17 => AV Send RC Command (Inc Volume)
    18 => AV Send RC Command (Dec Volume)
    19 => AV Close RC
    20 => AV Send Absolute Vol RC Command
    21 => AV Configure UIPC
    22 => AV Change Content Protection (Currently:NONE)
    23 => AV Test SEC codec
    24 => AV Set Tone sampling frequency
    25 => AV Send Meta Response to Get Element Attributes Command
    26 => AV Send Meta Response to Get Play Status Command
    27 => AV Send Metadata Change Notifications
    99 => Quit
Select action =>
```

5、输入 2，开始发现蓝牙设备



```
Select action => 2
Start Regular Discovery
BSA_trace 21@ 12/31 19h:05m:32s:876ms: BSA_DiscStartInit
BSA_trace 22@ 12/31 19h:05m:32s:876ms: BSA_DiscStart
```



```
Select action => New Discovered device:0
        Bdaddr:94:87:e0:b6:6d:ae
        Name:rk_mi6
        ClassOfDevice:5a:02:0c => Phone
        Services:0x00000000 ()
        Rssi:-43
        DeviceType:BR/EDR InquiryType:BR AddressType:Public
        Extended Information:
            FullName: rk_mi6
            Complete Service [UUID16]:
                0x1105 [OBEX Object Push]
                0x110A [Audio Source]
                0x110C [A/V Remote Control Target]
                0x110E [A/V Remote Control]
                0x1112 [Headset Audio Gateway]
                0x1115 [PANU]
                0x1116 [NAP]
                0x111F [Handsfree Audio Gateway]
                0x112D [SIM Access]
                0x112F [Phonebook Server]
                0x1200 [PnP Information]
                0x1132 [Message Access Server]
            Complete Service [UUID32]:
            Complete Service [UUID128]:
New Discovered device:1
        Bdaddr:71:3c:98:5c:f0:f3
        Name:
        ClassOfDevice:00:00:00 => Misc device
```

6、输入 6 => AV Open (Connect)

```
Bluetooth AV Open menu:
    0 Device from XML database (already paired)
    1 Device found in last discovery
Select source =>
```

7、输入 1，显示最近一次发现的设备列表（如果已经连接设备，直接选择 0）

```
Select source => 1
Dev:0
        Bdaddr:94:87:e0:b6:6d:ae
        Name:rk_mi6
        ClassOfDevice:5a:02:0c => Phone
        Rssi:-37
Dev:1
        Bdaddr:20:a6:0c:2c:69:98
        Name:hertz's tablet
        ClassOfDevice:5a:02:0c => Phone
        Rssi:-42
Dev:2
        Bdaddr:71:3c:98:5c:f0:f3
        Name:
        ClassOfDevice:00:00:00 => Misc device
        Rssi:-52
Dev:3
        Bdaddr:b0:f1:a3:00:3d:f2
        Name:Fiil Wireless
        ClassOfDevice:24:04:18 => Audio/Video
        Rssi:-53
        VidSrc:1 Vid:0x0234 Pid:0x0002 Version:0x0112
```

8、输入想要连接的设备对应的 Dev 序号，此处输入 3，选择 Dev:3 Fiil Wireless 进行配对连接

```
Select device => 3
Connecting to AV device
BSA_trace 25@ 12/31 19h:11m:25s:362ms: BSA_AvOpenInit
BSA_trace 26@ 12/31 19h:11m:25s:362ms: BSA_AvOpen
```

9、play wav file

输入 11，显示/data/bsa/config/test_files/av 目录中可播放 wav 文件列表；可以将想要播放的 wav 文件 push 到该目录，也可以在 buildroot/package/rockchip/broadcom_bsa/broadcom_bsa.mk 中进行拷贝。选择想要播放的文件对应的序列号，播放相应文件

```
11
Play list:
    0 : ./test_files/av/8k16bpsStereo.wav
        codec(PCM) ch(2) bits(16) rate(8000)
    1 : ./test_files/av/8k8bpsMono.wav
        codec(PCM) ch(1) bits(8) rate(8000)
Select file => 1
1 :./test_files/av/8k8bpsMono.wav
    codec(PCM) ch(1) bits(8) rate(8000)
```

10、当前有 wav 文件正在播放时，需要输入 14 => AV Stop 停止播放，才能重复步骤 9 播放新的文件。
还可以通过以下命令控制播放：

14 => AV Stop
15 => AV Pause
16 => AV Resume
17 => AV Send RC Command (Inc Volume)
18 => AV Send RC Command (Dec Volume)

3.1.8 HFP

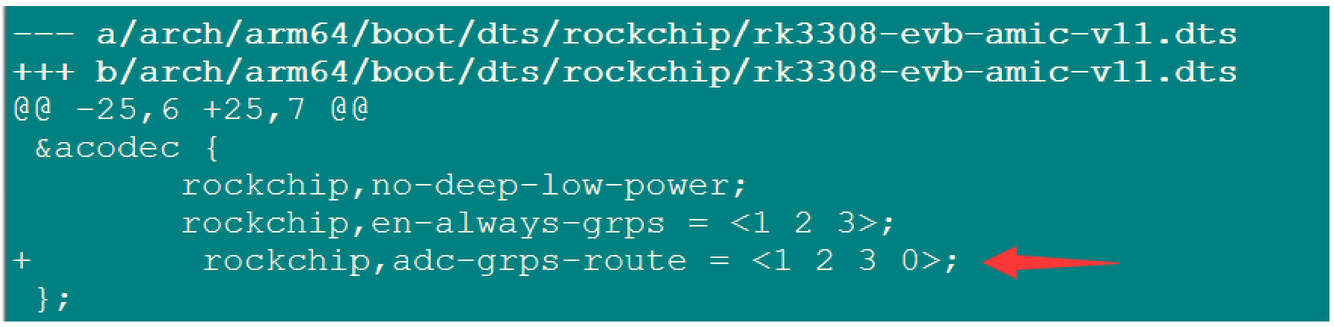
1、只针对正基模组

2、external/broadcom_bsa 必须包含如下提交：

```
commit 756a6c3d984a085b3a5aaf5a9691a3f39dcc7c8c
Author: ctf <ctf@rock-chips.com>
Date:    Wed Sep 26 18:00:28 2018 +0800

    bluetooth: broadcom_bsa: integrate arecord and aplay into HFP.

    Change-Id: Ic3fef580a24eaa4126dd6eb53a59adb4e2ed6d01
    Signed-off-by: ctf ctf@rock-chips.com
```

3、代码路径：external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux/app_hs

4、v11：default 0,1channel map loopback，should set the adc-channel map in dts; as follows:
以 rk3308-evb-amic-v11.dts 为例：



```
--- a/arch/arm64/boot/dts/rockchip/rk3308-evb-amic-v11.dts
+++ b/arch/arm64/boot/dts/rockchip/rk3308-evb-amic-v11.dts
@@ -25,6 +25,7 @@
 &acodec {
        rockchip,no-deep-low-power;
        rockchip,en-always-grps = <1 2 3>;
+        rockchip,adc-grps-route = <1 2 3 0>;
 };
```

5、添加蓝牙设备节点，并修改 playback 的 period_size 和 buffer_size：
在 buildroot/board/rockchip/rk3308/fs-overlay/etc/ asound.conf 末尾添加如下代码：

```
    pcm.bluetooth {
        type asym
        playback.pcm {
            type plug
            slave {
                pcm "hw:1,0"
                channels 2
                rate 16000
            }
        }
        capture.pcm {
            type plug
            slave {
                pcm "hw:1,0"
            }
        }
    }
```

修改 playback 的 period_size 和 buffer_size：

```
pcm.playback {
    type dmix
    ipc_key 5978293 # must be unique for all dmix plugins!!!!
    ipc_key_add_uid yes
    slave {
        pcm "hw:7,0,0"
        channels 2
-           period_size 1024
-           buffer_size 4096
+           period_size 3072
+           buffer_size 12288
    }
    bindings {
        0 0
        1 1
    }
}
```

6、执行 bsa_bt_hfp.sh start
7、连接 My BSA Bluetooth Device XXXX 设备，即可实现 hfp 双向通话

4 WiFi 的无线唤醒（WoWLAN）

目前 WiFi 支持无线网络包唤醒系统，例如：音响设备正常连接 WiFi 并获取到正确的 IP 地址，则当设备休眠后，我们可以通过无线网络包唤醒系统，唤醒规则：只要是发给这个设备 IP 地址的网络包，都会唤醒系统.

AP6XXX/RTL 模组上层配置：修改 wpa_supplicant.conf 文件，添加如下配置：

```
wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
update_config=1
ap_scan=1
+wowlan_triggers=any //确保修改生效
```

Realtek 模组请打开对应驱动的 Makefile 里面的如下配置：

```
/drivers/net/wireless/rockchip_wlan/rtl8xxx/Makefile
CONFIG_WOWLAN = y
```

CONFIG_GPIO_WAKEUP = y

注意：

1、确保配置 **WIFI_WAKE_HOST: WIFI 唤醒主控的 PIN 脚**

Dts 的 WiFi 配置：WIFI,host_wake_irq = <&gpio0 RK_PA0 GPIO_ACTIVE_HIGH>;

//WIFI_WAKE_HOST GPIO_ACTIVE_HIGH 特别注意：确认下这个 wifi pin 脚跟主控的连接关系，如果中间加了一个反向管就要改成低电平触发

2、休眠前请确保 hostapd 进程关掉，网络唤醒功能要求必须关掉 hostapd 进程；

软件参考 ping 源码即可；

测试方法：设备连上 WiFi 并正常获取到 IP 地址，（echo mem > sys/power/state）进休眠后，手机端下载一个 ping 软件（确保手机或者 PC 连接到同一局域网），然后去 ping 设备的 IP 地址，正常的话，可以看到设备会被唤醒.

问题排查：AP 和 RTL 芯片默认都是高电平触发，假设 WiFi_Wake_Host 脚和主控直连，则设备进入休眠后，pin 脚默认低电平，当有网络包唤醒时，这个脚用示波器可以测得高脉冲进来；所以当设备没有被唤醒时请用示波器测下这个 PIN 脚是否符合上述行为；如果实在无法解决请提供：完整的 dmesg 的内核 log 以及 cat proc/interrupts 的打印输出。

5 WiFi 的 monitor 模式

启用 WiFi 的 monitor 模式:

5.1 Broadcom 芯片

dhd_priv SDK 自带该命令

设置信道：

dhd_priv channel 6 // channal numbers

开 monitor 模式：

dhd_priv monitor 1

再用附件 2 中的 sample code raw 代码就可以读到抓到的包了

关 monitor 模式：

dhd_priv monitor 0

5.2 Realtek 芯片

驱动 Makefile 需要打开：

CONFIG_WIFI_MONITOR = y

```
1. ifconfig wlan0 up
   ifconfig p2p0 down
2. iwconfig wlan0 mode monitor    /*support wext sulution.*/
   or iw dev wlan0 set type monitor    /*support cfg80211 sulution.*/
3. echo "<chan> 0 0" > /proc/net/<rtk_module>/wlan0/monitor    /*<rtk_module> is the
realtek wifi module name, such like rtl8812au, rtl8188eu ..etc */
4. tcpdump -i wlan0 -s 0 -w snf_pkts.pcap  /*capture the sniffer packets and save it as a file
"snf_pkts.pcap"  也可以使用附件 2 中的 sample code raw 代码就可以读到抓到的包了*/
```

6 WiFi RF 指标

请让硬件工程师确认 WiFi 的 RF 指标、频偏是否正常；Realtek 的芯片需要提供信令模式硬件测试报告，找模组厂提供对应芯片的 map 校准文件（校准文件的烧录，参考附录 1）；确保硬件指标合格。

7 WiFi 问题排查

7.1 WiFi 无法识别

首先硬件测量 WIFI_REG_ON/VDDIO VBAT/SDIO_CLK/SDIO_CMD/SDIO_DATA0~SDIO_DATA3 的电压,详细描述如下：

（1）确认 WIFI_CLK 的信号是否常，确认上电后 37.4/24/26M 有正常起振，加载驱动时，SDIO_CLK 有时钟吐出来，识卡 400K 左右，稳定后会达到 DTS 设置的 CLK；

（3）确认 32.768K 方波信号是否正常，峰峰值有要求，必须是 0.7 * VDDIO ~ VDDIO 这个范围内才行。否则会有问题；

（4）确认 WL/WIFI_REG_ON WIFI 上电管脚在打开 WIFI 的时候是否正常被拉高，而且异常的时候是否电平有变化，VBAT 供电是否有波动；其中 DTS 部分的 WIFI_REG_ON（sdio-pwrseq）的脚配置对应的解析代码操作参见：drivers/mmc/core/pwrseq_simple.c

```
解析 reset-gpio：
    mmc_pwrseq_simple_alloc
上下电：
    static struct mmc_pwrseq_ops mmc_pwrseq_simple_ops = {
        .pre_power_on = mmc_pwrseq_simple_pre_power_on, // 拉低 WiFi_REG_ON
        .post_power_on = mmc_pwrseq_simple_post_power_on, // 拉高 WiFi_REG_ON
        .power_off = mmc_pwrseq_simple_power_off, // 拉低 WiFi_REG_ON
        .free = mmc_pwrseq_simple_free,
    };
```

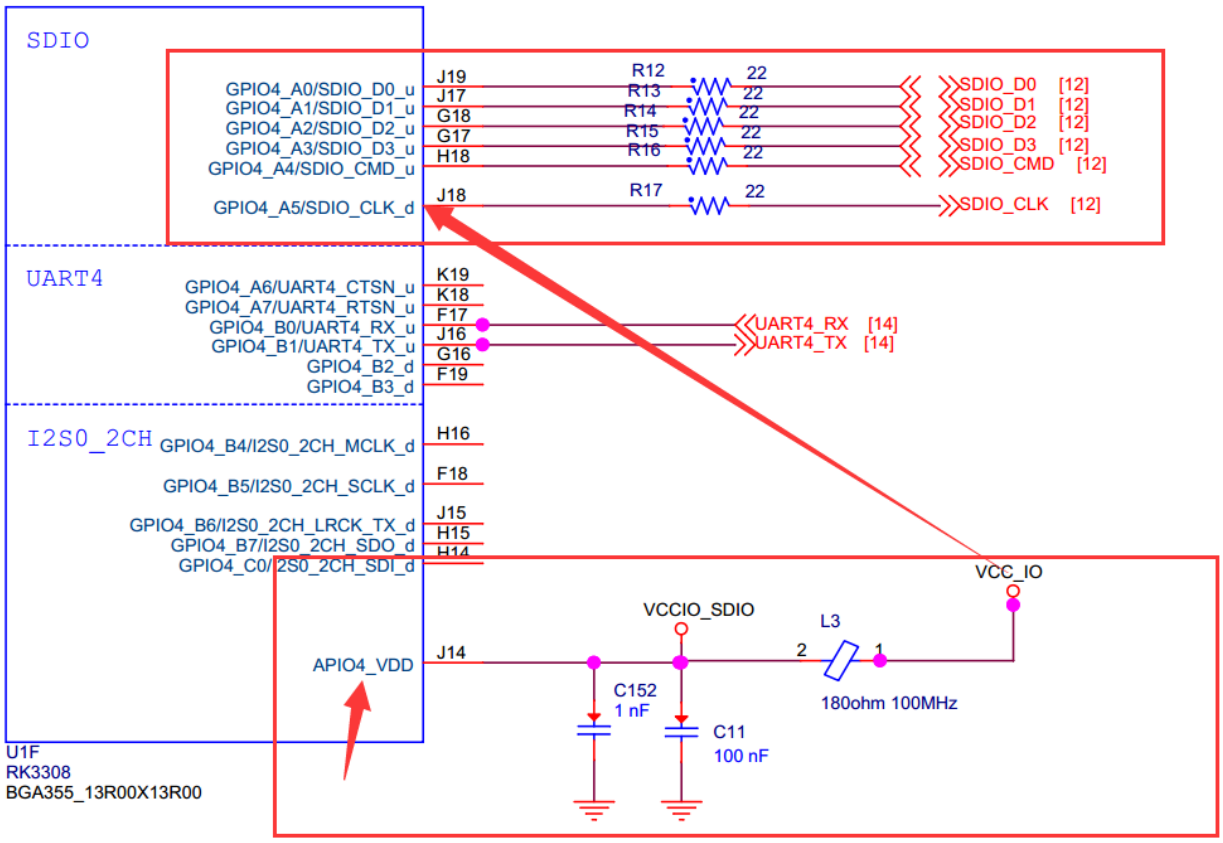
通过示波器可以开到：在进到内核后 WiFi 初始化时 WiFi_REG_ON 会被先拉低再拉高，如果观测不到，请在上面代码处加些 debug 确认；

（5）确认晶体部分外围器件物料是否有漏焊或者焊错；

（6）确认 sdio 的 4 根 data 走线是否有问题，是否有干扰

（7）SDIO data 传输异常，检查 sdio wifi 硬件使用的物料是否符合标准，比如电容，电阻有没有错接或者遗漏；

(8) &io-domains，如下图 VCCIO/VCCIO_SDIO 给 SDIO_DX/CMD/CLK 供电，则需要跟软件配置一致：比如 APIO4 对应于 vccio4-supply = <&vccio_sdio>; //这里谁给 Apio4 供电就填谁，由于 VCC_IO 和 vccio_sdio 是同一路，也可以填 vcc_io.



如果 WiFi 正常识别会在 log 中得到如下类似打印：

mmcX: new ultra high speed SDR104 SDIO card at address 0001

或

mmcX: new high speed SDIO card at address 0001

7.2 WiFi 无法连接路由器

1、请先确保如下两个进程是否有跑起来：

```
wpa_supplicant -B -i wlan0 -c /data/cfg/wpa_supplicant.conf
/sbin/dhcpcd -f /etc/dhcpcd.conf
```

2、wpa_cli scan 、wpa_scan_r 命令去扫描热点，如果执行失败可以多次执行，确认是否能扫描到 wifi：正常的话会有如下信息：在下面的信息中找下是否有的 WiFi，如果扫不到，或者扫到的 wifi 跟你们手机或其他设备的数量差距很大，或者信号强度很弱（signal level），请检测 WiFi 天线是否达标，频偏是否符合要求等。

```
Selected interface 'wlan0'
bssid / frequency / signal level / flags / ssid
dc:ef:09:a7:77:52       5765    -33     [WPA2-PSK-CCMP][ESS]      NETGEAR75-5G
10:be:f5:1d:a3:76       5220    -53     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]      D-Link_DIR-880L_5GHz
d0:ee:07:1c:2d:18       5745    -54     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]        ROCKROOM_5G
a0:63:91:2e:16:fa       5765    -56     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]      hjk_5GEXT
30:fc:68:bb:09:bb       5745    -67     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]        TP-LINK_5G_09B9
64:09:80:0a:13:b1       5805    -59     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]      diaozhatian
74:7d:24:61:39:d0       5180    -48     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]      @PHICOMM_CE_5G
… …
```

7.3 WiFi 连接不稳定

参考 WIFI RF 指标章节

7.4 WiFi 其他问题

请提供 kernel dmesg 和 wpa_supplicant 的 log（方法：在启动 wpa_supplicant 程序的地方加上 debug 选项，如：wpa_supplicant -B -i wlan0 -c /data/cfg/wpa_supplicant.conf **-f debug.txt //将 log 重定向到 debug.txt** 文件里面）。

附录 1: Realtek Map 校准文件的加载

KO 方式：为了方便调试，可以改为 ko 方式加载 map 文件，下面以 8723DS 为例，其他类似：

（1）、更改内核配置：

```
CONFIG_RTL8723DS=m
```

修改你们认为放便调试目录：

```
+EXTRA_CFLAGS += -DEFUSE_MAP_PATH=\"/data/wifi_efuse_8723ds.map\"
+++ b/drivers/net/wireless/rockchip_wlan/rtl8723ds/Makefile
@@ -797,7 +797,7 @@ EXTRA_CFLAGS += -DEFUSE_MAP_PATH=\"$(USER_EFUSE_MAP_PATH)\"
 else ifeq ($(MODULE_NAME), 8189es)
 EXTRA_CFLAGS += -DEFUSE_MAP_PATH=\"/system/etc/wifi/wifi_efuse_8189e.map\"
 else ifeq ($(MODULE_NAME), 8723ds)
-EXTRA_CFLAGS += -DEFUSE_MAP_PATH=\"/vendor/etc/firmware/wifi_efuse_8723ds.map\"
+EXTRA_CFLAGS += -DEFUSE_MAP_PATH=\"/data/wifi_efuse_8723ds.map\"
 else
 EXTRA_CFLAGS +=
-DEFUSE_MAP_PATH=\"/system/etc/wifi/wifi_efuse_$(MODULE_NAME).map\"
```

重新编译生成 ko
drivers/net/wireless/rockchip_wlan/rtl8723ds/8723ds.ko
更新内核

（2）、放入必要文件

把 wifi_efuse_8723ds.map 文件 push 到上面修改的目录：+EXTRA_CFLAGS += -DEFUSE_MAP_PATH=\"/data/wifi_efuse_8723ds.map\"

把 8723ds.ko push 到 data 或者其他认为方便的目录

```
adb push wifi_efuse_8723ds.map /data/
adb push 8723ds.ko /data
```

（3）、开机后执行：

```
insmod /data/8723ds.ko //注意一定要跟你上面 push 的目录一致
```

log 中显示：

```
[   29.002020] RTW: efuse file:/oem/wifi_efuse_8723ds.map, 0x200 byte content read
[   29.002065] RTW: EFUSE FILE
[   29.002098] RTW: 0x000: 29 81 00 7C  01 88 07 00  A0 04 EC 35  12 C0 A3 D8
[   29.002289] RTW: 0x010: 28 28 28 28  28 28 28 28  28 28 28 02  FF FF FF FF
[   29.002477] RTW: 0x020: FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
```

Buildin 方式：
有时为了加快开机联网速度，希望采用 buildin，则可以通过如下方式实现：
drivers/net/wireless/rockchip_wlan/rtl8189fs/core/efuse/rtw_efuse.c
找到该 rtw_read_efuse_from_file 函数：

```
{
... ...
map = rtw_vmalloc(map_size); //申请内存

for (i = 0 ; i < map_size ; i++) {
    ... ... //解析 map 文件里面的内容，赋值为 map 指针
}

DBG_871X_LEVEL(_drv_always_,"efuse file:%s, 0x%03xbyte content read\n",path,i);
//先用 ko 的方式加载，在这里把 map 指针的内容打印出来，然后把内容做成数组，直接赋值给 map 即可，就省掉解析动作了。
_rtw_memcpy(buf, map, map_size); //这里是最终的赋值操作。
... ...
}
```

附件 2: Wifi Monitor 模式抓包代码示例

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <linux/socket.h>
#include <stdio.h>
#include <string.h>
#include <netpacket/packet.h>
#include <net/if.h>
#include <netinet/in.h>

#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>

//unsigned short protocol = 0x888e;
unsigned short protocol = 0x0003;
#define NAME "wlan0"
int cc = 0;

int main(int argc, char ** argv)
{
    struct ifreq ifr;
```

```
	struct sockaddr_ll ll;
	int fd;


	fd = socket(PF_PACKET, SOCK_RAW, htons(protocol));
	printf("fd = %d \n", fd);
	if(argv[1])
		printf("name = %s \n", argv[1]);

	memset(&ifr, 0, sizeof(ifr));
	if(argv[1])
		strlcpy(ifr.ifr_name, argv[1], sizeof(ifr.ifr_name));
	else
		strlcpy(ifr.ifr_name, NAME, sizeof(ifr.ifr_name));

	printf("ifr.ifr_name = %s \n", ifr.ifr_name);

	if (ioctl(fd, SIOCGIFINDEX, &ifr) < 0) {
		close(fd);
		printf("get infr fail\n");
		return -1;
	}

	memset(&ll, 0, sizeof(ll));
	ll.sll_family = PF_PACKET;
	ll.sll_ifindex = ifr.ifr_ifindex;
	ll.sll_protocol = htons(protocol);
	if (bind(fd, (struct sockaddr *) &ll, sizeof(ll)) < 0) {
		printf("bind fail \n");
		close(fd);
		return -1;
	}

	while(1) {
		unsigned char buf[2300];
		int res;
		socklen_t fromlen;
		int i = 0;

		memset(&ll, 0, sizeof(ll));
		fromlen = sizeof(ll);
		res = recvfrom(fd, buf, sizeof(buf), 0, (struct sockaddr *) &ll,
				&fromlen);
		if (res < 0) {
			printf("res < 0\n");
			return -1;
```

```
            } else {
                cc++;
                printf("%04d(%03d) - ", cc, res);
                //for(i = 0; i < res && i < 8; i++)
            for(i = 0; i < res; i++)
                    printf("%02x ", buf[i]);
                printf("\n");
            }
        }

        close(fd);

        return 0;

    }
```