Rockchip Linux Wi-Fi/BT Developer Guide

ID: RK-KF-YF-381

Release Version: V6.0.2

Release Date: 2021-01-15

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document is going to introduce Wi-Fi/BT development, porting and debugging on Rockchip Linux platform.

Product Version

Chipset	**Kernel Version**
RK3399/3326/3288/3308/1109/1126/PX30	Linux 4.4/4.19

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	**Author**	**Date**	**Change Description**
V6.0.1	xy	2020-08-25	Added new module porting instructions, building instructions, RF test examples, P2P bridge function, more detailed troubleshooting instructions, etc.
V6.0.2	Ruby Zhang	2021-01-15	Update expression of some content.

Contents

1. Wi-Fi/BT Configuration

1.1 DTS Configuration

1.1.1 Wi-Fi Configuration

The following items are included in Wi-Fi/BT hardware pin configurations:

Remember to configure according to the schematic, and make sure that the dts/dtsi contains the following nodes:

```
/* WIFI_REG_ON: power pin of Wi-Fi */
sdio_pwrseq: sdio-pwrseq {
    /* Special attention: the following clock attribute is used to enabale the
32.768k of RK8XX PMU, please fill in according to the used model actually,
otherwise SDIO/Wi-Fi cannot be used in, it has been moved to the wireless-wlan
node in the latest SDK , as long as the configuration is not repeated*/
    //clocks = <&rk809 1>;
    //clock-names = "clk_wifi";
    compatible = "mmc-pwrseq-simple";
    pinctrl-names = "default";
    pinctrl-0 = <&wifi_enable_h>;
/* The level state here is just the opposite of the enable state. For example, if
REG_ON is high and effective, then it is LOW; if REG_ON is low and effective,
then it is HIGH */
    reset-gpios = <&gpio0 RK_PA2 GPIO_ACTIVE_LOW>;
};
&pinctrl {
    sdio-pwrseq {
        wifi_enable_h: wifi-enable-h {
            rockchip,pins =
            /* Corresponds to the WIFI_REG_ON above */
            <0 RK_PA2 RK_FUNC_GPIO &pcfg_pull_none>;
        };
    };
};

/* Open the SDIO interface */
&sdio {
    max-frequency = <150000000>; /* The maximum frequency of the sdio
interface, adjustable */
    bus-width = <4>; /* 4-wire mode */
    sd-uhs-sdr104; /* Support SDIO3.0 */
    ……
    status = "okay";
};

/* WIFI_WAKE_HOST: the pin of Wi-Fi to wake up controller*/
wireless-wlan {
    compatible = "wlan-platdata";
    rockchip,grf = <&grf>;
    /* Note: The clock attribute below is the 32.768k used to enable PMU, please
fill in according to the actual model used, otherwise it cannot be used in
SDIO/WiFi */
    //rockchip,grf = <&grf>;
    //clocks = <&rk809 1>;
    /* Azurewave/AMPAK modules are compatible without modifying this name,
Realtek's needs to fill in accordingly */
    wifi_chip_type = "ap6255";
    /* Pay attention to WIFI_WAKE_HOST GPIO_ACTIVE_HIGH: confirm the connection
relationship between this wifi pin and the controller, it is HIGH when directly
connected, if there is a reverse tube in the middle, it must be changed to low
level trigger */
    WIFI,host_wake_irq = <&gpio0 RK_PA0 GPIO_ACTIVE_HIGH>;
    /* Note that the Wi-Fi with USB interface needs to add this configuration,
which corresponds to WIFI_REG_ON PIN, and does not require nodes such as
sdio_pwrseq/sdio */
    //WIFI,poweren_gpio = <&gpio0 RK_PA2 GPIO_ACTIVE_HIGH>;
    status = "okay";
};

/* pinctrl configuration */
wireless-wlan {
    /omit-if-no-ref/
    wifi_wake_host: wifi-wake-host {
        /* Note that the wake up pin of regular Wi-Fi is triggered by a high
level, so it must be configured as a pull-down by default. If the customer’s
hardware design is reversed, it must be changed to a pull-up. In short, it must
be initialized with the opposite state with trigger circuit.*/
        rockchip,pins = <0 RK_PA0 0 &pcfg_pull_down>;
    };
};
```

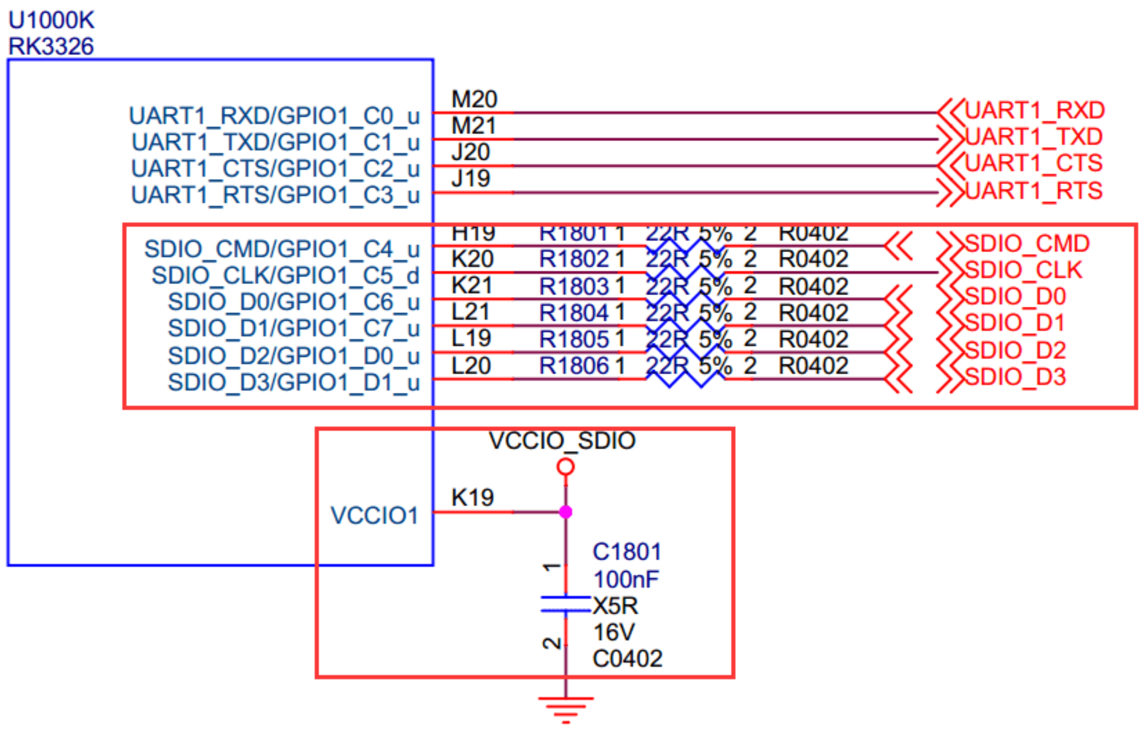
1.1.2 Bluetooth Configuration

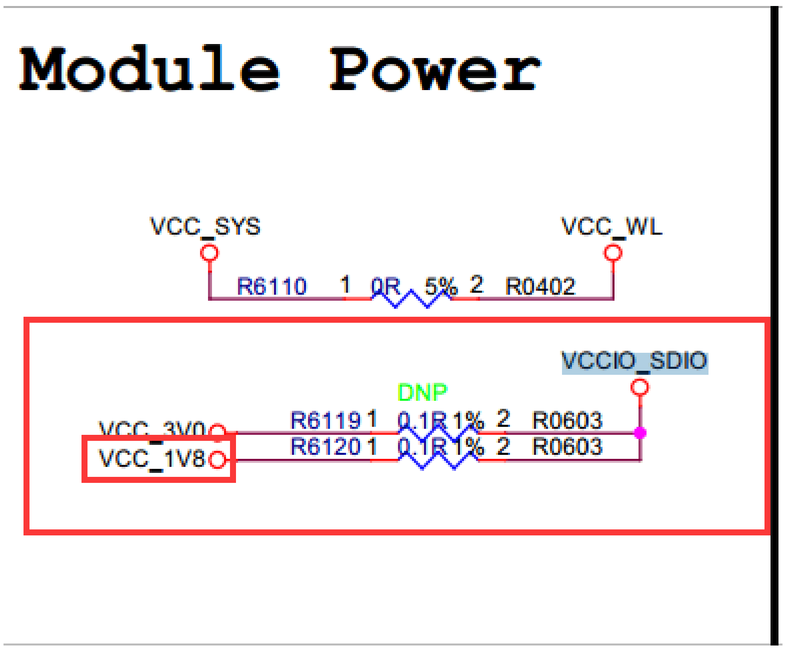
The following UART related items should be configured as the corresponding PIN of actual used UART port. Note that RTS/CTS PIN must be connected according to SDK design. So many abnormalities reported by customers are all because these two pins are not connected, causing initialization abnormal:

```
wireless-bluetooth {
    compatible = "bluetooth-platdata";
    /* It is used to configure the rts pin of uart used by Bluetooth */
    uart_rts_gpios = <&gpio4 RK_PA7 GPIO_ACTIVE_LOW>;
    pinctrl-names = "default", "rts_gpio";
    pinctrl-0 = <&uart4_rts>;
    pinctrl-1 = <&uart4_rts_gpio>;
    /* BT_REG_ON is Bluetooth power switch */
    BT,power_gpio = <&gpio4 RK_PB3 GPIO_ACTIVE_HIGH>;
    /* BT_WAKE_HOST */
    BT,wake_host_irq = <&gpio4 RK_PB4 GPIO_ACTIVE_HIGH>;
    status = "okay";
};
/* Open the corresponding UART configuration, note that the name of each platform
of uart4_xfer/uart4_ctsn may be different, you need to find the corresponding
word in the dts of the chip */
&uart4 {
    pinctrl-names = "default";
    /* For example, uart4_ctsn is named uart4_cts in some platforms */
    pinctrl-0 = <&uart4_xfer &uart4_ctsn>;
    status = "okay";
};
```

1.1.3 IO Power Domain Configuration

Note: The external power such as VCCIO_SDIO (3.3/1.8V) will supply power to the IO of chip and the IO of Wi-Fi module at the same time to ensure the voltage matching between them, and the IO of the controller needs to set the power domain register to match the external power supply voltage, so a specific configuration is required in the dts to tell the driver that the IO of the module supporting SDIO3.0 must be connected to 1.8V, SDIO2.0 connect to 3.3V or 1.8V optionally, but remember to match, in other words , 1.8V power supply cannot be configured with 3.3V, and vice versa.





Check the above schematic diagram and find the sdio interface part corresponding to Wi-Fi. In the figure, there is a mark VCCIOX (**may be with the name of APIOX or others in some chipset**). For example, it is VCCIO1, which is powered by VCCIO_SDIO, check VCCIO_SDIO is connected network is **3.3v** or **1.8v**, you will see that the VCCIO_SDIO in the above figure is powered by vcc_1v8, and the corresponding dts/dtsi configuration is as follows:

```
&io_domains {
    /* vccio1 references the voltage of vccio_sdio */
    vccio1-supply = <&vccio_sdio>;
};

/* vccio_sdio references vcc_1v8 */
vccio_sdio: vcc_1v8: vcc-1v8 {
    compatible = "regulator-fixed";
    regulator-name = "vcc_1v8";
    regulator-always-on;
    regulator-boot-on;
    /* vcc_1v8 with power supply of 1.8v */
    regulator-min-microvolt = <1800000>;
    regulator-max-microvolt = <1800000>;
    vin-supply = <&vcc_io>;
};
```

The above configuration should be matched respectively. If the hardware is 3.3v, modify it according to the matched relationship, don't to mismatch.

1.1.4 The 32.768K Configuration

Modules of Azurewave/AMPAK should be supplied with external 32.768k, while Realtek's modules are packaged internally, so only COB chips will be supplied externally.

There are two cases which Wi-Fi modules need to provide this frequency externally:

1. From the schematic diagram, you can see that PMU of RK8XX model will provide 32k for Wi-Fi. Generally, the PMU turns on 32k by default. If it is not turned on, you need to add the following configuration:

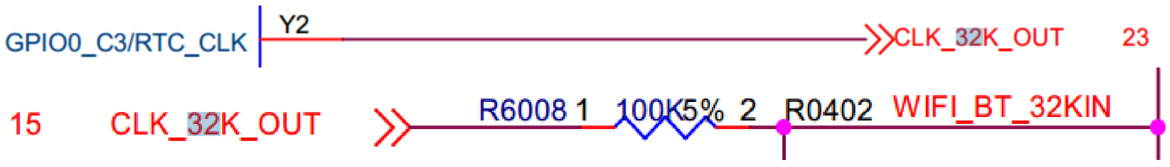
```
wireless-wlan {
        compatible = "wlan-platdata";
        rockchip,grf = <&grf>;
        /* rk809 must be changed to the actual model used */
+         clocks = <&rk809 1>;
+         clock-names = "clk_wifi";
};
```

Note: if RK's PMU are not used, you should not configure it like this; find how the 32k in the schematic is provided, and then open the 32k accordingly.

2. If you need CPU to supply 32K, the following configuration needs to be added in dts (this way is not recommended):

```
+&pinctrl {
+    pinctrl-names = "default";
+    pinctrl-0 = <&rtc_32k>;
+}
```

The hardware connection is shown in the figure below:



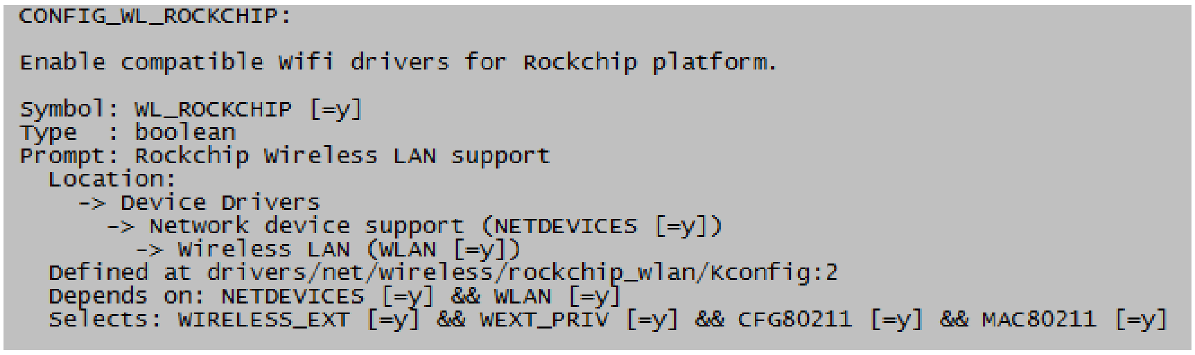
1.2 SDMMC Interface Is Connected to Wi-Fi Chip

For some special requirements, the Wi-Fi chip needs to be connected to SDMMC interface, so the configuration will be modified as follows:

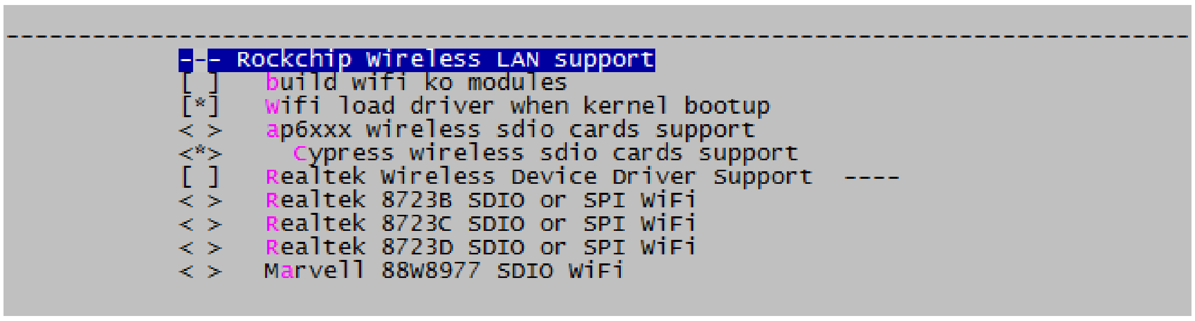
```
//Find the following two configurations, change &sdio to &sdmmc, and disabled
useless nodes;
&sdmmc {
    bus-width = <4>;
    cap-mmc-highspeed;
    cap-sd-highspeed;
    card-detect-delay = <200>;
    rockchip,default-sample-phase = <90>;
    supports-sd;
    sd-uhs-sdr12;
    sd-uhs-sdr25;
    sd-uhs-sdr104;
    vqmmc-supply = <&vccio_sd>;
-   status = "okay";
+   status = "disabled";
};

+&sdmmc {
-&sdio {
    max-frequency = <200000000>;
    bus-width = <4>;
    cap-sd-highspeed;
    cap-sdio-irq;
    keep-power-in-suspend;
    non-removable;
    rockchip,default-sample-phase = <90>;
    sd-uhs-sdr104;
    supports-sdio;
    mmc-pwrseq = <&sdio_pwrseq>;
    status = "okay";
};
```

1.3 Kernel Configuration

1.3.1 Wi-Fi Configuration



```
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
```

Wi-Fi driver can be built into kernel or in ko mode:

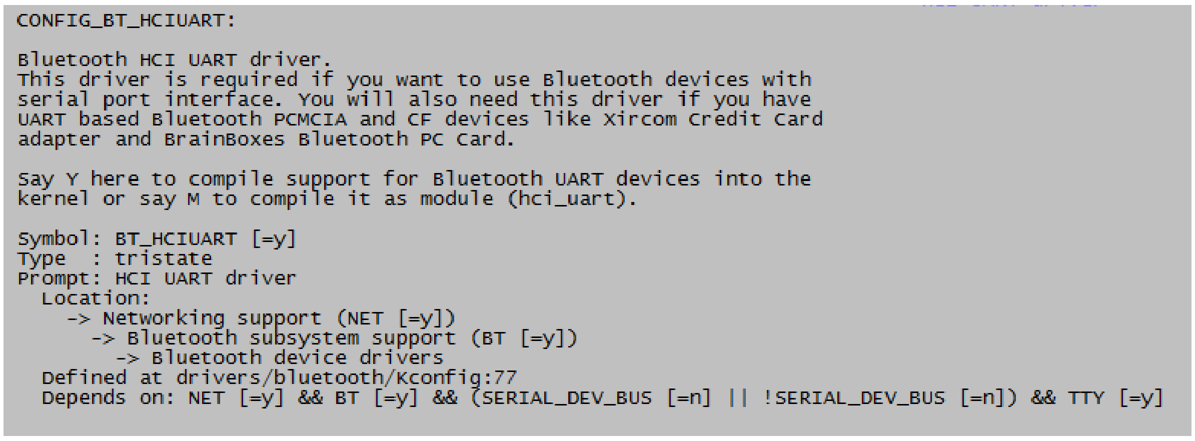
Remember that one of the following two configurations must be selected, otherwise Wi-Fi cannot be loaded!

```
# KO configuration is as follows
[*]   build wifi ko modules
[ ]   Wifi load driver when kernel bootup
# buildin configuration is as follows
[ ]   build wifi ko modules
[*]   Wifi load driver when kernel bootup
```

- **Only one model can be selected in Buildin, Realtek modules and ap6xxx modules cannot be selected as y at the same time, and Realtek's can only choose one of them;**
- ap6xxx and cypress are also mutually exclusive, you can only choose one and if you choose ap6xxx, the cypress configuration will disappear automatically, and if you remove the ap configuration, cypress will appear automatically;
- You can select multiple Wi-Fi in ko mode.

1.3.2 Bluetooth Configuration

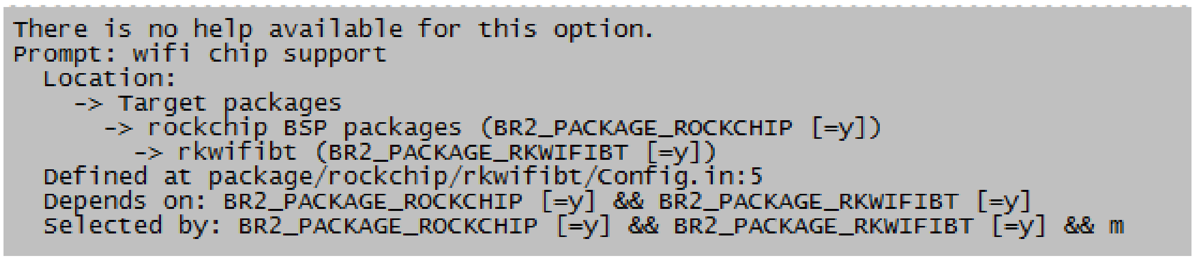
Both of Azurewave/AMPAK modules use the CONFIG_BT_HCIUART driver of kernel by default, while Realtek uses its own hci uart driver. The source code directory is as follows: `external\rkwifibt\realtek\bluetooth_uart_driver`, and they are loaded in ko mode, so when Realtek is used, don't forget to remove the **CONFIG_BT_HCIUART** configuration of kernel!



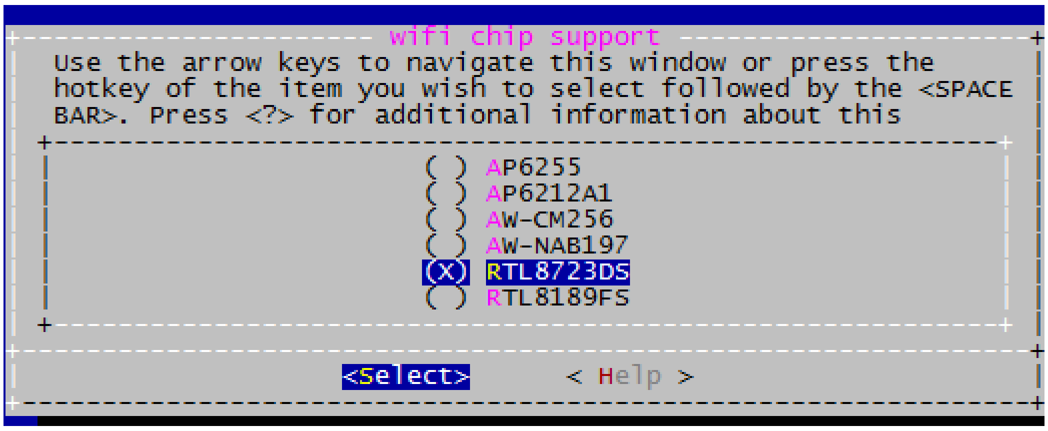
```
CONFIG_BT_HCIUART:

Bluetooth HCI UART driver.
This driver is required if you want to use Bluetooth devices with
serial port interface. You will also need this driver if you have
UART based Bluetooth PCMCIA and CF devices like Xircom Credit Card
adapter and BrainBoxes Bluetooth PC Card.

Say Y here to compile support for Bluetooth UART devices into the
kernel or say M to compile it as module (hci_uart).

Symbol: BT_HCIUART [=y]
Type  : tristate
Prompt: HCI UART driver
  Location:
    -> Networking support (NET [=y])
      -> Bluetooth subsystem support (BT [=y])
        -> Bluetooth device drivers
  Defined at drivers/bluetooth/Kconfig:77
  Depends on: NET [=y] && BT [=y] && (SERIAL_DEV_BUS [=n] || !SERIAL_DEV_BUS [=n]) && TTY [=y]
```

1.4 Buildroot Configuration

Choose the corresponding configuration according to the Wi-Fi used actually, which should be consistent with the kernel configuration:

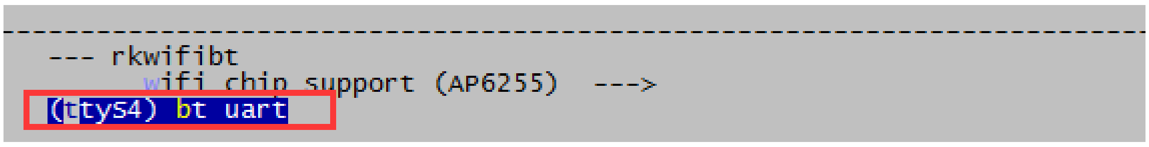


```
There is no help available for this option.
Prompt: wifi chip support
  Location:
    -> Target packages
      -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
        -> rkwifibt (BR2_PACKAGE_RKWIFIBT [=y])
  Defined at package/rockchip/rkwifibt/Config.in:5
  Depends on: BR2_PACKAGE_ROCKCHIP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

For modules with Bluetooth, the corresponding tty number and the hardware uart port should be configured (For USB modules does not need this configuration):



```
--- rkwifibt
      wifi chip support (AP6255)  --->
(ttyS4) bt uart
```

2. Wi-Fi/BT Files and Building Instructions

2.1 Building Files

Files directory corresponding to different Wi-Fi drivers:

```
kernel/drivers/net/wireless/rockchip_wlan/
kernel/drivers/net/wireless/rockchip_wlan/rkwifibt/ #AMPAK modules
kernel/drivers/net/wireless/rockchip_wlan/cywdhd/ #Cypress/Azurewave modules
kernel/drivers/net/wireless/rockchip_wlan/rtlxxx #Realtek module
```

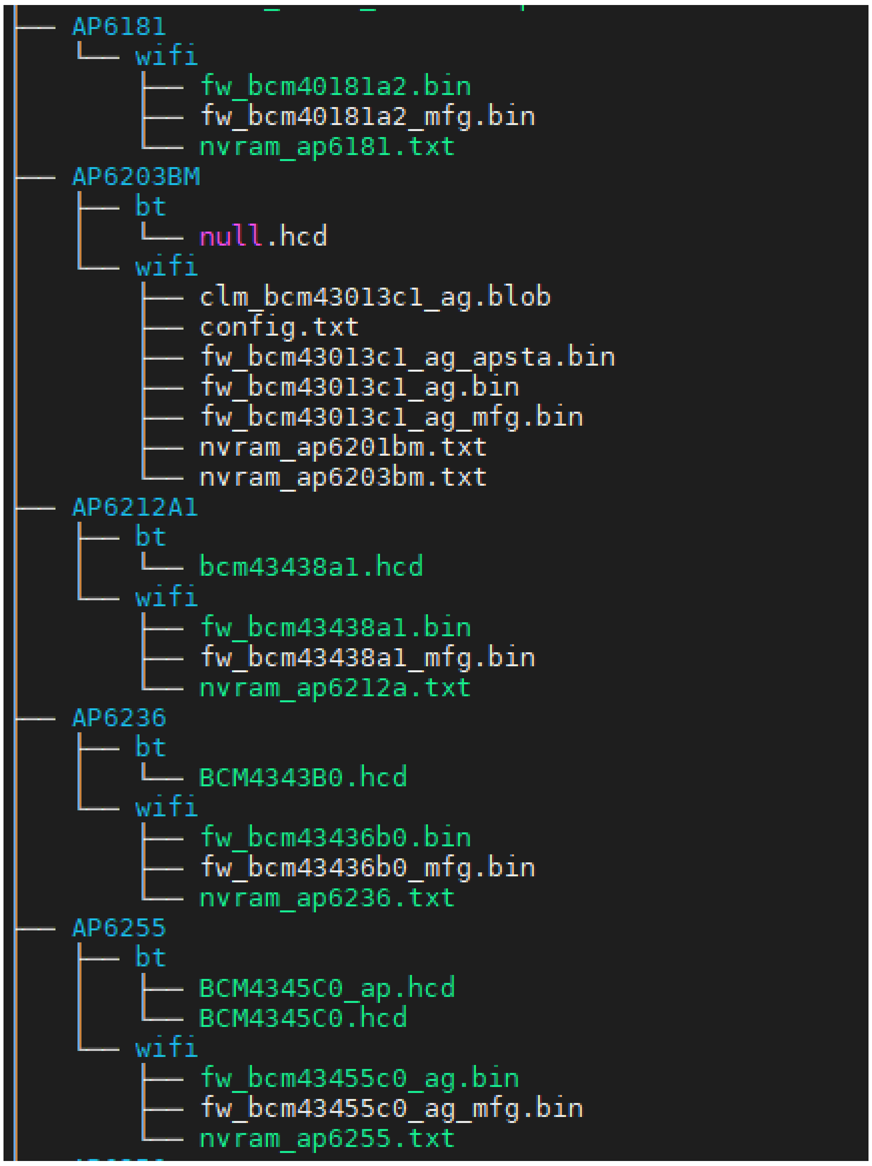
The directories of Firmware, Realtek BT drivers and Bluetooth Firmware files of Wi-Fi are as follows:

`external/rkwifibt/`

AP (CY) Wi-Fi/BTFirmware:

`external/rkwifibt/firmware/broadcom/`

From the picture below, you can see the names of Wi-Fi/BT firmware corresponding to each model of AMPAK Wi-Fi:



```
├── AP6181
│   └── wifi
│       ├── fw_bcm40181a2.bin
│       ├── fw_bcm40181a2_mfg.bin
│       └── nvram_ap6181.txt
├── AP6203BM
│   ├── bt
│   │   └── null.hcd
│   └── wifi
│       ├── clm_bcm43013c1_ag.blob
│       ├── config.txt
│       ├── fw_bcm43013c1_ag_apsta.bin
│       ├── fw_bcm43013c1_ag.bin
│       ├── fw_bcm43013c1_ag_mfg.bin
│       ├── nvram_ap6201bm.txt
│       └── nvram_ap6203bm.txt
├── AP6212A1
│   ├── bt
│   │   └── bcm43438a1.hcd
│   └── wifi
│       ├── fw_bcm43438a1.bin
│       ├── fw_bcm43438a1_mfg.bin
│       └── nvram_ap6212a.txt
├── AP6236
│   ├── bt
│   │   └── BCM4343B0.hcd
│   └── wifi
│       ├── fw_bcm43436b0.bin
│       ├── fw_bcm43436b0_mfg.bin
│       └── nvram_ap6236.txt
├── AP6255
│   ├── bt
│   │   ├── BCM4345C0_ap.hcd
│   │   └── BCM4345C0.hcd
│   └── wifi
│       ├── fw_bcm43455c0_ag.bin
│       ├── fw_bcm43455c0_ag_mfg.bin
│       └── nvram_ap6255.txt
```

Realtek BT driver and Firmware:

```
external/rkwifibt/realtek/bluetooth_uart_driver/ #BT driver
external/rkwifibt/realtek/rtk_hciattach/         #Initialization program
external/rkwifibt/realtek/RTL8723DS/             #BT firmware
```

2.2 Files and Their Path During Running

After the system is started, we have to know the files and their paths to be used when Wi-Fi Bluetooth is started, so when encountering problems with Wi-Fi/BT starting abnormally, first do confirm the corresponding Wi-Fi Bluetooth files exist! If they are not exist, there must be an error in the configuration, please check carefully!

For AMPAK/Azurewave modules, take AP6255 as an example (refer to the picture in chapter 2.1 for detailed relationship). The files and their paths required for Wi-Fi running: (for kernel4.19, changed from system to vendor directory):

```
/system/lib/modules/bcmdhd.ko               #Driver ko(if build in ko model)
/system/etc/firmware/fw_bcm43455c0_ag.bin #Driver firmware file path
/system/etc/firmware/nvram_ap6255.txt
/system/etc/firmware/BCM4345C0.hcd        #Bluetooth firmware file (if there is
Bluetooth function)
```

For Realtek modules, take RTL8723DS as an example:

```
/system/lib/modules/8723ds.ko         #Driver ko(if build in ko model)

# If there is Bluetooth function
/usr/lib/modules/hci_uart.ko          #Bluetooth ko
/lib/firmware/rtlbt/rtl8723d_config   #Bluetooth firmware
/lib/firmware/rtlbt/rtl8723d_fw
/lib/firmware/rtlbt/mp_rtl8723d_config #Bluetooth test firmware
/lib/firmware/rtlbt/mp_rtl8723d_fw
```

The files and directories corresponding to some Realtek chips are listed below. Note that USB interface is special, and the firmware files are placed in the /lib/firmware/ directory:

Chip	I/F for BT driver	FW/Config Path	FW Filename	Config Filename
RTL8723DS	UART	/lib/firmware/rtlbt/	rtl8723d_fw	rtl8723d_config
RTL8821CS	UART	/lib/firmware/rtlbt/	rtl8821c_fw	rtl8821c_config
RTL8821CU	USB	/lib/firmware/	rtl8821cu_fw	rtl8821cu_config

2.3 Building Rules

The corresponding building rule files:

```
buildroot/package/rockchip/rkwifibt/Config.in   # The same rules as regular
Kconfig
buildroot/package/rockchip/rkwifibt/rkwifibt.mk # Similar to Makefile
```

Please read these two rkwifibt.mk and Config.in files carefully. The core work of these two files are:

- **Build module Bluetooth driver KO files, such as Realtek's uart/usb Bluetooth driver, and some vendor's private executable binary tools such as AMPAK's wl, Realtek's rtwpriv and other tools;**
- **According to the configured Wi-Fi/BT model, copy and install the corresponding firmware/driver KO/executable files to the specified directory, please refer to the location and corresponding directory in chapter 2.2;**

So developers must be familiar with building rules, which is very important for following debugging.

As the progress and time of SDK version of each chip platform are inconsistent, the content of the in/mk file obtained by customers may be different, but the general rules are the same.

rkwifibt.mk: all modifications are carried out according to the following 3 key rules:

```
# Specify the source directory of rkwifibt
RKWIFIBT_SITE = $(TOPDIR)/../external/rkwifibt
#Is the function of the building process, pass the building and link options to the source code, and call the source code Makefile to execute the building
RKWIFIBT_BUILD_CMDS
# After building, install automatically, Buildroot will install the built libraries and bin files to the specified directory
RKWIFIBT_INSTALL_TARGET_CMDS
```

Config.in: specify the corresponding Wi-Fi/BT model and pass it to the mk file:

`config BR2_PACKAGE_RKWIFIBT_AP6236`

Configure building or update:

```
#Project root directory: first configure the board file of the whole project, according to corresponding sdk documents
make menuconfig
#Select the corresponding Wi-Fi Bluetooth model
make savedefconfig #Remember to save the corresponding configuration
make rkwifibt-dirclean #Clear previous building
make rkwifibt-rebuild #Rebuild
```

Note: please learn to see the output of the printed log when `make rkwifibt-rebuild` builds, which contains the building/copy process of the mk file above, which is helpful to analyze and solve problems such as building errors/copy errors.

2.4 Update

1. For the modification of kernel Wi-Fi configuration, after selecting `make menuconfig`, be sure to save the corresponding defconfig file. For example, If the following is used: `kernel/arch/arm/configs/rockchip_xxxx_defconfig`, and the corresponding modification should be updated to this file, otherwise the update will not take effect.

```
# Kernel directory
make menuconfig
# Modify the corresponding configuration
make savedefconfig
cp defconfig arch/arm/configs/rockchip_xxxx_defconfig
```

2. **For the modification of Buildroot configuration, after selecting `make menuconfig`, execute `make savedeconfig` in the root directory to save.**

```
# Top directory
source xxxx             #First select the configuration of the corresponding
project, please refers to the SDK development configuration document
make menuconfig
# Select the corresponding Wi-Fi model
make savedeconfig       #Save configuration
make rkwifibt-dirclean #Clear the previous
make rkwifibt-rebuild  #Rebuild
./build.sh              #Repackage to generate firmware
```

Note: Be sure to `make savedefconfig`, otherwise it will be overwritten by the original when building, resulting in the modification not taking effect.

3. Wi-Fi/BT Function Verification

3.1 Wi-Fi STA Test

3.1.1 Turn Wi-Fi on and off

If there is no wlan0 node, please first check whether the dts/driver is configured correctly and the driver is loaded (ko or buildin). If it is correct, please refer to Chapter 6 for troubleshooting.

Turn on Wi-Fi

```
echo 1> /sys/class/rfkill/rfkill1/state # it is rfkill0 when the Bluetooth node
is not turned on
ifconfig wlan0 up
wlan0 Link encap:Ethernet HWaddr F0:85:C1:0F:9C:02
          UP BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

Then check whether the Wi-Fi service process is started: check whether there is a wpa_supplicant process, if it is not started, you can start it manually:

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

Note: The `wpa_supplicant.conf` file should be modified according to the storage location of the platform used actually.

Turn off Wi-Fi:

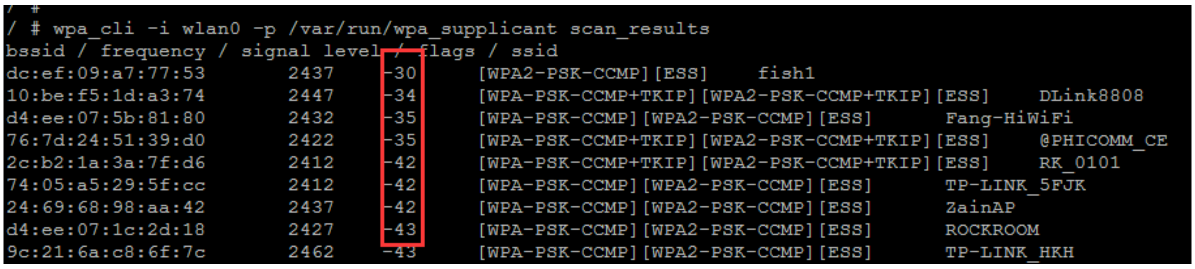
`ifconfig wlan0 down`

`killall wpa_supplicant`

3.1.2 Scan APs nearby

`wpa_cli -i wlan0 -p /var/run/wpa_supplicant scan`

`wpa_cli -i wlan0 -p /var/run/wpa_supplicant scan_results`



```
/ #
/ # wpa_cli -i wlan0 -p /var/run/wpa_supplicant scan_results
bssid / frequency / signal level / flags / ssid
dc:ef:09:a7:77:53       2437    -30     [WPA2-PSK-CCMP][ESS]    fish1
10:be:f5:1d:a3:74       2447    -34     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    DLink8808
d4:ee:07:5b:81:80       2432    -35     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      Fang-HiWiFi
76:7d:24:51:39:d0       2422    -35     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    @PHICOMM_CE
2c:b2:1a:3a:7f:d6       2412    -42     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    RK_0101
74:05:a5:29:5f:cc       2412    -42     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      TP-LINK_5FJK
24:69:68:98:aa:42       2437    -42     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      ZainAP
d4:ee:07:1c:2d:18       2427    -43     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      ROCKROOM
9c:21:6a:c8:6f:7c       2462    -43     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      TP-LINK_HKH
```

Note: check whether the number of scanned hotspots matches the number of routers around you similarly, you can compare it with Wi-Fi number scanned by your mobile phone (if your module does not support 5G, only compare 2.4G Wi-Fi numbers); also check the signal strength of the router closest to you, if the router is very close to you, but the signal strength is very weak (regular: -20 to -65; weak: -65 to -70 ; Poor -70 to -90), then check whether your Wi-Fi module is connected to antenna, whether the module's RF index is qualified, etc. (please refer to Chapter 5 Wi-Fi/BT Hardware Test) .

3.1.3 Connect to Router

The first way:

```
//If ctrl_interface interface configuration has to be modified, the corresponding
wpa_cli command -p parameter should be modified accordingly, wpa_cli -i wlan0 -p
<ctrl_interface> xxx
vi /data/cfg/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant //It is not recommended to modify by
default
ap_scan=1
update_config=1 //This configuration will save the hotspots configured by the
wpa_cli command to the conf file (wpa_cli save_config)
//Add the following configuration items
network={
        ssid="WiFi-AP" // The name of the Wi-Fi
        psk="12345678" // The password of the Wi-Fi
        key_mgmt=WPA-PSK // Encryption configuration; change to key_mgmt=NONE if
not encrypted
}
//Let the wpa_supplicant process read the above configuration again by the
following command:
wpa_cli -i wlan0 -p /var/run/wpa_supplicant reconfigure
//Send a connection:
wpa_cli -i wlan0 -p /var/run/wpa_supplicant reconnect
```

The second way:

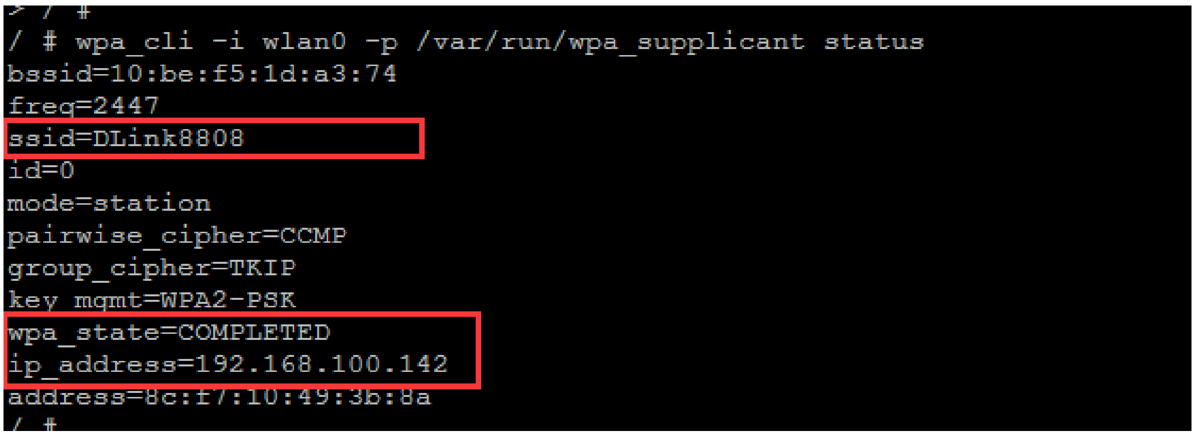
The latest SDK integrates the wifi_start.sh script, you can directly add ssid and password behind the script if it exists:

`wifi_start.sh fanxing 12345678`

The third way:

```
#encryption:
wpa_cli -i wlan0 -p /var/run/wpa_supplicant remove_network 0
wpa_cli -i wlan0 -p /var/run/wpa_supplicant ap_scan 1
wpa_cli -i wlan0 -p /var/run/wpa_supplicant add_network
wpa_cli -i wlan0 -p /var/run/wpa_supplicant set_network 0 ssid "dlink"
wpa_cli -i wlan0 -p /var/run/wpa_supplicant set_network 0 key_mgmt WPA-PSK
wpa_cli -i wlan0 -p /var/run/wpa_supplicant set_network 0 psk'"12345678"'
wpa_cli -i wlan0 -p /var/run/wpa_supplicant select_network 0
wpa_cli -i wlan0 -p /var/run/wpa_supplicant save_config # Save the above
configurations to the conf file

#No encryption:
wpa_cli -i wlan0 -p /var/run/wpa_supplicant remove_network 0
wpa_cli -i wlan0 -p /var/run/wpa_supplicant ap_scan 1
wpa_cli -i wlan0 -p /var/run/wpa_supplicant add_network
wpa_cli -i wlan0 -p /var/run/wpa_supplicant set_network 0 ssid "dlink"
wpa_cli -i wlan0 -p /var/run/wpa_supplicant set_network 0 key_mgmt NONE
wpa_cli -i wlan0 -p /var/run/wpa_supplicant select_network 0
wpa_cli -i wlan0 -p /var/run/wpa_supplicant save_config
```

Successful connection:



```
/ # wpa_cli -i wlan0 -p /var/run/wpa_supplicant status
bssid=10:be:f5:1d:a3:74
freq=2447
ssid=DLink8808
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=TKIP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.100.142
address=8c:f7:10:49:3b:8a
/ #
```

If there is `wpa_state=COMPLETED` but no valid ip_address, check whether the process of obtaining IP address by **dhcpcd** is started; if wpa_state is not COMPLETED, please **check the 2.1.1 scanning chapter first**.

3.2 Wi-Fi AP Hotspot Verification

SDK integrates related programs, execute: softapDemo apName (to open the hotspot with the name of apName without encryption by default) to enable hotspot mode.
Code and building file path:

```
/external/softapDemo/src/main.c
buildroot/package/rockchip/softap/Config.in softap.mk
make softap-dirclean
make softap
```

RTL module: take p2p0 as softap function to generate p2p0 through kernel driver configuration. If there is no p2p0 node, please check the configuration below:

```
+++ b/drivers/net/wireless/rockchip_wlan/rtl8xxx/Makefile
@@ -1593,7 +1593,7 @@ endif
ifeq ($(CONFIG_PLATFORM_ARM_RK3188), y)
EXTRA_CFLAGS += -DCONFIG_PLATFORM_ANDROID
+EXTRA_CFLAGS += -DCONFIG_CONCURRENT_MODE
```

AP/Azurewave module: wlan1 is used as softap function, and generate wlan1 nodes by iw command:

`iw phy0 interface add wlan1 type managed`

Debug and customized modification:

```
//You can add encryption, modify IP address and dns and other related information
here by yourself
int wlan_accesspoint_start(const char* ssid, const char* password)
{
    //Configuration of creating a hotspot
    create_hostapd_file(ssid, password);
    //softap_name: wlan1/p2p0
    sprintf(cmdline, "ifconfig %s up", softap_name);
    //Set a customize IP address
    sprintf(cmdline, "ifconfig %s 192.168.88.1 netmask 255.255.255.0",
softap_name);
    //Create dns file
    creat_dnsmasq_file();
}

//Create a dns file, which must be consistent with your customized IP address,
otherwise your phone will not obtain IP
bool creat_dnsmasq_file()
{
    FILE* fp;
    fp = fopen(DNSMASQ_CONF_DIR, "wt+");
    if (fp != 0) {
        fputs("user=root\n", fp);
        fputs("listen-address=", fp);
        fputs(SOFTAP_INTERFACE_STATIC_IP, fp);
        fputs("\n", fp);
        fputs("dhcp-range=192.168.88.50,192.168.88.150\n", fp);
        fputs("server=/google/8.8.8.8\n", fp);
        fclose(fp);
        return true;
    }
    DEBUG_ERR("---open dnsmasq configuarion file failed!!---");
    return true;
}

//Create AP hotspot configuration file, please consult Wi-Fi vendors for details,
the parameters here have a close relationship with chip specifications
int create_hostapd_file(const char* name, const char* password)
{
    FILE* fp;
    char cmdline[256] = {0};

    fp = fopen(HOSTAPD_CONF_DIR, "wt+");

    if (fp != 0) {
        sprintf(cmdline, "interface=%s\n", softap_name);
        fputs(cmdline, fp);
        fputs("ctrl_interface=/var/run/hostapd\n", fp);
        fputs("driver=nl80211\n", fp);
        fputs("ssid=", fp);
        fputs(name, fp);
        fputs("\n", fp);
        fputs("channel=6\n", fp); // Channel settings
        fputs("hw_mode=g\n", fp);
        fputs("ieee80211n=1\n", fp);
        fputs("ignore_broadcast_ssid=0\n", fp);
#if 0 //If you choose encryption, modify here
        fputs("auth_algs=1\n", fp);
        fputs("wpa=3\n", fp);
        fputs("wpa_passphrase=", fp);
        fputs(password, fp);
        fputs("\n", fp);
        fputs("wpa_key_mgmt=WPA-PSK\n", fp);
        fputs("wpa_pairwise=TKIP\n", fp);
        fputs("rsn_pairwise=CCMP", fp);
#endif
        fclose(fp);
        return 0;
    }
    return -1;
}

int main(int argc, char **argv)
    //Set the corresponding hotspot interface according to the WiFi model
    if (!strncmp(wifi_type, "RTL", 3))
        strcpy(softap_name, "p2p0");
    else
        strcpy(softap_name, "wlan1");

    ... ...
    if (!strncmp(wifi_type, "RTL", 3)) {
        //Realtek module will generate p2p0 node automatically after opening the
coexistence mode
console_run("ifconfig p2p0 down");
        console_run("rm -rf /userdata/bin/p2p0");
        wlan_accesspoint_start(apName, NULL);
    } else {
        console_run("ifconfig wlan1 down");
        console_run("rm -rf /userdata/bin/wlan1");
        console_run("iw dev wlan1 del");
        console_run("ifconfig wlan0 up");
        //AP module needs iw command to generate wlan1 node for softap use
        console_run("iw phy0 interface add wlan1 type managed");
        wlan_accesspoint_start(apName, NULL);
    }
```

After executing the command, you will see the corresponding AP under the setting Wi-Fi interface of the phone. If not, please troubleshoot:

First: ensure that the configuration files mentioned above are configured correctly;
Second: confirm whether there is wlan1 or p2p0 node in ifconfig;
Third: whether the hostapd/dnsmasq process has started successfully;

3.3 BT Verification Test

First of all, make sure that dts and the corresponding Buildroot configuration are correct by referring to Chapter 1. Here are two types of commonly used BT modules. If the configuration is correct, the system will generate a **bt_pcba_test** (or the latest SDK will generate a **bt_init.sh**) script program (refer to the building configuration file in chapter 1.4).

Realtek modules:

```
/ # cat usr/bin/bt_pcba_test (bt_init.sh)
#!/bin/sh

killall rtk_hciattach

echo 0> /sys/class/rfkill/rfkill0/state #Power off
sleep 1
echo 1> /sys/class/rfkill/rfkill0/state #Power on
sleep 1
insmod /usr/lib/modules/hci_uart.ko            # Realtek modules need to load a
specific driver
rtk_hciattach -n -s 115200 /dev/ttyS4 rtk_h5 & # The blue refers to which uart
port is used by Bluetooth

#Note: every time you start a test, you have to kill the rtk_hciattach process
firstly
```

Azurewave/AMPAK Modules:

```
/ # cat usr/bin/bt_pcba_test (bt_init.sh)
#!/bin/sh

killall brcm_patchram_plus1

echo 0> /sys/class/rfkill/rfkill0/state # Power off
sleep 2
echo 1> /sys/class/rfkill/rfkill0/state # Power on
sleep 2

brcm_patchram_plus1 --bd_addr_rand --enable_hci --no2bytes --
use_baudrate_for_download --tosleep 200000 --baudrate 1500000 --patchram
/system/etc/firmware/bcm43438a1.hcd /dev/ttyS4 &

#Note: every time you start a test, you have to kill the brcm_patchram_plus1
process firstly
```

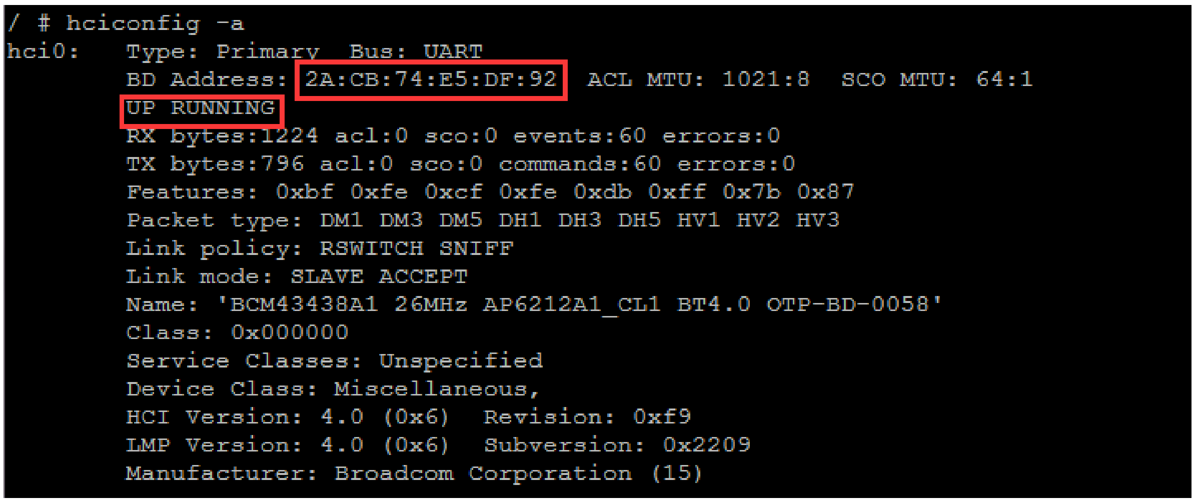
bcm43438a1.hcd represents the firmware file corresponding to the BT model, and /dev/ttyS4 is the UART port Bluetooth used.

Note: `rtk_hciattach, hci_uart.ko, bcm43438a1.hcd` and other files are generated only when the correct Wi-Fi/BT modules are selected in Buildroot configuration in Chapter 1. If these files are not available, please check the above configurations (please refer to the building configuration file in Chapter 1.4).

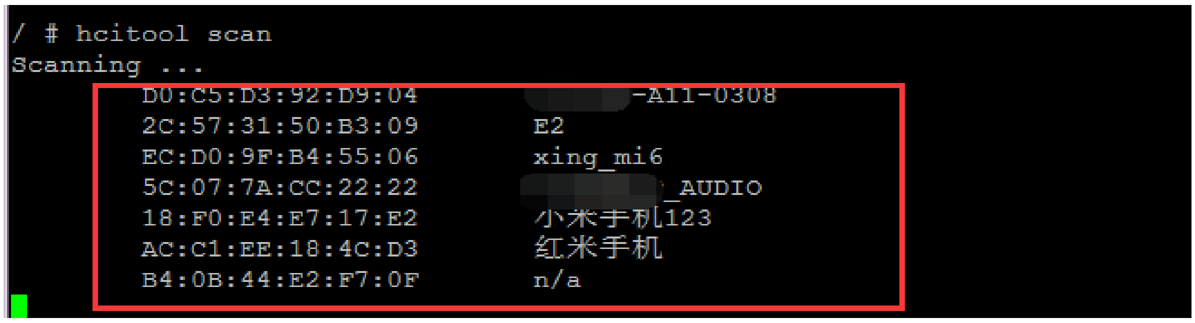
After executing the script, execute: **(Note: If there is no hciconfig command, please select BR2_PACKAGE_BLUEZ5_UTILS in Buildroot configuration to build and update test)**

`hciconfig hci0 up`
`hciconfig -a`

Normally, you will see:


```
/ # hciconfig -a
hci0:   Type: Primary  Bus: UART
        BD Address: 2A:CB:74:E5:DF:92  ACL MTU: 1021:8  SCO MTU: 64:1
        UP RUNNING
        RX bytes:1224 acl:0 sco:0 events:60 errors:0
        TX bytes:796 acl:0 sco:0 commands:60 errors:0
        Features: 0xbf 0xfe 0xcf 0xfe 0xdb 0xff 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH SNIFF
        Link mode: SLAVE ACCEPT
        Name: 'BCM43438A1 26MHz AP6212A1_CL1 BT4.0 OTP-BD-0058'
        Class: 0x000000
        Service Classes: Unspecified
        Device Class: Miscellaneous,
        HCI Version: 4.0 (0x6)  Revision: 0xf9
        LMP Version: 4.0 (0x6)  Subversion: 0x2209
        Manufacturer: Broadcom Corporation (15)
```

BT scanning: `hcitool scan`


```
/ # hcitool scan
Scanning ...
        D0:C5:D3:92:D9:04       [redacted]-A11-0308
        2C:57:31:50:B3:09       E2
        EC:D0:9F:B4:55:06       xing_mi6
        5C:07:7A:CC:22:22       [redacted]_AUDIO
        18:F0:E4:E7:17:E2       小米手机123
        AC:C1:EE:18:4C:D3       红米手机
        B4:0B:44:E2:F7:0F       n/a
```

3.4 Wi-Fi Suspend and Resume

At present, Wi-Fi supports the network resume function. For example, when the device connects to an AP and obtains an IP address, when the device suspends, we can resume the system through a wireless network packet (ping). Generally, any network packet sent to the device can resume the system.

Modify the wpa_supplicant.conf file and add the following configuration:

```
wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
update_config=1
ap_scan=1
+wowlan_triggers=any # Add this configuration
```

For Realtek Wi-Fi, please check whether the following configuration is in the Makefile of the corresponding driver:

```
kernel/drivers/net/wireless/rockchip_wlan/rtl8xxx/Makefile
+CONFIG_WOWLAN = y
+CONFIG_GPIO_WAKEUP = y
```

DTS configuration: check the schematic diagram to ensure that WIFI_WAKE_HOST (or WL_HOST_WAKE) PIN is connected to the controller, and then check whether the following configuration of dts is correct:

```
WIFI,host_wake_irq = <&gpio0 RK_PA0 GPIO_ACTIVE_HIGH>
```

System and Wi-Fi suspend testing:

```
dhd_priv setsuspendmode 1 # Only for Zhengji Haihua module, Realtek does not need
this command
echo mem> /sys/power/state
```

At this time, devices in the same local area network can ping this device. Normally, you can find that the system is resumed. Note to return to normal Wi-Fi working state after the system is resumed:

```
dhd_priv setsuspendmode 0 # Only for AMPAK and AzureWave modules, Realtek's do
not need this command
```

Troubleshooting: If the system does not wake up as expected, please check whether the wake pin is configured correctly and the level status is correct, whether 32.768k is turned off, etc.

3.5 Wi-Fi Monitor Mode

AMPAK or AzureWave Wi-Fi modules:

```
#Set up monitoring channel:
dhd_priv channel 6 //channal numbers

#Open monitor mode:
dhd_priv monitor 1

#Close monitor mode:
dhd_priv monitor 0
```

Realtek Wi-Fi modules:

```
#Driver Makefile should be opened:
+ CONFIG_WIFI_MONITOR = y

#Open wlan0 and close p2p0
ifconfig wlan0 up
ifconfig p2p0 down

#Open monitor mode
iwconfig wlan0 mode monitor
or
iw dev wlan0 set type monitor

#Switch channels
echo "<chan> 0 0"> /proc/net/<rtk_module>/wlan0/monitor // <rtk_module> is the
realtek wifi module name, such like rtl8812au, rtl8188eu ..etc
```

3.6 Wi-Fi P2P Verification

```
#New configuration file: p2p_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
update_config=1
device_name=p2p_name
device_type=10-0050F204-5
config_methods=display push_button keypad virtual_push_button physical_display
p2p_add_cli_chan=1
pmf=1

#Start: (kill the previous firstly)
wpa_supplicant -B -i wlan0 -c /tmp/p2p_supplicant.conf
wpa_cli
> p2p_find
>

#At this time, open p2p on the mobile phone, you wll search for the above
device_name=p2p_name, click to connect

#At this time it will display on the device:  //The following is my phone
> <3>P2P-PROV-DISC-PBC-REQ 26:31:54:8e:14:e7 p2p_dev_addr=26:31:54:8e:14:e7
pri_dev_type=10-0050F204-5 name='www' config_methods =0x188 dev_capab=0x25
group_capab=0x0

#The device responds and send a connection command, and pay attention to the MAC
address to be consistent with the above
>p2p_connect 26:31:54:8e:14:e7 pbc go_intent=1

> p2p_connect 26:31:54:8e:14:e7 pbc go_intent=1
OK
<3>P2P-FIND-STOPPED
<3>P2P-GO-NEG-SUCCESS role=client freq=5200 ht40=0 peer_dev=26:31:54:8e:14:e7
peer_iface=26:31:54:8e:94:e7 wps_method=PBC
<3>P2P-GROUP-FORMATION-SUCCESS
<3>P2P-GROUP-STARTED p2p-wlan0-2 client ssid="DIRECT-24-www" freq=5200
psk=3d67671b71f7a171118c1ace34ae5e4bcc8e17394394e258be91f55b7ab63748
go_dev_addr=26:31:54:8e:14e:14e:14
> #At this time the connection is successful
> quit

ifconfig
p2p-wlan0-2 Link encap:Ethernet HWaddr 82:C5:F2:2E:7F:89
          inet addr:192.168.49.220 Bcast:192.168.49.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:470 errors:0 dropped:0 overruns:0 frame:0
          TX packets:344 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes: 71779 (70.0 KiB) TX bytes: 33829 (33.0 KiB)
#You can see that the device is connected to the phone, and it is normal if you
can ping each other.
```

3.7 Connection Function

Case: Wi-Fi enables wlan0 to connect to an AP that can access Internet, together with opening wlan1 or p2p0 as a hotspot in Chapter 3.2, so that the mobile phone can connect to the hotspot for Internet access. The configuration is as follows, and open the following configuration in the kernel :

```
+CONFIG_NETFILTER=y
+CONFIG_NF_CONNTRACK=y
+CONFIG_NF_TABLES=y
+CONFIG_NF_TABLES_INET=y
+CONFIG_NF_CONNTRACK_IPV4=y
+CONFIG_IP_NF_IPTABLES=y
+CONFIG_IP_NF_NAT=y
+CONFIG_IP_NF_TARGET_MASQUERADE=y
+CONFIG_BRIDGE=y
```

Execute the following two commands to start the connection function, the following IP address is the address configured when softap is started:

```
iptables -t nat -A POSTROUTING -s 192.168.43.0/24 -o wlan0 -j MASQUERADE
echo "1"> /proc/sys/net/ipv4/ip_forward
```

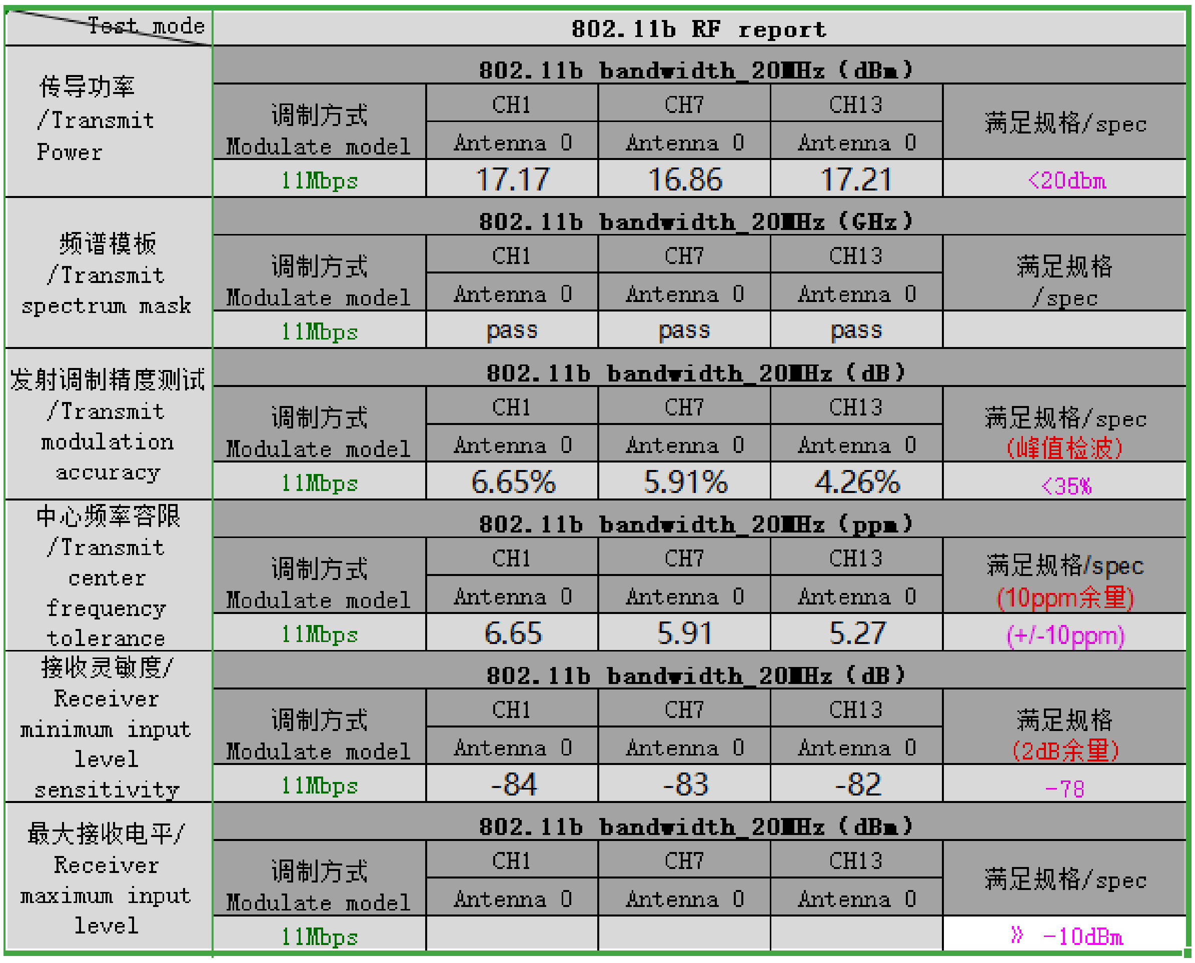
4. Wi-Fi/BT Hardware RF Target

4.1 Test Items

Wi-Fi/BT test items:

For example: transmit power, EVM, crystal frequency offset, receiving sensitivity, etc. (Professional equipments are required, you can contact the module vendors or the manufacturers for assistance);

Example: (b/g/n/ac):

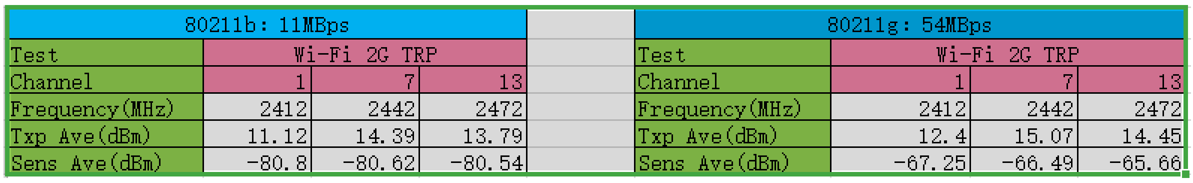


Test mode	802.11b RF report				
传导功率 /Transmit Power	802.11b bandwidth_20MHz (dBm)				
	调制方式 Modulate model	CH1	CH7	CH13	满足规格/spec
		Antenna 0	Antenna 0	Antenna 0	
	11Mbps	17.17	16.86	17.21	<20dbm
频谱模板 /Transmit spectrum mask	802.11b bandwidth_20MHz (GHz)				
	调制方式 Modulate model	CH1	CH7	CH13	满足规格 /spec
		Antenna 0	Antenna 0	Antenna 0	
	11Mbps	pass	pass	pass	
发射调制精度测试 /Transmit modulation accuracy	802.11b bandwidth_20MHz (dB)				
	调制方式 Modulate model	CH1	CH7	CH13	满足规格/spec (峰值检波)
		Antenna 0	Antenna 0	Antenna 0	
	11Mbps	6.65%	5.91%	4.26%	<35%
中心频率容限 /Transmit center frequency tolerance	802.11b bandwidth_20MHz (ppm)				
	调制方式 Modulate model	CH1	CH7	CH13	满足规格/spec (10ppm余量)
		Antenna 0	Antenna 0	Antenna 0	
	11Mbps	6.65	5.91	5.27	(+/-10ppm)
接收灵敏度/ Receiver minimum input level sensitivity	802.11b bandwidth_20MHz (dB)				
	调制方式 Modulate model	CH1	CH7	CH13	满足规格 (2dB余量)
		Antenna 0	Antenna 0	Antenna 0	
	11Mbps	-84	-83	-82	-78
最大接收电平/ Receiver maximum input level	802.11b bandwidth_20MHz (dBm)				
	调制方式 Modulate model	CH1	CH7	CH13	满足规格/spec
		Antenna 0	Antenna 0	Antenna 0	
	11Mbps				》 -10dBm

Antenna test items:

Passive S11, OTA test of the whole active Wi-Fi antenna (you need to go to the antenna factory with professional equipment to test, you can contact the module vendors or the manufacturers for assistance);

Example:



80211b: 11MBps					80211g: 54MBps			
Test	Wi-Fi 2G TRP				Test	Wi-Fi 2G TRP		
Channel	1	7	13		Channel	1	7	13
Frequency(MHz)	2412	2442	2472		Frequency(MHz)	2412	2442	2472
Txp Ave(dBm)	11.12	14.39	13.79		Txp Ave(dBm)	12.4	15.07	14.45
Sens Ave(dBm)	-80.8	-80.62	-80.54		Sens Ave(dBm)	-67.25	-66.49	-65.66

4.2 Test Tools and Methods

The PDF/TXT documents mentioned below can be found in the docs/linux/wifibt directory, and if customers have test problems or without professional test equipment, please directly contact module vendors for assistance.

4.2.1 Realtek Test

Generally, there are two types of COB and module. Modules are generally strictly tested by the module factory and flashed calibrated data to internal efuse by default. Customers only need to test and verify whether the indicators are qualified; while COB need to design Wi-Fi peripherals circuits and additional components, so you need to complete RF calibration test with Realtek, and integrate the calibrated data into efuse of the chip or load it by driver. Please contact module vendor directly for details.

Wi-Fi test:

```
Please refer to Quick_Start_Guide_V6.txt, pay special attention to replace the
command iwpriv with rtwpriv.
```

BT test:

Please refer to MP tool user guide for linux20180319.pdf (please contact module vendors or manufacturers for detailed test items).

```
# Note: please turn on the BT power before testing
echo 0> sys/class/rfkill/rfkill0/state
sleep 1
echo 1> sys/class/rfkill/rfkill0/state

# Special attention: the process rtk_hciattach cannot be run, please kill killall
rtk_hciattach

///////////
/ #
/ # rtlbtmp
::::::::::::::::::::::::::::::::::::::::::::::
:::::::: Bluetooth MP Test Tool Starting ::::::::

>
>
>enable uart:/dev/ttyS4 # Note that ttySX corresponds to connected hardware uart
port actually
>
>>> enable[Success:0]
```

If it is Realtek's COB solution, and you need to integrate the test calibration data map file into driver, please refer to the following way:

```
drivers/net/wireless/rockchip_wlan/rtl8xxx/core/efuse/rtw_efuse.c
#ifdef CONFIG_EFUSE_CONFIG_FILE
u32 rtw_read_efuse_from_file(const char *path, u8 *buf, int map_size)
{
    u32 i;
    u8 c;
    u8 temp[3];
    u8 temp_i;
    u8 end = _FALSE;
    u32 ret = _FAIL;

    u8 *file_data = NULL;
    u32 file_size, read_size, pos = 0;
    u8 *map = NULL;

    if (rtw_is_file_readable_with_size(path, &file_size) != _TRUE) {
        RTW_PRINT("%s %s is not readable\n", __func__, path);
        goto exit;
    }

    file_data = rtw_vmalloc(file_size);
    if (!file_data) {
        RTW_ERR("%s rtw_vmalloc(%d) fail\n", __func__, file_size);
        goto exit;
    }

    #if 0 //Block out here
    read_size = rtw_retrieve_from_file(path, file_data, file_size);
    if (read_size == 0) {
        RTW_ERR("%s read from %s fail\n", __func__, path);
        goto exit;
    }
    ... ...
    RTW_PRINT("efuse file:%s, 0x%03x byte content read\n", path, i);
    #endif

//Change the calibration "map file" provided by the module vendor into an array
form and assign it to map.
    _rtw_memcpy(buf, map, map_size); //It is the operation that final assignment
of map to buf

    ret = _SUCCESS;

exit:
    if (file_data)
        rtw_vmfree(file_data, file_size);
    if (map)
        rtw_vmfree(map, map_size);

    return ret;
}
```

4.2.2 AP/CY Test

Wi-Fi Test

Firstly, replace it with test firmware: the test firmware of each AP module is different, such as:

```
AP6236 -> fw_bcm43436b0_mfg.bin
AP6212A -> fw_bcm43438a1_mfg.bin
```

The fw_bcmxxx_mfg.bin and APxxxx should be matched according to your module model, otherwise it cannot be tested! So confirm whether there is a test firmware of the corresponding model firstly, if you don't find it, please ask module vendor to provide it.

The latest SDK has built-in test firmware for supporting models, so use the built-in test script to let WiFi enter RF test mode directly:

```
external\rkwifibt\wifi_ap6xxx_rftest.sh
#!/bin/sh
killall ipc-daemon netserver connmand wpa_supplicant
echo "Pull BT_REG_ON to Low"
echo 0> /sys/class/rfkill/rfkill0/state
echo "Pull WL_REG_ON to Up"
echo 1> /sys/class/rfkill/rfkill1/state
sleep 1
echo "update wifi test fw"
echo /vendor/etc/firmware/fw_bcmdhd_mfg.bin>
/sys/module/bcmdhd/parameters/firmware_path
```

```
sleep 1
ifconfig wlan0 down
ifconfig wlan0 up
sleep 1
echo "wl ver"
wl ver
```

The previous SDK does not have a built-in test firmware. The following takes AP6236 as an example:

```
# Push fw_bcm43436b0_mfg.bin to data or other writable partitions, and then
execute the following command: (note the path below)
mount --bind /data/fw_bcm43436b0_mfg.bin /system/etc/firmware/fw_bcm43436b0.bin
ifconfig wlan0 down
ifconfig wlan0 up
wl ver
```

Normally, executing `wl ver` will print a string of characters with the word WL_TEST in it, indicating that it has entered the test mode. Please refer to the following document for detailed test:

```
Wi-Fi RF Test Commands for Linux-v03.pdf
```

Bluetooth Test

After executing the bt_init.sh script (it is in SDK by default, please refer to Chapter 3.3), execute **(Note: If there is no hciconfig command, please select BR2_PACKAGE_BLUEZ5_UTILS in the Buildroot configuration to build and update the test)**:

```
hciconfig hci0 up
hciconfig -a
```

It does not finish the initialization until hci0 node appears. But when it does not appear, there are two possibilities:

1. The Bluetooth dts configuration is abnormal or the hardware is abnormal or the uart port is configured incorrectly, causing the failed initialization;
2. The Bluetooth firmware file is configured incorrectly or there is no such file;

Please refer to the BT related troubleshooting in Chapter 1or 2;

For detailed test instructions, please refer to:

```
BT RF Test Commands for Linux-v05.pdf #Test 1/2 steps in the document have been
executed in the script, no need to execute)
```

4.3 Report

After confirming the above hardware tests, please output a test report, which should be provided to us when you encounter performance or stability problems.

5. Wi-Fi Performance Test

Pleas test performance by iperf

Pay attention to the following two items that affect performance:

- After finishing Wi-Fi RF test and OTA test of antenna, and make sure that there is no problem with the indicators before testing performance, otherwise it is meaningless;
- If you find the data fluctuates greatly, please go to an open space or basement or other places with little interference to confirm again (it is best to test in a shielded room);

Test environment: **due to the large interference factors in an open environment, it is recommended to test in a shielded room**. First, ensure that Wi-Fi can connect to an AP normally and obtain an IP address;
Test points: the size of throughput rate, and stability, whether there is up and down fluctuations, etc.;
Router channel: choose low, medium, and high channels to test separately, such as 1/6/11 channel:

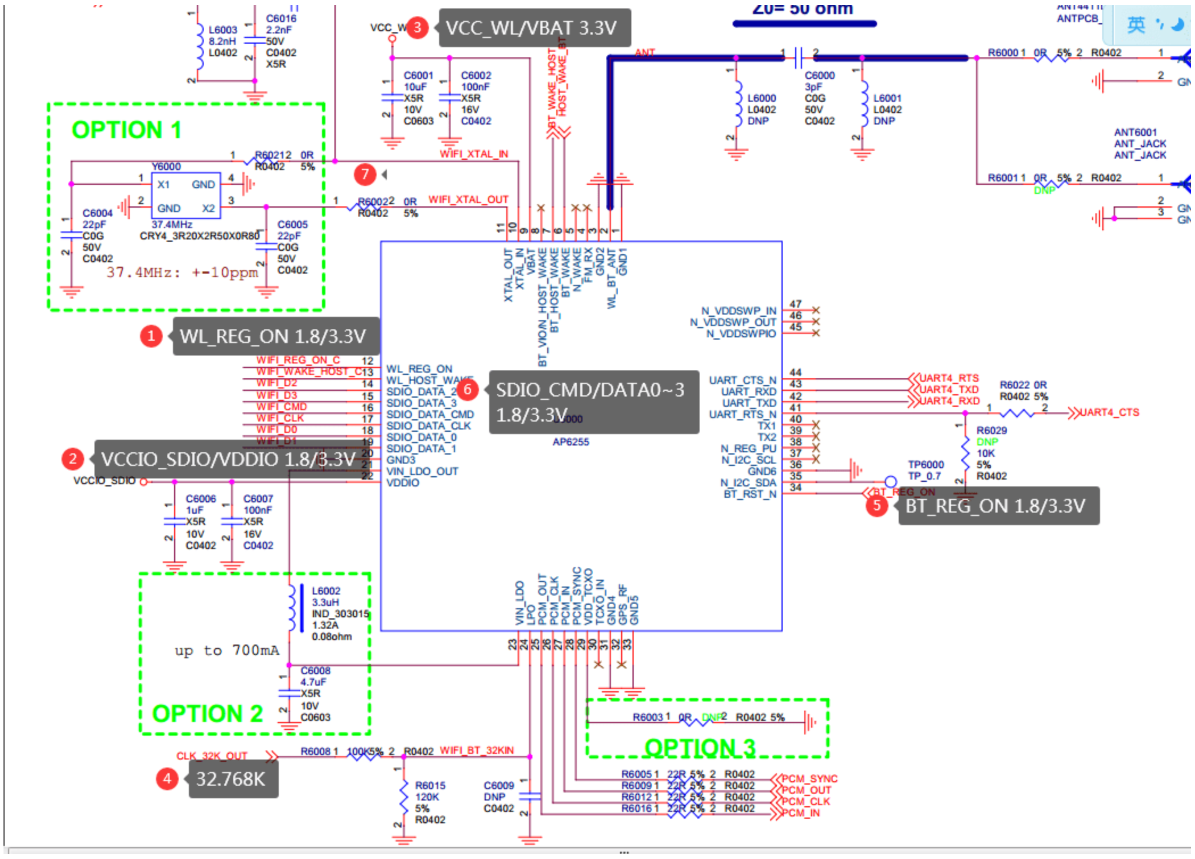
```
# TCP
Down:
    Board: iperf -s -i 1
    Computer: iperf -c xxxxxxx(IP address of the board) -i 1 -w 2M -t 120

Up:
    Compute: iperf -s -i 1
    Board: iperf -c xxxxxxx(IP address of the computer) -i 1 -w 2M -t 120

# UDP
Down:
    Board: iperf -s -u -i 1
    Compute: iperf -c xxxxxxx(IP address of the board) -u -i 1 -b 100M -t 120

Up:
    Compute: iperf -s -u -i 1
    Board: iperf -c xxxxxxx(IP address of the computer) -u -i 1 -b 100M -t 120

# Note: The iperf command of the board should be configured in the Buildroot:
BR2_PACKAGE_IPERF = y
```

6. Wi-Fi/BT Troubleshooting

6.1 Wi-Fi Issues

6.1.1 Wi-Fi Initialization Failed and No wlan0 Node

As shown in the figure below, measure the voltage level of the corresponding pin and whether the clk frequency is correct according to the marked values (**Note: SDIO3.0 mode must be 1.8V**):



1. WL_REG_ON: DTS configuration error, resulting in uncontrollable, **test the waveform by an oscilloscope to see if it is pulled down and pulled up, and whether the voltage amplitude meets the requirements;**
2. WIFI_WAKE_HOST: **PIN** configuration error or **voltage level status** configuration error;
3. SDIO_CMD/SDIO_DATA0~3: **must be 3.3V or 1.8V**;
4. VDDIO_SDIO: **the power supply level is wrong; or the power domain configuration is wrong (please refer to Chapter 1.1 for IO power domain configuration)**;
5. VCC_WL/VBAT/VDDIO_SDIO: **power supply is wrong or the ripple is too large**:

```
[ 6.173685] mmc_host mmc0: Bus speed (slot 0) = 50000000Hz (slot req
50000000Hz, actual 50000000HZ div = 0)
[ 6.173771] rk_sdmmc: BOOT dw_mci_setup_bus: argue clk_mmc workaround out
normal clock [mmc0]
[ 6.175555] rk_sdmmc: BOOT Bus speed=50000000Hz,Bus width=4bits.[mmc0]
[ 6.177736] mmc0: new high speed SDIO card at address 0001
[ 6.179469] RTL871X: ### rtw_hal_ops_check - Error : Please hook
HalFunc.read_chip_version ###
[ 6.179549] RTL871X: ### rtw_hal_ops_check - Error : Please hook
HalFunc.init_default_value ###
[ 6.179593] RTL871X: ### rtw_hal_ops_check - Error : Please hook
HalFunc.intf_chip_configure ###
[ 6.179627] RTL871X: ### rtw_hal_ops_check - Error : Please hook
HalFunc.read_adapter_info ###
[ 6.179666] RTL871X: ### rtw_hal_ops_check - Error : Please hook
HalFunc.hal_power_on ###
[ 6.179702] RTL871X: ### rtw_hal_ops_check - Error : Please hook
HalFunc.hal_power_off ###
```

6. CLK_32K_OUT:

There is no 32.768K waveform, or the waveform amplitude or accuracy is not accurate (0.7 * VDDIO ~ VDDIO);

```
  #the abnormal log of wihout 32k:
  [11.068180] dhdsdio_htclk: HT Avail timeout (1000000): clkctl 0x50
  [11.074372] dhd_bus_init: clock state is wrong. state = 1
  [12.078468] dhdsdio_htclk: HT Avail timeout (1000000): clkctl 0x50
  [12.086051] dhd_net_bus_devreset: dhd_bus_devreset: -1
```

7. SDMMC_CLK/CMD/DATAXXX **PCB layout abnormality or poor connection and poor soldering lead to fail to initialize or run high frequency, you can reduce the frequency appropriately to confirm (modify the max-frequency under the &sdio node)**:

```
  # Abnormal log1, such as the log of abnormal data communication like failure to
download firmware:
  [272.947090] sdioh_buffer_tofrom_bus: TX FAILED ede95000,addr=0x08000,
pkt_len=1968, ERR=-84
  [273.504437] _dhdsdio_download_firmware: dongle image file download failed
  [273.512210] dhd_bus_devreset Failed to download binary to the donglesdio

  # Exception log2
  [9.007297] dwmmc_rockchip 30120000.rksdmmc: Busy; trying anyway
  [9.008401] sdioh_buffer_tofrom_bus: RX FAILED c52ce000,addr=0x0f154,
pkt_len=3752, ERR=-5
  [9.008425] dhdsdio_membytes: membytes transfer failed
  [9.008488] bcmsdh_sdmmc: Failed to Write byte F1:@0x1000a=00, Err: -5
  [9.009523] bcmsdh_sdmmc: Failed to Write byte F1:@0x1000a=00, Err: -5
  [9.010564] bcmsdh_sdmmc: Failed to Write byte F1:@0x1000a=00, Err: -5
  [9.010583] dhdsdio_membytes: FAILED to set window back to 0x18100000

  # Look at the current running frequency of sdio from the log:
  mmc_host mmc1: Bus speed (slot 0) = 148500000Hz (slot req 150000000Hz, actual
148500000HZ div = 0)

  # The way to modify frequency:
  &sdio {
  + max-frequency = <50000000>; # Modify here to limit frequency
```

8. The log shows `sdio host isn't initialization successfully`:

```
  # Two possibilities:
  &sdio node is not open
  The clk attribute in the &sdio_pwrseq node is incorrectly configured
  # The above two errors will cause sdio initialization abnormity
```

9. Modules/chips failure, so you can replace them with new modules or new boards for verification.
10. WAKE UP PIN configuration error or voltage level status configuration error:

```
  [   10.432165] dhd_bus_rxctl: resumed on timeout, INT status=0x208000C0
  [   10.433207] dhd_bus_rxctl: rxcnt_timeout=1, rxlen=0
```

11. There is no firmware. Please troubleshoot configuration or building problems:

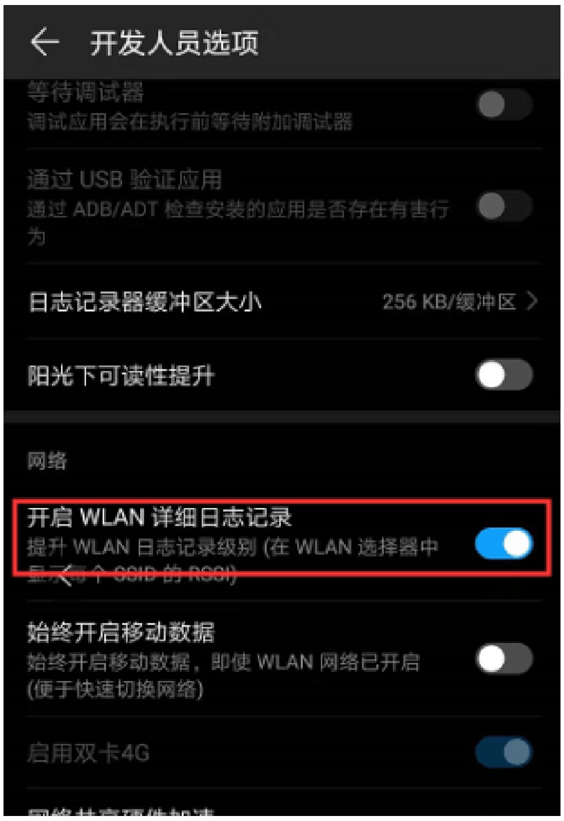
```
   [3.514050] [dhd] dhd_conf_set_path_params : Final
clm_path=/vendor/etc/firmware/clm_bcm43438a1.blob
   [3.514067] [dhd] dhd_conf_set_path_params : Final
conf_path=/vendor/etc/firmware/config.txt
   [3.515918] dhdsdio_download_code_file: Open firmware file failed
/vendor/etc/firmware/fw_bcm43438a1.bin
   [3.515954] _dhdsdio_download_firmware: dongle image file download failed
```

12. If the above troubleshooting can not fix your problem, please upload to our Redmine system: dts/dtsi configuration, kernel complete log dmesg, pdf schematic diagram and other files, as well as waveform diagrams in the following two situations:
    - Power-on waveforms of WIFI_REG_ON and WIFI_CMD when powering on;
    - Power-on waveforms of WIFI_REG_ON and WIFI_CLK when powering on;
    - It is best to capture waveform in three channels at the same time;

6.1.2 Wi-Fi Cannot Connect to Router or Is Disconnected or Unstable

These problems are almost caused by unqualified RF and antenna of WiFi chip or module. The debug method are as follows:

1. First get WiFi RF indicators and OTA test report of antenna of the whole device from hardware engineers to ensure that the hardware indicators are normal (**transmission power, EVM, crystal frequency offset, receiving sensitivity**);
2. Make a basic scanning comparison: put the device to be tested and a mobile phone at the same distance from the router, and **by comparing the numbers of APs scanned and their signal strength with the mobile phone (or competing products of the same specification)** to make sure whether the hardware indicators are normal preliminarily, please refer to chapter 2.1.1 for the scanning method. Here is going to introduce how to compare with the mobile phone (as shown in the figure below). By comparing the numbers of hotspots and signal strength scanned by the mobile phone to check whether the hardware indicators are qualified preliminarily;



开发人员选项

等待调试器
调试应用会在执行前等待附加调试器

通过 USB 验证应用
通过 ADB/ADT 检查安装的应用是否存在有害行为

日志记录器缓冲区大小 256 KB/缓冲区

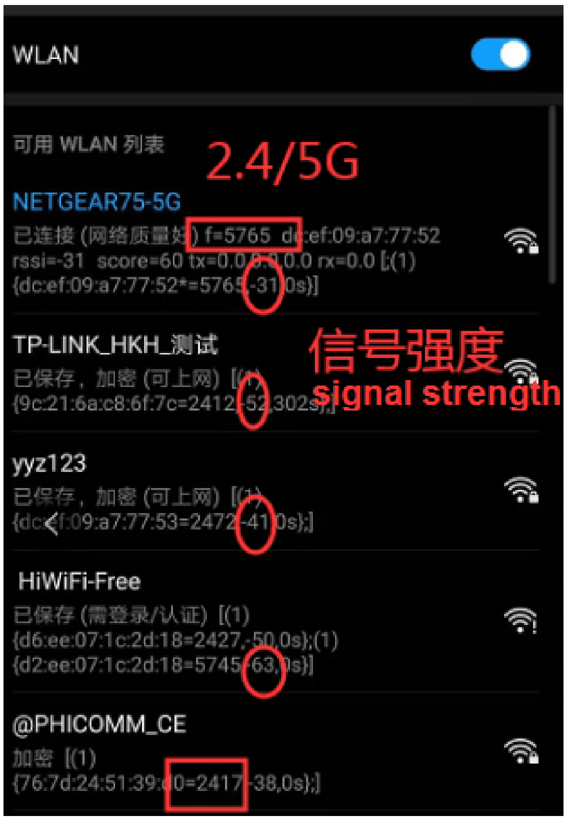
阳光下可读性提升

网络

开启 WLAN 详细日志记录
提升 WLAN 日志记录级别 (在 WLAN 选择器中显示每个 SSID 的 RSSI)

始终开启移动数据
始终开启移动数据，即使 WLAN 网络已开启 (便于快速切换网络)

启用双卡4G



WLAN

可用 WLAN 列表

NETGEAR75-5G
已连接 (网络质量好) f=5765 dc:ef:09:a7:77:52
rssi=-31 score=60 tx=0.0,0.0,0.0 rx=0.0 [:(1)
{dc:ef:09:a7:77:52*=5765,-31,0s)]

TP-LINK_HKH_测试
已保存，加密 (可上网) [(1)
{9c:21:6a:c8:6f:7c=2412,-52,302s};]

yyz123
已保存，加密 (可上网) [(1)
{dc:ef:09:a7:77:53=2472,-41,0s};]

HiWiFi-Free
已保存 (需登录/认证) [(1)
{d6:ee:07:1c:2d:18=2427,-50,0s};(1)
{d2:ee:07:1c:2d:18=5745,-63,0s}]

@PHICOMM_CE
加密 [(1)
{76:7d:24:51:39:d0=2417,-38,0s};]

3. Troubleshoot interference factors. For example, there are many 2.4/5G wireless devices connected at the same time in the current environment, in which interference is very large, you can compare the test device **with a mobile phone (or a competing product of the same specification)**, put them in the same distance, if both are abnormalities, it is interference, if the phone is normal, it can concluded that the device's hardware indicators are abnormal;
4. Troubleshoot the distance factor. The distance is too far to cause weak signal (by scanning wpa_cli scan/scan_r, the signal strength of the connected AP is between -68~-90), which leads to communication failure, you can shorten the distance between the two to confirm;
5. Troubleshoot router compatibility problems, you can replace routers of different manufacturers to confirm;
6. Troubleshoot abnormal board , you can take two or three devices for comparison test;
7. If the above troubleshooting cannot solve the problem, please provide detailed reproduce steps and kernel log (refer to Chapter 5.4);
8. Sometimes the wrong ssid name will cause disconnection, please check whether the ssid configured in wpa_supplicant.conf is correct;
9. If the above troubleshoot are normal:
    - Directly contact module vendors or Wi-Fi manufacturers who get professional packet capture equipment to capture packets for quickly fixing problems;
    - Or send us the board with problems for debugging;

6.1.3 Connect Network or Get IP Address Slowly

Generally, it is caused by RF indicators or antenna indicators cannot meet the requirement. First check Chapter 3 to ensure that the indicators are qualified, and then upload a complete kernel log (dmesg, open the driver debug option according to Chapter 8.9, get the complete log) to Redmine.

6.1.4 Cannot Get IP Aaddress or IP Address Conflict

Please confirm whether the dhcpcd or udhcpc process is enabled;

dhcpcd: is enabled in the SDK by default and starts when the system starts. It is a dhcp client with relatively complete functions;

udhcpcd: is a compact dhcp client of busybox;

Note: These two processes must not be enabled at the same time, only one of them can be used!

6.1.5 System is Resumed Frequently by Wi-Fi after Suspending

The following situations are not included:

1. The 32.768k of the WiFi module is turned off after suspending (suitable for 32.768k is supplied externally);
2. WIFI_WAKE_HOST PIN hardware is unstable/or soldering is unstable, resulting in voltage level unstable (normally, it is low level, and high level trigger);
3. The AP/CY module did not execute the following commands before and after WiFi suspending to filter broadcast or group broadcast packets:

```
#Execute before suspending:
dhd_priv setsuspendmode 1
#Execute after resuming:
dhd_priv setsuspendmode 0
```

6.1.6 Fail to Ping or Delay to Ping with High Probability

1. The high probability is because WiFi is in scanning operation, causing a large ping delay;
2. The router or device is forbidden to ping, you can use other devices for comparison test;

6.1.7 Abnormality Caused by Modifying the Original Wi-Fi Driver Configuration

We often receive some strange issues caused by customers modified some Wi-Fi/BT driver configuration, so please check whether you have modified the original code first. If so, please return and make sure where you modified and the reason for the modification, and send to us through Redmine.

6.1.8 Realtek Wi-Fi SDIO 3.0 Is Abnormal

When using high-end such as RTL8821CS that supports 3.0 modules, fail to initialize with high probability, and the abnormal log is as follows:

```
Line 969: [    1.916211] mmc_host mmc1: Bus speed (slot 0) = 400000Hz (slot req
400000Hz, actual 400000HZ div = 0)
Line 1239: [    1.949247] mmc_host mmc1: Voltage change didn't complete
Line 1294: [    1.951085] mmc1: error -5 whilst initialising SDIO card
```

Please add the following patch:

```
diff --git a/drivers/mmc/core/sdio.c b/drivers/mmc/core/sdio.c
index 2046eff..6626752 100644
--- a/drivers/mmc/core/sdio.c
+++ b/drivers/mmc/core/sdio.c
@@ -646,7 +646,7 @@ static int mmc_sdio_init_card(struct mmc_host *host, u32 ocr,
* try to init uhs card. sdio_read_cccr will take over this task
* to make sure which speed mode should work.
*/
- if (!powered_resume && (rocr & ocr & R4_18V_PRESENT)) {
+ /*if (!powered_resume && (rocr & ocr & R4_18V_PRESENT)) {
        err = mmc_set_uhs_voltage(host, ocr_card);
        if (err == -EAGAIN) {
            mmc_sdio_resend_if_cond(host, card);
@@ -655,7 +655,10 @@ static int mmc_sdio_init_card(struct mmc_host *host, u32
ocr,
        } else if (err) {
            ocr &= ~R4_18V_PRESENT;
        }
- }
+ }*/
+
+ ocr &= R4_18V_PRESENT;
```

```
/*
* For native busses: set card RCA and quit open drain mode
```

6.1.9 Fail to Scan Any AP

1. Check whether the crystal oscillator corresponding to the module is consistent with the requirements of the chip, for example, Wi-Fi requires 24M, but a 37M is connected;
2. Whether the accuracy of the crystal oscillator of 32.768k meets the requirements;

6.1.10 Dual Wi-Fi_AP+RTL Abnormality

Connect two Wi-Fi, one is AP6xxx with sdio interface and the other is RTL8xxxu with USB interface; after the kernel is started, both initializations are normal, but when doing "down" operation of RTLxxxbu module interface, kernel hangs.

```
diff --git a/drivers/net/wireless/rockchip_wlan/rkwifi/bcmdhd/wl_cfg80211.c
b/drivers/net/wireless/rockchip_wlan/rkwifi/bcmdhd/wl_cfg80211.c
index f4838a8..ceb2a00 100644
--- a/drivers/net/wireless/rockchip_wlan/rkwifi/bcmdhd/wl_cfg80211.c
+++ b/drivers/net/wireless/rockchip_wlan/rkwifi/bcmdhd/wl_cfg80211.c
@@ -14640,6 +14640,9 @@ wl_cfg80211_netdev_notifier_call(struct notifier_block *
nb,
 if (!wdev || !cfg || dev == bcmcfg_to_prmry_ndev(cfg))
 return NOTIFY_DONE;

+ if(strncmp(dev->name, "wlan0",strlen("wlan0"))) {
+ return NOTIFY_DONE;
+}
```

6.2 Bluetooth Issues

6.2.1 Realtek's Initialization Failed

1. The dts configuration is incorrect, or hci_uart.ko is not built correctly, or the firmware is not copied correctly. For example, the following files will be used during initialization of RTL8723DS. Please check whether the following files exist:

```
/usr/lib/modules/hci_uart.ko
/lib/firmware/rtlbt/rtl8723d_fw
/lib/firmware/rtlbt/rtl8723d_config
```

2. Many initialization recognition abnormalities are caused by incorrect hardware connection:

```
#For the COB chips using Realtek Bluetooth directly: The hardware connection of
UART interface is as follows:
    //Make sure your UART setting is correct.
    host tx  - controller rx
    host rx  - controller tx
    host rts - controller cts
    host cts-ground #Controller's cts to be grounded

    #for RTL8822C module is special, 4 lines must be connected to the controller
    ost tx-controller rx
    host rx-controller tx
    host rts-controller cts
    host cts-controller rts

#Generally, for modules, the module vendors grounds the controller rts
internally, so it does not need ground on the controller side, just connect it
directly to the controller rts. Note: Realtek has a large number of agents and
each module may be different, so please confirm with the module manufacturer. If
the controller rts is not grounded, grounding on the controller side is required.
    host tx-controller rx
    host rx-controller tx
    host rts-controller cts
    host cts-controller rts
```

3. The firmware file is wrong or there is no corresponding files at all. It is also caused by the compilation configuration problem like above. The following are the firmware names and paths corresponding to some models. Note that the USB firmware path is special.

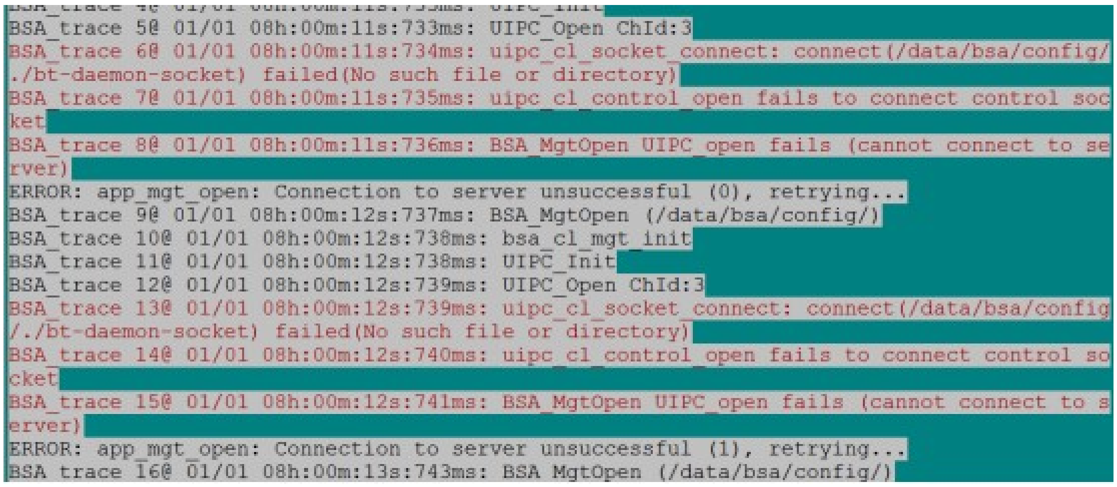
Chip	I/F for BT driver	FW/Config Path	FW Filename	Config Filename
RTL8723DS	UART	/lib/firmware/rtlbt/	rtl8723d_fw	rtl8723d_config
RTL8821CS	UART	/lib/firmware/rtlbt/	rtl8821c_fw	rtl8821c_config
RTL8821CU	USB	/lib/firmware/	rtl8821cu_fw	rtl8821cu_config

Please carefully refer to the description of Bluetooth and Building update in Chapter 1/2 for troubleshooting!

6.2.2 Abnormal to Develop Bluetooth by deviceio

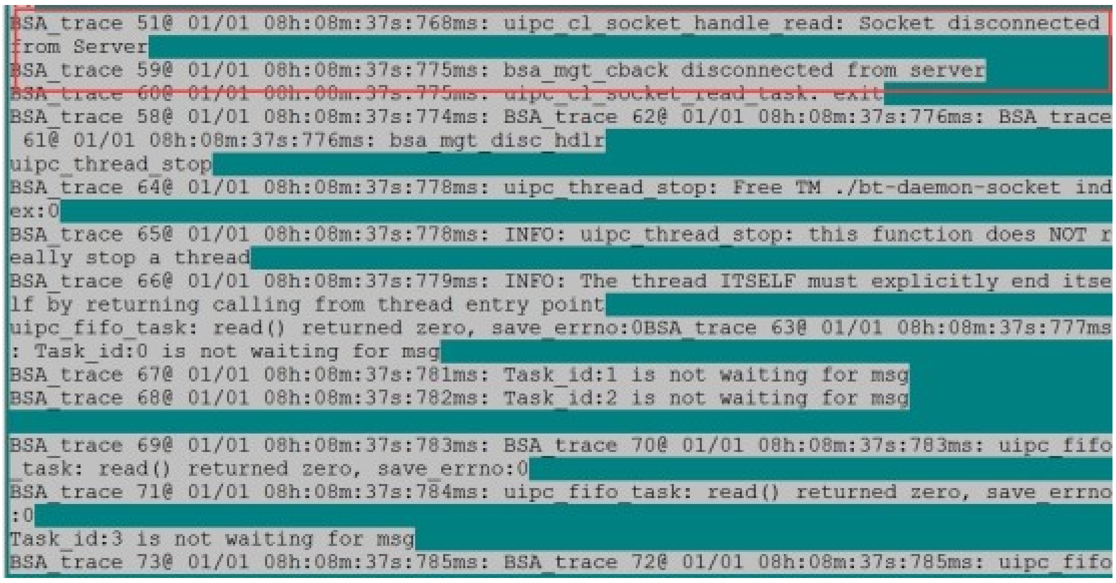
1. Bluetooth initialization failed and always print the following log:

```
'Uipc_cl_socket_connect: connect(/data/bsa/config/./bt-daemon-socket) failed(No
such file or directory)'
```



```
BSA_trace 4@ 01/01 08h:00m:11s:733ms: UIPC_Init
BSA_trace 5@ 01/01 08h:00m:11s:733ms: UIPC_Open ChId:3
BSA_trace 6@ 01/01 08h:00m:11s:734ms: uipc_cl_socket_connect: connect(/data/bsa/config/
./bt-daemon-socket) failed(No such file or directory)
BSA_trace 7@ 01/01 08h:00m:11s:735ms: uipc_cl_control_open fails to connect control soc
ket
BSA_trace 8@ 01/01 08h:00m:11s:736ms: BSA_MgtOpen UIPC_open fails (cannot connect to se
rver)
ERROR: app_mgt_open: Connection to server unsuccessful (0), retrying...
BSA_trace 9@ 01/01 08h:00m:12s:737ms: BSA_MgtOpen (/data/bsa/config/)
BSA_trace 10@ 01/01 08h:00m:12s:738ms: bsa_cl_mgt_init
BSA_trace 11@ 01/01 08h:00m:12s:738ms: UIPC_Init
BSA_trace 12@ 01/01 08h:00m:12s:739ms: UIPC_Open ChId:3
BSA_trace 13@ 01/01 08h:00m:12s:739ms: uipc_cl_socket_connect: connect(/data/bsa/config
/./bt-daemon-socket) failed(No such file or directory)
BSA_trace 14@ 01/01 08h:00m:12s:740ms: uipc_cl_control_open fails to connect control so
cket
BSA_trace 15@ 01/01 08h:00m:12s:741ms: BSA_MgtOpen UIPC_open fails (cannot connect to s
erver)
ERROR: app_mgt_open: Connection to server unsuccessful (1), retrying...
BSA_trace 16@ 01/01 08h:00m:13s:743ms: BSA_MgtOpen (/data/bsa/config/)
```

2. Bluetooth is unstable. After Bluetooth is successfully started, the following disconnect form server log appears suddenly after being placed for a period of time or during operation:



```
BSA_trace 51@ 01/01 08h:08m:37s:768ms: uipc_cl_socket_handle_read: Socket disconnected
from Server
BSA_trace 59@ 01/01 08h:08m:37s:775ms: bsa_mgt_cback disconnected from server
BSA_trace 60@ 01/01 08h:08m:37s:775ms: uipc_cl_socket_read_task: exit
BSA_trace 58@ 01/01 08h:08m:37s:774ms: BSA_trace 62@ 01/01 08h:08m:37s:776ms: BSA_trace
 61@ 01/01 08h:08m:37s:776ms: bsa_mgt_disc_hdlr
uipc_thread_stop
BSA_trace 64@ 01/01 08h:08m:37s:778ms: uipc_thread_stop: Free TM ./bt-daemon-socket ind
ex:0
BSA_trace 65@ 01/01 08h:08m:37s:778ms: INFO: uipc_thread_stop: this function does NOT r
eally stop a thread
BSA_trace 66@ 01/01 08h:08m:37s:779ms: INFO: The thread ITSELF must explicitly end itse
lf by returning calling from thread entry point
uipc_fifo_task: read() returned zero, save_errno:0BSA_trace 63@ 01/01 08h:08m:37s:777ms
: Task_id:0 is not waiting for msg
BSA_trace 67@ 01/01 08h:08m:37s:781ms: Task_id:1 is not waiting for msg
BSA_trace 68@ 01/01 08h:08m:37s:782ms: Task_id:2 is not waiting for msg

BSA_trace 69@ 01/01 08h:08m:37s:783ms: BSA_trace 70@ 01/01 08h:08m:37s:783ms: uipc_fifo
_task: read() returned zero, save_errno:0
BSA_trace 71@ 01/01 08h:08m:37s:784ms: uipc_fifo_task: read() returned zero, save_errno
:0
Task_id:3 is not waiting for msg
BSA_trace 73@ 01/01 08h:08m:37s:785ms: BSA_trace 72@ 01/01 08h:08m:37s:785ms: uipc_fifo
```

3. The Bluetooth initialization of Realtek modules failed:

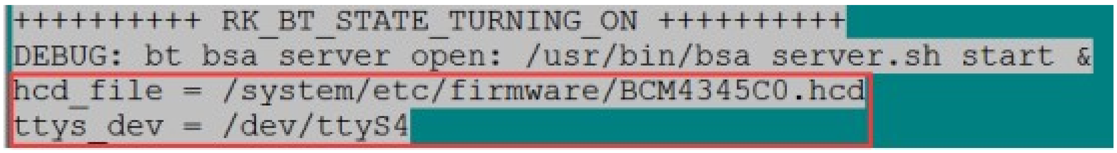
```
"Can't open serial port, 2, No such file or directory" log
```

4. Show the following log:

```
[BT_OPEN] bt_init.sh not exist !!!
```

There are only following possible reasons:

1. The rkwifibt module model or serial port configuration is incorrect. The firmware name and ttySx will be printed when the log is initialized. Please refer to the description of Bluetooth and building in Chapter 1/2 for troubleshooting;



```
++++++++++ RK_BT_STATE_TURNING_ON ++++++++++
DEBUG: bt bsa server open: /usr/bin/bsa server.sh start &
hcd_file = /system/etc/firmware/BCM4345C0.hcd
ttys_dev = /dev/ttyS4
```

2. Before using deviceio bt init, the bt_init.sh script started brcm_patchram_plus1/rtk_hciattach, first kill these processes before using deviceio: `killall brcm_patchram_plus1 killall rtk_hciattach`;
3. bsa building configuration is error: AMPAK module is used to build cypress_bsa; or AzureWave module is used to build broadcom_bsa.

7. New Module Porting or Old Module Driver Update

Pay attention to:

- **It is strongly recommended to read and understand the files storage rules and building rules in Chapter 1.5, which is very important for porting. Please make sure you are familiar with the building rules;**
- **The SDK obtained by customers may be outdated, so the rules of Config.in and rkwifibt.mk may be updated but the principles are the same;**
- **For updating driver case, you can skip some of the following steps accordingly.**

7.1 Realtek Modules

7.1.1 Wi-Fi Modules

Take RTL8821CS as an example:

Get the corresponding porting package from module vendors or the manufacturers:

```
20171225_RTL8821CS_WiFi_linux_v5.2.18_25830_BT_ANDROID_UART_COEX_8821CS-
B1d1d_COEX20170908-1f1f
```

For Wi-Fi driver part , go to the following directory:

```
WiFi\RTL8821CS_WiFi_linux_v5.2.18_25830_COEX20170908-1f1f.20171225\driver
```

- Modify Makefile

```
#Change to RK platform:
CONFIG_PLATFORM_I386_PC = n
CONFIG_PLATFORM_ARM_RK3188 = y

#The following configuration needs to be removed on RK platform:
ifeq ($(CONFIG_PLATFORM_ARM_RK3188), y)
-EXTRA_CFLAGS += -DRTW_SUPPORT_PLATFORM_SHUTDOWN #remove this configuration
MODULE_NAME := 8821cs #Modify ko's name

#If there is a need for WiFi sleep to keep the connection, open the following
configuration
CONFIG_WOWLAN = y
CONFIG_GPIO_WAKEUP = y
```

- Add WAKE UP PIN configuration

```
#Modify platform\platform_ops.c
+#include <linux/rfkill-wlan.h>
+extern unsigned int oob_irq;
int platform_wifi_power_on(void)
{
    int ret = 0;

+   oob_irq = rockchip_wifi_get_oob_irq();  //corresponding to WIFI_WAKE_HOST PIN
of dts

    return ret;
}
```

- Customize MAC address requirements

```
# Modification: core\rtw_ieee80211.c
# include <linux/rfkill-wlan.h> //Add header file
# Find rtw_macaddr_cfg function
void rtw_macaddr_cfg(u8 *out, const u8 *hw_mac_addr)
-/* Use the mac address stored in the Efuse */
-if (hw_mac_addr) {
-   rtw_memcpy(mac, hw_mac_addr, ETH_ALEN);
-   goto err_chk;
-}

+ /* Use the mac address stored in the Efuse */
+ if (hw_mac_addr) {
+   _rtw_memcpy(mac, hw_mac_addr, ETH_ALEN);
+ }

+ if (!rockchip_wifi_mac_addr(mac)) {
+   printk("get mac address from flash=[%02x:%02x:%02x:%02x:%02x:%02x]\n",
mac[0], mac[1],
+       mac[2], mac[3], mac[4], mac[5]);
+   }
+ }
```

- Add driver loading entrance

```
//os_dep\linux\sdio_intf.c
//Add the following code at the end of the file:
#include "rtw_version.h"
#include <linux/rfkill-wlan.h>
extern int get_wifi_chip_type(void);
extern int rockchip_wifi_power(int on);
extern int rockchip_wifi_set_carddetect(int val);

int rockchip_wifi_init_module_rtkwifi(void)
{
#ifdef CONFIG_WIFI_LOAD_DRIVER_WHEN_KERNEL_BOOTUP
    int type = get_wifi_chip_type();
    if (type < WIFI_AP6XXX_SERIES || type == WIFI_ESP8089) return 0;
#endif
    printk("\n");
    printk("=======================================================\n");
    printk("==== Launching Wi-Fi driver! (Powered by Rockchip) ====\n");
    printk("=======================================================\n");
    printk("Realtek 8XXX SDIO WiFi driver (Powered by Rockchip,Ver %s) init.\n",
DRIVERVERSION);

    rockchip_wifi_power(1);
    rockchip_wifi_set_carddetect(1);

    return rtw_drv_entry();
}

void rockchip_wifi_exit_module_rtkwifi(void)
{
#ifdef CONFIG_WIFI_LOAD_DRIVER_WHEN_KERNEL_BOOTUP
    int type = get_wifi_chip_type();
    if (type < WIFI_AP6XXX_SERIES || type == WIFI_ESP8089) return;
#endif
    printk("\n");
    printk("=======================================================\n");
    printk("==== Dislaunching Wi-Fi driver! (Powered by Rockchip) ====\n");
    printk("=======================================================\n");
    printk("Realtek 8XXX SDIO WiFi driver (Powered by Rockchip,Ver %s) init.\n",
DRIVERVERSION);

    rtw_drv_halt();

    rockchip_wifi_set_carddetect(0);
    rockchip_wifi_power(0);

}

#ifdef CONFIG_WIFI_BUILD_MODULE
module_init(rockchip_wifi_init_module_rtkwifi);
module_exit(rockchip_wifi_exit_module_rtkwifi);
#else
#ifdef CONFIG_WIFI_LOAD_DRIVER_WHEN_KERNEL_BOOTUP
late_initcall(rockchip_wifi_init_module_rtkwifi);
module_exit(rockchip_wifi_exit_module_rtkwifi);
#else
EXPORT_SYMBOL(rockchip_wifi_init_module_rtkwifi);
EXPORT_SYMBOL(rockchip_wifi_exit_module_rtkwifi);
#endif
#endif

//comment out the followings, pay attention to delete the entry __init __exit of
the following two functions
//module_init(rtw_drv_entry);
//module_exit(rtw_drv_halt);
```

- Add the corresponding compilation configuration (this operation is not necessary for updating driver case)

```
drivers/net/wireless/rockchip_wlan/Makefile
+ obj-$(CONFIG_RTL8821CS) += rtl8821cs/

drivers/net/wireless/rockchip_wlan/Kconfig
+ source "drivers/net/wireless/rockchip_wlan/rtl8821cs/Kconfig"
```

- Add the identification function of models (this operation is not necessary in updating driver case)

```
diff --git a/drivers/net/wireless/rockchip_wlan/wifi_sys/rkwifi_sys_iface.c
b/drivers/net/wireless/rockchip_wlan/wifi_sys/rkwifi_sys_iface.c
index 88db4de..2e3679a 100755
--- a/drivers/net/wireless/rockchip_wlan/wifi_sys/rkwifi_sys_iface.c
+++ b/drivers/net/wireless/rockchip_wlan/wifi_sys/rkwifi_sys_iface.c
@@ -133,6 +133,11 @@ static ssize_t wifi_chip_read(struct class *cls, struct
class_attribute *attr, c

+    if (type == WIFI_RTL8821CS) {
+         count = sprintf(_buf, "%s", "RTL8821CS");
+         printk("Current WiFi chip is RTL8821CS.\n");
+    }
+
    if(type == WIFI_ESP8089) {
        count = sprintf(_buf, "%s", "ESP8089");
        printk("Current WiFi chip is ESP8089.\n");
diff --git a/include/linux/rfkill-wlan.h b/include/linux/rfkill-wlan.h
index 4218b84..698b685 100755
--- a/include/linux/rfkill-wlan.h
+++ b/include/linux/rfkill-wlan.h
@@ -73,6 +73,7 @@ enum {
    WIFI_RTL8189ES,
    WIFI_RTL8189FS,
    WIFI_RTL8812AU,
+   WIFI_RTL8821CS,
    WIFI_RTL_SERIES,
    WIFI_ESP8089,
    WIFI_MVL88W8977,
diff --git a/net/rfkill/rfkill-wlan.c b/net/rfkill/rfkill-wlan.c
index a17810d..7bbce01 100755
--- a/net/rfkill/rfkill-wlan.c
+++ b/net/rfkill/rfkill-wlan.c
@@ -156,6 +156,8 @@ int get_wifi_chip_type(void)
        type = WIFI_RTL8189FS;
    } else if (strcmp(wifi_chip_type_string, "rtl8812au") == 0) {
        type = WIFI_RTL8812AU;
+   } else if (strcmp(wifi_chip_type_string, "rtl8821cs") == 0) {
+       type = WIFI_RTL8821CS;
    } else if (strcmp(wifi_chip_type_string, "esp8089") == 0) {
        type = WIFI_ESP8089;
    } else if (strcmp(wifi_chip_type_string, "mvl88w8977") == 0) {
```

- Deal with the mostly likely errors:

```
rtl8xxx\os_dep\linux\rtw_android.c //Comment out the following two command
#if (LINUX_VERSION_CODE >= KERNEL_VERSION(2, 6, 39)) ||
defined(COMPAT_KERNEL_RELEASE)
void *wifi_get_country_code(char *ccode)
{
    RTW_INFO("%s\n", __FUNCTION__);
    if (!ccode)
        return NULL;
-   if (wifi_control_data && wifi_control_data->get_country_code)
-       return wifi_control_data->get_country_code(ccode);
    return NULL;
}
#endif /* (LINUX_VERSION_CODE >= KERNEL_VERSION(2, 6, 39)) */
```

For Buildroot part: (This operation is not necessary in updating driver case)

```
buildroot\package\rockchip\rkwifibt\Config.in
Add the following in turn:
+config BR2_PACKAGE_RKWIFIBT_RTL8821CS
+ bool "RTL8821CS"

config BR2_PACKAGE_RKWIFIBT_CHIPNAME
+ default "RTL8821CS" if BR2_PACKAGE_RKWIFIBT_RTL8821CS

config BR2_PACKAGE_RKWIFIBT_VENDOR
+ default "REALTEK" if BR2_PACKAGE_RKWIFIBT_RTL8821CS

config BR2_PACKAGE_RKWIFIBT_WIFI_KO
+ default "8821cs.ko" if BR2_PACKAGE_RKWIFIBT_RTL8821CS #The ko name here
corresponds to the one in the Makefile
```

7.1.2 BT Modules

7.1.2.1 UART Interface

If modules support Bluetooth, you have to ask the vendors to provide a software package similar to the following:

```
Linux_BT_UART_v3.10_20181226_8723DS_BTCOEX_20181226-slave
```

Take RTL8723DS as an example (you only need to replace the corresponding file in updating driver case):

```
#There are Bluetooth firmware and configuration files in the 8723D folder under
the directory(rtl8723d_config / rtl8723d_fw), place them in the following
directory:
ls
external/rkwifibt/realtek/
#The structure is as follows:
external/rkwifibt/realtek/RTL8723DS$ tree
├── mp_rtl8723d_config #The file beginning with mp is the Bluetooth RF test file,
which can be obtained directly from module vendors, otherwise the Bluetooth RF
index cannot be tested
├── mp_rtl8723d_fw
├── rtl8723d_config
└── rtl8723d_fw

#Note: The rules for renaming the RTL8723DS directory command should be
consistent with the configuration in the Config.in of WiFi above:
BR2_PACKAGE_RKWIFIBT_RTL8723DS
```

For Buildroot part (no need to do this in updating driver case)

```
config BR2_PACKAGE_RKWIFIBT_BT_EN
+ default "ENABLE" if BR2_PACKAGE_RKWIFIBT_RTL8723DS
```

7.1.2.2 USB Interface

If the Bluetooth with USB interface, the vendor will provide a porting package similar to the following::

```
tree
Linux_BT_USB_v3.10_20190430_8821CU_BTCOEX_20190509-4139
|-8821CU //firmware file
|-bluetooth_usb_driver //usb driver
```

Add the building ko option and install command in rkwfiibt.mk, as follows:

```
+++ b/package/rockchip/rkwifibt/rkwifibt.mk
@@ -73,6 +73,7 @@ define RKWIFIBT_REALTEK_BT_INSTALL
    $(INSTALL) -D -m 0644 $(@D)/realtek/$(BR2_PACKAGE_RKWIFIBT_CHIPNAME)/mp_*
$(TARGET_DIR)/lib/firmware/
    $(INSTALL) -D -m 0755 $(@D)/bt_realtek* $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 0644 $(@D)/realtek/bluetooth_uart_driver/hci_uart.ko
$(TARGET_DIR)/usr/lib/modules/hci_uart.ko
+     $(INSTALL) -D -m 0644 $(@D)/realtek/bluetooth_usb_driver/rtk_btusb.ko
$(TARGET_DIR)/usr/lib/modules/rtk_btusb.ko
    $(INSTALL) -D -m 0755 $(@D)/bt_load_rtk_firmware $(TARGET_DIR)/usr/bin/
    $(SED) 's/BT_TTY_DEV/\/dev\/$(BT_TTY_DEV)/g'
$(TARGET_DIR)/usr/bin/bt_load_rtk_firmware
    $(INSTALL) -D -m 0755 $(TARGET_DIR)/usr/bin/bt_load_rtk_firmware
$(TARGET_DIR)/usr/bin/bt_pcba_test
@@ -92,8 +93,10 @@ define RKWIFIBT_BUILD_CMDS
    $(TARGET_CC) -o $(@D)/brcm_tools/brcm_patchram_plus1
$(@D)/brcm_tools/brcm_patchram_plus1.c
    $(TARGET_CC) -o $(@D)/brcm_tools/dhd_priv $(@D)/brcm_tools/dhd_priv.c
    $(TARGET_CC) -o $(@D)/src/rk_wifi_init $(@D)/src/rk_wifi_init.c
    $(MAKE) -C $(@D)/realtek/rtk_hciattach/ CC=$(TARGET_CC)
    $(TARGET_CONFIGURE_OPTS) $(MAKE) -C $(TOPDIR)/../kernel/
M=$(@D)/realtek/bluetooth_uart_driver ARCH=$(RK_ARCH)
+    $(TARGET_CONFIGURE_OPTS) $(MAKE) -C $(TOPDIR)/../kernel/
M=$(@D)/realtek/bluetooth_usb_driver ARCH=$(RK_ARCH)
```

7.2 AMPAK Modules

Take AP6256 as an example, obtain the Wi-Fi and BT firmware file package of AP6256 from the module vendors (if it is updating driver case, just replace the corresponding file):

```
external\rkwifibt\firmware\broadcom\
//Create a folder named AP6256 in this directory, and store the files inside
according to the following structure
external/rkwifibt/firmware/broadcom/AP6256$ tree
├── bt
│   └── BCM4345C5.hcd
└── wifi
    ├── fw_bcm43456c5_ag.bin
    ├── fw_bcm43456c5_ag_mfg.bin
    └── nvram_ap6256.txt
//Pay attention to the directory name of AP6256, which should be consistent with
the configuration in the following Config.in of WiFi: BR2_PACKAGE_RKWIFIBT_AP6256
```

Buildroot configuration (no need to do this in updating driver case):

```
buildroot\package\rockchip\rkwifibt\Config.in
Add the followings in turn:
+config BR2_PACKAGE_RKWIFIBT_AP6256
+   bool "AP6256"

config BR2_PACKAGE_RKWIFIBT_CHIPNAME
+   default "AP6256" if BR2_PACKAGE_RKWIFIBT_AP6256

config BR2_PACKAGE_RKWIFIBT_VENDOR
+   default "BROADCOM" if BR2_PACKAGE_RKWIFIBT_AP6256

config BR2_PACKAGE_RKWIFIBT_WIFI_KO
+   default "bcmdhd.ko" if BR2_PACKAGE_RKWIFIBT_AP6256

config BR2_PACKAGE_RKWIFIBT_BT_FW
+   default "BCM4345C5.hcd" if BR2_PACKAGE_RKWIFIBT_AP6256  //pay attention to
this name of "BCM4345C5.hcd" , change it to the corresponding model

config BR2_PACKAGE_RKWIFIBT_BT_EN
+   default "ENABLE" if BR2_PACKAGE_RKWIFIBT_AP6256
```

There is no need to change the kernel driver part, and it is basically compatible with all AP modules. the configuration of CONFIG_AP6XXX can be used by default.

8. Other Functions and Configurations

8.1 RV1109 / connmand

RV1109/1126 platform uses connman to manage WiFi by default, and the start way of the core process wpa_supplicant of Wi-Fi is started by:

```
# ps
//You will see the following two processes
conmand //It uses dbus to communicate with wpa_supplicant
wpa_supplicant -u //Open support for dbus communication
```

Standard usage: Wi-Fi can be operated through RV1109 web interface, please refer to the related documents of the RV1109/RV1126 platform;

The simple way to test terminal is as follows:

```
/ # killall ipc-daemon netserver
/ # connmanctl
connmanctl> enable wifi
connmanctl> scan wifi #Can scan multiple times
connmanctl> scan wifi #Can scan multiple times
connmanctl> agent on
connmanctl> services  #List the scanned wifi list
connmanctl>             *AO yyz123
wifi_c0847daf6f42_79797a313233_managed_psk
    NETGEAR75-5G         wifi_c0847daf6f42_4e45544745415237352d3547_managed_psk
    aaabbb               wifi_c0847daf6f42_616161626262_managed_psk
     HiWiFi-Free         wifi_c0847daf6f42_204869576946692d46726565_managed_none
    Fang-HiWiFi          wifi_c0847daf6f42_46616e672d486957694669_managed_psk
    yyz123               wifi_c0847daf6f42_79797a313233_managed_psk

connmanctl> connect wifi_c0847daf6f42_79797a313233_managed_psk #If you want to
connect to yyz123 above, the connect parameter is the following wifixxx_psk
connmanctl> Connected wifi_c0847daf6f42_79797a313233_managed_psk #If the
connection is successful, there will be this print
connmanctl> quit #Exit connection mode
/ # ifconfig wlan0 #You will see the IP address
```

If you want to use the traditional wpa_supplicant/wpa_cli method instead of connman, then remove the connman configuration in Buildroot part:

```
BR2_PACKAGE_CONNMAN
```

And delete the related files generated before:

```
buildroot/output/rockchip_rv1126_rv1109/target/etc/init.d/S45connman
buildroot/output/rockchip_rv1126_rv1109/target/usr/bin/connmanctl
buildroot/output/rockchip_rv1126_rv1109/target/usr/sbin/connmand
```

8.2 Set Static IP and Other Parameters at Boot Automatically

```
//The standard Linux ifupdown command is used in Buildroot system  by default,
the corresponding boot script is in the following directory:
Buildroot/package/ifupdown-scripts
> S40network //Started by /etc/init.d/rcS
//But S40network will call the ifup command to read the default configuration
script: /etc/network/interfaces
//So the default configuration can be written into the /etc/network/interfaces
file, and the way to modify the interfaces file is as follows:
Buildroot/package/ifupdown-scripts/ifupdown-scripts.mk
define IFUPDOWN_SCRIPTS_LOCALHOST
(\
    echo "# interface file auto-generated by buildroot"; \
    echo ; \
    echo "auto lo"; \
    echo "iface lo inet loopback"; \
    echo "auto wlan0"; \
    echo "iface wlan0 inet static"; \
    echo "address 192.168.1.111"; \
    echo "gateway 192.168.1.1"; \
    echo "netmask 255.255.255.0"; \
    echo "broadcast 192.168.1.0"; \
)> $(TARGET_DIR)/etc/network/interfaces // when more configuration needs to be
added, please check the related materials by yourself, which are all linux
standard
endef
//Build upgrade: make ifupdown-scripts-dirclean && make ifupdown-scripts-rebuild

//For the modification of the default dns, Buildroot system does not have a pre-
build configuration. You have to add it manually when the DHCP process is not
running:
echo 'nameserver 114.114.114.114' >> /etc/resolv.conf // Add customized dns
configuration

//Dynamic setting
//Set IP address
ifconfig wlan0 xx.xx.xx.xx netmask xx.xx.xx.xx
//Set the default route
route add default gw xx.xx.xx.xx
//Add dns
echo 'nameserver xx.xx.xx.xx' > /etc/resolv.conf
```

8.3 DHCP Client

dhcpcd: it is used in the SDK by default and starts when the system is started. It is a dhcp client with relatively complete functions;

udhcpcd: is a compact dhcp client of busybox;

Note: these two processes must not be enabled at the same time, only one of them can be used!

If you need to get IP addresses faster when using dhcpcd client, modify as follows:

```
//Modify dhcpcd.conf
buildroot/package/dhcpcd/
--- a/src/dhcpcd.conf
+++ b/src/dhcpcd.conf
@@ -8,12 +8,12 @@
 hostname

 # Use the hardware address of the interface for the Client ID.
-#clientid
+clientid
 # or
 # Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID as per RFC4361.
 # Some non-RFC compliant DHCP servers do not reply with this set.
 # In this case, comment out duid and enable clientid above.
-duid
+# duid

 # Persist interface configuration when dhcpcd exits.
 persistent

//Modify the S41dhcpcd file
index a2e87ca054..f8b924ab0f 100755
/buildroot/package/dhcpcd/S41dhcpcd
@@ -13,7 +13,7 @@ PIDFILE=/var/run/dhcpcd.pid
 case "$1" in
   start)
        echo "Starting dhcpcd..."
-       start-stop-daemon -S -x "$DAEMON" -p "$PIDFILE" -- -f "$CONFIG"
+       start-stop-daemon -S -x "$DAEMON" -p "$PIDFILE" -- -AL -f "$CONFIG"
        ;;

//Repackage a firmware
make dhcpcd-dirclean
make dhcpcd-rebuild
```

8.4 Wi-Fi/BT MAC Address

Generally, Wi-Fi's MAC address is built-in in the chip. If you need to customize the MAC address, you need to use RK special tool to write to customize vendor partition of Flash (Please refer to the related documents of vendor storage operation for details, we won't go into much detail here.);

AzureWave/AMPAK Wi-Fi Modules

Modify the Makefile and add the following configuration:

```
+   -DGET_CUSTOM_MAC_ENABLE

AMPAK: drivers/net/wireless/rockchip_wlan/rkwifi/bcmdhd/Makefile
AzureWave: drivers/net/wireless/rockchip_wlan/cywdhd/bcmdhd/Makefile
```

Note: **AMPAK modules need additional modification of WiFi to work normally. The first 3 bytes of the MAC address are called OUI. Each OUI corresponds to a group of macpad. If you want to modify the OUI, you need to apply to AMPAK for corresponding macpad**, then modify as follows:

```
drivers/net/wireless/rockchip_wlan/rkwifi/bcmdhd/dhd_gpio.c
static int dhd_wlan_get_mac_addr(unsigned char *buf)
{
    int err = 0;
    printf("======== %s ========\n", __FUNCTION__);
#ifdef EXAMPLE_GET_MAC
    /* EXAMPLE code */
    {
    struct ether_addr ea_example = {{0x00, 0x11, 0x22, 0x33, 0x44, 0xFF}};
    bcopy((char *)&ea_example, buf, sizeof(struct ether_addr));
    }
#endif /* EXAMPLE_GET_MAC */

    //Method 1: If we use our vendor solution and flash a custom MAC to the
vendor partition, the following function will be read from the vendor partition.
    err = rockchip_wifi_mac_addr(buf);
    //The function is used to fill the MAC address into the first 6 positions of
buf, please refer to the above ea_example.

    // Method 2: If the MAC address is stored in your customized place, you need
to implement the read function by yourself
    // TODO: Fill the MAC address into the first 6 positions of buf, please efer
to ea_example above

    //#ifdef EXAMPLE_GET_MAC_VER2 //Define or comment out this macro to make the
following code effective
/* EXAMPLE code */
    {
        char macpad[56]= {//Replace with the macpad provided by the vendors
                0x43,0xdf,0x6c,0xb3,0x06,0x3e,0x8e,0x94,
                0xc7,0xa9,0xd3,0x41,0xc8,0x6f,0xef,0x67,
                0x05,0x30,0xf1,0xeb,0x4b,0xa9,0x0a,0x05,
                0x41,0x73,0xbc,0x8c,0x30,0xe5,0x74,0xc6,
                0x88,0x36,0xad,0x0c,0x34,0x7d,0x5b,0x60,
                0xe7,0xd7,0x98,0x64,0xd0,0xfa,0xe3,0x83,
                0x76,0x35,0x1a,0xc8,0x2b,0x0b,0x65,0xb1};
        bcopy(macpad, buf+6, sizeof(macpad));
    }
    //#endif /* EXAMPLE_GET_MAC_VER2 */
    return err;
}
```

Realtek Modules

It will read the customize MAC address from our vendor storage partition when Realtek driver is loading. Please refer to the customize MAC in Chapter 7.1 for the code.

8.5 AMPAK Module Compatible Version (Debian/Ubuntu)

A compatible configuration Buildroot rkwifibt is provided in the SDK for regular AMPAK chips:

```
BR2_PACKAGE_RKWIFIBT_AMPAKALL
```

This configuration can be compatible with the modules supported by SDK. It will automatically detect the Wi-Fi/BT chip model and load the corresponding firmware when it is turned on. The source code directory is:

```
external/rkwifibt/src/rk_wifi_init.c
```

The following points should be noted:

- This configuration will copy the firmware of all AMPAK modules and cause the rootfs file system to become larger, a larger flash is needed;
- This configuration does not support BSA (AMPAK Bluetooth proprietary protocol stack) compatibility, resulting in BSA unavailable;

So only Debian/Ubuntu systems are recommended to use this configuration, these open source systems have their own Bluetooth protocol stack, we only need to do the initialization (rk_wifi_init).

8.6 Modify Realtek Wi-Fi Scan Time

```
//Realtek wifi scan during each channel, modify include/rtw_mlme_ext.h
#define SURVEY_TO                     (100)  //The unit is ms, modify as required
```

8.7 Realtek Country Code

Modify the Makefile of driver and add the following items in platform building:

```
+++ b/drivers/net/wireless/rockchip_wlan/rtlxxx/Makefile
@@ -1270,6 +1270,7 @@ EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -
DCONFIG_PLATFORM_ANDROID -DCONFIG_PLATFO
 # default setting for Android 4.1, 4.2, 4.3, 4.4
 EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
 EXTRA_CFLAGS += -DCONFIG_CONCURRENT_MODE
+EXTRA_CFLAGS += -DCONFIG_RTW_IOCTL_SET_COUNTRY
 # default setting for Power control
 #EXTRA_CFLAGS += -DRTW_ENABLE_WIFI_CONTROL_FUNC
 #EXTRA_CFLAGS += -DRTW_SUPPORT_PLATFORM_SHUTDOWN
```

In this way, rtw_set_country_cmd can be implemented by the following two ioctl methods (choose one only):

1. By the way of proc `echo X> /proc/net/rtlxxx/wlan0/country_code`, such as: `echo CN> /proc/net/rtlxxx/wlan0/country_code`
2. wpa_supplicant.conf configuration parameter country=X, if it is softap, configuration parameter of hostapd.conf is country_code=X;
   Note: the way to confirm country code X can be searched through the website, such as https://countrycode.org/to see the combination of two capital letters in ISO CODES.

8.8 Wi-Fi KO Mode

If there are multiple loading/unloading operations in WiFi built in KO mode, the following patch needs to be added:

```
--- a/arch/arm64/boot/dts/rockchip/rk3xxx.dts
+++ b/arch/arm64/boot/dts/rockchip/rk3xxx.dts
@@ -112,6 +112,7 @@
```

```
    wireless-wlan {
        rockchip,grf = <&grf>;
        wifi_chip_type = "ap6354";
        sdio_vref = <1800>;
+       WIFI,poweren_gpio = <&gpio1 18 GPIO_ACTIVE_HIGH>; //Configure the PIN
corresponding to WIFI_REG_ON
        WIFI,host_wake_irq = <&gpio1 19 GPIO_ACTIVE_HIGH>;
        status = "okay";
    };

&sdio { //remove the following two configurations
    disable-wp;
    keep-power-in-suspend;
    max-frequency = <150000000>;
-   mmc-pwrseq = <&sdio_pwrseq>;
-   non-removable;
    num-slots = <1>;
}

diff --git a/drivers/mmc/host/dw_mmc.c b/drivers/mmc/host/dw_mmc.c
index 8730e2e..04b9cb8 100644
--- a/drivers/mmc/host/dw_mmc.c
+++ b/drivers/mmc/host/dw_mmc.c
@@ -1518,6 +1518,9 @@ static int dw_mci_get_cd(struct mmc_host *mmc)
    struct dw_mci *host = slot->host;
    int gpio_cd = mmc_gpio_get_cd(mmc);
+   if (mmc->restrict_caps & RESTRICT_CARD_TYPE_SDIO)
+       return test_bit(DW_MMC_CARD_PRESENT, &slot->flags);
+
    /* Use platform get_cd function, else try onboard card detect */
    if ((brd->quirks & DW_MCI_QUIRK_BROKEN_CARD_DETECTION) ||
        (mmc->caps & MMC_CAP_NONREMOVABLE))
@@ -2755,6 +2758,9 @@ static int dw_mci_init_slot(struct dw_mci *host, unsigned
int id)
    dw_mci_get_cd(mmc);
+   if (mmc->restrict_caps & RESTRICT_CARD_TYPE_SDIO)
+       clear_bit(DW_MMC_CARD_PRESENT, &slot->flags);
+
    ret = mmc_add_host(mmc);
    if (ret)
        goto err_host_allocated;

--- a/drivers/mmc/core/sdio.c
+++ b/drivers/mmc/core/sdio.c
@@ -996,9 +996,7 @@ static int mmc_sdio_resume(struct mmc_host *host)
        }

        /* No need to reinitialize powered-resumed nonremovable cards */
-       if (mmc_card_is_removable(host) || !mmc_card_keep_power(host)) {
-               err = mmc_sdio_reinit_card(host, mmc_card_keep_power(host));
-       } else if (mmc_card_keep_power(host) && mmc_card_wake_sdio_irq(host)) {
+       if (mmc_card_keep_power(host) && mmc_card_wake_sdio_irq(host)) {
                /* We may have switched to 1-bit mode during suspend */
                err = sdio_enable_4bit_bus(host->card);
        }
```

8.9 Debug Option

8.9.1 Wi-Fi Driver Debug

Sometimes, a more detailed log is needed to debug problems. For Realtek chips, please enable the following options to enable the kernel to print a more complete driver log:

```
#Modify directory: kernel/drivers/net/wireless/rockchip_wlan/rtl8xxx
#Newer driver:
Makefile
+CONFIG_RTW_DEBUG = y
+CONFIG_RTW_LOG_LEVEL = 2 #IT is 2 by defaul, and it can be changed to 4 during
debugging to print more complete log information

#Some old drivers do not have this configuration, and enable the following
configuration:
include/autoconf.h
+#define CONFIG_DEBUG /* DBG_871X, etc... */
```

8.9.2 TCPDUMP Capture Packet

Sometimes you need to capture packet to confirm problems, open the configuration, build and generate the tcpdump executable program, the packet capture command:

```
BR2_PACKAGE_TCPDUMP
tcpdump -h
tcpdump -i wlan0 -w /data/xxxx.pcap
```

8.9.3 wpa_supplicant Debugging

Sometimes the log of wpa_supplicant is needed to debug problems:

```
#Open the following configuration in Buildroot
#Remember to save by "make savedefconfig"
+ BR2_PACKAGE_WPA_SUPPLICANT_DEBUG_SYSLOG

#Rebuild
make wpa_supplicant-dirclean
make wpa_supplicant-rebuild

#When starting wpa_supplicant, add the -s parameter so that the log will be
output to the /var/log/messages file
"  -s = log output to syslog instead of stdout"

#-d print more logs
"  -d = increase debugging verbosity (-dd even more)"

#Because there are many logs of wpa, but the size of the messages file is small,
you can change the following configuration to increase the file size
buildroot/package/busybox/S01logging
```

```
@@ -3,7 +3,7 @@
# Start logging
#
-SYSLOGD_ARGS=-n
+SYSLOGD_ARGS="-n -s 8192"

#Rebuild busybox
make busybox-dirclean
make busybox-rebuild

#Package again at last
./build.sh

#Enable wpa_supplicant and add debug options such as -s -d
wpa_supplicant -B -i wlan0 -c /xxx/wpa_supplicant.conf -s -ddd
```

8.9.4 The mmc sdio Driver Debugging

```
diff --git a/drivers/mmc/host/dw_mmc.c b/drivers/mmc/host/dw_mmc.c
old mode 100644
new mode 100755
index 0ed1854..3019413
--- a/drivers/mmc/host/dw_mmc.c
+++ b/drivers/mmc/host/dw_mmc.c
@@ -410,6 +410,9 @@ static void dw_mci_start_command(struct dw_mci *host,
        "start command: ARGR=0x%08x CMDR=0x%08x\n",
        cmd->arg, cmd_flags);
+   if (host->slot[0]->mmc->restrict_caps & RESTRICT_CARD_TYPE_SDIO)
+       pr_err("start command: ARGR=0x%08x CMDR=0x%08x\n", cmd->arg, cmd_flags);
+
    mci_writel(host, CMDARG, cmd->arg);
    wmb(); /* drain writebuffer */
    dw_mci_wait_while_busy(host, cmd_flags);
@@ -2529,6 +2532,9 @@ static irqreturn_t dw_mci_interrupt(int irq, void *dev_id)
    pending = mci_readl(host, MINTSTS); /* read-only mask reg */
+   if (host->slot[0]->mmc->restrict_caps & RESTRICT_CARD_TYPE_SDIO)
+       pr_err("=== dw_mci_interrupt: pending: 0x%x ===\n", pending);
+
    /*
     * DTO fix - version 2.10a and below, and only if internal DMA
     * is configured.
@@ -2558,6 +2564,8 @@ static irqreturn_t dw_mci_interrupt(int irq, void *dev_id)
        }
        if (pending & DW_MCI_CMD_ERROR_FLAGS) {
+           if (host->slot[0]->mmc->restrict_caps & RESTRICT_CARD_TYPE_SDIO)
+               pr_err("=== CMD ERROR: 0x%x ===\n", pending);
            spin_lock_irqsave(&host->irq_lock, irqflags);
            del_timer(&host->cto_timer);
@@ -2573,6 +2581,8 @@ static irqreturn_t dw_mci_interrupt(int irq, void *dev_id)
        }
        if (pending & DW_MCI_DATA_ERROR_FLAGS) {
+           if (host->slot[0]->mmc->restrict_caps & RESTRICT_CARD_TYPE_SDIO)
+               pr_err("=== DATA ERROR: 0x%x ===\n", pending);
            /* if there is an error report DATA_ERROR */
            mci_writel(host, RINTSTS, DW_MCI_DATA_ERROR_FLAGS);
            host->data_status = pending;
@@ -2582,6 +2592,8 @@ static irqreturn_t dw_mci_interrupt(int irq, void *dev_id)
        }
        if (pending & SDMMC_INT_DATA_OVER) {
+           if (host->slot[0]->mmc->restrict_caps & RESTRICT_CARD_TYPE_SDIO)
+               pr_err("=== SDMMC_INT_DATA_OVER: 0x%x ===\n", pending);
            spin_lock_irqsave(&host->irq_lock, irqflags);
            if (host->quirks & DW_MCI_QUIRK_BROKEN_DTO)
```

9. Application Development

9.1 Introduction to Deviceio

Deviceio (external/deviceio_release) is an application development library, which eliminating underlying complex Wi-Fi/BT operations such as wpa_supplicant/bluez/bsa, etc., and provides friendly application development interfaces. Please refer to the docs\Linux\Wifibt directory Documents for details:

```
WIFI development: Rockchip_Developer_Guide_DeviceIo_WIFI_CN.pdf
Bluetooth development: Rockchip_Developer_Guide_DeviceIo_Bluetooth_CN.pdf
Network configuration development (BLE/SOFTAP):
Rockchip_Developer_Guide_Network_Config_CN.pdf
```

9.2 Configuration

Linux standard bluez protocol stack is used by Realtek Bluetooth: `buildroot/packages/bluez5_utils`

AzureWave/AMPAK take the proprietary bsa protocol stack: `external/broadcom_bsa and external/cypress_bsa`

- The previous SDK versions may need to be configured manually, remember that one of the three protocols must be selected, and the other two must be removed;
- Special attention: during the Bluetooth RF test of AzureWave/AMPAK , bluez hcitool/hciconfig tool will be used, and bluez can be manually configured temporarily;
- For latest release versions, Buildroot rkwifibt will automatically configure the corresponding protocol after selecting the corresponding module;

1. It will automatically open the bluez5 protocol stack and bluez-alsa (HFP/A2DP audio management component) when a Realtek module (such as RTL8723DS) is selected :

   ```
   +BR2_PACKAGE_RKWIFIBT_RTL8723DS
   +BR2_PACKAGE_BLUEZ5_UTILS
   +BR2_PACKAGE_BLUEZ_ALSA
   ```

2. Select an AMPAK module such as AP6255, the broadcom_bsa protocol stack will be opened automatically:

   ```
   +BR2_PACKAGE_RKWIFIBT_AP6255
   +BR2_PACKAGE_BROADCOM_BSA
   ```

3. Select an AzureWave module such as AW-CM256, the cypress_bsa protocol stack will be opened automatically:

   ```
   +BR2_PACKAGE_RKWIFIBT_AWCM256
   +BR2_PACKAGE_CYPRESS_BSA
   ```

4. After modifying the configuration by `make menuconfig`, you need to save the configuration by `make savedefconfig`;
5. The way to build and update rkwifibt module:

   ```
   make rkwifibt-dirclean && make rkwifibt-rebuild
   ```

6. Build and update the protocol stack. Choose one of three according to the module model:

   AMPAK modules:

   ```
   make broadcom_bsa-dirclean && make broadcom_bsa-rebuild
   ```

   AzureWave Modules:

   ```
   make cypress_bsa-dirclean && make cypress_bsa-rebuild
   ```

   Realtek modules:

   ```
   make bluez5_utils-dirclean && make bluez5_utils-rebuild
   make bluez-alsa-dirclean && make bluez-alsa-rebuild
   #Note: bluez-alsa is supported with bluez: A2DP and HFP functions
   ```