

Rockchip

WIFI/BT Developer Guide

Version: 4.0

Date: 2018.11

Warranty Disclaimer

This document is provided by “status” and Fuzhou Rockchip Electronics Co., Ltd ("our company ", the same below) does not provide any express or implied statement or warranty of any accuracy, reliability, integrity, merchantability, specific purpose and non-infringement of any statement, information and content of this document. This document is intended as a guide only for use.

Due to product version upgrades or other reasons, this document may be updated or modified from time to time without notice.

Trademarks

Rockchip and “瑞芯微”、“瑞芯” are trademarks of Fuzhou Rockchip Electronics Co., Ltd. and are exclusively owned by Fuzhou Rockchip Electronics Co., Ltd.

References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

Copyright © 2018 Fuzhou Rockchip Electronics Co., Ltd.

Exceeding the scope of reasonable use, No part of this publication may be copied or reproduced without the written permission of our company, and may not be transmitted in any form.

Fuzhou Rockchip Electronics Co., Ltd

Address: No.18 Building, A District, No.89 Software Boulevard, FuZhou, FuJian, PRC

Website: www.rock-chips.com

Tel: +86-591-83991906

Fax: +86-591-83951833

Mail: service@rock-chips.com

Preface

Overview

This document mainly introduces kernel configurations and related functions development, etc. of WIFI and BT based on Rockchip platform.

Chipset model

Chip name	Kernel version
RK3308	4.4

Applicable object

This document is mainly suitable for the following engineers

- Field application engineers
- Software development engineers

Revision history

Revision Date	Version No.	Author	Revision Description
2018/05/02	0.01	XY	Initial Release
2018/05/16	1.0	XY	Release version
2018/06/22	2.0	CTF	Release version
2018/07/21	3.0	CTF	Release version
2018/11/20	4.0	XY	Release version
2018/11/22	5.0	CTF	Release version

Table of content

Preface	1-1
Chapter 1. WIFI or BT Kernel configurations.....	1-1
1.1 DTS	1-1
1.2 Kernel	1-2
Chapter 2. Network configuration develop.....	2-3
2.1 Configure network by Command line:	2-3
2.2 Mobile phones configure network.....	2-3
2.2.1 Softap network configuration	2-3
2.2.2 Bluetooth configure network.....	2-6
2.2.3 Simple config for network configuration.....	2-8
Chapter 3. Bluetooth application develop	3-10
3.1 AzureWave /AMPAK modules.....	3-10
3.1.1 Configuration	3-10
3.1.2 Source code directory	3-11
3.1.3 Application Demo.....	3-13
3.1.4 Set Bluetooth device name and mac address	3-14
3.1.5 Ble configure network.....	3-14
3.1.6 A2DP SINK.....	3-14
3.1.7 A2DP SRC.....	3-15
3.1.8 HFP.....	3-18
Chapter 4. WiFi wireless resume (WoWLAN)	4-19
Chapter 5. Monitor mode of WiFi	5-20
5.1 Broadcom chips	5-20
5.2 Realtek chips.....	5-20
Chapter 6. WiFi RF indicators	6-21
Chapter 7. WiFi troubleshoot.....	7-21
7.1 WiFi is unrecognizable	7-21
7.2 WiFi cannot connect to router	7-22
7.3 WiFi connection is unstable.....	7-23
7.4 Other problems of WiFi	7-23
Appendix 1: Load Realtek Map calibration files	7-23
Appendix 2: Capture packet code example in Wifi Monitor mode.....	7-24

Preface

This document mainly introduces WiFi or BT development on RK linux platform, RTL series chip as a supplement to this document: \docs\Linux reference documents: RK3308_RTL8723DS_WIFI_BT_Instruction_V1.20.pdf

Chapter 1. WIFI or BT Kernel configurations

1.1 DTS

WIFI hardware pin configurations mainly includes the following items:

WIFI_REG_ON: Pin of WiFi power

```
sdio_pwrseq: sdio-pwrseq {
    compatible = "mmc-pwrseq-simple";
    pinctrl-names = "default";
    pinctrl-0 = <&wifi_enable_h>;
    reset-gpios = <&gpio0 RK_PA2 GPIO_ACTIVE_LOW>; // Note: the level state here is
    exactly opposite of enable state. For example, if REG_ON is active high, here is LOW; if
    REG_ON is active low, it is HIGH here.
};
&pinctrl {
    sdio-pwrseq {
        wifi_enable_h: wifi-enable-h {
            rockchip,pins =
                <0 RK_PA2 RK_FUNC_GPIO &pcfg_pull_none>; // WIFI_REG_ON
        };
    };
};
```

WIFI_WAKE_HOST: WIFI resume the PIN of controller

```
wireless-wlan {
    compatible = "wlan-platdata";
    rockchip,grf = <&grf>;
    wifi_chip_type = "ap6255"; // The name cant not be change when using AzureWave
    /AMPK modules, realtek needs to be filled in according to actual situation.
    WIFI,host_wake_irq = <&gpio0 RK_PA0 GPIO_ACTIVE_HIGH>; //WIFI_WAKE_HOST
    GPIO_ACTIVE_HIGH needs special attention: Check the connection relationship of the wifi pin
    and controller, if a reverse tube is added in the middle, it should be changed to low level
    trigger.
    status = "okay";
};
```

```
wireless-bluetooth {
    compatible = "bluetooth-platdata";
    uart_rts_gpios = <&gpio4 RK_PA7 GPIO_ACTIVE_LOW>;
    pinctrl-names = "default", "rts_gpio";
```

```
pinctrl-0 = <&uart4_rts>;
pinctrl-1 = <&uart4_rts_gpio>;
BT,power_gpio    = <&gpio4 RK_PB3 GPIO_ACTIVE_HIGH>; // BT_REG_ON
BT,wake_host_irq = <&gpio4 RK_PB4 GPIO_ACTIVE_HIGH>; // BT_WAKE_HOST
status = "okay";
};
```

1.2 Kernel

Select corresponding configuration according to actual WiFi

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

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

Buildroot configuration:

Select corresponding configurations according to actual WiFi, it must be consistent with the kernel configurations:

```
There is no help available for this option.
Prompt: wifi chip support
Location:
-> Target packages
  -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
    -> rkfwifibt (BR2_PACKAGE_RKWFIBT [=y])
Defined at package/rockchip/rkfwifibt/Config.in:5
Depends on: BR2_PACKAGE_ROCKCHIP [=y] && BR2_PACKAGE_RKWFIBT [=y]
Selected by: BR2_PACKAGE_ROCKCHIP [=y] && BR2_PACKAGE_RKWFIBT [=y] && m
```

puma



Chapter 2. Network configuration develop

2.1 Configure network by Command line:

First ensure that WiFi service process starts: `ps | grep wpa_supplicant`, If it does not start, please start it manually:

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

Modify the following file:

```
/ # vi /data/cfg/wpa_supplicant.conf
```

```
ctrl_interface=/var/run/wpa_supplicant
```

```
ap_scan=1
```

#Add the following configuration items

```
network={
    ssid="WiFi-AP"           // WiFi name
    psk="12345678"          // WiFi password
    key_mgmt=WPA-PSK // Optional encryption method, automatically recognized if
not filled
    # key_mgmt=NONE         // Not encrypted
}
```

Reread the above configurations: `wpa_cli reconfigure`

And reconnect: `wpa_cli reconnect`

2.2 Mobile phones configure network

2.2.1 Softap network configuration

APP: `/external/app/RkEcho.apk`

Brief introduction: Firstly, create an AP hotspot using the WiFi of SDK board, connect the AP hotspot on the mobile phone side; secondly, obtain the currently scanned hotspot list of the SDK board through the mobile phone apk, and fill in the password of the AP to be connected on the mobile phone. Apk will send AP's ssid and password to SDK; finally, the SDK will connect to WiFi based on the received information.

Buildroot configuration:

```

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

```

Source code directory:

/external/softapServer/ -- WiFi and APK related operations

/external/softapDemo/ -- WiFi related operations

Prepare a phone to install apk:

Make sure wifi server process starts

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

Step 1: Board command line runs:

softapServer Rockchip-Echo-123 (The name of wifi hotspot must be prefixed to Rockchip-Echo-xxx)

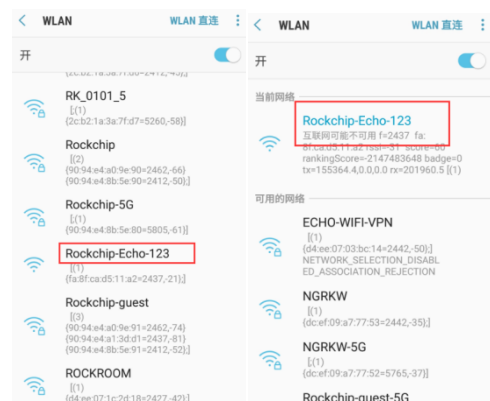
```

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

```

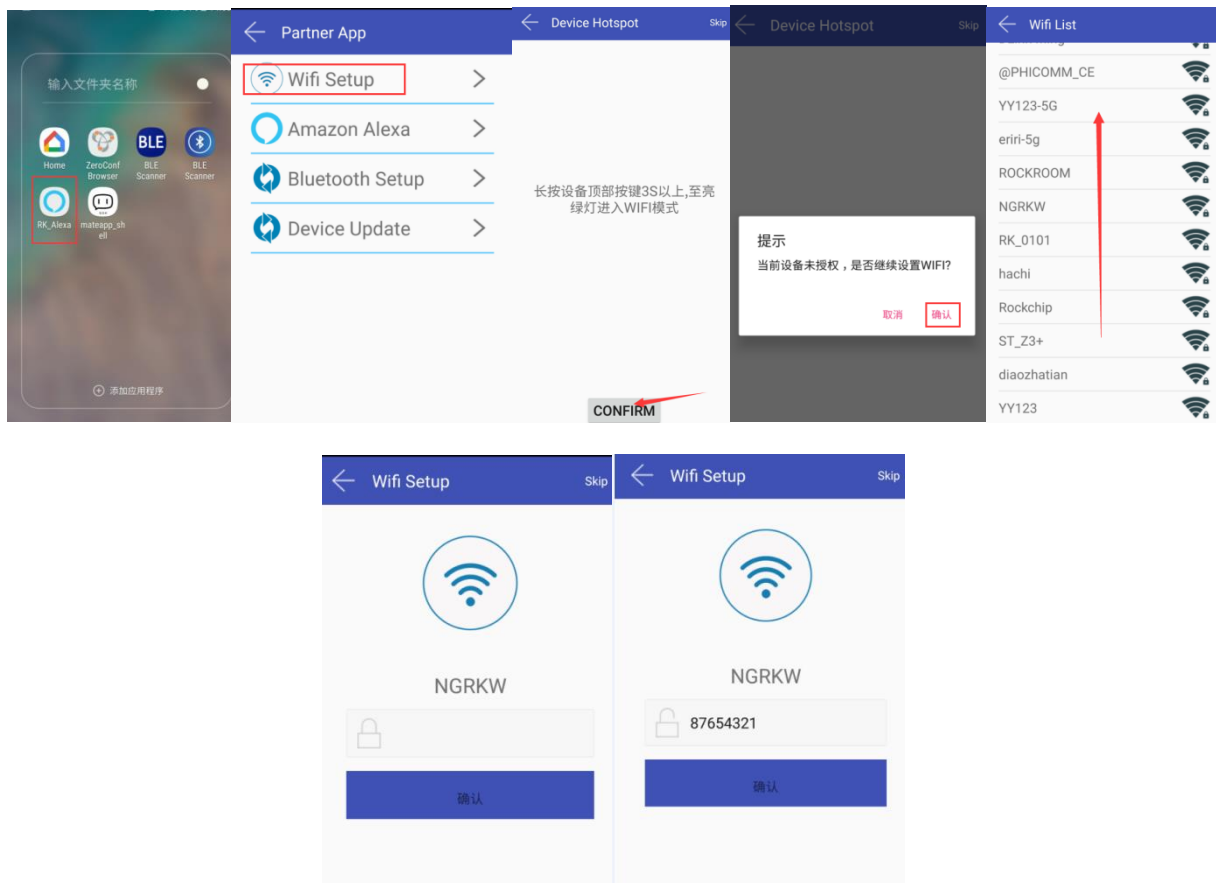
Step 2: Open wifi setting interface of the mobile phone:

Found Rockchip-Echo-123, click connection;



Step 3: Open the apk of the mobile phone:

Open apk, click wifi setup->CONFIRM-> CONFIRM ->wifi list-> Click on the name of network you want to connect to ->enter the password->click confirm



The serial port of the board will display:

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

[Server]: console_run: udhcpd -n -t 10 -i wlan0
udhcpd: started, v1.27.2
udhcpd: sending discover
udhcpd: sending discover
udhcpd: sending discover
udhcpd: sending discover
udhcpd: sending select for 192.168.1.16
udhcpd: lease of 192.168.1.16 obtained, lease time 86400
[Server]: console_run: wpa_cli -iwlan0 status
[Server]: Congratulation: wifi connected.
[Server]: getpid cmdResult:30840 30543
. self:30840.
DEBUG 263: check_wifi_chip_type_string: AP6255DEBUG 274:
wifi type: AP6255
DEBUG 286: -stop softap-
DEBUG 58: --- hostapd pid = 30605 ---
DEBUG 30: cmdline = kill 30605
wlan1: interface state ENABLED->DISABLED
wlan1: AP-STA-DISCONNECTED a0:cc:2b:cb:90:f5
DEBUG 30: cmdline = killall dnsmasq
ERROR Resource not found. for file:///data/mode_sound/wifi_connected.mp3
ERROR debug information: gstfilesrc.c(535): gst_file_src_start (): /GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0/GstFileSrc:
No such file "/data/mode_sound/wifi_connected.mp3"
DEBUG 30: cmdline = ifconfig wlan1 down
wlan1: AP-DISABLED
n180211: deinit ifname=wlan1 disabled 11b rates=0
DEBUG 30: cmdline = rm -rf /data/bin/wlan1
[Server]: Application exit.[Server]: accept error:Bad file descriptor
```

可以看到你输入名字和密码

获取到ip地址

自动退出程序

Check if the network is connected:

```
/ # echo nameserver 8.8.8.8 > etc/resolv.conf // Add dns domain name resolution
/ # ping www.baidu.com // check whether it will ping successfully
```

Notes:

1, After softspServer Rockchip-Echo-123 is executed, the command line cannot be exited

until network configuration is completed.

2, Don't write a wrong name, otherwise the apk can't enter the confirmation interface.
(Rockchip-Echo-xxx)

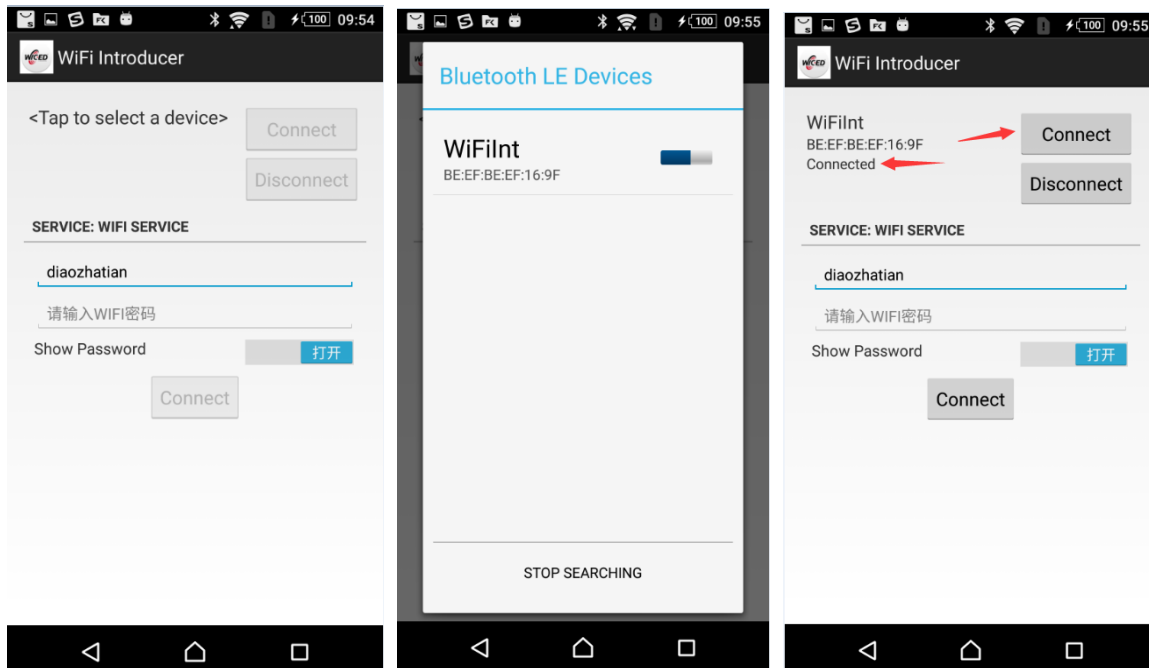
2.2.2 Bluetooth configure network

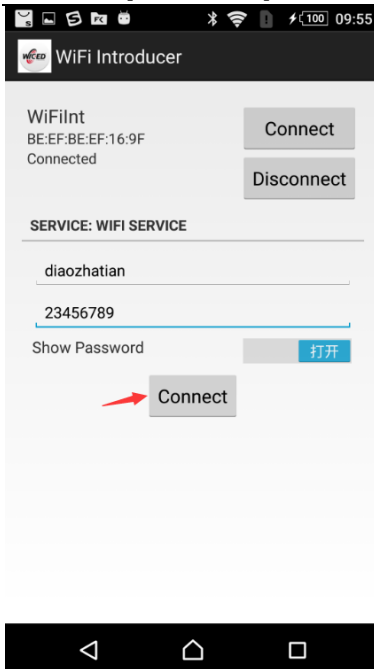
1, Realtek can refer to: \docs\Linux reference documents:
RK3308_RTL8723DS_WIFI_BT_Instruction_V1.20.pdf。

2, apk path: external/app/ WiFiIntroducer.apk

3, AP Bluetooth network configuration, run bsa_ble_wifi_introducer.sh start on the board.

4, Mobile phone app: click Tap to select a device -> select the device with a network configuration service -> connect device -> device option box shows the connection status-> If don't match, a window or a notification bar will prompt to match. Mismatch will result in matching timeout and Bluetooth disconnection -> Enter the WiFi name and password, Connect





5, After the board receives SSID and Passphrase, store to data/bsa/wpa_supplicant.conf; Command line menu select 4 => Display WiFi Introducer Sensor Information, can view the configured SSID and Passphrase

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

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

6, Starts wpa_supplicant and dhcpcd on the board to start configuring network. After network configuration is completed, command line runs wpa_cli status to check network connection status and ip address.

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

2.2.3 Simple config for network configuration

Realtek modules:

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

Kernel changes:

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

Only Realtek modules are supported

external/app/SimpleConfigApp.apk

Command line runs: `rtw_simple_config -D` & (`rtw_simple_config -h` to view help)

Install app on mobile phone: select Network -> enter Password -> click start to send -> finish configuration

[illegible]

Chapter 3. Bluetooth application develop

3.1 AzureWave / AMPAK modules

3.1.1 Configuration

1, source buildroot/build/envsetup.sh, select the version you want to compile, usually choose rockchip_rk3308_release

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

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

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

2, make menuconfig,

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

2.1 AMPAK modules

Select 'broadcom(ampak) bsa server and app'

Enter 'wifi/bt chip support (AP6255) --->' select chip model

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

2.2 AzureWave modules

Select 'broadcom(cypress) bsa server and app'

Enter 'wifi/bt chip support (AW-CM256) --->' select chip model

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

2.3 Exit option box, select Yes, make savedefconfig to save configuration;

2.4 Compile and package

AMPAK modules: make broadcom_bsa-rebuild

AzureWave modules: make cypress bsa-rebuild

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

3.1.2 Source code directory

Broadcom-based AzureWave / AMPAK modules supports BSA protocol stack, while BSA protocol stack is a Bluetooth protocol stack developed by Broadcom, which is similar to BLUEZ, developers can develop various Bluetooth apps based on it and provides rich app demos.

Note: It is only a brief introduction here. Since BSA protocol stack is not open source and we do not have any code. Therefore, if you need detailed development documentations, please consult the module vendors. They will provide technical support. If you can't get in touch with them, you can contact us; of course, if you don't want to use BSA, you can also use the open source BlueZ which is universal, so you can refer directly to: RK3308 RTL8723DS WIFI BT Instruction V1.20.pdf.

1, AMPAK modules:

Script and compile mk file: buildroot/package/rockchip/broadcom_bsa

Source code: external/broadcom_bsa

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

2, AzureWave modules:

Script and compile mk file: buildroot/package/rockchip/cypress_bsa

Source code: external/bluetooth_bsa

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

*Bsa Application Demo List

app_3d -- This application is used to connect 3D stereo glasses

app_ag -- Audio Gateway (AG) application. AG applications (usually located in cellular phones) are used to connect to mono headsets.

app_av -- This application is used to stream music from an AV Source (AVS) profile.

app_avk -- This application is used to receive a music steam from an AV sink (AVK) profile.

app_ble -- This application shows how to use BLE client/server module BSAs.

app_ble_blp -- This application is sample code for the BLE blood-pressure collector.

app_ble_cscc -- This application is sample code for the BLE cycling speed and cadence collector.

app_ble_hrc -- This application is sample code for the BLE heart rate controller.

app_ble_hrp -- This application is sample code for the BLE health thermometer collector.

app_ble_pm -- This application is sample code for the BLE proximity monitor.

app_ble_rsc -- This application is sample code for the BLE running speed and cadence collector.

app_ble_wifi -- This sample application makes it easier to set up a Wi-Fi Direct connection between peers by exchanging Wi-Fi Direct information over BLE.

app_dg -- Data Gateway (DG). Two Bluetooth devices can connect using this application to exchange data via serial communications port emulation.

app_fm -- This application shows how to use the FM radio APIs.

app_ftc -- FTP client (FTC). FTP clients can connect and access (browse/read/write) files located on the FTP server.

app_fts -- File Transfer Server (FTS). The FTS can connect and access (browse/read/write) files located on the FTP server.

app_hd -- Demonstrates Bluetooth HID device and remote control functionality.

app_headless -- Shows how the Headless mode is controlled.

app_hd -- Connects Human Interface Devices (HIDs) such as a keyboard, mouse, or remote control.

app_hh -- Connects Human Interface Devices (HIDs) such as a keyboards, mice, or remote controls.

app_hl -- This application is used to connect the health devices.

app_hs -- Headset (HS) application. HS applications are used to connect to cellular phones which run AG applications.

app_manager -- The main regular application. Governs security and device management. It must always be running.

app_mce -- This application provides the Bluetooth MAP client (Message Client Equipment [MCE]) function.

app_opc -- OP client (OPC). OP clients can exchange (send/receive) files with the OPS.

app_ops -- Object Push (OP) server (OPS). OP clients can exchange (send/receive) files with the OP server.

app_pan -- This application provides the test framework for Bluetooth PAN functionality.

app_pbc -- This application provides the Bluetooth PBAP client (PBC) function.

app_pbs -- Phone Book (PB) server (PBS). PB clients can connect to the PBS to download/upload phone books.

app_sac -- This application provides the Bluetooth SIM Access Profile (SAP) client function.

app_sc -- This application provides the Bluetooth SAP client (SC) function.

app_switch -- This application demonstrates the sequence of operations necessary to implement role switching between AV/AG and AVK/HF.

app_tm -- This application describes Test Module (TM) APIs.

3.1.3 Application Demo

1, Power on:

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

2, All Bluetooth related processes must be run in the /data/bsa/config directory.

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

3, Start server process

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

XXXX.hcd: firmwares corresponding to each chip

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

4, Start management process: (Run to this step, you will discover and connect device on the phone Bluetooth list)

```
app_manager &
```

5, Start the client you want to run (app_av, app_avk, etc.):

```
app_xxxx &
```

3.1.4 Set Bluetooth device name and mac address

- 1, File: app_manager.c
- 2, Interface: app_mgr_config
- 3, Example:

```

1482: #ifdef DUEOS
1483:     app_mgr_get_bt_config((char *)app_xml_config.bd_addr, BD_ADDR_LEN);
1484:     sprintf((char *)app_xml_config.name, "%s%02X%02X", "DuerOS_",
1485:         app_xml_config.bd_addr[4], app_xml_config.bd_addr[5]);
1486:     APP_DEBUG1("device name: %s", app_xml_config.name);
1487: #else
1488:     bdcpy(app_xml_config.bd_addr, local_bd_addr);
1489:     /* let's use a random number for the last two bytes of the BdAddr */
1490:     gettimeofday(&tv, NULL);
1491:     rand_seed = tv.tv_sec * tv.tv_usec * getpid();
1492:     app_xml_config.bd_addr[4] = rand_r(&rand_seed);
1493:     app_xml_config.bd_addr[5] = rand_r(&rand_seed);
1494:     sprintf((char *)app_xml_config.name, "My BSA Bluetooth Device %02x%02x",
1495:         app_xml_config.bd_addr[4], app_xml_config.bd_addr[5]);
1496: #endif

```

Note:

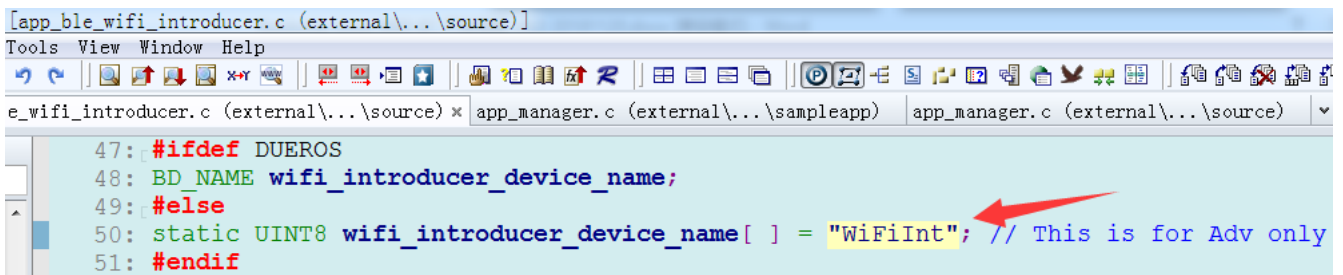
- 1), app_xml_config.name: Store Bluetooth name
- 2), app_xml_config.bd_addr: Store mac address
- 3), app_mgr_get_bt_config: Get mac address from Bluetooth chip
- 4), bdcpy(app_xml_config.bd_addr, local_bd_addr): Write a fixed mac address, the last two bits use a random value

4, Setting method:

Write your desired Bluetooth name and mac address in app_xml_config.name and app_xml_config.bd_addr

3.1.5 Ble configure network

- 1, Execute bsa_ble_wifi_introducer.sh start
- 2, Refer to chapter 2.2.2 Bluetooth configure network in this document for configuration network with apk
- 3, Code directory: external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux/app_ble_wifi_introducer
- 4, Modify device name when configuring network:



```

[app_ble_wifi_introducer.c (external\...\source)]
Tools View Window Help
e_wifi_introducer.c (external\...\source) * app_manager.c (external\...\sampleapp) app_manager.c (external\...\source)
47: #ifdef DUEOS
48: BD_NAME wifi_introducer_device_name;
49: #else
50: static UINT8 wifi_introducer_device_name[ ] = "WiFiInt"; // This is for Adv only
51: #endif

```

3.1.6 A2DP SINK

- 1, Execute: bsa_bt_sink.sh start
- 2, Turn on the mobile phone Bluetooth, it will display My BSA Bluetooth Device XXXX, click connection to achieve music playing function;

Note: If there is no sound after connection, check sound card configuration.

3, Code path: external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux/app_avk

3.1.7 A2DP SRC

1, Code path: external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux/app_av

2, Run: bsa_bt_source.sh start

3, cd data/bsa/config/

4, Run app_av, will display below app_av menu

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

5, Enter 2, start discovering Bluetooth devices

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

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

6, Enter 6 => AV Open (Connect)

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

7, Enter 1, display the list of devices that were last discovered (if device is already connected, select 0 directly)

```

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

```

8, Enter the Dev serial number corresponding to the device you want to connect , enter 3 here and select Dev:3 Fiil Wireless for matching and connection.

```

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

```

9, Play wav file

Enter 11 to display a list of wav files that can be played in the /data/bsa/config/test_files/av directory; you can push the wav file you want to play to this directory, or you can make a copy in buildroot/package/rockchip/broadcom_bsa/broadcom_bsa.mk. Select the serial number corresponding to the file you want to play to play the corresponding file.

```

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

```

10, When there is a wav file currently playing, you need to enter 14 => AV Stop to stop playing, in order to repeat step 9 to play a new file. You can also control playback with the following commands:

- 14 => AV Stop
- 15 => AV Pause

- 16 => AV Resume
- 17 => AV Send RC Command (Inc Volume)
- 18 => AV Send RC Command (Dec Volume)

3.1.8 HFP

1, Only suitable for AMPAK modules

2, The external/broadcom_bsa must contain the following submission:

```
commit 756a6c3d984a085b3a5aaf5a9691a3f39dcc7c8c
```

```
Author: ctf <ctf@rock-chips.com>
```

```
Date: Wed Sep 26 18:00:28 2018 +0800
```

```
bluetooth: broadcom_bsa: integrate arecord and aplay into HFP.
```

```
Change-Id: Ic3fef580a24eaa4126dd6eb53a59adb4e2ed6d01
```

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

3, Code path: external/broadcom_bsa/3rdparty/embedded/bsa_examples/linux/app_hs

4, Analog mic v11: default 0,1channel map loopback, should set the adc-channel map in dts; as follows:

Take rk3308-evb-amic-v11.dts as an example:

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

5, Add a Bluetooth device node, and modify period_size and buffer_size of playback

Add code below at the end of buildroot/board/rockchip/rk3308/fs-overlay/etc/ asound.conf:

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

```

    slave {
        pcm "hw:1,0"
    }
}
}

```

Modify period_size and buffer_size of playback:

```

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

```

6, Execute bsa_bt_hfp.sh start

7, Connect My BSA Bluetooth Device XXXX device to achieve Hfp two-way conversation

Chapter 4. WiFi wireless resume (WoWLAN)

At present, WiFi supports wireless network packet resume system. For example, if an audio device is normally connected to WiFi and obtains the correct IP address, then when the device suspends, we can resume the system through wireless network packet. **Wake up rule: as long as they are network packages which are sent to this device IP address, they will resume the system.**

AP6XXX/RTL modules upper layer configuration: modify "wpa_supplicant.conf" file, add the following configuration:

```

wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
update_config=1
ap_scan=1
+wowlan_triggers=any // Make sure the changes take effect

```

Please open the following configuration of Realtek modules in the corresponding Makefile:

```

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

```

```
CONFIG_GPIO_WAKEUP = y
```

Note:

1, Make sure that **WIFI_WAKE_HOST** is configured: **WIFI resumes the PIN of controller**

WiFi configuration of Dts: `WIFI,host_wake_irq = <&gpio0 RK_PA0 GPIO_ACTIVE_HIGH>;`
 // Special attention of **WIFI_WAKE_HOST GPIO_ACTIVE_HIGH**: Confirm the connection relationship between the wifi pin and controller. If a reverse tube is added in the middle, it should be changed to low level trigger.

2, **Make sure hostapd process is turned off before suspending. The network wakeup function requires that hostapd process should be turn off;**

Software can refer to source code of ping

Test method: After the device is connected to WiFi and obtains IP address normally, (echo mem > sys/power/state), after suspending, mobile phone downloads a ping software (make sure the mobile phone or PC is connected to the same LAN), and then ping the device IP address, if normal, you can see that the device will be resume.

Troubleshooting: AP and RTL chips are triggered by high level by default. Assume that WiFi_Wake_Host pin is directly connected to the controller, after the device enters suspending, the pin is low level by default. When there is a network packet to resume, this pin can be measured in high pulse input with an oscilloscope; so when the device is not resumed, please use the oscilloscope to measure whether the pin meets the above behavior; if you still can't solve issues, please provide: the whole dmesg kernel log and cat proc/interrupts printout.

Chapter 5. Monitor mode of WiFi

Turn on monitor mode of WiFi:

5.1 Broadcom chips

dhd_priv SDK comes with this command

Set channel:

```
dhd_priv channel 6 // channel numbers
```

Turn on monitor mode:

```
dhd_priv monitor 1
```

Then use the sample code raw code in attachment 2 to read the captured package.

Turn off monitor mode:

```
dhd_priv monitor 0
```

5.2 Realtek chips

Need to open driver Makefile:

```
CONFIG_WIFI_MONITOR = y
```

1. ifconfig wlan0 up

ifconfig p2p0 down

2. iwconfig wlan0 mode monitor /*support wext solution.*/

or iw dev wlan0 set type monitor /*support cfg80211 solution.*/

3. echo "<chan> 0 0" > /proc/net/<rtk_module>/wlan0/monitor /*<rtk_module> is the

realtek wifi module name, such like rtl8812au, rtl8188eu ..etc */

4. tcpdump -i wlan0 -s 0 -w snf_pkts.pcap /*capture the sniffer packets and save it as a file "snf_pkts.pcap" or use the sample code raw code in attachment 2 to read the captured package. */

Chapter 6. WiFi RF indicators

Hardware engineers have to confirm whether RF indicator and frequency offset of WiFi are normal; Realtek chips need to provide signaling mode hardware test reports, and ask the modules vendor for the map calibration file of the corresponding chip (about calibration file download, refer to Appendix 1); Make sure the hardware indicators are qualified.

Chapter 7. WiFi troubleshoot

7.1 WiFi is unrecognizable

First, measures the hardware voltage of WIFI_REG_ON/VDDIO VBAT/SDIO_CLK/SDIO_CMD/SDIO_DATA0~SDIO_DATA3, which is described in detail as follows:

(1) Confirm whether the signal of WIFI_CLK is normal, **confirm that 37.4/24/26M has normal oscillation started after power-on**. When loading drive, there is a clock to be discharged on SDIO_CLK, initialization 400K. After stabilization, it will reach the CLK set by DTS.

(2) Confirm whether the 32.768K square wave signal is normal, the peak-to-peak value is required to be within the range of $0.7 * VDDIO \sim VDDIO$. Otherwise there will be problems;

(3) Confirm whether the WL/WIFI_REG_ON WIFI power-on pin is normally pulled high when WIFI is turned on, and whether there are level changes during abnormal time, whether the VBAT power supply is rippled or not; Pin configurations of WIFI_REG_ON (sdio-pwrseq) in the DTS part corresponding parsing code operation, refer to:

drivers/mmc/core/pwrseq_simple.c

Parse reset-gpio:

```
mmc_pwrseq_simple_alloc
```

Power up and down:

```
static struct mmc_pwrseq_ops mmc_pwrseq_simple_ops = {
    .pre_power_on = mmc_pwrseq_simple_pre_power_on, // pull down WiFi_REG_ON
    .post_power_on = mmc_pwrseq_simple_post_power_on, // pull up WiFi_REG_ON
    .power_off = mmc_pwrseq_simple_power_off, // pull down WiFi_REG_ON
    .free = mmc_pwrseq_simple_free,
};
```

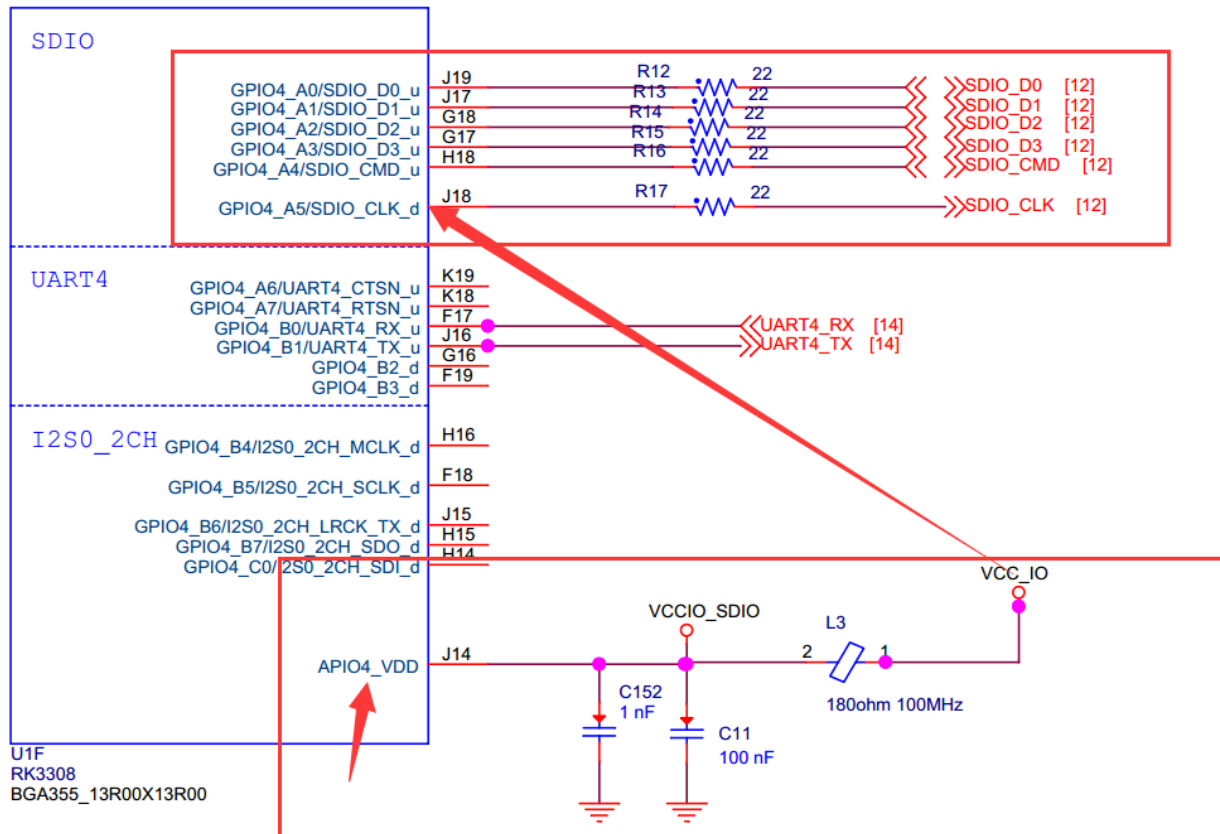
You can see through an oscilloscope: WiFi_REG_ON will be pulled down and then pulled up after WiFi initialization. If it is not observed, please add some debug at the above code to confirm.

(4) Confirm whether there are missing or soldering errors in the peripheral component materials of crystal part;

(5) Check if there are any problems with the 4 data traces of sdio, if there is any interference.

(6)SDIO data transmission is abnormal, check whether the materials used by sdio wifi hardware meet the standard, such as whether capacitors or resistors are misconnected or missing;

(7) The &io-domains, as shown in the following figure VCCIO/VCCIO_SDIO to supply SDIO_DX/CMD/CLK, it needs to be consistent with the software configuration: for example, APiO4 corresponds to vccio4-supply = <&vccio_sdio>; //it will fill the name which will supply to Apio4, as VCC_IO and Vccio_sdio are the same supply, you can also fill vcc_io.



If WiFi is properly recognized, it will be printed as follows in the log:

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

or

mmcX: new high speed SDIO card at address 0001

7.2 WiFi cannot connect to router

1, Please make sure that the following two processes are running.

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

2, wpa_cli scan and wpa_scan_r commands will scan hotspots. If execution fails, it can be executed multiple times to confirm whether it can scan to wifi: normally, it will contain below information: check if there is WiFi in the following information, If you can't capture it, or if the captured wifi number is different from which captured by your mobile phone or other devices, or the signal is very weak (signal level), please check if the WiFi antenna meets the standard and whether the frequency offset meets the requirement.

```

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

```

7.3 WiFi connection is unstable

Refer to the WIFI RF target chapter

7.4 Other problems of WiFi

Please provide the log of kernel dmesg and wpa_supplicant (method: add the debug option to the place where the wpa_supplicant program is started, such as: `wpa_supplicant -B -i wlan0 -c /data/cfg/wpa_supplicant.conf -f debug.txt // log will be redirected to the debug.txt file`).

Appendix 1: Load Realtek Map calibration files

KO mode: for convenience of debugging, you can change to the ko mode to load map files. Let's take 8723DS as an example. Others are similar:

(1), Change kernel configuration:

```
CONFIG_RTL8723DS=m
```

change the directory where you think is convenient:

```

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

```

Recompile and generate the ko

```
drivers/net/wireless/rockchip_wlan/rtl8723ds/8723ds.ko
```

Update kernel

(2), Push necessary documents

Push wifi_efuse_8723ds.map file to the directory changed above:

```
+EXTRA_CFLAGS += -DEFUSE_MAP_PATH=\"/data/wifi_efuse_8723ds.map\"
```

Push the 8723ds.ko to data or other directory where you think is convenient

```
adb push wifi_efuse_8723ds.map /data/
```

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

(3), Execute after power on

```
insmod /data/8723ds.ko // Note to be consistent with the directory you pushed above.
```

will display in log:

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

Buildin mode:

Sometimes in order to speed up network speed when power on, it hope to use buildin, you can achieve by the following ways:

drivers/net/wireless/rockchip_wlan/rtl8189fs/core/efuse/rtw_efuse.c

Find rtw_read_efuse_from_file function:

```
{
... ..
map = rtw_vmalloc(map_size); // Apply for memory

for (i = 0 ; i < map_size ; i++) {
... .. //Parse the contents of the map file, assign value to map pointer
}

DBG_871X_LEVEL(_drv_always_, "efuse file:%s, 0x%03xbyte content read\n", path, i);
//First load it in the ko mode, print the contents of map pointer here, and
then make the contents into an array, directly assign to the map, and skip
parsing action.
_rtw_memcpy(buf, map, map_size); // Here is the final assignment operation.
... ..
}
```

Appendix 2: Capture packet code example in Wifi Monitor mode

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <linux/socket.h>
```

```
#include <stdio.h>
#include <string.h>
#include <netpacket/packet.h>
#include <net/if.h>
#include <netinet/in.h>

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

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

int main(int argc, char ** argv)
{
    struct ifreq ifr;
    struct sockaddr_ll ll;
    int fd;

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

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

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

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

    memset(&ll, 0, sizeof(ll));
    ll.sll_family = PF_PACKET;
    ll.sll_ifindex = ifr.ifr_ifindex;
    ll.sll_protocol = htons(protocol);
```

```
if (bind(fd, (struct sockaddr *) &ll, sizeof(ll)) < 0) {
    printf("bind fail \n");
    close(fd);
    return -1;
}

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

    memset(&ll, 0, sizeof(ll));
    fromlen = sizeof(ll);
    res = recvfrom(fd, buf, sizeof(buf), 0, (struct sockaddr *) &ll,
        &fromlen);
    if (res < 0) {
        printf("res < 0\n");
        return -1;
    } else {
        cc++;
        printf("%04d(%03d) - ", cc, res);
        //for(i = 0; i < res && i < 8; i++)
        for(i = 0; i < res; i++)
            printf("%02x ", buf[i]);
        printf("\n");
    }
}

close(fd);

return 0;

}
```