FCS866R&FCS94xR&FCSx50R&FCU741R Series Linux&Android Wi-Fi User guide

Short-Range Module Series

Version: 1.0.0

Date: 2024-03-12

Status: Preliminary

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an "as available" basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2023-08-04	Fengy FENG	Creation of the document
1.0.0	2024-03-12	Elijah ZHOU/ Sherk ZHANG	Preliminary

Contents

Table Index

Figure Index

1 Introduction

FCS866R, FCS94xR, FCSx50R and FCU741R series modules are highly integrated Wi-Fi modules provided by Quectel. The modules support the implementation of Wi-Fi functionality on Linux and Android platforms. This document takes the RK3568-WF EVB with Quectel Wi-Fi module installed as an example to outline the process for porting Wi-Fi driver, methods for verifying Wi-Fi functionality and procedures for capturing logs.

NOTE

Unless otherwise stated, the blue font in this document indicates differences or modifications.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Family	Module
-	FCS866R
FCS94xR	FCS940R
	FCS945R
FCS85xR	FCS850R Series
	FCS950R
-	FCU741R

1.2. Differences Among Wi-Fi Modules

The differences among the Wi-Fi modules are as shown in the following table:

Table 2: Differences Among Wi-Fi Module

Module Model	Interface Type	Supported Band	Wi-Fi Standard	Number of Antennas	Supported Bandwidth (Unit: MHz)	Maximum Theoretical Rate (Unit: Mbps)	Encryption Mode
FCS850R	SDIO	2.4 GHz/ 5 GHz	Wi-Fi 5	Dual antennas	20/40/80	867	WPA/WPA2/WPA3
FCS850R-B	SDIO	2.4 GHz/ 5 GHz	Wi-Fi 5	Three antennas	20/40/80	867	WPA/WPA2/WPA3
FCS866R	SDIO	2.4 GHz/ 5 GHz	Wi-Fi 6	Dual antennas/ Three antennas	20/40/80	1201	WPA/WPA2/WPA3
FCS940R	SDIO	2.4 GHz	Wi-Fi 4	Single antenna	20/40	150	WPA/WPA2/WPA3
FCS945R	SDIO	2.4 GHz/ 5 GHz	Wi-Fi 4	Single antenna	20/40	150	WPA/WPA2/WPA3
FCS950R	SDIO	2.4 GHz/ 5 GHz	Wi-Fi 5	Single antenna	20/40/80	433	WPA/WPA2/WPA3
FCU741R	USB	2.4 GHz/ 5 GHz	Wi-Fi 4	Single antenna	20/40	150	WPA/WPA2/WPA3

2 Linux Platform

2.1. Environment Preparations

2.1.1. Hardware Environment

Table 3: Hardware Environment

Name	Quantity
Quectel RK3568-WF EVB	1
Quectel Wi-Fi module	1
Antenna	1
USB Type C cable	1
Micro USB cable	1
Power cable	1

The hardware connection diagram is as follows. Please install the corresponding module on the M.2 port of the EVB board as required.

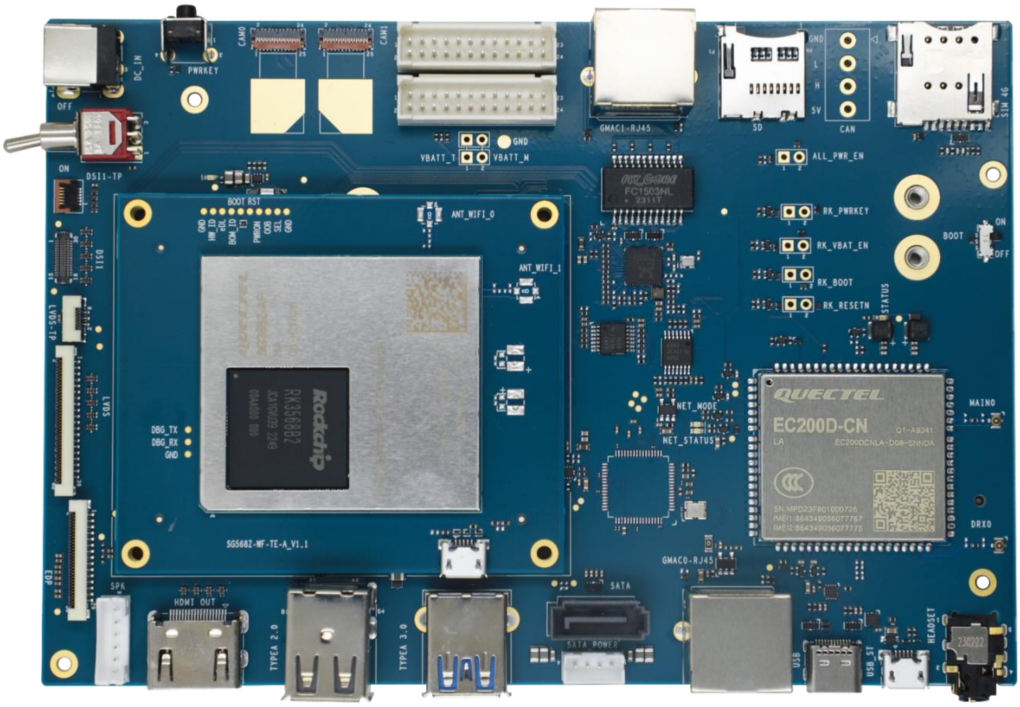

Figure 1: Top View of RK3568-WF EVB

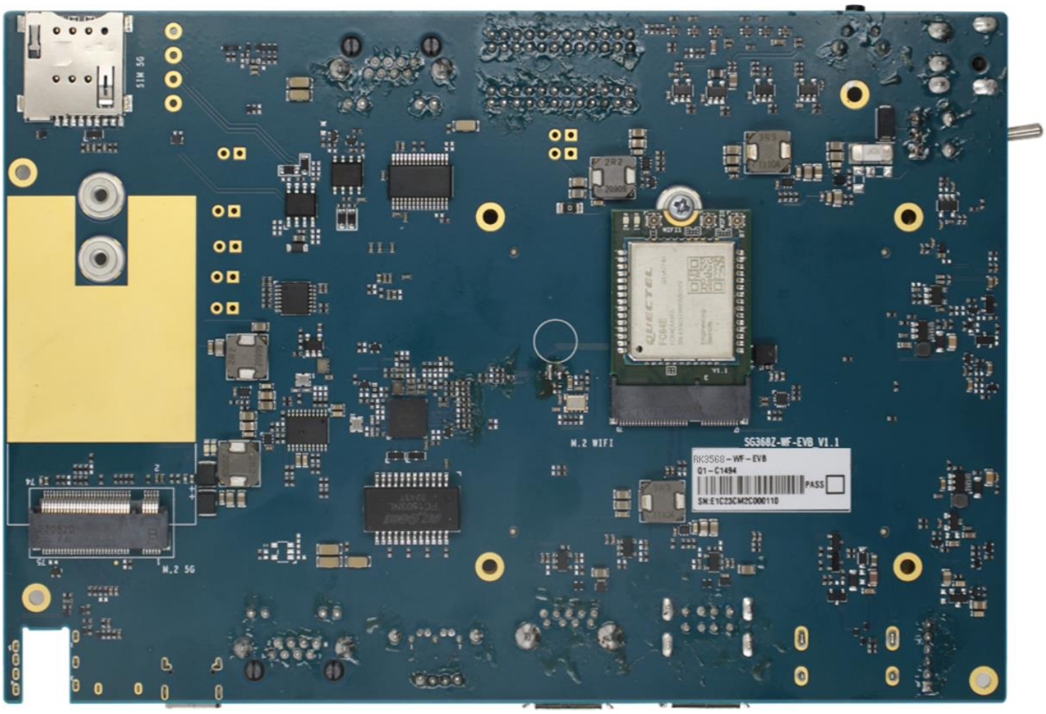

Figure 2: Bottom View of RK3568-WF EVB

FCU741R is a module with USB interface, and it is connected to the EVB as follows:

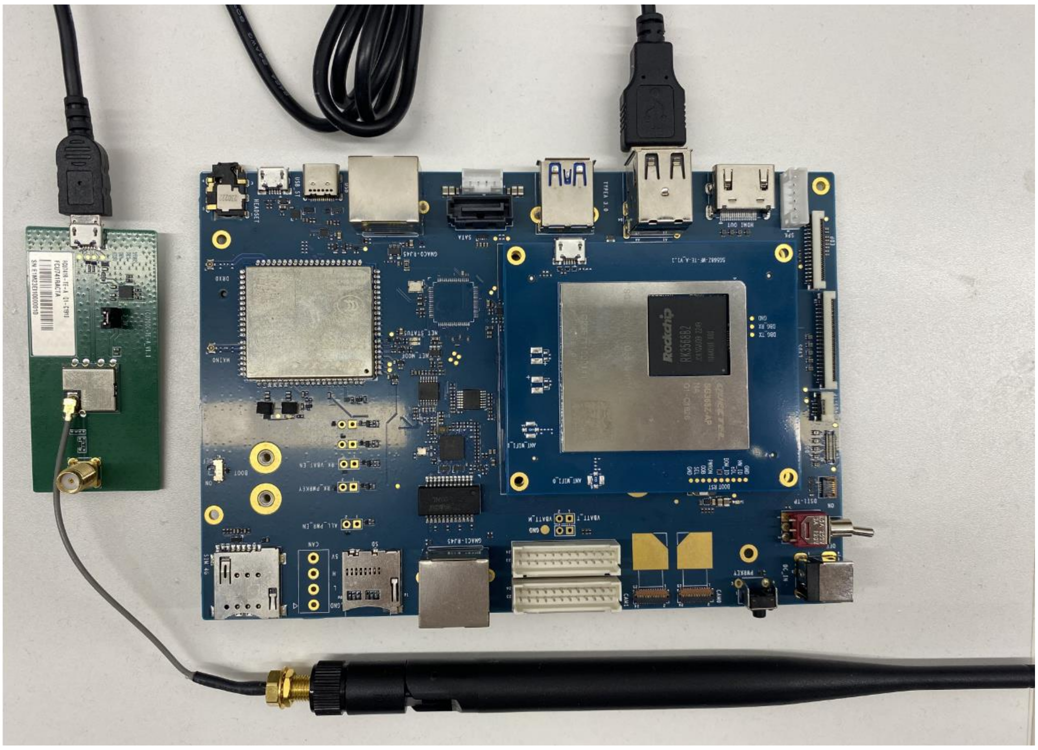

Figure 3: RK3568-WF EVB + FCU741R Module

For M.2 EVB installed with a SDIO interface module such as FCS850R series, FCS866R, FCS940R, FCS945R and FCS950R modules, the VDD_IO uses 1.8 V power supply by default. The VDD_IO power supply of the module must match that of the host controller. If the host controller uses 3.3 V power supply, the module must also use 3.3 V power supply. This can be achieved by soldering the 0-ohm resistors on the M.2 EVB to select either 1.8 V or 3.3 V. For the RK3568-WF EVB, the default VDD_IO is set to 1.8 V, so there is no need to modify the VDD_IO power supply on the M.2 EVB.

Taking installing the FCS945R module on the M.2 EVB as an example, you can solder the resistor to location 1 to select 1.8 V power supply, while solder the resistor to location 2 to select 3.3 V power supply.

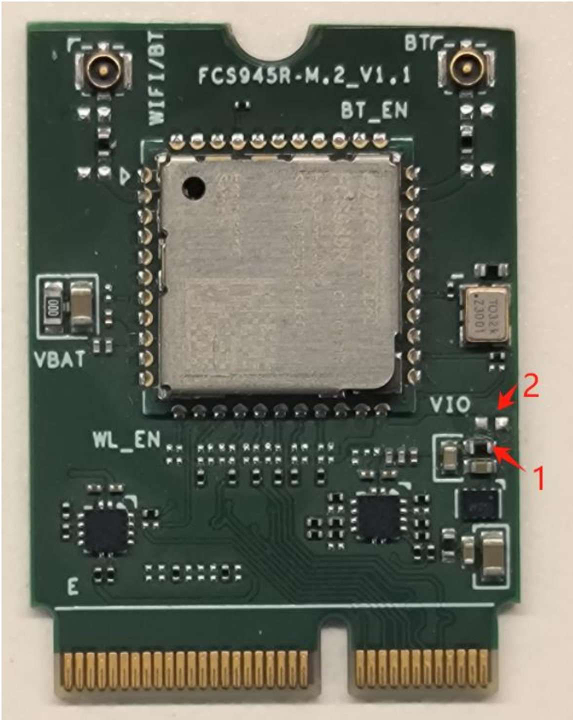

Figure 4: Modification of Module IO Voltage

If you need to change the VDD_IO power supply of the M.2 EVB, please contact Quectel Technical Support for confirmation before proceeding.

2.1.2. Software Environment

Table 4: Linux Software Environment

Type	Description
Code environment	Linux 4.19.232
Driver package	Quectel Wi-Fi driver package
Compilation environment	Ubuntu 18.04
Tool	Install RKDevTool on Windows PC

NOTE

1. If you are developing and debugging the module based on Quectel RK3568-WF EVB, please contact Quectel Technical Support to obtain the corresponding Linux source code package.
2. For the downloading and use of RKDevTool, please visit official website: https://www.t-

firefly.com/doc/download/136.html.

2.1.3. Obtaining Wi-Fi Driver Package

Step 1: Contact Quectel Technical Support to obtain the GitLab account.
Step 2: Execute **ssh-keygen** on Ubuntu PC to generate SSH public key.
Step 3: Log in to GitLab (https://git-master.quectel.com/) and add the SSH public key generated in ***Step 2***.
Step 4: Execute the following command on Ubuntu PC to obtain the module source code package.

Table 5: Commands for Obtaining Module Source Code Package

Module	Command
FCS850R series	**git clone ssh://git@git-master.quectel.com:8407/wifi.bt/fcs850r.git**
FCS866R	**git clone ssh://git@git-master.quectel.com:8407/wifi.bt/fcs866r.git**
FCS940R	**git clone ssh://git@git-master.quectel.com:8407/wifi.bt/fcs940r.git**
FCS945R	**git clone ssh://git@git-master.quectel.com:8407/wifi.bt/fcs945r.git**
FCS950R	**git clone ssh://git@git-master.quectel.com:8407/wifi.bt/fcs950r.git**
FCU741R	**git clone ssh://git@git-master.quectel.com:8407/wifi.bt/fcu741r.git**

Taking FCS945R module as an example, the directory of the obtained package is as follows:

```
~/fcs945r$ tree -L 2
.
├── Code
|   ├── BT
|   ├── Tools
|   └── WIFI
└── Doc
    ├── CN
    └── EN
```

The *Doc* directory stores the Bluetooth and Wi-Fi driver user guide of the module. The Wi-Fi driver package contains the *WIFI* and *Tools* directories under the *Code* directory, which including the following files:

Table 6: Wi-Fi Files

File	Description
WIFI/wlan_src	Wi-Fi driver code
WIFI/txpower	RF power parameter file
Tools	Non-signaling Test Tool

Taking FCS945R module as an example, the directory of the Wi-Fi driver package is as follows:

```
~/fcs945r/Code$ tree -L 3
 ├─── Tools
 |    ├─── rtwpriv_arm
 |    └─── rtwpriv_arm64
 └─── WIFI
     ├─── txpower
     |    ├─── PHY_REG_PG.txt
     |    └─── TXPWR_LMT.txt
     └─── wlan_src
         ├─── clean
         ├─── core
         ├─── hal
         ├─── halmac-rs.mk
         ├─── ifcfg-wlan0
         ├─── include
         ├─── Kconfig
         ├─── Makefile
         ├─── os_dep
         ├─── platform
         ├─── rtl8733b.mk
         ├─── runwpa
         └─── wlan0dhcp
```

The *Tools* directory contains the non-signaling test tools*. rtwpriv_arm* in *Tools directory* is the tool used on 32-bit system, while *rtwpriv_arm64* in *Tools* directory is the tool used on 64-bit system.

The *WIFI/txpower* directory stores RF power parameter files. For FCU741R, FCS850R series, FCS940R, FCS945R and FCS950R modules, the RF power parameter files are *PHY_REG_PG.txt* and *TXPWR_LMT.txt*. For FCS866R module, the RF power parameter files are *TXPWR_ByRate.txt* and *TXPWR_LMT.txt*. Please refer to the files in the Wi-Fi driver package for each module. Different modules require different drivers and RF power parameter files; please do not mix them.

The *wlan_src* directory contains the driver source code.

2.2. Integration and Compilation

This chapter introduces how to port Wi-Fi driver on Linux platform.

NOTE

Unless otherwise stated, *rk3568_linux* mentioned below represents the Linux SDK source code directory.

2.2.1. Code Integration

2.2.1.1. Integrating Driver Code

Step 1: Copy *wlan_src* from the Wi-Fi driver package to the *kernel/drivers/net/wireless/rockchip_wlan/* directory in the Linux code project. The directory structure appear as follows.

```
rk3568_linux/kernel/drivers/net/wireless/rockchip_wlan$ tree -L 1
.
├── cywdhd
├── Kconfig
├── Makefile
├── modules.order
├── mvl88w8977
├── rkwifi
├── rtl8188eu
├── rtl8188fu
├── rtl8189es
├── rtl8189fs
├── rtl8723bs
├── rtl8723bu
├── rtl8723cs
├── rtl8723ds
├── rtl8733bs
├── rtl8821cs
├── rtl8822be
├── rtl8822bs
├── ssv6xxx
├── uwe5621ds
└── wlan_src
```

Step 2: Modify *Makefile* in the *kernel/drivers/net/wireless/rockchip_wlan/* directory of Linux code.

Taking FCS945R module as an example, add compilation rule as shown in the following blue font to the *Makefile* :

```
--- a/Makefile  2023-09-06 15:25:58.453858546 +0800
+++ b/Makefile  2023-09-06 15:25:33.818325928 +0800
@@ -11,3 +11,4 @@
 obj-$(CONFIG_WL_ROCKCHIP)      += rkwifi/rk_wifi_config.o
 obj-$(CONFIG_CYW_BCMDHD)       += cywdhd/
 obj-$(CONFIG_UWE5621DS)        += uwe5621ds/
+obj-$(CONFIG_RTL8733BS) += wlan_src/
```

Other modules use different compilation rules to control driver code compilation. You can refer to the blue font above to modify the corresponding configuration items in the *Makefile* of each module.

- FCU741R module: +obj-$(CONFIG_RTL8733BU) += wlan_src/
- FCS850R series module: +obj-$(CONFIG_RTL8822CS) += wlan_src/
- FCS866R module: +obj-$(CONFIG_RTL8852BS) += wlan_src/
- FCS940R module: +obj-$(CONFIG_RTL8723DS) += wlan_src/
- FCS950R module: +obj-$(CONFIG_RTL8821CS) += wlan_src/

Step 3: Modify *Kconfig* in the*kernel/drivers/net/wireless/rockchip_wlan/* directory of Linux code. The modification is shown in the following blue font.

```
--- a/Kconfig   2023-09-06 15:44:06.514150182 +0800
+++ b/Kconfig   2023-09-06 15:43:56.706163556 +0800
@@ -49,6 +49,7 @@
 source "drivers/net/wireless/rockchip_wlan/rtl8723ds/Kconfig"
 source "drivers/net/wireless/rockchip_wlan/rtl8821cs/Kconfig"
 source "drivers/net/wireless/rockchip_wlan/rtl8822bs/Kconfig"
+source "drivers/net/wireless/rockchip_wlan/wlan_src/Kconfig"
 endif

 source "drivers/net/wireless/rockchip_wlan/mvl88w8977/Kconfig"
```

Step 4: Modify *Kconfig* in the *kernel/drivers/net/wireless/rockchip_wlan/wlan_src/* directory of Linux code to compile the driver as kernel module. The modification is shown in the following blue font.

```
--- a/Kconfig   2023-09-06 15:40:01.150640605 +0800
+++ b/Kconfig   2023-09-06 15:39:47.130680166 +0800
@@ -1,6 +1,7 @@
 config RTL8733BS
        depends on MMC
        tristate "Realtek 8733B SDIO WiFi"
+       default m
        ---help---
```

```
Realtek RTL8733BS chipset driver
802.11ac SDIO wireless network adapter
```

Step 5: Modify *Makefile* in the *kernel/drivers/net/wireless/rockchip_wlan/wlan_src/* directory of Linux code. The modifications are shown in the following blue font.

- FCS850R series, FCS866R, FCS940R, FCS945R and FCS950R modules:

```
--- a/Makefile  2023-09-07 15:01:40.175293443 +0800
+++ b/Makefile  2023-09-07 15:01:22.218964077 +0800
@@ -82,7 +82,7 @@
 CONFIG_TXPWR_BY_RATE = y
 CONFIG_TXPWR_BY_RATE_EN = y
 CONFIG_TXPWR_LIMIT = y
-CONFIG_TXPWR_LIMIT_EN = n
+CONFIG_TXPWR_LIMIT_EN = y
 CONFIG_RTW_CHPLAN = 0xFFFF
 CONFIG_RTW_ADAPTIVITY_EN = disable
 CONFIG_RTW_ADAPTIVITY_MODE = normal
```

- FCU741R module:

```
diff -Naur fcu741r_original/Makefile fcu741r_modified/Makefile
--- fcu741r_original/Makefile      2023-06-20 14:30:08.000000000 +0800
+++ fcu741r_modified/Makefile 2023-07-05 14:30:20.454869758 +0800
@@ -68,8 +68,8 @@
 CONFIG_P2P = y
 CONFIG_MP_INCLUDED = y
 CONFIG_POWER_SAVING = y
-CONFIG_IPS_MODE = 0
-CONFIG_LPS_MODE = 0
+CONFIG_IPS_MODE = default
+CONFIG_LPS_MODE = default
 CONFIG_USB_AUTOSUSPEND = n
 CONFIG_HW_PWRP_DETECTION = n
 CONFIG_BT_COEXIST = y
@@ -81,7 +81,7 @@
 CONFIG_TXPWR_BY_RATE = y
 CONFIG_TXPWR_BY_RATE_EN = y
 CONFIG_TXPWR_LIMIT = y
-CONFIG_TXPWR_LIMIT_EN = n
+CONFIG_TXPWR_LIMIT_EN = y
 CONFIG_RTW_CHPLAN = 0xFF
 CONFIG_RTW_ADAPTIVITY_EN = disable
```

```
 CONFIG_RTW_ADAPTIVITY_MODE = normal
@@ -149,7 +149,7 @@
 #bit0: ROAM_ON_EXPIRED, #bit1: ROAM_ON_RESUME, #bit2: ROAM_ACTIVE
 CONFIG_ROAMING_FLAG = 0x3
 ###################### Platform Related #######################
-CONFIG_PLATFORM_I386_PC = y
+CONFIG_PLATFORM_I386_PC = n
 CONFIG_PLATFORM_ANDROID_X86 = n
 CONFIG_PLATFORM_ANDROID_INTEL_X86 = n
 CONFIG_PLATFORM_JB_X86 = n
@@ -175,6 +175,7 @@
 CONFIG_PLATFORM_ARM_RK2818 = n
 CONFIG_PLATFORM_ARM_RK3066 = n
 CONFIG_PLATFORM_ARM_RK3188 = n
+CONFIG_PLATFORM_ARM_RK3568 = y
 CONFIG_PLATFORM_ARM_URBETTER = n
 CONFIG_PLATFORM_ARM_TI_PANDA = n
 CONFIG_PLATFORM_MIPS_JZ4760 = n
@@ -1741,6 +1742,17 @@
 MODULE_NAME := 8731bu
 endif

+ifeq ($(CONFIG_PLATFORM_ARM_RK3568), y)
+EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_PLATFORM_ANDROID -DCONFIG_PLATFORM_ROCKCHIPS
+# default setting for Android 4.1, 4.2, 4.3, 4.4
+EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
+EXTRA_CFLAGS += -DCONFIG_CONCURRENT_MODE
+EXTRA_CFLAGS += -DCONFIG_RTW_HOSTAPD_ACS
+EXTRA_CFLAGS += -DCONFIG_RTW_IOCTL_SET_COUNTRY
+MODULE_NAME := rtl8731bu
+endif
+
 ifeq ($(CONFIG_PLATFORM_ARM_RK3066), y)
 EXTRA_CFLAGS += -DCONFIG_PLATFORM_ARM_RK3066
        EXTRA_CFLAGS += -DRTW_ENABLE_WIFI_CONTROL_FUNC
```

2.2.1.2. Modifying Kernel Code

For some modules that support SDIO 3.0 or Wi-Fi 6 hotspots, the kernel code needs to be modified accordingly.

1. Modules supporting SDIO 3.0:

FCS850R series and FCS866R modules support SDIO 3.0, and the module's IO voltage needs to be set to 1.8 V. On the current host platform RK3568, the use of 1.8 V cannot be automatically negotiated during the SDIO enumeration process. The following code modification is required to force the switch of the IO voltage to 1.8 V.

Modify the *core.c* in the *kernel/drivers/mmc/core/* directory and *sdio.c* in the *kernel/drivers/mmc/core/* directory of Linux code. The modifications are shown in the following blue font.

```
diff --git a/kernel/drivers/mmc/core/core.c b/kernel/drivers/mmc/core/core.c
index 2e044daaf..5c12229ce 100644
--- a/kernel/drivers/mmc/core/core.c
+++ b/kernel/drivers/mmc/core/core.c
@@ -1486,7 +1486,11 @@ int mmc_set_signal_voltage(struct mmc_host *host, int signal_voltage)
 void mmc_set_initial_signal_voltage(struct mmc_host *host)
 {
	/* Try to set signal voltage to 3.3V but fall back to 1.8v or 1.2v */
-	if (!mmc_set_signal_voltage(host, MMC_SIGNAL_VOLTAGE_330))
+	dev_err(mmc_dev(host), "mmc_power_up: Setting is %d\n",host->index);
+	if (host->index == 2) {
+		mmc_set_signal_voltage(host, MMC_SIGNAL_VOLTAGE_180);
+		dev_err(mmc_dev(host), "mmc_power_up: Setting 1.8V\n");
+	} else if (!mmc_set_signal_voltage(host, MMC_SIGNAL_VOLTAGE_330))
		dev_dbg(mmc_dev(host), "Initial signal voltage of 3.3v\n");
	else if (!mmc_set_signal_voltage(host, MMC_SIGNAL_VOLTAGE_180))
		dev_dbg(mmc_dev(host), "Initial signal voltage of 1.8v\n");

diff --git a/kernel/drivers/mmc/core/sdio.c b/kernel/drivers/mmc/core/sdio.c
index 2dafc562a..77ef71840 100644
--- a/kernel/drivers/mmc/core/sdio.c
+++ b/kernel/drivers/mmc/core/sdio.c
@@ -663,7 +663,14 @@ try_again:
	 * to make sure which speed mode should work.
	 */
	if (!powered_resume && (rocr & ocr & R4_18V_PRESENT)) {
+		//err = mmc_set_uhs_voltage(host, ocr_card);
+		if (host->index == 2) {
+			err=0;
+			pr_err("skippingmmc_set_uhs_voltage \n");
+		} else {
+			err = mmc_set_uhs_voltage(host, ocr_card);
+		}
+
		if (err == -EAGAIN) {
			mmc_sdio_resend_if_cond(host, card);
```

```
        retries--;
```

2. Modules supporting Wi-Fi 6 hotspot:

1) Enabling Wi-Fi 6 hotspot is supported on FCS866R module. As the kernel version of RK3568-WF EVB SDK is V4.19.232, and it lacks the relevant definitions for Wi-Fi 6 hotspot. To support Wi-Fi 6 hotspot, the following modifications in blue font are required in *kernel/include/linux/ieee80211.h* file in Linux code:

```
diff --git a/kernel/include/linux/ieee80211.h b/kernel/include/linux/ieee80211.h
index f3586f0b2..df234565a
--- a/kernel/include/linux/ieee80211.h
+++ b/kernel/include/linux/ieee80211.h
@@ -1554,11 +1554,11 @@ struct ieee80211_vht_operation {
  * struct ieee80211_he_cap_elem - HE capabilities element
  *
  * This structure is the "HE capabilities element" fixed fields as
- * described in P802.11ax_D2.0 section 9.4.2.237.2 and 9.4.2.237.3
+ * described in P802.11ax_D3.0 section 9.4.2.237.2 and 9.4.2.237.3
  */
 struct ieee80211_he_cap_elem {
-       u8 mac_cap_info[5];
-       u8 phy_cap_info[9];
+       u8 mac_cap_info[6];
+       u8 phy_cap_info[11];
 } __packed;

 #define IEEE80211_TX_RX_MCS_NSS_DESC_MAX_LEN     5
@@ -1716,15 +1716,15 @@ struct ieee80211_mu_edca_param_set {
 #define IEEE80211_HE_MAC_CAP1_TF_MAC_PAD_DUR_8US        0x04
 #define IEEE80211_HE_MAC_CAP1_TF_MAC_PAD_DUR_16US       0x08
 #define IEEE80211_HE_MAC_CAP1_TF_MAC_PAD_DUR_MASK           0x0c
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_1       0x00
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_2       0x10
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_3       0x20
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_4       0x30
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_5       0x40
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_6       0x50
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_7       0x60
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_8       0x70
-#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_QOS_MASK        0x70
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_1 0x00
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_2 0x10
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_3 0x20
```

```
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_4  0x30
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_5  0x40
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_6  0x50
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_7  0x60
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_8  0x70
+#define IEEE80211_HE_MAC_CAP1_MULTI_TID_AGG_RX_QOS_MASK 0x70

 /* Link adaptation is split between byte HE_MAC_CAP1 and
  * HE_MAC_CAP2. It should be set only if IEEE80211_HE_MAC_CAP0_HTC_HE
@@ -1738,7 +1738,7 @@ struct ieee80211_mu_edca_param_set {

 #define IEEE80211_HE_MAC_CAP2_LINK_ADAPTATION                  0x01
 #define IEEE80211_HE_MAC_CAP2_ALL_ACK                  0x02
-#define IEEE80211_HE_MAC_CAP2_UL_MU_RESP_SCHED                  0x04
+#define IEEE80211_HE_MAC_CAP2_TRS                  0x04
 #define IEEE80211_HE_MAC_CAP2_BSR                  0x08
 #define IEEE80211_HE_MAC_CAP2_BCAST_TWT                  0x10
 #define IEEE80211_HE_MAC_CAP2_32BIT_BA_BITMAP                  0x20
@@ -1753,22 +1753,32 @@ struct ieee80211_mu_edca_param_set {
  * A-MDPU Length Exponent field in the HT capabilities, VHT capabilities and the
  * same field in the HE capabilities.
  */
-#define IEEE80211_HE_MAC_CAP3_MAX_A_AMPDU_LEN_EXP_USE_VHT   0x00
-#define IEEE80211_HE_MAC_CAP3_MAX_A_AMPDU_LEN_EXP_VHT_1       0x08
-#define IEEE80211_HE_MAC_CAP3_MAX_A_AMPDU_LEN_EXP_VHT_2       0x10
-#define IEEE80211_HE_MAC_CAP3_MAX_A_AMPDU_LEN_EXP_RESERVED      0x18
-#define IEEE80211_HE_MAC_CAP3_MAX_A_AMPDU_LEN_EXP_MASK        0x18
-#define IEEE80211_HE_MAC_CAP3_A_AMSDU_FRAG              0x20
-#define IEEE80211_HE_MAC_CAP3_FLEX_TWT_SCHED                0x40
-#define IEEE80211_HE_MAC_CAP3_RX_CTRL_FRAME_TO_MULTIBSS       0x80
-
-#define IEEE80211_HE_MAC_CAP4_BSRP_BQRP_A_MPDU_AGG        0x01
-#define IEEE80211_HE_MAC_CAP4_QTP                  0x02
-#define IEEE80211_HE_MAC_CAP4_BQR                  0x04
-#define IEEE80211_HE_MAC_CAP4_SR_RESP                  0x08
-#define IEEE80211_HE_MAC_CAP4_NDP_FB_REP                  0x10
-#define IEEE80211_HE_MAC_CAP4_OPS                  0x20
-#define IEEE80211_HE_MAC_CAP4_AMDSU_IN_AMPDU                0x40
+#define IEEE80211_HE_MAC_CAP3_MAX_AMPDU_LEN_EXP_USE_VHT      0x00
+#define IEEE80211_HE_MAC_CAP3_MAX_AMPDU_LEN_EXP_VHT_1    0x08
+#define IEEE80211_HE_MAC_CAP3_MAX_AMPDU_LEN_EXP_VHT_2    0x10
+#define IEEE80211_HE_MAC_CAP3_MAX_AMPDU_LEN_EXP_RESERVED  0x18
+#define IEEE80211_HE_MAC_CAP3_MAX_AMPDU_LEN_EXP_MASK     0x18
+#define IEEE80211_HE_MAC_CAP3_AMSDU_FRAG     0x20
```

```
+#define IEEE80211_HE_MAC_CAP3_FLEX_TWT_SCHED    0x40
+#define IEEE80211_HE_MAC_CAP3_RX_CTRL_FRAME_TO_MULTIBSS 0x80
+
+#define IEEE80211_HE_MAC_CAP4_BSRP_BQRP_A_MPDU_AGG  0x01
+#define IEEE80211_HE_MAC_CAP4_QTP 0x02
+#define IEEE80211_HE_MAC_CAP4_BQR 0x04
+#define IEEE80211_HE_MAC_CAP4_SRP_RESP    0x08
+#define IEEE80211_HE_MAC_CAP4_NDP_FB_REP      0x10
+#define IEEE80211_HE_MAC_CAP4_OPS 0x20
+#define IEEE80211_HE_MAC_CAP4_AMDSU_IN_AMPDU    0x40
+/* Multi TID agg TX is split between byte #4 and #5
+ * The value is a combination of B39,B40,B41
+ */
+#define IEEE80211_HE_MAC_CAP4_MULTI_TID_AGG_TX_QOS_B39     0x80
+
+#define IEEE80211_HE_MAC_CAP5_MULTI_TID_AGG_TX_QOS_B40     0x01
+#define IEEE80211_HE_MAC_CAP5_MULTI_TID_AGG_TX_QOS_B41     0x02
+#define IEEE80211_HE_MAC_CAP5_SUBCHAN_SELECVITE_TRANSMISSION    0x04
+#define IEEE80211_HE_MAC_CAP5_UL_2x996_TONE_RU  0x08
+#define IEEE80211_HE_MAC_CAP5_OM_CTRL_UL_MU_DATA_DIS_RX 0x10

 /* 802.11ax HE PHY capabilities */
 #define IEEE80211_HE_PHY_CAP0_DUAL_BAND                         0x01
@@ -1789,9 +1799,9 @@ struct ieee80211_mu_edca_param_set {
 #define IEEE80211_HE_PHY_CAP1_LDPC_CODING_IN_PAYLOAD              0x20
 #define IEEE80211_HE_PHY_CAP1_HE_LTF_AND_GI_FOR_HE_PPDUS_0_8US        0x40
 /* Midamble RX Max NSTS is split between byte #2 and byte #3 */
-#define IEEE80211_HE_PHY_CAP1_MIDAMBLE_RX_MAX_NSTS                0x80
+#define IEEE80211_HE_PHY_CAP1_MIDAMBLE_RX_TX_MAX_NSTS                0x80

-#define IEEE80211_HE_PHY_CAP2_MIDAMBLE_RX_MAX_NSTS                0x01
+#define IEEE80211_HE_PHY_CAP2_MIDAMBLE_RX_TX_MAX_NSTS                0x01
 #define IEEE80211_HE_PHY_CAP2_NDP_4x_LTF_AND_3_2US             0x02
 #define IEEE80211_HE_PHY_CAP2_STBC_TX_UNDER_80MHZ              0x04
 #define IEEE80211_HE_PHY_CAP2_STBC_RX_UNDER_80MHZ              0x08
@@ -1892,7 +1902,19 @@ struct ieee80211_mu_edca_param_set {
 #define IEEE80211_HE_PHY_CAP8_20MHZ_IN_160MHZ_HE_PPDU            0x04
 #define IEEE80211_HE_PHY_CAP8_80MHZ_IN_160MHZ_HE_PPDU            0x08
 #define IEEE80211_HE_PHY_CAP8_HE_ER_SU_1XLTF_AND_08_US_GI       0x10
-#define IEEE80211_HE_PHY_CAP8_MIDAMBLE_RX_2X_AND_1XLTF             0x20
+#define IEEE80211_HE_PHY_CAP8_MIDAMBLE_RX_TX_2X_AND_1XLTF          0x20
+#define IEEE80211_HE_PHY_CAP8_DCM_MAX_BW_20MHZ                     0x00
+#define IEEE80211_HE_PHY_CAP8_DCM_MAX_BW_40MHZ                     0x40
+#define IEEE80211_HE_PHY_CAP8_DCM_MAX_BW_80MHZ                     0x80
```

```
+#define IEEE80211_HE_PHY_CAP8_DCM_MAX_BW_160_OR_80P80_MHZ       0xc0
+#define IEEE80211_HE_PHY_CAP8_DCM_MAX_BW_MASK                   0xc0
+
+#define IEEE80211_HE_PHY_CAP9_LONGER_THAN_16_SIGB_OFDM_SYM      0x01
+#define IEEE80211_HE_PHY_CAP9_NON_TRIGGERED_CQI_FEEDBACK        0x02
+#define IEEE80211_HE_PHY_CAP9_TX_1024_QAM_LESS_THAN_242_TONE_RU     0x04
+#define IEEE80211_HE_PHY_CAP9_RX_1024_QAM_LESS_THAN_242_TONE_RU     0x08
+#define IEEE80211_HE_PHY_CAP9_RX_FULL_BW_SU_USING_MU_WITH_COMP_SIGB
     0x10
+#define    IEEE80211_HE_PHY_CAP9_RX_FULL_BW_SU_USING_MU_WITH_NON_COMP_SIGB
         0x20

 /* 802.11ax HE TX/RX MCS NSS Support  */
 #define IEEE80211_TX_RX_MCS_NSS_SUPP_HIGHEST_MCS_POS          (3)
```

2) To support enabling Wi-Fi 6 hotspot on FCS866R module, in addition to modifying the aforementioned kernel code, further modifications of driver code are required in *platform/arm_rk.mk* located in *kernel/drivers/net/wireless/rockchip_wlan/wlan_src/* directory of the Linux code. The modifications are shown in the following blue font.

```
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_PLATFORM_ANDROID
 EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
 EXTRA_CFLAGS += -DCONFIG_RADIO_WORK
 EXTRA_CFLAGS += -DCONFIG_CONCURRENT_MODE
+EXTRA_CFLAGS += -DCPTCFG_VERSION
 ifeq ($(shell test $(CONFIG_RTW_ANDROID) -ge 11; echo $$?), 0)
 EXTRA_CFLAGS += -DCONFIG_IFACE_NUMBER=2
 #EXTRA_CFLAGS += -DCONFIG_PLATFORM_ROCKCHIPS
```

NOTE

When using FCS866R module, if the host controller's kernel version is equal to or greater than 4.20, there is no need to modify the kernel and driver code as mentioned above.

2.2.2. Compilation and Flashing

2.2.2.1. Compiling Code

Execute the following commands in the *rk3568_linux* directory to compile the code:

```
source build-quec.sh
```

```
source envsetup.sh rockchip_rk3568
buildclean
build-all-image
```

update.img is generated in the *rk3568_linux/rockdev* directory after a successful compilation.

```
rk3568_linux/rockdev$ tree -L 1
.
├── boot.img -> /home/quectel/share/wifi/bpi-rk/sg368/fcs945/RK3568/kernel/boot.img
├── MiniLoaderAll.bin -> ../u-boot/rk356x_spl_loader_v1.16.112.bin
├── misc.img -> ../device/rockchip/rockimg/blank-misc.img
├── nvdata1.img -> ../device/rockchip/rockimg/blank-misc.img
├── nvdata2.img -> ../device/rockchip/rockimg/blank-misc.img
├── oem.img
├── parameter.txt -> ../device/rockchip/rk356x/parameter-buildroot-fit.txt
├── persist.img
├── recovery.img -> ../buildroot/output/rockchip_rk356x_recovery/images/recovery.img
├── rootfs.ext4 -> ../buildroot/output/rockchip_rk3568/images/rootfs.ext2
├── rootfs.img -> ../buildroot/output/rockchip_rk3568/images/rootfs.ext2
├── uboot.img -> ../u-boot/uboot.img
├── update.img
└── userdata.img
```

And also a .ko file is generated in the *rk3568_linux/kernel/drivers/net/wireless/rockchip_wlan/wlan_src* directory. Taking FCS945R module as an example, *8733bs.ko is generated after compilation, as shown below:*

```
~rk3568_linux$ ls kernel/drivers/net/wireless/rockchip_wlan/wlan_src/8733bs.ko
kernel/drivers/net/wireless/rockchip_wlan/wlan_src/8733bs.ko
```

Different .ko files are generated for different Wi-Fi modules. As shown below:

- FCU741R module: *8731bu.ko*
- FCS850R series module: *8822cs.ko*
- FCS866R module: *8852bs.ko*
- FCS940R module: *8723ds.ko*
- FCS945R module: *8733bs.ko*
- FCS950R module: *8821cs.ko*

2.2.2.2. Flashing Image

Step 1: Connect the EVB to PC through a USB to Type-C data cable. Open RKDevTool and select "**Upgrade Firmware**" menu on tool interface.

Step 2: Click "**Fimware**" to select and import *update.img*.

Step 3: Set the "BOOT" switch on the RK3568-WF EVB to "ON." After connecting the power supply, press the "**PWRKET**" button on the SG368Z-WF-EVB. The RK3568-WF EVB will automatically enter "LOADER" mode. At this point, the tool interface will display "Found One LOADER Device".

Step 4: Click "**Upgrade**" on the tool interface to start flashing the image.

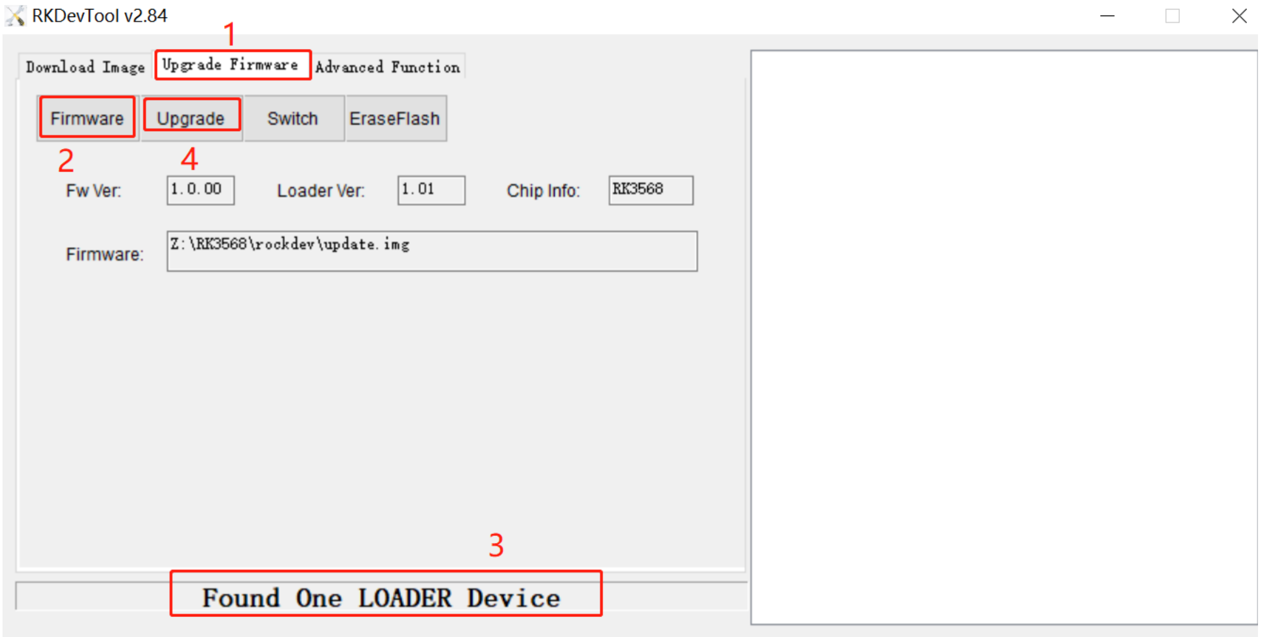

Figure 5: Flash Image

After the image is flashed successfully, the following interface displays:

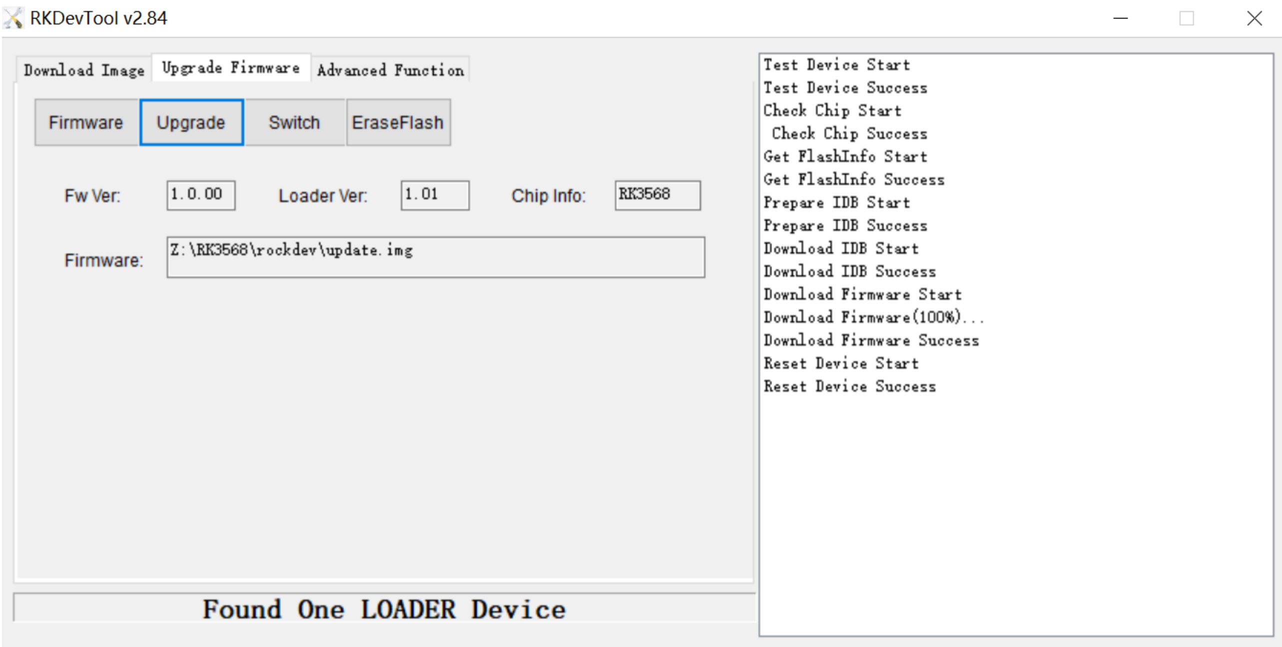

Figure 6: Flashing Image is Completed

Step 5: Set the "BOOT" switch on the RK3568-WF EVB back to "OFF," and then restart the EVB.

2.3. Driver Loading

Step 1: check whether the module is enumerated properly before loading the Wi-Fi driver.

1) FCU741R module (USB interface):

Execute **lsusb**. If the USB device whose ID is 0bda:f72b is displayed, the module is enumerated successfully. Otherwise, check the hardware connection and platform configuration.

```
root@rockchip:/# lsusb
Bus 005 Device 001: ID 1d6b:0002
Bus 003 Device 001: ID 1d6b:0001
Bus 002 Device 002: ID 0bda:f72b
```

2) FCS850R series, FCS866R, FCS940R, FCS945R and FCS950R modules (SDIO interface):

Execute **ls -l /sys/bus/sdio/devices/**. If the mmc2 device is displayed as follows, is enumerated successfully. Otherwise, check the hardware connection and platform configuration.

```
root@rockchip:/# ls -l /sys/bus/sdio/devices/
total 0
lrwxrwxrwx 1 root root 0 Jan 1 00:00 mmc2:0001:1 -
> ../../../devices/platform/fe2c0000.dwmmc/mmc_host/mmc2/mmc2:0001/mmc2:0001:1
```

After the module is successfully enumerated, you can proceed with the following operations.

Step 2: Push .ko file to the *mnt* directory of the EVB.

For example, to push *8733bs.ko* to the *mnt* directory of the EVB, the command: is as follows:

```
D:\adb_scrcpy>adb push 8733bs.ko /mnt
```

Step 3: Execute the following commands to push the RF power parameter files in *txpower*to the *lib/firmware* directory of the EVB.

- If a FCU741R, FCS850R series, FCS940R, FCS945R or FCS950R module is used, you need to push *PHY_REG_PG.tx*t and *TXPWR_LMT.txt* to the *lib/firmware* directory of the EVB. The commands are as follows:

```
adb push  PHY_REG_PG.txt  /lib/firmware/
adb push  TXPWR_LMT.txt  /lib/firmware/
```

- If a FCS866R module is used, you need to push *TXPWR_ByRate.txt* and *TXPWR_LMT.txt* to the *lib/firmware* directory of the EVB. The commands are as follows:

```
adb push  TXPWR_ByRate.txt  /lib/firmware/
adb push  TXPWR_LMT.txt  /lib/firmware/
```

The driver reads the RF power parameter files from the *lib/firmware/* directory of the EVB by default. If you need to change the file loading path, please modify the *Makefile* in the *rk3568_linux/kernel/drivers/net/wireless/rockchip_wlan/wlan_src/* directory as shown in the following blue font:

```
ifeq ($(CONFIG_LOAD_PHY_PARA_FROM_FILE), y)
EXTRA_CFLAGS += -DCONFIG_LOAD_PHY_PARA_FROM_FILE
#EXTRA_CFLAGS += -DREALTEK_CONFIG_PATH_WITH_IC_NAME_FOLDER
EXTRA_CFLAGS += -DREALTEK_CONFIG_PATH=\"/lib/firmware/\"
endif
```

Step 4: If you need to perform RF non-signaling tests, push the non-signaling test tool to the *usr/bin/* directory of the EVB and grant executable permissions to the tool. For example, to use *rtwpriv_arm64*, the commands are as follows:

```
adb push  rtwpriv_arm64  /usr/bin/
adb shell
root@rockchip:/# chmod +x /usr/bin/rtwpriv_arm64
```

Step 5: Load the driver.

Taking FCS945R module as an example, execute the following command to load the driver. *rtw_country_code=CN* in the command specifies the country code as China. Please load the correct country code based on the region where the module is intended to be used

```
root@rockchip:/# insmod  /mnt/8733bs.ko rtw_country_code = CN
```

Then execute **lsmod** to check whether the module driver is loaded successfully. If 8733bs is displayed as follows, the driver of the FCS945R module is loaded successfully.

```
root@rockchip:/# lsmod
Module                  Size  Used by
8733bs              2469888  0
```

Step 6: Execute **dmesg** to view the printed kernel log.

The driver version and chip information, as well as the RF power parameter file reading process will be printed during the driver loading process.

The following is an example of the printed driver version and chip information:

```
root@rockchip:/# dmesg
[    6.151274] RTW: module init start
[    6.151319] RTW: rtl8733bs v5.14.1.1-29-ga4429ac3f.20230614_COEX20211210-2706
[    6.151326] RTW: build time: Sep  7 2023 14:22:55
[    6.151331] RTW: rtl8733bs BT-Coex version = COEX20211210-2706
[    6.151385] RTW: rtw_inetaddr_notifier_register
[    6.152998] RTW: == SDIO Card Info ==
[    6.153023] RTW:   card: 000000006faf3a2d
[    6.153028] RTW:   clock: 50000000 Hz
[    6.153033] RTW:   timing spec: sd high-speed
[    6.153042] RTW:   sd3_bus_mode: FALSE
[    6.153046] RTW:   func num: 1
[    6.153052] RTW:   func1: 00000000de5ff045 (*)
[    6.153057] RTW: ================
[    6.153063] RTW: CHIP TYPE: RTL8733B
```

The following is an example of the process of reading the RF power parameter files. If the following log is displayed, the RF power parameter files are successfully loaded. Ensure that the RF power parameter files are loaded successfully. Otherwise, the output power of the module will be affected.

```
[    6.421123] RTW: retriveFromFile openFile path:/vendor/etc/firmware/PHY_REG_PG.txt fp=0000000059616321
[    6.421182] RTW: retriveFromFile readFile, ret:1406
[    6.421396] RTW: phy_ParseBBPgParaFile return 1
[    6.448244] RTW: retriveFromFile openFile path:/vendor/etc/firmware/TXPWR_LMT.txt fp=00000000637aa60a
[    6.448300] RTW: retriveFromFile readFile, ret:8055
[    6.450060] RTW: phy_ParsePowerLimitTableFile return 1
```

Step 7: Execute **ifconfig wlan0** to check whether wlan0 interface is enabled successfully.

If the information of wlan0 interface is displayed as follows, it indicates that the Wi-Fi driver and Wi-Fi firmware have been loaded successfully, and the Wi-Fi functionality is normal.

```
root@rockchip:/# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:E0:4C:15:5E:1F
          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)
```

2.4. Non-signaling Test

After the driver is loaded successfully, you can use rtwpriv_arm or rtwpriv_arm64 for non-signaling tests. Taking rtwpriv_arm64 as an example, you can execute the following command to confirm whether the tool can be used correctly.

```
root@rockchip:/mnt# rtwpriv_arm64 wlan0 efuse_get realmap
```

If the following information is returned, it indicates that the non-signaling tool is functioning correctly, and you can proceed with non-signaling tests. For detailed steps for non-signaling test, please contact Quectel Technical Support.

```
wlan0    efuse_get:
0x00    29 81 80 86 10 00 84 00        B2 0D B5 14 08 94 8A 1B
0x10    FF FF FF FF FF FF FF FF        FF FF FF FF FF FF FF FF
```

2.5. Function Verification

This chapter uses the RK3568-WF EVB installed with a Wi-Fi module as an example to describe how to verify the Wi-Fi function of the module.

2.5.1. Soft AP Mode

Using hostapd to enable Soft AP involves the configuration of encryption mode and wireless mode, which are independent and can be combined freely.

2.5.1.1. Open Mode

Taking enabling an unencrypted 2.4 GHz hotspot whose SSID is "TETS_WIFI", Wi-Fi standard is IEEE 802.11n, bandwidth is 20 MHz and channel is 6 as an example, the steps are as follows:

Step 1: Create a configuration file named *hostapd.conf* in the *etc* directory of the EVB and add the following contents to the file.

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
```

```
ssid=TEST_WIFI
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[HT20][SHORT-GI-20]
```

Step 2: Create a configuration file named *dnsmasq.conf* in the *etc* directory of the EVB and add the following contents to the file.

```
user=root
listen-address=192.168.11.1
dhcp-range=192.168.11.2,192.168.11.20
server=/google/8.8.8.8
```

Step 3: Execute the following command to enable thehotspot.

```
hostapd  /etc/hostapd.conf    -dd  &
ifconfig wlan0 192.168.11.1
systemctl stop systemd-resolved
dnsmasq -i wlan0 -C etc/dnsmasq.conf  &
```

Step 4: Connect your mobile phone to the hotspot. If your mobile phone successfully connects to TEST_WIFI, it indicates that the module works normally in AP mode.

Figure 7: TEST_WIFI Mobile Phone Connects to TEST_WIFI Successfully

2.5.1.2. Encryption Mode

2.5.1.2.1. WPA2 Encryption Mode

Taking enabling a WPA2 encrypted 2.4 GHz hotspot whose SSID is "TETS_WIFI", password is 12345678, Wi-Fi standard is IEEE 802.11n, bandwidth is 20 MHz, and channel is 6 as an example, the steps are as follows:

Create a configuration file named *hostapd.conf* in the *etc* directory of the EVB and add the following contents to the file. The contents in blue font are the WPA2 configurations.

```
interface=wlan0
driver=nl80211
```

```
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[HT20][SHORT-GI-20]

auth_algs=3
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
```

Follow the ***Steps 2*** – ***4*** in ***Chapter 2.5.1.1*** for subsequent verification.

2.5.1.2.2. WPA2/WPA3 Transition Mode

Taking enabling a WPA2/WPA3 encrypted 2.4 GHz hotspot whose SSID is "TETS_WIFI", password is 12345678, Wi-Fi standard is IEEE 802.11n, bandwidth is 20 MHz, and channel is 6 as an example, the steps are as follows:

Create a configuration file named *hostapd.conf* in the *etc* directory of the EVB and add the following contents to the file. The contents in blue font are the WPA2/WPA3 configurations.

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[HT20][SHORT-GI-20]

auth_algs=3
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK SAE
rsn_pairwise=CCMP
ieee80211w=1
```

```
sae_pwe=2
```

Follow the ***Steps 2*** – ***4*** in ***Chapter 2.5.1.1*** for subsequent verification.

2.5.1.3. Wireless Mode with Open Mode

This chapter describes how to configure and verify the common wireless modes with open mode.

2.5.1.3.1. IEEE 802.11n Wireless Mode

IEEE 802.11n wireless mode, also known as HT mode, supports 2.4 GHz and 5 GHz bands, 20 MHz and 40MHz bandwidths. And the common hostapd configurations for IEEE 802.11n wireless mode are as follows:

Create a configuration file named *hostapd.conf* in the *etc* directory of the EVB and add the following contents to the file.

1. Taking enabling an unencrypted 2.4 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 20 MHz/40 MHz, and channel is 6 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=6
hw_mode=g
ieee80211n=1
ht_capab=[HT20][SHORT-GI-20][HT40-][HT40+][SHORT-GI-40]
```

2. Taking enabling an unencrypted 2.4 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 20 MHz, and channel is 6 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI
```

```
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[HT20][SHORT-GI-20]
```

3. Taking enabling an unencrypted 5 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 20 MHz/40 MHz, and channel is 149 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=149
hw_mode=a
ieee80211n=1
ht_capab=[HT20][SHORT-GI-20][HT40-][HT40+][SHORT-GI-40]
```

Follow the ***Steps 2*** – ***4*** in ***Chapter 2.5.1.1*** for subsequent verification.

2.5.1.3.2. IEEE 802.11ac Wireless Mode

IEEE 802.11ac wireless mode, also known as VHT mode, generally supports only 5 GHz band, supports 20 MHz, 40 MHz, 80 MHz and 160 MHz bandwidths. And the common hostapd configurations for IEEE 802.11ac wireless mode are as follows:

Create a configuration file named *hostapd.conf* in the *etc* directory of the EVB and add the following contents to the file.

1. Taking enabling an unencrypted 5 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 80 MHz, and channel is 36 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=36
```

```
hw_mode=a
ieee80211n=1
ieee80211ac=1
ht_capab=[HT20][HT40+][HT40-][SHORT-GI-40][SHORT-GI-20]
vht_capab=[SHORT-GI-80]
vht_oper_chwidth=1
vht_oper_centr_freq_seg0_idx=42
```

2. Taking enabling an unencrypted 5 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 80 MHz, and channel is 149 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=149
hw_mode=a
ieee80211n=1
ieee80211ac=1
ht_capab=[HT20][HT40+][HT40-][SHORT-GI-40][SHORT-GI-20]
vht_capab=[SHORT-GI-80]
vht_oper_chwidth=1
vht_oper_centr_freq_seg0_idx=155
```

Follow the ***Steps 2*** – ***4*** in ***Chapter 2.5.1.1*** for subsequent verification.

2.5.1.3.3. IEEE 802.11ax Wireless Mode

IEEE 802.11ax wireless mode, also known as HE mode, supports 2.4 GHz and 5 GHz bands, and supports 20 MHz, 40 MHz, 80 MHz and 160 MHz bandwidths. And the common hostapd configurations for IEEE 802.11ax wireless mode are as follows:

Create a configuration file named *hostapd.conf* in the *etc* directory of the EVB and add the following contents to the file.

1. Taking enabling an unencrypted 2.4 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 20 MHz/40MHz, and channel is 6 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
```

```
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=6
hw_mode=g
ieee80211n=1
ieee80211ax=1
ht_capab=[HT20][SHORT-GI-20][HT40-][HT40+][SHORT-GI-40]
he_basic_mcs_nss_set=65531
```

2. Taking enabling an unencrypted 2.4 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 20 MHz, and channel is 6 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=6
hw_mode=g
ieee80211n=1
ieee80211ax=1
ht_capab=[HT20][SHORT-GI-20]
he_basic_mcs_nss_set=65531
```

3. Taking enabling an unencrypted 5 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 80 MHz, and channel is 36 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=36
hw_mode=a
ieee80211n=1
ieee80211ac=1
ieee80211ax=1
```

```
ht_capab=[HT20][HT40+][HT40-][SHORT-GI-40][SHORT-GI-20]
vht_capab=[SHORT-GI-80]
vht_oper_chwidth=1
he_oper_chwidth=1
vht_oper_centr_freq_seg0_idx=42
he_oper_centr_freq_seg0_idx=42
he_basic_mcs_nss_set=65531
```

4. Taking enabling an unencrypted 5 GHz hotspot whose SSID is "TETS_WIFI", bandwidth is 80 MHz, and channel is 149 as an example, the file contents are as follows:

```
interface=wlan0
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=149
hw_mode=a
ieee80211n=1
ieee80211ac=1
ieee80211ax=1
ht_capab=[HT20][HT40+][HT40-][SHORT-GI-40][SHORT-GI-20]
vht_capab=[SHORT-GI-80]
vht_oper_chwidth=1
he_oper_chwidth=1
vht_oper_centr_freq_seg0_idx=155
he_oper_centr_freq_seg0_idx=155
he_basic_mcs_nss_set=65531
```

Follow the ***Steps 2*** – ***4*** in ***Chapter 2.5.1.1*** for subsequent verification.

2.5.1.4. Wireless Mode with Encryption Mode

To use wireless mode with encryption mode, create a configuration file *hostapd.conf* and add the corresponding encryption mode and wireless mode configurations to the corresponding location in the file. You can refer to the encryption mode and wireless mode configurations as shown in the bule font in the configuration files described in ***Chapter 2.5.1.2*** and ***Chapter 2.5.1.3***. Taking enabling a WPA2 encrypted 5 GHz hotspot whose SSID is "TETS_WIFI", password is 12345678, bandwidth is 80 MHz, and channel is 149 as an example, the *hostapd.conf* contains the following information:

```
interface=wlan0
```

```
driver=nl80211
ctrl_interface_group=0
ctrl_interface=/var/run/hostapd
beacon_int=100
ssid=TEST_WIFI

channel=149
hw_mode=a
ieee80211n=1
ieee80211ac=1
ieee80211ax=1
ht_capab=[HT20][HT40+][HT40-][SHORT-GI-40][SHORT-GI-20]
vht_capab=[SHORT-GI-80]
vht_oper_chwidth=1
he_oper_chwidth=1
vht_oper_centr_freq_seg0_idx=155
he_oper_centr_freq_seg0_idx=155
he_basic_mcs_nss_set=65531

auth_algs=3
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
```

Follow the ***Steps 2*** – ***4*** in ***Chapter 2.5.1.1*** for subsequent verification.

NOTE

1. This chapter lists only common hostapd configurations. If you need to enable other configurations, you can add the relevant configurations by yourself or contact Quectel Technical Support.
2. Before enabling the AP mode, check the Wi-Fi standards supported by the module. For example, only the FCS866R module support IEEE802.11ax hotspots.

2.5.2. STA Mode

2.5.2.1. Connecting AP in Open Mode

Taking connecting to an unencrypted AP hotspot whose SSID is "TESTAP_2G" as an example, the steps are as follows:

Step 1: Create a configuration file named *wpa_supplicant.conf* in the *etc* directory of the EVB and add the following contents to the file.

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
network={
    ssid="TESTAP_2G"
   key_mgmt=NONE
}
```

Step 2: Execute the following commands to connect the AP hotspot.

```
wpa_supplicant -Dnl80211 -i wlan0 -c /etc/wpa_supplicant.conf &
udhcpc -i wlan0
```

Step 3: Execute **wpa_cli -i wlan0 status** to check the connection status. If wpa_state=COMPLETED is displayed, it indicates that the module connects to the AP hotspot successfully.

```
root@rockchip:/mnt# wpa_cli -i wlan0 status
bssid=1a:19:df:97:26:1e
freq=2467
ssid=TESTAP_2G
id=0
mode=station
wifi_generation=4
pairwise_cipher=NONE
group_cipher=NONE
key_mgmt=NONE
wpa_state=COMPLETED
ip_address=192.168.15.114
p2p_device_address=00:e0:4c:7e:76:51
address=00:e0:4c:7e:76:51
uuid=09c03d49-e1e4-5f2d-aa76-acc44070aeb9
```

2.5.2.2. Connecting AP in Encryption Mode

Taking connecting to an unencrypted AP hotspot whose SSID is “TESTAP_2G” and password is 12345678 as an example, the steps are as follows:

Step 1: Create a configuration file named *wpa_supplicant.conf* in the *etc* directory of the EVB and add the following contents to the file.

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
network={
```

```
    ssid="TESTAP_2G"
    psk="12345678"
}
```

Step 2: Execute the following commands to connect the AP hotspot.

```
wpa_supplicant -Dnl80211 -i wlan0 -c /etc/wpa_supplicant.conf &
udhcpc -i wlan0
```

Step 3: Execute **wpa_cli -i wlan0 status** to check the connection status. If wpa_state=COMPLETED is displayed, it indicates that the module connects to the AP hotspot successfully.

3 Android Platfotm

3.1. Environment Preparations

3.1.1. Hardware Environment

Please follow the hardware environment in ***Chapter 2.1.1***.

3.1.2. Software Environment

Table 7: Android Software Environment

Type	Description
Code environment	Android 11 kernel-4.19.232
Driver package	Quectel Wi-Fi driver package
Compilation environment	Ubuntu 18.04
Tool	Install RKDevTool on Windows PC

NOTE

If you are developing and debugging the module based on Quectel RK3568-WF EVB, please contact Quectel Technical Support to obtain the corresponding Android 11 source code package.

3.2. Integration and Compilation

This chapter describes how to port the Wi-Fi driver on the Android platform.

3.2.1. Code Integration

3.2.1.1. Porting Wi-Fi Driver

Part of the code integration operations of the Android platform and Linux platform are the same. Please refer to ***Chapter 2.2.1*** to integrate the driver code and modify the kernel code, and then perform the following operations:

Step 1: Modify *Makefile* in the *kernel/drivers/net/wireless/rockchip_wlan/wlan_src/*directory of Android source code package.

1) Configure the *CONFIG_RTW_ANDROID* definition in the *Makefile* according to the actual Android v ersion. Taking RK3568-WF EVB Android 11 as an example, *CONFIG_RTW_ANDROID=11* is requir ed, as shown in the following blue font:

```
 ############################ Android ############################
 # CONFIG_RTW_ANDROID - 0: no Android, 4/5/6/7/8/9/10/11 : Android version
-CONFIG_RTW_ANDROID = 0
+CONFIG_RTW_ANDROID = 11

 ifeq ($(shell test $(CONFIG_RTW_ANDROID) -gt 0; echo $$?), 0)
 EXTRA_CFLAGS += -DCONFIG_RTW_ANDROID=$(CONFIG_RTW_ANDROID)
```

2) As there is no */lib/firmware/* directory in the RK3568-WF EVB Android 11 file system, the path from w hich the driver reads the RF power parameter files in the *Makefile* needs to be changed to the */vend or/etc/firmware/* directory, as shown in the following blue font:

```
  ifeq ($(CONFIG_LOAD_PHY_PARA_FROM_FILE), y)
 EXTRA_CFLAGS += -DCONFIG_LOAD_PHY_PARA_FROM_FILE
 #EXTRA_CFLAGS += -DREALTEK_CONFIG_PATH_WITH_IC_NAME_FOLDER
-EXTRA_CFLAGS += -DREALTEK_CONFIG_PATH=\"/lib/firmware/\"
+EXTRA_CFLAGS += -DREALTEK_CONFIG_PATH=\"/vendor/etc/firmware/\"
 endif
```

Step 2: Copy the RF power parameter files *TXPWR_LMT.txt* and *PHY_REG_PG.txt* (or *TXPWR_LMT.t x* and *TXPWR_ByRate.txt*) to the *vendor/rockchip/common/wifi/firmware* directory of Android source code package. These two files are automatically packaged into the *vendor/etc/firmware/* directory of the file system.

Step 3: If you need to perform a RF non-signaling test, you can copy the non-signal testing tools rtwpriv_arm or rtwpriv_arm64 to the *vendor/rockchip/common/wifi/* directory of Android 11 source package. Taking rtwpriv_arm64 as an example, you can refer to the following blue font to modify *vendor/rockchip/common/wifi/wifi.mk* to package the rtwpriv_arm64 to the file system.

```
--- a/wifi.mk   2023-09-07 13:56:21.687402075 +0800
+++ b/wifi.mk   2023-09-07 14:39:02.926397780 +0800
@@ -5,7 +5,8 @@

 PRODUCT_COPY_FILES += \
        $(CUR_PATH)/wifi/iwconfig:$(TARGET_COPY_OUT_VENDOR)/bin/iwconfig \
     $(CUR_PATH)/wifi/iwlist:$(TARGET_COPY_OUT_VENDOR)/bin/iwlist
+       $(CUR_PATH)/wifi/rtwpriv_arm64:$(TARGET_COPY_OUT_VENDOR)/bin/rtwpriv_arm64

 WifiFirmwareFile := $(shell ls $(CUR_PATH)/wifi/firmware)
 PRODUCT_COPY_FILES += \
```

NOTE

If Android version is Android 11 or later and the Kernel version is Kernel 5.4 or later, the Wi-Fi driver will call the following API of the Linux Firmware subsystem to read the RF power parameter file from user space:
*int request_firmware(const struct firmware **fw, const char *name,struct device *device)*

request_firmware() reads the file with the specified name (specified by parameter *name*) from several directories defined by the Linux Firmware subsystem. Therefore, you need to store the RF power parameter file in one of these directories.

For directories defined by the Linux Firmware subsystem, see *fw_path* structure in *kernel/drivers/base/firmware_loader/main c*, and you can also add a directory in the structure.

For example, adding the directory */vendor/firmware/*, the code example is as follows:

```
static const char * const fw_path[] = {
       fw_path_para,
       "/lib/firmware/updates/" UTS_RELEASE,
       "/lib/firmware/updates",
       "/lib/firmware/" UTS_RELEASE,
       "/lib/firmware",
      "/vendor/firmware/"
};
```

Then package the RF power parameter file to */vendor/firmware/* directory in the file system. *request_firmware()* will read the file according to the file name. You do not need to perform the operation as described in ***2)*** of ***Step 1*** mentioned above, just keep the following definition:

```
EXTRA_CFLAGS += -DREALTEK_CONFIG_PATH=\"/lib/firmware/\"
```

3.2.1.2. Modifying Framework Code

When the Wi-Fi is started up on RK3568-WF EVB Android 11 system, the system identifies the installed module based on the module's VID and PID (vendor ID and device ID), and loads the driver corresponding to the module. To achieve this function, you need to modify the *rk_wifi_ctrl.cpp in the frameworks/opt/net/wifi/libwifi_hal/* directory of the Android source code package. The modifications are shown in the following blue font.

```
diff --git a/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp
b/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp
index dfc5aa11dc..5e8168acfc 100755
--- a/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp
+++ b/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp
@@ -68,6 +68,14 @@ static wifi_device supported_wifi_devices[] = {
       {"AP6335",   "02d0:4335"},
       {"AP6255",    "02d0:a9bf"},
       {"RTL8822BE",   "10ec:b822"},
+      {"RTL8731BU",  "0bda:f72b"}, //FCU741R
+      {"RTL8822CS",  "024c:c822"}, //FCS850R
+      {"RTL8852BS",  "024c:b852"}, //FCS866R
+      {"RTL8723DS",  "024c:d723"}, //FCS940R
+      {"RTL8733BS",  "024c:b733"}, //FCS945R
+      {"RTL8821CS",  "024c:c821"}, //FCS945R
       {"MVL88W8977", "02df:9145"},
       {"SPRDWL", "0000:0000"},
 };
diff --git a/frameworks/opt/net/wifi/libwifi_hal/wifi_hal_common.cpp
b/frameworks/opt/net/wifi/libwifi_hal/wifi_hal_common.cpp
index 07029c7677..c791fcc9cd 100755
--- a/frameworks/opt/net/wifi/libwifi_hal/wifi_hal_common.cpp
+++ b/frameworks/opt/net/wifi/libwifi_hal/wifi_hal_common.cpp
@@ -54,6 +54,16 @@ extern "C" int delete_module(const char *, unsigned int);
 #define MVL_DRIVER_MODULE_PATH      WIFI_MODULE_PATH"sd8xxx.ko"
 #define RK912_DRIVER_MODULE_PATH      WIFI_MODULE_PATH"rk912.ko"
 #define SPRDWL_DRIVER_MODULE_PATH     WIFI_MODULE_PATH"sprdwl_ng.ko"
+
+#define RTL8731BU_DRIVER_MODULE_PATH    WIFI_MODULE_PATH"8731bu.ko" //FCU741R
+#define RTL8822CS_DRIVER_MODULE_PATH    WIFI_MODULE_PATH"8822cs.ko" //FCS850R
+#define RTL8852BS_DRIVER_MODULE_PATH    WIFI_MODULE_PATH"8852bs.ko" //FCS866R
+#define RTL8723DS_DRIVER_MODULE_PATH    WIFI_MODULE_PATH"8723ds.ko" //FCS940R
```

```
+#define RTL8733BS_DRIVER_MODULE_PATH     WIFI_MODULE_PATH"8733bs.ko" //FCS945R
+#define RTL8821CS_DRIVER_MODULE_PATH     WIFI_MODULE_PATH"8821cs.ko" //FCS950R
+
 #define DRIVER_MODULE_PATH_UNKNOW     ""

 #define RTL8188EU_DRIVER_MODULE_NAME "8188eu"
@@ -79,6 +89,16 @@ extern "C" int delete_module(const char *, unsigned int);
 #define MVL_DRIVER_MODULE_NAME            "sd8xxx"
 #define RK912_DRIVER_MODULE_NAME        "rk912"
 #define SPRDWL_DRIVER_MODULE_NAME     "sprdwl"
+
+#define RTL8731BU_DRIVER_MODULE_NAME     "8731bu" //FCU741R
+#define RTL8852BS_DRIVER_MODULE_NAME     "8852bs" //FCS866R
+#define RTL8723DS_DRIVER_MODULE_NAME     "8723ds" //FCS940R
+#define RTL8733BS_DRIVER_MODULE_NAME     "8733bs" //FCS945R
+#define RTL8821CS_DRIVER_MODULE_NAME     "8821cs" //FCS950R
+#define RTL8822CS_DRIVER_MODULE_NAME     "8822cs" //FCS850R
+
 #define DRIVER_MODULE_NAME_UNKNOW     ""

 #ifndef WIFI_DRIVER_FW_PATH_STA
@@ -159,6 +179,14 @@ wifi_ko_file_name module_list[] =
     {"MVL88W8977",          MVL_DRIVER_MODULE_NAME,          MVL_DRIVER_MODULE_PATH,
MVL88W8977_DRIVER_MODULE_ARG},
     {"RK912",               RK912_DRIVER_MODULE_NAME,        RK912_DRIVER_MODULE_PATH,
UNKKOWN_DRIVER_MODULE_ARG},
     {"SPRDWL",     SPRDWL_DRIVER_MODULE_NAME,       SPRDWL_DRIVER_MODULE_PATH,
UNKKOWN_DRIVER_MODULE_ARG},
+    {"RTL8731BU",                                     RTL8731BU_DRIVER_MODULE_NAME,
RTL8731BU_DRIVER_MODULE_PATH, UNKKOWN_DRIVER_MODULE_ARG}, //FCU741R
+    {"RTL8822CS",                                     RTL8822CS_DRIVER_MODULE_NAME,
RTL8822CS_DRIVER_MODULE_PATH, UNKKOWN_DRIVER_MODULE_ARG}, //FCS850R
+    {"RTL8852BS",                                     RTL8852BS_DRIVER_MODULE_NAME,
RTL8852BS_DRIVER_MODULE_PATH, UNKKOWN_DRIVER_MODULE_ARG}, //FCS866R
+    {"RTL8723DS",                                     RTL8723DS_DRIVER_MODULE_NAME,
RTL8723DS_DRIVER_MODULE_PATH, UNKKOWN_DRIVER_MODULE_ARG}, //FCS940R
+    {"RTL8733BS",                                     RTL8733BS_DRIVER_MODULE_NAME,
RTL8733BS_DRIVER_MODULE_PATH, UNKKOWN_DRIVER_MODULE_ARG}, //FCS945R
+    {"RTL8821CS",                                     RTL8821CS_DRIVER_MODULE_NAME,
RTL8821CS_DRIVER_MODULE_PATH, UNKKOWN_DRIVER_MODULE_ARG}, //FCS950R
     {"UNKNOW",       DRIVER_MODULE_NAME_UNKNOW,    DRIVER_MODULE_PATH_UNKNOW,
UNKKOWN_DRIVER_MODULE_ARG}

 };
```

The VID and PID added in the above modification must be consistent with the actual VID and PID of the module. You can obtain and check the VID and PID in the following ways:

1. The VID of the USB interface module is 0bda. You can execute **lsusb** or **cat /sys/bus/usb/devices/*/uvent** to obtain the PID of the module. Taking FCU741R module as an example, after **lsusb** is executed, the returned 0bda:f72b as follows is the VID and PID of FCU741R.

```
root@rockchip:/# lsusb
Bus 005 Device 001: ID 1d6b:0002
Bus 003 Device 001: ID 1d6b:0001
Bus 001 Device 001: ID 1d6b:0002
Bus 005 Device 002: ID 0bda:f72b
Bus 006 Device 001: ID 1d6b:0003
Bus 004 Device 001: ID 1d6b:0001
Bus 002 Device 001: ID 1d6b:000
```

2. The VID of the SDIO interface module is 024c. You can execute **cat /sys/bus/sdio/devices/*/uevent** to obtain the PID of the module. Taking the FCS850R series module as an example, the returned SDIO_ID=024C:C822 is the VID and PID of the FCS850R series module.

```
rk3568_r:/ # cat /sys/bus/sdio/devices/*/uevent
SDIO_CLASS=07
SDIO_ID=024C:C822
MODALIAS=sdio:c07v024CdC822
```

3.2.2. Compilation and Flashing

3.2.2.1. Compiling Code

Execute he following commands under the Android 11 code project to compile the Android source code.

```
source build/envsetup.sh
lunch rk3568_r-userdebug
./build.sh -UCKAu -d sg368z-rk3568
```

update.img is generated in the *rockdev/Image-rk3568_r* directory of Android 11 code project after a successful compilation.

```
rockdev/Image-rk3568_r
 ├─── baseparameter.img
 ├─── boot-debug.img
 ├─── boot.img
 ├─── config.cfg
```

```
├── dtbo.img
├── MiniLoaderAll.bin
├── misc.img
├── parameter.txt
├── pcba_small_misc.img
├── pcba_whole_misc.img
├── recovery.img
├── resource.img
├── super.img
├── uboot.img
├── update.img
└── vbmeta.img
```

3.2.2.2. Flashing Image

For details about image flashing, see ***Chapter 2.2.2.2***.

3.3. Driver Loading

After Android 11 system starts up, the Wi-Fi driver is automatically loaded. Taking FCS945R module as an example, you can check if the driver has been correctly loaded by following the steps below:

Step 1: Check whether the module is enumerated normally as described in ***Step 1*** in ***Chapter 2.3.***

Step 2: Execute **lsmod** to check whether the driver corresponding to the module is loaded. Taking FCS945R module as an example, if 8733bs is returned as follows, the driver of the FCS945R module is loaded successfully.

```
rk3568_r:/ # lsmod
Module                  Size  Used by
8733bs               2469888  0
```

Step 3: Execute **ifconfig wlan0** to check whether wlan0 interface is enabled successfully.

If the information of wlan0 interface is displayed as follows, it indicates that the Wi-Fi driver and Wi-Fi firmware have been loaded successfully, and the Wi-Fi functionality is normal.

```
rk3568_r:/ # ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:e0:4c:da:74:08  Driver rtl8733bs
          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 TX bytes:0
```

3.4. Function Verification

3.4.1. AP Mode

Start the RK3568 Android system to enable AP mode. In this example, the hotspot name is "AndroidAP_9837". If the AP is successfully searched and connected through the mobile phone, it indicates that the AP mode of the Android system is enabled. The interface is displayed as follows:

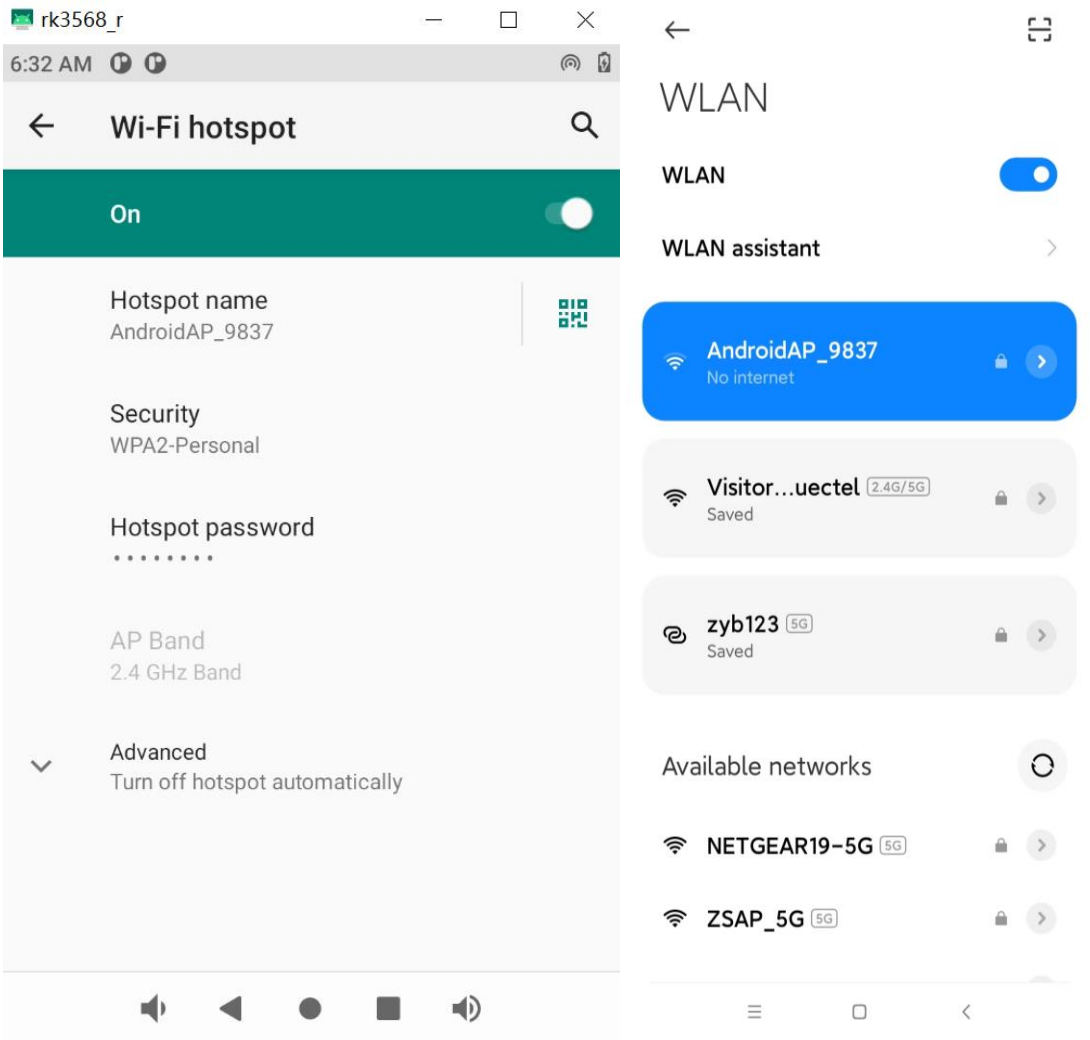

Figure 8: Enable AP Mode

3.4.2. STA Mode

Start the RK3568 Android system to enable STA mode. If the existing AP is successfully searched and connected and then you can surf the internet, it indicates that the STA mode of the Android system is enabled. The interface is displayed as follows:

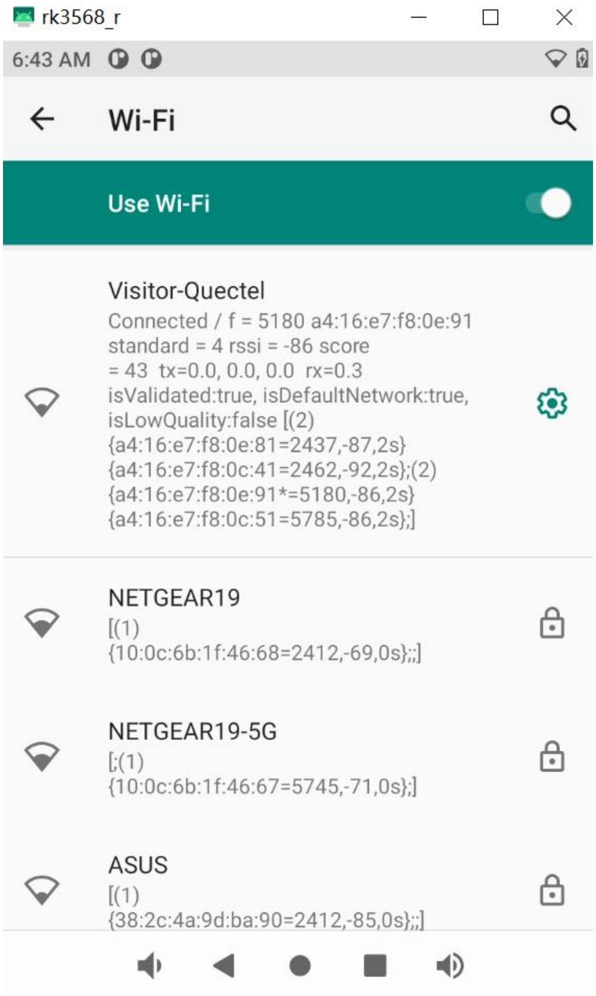

Figure 9: Enable STA Mode

4 Log Capture and Analysis

This chapter, using the RK3568-WF EVB with Quectel Wi-Fi module as an example, describes how to capture logcat and dmesg logs on the Android platform, hostapd, wpa_supplicant and driver logs on the Linux platform.

4.1. Log Capture on Android Platform

Commonly used logs on the Android platform include logcat log and dmesg log. Logcat log records upper-level information, while dmesg log records information from the kernel.

4.1.1. Logcat Log

Step 1: Increase the log printing level.

Because the Android platform, by default, prints only a minimal set of upper-level logs to save power and extend battery life, it is generally necessary to increase the log printing level before capturing logs. The specific steps are as follows:

1. Open the Android system, click “**Settings**” – “**About tablet**” – “**Build number**” Click eight times to enable Developer Mode.
2. Click “**Settings**” – “**System**” – “**Advanced**” – “**Developer options**” and turn on “**Enable Wi-Fi Verbose Logging**”.

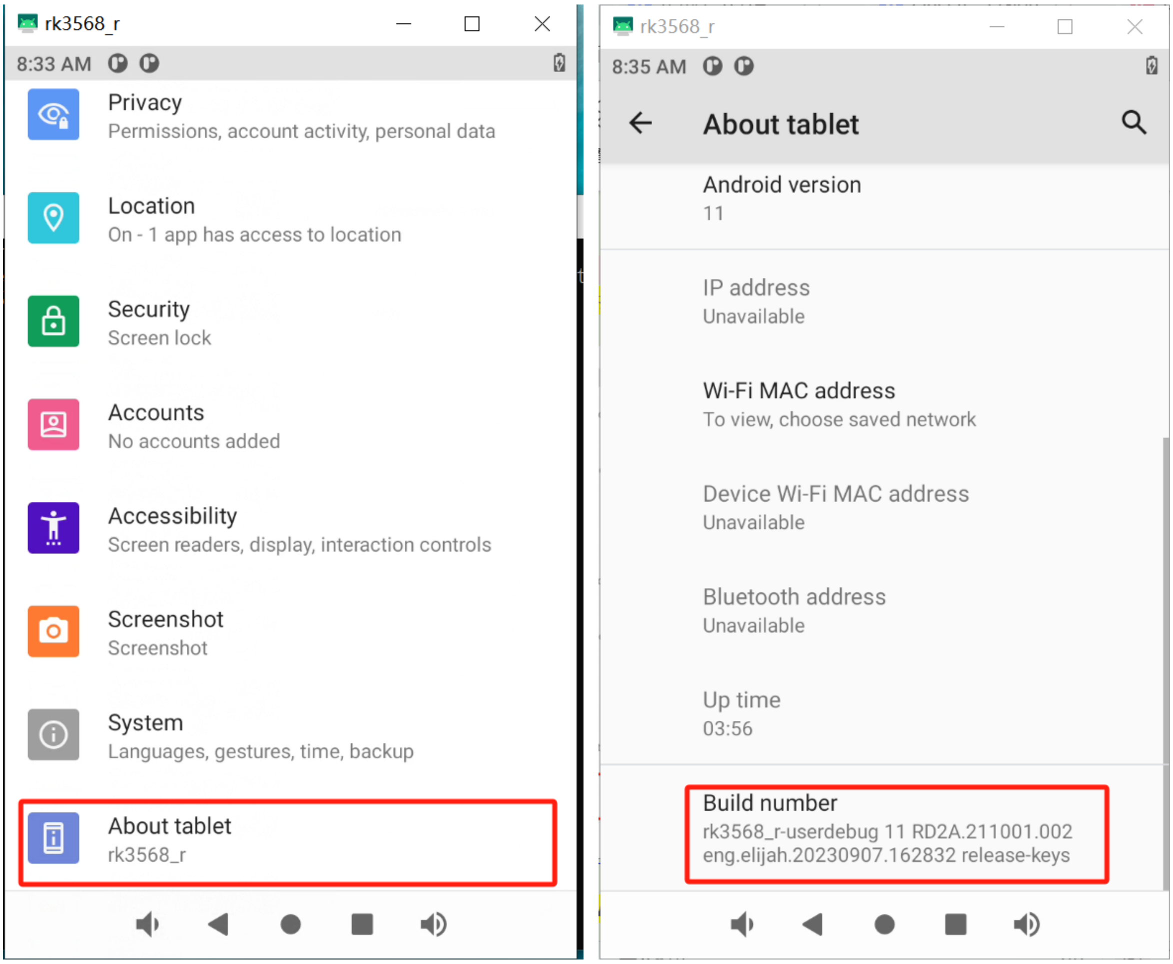

Figure 10: Enable Developer Mode

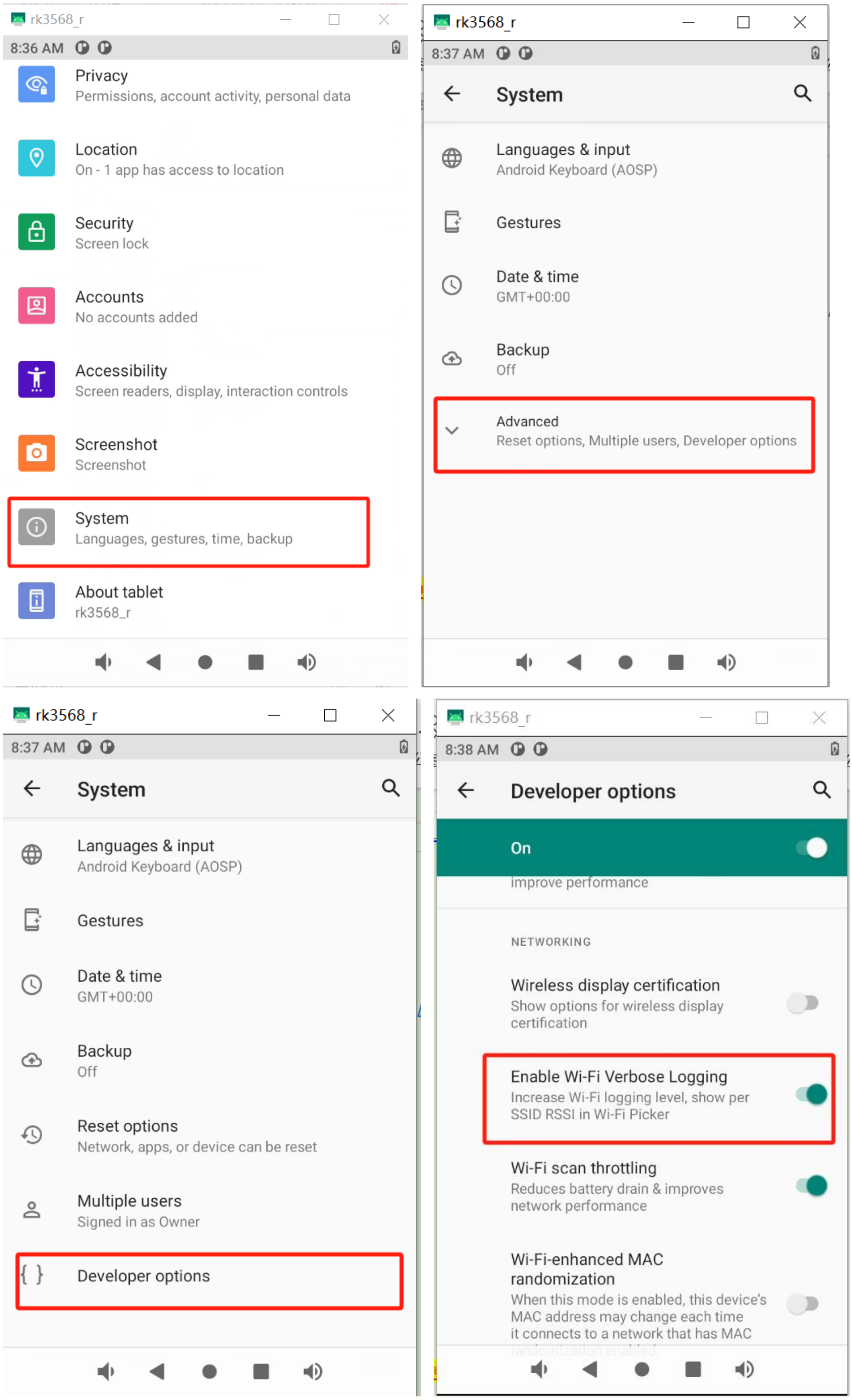

Figure 11: Enable Wi-Fi Verbose Logging

Step 2: Connect the EVB to the PC through a USB to Type-C data cable, then execute **adb** command to capture logcat log. An example of the command is as follows:

```
adb logcat -v time > D:\logcat.txt
```

D:\logcat.txt represents the output log file named *logcat.txt* in the root directory of the D drive, *-v time* indicates real-time capture, and *logcat* specifies the type of logs to be captured.

Step 3: Reproduce the issue following the steps that lead to the problem.

Step 4: After the issue is reproduced, you can press **Ctrl+C** to terminate the log capturing process.

4.1.2. Dmesg Log

Step 1: Increase the log storage space.

To prevent insufficient log storage space where new dmesg logs overwrite old ones, causing incomplete or lost logs during log capturing, it is necessary to increase the log storage space before capturing logs.

1. Open the Android system, click "**Settings**" – "**About tablet**" – "**Build number**" Click eight times to enable Developer Mode.

2. Click "**Settings**" – "**System**" – "**Advanced**" – "**Developer options**" – "**Logger buffer sizes**", and then select "**4M**".

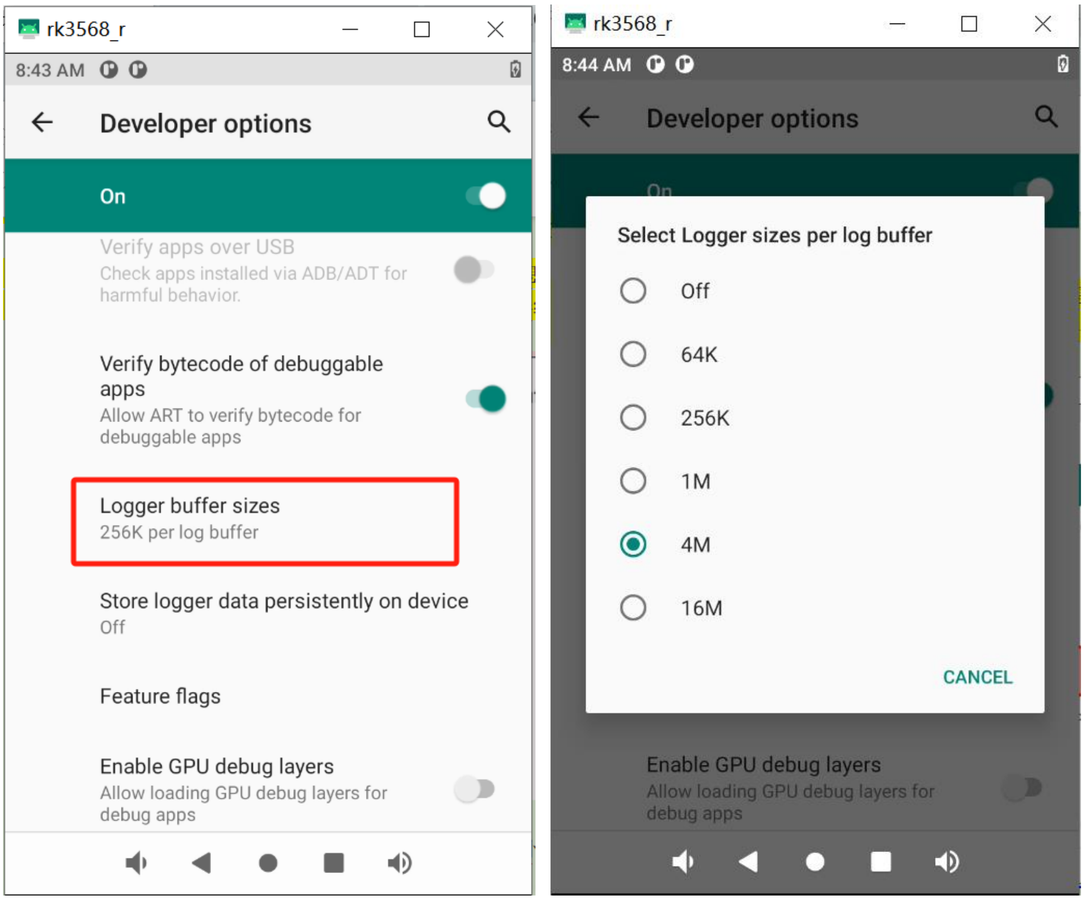

Figure 12: Select Logger Sizes Per Log Buffer

Step 2: Reproduce the issue following the steps that lead to the problem.

Step 3: After reproducing the issue, use a USB to Type-C data cable to connect PC and the EVB, then execute **adb** command to capture dmesg log. An example of the command is as follows:

```
adb shell dmesg > D:\dmesg.txt
```

D:\dmesg.txt represents the output log file named *dmesg.txt* in the root directory of the D drive, and *dmesg* specifies the type of log to be captured.

4.2. Log Capture on Linux Platform

On the Linux platform, if you need to diagnose issues related to AP mode, you will need to capture hostapd log and driver log. If you need to diagnose issues related to STA mode, you will need to capture wpa_supplicant log and driver log. Below are the steps to capture these logs.

Step 1: hostapd and wpa_supplicant output limited logs by default. To generate more detailed logs with timestamps, you need to include the ***-dd -t*** parameters in the startup commands. Here is an example of the commands:

```
hostapd /etc/wifi/hostapd.conf -dd -t &
wpa_supplicant -i wlan0 -Dnl80211 -dd -t -c /etc/wifi/wpa_supplicant.conf &
```

If you start hostapd and wpa_supplicant through executing the above commands in terminal command window, the logs will be printed directly on the terminal window. However, you can also add the ***-f*** parameter to make hostapd and wpa_supplicant logs output to a specified file. Here are examples of the commands:

```
hostapd /etc/wifi/hostapd.conf -dd -t -f /tmp/hostapd.log &
wpa_supplicant -i wlan0 -Dnl80211 -dd -t -c /etc/wifi/wpa_supplicant.conf -f /tmp/wpa_supplicant.log &
```

Step 2: Capture driver log.

Execute the following command to adjust the kernel log printing level.

```
echo 7 > /proc/sys/kernel/printk
```

Execute the following command to print the kernel and driver logs in real time.

```
cat /proc/kmsg &
```

You can also execute the following command to export the logs all at once after the issue reproduces.

```
dmesg
```

Step 3: Reproduce the issue following the steps that lead to the problem.

Step 4: After the issue is reproduced, collect the hostapd or wpa_supplicant and driver logs.

4.3. Keywords in Driver Log

The driver loading logs may be different because the driver loading information for different modules is different.

Taking FCS945R module as an example:

If the following keywords are displayed in the driver loading log, it indicates that the Wi-Fi driver starts being loaded. *v5.14.1.1-29-ga4429ac3f.20230614_COEX20211210-2706* indicates the driver version.

```
[    6.496304] RTW: rtl8733bs v5.14.1.1-29-ga4429ac3f.20230614_COEX20211210-2706
```

The following keywords indicate that the RF power parameter files have been successfully opened, and the path and filename are printed.

```
[    6.421123] RTW: retriveFromFile openFile path:/vendor/etc/firmware/PHY_REG_PG.txt fp=0000000059616321
[    6.421182] RTW: retriveFromFile readFile, ret:1406
[    6.421396] RTW: phy_ParseBBPgParaFile return 1
[    6.448244] RTW: retriveFromFile openFile path:/vendor/etc/firmware/TXPWR_LMT.txt fp=00000000637aa60a
[    6.448300] RTW: retriveFromFile readFile, ret:8055
[    6.450060] RTW: phy_ParsePowerLimitTableFile return 1
```

The following logs indicate that the NIC is successfully registered and the driver is successfully loaded:

```
[    6.455031] RTW: rtw_wiphy_alloc(phy0)
[    6.455087] RTW: rtw_wdev_alloc(padapter=00000000842e1066)
[    6.455101] RTW: rtw_wdev_alloc(padapter=00000000ac397f66)
[    6.455109] RTW: rtw_wiphy_register(phy0)
[    6.455115] RTW: Register RTW cfg80211 vendor cmd(0x67) interface
[    6.466115] RTW: rtw_ndev_init(wlan0) if1 mac_addr=ac:bb:cc:dd:ee:ff
[    6.466348] RTW: rtw_ndev_notifier_call(wlan0) state:16
[    6.471119] RTW: cfg80211_rtw_get_txpower(wlan0) total max: -10000 mbm
[    6.471264] RTW: rtw_ndev_notifier_call(wlan0) state:5
[    6.471353] RTW: rtw_ndev_init(p2p0) if2 mac_addr=ae:bb:cc:dd:ee:ff
[    6.471533] RTW: rtw_ndev_notifier_call(p2p0) state:16
[    6.476355] RTW: cfg80211_rtw_get_txpower(p2p0) total max: -10000 mbm
[    6.476522] RTW: rtw_ndev_notifier_call(p2p0) state:5
[    6.481040] RTW: module init ret=0
```

5 Appendix References

Table 8: Terms and Abbreviations

Abbreviation	Description
ADB	Android Debug Bridge
AP	Access Point
EVB	Evaluation Board
P2P	Peer-to-peer
PC	Personal Computer
PID	Product ID
SDIO	Secure Digital Input/Output
SDK	Software Development Kit
STA	Station
USB	Universal Serial Bus
VID	Vender ID
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access