FGH100M Linux Platform Wi-Fi HaLow User Guide

Short-Range Module Series

Version: 1.1

Date: 2024-01-26

Status: Released

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

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

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
-	2023-04-07	Mark HUANG	Creation of the document
1.0	2023-12-18	Yaobing ZHU	First official release
1.1	2024-01-26	Yaobing ZHU	1. Updated the directory structure of *Code/linux* folder (Chapter 2.2). 2. Updated the operation of pushing Wi-Fi firmware to the EVB and added instructions for using the firmware files (Chapter 3.2.3).

Contents

Table Index

Figure Index

1 Introduction

FGH100M module is a highly integrated Wi-Fi HaLow module provided by Quectel. The module supports the implementation of Wi-Fi HaLow functionality on Linux platform. This document uses the RK3568-WF EVB (which is controlled by RK3568) with Quectel SG368Z-AP module as an example to outline the process for porting Wi-Fi driver, methods for verifying Wi-Fi HaLow functionality, and procedures for capturing logs.

2 Environment Preparation

2.1. Hardware Environment

Table 1: Hardware Components

Item	Amount
RK3568-WF EVB	2
SG368Z-AP module	2
Quectel FGH100M-M.2	2
Antenna	2
USB Type-C cable	2
Power cable	2

NOTE

FGH100M has special band frequency, it cannot perform functional testing with ordinary devices such as smartphones and laptops. Therefore, two sets of devices are used for testing, one acting as an AP and the other as an STA.

The hardware connection is illustrated below.

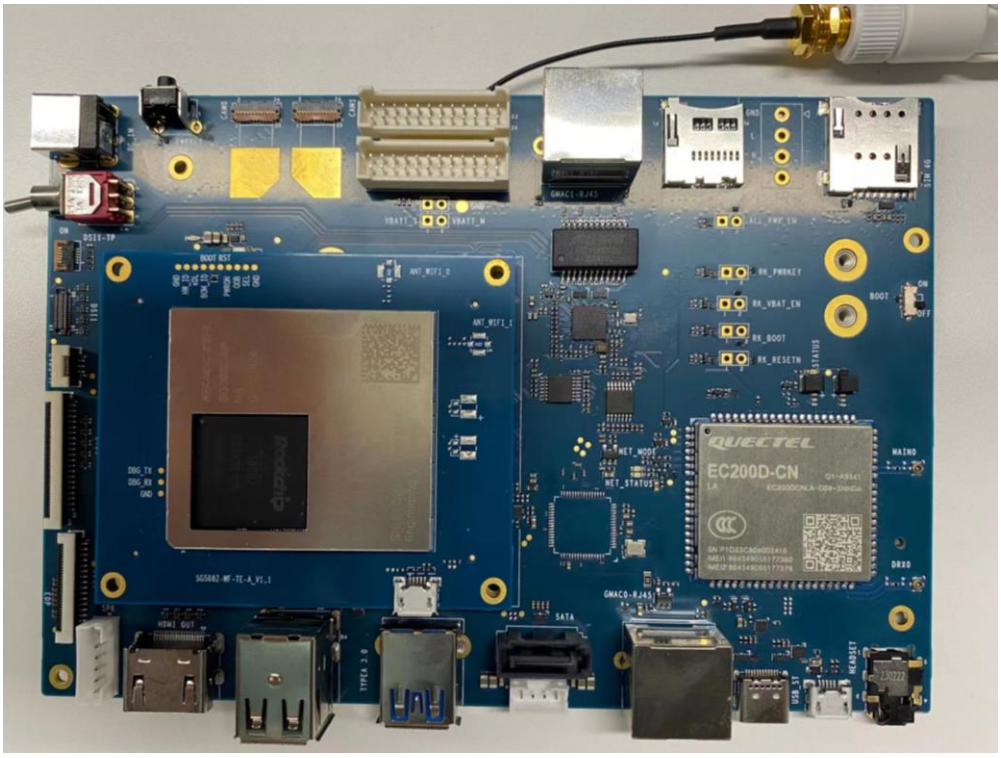


Figure 1: Top View of RK3568-WF EVB

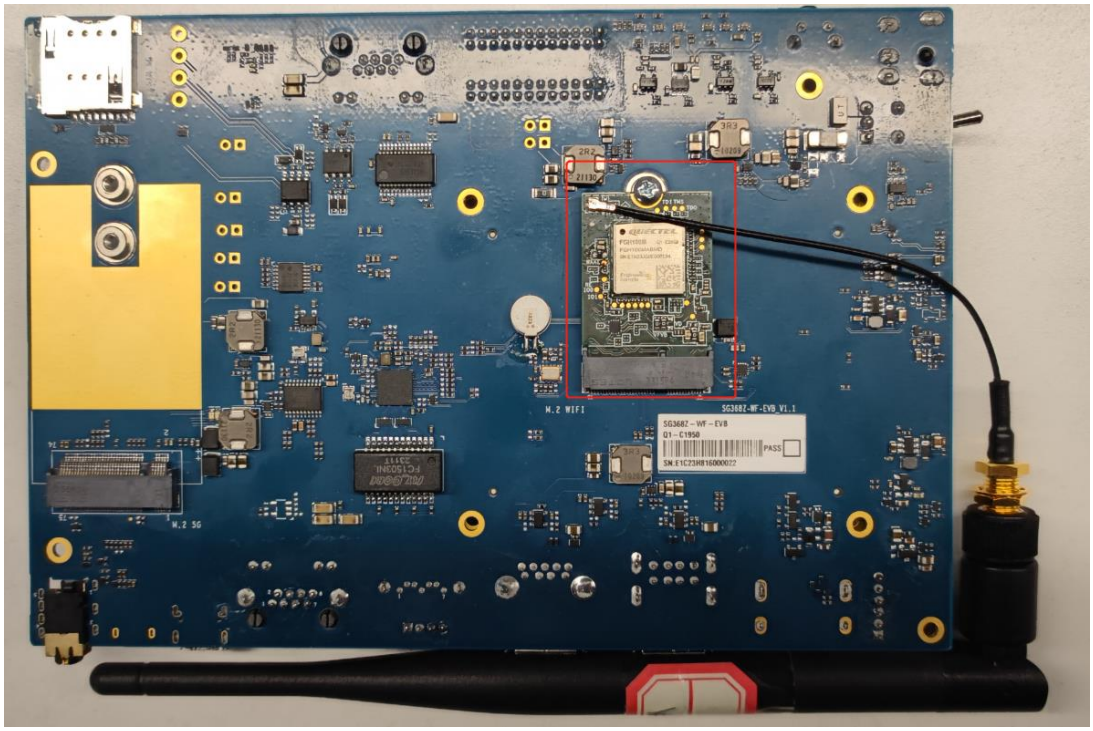


Figure 2: Bottom View of RK3568-WF EVB

2.2. Software Environment

Table 2: Software Components

Type	Description
Code environment	RK3568 Linux 4.19.232 SDK
Driver package	Wi-Fi driver package of Quectel FGH100M module
Compilation environment	Ubuntu 18.04
Tool	1. Install ADB on Windows PC 2. Install upgrade tool *RKDevTool.exe* (https://git-master.quectel.com/wifi.bt/fgh100m/-/tree/master/Tools)

NOTE

Please contact Quectel Technical Support to obtain the Wi-Fi driver package of FGH100M and the upgrade tool *RKDevTool.exe*.

Table 3: Wi-Fi Driver Package

Folder	Description
Docs	Wi-Fi driver porting document and other instruction documents
Code/Linux	Driver source code, firmware, tools, configuration files, and other resources required for Linux system adaptation
RTOS Code	Documents for RTOS system adaptation
Tools	RK3568 image download tool, UART debugging software package

The directory structure of *Code/linux* folder is as follows:

```
@WH-D-006316B:~ /fgh100m/Code/linux$ tree -L 2
├──── firmware
│    ├──── bcf_mf08251.bin
│    ├──── bcf_mf08551.bin
│    ├──── bcf_quectel
│    ├──── mm6108.bin
│    └──── mm6108_dvt.bin
```

```
├──── patchs
│     └──── sys_patch
├──── tools
│     ├──── apps
│     ├──── confs
│     └──── source
└──── wlan_drv
      └──── fgh100m
```

3 Driver Porting

3.1. Code Integration

Step 1: Port the Wi-Fi driver code. Copy the module's Wi-Fi driver code to the *RK3568/kernel/drivers/net/wireless* directory, and rename the folder to *fgh100m*. The directory structure should be as follows.

```
RK3568/kernel/drivers/net/wireless/fgh100m
```

Step 2: Modify the configuration in the *RK3568/kernel/drivers/net/wireless/Makefile* file as indicated below in blue font.

```
--- a/kernel/drivers/net/wireless/Makefile
+++ b/kernel/drivers/net/wireless/Makefile
@@ -30,3 +30,5 @@ obj-$(CONFIG_MAC80211_HWSIM)  += mac80211_hwsim.o

 obj-$(CONFIG_VIRT_WIFI)        += virt_wifi.o
 obj-$(CONFIG_WL_ROCKCHIP)      += rockchip_wlan/
+obj-$(CONFIG_WLAN_VENDOR_MORSE) += fgh100m/
```

Step 3: Modify the configuration in the *RK3568/kernel/drivers/net/wireless/Kconfig* file as indicated below in blue font.

```
--- a/kernel/drivers/net/wireless/Kconfig
+++ b/kernel/drivers/net/wireless/Kconfig
@@ -47,6 +47,8 @@ source "drivers/net/wireless/ti/Kconfig"
 source "drivers/net/wireless/zydas/Kconfig"
 source "drivers/net/wireless/quantenna/Kconfig"
 source "drivers/net/wireless/rockchip_wlan/Kconfig"
+source "drivers/net/wireless/fgh100m/Kconfig"
```

Step 4: Modify the configuration in the *RK3568/kernel/arch/arm64/configs/rockchip_linux_defconfig* in Linux source code as indicated below in blue font.

```
--- a/kernel/arch/arm64/configs/rockchip_linux_defconfig
+++ b/kernel/arch/arm64/configs/rockchip_linux_defconfig
```

```
@@ -219,6 +220,10 @@ CONFIG_RTL_WIRELESS_SOLUTION=y
 # CONFIG_RTL8723DS is not set
 # CONFIG_RTL8821CS is not set
 # CONFIG_RTL8822BS is not set
+CONFIG_WLAN_VENDOR_MORSE=m
+CONFIG_MORSE_SDIO=y
+CONFIG_MORSE_USER_ACCESS=y
+CONFIG_MORSE_VENDOR_COMMAND=y
+CONFIG_MORSE_MONITOR=n
+CONFIG_ANDROID=n
+CONFIG_MAC80211_MESH=y
 CONFIG_USB_NET_RNDIS_WLAN=y
 CONFIG_INPUT_FF_MEMLESS=y
```

Step 5: Modify the configuration in the *kernel/drivers/mmc/core/sdio.c* file as indicated below in blue font. This step is only required for the Rockchip platform.

```
diff --git a/kernel/drivers/mmc/core/sdio.c b/kernel/drivers/mmc/core/sdio.c
index 2dafc562a..a7cc9e19d 100644
--- a/kernel/drivers/mmc/core/sdio.c
+++ b/kernel/drivers/mmc/core/sdio.c
@@ -662,7 +662,7 @@ try_again:
        * try to init uhs card. sdio_read_cccr will take over this task
        * to make sure which speed mode should work.
        */
-      if (!powered_resume && (rocr & ocr & R4_18V_PRESENT)) {
+      if (!powered_resume && (rocr & ocr & R4_18V_PRESENT &  ocr_card )) {
             err = mmc_set_uhs_voltage(host, ocr_card);
             if (err == -EAGAIN) {
                   mmc_sdio_resend_if_cond(host, card);   status = "okay";
 };
```

Step 6: Modify the configuration in the *kernel/drivers/net/wireless/fgh100m/mac.c* file as indicated below in blue font.

```
diff --git a/kernel/drivers/net/wireless/fgh100m/mac.c  b/kernel/drivers/net/wireless/fgh100m/mac.c
index 764c474..350b982 100644
--- a/kernel/drivers/net/wireless/fgh100m/mac.c
+++ b/kernel/drivers/net/wireless/fgh100m/mac.c
@@ -5739,7 +5739,7 @@ static int morse_ieee80211_init(struct morse *mors)
      * Ref: https://lwn.net/Articles/507065/
      *      https://lwn.net/Articles/757643/
      */
-      hw->tx_sk_pacing_shift = 3;
```

```
+    //hw->tx_sk_pacing_shift = 3;
     SET_IEEE80211_PERM_ADDR(hw, mors->macaddr);
     morse_mac_config_ieee80211_hw(mors, hw);
```

3.2. Compiling and Download

3.2.1. Driver Compiling

Execute the following commands in the *RK3568* directory.

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

After successful compilation, the image file *update.img* will be generated in the *RK3568/rockdev* directory, as shown in the figure below.

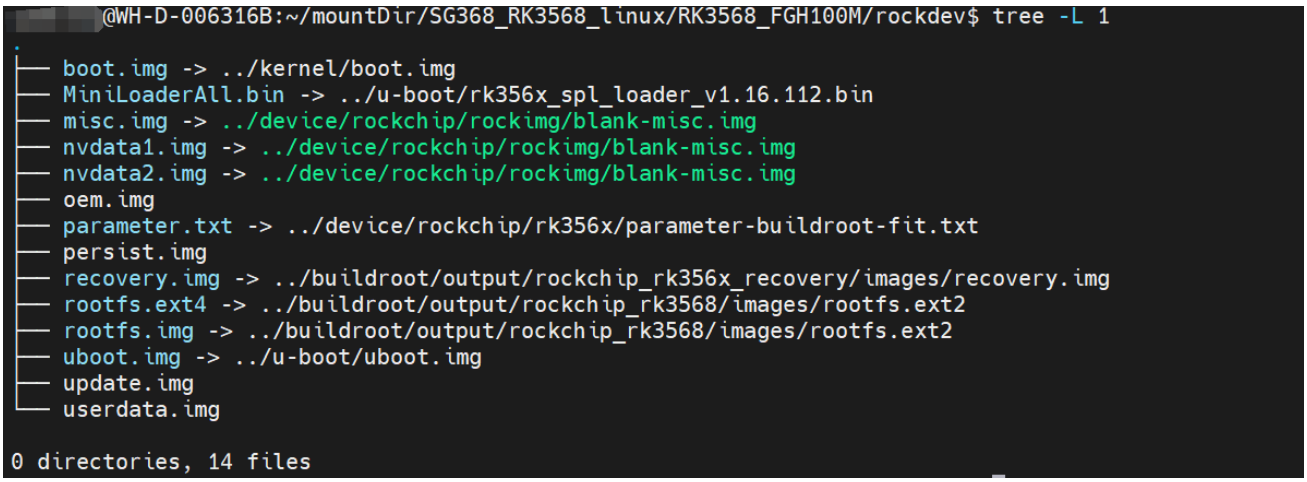


Figure 3: rockdev Directory Structure

At the same time, the *morse.ko* file will be generated in the *RK3568/kernel/drivers/net/wireless/fgh100m* directory, and the *dot11ah.ko* file will be generated in the *RK3568/kernel/drivers/net/wireless/fgh100m/dot11ah* directory, as shown in the figure below.

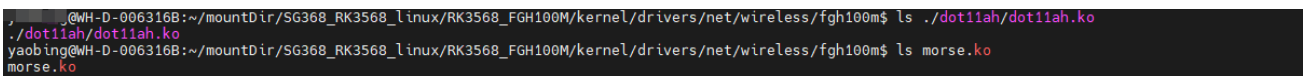


Figure 4: .ko File Directory

3.2.2. Image Download

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

Step 2: Click "**Firmware**" to select and push *update.img*.

Step 3: Set the "BOOT" switch on the RK3568-WF EVB to "ON". After connecting the power supply, press the "**PWRKEY**" button on the RK3568-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 downloading the image.

Figure 5: Download Image

After the image is downloaded successfully, the following interface is displayed:

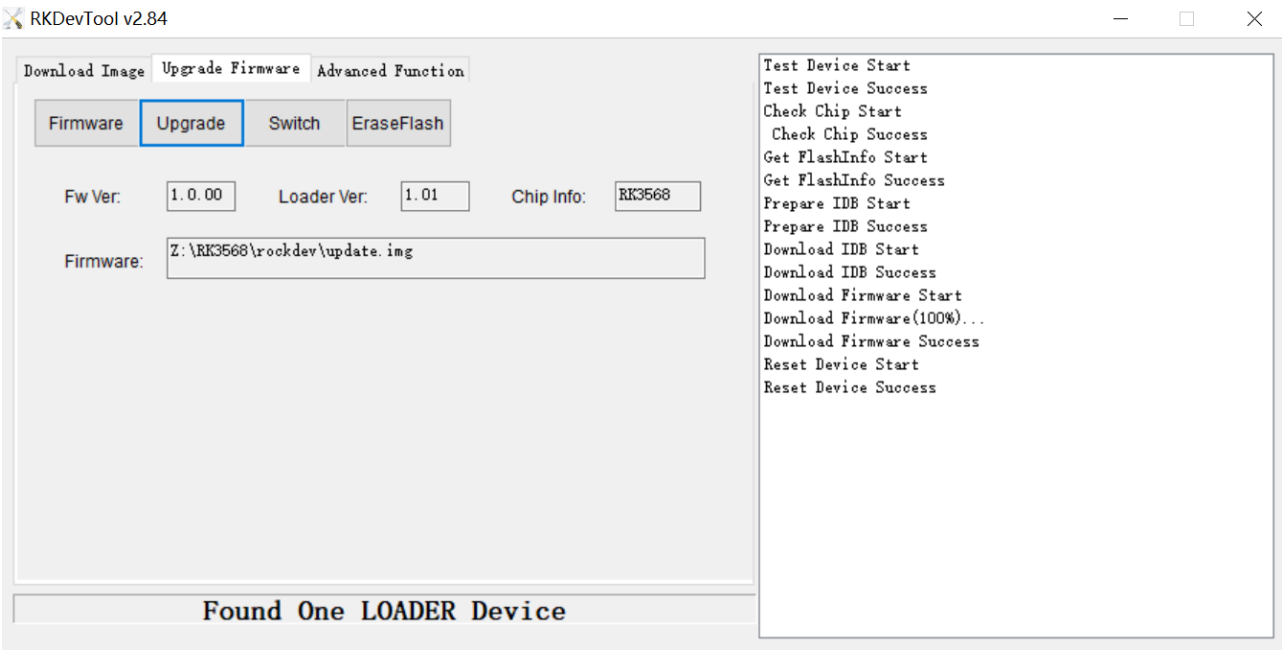


Figure 6: Complete the Download

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

3.2.3. Push Files

Step 1: Push .ko file to EVB.

```
# mount -o remount rw /                  //Modify file system permission, and it is invalid after reboot.
#adb.exe push  dot11ah.ko /mnt          //Push .ko file
#adb.exe push  morse.ko /mnt
#chmod 777 /mnt/*.ko                    //Modify .ko file permission to 777
```

Step 2: Push Wi-Fi firmware to the EVB:

Get three firmware files (*bcf_default.bin*, *mm6108.bin* and *mm6108_dvt.bin*) from the Wi-Fi driver package *fgh100m/Code/linux/firmware*, and use the ADB tool to export the firmware file to the system's */lib/firmware/morse* directory. Details as follows:

```
//Modify file system permissions, which will become invalid after reboot.
#adb shell  mount -o remount rw /
//Save the firmware files (.bin) in the morse folder and export them to the system's /lib/firmware directory
#adb push  morse   /lib/firmware
//Modify the permissions of the firmware files (.bin)
# adb shell chmod 777 /lib/firmware/morse/*
```

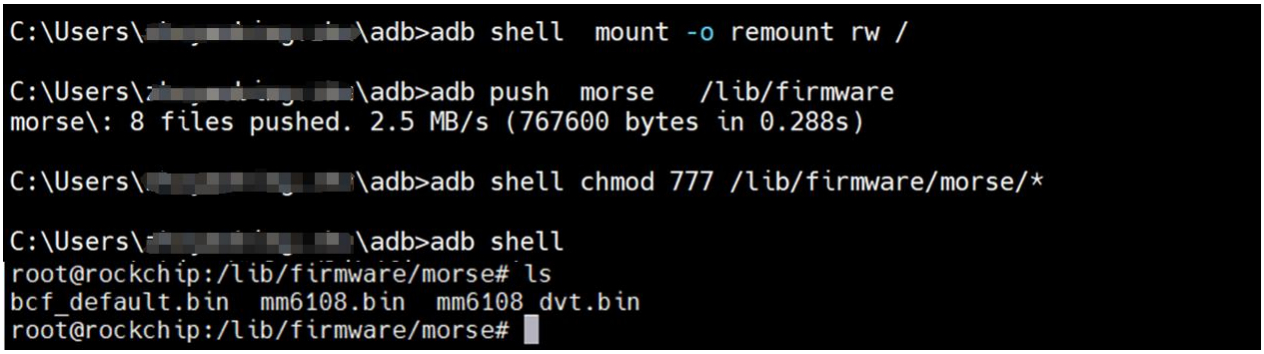


Figure 7: Push Firmware

The directory structure of *Code/linux/firmware* folder is as follows:

```
@WH-D-006316B:~ /fgh100m/Code/linux/firmware$ tree -L 3
.
├── bcf_mf08251.bin
├── bcf_mf08551.bin
├── bcf_quectel
│   ├── FGH100MAAMD
│   │   └── bcf_default.bin
│   └── FGH100MABMD
│       └── bcf_default.bin
├── mm6108.bin
└── mm6108_dvt.bin
```

The instructions for using the *bcf_default.bin*, *mm6108.bin* and *mm6108_dvt.bin* files are as follows.

1. It is recommended to choose the *bcf_default.bin* file calibrated by Quectel and the file is stored in the *Code/linux/firmware/bcf_quectel* directory. And you can choose the corresponding *bcf_default.bin* file based on the module's OC.
2. The *mm6108.bin* file is applicable for all models of FGH100M module.
3. The *mm6108_dvt.bin* file is used only for RF test and it should be renamed to *mm6108.bin* when you perform a RF test.

3.3. Tool Compilation

The wpa_supplicant, hostapd and morsectrl used for module Wi-Fi function verification need to be cross-compiled first and then imported into the system. They need to be stored in the same path as the library files and source code. Examples are as follows:

```
@WH-D-006316B:~/RF-WifiProject/fgh100m/tools$ tree -L 1
.
├── hostapd-rel_1_9_3_2023_Aug_28
├── libnl
├── morsectrl_rel_1_9_3_2023_Aug_28
├── openssl
└── wpa_supplicant-rel_1_9_3_2023_Aug_28
6 directories, 0 files
```

NOTE

The source codes of wpa_supplicant, hostapd, and morsectrl can be obtained in *Linux/tools/source* directory of the Wi-Fi driver package.

3.3.1. Cross-compiling libnl and openssl Libraries

Before cross-compiling the wpa_supplicant, hostapd and morsectrl source codes, you must first compile the libnl and openssl libraries to generate the necessary library files. Examples are as follows:

```
//Set up the compilation environment by specifying the cross-compilation toolchain used in RK3568.
#export PATH=/home/yaobing/RK3568/buildroot/output/rockchip_rk3568/host/bin:$PATH
//Compile libnl library
#git clone https://github.com/thom311/libnl.git
#git checkout libnl3_5_0
#./autogen.sh
#./configure --host=arm CC=aarch64-buildroot-linux-gnu-gcc --prefix=${PWD}/../out --disable-shared
#make clean
#make
#make install
//Compile openssl library
#git clone https://github.com/openssl/openssl.git
#git checkout OpenSSL_1_0_2u
#./Configure linux-generic32 no-shared no-dso -DL_ENDIAN --prefix=${PWD}/../out --openssldir=${PWD}/../out
#make clean
#make CC=aarch64-buildroot-linux-gnu-gcc RANLIB=aarch64-buildroot-linux-gnu-ranlib LD=aarch64-buildroot-linux-gnu--ld MAKEDEPPROG=aarch64-buildroot-linux-gnu-gcc PROCESSOR=ARM
#make install
```

After executing the above command, the .so library file will be generated in the *out* directory of the same level as the library file and source code. Examples are as follows:

```
@WH-D-006316B:~/RF-WifiProject/fgh100m/tools/out$ tree -L 1
```

```
.
├── bin
├── certs
├── etc
├── include
├── lib
├── man
├── misc
├── openssl.cnf
├── private
└── share
```

3.3.2. Cross-compiling wpa_supplicant_s1g

After compiling the wpa_supplicant_s1g source code, the executable files *wpa_supplicant_s1g* and *wpa_cli_s1g* will be generated, located in the *wpa_supplicant-rel_1_9_3_2023_Aug_28/wpa_supplicant* directory, which are used to connect to Wi-Fi hotspots in STA mode. The compilation method is as follows:

```
//Set up the compilation environment by specifying the cross-compilation toolchain consistent with RK3568
#export PATH=/home/yaobing/RK3568/buildroot/output/rockchip_rk3568/host/bin:$PATH

//Compile wpa_supplicant_s1g
#cd  wpa_supplicant-rel_1_7_2_2023_Mar_15/wpa_supplicant
//Rename the defconfig file to .config and add the following content highlighted in blue
#mv defconfig .config
#vim .config
CC=aarch64-buildroot-linux-gnu-gcc
STRIP=aarch64-buildroot-linux-gnu-strip
#Specify the path of libnl and openssl libraries
PWD-libnl=/home/yaobing/RF-WifiProject/fgh100m/tools/out
PWD-openssl=/home/yaobing/RF-WifiProject/fgh100m/tools/out
CFLAGS += -Os
CFLAGS += -I $(PWD-libnl)include/libnl3/
CFLAGS += -I $(PWD-libnl)include/linux-private/
LIBS += -L $(PWD-libnl)/lib
LIBS_c += -L $(PWD-libnl)/lib
LIBS_p += -L $(PWD-libnl)/lib
CFLAGS += -I $(PWD-openssl)/include/
LIBS += -L $(PWD-openssl)/lib
LIBS_c += -L $(PWD-openssl)/lib
LIBS_p += -L $(PWD-openssl)/lib
LDFLAGS += -lpthread -lm
```

```
CONFIG_LIBNL32=y
CONFIG_EAP_FAST=y
CONFIG_DPP2=y
CONFIG_INTERWORKING=y
//Compile source code
#make clean
#make
```

3.3.3. Cross-compiling hostapd_s1g

After compiling the hostapd_s1g source code, the executable files *hostapd_s1g* and *hostapd_cli_s1g* will be generated, located in the *hostapd-rel_1_9_3_2023_Aug_28/hostapd* directory, which are used to turn on Wi-Fi hotspots in AP mode. The compilation method is as follows:

```
//Set up the compilation environment by specifying the cross-compilation toolchain consistent with RK3568
#export PATH=/home/yaobing/RK3568/buildroot/output/rockchip_rk3568/host/bin:$PATH
//Compile hostap_s1g
#cd  ./hostapd-rel_1_7_2_2023_Mar_15/hostapd
//Rename the defconfig file to .config and add the following content in blue font
#mv defconfig .config
#vim .config
CC=aarch64-buildroot-linux-gnu-gcc
STRIP=aarch64-buildroot-linux-gnu-strip
PWD-libnl=/home/yaobing/RF-WifiProject/fgh100m/tools/out
PWD-openssl=/home/yaobing/RF-WifiProject/fgh100m/tools/out
CFLAGS += -Os
CFLAGS += -I $(PWD-libnl)include/libnl3/
CFLAGS += -I $(PWD-libnl)include/linux-private/
LIBS += -L $(PWD-libnl)/lib
LIBS_c += -L $(PWD-libnl)/lib
LIBS_p += -L $(PWD-libnl)/lib
CFLAGS += -I $(PWD-openssl)/include/
LIBS += -L $(PWD-openssl)/lib
LIBS_c += -L $(PWD-openssl)/lib
LIBS_p += -L $(PWD-openssl)/lib
LDFLAGS += -lpthread -lm
CONFIG_LIBNL32=y
CONFIG_EAP_FAST=y
CONFIG_DPP2=y
CONFIG_INTERWORKING=y
//Compile source code
#make clean
#make
```

3.3.4. Cross-compiling morsectl

After compiling the morsectrl source code, the executable files *morsectrl* and *morse_cli* will be generated, located in the *morsectrl_rel_1_9_3_2023_Aug_28* directory, which are used for RF testing, and the specific usage method can be queried through the **morsectrl --help** command. The compilation method is as follows:

```
//Set up the compilation environment by specifying the cross-compilation toolchain consistent with RK3568
#export PATH=/home/yaobing/RK3568/buildroot/output/rockchip_rk3568/host/bin:$PATH
//Compile morsectrl
#cd  ./morsectrl_rel_1_9_3_2023_Aug_28
#make clean
#make CC=aarch64-buildroot-linux-gnu-gcc CONFIG_MORSE_TRANS_NL80211=1 CFLAGS=-I/home/yaobing/RF-WifiProject/fgh100m/tools/out/include/libnl3 LDFLAGS=-L/home/yaobing/RF-WifiProject/fgh100m/tools/out/lib
```

NOTE

1. The content following **make CC** is the name of the compiler.
2. The content after **CFLAGS=-I** and **LDFLAGS=-L** refers to the libnl3 and lib library paths in the *out* folder.

4 Function Verification

This chapter demonstrates how to verify the Wi-Fi HaLow functionality of the module.

4.1. Driver Loading

Execute **dmesg |grep "mmc*"**. If the blue fields similar to the following are displayed, it indicates a successful SDIO enumeration.

```
[    2.690869] mmc_host mmc2: card is non-removable.
[    2.909495] mmc_host mmc2: Bus speed (slot 0) = 375000Hz (slot req 400000Hz, actual 375000HZ div = 0)
[    2.925063] sdhci-dwcmshc fe310000.sdhci: Looking up vmmc-supply from device tree
[    2.925093] sdhci-dwcmshc fe310000.sdhci: Looking up vmmc-supply property in node /sdhci@fe310000 failed
[    2.925185] sdhci-dwcmshc fe310000.sdhci: Looking up vqmmc-supply from device tree
[    2.925209] sdhci-dwcmshc fe310000.sdhci: Looking up vqmmc-supply property in node /sdhci@fe310000 failed
[    2.925286] mmc0: Unknown controller version (5). You may experience problems.
[    2.958404] mmc0: SDHCI controller on fe310000.sdhci [fe310000.sdhci] using ADMA
[    2.989892] mmc0: Host Software Queue enabled
[    2.989925] mmc0: new HS200 MMC card at address 0001
[    2.990980] mmcblk0: mmc0:0001 Y2P032 29.1 GiB
[    2.991603] mmcblk0boot0: mmc0:0001 Y2P032 partition 1 4.00 MiB
[    2.992226] mmcblk0boot1: mmc0:0001 Y2P032 partition 2 4.00 MiB
[    2.992527] mmcblk0rpmb: mmc0:0001 Y2P032 partition 3 16.0 MiB, chardev (238:0)
[    2.995111]  mmcblk0: p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11
[    3.033965] panel-simple-dsi fe070000.dsi.0: fe070000.dsi.0 supply vsp not found, using dummy regulator
[    3.034066] panel-simple-dsi fe070000.dsi.0: fe070000.dsi.0 supply vsn not found, using dummy regulator
[    3.053491] fts_ts 1-0038: 1-0038 supply vdd not found, using dummy regulator
[    3.053623] fts_ts 1-0038: 1-0038 supply vcc_i2c not found, using dummy regulator
[    3.704422] mmc_host mmc2: Bus speed (slot 0) = 50000000Hz (slot req 50000000Hz, actual 50000000HZ div =
0)
[    3.710451] mmc2: new high speed SDIO card at address 0001
```

```
[    4.332748] iommu: Adding device fde40000.npu to group 0
```

Execute the following commands to load the driver.

```
#cd /mnt
# insmod dot11ah.ko
# insmod morse.ko
```

Execute **dmesg** to view driver loading logs. If the driver is loaded successfully, the following information will be printed.

```
root@rockchip:/# dmesg
6,1205,1122221961,-;Morse Micro Dot11ah driver registration. Version 0-rel_1_7_2_quectel_NFP_2023_Mar_15-gc7835913
6,1206,1126838239,-;morse micro driver registration. Version 0-rel_1_7_2_quectel_NFP_2023_Mar_15-gc7835913
6,1207,1126838671,-;morse_sdio mmc2:0001:1: sdio new func 1 vendor 0x325b device 0x306 block 0x8/0x8
 SUBSYSTEM=sdio
 DEVICE=+sdio:mmc2:0001:1
6,1208,1126839455,-;morse_sdio mmc2:0001:2: sdio new func 2 vendor 0x325b device 0x306 block 0x200/0x200
 SUBSYSTEM=sdio
 DEVICE=+sdio:mmc2:0001:2
6,1209,1126839483,-;morse_sdio mmc2:0001:2: Reading gpio pins configuration from device tree
 SUBSYSTEM=sdio
 DEVICE=+sdio:mmc2:0001:2
7,1210,1127367387,-;ieee80211 phy0: Selected rate control algorithm 'minstrel_ht'
 SUBSYSTEM=ieee80211
 DEVICE=+ieee80211:phy0
6,1211,1127368696,-;uaccess char driver major number is 510
6,1212,1127369208,-;morse_io: Device node '/dev/morse_io' created successfully
6,1213,1127715057,-;IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready6,1206,273601905,-;IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
```

Execute **ifconfig -a** to view ports. If the following information of wlan0 is displayed, it indicates that the Wi-Fi driver is loaded successfully.

```
root@RK356X:/ifconfig -a
wlan0     Link encap:Ethernet  HWaddr 02:00:00:03:02:02
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

```
RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

4.2. Test and Verification

The following steps should be taken before test and verification:

Step 1: Execute **mount -o remount rw /** to modify file system permission.

Step 2: Use ADB tool to push the compiled executable files such as *hostapd_s1g*, *hostapd_cli_s1g*, *wpa_supplicant_s1g* and *wpa_cli_s1g* into the */usr/sbin* directory, and push *dnsmasq.conf*, *hostapd_s1g.conf*, and *wpa_supplicant_s1g.conf* files into the */etc/wifi* directory.

```
root@rockchip:/# ls /usr/sbin/*_s1g
/usr/sbin/hostapd_cli_s1g  /usr/sbin/wpa_cli_s1g
/usr/sbin/hostapd_s1g      /usr/sbin/wpa_supplicant_s1g
root@rockchip:/# ls /etc/wifi/
dnsmasq.conf  hostapd_s1g.conf     wpa_supplicant_s1g.conf
hostapd.conf  wpa_supplicant.conf
```

Figure 8: Path of Related Files

NOTE

1. *dnsmasq.conf*, *hostapd_s1g.conf* and *wpa_supplicant_s1g.conf* can be obtained from *Code/linux/tools/confs* of the Wi-Fi driver package.
2. *dnsmasq.conf* is the process configuration file used to automatically allocate IPs in AP mode. It can be replaced by the udhcpd configuration file.

4.2.1. Connection Status Verification

1. Connect STA to AP

Execute the following commands on the AP side to enable the hotspot.

```
#hostapd_s1g -dd /etc/wifi/hostapd_s1g.conf &
#ifconfig wlan0 192.168.11.1
#killall  dnsmasq
#dnsmasq -i wlan0 -C  /etc/wifi/dnsmasq.conf &
```

Execute the following commands on the STA side to connect to the hotspot and obtain the IP address.

```
#wpa_supplicant_s1g -Dnl80211 -iwlan0 -c /etc/wifi/wpa_supplicant_s1g.conf -dd &
#udhcpc -i wlan0 &
```

2. Check connection status

Execute **hostapd_cli_s1g -i wlan0 status** on the AP side to check connection status.

```
root@rockchip:/# hostapd_cli_s1g -i wlan0 status
s1g_freq=922000
s1g_bw=4
s1g_prim_chwidth=2
s1g_prim_1mhz_chan_index=0
state=ENABLED
phy=phy0
freq=5745
num_sta_non_erp=0
num_sta_no_short_slot_time=1
num_sta_no_short_preamble=1
olbc=0
num_sta_ht_no_gf=1
num_sta_no_ht=0
num_sta_ht_20_mhz=0
num_sta_ht40_intolerant=0
olbc_ht=0
ht_op_mode=0x4
hw_mode=a
country_code=ZZ
country3=0x20
cac_time_seconds=0
cac_time_left_seconds=N/A
channel=149
edmg_enable=0
edmg_channel=0
secondary_channel=1
ieee80211n=1
ieee80211ac=1
ieee80211ax=0
ieee80211be=0
beacon_int=100
dtim_period=1
vht_oper_chwidth=1
vht_oper_centr_freq_seg0_idx=155
vht_oper_centr_freq_seg1_idx=0
vht_caps_info=00000020
rx_vht_mcs_map=fffd
tx_vht_mcs_map=fffd
ht_caps_info=006e
ht_mcs_bitmask=ff030000000000000000
supported_rates=01 02 03 04 05 06 07 08 09 0b 0c 0d 0e 11 12 16 18 19 1b 1e 24 30
max_txpower=0
bss[0]=wlan0
bssid[0]=0c:bf:74:2c:9e:28
ssid[0]=Quectel-halow-100h
num_sta[0]=1
root@rockchip:/#
```

Figure 9: AP Connection Status

NOTE

The parameters such as *freq* and *channel* in the *hostapd_cli_s1g* may not match the actual settings and the actual settings are based on the configuration in *hostapd_s1g.conf*.

Execute **wpa_cli_s1g -i wlan0 status** on the STA side to check connection status.

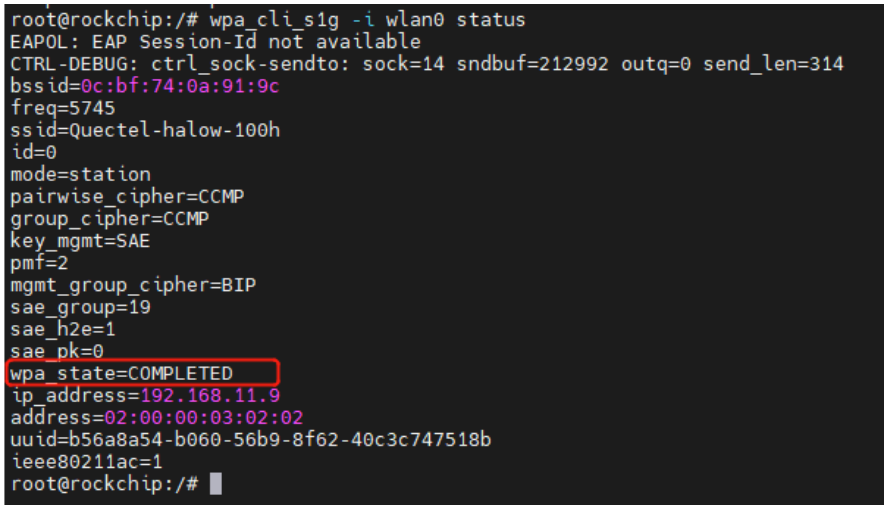


Figure 10: STA Connection Status

Execute **ifconfig wlan0** to check the IP address obtained by the STA, which is 192.168.11.9 in this case. Verify whether the module works properly by pinging the IP address of the STA on the AP side. The result is as follows.

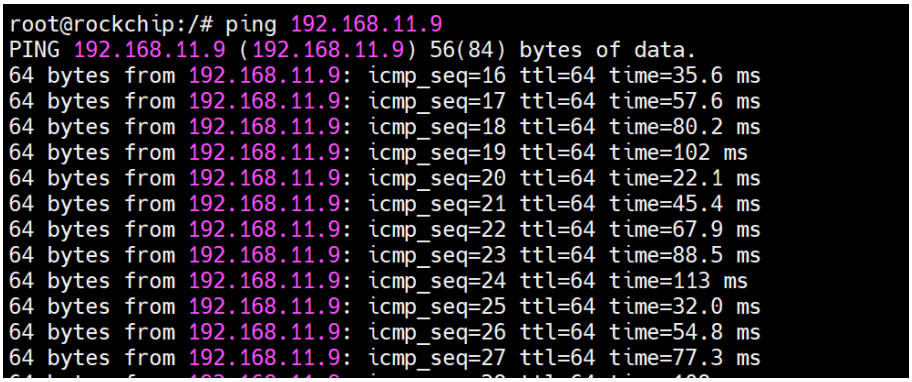


Figure 11: Connection Verification

3. Configure the hotspots

To configure different hotspots on the AP side, you can modify parameters such as *channel*, *op_class*, *country_code*, *s1g_prim_chwidth*, *ssid* and *sae_password* in the *hostapd_s1g.conf* file. For more details, please refer to *CustomerApps-How to set up the channel by modifying hostapd_s1g.conf-260623-020740* and *MM610x-CHANNELS-UG102*. To obtain these documents, you can contact Quectel Technical Support.

The STA side configures parameters such as *ssid*, *key_mgmt* and *psk* based on the encryption method, Wi-Fi name, and password of the hotspot on the AP side. Taking the modification of the country code to US as an example, follow these steps to modify the configuration:

Step 1: Execute the following commands to load the driver and specify the country code as US.

```
#insmod  /mnt/dot11ah.ko
#insmod /mnt/morse.ko country=US
```

Step 2: Execute the following command to view the country code.

```
cat /sys/module/morse/parameters/country
```

Step 3: Configure the country code in the *hostapd_s1g.conf* file as US.

```
# E.g. EU channel 36 is 863 MHz, JP channel 36 is 917MHz.
country_code=US
```

4.2.2. Data Throughput Test

- **Testing Conditions:**

1. Open environment (Not in a shield room)
2. Hotspot configuration (*channel* = 40, *country_code* = US, *s1g_prim_chwidth* = 1)

- **Data Throughput over TCP:**

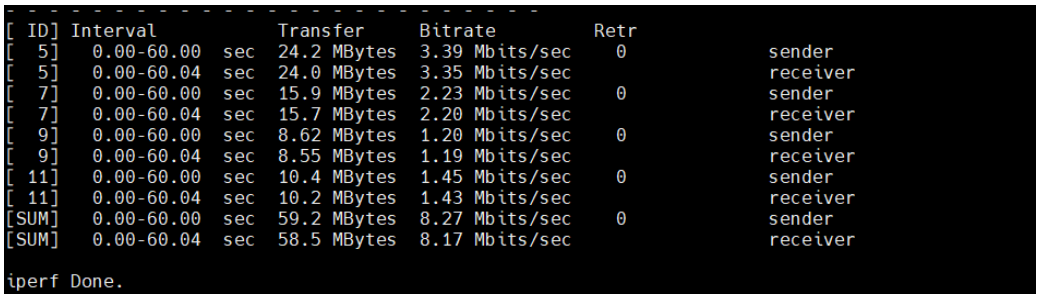
Execute the following command on the server.

```
iperf3 -s
```

Execute the following command on the client.

```
TX: iperf3 -c 192.168.11.9 -P 4 -t 30
RX: iperf3 -c 192.168.11.9 -P 4 -t 30 -R
```

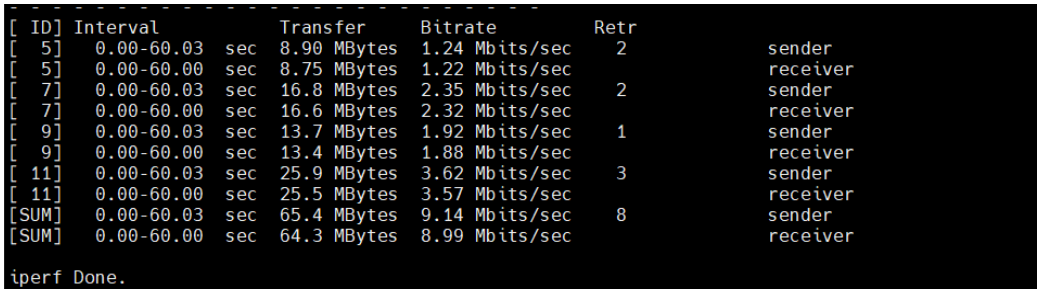
The data throughput over TCP is as follows.



```
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-60.00  sec  24.2 MBytes  3.39 Mbits/sec    0             sender
[  5]   0.00-60.04  sec  24.0 MBytes  3.35 Mbits/sec                  receiver
[  7]   0.00-60.00  sec  15.9 MBytes  2.23 Mbits/sec    0             sender
[  7]   0.00-60.04  sec  15.7 MBytes  2.20 Mbits/sec                  receiver
[  9]   0.00-60.00  sec  8.62 MBytes  1.20 Mbits/sec    0             sender
[  9]   0.00-60.04  sec  8.55 MBytes  1.19 Mbits/sec                  receiver
[ 11]   0.00-60.00  sec  10.4 MBytes  1.45 Mbits/sec    0             sender
[ 11]   0.00-60.04  sec  10.2 MBytes  1.43 Mbits/sec                  receiver
[SUM]   0.00-60.00  sec  59.2 MBytes  8.27 Mbits/sec    0             sender
[SUM]   0.00-60.04  sec  58.5 MBytes  8.17 Mbits/sec                  receiver

iperf Done.
```

Figure 12: TCP-TX Data Throughput



```
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-60.03  sec  8.90 MBytes  1.24 Mbits/sec    2             sender
[  5]   0.00-60.00  sec  8.75 MBytes  1.22 Mbits/sec                  receiver
[  7]   0.00-60.03  sec  16.8 MBytes  2.35 Mbits/sec    2             sender
[  7]   0.00-60.00  sec  16.6 MBytes  2.32 Mbits/sec                  receiver
[  9]   0.00-60.03  sec  13.7 MBytes  1.92 Mbits/sec    1             sender
[  9]   0.00-60.00  sec  13.4 MBytes  1.88 Mbits/sec                  receiver
[ 11]   0.00-60.03  sec  25.9 MBytes  3.62 Mbits/sec    3             sender
[ 11]   0.00-60.00  sec  25.5 MBytes  3.57 Mbits/sec                  receiver
[SUM]   0.00-60.03  sec  65.4 MBytes  9.14 Mbits/sec    8             sender
[SUM]   0.00-60.00  sec  64.3 MBytes  8.99 Mbits/sec                  receiver

iperf Done.
```

Figure 13: TCP-RX Data Throughput

5 Log Capture

On the Linux platform you can capture various logs, including kmsg, dmesg, and wpa_supplicant logs, all of which provide kernel-related information.

5.1. Kernel Log

Step 1: Power on the RK3568-WF EVB;

Step 2: Connect the EVB to the PC via a USB port, open ADB tool on the PC, and execute **adb shell** first, and then execute **dmesg > dmesg.txt** or **cat dev/kmsg > kmsg.txt &** to export kernel log as *dmesg.txt* or *kmsg.txt*, and save it to a directory of your choice.

5.2. wpa_supplicant Log

Step 1: Power on the RK3568-WF EVB;

Step 2: Connect the EVB to the PC via a USB port, open ADB tool on the PC and then execute **wpa_supplicant_s1g -iwlan0 -c/etc/wpa_supplicant.conf -Dnl80211 -B -t -ddd -f /etc/wpa_supplicant.log** to export wpa_supplicant log as *wpa_supplicant.log* and save it to */etc* directory.

Step 3: Execute **adb pull /etc/wpa_supplicant.log ./** on the ADB tool on the PC to export the *wpa_supplicant.log* and save it to to a directory of your choice.

6 Appendix Terms and Abbreviations

Table 4: Terms and Abbreviations

Abbreviation	Description
ADB	Android Debug Bridge
AP	Access Point
EVB	Evaluation Board
IP	Internet Protocol
OC	Ordering Code
PC	Personal Computer
RX	Receive
SDK	Software Development Kit
STA	Station
TCP	Transmission Control Protocol
TX	Transmit
UDP	User Datagram Protocol
USB	Universal Serial Bus
WLAN	Wireless Local Area Network