FC2x&FC900E Series Third-Party Android Wi-Fi User Guide

带格式的: 字体: 28 磅

带格式的: 字体: 28 磅

Wi-Fi&Bluetooth Module Series

Version: 1.0

Date: 2023-01-12

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.

About the Document

Revision History

Version	Date	Author	Description
-	2022-10-21	Sai YANG	Creation of the document
1.0	2023-01-12	Sai YANG	First official release

Contents

Table Index

Figure Index

1 Introduction

Quectel Wi-Fi & Bluetooth modules FC20 series, FC21 and FC900E support Wi-Fi feature on the third-party Android platform. This document explains how to use the modules in Android 11 of the third-party platform RK356X (taking RK3566 as an example), including how to build Wi-Fi driver, how to build and download RK356X Android 11, the addition of Wi-Fi firmware and verification of Wi-Fi function.

2 RK3566 Android 11 Platform

2.1. Environment Preparation

2.1.1. Hardware Environment

1. RK356X EVB × 1 (RK3566 EVB is used in the examples below);
2. Quectel Wi-Fi & Bluetooth module × 1 (FC20 series, FC21 or FC900E module);
3. Antenna × 1;
4. USB Type-C cable × 1;
5. Power cable × 1.

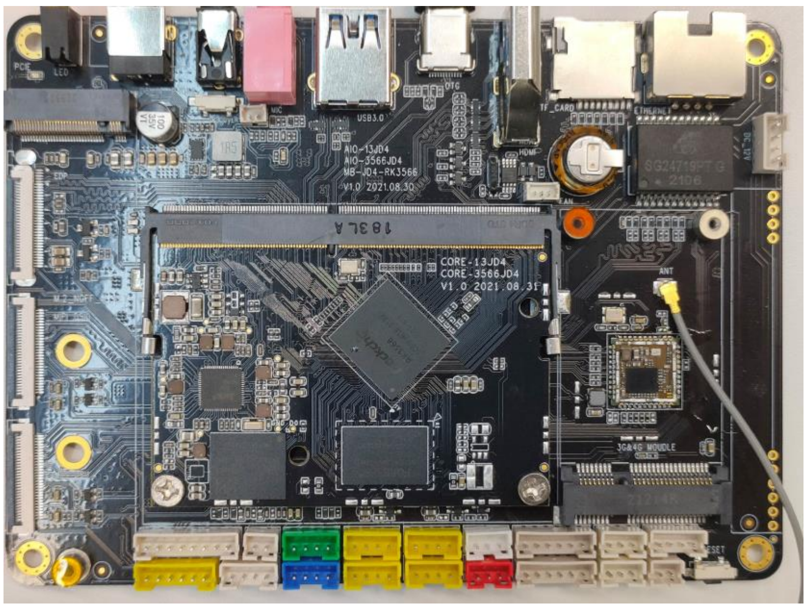


Figure 1: RK3566 EVB

2.1.2. Software Environment

1. Configure code environment of RK356X Android 11, that is, download the RK356X EVB SDK code;
2. Download Quectel FC20 series, FC21 or FC900E module Wi-Fi driver code;
3. Download Quectel FC20 series, FC21 or FC900E module Wi-Fi firmware;
4. Ubuntu compilation environment;
5. RKDevTool installed on Windows (RKDevTool v2.84 is used in the examples below).

NOTE

1. For downloading and usage of RK356X EVB SDK code, refer to the official website: https://wiki.t-firefly.com/en/Core-3566JD4/.
2. Download RKDevTool from: https://wiki.t-firefly.com/en/Core-3566JD4/compile_android11.0_firmware.html.

2.2. Integrate Wi-Fi Driver into Android 11

2.2.1. Obtain Module Wi-Fi Driver Source Code

Execute the following command to obtain FC2x driver source code:

```
git clone https://git-master.quectel.com/wifi.bt/fc2x.git
```

The driver source code is controlled through the relevant branch shown below:

- *remotes/origin/FC20*: FC20 series driver code
- *remotes/origin/FC21SA* or *remotes/origin/FC21SD*: FC21 driver code

Execute the following command to obtain FC900E driver source code:

```
git clone    https://git-master.quectel.com/wifi.bt/fc900e.git
```

Table 1: Wi-Fi Files in Driver Source Code

File	Description
cnss_host_LEA/chss_proc/host/AIO/drivers/qcacld-new	Wi-Fi driver code
meta_build/load_meta/wlan_firmware/sdio	Wi-Fi firmware

2.2.2. View Module VID and PID

You need to know the module's VID and PID to modify the driver source code. Execute the following command to get the module's VID and PID.

```
cat /sys/bus/sdio/devices/mmc1\:0001\:1/uevent
```

The result is displayed as follows:

```
SDIO_CLASS=00
SDIO_ID=0271:0701
MODALIAS=sdio:c00v0271d0701
```

In the result, 0271 in *SDIO_ID=0271* is VID, and 0701 is PID.

2.2.3. Modify Android 11 Source Code

Because the Android 11 kernel version is not compatible with the Wi-Fi-driver version, it is necessary to modify the configuration of Android kernel as shown below. The Android 11 source code file that needs to be modified is *<AndroidRoot>/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp* and the structure that needs to be modified is *supported_wifi_devices* in the *rk_wifi_ctrl.cpp* file. The specific modification are shown in blue font as follows:

```
   diff --git a/build/soong/ui/build/finder.go b/build/soong/ui/build/finder.go
index 0f34b5e702..6637d814cd 100644
--- a/build/soong/ui/build/finder.go
+++ b/build/soong/ui/build/finder.go
@@ -56,7 +56,7 @@ func NewSourceFinder(ctx Context, config Config) (f *finder.Finder) {
        cacheParams := finder.CacheParams{
                WorkingDirectory: dir,
                RootDirs:           []string{"."},
-               ExcludeDirs:        []string{".git", ".repo"},
+               ExcludeDirs:        []string{".git", ".repo", "fc2x"},
                PruneFiles:         pruneFiles,
                IncludeFiles: []string{
                        "Android.mk",
diff –git a/device/rockchip/common/wpa_config.txt    b/device/rockchip/common/wpa_config.txt
index9ca4ac78f4..88eaf47c83100644
---a/device/rockchip/common/wpa_config.txt
+++b/device/rockchip/common/wpa_config.txt
@@-25,3+25,8@@
-O/data/vendor/wifi/wpa/sockets
-g@android:wpa_wlan0
```

```
+[fc2x]
+/vendor/bin/hw/wpa_supplicant
+-O/data/vendor/wifi/wpa/sockets
+-puse_p2p_group_interface=1
+-g@android:wpa_wlan0
diff--gita/external/wpa_supplicant_8/wpa_supplicant/main.c
b/external/wpa_supplicant_8/wpa_supplicant/main.c
index04ba5b1371..8c1c3b5115100755
---a/external/wpa_supplicant_8/wpa_supplicant/main.c
+++b/external/wpa_supplicant_8/wpa_supplicant/main.c
@@-413,6+413,7@@out:
#defineSSV_MODULE_NAME"[ssv]"
#defineESP_MODULE_NAME"[esp]"
#defineSPRDWL_MODULE_NAME"[sprdwl]"
+#defineFC2X_MODULE_NAME"[fc2x]"

int read_wpa_param_config(char*module_name,char*file_path){
        char*wpa_param[50]={0};
@@-426,6+427,7@@int read_wpa_param_config(char*module_name,char*file_path){
        inti;
        fp=fopen(file_path,"r");
        wpa_printf(MSG_INFO,"module_name=%s\n",module_name);
+       wpa_printf(MSG_INFO,"file_path=%s\n",file_path);
        if(fp==NULL){
                wpa_printf(MSG_ERROR,"%snotfound\n",file_path);
                return-1;
@@-492,7+494,11@@int main(intargc,char*argv[])
                }else if(0==strncmp(wifi_type,"SPRDWL",6)){
                        wpa_printf(MSG_INFO,"Startsprdwl_wpa_supplicant\n");
                        ret=read_wpa_param_config(SPRDWL_MODULE_NAME,argv[1]);
-               }else{
+               }else if(0==strncmp(wifi_type,"FC2X",4)){
+                       wpa_printf(MSG_INFO,"Startfc2x_wpa_supplicant\n");
+                       ret=read_wpa_param_config(FC2X_MODULE_NAME,argv[1]);
+               }
+               else{
                        wpa_printf(MSG_INFO,"Startwpa_supplicant\n");
                        sprintf(module_type,"[%s]",wifi_type);
                        ret=read_wpa_param_config(module_type,argv[1]);
diff—gita/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp
b/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp
index95e5ed0ab6..3602e7df07100755
---a/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp
+++b/frameworks/opt/net/wifi/libwifi_hal/rk_wifi_ctrl.cpp
@@-73,6+73,7@@static wifi_device supported_wifi_devices[]={
      {"RTL8822BE",    "10ec:b822"},
      {"MVL88W8977", "02df:9145"},
      {"SPRDWL", "0000:0000"},
+     {"FC2X","0271:0701"},
};

int get_wifi_device_id(constchar*bus_dir,constchar*prefix)
diff—gita/frameworks/opt/net/wifi/libwifi_hal/wifi_hal_common.cpp
b/frameworks/opt/net/wifi/libwifi_hal/wifi_hal_common.cpp
index4b6c6ec215..5fdfb984e7100755
---a/frameworks/opt/net/wifi/libwifi_hal/wifi_hal_common.cpp
+++b/frameworks/opt/net/wifi/libwifi_hal/wifi_hal_common.cpp
@@-56,6+56,7@@extern"C"intdelete_module(constchar*,unsignedint);
#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    FC2X_DRIVER_MODULE_PATH WIFI_MODULE_PATH"wlan.ko"
#define DRIVER_MODULE_PATH_UNKNOW     ""

#define RTL8188EU_DRIVER_MODULE_NAME "8188eu"
@@-81,6+82,7@@extern"C"intdelete_module(constchar*,unsignedint);
#define   MVL_DRIVER_MODULE_NAME    "sd8xxx"
#define   RK912_DRIVER_MODULE_NAME"rk912"
#define   SPRDWL_DRIVER_MODULE_NAME   "sprdwl"
+#define FC2X_DRIVER_MODULE_NAME         "wlan"
#define     DRIVER_MODULE_NAME_UNKNOW      ""

#ifndef WIFI_DRIVER_FW_PATH_STA
@@-164,6+166,7@@wifi_ko_file_namemodule_list[]=
      {"MVL88W8977",MVL_DRIVER_MODULE_NAME,MVL_DRIVER_MODULE_PATH,MVL88W897
7_DRIVER_MODULE_ARG},
{"RK912",RK912_DRIVER_MODULE_NAME,RK912_DRIVER_MODULE_PATH,UNKKOWN_DRIVE
R_MODULE_ARG},
      {"SPRDWL",
      SPRDWL_DRIVER_MODULE_NAME,SPRDWL_DRIVER_MODULE_PATH,UNKKOWN_DRIVE
R_MODULE_ARG},
+     {"FC2X",
      FC2X_DRIVER_MODULE_NAME,FC2X_DRIVER_MODULE_PATH,UNKKOWN_DRIVER_MOD
ULE_ARG},
      {"UNKNOW",DRIVER_MODULE_NAME_UNKNOW,DRIVER_MODULE_PATH_UNKNOW,UNKK
OWN_DRIVER_MODULE_ARG}

};
```

```
@@-453,12+456,16@@constchar*wifi_get_fw_path(intfw_type){
}

int wifi_change_fw_path(constchar*fwpath){
-int len;
-int fd;
+//int len;
+//int fd;
int ret=0;

if(wifi_type[0]==0)
        check_wifi_chip_type_string(wifi_type);
+
+
+#if 0
if(0!=strncmp(wifi_type,"AP",2)) return ret;
if(!fwpath)return ret;
fd=TEMP_FAILURE_RETRY(open(WIFI_DRIVER_FW_PATH_PARAM,O_WRONLY));
@@-472,5+479,6@@int wifi_change_fw_path(constchar*fwpath){
ret=-1;
}
close(fd);
+#endif
return ret;
}
diff--gita/kernel/arch/arm64/configs/firefly_defconfig b/kernel/arch/arm64/configs/firefly_defconfig
index80ce6690d2..c4f375ba15100644
---a/kernel/arch/arm64/configs/firefly_defconfig
+++b/kernel/arch/arm64/configs/firefly_defconfig
@@-415,6+415,7@@CONFIG_USB_SIERRA_NET=y
#CONFIG_WLAN_VENDOR_ZYDAS is not set
#CONFIG_WLAN_VENDOR_QUANTENNA is not set
CONFIG_WL_ROCKCHIP=y
+CONFIG_NL80211_TESTMODE=y
CONFIG_WIFI_BUILD_MODULE=y
CONFIG_AP6XXX=m
CONFIG_RTL8723CS=m
diff--gita/kernel/drivers/base/firmware_loader/main.c b/kernel/drivers/base/firmware_loader/main.c
index7ed12a523d..583578deb8100644
---a/kernel/drivers/base/firmware_loader/main.c
+++b/kernel/drivers/base/firmware_loader/main.c
@@-279,10+279,10@@static void free_fw_priv(structfw_priv*fw_priv)
static char fw_path_para[256];
static const char* constfw_path[]={
      fw_path_para,
-     "/lib/firmware/updates/"UTS_RELEASE,
-     "/lib/firmware/updates",
-     "/lib/firmware/"UTS_RELEASE,
-     "/lib/firmware"
+     "/data/lib/firmware/updates/"UTS_RELEASE,
+     "/data/lib/firmware/updates",
+     "/data/lib/firmware/"UTS_RELEASE,
+     "/data/lib/firmware"
};


diff--gita/kernel/include/net/cfg80211.h b/kernel/include/net/cfg80211.h
index dec1f4303d..f597a78f6b100644
---a/kernel/include/net/cfg80211.h
+++b/kernel/include/net/cfg80211.h
@@-3661,6+3661,7@@enum wiphy_flags{
      WIPHY_FLAG_SUPPORTS_5_10_MHZ          =BIT(22),
      WIPHY_FLAG_HAS_CHANNEL_SWITCH         =BIT(23),
      WIPHY_FLAG_HAS_STATIC_WEP           =BIT(24),
+     WIPHY_FLAG_DFS_OFFLOAD=BIT(25)
};

/**
@@-4965,6+4966,31@@void cfg80211_send_layer2_update(struct net_device*dev,const u8*addr);
*/
int regulatory_hint(struct wiphy*wiphy,const char* alpha2);

+/**
+*regulatory_hint_user-hint to the wireless core a regulatory domain
+*which the driver has received from an application
+*@alpha2:the ISO/IEC 3166 alpha2 the driver claims its regulatory domain
+*should be in.If @rd is set this should be NULL.Note that if you
+*set this to NULL you should still set rd->alpha2 to some accepted
+*alpha2.
+*@user_reg_hint_type:the type of user regulatory hint.
+*
+*Wireless driver scan use this function to hint to the wireless core
+*the current regulatory domain as specified by trusted applications,
+*it is the driver's responsibility to establish which applications it
+*trusts.
+*
+*The wiphy should be registered to cfg80211 prior to this call.
+*For cfg80211 drivers this means you must first use wiphy_register(),
+*for mac 80211 drivers you must first use ieee80211_register_hw().
+*
+*Drivers should check there turn value, it's possible you can get
+*an-ENOMEMoran-EINVAL.
+*
+*Return:0 on success.-ENOMEM,-EINVAL.
+*/
+int regulatory_hint_user(const char*alpha2,enum nl80211_user_reg_hint_type user_reg_hint_type);
+
/**
*regulatory_set_wiphy_regd-set regdom info for self-managed drivers
*@wiphy:the wireless device we want to process the regulatory domain on
@@-6778,6+6804,16@@void cfg80211_nan_func_terminated(struct wireless_dev* wdev,
/*eth tool helper*/
void cfg80211_get_drvinfo(struct net_device*dev,struct ethtool_drvinfo* info);

+/**
+**cfg80211_is_gratuitous_arp_unsolicited_na – packet is grat.ARP/unsol.NA
+**@skb:the input packet,must bean ethernet frame already
+**
+**Return:%true if the packet is agratuitous ARP or unsolicited NA packet.
+**This is used to drop packets that shouldn't occur because the AP implements
+**a proxy service.
+**/
+bool cfg80211_is_gratuitous_arp_unsolicited_na(struct sk_buff*skb);
+
/**
*cfg80211_external_auth_request-userspace request for authentication
*@netdev:network device
diff--gita/kernel/net/wireless/reg.cb/kernel/net/wireless/reg.c
indexc7825b951f..0fa9aaaf2c100644
---a/kernel/net/wireless/reg.c
+++b/kernel/net/wireless/reg.c
@@-2956,6+2956,7@@int regulatory_hint_user(const char*alpha2,

    return0;
}
+EXPORT_SYMBOL(regulatory_hint_user);

int regulatory_hint_indoor(bool is_indoor,u32 portid)
{
diff--gita/kernel/net/wireless/util.cb/kernel/net/wireless/util.c
index1178128361..74b85aec54100644
---a/kernel/net/wireless/util.c
```

```
+++b/kernel/net/wireless/util.c
@@-1980,6+1980,57@@const unsigned char bridge_tunnel_header[]__aligned(2)=
        {0xaa,0xaa,0x03,0x00,0x00,0xf8};
EXPORT_SYMBOL(bridge_tunnel_header);

+bool cfg80211_is_gratuitous_arp_unsolicited_na(struct sk_buff*skb)
+{
+const struct ethhdr*eth=(void*)skb->data;
+const struct{
+struct arphdrhdr;
+u8 ar_sha[ETH_ALEN];
+u8 ar_sip[4];
+u8 ar_tha[ETH_ALEN];
+u8 ar_tip[4];
+}__packed*arp;
+const struct ipv6hdr* ipv6;
+const struct icmp6hdr* icmpv6;
+
+switch(eth->h_proto){
+case cpu_to_be16(ETH_P_ARP):
+/*can'tsay-but will probably be dropped later anyway*/
+if(!pskb_may_pull(skb,sizeof(*eth)+sizeof(*arp)))
+return false;
+
+arp=(void*)(eth+1);
+
+if((arp->hdr.ar_op==cpu_to_be16(ARPOP_REPLY)||
+arp->hdr.ar_op==cpu_to_be16(ARPOP_REQUEST))&&
+!memcmp(arp->ar_sip,arp->ar_tip,sizeof(arp->ar_sip)))
+return true;
+break;
+case cpu_to_be16(ETH_P_IPV6):
+/*can't say-but will probably be dropped later anyway*/
+if(!pskb_may_pull(skb,sizeof(*eth)+sizeof(*ipv6)+
+sizeof(*icmpv6)))
+return false;
+
+ipv6=(void*)(eth+1);
+icmpv6=(void*)(ipv6+1);
+
+if(icmpv6->icmp6_type==NDISC_NEIGHBOUR_ADVERTISEMENT&&
+!memcmp(&ipv6->saddr,&ipv6->daddr,sizeof(ipv6->saddr)))
+return true;
+break;
+default:
+/*
+*no need to support other protocols,proxy service isn't
+*specified for any others
+*/
+break;
+}
+
+return false;
+}
+EXPORT_SYMBOL(cfg80211_is_gratuitous_arp_unsolicited_na);
+
bool cfg80211_iftype_allowed(struct wiphy*wiphy,enum nl80211_iftype iftype,
                    bool is_4addr,u8 check_swif)
```

2.2.4. Build Wi-Fi Driver to Android 11

There are two ways to compile the Wi-Fi driver source code. You can choose between ***Chapter 2.2.4.1*** and ***Chapter 2.2.4.2*** depending on your needs.

2.2.4.1. Integrate the Build-in Wi-Fi Driver to Android

2.2.4.1.1. Storage the Wi-Fi Driver Source Code to Kernel

Store the cloned Wi-Fi driver source code folder *fc2x* to the *{Kernelroot}/drivers/net/wireless/rockchip_wlan* directory.

2.2.4.1.2. Modify the Relevant Configuration of the Built-in Wi-Fi Driver

Modify the relevant configuration of the built-in Wi-Fi driver as follows:

```
diff--gita/kernel/drivers/net/wireless/rockchip_wlan/Kconfig
b/kernel/drivers/net/wireless/rockchip_wlan/Kconfig
indexc68755edcb..3f1836c06a100644
---a/kernel/drivers/net/wireless/rockchip_wlan/Kconfig
+++b/kernel/drivers/net/wireless/rockchip_wlan/Kconfig
@@-29,11+29,15@@config WIFI_GENERATE_RANDOM_MAC_ADDR
        Wifi generate random mac address and save to vendor storage for cob chip

Menuconfig BCMDHD
-       bool "Broadcom Wireless Device Driver Support"
-       default y
```

```
+    tristate "Broadcom Wireless Device Driver Support"
+    default m
+
+menuconfig QCA_CLD_WLAN
+    tristate "Quectel Wireless Device Driver Support"
+    default m

if BCMDHD
-source"drivers/net/wireless/rockchip_wlan/rkwifi/Kconfig"
+#source"drivers/net/wireless/rockchip_wlan/rkwifi/Kconfig"
endif

menuconfigRTL_WIRELESS_SOLUTION
@@-50,7+54,11@@source"drivers/net/wireless/rockchip_wlan/rtl8821cs/Kconfig"
source"drivers/net/wireless/rockchip_wlan/rtl8822bs/Kconfig"
endif

source"drivers/net/wireless/rockchip_wlan/ahd/Kconfig"
source"drivers/net/wireless/rockchip_wlan/mvl88w8977/Kconfig"
source"drivers/net/wireless/rockchip_wlan/cywdhd/Kconfig"
+source"drivers/net/wireless/rockchip_wlan/fc2x/cnss_host_LEA/chss_proc/host/AIO/drivers/qcacld-ne
w/Kconfig"
+

endif#WL_ROCKCHIP
diff--gita/kernel/drivers/net/wireless/rockchip_wlan/Makefile
b/kernel/drivers/net/wireless/rockchip_wlan/Makefile
index15b6ce2e86..e9b25880d1100644
---a/kernel/drivers/net/wireless/rockchip_wlan/Makefile
+++b/kernel/drivers/net/wireless/rockchip_wlan/Makefile
@@-1,12+1,19@@
#SPDX-License-Identifier:GPL-2.0
obj-$(CONFIG_BCMDHD) +=rkwifi/
obj-$(CONFIG_RTL8188EU)    +=rtl8188eu/
obj-$(CONFIG_RTL8188FU)    +=rtl8188fu/
obj-$(CONFIG_RTL8189FS)    +=rtl8189fs/
obj-$(CONFIG_BCMDHD) +=rkwifi/
+
obj-$(CONFIG_BCMDHD) +=ahd/
obj-$(CONFIG_RTL8188EU)    +=rtl8188eu/
obj-$(CONFIG_RTL8188FU)    +=rtl8188fu/
obj-$(CONFIG_RTL8189FS)    +=rtl8189fs/
obj-$(CONFIG_RTL8723CS)    +=rtl8723cs/
obj-$(CONFIG_RTL8723DS)    +=rtl8723ds/
```

```
obj-$(CONFIG_RTL8821CS)+=rtl8821cs/
obj-$(CONFIG_RTL8822BS)+=rtl8822bs/
obj-$(CONFIG_MVL88W8977) +=mvl88w8977/
obj-$(CONFIG_RTL8822BS)+=rtl8822bs/
obj-$(CONFIG_MVL88W8977) +=mvl88w8977/
obj-$(CONFIG_WL_ROCKCHIP)     +=rkwifi/rk_wifi_config.o
+obj-$(CONFIG_QCA_CLD_WLAN)+=fc2x/cnss_host_LEA/chss_proc/host/AIO/drivers/qcacld-new/

diff--git
---a/kernel/drivers/net/wireless/rockchip_wlan/fc2x/cnss_host_LEA/chss_proc/host/AIO/
drivers/qcacld-new/Kbuild
+++b/kernel/drivers/net/wireless/rockchip_wlan/fc2x/cnss_host_LEA/chss_proc/host/AIO/
drivers/qcacld-new/Kbuild

@@-1,5+1,68@@
+AIO_VER=1.4.2.te-f30..001
+BOARD_TYPE=te-f30
+HOST_DRIVER_VERSION=4.5.25.51
+BLUETOOTHSTACK=
+BOARD_TYPE_AIO_PATCH_CAF=https://portland.source.codeaurora.org/patches/external/wlan/fixc
e/3rdparty/patches/wlan_patches
+ENG_PATCH=0
+KTAG=
+
+CONFIG_NO_USE_BACKPORTS=y
+CONFIG_CFG80211_DEPEND_ON_KERNEL=y
+
+CONFIG_CFG80211_INTERNAL_REGDB=y
+CONFIG_PMF_SUPPORT=y
+
+CONFIG_LINUX_QCMBR=y
+
+CONFIG_NON_QC_PLATFORM=y
+
+CONFIG_WLAN_THERMAL_SHUTDOWN=0
+
+CONFIG_CLD_HL_SDIO_CORE=y
+CONFIG_LINUX_QCMBR=y
+CONFIG_PER_VDEV_TX_DESC_POOL=1
+SAP_AUTH_OFFLOAD=1
+CONFIG_QCA_LL_TX_FLOW_CT=1
+CONFIG_WLAN_FEATURE_FILS=y
+CONFIG_FEATURE_COEX_PTA_CONFIG_ENABLE=y
+CONFIG_QCA_SUPPORT_TXRX_DRIVER_TCP_DEL_ACK=y
```

```
+CONFIG_WLAN_WAPI_MODE_11AC_DISABLE=y
+CONFIG_WLAN_WOW_PULSE=y
+CONFIG_TXRX_PERF=y

+
+KBUILD_OPTIONS:=WLAN_ROOT=$(PWD)
+
+KBUILD_OPTIONS+=MODNAME?=wlan+
+
+LICENSE_FILE?=/home/sai/git/tpbgit/firefly/Android11.0/Firefly-RK356X_Android11.0_git_20210824/
ROCKCHIP_ANDROID11.0_SDK_RELEASE/RK356X_Android11.0/kernel/drivers/net/wireless/rockchi
p_wlan/fc2x/cnss_host_LEA/chss_proc/host/AIO/drivers/qcacld-new/CORE/HDD/src/wlan_hdd_main.c
+
+$(info"LICENSE_FILEis"$(LICENSE_FILE))
+WLAN_OPEN_SOURCE=$(shellifgrep-q"MODULE_LICENSE(\"DualBSD/GPL\")"\
+$(LICENSE_FILE);thenecho1;elseecho0;fi)
+
+#BydefaultbuildforCLD
+WLAN_SELECT:=CONFIG_QCA_CLD_WLAN=m
+KBUILD_OPTIONS+=CONFIG_QCA_WIFI_ISOC=0
+KBUILD_OPTIONS+=CONFIG_QCA_WIFI_2_0=1
+KBUILD_OPTIONS+=$(WLAN_SELECT)
+KBUILD_OPTIONS+=WLAN_OPEN_SOURCE=$(WLAN_OPEN_SOURCE)
+KBUILD_OPTIONS+=$(KBUILD_EXTRA)#Extraconfigifany
+
+
ifeq($(MODNAME),)
KERNEL_BUILD:=1
else
@@-18,8+81,10@@endif
ifeq($(KERNEL_BUILD),1)
#These are provided in external module-based builds
#Need to explicitly define for Kernel-based builds
MODNAME:=wlan
-WLAN_ROOT:=drivers/staging/qcacld-2.0
+WLAN_ROOT:=drivers/net/wireless/rockchip_wlan/fc2x/cnss_host_LEA/chss_proc/host/AIO/drivers/q
cacld-new
WLAN_OPEN_SOURCE:=1
endif

@@-299,6+364,7@@CONFIG_MDNS_OFFLOAD_SUPPORT:=1
endif
endif
```

```
+
#Enable power management suspend/resume functionality on PCI device
CONFIG_ATH_BUS_PM:=1

@@-377,6+443,7@@HAVE_CFG80211:=0
endif
endif

+
############COMMON############
COMMON_DIR:=CORE/SERVICES/COMMON
COMMON_INC:=-I$(WLAN_ROOT)/$(COMMON_DIR)
@@-454,6+521,12@@HDD_SRC_DIR:=$(HDD_DIR)/src
HDD_INC:=-I$(WLAN_ROOT)/$(HDD_INC_DIR)\
-I$(WLAN_ROOT)/$(HDD_SRC_DIR)
```

NOTE

When compiling the Android 11 system, the built-in Wi-Fi driver can realize automatic compilation.

2.2.4.2. Build External Wi-Fi Driver to Android

2.2.4.2.1. Configure Cross-Compilation Tool

Before compiling the Wi-Fi driver, execute the following commands to set the environment variables for the cross-compilation tool.

```
export CROSS_COMPILE=aarch64-linux-gnu-
export ARCH=arm64
export PATH=${androidpath}/prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin:$PATH
```

NOTE

${androidpath} is the actual storage path of the user's Android source code.

2.2.4.2.2. Configure Build Environment

Execute the following commands to configure the corresponding kernel path and cross compilation tool chain of the Wi-Fi driver:

```
$ cd <FC2x_target_root>/cnss_host_LEA/chss_proc/host/AIO/build
$ vim scripts/te-f30/config.te-f30            // Modify KERNELPATH, KERNELARCH and TOOLPREFIX
            export KERNELPATH=${androidpath}/kernel
            export KERNELARCH=arm64
            export TOOLPREFIX=${CROSS_COMPILE}
$ cd <FC2x_target_root>/cnss_host_LEA/chss_proc/host/AIO/drivers/qcacld-new
$ vim Makefile                                // Modify KERNEL_SRC
      KERNEL_SRC ?= ${androidpath}/kernel

$ cd <FC2x_target_root>/cnss_host_LEA/chss_proc/host/AIO/driver
$ mkdir firmware
$ cd firmware
$ mkdir WLAN-firmware
$ mkdir BT-firmware
```

NOTE

<FC2x_target_root> in this document is the folder where the user actually stores the Wi-Fi driver code.

2.2.4.2.3. Build Wi-Fi Driver

Execute the following commands to build the Wi-Fi driver. After a successful compilation, the generated .ko file is in: *<FC2x_target_root>/cnss_host_LEA/chss_proc/host/AIO/drivers/qcacld-new* directory:

```
$ cd <FC2x_target_root>/cnss_host_LEA/chss_proc/host/AIO/build
$ make   drivers
```

2.2.5. Build and Download Android 11

2.2.5.1. Modify Android 11 Build Script

To avoid image file generation failure caused by insufficient memory space of Android 11 kernel, the corresponding **strip** commands should be added to the *make.sh* script under *RK356X_Android11.0/FFTools* directory to reduce the size of the .ko file. The modifying part before **#build android** is shown in the blue font.

```
---/home/sai/git/labgit/fc900e/aosp/RK356X_Android11.0/FFTools/make.sh2022-07-2009:21:54.028217596+0800
+++make.sh2022-08-1113:47:40.472012201+0800
@@-122,6+122,10@@fi

fi
+aarch64-linux-gnu-strip --strip-debug /home/sai/git/tpbgit/firefly/Android11.0/Firefly-RK356X_Android11.0_git_20210824/ROCKCHIP_ANDROID11.0_SDK_RELEASE/RK356X_Android11.0/kernel/drivers/net/wireless/rockchip_wlan/fc2x/cnss_host_LEA/chss_proc/host/AIO/drivers/qcacld-new/wlan.ko

#build android
if["$BUILD_ANDROID"=true];then
echo
```

NOTE

Please set up the environment variables of the cross-compilation tool accordingly.

2.2.5.2. Build Android 11

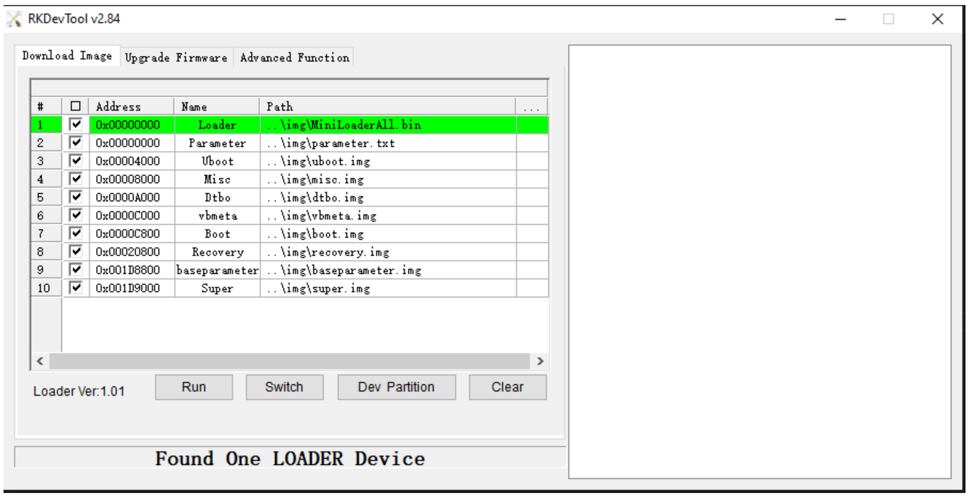
Execute the following two commands under the RK3566 Android 11 directory:

```
./FFTools/make.sh-drk3566-firefly-aiojd4-j8-lrk3566_firefly_aiojd4-userdebug
./FFTools/mkupdate/mkupdate.sh-lrk3566_firefly_aiojd4-userdebug
```

Once the compilation is completed, the image file is generated under the *andorid/rockdev/Image-rk3566_firefly_aiojd4* folder of RK3566 Android 11.

2.2.5.3. Download Android

Use RKDevTool v2.84 to download the image file. Once RK3566 EVB board enters Loader mode, start the downloading process shown in the figure below:



#	☐	Address	Name	Path	...
1	✓	0x00000000	Loader	..\img\MiniLoaderAll.bin	
2	✓	0x00000000	Parameter	..\img\parameter.txt	
3	✓	0x00004000	Uboot	..\img\uboot.img	
4	✓	0x00008000	Misc	..\img\misc.img	
5	✓	0x0000A000	Dtbo	..\img\dtbo.img	
6	✓	0x0000C000	vbmeta	..\img\vbmeta.img	
7	✓	0x0000C800	Boot	..\img\boot.img	
8	✓	0x00020800	Recovery	..\img\recovery.img	
9	✓	0x001D8800	baseparameter	..\img\baseparameter.img	
10	✓	0x001D9000	Super	..\img\super.img	

Loader Ver:1.01

Found One LOADER Device

Figure 2: Download Image File

NOTE

For more information about how to enter loader mode, see the following official website: https://wiki.t-firefly.com/en/Core-3566JD4/01-bootmode.html.

2.2.6. Add Wi-Fi Firmware Files to Android 11

Step 1: Once Android is successfully downloaded, the host connects to Android 11 through the ADB tool and obtains Root permission:

```
D:tool\scrcpy-win64-v1.24>adb root
D:tool\scrcpy-win64-v1.24>adb remount
D:tool\scrcpy-win64-v1.24>adb shell
Rk3566_firefly_aiojd4:/#
```

If an error is reported after **adb root** is executed, follow the steps below in the device Android system interface, and then re-execute the above commands.

- **setting** -> **About tablet** -> Click **Build number** 7 times consecutively
- **setting** -> **System** -> **Advanced** -> **Developer options** -> **Root access** -> **ADB only** option

Step 2: Push Wi-Fi firmware files to */data/lib/firmware* and */data/lib/firmware/wlan* via **adb push**. For example, if the user stores the firmware in *E:\driver\adb\adb>adb*:

```
E:\driver\adb\adb>adb push qcom_cfg.ini /data/lib/firmware/wlan
E:\driver\adb\adb>adb push utf30.bin /data/lib/firmware
E:\driver\adb\adb>adb push bdwlan30.bin /data/lib/firmware
E:\driver\adb\adb>adb push otp30.bin /data/lib/firmware
E:\driver\adb\adb>adb push qwlan30.bin /data/lib/firmware
```

NOTE

1. If there is no corresponding folder */data/lib/firmware* or */data/lib/firmware/wlan* when **adb push** is executed, create them by yourself.
2. Wi-Fi firmware files include *qwlan30.bin*, *bdwlan30.bin*, *otp30.bin*, *utf30.bin* and *qcom_cfg.ini*. Among them, .bin files are in the *<FC2x_target_root>/meta_build/load_meta/wlan_firmware/sdio* path while .ini file is in *<FC2x_target_root>/meta_build/load_meta/host/wlan_host/sdio* in the driver source code. The .bin files need to be transmitted to */data/lib/firmware*; the .ini file is transmitted to */data/lib/firmware/wlan*.
3. If you build external Wi-Fi driver to Android, the *wlan.ko* file needs to be pushed to RK356X Android 11 system with the following command:
   E:\driver\adb\adb>adb push wlan.ko /vendor/lib/modules.

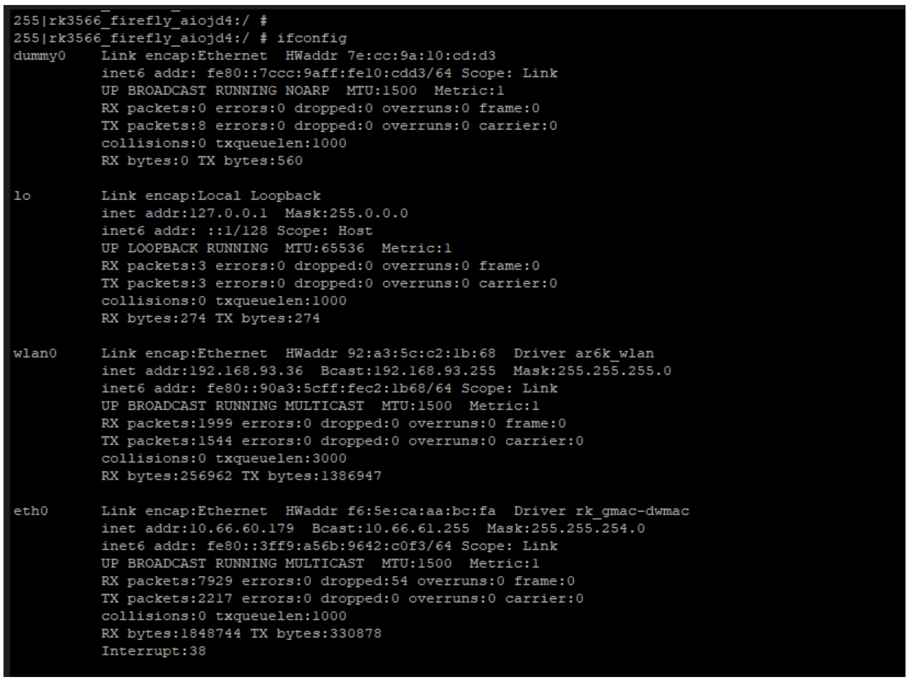
3 Function Verification

3.1. Wi-Fi Function Verification

Input the following commands in Android command line:

```
ifconfig   wlan0 up
ifconfig
```

If wlan0 port information is displayed in the command line, it indicates that the Wi-Fi driver and firmware have been loaded successfully, and the Wi-Fi function can be used normally. The results are shown below.

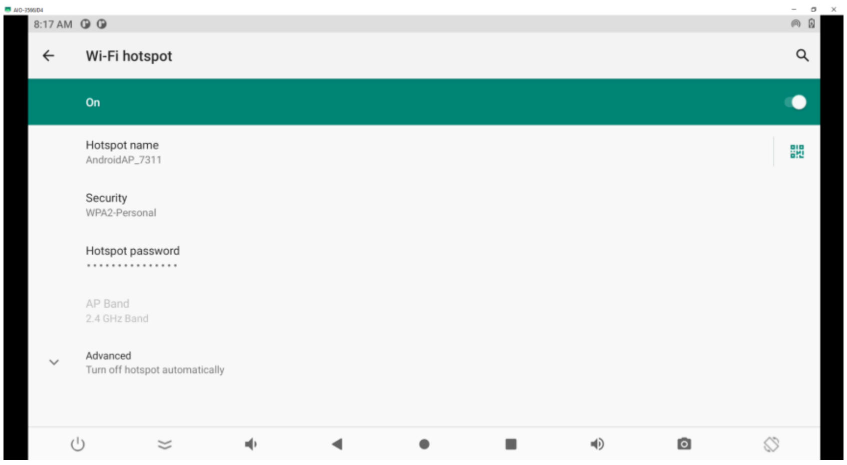


```
255|rk3566_firefly_aiojd4:/ #
255|rk3566_firefly_aiojd4:/ # ifconfig
dummy0    Link encap:Ethernet  HWaddr 7e:cc:9a:10:cd:d3
          inet6 addr: fe80::7ccc:9aff:fe10:cdd3/64 Scope: Link
          UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:560

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope: Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:274 TX bytes:274

wlan0     Link encap:Ethernet  HWaddr 92:a3:5c:c2:1b:68  Driver ar6k_wlan
          inet addr:192.168.93.36  Bcast:192.168.93.255  Mask:255.255.255.0
          inet6 addr: fe80::90a3:5cff:fec2:1b68/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1999 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1544 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:256962 TX bytes:1386947

eth0      Link encap:Ethernet  HWaddr f6:5e:ca:aa:bc:fa  Driver rk_gmac-dwmac
          inet addr:10.66.60.179  Bcast:10.66.61.255  Mask:255.255.254.0
          inet6 addr: fe80::3ff9:a56b:9642:c0f3/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7929 errors:0 dropped:54 overruns:0 frame:0
          TX packets:2217 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1848744 TX bytes:330878
          Interrupt:38
```

3.2. Wi-Fi Operating Mode Verification

3.2.1. AP Mode

Start the RK3566 Android 11 system to enable AP mode. In this example, the hotspot name is "AndroidAP_7311". If the AP is successfully searched and connected through the mobile phone, it indicates that the AP mode of RK3566 Android 11 system is enabled.


8:17 AM

Wi-Fi hotspot

On

Hotspot name
AndroidAP_7311

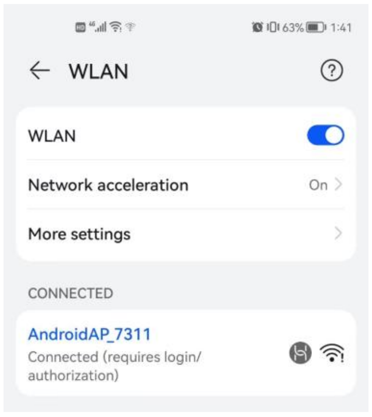
Security
WPA2-Personal

Hotspot password
•••••••••••••••

AP Band
2.4 GHz Band

Advanced
Turn off hotspot automatically

Figure 3: Search AP


63% 1:41

WLAN

WLAN

Network acceleration On

More settings

CONNECTED

AndroidAP_7311
Connected (requires login/ authorization)

Figure 4: Connect to AP Through Mobile Phone

3.2.2. STA Mode

Start RK3566 Android 11 system to enable STA mode. To enable STA mode, you need to make the following modifications in *wpa_supplicant_overlay.conf* under the */vendor/etc/wifi* path of the Android system.

```
p2p_disabled=1
disable_scan_offload=1
wowlan_triggers=any
filter_rssi=-75
no_ctrl_interface=
```

If the existing AP is successfully searched and connected and then you can surf the internet, it indicates that the STA mode of RK3566 Android 11 system is enabled.

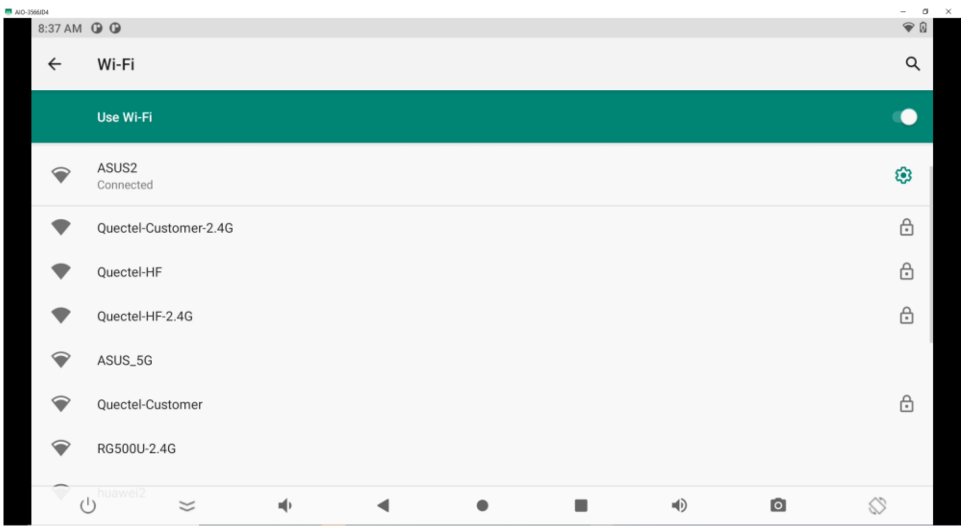


Figure 5: Connect AP Successfully

4 Appendix References

Table 2: Terms and Abbreviations

Abbreviation	Description
ADB	Android Debug Bridge
AP	Access Point
EVB	Evaluation Board
PCI	Peripheral Component Interconnect
PID	Product ID
SDK	Software Development Kit
STA	Station
USB	Universal Serial Bus
VID	Vendor ID