INGENIC®

T Series Chip WiFi Porting Guide

Date：2022-01

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

Ingenic Semiconductor Co. Ltd

Add: Junzheng R&D Center, Phase II, Zhongguancun Software Park, Dongbeiwangxi Road, Haidian District, Beijing, China
Tel: 86-10-56345000
Fax: 86-10-56345001
Http: //www.ingenic.com

Foreword

Overview

This document mainly introduces how to port WiFi with ingenIC-T series chips and analyzes the problems in porting process.

Product version

The following table lists the product versions corresponding to this document:

Product Name	Product Version

Target Readers

This document is intended for the following engineers:

- Technical Support Engineers
- Software Development Engineer

Revision records

Revision records contain the description of each document update, the latest document contains all updates made in previous version.

Date	Version	Revision Content
2022-01	1.0	First official version released

Contents

1 Classifications of WiFi Modules

WiFi is one of the most widely used wireless network transmission technologie s and is widely used in various intelligent IOT devices. Different application scenari os have different functional requirements for WiFi modules. WiFi can be divided int o two categories: transparent WiFi module and universal WiFi module.

Transparent WiFi module: Generally used in point-to-point data transmission sc enarios, the module has built-in WiFi driver and protocol. Users do not need to ca re about how WiFi protocol is implemented. Transparent WiFi module is generally used in MCU.

Universal WiFi module: such as USB or SDIO interface WiFi module on notebo ok, mobile phone and various embedded devices. WiFi protocol stack and driver w ork in Windows, Android, Linux and other operating systems. This document mainly introduces how to port the universal WiFi module to Ingenic platform.

Currently, Junzheng T series chips support the following WiFi models:

USB Interface	RTL8188FTV、RTL8188EUS、RTL8188ETV、RTL8723BU
	MT7601
	ATBM6022
	RDA5995
SDIO Interface	RTL8189ES、RTL8189ETV、RTL8189FTV
	MARVELL8801
	BCM6212A
	SSV6x5x
	Hi3881V100

Note: The WiFi module mentioned above was coordinated by Ingenic customers an d Ingenic engineers in actual projects.

2 WiFi Kernel Configuration

For SDIO/USB interface WiFi, first it is an SDIO/USB device, and then it has th e function of WiFi. Therefore, the device is registered with the SDIO card /USB de vice first. When the device is detected, the WiFi function of the device is driven. After that, commands and data are sent through SDIO protocol /USB protocol to in teract with the WiFi module. Therefore, we need to enable the kernel's SDIO/USB function first so that WiFi devices can be recognized.

The Kernel version used by T21/T30/T31 is kernel-3.10.14, and that used by T 40 is kernel-4.4.94. The configuration of the kernel varies according to the kernel v ersion.

2.1 Kernel-3.10.14 Configuration

2.1.1 USB WiFi Kernel Configuration

For USB interface WiFi, just configure the DWC2 controller to Host mode.

```
Device Drivers  --->
[*] USB support  --->
    <*>   Support for Host-side USB
    <*>   DesignWare USB2 DRD Core Support
          Driver Mode (Host Mode Only)  --->
```

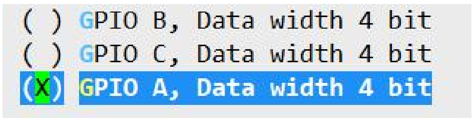
After the WiFi module is powered on, lsusb command can be used to check whether the USB WiFi module is correctly identified.

2.1.2 SDIO WiFi Kernel Configuration

There are two MSC interfaces on T series chips. Generally, MSC0 is reserved f or SD cards, while MSC1 interface is usually used by SDIO WiFi. The following take s MSC1 interface as an example to introduce how to configure the kernel.

```
Device Drivers  --->
   <*> MMC/SD/SDIO card support  --->
        [*] JZMMC_V12 MMC1
            JZMMC_V12 GPIO function pins select (GPIO A, Data wid
th 4 bit)
```

T series chips have multiple groups of GPIO which can be configured as MSC1 function. In actual projects, according to the design of hardware, the kernel can b e configured with which group of GPIO pins. For example, T31 GPIO which can be configured as MSC1 has three groups: PA, PB and PC.



```
( ) GPIO B, Data width 4 bit
( ) GPIO C, Data width 4 bit
(X) GPIO A, Data width 4 bit
```

2.2 Kernel-4.4.94 Configuration

2.2.3 USB Interface WiFi Kernel Configuration

For USB interface WiFi, just configure the DWC2 controller to Host mode.

```
Device Drivers  --->
   [*] USB support  --->
        <*>   Support for Host-side USB
        <*>   DesignWare USB2 DRD Core Support
               DWC2 Mode Selection (Host only mode)  --->
```

After the WiFi module is powered on, lsusb command can be used to check whether the USB WiFi module is correctly identified.

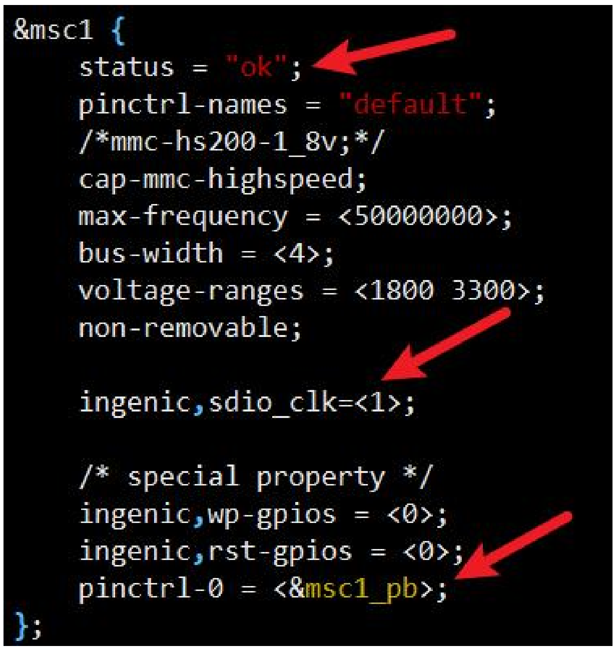
2.2.4 SDIO WiFi Kernel Configuration

There are two MSC interfaces on T series chips. Generally, MSC0 is reserved f or SD cards, while MSC1 interface is usually used by SDIO WiFi. The following take s MSC1 interface as an example to introduce how to configure the kernel.

Start by configuring the kernel to support MSC functionality:

```
  Device Drivers  --->
    <*> MMC/SD/SDIO card support  --->
      <*>   Ingenic(XBurst2)  MMC/SD Card Controller(MSC) supp
ort
```

Then in device tree file: kernel-4.4.94/arch/mips/boot/dts/ingenic/shark.dts:



```
&msc1 {
    status = "ok";
    pinctrl-names = "default";
    /*mmc-hs200-1_8v;*/
    cap-mmc-highspeed;
    max-frequency = <50000000>;
    bus-width = <4>;
    voltage-ranges = <1800 3300>;
    non-removable;

    ingenic,sdio_clk=<1>;

    /* special property */
    ingenic,wp-gpios = <0>;
    ingenic,rst-gpios = <0>;
    pinctrl-0 = <&msc1_pb>;
};
```

① Set status to OK to enable the MSC1 function

② Add property: ingenic,sdio_clk = <1>;

③ There are two sets of GPIO on the T40 that can be configured as MSC1 function: PB and PC. Pinctrl-0 can be configured as MSC1_PB or MSC1 _PC depending on the specific hardware design.

3 WiFi Driver Porting Process

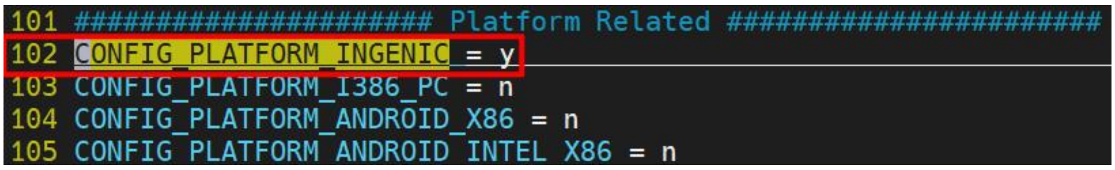
3.1 SDIO WiFi Driver Porting

The following use RTL8189FTV to explain the transplantation process of SDIO WiFi:

（1）Obtaining the driver source code of RTL8189FTV，rtl8189FS_linux_v5.3.16_32695.20190327-ingenic-20190618

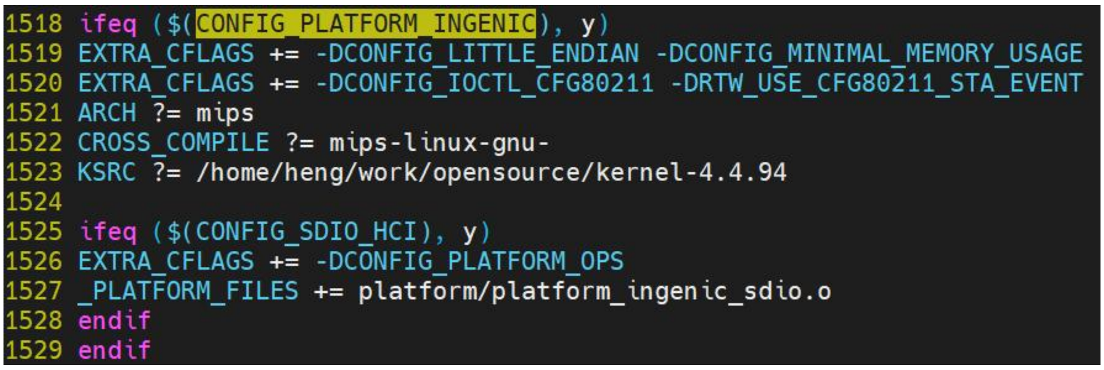
（2）SDIO WiFi Driver Porting

① Add a PLATFORM_INGENIC macro to the Makefile



```
######################## Platform Related ########################
CONFIG_PLATFORM_INGENIC = y
CONFIG_PLATFORM_I386_PC = n
CONFIG_PLATFORM_ANDROID_X86 = n
CONFIG_PLATFORM_ANDROID_INTEL_X86 = n
```

② Add an action about SDIO support to the Makefile



```
ifeq ($(CONFIG_PLATFORM_INGENIC), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_MINIMAL_MEMORY_USAGE
EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
ARCH ?= mips
CROSS_COMPILE ?= mips-linux-gnu-
KSRC ?= /home/heng/work/opensource/kernel-4.4.94

ifeq ($(CONFIG_SDIO_HCI), y)
EXTRA_CFLAGS += -DCONFIG_PLATFORM_OPS
_PLATFORM_FILES += platform/platform_ingenic_sdio.o
endif
endif
```

③ Create the platform_ingenic_sdio.c file in platform directory of driver source code and add corresponding functions. Related to Ingenic platform is the jzmmc_manual_detect (int sdio_index, int on) function, which is used to actively detect/remove SDIO devices, where:

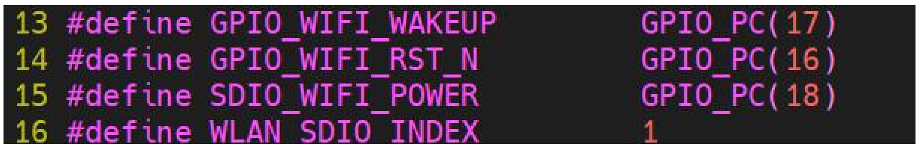
(i) The sdio_index parameter indicates the serial number of MSC controller. T series has two MSC controllers, corresponding to 0 and 1.

(ii) on=1 means actively detect device; on=0 means actively remove device;

(iii) Note that T2x/T3x does not add the same header file as T40

- T2x/T3x correspond to #include <mach/jzmmc.h>
- T40 correspond to #include <soc/mmc.h>

Note: the operation of the pins in this file should be consistent with the hardware.

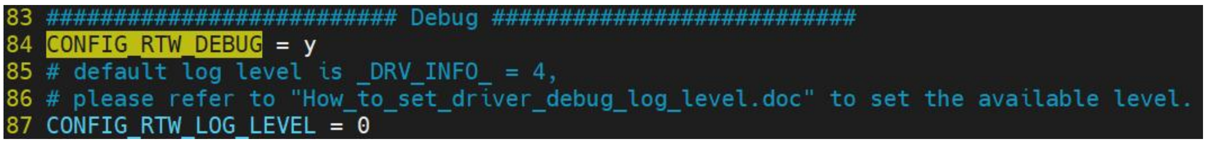


```
#define GPIO_WIFI_WAKEUP        GPIO_PC(17)
#define GPIO_WIFI_RST_N         GPIO_PC(16)
#define SDIO_WIFI_POWER         GPIO_PC(18)
#define WLAN_SDIO_INDEX         1
```

SDIO_WIFI_POWER is the power pin, which controls the power supply of WiFi module by setting GPIO to output mode.

After the migration, the compiled code generates the 8189Fs.ko driver, which is then used for testing. By default, the driver code turns on the DEBUG macro, which generates a large number of DEBUG messages during debugging. If you want to disable or reduce the output of DEBUG messages, you can modify the following sections in the Makefile:

CONFIG_RTW_LOG_LEVEL 为 0 ~ 4



```
########################## Debug ###########################
CONFIG_RTW_DEBUG = y
# default log level is _DRV_INFO_ = 4,
# please refer to "How_to_set_driver_debug_log_level.doc" to set the available level.
CONFIG_RTW_LOG_LEVEL = 0
```

3.2 SDIO WiFi Function Test

WIFI has two working modes, AP mode and STA mode. Test whether SDIO WIFI functions normally in these two modes.

3.2.1 STA Mode Network Test

In this mode, T series products equipped with WiFi serve as a terminal in the wireless network. The applications, configuration files, and dynamic libraries required for this pattern are as follows:

（1）Applications and configuration file

wpa_supplicant //Responsible for wifi authentication related login, encryption.etc

wpa_supplicant.conf //Configure the name and password of the wifi hotspot

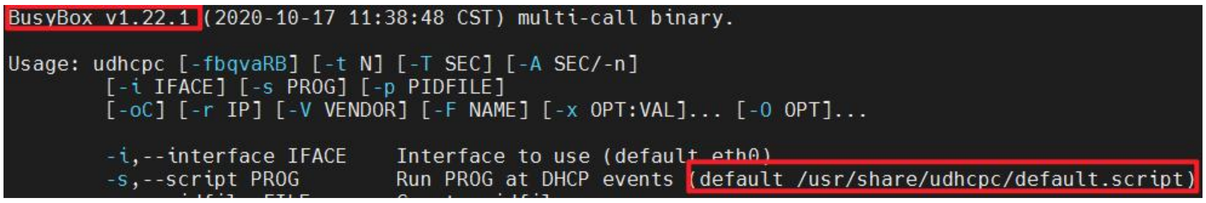
```
ctrl_interface=/var/run/wpa_supplicant
update_config=1

network={
```

```
            ssid="JZ_Guest"
            psk="#wwwingeniccn#"
}
```

udhcpc //used to dynamically obtain IP addresses, gateways, and DNS information

default.script//Used to automatically set the IP, gateway, DNS, etc. This file is renamed from busybox-x.x/examples/udhcp/simple.script file in the source code of busybox, Use udhcpc -h to see which path need to put default.script under.



```
BusyBox v1.22.1 (2020-10-17 11:38:48 CST) multi-call binary.

Usage: udhcpc [-fbqvaRB] [-t N] [-T SEC] [-A SEC/-n]
        [-i IFACE] [-s PROG] [-p PIDFILE]
        [-oC] [-r IP] [-V VENDOR] [-F NAME] [-x OPT:VAL]... [-O OPT]...

        -i,--interface IFACE    Interface to use (default eth0)
        -s,--script PROG        Run PROG at DHCP events (default /usr/share/udhcpc/default.script)
```

（2）The dynamic library used

libcrypto.so.1.0.0 // Encryption related

libiw.so.29 // iw related command use this library: iwlist,iwconfig,etc

libnl.so.1 // library required by hostpad

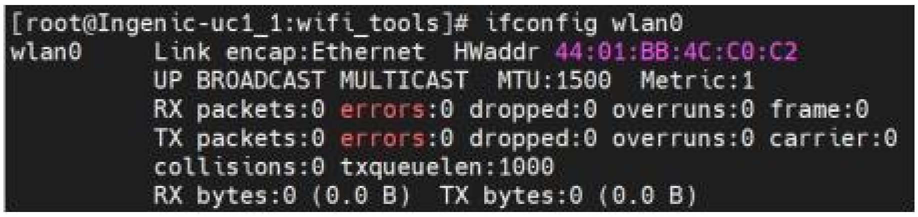
libssl.so.1.0.0 // library required by openssl

Place the dynamic libraries under the appropriate lib directory on your system.

（3）Test Process

① Load 8189fs.ko, if successful, the WLAN0 nic is generated

② Start the nic ifconfig wlan0 up

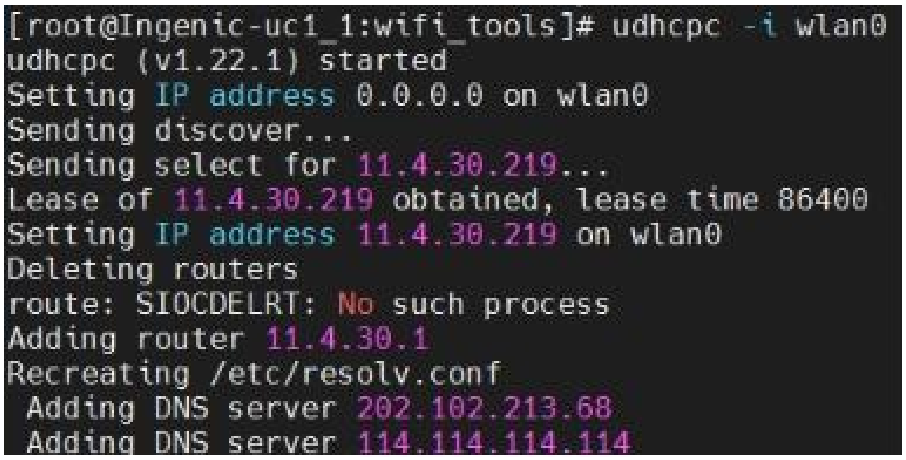


```
[root@Ingenic-uc1_1:wifi_tools]# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 44:01:BB:4C:C0:C2
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

③ Put the above four dynamic library files in the /system/lib directory

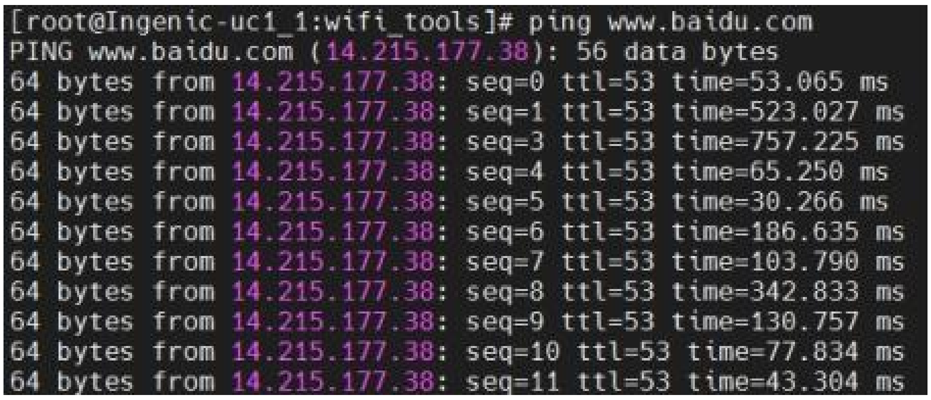
④ ./wpa_supplicant -i wlan0 -Dnl80211 -c wpa_supplicant.conf &

⑤ udhcpc -i wlan0 // Dynamically obtain IP address



```
[root@Ingenic-uc1_1:wifi_tools]# udhcpc -i wlan0
udhcpc (v1.22.1) started
Setting IP address 0.0.0.0 on wlan0
Sending discover...
Sending select for 11.4.30.219...
Lease of 11.4.30.219 obtained, lease time 86400
Setting IP address 11.4.30.219 on wlan0
Deleting routers
route: SIOCDELRT: No such process
Adding router 11.4.30.1
Recreating /etc/resolv.conf
 Adding DNS server 202.102.213.68
 Adding DNS server 114.114.114.114
```

⑥ ping www.baidu.com // Test if internet works normally



```
[root@Ingenic-uc1_1:wifi_tools]# ping www.baidu.com
PING www.baidu.com (14.215.177.38): 56 data bytes
64 bytes from 14.215.177.38: seq=0 ttl=53 time=53.065 ms
64 bytes from 14.215.177.38: seq=1 ttl=53 time=523.027 ms
64 bytes from 14.215.177.38: seq=3 ttl=53 time=757.225 ms
64 bytes from 14.215.177.38: seq=4 ttl=53 time=65.250 ms
64 bytes from 14.215.177.38: seq=5 ttl=53 time=30.266 ms
64 bytes from 14.215.177.38: seq=6 ttl=53 time=186.635 ms
64 bytes from 14.215.177.38: seq=7 ttl=53 time=103.790 ms
64 bytes from 14.215.177.38: seq=8 ttl=53 time=342.833 ms
64 bytes from 14.215.177.38: seq=9 ttl=53 time=130.757 ms
64 bytes from 14.215.177.38: seq=10 ttl=53 time=77.834 ms
64 bytes from 14.215.177.38: seq=11 ttl=53 time=43.304 ms
```

3.2.2 AP Mode Network Test

AP mode is the wireless access point, which is the founder of a wireless network and the central node of the network.

（1）Application and configuration file

hostapd // Switch the wifi network adapter to master mode and simulate the AP function as the AP authentication server. It controls the access and authentication of the Stations

hostapd.conf // hostapd configuration file: generated wifi hotspot name, password, and encryption mode.etc

```
interface=wlan0
ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
ssid=TEST_Wifi
hw_mode=g
channel=1
beacon_int=100
driver=nl80211
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
own_ip_addr=127.0.0.1
wpa=2
wpa_passphrase=12345678
rsn_pairwise=TKIP CCMP
```

udhcpd // dhcp server: assigns IP addresses, gateways to DHCP clients.etc

udhcpd.conf // udhcpd configuration file: responsible for assigning IP addresses, gateways, and DNS

```
start           192.168.1.20    #default: 192.168.0.20
end             192.168.1.254   #default: 192.168.0.254
interface       wlan0
```

（2）Dynamic library used

libcrypto.so.1.0.0 //Encryption related

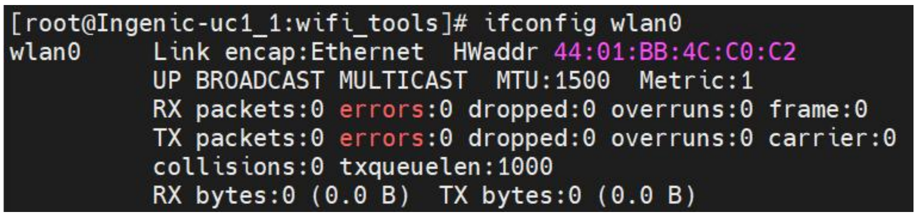
libnl.so.1 // Library required by hostpad

libssl.so.1.0.0 //Library required by openssl

（3）Test process

① Load 8189fs.ko, After the loading, the WLAN0 nic is generated

② Start the network adapter and configure the IP address, ifconfig wlan0 192.168.1.10 up



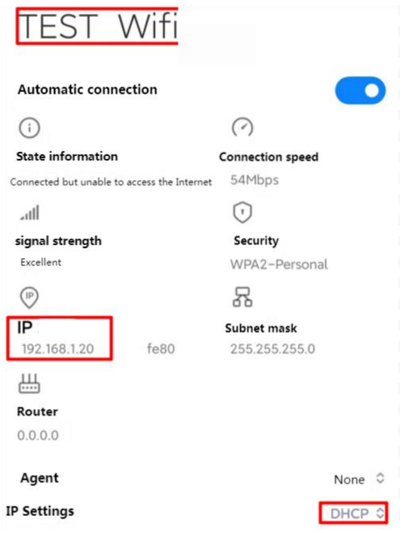
```
[root@Ingenic-uc1_1:wifi_tools]# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 44:01:BB:4C:C0:C2
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

③ Put the three dynamic library files in the /system/lib directory

④ ./hostapd -B hostapd.conf // Create hotspot

⑤ udhcpd udhcpd.conf // Configure dhcp server

⑥ Connected to generated TEST_wifi with cellphone and ping 192.168.1.20



TEST Wifi

Automatic connection

State information: Connected but unable to access the Internet

Connection speed: 54Mbps

signal strength: Excellent

Security: WPA2-Personal

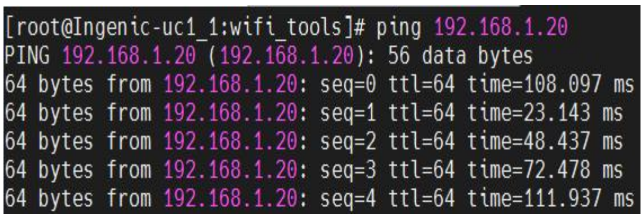
IP: 192.168.1.20 fe80

Subnet mask: 255.255.255.0

Router: 0.0.0.0

Agent: None

IP Settings: DHCP



```
[root@Ingenic-uc1_1:wifi_tools]# ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20): 56 data bytes
64 bytes from 192.168.1.20: seq=0 ttl=64 time=108.097 ms
64 bytes from 192.168.1.20: seq=1 ttl=64 time=23.143 ms
64 bytes from 192.168.1.20: seq=2 ttl=64 time=48.437 ms
64 bytes from 192.168.1.20: seq=3 ttl=64 time=72.478 ms
64 bytes from 192.168.1.20: seq=4 ttl=64 time=111.937 ms
```

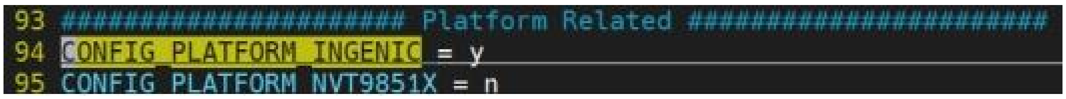
3.3 USB WiFi Driver Porting

USB WIFI and SDIO WIFI are only different in underlying interface, the tools and methods used in testing STA mode and AP mode are the same. Therefore, this part only explains how to transplant the driver, not the specific test process.

The following use rTL8188FTV to explain the driver transplant process:

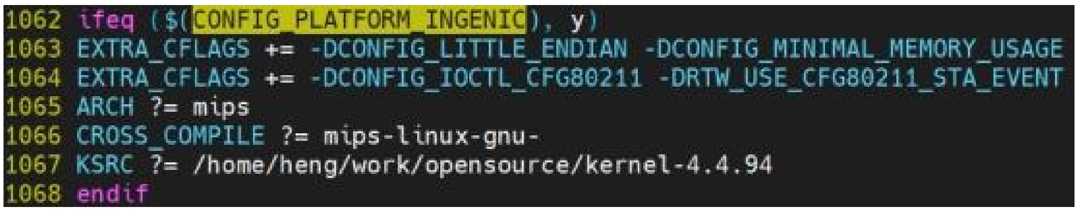
（1）Obtain rTL8188FTV driver source code，rtl8188FU_linux_v5.2.11.1_22663.20170607

（2）Edit Makefile, add Ingenic platform



```
######################## Platform Related ########################
CONFIG_PLATFORM_INGENIC = y
CONFIG_PLATFORM_NVT9851X = n
```

（3）Add a macro to Makefile about Ingenic support



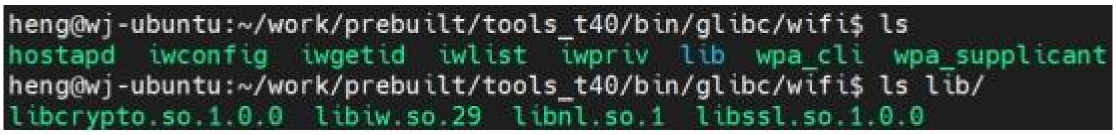
```
ifeq ($(CONFIG_PLATFORM_INGENIC), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_MINIMAL_MEMORY_USAGE
EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
ARCH ?= mips
CROSS_COMPILE ?= mips-linux-gnu-
KSRC ?= /home/heng/work/opensource/kernel-4.4.94
endif
```

Compiling the code after porting, driver file 8188fu.ko will be generated.

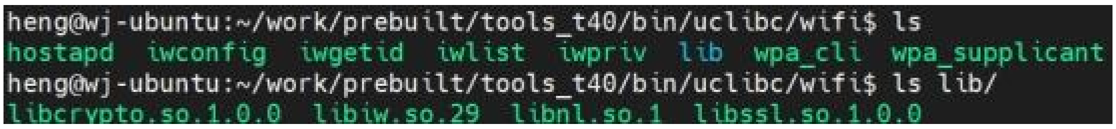
3.4 USB WiFi Function Test

The specific network function test is the same as SDIO WiFi, which will not be described here.

The SDK already includes STA mode and AP mode related programs and lib libraries generated by 720 compiler. Which is under prebuilt/tools_t40/bin/uclibc/wifi 和 prebuilt/tools_t40/bin/glibc/wifi directory.



```
heng@wj-ubuntu:~/work/prebuilt/tools_t40/bin/glibc/wifi$ ls
hostapd  iwconfig  iwgetid  iwlist  iwpriv  lib  wpa_cli  wpa_supplicant
heng@wj-ubuntu:~/work/prebuilt/tools_t40/bin/glibc/wifi$ ls lib/
libcrypto.so.1.0.0  libiw.so.29  libnl.so.1  libssl.so.1.0.0
```



```
heng@wj-ubuntu:~/work/prebuilt/tools_t40/bin/uclibc/wifi$ ls
hostapd  iwconfig  iwgetid  iwlist  iwpriv  lib  wpa_cli  wpa_supplicant
heng@wj-ubuntu:~/work/prebuilt/tools_t40/bin/uclibc/wifi$ ls lib/
libcrypto.so.1.0.0  libiw.so.29  libnl.so.1  libssl.so.1.0.0
```

4 WiFi FQA Analysis

4.1 Incorrect configuration of GPIO Function corresponding to SDIO

There are probably two reasons for this problem:

A：The kernel itself is incorrectly configured. Please refer to the hardware circuit design to see which group of GPIO is used.

B: Other drivers have modified the Function of GPIO Function corresponding to SDIO. For example, when one of our customers was running IPC program, the communication of SDIO WIFI network was abnormal. Finally, the reason was that the driver of sensor modified the Function configuration corresponding to SDIO GPIO.

Here's how to troubleshoot this error:

Here, the PA port of T31 is used as an example. Configure the PA08, PA09, PA10, PA11, PA16, and PA17 pins of the PA port as FUNCTION3, which is the MSC1 function.

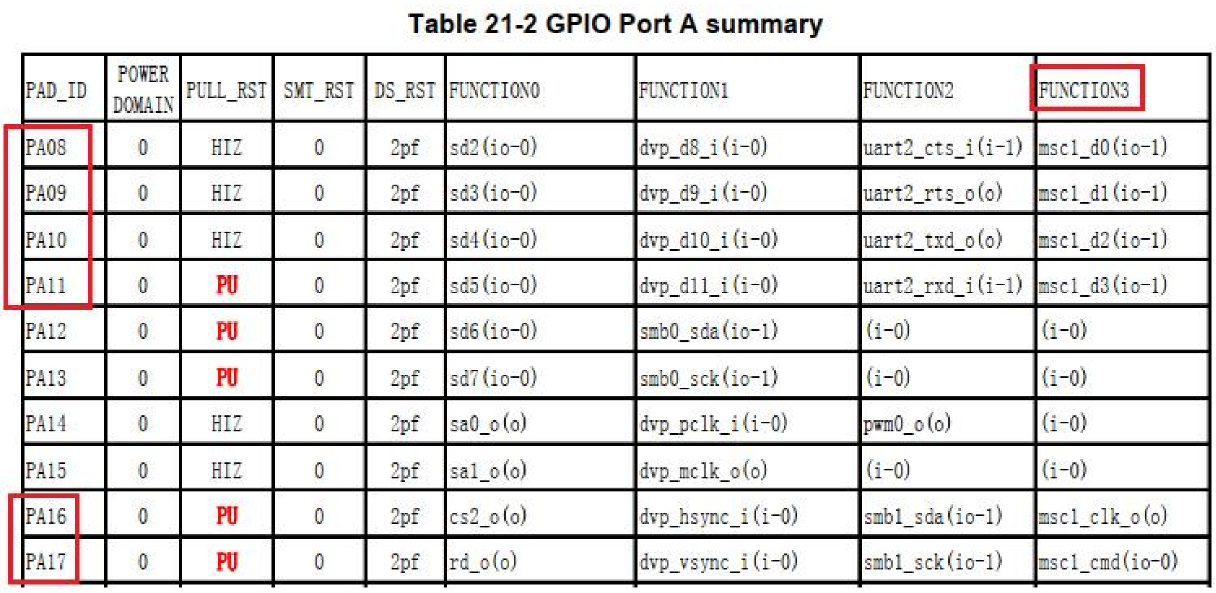


Table 21-2 GPIO Port A summary

PAD_ID	POWER DOMAIN	PULL_RST	SMT_RST	DS_RST	FUNCTION0	FUNCTION1	FUNCTION2	FUNCTION3
PA08	0	HIZ	0	2pf	sd2(io-0)	dvp_d8_i(i-0)	uart2_cts_i(i-1)	msc1_d0(io-1)
PA09	0	HIZ	0	2pf	sd3(io-0)	dvp_d9_i(i-0)	uart2_rts_o(o)	msc1_d1(io-1)
PA10	0	HIZ	0	2pf	sd4(io-0)	dvp_d10_i(i-0)	uart2_txd_o(o)	msc1_d2(io-1)
PA11	0	PU	0	2pf	sd5(io-0)	dvp_d11_i(i-0)	uart2_rxd_i(i-1)	msc1_d3(io-1)
PA12	0	PU	0	2pf	sd6(io-0)	smb0_sda(io-1)	(i-0)	(i-0)
PA13	0	PU	0	2pf	sd7(io-0)	smb0_sck(io-1)	(i-0)	(i-0)
PA14	0	HIZ	0	2pf	sa0_o(o)	dvp_pclk_i(i-0)	pwm0_o(o)	(i-0)
PA15	0	HIZ	0	2pf	sa1_o(o)	dvp_mclk_o(o)	(i-0)	(i-0)
PA16	0	PU	0	2pf	cs2_o(o)	dvp_hsync_i(i-0)	smb1_sda(io-1)	msc1_clk_o(o)
PA17	0	PU	0	2pf	rd_o(o)	dvp_vsync_i(i-0)	smb1_sck(io-1)	msc1_cmd(io-0)

The addresses base of each group of GPIO of T31 are as follows:

Name	Base	Description
PA_BASE	0x10010000	Address base of GPIO Port A
PB_BASE	0x10011000	Address base of GPIO Port B
PC_BASE	0x10012000	Address base of GPIO Port C

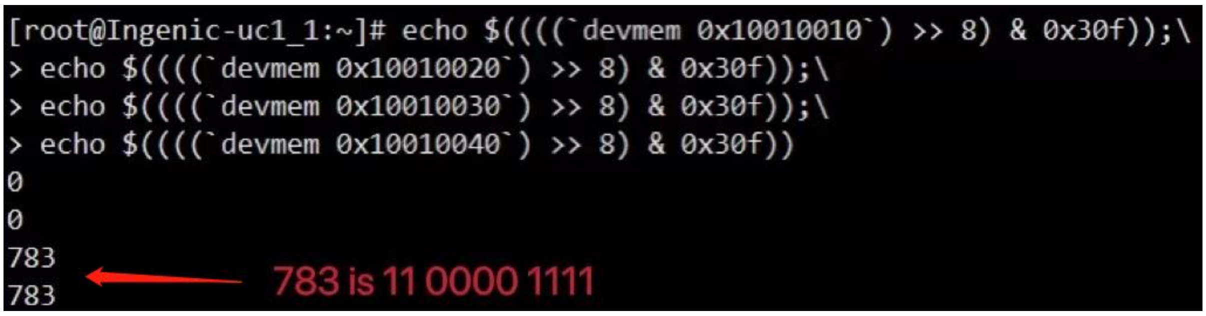
We only need to query four registers to determine the function of GPIO:

PORTx Interrupt Registers	Offset 0x10
PORTx Mask Registers	Offset 0x20
PORTx PAT1/Direction Registers	Offset 0x30
PORTx PAT0/Data Registers	Offset 0x40

We use the following command to query the function of PA8, PA9, PA10, PA11, PA 16, PA17.

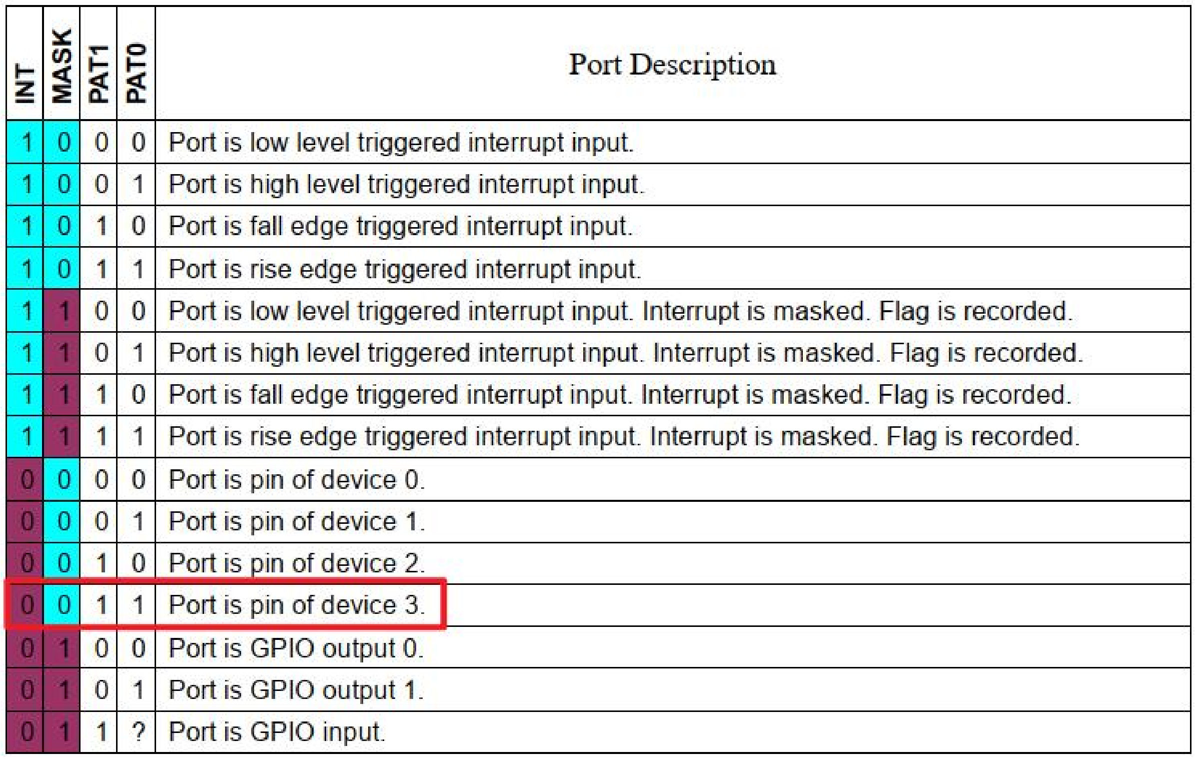
```
echo $(((`devmem 0x10010010`) >> 8) & 0x30f));\
echo $(((`devmem 0x10010020`) >> 8) & 0x30f));\
echo $(((`devmem 0x10010030`) >> 8) & 0x30f));\
echo $(((`devmem 0x10010040`) >> 8) & 0x30f))
```

First, move the read register value 8 bits to the right, since it starts from PA08. A nd then follow 0x30F for the & operation. 0x30F corresponds to PA08, PA09, PA10, PA11, PA16, PA17. If it is another GPIO pin, modify it here.



```
[root@Ingenic-uc1_1:~]# echo $(((`devmem 0x10010010`) >> 8) & 0x30f));\
> echo $(((`devmem 0x10010020`) >> 8) & 0x30f));\
> echo $(((`devmem 0x10010030`) >> 8) & 0x30f));\
> echo $(((`devmem 0x10010040`) >> 8) & 0x30f))
0
0
783
783
```

The values of the four registers corresponding to these six pins are 0, 0, 1, 1. It c an be seen from the table below that the current six GPIO pins are configured in FUNCTION3 mode, namely mSC1 function.



INT	MASK	PAT1	PAT0	Port Description
1	0	0	0	Port is low level triggered interrupt input.
1	0	0	1	Port is high level triggered interrupt input.
1	0	1	0	Port is fall edge triggered interrupt input.
1	0	1	1	Port is rise edge triggered interrupt input.
1	1	0	0	Port is low level triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	0	1	Port is high level triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	1	0	Port is fall edge triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	1	1	Port is rise edge triggered interrupt input. Interrupt is masked. Flag is recorded.
0	0	0	0	Port is pin of device 0.
0	0	0	1	Port is pin of device 1.
0	0	1	0	Port is pin of device 2.
0	0	1	1	Port is pin of device 3.
0	1	0	0	Port is GPIO output 0.
0	1	0	1	Port is GPIO output 1.
0	1	1	?	Port is GPIO input.