



Perfect Wireless Experience
完美无线体验

H3 Family System Driver Integration and Application Guidance

Document version: V3.0.0

Update date: 20.04.2016



Applicability Table

No.	Product model	Description
1	H3-Family	NA

Copyright

Copyright ©2015Fibocom Wireless Inc. All rights reserved.

Without the prior written permission of the copyright holder, any company or individual is prohibited to excerpt, copy any part of or the entire document, or distribute the document in any form.

Notice

The document is subject to update from time to time owing to the product version upgrade or other reasons. Unless otherwise specified, the document only serves as the user guide. All the statements, information and suggestions contained in the document do not constitute any explicit or implicit guarantee.

Trademark



The trademark is registered and owned by Fibocom Wireless Inc.

Version Record

Version	Update Date	Remarks
V1.0.0	10-17-2015	Initial version
V2.0.0	11-24-2015	Optimize document name and title; Optimize document structure and content; Optimize document figure;
V3.0.0	20-04-2016	Added device_init_wakeup() in chapter 2.2

Contents

1 Introduction.....	5
1.1 Purpose.....	5
1.2 Scope.....	5
2 Instructions for Linux system.....	6
2.1 Linux Kernel Device Driver Architecture.....	6
2.2 Linux ACM Driver Integration.....	7
2.2.1 ACM Driver Code Modification.....	7
2.2.2 Detailed Configuration Setup.....	10
2.3 ACM Driver Configuration Confirmation.....	11
2.4 Port Form Description.....	12
2.5 Port Testing.....	12
2.5.1 Command Line Testing.....	12
2.5.2 Program Testing.....	13
2.6 Connect Internet via PPP Dial-up.....	15
2.7 PPP Dial-up Script Description.....	17
3 Android System Instructions.....	20
3.1 Android Kernel Driver Porting and Loading.....	20
3.2 System Integration and Debugging.....	20
3.2.1 Communication between System and Module Ports.....	20
3.2.2 RIL Integration.....	20
3.2.3 adb Tool Installation.....	21
3.2.4 RIL Library Replacing.....	22
3.2.4.1 Check for RIL Driver Loading.....	23
3.2.4.2 Check for RIL Version Number.....	23
3.2.5 Debug for Signal Quality, Telephone, SMS.....	24
3.2.5.1 Query for Module IMEI.....	24
3.2.5.2 Signal Strength Query.....	24
3.2.5.3 Voice and SMS Detection.....	25
3.2.5.4 Enabling Data Network.....	26
3.2.5.5 Preferred Network.....	26
3.2.5.6 Debug Audio Channel Switch and Volume Adjustment.....	26

1 Introduction

1.1 Purpose

- This article is the guidance for driver integration development activities for H3 series 3G module devices based on Android/Linux systems. This document is mainly for driver developers for product developers based on the above systems.

1.2 Scope

The document applies to the following:

- Android 4.0 and higher version.
- Linux2.6.22 and higher version.

2 Instructions for Linux system

2.1 Linux Kernel Device Driver Architecture

Linux kernel will load the USB driver according to the USB device interfaces reported by the 3G module. After the correct driver is loaded, the module can begin to work.

Linux kernel driver architecture of the Linux system is shown in Figure 2-1:

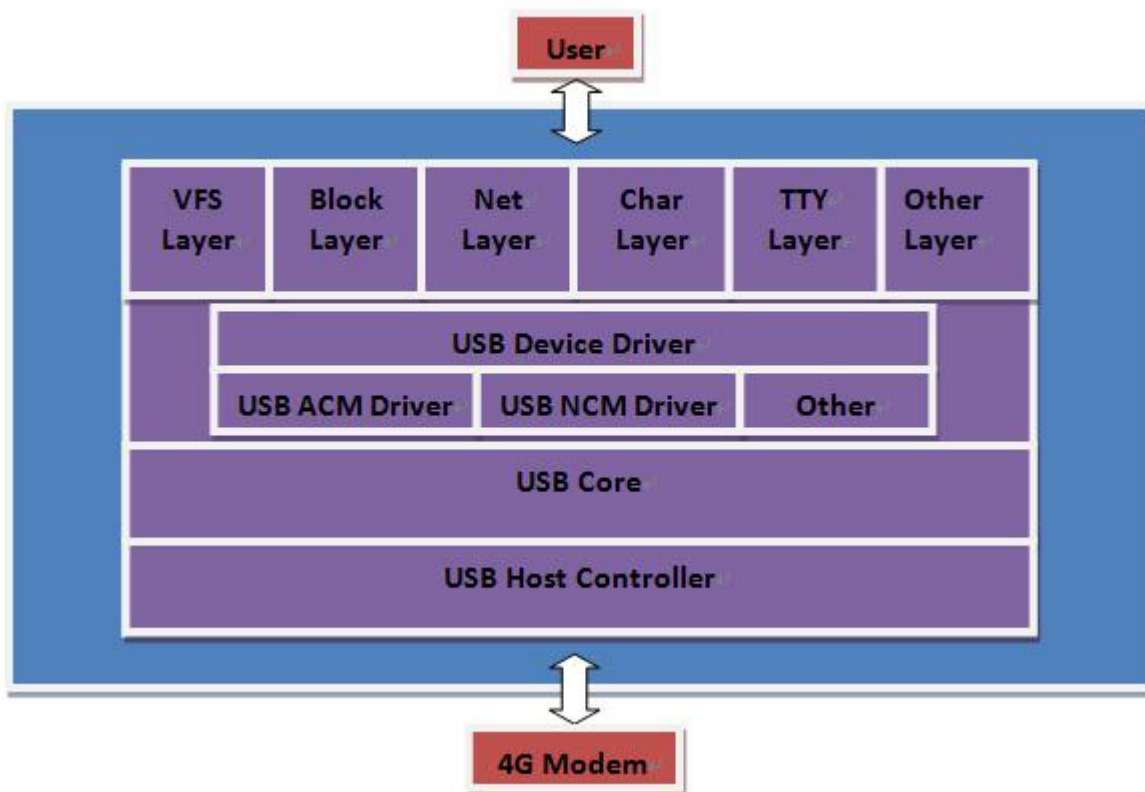


Figure 2-1 Driver Architecture

As is shown in Figure 2-1, driver modules related to 3G devices in the USB driver architecture of Linux system are: USB ACM driver modules.



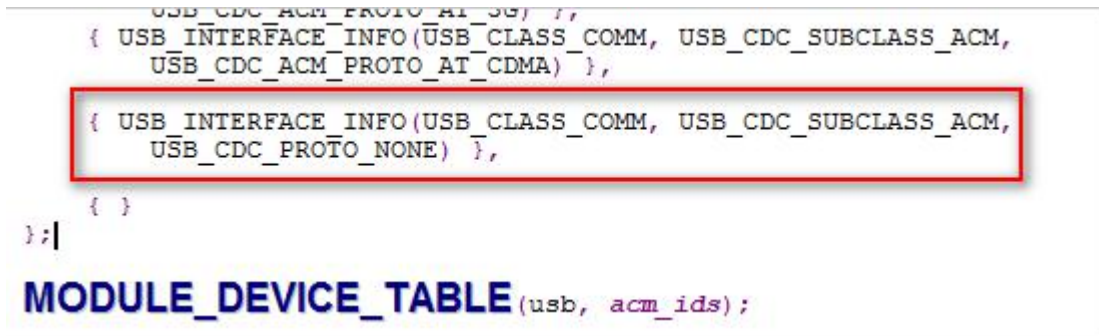
Attention:

ACM Driver: USB ACM driver supports modem ports, AT ports and so on; the source code (cdc-acm.c) of the ACM driver is built-in in the Linux kernel.

2.2 Linux ACM Driver Integration

2.2.1 ACM Driver Code Modification

1. Modification for driver code: As shown in Figure 2-2, add the codes in the red box to array “static const struct usb_device_id acm_ids[]” in file “drivers/usb/class/cdc-acm.c” :



```

USB_CDC_ACM_PROTO_AT_CDMA) },
{ USB_INTERFACE_INFO(USB_CLASS_COMM, USB_CDC_SUBCLASS_ACM,
  USB_CDC_ACM_PROTO_AT_CDMA) },
{ USB_INTERFACE_INFO(USB_CLASS_COMM, USB_CDC_SUBCLASS_ACM,
  USB_CDC_PROTO_NONE) },
{ }
};

MODULE_DEVICE_TABLE(usb, acm_ids);

```

Figure 2-2 Codes of acm_ids

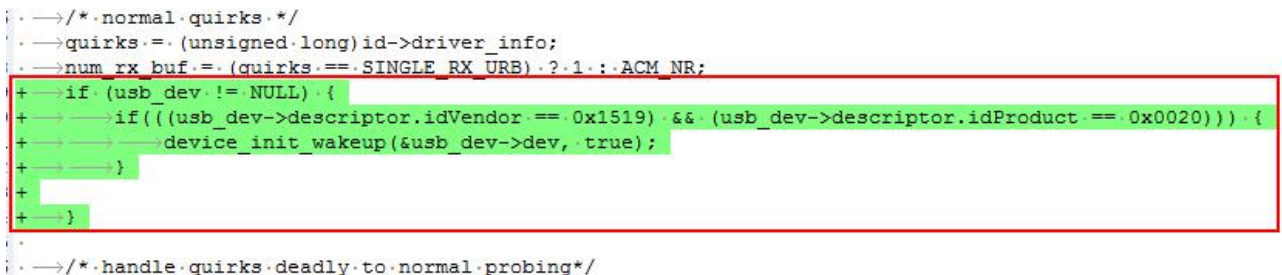
Codes in Figure 2-2 are as follows:

```

{ USB_INTERFACE_INFO(USB_CLASS_COMM, USB_CDC_SUBCLASS_ACM,
  USB_CDC_PROTO_NONE) },

```

2. Modification for driver code: As shown in following Figure, add the codes in the red box to function “static int acm_probe” in files “drivers/usb/class/cdc-acm.c”:



```

/* normal quirks */
quirks = (unsigned long) id->driver_info;
num_rx_buf = (quirks == SINGLE_RX_URB) ? 1 : ACM_NR;
if (usb_dev != NULL) {
    if(((usb_dev->descriptor.idVendor == 0x1519) && (usb_dev->descriptor.idProduct == 0x0020))) {
        device_init_wakeup(&usb_dev->dev, true);
    }
}
/* handle quirks deadly to normal probing */

```

Codes are as follows:

```

if (usb_dev != NULL) {
    if(((usb_dev->descriptor.idVendor == 0x1519) && (usb_dev->descriptor.idProduct == 0x0020))) {
        device_init_wakeup(&usb_dev->dev, true);
    }
}

```

3. Modification for driver code: As shown in following Figure, add the codes in the red box to function “acm_tty_write” in files “drivers/usb/class/cdc-acm.c”:

```

@@ -711,7 +711,7 @@ static int acm_tty_write(struct tty_struct *tty,
    unsigned long flags;
    int wbn;
    struct acm_wb *wb;

+   struct usb_host_endpoint *ep;
    dbg("Entering acm_tty_write to write %d bytes,", count);

    if (!ACM_READY(acm))
@@ -731,6 +731,13 @@ static int acm_tty_write(struct tty_struct *tty,
    dbg("Get %d bytes...", count);
    memcpy(wb->buf, buf, count);
    wb->len = count;

+   if((0x1519 == acm->dev->descriptor.idVendor) && (0x0110 != acm->dev->descriptor.bcdUSB)) {
+       ep = usb_pipe_endpoint(wb->urb->dev, wb->urb->pipe);
+       if (ep && (0 != wb->urb->transfer_buffer_length)
+           && (0 == wb->urb->transfer_buffer_length % ep->desc.wMaxPacketSize)) {
+           wb->urb->transfer_flags |= URB_ZERO_PACKET;
+       }
+   }
    spin_unlock_irqrestore(&acm->write_lock, flags);

```

Codes are as follows:

```
struct usb_host_endpoint *ep;
```

```

if((0x1519 == acm->dev->descriptor.idVendor) && (0x0110 != acm->dev->descriptor.bcdUSB)) {

    ep = usb_pipe_endpoint(wb->urb->dev, wb->urb->pipe);

    if (ep && (0 != wb->urb->transfer_buffer_length)

        && (0 == wb->urb->transfer_buffer_length % ep->desc.wMaxPacketSize)) {

        wb->urb->transfer_flags |= URB_ZERO_PACKET;

    }

}

```

4. Header file modification

First to see “usb_pipe_endpoint” function is defined or not in file “include/linux/usb.h”, If defined, it is no need to modify, If not, following steps to modify.

File: include/linux/usb.h

Add Function: usb_pipe_endpoint


```

@@ -1552,6 +1552,13 @@ static inline unsigned int __create_pipe(struct usb_device *dev,
    ((PIPE_INTERRUPT <= 30) | __create_pipe(dev, endpoint) | USB_DIR_IN)

    /*-----*/
+static inline struct usb_host_endpoint *
+usb_pipe_endpoint(struct usb_device *dev, unsigned int pipe)
+{
+    struct usb_host_endpoint **eps;
+    eps = usb_pipein(pipe) ? dev->ep_in : dev->ep_out;
+    return eps[usb_pipeendpoint(pipe)];
+}

static inline __u16
usb_maxpacket(struct usb_device *udev, int pipe, int is_out)

```

Codes are as follows:

```

static inline struct usb_host_endpoint *

usb_pipe_endpoint(struct usb_device *dev, unsigned int pipe)

{

    struct usb_host_endpoint **eps;

    eps = usb_pipein(pipe) ? dev->ep_in : dev->ep_out;

    return eps[usb_pipeendpoint(pipe)];

}

```

5. Modify the compiled configuration of kernel (config files under the kernel directory) and ensure the following configuration items have been selected:

1) Related configuration items of PPP dial-up:

```

CONFIG_PPP=y
CONFIG_PPP_MULTILINK=y
CONFIG_PPP_FILTER=y
CONFIG_PPP_ASYNC=y
CONFIG_PPP_SYNC_TTY=y
CONFIG_PPP_DEFLATE=y
CONFIG_PPP_BSDCOMP=y

```

2) Related configuration items of USB ACM:

```

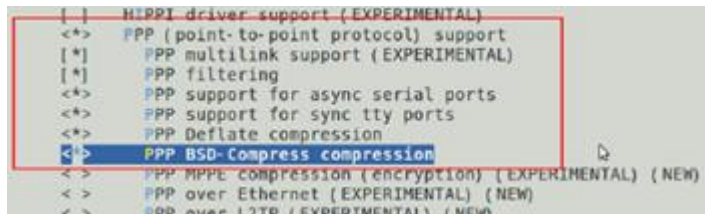
CONFIG_USB_ANNOUNCE_NEW_DEVICES=y (if such option exists, it's suggested to
configure; if not, please ignore)
CONFIG_USB_ACM=y

```

2.2.2 Detailed Configuration Setup

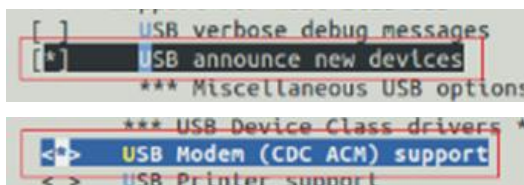
1. Open the Terminal tool, enter the kernel directory (it is assumed to be "/linux-3.0.8/ /home/ght"), and execute the <configuration> make command (it's assumed to use standard menuconfig).
2. Complete configurations of PPP dial-up as the following guidelines:

Enter **"Device Drivers"→"Network device support"** menu and select all items in the red border



3. Complete configurations of ACM driver as the following guidelines:

Enter **"Device Drivers"→"USB support"** menu and select USB announce new devices and USB Modem (CDC ACM) support items:



4. After the configuration, exit the configuration interface step by step by selecting "<Exit>". And then select "<Yes>" and exit the save interface.
5. After completing configurations, run the make command to edit the modified kernel.

2.3 ACM Driver Configuration Confirmation

When the system starts up, execute the `dmesg` command and check the kernel messages. The information as shown in the red box in Figure 2-3 indicate that the ACM driver in the system has been successfully configured.

```
[1193149.843866] usbcore: registered new interface driver cdc_acm
[1193149.843867] cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters
```

Figure 2-3 ACM Configuration

After the system is started and the 3G module is powered up, execute the `dmesg` command to check the kernel messages; the information as shown in the red box of in Figure 2-4 indicate that ACM driver has been successfully loaded. When loading drivers, the device nodes can be verified as shown in the red box in Figure 2-4. The ACM device nodes are `ttyACM0 ~ ttyACMx`.

```
usb 1-1.2: new high-speed USB device number 75 using ehci_hcd
usb 1-1.2: New USB device found, idVendor=1519, idProduct=0020
usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1.2: Product: 7 CDCs
usb 1-1.2: Manufacturer: Comneon
usb 1-1.2: SerialNumber: 003580023000167
cdc_acm 1-1.2:1.0: This device cannot do calls on its own. It is not a modem.
cdc_acm 1-1.2:1.0: ttyACM0: USB ACM device
cdc_acm 1-1.2:1.2: This device cannot do calls on its own. It is not a modem.
cdc_acm 1-1.2:1.2: ttyACM1: USB ACM device
cdc_acm 1-1.2:1.4: This device cannot do calls on its own. It is not a modem.
cdc_acm 1-1.2:1.4: ttyACM2: USB ACM device
cdc_acm 1-1.2:1.6: This device cannot do calls on its own. It is not a modem.
cdc_acm 1-1.2:1.6: ttyACM3: USB ACM device
cdc_acm 1-1.2:1.8: This device cannot do calls on its own. It is not a modem.
cdc_acm 1-1.2:1.8: ttyACM4: USB ACM device
cdc_acm 1-1.2:1.10: This device cannot do calls on its own. It is not a modem.
cdc_acm 1-1.2:1.10: ttyACM5: USB ACM device
cdc_acm 1-1.2:1.12: This device cannot do calls on its own. It is not a modem.
cdc_acm 1-1.2:1.12: ttyACM6: USB ACM device
```

Figure 2-4 Driver Loading

Execute `ls -al /dev/ttyACM*` command to inquire `ttyACM0`, `ttyACM1` and `ttyACMx`.

```
/ # ls /dev/ttyACM*
/dev/ttyACM0
/dev/ttyACM1
/dev/ttyACM2
```

2.4 Port Form Description

No.	Port name	Port form	Remarks
1	ttyACM0	Modem Port	For PPP data traffic, or for sending and receiving AT command under the non-data mode
2	ttyACM1	Trace Port	For capturing module debug information
3	ttyACM2	3G-trace	For Artemis Trace Tool to capture 3G-trace
4	ttyACM3 ttyACM4 ttyACM5	AT Port	For AT communications, namely, sending and receiving AT commands
5	ttyACM6	OCT-trace	For IRTC Tool to capturing OCT-trace

2.5 Port Testing

2.5.1 Command Line Testing

1. Open the terminal.
2. Execute `echo -e "ate0\r\n" > /dev/ttyACM3` (Execute this command before any other command or the cat command will be abnormal.)
3. Execute `echo -e "at+cgmr\r\n" > /dev/ttyACM3` to inquire the software version.
4. Execute `cat /dev/ttyACM3` to read the result.
5. Executer `echo -e "at+cgdcont=1,\"ip\", \"3gnet\" \r\n" > /dev/ttyACM3` to set up APN.
6. Execute `cat /dev/ttyACM3` to read settings and return result

If AT command contains double quotation marks, ESC" should be added, please refer to Step 5 for the format.

```

ght@fibocom:~$ sudo chmod 777 /dev/ttyACM*
ght@fibocom:~$ echo -e "ate0\r\n" > /dev/ttyACM3
ght@fibocom:~$ echo -e "at+cgmr\r\n" > /dev/ttyACM3
ght@fibocom:~$ cat /dev/ttyACM3 &
[1] 30743
ght@fibocom:~$
"+CGMR: "H330S_V2H.00.20_EXT6"

OK

ght@fibocom:~$ echo -e "at+cgdcont=1,\"ip\", \"3gnet\" \r\n" > /dev/ttyACM3
ght@fibocom:~$
OK

```

2.5.2 Program Testing

The C program below can be used to test send and receive of AT commands. The program opens the /dev/ttyACM3 device node, and calls the write and read function to send AT commands and receive the reply.

```
#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <errno.h>

#include <termios.h>

#define ATPORT "/dev/ttyACM3"

#define BUFSIZE 1000

#define BAUDRATE B115200

int open_port(char *port)

{

    struct termios options;

    int fd;

    fd = open(port, O_RDWR | O_NOCTTY | O_NDELAY);

    if (fd == -1) {

        printf("%s: Unable to open the port - \r\n", __func__);

    } else {

        fcntl(fd, F_SETFL, FNDELAY);

        tcgetattr(fd, &options );

        cfsetispeed(&options, BAUDRATE );

        cfsetospeed(&options, BAUDRATE );

        options.c_cflag |= ( CLOCAL | CREAD);

        options.c_cflag&= ~(CSIZE | PARENB | CSTOPB | CSIZE);
```

Reproduction forbidden without Fibocom Wireless Inc. written authorization - All Rights Reserved.

```

options.c_cflag |= CS8;

options.c_cflag&= ~CRTSCTS;

options.c_lflag&= ~(ICANON | ECHO | ECHOE | ISIG);

options.c_iflag&= ~(IXON | IXOFF | IXANY | ICRNL | INLCR | IGNCR);

options.c_oflag&= ~OPOST;

if ( tcsetattr( fd, TCSANOW, &options ) == -1 ) {

    printf ("Error with tcsetattr = %s\r\n", strerror ( errno ) );

} else {

    printf ( "Open port succeed\r\n");

}

}

return (fd);

}

int main()

{

    int fd = open_port(ATPORT);

    char at_cmd_ch[50]="AT+CGMR\r\n";

    char buf[BUFSIZE];

    memset(buf,0,BUFSIZE);

    printf("AtSend: %s\r\n", at_cmd_ch);

    write(fd, at_cmd_ch , strlen(at_cmd_ch));

    sleep(1);

    read(fd, buf, BUFSIZE );

    printf("AtRecevie: %s\r\n", buf);

    close(fd);

    return 0;

}

```

Save the above code in the TestPort.c text, execute the `o - TestPortTestPort.c GCC` command to compile the TestPort program, and then execute the compiled program to see the returned results.

```
ght@fibocom:~$ ./TestPort
Open port succeed
AtSend: AT+CGMR

AtRecevie:
+SIM READY
AT+CGMR
+CGMR: "H3305_V2H.00.20_EXT6"

OK
ght@fibocom:~$
```

Because the 3G module needs time to process after sending out the "AT+CGMR" command, it is necessary to delay at least 500ms before reading. Sleep (1) in the demo code is only for reference.



Attention:

sleep(1) means delaying 1 second.

2.6 Connect Internet via PPP Dial-up

In application scenarios where NCM driver can't be supported, it's necessary to use PPP dial-up.

There are a total of three script files for PPP dial-up: chat-wcdma-connect, chat-wcdma-disconnect and wcdma, and the content for scripts is as shown in [2.7](#).

1. Put the above three script files in the `/etc/ppp/peers/` directory, and use `Chmod 777 XXX` command to give the file read and execute permissions. Input the following in the command line:
PPPD call <dial-up script >

For example, if file name of the dial-up script is "wcdma", the command is as follows:

```
pppd call wcdma
```

2. After successful dial-up, execute the `ifconfig` command to inquire IP address.

Figure 2-5 shows the query results after executing `ifconfig` command after successful ppp dial-up.


```
[root@wavelet peers]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:19:D1:75:1F:3A
          inet6 addr: fe80::219:d1ff:fe75:1f3a/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:147400 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29822 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:52099010 (49.6 MiB)  TX bytes:3672236 (3.5 MiB)
          Interrupt:21 Memory:dfde0000-dfe00000

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:16436  Metric:1
          RX packets:70 errors:0 dropped:0 overruns:0 frame:0
          TX packets:70 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7024 (6.8 KiB)  TX bytes:7024 (6.8 KiB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:172.20.19.220  P-t-P:172.20.19.220  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1280  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:58 (58.0 b)  TX bytes:108 (108.0 b)
```

Figure 2-5 ifconfig query



Attention:

Precondition for dial-up:

- A valid SIM card is inserted.
- Module is powered up and running.
- Module can register network.

2.7 PPP Dial-up Script Description

Example of wcdma script:

```
nodetach  
lock  
/dev/ttyACM0  
115200  
crtsets  
debug  
#logfile /data/logfile  
modem  
hide-password  
usepeerdns  
noauth  
noipdefault  
novj  
novjccomp  
noccp  
defaultroute  
ipcp-accept-local  
ipcp-accept-remote  
connect 'chat -s -v -f /etc/ppp/peers/chat-wcdma-connect'  
disconnect 'chat -s -v -f /etc/ppp/peers/chat-wcdma-disconnect'
```



Attention:

/dev/ttyACM0 assigns the port for dial-up; if it's necessary to use ACM3 port for dial-up, just modify ttyACM0 into ttyACM3.

Example of chat-wcdma-connect script:

```
" AT
OK "
ABORT 'NO CARRIER'
ABORT 'ERROR'
ABORT 'NO DIALTONE'
ABORT 'BUSY'
ABORT 'NO ANSWER'
" AT
OK ATZ
OK AT+GTRAT?
OK AT+CMEE=2
OK AT+CSQ
OK AT+CPIN?
OK AT+COPS?
OK AT+CGACT=0,1
OK AT+CGDCONT=1,\"IP\",\"cmnet\",,0,0
OK ATDT*99#
CONNECT "
```



Attention:

AT+CGDCONT=1,\"IP\",\"cmnet\",,0,0 (cmnet represents APN for China Mobile and APN for China Unicom is 3gnet)

Example of chat-wcdma-disconnect script:

```
ABORT OK
ABORT BUSY
ABORT DELAYED
ABORT "NO ANSWER"
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT VOICE
ABORT ERROR
ABORT RINGING
TIMEOUT 12
"" \K
"" \K
"" \K
"" +++ATH
"" +++ATH
"" +++ATH
"" ATZ
SAY "\nGoodbay\n"
```

3 Android System Instructions

3.1 Android Kernel Driver Porting and Loading

Android kernel is based on the Linux kernel; as for the configuration of Android kernel, please refer to sections [2.2](#), [2.3](#) and [2.4](#).

3.2 System Integration and Debugging

3.2.1 Communication between System and Module Ports

The communication function of Android system is realized by the interaction of AT commands between RIL and the module. The system shall have the hardware interface for data communication, such as USB and UART. If the system uses the USB port to send and receive AT command, the corresponding USB driver shall be loaded in the Android system kernel; please refer to Section [3.1](#) for detailed upload methods.

RIL (Interface Layer Radio) abstracts the various functions used by users to corresponding requests and unsolicited commands, which are eventually converted into 3GPP standard AT commands and sent to the 3G module, and the processing results will be returned to UI.

3.2.2 RIL Integration

1. Modify ril-daemon service in init.rc files:

```
#begin
serviceril-daemon /system/bin/rild -l /system/lib/libreference-ril.so -- -d /dev/ttyACM3
class main
socketrild stream 660 root radio
socketrild-debug stream 660 radio system
user root
group radio cache inetmisc audio sdcard_rw log
#end
```

The parameter after `-d` is the actual port mapped by USB (AT communication port). In general, `ttyACM2` is the AT communication port for 4G module.

2. Modify `android/hardware/ril/rild/rild.c` file, and mark the `switchUser()` function call.

```

1
OpenLib:
#endif
// switchUser();

dlHandle = dlopen(rilLibPath, RTLD_NOW);

```

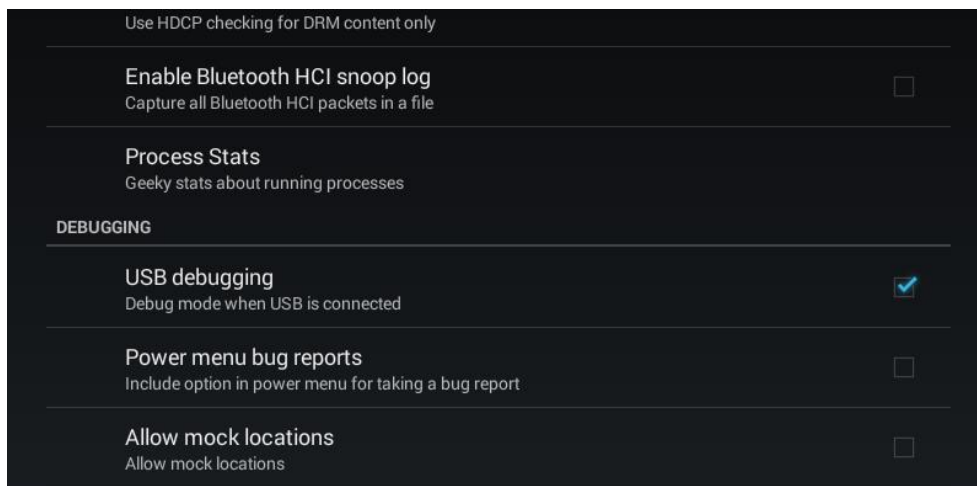
During compiling, pack libreference-ril.so provided by Fibocom into the system mirror image directory /system/lib/, and then burn the new system image.

3.2.3 adb Tool Installation

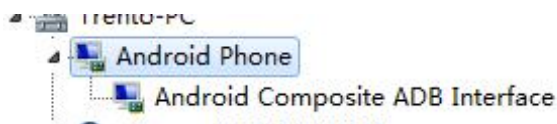
To see if the RIL is normally loaded, it's necessary to see RIL operation LOG using adb tool.

1. adb tool installation and instructions

After starting up, check “developer option”->>“USB debugging” option in the Android system settings to use adb debugging features.



Connect Android device to PC, and install adb driver (which can be installed via the commonly used Android mobile assistant tool). After successful installation, an adb device is found in the device management.



Open a cmd window in the windows system, enter the directory of adb tool and input “adb shell” command to enter the command line terminal of Android equipment for all kinds of debugging work.

3.2.4 RIL Library Replacing

In the debugging process, it's necessary to update the RIL library from time to time, and the following steps can be performed in order to update the RIL library.

1. adb devices --- inquire whether the device is identified by adb tool

```
C:\Users\Administrator>adb devices
List of devices attached
0123456789ABCDEF      device
```

2. adb root --- switch to root for adb terminal

```
C:\Users\Administrator>adb root
adbd is already running as root
```

3. adb remount --- give /system/directory read-write permission

```
C:\Users\Administrator>adb remount
remount succeeded
```

4. adb shell ls init*.rc --- inquire how many init.xx.rc files are in android system

```
C:\Users\Administrator>adb shell ls init*.rc
E:\intel-update\Intel_update>adb shell ls init*.rc
init.am335xevm.rc
init.am335xevm.usb.rc
init.goldfish.rc
init.rc
init.trace.rc
init.usb.rc
```

5. adb shell cat init.xx.rc – inquire the init.xx.rc file one by one, until the /system/bin/rild keyword is found; use the key word rild to search the RIL library name in use: lib reference-ril.so (Intel platform rild keyword is generally in the init.modem.rc file)

```
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so -- -d /dev/ttyACM2
class main
socket rild stream 660 root radio
socket rild-debug stream 660 radio system
user root
group radio cache inet misc audio sdcard_rw log
```

6. adb shell push xxx\libght-ril.so /system/lib/libreference-ril.so --- replace ril library

```
C:\Users\Administrator>adb push E:\libreference-ril.so /system/lib/libreference-ril.so
```



Attention:

The first parameter of push is the ril library path in the local directory of the computer, and the second parameter is the ril library path used in android system. The ril library name given from the second parameter must be the same as the ril library name inquired in Step 5, or ril can't run normally.

7. adb shell stop ril-daemon ---suspend ril service
8. adb shell logcat -b radio -c --- clear up ril log
9. adb shell start ril-daemon --- start up ril service
10. adb shell logcat -b radio -v time > D:/radio.txt --- capture ril log and confirm version number

Use ctrl+c to terminate log capturing, and open radio.txt files under the root directory in D:\, then search for key words RIL Daemon version to inquire the RIL version number and confirm whether RIL library is updated successfully.

```
C:\Users\Administrator>adb shell stop ril-daemon
C:\Users\Administrator>adb shell logcat -b radio -c
C:\Users\Administrator>adb shell start ril-daemon
C:\Users\Administrator>adb shell logcat -b radio -v time > D:/radio.txt
^C
C:\Users\Administrator>
```

3.2.4.1 Check for RIL Driver Loading

Open adb shell and input logcat - b radio; check the AT command interaction through the output log, and the following figure is an example of log with normal RIL initialization:

```
D/AT      < 1484>: AT> ATE0Q0U1
D/AT      < 1484>: AT< ATE0Q0U1
D/AT      < 1484>: AT< OK
D/AT      < 1484>: AT> ATE0Q0U1
D/AT      < 1484>: AT< OK
D/AT      < 1484>: AT> AT$0=0
D/AT      < 1484>: AT< OK
D/AT      < 1484>: AT> AT+CMEE=1
D/AT      < 1484>: AT< OK
D/AT      < 1484>: AT> AT+CREG=2
D/AT      < 1484>: AT< OK
D/AT      < 1484>: AT> AT+CGREG=1
D/AT      < 1484>: AT< OK
D/AT      < 1484>: AT> AT+CCWA=1
D/AT      < 1484>: AT< OK
D/AT      < 1484>: AT> AT+CMOD=0
```

3.2.4.2 Check for RIL Version Number

Open adb shell and input logcat -b radio; check RIL version number through the output log, and search for the key word "RIL Daemon version", of which RIL_V4x.00.02 is the version number of RIL.

```
I/RILJ    < 505>: Connected to 'rild' socket
I/RILC    < 1264>: libril: new connection
I/RILC    < 1264>: RIL Daemon version: RIL_V4x.00.02
D/RILJ    < 505>: [UNSLI]< UNSOL_RIL_CONNECTED {?}
D/RILJ    < 505>: [0001]> RADIO_POWER off
D/RIL     < 1264>: onRequest: RADIO_POWER
D/RIL     < 1264>: mly1:onRequest: RADIO_POWER
```

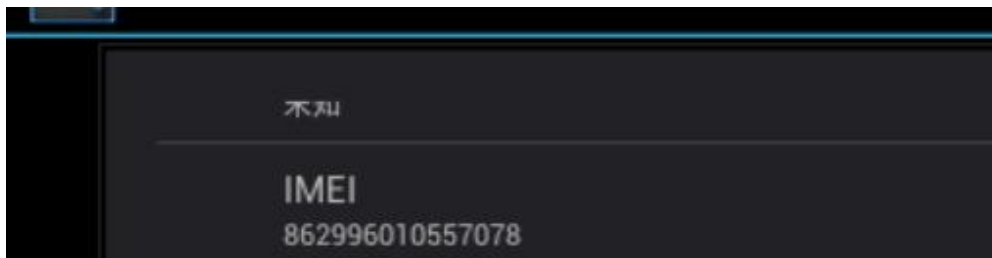
3.2.5 Debug for Signal Quality, Telephone, SMS

3.2.5.1 Query for Module IMEI

After the completion of the initialization, the upper-level application will query the IMEI number of the module, and the following is the normal query results of IMEI number in RIL log.

```
D/RIL < 1264>: onRequest: GET_IMEI
D/RIL < 1264>: mly1:onRequest: GET_IMEI
D/RIL < 1264>: mly2:onRequest: 38
D/RIL < 1264>: mly3:sState: 4
D/AT < 1264>: AT> AT+CGSN?
```

Enter the "Settings" - > "panel information" - > "state information" to query IMEI. If it's "Unkown", the interaction between the RIL and the module is abnormal.



3.2.5.2 Signal Strength Query

After the successful operation of RIL, the upper level can query the signal strength regularly, and the following is the normal query results of signal intensity in log RIL:

```
D/RIL < 1505>: onRequest: SIGNAL_STRENGTH
D/RIL < 1505>: mly1:onRequest: SIGNAL_STRENGTH
D/RIL < 1505>: mly2:onRequest: 19
D/RIL < 1505>: mly3:sState: 2
D/AT < 1505>: AT> AT+CSQ
D/AT < 1505>: AT< +CSQ: 13,99
D/AT < 1505>: AT< OK
D/RILJ < 505>: [0097]< SIGNAL_STRENGTH SignalStrength: 13
```

Signal strength displayed on the interface:



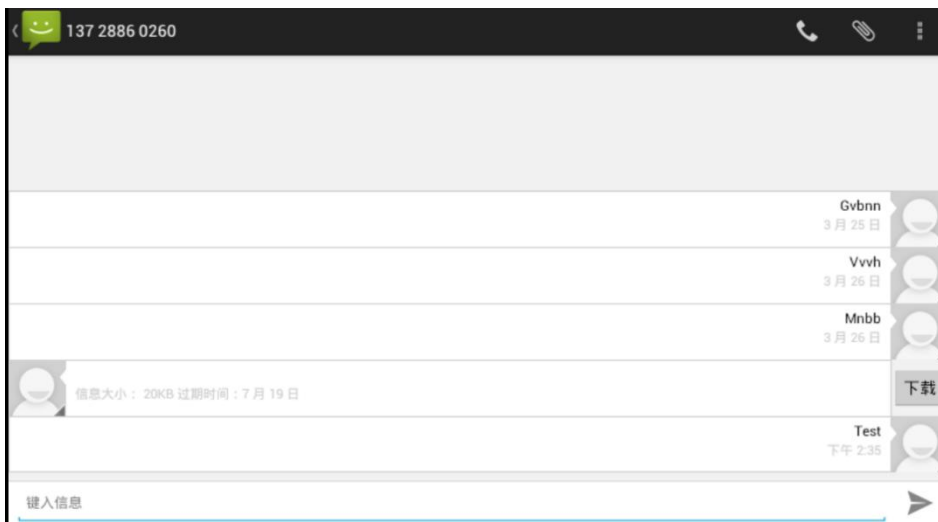
3.2.5.3 Voice and SMS Detection

If the above debug and test results are normal, then we can proceed the voice call as well as sending and receiving SMS. Check whether the corresponding commands are sent via adb by logcat -b radio command during corresponding operations.

Dial 10086, and RIL information are displayed as follows:

```
D/use-Rlog/RLOG-AT< 227>: AT> AT+CMUT=0
D/use-Rlog/RLOG-AT< 227>: AT< OK
D/use-Rlog/RLOG-RIL< 227>: onRequest: DIAL
D/use-Rlog/RLOG-AT< 227>: AT> ATD10086;
D/RILJ < 797>: [4989]< SET_MUTE
D/use-Rlog/RLOG-AT< 227>: AT< +STKCTRLIND: 0,0,, "10086",129
D/use-Rlog/RLOG-AT< 227>: AT< OK
D/use-Rlog/RLOG-AT< 227>: AT< +CLCC: 1,0,2,0,0, "10086",129
```

Send "test" to the other side through SMS:

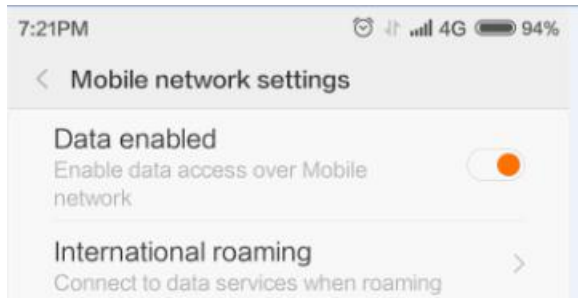


Corresponding log of RIL:

```
D/RILJ < 2033>: [0058]< SEND_SMS
D/RIL < 1484>: onRequest: SEND_SMS
D/AT < 1484>: AT> AT+CMGS=17
D/AT < 1484>: AT< >
D/AT < 1484>: AT> 0001000b813127880662f0000004d4f29c0e^Z
D/AT < 1484>: AT< +CMGS: 24
D/AT < 1484>: AT< OK
D/RILJ < 2033>: [0058]< SEND_SMS < messageRef = 0, errorCode
11>
```

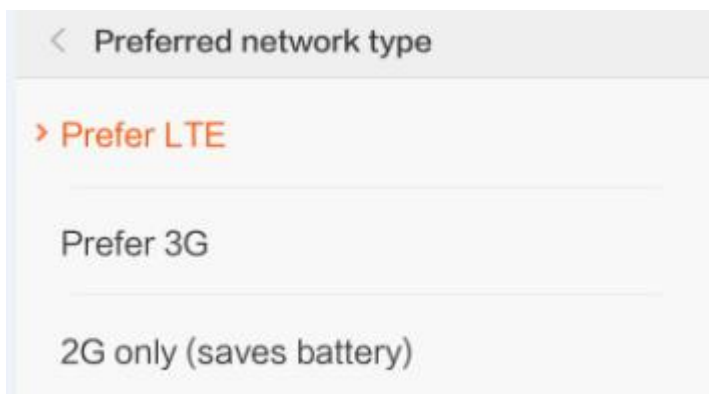
3.2.5.4 Enabling Data Network

Enter “settings”->“mobile network”->“enable data network”, and click to enable, the upper left corner of the signal bar will display the icon of the current network type:



3.2.5.5 Preferred Network

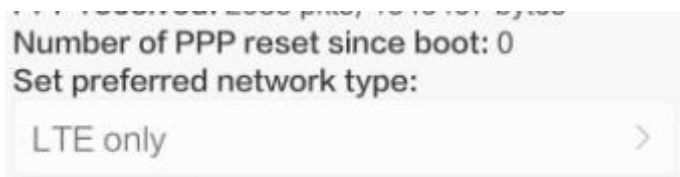
Enter “settings”->“mobile network”->“preferred network”, and click the corresponding menu to switch to the corresponding network



If there is no “Prefer 4G” in the menu, the customer needs to modify the UI layer to add the option.

If there is no “Prefer 4G” in the menu, customers can enter the engineering mode for network switching:

Open the Phone app -> input `***#4636***` -> select “Tablet information” -> pull down “Set preferred network type” menu, and select LTE Only to switch to 4G network. This method is only used for testing, and the final solution is still adding the “Prefer 4G” menu in the UI layer.



3.2.5.6 Debug Audio Channel Switch and Volume Adjustment

Android engineers add contents to telephone service layer, HAL layer or the driver layer for corresponding platforms to implement audio channel switch and volume adjustment functions by AT commands