Wireless Module Expert

Android RIL Driver User Guide

GSM/GPRS/UMTS/HSPA/LTE Module Series

Rev. Android_RIL_Driver_User_Guide_V1.3

Date: 2015-07-10

www.quectel.com

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Office 501, Building 13, No.99, Tianzhou Road, Shanghai, China, 200233
Tel: +86 21 5108 6236
Mail:info@quectel.com

Or our local office, for more information, please visit:
http://www.quectel.com/support/salesupport.aspx

For technical support, to report documentation errors, please visit:
http://www.quectel.com/support/techsupport.aspx
Or Email: Support@quectel.com

GENERAL NOTES

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2015-02-27	Carl YIN	Initial
1.1	2015-03-25	Carl YIN	Updated supported products
1.2	2015-04-07	Kent XU	Added Zero Packet feature in Section 3.3.3.
1.3	2015-07-10	Kent XU	1. Added GSM modules in Supported Products 2. Added Android 5.x in Supported Android Versions

Contents

Table Index

1 Introduction

This document mainly introduces how to integrate RIL Driver into Android OS of your target machine and how to modify the configuration files for starting RIL service and PPP dialling.

1.1. Directory Structure

The file structure of Quectel RIL Driver Package:

```
├── Android.mk
├── atchannel.c
├── atchannel.h
├── at_tok.c
├── at_tok.h
├── misc.c
├── misc.h
├── ql-pppd.c
├── gsm0710muxd_bp.c
├── quectel_ril_porting_guide.txt
└── reference-ril.c
```

2 Overview of Android RIL Driver

2.1. Supported Products

Table 1: Supported Products

Product	Support or Not
UC20	YES
UC15	YES
UGxx	YES
EC20	YES
Quectel GSM Modules	YES

2.2. Supported Functions

Table 2: Supported Functions

Function	Support or Not
SMS	YES
VOICE CALL	YES
DATA SERVICE	YES
SIM TOOL KIT	NO
PHONEBOOK	YES

2.3. Supported Android Versions

At present, Quectel RIL driver supports the following Android versions:

Table 3: Supported Android Versions

Versions	Support or not
Android 2.x	YES
Android 3.x	YES
Android 4.x	YES
Android 5.x	YES

3 RIL Integration

The first part describes the RIL driver structure. The rest introduce how to set up Android system with the RIL driver.

3.1. RIL Driver Structure

Android RIL (Radio Interface Layer) provides the abstract layer between Telephony service and Radio hardware.

The following illustration describes the RIL's position in the Android architecture.

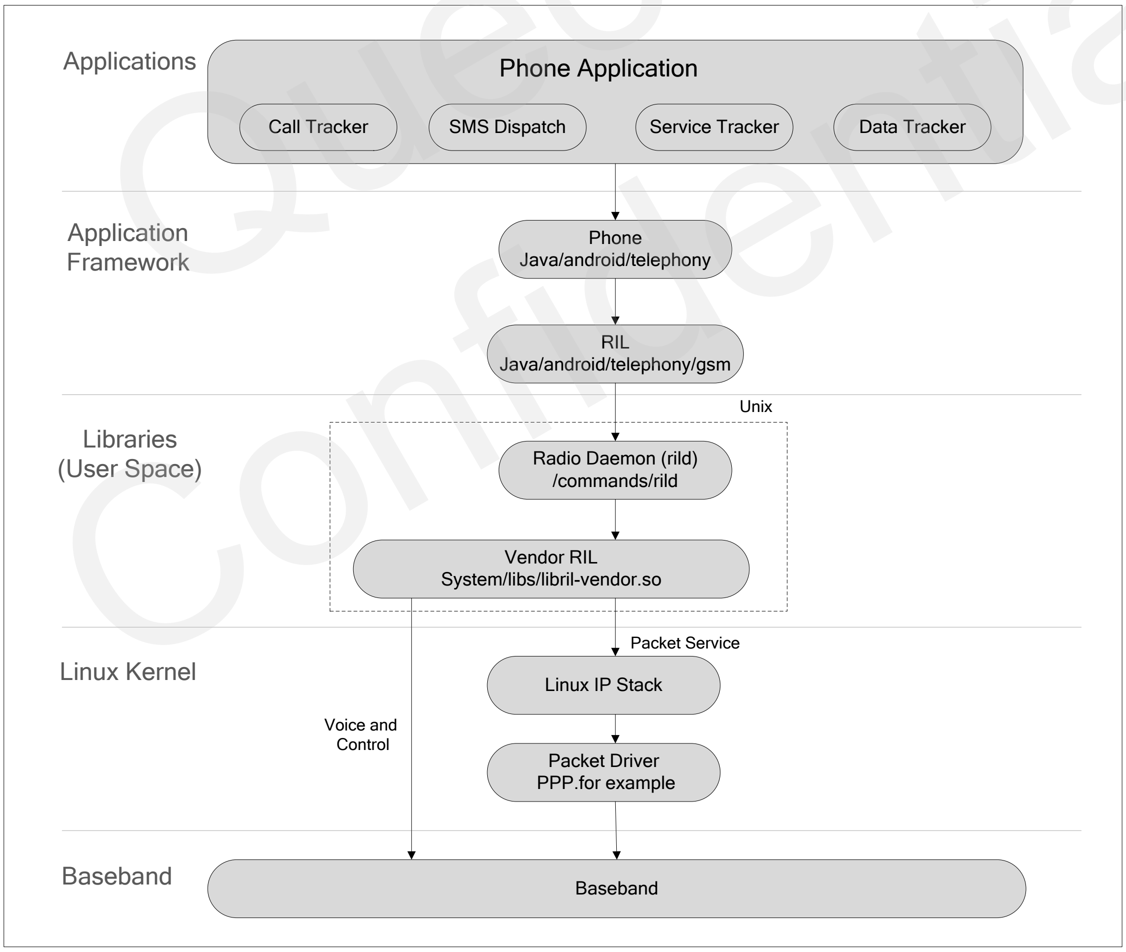


Figure 1: RIL Driver Structure

The RIL in Android is located between Kernel and Application Framework. It is divided into two parts, one is RILD and the other is Vendor RIL.

RILD is responsible for the communication between Socket and Application Framework.

Vendor RIL is responsible for communication with Radio via AT command channel and Packet data channel (PDCH). AT command channel is used for communicating with Radio directly and PDCH is used for data service.

The java framework of RIL is divided into two parts too. One is RIL module and the other is Phone module. The RIL module is used to communicate with the lower RILD. The Phone module directly provides phone function interfaces to application through which you can call to realize the phone functions.

3.2. PPP Configuration in Linux Kernel

You need to configure the kernel to support PPP dial up.

There are several mandatory selected items in kernel configuration, you should follow the steps below to configure the kernel:

Step 1:

```
cd <your kernel directory>
```

Step 2:

```
make menuconfig
```

Step 3:

```
Device Drivers    --->
    [*] Network device support    --->
        <*> PPP (point-to-point protocol) support
        <*> PPP support for async serial ports
        <*> PPP support for sync tty ports
        <*> PPP Deflate compression
```

3.3. USB Drivers For UC15/UC20/UGxx/EC20 in Linux Kernel

If you are using Quectel GSM modules, please skip this section.

You need to configure the kernel to support the UC15/UC20/UGxx/EC20 modules and connect it to the applicable interface.

There are several mandatory selected items in kernel configuration, you should follow the steps below to configure the kernel:

Step 1:

```
cd <your kernel directory>
```

Step 2:

```
make menuconfig
```

3.3.1. USB Driver Configuration for UC15/UC20/EC20

```
[*]Device Drivers →
    [*]USB Support →
        [*]USB Serial Converter support →
            [*]USB driver for GSM and CDMA modems
```

Add UC15/UC20/EC20's Vendor ID and Product ID in "option_ids[]" in the file "drivers/usb/serial/option.c":

```
    static const struct usb_device_id option_ids[] = {
    #start // Added by Quectel
      { USB_DEVICE(0x05C6, 0x9090) },   //For UC15
      { USB_DEVICE(0x05C6, 0x9003) },   //For UC20
      { USB_DEVICE(0x05C6, 0x9215) },   //For EC20
    #end
```

3.3.2. USB Driver Configuration for UGxx

```
[*]Device Drivers →
    [*]USB Support →
        [*]USB Modem (CDC ACM) support
```

3.3.3. Add the Zero Packet Mechanism

3.3.3.1. Zero Packet Mechanism for UC15/UC20/EC20

As required by the USB protocol, you need to add the mechanism for processing zero packets during transmission of “usb_wwan.c” file under “[KERNEL]/drivers/usb/serial”.

However, zero packet mechanism is valid for **Linux 2.6.35** or **later** versions. This document takes the **Linux 3.2** as an example, the other versions could be a bit different, but it is basically the same.

You need to add the following statements to the “**usb_wwan_setup_urb**” function, as shown below:

```
static struct urb *usb_wwan_setup_urb(struct usb_serial *serial, int endpoint,
                                      int dir, void *ctx, char *buf, int len,void (*callback) (struct urb *))
{
    struct urb *urb;
    if (endpoint == -1)
        return NULL;    /* endpoint not needed */
    urb = usb_alloc_urb(0, GFP_KERNEL);    /* No ISO */
    if (urb == NULL) {
        dbg("%s: alloc for endpoint %d failed.", __func__, endpoint);
        return NULL;
    }
    /* Fill URB using supplied data. */
    usb_fill_bulk_urb(urb, serial->dev,
              usb_sndbulkpipe(serial->dev, endpoint) | dir,
              buf, len, callback, ctx);
    #start   //Added by Quectel for Zero Packet
    if (dir == USB_DIR_OUT) {
        struct usb_device_descriptor *desc = &serial->dev->descriptor;
        if (desc->idVendor == 0x05C6 && (desc->idProduct == 0x9003 || desc->idProduct == 
0x9090 || desc->idProduct == 0x9215))
            urb->transfer_flags |= URB_ZERO_PACKET;
    }
    #end
    return urb;
}
```

3.3.3.2. Zero Packet Mechanism for UGxx

As required by the USB protocol, you need to add the mechanism for processing zero packets during transmission of “cdc-acm.c” file under “[KERNEL]/drivers/usb/class”.

This document takes the **Linux 3.2** as an example, the other versions could be a bit different, but it is basically the same.

You need to add the following statements to the “**acm_probe**” function, as shown below:

```
…….
for (i = 0; i < ACM_NW; i++) {
    struct acm_wb *snd = &(acm->wb[i]);
    snd->urb = usb_alloc_urb(0, GFP_KERNEL);
    if (snd->urb == NULL) {
        dev_err(&intf->dev,
            "out of memory (write urbs usb_alloc_urb)\n");
        goto alloc_fail7;
    }
    if (usb_endpoint_xfer_int(epwrite))
        usb_fill_int_urb(snd->urb, usb_dev,
            usb_sndbulkpipe(usb_dev, epwrite->bEndpointAddress),
            NULL, acm->writesize, acm_write_bulk, snd, epwrite->bInterval);
    else
        usb_fill_bulk_urb(snd->urb, usb_dev,
            usb_sndbulkpipe(usb_dev, epwrite->bEndpointAddress),
            NULL, acm->writesize, acm_write_bulk, snd);
    snd->urb->transfer_flags |= URB_NO_TRANSFER_DMA_MAP;
    #start   //Added by Quectel for Zero Packet
    if (usb_dev->descriptor.idVendor == 0x1519 && usb_dev->descriptor.idProduct == 0x0020)
        snd->urb->transfer_flags |= URB_ZERO_PACKET;
    #endif
   snd->instance = acm;
}
usb_set_intfdata(intf,acm)
…….
```

3.4. RIL Driver Integration

At present, Quectel provides RIL driver in the form of source code. You only need to copy the RIL driver source code files to the correct path on your project directory, and recompile the Android system.

The source path of the RIL driver files in RIL Driver package is:

Driver package/reference-ril

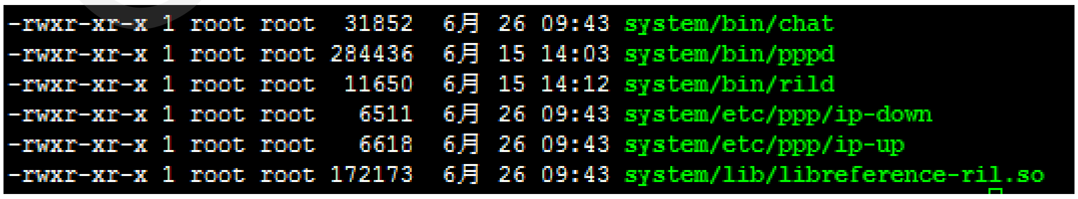
The destination path in Android system is:

($Android_src)/hardware/ril/

Please use following command to update these files's date stamp to make sure they will be compiled.

```
touch hardware/ril/*
```

After you recompile the Android system successfully, Please make sure following files exists in folder "out/target/product/($your_board_name)/system".



```
-rwxr-xr-x 1 root root  31852  6月 26 09:43 system/bin/chat
-rwxr-xr-x 1 root root 284436  6月 15 14:03 system/bin/pppd
-rwxr-xr-x 1 root root  11650  6月 15 14:12 system/bin/rild
-rwxr-xr-x 1 root root   6511  6月 26 09:43 system/etc/ppp/ip-down
-rwxr-xr-x 1 root root   6618  6月 26 09:43 system/etc/ppp/ip-up
-rwxr-xr-x 1 root root 172173  6月 26 09:43 system/lib/libreference-ril.so
```

Figure 2: RIL Files

NOTE

Quectel would not give the PPP script, because we have integrated the PPP script in the source code of the RIL driver. And the chat, ip-up and ip-down would be generated when the RIL driver has been compiled.

3.5. System Configuration

In order to use the RIL driver normally, you also have to modify some Android system configuration files.

3.5.1. Configure "init.rc"

Add service "ril-daemon" in "init.rc":

If you are using UC15/UC20/UGxx/EC20, these products are accessed by USB Interface. Please add following service "ril-daemon" in "init.rc":

```
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so
    class main
    socket rild stream 660 root radio
    socket rild-debug stream 666 radio system
    user root
    group radio cache inet misc audio sdcard_rw log
```

If you are using GSM Modules, these products are accessed by UART Interface. Please add following service "ril-daemon" in "init.rc":

```
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so -- -d <UART port name> –B
<baud rate> –C <hardware flow control>
    class main
    socket rild stream 660 root radio
    socket rild-debug stream 666 radio system
    user root
    group radio cache inet misc audio sdcard_rw log
```

The following parameter needs to be configured:

- **-d <UART port name>**

The name of UART port you are using. For example: /dev/ttyS1

It is optional to configure the following two parameters:

- **-B <baud rate>**

The speed UART port. For example: 115200、230400、460800. And the default baud rate is 115200.

- **-C <hardware flow control>**

1: open hardware flow control
0: close hardware flow control

And the default hardware flow control is closed.

The "init.rc" file may be located in different paths according to different projects, for example:

- system/core/rootdir/init.rc
- device/fsl/imx6/init.rc
- device/ti/am335xevm_sk/init.am335xevm.rc
- device/generic/x86/init.rc
- device/samsung/smdkv210/init.smdkv210_sdmmc.rc

3.5.2. Modify the Right of RILD

RILD (ril-daemon) requires root privilege. So you need to comment out the function ***switchUser()*** in the file "($Android_src)/hardware/ril/rild/rild.c":

```
OpenLib:
#endif
//switchUser();

dlHandle = dlopen(rilLibPath, RTLD_NOW)
```

4 Debugging Method

4.1. Method of Catching LOG

1) Catch the log of RIL module by typing the following command in window’s CMD tool:

```
adb logcat –b radio –v time
```

2) Catch the log of Android system by typing the following command in window’s CMD tool:

```
adb logcat –v time
```

3) Sometimes, you may want to make tests on lots of devices or for long time, it is not convenient to connect all devices with PC via USB cables. You can log file by following command:

```
adb logcat –b radio –v time –f <filename> &
```

The char ‘&’ makes the ‘logcat’ process running in background, so you can disconnect your devices.

4) When you finish your tests, you can fetch log files from devices to local directory by following command:

```
adb pull <filename> <local directory>
```

4.2. Some Common LOG Tags

RIL	/hardware/ril/reference-ril/refereince-ril.c
AT	/hardware/ril/reference-ril/atchannel.c
RILD	/hardware/ril/rild/rild.c
RILC	/hardware/ril/libril/ril.cpp
RILB	/frameworks/base/telephony/java/com/android/internal/telephony/BaseCommands.java
RILJ	/frameworks/base/telephony/java/com/android/internal/telephony/gsm/RIL.java
GSM	/frameworks/base/telephony/java/com/android/internal/telephony/gsm/GSMPhone.java

5 Appendix A Reference

Table 4: Terms and Abbreviations

Abbreviation	Description
RIL	Radio Interface Layer
TA	Terminal Adapter
MS	Mobile Station
GSM	Global System for Mobile Communications
WCDMA	Wideband Code Division Multiple Access
VID	Vendor ID
PID	Product ID