RG500U-CN&RM500U-CN Driver User Guide

Content

1. INTRODUCTION

This document mainly introduces how to integrate the USB-to-serial driver and USB network card driver of Quectel 5G modules RG500U-CN and RM500U-CN into Linux system, how to test AT command and USB network card dial-up function, and common problems related to driver migration.

2. OVERVIEW OF LINUX USB INTERFACE

The USB drivers of Quectel RG500U-CN and RM500U-CN modules contain several different functional interfaces. The following table takes RG500U-CN as an example to describe the details of the module's USB interface under the Linux operating system:

Table 1: Linux USB Interface Information

Module's VID and PID	USB Drivers	Interfaces
VID：0x2c7c PID：0x0900	USB RNDIS/ECM/NCM/MBIM Network Card	0/1: USB network adapter
	USB serial option	2: DIAG command communication port
		3: LOG Port
		4: AT Command Communication Port
		5：Modem Command Communication Port
		6: NMEA Command Communication Port
	USBFS	7: ADB Command Communication Port

3. LINUX USB DRIVER

3.1. USB TO SERIAL PORT

When the module successfully loads the USB-to-serial option driver, Linux will create multiple serial device files with names such as ttyUSB0, ttyUSB1, ttyUSB2, etc. name).

The following chapters introduce how to change USB to serial port option driver.

3.1.1. ADD VID AND PID

Add the module's VID and PID information in the file [KERNEL]/drivers/usb/serial/option.c as follows:

```
static const struct usb_device_id option_ids[] = {
#if 1 //Added by Quectel
{ USB_DEVICE_AND_INTERFACE_INFO(0x2c7c, 0x0900, 0xff, 0x00, 0x00) },
#endif
```

Note

If the user uses the driver file (option.c) provided by Quectel, it is recommended that the user check the option_probe function in [KERNEL]/drivers/usb/serial/option.c. According to the introduction in Chapter 2, the module interface 2/3/4/5/6 is the serial port. It is necessary to ensure that the interface after the USB interface number exceeds 4 will not be filtered out.

3.1.2. ADD THE ZERO PACKET MECHANISM

For the USB Bulk Out transmission mode, if the length of the data to be sent is an integer multiple of the length of the USB data packet, an additional data packet with a length of zero needs to be sent to notify the peer end that the data transfer is complete.

- For Linux kernel 2.6.35 and above, please add the following statement to the file

[KERNEL]/drivers/usb/serial/usb_wwan.c.

```
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 *))
{
......
    usb_fill_bulk_urb(urb, serial->dev,
                usb_sndbulkpipe(serial->dev, endpoint) | dir,
                buf, len, callback, ctx);
    #if 1     //Added by Quectel for zero packet
    if (dir == USB_DIR_OUT) {
        struct usb_device_descriptor *desc = &serial->dev->descriptor;

        if (desc->idVendor == cpu_to_le16(0x2C7C))
            urb->transfer_flags |= URB_ZERO_PACKET;
    }
    #endif
    return urb;
}
```

- For Linux kernel 2.6.34 and below, please add the following statement to the file [KERNEL]/drivers/usb/serial/option.c

```
/* Helper functions used by option_setup_urbs */
static struct urb *option_setup_urb(struct usb_serial *serial, int endpoint,
        int dir, void *ctx, char *buf, int len,
        void (*callback)(struct urb *))
{
.....
    usb_fill_bulk_urb(urb, serial->dev,
              usb_sndbulkpipe(serial->dev, endpoint) | dir,
              buf, len, callback, ctx);
    #if 1    //Added by Quectel for zero packet
    if (dir == USB_DIR_OUT) {
        struct usb_device_descriptor *desc = &serial->dev->descriptor;

        if (desc->idVendor == cpu_to_le16(0x2C7C))
            urb->transfer_flags |= URB_ZERO_PACKET;
    #endif
    return urb;
}
```

3.1.3. ADD RESET-RESUME MECHANISM

Some USB host controllers or USB hubs may power down or reset when the MCU enters suspend or sleep mode, and the MCU cannot automatically resume USB devices after exiting suspend or sleep mode. Add the following statement to enable reset recovery flow.

- For Linux kernel 3.5 and above, please add the following statement to the file [KERNEL]/drivers/usb/serial/option.c

```
static struct usb_serial_driver option_1port_device = {
.....
#ifdef CONFIG_PM
    .suspend            = usb_wwan_suspend,
    .resume             = usb_wwan_resume,
#if 1   //Added by Quectel
    .reset_resume    = usb_wwan_resume,
#endif
#endif
};
```

- For Linux kernel 3.4 and below, please add the following statement to the file [KERNEL]/drivers/usb/serial/usb-serial.c

```
/* Driver structure we register with the USB core */
static struct usb_driver usb_serial_driver = {
        .name =         "usbserial",
        .probe =        usb_serial_probe,
        .disconnect =   usb_serial_disconnect,
        .suspend =      usb_serial_suspend,
        .resume =       usb_serial_resume,
#if 1 //Added by Quectel
        .reset_resume = usb_serial_resume,
#endif
        .no_dynamic_id =        1,
        .supports_autosuspend = 1,
};
```

3.1.4. INCREASE THE QUANTITY AND CAPACITY OF THE BULK OUT URBS (LINUX KERNEL 2.6.29 AND BELOW)

For Linux kernel 2.6.29 and below, the number and capacity of batch output URBs need to be increased to obtain faster uplink rate. Please add the following statement to the file [KERNEL]/drivers/usb/serial/option.c.

```
#define N_IN_URB 4
#define N_OUT_URB 4          //Increase the quantity of the bulk out URBs to 4
#define IN_BUFLEN 4096
#define OUT_BUFLEN 4096      //Increase the capacity of the bulk out URBs to 4096
```

3.1.5. MODIFY KERNEL CONFIGURATION

In order to use the USB-to-serial option driver, the following Linux kernel configuration items must be enabled:

- CONFIG_USB_SERIAL
- CONFIG_USB_SERIAL_WWAN
- CONFIG_USB_SERIAL_OPTION

3.2. USB NETWORK CARD DRIVER

The module supports four network card functions of MBIM/RNDIS/ECM/NCM. The Linux system supports these USB network card functions by default. The system has a built-in driver module and does not require any modification to the driver files of the Linux system. The driver source code is maintained by GNU Linux.

After the module is connected to the Linux Host and successfully loaded with the corresponding network card driver for USB, a network card will be generated on the Host. cdc_mbim also generates a cdc-wdm character device for command interaction. The network card mode of the module can be configured by AT commands, as shown in the following table:

USB Network Card Mode	Kernel Driver	AT Command
ECM	cdc_ether	AT+QCFG="usbnet" ,1 Configure ECM Mode
MBIM	cdc_mbim (Kernel 3.18 and above)	AT+QCFG="usbnet",2 Configure MBIM Mode
RNDIS	rndis_host	AT+QCFG="usbnet",3 Configure RNDIS Mode
NCM	cdc_ncm	AT+QCFG="usbnet",5 Configure NCM Mode

For details on the above AT commands, please refer to document [1].

To use the USB NIC function, follow the steps below to configure the kernel.

Step 1: Execute the following command to switch to the kernel directory.

CD<KERNEL content>

Step 2: Execute the following commands to set environment variables and export the defconfig file in the user device operating system.

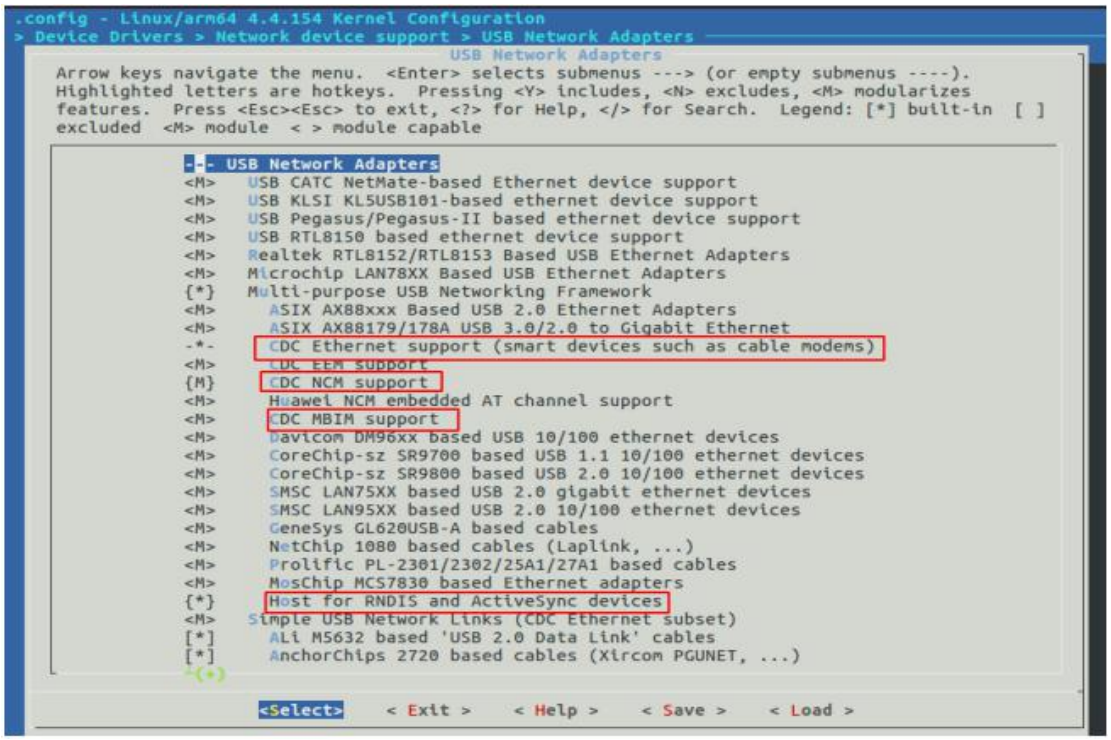
```
export ARCH=arm
export CROSS_COMPILE=arm-none-linux-gnueabi-
make bcmrpi_defconfig
```

Step 3: Execute the following command to compile the kernel.

```
make menuconfig
```

Step 4: Execute the following command to enable the USB network card function through the options shown in the figure below.

```
> Device Drivers > Network device support > USB Network Adapters
```



3.3. ENABLE PPP DIALING (NOT RECOMMENDED)

PPP dial-up has the following disadvantages compared to the USB network card Internet access method:

- More complicated to use
- Higher CPU consumption under the same internet speed
- The data transmission cannot reach the theoretical rate

Therefore, PPP dialing is not recommended. If required, the following Linux kernel configuration items must be enabled:

- CONFIG_PPP
- CONFIG_PPP_ASYNC
- CONFIG_PPP_SYNC_TTY
- CONFIG_PPP_DEFLATE

4. AT AND USB NETWORK CARD DIAL-UP FUNCTION TEST

4.1. AT FUNCTION TEST

After the module successfully loads the USB-to-serial option driver, Linux will create multiple serial device files with names such as ttyUSB0, ttyUSB1, ttyUSB2, etc. in the /dev directory (the serial device file names under Linux are not fixed, and the system automatically assigns available names), where the third serial port is the AT command port of the module. A serial port tool such as minicom or busybox microcom can be used to test AT functionality.

The following figure shows the AT function result tested by the busybox microcom tool. The third serial port name assigned by the system in the example is /dev/ttyUSB2.

4.2. USB NETWORK CARD DIAL TEST

For details of USB network card dialing, please refer to the document [2]

5. FAQ

5.1. HOW TO CHECK WHETHER USB DRIVER EXISTS IN THE MODULE

The list of files under the directory /sys/bus/usb/drivers can be used to see which USB drivers have been existed to the Linux system. E.g

```
carl@carl-OptiPlex-7010:~$ ls /sys/bus/usb/drivers
hub option usb usbfs usbhid usbserial usbserial_generic rndis_host cdc_ether cdc_ncm cdc_mbim
```

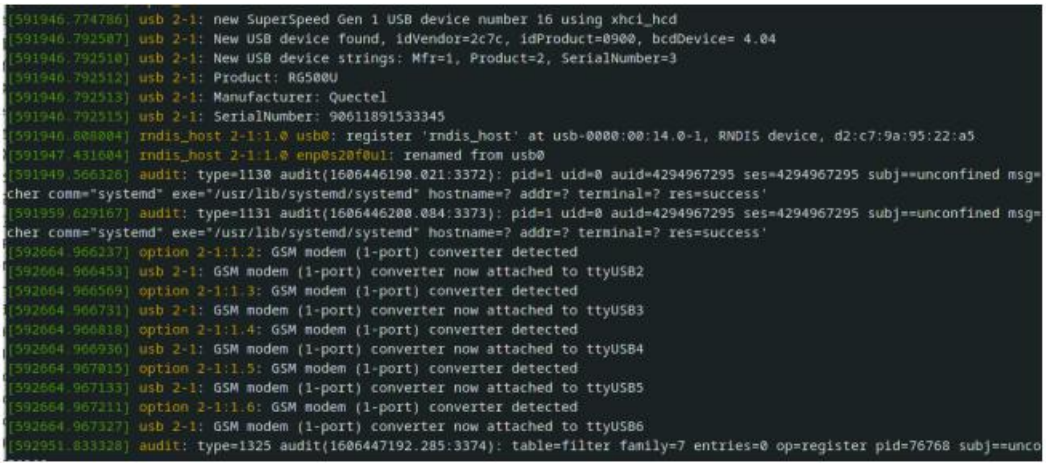
If you need to change USB to serial driver, please make sure option exists. If you need to port the USB NCM driver, please make sure that cdc_ncm exists;

If you need to port the USB ECM driver, please ensure that cdc_ether exists; if you need to migrate the USB MBIM driver, please ensure that cdc_mbim exists;

To port USB RNDIS driver, make sure rndis_host exists.

5.2. HOW TO CHECK WHETHER THE MODULE WORKS WELL WITH THE CORRESPONDING USB DRIVER

This chapter shows the corresponding log information printed by the Linux system when the module loads the USB driver correctly. Users can check whether the module has correctly loaded the USB driver by comparing the logs in this chapter with the actual logs obtained.



6. APPENDIX A REFERENCES

Reference Document

Number	Document	Description
[1]	Quectel_RG500U-CN&RM500U-CN_AT Command	Quectel_RG500U-CN&RM500U-CN_ Module AT Command Manual
[2]	Quectel_RG500U-CN&RM500U-CN_ Network Dial Guide	Quectel_RG500U-CN&RM500U-CN_ Network Dial Application Guide

Terms and Abbreviations

Abbreviations	Descriptions
APN	Access Point Name
ADB	Android Debug Bridge
CDC	Communications Device Class
CPU	Central Processing Unit
DNS	Central Processing Unit
ECM	Ethernet Control Mode
IP	Internet Protocol
MCU	Microcontroller Unit
MBIM	Mobile Broadband Interface Model
NCM	Network Control Model
NMEA	NMEA (National Marine Electronics Association) 0183 Interface Standard
RNDIS	Remote Network Driver Interface Specification
PID	Product ID
PPP	Point-to-Point Protocol
VID	Vendor ID
URB	USB Request Block
USB	Universal Serial Bus