EC200T Linux USB Driver User Guide

LTE Standard Module Series

Rev. EC200T_Linux_USB_Driver_User_Guide_V1.0

Date: 2020-01-08

Status: Released

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

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai, China 200233
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. 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 to: support@quectel.com

GENERAL NOTES

COPYRIGHT

About the Document

Revision History

Revision	Date	Author	Description
1.0	2020-01-08	Carl YIN	Initial

Contents

Table Index

Figure Index

1 Introduction

This document introduces how to integrate the USB driver (including USB serial option driver, ECM and RNDIS) for Quectel EC200T module in Linux operating system, and how to test the module after the USB driver is loaded successfully.

Table 1: Applicable Module and Supported Drivers

Module Series	Module	Supported Drivers	Comment
LTE Standard	EC200T	USB Serial	Refer to ***Chapter 3.2***.
		ECM or RNDIS	Refer to ***Chapter 3.3***.

2 Linux USB Driver Overview

USB on Quectel EC200T module contains several different functional interfaces. The following table describes the interface information of EC200T module in the Linux system.

Table 2: Interface Information

Module	USB Driver	Interface
EC200T (VID: 0x02c7c PID: 0x6026)	ECM or RNDIS	usbX refers to interface 0&1 that can be used as USB network adapter
	USB Serial Option	ttyUSB0 used for DM.
	USB Serial Option	ttyUSB1 used for AT command communication.
	USB Serial Option	ttyUSB2 used for PPP connections or AT command communication.

3 System Setup

This chapter mainly describes the general structure of the USB stack in Linux and how to use USB Serial option, ECM and RNDIS drivers, as well as how to compile and load the drivers.

3.1. Linux USB Driver Structure

USB is a kind of hierarchical bus structure. Linux USB host driver includes three parts: USB host controller driver, USB core and USB device drivers. The data transmission between USB devices and host is achieved by USB controller. The following picture illustrates the architecture of USB driver.

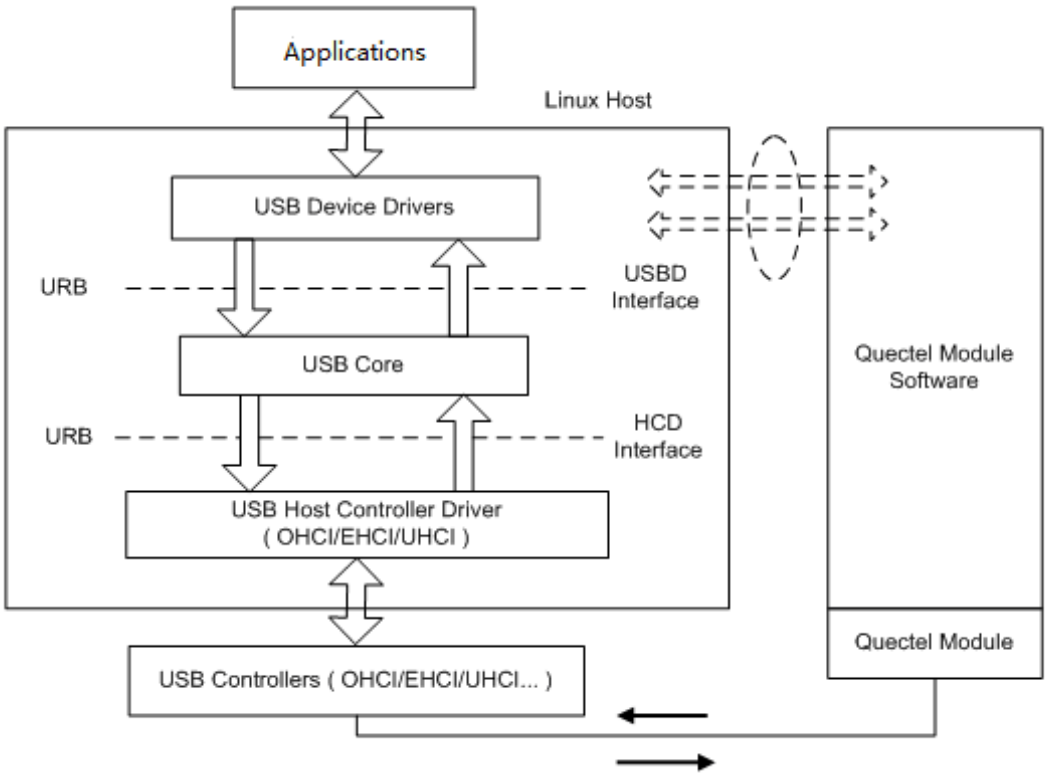

Figure 1: Linux USB Driver Structure

USB host controller driver, the bottom of the hierarchical structure, is a USB driver which interacts directly with hardware.

USB core, the core of the whole USB host driver, is used for the management of USB bus, USB bus devices and USB bus bandwidth; it provides the interfaces for USB device drivers, through which the applications can access the USB system files.

USB device drivers interact with the applications, and mainly provide the interfaces for accessing the specific USB devices.

3.2. USB Serial Option Driver

When EC200T is attached to the USB serial option driver, the driver will create device files named as *ttyUSB0/ttyUSB1/ttyUSB2…*in directory */dev*.

The following parts show how to integrate USB serial driver to Linux operating system.

3.2.1. Add VID and PID

In order to recognize the module, the module's VID and PID information as below need to be added to the file *[KERNEL]/drivers/usb/serial/option.c*.

```
static const struct usb_device_id option_ids[] = {
#if 1 //Added by Quectel
    { USB_DEVICE(0x2C7C, 0x6026) }, /* Quectel EC200T */
    { USB_DEVICE(0x2C7C, 0x6000) }, /* Quectel EC200T */
#endif
```

3.2.2. Add the Zero Packet Mechanism

As required by the USB protocol, the mechanism for processing zero packets during bulk-out transmission need to be added to the corresponding files.

- For Linux kernel version higher than 2.6.34, please add the following statements 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 version lower than 2.6.35, please add the following statements 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;
}
```

3.2.3. Add Reset-resume Mechanism

When MCU entering Suspend/Sleep mode, maybe USB host controllers/USB hubs will lose power or be reset, which cannot resume USB devices after MCU exits from Suspend/Sleep mode. Please add the following statements to enable reset-resume process.

- For Linux kernel version higher than 3.4, please add the following statements 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 version lower than 3.5, please add the following statements 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,
};
```

3.2.4. Increase the Quantity and Capacity of the Bulk out URBs

For Linux kernel version lower than 2.6.29, bulk out URBs need to be enlarged to get faster uplink speed.

Please add the following statements to the file *[KERNEL]/drivers/usb/serial/option.c*.

```
#define N_IN_URB 4
#define N_OUT_URB 4 //Quectel 1
#define IN_BUFLEN 4096
#define OUT_BUFLEN 4096 //Quectel 128
```

3.2.5. Use ECM or RNDIS

If ECM or RNDIS is required, please add the following statements to prevent modules' interface 0 from being used as USB serial device.

- For Linux kernel version higher than 2.6.30, please add the following statements to the file [KERNEL]/drivers/usb/serial/option.c.

```
static int option_probe(struct usb_serial *serial, const struct usb_device_id *id) {
    struct usb_wwan_intf_private *data;
    ……
```

```
#if 1    //Added by Quectel
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        __u16 idProduct = le16_to_cpu(serial->dev->descriptor.idProduct);

        //Quectel EC200T's interface 0 can be used as USB Network device (ecm, rndis)
        if (serial->interface->cur_altsetting->desc.bInterfaceClass != 0xFF)
            return -ENODEV;
    }
#endif

    /* Store device id so we can use it during attach. */
    usb_set_serial_data(serial, (void *)id);
    return 0;
}
```

- For Linux kernel version lower than 2.6.31, please add the following statements to the file [KERNEL]/drivers/usb/serial/option.c.

```
static int option_startup(struct usb_serial *serial)
{
……
    dbg("%s", __func__);
#if 1    //Added by Quectel
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        __u16 idProduct = le16_to_cpu(serial->dev->descriptor.idProduct);

        //Quectel EC200T's interface 0 can be used as USB Network device (ECM)
        if (serial->interface->cur_altsetting->desc.bInterfaceClass != 0xFF)
            return -ENODEV;
}
```

3.2.6. Modify Kernel Configuration

There are several items needed to be selected manually in kernel configuration, please follow the steps below to configure the kernel:

Step 1: Switch to kernel directory with the command below.

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

Step 2: Set environment variables, and import the board's "defconfig" file (taking Raspeberry Pi board as an example) with the command below.

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

```
make bcmrpi_defconfig
```

Step 3: Compile the kernel with the command below.

```
make menuconfig
```

Step 4: Enable CONFIG_USB_SERIAL_OPTION with the following options.

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

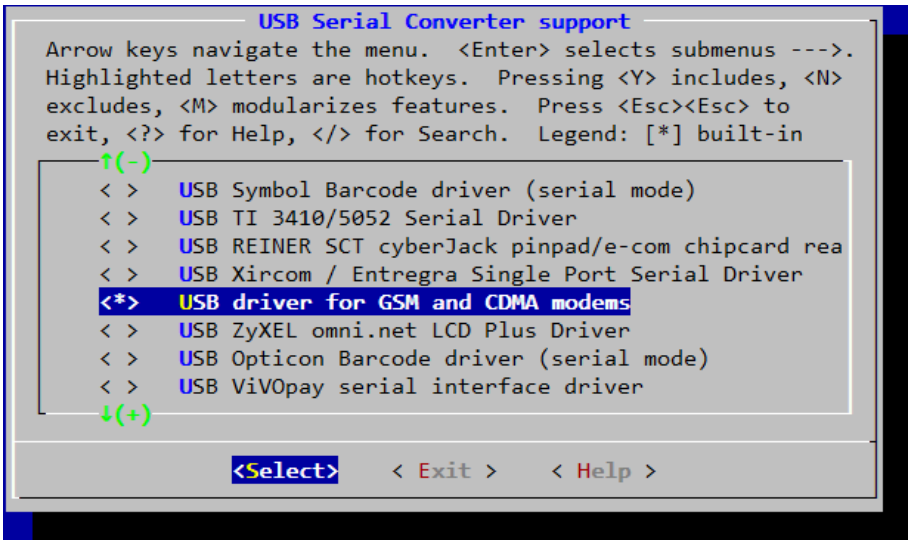

Figure 2: Configure USB Serial in Kernel

3.3. Configure Kernel to Support ECM or RNDIS

If ECM or RNDIS drivers need to be used, please follow the steps below to configure kernel to support the two drivers.

Step 1: Switch to kernel directory with the command below.

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

Step 2: Set environment variables, and import the board's "defconfig" file with the command below.

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

Step 3: Compile the kernel with the command below.

```
make menuconfig
```

Step 4: Enable CONFIG_USB_NET CONFIG_USB_NET_RNDIS_HOST with the following options.

```
[*] Device Drivers  →
    [*] Network device support  →
        <*>    USB Network Adapters    --->
                <*>        Host for RNDIS and ActiveSync devices
```

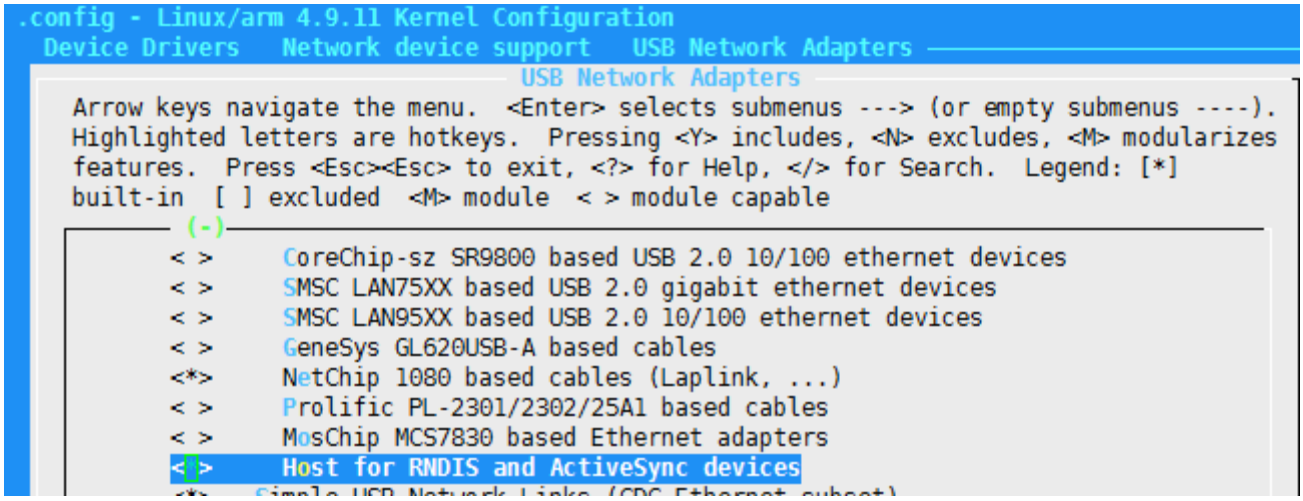

Figure 3: Configure ECM and RNDIS in Kernel

NOTE

ECM and RDNIS drivers are both maintained within the upstream Linux kernel (in-tree). And ECM and RDNIS have own USB Interface Class defined by USB-IF. Therefore, there is no need to insert Quectel modules' VID and PID to the source code files.

3.4. Configure Kernel to Support PPP

If PPP function needs to be used, please follow the steps below to configure kernel to support PPP.

Step 1: Switch to kernel directory with the command below.

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

Step 2: Set environment variables, and import board's "defconfig" file with the command below.

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

Step 3: Compile the kernel with the command below.

```
make menuconfig
```

Step 4: Enable CONFIG_PPP_ASYNC CONFIG_PPP_SYNC_TTY CONFIG_PPP_DEFLATE with the following options.

```
[*] Device Drivers →
    [*] Network device support →
        [*] PPP (point-to-point protocol) support
```

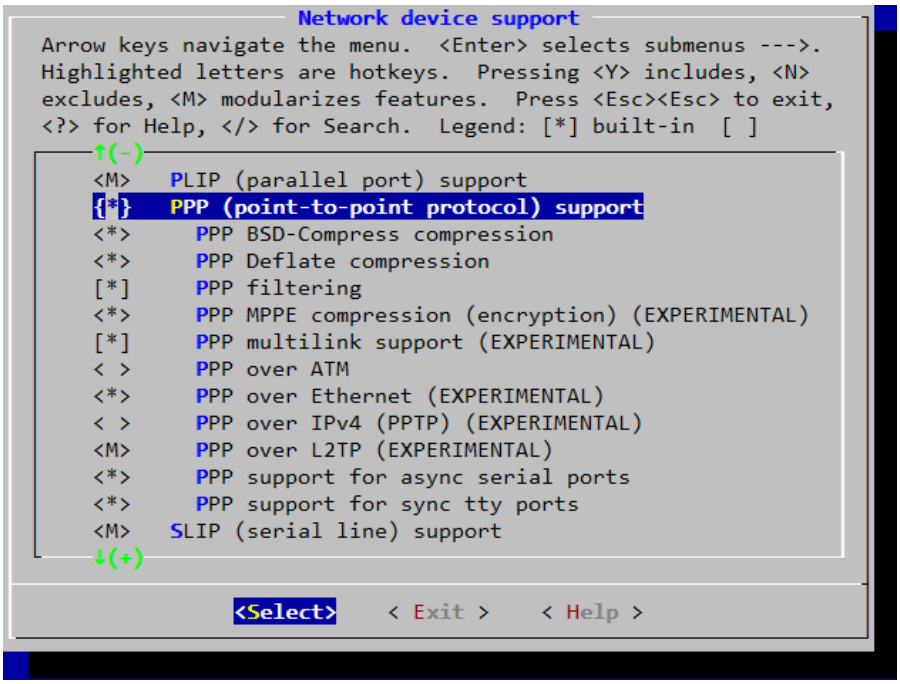

Figure 4: Configure PPP in Kernel

3.5. Install and Load Driver as a Kernel Module for PC in Linux

Quectel will provide the source files of USB serial option driver for testing modules on PC with Linux operating system like Ubuntu, and USB drivers can be installed with the command "make install" below, after that, please reboot the PC to take effect USB Drivers.

Install USB Serial option Driver with the following command:

```
# First use command `uanme –r` to query the current using kernel version
carl@carl-OptiPlex-7050:~/quectel/usb-serial-option$ uname    -r
4.4.0-31-generic
# Switch to the correspond kernel source directory
carl@carl-OptiPlex-7050:~/quectel/usb-serial-option$ cd 4.4.0/
carl@carl-OptiPlex-7050:~/quectel/usb-serial-option/4.4.0$ cp ../Makefile    ./
carl@carl-OptiPlex-7050:~/quectel/usb-serial-option/4.4.0$ sudo make install
```

4 Test the Module

Generally, AT and PPP functions are supported by the module. If ECM or RNDIS drivers have been installed, the USB network adapter function can also be used on the module. The following part shows how to test the functions of AT, PPP and ECM or RNDIS.

4.1. Test AT Function

After the module is connected and USB driver is loaded successfully, several device files will be created in */dev*.

The AT port of EC200T is */dev/ttyUSB1*.

Then UART port tools such as “minicom” or “busybox microcom” can be used to test AT function.

For example:

```
root@cqh6:~# busybox  microcom /dev/ttyUSB1
ati;+cpin?+csq;+cops?;+cgreg?
Quectel
EC200T
Revision: EC200TCNAAR02A03M1G

+CGREG: 0,1

+CPIN: READY

+COPS: 0,0,"CHINA MOBILE",7

OK
```

Figure 5: AT Test Result for EC200T

4.2. Test PPP Function

In order to set up PPP call, the following files are required. Please check if there are such files in products:

1. PPPD and chat program:
   If the two programs do not exist, the source code can be downloaded from https://ppp.samba.org/download.html and port them to the module.

2. One PPP script file named as */etc/ppp/ip-up* which is used to set DNS (Domain Name System). If there is no such file, please use the file of *linux-ppp-scripts\ip-up* provided by Quectel.

3. Three scripts named as *quectel-ppp*, *quectel-chat-connect* and *quectel-chat-disconnect* which are provided by Quectel in directory *linux-ppp-scripts*. Some changes may need to be made for different modules. For more information, please refer to *linux-ppp-scripts\readme*.

quectel-ppp, *quectel-chat-connect* and *quectel-chat-disconnect* need to be copied to the directory */etc/ppp/peers*.

And the default modem port in file of*'/etc/ppp/peers/quectel-ppp*' refers to */dev/ttyUSB3*, which should be changed to */dev/ttyUSB2*, as follows:

```
root@cqh6:/etc/ppp/peers# cat /etc/ppp/peers/quectel-ppp
# /etc/ppp/peers/quectel-pppd
# Usage:root>pppd call quectel-pppd
#Modem path, like /dev/ttyUSB3,/dev/ttyACM0, depend on your module, default path is /dev/ttyUSB3
/dev/ttyUSB2 115200
```

Then start to set up PPP call via the following command:

```
# pppd call quectel-ppp &
```

The process of dialing is shown as below (take the EC200T as an example):

```
root@cqh6:~# pppd call quectel-ppp &
root@cqh6:~# pppd options in effect:
debug           # (from /etc/ppp/peers/quectel-ppp)
nodetach            # (from /etc/ppp/peers/quectel-ppp)
dump            # (from /etc/ppp/peers/quectel-ppp)
noauth          # (from /etc/ppp/peers/quectel-ppp)
user test       # (from /etc/ppp/peers/quectel-ppp)
password ??????       # (from /etc/ppp/peers/quectel-ppp)
remotename 3gppp            # (from /etc/ppp/peers/quectel-ppp)
/dev/ttyUSB2        # (from /etc/ppp/peers/quectel-ppp)
115200          # (from /etc/ppp/peers/quectel-ppp)
```

```
lock       # (from /etc/ppp/peers/quectel-ppp)
connect chat -s -v -f /etc/ppp/peers/quectel-chat-connect          # (from /etc/ppp/peers/quectel-ppp)
disconnect chat -s -v -f /etc/ppp/peers/quectel-chat-disconnect        # (from /etc/ppp/peers/quectel-ppp)
nocrtscts             # (from /etc/ppp/peers/quectel-ppp)
modem         # (from /etc/ppp/peers/quectel-ppp)
asyncmap 0         # (from /etc/ppp/options)
lcp-echo-failure 4         # (from /etc/ppp/options)
lcp-echo-interval 30           # (from /etc/ppp/options)
hide-password            # (from /etc/ppp/peers/quectel-ppp)
novj       # (from /etc/ppp/peers/quectel-ppp)
novjccomp           # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-local         # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-remote           # (from /etc/ppp/peers/quectel-ppp)
ipparam 3gppp           # (from /etc/ppp/peers/quectel-ppp)
noipdefault           # (from /etc/ppp/peers/quectel-ppp)
ipcp-max-failure 30           # (from /etc/ppp/peers/quectel-ppp)
defaultroute         # (from /etc/ppp/peers/quectel-ppp)
usepeerdns         # (from /etc/ppp/peers/quectel-ppp)
noccp          # (from /etc/ppp/peers/quectel-ppp)
noipx          # (from /etc/ppp/options)
abort on (BUSY)
abort on (NO CARRIER)
abort on (NO DIALTONE)
abort on (ERROR)
abort on (NO ANSWER)
timeout set to 30 seconds
send (AT^M)
expect (OK)
^M
OK
 -- got it

send (ATE0^M)
expect (OK)
^M
^M
OK
 -- got it

send (ATI;+CSUB;+CSQ;+CPIN?;+COPS?;+CGREG?;&D2^M)
expect (OK)
^M
^M
Quectel^M
```

```
EC200T^M
Revision: EC200TCNAAR02A03M1G^M
^M
SubEdition: V01^M
^M
+CSQ: 31,99^M
^M
+CGREG: 0,1^M
^M
+CPIN: READY^M
^M
+COPS: 0,0,"CHINA MOBILE",7^M
^M
OK
 -- got it

send (AT+CGDCONT=1,"IP","3gnet",,0,0^M)
expect (OK)
^M
^M
OK
 -- got it

send (ATD*99#^M)
expect (CONNECT)
^M
^M
CONNECT
 -- got it

Script chat -s -v -f /etc/ppp/peers/quectel-chat-connect finished (pid 3893), status = 0x0
Serial connection established.
using channel 3
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x82386807> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <auth pap> <magic 0x1eae5052> <pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <auth pap> <magic 0x1eae5052> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x82386807> <pcomp> <accomp>]
sent [LCP EchoReq id=0x0 magic=0x82386807]
sent [PAP AuthReq id=0x1 user="test" password=<hidden>]
rcvd [LCP EchoRep id=0x0 magic=0x82386807 82 38 68 07]
appear to have received our own echo-reply!
rcvd [PAP AuthAck id=0x1 "" 00]
```

```
PAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x2]
sent [IPCP ConfNak id=0x2 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.194.55.145> <ms-dns1 211.138.180.2> <ms-dns2 211.138.180.3>]
sent [IPCP ConfReq id=0x2 <addr 10.194.55.145> <ms-dns1 211.138.180.2> <ms-dns2 211.138.180.3>]
rcvd [IPCP ConfReq id=0x3]
sent [IPCP ConfAck id=0x3]
rcvd [IPCP ConfAck id=0x2 <addr 10.194.55.145> <ms-dns1 211.138.180.2> <ms-dns2 211.138.180.3>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 10.194.55.145
remote IP address 10.64.64.64
primary     DNS address 211.138.180.2
secondary DNS address 211.138.180.3
Script /etc/ppp/ip-up started (pid 3905)
Script /etc/ppp/ip-up finished (pid 3905), status = 0x0
```

Now PPP call is set up successfully.

Please use the following commands to check whether the information of IP, DNS, and route in customers' system belongs to Quectel modules.

```
root@cqh6:~# ifconfig ppp0
ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
          inet 10.194.55.145  netmask 255.255.255.255  destination 10.64.64.64
          RX bytes:1057 (1.0 KiB)  TX bytes:1228 (1.1 KiB)

root@cqh6:~# cat /etc/resolv.conf
nameserver 211.138.180.2
nameserver 211.138.180.3
root@cqh6:~# ip route show
default dev ppp0 scope link
10.64.64.64 dev ppp0 proto kernel scope link src 10.194.55.145 0.0.0.0 U 0 0 0 ppp0

root@cqh6:~# ping www.baidu.com
PING www.a.shifen.com (115.239.211.112) 56(84) bytes of data.
64 bytes from 115.239.211.112: icmp_seq=1 ttl=54 time=46.4 ms
```

Following commands can be used to terminate PPPD process to disconnect a PPP call:

```
root@cqh6:~# killall pppd
Terminating on signal 15
Connect time 0.4 minutes.
Sent 0 bytes, received 0 bytes.
```

4.3. Test ECM or RNDIS

The USB Interface 0 of EC200T can be configured as ECM/RNDIS types of USB network devices.

The default USB network type is ECM.

The current USB network type of the modules can be queried and set by **AT+QCFG="usbnet"**.

It is recommend to use RNDIS driver because it has better compatibility than ECM driver.

Table 3: USB Network Type

AT+QCFG="usbnet"	USB Driver
AT+QCFG="usbnet",1	ECM Configuration option: CONFIG_USB_NET_CDCETHER Source codes file: [KERNEL]*/drivers/net/usb/cdc_ether.c*
AT+QCFG="usbnet",3	RNDIS Configuration option: CONFIG_USB_NET_RNDIS_HOST Source codes file: [KERNEL]*/drivers/net/usb/rndis_host.c*

Please follow steps below to setup data call.

Step 1: Use **AT+QICSGP** to set APN/User name/Password/APN authentication. For more details about the AT command, please refer to ***document [1]***.
Step 2: If the module registers to 2G/3G network, please use **AT+QIACT=1** to active PDP.
Step 3: Use **AT+QNETDEVCTL=1,1,1** to setup data call.
Step 4: Call DHCP tool to obtain IP and DNS. And the format of IPv4 address is 192.168.43.X.

The following displays the log information of above steps.

```
root@carl-Lenovo-ideapad-110-15ISK:~# busybox microcom /dev/ttyUSB1
at+cpin?
+CPIN: READY
OK
at+csq
+CSQ: 31,99
OK
at+qicsgp=1,1,"cmnet"
OK
at+cops?
+COPS: 0,0,"CHINA MOBILE",7
OK
```

```
at+qnetdevctl=1,1,1
OK
root@carl-Lenovo-ideapad-110-15ISK:~# busybox udhcpc -fnq -i usb0
udhcpc (v1.21.1) started
Sending discover...
Sending select for 192.168.43.100...
Lease of 192.168.43.100 obtained, lease time 86400
/etc/udhcpc/default.script: Resetting default routes
/etc/udhcpc/default.script: Adding DNS 192.168.43.1
root@carl-Lenovo-ideapad-110-15ISK:~# cat /etc/resolv.conf
nameserver 192.168.43.1
root@carl-Lenovo-ideapad-110-15ISK:~# ip route show
default via 192.168.43.1 dev usb0
192.168.43.0/24 dev usb0    proto kernel    scope link    src 192.168.43.100
root@carl-Lenovo-ideapad-110-15ISK:~#
```

5 Power Management

The Linux USB system provides two advanced power management features: USB Auto Suspend and USB Remote Wakeup. This chapter introduces how to enable these features, particularly for developers in need.

When USB communication between the USB host and the USB devices is idle for some time (for example 3 seconds), the USB host can make the USB devices enter into suspend mode automatically. This feature is called USB Auto Suspend.

USB Remote Wakeup allows a suspended USB device to remotely wake up the USB host over the USB which may also be suspended (e.g. deep sleep mode). The USB device performs an activity to wake up the USB host, then the USB host will be woken up by the remote activity.

5.1. Enable USB Auto Suspend

For USB serial driver, please add the following statements to *option_probe()* function in file *[KERNEL]/drivers/usb/serial/option.c* for enabling USB auto suspend feature.

```
static int option_probe(struct usb_serial *serial, const struct usb_device_id *id) {
    struct usb_wwan_intf_private *data;
    ……
#if 1 //Added by Quectel
//For USB Auto Suspend
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        pm_runtime_set_autosuspend_delay(&serial->dev->dev, 3000);
        usb_enable_autosuspend(serial->dev);
    }
#endif
    /* Store device id so we can use it during attach. */
    usb_set_serial_data(serial, (void *)id);
    return 0;
}
```

5.2. Enable USB Remote Wakeup

For USB serial driver, please add the following statements to *option_probe()* function in file *[KERNEL]/drivers/usb/serial/option.c* for enabling USB remote wakeup feature.

```
static int option_probe(struct usb_serial *serial, const struct usb_device_id *id) {
    struct usb_wwan_intf_private *data;
    ……
#if 1 //Added by Quectel
//For USB Remote Wakeup
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        device_init_wakeup(&serial->dev->dev, 1); //usb remote wakeup
    }
#endif
    /* Store device id so we can use it during attach. */
    usb_set_serial_data(serial, (void *)id);
    return 0;
}
```

6 FAQ and Kernel Log

6.1. Check Whether USB Driver Exists in the Module

The existence of the USB driver can be checked from the content of the directory */sys/bus/usb/drivers*. For example:

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

If USB serial driver is required, please make sure *option* exists. If ECM driver is required, please make sure *cdc_ether* exists. If RNDIS driver is required, please make sure *rndis_host* exists.

6.2. Check Whether the Module Works Well with the USB Driver

This chapter shows the kernel log about the module with the corresponding USB driver installed in the Linux operating system. Compare the kernel log in the module with that in this chapter to check whether the module works well with the USB driver.

1. For USB serial option and ECM driver: Kernel logs of different modules are almost the same except for the VID&PID information (framed in red in the following figure).

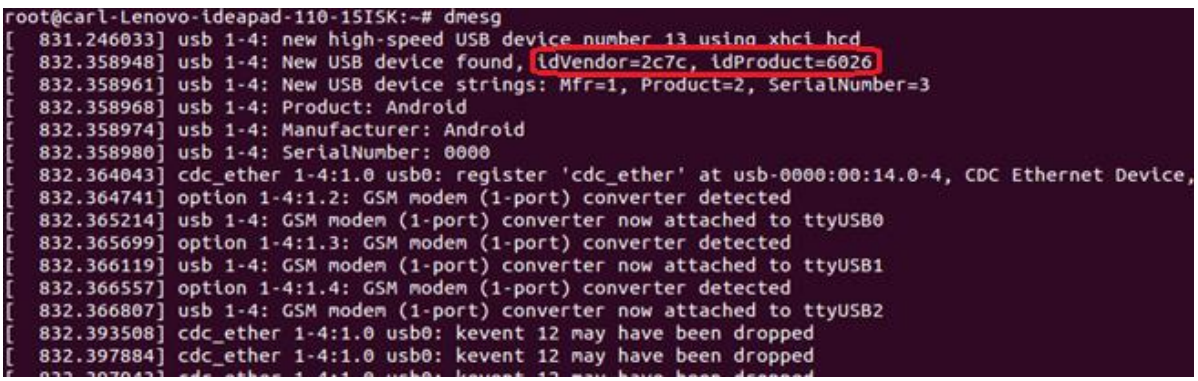

Figure 6: USB Serial and ECM for EC200T

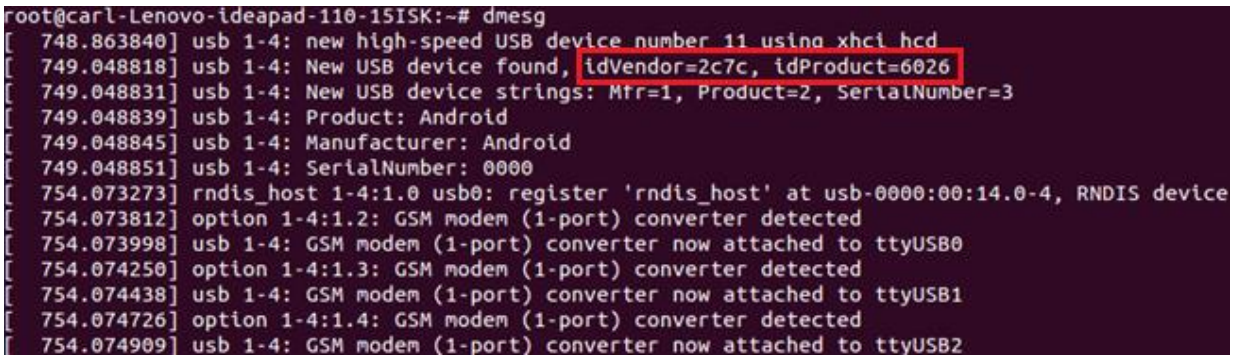

Figure 7: USB Serial and RNDIS for EC200T

7 Appendix A References

Table 4: Related Document

SN.	Document Name	Remark
[1]	Quectel_EC200T_TCP/IP_Application_Note	EC200T TCP/IP Application Note

Table 5: Terms and Abbreviations

Abbreviations	Descriptions
ATM	Asynchronous Transfer Mode
APN	Access Point Name
CDC	Communications Device Class
DNS	Domain Name System
DM	Device management
DHCP	Dynamic Host Configuration Protocol
ECM	Ethernet Networking Control Model
EHCI	Enhanced Host Controller Interface
HCD	Host Controller Driver
MCU	Microcontroller Unit
OS	Operating System
OHCI	Open Host Controller Interface
PC	Personal Computer
PID	Product ID
PPP	Point-to-Point Protocol

PPTP	Point to Point Tunneling Protocol
UART	Universal Asynchronous Receiver/Transmitter
VID	Vendor ID
URB	USB Request Block
USB	Universal Serial Bus
UHCI	Universal Host Controller Interface
NDIS	Network Driver Interface Specification
RNDIS	Remote NDIS