LTE&5G Linux USB Driver User Guide

LTE/5G Module Series

Rev. LTE&5G_Linux_USB_Driver_User_Guide_V2.0

Date: 2019-12-11

Status: Released

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.
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

About the Document

History

Revision	Date	Author	Description
1.0	2015-02-27	Joe WANG	Initial
1.1	2015-3-25	Carl YIN	Updated supported products
1.2	2015-3-30	Kent XU	Added Zero Packet feature in Section 3.2.2 and 3.3.2
1.3	2015-06-24	Carl YIN	1. Added GobiNet and QMI WWAN description in Section 3.4 and 3.5 2. Added building drivers as a kernel module in Section 3.2.4/3.3.4/3.4.3/3.5.4 3. Added power management in Chapter 4 4. Added FAQ and kernel log in Chapter 6
1.4	2015-12-16		1. Deleted Auto-Connect of GobiNet and QMI WWAN 2. Updated the usage of quectel-CM
1.5	2016-05-13	Carl YIN/ Neo HOU	Updated supported modules
1.6	2016-08-23	Kent XU	Added EC20 R2.0 in supported modules
1.7	2017-05-24	Kent XU	Added EG91/EG95/EG06/EP06/EM06/BG96 in supported modules
1.8	2017-09-01	Kent XU	Updated description of supported modules and added AG35 in supported modules.
2.0	2019-12-11	Carl YIN	1. Added applicable modules, which can be referred in Table 1. 2. Updated USB driver interface description in Table 2. 3. Updated description of USB serial option, GobiNet and QMI_WWAN drivers in Chapter 3.2, 3.3 and 3.4. 4. Added related content of testing command “AT$QCRMCALL” and protocol QMAP on GobiNet or QMI_WWAN driver in Chapter 5.4 and 5.6 as well as

			testing MBIM driver in Chapter 5.5. 5. Added FAQs in Chapter 6.

Contents

About the Document 2
Contents 4
Table Index 6
Figure Index 7

1 Introduction 8
1.1. Applicable Modules 8

2 Overview of Linux USB Driver 9

3 System Setup 11
3.1. Linux USB Driver Structure 11
3.2. USB Serial Option Driver 12
3.2.1. Add VID and PID 12
3.2.2. Add the Zero Packet Mechanism 13
3.2.3. Add Reset-resume Mechanism 14
3.2.4. Increase the Quantity and Capacity of the Bulk Out URBs 15
3.2.5. Use MBIM, GobiNet or QMI_WWAN Driver 15
3.2.6. Modify Kernel Configuration 16
3.3. GobiNet Driver 17
3.3.1. Modify Source Codes of the Driver 17
3.3.2. Modify Kernel Configuration 17
3.4. QMI_WWAN Driver 18
3.4.1. Modify Source Codes of the Driver 18
3.4.2. Modify Kernel Configuration 19
3.5. Configure Kernel to Support PPP 20
3.6. Install and Load Driver as a Kernel Module for PC in Linux 21

4 Test the Module 23
4.1. Test AT Function 23
4.2. Test PPP Function 23
4.3. Test GobiNet/QMI_WWAN Driver 26
4.4. Test "AT$QCRMCALL" on GobiNet/QMI_WWAN Driver 29
4.5. Test MBIM Driver 30
4.6. Test QMAP on GobiNet/QMI_WWAN Driver 32

5 Power Management 33
5.1. Enable USB Auto Suspend 33
5.2. Enable USB Remote Wakeup 34

6 FAQs and Kernel Log 35
6.1. How to Check Whether USB Driver Exists in the Module 35
6.2. How to Check Whether the Module Works Well with the Corresponding USB Driver 35
6.3. How to Check Which USB Driver Has Been Installed 36

7 Appendix A References.. 38

Table Index

TABLE 1: APPLICABLE MODULES 8
TABLE 2: DESCRIPTION OF USB DRIVER INTERFACE IN LINUX OPERATING SYSTEM 9
TABLE 3: DESCRIPTION OF QUECTEL-CM ARGUMENTS 27
TABLE 4: USB NETWORK TYPE 31
TABLE 5: TERMS AND ABBREVIATIONS 38

Figure Index

FIGURE 1: LINUX USB DRIVER STRUCTURE 11
FIGURE 2: CONFIGURE USB SERIAL IN KERNEL 17
FIGURE 3: CONFIGURE QMI_WWAN DRIVER IN KERNEL 20
FIGURE 4: CONFIGURE PPP IN KERNEL 21
FIGURE 5: AT COMMAND TEST RESULT 23
FIGURE 6: USB SERIAL OPTION AND GOBINET FOR RG500Q 36
FIGURE 7: USB SERIAL OPTION AND QMI_WWAN FOR RG500Q 36
FIGURE 8: USB INTERFACE AND DRIVER FOR RG500Q 37

1 Introduction

This document mainly introduces how to integrate the supported USB drivers for Quectel LTE&5G modules into the Linux operating system, and how to test the module after the USB driver is integrated successfully. The USB driver includes the USB serial option, GobiNet, QMI_WWAN, MBIM and ECM drivers. This document mainly describes the USB serial option, GobiNet and QMI_WWAN drivers in ***Chapter 3.2***, ***3.3*** and ***3.4***.

1.1. Applicable Modules

The document is applicable to the following Quectel modules.

Table 1: Applicable Modules

Module Series	Modules
LTE Standard	● EC2x: EC25/EC21/EC20 R2.0/EC20 R2.1 ● EG9x: EG91/EG95 ● EG2x-G: EG25-G/EG21-G ● EM05
Automotive	● AGxx: AG35/AG15/AG520R/AG550R
LTE-A	● Ex06: EG06/EP06/ EM06 ● Ex12: EG12/EM12 ● EG18 ● EM20
LPWA	● BGxx: BG96/BG95 1)/BG77 1)/BG600L-M3 1)/BC69 1)
5G	● Rx500Q: RG500Q/RM500Q ● Rx510Q: RG510Q/RM510Q

NOTES

1. GobiNet and QMI_WWAN cannot be ported simultaneously for Linux operating system, and this is an either-or option.
2. 1) BG95, BG77, BG600L-M3 and BC69 only support ECM driver in Linux operating system.

2 Overview of Linux USB Driver

The USB driver on Quectel LTE&5G module contains several different functional interfaces. The following table describes the interface information of different modules in Linux operating system.

Table 2: Description of USB Driver Interface in Linux Operating System

Module's VID and PID	USB Drivers	Interfaces
EC20 R2.0/EC20 R2.1/EC25/EG25-G/EM05: VID: 0x2c7c PID: 0x0125 **EC21/EG21-G:** VID: 0x2c7c PID: 0x0121 **EG91:** VID: 0x2c7c PID: 0x0191 **EG95:** VID: 0x2c7c PID: 0x0195 **BG96:** VID: 0x2c7c PID: 0x0296 **BG95/BG77/BG600L-M3/BC69:** VID: 0x2c7c PID: 0x0700 **AG35:** VID: 0x2c7c PID: 0x0435 **AG15:** VID: 0x2c7c PID: 0x0415 **AG520R:** VID: 0x2c7c PID: 0x0452 **AG550R:** VID: 0x2c7c PID: 0x0455	USB serial option	ttyUSB0, USBDM
		ttyUSB1 used for GPS NMEA message output
		ttyUSB2 used for AT command communication
		ttyUSB3 used for PPP connection or AT command communication
	GobiNet or QMI_WWAN	ethX or usbX for GobiNet. wwanX for QMI_WWAN. The module's USB interface 4 can be used as a USB network adapter.

EG06/EP06/EM06:
VID: 0x2c7c PID: 0x0306

EG12/EM12/EG18:
VID: 0x2c7c PID: 0x0512

EG20:
VID: 0x2c7c PID: 0x0620

RG500Q/RM500Q/RG510Q/RM510Q:
VID: 0x2c7c PID: 0x0800

3 System Setup

This chapter mainly describes the general structure of the USB stack in Linux and how to use USB serial option, GobiNet and QMI_WWAN 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. The data transmission between USB devices and the host is realized by the USB controller. The following picture illustrates the structure of the Linux USB driver. Linux USB host driver comprises three parts: USB host controller driver, USB core and USB device drivers.

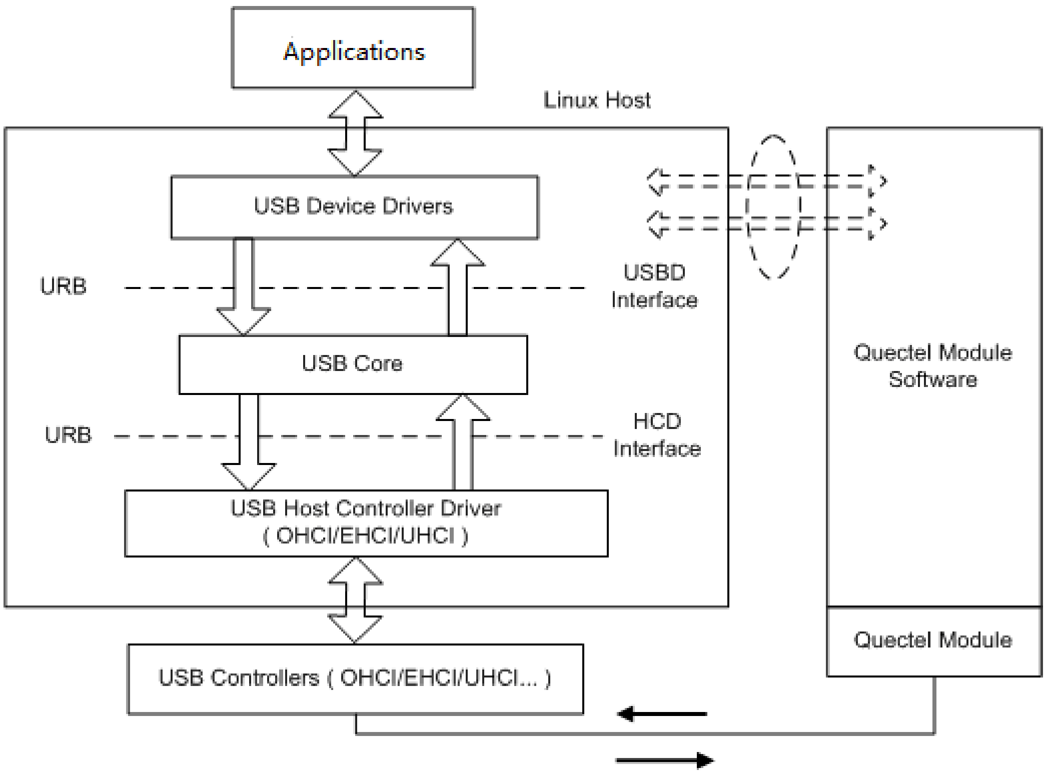


Figure 1: Linux USB Driver Structure

USB host controller driver, the bottom of the hierarchical structure, is a USB driver which interacts directly with the 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 the USB serial option driver has been installed in the module, the device files named as *ttyUSB0, ttyUSB1, ttyUSB2*, etc. will be created in directory */dev*.

The following chapters show how to integrate USB serial option driver into the 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, 0x0125) }, /* Quectel EC20 R2.0/EC20 R2.1/EC25/EG25-G/EM05 */
	{ USB_DEVICE(0x2C7C, 0x0121) }, /* Quectel EC21/EG21-G */
	{ USB_DEVICE(0x2C7C, 0x0191) }, /* Quectel EG91 */
	{ USB_DEVICE(0x2C7C, 0x0195) }, /* Quectel EG95 */
	{ USB_DEVICE(0x2C7C, 0x0306) }, /* Quectel EG06/EP06/EM06 */
	{ USB_DEVICE(0x2C7C, 0x0512) }, /* Quectel EG12/EM12/EG18 */
	{ USB_DEVICE(0x2C7C, 0x0296) }, /* Quectel BG96 */
	{ USB_DEVICE(0x2C7C, 0x0700) }, /* Quectel BG95/BG77/BG600L-M3/BC69 */
	{ USB_DEVICE(0x2C7C, 0x0435) }, /* Quectel AG35 */
	{ USB_DEVICE(0x2C7C, 0x0415) }, /* Quectel AG15 */
	{ USB_DEVICE(0x2C7C, 0x0452) }, /* Quectel AG520R */
	{ USB_DEVICE(0x2C7C, 0x0455) }, /* Quectel AG550R */
	{ USB_DEVICE(0x2C7C, 0x0620) }, /* Quectel EG20 */
	{ USB_DEVICE(0x2C7C, 0x0800) }, /* Quectel RG500Q/RM500Q/RG510Q/RM510Q */
#endif
```

3.2.2. Add the Zero Packet Mechanism

As required by the USB protocol, the mechanism for processing zero packets needs to be added during bulk-out transmission by adding the following statements.

- For Linux kernel version higher than 2.6.34, 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, 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

Some USB host controllers/USB hubs will lose power or be reset when MCU enters the Suspend/Sleep mode, and cannot be used for USB resume after MCU exits from the Suspend/Sleep mode. The reset-resume mechanism needs to be enabled by adding the following statements.

- For Linux kernel version higher than 3.4, 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, 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, the quantity and capacity of the bulk out URBs need to be increased to get faster uplink speed by adding the following statements 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 128.
```

3.2.5. Use MBIM, GobiNet or QMI_WWAN Driver

If MBIM, GobiNet or QMI_WWAN driver is required, add the following statements to prevent the module's interface 4 from being used as a USB serial device.

- For Linux kernel version higher than 2.6.30, 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
      //Quectel modules's interface 4 can be used as USB network device
       if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
               //some interfaces can be used as USB Network device (ecm, rndis, mbim)
               if (serial->interface->cur_altsetting->desc.bInterfaceClass != 0xFF) {
                       return -ENODEV;
               }
               //interface 4 can be used as USB Network device (qmi)
               else if (serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4) {
                       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, 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)) {
                      //some interfaces can be used as USB Network device (ecm, rndis, mbim)
                      if (serial->interface->cur_altsetting->desc.bInterfaceClass != 0xFF) {
                                  return -ENODEV;
                      }
                      //interface 4 can be used as USB Network device (qmi)
                      else if (serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4) {
                                  return -ENODEV;
                      }
          }
#endif
……
}
```

3.2.6. Modify Kernel Configuration

There are several mandatory selected items in the kernel configuration, please follow the steps and the corresponding commands 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
```

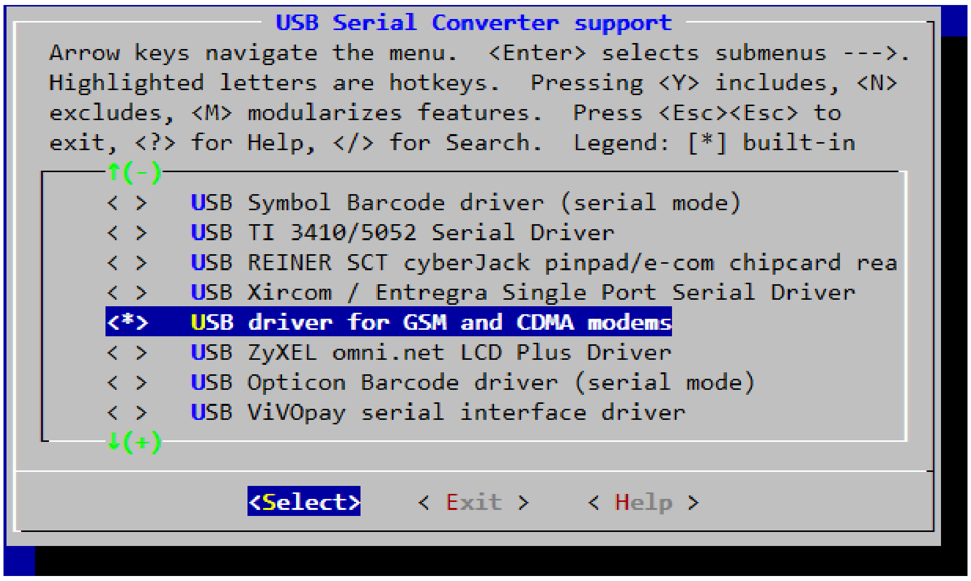
Step 3: Compile the kernel with the command below.

```
make menuconfig
```

Step 4: Enable CONFIG_USB_SERIAL_OPTION with the options below.

```
[*] Device Drivers →
      [*] USB Support →
```

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



```
                  USB Serial Converter support
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N>
  excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in
      ↑(-)
      < >   USB Symbol Barcode driver (serial mode)
      < >   USB TI 3410/5052 Serial Driver
      < >   USB REINER SCT cyberJack pinpad/e-com chipcard rea
      < >   USB Xircom / Entregra Single Port Serial Driver
      <*>   USB driver for GSM and CDMA modems
      < >   USB ZyXEL omni.net LCD Plus Driver
      < >   USB Opticon Barcode driver (serial mode)
      < >   USB ViVOpay serial interface driver
      ↓(+)

              <Select>    < Exit >    < Help >
```

Figure 2: Configure USB Serial in Kernel

3.3. GobiNet Driver

When the GobiNet driver has been installed in the module, a network device and a QMI channel will be created. The network device is named as *ethX* (*usbX* if the kernel version is 2.6.39 or lower) and the QMI channel is */dev/qcqmiX*. The network device is used for data transmission, and QMI channel is used for QMI message interaction.

The following chapters explain how to integrate the GobiNet driver into the Linux operating system.

3.3.1. Modify Source Codes of the Driver

The GobiNet driver is provided by Quectel in the form of the source file containing source codes. The source file should be copied to the file *[KERNEL]/drivers/net/usb/* (or *[KERNEL]/drivers/usb/net/* if the kernel version is lower than 2.6.22).

3.3.2. Modify Kernel Configuration

There are several mandatory selected items in the kernel configuration, please follow the steps and the corresponding commands 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 for example) with the commands 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_USBNET with the options below.

```
[*] Device Drivers →
    -*- Network device support →
        USB Network Adapters →
            {*} Multi-purpose USB Networking Framework
```

Step 5: Please add the following statements to the file *[KERNEL]/drivers/net/usb/Makefile* (or *[KERNEL]/drivers/usb/net/Makefile* if the kernel version is lower than 2.6.22).

```
obj-y += GobiNet.o
GobiNet-objs := GobiUSBNet.o QMIDevice.o QMI.o
```

3.4. QMI_WWAN Driver

When the QMI_WWAN driver has been installed in the module, a network device and a QMI channel will be created. The network device is named as *wwanX* and the QMI channel is */dev/cdc-wdmX*. The network device is used for data transmission, and QMI channel is used for QMI message interaction.

The following chapters explain how to integrate the QMI_WWAN driver.

3.4.1. Modify Source Codes of the Driver

The source file containing source codes of QMI_WWAN driver is *[KERNEL]/drivers/net/usb/qmi_wwan.c*. In order to use the QMI_WWAN driver along with the Quectel module, the source file needs certain modification.

To simplify works, Quectel provides the source file *qmi_wwan_q.c*, which can coexist with *qmi_wwan.c* and only be used for Quectel's modules. The source file *qmi_wwan_q.c* should be copied to the file *[KERNEL]/drivers/net/usb/*.

3.4.2. Modify Kernel Configuration

There are several mandatory selected items in the kernel configuration, please follow the steps and the corresponding commands 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 for example) with the command below.

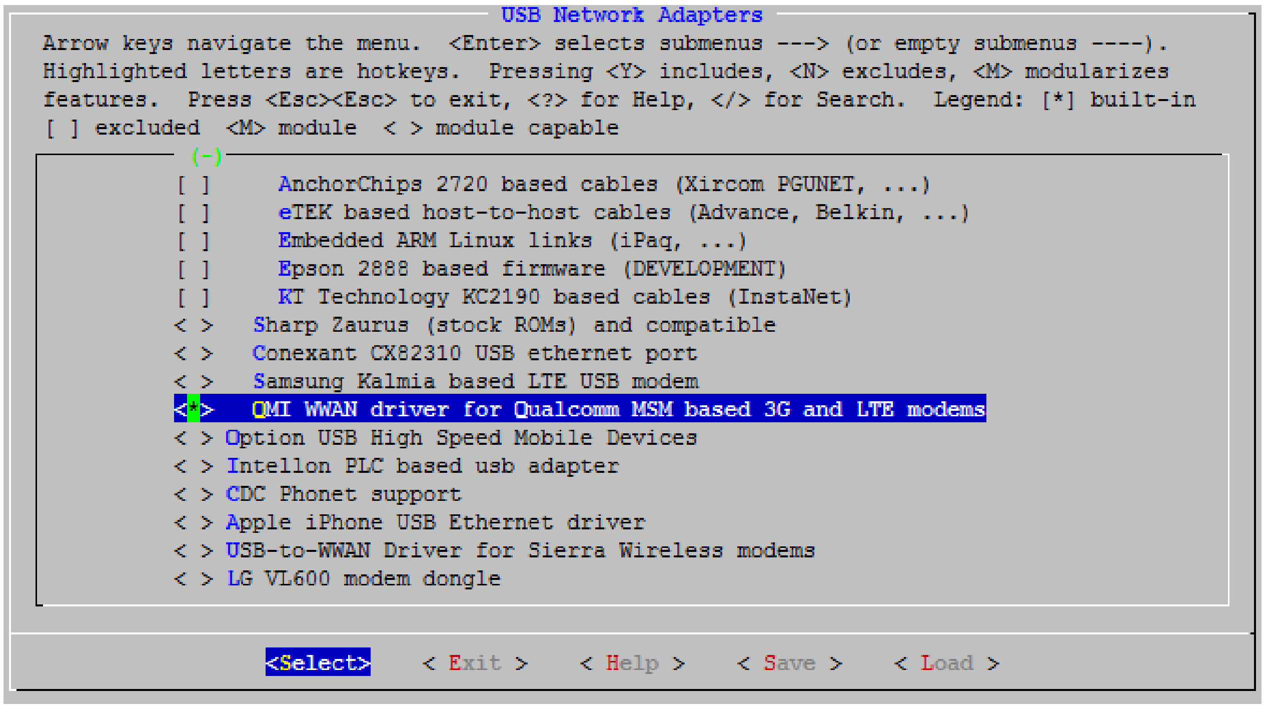
```
export ARCH=arm
export CROSS_COMPILE=arm-none-linux-gnueabi-
make bcmrpi_defconfig
```

Step 3: Compile the kernel with the command below.

```
make menuconfig
```

Step 4: Enable CONFIG_USB_NET_QMI_WWAN with the options below.

```
[*] Device Drivers →
    -*- Network device support →
        USB Network Adapters →
            {*} Multi-purpose USB Networking Framework
                <*>     QMI_WWAN driver for Qualcomm MSM based 3G and LTE modems
```



```
                         USB Network Adapters
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in
  [ ] excluded  <M> module  < > module capable
              (-)
              [ ]     AnchorChips 2720 based cables (Xircom PGUNET, ...)
              [ ]     eTEK based host-to-host cables (Advance, Belkin, ...)
              [ ]     Embedded ARM Linux links (iPaq, ...)
              [ ]     Epson 2888 based firmware (DEVELOPMENT)
              [ ]     KT Technology KC2190 based cables (InstaNet)
              < >   Sharp Zaurus (stock ROMs) and compatible
              < >   Conexant CX82310 USB ethernet port
              < >   Samsung Kalmia based LTE USB modem
              <*>   QMI WWAN driver for Qualcomm MSM based 3G and LTE modems
              < > Option USB High Speed Mobile Devices
              < > Intellon PLC based usb adapter
              < > CDC Phonet support
              < > Apple iPhone USB Ethernet driver
              < > USB-to-WWAN Driver for Sierra Wireless modems
              < > LG VL600 modem dongle

              <Select>    < Exit >    < Help >    < Save >    < Load >
```

Figure 3: Configure QMI_WWAN Driver in Kernel

Step 5: Please add the following statements to the file *[KERNEL]/drivers/net/usb/Makefile.*

```
# must insert qmi_wwan_q.o before qmi_wwan.o
obj-${CONFIG_USB_NET_QMI_WWAN} += qmi_wwan_q.o
obj-${CONFIG_USB_NET_QMI_WWAN} += qmi_wwan.o
```

3.5. Configure Kernel to Support PPP

If PPP function is used, please follow the steps and the corresponding commands below to configure the 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 the board’s “defconfig” file (taking Raspeberry Pi board for example) with the commands below.

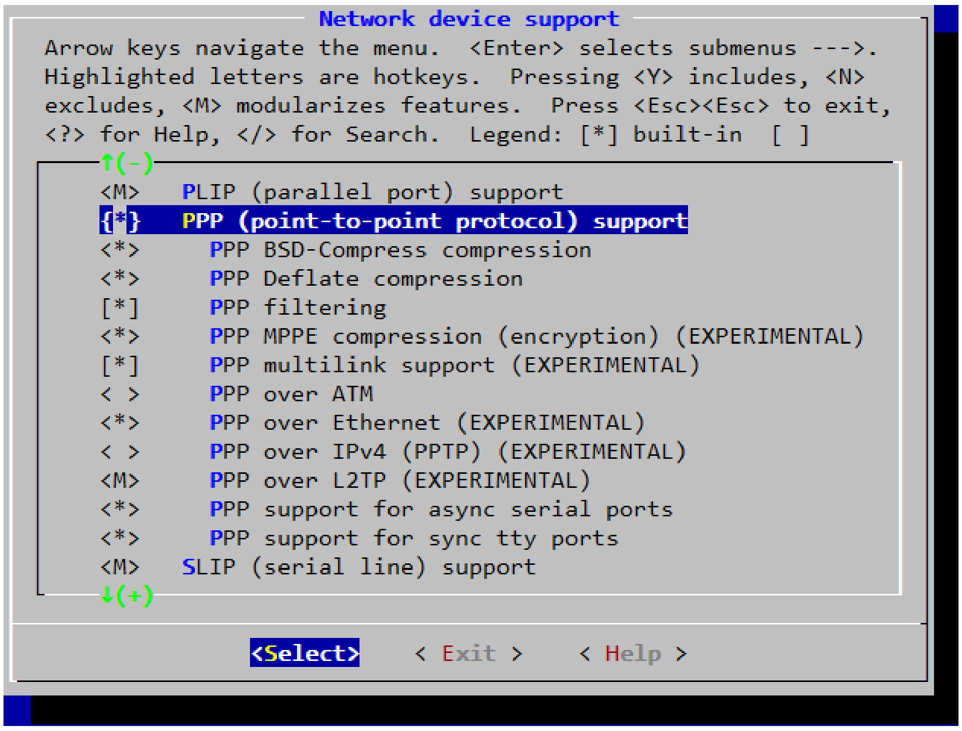
```
export ARCH=arm
export CROSS_COMPILE=arm-none-linux-gnueabi-
make bcmrpi_defconfig
```

Step 3: Compile the kernel.

```
make menuconfig
```

Step 4: Enable CONFIG_PPP_ASYNC, CONFIG_PPP_SYNC_TTY, and CONFIG_PPP_DEFLATE with the options below.

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



```
                          Network device support
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N>
  excludes, <M> modularizes features.  Press <Esc><Esc> to exit,
  <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
          ↑(-)
          <M>   PLIP (parallel port) support
          {*}   PPP (point-to-point protocol) support
          <*>     PPP BSD-Compress compression
          <*>     PPP Deflate compression
          [*]     PPP filtering
          <*>     PPP MPPE compression (encryption) (EXPERIMENTAL)
          [*]     PPP multilink support (EXPERIMENTAL)
          < >     PPP over ATM
          <*>     PPP over Ethernet (EXPERIMENTAL)
          < >     PPP over IPv4 (PPTP) (EXPERIMENTAL)
          <M>     PPP over L2TP (EXPERIMENTAL)
          <*>     PPP support for async serial ports
          <*>     PPP support for sync tty ports
          <M>   SLIP (serial line) support
          ↓(+)

                  <Select>    < Exit >    < Help >
```

Figure 4: Configure PPP in Kernel

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

For developers requiring to test Quectel modules on PC with Linux operating system like Ubuntu, Quectel can provide source files of USB serial option/GobiNet/QMI_WWAN drivers. These USB drivers can be installed and used by using the following commands and then rebooting the PC.

- Install QMI_WWAN driver.

```
carl@carl-OptiPlex-7050:~/quectel/qmi_wwan$ sudo make install
```

- Install GobiNet driver.

```
carl@carl-OptiPlex-7050:~/quectel/ GobiNet$ sudo make install
```

- Install USB serial option driver.

```
# 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. If GobiNet or QMI_WWAN driver has been installed, the USB network adapter function can also be used on the module. The following chapters explain how to test these functions.

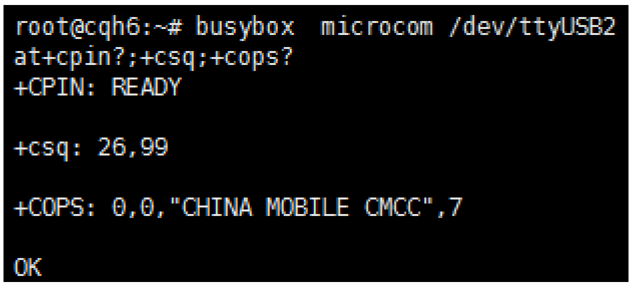
4.1. Test AT Function

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

The AT port is usually */dev/ttyUSB2*, which is the second ttyUSB port created by the USB serial option driver.

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

For example:



```
root@cqh6:~# busybox  microcom /dev/ttyUSB2
at+cpin?;+csq;+cops?
+CPIN: READY

+csq: 26,99

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

OK
```

Figure 5: AT Command Test Result

4.2. Test PPP Function

Other network cards are recommended to be used such as GobiNet or QMI_WWAN for dialing instead of PPP. PPP is more complex to use than others, will cause CPU overloaded and provide small maximum download speed.

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

1. PPPD and chat programs. If these two programs are not existed, developers can download the source codes of the two programs 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. If there is no such file, please use *linux-ppp-scripts\ip-up* provided by Quectel.

3. Three scripts named as *quectel-ppp*, *quectel-chat-connect* and *quectel-chat-disconnect*. They are provided by Quectel in directory *linux-ppp-scripts*. Developers may need to make corresponding changes based on different modules. For more information, please refer to *linux-ppp-scripts\readme*.

quectel-ppp, *quectel-chat-connect* and *quectel-chat-disconnect* should be copied to the directory */etc/ppp/peers*, then start to set up PPP call via the following command:

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

The process of PPP calling setup is shown as below:

```
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)
AT^M^M
OK
 -- got it

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

Script chat -s -v -f /etc/ppp/peers/quectel-chat-connect finished (pid 2912), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB3
```

```
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x588fbf7f> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0xea02c208> <pcomp>
<accomp>]
sent [LCP ConfAck id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0xea02c208> <pcomp>
<accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x588fbf7f> <pcomp> <accomp>]
sent [LCP EchoReq id=0x0 magic=0x588fbf7f]
rcvd [LCP DiscReq id=0x1 magic=0xea02c208]
rcvd [CHAP Challenge id=0x1 <86b3d5669380a4bcfa502b8e92a4cc93>, name =
"UMTS_CHAP_SRVR"]
sent [CHAP Response id=0x1 <9efc37eaf3dd8d819ac3e452d242e026>, name = "test"]
rcvd [LCP EchoRep id=0x0 magic=0xea02c208 58 8f bf 7f]
rcvd [CHAP Success id=0x1 ""]
CHAP authentication succeeded
CHAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
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=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.187.151.143> <ms-dns1 202.102.213.68> <ms-dns2
61.132.163.68>]
sent [IPCP ConfReq id=0x2 <addr 10.187.151.143> <ms-dns1 202.102.213.68> <ms-dns2
61.132.163.68>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfAck id=0x2 <addr 10.187.151.143> <ms-dns1 202.102.213.68> <ms-dns2
61.132.163.68>]
Could not determine remote IP address: defaulting to 10.64.64.64
not replacing default route to eth0 [172.18.112.1]
local  IP address 10.187.151.143
remote IP address 10.64.64.64
primary   DNS address 202.102.213.68
secondary DNS address 61.132.163.68
Script /etc/ppp/ip-up started (pid 2924)
Script /etc/ppp/ip-up finished (pid 2924), 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.

```
# ifconfig ppp0
ppp0       Link encap:Point-to-Point Protocol
           inet addr: 10.187.151.143  P-t-P:10.64.64.64  Mask:255.255.255.255
```

```
          UP POINTOPOINT RUNNING NOARP MULTICAST   MTU:1500   Metric:1
          RX packets:15 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:1057 (1.0 KiB)   TX bytes:1228 (1.1 KiB)

# cat /etc/resolv.conf
nameserver 61.132.163.68
nameserver 202.102.213.68

# route   -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.64.64.64     0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
0.0.0.0         0.0.0.0         0.0.0.0         U     0      0        0 ppp0

# 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=46.4ms
```

The following commands can be used to terminate the PPPD process to disconnect a PPP call:

```
# killall pppd
Terminating on signal 15
Connect time 0.4 minutes.
Sent 0 bytes, received 0 bytes.
```

4.3. Test GobiNet/QMI_WWAN Driver

Please follow the steps below to test the GobiNet or QMI_WWAN driver:

Step 1: Compile the Connect Manager program with the following commands. Quectel provides a Connect Manager program ("quectel-CM") for developers to set up data connection manually. The Connect Manager is provided in the form of source code in the directory *quectel-CM*.

- For PC Linux:

```
# make
```

- For embedded Linux:

```
# make CROSS-COMPILE=arm-none-linux-gnueabi-
```

Please replace *arm-none-linux-gnueabi-* by the cross compiler on the module.

Program “quectel-CM” will be output in this step.

Step 2: Prepare “busybox udhcpc” tool.

quectel-CM will call “busybox udhpc” to obtain IP and DNS, and “busybox udhpc” will call script file */usr/share/udhcpc/default.script* to set IP, DNS and routing table for Linux board.

The source codes of “busybox udhpc” tool can be downloaded from https://busybox.net/, then enable CONFIG_UDHCPC with the command below and copy the script file *[BUSYBOX]/examples/udhcp/simple.script* to Linux board (renamed as */usr/share/udhcpc/default.script*).

```
busybox menuconfig
```

Step 3: Use “quectel-CM” to set up a data call.

After the module is connected and the GobiNet or QMI_WWAN driver is installed successfully, a USB network adapter and a QMI channel will be created. The USB network adapter of the GobiNet driver is named as *ethX* (or *usbX* if the kernel version is 2.6.39 or lower), and the QMI channel is */dev/qcqmiX*. The USB network adapter of the QMI_WWAN driver is named as *wwanX*, and the QMI channel is */dev/cdc-wdmX*.

quectel-CM will send QMI message to the module via QMI channel to set up data connection. Please refer to the following table about the usage of quectel-CM:

Table 3: Description of quectel-CM Arguments

Number	Argument	Optional/ Non-optional	Description
1	-s <apn>	Optional	Set APN/user/password/authentication type, which could be obtained from developers’ network provider (for example: -s 3gnet carl 1234 1). For authentication type, 0 means none, 1 means PAP and 2 means CHAP. If this information has been set by **AT+CGDCONT** and **AT+QICGPS**, then this argument can be ignored.
2	-p <pincode>	Optional	Verify (U)SIM card’s PIN code if it is locked
3	-f <logfilename>	Optional	Save log message of program “quectel-CM” to file
4	-i <interface>	Optional	If there are multiple Quectel modules applied on developers’ board, the specific module can be selected to call by using the network

			adapter's name (for example: wwan0). If there is only one Quectel module applied on developers' board, then this argument can be ignored since this program supports automatic detection.
5	-4 -6	Optional	Set the type of data call: -4 means IPv4 data call (default) -6 means IPv6 data call -4 -6 means IPv4&IPv6 data calls
6	-v	Optional	Print all QMI message for debugging

The working process of quectel-CM is shown as below (taking EM12 running QMI_WWAN driver for example):

```
root@cqh6:~# ./quectel-CM/quectel-CM &
[07-03_06:56:28:172] WCDMA&LTE_QConnectManager_Linux&Android_V1.3.4
[07-03_06:56:28:172] ./quectel-CM/quectel-CM profile[1] = (null)/(null)/(null)/0, pincode = (null)
[07-03_06:56:28:174] Find /sys/bus/usb/devices/2-1.2 idVendor=2c7c idProduct=0512
[07-03_06:56:28:174] Find /sys/bus/usb/devices/2-1.2:1.4/net/wwan0
[07-03_06:56:28:174] Find usbnet_adapter = wwan0
[07-03_06:56:28:175] Find /sys/bus/usb/devices/2-1.2:1.4/usbmisc/cdc-wdm0
[07-03_06:56:28:175] Find qmichannel = /dev/cdc-wdm0
[07-03_06:56:28:197] cdc_wdm_fd = 7
[07-03_06:56:28:381] Get clientWDS = 18
[07-03_06:56:28:445] Get clientDMS = 1
[07-03_06:56:28:509] Get clientNAS = 2
[07-03_06:56:28:573] Get clientUIM = 2
[07-03_06:56:28:637] Get clientWDA = 1
[07-03_06:56:28:701] requestBaseBandVersion EM12GPAR01A06M4G
[07-03_06:56:28:957] requestGetSIMStatus SIMStatus: SIM_READY
[07-03_06:56:29:021] requestGetProfile[1] cmnet///0
[07-03_06:56:29:085] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: LTE
[07-03_06:56:29:149] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[07-03_06:56:29:277] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: LTE
[07-03_06:56:29:341] requestSetupDataCall WdsConnectionIPv4Handle: 0x127b42c0
[07-03_06:56:29:469] requestQueryDataCall IPv4ConnectionStatus: CONNECTED
[07-03_06:56:29:533] ifconfig wwan0 up
[07-03_06:56:29:543] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.27.2
udhcpc: sending discover
udhcpc: sending select for 10.170.235.201
udhcpc: lease of 10.170.235.201 obtained, lease time 7200
[07-03_06:56:29:924] /etc/udhcpc/default.script: Resetting default routes
[07-03_06:56:29:936] /etc/udhcpc/default.script: Adding DNS 211.138.180.2
```

```
[07-03_06:56:29:936] /etc/udhcpc/default.script: Adding DNS 211.138.180.3
```

Step 4: Use the following commands to check the information about IP, DNS, and route.

```
root@cqh6:~# ifconfig wwan0
wwan0: flags=4291<UP,BROADCAST,RUNNING,NOARP,MULTICAST>    mtu 1500
        inet 10.170.235.201    netmask 255.255.255.252    broadcast 10.170.235.203

root@cqh6:~# cat /etc/resolv.conf
nameserver 211.138.180.2
nameserver 211.138.180.3

root@cqh6:~# ip route show
default via 10.170.235.202 dev wwan0
10.170.235.200/30 dev wwan0 proto kernel scope link src 10.170.235.201
172.18.112.0/23 dev eth0 proto kernel scope link src 172.18.112.13

# 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=53 time=24.8 ms
```

Step 5: Use the following command to terminate the quectel-CM process to disconnect data connection:

```
root@cqh6:~# killall quectel-CM
[07-03_07:00:10:145] requestDeactivateDefaultPDP err = 0
[07-03_07:00:10:145] ifconfig wwan0 down
[07-03_07:00:10:152] ifconfig wwan0 0.0.0.0
[07-03_07:00:10:553] QmiWwanThread exit
[07-03_07:00:10:554] main exit
```

4.4. Test "AT$QCRMCALL" on GobiNet/QMI_WWAN Driver

This chapter mainly introduces how to use **AT$QCRMCALL** to set up a data call.

Although it is recommended to use QMI tools like *quectel-CM/libqmi/uqmi* to setup data call, but **AT$QCRMCALL** are preferred for some developers.
And if the developers' MCU's USB Host Controller does not fully support USB interrupt type endpoint. It has to use **AT$QCRMCALL** instead of QMI tools.

For GobiNet driver, in order to use **AT$QCRMCALL**, "qcrmcall_mode" in *GobiUSBNet.c* needs to be modified as "1". While for QMI_WWAN driver, no extra modification is required.

The following logs show how to use **AT$QCRMCALL** to set up data call. For details, please contact Quectel Technical Support Team.

```
root@imx6qdlsabresd:~# busybox microcom /dev/ttyUSB2
at+csq;+cgreg?;+cops?
+CSQ: 27,99
+CGREG: 0,1
+COPS: 0,0,"CHINA MOBILE",7
OK

AT$QCRMCALL=1,1
$QCRMCALL: 1,V4
OK

AT+QNETDEVSTATUS?
+QNETDEVSTATUS: 0,1,4,1
OK

root@imx6qdlsabresd:~# busybox udhcpc -fnq -i wwan0
udhcpc (v1.24.1) started
Sending discover...
Sending select for 10.166.47.120...
Lease of 10.166.47.120 obtained, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 211.138.180.2
/etc/udhcpc.d/50default: Adding DNS 211.138.180.3
root@imx6qdlsabresd:~#
```

4.5. Test MBIM Driver

The interface 4 of these modules can be configured as GobiNet/QMI_WWAN types of USB network devices.

Please note that for EM05, EM06, EM12, EM20, RM500Q and RM510Q modules, the default USB network type is MBIM. While for other modules, the default is GobiNet or QMI_WWAN.

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

Table 4: USB Network Type

AT+QCFG="usbnet"	USB Driver
AT+QCFG="usbnet",0	Please refer to ***Chapter 3.2*** GobiNet or ***Chapter 3.3*** QMI_WWAN
AT+QCFG="usbnet",2	CDC_MBIM Configuration option: USB_NET_CDC_MBIM Source codes file: *[KERNEL]/drivers/net/usb/cdc_mbim.c*

The USB Interface Class of MBIM is defined by USB-IF. Therefore, there is no need to insert Quectel modules' VID and PID to the source code file.

For MBIM mode, MBIM tools like "mbimcli", "umbim" or "quectel-CM", which is provided by Quectel can be used to set up a data call.

The working process of "quectel-CM" tool is shown as below (taking EM12 for example):

```
root@cqh6:~# ./quectel-CM -s ctnet &
[10-08_06:16:03:686] Quectel_QConnectManager_Linux_V1.3.9
[10-08_06:16:03:689] Find /sys/bus/usb/devices/2-1.1 idVendor=0x2c7c idProduct=0x512
[10-08_06:16:03:689] Auto find qmichannel = /dev/cdc-wdm0
[10-08_06:16:03:689] Auto find usbnet_adapter = wwan0
[10-08_06:16:03:690] Modem works in MBIM mode
[10-08_06:16:03:690] apn ctnet, user , passwd , auth 0
[10-08_06:16:03:690] IP Proto MBIMContextIPTypeIPv4
[10-08_06:16:03:691] mbim_read_thread is created
[10-08_06:16:03:700] system(ip address flush dev wwan0)=0
[10-08_06:16:03:710] system(ip link set dev wwan0 down)=0
[10-08_06:16:03:711] mbim_open_device()
[10-08_06:16:03:788] mbim_device_caps_query()
[10-08_06:16:03:853] DeviceId:        869710030002905
[10-08_06:16:03:853] FirmwareInfo: EM12GBATE1127
[10-08_06:16:03:853] HardwareInfo: EM12-G
[10-08_06:16:03:853] mbim_subscriber_status_query()
[10-08_06:16:03:917] SubscriberReadyState NotInitialized -> Initialized
[10-08_06:16:03:917] mbim_register_state_query()
[10-08_06:16:03:981] RegisterState Unknown -> Home
[10-08_06:16:03:981] mbim_packet_service_query()
[10-08_06:16:04:045] PacketServiceState Unknown -> Attached
[10-08_06:16:04:045] mbim_query_connect(sessionID=0)
[10-08_06:16:04:109] ActivationState Unknown -> Deactivated
[10-08_06:16:04:109] mbim_set_connect(onoff=1, sessionID=0)
[10-08_06:16:04:333] ActivationState Deactivated -> Activated
[10-08_06:16:04:333] mbim_ip_config(sessionID=0)
```

```
[10-08_06:16:04:397] < SessionId = 0
[10-08_06:16:04:397] < IPv4ConfigurationAvailable = 0xf
[10-08_06:16:04:397] < IPv6ConfigurationAvailable = 0x0
[10-08_06:16:04:398] < IPv4AddressCount = 0x1
[10-08_06:16:04:398] < IPv4AddressOffset = 0x3c
[10-08_06:16:04:398] < IPv6AddressCount = 0x0
[10-08_06:16:04:398] < IPv6AddressOffset = 0x0
[10-08_06:16:04:399] < IPv4 = 10.172.205.26/30
[10-08_06:16:04:399] < gw = 10.172.205.25
[10-08_06:16:04:399] < dns1 = 202.102.213.68
[10-08_06:16:04:399] < dns2 = 61.132.163.68
[10-08_06:16:04:399] < ipv4 mtu = 1500
[10-08_06:16:04:410] system(ip link set dev wwan0 up)=0
[10-08_06:16:04:420] system(ip -4 address flush dev wwan0)=0
[10-08_06:16:04:431] system(ip -4 address add 10.172.205.26/30 dev wwan0)=0
[10-08_06:16:04:441] system(ip -4 route add default via 10.172.205.25 dev wwan0)=0
[10-08_06:16:04:451] system(ip -4 link set dev wwan0 mtu 1500)=0
```

4.6. Test QMAP on GobiNet/QMI_WWAN Driver

This chapter introduces how to test the QMAP (Qualcomm Multiplexing and Aggregation protocol) on GobiNet or QMI_WWAN driver, especially catering for developers using GobiNet or QMI_WWAN driver and requiring QMAP.

When using GobiNet or QMI_WWAN driver, only one physical network card can be created by default, so only one PDN data call can be set up. However, through using the multiplexing protocol, multiple virtual network cards can be created over one physical network card, so multiple PDN data calls can be set up.

When using GobiNet or QMI_WWAN driver, only one IP Packet in one URB can be transferred, so when there are high throughput and frequent URB interrupts, the Host CPU will be overloaded. However, the aggregation protocol can be used to transfer multiple IP Packets in one URB with increased throughput by reducing the number of URB interrupts.

If multiplexing or aggregation protocol is needed, please contact Quectel Technical Supports support@quectel.com.

5 Power Management

The USB system in Linux 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 Suspend mode automatically. This feature is called USB Auto Suspend.

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

5.1. Enable USB Auto Suspend

For USB serial option driver, please add the following statements to *option_probe()* function in the file *[KERNEL]/drivers/usb/serial/option.c* to enable 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 option driver, please add the following statements to *option_probe()* function in the file *[KERNEL]/drivers/usb/serial/option.c* to enable 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 FAQs and Kernel Log

6.1. How to 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
cdc_acm cdc_wdm ftdi_sio GobiNet hub option qmi_wwan usb usbfs usbhid usbserial usbserial_generic
```

If the USB serial option driver is required, please make sure *option* exists in the content the content of the directory */sys/bus/usb/drivers*.

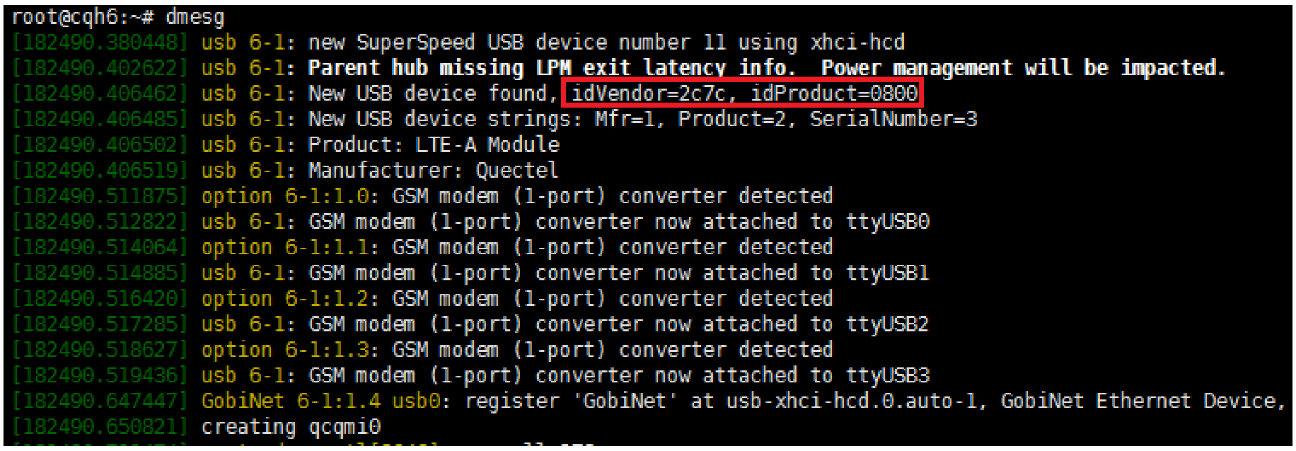
Similarly, If GobiNet driver is required, please make sure *GobiNet* exists.

And if QMI_WWAN driver is required, please make sure *qmi_wwan* exists.

6.2. How to Check Whether the Module Works Well with the Corresponding USB Driver

This chapter shows the kernel log about the module with the corresponding USB driver installed in the Linux operating system. If the module does not work well, please compare the kernel log in the module with that in this chapter to help with troubleshooting.

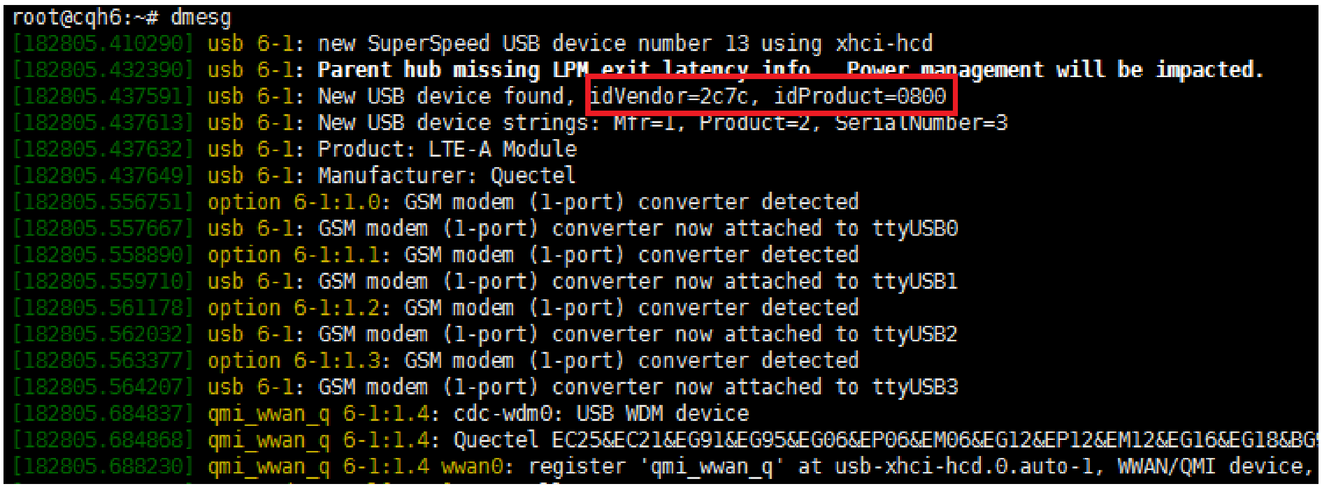
- For USB serial option and GobiNet driver: Kernel logs of different modules are almost the same except for the VID&PID information (framed in red in the following figure).



```
root@cqh6:~# dmesg
[182490.380448] usb 6-1: new SuperSpeed USB device number 11 using xhci-hcd
[182490.402622] usb 6-1: Parent hub missing LPM exit latency info.  Power management will be impacted.
[182490.406462] usb 6-1: New USB device found, idVendor=2c7c, idProduct=0800
[182490.406485] usb 6-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[182490.406502] usb 6-1: Product: LTE-A Module
[182490.406519] usb 6-1: Manufacturer: Quectel
[182490.511875] option 6-1:1.0: GSM modem (1-port) converter detected
[182490.512822] usb 6-1: GSM modem (1-port) converter now attached to ttyUSB0
[182490.514064] option 6-1:1.1: GSM modem (1-port) converter detected
[182490.514885] usb 6-1: GSM modem (1-port) converter now attached to ttyUSB1
[182490.516420] option 6-1:1.2: GSM modem (1-port) converter detected
[182490.517285] usb 6-1: GSM modem (1-port) converter now attached to ttyUSB2
[182490.518627] option 6-1:1.3: GSM modem (1-port) converter detected
[182490.519436] usb 6-1: GSM modem (1-port) converter now attached to ttyUSB3
[182490.647447] GobiNet 6-1:1.4 usb0: register 'GobiNet' at usb-xhci-hcd.0.auto-1, GobiNet Ethernet Device,
[182490.650821] creating qcqmi0
```

Figure 6: USB Serial Option and GobiNet for RG500Q

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

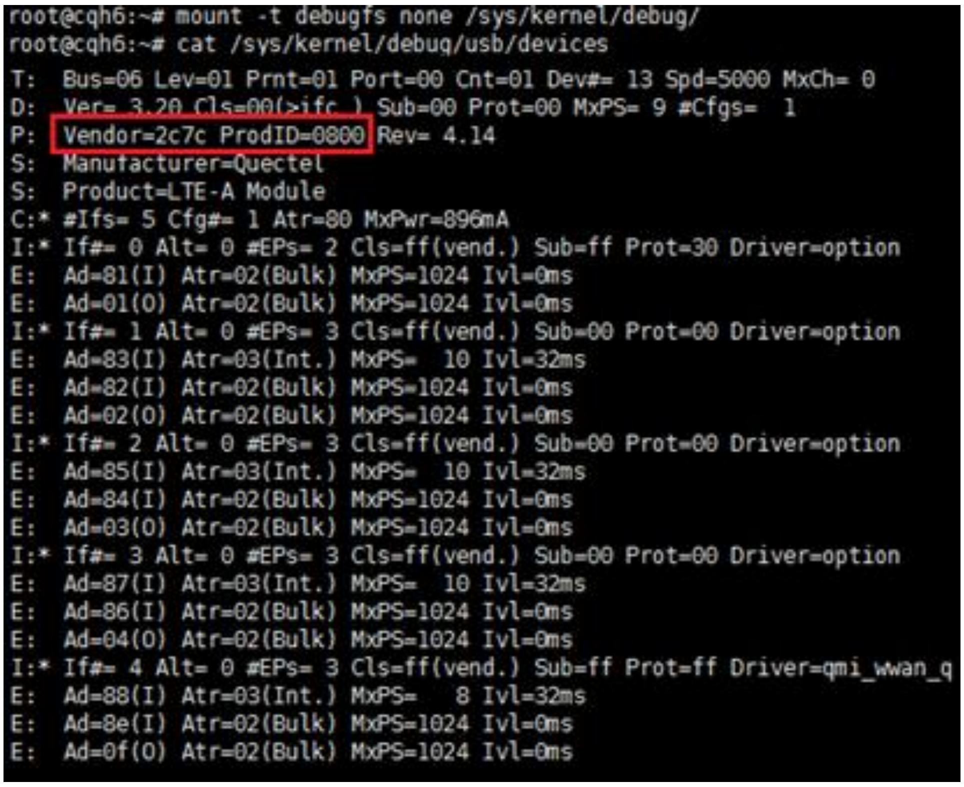


```
root@cqh6:~# dmesg
[182805.410290] usb 6-1: new SuperSpeed USB device number 13 using xhci-hcd
[182805.432390] usb 6-1: Parent hub missing LPM exit latency info.  Power management will be impacted.
[182805.437591] usb 6-1: New USB device found, idVendor=2c7c, idProduct=0800
[182805.437613] usb 6-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[182805.437632] usb 6-1: Product: LTE-A Module
[182805.437649] usb 6-1: Manufacturer: Quectel
[182805.556751] option 6-1:1.0: GSM modem (1-port) converter detected
[182805.557667] usb 6-1: GSM modem (1-port) converter now attached to ttyUSB0
[182805.558890] option 6-1:1.1: GSM modem (1-port) converter detected
[182805.559710] usb 6-1: GSM modem (1-port) converter now attached to ttyUSB1
[182805.561178] option 6-1:1.2: GSM modem (1-port) converter detected
[182805.562032] usb 6-1: GSM modem (1-port) converter now attached to ttyUSB2
[182805.563377] option 6-1:1.3: GSM modem (1-port) converter detected
[182805.564207] usb 6-1: GSM modem (1-port) converter now attached to ttyUSB3
[182805.684837] qmi_wwan_q 6-1:1.4: cdc-wdm0: USB WDM device
[182805.684868] qmi_wwan_q 6-1:1.4: Quectel EC25&EC21&EG91&EG95&EG06&EP06&EM06&EG12&EP12&EM12&EG16&EG18&BG
[182805.688230] qmi_wwan_q 6-1:1.4 wwan0: register 'qmi_wwan_q' at usb-xhci-hcd.0.auto-1, WWAN/QMI device,
```

Figure 7: USB Serial Option and QMI_WWAN for RG500Q

6.3. How to Check Which USB Driver Has Been Installed

This chapter shows how to query which USB driver that the USB interface of the Quectel module is attached to. The USB driver's name is identified by the keyword "Driver=". If "Driver=none" is shown, the reason may be that the corresponding configuration item is not enabled in customers' kernel configuration, or the VID and PID of Quectel modules are not inserted to the corresponding USB driver source files. In such a case, please follow the steps mentioned in Chapter 3 to check again.



```
root@cqh6:~# mount -t debugfs none /sys/kernel/debug/
root@cqh6:~# cat /sys/kernel/debug/usb/devices
T:  Bus=06 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#= 13 Spd=5000 MxCh= 0
D:  Ver= 3.20 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 9 #Cfgs=  1
P:  Vendor=2c7c ProdID=0800 Rev= 4.14
S:  Manufacturer=Quectel
S:  Product=LTE-A Module
C:* #Ifs= 5 Cfg#= 1 Atr=80 MxPwr=896mA
I:* If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=30 Driver=option
E:  Ad=81(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=01(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
I:* If#= 1 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=83(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=82(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=02(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
I:* If#= 2 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=85(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=84(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=03(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
I:* If#= 3 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=87(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=86(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=04(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
I:* If#= 4 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=ff Driver=qmi_wwan_q
E:  Ad=88(I) Atr=03(Int.) MxPS=   8 Ivl=32ms
E:  Ad=8e(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=0f(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
```

Figure 8: USB Interface and Driver for RG500Q

7 Appendix A References

Table 5: Terms and Abbreviations

Abbreviations	Descriptions
APN	Access Point Name
DNS	Domain Name System
EHCI	Enhanced Host Controller Interface
GPS	Global Positioning System
HCD	Host Controller Driver
MBIM	Mobile Interface Broadband Model
NDIS	Network Driver Interface Specification
NMEA	National Marine Electronics Association
OHCI	Open Host Controller Interface
OS	Operating System
PID	Product ID
PPP	Point to Point Protocol
QMAP	Qualcomm Multiplexing and Aggregation Protocol
QMI	Qualcomm Messaging Interface
UHCI	Universal Host Controller Interface
URB	USB Request Block
USB	Universal Serial Bus
VID	Vendor ID