UMTS LTE 5G Linux USB Driver User Guide

UMTS/HSPA+/LTE/5G Module Series

Version: 3.1

Date: 2023-02-01

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

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

Or our local offices. 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 us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an "as available" basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	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.
3.0	2022-03-18	Carl YIN	1. Added applicable modules. 2. Updated the overview of Linux USB driver (Chapter 2). 3. Updated the steps of integrating USB serial option driver into the Linux operating system (Chapter 3.2). 4. Updated the kernel configuration modification of GobiNet driver (Chapter 3.3.2). 5. Updated the kernel configuration modification of QMI_WWAN_Q (Chapter 3.4.2). 6. Added the description of ACM/ECM/RNDIS/NCM/MBIM drivers (Chapter 3.5). 7. Updated the information about how to configure the kernel to support PPP (Chapter 3.6). 8. Added the steps of how to configure the kernel (Chapter 3.7). 9. Updated AT command testing (Chapter 4.1). 10. Removed the description of quectel-CM (Chapter 4.3). 11. Added the information about how to test ECM/RNDIS/NCM/MBIM drivers (Chapter 4.6). 12. Updated the example (Chapter 6.1).
3.1	2023-02-01	Jerry MENG	1. Updated applicable modules: Added applicable modules EG25-GL, EG912N-EN, EG912U-GL, EG060V-EA, EG060W-EA, RG500U series, RG200U-CN, RM500U-CN, EM120K-GL, EM060K series, RG520F series, RG520N series, RM520N series, RG530F series, RM530N-GL, EG800Q-EU and RG500L series; Updated EG915N-EU to EG915N series and updated AG550Q to AG55xQ series; Deleted AG510R series, EC200T series and UC200T series. 2. Updated the module functionalities implemented by Linux system (Chapter 2). 3. Updated the name of network device to usbX (Chapter 3.3). 4. Updated the commands used to install and use USB drivers (Chapter 3.8). 5. Updated data call methods for modules with ECM, RNDIS and NCM drivers (Chapter 4.6).

Contents

Table Index

Figure Index

1 Introduction

This document outlines how to port the supported USB drivers for Quectel UMTS&LTE&5G modules into the Linux operating system, and how to test the module after the USB driver is integrated successfully. USB drivers include:

- USB serial drivers like option and ACM
- USBNet drivers like GobiNet, QMI_WWAN, MBIM, NCM, RNDIS and ECM

2 Overview of Linux USB Drivers

Quectel UMTS&LTE&5G modules are USB composite devices with multiple USB interfaces. Each USB interface supports different functionalities, which are implemented by loading different USB interface drivers. After a driver is loaded successfully, the corresponding device node is generated, which can be used by the Linux system to implement the module functionalities, such as AT command, GNSS, DIAG, log and USB network adapter.

The following table describes the USB interface information of different modules in the Linux system, including USB driver, interface number, device name and interface function.

You can obtain the corresponding VID, PID and interface information of the relevant model, and then port the USB interface driver listed in the following table.

Table 1: Applicable Modules and USB Interface Information

Module VID and PID	USB Drivers	Interface Number	Device Names	Functions
EC20-CE/ **EC25 series/** **EG25-G/** **EG25-GL/** **EM05 series:** VID: 0x2c7c PID: 0x0125 **EC21 series/** **EG21-G:** VID: 0x2c7c PID: 0x0121 **EG91 series:**	USB serial option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	AT command
		3	/dev/ttyUSB3	Modem
	GobiNet	4	usb0 /dev/qcqmi0	USB network adapter Configure the type of USBnet interface as RmNet by **AT+QCFG="usbnet",0**.

VID: 0x2c7c PID: 0x0191 **EG95 series:** VID: 0x2c7c PID: 0x0195 **AG35 series:** VID: 0x2c7c PID: 0x0435 **EG06 series/** **EP06 series/** **EM06 series:** VID: 0x2c7c PID: 0x0306 **EG12 series/** **EM12-G/** **EG18 series:** VID: 0x2c7c PID: 0x0512 **EG512R-EA/** **EM160R-GL/** **EM120R-GL/** **EM121R-GL:** VID: 0x2c7c PID: 0x0620 **RG500Q series/** **RM500Q series/** **RG501Q-EU/** **RG502Q series/** **RM502Q-AE/** **RM505Q-AE/** **RM510Q-GL:** VID: 0x2c7c PID: 0x0800	QMI_WWAN	4	wwan0 /dev/cdc-wdm0	USB network adapter Configure the type of USBnet interface as RmNet by **AT+QCFG="usbnet",0**.
	MBIM	4	wwan0 /dev/cdc-wdm0	USB network adapter Configure the type of USBnet interface as MBIM by **AT+QCFG="usbnet",2**.
		5		
BG95 series/ **BG77/** **BG600L-M3:** VID: 0x2c7c PID: 0x0700	USB serial option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	Modem

		4	/dev/ttyUSB3	Configure USB composition ID as Modem interface mode by **AT+QCFGEXT="usbnet","modem"**. Corresponds to Modem USB combination: USB DM + NEMA + Modem + Modem.
	ECM	3 4	usb0	Configure USB composition as ECM interface mode by **AT+QCFGEXT="usbnet","ecm"**. Corresponds to ECM USB combination: USB DM + NEMA + Modem + ECM.
BG96: VID: 0x2c7c PID: 0x0296	USB serial option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	AT command
		3	/dev/ttyUSB3	Modem
	QMI_WWAN	4	wwan0 /dev/cdc-wdm0	RmNet
EC200S series/ EG912N-EN/ EG915N series: VID: 0x2c7c PID: 0x6002 **EC200A series (RTOS):** VID: 0x02c7c PID: 0x6005 **UC200A-GL:** VID: 0x02c7c PID: 0x6006 **EG912Y series:** VID: 0x02c7c PID: 0x6001	ECM/ RNDIS	0 1	usb0	Configure the type of USBnet interface as ECM via **AT+QCFG="usbnet",1**. Configure the type of USBnet interface as RNDIS by **AT+QCFG="usbnet",3**.
	USB serial option	2	/dev/ttyUSB0	DIAG
		3	/dev/ttyUSB1	AT command
		4	/dev/ttyUSB2	Modem
EC200U series/	ECM/	0	usb0	Configure the type of USBnet interface

EG912U-GL/ EG915U series: VID: 0x2c7c PID: 0x0901	RNDIS	1		as ECM by **AT+QCFG="usbnet",1**. Configure the type of USBnet interface as RNDIS by **AT+QCFG="usbnet",3**.
	USB serial option	2	/dev/ttyUSB0	AT command
		3	/dev/ttyUSB1	DIAG
		4	/dev/ttyUSB2	MOS
		5	/dev/ttyUSB3	CP log
		6	/dev/ttyUSB4	AP log
		7	/dev/ttyUSB5	Modem
		8	/dev/ttyUSB6	GNSS
EG060V-EA/ EG060W-EA: VID: 0x2c7c PID: 0x6004 **EC200A series (Linux QuecOpen):** VID: 0x2c7c PID: 0x6005	ECM/ RNDIS/ NCM	0	usb0	Configure the type of USBnet interface as ECM by **AT+QCFG="usbnet",1**. Configure the type of USBnet interface as RNDIS by **AT+QCFG="usbnet",3**. Configure the type of USBnet interface as NCM by **AT+QCFG="usbnet",4**.
		1		
	USB serial option	2	/dev/ttyUSB0	DIAG
		3	/dev/ttyUSB1	AT command
		4	/dev/ttyUSB2	Modem
RG500U series/ RG200U-CN/ RM500U-CN: VID: 0x2c7c PID: 0x0900	ECM/ RNDIS/ NCM	0	usb0	Configure the type of USBnet interface as ECM by **AT+QCFG="usbnet",1**. Configure the type of USBnet interface as RNDIS by **AT+QCFG="usbnet",3**. Configure the type of USBnet interface as NCM by **AT+QCFG="usbnet",5**.
		1		
	USB serial option	2	/dev/ttyUSB0	DIAG
		3	/dev/ttyUSB1	Log
		4	/dev/ttyUSB2	AT command
		5	/dev/ttyUSB3	Modem

		6	/dev/ttyUSB4	GNSS
EG065K series/ EG060K series/ EG120K series/ EM120K-GL/ EM060K series: VID: 0x2c7c PID: 0x030b	USB serial option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	AT command
		3	/dev/ttyUSB3	Modem
RG520F series/RG520N series/ RM520N series/ RG530F series/ RM530N-GL: VID: 0x2c7c PID: 0x0801	QMI_WWAN	4	wwan0 /dev/cdc-wdm0	Configure the type of USBnet interface as RmNet by **AT+QCFG="usbnet",0**.
	MBIM	8	wwan0 /dev/cdc-wdm0	Configure the type of USBnet interface as MBIM by **AT+QCFG="usbnet",2**.
		9		
	ECM	10	usb0	Configure the type of USBnet interface as ECM by **AT+QCFG="usbnet",1**.
		11		
EG800Q-EU: VID: 0x2c7c PID: 0x6007	ECM/ RNDIS	0	usb0	Configure the type of USBnet interface as ECM by **AT+QCFG="usbnet",1**. Configure the type of USBnet interface as RNDIS by **AT+QCFG="usbnet",3**.
		1		
	ACM	2	/dev/ttyACM0	AT command
		3		
		4	/dev/ttyACM1	Log
		5		
		6	/dev/ttyACM2	modem
		7		
RG500L series: VID: 0x2c7c PID: 0x7003	RNDIS	0	usb0	RNDIS
		1		
	ACM	2	/dev/ttyACM0	Modem AT command
		3		
		4	/dev/ttyACM1	Modem ELT
		5		

AG520R series: VID: 0x2c7c PID: 0x0452 **AG55xQ series:** VID: 0x2c7c PID: 0x0455	USB serial option	0	/dev/ttyUSB0	DIAG
	QMI_WWAN	2	wwan0 /dev/cdc-wdm0	Configure the type of USBnet interface as RmNet by **AT+QCFG="usbnet",0**.
	GobiNet	2	usb0 /dev/qcqmi0	
	ECM	4	usb0	Configure the type of USBnet interface as ECM by **AT+QCFG="usbnet",1**.
		5		
	USB serial option	6	/dev/ttyUSB1	GNSS
		7	/dev/ttyUSB2	AT command
		8	/dev/ttyUSB3	Modem
	RNDIS	9	usb0	Configure the type of USBnet interface as RNDIS by **AT+QCFG="usbnet",3**.
		10		
	MBIM	11	wwan0 /dev/cdc-wdm0	Configure the type of USBnet interface as MBIM by **AT+QCFG="usbnet",2**.
		12		

NOTE

1. GobiNet and QMI_WWAN can be ported simultaneously for Linux operating system, but only one of them can work at a time.
2. The device names of the modules are not fixed. If no other USB serial device is connected to user's system, the device names of the modules start from /dev/ttyUSB0 as shown above; If another USB serial device is connected to the user's system, the device names of the modules are determined by the number of device nodes generated by the USB serial device. For example, if a USB serial device is connected to user's system and generates one device node, /dev/ttyUSB0 is occupied by USB serial device, then the device name of current module starts from /dev/ttyUSB1.
3. For more details about **AT+QCFG**, see ***document [2]***.
4. For more details about **AT+QCFGEXT**, see ***document [3]***.

3 System Setup

This chapter describes the general structure of the USB stack in Linux, as well as how to use, compile and load USB 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 figure 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.

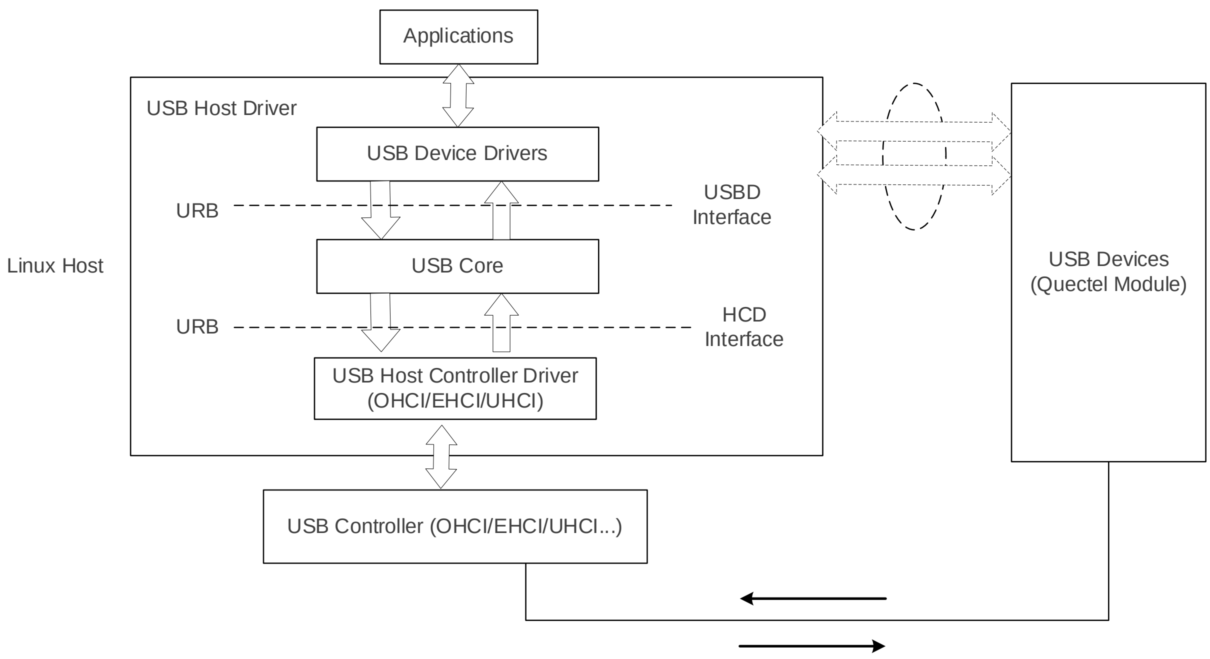

Figure 1: Linux USB Driver Structure

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

USB core, the core of the whole USB host driver, is used for managing 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 provide the interfaces for accessing specific USB devices.

3.2. USB Serial Option Driver

Once the module loads the USB serial option driver successfully, the device files named *ttyUSB0, ttyUSB1, ttyUSB2* and so on are created in directory */dev*.

The following chapters show how to port USB serial option driver into the Linux operating system.

3.2.1. Add VID and PID

To identify the module, add the module VID and PID information to the *[KERNEL]/drivers/usb/serial/option.c* file and the corresponding VID and PID can be obtained in ***Table 1***.

EC25 series Example:

```
static const struct usb_device_id option_ids[] = {
#if 1 //Added by Quectel
	{ USB_DEVICE(0x2C7C, 0x0125) },
#endif
```

3.2.2. Use USBNet Driver

The configuration in ***Chapter 3.2.1*** makes all USB interfaces of the module attach to USB serial option driver, which prevents USBNet driver interfaces from working. You can add the following statements to prevent USBNet driver interfaces from attaching to USB serial option driver.

- For Linux kernel versions beyond 2.6.30, you can add the following statements to the *[KERNEL]/drivers/usb/serial/option.c* file.

```
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);
```

```
                struct usb_interface_descriptor *intf = &serial->interface->cur_altsetting->desc;

                if (intf->bInterfaceClass != 0xFF || intf->bInterfaceSubClass == 0x42) {
                        //ECM, RNDIS, NCM, MBIM, ACM, UAC, ADB
                        return -ENODEV;
                }

                if ((idProduct&0xF000) == 0x0000) {
                        //MDM interface 4 is QMI
                        if (intf->bInterfaceNumber == 4 && intf->bNumEndpoints == 3
                                && intf->bInterfaceSubClass == 0xFF && intf->bInterfaceProtocol
== 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 versions before 2.6.31, you can add the following statements to the *[KERNEL]/drivers/usb/serial/option.c* file.

```
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);
                struct usb_interface_descriptor *intf = &serial->interface->cur_altsetting->desc;

                if (intf->bInterfaceClass != 0xFF || intf->bInterfaceSubClass == 0x42) {
                        //ECM, RNDIS, NCM, MBIM, ACM, UAC, ADB
                        return -ENODEV;
                }

                if ((idProduct&0xF000) == 0x0000) {
                        //MDM interface 4 is QMI
                        if (intf->bInterfaceNumber == 4 && intf->bNumEndpoints == 3
                                && intf->bInterfaceSubClass == 0xFF && intf->bInterfaceProtocol
== 0xFF)
                                return -ENODEV;
```

```
                    }
          }
#endif
……
}
```

3.2.3. Modify Kernel Configuration

You need to enable the following configuration items. See ***Chapter 3.7*** for the explanation how to configure the kernel.

```
CONFIG_USB_SERIAL
CONFIG_USB_SERIAL_WWAN
CONFIG_USB_SERIAL_OPTION
```

3.2.4. Add 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 beyond 2.6.34, you need to add the following statements to the *[KERNEL]/drivers/usb/serial/usb_wwan.c* file.

```
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 below 2.6.35, you need to add the following statements to the *[KERNEL]/drivers/usb/serial/option.c* file.

```
/* 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;
}
```

3.2.5. Add Reset-Resume Mechanism

When MCU enters the Suspend/Sleep mode, some USB host controllers/USB hubs will lose power or reset and cannot be used for module 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 beyond 3.4, you need to add the following statements to the *[KERNEL]/drivers/usb/serial/option.c* file.

```
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 below 3.5, you need to add the following statements to the *[KERNEL]/drivers/usb/serial/usb-serial.c file.*

```
/* 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,
};
```

3.2.6. Increase Quantity and Capacity of Bulk Out URBs

For Linux kernel version below 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 *[KERNEL]/drivers/usb/serial/option.c* file.

```
#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.
```

3.3. GobiNet Driver

After the module loads the GobiNet driver successfully, a network device and a QMI device node are created. The network device is named *usbX* and the QMI device node is */dev/qcqmiX*. The network device is used for data transmission, and QMI device node is used for QMI message interaction.

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

3.3.1. Modify Driver Source Codes

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

3.3.2. Modify Kernel Configuration

You need to enable the following configuration item first. See ***Chapter 3.7*** for the explanation how to configure the kernel.

```
CONFIG_USB_NET_DRIVERS
CONFIG_USB_USBNET
```

Then you can add the following statements to *[KERNEL]/drivers/net/usb/Makefile* (or *[KERNEL]/drivers/usb/net/Makefile* if the kernel version is below 2.6.22).

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

3.4. QMI_WWAN Driver

After the module loads the QMI_WWAN driver successfully, a network device and a QMI device node are created. The network device is named *wwanX* and the QMI device node is */dev/cdc-wdmX*. The network device is used for data transmission, and QMI device node is used for QMI message interaction.

The following chapters explain how to port the QMI_WWAN driver into the Linux operating system.

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*. To use the QMI_WWAN driver along with the Quectel module, the source file needs to be modified.

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 *[KERNEL]/drivers/net/usb/*.

3.4.2. Modify Kernel Configuration

Enable the following configuration items first. See ***Chapter 3.7*** for the explanation how to configure the kernel.

```
CONFIG_USB_NET_DRIVERS
CONFIG_USB_USBNET
CONFIG_USB_NET_QMI_WWAN
CONFIG_USB_WDM
```

Then add the following statements to *[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. ACM/ECM/RNDIS/NCM/MBIM Driver

ACM, ECM, RNDIS, NCM and MBIM drivers are USB interface class drivers. The Linux system automatically attaches to the corresponding diver according to the interface class, sub-class and protocol. These drivers are available in the upstream Linux releases if you use Linux distribution such as Ubuntu and Fedora. The drivers are automatically loaded when the module is connected to the Linux PC through the USB interface. If you use an embedded system, you just need to enable the corresponding configuration items.

The configuration items to be enabled for each driver:

Table 2: Configuration Items for USB Class Drivers

USB Driver	Configuration Item	Source File
ACM	CONFIG_USB_ACM	*[KERNEL]/drivers/net/usb/cdc-acm.c*
ECM	CONFIG_USB_NET_DRIVERS CONFIG_USB_USBNET CONFIG_USB_NET_CDCETHER	*[KERNEL]/drivers/net/usb/cdc_ether.c*
RNDIS	CONFIG_USB_NET_DRIVERS CONFIG_USB_USBNET CONFIG_USB_NET_RNDIS_HOST	*[KERNEL]/drivers/net/usb/rndis_host.c*
NCM	CONFIG_USB_NET_DRIVERS CONFIG_USB_USBNET CONFIG_USB_NET_CDC_NCM	*[KERNEL]/drivers/net/usb/cdc_ncm.c*
MBIM	CONFIG_USB_NET_DRIVERS CONFIG_USB_USBNET CONFIG_USB_NET_CDC_MBIM	*[KERNEL]*/drivers/net/usb/cdc_mbim.c

3.6. How to Support PPP

Before using PPP function, you need to enable the following configuration items to configure the kernel to support PPP. See ***Chapter 3.7*** for the explanation how to configure the kernel.

```
CONFIG_PPP
CONFIG_PPP_ASYNC
CONFIG_PPP_SYNC_TTY
CONFIG_PPP_DEFLATE
```

3.7. Configure Kernel

To configure the kernel follow the steps and the corresponding commands below.

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

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

Step 2: Execute the following command to set environment variables and import the board's "defconfig" file (Raspberry Pi board example is given below).

```
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: Enable the configuration item.

Selecting <*> means to compile the driver to kernel image.

Selecting <M> means to compile the driver as module.

Taking USB serial option as an example, you can enable CONFIG_USB_SERIAL_OPTION with the options below to compile the USB serial option driver into the kernel image.

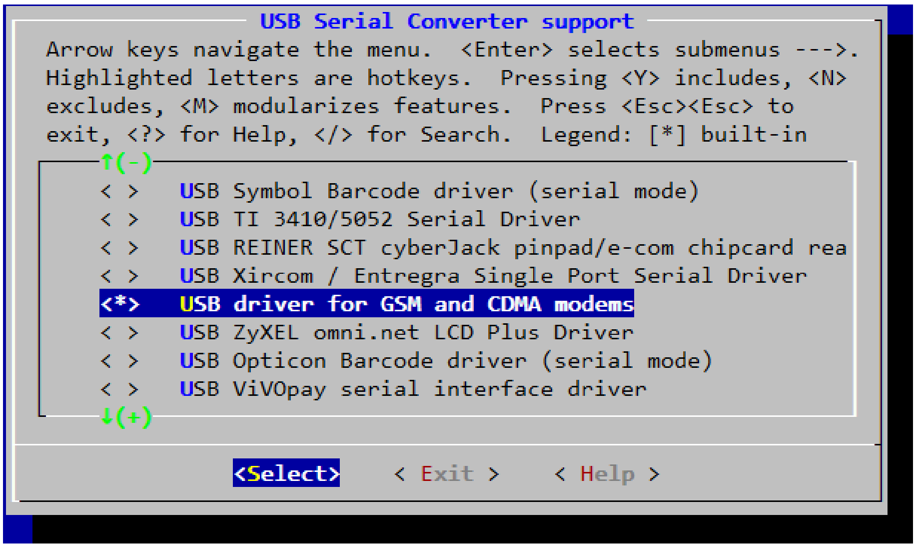

Figure 2: Configure USB Serial Option in Kernel

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

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

- Install QMI_WWAN driver.

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

- Install GobiNet driver.

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

- Install USB serial option driver.

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

4 Test Module

Generally, AT and PPP functions are supported. If a USBNet 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 are created in the */dev* directory.

The AT port is the ttyUSB port created by the USB serial option driver. See ***Table 1*** to view the device name corresponding to AT Command or Modem function.

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

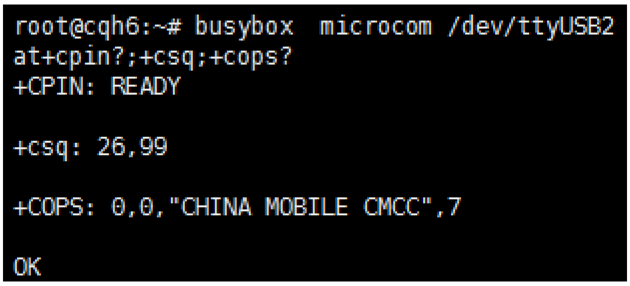

Figure 3: AT Command Test Result

4.2. Test PPP Function

If the module supports USBNet drivers, it is recommended to use USBNet driver interface.

PPP dial-up is more complex than network card dial-up, and it causes higher current consumption of CPU, so it is not recommended to use PPP dial-up.

To set up a PPP call, the following files are required. Check if the following files exist in your product:

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

2. One PPP script file named */etc/ppp/ip-up*, which is used to set DNS. If there is no such file, use *linux-ppp-scripts\ip-up* provided by Quectel.

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

Copy *quectel-ppp*, *quectel-chat-connect* and *quectel-chat-disconnect* to the directory */etc/ppp/peers*, then set up a PPP call by executing the following command:

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

The process of PPP call setup is shown 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 a PPP call is set up successfully.

Please use the following commands to check whether the information of IP, DNS and route in your 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.
```

NOTE

PPP call is not supported on Quectel 5G module series and LTE module series with data rates higher than Cat 4.

4.3. Test GobiNet/QMI_WWAN Driver

To test the GobiNet or QMI_WWAN driver, follow the steps below:

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

- For PC Linux:

```
# make
```

- For embedded Linux:

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

Replace *arm-none-linux-gnueabi-* with the cross compiler on the module.

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

Step 2: Prepare “busybox udhcpc” tool.

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

You can download the source codes of “busybox udhcpc” tool from https://busybox.net, then enable CONFIG_UDHCPC with the command below and copy the *[BUSYBOX]/examples/udhcp/simple.script* script file 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 device node are created. The USB network adapter of the GobiNet driver is named *usbX*, and the QMI device node is */dev/qcqmiX*. The USB network adapter of the QMI_WWAN driver is named *wwanX*, and the QMI channel is */dev/cdc-wdmX*.

quectel-CM sends QMI messages to the module through QMI device node to set up a data connection. See ***document [1]*** for the usage of quectel-CM.

The working process of quectel-CM is shown below (EM12-G running QMI_WWAN driver 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 the 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 explains 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 set up a data call, if your MCU's USB Host Controller does not fully support USB interrupt type endpoint, you need to use **AT$QCRMCALL** instead of QMI tools.

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

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

```
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:~#
```

NOTE

AT$QCRMCALL is only supported on EC25 series, EG25-G, EC21 series, EG91 series, EG21-G, EC20-CE, EG95 series and EM05 series modules.

4.5. Test QMAP on GobiNet/QMI_WWAN Driver

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

When GobiNet or QMI_WWAN driver are used, only one physical network card can be created by default, so only one PDN data call can be set up. However, by 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 GobiNet or QMI_WWAN driver are used, only one IP Packet in one URB can be transferred, which can lead to Host CPU overload in case of high throughput and frequent URB interrupts. 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, contact Quectel Technical Support support@quectel.com.

4.6. Test ECM/RNDIS/NCM/MBIM Driver

On modules using ECM, RNDIS and NCM drivers, you can also perform a call by executing quectel-CM provided by Quectel (see ***document [1]*** for details). You can contact Quectel Technical Support for details if needed.

In MBIM mode, MBIM tools like “mbimcli” and “umbim” can be used to set up a data call. quectel-CM, which is provided by Quectel, can also be used to set up a data call.

5 Power Management

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

If USB communication between the USB host and the USB devices has been 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 suspended USB device to remotely wake up the USB host over the USB, which may also be suspended (for example, in Deep Sleep mode). The USB device, which has a remote wakeup capability, performs an activity to wake up the USB host, and then the USB host is woken up by the remote activity.

USB Auto Suspend and USB Remote Wakeup features of the drivers described in this document except USB serial option driver are enabled by default.

5.1. Enable USB Auto Suspend

For USB serial option driver, add the following statements to *option_probe()* in the *[KERNEL]/drivers/usb/serial/option.c* file 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, add the following statements to *option_probe()* in the *[KERNEL]/drivers/usb/serial/option.c* file 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 If USB Driver Exists in Module

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

Example:

```
root@OpenWrt:~# ls /sys/bus/usb/drivers
GobiNet           cdc_wdm         rndis_host        usbfs
cdc_ether          hub             uas              usbserial
cdc_mbim          option          usb               usbserial_generic
cdc_ncm           qmi_wwan_q      usb-storage
```

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

Similarly, if GobiNet driver is required, make sure *GobiNet* exists.

If QMI_WWAN driver is required, make sure *qmi_wwan_q* exists, and so forth.

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

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

- For USB serial option and GobiNet driver: Kernel logs of different modules are almost the same except for the VID&PID information (marked in red in the following figure). USB serial option and GobiNet for RG502Q series module example is as follows:

```
root@OpenWrt:/# dmesg
[  683.624602] usb 4-1: new SuperSpeed USB device number 5 using xhci-hcd
[  683.650397] usb 4-1: New USB device found, idVendor=2c7c, idProduct=0800
[  683.650425] usb 4-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  683.656207] usb 4-1: Product: RG502Q-EA
[  683.663217] usb 4-1: Manufacturer: Quectel
[  683.666861] usb 4-1: SerialNumber: 39249701
[  683.674432] option 4-1:1.0: GSM modem (1-port) converter detected
[  683.675317] usb 4-1: GSM modem (1-port) converter now attached to ttyUSB0
[  683.681796] option 4-1:1.1: GSM modem (1-port) converter detected
[  683.688537] usb 4-1: GSM modem (1-port) converter now attached to ttyUSB1
[  683.694777] option 4-1:1.2: GSM modem (1-port) converter detected
[  683.701320] usb 4-1: GSM modem (1-port) converter now attached to ttyUSB2
[  683.707574] option 4-1:1.3: GSM modem (1-port) converter detected
[  683.714186] usb 4-1: GSM modem (1-port) converter now attached to ttyUSB3
[  683.737886] GobiNet 4-1:1.4 usb0: register 'GobiNet' at usb-xhci-hcd.1.auto-1, GobiNet Ethernet Device,
[  683.739546] creating qcqmi0
[  685.973561] GobiNet::QMIWDASetDataFormat qmap settings qmap_version=9, rx_size=31744, tx_size=4096
[  685.973598] GobiNet::QMIWDASetDataFormat qmap settings ul_data_aggregation_max_size=4096, ul_data_aggre
```

Figure 4: USB Serial Option and GobiNet for RG502Q Series Module

- 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). USB serial option and QMI_WWAN for RG502Q series module example is as follows:

```
root@OpenWrt:/# dmesg
[  119.804631] usb 4-1: new SuperSpeed USB device number 3 using xhci-hcd
[  119.827695] usb 4-1: New USB device found, idVendor=2c7c, idProduct=0800
[  119.827723] usb 4-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  119.833479] usb 4-1: Product: RG502Q-EA
[  119.840610] usb 4-1: Manufacturer: Quectel
[  119.844146] usb 4-1: SerialNumber: 39249701
[  119.874871] option 4-1:1.0: GSM modem (1-port) converter detected
[  119.875098] usb 4-1: GSM modem (1-port) converter now attached to ttyUSB0
[  119.880477] option 4-1:1.1: GSM modem (1-port) converter detected
[  119.887234] usb 4-1: GSM modem (1-port) converter now attached to ttyUSB1
[  119.893319] option 4-1:1.2: GSM modem (1-port) converter detected
[  119.899995] usb 4-1: GSM modem (1-port) converter now attached to ttyUSB2
[  119.906211] option 4-1:1.3: GSM modem (1-port) converter detected
[  119.912818] usb 4-1: GSM modem (1-port) converter now attached to ttyUSB3
[  119.936455] qmi_wwan_q 4-1:1.4: cdc-wdm0: USB WDM device
[  119.936738] qmi_wwan_q 4-1:1.4: Quectel RG502Q-EA work on RawIP mode
[  119.941518] qmi_wwan_q 4-1:1.4: rx_urb_size = 31744
[  119.948090] qmi_wwan_q 4-1:1.4 wwan0: register 'qmi_wwan_q' at usb-xhci-hcd.1.auto-1,
```

Figure 5: USB Serial Option and QMI_WWAN for RG502Q Series Module

6.3. How to Check Which USB Driver is Installed

This chapter explains how to query the name of the USB driver for the Quectel module USB interface. The USB driver name is identified by the keyword “Driver=”. If “Driver=none”, the reason may be that the corresponding configuration item is not enabled in your kernel configuration, or the VID and PID of Quectel modules are not inserted to the corresponding USB driver source files. In such a case, follow the steps referred to in ***Chapter 2*** to check again.

USB interface and driver for RG502Q series modules example is as follows:

```
root@OpenWrt:/# mount -t debugfs none /sys/kernel/debug/
root@OpenWrt:/# cat /sys/kernel/debug/usb/devices
T:  Bus=04 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#=  3 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=RG502Q-EA
S:  SerialNumber=39249701
C:* #Ifs= 5 Cfg#= 1 Atr=a0 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 6: USB Interface and Driver for RG502Q Series Modules

7 Appendix References

Table 3: Related Documents

Document Name
[1] Quectel_QConnectManager_Linux_User_Guide
[2] Quectel_EC2x&EG2x&EG9x&EM05_Series_QCFG_AT_Commands_Manual
[3] Quectel_BG95&BG77&BG600L_Series_QCFGEXT_AT_Commands_Manual

Table 4: Terms and Abbreviations

Abbreviation	Description
ACM	Abstract Control Model
AP	Application Processor
APN	Access Point Name
CP	Coprocessor
CPU	Central Processing Unit
DNS	Domain Name System
ECM	Ethernet Control Model
EHCI	Enhanced Host Controller Interface
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HCD	Host Controller Driver
IP	Internet Protocol
MBIM	Mobile Interface Broadband Model

MCU	Microcontroller Unit
NCM	Network Control Model
NDIS	Network Driver Interface Specification
NMEA	National Marine Electronics Association
OHCI	Open Host Controller Interface
OS	Operating System
PC	Personal Computer
PDN	Packet Data Network
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