WCDMA&LTE Linux USB Driver User Guide

UMTS/HSPA/LTE Module Series

Rev. WCDMA&LTE_Linux_USB_Driver_User_Guide_V1.5

Date: 2016-05-13

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

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

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

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

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

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 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 products

Contents

Table Index

Figure Index

1 Introduction

This document introduces how to generate the USB driver for Quectel module in Linux OS, and how to use the module after the USB driver is loaded successfully.

This document is applicable to Quectel UC15, UC20, EC20, EC21, EC25 and UGxx modules. The following table shows the details.

Table 1: Supported Products

Product	Driver	Supported	Note
UC20	USB Serial	√	Refer to the Section 3.2 for USB Serial driver
	GobiNet	√	Refer to the Section 3.4 for GobiNet driver
	QMI WWAN	√	Refer to the Section 3.5 for QMI WWAN driver
UC15	USB Serial	√	Refer to the Section 3.2 for USB Serial driver
EC20 EC21 EC25	USB Serial	√	Refer to the Section 3.2 for USB Serial driver
	GobiNet	√	Refer to the Section 3.4 for GobiNet driver
	QMI WWAN	√	Refer to the Section 3.5 for QMI WWAN driver
UGxx	CDC ACM	√	Refer to the Section 3.3 for CDC ACM driver

2 Products Overview

USB on Quectel UMTS/HSPA/LTE module contains several different functional interfaces. Table 2 describes the interface information of different modules in the Linux system.

Table 2: Interface Information

Product	USB Driver	Interface
UC15 VID:0x05c6 PID:0x9090 UC20 VID:0x05c6 PID:0x9003 EC20 VID:0x05c6 PID:0x9215 EC21 VID:0x2c7c PID:0x0121 EC25 VID:0x2c7c PID:0x0125	USB Serial	ttyUSB0 → DM
		ttyUSB1 → For GPS NMEA message output
		ttyUSB2 → For AT commands
		ttyUSB3 → For PPP connections or AT commands
UC20 VID:0x05c6 PID:0x9003 EC20 VID:0x05c6 PID:0x9215 EC21 VID:0x2c7c PID:0x0121 EC25 VID:0x2c7c PID:0x0125	GobiNet or QMI WWAN	ethX or wwanX → Interface 4 can be used as USB Network Adapter
UGxx VID:0x1519 PID:0x0020	CDC ACM	ttyACM0→ For PPP connections or AT commands
		ttyACM1 → Trace 1
		ttyACM2 → Trace 2
		ttyACM3 → For AT commands
		ttyACM4 → For AT commands
		ttyACM5 → Reserved
		ttyACM6 → Reserved

3 System Setup

This chapter mainly describes the general organization of the USB stack in Linux and how to use USB serial, CDC ACM driver, GobiNet driver and QMI WWAN driver. Also, it introduces how to compile and load the driver.

3.1. Linux USB Driver Structure

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

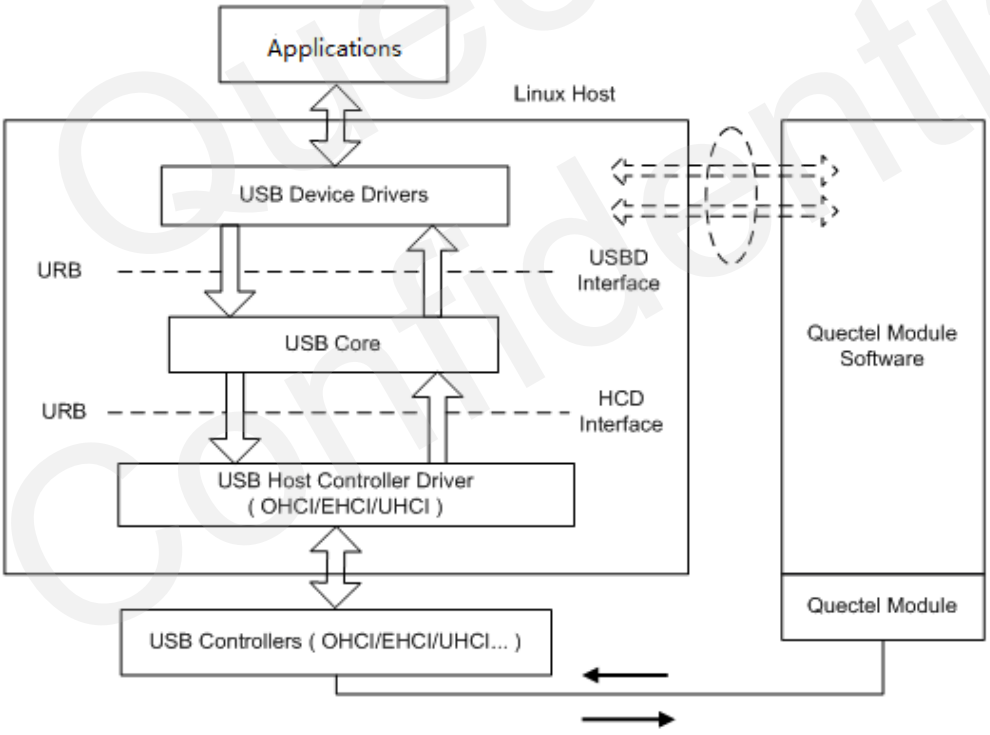

Figure 1: USB Driver Structure

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

USB core, the core of the whole USB host driver, is responsible for the management of USB bus, USB bus devices, and USB bus bandwidth; it provides the interfaces for USB device driver, 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 Driver for UC15/UC20/EC20/EC21/EC25

If you are using UC15/UC20/EC20/EC21/EC25 and require USB serial driver, please read this section for details. Otherwise, please skip this section.

When a Quectel module is attached to the USB Serial driver, the driver will create device files in directory "/dev", named as below:

ttyUSB0/ttyUSB1/ttyUSB2…

The following parts show how to integrate USB Serial driver.

3.2.1. Add VID and PID

In order to recognize Quectel module, you should add module VID and PID information as below:

File: [KERNEL]/drivers/usb/serial/option.c

```
static const struct usb_device_id option_ids[] = {
#if 1 //Added by Quectel
    { USB_DEVICE(0x05C6, 0x9090) }, /* Quectel UC15 */
    { USB_DEVICE(0x05C6, 0x9003) }, /* Quectel UC20 */
    { USB_DEVICE(0x05C6, 0x9215) }, /* Quectel EC20 */
    { USB_DEVICE(0x2C7C, 0x0125) }, /* Quectel EC25 */
    { USB_DEVICE(0x2C7C, 0x0121) }, /* Quectel EC21 */
#endif
```

If you are using EC20 and following files and statements exist in your kernel source files, please delete them, as they will conflict with EC20's USB Drivers.

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

```
{USB_DEVICE(0x05c6, 0x9215)}, /* Acer Gobi 2000 Modem device (VP413) */
```

[KERNEL]/drivers/net/usb/qmi_wwan.c

```
{QMI_GOBI_DEVICE(0x05c6, 0x9215)},          /* Acer Gobi 2000 Modem device (VP413) */
```

3.2.2. Add the Zero Packet Mechanism

As required by the USB protocol, you need to add the mechanism for processing zero packets during bulk out transmission.

For Linux Kernel Version newer than 2.6.34:

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(0x05C6) && desc->idProduct == cpu_to_le16(0x9090))
            urb->transfer_flags |= URB_ZERO_PACKET;
        if (desc->idVendor == cpu_to_le16(0x05C6) && desc->idProduct == cpu_to_le16(0x9003))
            urb->transfer_flags |= URB_ZERO_PACKET;
        if (desc->idVendor == cpu_to_le16(0x05C6) && desc->idProduct == cpu_to_le16(0x9215))
            urb->transfer_flags |= URB_ZERO_PACKET;
        if (desc->idVendor == cpu_to_le16(0x2C7C))
            urb->transfer_flags |= URB_ZERO_PACKET;
    }
    #endif
    return urb;
}
```

For Linux Kernel Version older than 2.6.35:

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(0x05C6) && desc->idProduct == cpu_to_le16(0x9090))
            urb->transfer_flags |= URB_ZERO_PACKET;
        if (desc->idVendor == cpu_to_le16(0x05C6) && desc->idProduct == cpu_to_le16(0x9003))
            urb->transfer_flags |= URB_ZERO_PACKET;
        if (desc->idVendor == cpu_to_le16(0x05C6) && desc->idProduct == cpu_to_le16(0x9215))
            urb->transfer_flags |= URB_ZERO_PACKET;
        if (desc->idVendor == cpu_to_le16(0x2C7C))
            urb->transfer_flags |= URB_ZERO_PACKET;
    #endif
    return urb;
}
```

3.2.3. Add Reset Resume

Some USB host controllers/USB hubs will lost power or be reset when MCU entering into suspend/sleep mode, and they cannot resume USB devices when MCU exiting from suspend/sleep mode; instead, they will operate reset-resume. You should add the following statements:

For Linux Kernel Version newer than 3.4:

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 older than 3.5:

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. Enlarge Bulk out URBs

For Linux kernel version older than 2.6.29. You need to enlarge bulk out URBs to get faster uplink speed.

File: [KERNEL]/drivers/usb/serial/option.c

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

3.2.5. Use GobiNet or QMI WWAN

If you are using UC20/EC20/EC21/EC25 and require GobiNet or QMI WWAN, you must add the following statements to prevent UC20/EC20/EC21/EC25's interface 4 from being used as USB serial device.

For Linux Kernel Version newer than 2.6.30:

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 UC20's interface 4 can be used as USB Network device
    if      (serial->dev->descriptor.idVendor       ==      cpu_to_le16(0x05C6)     &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9003)
        && serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4)
        return -ENODEV;
//Quectel EC20's interface 4 can be used as USB Network device
    if      (serial->dev->descriptor.idVendor       ==      cpu_to_le16(0x05C6)     &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9215)
        && serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4)
```

```
        return -ENODEV;
//Quectel EC25&EC21's interface 4 can be used as USB Network device
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)
        && 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 older than 2.6.31:

File: [KERNEL]/drivers/usb/serial/option.c

```
static int option_startup(struct usb_serial *serial)
{
……
    dbg("%s", __func__);
#if 1   //Added by Quectel
//Quectel UC20's interface 4 can be used as USB Network device
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9003)
        && serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4)
        return -ENODEV;
//Quectel EC20's interface 4 can be used as USB Network device
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9215)
        && serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4)
        return -ENODEV;
//Quectel EC25&EC21's interface 4 can be used as USB Network device
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)
        && serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4)
        return -ENODEV;
#endif
……
}
```

3.2.6. Modify Kernel Configuration

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

Step 1:

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

Step 2: Set your environment variables, and import your board's defconfig. The following is an example for Raspeberrypi board

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

Step 3:

```
make menuconfig
```

Step 4: Enable CONFIG_USB_SERIAL_OPTION

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

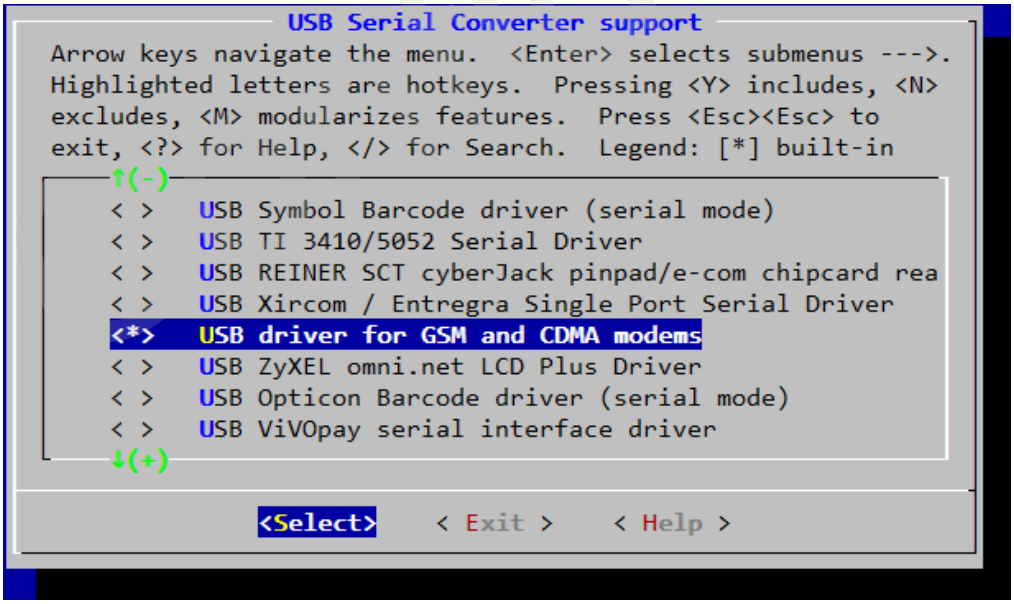

Figure 2: Configure USB Serial in Kernel

3.2.7. Build and Load Driver as Kernel Module for PC in Linux

If you are using Linux on PC, you can follow the steps below to build the driver as Kernel module, and use modprobe command to load the module.

Step 1:

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

Step 2:

```
sudo make -C /lib/modules/`uname -r`/build M=`pwd`/drivers/usb/serial obj-m=option.o modules
sudo make -C /lib/modules/`uname -r`/build M=`pwd`/drivers/usb/serial obj-m=usb_wwan.o modules
sudo make -C /lib/modules/`uname -r`/build M=`pwd`/drivers/usb/serial obj-m=qcserial.o modules
```

Step 3:

```
sudo cp drivers/usb/serial/option.ko /lib/modules/`uname -r`/kernel/drivers/usb/serial
sudo cp drivers/usb/serial/usb_wwan.ko /lib/modules/`uname -r`/kernel/drivers/usb/serial
sudo cp drivers/usb/serial/qcserial.ko /lib/modules/`uname -r`/kernel/drivers/usb/serial
sudo depmod
sudo reboot
```

3.3. CDC ACM Driver for UGxx

If you are using UGxx and require CDC ACM driver, please read this section for details. Otherwise, please skip this section.

When a Quectel module is attached to CDC ACM driver, the driver will create device files in directory "/dev", named as below:

ttyACM0/ttyACM1/ttyACM2…

The following parts show how to integrate the CDC ACM driver.

3.3.1. Modify Driver Source Code

The device is attached to CDC ACM driver according to the USB Class Type, so you do not need to add PID and VID information in driver source code.

3.3.2. Add the Zero Packet Mechanism

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

This document takes the **Linux 3.2** as an example, and there may be a little difference to other versions; but they are basically the same.

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

```
.....
for (i = 0; i < ACM_NW; i++) {
      struct acm_wb *snd = &(acm->wb[i]);
      snd->urb = usb_alloc_urb(0, GFP_KERNEL);
      if (snd->urb == NULL) {
            dev_err(&intf->dev,
                  "out of memory (write urbs usb_alloc_urb)\n");
            goto alloc_fail7;
      }
      if (usb_endpoint_xfer_int(epwrite))
            usb_fill_int_urb(snd->urb, usb_dev,
                  usb_sndbulkpipe(usb_dev, epwrite->bEndpointAddress),
                  NULL, acm->writesize, acm_write_bulk, snd, epwrite->bInterval);
      else
            usb_fill_bulk_urb(snd->urb, usb_dev,
                  usb_sndbulkpipe(usb_dev, epwrite->bEndpointAddress),
                  NULL, acm->writesize, acm_write_bulk, snd);
      snd->urb->transfer_flags |= URB_NO_TRANSFER_DMA_MAP;
      #if 1    //Added by Quectel for Zero Packet
      if (usb_dev->descriptor.idVendor == 0x1519 && usb_dev->descriptor.idProduct == 0x0020)
            snd->urb->transfer_flags |= URB_ZERO_PACKET;
      #endif
     snd->instance = acm;
}
usb_set_intfdata(intf,acm)
.....
```

3.3.3. Add Reset Resume

Some USB host controllers/USB hubs will lost power or be reset when MCU entering into suspend/sleep mode, and they cannot resume USB devices when MCU exiting from suspend/sleep mode; instead, they will operate reset-resume. You should add the following statements:

For Linux Kernel Version older than 2.6.35

File: [KERNEL]/drivers/usb/class/cdc-acm.c

```
static struct usb_driver acm_driver = {
        .name =         "cdc_acm",
        .probe =        acm_probe,
        .disconnect =   acm_disconnect,
#ifdef CONFIG_PM
        .suspend =      acm_suspend,
        .resume =       acm_resume,
#if 1 //Added by Quectel
        .reset_resume = acm_resume,
#endif
#endif
        .id_table =     acm_ids,
#ifdef CONFIG_PM
        .supports_autosuspend = 1,
#endif
};
```

3.3.4. Modify Kernel Configuration

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

Step 1:

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

Step 2: Set your environment variables, and import your board's defconfig. The following is an example for Raspeberrypi board

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

Step 3:

```
make menuconfig
```

Step 4: enable CONFIG_USB_ACM

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

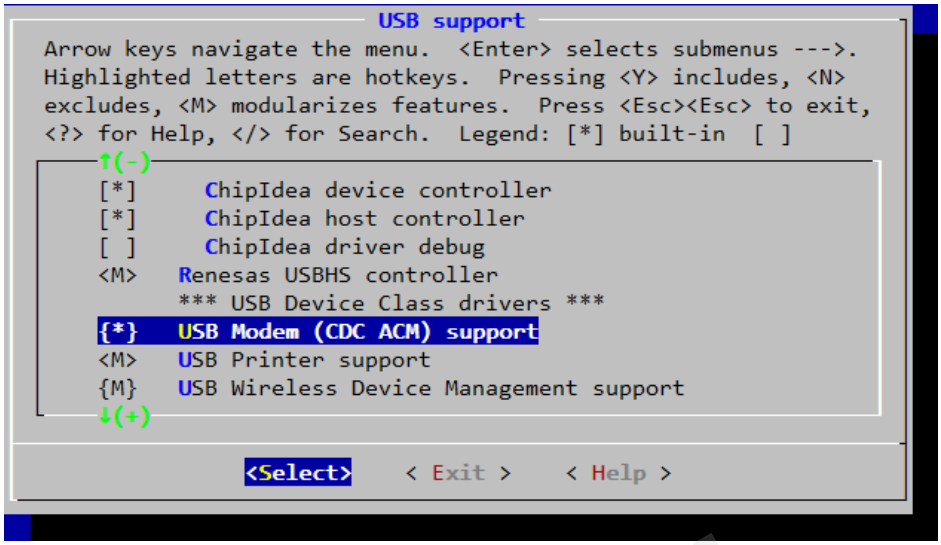

Figure 3: Configure CDC ACM Driver in Kernel

3.3.5. Build and Load Driver as Kernel Module for PC in Linux

If you are using Linux on PC, you can follow the steps below to build the driver as kernel module, and use modprobe command to load the module.

Step 1:

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

Step 2:

```
sudo make -C /lib/modules/`uname -r`/build M=`pwd`/drivers/usb/class obj-m=cdc-acm.o modules
```

Step 3:

```
sudo cp drivers/usb/class/cdc-acm.ko /lib/modules/`uname -r`/kernel/drivers/usb/class
sudo depmod
sudo reboot
```

3.4. GobiNet Driver for UC20/EC20/EC21/EC25

If you are using UC20/EC20/EC21/EC25 and require GobiNet driver, please read this section for details. Otherwise, please skip this section.

When a Quectel module is attached to GobiNet driver, the driver will create a network device and a QMI channel. The network device is named as ethX (usbX if the kernel version is 2.6.39 or older), and the QMI channel is named as /dev/qcqmiX. The network device is used for data transmission, and QMI channel is

used for QMI message interaction.

The following parts show how to integrate the GobiNet driver.

3.4.1. Modify Driver Source Code

The GobiNet driver is provided by Quectel as a form of source file. You should copy the source files to "[KERNEL]/drivers/net/usb/" ([KERNEL]/drivers/usb/net/ if the kernel version is older than 2.6.22).

3.4.2. Modify Kernel Configuration

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

Step 1:

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

Step 2: Set your environment variables, and import your board's defconfig. The following is an example for Raspeberrypi board

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

Step 3:

```
make menuconfig
```

Step 4: Enable CONFIG_USB_USBNET

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

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

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

If you are using EC20 and following files and statements exist in your kernel source files, please delete

them, as they will conflict with EC20's USB Drivers.

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

```
{USB_DEVICE(0x05c6, 0x9215)}, /* Acer Gobi 2000 Modem device (VP413) */
```

[KERNEL]/drivers/net/usb/qmi_wwan.c

```
{QMI_GOBI_DEVICE(0x05c6, 0x9215)},          /* Acer Gobi 2000 Modem device (VP413) */
```

3.4.3. Build and Load Driver as Kernel Module for PC in Linux

If you are using Linux on PC, you can follow the steps below to build the driver as Kernel module, and use modprobe command to load the module.

Step 1:

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

Step 2:

```
sudo make -C /lib/modules/`uname -r`/build M=`pwd`/drivers/net/usb obj-m=GobiNet.o modules
sudo make -C /lib/modules/`uname -r`/build M=`pwd`/drivers/usb/serial obj-m=qcserial.o modules
```

Step 3:

```
sudo cp drivers/net/usb/GobiNet.ko /lib/modules/`uname -r`/kernel/drivers/net/usb
sudo cp drivers/usb/serial/qcserial.ko /lib/modules/`uname -r`/kernel/drivers/usb/serial
sudo depmod
sudo reboot
```

3.5. QMI WWAN Driver for UC20/EC20/EC21/EC25

If you are using UC20/EC20/EC21/EC25 and the Linux kernel version is newer than 3.3 and require QMI WWAN driver, please read this section for details. Otherwise, please skip this section.

When a Quectel module is attached to QMI WWAN driver, the driver will create a network device and a QMI channel. The network device is named as wwanX, and QMI channel is named as /dev/cdc-wdmX. The network device is working for data transmission, and QMI channel is working for QMI message interaction.

The following parts show how to integrate the QMI WWAN driver.

3.5.1. Add VID and PID

QMI WWAN driver source file is "[KERNEL]/drivers/net/usb/qmi_wwan.c".

In order to recognize Quectel module, you should add module PID and VID information as below:

[KERNEL]/drivers/net/usb/qmi_wwan.c

```
static const struct usb_device_id products[] = {
#if 1     //Added by Quectel
#ifndef QMI_FIXED_INTF
/* map QMI/wwan function by a fixed interface number */
#define QMI_FIXED_INTF(vend, prod, num) \
        .match_flags                                    =       USB_DEVICE_ID_MATCH_DEVICE      |
USB_DEVICE_ID_MATCH_INT_INFO, \
        .idVendor               = vend, \
        .idProduct              = prod, \
        .bInterfaceClass        = 0xff, \
        .bInterfaceSubClass = 0xff, \
        .bInterfaceProtocol = 0xff, \
        .driver_info            = (unsigned long)&qmi_wwan_force_int##num,
#endif
    { QMI_FIXED_INTF(0x05C6, 0x9003, 4) },    /* Quectel UC20 */
    { QMI_FIXED_INTF(0x05C6, 0x9215, 4) },    /* Quectel EC20 */
    { QMI_FIXED_INTF(0x2C7C, 0x0125, 4) },    /* Quectel EC25 */
    { QMI_FIXED_INTF(0x2C7C, 0x0121, 4) },    /* Quectel EC21 */
#endif
```

If you are using EC20 and following files and statements exist in your kernel source files, please delete them, as they will conflict with EC20's USB Drivers.

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

```
{USB_DEVICE(0x05c6, 0x9215)}, /* Acer Gobi 2000 Modem device (VP413) */
```

[KERNEL]/drivers/net/usb/qmi_wwan.c

```
{QMI_GOBI_DEVICE(0x05c6, 0x9215)},          /* Acer Gobi 2000 Modem device (VP413) */
```

3.5.2. Add Support for Raw IP Mode for EC21/EC25

QMI WWAN driver source file is "[KERNEL]/drivers/net/usb/qmi_wwan.c".

EC21/EC25 only supports raw IP mode (IP packets not encapsulated in Ethernet frames). So Ethernet header must be stripped when packets sent to Quectel modules, and be added when packets received

from Quectel modules.

You must add the following statements to support raw IP mode.

[KERNEL]/drivers/net/usb/qmi_wwan.c

```
#include <linux/usb/usbnet.h>
#include <linux/usb/cdc-wdm.h>

#if 1     //Added by Quectel
#include <linux/etherdevice.h>
struct sk_buff *qmi_wwan_tx_fixup(struct usbnet *dev, struct sk_buff *skb, gfp_t flags)
{
    if (dev->udev->descriptor.idVendor != cpu_to_le16(0x2C7C))
        return skb;

    // Skip Ethernet header from message
    if (skb_pull(skb, ETH_HLEN)) {
        return skb;
    } else {
        dev_err(&dev->intf->dev,    "Packet Dropped ");
    }

    // Filter the packet out, release it
    dev_kfree_skb_any(skb);
    return NULL;
}

#include <linux/version.h>
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,9,1 ))
static int qmi_wwan_rx_fixup(struct usbnet *dev, struct sk_buff *skb)
{
    __be16 proto;

    if (dev->udev->descriptor.idVendor != cpu_to_le16(0x2C7C))
        return 1;

    /* This check is no longer done by usbnet */
    if (skb->len < dev->net->hard_header_len)
        return 0;

    switch (skb->data[0] & 0xf0) {
    case 0x40:
        proto = htons(ETH_P_IP);
        break;
```

```
    case 0x60:
        proto = htons(ETH_P_IPV6);
        break;
    case 0x00:
        if (is_multicast_ether_addr(skb->data))
            return 1;
        /* possibly bogus destination - rewrite just in case */
        skb_reset_mac_header(skb);
        goto fix_dest;
    default:
        /* pass along other packets without modifications */
        return 1;
    }
    if (skb_headroom(skb) < ETH_HLEN)
        return 0;
    skb_push(skb, ETH_HLEN);
    skb_reset_mac_header(skb);
    eth_hdr(skb)->h_proto = proto;
    memset(eth_hdr(skb)->h_source, 0, ETH_ALEN);
fix_dest:
    memcpy(eth_hdr(skb)->h_dest, dev->net->dev_addr, ETH_ALEN);
    return 1;
}

/* very simplistic detection of IPv4 or IPv6 headers */
static bool possibly_iphdr(const char *data)
{
    return (data[0] & 0xd0) == 0x40;
}
#endif
#endif

……

/* if follow function exist, modify it as below */
static int qmi_wwan_bind(struct usbnet *dev, struct usb_interface *intf)
{
……

#if 1 //Added by Quectel
    if (dev->udev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        dev_info(&intf->dev,    "Quectel EC21&EC25 work on RawIP mode\n");
        dev->net->flags |= IFF_NOARP;
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,9,1 ))
```

```
        /* make MAC addr easily distinguishable from an IP header */
        if (possibly_iphdr(dev->net->dev_addr)) {
            dev->net->dev_addr[0] |= 0x02;    /* set local assignment bit */
            dev->net->dev_addr[0] &= 0xbf;    /* clear "IP" bit */
        }
#endif
        usb_control_msg(
            interface_to_usbdev(intf),
            usb_sndctrlpipe(interface_to_usbdev(intf), 0),
            0x22, //USB_CDC_REQ_SET_CONTROL_LINE_STATE
            0x21, //USB_DIR_OUT | USB_TYPE_CLASS | USB_RECIP_INTERFACE
            1, //active CDC DTR
            intf->cur_altsetting->desc.bInterfaceNumber,
            NULL, 0, 100);
    }
#endif
err:
    return status;
}

……

/* if follow function exist, modify it as below */
static int qmi_wwan_bind_shared(struct usbnet *dev, struct usb_interface *intf)
{
……

#if 1 //Added by Quectel
    if (dev->udev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        dev_info(&intf->dev,    "Quectel EC21&EC25 work on RawIP mode\n");
        dev->net->flags |= IFF_NOARP;
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,9,1 ))
        /* make MAC addr easily distinguishable from an IP header */
        if (possibly_iphdr(dev->net->dev_addr)) {
            dev->net->dev_addr[0] |= 0x02;    /* set local assignment bit */
            dev->net->dev_addr[0] &= 0xbf;    /* clear "IP" bit */
        }
#endif
        usb_control_msg(
            interface_to_usbdev(intf),
            usb_sndctrlpipe(interface_to_usbdev(intf), 0),
            0x22, //USB_CDC_REQ_SET_CONTROL_LINE_STATE
            0x21, //USB_DIR_OUT | USB_TYPE_CLASS | USB_RECIP_INTERFACE
            1, //active CDC DTR
```

```
                intf->cur_altsetting->desc.bInterfaceNumber,
                NULL, 0, 100);
    }
#endif
err:
    return status;
}
……

/* if follow struct exist, modify it as below */
static const struct driver_info      qmi_wwan_info =
{
……
#if 1 //Added by Quectel
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 4,5,0 ))
    .tx_fixup           = qmi_wwan_tx_fixup,
#endif
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,9,1 ))
    .rx_fixup           = qmi_wwan_rx_fixup,
#endif
#endif
}
……

/* if follow struct exist, modify it as below */
static const struct driver_info qmi_wwan_force_int4 = {
……
#if 1 //Added by Quectel
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 4,5,0 ))
    .tx_fixup           = qmi_wwan_tx_fixup,
#endif
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,9,1 ))
    .rx_fixup           = qmi_wwan_rx_fixup,
#endif
#endif
};

/* if follow struct exist, modify it as below */
static const struct driver_info qmi_wwan_shared = {
……
#if 1 //Added by Quectel
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 4,5,0 ))
    .tx_fixup           = qmi_wwan_tx_fixup,
#endif
```

```
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,9,1 ))
    .rx_fixup           = qmi_wwan_rx_fixup,
#endif
#endif
};
```

3.5.3. Modify Kernel Configuration

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

Step 1:

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

Step 2: Set your environment variables, and import your board's defconfig. The following is an example for Raspeberrypi board

```
export ARCH=arm

export CROSS_COMPILE=arm-none-linux-gnueabi-

make bcmrpi_defconfig
```

Step 3:

```
make menuconfig
```

Step 4: Enable CONFIG_USB_NET_QMI_WWAN

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

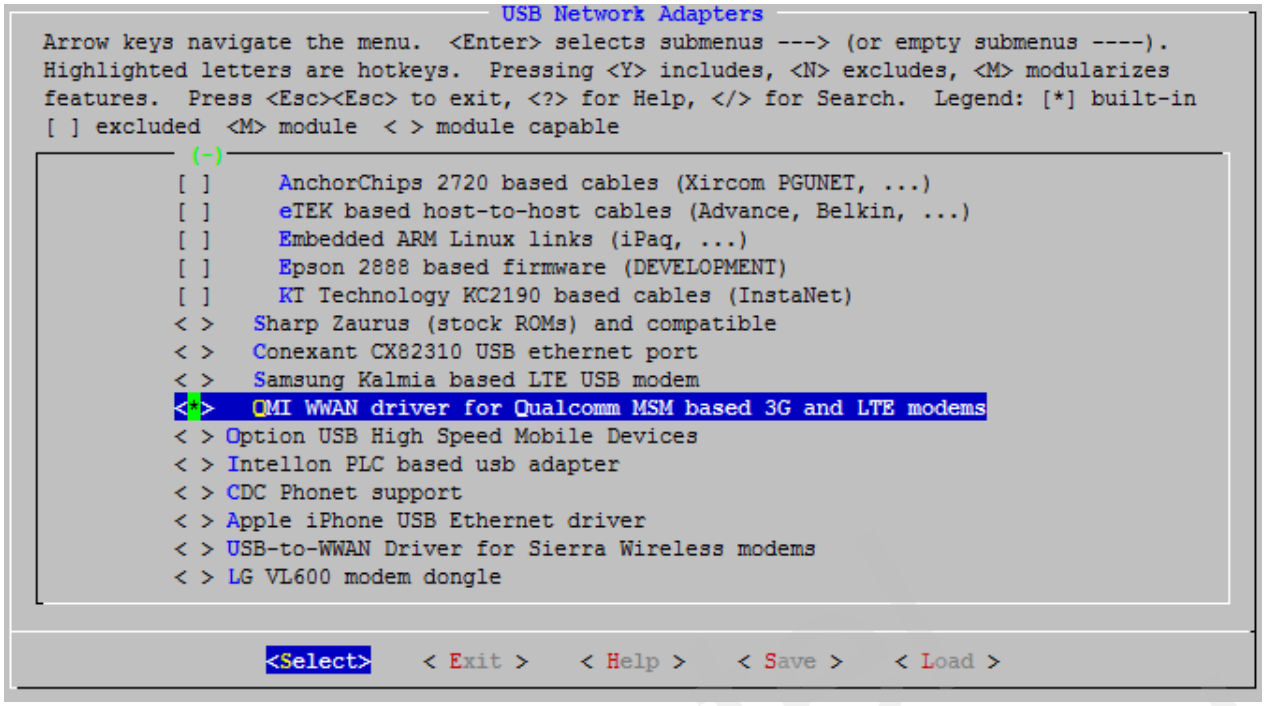

Figure 4: Configure QMI WWAN Driver in Kernel

3.5.4. Build and Load Driver as Kernel Module for PC in Linux

If you are using Linux on PC, you can follow steps below to build the driver as Kernel module, and use modprobe command to load the module.

Step 1:

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

Step 2:

```
sudo make -C /lib/modules/`uname -r`/build M=`pwd`/drivers/net/usb obj-m=qmi_wwan.o modules
sudo make -C /lib/modules/`uname -r`/build M=`pwd`/drivers/usb/serial obj-m=qcserial.o modules
```

Step 3:

```
sudo cp drivers/net/usb/qmi_wwan.ko /lib/modules/`uname -r`/kernel/drivers/net/usb
sudo cp drivers/usb/serial/qcserial.ko /lib/modules/`uname -r`/kernel/drivers/usb/serial
sudo depmod
sudo reboot
```

3.6. Configure Kernel to Support PPP

If you need to use PPP function, then you should configure kernel to support PPP. Here shows how to configure kernel.

Step 1:

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

Step 2: Set your environment variables, and import your board's defconfig. The following is an example.

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

Step 3:

```
make menuconfig
```

Step 4: Enable CONFIG_PPP_ASYNC CONFIG_PPP_SYNC_TTY CONFIG_PPP_DEFLATE.

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

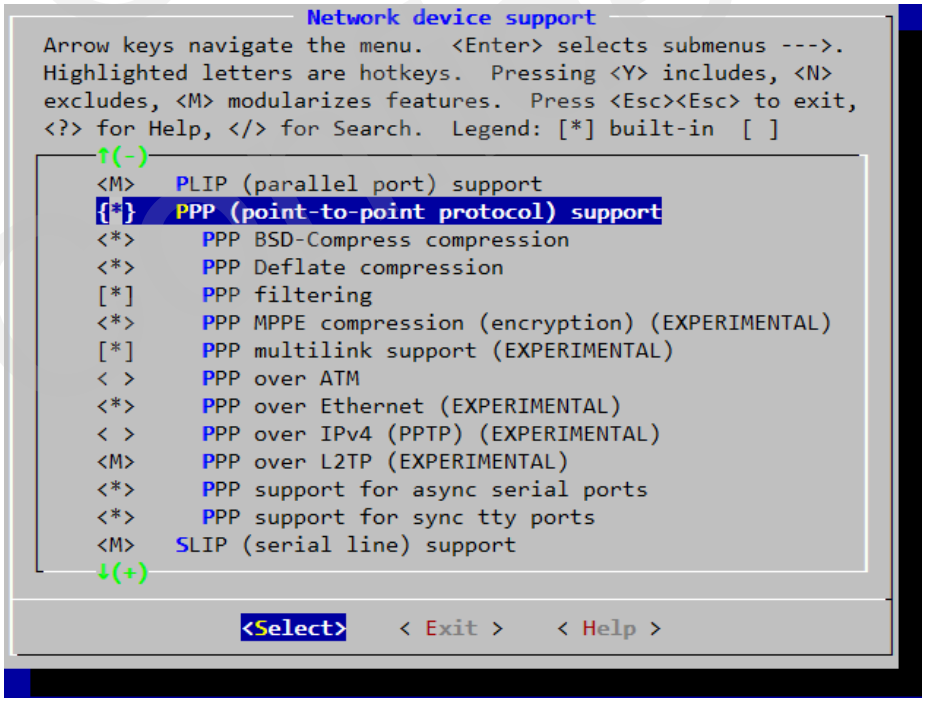

Figure 5: Configure PPP in Kernel

4 Power Management

The Linux USB system provides two advanced Power Management features: USB Auto Suspend and USB Remote Wakeup. This chapter introduces how to enable the features. If they are required by your product, please read this chapter for details. Otherwise, please ignore this chapter.

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

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

4.1. Enable USB Auto Suspend

For USB Serial Driver, please add the following statements to function option_probe() in 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
//For USB Auto Suspend
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9090)) {
        pm_runtime_set_autosuspend_delay(&serial->dev->dev, 3000);
        usb_enable_autosuspend(serial->dev);
    }
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9003)) {
        pm_runtime_set_autosuspend_delay(&serial->dev->dev, 3000);
        usb_enable_autosuspend(serial->dev);
    }
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9215)) {
        pm_runtime_set_autosuspend_delay(&serial->dev->dev, 3000);
```

```
        usb_enable_autosuspend(serial->dev);
    }
    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;
}
```

For CDC ACM Driver, please add the following statements to function acm_probe () in file "[KERNEL]/drivers/usb/class/cdc-acm.c".

```
static int acm_probe(struct usb_interface *intf,
                     const struct usb_device_id *id)
{
    struct usb_cdc_union_desc *union_header = NULL;
    ……
#if 1 //Added by Quectel
//For USB Auto Suspend
 if((usb_dev->descriptor.idVendor == 0x1519) && (usb_dev->descriptor.idProduct == 0x0020))
{
        pm_runtime_set_autosuspend_delay(&usb_dev->dev, 3000);
        usb_enable_autosuspend(usb_dev);
    }
#endif

    return 0;
alloc_fail8:
    if (acm->country_codes) {
……
}
```

4.2. Enable USB Remote Wakeup

For USB Serial Driver, please add the following statements to function option_probe() in 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
//For USB Remote Wakeup
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9090)) {
        device_init_wakeup(&serial->dev->dev, 1); //usb remote wakeup
    }
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9003)) {
        device_init_wakeup(&serial->dev->dev, 1); //usb remote wakeup
    }
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x9215)) {
        device_init_wakeup(&serial->dev->dev, 1); //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;
}
```

For CDC ACM Driver, please add the following statements to function acm_probe () in file "[KERNEL]/drivers/usb/class/cdc-acm.c".

```
static int acm_probe(struct usb_interface *intf,
                     const struct usb_device_id *id)
{
    struct usb_cdc_union_desc *union_header = NULL;
    ……
#if 1 //Added by Quectel
//For USB Remote Wakeup
    if ((usb_dev->descriptor.idVendor == 0x1519) && (usb_dev->descriptor.idProduct == 0x0020))
{
        device_init_wakeup(&usb_dev->dev, 1); //usb remote wakeup
    }
#endif
    return 0;
alloc_fail8:
    if (acm->country_codes) {
……
}
```

5 Test the Module

Generally, AT and PPP functions will be used in your product. If you are using UC20/EC20/EC21/EC25 and have installed GobiNet or QMI WWAN driver, the USB network adapter function can also be used in your product. Here shows how to test these functions.

5.1. Test AT Function

After the module is connected and USB driver is loaded successfully, there will create several device files in "/dev".

The AT port of UC15/UC20/EC20/EC21/EC25 is /dev/ttyUSB2, and the AT port of UGxx is /dev/ttyACM3.

Now you can use UART port tools like “minicom” or “busybox microcom” to test AT function, as shown below:

```
# busybox microcom /dev/ttyUSB2
```

The following is an example for EC20:

Figure 6: AT Test Result for EC20

5.2. Test PPP Function

In order to set up PPP call, the following files are required. Please check if they exist in your product.

1. pppd and chat program:
   If the two programs do not exist, you can download the source code of them from https://ppp.samba.org/download.html and port them to your product.
2. One PPP script file named as "/etc/ppp/ip-up" which is used to set DNS (Domain Name System). If there is no such file on your product, you can 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". Depending on your product, you may need to make some changes. More information please refers to "linux-ppp-scripts\readme".

You should copy "quectel-ppp", "quectel-chat-connect" and "quectel-chat-disconnect" to the directory "/etc/ppp/peers". Then you can start to set up PPP call via the following command:

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

The process of dialing is shown as below (example of EC20):

```
# pppd options in effect:
debug           # (from /etc/ppp/peers/quectel-ppp)
nodetach            # (from /etc/ppp/peers/quectel-ppp)
dump            # (from /etc/ppp/peers/quectel-ppp)
noauth          # (from /etc/ppp/peers/quectel-ppp)
user test       # (from /etc/ppp/peers/quectel-ppp)
password ??????       # (from /etc/ppp/peers/quectel-ppp)
remotename 3gppp            # (from /etc/ppp/peers/quectel-ppp)
/dev/ttyUSB3       # (from /etc/ppp/peers/quectel-ppp)
115200          # (from /etc/ppp/peers/quectel-ppp)
lock        # (from /etc/ppp/peers/quectel-ppp)
connect chat -s -v -f /etc/ppp/peers/quectel-chat-connect          # (from /etc/ppp/peers/quectel-ppp)
disconnect chat -s -v -f /etc/ppp/peers/quectel-chat-disconnect        # (from /etc/ppp/peers/quectel-ppp)
nocrtscts       # (from /etc/ppp/peers/quectel-ppp)
modem           # (from /etc/ppp/peers/quectel-ppp)
hide-password           # (from /etc/ppp/peers/quectel-ppp)
```

```
novj       # (from /etc/ppp/peers/quectel-ppp)
novjccomp          # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-local        # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-remote           # (from /etc/ppp/peers/quectel-ppp)
ipparam 3gppp          # (from /etc/ppp/peers/quectel-ppp)
noipdefault          # (from /etc/ppp/peers/quectel-ppp)
ipcp-max-failure 10          # (from /etc/ppp/peers/quectel-ppp)
defaultroute         # (from /etc/ppp/peers/quectel-ppp)
usepeerdns         # (from /etc/ppp/peers/quectel-ppp)
noccp          # (from /etc/ppp/peers/quectel-ppp)
abort on (BUSY)
abort on (NO CARRIER)
abort on (NO DIALTONE)
abort on (ERROR)
abort on (NO ANSWER)
timeout set to 30 seconds
send (AT^M)
expect (OK)
^M
OK
 -- got it

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

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

```
expect (OK)
^M
^M
Quectel^M
EC20^M
Revision: EC20CQAR02A03E2G_BETA0914^M
^M
SubEdition: V01^M
^M
+CSQ: 23,99^M
^M
+CPIN: READY^M
^M
+COPS: 0,0,"CHN-CT",7^M
^M
+CGREG: 2,1,"FFFE","6916934",7^M
^M
OK
 -- got it

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

send (ATD*99#^M)
expect (CONNECT)
^M
```

```
^M

CONNECT

 -- got it


Script chat -s -v -f /etc/ppp/peers/quectel-chat-connect finished (pid 3017), status = 0x0

Serial connection established.

using channel 3

Using interface ppp0

Connect: ppp0 <--> /dev/ttyUSB3

sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0xf2b7d6ee> <pcomp> <accomp>]

rcvd [LCP ConfReq id=0x4 <asyncmap 0x0> <auth chap MD5> <magic 0x45c0e381> <pcomp>
<accomp>]

sent [LCP ConfAck id=0x4 <asyncmap 0x0> <auth chap MD5> <magic 0x45c0e381> <pcomp>
<accomp>]

rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0xf2b7d6ee> <pcomp> <accomp>]

rcvd [LCP DiscReq id=0x5 magic=0x45c0e381]

rcvd [CHAP Challenge id=0x1 <f8d54e0fa294c100101805a512176ff1>, name = "UMTS_CHAP_SRVR"]

sent [CHAP Response id=0x1 <e8ad86182138523599fb54a172da7154>, name = "test"]

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>]

rcvd [IPCP ConfReq id=0x4]

sent [IPCP ConfNak id=0x4 <addr 0.0.0.0>]

rcvd [IPCP ConfNak id=0x1 <addr 100.65.245.137> <ms-dns1 61.132.163.68> <ms-dns2
202.102.213.68>]

sent [IPCP ConfReq id=0x2 <addr 100.65.245.137> <ms-dns1 61.132.163.68> <ms-dns2
202.102.213.68>]

rcvd [IPCP ConfReq id=0x5]

sent [IPCP ConfAck id=0x5]

rcvd [IPCP ConfAck id=0x2 <addr 100.65.245.137> <ms-dns1 61.132.163.68> <ms-dns2
```

```
202.102.213.68>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 100.65.245.137
remote IP address 10.64.64.64
primary     DNS address 61.132.163.68
secondary DNS address 202.102.213.68
Script /etc/ppp/ip-up started (pid 3020)
Script /etc/ppp/ip-up finished (pid 3020), status = 0x0
```

Now PPP call is set up successfully.

Use following commands to check IP/DNS/Route.

```
# ifconfig ppp0
ppp0          Link encap:Point-to-Point Protocol
              inet addr:100.65.245.137  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=46.4 ms
```

You can use following commands to terminate PPPD process to disconnect a PPP call:

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

5.3. Test GobiNet or QMI WWAN

If you are using UC20/EC20/EC21/EC25 and require GobiNet or QMI WWAN driver, please read this section for details. Otherwise, please skip this section.

If you want to set up data connection manually, Quectel provides a Connect Manager program to set up data connection. The Connect Manager is provided in the form of source code in directory "quectel-CM".

Please follow steps below to test GobiNet or QMI WWAN:

Step 1: Compile Connect Manager.

For PC Linux:

```
# make
```

For Emended Linux:

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

Please replace "arm-none-linux-gnueabi-" by your product cross compiler.

The output of this step is quectel-CM.

Step 2: Prepare busybox udhcpc tool.

quectel-CM will call busybox udhpc to obtain IP and NDS, and busybox udhpc will call script file /usr/share/udhcpc/default.script to set IP/DNS/Routing table for Linux board. You can download this tool's source code from https://busybox.net/. You should enable CONFIG_UDHCPC in busybox menuconfig,

and copy the script file [BUSYBOX]/examples/udhcp/simple.script to your Linux board (renamed as /usr/share/udhcpc/default.script).

Step 3: Use quectel-CM to setup data call.

After the module is connected and GobiNet or QMI WWAN driver is loaded successfully, there will create a USB network adapter and a QMI channel. The USB network adapter of GobiNet is named as ethX (usbX if the kernel version is 2.6.39 or older), and the QMI channel is named as /dev/qcqmiX. The USB network adapter of QMI WWAN is named as wwanX, and the QMI channel name is named as /dev/cdc-wdmX.

quectel-CM will send QMI Message to the module via QMI channel to setup data connection. Please refer to the following message to use quectel-CM:

```
# quectel-CM -h
Usage: ./quectel-CM [-s [apn [user password auth]]] [-p pincode] [-f logfilename]
-s [apn [user password auth]] Set apn/user/password/auth get from your network provider
-p pincode                    Verify sim card pin if sim card is locked
-f logfilename                Save log message of this program to file
Example 1: ./quectel-CM
Example 2: ./quectel-CM -s 3gnet
Example 3: ./quectel-CM -s 3gnet carl 1234 0 -p 1234 -f gobinet_log.txt
```

The process of quectel-CM is shown as below (example of EC20&GobiNet):

```
# quectel-CM –s ctnet &
[01-01_00:26:45:355] Quectel_ConnectManager_SR01A01V10
[01-01_00:26:45:356] ./quectel-CM profile = ctnet///, pincode =
[01-01_00:26:45:357] Find qmichannel = /dev/qcqmi2
[01-01_00:26:45:358] Find usbnet_adapter = eth2
[01-01_00:26:45:368] Get clientWDS = 7
[01-01_00:26:45:400] Get clientDMS = 8
[01-01_00:26:45:432] Get clientNAS = 9
[01-01_00:26:45:464] Get clientWDA = 10
[01-01_00:26:45:496] requestBaseBandVersion EC20CQAR02A03E2G_BETA0914  1  [Sep 14 2015 13:51:27]
[01-01_00:26:45:560] requestGetSIMStatus SIMStatus: SIM_READY
```

```
[01-01_00:26:45:624] requestGetProfile ctnet///0
[01-01_00:26:45:656] requestRegistrationState MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[01-01_00:26:45:688] requestQueryDataCall ConnectionStatus: DISCONNECTED
[01-01_00:26:45:720] requestRegistrationState MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[01-01_00:26:45:752] requestQueryDataCall ConnectionStatus: DISCONNECTED
[01-01_00:26:45:816] requestSetupDataCall WdsConnectionIPv4Handle: 0x43cc4478
[01-01_00:26:45:912] requestQueryDataCall ConnectionStatus: CONNECTED
[01-01_00:26:45:937] udhcpc (v1.20.2) started
[01-01_00:26:45:956] Sending discover...
[01-01_00:26:45:960] Sending select for 10.172.27.151...
[01-01_00:26:45:964] Lease of 10.172.27.151 obtained, lease time 7200
[01-01_00:26:45:984] deleting routers
route: SIOCDELRT: No such process
[01-01_00:26:46:003] adding dns 61.132.163.68
[01-01_00:26:46:003] adding dns 202.102.213.68
```

Step 4: Use the following commands to check IP/DNS/Route.

```
# ifconfig eth2
eth2      Link encap:Ethernet  HWaddr D2:B6:0C:28:AA:C6
          inet addr:10.172.27.151  Bcast:10.172.27.159  Mask:255.255.255.240
          inet6 addr: fe80::d0b6:cff:fe28:aac6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1224 (1.1 KiB)  TX bytes:1960 (1.9 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
0.0.0.0         10.172.27.145   0.0.0.0         UG    0      0        0 eth2
10.172.27.144   0.0.0.0         255.255.255.240 U     0      0        0 eth2

# 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: You can use the following command to terminate quectel-CM process to disconnect data connection:

```
# killall quectel-CM
[01-01_00:32:11:341] requestDeactivateDefaultPDP err = 0
[01-01_00:32:11:544] GobiNetThread exit
[01-01_00:32:11:545] main exit
```

6 FAQ and Kernel Log

6.1. How to Check Whether USB Driver Exists in Your Product

The content of directory "/sys/bus/usb/drivers" is USB drivers which exist in your product. Here is an 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 USB serial driver is required, please make sure "option" exists. If CDC ACM driver is required, please make sure "cdc_acm" exists. If GobiNet driver is required, please make sure "GobiNet" exists. 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 attaching the corresponding USB driver in Linux. If the module does not work well in your product, you can compare the kernel log in your product with the kernel log in this chapter to help you find the problem.

1. UC15/UC20/EC20/EC21/EC25 with USB Serial Driver

   Kernel logs of these modules are almost the same except for the VID&PID information (marked by read box in the following figure).

```
root@carl-OptiPlex-7010:/home/carl# dmesg
[ 1046.164307] usb 3-1: new high-speed USB device number 8 using xhci_hcd
[ 1046.183703] usb 3-1: New USB device found, idVendor=05c6, idProduct=9090
[ 1046.183708] usb 3-1: New USB device strings: Mfr=3, Product=2, SerialNumber=4
[ 1046.183711] usb 3-1: Product: UMTS/HSPA Module
[ 1046.183714] usb 3-1: Manufacturer: Quectel, Incorporated
[ 1046.191922] option 3-1:1.0: GSM modem (1-port) converter detected
[ 1046.192064] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB1
[ 1046.192161] option 3-1:1.1: GSM modem (1-port) converter detected
[ 1046.192338] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB2
[ 1046.192449] option 3-1:1.2: GSM modem (1-port) converter detected
[ 1046.192574] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB3
[ 1046.192667] option 3-1:1.3: GSM modem (1-port) converter detected
[ 1046.192791] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB4
[ 1046.192893] option 3-1:1.4: GSM modem (1-port) converter detected
[ 1046.193000] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB5
```

Figure 7: UC15/UC20/EC20/EC21/EC25 with USB Serial

2. UC20/EC20/EC21/EC25 with USB Serial and GobiNet Driver

   Kernel logs of these modules are almost the same except for the VID&PID information (marked by read box in the following figure).

```
root@carl-OptiPlex-7010:/home/carl# dmesg
[ 1144.533797] usb 3-1: new high-speed USB device number 9 using xhci_hcd
[ 1144.552092] usb 3-1: New USB device found, idVendor=05c6, idProduct=9003
[ 1144.552098] usb 3-1: New USB device strings: Mfr=3, Product=2, SerialNumber=4
[ 1144.552101] usb 3-1: Product: UMTS/HSPA Module
[ 1144.552103] usb 3-1: Manufacturer: Quectel, Incorporated
[ 1144.554387] option 3-1:1.0: GSM modem (1-port) converter detected
[ 1144.554488] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB1
[ 1144.554569] option 3-1:1.1: GSM modem (1-port) converter detected
[ 1144.554659] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB2
[ 1144.554731] option 3-1:1.2: GSM modem (1-port) converter detected
[ 1144.554839] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB3
[ 1144.554911] option 3-1:1.3: GSM modem (1-port) converter detected
[ 1144.554985] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB4
[ 1144.556332] GobiNet 3-1:1.4 eth1: register 'GobiNet' at usb-0000:00:14.0-1, Gob
iNet Ethernet Device, 06:7e:f7:9f:71:8e
[ 1147.588354] creating qcqmi1
```

Figure 8: UC20/EC20/EC21/EC25 with USB Serial and GobiNet

3. UC20/EC20/EC21/EC25 with USB Serial and QMI WWAN Driver

   Kernel logs of these modules are almost the same except for the VID&PID information (marked by read box in the following figure).

```
root@carl-OptiPlex-7010:/home/carl# dmesg
[ 1331.037072] usb 3-1: new high-speed USB device number 10 using xhci_hcd
[ 1331.055362] usb 3-1: New USB device found, idVendor=05c6, idProduct=9003
[ 1331.055368] usb 3-1: New USB device strings: Mfr=3, Product=2, SerialNumber=4
[ 1331.055371] usb 3-1: Product: UMTS/HSPA Module
[ 1331.055373] usb 3-1: Manufacturer: Quectel, Incorporated
[ 1331.057614] option 3-1:1.0: GSM modem (1-port) converter detected
[ 1331.057724] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB1
[ 1331.057796] option 3-1:1.1: GSM modem (1-port) converter detected
[ 1331.057888] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB2
[ 1331.057952] option 3-1:1.2: GSM modem (1-port) converter detected
[ 1331.058041] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB3
[ 1331.058102] option 3-1:1.3: GSM modem (1-port) converter detected
[ 1331.058195] usb 3-1: GSM modem (1-port) converter now attached to ttyUSB4
[ 1331.059426] qmi_wwan 3-1:1.4: cdc-wdm0: USB WDM device
[ 1331.060565] qmi_wwan 3-1:1.4 wwan0: register 'qmi_wwan' at usb-0000:00:14.0-1,
WWAN/QMI device, 06:7e:f7:9f:71:8e
```

Figure 9: UC20/EC20/EC21/EC25 with USB Serial and QMI WWAN

4. UGxx with CDC ACM Driver

```
root@carl-OptiPlex-7010:/home/carl# dmesg
[ 1598.042312] usb 3-1: new high-speed USB device number 11 using xhci_hcd
[ 1598.060159] usb 3-1: config 1 interface 0 altsetting 0 endpoint 0x81 has an invalid bInt
erval 255, changing to 11
[ 1598.060166] usb 3-1: New USB device found, idVendor=058b, idProduct=0041
[ 1598.060169] usb 3-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 1598.080571] cdc_acm 3-1:1.0: This device cannot do calls on its own. It is not a modem.
[ 1598.080639] cdc_acm 3-1:1.0: ttyACM0: USB ACM device
[ 1601.696555] usb 3-1: USB disconnect, device number 11
[ 1601.696609] usbcore: registered new interface driver cdc_acm
[ 1601.696614] cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters
[ 1603.094201] usb 3-1: new high-speed USB device number 12 using xhci_hcd
[ 1603.122232] usb 3-1: New USB device found, idVendor=1519, idProduct=0020
[ 1603.122237] usb 3-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 1603.122240] usb 3-1: Product: 7 CDC-ACM
[ 1603.122243] usb 3-1: Manufacturer: Comneon
[ 1603.122245] usb 3-1: SerialNumber: 004999010649993
[ 1603.153758] cdc_acm 3-1:1.0: This device cannot do calls on its own. It is not a modem.
[ 1603.153791] cdc_acm 3-1:1.0: ttyACM0: USB ACM device
[ 1603.155535] cdc_acm 3-1:1.2: This device cannot do calls on its own. It is not a modem.
[ 1603.155605] cdc_acm 3-1:1.2: ttyACM1: USB ACM device
[ 1603.157530] cdc_acm 3-1:1.4: This device cannot do calls on its own. It is not a modem.
[ 1603.157599] cdc_acm 3-1:1.4: ttyACM2: USB ACM device
[ 1603.159036] cdc_acm 3-1:1.6: This device cannot do calls on its own. It is not a modem.
[ 1603.159106] cdc_acm 3-1:1.6: ttyACM3: USB ACM device
[ 1603.161280] cdc_acm 3-1:1.8: This device cannot do calls on its own. It is not a modem.
[ 1603.161347] cdc_acm 3-1:1.8: ttyACM4: USB ACM device
[ 1603.163114] cdc_acm 3-1:1.10: This device cannot do calls on its own. It is not a modem.
[ 1603.163180] cdc_acm 3-1:1.10: ttyACM5: USB ACM device
[ 1603.164474] cdc_acm 3-1:1.12: This device cannot do calls on its own. It is not a modem.
[ 1603.164548] cdc_acm 3-1:1.12: ttyACM6: USB ACM device
```

Figure 10: UGxx with CDC ACM

7 Appendix A References

Table 3: Terms and Abbreviations

Abbreviations	Descriptions
OS	Operating System
PID	Product ID
VID	Vendor ID
PPP	Point to Point Protocol
CDC	Communications Device Class
ACM	Abstract Control Model
NDIS	Network Driver Interface Specification
NMEA	National Marine Electronics Association
PC	Personal Computer