WCDMA&LTE Linux USB Driver User Guide

UMTS/HSPA/LTE Module Series

Rev. WCDMA&LTE_Linux_USB_Driver_User_Guide_V1.2

Date: 2015-04-07

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
Mail: info@quectel.com

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

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

GENERAL NOTES

QUECTEL OFFERS THIS 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	2015-04-07	Kent XU	Added Zero Packet feature in Section 3.2.3 and 3.3.3

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 suitable for Quectel UC20, UC15, EC20 and UGxx modules. The following table shows the details.

Table 1: Supported Products

Product	Driver	Supported	Note
UC20	USB Serial	√	Refer to the Chapter 3.2 for USB Serial driver
UC15	USB Serial	√	Refer to the Chapter 3.2 for USB Serial driver
EC20	USB Serial	√	Refer to the Chapter 3.2 for USB Serial driver
UGxx	CDC ACM	√	Refer to the Chapter 3.3 for CDC ACM driver

2 Products Overview

USB on Quectel UMTS/HSPA/LTE module will create several different functional interfaces. Table 2 describes function of these interfaces.

Table 2: Interface Description

DM Interface	Diagnose port
NMEA Interface	For GPS NMEA message output
AT Interface	For AT commands
Modem Interface	For PPP connections and AT commands
NDIS Interface	Network Driver Interface
Trace Interface	For trace interface

Table 3 describes the interface information of different modules in the Linux system.

Table 3: Interface Information

Product	Driver	Interface
UC20 VID:0x05c6 PID:0x9003	USB Serial	ttyUSB0 -> DM
		ttyUSB1 -> NMEA
		ttyUSB2 -> AT
		ttyUSB3 -> Modem
		ttyUSB4 -> NDIS
UC15 VID:0x05c6 PID:0x9090	USB Serial	ttyUSB0 -> DM
		ttyUSB1 -> Reserved
		ttyUSB2 -> AT

		ttyUSB3 -> Modem
		ttyUSB4 -> Reserved
EC20 VID:0x05c6 PID:0x9215	USB Serial	ttyUSB0 -> DM
		ttyUSB1 -> NMEA
		ttyUSB2 -> AT
		ttyUSB3 -> Modem
		ttyUSB4 -> NDIS
UGxx VID:0x1519 PID:0x0020	CDC ACM	ttyACM0 -> Modem
		ttyACM1 -> Trace 1
		ttyACM2 -> Trace 2
		ttyACM3 -> AT 1
		ttyACM4 -> AT 2
		ttyACM5 -> Reserved
		ttyACM6 -> Reserved

3 System Setup

This part mainly describes the general organization of the USB stack in Linux and how to use USB serial and CDC ACM driver. Meanwhile, 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.

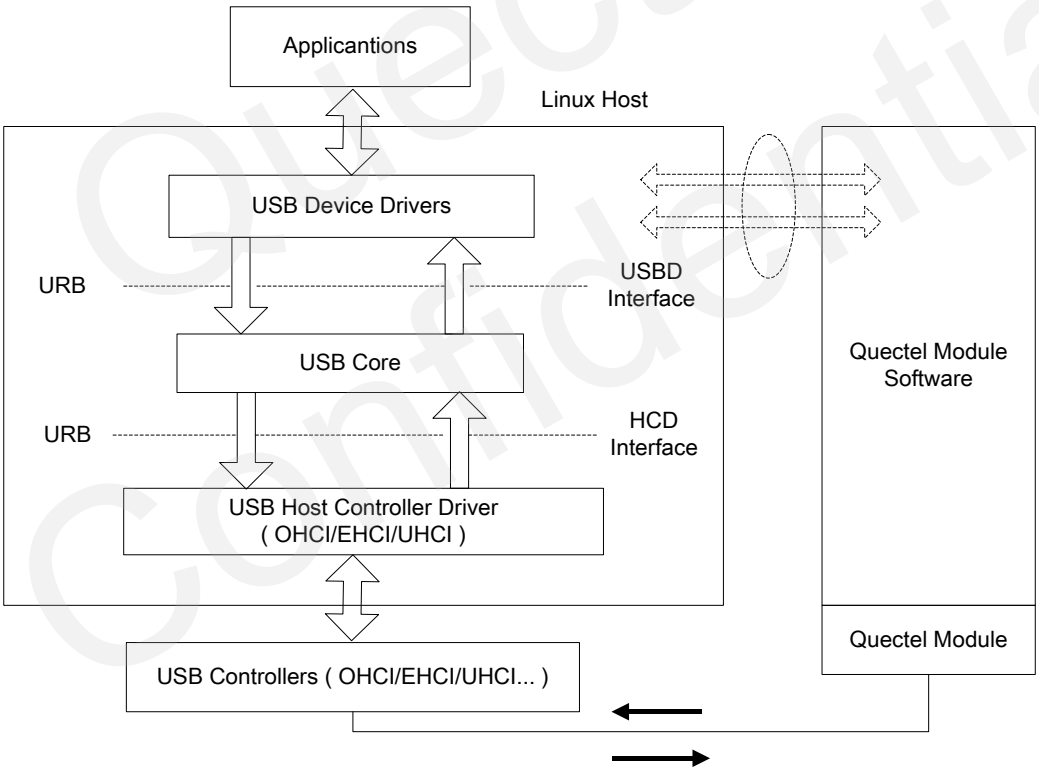

Figure 1: USB Driver Structure

The 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, providing 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

If you use USB serial driver, please read this section for details. Otherwise, please skip this section.

When device is attached to the USB Serial driver, driver will create device files in directory “/dev”, named as below:

ttyUSB0/ttyUSB1/ttyUSB2…

The following part shows how to integrate USB Serial driver.

3.2.1. Modify Driver Source Code

USB Serial driver source code is in the directory "[KERNEL]/drivers/usb/serial" and file name is "option.c". In order to recognize Quectel module, you should add module PID and VID information in "option.c" as below:

```
static const struct usb_device_id option_ids[] = {
    { USB_DEVICE( <you module’s VID>, <your module’s PID> ) },
```

3.2.2. 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:

```
make menuconfig
```

Step 3:

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

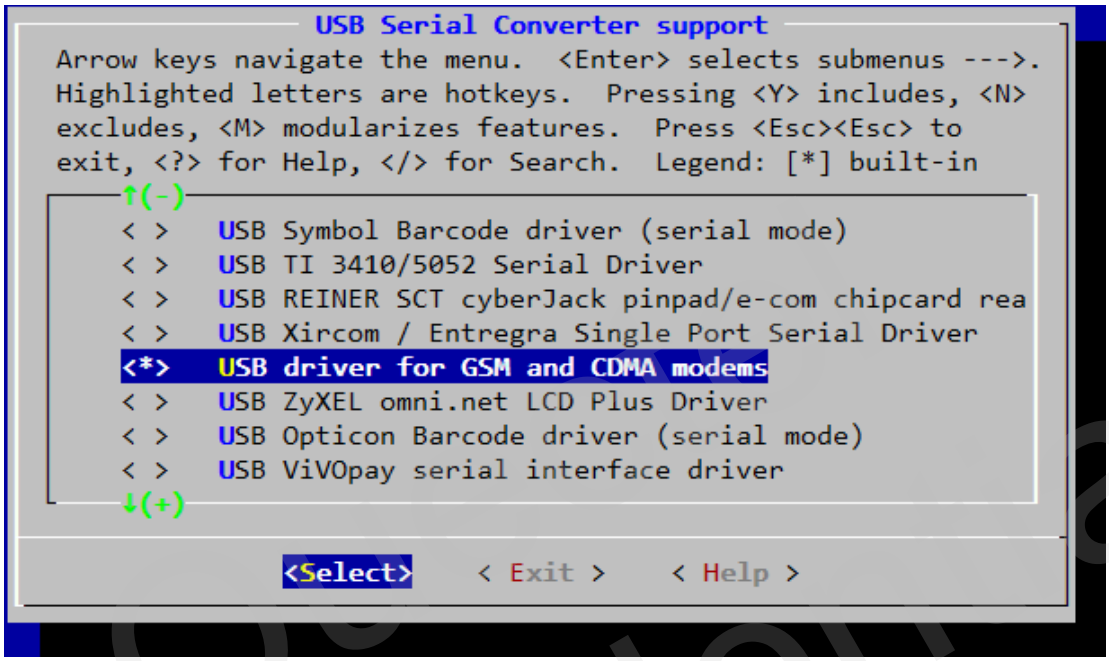

Figure 2: Configure USB Serial in Kernel

3.2.3. Add the Zero Packet Mechanism

As required by the USB protocol, you need to add the mechanism for processing zero packets during transmission of “usb_wwan.c” file under “[KERNEL]/drivers/usb/serial”.

However, zero packet mechanism is valid for **Linux 2.6.35** or **later** versions. This document takes the **Linux 3.2** as an example, the other versions could be a bit different, but it is basically the same.

You need to add the following statements to the “**usb_wwan_setup_urb**” function, as shown below:

```
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 *))
{
    struct urb *urb;
```

```
    if (endpoint == -1)
        return NULL;      /* endpoint not needed */
    urb = usb_alloc_urb(0, GFP_KERNEL);      /* No ISO */
    if (urb == NULL) {
        dbg("%s: alloc for endpoint %d failed.", __func__, endpoint);
        return NULL;
    }
    /* Fill URB using supplied data. */
    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 == 0x05C6 && (desc->idProduct == 0x9003 || desc->idProduct ==
0x9090 || desc->idProduct == 0x9215))
            urb->transfer_flags |= URB_ZERO_PACKET;
    }
    #end
    return urb;
}
```

3.3. CDC ACM Driver

If you use CDC ACM driver, please read this section for details. Otherwise, please skip this section.

When device is attached to CDC ACM driver, driver will create device files in directory “/dev”, named as below:

ttyACM0/ttyACM1/ttyACM2…

The following part shows 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 descriptor, so you do not need to add PID and VID information in driver source code.

3.3.2. 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:

```
make menuconfig
```

Step 3:

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

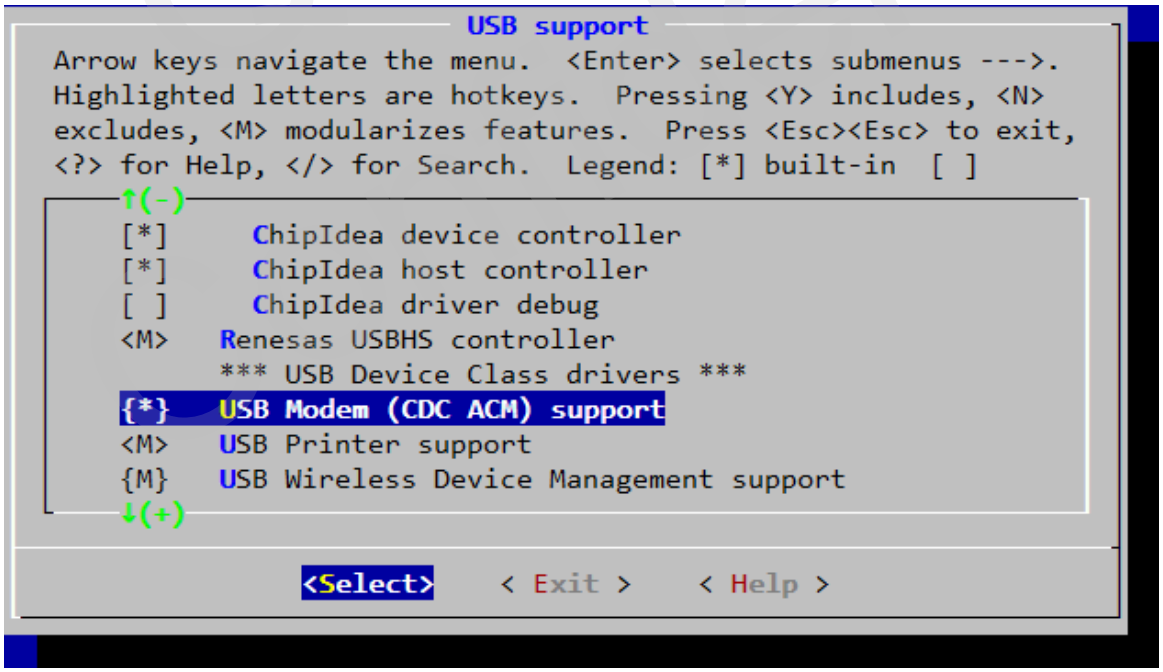

Figure 3: Configure CDC ACM Driver in Kernel

3.3.3. Add the Zero Packet Mechanism

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

This document takes the **Linux 3.2** as an example, the other versions could be a bit different, but it is 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)

.....
```

3.4. 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:

```
make menuconfig
```

Step 3:

```
[*] Device Drivers →

    [*] Network device support →

        [*] PPP (point-to-point protocol) support
```

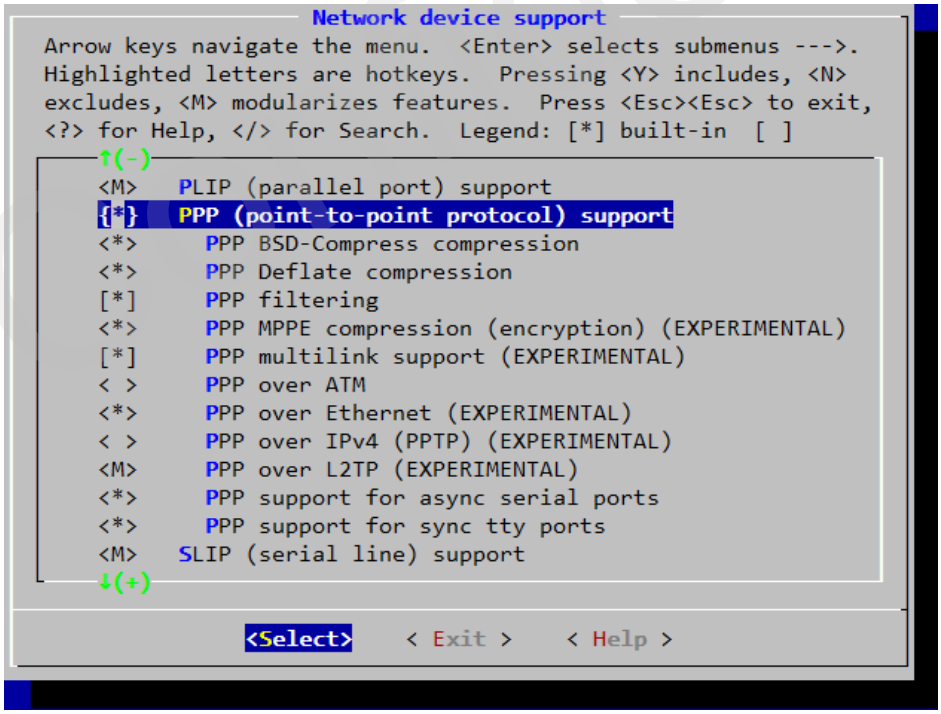

Figure 4: Configure PPP in Kernel

3.5. Compile and Load the Driver

If driver is for Linux on PC, please refer to 3.5.1, and if driver is for Linux on embedded systems, please refer to 3.5.2.

3.5.1. Driver for Linux on PC

Here shows how to compile driver for PC.

Step 1:

```
make bzImage
```

Step 2:

```
make modules
```

Step 3:

```
make modules_install
```

Step 4:

```
make install
```

Now reboot your PC and driver will be loaded.

3.5.2. Driver for Linux on Embedded Systems

Here shows how to compile driver for Linux on embedded systems.

```
make
```

After compiling successfully, you can find "zImage" in "$(kernel_src)/arch/arm/boot/". Download it to your embedded device and restart the device, the driver will be loaded.

4 Test the Module

Generally, AT and PPP functions will be used in you project. Here shows how to test these functions.

4.1. Test AT Function

After module is connected and USB driver is loaded successfully, there will create several device files in "/dev". Now you can use shell command to test AT function, as shown below:

Step 1:

cat *<AT port>* &

You should choose <AT port> according to your module, please refer to Table 3 for details.

Step 2:

echo "AT\r\n" > *<AT port>*

Now terminal will show the AT response.

Below is an example for UC20 and module AT port is ttyUSB2:

```
root@joe-OptiPlex-790:~# ls /dev/ttyUSB*
/dev/ttyUSB0  /dev/ttyUSB1  /dev/ttyUSB2  /dev/ttyUSB3  /dev/ttyUSB4
root@joe-OptiPlex-790:~# cat /dev/ttyUSB2 &
[1] 23393
root@joe-OptiPlex-790:~# echo -en "AT\r\n" > /dev/ttyUSB2
root@joe-OptiPlex-790:~# AT
OK
```

Figure 5: AT Test Result for UC20

Below is an example for UG95 and module AT port is ttyACM3:

```
root@joe-OptiPlex-790:~# ls /dev/ttyACM*
/dev/ttyACM0  /dev/ttyACM1  /dev/ttyACM2  /dev/ttyACM3  /dev/ttyACM4  /dev/ttyACM5  /dev/ttyACM6
root@joe-OptiPlex-790:~# cat /dev/ttyACM3 &
[2] 23353
root@joe-OptiPlex-790:~# echo -en "AT\r\n" > /dev/ttyACM3
root@joe-OptiPlex-790:~# AT
OK
```

Figure 6: AT Test Result for UG95

4.2. Test PPP Function

In order to setup PPP call, there are three scripts needed, they are "quectel-ppp", "quectel-chat-connect" and "quectel-chat-disconnect".

The content of the file "quectel_ppp" is shown as below:

```
#/etc/ppp/peers/quectel-ppp
# Usage:root>pppd call quectel-ppp
# Hide password in debug messages
hide-password
# The phone is not required to authenticate
noauth
# The chat script (be sure to edit that file,too!)
connect '/usr/sbin/chat -s -v -f /etc/ppp/peers/quectel-chat-connect'
# The close script(be sure to edit that file,too!)
disconnect '/usr/sbin/chat -s -v -f /etc/ppp/peers/quectel-chat-disconnect'
# Debug info from pppd
debug
# Serial Device to which the HSPDA phone is connected
#Modem path, like /dev/ttyUSB3,/dev/ttyACM0,it depend on your module.
<insert your module's Modem interface whole path>
# Serial port line speed
115200
# If you want to use the HSDPA link as your gateway
defaultroute
# pppd must not propose any IP address to the peer
noipdefault
# No ppp compression
novj
novjccomp
noccp
ipcp-accept-local
```

```
ipcp-accept-remote
local
# For sanity, keep a lock on the serial line
lock
dump
# Keep pppd attached to the terminal
# Comment this to get daemon mode pppd
nodetach
user <insert here the correct username for authentication>
password <insert here the correct password for authentication>
# Hardware flow control
crtscts
remotename 3gppp
ipparam 3gppp
# Ask the peer for up to 2 DNS server addresses
usepeerdns
```

The content of the file "quectel-chat-connect" is shown as below:

```
ABORT "BUSY"
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
TIMEOUT 120
"" AT
OK ATE0
OK ATI
OK AT+CSQ
OK AT+CPIN?
OK AT+COPS?
OK AT+CGREG?
OK ATZ
# Connection to the network
OK AT+CGDCONT=1,"IP","<insert here the correct APN provided by your network operator>",,0,0
# Dial the number
OK ATDT*99#
CONNECT
```

The content of the file "quectel-chat-disconnect" is shown as below:

```
ABORT "ERROR"
ABORT "NO DIALTONE"
SAY "\NSending break to the modem\n"
```

```
"" +++ATH
SAY "\nGood bye\n"
```

You should copy "quectel-ppp", "quectel-chat-connect" and "quectel-chat-disconnect" to the directory "/etc/ppp/peers". Then you can start to setup PPP call by below command:

```
pppd call quectel-ppp
```

The process of dialing is shown as below (example of UC20):

```
root@joe-OptiPlex-790:~# pppd call quectel-ppp
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 /usr/sbin/chat -s -v -f /etc/ppp/peers/quectel-chat-connect                 # (from
/etc/ppp/peers/quectel-ppp)
disconnect /usr/sbin/chat -s -v -f /etc/ppp/peers/quectel-chat-disconnect                    # (from
/etc/ppp/peers/quectel-ppp)
nocrtscts                   # (from /etc/ppp/peers/quectel-ppp)
local                 # (from /etc/ppp/peers/quectel-ppp)
asyncmap 0                  # (from /etc/ppp/options)
lcp-echo-failure 4                  # (from /etc/ppp/options)
lcp-echo-interval 30                # (from /etc/ppp/options)
hide-password               # (from /etc/ppp/peers/quectel-ppp)
novj                  # (from /etc/ppp/peers/quectel-ppp)
novjccomp                   # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-local                   # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-remote                  # (from /etc/ppp/peers/quectel-ppp)
ipparam 3gppp               # (from /etc/ppp/peers/quectel-ppp)
noipdefault                 # (from /etc/ppp/peers/quectel-ppp)
ipcp-max-failure 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)
noipx                 # (from /etc/ppp/options)
abort on (BUSY)
```

```
abort on (NO CARRIER)
abort on (NO DIALTONE)
abort on (ERROR)
abort on (NO ANSWER)
timeout set to 120 seconds
send (AT^M)
expect (OK)
^M
OK
 -- got it

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

send (ATI^M)
expect (OK)
^M
Quectel^M
EC20^M
Revision: EC20EQAR01A01E2G^M
^M
OK
 -- got it

send (AT+CSQ^M)
expect (OK)
^M
+CSQ: 24,99^M
^M
OK
 -- got it

send (AT+CPIN?^M)
expect (OK)
^M
+CPIN: READY^M
^M
OK
 -- got it

send (AT+COPS?^M)
```

```
expect (OK)
^M
+COPS: 0,0,"CHN-UNICOM",2^M
^M
OK
 -- got it

send (AT+CGREG?^M)
expect (OK)
^M
+CGREG: 0,1^M
^M
OK
 -- got it

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

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

Script /usr/sbin/chat -s -v -f /etc/ppp/peers/quectel-chat-connect finished (pid 11384), status = 0x0
Serial connection established.
using channel 4
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB3
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x2b4c7334> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x6 <asyncmap 0x0> <auth chap MD5> <magic 0x66fcba94> <pcomp>
<accomp>]
sent [LCP ConfAck id=0x6 <asyncmap 0x0> <auth chap MD5> <magic 0x66fcba94> <pcomp>
<accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x2b4c7334> <pcomp> <accomp>]
sent [LCP EchoReq id=0x0 magic=0x2b4c7334]
rcvd [LCP DiscReq id=0x7 magic=0x66fcba94]
rcvd [CHAP Challenge id=0x1 <7458cea8407e1183f8de77f1a0e7cff4>, name = "UMTS_CHAP_SRVR"]
sent [CHAP Response id=0x1 <36b99fd1857f541d972737fa57cbe45e>, name = "test"]
rcvd [LCP EchoRep id=0x0 magic=0x66fcba94 2b 4c 73 34]
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 ConfNak id=0x1 <ms-dns1 10.11.12.13> <ms-dns2 10.11.12.14> <ms-wins 10.11.12.13>
<ms-wins 10.11.12.14>]
sent [IPCP ConfReq id=0x2 <addr 0.0.0.0> <ms-dns1 10.11.12.13> <ms-dns2 10.11.12.14> <ms-wins
10.11.12.13> <ms-wins 10.11.12.14>]
rcvd [IPCP ConfReq id=0x4]
sent [IPCP ConfNak id=0x4 <addr 0.0.0.0>]
rcvd [IPCP ConfRej id=0x2 <ms-wins 10.11.12.13> <ms-wins 10.11.12.14>]
sent [IPCP ConfReq id=0x3 <addr 0.0.0.0> <ms-dns1 10.11.12.13> <ms-dns2 10.11.12.14>]
rcvd [IPCP ConfReq id=0x5]
sent [IPCP ConfAck id=0x5]
rcvd [IPCP ConfNak id=0x3 <addr 10.25.72.216> <ms-dns1 58.242.2.2> <ms-dns2 218.104.78.2>]
sent [IPCP ConfReq id=0x4 <addr 10.25.72.216> <ms-dns1 58.242.2.2> <ms-dns2 218.104.78.2>]
rcvd [IPCP ConfAck id=0x4 <addr 10.25.72.216> <ms-dns1 58.242.2.2> <ms-dns2 218.104.78.2>]
Could not determine remote IP address: defaulting to 10.64.64.64
not replacing existing default route via 192.168.10.1
local  IP address 10.25.72.216
remote IP address 10.64.64.64
primary     DNS address 58.242.2.2
secondary DNS address 218.104.78.2
Script /etc/ppp/ip-up started (pid 11399)
Script /etc/ppp/ip-up finished (pid 11399), status = 0x0
```

Now PPP call is setup successfully.

5 Appendix A Reference

Table 4: Terms and Abbreviations

Abbreviation	Description
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