UC20

Embedded Linux USB Driver User Guide

UMTS/HSPA Module Series

Rev. UC20_EMBEDDED_LINUX_USB_DRIVER_USER_GUIDE_V1.0

Date: 2013-06-04
Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarter:

Quectel Wireless Solutions Co., Ltd.
Room 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

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 ARE SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT
THIS INFORMATION CONTAINED HERE IS PROPRIETARY TECHNICAL INFORMATION OF QUECTEL CO., LTD. TRANSMITTABLE, REPRODUCTION, DISSEMINATION AND EDITING OF THIS DOCUMENT AS WELL AS UTILIZATION OF THIS CONTENTS ARE FORBIDDEN WITHOUT PERMISSION. OFFENDERS WILL BE HELD LIABLE FOR PAYMENT OF DAMAGES. ALL RIGHTS ARE RESERVED IN THE EVENT OF A PATENT GRANT OR REGISTRATION OF A UTILITY MODEL OR DESIGN.

Copyright © Quectel Wireless Solutions Co., Ltd. 2013. All rights reserved.
About the document

History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2012-05-31</td>
<td>Clare CHEN</td>
<td>Initial</td>
</tr>
</tbody>
</table>
Contents

About the document ................................................................................................................................... 2
Contents ....................................................................................................................................................... 3
Table Index ................................................................................................................................................... 4
Figure Index ................................................................................................................................................. 5

1 Introduction .......................................................................................................................................... 6

2 Product Overview ................................................................................................................................ 7

3 System Setup ....................................................................................................................................... 8
  3.1. Linux USB Drivers Structure ...................................................................................................... 8
  3.2. Building the driver ...................................................................................................................... 9
    3.2.1. Install cross compiler ........................................................................................................ 9
    3.2.2. Modify the source code file of Linux kernel ...................................................................... 9
    3.2.3. Modify kernel configuration ............................................................................................ 10
    3.2.4. Compiling the kernel ...................................................................................................... 13
  3.3. Loading the driver .................................................................................................................... 13

4 Application of reference .................................................................................................................... 15
  4.1. Modifying the rights of the devices port ................................................................................... 15
  4.2. Testing AT commands on the devices port .............................................................................. 15
  4.3. Create a PPP connection ........................................................................................................ 15

5 Appendix A Reference ....................................................................................................................... 18
Table Index

TABLE 1: INTERFACE DESCRIPTION ............................................................................................................... 7
TABLE 2: RELATIONSHIP BETWEEN INTERFACES AND DEVICES ............................................................ 14
TABLE 3: TERMS AND ABBREVIATIONS ........................................................................................................ 18
Figure Index

FIGURE 1: USB DRIVER STRUCTURE ................................................................. 8
FIGURE 2: ADD UC20 SUPPORT ....................................................................... 10
FIGURE 3: KERNEL CONFIGURATION – SELECT DEVICE DRIVERS ............... 11
FIGURE 4: KERNEL CONFIGURATION – SELECT USB SUPPORT .................... 11
FIGURE 5: KERNEL CONFIGURATION – SELECT USB SERIAL CONVERTER SUPPORT ........ 12
FIGURE 6: KERNEL CONFIGURATION – SELECT USB GENERIC SERIAL DRIVER .......... 12
FIGURE 7: KERNEL CONFIGURATION – SELECT USB DRIVER FOR GSM AND CDMA MODEMS .... 13
1 Introduction

This document introduces how to generate the USB driver for UC20 module in Embedded Linux OS, and how to use the module after the USB driver being loaded successfully.
2 Product Overview

Quectel UC20 is a wireless WCDMA modem. You can use it to implement some functions such as VOICE CALL and browsing internet and so on.

In general, the UC20 module will create five interfaces when you connect it with embedded equipments. These five interfaces have different functionalities. The details are shown as below:

Table 1: Interface Description

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM interface</td>
<td>Diagnose port</td>
</tr>
<tr>
<td>NMEA interface</td>
<td>For GPS NMEA sentence output</td>
</tr>
<tr>
<td>AT interface</td>
<td>For AT commands</td>
</tr>
<tr>
<td>Modem interface</td>
<td>For PPP connections and AT commands</td>
</tr>
<tr>
<td>NDIS interface</td>
<td>Network driver interface</td>
</tr>
</tbody>
</table>

**NOTES**

The NDIS interface is temporarily unavailable.
3 System Setup

Linux OS includes a generic USB to serial driver for modem. You can make the UC20 module available in the Embedded Linux OS by adding some kernel configuration items and adding the information (VID/PID) in Linux kernel.

The first part of this chapter is to describe the structure of Linux USB Driver and the rest is to explain how to build the USB driver for UC20 module.

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.

![USB Driver Structure Diagram](image)

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. Building the Driver

During the development based on embedded Linux OS, you must retrieve the Linux kernel source code files and install an appropriate cross compiler first, then modify the kernel configuration and corresponding source code files, and compile the kernel to generate image file, then burn the file into the target machine (The OS of the target machine is Android 4.0.3, and the corresponding Linux kernel version is 3.0.8). The detailed steps are shown as below:

#### 3.2.1. Install Cross Compiler

Cross-compilation is an important technology for embedded development. Its feature is that the source code files are not compiled in native machine but the other one. In general we call the former target machine and the latter host machine.

The reason of adopting cross-compilation is that most embedded target system cannot provide enough resources for compiling source code files, so we have to do that in a high-performance host machine in which we will create an environment of cross-compilation for the target machine.

In general, the vendor of the embedded machine would provide the cross compiler and the installment method about it. Here, we use the cross compiler "arm-linux-gcc-4.5.1", install it and add the compiler’s path in the system environment variables, then re-login system, then you can use the cross compiler to compile the source code files.

#### 3.2.2. Modify the Source Code File of Linux Kernel

Modify the source code file "option.c" in Linux kernel by adding VID and PID of UC20, so that the OS can recognize it.
The UC20’s VID and PID as follows:

- VID – 0x05c6
- PID – 0x9003

Open the file "option.c" in the path of "drivers\usb\serial" and find the struct array "struct usb_device_id option_ids[]". Insert "{USB_DEVICE (0x05c6,0x9003)}," to the array, then save and close it. The content of the file "option.c" is shown as below:

```
Figure 2: Add UC20 Support

3.2.3. Modify Kernel Configuration

Select the configuration items of USB to serial driver of the Linux kernel, so that the OS can support the UC20 module.

Retrieve the appropriate kernel source code version for your embedded system. Unpack it to your host machine and put it in its root directory type:

```
#make menuconfig
```

Configure the kernel according to the considered system configuration; then browse through the menus "Device Driver"→"USB Support"→ "USB Serial Converter support" and choose "USB Generic Serial Driver" and "USB driver for GSM and CDMA modems" as build-in , the illustration is shown as below:
Figure 3: Kernel Configuration – Select Device Drivers

Figure 4: Kernel Configuration – Select USB Support
Figure 5: Kernel Configuration – Select USB Serial Converter Support

Figure 6: Kernel Configuration – Select USB Generic Serial Driver
Figure 7: Kernel Configuration – Select USB Drivers for GSM and CDMA Modems

Make sure the mandatory items have been selected, then save and exit.

3.2.4. Compiling the Kernel

The last step of building the driver is to use the cross compiler to compile the kernel: locate the kernel’s root directory and type:

```
#make
```

After compiling the kernel successfully, the “Zimage” file will be created in the path 
"$(kernel_src)/arch/arm/boot/", then you can burn it into the target machine and connect the UC20 module to the machine.

3.3. Loading the Driver

When UC20 module is connected with the Linux Kernel System mentioned above, the system will firstly recognize UC20 module and read its device descriptor, then create five interface devices automatically, listed as below. After that, you can use these five interface devices.

- /dev/ttyUSB0
- /dev/ttyUSB1
- /dev/ttyUSB2
- /dev/ttyUSB3
You can check the result in the terminal, using the following command:

```
# ls /dev/ttyUSB*
```

If the five device node files are listed, it is certain that the UC20 module has been recognized by Linux/Android OS. And the corresponding relations to the interfaces of the devices are shown as below:

**Table 2: Relationship between Interfaces and Devices**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>Interface Name</th>
<th>Device Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DM interface</td>
<td>/dev/ttyUSB0</td>
</tr>
<tr>
<td>1</td>
<td>NMEA interface</td>
<td>/dev/ttyUSB1</td>
</tr>
<tr>
<td>2</td>
<td>AT interface</td>
<td>/dev/ttyUSB2</td>
</tr>
<tr>
<td>3</td>
<td>Modem interface</td>
<td>/dev/ttyUSB3</td>
</tr>
<tr>
<td>4</td>
<td>NDIS interface</td>
<td>/dev/ttyUSB4</td>
</tr>
</tbody>
</table>
4 Instructions for Use

After the USB driver of UC20 module being loaded successfully, you can make the applications of the UC20 module.

It is suggested that you dispose the VoiceCall and SMS service on AT interface and dispose the Data service on Modem interface.

4.1. Modifying the Rights of the Devices’ Port

Before using the UC20 module, make sure that the two ports possess readable, writable, and executive rights.

For example, type the commands below in the terminal

```bash
chmod 777 /dev/ttyUSB2
chmod 777 /dev/ttyUSB3
```

4.2. Testing AT commands on the Devices’ Port

You can use serial debugging tools to send AT commands, and check on the working of the UC20 module.

When you configure the serial debugging tools, the serial port must be "/dev/ttyUSB2" or "/dev/ttyUSB3 "and the sending data may be as follows:

Sending data: AT
Received data: OK

If the received data is "OK ", it proves that the UC20 module is available.

4.3. Create a PPP connection

In general, you should create a PPP connection before using the data service of UC20 modules. The command of creating a PPP connection terminal is shown as below:
# pppd call Module-UC20

The parameter Module-UC20 is a script file of PPP dial. In general, the PPP dial script files include three files: "Module-UC20", "Chat-Module-UC20-connect" and "Chat-Module-UC20-disconnect".

The content of the file "Module-UC20" is shown as below:

```
#/etc/ppp/peers/Module-UC20
# Usage: root>pppd call Module-UC20
# Keep pppd attached to the terminal
# Comment this to get daemon mode pppd
nodetach
# For sanity,keep a lock on the serial line
lock
# Serial Device to which the HSPDA phone is connected
/dev/ttyUSB3
# Serial port line speed
115200
user<insert here the correct username for authentication>
password <insert here the correct password for authentication>
# No hardware flow control
nocrtscts
# Ask the peer for up to 2 DNS server addresses
usepeerdns
# The phone is not required to authenticate
noauth
# pppd must not propose any IP address to the peer
noipdefault
# No ppp compression
novj
novjccomp
noccp
# If you want to use the HSDPA link as your gateway
defaultroute
ipcp-accept-local
ipcp-accept-remote
# The chat script(be sure to edit that file,too!)
connect 'chat -s -v -f /etc/ppp/peers/Chat-Module-UC20-connect'
# The close script(be sure to edit that file,too!)
disconnect 'chat -s -v -f /etc/ppp/peers/Chat-Module-UC20-disconnect'
```

The content of the file "Chat-Module-UC20-connect" is shown as below:

```
ABORT 'NO CARRIER'
```
ABORT 'ERROR'
ABORT 'NO DIALTONE'
ABORT 'BUSY'
ABORT 'NO ANSWER'
" AT
" ATE0
# Dial the number
OK ATD*99#
CONNECT "

The content of the file "Chat-Module-UC20-disconnect" is shown as below:

ABORT  OK
ABORT  BUSY
ABORT  DELAYED
ABORT  "NO ANSWER"
ABORT  "NO CARRIER"
ABORT  "NO DIALTONE"
ABORT  VOICE
ABORT  ERROR
ABORT  RINGING
TIMEOUT  12
" " \K
" " +++ATH
SAY "\nGoodbay\n"

After creating PPP connection successfully, you can browse internet with the default browser of Android OS.
5 Appendix A Reference

Table 3: Terms and Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PID</td>
<td>Product ID</td>
</tr>
<tr>
<td>VID</td>
<td>Vendor ID</td>
</tr>
</tbody>
</table>