Wireless Module Expert

UC20 Linux USB Driver User Guide

UMTS/HSPA Module Series

Rev. UC20_Linux_USB_Driver_User_Guide_V1.0

Date: 2013-06-09

www.quectel.com

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

About the document

History

Revision	Date	Author	Description
1.0	2012-06-03	Clare CHEN	Initial

Contents

Table Index

Figure Index

1 Introduction

This document introduces how to generate the USB driver for UC20 module in Linux OS, and how to use the module after the USB driver is loaded successfully.

2 Product Overview

Quectel UC20 is a wireless WCDMA module. With it, you can implement some functions such as VOICE CALL and browsing internet and so on.

In general, 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

DM interface	Diagnose port
NMEA interface	For GPS NMEA sentence output
AT interface	For AT commands
Modem interface	For PPP connections and AT commands
NDIS interface	Network driver interface

3 System Setup

Linux OS includes a generic USB to serial driver for communication module. You can use the module in the Linux OS only by adding the module's information (VID/PID) in the 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 Drivers 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, USB device drivers.

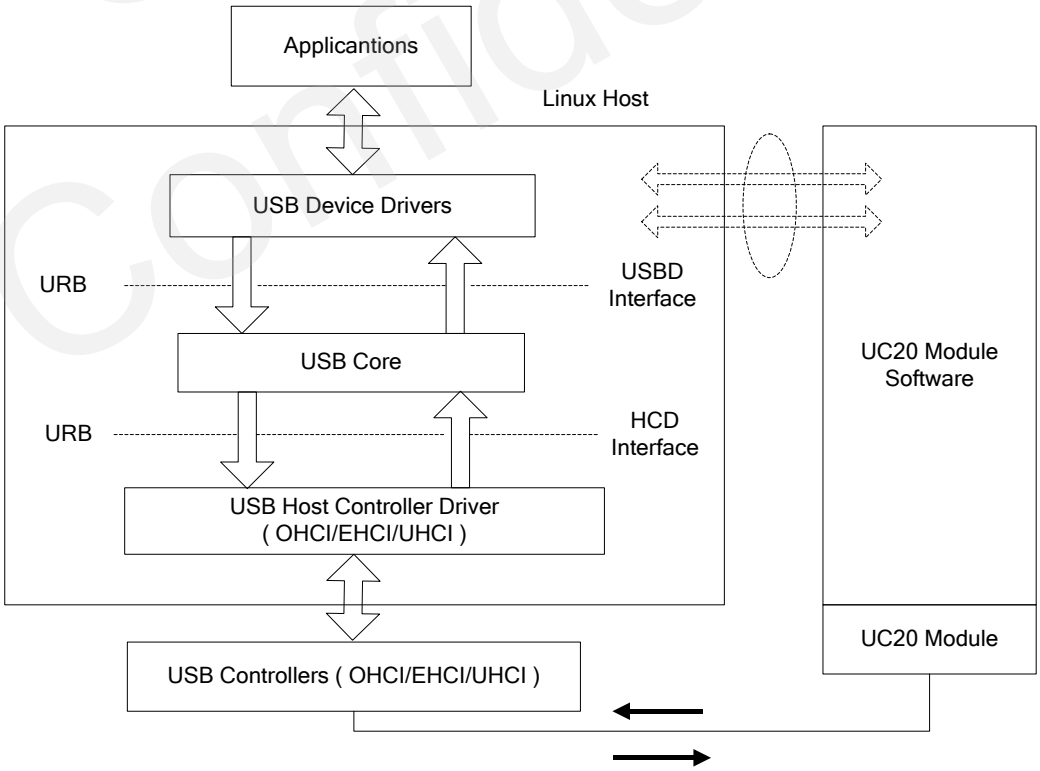

Figure 1: USB Driver structure

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. Building the Driver

The USB driver is a part of Linux kernel. You can load the USB to Serial Driver in two ways:

1. Compile the driver into kernel.
2. Load the driver module dynamically.

The second way is to load the driver module with the command "insmod" after compiling the corresponding source code files, but the premise is that you have had the Linux kernel source code files and you have already compiled the whole kernel.

Here we only introduce the first method. Before replacing the Linux kernel, you need to modify the kernel configuration and the source code files responding to USB to serial driver by adding the VID and PID of UC20 in the file. The detailed steps are shown as below. (Take Fedora 14 as an example)

3.2.1. Download and Modify Source Code of Linux Kernel

Download the Linux kernel file and decompress the file into the directory "/usr/src/kernels". You can decompress the file by executing the following commands.

```
#cd /usr/src/kernels
#tar –jxf linux-2.6.34.14.tar.bz2 –c /usr/src/kernels/
```

Then you should modify the driver source code file so that the UC20 module can be recognized by Linux OS. The file path is "/usr/src/kernels/driver/usb/serial/option.c". Open it and insert the VID and PID of UC20 module into the file, then save it and get ready to compile the kernel.

The UC20's VID and PID are listed as follows:

1. VID – 0x05c6
2. PID – 0x9003

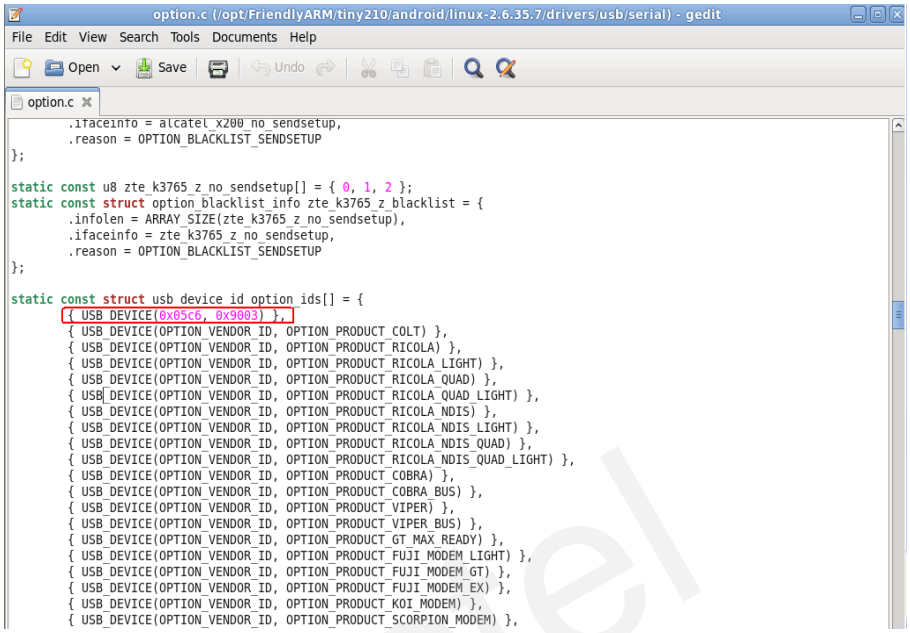

Figure 2: Add UC20 support

3.2.2. Clear Old Compiled File and Other Configuration File

Execute the following command in Terminal

```
#cd linux-2.6.34.14
#make mrproper
```

3.2.3. Modify Kernel Configuration

There are several mandatory selected items in kernel configuration:

USB Support

USB Support →

USB Serial Converter Support→

USB Generic Serial Driver

USB driver for GSM and CDMA modems

PPP Support

For PPP dialing function you must select the below items:

PPP (point-to-point protocol) support

And you'd better select the following items to support the modules' dynamic loading.

Enable loadable module support→

Forced module loading

Module unloading

Source checksum for all modules

Modify kernel configuration by executing the following command:

```
#make menuconfig
```

You must modify the kernel configuration manually, as shown in the following figures below:
Device Drivers → USB Support

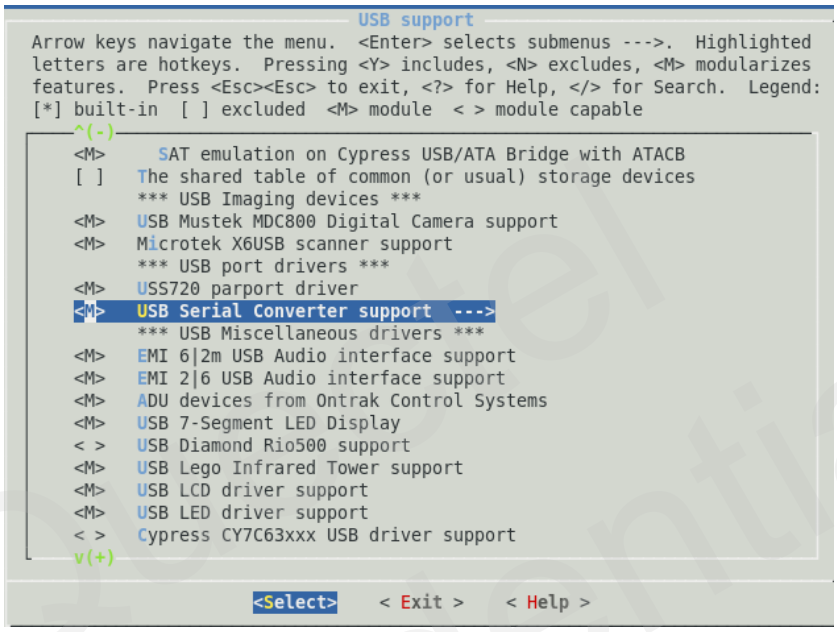

Figure 3: Select USB Serial Converter Support

Device Drivers → USB support → USB Serial Converter Support.

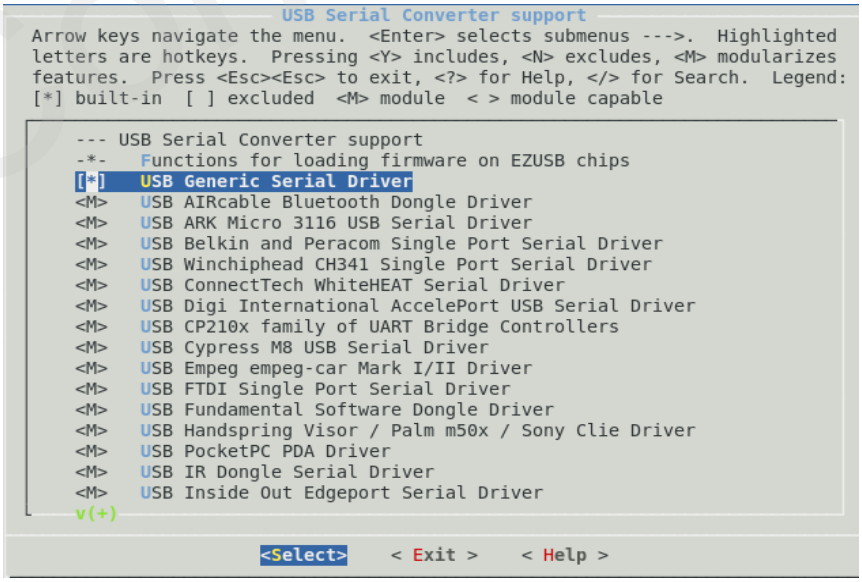

Figure 4: Select USB Generic Serial Driver

Device Drivers → USB support → USB Serial Converter Support.

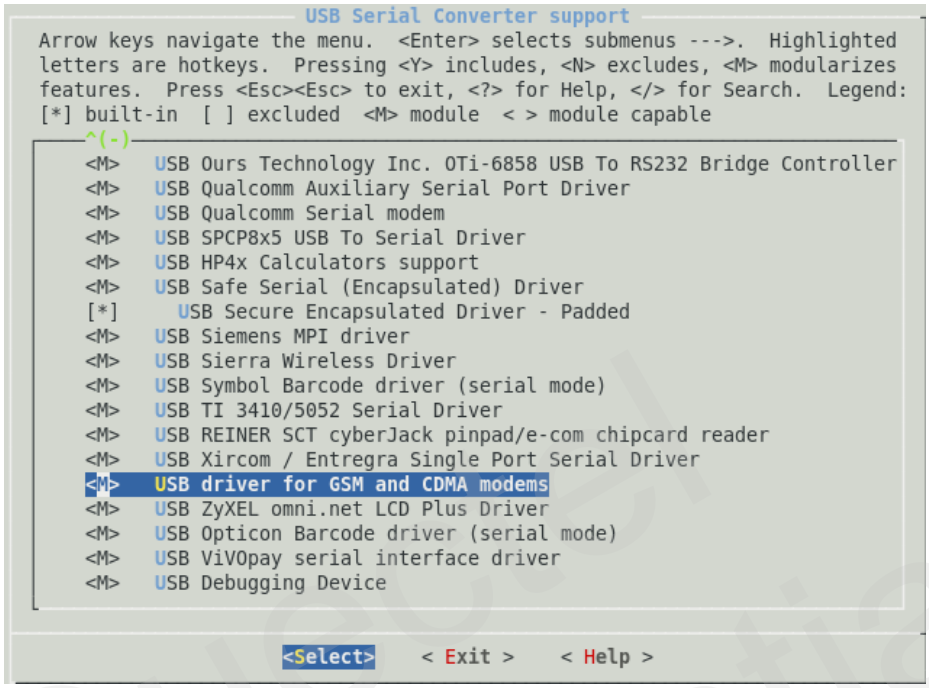

Figure 5: Select USB Driver for GSM and CDMA Modems

Enable Loadable Module Support

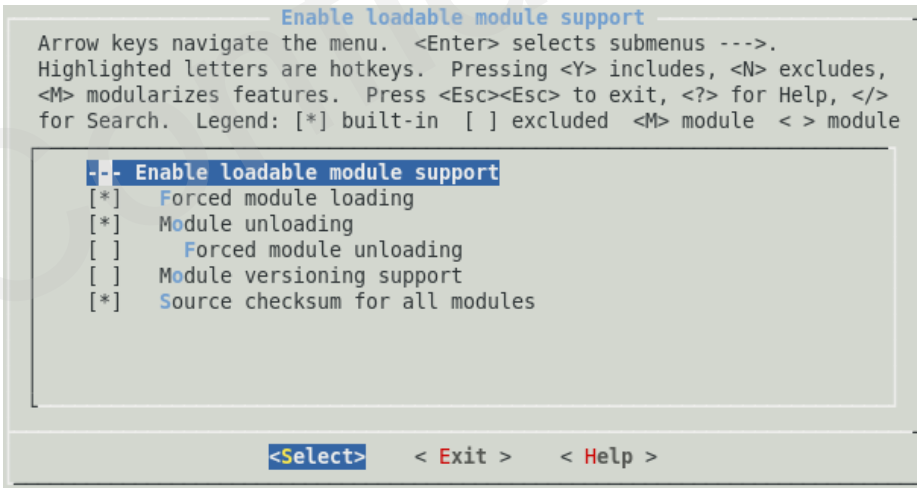

Figure 6: Select Enable Loadable Module Support

Device Drivers →Network Device Support

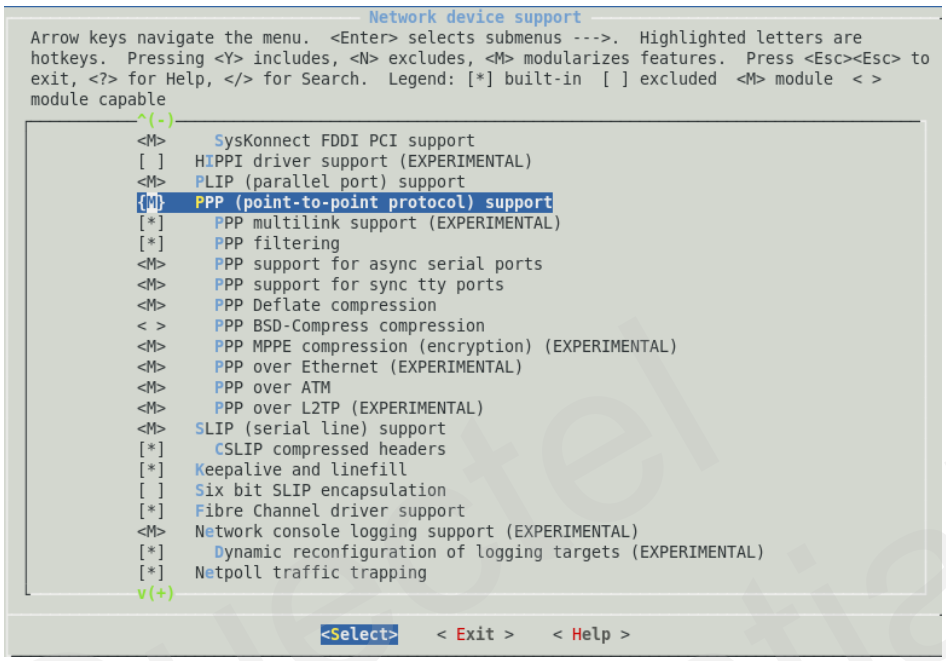

Figure 7: Select PPP (point-to-point protocol) Support

3.2.4. Compile and Install Linux Kernel

After finishing the above steps, you can compile and install the Linux kernel step by step as follows:

1. Before compiling, you should clean the configuration and executable file.

```
#make clean
```

2. Generate core file

```
#make bzImage
```

3. Make modules

```
#make modules
```

4. Install modules

```
#make modules_install
```

After installing successfully, all modules will be installed into the path "/lib/modules/$(uname –r)/ ".

5. Install kernel

There are two ways to install kernel: automatic way and manual way.

Automatic way: Just execute the following command in Terminal

```
#make install
```

Manual way: Available for the version earlier than 2.6.

```
#cp /usr/src/kernels/linux-2.6.34.14/arch/i386/boot/bzImage \
>/boot/vmlinuz-2.6.34.14-2

#cp /usr/src/kernels/linux-2.6.34.14/system.map \
>/boot/system.map-2.6.34.14-2

#mkinitrd –v /boot/initrd-2.6.33.img     2.6.33.2                    //create "initrd"
```

Edit the grub configuration file: " /boot/grub/menu.lst ". The content is shown as below:

mymenu.lst

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/vg_clare-lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=0
timeout=0
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Fedora (2.6.35.14-106.fc14.i686)
        root (hd0,0)
        kernel /vmlinuz-2.6.34.14 ro root=/dev/mapper/vg_clare-lv_root rd_LVM_LV=vg_clare/lv_root rd_LVM_LV=vg_clare/lv_swap
rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYTABLE=us rhgb quiet
        initrd /initramfs-2.6.35.14-106.fc14.i686.img
title Fedora (2.6.34.14)
        root (hd0,0)
        kernel /vmlinuz-2.6.34.14 ro root=/dev/mapper/vg_clare-lv_root rd_LVM_LV=vg_clare/lv_root rd_LVM_LV=vg_clare/lv_swap
rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYTABLE=us rhgb quiet
        initrd /initramfs-2.6.35.6-45.fc14.i686.img
```

Figure 8: Content of the Grub Configuration "menu.lst"

6. Restart the Linux OS.

After operating system is restarted, it will list all kernel versions. You only need to select the new kernel to enter the system.

3.3. Loading the Driver

When UC20 module was connected to host computer via USB，meanwhile，the USB deriver is loaded successfully, the following five devices will be created.

- /dev/ttyUSB0
- /dev/ttyUSB1
- /dev/ttyUSB2
- /dev/ttyUSB3
- /dev/ttyUSB4

You can check the result as below:

```
[root@clare /]# cd dev
[root@clare dev]# ls ttyUSB*
ttyUSB0  ttyUSB1  ttyUSB2  ttyUSB3  ttyUSB4
```

If the five device node files are shown as above, it is certain that the UC20 module has been recognized by Linux OS, and the corresponding relations between interfaces and devices are shown as below:

Table 2: Relationship between Interfaces and Devices

INDEX	Interface Name	Device Name
0	DM interface	/dev/ttyUSB0
1	NMEA interface	/dev/ttyUSB1
2	AT interface	/dev/ttyUSB2
3	Modem interface	/dev/ttyUSB3
4	NDIS interface	/dev/ttyUSB4

4 System Setup

After the USB driver of UC20 module is loaded successfully, you can use the functions of the UC20 module.

It is suggested that you dispose the Voice Call 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, you should make sure the two ports possess readable, writable and executable.

For example: type the following commands in the terminal.

```
Chomd 777 /dev/ttyUSB2
Chomd 777 /dev/ttyUSB3
```

4.2. Testing AT commands on the Devices' Port

In Linux OS, module's AT interface and Modem interface are serial ports. You can use the tool "minicom" to test the communication via the two ports. The method of using "minicom" is shown as below:

First, set the serial port configuration:

In the terminal type

```
# minicom –s
```

Then select "Serial port setup" in the configuration window, shown as below:

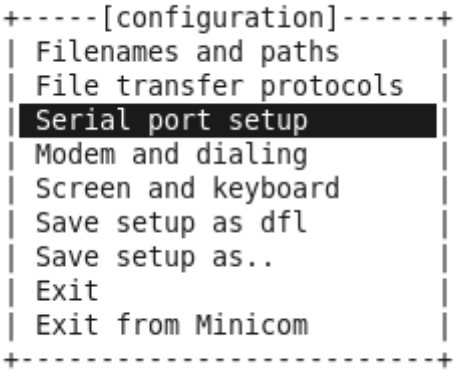

Figure 9: Serial Port Setup in minicom

Enter and set the "Serial Device" to be "**/dev/ttyUSB2**" or "**/dev/ttyUSB3**" in the following window.

Figure 10: Serial Port Configuration in minicom

After setting the tool's communicative parameters, you can use it to send AT commands.

In the Terminal type "minicom", press enter, and then you can input AT commands to send it in the following place.

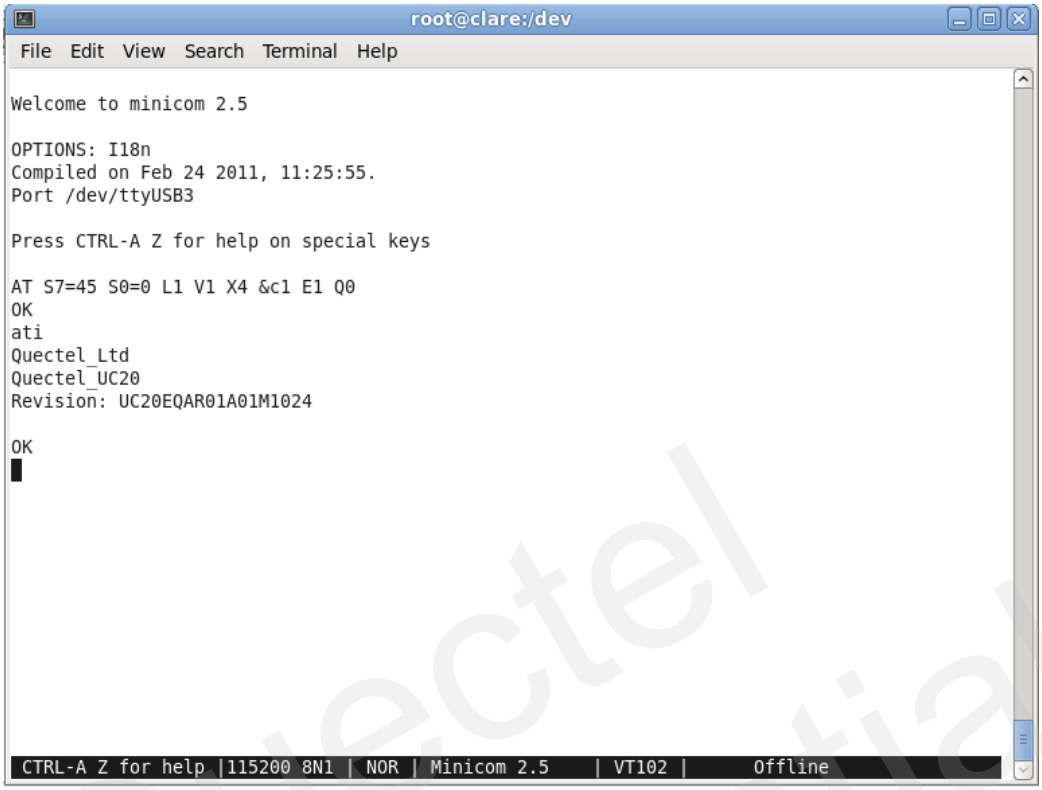

Figure 11: Send AT Command in minicom

To close the minicom, press " CTRL+A" and press " Q" a few seconds later, and select "Yes" to exit it.

4.3. Create a PPP Connection

In general, you should create a PPP connection before you apply data service into UC20 module.
The command of creating a PPP connection is shown as below:

```
# pppd call wcdma
```

The parameter "wcdma" is a script file of PPP dial. In general, the PPP dial script file includes three files, which are "wcdma", "wcdma-chat-connect", and "wcdma-chat-disconnect".

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

```
#/etc/ppp/peers/wcdma
# Usage:root>pppd call wcdma
# 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/wcdma-chat-connect'
# The close script(be sure to edit that file,too!)
disconnect '/usr/sbin/chat -s -v -f /etc/ppp/peers/wcdma-chat-disconnect'
# Debug info from pppd
debug
# Serial Device to which the HSPDA phone is connected
/dev/ttyUSB3
# 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 "wcdma-chat-connect" is shown as below:

```
ABORT "BUSY"
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
TIMEOUT 120
"" AT
OK \rATZ
# Connection to the network
OK \rAT+CGDCONT=1,"IP","<insert here the correct APN provided by your network operator>",,0,0
# Dial the number
OK-AT-OK ATDT*99#
CONNECT \d\c
```

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

```
ABORT "ERROR"
ABORT "NO DIALTONE"
SAY "\NSending break to the modem\n"
""\k"
""+++ATH"
SAY "\nGood bay\n"
```

After creating the three files under the path "/etc/ppp/peers", you can do the PPP dialing:

Execute the following command in Terminal:

#pppd call wcdma

The process of dialing is shown as below:

```
[root@clare linux-2.6.34.14]# pppd call wcdma
pppd options in effect:
debug           # (from /etc/ppp/peers/wcdma)
nodetach                # (from /etc/ppp/peers/wcdma)
dump            # (from /etc/ppp/peers/wcdma)
noauth          # (from /etc/ppp/peers/wcdma)
user            # (from /etc/ppp/peers/wcdma)
password ??????         # (from /etc/ppp/peers/wcdma)
remotename 3gppp                # (from /etc/ppp/peers/wcdma)
/dev/ttyUSB3            # (from /etc/ppp/peers/wcdma)
115200          # (from /etc/ppp/peers/wcdma)
lock            # (from /etc/ppp/peers/wcdma)
connect /usr/sbin/chat -s -v -f /etc/ppp/peers/wcdma-chat-connect               # (fr
om /etc/ppp/peers/wcdma)
disconnect /usr/sbin/chat -s -v -f /etc/ppp/peers/wcdma-chat-disconnect         # (fr
om /etc/ppp/peers/wcdma)
crtscts         # (from /etc/ppp/peers/wcdma)
local           # (from /etc/ppp/peers/wcdma)
hide-password           # (from /etc/ppp/peers/wcdma)
novj            # (from /etc/ppp/peers/wcdma)
novjccomp               # (from /etc/ppp/peers/wcdma)
ipcp-accept-local               # (from /etc/ppp/peers/wcdma)
ipcp-accept-remote              # (from /etc/ppp/peers/wcdma)
ipparam 3gppp           # (from /etc/ppp/peers/wcdma)
noipdefault             # (from /etc/ppp/peers/wcdma)
defaultroute            # (from /etc/ppp/peers/wcdma)
usepeerdns              # (from /etc/ppp/peers/wcdma)
noccp           # (from /etc/ppp/peers/wcdma)
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)
```

Figure 12: PPP Dial-1

Figure 13: PPP Dial-2

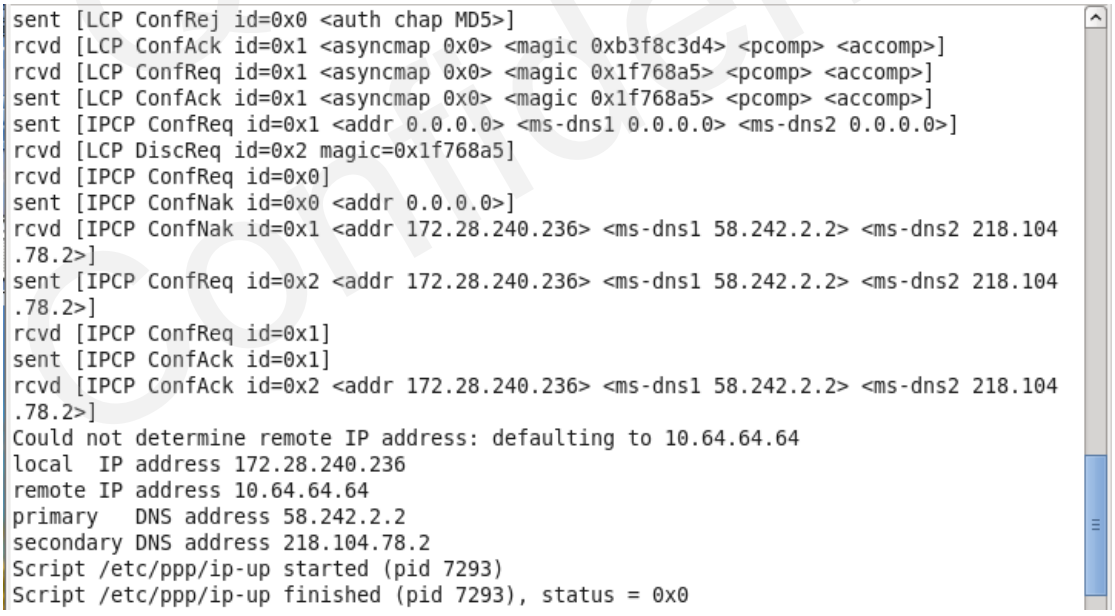

Figure 14: PPP Dial-3

After PPP connection is created successfully, you must add the DNS information in the file "etc/resolv.conf". The content to be inserted is shown as below:

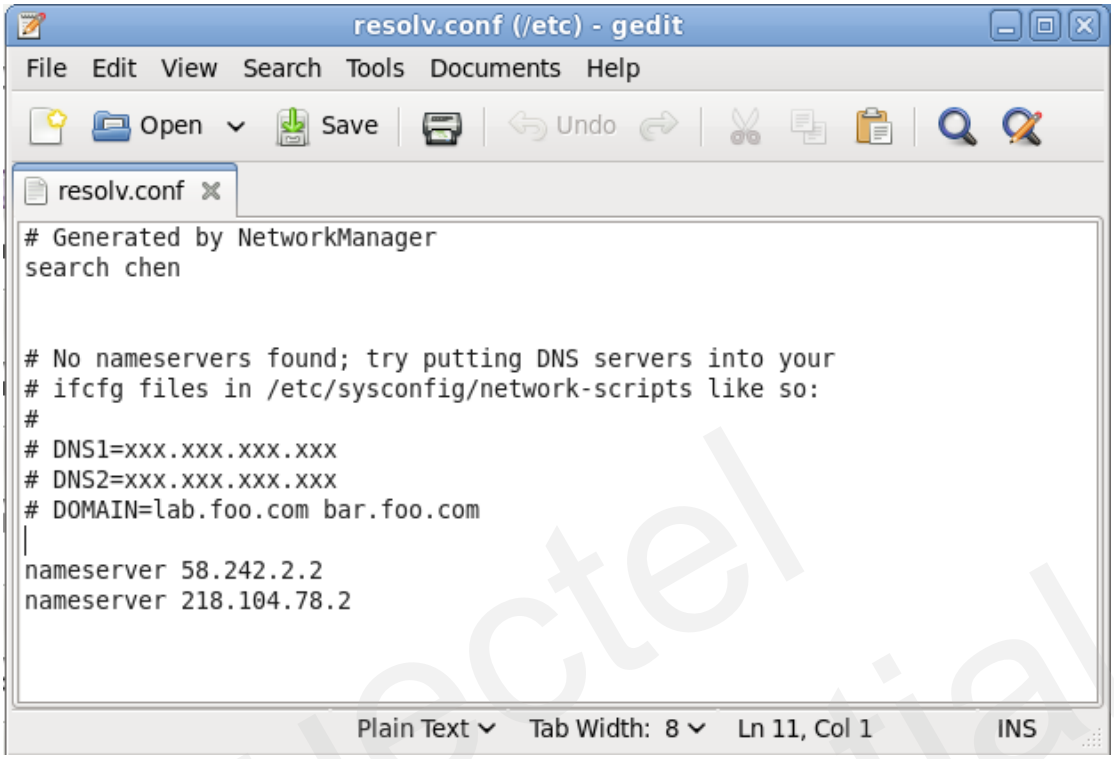

Figure 15: Content of the DNS Configuration File

Then you must set the default route as ppp0 before browsing internet. The commands are shown as below:

```
[root@clare ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.64.64.64     *               255.255.255.255 UH    0      0        0 ppp0
default         *               0.0.0.0         U     0      0        0 eth0
[root@clare ~]#
```

```
[root@clare ~]# route del default
[root@clare ~]# route add default dev ppp0
[root@clare ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.64.64.64     *               255.255.255.255 UH    0      0        0 ppp0
default         *               0.0.0.0         U     0      0        0 ppp0
```

Figure 16: route setting

Then you can browse internet:

Figure 17: Browsing Internet in Linux OS

5 Appendix A Reference

Table 3: Terms and Abbreviations

Abbreviation	Description
OS	Operating System
PID	Product ID
VID	Vendor ID