Wireless Module Expert

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

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

COPYRIGHT

About the document

History

Revision	Date	Author	Description
1.0	2012-05-31	Clare CHEN	Initial

Contents

Table Index

Figure Index

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

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

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.

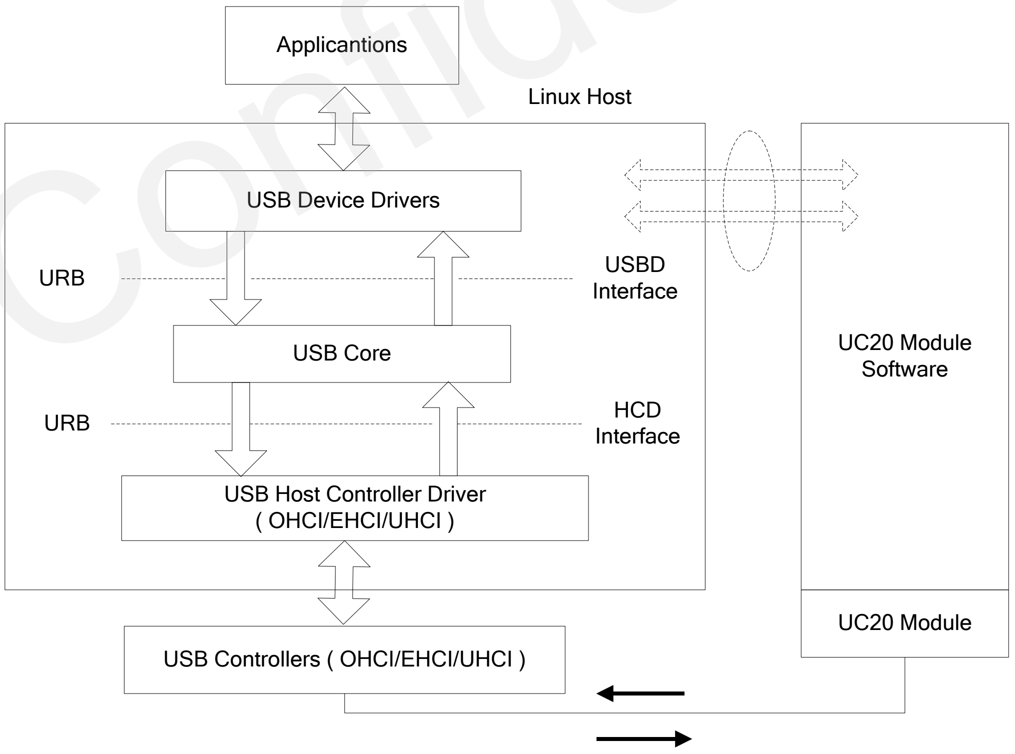


Figure 1: USB Driver Structure

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-logout system, then you can use the cross compiler to compile the source code files.

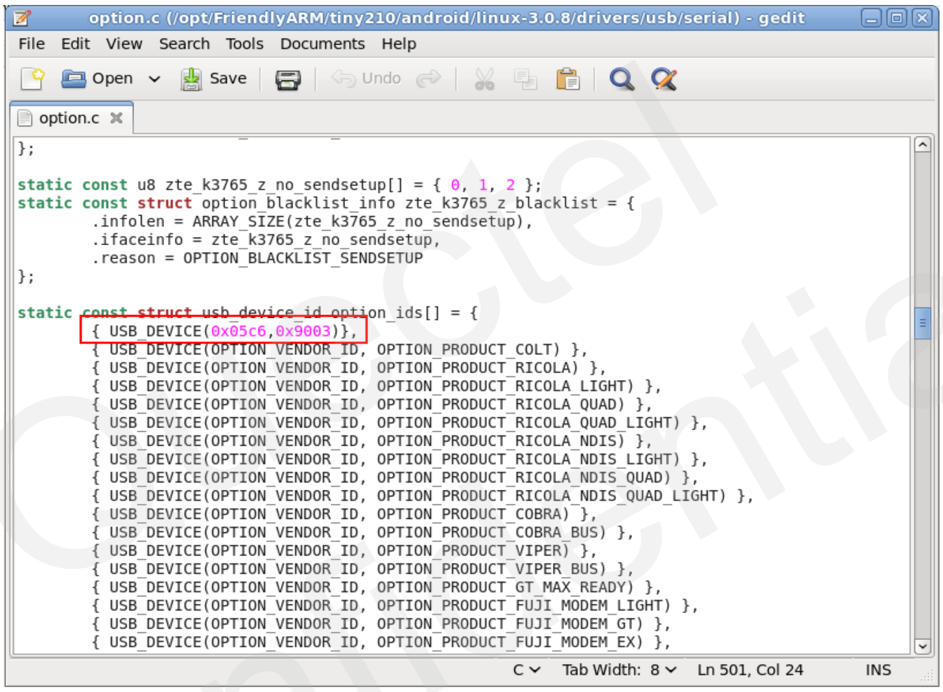
3.2.2. Modify the Source Code File of Linux Kernel

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 "static structusb_device_idoption_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:


option.c (/opt/FriendlyARM/tiny210/android/linux-3.0.8/drivers/usb/serial) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
option.c

```
};

static const u8 zte_k3765_z_no_sendsetup[] = { 0, 1, 2 };
static const struct option_blacklist_info zte_k3765_z_blacklist = {
        .infolen = ARRAY_SIZE(zte_k3765_z_no_sendsetup),
        .ifaceinfo = zte_k3765_z_no_sendsetup,
        .reason = OPTION_BLACKLIST_SENDSETUP
};

static const struct usb_device_id option_ids[] = {
        { USB_DEVICE(0x05c6,0x9003)},
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_COLT) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_LIGHT) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_QUAD) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_QUAD_LIGHT) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_NDIS) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_NDIS_LIGHT) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_NDIS_QUAD) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_NDIS_QUAD_LIGHT) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_COBRA) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_COBRA_BUS) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_VIPER) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_VIPER_BUS) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_GT_MAX_READY) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_FUJI_MODEM_LIGHT) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_FUJI_MODEM_GT) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_FUJI_MODEM_EX) },
```

Figure 2: Add UC20 Support

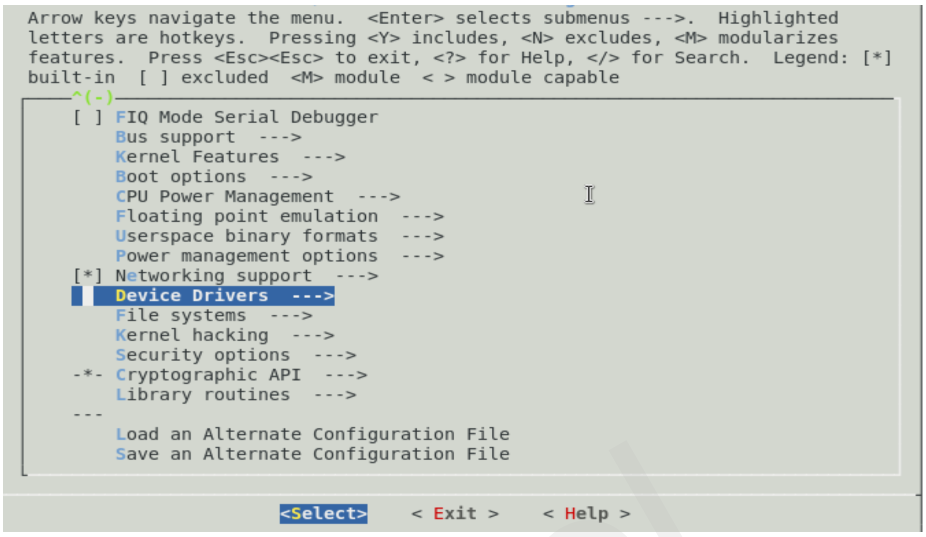
3.2.3. Modify Kernel Configuration

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:

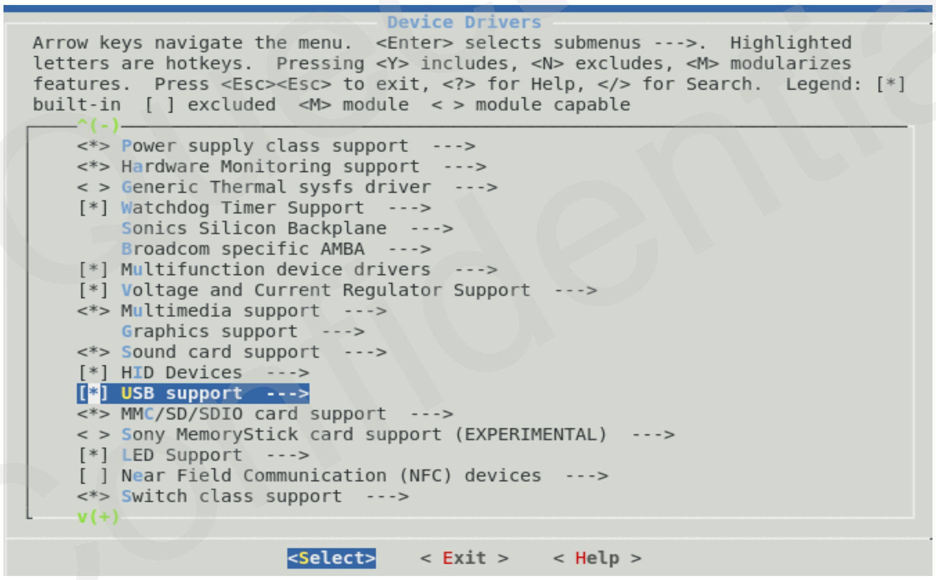
```
#make menuconfig
```

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:



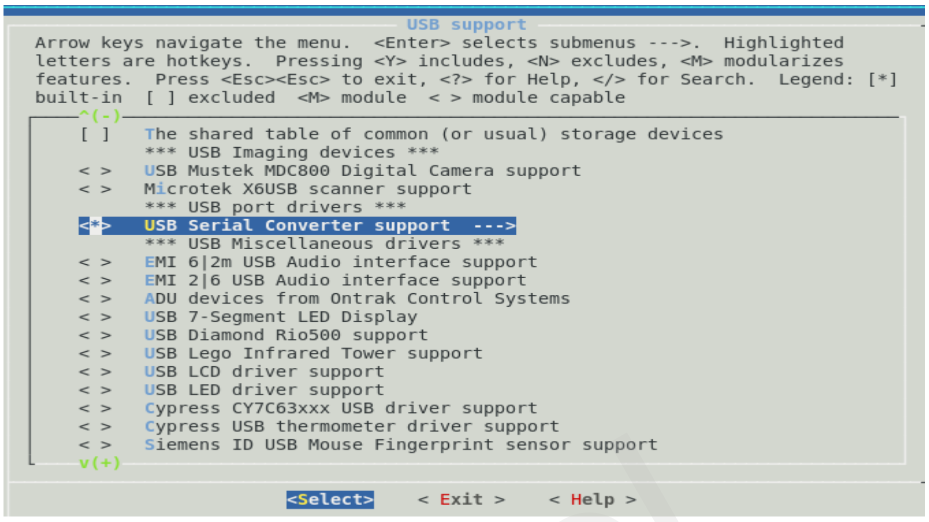
```
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
built-in  [ ] excluded  <M> module  < > module capable

    [ ] FIQ Mode Serial Debugger
        Bus support  --->
        Kernel Features  --->
        Boot options  --->
        CPU Power Management  --->
        Floating point emulation  --->
        Userspace binary formats  --->
        Power management options  --->
    [*] Networking support  --->
        Device Drivers  --->
        File systems  --->
        Kernel hacking  --->
        Security options  --->
    -*- Cryptographic API  --->
        Library routines  --->
    ---
        Load an Alternate Configuration File
        Save an Alternate Configuration File

            <Select>    < Exit >    < Help >
```

Figure 3: Kernel Configuration – Select Device Drivers



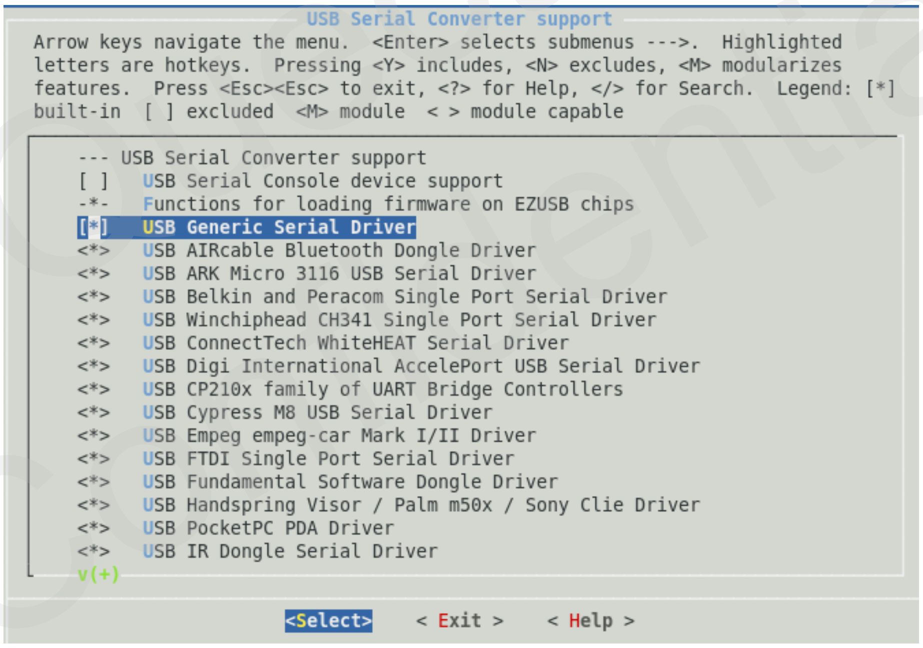
```
                          Device Drivers
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
built-in  [ ] excluded  <M> module  < > module capable

    <*> Power supply class support  --->
    <*> Hardware Monitoring support  --->
    < > Generic Thermal sysfs driver  --->
    [*] Watchdog Timer Support  --->
        Sonics Silicon Backplane  --->
        Broadcom specific AMBA  --->
    [*] Multifunction device drivers  --->
    [*] Voltage and Current Regulator Support  --->
    <*> Multimedia support  --->
        Graphics support  --->
    <*> Sound card support  --->
    [*] HID Devices  --->
    [*] USB support  --->
    <*> MMC/SD/SDIO card support  --->
    < > Sony MemoryStick card support (EXPERIMENTAL)  --->
    [*] LED Support  --->
    [ ] Near Field Communication (NFC) devices  --->
    <*> Switch class support  --->
    v(+)

            <Select>    < Exit >    < Help >
```

Figure 4: Kernel Configuration – Select USB Support



```
                              USB support
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
  built-in  [ ] excluded  <M> module  < > module capable
  ┌──────^(-)───────────────────────────────────────────────────────────────────┐
  │     [ ]    The shared table of common (or usual) storage devices            │
  │            *** USB Imaging devices ***                                      │
  │     < >    USB Mustek MDC800 Digital Camera support                         │
  │     < >    Microtek X6USB scanner support                                   │
  │            *** USB port drivers ***                                         │
  │     <*>    USB Serial Converter support  --->                               │
  │            *** USB Miscellaneous drivers ***                                │
  │     < >    EMI 6|2m USB Audio interface support                             │
  │     < >    EMI 2|6 USB Audio interface support                              │
  │     < >    ADU devices from Ontrak Control Systems                          │
  │     < >    USB 7-Segment LED Display                                        │
  │     < >    USB Diamond Rio500 support                                       │
  │     < >    USB Lego Infrared Tower support                                  │
  │     < >    USB LCD driver support                                           │
  │     < >    USB LED driver support                                           │
  │     < >    Cypress CY7C63xxx USB driver support                             │
  │     < >    Cypress USB thermometer driver support                           │
  │     < >    Siemens ID USB Mouse Fingerprint sensor support                  │
  └─────v(+)────────────────────────────────────────────────────────────────────┘
                          <Select>    < Exit >    < Help >
```

Figure 5: Kernel Configuration – Select USB Serial Converter Support



```
                     USB Serial Converter support
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
  built-in  [ ] excluded  <M> module  < > module capable
  ┌─────────────────────────────────────────────────────────────────────────────┐
  │     --- USB Serial Converter support                                        │
  │     [ ]    USB Serial Console device support                                │
  │     -*-    Functions for loading firmware on EZUSB chips                    │
  │     [*]    USB Generic Serial Driver                                        │
  │     <*>    USB AIRcable Bluetooth Dongle Driver                             │
  │     <*>    USB ARK Micro 3116 USB Serial Driver                             │
  │     <*>    USB Belkin and Peracom Single Port Serial Driver                 │
  │     <*>    USB Winchiphead CH341 Single Port Serial Driver                  │
  │     <*>    USB ConnectTech WhiteHEAT Serial Driver                          │
  │     <*>    USB Digi International AccelePort USB Serial Driver              │
  │     <*>    USB CP210x family of UART Bridge Controllers                     │
  │     <*>    USB Cypress M8 USB Serial Driver                                 │
  │     <*>    USB Empeg empeg-car Mark I/II Driver                             │
  │     <*>    USB FTDI Single Port Serial Driver                               │
  │     <*>    USB Fundamental Software Dongle Driver                           │
  │     <*>    USB Handspring Visor / Palm m50x / Sony Clie Driver              │
  │     <*>    USB PocketPC PDA Driver                                          │
  │     <*>    USB IR Dongle Serial Driver                                      │
  └─────v(+)────────────────────────────────────────────────────────────────────┘
                          <Select>    < Exit >    < Help >
```

Figure 6: Kernel Configuration – Select USB Generic Serial Driver

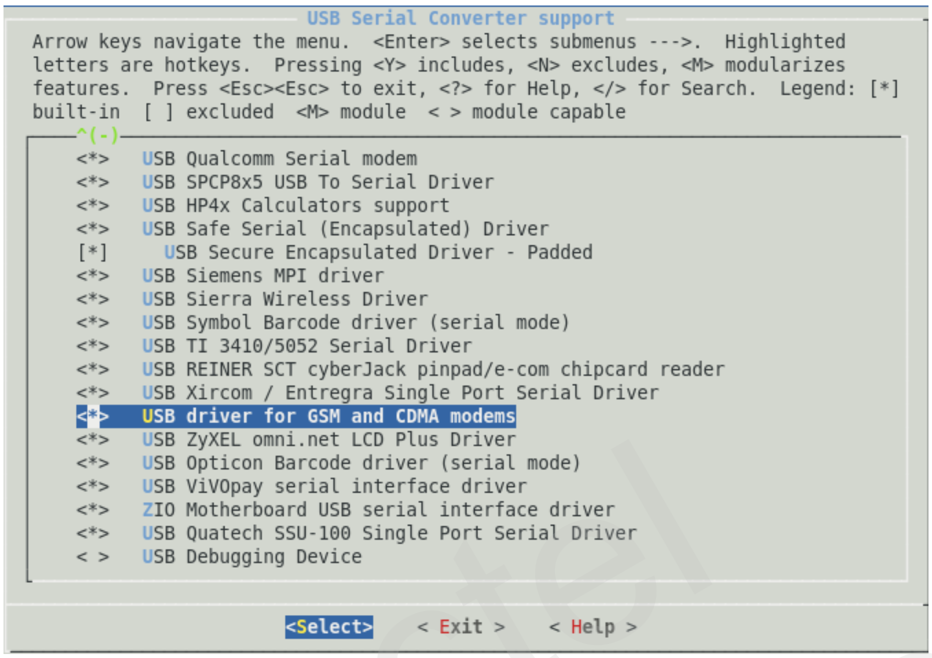

```
                    USB Serial Converter support
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
 letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
 features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
 built-in  [ ] excluded  <M> module  < > module capable
   ^(-)
     <*>   USB Qualcomm Serial modem
     <*>   USB SPCP8x5 USB To Serial Driver
     <*>   USB HP4x Calculators support
     <*>   USB Safe Serial (Encapsulated) Driver
     [*]     USB Secure Encapsulated Driver - Padded
     <*>   USB Siemens MPI driver
     <*>   USB Sierra Wireless Driver
     <*>   USB Symbol Barcode driver (serial mode)
     <*>   USB TI 3410/5052 Serial Driver
     <*>   USB REINER SCT cyberJack pinpad/e-com chipcard reader
     <*>   USB Xircom / Entregra Single Port Serial Driver
     <*>   USB driver for GSM and CDMA modems
     <*>   USB ZyXEL omni.net LCD Plus Driver
     <*>   USB Opticon Barcode driver (serial mode)
     <*>   USB ViVOpay serial interface driver
     <*>   ZIO Motherboard USB serial interface driver
     <*>   USB Quatech SSU-100 Single Port Serial Driver
     < >   USB Debugging Device

                      <Select>    < Exit >    < Help >
```

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

- /dev/ttyUSB4

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

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 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

```
chomd 777 /dev/ttyUSB2
chomd 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\r\n**
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

Abbreviation	Description
OS	Operating System
PID	Product ID
VID	Vendor ID