Contents

SCOPE

This document is a brief descriptions on:

1. how to build and use the driver on linux issued by simcom in order to use simcom modules.
2. how to modify, build and use the driver on linux issued by linux kernel in order to use simcom modules.

1 driver issued by simcom

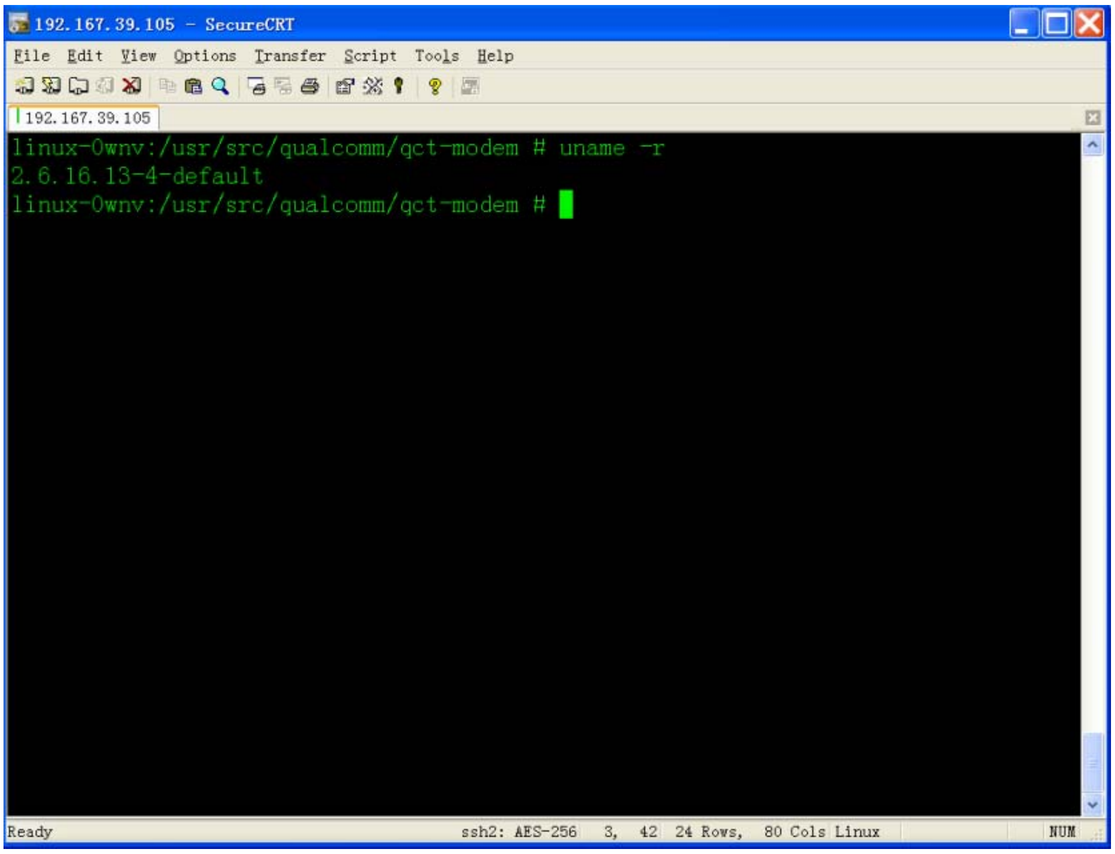
we use qct-modem driver as an example.

1.1 build the driver

1.1.1 prepare

Before building the driver you must have the kernel's source code ready which verion must be match up to the running kernel's version if you want to use the driver on current kernel. You can use the following command to query the running kernel's version:

```
uname -r
```

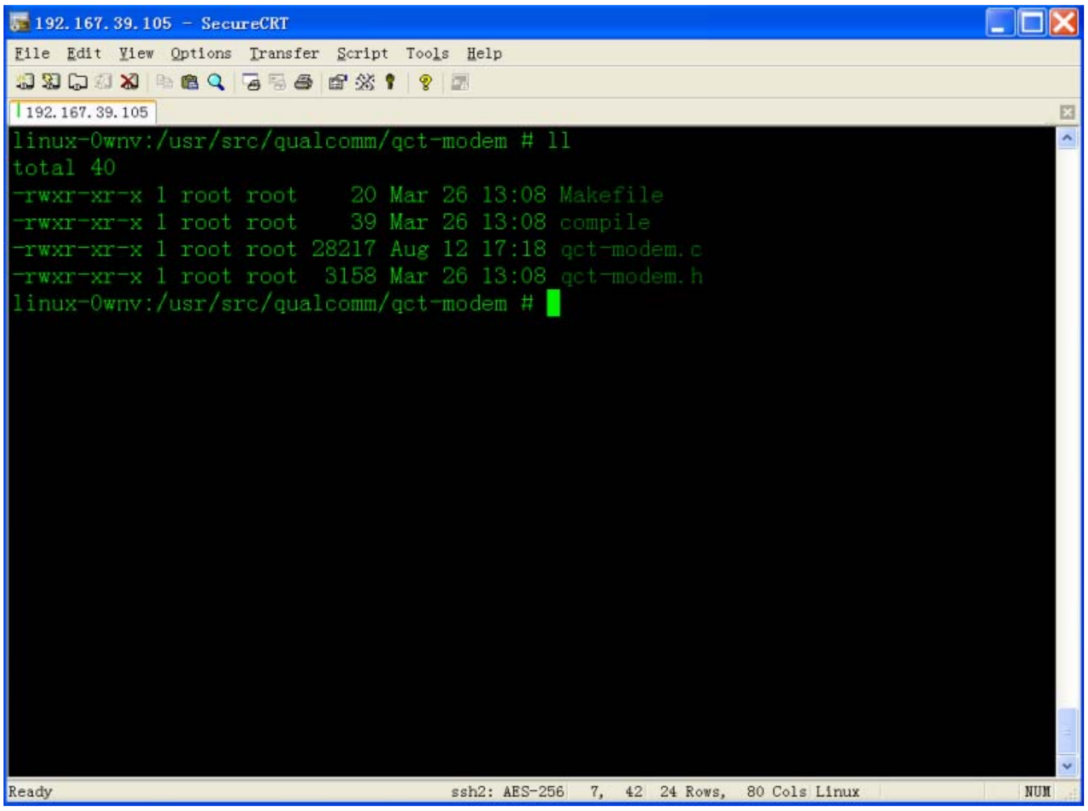

192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105
Ready
ssh2: AES-256 3, 42 24 Rows, 80 Cols Linux
NUM

```
linux-0wnv:/usr/src/qualcomm/qct-modem # uname -r
2.6.16.13-4-default
linux-0wnv:/usr/src/qualcomm/qct-modem #
```

Note: Sometimes we had to build the kernel first in order to get the corresponding files, but fortunately we only need to build the kernel very short time and press *"ctrl+c" to break the procedure, then the files we need have been created.*

1.1.2 build

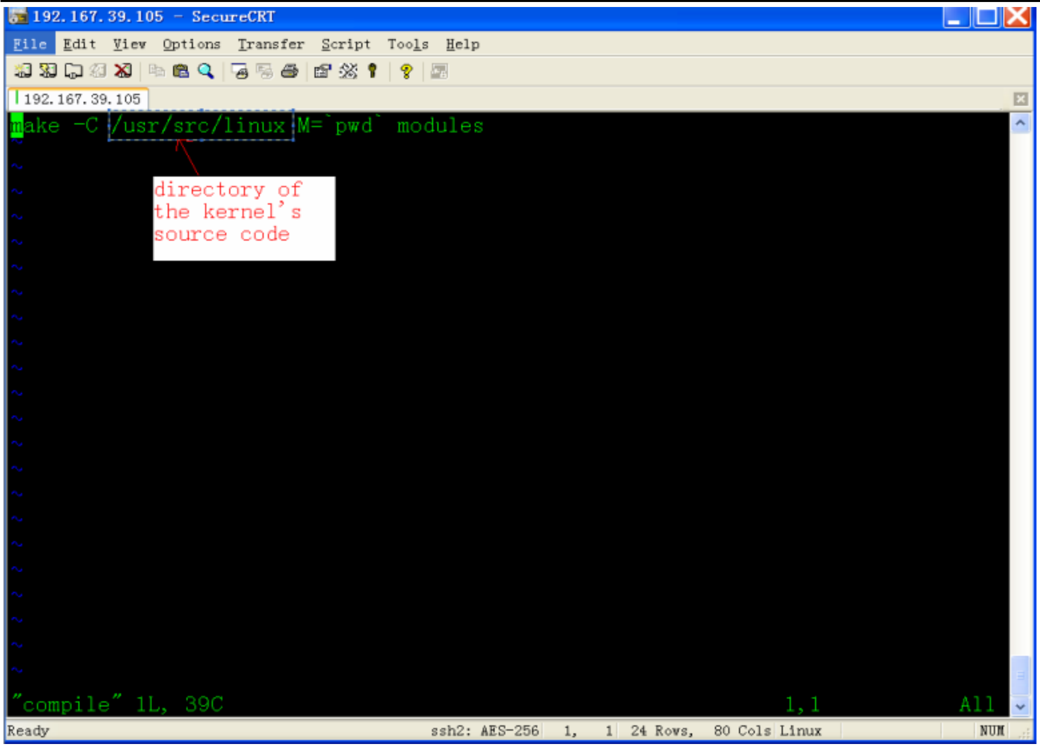
Our driver has four source code files:


192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105
Ready
ssh2: AES-256 7, 42 24 Rows, 80 Cols Linux
NUM

```
linux-0wnv:/usr/src/qualcomm/qct-modem # ll
total 40
-rwxr-xr-x 1 root root    20 Mar 26 13:08 Makefile
-rwxr-xr-x 1 root root    39 Mar 26 13:08 compile
-rwxr-xr-x 1 root root 28217 Aug 12 17:18 qct-modem.c
-rwxr-xr-x 1 root root  3158 Mar 26 13:08 qct-modem.h
linux-0wnv:/usr/src/qualcomm/qct-modem #
```

Makefile: make file used to compile the dirver
compile: this file cooperates with Makefile to compile the driver
qct-modem.c / qct-modem.h: driver source code

in order to compile the driver correctly we need to modify the compile file:


192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105

```
make -C /usr/src/linux M=`pwd` modules
```

directory of the kernel's source code

```
"compile" 1L, 39C                                    1,1           All
```

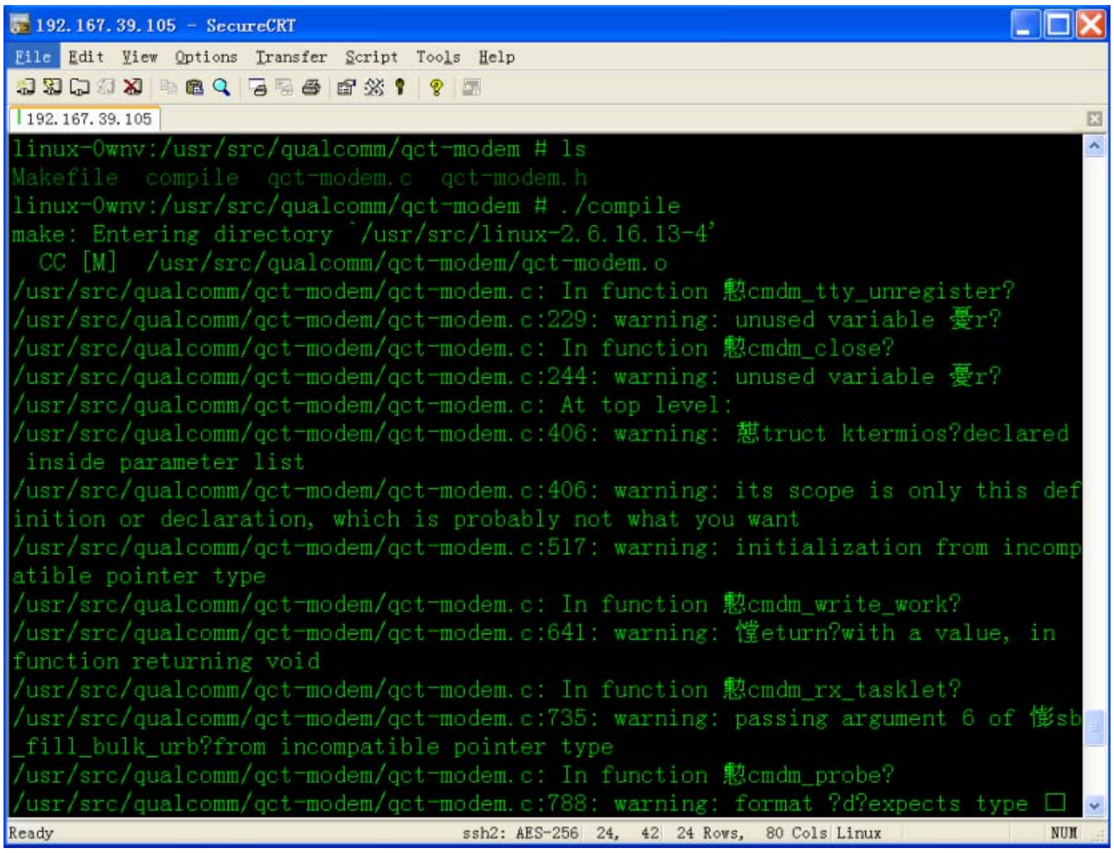
Ready ssh2: AES-256 1, 1 24 Rows, 80 Cols Linux NUM

Based on your system you need to modify the residence of the kernel's source code to the right directory in this file.

Ok, if you done all up things then you can compile the driver just use the following command:

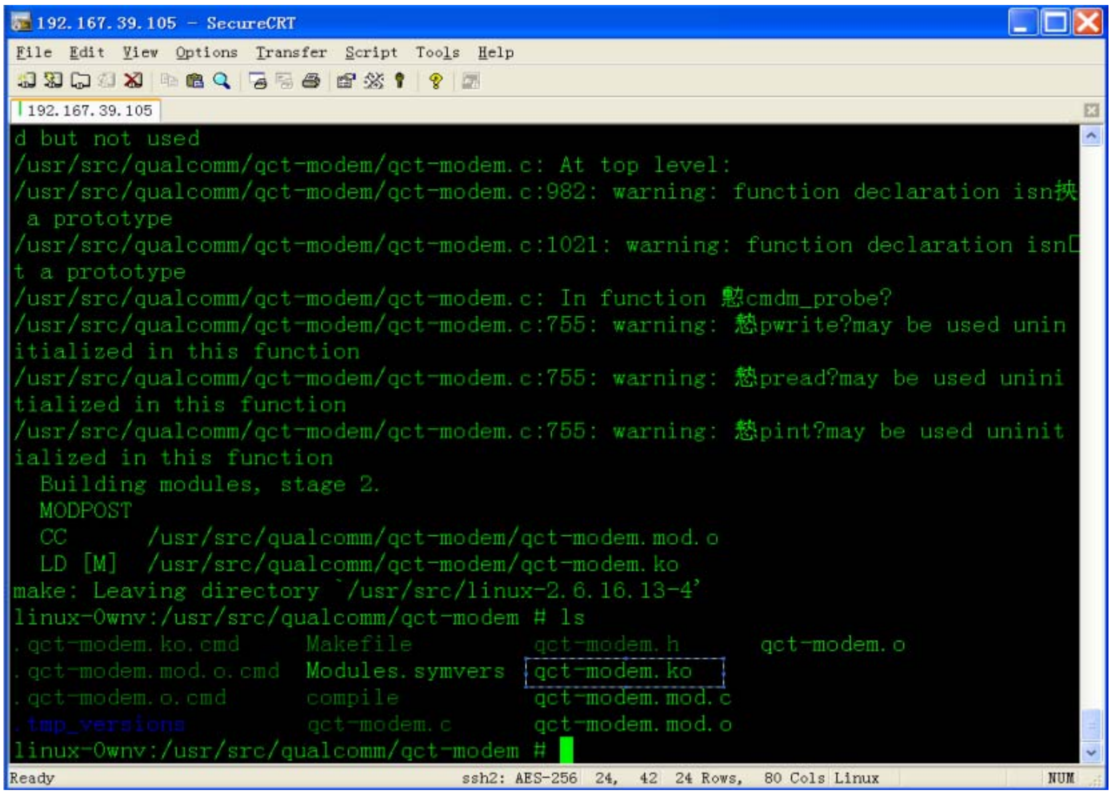
```
./compile
```

Note: you must go to driver directory first.


192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105
Ready ssh2: AES-256 24, 42 24 Rows, 80 Cols Linux NUM

```
linux-0wnv:/usr/src/qualcomm/qct-modem # ls
Makefile  compile  qct-modem.c  qct-modem.h
linux-0wnv:/usr/src/qualcomm/qct-modem # ./compile
make: Entering directory `/usr/src/linux-2.6.16.13-4'
  CC [M]  /usr/src/qualcomm/qct-modem/qct-modem.o
/usr/src/qualcomm/qct-modem/qct-modem.c: In function 慳cmdm_tty_unregister?
/usr/src/qualcomm/qct-modem/qct-modem.c:229: warning: unused variable 憂r?
/usr/src/qualcomm/qct-modem/qct-modem.c: In function 慳cmdm_close?
/usr/src/qualcomm/qct-modem/qct-modem.c:244: warning: unused variable 憂r?
/usr/src/qualcomm/qct-modem/qct-modem.c: At top level:
/usr/src/qualcomm/qct-modem/qct-modem.c:406: warning: 慡truct ktermios?declared
 inside parameter list
/usr/src/qualcomm/qct-modem/qct-modem.c:406: warning: its scope is only this def
inition or declaration, which is probably not what you want
/usr/src/qualcomm/qct-modem/qct-modem.c:517: warning: initialization from incomp
atible pointer type
/usr/src/qualcomm/qct-modem/qct-modem.c: In function 慳cmdm_write_work?
/usr/src/qualcomm/qct-modem/qct-modem.c:641: warning: 憆eturn?with a value, in
function returning void
/usr/src/qualcomm/qct-modem/qct-modem.c: In function 慳cmdm_rx_tasklet?
/usr/src/qualcomm/qct-modem/qct-modem.c:735: warning: passing argument 6 of 慤sb
_fill_bulk_urb?from incompatible pointer type
/usr/src/qualcomm/qct-modem/qct-modem.c: In function 慳cmdm_probe?
/usr/src/qualcomm/qct-modem/qct-modem.c:788: warning: format ?d?expects type □
```

if no wrong happened then the module named "qct-modem.ko" is created in current directory which is the driver module.


192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105
Ready ssh2: AES-256 24, 42 24 Rows, 80 Cols Linux NUM

```
d but not used
/usr/src/qualcomm/qct-modem/qct-modem.c: At top level:
/usr/src/qualcomm/qct-modem/qct-modem.c:982: warning: function declaration isn抰
 a prototype
/usr/src/qualcomm/qct-modem/qct-modem.c:1021: warning: function declaration isn□
t a prototype
/usr/src/qualcomm/qct-modem/qct-modem.c: In function 慳cmdm_probe?
/usr/src/qualcomm/qct-modem/qct-modem.c:755: warning: 慹pwrite?may be used unin
itialized in this function
/usr/src/qualcomm/qct-modem/qct-modem.c:755: warning: 慹pread?may be used unini
tialized in this function
/usr/src/qualcomm/qct-modem/qct-modem.c:755: warning: 慹pint?may be used uninit
ialized in this function
  Building modules, stage 2.
  MODPOST
  CC      /usr/src/qualcomm/qct-modem/qct-modem.mod.o
  LD [M]  /usr/src/qualcomm/qct-modem/qct-modem.ko
make: Leaving directory `/usr/src/linux-2.6.16.13-4'
linux-0wnv:/usr/src/qualcomm/qct-modem # ls
.qct-modem.ko.cmd      Makefile          qct-modem.h      qct-modem.o
.qct-modem.mod.o.cmd   Modules.symvers   qct-modem.ko
.qct-modem.o.cmd       compile           qct-modem.mod.c
.tmp_versions          qct-modem.c       qct-modem.mod.o
linux-0wnv:/usr/src/qualcomm/qct-modem #
```

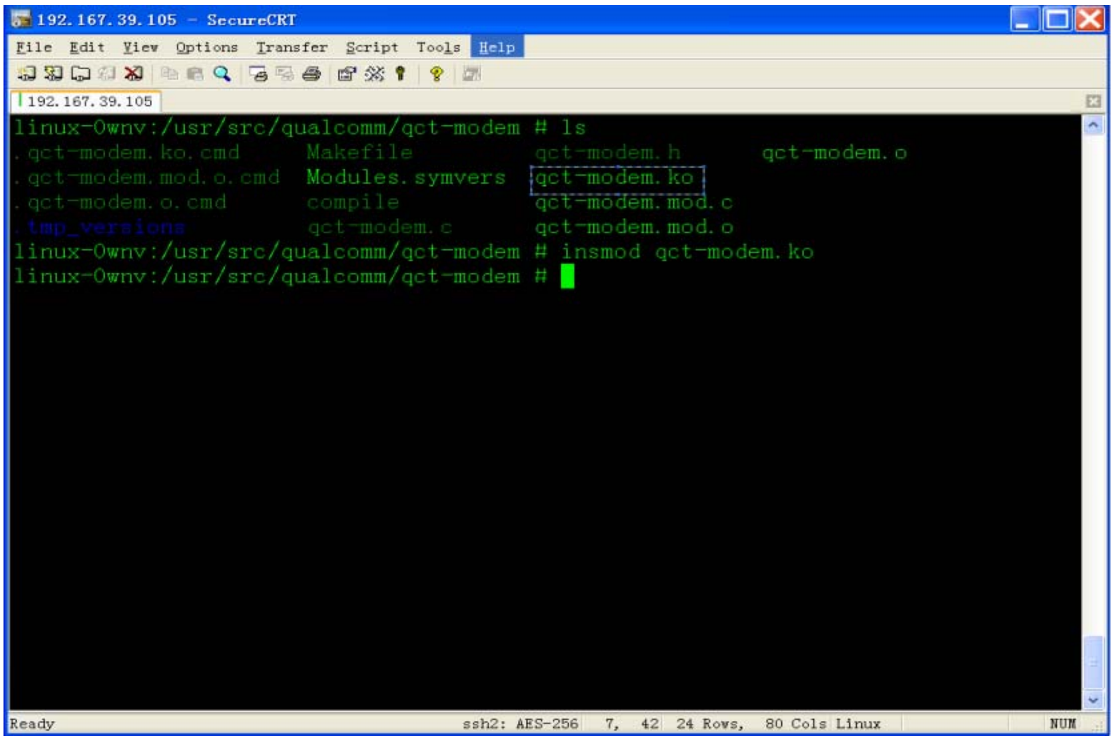
1.2 use the driver

From the first step you get the driver for the device: qct-modem.ko

1.2.1 install the driver

We can use the following command to install the driver:

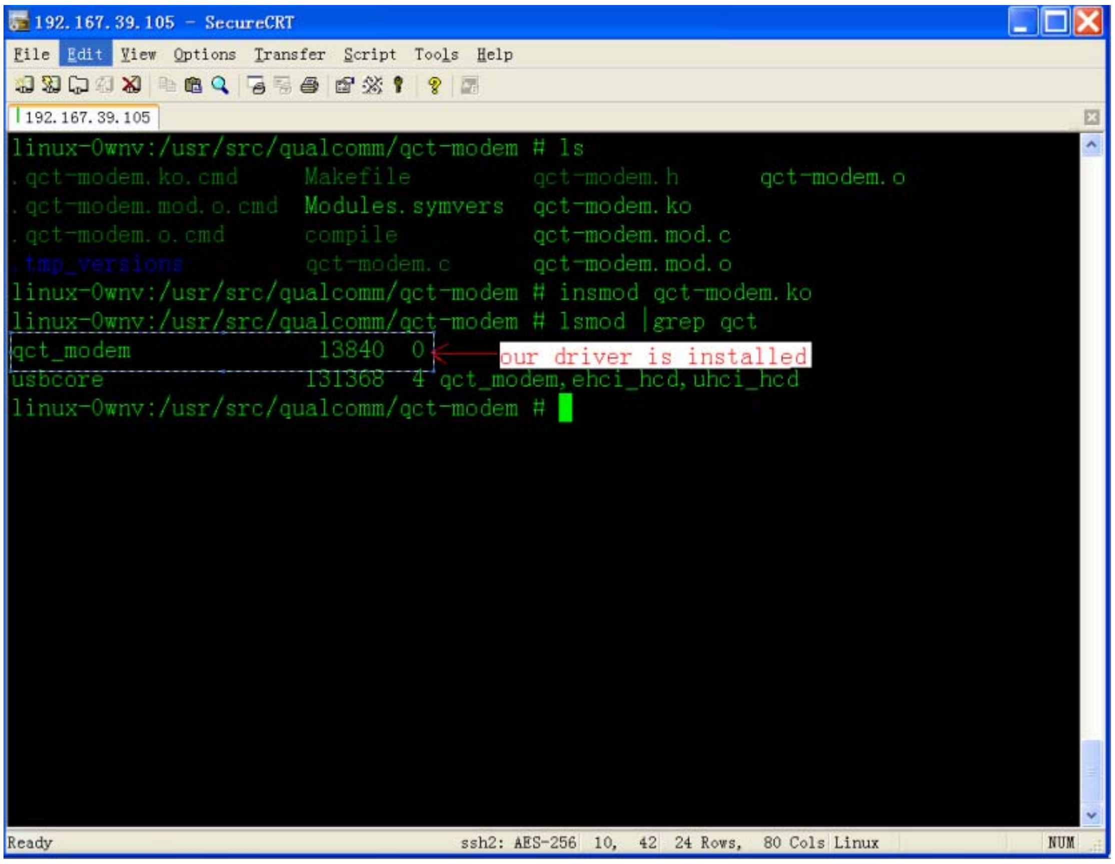
```
insmod qct-modem.ko
```


192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105
Ready
ssh2: AES-256
7, 42
24 Rows, 80 Cols Linux
NUM

```
linux-Ownv:/usr/src/qualcomm/qct-modem # ls
.qct-modem.ko.cmd      Makefile           qct-modem.h       qct-modem.o
.qct-modem.mod.o.cmd   Modules.symvers    qct-modem.ko
.qct-modem.o.cmd       compile            qct-modem.mod.c
.tmp_versions          qct-modem.c        qct-modem.mod.o
linux-Ownv:/usr/src/qualcomm/qct-modem # insmod qct-modem.ko
linux-Ownv:/usr/src/qualcomm/qct-modem #
```

if all right the driver will be installed to the system, we can use the following command to query if the driver is installed:

```
lsmod |grep qct
```



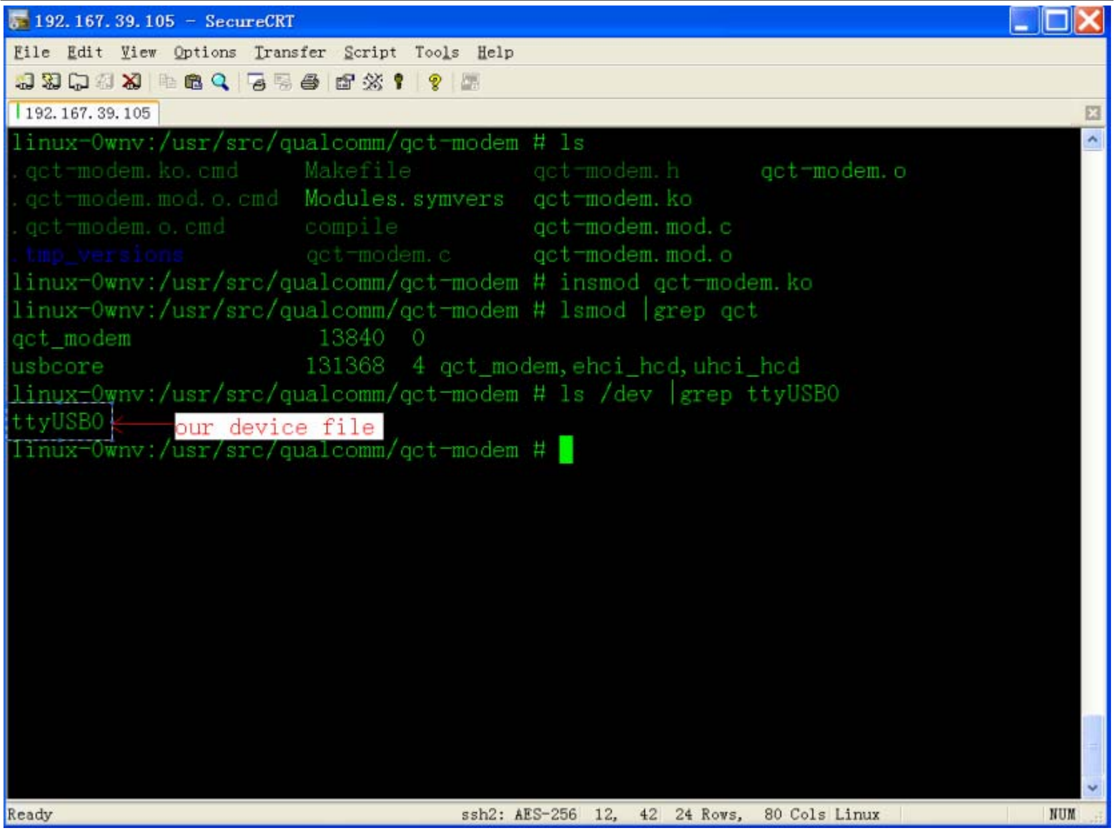
192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105
our driver is installed
Ready
ssh2: AES-256 10, 42 24 Rows, 80 Cols Linux
NUM

```
linux-0wnv:/usr/src/qualcomm/qct-modem # ls
.qct-modem.ko.cmd      Makefile          qct-modem.h       qct-modem.o
.qct-modem.mod.o.cmd   Modules.symvers   qct-modem.ko
.qct-modem.o.cmd       compile           qct-modem.mod.c
.tmp_versions          qct-modem.c       qct-modem.mod.o
linux-0wnv:/usr/src/qualcomm/qct-modem # insmod qct-modem.ko
linux-0wnv:/usr/src/qualcomm/qct-modem # lsmod |grep qct
qct_modem                13840  0
usbcore                 131368  4 qct_modem,ehci_hcd,uhci_hcd
linux-0wnv:/usr/src/qualcomm/qct-modem #
```

Note: this installation procedure is invalid when rebooting the system, so if you want to install the driver automaticly when starting the system, you'd better put the installation instruction to the startup script.

1.2.2 use the driver

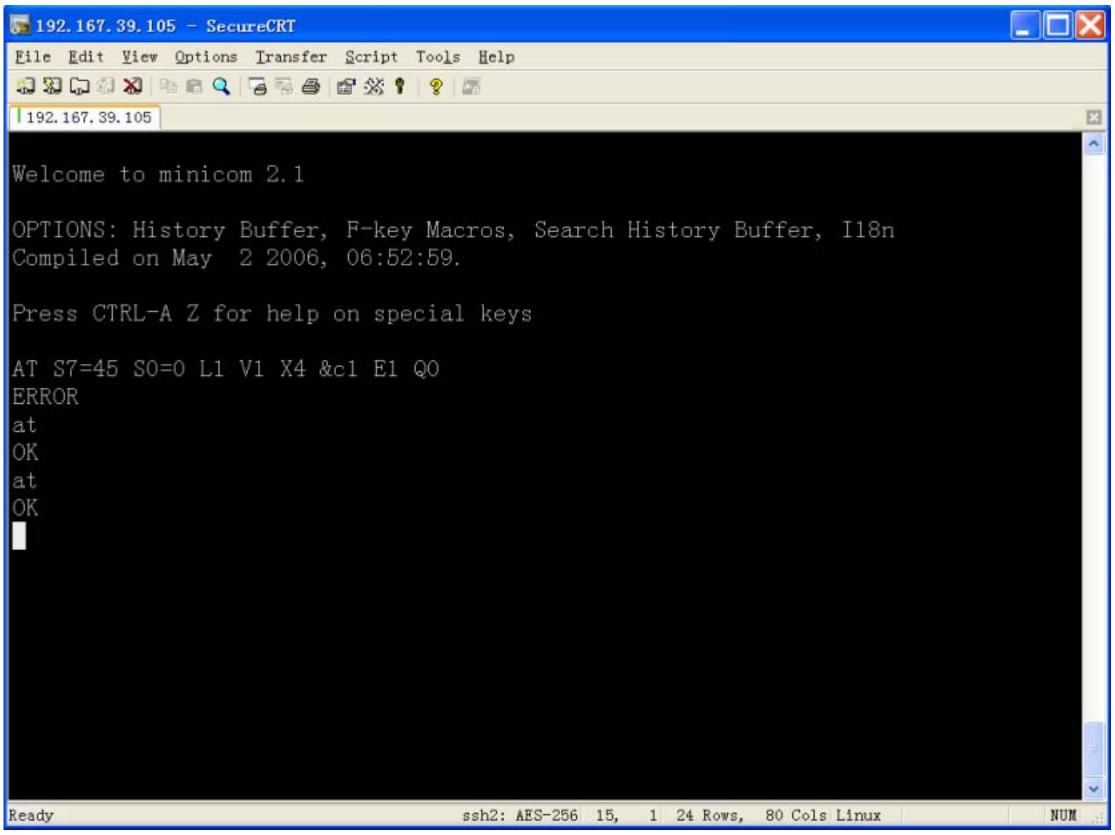
After the driver installed you can use the device via the driver, now plugging the device to the PC via USB connector, and if the device is identified by the driver there will be a device file named ttyUSB0 created in directory /dev



File Edit View Options Transfer Script Tools Help

```
linux-0wnv:/usr/src/qualcomm/qct-modem # ls
.qct-modem.ko.cmd      Makefile          qct-modem.h      qct-modem.o
.qct-modem.mod.o.cmd   Modules.symvers   qct-modem.ko
.qct-modem.o.cmd       compile           qct-modem.mod.c
.tmp_versions          qct-modem.c       qct-modem.mod.o
linux-0wnv:/usr/src/qualcomm/qct-modem # insmod qct-modem.ko
linux-0wnv:/usr/src/qualcomm/qct-modem # lsmod |grep qct
qct_modem              13840  0
usbcore               131368  4 qct_modem,ehci_hcd,uhci_hcd
linux-0wnv:/usr/src/qualcomm/qct-modem # ls /dev |grep ttyUSB0
ttyUSB0    our device file
linux-0wnv:/usr/src/qualcomm/qct-modem #
```

if we get our device file ready then we can use tools such as minicom, wvdial etc to use the device。



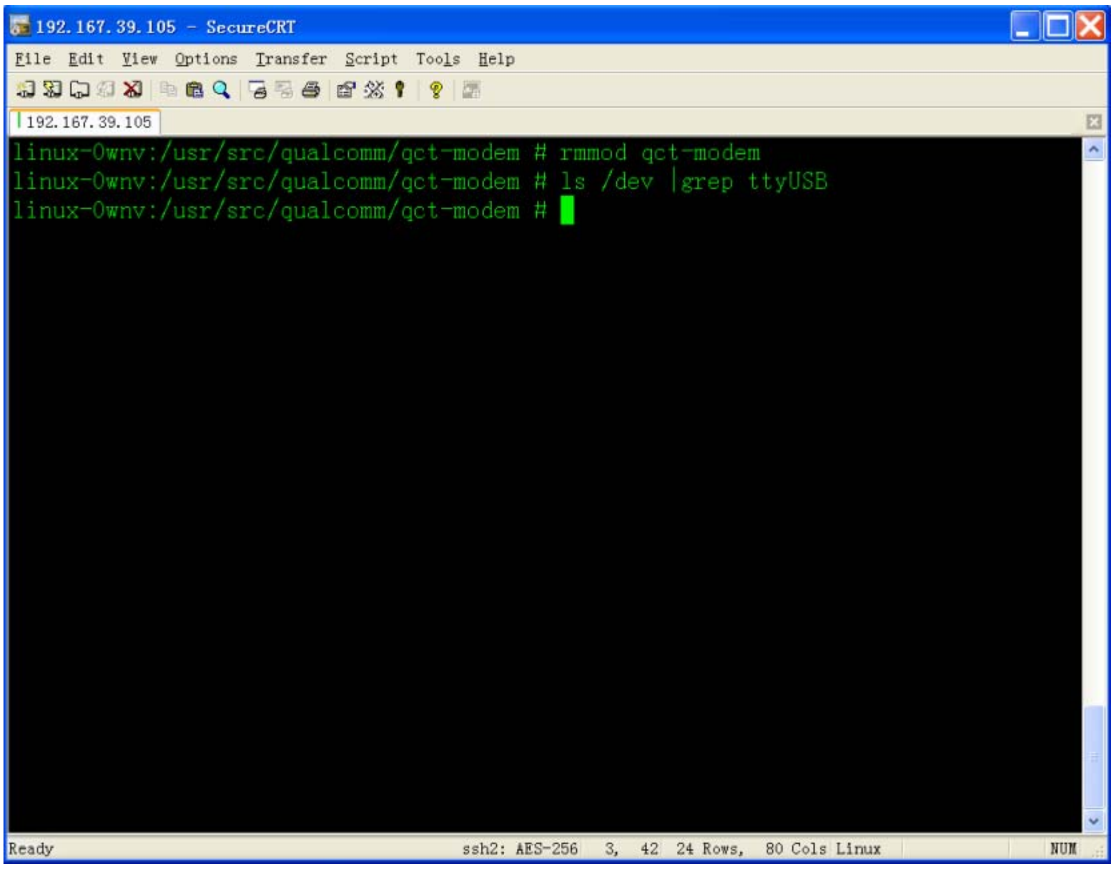
File Edit View Options Transfer Script Tools Help

```
Welcome to minicom 2.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on May  2 2006, 06:52:59.

Press CTRL-A Z for help on special keys

AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
ERROR
at
OK
at
OK
```

1.2.3 remove the driver

after we had used the device maybe we want to delete the driver from the system, so just use the following command to do such thing:

rmmod qct-modem



```
linux-0wnv:/usr/src/qualcomm/qct-modem # rmmod qct-modem
linux-0wnv:/usr/src/qualcomm/qct-modem # ls /dev |grep ttyUSB
linux-0wnv:/usr/src/qualcomm/qct-modem #
```

192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105
Ready ssh2: AES-256 3, 42 24 Rows, 80 Cols Linux NUM

after removed we can use "ls /dev |grep ttyUSB" to see if the driver is removed correctly.

Note: when removing the driver we must disconnect the device and close all the tools using the device first.

2 driver issued by linux kernel

in fact the kernel with version of 2.6.20 and later has a common driver named usbserial which can be used by our device except the NMEA port. and if we do some modification to this driver it can be used by NMEA port too.

Next we will use the source code of 2.6.22 to depict how to modify, build and use such driver.

2.1 modify the driver

As the NMEA port belonged to simcom's composite device can only send nmea data but can't receive any data, so we need to modify the driver to write nothing to this port.

 There are two files need to be modified:

drivers\usb\serial\Usb-serial.c

drivers\usb\serial\Generic.c

the flammulated code is our added.

drivers\usb\serial\Usb-serial.c:

```
int usb_serial_probe(struct usb_interface *interface,
                   const struct usb_device_id *id)
{
    ……
    int max_endpoints;
    int isnmea = 0;  /*add by aaron*/

    ……

    /* descriptor matches, let's find the endpoints needed */
    /* check out the endpoints */
    iface_desc = interface->cur_altsetting;

    /*add by aaron*/
    /*
     * we check if this interface is our nmea port, the number
     * of nmea interface in our device is 1.
     */
    if(dev-> descriptor.idVendor == 0x05c6 &&
       dev-> descriptor.idProduct == 0x9000 &&
       iface_desc-> desc.bInterfaceNumber == 1)
            isnmea = 1;
    /*end by aaron*/

    for (i = 0; i < iface_desc->desc.bNumEndpoints; ++i) {
        endpoint = &iface_desc->endpoint[i].desc;

            ……

            if (usb_endpoint_is_bulk_out(endpoint)) {
                /* we found a bulk out endpoint */
                dbg("found bulk out on endpoint %d", i);
                if(!isnmea) {   /*add by aaron*/
                                /*it mustn't send data to our nmea port*/
                    bulk_out_endpoint[num_bulk_out] = endpoint;
                    ++num_bulk_out;
                    }    /*add by aaron*/
                }

                    ……
        }

        ……

        #ifdef CONFIG_USB_SERIAL_GENERIC
            if (type == &usb_serial_generic_device) {
                num_ports = num_bulk_out;
                if (num_ports == 0 && !isnmea) {  /*modified by aaron*/
                    unlock_kernel();
                    dev_err(&interface->dev, "Generic device with no bulk
                            out, not allowed.\n");
                    kfree (serial);
                    return -EIO;
                }
            }
        #endif

        ……

}  /* usb_serial_probe()*/
```

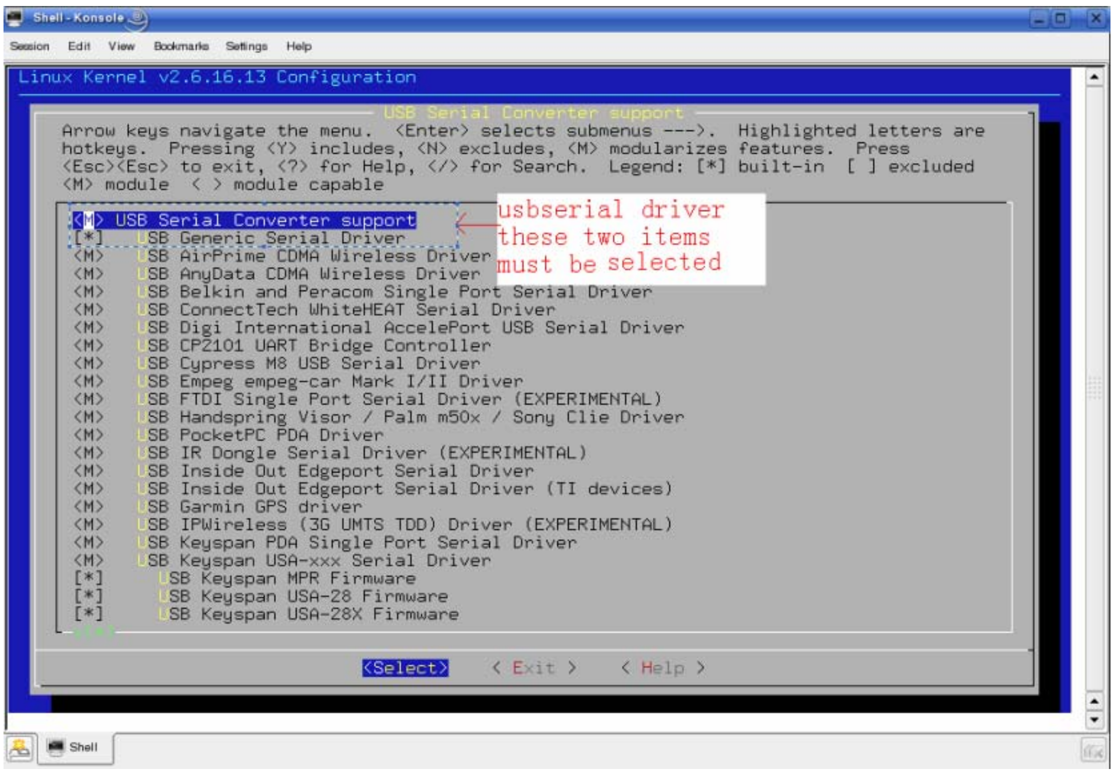
drivers\usb\serial\Generic.c

```
int usb_serial_generic_write(struct usb_serial_port *port, const unsigned char
                             *buf, int count)
{

    ……

    /* no bulk out, so return 0 bytes written */
   /*modified by aaron*/
   /*
    * as most applications will not return until all the data be sent out,
    * so if the write operation is doing on our nmea port we just return the count
    * and send nothing. Other interface using this driver will not go here as
    * usb_serial_probe will filter them if they have on bulk out endpoint.
    */
    //return 0;
    return count;
   /*end by aaron*/
}
```

2.2 Build the driver

Following is the steps on how to build the driver

1) get the kernel source code from www.kernel.org.
2) unzip the source code and modify the driver.
3) Use "make menuconfig" to let the driver to be compiled as module.



```
Linux Kernel v2.6.16.13 Configuration
                         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

  <M> USB Serial Converter support
  [*]   USB Generic Serial Driver
  <M>   USB AirPrime CDMA Wireless Driver
  <M>   USB AnyData CDMA Wireless Driver
  <M>   USB Belkin and Peracom Single Port Serial Driver
  <M>   USB ConnectTech WhiteHEAT Serial Driver
  <M>   USB Digi International AccelePort USB Serial Driver
  <M>   USB CP2101 UART Bridge Controller
  <M>   USB Cypress M8 USB Serial Driver
  <M>   USB Empeg empeg-car Mark I/II Driver
  <M>   USB FTDI Single Port Serial Driver (EXPERIMENTAL)
  <M>   USB Handspring Visor / Palm m50x / Sony Clie Driver
  <M>   USB PocketPC PDA Driver
  <M>   USB IR Dongle Serial Driver (EXPERIMENTAL)
  <M>   USB Inside Out Edgeport Serial Driver
  <M>   USB Inside Out Edgeport Serial Driver (TI devices)
  <M>   USB Garmin GPS driver
  <M>   USB IPWireless (3G UMTS TDD) Driver (EXPERIMENTAL)
  <M>   USB Keyspan PDA Single Port Serial Driver
  <M>   USB Keyspan USA-xxx Serial Driver
  [*]     USB Keyspan MPR Firmware
  [*]     USB Keyspan USA-28 Firmware
  [*]     USB Keyspan USA-28X Firmware

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

After configuration, these items will be configed:

CONFIG_USB = m

CONFIG_USB_SERIAL=m

CONFIG_USB_SERIAL_GENERIC=y

4) Use "make modules" to compile the usbserial driver to usbserial.ko.
   Note: if you haven't compiled the kernel yet, you need to use "make" to Compile the kernel first, otherwise the "make modules" command will be error.
5) Usb "make modules_install" to install the usbserial.ko to /lib/modules/…

2.3 use the drivers

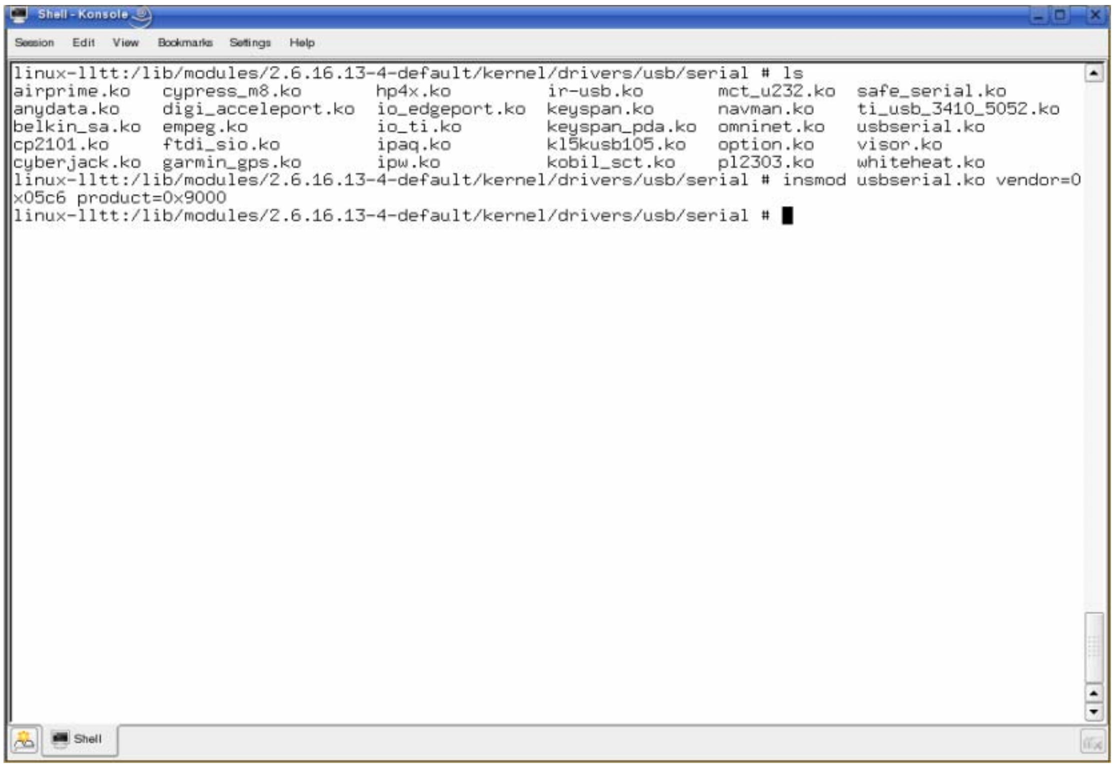
From the upper steps you get the driver usbserial.ko

2.3.1 install the driver

We can use the following command to install the driver:

    insmod usbserial.ko vendor=0x05c6 product=0x9000

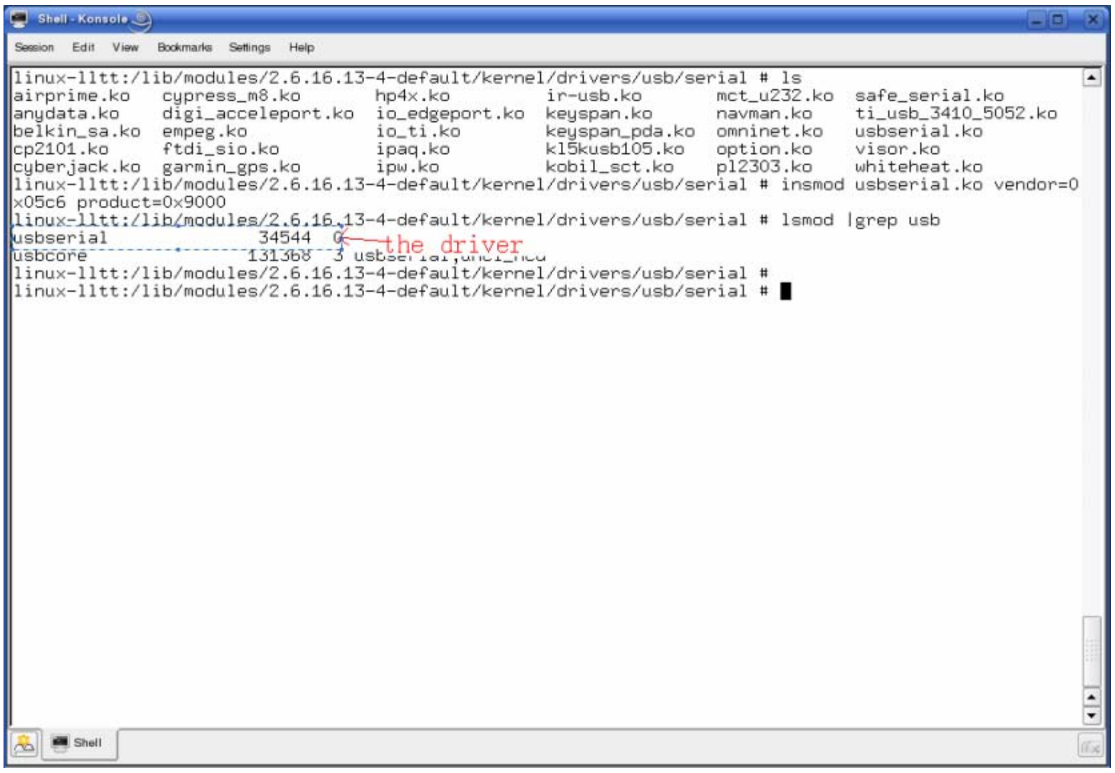
vendor and product are the parameters to the driver, 0x05c6 and 0x9000 is our device id.



```
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # ls
airprime.ko    cypress_m8.ko       hp4x.ko          ir-usb.ko        mct_u232.ko   safe_serial.ko
anydata.ko     digi_acceleport.ko  io_edgeport.ko   keyspan.ko       navman.ko     ti_usb_3410_5052.ko
belkin_sa.ko   empeg.ko            io_ti.ko         keyspan_pda.ko   omninet.ko    usbserial.ko
cp2101.ko      ftdi_sio.ko         ipaq.ko          kl5kusb105.ko    option.ko     visor.ko
cyberjack.ko   garmin_gps.ko       ipw.ko           kobil_sct.ko     pl2303.ko     whiteheat.ko
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # insmod usbserial.ko vendor=0
x05c6 product=0x9000
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # █
```

if all right the driver will be installed to the system, we can use the following command to query if the driver is installed:

lsmod |grep usb



```
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # ls
airprime.ko    cypress_m8.ko       hp4x.ko          ir-usb.ko        mct_u232.ko   safe_serial.ko
anydata.ko     digi_acceleport.ko  io_edgeport.ko   keyspan.ko       navman.ko     ti_usb_3410_5052.ko
belkin_sa.ko   empeg.ko            io_ti.ko         keyspan_pda.ko   omninet.ko    usbserial.ko
cp2101.ko      ftdi_sio.ko         ipaq.ko          kl5kusb105.ko    option.ko     visor.ko
cyberjack.ko   garmin_gps.ko       ipw.ko           kobil_sct.ko     pl2303.ko     whiteheat.ko
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # insmod usbserial.ko vendor=0
x05c6 product=0x9000
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # lsmod |grep usb
usbserial             34544  0
usbcore              131368  3 usbserial,uhci_hcd
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial #
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # █
```

Note: this installation procedure is invalid when rebooting the system, so if you *want to install the driver automaticly when starting the system, you' d better put the installation instruction to the startup script.*

2.3.2 use the driver

After the driver installed you can use the device via the driver, now plugging the device to the PC via USB connector, and if the device is identified by the driver there will be 6 device files named ttyUSB0,ttyUSB1, ttyUSB2, ttyUSB3 and ttyUSB4 which are created in directory /dev

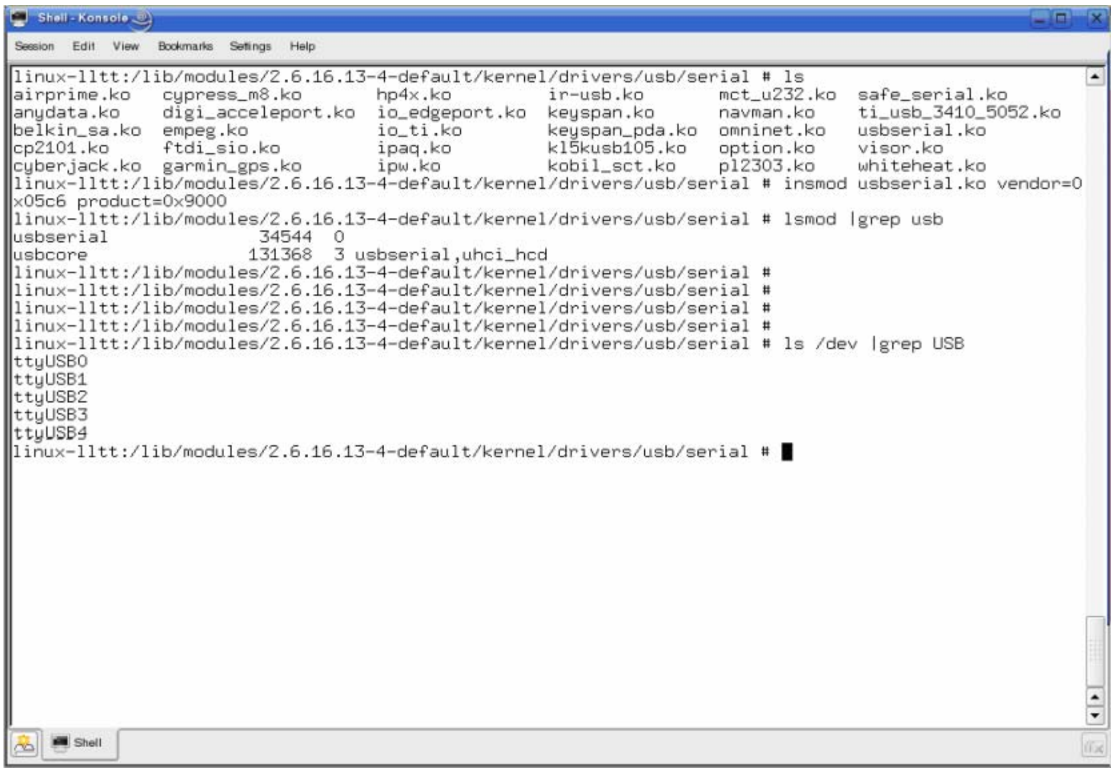
The relationship between the device files and our device interfaces is like this:

Device file	simcom' s composite device
ttyUSB0	diag interface
ttyUSB1	nmea interface
ttyUSB2	at interface
ttyUSB3	modem interface
ttyUSB4	Wireless Ethernet Adapter interface

NOTE:

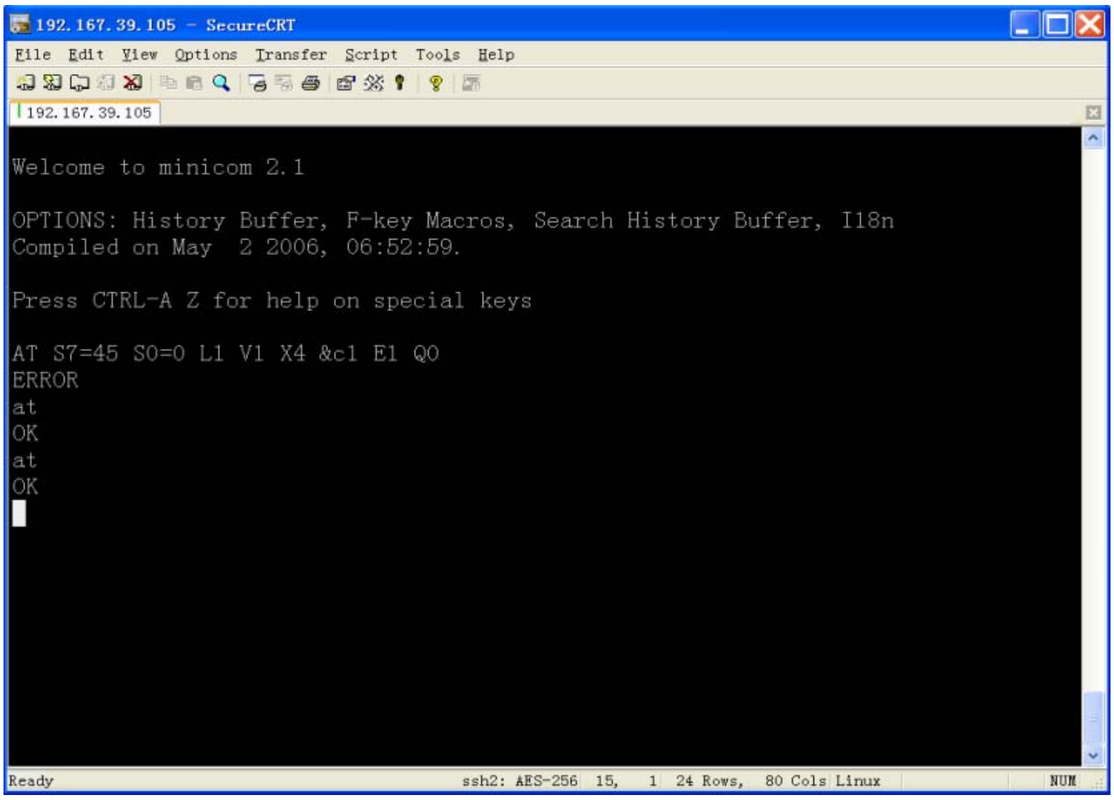
1 in some composite devices of simcom not all of the interfaces are existed, so the relationship is dynamic.

2 only the nmea, at and modem interface can be worked correctly with this driver.



```
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # ls
airprime.ko   cypress_m8.ko       hp4x.ko        ir-usb.ko       mct_u232.ko  safe_serial.ko
anydata.ko    digi_acceleport.ko  io_edgeport.ko keyspan.ko      navman.ko    ti_usb_3410_5052.ko
belkin_sa.ko  empeg.ko            io_ti.ko       keyspan_pda.ko  omninet.ko   usbserial.ko
cp2101.ko     ftdi_sio.ko         ipaq.ko        kl5kusb105.ko   option.ko    visor.ko
cyberjack.ko  garmin_gps.ko       ipw.ko         kobil_sct.ko    pl2303.ko    whiteheat.ko
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # insmod usbserial.ko vendor=0
x05c6 product=0x9000
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # lsmod |grep usb
usbserial              34544  0
usbcore               131368  3 usbserial,uhci_hcd
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial #
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial #
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial #
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial #
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # ls /dev |grep USB
ttyUSB0
ttyUSB1
ttyUSB2
ttyUSB3
ttyUSB4
linux-lltt:/lib/modules/2.6.16.13-4-default/kernel/drivers/usb/serial # █
```

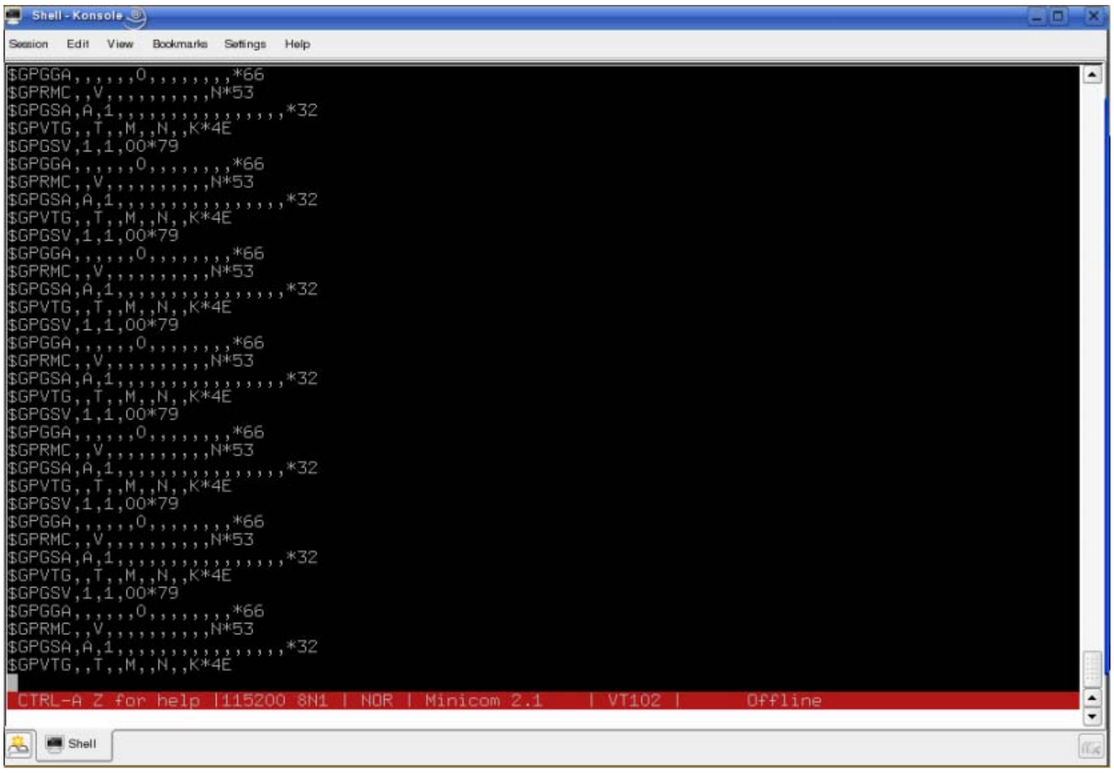
if we get our device file ready then we can use tools such as minicom, wvdial etc to use the device。



```
Welcome to minicom 2.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on May  2 2006, 06:52:59.

Press CTRL-A Z for help on special keys

AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
ERROR
at
OK
at
OK
```

192.167.39.105 - SecureCRT
File Edit View Options Transfer Script Tools Help
192.167.39.105
Ready
ssh2: AES-256 15, 1 24 Rows, 80 Cols Linux
NUM

At interface



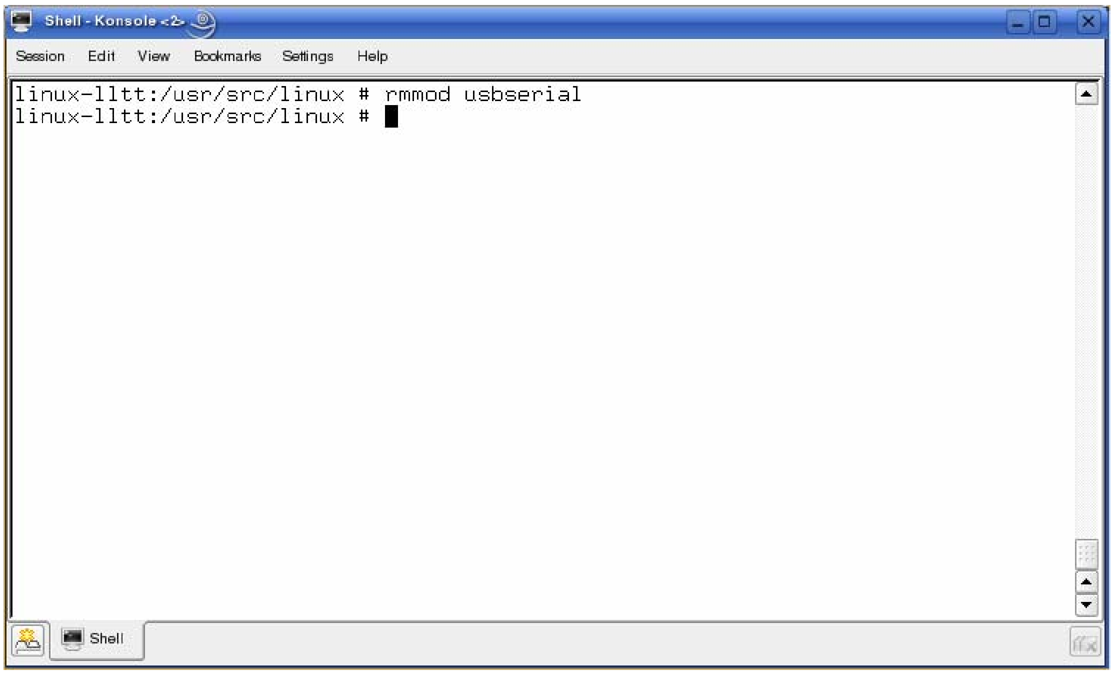
```
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSA,A,1,,,,,,,,,,,,,,,*32
$GPVTG,,T,,M,,N,,K*4E
$GPGSV,1,1,00*79
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSA,A,1,,,,,,,,,,,,,,,*32
$GPVTG,,T,,M,,N,,K*4E
$GPGSV,1,1,00*79
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSA,A,1,,,,,,,,,,,,,,,*32
$GPVTG,,T,,M,,N,,K*4E
$GPGSV,1,1,00*79
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSA,A,1,,,,,,,,,,,,,,,*32
$GPVTG,,T,,M,,N,,K*4E
$GPGSV,1,1,00*79
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSA,A,1,,,,,,,,,,,,,,,*32
$GPVTG,,T,,M,,N,,K*4E
$GPGSV,1,1,00*79
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSA,A,1,,,,,,,,,,,,,,,*32
$GPVTG,,T,,M,,N,,K*4E
$GPGSV,1,1,00*79
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSA,A,1,,,,,,,,,,,,,,,*32
$GPVTG,,T,,M,,N,,K*4E

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.1     | VT102 |        Offline
```

Nmea interface

2.3.3 remove the driver

after we had used the device maybe we want to delete the driver from the system, so just use the following command to do such thing:

rmmod usbserial



```
linux-lltt:/usr/src/linux # rmmod usbserial
linux-lltt:/usr/src/linux #
```

after removed we can use "lsmod |grep serial" to see if the driver is removed correctly.

Note: when removing the driver we must disconnect the device and close all the tools using the device first.