INSTALLATION OF QMI_WWAN AND USB SERIAL OPTION DRIVERS ON A UBUNTU FOR QUECTEL RM500Q-GL MODULE

NOTE : In this tutorial an alongside installation is proceeded. Therefore, the example outputs will be given in the form like this:

$ This example code
 # This is the example output

NOTE 2 : This document is generated by following the *"LTE&5G Linux USB Driver User Guide"* document ,referred to [1]

1-) Kernel Modification to enable correct build of USB Serial Option driver (Chapter 3.2.6 in [1])

- **SWITCH TO YOUR KERNEL DIRECTORY**
  - **$ cd /usr/src/linux-headers-$(uname -r)**
  - **# uname -r command tells us our kernel version which is 5.4.0-146-generic for my case**
- **SET ENVIRONMENT VARIABLES**
  - **$ export ARCH=arm**
  - **$ export CROSS_COMPILE=arm-none-linux-gnueabi-**

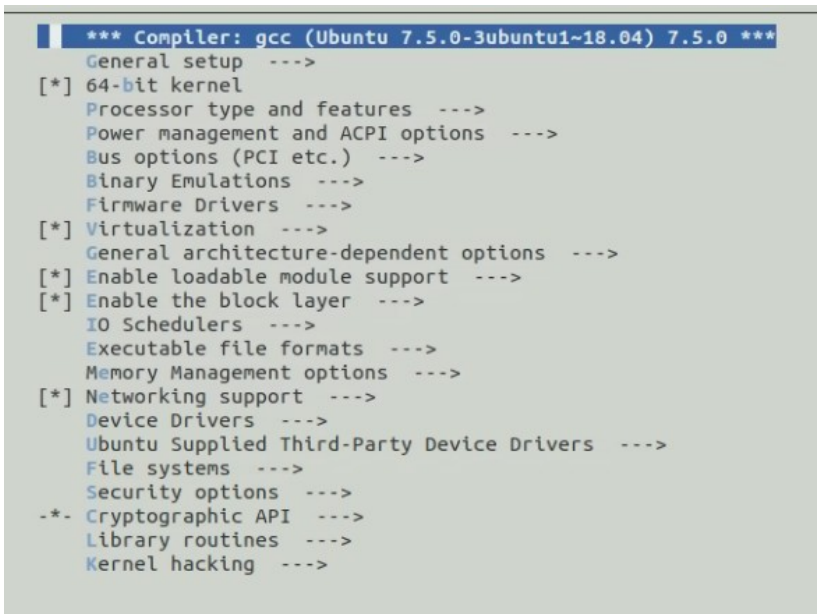
  Before going to step 3, make sure that ***"make, gcc, flex, bison"*** packages are already installed in the system. Therefore, run:

  - **$ sudo apt-get install make gcc flex bison**
  - **$ sudo make bcmrpi_defconfig**

    # The output should be like this:

    ***
    *** Can't find default configuration "arch/x86/configs/bcmrpi_defconfig"!
    ***
    scripts/kconfig/Makefile:90: recipe for target 'bcmrpi_defconfig' failed
    make[1]: *** [bcmrpi_defconfig] Error 1
    Makefile:617: recipe for target 'bcmrpi_defconfig' failed
    make: *** [bcmrpi_defconfig] Error 2

    Getting this error is OK, please proceed with the next step.
- **COMPILE THE KERNEL**
  - **$ sudo make menuconfig**
  - From the menu whose screenshot is given below, navigate the followings:



Select: ***Device Drivers***
→ ***USB support***
→ ***USB Serial Converter Support*** **(default value <M>, press Y to make it <*> , then press Enter)**
→ ***USB Driver for GSM and CDMA modems*** **(default value <M>, press Y to make it <*> , then press Enter)**

Then select <Save> on the same screen, overwrite the .config file, then select <Exit> several times to leave.

2-) Kernel Modification to enable correct build of QMI_WWAN driver (Chapter 3.4.2 in [1])

- **SWITCH TO YOUR KERNEL DIRECTORY (if you are not in this directory)**

  $ cd /usr/src/linux-headers-$(uname -r)

- **MAKE SURE YOUR ENVIRONMENTAL VARIABLES ARE ALREADY SET**

  - **$ export ARCH=arm**
  - **$ export CROSS_COMPILE=arm-none-linux-gnueabi-**

  Before going to step 3, make sure that ***"make, gcc, flex, bison"*** packages are already installed in the system. Therefore, run:

  - **$ sudo apt-get install make gcc flex bison**

  - **$ sudo make bcmrpi_defconfig**

    # The output should be like this:

    *****
    **** Can't find default configuration "arch/x86/configs/bcmrpi_defconfig"!*
    *****
    scripts/kconfig/Makefile:90: recipe for target 'bcmrpi_defconfig' failed
    *make[1]: *** [bcmrpi_defconfig] Error 1*
    Makefile:617: recipe for target 'bcmrpi_defconfig' failed
    *make: *** [bcmrpi_defconfig] Error 2*

  Getting this error is OK, please proceed with the next step.

- **COMPILE THE KERNEL**

  - **$ sudo make menuconfig**

  From the same menu given the screenshot above, navigate these configurations and change their values accordingly

  Select: ***Device Drivers***

  → **Network device *support***
  → ***USB Network Adapters*** **(default value {M}, press Y to make it {*} , then press Enter**
  → ***Multi-purpose USB Networking Framework*** **(default value {M}, press Y to make it {*})**
  → ***QMI WWAN driver for Qualcomm MSM based 3G and LTE Modems*** ***(default value <M>, press Y to make it <*>)***

  Then select <Save> on the same screen, overwrite the .config file, then select <Exit> several times to leave.

3-) BUILD OF DRIVERS (QMI_WWAN and USB_SERIAL_OPTION) FROM THE SOURCE FILES (Chapter 3.6 in [1])

BUILD OF QMI_WWAN

- **Extract *Quectel_Linux&Android_QMI_WWAN_Driver_V1.2.1.zip* into a folder**
- **Change the name of the folder *qmi_wwan* (can be any name but & symbol is not supported, it is the reason of why we change it)**
- **Then in the terminal**

  - **$ cd [YOUR_PATH]/qmi_wwan/qmi_wwan_q**

  - **$ sudo make install**

  - **# The expected output should be**

    make ARCH=x86_64 CROSS_COMPILE= -C /lib/modules/5.4.0-147-generic/build
    M=/home/lkn/Desktop/serden/quectel/qmi_wwan/qmi_wwan_q modules
    make[1]: Entering directory '/usr/src/linux-headers-5.4.0-147-generic'
    CC [M] /home/lkn/Desktop/serden/quectel/qmi_wwan/qmi_wwan_q/qmi_wwan_q.o
    Building modules, stage 2.
    MODPOST 1 modules
    CC [M] /home/lkn/Desktop/serden/quectel/qmi_wwan/qmi_wwan_q/qmi_wwan_q.mod.o
    LD [M] /home/lkn/Desktop/serden/quectel/qmi_wwan/qmi_wwan_q/qmi_wwan_q.ko
    make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-147-generic'
    cp /home/lkn/Desktop/serden/quectel/qmi_wwan/qmi_wwan_q/qmi_wwan_q.ko
    /lib/modules/5.4.0-147-generic/kernel/drivers/net/usb/
    depmod

BUILD OF USB_SERIAL_OPTION

- **Extract Quectel_Linux_USB_Serial_Option_Driver_20200720.tgz into a folder**
- **Then in the terminal, run:**
  - **$ uname -r**
  - # 5.4.0-147-generic (This my version, but the exact version is'nt important as long as it is ***NOT*** higher then v5.6 and lower than v2.6)
- **Then run:**
  - **$ cd [YOUR_PATH]/Quectel_Linux_USB_Serial_Option_Driver_20200720/**
    - In this folder there is folders of the kernel versions, select the one that matches your kernel version that you learned by the command uname -r, and if there is no exact match, choose the version that is closest by version number.
    - For example my version is 5.4.0 but since there is no v5.4.0, I selected v5.4.1
    - **$ cd v5.4.1**
    - **$ sudo make install**
- **If there is no error, the build of the usb_serial_option should be fine. The expected output will be similar to the output of the building of the qmi_wwan driver but this one will be a bit longer.**

4-) CHECK IF THE DRIVERS ARE INSTALLED CORRECTLY (Chapter 6.1 in [1])

- **Run the command to see if the drivers are listed : $ ls /sys/bus/usb/drivers**

  This is the expected output, if you're seeing the option and qmi_wwan drivers listed by this command, that means the installation procedure is succesfull.

  # cdc_wdm hub option qmi_wwan usb usbfs usbhid usbserial_generic

  If you're not seeing the option and qmi_wwan drivers listed, then try rebooting the PC, and if rebooting doesn't list the drivers too, then try to plug the QUECTEL module to an USB3 port and reboot again.

5-) INSTALLING QCONNECTMANAGER (Chapter 4.3 only step 1 in [1])

- **QconnectManager is helpful to create PDU sessions, therefore the source files are provided for installment.**
- **Extract QconnectManager_Linux_V1.6.1.zip into a folder**
  - **$ cd [YOUR_PATH]/QconnectManager_Linux_V1.6.1/quectel-CM**
  - **$ sudo make**

6-) RUNNING THE QUECTEL MODULE (MAKE SURE YOUR GNB AND CORE NETWORK ALREADY WORKS)

- **After connecting the module to the PC via USB3 port, run this command to see if the drivers are working properly and the device is seen by PC:**
  - **$ ls /dev/ttyUSB***
  - Expected output is like this: If you're seein the /ttyUSB2, that means the Quectel Module is seen by PC and works fine. (According to Chapter 4.1 in [1], usually /dev/ttyUSB2 is the interface)
  - **# /dev/ttyUSB0 /dev/ttyUSB1 /dev/ttyUSB2 /dev/ttyUSB3**
- **In my case, the default behavior of the device is to start whenever I connect it to the PC via the USB3 port. However, if it is not the case for you, use the following command to send directives for the module as in the Windows's GUI.**
  - **$ sudo busybox microcom /dev/ttyUSB2**
  - After this command, you'll see no output, but the terminal will be expecting for an input, therefore you can directly copy the following for starting the quectel module (blue LED means it is started already)
    - **AT+CFUN = 1**
    - **Probably, after this command, quectel module will try to connect the gNB but without the QconnectManager, PDU session will not be established because the required configuration is made by this script. Therefore run the quectel-CM in the next step for PDU sessions**
  - For other commands, such as changing the DNN **(AT+CGDCONT=1,"IPV4V6","dnn"**) and so please refer to the document
- **After the directives has been sent to the module via busybox microcom, run the following commands to run QconnectManager, to be succesfully start transmission for PDU establishment:**
  - **$ cd [YOUR_PATH]/QConnectManager_Linux_V1.6.1/quectel-CM**
  - **$ sudo ./quectel-CM**

- **If the gNB and CoreNetwork are already working, you'll see an output like this**

```
lkn@LKNCN:~/Desktop/serden/quectel/QConnectManager_Linux_V1.6.1/quectel-CM$ sudo ./quectel-CM
[05-23_17:14:19:865] QConnectManager_Linux_V1.6.1
[05-23_17:14:19:865] Find /sys/bus/usb/devices/1-1 idVendor=0x2c7c idProduct=0x800, bus=0x001, dev=0x010
[05-23_17:14:19:866] Auto find qmichannel = /dev/cdc-wdm1
[05-23_17:14:19:866] Auto find usbnet_adapter = wwp0s20f0u1i4
[05-23_17:14:19:866] netcard driver = qmi_wwan, driver version = 22-Aug-2005
[05-23_17:14:19:866] Modem works in QMI mode
[05-23_17:14:19:879] cdc_wdm_fd = 7
[05-23_17:14:19:969] Get clientWDS = 15
[05-23_17:14:20:001] Get clientDMS = 2
[05-23_17:14:20:033] Get clientNAS = 4
[05-23_17:14:20:065] Get clientUIM = 1
[05-23_17:14:20:097] Get clientWDA = 1
[05-23_17:14:20:129] requestBaseBandVersion RM500QGLABR11A04M4G
[05-23_17:14:20:257] requestGetSIMStatus SIMStatus: SIM_READY
[05-23_17:14:20:290] requestGetProfile[1] internet///0/IPV4V6
[05-23_17:14:20:321] requestRegistrationState2 MCC: 999, MNC: 70, PS: Attached, DataCap: 5G_SA
[05-23_17:14:20:352] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[05-23_17:14:20:352] ifconfig wwp0s20f0u1i4 0.0.0.0
[05-23_17:14:20:354] ifconfig wwp0s20f0u1i4 down
[05-23_17:14:20:448] requestSetupDataCall WdsConnectionIPv4Handle: 0x794bba50
[05-23_17:14:20:577] change mtu 1500 -> 1400
[05-23_17:14:20:577] ifconfig wwp0s20f0u1i4 up
[05-23_17:14:20:579] No default.script found, it should be in '/usr/share/udhcpc/' or '/etc//udhcpc' depend
on your udhcpc version!
[05-23_17:14:20:579] busybox udhcpc -f -n -q -t 5 -i wwp0s20f0u1i4
udhcpc: started, v1.27.2
udhcpc: sending discover
udhcpc: sending select for 10.60.0.5
udhcpc: lease of 10.60.0.5 obtained, lease time 7200
[05-23_17:14:20:767] ip -4 address flush dev wwp0s20f0u1i4
[05-23_17:14:20:770] ip -4 address add 10.60.0.5/30 dev wwp0s20f0u1i4
[05-23_17:14:20:772] ip -4 route add default via 10.60.0.6 dev wwp0s20f0u1i4
```

- **It can be seen from the log that, an interface named wwp0s20f0u1i4 is established, you should also be able to see that a new user is connected to the network from the core network logs**

- **You can also try to ping somewhere to see if the connection is working properly, for example:**

```
wwp0s20f0u1i4: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1400
        inet 10.60.0.5  netmask 255.255.255.252  destination 10.60.0.5
        inet6 fe80::2e14:ebe7:e3d4:9eac  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 10  bytes 1528 (1.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 30  bytes 2244 (2.2 KB)
        TX errors 2  dropped 0 overruns 0  carrier 0  collisions 0

lkn@LKNCN:~/Desktop/serden/quectel/QConnectManager_Linux_V1.6.1/quectel-CM$ ping -I wwp0s20f0u1i4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 10.60.0.5 wwp0s20f0u1i4: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=17.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=18.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=18.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=52 time=19.1 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=52 time=17.8 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=52 time=18.1 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=52 time=17.1 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=52 time=17.7 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=52 time=18.0 ms
^C
--- 8.8.8.8 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8012ms
rtt min/avg/max/mdev = 17.138/18.099/19.122/0.540 ms
lkn@LKNCN:~/Desktop/serden/quectel/QConnectManager_Linux_V1.6.1/quectel-CM$
```

- **If everything worked fine, then you should see the PDU session is established via capturing the packets in wireshark:**