Build a Smarter World

Quick start guide for Quectel cellular modules on Linux

North America version

Last updated May 13, 2024

Contents

Introduction

This document covers basic driver setup and data call bring up, mainly for both Raspberry Pi and Ubuntu for PC.
Linux is a "free" software, and according to https://www.gnu.org/philosophy/free-sw.en.html, the definition is:

"Free software" means software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and improve the software. Thus, "free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer." We sometimes call it "libre software," borrowing the French or Spanish word for "free" as in freedom, to show we do not mean the software is gratis.

But the side effect of such freedom is that everybody is making variations to the software. Unfortunately, there is no hard standard in Linux. Every version is different. Please take this document as a general guide.

If you have any questions, please get in touch with me by email: kasper.cheng@quectel.com

Prerequisites

I am using this Raspberry OS image as an example. And we are trying to build the drivers on the target board itself.

2020-02-13-raspbian-buster.img

After the system is up and running, we can install some software.

sudo apt install build-essential resolvconf minicom net-tools nano

For Raspberry Pi, you can install kernel headers with this command.

sudo apt install raspberrypi-kernel-headers

For Ubuntu, use this command

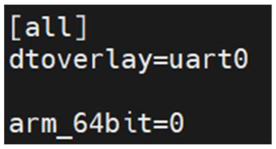
sudo apt install linux-headers-$(uname -r)

32bit and 64bit OS

As of May 2023, Raspberry Pi is moving to 64bit kernel even if the OS image is 32bit. i.e., a mixed environment with 64bit kernel + 32bit user space. Kernel driver building is an on-going issue (https://github.com/raspberrypi/linux/issues/5408)

Please consider moving to 64bit OS, or stay with 32bit kernel by adding arm_64bit=0 to /boot/config.txt

sudo nano /boot/config.txt

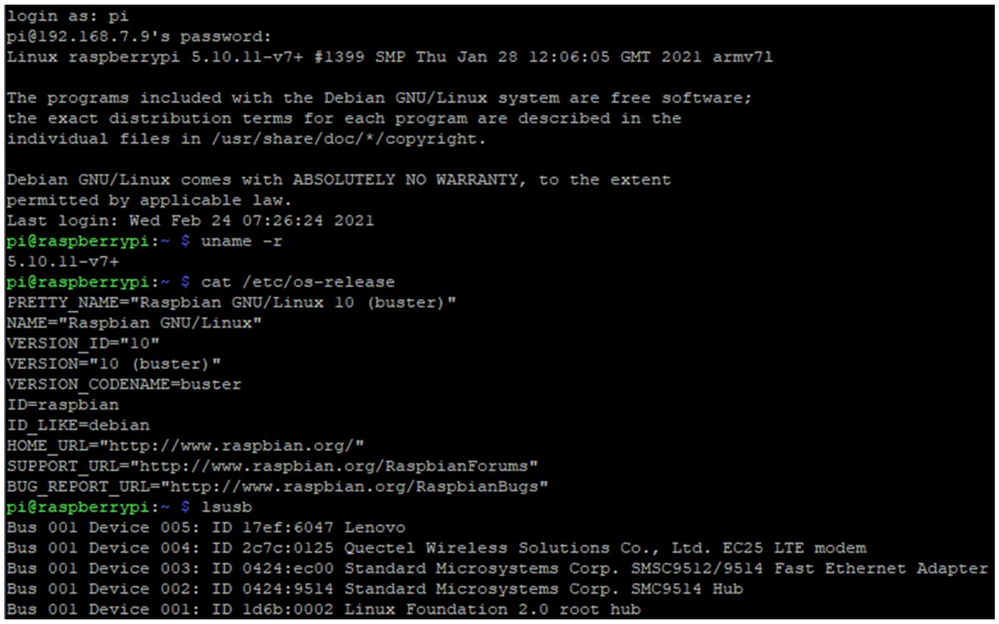


```
[all]
dtoverlay=uart0

arm_64bit=0
```

Checking system information

we can use the ***uname -r*** command to check the kernel version,

and use the ***cat /etc/os-release*** to check detailed OS version.

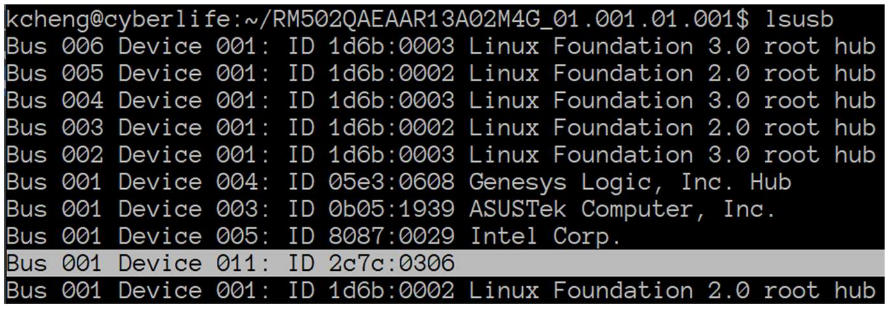


```
login as: pi
pi@192.168.7.9's password:
Linux raspberrypi 5.10.11-v7+ #1399 SMP Thu Jan 28 12:06:05 GMT 2021 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Feb 24 07:26:24 2021
pi@raspberrypi:~ $ uname -r
5.10.11-v7+
pi@raspberrypi:~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 10 (buster)"
NAME="Raspbian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
pi@raspberrypi:~ $ lsusb
Bus 001 Device 005: ID 17ef:6047 Lenovo
Bus 001 Device 004: ID 2c7c:0125 Quectel Wireless Solutions Co., Ltd. EC25 LTE modem
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp. SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Device enumeration(detection)

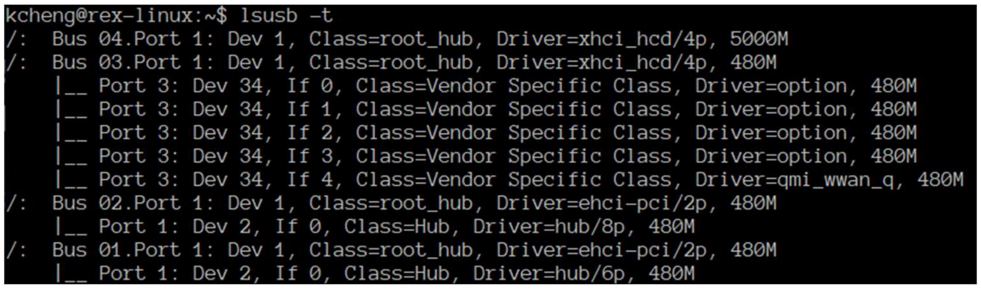
The system should see (enumerate) the device even if no driver software is installed.

lsusb can be used to list all USB devices detected by the OS. There may not be a name, but 2c7c:0306 is the USB Vendor ID and Product ID. Most of our USB devices are using the VID 2c7c.



```
kcheng@cyberlife:~/RM502QAEAAR13A02M4G_01.001.01.001$ lsusb
Bus 006 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 05e3:0608 Genesys Logic, Inc. Hub
Bus 001 Device 003: ID 0b05:1939 ASUSTek Computer, Inc.
Bus 001 Device 005: ID 8087:0029 Intel Corp.
Bus 001 Device 011: ID 2c7c:0306
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

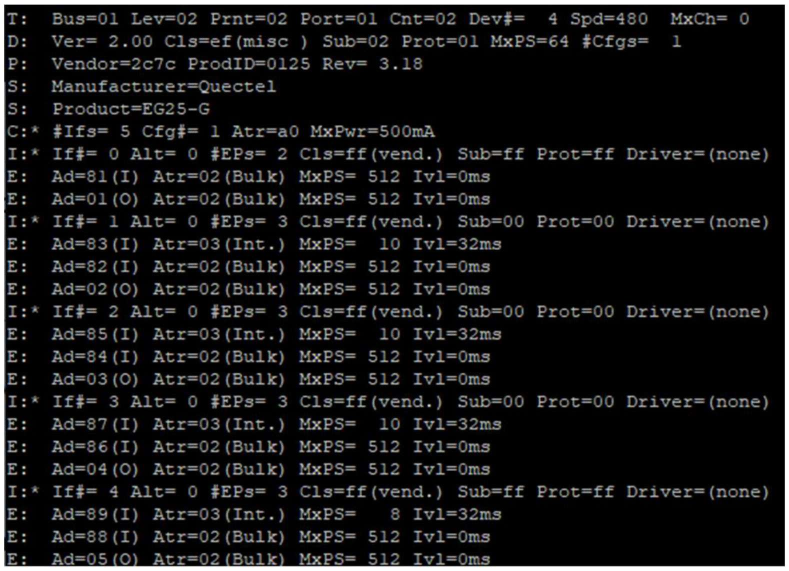
A usb device may provide multiple "interfaces". In Quectel module's case, we provide multiple serial ports (usually Debug port, GNSS port, AT command port, and modem port.), and a USB network interface for high-speed modules. **lsusb -t** gives more information about USB devices, but vendor ID and product ID are not listed here. Note that some embedded systems do not support the -t parameter.



```
kcheng@rex-linux:~$ lsusb -t
/:  Bus 04.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/4p, 5000M
/:  Bus 03.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/4p, 480M
    |__ Port 3: Dev 34, If 0, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 3: Dev 34, If 1, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 3: Dev 34, If 2, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 3: Dev 34, If 3, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 3: Dev 34, If 4, Class=Vendor Specific Class, Driver=qmi_wwan_q, 480M
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/2p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/8p, 480M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/2p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/6p, 480M
```

You also use this command to list detected USB devices in more details if sysfs is enabled in kernel.

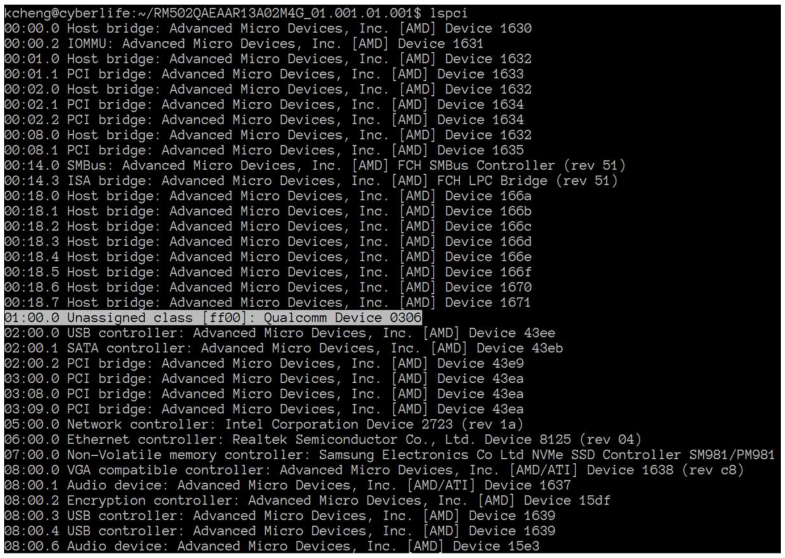
sudo cat /sys/kernel/debug/usb/devices



```
T:  Bus=01 Lev=02 Prnt=02 Port=01 Cnt=02 Dev#=  4 Spd=480  MxCh= 0
D:  Ver= 2.00 Cls=ef(misc ) Sub=02 Prot=01 MxPS=64 #Cfgs=  1
P:  Vendor=2c7c ProdID=0125 Rev= 3.18
S:  Manufacturer=Quectel
S:  Product=EG25-G
C:* #Ifs= 5 Cfg#= 1 Atr=a0 MxPwr=500mA
I:* If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=(none)
E:  Ad=81(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=01(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 1 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=(none)
E:  Ad=83(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=82(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=02(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 2 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=(none)
E:  Ad=85(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=84(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=03(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 3 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=(none)
E:  Ad=87(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=86(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=04(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 4 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=ff Driver=(none)
E:  Ad=89(I) Atr=03(Int.) MxPS=   8 Ivl=32ms
E:  Ad=88(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=05(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
```

Similarly, we can list the detected PCIE devices by this command. The highlighted line is the RM502Q-AE module.

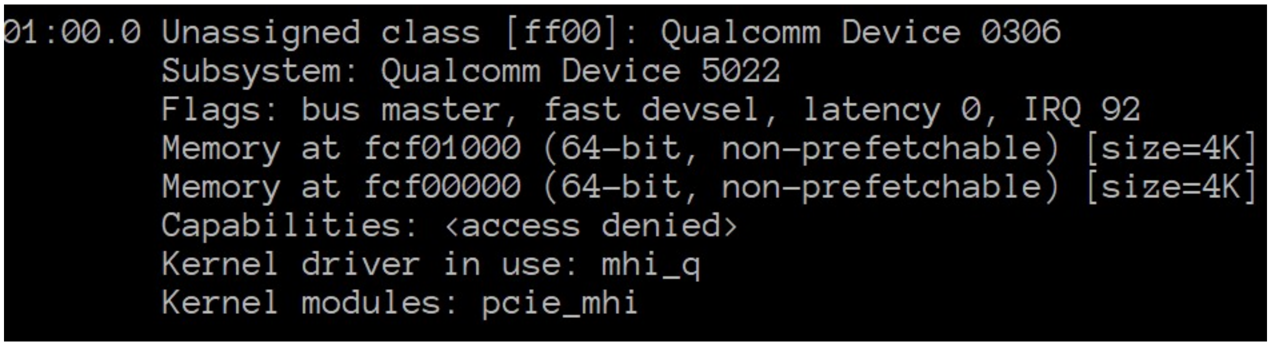
lspci



```
kcheng@cyberlife:~/RM502QAEAAR13A02M4G_01.001.01.001$ lspci
00:00.0 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 1630
00:00.2 IOMMU: Advanced Micro Devices, Inc. [AMD] Device 1631
00:01.0 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 1632
00:01.1 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 1633
00:02.0 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 1632
00:02.1 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 1634
00:02.2 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 1634
00:08.0 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 1632
00:08.1 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 1635
00:14.0 SMBus: Advanced Micro Devices, Inc. [AMD] FCH SMBus Controller (rev 51)
00:14.3 ISA bridge: Advanced Micro Devices, Inc. [AMD] FCH LPC Bridge (rev 51)
00:18.0 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166a
00:18.1 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166b
00:18.2 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166c
00:18.3 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166d
00:18.4 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166e
00:18.5 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166f
00:18.6 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 1670
00:18.7 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 1671
01:00.0 Unassigned class [ff00]: Qualcomm Device 0306
02:00.0 USB controller: Advanced Micro Devices, Inc. [AMD] Device 43ee
02:00.1 SATA controller: Advanced Micro Devices, Inc. [AMD] Device 43eb
02:00.2 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 43e9
03:00.0 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 43ea
03:08.0 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 43ea
03:09.0 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 43ea
05:00.0 Network controller: Intel Corporation Device 2723 (rev 1a)
06:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. Device 8125 (rev 04)
07:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd NVMe SSD Controller SM981/PM981
08:00.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] Device 1638 (rev c8)
08:00.1 Audio device: Advanced Micro Devices, Inc. [AMD/ATI] Device 1637
08:00.2 Encryption controller: Advanced Micro Devices, Inc. [AMD] Device 15df
08:00.3 USB controller: Advanced Micro Devices, Inc. [AMD] Device 1639
08:00.4 USB controller: Advanced Micro Devices, Inc. [AMD] Device 1639
08:00.6 Audio device: Advanced Micro Devices, Inc. [AMD] Device 15e3
```

To read more details, add the -v parameter. The driver has been loaded in this case, i.e. pcie_mhi

lspci -v



```
01:00.0 Unassigned class [ff00]: Qualcomm Device 0306
        Subsystem: Qualcomm Device 5022
        Flags: bus master, fast devsel, latency 0, IRQ 92
        Memory at fcf01000 (64-bit, non-prefetchable) [size=4K]
        Memory at fcf00000 (64-bit, non-prefetchable) [size=4K]
        Capabilities: <access denied>
        Kernel driver in use: mhi_q
        Kernel modules: pcie_mhi
```

Download links

The Quectel drivers and tools for Linux mentioned in this document can be found in this link.
https://cnquectel-my.sharepoint.com/:f:/g/personal/america-fae_quectel_com/Eojcy41tWUFOnUwLTDRIREgB_sxQXcjY6CJIjJ9mTrZ35g?e=y89kBf

Serial interface driver for USB modules
Quectel_Linux_USB_Serial_Option_Driver_20XXXXXX.tgz

QMI/RMNET network adapter driver for USB modules
Quectel_Linux_Android_QMI_WWAN_Driver_VX.X.X.zip

GobiNet network adapter driver for USB modules, depreciated, replaced by QMI_WWAN and QMI_WWAN_Q
Quectel_Linux_Android_GobiNet_Driver_VX.X.X.zip

Data call manager for both PCIE and USB modules. Supports both QMI and MBIM modes
Quectel_QConnectManager_Linux_VX.X.X.X.zip

Driver for PCIE modules
Quectel_Linux_PCIE_MHI_Driver_VX.X.X.zip

Module debug log collection tool
QLog_Linux_Android_VX.X.XX.zip

Module firmware update tool
QFirehose_Linux_Android_VX.X.XX.zip

example PPP call script
linux-ppp-scripts_V.X.zip

Example commands lines to decompress the files.

unzip Quectel_Linux_Android_QMI_WWAN_Driver_V1.2.1.zip

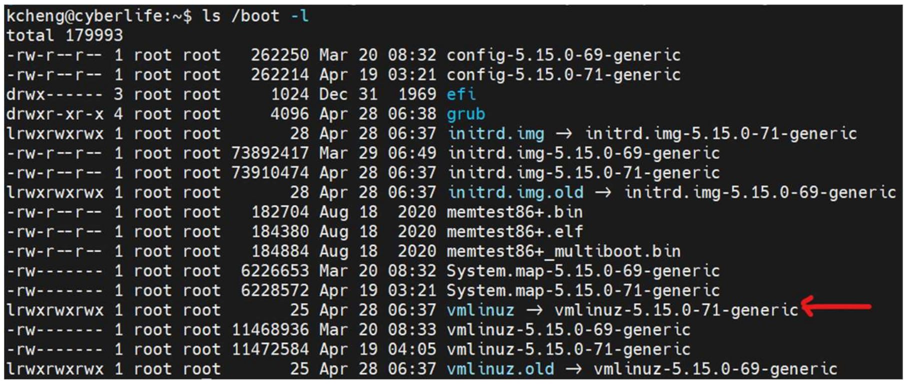
tar -xvf Quectel_Linux_USB_Serial_Option_Driver_20211204.tgz

note that quectel-CM zip package has no top-level folder, let’s make a folder for it before decompressing
mkdir quectel-CM-1.6.1
cd quectel-CM-1.6.1
unzip ../Quectel_QConnectManager_Linux_V1.6.1_20210720.zip

Please make sure there are no special characters like & in the path or in the file name. Or else the Linux build command will be confused and give you funny errors.

About Linux kernel and modules

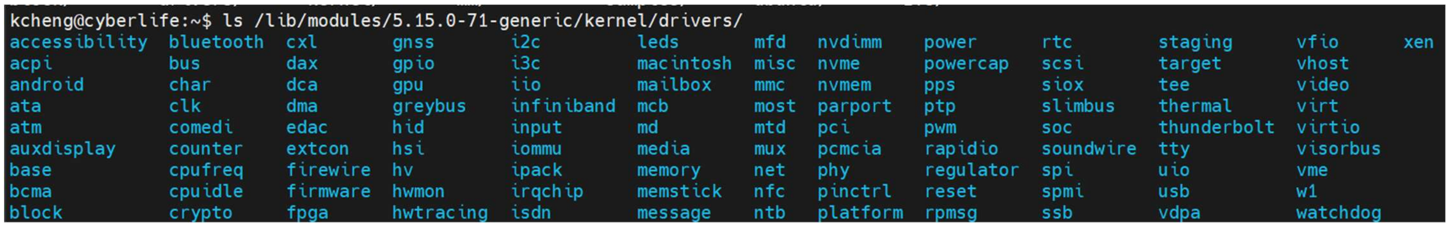
Linux Kernel can be considered as a single, big program. It is usually compressed on disk, and a few different compression methods are supported. We will not discuss initial board bring up or kernel build here. Please consult the board manufacturer for initial setup.



```
kcheng@cyberlife:~$ ls /boot -l
total 179993
-rw-r--r-- 1 root root   262250 Mar 20 08:32 config-5.15.0-69-generic
-rw-r--r-- 1 root root   262214 Apr 19 03:21 config-5.15.0-71-generic
drwx------ 3 root root     1024 Dec 31  1969 efi
drwxr-xr-x 4 root root     4096 Apr 28 06:38 grub
lrwxrwxrwx 1 root root       28 Apr 28 06:37 initrd.img -> initrd.img-5.15.0-71-generic
-rw-r--r-- 1 root root 73892417 Mar 29 06:49 initrd.img-5.15.0-69-generic
-rw-r--r-- 1 root root 73910474 Apr 28 06:37 initrd.img-5.15.0-71-generic
lrwxrwxrwx 1 root root       28 Apr 28 06:37 initrd.img.old -> initrd.img-5.15.0-69-generic
-rw-r--r-- 1 root root   182704 Aug 18  2020 memtest86+.bin
-rw-r--r-- 1 root root   184380 Aug 18  2020 memtest86+.elf
-rw-r--r-- 1 root root   184884 Aug 18  2020 memtest86+_multiboot.bin
-rw------- 1 root root  6226653 Mar 20 08:32 System.map-5.15.0-69-generic
-rw------- 1 root root  6228572 Apr 19 03:21 System.map-5.15.0-71-generic
lrwxrwxrwx 1 root root       25 Apr 28 06:37 vmlinuz -> vmlinuz-5.15.0-71-generic
-rw------- 1 root root 11468936 Mar 20 08:33 vmlinuz-5.15.0-69-generic
-rw------- 1 root root 11472584 Apr 19 04:05 vmlinuz-5.15.0-71-generic
lrwxrwxrwx 1 root root       25 Apr 28 06:37 vmlinuz.old -> vmlinuz-5.15.0-69-generic
```

Kernel modules are part of the kernel that provide certain functionality (e.g. IP stack) or they can be drivers.

They can be built into the kernel as a single big file, or they can be built as a separated, loadable file. They are stored in the ***/lib/modules/<kernel version>/*** folder. And they are further sorted into different subfolders.

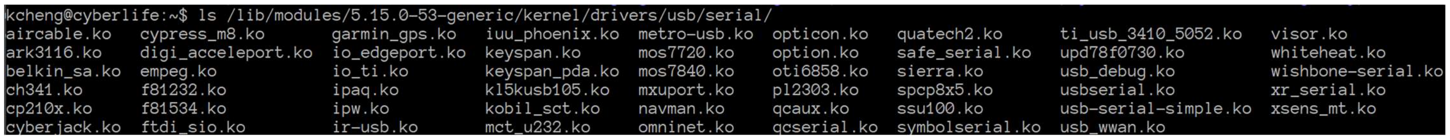


```
kcheng@cyberlife:~$ ls /lib/modules/5.15.0-71-generic/kernel/drivers/
accessibility  bluetooth  cxl       gnss      i2c         leds       mfd   nvdimm    power      rtc        staging      vfio      xen
acpi           bus        dax       gpio      i3c         macintosh  misc  nvme      powercap   scsi       target       vhost
android        char       dca       gpu       iio         mailbox    mmc   nvmem     pps        siox       tee          video
ata            clk        dma       greybus   infiniband  mcb        most  parport   ptp        slimbus    thermal      virt
atm            comedi     edac      hid       input       md         mtd   pci       pwm        soc        thunderbolt  virtio
auxdisplay     counter    extcon    hsi       iommu       media      mux   pcmcia    rapidio    soundwire  tty          visorbus
base           cpufreq    firewire  hv        ipack       memory     net   phy       regulator  spi        uio          vme
bcma           cpuidle    firmware  hwmon     irqchip     memstick   nfc   pinctrl   reset      spmi       usb          w1
block          crypto     fpga      hwtracing isdn        message    ntb   platform  rpmsg      ssb        vdpa         watchdog
```

About compressed kernel modules

Traditionally, Linux drivers are .ko (kernel object) files.

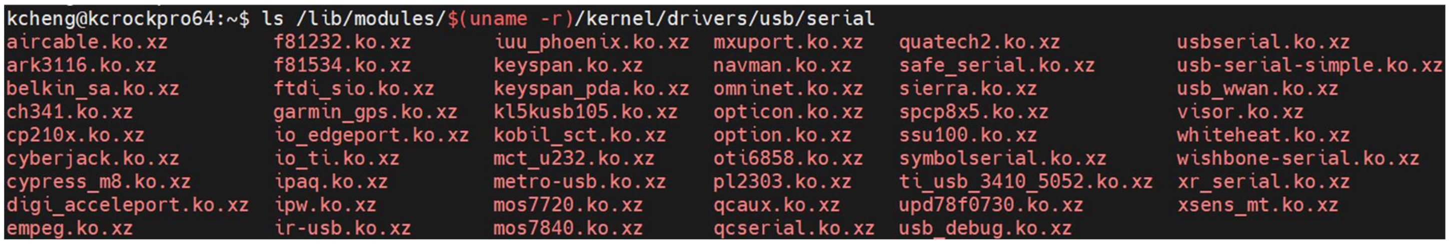
ls /lib/modules/$(uname -r)/kernel/drivers/usb/serial



```
kcheng@cyberlife:~$ ls /lib/modules/5.15.0-53-generic/kernel/drivers/usb/serial/
aircable.ko   cypress_m8.ko        garmin_gps.ko    iuu_phoenix.ko   metro-usb.ko  opticon.ko   quatech2.ko      ti_usb_3410_5052.ko    visor.ko
ark3116.ko    digi_acceleport.ko   io_edgeport.ko   keyspan.ko       mos7720.ko    option.ko    safe_serial.ko   upd78f0730.ko          whiteheat.ko
belkin_sa.ko  empeg.ko             io_ti.ko         keyspan_pda.ko   mos7840.ko    oti6858.ko   sierra.ko        usb_debug.ko           wishbone-serial.ko
ch341.ko      f81232.ko            ipaq.ko          kl5kusb105.ko    mxuport.ko    pl2303.ko    spcp8x5.ko       usbserial.ko           xr_serial.ko
cp210x.ko     f81534.ko            ipw.ko           kobil_sct.ko     navman.ko     qcaux.ko     ssu100.ko        usb-serial-simple.ko   xsens_mt.ko
cyberjack.ko  ftdi_sio.ko          ir-usb.ko        mct_u232.ko      omninet.ko    qcserial.ko  symbolserial.ko  usb_wwan.ko
```

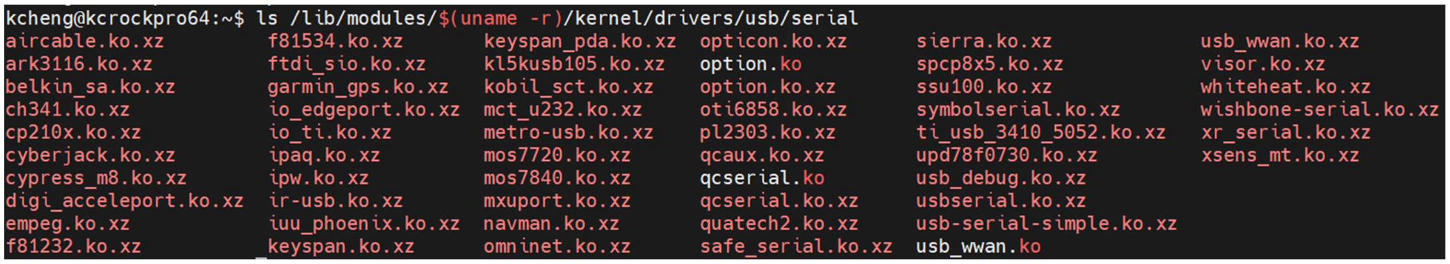
But for some newer systems, the driver could be compressed to save disk space.

ls /lib/modules/$(uname -r)/kernel/drivers/usb/serial



```
kcheng@kcrockpro64:~$ ls /lib/modules/$(uname -r)/kernel/drivers/usb/serial
aircable.ko.xz         f81232.ko.xz       iuu_phoenix.ko.xz  mxuport.ko.xz   quatech2.ko.xz          usbserial.ko.xz
ark3116.ko.xz          f81534.ko.xz       keyspan.ko.xz      navman.ko.xz    safe_serial.ko.xz       usb-serial-simple.ko.xz
belkin_sa.ko.xz        ftdi_sio.ko.xz     keyspan_pda.ko.xz  omninet.ko.xz   sierra.ko.xz            usb_wwan.ko.xz
ch341.ko.xz            garmin_gps.ko.xz   kl5kusb105.ko.xz   opticon.ko.xz   spcp8x5.ko.xz           visor.ko.xz
cp210x.ko.xz           io_edgeport.ko.xz  kobil_sct.ko.xz    option.ko.xz    ssu100.ko.xz            whiteheat.ko.xz
cyberjack.ko.xz        io_ti.ko.xz        mct_u232.ko.xz     oti6858.ko.xz   symbolserial.ko.xz      wishbone-serial.ko.xz
cypress_m8.ko.xz       ipaq.ko.xz         metro-usb.ko.xz    pl2303.ko.xz    ti_usb_3410_5052.ko.xz  xr_serial.ko.xz
digi_acceleport.ko.xz  ipw.ko.xz          mos7720.ko.xz      qcaux.ko.xz     upd78f0730.ko.xz        xsens_mt.ko.xz
empeg.ko.xz            ir-usb.ko.xz       mos7840.ko.xz      qcserial.ko.xz  usb_debug.ko.xz
```

Our driver's install script does not take this compression into account. In this case the system loads the original old compressed .ko.xz drivers instead of our new .ko drivers.



```
kcheng@kcrockpro64:~$ ls /lib/modules/$(uname -r)/kernel/drivers/usb/serial
aircable.ko.xz         f81534.ko.xz       keyspan_pda.ko.xz  opticon.ko.xz      sierra.ko.xz             usb_wwan.ko.xz
ark3116.ko.xz          ftdi_sio.ko.xz     kl5kusb105.ko.xz   option.ko          spcp8x5.ko.xz            visor.ko.xz
belkin_sa.ko.xz        garmin_gps.ko.xz   kobil_sct.ko.xz    option.ko.xz       ssu100.ko.xz             whiteheat.ko.xz
ch341.ko.xz            io_edgeport.ko.xz  mct_u232.ko.xz     oti6858.ko.xz      symbolserial.ko.xz       wishbone-serial.ko.xz
cp210x.ko.xz           io_ti.ko.xz        metro-usb.ko.xz    pl2303.ko.xz       ti_usb_3410_5052.ko.xz   xr_serial.ko.xz
cyberjack.ko.xz        ipaq.ko.xz         mos7720.ko.xz      qcaux.ko.xz        upd78f0730.ko.xz         xsens_mt.ko.xz
cypress_m8.ko.xz       ipw.ko.xz          mos7840.ko.xz      qcserial.ko        usb_debug.ko.xz
digi_acceleport.ko.xz  ir-usb.ko.xz       mxuport.ko.xz      qcserial.ko.xz     usbserial.ko.xz
empeg.ko.xz            iuu_phoenix.ko.xz  navman.ko.xz       quatech2.ko.xz     usb-serial-simple.ko.xz
f81232.ko.xz           keyspan.ko.xz      omninet.ko.xz      safe_serial.ko.xz  usb_wwan.ko
```

To remove the old drivers to avoid duplication

cd /lib/modules/$(uname -r)/kernel/drivers/usb/serial
sudo rm option.ko.xz
sudo rm qcserial.ko.xz
sudo rm usb_mon.ko.xz

and optionally compress our new version
sudo xz option.ko
sudo xz qcserial.ko
sudo xz usb_mon.ko

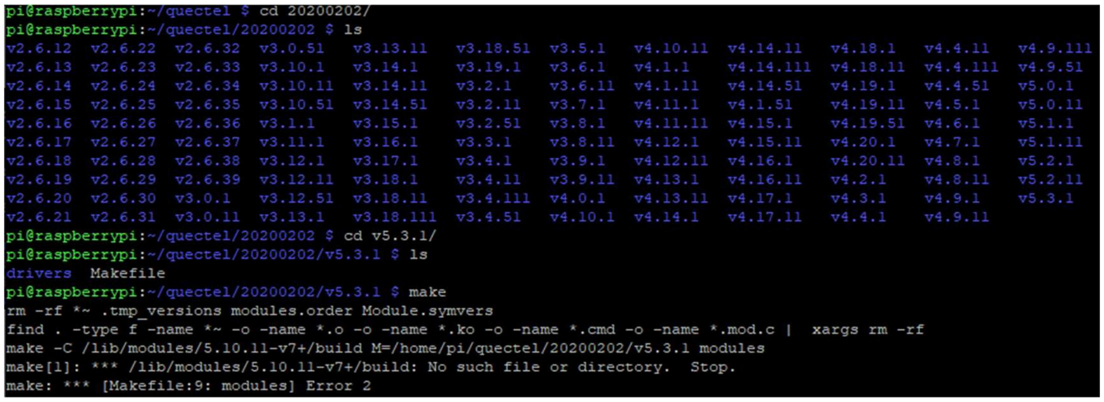
For qmi_wwan_q and pcie_mhi, they are additionally added to the system instead of replacing the old files, so there is no conflict and seems the system is loading .ko files fine.

USB serial interfaces driver, option.ko

The kernel bundled option.ko may work for our modules, but our pre-patched driver offers better compatibility and it is recommended to use our pre-patched driver. After unzipping the driver archive, look for a version that is closest to the running kernel.

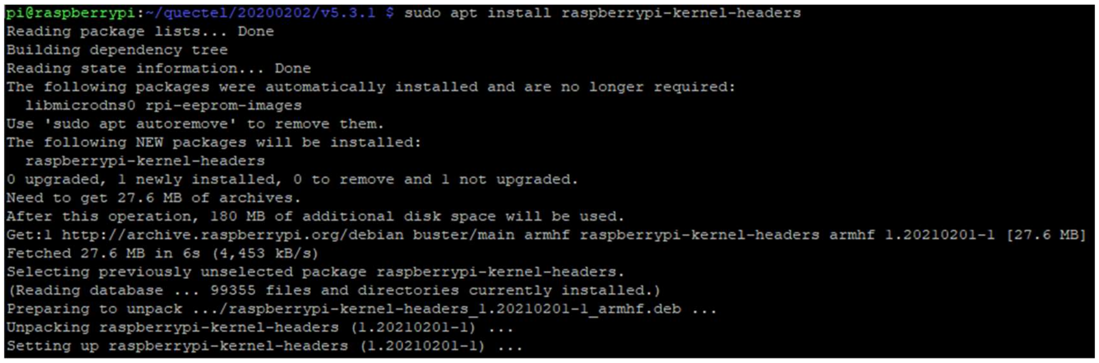
In this case I am picking v5.3.1 and execute the make command

If kernel header is not installed in the system, you may see an error like this.



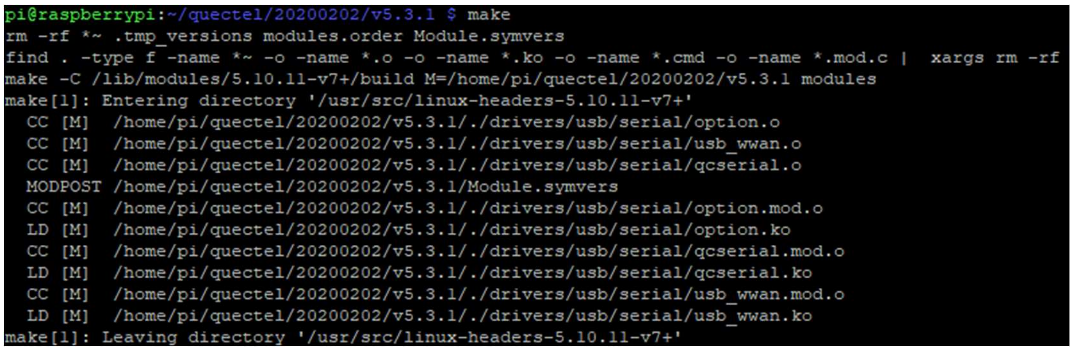
```
pi@raspberrypi:~/quectel $ cd 20200202/
pi@raspberrypi:~/quectel/20200202 $ ls
v2.6.12  v2.6.22  v2.6.32  v3.0.51   v3.13.11   v3.18.51  v3.5.1   v4.10.11  v4.14.11   v4.18.1   v4.4.11   v4.9.111
v2.6.13  v2.6.23  v2.6.33  v3.10.1   v3.14.1    v3.19.1   v3.6.1   v4.1.1    v4.14.111  v4.18.11  v4.4.111  v4.9.51
v2.6.14  v2.6.24  v2.6.34  v3.10.11  v3.14.11   v3.2.1    v3.6.11  v4.1.11   v4.14.51   v4.19.1   v4.4.51   v5.0.1
v2.6.15  v2.6.25  v2.6.35  v3.10.51  v3.14.51   v3.2.11   v3.7.1   v4.11.1   v4.1.51    v4.19.11  v4.5.1    v5.0.11
v2.6.16  v2.6.26  v2.6.36  v3.1.1    v3.15.1    v3.2.51   v3.8.1   v4.11.11  v4.15.1    v4.19.51  v4.6.1    v5.1.1
v2.6.17  v2.6.27  v2.6.37  v3.11.1   v3.16.1    v3.3.1    v3.8.11  v4.12.1   v4.15.11   v4.20.1   v4.7.1    v5.1.11
v2.6.18  v2.6.28  v2.6.38  v3.12.1   v3.17.1    v3.4.1    v3.9.1   v4.12.11  v4.16.1    v4.20.11  v4.8.1    v5.2.1
v2.6.19  v2.6.29  v2.6.39  v3.12.11  v3.18.1    v3.4.11   v3.9.11  v4.13.1   v4.16.11   v4.2.1    v4.8.11   v5.2.11
v2.6.20  v2.6.30  v3.0.1   v3.12.51  v3.18.11   v3.4.111  v4.0.1   v4.13.11  v4.17.1    v4.3.1    v4.9.1    v5.3.1
v2.6.21  v2.6.31  v3.0.11  v3.13.1   v3.18.111  v3.4.51   v4.10.1  v4.14.1   v4.17.11   v4.4.1    v4.9.11
pi@raspberrypi:~/quectel/20200202 $ cd v5.3.1/
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ ls
drivers  Makefile
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ make
rm -rf *~ .tmp_versions modules.order Module.symvers
find . -type f -name *~ -o -name *.o -o -name *.ko -o -name *.cmd -o -name *.mod.c |  xargs rm -rf
make -C /lib/modules/5.10.11-v7+/build M=/home/pi/quectel/20200202/v5.3.1 modules
make[1]: *** /lib/modules/5.10.11-v7+/build: No such file or directory.  Stop.
make: *** [Makefile:9: modules] Error 2
```

If it is not previously installed, please install it, and try again.



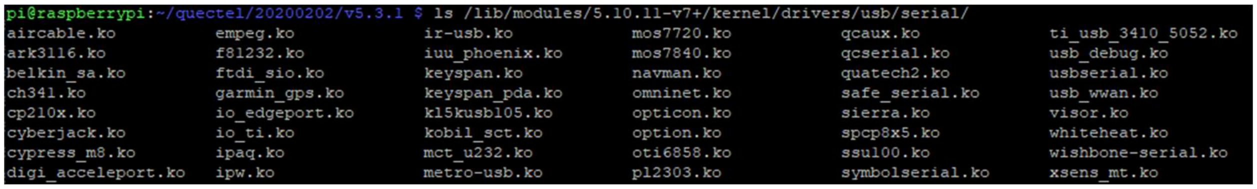
```
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ sudo apt install raspberrypi-kernel-headers
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libmicrodns0 rpi-eeprom-images
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  raspberrypi-kernel-headers
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 27.6 MB of archives.
After this operation, 180 MB of additional disk space will be used.
Get:1 http://archive.raspberrypi.org/debian buster/main armhf raspberrypi-kernel-headers armhf 1.20210201-1 [27.6 MB]
Fetched 27.6 MB in 6s (4,453 kB/s)
Selecting previously unselected package raspberrypi-kernel-headers.
(Reading database ... 99355 files and directories currently installed.)
Preparing to unpack .../raspberrypi-kernel-headers_1.20210201-1_armhf.deb ...
Unpacking raspberrypi-kernel-headers (1.20210201-1) ...
Setting up raspberrypi-kernel-headers (1.20210201-1) ...
```

We can retry make after making sure kernel headers are installed.



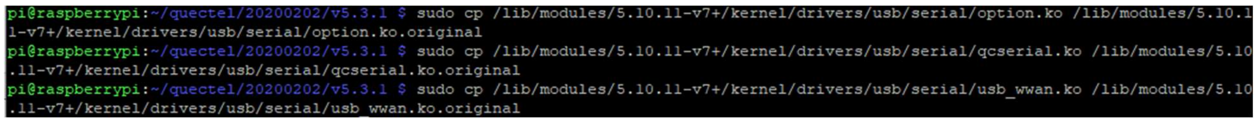
```
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ make
rm -rf *~ .tmp_versions modules.order Module.symvers
find . -type f -name *~ -o -name *.o -o -name *.ko -o -name *.cmd -o -name *.mod.c |  xargs rm -rf
make -C /lib/modules/5.10.11-v7+/build M=/home/pi/quectel/20200202/v5.3.1 modules
make[1]: Entering directory '/usr/src/linux-headers-5.10.11-v7+'
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/option.o
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/usb_wwan.o
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/qcserial.o
  MODPOST /home/pi/quectel/20200202/v5.3.1/Module.symvers
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/option.mod.o
  LD [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/option.ko
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/qcserial.mod.o
  LD [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/qcserial.ko
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/usb_wwan.mod.o
  LD [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/usb_wwan.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.10.11-v7+'
```

The .ko files are drivers in kernel module format.



```
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ ls /lib/modules/5.10.11-v7+/kernel/drivers/usb/serial/
aircable.ko          empeg.ko         ir-usb.ko        mos7720.ko   qcaux.ko          ti_usb_3410_5052.ko
ark3116.ko           f81232.ko        iuu_phoenix.ko   mos7840.ko   qcserial.ko       usb_debug.ko
belkin_sa.ko         ftdi_sio.ko      keyspan.ko       navman.ko    quatech2.ko       usbserial.ko
ch341.ko             garmin_gps.ko    keyspan_pda.ko   omninet.ko   safe_serial.ko    usb_wwan.ko
cp210x.ko            io_edgeport.ko   kl5kusb105.ko    opticon.ko   sierra.ko         visor.ko
cyberjack.ko         io_ti.ko         kobil_sct.ko     option.ko    spcp8x5.ko        whiteheat.ko
cypress_m8.ko        ipaq.ko          mct_u232.ko      oti6858.ko   ssu100.ko         wishbone-serial.ko
digi_acceleport.ko   ipw.ko           metro-usb.ko     pl2303.ko    symbolserial.ko   xsens_mt.ko
```

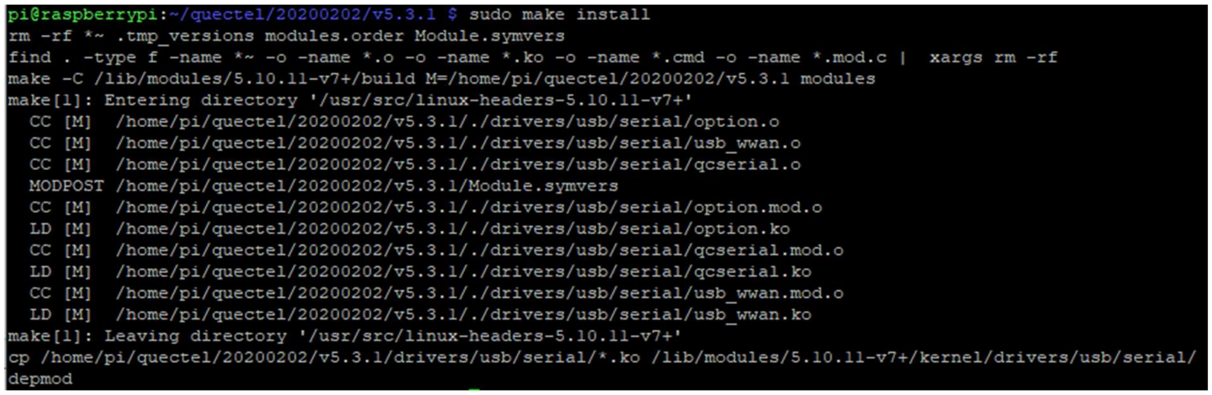
Our install script doesn't keep the original files, you may want to back up the original files, but typically it is not needed.



```
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ sudo cp /lib/modules/5.10.11-v7+/kernel/drivers/usb/serial/option.ko /lib/modules/5.10.1
1-v7+/kernel/drivers/usb/serial/option.ko.original
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ sudo cp /lib/modules/5.10.11-v7+/kernel/drivers/usb/serial/qcserial.ko /lib/modules/5.10
.11-v7+/kernel/drivers/usb/serial/qcserial.ko.original
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ sudo cp /lib/modules/5.10.11-v7+/kernel/drivers/usb/serial/usb_wwan.ko /lib/modules/5.10
.11-v7+/kernel/drivers/usb/serial/usb_wwan.ko.original
```

And we can install the driver with this command.

sudo make install



```
pi@raspberrypi:~/quectel/20200202/v5.3.1 $ sudo make install
rm -rf *~ .tmp_versions modules.order Module.symvers
find . -type f -name *~ -o -name *.o -o -name *.ko -o -name *.cmd -o -name *.mod.c |  xargs rm -rf
make -C /lib/modules/5.10.11-v7+/build M=/home/pi/quectel/20200202/v5.3.1 modules
make[1]: Entering directory '/usr/src/linux-headers-5.10.11-v7+'
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/option.o
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/usb_wwan.o
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/qcserial.o
  MODPOST /home/pi/quectel/20200202/v5.3.1/Module.symvers
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/option.mod.o
  LD [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/option.ko
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/qcserial.mod.o
  LD [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/qcserial.ko
  CC [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/usb_wwan.mod.o
  LD [M]  /home/pi/quectel/20200202/v5.3.1/./drivers/usb/serial/usb_wwan.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.10.11-v7+'
cp /home/pi/quectel/20200202/v5.3.1/drivers/usb/serial/*.ko /lib/modules/5.10.11-v7+/kernel/drivers/usb/serial/
depmod
```

General driver loading commands

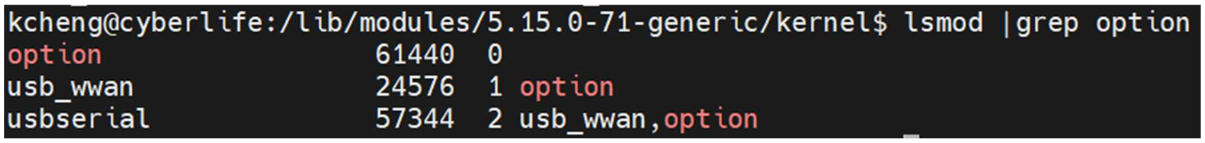
When we want to install our own version of the drivers (kernel modules), It is possible that the drivers from the original kernel are already running. We can find out the list of modules in memory by this command.

lsmod

And in this example, I am listing out entries with the text string “option” in it. We can see that the kernel module “option” is loaded but it is not in use by another module (number 0)

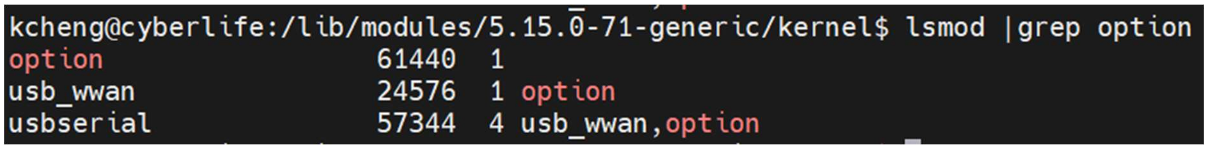
usb_wwan is required by usb_wwan, and usbserial is required by both usb_wwan and option.

lsusb |grep option



```
kcheng@cyberlife:/lib/modules/5.15.0-71-generic/kernel$ lsmod |grep option
option                 61440  0
usb_wwan               24576  1 option
usbserial              57344  2 usb_wwan,option
```

In this example, I started a software talking to my cellular module via the option driver. We can see the number of dependencies has become 1.



```
kcheng@cyberlife:/lib/modules/5.15.0-71-generic/kernel$ lsmod |grep option
option                 61440  1
usb_wwan               24576  1 option
usbserial              57344  4 usb_wwan,option
```

We can remove the driver from memory by the “modprobe -r” command and load the new driver from disk again with the “modprobe” command. Or you can just reboot the system so that everything is flash.

pi@raspberrypi:~/quectel/20200202/v5.3.1 $ sudo modprobe -r option

pi@raspberrypi:~/quectel/20200202/v5.3.1 $ sudo modprobe option

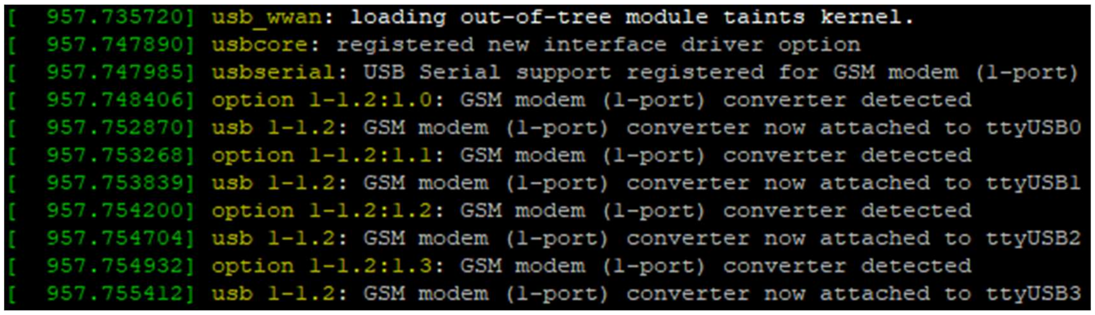
With the ***dmesg*** command we can see that the new driver is detecting 4 serial ports. I am using EG25-G as an example.

The taints kernel warning means that Linux is not able to verify open-source status as our driver is not built altogether with the kernel. It is not harmful.

In this example, ttyUSB0 to ttyUSB3 are detected by the system. If you have another USB serial device in the system, they may share the same ttyUSB name so that numbers may change.

Quectel cellular modules usually provide these USB serial ports in this order. But there are exceptions.

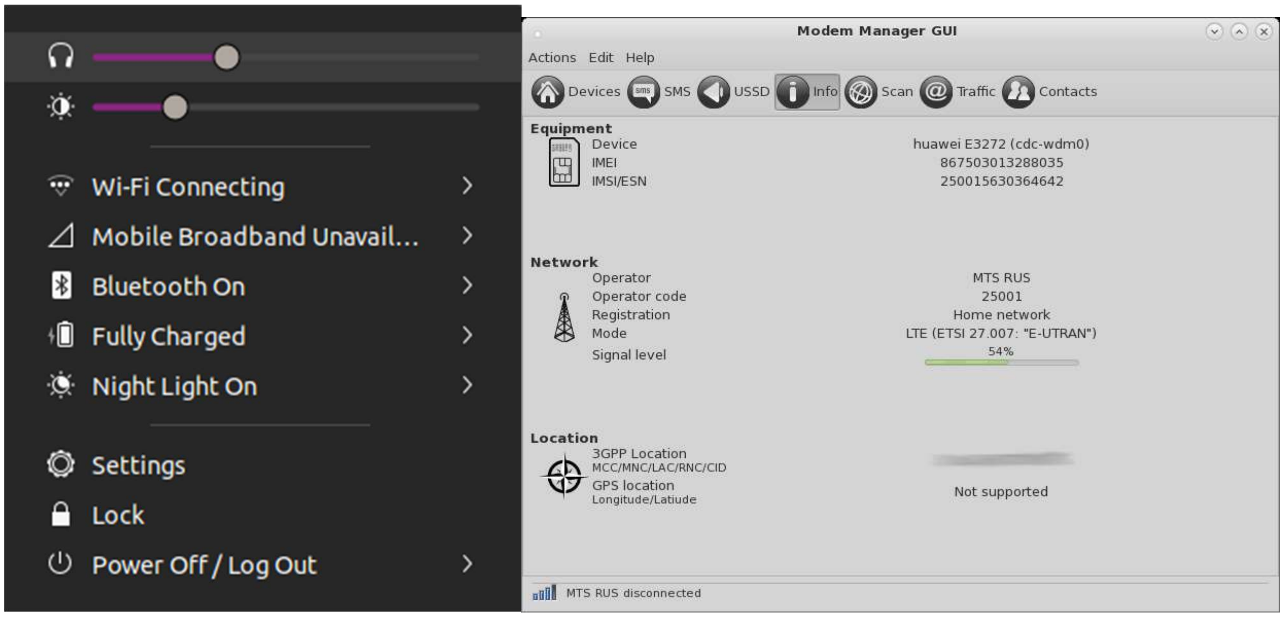
ttyUSB0: Module debug port, aka DM port. Provide access to module debug information.
ttyUSB1: NMEA port. Provide access to GNSS/GPS NMEA messages.
ttyUSB2: AT/Modem port. Provide access to AT command, and optionally PPP data call.
ttyUSB3: AT/Modem port. Provide access to AT command, and optionally PPP data call.



```
[  957.735720] usb_wwan: loading out-of-tree module taints kernel.
[  957.747890] usbcore: registered new interface driver option
[  957.747985] usbserial: USB Serial support registered for GSM modem (1-port)
[  957.748406] option 1-1.2:1.0: GSM modem (1-port) converter detected
[  957.752870] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB0
[  957.753268] option 1-1.2:1.1: GSM modem (1-port) converter detected
[  957.753839] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB1
[  957.754200] option 1-1.2:1.2: GSM modem (1-port) converter detected
[  957.754704] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB2
[  957.754932] option 1-1.2:1.3: GSM modem (1-port) converter detected
[  957.755412] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB3
```

Modem Manager

Many modern Linux distributions come with a software called modem manager. It provides both graphical and command line-based control over cellular modules, but it is not too mature in my point of view, the information from GUI and command line can go out of sync. Also, it has a long list of software dependency, and is quite big in terms of disk size.



Equipment
Device huawei E3272 (cdc-wdm0)
IMEI 867503013288035
IMSI/ESN 250015630364642

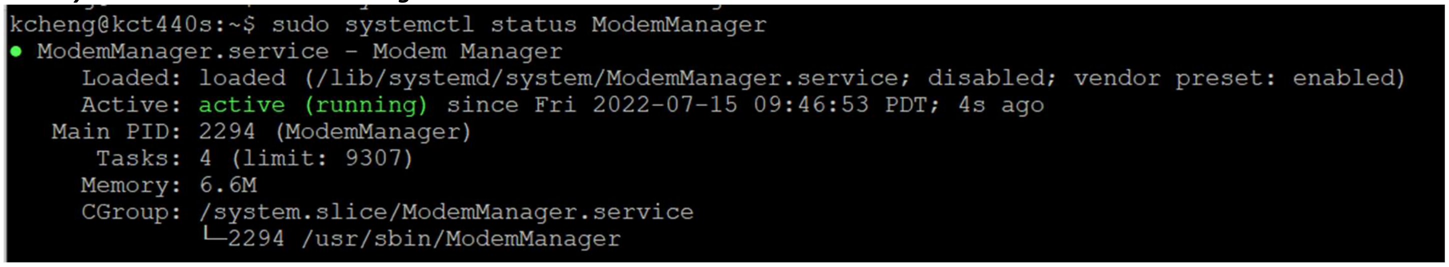
Network
Operator MTS RUS
Operator code 25001
Registration Home network
Mode LTE (ETSI 27.007: "E-UTRAN")
Signal level 54%

Location
3GPP Location MCC/MNC/LAC/RNC/CID
GPS location Longitude/Latiude Not supported

MTS RUS disconnected

For embedded system use it is not ideal. Plus, it keeps on talking to the module and will affect our test. We recommend customers to use our quectel-CM data call manager instead.

To check the status of modem manager, use this command.
sudo systemctl status ModemManager



```
kcheng@kct440s:~$ sudo systemctl status ModemManager
● ModemManager.service - Modem Manager
     Loaded: loaded (/lib/systemd/system/ModemManager.service; disabled; vendor preset: enabled)
     Active: active (running) since Fri 2022-07-15 09:46:53 PDT; 4s ago
   Main PID: 2294 (ModemManager)
      Tasks: 4 (limit: 9307)
     Memory: 6.6M
     CGroup: /system.slice/ModemManager.service
             └─2294 /usr/sbin/ModemManager
```

To stop modem manager,
sudo systemctl stop ModemManager

To start modem manager,
sudo systemctl start ModemManager

To stop modem manager from auto startup during system boot,
sudo systemctl disable ModemManager

To restore automatic startup of modem manager,
sudo systemctl enable ModemManager

To uninstall modem manager altogether with its settings,
sudo apt-get purge modemmanager

Serial terminals

To access AT command interactively with a USB module, we can use the minicom command.

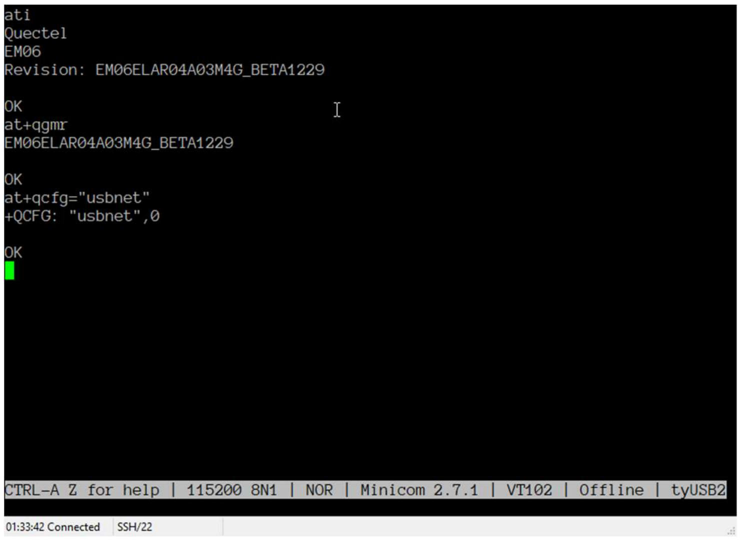
sudo minicom -w -D /dev/ttyUSB2

Just type the command and press ENTER to execute. To exit minicom, press CTRL+A first, and press X to exit.

The -w parameter tells minicom to wrap the text into new line if it is too long, and -D specify the device node of the AT command prompt. The device name could change for some modules with different design, or if there are some other USB to serial devices in the system.

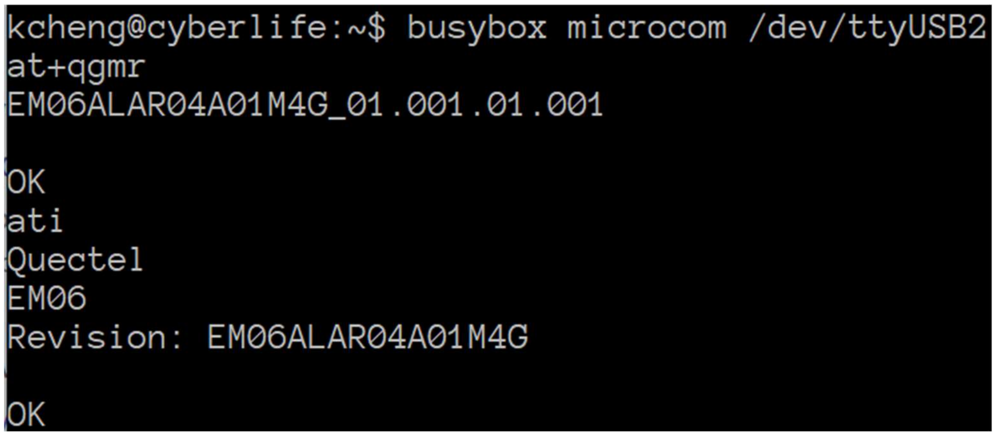
For PCI Express modules, the device node for AT command access is **/dev/mhi_DUN**

In this example, I am using the command AT+QGMR to read the module's detailed firmware information, and AT+QCFG="usbnet" to check the module's current "usbnet" mode, which is 0 in this case. Generally, we use mode 0(QMI) and 2(MBIM)



```
ati
Quectel
EM06
Revision: EM06ELAR04A03M4G_BETA1229

OK
at+qgmr
EM06ELAR04A03M4G_BETA1229

OK
at+qcfg="usbnet"
+QCFG: "usbnet",0

OK

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | tyUSB2
01:33:42 Connected   SSH/22
```

microcom is a smaller application like minicom. CTRL+X to exit.



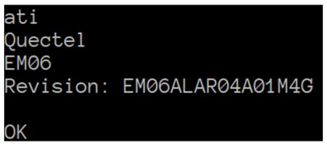
```
kcheng@cyberlife:~$ busybox microcom /dev/ttyUSB2
at+qgmr
EM06ALAR04A01M4G_01.001.01.001

OK
ati
Quectel
EM06
Revision:  EM06ALAR04A01M4G

OK
```

Types of AT commands

https://en.wikipedia.org/wiki/Hayes_command_set

AT command was introduced been since 1981 and has been extended again and again as network technology evolves. It is not surprising that the design is not very unified. There are a few different forms.

Execution type, a simple command with no parameters, e.g.
ATI

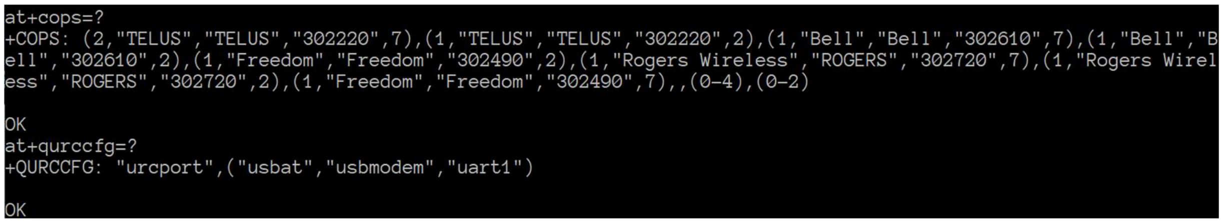


```
ati
Quectel
EM06
Revision: EM06ALAR04A01M4G

OK
```

Execution type, but the parameter is directly added to as suffix, e.g.
ATE1

Help/Query type, a command ending with =?. The first one is a network scan which may take some time to complete, and the second one provides help information only.

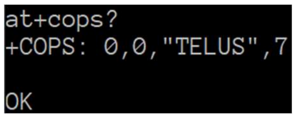
at+cops=? and ***at+qurccfg=?***



```
at+cops=?
+COPS: (2,"TELUS","TELUS","302220",7),(1,"TELUS","TELUS","302220",2),(1,"Bell","Bell","302610",7),(1,"Bell","B
ell","302610",2),(1,"Freedom","Freedom","302490",2),(1,"Rogers Wireless","ROGERS","302720",7),(1,"Rogers Wirel
ess","ROGERS","302720",2),(1,"Freedom","Freedom","302490",7),,(0-4),(0-2)

OK
at+qurccfg=?
+QURCCFG: "urcport",("usbat","usbmodem","uart1")

OK
```

Query type, a command with a question mark

at+cops?



```
at+cops?
+COPS: 0,0,"TELUS",7

OK
```

Value writing type, a command with a value assignment. e.g.
AT+CFUN=1

Commands with a string as verb
AT+QENG="SERVINGCELL"

Commands with a string as verb, plus parameters
at+qcfg="dbgctl",0

URCs

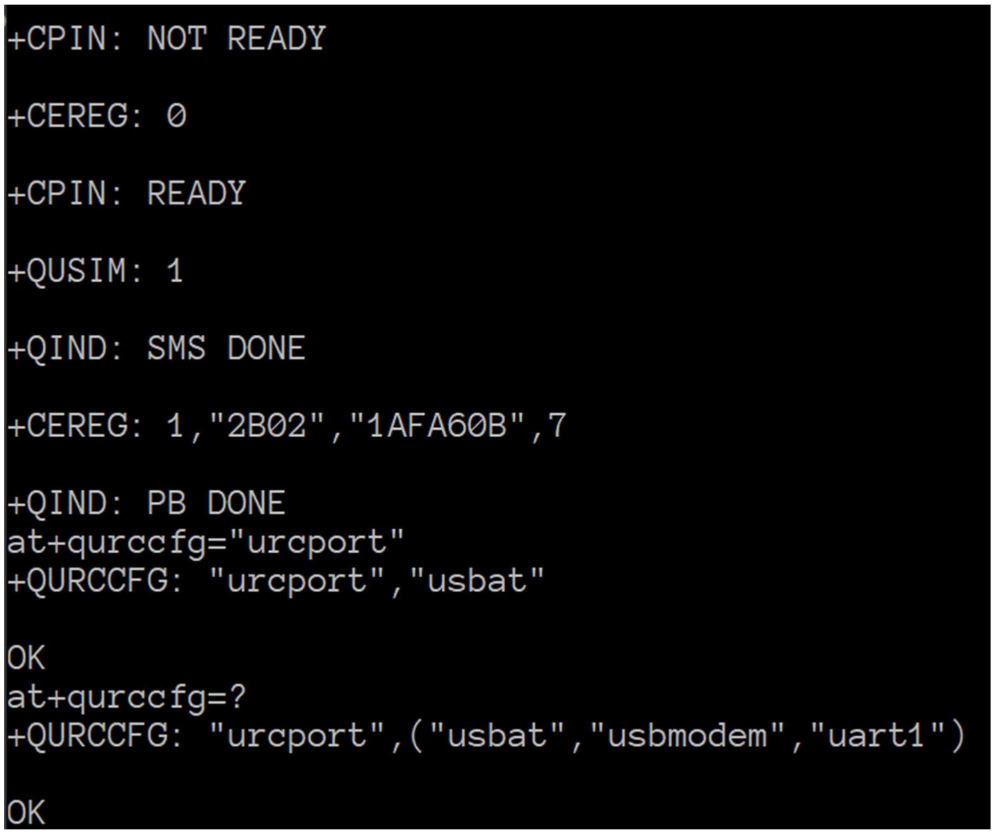
URC stands for unsolicited result code; they are string prompts coming to the port to indicate events. The most common URCs are +CPIN that indicates SIM PIN lock status, and +CEREG that indicates network registration events.
Note that URCs could be configured to go to specific AT port only, if the module has multiple AT/Modem ports. The command to configure URC is AT+QURCCFG. Sometimes it could be easier to handle AT commands and URC programmatically if URC goes to another port.

To read current value
AT+QURCCFG="urcport"

To check for valid values
AT+QURCCFG=?

To modify the setting
AT+QURCCFG="urcport","usbat"

In this example I have turned on GPIO based SIM hot-plug and inserted a SIM.
+CPIN indicates the status of the PIN status. It changes from NOT READY to READY.
+QIND: SMS DONE tells you that SMS initialization has completed.
+CEREG tells you that the module has register to a cell, radio technology is 7 (LTE)
+QIND: PB DONE tells you that the phone book initialization has completed.

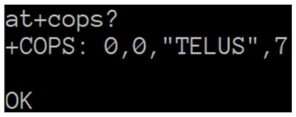


```
+CPIN: NOT READY

+CEREG: 0

+CPIN: READY

+QUSIM: 1

+QIND: SMS DONE

+CEREG: 1,"2B02","1AFA60B",7

+QIND: PB DONE
at+qurccfg="urcport"
+QURCCFG: "urcport","usbat"

OK
at+qurccfg=?
+QURCCFG: "urcport",("usbat","usbmodem","uart1")

OK
```

Frequently used AT commands

Carrier name and connected cell technology check. The result code 7 means LTE.

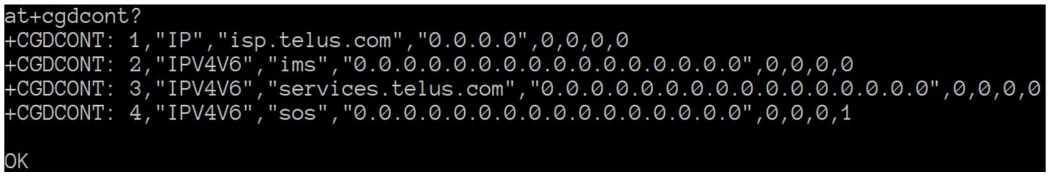
at+cops?



```
at+cops?
+COPS: 0,0,"TELUS",7

OK
```

APN check

at+cgdcont?



```
at+cgdcont?
+CGDCONT: 1,"IP","isp.telus.com","0.0.0.0",0,0,0,0
+CGDCONT: 2,"IPV4V6","ims","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 3,"IPV4V6","services.telus.com","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 4,"IPV4V6","sos","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1

OK
```

APN setup

Setting up APN #1, type IPV4V6, with the name isp.telus.com

at+cgdcont=1,"IPV4V6","isp.telus.com"

Deleting an APN (#4)

AT+CGDCONT=4

Check IP address assigned to PDP.

at+cgcontrdp

Network attach failure reason, supported by newer modules only.

AT+QCFG="emmcause"

Read module IMEI number.

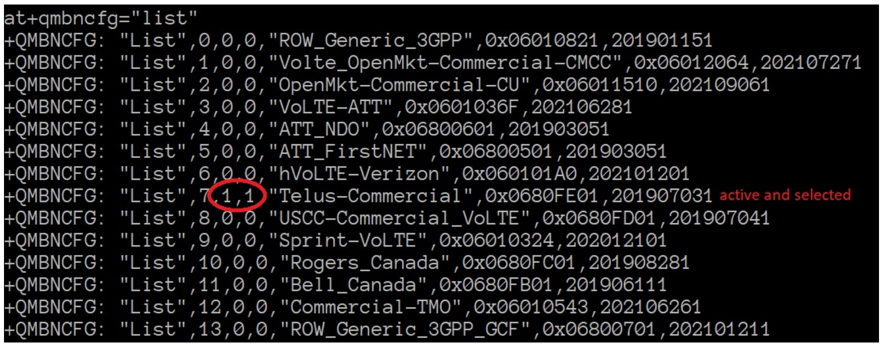
AT+CGSN

Read SIM IMSI number, also useful to confirm if SIM is detected.

AT+CIMI

Read carrier profile list.

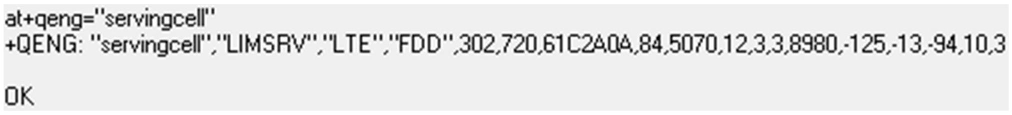
AT+QMBNCFG="list"



```
at+qmbncfg="list"
+QMBNCFG: "List",0,0,0,"ROW_Generic_3GPP",0x06010821,201901151
+QMBNCFG: "List",1,0,0,"Volte_OpenMkt-Commercial-CMCC",0x06012064,202107271
+QMBNCFG: "List",2,0,0,"OpenMkt-Commercial-CU",0x06011510,202109061
+QMBNCFG: "List",3,0,0,"VoLTE-ATT",0x0601036F,202106281
+QMBNCFG: "List",4,0,0,"ATT_NDO",0x06800601,201903051
+QMBNCFG: "List",5,0,0,"ATT_FirstNET",0x06800501,201903051
+QMBNCFG: "List",6,0,0,"hVoLTE-Verizon",0x060101A0,202101201
+QMBNCFG: "List",7,1,1,"Telus-Commercial",0x0680FE01,201907031 active and selected
+QMBNCFG: "List",8,0,0,"USCC-Commercial_VoLTE",0x0680FD01,201907041
+QMBNCFG: "List",9,0,0,"Sprint-VoLTE",0x06010324,202012101
+QMBNCFG: "List",10,0,0,"Rogers_Canada",0x0680FC01,201908281
+QMBNCFG: "List",11,0,0,"Bell_Canada",0x0680FB01,201906111
+QMBNCFG: "List",12,0,0,"Commercial-TMO",0x06010543,202106261
+QMBNCFG: "List",13,0,0,"ROW_Generic_3GPP_GCF",0x06800701,202101211
```

Get information of the currently connected cell

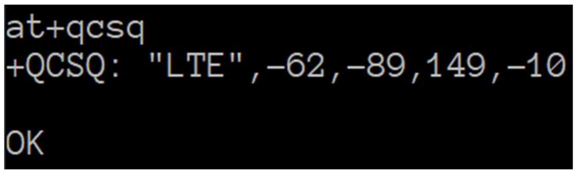
at+qeng="servingcell"



```
at+qeng="servingcell"
+QENG: "servingcell","LIMSRV","LTE","FDD",302,720,61C2A0A,84,5070,12,3,3,8980,-125,-13,-94,10,3

OK
```

Checking signal level

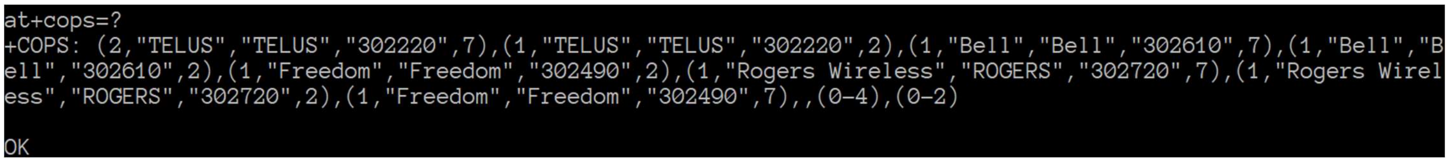
AT+QCSQ



```
at+qcsq
+QCSQ: "LTE",-62,-89,149,-10

OK
```

Network carrier scan, it will take some time to finish. And the module will be disconnected from the network during the scan

AT+COPS=?



```
at+cops=?
+COPS: (2,"TELUS","TELUS","302220",7),(1,"TELUS","TELUS","302220",2),(1,"Bell","Bell","302610",7),(1,"Bell","B
ell","302610",2),(1,"Freedom","Freedom","302490",2),(1,"Rogers Wireless","ROGERS","302720",7),(1,"Rogers Wirel
ess","ROGERS","302720",2),(1,"Freedom","Freedom","302490",7),,(0-4),(0-2)

OK
```

Command mode and data mode

Both the AT command and the Modem port operate in two different modes.

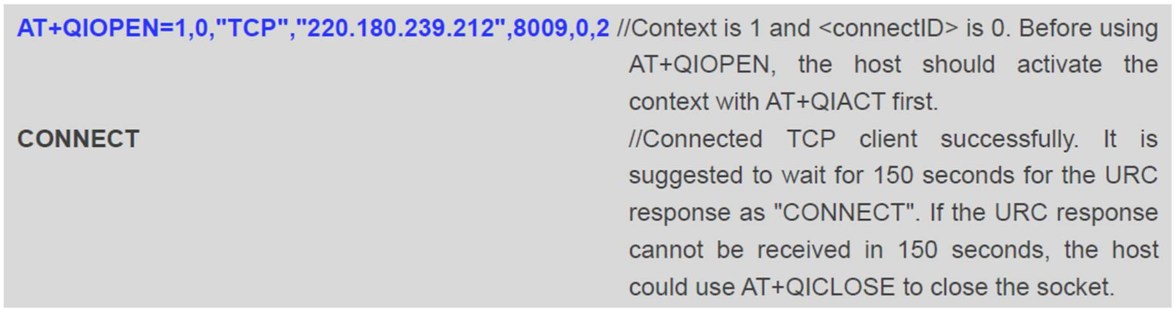
Typically, the ports operate AT command modes. This is the standard mode we tested with different AT commands in the previous chapter.

When there is a need, the port can turn into DATA mode. It is indicated by a “CONNECT” URC and the DCD signal of the port will be turned on.

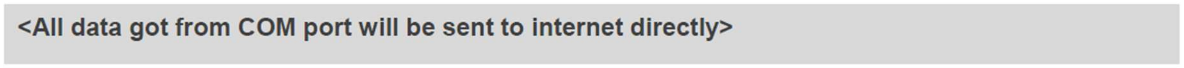
In this mode instead of AT commands, we can send data to the data session.

e.g., opening TCP connection with a transparent access mode using the AT+QIOPEN command

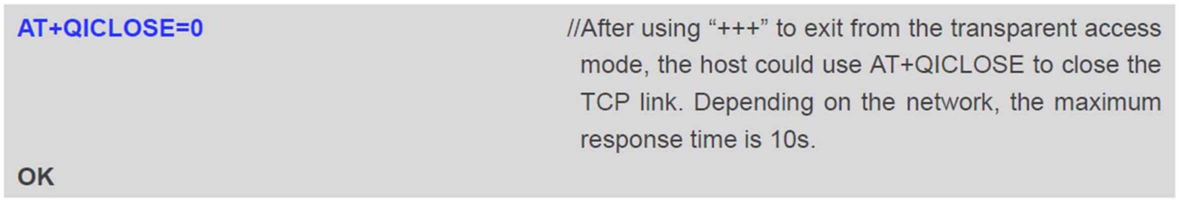
Quoting from the Quectel_BG96_TCP(IP)_AT_Commands_Manual_V1.1.pdf



```
AT+QIOPEN=1,0,"TCP","220.180.239.212",8009,0,2 //Context is 1 and <connectID> is 0. Before using
                                                 AT+QIOPEN, the host should activate the
                                                 context with AT+QIACT first.
CONNECT                                          //Connected TCP client successfully. It is
                                                 suggested to wait for 150 seconds for the URC
                                                 response as "CONNECT". If the URC response
                                                 cannot be received in 150 seconds, the host
                                                 could use AT+QICLOSE to close the socket.
```



<All data got from COM port will be sent to internet directly>



```
AT+QICLOSE=0                        //After using “+++” to exit from the transparent access
                                      mode, the host could use AT+QICLOSE to close the
                                      TCP link. Depending on the network, the maximum
                                      response time is 10s.
OK
```

While the port is in DATA mode, we cannot send AT commands with the same port. We can switch back to AT command mode by using the +++ escape sequence.

- Stop sending data for 1 second.
- Send +++ (3 plus sign) quickly.
- Stop sending data for 1 second again.
- The module will prompt OK to indicate it is back to command mode.
- Some data connection can be held for some time, as specified by the command that triggered data mode. The conditions for holding the data session in the background are different for different commands. Check the document for the command for details.
- Use the ATO command to go back to the data session which is on hold.

The other use of the data mode is the PPP data call. More about it in the following chapter.

USB network adapter modes

Our high-speed modules support different usb network adapter(usbnet) modes. Slower LPWA (low power wide area) modules could be limited to serial ports though.

0: RMNET/QMI: The Qualcomm preparatory mode, supported by the kernel qmi_wwan.ko driver, and our branched qmi_wwan_q.ko driver which adds more function. It takes a data call manager like our quectel-CM to start and maintain a data call. There is also a Qualcomm GobiNet driver which is now depreciated. QMI is an alternative module control command set, which is binary and not human readable.

1: ECM mode: In this mode the module will maintain the data call automatically, and act as a router. i.e., the host system will be allocated a 192.168.x.x local private IP address via DHCP. it is supported by the Linux kernel bundled cdc_ether.ko driver. It is easier to use, you can go to the internet, but not accepting incoming connections. i.e. you cannot host a server.

2: MBIM mode: This is the new industrial standard, newer modules default to this mode. Both Windows and Linux support MBIM natively. It is supported by the kernel bundled cdc_mbim.ko driver. It takes a data call manager like our quectel-CM to start and maintain a data call. MBIM is an alternative module control command set, which is binary and not human readable.

There are pros and cons for all these modes, you can pick on that suits you needs most.

To read current USBNET mode,

AT+QCFG="USBNET"

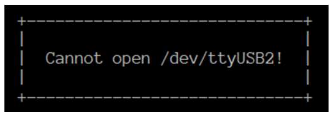
To switch to QMI mode (0), use this command.

AT+QCFG="USBNET", 0

Some modules reboot themselves automatically, some require a manual restart. To restart the module, use this AT command.

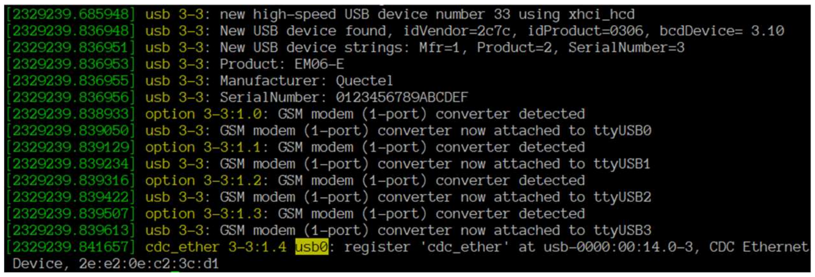
AT+CFUN=1,1

During a module reboot, minicom reports the AT port is not accessible like this.



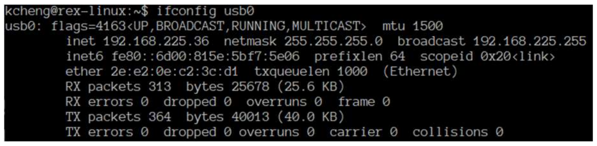
```
+-----------------------------+
|                             |
|  Cannot open /dev/ttyUSB2!  |
|                             |
+-----------------------------+
```

In this example, the module is in ECM mode and the network interface is detected as usb0.



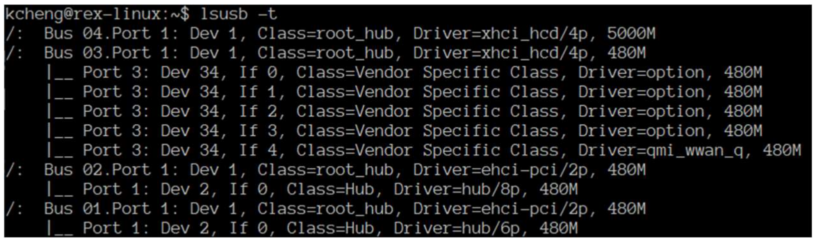
```
[2329239.685948] usb 3-3: new high-speed USB device number 33 using xhci_hcd
[2329239.836948] usb 3-3: New USB device found, idVendor=2c7c, idProduct=0306, bcdDevice= 3.10
[2329239.836951] usb 3-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[2329239.836953] usb 3-3: Product: EM06-E
[2329239.836955] usb 3-3: Manufacturer: Quectel
[2329239.836956] usb 3-3: SerialNumber: 0123456789ABCDEF
[2329239.838933] option 3-3:1.0: GSM modem (1-port) converter detected
[2329239.839050] usb 3-3: GSM modem (1-port) converter now attached to ttyUSB0
[2329239.839129] option 3-3:1.1: GSM modem (1-port) converter detected
[2329239.839234] usb 3-3: GSM modem (1-port) converter now attached to ttyUSB1
[2329239.839316] option 3-3:1.2: GSM modem (1-port) converter detected
[2329239.839422] usb 3-3: GSM modem (1-port) converter now attached to ttyUSB2
[2329239.839507] option 3-3:1.3: GSM modem (1-port) converter detected
[2329239.839613] usb 3-3: GSM modem (1-port) converter now attached to ttyUSB3
[2329239.841657] cdc_ether 3-3:1.4 usb0: register 'cdc_ether' at usb-0000:00:14.0-3, CDC Ethernet
 Device, 2e:e2:0e:c2:3c:d1
```

For ECM, the IP address of the USB0 interface is usually setup by DHCP, you should be able to go online without a data call manager if the cellular settings like APN and frequency band are correct.



```
kcheng@rex-linux:~$ ifconfig usb0
usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.225.36  netmask 255.255.255.0  broadcast 192.168.225.255
        inet6 fe80::6d00:815e:5bf7:5e06  prefixlen 64  scopeid 0x20<link>
        ether 2e:e2:0e:c2:3c:d1  txqueuelen 1000  (Ethernet)
        RX packets 313  bytes 25678 (25.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 364  bytes 40013 (40.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

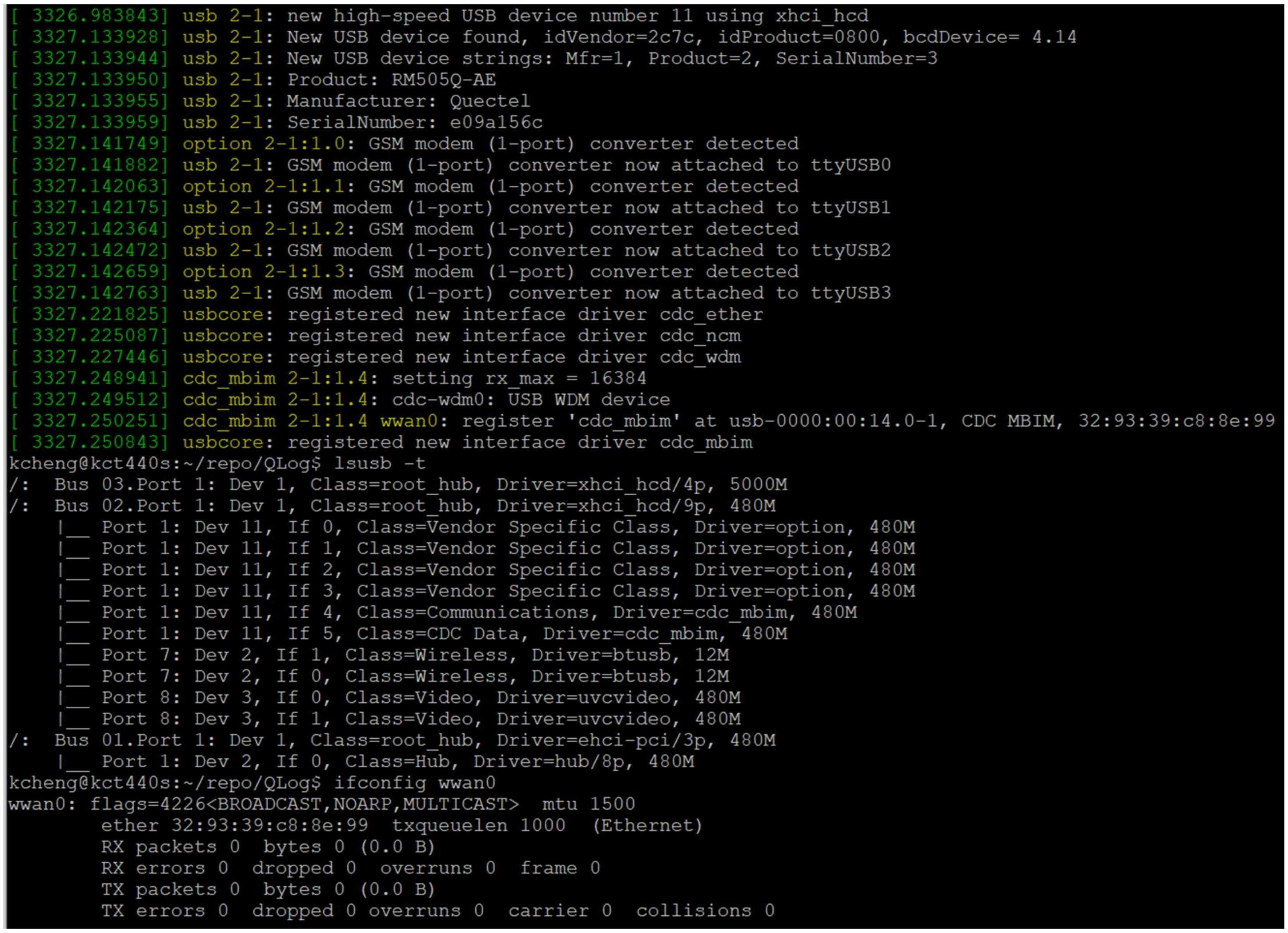
This is a module in QMI/RMNET mode. The driver in use could be the kernel built in qmi_wwan or our qmi_wwan_q



```
kcheng@rex-linux:~$ lsusb -t
/:  Bus 04.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/4p, 5000M
/:  Bus 03.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/4p, 480M
    |__ Port 3: Dev 34, If 0, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 3: Dev 34, If 1, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 3: Dev 34, If 2, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 3: Dev 34, If 3, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 3: Dev 34, If 4, Class=Vendor Specific Class, Driver=qmi_wwan_q, 480M
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/2p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/8p, 480M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/2p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/6p, 480M
```

This is an example of modules running in MBIM mode.

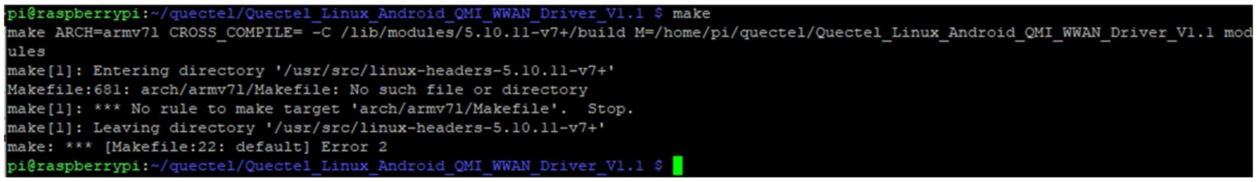
The driver handling the MBIM port is cdc_mbim, there could be 1 or 2 interfaces (interface 4 and 5 in this case).



```
[ 3326.983843] usb 2-1: new high-speed USB device number 11 using xhci_hcd
[ 3327.133928] usb 2-1: New USB device found, idVendor=2c7c, idProduct=0800, bcdDevice= 4.14
[ 3327.133944] usb 2-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 3327.133950] usb 2-1: Product: RM505Q-AE
[ 3327.133955] usb 2-1: Manufacturer: Quectel
[ 3327.133959] usb 2-1: SerialNumber: e09a156c
[ 3327.141749] option 2-1:1.0: GSM modem (1-port) converter detected
[ 3327.141882] usb 2-1: GSM modem (1-port) converter now attached to ttyUSB0
[ 3327.142063] option 2-1:1.1: GSM modem (1-port) converter detected
[ 3327.142175] usb 2-1: GSM modem (1-port) converter now attached to ttyUSB1
[ 3327.142364] option 2-1:1.2: GSM modem (1-port) converter detected
[ 3327.142472] usb 2-1: GSM modem (1-port) converter now attached to ttyUSB2
[ 3327.142659] option 2-1:1.3: GSM modem (1-port) converter detected
[ 3327.142763] usb 2-1: GSM modem (1-port) converter now attached to ttyUSB3
[ 3327.221825] usbcore: registered new interface driver cdc_ether
[ 3327.225087] usbcore: registered new interface driver cdc_ncm
[ 3327.227446] usbcore: registered new interface driver cdc_wdm
[ 3327.248941] cdc_mbim 2-1:1.4: setting rx_max = 16384
[ 3327.249512] cdc_mbim 2-1:1.4: cdc-wdm0: USB WDM device
[ 3327.250251] cdc_mbim 2-1:1.4 wwan0: register 'cdc_mbim' at usb-0000:00:14.0-1, CDC MBIM, 32:93:39:c8:8e:99
[ 3327.250843] usbcore: registered new interface driver cdc_mbim
kcheng@kct440s:~/repo/QLog$ lsusb -t
/:  Bus 03.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/4p, 5000M
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/9p, 480M
    |__ Port 1: Dev 11, If 0, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 1: Dev 11, If 1, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 1: Dev 11, If 2, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 1: Dev 11, If 3, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 1: Dev 11, If 4, Class=Communications, Driver=cdc_mbim, 480M
    |__ Port 1: Dev 11, If 5, Class=CDC Data, Driver=cdc_mbim, 480M
    |__ Port 7: Dev 2, If 1, Class=Wireless, Driver=btusb, 12M
    |__ Port 7: Dev 2, If 0, Class=Wireless, Driver=btusb, 12M
    |__ Port 8: Dev 3, If 0, Class=Video, Driver=uvcvideo, 480M
    |__ Port 8: Dev 3, If 1, Class=Video, Driver=uvcvideo, 480M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/3p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/8p, 480M
kcheng@kct440s:~/repo/QLog$ ifconfig wwan0
wwan0: flags=4226<BROADCAST,NOARP,MULTICAST>  mtu 1500
        ether 32:93:39:c8:8e:99  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

USB QMI/RMNET driver qmi_wwan_q.ko

The driver is provided in source code form, we have to build the ethernet driver qmi_wwan_q.ko for the target board. Note that in Raspberry Pi, the operation system could be 32bit or 64bit and sometimes the build system is not able to detect which architecture is in use, you may see errors like this.



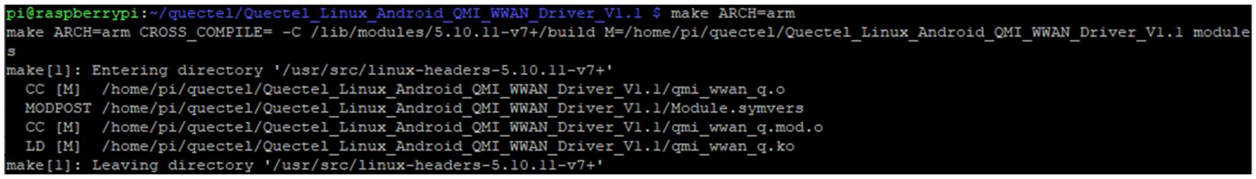
```
pi@raspberrypi:~/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 $ make
make ARCH=armv7l CROSS_COMPILE= -C /lib/modules/5.10.11-v7+/build M=/home/pi/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 mod
ules
make[1]: Entering directory '/usr/src/linux-headers-5.10.11-v7+'
Makefile:681: arch/armv7l/Makefile: No such file or directory
make[1]: *** No rule to make target 'arch/armv7l/Makefile'.  Stop.
make[1]: Leaving directory '/usr/src/linux-headers-5.10.11-v7+'
make: *** [Makefile:22: default] Error 2
pi@raspberrypi:~/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 $
```

To build it correctly we can specify the correct CPU architecture. I am using 32bit arm in my case. The name of the architecture depends on the operating system in use (Raspberry OS, Ubuntu, Armbian). Some of them use the name **arm (and 64)** while some of the use the name **aarch (and 64)**

make ARCH=arm

or

make ARCH=arm64



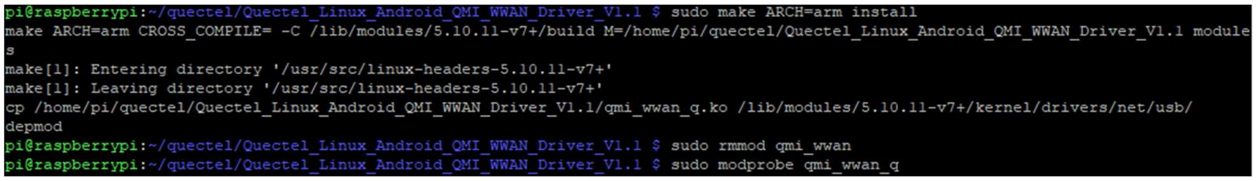
```
pi@raspberrypi:~/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 $ make ARCH=arm
make ARCH=arm CROSS_COMPILE= -C /lib/modules/5.10.11-v7+/build M=/home/pi/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 module
s
make[1]: Entering directory '/usr/src/linux-headers-5.10.11-v7+'
  CC [M]  /home/pi/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1/qmi_wwan_q.o
  MODPOST /home/pi/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1/Module.symvers
  CC [M]  /home/pi/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1/qmi_wwan_q.mod.o
  LD [M]  /home/pi/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1/qmi_wwan_q.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.10.11-v7+'
```

Installing it doesn't overwrite the original kernel driver. Let's unload the original driver and load the new one

sudo make ARCH=arm install

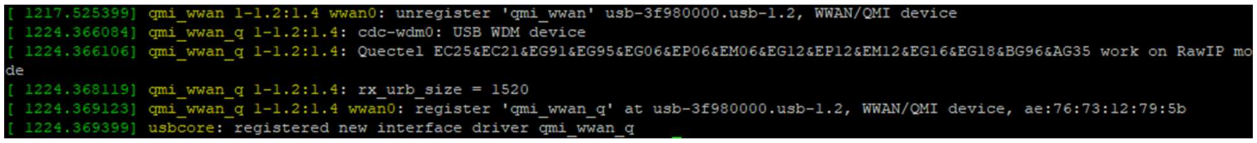
sudo rmmod qmi_wwan

sudo modprobe qmi_wwan_q



```
pi@raspberrypi:~/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 $ sudo make ARCH=arm install
make ARCH=arm CROSS_COMPILE= -C /lib/modules/5.10.11-v7+/build M=/home/pi/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 module
s
make[1]: Entering directory '/usr/src/linux-headers-5.10.11-v7+'
make[1]: Leaving directory '/usr/src/linux-headers-5.10.11-v7+'
cp /home/pi/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1/qmi_wwan_q.ko /lib/modules/5.10.11-v7+/kernel/drivers/net/usb/
depmod
pi@raspberrypi:~/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 $ sudo rmmod qmi_wwan
pi@raspberrypi:~/quectel/Quectel_Linux_Android_QMI_WWAN_Driver_V1.1 $ sudo modprobe qmi_wwan_q
```

From ***dmesg*** we can see the old driver is removed, and the new one is loaded



```
[ 1217.525399] qmi_wwan 1-1.2:1.4 wwan0: unregister 'qmi_wwan' usb-3f980000.usb-1.2, WWAN/QMI device
[ 1224.366084] qmi_wwan_q 1-1.2:1.4: cdc-wdm0: USB WDM device
[ 1224.366106] qmi_wwan_q 1-1.2:1.4: Quectel EC25&EC21&EG91&EG95&EG06&EP06&EM06&EG12&EP12&EM12&EG16&EG18&BG96&AG35 work on RawIP mo
de
[ 1224.368119] qmi_wwan_q 1-1.2:1.4: rx_urb_size = 1520
[ 1224.369123] qmi_wwan_q 1-1.2:1.4 wwan0: register 'qmi_wwan_q' at usb-3f980000.usb-1.2, WWAN/QMI device, ae:76:73:12:79:5b
[ 1224.369399] usbcore: registered new interface driver qmi_wwan_q
```

As now we have two drivers (qmi_wwan and qmi_wwan_q) handling the same hardware, it would be good to block the old one from loading.

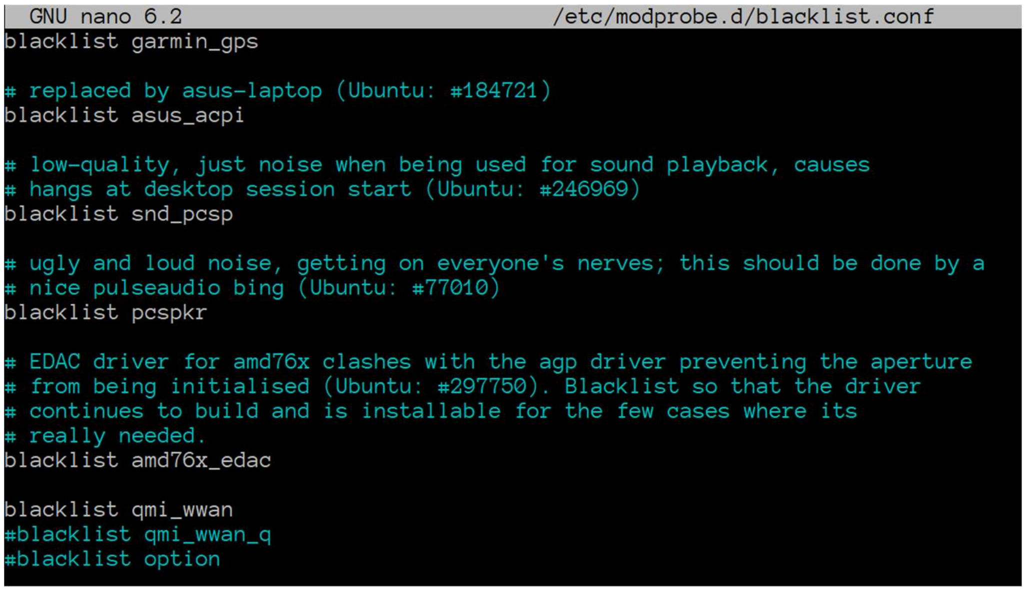
Driver blacklisting

We can put qmi_wwan or some other loadable drivers into blacklist by creating the file.

sudo nano /etc/modprobe.d/blacklist.conf

And add the name of the driver we want to disable to it, e.g.,

blacklist qmi_wwan



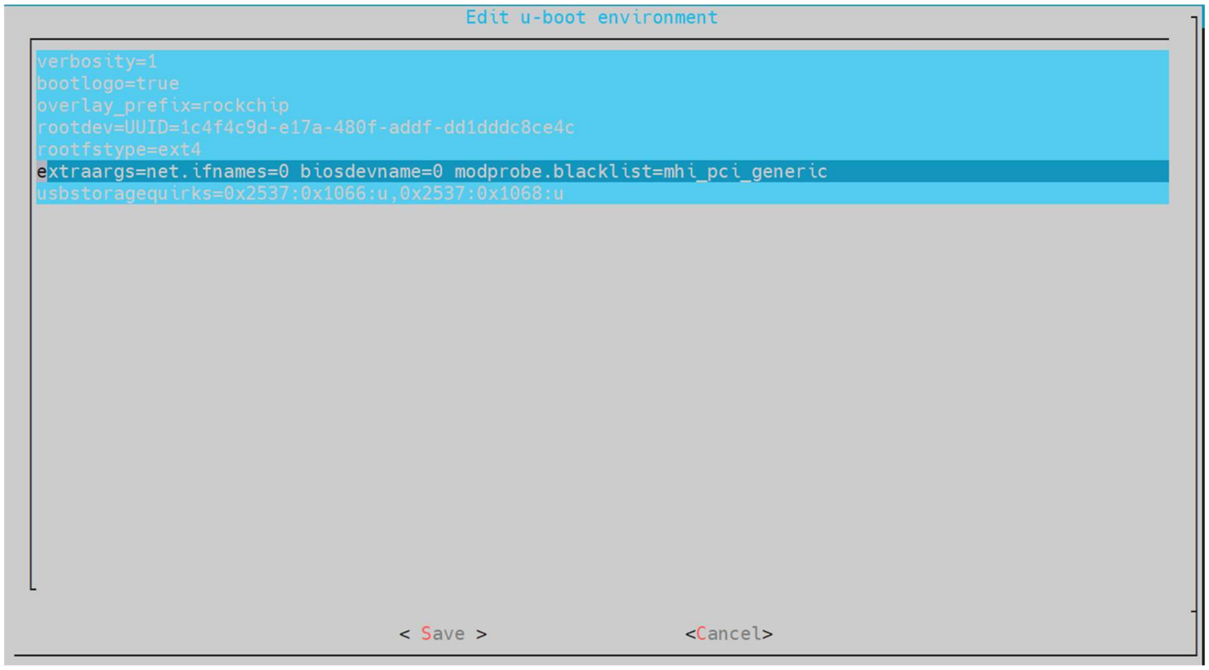
```
  GNU nano 6.2                          /etc/modprobe.d/blacklist.conf
blacklist garmin_gps

# replaced by asus-laptop (Ubuntu: #184721)
blacklist asus_acpi

# low-quality, just noise when being used for sound playback, causes
# hangs at desktop session start (Ubuntu: #246969)
blacklist snd_pcsp

# ugly and loud noise, getting on everyone's nerves; this should be done by a
# nice pulseaudio bing (Ubuntu: #77010)
blacklist pcspkr

# EDAC driver for amd76x clashes with the agp driver preventing the aperture
# from being initialised (Ubuntu: #297750). Blacklist so that the driver
# continues to build and is installable for the few cases where its
# really needed.
blacklist amd76x_edac

blacklist qmi_wwan
#blacklist qmi_wwan_q
#blacklist option
```

But for a bus driver like the mhi_pci_generic.ko will be loaded in by the system really early and the blacklist.conf will not be able to stop it. We will have to modify the kernel command line or even remove the file from the system to stop it.

Updating the kernel command line is a system dependent operation, please refer to the documentation of the Linux distribution you are using.

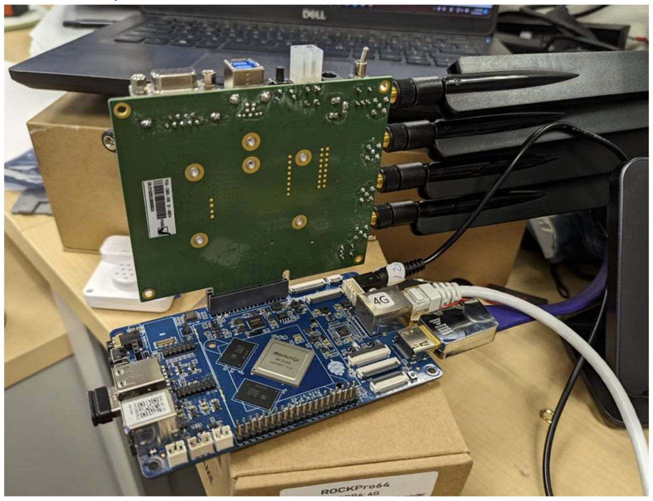
For example, with Armbian you can use the armbian-config tool, in system->Boot environment, and add this line.

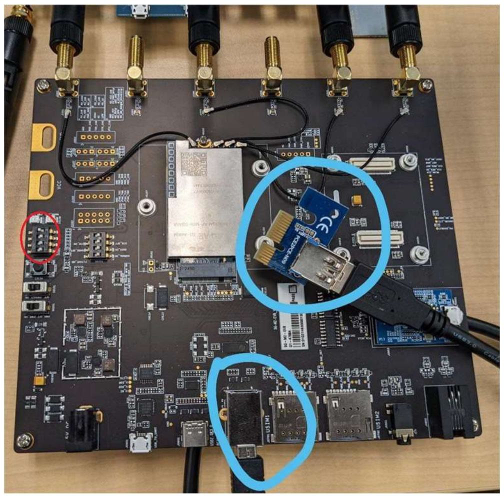
extraargs=modprobe.blacklist=mhi_pci_generic,qmi_wwan



```
                              Edit u-boot environment

verbosity=1
bootlogo=true
overlay_prefix=rockchip
rootdev=UUID=1c4f4c9d-e17a-480f-addf-dd1dddc8ce4c
rootfstype=ext4
extraargs=net.ifnames=0 biosdevname=0 modprobe.blacklist=mhi_pci_generic
usbstoragequirks=0x2537:0x1066:u,0x2537:0x1068:u
```

PCI-Express devices



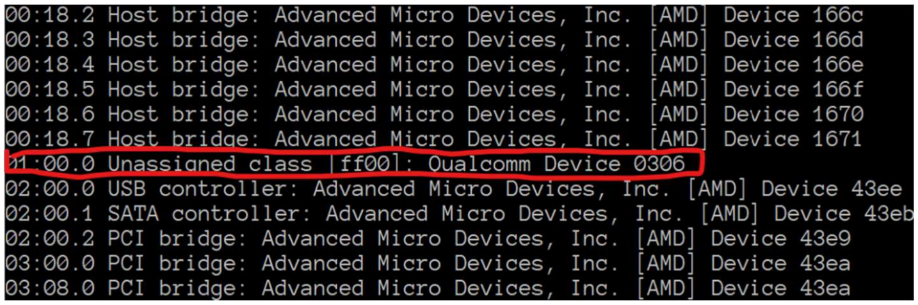


Our PCIE capable modules can come with 3 different favors.

- Switchable but defaults to USB
- Switchable but defaults to PCIE
- Fused down to PCIE only.

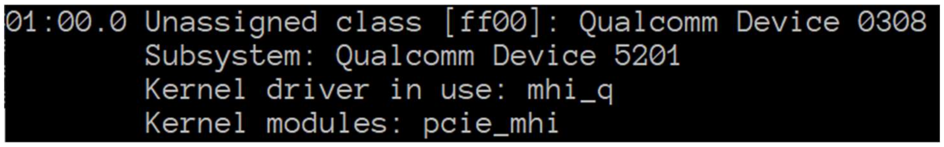
The power up sequence of a switchable module and a fused down module is different, and the compatibility is lower with a switchable ver. So, we should use a PCIE only version whenever possible.

To see if the device is detected by the system, use the command ***lspci***



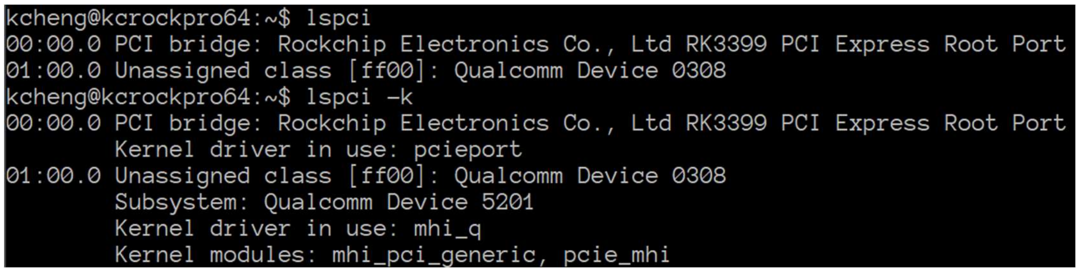
```
00:18.2 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166c
00:18.3 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166d
00:18.4 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166e
00:18.5 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 166f
00:18.6 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 1670
00:18.7 Host bridge: Advanced Micro Devices, Inc. [AMD] Device 1671
01:00.0 Unassigned class [ff00]: Qualcomm Device 0306
02:00.0 USB controller: Advanced Micro Devices, Inc. [AMD] Device 43ee
02:00.1 SATA controller: Advanced Micro Devices, Inc. [AMD] Device 43eb
02:00.2 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 43e9
03:00.0 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 43ea
03:08.0 PCI bridge: Advanced Micro Devices, Inc. [AMD] Device 43ea
```

lspci -k for more information



```
01:00.0 Unassigned class [ff00]: Qualcomm Device 0308
        Subsystem: Qualcomm Device 5201
        Kernel driver in use: mhi_q
        Kernel modules: pcie_mhi
```

Another example with a RockPro64 board https://www.pine64.org/rockpro64/



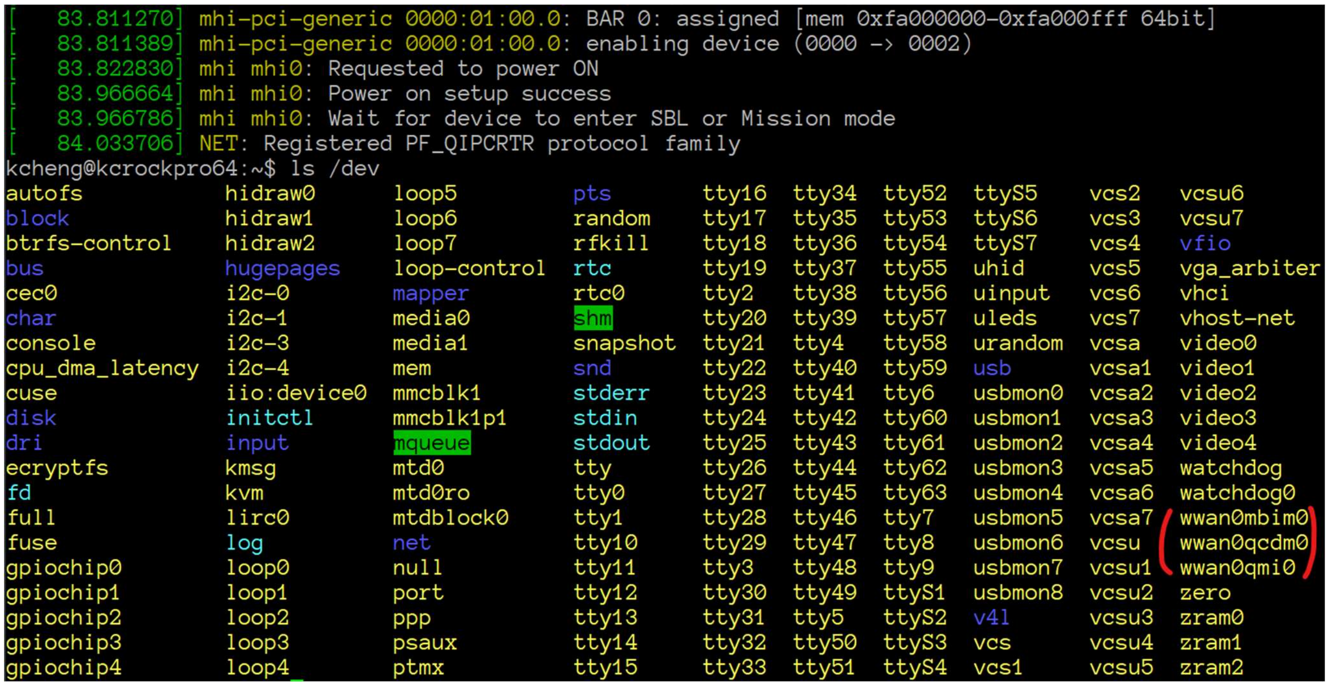
```
kcheng@kcrockpro64:~$ lspci
00:00.0 PCI bridge: Rockchip Electronics Co., Ltd RK3399 PCI Express Root Port
01:00.0 Unassigned class [ff00]: Qualcomm Device 0308
kcheng@kcrockpro64:~$ lspci -k
00:00.0 PCI bridge: Rockchip Electronics Co., Ltd RK3399 PCI Express Root Port
        Kernel driver in use: pcieport
01:00.0 Unassigned class [ff00]: Qualcomm Device 0308
        Subsystem: Qualcomm Device 5201
        Kernel driver in use: mhi_q
        Kernel modules: mhi_pci_generic, pcie_mhi
```

PCIE express drivers

MHI stands for modem host interface, and it is a Qualcomm protocol designed for cellular modules.

Newer kernels have an in-tree driver for the MHI interface named mhi_pci_generic.ko, but is not compatible with our software yet so we will disable it for now.

To stop the kernel from loading the mhi_pci_generic.ko, please refer to the previous chapter **Driver blacklisting**.

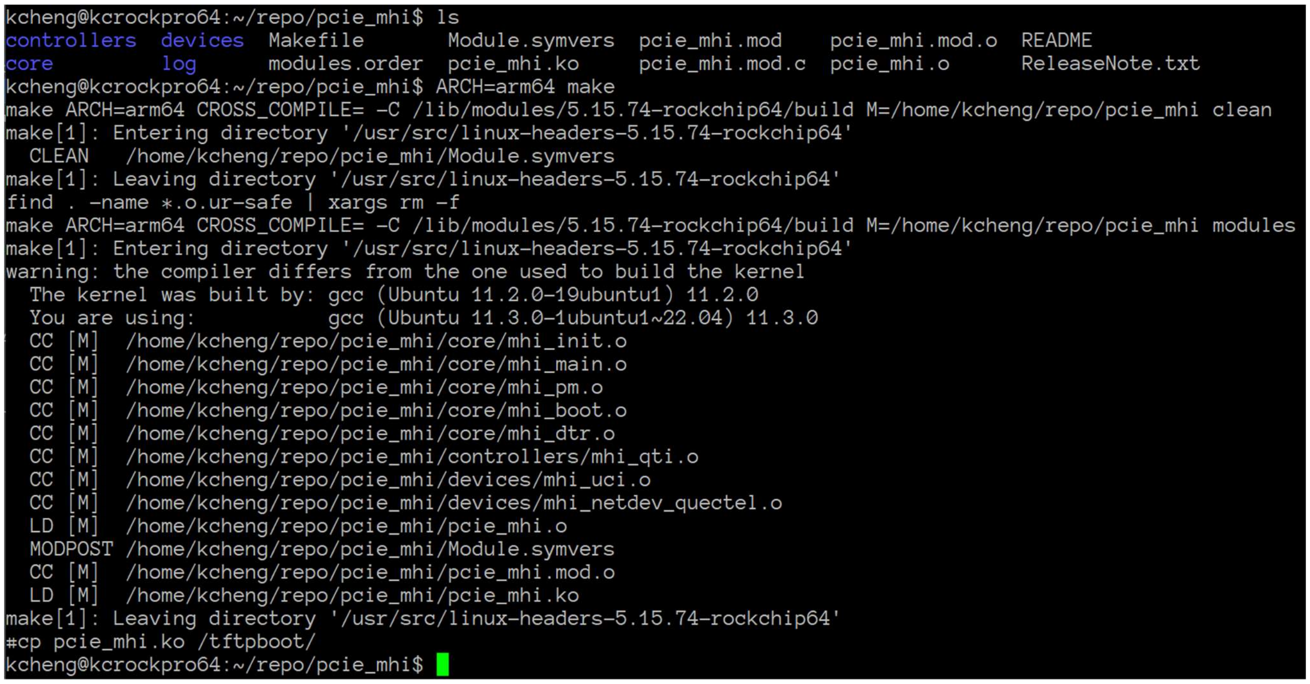


```
[   83.811270] mhi-pci-generic 0000:01:00.0: BAR 0: assigned [mem 0xfa000000-0xfa000fff 64bit]
[   83.811389] mhi-pci-generic 0000:01:00.0: enabling device (0000 -> 0002)
[   83.822830] mhi mhi0: Requested to power ON
[   83.966664] mhi mhi0: Power on setup success
[   83.966786] mhi mhi0: Wait for device to enter SBL or Mission mode
[   84.033706] NET: Registered PF_QIPCRTR protocol family
kcheng@kcrockpro64:~$ ls /dev
autofs           hidraw0      loop5         pts       tty16  tty34  tty52  ttyS5    vcs2   vcsu6
block            hidraw1      loop6         random    tty17  tty35  tty53  ttyS6    vcs3   vcsu7
btrfs-control    hidraw2      loop7         rfkill    tty18  tty36  tty54  ttyS7    vcs4   vfio
bus              hugepages    loop-control  rtc       tty19  tty37  tty55  uhid     vcs5   vga_arbiter
cec0             i2c-0        mapper        rtc0      tty2   tty38  tty56  uinput   vcs6   vhci
char             i2c-1        media0        shm       tty20  tty39  tty57  uleds    vcs7   vhost-net
console          i2c-3        media1        snapshot  tty21  tty4   tty58  urandom  vcsa   video0
cpu_dma_latency  i2c-4        mem           snd       tty22  tty40  tty59  usb      vcsa1  video1
cuse             iio:device0  mmcblk1       stderr    tty23  tty41  tty6   usbmon0  vcsa2  video2
disk             initctl      mmcblk1p1     stdin     tty24  tty42  tty60  usbmon1  vcsa3  video3
dri              input        mqueue        stdout    tty25  tty43  tty61  usbmon2  vcsa4  video4
ecryptfs         kmsg         mtd0          tty       tty26  tty44  tty62  usbmon3  vcsa5  watchdog
fd               kvm          mtd0ro        tty0      tty27  tty45  tty63  usbmon4  vcsa6  watchdog0
full             lirc0        mtdblock0     tty1      tty28  tty46  tty7   usbmon5  vcsa7  wwan0mbim0
fuse             log          net           tty10     tty29  tty47  tty8   usbmon6  vcsu   wwan0qcdm0
gpiochip0        loop0        null          tty11     tty3   tty48  tty9   usbmon7  vcsu1  wwan0qmi0
gpiochip1        loop1        port          tty12     tty30  tty49  ttyS1  usbmon8  vcsu2  zero
gpiochip2        loop2        ppp           tty13     tty31  tty5   ttyS2  v4l      vcsu3  zram0
gpiochip3        loop3        psaux         tty14     tty32  tty50  ttyS3  vcs      vcsu4  zram1
gpiochip4        loop4        ptmx          tty15     tty33  tty51  ttyS4  vcs1     vcsu5  zram2
```

We are providing our driver named pcie_mhi and we can build it the same way as the USB drivers. I am using a Arm64 device here,

ARCH=arm64 make

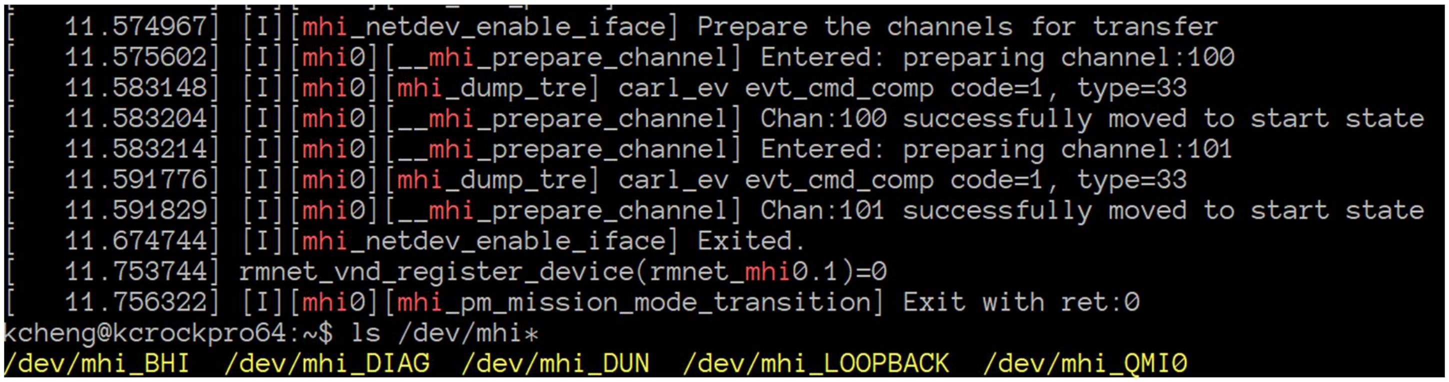
sudo ARCH=arm64 make install



```
kcheng@kcrockpro64:~/repo/pcie_mhi$ ls
controllers  devices  Makefile       Module.symvers  pcie_mhi.mod    pcie_mhi.mod.o  README
core         log      modules.order  pcie_mhi.ko     pcie_mhi.mod.c  pcie_mhi.o      ReleaseNote.txt
kcheng@kcrockpro64:~/repo/pcie_mhi$ ARCH=arm64 make
make ARCH=arm64 CROSS_COMPILE= -C /lib/modules/5.15.74-rockchip64/build M=/home/kcheng/repo/pcie_mhi clean
make[1]: Entering directory '/usr/src/linux-headers-5.15.74-rockchip64'
  CLEAN   /home/kcheng/repo/pcie_mhi/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-5.15.74-rockchip64'
find . -name *.o.ur-safe | xargs rm -f
make ARCH=arm64 CROSS_COMPILE= -C /lib/modules/5.15.74-rockchip64/build M=/home/kcheng/repo/pcie_mhi modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.74-rockchip64'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0
  You are using:           gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
  CC [M]  /home/kcheng/repo/pcie_mhi/core/mhi_init.o
  CC [M]  /home/kcheng/repo/pcie_mhi/core/mhi_main.o
  CC [M]  /home/kcheng/repo/pcie_mhi/core/mhi_pm.o
  CC [M]  /home/kcheng/repo/pcie_mhi/core/mhi_boot.o
  CC [M]  /home/kcheng/repo/pcie_mhi/core/mhi_dtr.o
  CC [M]  /home/kcheng/repo/pcie_mhi/controllers/mhi_qti.o
  CC [M]  /home/kcheng/repo/pcie_mhi/devices/mhi_uci.o
  CC [M]  /home/kcheng/repo/pcie_mhi/devices/mhi_netdev_quectel.o
  LD [M]  /home/kcheng/repo/pcie_mhi/pcie_mhi.o
  MODPOST /home/kcheng/repo/pcie_mhi/Module.symvers
  CC [M]  /home/kcheng/repo/pcie_mhi/pcie_mhi.mod.o
  LD [M]  /home/kcheng/repo/pcie_mhi/pcie_mhi.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.15.74-rockchip64'
#cp pcie_mhi.ko /tftpboot/
kcheng@kcrockpro64:~/repo/pcie_mhi$
```

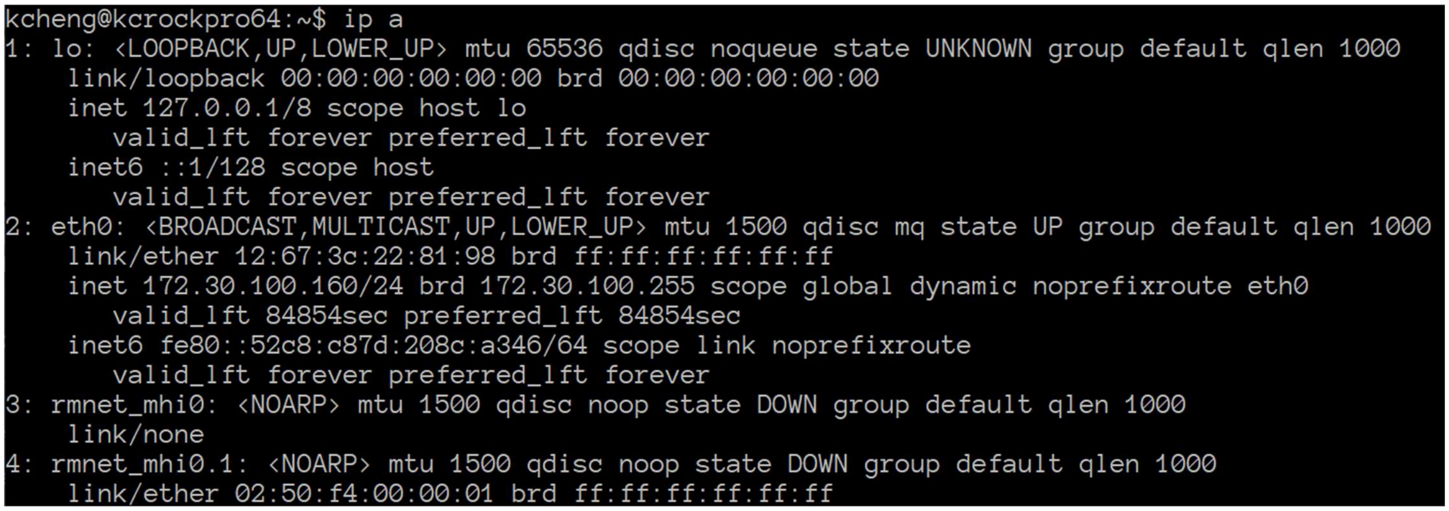
The system should load the pcie_mhi driver when it reboots. If we want to load the driver manually, use the modprobe command.

sudo modprobe pcie_mhi



```
[   11.574967] [I][mhi_netdev_enable_iface] Prepare the channels for transfer
[   11.575602] [I][mhi0][__mhi_prepare_channel] Entered: preparing channel:100
[   11.583148] [I][mhi0][mhi_dump_tre] carl_ev evt_cmd_comp code=1, type=33
[   11.583204] [I][mhi0][__mhi_prepare_channel] Chan:100 successfully moved to start state
[   11.583214] [I][mhi0][__mhi_prepare_channel] Entered: preparing channel:101
[   11.591776] [I][mhi0][mhi_dump_tre] carl_ev evt_cmd_comp code=1, type=33
[   11.591829] [I][mhi0][__mhi_prepare_channel] Chan:101 successfully moved to start state
[   11.674744] [I][mhi_netdev_enable_iface] Exited.
[   11.753744] rmnet_vnd_register_device(rmnet_mhi0.1)=0
[   11.756322] [I][mhi0][mhi_pm_mission_mode_transition] Exit with ret:0
kcheng@kcrockpro64:~$ ls /dev/mhi*
/dev/mhi_BHI  /dev/mhi_DIAG  /dev/mhi_DUN  /dev/mhi_LOOPBACK  /dev/mhi_QMI0
```

ip -a



```
kcheng@kcrockpro64:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 12:67:3c:22:81:98 brd ff:ff:ff:ff:ff:ff
    inet 172.30.100.160/24 brd 172.30.100.255 scope global dynamic noprefixroute eth0
       valid_lft 84854sec preferred_lft 84854sec
    inet6 fe80::52c8:c87d:208c:a346/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: rmnet_mhi0: <NOARP> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/none
4: rmnet_mhi0.1: <NOARP> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 02:50:f4:00:00:01 brd ff:ff:ff:ff:ff:ff
```

About the ports,

- mhi_BHI is the boot host interface. It should always be present even if the firmware is corrupted. It is used for firmware updates.
- mhi_DIAG is the diagnostic port. Both BHI and DIAG added together is pretty similar to the DM port available in USB
- mhi_DUN is the dial up networking port. i.e., AT/modem port.
- mhi_LOOPBACK is a loop back network interface for use with performance testing.
- mhi_QMI0 is the QMI command interface.
- rmnet_mhi0 and rmnet_mhi0.1 are the network interfaces.

Our data call manager quectel-CM will recognize the ports and bring up data call with them. Quectel-CM will be covered in its own chapter.

Cellular network registration with generic network carriers

Traditionally, in 2G(GSM) networks the SIM card represents the user identity, and it is all we need to register to the network. Data calls were made by dialing a specific phone number and access fee is usually calculated by call duration, as continuous network resource must be allocated for the UE (user equipment) even if there is no active data transmission.

With the introduction of 2.5G (GPRS), the network is separated into traditional CS (circuit switching) domain and a new PS (packet switching) domain. Data access fee is usually charged per KB as the UE is no longer using network resources when it is idle. A new concept of APN (access point name) replaces phone numbers for data call. But if you are not making data calls, i.e. the phone is just standing by, sending, and receiving text messages and voice calls), the APN is not needed at all. APN can be setup before you bring up the data call.

An APN setting is usually a simple string name, sometimes also with more details like proxy server settings, username, and password.

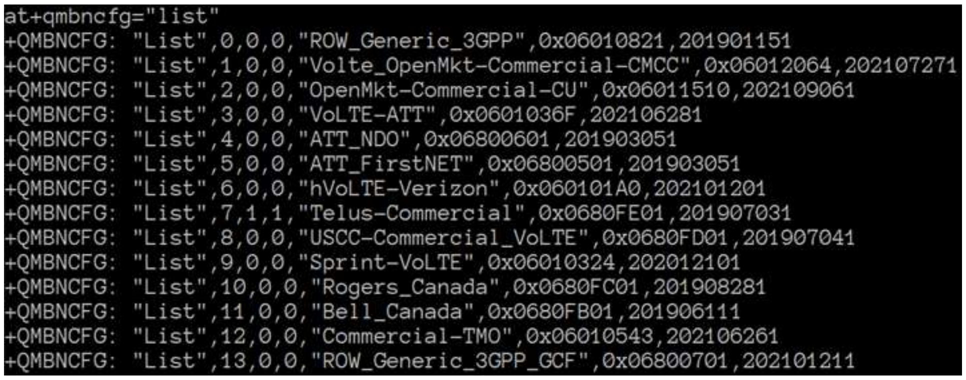
Moving forward, things didn’t change much for 3G in terms of connection setup. But starting with 4G LTE (long term evolution), the whole CS domain has been dropped. Voice calls and traditional text messages could be handled in two ways: fall back to 2G/3G, or to be handled by PS domain. i.e., VoLTE (voice over LTE), iMessage or RCS

VoLTE is a form of VOIP (voice over IP) and if it is implemented properly, we can save network resources as people don’t speak 100% nonstop during a call. Resources can be conserved if there is no actual voice communication. As the deployment of LTE is going smooth, carriers are now shutting down 3G or even 2G networks in favor of 4G and 5G networks. Frequency bands previously allocated for 2G and 3G networks are now being “refarmed” for 4G or 5G networks.

Now we have a problem. Previously we could ignore APN name until data call is made, as the module will register to 2G or 3G network even if APN is not setup properly. With the shutdown of 2G and 3G networks, we have nothing to fall back to when the APN is incorrect. We cannot even see the network carrier name and signal strength without a proper APN. The SIM alone is no longer enough to connect us to the network.

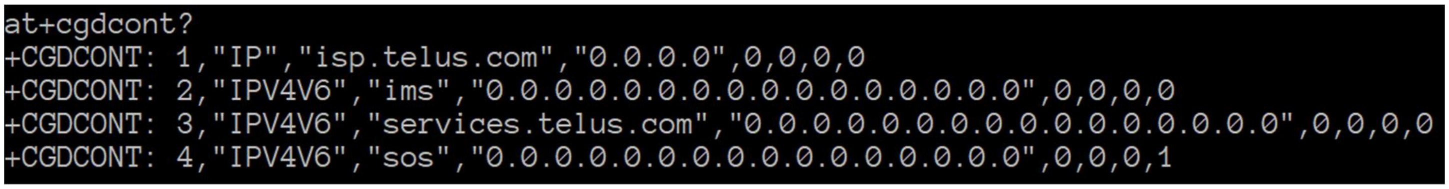
To ease the problem, Quectel modules have some carrier specific settings stored as loadable profiles (MBNs). A MBN profile has carrier specific settings, including the commonly used APN for the carrier in it. By default, when the module detects a SIM card, it will check its MCC and MNC (mobile country code and mobile network code) and load a matching MBN. A generic fall back MBN is also available when a good match is not available.

In this example, I am using a Canada Telus SIM and the 7th MBN is currently selected and activated. If you are using an unlisted carrier, the module should load the ROW_Generic profile instead.



```
at+qmbncfg="list"
+QMBNCFG: "List",0,0,0,"ROW_Generic_3GPP",0x06010821,201901151
+QMBNCFG: "List",1,0,0,"Volte_OpenMkt-Commercial-CMCC",0x06012064,202107271
+QMBNCFG: "List",2,0,0,"OpenMkt-Commercial-CU",0x06011510,202109061
+QMBNCFG: "List",3,0,0,"VoLTE-ATT",0x0601036F,202106281
+QMBNCFG: "List",4,0,0,"ATT_NDO",0x06800601,201903051
+QMBNCFG: "List",5,0,0,"ATT_FirstNET",0x06800501,201903051
+QMBNCFG: "List",6,0,0,"hVoLTE-Verizon",0x060101A0,202101201
+QMBNCFG: "List",7,1,1,"Telus-Commercial",0x0680FE01,201907031
+QMBNCFG: "List",8,0,0,"USCC-Commercial_VoLTE",0x0680FD01,201907041
+QMBNCFG: "List",9,0,0,"Sprint-VoLTE",0x06010324,202012101
+QMBNCFG: "List",10,0,0,"Rogers_Canada",0x0680FC01,201908281
+QMBNCFG: "List",11,0,0,"Bell_Canada",0x0680FB01,201906111
+QMBNCFG: "List",12,0,0,"Commercial-TMO",0x06010543,202106261
+QMBNCFG: "List",13,0,0,"ROW_Generic_3GPP_GCF",0x06800701,202101211
```

The typical APN setting for Telus’s end user retail SIM is preloaded.



```
at+cgdcont?
+CGDCONT: 1,"IP","isp.telus.com","0.0.0.0",0,0,0,0
+CGDCONT: 2,"IPV4V6","ims","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 3,"IPV4V6","services.telus.com","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 4,"IPV4V6","sos","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1
```

The 1st entry isp.telus.com is the generic APN for Telus for retail customers. If you are using a business SIM account, it is very likely that you need a special APN. Please contact your network carrier for details.

To update the APN, use this AT command.

AT+CGDONCT=1,"IPV4V6","apn-name-to-use"

Usually, the default IPV4V6 is fine, you may want to specify IP, IPV6, or IPV4V6 depending on your network design.

Some carriers accept a blank APN name, you may try an empty string "". But it is still possible that IP traffic will fail even if network registration works.

Manually restart the radio part of the module for the APN change to take effect.

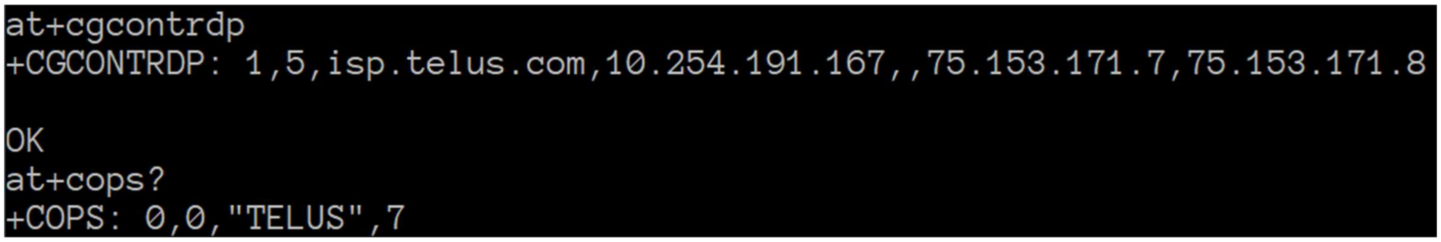
AT+CFUN=0

AT+CFUN=1

The 2nd entry IMS is the PDP for VoLTE. The 4th entry SOS is the PDP for emergency call. These APNs have some additional settings inside and could not be created/modified using the usual AT+CGDCONT command. If it has been overwritten, the best way to restore them is to reload the MBN which will be covered in the next chapter.

In this example, the isp.telus.com APN has been activated and an IP V4 address is assigned to the module.

AT+COPS? Returns the correct carrier's name Telus, and technology is 7 i.e. LTE



```
at+cgcontrdp
+CGCONTRDP: 1,5,isp.telus.com,10.254.191.167,,75.153.171.7,75.153.171.8

OK
at+cops?
+COPS: 0,0,"TELUS",7
```

Reload APN settings from default

To reload default APN settings from the MBN, you may try to swap the SIM with another SIM from another carrier to trigger a MBN reload. If SIM hot plug is enabled, it is done automatically. If SIM hot plug is not setup, you can power cycle the module or trigger a radio restart by AT+CFUN=0 followed by AT+CFUN=1. Now put back the SIM you want to use (and power cycle/restart radio) to trigger a reload again. After a new SIM is detected, the module should load the other MBN profile.

The auto MBN loading setting can be turned off by

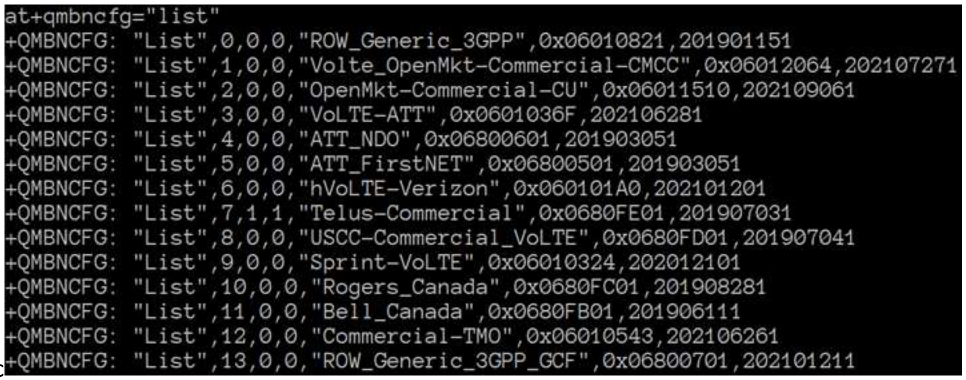
AT+QMBNCFG="autosel",0

Turn it back on by

AT+QMBNCFG="autosel",1

To reload a MBN manually, you can look up the name of the carrier MBN from the list with this AT command,

AT+QMBNCFG="list"



```
at+qmbncfg="list"
+QMBNCFG: "List",0,0,0,"ROW_Generic_3GPP",0x06010821,201901151
+QMBNCFG: "List",1,0,0,"Volte_OpenMkt-Commercial-CMCC",0x06012064,202107271
+QMBNCFG: "List",2,0,0,"OpenMkt-Commercial-CU",0x06011510,202109061
+QMBNCFG: "List",3,0,0,"VoLTE-ATT",0x0601036F,202106281
+QMBNCFG: "List",4,0,0,"ATT_NDO",0x06800601,201903051
+QMBNCFG: "List",5,0,0,"ATT_FirstNET",0x06800501,201903051
+QMBNCFG: "List",6,0,0,"hVoLTE-Verizon",0x060101A0,202101201
+QMBNCFG: "List",7,1,1,"Telus-Commercial",0x0680FE01,201907031
+QMBNCFG: "List",8,0,0,"USCC-Commercial_VoLTE",0x0680FD01,201907041
+QMBNCFG: "List",9,0,0,"Sprint-VoLTE",0x06010324,202012101
+QMBNCFG: "List",10,0,0,"Rogers_Canada",0x0680FC01,201908281
+QMBNCFG: "List",11,0,0,"Bell_Canada",0x0680FB01,201906111
+QMBNCFG: "List",12,0,0,"Commercial-TMO",0x06010543,202106261
+QMBNCFG: "List",13,0,0,"ROW_Generic_3GPP_GCF",0x06800701,202101211
```

c

For some modules, an additional command is needed to de-select the current MBN before selecting (another) one

at+qmbncfg="Deactivate"

Select it again with this command,

AT+QMBNCFG="select","name from the previous command"

e.g.

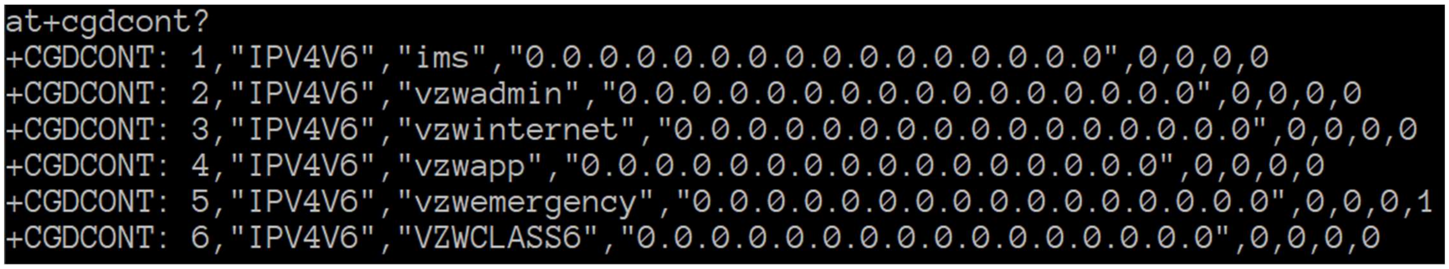
AT+QMBNCFG="select","hVoLTE-Verizon"

And reboot the module for it to take effect. Simple radio restart won't work in this case.

AT+CFUN=1,1

Network registration with Verizon

Verizon has some special requirements with network registration, compared with most other carriers.



```
at+cgdcont?
+CGDCONT: 1,"IPV4V6","ims","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 2,"IPV4V6","vzwadmin","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 3,"IPV4V6","vzwinternet","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 4,"IPV4V6","vzwapp","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 5,"IPV4V6","vzwemergency","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1
+CGDCONT: 6,"IPV4V6","VZWCLASS6","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
```

- For LPWA modules, it’s pretty much the same as other carriers. You can setup the APN in PDP#1
- For LTE and 5G networks, the APN of PDP#1 must be IMS and it is used for initial network attach.
- PDP#2 is used for remote administration, with the OMA-DM or LWM2M protocols.
- PDP#3 is used for data call, it can also be setup by OMA-DM/LWM2M. When you insert a new SIM to the module, the module will talk with the Verizon administration server using the vzwadmin PDP and the correct APN will be set. After that you can setup manually and Verizon won’t push a new APN to the module again unless you swap a SIM.
- If APNs have been accidentally changed manually, the module won’t register to the network, or strange things can happen. Please reload the default in this case.

Data call with quectel-CM

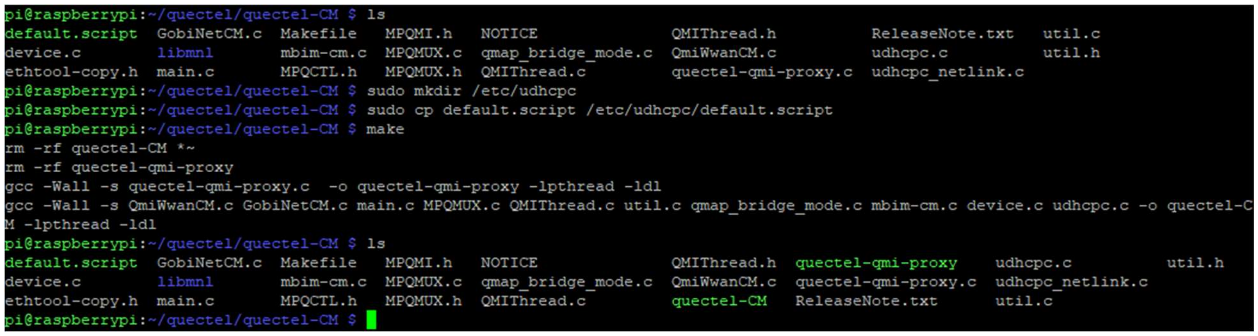
Finally, we can build the quectel-CM connection manager. It is provided with source code, and you may want to customize it to suit your application if needed. It supports both QMI and MBIM mode. We are using QMI mode as an example here.

Unzip the quectel-CM package. Note that quectel-CM zip package has no top-level folder, let's make a folder for it before decompressing. e.g.
mkdir quectel-CM-1.6.1
cd quectel-CM-1.6.1
unzip ../Quectel_QConnectManager_Linux_V1.6.1_20210720.zip

Copy our sample DHCP script (default.script or default.script_ip) to /etc/udhcpc/default.script. It is a helper script used by udhcpc to setup system's DNS and route. You may want to customize it if you have special needs. Please also enable execution right for it.
sudo mkdir /etc/udhcpc
sudo cp default.script /etc/udhcpc/default.script
sudo chmod a+x /etc/udhcpc/default.script

To compile the source code into Linux executable,

make



```
pi@raspberrypi:~/quectel/quectel-CM $ ls
default.script  GobiNetCM.c  Makefile   MPQMI.h   NOTICE               QMIThread.h          ReleaseNote.txt   util.c
device.c        libmnl       mbim-cm.c  MPQMUX.c  qmap_bridge_mode.c   QmiWwanCM.c          udhcpc.c          util.h
ethtool-copy.h  main.c       MPQCTL.h   MPQMUX.h  QMIThread.c          quectel-qmi-proxy.c  udhcpc_netlink.c
pi@raspberrypi:~/quectel/quectel-CM $ sudo mkdir /etc/udhcpc
pi@raspberrypi:~/quectel/quectel-CM $ sudo cp default.script /etc/udhcpc/default.script
pi@raspberrypi:~/quectel/quectel-CM $ make
rm -rf quectel-CM *~
rm -rf quectel-qmi-proxy
gcc -Wall -s quectel-qmi-proxy.c  -o quectel-qmi-proxy -lpthread -ldl
gcc -Wall -s QmiWwanCM.c GobiNetCM.c main.c MPQMUX.c QMIThread.c util.c qmap_bridge_mode.c mbim-cm.c device.c udhcpc.c -o quectel-C
M -lpthread -ldl
pi@raspberrypi:~/quectel/quectel-CM $ ls
default.script  GobiNetCM.c  Makefile   MPQMI.h   NOTICE               QMIThread.h  quectel-qmi-proxy    udhcpc.c          util.h
device.c        libmnl       mbim-cm.c  MPQMUX.c  qmap_bridge_mode.c   QmiWwanCM.c  quectel-qmi-proxy.c  udhcpc_netlink.c
ethtool-copy.h  main.c       MPQCTL.h   MPQMUX.h  QMIThread.c          quectel-CM   ReleaseNote.txt      util.c
pi@raspberrypi:~/quectel/quectel-CM $
```

We can execute quectel-CM with command ***sudo ./quectel-CM*** You may want to add **-n 3** if you are using a VoLTE capable module on Verizon. The correct data call APN is in PDP 3 for such cases.

You may want to use command line parameter **-s <APN name>** to specify a customer APN. **Not recommended for QMI mode unless you are making multiple APN calls. And as of today, quectel-CM is not able to detect the correct APN name in MBIM mode** and it is needed to use this command line parameter.

From the output of quectel-CM, you can see useful information like module FW version, SIM status, APN name, and current network information.

Before dialing the **IPv4ConnectionStatus** is **DISCONNECTED**, and after dialing we have a **WdsConnectionIPv4Handle: XXXXXXXX** it means that the data call is connected.

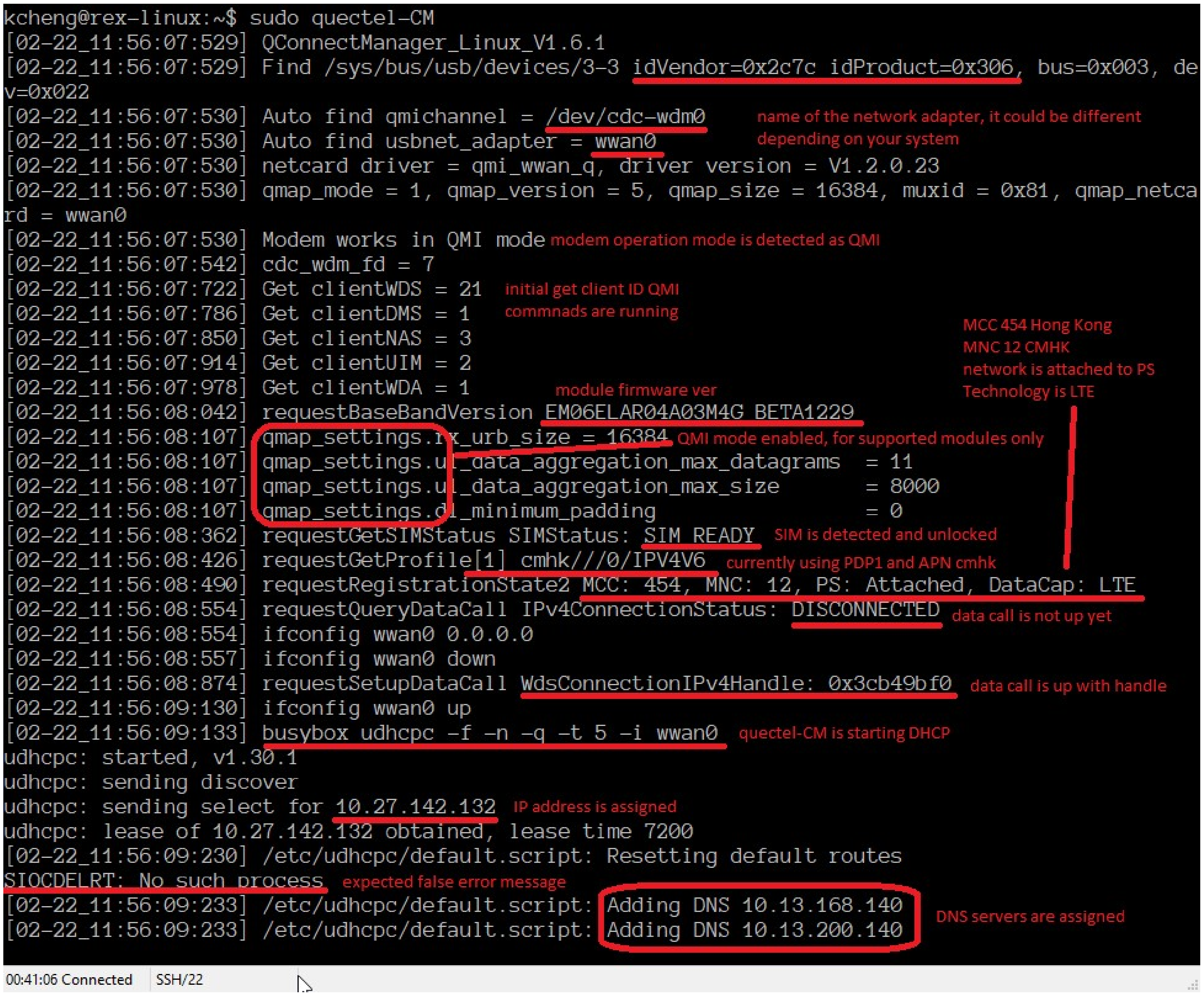
Depending on the module you use, quectel-CM may config the network interface by calling udhcpc, or it may issue "ip" shell commands to configure the network adapter.

The error message "Fail to access /usr/share/udhcpc/default.script, errno: 2 (No such file or directory)" may not be valid for your system. Older versions of udhcpc are using this path and quectel-CM is checking if this file is there, but newer udhcpc is using **/etc/udhcpc/default.script** instead.

The last 4 lines come from default.script and it is setting up route and DNS. The error message SIOCDELRT: No such process is also harmless. It is also expected.

You may see an error message "**ioctl (0x89f3, qmap_settings) failed: Operation not supported, rc=-1**" if the module does not support the newer QMAP mode. It can be ignored, and it is expected.

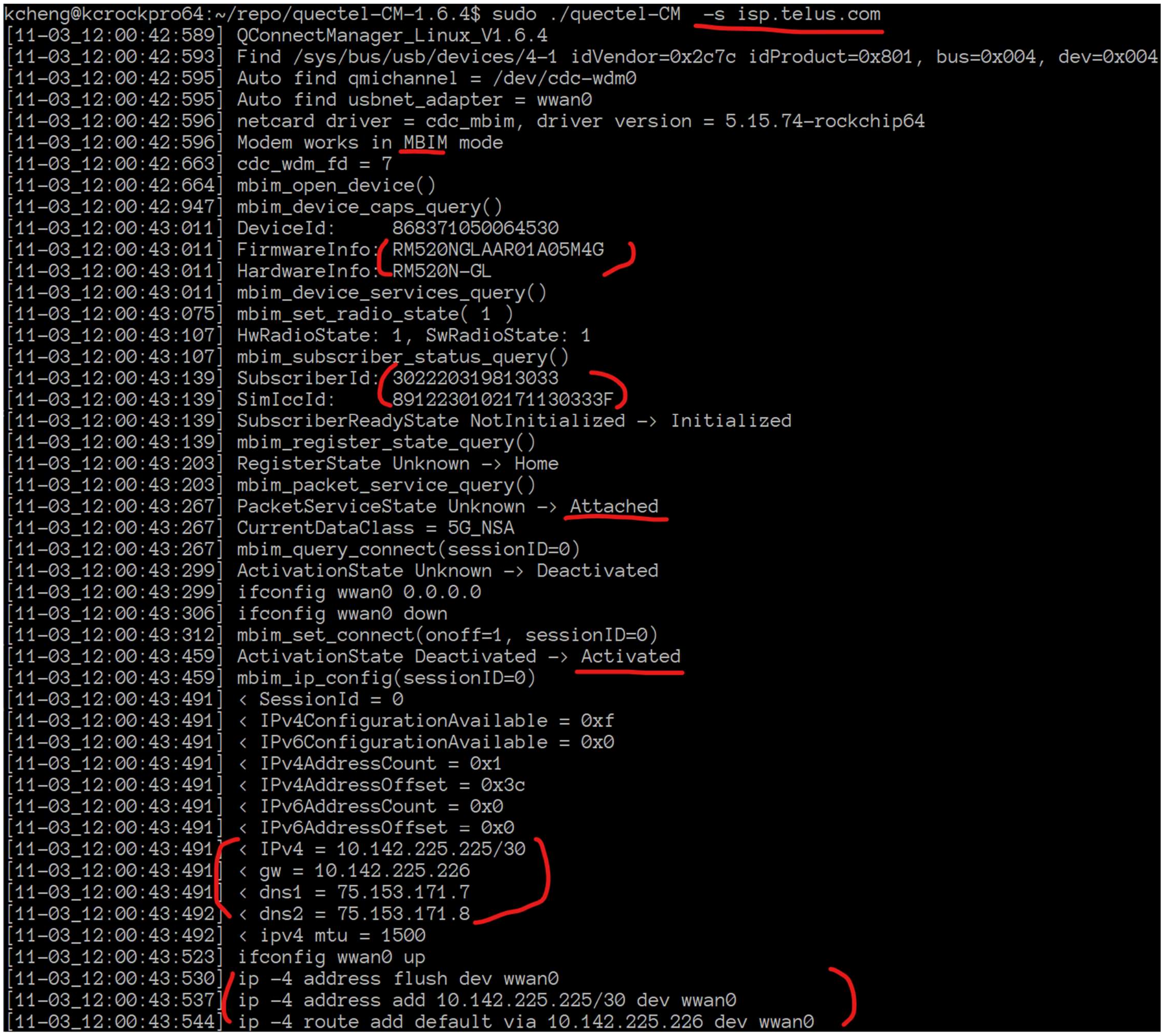
Example call log in QMI mode. We can see quectel-CM is calling udhcpc to set up the network interface.



```
kcheng@rex-linux:~$ sudo quectel-CM
[02-22_11:56:07:529] QConnectManager_Linux_V1.6.1
[02-22_11:56:07:529] Find /sys/bus/usb/devices/3-3 idVendor=0x2c7c idProduct=0x306, bus=0x003, de
v=0x022
[02-22_11:56:07:530] Auto find qmichannel = /dev/cdc-wdm0
[02-22_11:56:07:530] Auto find usbnet_adapter = wwan0
[02-22_11:56:07:530] netcard driver = qmi_wwan_q, driver version = V1.2.0.23
[02-22_11:56:07:530] qmap_mode = 1, qmap_version = 5, qmap_size = 16384, muxid = 0x81, qmap_netca
rd = wwan0
[02-22_11:56:07:530] Modem works in QMI mode
[02-22_11:56:07:542] cdc_wdm_fd = 7
[02-22_11:56:07:722] Get clientWDS = 21
[02-22_11:56:07:786] Get clientDMS = 1
[02-22_11:56:07:850] Get clientNAS = 3
[02-22_11:56:07:914] Get clientUIM = 2
[02-22_11:56:07:978] Get clientWDA = 1
[02-22_11:56:08:042] requestBaseBandVersion EM06ELAR04A03M4G_BETA1229
[02-22_11:56:08:107] qmap_settings.rx_urb_size = 16384
[02-22_11:56:08:107] qmap_settings.ul_data_aggregation_max_datagrams  = 11
[02-22_11:56:08:107] qmap_settings.ul_data_aggregation_max_size       = 8000
[02-22_11:56:08:107] qmap_settings.dl_minimum_padding                 = 0
[02-22_11:56:08:362] requestGetSIMStatus SIMStatus: SIM_READY
[02-22_11:56:08:426] requestGetProfile[1] cmhk///0/IPV4V6
[02-22_11:56:08:490] requestRegistrationState2 MCC: 454, MNC: 12, PS: Attached, DataCap: LTE
[02-22_11:56:08:554] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[02-22_11:56:08:554] ifconfig wwan0 0.0.0.0
[02-22_11:56:08:557] ifconfig wwan0 down
[02-22_11:56:08:874] requestSetupDataCall WdsConnectionIPv4Handle: 0x3cb49bf0
[02-22_11:56:09:130] ifconfig wwan0 up
[02-22_11:56:09:133] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.30.1
udhcpc: sending discover
udhcpc: sending select for 10.27.142.132
udhcpc: lease of 10.27.142.132 obtained, lease time 7200
[02-22_11:56:09:230] /etc/udhcpc/default.script: Resetting default routes
SIOCDELRT: No such process
[02-22_11:56:09:233] /etc/udhcpc/default.script: Adding DNS 10.13.168.140
[02-22_11:56:09:233] /etc/udhcpc/default.script: Adding DNS 10.13.200.140
```

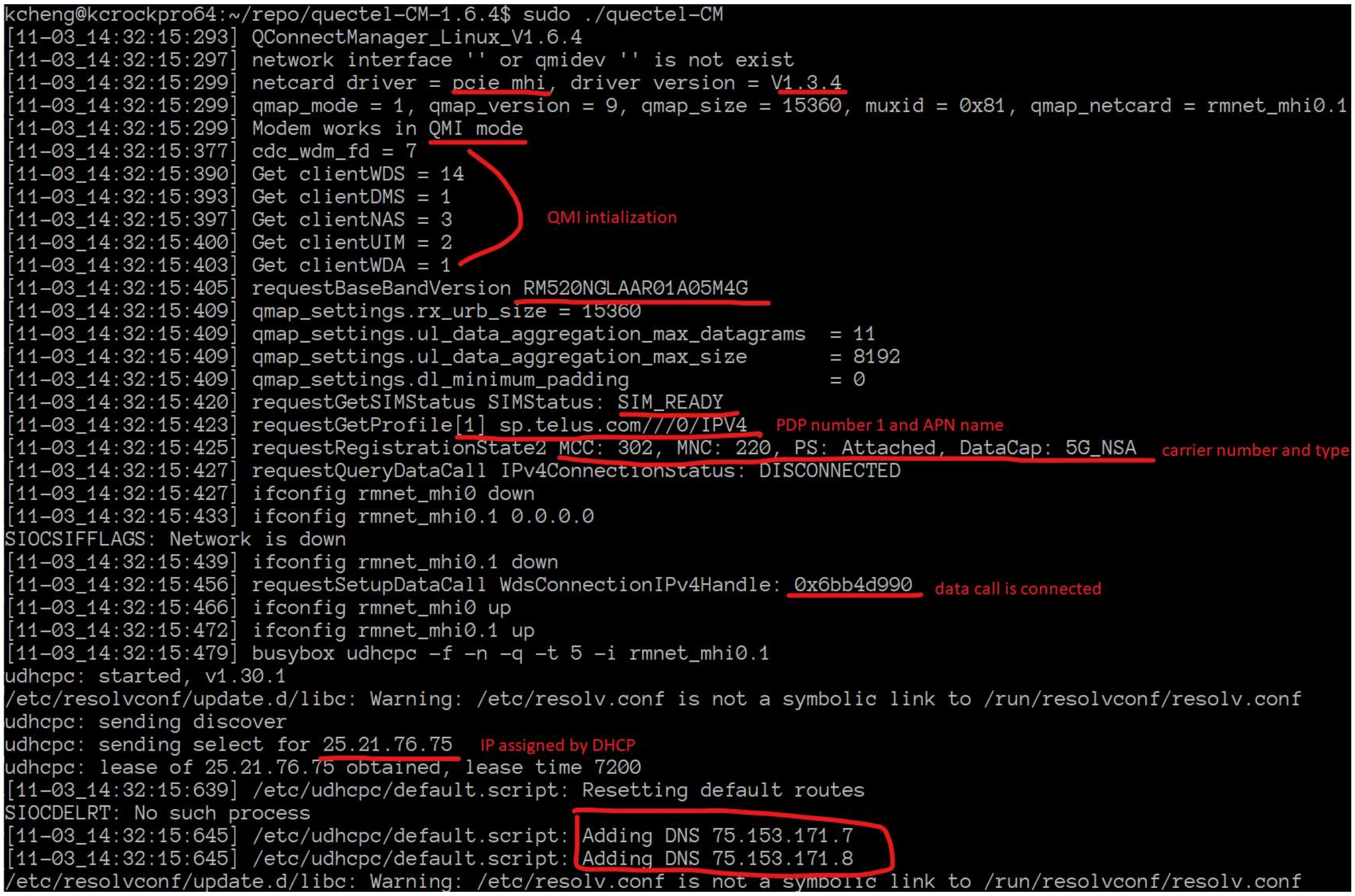
name of the network adapter, it could be different depending on your system
modem operation mode is detected as QMI
initial get client ID QMI commnads are running
MCC 454 Hong Kong
MNC 12 CMHK
network is attached to PS
Technology is LTE
module firmware ver
QMI mode enabled, for supported modules only
SIM is detected and unlocked
currently using PDP1 and APN cmhk
data call is not up yet
data call is up with handle
quectel-CM is starting DHCP
IP address is assigned
expected false error message
DNS servers are assigned
00:41:06 Connected
SSH/22

Example call log in MBIM mode. In this case we can see quectel-CM is calling “ip” command line tool to setup the network interface.



```
kcheng@kcrockpro64:~/repo/quectel-CM-1.6.4$ sudo ./quectel-CM  -s isp.telus.com
[11-03_12:00:42:589] QConnectManager_Linux_V1.6.4
[11-03_12:00:42:593] Find /sys/bus/usb/devices/4-1 idVendor=0x2c7c idProduct=0x801, bus=0x004, dev=0x004
[11-03_12:00:42:595] Auto find qmichannel = /dev/cdc-wdm0
[11-03_12:00:42:595] Auto find usbnet_adapter = wwan0
[11-03_12:00:42:596] netcard driver = cdc_mbim, driver version = 5.15.74-rockchip64
[11-03_12:00:42:596] Modem works in MBIM mode
[11-03_12:00:42:663] cdc_wdm_fd = 7
[11-03_12:00:42:664] mbim_open_device()
[11-03_12:00:42:947] mbim_device_caps_query()
[11-03_12:00:43:011] DeviceId:     868371050064530
[11-03_12:00:43:011] FirmwareInfo: RM520NGLAAR01A05M4G
[11-03_12:00:43:011] HardwareInfo: RM520N-GL
[11-03_12:00:43:011] mbim_device_services_query()
[11-03_12:00:43:075] mbim_set_radio_state( 1 )
[11-03_12:00:43:107] HwRadioState: 1, SwRadioState: 1
[11-03_12:00:43:107] mbim_subscriber_status_query()
[11-03_12:00:43:139] SubscriberId: 302220319813033
[11-03_12:00:43:139] SimIccId:     8912230102171130333F
[11-03_12:00:43:139] SubscriberReadyState NotInitialized -> Initialized
[11-03_12:00:43:139] mbim_register_state_query()
[11-03_12:00:43:203] RegisterState Unknown -> Home
[11-03_12:00:43:203] mbim_packet_service_query()
[11-03_12:00:43:267] PacketServiceState Unknown -> Attached
[11-03_12:00:43:267] CurrentDataClass = 5G_NSA
[11-03_12:00:43:267] mbim_query_connect(sessionID=0)
[11-03_12:00:43:299] ActivationState Unknown -> Deactivated
[11-03_12:00:43:299] ifconfig wwan0 0.0.0.0
[11-03_12:00:43:306] ifconfig wwan0 down
[11-03_12:00:43:312] mbim_set_connect(onoff=1, sessionID=0)
[11-03_12:00:43:459] ActivationState Deactivated -> Activated
[11-03_12:00:43:459] mbim_ip_config(sessionID=0)
[11-03_12:00:43:491] < SessionId = 0
[11-03_12:00:43:491] < IPv4ConfigurationAvailable = 0xf
[11-03_12:00:43:491] < IPv6ConfigurationAvailable = 0x0
[11-03_12:00:43:491] < IPv4AddressCount = 0x1
[11-03_12:00:43:491] < IPv4AddressOffset = 0x3c
[11-03_12:00:43:491] < IPv6AddressCount = 0x0
[11-03_12:00:43:491] < IPv6AddressOffset = 0x0
[11-03_12:00:43:491] < IPv4 = 10.142.225.225/30
[11-03_12:00:43:491] < gw = 10.142.225.226
[11-03_12:00:43:491] < dns1 = 75.153.171.7
[11-03_12:00:43:492] < dns2 = 75.153.171.8
[11-03_12:00:43:492] < ipv4 mtu = 1500
[11-03_12:00:43:523] ifconfig wwan0 up
[11-03_12:00:43:530] ip -4 address flush dev wwan0
[11-03_12:00:43:537] ip -4 address add 10.142.225.225/30 dev wwan0
[11-03_12:00:43:544] ip -4 route add default via 10.142.225.226 dev wwan0
```

Example call log with PCIE

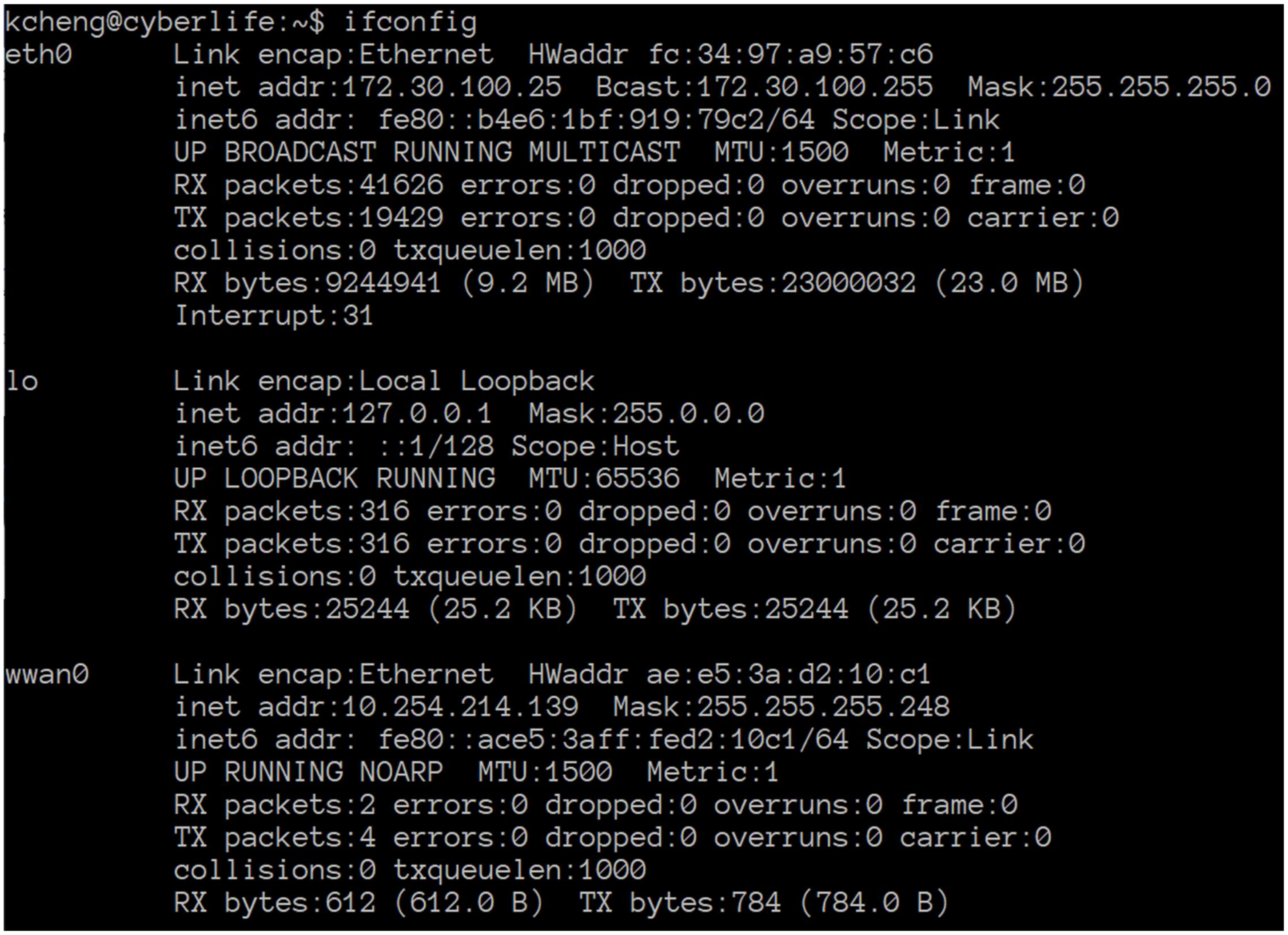


```
kcheng@kcrockpro64:~/repo/quectel-CM-1.6.4$ sudo ./quectel-CM
[11-03_14:32:15:293] QConnectManager_Linux_V1.6.4
[11-03_14:32:15:297] network interface '' or qmidev '' is not exist
[11-03_14:32:15:299] netcard driver = pcie_mhi, driver version = V1.3.4
[11-03_14:32:15:299] qmap_mode = 1, qmap_version = 9, qmap_size = 15360, muxid = 0x81, qmap_netcard = rmnet_mhi0.1
[11-03_14:32:15:299] Modem works in QMI mode
[11-03_14:32:15:377] cdc_wdm_fd = 7
[11-03_14:32:15:390] Get clientWDS = 14
[11-03_14:32:15:393] Get clientDMS = 1
[11-03_14:32:15:397] Get clientNAS = 3
[11-03_14:32:15:400] Get clientUIM = 2
[11-03_14:32:15:403] Get clientWDA = 1
[11-03_14:32:15:405] requestBaseBandVersion RM520NGLAAR01A05M4G
[11-03_14:32:15:409] qmap_settings.rx_urb_size = 15360
[11-03_14:32:15:409] qmap_settings.ul_data_aggregation_max_datagrams  = 11
[11-03_14:32:15:409] qmap_settings.ul_data_aggregation_max_size       = 8192
[11-03_14:32:15:409] qmap_settings.dl_minimum_padding                 = 0
[11-03_14:32:15:420] requestGetSIMStatus SIMStatus: SIM_READY
[11-03_14:32:15:423] requestGetProfile[1] sp.telus.com///0/IPV4
[11-03_14:32:15:425] requestRegistrationState2 MCC: 302, MNC: 220, PS: Attached, DataCap: 5G_NSA
[11-03_14:32:15:427] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[11-03_14:32:15:427] ifconfig rmnet_mhi0 down
[11-03_14:32:15:433] ifconfig rmnet_mhi0.1 0.0.0.0
SIOCSIFFLAGS: Network is down
[11-03_14:32:15:439] ifconfig rmnet_mhi0.1 down
[11-03_14:32:15:456] requestSetupDataCall WdsConnectionIPv4Handle: 0x6bb4d990
[11-03_14:32:15:466] ifconfig rmnet_mhi0 up
[11-03_14:32:15:472] ifconfig rmnet_mhi0.1 up
[11-03_14:32:15:479] busybox udhcpc -f -n -q -t 5 -i rmnet_mhi0.1
udhcpc: started, v1.30.1
/etc/resolvconf/update.d/libc: Warning: /etc/resolv.conf is not a symbolic link to /run/resolvconf/resolv.conf
udhcpc: sending discover
udhcpc: sending select for 25.21.76.75
udhcpc: lease of 25.21.76.75 obtained, lease time 7200
[11-03_14:32:15:639] /etc/udhcpc/default.script: Resetting default routes
SIOCDELRT: No such process
[11-03_14:32:15:645] /etc/udhcpc/default.script: Adding DNS 75.153.171.7
[11-03_14:32:15:645] /etc/udhcpc/default.script: Adding DNS 75.153.171.8
/etc/resolvconf/update.d/libc: Warning: /etc/resolv.conf is not a symbolic link to /run/resolvconf/resolv.conf
```

After the data call is connected, we can confirm the data call is up by these commands

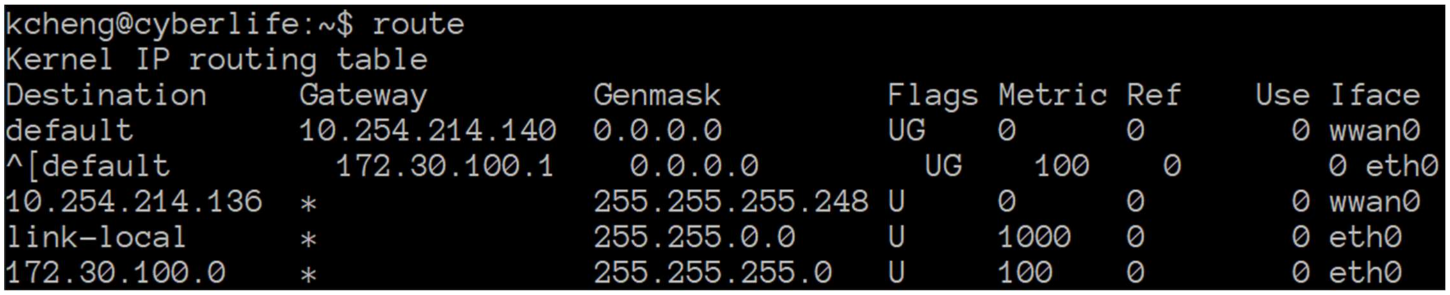
ifconfig
Or
ip a

You may note that in this example the names of the network interfaces are traditional simple names, but you may see something like ens33 on newer systems. It is a new behavior introduced by systemd.
If you prefer old style names like I do, please refer to this link.
https://www.itzgeek.com/how-tos/mini-howtos/change-default-network-name-ens33-to-old-eth0-on-ubuntu-16-04.html



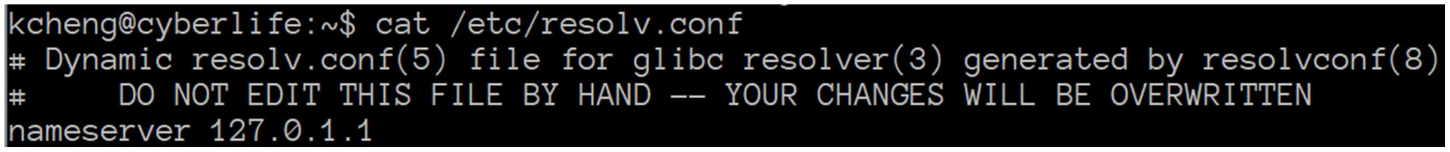
```
kcheng@cyberlife:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr fc:34:97:a9:57:c6
          inet addr:172.30.100.25  Bcast:172.30.100.255  Mask:255.255.255.0
          inet6 addr: fe80::b4e6:1bf:919:79c2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:41626 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19429 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9244941 (9.2 MB)  TX bytes:23000032 (23.0 MB)
          Interrupt:31

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:316 errors:0 dropped:0 overruns:0 frame:0
          TX packets:316 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:25244 (25.2 KB)  TX bytes:25244 (25.2 KB)

wwan0     Link encap:Ethernet  HWaddr ae:e5:3a:d2:10:c1
          inet addr:10.254.214.139  Mask:255.255.255.248
          inet6 addr: fe80::ace5:3aff:fed2:10c1/64 Scope:Link
          UP RUNNING NOARP  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:612 (612.0 B)  TX bytes:784 (784.0 B)
```

route -n



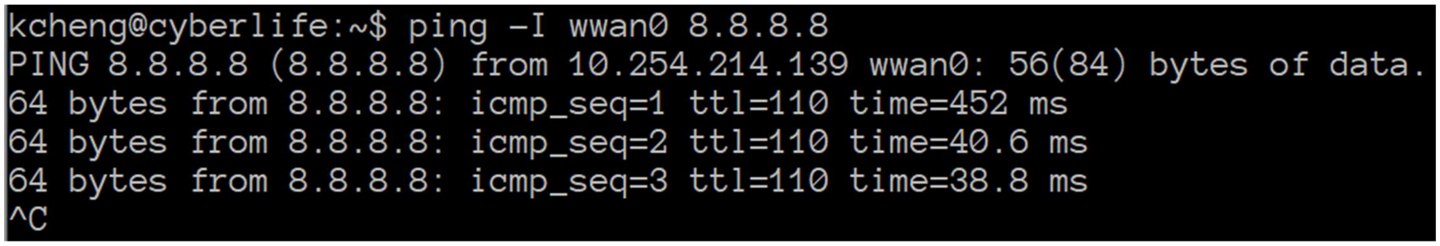
```
kcheng@cyberlife:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.254.214.140  0.0.0.0         UG    0      0        0 wwan0
^[default         172.30.100.1    0.0.0.0         UG    100    0        0 eth0
10.254.214.136  *               255.255.255.248 U     0      0        0 wwan0
link-local      *               255.255.0.0     U     1000   0        0 eth0
172.30.100.0    *               255.255.255.0   U     100    0        0 eth0
```

cat /etc/resolv.conf



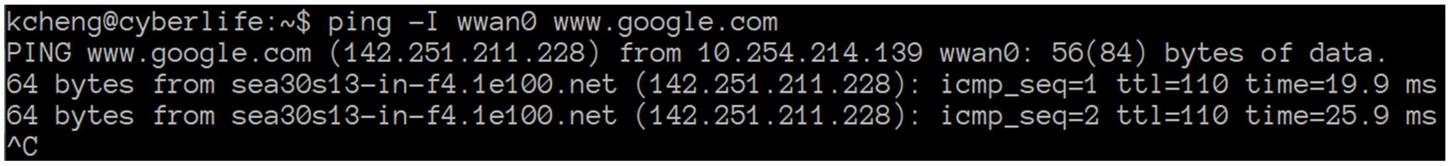
```
kcheng@cyberlife:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.1.1
```

Ping with IP



```
kcheng@cyberlife:~$ ping -I wwan0 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 10.254.214.139 wwan0: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=110 time=452 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=110 time=40.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=110 time=38.8 ms
^C
```

Ping with domain name



```
kcheng@cyberlife:~$ ping -I wwan0 www.google.com
PING www.google.com (142.251.211.228) from 10.254.214.139 wwan0: 56(84) bytes of data.
64 bytes from sea30s13-in-f4.1e100.net (142.251.211.228): icmp_seq=1 ttl=110 time=19.9 ms
64 bytes from sea30s13-in-f4.1e100.net (142.251.211.228): icmp_seq=2 ttl=110 time=25.9 ms
^C
```

PPP call over serial ports

Point to point protocol (PPP) is an ancient protocol designed for serial ports (UART), and it is still in use today for low end processors that lack USB support. It is possible to make a PPP call over either UART or USB serial ports (ttyUSB).

I am using a Raspberry Pi 4 to connect to an EG91NAXD as an example here.

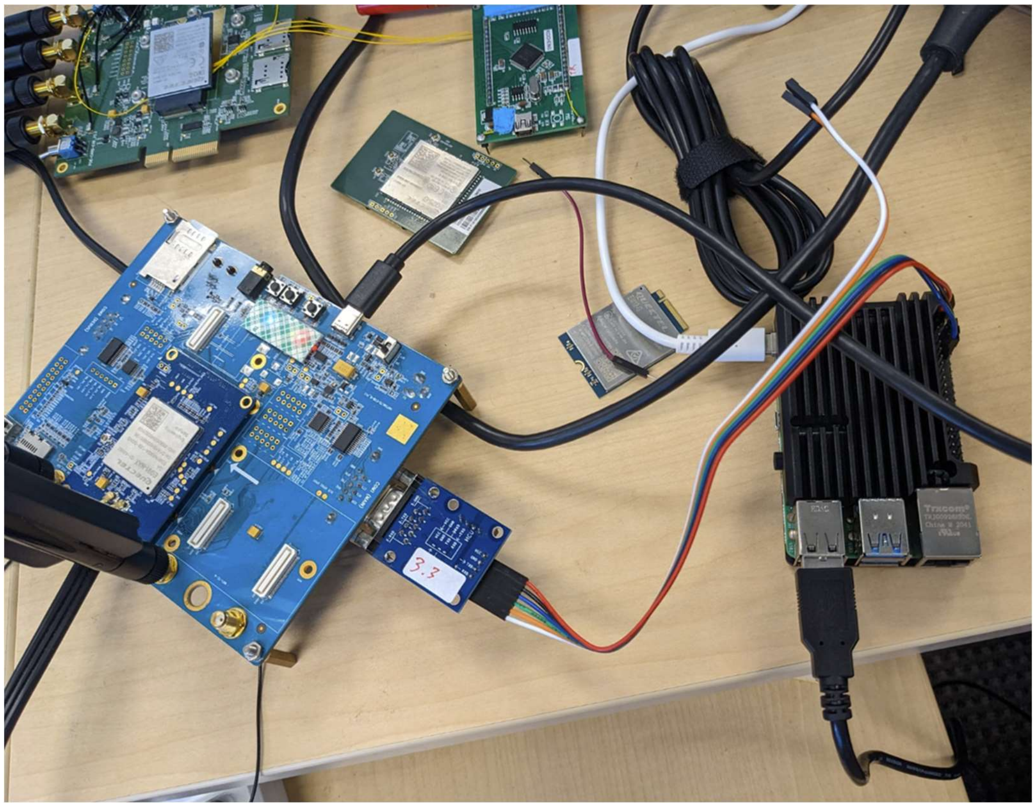
The COM1(MAIN) port on the UMTS&LTE-EVB-B_V1.1 connects to the module's UART. It is running at RS232 voltage level.

https://en.wikipedia.org/wiki/RS-232

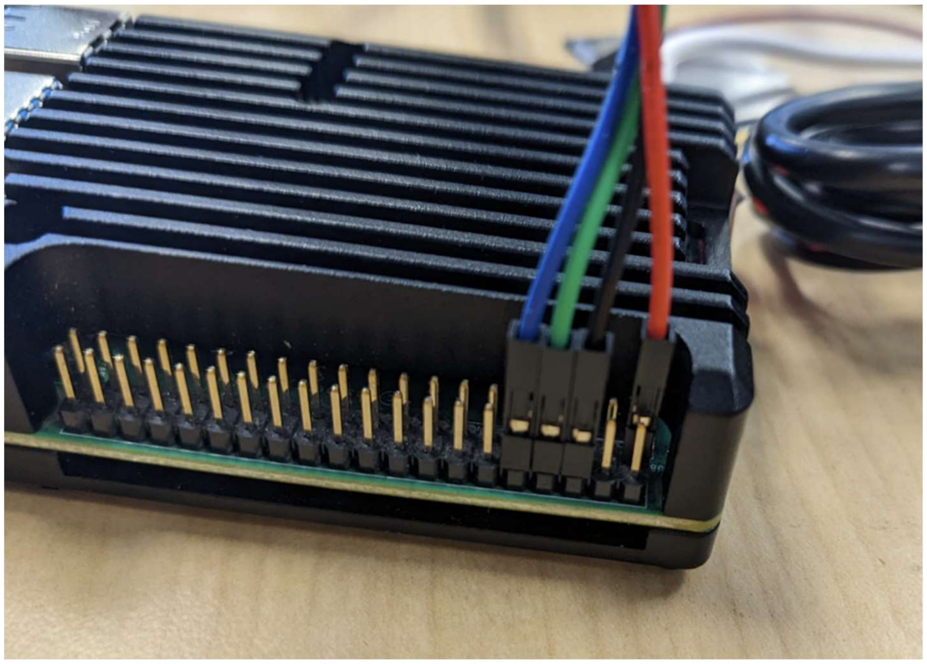
A voltage level adapter is needed to connect to Raspberry's UART port, as shown in the picture below, i.e., the dark blue board labelled 3.3

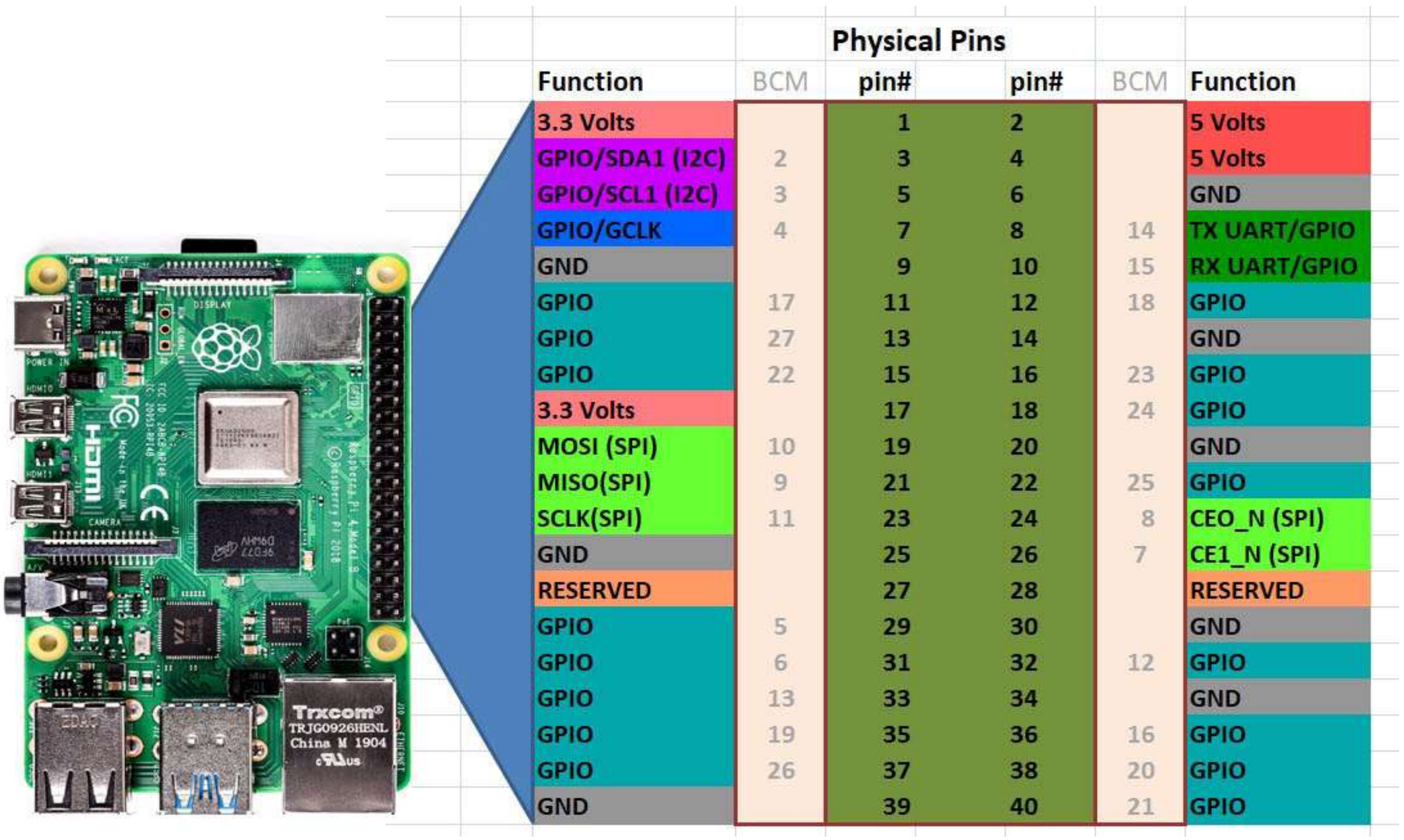
I am using this page as reference on raspberry's UART port

https://forums.raspberrypi.com/viewtopic.php?t=307094



serial 0 and 1 on the raspberry are sharing the same header pins, they cannot be used at the same time.
My voltage adapter board takes 3.3V from the Raspberry Pi

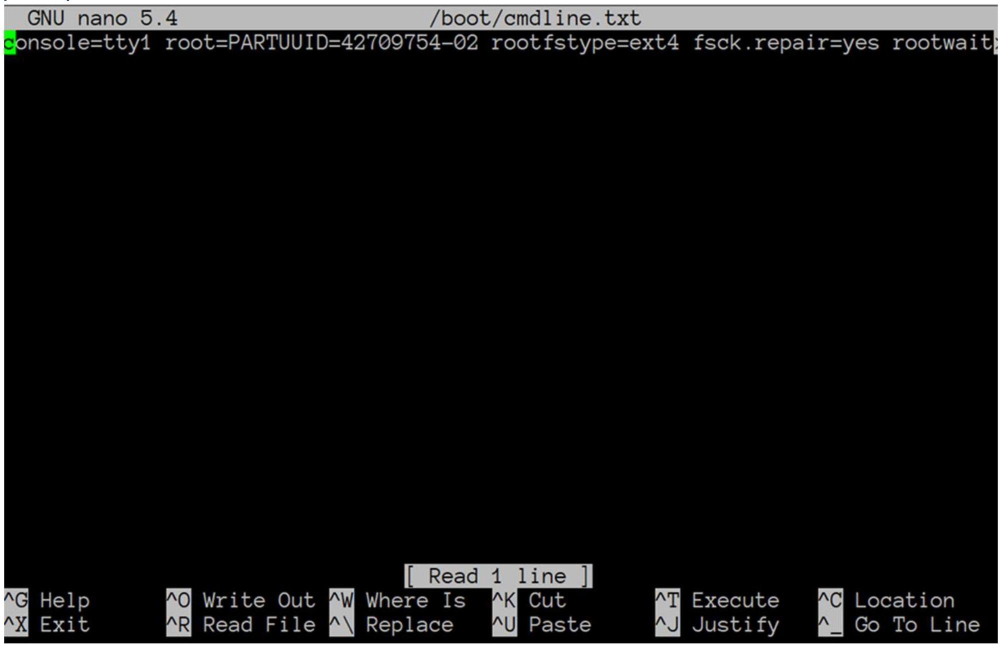




		Physical Pins			
Function	BCM	pin#	pin#	BCM	Function
3.3 Volts		1	2		5 Volts
GPIO/SDA1 (I2C)	2	3	4		5 Volts
GPIO/SCL1 (I2C)	3	5	6		GND
GPIO/GCLK	4	7	8	14	TX UART/GPIO
GND		9	10	15	RX UART/GPIO
GPIO	17	11	12	18	GPIO
GPIO	27	13	14		GND
GPIO	22	15	16	23	GPIO
3.3 Volts		17	18	24	GPIO
MOSI (SPI)	10	19	20		GND
MISO(SPI)	9	21	22	25	GPIO
SCLK(SPI)	11	23	24	8	CEO_N (SPI)
GND		25	26	7	CE1_N (SPI)
RESERVED		27	28		RESERVED
GPIO	5	29	30		GND
GPIO	6	31	32	12	GPIO
GPIO	13	33	34		GND
GPIO	19	35	36	16	GPIO
GPIO	26	37	38	20	GPIO
GND		39	40	21	GPIO

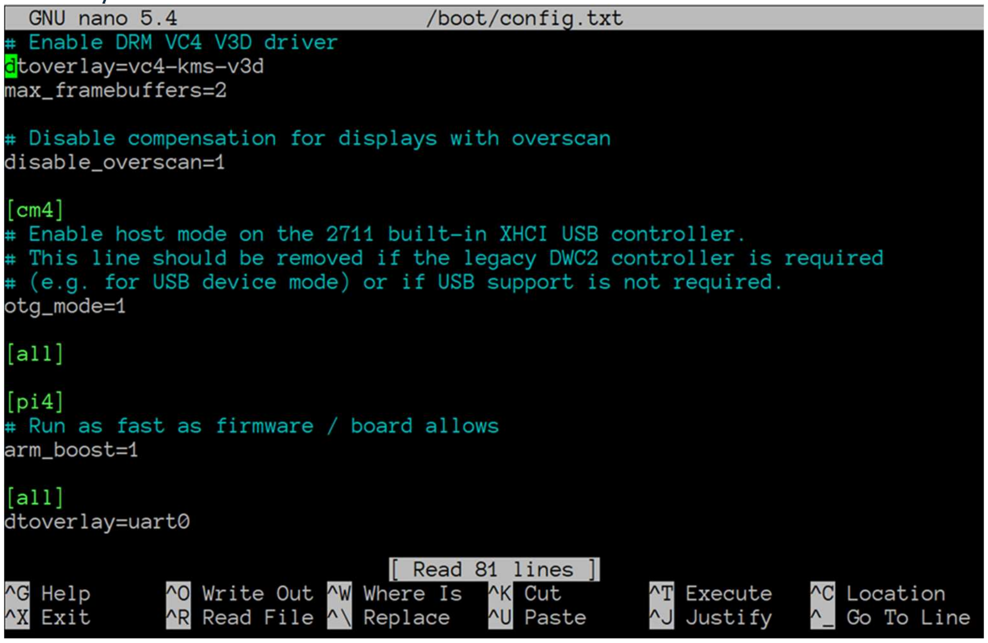
For some reason I am enabling uart0 in raspberry pi 4 but it is named /dev/serial1 in the OS
I am not 100% sure how Linux kernel assigns the number, but it works. Please check with the documents for the platform of your choice for details on how to configure the board.

Make sure Linux console is not using the serial port, neither serial0 nor serial 1 should be in use by the kernel.
There was a console=serial1 in the kernel command line and I removed it.
/boot/cmdline.txt



```
  GNU nano 5.4                        /boot/cmdline.txt
console=tty1 root=PARTUUID=42709754-02 rootfstype=ext4 fsck.repair=yes rootwait
```

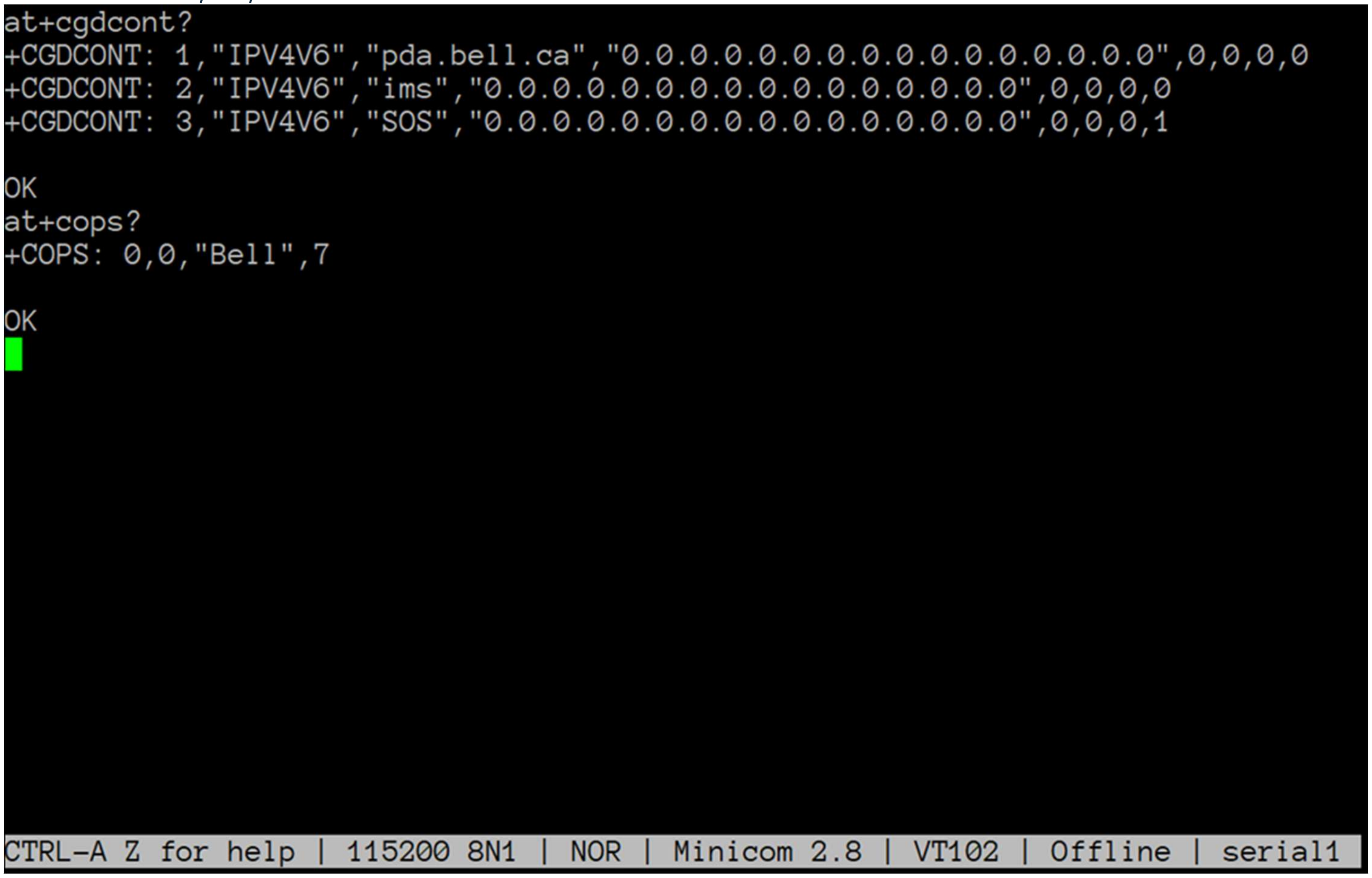
And I added this line to /boot/config.txt
dtoverlay=uart0



```
  GNU nano 5.4                        /boot/config.txt
# Enable DRM VC4 V3D driver
dtoverlay=vc4-kms-v3d
max_framebuffers=2

# Disable compensation for displays with overscan
disable_overscan=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[all]

[pi4]
# Run as fast as firmware / board allows
arm_boost=1

[all]
dtoverlay=uart0
```

please reboot the raspberry to for the changes to take effect.

Test AT port with minicom
sudo minicom -D /dev/serial1 -b 115200



```
at+cgdcont?
+CGDCONT: 1,"IPV4V6","pda.bell.ca","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 2,"IPV4V6","ims","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 3,"IPV4V6","SOS","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1

OK
at+cops?
+COPS: 0,0,"Bell",7

OK

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.8 | VT102 | Offline | serial1
```

PPP call script
linux-ppp-scripts_V1.2.zip

https://cnquectel-my.sharepoint.com/:u:/g/personal/america-fae_quectel_com/EV9khtiXDuFFsVokT9SOEtoBK1Si2pjZf26-7ttRVz23VQ?e=EfqZ19

Unzip linux-ppp-scripts_V1.2.zip to /etc/ppp/peers
convert to UNIX format just in case.
dos2unix /etc/ppp/peers/*

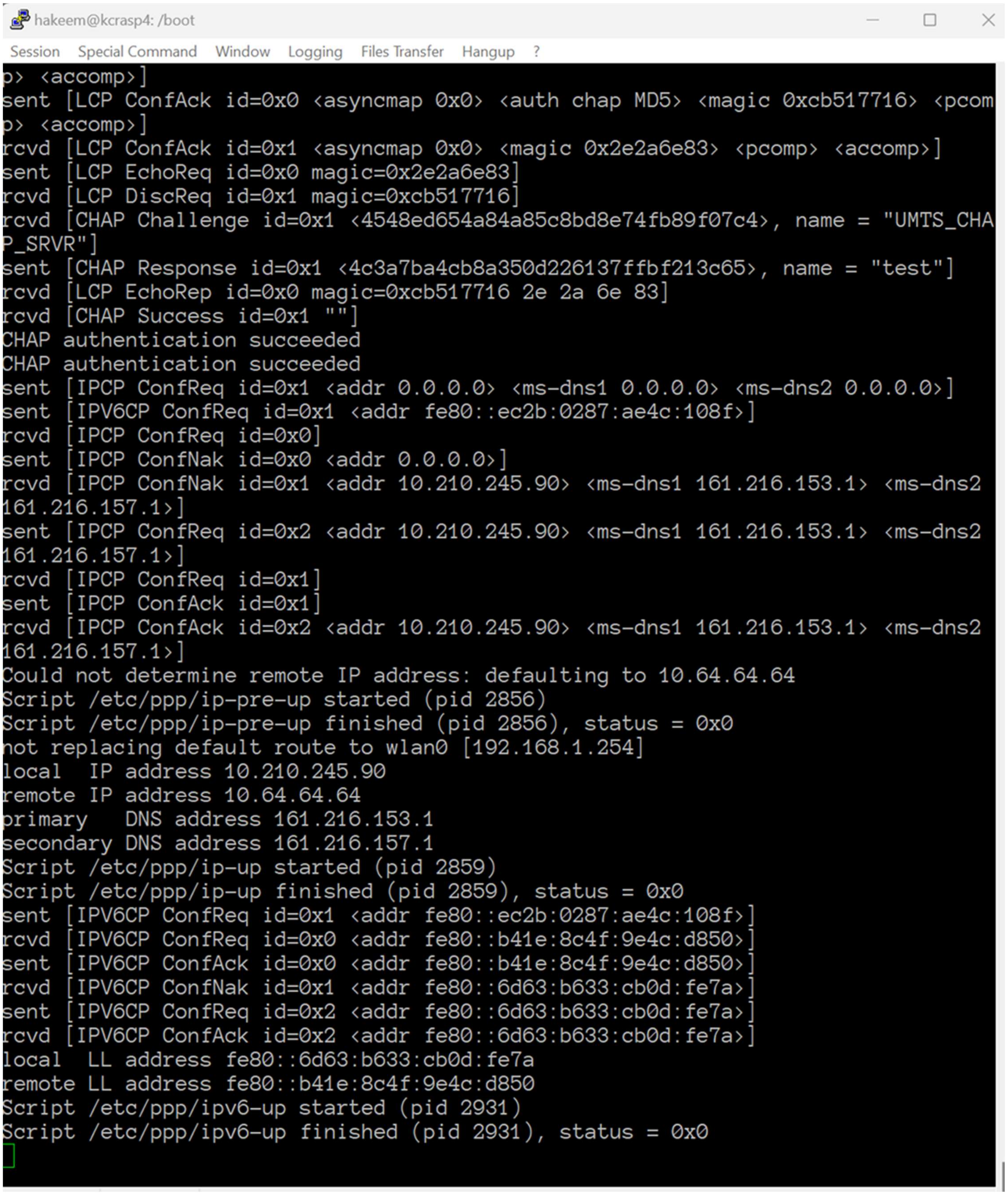
Please update or remove APN setup, in line 12 for quectel-chat-connect. For LTE and up, APN should have been set properly before making the call. Please don't update APN on the fly.
sudo nano quectel-chat-connect

update serial port name in quectel-ppp line 4. Typically, it could be /dev/ttyUSB2, /dev/ttyUSB3, or in the case of Raspberry's UART, /dev/serial1
sudo nano quectel-ppp

make a data call.
sudo pppd call quectel-ppp&

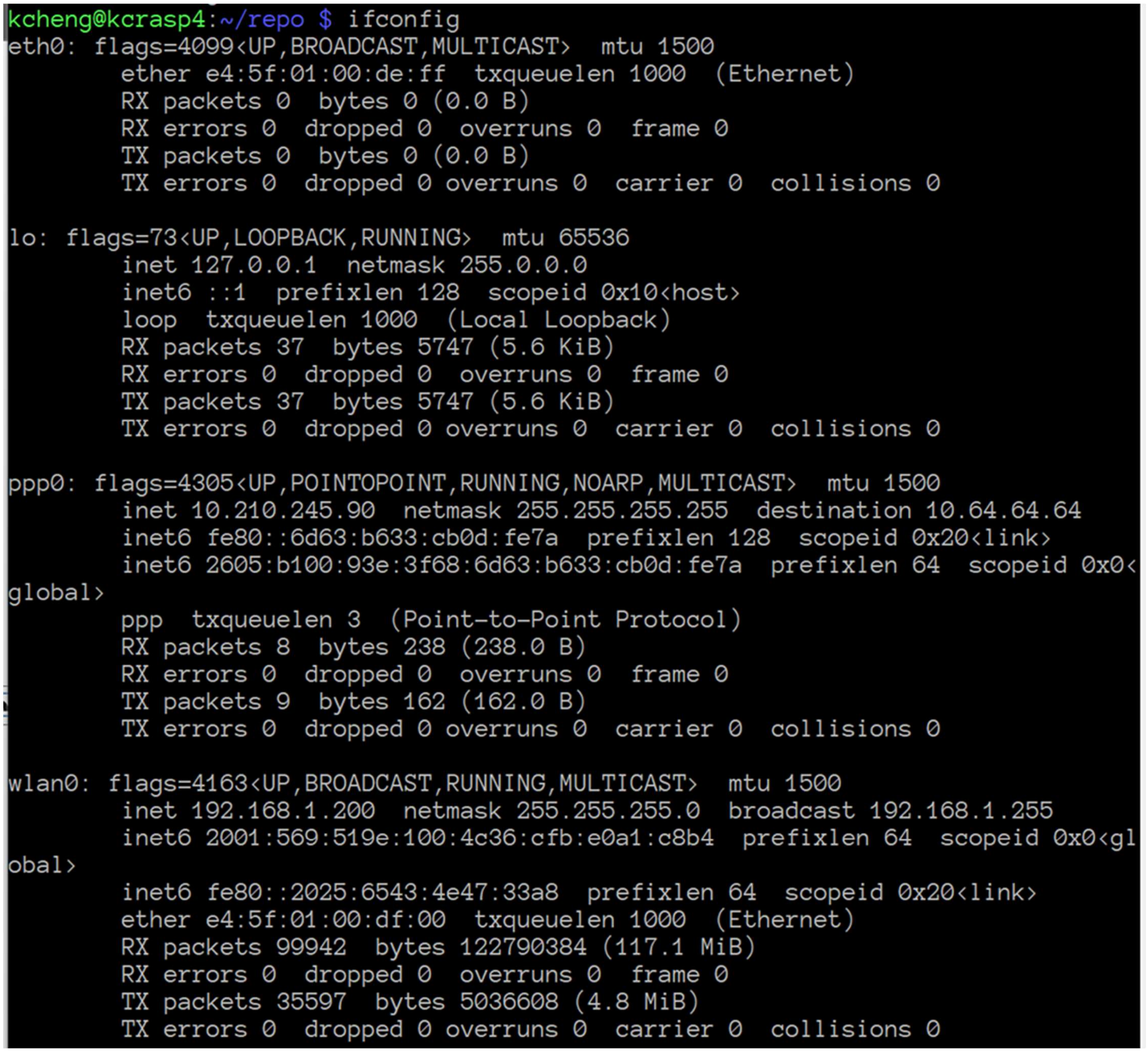
disconnect data call.
sudo killall pppd

Example PPP call progress


hakeem@kcrasp4: /boot
Session Special Command Window Logging Files Transfer Hangup ?

```
p> <accomp>]
sent [LCP ConfAck id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0xcb517716> <pcom
p> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x2e2a6e83> <pcomp> <accomp>]
sent [LCP EchoReq id=0x0 magic=0x2e2a6e83]
rcvd [LCP DiscReq id=0x1 magic=0xcb517716]
rcvd [CHAP Challenge id=0x1 <4548ed654a84a85c8bd8e74fb89f07c4>, name = "UMTS_CHA
P_SRVR"]
sent [CHAP Response id=0x1 <4c3a7ba4cb8a350d226137ffbf213c65>, name = "test"]
rcvd [LCP EchoRep id=0x0 magic=0xcb517716 2e 2a 6e 83]
rcvd [CHAP Success id=0x1 ""]
CHAP authentication succeeded
CHAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
sent [IPV6CP ConfReq id=0x1 <addr fe80::ec2b:0287:ae4c:108f>]
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.210.245.90> <ms-dns1 161.216.153.1> <ms-dns2
161.216.157.1>]
sent [IPCP ConfReq id=0x2 <addr 10.210.245.90> <ms-dns1 161.216.153.1> <ms-dns2
161.216.157.1>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfAck id=0x2 <addr 10.210.245.90> <ms-dns1 161.216.153.1> <ms-dns2
161.216.157.1>]
Could not determine remote IP address: defaulting to 10.64.64.64
Script /etc/ppp/ip-pre-up started (pid 2856)
Script /etc/ppp/ip-pre-up finished (pid 2856), status = 0x0
not replacing default route to wlan0 [192.168.1.254]
local  IP address 10.210.245.90
remote IP address 10.64.64.64
primary   DNS address 161.216.153.1
secondary DNS address 161.216.157.1
Script /etc/ppp/ip-up started (pid 2859)
Script /etc/ppp/ip-up finished (pid 2859), status = 0x0
sent [IPV6CP ConfReq id=0x1 <addr fe80::ec2b:0287:ae4c:108f>]
rcvd [IPV6CP ConfReq id=0x0 <addr fe80::b41e:8c4f:9e4c:d850>]
sent [IPV6CP ConfAck id=0x0 <addr fe80::b41e:8c4f:9e4c:d850>]
rcvd [IPV6CP ConfNak id=0x1 <addr fe80::6d63:b633:cb0d:fe7a>]
sent [IPV6CP ConfReq id=0x2 <addr fe80::6d63:b633:cb0d:fe7a>]
rcvd [IPV6CP ConfAck id=0x2 <addr fe80::6d63:b633:cb0d:fe7a>]
local  LL address fe80::6d63:b633:cb0d:fe7a
remote LL address fe80::b41e:8c4f:9e4c:d850
Script /etc/ppp/ipv6-up started (pid 2931)
Script /etc/ppp/ipv6-up finished (pid 2931), status = 0x0
```

And after the call, ifconfig shows the ppp0 interface.



```
kcheng@kcrasp4:~/repo $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether e4:5f:01:00:de:ff  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 37  bytes 5747 (5.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 37  bytes 5747 (5.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.210.245.90  netmask 255.255.255.255  destination 10.64.64.64
        inet6 fe80::6d63:b633:cb0d:fe7a  prefixlen 128  scopeid 0x20<link>
        inet6 2605:b100:93e:3f68:6d63:b633:cb0d:fe7a  prefixlen 64  scopeid 0x0<
global>
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 8  bytes 238 (238.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 162 (162.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.200  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 2001:569:519e:100:4c36:cfb:e0a1:c8b4  prefixlen 64  scopeid 0x0<gl
obal>
        inet6 fe80::2025:6543:4e47:33a8  prefixlen 64  scopeid 0x20<link>
        ether e4:5f:01:00:df:00  txqueuelen 1000  (Ethernet)
        RX packets 99942  bytes 122790384 (117.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 35597  bytes 5036608 (4.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Depending on OS you are using, you may have to update default route and setup DNS. Try these.

route add default gw ppp0

route add default ppp0

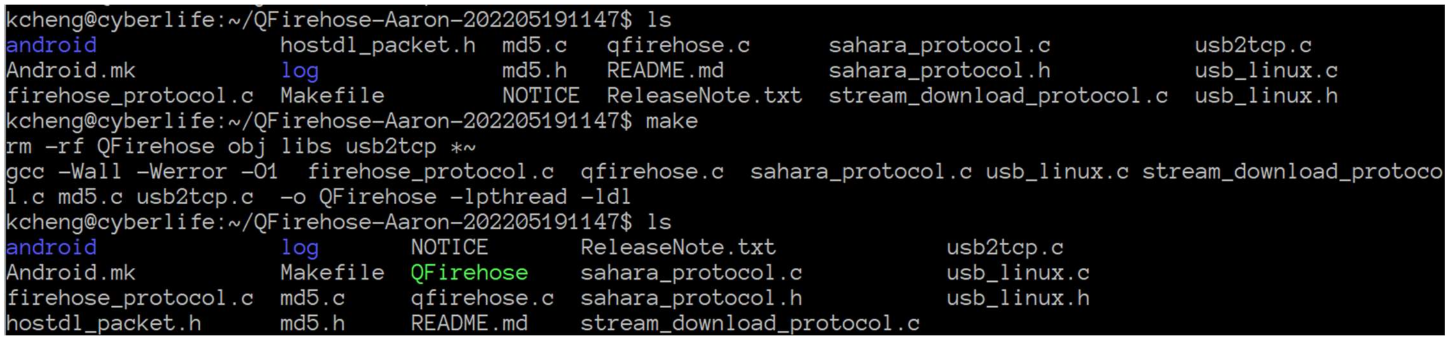
https://wiki.archlinux.org/title/systemd-resolved

Custom routing and DNS settings

https://askubuntu.com/questions/157154/how-do-i-include-lines-in-resolv-conf-that-wont-get-lost-on-reboot

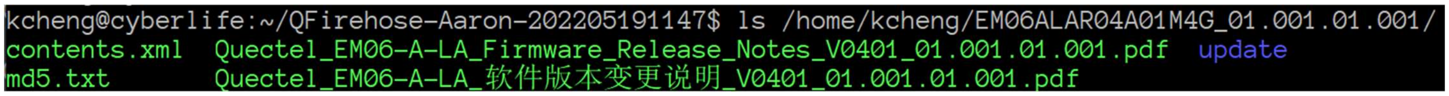
Firmware upgrade using QFirehose

The QFirehose firmware upgrade tool is provided in source code form. We can build the source in the target board with a simple "make" command.



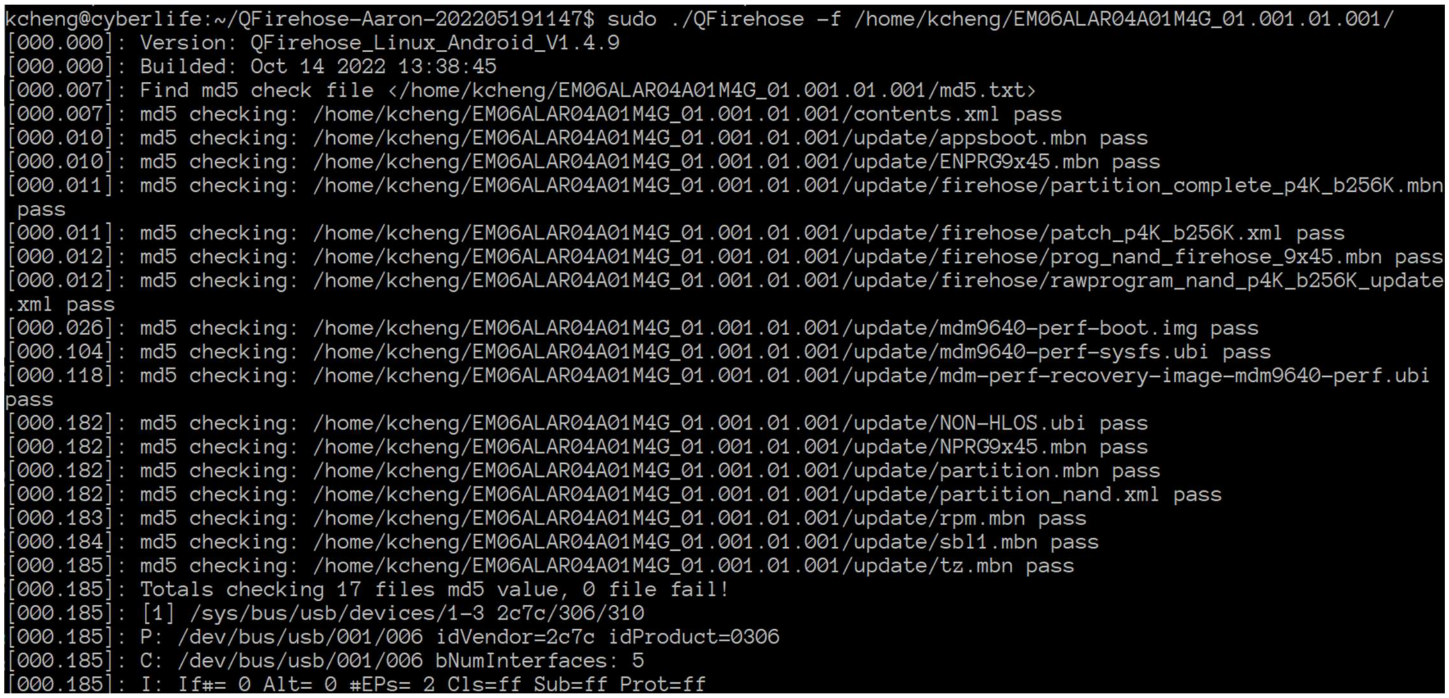
```
kcheng@cyberlife:~/QFirehose-Aaron-202205191147$ ls
android             hostdl_packet.h  md5.c   qfirehose.c      sahara_protocol.c           usb2tcp.c
Android.mk          log              md5.h   README.md        sahara_protocol.h           usb_linux.c
firehose_protocol.c Makefile         NOTICE  ReleaseNote.txt  stream_download_protocol.c  usb_linux.h
kcheng@cyberlife:~/QFirehose-Aaron-202205191147$ make
rm -rf QFirehose obj libs usb2tcp *~
gcc -Wall -Werror -O1  firehose_protocol.c  qfirehose.c  sahara_protocol.c usb_linux.c stream_download_protoco
l.c md5.c usb2tcp.c  -o QFirehose -lpthread -ldl
kcheng@cyberlife:~/QFirehose-Aaron-202205191147$ ls
android             log       NOTICE     ReleaseNote.txt             usb2tcp.c
Android.mk          Makefile  QFirehose  sahara_protocol.c           usb_linux.c
firehose_protocol.c md5.c     qfirehose.c sahara_protocol.h          usb_linux.h
hostdl_packet.h     md5.h     README.md  stream_download_protocol.c
```

Start the firmware upgrade by running ***QFirehose -f <path to unzipped firmware>*** with root privileges. You may have to add -n for beta firmware release to skip MD5 check. Most module will reboot itself after firmware update is completed.



```
kcheng@cyberlife:~/QFirehose-Aaron-202205191147$ ls /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/
contents.xml  Quectel_EM06-A-LA_Firmware_Release_Notes_V0401_01.001.01.001.pdf  update
md5.txt       Quectel_EM06-A-LA_软件版本变更说明_V0401_01.001.01.001.pdf
```

sudo ./QFirehose -f /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/



```
kcheng@cyberlife:~/QFirehose-Aaron-202205191147$ sudo ./QFirehose -f /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/
[000.000]: Version: QFirehose_Linux_Android_V1.4.9
[000.000]: Builded: Oct 14 2022 13:38:45
[000.007]: Find md5 check file </home/kcheng/EM06ALAR04A01M4G_01.001.01.001/md5.txt>
[000.007]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/contents.xml pass
[000.010]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/appsboot.mbn pass
[000.010]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/ENPRG9x45.mbn pass
[000.011]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/firehose/partition_complete_p4K_b256K.mbn
 pass
[000.011]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/firehose/patch_p4K_b256K.xml pass
[000.012]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/firehose/prog_nand_firehose_9x45.mbn pass
[000.012]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/firehose/rawprogram_nand_p4K_b256K_update
.xml pass
[000.026]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/mdm9640-perf-boot.img pass
[000.104]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/mdm9640-perf-sysfs.ubi pass
[000.118]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/mdm-perf-recovery-image-mdm9640-perf.ubi
pass
[000.182]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/NON-HLOS.ubi pass
[000.182]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/NPRG9x45.mbn pass
[000.182]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/partition.mbn pass
[000.182]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/partition_nand.xml pass
[000.183]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/rpm.mbn pass
[000.184]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/sbl1.mbn pass
[000.185]: md5 checking: /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/update/tz.mbn pass
[000.185]: Totals checking 17 files md5 value, 0 file fail!
[000.185]: [1] /sys/bus/usb/devices/1-3 2c7c/306/310
[000.185]: P: /dev/bus/usb/001/006 idVendor=2c7c idProduct=0306
[000.185]: C: /dev/bus/usb/001/006 bNumInterfaces: 5
[000.185]: I: If#= 0 Alt= 0 #EPs= 2 Cls=ff Sub=ff Prot=ff
```

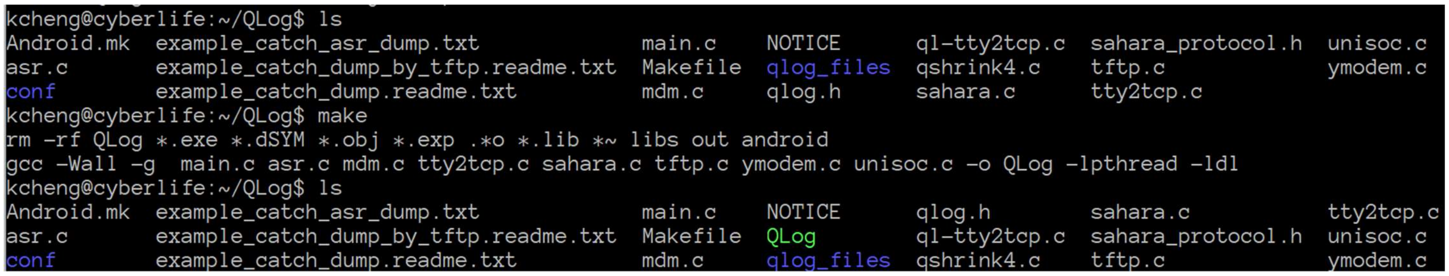
To specify a specific port name if you have multiple modules, or you are using a PCIE module, use the **-p <path-to-port>** command line parameter. e.g.,

sudo ./QFirehose -f /home/kcheng/EM06ALAR04A01M4G_01.001.01.001/ -p /dev/mhi_BHI

Module logging

QLog is a tool to collect module logs for debugging purposes. It is also available in source code form.

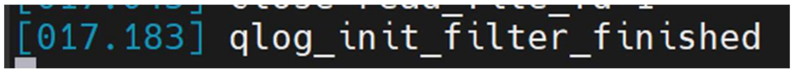
Just "make" it



```
kcheng@cyberlife:~/QLog$ ls
Android.mk  example_catch_asr_dump.txt            main.c    NOTICE      ql-tty2tcp.c  sahara_protocol.h  unisoc.c
asr.c       example_catch_dump_by_tftp.readme.txt Makefile  qlog_files  qshrink4.c    tftp.c             ymodem.c
conf        example_catch_dump.readme.txt         mdm.c     qlog.h      sahara.c      tty2tcp.c
kcheng@cyberlife:~/QLog$ make
rm -rf QLog *.exe *.dSYM *.obj *.exp .*o *.lib *~ libs out android
gcc -Wall -g  main.c asr.c mdm.c tty2tcp.c sahara.c tftp.c ymodem.c unisoc.c -o QLog -lpthread -ldl
kcheng@cyberlife:~/QLog$ ls
Android.mk  example_catch_asr_dump.txt            main.c    NOTICE      qlog.h        sahara.c           tty2tcp.c
asr.c       example_catch_dump_by_tftp.readme.txt Makefile  QLog        ql-tty2tcp.c  sahara_protocol.h  unisoc.c
conf        example_catch_dump.readme.txt         mdm.c     qlog_files  qshrink4.c    tftp.c             ymodem.c
```

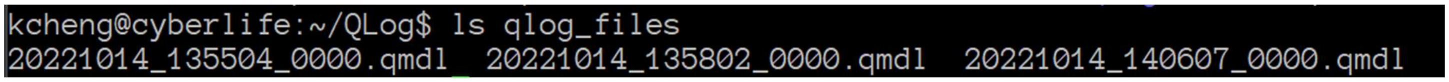
Execute it with root privileges. We may provide a log filer file in .cfg form. Use the -f command line parameter to specify the filter file.

QLog will take some time to initialize, for LTE modules it takes a few seconds, but for 5G modules it can take half a minute. Please wait for this "qlog_init_filter_finished" prompt before trying to reproduce the issue.



```
[017.183] qlog_init_filter_finished
```

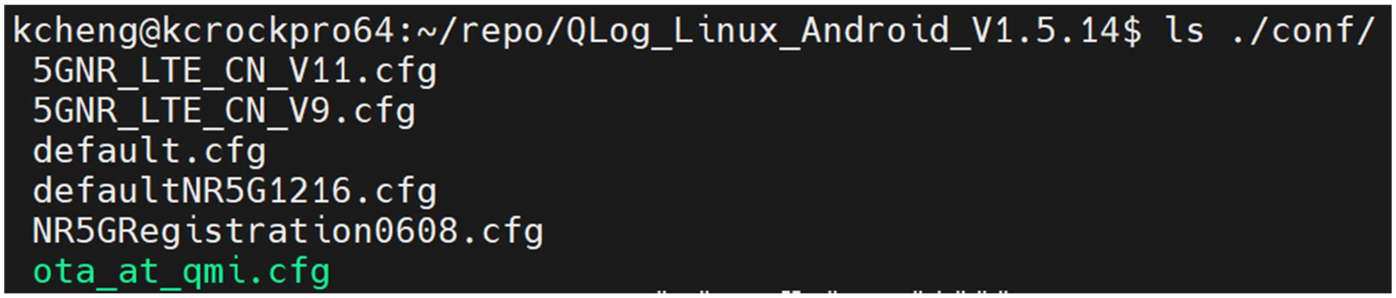
Once QLog is running, you can press CTRL+C to stop QLog. The log files will be stored in a subfolder called qlog_files, like this.



```
kcheng@cyberlife:~/QLog$ ls qlog_files
20221014_135504_0000.qmdl  20221014_135802_0000.qmdl  20221014_140607_0000.qmdl
```

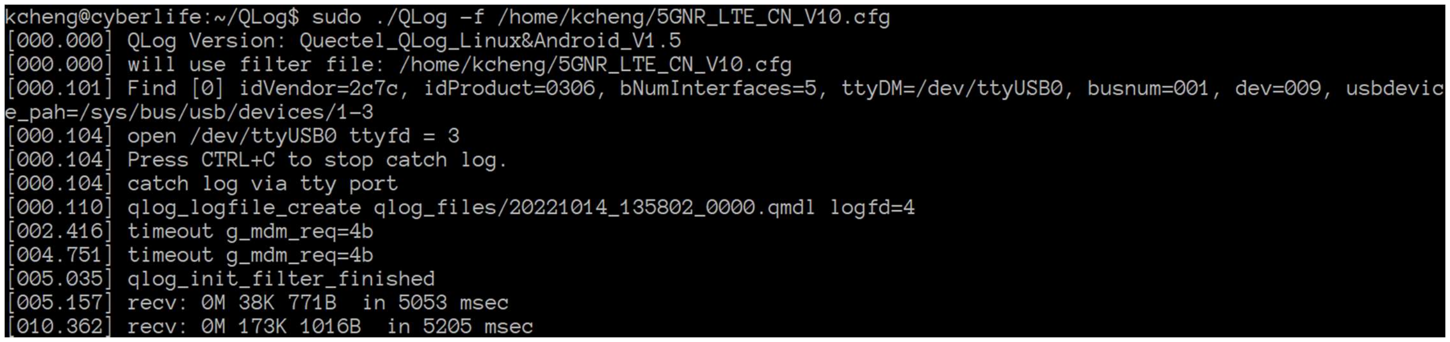
For some newer modules an AT command is needed to enable debug logging. Without it the collected log file will be almost empty. To enable debug logging, use this command. The setting is persistent across power cycle.
at+qcfg="dbgctl",0

To check current setting
at+qcfg="dbgctl"
example command line. Note the V10 part of the name 5GNR_LTE_CN_**V10**.cfg is a revision number and it can be updated over time.



```
kcheng@kcrockpro64:~/repo/QLog_Linux_Android_V1.5.14$ ls ./conf/
 5GNR_LTE_CN_V11.cfg
 5GNR_LTE_CN_V9.cfg
 default.cfg
 defaultNR5G1216.cfg
 NR5GRegistration0608.cfg
 ota_at_qmi.cfg
```

sudo ./QLog -f /home/kcheng/5GNR_LTE_CN_VXX.cfg



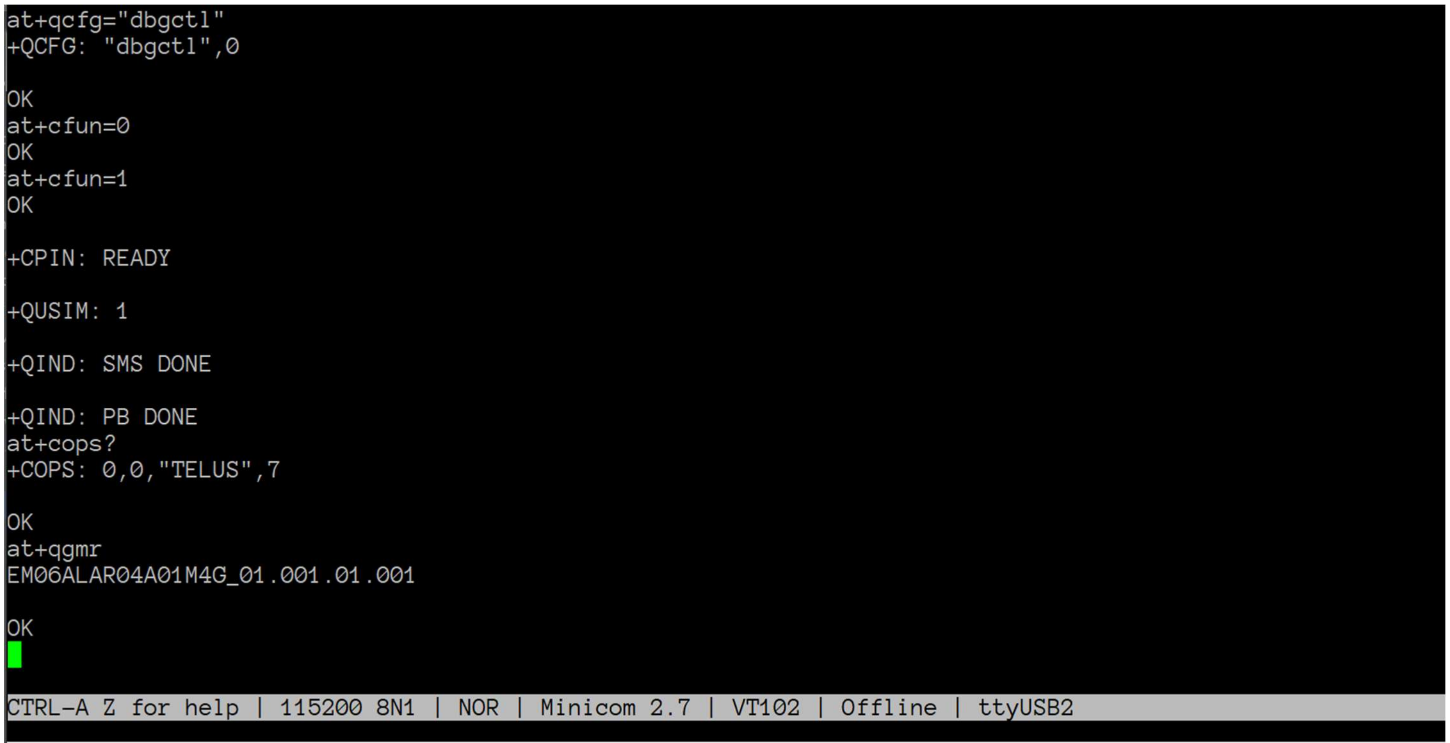
```
kcheng@cyberlife:~/QLog$ sudo ./QLog -f /home/kcheng/5GNR_LTE_CN_V10.cfg
[000.000] QLog Version: Quectel_QLog_Linux&Android_V1.5
[000.000] will use filter file: /home/kcheng/5GNR_LTE_CN_V10.cfg
[000.101] Find [0] idVendor=2c7c, idProduct=0306, bNumInterfaces=5, ttyDM=/dev/ttyUSB0, busnum=001, dev=009, usbdevic
e_pah=/sys/bus/usb/devices/1-3
[000.104] open /dev/ttyUSB0 ttyfd = 3
[000.104] Press CTRL+C to stop catch log.
[000.104] catch log via tty port
[000.110] qlog_logfile_create qlog_files/20221014_135802_0000.qmdl logfd=4
[002.416] timeout g_mdm_req=4b
[004.751] timeout g_mdm_req=4b
[005.035] qlog_init_filter_finished
[005.157] recv: 0M 38K 771B  in 5053 msec
[010.362] recv: 0M 173K 1016B  in 5205 msec
```

To specify a specific port name if you have multiple modules, or you are using a PCIE module, use the **-p <path-to-port>** command line parameter. e.g.,
sudo ./QLog -f /home/kcheng/5GNR_LTE_CN_V10.cfg -p /dev/mhi_DIAG

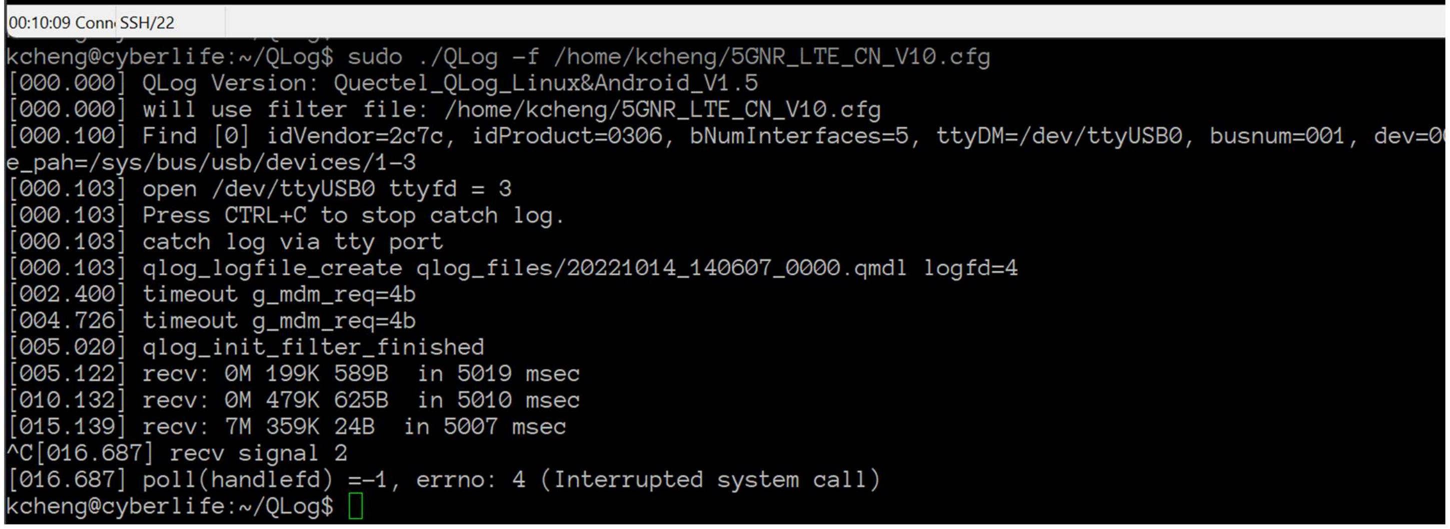
For data call or network registration related issues, a log starting from the beginning including network attach is usually needed. We can use AT command to disable radio and start logging. i.e.

- AT+QCFG="dbgctl",0 (it may return ERROR for some modules, ignore it)
- AT+CFUN=0
- Start QLog and wait for a few seconds for it to initialize. Wait for the prompt "**qlog_init_filter_finished",** for the log filter to take effect. For 5G modules it may take up to half a minute.
- AT+CFUN=1
- Wait for a few seconds and run some AT commands to collect basic information.
- AT+QGMR
- AT+CIMI
- AT+CGSN
- AT+QMBNCFG="list"
- AT+CGDCONT?
- AT+CGCONTRDP
- AT+COPS?
- AT+QNWINFO
- AT+QCSQ
- AT+CREG?
- AT+CEREG?
- AT+C5GREG?
- AT+QENG="SERVINGCELL"
- AT+QENG="NEIGHBOURCELL"
- AT+QCFG="NWSCANMODE" (for LPWA modules)
- AT+QOPSCFG="SCANCONTROL" (for LPWA modules)
- Maybe even AT+COPS=?
- Make data call with quectel-CM if it is related.

For example



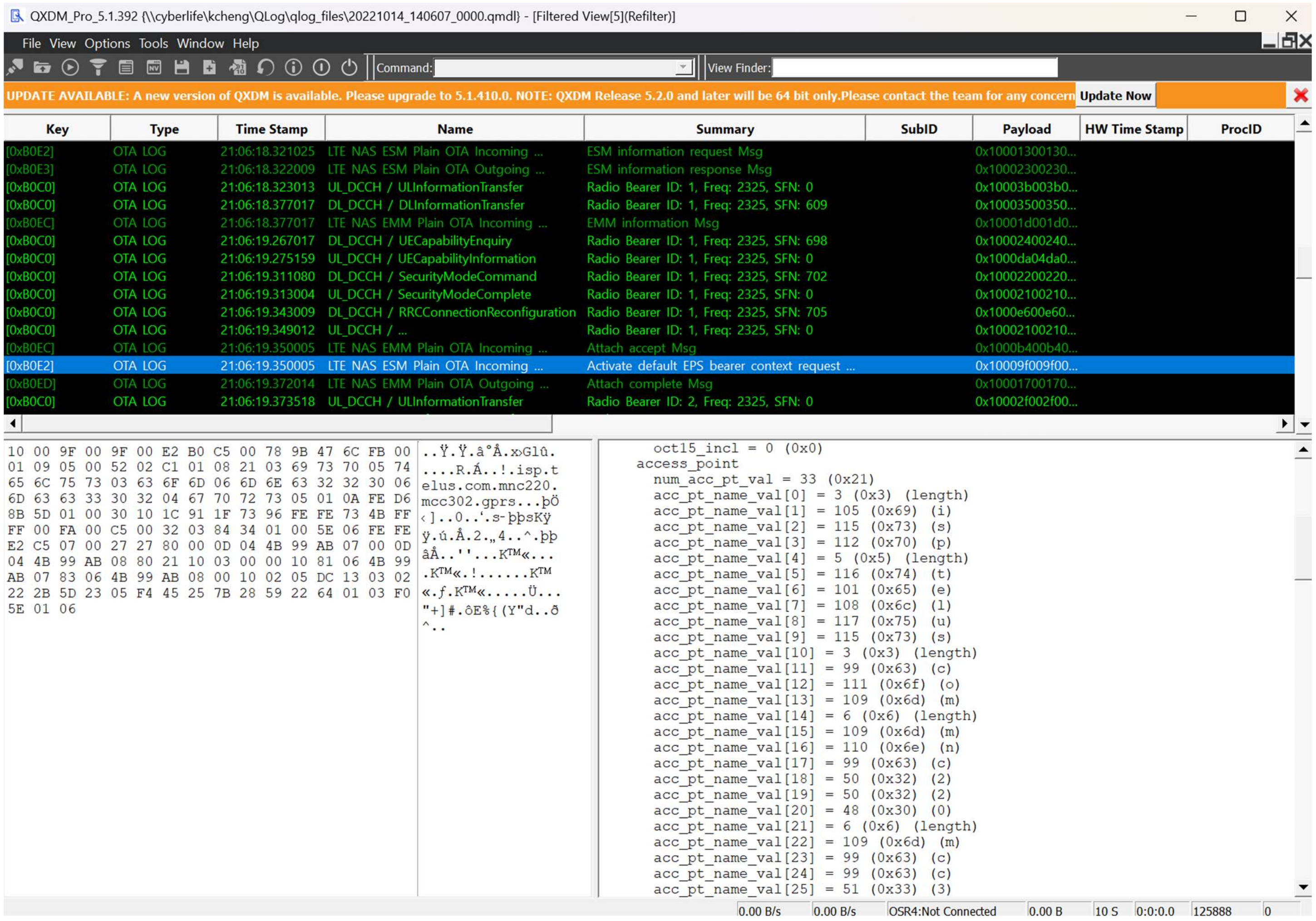
```
at+qcfg="dbgctl"
+QCFG: "dbgctl",0

OK
at+cfun=0
OK
at+cfun=1
OK

+CPIN: READY

+QUSIM: 1

+QIND: SMS DONE

+QIND: PB DONE
at+cops?
+COPS: 0,0,"TELUS",7

OK
at+qgmr
EM06ALAR04A01M4G_01.001.01.001

OK

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB2
```



```
00:10:09 Conn SSH/22
kcheng@cyberlife:~/QLog$ sudo ./QLog -f /home/kcheng/5GNR_LTE_CN_V10.cfg
[000.000] QLog Version: Quectel_QLog_Linux&Android_V1.5
[000.000] will use filter file: /home/kcheng/5GNR_LTE_CN_V10.cfg
[000.100] Find [0] idVendor=2c7c, idProduct=0306, bNumInterfaces=5, ttyDM=/dev/ttyUSB0, busnum=001, dev=0
e_pah=/sys/bus/usb/devices/1-3
[000.103] open /dev/ttyUSB0 ttyfd = 3
[000.103] Press CTRL+C to stop catch log.
[000.103] catch log via tty port
[000.103] qlog_logfile_create qlog_files/20221014_140607_0000.qmdl logfd=4
[002.400] timeout g_mdm_req=4b
[004.726] timeout g_mdm_req=4b
[005.020] qlog_init_filter_finished
[005.122] recv: 0M 199K 589B  in 5019 msec
[010.132] recv: 0M 479K 625B  in 5010 msec
[015.139] recv: 7M 359K 24B  in 5007 msec
^C[016.687] recv signal 2
[016.687] poll(handlefd) =-1, errno: 4 (Interrupted system call)
kcheng@cyberlife:~/QLog$
```

Please share the log file with us and we should be able to extract IP traffic (partial), see cellular handshakes and possibly find bugs. The log file could be quite big, and we can try dropbox or google drive to share the file.

As we can see in this example, cellular OTA traffic is captured, and the APN used in network attach is seen.


QXDM_Pro_5.1.392 {\\cyberlife\kcheng\QLog\qlog_files\20221014_140607_0000.qmdl} - [Filtered View[5](Refilter)]

UPDATE AVAILABLE: A new version of QXDM is available. Please upgrade to 5.1.410.0. NOTE: QXDM Release 5.2.0 and later will be 64 bit only.Please contact the team for any concern

Key	Type	Time Stamp	Name	Summary	SubID	Payload	HW Time Stamp	ProcID
[0xB0E2]	OTA LOG	21:06:18.321025	LTE NAS ESM Plain OTA Incoming ...	ESM information request Msg		0x10001300130...		
[0xB0E3]	OTA LOG	21:06:18.322009	LTE NAS ESM Plain OTA Outgoing ...	ESM information response Msg		0x10002300230...		
[0xB0C0]	OTA LOG	21:06:18.323013	UL_DCCH / ULInformationTransfer	Radio Bearer ID: 1, Freq: 2325, SFN: 0		0x10003b003b0...		
[0xB0C0]	OTA LOG	21:06:18.377017	DL_DCCH / DLInformationTransfer	Radio Bearer ID: 1, Freq: 2325, SFN: 609		0x10003500350...		
[0xB0EC]	OTA LOG	21:06:18.377017	LTE NAS EMM Plain OTA Incoming ...	EMM information Msg		0x10001d001d0...		
[0xB0C0]	OTA LOG	21:06:19.267017	DL_DCCH / UECapabilityEnquiry	Radio Bearer ID: 1, Freq: 2325, SFN: 698		0x10002400240...		
[0xB0C0]	OTA LOG	21:06:19.275159	UL_DCCH / UECapabilityInformation	Radio Bearer ID: 1, Freq: 2325, SFN: 0		0x1000da04da0...		
[0xB0C0]	OTA LOG	21:06:19.311080	DL_DCCH / SecurityModeCommand	Radio Bearer ID: 1, Freq: 2325, SFN: 702		0x10002200220...		
[0xB0C0]	OTA LOG	21:06:19.313004	UL_DCCH / SecurityModeComplete	Radio Bearer ID: 1, Freq: 2325, SFN: 0		0x10002100210...		
[0xB0C0]	OTA LOG	21:06:19.343009	DL_DCCH / RRCConnectionReconfiguration	Radio Bearer ID: 1, Freq: 2325, SFN: 705		0x1000e600e60...		
[0xB0C0]	OTA LOG	21:06:19.349012	UL_DCCH / ...	Radio Bearer ID: 1, Freq: 2325, SFN: 0		0x10002100210...		
[0xB0EC]	OTA LOG	21:06:19.350005	LTE NAS EMM Plain OTA Incoming ...	Attach accept Msg		0x1000b400b40...		
[0xB0E2]	OTA LOG	21:06:19.350005	LTE NAS ESM Plain OTA Incoming ...	Activate default EPS bearer context request ...		0x10009f009f00...		
[0xB0ED]	OTA LOG	21:06:19.372014	LTE NAS EMM Plain OTA Outgoing ...	Attach complete Msg		0x10001700170...		
[0xB0C0]	OTA LOG	21:06:19.373518	UL_DCCH / ULInformationTransfer	Radio Bearer ID: 2, Freq: 2325, SFN: 0		0x10002f002f00...		

```
10 00 9F 00 9F 00 E2 B0 C5 00 78 9B 47 6C FB 00   ..Ÿ.Ÿ.â°Å.x›Glû.
01 09 05 00 52 02 C1 01 08 21 03 69 73 70 05 74   ....R.Á..!.isp.t
65 6C 75 73 03 63 6F 6D 06 6D 6E 63 32 32 30 06   elus.com.mnc220.
6D 63 63 33 30 32 04 67 70 72 73 05 01 0A FE D6   mcc302.gprs...þÖ
8B 5D 01 00 30 10 1C 91 1F 73 96 FE FE 73 4B FF   ‹]..0..‘.s–þþsKÿ
FF 00 FA 00 C5 00 32 03 84 34 01 00 5E 06 FE FE   ÿ.ú.Å.2.„4..^.þþ
E2 C5 07 00 27 27 80 00 0D 04 4B 99 AB 07 00 0D   âÅ..''...K™«...
04 4B 99 AB 08 80 21 10 03 00 00 10 81 06 4B 99   .K™«.!......K™
AB 07 83 06 4B 99 AB 08 00 10 02 05 DC 13 03 02   «.ƒ.K™«.....Ü...
22 2B 5D 23 05 F4 45 25 7B 28 59 22 64 01 03 F0   "+]#.ôE%{(Y"d..ð
5E 01 06                                          ^..
```

```
      oct15_incl = 0 (0x0)
    access_point
      num_acc_pt_val = 33 (0x21)
      acc_pt_name_val[0] = 3 (0x3) (length)
      acc_pt_name_val[1] = 105 (0x69) (i)
      acc_pt_name_val[2] = 115 (0x73) (s)
      acc_pt_name_val[3] = 112 (0x70) (p)
      acc_pt_name_val[4] = 5 (0x5) (length)
      acc_pt_name_val[5] = 116 (0x74) (t)
      acc_pt_name_val[6] = 101 (0x65) (e)
      acc_pt_name_val[7] = 108 (0x6c) (l)
      acc_pt_name_val[8] = 117 (0x75) (u)
      acc_pt_name_val[9] = 115 (0x73) (s)
      acc_pt_name_val[10] = 3 (0x3) (length)
      acc_pt_name_val[11] = 99 (0x63) (c)
      acc_pt_name_val[12] = 111 (0x6f) (o)
      acc_pt_name_val[13] = 109 (0x6d) (m)
      acc_pt_name_val[14] = 6 (0x6) (length)
      acc_pt_name_val[15] = 109 (0x6d) (m)
      acc_pt_name_val[16] = 110 (0x6e) (n)
      acc_pt_name_val[17] = 99 (0x63) (c)
      acc_pt_name_val[18] = 50 (0x32) (2)
      acc_pt_name_val[19] = 50 (0x32) (2)
      acc_pt_name_val[20] = 48 (0x30) (0)
      acc_pt_name_val[21] = 6 (0x6) (length)
      acc_pt_name_val[22] = 109 (0x6d) (m)
      acc_pt_name_val[23] = 99 (0x63) (c)
      acc_pt_name_val[24] = 99 (0x63) (c)
      acc_pt_name_val[25] = 51 (0x33) (3)
```

```
0.00 B/s   0.00 B/s   OSR4:Not Connected   0.00 B   10 S   0:0:0.0   125888   0
```

FAQs

Installed drivers do not take effect.

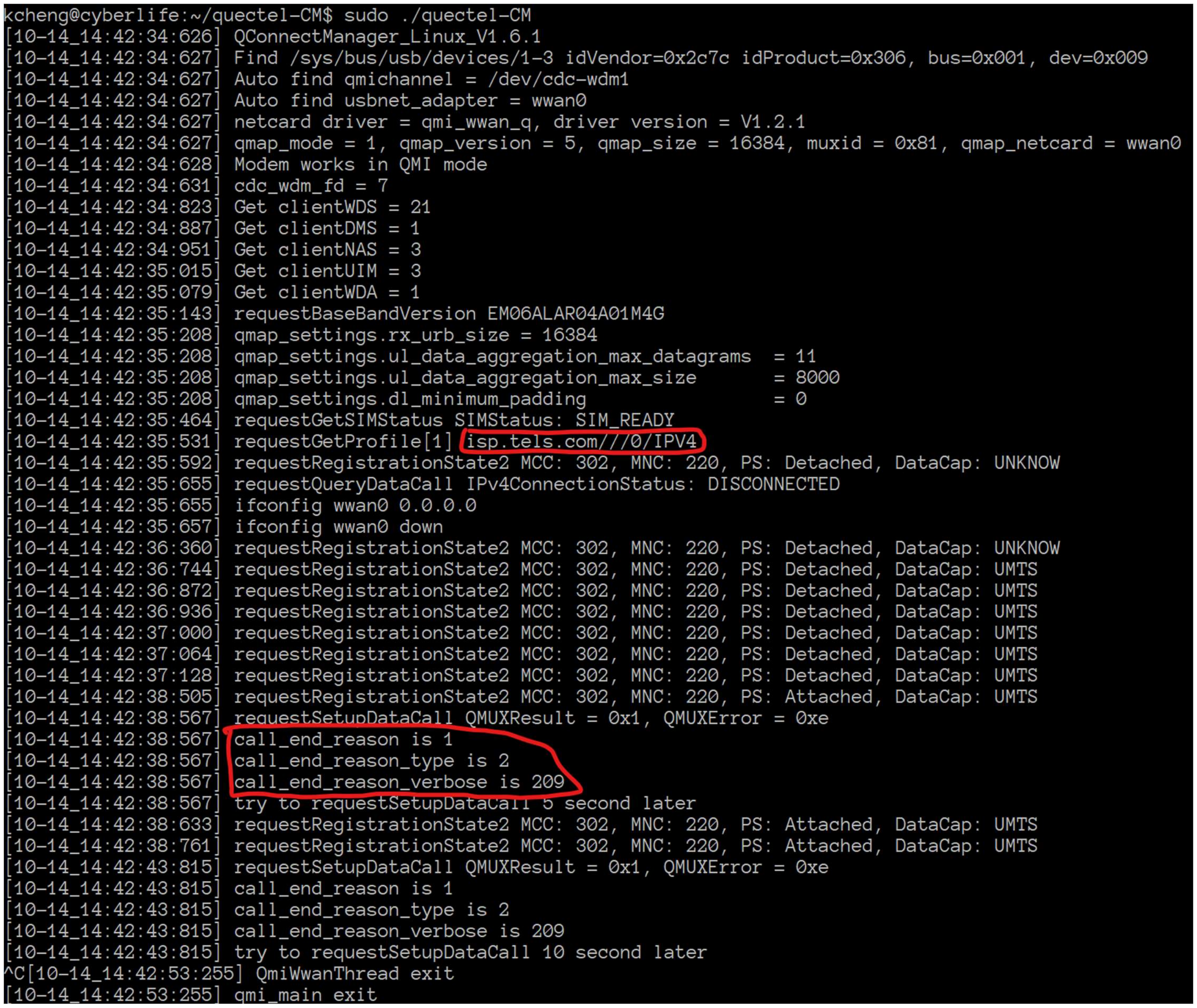
Please refer to the previous chapter "Note on driver installation."

Data call failure

Typo in APN. Update/restore APN, restart module and try again.

https://github.com/abferm/libqmi/blob/master/src/libqmi-glib/qmi-enums-wds.h

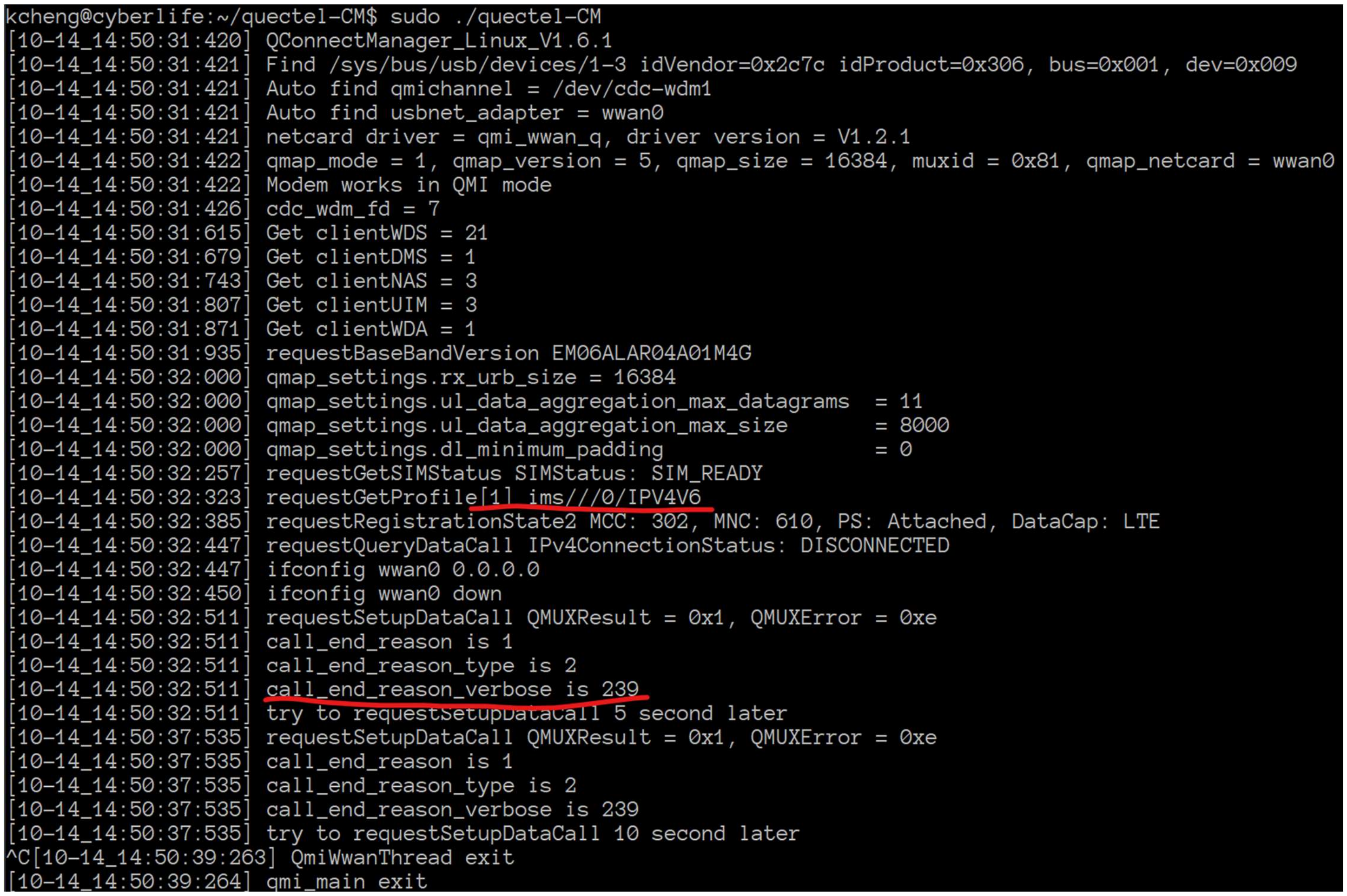
`QMI_WDS_VERBOSE_CALL_END_REASON_INTERNAL_PDN_IPV4_CALL_THROTTLED          = 209`



```
kcheng@cyberlife:~/quectel-CM$ sudo ./quectel-CM
[10-14_14:42:34:626] QConnectManager_Linux_V1.6.1
[10-14_14:42:34:627] Find /sys/bus/usb/devices/1-3 idVendor=0x2c7c idProduct=0x306, bus=0x001, dev=0x009
[10-14_14:42:34:627] Auto find qmichannel = /dev/cdc-wdm1
[10-14_14:42:34:627] Auto find usbnet_adapter = wwan0
[10-14_14:42:34:627] netcard driver = qmi_wwan_q, driver version = V1.2.1
[10-14_14:42:34:627] qmap_mode = 1, qmap_version = 5, qmap_size = 16384, muxid = 0x81, qmap_netcard = wwan0
[10-14_14:42:34:628] Modem works in QMI mode
[10-14_14:42:34:631] cdc_wdm_fd = 7
[10-14_14:42:34:823] Get clientWDS = 21
[10-14_14:42:34:887] Get clientDMS = 1
[10-14_14:42:34:951] Get clientNAS = 3
[10-14_14:42:35:015] Get clientUIM = 3
[10-14_14:42:35:079] Get clientWDA = 1
[10-14_14:42:35:143] requestBaseBandVersion EM06ALAR04A01M4G
[10-14_14:42:35:208] qmap_settings.rx_urb_size = 16384
[10-14_14:42:35:208] qmap_settings.ul_data_aggregation_max_datagrams  = 11
[10-14_14:42:35:208] qmap_settings.ul_data_aggregation_max_size       = 8000
[10-14_14:42:35:208] qmap_settings.dl_minimum_padding                 = 0
[10-14_14:42:35:464] requestGetSIMStatus SIMStatus: SIM_READY
[10-14_14:42:35:531] requestGetProfile[1] isp.tels.com///0/IPV4
[10-14_14:42:35:592] requestRegistrationState2 MCC: 302, MNC: 220, PS: Detached, DataCap: UNKNOW
[10-14_14:42:35:655] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[10-14_14:42:35:655] ifconfig wwan0 0.0.0.0
[10-14_14:42:35:657] ifconfig wwan0 down
[10-14_14:42:36:360] requestRegistrationState2 MCC: 302, MNC: 220, PS: Detached, DataCap: UNKNOW
[10-14_14:42:36:744] requestRegistrationState2 MCC: 302, MNC: 220, PS: Detached, DataCap: UMTS
[10-14_14:42:36:872] requestRegistrationState2 MCC: 302, MNC: 220, PS: Detached, DataCap: UMTS
[10-14_14:42:36:936] requestRegistrationState2 MCC: 302, MNC: 220, PS: Detached, DataCap: UMTS
[10-14_14:42:37:000] requestRegistrationState2 MCC: 302, MNC: 220, PS: Detached, DataCap: UMTS
[10-14_14:42:37:064] requestRegistrationState2 MCC: 302, MNC: 220, PS: Detached, DataCap: UMTS
[10-14_14:42:37:128] requestRegistrationState2 MCC: 302, MNC: 220, PS: Detached, DataCap: UMTS
[10-14_14:42:38:505] requestRegistrationState2 MCC: 302, MNC: 220, PS: Attached, DataCap: UMTS
[10-14_14:42:38:567] requestSetupDataCall QMUXResult = 0x1, QMUXError = 0xe
[10-14_14:42:38:567] call_end_reason is 1
[10-14_14:42:38:567] call_end_reason_type is 2
[10-14_14:42:38:567] call_end_reason_verbose is 209
[10-14_14:42:38:567] try to requestSetupDataCall 5 second later
[10-14_14:42:38:633] requestRegistrationState2 MCC: 302, MNC: 220, PS: Attached, DataCap: UMTS
[10-14_14:42:38:761] requestRegistrationState2 MCC: 302, MNC: 220, PS: Attached, DataCap: UMTS
[10-14_14:42:43:815] requestSetupDataCall QMUXResult = 0x1, QMUXError = 0xe
[10-14_14:42:43:815] call_end_reason is 1
[10-14_14:42:43:815] call_end_reason_type is 2
[10-14_14:42:43:815] call_end_reason_verbose is 209
[10-14_14:42:43:815] try to requestSetupDataCall 10 second later
^C[10-14_14:42:53:255] QmiWwanThread exit
[10-14_14:42:53:255] qmi_main exit
```

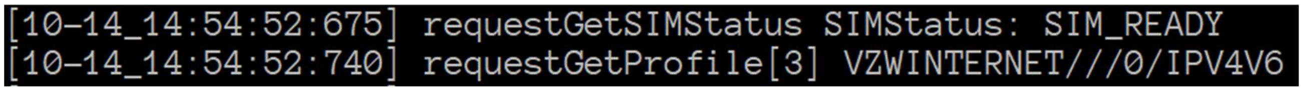
Call_end_reason_verbose is 239, i.e., APN_DISALLOWED_ON_ROAMING

It is a Verizon SIM, but the data call was made without the "-n 3" command line parameter. i.e., incorrectly using default PDP1 for data call. Adding -n 3 to the command line will fix the issue



```
kcheng@cyberlife:~/quectel-CM$ sudo ./quectel-CM
[10-14_14:50:31:420] QConnectManager_Linux_V1.6.1
[10-14_14:50:31:421] Find /sys/bus/usb/devices/1-3 idVendor=0x2c7c idProduct=0x306, bus=0x001, dev=0x009
[10-14_14:50:31:421] Auto find qmichannel = /dev/cdc-wdm1
[10-14_14:50:31:421] Auto find usbnet_adapter = wwan0
[10-14_14:50:31:421] netcard driver = qmi_wwan_q, driver version = V1.2.1
[10-14_14:50:31:422] qmap_mode = 1, qmap_version = 5, qmap_size = 16384, muxid = 0x81, qmap_netcard = wwan0
[10-14_14:50:31:422] Modem works in QMI mode
[10-14_14:50:31:426] cdc_wdm_fd = 7
[10-14_14:50:31:615] Get clientWDS = 21
[10-14_14:50:31:679] Get clientDMS = 1
[10-14_14:50:31:743] Get clientNAS = 3
[10-14_14:50:31:807] Get clientUIM = 3
[10-14_14:50:31:871] Get clientWDA = 1
[10-14_14:50:31:935] requestBaseBandVersion EM06ALAR04A01M4G
[10-14_14:50:32:000] qmap_settings.rx_urb_size = 16384
[10-14_14:50:32:000] qmap_settings.ul_data_aggregation_max_datagrams  = 11
[10-14_14:50:32:000] qmap_settings.ul_data_aggregation_max_size       = 8000
[10-14_14:50:32:000] qmap_settings.dl_minimum_padding                 = 0
[10-14_14:50:32:257] requestGetSIMStatus SIMStatus: SIM_READY
[10-14_14:50:32:323] requestGetProfile[1] ims///0/IPV4V6
[10-14_14:50:32:385] requestRegistrationState2 MCC: 302, MNC: 610, PS: Attached, DataCap: LTE
[10-14_14:50:32:447] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[10-14_14:50:32:447] ifconfig wwan0 0.0.0.0
[10-14_14:50:32:450] ifconfig wwan0 down
[10-14_14:50:32:511] requestSetupDataCall QMUXResult = 0x1, QMUXError = 0xe
[10-14_14:50:32:511] call_end_reason is 1
[10-14_14:50:32:511] call_end_reason_type is 2
[10-14_14:50:32:511] call_end_reason_verbose is 239
[10-14_14:50:32:511] try to requestSetupDataCall 5 second later
[10-14_14:50:37:535] requestSetupDataCall QMUXResult = 0x1, QMUXError = 0xe
[10-14_14:50:37:535] call_end_reason is 1
[10-14_14:50:37:535] call_end_reason_type is 2
[10-14_14:50:37:535] call_end_reason_verbose is 239
[10-14_14:50:37:535] try to requestSetupDataCall 10 second later
^C[10-14_14:50:39:263] QmiWwanThread exit
[10-14_14:50:39:264] qmi_main exit
```

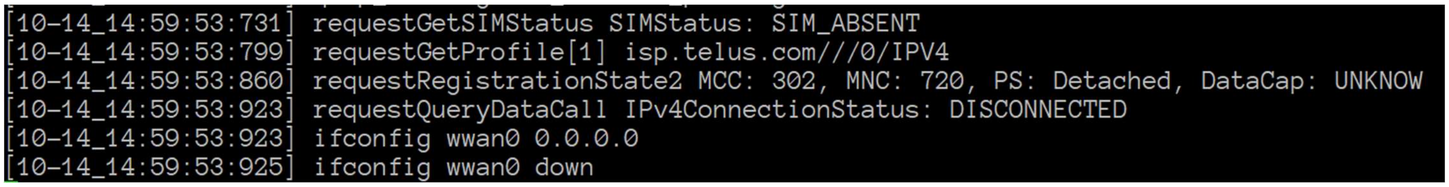
Correct case:



```
[10-14_14:54:52:675] requestGetSIMStatus SIMStatus: SIM_READY
[10-14_14:54:52:740] requestGetProfile[3] VZWINTERNET///0/IPV4V6
```

FAQs cont.

SIM is not detected



```
[10-14_14:59:53:731] requestGetSIMStatus SIMStatus: SIM_ABSENT
[10-14_14:59:53:799] requestGetProfile[1] isp.telus.com///0/IPV4
[10-14_14:59:53:860] requestRegistrationState2 MCC: 302, MNC: 720, PS: Detached, DataCap: UNKNOW
[10-14_14:59:53:923] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[10-14_14:59:53:923] ifconfig wwan0 0.0.0.0
[10-14_14:59:53:925] ifconfig wwan0 down
```

Call end reasons

These are usually APN/SIM account related. Some SIM plans only allow IPV6; please check with the carrier about APN setup and data call restrictions.

It is also possible that another data call is already in progress.

e.g., The module may have connected to a Windows or Linux PC that brings up data calls automatically, and the module is re-connected to another host without a power cycle. The previous data session is kept, and making a new one is not allowed.

```
QMI_WDS_VERBOSE_CALL_END_REASON_INTERNAL_PDN_IPV4_CALL_DISALLOWED    = 208,
QMI_WDS_VERBOSE_CALL_END_REASON_INTERNAL_PDN_IPV4_CALL_THROTTLED     = 209,
QMI_WDS_VERBOSE_CALL_END_REASON_INTERNAL_PDN_IPV6_CALL_DISALLOWED    = 210,
QMI_WDS_VERBOSE_CALL_END_REASON_INTERNAL_PDN_IPV6_CALL_THROTTLED     = 211,
QMI_WDS_CALL_END_REASON_GSM_WCDMA_OPTION_NOT_SUPPORTED               = 1017,
QMI_WDS_CALL_END_REASON_GSM_WCDMA_OPTION_UNSUBSCRIBED                = 1018,
```

Traditional network interface names like eth0

You may see random-looking names like ens33 on newer systems. It is a new behavior introduced by systemd.
Please refer to this link if you prefer old-style names like I do.
https://www.itzgeek.com/how-tos/mini-howtos/change-default-network-name-ens33-to-old-eth0-on-ubuntu-16-04.html

SIOCDELRT: No such process

It is an expected behavior. /etc/udhcpc/default.script deletes all existing default routes with a loop, and the system reports this error message when there is no more default route to delete. We can safely ignore it.

Custom DNS server address

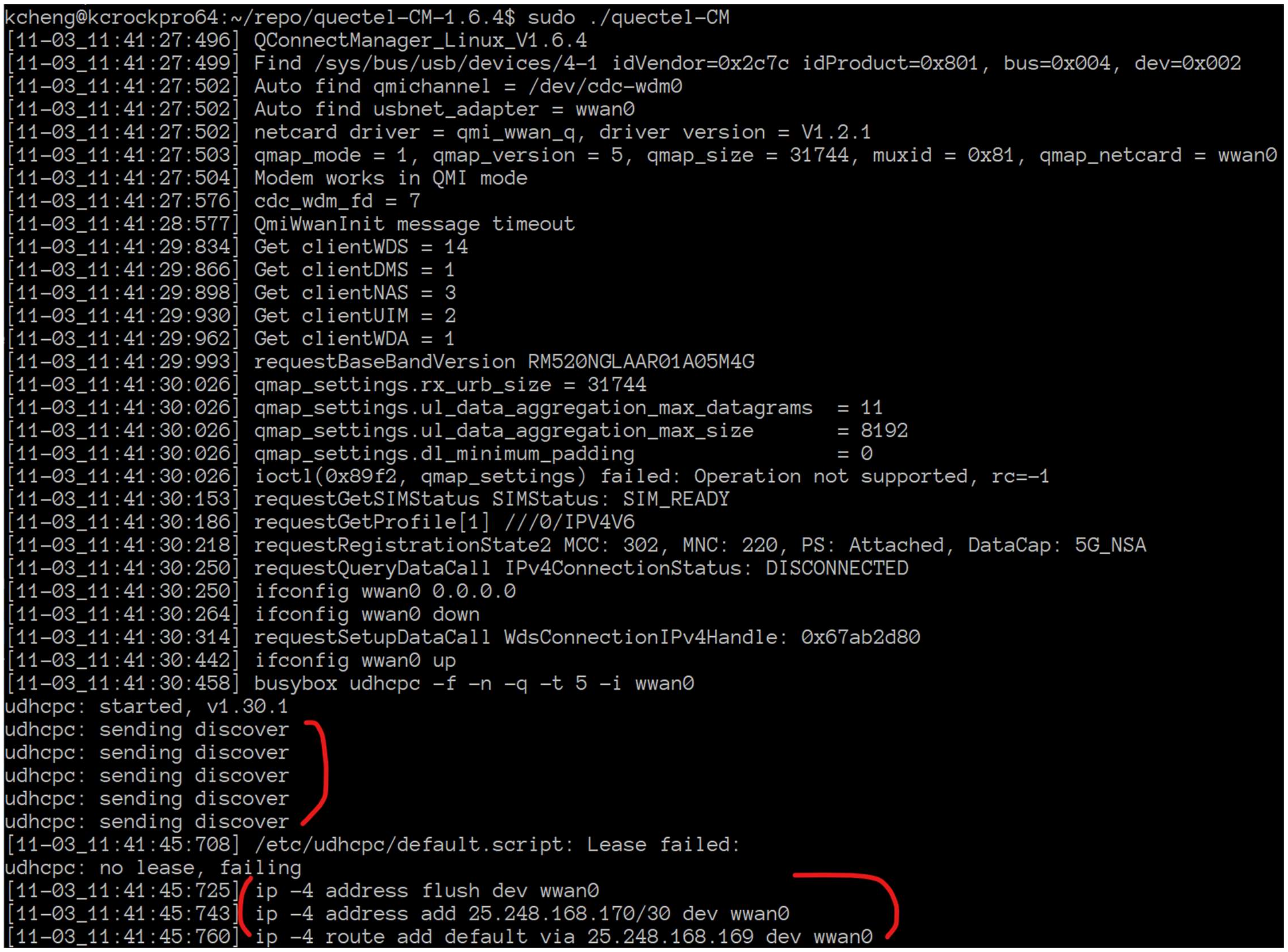
https://wiki.archlinux.org/title/systemd-resolved

DHCP retries

Reason 1: qmi_wwan was used instead of qmi_wwan_q. quectel-CM will enable RAW IP after the first DHCP attempt fails. Please consider switching to qmi_wwan_q

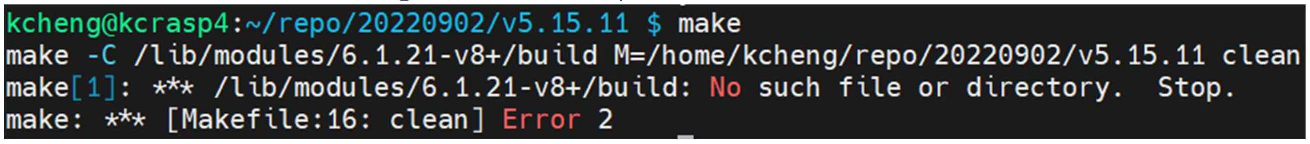
Reason 2: qmi_wwan_q 1.2.1 is buggy. Please update to 1.2.2 or newer

Reason 3: pcie_mhi driver issue. Please update to 1.3.4 or newer.



```
kcheng@kcrockpro64:~/repo/quectel-CM-1.6.4$ sudo ./quectel-CM
[11-03_11:41:27:496] QConnectManager_Linux_V1.6.4
[11-03_11:41:27:499] Find /sys/bus/usb/devices/4-1 idVendor=0x2c7c idProduct=0x801, bus=0x004, dev=0x002
[11-03_11:41:27:502] Auto find qmichannel = /dev/cdc-wdm0
[11-03_11:41:27:502] Auto find usbnet_adapter = wwan0
[11-03_11:41:27:502] netcard driver = qmi_wwan_q, driver version = V1.2.1
[11-03_11:41:27:503] qmap_mode = 1, qmap_version = 5, qmap_size = 31744, muxid = 0x81, qmap_netcard = wwan0
[11-03_11:41:27:504] Modem works in QMI mode
[11-03_11:41:27:576] cdc_wdm_fd = 7
[11-03_11:41:28:577] QmiWwanInit message timeout
[11-03_11:41:29:834] Get clientWDS = 14
[11-03_11:41:29:866] Get clientDMS = 1
[11-03_11:41:29:898] Get clientNAS = 3
[11-03_11:41:29:930] Get clientUIM = 2
[11-03_11:41:29:962] Get clientWDA = 1
[11-03_11:41:29:993] requestBaseBandVersion RM520NGLAAR01A05M4G
[11-03_11:41:30:026] qmap_settings.rx_urb_size = 31744
[11-03_11:41:30:026] qmap_settings.ul_data_aggregation_max_datagrams  = 11
[11-03_11:41:30:026] qmap_settings.ul_data_aggregation_max_size       = 8192
[11-03_11:41:30:026] qmap_settings.dl_minimum_padding                 = 0
[11-03_11:41:30:026] ioctl(0x89f2, qmap_settings) failed: Operation not supported, rc=-1
[11-03_11:41:30:153] requestGetSIMStatus SIMStatus: SIM_READY
[11-03_11:41:30:186] requestGetProfile[1] ///0/IPV4V6
[11-03_11:41:30:218] requestRegistrationState2 MCC: 302, MNC: 220, PS: Attached, DataCap: 5G_NSA
[11-03_11:41:30:250] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[11-03_11:41:30:250] ifconfig wwan0 0.0.0.0
[11-03_11:41:30:264] ifconfig wwan0 down
[11-03_11:41:30:314] requestSetupDataCall WdsConnectionIPv4Handle: 0x67ab2d80
[11-03_11:41:30:442] ifconfig wwan0 up
[11-03_11:41:30:458] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.30.1
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
[11-03_11:41:45:708] /etc/udhcpc/default.script: Lease failed:
udhcpc: no lease, failing
[11-03_11:41:45:725] ip -4 address flush dev wwan0
[11-03_11:41:45:743] ip -4 address add 25.248.168.170/30 dev wwan0
[11-03_11:41:45:760] ip -4 route add default via 25.248.168.169 dev wwan0
```

Driver build error about missing files or directory



```
kcheng@kcrasp4:~/repo/20220902/v5.15.11 $ make
make -C /lib/modules/6.1.21-v8+/build M=/home/kcheng/repo/20220902/v5.15.11 clean
make[1]: *** /lib/modules/6.1.21-v8+/build: No such file or directory.  Stop.
make: *** [Makefile:16: clean] Error 2
```

Reason 1: Did you install kernel header?

Reason 2: Did you specify the CPU architecture? See page 22.

Reason 3: Kernel is 64 bit but the installed kernel header is 32 bit. See Page 3.