# Quectel USB driver installation Of Raspberry OS

Date：Sep. 26, 2021

Written by：Bakewell.Wang

# Content

# 1. Background

Many customers develop quectel module based on the embedded linux platform, which requires the linux os to install the module driver in advance. One of the methods is to compile and install the kernel source code. This article will use raspberry 4B to explain cross-compilation and installation of quectel USB driver.

# 2. Environment setup

- Host machine ---ubuntu 16.04
- Cross compilation tool chain
- Raspberry latest linux kernel source code(based on linux kernel 5.9)

## 2.1 Host PC Install Required Dependencies

```
sudo apt install git bc bison flex libssl-dev make libc6-dev libncurses5-dev
```

```
ol@ql-Ubuntu:~/work$ sudo apt install git bc bison flex libssl-dev make libc6-dev libncurses5-dev
[sudo] password for ol:
Reading package lists... Done
Building dependency tree
Reading state information... Done
bc is already the newest version (1.06.95-9build1).
bison is already the newest version (2:3.0.4.dfsg-1).
flex is already the newest version (2.6.0-11).
libncurses5-dev is already the newest version (6.0+20160213-1ubuntu1).
make is already the newest version (4.1-6).
git is already the newest version (1:2.7.4-0ubuntu1.10).
libc6-dev is already the newest version (2.23-0ubuntu11.3).
You might want to run 'apt-get -f install' to correct these:
The following packages have unmet dependencies:
 debconf-i18n : Depends: debconf (= 1.5.58ubuntu2) but 1.5.58ubuntu1 is to be installed
 libssl-dev : Depends: libssl1.0.0 (= 1.0.2g-1ubuntu4.20) but 1.0.2g-1ubuntu4.19 is to be installed
E: Unmet dependencies. Try 'apt-get -f install' with no packages (or specify a solution).
```

The figure above shows that the related dependencies have been installed.

BTW, Customers may use different embedded Linux platforms. Usually the manufacturer will provide the corresponding build dependencies. In addition, when you fail to execute the compilation, the command line will also prompt which dependencies are missing.

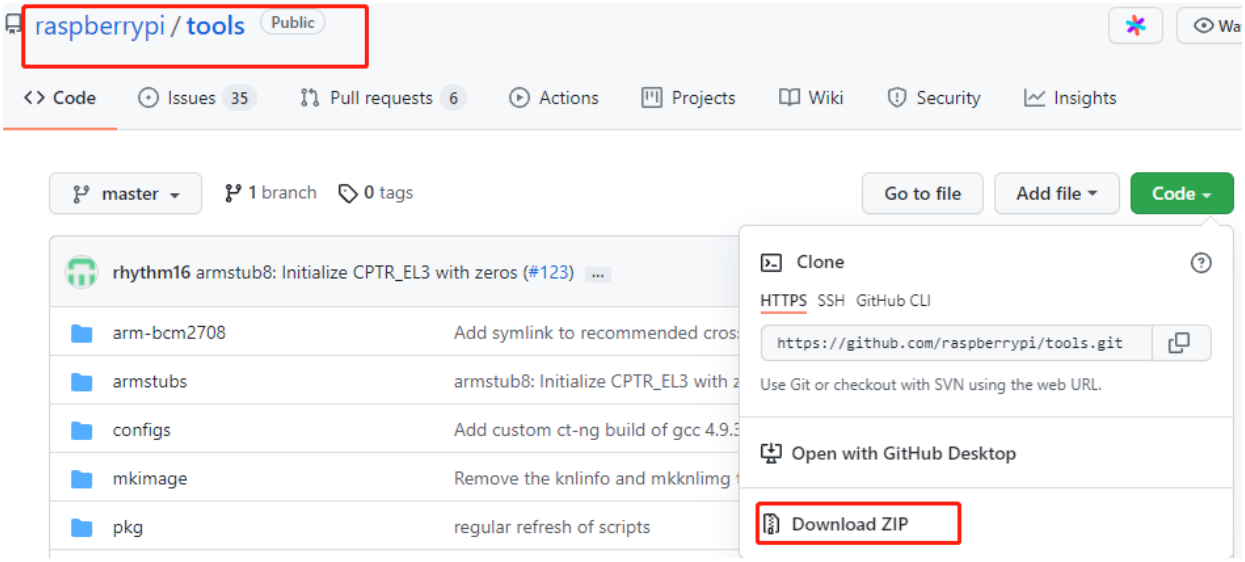
## 2.2 Install cross compilation tool chain

Download toolchain from Raspberry's official Git.

```
git clone https://github.com/raspberrypi/tools
```

Due to git clone fail ,here I enter the link to download the zip package directly:

```
ol@ql-Ubuntu:~/work/raspberry$ git clone https://github.com/raspberrypi/tools
Cloning into 'tools'...

fatal: unable to access 'https://github.com/raspberrypi/tools/': Could not resolve host: github.com
```




Next, the toolchain needs to be added to the system environment variables so that it can be accessed globally.

```
echo PATH=\$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin >> ~/.bashrc
source ~/.bashrc  //enable set
vi ~/.bashrc  //check setting
```

If the setting is successful, will add the toolchain path to the .bashrc file endline.

```
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
PATH=$PATH:/opt/tools-master/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/

PATH=$PATH:~/gcc-linaro/bin
```

```
ol@ql-Ubuntu:~$ arm
arm2hpdl                        arm-linux-gnueabihf-gcc-4.8.3       arm-linux-gnueabihf-ld.gold
arm-linux-gnueabihf-addr2line   arm-linux-gnueabihf-gcc-ar          arm-linux-gnueabihf-nm
arm-linux-gnueabihf-ar          arm-linux-gnueabihf-gcc-nm          arm-linux-gnueabihf-objcopy
arm-linux-gnueabihf-as          arm-linux-gnueabihf-gcc-ranlib      arm-linux-gnueabihf-objdump
arm-linux-gnueabihf-c++         arm-linux-gnueabihf-gcov            arm-linux-gnueabihf-pkg-config
arm-linux-gnueabihf-c++filt     arm-linux-gnueabihf-gdb             arm-linux-gnueabihf-pkg-config-real
arm-linux-gnueabihf-cpp         arm-linux-gnueabihf-gfortran        arm-linux-gnueabihf-ranlib
arm-linux-gnueabihf-dwp         arm-linux-gnueabihf-gprof           arm-linux-gnueabihf-readelf
arm-linux-gnueabihf-elfedit     arm-linux-gnueabihf-ld              arm-linux-gnueabihf-size
arm-linux-gnueabihf-g++         arm-linux-gnueabihf-ld.bfd          arm-linux-gnueabihf-strings
arm-linux-gnueabihf-gcc         arm-linux-gnueabihf-ldd             arm-linux-gnueabihf-strip
```

### 2.3 Get the Kernel Sources and configure the kernel

To download the minimal source tree for the current branch, run:

```
git clone --depth=1 https://github.com/raspberrypi/linux
```

Ener kernel source code dir and confgs:

```
ol@ql-Ubuntu:~/work/raspberry/rpi-5.9.y$ KERNEL=kernel8 \
> make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- bcm2711_defconfig
#
# No change to .config
#
ol@ql-Ubuntu:~/work/raspberry/rpi-5.9.y$
```

Note: due raspberry pi 4 supports 64-bit architecture, so build 64-bit version, in fact, different embedded linux platforms have corresponding reference settings.

## 3. Modify the kernel source code of the usb driver

### 3.1 USB Serial Option Driver

**Add VID and PID**

```
vi driver/usb/serial/options.c
```

```
static const struct usb_device_id option_ids[] = {
#if 1//Added by Quectel
        {USB_DEVICE(0x2C7C,0x0125)},/*Quectel EC20 R2.0/EC20 R2.1/EC25/EG25-G/EM05*/
        {USB_DEVICE(0x2C7C,0x0121)},/*EC21/EG21-G*/
        {USB_DEVICE(0x2C7C,0x0191)},/*EG91*/
        {USB_DEVICE(0x2C7C,0x0195)},/*EG95*/
        {USB_DEVICE(0x2C7C,0x0306)},/*EG06/EP06/EM06*/
        {USB_DEVICE(0x2C7C,0x0512)},/*EG12/EM12/EG18*/
        {USB_DEVICE(0x2C7C,0x0296)},/*BG96*/
        {USB_DEVICE(0x2C7C,0x0700)},/*BG95/BG77/BG600L-M3/BC69*/
        {USB_DEVICE(0x2C7C,0x0435)},/*AG35*/
        {USB_DEVICE(0x2C7C,0x0415)},/*AG15*/
        {USB_DEVICE(0x2C7C,0x0452)},/*AG520R*/
        {USB_DEVICE(0x2C7C,0x0455)},/*AG550R*/
        {USB_DEVICE(0x2C7C,0x0620)},/*EG20*/
        {USB_DEVICE(0x2C7C,0x0800)},/*RG500Q/RM500Q/RG510Q/RM510Q*/

#endif
```

In order to recognize the module, the module's VID and PID information as below need to be added to the file [KERNEL]/drivers/usb/serial/option.c.

### Add the Zero Packet Mechanism

As required by the USB protocol, the mechanism for processing zero packets needs to be added during bulk-out transmission by adding the following statements.

```
static struct urb *usb_wwan_setup_urb(struct usb_serial_port *port,
                                      int endpoint,
                                      int dir, void *ctx, char *buf, int len,
                                      void (*callback) (struct urb *))
{
        struct usb_serial *serial = port->serial;
        struct usb_wwan_intf_private *intfdata = usb_get_serial_data(serial);
        struct urb *urb;

        urb = usb_alloc_urb(0, GFP_KERNEL);     /* No ISO */
        if (!urb)
                return NULL;

        usb_fill_bulk_urb(urb, serial->dev,
                          usb_sndbulkpipe(serial->dev, endpoint) | dir,
                          buf, len, callback, ctx);
        if (/*intfdata->use_zlp &&*/ dir == USB_DIR_OUT){//Added by Quectel for zero packet
                struct usb_device_descriptor *desc= &serial->dev->descriptor;
                if(desc->idVendor == cpu_to_le16(0x2C7C))
                        urb->transfer_flags |= URB_ZERO_PACKET;
        }

        return urb;
```

### Add Reset-resume Mechanism

Some USB host controllers/USB hubs will lose power or be reset when MCU enters the Suspend/Sleep mode, and cannot be used for USB resume after MCU exits from the Suspend/Sleep mode. The reset-resume mechanism needs to be enabled by adding the following statements.

```
static struct usb_serial_driver option_1port_device = {
        .driver = {
                .owner =        THIS_MODULE,
                .name =         "option1",
        },
        .description       = "GSM modem (1-port)",
        .id_table          = option_ids,
        .num_ports         = 1,
        .probe             = option_probe,
        .open              = usb_wwan_open,
        .close             = usb_wwan_close,
        .dtr_rts           = usb_wwan_dtr_rts,
        .write             = usb_wwan_write,
        .write_room        = usb_wwan_write_room,
        .chars_in_buffer   = usb_wwan_chars_in_buffer,
        .tiocmget          = usb_wwan_tiocmget,
        .tiocmset          = usb_wwan_tiocmset,
        .get_serial        = usb_wwan_get_serial_info,
        .set_serial        = usb_wwan_set_serial_info,
        .attach            = option_attach,
        .release           = option_release,
        .port_probe        = usb_wwan_port_probe,
        .port_remove       = usb_wwan_port_remove,
        .read_int_callback = option_instat_callback,
#ifdef CONFIG_PM
        .suspend           = usb_wwan_suspend,
        .resume            = usb_wwan_resume,
#endif
        .reset_resume      = usb_wwan_resume,   // add by quectel
};
```

### Use MBIM, GobiNet or QMI_WWAN Driver

If MBIM, GobiNet or QMI_WWAN driver is required, add the following statements to prevent the module's interface 4 from being used as a USB serial device

```
static int option_probe(struct usb_serial *serial,
                        const struct usb_device_id *id)
{
        struct usb_interface_descriptor *iface_desc =
                                &serial->interface->cur_altsetting->desc;
        unsigned long device_flags = id->driver_info;

        /* Never bind to the CD-Rom emulation interface */
        if (iface_desc->bInterfaceClass == USB_CLASS_MASS_STORAGE)
                return -ENODEV;

        /*
         * Don't bind reserved interfaces (like network ones) which often have
         * the same class/subclass/protocol as the serial interfaces.  Look at
         * the Windows driver .INF files for reserved interface numbers.
         */
        if (device_flags & RSVD(iface_desc->bInterfaceNumber))
                return -ENODEV;

        /*
         * Allow matching on bNumEndpoints for devices whose interface numbers
         * can change (e.g. Quectel EP06).
         */
        if (device_flags & NUMEP2 && iface_desc->bNumEndpoints != 2)
                return -ENODEV;
#if 1 //Added by Quectel
      //Quectel modules's interface 4 can be used as USB network device
      if(serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)){
        //some interface can be used as USB Network device(ecm,rndis,mbim)
        if(serial->interface->cur_altsetting->desc.bInterfaceClass != 0xFF){
                return -ENODEV;
        }
        //interface 4 can be used as USB Network device(qmi)
        else if(serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4){
                return -ENODEV;
```

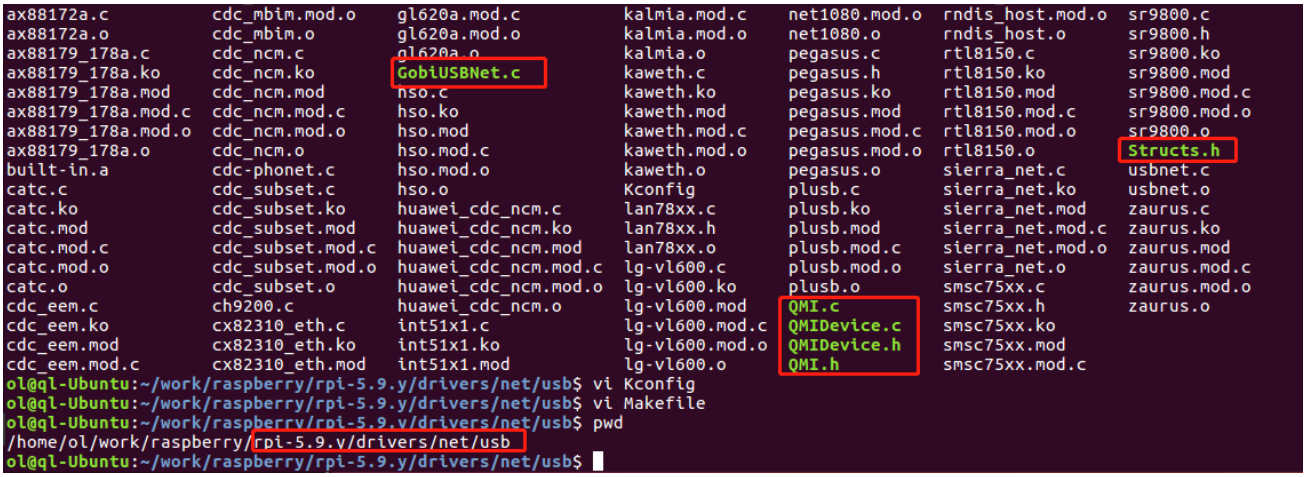
## 3.2 Add GobiNet Driver

When the GobiNet driver has been installed in the module, a network device and a QMI channel will be created. The network device is named as ethX (usbX if the kernel version is 2.6.39 or lower) and the QMI channel is /dev/qcqmiX. The network device is used for data transmission, and QMI channel is used for QMI message interaction.

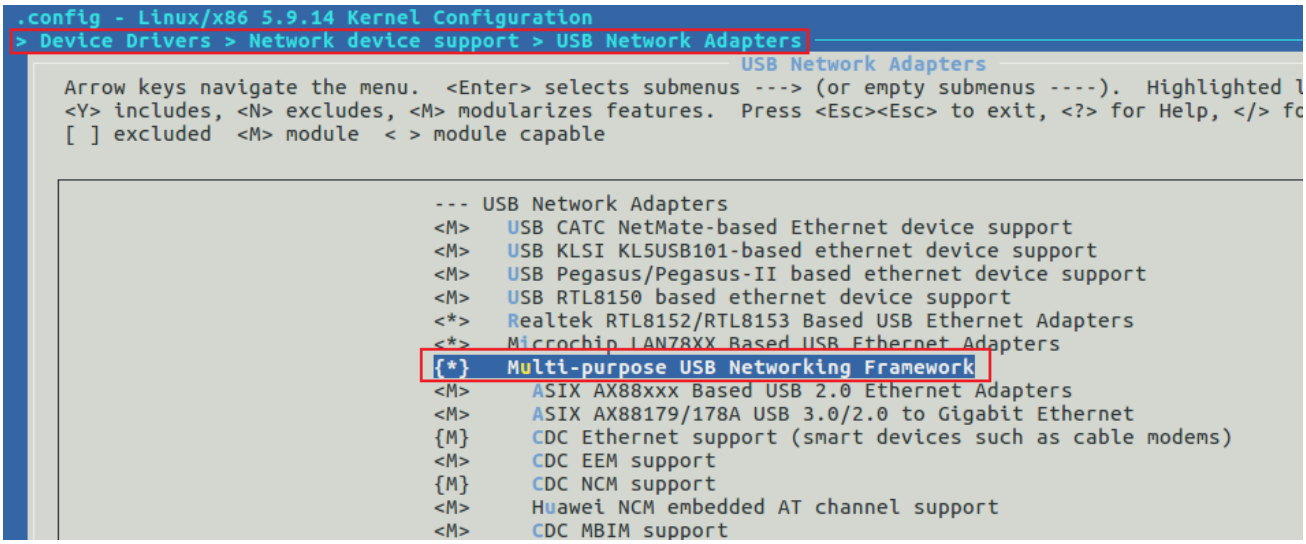
## Modify Source Codes of the Driver

The GobiNet driver is provided by Quectel in the form of the source file containing source codes.

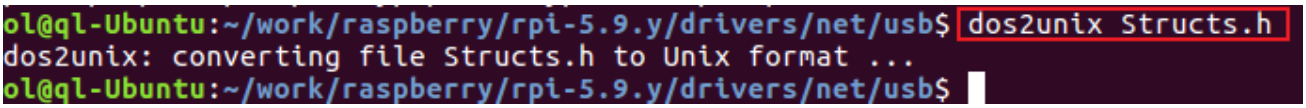
The source file should be copied to the file [KERNEL]/drivers/net/usb/ (or [KERNEL]/drivers/usb/net/ if the kernel version is lower than 2.6.22)




## Modify Kernel Configuration




Note: Copy file to linux from windows, you need to convert the file format



## 3.3 Add QMI_WWAN Driver

The source file containing source codes of QMI_WWAN driver is [KERNEL]/drivers/net/usb/qmi_wwan.c.
In order to use the QMI_WWAN driver along with the Quectel module, the source file needs certain modification.
To simplify works, Quectel provides the source file qmi_wwan_q.c, which can coexist with qmi_wwan.c and only be used for Quectel's modules. The source file qmi_wwan_q.c should be copied to the file [KERNEL]/drivers/net/usb/.

```
cd [KERNEL]/drivers/net/usb
vi Makefile
```

```
obj-$(CONFIG_USB_VL600)          += lg-vl600.o
#must insert qmi_wwan_q.o before qmi_wwan.o
obj-$(CONFIG_USB_NET_QMI_WWAN)   += qmi_wwan_q.o
obj-$(CONFIG_USB_NET_QMI_WWAN)   += qmi_wwan.o
obj-$(CONFIG_USB_NET_CDC_MBIM)   += cdc_mbim.o
obj-$(CONFIG_USB_NET_CH9200)     += ch9200.o
obj-$(CONFIG_USB_NET_AQC111)     += aqc111.o
obj-y +=GobiNet.o
GobiNet-objs := GobiUSBNet.o QMIDevice.o QMI.o
~
```

```
cd [KERNEL]
make menuconfig
```

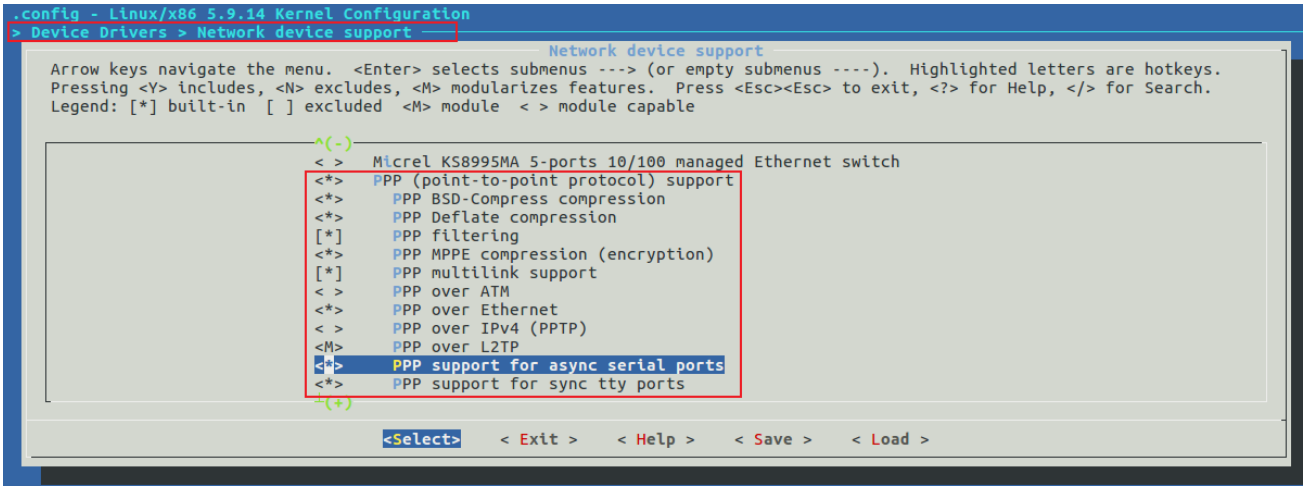
```
.config - Linux/x86 5.9.14 Kernel Configuration
> Device Drivers > Network device support > USB Network Adapters
                        USB Network Adapters
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are hotkeys.
  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                        ^(-)
                        <M>   Simple USB Network Links (CDC Ethernet subset)
                        [*]     ALi M5632 based 'USB 2.0 Data Link' cables
                        [*]     AnchorChips 2720 based cables (Xircom PGUNET, ...)
                        [*]     eTEK based host-to-host cables (Advance, Belkin, ...)
                        [*]     Embedded ARM Linux links (iPaq, ...)
                        [*]     Epson 2888 based firmware (DEVELOPMENT)
                        [*]     KT Technology KC2190 based cables (InstaNet)
                        <M>   Sharp Zaurus (stock ROMs) and compatible
                        <M>   Conexant CX82310 USB ethernet port
                        <M>   Samsung Kalmia based LTE USB modem
                        <M>   QMI WWAN driver for Qualcomm MSM based 3G and LTE modems
                        <M>   Option USB High Speed Mobile Devices
                        <M>   Intellon PLC based usb adapter
                        v(+)

                              <Select>    < Exit >    < Help >    < Save >    < Load >
```

Note: GobiNet and qmi_wwan drivers can be installed at the same time.

## 3.4 Configure Kernel to Support PPP

If PPP function is used, please follow the steps and the corresponding commands below to configure the kernel to support PPP.

```
cd [KERNEL]
make menuconfig
```




At this point, the relevant configuration has been completed.

# 4. Compile kernel and install the image

## 4.1 Configure the kernel and compile

### Configure the kernel

```
KERNEL=kernel7l
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- bcm2711_defconfig
```

```
ol@ql-Ubuntu:~/work/raspberry/rpi-5.9.y$ KERNEL=kernel8
ol@ql-Ubuntu:~/work/raspberry/rpi-5.9.y$ make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- bcm2711_defconfig
#
# configuration written to .config
#
ol@ql-Ubuntu:~/work/raspberry/rpi-5.9.y$
```

### Compile kernel

```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- Image modules dtbs
```

This process will last a long time, depends on the power of the CPU,
To speed up compilation on multiprocessor systems, and get some improvement on single processor ones, use , where n is the number of processors * 1.5. Alternatively, feel free to experiment and see

what works! -j n.
In addition, If there is a compilation error, you need to further check with error info.

## 4.2 Install Image

Take the SD card out of the Raspberry Pi and connect it to the Host machine.
On my Ubuntu machine, the two partitions in the SD card, boot and rootfs will be automatically mounted.

```
ol@ql-Ubuntu:~/work/raspberry/rpi-5.9.y$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdb      8:16   1  14.9G  0 disk
├─sdb2   8:18   1  14.6G  0 part /media/ol/rootfs
└─sdb1   8:17   1   256M  0 part /media/ol/boot
sr0     11:0    1  1024M  0 rom
sda      8:0    0   400G  0 disk
├─sda4   8:4    0   2.1G  0 part [SWAP]
├─sda2   8:2    0 117.4G  0 part /
├─sda3   8:3    0   1.1G  0 part /boot
└─sda1   8:1    0 279.4G  0 part /home
ol@ql-Ubuntu:~/work/raspberry/rpi-5.9.y$
```

**install the kernel modules onto the SD card**

```
sudo env PATH=$PATH make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- INSTALL_MOD_PATH=/media/ol/rootfs modules_install
```

```
ol@ql-Ubuntu:~/work/raspberry/rpi-5.9.y$ sudo env PATH=$PATH make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- INSTALL_MOD_PATH=/media/ol/rootfs modules_install
  INSTALL arch/arm64/crypto/aes-arm64.ko
  INSTALL arch/arm64/crypto/aes-neon-blk.ko
  INSTALL arch/arm64/crypto/aes-neon-bs.ko
  INSTALL arch/arm64/crypto/chacha-neon.ko
```

Note:

1. In fact, for installing the Raspberry Pi OS, SD has already formatted the partition in advance.
2. For the embedded linux platform used by the customer, the manufacturer will provide SDK for building and installing Image.

**copy the kernel and Device Tree blobs onto the SD card, making sure to back up your old kernel:**

```
sudo cp /media/bryan/boot/$KERNEL.img /media/ol/boot/$KERNEL-backup.img
sudo cp arch/arm64/boot/Image /media/ol/boot/$KERNEL.img
sudo cp arch/arm64/boot/dts/broadcom/*.dtb /media/ol/boot
sudo cp arch/arm64/boot/dts/overlays/*.dtb* /media/ol/boot/overlays/
sudo cp arch/arm64/boot/dts/overlays/README /media/ol/boot/overlays/
```

Note: $KERNEL is a temporary environment variable, closing the current terminal will be invalid

Finally, plug the card into the Pi and boot it!

# 5. Test the module

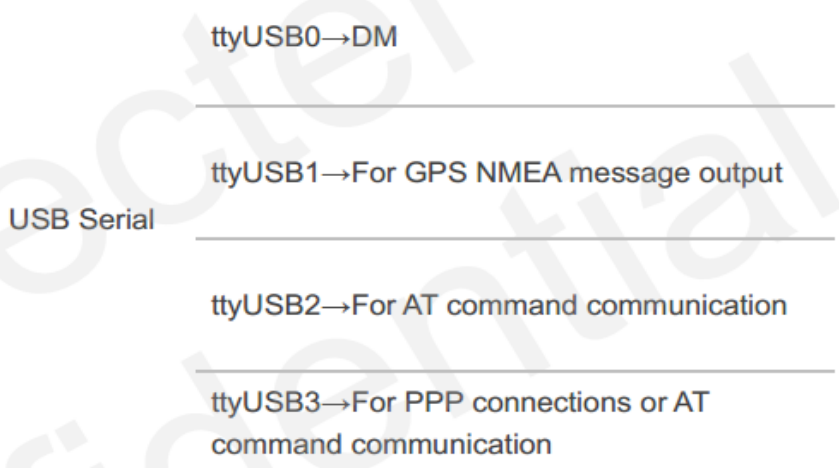
Generally, AT and PPP functions are supported. If GobiNet or QMI_WWAN driver has been installed, the USB network adapter function can also be used on the module. The following chapters explain how to test these functions, Take EC21-E as an example:

```
dmesg //check driver loading status
```

```
[  862.504459] usb 1-1.2: new high-speed USB device number 6 using xhci_hcd
[  862.614507] usb 1-1.2: New USB device found, idVendor=2c7c, idProduct=0121, bcdDevice= 3.18
[  862.614523] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[  862.614536] usb 1-1.2: Product: Android
[  862.614548] usb 1-1.2: Manufacturer: Android
[  862.625251] option 1-1.2:1.0: GSM modem (1-port) converter detected
[  862.626408] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB0
[  862.626816] option 1-1.2:1.1: GSM modem (1-port) converter detected
[  862.628760] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB1
[  862.629234] option 1-1.2:1.2: GSM modem (1-port) converter detected
[  862.629571] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB2
[  862.630008] option 1-1.2:1.3: GSM modem (1-port) converter detected
[  862.630295] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB3
[  862.636682] qmi_wwan_q 1-1.2:1.4: cdc-wdm0: USB WDM device
[  862.638184] qmi_wwan_q 1-1.2:1.4: Quectel Android work on RawIP mode
[  862.639178] qmi_wwan_q 1-1.2:1.4: rx_urb_size = 1520
[  862.639828] qmi_wwan_q 1-1.2:1.4 wwan0: register 'qmi_wwan_q' at usb-0000:01:00.0-1.2, WWAN/QMI device, ce:22:d9:e5:91:ad
pi@raspberrypi:~$
```

It can be seen that the actual use here is only driven by qmi_wwan.

Note: I tried to install them at the same time, but in fact only qmi_wwan is displayed, but there is no qcqmix or cdc-wdmX in /dev, so qmi dialing cannot be performed



## 5.1 Test AT Function

The AT port is usually /dev/ttyUSB2, which is the second ttyUSB port created by the USB serial option driver.

For example:

Busybox microcom /dev/ttyUSB2

```
pi@raspberrypi:~$ busybox microcom /dev/ttyUSB2

+CREG: 1,"691D","6908430",7

+CGREG: 1,"691D","6908430",7

+CEREG: 1,"691D","6908430",7
```

## 5.2 Test PPP Function

Other network cards are recommended to be used such as GobiNet or QMI_WWAN for dialing instead of PPP. PPP is more complex to use than others, will cause CPU overloaded and provide small maximum download speed.

sudo ./quectel-pppd.sh

```
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.139.85.173> <ms-dns1 61.132.163.68> <ms-dns2 202.102.213.68>]
sent [IPCP ConfReq id=0x2 <addr 10.139.85.173> <ms-dns1 61.132.163.68> <ms-dns2 202.102.213.68>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfAck id=0x2 <addr 10.139.85.173> <ms-dns1 61.132.163.68> <ms-dns2 202.102.213.68>]
Could not determine remote IP address: defaulting to 10.64.64.64
not replacing default route to wwan0 [0.0.0.0]
local  IP address 10.139.85.173
remote IP address 10.64.64.64
primary   DNS address 61.132.163.68
secondary DNS address 202.102.213.68
Script /etc/ppp/ip-up started (pid 1374)
Script /etc/ppp/ip-up finished (pid 1374), status = 0x0
```

```
ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.139.85.173  netmask 255.255.255.255  destination 10.64.64.64
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 4  bytes 52 (52.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 58 (58.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Note: Regarding the PPP dialing procedure, pls refer "linux-ppp-scripts".

### 5.2 Test qmi_wwan call Function

```
sudo ./quectel-CM
```

```
pi@raspberrypi:~/willie/quectel-CM$ sudo ./quectel-CM
[09-25_11:26:04:963] Quectel_QConnectManager_Linux_V1.6.0.16
[09-25_11:26:04:965] Find /sys/bus/usb/devices/1-1.2 idVendor=0x2c7c idProduct=0x121, bus=0x001, dev=0x007
[09-25_11:26:04:965] Auto find qmichannel = /dev/cdc-wdm0
[09-25_11:26:04:965] Auto find usbnet_adapter = wwan0
[09-25_11:26:04:965] netcard driver = qmi_wwan_q, driver version = V1.2.0.23
[09-25_11:26:04:966] ioctl(0x89f3, qmap_settings) failed: Operation not supported, rc=-1
[09-25_11:26:04:966] Modem works in QMI mode
[09-25_11:26:04:978] cdc_wdm_fd = 7
[09-25_11:26:05:075] Get clientWDS = 20
[09-25_11:26:05:108] Get clientDMS = 1
[09-25_11:26:05:139] Get clientNAS = 4
[09-25_11:26:05:171] Get clientUIM = 1
[09-25_11:26:05:204] Get clientWDA = 1
[09-25_11:26:05:236] requestBaseBandVersion EC21EFAR06A04M4G
[09-25_11:26:05:365] requestGetSIMStatus SIMStatus: SIM_READY
[09-25_11:26:05:397] requestGetProfile[1] ctnet///1
[09-25_11:26:05:428] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[09-25_11:26:05:460] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[09-25_11:26:05:460] ifconfig wwan0 0.0.0.0
[09-25_11:26:05:469] ifconfig wwan0 down
[09-25_11:26:05:524] requestSetupDataCall WdsConnectionIPv4Handle: 0x872e5590
[09-25_11:26:05:653] ifconfig wwan0 up
[09-25_11:26:05:661] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.30.1
Too few arguments.
Too few arguments.
udhcpc: sending discover
udhcpc: sending select for 10.139.85.173
udhcpc: lease of 10.139.85.173 obtained, lease time 7200
[09-25_11:26:05:882] /etc/udhcpc/default.script: Resetting default routes
SIOCDELRT: No such process
[09-25_11:26:05:891] /etc/udhcpc/default.script: Adding DNS 61.132.163.68
```

## 6. Install and Load Driver as a Kernel Module for PC in Linux

For developers requiring to test Quectel modules on PC with Linux operating system like Ubuntu, Quectel can provide source files of USB serial option/GobiNet/QMI_WWAN drivers. These USB drivers can be installed and used by using the following commands and then rebooting the PC.

Based on local PC (ubuntu) compiler QMI_WWAN driver:

```
sudo ./make install
```

```
ol@ql-Ubuntu:~/quec/qmi_wwan$ make
make ARCH=x86_64 CROSS_COMPILE= -C /lib/modules/4.15.0-142-generic/build M=/home/ol/quec/qmi_wwan modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-142-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-142-generic'
ol@ql-Ubuntu:~/quec/qmi_wwan$ file qmi_wwan_q.ko
qmi_wwan_q.ko: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), BuildID[sha1]=c48468d10c626677353b73bc9d33c6bbfe134865, not stripped
ol@ql-Ubuntu:~/quec/qmi_wwan$
```

Based on raspberry kernel Cross compile QMI_WWAN driver on Ubuntu:

```
ol@ql-Ubuntu:~/quec/qmi_wwan$ make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- -C /home/ol/work/raspberry/rpi-5.9.y M=/home/ol/quec/qmi_wwan modules
make[1]: Entering directory '/home/ol/work/raspberry/rpi-5.9.y'
  CC [M]  /home/ol/quec/qmi_wwan/qmi_wwan_q.o
  MODPOST /home/ol/quec/qmi_wwan/Module.symvers
  CC [M]  /home/ol/quec/qmi_wwan/qmi_wwan_q.mod.o
  LD [M]  /home/ol/quec/qmi_wwan/qmi_wwan_q.ko
make[1]: Leaving directory '/home/ol/work/raspberry/rpi-5.9.y'
ol@ql-Ubuntu:~/quec/qmi_wwan$ file qmi_wwan_q.ko
qmi_wwan_q.ko: ELF 64-bit LSB relocatable, ARM aarch64, version 1 (SYSV), BuildID[sha1]=094620e9b22e6b4f33a89033cda2ad58d6d61cd8, not stripped
ol@ql-Ubuntu:~/quec/qmi_wwan$
```

Note:Cross compiling needs to specify the corresponding kernel source directory.

Manually load the driver：

```
ol@ql-Ubuntu:~/quec/qmi_wwan$ sudo insmod qmi_wwan_q.ko
insmod: ERROR: could not insert module qmi_wwan_q.ko: File exists
ol@ql-Ubuntu:~/quec/qmi_wwan$ sudo rmmod qmi_wwan_q
ol@ql-Ubuntu:~/quec/qmi_wwan$ sudo insmod qmi_wwan_q.ko
ol@ql-Ubuntu:~/quec/qmi_wwan$ dmesg
```

insmod loads the driver, rmmod unloads the driver, it shows that it has been loaded before.

```
[  840.374232] qmi_wwan_q 1-1:1.4 wwp0s11u1i4: unregister 'qmi_wwan_q' usb-0000:00:0b.0-1, WWAN/QMI device
[  843.946468] qmi_wwan_q 1-1:1.4: cdc-wdm0: USB WDM device
[  843.946470] qmi_wwan_q 1-1:1.4: Quectel EC25&EC21&EG91&EG95&EG06&EP06&EM06&EG12&EP12&EM12&EG16&EG18&BG96&AG35 work on RawIP mode
[  843.951794] qmi_wwan_q 1-1:1.4: rx_urb_size = 1520
[  843.952001] qmi_wwan_q 1-1:1.4 wwan0: register 'qmi_wwan_q' at usb-0000:00:0b.0-1, WWAN/QMI device, 76:53:d9:90:a5:17
[  843.952336] usbcore: registered new interface driver qmi_wwan_q
[  843.955088] qmi_wwan_q 1-1:1.4 wwp0s11u1i4: renamed from wwan0
```

For manually loaded drivers, system restarts need to be manually reloaded

Note: modprobe also used to load the driver, but different from insmod:

1. Insmod can only load a specific device driver at a time, and the specific address of the driver is required. Written as:

insmod drv.ko

2. modprobe can load all dependent drivers into the kernel at one time. The specific address of the driver is not added, but the driver module needs to be installed in the way of make modues_install when installing the file system. The driver is installed under /lib/modules/$(uname -r)/... Written as:

modprob drv