Linux Drivers Overview

Daniele Palmas

The Linux way

- As a general rule, Telit relies on drivers released with the kernel sources: if needed these drivers are modified for supporting Telit modems

- If a driver has been modified for supporting a new modem, the most convenient way to ensure a long-lasting support is submitting the change to the kernel mantainers to be accepted for future kernel releases

- Given the huge number of kernel versions and architectures Telit does not release binary, nor source code: the customer should rely on common kernel releases. If needed the appropriate driver should be modified according to Telit generic instructions

What characterizes a device

- VID / PID: Vendor Id, Product Id
- Telit Vendor Id is 0x1bc7 (7111)
- The product id depends on the modem family (e.g. UE910v2 PID is 0x1012). Different modem families that have the same PID, have also the same features at the USB level (e.g. LE920 and LE910).
- USB descriptors spread at various levels
- USB interfaces. Important descriptors of an interface are: number of endpoints, interface class, subclass and protocol
- Endpoints. Important descriptors of an endpoint are: address, type, max packet size
- The standard: http://www.usb.org/developers/docs/usb20_docs/#usb20spec
- An easier approach: http://www.beyondlogic.org/usbnutshell/usb1.shtml

Identifying the Chipset

- Telit modems are based on two different family of chipsets: Qualcomm and Infineon
- Usually chipsets of the same family uses a common USB architecture that means also a common driver
- The first time a driver for a specific modem is to be chosen, the chipset family should be identified and its USB descriptors should be analyzed
- - Usually two kinds of devices are emulated through USB:
  - Serial port
  - Network card

Infineon

Serial port

- For serial port emulation Telit Infineon based modems follow CDC ACM standard

- CDC ACM: Communication Device Class, Abstract Control Model

- Standard: http://www.usb.org/developers/docs/devclass_docs/CDC1.2_WMC1.1_012011.zip

- In a few word each emulated serial port is created using:

  1 interface with an interrupt endpoint
  1 interface with two bulk endpoints

- The interrupt endpoint is used for receiving unsolicited information from the modem (e.g. modem state)

- The bulk endpoints are used for data transfer (e.g. AT commands)

cdc-acm

- For devices following CDC ACM standard in Linux the driver cdc-acm is used
- For recent kernel releases the source code can be found in /drivers/usb/class
- For recent kernel there is no need to modify the driver since it automatically binds itself to a device which has the proper CDC ACM USB descriptors
- For older kernel the device VID / PID should be added to the list of devices supported by the driver
- As a general advice it is better to build the driver as a kernel module (except for Android devices)
- Device created by the drivers are called ttyACMx (where x depends on the modem and on the configuration of the host)
- Examples of Telit modem driven by cdc-acm: HE910, UE910

Network card

- Network card emulation is supported only in a firmware variant for HE910

- This firmware variant follows CDC ECM (Communication Device Class, Ethernet Control Model)

- Standard:
http://www.usb.org/developers/docs/devclass_docs/CDC1.2_WMC1.1_012011.zip

- In a few word an emulated network card is created using:

  1 interface with an interrupt endpoint
  1 interface with two bulk endpoints

- The interrupt endpoint is used for receiving unsolicited information from the modem (e.g. network connection status)

- The bulk endpoints are used for data transfer

cdc_ether

- For devices following CDC ECM standard in Linux the driver cdc_ether is used
- For recent kernel releases the source code can be found in /drivers/net/usb
- For recent kernel there is no need to modify the driver since it automatically binds itself to a device which has the proper CDC ECM USB descriptors
- As a general advice it is better to build the driver as a kernel module (except for Android devices)
- Network card created by the driver is called ethx or wwanx (where x depends on the configuration of the host). The different name depends on the kernel version

Flashing driver

- Infineon based modems for supporting firmware upgrade in Linux need a special driver
- Telit released tltflash for kernel version 3.0: the driver creates the device ttyTLT that can be used by the lxfp application for upgrading the firmware
- At the moment no other kernel versions are supported: in the most recent kernel releases minor modification could be needed for proper working
- It is possible that the cdc-acm driver should be modified for not catching the flashing device presented by the modem

Qualcomm

Serial port

- For serial port emulation Telit Qualcomm based modems follow a modified version of the CDC ACM standard (sometimes called reduced ACM)
- The main difference is found in the USB presentation. Each emulated serial port is created using only one interface with one interrupt and two bulk endpoints
- The interrupt endpoint is used for receiving unsolicited information from the modem (e.g. modem state)
- The bulk endpoints are used for data transfer (e.g. AT commands)

option

- For devices following Qualcomm implementation of ACM in Linux the driver option is used
- For recent kernel releases the source code can be found in /drivers/usb/serial
- According to the modem family it is possible that the source code of the driver has to be modified for adding the proper VID / PID
- It is possible to add a modem at runtime

  echo <vid> <pid> > /sys/bus/usb-serial/drivers/option1/new_id
- As a general advice it is better to build the driver as a kernel module (except for Android devices)
- Device created by the drivers are called ttyUSBx (where x depends on the modem and on the configuration of the host)
- Examples of Telit modem driven by option: UE910v2, DE910, LE920

Network card

- Network card emulation is supported in LE9x0 families
- It seems not to explicitly follow any specific standard
- Network card management is done through a vendor specific management protocol called QMI. QMI can be used to replace the more common AT commands over serial interface management (except for non standard operations)
- An emulated network card is created using one interface with one interrupt endpoints and two bulk. QMI messages passes on the control endpoint
- It seems that the interrupt endpoint is used for signaling that a QMI message is waiting to be read in the control endpoint
- The bulk endpoints are used for data transfer

qmi_wwan

- Since kernel version 3.4 the driver qmi_wwan is used for Qualcomm modems that use QMI control management
- For recent kernel releases the source code can be found in /drivers/net/usb
- For recent kernel there is no need to modify the driver since LE9x0 VID / PID have already been inserted
- As a general advice it is better to build the driver as a kernel module (except for Android devices)
- The management device created by the driver is called cdc-wdm
- - Network card created by the driver is called wwanx (where x depends on the configuration of the host)

Gobi driver

- LE9x0 can be driven also by a different set of drivers coming from the Code Aurora project
- The Code Aurora project is sponsored by Qualcomm, but Qualcomm does not officially support these drivers
- GobiSerial: for emulated serial ports
- GobiNet: for emulated network card
- These drivers were never accepted in the kernel mainline due to quality reasons
- GobiNet driver can also be used for older kernel versions (2.6.x)

QMI in userspace

- Due to the QMI protocol complexity, modem management with QMI messages is usually done with helping libraries
- There are two common choices: libQMI and GobiSDK
- libQMI: http://www.freedesktop.org/wiki/Software/libqmi/
- GobiSDK: project Code Aurora
- Both the solutions have many requirements, so choosing the correct one depends on the customer's system

RIL (Android, Windows CE)

Firmware flashing

lxfp

- For modems that support the xfp flashing protocol, the Linux application lxfp can be used for flashing a firmware
- For Infineon devices a special flashing driver is needed
- For Qualcomm devices the driver option can be used
- A c++ toolchain is needed
- lxfp supports also Android
- If a modem does not support the xfp flashing protocol, then at the moment there is no other way for flashing a firmware

Debugging common problems

- **Check that the modem is seen by Linux at USB level**

lsusb

Bus 002 Device 002: ID 058f:6362 Alcor Micro Corp. Flash Card Reader/Writer
Bus 004 Device 002: ID 413c:2003 Dell Computer Corp. Keyboard
Bus 004 Device 003: ID 0461:4d51 Primax Electronics, [Dell Laser Mouse Kit]
Bus 002 Device 006: ID 1bc7:1201

- **Check that the proper driver has been loaded**

If a driver is built as a module lsmod

Module	Size	Used by
GobiSerial	12904	0
usbserial	47107	1 GobiSerial
GobiNet	50507	0
usbnet	26212	1 GobiNet

- **Kernel output**

dmesg

```
[79948.492071] usbcore: registered new interface driver usbserial
[79948.492083] USB Serial support registered for generic
[79948.606696] creating qcqmi0
[79948.606871] usbcore: registered new interface driver usbserial_generic
[79948.606874] usbserial: USB Serial Driver core
[79948.607189] USB Serial support registered for GobiSerial
[79948.613583] GobiSerial 2-1:1.0: GobiSerial converter detected
[79948.613703] usb 2-1: GobiSerial converter now attached to ttyUSB0
[79948.614567] GobiSerial 2-1:1.4: GobiSerial converter detected
[79948.614645] usb 2-1: GobiSerial converter now attached to ttyUSB1
[79948.615316] GobiSerial 2-1:1.5: GobiSerial converter detected
[79948.615383] usb 2-1: GobiSerial converter now attached to ttyUSB2
[79948.615421] usbcore: registered new interface driver GobiSerial
[79948.615423] GobiSerial: Q0.00.00
[79948.615430] usbcore: registered new interface driver GobiNet
```

- **Check for the device presence**

ls /dev/tty*

ifconfig

- **If no answer for AT commands check that the correct ports are used**

minicom -D /dev/ttyUSBx

or

cat /dev/ttyUSBx &
echo -ne “AT\r” > /dev/ttyUSBx

pppd issues

- **Check the pppd option file**
- Check the various chat scripts
- **Check the syslog**