Gobi2000™ Linux Installation User Guide

80-VP531-1 D

April 29, 2010

Submit technical questions at:
https://support.cdmatech.com

Qualcomm Confidential and Proprietary

Restricted Distribution. Not to be distributed to anyone who is not an employee of either Qualcomm or a subsidiary of Qualcomm without the express approval of Qualcomm's Configuration Management.

Not to be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm.

Qualcomm reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed for any damages arising directly or indirectly by their use or application. The information provided in this document is provided on an "as is" basis.

This document contains Qualcomm confidential and proprietary information and must be shredded when discarded.

QUALCOMM is a registered trademark of QUALCOMM Incorporated in the United States and may be registered in other countries. Other product and brand names may be trademarks or registered trademarks of their respective owners. CDMA2000 is a registered certification mark of the Telecommunications Industry Association, used under license. ARM is a registered trademark of ARM Limited. QDSP is a registered trademark of QUALCOMM Incorporated in the United States and other countries.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
U.S.A.

Contents

Figures

Tables

1 Introduction

NOTE Numerous changes were made to this document. It should be read in its entirety.

1.1 Purpose

The Gobi2000™ software package is based on the MDM2000™ ASIC. The Gobi™ platform offers support for UMTS, HSPA, CDMA 1xEV-DO Rev A, and GSM/GPRS/EDGE technologies through the use of downloadable technology-specific images.

This document provides instructions for installing the software on a Linux operating system.

1.2 Scope

This document is intended for internal users, carriers, and notebook OEMs (nOEMs) who are either evaluating Gobi or using modules programmed with Qualcomm NV configuration during the initial stages of Gobi integration.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Code variables appear in angle brackets, e.g., `<number>`.

Commands and command variables appear in a different font, e.g., **`copy`** ***`a:*.* b:`***.

Shading indicates content that has been added or changed in this revision of the document.

1.4 Revision history

The revision history for this document is shown in Table 1-1.

Table 1-1 Revision history

Version	Date	Description
A	May 2009	Initial release
B	Oct 2009	Added drivers information; made additional engineering updates
C	Mar 2010	Added EULA screen and description in Section 2.2; added and corrected carrier information in Table 2-1; added details about downloading the QCSerial2k*.c file and the Makefile to Section 2.3.4.
D	Apr 2010	Numerous changes were made to this document. It should be read in its entirety.

1.5 References

Reference documents, which may include QUALCOMM®, standards, and resource documents, are listed in Table 1-2. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 1-2 Reference documents and standards

Ref.	Document	
QUALCOMM		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1

1.6 Technical assistance

For assistance or clarification on information in this guide, submit a case to Qualcomm CDMA Technologies at https://support.cdmatech.com/.

If you do not have access to the CDMATech Support Service website, register for access or send email to support.cdmatech@qualcomm.com.

1.7 Acronyms

For definitions of terms and abbreviations, see [Q1]. The following terminology is specific to this document:

- udev – A constantly running daemon that manages devices for Linux 2.6 kernel series
  - Manages device nodes in /dev
  - Handles the /dev directory
  - Handles all automated user space actions within adding/removing devices, including firmware load

- udev rules – Flexible and very powerful configuration files for udev
    - Renames a device node from the default name to something else
    - Provides an alternative/persistent name for a device node by creating a symbolic link to the default device node
    - Names a device node based on the output of a program
    - Changes permissions and ownership of a device node
    - Launches a script when a device node is created or removed (typically when a device is attached or unplugged)
    - Renames network interfaces
    - Creates the device node with the default name supplied by the kernel, even if there are no matching rules

  For more information on udev rules, see http://www.reactivated.net/writing_udev_rules.html.

uname – Short for UNIX name; a software command in UNIX and UNIX-like operating systems that prints system details about the machine and OS running on it on screen

- uname –r – Returns the kernel version
- uname –m – Returns the machine architecture (i.e., i686,x64_86)

- .DEB files – Extension for Debian software packages
    - Software package refers to files packaged in an archive format to be installed by a package management system or a self-sufficient installer.
    - Debian packages are standard UNIX archives that include two gzipped or bzipped tar archives, one that holds control information and one that holds data information.
    - A canonical program for handling these packages is dpkg, most commonly via apt/aptitude.
    - Debian packages can be converted into other packages, and vice versa, using Alien.
- .rpm files – Extension for Red Hat software packages; originally developed by Red Hat for Red Hat Linux, but now used by many Linux distributions
- .spec file – Recipe for creating an RPM package
    - Multiple RPM packages can be built from a single .spec file.
    - RPM packages are built from .spec files by using the rpmbuild tool.

2 Qualcomm Software Installation – Closed Source

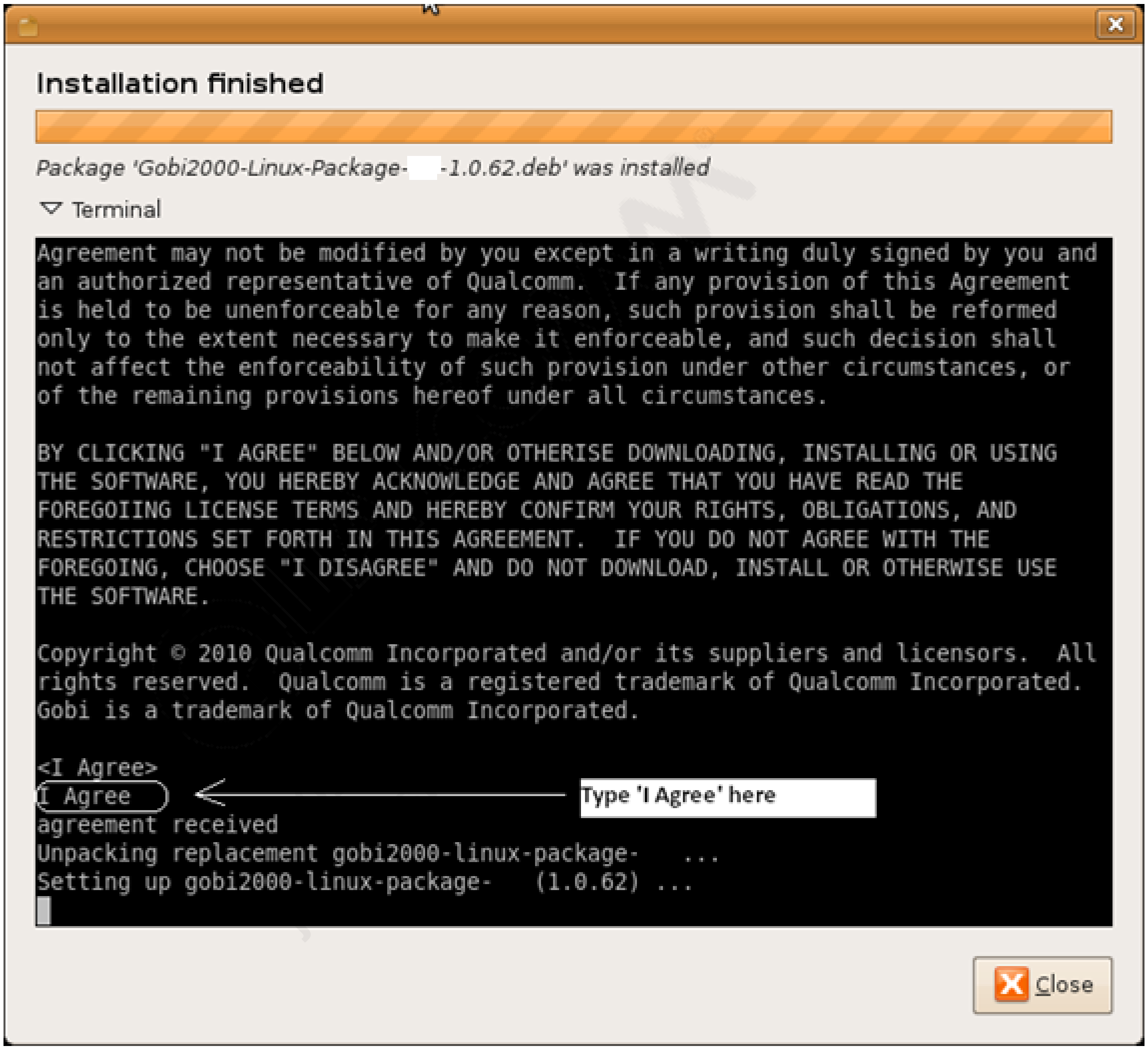
2.1 Software deliverables

Qualcomm provides the Gobi2000 software in two parts. One part is proprietary software and the other is open-source. The proprietary portion consists of the QDL Service and the carrier images. The open-source portion consists of the USB driver and its associated Makefile.

2.2 QDL service and images installation

Install the software contained in the Gobi Linux Package (1.0.03 or later) by running the installer Gobi2000-Linux-Package-<nOEM>-x.y.zz.rpm/deb in the GUI. Here, x, y, and z give the version number of the installer. The GUI-based installation is part of the RPM or DEB package managers that ship with most releases of Linux. The shell commands dpkg -i <debfilename> for DEB packages, or rpm -i <rpmfilename> for RPM packages can also be used for installation.

From Gobi Linux Package 1.0.50 onward, the installer presents an End User License Agreement (EULA) screen to the user. The user is required to type **I Agree** to accept the EULA to install the Linux package. The EULA screen is shown in Figure 2-1.



Installation finished

Package 'Gobi2000-Linux-Package-[redacted]-1.0.62.deb' was installed

```
Agreement may not be modified by you except in a writing duly signed by you and
an authorized representative of Qualcomm.  If any provision of this Agreement
is held to be unenforceable for any reason, such provision shall be reformed
only to the extent necessary to make it enforceable, and such decision shall
not affect the enforceability of such provision under other circumstances, or
of the remaining provisions hereof under all circumstances.

BY CLICKING "I AGREE" BELOW AND/OR OTHERISE DOWNLOADING, INSTALLING OR USING
THE SOFTWARE, YOU HEREBY ACKNOWLEDGE AND AGREE THAT YOU HAVE READ THE
FOREGOIING LICENSE TERMS AND HEREBY CONFIRM YOUR RIGHTS, OBLIGATIONS, AND
RESTRICTIONS SET FORTH IN THIS AGREEMENT.  IF YOU DO NOT AGREE WITH THE
FOREGOING, CHOOSE "I DISAGREE" AND DO NOT DOWNLOAD, INSTALL OR OTHERWISE USE
THE SOFTWARE.

Copyright © 2010 Qualcomm Incorporated and/or its suppliers and licensors.  All
rights reserved.  Qualcomm is a registered trademark of Qualcomm Incorporated.
Gobi is a trademark of Qualcomm Incorporated.

<I Agree>
I Agree
agreement received
Unpacking replacement gobi2000-linux-package-   ...
Setting up gobi2000-linux-package-   (1.0.62) ...
```

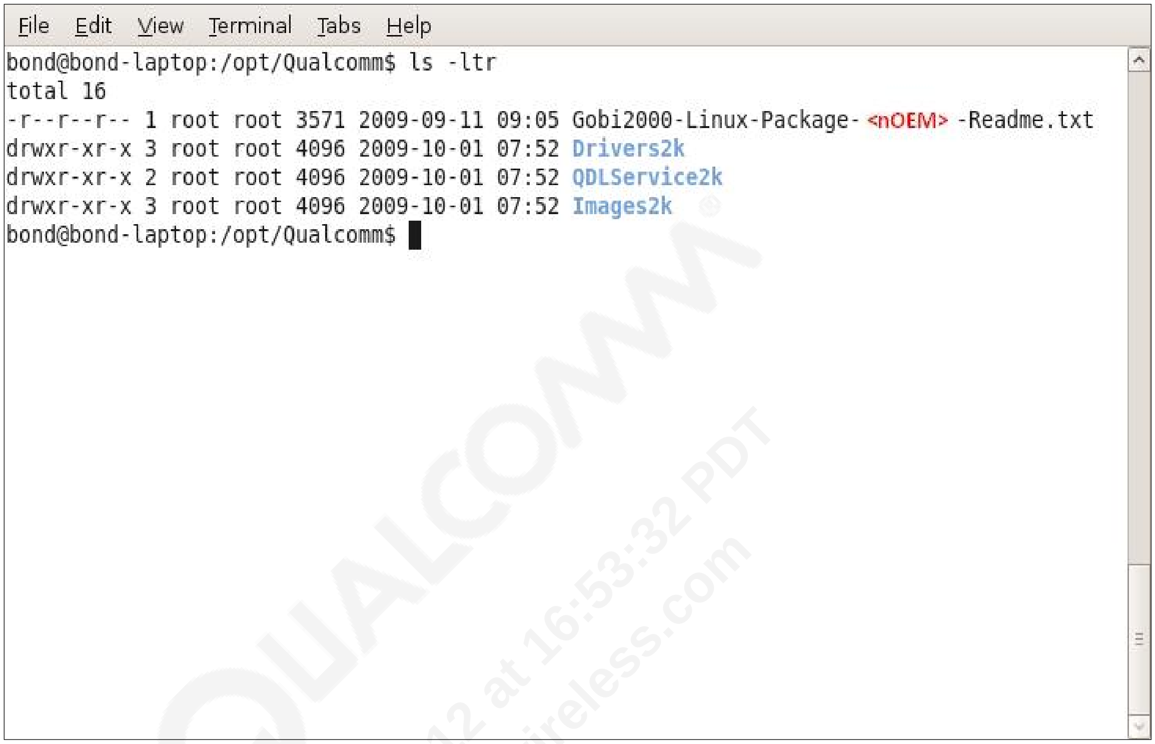
Type 'I Agree' here

Figure 2-1 EULA screen

The Gobi Linux package contains QDL service, specific carrier images, and QCQMI drivers. The default locations of QDL service, the images, the QCQMI drivers, and the object file are:

- QDL service – /opt/Qualcomm/QDLService2k
- Images – /opt/Qualcomm/Images2k
- Drivers – /opt/Qualcomm/Drivers2k

Figure 2-2 illustrates Gobi2000 installed folders.



```
bond@bond-laptop:/opt/Qualcomm$ ls -ltr
total 16
-r--r--r-- 1 root root 3571 2009-09-11 09:05 Gobi2000-Linux-Package- <nOEM> -Readme.txt
drwxr-xr-x 3 root root 4096 2009-10-01 07:52 Drivers2k
drwxr-xr-x 2 root root 4096 2009-10-01 07:52 QDLService2k
drwxr-xr-x 3 root root 4096 2009-10-01 07:52 Images2k
bond@bond-laptop:/opt/Qualcomm$ 
```

Figure 2-2 Installed folders/files

NOTE This package does *not* contain the open-source USB driver and must be installed prior to installing the USB driver.

Download the installation package from Documents and Downloads at https://support.cdmatech.com.

- /Documents and Downloads/Gobi/<Customer> Software Code/Gobi 2000 Linux SW: Installer/GOBI2000_LINUX_PACKAGE<xxxx>, where xxxx is the version number

2.2.1 Images2k

The carrier images for each nOEM are located in \opt\Qualcomm\Images2k\<nOEM>. Each nOEM is provided with multiple carrier images, based on their customized requirement.

Every carrier image has three parts: amss.mbn, apps.mbn, and uqcn.mbn.

The UMTS carriers share the same amss.mbn and apps.mbn. However, carrier-specific changes are present in uqcn.mbn. The uqcn.mbn distinguishes one UMTS carrier from another.

Each CDMA carrier has a unique amss.mbn, apps.mbn, and uqcn.mbn.

UMTS carrier images are located as follows:

- /opt/Qualcomm/Images2k/<nOEM>/UMTS/amss.mbn
- /opt/Qualcomm/Images2k/<nOEM>/UMTS/apps.mbn
- /opt/Qualcomm/Images2k/<nOEM>/#/uqcn.mbn

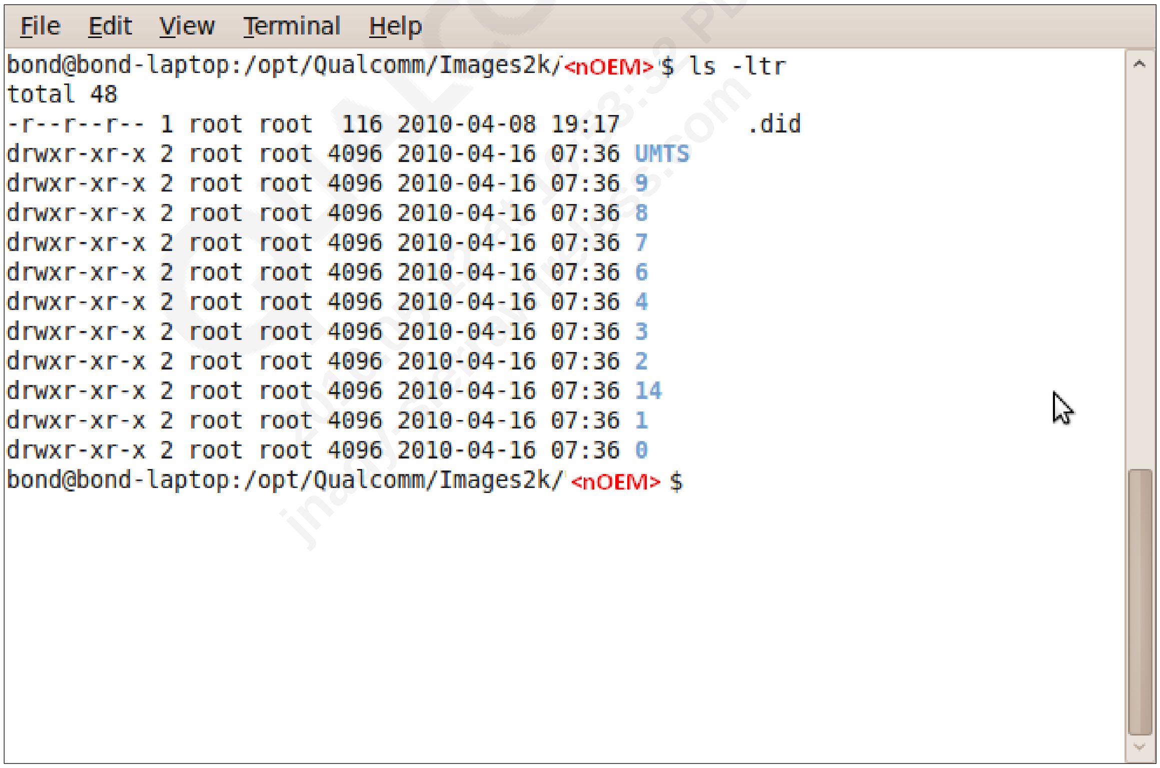
# indicates the number assigned for a specific carrier.

CDMA carrier images are located as follows:

- /opt/Qualcomm/Images2k/<nOEM>/#/amss.mbn
- /opt/Qualcomm/Images2k/<nOEM>/#/apps.mbn
- /opt/Qualcomm/Images2k/<nOEM>/#/uqcn.mbn

# indicates the number assigned for a specific carrier.

Figure 2-3 illustrates the Images2k installed folders.



```
bond@bond-laptop:/opt/Qualcomm/Images2k/<nOEM>$ ls -ltr
total 48
-r--r--r-- 1 root root  116 2010-04-08 19:17        .did
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 UMTS
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 9
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 8
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 7
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 6
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 4
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 3
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 2
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 14
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 1
drwxr-xr-x 2 root root 4096 2010-04-16 07:36 0
bond@bond-laptop:/opt/Qualcomm/Images2k/ <nOEM> $
```

Figure 2-3 Installed Images2k folders

Table 2-1 lists the folder mapping.

Table 2-1 Folder mapping of carrier images

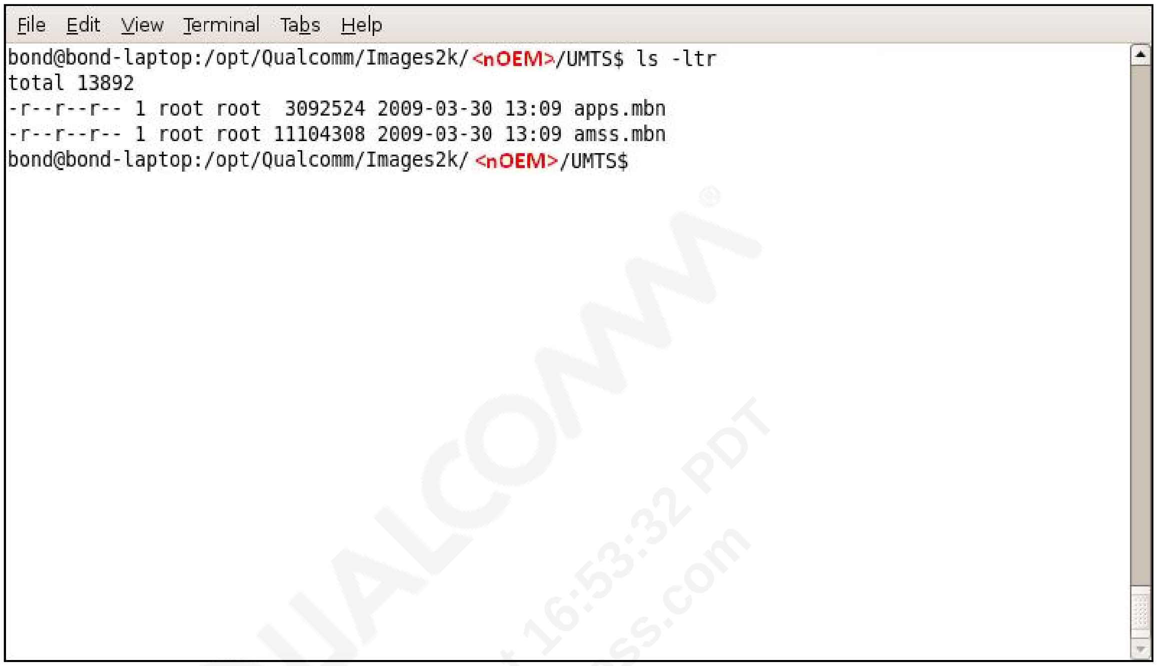
ID – Carrier	Image build ID	UQCN build ID**	Technology
0 – Vodafone	D1025MSTUTABGD	D1025MUQCNABFM	UMTS
1 – Verizon	D1055MSTUTDSVD	D1055MUQCNDSVM	CDMA
2 – AT&T	D1025MSTUTABGD	D1025MUQCNASDM	UMTS
3 – Sprint	D1055MSTUTCSFD	D1055MUQCNCSFM	CDMA
4 – T-Mobile	D1025MSTUTABGD	D1025MUQCNABLM	UMTS
6 – Generic EU	D1025MSTUTABGD	D1025MUQCNABGM	UMTS
7 – Telefonica	D1025MSTUTABGD	D1025MUQCNABHM	UMTS
8 – Telecom Italia	D1025MSTUTABGD	D1025MUQCNABIM	UMTS
9 – Orange	D1025MSTUTABGD	D1025MUQCNABOM	UMTS
11 – Alltel*	D1055MSTUTCSLD	D1055MUQCNCSLM	CDMA
12 – DOCOMO	D1025MSTUTABED	D1025MUQCNABEM	UMTS
14 – Telcel	D1025MSTUTABGD	D1025MUQCNABKM	UMTS

Note: D1025 → UMTS, D1055 → CDMA

*****This carrier image is no longer supported or packaged in a Linux release. Alltel functionalities are supported through the Verizon image.

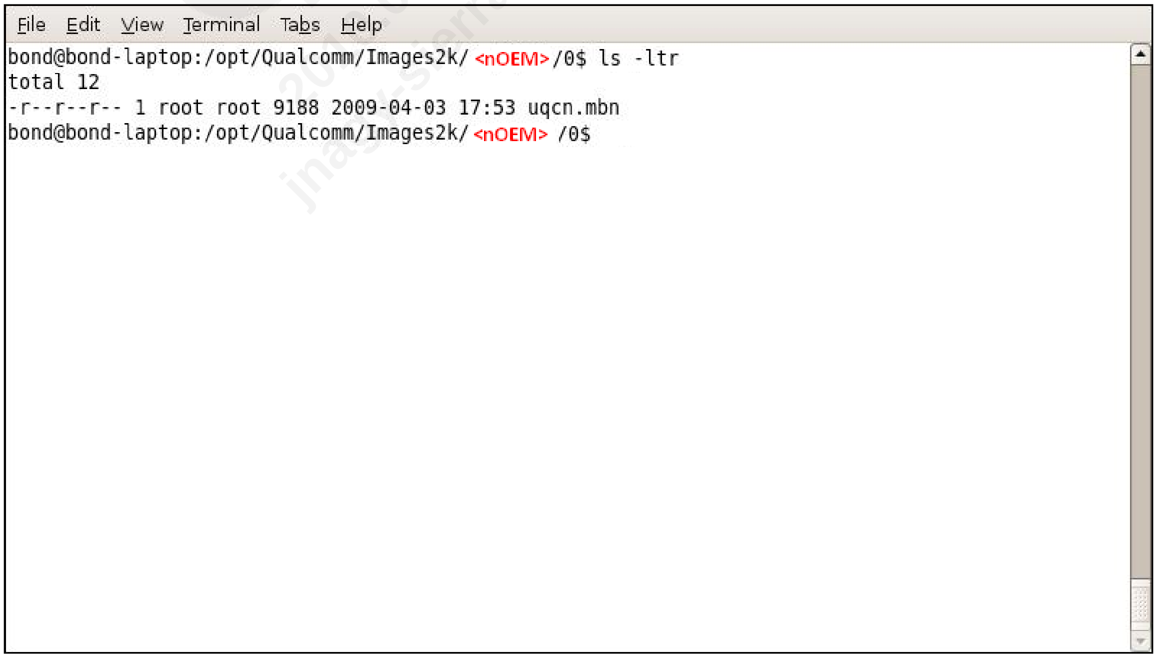
**UQCN build ID may be the same as the image build ID for certain releases.

All UMTS carrier images use the generic image and a carrier-specific UQCN file (Figure 2-4 through Figure 2-6).



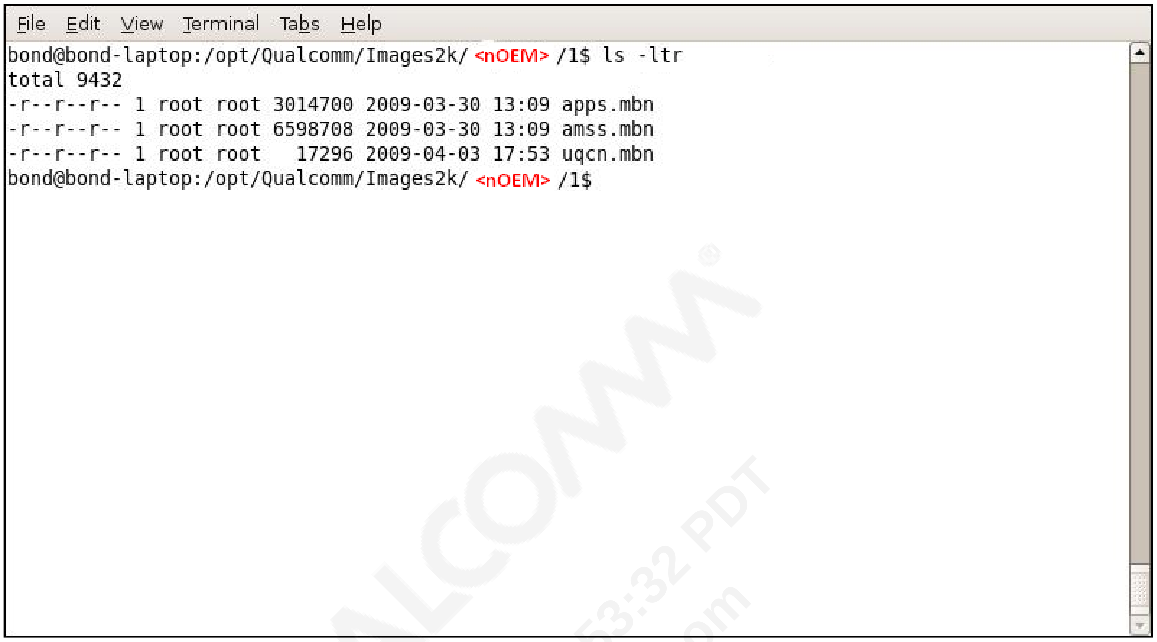
```
bond@bond-laptop:/opt/Qualcomm/Images2k/<nOEM>/UMTS$ ls -ltr
total 13892
-r--r--r-- 1 root root  3092524 2009-03-30 13:09 apps.mbn
-r--r--r-- 1 root root 11104308 2009-03-30 13:09 amss.mbn
bond@bond-laptop:/opt/Qualcomm/Images2k/ <nOEM>/UMTS$
```

Figure 2-4 UMTS images files



```
bond@bond-laptop:/opt/Qualcomm/Images2k/ <nOEM>/0$ ls -ltr
total 12
-r--r--r-- 1 root root 9188 2009-04-03 17:53 uqcn.mbn
bond@bond-laptop:/opt/Qualcomm/Images2k/ <nOEM> /0$
```

Figure 2-5 UMTS UQCN files



```
bond@bond-laptop:/opt/Qualcomm/Images2k/<nOEM>/1$ ls -ltr
total 9432
-r--r--r-- 1 root root 3014700 2009-03-30 13:09 apps.mbn
-r--r--r-- 1 root root 6598708 2009-03-30 13:09 amss.mbn
-r--r--r-- 1 root root   17296 2009-04-03 17:53 uqcn.mbn
bond@bond-laptop:/opt/Qualcomm/Images2k/<nOEM>/1$
```

Figure 2-6 CDMA images files

2.2.2 QDLService2k

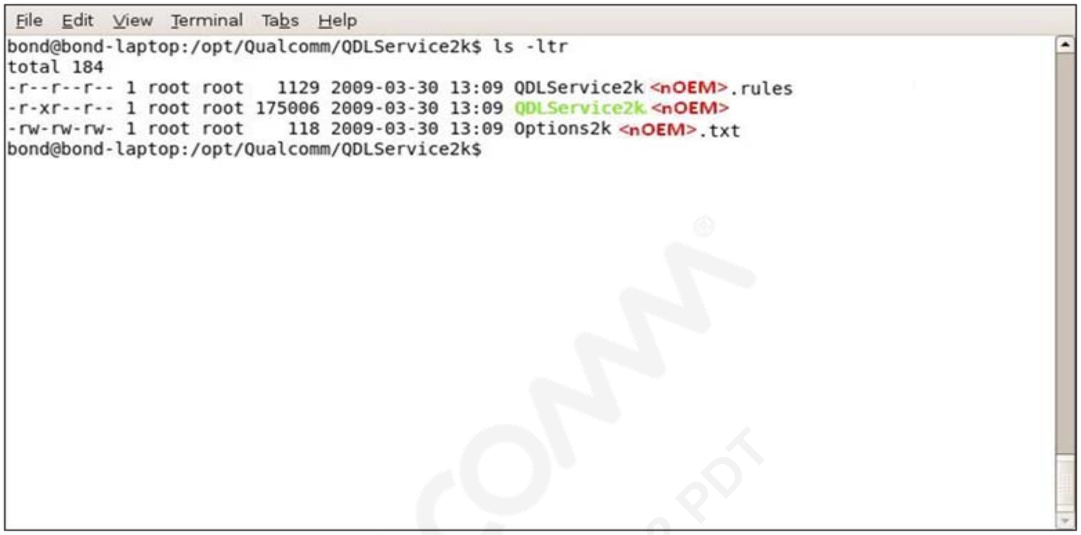
The QDLService2k folder in /opt/Qualcomm/ contains the following upon installation:

- QDLService2k*, * representing the nOEM name
- QDLService2k*.rules
- Options2k*.txt

Upon the first attempt to download firmware to the Gobi2000 module, the QDLService2k folder will have an additional file:

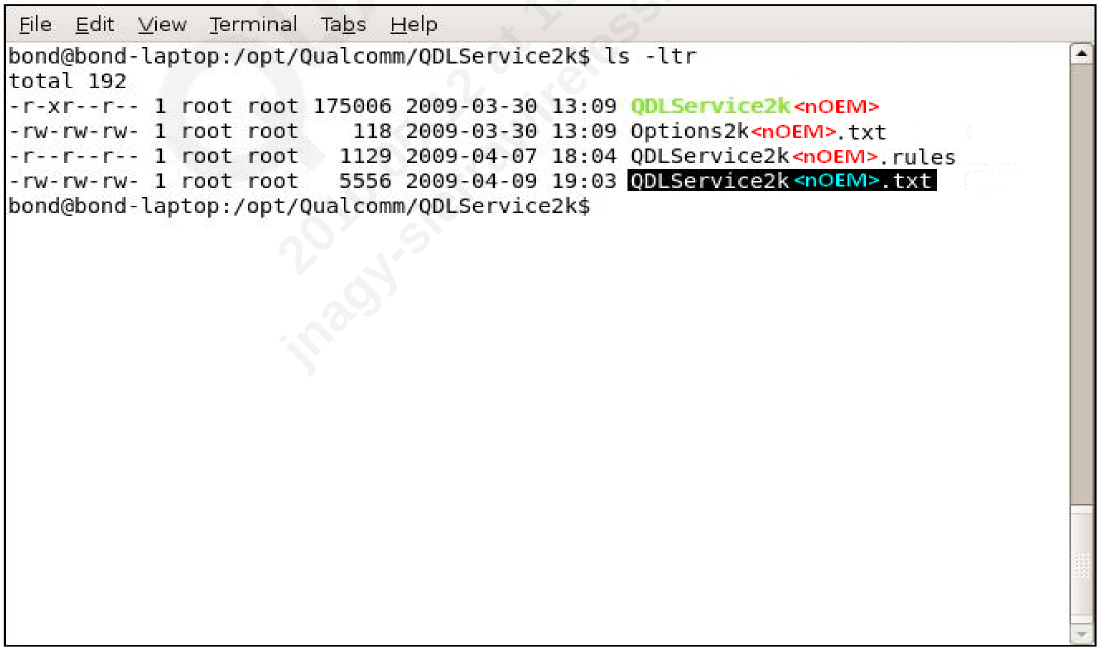
- QDLService2k*.txt

The folder details are illustrated in Figure 2-7 and Figure 2-8.



```
bond@bond-laptop:/opt/Qualcomm/QDLService2k$ ls -ltr
total 184
-r--r--r-- 1 root root   1129 2009-03-30 13:09 QDLService2k<nOEM>.rules
-r-xr--r-- 1 root root 175006 2009-03-30 13:09 QDLService2k<nOEM>
-rw-rw-rw- 1 root root    118 2009-03-30 13:09 Options2k<nOEM>.txt
bond@bond-laptop:/opt/Qualcomm/QDLService2k$
```

Figure 2-7 QDL file locations



```
bond@bond-laptop:/opt/Qualcomm/QDLService2k$ ls -ltr
total 192
-r-xr--r-- 1 root root 175006 2009-03-30 13:09 QDLService2k<nOEM>
-rw-rw-rw- 1 root root    118 2009-03-30 13:09 Options2k<nOEM>.txt
-r--r--r-- 1 root root   1129 2009-04-07 18:04 QDLService2k<nOEM>.rules
-rw-rw-rw- 1 root root   5556 2009-04-09 19:03 QDLService2k<nOEM>.txt
bond@bond-laptop:/opt/Qualcomm/QDLService2k$
```

Figure 2-8 QDL file locations with QDLService2k*.txt

2.2.3 QDLService2k*

QDLService2k* is an executable that loads the firmware images to the Gobi module. This executable is not designed to be run manually by the user, but instead by the udev daemon.

Location: /opt/Qualcomm/QDLService2k/QDLService2k* (see Figure 2-8).

2.2.4 QDLService2k* rules

This udev rule tells the udev daemon to execute the QDL Service when it sees a new device with:

- Correct Boot VID/PID
- Device node at ttyUSB<#> (which requires the driver to be loaded)

Location: /opt/Qualcomm/QDLService2k/QDLService2k*.rules (see Figure 2-8).

2.2.5 QDLService2k*.txt

QDLService2k*.txt is a log file that is updated by the service while it is running. The messages written by the service into the text file are important, because they may help to solve problems (i.e., when the service does not load the AMSS image to the device). Due to the nature of this file, it may become very large, so the service deletes it the first time the QDL Service is run after a reboot.

Location: /opt/Qualcomm/QDLService2k/QDLService2k*.txt (see Figure 2-8)

2.2.6 Starting and stopping QDLService

QDLService does not run continuously. It is started based on the udev rules given in Section 2.2.4. When download is complete, the QDLService stops running. The QDLService can also be manually started by:

```
/opt/Qualcomm/QDLService2k/QDLService2k<nOEM> /dev/ttyUSB<#>
```

A super user must perform the manual start.

NOTE QDLService2k* is not a service.

2.2.7 Options2k*.txt

Options2k*.txt is a text file that contains the paths of the required carrier images. The paths given in the Options2k*.txt file are used for the default carrier images chosen by the nOEM. Options2k*.txt points to three images: amss.mbn, apps.mbn, and uqcn.mbn. The contents of Options2k*.txt are as follows (see Figure 2-9 and Figure 2-10):

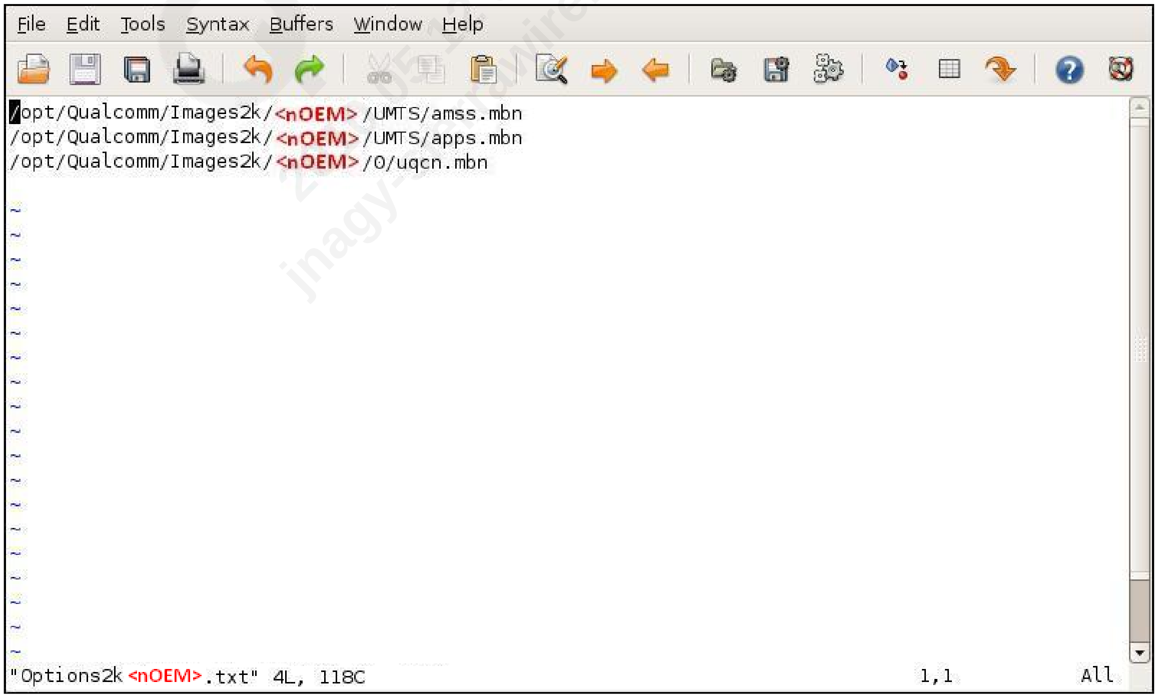
- For UMTS carriers used as default
  - ❐ /opt/Qualcomm/Images2k/<nOEM>/UMTS/amss.mbn
  - ❐ /opt/Qualcomm/Images2k/<nOEM>/UMTS/apps.mbn
  - ❐ /opt/Qualcomm/Images2k/<nOEM>/#/uqcn.mbn

# indicates the number assigned for a specific carrier.

- For CDMA carriers used as default
  - ❐ /opt/Qualcomm/Images2k/<nOEM>/#/amss.mbn
  - ❐ /opt/Qualcomm/Images2k/<nOEM>/#/apps.mbn
  - ❐ /opt/Qualcomm/Images2k/<nOEM>/#/uqcn.mbn

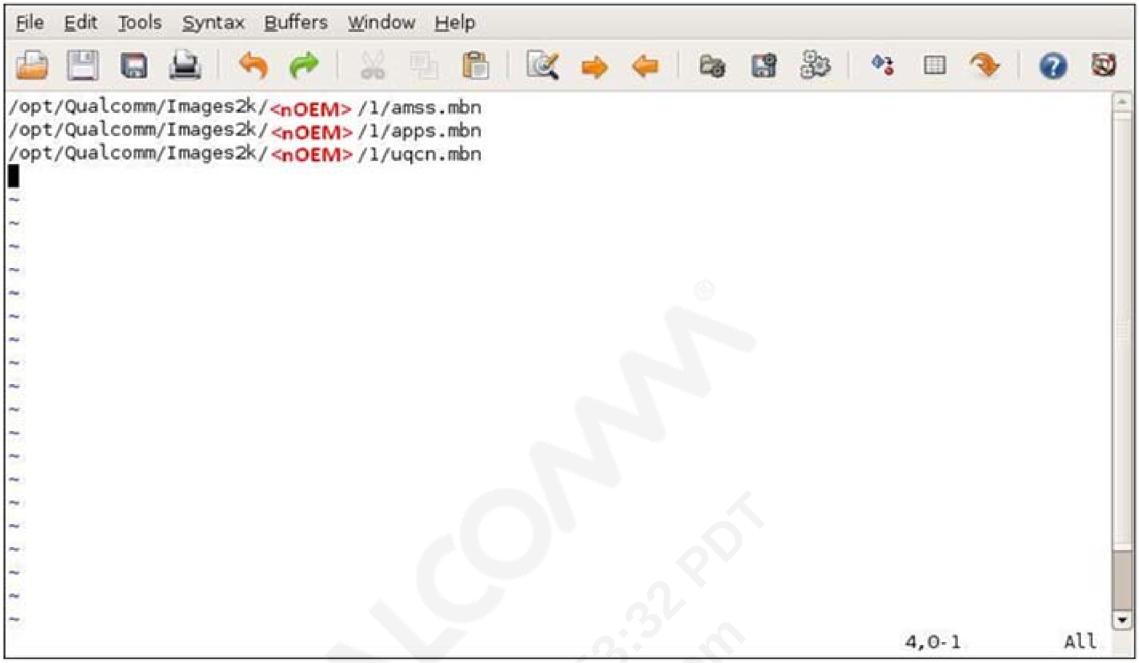
# indicates the number assigned for a specific carrier.

Location: /opt/Qualcomm/QDLService2k/Options2k*.txt (see Figure 2-8)



```
/opt/Qualcomm/Images2k/<nOEM>/UMTS/amss.mbn
/opt/Qualcomm/Images2k/<nOEM>/UMTS/apps.mbn
/opt/Qualcomm/Images2k/<nOEM>/0/uqcn.mbn
"Options2k<nOEM>.txt" 4L, 118C                                1,1           All
```

Figure 2-9 Options2k*.txt for UMTS (Image 0)



```
/opt/Qualcomm/Images2k/<nOEM>/1/amss.mbn
/opt/Qualcomm/Images2k/<nOEM>/1/apps.mbn
/opt/Qualcomm/Images2k/<nOEM>/1/uqcn.mbn
```

Figure 2-10 Options2k*.txt for CDMA

2.2.8 Modifying Options2k*.txt

Modifying the Options2k*.txt file is not recommended. However, if the file is edited manually, the Gobi2000 module must be reset for the change to take effect. This manual firmware change is applicable only when the NV item NV_IRDA_PNP_STATUS_I (3186) is either inactive or active, with a value of 0xFFFFFFFF.

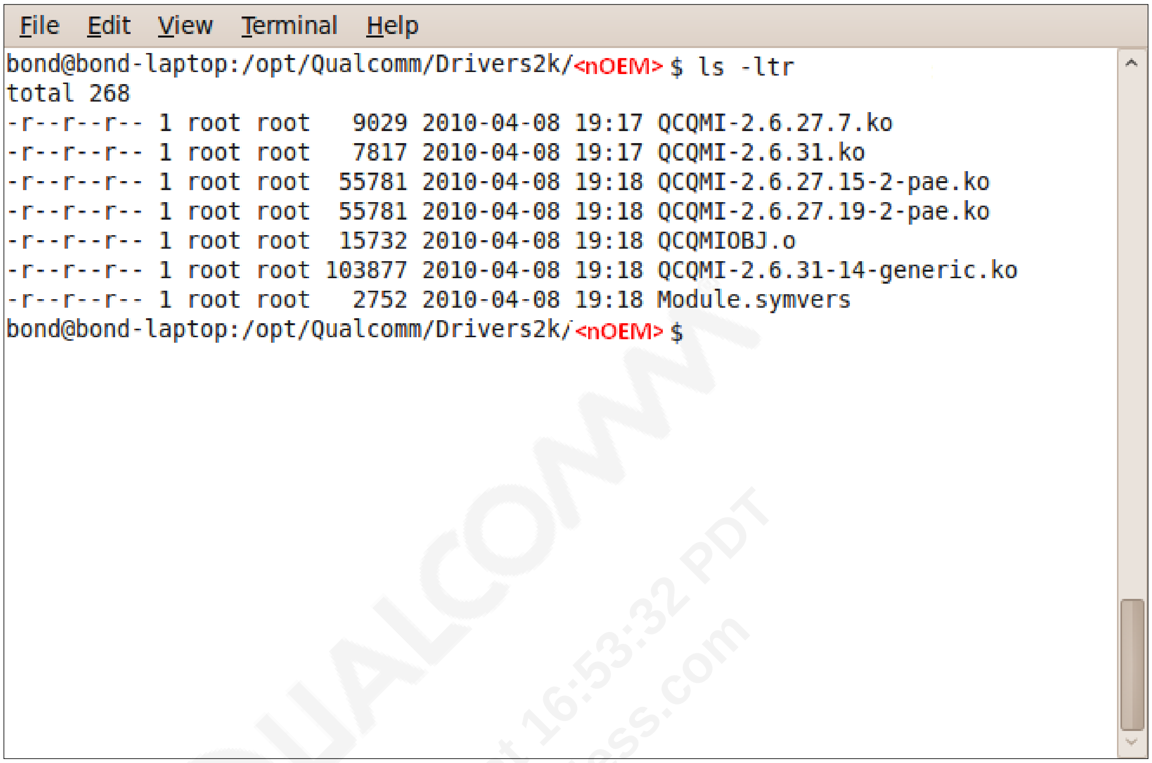
2.2.9 Drivers2k

The Drivers2k folder contains:

- Module.symvers
- QCQMI kernel objects for different kernel versions supported
- QCQMIOBJ.o object to be used for building different QCQMI kernels

The QCQMI driver and its Module.symvers file are required to build the QCUSBNet2k*.c driver.

Location: /opt/Qualcomm/Drivers2k/<nOEM> (see Figure 2-11)



```
bond@bond-laptop:/opt/Qualcomm/Drivers2k/<nOEM> $ ls -ltr
total 268
-r--r--r-- 1 root root   9029 2010-04-08 19:17 QCQMI-2.6.27.7.ko
-r--r--r-- 1 root root   7817 2010-04-08 19:17 QCQMI-2.6.31.ko
-r--r--r-- 1 root root  55781 2010-04-08 19:18 QCQMI-2.6.27.15-2-pae.ko
-r--r--r-- 1 root root  55781 2010-04-08 19:18 QCQMI-2.6.27.19-2-pae.ko
-r--r--r-- 1 root root  15732 2010-04-08 19:18 QCQMIOBJ.o
-r--r--r-- 1 root root 103877 2010-04-08 19:18 QCQMI-2.6.31-14-generic.ko
-r--r--r-- 1 root root   2752 2010-04-08 19:18 Module.symvers
bond@bond-laptop:/opt/Qualcomm/Drivers2k/<nOEM> $
```

Figure 2-11 QCQMI and Module.symvers in Drivers2k

3 Qualcomm Software Installation – Open Source

3.1 Downloading open source files

To access the open source software code files, go to https://www.codeaurora.org/ and select the Gobi project through the Gobi Module in Forum World or through the View all Projects link in Forum World, or go directly to https://www.codeaurora.org/patches/quic/gobi/.

The open-source release accompanies release of the proprietary software. Customers are advised to check the Gobi folder at the CAF when they receive notification for release of the proprietary software. At times, there is a delay of one or two days between release of the proprietary software and the open-source software.

The open-source package is stored as <AAXXX>.Gobi2000Drivers.tar.gz, where AAXXX is the nOEM-specific item number.

3.2 USB Serial driver installation

The USB Serial driver is called QCSerial2k*.c. The USB Serial driver must be built before it can be used. QCSerial2k*.ko is the driver binary that will be installed and used.

3.2.1 Downloading files for USB Serial driver

The USB Serial driver files are QCSerial2k*.c and its Makefile. These files are released as the open-source software from CAF and are available at the QCSerial2k folder.

3.2.2 Prerequisites

Prerequisites for building a USB Serial driver are:

- Kernel headers or full source must be installed for the currently running kernel. There must be a link, /lib/modules/<uname –r>/build, that points to these kernel headers or kernel sources.
- The kernel must support the usbcore and usbserial drivers, either as modules or built into the kernel.
- Tools required for building the kernel must be used. These tools usually come in a package called linux-kernel-devel.
- gcc compiler
- make tool

3.2.3 Building the driver (QCSerial2k*.c)

To build the driver:

1. Navigate to the folder that contains QCSerial2k*.c and Makefile.
2. Run the `make` command. The driver is built for the current kernel and placed in the current directory as QCSerial2k*.ko.

3.2.4 Installing QCSerial2k*.ko

Perform the following steps to install the newly created driver module. The steps must be performed as a super user.

1. Copy the newly created QCSerial2k*.ko into the directory /lib/modules/<uname –r>/kernel/drivers/usb/serial/.
2. Run the `depmod` command. This regenerates the module dependencies and allows the QCSerial2k* driver to be loaded when the Gobi device is connected.
3. Add the line QCSerial2k* to /etc/modules. (This file exists only on some Debian-based systems. If it does not exist, this step can be omitted.) This is a system-specific file that ensures the driver is loaded when the computer is started.
4. (Optional) Load the driver manually with the command modprobe QCSerial2k*.

   This is required only the first time, after installation of the QCSerial2k* driver. The QCSerial2k* driver is automatically loaded following a restart, or if the Gobi device is plugged in.

NOTE When enumerated, the Gobi device appears as /dev/ttyUSB<#>, where <#> signifies the next available ttyUSB port. In Composite mode, some connection managers may require the modem to be renamed. A user can create a symbolic link by typing:

```
ln -f /dev/ttyUSB<#> /dev/modem
```

3.3 QCQMI driver installation

The QCQMI driver is provided pre-built, as a part of the GOBI2000_LINUX_PACKAGE (see section 2.2.9). If a user desires to rebuild this driver, they may do so by using the instructions in the following sections.

The QCQMI driver is necessary to build the USB network driver. QCQMI.ko, which is the driver binary, will be used to build the network driver.

3.3.1 Downloading files for QCQMI driver

The files related to the QCQMI driver are QCQMIExports.h, ModCore.c, and its Makefile. These are released as open-source software from CAF, and are available in the QCQMI folder.

3.3.2 Prerequisites

Prerequisites for building a QCQMI driver:

- Kernel headers or full source must be installed for the currently running kernel. There must be a link, /lib/modules/<uname –r>/build, that points to these kernel headers or kernel sources.
- Tools required for building the kernel must be used. These tools usually come in a package called linux-kernel-devel.
- gcc compiler
- make tool
- QCQMIOBJ.o file, installed by the Gobi2000-Linux-Package

3.3.3 Building the driver (QCQMI.ko)

To build the driver:

1. Navigate to the QCQMI folder that contains QCQMIExports.h, ModCore.c, and Makefile.
2. Copy the QCQMIOBJ.o object file to the current directory by typing
   `cp /opt/Qualcomm/Drivers2k/<nOEM>/QCQMIOBJ.o ./`
3. Run the `make` command. The driver is built for the current kernel and placed in the current directory as QCQMI.ko.

3.3.4 Installing QCQMI.ko

Perform the following steps to install the newly created driver module. The steps must be performed as a super user.

1. Copy the newly created QCQMI.ko into the directory /lib/modules/<uname –r>/kernel/drivers/net/usb/.
2. Run the `depmod` command. This regenerates the module dependencies and allows the QCQMI driver to be loaded when the Gobi device is connected.
3. Add the line QCQMI to /etc/modules. (This file exists only on some Debian-based systems. If it does not exist, this step can be omitted.) This is a system-specific file that ensures the driver is loaded when the computer is started.
4. (Optional) Load the driver manually with the command modprobe QCQMI.

   This is required only the first time after installation of the QCQMI driver. The QCQMI driver is automatically loaded following a restart, or if the Gobi device is plugged in.

NOTE A warning message is printed by the driver when the QCQMI module is loaded. The message is as follows: QCQMI: module license ‘unspecified’ taints kernel. This is expected, because the QCQMI driver is not licensed under GPL.

3.4 USB network driver installation

The USB network driver is called QCUSBNet2k*.c. The network driver must be built before it can be used. CAF provides the Makefile, with QCUSBNet2k*.c, to build it into QCUSBNet2k*.ko. This is the driver binary that will be installed and used.

3.4.1 Downloading files for USB network driver

The files related to the USB network driver are QCUSBNet2k*.c, QMIDevice.c, QMIDevice.h, Structs.h, QCQMIExports.h, and the Makefile. These are released as open-source software from CAF, and are available in the QCUSBNet2k folder.

3.4.2 Prerequisites

Prerequisites for building a USB network driver:

- Kernel headers or full source must be installed for the currently running kernel. There must be a link, /lib/modules/<uname –r>/build, that points to these kernel headers or kernel sources.
- The kernel must support the usbcore and usbnet drivers, either as modules or built into the kernel.
- Tools required for building the kernel must be used. These tools usually come in a package called linux-kernel-devel.
- gcc compiler
- make tool
- QCQMI driver, installed by the Gobi2000-Linux-Package for supported kernels, or as built in the preceding section for any other kernel

3.4.3 Building the driver (QCUSBNet2k*.c)

To build the driver:

1. Navigate to the folder that contains QCUSBNet2k*.c, QCQMIExports.h, QMIDevice.c, QMIDevice.h, Structs.h, and Makefile.
2. For kernels prior to 2.6.25, create a symbolic link to the usbnet.h file in the kernel sources by typing **`ln –s <linux sources>/drivers/net/usb/usbnet.h ./`**
3. Copy the QCQMI driver Module.symvars file to the current directory by typing
   `cp /opt/Qualcomm/Drivers2k/<nOEM>/Module.symvers ./`
4. Run the `make` command. The driver is built for the current kernel and placed in the current directory as QCUSBNet2k*.ko.

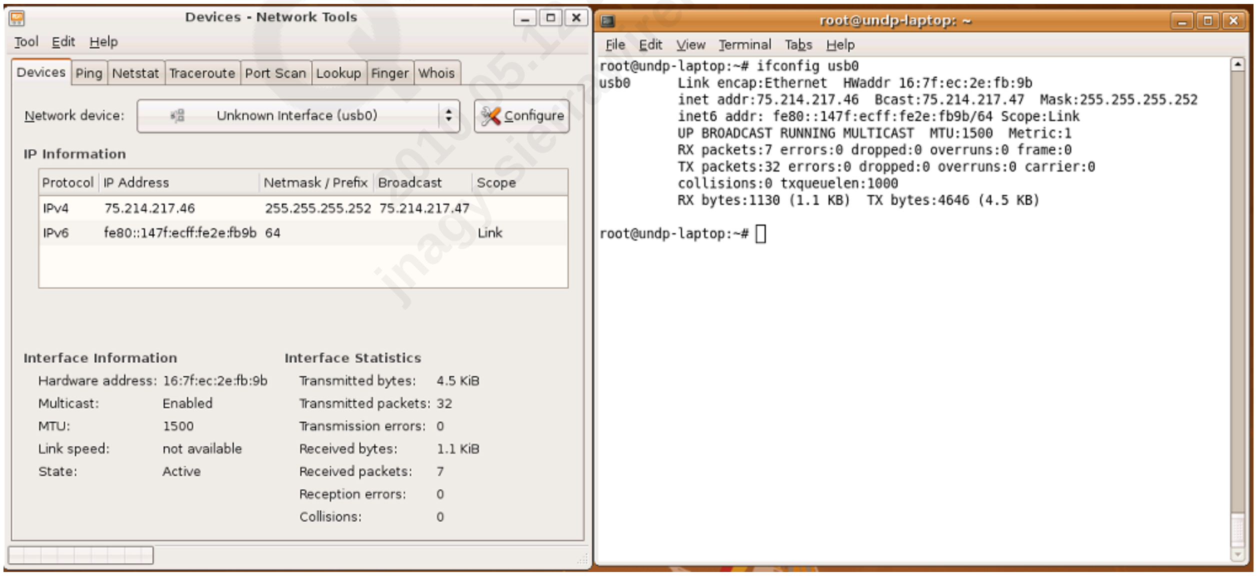
3.4.4 Installing QCUSBNet2k*.ko

Perform the following steps to install the newly created driver module. The steps must be performed as a super user.

1. Copy the newly created QCUSBNet2k*.ko into the directory /lib/modules/<uname –r>/kernel/drivers/net/usb/
2. Run the `depmod` command. This regenerates the module dependencies and allows the QCUSBNet2k* driver to be loaded when the Gobi device is connected.
3. Add the line QCUSBNet2k* to /etc/modules. (This file exists only on some Debian-based systems. If it does not exist, this step can be omitted.) This is a system-specific file that ensures the driver is loaded when the computer is started.
4. (Optional) Load the driver manually with the command modprobe QCUSBNet2k*.

This is required only the first time after installation of the QCUSBNet2k* driver. The QCUSBNet2k* driver is automatically loaded following a restart, or if the Gobi device is plugged in.

NOTE When enumerated in composite mode, the Gobi device appears as a network interface usb<#>, where <#> signifies the next available USB network interface. The device node /dev/qcqmi<#> will also be created for use by QCWWANCMAPI2k. See Figure 3-1.



Network device: Unknown Interface (usb0)

IP Information

Protocol	IP Address	Netmask / Prefix	Broadcast	Scope
IPv4	75.214.217.46	255.255.255.252	75.214.217.47	
IPv6	fe80::147f:ecff:fe2e:fb9b	64		Link

Interface Information

Hardware address:	16:7f:ec:2e:fb:9b
Multicast:	Enabled
MTU:	1500
Link speed:	not available
State:	Active

Interface Statistics

Transmitted bytes:	4.5 KiB
Transmitted packets:	32
Transmission errors:	0
Received bytes:	1.1 KiB
Received packets:	7
Reception errors:	0
Collisions:	0

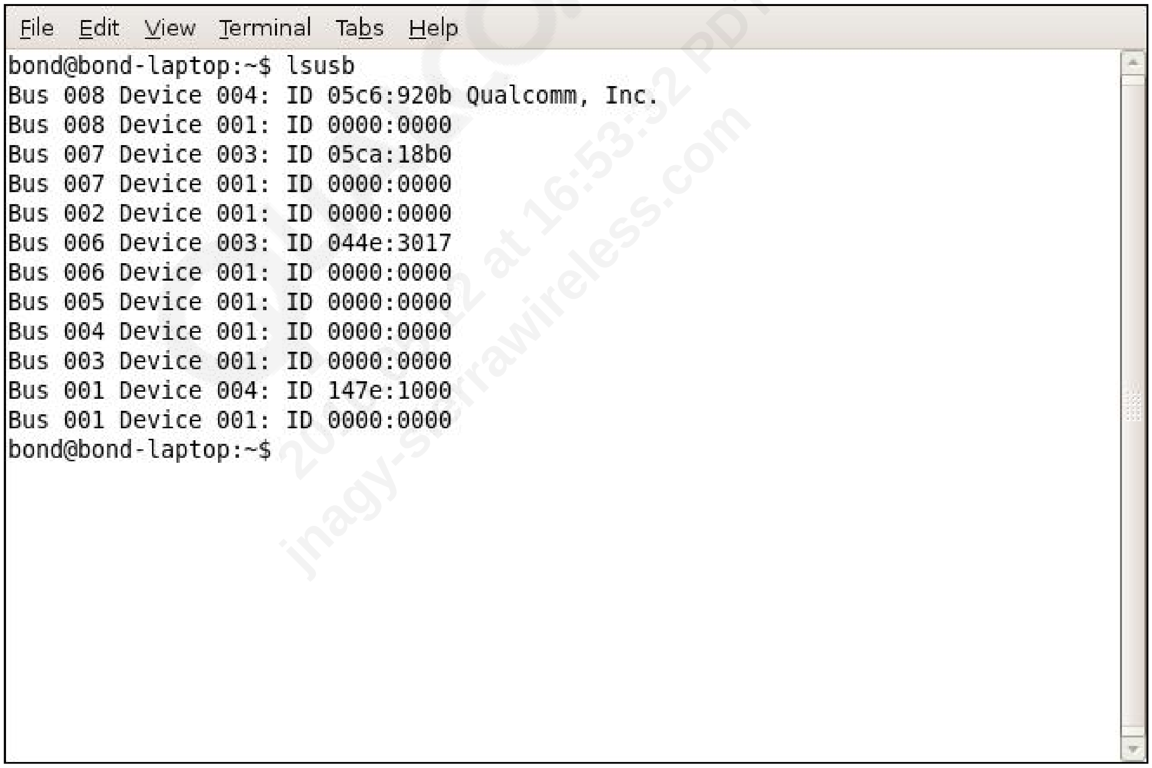
```
root@undp-laptop:~# ifconfig usb0
usb0      Link encap:Ethernet  HWaddr 16:7f:ec:2e:fb:9b
          inet addr:75.214.217.46  Bcast:75.214.217.47  Mask:255.255.255.252
          inet6 addr: fe80::147f:ecff:fe2e:fb9b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1130 (1.1 KB)  TX bytes:4646 (4.5 KB)

root@undp-laptop:~# 
```

Figure 3-1 Network interface

4 Gobi Troubleshooting

4.1 Verifying enumeration

To verify that the device is connected to the USB bus, type the `lsusb` command. This shows the bus and device ID, followed by the VID/PID and name of each device connected to the USB bus. An example is provided in Figure 4-1.



```
bond@bond-laptop:~$ lsusb
Bus 008 Device 004: ID 05c6:920b Qualcomm, Inc.
Bus 008 Device 001: ID 0000:0000
Bus 007 Device 003: ID 05ca:18b0
Bus 007 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 006 Device 003: ID 044e:3017
Bus 006 Device 001: ID 0000:0000
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 001 Device 004: ID 147e:1000
Bus 001 Device 001: ID 0000:0000
bond@bond-laptop:~$
```

Figure 4-1 lsusb command

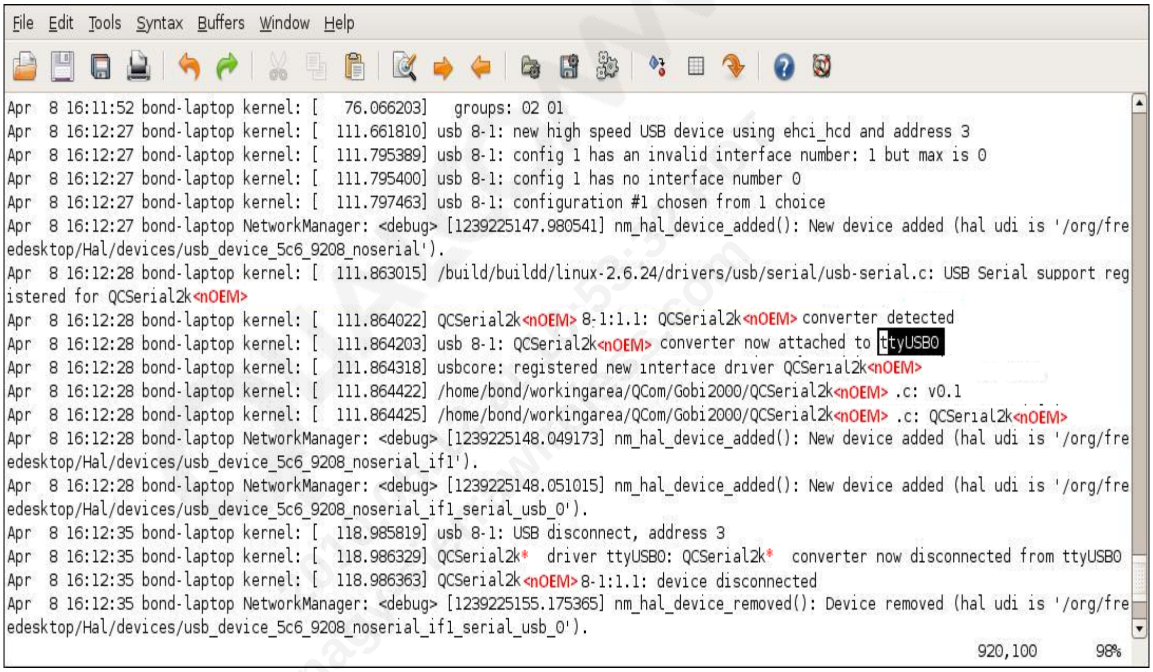
The enumeration of the Gobi device with the VID/PID, as illustrated, occurs after the driver and the installer package is installed. This is a system tool and is not Gobi-specific.

4.2 Debugging

There are several ways to debug if the Gobi card is not detected or enumerated.

4.2.1 Kernel logs

The syslogd daemon logs all the activities of the hardware configured during boot time and runtime. The logs are saved at /var/log/. The log files are named kern.log, daemon.log, messages, and syslog. The new USB port is enumerated in the form ttyUSB<#>, as shown in Figure 4-2.

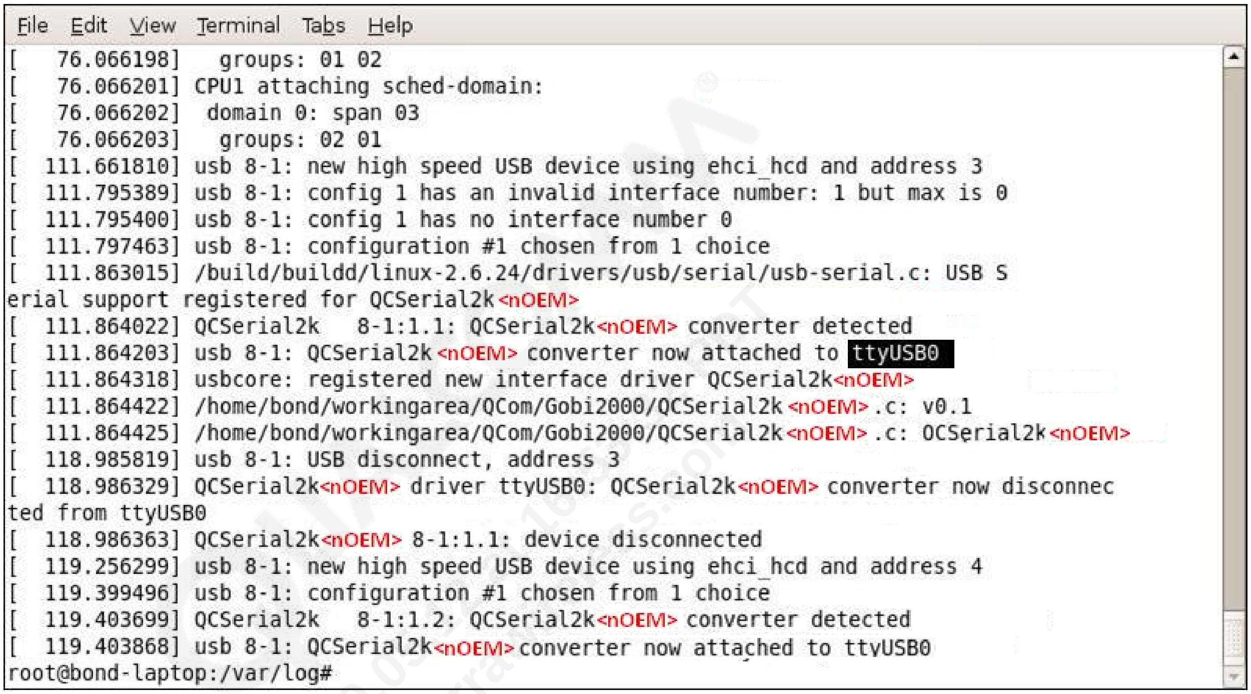


```
Apr  8 16:11:52 bond-laptop kernel: [   76.066203]   groups: 02 01
Apr  8 16:12:27 bond-laptop kernel: [  111.661810] usb 8-1: new high speed USB device using ehci_hcd and address 3
Apr  8 16:12:27 bond-laptop kernel: [  111.795389] usb 8-1: config 1 has an invalid interface number: 1 but max is 0
Apr  8 16:12:27 bond-laptop kernel: [  111.795400] usb 8-1: config 1 has no interface number 0
Apr  8 16:12:27 bond-laptop kernel: [  111.797463] usb 8-1: configuration #1 chosen from 1 choice
Apr  8 16:12:27 bond-laptop NetworkManager: <debug> [1239225147.980541] nm_hal_device_added(): New device added (hal udi is '/org/fre
edesktop/Hal/devices/usb_device_5c6_9208_noserial').
Apr  8 16:12:28 bond-laptop kernel: [  111.863015] /build/buildd/linux-2.6.24/drivers/usb/serial/usb-serial.c: USB Serial support reg
istered for QCSerial2k<nOEM>
Apr  8 16:12:28 bond-laptop kernel: [  111.864022] QCSerial2k<nOEM> 8-1:1.1: QCSerial2k<nOEM> converter detected
Apr  8 16:12:28 bond-laptop kernel: [  111.864203] usb 8-1: QCSerial2k<nOEM> converter now attached to ttyUSB0
Apr  8 16:12:28 bond-laptop kernel: [  111.864318] usbcore: registered new interface driver QCSerial2k<nOEM>
Apr  8 16:12:28 bond-laptop kernel: [  111.864422] /home/bond/workingarea/QCom/Gobi2000/QCSerial2k<nOEM> .c: v0.1
Apr  8 16:12:28 bond-laptop kernel: [  111.864425] /home/bond/workingarea/QCom/Gobi2000/QCSerial2k<nOEM> .c: QCSerial2k<nOEM>
Apr  8 16:12:28 bond-laptop NetworkManager: <debug> [1239225148.049173] nm_hal_device_added(): New device added (hal udi is '/org/fre
edesktop/Hal/devices/usb_device_5c6_9208_noserial_if1').
Apr  8 16:12:28 bond-laptop NetworkManager: <debug> [1239225148.051015] nm_hal_device_added(): New device added (hal udi is '/org/fre
edesktop/Hal/devices/usb_device_5c6_9208_noserial_if1_serial_usb_0').
Apr  8 16:12:35 bond-laptop kernel: [  118.985819] usb 8-1: USB disconnect, address 3
Apr  8 16:12:35 bond-laptop kernel: [  118.986329] QCSerial2k*  driver ttyUSB0: QCSerial2k*  converter now disconnected from ttyUSB0
Apr  8 16:12:35 bond-laptop kernel: [  118.986363] QCSerial2k<nOEM> 8-1:1.1: device disconnected
Apr  8 16:12:35 bond-laptop NetworkManager: <debug> [1239225155.175365] nm_hal_device_removed(): Device removed (hal udi is '/org/fre
edesktop/Hal/devices/usb_device_5c6_9208_noserial_if1_serial_usb_0').
```

Figure 4-2 SYSLOG

4.2.2 DMESG

The dmesg command can be run on the command prompt to see the kernel/module log at runtime. Run `dmesg -c` (clear the kernel ring buffer before printing) and then plug in the Gobi card, as shown in Figure 4-3.

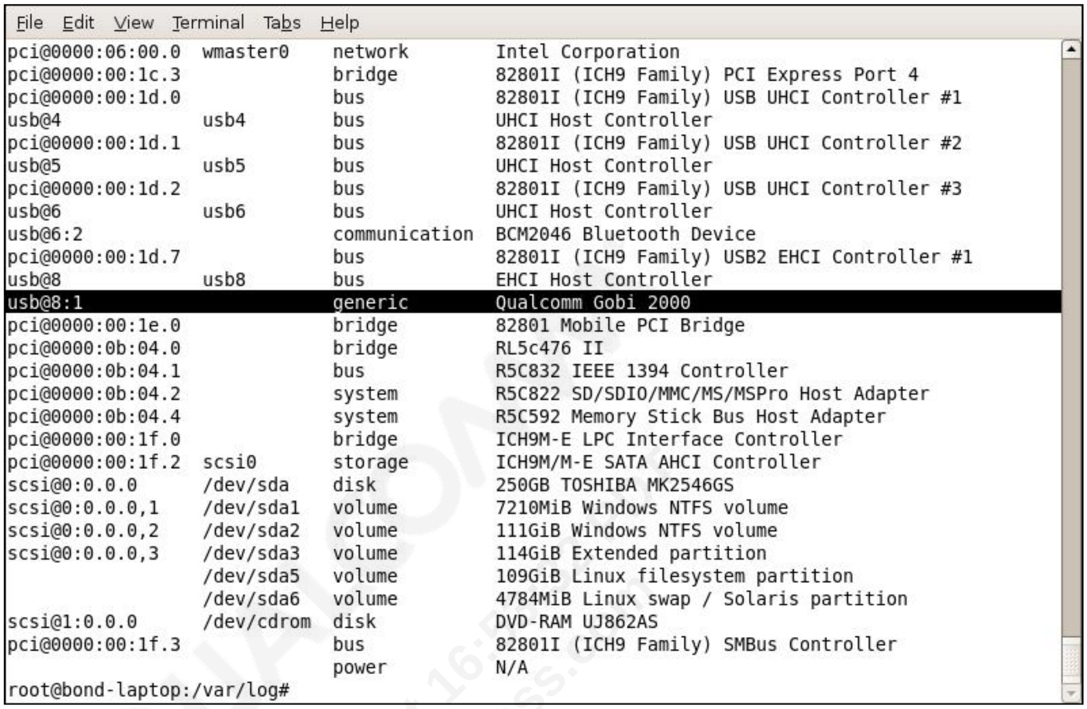


```
[   76.066198]   groups: 01 02
[   76.066201] CPU1 attaching sched-domain:
[   76.066202]  domain 0: span 03
[   76.066203]   groups: 02 01
[  111.661810] usb 8-1: new high speed USB device using ehci_hcd and address 3
[  111.795389] usb 8-1: config 1 has an invalid interface number: 1 but max is 0
[  111.795400] usb 8-1: config 1 has no interface number 0
[  111.797463] usb 8-1: configuration #1 chosen from 1 choice
[  111.863015] /build/buildd/linux-2.6.24/drivers/usb/serial/usb-serial.c: USB S
erial support registered for QCSerial2k<nOEM>
[  111.864022] QCSerial2k   8-1:1.1: QCSerial2k<nOEM> converter detected
[  111.864203] usb 8-1: QCSerial2k<nOEM> converter now attached to ttyUSB0
[  111.864318] usbcore: registered new interface driver QCSerial2k<nOEM>
[  111.864422] /home/bond/workingarea/QCom/Gobi2000/QCSerial2k<nOEM>.c: v0.1
[  111.864425] /home/bond/workingarea/QCom/Gobi2000/QCSerial2k<nOEM>.c: QCSerial2k<nOEM>
[  118.985819] usb 8-1: USB disconnect, address 3
[  118.986329] QCSerial2k<nOEM> driver ttyUSB0: QCSerial2k<nOEM> converter now disconnec
ted from ttyUSB0
[  118.986363] QCSerial2k<nOEM> 8-1:1.1: device disconnected
[  119.256299] usb 8-1: new high speed USB device using ehci_hcd and address 4
[  119.399496] usb 8-1: configuration #1 chosen from 1 choice
[  119.403699] QCSerial2k   8-1:1.2: QCSerial2k<nOEM> converter detected
[  119.403868] usb 8-1: QCSerial2k<nOEM>converter now attached to ttyUSB0
root@bond-laptop:/var/log#
```

Figure 4-3 DMESG

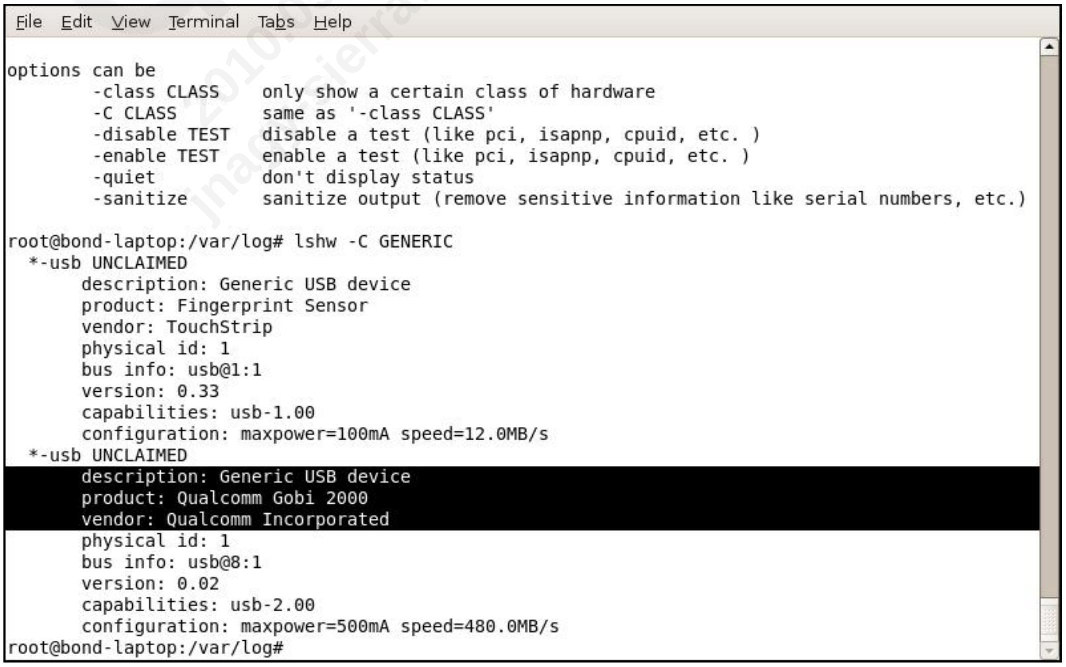
If the device does not enumerate in the /dev directory in the form of /dev/ttyUSB<#>, and the device enumerates under a Windows OS, look for one of the following possible reasons:

- The kernel is not detecting the Gobi hardware on the USB port. Verify that the USB port is working properly, by plugging in a working USB mouse. If the mouse is not detected, the USB port might be faulty. Plug in the Gobi device on another USB port and recheck.
- If the USB mouse is detected, but not the Gobi device, the Gobi USB serial driver (module) is not properly installed. To debug this issue, perform the following steps:
  a. List the hardware and associated drivers (modules) currently detected by the Linux kernel by using the command `lshw -businfo` (Figure 4-4), or `lshw -C GENERIC` (Figure 4-5). Check for Qualcomm Gobi2000 in that file.



```
pci@0000:06:00.0  wmaster0    network        Intel Corporation
pci@0000:00:1c.3              bridge         82801I (ICH9 Family) PCI Express Port 4
pci@0000:00:1d.0              bus            82801I (ICH9 Family) USB UHCI Controller #1
usb@4             usb4        bus            UHCI Host Controller
pci@0000:00:1d.1              bus            82801I (ICH9 Family) USB UHCI Controller #2
usb@5             usb5        bus            UHCI Host Controller
pci@0000:00:1d.2              bus            82801I (ICH9 Family) USB UHCI Controller #3
usb@6             usb6        bus            UHCI Host Controller
usb@6:2                       communication  BCM2046 Bluetooth Device
pci@0000:00:1d.7              bus            82801I (ICH9 Family) USB2 EHCI Controller #1
usb@8             usb8        bus            EHCI Host Controller
usb@8:1                       generic        Qualcomm Gobi 2000
pci@0000:00:1e.0              bridge         82801 Mobile PCI Bridge
pci@0000:0b:04.0              bridge         RL5c476 II
pci@0000:0b:04.1              bus            R5C832 IEEE 1394 Controller
pci@0000:0b:04.2              system         R5C822 SD/SDIO/MMC/MS/MSPro Host Adapter
pci@0000:0b:04.4              system         R5C592 Memory Stick Bus Host Adapter
pci@0000:00:1f.0              bridge         ICH9M-E LPC Interface Controller
pci@0000:00:1f.2  scsi0       storage        ICH9M/M-E SATA AHCI Controller
scsi@0:0.0.0      /dev/sda    disk           250GB TOSHIBA MK2546GS
scsi@0:0.0.0,1    /dev/sda1   volume         7210MiB Windows NTFS volume
scsi@0:0.0.0,2    /dev/sda2   volume         111GiB Windows NTFS volume
scsi@0:0.0.0,3    /dev/sda3   volume         114GiB Extended partition
                  /dev/sda5   volume         109GiB Linux filesystem partition
                  /dev/sda6   volume         4784MiB Linux swap / Solaris partition
scsi@1:0.0.0      /dev/cdrom  disk           DVD-RAM UJ862AS
pci@0000:00:1f.3              bus            82801I (ICH9 Family) SMBus Controller
                              power          N/A
root@bond-laptop:/var/log#
```

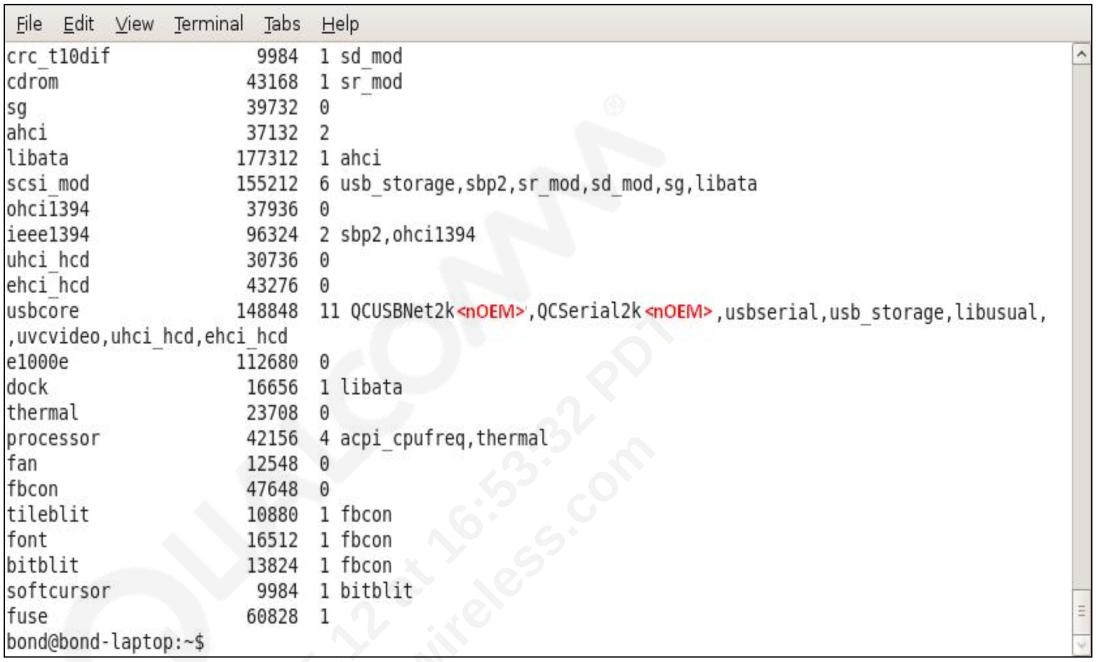
Figure 4-4 LSHW – BUSINFO



```

options can be
        -class CLASS    only show a certain class of hardware
        -C CLASS        same as '-class CLASS'
        -disable TEST   disable a test (like pci, isapnp, cpuid, etc. )
        -enable TEST    enable a test (like pci, isapnp, cpuid, etc. )
        -quiet          don't display status
        -sanitize       sanitize output (remove sensitive information like serial numbers, etc.)

root@bond-laptop:/var/log# lshw -C GENERIC
  *-usb UNCLAIMED
       description: Generic USB device
       product: Fingerprint Sensor
       vendor: TouchStrip
       physical id: 1
       bus info: usb@1:1
       version: 0.33
       capabilities: usb-1.00
       configuration: maxpower=100mA speed=12.0MB/s
  *-usb UNCLAIMED
       description: Generic USB device
       product: Qualcomm Gobi 2000
       vendor: Qualcomm Incorporated
       physical id: 1
       bus info: usb@8:1
       version: 0.02
       capabilities: usb-2.00
       configuration: maxpower=500mA speed=480.0MB/s
root@bond-laptop:/var/log#
```

Figure 4-5 LSHW – C GENERIC

b. If lshw is run and a driver bound to the Gobi device is found, ensure it is communicating to the kernel by running the command `lsmod` to determine if the driver is loaded (search for usbserial and QCSerial2k* in the output). If the driver module is not shown in the list (Figure 4-6), load it by using the modprobe commands.



```
crc_t10dif              9984  1 sd_mod
cdrom                  43168  1 sr_mod
sg                     39732  0
ahci                   37132  2
libata                177312  1 ahci
scsi_mod              155212  6 usb_storage,sbp2,sr_mod,sd_mod,sg,libata
ohci1394               37936  0
ieee1394               96324  2 sbp2,ohci1394
uhci_hcd               30736  0
ehci_hcd               43276  0
usbcore               148848  11 QCUSBNet2k<nOEM>,QCSerial2k<nOEM>,usbserial,usb_storage,libusual,
,uvcvideo,uhci_hcd,ehci_hcd
e1000e                112680  0
dock                   16656  1 libata
thermal                23708  0
processor              42156  4 acpi_cpufreq,thermal
fan                    12548  0
fbcon                  47648  0
tileblit               10880  1 fbcon
font                   16512  1 fbcon
bitblit                13824  1 fbcon
softcursor              9984  1 bitblit
fuse                   60828  1
bond@bond-laptop:~$
```

Figure 4-6 LSMOD

Note that, to get additional debug messages in the kernel/dmesg logs, the user should load the drivers with the debug parameter. This can be done as follows:

- For QCSerial2k*

  rmmod QCSerial2k<nOEM>

  rmmod usbserial

  modprobe usbserial debug

  modprobe QCSerial2k<nOEM> debug

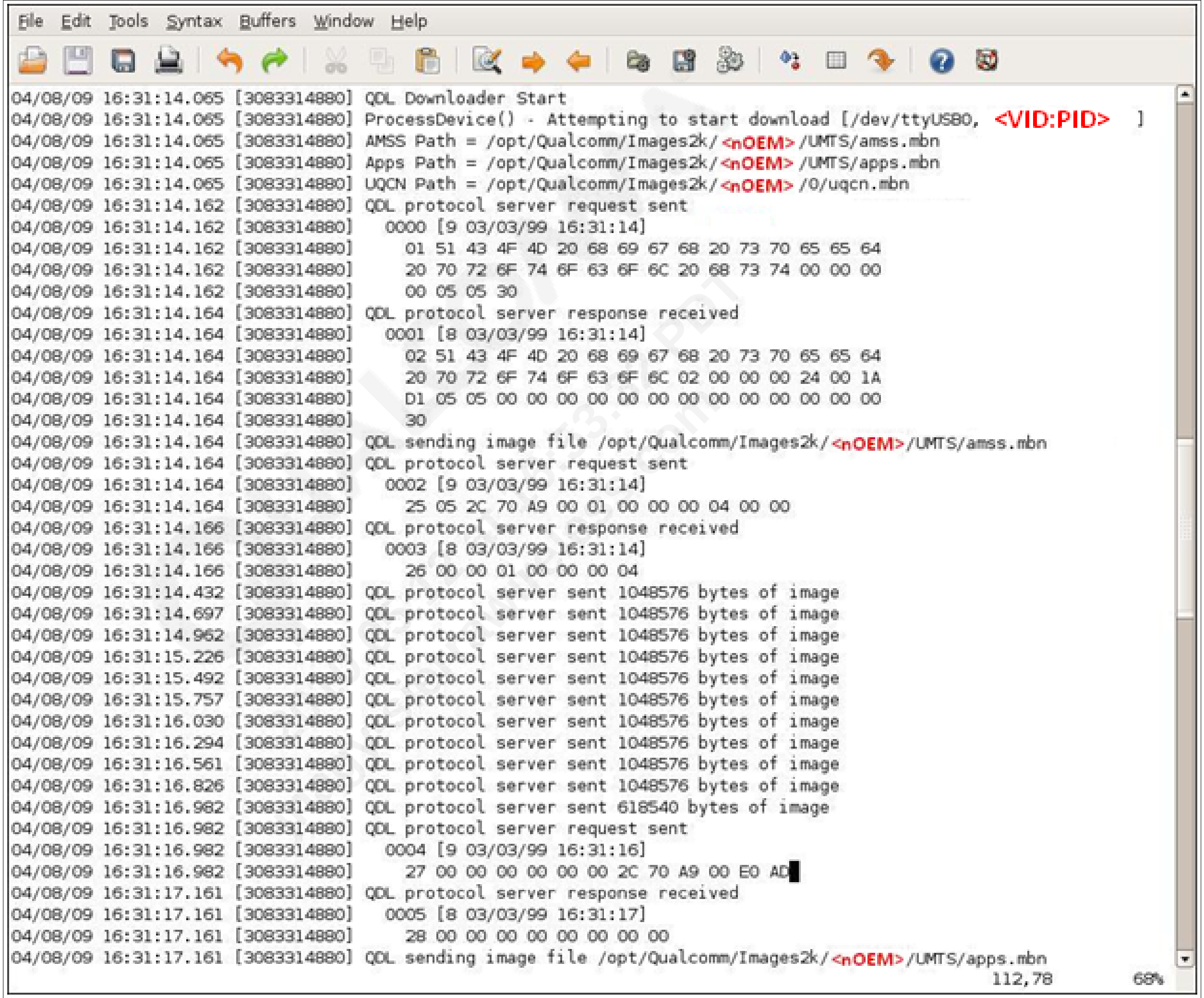
- For QCUSBNet2k*

  rmmod QCUSBNet2k<nOEM>

  modprobe QCUSBNet2k<nOEM> debug

4.2.3 QDLService.txt

If the driver has loaded but the firmware has not loaded, there may be an issue with the firmware or the QDL service. View the QDLService.txt file for related errors. QDLService.txt is illustrated in Figure 4-7.



```
04/08/09 16:31:14.065 [3083314880] QDL Downloader Start
04/08/09 16:31:14.065 [3083314880] ProcessDevice() - Attempting to start download [/dev/ttyUSB0, <VID:PID>   ]
04/08/09 16:31:14.065 [3083314880] AMSS Path = /opt/Qualcomm/Images2k/<nOEM>/UMTS/amss.mbn
04/08/09 16:31:14.065 [3083314880] Apps Path = /opt/Qualcomm/Images2k/<nOEM>/UMTS/apps.mbn
04/08/09 16:31:14.065 [3083314880] UQCN Path = /opt/Qualcomm/Images2k/<nOEM>/0/uqcn.mbn
04/08/09 16:31:14.162 [3083314880] QDL protocol server request sent
04/08/09 16:31:14.162 [3083314880]   0000 [9 03/03/99 16:31:14]
04/08/09 16:31:14.162 [3083314880]     01 51 43 4F 4D 20 68 69 67 68 20 73 70 65 65 64
04/08/09 16:31:14.162 [3083314880]     20 70 72 6F 74 6F 63 6F 6C 20 68 73 74 00 00 00
04/08/09 16:31:14.162 [3083314880]     00 05 05 30
04/08/09 16:31:14.164 [3083314880] QDL protocol server response received
04/08/09 16:31:14.164 [3083314880]   0001 [8 03/03/99 16:31:14]
04/08/09 16:31:14.164 [3083314880]     02 51 43 4F 4D 20 68 69 67 68 20 73 70 65 65 64
04/08/09 16:31:14.164 [3083314880]     20 70 72 6F 74 6F 63 6F 6C 02 00 00 00 24 00 1A
04/08/09 16:31:14.164 [3083314880]     D1 05 05 00 00 00 00 00 00 00 00 00 00 00 00 00
04/08/09 16:31:14.164 [3083314880]     30
04/08/09 16:31:14.164 [3083314880] QDL sending image file /opt/Qualcomm/Images2k/<nOEM>/UMTS/amss.mbn
04/08/09 16:31:14.164 [3083314880] QDL protocol server request sent
04/08/09 16:31:14.164 [3083314880]   0002 [9 03/03/99 16:31:14]
04/08/09 16:31:14.164 [3083314880]     25 05 2C 70 A9 00 01 00 00 00 04 00 00
04/08/09 16:31:14.166 [3083314880] QDL protocol server response received
04/08/09 16:31:14.166 [3083314880]   0003 [8 03/03/99 16:31:14]
04/08/09 16:31:14.166 [3083314880]     26 00 00 01 00 00 00 04
04/08/09 16:31:14.432 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:14.697 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:14.962 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:15.226 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:15.492 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:15.757 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:16.030 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:16.294 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:16.561 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:16.826 [3083314880] QDL protocol server sent 1048576 bytes of image
04/08/09 16:31:16.982 [3083314880] QDL protocol server sent 618540 bytes of image
04/08/09 16:31:16.982 [3083314880] QDL protocol server request sent
04/08/09 16:31:16.982 [3083314880]   0004 [9 03/03/99 16:31:16]
04/08/09 16:31:16.982 [3083314880]     27 00 00 00 00 00 00 2C 70 A9 00 E0 AD
04/08/09 16:31:17.161 [3083314880] QDL protocol server response received
04/08/09 16:31:17.161 [3083314880]   0005 [8 03/03/99 16:31:17]
04/08/09 16:31:17.161 [3083314880]     28 00 00 00 00 00 00 00 00
04/08/09 16:31:17.161 [3083314880] QDL sending image file /opt/Qualcomm/Images2k/<nOEM>/UMTS/apps.mbn
                                                                                     112,78        68%
```

Figure 4-7 QDLService.txt