BSP User's Guide

Generic x86 BIOS and APIC

A Subsidiary of BlackBerry

QNX Software Systems Limited
1001 Farrar Road
Ottawa, Ontario
K2K 0B3
Canada

Voice: +1 613 591-0931
Fax: +1 613 591-3579
Email: info@qnx.com
Web: http://www.qnx.com/

Electronic edition published: May 02, 2016

Contents

About this guide

In this guide

The *BSP User's Guide: Generic x86 BIOS and APIC* contains installation and start-up instructions for QNX Board Support Packages (BSPs) for x86 boards, including the ADI Engineering RCC (Rangeley Communications Collateral) Reference Design Platform.

If you are working with an ADI RCC hardware platform, see "*ADI RCC support*".

If you are working with QNX Neutrino RTOS release 6.6, get the BSP and *BSP User's Guide* for that release.

To find out about:	See:
The resources available to you, and what you should know before starting to work with this BSP	*Before you begin*
What's included in the BSP, supported host OSs and boards, and known issues	*About this BSP*
Installing this BSP	*Installation notes*
Using the Screen Graphics Subsystem	*What's next with Screen*
Driver commands	*Driver commands*
Using the BSP for the ADI Engineering RCC Reference Design Platform	*ADI RCC support*
The `drm-probe-displays` utility for Intel platforms	*drm-probe-displays*

Typographical conventions

Throughout this manual, we use certain typographical conventions to distinguish technical terms. In general, the conventions we use conform to those found in IEEE POSIX publications.

The following table summarizes our conventions:

Reference	Example
Code examples	`if( stream == NULL)`
Command options	`-lR`
Commands	`make`
Constants	NULL
Data types	**unsigned short**
Environment variables	*PATH*
File and pathnames	**/dev/null**
Function names	*exit()*
Keyboard chords	**Ctrl–Alt–Delete**
Keyboard input	`Username`
Keyboard keys	**Enter**
Program output	`login:`
Variable names	*stdin*
Parameters	*parm1*
User-interface components	**Navigator**
Window title	**Options**

We use an arrow in directions for accessing menu items, like this:

You'll find the Other... menu item under **Perspective** --> **Show View**.

We use notes, cautions, and warnings to highlight important messages:

Notes point out something important or useful.

CAUTION: Cautions tell you about commands or procedures that may have unwanted or undesirable side effects.

WARNING: Warnings tell you about commands or procedures that could be dangerous to your files, your hardware, or even yourself.

Note to Windows users

In our documentation, we typically use a forward slash (/) as a delimiter in pathnames, including those pointing to Windows files. We also generally follow POSIX/UNIX filesystem conventions.

Technical support

Technical assistance is available for all supported products.

To obtain technical support for any QNX product, visit the Support area on our website (*www.qnx.com*). You'll find a wide range of support options, including community forums.

Technical support for this Alpha release

If you need help with installing or exploring your QNX Hypervisor, please post your questions (with as much precision and detail as you can) to the QNX Hypervisor *Discussions* on Foundry27, or contact your QNX Project Manager or FAE.

Chapter 1
Before you begin

Before you begin working with this BSP you should become familiar with the resources available to you when building QNX embedded systems.

Board and QNX Neutrino documentation

Before you begin building and installing your BSP, you should review the board hardware and firmware documentation provided by the board vendor.

The QNX Neutrino RTOS 6.5.0 SP1 documentation is available on the QNX website at *www.qnx.com/developers/docs/index.html*. In particular, see *Building Embedded Systems*.

Technical Support

To obtain technical support for any QNX product, visit the *Support* area on our website. You'll find a wide range of support options, including community forums.

Latest version of this BSP

For the most up-to-date version of the BSPs and of this user's guide, log in to your *myQNX account*, and download it. You can check the build number in the BSP archive name, and the publication date in the guide to see if the BSP or the guide have been updated.

Chapter 2
About this BSP

These notes list what's included in the BSP, and identify the host OSs and boards it supports.

What's in this BSP

This BSP contains the following components:

Component	Format	Comments
Startup	Source	x86 startup library and **startup-apic**
Serial driver	Source	**devc-ser8250** (16550-compatible UART driver)
PCI server	Source	**pci-bios** and **pci-bios-v2**
Network driver	Source	**devnp-e1000.so** gigabit Ethernet driver
USB	Binary only	EHCI USB driver
Intel HD audio driver	Source	Driver for Intel HD audio interface and mixer (codec support)
SD/MMC	Source	SD/MMC disk driver

In addition, you should (and likely did) download this *User's Guide* from the same location as the BSP archive.

Supported OSs

This BSP for the x86 platforms supports the following target OS:

- QNX Neutrino RTOS 6.5.0 SP1

In order to install and use this BSP, you must have installed on your host system the QNX Software Development Platform (SDP) for the QNX Neutrino RTOS release 6.5.0 SP1.

You can install the SDP on either a Windows or Linux host system. With these host environments, you also need a terminal emulation program (Qtalk, QNX Momentics IDE Terminal, tip, Hyperterminal, etc.).

Supported boards

This BSP supports the following boards:

- x86 (generic)
- ADI RCC platform (see "*ADI RCC support*")

Known issues

None.

Chapter 3
Installation notes

These installation notes describe the tasks you must complete to get a QNX Neutrino 6.5.0 SP1 system running on an x86 hardware platform.

Building and installing the Generic x86 BSP requires the followings tasks, completed in order:

1. Connect the hardware (see below).
2. Download and extract the BSP (see "*Download and extract the BSP*").
3. Build the BSP and prepare the IFS image (see "*Build the BSP*").
4. Create a bootable USB device (see "*Prepare a bootable image*").
5. Transfer the QNX IFS image to the USB device ("*Transfer the IFS image*").
6. Insert the USB device and power up the board (see "*Boot the target*").

The **/images** directory is either:

- if you are working in the commandline, the BSP's **/images** directory

  or

- if you are working in the IDE, the **/images** directory in your BSP project in the IDE's workspace (e.g., **C:\ide-4.7-workspace\intel-atom-c2000-x86-650SP1\Images**, on Windows)

Connect the hardware

WARNING:

Use only the power supply provided with the board. Use of any other power supply may permanently damage the board. When power cycling the board, turn the power on or off from the AC adapter side, rather than removing and inserting the DC power plug on the board, to prevent damaging the board's power-regulation circuitry.

To connect to your board:

1. Connect the board COM1 serial port to your host machine.
2. Connect the board to its power supply.
3. On your host machine, start your favourite terminal program with these settings:
   - Baudrate: **115200**
   - Data: **8 bit**
   - Parity:**no**
   - Stop: **1 bit**
   - Flow control: **none**

Modifying the IFS image

By default, this BSP generates a BIOS-compatible boot image. This image can be found in the BSP's **/images** directory. If you need to modify this BSP, you can edit the ***.build** buildfile found in the same directory.

An **/images** directory may contain both generic and board-specific build files and binaries (as well as the `Makefile`. For example:

```
adi-rcc-c2000.build
ifs-adi-rcc-c2000.bin
ifs-x86-generic.bin
Makefile
x86-generic.build
```

Download and extract the BSP

You can download a BSP from our QNX website, then extract it to a working directory of your choosing or import it into the IDE.

BSPs are provided in a `zip` archive, which you can download from the QNX website (*www.qnx.com*). After you have downloaded the BSP archive to a convenient location, to begin using it can either:

- `unzip` it from the command line

  or

- import it into the IDE

The `unzip` utility ships with the QNX SDP, for all supported host platforms.

Extract from the command line

We recommend that you create a default directory with the same name as your BSP and `unzip` the archive from there:

1. Change the directory to where you want to extract the BSP (e.g., **/home/bspdir**). The archive will be extracted to the current directory, so you should create a directory specifically for your BSP. For example:

   ```
   mkdir /home/bspdir/atom_c2000
   ```

2. In the directory you've just created, extract the BSP. For example:

   ```
   cd /home/bspdir/atom_c2000
   unzip BSP_x86-generic_br-650_be-650sp1_SVNbuild.zip
   ```

   where *build* is the BSP build number (e.g., `800138_JBN20`).

You should now be ready to build your BSP (see "*Build the BSP*").

Import to the IDE

To import the BSP code, from the IDE:

1. If not already opened, open the workbench from IDE. From the C/C++ Perspective, select **File → Import**.
2. Expand the **QNX** folder.
3. Select **QNX Source Package and BSP** from the list. Click **Next**.
4. In the **Select the package to import** dialog, select the **Import from local archive file** radio button. Then click **Browse...** to choose the BSP archive that you've downloaded by using the file browser. Once you've chosen your BSP archive, click **Next**.
5. Confirm that this is the BSP package you want in the **Package selected to import** dialog. There's a brief description of the package. Click **Next** to proceed with this package.
6. Set the projects to be created. Specify a directory for your projects as well as a project prefix. Defaults are provided, but you can override them if you choose.
7. Click **Finish**. All the projects are created and the source brought from the archive.

You should now be ready to build your BSP (see "*Build the BSP*").

Structure of a BSP

After you unzip a BSP archive, the resulting directory structure looks something like this:

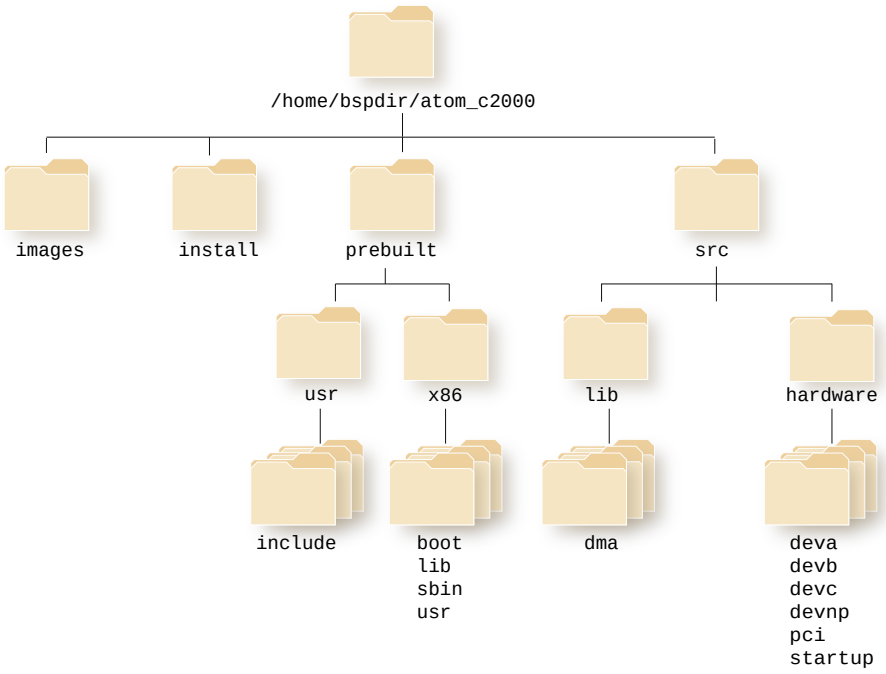

Figure 1: Structure of a BSP

Build the BSP

You can use the QNX Momentics IDE or the command line on Linux, Windows or QNX self-hosted to work with this BSP.

Screen doesn't support QNX self-hosted.

Generic instructions on how to extract and build a BSP can be found in the *Building Embedded Systems* guide, available as part of the QNX Software Development Platform OS Core Components documentation. For instructions on using the IDE, see the *QNX Momentics User's Guide*. This documentation is available in the QNX *Infocentre*. Make sure you refer to the documentation for QNX Neutrino 6.5.0 SP1.

If you plan to work with multiple, different BSPs, we recommend that you create a top-level BSP directory, then create subdirectories for each different BSP. The directory you create for each BSP will be that BSP's root directory.

To build your BSP, complete the following tasks:

1. Build the BSP, either from the command line or in the IDE.
2. Prepare the IFS.

After you have unzipped the BSP, you may notice prebuilt IFSs files, such as **ifs-adi-rcc-c2000.bin**, in the BSP's **/images** directory. You need to run `make clean` to remove these files before you build the BSP in the command line. You don't need to remove it if you will use the IDE to build the BSP, however; the IDE looks after cleaning up before the build.

Support for systems with or without Screen

Screen isn't included in QNX Neutrino 6.5.0 SP1. If you plan on using Screen, you must download and install it before you build the BSP.

BSPs for specific x86 platforms that support Screen (e.g., NEXCOM VTC 1010) include two buildfiles under the **/images** directory. One buildfile generates a target image with Screen; the other generates a target image without Screen. The buildfile that includes Screen has **-graphics** in its file name (e.g., **nexcom-vtc1010-graphics.build**).

This BSP's default buildfile (e.g., **nexcom-vtc1010-graphics.build**) attempts to build a BSP that includes Screen. If you don't have Screen installed on your host and use this buildfile, the build will display numerous warnings, but it should complete properly (without Screen) and start. For example:

```
Warning: Unable to find libgestures.so.1 in search paths.
      Item target location: /proc/boot
      Item target name: libgestures.so.1
...
Warning: Host file 'libGAL.so.1' missing.
Warning: Host file 'libGalcore.so' missing.
...
Starting watchdog...
```

```
Starting I2C1,2,3 driver (/dev/i2c1,2,3)...
Starting SD3 memory card driver...
Starting Emmc Nand Flash driver
...
done
Starting Ethernet driver
Starting Audio driver
Unable to start "screen" (2)
```

The next time you build the BSP for a system without Screen, use the buildfile without the **-graphics** suffix. It doesn't look for Screen components.

Build the BSP (command line)

You can use the command line, on either Linux or Windows, to work with this BSP.

The **Makefile** is located under the BSP's **/images** directory. It defines more than one target:

all

Invokes both targets.

`ifs-x86-generic`

Invokes the generic build for an x86 platform.

`ifs-adi-rcc-c2000`

Invokes the build for the ADI RCC platform.

If you don't specify a target, `make` invokes the `all` target.

Build from the command line

To build the BSP using the command line, open a terminal, then:

1. Download the BSP archive, and unzip it (see "*Extract from the command line*").
2. In the BSP's root directory, type `make clean` to remove the default image files from the BSP's **/images** directory.

   This action removes both default IFS images (the one with Screen and the one without Screen).
3. In the BSP's root directory, type `make` to build the BSP, and create its directories. This command:
   - builds the BSP and creates its directories
   - places the IFS images in the BSP's **/images** directory

You should now have IFS files called **ifs-x86-generic.bin** and **ifs-adi-rcc-c2000.bin**.

We recommend that you use `make` to build the OS image. If you use `mkifs` directly, you need to use the `-r` option to specify where to find the binaries.

Build the BSP (IDE)

You can use the QNX Momentics IDE, on either Linux or Windows, to work with this BSP.

Build in the IDE

If you are using an ADI RCC target, build from the command line (see "*Build the BSP (command line)*"). Building from the IDE will generate only the **ifs-x86-generic.bin** binary, and *not* the **ifs-adi-rcc-c2000.bin** binary you need for your board.

To build the BSP in the QNX Momentics IDE, launch the IDE, then:

1. Download the BSP archive, and import it into the IDE (see "*Import to the IDE*").

   After you've imported your BSP source into IDE, you're ready to build.
2. With your project opened in the C/C++ Perspective, right-click on the BSP source project and select **Build Project**.
3. Switch from the C/C++ Perspective to the QNX System Builder Perspective.
4. Select and open **project.bld** from under your BSP project.
5. Under **Images**, right-click on the project and select **Build**.
6. The IDE starts building your selected image. You can check in the Console and Problems tabs in your perspective to monitor any issues with the build. After your image is built, you can see it as shown in the figure below:

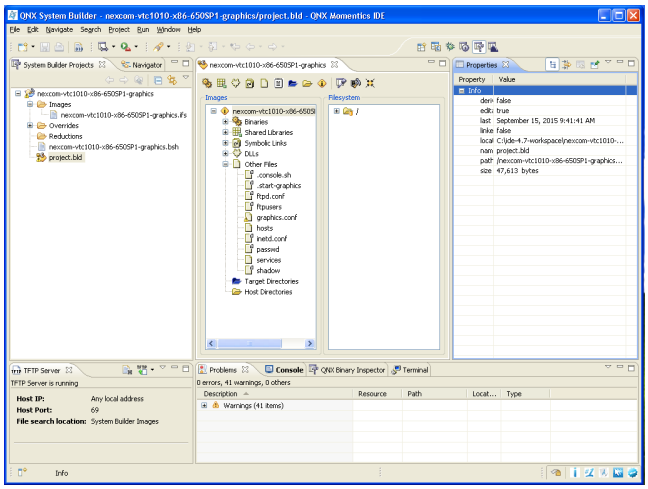

Figure 2: Built IFS

You should now have an IFS file.

Prepare a bootable image

You must prepare a bootable USB device to which you can transfer your IFS.

If you are working with an ADI RCC hardware platform, see "*ADI RCC support*".

If you are working on a Windows host, open a terminal and run the `bash` script. All commands should be made inside `bash`.

Patches

To create a bootable USB image, you must have the following patches installed on your host system:

- QNX Software Development Platform 6.5.0 SP1: diskimage Patch [patch ID 4245]
- QNX Software Development Platform 6.5.0 SP1 mkxfs Patch [Patch ID 4392]

You can download these patches from the QNX Download Centre (*www.qnx.com/download/*).

Prepare the image

To create a bootable USB image:

1. Go to the BSP's **/images** directory. If you are in the BSP root directory:

   ```
   cd images
   ```

2. Create a plain text file **root.build** file to describe the root file system contents for your USB disk image. This file should have the following contents, where *board* is the name of your hardware platform (e.g., `adi-rcc-c2000`):

   ```
   [num_sectors=524288]
   .boot/ifs-board.bin=ifs-board-graphics.bin
   # .boot/ifs-board.bin=ifs-board.bin
   ```

   If you are building a system without Screen, use this:

   ```
   [num_sectors=524288]
   # .boot/ifs-board.bin=ifs-board-graphics.bin
   .boot/ifs-board.bin=ifs-board.bin
   ```

3. Create the QNX6 partition image:

   ```
   mkxfs -t qnx6fsimg root.build partition.img
   ```

4. Create a plain text file **disk.cfg** with the following contents that describe the partition layout of your USB disk image:

   ```
   [heads=64]
   [sectors_per_track=32]
   [sector_size=512]
   [cylinders=513]
   [start_at_cylinder=1]
   [partition=1 type=177 num_sectors=524288 boot=true]
   "partition.img"
   ```

5. Create the USB disk image:

   ```
   diskimage -o usb.img -c disk.cfg -b $QNX_TARGET/x86/boot/sys/ipl-diskpc1
   ```

On a Windows host:

```
diskimage -o usb.img -c disk.cfg -b %QNX_TARGET%/x86/boot/sys/ipl-diskpc1
```

You should now have a bootable USB image.

Transfer the IFS image

After you have prepared a bootable USB device, transfer the IFS image to this device.

Transfer (Linux)

On a Linux system, to transfer the boot image to your bootable USB device, from the command line, where *path* is the path to your USB device:

- If your version of Linux supports the `sudo` command, enter:

  ```
  sudo dd if=usb.img of=path
  ```

- If your version of Linux does *not* support the `sudo` command, enter:

  ```
  su -c 'dd if=usb.img of=path'
  ```

When the copy operation is complete, your USB device should be ready to *boot QNX* on your x86-based target.

Transfer (Windows)

On a Windows system, to transfer the boot image to your bootable USB device:

1. Download and install Win32 Disk Imager from *http://sourceforge.net/projects/win32diskimager/*
2. Run Win32 Disk Imager and select:
   - the USB disk image
   - the USB device
3. Click **Write** to copy the image to the USB device.

When the copy operation is complete, your USB device should be ready to *boot QNX* on your x86-based target.

Boot the target

After you have built your BSP and installed it on your target, you can boot the QNX Neutrino release you installed.

To boot QNX Neutrino on your x86-based target:

1. Remove the USB device from your host machine, and insert it into a USB port on your target. On the board, the USB 3.0 connection is often beside a flashing light.
2. Power up the target, and enter the BIOS, to configure the machine to boot from the USB drive that you've just inserted.
3. After configuring the BIOS to use the USB drive as the primary boot device, reboot the machine.

You should see your target board boot into the QNX environment and launch `screen`.

ADI RCC support

The QNX Neutrino 6.5.0 SP1 BSP for x86 platforms includes support for the ADI Engineering RCC (Rangeley Communications Collateral) Reference Design Platform.

For information about the hardware setup, jumper and DIP switch settings, peripheral connections, etc., please refer to the manufacturer's reference manual, which can be found at the following location:

www.adiengineering.com/products/rcc/

The addition of ADI RCC support hasn't changed the name of the zip files for the QNX Neutrino 6.5.0 SP1 BSP for x86 platforms. These files are still called:

`BSP_x86-generic_br-650_be-650sp1_SVN`*build*`.zip`

where *build* is the BSP build number.

About ADI RCC support

The QNX BSP for x86 platforms includes an IFS build file specifically for the ADI RCC platform (**adi-rcc-c2000.build**, as well as the build file for the generic IFS (**x86-generic.build**).

This platform-specific build file is needed because the ADI RCC doesn't ship with an on-board graphics controller. The absence of a graphics controller means that when QNX boots the default debug output that is normally displayed to a VGA monitor must be redirected to another device .

The ADI RCC routes the boot information from the BIOS out to a USB-to-Dual UART console, via the mini-USB “CON” port on the RCC's rear panel. Similarly, the QNX BSP's build file for the ADI RCC platform routes the debug output from the startup code, as well as subsequent output from the QNX kernel and serial driver, to this UART port.

Included build files

Two build files are included in the BSP's **/images** directory:

x86-generic.build

Invokes the generic build for an x86 platform.

adi-rcc-c2000.build

Invokes the build for the ADI RCC platform.

When you build a BSP without specifying a target, two IFS images are created: **ifs-x86-generic.bin** and **ifs-adi-rcc-c2000.bin** (see “*Build the BSP (command line)*”). You can use the **ifs-adi-rcc-c2000.bin** IFS image to boot the RCC platform into QNX 6.5.0 SP1.

Transferring the IFS image and booting

Booting QNX Neutrino on an x86 target, is usually a matter of putting the generated boot image onto a bootable USB device, such as a USB stick, then connecting this device to the target. Before you can

boot the ADI RCC platform from a USB device, in the ADI RCC BIOS, ensure that you have the following values set:

- `Advanced > CSM Settings > Legacy Mode Only`
- `USB Configuration > Mass Storage Device > Enabled`

For information about how to create a bootable USB device, see "*Prepare a bootable image*".

Chapter 4
What's next with Screen

If your x86 hardware platform supports Screen, you should now be ready to run and create your own Screen applications.

Running Screen applications

A set of demo Screen applications are provided and are built into your image if you've used the BSP archives recommended for Screen.

It's a good idea to run some of the demos on your target to confirm that Screen is running and that all necessary drivers are in place and can be started. There are several demos available, but if you can run the following on your target, then you're off to a good start:

`sw-vsync`

Shows `vsync` that uses software rendering.

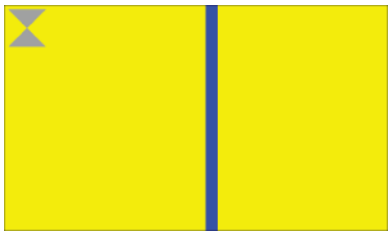

Figure 3: sw-vsync

`gles2-gears`

Shows gears that use OpenGL ES 2.X for the rendering API; if `gles2-gears` runs, then it confirms that `screen` and drivers necessary for OpenGL ES has started successfuly.

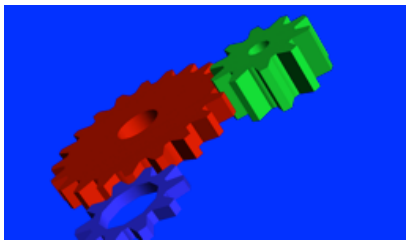

Figure 4: gles2-gears

If you encounter problems when running these applications, see the “Debugging” section of the *Screen Developer's Guide*.

Creating Screen applications

Now you can create your own Screen applications. However, we recommend that you not start until you've read the *Screen Developer's Guide*.

Your Screen application must link against **libscreen**. This API enables communication between your application and Screen.

In addition to **libscreen**, other libraries you need to link against depend on your application. Here are some libraries that are common to most Screen applications:

libEGL

The interface to access vendor-specific EGL functionality. Refer to the EGL standard from *www.khronos.org*.

libGLESv2

The interface to access vendor-specific OpenGL ES functionality. Refer to the OpenGL ES standard from *www.khronos.org*.

libimg

The interface to load and decode images. Refer to the "QNX Image Library Reference" section in the *QNX Neutrino Advanced Graphics Developer's Guide*.

The "QNX Image Library Reference" section of the *QNX Neutrino Advanced Graphics Developer's Guide* is the *only* section that's valid for Screen applications.

Depending on your choice of development environment, refer to either the "Extra libraries" section of the *IDE User's Guide*, or the "Compiling and Debugging" section of the *Programmer's Guide* for more information about using libraries.

drm-probe-displays

Detect ports and pipelines for Intel graphics hardware (available for Intel platforms only)

Syntax:

```
drm-probe-displays
```

Runs on:

QNX Neutrino

Options:

None.

Description:

The `drm-probe-displays` utility is a command-line tool that detects ports and pipelines for Intel graphics hardware. This utility lists the available displays and pipelines.

The information retrieved from this utility can be used to modify **graphics.conf** for a specific target. For more information about configuring **graphics.conf** for Intel graphics hardware, refer to the usage message of **libWFDintel-drm.so**, and to the *Screen Developer's Guide*.

To invoke drm-probe-displays on your target:

1. Ensure that `screen` is running.
2. Run `drm-probe-displays` from a shell.

Examples:

Query displays and pipelines on the target, such as a Nexcom VTC-1010:

```
# drm-probe-displays
count_displays : 3
count_pipelines: 2

display 1: VGA, connected
Mode: "1280x1024" 1280x1024 60
Mode: "1280x1024" 1280x1024 75
Mode: "1152x864" 1152x864 75
Mode: "1024x768" 1024x768 75
Mode: "1024x768" 1024x768 60
Mode: "800x600" 800x600 75
Mode: "800x600" 800x600 60
Mode: "640x480" 640x480 75
Mode: "640x480" 640x480 60
Mode: "720x400" 720x400 70
```

```
display 2: HDMI-A, disconnected
display 3: displayport, connected
Mode: "1280x720" 1280x720 60
Mode: "1920x1080" 1920x1080 60
Mode: "1920x1080" 1920x1080 60
Mode: "1920x1080i" 1920x1080 60
Mode: "1920x1080i" 1920x1080 60
Mode: "1920x1080" 1920x1080 50
Mode: "1920x1080i" 1920x1080 50
Mode: "1280x1024" 1280x1024 75
Mode: "1440x900" 1440x900 85
Mode: "1440x900" 1440x900 75
Mode: "1440x900" 1440x900 60
Mode: "1280x720" 1280x720 60
Mode: "1280x720" 1280x720 50
Mode: "1024x768" 1024x768 75
Mode: "1024x768" 1024x768 70
Mode: "1024x768" 1024x768 60
Mode: "800x600" 800x600 75
Mode: "800x600" 800x600 72
Mode: "800x600" 800x600 60
Mode: "800x600" 800x600 56
Mode: "720x576" 720x576 50
Mode: "720x480" 720x480 60
Mode: "720x480" 720x480 60
Mode: "640x480" 640x480 75
Mode: "640x480" 640x480 73
Mode: "640x480" 640x480 60
Mode: "640x480" 640x480 60
Mode: "720x400" 720x400 70

pipeline 1
pipeline 2
```

Chapter 5
Driver commands

The tables below provide a summary of driver commands.

For more information about these and other commands, see the *Neutrino Utilities Reference*.

Startup

Device	STARTUP
Command	`startup-apic`
Command	`startup-apic`
Required binaries	**startup-apic**
Required libraries	**libstartup.a**
Source location	**src/hardware/startup/boards/apic**

PCI

Both **pci-bios** and **pci-bios-v2** are included. See *Build the BSP* for details.

Device	Intel Host/PCI bridge
Command	`pci-bios-v2` (for use with **startup-apic**)
Required libraries	
Source location	**src/hardware/pci**

Serial

Device	SERIAL
Command	`devc-ser8250 -e -b115200 3f8,4 2f8,3`
Required binaries	**devc-ser8250**
Required libraries	
Source location	**src/hardware/devc/ser8250**

Ethernet

Device	NETWORK (Intel gigabit Ethernet driver)
Command	`io-pkt-v6 -de1000`
Required binaries	**io-pkt-v6**, **ifconfig**, **dhcp.client**
Required libraries	**devnp-e1000.so**
Source location	**src/hardware/devnp/e1000**

USB

Make sure that the BIOS USB mode is set to “EHCI Mode”.

Device	EHCI USB driver
Command	`io-usb -dehci pindex=0,irq=23 &`
Required binaries	**io-usb**
Required libraries	**devu-ehci.so**
Source location	Prebuilt binary only

HD audio

Device	Intel HD audio controller
Command	`io-audio -d intel_hda`
Required binaries	**io-audio**
Required libraries	**libasound.so**, **deva-ctrl-intel_hda.so**, **deva-mixer-hda.so**
Source location	**src/hardware/deva/**

For more information about using the Intel HD audio driver, please refer to the QNX Neutrino RTOS SDP 6.5.0 SP1 *Utilities Reference* entries for:

- **io-audio**
  http://www.qnx.com/developers/docs/6.5.0SP1/neutrino/utilities/i/io-audio.html
- **deva-ctrl-intel_hda.so**
  www.qnx.com/developers/docs/6.5.0SP1/neutrino/utilities/d/deva-ctrl-intel_hda.so.html
- **deva-mixer-hda.so**
  http://www.qnx.com/developers/docs/6.5.0SP1/neutrino/utilities/d/deva-mixer-hda.so.html

SD/MMC

Device	SD/MMC
Command	`devb-sdmmc cam pnp sdio idx=0,vid=0x8086,did=0xf16`
Required binaries	**devb-sdmmc**
Required libraries	
Source location	**src/hardware/devb/sdmmc**

Index

M

O

P

R

S

T

U

V

X