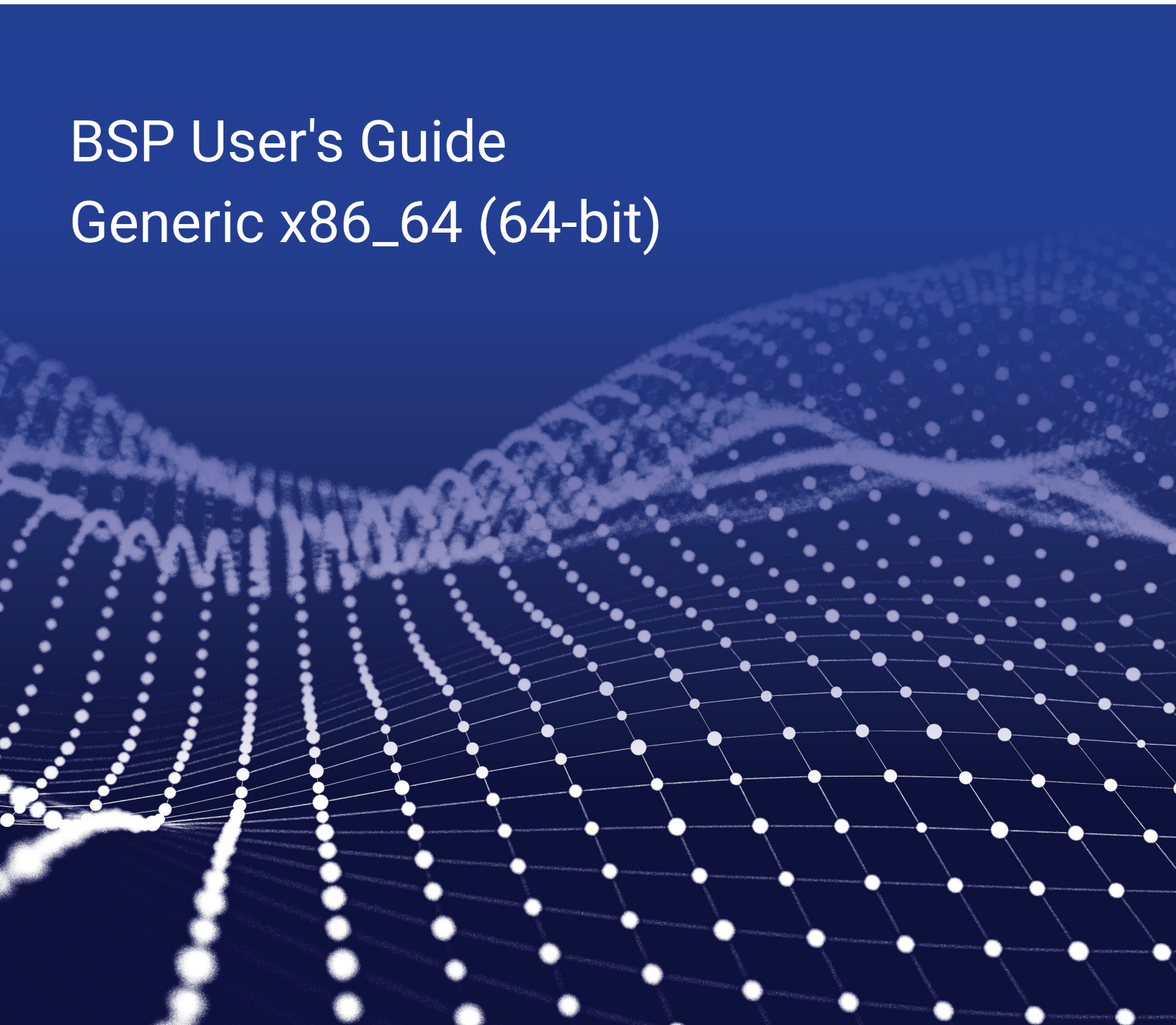




QNX® Software Development Platform 8.0

BSP User's Guide

Generic x86_64 (64-bit)



Contents

- About This Guide..... iv**
 - Typographical conventions..... v
 - Technical support..... vii
- Before You Begin..... 8**
- About This BSP..... 9**
- Installation Notes.....12**
 - Download and set up the BSP.....13
 - Working with the hardware..... 17
 - Create and transfer an image to a bootable USB device..... 18
 - Create and transfer a bootable disk image (offline method).....19
 - Create a bootable USB device (interactive method using VMware).....21
 - Boot the target..... 26
- Build the BSP.....27**
 - Build the BSP (commandline).....29
 - Build the BSP (IDE)..... 32
- Generic x86_64_Driver Commands.....34**
- Working with Intel Elkhart Lake CRB platform.....39**
 - Intel Elkhart Lake CRB driver commands..... 42

Copyright and patent notice

Copyright © 2023, BlackBerry Limited. All rights reserved.

BlackBerry Limited
1001 Farrar Road
Ottawa, Ontario
K2K 0B3
Canada

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

November 28, 2023

Trademarks, including but not limited to BLACKBERRY, EMBLEM Design, QNX, MOMENTICS, NEUTRINO, and QNX CAR, are the trademarks or registered trademarks of BlackBerry Limited, its subsidiaries and/or affiliates, used under license, and the exclusive rights to such trademarks are expressly reserved. All other trademarks are the property of their respective owners.

Patents per 35 U.S.C. § 287(a) and in other jurisdictions, where allowed: <https://www.blackberry.com/patents>

About This Guide

This guide contains installation and start-up instructions for the QNX Board Support Package (BSP) for x86_64 platforms; x86 (32-bit) isn't supported in this release.

At times, there's specific information for variants of the x86_64 platforms based on a particular processor family or SoC. You should refer to information in those chapters if your board uses that processor family or SoC.

This documentation includes:

- an overview of QNX BSP
- links to documentation about how to build QNX embedded systems, which you should read before you begin working with this BSP
- what's new in the BSPs for this release
- the structure and contents of the BSP
- how to prepare a bootable USB device (e.g., USB memory stick)
- how to install the BSP and boot it on a board
- how to build your BSP

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 operating systems, and boards	About This BSP
Installing this BSP	Installation Notes
Building this BSP	Build the BSP
Driver commands	Generic x86_64_Driver Commands
Working with Intel Elkhartlake CRB platform	Working with Intel Elkhart Lake CRB platform

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	<code>if(stream == NULL)</code>
Command options	<code>-lR</code>
Commands	<code>make</code>
Constants	<code>NULL</code>
Data types	<code>unsigned short</code>
Environment variables	<code>PATH</code>
File and pathnames	<code>/dev/null</code>
Function names	<code>exit()</code>
Keyboard chords	Ctrl–Alt–Delete
Keyboard input	Username
Keyboard keys	Enter
Program output	login:
Variable names	<code>stdin</code>
Parameters	<code>parm1</code>
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:



NOTE: 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.



ALERT: Alerts tell you about commands or procedures that could expose private information or otherwise compromise security.

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 (<https://blackberry.qnx.com/en/support>). You'll find a wide range of support options.

Before You Begin

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

Essential information

Before you begin to install or build your BSP, review the following documentation:

- Information about your board's hardware and firmware provided by the board vendor. This differs for each board.
- *Building Embedded Systems*. This guide contains general information about working with BSPs from QNX Software Systems that's common to all BSPs and is included in QNX Software Development Platform 8.0 documentation.

Required software

To work with the BSP for the x86_64 board, you require the following:

- the ZIP file for the BSP
- QNX Software Development Platform 8.0 installed on a Linux or Windows host system
- Virtual COM Port (VCP) Driver. This driver is required on your host system if your board has a COM or USB-Serial port for debugging. To get the driver and read more information about it, see the Future Technology Devices International Ltd. (FTDI) website at <http://www.ftdichip.com/FTDrivers.htm>
- a terminal emulation program (e.g., Qtalk, Hyperterminal, Tera Term, PuTTY) to connect to the board for debugging. Alternatively, you can use a **ssh** session from the QNX Momentics IDE.

About This BSP

This chapter lists what's included in the BSP, the supported host system operating systems, and other relevant information to use this BSP.

Embedding QNX SDP 8.0 on targets

QNX SDP 8.0 (like QNX SDP 7.0 and QNX SDP 6.6) doesn't support a self-hosted option. If you've worked with previous QNX SDP versions, such as QNX SDP 6.5.x, which supported a self-hosted development environment, the self-hosted option is available.

QNX Software Systems provides a QNX virtual machine by the QNX Momentics IDE. The virtual machine provides a collection of the standard x86_64-based tools and utilities to explore the QNX OS. The tools provided with this virtual machine can be used to put QNX OS on an actual target. For more information about creating a QNX virtual machine, see https://docserver.bts.rim.net/mainlinewebsite/com.qnx.doc.ide.userguide/topic/creating_qnx_vm.html.



NOTE: You can still install QNX SDP 8.0 on a PC, but an installation CD-ROM to configure a self-host, GUI-hosted development is no longer available. QNX OS 8.0 on x86-64-based (64-bit) hardware is treated as target and no different from an embedded target.

Why a generic x86_64 BSP?

Unlike other QNX SDP 8.0 BSPs, which are targeted to a specific board, the BSP isn't intended to be an exact match to any particular x86_64 (64-bit) platform. That being said, we do in some cases provide steps for some specific boards.

There are many different models of x86_64-based PCs and targets, each with different chipsets, and each with its own collection of peripheral devices, such as USB interfaces, network devices, and graphics controllers. QNX SDP 8.0 includes startup modules, device drivers, and general support for a wide variety of different x86_64 chipsets and peripherals. In most cases, these components can be run as-is on x86_64 platforms; it's simply a matter of selecting the correct device drivers and utilities for the specific hardware, and creating a custom bootable image for that board. In case there is modification required, this BSP provides the source code for startup modules and some device drivers, which you can modify as needed.

This BSP user guide provides instructions for:

- generating a QNX Image File System (IFS) bootable image for your target
- preparing a bootable drive image (USB memory stick or other boot media) you can use to boot your target
- transferring your IFS image to your target

What's in this BSP

This BSP contains the following components; other components used in this BSP are provided as part of QNX SDP 8.0:

Component	Format	Comments
Startup	Source	
PCI	Binary	
I2C	Source	
SPI	Source	
Serial driver	Binary	
SD/MMC	Source	
SATA	Binary	
NVMe	Binary	
USB	Binary	
Network	Binary	
Hypervisor	Source	shmem-host and shmem-linux
NOR flash	Binary	devf-ram for storage filesystem



NOTE: This BSP supports the `/dev/umass0t179` USB partition as an optional persistent filesystem. It detects whether the `/dev/umass0t179` partition exists on target board at system startup and uses it as a filesystem to save your executable files and data. If it does not detect the required partition, the system starts and uses the devf-ram NOR flash instead. See the `/proc/boot/.storage-server.sh` script in the buildfile for details.



NOTE: This BSP contains support for QNX Hypervisor 3.0 (and later versions supported by QNX SDP 8.0). In order to make use of hypervisor support, it is required to install a compatible QNX Hypervisor product in your installation environment. The Hypervisor-supported boot image does not build by default, see [Build the BSP](#) chapter for more details how to build the Hypervisor boot image (Makefile target “hyp”).



NOTE: Binaries for the PCI server, serial driver, network driver, USB driver, SATA driver and NVMe driver are provided as part of the QNX SDP 8.0 installation.

Supported OSs

For QNX OS 8.0, this BSP only supports x86_64 (64-bit) platforms.

Before you install and use this BSP, you must install the QNX Software Development Platform 8.0 (QNX SDP 8.0) on your host system. You can install the QNX SDP 8.0 on a Windows or Linux host system.

Supported boards and chipsets

In the QNX Software Center, see the release notes for this BSP for the list of supported boards using these steps:

1. On the **Available** or **Installed** tab, navigate to Board Support Packages BSP, right-click *Name_of_your_BSP*, and then select **Properties**.
2. In the **Properties** window, on the **General Information** pane, click the link beside **Release Notes**.

You should be redirected to the release notes on the QNX website as long as you're logged in with your myQNX account.

Known issues

For the list of known issues, see the release notes for this BSP.

Installation Notes

These installation notes describe how to install and boot a board using this BSP.

About installing a BSP

Here's the process to install the BSP and run it on your board:

1. Download the BSP, set up your environment, and extract the BSP. For more information, see [“Download and set up the BSP.”](#)
2. Connect your target board. After you connect the hardware, it may be necessary set DIP switches on your board to further configure it. For more information about how to connect to your board, see [“Working with the hardware.”](#)
3. [Create and transfer an image to a bootable USB device.](#) If this is the first time you're bringing up the BSP, you can transfer the provided QNX IFS image to a USB device to boot your board. You can either transfer a bootable USB disk image (offline method) or create a bootable USB device using VMware.

Depending on the method you use to prepare a bootable USB device, follow the instructions in either [“Create a bootable USB device \(interactive method using VMware\)”](#) or [“Create and transfer a bootable disk image \(offline method\).”](#) The two methods differ in how the bootable USB device is created, and how the IFS image is transferred to the USB device.

4. After the USB device has a bootable partition and QNX IFS image, you can insert it into a USB port on the x86_64 target, and use it to boot the target to run the QNX OS. For more information about booting your board, see [“Boot the target.”](#)

After your board boots, you can use that image to understand how it works. Depending on the other devices you've connected to the board, you may need to modify configuration settings on the image.

At some point, you may want to customize the image, rebuild it, and then put the image onto a bootable USB device. For information about how to create a bootable USB device and how to customize the image, see the *Building Embedded Systems* guide, which is available in the QNX SDP 8.0 documentation. For specific information about building the image for this BSP, see the [“Build the BSP”](#) chapter in this guide.

Download and set up the BSP

You can download Board Support Packages (BSPs) using the QNX Software Center.

BSPs are provided in a ZIP archive file. You must use the QNX Software Center to install the BSP, which downloads the BSP archive file for you into the **\$QNX_BASE/bsp** directory where *QNX_BASE* represents the directory where your QNX SDP 8.0 installation is located. To set up your BSP, you can do one of the following steps:

- Import the BSP archive file into the QNX Momentics IDE (extracting the BSP isn't required)
- Navigate to where the BSP archive file has been downloaded, and then extract it using the `unzip` utility on the commandline to a working directory that's outside of your QNX SDP 8.0 installation.



NOTE: The `unzip` utility is available on all supported host platforms and is included with your installation of QNX SDP 8.0.

The installation and build paths that are used in this BSP can function independently as long as you have the QNX Software Development Platform 8.0 installed.

Extract from the commandline

We recommend that you create a directory named after your BSP and extract the archive from there:

1. In a Command Prompt window (Windows) or a terminal (Linux), run **qnxsdp-env.bat** (Windows) or source **qnxsdp-env.sh** (Linux) from your QNX SDP 8.0 installation to set up your environment. You must run the shell or batch file to use the tools provided with QNX SDP 8.0.



NOTE: If you type `env`, you should see environment variables that are set, such as *QNX_TARGET*.

2. Create and navigate to the directory where you want to extract the BSP (e.g., `cd /BSPs/bsp_x86_64-generic`). The directory where you extracted the BSP is referred to throughout this document as *bsp_working_dir*. The archive is extracted to the current directory, so you should create a directory specifically for your BSP. For example:

```
$ mkdir bsp_working_dir
$ cd bsp_working_dir
```

You should also set the *BSP_ROOT_DIR* environment variable to point to your *bsp_working_dir*. You can use the `export BSP_ROOT_DIR= bsp_working_dir` (Linux) or `set BSP_ROOT_DIR= bsp_working_dir` (Windows) command to set the environment variable.

3. Use the `unzip` command to extract the ZIP file that was downloaded by the QNX Software Center for this BSP to the directory you had created in the previous step. For example:

```
$ unzip -d bsp_working_dir $QNX_BASE/bsp/BSP_x86_64_br-hw-rel_be-800 build_ID.zip
```

Where *build_ID* is the BSP build number (e.g., SVN986660_JBN13).

You should now be ready to build your BSP (see “[Build the BSP](#)” chapter).

Import to the QNX Momentics IDE

If you use the QNX Momentics IDE to build your BSP, you don't need to extract the ZIP file. To import the BSP code into the IDE:

1. Open the QNX Momentics IDE.
2. From the **C/C++** Perspective, right click in **Project Explorer** view, select **Import**.
3. Expand the **QNX** folder.
4. Select **QNX Source Package and BSP** from the list and then click **Next**.
5. In the **Select the package** dialog box, click **Browse for ZIP Archive....**
6. In the **Open** dialog box, navigate to and select the BSP archive file that you downloaded earlier, and then click **Open** and then click **Next**.
7. Confirm the BSP package you want is shown in the **Selected Package** list and then click **Next** to proceed importing the BSP archive file.
8. Optionally, set the **Project Name** and **Location** to import the BSP archive file. Defaults are provided for the project name and location, but you can override them if you choose.
9. Click **Finish**. The project is created, and the source brought from the archive.

You should see the project in the **Project Explorer** view (e.g., **hw.x86_64**). You can now build your BSP. For more information, see the “[Build the BSP \(IDE\)](#)” section in the “[Build the BSP](#)” chapter of this guide.

Structure of a BSP

After you unzip a BSP archive using the commandline, or if you look at the source that's extracted by the IDE in your workspace, the resulting directory structure looks something like this:

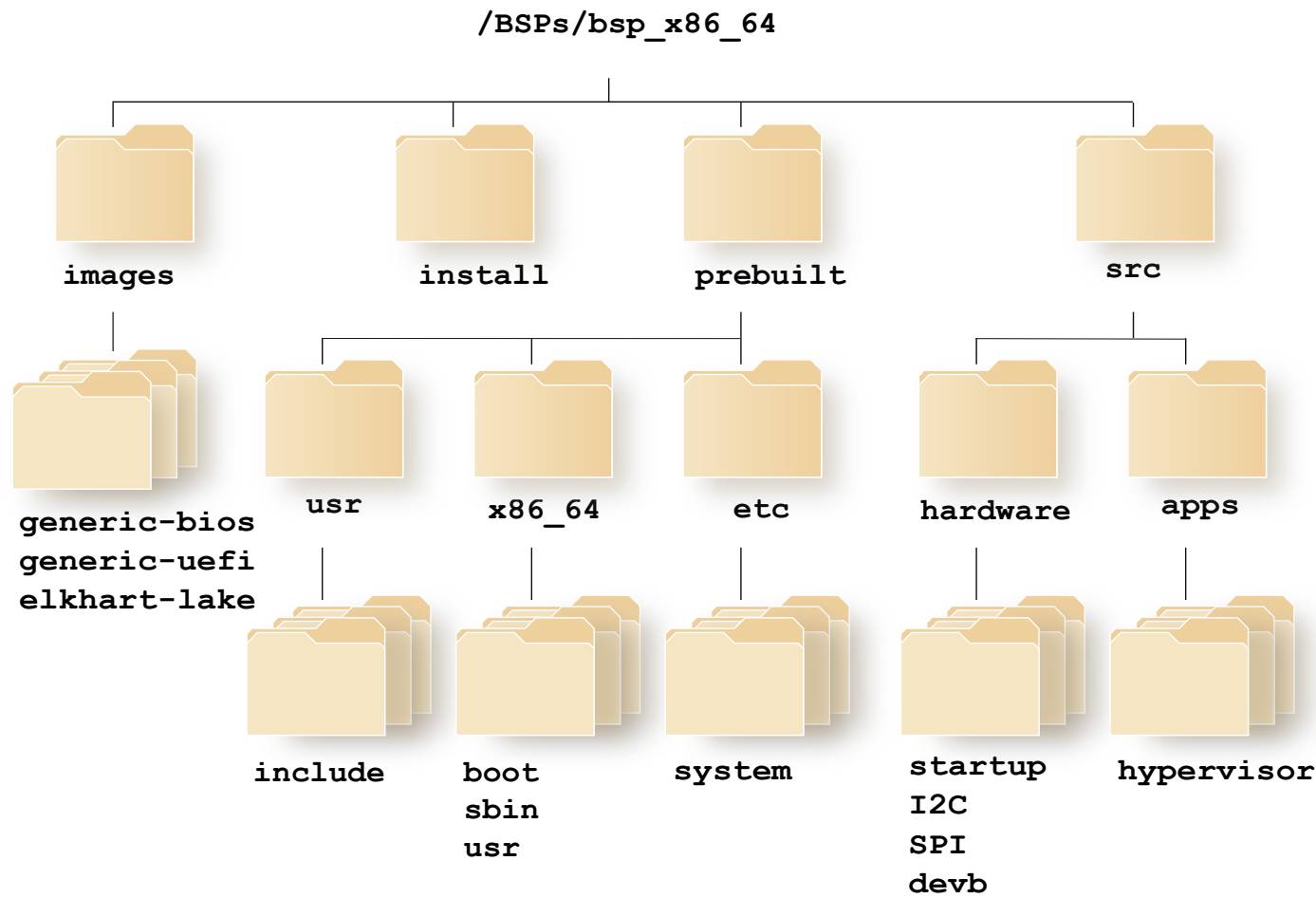


Figure 1: Structure of a BSP

For more information about the contents of the directory structure, see the *Building Embedded Systems* guide in the QNX SDP 8.0 documentation.

Working with the hardware

You must attach the appropriate connectors to your hardware.

Since this is a generic BSP guide, you must refer to the hardware guide provided by your target board provider for more information. Here are things that you should consider:

- How to connect to power and the proper power supply to use.
- The location of the USB ports, console, USB serial ports, Ethernet port, or other connectors.
- Whether any DIP switches on the board need to be set. Some boards require certain DIP switches to boot from USB or even other storage devices, such as Flash memory on the board.
- How to configure the BIOS setting to boot from the proper device. For example, if you're booting from a USB device, you may need to modify your BIOS settings so that it boots from the USB device.

Connect serial device driver

If there is no serial port available on your target, You can connect an USB serial device to the target board as the serial port. To use this device, you require a USB serial device driver on your host development machine to establish a terminal session with the target. You need to install an FTDI driver on your host system to use it.

You can download the appropriate device driver for your host system from the Future Technology Devices International Ltd. (FTDI) website at <http://www.ftdichip.com/FTDrivers.htm>.

Here are some typical settings for the target board. You should refer to the target board documentation the appropriate information:

On your host machine, using a terminal software and the appropriate COM port, these are typical settings that can be used with recent x86_64-based boards:

- Baud rate: 115200
- Data: 8 bit
- Stop: 1 bit
- Parity: no
- Flow control: none

For information about your particular board, see the platform or user guide for your board. These guides are usually available on the vendor's website.

Create and transfer an image to a bootable USB device

You can transfer the provided prebuilt IFS image to your USB device.

Alternatively, you can build the BSP to create an IFS with changes you make to the buildfile. For more information about building this BSP, see the [“Build the BSP”](#) chapter in this guide.

To create and transfer a bootable image on a USB device, use one of the following methods:

Offline method

This method allows you to use host tools to build a disk image (which is bootable). You can later flash that disk image to a USB device. This option has the advantage that you can do everything using development tools provided with your SDP installation. For information about how to create a disk image and copy it to your USB device, see [“Create and transfer a bootable disk image \(offline method\).”](#)

Interactive method

This method requires that you use a virtual machine running the QNX OS. Users who are familiar with using the self-hosted option can use this method. This method has the advantage that you can quickly copy a new QNX IFS image onto the bootable device. For information about creating a bootable USB and copying the QNX IFS image to the USB device, see [“Create a bootable USB device \(interactive method using VMware\).”](#)



NOTE: This mechanism requires that you use a virtual machine running the QNX OS. You can start a virtual machine in QNX Momentics IDE.

Create and transfer a bootable disk image (offline method)

To boot QNX OS on your x86_64 target, you can create a bootable disk image that contains the IFS file. The disk image can be copied over to a USB device.

The offline method is the simplest approach to use the QNX IFS image from your BSP because it requires that you use the host tools provided with your QNX SDP 8.0 installation. This method requires that you do these tasks:

1. Create a disk image, which builds a bootable BIOS and UEFI disk image file (e.g., **disk.img**). For more information, see [“Create a disk image.”](#)
2. Transfer the disk image to a USB device, such as a USB memory stick. For more information, depending on your host environment, see [“Transfer the USB image using host tools on Linux”](#), or [“Transfer the USB image using host tools on Windows.”](#)

Create a disk image

You can create a disk image for both BIOS and UEFI boot mode. For more information, see “Transferring an OS image onto your board” in the “Working with a BSP” chapter of the *Building Embedded Systems* guide.

Create a disk image for both BIOS and UEFI boot mode

To build a disk image for both BIOS and UEFI boot mode, in a terminal or Command Prompt on your host machine, navigate to the **\$BSP_ROOT_DIR/images/generic-bios** or **\$BSP_ROOT_DIR/images/generic-uefi** directory and run the `make disk_image` command. The `mkdiskimage` script does the following:

- calls the `mkqnx6fsimg` utility and a buildfile named **part_bios_boot.build** to create a Power-safe filesystem image named **part_bios_boot.img**;
- calls the `mkfatfsimg` utility with a buildfile called **part_uefi_boot.build** to create a DOS file system image called **part_uefi_boot.img**;
- calls the `mkqnx6fsimg` utility and a buildfile named **part_qnx_data.build** to create a Power-safe filesystem image named **part_qnx_data.img**;
- calls the `diskimage` utility using the configuration file called **disk.cfg** and the IPL binary called `ipl-diskpc1` to create a bootable disk image named **disk.img**.

The buildfile **part_bios_boot.build** configures the QNX6 filesystem for BIOS boot mode and specifies the QNX IFS image as **./boot/primary_boot_image.bin**. By default, the **x86_64-generic-bios.bin** file is used, which is provided with this BSP. If you want to use a different QNX IFS image in the disk image, modify this buildfile and specify the name of your BIN file.



NOTE: For BIOS boot mode, The name of QNX IFS image you choose must be less than 27 characters in length and the QNX IFS image you specify can't be more than 16 MB in size.

The buildfile **part_uefi_boot.build** configures the DOS filesystem for UEFI boot mode and specifies the QNX IFS image as **EFI/BOOT/BOOTX64.EFI**. By default, the **x86_64-generic-uefi.efi** file is used, which is provided with this BSP. If you want to use a different QNX IFS image in the disk image, modify this buildfile and specify the name of your EFI file.

The steps to create a disk image using host tools (offline method) are as follows:

1. Navigate to the `$BSP_ROOT_DIR/images/generic-bios` or `$BSP_ROOT_DIR/images/generic-uefi` directory. If you're in the BSP root directory (`$BSP_ROOT_DIR`):

```
$ cd images/generic-bios
or
$ cd images/generic-uefi
```

2. Create the disk image (**disk.img**) using the following commands:

```
$ make disk_image
```

After you build your bootable disk images, transfer the image to a USB device. For more information, depending on your host environment, see [“Transfer the USB image using host tools on Linux”](#), or [“Transfer the USB image using host tools on Windows.”](#)

Transfer the disk image using host tools on Linux

On a Linux system, you transfer the bootable disk image to your USB device from the commandline. Use these steps, where *path* is the path to your USB device, such as `/dev/sdb`:

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

```
$ sudo dd if=disk.img of=path
```

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

```
$ su -c dd if=disk.img of=path
```

When the copy operation is complete, your USB device should be ready to [boot QNX OS](#) on your x86_64-based target.

Transfer the USB image using host tools on Windows

On a Windows system, to transfer the disk image to your USB device, use these steps:

1. Download and install Win32 Disk Imager from <http://sourceforge.net/projects/win32diskimager/>
2. Insert your USB device, such as a USB stick into your host computer.
3. Run Win32 Disk Imager.
4. In the Win32 Disk Imager, in the Image File box, browse to the location where your image file (**disk.img**) is located and click **Open**. The selected image file appears in the box.
5. Select the letter representing your USB device.
6. Click **Write** and click **Yes** to overwrite the contents on your USB device.

When the copy operation is complete, your USB device should be ready to [boot QNX OS](#) on your x86_64-based target.

Create a bootable USB device (interactive method using VMware)

To boot QNX OS by BIOS mode on your x86_64 target, you need create and format an bootable QNX partition on the boot media and copy the QNX IFS image **x86_64-generic-bios.bin** to **/.boot** direction; To boot QNX OS by UEFI mode, you need create and format a DOS partition on the boot media and copy the **x86_64-generic-uefi.efi** to the DOS filesystem as the specific UEFI boot file **EFI/BOOT/BOOTX64.EFI**.

You can use these steps to partition a USB device, such as a USB storage key, format the partition, and set up the boot loader to make the USB device bootable. After your USB device is ready, copy the QNX IFS image to the USB device. To use this method, you should follow these steps:

1. Create a bootable USB device. Typically, this mechanism needs to be done once. For more information, see “[Create a bootable USB device using a virtual machine running QNX OS.](#)”
2. Copy the QNX IFS image to the bootable USB device. For more information, see “[Copy the QNX IFS image to the bootable USB device.](#)”



CAUTION: It's important to mention that if you have any data on this USB device (even in separate partitions), it'll be lost after completing this procedure.

Create a bootable USB device using a virtual machine running QNX OS

To complete the steps, you can start a VMware machine running QNX OS in QNX Momentics IDE (referred to as simply VMware here). You can use the VMware to create a bootable USB device, such as a USB memory stick or USB storage key. The size of the storage should be at least 4 GB.

The steps for creating a bootable USB device using the interactive method using a VMware are as follows:

1. Start the VMware in QNX Momentics IDE. For more information, see the *Creating a QNX virtual machine*.
After you boot the VMware, you're running a QNX OS environment.
2. Insert a USB device into an unused USB port on your host PC. Your VMware software should detect the USB drive, and prompt you to connect it either to your host computer or to the VMware. In the prompt, select **Connect to QNX VMware**.
3. In the VMware, start the **devb-umass** driver. This driver supports generic USB mass storage devices, such as USB memory sticks and hard drives. When you run the driver, you should see output on your VMWare that's similar to the following:

```
# devb-umass cam pnp disk name=umass
# Path=0 - QNX USB Storage
target = 0 lun = 0 Direct Access(0) - Kingston DataTraveler G3

#
```

For information about the **devb-umass**, see the QNX OS *Utilities Reference*.

4. Run the following command to determine the drive number of the USB device.

Your USB drive is likely **/dev/umass0**. If the drive already has an existing filesystem partition on it, this partition will be represented as **/dev/umass0tⁿⁿ**, where “nn” represents the type of partition (e.g. 11 or 12 is a DOS FAT32; 177, 178, or 179 is the QNX6 filesystem, etc.).



NOTE: Don't worry if there's no partition there. Any existing partitions are deleted when the new partition is created in the next step.

```
# ls /dev/umass*
/dev/umass0 /dev/umass0t12 /dev/umass0t177 /dev/umass0t179
#
```

5. Run the following commands to delete any existing partitions on the USB drive:



CAUTION: The first command below deletes all partitions on your USB drive!

```
# fdisk /dev/umass0 delete -a
# fdisk /dev/umass0 show
```

	____OS____		Start	End	____Number____	Size	Boot
	name	type	Cylinder	Cylinder	Cylinders	Blocks	
1.	-----	---	-----	-----	-----	-----	----
2.	-----	---	-----	-----	-----	-----	----
3.	-----	---	-----	-----	-----	-----	----
4.	-----	---	-----	-----	-----	-----	----

```
#
```

6. Add a type 177 partition (1GB size) into slot 1, and make partition slot 1 bootable:



NOTE: The bootable type 177 partition is used for BIOS boot mode (Legacy boot mode).

```
# fdisk /dev/umass0 add -s1 -t177 -c2,1001
# fdisk /dev/umass0 boot -s1
# fdisk /dev/umass0 show
```

	____OS____		Start	End	____Number____	Size	Boot
	name	type	Cylinder	Cylinder	Cylinders	Blocks	
1.	QNX6	177	2	1001	1000	2048000	1000 MB *
2.	-----	---	-----	-----	-----	-----	----

```

3.  -----  ---  -----  -----  -----  -----  -----
4.  -----  ---  -----  -----  -----  -----  -----

#

```

7. Enumerate the type 177 partition that was created in the previous step:

```

# mount -e /dev/umass0
#
# ls /dev/umass*
/dev/umass0  /dev/umass0t177
#

```

8. Format the type 177 partition with a QNX Power-Safe filesystem, using the **mkqnx6fs** utility, as follows:

```

# mkqnx6fs /dev/umass0t177
All files on /dev/umass0t177 will be lost!
Confirm filesystem re-format (y) or (n): y
Format fs-qnx6: 255996 blocks, 32000 inodes, 4 groups
#

```

9. Add a type 12 partition (1GB size) and Format it with a DOS filesystem, using the **mkdosfs** utility, and create the EFI boot directory as follows:



NOTE: The type 12 partition is used for UEFI boot mode.

```

# fdisk /dev/umass0 add -t12 -c1002,2001
# mount -e /dev/umass0
# ls /dev/umass0*
/dev/umass0  /dev/umass0t12  /dev/umass0t177
#
# mkdosfs /dev/umass0t12

Format complete: FAT32 (4096-byte clusters), 1021984 kB available.
#
# fdisk /dev/umass0 show

```

	OS		Start	End	Number	Size	Boot
	name	type	Cylinder	Cylinder	Cylinders	Blocks	
1.	QNX6	177	2	1001	1000	2048000	1000 MB *
2.	FAT32	12	1002	2001	1000	2048000	1000 MB
3.	-----	---	-----	-----	-----	-----	
4.	-----	---	-----	-----	-----	-----	

```
# mount -t dos /dev/umass0t12 /dos
# mkdir /dos/EFI
# mkdir /dos/EFI/BOOT
#
```

10. Optionally add a type 179 QNX partition (1GB size) on the USB drive for the persistent storage:

```
# fdisk /dev/umass0 add -t179 -c2002,3001
# mount -e /dev/umass0
# ls /dev/umass0*
/dev/umass0 /dev/umass0t12 /dev/umass0t177 /dev/umass1t179
#
# mkqnx6fs /dev/umass0t179
All files on /dev/umass1t179 will be lost!
Confirm filesystem re-format (y) or (n): y
Format fs-qnx6: 255996 blocks, 32000 inodes, 4 groups
#
# fdisk /dev/umass0 show
```

	OS		Start	End	Number		Size	Boot
	name	type	Cylinder	Cylinder	Cylinders	Blocks		
1.	QNX6	177	2	1001	1000	2048000	1000 MB	*
2.	FAT32	12	1002	2001	1000	2048000	1000 MB	
3.	QNX6	179	2002	3001	1000	2048000	1000 MB	
4.	-----	---	-----	-----	-----	-----	-----	

```
#
```

You now have a bootable USB device for both BIOS and UEFI boot mode that you can transfer QNX IFS image to it.

Copy the QNX IFS image to the bootable USB device

After you've created a bootable USB device, you must transfer the QNX IFS image that you built using this BSP to the USB device:

1. Remove and re-insert the USB device into a USB port on your Linux or Windows host that installed the QNX SDP 8.0 host environment and this BSP.



WARNING: Ensure that the USB device is connected to the host machine instead of the virtual machine running QNX OS.

2. For UEFI boot mode, copy the QNX IFS image (**x86_64-generic-uefi.efi**) from the **\$BSP_ROOT_DIR/images/generic-uefi** directory to the file **BOOTX64.EFI** in **/EFI/BOOT** directory of the bootable USB device. For example, in Linux host, the command similar to the following:

```
$ cp $BSP_ROOT_DIR/images/generic-uefi/x86_64-generic-uefi.efi /media/qnx/0A0C-1805/EFI/BOOT/BOOTX64.EFI
```



NOTE: Change the **/media/qnx/0A0C-1805/** directory in command above to match your USB device in your host.

You now have a bootable USB device with QNX IFS image for UEFI boot mode. Your USB drive is ready to [boot QNX OS](#) on your x86_64 target.

3. For BIOS boot mode, you need copy the QNX IFS image (**x86_64-generic-bios.bin**) in the **\$BSP_ROOT_DIR/images/generic-bios** directory to the DOS filesystem of the USB device, and then transfer it to the QNX filesystem (type 177) by using the VMware. copy the QNX IFS image to the USB device on your host. For example, in Linux host, the command similar to the following:

```
$ cp $BSP_ROOT_DIR/images/generic-bios/x86_64-generic-bios.bin /media/qnx/0A0C-1805/x86_64-generic-bios.bin
```



NOTE: Change the **/media/qnx/0A0C-1805/** directory in the command above to match your USB device in your host.

Remove and re-insert the USB device. When the prompt from your VMware software appears, connect the USB device to the VMware instead of the host machine. In VMware, select **VM > Removable Devices > Connect [name of your USB device]** menu to connect the USB device to the VMware, where *name of your USB device* is a string name matching your device. For example, Kingston USB device. After the USB device is connected to the VMware, you can transfer the QNX IFS image from the DOS filesystem to QNX filesystem using these steps (this example assumes that the USB device has appeared as **/dev/umass0**):

```
# mount -t dos /dev/umass0t12 /dos
# mount -t qnx6 -o sync=optional /dev/umass0t177 /qnx
# cp /dos/x86_64-generic-bios.bin /qnx/.boot/x86_64-generic-bios.bin
#
```



NOTE: Make sure that you have run the **devb-umass** command.

You now have a bootable USB device with QNX IFS image for BIOS boot mode. Your USB drive is ready to [boot QNX OS](#) on your x86_64 target.

Boot the target

After you've transferred your image to a bootable USB device, you can use it to boot the QNX OS on the system.

To boot QNX OS on your x86_64-based target:

1. Remove the USB device from your host machine, and insert it into a USB port on your x86_64 target.
2. Turn on the power for your x86_64 board, and enter the BIOS, to configure the machine to boot from the USB device you're using.
3. After you configure the BIOS to use the USB device as the primary boot device, reboot the target.

You should see your target boot the QNX IFS image. At this point, you can edit the default buildfile in the BSP to customize the boot image to meet the specific hardware and peripherals on your target, rebuild your customized image (QNX IFS image), and put the new image on your USB device.

Build the BSP

You can use the QNX Momentics IDE or the commandline on Linux or Windows to build an image.

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 8.0 documentation.

For instructions on how to build using the IDE, see [“Build the BSP \(IDE\).”](#) For generic information on how to build using the IDE, see the *IDE User's Guide* as part of the QNX SDP 8.0 documentation.

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 (*BSP_ROOT_DIR*). This allows you to conveniently switch between different BSPs by simply setting the *BSP_ROOT_DIR* environment variable.

This BSP includes prebuilt IFS images that are provided as convenience for you to quickly get QNX OS running on your board, however these prebuilt images might not have the same components as your development environment.

For this reason, we recommend that you rebuild the IFS image on your host system to ensure that you pick up the same components from your development environment.

To build your BSP, complete the following tasks:

1. Build the BSP, either using the commandline or the QNX Momentics IDE.
2. Create a bootable USB device using your host or using a VMWare machine running QNX OS.



NOTE:

Once you've unzipped the BSP, you may notice prebuilt IFS images, such as **generic-bios/x86_64-generic-bios.bin**, **generic-uefi/x86_64-generic-uefi.efi**, **elkhart-lake/x86_64-ehl-crb-uefi.efi** in the ***\$BSP_ROOT_DIR/images*** directory. You need to run `make clean` if you need remove these files in the commandline. You don't need to remove them if you use the IDE to build the BSP because the IDE does a `make clean` each time it builds.

\

Support for AMD-based systems

By default, the BSP are configured to build for Intel-based targets and therefore, the prebuilt QNX IFS images provided with this BSP works only on Intel-based targets. To build an image for AMD-based targets, you need edit the buildfile and comment out the Intel-specific drivers and uncomment the AMD-based drivers.

For example to build for AMD-based targets, make these changes in buildfile **generic-uefi/x86_64-generic-uefi.build** and rebuild your QNX IFS image:

1. Change the code from:

```
PCI_HW_MODULE=/lib/dll/pci/pci_hw-Intel_x86.so
```

to

```
PCI_HW_MODULE=/lib/dll/pci/pci_hw-AMD_x86.so
```

2. Change the code from:

```
export PCI_HW_MODULE=/lib/dll/pci/pci_hw-Intel_x86.so
```

to

```
export PCI_HW_MODULE=/lib/dll/pci/pci_hw-AMD_x86.so
```

Build the BSP (commandline)

You can use the commandline on Linux or Windows to work with this BSP.

The Makefile found in the **`$BSP_ROOT_DIR/images/generic-uefi`** directory is used for the generic x86_64 board with UEFI boot mode. It defines the following targets:

all

Builds the QNX IFS images (**`x86_64-generic-uefi.efi`**).

clean

Delete all **`*.efi`**, **`*.sym`** and **`*.img`** files.

x86_64-generic-uefi.efi

Builds a QNX IFS image (**`x86_64-generic-uefi.efi`**) for a generic x86_64 platforms that loaded using UEFI. The buildfile used to build this image is **`x86_64-generic-uefi.build`**.

x86_64-generic-uefi-hyp.efi or hyp

Builds a QNX IFS image (**`x86_64-generic-uefi-hyp.efi`**) for a generic x86_64 platform that loaded using UEFI. This IFS image supports QNX hypervisor. The buildfile used to build this image is **`x86_64-generic-uefi-hypervisor.build`**.

disk_image

Creates a bootable disk image using the `diskimage` utility. This disk image include both UEFI (type 12) and BIOS (type 177) boot partitions with QNX IFS images. It also include a QNX partition (type 179) that could be used as a persistent storage to save executable or data.

The Makefile found in the **`$BSP_ROOT_DIR/images/generic-bios`** directory is used for the generic x86_64 board with BIOS boot mode. It defines the following targets:

all

Builds the QNX IFS images (**`x86_64-generic-bios.bin`**).

clean

Delete all **`*.bin`**, **`*.sym`** and **`*.img`** files.

x86_64-generic-bios.bin

Builds a QNX IFS image (**`x86_64-generic-bios.bin`**) for a generic x86_64 platforms that loaded using BIOS (Legacy boot mode). The buildfile used to build this image is **`x86_64-generic-bios.build`**.

x86_64-generic-bios-hyp.bin or hyp

Builds a QNX IFS image (**`x86_64-generic-bios-hyp.bin`**) for a generic x86_64 platform that loaded using BIOS (Legacy boot mode). This IFS image supports QNX hypervisor. The buildfile used to build this image is **`x86_64-generic-bios-hypervisor.build`**.

disk_image

Creates a bootable disk image using the `diskimage` utility. This disk image include both BIOS (type 177) and UEFI (type 12) boot partitions with QNX IFS images. It also include a QNX partition (type 179) that could be used as a persistent storage to save executable or data.

The Makefile found in the `$BSP_ROOT_DIR/images/elkhart-lake` directory is used for the Intel Elkhartlake CRB board with UEFI boot mode. It defines the following targets:

all

Builds the QNX IFS images (`x86_64-eh1-crb-uefi.efi`).

clean

Delete all `*.efi`, `*.sym` and `*.img` files.

x86_64-eh1-crb-uefi.efi

Builds a QNX IFS image (`x86_64-eh1-crb-uefi.efi`) for the Intel Elkhartlake CRB board that loaded using UEFI. The buildfile used to build this image is `x86_64-generic-uefi.build`.

x86_64-eh1-crb-uefi-hyp.efi or hyp

Builds a QNX IFS image (`x86_64-eh1-crb-uefi-hyp.efi`) for the Intel Elkhartlake CRB board that loaded using UEFI. This IFS image supports QNX hypervisor. The buildfile used to build this image is `x86_64-eh1-crb-uefi-hypervisor.build`.

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

Build from the commandline

To build the BSP on your host system, in a Command Prompt window (Windows) or a terminal (Linux), perform the following steps:

1. Download the BSP archive, unzip it, and set up your environment to build. For more information, see [“Download and set up the BSP”](#) in the [“Installation Notes](#) chapter.”
2. Navigate to the `$BSP_ROOT_DIR` directory. If you want to remove all prebuilt IFS images and re-build them later, run `make clean`:

```
$ cd $BSP_ROOT_DIR
$ make clean
```



NOTE: If you run the `make clean` command, it removes all prebuilt IFS images.

3. Run `make` command in the `BSP_ROOT_DIR`. When you run `make`, it does the following:
 - copy the **prebuilt** directory to **install** directory;
 - copy header files to **install** directory;

- build the source files in this BSP;
- build the IFS images if they are removed in previous step.



NOTE: No buildfile is generated. The buildfiles are preloaded in the `$BSP_ROOT_DIR/images/generic-bios`, `$BSP_ROOT_DIR/images/generic-uefi`, and `$BSP_ROOT_DIR/images/elkhart-lake` directories.

```
$ make
```

4. If you changed any source file in this BSP, you need navigate to the directory that include the source file you've changed, and run `make install` command to rebuild the new code and copy it to the **install** directory, for example, you've updated the x86 startup code:

```
$ cd src/hardware/startup/boards/x86  
$ make install
```



NOTE: If you run the `make install` command, it would build and copy the new binary to the **install** directory.

You should now have IFS files called **images/generic-bios/x86_64-generic-bios.bin**, **images/generic-uefi/x86_64-generic-uefi.efi**, **images/elkhart-lake/x86_64-ehl-crb-uefi.efi**.



NOTE: We recommend that you use the `make` command to build your IFS image. If you use the `mkifs` utility directly, you must ensure that you use the `-r` option to specify the location of the binaries.

Build the BSP (IDE)

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

Build in the QNX Momentics IDE

You can build the entire BSP in the QNX Momentics IDE, which includes building the source and associated images.

1. If you haven't done so, launch the IDE, and then import the BSP archive (which was downloaded for you using the QNX Software Center) into the IDE. For more information see ["Import to the QNX Momentics IDE"](#) in the ["Installation Notes"](#) chapter.

After you've imported your BSP source, you're ready to build.

2. If you haven't already, switch to the C/C++ Perspective, then in the Project Explorer view, right-click the BSP source project (such as **hw.x86_64**) and select **Build Project**. This step builds the entire BSP, which includes the images and the binaries required for the images. You can check the progress and outcome of your build in the Console view.
3. If you want to modify the buildfile, open the **.build** file under the **System Builder Files** folder. After you've made your changes, right-click the BSP source project and select **Build Project**.

You can make changes to the buildfile in the **images** folder, see ["Building the images folder in the IDE"](#) section in this chapter.

Building the images folder in the IDE

You can make changes to the buildfiles in the **images** folder.



CAUTION: Changes you make to the buildfile in the **images** folder are overwritten if you build the entire BSP Project as described in ["Build the IDE."](#) Ensure that you save a backup copy of the buildfile elsewhere.

To build the changes you've made to a buildfile, create a *make target* in the **images** folder and then map it to an existing make target in the **Makefile** located in the **images** folder, which is equivalent to running the make command from the **images** directory on the commandline.

For more information about creating make targets, see **Tasks > Building Projects > Creating a make target** in the *C/C++ development User Guide* in the IDE Help.

1. In the Project Explorer view, expand the **images** folder, right-click the IFS file you want recreated, and select **Delete**. The IFS must be deleted or it won't be rebuilt.
2. In the example view, right click the **images** folder and select **Build Targets > Create...**
3. In the Create Build Target dialog box, type the name of an existing target in the **Makefile**, and then click **OK**.

For example, if you wanted to build the default IFS file, **x86_64-generic-uefi.efi**, it maps to the **x86_64-generic-uefi.efi** target in the Makefile, you would type **x86_64-generic-uefi.efi** as the target. For more information on the targets available, see ["Build the BSP \(commandline\)."](#)



NOTE: It's a good idea to create `clean` and `all` build targets, which run the `make clean` (to remove all created files in the **images** folder) and `make all` (to rebuild all the IFS files) commands, respectively.

4. In the Project Explorer view, under the **images** folder, you should see **Build Targets** created. Right-click **Build Targets**, select the build target that you created in the previous step, and click **Build**.

The IDE starts to build the image. You can look for any issues that may occur in the **Console** and **Problems** views. After your image is built, you should see your rebuilt **.bin** file beneath the **images** folder as shown in the following illustration:

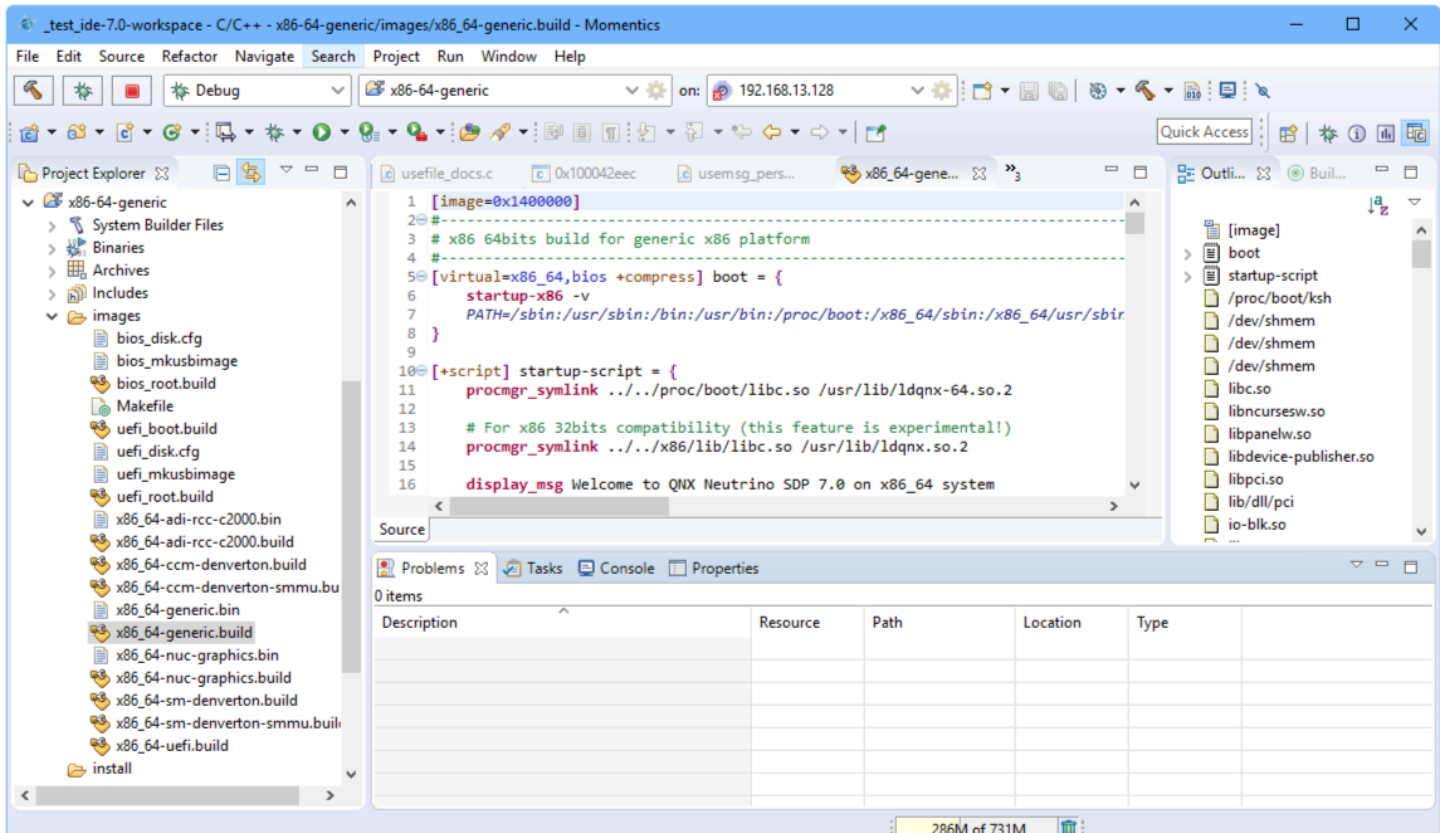


Figure 2: Build image in the IDE

You should now have an IFS file, such as **x86_64-generic.bin**. You can copy the file to a USB device that has been made bootable or you can build a USB image and flash a USB device with that image; see [“Create and transfer a bootable disk image \(offline method\).”](#) If your target can boot from a USB device, see [“Create a bootable USB device \(interactive method using VMware\)”](#) to create a USB image. Both of these sections are in the [“Installation Notes”](#) chapter.

Generic x86_64_Driver Commands

The tables below provide a summary of driver commands.

Some of the drivers are commented out in the buildfile provided in the **images** directory of this BSP. To use these drivers on the target hardware, you may need to uncomment them in the buildfile, rebuild the image, and load the image onto the board.

The order that the drivers are started in the provided buildfile is one way to start them. The order that you start the drivers is completely customizable and is often specific to the requirements of your system. It's important to recognize that the order you choose impacts the boot time.



NOTE: Some drivers depend on other drivers, so it's sometimes necessary to start them in a specific order because of these dependencies. For example, in the provided buildfile, the USB host driver is started before the mass storage drivers because they rely on it.

- For more information on best practices for building an embedded system and optimizing boot times for your system, see the *Building Embedded Systems* and the *Boot Optimization* guides in the QNX Software Development Platform 8.0 documentation.
- For more information about the drivers and commands listed here, see the QNX OS *Utilities Reference*. This chapter provides information about BSP-specific options for any of the commands and drivers that aren't described in the *Utilities Reference*.

Here's the list of drivers available for this board:

- [Startup](#)
- [PCI Server](#)
- [Serial](#)
- [Network](#)
- [USB \(Host Controller\)](#)
- [NVMe](#)
- [SATA](#)
- [EIDE](#)
- [SDMMC](#)
- [UMASS](#)

Startup

Device	STARTUP
Command	startup-x86

Required binaries	startup-x86
Source location	\$BSP_ROOT_DIR/src/hardware/startup/boards/x86

For more information about the **startup-x86** driver, see the “**startup-x86**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

PCI Server

Device	Intel Host/PCI bridge
Environment variables	<i>PCI_HW_MODULE</i> , <i>PCI_SLOG_MODULE</i> , <i>PCI_DEBUG_MODULE</i> , <i>PCI_BASE_VERBOSITY</i> , <i>PCI_HW_CONFIG_FILE</i> ,
Command	<code>pci-server -aspace-enable</code>
Required libraries	ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, pci/pci_*.so, libsecpol.so, libslog2.so
Source location	Prebuilt only

For more information about the PCI Server, see the *PCI Server User's Guide*; For more information about the **pci-server**, see the “**pci-server**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.



NOTE: Use **pc_hw_Intel_x86.so** for Intel-based platforms (by default in buildfiles);
Use **pci_hw_AMD_x86.so** for AMD-based platforms.

Serial

Device	8250-compatible UART
Command	<code>devc-ser8250 -e -b115200</code>
Required binaries	devc-ser8250
Required libraries	ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, libsecpol.so
Source location	Prebuilt only

For more information about the **devc-ser8250** driver, see the “**devc-ser8250**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

Device	USB serial UART
--------	-----------------

Command	<code>devc-serusb -e -b115200</code>
Required binaries	devc-serusb
Required libraries	ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, libsecpol.so, libusbdi.so
Source location	Prebuilt only

For more information about the **devc-serusb** driver, see the “**devc-serusb**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.



NOTE: You must have the [USB \(Host Controller\)](#) driver started before you start the USB serial driver.

Network

Device	Network devices
Command	<code>io-sock -m phy -d vmx -m pci -d em -d ix -d re -m usb -d axe -d axge -d cdce -d smsc</code>
Required binaries	io-sock, dhcpcd, ifconfig, sshd, qconn
Required libraries	devs-vmx.so, devs-em.so, devs-ix.so, devs-re.so, devs-axe.so, devs-axge.so, devs-cdce.so, devs-smisc.so, mods-phy.so, mods-pci.so, mods-usb.so, libfdt.so, libsocket.so, libqcrypto.so, libz.so, ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, libsecpol.so, libslog2.so
Source location	Prebuilt only

For more information about the **io-sock**, see the *High-Performance Networking Stack (io-sock) User's Guide*. For more information about the **devs-*** and **mod-*** drivers, see the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

USB (Host Controller)

Device	USB (Host driver)
Command	<code>io-usb-otg -d xhci -d ehci -d ohci -d uhci</code>
Required binaries	io-usb-otg, usb
Required libraries	devu-hcd-xhci.so, devu-hcd-ehci.so, devu-hcd-uhci.so, devu-hcd-ohci.so, ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, pci/p*_*.so, libsecpol.so, libslog2.so, libm.so
Source location	Prebuilt only

For more information about the **io-usb-otg** driver and **usb** utility, see the **io-usb-otg** and **usb** section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

NVMe

Device	Non-Volatile Memory Express (NVMe)
Command	<code>devb-nvme disk name=nvme</code>
Required binaries	devb-nvme
Required libraries	libcam.so, cam-disk.so, io-blk.so, fs-dos.so, fs-qnx6.so, ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, pci/pci_*.so, libsecpol.so, libslog2.so, libfsnotify.so
Source location	Prebuilt only

For more information about the **devb-nvme** driver, see the “**devb-nvme**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

SATA

Device	SATA
Command	<code>devb-ahci disk name=sata</code>
Required binaries	devb-ahci
Required libraries	libcam.so, cam-disk.so, io-blk.so, fs-dos.so, fs-qnx6.so, ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, pci/pci_*.so, libsecpol.so, libslog2.so, libfsnotify.so
Source location	Prebuilt only

For more information about the **devb-ahci** driver, see the “**devb-ahci**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

EIDE

Device	Fast IDE (EIDE)
Command	<code>devb-eide disk name=hd</code>
Required binaries	devb-eide
Required libraries	libcam.so, cam-disk.so, io-blk.so, fs-dos.so, fs-qnx6.so, ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, pci/pci_*.so, libsecpol.so, libslog2.so, libfsnotify.so

Source location	Prebuilt only
-----------------	---------------

For more information about the **devb-eide** driver, see the “**devb-eide**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

SDMMC

Device	SDMMC
Command	devb-sdmmc cam pnp sdio idx=0,vid=0x8086,did=0x5acc disk name=sdmmc
Required binaries	devb-sdmmc
Required libraries	libcam.so, cam-disk.so, io-blk.so, fs-dos.so, fs-qnx6.so, ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, pci/pci_*.so, libsecpol.so, libslog2.so, libfsnotify.so
Source location	Prebuilt only

For more information about the **devb-sdmmc** driver, see the “**devb-sdmmc**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

UMASS

Device	USB Mass Storage interface
Command	devb-umass cam pnp dos exe=all disk name=umass
Required binaries	devb-umass
Required libraries	libcam.so, cam-disk.so, io-blk.so, fs-dos.so, fs-qnx6.so, ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, libusbdi.so, libsecpol.so, libslog2.so, libfsnotify.so
Source location	Prebuilt only

For more information about the **devb-umass** driver, see the “**devb-umass**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.



NOTE: You must have the [USB \(Host Controller\)](#) driver started before you start the USB serial driver.

Working with Intel Elkhart Lake CRB platform

The Intel Elkhart Lake CRB platform uses many of the same source and binaries provided for the generic BSP, however, there are a few differences. The boards we provide buildfiles for are Intel Elkhart Lake CRB boards.

- For information about connecting the power, the UART port and other peripheral, see “[Using Intel Elkhart Lake CRB platform](#)”, “[Connecting power supply to Intel Elkhart Lake CRB platform](#)” and “[Connecting UART port to Intel Elkhart Lake CRB platform](#)”.
- There are two different Intel BIOS variants for using TSN or SATA interfaces. For more information about BIOS setup, see “[BIOS setup](#)”.
- There's a separate buildfile, and separate image is built for Intel Elkhart Lake CRB platform. For more information, see “[Modifying the IFS for Intel Elkhart Lake CRB platform](#)”.
- You can boot the board with a QNX IFS image from an USB device. For more information, see “[Boot Intel Elkhart Lake CRB board](#)”.

Using Intel Elkhart Lake CRB platform

For information about the hardware setup, jumper, and peripheral connections, etc., refer to **Intel Elkhart Lake CRB User Guide** (Content ID: 615859). For Intel BIOS binaries, refer to **Elkhart Lake IAFW (BIOS), FSP, and IFWI binaries for Maintenance Release 1** (Content ID: 646827). They can be found at the following location: rdc.intel.com.

Connecting power supply to Intel Elkhart Lake CRB platform

Connect the power supply to J9A1 power jack.

Connecting UART port to Intel Elkhart Lake CRB platform

Connect DB9 female connector of the serial cable to J4J1 HSUART0 (LOWER) connector on Intel Elkhart Lake CRB board, and connect another side of the serial cable to your host machine.

Connect micro USB connector of the serial cable to J1H1 DEBUG_uUSB connector on Intel Elkhart Lake CRB board, and connect another side of the serial cable to your host machine.

On your host machine, using a terminal software and the appropriate COM port, these are typical settings that can be used with Intel Elkhart Lake CRB board:

- Baud rate: 115200
- Data: 8 bit
- Stop: 1 bit
- Parity: no
- Flow control: none

BIOS setup

In Intel package (Content ID: 646827), **Elkhart Lake IAFW (BIOS), FSP, and IFWI binaries for Maintenance Release 1**, there are two different Intel BIOS variants (TSN and SATA) for Elkhartlake CRB platform as below. By default, this BSP use the TSN variant for using TSN interface.

- **IFWI_SPI_Boot/EHL_NR01_B0B0_XXXXMCC_CPSF_SEP5_D40B00_2021_WW31.2.1_SPI_ES0_TSN_EXT.bin**
- **IFWI_SPI_Boot/EHL_NR01_B0B0_XXXXMCC_CPSF_SEP5_D40B00_2021_WW31.2.1_SPI_ES0_SATA_EXT.bin**

To use the SATA interface, you need flash the SATA variant of BIOS on your target board. Then enable SATA interface, and disable TSN GBE interface. Use the following steps to setup BIOS:

- Power on the target
- Press **F2** key to entry BIOS
- Enter **Intel Advanced Menu**
- Enter **PCH-IO Configuration**
- Enter **TSN GBE Configuration** in **PCH-IO Configuration**
- Chose **PCH TSN LAN Controller** and set as **Disabled**
- Enter **SATA Configuration** in **PCH-IO Configuration**
- Chose **SATA Controller** and set as **Enabled**
- Press **F4** key to save the changes

To use the SPI interface, you need enable the SPI1 interface. Use the following steps to setup BIOS:

- Power on the target
- Press **F2** key to entry BIOS
- Enter **Intel Advanced Menu**
- Enter **PCH-IO Configuration**
- Enter **SerialIO Configuration** in **PCH-IO Configuration**
- Chose **SPI1 Controller** and set as **Enabled**
- Press **F4** key to save the changes

To enable the I2C and SPI devices PCI memory address, you need disable the LPSS Device D3 feature in BIOS. Use the following steps to disable the LPSS Device D3 feature:

- Power on the target
- Press **F2** key to entry BIOS
- Enter **Intel Advanced Menu**
- Enter **PCH-IO Configuration**
- Enter **SerialIO Configuration**
- Chose **LPSS Device D3 state** and set as **Disabled**

- Press **F4** key to save the changes

Modifying the QNX IFS image for Intel Elkhart Lake CRB platform

By default, the **x86_64-ehl-crb-uefi.build** buildfile generate an IFS image that can be used to boot Intel Elkhart Lake CRB board. The **x86_64-ehl-crb-uefi.efi** IFS image is found in the **\$BSP_ROOT_DIR/images/elkhart-lake** directory. To see what's included in the default IFS images, you can view the buildfiles.

Boot Intel Elkhart Lake CRB board

Booting an Intel Elkhart Lake CRB board uses these steps:

- Copy the QNX IFS image **x86_64-ehl-crb-uefi.build** to a DOS-formatted USB stick as the file name **/EFI/BOOT/BOOTX64.EFI**.
- Insert the USB stick to any USB port on the target board.
- Turn on the power and configure the BIOS to boot from UEFI USB device.
- You should see the prompt on your serial terminal after QNX system up.

Intel Elkhart Lake CRB driver commands

The tables below provide a summary of driver commands for Intel Elkhart Lake CRB platform.

Any other drivers used by the buildfile for Intel Elkhart Lake CRB platform are available as part of the generic driver commands. For more information about the generic commands, see the “[Generic x86_64_Driver Commands](#)” chapter in this guide. For more information about these and other commands, see the *QNX OS Utilities Reference*.

Here are the Elkhartlake-specific drivers:

- [Startup](#)
- [Serial](#)
- [SPI](#)
- [I2C](#)
- [SDMMC](#)



NOTE: The commands summarized below represent the commands that are used on Intel Elkhart Lake CRB platform. We've indicated which commands are board-specific. For SATA, NVMe, USB and network driver and other drivers, see [Generic x86_64_Driver Commands](#). or more information, see the buildfiles.

Startup

Device	STARTUP
Command	startup-x86
Required binaries	startup-x86
Source location	<code>\$BSP_ROOT_DIR/src/hardware/startup/boards/x86</code>

For more information about the **startup-x86** driver, see the “**startup-x86**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.

Serial

Device	SERIAL
Commands	<code>devc-serpci vid=0x8086,did=0x4b96,pci=0 -e -b115200</code> <code>devc-serpci vid=0x8086,did=0x4b4d,pci=0 -e -b115200</code>
Required binaries	<code>devc-serpci</code>
Required libraries	<code>\$QNX_TARGET/x86_64/lib/dll/pci</code>
Source location	Prebuilt only

For more information about the **devc-serpci** driver, see the “**devc-serpci**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.



NOTE: To use the serial driver, setup BIOS as below:

1. Disable the the LPSS Device D3 feature

SPI

Device	SPI
Commands	<code>spi-pxa</code> <code>spi-pxa -c /etc/system/config/spi/spi.conf</code>
Required binaries	spi-pxa
Configuration file	/etc/system/config/spi/spi.conf
Source location	src/hardware/spi/pxa



NOTE: To use the SPI driver, setup BIOS as below:

1. Disable the the LPSS Device D3 feature
2. Enable the SPI1 interface

I2C

Device	I2C
Commands	<code>i2c-baytrail -d 0x4b7b -r 0x8086 --u3</code> <code>i2c-baytrail -d 0x4b4b -r 0x8086 --u4</code>
Required binaries	i2c-baytrail, isendrecv, isend
Source location	src/hardware/i2c/baytrail



NOTE: To use the I2C driver, setup BIOS as below:

1. Disable the the LPSS Device D3 feature

SDMMC

Device	SDMMC
--------	-------

Command	EMMC device: devb-sdmmc cam quiet,pnp sdio idx=0,vid=0x8086,did=0x4b47,irq=16 disk name=emmc SDC device: devb-sdmmc cam quiet,pnp sdio idx=0,vid=0x8086,did=0x4b48,irq=17 disk name=sd
Required binaries	devb-sdmmc
Required libraries	libcam.so, cam-disk.so, io-blk.so, fs-dos.so, fs-qnx6.so, ldqnx-64.so.2, libc.so, libgcc_s.so, libpci.so, pci/pci_*.so, libsecpol.so, libslog2.so, libfsnotify.so
Source location	src/hardware/devb/sdmmc

For more information about the **devb-sdmmc** driver, see the “**devb-sdmmc**” section in the *Utilities Reference* guide of the QNX SDP 8.0 documentation.