BSP User's Guide

Raspberry Pi 4 Board

BlackBerry | QNX

Copyright © 2018–2020, BlackBerry Limited. All rights reserved.

QNX Software Systems
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/

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: http://www.blackberry.com/patents

Electronic edition published: December 09, 2020

Contents

About This Guide **5**
Typographical conventions 6
Technical support 8

Chapter 1: Before You Begin **9**

Chapter 2: About This BSP **11**

Chapter 3: Installation Notes **13**
Download and set up the BSP 14
Set up the hardware 17
System Layout 18
Prepare a bootable microSD 19
Transfer the image 24
Copy the necessary binaries and IFS to target 24
Boot the board 26
Bootup using UART3 as debug port 28

Chapter 4: Build the BSP **29**
Build the BSP (commandline) 30
Build the BSP (IDE) 31

Chapter 5: Driver Commands **33**

Chapter 6: BSP-specific Drivers and Utilities **41**
`deva-ctrl-bcm2711_pwm.so` 42
`deva-ctrl-bcm2711_pcm.so` 44
`deva-mixer-pcm512x.so` 48
`devc-serminiuart` 50
`devc-serpl011` 52
`devnp-genet.so` 54
`devu-hcd-bcm2711-xhci.so` 56
`i2c-bcm2711` 58
`spi-master` 60

About This Guide

This guide contains installation and start-up instructions for the QNX Board Support Package (BSP) for the Raspberry Pi 4 Board (Model B). This BSP supports Raspberry Pi 4 Board (Model B) 1G, 2G, and 4G. Only 64-bit support is provided by QNX Software Systems.

For convenience, this document refers to the Raspberry Pi 4 Board (Model B) as the Raspberry Pi 4 board. This guide covers the following information:

- an overview of the BSP for the Raspberry Pi 4 board
- structure and contents of the BSP
- how to build the BSP using commandline or the QNX Momentics IDE
- how to prepare a bootable microSD
- how to transfer the image to a bootable microSD
- how to boot your board

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, and the supported host OSs and boards	*About This BSP*
Booting the reference board with the provided reference image	*Installation Notes*
Building this BSP	*Build the BSP*
Running the driver commands	*Driver Commands*

Copyright © 2020, BlackBerry Limited

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.

Copyright © 2020, BlackBerry Limited

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

DANGER: 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.

Copyright © 2020, BlackBerry Limited

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.

Copyright © 2020, BlackBerry Limited

Chapter 1
Before You Begin

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

Essential information

Before you install your BSP, review the following documentation:

- Information about your board's hardware and firmware provided by the vendor of the board. Documentation for the Raspberry Pi 4 Board (Model B) can be found on the Raspberry PI website at *https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications*.
- *Building Embedded Systems*. This guide contains general information about working with BSPs from QNX Software Systems that's common to all BSPs and is available from the QNX Software Development Platform 7.0 documentation.

Required software

To work with this BSP, you require the following:

- BSP ZIP file for the board.
- QNX Software Development Platform 7.0 installed on a Linux, macOS, or Windows host system.
- Virtual COM Port (VCP) Driver. This driver must be installed on your host system that connects to the board. 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 (Qtalk, Hyperterminal, PuTTy, etc.) to connect to the board for debugging. Alternatively, you can use a `telnet` or `ssh` session from the QNX Momentics IDE.

Copyright © 2020, BlackBerry Limited

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: For the status of a driver, see the release notes for this BSP.

Component	Format	Comments
I2C driver	Binary and source	
Serial driver	Binary and source	
Audio driver	Binary and source	
Network driver	Binary and source	
PCI driver	Binary only	
SD/MMC driver	Binary and source	
SPI driver	Binary and source	
Startup	Binary and source	
USB Host Controller driver	Binary only	
Watchdog	Binary and source	
Wireless driver (WiFi)	Binary only	

Supported OSs

To install and use this BSP, you must have installed the QNX SDP 7.0 on a Linux, macOS, or Windows host. This BSP supports the following target OS:

- QNX Neutrino RTOS 7.0

Supported boards

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**.

Copyright © 2020, BlackBerry Limited

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.

Copyright © 2020, BlackBerry Limited

Chapter 3
Installation Notes

These installation notes describe how to build, work with, and run this BSP.

Overview

Here's the process to build and run your BSP:

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 to set DIP switches on your board to further configure it. For more information about how to connect to your board, see "*Set up the hardware*."
3. Create a bootable microSD . After you create this card, you can either transfer prebuilt images that come with the BSP or change the buildfile to build your own, customized image. For more information about how to create a bootable microSD , see "*Prepare a bootable microSD*."
4. If this is the first time you're running the BSP, you must transfer the provided IFS image file to the microSD to boot your board.
5. After you've put your image on the microSD, you can boot the board. At this point, your board is running QNX Neutrino.

After your board boots, you can use that image to understand how it works. Depending on the other devices that 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 buildfile and build a custom image. For specific information about building the image for this board, see the"*Build the BSP*" chapter in this guide. For generic information about modifying and building an image, see the *Building Embedded Systems* guide in the QNX SDP 7.0 documentation.

Copyright © 2020, BlackBerry Limited

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 **bsp** directory located in your QNX SDP 7.0 installation. 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 7.0 installation.

  The `unzip` utility is available on all supported host platforms and is included with your installation of QNX SDP 7.0.

  The QNX-packaged BSP is independent of the installation and build paths, if the QNX Software Development Platform 7.0 is installed. To determine the base directory, open a Command Prompt window (Windows) or a terminal (Linux and macOS), and then type `qconfig`.

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 and macOS), run **qnxsdp-env.bat** (Windows) or `source` **qnxsdp-env.sh** (Linux, macOS) from your QNX SDP 7.0 installation to set up your environment. You must run the shell or batch file to use the tools provided with QNX SDP 7.0.

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

2. Navigate to the directory where you want to extract the BSP (e.g., `cd` **/BSPs/raspberrypi-bcm2711-rpi4**). 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*

   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 and macOS) or `set BSP_ROOT_DIR=`*bsp_working_dir* (Windows) command to set the environment variable.

3. In the directory you created in the previous step, use the `unzip` command to extract the BSP. For example:

   `unzip -d` *bsp_working_dir* **BSP_raspberrypi-bcm2711-rpi4_br-700_be-700_***build_ID***.zip**

   where *build_ID* is the BSP build number (e.g., `JBN9`).

You should now be ready to build your BSP. See the "*Build the BSP (commandline)*" chapter for more information.

Copyright © 2020, BlackBerry Limited

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, select **File** → **Import**.
3. Expand the **QNX** folder.
4. Select **QNX Source Package and BSP (archive)** from the list and then click **Next**.
5. In the **Select the archive file** dialog, click **Browse...**.
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., **bsp-raspberrypi-bcm2711-rpi4**). 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 7.0 documentation.

Copyright © 2020, BlackBerry Limited

Set up the hardware

Connect the hardware

In addition to the board, you require these cables:

- RS-232 serial device to a USB port cable
- an Ethernet cable
- 5V DC power cable

CAUTION:

Use only the power supplies 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 the Raspberry Pi 4 board:

1. Insert the microSD with your QNX IFS image file into the selected boot slot on the board.
2. Connect a RS-232 serial to USB cable from the board's serial pins and the USB port on your host machine.

Figure 2: Serial pins on the GPIO expander header (J8)

Copyright © 2020, BlackBerry Limited

Figure 3: Identifying TX and RX pins for serial communication on the GPIO Expander

3. Connect the Ethernet cable to the Ethernet jack on the board.
4. Connect the board's 5V DC power supply to the **POWER IN** (USB-C) connector on the board.
5. Identify the host serial port on your host system:

   On a Linux or macOS host, you can check which port is the host serial port by looking at what port appears when the cable is inserted. To do this, type the command:

   ```
   $ ls /dev/ttyUSB*
   ```

   On a Windows system, open the **Device Manager** and expand the “Ports (COM and LPT)” section. Then, look for a COM port named Gadget Serial or USB Serial Port.

6. Connect to the COM port using your favorite terminal program with these settings:
   - Baud rate: **115200**
   - Data: **8 bit**
   - Parity: **none**
   - Stop: **1 bit**
   - Flow control: **none**

System Layout

The following is the memory layout required for the Raspberry Pi 4 board.

The table below shows the memory layout for the BSP image:

Item	Start	End
OS image location	`0x80000`	

Copyright © 2020, BlackBerry Limited

Prepare a bootable microSD

You can prepare a bootable microSD on a Linux, macOS, or Windows host system. We recommend that you use Class 10 (or UHS-1) microSDs.

Depending on your host OS, follow the steps in either the "Prepare a bootable microSD on a Linux or macOS host" or "Partition and format an microSD on a Windows host" section to prepare your microSD. After you've prepared the card, you can copy the image to it so that you can boot the board.

CAUTION: Ensure that your microSD isn't write-protected.

Typically there's a lock switch on the side of an microSD or the SD case (for microSD cards). When this switch is in the lock position, you can't modify or delete the contents of the card. Make sure that this switch is in the unlock position so that you can format and partition the card.

Prepare a bootable microSD on a Linux or macOS host

The following is a quick, step-by-step method for formatting the microSD from a terminal:

1. Run the following command, once with the microSD out of the reader on your host system:

   ```
   $ mount
   . . .
   ```

   Note the mounted devices listed. Now insert the card and run the same command a second time.

   ```
   $ mount
   . . .
   /dev/sda1
   ```

   When the command is run with the card inserted, an additional device should appear (for example, **sda1** or **mmcblk0p1**; what you see may differ on your computer). This additional device is the target device.

   For the remainder of these instructions, we'll use **sda1** for the device used. Substitute your own device that you noted when you ran the `mount` command for the second time.

2. If the device is mounted, unmount it. For example:

   ```
   $ umount /dev/sda1
   ```

   After you run the command, your microSD is now ready to be partitioned.

3. Using administrator privileges, run the following commands, substituting your device for **sda1**, and responding to the prompts as indicated. For example:

   ```
   $ sudo fdisk /dev/sda
   ```

   Since you are formatting the whole microSD and not just a partition, you may need to strip the identifier at the end of the mountpoint when you run the `fdisk` command. For example, for **sda1**, the identifier is "1" or for **mmcblk0p1**, the identifer is "p1." For more information about the identifier, see the user guide for the variant of Linux or macOS that you're running.

Copyright © 2020, BlackBerry Limited

4. Remove the existing partitions. Keep typing the `d` command until no partitions are left. For example:

```
Command (m for help): d
No partition is defined yet!
```

5. Type `u` and then `o` to create an empty DOS partition table:

```
...

Command (m for help): u
Changing display/entry units to cylinders

Command (m for help): o

Building a new DOS disklabel with disk identifier 0xcdd1b702.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by write.

WARNING: cylinders as display units are deprecated. Use command 'u' to change units to sectors.
```

6. Type `n` followed by `p` to create a new primary partition. For example:

```
Command (m for help): n

Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): p
```

7. Type `1` to set the partition as the primary partition and specify the defaults for your card. For example, enter the specified defaults for *start cylinder* and *end cylinder*, which in this example are 1 and 240. Your defaults may be different depending on the size of your microSD.

```
Partition number (1-4, default 1): 1

First cylinder (1-240, default 1): start cylinder

Last cylinder, (100-240, default 240): end cylinder

Using default value 240
```

8. Type `a` to set the active partition. Generally, this should be partition `1`.

```
Command (m for help): a Partition number (1-4): 1
```

9. Type `t` and then `c` to set the partition type (1 is default).

```
Command (m for help): t
Selected partition 1

Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

10. Type `w` to write the changes.

```
Command (m for help): w

The partition table has been altered!
```


Copyright © 2020, BlackBerry Limited

```
Calling ioctl() to re-read partition table.
Syncing disks.
```

11. Format the microSD . Note that the identifier, such as “1” or “p1” at the end of the mountpoint must be included:

```
$ sudo mkfs.vfat /dev/sda1
mkfs.msdos 3.0.12 (16 October 2018)
```

Partition and format an microSD on a Windows host

The default Windows formatting tool that appears when you insert a blank (or unrecognized) microSD into a Windows host isn't sufficient to format the microSD with a bootable partition. In addition, some microSD s come with pre-created partitions that aren't suitable to boot the board. Instead, use the diskpart command-line tool to format your microSD .

1. Start a Command Prompt window (e.g., on Windows 10, run **Start Menu** → **All Programs** → **Accessories** → **Command Prompt**). Ensure that you start the Command Prompt window using Administrator privileges.
2. Run the `diskpart` utility from your Command Prompt window:

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\windows\system32>diskpart

Microsoft DiskPart version 6.3.9600

Copyright (C) 1999-2013 Microsoft Corporation.
On computer: CI0700000001064

DISKPART>
```

3. Run the `list disk` command to get a list of available drives, then identify your microSD on this list (in this example, Disk 1):

```
DISKPART> list disk

Disk ###     Status       Size       Free      Gyn      Gpt
--------     ------       ----       ----      ---      ---
Disk 0       Online       465 GB     1024 KB
Disk 1       Online      7580 MB      4401 MB

DISKPART>
```

When the command is run with the card inserted, an additional disk should appear (in this case Disk 1). This additona is the target device. For the steps that follow, we'll be using Disk 1. Ensure you use the disk that matches your microSD .

4. Run the `select disk` command to perform operations on the microSD :

```
DISKPART > select disk 1

Disk 1 is now the selected disk.

DISKPART >
```

5. Show the partitions on your disk on the microSD. In this case, there are two partitions:

```
DISKPART> list part
Partition ### Type Size Offset
------------- ---------------- ------- -------
Partition 1 Primary 512 MB 1024 KB
Partition 0 Primary 2048 MB 1125 MB

DISKPART>_
```

6. Delete all available partitions on the microSD:

```
DISKPART> select part 1

Partition 1 is now the selected partition.

DISKPART> delete part

DiskPart successfully deleted the selected partition.

DISKPART> select part 1

Partition 1 is now the selected partition.

DISKPART> delete part

DiskPart successfully deleted the selected partition.

DISKPART> list part

There are no partitions on this disk to show.

DISKPART>_
```

7. Create a partition with a 1 Mb offset, by running the following command:

```
DISKPART> create partition primary offset=1024

DiskPart succeeded in creating the specified partition.

DISKPART>
```

You can choose a larger size for the partition. The offset you use may vary based on the hardware platform. For more information regarding the offset to use, see the documentation for your hardware platform.

8. Run the `list partition` command, to see the partition you created:

```
DISKPART> list partition

  Partition ###  Type              Size     Offset
  -------------  ----------------  -------  -------
* Partition 1    Primary            256 MB  1024 KB
```


Copyright © 2020, BlackBerry Limited

9. Run the `select partition` command to perform operations on the partition that you created in the previous step:

```
DISKPART> select partition 1

Partition 1 is now the selected partition.

DISKPART>
```

10. Run the `active` command to make the partition active:

```
DISKPART> active

DiskPart marked the current partition as active.
```

11. Run the `format` command to format the partition:

```
DISKPART> format fs=fat32 quick

  100 percent completed

DiskPart successfully formatted the volume.

DISKPART>
```

WARNING: Ensure you specify that drive where microSD is located. If you incorrectly specify the wrong disk, it could result in data loss.

12. Run the `list partition` command to verify that the partition is active and that you have the correct partition size and offset:

```
DISKPART> list partition

  Partition ###  Type              Size     Offset
  -------------  ----------------  -------  -------
* Partition 1    Primary            256 MB  1024 KB

DISKPART>
```

The "*" (asterisk) beside the partition name indicates that the partition is active. You can now copy the QNX IFS file to the partition.

13. Exit the `diskpart` utility:

```
DISKPART> exit

Leaving DiskPart...

C:\windows\system32>
```

Copyright © 2020, BlackBerry Limited

Transfer the image

After you format and make your microSD bootable, you're ready to transfer the image to it.

Provided with your BSP are prebuilt files that you can use. Alternatively, if you make modifications to the buildfile, you can rebuild the IFS. For information about building your BSP, see the "*Build the BSP*" section in this chapter.

Copy the necessary binaries and IFS to target

Obtain boot firmware files

The Raspberry Pi 4 board needs the following firmware files to boot:

```
bcm2711-rpi-4-b.dtb, start4.elf, start4db.elf, start4cd.elf, fixup4.dat,
fixup4cd.dat, fixup4db.dat
```

If you already have a bootable microSD which has files listed above, you can skip downloading them, otherwise these files can be downloaded from *https://github.com/raspberrypi/firmware/tree/stable/boot* and should be saved to ***$BSP_ROOT_DIR*/images/tools/sdboot_images** folder.

For Linux and macOS hosts

Copy the binaries and files from the **/images/tools/sdboot_images** folder and IFS to an microSD, by using the following steps:

1. Insert the microSD into your host system and mount it if necessary.

   The microSD should appear in the list of mounted devices. If it doesn't appear, remove and re-insert the card into your host system. You may need to run the `mount` command before and after inserting the microSD to determine its mountpoint. For example, the microSD mountpoint may appear like:

   ```
   $ mount
   ...
   /dev/sda1 on /media/074B-DAC7
   ...
   ```

2. In a terminal on your host system, navigate to the ***$BSP_ROOT_DIR*/images/tools/sdboot_images** directory and copy all files inside to the microSD using the `cp` command.

   ```
   $ cd $BSP_ROOT_DIR/images/tools/sdboot_images
   $ cp * /media/074B-DAC7/
   ```

3. Copy the QNX IFS image to the microSD as ifs-rpi4.bin. The filename must be ifs-rpi4.bin which is the boot image name defined in the **config.txt** file.

   ```
   $ cd $BSP_ROOT_DIR/images
   $ cp ifs-rpi4.bin /media/074B-DAC7/
   ```

The microSD should now be ready to boot your Raspberry Pi 4 board.

For Windows hosts

1. In a Command Prompt window, run `bash`. You might need to run **qnxsdp-env.bat** from your QNX SDP 7.0 installation to set up your environment to use `bash`.

   ```
   $ bash
   ```


Copyright © 2020, BlackBerry Limited

2. Navigate to the ***$BSP_ROOT_DIR*/images** directory.
3. Copy all binaries and files from **tools/sdboot_images** location to the microSD.

```
$ cd $BSP_ROOT_DIR/images
$ cp tools/sdboot_images/* G:/
```

4. Copy the QNX IFS image to the microSD as ifs-rpi4.bin. The filename must be ifs-rpi4.bin which is the name defined in the **config.txt** file. In the example below, the microSD appears as G: drive.

```
$ cd $BSP_ROOT_DIR/images
$ cp ifs-rpi4.bin G:/
```

Copyright © 2020, BlackBerry Limited

Boot the board

After you transfer the image to an microSD, you can use it to boot your board.

After you boot your board, you should see the board printing to the console. Here's an example of what you should see in the console connection for the Raspberry Pi 4 board:

```
MMU: 16-bit ASID 44-bit PA TCR_EL1=b5183519
GICv2: 256 interrupts
GICv2: routing SPIs to gic cpu 0
cpu0: MPIDR=80000000
cpu0: MIDR=410fd083 Cortex-A72 r0p3
cpu0: CWG=4 ERG=4 Dminline=4 Iminline=4 PIPT
cpu0: CLIDR=a200023 LoUU=1 LoC=2 LoUIS=1
cpu0: L1 Icache 48K linesz=64 set/way=256/3
cpu0: L1 Dcache 32K linesz=64 set/way=256/2
cpu0: L2 Unified 1024K linesz=64 set/way=1024/16
cpu0: GICv2 cpu interface 0
Loading IFS...done
cpu1: MPIDR=80000001
cpu1: MIDR=410fd083 Cortex-A72 r0p3
cpu1: CWG=4 ERG=4 Dminline=4 Iminline=4 PIPT
cpu1: CLIDR=a200023 LoUU=1 LoC=2 LoUIS=1
cpu1: L1 Icache 48K linesz=64 set/way=256/3
cpu1: L1 Dcache 32K linesz=64 set/way=256/2
cpu1: L2 Unified 1024K linesz=64 set/way=1024/16
cpu1: GICv2 cpu interface 1
cpu2: MPIDR=80000002
cpu2: MIDR=410fd083 Cortex-A72 r0p3
cpu2: CWG=4 ERG=4 Dminline=4 Iminline=4 PIPT
cpu2: CLIDR=a200023 LoUU=1 LoC=2 LoUIS=1
cpu2: L1 Icache 48K linesz=64 set/way=256/3
cpu2: L1 Dcache 32K linesz=64 set/way=256/2
cpu2: L2 Unified 1024K linesz=64 set/way=1024/16
cpu2: GICv2 cpu interface 2
cpu3: MPIDR=80000003
cpu3: MIDR=410fd083 Cortex-A72 r0p3
cpu3: CWG=4 ERG=4 Dminline=4 Iminline=4 PIPT
cpu3: CLIDR=a200023 LoUU=1 LoC=2 LoUIS=1
cpu3: L1 Icache 48K linesz=64 set/way=256/3
cpu3: L1 Dcache 32K linesz=64 set/way=256/2
cpu3: L2 Unified 1024K linesz=64 set/way=1024/16
cpu3: GICv2 cpu interface 3

System page at phys:0000000000012000 user:ffffff8040206000 kern:ffffff8040204000
Starting next program at vffffff8060059ae0
MMFLAGS=1
All ClockCycles offsets within tolerance
```


Copyright © 2020, BlackBerry Limited

```
Welcome to QNX Neutrino on the Raspberry Pi 4 (Armv8-A Cortex-A72 core)
Starting watchdog...
Starting random service...
Starting serial driver (/dev/ser1)
Starting serial driver (/dev/ser3)
Starting I2C driver (/dev/i2c1)...
Starting SPI 0 driver (/dev/spi0)...
Starting SDMMC driver for SD card (SDIO0)...
Starting PCI Server...
Starting USB xHCI controller in the host mode (/dev/usb/*)...
Path=0 - bcm2711
 target=0 lun=0     Direct-Access(0) - SDMMC: SM32G Rev: 9.1
Starting Network driver...
Starting Audio driver...
#
```

QNX Neutrino RTOS should now be running on your target. You can test it by executing any shell command, or any command residing within the OS image (`ls`, `pidin`, etc.). You can also type `uname` and it should return `QNX`.

Bootup using UART3 as debug port

The QNX IFS image uses target's mini UART port on the Raspberry Pi 4 board as default debug console port, This BSP also supports using the UART3 port as the default debug console port.

The board's mini UART port serial pins, see "*Set up the hardware*", are:

```
GPIO14 - UART_TXD0, GPIO15- UART_RXD0
```

The board's UART3 port serial pins are:

```
GPIO4 - UART_TXD3, GPIO5 - UART_RXD3
```

Connect a RS-232 serial to USB cable from the board's UART3 serial pins and the USB port on your host machine. The rest of target connection is the same as described in "*Set up the hardware*".

To use UART3 as default debug console port, the BSP IFS needs to be rebuilt. The instruction of using UART3 in the IFS is described in the BSP buildfile, simplay modify the buildfile by uncommenting UART3 related lines and commenting out mini UART related lines. For information about rebuilding BSP, see the "*Build the BSP (commandline)*" section.

Copyright © 2020, BlackBerry Limited

Chapter 4
Build the BSP

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

Generic instructions to modify your BSP (add contents and modify the buildfile) can be found in the *Building Embedded Systems* guide, which is available as part of the QNX SDP 7.0 documentation.

For instructions on how to build this BSP using the IDE, see "*Build the BSP (IDE)*" section in this chapter. For detailed information on how to build using the IDE, see the *IDE User's Guide*, which is available as part of the QNX SDP 7.0 documentation.

If you plan to work with multiple BSPs, we recommend that you create a top-level BSP directory and then subdirectories for each different BSP. The directory you create for a specific 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 a convenience to quickly get QNX Neutrino 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 own environment.

Prebuilt image

After you've unzipped the BSP, a prebuilt QNX IFS image is available in the BSP's **/images** directory. The prebuilt IFS image (**ifs-rpi4.bin**) can be regenerated with the BSP `make` file, and is configured for the BSP device drivers already available for your board.

If you modify and build the IFS, the original IFS files are overwritten. If you need to revert to this prebuilt image, simply run the `make` command from the BSP's root directory using the original buildfiles. To determine the location of the buildfile to modify, open the ***$BSP_ROOT_DIR*/source.xml** file.

If you forget to make a backup copy of the IFS file, you can still recover the original prebuilt IFS; simply extract the BSP from the **.zip** archive into a new directory and get the prebuilt image from that directory.

Build the BSP (commandline)

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

Makefile targets in the BSP

The **Makefile** is located under the ***$BSP_ROOT_DIR*/images** directory. It defines more than one target:

`all`

Builds the QNX IFS for the target.

`ifs-rpi4.bin`

Builds the IFS file.

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, macOS), you must first build at the root directory of the BSP (*BSP_ROOT_DIR*), then you can build using the Makefile targets described previously in the ***$BSP_ROOT_DIR*/images** directory. To build your BSP, 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*."
2. In the BSP's **images** directory (***$BSP_ROOT_DIR*/images**), type `make clean` to remove the default image files from the ***$BSP_ROOT_DIR*/images** directory.

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

   This action removes all existing image files.
3. Navigate back to the *BSP_ROOT_DIR* and type `make`. Running `make` does the following:
   - builds the BSP and creates its directories
   - places any images required for the board into the ***$BSP_ROOT_DIR*/images** directory

```
cd $BSP_ROOT_DIR
make
```

You should now have an IFS file called ifs-rpi4.bin.

After you build, there might be multiple IFS files in the ***$BSP_ROOT_DIR*/images** directory if your **Makefile** builds multiple IFS files.

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

Copyright © 2020, BlackBerry Limited

Build the BSP (IDE)

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

Build in the IDE

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

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 (see "*Import to the QNX Momentics IDE*" for details).
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 **bsp-raspberrypi-bcm2711-rpi4**) 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**.

Build changes to the buildfile

You can make changes to the buildfile(s) 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 the previous section. 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.

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, **ifs-rpi4.bin**; it maps to the `ifs-rpi4.bin` target in the Makefile, you would type `ifs-rpi4.bin.bin` as the target. For more information on the targets available, see the "*Build the BSP (commandline)*" section.

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

Copyright © 2020, BlackBerry Limited

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**.

After your image builds, you should see it appear under the **images** folder in the IDE.

Copyright © 2020, BlackBerry Limited

Chapter 5
Driver Commands

The tables below provide a summary of driver commands for the Raspberry Pi 4 board.

Some of the drivers are commented out in the buildfile. To use these drivers on the target hardware, you may need to uncomment them in the buildfile, rebuild the image, and then 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 your system requirements. For example, if you need an audible sound to play immediately, you must start the audio driver earlier than other drivers. It's important to recognize that the order you choose impacts the boot time.

Some drivers depend on other drivers, so it's sometimes necessary to start them in a specific order.

- 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 7.0 documentation.
- For more information about the drivers and commands listed here, see the QNX Neutrino *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*. In some cases, the driver or command is specific to this BSP, in which case, you'll find the information in the "*BSP-specific Drivers and Utilities*" chapter.

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

- *Audio*
- *Inter-integrated Circuit (I2C)*
- *Network*
- *PCI Server*
- *SD/MMC*
- *Serial*
- *SPI*
- *Startup*
- *USB OTG host controller (io-usb-otg stack)*
- *Watchdog*
- *Wireless (WiFi)*

Audio

By default, we load the BCM2711-PWM audio driver using parameters found in the audio configuration file.

Device	Audio
Command	`io-audio -c /etc/system/config/audio/io_audio.conf`
Required binaries	`io-audio`
Required libraries	**deva-ctrl-bcm2711_pwm.so**, **deva-ctrl-bcm2711_pcm.so**, **deva-mixer-pcm512x.so**, **libasound.so**
Configuration files	**io_audio.conf**
Source locations	***$BSP_ROOT_DIR*/src/hardware/deva/ctrl/bcm2711_pwm**, ***$BSP_ROOT_DIR*/src/hardware/deva/ctrl/bcm2711_pcm**, ***$BSP_ROOT_DIR*/src/hardware/deva/mixer/pcm512x**

The BCM2711-PWM audio driver setup is used when accessing the on-board audio. If using a DAC HAT add-on expansion card, the user will need to switch to using BCM2711-PCM audio driver by uncommenting the DAC HAT section in the io_audio.conf file.

For more information about the **deva-ctrl-bcm2711_pwm.so**, **deva-ctrl-bcm2711_pcm.so**, and the **deva-mixer-pcm512x.so** drivers, see the "*deva-ctrl-bcm2711_pwm.so*" "*deva-ctrl-bcm2711_pcm.so*", and "*deva-mixer-pcm512x.so*" sections, respectively in the "*BSP-specific Drivers and Utilities*" chapter of this guide.

Inter-integrated Circuit (I2C)

Device	I2C
Command	`i2c-bcm2711 -p0xfe804000 --b100000 --u1`
Required binaries	**i2c-bcm2711**
Required libraries	N/A
Source location	***$BSP_ROOT_DIR*/src/hardware/i2c/bcm2711**

For information about this driver, see the "*i2c-bcm2711*" section in the "*BSP-specific Drivers and Utilities*" chapter of this guide.

Network

Device	Ethernet
Command	`io-pkt-v6-hc` `mount -T io-pkt /lib/dll/devnp-genet.so`
Required binaries	**io-pkt-v6-hc**, **if_up**, **dhclient**, **nicinfo**, **ifconfig**, **ping**

Copyright © 2020, BlackBerry Limited

Required libraries	**devnp-genet.so**, **libsocket.so**
Source location	***$BSP_ROOT_DIR*/src/hardware/devnp/genet**

For more information about this driver, see "*devnp-genet.so*" in the "*BSP-specific Drivers and Utilities*" chapter of this guide.

PCI Server

Device	PCI
Command	`pci-server -c --bus-scan-limit=1`
Required binaries	`pci-server`
Required libraries	**pci_hw-bcm2711-rpi4.so**, **pci_server-buscfg-generic.so**, **pci_debug2.so**, **pci_slog2.so**, **pci_strings.so**, **pci_cap-0x01.so**, **pci_cap-0x05.so**, **pci_cap-0x10.so**, **pci_cap-0x11.so**, **pci_bkwd_compat.so**
Source location	Prebuilt only

SD/MMC

Device	SD/MMC
Command (microSD driver on CPU board)	`devb-sdmmc-bcm2711 sdio addr=0xfe340000,irq=158,bs=bmstr_base=0xc0000000 disk name=sd`
Required binaries	**devb-sdmmc-bcm2711**
Required libraries	**libc.so**, **libcam.so**, **io-blk.so**, **cam-disk.so**
Source location	***$BSP_ROOT_DIR*/src/hardware/devb/sdmmc**

Example of mounting a DOS partition on the microSD to the **/fs/sd** directory:

```
devb-sdmmc-bcm2711 sdio addr=0xfe340000,irq=158,bs=bmstr_base=0xc0000000 disk name=sd
waitfor /dev/sd0 3
mount -t dos /dev/sd0t12 /fs/sd
```

Serial

Device	Serial
Command for MINI-UART (USB connected to RS-232 bridge for console)	`devc-serminiuart -b115200 -c500000000 -e -F -u1 0xfe215000,125`

Copyright © 2020, BlackBerry Limited

Command for PL011 (UART3)	`devc-serpl011 -b115200 -c48000000 -e -F -u3 0xfe201600,153`
Required binaries	`devc-serminiuart`, `devc-serpl011`
Required libraries	N/A
Source location	***$BSP_ROOT_DIR*/src/hardware/devc**

For more information about the serial drivers, see the "*devc-serminiuart*" and "*devc-serpl011*" section in the "*BSP-specific Drivers and Utilities*" chapter of this guide.

SPI

Device	SPI
Command—Slot 0 (without DMA)	`spi-master -u0 -d bcm2711 base=0xfe204000,opmode=1`
Command—Slot 0 (with DMA)	`spi-master -u0 -d bcm2711 base=0xfe204000,opmode=2,dmarxbase=0xfe007200,dmatxbase=0xfe007300`
Required binaries	`spi-master`
Required libraries	**spi-bcm2711.so**
Source location	***$BSP_ROOT_DIR*/src/hardware/spi/bcm2711** ***$BSP_ROOT_DIR*/src/hardware/spi/master**

For more information about this driver, see the "*spi-master*" section in the "*BSP-specific Drivers and Utilities*" chapter of this guide.

Startup

Device	Startup
Command	`startup-bcm2711-rpi4 -v -D miniuart -W 2500`
Required binaries	`startup-bcm2711-rpi4`
Required libraries	N/A
Source location	***$BSP_ROOT_DIR*/src/hardware/startup/boards/bcm2711** (common) ***$BSP_ROOT_DIR*/src/hardware/startup/boards/bcm2711/rpi4** (Raspberry PI 4 board)

In addition to the "`startup-*`" options described in the *Utilities Reference*, you can also use the following option with this command:

Copyright © 2020, BlackBerry Limited

-W

Enable watchdog timer support. Ensure that you start the *watchdog* driver after when you use this option.

USB OTG host controller (`io-usb-otg` stack)

CAUTION: It is mandatory to run the `pci-server` before starting this driver since the USB driver on this board is dependent on PCI bus being functional.

USB host controller and OTG controller; host mode uses `io-usb-otg` stack.

Device	USB host controller
Command	XHCI: `io-usb-otg -d bcm2711-xhci ioport=0x600000000,iosize=4096,irq=175`
Required binaries	`io-usb-otg`, `usb`, `devb-umass`
Required libraries	**devu-hcd-bcm2711-xhci.so**
Source location	Prebuilt only

Example of xHCI controller:

```
io-usb-otg -d bcm2711-xhci ioport=0x600000000,iosize=4096,irq=175
waitfor /dev/usb/io-usb-otg 4
waitfor /dev/usb/devu-hcd-bcm2711-xhci.so 4 4
devb-umass cam pnp
```

When a mass storage device is plugged in, `devb-umass` creates a resource manager under **/dev/hd*X*t*Y*** where *X* represents the partition number and *Y* represents the partition type. For example, if you use a mass storage with one FAT32 partition (MBR), it creates two resource manager paths, which are **/dev/hd0** and **/dev/hd0t11**. You would mount the second resource manager, where in this example, "11" specifies a FAT32 filesystem (i.e., `mount -t dos /dev/hd0t11 /fs/usb0`).

For more information about the **devu-hcd-bcm2711-xhci.so** driver, see the "*devu-hcd-bcm2711-xhci.so*" section in the "*BSP-specific Drivers and Utilities*" chapter.

Watchdog

To enable the watchdog:

1. Launch `startup` with the `-W` option, with an optional parameter for timeout:

   ```
   startup-bcm2711-rpi4 -v -D miniuart -W 2500
   ```

2. Launch the watchdog timer utility early on in the boot script:

   ```
   wdtkick -W0x24:0x5A028E4C -W0x1c:0x5A000020
   ```

3. To adjust the watchdog timeout value of the PM_WDOG register, use the formula:

   ```
   register_val = ((timeout_in_ms * 67) | 0x5a000000)
   ```

Copyright © 2020, BlackBerry Limited

For example, use 0x5A028E4C for 2500 ms timeout value:

```
((2500 * 67) | 0x5a000000) = 0x28E4C | 0x5a000000 = 0x5A028E4C
```

Device	Watchdog timer
Commands	`wdtkick -W0x24:0x5A01000A -W0x1c:0x5A000020`
Required binaries	**wdtkick**
Required libraries	N/A
Source location	***$BSP_ROOT_DIR*/src/hardware/support/wdtkick**

Wireless (WiFi)

To support WiFi:

1. Obtain WiFi driver

   `devnp-qwdi-2.5_bcm4339-rpi4.so`

   from QNX and save it to ***$BSP_ROOT_DIR*/prebuilt/aarch64le/lib/dll** directory.

   Note that the WiFi driver (devnp-qwdi-2.5_bcm4339-rpi4.so) is not included in the BSP and may not be available with the general QNX SDP product distribution at the time of the BSP's release.

2. Downloaded WiFi firmware files

   `brcmfmac43455-sdio.bin, brcmfmac43455-sdio.clm_blob, brcmfmac43455-sdio.txt`

   from *https://github.com/RPi-Distro/firmware-nonfree/tree/master/brcm* and save them to ***$BSP_ROOT_DIR*/prebuilt/etc/firmware** directory.

3. Rebuild BSP by modifying .build file in ***$BSP_ROOT_DIR*/images** directory to include the WiFi driver and firmware files, and adding binaries to support WiFi network configuration and administration, such as:

   `wpa_supplicant, wpa_passphrase and wpa_cli`

CAUTION: Make sure **io-pkt-v6-hc** is already running before mounting the WiFi driver.

Device	Wifi driver
Commands	`mount -T io-pkt -o fw=/etc/hotspot/firmware.bin,nvram=/etc/hotspot/nvram.txt, clm_blob=/etc/hotspot/firmware.clm_blob /lib/dll/devnp-qwdi-2.5_bcm4339-rpi4.so` `wpa_supplicant -D qwdi -i bcm0 -c /tmp/wpa_supplicant.conf &`
Required binaries	**firmware.bin**, **firmware.clm_blob**, **nvram.txt**, **io-pkt-v6-hc**, **if_up**, **dhclient**, **nicinfo**, **ifconfig**, **ping**

Copyright © 2020, BlackBerry Limited

Required libraries	**devnp-qwdi-2.5_bcm4339-rpi4.so**, **libsocket.so**
Source location	Prebuilt only

Copyright © 2020, BlackBerry Limited

Chapter 6
BSP-specific Drivers and Utilities

These are the specific drivers and utilities for this BSP.

- *deva-ctrl-bcm2711_pwm.so*
- *deva-ctrl-bcm2711_pcm.so*
- *deva-mixer-pcm512x.so*
- *devc-serminiuart*
- *devc-serpl011*
- *devnp-genet.so*
- *devu-hcd-bcm2711-xhci.so*
- *i2c-bcm2711*
- *spi-master*

Copyright © 2020, BlackBerry Limited

deva-ctrl-bcm2711_pwm.so

BCM2711-PWM Sound driver for on-board audio on the Raspberry Pi 4 board

Syntax:

From a configuration file:

```
io-audio -c location_of_audio_config_file
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-bcm2711_pwm.so
```

Runs on:

QNX Neutrino

Options:

pwm=*value*

PWM controller (0-1).

dma=*value*

DMA channel (0-14, default value is 2).

rate=*value [:value]*

Sample rate min/max value in Hz (default 44,100/48,000), or locked sample rate if only one value is provided.

clk_mash=*value*

Clock manager mash value (0-3, default is 1).

msen=*value*

M/S enable (0..1, default is 1) - operation mode is PWM N/M modulation (0) or PWM M/S serialization (1).

dreq=*value*

DMA Threshold for DREQ signal (0-127, default is 7).

panic=*value*

DMA Threshold for PANIC signal (0-127, default is 7).

For card options that apply to all sound drivers, see "`io-audio`" in the *QNX Neutrino Utilities Reference* guide.

The pool_size should be indicated as 260 (256+4) kB. The 4k addition accounts for the memory used for DMA control blocks.

Copyright © 2020, BlackBerry Limited

With increased sample rate values, the sample precision (how many MSB bits of the sample are actually used) decreases. The optimal range is 44,100 to 48,000 Hz, where we have an optimal compromise between sample precision (12 bits) and sample rate.

Description:

The `deva-ctrl-bcm2711_pwm.so` shared object is a device driver DLL used by the `io-audio` manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-bcm2711_pwm.so` is running, you can use applications with sound, and those that control the sound system. Ensure that you use the `pool_name` and `pool_size` global options so that DMA buffer allocation is below the 1 GB address space.

```
io-audio -m pool_name=below1G,pool_size=desired_pool_size_in_KB  -d bcm2711_pwm
```

Examples:

- Invoke `deva-ctrl-bcm2711_pwm.so` using a configuration file `io-audio`:

  ```
  io-audio -c /etc/system/config/audio/io_audio.conf
  ```

  And then in the audio configuration file (**io_audio.conf**):

  ```
  [global]
  pool_name=below1G
  pool_size=260

  [ctrl]
  name=bcm2711_pwm
  ```

  if only the `bcm2711_pwm` driver instance is used, the required `pool_size` is 260 kB. When using the `bcm2711_pcm` driver instance in conjuction with the `bcm2711_pwm`, the required `pool_size` is 520 kB.

- Mount `deva-ctrl-bcm2711_pwm.so` (`io-audio` must be running and where ***mydriver*.conf** is the name of your driver-specific configuration file):

  ```
  mount -Tio-audio -oconf=mydriver.conf /lib/dll/deva-ctrl-bcm2711_pwm.so
  ```

  For information about how to use the `mount -Tio-audio` command, see the "Mounting a driver" section in the "`io_audio`" chapter of the QNX Neutrino *Utilities Reference*.

Errors:

When an error occurs, `deva-ctrl-bcm2711_pwm.so` sends a description of the error to the system logger (see `slogger2` in the *QNX Neutrino Utilities Reference* guide).

deva-ctrl-bcm2711_pcm.so

PCM/I2S Sound driver for the PCM interface which is routed to the GPIO expander connector.

Any PCM device connected through the GPIO expander can be controlled using this driver. However, for purposes of this User Guide, we will be focusing on expansion boards which offer the TI PCM512x DAC codec. For more details, refer to the "*deva-mixer-pcm512x.so*" section of this guide.

Syntax:

From a configuration file:

```
io-audio -c location_of_audio_config_file
```

Mounting (requires that `io-audio` already be running):

```
mount -Tio-audio [-oopt[,opt...]] /lib/dll/deva-ctrl-bcm2711_pcm.so
```

CAUTION: If using the mixer DLL `deva-mixer-pcm512x.so`, users should use the *configuration file* syntax. For more details, refer to the "*deva-mixer-pcm512x.so*" section of this guide.

Runs on:

QNX Neutrino

Options:

tx_voices=*value*

Playback channel voices (0 if playback not configured, 2 otherwise, default is 2).

rx_voices=*value*

Capture channel enable (0 if capture not configured, 2 otherwise, default is 0).

tx_dma=*value*

DMA channel used for playback (0-14, default is 2).

rx_dma=*value*

DMA channel used for capture (0-14, default is 3).

tx_dreq=*value*

DMA threshold for DREQ signal for playback (0-127, default is 48).

tx_panic=*value*

DMA threshold for PANIC signal for playback (0-127, default is 16).

Copyright © 2020, BlackBerry Limited

rx_dreq=*value*

DMA threshold for DREQ signal for capture (0-127, default is 32).

rx_panic=*value*

DMA threshold for PANIC signal for capture (0-127, default is 48).

gpio=*value [:value]*

GPIO pin set ("primary" or "secondary", default is "primary").

clk_src=*value*

Clock manager source ("osc" or "plld", default is "osc").

clk_mash=*value*

Clock manager mash value (0-3, default is 0).

clk_mode=*value*

Clock mode for CLK and FS clocks signals ("master" or "slave", default is "master").

rate=*value [:value]*

Sample rate min/max value in Hz (default 8,000/192,000), or locked sample rate if only one value is provided.

bits=*value*

Sample bit width in bits (16, 24 or 32, default is 32).

bit_delay=*value*

Bit delay of first bit in a frame in bit clock cycles (default is 1, corresponding to I2S mode).

clk_pol=*value*

Clock polarity (0-1, default is 0).

0 - Outputs change on falling edge of clock, inputs are sampled on rising edge (I2S mode).

1 - Outputs change on rising edge of clock, inputs are sampled on falling edge (PCM mode).

fs_pol=*value*

Frame Sync polarity (0-1, default is 0).

0 - FS is normally high and goes low to indicate frame sync (I2S mode).

1 - FS is normally low and goes high to indicate frame sync (PCM mode).

For card options that apply to all sound drivers, see "`io-audio`" in the *QNX Neutrino Utilities Reference* guide.

Description:

The `deva-ctrl-bcm2711_pcm.so` shared object is a device driver DLL used by the `io-audio` manager. It uses the API described in the *Audio Developer's Guide*.

While `deva-ctrl-bcm2711_pcm.so` is running, you can use applications with sound, and those that control the sound system. Ensure that you use the `pool_name` and `pool_size` global options so that DMA buffer allocation is below the 1 GB address space.

```
io-audio -m pool_name=below1G,pool_size=desired_pool_size_in_KB  -d bcm2711_pcm
```

Additional configuration:

You can load a codec mixer DLL using the `mixer_dll` key in the `[CTRL]` section of the audio configuration file or driver-specific configuration file. For more information about the audio configuration file and driver-specific configuration files, see the "Audio configuration file" and "Mounting driver" sections in the the "io-audio" chapter of the QNX Neutrino *Utilities Reference*. You can also start this driver without the `mixer_dll` key in the audio configuration file if want to run without attaching an audio codec. (i.e., connect to a BT chip, DSP or a codec that doesn't have any software controls).

Examples:

- Invoke `deva-ctrl-bcm2711_pcm.so` using a configuration file `io-audio`:

  ```
  io-audio -c /etc/system/config/audio/io_audio.conf
  ```

  And then in the audio configuration file (**io_audio.conf**):

  ```
  [global]
  pool_name=below1G
  pool_size=260

  [ctrl]
  name=bcm2711_pcm
  options=rate=48000,bits=16,tx_dma=4
  mixer_dll=pcm512x
  pcm512x_i2c_dev=1
  pcm512x_i2c_addr=77
  ```

  > if only the `bcm2711_pcm` driver instance is used, the required `pool_size` is 260 kB. When using the `bcm2711_pwm` driver instance in conjuction with the `bcm2711_pcm`, the required `pool_size` is 520 kB.

- Mount `deva-ctrl-bcm2711_pcm.so` (`io-audio` must be running and where ***mydriver*.conf** is the name of your driver-specific configuration file):

  ```
  mount -Tio-audio -oconf=mydriver.conf /lib/dll/deva-ctrl-bcm2711_pcm.so
  ```

  For information about how to use the `mount -Tio-audio` command, see the "Mounting a driver" section in the "`io_audio`" chapter of the QNX Neutrino *Utilities Reference*.

- Invoke `deva-ctrl-bcm2711_pcm.so` directly from `io-audio`:

  ```
  io-audio -m pool_name=below1G,pool_size=260 -d bcm2711_pcm rate=48000,bits=16,tx_dma=4
  ```


Copyright © 2020, BlackBerry Limited

Files:

deva-mixer-pcm512x.so

Supports the mixer for the TI PCM512X DAC codec.

Errors:

When an error occurs, `deva-ctrl-bcm2711_pcm.so` sends a description of the error to the system logger (see `slogger2` in the *QNX Neutrino Utilities Reference* guide).

Copyright © 2020, BlackBerry Limited

deva-mixer-pcm512x.so

Mixer DLL for the TI PCM512X DAC codec.

You can't invoke this shared object. The audio driver (`deva-ctrl-bcm2711_pcm.so`) loads the mixer DLL based on the audio configuration file (i.e., use the `mixer_dll` key in the `[CTRL]` section).

Syntax:

N/A

Runs on:

QNX Neutrino

Options:

None.

Description:

The **deva-mixer-pcm512x.so** shared object provides an interface between the PCM512x codec and audio driver. The mixer's configuration options should be incorporated under the `[CTRL]` section of the corresponding audio controller driver. For more information, see the io-audio in the QNX Neutrino *Utilities Reference*.

The following are configuration file options that are **mandatory** to be specified in the `[CTRL]` section of the audio configuration file:

`pcm512x_i2c_dev=`*value*

I2C device number corresponding to the PCM512X DAC.

`pcm512x_i2c_addr=`*value*

I2C slave address corresponding to the PCM512X DAC.

The following are configuration file options that are **optional** to be specified in the `[CTRL]` section of the audio configuration file:

`pcm512x_joint_vol=`*"enable" or "disable"*

Enable joint volume (when setting L-channel volume, it is applied to both channels, default: disable).

`pcm512x_analog_gain=`*float_value*

Set the analog gain in dB (possible values are -6, -5.2, 0, 0.8, default: 0).

Copyright © 2020, BlackBerry Limited

pcm512x_gpio_out(n)=*selection*

Enable GPIO (n) ((n)=1..6) as output and output the selected signal. Possible "selection" values are:

"dsp" - DSP output

"high" - GPIO set as high

"low" - GPIO set a low

"automute_b" - auto mute flag asserted for both channels

"automute_l" - auto mute flag asserted for left channel

"automute_r" - auto mute flag asserted for right channel

"clkinval" - clock invalid flag

"sdout" - serial data output

"anamute_l" - analog mute flag for left channel

"anamute_r" - analog mute flag for right channel

"pllock" - PLL lock flag

"cpclk" - charge pump clock

"undervolt0_7" - under voltage flag, asserted when XSMUTE voltage is higher than 0.7 DVDD

"undervolt0_3" - under voltage flag, asserted when XSMUTE voltage is higher than 0.3 DVDD

"pllout" - PLL output

By adding the suffix "_inv" to any of these selection value other than "high" and "low", the selected signal is inverted.

Default: GPIO (n) is an input.

pcm512x_debug=*"enable" or "disable"*

Enable register dumps (default: disable).

Errors:

When an error occurs, **deva-mixer-pcm512x.so** sends a description of the error to the system logger (see slogger2 in the QNX Neutrino *Utilities Reference*).

devc-serminiuart

Serial driver for the mini UART on the Raspberry Pi 4 board.

Syntax:

```
devc-serminiuart [-b number]
                 [-B address]
                 [-c clock[/div]]
                 [-C size]
                 [-e] [-E]
                 [-f] [-F]
                 [-I size]
                 [-o opt[,opt...]
                 [-O number]
                 [-s] [-S]
                 [-t level]
                 [-T level]
                 [-u serial_unit_num]
                 [-v]
```

Runs on:

QNX Neutrino

Options:

This driver supports generic `devc-ser*` options. For information about those options, see "`devc-ser*`" in the QNX Neutrino *Utilities Reference* guide in the QNX Software Development Platform 7.0 documentation.

In addition to the generic `devc-ser*` options, this driver supports the following options:

-c ***clk[/div]***

Set the input clock rate and an optional divisor.

-i ***mode***

Set the interrupt mode. This can be one of these values:

- 0 — event mode
- 1 — ISR mode mode

-u ***serial_unit_num***

Set serial unit number. The default is 1.

Copyright © 2020, BlackBerry Limited

Examples:

Disable the hardware and software flow control for a UART device with the base address of 0xFE215000 and the IRQ 125.

```
devc-serminiuart -b115200 -c500000000 -e -F -u1 0xfe215000,125
```

Copyright © 2020, BlackBerry Limited

devc-serpl011

Serial driver for PL011 on the Raspberry Pi 4 board.

Syntax:

```
devc-serpl011 [-b number]
              [-B address]
              [-c clock[/div]]
              [-C size]
              [-e] [-E]
              [-f] [-F]
              [-i interrupt_mode]
              [-I size]
              [-o opt[,opt...]
              [-O number]
              [-s] [-S]
              [-u serial_unit_num]
              [-v]
```

Runs on:

QNX Neutrino

Options:

This driver supports generic `devc-ser*` options. For information about those options, see "`devc-ser*`" in the QNX Neutrino *Utilities Reference* guide in the QNX Software Development Platform 7.0 documentation.

In addition to the generic `devc-ser*` options, this driver supports the following options:

-c ***clk[/div]***

Set the input clock rate and an optional divisor.

-t ***level***

Set the receive FIFO trigger level; can be set between 0 - 3. The default trigger level is 2.

-T ***level***

Set the transmit FIFO data trigger level; can be set between 0 - 3. The default trigger level is 2.

-u ***serial_unit_num***

Set serial unit number. The default is 1.

Copyright © 2020, BlackBerry Limited

Examples:

Disable the hardware and software flow control for a UART device with the base address of 0xFE201600 and the IRQ 153.

```
devc-serpl011 -b115200 -c48000000 -e -F -u3  0xfe201600,153
```

Copyright © 2020, BlackBerry Limited

devnp-genet.so

Driver for the Raspberry Pi GENET ETHERNET

Syntax:

```
io-pkt-v6-hc -d genet [option[,option ...]]
```

Runs on:

QNX Neutrino

Options:

The following are the options for this driver.

Generic options

The following are generic options used by this driver.

speed=10|100|1000

Force the media data rate in Mbits/second.

duplex=0|1

Force half (0) or full (1) duplex mode. The default is automatically detected on supported hardware. If you set to duplex, you should also set the `speed` for this driver.

irq=*irq_num*

IRQ of the interface.

mac=*mac_interface_addr*

Interface address of the controller.

verbose[=*level*]

Set the verbosity level. When *level* is not specified, then level 1 verbosity is used and increases each time you specify the verbosity option. You can set the verbosity level explicitly as follows:

- 1–general debugging
- 2–general and PHY debugging
- 4–Rx, PHY, and general debugging
- 8–Tx, Rx, PHY, and general debugging

Board-specific options

The driver currently supports no board-specific options.

Copyright © 2020, BlackBerry Limited

The MAC address for the board is stored in the SYSPAGE. The driver has built-in support to access the unique MAC address from the SYSPAGE.

Examples:

Start networking with the devnp-genet driver:

```
io-pkt-v6-hc -d genet
ifconfig genet0 192.0.2.1
```

Copyright © 2020, BlackBerry Limited

devu-hcd-bcm2711-xhci.so

Driver for Extensible Host Controller Interface (XHCI) USB controllers on Raspberry Pi 4 board variants

Syntax:

```
io-usb-otg -d hcd-bcm2711-xhci [option[,option...]]
```

or

io-usb-otg -d bcm2711-xhci [*option*[,*option*...]]

Runs on:

QNX Neutrino

Targets:

aarchle64

Options:

cerr=*retries*

Set the number of hardware retries. You can set *retries* to a value of 0 to 3, where 0 indicates unlimited retries. The default is 3 retries.

debug_listener=*retries*

Starts the PPS debug command handler at **/pps/usb/debug**.

error_threshold=*num*

The number (*num*) of cumulative errors before triggering the driver to reinitialize. The error-based triggering is disabled by default.

ioport=*addr*

USB controller register base address. The default is to scan the PCI bus.

iosize=*size*

The register space size. The default is 64 kilobytes.

irq=*num*

The interrupt request number.

max_reinit_attempts=*num*

The maximum number of reinitializations allowed. No limit is the default.

memory=*name*

Set the typed-memory name for descriptors. The default is **/memory/below4G**.

Copyright © 2020, BlackBerry Limited

nousb3

Do not enable USB3 mode.

pindex=*num*

Instance of controller on PCI bus to apply argument.

prio=*num*

Set the pulse-handler thread's priority. The default is 21.

soft_retries=*num_retries*

Set the number of software retries for a failed transfer. Supported only on control, bulk, and interrupt endpoints. Software retries are disabled by default.

soft_retries_mask=*num*

Specifies the endpoints to use when software retries (`soft_retries`) are set. The endpoint to use is specified using a mask. The bit masks include:

- 0x1 – Control endpoint, which is set (ON) by default.
- 0x2 – Reserved endpoint, which isn't set (OFF). This bit should never be set.
- 0x4 – Bulk endpoint, which is set (ON) by default.
- 0x8 – interrupt endpoint, which is set (ON) by default.

The default mask is `0xD`, which indicates that the control, bulk, and interrupt endpoint bits are set (ON).

verbose=*level*

Set the verbosity level (0-7).

Description:

The `devu-hcd-bcm2711-xhci.so` server provides support for computers that have Extensible Host Controller Interface (XHCI) USB controllers. This server creates the **/dev/usb/devu-hcd-bcm2711-xhci.so device** device.

The `devu-hcd-bcm2711-xhci` controller supports SuperSpeed, high, full, and low speed devices.

Examples:

Attach to the controller after scanning the PCI bus.

```
io-usb-otg -d hcd-bcm2711-xhci
```

i2c-bcm2711

Driver for inter-integrated circuits on the Raspberry Pi 4 Board (Model B).

Syntax:

```
i2c-bcm2711 [board_specific_option [board_specific_option]...]
        [generic_option [generic_option]...]]
```

Runs on:

QNX Neutrino

Options:

There are Raspberry Pi 4 board-specific (single-dash) and generic (double-dash) options available.

CAUTION: The Raspberry Pi 4 board-specific options must always appear before generic options, otherwise the driver fails to initialize. For example, the correct sequence would as follows:

```
i2c-bcm2711 -v --u0
```

However, if you incorrectly issue the command in the wrong sequence as shown here, it will fail:

```
i2c-bcm2711 --u0 -v
```

Raspberry Pi 4 board-specific options:

These are the board-specific options available for this driver.

-c ***clockspeed***

Input clock. If this option isn't specified, the default is set as -1.

-g ***gpio_num***

The GPIO pin used for I2C SDA. The default is set as 255.

-i ***irq***

The I2C interrupt event number. The default is 149.

-p address

The I2C physical base address.

-s slave_address

The slave address. The default is set as 0.

-v

Set to be verbose.

Copyright © 2020, BlackBerry Limited

Generic options:

These are generic options for the driver.

--b

The default bus speed. If this option isn't specified, the default is 100000.

--m*max_msg_length*

The minimum length of the resource manager receive buffer. The default is 128 bytes.

--n*nparts*

The minimum number of IOV_t structures in the resource manager context. The default is 2.

--u*unit*

The unit number to spend to the device name. The default is 0.

Example:

Specify a port of address 0xFE804000 with an IRQ of 149 to the first unit:

```
i2c-bcm2711  -p0xfe804000  --b100000 --u1
```

Copyright © 2020, BlackBerry Limited

spi-master

Enhanced Serial Peripheral Interface (SPI)

Syntax:

```
spi-master [-P permissions] [-u unit]
           [-d modulename] [override_options,[override_options]...]
```

Runs on:

QNX Neutrino

Options:

These are the options available for this driver.

-d ***modulename***

The driver module name.

-P ***permissions***

Set the permissions for the device. The default permissions are 0666 (non-directory user: read/write, group: read/write, and other: read/write.

-u ***unit***

Set the SPI unit number. The default is 0.

Options to override autodetected options:

base=*address*

The base address of SPI controller. The default is 0xFE204000.

clock=*num*

Defines the value of the clock source of the SPI. The default reference clock is set at 500 MHz.

dmarxbase=*address*

Defines DMA RX channel base address. Default is set as DMA4 base: 0xFE007400.

dmatxbase=*address*

Defines DMA TX channel base address. Default is set as DMA5 base: 0xFE007500.

opmode=*1 or 2*

Software operation mode, where:

1 is for interrupt-mode

2 is for DMA-mode

Copyright © 2020, BlackBerry Limited

rdreq=*num*

Defines DMA read request threshold. The default is set at 32.

rpanic=*num*

Defines the read panic threshold. The default is set at 48.

tdreq=*num*

Defines DMA write request threshold. The default is set at 32.

tpanic=*num*

Defines the write panic threshold. The default is set at 16.

Examples:

Start SPI driver for SPI0 with a base address specified and interrupt-mode selected:

```
spi-master -u0 -d bcm2711 base=0xfe204000,opmode=1
```

Start SPI driver for SPI0 with a base address and IRQ and DMA set-up:

```
spi-master -u0  -d bcm2711 base=0xfe204000,
                opmode=2,dmarxbase=0xfe007200,dmatxbase=0xfe007300
```

Copyright © 2020, BlackBerry Limited