BSP User's Guide

Renesas R-Car V3H System Evaluation Board (Condor)
and Renesas R-Car V3H Starter Kit

BlackBerry | QNX

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

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

Electronic edition published: March 28, 2020

Contents

About This Guide

This guide contains installation and startup instructions for the Renesas R-Car V3H System Evaluation Board (Condor) (referred to as the R-Car V3H board) and Renesas R-Car V3H Starter Kit (referred to as the R-Car V3H Starter Kit) Board Support Package (BSP).

Only the 64-bit variant of this board is supported from QNX Software Systems. Unless stated, the instructions for the R-Car V3H board are applicable to the R-Car V3H Starter Kit.

This guide provides the following information:

- an overview of the BSP
- structure and contents of a BSP
- how to build the BSP using the command line or the QNX Momentics IDE
- how to transfer the image to boot the board
- 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 supported host OSs and boards	*About This BSP*
Download and extract the contents of the BSP	*Download and set up the BSP*
Installing this BSP	*Installation Notes*
Using the Screen Graphics Subsystem	*Using the Screen Graphics Subsystem*
Building this BSP	*Build the BSP*
Driver commands	*Driver Commands*

Typographical conventions

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

The following table summarizes our conventions:

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

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

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

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

Notes point out something important or useful.

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

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.

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.

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 begin building and installing your BSP, review the following documentation:

- Information about your board's hardware and firmware provided by the board vendor. The Renesas website is at *www.renesas.com*.
- *Building Embedded Systems*. This guide contains information that's relevant to all QNX BSPs; it's available from QNX Software Development Platform 7.0 documentation.

Required software

To work with this BSP, you require the following:

- the ZIP file for the BSP
- the QNX Software Development Platform 7.0 installed on a Linux, macOS, or Windows host system.
- a 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 (e.g., Tera Term) to connect to the board for debugging. Alternatively, you can use a `telnet` or `ssh` session from the QNX Momentics IDE.

Chapter 2
About This BSP

These notes list what's included in the BSP, supported host system operating systems, and other relevant information to use this BSP.

What's in this BSP

This BSP contains the components listed in the table below. For more detailed information about binaries and libraries, see "*Driver commands*."

Component	Format	Comments
eMMC driver	Source/Binary	
Graphics driver	Provided as a separate package	Screen Graphics Subsystem
Inter-integrated Circuit (I2C) driver	Source/Binary	
Network driver	Source/Binary	
PCI server	Binary	
QNX IPL	Source/Binary	
Serial driver	Source/Binary	SCIF
SPI	Source/Binary	
SPI Flash driver	Source/Binary	
Startup	Source/Binary	
Thermal driver	Source/Binary	
Watchdog driver	Source/Binary	

Supported OSs

In order to install and use this BSP, you must have installed the QNX SDP 7.0, on a Linux, macOS, or Windows host system.

This BSP supports QNX Neutrino RTOS 7.0.

Interrupt table for this board

The interrupt vector table can be found in the buildfile located at: **image/**. For example, **rcar_v3h.build** is the buildfile for the R-Car V3H board and R-Car V3H Starter Kit. This interrupt table is useful when you want to write custom resource managers or customize the BSP.

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**.

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

Required cables

To connect to your board, you need:

- a miniUSB-serial cable
- an Ethernet cable
- the power supply that came with your board

Known issues

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

Chapter 3
Installation Notes

These installation notes describe how to build, install, and start this BSP.

To build and install the R-Car V3H Board and R-Car V3H Starter Kit BSP, you need to:

1. Download the BSP, setup your environment, and extract the BSP. For more information, see "*Download and set up the BSP*."
2. Flash the loaders to your board. The steps include setting up and configuring the board, booting to MiniMonitor, and then programming the boot loaders to the QSPI memory on the board. The steps you choose depend on the board you have. For more information, see "*Flash loaders to the board*."

   When you connect your board, it may be necessary set DIP switches on your board to further configure it. For more information about how to connect to your board and set DIP switches, see appropriate "Set up and configure the hardware" section for your board.
3. Transfer the image and boot the board. If this is the first time you're bringing up the BSP, you can transfer the provided prebuilt IFS image file into RAM from a TFTP server that's connected to your board if you want to boot using U-Boot; otherwise, if you use want to boot using QNX IPL, you must transfer both the QNX IPL and IFS to the board.

The default buildfile creates a generic QNX-IFS and QNX IPL file that starts all available drivers available with this BSP. At some point, you may want to customize the buildfile to meet your own configuration requirements.

For specific information about building the image for this board, see the "*Build the BSP*" chapter in this guide. For information about modifying the image and generic information about building the image, see the *Building Embedded Systems* 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 **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 command line to a working directory that's outside 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 command line

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 BSP was downloaded and create a directory where you want to extract the BSP (e.g., `cd` **/BSPs/r-car-v3h**). 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_renesas-r-car-vh_br-mainline_be-700***build_ID***.zip**

   where *build_ID* is the BSP build number (e.g., `JBN7`).

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

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-renesas-r-car-v3h**). 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 command line, or if you look at the source that's extracted by the IDE in your workspace, the resulting directory structure looks something like this:

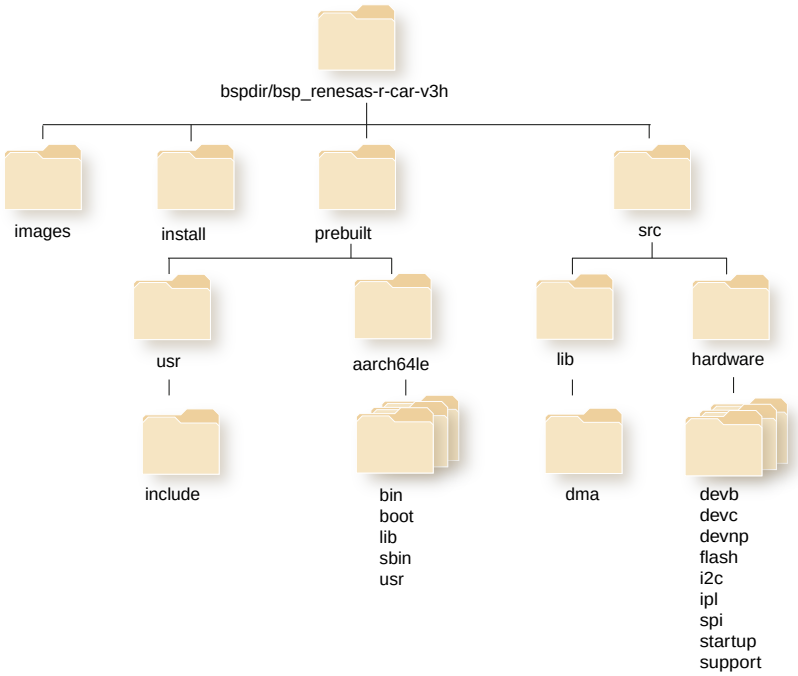

Figure 1: Structure of a BSP

For more information about the contents of the directory structure, see the *Building Embedded Systems* guide in the QNX SDP 7.0 documentation.

Set up and configure the hardware

You must attach the appropriate connectors to your hardware.

You should refer to the hardware guide provided by your target board provider for more information about the board. Here are things that you should consider:

- how to connect to power and the proper power supply to use
- console, Ethernet port, or other connectors such as the parallel port for a camera
- any board switches that require setting to boot or storage devices, such as Flash memory on the board

Connect to the board

CAUTION:

Use only the power supply provided with the board. Refer to the hardware manufacturer's documentation for your board. If you use an incorrect power supply, you may permanently damage the board!

For this board, connecting the power supply to the board powers it on.

Refer to the hardware manuals Renesas provides for detailed information about connecting your target board with your host system and location of the power switch.

To connect to your board, after you've installed the virtual COM drivers on your host system:

1. Connect the miniUSB-serial cable from the board's DebugSerial port to your host system. It'll show up as the first USB Serial Device on your host system (e.g., **/dev/ttyUSB*** on Linux).
2. Connect an Ethernet cable from the board's RJ45 connector to your network.
3. Connect the power supply to the power connector on the board. Depending on the board you have, this can either be a DC 12V or DC 5V power supply. Ensure that you check the manufacturer's documentation for the proper power supply to use for your board.
4. On your host machine, start your favorite terminal program (e.g., Tera Term version 4.75 or later) with these settings:
   - Baud rate: **115200**
   - Data: **8 bit**
   - Parity: **none**
   - Flow control: **none**
   - Stop: 1-bit

Configure the board switches for the R-Car V3H board

You need to change the settings on SW5 (selects QSPI0 Connection A), SW6 (selects QSPI0 Connection B), and SW7 (selects QSPI0 Connection C)on your board when you want to:

- Boot from the QSPI flash (S25FS128) at 40 MHz DMA to load QSPI MiniMonitor.

The following table summarizes the settings to boot QSPI flash memory to load the QSPI MiniMonitor for SW5, SW6, and SW7:

Switch number	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6
SW5	OFF (1-pin side)	–	–	–	–	–
SW6	ON [pin 3 side (to the dot)]	–	–	–	–	–
SW7	ON	ON	ON	ON	ON	ON

This setting represents also the initial DIP switch setting on this board.

- Settings to use the QSPI Flash memory (S25FS512) at 40 MHz DMA:

  The following table summarizes the settings for the switches SW5, SW6, and SW7 to program the QSPI Flash memory (S25FS512).

Switch number	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6
SW5	ON [pin 3 side (to the dot)]	–	–	–	–	–
SW6	OFF [pin 1 side]	–	–	–	–	–
SW7	OFF	OFF	OFF	OFF	OFF	OFF

For more information, see the hardware and/or software manual for the board from the hardware vendor.

Configure the board switches for the R-Car V3H Starter Kit

- The following table summarizes the settings to select the SW4 switches you must set on the board:

Boot Source Selection	SW4.1	SW4.2	SW4.3	SW4.4	SW4.5	SW4.6	SW4.7	SW48
HyperFlash (160 MHz, DMA)	ON	OFF	ON	ON	OFF	OFF	OFF	–
HyperFlash (160 MHz, XIP)	ON	OFF	ON	OFF	OFF	OFF	OFF	–
HyperFlash (160 MHz, XIP)	OFF	OFF	ON	ON	OFF	OFF	OFF	–
HyperFlash (160 MHz, XIP)	OFF	OFF	ON	OFF	OFF	OFF	OFF	–
eMMC	OFF	ON	OFF	OFF	OFF	OFF	OFF	–

Boot Source Selection	SW4.1	SW4.2	SW4.3	SW4.4	SW4.5	SW4.6	SW4.7	SW48
QSPI on board (S25FS512S)	ON	ON	OFF	ON	OFF	OFF	OFF	OFF
Com-Express QSPI	ON	ON	OFF	ON	OFF	OFF	OFF	ON

This setting represents also the initial DIP switch setting on this board.

For more information, see the hardware and/or software manual for the board from the hardware vendor.

Memory layout

The memory devices for the R-Car V3H board and R-Car V3H Starter Kit are as follows:

For the R-Car V3H board:

SPI0 Flash memory - U6: S25FS512SDSMFV01 (64 MiB) from Cypress

64 MiB (or 512 MBits) of flash memory is available to store Boot Parameters, Loader, and U-Boot.

SPI0 Flash Memory- U7: S25FS128SAGMFV10 (16 MiB) from Cypress

16 MiB (or 128 MBits) of flash memory that's preflashed with QSPI MiniMonitor. Don't modify this memory.

SPI1 Flash memory (QSPI1) - U8: S25FS512SDSMFV01 (64 MiB) from Cypress

64 MiB (or 512 MBits) of flash that's only used with SPI0 and an 8-bit width.

For the R-Car V3H Starter Kit:

HyperFlash Flash Memory - IC5: S26KS512SDPBHI020 (64 MiB)

64 MiB (or 512 MBits) of flash memory of flash memory is available to store Boot Parameters, Loaders, and U-Boot.

QSPI Flash - IC4: S25FS512SAGBHI213 (64 MiB)

64 MiB (or 512 MBits) of flash memory of flash memory is available to store Boot Parameters, Loader, and U-Boot.

For more detailed information, see the Hardware Manual for the board from the manufacturer.

QSPI on the R-Car V3H board

Using QNX IPL

You'll need the memory layout details for the QSPI flash memory in the following table when you program the IPL and the IFS. You can use QNX IP for the R-Car V3H board only. For more information, see "*Program bootloaders for the R-Car V3H board*"):

Addresses for the R-Car V3H board

Files	Description	Program Top Address	QSPI Save Address
bootparam_sa0.srec	Boot Parameters	0xEB220000	0x000000
cr7-condor.srec	CR7 Loader	0xEB204000	0x040000
cert_header_sa3.srec	Loader (Certification)	0xEB220000	0x0C0000
bl2-condor.srec	CA53Loader	0xEB244000	0x140000
cert_header_sa6.srec	Loader (Certification)	0xEB220000	0x180000
bl31-condor.srec	ARM Trusted Firmware	0x44000000	0x1C0000
ipl-rcar_gen3-v3h.srec	QNX IPL	0x50000000	0x640000
ifs-rcar_v3h.srec	QNX IFS	0x40100000	0x800000

Using U-Boot

You'll need the memory layout details for the QSPI flash memory in the following table when you program the U-Boot. You can use U-Boot as the initial program bootloader for the R-Car V3H board and R-Car V3H Starter Kit:

Addresses for the R-Car V3H board

Files	Description	Program Top Address	QSPI Save Address
bootparam_sa0.srec	Boot Parameters	0xEB220000	0x000000
cr7-condor.srec	CR7 Loader	0xEB204000	0x040000
cert_header_sa3.srec	Loader (Certification)	0xEB220000	0x0C0000
bl2-condor.srec	CA53Loader	0xEB244000	0x140000
cert_header_sa6.srec	Loader (Certification)	0xEB220000	0x180000
bl31-condor.srec	ARM Trusted Firmware	0x44000000	0x1C0000
u-boot-elf-condor.srec	U-Boot	0x50000000	0x640000
ifs-rcar_v3.srec	QNX IFS	0x40100000	0x800000

Addresses for the R-Car V3H Starter Kit

Files	Description	Program Top Address	QSPI Save Address
bootparam_sa0.srec	Boot Parameters	0xEB220000	0x000000
cr7-v3hsk.srec	CR7 Loader	0xEB204000	0x040000
cert_header_sa3.srec	Loader (Certification)	0xEB220000	0x0C0000
bl2-v3hsk.srec	CA53Loader	0xEB244000	0x140000
cert_header_sa6.srec	Loader (Certification)	0xEB220000	0x180000
bl31-v3hsk.srec	ARM Trusted Firmware	0x44000000	0x1C0000
u-boot-elf-v3hsk.srec	U-Boot	0x50000000	0x640000

Flash loaders to the board

The loaders must be flashed to the board. MiniMonitor should be preflashed to your board; if not, you'll need to contact your hardware provider. Before you can flash the loaders to the R-Car V3H board or the R-Car V3H Starter Kit, you may need to contact Renesas Support to obtain the following files:

For the R-Car V3H board:

- `MiniMonitor`
- **u-boot-elf-condor.srec** (required if you want to boot using U-Boot)
- **bootparam_sa0.srec**
- **bl2-condor.srec**
- **bl31-condor.srec**
- **cert_header_sa3.srec**
- **cert_header_sa6.srec**
- **cr7-condor.srec**
- **u-boot-elf-condor.srec**

For the R-Car V3H Starter Kit:

- `MiniMonitor` [**AArch32_Gen3_V3H_Scif_MiniMon_V4.01.mot** (for ES1.0 revisions of the board) or **AArch32_Gen3_V3H_Scif_MiniMon_V5.04C.MOT** (for ES1.1 revisions of the board)]
- **u-boot-elf-v3hsk.srec** (required if you want to boot using U-Boot)
- **bootparam_sa0.srec**
- **bl2-v3hsk.srec**
- **bl31-v3hsk.srec**
- **cert_header_sa3.srec**
- **cert_header_sa6.srec**
- **cr7-v3hsk.srec** (for ES1.0 revisions of the board) or **cr7-v3h-1.1.srec** (for ES1.1 revisions of the board)

To flash the loaders to the R-Car V3H board or the R-Car V3H Starter Kit, you must:

1. Boot using QSPI Flash.
   - For the R-Car V3H board, set the DIP switches to boot using U7 QSPI Flash (S25FS128). MiniMonitor is preloaded on this QSPI Flash.
   - For the R-Car V3H Starter Kit, set the DIP switches to SCIF Download mode and send the MiniMonitor via serial (SCIF).

   For information about the DIP switches, see "*Configure the board switches for the R-Car V3H board*" and "*#unique_20*" for the settings to use.
2. Write the boot files to the QSPI flash. See the "*Program bootloaders for the R-Car V3H board*" section of this chapter.

Program bootloaders for the R-Car V3H board

You must flash the files using MiniMonitor to the board.

To complete these steps, you must have terminal emulator that allows you transmit files, such as Tera Term. To flash the files, you use MiniMonitor.

The following are the steps to flash the files to program the bootloaders to the board. The steps also include flashing either the QNX IPL or U-Boot, depending on what you want to use as your initial program loader to load the QNX Neutrino RTOS.

1. Set the board into QSPI mode as specified in the "*Configure the board switches for the R-Car V3H board*" for the QSPI Flash S25FS128, and then boot the board. Ensure that you follow the procedures in the hardware guide provided by the vendor to boot your board. Your board initially boots into the QSPI MiniMonitor showing this prompt in your console:

```
R-Car Gen3 Sample Loader develop_V3H_Vx.xx 20xx.xx.xx
For Condor.(RCarV3H)
DDR_Init          : boardcnf[12] Condor (V3H)
INITIAL SETTING   : Condor / R-Car V3H ESx.x
 CPU / BOOT MODE  : CR7-CPU0 / AArch64 CR7 Boot Mode
 DRAM             : LPDDR4 DDR3200
 DEVICE           : QSPI Flash(S25FS128) at 40MHz DMA
 BOOT             : Normal Boot
 Set CA53 Boot Address : 0xEB280000
 WAKE UP CA53
R-Car Gen3 MiniMonitor develop_V3H_Vx.xx 20xx.xx.xx
 Work Memory    : RT-SRAM
 Board Name     : Condor
 Product Code   : R-Car V3H ESx.x
>
```

2. Set the DIP switches so that you can program the HyperFlash memory (S26KS512S). For more information on the settings, see "*Configure the board switches for the R-Car V3H board*" HyperFlash settings.
3. (Optional) Type the `xcs` command to clear the HyperFlash memory, type `3` to choose the HyperFlash.

```
ALL ERASE SpiFlash or HyperFlash memory
Please select,FlashMemory.
   Please select,FlashMemory.
   1 : QspiFlash       (U5 : S25FS128S)
   2 : QspiFlash Board (CN3: S25FL512S)
   3 : HyperFlash      (U86: S26KS512S)
  Select (1-3)>
```

Press the `Y` key after you have changed and verified that the DIP switch settings are correct on your board:

- SW1 ALL OFF! Setting OK? (Push Y key)
- SW2 ALL OFF? Setting OK? (Push Y key)
- SW3 OFF! Setting OK? (Press Y key)
- SW31 OFF! Setting OK (Press Y key)
- SW10 ON-OFF-OFF-ON-ON-OFF-ON-ON! Setting OK? (Press Y)

4. At the prompt in your console connection, type the `xls2` command to clear the flash and when prompted, type `3` to choose the QSPI memory:

```
>xls2
===== Qspi/HyperFlash writing of Gen3 Board Command ========
Load Program to Spiflash
Please select,FlashMemory.
Please select,FlashMemory.
1 : QspiFlash       (U5 : S25FS128S)
2 : QspiFlash Board (CN3: S25FL512S)
3 : HyperFlash      (U86: S26KS512S)
Select (1-3)>3
```

After selecting the HyperFlash memory, the following messages appear in your console to indicate the DIP switches to set on your board. Press the `Y` key after you have changed and verified that the DIP switch settings are correct:

- SW1 ALL ON! Setting OK? (Push Y key)
- SW2 ALL ON! Setting OK? (Push Y key)
- SW3 ON! Setting OK? (Press Y key)
- SW31 ON! Setting OK (Press Y key)
- SW10 ON-OFF-OFF-ON-ON-ON-OFF-ON! Setting OK? (Press Y)

5. When you see the Please Input Program Top Address prompt appear, type the Program Top Address for the **bootparam_sa0.srec** file as `EB220000` and then press the **Enter** key.
6. Next, you should see a prompt for the Save Address. Type `000000` and then press the **Enter** key.

The value specified for Program Top Address (previous step) and Save Address (this step) is the same as in "*Memory layout*", but without the `0x` in front of the number.

7. When MiniMonitor prompts you to send the file, transmit the file from your host.

If you are using Tera Term, you must select **File** > **Send File** to transmit the file.

8. Repeat steps *4* to *7* for each of the files and its corresponding Program Top Address and Save Address as listed in the following table:

Filename	Value to enter for the Program Top Address	Value to enter for the Save Address
cr7-condor.srec	EB204000	040000
cert_header_sa3.srec	EB220000	0C0000
bl2-condor.srec	EB244000	140000
cert_header_sa6.srec	EB220000	180000
bl31-condor.srec	44000000	1C0000

9. You can either specify to use the U-Boot or QNX IPL as the bootloader. To do so, repeat steps *4* to *7* to enter the addresses specified in the table based on the bootloader you want to use:

   To use U-Boot:

Filename	Value to enter for the Program Top Address	Value to enter for the Save Address
u-boot-elf-condor.srec	50000000	640000

   To use QNX IPL:

Filename	Value to enter for the Program Top Address	Value to enter for the Save Address
ipl-rcar_gen3-v3h.srec	50000000	640000
ifs-rcar_v3h.srec	40100000	800000

10. Restart the board, and wait for U-Boot or QNX IPL to start. For more information, see "*Flash loaders to the board*."

Program bootloaders for the R-Car V3H Starter Kit

You must flash the files using MiniMonitor to the board.

To complete these steps, you must have terminal emulator that allows you transmit files, such as Tera Term. To flash the files, you use MiniMonitor.

The following are the steps to flash the files to program the bootloaders to the board. The steps also include flashing either the QNX IPL or U-Boot, depending on what you want to use as your initial program loader to load the QNX Neutrino RTOS.

1. Power off the board (SW1). Ensure that you follow the procedures in the hardware guide provided by the vendor.

2. Set the SW4 DIP switches all to OFF.
3. Power on the board (SW1) and your board boots into SCIF Mode. Here's a summary of the steps:
   - Type Ctrl-A
   - Select the “ascii” as your upload method.
   - Based on whether you are using an ES 1.0 or ES 1.1 revision of the board, send either **AArch32_Gen3_V3H_Scif_MiniMon_V4.01.mot** (for ES1.0 revisions of the board) or **AArch32_Gen3_V3H_Scif_MiniMon_V5.04C.MOT** file, respectively.

   After the upload completes, press any key. Once MiniMonitor starts, prompts are available in your console. For more information, see the procedures in the hardware guide provided by the vendor to send the MiniMonitor via serial SCIF or refer to *https://elinux.org/R-Car/Boards/V3HSK*.
4. Set the DIP switches SW4.1, SW4.2, and SW4.4 to ON so that you can program the QSPI on the board. For more information on the settings, see “*Configure the board switches for the R-Car V3H Starter Kit*” section in this chapter.
5. Type the `xcs` command to clear the memory and then type `3` to choose the QSPI Flash on the board and then type `y`, and then `y`.
6. At the prompt in your console connection, type the `xls2` command and follow the prompts. You should see the `Please Input Program Top Address` prompt appear where you would type the Program Top Address for the **bootparam_sa0.srec** file as `EB220000` and then press the **Enter** key.
7. Next, you should see a prompt for the Save Address. Type `000000` and then press the **Enter** key.

The value specified for Program Top Address (previous step) and `Save Address` (this step) is the same as in “*Memory layout*”, but without the `0x` in front of the number.

8. When MiniMonitor prompts you, select the **bootparam_sa0.srec** to send the file, transmit the file from your host to the target.

If you are using Tera Term, you must select **File** > **Send File** to transmit the file.

9. Repeat steps *6* to *8* for each of the files and its corresponding Program Top Address and Save Address as listed in the following table:

Filename	Value to enter for the Program Top Address	Value to enter for the Save Address
cr7-v3hsk.srec for ES1.0 revisions of the board or **cr7-v3h-1.1.srec** for ES1.1 revisions the board	`EB204000`	`040000`
cert_header_sa3.srec	`EB220000`	`0C0000`
bl2-v3hsk.srec	`EB244000`	`140000`
cert_header_sa6.srec	`EB220000`	`180000`

Filename	Value to enter for the Program Top Address	Value to enter for the Save Address
bl31-v3hsk.srec	44000000	1C0000
u-boot-elf-v3hsk.srec	50000000	640000

Boot board

You must first flash the loader onto the board before you can boot QNX Neutrino RTOS onto the board. The examples in this section are for the R-Car V3H board. Note that the steps listed in this section to boot using U-Boot are applicable for the R-Car V3H Starter Kit as well.

After you flash either the QNX IPL/QNX IFS or U-Boot to the QSPI memory on the board, turn on the power, your board should boot. Depending on the bootloader you choose, see the following sections for steps to complete the boot process:

- "*Boot using U-Boot*"
- "*Boot using QNX IPL*"

If your board doesn't boot, then may need to recover it using the steps described in "*Recovery boot with serial download mode*."

Boot using U-Boot

If you're using U-Boot, you must set up a TFTP server to load the QNX IFS file from your host machine. Here are some resources to help you with setting up a TFTP server based on the operating system running on your host machine:

- Linux: For information, see the following web resources: *http://askubuntu.com/questions/201505/how-do-i-install-and-run-a-tftp-server* and *https://help.ubuntu.com/community/TFTP*
- For macOS: macOS comes with TFTP. To enable the TFTP server, see *https://rick.cogley.info/post/run-a-tftp-server-on-mac-osx/*
- Windows: Install third-party software, such as Tftpd64 to run a TFTP server or turn on an FTP server under Internet Information Services (IIS) Manager. For more information, see *http://techzain.com/how-to-setup-tftp-server-tftpd64-tfptd32-windows/* and *https://www.windowscentral.com/how-set-and-manage-ftp-server-windows-10* .

After you set up your TFTP server and your board successfully boots with U-Boot, you should see the following prompt in your console or terminal connection:

```
NOTICE:  BL2: R-Car V3H Initial Program Loader(CA53) Rev.0.0.2


U-Boot 2015.04 (Sep 25 2017 - 12:17:21)

R-Car Gen3 Sample Loader develop_V3H_Vx.xx 20xx.xx.xx
For Condor.(RCarV3H)
 DDR_Init          : boardcnf[12] Condor (V3H)
 INITIAL SETTING   : Condor / R-Car V3H ESx.x
 CPU / BOOT MODE   : CR7-CPU0 / AArch64 CR7 Boot Mode
 DRAM              : LPDDR4 DDR3200
 DEVICE            : QSPI Flash(S25FS128) at 40MHz DMA
 BOOT              : Normal Boot
 Set CA53 Boot Address : 0xEB280000
 WAKE UP CA5
=>
```

Depending on the U-Boot version you're using, you may see the errors “Board Net Initialization Failed” and “No ethernet found” after you boot using U-Boot. This is because the MAC address wasn't found and you'll need to set it. To do so, set the *ethaddr* with a valid MAC address, which should be listed on a sticker on your board: For example:

```
=> setenv ethaddr 2E:09:0A:02:EA:37
```

Now that U-Boot is running, you must set up U-Boot to load the QNX IFS using an TFTP server (normally set up on your host machine). To do so, set the following environment variables:

- *serverip* with the IP address of your TFTP server is running (e.g., your host machine).
- *ipaddr* with the IP address of the board.

For example, set the environment variables and save them using the `saveenv` command:

```
=>setenv serverip 10.1.2.100
=>setenv ipaddr 10.1.2.101
=>saveenv
```

After you set the environment variables, you can load QNX IFS using these commands:

```
=> tftp 0x40100000 ifs-rcar_v3h.bin
=> go 0x40100000
```

You should see the following output on your console as U-Boot loads the QNX IFS file and boots the board to run QNX Neutrino:

```
NOTICE:  R-Car Gen3 Initial Program Loader(CR7) Rev.1.0.1
NOTICE:  PRR is R-Car V3H ES1.0
NOTICE:  CR7: DDR3200(rev.0.28rc05)NOTICE:  [COLD_BOOT]
NOTICE:  CR7: DRAM Split is OFF
NOTICE:  CR7: QoS is None
NOTICE:  Normal boot(CR7)
NOTICE:  CA5x Loader load address=0xeb244000 CA5x Loader image size=0x00020000
[    0.000140] NOTICE:  BL2: R-Car Gen3 Initial Program Loader(CA53) Rev.1.0.17
[    0.005702] NOTICE:  BL2: PRR is R-Car V3H Ver1.0
[    0.010365] NOTICE:  BL2: Board is unknown Rev---
[    0.015032] NOTICE:  BL2: Boot device is QSPI Flash(40MHz)
[    0.020471] NOTICE:  BL2: LCM state is unknown
[    0.024890] NOTICE:  BL2: v1.3(release):b15fefa
[    0.029358] NOTICE:  BL2: Built : 11:53:08, Mar  7 2018
[    0.034542] NOTICE:  BL2: Normal boot
[    0.038188] NOTICE:  BL2: dst=0xeb25e210 src=0x8180000 len=512(0x200)
[    0.044768] NOTICE:  BL2: dst=0x43f00000 src=0x8180400 len=6144(0x1800)
[    0.052407] NOTICE:  BL2: dst=0x44000000 src=0x81c0000 len=65536(0x10000)
[    0.071330] NOTICE:  BL2: dst=0x50000000 src=0x8640000 len=1048576(0x100000)


U-Boot 2015.04 (Mar 07 2018 - 11:53:14)

CPU: Renesas Electronics R8A7798 rev 1.0
Board: Condor
I2C:   ready
DRAM:  1.9 GiB
MMC:   sh-sdhi: 0
SF: Detected s25fs512s with page size 256 Bytes, erase size 256 KiB, total 512 MiB
*** Warning - bad CRC, using default environment

In:    serial
```

```
Out:   serial
Err:   serial
Net:   sh_eth
Hit any key to stop autoboot:  0
=>
=>
=>
=> setenv ipaddr 10.135.50.66; setenv serverip 10.135.51.44; tftp 0x40100000 ifs-rcar_v3h.bin; go 0x40100000
sh_eth connected to Micrel ksz9031
sh_eth Waiting for PHY auto negotiation to complete... done
sh_eth: 1000Base/Full
Using sh_eth device
TFTP from server 10.135.51.44; our IP address is 10.135.50.66
Filename 'ifs-rcar_v3h.bin'.
Load address: 0x40100000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #####################
         1017.6 KiB/s
done
Bytes transferred = 7430200 (716038 hex)
## Starting application at 0x40100000 ...
board_smp_num_cpu: 4 cores, max allowed 4
MMFLAGS=1
Welcome to QNX Neutrino mainline on the R-Car V3H Condor Board
Starting slogger and pipe servers...
Starting watchdog ...
Starting Serial driver...
Starting I2C driver ...
Starting SPI Flash driver...
Starting MMC memory flash driver ...
PathStarting Network driver...
=0 - rcar_gen3
 target=0 lun=0     Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
Starting SPI driver...
#
```

Boot using QNX IPL

After you flash the loader with the addresses for the QNX IPL and IFS, when the board boots, the following prompt appears:

```
QNX Neutrino Initial Program Loader for the R-Car V3H
             Condor
IPL compiled Feb 21 2018 14:09:57
Commands:
 Press 'S' for serial download, using the 'sendnto' utility
 Press 'E' for eMMC download, IFS filename MUST be 'QNX-IFS'
 Press 'Q' for QSPI flash Download
```

After you press `Q` to load the QNX IFS, you should see the following output on your console as the board boots to run QNX Neutrino:

```
NOTICE:  R-Car Gen3 Initial Program Loader(CR7) Rev.1.0.1
NOTICE:  PRR is R-Car V3H ES1.0
NOTICE:  CR7: DDR3200(rev.0.28rc05)NOTICE:  [COLD_BOOT]
NOTICE:  CR7: DRAM Split is OFF
NOTICE:  CR7: QoS is None
NOTICE:  Normal boot(CR7)
NOTICE:  CA5x Loader load address=0xeb244000 CA5x Loader image size=0x00020000
[    0.000140] NOTICE:  BL2: R-Car Gen3 Initial Program Loader(CA53) Rev.1.0.17
[    0.005702] NOTICE:  BL2: PRR is R-Car V3H Ver1.0
[    0.010365] NOTICE:  BL2: Board is unknown Rev---
[    0.015032] NOTICE:  BL2: Boot device is QSPI Flash(40MHz)
[    0.020471] NOTICE:  BL2: LCM state is unknown
[    0.024890] NOTICE:  BL2: v1.3(release):b15fefa
[    0.029358] NOTICE:  BL2: Built : 11:53:08, Mar  7 2018
[    0.034542] NOTICE:  BL2: Normal boot
[    0.038188] NOTICE:  BL2: dst=0xeb25e210 src=0x8180000 len=512(0x200)
[    0.044768] NOTICE:  BL2: dst=0x43f00000 src=0x8180400 len=6144(0x1800)
[    0.052407] NOTICE:  BL2: dst=0x44000000 src=0x81c0000 len=65536(0x10000)
[    0.071330] NOTICE:  BL2: dst=0x50000000 src=0x8640000 len=1048576(0x100000)

QNX Neutrino Initial Program Loader for the R-Car V3H-Condor
IPL compiled Jul 11 2018 11:36:55
Commands:
 Press 'S' for serial download, using the 'sendnto' utility
 Press 'E' for eMMC download, IFS filename MUST be 'QNX-IFS'
 Press 'Q' for QSPI flash Download
Flash download header...
Flash download image...
Scanning for image at @ 0x41000FB4
Found image           @ 0x41000FB4
Jumping to startup    @ 0x40101800

board_smp_num_cpu: 4 cores, max allowed 4
MMFLAGS=1
Welcome to QNX Neutrino mainline on the R-Car V3H Condor Board
Starting slogger and pipe servers...
Starting watchdog ...
Starting Serial driver...
Starting I2C driver ...
Starting SPI Flash driver...
Starting MMC memory flash driver ...
PathStarting Network driver...
=0 - rcar_gen3
```

```
 target=0 lun=0     Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
Starting SPI driver...
```

Chapter 4
Using the Screen Graphics Subsystem

The **rcar_v3h.build** that's provided with this BSP doesn't start the Screen Graphics Subsystem (Screen) driver by default in the image.Instead, you must use the image created from the **rcar_v3h-graphics.build** file to start Screen.

Screen is used to run and manage graphics shown on the display. For more information about using Screen, see the *Screen Developer's Guide*.

Graphics configuration

The graphics configuration file is located in ***$QNX_TARGET*/aarch64le/usr/lib/graphics/rcarv3h/graphics.conf**. The supported resolutions per display can be found in this file. Screen needs to be restarted for any change in graphics configuration to take effect as follows:

```
# slay screen
# screen
```

For more information about the Screen Graphics Subsystem, see the *Screen Developer's Guide* in the QNX Software Development Platform 7.0 documentation.

Configure the display for Screen

In the **graphics.conf** file, the *video-mode* setting must match the screen resolution and frame rate supported by the display that's connected to your board. Screen reads these parameters from the **graphics.conf** file to initialize the display drivers. For example, for a 1280 x 720, 60 Hz display, set the `video-mode` parameter in the **graphics.conf** file as shown here:

```
  ...
  ...
begin display 1
    video-mode = 1280 x 720 @ 60
  end display
...
...
```

For more information about configuring the `display` subsection in the **graphics.conf** file, see the "Configure display subsection" in the *Screen Developer's Guide*.

Display driver configuration library

The video display configuration libraries are located at *$QNX_TARGET*/aarch64le/usr/lib/graphics/rcar3vh/**libWFDrcar3.so** and ***$QNX_TARGET*/aarch64le/usr/lib/graphics/rcarv3h/libwfdcfg-rcarv3h-starterkit.so**. It has support for multiple displays. Each display has a list of supported resolutions as specified by the *video-mode* setting for each display in the graphics configuration file.

Run Screen applications

The BSP archive comes with built-in demo applications that show how to use Screen. Running these applications can help validate that Screen is started and has the necessary drivers loaded. Depending on the BSP, these are the basic demo applications, but additional demo applications can be added from your QNX SDP 7.0 installation. For more information, see the Utilities and Binaries chapter in the *Screen Graphics Subsystem Developer's Guide*.

`sw-vsync`

Shows `vsync` that uses software rendering, but doesn't use GLES. This application is useful to verify that your display works with basic Screen functionality.

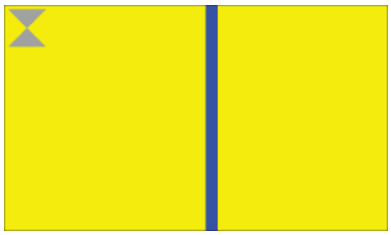

Figure 2: sw-vsync

To troubleshoot problems while running these applications, see the "Debugging" section in the *Screen Graphics Subsystem Developer's Guide*.

Create Screen applications

Before you create applications that use Screen, read the *Screen Graphics Subsystem Developer's Guide*.

Screen applications must link in **libscreen.so**. This library provides the interfaces for applications to the Screen API. Depending on the graphic requirements of your application, you may need to link in additional libraries. For example, you can link the **libimg** library, which is an interface to load and decode images. For more information, see the *Image Library Reference* guide in the QNX Software Development Platform 7.0 documentation.

For more information about linking libraries and depending on whether you use the IDE or command line to develop for development, see either the "QNX C/C++ Project properties" section in the Reference chapter of the *QNX Momentics IDE User's Guide*, or the "Compiling and Debugging" chapter of the *QNX Neutrino Programmer's Guide*.

Chapter 5
Build the BSP

You can use the QNX Momentics IDE or the command line 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. This prebuilt IFS image (**ifs-rcar_v3h.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 (command line)

You can use the command line 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`

Invokes all targets to generate IPL and IFS images.

`clean`

Removes all the ***.bin** and ***.srec** files.

`ifs`

Builds the following files for the R-Car V3H board and R-Car V3H Starter Kit:

- **ifs-rcar_v3h.bin**
- **ifs-rcar_v3h.srec**
- **ifs-racar_v3h-graphics.bin**
- **ifs-rcar_v3h-graphics.srec**

Graphics isn't available on the IFS.

`ipl`

Builds the following files for the R-Car V3H board:

- **ipl-rcar_gen3-v3h_condor.srec**
- **ipl-rcar_gen3-v3h_condor.bin**

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

Build from the command line

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 root directory (*BSP_ROOT_DIR*), 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-rcar_v3h.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 IFS image. If you use the `mkifs` utility directly, 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, 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 IPL and IFS images.

1. If you haven't done so, launch the IDE, 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-renesas-r-car-v3h**) 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-rcar_v3h.bin**, it maps to the ifs-rcar_v3h.bin target in the Makefile, you would type `ifs-rcar_v3h.bin` as the target. For more information on the targets available, see the "*Build the BSP (command line)*" section.

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

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

Chapter 6
Driver Commands

The tables below provide a summary of driver commands for the R-Car V3H board.

Some of the drivers are commented out in the buildfile provided in the **images** directory of this BSP. To use some of the 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 start the drivers are completely customizable and are often specific to the requirements of your system. For example, if you require networking as soon as possible, you must start the networking driver earlier than other drivers. It's important to recognize the order in which the drivers start may impact the boot time.

Some drivers depend on other drivers so it's sometimes necessary to start them in a specific order because of these dependencies.

- 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 and the order they appear in the provided buildfile:

- *Startup*
- *Watchdog*
- *Serial*
- *Inter-integrated Circuit (I2C)*
- *SPI Flash*
- *eMMC*
- *Network*
- *SPI*
- *PCI Server*
- *Graphics*

The following driver is available, but not called in the buildfile:

- *Thermal driver*

eMMC

Device	eMMC flash driver
Command	`devb-sdmmc-rcar_gen3 blk cache=64m cam bounce=128k mem name=below4G sdio idx=0,emmc disk name=mmc@0`
Required binaries	`devb-sdmmc-rcar_gen3`
Required libraries	**libcam.so**, **cam-disk.so**, **io-blk.so**, **fs-qnx6.so**, **fs-dos.so**
Source location	***$BSP_ROOT_DIR*/src/hardware/devb/sdmmc**

In addition to the "devb-sdmmc-*" options described in the QNX Neutrino *Utilities Reference* guide found in the QNX SDP 7.0 documentation, these are the board-specific options that are available for this board, which aren't documented in the *Utilities Reference* guide in the QNX SDP documentation:

bs=*board-specific option=board-specific option value*:[*board-specific option=board-specific option value* ...]

These are the board-specific options supported, which are separated by colons:

nowp

Disable Write-protect capability.

pwr

Power base register, size, offset, where each value is separated by a forward dash (/).

pwr_vdd

Use the VDD Pin.

pwr_vdd_base

Power VDD GPIO register. This option is required when the PWR_VDD pin is on a different GPIO bank other than PWR.

pwr_if

Use the VDDQVA Pin.

pfc

Specify the I/O base register, size, offset, PMMR where each value is separated by a forward dash (/).

pfc_mask=address

Specify the I/O pin mask.

Examples:

eMMC Flash:

```
devb-sdmmc-rcar_gen3 blk cache=64m cam
 bounce=128k mem name=below4G sdio idx=2,
 emmc disk name=mmc@0
```

You can choose to use more aggressive eMMC read/write performance, with this command:

```
devb-sdmmc-rcar_gen3 blk maxio=256,cache=1m,ra=256k:512k,rapolicy=aggressive,commit=none
                                    cam cache,async mem name=below4G sdio
                                    idx=2,emmc,~ac12 sdmmc cache=on
                                    disk name=mmc@0
```

For more information the risks when use aggressive eMMC read/write performance, see *http://www.qnx.com/support/knowledgebase.html?id=501a0000000Mpbr* on the QNX website.

Graphics

Device	GRAPHICS
Command	`screen -c /usr/lib/graphics/rcarv3h/graphics.conf`
Required binaries	`screen`, `sw-vsync`
Required libraries	Refer to the *Screen Developer's Guide* in the QNX SDP 7.0 documentation.
Source location	**Prebuilt only**

Before starting Screen, you must set the *LD_LIBRARY_PATH* environment variable as follows:

```
LD_LIBRARY_PATH=/usr/lib:/lib:/lib/dll:$LD_LIBRARY_PATH
```

Inter-integrated Circuit (I2C)

Device	I2C
Command (I2C0)	`i2c-rcar-A -I0 -v --u0`
Command (I2C1)	`i2c-rcar-A -I1 -v --u1`
Command (I2C3)	`i2c-rcar-A -I3 -v --u3`
Required binaries	`i2c-rcar-A`
Required libraries	N/A
Source location	***$BSP_ROOT_DIR*/src/hardware/i2c**

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

Network

Device	Ethernet
Command (generates a MAC address)	`sh -c "io-pkt-v6-hc -drgbe mac=`genmac-random -m`, -ptcpip pkt_typed_mem=below4G"` `dhclient -m -lf /dev/shmem/dhclient.leases -pf /dev/shmem/dhclient.pid -nw rg0`
Required binaries	`io-pkt-v6-hc`, `ifconfig`, `dhclient`, `genmac-random`
Required libraries	**devnp-rgbe.so**
Source location	***$BSP_ROOT_DIR*/src/hardware/devnp/rbge**

By default, a random MAC address is assigned each time the board boots because the MAC address is assigned using `genmac-random` via the `mac` option for the `devnp-rgbe` DLL.

You can set a static MAC address using the `mac` option, which is recommended in environments where you run more than one target with this BSP or on networks that assign IP addresses dynamically via DHCP. The static MAC address that you use must be a valid MAC address. The static MAC address that you use must be a valid MAC address. We recommend that you use the MAC address listed on the sticker on your board.

If you need to run a DHCP client, you can run it using the `dhclient` command. For example:

```
dhclient -m -lf /dev/shmem/dhclient.leases -pf
/dev/shmem/dhclient.pid -nw rg0
```

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

PCI

Device	PCI
Command	pci-server --bus-scan-limit=1 -c
Required binaries	pci-server, pci-tool
Required libraries	**libpci.so**, **pci_bkwd_compat.so**, **pci_strings.so**, **pci_cap-0x10.so**, **/pci_cap-0x11.so**, **/pci_cap-0x05.so**, **/pci_cap-0x01.so**, **pci_hw-rcar-v3h.so**, **pci_debug2.so**, **pci_slog2.so**
Source location	Prebuilt only

Serial

Device	Serial drivers
Command	`devc-serscif -e -F -b115200 -t 14 scif0`
Required binaries	`devc-serscif`
Required libraries	**libsocket.so**
Source location	***$BSP_ROOT_DIR*/src/hardware/devc/serscif**

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

SPI Flash

Device	SPI
Command	`devf-rcar_qspi-gen3`
Required binaries	`devf-rcar_qspi-gen3`
Required libraries	N/A
Source location	***$BSP_ROOT_DIR*/src/hardware/flash/boards/rcar_qspi**

For more information about this driver, see "*devf-rcar_qspi-gen3*" in the "BSP-specific Drivers and Utilities" chapter of this guide.

Startup

Device	Startup
Command	`startup-rcar_v3h -P4 -W -Dscif0`
Required binaries	`startup-rcar_v3h`
Required libraries	**libc.so**
Source location	***$BSP_ROOT_DIR*/src/hardware/startup/boards/rcar_gen3**

In additional to the common options available for the `startup-*` command as described in the *Utilities Reference* in the SDP 7.0 documentation, this driver supports these options:

`-d`

Enable automatic RAM size detection.

-L ***address*****,** ***size*****[,** ***name*****]**

Remove the *size* of memory from the system started at the specified *address*. If you don't specify the *size* option, the typed memory isn't removed. The typed memory *name* is optional and if you don't specify it, the default `rcar_reserved` is used.

-p ***CPUs_on_boot_cluster***

Set the maximum number (*CPUs_on_boot_cluster*) of CPUs to run in the boot cluster. This option can be used on multi-cluster SoCs. You can use this option with the `-P` option, which specifies the number of CPUs to use. For example, if you specify the options `-P6 -p3`, this means to use a total of six CPUs and three would be for the boot cluster.

-W

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

SPI

Device	SPI driver
Command	`spi-master -u 0 -d /proc/boot/spi-rcar.so channel=0,blksize=64`
Required binaries	**spi-master**
Required libraries	**spi-rcar.so**
Source location	***$BSP_ROOT_DIR*/src/hardware/spi/**

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

Thermal Driver

The driver used for thermal management on the board.

Device	Thermal driver
Command	`rcar-thermal -I1 -t20 -v` (use sensor THS1 and an interrupt is triggered at 20 degrees Celsius) `rcar-thermal -I2 -t-10,20 -v`(use sensor THS2 and an interrupt is triggered at -10 or 20 degrees Celsius)
Required binaries	`rcar-thermal`
Required libraries	N/A
Source location	***$BSP_ROOT_DIR*/src/hardware/support/rcar-thermal**

For more information about this driver, see "*rcar-thermal*" in the "*BSP-specific Drivers and Utilities*" chapter.

Watchdog

To enable the watchdog:

1. Modify the buildfile so that `startup` launches with the `-W` option:
   ```
   startup-rcar_v3h -P4 -W -Dscif0
   ```
2. Launch the watchdog timer utility early on in the boot script:
   ```
   wdtkick -W 0x0:0x5A5AFF00
   ```

Device	WATCHDOG
Command	`wdtkick -W 0x0:0x5A5AFF00`
Required binaries	`wdtkick`
Required libraries	N/A
Source location	***$BSP_ROOT_DIR*/src/hardware/support/wdtkick**

Chapter 7
BSP-specific Drivers and Utilities

This chapter describes drivers and utilities are specific for this BSP. It also describes drivers and utilities drivers and utilities that aren't available in the QNX SDP documentation at the time this guide was published.

- *devc-serscif*
- *devf-rcar_qspi-gen3*
- *devnp-rgbe.so*
- *i2c-rcar-A*
- *rcar-thermal*
- *spi-master*

devc-serscif

Serial driver for asynchronous SCIF (Serial Communication Interface) and HSCIF (High Speed Serial Interface)

Syntax:

```
devc-serscif [-B address]
             [-c clock[/div]]
             [-C size]
             [-D size]
             [-d
             [-e] [-E]
             [-f] [-F]
             [-h][-H]
             [-i irq_num]
             [-I size]
             [-m interrupt_mode]
             [-O options=value
             [-s] [-S]
             [-t level]
             [-T level]
             [-r level]
             [-u serial_unit_num]
             [-v[v]...]
             [-x]
[scif_num]
```

where *scif_num* can be `scif1` or `scif2`

Runs on:

QNX Neutrino

Options:

In addition to the `devc_ser*` options (see the QNX Neutrino *Utilities Reference*), this driver supports the following options:

`-B` *address*

The base address.

`-c` *clk[/div]*

Set the input clock rate and an optional divisor.

`-d`

Enable DMA on the board. The default is disabled.

-D ***Rxch1*[/*Txch2*]**

Enable DMA with a specific channel. The Rx channel is specified first and optionally, the Tx channels when it is delimited by a slash ("/"). If a Tx channel is not specified, the Tx channel is presumed to be one higher than the Rx channel. DMA is disabled by default.

-h

Enable RS-232 use of RTS and CTS lines (default)

-H

Permanently disable RS-232 use of RTS and CTS lines (for use as GPIOs)

-i ***irq_num***

The IRQ.

-m ***interrupt_mode***

Set interrupt mode, which can be 0 or event mode or 1 for ISR. The default is 1 (ISR).

-r ***level***

Set RTS trigger level. The SCIF default is 15.

-t ***level***

Set receive FIFO trigger level. The SCIF default is 14.

-T ***level***

Set transmit FIFO data trigger level. The SCIF default is 0.

-u ***serial_unit_num***

Set serial unit number. The default is 1.

-x

Use an external clock.

devf-rcar_qspi-gen3

Driver for the SPI Flash memory

Syntax:

```
devf-rcar_qspi-gen3
[-aclrvV] [-b priority] [-f verifylevel] [-i index]
[-m mountover] [-p backgroundpercent[,superlimit]]
[-s baseaddress[,windowsize[,arrayoffset[,arraysize[,unitsize[,buswidth[,interleave]]]]]]]
[-t threads] [-u updatelevel] [-w buffersize]
[bs=[board-specific_option][,board-specific_option]...]
```

Runs on:

QNX Neutrino

Options:

-A

When registering the path names for the partitions with *resmgr_attach()*, use the _RESMGR_FLAG_AFTER flag to force the path to be resolved after others with the same pathname at the same mountpoint.

-a

Don't automount or enumerate filesystem partitions present on the media. If you specify both the `-a` and `-R` options, the driver ignores the `-R` option.

-b *priority*

Enable background reclaim at the specified *priority*. By default, background reclamation is disabled.

-D

Enable automatic detection of error-correcting code (ECC) mode. If you specify this option, you don't need to specify `-x`.

Don't mix ECC-enabled partitions and ECC-disabled partitions; the driver doesn't support this.

-d *log_method*

Control the logging from the flash driver. The possible values for *log_method* are:

- 0 — log to *stdout* (the default)
- 1 — log to `slogger2`
- 2 — log to both *stdout* and `slogger2`

-E

Don't daemonize. If the driver detects a corrupt filesystem, the exit status is that filesystem's partition number plus 1.

-e ***arg***

Only enumerate the flash partitions, instead of doing a full scan and `mount`:

- If *arg* is an integer, the flash driver automounts all partitions with a partition number less than or equal to *arg*.

  For example, assume we have a flash layout as follows:

  - **/dev/fs0p0** — raw
  - **/dev/fs0p1** — formatted
  - **/dev/fs0p2** — formatted
  - **/dev/fs0p3** — formatted

  If you start the driver with `-e 1`, the driver creates all the raw entries in **/dev**, but mounts only **/dev/fs0p1** (**/dev/fs0p0** is raw, so it's never mounted, regardless of the `-e` option).

- If *arg* is a string, the driver interprets it as a colon-separated list of exact paths to mount, if found.

-f ***verifylevel***

Enable flash verification (default=0, 0=none, write=1, erase=2, all=3).

-i ***arrayindex*[,*partindex*]**

Starting socket index and first partition index; 0 *index* 15. The default is 0,0. Use this to give multiple drivers unique IDs. The `-i` option is just a suggestion for the resource database manager; the selected indexes can be larger.

-L ***limit***

The number of retries to make if the physical flash erase function for a unit fails. The default is 0.

-l

List the available flash databases and then exit.

-m ***mountover***

Override the mountpoints assigned to the file system that are formatted with an empty (i.e., `flashctl -p/dev/fs0p0 -f -n ""`) mountpoint. The *mountover* argument can include two `%X` format specifiers (like those for *printf()*) that are replaced by the socket index and the partition index.

The `-m` option doesn't override a mountpoint specified with `mkefs`.

-O

Use a directory lock for I/O operations.

-o *file_max*

Set the maximum number of files to cache. The default is 64.

-P *lock_mode*

Set the protection mode for Spansion-compatible devices:

- 0 — no lock (the default)
- 1 — persistent lock mode
- 2 — dynamic lock mode

-p *backgroundpercent*[,*superlimit*]

Set the background-reclaim percentage trigger (stale space over free space) and, optionally, the superseded extent limit before reclaim. The default is 100,16.

-R

Mount any automount filesystems as read-only. This option doesn't affect raw partition mounts, and it has an effect only at startup and initialization. Any subsequent mounting (with either `flashctl` or `mount`) ignores the `-R` option. If you also specify the `-a` option, the driver ignores the `-R` option.

-r

Enable fault recovery for dirty extents, dangling extents (dirty headers or damaged pointers), and partial reclaims.

You should always specify the `-r` option unless you're trying to debug an issue concerning flash corruption.

If you don't specify `-r`, and a power failure occurs, the following can happen:

- You can waste space. If an erasure was happening when the power was cut off, there will be some "dangling" extents (i.e., marked for deletion, but not actually deleted). If you specify the `-vv` option, the driver prints `dangle` for every dangling extent found. These extents will continue to occupy space forever, until they're deleted. Using the `-r` option will cause them to be reclaimed.
- The system may be marked as read-only. If the driver detects an error in the structure of the filesystem, and you haven't specified the `-r` option, the driver marks the partition as read-only, so that it can't be further damaged.
- If a reclaim operation is interrupted by a power loss, the spare block may be unusable. In this case, if you specify the `-vv` option, the driver prints `partial` to the console. The partition is still read-write, but reclaims are turned off; if you continue to overwrite files, you'll eventually fill the filesystem with stale data.

-S ***sector_erase_latency***

Set the simulated sector erase latency in milliseconds (max = 10000).

-s ***base*[,*wsize*[,*aoffset*[,*asize*[,*usize*[,*bwidth*[,*ileave*]]]]]]**

Set socket options, normally the base physical address, window size, array offset, array size, unit size, bus width, and interleave. The format is left flexible for socket services with customized drivers. This option must be specified.

The arguments are:

base

Physical base address of the flash part. This value is board-specific.

wsize

Size of the physically contiguous flash part.

aoffset

For SRAM, the offset from the base address to the start of the flash array.

asize

For SRAM, the size of the flash array. The default is equal to *wsize*.

usize

The size of a physical erase sector. For SRAM, this number can be any power of two. 64 KB should be the minimum, for performance reasons.

bwidth

The total width of the data bus, as seen from the microprocessor's perspective. This is the width of one flash chip multiplied by the interleave. The value must be a power of 2 (1, 2, 4, or 8).

ileave

The number of flash chips arranged on the data bus. Two 16-bit wide chips used as the upper and lower halves of a 32-bit databus give an interleave of 2. This number must be a power of 2 (1, 2, 4, or 8).

You can specify the base physical address, sizes, and offset in octal (`1000`), hexadecimal (`0x200`), or decimal (`512`). The sizes must be a power of two, and you can specify them with any of the following suffixes:

- (nothing) — bytes
- `k` — kilobytes
- `m` — megabytes

-T ***max_erase_diff***

Set the threshold value (maximum erase count – minimum erase count in a partition) to trigger wear-leveling. The default value is two times the sector number in the partition.

Typically, for very large partitions containing more than 1000 sectors, you should use this option to specify a threshold (for example, 1000) to make the sector erasure counts more evenly distributed across the entire partition.

-t *hi_water*[,*lo_water*[,*max*]]

Set the high water, low water, and maximum attributes of the thread pool; the increment (i.e., the number of threads created at one time) is 1. The default is 4,2,100. The values must be related as follows:

- 0 < *hi_water* < *max*
- 0 *lo_water* *hi_water*
- *hi_water* < *max* 100

-u *update*

Specify the update level for timestamps. POSIX specifies that timestamps be kept when you access, create, or modify a file. FFSv3 is documented as not supporting the access timestamp, in order to reduce wear on the hardware.

The values for *update* are:

- 0 — don't update the modification time for files (the default).
- 1 — update the modification time for files according to the POSIX rules.
- 2 — update the modification time for files, as well as for the parent directory.

The `-u2` option is very, very expensive and will cause many reclaims because the time updates have to flow right up to the root directory, so one file update may cause many directory updates.

-V

Display filesystem and MTD version information, and then exit.

-v

Be verbose; specify additional `v` characters for more verbosity.

-W *num*

Use the workaround identifed by *num*. The workarounds available (if any) depend on the board.

-w *buffersize*

Write (append) buffer size in bytes. The default *buffersize* is 512. Using a larger write buffer prevents the creation of very small extents, reducing overhead. If *buffersize* is 0, appending is disabled.

-x *type*

Enable software ECC mode. Specify *type* as 1 for 64-byte alignment ECC, or 2 for 32-byte.

Don't mix ECC-enabled partitions and ECC-disabled partitions; the driver doesn't support this.

bs=*board_specific_option*

This driver has the following board-specific options (*board_specific_option*), which are specified using these parameters:

cs=chip_selected

The chip to select.

drate=rate

Specify the data rate as a numeric value in Mbps. The default is 160000000.

mode=mode

The CPU mode, which can be set to `Single`, `Dual`, or `Quad`. The default is `Quad`.

clock=clockrate

Specify the clock rate in Hertz.

base=controller_base

The base address for the controller. No default value is provided.

size=controller_reg_size

The size of controller register, specified as a hexadecimal or integer value.

buffer=base_address

The base address for the buffer. No default value is provided.

buffersize=buffer_size

The size of the buffer, specified as a hexadecimal or integer value.

ver=dma_ver

Version of the DMA Library to use. Set the value representing the Renesas board you are using as follows (not case-sensitive):

- `H2`
- `M2`
- `E2`
- `V2`
- `H3`
- `M3`
- `D3`
- `E3`

- V3M
- V3H
- W2H

Description:

This driver provides Flash filesystem support for Renesas boards.

Example:

```
devf-rcar_qspi-gen3 -s0x08000000,0x04000000 bs=base=0xee200000,size=0xac,buffer=0xee208000,
                    buffersize=0xff,ver=v3h
```

devnp-rgbe.so

Driver for the Renesas Gigabit Ethernet interface using the KSZ9031RNXVB PHY from Microchip Technology

Syntax:

```
io-pkt-v6-hc -d rgbe [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 as Megabits/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.

iorange=0x*io_base_addr*

I/O base address.

irq=*irq_num*

IRQ of the interface.

mac=*mac_interface_addr*

Interface address of the controller.

verbose[=*level*]

Set 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 following are board-specific options.

event

Use InterruptAttachEvent mode when this option is specified; otherwise, the default is to use ISR.

flow=0|1|2|3

Force flow control, which can be one of the following values:

- 0–off
- 1–bi-directional pause
- 2–Rx pause frames
- 3–Tx pause frames

Examples:

Start networking with the devnp-rgbe driver:

```
io-pkt-v6-hc -d rgbe
ifconfig rgbe0 192.0.2.1
```

i2c-rcar-A

Driver for inter-integrated circuits on the Renesas boards

Syntax:

```
i2c-rcar-A [rcar_options [rcar_option]...]
           [generic_option [generic option]...]
```

Runs on:

QNX Neutrino

Options:

There are R-Car-specific (single-dash) and generic (double-dash) options available.

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

```
i2c-rcar-A -v --u0
```

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

```
i2c-rcar-A --u0 -v
```

R-Car options:

These are R-Car-specific drivers.

-i ***irq***

The IRQ. There is no default IRQ.

-p ***address***

The port address. There is no default address.

-v ***level***

Set the verbosity level. The default is 0 (zero).

Generic options:

These are the generic options for the driver.

--b ***bus_speed***

The default bus speed. Default is 100000.

--m ***max_msg_len***

The minimum length of the resource manager (`resmgr`) receive buffer. The buffer should be at least the size of the specified by *max_msg_len*. The default is 128 bytes.

--n ***nparts***

the minimum number of iov_t's in the resource manager (resmgr) context. The default is 2.

--u ***unit***

The unit number, which is the number to append to device name prefix (e.g., **/dev/i2c**). The default to use is 0 (zero).

Example:

Specify a port of address 0xe6510000 with an IRQ of 318 to the second unit:

```
i2c-rcar-A -p0xe6510000 -i318 -v --u2
```

rcar-thermal

Driver for thermal management on the board

Syntax:

```
rcar-thermal [options]
```

Runs on:

QNX Neutrino

Options:

-b ***register***

Specify the thermal sensor base register. There is no default value.

-i ***irq_num*[,*irq_num*...]**

Specify the list of IRQ numbers delimited by commas. (e.g., 99, 100, 101).

-I ***thermal_sensor***

Specify the thermal sensor interface to use (THS1, THS2, or THS3), using a numeric value.

-t ***threshold***

Specify the temperature thresholds (Celsius) that triggers an interrupt to occur. You can specify a single temperature or a low and high, delimited by a comma.

-V ***soc_version***

The Renesas SoC version on the board, where *soc_version* is one of the following case-sensitive, string values representing the SoC version on the board:

- `d3` — Renesas R-Car D3 board
- `e3` — Renesas R-Car E3 board
- `h3es1` — Renesas R-Car H3 board (revision WS1.1)
- `h3` — Renesas R-Car H3 board (revision WS2.0 and later)
- `m3w` — Renesas R-Car M3-W boards (revision WS2.0 and later)
- `m3n` — Renesas R-Car M3-N boards (revision WS2.0 and later)
- `v3m`— Renesas R-Car V3M
- `v3h` — Renesas R-Car V3H board

This option must be used for Hypervisor guests since HWI info isn't passed from a guest system. We recommend that you pass the SoC version information as part of the HWI information as shown in this BSP (***$BSP_ROOT_DIR*/src/hardware/startup/rcar_gen3-/rcar_*[SOC_VERSION]*/init_hwinfo.c**,

where *[SOC_VERSION]* represents your SoC, such as `v3h` for the Renesas R-Car V3H board).

-v [-v]

Increase output verbosity. The `-v` option is cumulative; each additional `v` (up to eight) adds an increased level of verbosity.

Examples:

Start THS1 with an interrupt that's triggered at 20 degrees Celsius for the Renesas R-Car D3 SoC:

```
rcar-thermal -I1 -t20 -V d3
```

Start THS2 with an interrupt that's trigger at -10 and 30 degrees Celsius:

```
rcar-thermal -I2 -t-10,30
```

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 0x40057000.

blksize=*size*

The transceiver's block size.

channel=*channel_num*

Defines the connected channel number as defined in hardware info. The default is 0.

clock=*num*

Enhanced SPI clock. The default is 133000000 Hertz. (26000000)

irq=*num*

The IRQ of the interface. The default is 93.

loopback=*enable*

Set the internal loopback for testing. The default is 0 (loopback disabled). Set to 1 to enable loopback testing.

verbose=*num*

Verbosity of logging. Default is 0 for no logs.

waitstate=*num*

Set the number of wait states between transfers. The default is 0.

Example:

Start SPI driver for SPI0 with a base address and IRQ:

```
# spi-master -u 0 -d /proc/boot/spi-rcar.so channel=0,blksize=64
```