

BSP User's Guide

Renesas R-Car H3/M3

©2016–2020, QNX Software Systems Limited, a subsidiary of BlackBerry Limited. All rights reserved.

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/>

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: March 28, 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.....	 15
Download and set up the BSP.....	16
Set up the hardware.....	19
Configure the R-Car H3 board.....	20
Configure the R-Car H3 Starter board.....	33
Configure the R-Car M3-W board.....	41
Configure the R-Car M3-W Starter board.....	51
Configure the R-Car M3-N board.....	56
Transfer image and boot the board.....	67
Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash.....	69
Boot the QNX IFS from an SD card, eMMC, HyperFlash, or serial using QNX IPL.....	72
Build U-Boot and Renesas Loader.....	77
Programming Renesas Loader and U-Boot to HyperFlash	80
Configure the board to boot using U-Boot and from an SD card.....	83
Configure the board to boot using U-Boot and from eMMC memory.....	85
Configure the board to boot using U-Boot and TFTP.....	88
 Chapter 4: Using the Screen Graphics Subsystem	 91
 Chapter 5: Build the BSP.....	 97
Build the BSP (command line).....	98
Build the BSP (IDE).....	100
 Chapter 6: Driver Commands (R-Car H3).....	 103
 Chapter 7: Driver Commands (R-Car H3 Starter).....	 117
 Chapter 8: Driver Commands (R-Car M3-W).....	 127
 Chapter 9: Driver Commands (R-Car M3-W Starter).....	 139

Chapter 10: Driver Commands (R-Car M3-N).....149**Chapter 11: BSP-specific Drivers and Utilities.....159**

deva-ctrl-rcar-cs2000.so.....	160
deva-mixer-ak4613.so.....	166
devb-rcarsata.....	168
devc-serscif.....	172
devf-rcar_qspi-gen3.....	174
devnp-ravb.so.....	181
devu-dcd-usbncm-hsusb-rcar.so.....	183
devu-dcd-usbser-hsusb-rcar.so.....	185
devu-dcd-usbmass-hsusb-rcar.so.....	187
i2c-rcar-A.....	189
i2c-rcar-B.....	191
rcar-thermal.....	193

About This Guide

This guide contains installation and start-up instructions for the Board Support Package (BSP) for the following boards:

- Renesas R-Car H3 (Salvator-X) (referred to as the R-Car H3)
- Renesas R-Car H3 Starter Kit Premier (referred to as R-Car H3 Starter)
- Renesas R-Car M3-W (Salvator-X) (referred to as the R-Car M3-W)
- Renesas R-Car M3-W Starter Kit Pro (referred to as the R-Car M3-W Starter)
- Renesas R-Car M3-N (Salvator-X) (referred to as the R-Car M3-N)

Only 64-bit support is provided by QNX Software Systems. This BSP doesn't provide 32-bit support for any of the boards.

This guide provides the following information:

- an overview of the BSP
- what's new with working with BSPs for this release
- structure and contents of a BSP
- how to build the BSP using the command line or the QNX Momentics IDE
- how to prepare a bootable SD card
- how to transfer the image to boot the card
- 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 (R-Car H3) Driver Commands (R-Car H3 Starter) Driver Commands (R-Car M3-W) Driver Commands (R-Car M3-W Starter) Driver Commands (R-Car M3-N)
About BSP-specific drivers and utilities	BSP-specific Drivers and Utilities

Typographical conventions

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

The following table summarizes our conventions:

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

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

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

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



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 general information about working with BSPs from QNX Software Systems that's common to all BSPs and is 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
- 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.

Chapter 2

About This BSP

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

This BSP contains the components listed in the table below. For more detailed information about binaries and libraries, see one of the following chapters based on the board you're using:

- [Driver Commands \(R-Car H3\)](#)
- [Driver Commands \(R-Car H3 Starter\)](#)
- [Driver Commands \(R-Car M3-W\)](#)
- [Driver Commands \(R-Car M3-W Starter\)](#)
- [Driver Commands \(R-Car M3-N\)](#)

Component	Format	Comments
Startup	Source	
Audio driver	Source	Includes new codec and DMA library
DMA libraries	Source	
eMMC driver	Source	
I2C driver	Source	
Network driver	Source	
QNX IPL	Source	
SATA driver	Source	
Screen Graphics Subsystem	Binary	Graphics
SDHI memory card driver	Source/Binary	For Secure Digital (SD) cards
Serial driver	Source	SCIF and HSCIF
SMMU Manager	Binary	Makes use of the DMA containment and memory management support available for the ARM architecture. The libraries and header files are part of QNX SDP 7.0 (QNX Neutrino 7.0.4).
SPI Flash driver	Source/Binary	
Thermal driver	Binary/Source	

Component	Format	Comments
USB device and host drivers	Binary	USB (host mode) drivers are provided in binary format as part of QNX SDP 7.0. The USB (device mode) drivers are provided as part of this BSP.
Watchdog timer	Source	

Supported OSs

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

This BSP supports the QNX Neutrino RTOS 7.0.

Interrupt tables for supported boards

The interrupt vector table can be found in the buildfiles located in the ***\$BSP_ROOT_DIR/images/*** directory. Based on the board you are using, open the corresponding ***rcar_X.build*** file where *X* represents the board. For example, ***rcar_h3.build*** is the buildfile for the R-Car H3 and ***rcar_h3ulcb.build*** is the buildfile for the R-Car H3 Starter. 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 for BSP_name** window, on the **General Information** pane, click the link beside **Release Notes**.

You should be redirected to the release notes on the QNX website. To see the notes, you must be logged in with your myQNX account.

Required software

To work with the boards supported by this BSP, in addition to the QNX Neutrino RTOS and QNX SDP 7.0, you need the following software:

- virtual COM port drivers, which are available from Silicon Labs at www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
- a terminal emulation program Serial Terminal View, Hyperterminal, Putty, Qtalk, Tera Term, etc. You can also use the QNX Momentics IDE to start a telnet session to the target.

Required cables

To connect to your board, you need:

- a USB to microUSB 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 in the QNX Software Center.

Chapter 3

Installation Notes

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

Here's an overview of the process to install this BSP and run it on your board:

1. Download the BSP, setup 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 the DIP switches on your board to further configure it. For more information about how to connect to your board and to set the DIP switches, see appropriate section for your board in the [“Set up the hardware”](#) section . If you're interested in understanding how memory is mapped on boards supported for this BSP, see [“Memory layout.”](#)
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 to the board. You can transfer the IFS file to RAM from a DOS FAT-formatted SD card or from a TFTP server.

If you want to boot the image using HyperFlash, you can either flash QNX IPL (provided), or U-Boot and the Bootloader to the board. After your images are ready, you can configure your board to boot from HyperFlash, from SD card, serial, eMMC, or using a TFTP server depending on whether you are using QNX IPL or U-Boot.

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

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 **`qnxsd-p-env.bat`** (Windows) or `source qnxsd-p-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-h3`). 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-h3_br-704_be-704build_ID.zip
```

where *build_ID* is the BSP build number (e.g., JBN136).

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-h3**). 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:

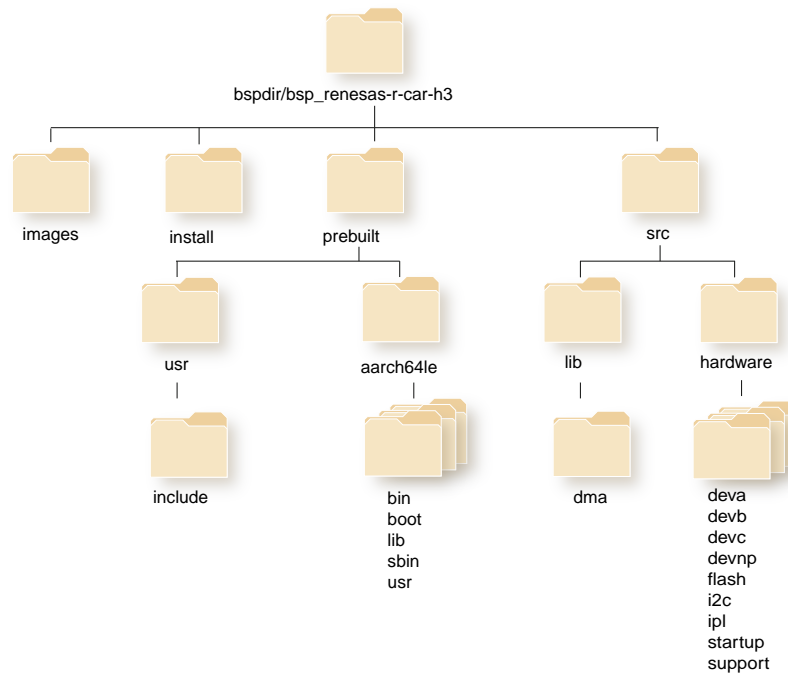


Figure 1: Structure of a BSP

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 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
- location of the USB ports, console, USB serial ports, Ethernet port, or other connectors such as the parallel port for a camera
- any board switches that require setting to boot from USB or even other 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!

Before you connect the power supply, make sure that the power switch on the board is set to OFF.

When you plug in the board's power supply, the fan on the board will immediately start running. This only means that the fan is running. It doesn't mean that the board (SOC, etc.) has powered up. The board only powers up when you set the power switch to 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 microUSB 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 the power supply to the board's power connector. Depending on the board, this can be either 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.
3. On your host machine, start your favorite terminal program (e.g., PuTTY, 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

You may need to set DIP switches on your board. Before you change the DIP switches, we recommend that you first disconnect the power from your board. In addition to modifying the DIP switches, you should be familiar with the memory layout for your board. See one of the following sections to determine the DIP switch settings required for your board and to understand the memory mappings:

- [“Configure the R-Car H3 board”](#)
- [“Configure the R-Car H3 Starter board”](#)
- [“Configure the R-Car M3-W board”](#)
- [“Configure the R-Car M3-W Starter board”](#)
- [“Configure the R-Car M3-N board”](#)

Configure the R-Car H3 board

Before booting your board, you may need to configure some of its switches.

Configure the board switches

For the WS1.0 Silicon revision, use the switch settings shown below. Note that the black block represents the physical lever inside the switch:

QSPI and HyperFlash modes (WS1.0)

Here are the SW1, SW2, SW3, and SW10 switch settings for QSPI mode:

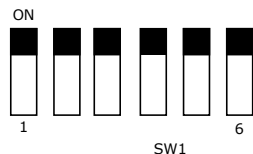


Figure 2: SW1: QSPI-A (WS1.0)

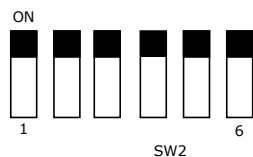


Figure 3: SW2: QSPI-B (WS1.0)



Figure 4: SW3: QSPI-C (WS1.0)

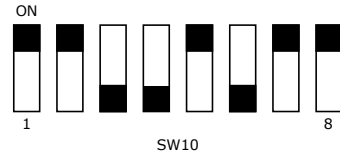


Figure 5: SW10: MODESW-A

The following table summarizes the QSPI mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	ON	ON	ON	ON	ON	ON	–	–
SW2	QSPI-B	ON	ON	ON	ON	ON	ON	–	–
SW3	QSPI-C	OFF	–	–	–	–	–	–	–
SW10	MODESW-A	ON	ON	OFF	OFF	ON	OFF	ON	ON

Below are the SW1, SW2, SW3, and SW10 switch settings for HyperFlash mode:



Figure 6: SW1: HyperFlash (WS1.0)



Figure 7: SW2: HyperFlash



Figure 8: SW3: HyperFlash (WS1.0)

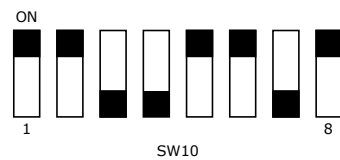


Figure 9: SW10: HyperFlash (WS1.0)

The following table summarizes the HyperFlash mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW2	QSPI-B	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW3	QSPI-C	ON	–	–	–	–	–	–	–
SW10	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	ON

32- and 64-bit modes (WS1.0)

Below are the SW12 switch settings for 32-bit mode:



Figure 10: SW12 MODESW-C for AArch32 - 32-bit mode (WS1.0)

Below are the SW12 switch settings for 64-bit mode:



Figure 11: SW12 MODESW-C for AArch64 - 64-bit mode (WS1.0)

For the WS1.1 and WS2.0 silicon revision, use the switch settings shown below. Note that the black block represents the physical lever inside the switch. “ON” and “1” are silk screen printed on the board:

QSPI and HyperFlash modes (WS 1.1, WS2.0, and WS3.0)

Below are the SW1, SW2, SW3, and SW10 switch settings for QSPI mode:

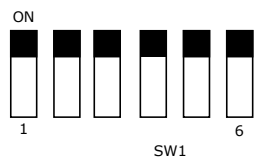


Figure 12: SW1: QSPI (WS 1.1, WS2.0, and WS3.0)

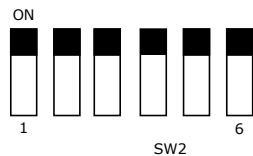


Figure 13: SW2: QSPI (WS 1.1, WS2.0, and WS3.0)



Figure 14: SW3: QSPI (WS1.1 WS2.0, and WS3.0)

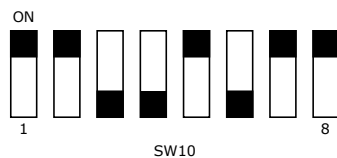


Figure 15: SW10: QSPI (WS 1.1, WS2.0, and WS3.0)

The following table summarizes the QSPI mode (WS1.1, WS2.0, and WS3.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	ON	ON	ON	ON	ON	ON	–	–
SW2	QSPI-B	ON	ON	ON	ON	ON	ON	–	–
SW3	QSPI-C	OFF	–	–	–	–	–	–	–
SW10	MODESW-A	ON	ON	OFF	OFF	ON	OFF	ON	ON

Below are the SW1, SW2, SW3, and SW10 switch settings for HyperFlash mode:

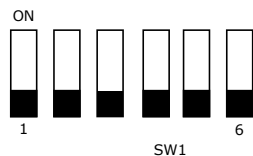


Figure 16: SW1: HyperFlash (WS 1.1, WS2.0, and WS3.0)



Figure 17: SW2: HyperFlash (WS 1.1, WS2.0, and WS3.0)



Figure 18: SW3: HyperFlash (WS 1.1, WS2.0, and WS3.0)

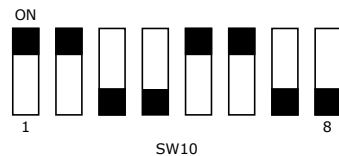


Figure 19: SW10: HyperFlash 80 Mhz (WS 1.1, WS2.0, and WS3.0)

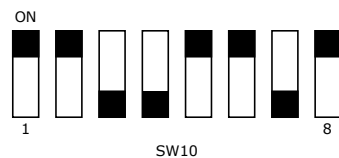


Figure 20: SW10: HyperFlash 160 Mhz (WS 1.1, WS2.0, and WS3.0)

The following table summarizes the HyperFlash mode (WS1.1, WS2.0, WS 3.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW2	QSPI-B	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW3	QSPI-C	ON	–	–	–	–	–	–	–
SW10 (80 Mhz)	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	OFF
SW10 (160 Mhz)	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	ON



For SW10, some boards work at 80 Mhz and others at 160 Mhz. You may need to try both to determine which works for your board.

32- and 64-bit modes (WS1.1 and WS2.0)

Below are the SW12 switch settings for 32-bit mode for WS1.1, and WS2.0:

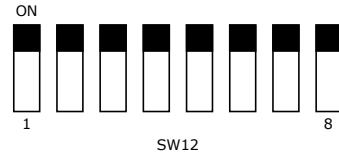


Figure 21: SW12 MODESW-C for AArch32 - 32-bit mode (WS1.1 and WS2.0)

Below are the SW12 switch settings for 64-bit mode for WS1.1 and WS2.0:

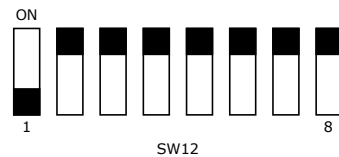


Figure 22: SW12 MODESW-C for AArch64 - 64-bit mode (WS1.1 and WS2.0)

32- and 64-bit modes for WS3.0

You can use the same WS 1.1 and WS 2.0 settings as described previously for the WS 3.0 revisions if you aren't using SATA. If you are using SATA, you must set the SW12 switch settings as described in this section; otherwise the SATA driver won't start when you boot your board.

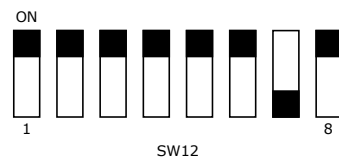


Figure 23: SW12 MODESW-C for AArch32 - 32-bit mode (WS3.0 with SATA)

Below are the SW12 switch settings for 64-bit mode for WS3.0 with SATA:

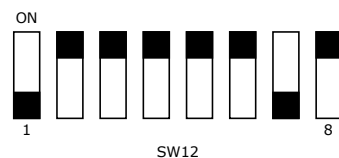


Figure 24: SW12 MODESW-C for AArch64 - 64-bit mode (WS3.0 with SATA)

Set DIP switch to boot with U-Boot

You must ensure that DIP switch SW7 is set to Pin 1 (left-side) to boot with the Renesas Loader (to load U-Boot), otherwise the boot process hangs because U-Boot isn't loaded.

Here's what the boot logs look like if SW7 (DDR BKUP) isn't set correctly:

```
NOTICE: BL2: R-Car Gen3 Initial Program Loader(CA57) Rev.1.0.9
NOTICE: BL2: PRR is R-Car H3 ES1.1
NOTICE: BL2: Boot device is HyperFlash(80MHz)
NOTICE: BL2: LCM state is CM
NOTICE: BL2: AVS setting succeeded. DVFS_SetVID=0x52
NOTICE: BL2: DDR1600(rev.0.15)
NOTICE: BL2: DRAM Split is 4ch
```

```

NOTICE: BL2: QoS is default setting(rev.0.32)
NOTICE: BL2: v1.1(release):3ad02ac
NOTICE: BL2: Built : 09:39:23, Dec 25 2016
NOTICE: BL2: Normal boot
NOTICE: BL2: Skip loading images. (SuspendToRAM)
(hangs)

```

Default settings for the board (WS 1.1, WS2.0, and WS3.0)

You may have changed the settings on various switches and want to get back to the default settings for the board. To do so, use these settings on the board. For more information, see the documentation for the board from the hardware vendor.



The black block represents the physical lever inside the switch in the following illustrations.

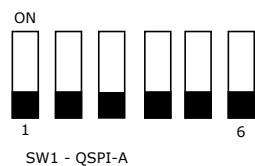


Figure 25: SW1: QSPI-A

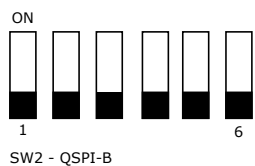


Figure 26: SW2: QSPI-B



Figure 27: SW3: QSPI-C

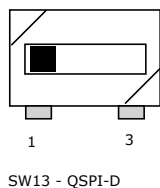


Figure 28: SW13: QSPI-D

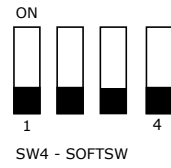


Figure 29: SW4: QSPI-D

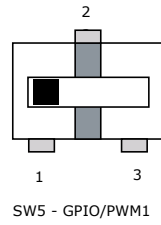


Figure 30: SW5: GPIO/PWM1

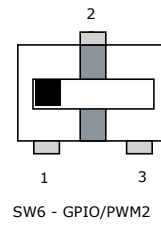


Figure 31: SW6: GPIO/PWM2

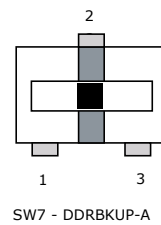


Figure 32: SW7: DDRBKUP-A

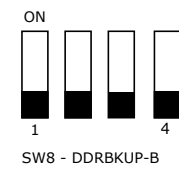


Figure 33: SW8: DDRBKUP-B

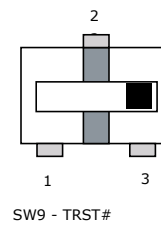
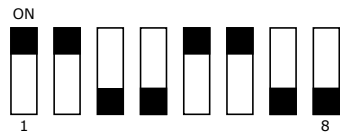


Figure 34: SW9: TRST#



SW10 - MODESW-A

Figure 35: SW10: MODESW-A



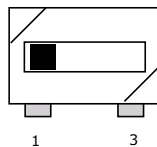
SW11 - MODESW-B

Figure 36: SW11: MODESW-B



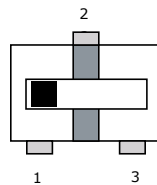
SW12 - MODESW-C

Figure 37: SW12: MODESW-C



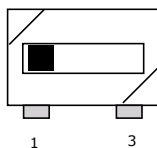
SW14- SSI78-M/S

Figure 38: SW14: SSI78-M/S



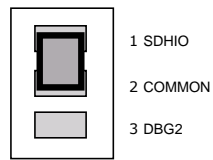
SW15 - USB-SW

Figure 39: SW15:USB-SW



SW16 - SDHIO/DBG2-A

Figure 40: SW16: SDHIO/DBG2-A



JP2 - SDHIO/DBG2-B

Figure 41: JP2: SDHIO/DBG2-B

SW28 - VDDQVA_SD0

Figure 42: SW28: VDDQVA_SD0

SW17 - LVDS

Figure 43: SW17: LVDS

SW29 - MIPI-SW

Figure 44: SW29: MIPI-SW

SW30 - PHYAD

Figure 45: SW30: PHYAD

The following table summarizes the default settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW2	QSPI-B	OFF	OFF	OFF	OFF	OFF	OFF	–	–

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW3	QSPI-C	ON	–	–	–	–	–	–	–
SW13	QSPI-D	SET	–		–	–	–	–	–
SW4	SOFTSW	OFF	OFF	OFF	OFF	–	–	–	–
SW5	GPIO/PWM1	SET			–	–	–	–	–
SW6	GPIO/PWM2	SET			–	–	–	–	–
SW7	DDRBKUP-A		SET		–	–	–	–	–
SW8	DDRBKUP-B	OFF	OFF	OFF	OFF	–	–	–	–
SW9	TRST#			SET	–	–	–	–	–
SW10	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	OFF
SW11	MODESW-B	OFF	ON	ON	ON	ON	ON	ON	ON
SW12	MODESW-C	OFF	ON	ON	ON	ON	ON	ON	ON
SW14	SSI78-M/S	SET	–		–	–	–	–	–
SW15	USB-SW	SET			–	–	–	–	–
SW16	SDH10/DBG2-A	SET	–		–	–	–	–	–
JP2	SDH10/DBG2-B	SET	–		–	–	–	–	–
SW28	VDDQVA_SD0	OFF	–	–	–	–	–	–	–
SW17	LVDS	ON	–	–	–	–	–	–	–
SW29	MIPI-SW	ON	ON	–	–	–	–	–	–
SW30	PHYAD	OFF	OFF	–	–	–	–	–	–

Memory layout for the R-Car H3 board (WS1.1, WS2.0, and WS3.0)

The following is the memory layout for the R-Car H3 board.

Flash memory

The flash memory devices for the R-Car H3 are as follows:

HyperFlash/RPC flash

64 MB of flash memory is available to store Boot Parameters, Loader, and U-Boot.

QSPI

16 MB of flash memory (U5) that contains a preloaded MiniMonitor program provided by Renesas; this program is used to program the software into the HyperFlash/RPC flash memory

SW1, SW2, SW3, SW10 switches control from which of these two flash memory devices the board boots. For more information, see the “Configure the board switches” for the R-Car H3 board.



See also the manufacturer's documentation, in particular the *QNX 7 BSP for R-Car H3 Salvator-X User's Manual: Software*, section 5.2, for information about the bootloader files and the switches and the Hardware Manual for the board.

HyperFlash on the R-Car H3 board for QNX IPL

You'll need the memory layout details in the following table when you program the QNX IPL and the QNX IFS (see “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)”):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	0xE6320000	0x000000
bl2-board_name.srec	Loader	0xE6302000 (0xE6304000 after Renesas BootLoader v1.0.14)	0x040000
cert_header_sa6.srec	Loader (Certification)	0xE6320000	0x180000
bl31-board_name.srec	ARM Trusted Firmware	0x44000000	0x1C0000
tee-board_name.srec	OP-Tee	0x44100000	0x200000
ipl-rcar_gen3-h3.srec	QNX IPL	0x50000000	0x640000
ifs-rcar_h3.srec	QNX IFS	0x40100000	0x740000



- By default, the **ifs-rcar_h3.srec** (or variant) isn't created by this BSP. To generate an S-record formatted (.srec) file, you must modify the buildfile to build it. For more information about how to generate an S-record formatted, see “[Build QNX IFS as a S-record formatted file](#)” in this chapter.

- When you use QNX IPL, the default maximum size of the QNX IFS can be 31 MB. This limit is calculated as follows:

```
QNX load address minus the program top address
0x42000000 - 0x40100000 = 0x1F00000 (31 MB)
```

If you need a larger QNX IFS and if your board has additional memory, you can modify `QNX_LOAD_ADDR` in the

`$BSP_ROOT_DIR/src/hardware/ipl/boards/rcar_gen3/board_name/board.h` file to change the maximum size. Note that the maximum size supported on the board is 62 MB and the

QNX IFS must fit between $0x40100000 - 0x43F00000$ [$0x43F00000 - 0x40100000 = 0x3E00000$ (62 MB)]. However, access to the memory ranges from $0x43F00000$ to $0x47FFFFFF$ aren't permitted because it's the SDRAM protected area. Therefore, if you require the maximum IFS size of 62 MB size, you must change the `QNX_LOAD_ADDR` to $0x48000000$.

HyperFlash on the R-Car H3 board for U-Boot

Below are the memory layout details for the U16 64 MB HyperFlash/RPC flash memory. You'll need this information when you program U-Boot and the IFS (see "[Programming Renesas Loader and U-Boot to HyperFlash](#) "):

Files	Description	Program Top Address	HyperFlash Save Address
<code>bootparam sa0.srec</code>	Boot Parameters	$0xE6320000$	$0x000000$
<code>bl2-board_name.srec</code>	Loader	$0xE6302000$ ($0xE6304000$ after Renesas BootLoader v1.0.14)	$0x040000$
<code>cert_header_sa6.srec</code>	Loader (Certification)	$0xE6320000$	$0x180000$
<code>bl31-board_name.srec</code>	ARM Trusted Firmware	$0x44000000$	$0x1C0000$
<code>tee-board_name.srec</code>	OP-Tee	$0x44100000$	$0x200000$
<code>u-boot-elf.srec</code>	U-Boot	$0x49000000$ ($0x50000000$ for versions after Renesas BootLoader v1.09)	$0x640000$

Secure Memory on the R-Car H3 board (WS 1.1, WS2.0, and WS3.0)

Below are the memory layout details for the secure RAM. You need this information when you program U-Boot and the IFS (see "[Programming Renesas Loader and U-Boot to HyperFlash](#) "):

Start address	Purpose
$0x08000000$	RPC Memory (HyperFlash) 64 MB Used to store Boot Parameters, Loader, and U-Boot.
$0x40000000$	SDRAM 1GB The QNX IFS image is loaded to $0x40100000$.
$0xE6300000$	System RAM 1MB. Memory loaded for the Loader.
$0x500000000$	SDRAM 1 GB

Start address	Purpose
0x600000000	SDRAM 1 GB
0x700000000	SDRAM 1 GB

Configure the R-Car H3 Starter board

Before booting your board, you may need to configure some of its switches.

Configure the board switches

For the WS1.0 Silicon revision, use the switch settings shown below. Note that the black block represents the physical lever inside the switch:

QSPI and HyperFlash modes (WS1.0)


Here are the SW1, SW6, and JP1 switch settings for QSPI mode:

1  ON

SW1 - QSPI/HyperFlash


Figure 46: SW1: HyperFlash/QSPI (WS1.0)



 1 ON


 4

SW2 - DIPSW Software Readable

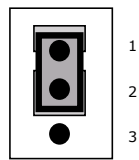
Figure 47: SW2: DIPSW Software Readable (WS1.0)

 1 ON


 4

SW6 - Mode Settings via CPLD

Figure 48: SW6: Mode Settings via CPLD (WS1.0)



JP1 - QSPI Selector

Figure 49: JP1: QSPI Selector (WS1.0)

The following table summarizes the QSPI mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW1	HyperFlash/QSPI	OFF	–	–	–
SW2	DIPSW Software Readable	OFF	OFF	OFF	OFF
SW6	Mode Settings via CPLD	OFF	OFF	OFF	OFF
JP1	QSPI Selector	SET	SET	–	–

Below are the SW1, SW6, and JP1 switch settings for HyperFlash mode:

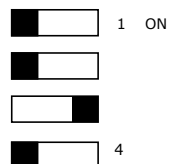


Figure 50: SW1: HyperFlash/QSPI (WS1.0)



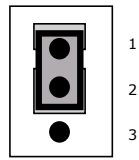
SW2 - DIPSW Software Readable

Figure 51: SW2: DIPSW Software Readable (WS1.0)



SW6 - Mode Settings via CPLD

Figure 52: SW6: Mode Settings via CPLD (WS1.0)



JP1 - QSPI Selector

Figure 53: JP1: QSPI Selector (WS1.0)

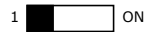
The following table summarizes the HyperFlash mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW1	HyperFlash/QSPI	ON	–	–	–
SW2	DIPSW Software Readable	OFF	OFF	OFF	OFF
SW6	Mode Settings via CPLD	OFF	OFF	ON	OFF
JP1	QSPI Selector	SET	SET	–	–

For the WS1.1 and WS2.0 silicon revisions, use the switch settings shown below. Note that the black block represents the physical lever inside the switch. “ON” and “1” are usually silk screen printed on the board:

QSPI and HyperFlash modes (WS1.1/WS2.0)

Here are the SW1, SW6, and JP1 switch settings for QSPI mode for WS1.1 and WS2.0:

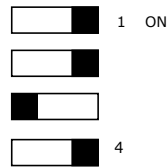


SW1 - QSPI/HyperFlash

Figure 54: SW1: HyperFlash/QSPI (WS1.1/WS2.0)

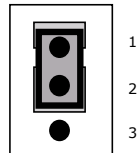
SW2 - DIPSW Software Readable

Figure 55: SW2: DIPSW Software Readable (WS1.1/WS2.0)



SW6 - Mode Settings via CPLD

Figure 56: SW6: Mode setting via CPLD (WS1.1/WS2.0)



JP1 - QSPI Selector

Figure 57: JP1: QSPI Selector (WS1.1/WS2.0)

The following table summarizes the QSPI mode (WS1.1/WS2.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW1	HyperFlash/QSPI	OFF	–	–	–
SW2	DIPSW Software Readable	OFF	OFF	OFF	OFF
SW6	Mode Settings via CPLD	ON	ON	OFF	ON
JP1	QSPI Selector	SET	SET	–	–

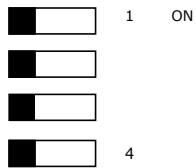
Below are the SW1, SW2, SW6, and JP1 switch settings for HyperFlash mode for WS1.1:



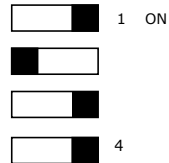
The switch settings are slightly different for WS2.0 for HyperFlash mode, as specified below.



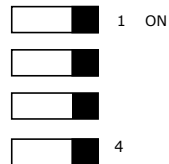
Figure 58: SW1: HyperFlash/QSPI (WS1.1/WS2.0)



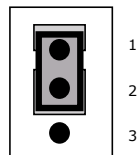
SW2 - DIPSW Software Readable

Figure 59: SW2: DIPSW Software Readable (WS1.1/WS2.0)

SW6 - Mode Settings via CPLD

Figure 60: SW6: Mode Settings via CPLD (WS1.1)

SW6 - Mode Settings via CPLD

Figure 61: SW6: Mode Settings via CPLD (WS2.0)

JP1 - QSPI Selector

Figure 62: JP1: QSPI Selector (WS1.1/WS2.0)

The following table summarizes the HyperFlash mode (WS1.1) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW1 (WS1.1/WS2.0)	HyperFlash/QSPI	ON	–	–	–
SW2 (WS1.1/WS2.0)	DIPSW Software Readable	OFF	OFF	OFF	OFF
SW6 (WS1.1)	Mode Settings via CPLD	ON	OFF	ON	ON
SW6 (WS2.0)	Mode Settings via CPLD	ON	ON	ON	ON
JP1 (WS1.1/WS2.0)	QSPI Selector	SET	SET	–	–

Default settings for the board (WS1.0, WS1.1, and WS2.0)

You may have changed the settings on various switches and want to get back to the default settings for the board. To do so, use these settings on the board. For more information, see the documentation for the board from the hardware vendor.



The black block represents the physical lever inside the switch in the following illustrations.

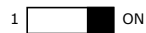
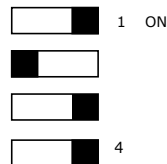


Figure 63: SW1: HyperFlash



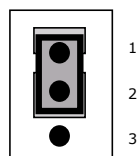
SW2 - DIPSW Software Readable

Figure 64: SW2: DIPSW Software Readable



SW6 - Mode Settings via CPLD

Figure 65: SW6: Mode settings via CPLD



JP1 - QSPI Selector

Figure 66: JP1: QSPI Selector

The following table summarizes the default settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW1	HyperFlash/QSPI	ON	—	—	—

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW2	DIPSW Software Readable	OFF	OFF	OFF	OFF
SW6	Mode Settings via CPLD	ON	OFF	ON	ON
JP1	QSPI Selector	SET	SET	–	–

Memory layout for the R-Car H3 Starter board

The following is the memory layout for the R-Car H3 Starter board.

Flash memory

The flash memory devices for the R-Car H3 Starter board are as follows:

HyperFlash/RPC flash

64 MB of flash memory is available to store Boot Parameters, Loader, and U-Boot.

QSPI

16 MB of flash memory (U5) that contains a preloaded MiniMonitor program provided by Renesas; this program is used to program the software into the HyperFlash/RPC flash memory



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

HyperFlash on the R-Car H3 Starter board for QNX IPL

You'll need the memory layout details in the following table when you program the QNX IPL and the QNX IFS (see “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)”):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	0xE6320000	0x000000
bl2-board_name.srec	Loader	0xE6302000 (0xE6304000 after Renesas BootLoader v1.0.14)	0x040000
cert_header_sa6.srec	Loader (Certification)	0xE6320000	0x180000
bl31-board_name.srec	ARM Trusted Firmware	0x44000000	0x1C0000
tee-board_name.srec	OP-Tee	0x44100000	0x200000
ipl-rcar_gen3-h3.srec	QNX IPL	0x50000000	0x640000
ifs-rcar_h3ulcb.srec	QNX IFS	0x40100000	0x740000



- By default, the **ifs-rcar_h3ulcb.srec** (or variant) isn't created by this BSP. To generate an S-record formatted (.srec) file, you must modify the buildfile to build it. For more information about how to generate an S-record formatted, see [“Build QNX IFS as a S-record formatted file”](#) in this chapter.

- When you use QNX IPL, the default maximum size of the QNX IFS can be 31 MB. This limit is calculated as follows:

```
QNX load address minus the program top address
0x42000000 - 0x40100000 = 0x1F00000 (31 MB)
```

If you need a larger QNX IFS and if your board has additional memory, you can modify `QNX_LOAD_ADDR` in the

`$BSP_ROOT_DIR/src/hardware/ipl/boards/rcar_gen3/board_name/board.h` file to change the maximum size. Note that the maximum size supported on the board is 62 MB and the QNX IFS must fit between `0x40100000 - 0x43F00000` [`0x43F00000 - 0x40100000 = 0x3E00000` (62 MB)]. However, access to the memory ranges from `0x43F00000` to `0x47FFFFFF` aren't permitted because it's the SDRAM protected area. Therefore, if you require the maximum IFS size of 62 MB size, you must change the `QNX_LOAD_ADDR` to `0x48000000`.

HyperFlash on the R-Car H3 Starter board for U-Boot

Below are the memory layout details for the U16 64 MB HyperFlash/RPC flash memory. You'll need this information when you program U-Boot and the IFS (see [“Programming Renesas Loader and U-Boot to HyperFlash”](#)):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	0xE6320000	0x000000
bl2-board_name.srec	Loader	0xE6302000 (0xE6304000 after Renesas BootLoader v1.0.14)	0x040000
cert_header_sa6.srec	Loader (Certification)	0xE6320000	0x180000
bl31-board_name.srec	ARM Trusted Firmware	0x44000000	0x1C0000
tee-board_name.srec	OP-Tee	0x44100000	0x200000
u-boot-elf.srec	U-Boot	0x49000000 (0x50000000 for versions after Renesas BootLoader v1.09)	0x640000

Secure Memory on the R-Car H3 Starter board

Below are the memory layout details for the secure RAM. You need this information when you program U-Boot and the IFS (see “[Programming Renesas Loader and U-Boot to HyperFlash](#) ”):

Start address	Purpose
0x08000000	RPC Memory (HyperFlash) 64 MB Used to store Boot Parameters, Loader, and U-Boot.
0x40000000	SDRAM 1GB The QNX IFS image is loaded to 0x40100000.
0xE6300000	System RAM 1MB. Memory loaded for the Loader.
0x500000000	SDRAM 1 GB
0x600000000	SDRAM 1 GB
0x700000000	SDRAM 1 GB

Configure the R-Car M3-W board

Before booting your board, you may need to configure some of its switches.

Configure the board switches

For the WS 1.0 Silicon revision, use the switch settings shown below. Note that the black block represents the physical lever inside the switch.

QSPI and HyperFlash modes (WS1.0)

Here are the SW1, SW2, SW3, and SW10 switch settings for QSPI mode:

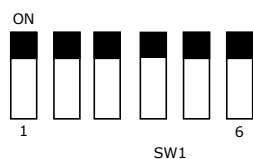


Figure 67: SW1: QSPI-A (WS1.0)

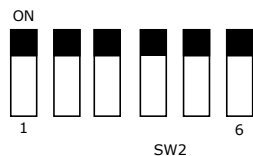


Figure 68: SW2: QSPI-B (WS1.0)



Figure 69: SW3: QSPI-C (WS1.0)



Figure 70: SW10: MODESW-A

The following table summarizes the QSPI mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	ON	ON	ON	ON	ON	ON	–	–
SW2	QSPI-B	ON	ON	ON	ON	ON	ON	–	–
SW3	QSPI-C	OFF	–	–	–	–	–	–	–
SW10	MODESW-A	ON	ON	OFF	OFF	ON	OFF	ON	ON

Below are the SW1, SW2, SW3, and SW10 switch settings for HyperFlash mode:

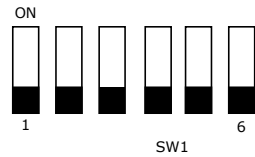


Figure 71: SW1: HyperFlash (WS1.0)

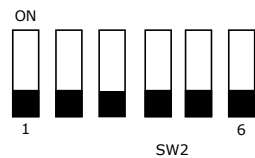


Figure 72: SW2: HyperFlash



Figure 73: SW3: HyperFlash (WS1.0)

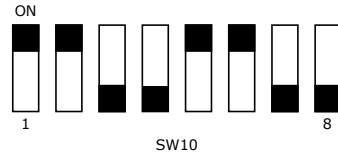


Figure 74: SW10: HyperFlash 80 Mhz (WS1.0)



Figure 75: SW10: HyperFlash 160 Mhz (WS1.0)

The following table summarizes the HyperFlash mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW2	QSPI-B	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW3	QSPI-C	ON	–	–	–	–	–	–	–
SW10 (80 Mhz)	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	OFF
SW10 (160 Mhz)	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	ON



For SW10, some boards work at 80 Mhz and others at 160 Mhz. You may need to try both to determine which works for your board.

32- and 64-bit modes (WS1.0)

Below are the SW12 switch settings for 32-bit mode:

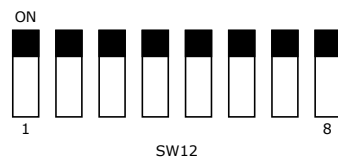


Figure 76: SW12 MODESW-C for AArch32 - 32-bit mode (WS1.0)

Below are the SW12 switch settings for 64-bit mode:

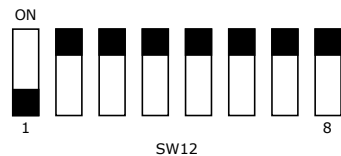


Figure 77: SW12 MODESW-C for AArch64 - 64-bit mode (WS1.0)

Default settings for the board (WS1.0)

You may have changed the settings on various switches and want to get back to the default settings for the board. To do so, use these settings on the board. For more information, see the documentation for the board from the hardware vendor.



The black block represents the physical lever inside the switch in the following illustrations.

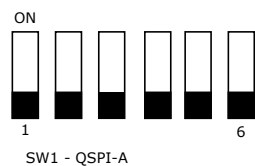


Figure 78: SW1: QSPI-A

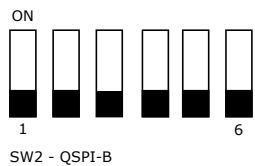


Figure 79: SW2: QSPI-B



Figure 80: SW3: QSPI-C

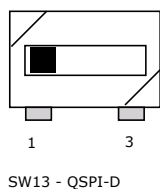


Figure 81: SW13: QSPI-D

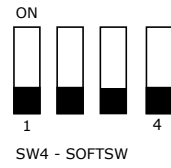


Figure 82: SW4: QSPI-D

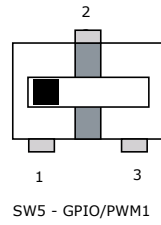


Figure 83: SW5: GPIO/PWM1

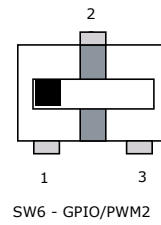


Figure 84: SW6: GPIO/PWM2

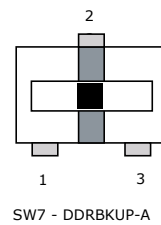


Figure 85: SW7: DDRBKUP-A

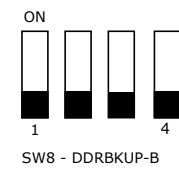


Figure 86: SW8: DDRBKUP-B

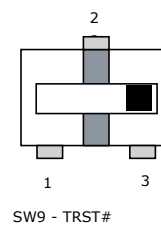
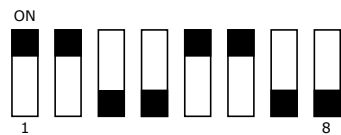


Figure 87: SW9: TRST#



SW10 - MODESW-A

Figure 88: SW10: MODESW-A



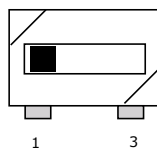
SW11 - MODESW-B

Figure 89: SW11: MODESW-B



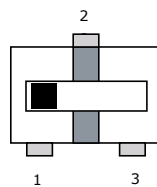
SW12 - MODESW-C

Figure 90: SW12: MODESW-C



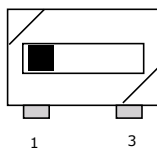
SW14- SSI78-M/S

Figure 91: SW14: SSI78-M/S



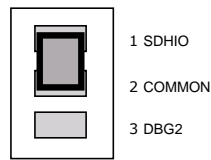
SW15 - USB-SW

Figure 92: SW15:USB-SW



SW16 - SDHIO/DBG2-A

Figure 93: SW16: SDHIO/DBG2-A



JP2 - SDHIO/DBG2-B

Figure 94: JP2: SDH10/DBG2-B

SW28 - VDDQVA_SD0

Figure 95: SW28: VDDQVA_SD0

SW17 - LVDS

Figure 96: SW17: LVDS

SW29 - MIPI-SW

Figure 97: SW29: MIPI-SW

SW30 - PHYAD

Figure 98: SW30: PHYAD

The following table summarizes the default settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW2	QSPI-B	OFF	OFF	OFF	OFF	OFF	OFF	–	–

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW3	QSPI-C	ON	–	–	–	–	–	–	–
SW13	QSPI-D	SET	–		–	–	–	–	–
SW4	SOFTSW	OFF	OFF	OFF	OFF	–	–	–	–
SW5	GPIO/PWM1	SET			–	–	–	–	–
SW6	GPIO/PWM2	SET			–	–	–	–	–
SW7	DDRBKUP-A		SET		–	–	–	–	–
SW8	DDRBKUP-B	OFF	OFF	OFF	OFF	–	–	–	–
SW9	TRST#			SET	–	–	–	–	–
SW10	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	OFF
SW11	MODESW-B	OFF	ON	ON	ON	ON	ON	ON	ON
SW12	MODESW-C	OFF	ON	ON	ON	ON	ON	ON	ON
SW14	SSI78-M/S	SET	–		–	–	–	–	–
SW15	USB-SW	SET			–	–	–	–	–
SW16	SDH10/DBG2-A	SET	–		–	–	–	–	–
JP2	SDH10/DBG2-B	SET	SET		–	–	–	–	–
SW28	VDDQVA_SD0	OFF	–	–	–	–	–	–	–
SW17	LVDS	ON	–	–	–	–	–	–	–
SW29	MIPI-SW	ON	ON	–	–	–	–	–	–
SW30	PHYAD	OFF	OFF	–	–	–	–	–	–

Memory layout for the R-Car M3-W board

The following is the memory layout for the R-Car M3-W board.

Flash memory

The flash memory devices for the R-Car M3-W are as follows:

HyperFlash/RPC flash

64 MB of flash memory is available to store Boot Parameters, Loader, and U-Boot.

QSPI

16 MB of flash memory (U5) that contains a preloaded MiniMonitor program provided by Renesas; this program is used to program the software into the HyperFlash/RPC flash memory

SW1, SW2, SW3, SW10 switches control from which of these two flash memory devices the board boots. For more information, see the “Configure the board switches” for the R-Car M3-W board.



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

HyperFlash on the R-Car M3-W board for QNX IPL

You'll need the memory layout details in the following table when you program the QNX IPL and the QNX IFS (see “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)”):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	0xE6320000	0x000000
bl2-board_name.srec	Loader	0xE6302000 (0xE6304000 after Renesas BootLoader v1.0.14)	0x040000
cert_header_sa6.srec	Loader (Certification)	0xE6320000	0x180000
bl31-board_name.srec	ARM Trusted Firmware	0x44000000	0x1C0000
tee-board_name.srec	OP-Tee	0x44100000	0x200000
ipl-rcar_gen3-m3.srec	QNX IPL	0x50000000	0x640000
ifs-rcar_m3.srec	QNX IFS	0x40100000	0x740000



- By default, the **ifs-rcar_m3.srec** (or variant) isn't created by this BSP. To generate an S-record formatted (.srec) file, you must modify the buildfile to build it. For more information about how to generate an S-record formatted, see “[Build QNX IFS as a S-record formatted file](#)” in this chapter.

- When you use QNX IPL, the default maximum size of the QNX IFS can be 31 MB. This limit is calculated as follows:

```
QNX load address minus the program top address
0x42000000 - 0x40100000 = 0x1F00000 (31 MB)
```

If you need a larger QNX IFS and if your board has additional memory, you can modify QNX_LOAD_ADDR in the

\$BSP_ROOT_DIR/src/hardware/ipl/boards/rcar_gen3/board_name/board.h file to change the maximum size. Note that the maximum size supported on the board is 62 MB and the QNX IFS must fit between 0x40100000 - 0x43F00000 [0x43F00000 -

0x40100000 = 0x3E00000 (62 MB)]. However, access to the memory ranges from 0x43F00000 to 0x47DFFFFFF aren't permitted because it's the SDRAM protected area. Therefore, if you require the maximum IFS size of 62 MB size, you must change the QNX_LOAD_ADDR to 0x48000000.

HyperFlash on the R-Car M3-W board for U-Boot

Below are the memory layout details for the U16 64 MB HyperFlash/RPC flash memory. You'll need this information when you program U-Boot and the IFS (see "[Programming Renesas Loader and U-Boot to HyperFlash](#) "):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	0xE6320000	0x000000
bl2-board_name.srec	Loader	0xE6302000 (0xE6304000 after Renesas BootLoader v1.0.14)	0x040000
cert_header_sa6.srec	Loader (Certification)	0xE6320000	0x180000
bl31-board_name.srec	ARM Trusted Firmware	0x44000000	0x1C0000
tee-board_name.srec	OP-Tee	0x44100000	0x200000
u-boot-elf.srec	U-Boot	0x49000000 (0x50000000 for versions after Renesas BootLoader v1.09)	0x640000

Secure Memory on the R-Car M3-W board

Below are the memory layout details for the secure RAM. You need this information when you program U-Boot and the IFS (see "[Programming Renesas Loader and U-Boot to HyperFlash](#) "):

Start address	Purpose
0x08000000	RPC Memory (HyperFlash) 64 MB Used to store Boot Parameters, Loader, and U-Boot.
0x40000000	SDRAM 2GB The QNX IFS image is loaded to 0x40100000.
0xE6300000	System RAM 1MB. Memory loaded for the Loader.
0x60000000	SDRAM 2 GB

Configure the R-Car M3-W Starter board

Before booting your board, you may need to configure some of its switches.

Configure the board switches

For the WS 1.0 Silicon revision, use the switch settings shown below. Note that the black block represents the physical lever inside the switch:

QSPI and HyperFlash modes (WS1.0)

Here are the SW1, SW6, and JP1 switch settings for QSPI mode:



SW1 - QSPI/HyperFlash

Figure 99: SW1: HyperFlash/QSPI (WS1.0)



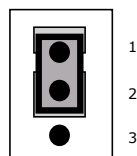
SW2 - DIPSW Software Readable

Figure 100: SW2: DIPSW Software Readable (WS1.0)



SW6 - Mode Settings via CPLD

Figure 101: SW6: Mode Settings via CPLD (WS1.0)



JP1 - QSPI Selector

Figure 102: JP1: QSPI Selector (WS1.0)

The following table summarizes the QSPI mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW1	HyperFlash/QSPI	OFF	–	–	–
SW2	DIPSW Software Readable	OFF	OFF	OFF	OFF
SW6	Mode Settings via CPLD	OFF	OFF	OFF	ON
JP1	QSPI Selector	SET	SET	–	–

Below are the SW1, SW6, and JP1 switch settings for HyperFlash mode:

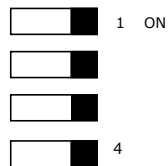


Figure 103: SW1: HyperFlash/QSPI (WS1.0)



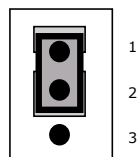
SW2 - DIPSW Software Readable

Figure 104: SW2: DIPSW Software Readable (WS1.0)



SW6 - Mode Settings via CPLD

Figure 105: SW6: Mode Settings via CPLD (WS1.0)



JP1 - QSPI Selector

Figure 106: JP1: QSPI Selector (WS1.0)

The following table summarizes the HyperFlash mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW1	HyperFlash/QSPI	ON	–	–	–
SW2	DIPSW Software Readable	OFF	OFF	OFF	OFF
SW6	Mode Settings via CPLD	ON	ON	ON	ON
JP1	QSPI Selector	SET	SET	–	–

Default settings for the board (WS1.0)

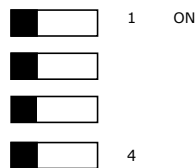
You may have changed the settings on various switches and want to get back to the default settings for the board. To do so, use these settings on the board. For more information, see the documentation for the board from the hardware vendor.



The black block represents the physical lever inside the switch in the following illustrations.

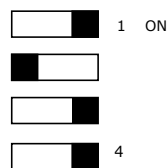


Figure 107: SW1: HyperFlash



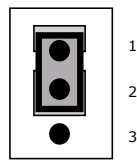
SW2 - DIPSW Software Readable

Figure 108: SW2: DIPSW Software Readable



SW6 - Mode Settings via CPLD

Figure 109: SW6: Mode settings via CPLD



JP1 - QSPI Selector

Figure 110: JP1: QSPI Selector

The following table summarizes the default settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4
SW1	HyperFlash/QSPI	ON	–	–	–
SW2	DIPSW Software Readable	OFF	OFF	OFF	OFF
SW6	Mode Settings via CPLD	ON	OFF	ON	ON
JP1	QSPI Selector	SET	SET	–	–

Memory layout for the R-Car M3-W Starter board

The following is the memory layout for the R-Car M3-W Starter board.

Flash memory

The flash memory devices for the R-Car M3-W Starter board are as follows:

HyperFlash/RPC flash

64 MB of flash memory is available to store Boot Parameters, Loader, and U-Boot.

QSPI

16 MB of flash memory (U5) that contains a preloaded MiniMonitor program provided by Renesas; this program is used to program the software into the HyperFlash/RPC flash memory



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

HyperFlash on the R-Car M3-W Starter board for QNX IPL

You'll need the memory layout details in the following table when you program the QNX IPL and the QNX IFS (see “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)”):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	0xE6320000	0x000000

Files	Description	Program Top Address	HyperFlash Save Address
bl2-board_name.srec	Loader	0xE6302000 (0xE6304000 after Renesas BootLoader v1.0.14)	0x040000
cert_header_sa6.srec	Loader (Certification)	0xE6320000	0x180000
bl31-board_name.srec	ARM Trusted Firmware	0x44000000	0x1C0000
tee-board_name.srec	OP-Tee	0x44100000	0x200000
ipl-rcar_gen3-m3.srec	QNX IPL	0x50000000	0x640000
ifs-rcar_m3ulcb.srec	QNX IFS	0x40100000	0x740000



- By default, the **ifs-rcar_m3ulcb.srec** (or variant) isn't created by this BSP. To generate an S-record formatted (.srec) file, you must modify the buildfile to build it. For more information about how to generate an S-record formatted, see [“Build QNX IFS as a S-record formatted file”](#) in this chapter.
- When you use QNX IPL, the default maximum size of the QNX IFS can be 31 MB. This limit is calculated as follows:

QNX load address minus the program top address
 $0x42000000 - 0x40100000 = 0x1F00000$ (31 MB)

If you need a larger QNX IFS and if your board has additional memory, you can modify `QNX_LOAD_ADDR` in the `$BSP_ROOT_DIR/src/hardware/ipl/boards/rcar_gen3/board_name/board.h` file to change the maximum size. Note that the maximum size supported on the board is 62 MB and the QNX IFS must fit between $0x40100000 - 0x43F00000$ [$0x43F00000 - 0x40100000 = 0x3E00000$ (62 MB)]. However, access to the memory ranges from $0x43F00000$ to $0x47FFFFFF$ aren't permitted because it's the SDRAM protected area. Therefore, if you require the maximum IFS size of 62 MB size, you must change the `QNX_LOAD_ADDR` to $0x48000000$.

HyperFlash on the R-Car M3-W Starter board for U-Boot

Below are the memory layout details for the U16 64 MB HyperFlash/RPC flash memory. You'll need this information when you program U-Boot and the IFS (see [“Programming Renesas Loader and U-Boot to HyperFlash”](#)):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	0xE6320000	0x000000

Files	Description	Program Top Address	HyperFlash Save Address
bl2-board_name.srec	Loader	0xE6302000 (0xE6304000 after Renesas BootLoader v1.0.14)	0x040000
cert_header_sa6.srec	Loader (Certification)	0xE6320000	0x180000
bl31-board_name.srec	ARM Trusted Firmware	0x44000000	0x1C0000
tee-board_name.srec	OP-Tee	0x44100000	0x200000
u-boot-elf.srec	U-Boot	0x49000000 (0x50000000 for versions after Renesas BootLoader v1.09)	0x640000

Secure Memory on the R-Car M3-W Starter board

Below are the memory layout details for the secure RAM. You need this information when you program U-Boot and the IFS (see “[Programming Renesas Loader and U-Boot to HyperFlash](#) ”):

Start address	Purpose
0x08000000	RPC Memory (HyperFlash) 64 MB Used to store Boot Parameters, Loader, and U-Boot.
0x40000000	SDRAM 2GB The QNX IFS image is loaded to 0x40100000.
0xE6300000	System RAM 1MB. Memory loaded for the Loader.
0x60000000	SDRAM 2 GB



Some early R-Car M3-W Starter boards (version WS1.0) have 1GB of SDRAM with a start address of 0x40000000 and an additional 1GB with a start address of 0x60000000.

Configure the R-Car M3-N board

Before booting your board, you may need to configure some of its switches.

Configure the board switches

For the WS 1.0 Silicon revision, use the switch settings shown below. Note that the black block represents the physical lever inside the switch.

QSPI and HyperFlash modes (WS1.0)

Here are the SW1, SW2, SW3, and SW10 switch settings for QSPI mode:

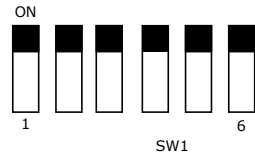


Figure 111: SW1: QSPI-A (WS1.0)

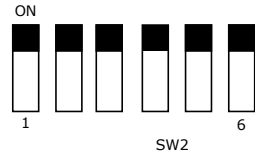


Figure 112: SW2: QSPI-B (WS1.0)



Figure 113: SW3: QSPI-C (WS1.0)

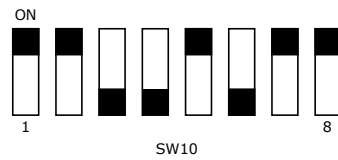


Figure 114: SW10: MODESW-A

The following table summarizes the QSPI mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	ON	ON	ON	ON	ON	ON	–	–
SW2	QSPI-B	ON	ON	ON	ON	ON	ON	–	–
SW3	QSPI-C	OFF	–	–	–	–	–	–	–
SW10	MODESW-A	ON	ON	OFF	OFF	ON	OFF	ON	ON

Below are the SW1, SW2, SW3, and SW10 switch settings for HyperFlash mode:

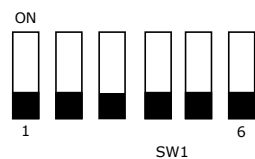


Figure 115: SW1: HyperFlash (WS1.0)

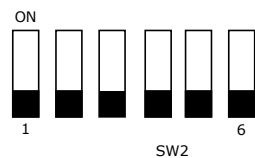


Figure 116: SW2: HyperFlash



Figure 117: SW3: HyperFlash (WS1.0)

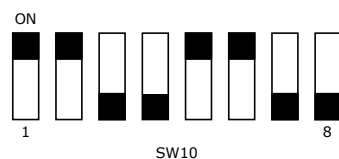


Figure 118: SW10: HyperFlash 80 Mhz (WS1.0)

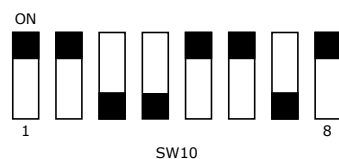


Figure 119: SW10: HyperFlash 160 Mhz (WS1.0)

The following table summarizes the HyperFlash mode (WS1.0) settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW2	QSPI-B	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW3	QSPI-C	ON	–	–	–	–	–	–	–
SW10 (80 MHz)	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	OFF
SW10 (160 MHz)	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	ON



For SW10, some boards work at 80 Mhz and others at 160 Mhz. You may need to try both to determine which works for your board.

32- and 64-bit modes (WS1.0)

Below are the SW12 switch settings for 32-bit mode:

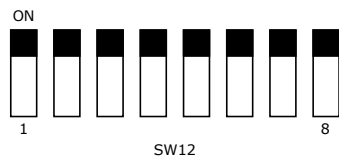


Figure 120: SW12 MODESW-C for AArch32 - 32-bit mode (WS1.0)

Below are the SW12 switch settings for 64-bit mode:

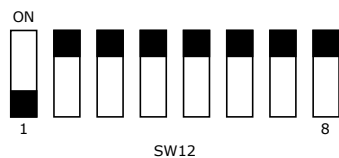


Figure 121: SW12 MODESW-C for AArch64 - 64-bit mode (WS1.0)

Default settings for the board (WS1.0)

You may have changed the settings on various switches and want to get back to the default settings for the board. To do so, use these settings on the board. For more information, see the documentation for the board from the hardware vendor.



The black block represents the physical lever inside the switch in the following illustrations.

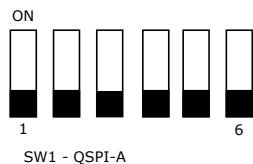


Figure 122: SW1: QSPI-A

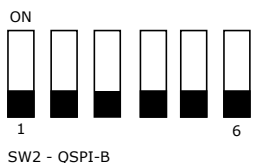


Figure 123: SW2: QSPI-B

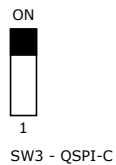


Figure 124: SW3: QSPI-C

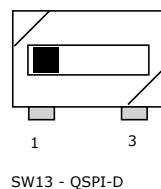


Figure 125: SW13: QSPI-D

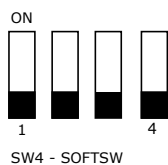


Figure 126: SW4: QSPI-D

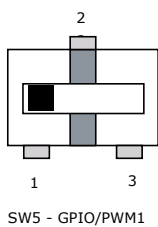


Figure 127: SW5: GPIO/PWM1

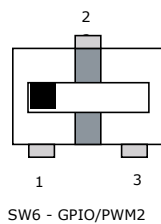


Figure 128: SW6: GPIO/PWM2

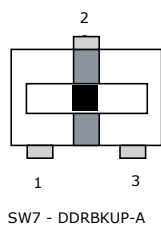
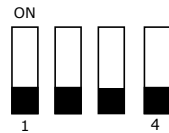
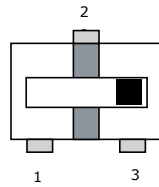


Figure 129: SW7: DDRBKUP-A



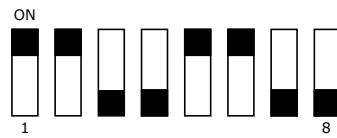
SW8 - DDRBKUP-B

Figure 130: SW8: DDRBKUP-B



SW9 - TRST#

Figure 131: SW9: TRST#



SW10 - MODESW-A

Figure 132: SW10: MODESW-A



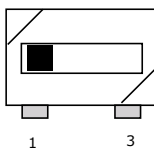
SW11 - MODESW-B

Figure 133: SW11: MODESW-B



SW12 - MODESW-C

Figure 134: SW12: MODESW-C



SW14- SSI78-M/S

Figure 135: SW14: SSI78-M/S

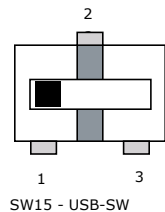


Figure 136: SW15:USB-SW

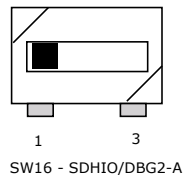


Figure 137: SW16: SDHIO/DBG2-A

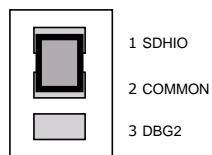


Figure 138: JP2: SDH10/DBG2-B



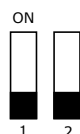
Figure 139: SW28: VDDQVA_SD0



Figure 140: SW17: LVDS



Figure 141: SW29: MIPI-SW



SW30 - PHYAD

Figure 142: SW30: PHYAD

The following table summarizes the default settings for the board:

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW1	QSPI-A	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW2	QSPI-B	OFF	OFF	OFF	OFF	OFF	OFF	–	–
SW3	QSPI-C	ON	–	–	–	–	–	–	–
SW13	QSPI-D	SET	–		–	–	–	–	–
SW4	SOFTSW	OFF	OFF	OFF	OFF	–	–	–	–
SW5	GPIO/PWM1	SET			–	–	–	–	–
SW6	GPIO/PWM2	SET			–	–	–	–	–
SW7	DDRBKUP-A		SET		–	–	–	–	–
SW8	DDRBKUP-B	OFF	OFF	OFF	OFF	–	–	–	–
SW9	TRST#			SET	–	–	–	–	–
SW10	MODESW-A	ON	ON	OFF	OFF	ON	ON	OFF	OFF
SW11	MODESW-B	OFF	ON	ON	ON	ON	ON	ON	ON
SW12	MODESW-C	OFF	ON	ON	ON	ON	ON	ON	ON
SW14	SSI78-M/S	SET	–		–	–	–	–	–
SW15	USB-SW	SET			–	–	–	–	–
SW16	SDH10/DBG2-A	SET	–		–	–	–	–	–
JP2	SDH10/DBG2-B	SET	SET		–	–	–	–	–
SW28	VDDQVA_SD0	OFF	–	–	–	–	–	–	–
SW17	LVDS	ON	–	–	–	–	–	–	–
SW29	MIPI-SW	ON	ON	–	–	–	–	–	–

Switch number	Switch name	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
SW30	PHYAD	OFF	OFF	–	–	–	–	–	–

Memory layout for the R-Car M3-N board

The following is the memory layout for the R-Car M3-N board.

Flash memory

The flash memory devices for the R-Car M3-N are as follows:

HyperFlash/RPC flash

64 MB of flash memory is available to store Boot Parameters, Loader, and U-Boot.

QSPI

16 MB of flash memory (U5) that contains a preloaded MiniMonitor program provided by Renesas; this program is used to program the software into the HyperFlash/RPC flash memory

SW1, SW2, SW3, SW10 switches control from which of these two flash memory devices the board boots. For more information, see the “Configure the board switches” for the R-Car M3-N board.



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

HyperFlash on the R-Car M3-N board for QNX IPL

You'll need the memory layout details in the following table when you program the QNX IPL and the QNX IFS (see “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)”):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	0xE6320000	0x000000
bl2-board_name.srec	Loader	0xE6302000 (0xE6304000 after Renesas BootLoader v1.0.14)	0x040000
cert_header_sa6.srec	Loader (Certification)	0xE6320000	0x180000
bl31-board_name.srec	ARM Trusted Firmware	0x44000000	0x1C0000
tee-board_name.srec	OP-Tee	0x44100000	0x200000
ipl-rcar_gen3-m3.srec	QNX IPL	0x50000000	0x640000
ifs-rcar_m3n.srec	QNX IFS	0x40100000	0x740000



- By default, the **ifs-rcar_m3n.srec** (or variant) isn't created by this BSP. To generate an S-record formatted (.srec) file, you must modify the buildfile to build it. For more information about how to generate an S-record formatted, see "[Build QNX IFS as a S-record formatted file](#)" in this chapter.
- When you use QNX IPL, the default maximum size of the QNX IFS can be 31 MB. This limit is calculated as follows:

QNX load address minus the program top address
 $0x42000000 - 0x40100000 = 0x1F00000$ (31 MB)

If you need a larger QNX IFS and if your board has additional memory, you can modify **QNX_LOAD_ADDR** in the

\$BSP_ROOT_DIR/src/hardware/ipl/boards/rcar_gen3/board_name/board.h file to change the maximum size. Note that the maximum size supported on the board is 62 MB and the QNX IFS must fit between $0x40100000 - 0x43F00000$ [$0x43F00000 - 0x40100000 = 0x3E00000$ (62 MB)]. However, access to the memory ranges from $0x43F00000$ to $0x47FFFFFF$ aren't permitted because it's the SDRAM protected area. Therefore, if you require the maximum IFS size of 62 MB size, you must change the **QNX_LOAD_ADDR** to $0x48000000$.

HyperFlash on the R-Car M3-N board for U-Boot

Below are the memory layout details for the U16 64 MB HyperFlash/RPC flash memory. You'll need this information when you program U-Boot and the IFS (see "[Programming Renesas Loader and U-Boot to HyperFlash](#) "):

Files	Description	Program Top Address	HyperFlash Save Address
bootparam sa0.srec	Boot Parameters	$0xE6320000$	$0x000000$
bl2-board_name.srec	Loader	$0xE6302000$ ($0xE6304000$ after Renesas BootLoader v1.0.14)	$0x040000$
cert_header_sa6.srec	Loader (Certification)	$0xE6320000$	$0x180000$
bl31-board_name.srec	ARM Trusted Firmware	$0x44000000$	$0x1C0000$
tee-board_name.srec	OP-Tee	$0x44100000$	$0x200000$
u-boot-elf.srec	U-Boot	$0x49000000$ ($0x50000000$ for versions after Renesas BootLoader v1.09)	$0x640000$

Secure Memory on the R-Car M3-N board

Below are the memory layout details for the secure RAM. You need this information when you program U-Boot and the IFS (see “[Programming Renesas Loader and U-Boot to HyperFlash](#) ”):

Start address	Purpose
0x08000000	RPC Memory (HyperFlash) 64 MB Used to store Boot Parameters, Loader, and U-Boot.
0x40000000	SDRAM 2GB The QNX IFS image is loaded to 0x40100000.
0xE6300000	System RAM 1MB. Memory loaded for the Loader.
0x600000000	SDRAM 2 GB

Transfer image and boot the board

You can transfer an QNX IFS image (or simply *IFS*) to the R-Car H3, R-Car H3 Starter, R-Car M3-W, R-Car M3-N, or R-Car M3-W Starter board's RAM to boot it and run the QNX Neutrino RTOS. The only difference is the name of the IFS file you'll use when you boot the board.

Board boot process

These are the three boot options available initially boot your board. The SPI Loader is provided so that you use it to program (or flash) the HyperFlash memory with either the QNX IPL, or the Renesas BootLoader and U-Boot. The QNX IPL and U-Boot can then be used to load the QNX IFS to boot your board to run the QNX Neutrino RTOS.

SPI Loader and MiniMonitor

QSPI is the default setting for boards from Renesas. The board should be pre-programmed with the SPI Loader and the MiniMonitor software on the QSPI, but if it isn't, contact Renesas to get the steps (documentation), the SPI loader, and the MiniMonitor software to flash the QSPI. The SPI Loader is a bootloader that loads the MiniMonitor software. You must use MiniMonitor to flash the board's HyperFlash with one of the following to load the QNX IFS:

- Renesas BootLoader and QNX IPL or;
- Renesas BootLoader and U-Boot

Renesas Bootloader and QNX IPL

The Renesas Bootloader is first required and you can get it from Renesas, or you can rebuilt it. The Renesas Bootloader completes the necessary on-chip module initialization and loads the QNX IPL and transfers execution to the QNX IPL. You can also flash the QNX IFS to HyperFlash using MiniMonitor. When you flash the QNX IFS using MiniMonitor, it must be an S-record (**.srec**) formatted file. For more information, see the “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)” section in this chapter.



When you use QNX IPL, the default maximum size of the IFS can be 31 MB. This limit is calculated as follows:

```
QNX load address minus the program top address
0x42000000 - 0x40100000 = 0x1F00000 (31 MB)
```

You can change this limit if your board has additional memory by doing the following:

- Modify the `QNX_LOAD_ADDR` in the `$BSP_ROOT_DIR/src/hardware/ipi/boards/rcar_gen3/board_name/board.h` where `board_name` represents the board your are using.
- Modify the corresponding `image` attribute in the buildfile.
- Rebuild the QNX IFS

For example, you can change the `QNX_LOAD_ADDR` to `0x48000000` so that you would have a maximum IFS size of 62 MB.

The QNX IPL isn't pre-programmed onto the board, therefore you'll need to flash it based on the steps described in the “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)” section of this chapter.

Renesas Bootloader and U-Boot

The Renesas Bootloader is first required and you can get it from Renesas, or you can rebuilt it. The Renesas Bootloader completes the necessary on-chip module initialization, loads U-Boot, and transfers execution to U-boot. Each time you boot the board using U-Boot, you must load the IFS to run QNX Neutrino. This bootloader isn't pre-programmed onto the board, therefore you'll need to flash it based on the steps described in the “[Programming Renesas Loader and U-Boot to HyperFlash](#)” section of this chapter.

The steps below summarize how to boot your board with the QNX Neutrino RTOS:

1. Set the power switch to OFF on the board.
2. Connect your board to required cables and power. For information about connecting your board, see the “[Set up the hardware](#)” section in the “[Installation Notes](#)” chapter of this guide.



CAUTION: Ensure that you use only the power supply provided with the board.

3. To boot the IFS to run the QNX Neutrino, perform one of the following options based on whether you want to use QNX IPL or U-Boot.
 - [Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)
 - [Programming Renesas Loader and U-Boot to HyperFlash](#)
4. Perform one of the following based on the bootloader you chose in the previous step:

For QNX IPL:

After you boot with QNX IPL, you can choose to load the QNX IFS from an SD card, the eMMC, or HyperFlash memory. You can also send the IFS file using a serial connection from your host computer (via the `sendnto` command). For more information, see “[Boot the QNX IFS from an SD card, eMMC, HyperFlash, or serial using QNX IPL.](#)”

For U-Boot:

After you have U-Boot running, you can load the QNX IFS into RAM using one of these options:

- Transfer the QNX IFS to RAM to boot the board using an SD card. For more information, see “[Configure the board to boot using U-Boot and from an SD card.](#)”
- After you've booted using an SD card, you can also transfer the QNX IFS to the eMMC to boot from there. For more information, see “[Configure the board to boot using U-Boot and from eMMC memory.](#)”
- Transfer the QNX IFS to RAM to boot the board using a TFTP Server. For more information, see “[Configure the board to boot using U-Boot and TFTP.](#)”

Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash

You can use the MiniMonitor to flash the QNX IPL and QNX IFS to HyperFlash.

The steps shown here describe how to use MiniMonitor to program the QNX IPL and the QNX IFS to HyperFlash memory. It's important to note that you must rebuild the QNX IFS so that you have an S-record formatted file (.srec) when you use MiniMonitor to program (flash) the HyperFlash. For more information, see “[Build QNX IFS as a S-record formatted file.](#)”



To use QNX IPL, you must have the DIP switch set correctly on your board otherwise the boot process hangs. For example, for the R-Car H3, see the “[Set DIP switch to boot with U-Boot](#)” section in this chapter.

You must use the MiniMonitor software provided by Renesas (preloaded on SPI Flash U17) to program the QNX IPL to HyperFlash. The steps below are for the R-Car H3, R-Car M3-W, and R-Car M3-N variants of the boards.

For information about how to program QNX IPL to HyperFlash on the R-Car H3 Starter, or R-Car M3-W Starter, see the following websites, but use the QNX IPL values described

- For the R-Car H3 Starter, see http://elinux.org/R-Car/Boards/H3SK#Flashing_firmware, but use the values specified in “[HyperFlash on the R-Car H3 Starter board for QNX IPL](#)” for QNX IPL. For the DIP switch settings to boot the board into a specific mode, see “[Configure the R-Car H3 Starter board](#)”.
 - For the R-Car M3-W Starter, see http://elinux.org/R-Car/Boards/M3SK#Flashing_firmware. For the DIP switch settings to boot a board in a specific mode, see “[Configure the R-Car M3-W Starter board](#).”
1. Boot the board in QSPI and AArch64 mode. Based on the board you are using, see one of these sections in this chapter for more information about configuring the board:
 - “[Configure the R-Car H3 board](#)”
 - “[Configure the R-Car H3 Starter board](#)”
 - “[Configure the R-Car M3-W board](#)”
 - “[Configure the R-Car M3-W Starter board](#)”
 - “[Configure the R-Car M3-N board](#)”

2. In your console to the board, you should see the MiniMonitor start.

```
SALVATOR SAMPLE LOADER V1.15 2016.03.08
CPU : AArch64 CA57
DRAM : LPDDR4 DDR1600 / 1RANK (4GB)
DEVICE: QSPI Flash(S25FS128) at 40MHz DMA
BOOT : Normal Boot
Backup: DDR Cold Boot

jump to 0xE6330000

SALVATOR MiniMonitor V0.20 2016.04.04
Work Memory SystemRAM (H'E6328000-H'E632FFFF)
>
```

3. After the MiniMonitor starts, enter the command `xls2`. You should see something similar to the following in your console:

```
===== Qspi/HyperFlash writing of Gen3 Board Command =====
Load Program to Spiflash
Writes to any of SPI address.
Please select,FlashMemory.
1 : QspiFlash      (U5 : S25FS128S)
2 : QspiFlash Board (CN3: S25FL512S)
3 : HyperFlash     (SiP internal)
```

4. Type 3 to select HyperFlash.
5. On the board, set the SW1 and SW2 switches all OFF and then type Y. You should see the following in your console.

```
SW1 SW2 All OFF! Setting OK? (Push Y key)
```

6. On the board set the SW3 switch to ON and then, press the Y key.

```
SW3 ON! Setting OK? (Push Y key)
```

7. Enter the Input Program Top Address for *bootparam_sa0.srec*: H'E6320000. Press **Enter**.

```
===== Please Input Program Top Address =====
Please Input : H'E6320000
```

8. Enter the HyperFlash Save Address for *bootparam_sa0.srec* of H'000000 and then press **Enter**.

```
===== Please Input Qspi/HyperFlash Save Address ===
Please Input : H'000000
```

9. When MiniMonitor prompts you to send, transmit *bootparam_sa0.srec* from your host.



If you're using a Tera Term, you transmit the bootloader file using, in the tool, select **File** > **File Transmission(s)** to send !.

```
please send ! ( '.' & CR stop load)
```

10. You may be prompted to clear the writing area if there is already data in that area. If so, Enter y.

```
SPI Data Clear(H'FF) Check :H'00000000-0003FFFF Clear OK?(y/n)
```

11. Wait for MiniMonitor to finish saving to flash. After it is done, the following message may appear in your console:

```
SAVE SPI-FLASH..... complete!
```

12. Repeat steps 3-11 to flash the other files as listed for the board you are using. Flashing the QNX IFS is optional, but if you want to flash it using the MiniMonitor, it must be an S-record formatted file — not a raw binary (**.bin**). For more information about rebuilding the IFS as an S-formatted file, see “[Rebuild QNX IFS as an S-record formatted file.](#)”

13. Turn the board OFF.

14. Put the board in HyperFlash and AArch64 mode (if applicable) and then boot the board, you should see the following prompt below to indicate that the QNX IPL was loaded successfully:

```
QNX Neutrino Initial Program Loader for the R-Car H3
IPL compiled Apr 26 2019 23:08:52
Commands:
Press 'S' for serial download, using the 'sendnto' utility
```

```

Press 'E' for eMMC download, IFS filename MUST be 'QNX-IFS'
Press 'M' for SD/MMC download, IFS filename MUST be 'QNX-IFS'
Press 'H' for Hyperflash Download

```

For more information to put the board in HyperFlash and AArch64 mode, see the configuration information for your board.

- [“Configure the R-Car H3 board”](#)
- [“Configure the R-Car H3 Starter board”](#)
- [“Configure the R-Car M3-W board”](#)
- [“Configure the R-Car M3-W Starter board”](#)
- [“Configure the R-Car M3-N board”](#)

15. (Optional) If you had used the MiniMonitor to program the QNX IFS to HyperFlash memory, you can press H and the board should be booted into QNX Neutrino as shown below:

```

Welcome to QNX Neutrino 7.0 on the R-Car H3 Salvator-X Board
Starting slogger and pipe servers...
Starting watchdog ...
Starting Serial driver...
Starting I2C driver ...
Starting USB Host driver ...
Starting SPI Flash driver...
Starting Audio driver ...
Starting SDHI memory card driver ...
Starting MMC memory flash driver ...
Path=0 - rcar_gen3
  target=0 lun=0      Direct-Access(0) - SDMMC: SA08G Rev: c.7
Starting Networking ...
Path=0 - rcar_gen3
  target=0 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=1 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=2 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=3 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=4 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=5 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=6 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=7 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
Starting SATA driver...
xpt_configure: rcarsata SIM attach failure
Unable to access /dev/sata0
#

```

Rebuild QNX IFS as an S-record formatted file

By default, this BSP doesn't provide an S-record formatted file so you must modify the buildfile to build one only if you want to flash the QNX IFS to HyperFlash memory using MiniMonitor. If you want to copy the QNX IFS to HyperFlash after you booted into QNX Neutrino, you don't need to rebuild the QNX IFS as an S-formatted file.

1. On your host computer, edit the buildfile that builds the QNX IFS for your board. Change the [virtual] attribute to build an S-formatted file instead of a raw binary. To make this change, in your buildfile modify the line [virtual=aarch64le,raw] .bootstrap = { to

[virtual=aarch64le,**srec**] .bootstrap = { For more information about formats, see “mkifs” in the QNX Neutrino *Utilities Reference*.

2. Rebuild the QNX IFS (image) as described in [Build the BSP](#).

After you build, you should have an QNX IFS file with the **.srec** extension.

Boot the QNX IFS from an SD card, eMMC, HyperFlash, or serial using QNX IPL

You must first flash the QNX IPL onto the board before you proceed with the steps in this section.

To do this, follow the steps described in “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)”, but simply don't flash the QNX IFS using MiniMonitor.

After you flash either the QNX IPL onto the board (using steps from “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)”), set the DIP switches to HyperFlash mode. After you do this, the QNX IPL boots and you can use it to load the QNX IFS using one of the following methods:

- SD card. For information, see “[Boot from SD card](#)”
- eMMC For information. For more information see “[Boot from eMMC](#)”
- HyperFlash memory. For information, see “[Boot from HyperFlash](#)”
- serial connection. For information, see “[Send QNX IFS to boot board](#)”



The IFS file you choose is specific to the board. In the steps below, the examples use the IFS for the R-Car H3 board.

Boot from SD card

Transfer the QNX IFS file to the SD card

1. On your host system, in a Command Prompt window (Windows) or a terminal (Linux, macOS), insert your SD to the host computer, and navigate to the ***\$BSP_ROOT_DIR/images*** directory.
2. Copy the IFS image to the SD card and rename it to **qnx-ifs**. For example:

```
cd $BSP_ROOT_DIR/images
cp ifs-rcar_h3.bin SD_card_location/qnx-ifs
```

3. Eject the SD card from your host system and insert it into the SD slot on your board.
4. Reset the board or cycle the power to reboot the board. After the QNX IPL loads, press **M** to boot the board from the SD card:

```
QNX Neutrino Initial Program Loader for the R-Car H3
IPL compiled Apr 26 2019 23:08:52
Commands:
  Press 'S' for serial download, using the 'sendnto' utility
  Press 'E' for eMMC download, IFS filename MUST be 'QNX-IFS'
  Press 'M' for SD/MMC download, IFS filename MUST be 'QNX-IFS'
  Press 'H' for Hyperflash Download
SD/MMC download...
load image done.
Scanning for image at @ 0x42000000
Found image           @ 0x42000FA0
Jumping to startup    @ 0x40101800
```


You should see the board boot and the command prompt appear when your board completes booting as shown here:



Since there isn't a SATA driver connected to our board, the `xpt_configure:` `rcarsata` SIM attach failure and Unable to access `/dev/sata0` errors you see in the following example is expected.

```
SD/MMC download...
load image done.
Scanning for image at @ 0x42000000
Found image           @ 0x42000FA0
Jumping to startup    @ 0x40101800

init_board_type board_id=0x00000020
init_timer board_ext_clk=16640000
board_smp_num_cpu: 8 cores, max allowed 4
MMU: 16-bit ASID 44-bit PA TCR_EL1=b5183519
GICv2: 512 interrupts
GICv2: routing SPIs to gic cpu 0
cpu0: MPIDR=80000000

...
...
...
System page at phys:0000000040011000 user:ffffff804020f000 kern:ffffff804020c000
Starting next program at vffffff80600859a0
MMFLAGS=1
Welcome to QNX Neutrino 7.0 on the R-Car H3 Salvator-X Board
Starting slogger and pipe servers...
Starting watchdog ...
Starting Serial driver...
Starting I2C driver ...
Starting USB Host driver ...
Starting SPI Flash driver...
Starting Audio driver ...
Starting SDHI memory card driver ...
Starting MMC memory flash driver ...
Path=0 - rcar_gen3
  target=0 lun=0      Direct-Access(0) - SDMMC: SA08G Rev: c.7
Starting Networking ...
Path=0 - rcar_gen3
  target=0 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=1 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=2 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=3 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=4 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=5 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=6 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=7 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
Starting SATA driver...
xpt_configure:  rcarsata SIM attach failure
```

```
Unable to access /dev/sata0
#
```

Boot from eMMC

You can prepare an eMMC memory device to load an QNX IFS image to the board. After you have QNX Neutrino running on your target board (e.g., booting from SD card), complete the following steps to load the IFS to the eMMC:

1. (Optional) Copy **ifs-rcar_h3.bin** as **QNX-IFS** to the **/tmp** directory on your target board. You can use the QNX Momentics IDE to copy the image to the board or FTP the image to your board.
2. If eMMC device contains a proper FAT32 partition, skip to to step 5; otherwise proceed with this step.
3. Create a FAT32 partition on the eMMC using the `fdisk` command:

```
# fdisk /dev/mmc0 add -t 11
```

4. Mount and format the eMMC device using these commands:

```
# mount -e /dev/mmc0
# mkdosfs /dev/mmc0t11
```

5. Mount the FAT32 partition of the eMMC using the following command:

```
# mount -t dos /dev/mmc0t11 /fs/emmc
```

6. If you performed step 1, you can copy the **QNX-IFS** file from the **/tmp** directory to the **fs/emmc** directory; otherwise, you can also copy it from the SD card you used to run QNX Neutrino on your board. The example below shows you how to copy it from the **/tmp** directory:

```
cp /tmp/QNX-IFS /fs/emmc/QNX-IFS
```

7. Reset the board or cycle the power to reboot the board. After the QNX IPL loads, press **E** to load the QNX IFS image from your eMMC partition. You should see the following occur on your console:

```
Welcome to QNX Neutrino 7.0 on the R-Car H3 Salvator-X Board
Starting slogger and pipe servers...
Starting watchdog ...
Starting Serial driver...
Starting I2C driver ...
Starting USB Host driver ...
Starting SPI Flash driver...
Starting Audio driver ...
Starting SDHI memory card driver ...
Starting MMC memory flash driver ...
Path=0 - rcar_gen3
  target=0 lun=0      Direct-Access(0) - SDMMC: SA08G Rev: c.7
Starting Networking ...
Path=0 - rcar_gen3
  target=0 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=1 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=2 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=3 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=4 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=5 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=6 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=7 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
```

```
Starting SATA driver...
xpt_configure: rcarsata SIM attach failure
Unable to access /dev/sata0
#
```

Boot from HyperFlash memory

You can program (flash)HyperFlash memory two ways. You can either load an S-formatted file using the steps described in the “[Programming Renesas Loader, QNX IPL, and QNX IFS to HyperFlash](#)” or you can copy the QNX IFS as a raw binary file (.bin) as provided in this BSP using these steps:

1. Erase the Hyperflash memory starting from offset 7427K (0x740000) using the following command:

```
# ls /dev/fs*
/dev/fs0          /dev/fs0p0
# flashctl -p /dev/fs0 -o7424K -l32M -ev
```

2. Format a new partition and restart the driver. You should see the device after you've restarted the flash driver (devf-rcar_qspi-gen3).

```
# flashctl -p /dev/fs0p0 -o7424K -l32M -fv
# slay devf-rcar_qspi-gen3; devf-rcar_qspi-gen3
# ls /dev/fs*
/dev/fs0          /dev/fs0p0      /dev/fs0p1      /dev/fs0p2
```

3. Erase the new partition. For example, the new device is /dev/fs0p1.

```
# flashctl -p /dev/fs0p1 -ev
Erasing device /dev/fs0p1
```

4. Copy the QNX IFS to HyperFlash memory and then restart the flash driver. Note that in this case, we've copied the QNX IFS file from the SD card.

```
# cp -V /sd/qnx-ifs /dev/fs0p1
cp: Copying /sd/qnx-ifs to /dev/fs0p1
100.00% (7220/7220 KBytes, 35 KBytes/s)
# slay devf-rcar_qspi-gen3; devf-rcar_qspi-gen3
```

5. Reboot the board and when prompted, press H to load the QNX IFS from HyperFlash memory:

```
QNX Neutrino Initial Program Loader for the R-Car H3
IPL compiled Apr 26 2019 23:08:52
Commands:
  Press 'S' for serial download, using the 'sendnto' utility
  Press 'E' for eMMC download, IFS filename MUST be 'QNX-IFS'
  Press 'M' for SD/MMC download, IFS filename MUST be 'QNX-IFS'
  Press 'H' for Hyperflash Download
  Hyperflash download
...
....
Welcome to QNX Neutrino 7.0 on the R-Car H3 Salvator-X Board
Starting slogger and pipe servers...
Starting watchdog ...
Starting Serial driver...
Starting I2C driver ...
Starting USB Host driver ...
Starting SPI Flash driver...
```

```
Starting Audio driver ...
Starting SDHI memory card driver ...
Starting MMC memory flash driver ...
Path=0 - rcar_gen3
  target=0 lun=0      Direct-Access(0) - SDMMC: SA08G Rev: c.7
Starting Networking ...
Path=0 - rcar_gen3
  target=0 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=1 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=2 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=3 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=4 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=5 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=6 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
  target=7 lun=0      Direct-Access(0) - SDMMC: BGSD3R Rev: 0.7
Starting SATA driver...
xpt_configure: rcarsata SIM attach failure
Unable to access /dev/sata0
#
```

Send QNX IFS to boot board

You can send the load the QNX IFS using the XModem protocol. These steps don't require that you prepare a bootable SD card or transfer the image to a SD card. These steps transfer the IFS images over a serial connection to the board.



If you're doing this in Windows, require a serial connection program that supports sending files via XModem, such as Tera Term.

1. Power on the board or cycle the power to reboot the board. After the QNX IPL loads, you should see the following in your console:

```
QNX Neutrino Initial Program Loader for the R-Car H3
IPL compiled Apr 26 2019 23:08:52
Commands:
  Press 'S' for serial download, using the 'sendnto' utility
  Press 'E' for eMMC download, IFS filename MUST be 'QNX-IFS'
  Press 'M' for SD/MMC download, IFS filename MUST be 'QNX-IFS'
  Press 'H' for Hyperflash Download
```

2. Type `S` to indicate that you want to send the QNX IFS file. You should see the following message in your console:

```
Waiting to receive IFS file...
```

3. If you're using a separate console program as you do on a Windows host, disconnect from it. You'll need that serial connection to send the IFS file to the board.

4. On your host, from command line, use the **`$QNX_HOST/usr/bin/sendnto`** with the `-r` option to send the file because the QNX IFS file for this BSP is declared as raw binary file (starting header is at `0xfb0`). For example:

- On Windows, before you use `sendnto`, you must set the COM port transmit settings using `mode` in a Command Prompt window (administrator mode). For example, if you want to set COM port 8, use the following command:

```
>mode COM8 BAUD=115200 PARITY=n DATA=8
```

```
Status for device COM8:
```

```
-----
Baud:          115200
Parity:         None
Data Bits:      8
Stop Bits:      1
Timeout:        ON
XON/XOFF:       OFF
CTS handshaking: OFF
DSR handshaking: OFF
DSR sensitivity: OFF
DTR circuit:    ON
RTS circuit:    ON
```

```
>sendnto -r -d com8 $BSP_ROOT_DIR/images/ifs-rcar_h3.bin
Percent Complete: 100%
```

- On Linux, for example on `ttyUSB0`:

```
root@qnx:~$ sudo sendnto -r -d
/dev/ttyUSB0 $BSP_ROOT_DIR/images/ifs-rcar_h3.bin
Percent Complete: 100%
```

QNX Neutrino should now be running on your target. If you type `uname` at the prompt, you should see `QNX` returned. You can test the OS by executing any shell command, or any command residing within the OS image (`ls`, `pidin`, etc.).

Build U-Boot and Renesas Loader

You can get the source for U-Boot and Renesas Bootloader (HyperFlash Loader).

The steps below are best done on a Linux host. If you use a Windows or macOS host, you may need to use the Yocto Project Build Appliance. For more information, see the *Build Appliance Manual* page at the Yocto project website; <https://www.yoctoproject.org/documentation/build-appliance-manual>.

1. Set up your host machine for the Yocto Project. For detailed information, see *3.2 The Build Host Packages* section on the Yocto Project Mega-Manual page; <http://www.yoctoproject.org/docs/2.1/mega-manual/mega-manual.html>.
2. Get the latest Yocto Project Files for the Krogth branch. You can either clone using Git or download the file at *Downloads* page; <https://www.yoctoproject.org/downloads>. If you downloaded the file manually, you must first extract it. After you download the project, you should see a directory named **`poky-krogth-release_number`**, which we will call your `POKY_BASE_DIR`.

3. Under *POKY_BASE_DIR*, download the Renesas R-Car Layer using these steps from your terminal:

```
> cd POKY_BASE_DIR
> git clone -b krogoth https://github.com/renesas-rcar/meta-renesas.git
```

4. Initialize a build. It will automatically change paths to *POKY_BASE_DIR/build*

```
cd POKY_BASE_DIR
source oe-init-build-env
```

5. Follow the contents of the **meta-renesas/meta-rcar-gen3/README** file to add the necessary layer dependencies and configurations. See **meta-renesas/meta-rcar-gen3/README.proprietary** for additional configurations.
6. Run the following commands to compile. **Bitbake** downloads packages while it builds, so ensure you're connected to the internet. You can use these commands to build:

bitbake core-image-minimal

Build all images (U-Boot, BootParam, Cert Header, BL2, BL3, and Optee).

bitbake u-boot

Build U-Boot only.

bitbake arm-trusted-firmware

Build the BootPram, Cert Header, BL2, and BL31.

bitbake optee-os

Build the Optee OS.

After the build is done, the binaries are located at:

POKY_BASE_DIR/build/tmp/ deploy/images/MACHINE where *MACHINE* is variable specified in the **meta-renesas/meta-rcar-gen3/README** file.

7. (Optional) If you would like to make modifications to the code, the source is located at ***POKY_BASE_DIR/build/tmp/work/MACHINE-poky-linux/component/component-ver/git***. For example, ***poky-krogoth-2.1/build/tmp/work/MACHINE-poky-linux/arm-trusted-firmware/v1.1+renesas+gitAUTOINC+3ad02acfc4-r0/git***. To build the code after your modifications, run **bitbake -c compile -f component**. For example:

```
> bitbake -c compile -f u-boot
> bitbake core-image-minimal
```

Building Bootloader to build modified secure settings

This board uses the LifeC module to control security/safety access protection of memory and the on-chip bus. This feature is enabled on the Renesas Loader on versions 1.0.9 and later. If you're using builds **meta-renesas** revision **4ae9100bec471fbc423dd5a9476b6aaf8510d5b2** and later, we recommend that you do the following to run on the QNX Neutrino RTOS:

Disable DRAM protection

The **startup** binary provided in this BSP reads the DRAM configuration registers to determine the total DRAM installed. These registers are protected by LifeC, which prevents **startup** from determining the total memory size. To disable the DRAM protection feature, compile with **LIFEC_DBSC_PROTECT_ENABLE** set to 0.

Disable RPC/HyperFlash protection

RPC (HyperFlash) is also protected by LifeC. For the flash driver in this BSP to function, you must set bit 22 to 1 on the SEC_SEL13 and set bit 22 to 0 on SEC_GRP0COND13 and SEC_GRP1COND13 in the `/plat/renesas/rcar/bl2_secure_setting.c` file.

Permit watchdog to trigger a reset

You must set `WDTRSTCR.RWDT_RSTMSK` (0xE6160054 bit 0) is set to 0 so that the watchdog can trigger a reset on the board. This bit is set to 1 by default, and can only be toggled in the BootLoader because the register can only be changed when in secure mode.

To build modified secure settings, use these steps:

1. Add the `LIFEC_DBSC_PROTECT_ENABLE` parameter to `arm-trusted-firmware-src/plat/renesas/rcar/platform.mk` file. Make sure it is before the `#ifndef LIFEC_DBSC_PROTECT_ENABLE` and check `LIFEC_DBSC_PROTECT_ENABLE := 0`.
2. Open `arm-trusted-firmware-src/plat/renesas/rcar/bl2_secure_setting.c` file and make these changes:

- Set SEC_SEL13 bit 22 to 1. Here's what the change looks like:

```
/** Security attribute setting for slave ports 13          */
/* Bit22: RPC slave ports.                                */
/*      0: registers can be accessed from secure resource only. */
{SEC_SEL13, 0xFFFFFFFFU}
```

- Set SEC_GRP0COND13 and SEC_GRP1COND13 bit 22 to 0:

```
/** Security group 0 attribute setting for slave ports 13 */
/** Security group 1 attribute setting for slave ports 13 */
/* Bit22: RPC slave ports.                                */
/*      SecurityGroup3                                    */
{SEC_GRP0COND13, 0x00000000U}
{SEC_GRP1COND13, 0x00000000U}
```

3. Modify `optee/core/arch/arm/plat-rcar/drivers/qspi_hyper_flash.c` so that you don't reset the Hyperflash (rpc) module in optee and lock the module again:

```
...
/* For Hyperflash on QNX OS */
/* Do not do soft reset RPC module, It will lock
 * the RPC module again which was unlocked in the ATF
 */
#if 0
    /* Reset RPC */

    dataL = 0x00020000; /* Bit17 RPC reset */
    static uint32_t init_rpc(void)
    *((volatile uint32_t *)CPG_CPGWPR) = ~dataL;
    *((volatile uint32_t *)CPG_SRSTCLR9) = dataL;
    soft_delay(1); /* wait 1ms (40us) */
#endif
...
...
```

If you reset the module here, you'll revert the changes you made in the previous step.

4. Modify the `bl2__swdt_disable()` function and ensure that `WDTRSTCR.RWDT_RSTMSK` (0xE6160054 bit 0) is set to 0 as follows:

```
...
...
rmsk = mmio_read_32(RST_WDTRSTCR) & WDTRSTCR_MASK_ALL;
rmsk &= ~(0x1);
mmio_write_32(RST_WDTRSTCR, (WDTRSTCR_UPPER_BYTE
    | (rmsk | SWDT_RSTMSK)));
while ((mmio_read_8(SWDT_WTCSRA) & WTCSRA_WRFLG) != 0U) {
...
...
}
```

5. Recompile using these commands:

```
> bitbake -c compile -f arm-trusted-firmware
> bitbake core-image-minimal
```

Building Bootloader to enable big.LITTLE

The default image when you build this BSP starts only the primary cores. To enable the secondary cores on your board to use all cores (four A53 and four A57 on H3, four A53 and two A57 on M3), you must rebuild the ARM trusted firmware BL31:

1. Add the string `PSCI_DISABLE_BIGLITTLE_IN_CA57BOOT=0` to `ATFW_OPT_r8a779x` in `meta-renesas/meta-rcar-gen3/recipes-bsp/arm-trusted-firmware/arm-trusted-firmware_git.bb`. For example:

```
...
ATFW_OPT_r8a7795 = "LSI=H3 RCAR_DRAM_SPLIT=1 ${ATFW_OPT_LOSSY}
    PSCI_DISABLE_BIGLITTLE_IN_CA57BOOT=0"
...
```

2. Recompile using these commands:

```
> bitbake -c compile -f arm-trusted-firmware
> bitbake core-image-minimal
```

3. Remove the `-P` option from your `startup` command.

Programming Renesas Loader and U-Boot to HyperFlash

You can use U-Boot as loader for your board as an alternative to using QNX IPL.

Some Renesas Bootloaders enable lossy decompression feature, which reserves the memory area 0x54000000 to 0x56ffffff. This memory area shouldn't be used by the QNX Neutrino RTOS otherwise memory faults occur. If this feature is used, you must specify the `-r` for the `startup-rcar_*` command you use. For more information, see the relevant “Driver Commands” section. For example, if you are using the R-Car H3, you would use the `startup-rcar_h3` command and see the information in the “[Startup](#)” section in the “[Driver Commands \(R-Car H3\)](#)” chapter.



- If you already have U-Boot programmed on your HyperFlash memory, you can skip the steps in this section.
- To use U-Boot, you must have the DIP switch set correctly on your board otherwise the boot process hangs. For more information, see the “[Set DIP switch to boot with U-Boot](#)” section in this chapter.

You must use MiniMonitor, which is a utility provided by Renesas (preloaded in SPI Flash U17) to program U-Boot to HyperFlash. The following steps provided describe how to program the Renesas Loader and U-Boot to HyperFlash for the R-Car H3 and R-Car M3 boards. For information about how to program U-Boot to HyperFlash on the R-Car H3 Starter or R-Car M3-W Starter, see the following websites:

- For the R-Car H3 Starter, see http://elinux.org/R-Car/Boards/H3SK#Flashing_firmware. For the DIP switch settings to boot a board in a specific mode, see “[Configure the R-Car H3 Starter board](#)”
 - For the R-Car M3-W Starter, see http://elinux.org/R-Car/Boards/M3SK#Flashing_firmware. For the DIP switch settings to boot a board in a specific mode, see “[Configure the R-Car M3-W Starter board](#).”
1. Boot the board in QSPI and AArch64 mode. Based on the board you are using, see one of these sections in this chapter for more information based on the board that you're using:
 - “[Configure the R-Car H3 board](#)”
 - “[Configure the R-Car H3 Starter board](#)”
 - “[Configure the R-Car M3-W board](#)”
 - “[Configure the R-Car M3-W Starter board](#)”
 - “[Configure the R-Car M3-N board](#)”

2. In your console to the board, you should see the MiniMonitor start.

```
SALVATOR SAMPLE LOADER V1.15 2016.03.08
CPU : AArch64 CA57
DRAM : LPDDR4 DDR1600 / 1RANK (4GB)
DEVICE: QSPI Flash(S25FS128) at 40MHz DMA
BOOT : Normal Boot
Backup: DDR Cold Boot

jump to 0xE6330000

SALVATOR MiniMonitor V0.20 2016.04.04
Work Memory SystemRAM (H'E6328000-H'E632FFFF)
>
```

3. After the MiniMonitor starts, enter the command `xls2`. You should see something similar to the following in your console:

```
===== Qspi/HyperFlash writing of Gen3 Board Command =====
Load Program to Spiflash
Writes to any of SPI address.
Please select,FlashMemory.
1 : QspiFlash          (U5 : S25FS128S)
2 : QspiFlash Board (CN3: S25FL512S)
3 : HyperFlash        (SiP internal)
```

4. Type 3 to select HyperFlash.
5. On the board, set the SW1 and SW2 switches all OFF and then type Y. You should see the following in your console.

```
SW1 SW2 All OFF! Setting OK? (Push Y key)
```

6. On the board set the SW3 switch to ON and then, press the Y key.

```
SW3 ON! Setting OK? (Push Y key)
```

7. Enter the Input Program Top Address for `bootparam_sa0.srec`: H'E6320000. Press **Enter**.

```
===== Please Input Program Top Address =====
Please Input : H'E6320000
```

8. Enter the HyperFlash Save Address for `bootparam_sa0.srec` of H'000000 and then press **Enter**.

```
===== Please Input Qspi/HyperFlash Save Address ===
Please Input : H'000000
```

9. When MiniMonitor prompts you to send, transmit `bootparam_sa0.srec` from your host.



If you're using a Tera Term, you transmit the bootloader file using, in the tool, select **File > File Transmission(s)** to send !.

```
please send ! ( '.' & CR stop load)
```

10. You may be prompted to clear the writing area if there is already data in that area. If so, Enter y.

```
SPI Data Clear(H'FF) Check :H'00000000-0003FFFF Clear OK?(y/n)
```

11. Wait for MiniMonitor to finish saving to flash. After it is done, the following message appears in your console:

```
SAVE SPI-FLASH..... complete!
```

12. Repeat steps 3-11 for the rest of the files as follows:

- for the R-Car H3, see “[HyperFlash on the R-Car H3 board for U-Boot](#)”
- for the R-Car H3 Starter board, see “[HyperFlash on the R-Car H3 Starter board for U-Boot](#)”
- for the R-Car M3-W board, see in the “[Installation Notes](#)” chapter “[HyperFlash on the R-Car M3-W board for U-Boot](#)” in the “[Installation Notes](#)” chapter”
- for the R-Car M3-W Starter board, see in the “[Installation Notes](#)” chapter “[HyperFlash on the R-Car M3-W Starter board for U-Boot](#)”
- for the R-Car M3-N board, see in the “[Installation Notes](#)” chapter “[HyperFlash on the R-Car M3-N board for U-Boot](#)” in the “[Installation Notes](#)” chapter

in the chapter) for your board.

13. Turn the board OFF.

14. Put the board in HyperFlash and AArch64 mode (if applicable) and then boot the board. For more information, see the configuration information for your board.

- “[Configure the R-Car H3 board](#)”
- “[Configure the R-Car H3 Starter board](#)”
- “[Configure the R-Car M3-W board](#)”
- “[Configure the R-Car M3-W Starter board](#)”

- [“Configure the R-Car M3-N board”](#)

You're now running U-Boot. You can now configure your board to boot from SD card (see [“Configure the board to boot using U-Boot and from an SD card”](#)) or using TFTP (see [“Configure the board to boot using U-Boot and TFTP”](#)). For example, here's what you should see in your console session:

```
NOTICE: BL2: R-Car Gen3 Initial Program Loader(CA57) Rev.1.0.6
NOTICE: BL2: PRR is R-Car H3 ES1.1
NOTICE: BL2: LCM state is CM
NOTICE: BL2: DDR2400(rev.0.15)
NOTICE: BL2: DRAM Split is 4ch
NOTICE: BL2: QoS is Gfx Oriented(rev.0.30)
NOTICE: BL2: AVS setting succeeded. DVFS_SetVID=0x52
NOTICE: BL2: Lossy Decomp areas
NOTICE:      Entry 0: DCMPCAREACRax:0x80000540 DCMPCAREACRBx:0x570
NOTICE:      Entry 1: DCMPCAREACRax:0x40000000 DCMPCAREACRBx:0x0
NOTICE:      Entry 2: DCMPCAREACRax:0x20000000 DCMPCAREACRBx:0x0
NOTICE: BL2: v1.1(release):
NOTICE: BL2: Built : 14:26:12, Jul 18 2016
NOTICE: BL2: Normal boot
NOTICE: BL2: dst=0xe63150c8 src=0x8180000 len=36(0x24)
NOTICE: BL2: dst=0x43f00000 src=0x8180400 len=3072(0xc00)
NOTICE: BL2: dst=0x44000000 src=0x81c0000 len=65536(0x10000)
NOTICE: BL2: dst=0x44100000 src=0x8200000 len=524288(0x80000)
NOTICE: BL2: dst=0x49000000 src=0x8640000 len=1048576(0x100000)
```

```
U-Boot 2015.04 (Jun 13 2016 - 09:39:10)
```

```
CPU: Renesas Electronics R8A7795 rev 1.1
Board: Salvator-X
I2C:   ready
DRAM:  3.9 GiB
MMC:   sh-sdhi: 0, sh-sdhi: 1, sh-sdhi: 2
In:    serial
Out:   serial
Err:   serial
Net:   ravb
Hit any key to stop autoboot:  3  2  0
=>
```

Configure the board to boot using U-Boot and from an SD card

After you have your boot images, you need to put them on your bootable SD card.

Then you can boot the board using U-Boot and load the IFS into RAM from a SD card. The IFS must be copied to your SD card. For information about copying the IFS to your card, see [Transfer the IFS file to the SD card](#). You can either manually load or configure the board to automatically load the IFS file from a chosen SD card slot. For more information, see [“Manually boot the IFS from SD card”](#) and [“Load and boot the IFS automatically.”](#)

Transfer the IFS file to the SD card



The IFS file you choose is specific to the board. In the steps below, the examples use the IFS for the R-Car H3.

1. On your host system, in a Command Prompt window (Windows) or a terminal (Linux, macOS), insert your SD to the host computer, and navigate to the **`$BSP_ROOT_DIR/images`** directory.
2. Copy the IFS image to the SD card.

```
cd $BSP_ROOT_DIR/images
cp ifs-rcar_h3.bin SD_card_location/ifs-rcar_h3.bin
```
3. Eject the card from your host system.

Manually boot the IFS from SD card

Choose this option when you want control when the board loads the IFS from the SD card. This might be useful when you change the slot that the IFS boots from.



The IFS file you choose is specific to the board. In the steps below, the examples use the IFS for the R-Car H3.

1. Put the board into HyperFlash and AArch64 mode. For more information, see the appropriate section for your board in the “[Set up the hardware](#)” section of this chapter.
2. Insert the SD card in either slot SD0 (CN13) or SD3 (CN14) on the board. The card should contain your IFS image. For more information, see “[Transfer the IFS file to the SD card](#).”
3. Turn on the power on the board, and after U-Boot runs, press any key to abort the autoboot process in your console.
4. In your console, load the IFS image to address 0x40100000 from the SD card. For example, for the R-Car H3, the value of `card_number` is 0 if you inserted the SD card in SD0 (CN13) or 2 if you inserted the card into SD3 (CN14).

```
fatload mmc number 0x40100000 name_of_the_QNX-IFS_image
```

For example, if you inserted your SD card into SD0 and you're using an IFS named (**`ifs-rcar_h3.bin`**), you would type the following command:

```
fatload mmc 0 0x40100000 ifs-rcar_h3.bin
```

5. Type `go 0x40100000`. This command boots the image and runs QNX Neutrino RTOS on your board:

```
go 0x40100000
```

Load and boot the IFS automatically from an SD card

You can configure your board to automatically boot the IFS from a predetermined SD card slot on the board.

1. Put the board into HyperFlash and AArch64 mode mode. For more information, see the appropriate section for your board in the “[Set up the hardware](#)” section of this chapter.

2. Insert the SD card in the slot on the board. The card should contain your IFS image. For more information, see “[Transfer the IFS file to the SD card.](#)”
3. Turn on the power on the board, and after U-Boot is runs, press any key to abort the autoboot process in your console.
4. Set the following environment variables and run the command `saveenv` after each environment variable you set to save the settings on the board. This ensures that the setting is preserved the next time the board reboots. You can also use command `printenv` to view the current board settings.

- *sdslot*: This variable specifies the slot number to use. For example, on the R-Car H3, set the slot number to 0 (CN13) or 2 (CN14) depending on the SD slot that you use:

```
setenv sdslot '0'
saveenv
```

- *loadaddr*: This specifies the address to load the image from the SD:

```
setenv loadaddr 0x40100000
saveenv
```

- *bootifs*: This specifies the name of the IFS file to copy to RAM from the SD. For example, if the name of your IFS file is **ifs-rcar_h3.bin**, here's what you would enter:

```
setenv bootifs ifs-rcar_h3.bin
saveenv
```

- *bootcmd_fatload*: The commands to load the IFS file into RAM. These commands can be more than one line. The following example shows the commands to load the IFS file the specified for the **ifs-rcar_h3.bin** for the R-Car H3 board.

```
setenv bootcmd_fatload 'mmc dev ${sdslot};
fatload mmc ${sdslot}:1 ${loadaddr} ${bootifs};
go ${loadaddr}'
saveenv
```

- *bootcmd*: This specifies the command to run when the board boots. Set this to the *bootcmd_fatload* environment variable.

```
setenv bootcmd 'run bootcmd_fatload'
saveenv
```

The next time you restart the board, it automatically loads and boots the IFS from the SD card you configured as the *sdslot* variable.

Configure the board to boot using U-Boot and from eMMC memory

After you have your boot images, you can copy the QNX IFS to and load the QNX IFS from the eMMC.

You can boot the board using U-Boot and load the IFS into RAM from the eMMC. The QNX IFS must be first copied to the eMMC. For information about copying the IFS to your card, see “[Copy the QNX IFS to the eMMC.](#)” You can then manually load the eMMC or configure the board to automatically load the QNX IFS file from the eMMC. For more information, see “[Manually boot the IFS from SD card](#)” and “[Load and boot the IFS automatically.](#)” The section below presumes that you have first booted

using an SD card. For more information about booting using an SD card, see “[Configure the board to boot using U-Boot and from an SD card](#)” in this chapter.

Copy the QNX IFS to the eMMC

The steps here presume that you have first booted the board to run QNX Neutrino using an SD card. For more information, see “[Configure the board to boot using U-Boot and from an SD card](#)” in this chapter.

1. (Optional) Copy **ifs-rcar_h3.bin** as **QNX-IFS** to the **/tmp** directory on your target board. You can use the QNX Momentics IDE to copy the image to the board or FTP the image to your board.
2. If eMMC device contains a proper FAT32 partition, skip to to step 5; otherwise proceed with this step.
3. Create a FAT32 partition on the eMMC using the `fdisk` command:

```
# fdisk /dev/mmc0 add -t 11
```

4. Mount and format the eMMC device using these commands:

```
# mount -e /dev/mmc0  
# mkdosfs /dev/mmc0t11
```

5. Mount the FAT32 partition of the eMMC using the following command:

```
# mount -t dos /dev/mmc0t11 /fs/emmc
```

6. If you performed step 1, you can copy the **QNX-IFS** file from the **/tmp** directory to the **fs/emmc** directory; otherwise, you can also copy it from the SD card you used to run QNX Neutrino on your board. The example below shows you how to copy it from the **/tmp** directory:

```
cp /tmp/QNX-IFS /fs/emmc/QNX-IFS
```

Boot the QNX IFS from eMMC

The following steps presume you have copied the IFS to the eMMC. For more information, see “[Copy the QNX IFS to the eMMC](#)” in this chapter.



The IFS file you choose is specific to the board. In the steps below, the examples use the IFS for the R-Car H3.

1. Put the board into HyperFlash and AArch64 mode. For more information, see the appropriate section for your board in the “[Set up the hardware](#)” section of this chapter.
2. Turn on the power on the board, and after U-Boot runs, press any key to abort the autoboot process in your console.
3. In your console, load the IFS image to address `0x40100000` from the eMMC. For example, for the R-Car H3, use the following command.

```
fatload mmc 1 0x40100000 name_of_the_QNX-IFS_image
```

Type `go 0x40100000`. This command boots the image and runs the QNX Neutrino RTOS on your board:

```
go 0x40100000
```

Load and boot the IFS automatically from eMMC

The following steps presume you have copied the IFS to the eMMC. For more information, see “[Copy the QNX IFS to the eMMC](#)” in this chapter.

1. Put the board into HyperFlash and AArch64 mode. For more information, see the appropriate section for your board in the “[Set up the hardware](#)” section of this chapter.
2. Turn on the power on the board, and after U-Boot is runs, press any key to abort the autoboot process in your console.
3. Set the following environment variables and run the command `saveenv` after each environment variable you set to save the settings on the board. This ensures that the setting is preserved the next time the board reboots. You can also use command `printenv` to view the current board settings.

- *emmc*: This variable specifies

```
setenv emmc '1'
saveenv
```

- *loadaddr*: This specifies the address to load the image from the eMMC:

```
setenv loadaddr 0x40100000
saveenv
```

- *bootifs*: This specifies the name of the IFS file to copy to RAM from the eMMC. For example, if the name of your IFS file is **ifs-rcar_h3.bin**, here's what you would enter:

```
setenv bootifs ifs-rcar_h3.bin
saveenv
```

- *bootcmd_fatload*: The commands to load the IFS file into RAM. These commands can be more than one line. The following example shows the commands to load the IFS file the specified for the `ifs-rcar_h3.bin` for the R-Car H3 board.

```
setenv bootcmd_fatload 'mmc dev ${emmc};
fatload mmc ${emmc}:1 ${loadaddr} ${bootifs};
go ${loadaddr}'
saveenv
```

- *bootcmd*: This specifies the command to run when the board boots. Set this to the *bootcmd_fatload* environment variable.

```
setenv bootcmd 'run bootcmd_fatload'
saveenv
```

The next time you restart the board, it automatically loads and boots the IFS from the eMMC.

Configure the board to boot using U-Boot and TFTP

After you program the board to boot using U-Boot, you can configure U-Boot to load the IFS into RAM via TFTP. After you configure the board to boot using TFTP, your board boots using TFTP after this procedure completes.

Running QNX IFS using TFTP

You need to have a functional TFTP Boot Server working on your host machine. The TFTP must have your IFS image copied to the TFTP server directory. The IFS image you use is specific to your board. In the steps below, the examples use the IFS for the R-Car H3. TFTP is a boot service that's used to perform OS installations on remote devices. Depending on your host, you can use one of the following:

- Install a utility to run TFTP. For example, Tftpd32 at <http://tftpd32.jounin.net/> (Windows)
- Configure a TFTP server (Linux, macOS).

1. Power on the board and after U-Boot runs, in your console, press any key to abort the autoboot process.
2. Set the following environment variables and run the command `saveenv` after each environment variable you set to save the settings on the board. This ensures that the setting is preserved the next time the board reboots. You can also use command `printenv` to view the current board settings.

- *ethaddr*: This variable specifies the MAC address assigned to the board. Refer to the label (sticker) on the Ethernet port (CN22) of your board to get the MAC address. For example:

```
setenv ethaddr 2e:09:0a:00:9f:95
saveenv
```

- *serverip*: This specifies the IP address to use to communicate with the TFTP server. Set this to the IP address that's assigned to your host machine that you configured the TFTP server:

```
setenv serverip your host address of the TFTP server
saveenv
```

- *ipaddr*: This specifies the IP address assigned to your board. Set this to IP address assigned to your board.

```
setenv ipaddr board IP address
saveenv
```

- *bootfile*: This specifies the image filename to be downloaded. This will be the filename of your QNX IFS on the TFTP Server.

```
setenv bootfile ifs-rcar_h3.bin
```

- *loadaddr*: The starting address in RAM to use to load the image. This should be 0x4010000

```
setenv loadaddr 0x40100000
saveenv
```

3. Run the `tftp` command to download the IFS to RAM as shown here:

```
tftp
```

4. After the transfer is done, type `go 0x40100000` to start running the IFS image:

```
go 0x40100000
```


Here's an example of what you should see as the IFS is boots:

```
System page at phys:0000000040011000 user:ffffff8040202000 kern:ffffff8040201000
Starting next program at vffffff8060088e00
MMFLAGS=1
Welcome to QNX Neutrino 7.0 on the R-Car H3 Salvator-X Board
Starting slogger and pipe servers...
Starting syslog...
Starting the session...
Starting watchdog ...
Starting Serial driver...
Starting I2C driver ...
Starting USB Host driver ...
Starting SPI Flash driver...
Unable to start "devf-spi-salvatorx" (2)
Starting Network driver...
Starting DHCP...
Starting Audio driver ...
#
```


Chapter 4

Using the Screen Graphics Subsystem

The Screen Graphics Subsystem (Screen) includes the libraries, services and tools required to run graphics applications on the board with supported graphics processors, and to manage graphics shown on the display. To use Screen, you often must configure the graphics and display. To confirm whether Screen is running properly on your target, we provide demo applications with this BSP that you can run on the target.

The following build files that are provided with this BSP don't start the Screen Graphics Subsystem (Screen) by default in the image:

- **rcar_h3.build**
- **rcar_m3.build**
- **rcar_h3ulcb.build**
- **rcar_m3ulcb.build**
- **rcar_m3n.build**

To enable graphics, use the following files to start Screen:

- **rcar_h3-graphics.build**
- **rcar_m3-graphics.build**
- **rcar_h3ulcb-graphics.build**
- **rcar_m3ulcb-graphics.build**
- **rcar_m3n-graphics.build**

Graphics configuration

The graphics configuration file is located in the following locations, based on the board that you're using:

- R-Car H3 or R-Car H3 Starter: **`$QNX_TARGET/aarch64le/usr/lib/graphics/rcarh3/graphics.conf`**
- R-Car M3-W or R-Car M3-W Starter: **`$QNX_TARGET/aarch64le/usr/lib/graphics/rcarm3/graphics.conf`**
- R-Car M3-N: **`$QNX_TARGET/aarch64le/usr/lib/graphics/rcarm3n/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 (in this case, for the R-Car H3 board :

```
# slay screen
# screen -c /usr/lib/graphics/rcarh3/graphics.conf
```

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. These are the suggested resolutions and frame rates for displays in the **graphics.conf** file:

R-Car H3 and R-Car H3 Starter boards:

- 1024 x 768 @ 60
- 1920 x 1080 @ 60
- 1280 x 720 @ 60

R-Car M3-W, R-Car M3-W Starter, or the R-Car M3-N boards:

- 1024 x 768 @ 60
- 1920 x 1080 @ 60

For example, for a 1920 x 1080, 60 Hz displays, set the `video-mode` in the **graphics.conf** file as shown here for the R-Car H3:

```
...
...
begin display 1
    formats = rgba8888, rgbx8888, rgb565, rgba5551, rgbx5551
    video-mode = 1024 x 768 @ 60
end display
...
...
begin display 2
    formats = rgba8888, rgbx8888, rgb565, rgba5551, rgbx5551
    video-mode = 1920 x 1080 @ 60
end display
...
...
begin display 3
    formats = rgba8888, rgbx8888, rgb565, rgba5551, rgbx5551 nv12
    video-mode = 1024 x 768 @ 60
end display
...
...
begin display 4
    formats = rgba8888, rgbx8888, rgb565, rgba5551, rgbx5551 nv12
    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

The video display configuration libraries are located at the following locations:

R-Car H3 and R-Car H3 Starter

- `$QNX_TARGET/aarch64le/usr/lib/graphics/rcarh3/libWFDrcar3.so`
- `$QNX_TARGET/aarch64le/usr/lib/graphics/rcarh3/libwfdcfg-salvator-x.so`

R-Car M3-W and R-Car M3-W Starter

- `$QNX_TARGET/aarch64le/usr/lib/graphics/rcarm3/libWFDrcar3.so`
- `$QNX_TARGET/aarch64le/usr/lib/graphics/rcarm3/libwfdcfg-salvator-x-m3.so`

R-Car M3-N

- `$QNX_TARGET/aarch64le/usr/lib/graphics/rcarm3n/libWFDrcar3.so`
- `$QNX_TARGET/aarch64le/usr/lib/graphics/rcarm3n/libwfdcfg-salvator-x-m3.so`

It has support for multiple displays.

Using the GPU

The boards for this BSP requires the following files to run applications that utilize the GPU. The files you use depend on the board that you're using:

R-Car H3 and R-Car H3 Starter The R-Car H3 requires the following files to run applications that utilize the GPU, which are listed in the `$BSP_ROOT_DIR/images/rcar_h3-graphics.build` or file in of this BSP:

- `/usr/lib/graphics/rcarh3/libglslcompiler.so`
- `/usr/lib/graphics/rcarh3/libIMGegl.so`
- `/usr/lib/graphics/rcarh3/libIMGGLSv1_CM.so`
- `/usr/lib/graphics/rcarh3/libIMGGLSv2.so`
- `/usr/lib/graphics/rcarh3/libPVRScopeServices.so`
- `/usr/lib/graphics/rcarh3/libpvrSCREEN_WSEGL.so`
- `/usr/lib/graphics/rcarh3/libsrv_init.so`
- `/usr/lib/graphics/rcarh3/libsrv_km.so`
- `/usr/lib/graphics/rcarh3/libsrv_um.so`
- `/usr/lib/graphics/rcarh3/libusc.so`



Depending on the settings in the `graphics.conf` file, you may require either the `/lib/dll/screen-roguetq.so` or `/lib/dll/screen-gles2blt.so` library.

R-Car M3-W and R-Car M3-W Starter

- `/usr/lib/graphics/rcarm3/libglslcompiler.so`
- `/usr/lib/graphics/rcarm3/libIMGegl.so`
- `/usr/lib/graphics/rcarm3/libIMGGLESv1_CM.so`
- `/usr/lib/graphics/rcarm3/libIMGGLESv2.so`
- `/usr/lib/graphics/rcarm3/libPVRScopeServices.so`
- `/usr/lib/graphics/rcarm3/libpvrSCREEN_WSEGL.so`
- `/usr/lib/graphics/rcarm3/libsrv_init.so`
- `/usr/lib/graphics/rcarm3/libsrv_km.so`
- `/usr/lib/graphics/rcarm3/libsrv_um.so`
- `/usr/lib/graphics/rcarm3/libusc.so`

R-Car M3-N

- `/usr/lib/graphics/rcarm3n/libglslcompiler.so`
- `/usr/lib/graphics/rcarm3n/libIMGegl.so`
- `/usr/lib/graphics/rcarm3n/libIMGGLESv1_CM.so`
- `/usr/lib/graphics/rcarm3n/libIMGGLESv2.so`
- `/usr/lib/graphics/rcarm3n/libPVRScopeServices.so`
- `/usr/lib/graphics/rcarm3n/libpvrSCREEN_WSEGL.so`
- `/usr/lib/graphics/rcarm3n/libsrv_init.so`
- `/usr/lib/graphics/rcarm3n/libsrv_km.so`
- `/usr/lib/graphics/rcarm3n/libsrv_um.so`
- `/usr/lib/graphics/rcarm3n/libusc.so`

Run Screen applications

This BSP archive comes with built-in demo applications that show how to use Screen. It's a good idea to run these demos on your target to confirm that Screen is configured properly and all the necessary drivers are available for Screen to run. If the following demo applications run successfully, it's a good indication that Screen is properly configured on your target.

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.

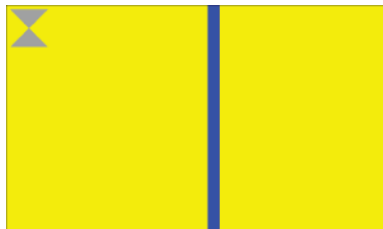


Figure 143: sw-vsync

gles2-gears

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

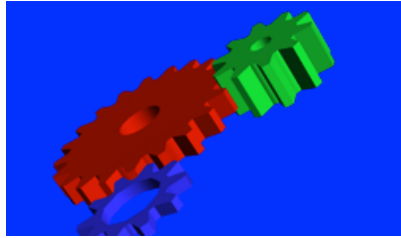


Figure 144: gles2-gears

For more information about using the demo applications or debugging them, see the “Utilities and Binaries” and the “Debugging” chapters in the *Screen Developer's Guide*, respectively.

Create Screen applications

When you create an application that supports Screen, it must link to the Screen library (**libscreen.so**). This library provides the interfaces for applications to use the Screen API. Depending on the graphical requirements of your application, you may need to link in additional libraries. Below are some of the libraries commonly used when building applications that use Screen:

libEGL

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

libGLv2

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

libimg

The interface to load and decode images. For more information, see the *Image Library Reference* guide in the QNX Software Development Platform 7.0 documentation.

For more information about application development using Screen, see the *Screen Developer's Guide*. For information about linking libraries, see “QNX C/C++ Project properties” 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, prebuilt QNX IFS images are available in the BSP's */images* directory. These prebuilt IFS images (e.g., *ifs-rcar_h3.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 IFS images.

clean

Removes all IFS images.

h3

Builds the QNX IFS (with and without graphics) for the R-Car H3 and R-Car H3 Starter boards. It also builds an IFS image with SMMU Manager (SMMUMAN) running for the R-Car H3 board.

m3

Builds the QNX IFS (with and without graphics) for the R-Car M3-W board and R-Car M3-W Starter board.

m3n

Builds the QNX IFS (with and without graphics) for the R-Car M3-N board.

ipl

Builds the QNX IPL file as an S-record formatted (**.srec**) and a raw binary (**.bin**) for the R-Car H3, R-Car H3 Starter, R-Car M3-W, R-Car M3-W Starter, and R-Car M3-N boards.



By default, this BSP builds the QNX IFS raw-formatted image (**.bin**). If you want to use an S-record formatted QNX IFS (required if you want use QNX IPL to boot an QNX IFS), you must modify the board's buildfile. For more information about how to generate an S-record formatted, see “[Build QNX IFS as a S-record formatted file](#)” in this chapter.

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_h3.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-h3**) 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_h3.bin**, it maps to the `ifs-rcar_h3.bin` target in the **Makefile**, you would type `ifs-rcar_h3.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 as shown in the figure below:

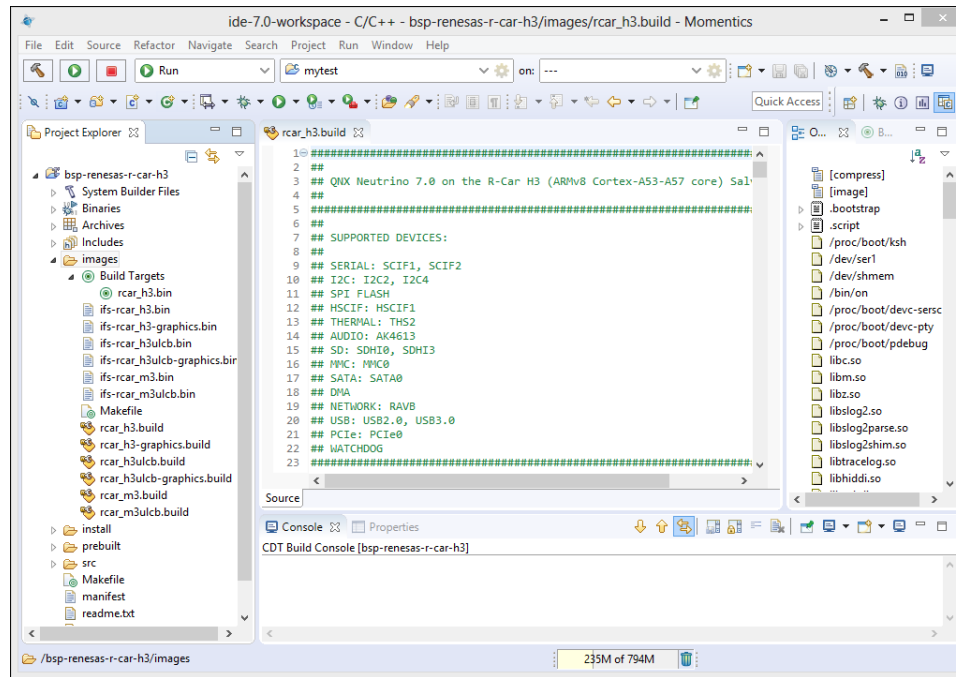


Figure 145: Build image in the IDE

Chapter 6

Driver Commands (R-Car H3)

The tables below provide a summary of driver commands for the R-Car H3 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 completely customizable and are often specific to the requirements of your system. 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 the order that you choose impacts 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 example, in the provided buildfile, the USB driver is started before the mass storage drivers because they rely on it.

- For more information on best practices for building an embedded system and optimizing boot times for your system, see the *Building Embedded Systems* and the *Boot Optimization* guides in the QNX Software Development Platform 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](#)
- [SMMU Manager](#)
- [Serial](#)
- [Inter-integrated Circuit \(I2C\)](#)
- [USB \(host mode\)](#)
- [USB function \(device mode\)](#)
- [SPI Flash](#)
- [Network](#)
- [Audio](#)
- [PCIe](#)
- [SDHI memory card and eMMC](#)
- [SATA](#)
- [Thermal](#)
- [Graphics](#)

Audio

Device	Audio
Command	<code>io-audio -c /etc/system/config/audio/io_audio.conf</code>
Required binaries	<code>io-audio</code>
Required libraries	<code>deva-ctrl-rcar-cs2000.so</code> , <code>deva-mixer-ak4613.so</code> , <code>libasound.so</code> , <code>libasound.so.2</code> , <code>libm.so.2</code> , <code>libsecpol.so</code>
Source location	<code>\$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar</code> , <code>\$BSP_ROOT_DIR/src/hardware/deva/mixer/ak4613</code>

For more information about the `deva-ctrl-rcar-cs2000.so` and `deva-mixer-ak4613.so` DLLs, see “[deva-ctrl-rcar-cs2000.so](#)” and “[deva-mixer-ak4613.so](#)”, respectively in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

Graphics

Device	GRAPHICS
Command	<code>screen -c /usr/lib/graphics/rcarh3/graphics.conf</code>
Required binaries	<code>screen</code> , <code>gles1-vsync</code> , <code>gles2-gears</code> , <code>sw-vsync</code>
Required libraries	Refer to the <i>Screen Developer's Guide</i> in the QNX SDP 7.0 documentation.
Required configuration files	<code>graphics.conf</code> and for its location in your QNX SDP 7.0 installation, see the “ Using the Screen Graphics Subsystem ” chapter in this guide.
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 (I2C2)	<code>i2c-rcar-A -p0xe6510000 -i318 -v --u2</code>
Command (I2C4)	<code>i2c-rcar-A -p0xe66D8000 -i51 -v --u4</code>
Required binaries	<code>i2c-rcar-A</code> <code>i2c-rcar-B</code>
Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/i2c</code>

For information about the `i2c_rcar-A` driver, see “[i2c-rcar-A](#).” If your board supports DVFS, you can use the `i2c_rcar-B` driver instead. For more information, see “[i2c-rcar-B](#).”

Network

Device	Ethernet
Command (user-defined MAC address)	<code>io-pkt-v6-hc -dravb mac=2e090a00830c -ptcpip pkt_typed_mem=below4G</code>
Command (generates a MAC address)	<code>sh -c "io-pkt-v6-hc -d ravb mac=genmac-random -m -p tcpip" if_up -p ravb0</code>
Command (user-defined MAC address when SMMUMAN is running)	<code>io-pkt-v6-hc -dravb mac=2e090a00830c -ptcpip pkt_typed_mem=below4G,smmu=on</code>
Command (generates a MAC address) when SMMUMAN is running	<code>sh -c "io-pkt-v6-hc -d ravb mac=genmac-random -m -p tcpip,smmu=on" if_up -p ravb0</code>
Required binaries	<code>io-pkt-v6-hc</code> , <code>ifconfig</code> , <code>dhclient</code> , <code>genmac-random</code> , <code>smmuman</code> (if SMMUMAN is running)
Required libraries	<code>devnp-ravb.so</code> , <code>smmu-rcar3.so</code> (if SMMUMAN is running)
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devnp/ravb</code>



By default, if a MAC address can be retrieved from `ethaddr` (U-Boot environment variable) when the image boots using U-Boot, that retrieved MAC address is used; otherwise a randomly generated MAC address is created.

If you want to use a static MAC address, set the `mac` option for the `devnp-ravb` command with the device's MAC address. You can also set a static MAC address but also keep the random MAC address generation as a fallback capability. To do so, you set the `ethaddr` U-Boot environment variable with the MAC address as follows:

```
U-Boot => setenv ethaddr your_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 -nw -lf /tmp/dhclient.leases ravb0`

For more information about the **devnp-ravb.so** DLL, see “[devnp-ravb.so](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

PCIe

Device	PCIe
Command	<code>pci-server --bus-scan-limit=16 -c</code>
Required binaries	<code>pci-server</code> , <code>pci-tool</code>
Required libraries	<code>libpci.so</code> , <code>pci_server-buscfg-generic.so</code> , <code>pci_hw-rcar-salvator-x.so</code> , <code>pci_debug.so</code> , <code>pci_slog.so</code> , <code>pci_strings.so</code> , <code>pci_cap-0x10.so</code> , <code>pci_cap-0x11.so</code> , <code>pci_cap-0x05.so</code> , <code>pci_cap-0x01.so</code>
Source location	Prebuilt only

SATA

Device	SATA
Command	<code>devb-rcarsata blk cache=64m cam cache rcarsata ioport=0xee300000,irq=137 disk name=sata</code>
Command when SMMUMAN is running	<code>devb-rcarsata blk cache=64m cam cache,bounce,smmu=on rcarsata ioport=0xee300000,irq=137 disk name=sata</code>
Required binaries	<code>devb-rcarsata</code> , <code>smmuman</code> (if SMMUMAN is running)
Required libraries	<code>smmu-rcar3.so</code> (if SMMUMAN is running)
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devb/rcarsata</code>



If you are using revision WS3.0 for the R-Car H3 (Salvator-XS), you must ensure that SW12, Pin 7 is set to the OFF (0) position to select SATA. For more information about the SW12 settings, see the settings for WS3.0 in the [32- and 64-bit modes for using SATA \(WS3.0\)](#) section in the [Installation Notes](#) chapter of this guide. In addition, you must set some pins using `i2c`, which is noted in the buildfile:

```
isend -n /dev/i2c4 -a 0x20 -b 50000 0x03 0x7f
isend -n /dev/i2c4 -a 0x20 -b 50000 0x01 0x7f
```

For more information about the `devb-rcarsata` driver, see “[devb-rcarsata](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

SD/MMC/eMMC

Device	SDHI memory card driver
--------	-------------------------

Command	<p>SD memory card:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp,bounce=128k mem name=below4G sdio idx=0 disk name=sd@0</pre> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp,bounce=128k mem name=below4G sdio idx=3 disk name=sd@3</pre> <p>eMMC Flash:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam bounce=128k mem name=below4G sdmmc partitions=on sdio idx=2,emmc disk name=mmc</pre> <p>Alternatively, you can choose to use more aggressive eMMC read/write performance, with this command: <code>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,~acl2 sdmmc cache=on disk name=mmc</code> For more information the risks when use aggressive eMMC read/write performance, see http://www.qnx.com/support/knowledgebase.html?id=501a0000000Mpbpr on the QNX website.</p>
Command to start eMMC and SD when SMMUMAN is running	<pre>devb-sdmmc-rcar_gen3 blk cache=64m cam bounce=128k,smmu=on mem name=below4G sdmmc partitions=on sdio idx=2,emmc sdio idx=0 sdio idx=3 disk name=sdmmc</pre>
Required binaries	<code>devb-sdmmc-rcar_gen3</code> , <code>smmuman</code> (if SMMUMAN is running)
Required libraries	<code>libcam.so</code> , <code>cam-disk.so</code> , <code>io-blk.so</code> , <code>fs-qnx6.so</code> , <code>smmu-rcar3.so</code> (if SMMUMAN is running)
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devb/sdmmc</code>

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
```

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
```

For more information the risks when use aggressive eMMC read/write performance, see <http://www.qnx.com/support/knowledgebase.html?id=501a000000Mpbr> on the QNX website.

Serial

Device	Serial drivers
Command	<code>devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif2</code>
Command when SMMUMAN is running	<code>devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 -D16 -o smmu=on scif2</code>
Required binaries	<code>devc-serscif</code> , <code>smmuman</code> (if SMMUMAN is running)
Required libraries	<code>libsocket.so</code> , <code>smmu-rcar3.so</code> (if SMMUMAN is running)
Source location	<i>\$BSP_ROOT_DIR/src/hardware/devc/serscif</i>

The SCIF port at the CN25 connector can be used to verify this debug port. Similarly, the following command can be used to launch the SCIF1 port at the CN26 connector:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif1 &
```

HSCIF shares the same pins as SCIF1. To use HSCIF, enable the initialization code `init_hscif()`, in `init_board.c`, and launch it as follows:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 hscif1 &
```

For more information about this driver, see “[devc-serscif](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter.

SMMU Manager

Device	Memory Management
Commands	<code>smmuman @/etc/rcar3-h3_r30.smmu</code>
Required binaries	<code>smmuman</code>
Required configuration file	<code>rcar3-h3_r30.smmu</code>
Required libraries	<code>smmu-rcar3.so</code>
Source location	N/A

It's recommended that you start SMMU Manager (SMMUMAN) before you start any drivers that use SMMU. Some drivers may not start until SMMU Manager has been started. You can enable SMMU Manager support for a driver using an option. SMMUMAN refers to providing DMA containment and memory-management support. For most drivers, you use the `smmu` option to enable SMMUMAN. In other cases, the `smmu` is specified with the `-o` option. To see whether your driver provides SMMU Manager support, see the documentation for the driver either in this guide or the QNX SDP 7.0 documentation.

For more information about the SMMU Manager, see the *SMMUMAN User's guide* in the QNX SDP 7.0 documentation.



It's important to note that SMMUMAN is intended for boards with Silicon WS 3.0 revisions only.

SPI Flash

Device	SPI
Command	<code>devf-rcar_qspi-gen3</code>
Command when SMMUMAN is running	<code>devf-rcar_qspi-gen3 bs=smmu=on</code>
Required binaries	<code>devf-rcar_qspi-gen3</code> , <code>smmuman</code> (if SMMUMAN is running)

Required libraries	smmu-rcar3.so (if SMMUMAN is running)
Source location	<i>\$BSP_ROOT_DIR/src/hardware/flash/boards/rcar_qspi</i>

For more information about this driver, see “[devf-rcar_qspi-gen3](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter.

Startup

Device	Startup
Command	<pre>startup-rcar_h3 -P4 -W -vvvvv startup-rcar_h3 -P4 -r0x54000000,0x3000000,1 -L0x70000000,0x10000000, codec -L0x5A200000,0x15E00000,codec_va_mem -W -vvvvv (if lossy compression is enabled in the Renesas Bootloader)</pre>
Required binaries	startup-rcar_h3
Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/startup/boards/rcar_gen3/rcar_h3</i>

In addition 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. To use this option, DRAM protection must be disabled in the Renesas Bootloader. For more information about disabling the DRAM protection in the Bootloader, see “[Build U-Boot and Renesas Loader](#)” in the “[Installation Notes](#)” chapter of this guide.

-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.

Some Renesas Bootloaders enable the lossy decompression feature for booting Linux images. This feature reserves memory area 0x54000000 to 0x56ffffff and prevents QNX Neutrino RTOS from using it as system memory. Because of this, memory faults occur. To prevent memory faults, you must use the `-r` option with the `startup-rcar_h3` command to exclude the memory range from being used by the QNX Neutrino for the `startup-rcar_h3` command. For example:

```
startup-rcar_h3 -r 0x54000000,0x3000000,1 -P4 -W -vvvvv
```

Lossy compression is enabled by default in Renesas Bootloaders v1.0.6 and v1.0.7, but is an optional feature in v1.0.9. Lossy decompression is enabled on your board if you see `NOTICE:BL2 :Lossy Decompr` areas in the BL2 logs when you boot the board as shown here:

```
NOTICE: BL2: R-Car Gen3 Initial Program Loader(CA57) Rev.1.0.9
NOTICE: BL2: PRR is R-Car H3 ES1.1
NOTICE: BL2: Boot device is HyperFlash(80MHz)
NOTICE: BL2: LCM state is CM
NOTICE: BL2: AVS setting succeeded. DVFS_SetVID=0x52
NOTICE: BL2: DDR1600(rev.0.10)
NOTICE: BL2: DRAM Split is 4ch
NOTICE: BL2: QoS is default setting(rev.0.32)
NOTICE: BL2: Lossy Decompr areas
NOTICE:      Entry 0: DCMPCAREACRAx:0x80000540 DCMPCAREACRBx:0x570
NOTICE:      Entry 1: DCMPCAREACRAx:0x40000000 DCMPCAREACRBx:0x0
NOTICE:      Entry 2: DCMPCAREACRAx:0x20000000 DCMPCAREACRBx:0x0
NOTICE: BL2: v1.1(release):3ad02ac
NOTICE: BL2: Built : 13:03:52, Sep 20 2016
NOTICE: BL2: Normal boot
NOTICE: BL2: dst=0xe631a208 src=0x8180000 len=512(0x200)
NOTICE: BL2: dst=0x43f00000 src=0x8180400 len=6144(0x1800)
NOTICE: BL2: dst=0x44000000 src=0x81c0000 len=65536(0x10000)
NOTICE: BL2: dst=0x44100000 src=0x8200000 len=524288(0x80000)
NOTICE: BL2: dst=0x49000000 src=0x8640000 len=1048576(0x100000)
```

Thermal Driver

The driver used for thermal management on the board.

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

For more information about this driver, see “[rcar-thermal](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter.

USB (host mode)

Device	USB host
Command (USB)	<pre>io-usb-otg -d xhci ioport=0xEE000000,irq=0x86 -d ehci ioport=0xEE080100,irq=0x8C,ioport=0xEE0A0100, irq=0x90,ioport=0xEE0C0100,irq=0x91 -d ohci ioport=0xEE080000,irq=0x8C,ioport=0xEE0A0000 irq=0x90,ioport=0xEE0C0000,irq=0x91 (for Silicon revisions WS1.0 and WS1.1) io-usb-otg -d xhci ioport=0xEE000000,irq=0x86 -d ehci ioport=0xEE080100,irq=0x8C,ioport=0xEE0A0100, irq=0x90,ioport=0xEE0C0100,irq=0x91,ioport=0xEE0E0100,irq=0x44 -d ohci ioport=0xEE080000,irq=0x8C,ioport=0xEE0A0000, irq=0x90,ioport=0xEE0C0000,irq=0x91,ioport=0xEE0E0000,irq=0x44 (for Silicon revision WS2.0 and WS3.0 boards) devb-umass blk noatime,commit=none,cache=10m cam pnp mem name=/memory/below4G</pre>
Command (USB) with DMA-enabled	<pre>io-usb-otg -s smmu=on -d xhci ioport=0xEE000000,irq=0x86 -d ehci ioport=0xEE080100,irq=0x8C,ioport=0xEE0A0100, irq=0x90,ioport=0xEE0C0100,irq=0x91 -d ohci ioport=0xEE080000,irq=0x8C,ioport=0xEE0A0000 irq=0x90,ioport=0xEE0C0000,irq=0x91 (for Silicon revisions WS1.0 and WS1.1) io-usb-otg -s smmu=on -d xhci ioport=0xEE000000,irq=0x86 -d ehci ioport=0xEE080100,irq=0x8C,ioport=0xEE0A0100, irq=0x90,ioport=0xEE0C0100,irq=0x91,ioport=0xEE0E0100,irq=0x44 -d ohci ioport=0xEE080000,irq=0x8C,ioport=0xEE0A0000, irq=0x90,ioport=0xEE0C0000,irq=0x91,ioport=0xEE0E0000,irq=0x44 (for Silicon revision WS2.0 and WS3.0 boards) devb-umass blk noatime,commit=none,cache=10m cam pnp mem name=/memory/below4G</pre>
Required binaries	io-usb-otg, devb-umass, smmuman (if SMMUMAN is enabled)
Required libraries	devu-hcd-ehci.so, devu-hcd-ohci.so, devu-hcd-xhci.so, smmu-rcar3.so (if SMMUMAN is enabled)
Source location	Prebuilt only

USB function (device mode)

You can have the USB function driver as mass storage device, for a CDC-ACM (serial) connection, or a Ethernet over USB connection. Make sure that the GPIOs are configured appropriately for this use. See the board manufacturer's documentation for the correct switch settings to Host mode.



To use the USB Function on CN9 you must make sure SW15 is in the middle position.

Device	USB OTG (Device mode)
Command (Mass storage)	<code>io-usb-otg -d usbumass-hsusb-rcar ioport=0xe6590000,irq=139</code>
Command (CDC-ACM (serial) device)	<code>io-usb-otg -d usbser-hsusb-rcar ioport=0xe6590000,irq=139</code>
Command NCM device (Ethernet over USB)	<code>io-usb-otg -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139</code>
Command (Mass storage) when SMMUMAN is running	<code>io-usb-otg -s smmu=on -d usbumass-hsusb-rcar ioport=0xe6590000,irq=139</code>
Command (CDC-ACM (serial) device) when SMMUMAN is running	<code>io-usb-otg -s smmu=on -d usbser-hsusb-rcar ioport=0xe6590000,irq=139</code>
Command NCM device (Ethernet over USB) when SMMUMAN is running	<code>io-usb-otg -s smmu=on -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139</code>
Required binaries	<code>io-usb-otg</code> , <code>ulink_ctrl</code> , <code>devu-umass_client-block</code> <code>devc-serusb_dcd</code> , <code>ifconfig</code> , <code>if_up</code> , <code>smmuman</code> (if SMMUMAN is running)
Required libraries	<code>libusbdc1.so</code> , <code>devnp-usbdnet.so</code> , <code>devu-dcd-hsusb-rcar.so</code> , <code>devu-dcd-usbncm-hsusb-rcar.so</code> , <code>devu-dcd-usbser-hsusb-rcar.so</code> , <code>devu-dcd-usbumass-hsusb-rcar.so</code> , <code>smmu-rcar3.so</code> (if SMMUMAN is running)
Source location	Prebuilt only



CAUTION: If you want a specific USB port (specified using the `ioport` option) to be started as your USB device controller, ensure that you don't start that port as a USB host controller. In other words, if you wanted to start `ioport=0xe6590000, irq=139` as a device controller, don't specify the same USB port when you start the USB host controller driver.

Example of Mass Storage

1. When your target board is ready, create and format a RAM disk to emulate a mass storage device:

```
devb-ram ram capacity=204800 nodinit disk name=ramdisk blk cache=512k
fdisk /dev/ramdisk0 add -t 12
mount -e /dev/ramdisk0
waitfor /dev/ramdisk0t12
mkdosfs -F32 /dev/ramdisk0t12
mount -t dos /dev/ramdisk0t12 /ramdisk
```

2. Launch the USB device stack:

```
io-usb-otg -d usbumass-hsusb-rcar ioport=0xe6590000,irq=139
```

3. Start the Mass Storage function driver, and enable USB soft connect:

```
devu-umass_client-block -l lun=0,fname=/dev/ramdisk0t12
ulink_ctrl -ll
```

4. Connect the R-Car H3 board USB OTG port to your host computer to be detected as a mass storage device.

Example of CDC-ACM (serial) device

1. Start USB device stack:

```
io-usb-otg -d usbser-hsusb-rcar ioport=0xe6590000,irq=139
waitfor /dev/usb-dcd/io-usb-otg 4
```

2. Start the USB CDC-ACM function driver, and enable USB soft connect:

```
devc-serusb_dcd -e -v -F -s -d iface_list=0
waitfor /dev/serusb1
ulink_ctrl -ll
```

Example of NCM device (Ethernet over USB)

1. Start USB device stack:

```
io-usb-otg -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139
waitfor /dev/usb/io-usb-otg 4
```

2. Start USB NCM function driver and enable USB soft connections:

```
mount -Tio-pkt -o protocol=ncm,mac=020022446688 devnp-usbdnet.so
if_up -p ncm0
ifconfig ncm0 192.168.0.1
ulink_ctrl -ll
```

For more information about the **devu-dcd-usbncm-hsusb-rcar.so**, **devu-dcd-usbser-hsusb-rcar.so** or **dcd-usbumass-hsusb-rcar.so** drivers, see “[devu-dcd-usbncm-hsusb-rcar.so](#)”, “[devu-dcd-usbser-hsusb-rcar.so](#)” and “[devu-dcd-usbumass-hsusb-rcar.so](#)”, respectively 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_h3 -P4 -W -vvvvv
```

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

Device	WATCHDOG
Command	<code>wdtkick -W 0x0:0x5A5AFF00</code>
Required binaries	<code>wdtkick</code>
Required libraries	N/A
Source location	<code><i>\$BSP_ROOT_DIR</i>/src/hardware/support/wdtkick</code>

Chapter 7

Driver Commands (R-Car H3 Starter)

The tables below provide a summary of driver commands for the R-Car H3 Starter.

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 completely customizable and are often specific to the requirements of your system. 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 the order that you choose impacts 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 example, in the provided buildfile, the USB driver is started before the mass storage drivers because they rely on it.

- For more information on best practices for building an embedded system and optimizing boot times for your system, see the *Building Embedded Systems* and the *Boot Optimization* guides in the QNX Software Development Platform 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 \(I2C\)](#)
- [USB \(host mode\)](#)
- [SPI Flash](#)
- [Network](#)
- [Audio](#)
- [PCIe](#)
- [SDHI memory card and eMMC](#)
- [Thermal](#)
- [Graphics](#)

Audio

Device	Audio
Command	<code>io-audio -c /etc/system/config/audio/io_audio.conf</code>
Required binaries	<code>io-audio</code>
Required libraries	<code>deva-ctrl-rcar-cs2000.so</code> , <code>deva-mixer-ak4613.so</code> , <code>libasound.so</code> , <code>libasound.so.2</code> , <code>libm.so.2</code> , <code>libsecpol.so</code>
Source location	<code>\$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar</code> , <code>\$BSP_ROOT_DIR/src/hardware/deva/mixer/ak4613</code>

For more information about the `deva-ctrl-rcar-cs2000.so` and `deva-mixer-ak4613.so` DLLs, see “[deva-ctrl-rcar-cs2000.so](#)” and “[deva-mixer-ak4613.so](#)”, respectively in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

Graphics

Device	GRAPHICS
Command	<code>screen -c /usr/lib/graphics/rcarh3/graphics.conf</code>
Required binaries	<code>screen</code> , <code>gles1-vsync</code> , <code>gles2-gears</code> , <code>sw-vsync</code>
Required libraries	Refer to the <i>Screen Developer's Guide</i> in the QNX SDP 7.0 documentation.
Required configuration files	<code>graphics.conf</code> and for its location in your QNX SDP 7.0 installation, see the “ Using the Screen Graphics Subsystem ” chapter in this guide.
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	I2C2 and I2C4
Command (I2C2)	<code>i2c-rcar-A -p0xe6510000 -i318 -v --u2</code>
Command (I2C4)	<code>i2c-rcar-A -p0xe66D8000 -i51 -v --u4</code>
Required binaries	<code>i2c-rcar-A</code> <code>i2c-rcar-B</code>
Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/i2c</code>

For information about the `i2c_rcar-A` driver, see “[i2c-rcar-A](#).” If your board supports DVFS, you can use the `i2c_rcar-B` driver instead. For more information, see “[i2c-rcar-B](#).”

Network

Device	Ethernet
Command (user-defined MAC address)	<code>io-pkt-v6-hc -dravb mac=2e090a00830c -ptcpip pkt_typed_mem=below4G</code>
Command (generates a MAC address)	<code>sh -c "io-pkt-v6-hc -d ravb mac=`genmac-random -m` -p tcpip pkt_typed_mem=below4G" if_up -p ravb0</code>
Required binaries	<code>io-pkt-v6-hc, ifconfig, dhclient, genmac-random</code>
Required libraries	devnp-ravb.so
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devnp/ravb</code>



By default, if a MAC address can be retrieved from `ethaddr` (U-Boot environment variable) when the image boots using U-Boot, that retrieved MAC address is used; otherwise a randomly generated MAC address is created.

If you want to use a static MAC address, set the `mac` option for the `devnp-ravb` command with the device's MAC address. You can also set a static MAC address but also keep the random MAC address generation as a fallback capability. To do so, you set the `ethaddr` U-Boot environment variable with the MAC address as follows:

```
U-Boot => setenv ethaddr your_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 -nw -lf /tmp/dhclient.leases ravb0`

For more information about the **devnp-ravb.so** DLL, see “[devnp-ravb.so](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

PCIe

Device	PCIe
Command	<code>pci-server --bus-scan-limit=16 -c</code>
Required binaries	<code>pci-server, pci-tool</code>

Required libraries	<code>libpci.so</code> , <code>pci_server-buscfg-generic.so</code> , <code>pci_hw-rcar-salvator-x.so</code> , <code>pci_debug.so</code> , <code>pci_slog.so</code> , <code>pci_strings.so</code> , <code>pci_cap-0x10.so</code> , <code>pci_cap-0x11.so</code> , <code>pci_cap-0x05.so</code> , <code>pci_cap-0x01.so</code>
Source location	Prebuilt only

SD/MMC/eMMC

Device	SDHI memory card driver
Command	<p>SD memory card:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp,bounce=128k mem name=below4G sdio idx=0,bs=nowp disk name=sd</pre> <p>eMMC Flash:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam bounce=128k mem name=below4G sdio idx=2,emmc disk name=mmc</pre> <p>Alternatively, you can choose to use more aggressive eMMC read/write performance, with this command: <code>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</code> 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.</p>
Required binaries	<code>devb-sdmmc-rcar_gen3</code>
Required libraries	<code>libcam.so</code> , <code>cam-disk.so</code> , <code>io-blk.so</code> , <code>fs-qnx6.so</code>
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devb/sdmmc</code>

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
```

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
```

For more information the risks when use aggressive eMMC read/write performance, see <http://www.qnx.com/support/knowledgebase.html?id=501a000000Mpbr> on the QNX website.

Serial

Device	Serial drivers
Command	<code>devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif2</code>
Required binaries	<code>devc-serscif</code>
Required libraries	libsocket.so
Source location	<i>\$BSP_ROOT_DIR/src/hardware/devc/serscif</i>

The SCIF port at the CN25 connector can be used to verify this debug port. Similarly, the following command can be used to launch the SCIF1 port at the CN26 connector:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif1 &
```

HSCIF shares the same pins as SCIF1. To use HSCIF, enable the initialization code `init_hscif()`, in `init_board.c`, and launch it as follows:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 hscif1 &
```

For more information about this driver, see “[devc-serscif](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter.

SPI Flash

Device	SPI
Command	devf-rcar_qspi-gen3
Required binaries	devf-rcar_qspi-gen3
Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/flash/boards/rcar_qspi</i>

For more information about this driver, see “[devf-rcar_qspi-gen3](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter.

Startup

Device	Startup
Command	<pre>startup-rcar_h3 -P4 -W -vvvvv startup-rcar_h3 -P4 -r0x54000000,0x30000000,1 -L0x70000000,0x10000000, codec -L0x5A200000,0x15E00000,codec_va_mem -W -vvvvv (if lossy compression is enabled in the Renesas Bootloader)</pre>
Required binaries	startup-rcar_h3
Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/startup/boards/rcar_gen3/rcar_h3</i>

In addition 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. To use this option, DRAM protection must be disabled in the Renesas Bootloader. For more information about disabling the DRAM protection in the Bootloader, see “[Build U-Boot and Renesas Loader](#)” in the “[Installation Notes](#)” chapter of this guide.

-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.

Some Renesas Bootloaders enable the lossy decompression feature for booting Linux images. This feature reserves memory area 0x54000000 to 0x56ffffff and prevents QNX Neutrino RTOS from using it as system memory. Because of this, memory faults occur. To prevent memory faults, you must use the **-r** option with the `startup-rcar_h3` command to exclude the memory range from being used by the QNX Neutrino for the `startup-rcar_h3` command. For example:

```
startup-rcar_h3 -r 0x54000000,0x3000000,1 -P4 -W -vvvvv
```

Lossy compression is enabled by default in Renesas Bootloaders v1.0.6 and v1.0.7, but is an optional feature in v1.0.9. Lossy decompression is enabled on your board if you see `NOTICE:BL2 :Lossy Decompr` areas in the BL2 logs when you boot the board as shown here:

```
NOTICE: BL2: R-Car Gen3 Initial Program Loader(CA57) Rev.1.0.9
NOTICE: BL2: PRR is R-Car H3 Starter ES1.1
NOTICE: BL2: Boot device is HyperFlash(80MHz)
NOTICE: BL2: LCM state is CM
NOTICE: BL2: AVS setting succeeded. DVFS_SetVID=0x52
NOTICE: BL2: DDR1600(rev.0.10)
NOTICE: BL2: DRAM Split is 4ch
NOTICE: BL2: QoS is default setting(rev.0.32)
NOTICE: BL2: Lossy Decomp areas
NOTICE:      Entry 0: DCMPCAREACRAx:0x80000540 DCMPCAREACRBx:0x570
NOTICE:      Entry 1: DCMPCAREACRAx:0x40000000 DCMPCAREACRBx:0x0
NOTICE:      Entry 2: DCMPCAREACRAx:0x20000000 DCMPCAREACRBx:0x0
NOTICE: BL2: v1.1(release):3ad02ac
NOTICE: BL2: Built : 13:03:52, July 1 2017
NOTICE: BL2: Normal boot
NOTICE: BL2: dst=0xe631a208 src=0x8180000 len=512(0x200)
NOTICE: BL2: dst=0x43f00000 src=0x8180400 len=6144(0x1800)
NOTICE: BL2: dst=0x44000000 src=0x81c0000 len=65536(0x10000)
NOTICE: BL2: dst=0x44100000 src=0x8200000 len=524288(0x80000)
NOTICE: BL2: dst=0x49000000 src=0x8640000 len=1048576(0x100000)
```

Thermal Driver

The driver used for thermal management on the board.

Device	Thermal driver
--------	----------------

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

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

USB (host mode)

Device	USB host
Command (USB)	<code>io-usb-otg -d ehci ioport=0xEE0A0100,irq=0x90 -d ohci ioport=0xEE0A0000,irq=0x90</code> <code>devb-umass blk cache=10m cam pnp</code> Alternatively, you can choose to use more aggressive mass storage read/write performance, with this command: <code>devb-umass blk maxio=256,cache=10m,ra=256k:512k,rapolicy=aggressive,commit=none cam cache,async,pnp</code> For more information the risks when use aggressive read/write performance, see http://www.qnx.com/support/knowledgebase.html?id=501a0000000Mpbr on the QNX website.
Required binaries	<code>io-usb-otg</code> , <code>devb-umass</code>
Required libraries	<code>devu-hcd-ehci.so</code>, <code>devu-hcd-ohci.so</code>, <code>libusbdi.so</code>, <code>libhiddi.so</code>
Source location	Prebuilt only

Watchdog

To enable the watchdog:

1. Modify the buildfile so that `startup` launches with the `-W` option:
`startup-rcar_h3 -P4 -W -vvvvv`
2. Launch the watchdog timer utility early on in the boot script

Device	WATCHDOG
Command	<code>wdtkick -W 0x0:0x5A5AFF00</code>
Required binaries	<code>wdtkick</code>

Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/support/wdtkick</i>

Chapter 8

Driver Commands (R-Car M3-W)

The tables below provide a summary of driver commands for the R-Car M3-W 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 completely customizable and are often specific to the requirements of your system. 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 the order that you choose impacts 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 example, in the provided buildfile, the USB driver is started before the mass storage drivers because they rely on it.

- For more information on best practices for building an embedded system and optimizing boot times for your system, see the *Building Embedded Systems* and the *Boot Optimization* guides in the QNX Software Development Platform 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 \(I2C\)](#)
- [USB \(host mode\)](#)
- [USB function \(device mode\)](#)
- [SPI Flash](#)
- [Network](#)
- [Audio](#)
- [SDHI memory card and eMMC](#)
- [PCIe](#)
- [SATA](#)
- [Thermal](#)
- [Graphics](#)

Audio

Device	Audio
Command	<code>io-audio -c /etc/system/config/audio/io_audio.conf</code>
Required binaries	<code>io-audio</code> , <code>mix_ctl</code> , <code>wave</code> , <code>waverec</code>
Required libraries	<code>deva-ctrl-rcar-cs2000.so</code> , <code>deva-mixer-ak4613.so</code> , <code>libasound.so</code> , <code>libasound.so.2</code> , <code>libm.so.2</code> , <code>deva-util-restore.so</code> , <code>libsecpol.so</code>
Source location	<code>\$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar</code> , <code>\$BSP_ROOT_DIR/src/hardware/deva/mixer/ak4613</code>

For more information about the `deva-ctrl-rcar-cs2000.so` and `deva-mixer-ak4613.so` DLLs, see “[deva-ctrl-rcar-cs2000.so](#)” and “[deva-mixer-ak4613.so](#)”, respectively in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

Graphics

Device	GRAPHICS
Command	<code>screen -c /usr/lib/graphics/rcarm3/graphics.conf</code>
Required binaries	<code>screen</code> , <code>gles1-vsync</code> , <code>gles2-gears</code> , <code>sw-vsync</code>
Required libraries	Refer to the <i>Screen Developer's Guide</i> in the QNX SDP 7.0 documentation.
Required configuration files	<code>graphics.conf</code> and for its location in your QNX SDP 7.0 installation, see the “ Using the Screen Graphics Subsystem ” chapter in this guide.
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 (I2C2)	<code>i2c-rcar-A -p0xe6510000 -i318 -v --u2</code>
Command (I2C4)	<code>i2c-rcar-A -p0xe66D8000 -i51 -v --u4</code>
Required binaries	<code>i2c-rcar-A</code> <code>i2c-rcar-B</code>
Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/i2c</code>

For information about the `i2c_rcar-A` driver, see “[i2c-rcar-A](#).” If your board supports DVFS, you can use the `i2c_rcar-B` driver instead. For more information, see “[i2c-rcar-B](#).”

Network

Device	Ethernet
Command (user-defined MAC address)	<code>io-pkt-v6-hc -dravb mac=2e090a00830c -ptcpip pkt_typed_mem=below4G</code>
Command (generates a MAC address)	<code>sh -c "sh -c "io-pkt-v6-hc -d ravb mac=`genmac-random -m` -p tcpip pkt_typed_mem=below4G"" if_up -p ravb0</code>
Required binaries	<code>io-pkt-v6-hc, ifconfig, dhclient, genmac-random</code>
Required libraries	devnp-ravb.so
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devnp/ravb</code>



By default, if a MAC address can be retrieved from `ethaddr` (U-Boot environment variable) when the image boots using U-Boot, that retrieved MAC address is used; otherwise a randomly generated MAC address is created.

If you want to use a static MAC address, set the `mac` option for the `devnp-ravb` command with the device's MAC address. You can also set a static MAC address but also keep the random MAC address generation as a fallback capability. To do so, you set the `ethaddr` U-Boot environment variable with the MAC address as follows:

```
U-Boot => setenv ethaddr your_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 -nw -lf /tmp/dhclient.leases ravb0`

For more information about the **devnp-ravb.so** DLL, see “[devnp-ravb.so](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

PCIe

Device	PCIe
Command	<code>pci-server --bus-scan-limit=16 -c</code>
Required binaries	<code>pci-server, pci-tool</code>


Required libraries	<code>libpci.so</code> , <code>pci_server-buscfg-generic.so</code> , <code>pci_hw-rcar-salvator-x.so</code> , <code>pci_debug.so</code> , <code>pci_slog.so</code> , <code>pci_strings.so</code> , <code>pci_cap-0x10.so</code> , <code>pci_cap-0x11.so</code> , <code>pci_cap-0x05.so</code> , <code>pci_cap-0x01.so</code>
Source location	Prebuilt only

SATA

Device	SATA
Command	<code>devb-rcarsata blk cache=64m cam cache rcarsata ioport=0xee300000,irq=137 disk name=sata</code>
Required binaries	<code>devb-rcarsata</code>
Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devb/rcarsata</code>

For more information about the `devb-rcarsata` driver, see “[devb-rcarsata](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

SD/MMC/eMMC

Device	SDHI memory card driver
Command	<p>SD memory card:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp, bounce=128k sdio idx=0</pre> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp, bounce=128k sdio idx=3</pre> <p>eMMC Flash:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam bounce=128k mem name=below4G sdio idx=2,emmc disk name=mmc</pre> <p>Alternatively, you can choose to use more aggressive eMMC read/write performance, with this command: <code>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</code> For more information the risks when use aggressive (volatile cache is enabled) read/write performance, see http://www.qnx.com/support/knowledgebase.html?id=501a00000000Mpbpr on the QNX website.</p> <hr/> <p> A reset with volatile cache enabled causes the Power-Safe filesystem to become read-only. Resets can occur when there's incorrect system tuning and unstable drivers.</p> <hr/>
Required binaries	<code>devb-sdmmc-rcar_gen3</code>

Required libraries	libcam.so, cam-disk.so, io-blk.so, fs-qnx6.so
Source location	<i>\$BSP_ROOT_DIR/src/hardware/devb/sdmmc</i>

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
```

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
```

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.

Serial

Device	Serial drivers
Command	<code>devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif2</code>
Required binaries	<code>devc-serscif</code>
Required libraries	libsocket.so
Source location	<i>\$BSP_ROOT_DIR/src/hardware/devc/serscif</i>

The SCIF port at the CN25 connector can be used to verify this debug port. Similarly, the following command can be used to launch the SCIF1 port at the CN26 connector:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif1 &
```

HSCIF shares the same pins as SCIF1. To use HSCIF, enable the initialization code `init_hscif()`, in `init_board.c`, and launch it as follows:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 hscif1 &
```

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

SPI Flash

Device	SPI
Command	<code>devf-rcar_qspi-gen3</code>
Required binaries	<code>devf-rcar_qspi-gen3</code>
Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/flash/boards/rcar_qspi</i>

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

Startup

Device	Startup
Command	<code>startup-rcar_m3 -P2 -r0x54000000,0x3000000,1 -W -vvvvv</code>
Required binaries	<code>startup-rcar_m3</code>

Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/startup/boards/rcar_gen3/rcar_m3</code>

In addition 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. To use this option, DRAM protection must be disabled in the Renesas Bootloader. For more information about disabling the DRAM protection in the Bootloader, see “[Build U-Boot and Renesas Loader](#)” in the “[Installation Notes](#)” chapter of this guide.

-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.

Some Renesas Bootloaders enable the lossy decompression feature for booting Linux images. This feature reserves memory area 0x54000000 to 0x56ffffff and prevents QNX Neutrino RTOS from using it as system memory. Because of this, memory faults occur. To prevent memory faults, you must use the `-r` option with the `startup-rcar_m3` command to exclude the memory range from being used by the QNX Neutrino for the `startup-rcar_m3` command. For example:

```
startup-rcar_m3 -P2 -r 0x54000000,0x3000000,1 -W -vvvvv
```

Lossy compression is enabled by default in Renesas Bootloaders v1.0.6 and v1.0.7, but is an optional feature in v1.0.9. Lossy decompression is enabled on your board if you see `NOTICE:BL2 :Lossy Decompr` areas in the BL2 logs when you boot the board as shown here:

```
NOTICE: BL2: R-Car Gen3 Initial Program Loader(CA57) Rev.1.0.9
NOTICE: BL2: PRR is R-Car M3 ES1.1
NOTICE: BL2: Boot device is HyperFlash(80MHz)
NOTICE: BL2: LCM state is CM
NOTICE: BL2: AVS setting succeeded. DVFS_SetVID=0x52
NOTICE: BL2: DDR1600(rev.0.10)
NOTICE: BL2: DRAM Split is 4ch
NOTICE: BL2: QoS is default setting(rev.0.32)
NOTICE: BL2: Lossy Decompr areas
NOTICE: Entry 0: DCMPCAREACRAx:0x80000540 DCMPCAREACRBx:0x570
NOTICE: Entry 1: DCMPCAREACRAx:0x40000000 DCMPCAREACRBx:0x0
```

```

NOTICE:      Entry 2: DCMPCAREACRAx:0x20000000 DCMPCAREACRBx:0x0
NOTICE: BL2: v1.1 (release):3ad02ac
NOTICE: BL2: Built : 13:03:52, July 1, 2017
NOTICE: BL2: Normal boot
NOTICE: BL2: dst=0xe631a208 src=0x8180000 len=512 (0x200)
NOTICE: BL2: dst=0x43f00000 src=0x8180400 len=6144 (0x1800)
NOTICE: BL2: dst=0x44000000 src=0x81c0000 len=65536 (0x10000)
NOTICE: BL2: dst=0x44100000 src=0x8200000 len=524288 (0x80000)
NOTICE: BL2: dst=0x49000000 src=0x8640000 len=1048576 (0x100000)

```

Thermal Driver

The driver used for thermal management on the board.

Device	Thermal driver
Command	<pre>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)</pre>
Required binaries	rcar-thermal
Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/support/rcar-thermal</i>

For more information about this driver, see “[rcar-thermal](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter.

USB (host mode)

Device	USB host
Command (USB)	<pre>io-usb-otg -d xhci ioport=0xEE000000,irq=0x86 -d ehci ioport=0xEE080100,irq=0x8C,ioport=0xEE0A0100,irq=0x90,ioport=0xEE0C0100 -d ohci ioport=0xEE080000,irq=0x8C,ioport=0xEE0A0000, irq=0x90,ioport=0xEE0C0000,irq=0x91 devb-umass blk cache=10m cam pnp</pre>
Required binaries	io-usb-otg, devb-umass
Required libraries	<i>devu-hcd-ehci.so, devu-hcd-ohci.so, devu-hcd-xhci.so</i>
Source location	Prebuilt only

USB function (device mode)

You can have the USB function driver as mass storage device, for a CDC-ACM (serial) connection, or a Ethernet over USB connection. Make sure the GPIOs are configured appropriately for this use. See the board manufacturer's documentation for the correct switch settings to Host mode.



To use the USB Function on CN9 you must make sure SW15 is in the middle position.

Device	USB OTG (Device mode)
Command (Mass storage)	<code>io-usb-otg -d usbumass-hsusb-rcar ioport=0xe6590000,irq=139</code>
Command (CDC-ACM (serial) device)	<code>io-usb-otg -d usbser-hsusb-rcar ioport=0xe6590000,irq=139</code>
Command NCM device (Ethernet over USB)	<code>io-usb-otg -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139</code>
Required binaries	<code>io-usb-otg, ulink_ctrl, devu-umass_client-block devc-serusb_dcd, ifconfig, if_up</code>
Required libraries	<code>libusbdc1.so, devnp-usbdnet.so, devu-dcd-hsusb-rcar.so, devu-dcd-usbncm-hsusb-rcar.so, devu-dcd-usbser-hsusb-rcar.so, devu-dcd-usbumass-hsusb-rcar.so</code>
Source location	Prebuilt only



CAUTION: If you want a specific USB port (specified using the `ioport` option) to be started as your USB device controller, ensure that you don't start that port as a USB host controller. In other words, if you wanted to start `ioport=0xe6590000, irq=139` as a device controller, don't specify the same USB port when you start the USB host controller driver.

Example of Mass Storage

1. When your target board is ready, create and format a RAM disk to emulate a mass storage device:

```
devb-ram ram capacity=204800 nodinit disk name=ramdisk blk cache=512k
fdisk /dev/ramdisk0 add -t 12
mount -e /dev/ramdisk0
waitfor /dev/ramdisk0t12
mkdosfs -F32 /dev/ramdisk0t12
mount -t dos /dev/ramdisk0t12 /ramdisk
```

2. Launch the USB device stack:

```
io-usb-otg -d usbumass-hsusb-rcar ioport=0xe6590000,irq=139
```

3. Start the Mass Storage function driver, and enable USB soft connect:

```
devu-umass_client-block -l lun=0,fname=/dev/ramdisk0t12
ulink_ctrl -ll
```

4. Connect the R-Car H3 board USB OTG port (CN9) to your host computer to be detected as a mass storage device.

Example of CDC-ACM (serial) device

1. Start USB device stack:

```
io-usb-otg -d usbser-hsusb-rcar ioport=0xe6590000,irq=139
waitfor /dev/usb-dcd/io-usb-otg 4
```

2. Start the USB CDC-ACM function driver, and enable USB soft connect:

```
devc-serusb_dcd -e -v -F -s -d iface_list=0
waitfor /dev/serusb1
ulink_ctrl -ll
```

Example of NCM device (Ethernet over USB)

1. Start USB device stack:

```
io-usb-otg -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139
waitfor /dev/usb/io-usb-otg 4
```

2. Start USB NCM function driver and enable USB soft connections:

```
mount -Tio-pkt -o protocol=ncm,mac=020022446688 devnp-usbdnet.so
if_up -p ncm0
ifconfig ncm0 192.168.0.1
ulink_ctrl -ll
```

For more information about the **devu-dcd-usbncm-hsusb-rcar.so**, **devu-dcd-usbser-hsusb-rcar.so** or **dcd-usbumass-hsusb-rcar.so** drivers, see “[devu-dcd-usbncm-hsusb-rcar.so](#)”, “[devu-dcd-usbser-hsusb-rcar.so](#)” and “[devu-dcd-usbumass-hsusb-rcar.so](#)”, respectively 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_m3 -P2 -r 0x54000000,0x3000000,1 -W -vvvvv
```

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

Device	WATCHDOG
Command	wdtkick -W 0x0:0x5A5AFF00
Required binaries	wdtkick
Required libraries	N/A

Source location	<i>\$BSP_ROOT_DIR/src/hardware/support/wdtkick</i>
-----------------	---

Chapter 9

Driver Commands (R-Car M3-W Starter)

The tables below provide a summary of driver commands for the R-Car M3-W Starter 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 completely customizable and are often specific to the requirements of your system. 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 the order that you choose impacts 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 example, in the provided buildfile, the USB driver is started before the mass storage drivers because they rely on it.

- For more information on best practices for building an embedded system and optimizing boot times for your system, see the *Building Embedded Systems* and the *Boot Optimization* guides in the QNX Software Development Platform 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 \(I2C\)](#)
- [USB \(host mode\)](#)
- [SPI Flash](#)
- [Network](#)
- [Audio](#)
- [SDHI memory card and eMMC](#)
- [PCIe](#)
- [Thermal](#)
- [Graphics](#)

Audio

Device	Audio
Command	<code>io-audio -c /etc/system/config/audio/io_audio.conf</code>
Required binaries	<code>io-audio</code>
Required libraries	<code>deva-ctrl-rcar-cs2000.so</code> , <code>deva-mixer-ak4613.so</code> , <code>libasound.so</code> , <code>libasound.so.2</code> , <code>libm.so.2</code> , <code>libsecpol.so</code>
Source location	<code>\$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar</code> , <code>\$BSP_ROOT_DIR/src/hardware/deva/mixer/ak4613</code>

For more information about the `deva-ctrl-rcar-cs2000.so` and `deva-mixer-ak4613.so` DLLs, see “[deva-ctrl-rcar-cs2000.so](#)” and “[deva-mixer-ak4613.so](#)”, respectively in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

Graphics

Device	GRAPHICS
Command	<code>screen -c /usr/lib/graphics/rcarm3/graphics.conf</code>
Required binaries	<code>screen</code> , <code>gles1-vsync</code> , <code>gles2-gears</code> , <code>sw-vsync</code>
Required libraries	Refer to the <i>Screen Developer's Guide</i> in the QNX SDP 7.0 documentation.
Required configuration files	<code>graphics.conf</code> and for its location in your QNX SDP 7.0 installation, see the “ Using the Screen Graphics Subsystem ” chapter in this guide.
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	I2C2 and I2C4
Command (I2C2)	<code>i2c-rcar-A -p0xe6510000 -i318 -v --u2</code>
Command (I2C4)	<code>i2c-rcar-A -p0xe66D8000 -i51 -v --u4</code>
Required binaries	<code>i2c-rcar-A</code> <code>i2c-rcar-B</code>
Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/i2c</code>

For information about the `i2c_rcar-A` driver, see “[i2c-rcar-A](#).” If your board supports DVFS, you can use the `i2c_rcar-B` driver instead. For more information, see “[i2c-rcar-B](#).”

Network

Device	Ethernet
Command (user-defined MAC address)	<code>io-pkt-v6-hc -dravb mac=2e090a00830c -ptcpip pkt_typed_mem=below4G</code>
Command (generates a MAC address)	<code>sh -c "io-pkt-v6-hc -d ravb mac=`genmac-random -m` -p tcpip pkt_typed_mem=below4G" if_up -p ravb0</code>
Required binaries	<code>io-pkt-v6-hc, ifconfig, dhclient, genmac-random</code>
Required libraries	devnp-ravb.so
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devnp/ravb</code>



By default, if a MAC address can be retrieved from `ethaddr` (U-Boot environment variable) when the image boots using U-Boot, that retrieved MAC address is used; otherwise a randomly generated MAC address is created.

If you want to use a static MAC address, set the `mac` option for the `devnp-ravb` command with the device's MAC address. You can also set a static MAC address but also keep the random MAC address generation as a fallback capability. To do so, you set the `ethaddr` U-Boot environment variable with the MAC address as follows:

```
U-Boot => setenv ethaddr your_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 -nw -lf /tmp/dhclient.leases ravb0`


For more information about the **devnp-ravb.so** DLL, see “[devnp-ravb.so](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

PCIe

Device	PCIe
Command	<code>pci-server --bus-scan-limit=16 -c</code>
Required binaries	<code>pci-server, pci-tool</code>

Required libraries	<code>libpci.so</code> , <code>pci_server-buscfg-generic.so</code> , <code>pci_hw-rcar-salvator-x.so</code> , <code>pci_debug.so</code> , <code>pci_slog.so</code> , <code>pci_strings.so</code> , <code>pci_cap-0x10.so</code> , <code>pci_cap-0x11.so</code> , <code>pci_cap-0x05.so</code> , <code>pci_cap-0x01.so</code>
Source location	Prebuilt only

SD/MMC/eMMC

Device	SDHI memory card driver
Command	<p>SD memory card:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp,bounce=128k mem name=below4G sdio idx=0,bs=nowp disk name=sd</pre> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp, bounce=128k sdio idx=3</pre> <p>eMMC Flash:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam bounce=128k mem name=below4G sdio idx=2,emmc disk name=mmc</pre> <p>Alternatively, you can choose to use more aggressive eMMC read/write performance, with this command: <code>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</code> For more information the risks when use aggressive read/write performance, see http://www.qnx.com/support/knowledgebase.html?id=501a0000000Mpbr on the QNX website.</p> <hr/> <p> A reset with volatile cache enabled causes the Power-Safe filesystem to become read-only. Resets can occur when there's incorrect system tuning and unstable drivers.</p> <hr/>
Required binaries	<code>devb-sdmmc-rcar_gen3</code>
Required libraries	<code>libcam.so</code> , <code>cam-disk.so</code> , <code>io-blk.so</code> , <code>fs-qnx6.so</code>
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devb/sdmmc</code>

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
```

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
```

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.

Serial

Device	Serial drivers
Command	<code>devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif2</code>
Required binaries	<code>devc-serscif</code>
Required libraries	libsocket.so
Source location	<i>\$BSP_ROOT_DIR/src/hardware/devc/serscif</i>

The SCIF port at the CN25 connector can be used to verify this debug port. Similarly, the following command can be used to launch the SCIF1 port at the CN26 connector:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif1 &
```

HSCIF shares the same pins as SCIF1. To use HSCIF, enable the initialization code *init_hscif()*, in *init_board.c*, and launch it as follows:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 hscif1 &
```

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

SPI Flash

Device	SPI
Command	<code>devf-rcar_qspi-gen3</code>
Required binaries	<code>devf-rcar_qspi-gen3</code>
Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/flash/boards/rcar_qspi</i>

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

Startup

Device	Startup
Command	<code>startup-rcar_m3 -P2 -W -vvvvv</code> <code>startup-rcar_m3 -P2 -r0x54000000,0x3000000,1 -W -vvvvv</code> (if lossy compression is enabled in the Renesas Bootloader)
Required binaries	<code>startup-rcar_m3</code>
Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/startup/boards/rcar_m3</i>

In addition 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. To use this option, DRAM protection must be disabled in the Renesas Bootloader. For more information about disabling the DRAM protection in the Bootloader, see “*Build U-Boot and Renesas Loader*” in the “*Installation Notes*” chapter of this guide.

-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.

Some Renesas Bootloaders enable the lossy decompression feature for booting Linux images. This feature reserves memory area 0x54000000 to 0x56ffffff and prevents QNX Neutrino RTOS from using it as system memory. Because of this, memory faults occur. To prevent memory faults, you must use the `-r` option with the `startup-rcar_m3` command to exclude the memory range from being used by the QNX Neutrino for the `startup-rcar_m3` command. For example:

```
startup-rcar_m3 -P2 -r 0x54000000,0x3000000,1 -W -vvvvv
```

Lossy compression is enabled by default in Renesas Bootloaders v1.0.6 and v1.0.7, but is an optional feature in v1.0.9. Lossy decompression is enabled on your board if you see `NOTICE:BL2 :Lossy Decompr` areas in the BL2 logs when you boot the board as shown here:

```
NOTICE: BL2: R-Car Gen3 Initial Program Loader(CA57) Rev.1.0.9
NOTICE: BL2: R-Car Gen3 Initial Program Loader(CA57) Rev.1.0.9

NOTICE: BL2: PRR is R-Car M3 ES1.0
NOTICE: BL2: Boot device is HyperFlash(80MHz)
NOTICE: BL2: LCM state is CM
NOTICE: BL2: AVS setting succeeded. DVFS_SetVID=0x52
NOTICE: BL2: DDR3200(rev.0.20rc8) [COLD_BOOT]..0
NOTICE: BL2: DRAM Split is 2ch
NOTICE: BL2: QoS is default setting(rev.0.15)
NOTICE: BL2: Lossy Decompr areas
NOTICE: Entry 0: DCMPCAREACRAx:0x80000540 DCMPCAREACRBx:0x570
NOTICE: Entry 1: DCMPCAREACRAx:0x40000000 DCMPCAREACRBx:0x0
NOTICE: Entry 2: DCMPCAREACRAx:0x20000000 DCMPCAREACRBx:0x0
NOTICE: BL2: v1.1(release):3ad02ac
NOTICE: BL2: Built : 16:12:30, July 1, 2017
NOTICE: BL2: Normal boot
NOTICE: BL2: dst=0xe6315208 src=0x8180000 len=512(0x200)
NOTICE: BL2: dst=0x43f00000 src=0x8180400 len=6144(0x1800)
NOTICE: BL2: dst=0x44000000 src=0x81c0000 len=65536(0x10000)
NOTICE: BL2: dst=0x44100000 src=0x8200000 len=524288(0x80000)
NOTICE: BL2: dst=0x50000000 src=0x8640000 len=1048576(0x100000)
```

Thermal Driver

The driver used for thermal management on the board.

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

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

USB (host mode)

Device	USB host
Command (USB)	<code>io-usb-otg -d ehci ioport=0xEE0A0100,irq=0x90 -d ohci ioport=0xEE0A0000,irq=0x90</code> <code>devb-umass blk cache=10m cam pnp</code> Alternatively, you can choose to use more aggressive mass storage read/write performance, with this command: <code>devb-umass blk maxio=256,cache=10m,ra=256k:512k,rapolicy=aggressive,commit=none cam cache,async,pnp</code> For more information the risks when use aggressive read/write performance, see http://www.qnx.com/support/knowledgebase.html?id=501a0000000Mpbr on the QNX website.
Required binaries	<code>io-usb-otg</code> , <code>devb-umass</code>
Required libraries	<code>devu-hcd-ehci.so</code>, <code>devu-hcd-ohci.so</code>, <code>libusbdi.so</code>, <code>libhiddi.so</code>
Source location	Prebuilt only

Watchdog

To enable the watchdog:

1. Modify the buildfile so that `startup` launches with the `-W` option:

```
startup-rcar_m3 -P2 -r 0x54000000,0x3000000,1 -W -vvvvv
```

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

Device	WATCHDOG
Command	<code>wdtkick -W 0x0:0x5A5AFF00</code>
Required binaries	<code>wdtkick</code>
Required libraries	N/A
Source location	<i><code>\$BSP_ROOT_DIR/src/hardware/support/wdtkick</code></i>

Chapter 10

Driver Commands (R-Car M3-N)

The tables below provide a summary of driver commands for the R-Car M3-N 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 completely customizable and are often specific to the requirements of your system. 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 the order that you choose impacts 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 example, in the provided buildfile, the USB driver is started before the mass storage drivers because they rely on it.

- For more information on best practices for building an embedded system and optimizing boot times for your system, see the *Building Embedded Systems* and the *Boot Optimization* guides in the QNX Software Development Platform 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 \(I2C\)](#)
- [USB \(host mode\)](#)
- [USB function \(device mode\)](#)
- [Network](#)
- [Audio](#)
- [SDHI memory card and eMMC](#)
- [PCIe](#)
- [SATA](#)
- [Thermal](#)
- [Graphics](#)

Audio

Device	Audio
Command	<code>io-audio -c /etc/system/config/audio/io_audio.conf</code>
Required binaries	<code>io-audio</code>
Required libraries	<code>deva-ctrl-rcar-cs2000.so</code> , <code>deva-mixer-ak4613.so</code> , <code>libasound.so</code> , <code>libasound.so.2</code> , <code>libm.so.2</code> , <code>libsecpol.so</code>
Source location	<code>\$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar</code> , <code>\$BSP_ROOT_DIR/src/hardware/deva/mixer/ak4613</code>

For more information about the `deva-ctrl-rcar-cs2000.so` and `deva-mixer-ak4613.so` DLLs, see “[deva-ctrl-rcar-cs2000.so](#)” and “[deva-mixer-ak4613.so](#)”, respectively in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

Graphics

Device	GRAPHICS
Command	<code>screen -c /usr/lib/graphics/rcarm3/graphics.conf</code>
Required binaries	<code>screen</code> , <code>gles1-vsync</code> , <code>gles2-gears</code> , <code>sw-vsync</code>
Required libraries	Refer to the <i>Screen Developer's Guide</i> in the QNX SDP 7.0 documentation.
Required configuration files	<code>graphics.conf</code> and for its location in your QNX SDP 7.0 installation, see the “ Using the Screen Graphics Subsystem ” chapter in this guide.
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	I2C2 and I2C4
Command (I2C2)	<code>i2c-rcar-A -p0xe6510000 -i318 -v --u2</code>
Command (I2C4)	<code>i2c-rcar-A -p0xe66D8000 -i51 -v --u4</code>
Required binaries	<code>i2c-rcar-A</code> <code>i2c-rcar-B</code>
Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/i2c</code>

For information about the `i2c_rcar-A` driver, see “[i2c-rcar-A](#).” If your board supports DVFS, you can use the `i2c_rcar-B` driver instead. For more information, see “[i2c-rcar-B](#).”

Network

Device	Ethernet
Command (user-defined MAC address)	<code>io-pkt-v6-hc -dravb mac=2e090a00830c -ptcpip pkt_typed_mem=below4G</code>
Command (generates a MAC address)	<code>sh -c "sh -c "io-pkt-v6-hc -d ravb mac=`genmac-random -m` -p tcpip pkt_typed_mem=below4G"" if_up -p ravb0</code>
Required binaries	<code>io-pkt-v6-hc, ifconfig, dhclient, genmac-random</code>
Required libraries	devnp-ravb.so
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devnp/ravb</code>



By default, if a MAC address can be retrieved from `ethaddr` (U-Boot environment variable) when the image boots using U-Boot, that retrieved MAC address is used; otherwise a randomly generated MAC address is created.

If you want to use a static MAC address, set the `mac` option for the `devnp-ravb` command with the device's MAC address. You can also set a static MAC address but also keep the random MAC address generation as a fallback capability. To do so, you set the `ethaddr` U-Boot environment variable with the MAC address as follows:

```
U-Boot => setenv ethaddr your_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 -nw -lf /tmp/dhclient.leases ravb0`

For more information about the **devnp-ravb.so** DLL, see “[devnp-ravb.so](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

PCIe

Device	PCIe
Command	<code>pci-server --bus-scan-limit=16 -c</code>
Required binaries	<code>pci-server, pci-tool</code>


Required libraries	<code>libpci.so, pci_server-buscfg-generic.so, pci_hw-rcar-salvator-x.so, pci_debug.so, pci_slog.so, pci_strings.so, pci_cap-0x10.so, pci_cap-0x11.so, pci_cap-0x05.so, pci_cap-0x01.so</code>
Source location	Prebuilt only

SATA

Device	SATA
Command	<code>devb-rcarsata blk cache=64m cam cache rcarsata ioport=0xee300000,irq=137 disk name=sata</code>
Required binaries	<code>devb-rcarsata</code>
Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devb/rcarsata</code>

For more information about the `devb-rcarsata` driver, see “[devb-rcarsata](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter of this guide.

SD/MMC/eMMC

Device	SDHI memory card driver
Command	<p>SD memory card:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp, bounce=128k sdio idx=0</pre> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam pnp, bounce=128k sdio idx=3</pre> <pre>waitfor /dev/sd0 1</pre> <p>eMMC Flash:</p> <pre>devb-sdmmc-rcar_gen3 blk cache=64m cam bounce=128k mem name=below4G sdio idx=2,emmc disk name=mmc</pre> <p>Alternatively, you can choose to use more aggressive eMMC read/write performance, with this command: <code>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</code> For more information the risks when use aggressive (volatile cache is enabled) read/write performance, see http://www.qnx.com/support/knowledgebase.html?id=501a0000000Mpbr on the QNX website.</p> <hr/> <p> A reset with volatile cache enabled causes the Power-Safe filesystem to become read-only. Resets can occur when there's incorrect system tuning and unstable drivers.</p> <hr/>

Required binaries	devb-sdmmc-rcar_gen3
Required libraries	libcam.so, cam-disk.so, io-blk.so, fs-qnx6.so
Source location	<i>\$BSP_ROOT_DIR/src/hardware/devb/sdmmc</i>

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
```

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
```

For more information the risks when use aggressive eMMC read/write performance, see <http://www.qnx.com/support/knowledgebase.html?id=501a000000Mpbpr> on the QNX website.

Serial

Device	Serial drivers
Command	<code>devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif2</code>
Required binaries	<code>devc-serscif</code>
Required libraries	libsocket.so
Source location	<code>\$BSP_ROOT_DIR/src/hardware/devc/serscif</code>

The SCIF port at the CN25 connector can be used to verify this debug port. Similarly, the following command can be used to launch the SCIF1 port at the CN26 connector:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 scif1 &
```

HSCIF shares the same pins as SCIF1. To use HSCIF, enable the initialization code `init_hscif()`, in `init_board.c`, and launch it as follows:

```
devc-serscif -e -F -b115200 -x -c 14745600/16 -t 14 hscif1 &
```

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

Startup

Device	Startup
Command	<code>startup-rcar_m3 -P2 -r0x54000000,0x3000000,1 -W -vvvvvv</code>
Required binaries	<code>startup-rcar_m3</code>
Required libraries	N/A
Source location	<code>\$BSP_ROOT_DIR/src/hardware/startup/boards/rcar_gen3/rcar_m3</code>

In addition 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. To use this option, DRAM protection must be disabled in the Renesas Bootloader. For more information about disabling the DRAM protection in the Bootloader, see “*Build U-Boot and Renesas Loader*” in the “*Installation Notes*” chapter of this guide.

-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.

Some Renesas Bootloaders enable the lossy decompression feature for booting Linux images. This feature reserves memory area 0x54000000 to 0x56ffffff and prevents QNX Neutrino RTOS from using it as system memory. Because of this, memory faults occur. To prevent memory faults, you must use the `-r` option with the `startup-rcar_m3` command to exclude the memory range from being used by the QNX Neutrino for the `startup-rcar_m3` command. For example:

```
startup-rcar_m3 -P2 -r 0x54000000,0x3000000,1 -W -vvvvv
```

Lossy compression is enabled by default in Renesas Bootloaders v1.0.6 and v1.0.7, but is an optional feature in v1.0.9. Lossy decompression is enabled on your board if you see `NOTICE:BL2 :Lossy Decompr` areas in the BL2 logs when you boot the board as shown here:

```
NOTICE: BL2: R-Car Gen3 Initial Program Loader(CA57) Rev.1.0.9
NOTICE: BL2: PRR is R-Car M3 ES1.1
NOTICE: BL2: Boot device is HyperFlash(80MHz)
NOTICE: BL2: LCM state is CM
NOTICE: BL2: AVS setting succeeded. DVFS_SetVID=0x52
NOTICE: BL2: DDR1600(rev.0.10)
NOTICE: BL2: DRAM Split is 4ch
NOTICE: BL2: QoS is default setting(rev.0.32)
NOTICE: BL2: Lossy Decompr areas
NOTICE: Entry 0: DCMPCAREACRAx:0x80000540 DCMPCAREACRBx:0x570
NOTICE: Entry 1: DCMPCAREACRAx:0x40000000 DCMPCAREACRBx:0x0
NOTICE: Entry 2: DCMPCAREACRAx:0x20000000 DCMPCAREACRBx:0x0
NOTICE: BL2: v1.1(release):3ad02ac
NOTICE: BL2: Built : 13:03:52, July 1, 2017
NOTICE: BL2: Normal boot
NOTICE: BL2: dst=0xe631a208 src=0x8180000 len=512(0x200)
NOTICE: BL2: dst=0x43f00000 src=0x8180400 len=6144(0x1800)
NOTICE: BL2: dst=0x44000000 src=0x81c0000 len=65536(0x10000)
NOTICE: BL2: dst=0x44100000 src=0x8200000 len=524288(0x80000)
NOTICE: BL2: dst=0x49000000 src=0x8640000 len=1048576(0x100000)
```

Thermal Driver

The driver used for thermal management on the board.

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

For more information about this driver, see “[rcar-thermal](#)” in the “[BSP-specific Drivers and Utilities](#)” chapter.

USB (host mode)

Device	USB host
Command (USB)	<code>io-usb-otg -d xhci ioport=0xEE000000,irq=0x86 -d ehci ioport=0xEE080100,irq=0x8C,ioport=0xEE0A0100,irq=0x90,ioport=0xEE0C0100 -d ohci ioport=0xEE080000,irq=0x8C,ioport=0xEE0A0000,irq=0x90,ioport=0xEE0C0000 devb-umass blk cache=10m cam pnp</code>
Required binaries	<code>io-usb-otg</code> , <code>devb-umass</code>
Required libraries	<code>devu-hcd-ehci.so</code>, <code>devu-hcd-ohci.so</code>, <code>devu-hcd-xhci.so</code>
Source location	Prebuilt only

USB function (device mode)

You can have the USB function driver as mass storage device, for a CDC-ACM (serial) connection, or a Ethernet over USB connection. Make sure the GPIOs are configured appropriately for this use. See the board manufacturer's documentation for the correct switch settings to Host mode.



To use the USB Function on CN9 you must make sure SW15 is in the middle position.

Device	USB OTG (Device mode)
Command (Mass storage)	<code>io-usb-otg -d usbumass-hsusb-rcar ioport=0xe6590000,irq=139</code>

Command (CDC-ACM (serial) device)	<code>io-usb-otg -d usbser-hsusb-rcar ioport=0xe6590000,irq=139</code>
Command NCM device (Ethernet over USB)	<code>io-usb-otg -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139</code>
Required binaries	<code>io-usb-otg, ulink_ctrl, devu-umass_client-block devc-serusb_dcd, ifconfig, if_up</code>
Required libraries	<code>libusbdc1.so, devnp-usbdnet.so, devu-dcd-hsusb-rcar.so, devu-dcd-usbncm-hsusb-rcar.so, devu-dcd-usbser-hsusb-rcar.so, devu-dcd-usbmass-hsusb-rcar.so</code>
Source location	Prebuilt only



CAUTION: If you want a specific USB port (specified using the `ioport` option) to be started as your USB device controller, ensure that you don't start that port as a USB host controller. In other words, if you wanted to start `ioport=0xe6590000, irq=139` as a device controller, don't specify the same USB port when you start the USB host controller driver.

Example of Mass Storage

1. When your target board is ready, create and format a RAM disk to emulate a mass storage device:

```
devb-ram ram capacity=204800 nodinit disk name=ramdisk blk cache=512k
fdisk /dev/ramdisk0 add -t 12
mount -e /dev/ramdisk0
waitfor /dev/ramdisk0t12
mkdosfs -F32 /dev/ramdisk0t12
mount -t dos /dev/ramdisk0t12 /ramdisk
```

2. Launch the USB device stack:

```
io-usb-otg -d usbmass-hsusb-rcar ioport=0xe6590000,irq=139
```

3. Start the Mass Storage function driver, and enable USB soft connect:

```
devu-umass_client-block -l lun=0,fname=/dev/ramdisk0t12
ulink_ctrl -l1
```

4. Connect the R-Car H3 board USB OTG port (CN9) to your host computer to be detected as a mass storage device.

Example of CDC-ACM (serial) device

1. Start USB device stack:

```
io-usb-otg -d usbser-hsusb-rcar ioport=0xe6590000,irq=139
waitfor /dev/usb-dcd/io-usb-otg 4
```

2. Start the USB CDC-ACM function driver, and enable USB soft connect:

```
devc-serusb_dcd -e -v -F -s -d iface_list=0
waitfor /dev/serusb1
ulink_ctrl -ll
```

Example of NCM device (Ethernet over USB)

1. Start USB device stack:

```
io-usb-otg -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139
waitfor /dev/usb/io-usb-otg 4
```

2. Start USB NCM function driver and enable USB soft connections:

```
mount -Tio-pkt -o protocol=ncm,mac=020022446688 devnp-usbdnet.so
if_up -p ncm0
ifconfig ncm0 192.168.0.1
ulink_ctrl -ll
```

For more information about the **devu-dcd-usbncm-hsusb-rcar.so**, **devu-dcd-usbser-hsusb-rcar.so** or **dcd-usbumass-hsusb-rcar.so** drivers, see “[devu-dcd-usbncm-hsusb-rcar.so](#)”, “[devu-dcd-usbser-hsusb-rcar.so](#)” and “[devu-dcd-usbumass-hsusb-rcar.so](#)”, respectively 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_m3 -P2 -r 0x54000000,0x3000000,1 -W -vvvvv
```

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

Device	WATCHDOG
Command	wdtkick -W 0x0:0x5A5AFF00
Required binaries	wdtkick
Required libraries	N/A
Source location	<i>\$BSP_ROOT_DIR/src/hardware/support/wdtkick</i>

Chapter 11

BSP-specific Drivers and Utilities

These are the specific drivers and utilities to this BSP.

- [*deva-ctrl-rcar-cs2000.so*](#)
- [*deva-mixer-ak4613.so*](#)
- [*devb-rcarsata*](#)
- [*devc-serscif*](#)
- [*devf-rcar_qspi-gen3*](#)
- [*devnp-ravb.so*](#)
- [*devu-dcd-usbncm-hsusb-rcar.so*](#)
- [*devu-dcd-usbser-hsusb-rcar.so*](#)
- [*devu-dcd-usbmass-hsusb-rcar.so*](#)
- [*i2c-rcar-A*](#)
- [*i2c-rcar-B*](#)
- [*rcar-thermal*](#)

deva-ctrl-rcar-cs2000.so

SSI Sound driver for Renesas R-Car boards

Syntax:

```
io-audio -d rcar-cs2000 [options]
```

Runs on:

QNX Neutrino

Options:

bit_delay=delay

Specifies the amount of delay between the frame sync and the start of data, which can be one of the following values:

- 0—none
- 1—one cycle (default)

clk_mode=mode

The clock mode, which can be one of the following settings:

- **master**—Usually configured as one of the SSI ports indicated by the `tx_ssi` and `rx_ssi` options, but it can be a different port that has the capability of providing the master clock for the SSI ports as indicated by the `tx_ssi` or `rx_ssi`.
- **slave**—All SSI ports indicated by `tx_ssi` and `rx_ssi` are configured as slave.

The default is **master**.

clk_pol=cpol

Specify the bit clock polarity, which can be one of the following values:

- 0—Data sampled at SCK falling edge.
- 1—data sampled at SCK rising edge (default).

debug

Turn on debugging.

fsync_pol=fpol

Specifies the frame sync (WS) polarity, which can be one of the following values:

- 0—active low (default)
- 1—active high

hdmi_chan=channel

The HDMI channel to use, which is applicable to boards that support HDMI, such as the R-Car H3. You can specify 0, 1, or none. The default is none.

mclk_1=rate

The MCLK RATE 1 in Hertz (Hz). The default value is 22579200, which is set using `MCLK_1` in `$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.le.cs2000/variant.h`.

mclk_2=rate

The MCLK RATE 2 in Hertz (Hz). The default value is 24576000, which is set using `MCLK_2` in `$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.le.cs2000/variant.h`.

mclk_fs=rate

The scale of MCLK RATE to sample rate. The default value is 0, which is set using `MCLK_FS` in `$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.le.cs2000/variant.h`.

mclk_clk_source1=source

MCLK source for MCLK RATE 1. The default is `AUDIO_CLKA`, which is set using `MCLK_SOURCE_1` in `$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.le.cs2000/variant.h`.

mclk_clk_source2=source

MCLK source for MCLK RATE 2. The default is `AUDIO_CLKB`, which is set using `MCLK_SOURCE_2` in `$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.le.cs2000/variant.h`.

op_mode=mode

Indicates the special time-division multiplexing (TDM) mode to use, which can be one of the following:

- `tdm_ext`—TDM extended mode is used. This is a mode when six input voices are extended to 8 output voices. For playback, this means that the client transfers six channel PCM data to DMA memory and the SSI outputs eight channel TDM, while capture SSI gets six channel TDM input and transfers eight channels of PCM data to DMA memory. The data for the extra channels is zeroed.
- `tdm_split`—TDM split mode is used. This mode can be used to split eight channel TDM into four pairs of stereo channels or four channel TDM into four mono channels. The channels resultings from the split are not visible to the client as they need to be routed individually to another peripheral of the SSIU or SCU, or transferred individually to different DMA channels. This mode is not currently supported on the available hardware.
- `tdm_exsplit`—TDM exsplit mode is used (default is 8 channels, but can be split up to 16 channels TDM data). This mode can split up to 16 channels of TDM data and the split can be put into chunks with different numbers of channels. Some of the resulting outputs can be stereo and some can be four, six, eight, or ten channel TDM data. This mode is not currently supported on the available hardware.

When the `voices` option is also specified, the following occurs depending on the mode you specified for this option:

- If you specified `tdm_ext`, you can specify 6 or 8 voices.
- If you specified `tdm_split` and `voices` is set to 4, it becomes four mono channels
- If you specified `tdm_split` and `voices` is set to 8, it becomes four stereo channels
- If you specified `tdm_exsplit` and `voices` is set to 8, it becomes 8 channels
- If you specified `tdm_exsplit` and `voices` is set to 16, it becomes 16 channels

`rx_ssi=rx_ssi_port`

Indicates the SSI port used for receive.

`rx_use_dvc= mode`

Indicates whether to use Digital Volume Control (DVC) for capture, which can be one of these values:

- 0—don't use DVC for capture
- 1—use DVC for capture

The DVC are functional blocks of the Sample Rate Conversion Unit (SCU) and the default is 0, which is set using `DEFAULT_USE_DVC_CAPTURE` in

`$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.ie.cs2000/variant.h`.

`rx_use_src=mode`

Indicates whether to use Sample Rate Conversion (SRC) for capture, which can be one of these values:

- 0—don't use SRC for capture
- 1—use SRC for capture

The SRC are functional blocks of the Sample Rate Conversion Unit (SCU) and the default is 1, which is set using `DEFAULT_USE_SRC_CAPTURE` in

`$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.ie.cs2000/variant.h`.

`sample_rate=minrate:maxrate`

The minimum and maximum sample rate ranges in Hertz (Hz), with the ranges delimited with a colon (:). The default is 8000:192000, which is the full range for the variants of the Renesas boards. The full range can be always supported with SRC enabled (i.e., `tx_use_src` and `tx_use_src` set to 1) , regardless of analog codec used; If you have SRC disabled, the supported range depends on the analog codec that's in use, if any. For Renesas boards equipped with the AK4643 codec, the supported sample rate range with SRC disabled is 8000-48000 Hz.

`sample_rate_list=srate_list`

The list of all supported sample rates [in Hertz (Hz)] the driver is allowed to use, delimited by a colon (:). These sample rates should be among the rates the R-Car SoC and analog codec supports (if any).

sample_size=size

The sample size, which can be set to 16, 24 or 32 bits. The default is 16 which is set using `DEFAULT_SAMPLE_SIZE` in

`$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.ie.cs2000/variant.h`.

slot_size=size

The number of bits in one phase of a frame clock, which can be 16 or 32. The default is 32 which is set using `DEFAULT_SLOT_SIZE` in

`$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.ie.cs2000/variant.h`.

tx_ssi=tx_ssi_ports

A string of one, three, or four digits indicating the SSI ports used for transmit. For multiple SSI ports, you can set them using only these combinations:

- 012—Port 0, 1, and 2 are used simultaneously to provide multichannel audio (two channels per SSI). This mode is used for HDMI. The number of voices supported is 6.
- 0129—Port 0, 1, 2, and 9 are used simultaneously to provide multichannel audio (two channels per SSI). This mode is used for HDMI. Port 0 is the playback channel on SSI port 0 and the number of voices is set to 8.
- Any single number, provided that the SSI port is supported.

tx_use_dvc=mode

Indicates whether to use Digital Volume Control (DVC) for playback, which can be one of these values:

- 0—don't use DVC for playback
- 1—use DVC for playback

The DVC are functional blocks of the Sample Rate Conversion Unit (SCU) and the default is 1, which is set using `DEFAULT_USE_DVC_PLAYBACK` in

`$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.ie.cs2000/variant.h`.

tx_use_src=mode

Indicates whether to use Sample Rate Conversion (SRC) for playback, which can be one of these values:

- 0—don't use SRC for playback
- 1—use SRC for playback

The SRC are functional blocks of the Sample Rate Conversion Unit (SCU) and the default is 1, which is set using `DEFAULT_USE_SRC_PLAYBACK` in

`$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.ie.cs2000/variant.h`.

ver=board_variant

The version of the board you're using. You must specify this option before the `tx_use_*` or `rx_use_*` options. Set the `board_variant` to one of following values, which represents the type of Renesas board you're using:

- H2

- M2
- E2
- V2
- H3
- M3
- D3
- E3
- W2H

voices=num_voices

Indicates number of channels to use, which can be one of the following:

- 1—single channel (mono)
- 2—two channels (stereo)
- 4—four channels (TDM)
- 6—six channels (TDM)
- 8—eight channels (TDM)
- 16—sixteen channels (TDM)

The default is 2, which is set using `DEFAULT_VOICES` in

`$BSP_ROOT_DIR/src/hardware/deva/ctrl/rcar/nto/aarch64/dll.le.cs2000/variant.h`. The behavior of this option is affected if you also set the `op_mode` option. For more information, see the `op_mode` option.

Description:

If you're using this driver with a hardware platform that has an analog codec, the analog codec DLL must be specified using an audio configuration file and `io-audio` must be started using `io-audio -c path-to-io_audio.conf`.



Graphics drivers run at a higher priority than applications, but they shouldn't run at a higher priority than the audio, or else breaks in the audio occur. You can use the `on` command to adjust the priorities of the audio and graphics drivers.

Examples:

The following examples use `io-audio` for cases where there isn't any analog codecs used.

Start audio driver on SSI 0 for Playback and SSI 1 for capture with source:

```
io-audio -d rcar-cs2000 tx_ssi=0,rx_ssi=1,voices=2,clk_mode=master,tx_use_src=1,rx_use_src=1
```

Start audio driver on SSI 0,1,2,9 for Playback with src:

```
io-audio -d rcar-cs2000 tx_ssi=0129,clk_mode=master,tx_use_src=1
```

Start audio driver on SSI 3 for Playback and SSI 4 for capture:

```
io-audio -d rcar-cs2000 tx_ssi=3,rx_ssi=4,clk_mode=master,src_start
```

The following are [CTRL] section for the audio configuration file for cases where an analog codec is used or the audio configuration file is used for any other reason. The driver options described previously are listed as key-value pairs, which are delimited by commas as a value for the `options` key.

```
[CTRL]
name=rcar-cs2000
options=ver=h3,tx_ssi=0,rx_ssi=1,tx_use_src=1,rx_use_src=1,tx_use_dvc=1,rx_use_dvc=0
mixer_dll=ak4613                      # Load deva-mixer-ak4613.so
ak4613_i2c_dev=2                      # /dev/i2c2
ak4613_i2c_addr=16                   # 0x10
ak4613_out1=enable:differential
ak4613_in1=enable:single-ended
```

Files:

deva-mixer-ak4613.so

Supports the mixer.

Errors:

When an error occurs, **deva-ctrl-rcar-cs2000.so** sends a description of the error to the system logger (see `slogger2` in the QNX Neutrino *Utilities Reference*).

deva-mixer-ak4613.so

Mixer DLL for the ASAHI KASEI AK4613 CODEC



You can't invoke this shared object. The audio driver (`deva-ctrl-rcar-cs2000`) loads it if the driver determines that your hardware needs it.

Syntax:

N/A

Runs on:

QNX Neutrino

Options:

None.

Description:

The `deva-mixer-ak4613.so` shared object provides an interface between the AK4613 CODEC and an audio driver. This is a DLL and you can use these configuration options in [CTRL] section for the audio card the mixer DLL is mounter under in the audio configuration file: Configuration file options:

`ak4643_i2c_dev=device_num`

I2C device number corresponding to the AK4613 codec.

`ak4643_i2c_addr=slave_value`

I2C slave address corresponding to the AK4613 codec.

`ak4613_debug=enable|disable`

Enable register dumps. The default is set to `disable`.

`ak4613_out1=enable|disable[:differential|single-ended]`

Enable DAC output 1 and specify whether DAC output 1 is `differential` or `single-ended`. Default is set to `disable`.

`ak4613_out2=enable|disable[:differential|single-ended]`

Enable DAC output 2 and specify whether DAC output 2 is `differential` or `single-ended`. Default is set to `disable`.

`ak4613_out3=enable|disable[:differential|single-ended]`

Enable DAC output 3 and specify whether DAC output 3 is `differential` or `single-ended`. Default is set to `disable`.

ak4613_out4=enable|disable[:differential|single-ended]

Enable DAC output 4 and specify whether DAC output 4 is differential or single_ended.
Default is set to disable.

ak4613_out5=enable|disable[:differential|single-ended]

Enable DAC output 5 and specify whether DAC output 5 is differential or single_ended.
Default is set to disable.

ak4613_out6=enable|disable[:differential|single-ended]

Enable DAC output 6 and specify whether DAC output 6 is differential or single_ended.
Default is set to disable.

ak4613_in1=enable|disable[:differential|single-ended]

Enable ADC input 1 and specify whether ADC input 1 is differential or single_ended. Default is set to disable.

ak4613_in2=enable|disable[:differential|single-ended]

Enable ADC input 2 and specify whether ADC input 2 is differential or single_ended. Default is set to disable.

Examples:

```
[CTRL]
name=rcar-ak4613
options=scu=1
mixer_dll=ak4613                # Load deva-mixer-ak4613.so
ak4613_i2c_dev=2                 # /dev/i2c2
ak4613_i2c_addr=16              # 0x10
ak4613_out1=enable:differential
ak4613_in1=enable:differential
```

When an error occurs, deva-mixer-ak4613.so sends a description of the error to the system logger (see `slogger2`).

Errors:

When an error occurs, deva-mixer-ak4613.so sends a description of the error to the system logger (see `slogger2`).

devb-rcarsata

Driver for R-Car SATA controller.



You must be **root** or have the right abilities to start this driver.

Syntax:

```
devb-rcarsata [cam option [, option] ...]
               [mem option [, option] ...]
               [sata option [, option] ...]
               [blk option [, option] ...] &
```

Runs on:

QNX Neutrino

Options:



Use commas (,) to separate the options. You can put the `cam`, `mem`, `sata`, and `blk` groups of options in any order.

cam options

The `cam` options control the common access methods:

async

Perform CPU cache invalidation after an I/O operation in a thread that's separate from the one that's notifying the upper layer. In some cases, and on some boards, this can provide a performance boost.

bounce=size

Specify the size of the driver's "bounce" buffer. The *size* argument can include a case-insensitive suffix that indicates the units: `B` (bytes; the default), `K` (kilobytes), `M` (megabytes), or `P` (pages). The bounce buffer is used if a specific piece of hardware can't directly access `io-blk` block cache buffers. The most common example is chips that don't support scatter-gather. The default is 32 KB.

cache[=on | off]

Specify to perform cache invalidation or cache flushes on the buffers. Map the bounce buffer as `PROT_NOCACHE` (`cache=off`) or not (`cache` or `cache=on`).

lun=*mask*

Enable Logical Unit Number (LUN) scan for the devices specified in *mask*. The *mask* is a hex bitmask specifying which IDs to scan for; the default is 0x00.

pnp

Enable CAM plug and play (i.e., don't exit at startup when no devices are found). The default is off.

smmu=[0|1|on|off]

QNX Neutrino 7.0.4 or later) Disable or enable the SMMU Manager support using one of these values:

- 0/off: (Default) Disable SMMU Manager support.
- 1/on: Enable SMMU Manager support.

If the SMMU Manager service (*smmuman*) service isn't running, the driver won't start and an error is logged to the *slogger*. This option works for all boards, but only boards with WS 3.0 Silicon revisions are supported.

user=*UID:GID*

Operate as the specified user and group IDs. The driver retains the necessary process-manager abilities.

verbose

Increase the verbosity of the low-level hardware driver.

mem options**name=*tname***

The typed memory name to use.



It's up to the startup to set up typed memory. For more information, see “Typed memory” in the “Interprocess Communication (IPC)” chapter of the *System Architecture* guide.

sata options

The *sata* options control the driver's interface to the R-Car SATA controller. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the *sata* keyword before each controller's set of options.

- Interface-specific options:

ioport=*addr*

The address of the SATA controller register memory map.

ioportlen=*length*

The length of the address space. The default is 0x2000.

irq=*req*

The interrupt used by the controller.

priority=*prio*

Set the priority of the processing thread. The default is 21.

timeout=*timeout*

Set the I/O request timeout, in seconds. The default is 5.

blk options

The blk options control `io-blk.so`. For more information, see “`io-blk.so`” in the QNX Neutrino *Utilities Reference* in the SDP 7.0 documentation.

Description:

The `devb-rcarsata` supports the Renesas SATA controller.

Examples:

Detect all SATA controllers, and list all connected devices:

```
devb-rcarsata &
```

Detect all SATA controllers and use a typed memory region named **dma** (which the startup must have set up):

```
devb-rcarsata mem name=/ram/dma &
```

Files:

The `devb-rcarsata` driver causes `io-blk.so` to adopt various block special devices under **/dev**. These devices are normally named **hd n** , where n is the physical unit number of the device.

This driver could also require the following shared objects:

Binary	Required
<code>cam-disk.so</code>	For hard-disk access.
<code>libcam.so</code>	Always

Exit status:

The `devb-rcarsata` driver terminates only if an error occurs during startup, or if it has successfully forked itself upon startup because it hadn't been initially started in the background.

0

The `devb-rcarsata` driver wasn't started in the background and therefore forked itself. The original process terminated with a zero exit status, the forked process continued.

> 0

An error occurred during startup.

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 opt [, opt ...]]
              [-O number]
              [-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:

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 SDP 7.0 documentation.

For this driver, the generic `-o` option supports the `smmu` option.

`smmu=0|1|off|on`

(QNX Neutrino 7.0.4 or later) Disable or enable the SMMU Manager support using one of these values:

- `0|off` — (Default) Disable SMMU Manager support.
- `1|on` — Enable SMMU Manager support.

If the SMMU Manager service (`smmuman`) service isn't running, the driver won't start and an error is logged to the `slogger`.

In addition to the generic `devc-ser*` options, 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/fsOp0** — raw
- **/dev/fsOp1** — formatted
- **/dev/fsOp2** — formatted
- **/dev/fsOp3** — formatted

If you start the driver with **-e 1**, the driver creates all the raw entries in **/dev**, but mounts only **/dev/fsOp1** (**/dev/fsOp0** 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/fsOp0 -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:

- $0 < hi_water < max$
- $0 \leq lo_water \leq hi_water$
- $hi_water < max - 100$

-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 identified by *num*. The workarounds available (if any) depend on the board.

-w *buffer size*

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

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.

buffer_size=buffer_size

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

clock=clockrate

Specify the clock rate in Hertz.

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*.

smmu=[0|1|on|off]

QNX Neutrino 7.0.4 or later) Disable or enable the SMMU Manager support using one of these values:

- *0|off* — (Default) Disable SMMU Manager support.
- *1|on* — Enable SMMU Manager support.

If the SMMU Manager service (*smmuman*) service isn't running, the driver won't start and an error is logged to the *slogger*. This option works for all boards, but only boards with WS 3.0 Silicon revisions are supported.

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.



This driver doesn't support boards that have the QSPI Flash chip (part number: S25FS128).

Example:

```
devf-rcar_qspi-gen3 -s0x08000000,0x04000000 bs=base=0xee200000,size=0xac,buffer=0xee208000,  
buffer_size=0xff,ver=h3
```

devnp-ravb.so

Driver for Renesas RAVB Ethernet interfaces. This driver also provides BCM89811 PHY and KSZ9031 PHY support

Syntax:

```
io-pkt-v6-hc -d ravb [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=100|1000

Force the media data rate in 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=0xio_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.

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

mii

Use MII mode between the MAC and the PHY instead of RGMII mode, which is the default mode.

miiLite

Use MII-Lite mode between the MAC and the PHY instead of RGMII mode, which is the default mode.

Examples:

Start networking with the devnp-ravb driver:

```
io-pkt-v6-hc -d ravb
ifconfig ravb0 192.0.2.1
```

devu-dcd-usbncm-hsusb-rcar.so

Driver for the USB device (peripheral) controller on Renesas boards for network devices.

Syntax:

```
io-usb-otg -d usbncm-hsusb-rcar [option[,option ...]] ... &
```

Runs on:

QNX Neutrino

Targets:

aarchle64

Options:

dmac=0|1

Enable (1) or disable (0) DMA transactions. The default is disabled (0).

ioport=addr

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

iosize=size

The number of device IO-memory bytes that we want to access, specified in bytes. The default is 4096 bytes (4 K).

irq=num

The interrupt request number.

ser=serstring

Override the serial string descriptor with the specified *serstring*. When you use this option, it overrides the default serial number for this driver, which is ABC-0123456789.

verbose=num

Set the verbosity level (0-7). If no *num* is specified with this option, then the verbosity level is 5.

Description:

This is the device-side controller driver for USBNET (USB Device Network Driver).

You can't run the **devu-hcd-ehci.so** driver as a host controller and the **devu-dcd-usbncm-hsusb-rcar.so** as a device controller at the same time. If you've run **devu-hcd-ehci.so** as a host controller, you must first `slay` the `io-usb-otg` instance associated with the host controller before you start **devu-dcd-usbncm-hsusb-rcar.so** as the device controller.

Examples:

Attach to the controller by specifying the I/O port and IRQ.

```
io-usb-otg -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139
```

If you are already running an instance of `io-usb-otg` with the default resource manager path (e.g., **/dev/usb/io-usb-otg**), you must use the `-n` option to specify a different resource manager path.

```
io-usb-otg -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139  
-n /dev/otg/io-usb-otg
```

To enable SMMU Manager (SMMUMAN), you must specify the `-s` option with the `smmu` option with the `io-usb-otg` command:

```
io-usb-otg -s smmu=on -d usbncm-hsusb-rcar ioport=0xe6590000,irq=139
```


devu-dcd-usbser-hsusb-rcar.so

Driver for the USB device (peripheral) controller on Renesas boards for serial devices.

Syntax:

```
io-usb-otg -d usbser-hsusb-rcar [option[,option ...]] ... &
```

Runs on:

QNX Neutrino

Targets:

aarchle64

Options:

dmac=0|1

Enable (1) or disable (0) DMA transactions. The default is disabled (0).

ioport=addr

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

iosize=size

The number of device IO-memory bytes that we want to access, specified in bytes. The default is 4096 bytes (4K).

irq=num

The interrupt request number.

ser=serstring

Override the serial string descriptor with the specified *serstring*. When you use this option, it overrides the default serial number for this driver, which is ABC-0123456789.

verbose=num

Set the verbosity level (0-7). If no *num* is specified with this option, then the verbosity level is 5.

Description:

This is the device-side controller driver for USBSER [CDC Serial(ACM)].

You can't run the **devu-hcd-ehci.so** driver as a host controller and the **devu-dcd-usbser-hsusb-rcar.so** as a device controller at the same time. If you've run **devu-hcd-ehci.so** as a host controller, you must first `slay` the `io-usb-otg` instance associated with the host controller before you start **devu-dcd-usbser-hsusb-rcar.so** as the device controller.

Examples:

Attach to the controller by specifying the I/O port and IRQ.

```
io-usb-otg -d usbser-hsusb-rcar ioport=0xe6590000,irq=139
```

If you are already running an instance of `io-usb-otg` with the default resource manager path (e.g., **/dev/usb/io-usb-otg**), you must use the `-n` option to specify a different resource manager path.

```
io-usb-otg -d usbser-hsusb-rcar ioport=0xe6590000,irq=139  
-n /dev/otg/io-usb-otg
```

To enable SMMU Manager (SMMUMAN), you must specify the `-s` option with the `smmu` option with the `io-usb-otg` command:

```
io-usb-otg -s smmu=on -d usbser-hsusb-rcar ioport=0xe6590000,irq=139
```

devu-dcd-usbmass-hsusb-rcar.so

Driver for the USB device (peripheral) controller on Renesas boards for mass storage devices.

Syntax:

```
io-usb-otg -d usbmass-hsusb-rcar [option[,option ...]] ... &
```

Runs on:

QNX Neutrino

Targets:

aarchle64

Options:

dmac=0|1

Enable (1) or disable (0) DMA transactions. The default is disabled (0).

ioport=addr

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

iosize=size

The number of device IO-memory bytes that we want to access, specified in bytes. The default is 4096 bytes (4K).

irq=num

The interrupt request number.

ser=serstring

Override the serial string descriptor with the specified *serstring*. When you use this option, it overrides the default serial number for this driver, which is ABC-0123456789.

smmu=0|1|off|on

(QNX Neutrino 7.0.4 or later) Disable or enable the SMMU Manager support using one of these values:

- *0|off* — (Default) Disable SMMU Manager support.
- *1|on* — Enable SMMU Manager support.

If the SMMU Manager service (`smmuman`) service isn't running, the driver won't start and an error is logged to the `slogger`. This option is intended with boards that use WS 3.0 Silicon revisions only.

`verbose=num`

Set the verbosity level (0-7). If no *num* is specified with this option, then the verbosity level is 5.

Description:

This is the device-side controller driver for USBUMASS (USB Mass Storage driver).

You can't run the **`devu-hcd-ehci.so`** driver as a host controller and the **`devu-dcd-usbmass-hsusb-rcar.so`** as a device controller at the same time. If you've run **`devu-hcd-ehci.so`** as a host controller, you must first **`slay`** the **`io-usb-otg`** instance associated with the host controller before you start **`devu-dcd-usbmass-hsusb-rcar.so`** as the device controller.

Examples:

Attach to the controller by specifying the I/O port and IRQ.

```
io-usb-otg -d usbmass-hsusb-rcar ioport=0xe6590000,irq=139
```

If you are already running an instance of **`io-usb-otg`** with the default resource manager path (e.g., **`/dev/usb/io-usb-otg`**), you must use the **`-n`** option to specify a different resource manager path.

```
io-usb-otg -d usbmass-hsusb-rcar ioport=0xe6590000,irq=139  
-n /dev/otg/io-usb-otg
```

To enable SMMU Manager (DMA-mode enable), you must specify the **`-s`** option with the **`smmu`** option with the **`io-usb-otg`** command:

```
io-usb-otg -s smmu=on -d usbmass-hsusb-rcar ioport=0xe6590000,irq=139
```

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: 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
```

i2c-rcar-B

Driver for inter-integrated circuits on the Renesas boards that support DVFS (Dynamic Voltage and Frequency Scaling)

Syntax:

```
i2c-rcar-B [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: 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-B -c 33000000 --u0
```

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

```
i2c-rcar-B --u0 -c 33000000
```

R-Car options:

These are R-Car-specific drivers.

-c *peripheral_clk*

The peripheral clock (Hz). The default is 33000000.

-d *duty_cycle*

The I2C clock duty cycle. The default is 1.25.

-i *irq*

The I2C IRQ. There is no default value.

-p *phys_base*

The I2C base address. There is no default value.

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).

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 h3 for the Renesas R-Car H3 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
```