RZ/T2H and RZ/N2H Evaluation Board

Linux Start-up Guide

R01US0682EJ0101
Rev.1.01
Dec. 23, 2024

Introduction

This document provides a guide to prepare RZ/T2H and RZ/N2H evaluation boards to boot up with the Board Support Package.

This guide provides the following information:

- Building procedure
- Preparation for use
- Boot loader and U-Boot
- How to run this Linux package on the target board
- How to create a software development kit (SDK)

Target Reference Board

- RZ/T2H Evaluation Board
- RZ/N2H Evaluation Board

Target Software

- RZ/T2H, RZ/N2H Board Support Package version 1.0.1 or later. (hereinafter referred to as “BSP”)

Contents

1. Environmental Requirement 4
2. Build Instructions 6
2.1 Building images 6
2.2 Notes for building 8
3. Preparing the SD Card 10
4. Reference Board Setting 11
4.1 Prerequisites 11
4.2 Overview of the evaluation environment 11
4.3 Board Settings of RZ/T2H Evaluation Board 13
4.3.1 How to set boot mode of RZ/T2H 13
4.3.2 How to set CN77 13
4.3.3 How to use debug serial (console ouput) 14
4.3.4 How to set CN15 for the HDMI output (Optional) 14
4.3.5 How to enable the ethernet port of ETH2 CN45 (Optional) 15
4.3.6 How to set for Ethernet Port as RGMII and MII 15
4.4 Board Settings of RZ/N2H 16
4.4.1 How to set boot mode of RZ/N2H 16
4.4.2 How to use debug serial (console ouput) 17
4.4.3 How to set CN20 for the HDMI output (Optional) 17
4.4.4 How to enable the ethernet port of ETH2 CN39 (Optional) 18
4.4.5 How to set for Ethernet Port as RGMII and MII 19
4.5 Startup Procedure of RZ/T2H and RZ/N2H 20
4.5.1 Power Supply 20
4.5.2 Building files to write 20
4.5.3 Settings of Tera Term and evaluation board 21
4.6 Download Flash Programmer to RAM 23
4.7 Write the Bootloader 24
4.8 Change Back to Normal Boot Mode 25
5. Booting and Running Linux 26
5.1 Set micro-SD card 26
5.2 Power on the board and Startup Linux 26
5.3 Shutdown the Board 27
6. Building the SDK 28
7. Application Building and Running 29
7.1 Make an application 29
7.1.1 How to extract SDK 29

7.1.2 How to build Linux application 30
7.2 Run a sample application 31

8. GUI Application Framework (Optional) 32
8.1 LVGL 32
8.2 Qt 34

9. Appendix 36
9.1 Ethernet port 36
9.2 Docker 38
9.3 How to boot from eMMC 39
9.4 How to boot from eSD 42

10. Revision History 43

Website and Support 44

1. Environmental Requirement

The environment for building the Board Support Package (hereinafter referred to as “BSP”) is listed in Table1. Please refer to the documents below for details about setting up the environment:

Figure 1 shows the recommended environment for this package. The picture of RZ/T2H is shown as an example.

A Linux PC is required for building the software.

A Windows PC can be used as a serial terminal interface with software such as TeraTerm.

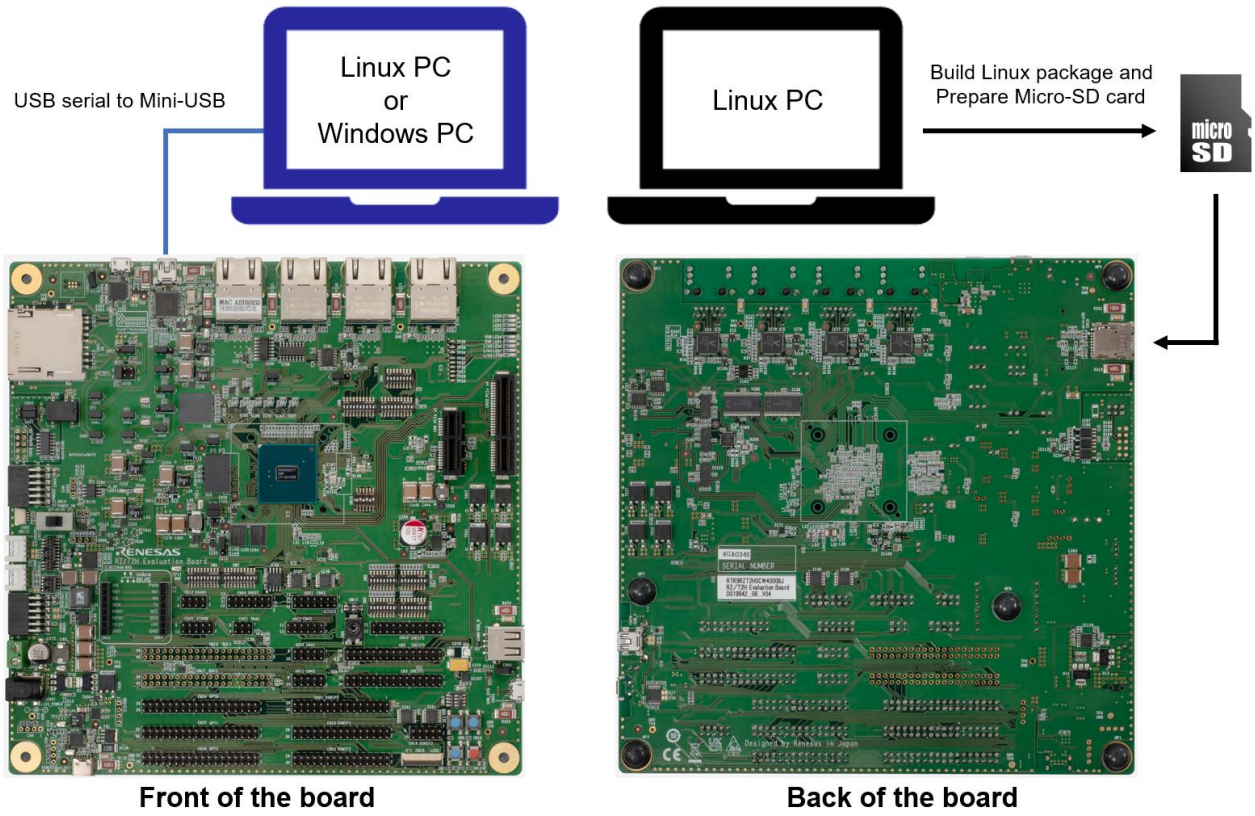


Figure 1. Recommend Environment

Table 1. Equipment and Software for Developing Environments of Linux Platform

Equipment		Description
Linux Host PC		Used as build/debug environment 100GB free space on HDD or SSD is necessary
	OS	Ubuntu 20.04 LTS (*1) 64 bit OS must be used. 20.04 inside a docker container also OK.
Windows Host PC		Used as debug environment, controlling with terminal software
	OS	Windows 10 or Windows 11
	Terminal software	Used for controling serial console of the target board Tera Term (latest version) is recommended Available at Releases TeraTermProject/teraterm(github.com)
	VCP Driver	Virtual COM Port driver which enables to communicate Windows Host PC and the target board via USB which is virtually used as serial port. Available at: http://www.ftdichip.com/Drivers/VCP.htm
USB serial to mini–USB Cable		Serial communication (UART) between the Evaluation Board Kit and Windows PC. The type of USB serial connector on the Evaluation Board Kit is USB type Mini-B.
micro–SD Card		Use to boot the system, and store applications.

Note *1) Please note that the build fails when Ubuntu 22.04 is used.

2. Build Instructions

2.1 Building images

This section describes the instructions to build the Board Support Package.
Before starting the build, run the command below on the Linux Host PC to install packages used for building the BSP.

```
$ sudo apt-get update
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat cpio python python3 python3-pip python3-pexpect \
xz-utils debianutils iputils-ping libsdl1.2-dev xterm p7zip-full libyaml-dev \
libssl-dev bmap-tools
```

Please refer to the URL below for detailed information:

- https://docs.yoctoproject.org/3.1.31/brief-yoctoprojectqs/brief-yoctoprojectqs.html

Run the commands below and set the username and email address before starting the build procedure. Without this setting, an error occurs when building procedure runs git command to apply patches.

```
$ git config --global user.email "you@example.com"
$ git config --global user.name "Your Name"
```

Copy all files obtained from Renesas into your Linux Host PC prior to the steps below. The directory which you put the files in is described as <package download directory> in the build instructions.

(1) Create a working directory at your home directory, and decompress Yocto recipe package

Run the commands below. The name and the place of the working directory can be changed as necessary.

```
$ mkdir ~/rzt2h_n2h_bsp_<package version>
$ cd ~/rzt2h_n2h_bsp_<package version>
$ cp ../<package download directory>/*.zip .
$ unzip ./RTK0EF0177Z00002ZJ-<package version>.zip
$ tar zxvf ./RTK0EF0177Z00002ZJ-<package version>/rzt2h_n2h_bsp_<package version>.\
tar.gz
```

Note) Please note that your build environment must have 100GB of free hard drive space in order to complete the minimum build. The Yocto BSP build environment is very large. Especially in case you are using a Virtual Machine, please check how much disk space you have allocated for your virtual environment.

Note) <package version>: e.g. v1.0.1

(2) Build Initialize

Initialize a build using the 'oe-init-build-env' script in Poky and point TEMPLATECONF to platform conf path.

```
$ TEMPLATECONF=$PWD/meta-renesas/meta-rzt2h/docs/template/conf/ source \
poky/oe-init-build-env build
```

(3) Start a build

Run the commands below to start a build. Building an image can take up to a few hours depending on the user's host system performance.

Build the target file system image using bitbake.

```
$ MACHINE=<board> bitbake core-image-<target>
```

<board> can be selected by referring to the following table.

Table 2. List of platforms and the board

Renesas MPU	<board>
RZ/T2H	rzt2h-dev
RZ/N2H	rzn2h-dev

<target> can be selected below. Please refer to the following table for supported image.

Table 3. List of platforms and the board

Target Image Name	Purpose
core-image-minimal	Minimal set of components.
core-image-bsp	Minimal component set with audio support and essential tools.

After the building is successfully completed, a similar output will be seen, and the command prompt will return.

```
NOTE: Tasks Summary: Attempted 3104 tasks of which 16 didn't need to be rerun and all succeeded.
```

All necessary files listed in the below table will be generated by the bitbake command and will be in the **"build/tmp/deploy/images"** directory.

Table 4. Image files

RZ/T2H	**Linux kernel**	Image-rzt2h-dev.bin
	Device tree file	Image-r9a09g077m44-dev.dtb
	root filesystem	core-image-<target>-rzt2h-dev.tar.bz2
	Boot loader	bl2_bp_xspi0-rzt2h-dev.srec fip-rzt2h-dev.srec
	SD image	core-image-<target>-rzt2h-dev.wic.gz core-image-<target>-rzt2h-dev.wic.bmap
RZ/N2H	**Linux kernel**	Image-rzn2h-dev.bin
	Device tree file	Image-r9a09g087m44-dev.dtb
	root filesystem	core-image-<target> -rzn2h-dev.tar.bz2
	Boot loader	bl2_bp_xspi0-rzn2h-dev.srec fip-rzn2h-dev.srec
	SD image	core-image-<target>-rzn2h-dev.wic.gz core-image-<target>-rzn2h-dev.wic.bmap

If you want to know the components installed to the root filesystem, please check the manifest file. The manifest file is created to the following path after building the images:

~/rzt2h_n2h_bsp_<package version>/build/tmp/deploy/images/rzt2h-dev/core-image-<target>-rzt2h-dev.manifest

~/rzt2h_n2h_bsp_<package version>/build/tmp/deploy/images/rzn2h-dev/core-image-<target>-rzn2h-dev.manifest

2.2 Notes for building

(1) Install ethtool command

"ethtool" command is a command for configuring network interfaces. If you want to install the ethtool command on Linux, add the following line to "~/rzt2h_n2h_bsp_<package version>/build/conf/local.conf" and rebuild the BSP.

```
IMAGE_INSTALL_append = " ethtool"
```

(2) Patch for Changing Ethernet port configuration on the evaluation board

The ethernet ports on the evaluation board are configured as follows:

Table 5. Default Port Configuration

Ethernet Port Number (*)	Configuration	Note
ETH0	MII	Ethernet port or Ethernet Switch port
ETH1	MII	Ethernet port or Ethernet Switch port
ETH2	RGMII	Ethernet port
ETH3	RGMII	Ethernet port

(*) The ethernet port number is printed on the evaluation board as follows.

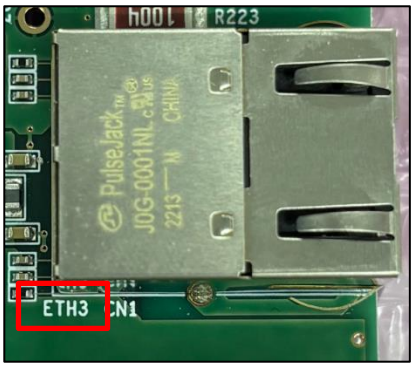


Figure 2. Example Image of Ethernet port

If you want to change the configuration, run the following commands to apply a patch, add a line in "~/rzt2h_n2h_bsp_<package version>/build/conf/local.conf" according to your board configuration, and build BSP as section 2.1. Please note that 2 patterns of configuration are supported now.

For RZ/T2H

```
$ cd ~/rzt2h_n2h_bsp_<package version>/meta-renesas
$ patch -p1 < ../extra/rzt2h/0001-rzt2-linux-renesas-Add-support-changing-ports-to.\
patch
$ cd ..
```

For RZ/N2H

```
$ cd ~/rzt2h_n2h_bsp_<package version>/meta-renesas
$ patch -p1 < ../extra/rzn2h/0001-rzt2-linux-renesas-add-support-changing-ports-to.\
patch
$ cd ..
```

Pattern 1: Change ETH0,1 to RGMII

Add the following line to local.conf.

```
MACHINE_FEATURES_append = " eth01_rgmii"
```

Pattern 2: Change ETH2,3 to MII

Add the following line to local.conf.

```
MACHINE_FEATURES_append = " eth23_mii"
```

After applying a patch and adding the line to local.conf, start the build according to step (3) in section 2.1.

3. Preparing the SD Card

This chapter describes how to prepare the micro-SD card. This is used to boot the evaluation board, so please follow the instructions. RZ/T2H is explained as an example.

(1) Insert the micro-SD card into your Linux PC

(2) Check the device name and unmount

Run the command below to determine the device name assigned to the micro-SD card. In this example, it's "/dev/sdb". If necessary, replace "/dev/sdb" with the correct device name for your system. After that, unmount this device. Example is as below:

```
$ sudo fdisk -l
Disk /dev/sdb: 3.74 GiB, 3997171712 bytes, 7806976 sectors
Disk model: Storage Device
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xxxxxxxxxx
$ umount /dev/sdb1
$ umount /dev/sdb2
```

(3) Expand disk image

Run the command below to expand the disk image onto the micro-SD card. When you complete the following steps, the micro-SD card is prepared correctly.

```
$ cd ~/rzt2h_n2h_bsp_<package version>
$ sudo bmaptool copy ./build/tmp/deploy/images/rzt2h-dev/core-image-<target>-rzt2h-\
dev.wic.gz /dev/sdb
```

Table 6. File and directory in the micro-SD card

Type/Number	Size	Filesystem	Contents
Primary #1	500MB (minimum 128MB)	FAT32	bl2_bp_xspi0-rzt2h-dev.srec fip-rzt2h-dev.srec
Primary #2	All remaining	Ext4	./ ├── bin ├── boot │ ├── Image │ └── <device>-dev.dtb ├── dev ├── etc ├── home ├── lib ├── lib64 ├── media ├── mnt ├── opt ├── proc ├── run ├── sbin ├── sys ├── tmp ├── usr └── var

4. Reference Board Setting

4.1 Prerequisites

Chapter 4 introduces the switches, jumper pin settings, and other configuration options on the evaluation board. These changes are based on the factory default settings of the evaluation board and are intended for running Linux. Please note that changing settings other than those described here may cause Linux to malfunction.

If you want to switch ethernet ports on the evaluation board to RGMII mode or MII mode, you will need to modify the board. Please refer to the manual below for your specific board for instructions on how to modify it. You can also find more detailed information about board settings in the manual for your specific board.

- RZ/T2H Evaluation Board User's Manual

4.2 Overview of the evaluation environment

The following environment of Hardware and Software is used in the evaluation.

Hardware preparation (Users should purchase the following equipment.):

- USB Type-C cable compatible with USB PD. (e.g. AK-A8485011 (manufactured by Anker))
- USB PD Charger 45W (15V 3.0A) or more. (e.g. PowerPort III 65W Pod (manufactured by Anker))
- USB Type-A miniB cable (Any cables)
- micro HDMI cable (Any cables)
- PC Installed FTDI VCP driver and Terminal software (Tera Term) (*1)

(*1) Please install the FTDI driver that can be following website (Releases TeraTermProject/teraterm(github.com)).

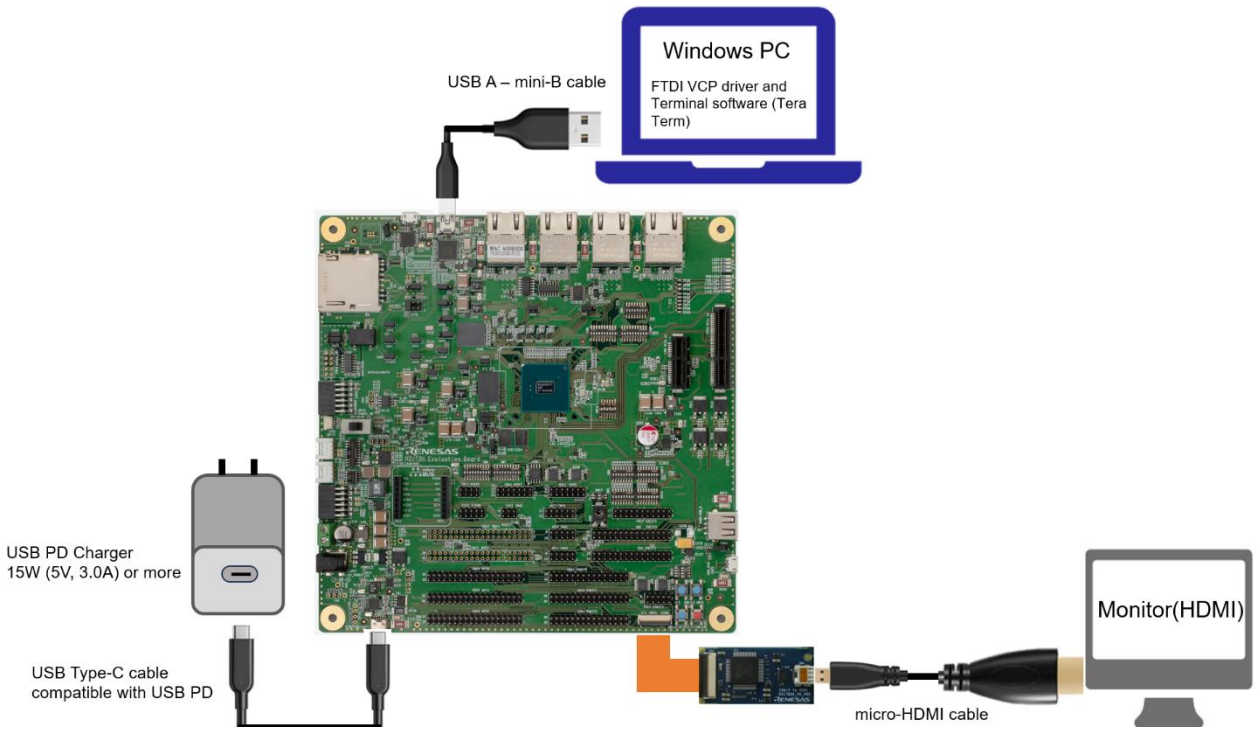


Figure 3. Recommend Environment (1)

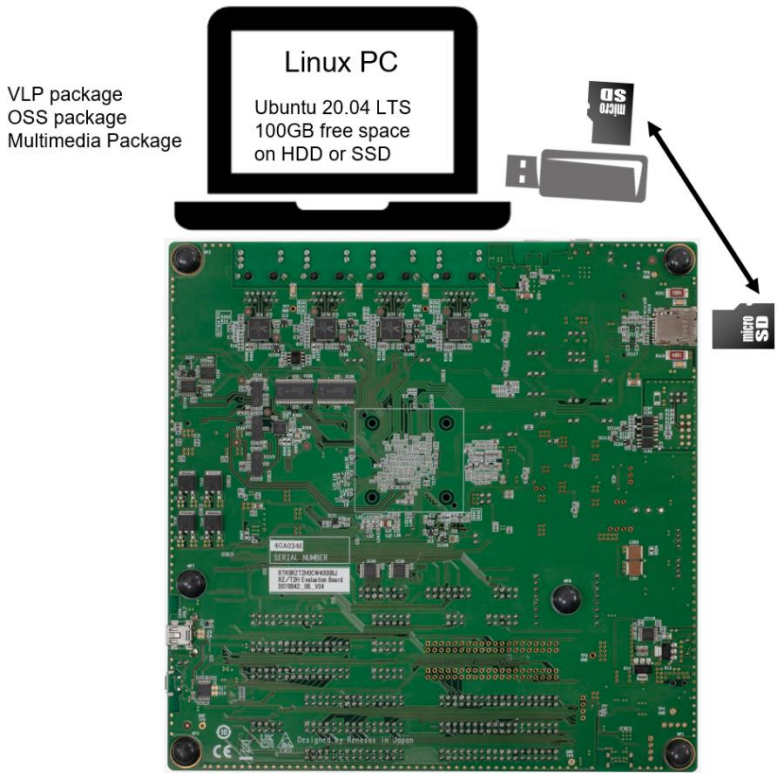


Figure 4. Recommend Environment (2)

4.3 Board Settings of RZ/T2H Evaluation Board

4.3.1 How to set boot mode of RZ/T2H

(1) SCIF Download Mode

Set the SW14 settings as follows. This mode is used to write bootloaders.

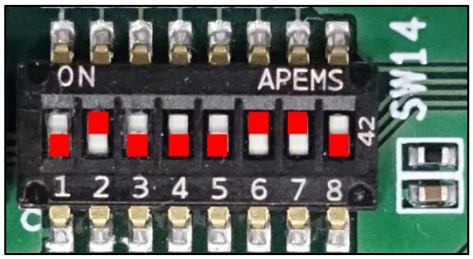


Figure 5. SW14 (1)

SW14-1	OFF
SW14-2	ON
SW14-3	OFF
SW14-4	OFF
SW14-5	OFF
SW14-6	ON
SW14-7	ON
SW14-8	OFF

(2) xSPI Boot Mode

Set the SW14 settings as follows. This mode is used to boot Linux.

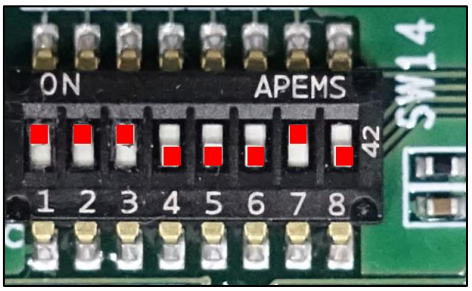


Figure 6. SW14 (2)

SW14-1	ON
SW14-2	ON
SW14-3	ON
SW14-4	OFF
SW14-5	OFF
SW14-6	OFF
SW14-7	ON
SW14-8	OFF

4.3.2 How to set CN77

Set CN77 as follows.

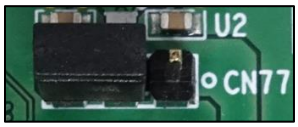


Figure 7. CN77 Setting

4.3.3 How to use debug serial (console ouput)

Connect USB Type Mini-B cable to CN34: USB Type Mini-B Connector.

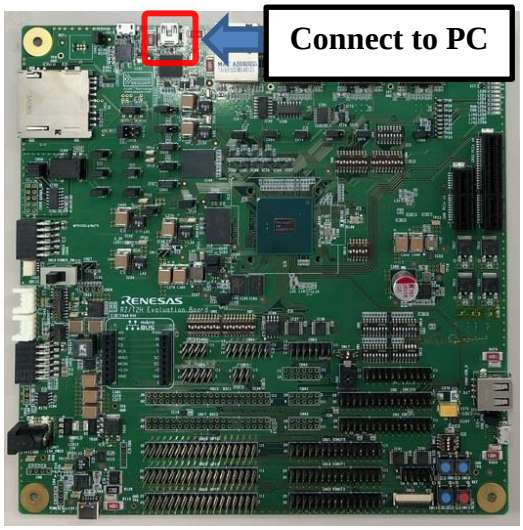


Figure 8. Connecting console for debug

4.3.4 How to set CN15 for the HDMI output (Optional)

Connect the flexible cable and the connector correctly. When the step is wrong, the HDMI output fails. This step is optional. Skip this step when you do not use the HDMI output.

- Connect the flexible cable to the CN1 on “24bit to HDMI” module. Install it with the side that does not have any silk facing upwards.

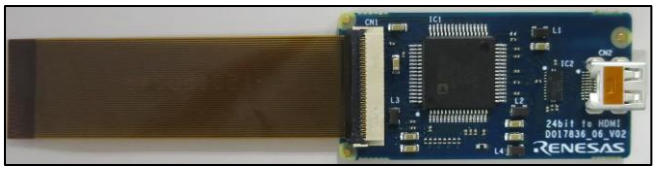


Figure 9. Setting of the CN15

- Connect the flexible cable to CN15 on the evaluation board as the image below. When installing it, ensure that you do not install it in the wrong orientation.

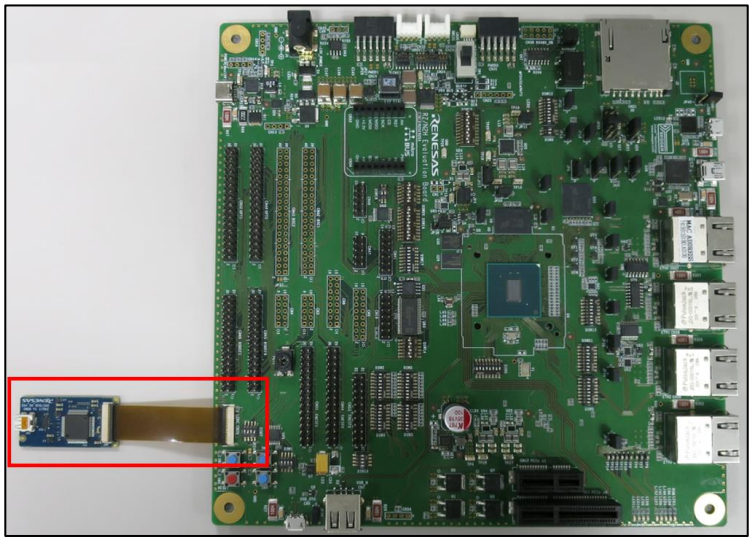


Figure 10. How to connect the module

4.3.5 How to enable the ethernet port of ETH2 CN45 (Optional)

The ethernet port of ETH3 CN1 is enabled as default. If you want to use ETH2 CN45 as well, refer to the below settings.

- Set SW2-6 OFF

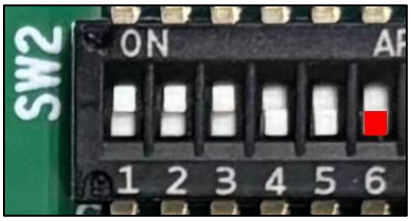


Figure 11 SW2-6 OFF

- Connect pins 2 and 3 on CN9 with a jumper.



Figure 12 Pins 2 and 3 on CN9

- Set SW6-2 ON and SW6-3 OFF

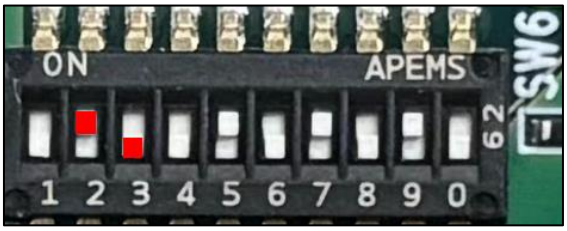


Figure 13 SW6-2 and SW6-3

4.3.6 How to set for Ethernet Port as RGMII and MII

When switching the Ethernet ports to RGMII or MII modes, you need to change some board configuration. Please refer to the manual below for instructions on how to modify it.

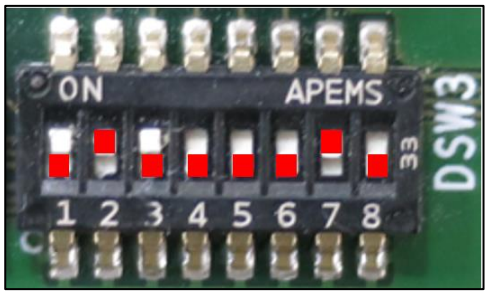
- RZ/T2H Evaluation Board User's Manual

4.4 Board Settings of RZ/N2H

4.4.1 How to set boot mode of RZ/N2H

(1) SCIF Download Mode

Set the DSW3 settings as follows. This mode is used to write bootloaders.

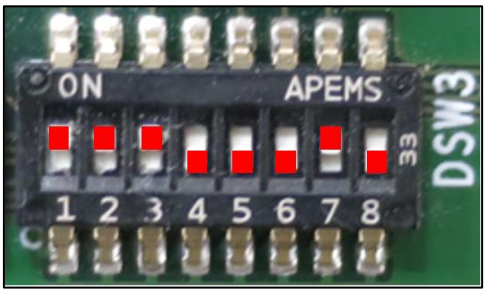


DSW3-1	OFF
DSW3-2	ON
DSW3-3	OFF
DSW3-4	OFF
DSW3-5	OFF
DSW3-6	OFF
DSW3-7	ON
DSW3-8	OFF

Figure 14. DSW3 (1)

(2) xSPI Boot Mode

Set the DSW3 settings as follows. This mode is used to boot Linux.



DSW3-1	ON
DSW3-2	ON
DSW3-3	ON
DSW3-4	OFF
DSW3-5	OFF
DSW3-6	OFF
DSW3-7	ON
DSW3-8	OFF

Figure 15. DSW3 (2)

4.4.2 How to use debug serial (console ouput)

Connect USB Type Mini-B cable to CN27: USB Type Mini-B Connector.

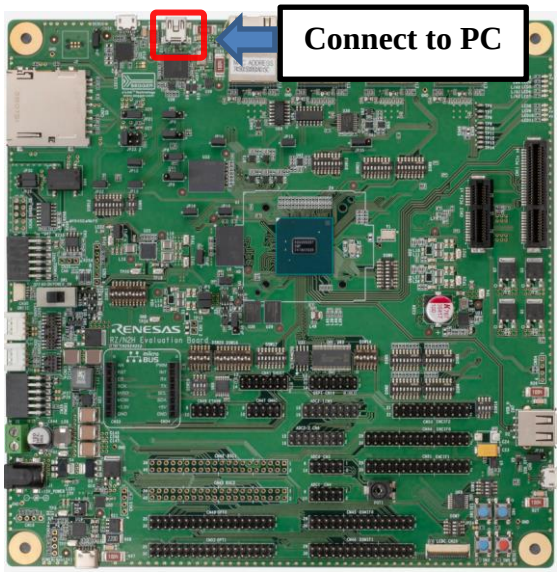


Figure 16. Connecting console for debug

4.4.3 How to set CN20 for the HDMI output (Optional)

Please connect the flexible cable and the connector correctly. When the step is wrong, the HDMI output fails. And set DIP switch DSW15-8 to ON. This step is optional. Skip this step when you do not use the HDMI output.

- Connect the flexible cable to the CN1 on 24bit to HDMI module. Install it with the side that does not have any silk facing upwards.

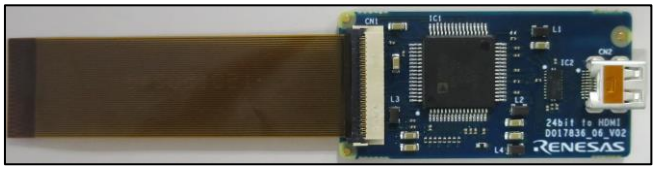


Figure 17. Setting of the CN20

- Connect the flexible cable to CN20 on the evaluation board as the image below. When installing it, ensure that you do not install it in the wrong orientation.

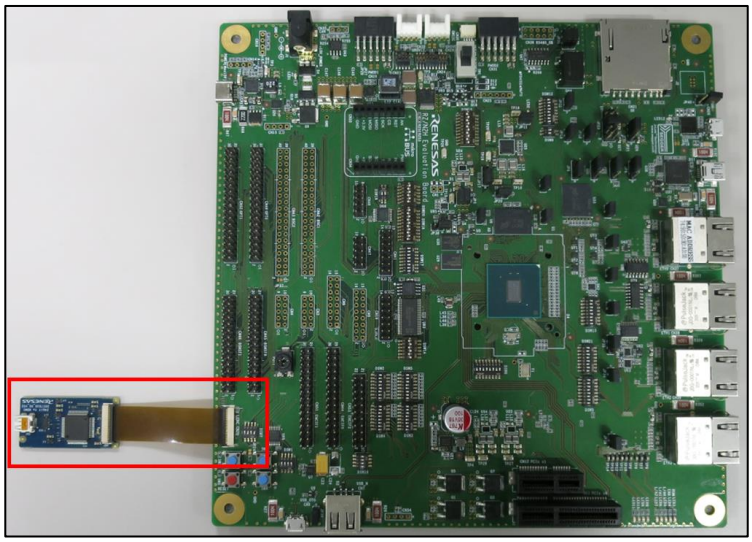


Figure 18. How to connect the module

- Set DSW15-8 to ON.

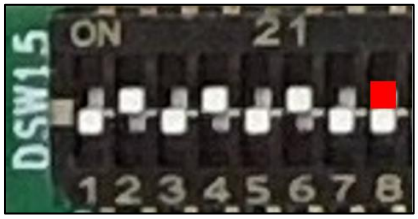


Figure 19. SW15-8

4.4.4 How to enable the ethernet port of ETH2 CN39 (Optional)

If you want to use ETH2 CN39, refer to the following settings.

- Set DSW5-6 OFF

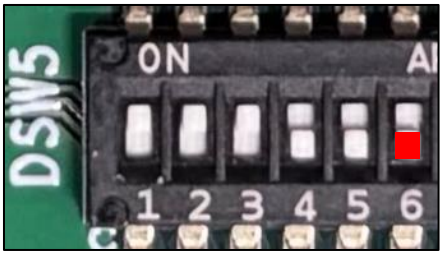
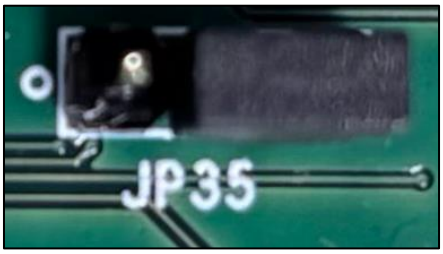


Figure 20 SW2-6 OFF

- Connect pins 2 and 3 on JP35 with a jumper.



4.4.5 How to set for Ethernet Port as RGMII and MII

When switching the Ethernet port to RGMII or MII modes, no board configuration is required, as the board is designed to support both MII and RGMII by default.

4.5 Startup Procedure of RZ/T2H and RZ/N2H

4.5.1 Power Supply

1. Connect USB-PD Power Charger to USB Type-C Connector (RZ/T2H: CN46, RZ/N2H: CN13).
2. LED lights up. (RZ/T2H: LED12, RZ/N2H: LED1)
3. Turn ON the power switch to supply the power. (RZ/T2H: SW16, RZ/N2H: SW1)
4. LED lights up. (RZ/T2H: LED11, RZ/N2H: LED2)

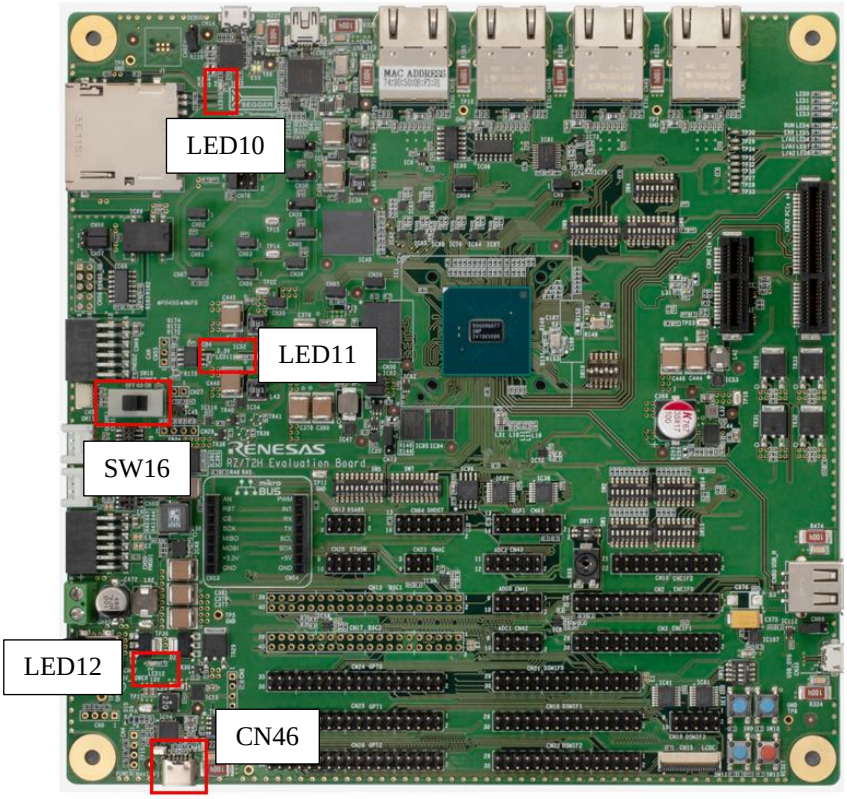


Figure 21. RZ/T2H Board example

4.5.2 Building files to write

The evaluation board uses the files in the below Table as the boot loaders. The boot loaders files are generated by the build instructions in section 2.1. Once the build is finished, copy these files to a PC which runs serial terminal software.

Table 7. File names of Boot loader

Board	File name of Boot loader
RZ/T2H Evaluation Board	• bl2_bp_xspi0-rzt2h-dev.srec • fip-rzt2h-dev.srec
RZ/N2H Evaluation Board Kit	• bl2_bp_xspi0-rzn2h-dev.srec • fip-rzn2h-dev.srec

4.5.3 Settings of Tera Term and evaluation board

(1) Connect between the board and a control PC by USB serial cable.

(2) Bring up the terminal software and select the "File" > "New Connection" to set the connection on the software.

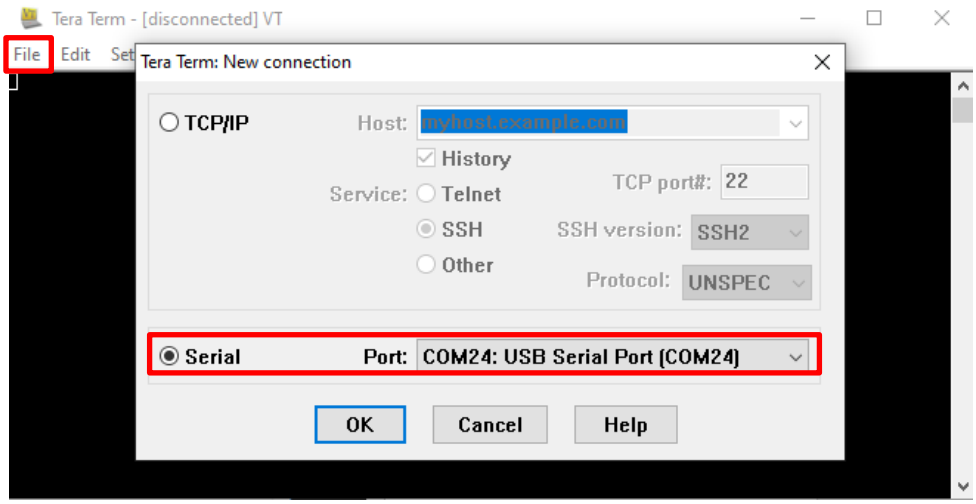


Figure 22. Set "Serial"

(3) Set the serial port

Select the "Setup" > "Serial port" to set the settings about serial communication protocol on the software. Set the settings about serial communication protocol on a terminal software as below:

- Speed: 115200 bps
- Data: 8bit
- Parity: none
- Stop bit: 1 bit
- Flow control: none
- Trasmit delay: 0 msec/line

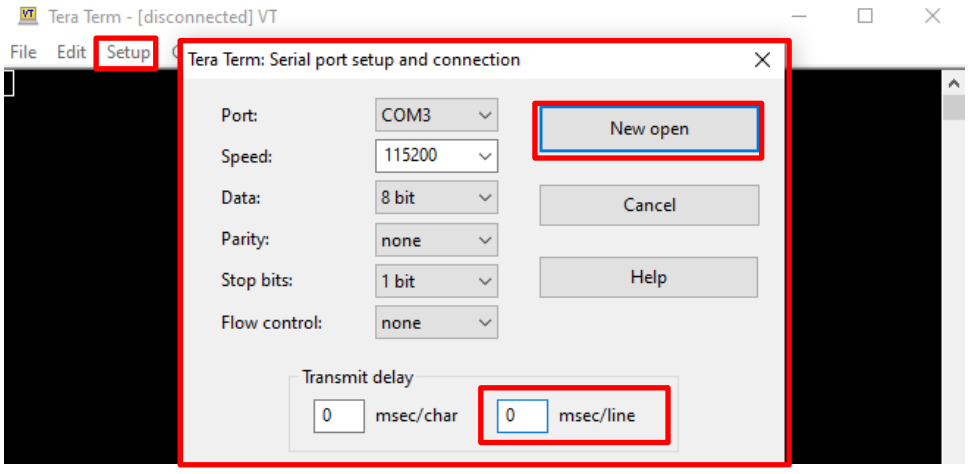


Figure 23. Terminal Setting

(4) To set the board to SCIF Download mode, set the SW14/DSW3 as below (please refer to section 4.2.1):

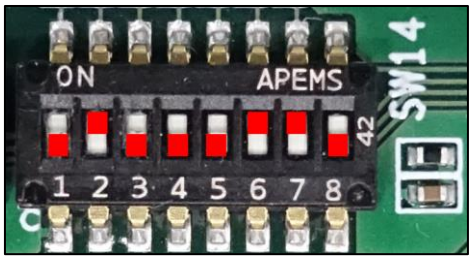


Figure 24. SW14 on RZ/T2H

SW14-1	OFF
SW14-2	ON
SW14-3	OFF
SW14-4	OFF
SW14-5	OFF
SW14-6	ON
SW14-7	ON
SW14-8	OFF

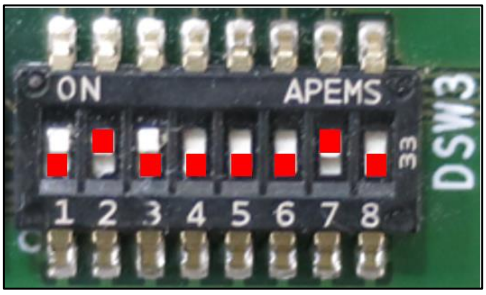


Figure 25. DSW3 on RZ/N2H

DSW3-1	OFF
DSW3-2	ON
DSW3-3	OFF
DSW3-4	OFF
DSW3-5	OFF
DSW3-6	OFF
DSW3-7	ON
DSW3-8	OFF

(5) After finishing all settings, turn on power (RZ/T2H: SW16, RZ/N2H: SW1), the messages below are displayed on the terminal.

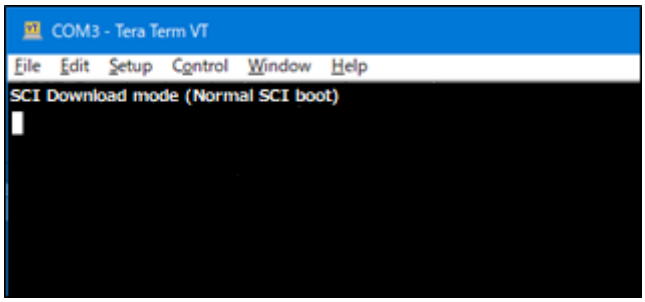


Figure 26. Terminal log of SCIF Download mode

4.6 Download Flash Programmer to RAM

Turn on the power switch of the board (RZ/T2H: SW16, RZ/N2H: SW1). The messages below are shown on the terminal.

```
SCI Download mode (Normal SCI boot)
```

Send 2 images of Flash Programmer using terminal software after the message “please send !” is shown. Send “**HDR NM**” firstly, then send “**Flash_Programmer_SCIF_CR52_RZT2H_EVK.mot**” secondary.

Below is a sample procedure with Tera Term.

Drag “**HDR NM**” file and drop the file to the TeraTerm window, then click “OK”.

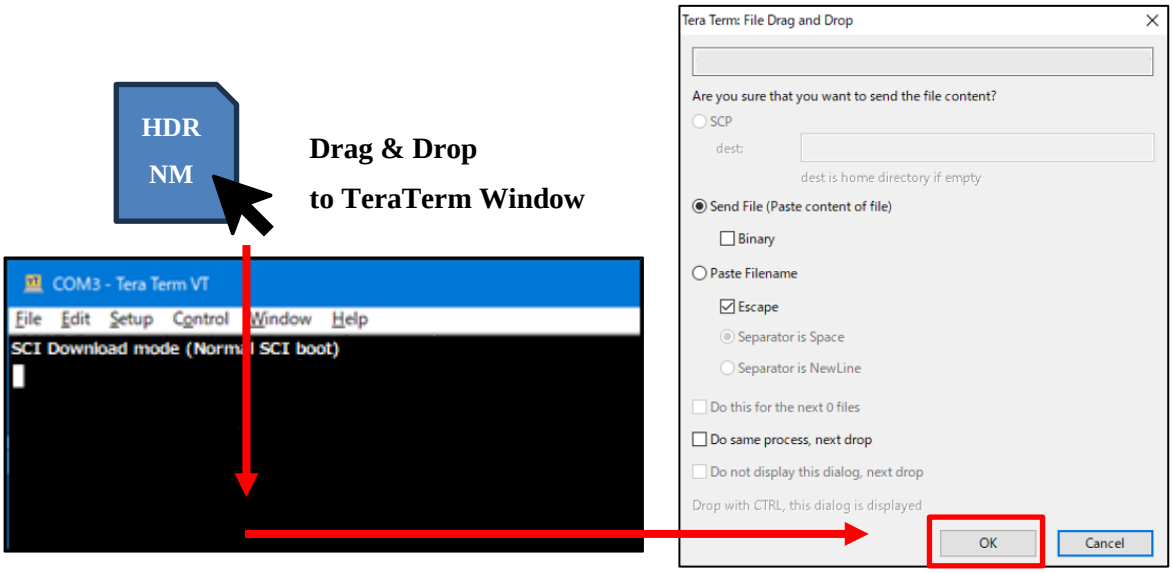


Figure 27. Send File

The image will be sent to the board via serial connection. After completing that, send “**Flash_Programmer_SCIF_CR52_RZT2H_EVK.mot**” as same as “HDR NM”. Send it as soon as possible.

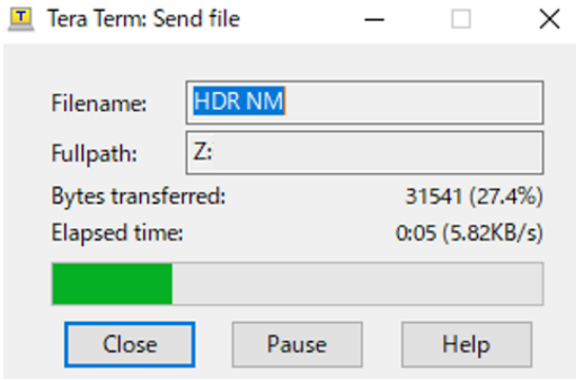


Figure 28. Sending File

When the transfer is complete, please transfer the file of “**Flash_Programmer_SCIF_CR52_RZT2H_EVK.mot**” using the same procedure within 10 seconds. If not completed within 10 seconds, please restart the transfer from the “HDR NM" file.

After successfully downloading the binary, Flash Programmer starts automatically and shows a message like below on the terminal.

```
SCI Download mode (Normal SCI boot)
-- Load Program to RAM  ---------------
-- Start Boot Program on RAM ---------

(Flash Programmer V1.04 (RZT2H)(CR52_0) (built: Oct  9 2024, 21:59:10)
```

4.7 Write the Bootloader

For the boot operation, two boot loader files need to be written to the target board.

"XSPIW" command of Flash Programmer is used to write boot loader binary files. This command receives binary data from the serial port and writes the data to a specified address of the Flash ROM with information where the data should be loaded on the address of the main memory.

```
> XSPIW 0 0x0 0x3c600
send file
```

Send the data of “bl2_bp_xspi0-rzt2h-dev.srec” or “bl2_bp_xspi0-rzn2h-dev.srec” from terminal software after the message “send file” is shown.

After successfully download the binary, messages like below are shown on the terminal.

```
Erased
Writen
```

```
SPI Data Clear(H'FF) Check : H'00000000-0000FFFF,Clear OK?(y/n)
```

In case a message to prompt to clear data like above, please enter “y”.

Next, write another loader file by using the XSPIW command again.

```
> XSPIW 0 0x060000 0xAB900
send file
```

Send the data of “fip-rzt2h-dev.srec” or “fip-rzn2h-dev.srec” from terminal software after the message “send file” is shown.

After successfully downloading the binary, messages like below are shown on the terminal.

```
Erased
Writen
```

After writing two loader files normally, turn off the power switch of the board (RZ/T2H: SW16, RZ/N2H: SW1).

4.8 Change Back to Normal Boot Mode

To set the board to SPI Boot mode, set the SW14 as below:

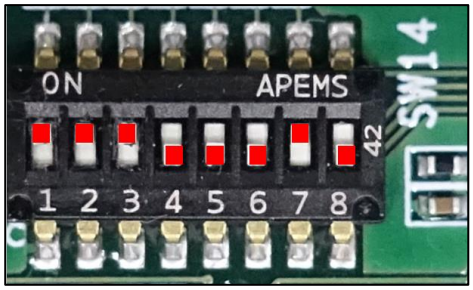


Figure 29. SW14 on RZ/T2H

SW14-1	ON
SW14-2	ON
SW14-3	ON
SW14-4	OFF
SW14-5	OFF
SW14-6	OFF
SW14-7	ON
SW14-8	OFF

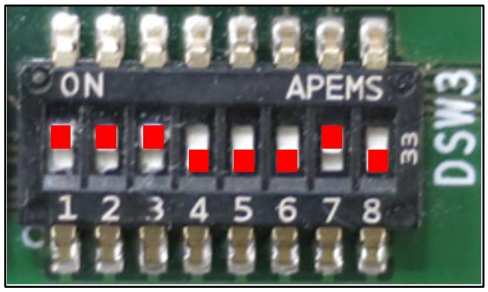


Figure 30. DSW3 on RZ/N2H

DSW3-1	ON
DSW3-2	ON
DSW3-3	ON
DSW3-4	OFF
DSW3-5	OFF
DSW3-6	OFF
DSW3-7	ON
DSW3-8	OFF

Turn on the power switch of the board (RZ/T2H: SW16, RZ/N2H: SW1).

```
U-Boot 2021.10 (Sep 30 2024 - 11:52:22 +0000)

CPU: Renesas Electronics RZ/T2H
Model: Renesas Development EVK based on r9a09g077m44
DRAM:  7.9 GiB
MMC:   mmc@92080000: 0, mmc@92090000: 1
Loading Environment from MMC... OK
In:    serial@80005000
Out:   serial@80005000
Err:   serial@80005000
Net:
Error: ethernet@92000000 address not set.
eth1: ethernet@92010000
Hit any key to stop autoboot:  0
=>
```

Following the messages above, many warning messages will be shown. These warnings are eliminated by setting correct environment variables. Please set default value and save them to the Flash ROM.

```
=> env default -a
## Resetting to default environment
=> saveenv
Saving Environment to MMC... Writing to MMC(0)….OK
=>
```

5. Booting and Running Linux

This chapter explains how to boot Linux on the evaluation board. The explanation is based on RZ/T2H as an example.

5.1 Set micro-SD card

Set micro-SD card to slot CN16 on the back of the evaluation board.

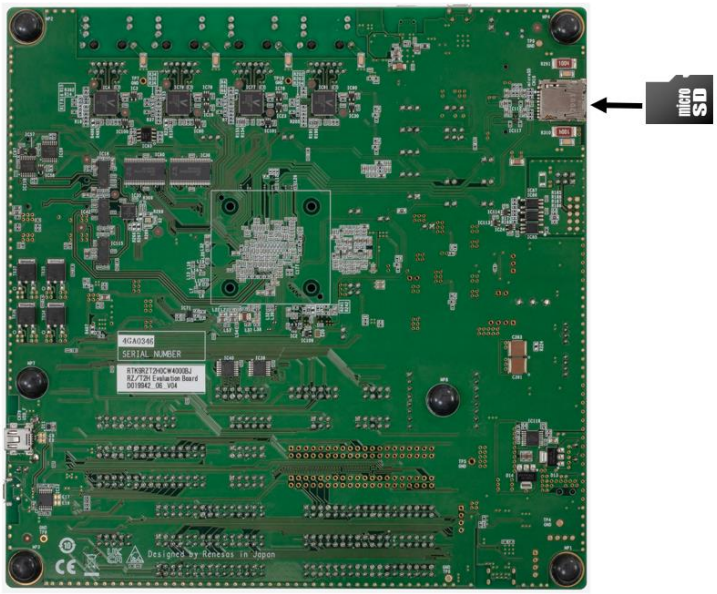


Figure 31. Set micro-SD card to the evaluation board

Now the board can boot up normally. Please turn off and on the power again to boot up the board.

5.2 Power on the board and Startup Linux

After obtaining your reference board, please be sure to follow the document and write the bootloaders to the Flash ROM before starting the evaluation.

Before booting the board, please be sure to confirm the bootloaders which are built with your BSP are written to your board.

```
U-Boot 2021.10 (Sep 30 2024 - 11:52:22 +0000)

CPU: Renesas Electronics RZ/T2H
Model: Renesas Development EVK based on r9a09g077m44
DRAM:  7.9 GiB
MMC:   mmc@92080000: 0, mmc@92090000: 1
Loading Environment from MMC... OK
In:    serial@80005000
Out:   serial@80005000
Err:   serial@80005000
Net:
Error: ethernet@92000000 address not set.
eth1: ethernet@92010000
Hit any key to stop autoboot:  1
 0
## Resetting to default environment
```

```
switch to partitions #0, OK
mmc1 is current device
18305536 bytes read in 726 ms (24 MiB/s)
34282 bytes read in 4 ms (8.2 MiB/s)
## Flattened Device Tree blob at c5f00000
   Booting using the fdt blob at 0xc5f00000
   Loading Device Tree to 00000000cfff4000, end 00000000cffff5e9 ... OK
Overlap found: 0x240000000..0x3fffffff / 0x240000000..0x3fffffff

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[    0.000000] Linux version 5.10.145-cip17-yocto-standard (oe-user@oe-host) (aarch
 :
 :
Poky (Yocto Project Reference Distro) 3.1.31 rzt2h-dev ttySC0

BSP: RZT2H/RZT2H-DEV/1.0.0
LSI: RZT2H
Version: 1.0.0
rzt2h-dev login: root
root@rzt2h-dev:~#
```

Note: As a reference, this log is written with the BSP v1.0.0.

5.3 Shutdown the Board

To power down the system, follow the step below.

(1) Run shutdown command

Run shutdown command on the console as below. After that, the shutdown sequence will start.

```
root@rzt2h-dev:~# shutdown -h now
```

Note: Run this command during the power-off sequence on rootfs.

(2) Confirm that the power is off and turn off the power switch

After executing the shutdown command, you will see the "reboot: Power down" message. Then, turn off the "POWER" SW16 (RZ/T2H).

6. Building the SDK

To build Software Development Kit (SDK), run the commands below after the steps (1) – (3) of the section 2.1 are finished.

The SDK allows you to build custom applications outside of the Yocto environment, even on a completely different PC. The results of the commands below are 'installer' that you will use to install the SDK on the same PC, or a completely different PC.

For building general applications:

```
$ cd ~/rzt2h_n2h_bsp_<package version>/build
$ MACHINE=<board> bitbake core-image-<target> -c populate_sdk
```

The resulting SDK installer will be located in "**build/tmp/deploy/sdk/**"

The SDK installer will have the extension .sh
To run the installer, you would execute the following command:

```
$ sudo sh poky-glibc-x86_64-core-image-<target>-aarch64-rzt2h-dev-toolchain-<version>\
.sh
```

Or

```
$ sudo sh poky-glibc-x86_64-core-image-<target>-aarch64-rzn2h-dev-toolchain-<version>\
.sh
```

7. Application Building and Running

This chapter explains how to make and run an application with this package. The explanation is based on RZ/T2H as an example.

7.1 Make an application

Here is an example of how to make an application running on BSP. The following steps will generate the “Hello World” sample application.

Note that you must build (bitbake) a core image for the target and prepare SDK before making an application.

7.1.1 How to extract SDK

(1) Install toolchain on a Host PC:

```
$ cd ~/rzt2h_n2h_bsp_<package version>
$ sudo sh ./build/tmp/deploy/sdk/poky-glibc-x86_64-core-image-<target>-aarch64-rzt2h-\
dev-toolchain-<version>.sh
```

Note:

sudo is optional in case user wants to extract SDK into a restricted directory (such as: /opt/).

If the installation is successful, the following messages will appear:

```
/mnt/host$ sudo sh ./build/tmp/deploy/sdk/poky-glibc-x86_64-core-image-<target>-aarch6
4-rzt2h-dev-toolchain-x.x.xx.sh
Poky (Yocto Project Reference Distro) SDK installer version x.x.xx
==========================================================================
Enter target directory for SDK (default: /opt/poky/x.x.xx):
You are about to install the SDK to "/opt/poky/x.x.xx". Proceed [Y/n]? Y
Extracting SDK.....................................................................
.....................................................done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the envir
onment setup script e.g.
 $ . /opt/poky/x.x.xx/environment-setup-aarch64-poky-linux
 $ . /opt/poky/x.x.xx/environment-setup-armv7vet2hf-neon-vfpv4-pokymllib32-linux-gnuea
bi
```

Note: This is the log of RZ/T2H Evaluation Board.

(2) Set up cross-compile environment:

```
$ source /<Location in which SDK is extracted>/environment-setup-aarch64-poky-linux
```

Note:

User needs to run the above command once for each login session.

```
$ source /opt/poky/x.x.xx/environment-setup-aarch64-poky-linux
```

7.1.2 How to build Linux application

(1) Make a work directory for the application on the Linux host PC.

```
$ mkdir ~/hello_apl
$ cd ~/hello_apl
```

(2) Make the following two files (an application file and Makefile) in the directory for the application. Here, the application is made by automake and autoconf.

- main.c

```
#include <stdio.h>
/* Display "Hello World" text on terminal software */
/* RZ/T2H Linux Start-up Guide */
/* R01US0682EJ0094 Rev.0.94 Page 29 of 42 */
/* Jul 24, 2024 */
int main(int argc, char** argv)
{
printf("\nHello World\n");
return 0;
}
```

- Makefile

```
APP = linux-helloworld
SRC = main.c
all: $(APP)
CC ?= gcc
# Options for development
CFLAGS = -g -O0 -Wall -DDEBUG_LOG
$(APP):
  $(CC) -o $(APP) $(SRC) $(CFLAGS)
install:
  install -D -m755 $(APP) $(DESTDIR)/home/root/$(APP)
clean:
  rm -rf $(APP)
```

(3) Make the application by the generated makefile.

```
$ make
```

```
renesas@49d1d5882435:/mnt/host/linux-helloworld$ make
aarch64-poky-linux-gcc -mtune=cortex-a55 -fstack-protector-strong -D_FORTIFY_SOURCE=
2 -Wformat -Wformat-security -Werror=format-security --sysroot=/opt/poky/3.1.31/sysroo
ts/aarch64-poky-linux -o linux-helloworld main.c -g -O0 -Wall -DDEBUG_LOG
renesas@49d1d5882435:/mnt/host/linux-helloworld$ ls -l
total 24
-rwxr-xr-x 1 renesas renesas 15320 May 10 23:32 linux-helloworld
-rw-r—r—1 renesas renesas 148 Sep 24 2020 main.c
-rw-r—r—1 renesas renesas 250 Sep 24 2020 Makefile
```

After making, confirm that the execute application (the sample file name is “linux-helloworld”) is generated in the hello_apl folder. Also, this application must be cross compiled for Aarch64. Mar.17.2023 4.2 Store a sample application.

The sample application could be written by the following procedure. The application should be stored in the ext3 partition.

```
$ sudo mount /dev/sdb2 /media/
$ cd /media/usr/bin
$ sudo cp ~/hello_apl/linux-helloworld .
$ sudo chmod +x linux-helloworld
```

Notes: 1. “sdb2” (above in red) may depend on using system.
2. is an optional directory name to store the application.

7.2 Run a sample application

Power on the evaluation board kit and start the system. After booting, run the sample application with the following command.

```
BSP: RZT2H/RZT2H-DEV/<package version>
LSI: RZT2H
Version: <package version>
rzt2h-dev login: root
root@rzt2h-dev:~# /usr/bin/linux-helloworld

Hello World
root@rzt2h-dev:~#
```

8. GUI Application Framework (Optional)

This chapter explains how to enable the GUI application framework. LVGL and Qt are options for the GUI application framework, enabling you to display a graphical user interface on the HDMI monitor. The steps outlined in this chapter are optional, so skip this section if it is not required for your purposes.

8.1 LVGL

When LVGL is required, follow the steps below to build LVGL after completing step (1) of section 2.1.

(1) Git clone the recipe of LVGL

Run the below commands to download meta-rz-demo-lvgl.

```
$ cd ~/rzt2h_n2h_bsp_<package version>
$ git clone https://github.com/renesas-rz/meta-rz-demo-lvgl.git -b v8.3.0_u1
```

(2) Build Initialize

Initialize a build using the 'oe-init-build-env' script in Poky and point TEMPLATECONF to platform conf path.

```
$ TEMPLATECONF=$PWD/meta-renesas/meta-rzt2h/docs/template/conf/ source \
poky/oe-init-build-env build
```

(3) Enable the LVGL demo layer

Run the below command to add the LVGL demo layer. The step adds the setting to "~/rzt2h_n2h_bsp_<package version>/build/conf/bblayers.conf".

```
$ bitbake-layers add-layer ../meta-rz-demo-lvgl
```

(4) Add the LVGL to the local.conf

Add below lines to "~/rzt2h_n2h_bsp_<package version>/build/conf/local.conf" to enable the LVGL demo and HDMI.

```
IMAGE_INSTALL_append = " lvgldemo "
```

After that, move to step (3) of section 2.1 and start a build.

(5) Prepare the SD card and the board environment

Follow section 3 and section 4 to prepare the board environment and the micro-SD card.

(6) Boot and Run Linux

Refer to section 5 and power on your evaluation board.

(7) Run the LVGL demo

Run the below commands to launch the LVGL demo on a HDMI monitor. When you want to know the usage of this demo, run “lvgldemo -h”.

```
root@rzt2h-dev:~# lvgldemo
```

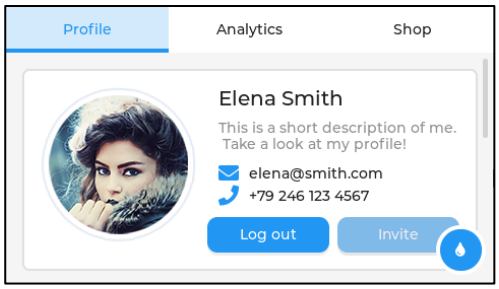


Figure 32. LVGL Demo (Widgets demo)

When you boot Linux with Weston, run the below command to stop Weston. When you launch the LVGL demo in the Weston environment, the demo performace is slow.

```
root@rzt2h-dev:~# systemctl stop weston@root
```

8.2 Qt

When Qt is needed, follow the following steps to build Qt after step (1) of section 2.1.

(1) Build Initialize

Initialize a build using the 'oe-init-build-env' script in Poky and point TEMPLATECONF to platform conf path.

```
$ cd ~/rzt2h_n2h_bsp_<package version>/
$ git clone https://github.com/meta-qt5/meta-qt5.git
$ cd meta-qt5
$ git checkout -b tmp c1b0c9f546289b1592d7a895640de103723a0305
$ git cherry-pick 77b6060cef9337b184100083746c2e35f531be74
$ cd ..
$ TEMPLATECONF=$PWD/meta-renesas/meta-rzt2h/docs/template/conf/ source \
poky/oe-init-build-env build
```

(2) Enable the Qt layer and the Qt demo layer

Run the command below to add the Qt layer and the Qt demo layer. The step adds the setting to "~/rzt2h_n2h_bsp_<package version>/build/conf/bblayers.conf".

```
$ bitbake-layers add-layer ../meta-qt5
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-demos
```

(3) Add the Qt setting to the local.conf

Add below lines to enable the Qt demo and HDMI to "~/rzt2h_n2h_bsp_<package version>/build/conf/local.conf".

```
QT_DEMO = "1"
IMAGE_INSTALL_append += " netdisp "
```

(4) Start the build with core-image-qt

Run the command below and start the build.

```
$ MACHINE=<board> bitbake core-image-qt
```

<board> can be selected by referring to the table below.

Table 8. List of the platforms and the boards

Renesas MPU	<board>
RZ/T2H	rzt2h-dev
RZ/N2H	rzn2h-dev

After the building is successfully completed, a similar output will be seen, and the command prompt will return.

```
NOTE: Tasks Summary: Attempted 7427 tasks of which 16 didn't need to be rerun and all
succeeded.
```

All necessary files listed in "Table 4. Image files" will be generated by the bitbake command and will be in the **"build/tmp/deploy/images"** directory.

(5) Prepare the SD card and the board environment

Follow section 3 and section 4 to prepare the board environment and the micro-SD card.

Note) Since the build command is " MACHINE=<board> bitbake core-image-qt ", then the WIC file needs to change to "sudo bmaptool copy core-image-qt-rzt2h-dev.wic.gz /dev/sdb".

(6) Boot and Run Linux

Refer to section 5 and power on your evaluation board.

(7) Launch the Qt demo

Run the command below to launch the Qt demo on a HDMI monitor.

```
root@rzt2h-dev:~# netdisp
```

9. Appendix

9.1 Ethernet port

(1) Check the Ethernet port on the evaluation board and in Linux

Please note that the Ethernet ports on the evaluation board are assigned in Linux as shown in the table below. “eth0” in Linux is used for Ethernet Switch management.

Table 9. Port name on Linux

Ethernet Port on the evaluation board	Hardware	Linux Ethernet Port
ETH0	GMAC0	lan0
ETH1	GMAC0	lan1
ETH2	GMAC2	Eth1
ETH3	GMAC1	Eth0

If you want to check it on Linux, run the ifconfig command after booting Linux. The following is an example.

```
root@rzt2h-dev:~# ifconfig

eth0      Link encap:Ethernet  HWaddr 3E:62:39:5E:33:D4
          inet6 addr: fe80::3c62:39ff:fe5e:33d4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1508  Metric:1
          RX packets:1042 errors:0 dropped:0 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:177828 (173.6 KiB)  TX bytes:6224 (6.0 KiB)
          Interrupt:102


eth1      Link encap:Ethernet  HWaddr 74:90:50:00:00:31
          inet addr:192.168.5.69  Bcast:192.168.5.255  Mask:255.255.255.0
          inet6 addr: fe80::7690:50ff:fe00:31/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15982 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9726 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:18150303 (17.3 MiB)  TX bytes:1472228 (1.4 MiB)
          Interrupt:105


lan0      Link encap:Ethernet  HWaddr 74:90:50:00:00:02
          inet addr:192.168.5.19  Bcast:192.168.5.255  Mask:255.255.255.0
          inet6 addr: fe80::7690:50ff:fe00:2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:504 errors:0 dropped:6 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:72348 (70.6 KiB)  TX bytes:1328 (1.2 KiB)


lan1      Link encap:Ethernet  HWaddr 74:90:50:00:00:30
          inet addr:192.168.5.68  Bcast:192.168.5.255  Mask:255.255.255.0
          inet6 addr: fe80::7690:50ff:fe00:30/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:469 errors:0 dropped:9 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:67320 (65.7 KiB)  TX bytes:1370 (1.3 KiB)
```

If you want to change the name of network interfaces on Linux, modify the interface name in the device tree source.

(2) Enable ETH0 (lan0) and ETH1 (lan1) on Linux

"ETH0" and "ETH1" on the evaluation board are enabled by the build steps in section 2.1. To use them on Linux, you need to enable the network interfaces and obtain an IP address. The following commands provide examples of how to enable interfaces and acquire an IP address.

```
root@rzt2h-dev:~# ifconfig lan0 up
root@rzt2h-dev:~# udhcpc -i lan0
root@rzt2h-dev:~# ifconfig lan1 up
root@rzt2h-dev:~# udhcpc -i lan1
```

"ETH2" and "ETH3" on the evaluation board are enabled on Linux as "eth1 (=ETH2)" and "eth0(=ETH3)" by default.

(3) Check Ethernet port on Linux

To check the ethernet port, run the following command as a reference.

```
$ ping -I eth1 <IP Address>
```

To check the performance of the ethernet port, iperf3 command can be used for one of the checking tools. To install the command, add the following line to "~/rzt2h_n2h_bsp_<package version>/build/conf/local.conf".

```
IMAGE_INSTALL_append = " iperf3"
```

9.2 Docker

This section explains how to build the BSP with Docker enabled and how to obtain and run the "hello-world" image.

(1) Add layers to enable Docker

After completing step (2) in section 2.1, run the following commands. Once executed, proceed until step (3) in section 2.1 to build the BSP with Docker enabled.

```
$ cd ~/rzt2h_n2h_bsp_<package version>/build
$ bitbake-layers add-layer ../meta-openembedded/meta-filesystems
$ bitbake-layers add-layer ../meta-openembedded/meta-networking
$ bitbake-layers add-layer ../meta-virtualization
```

(2) Boot the evaluation board

Follow chapters 4 and 5, boot the evaluation board, and proceed to the next step.

(3) Check Docker on the evaluation board (Optional)

After booting the board, optionally run the following commands to display the status of the Docker Daemon and its version.

```
root@rzt2h-dev:~# systemctl status docker.service
root@rzt2h-dev:~# docker version
```

(4) Hello world

Run the commands below to get the docker image of hello-world and run the image.

```
root@rzt2h-dev:~# docker pull hello-world
root@rzt2h-dev:~# docker run hello-world
Hello from Docker!
```

9.3 How to boot from eMMC

This section explains how to boot Linux from eMMC mode.

(1) Writing Bootloader for eMMC Boot

For the boot operation, EXT_CSD register of eMMC needs to be modified and two boot loader files need to be written to the target board. Modifying register address and value information are described in the following table.

After booting the Flash Programmer (Refer to “4.6 Download Flash Programmer to RAM”),

Then, "EMMCW" command of Flash Programmer is used to write boot loader binary files. This command receives binary data from the serial port and writes the data to a specified address of the eMMC with information where the data should be loaded on the address of the main memory.

Table 10. Address of EXT_CSD register of eMMC for eMMC boot

Address	Value to write
0xB1	0x02
0xB3	0x08

```
SCI Download mode (Normal SCI boot)
-- Load Program to RAM  ---------------
-- Start Boot Program on RAM ---------

(Flash Programmer V1.03 (RZT2H)(CR52_0) (built: Oct  9 2024, 21:59:10)
> EMMCWECSD 0xb3 0x9
 EXT_CSD[b3] = 0x9
> EMMCW 0x1 0x0
send file
```

Send the data of “bl2_bp_emmc-rzt2h-dev.srec” from terminal software after the message “send file” is shown.

After successfully downloading the binary, messages like below are shown on the terminal.

```
EMMCW Complete!
```

Next, write another loader file by using EMMCW command again.

```
> EMMCW 0x300 0x0
send file
```

Send the data of “fip-rzt2h-dev.srec” from terminal software after the message “send file” is shown.

After successfully downloading the binary, messages like below are shown on the terminal.

```
EMMCW Complete!
```

Then modify EXT_CSD register.

```
> EMMCWECSD 0xb3 0x8
 EXT_CSD[b3] = 0x8
> EMMCWECSD 0xb1 0x2
 EXT_CSD[b1] = 0x2
>
```

After writing two loader files normally, turn off the power switch of the board (RZ/T2H: SW16, RZ/N2H: SW1).

(2) Create a microSD card to boot Linux for eMMC boot

Create a microSD card by “3 Preparing the SD Card”.

(3) Setting the U-boot and boot with eMMC boot mode

Please Set the SW14 settings as follows. This mode is used to boot Linux.

Table 11. SW14 settings

SW14-1	ON
SW14-2	ON
SW14-3	OFF
SW14-4	OFF
SW14-5	OFF
SW14-6	ON
SW14-7	ON
SW14-8	OFF

Set CN78 as follows.



Figure 33. CN78 Setting

Then, turn on the power switch of the board (RZ/T2H: SW16, RZ/N2H: SW1).

After booting Linux, please login as root.

```
NOTICE:  BL2: v2.7(release):
NOTICE:  BL2: Built : 14:31:25, Jul  5 2024
ERROR:   Reported Core count (0) is different from PLATFORM_CORE_COUNT (4)
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.7(release):
NOTICE:  BL31: Built : 14:31:25, Jul  5 2024


U-Boot 2021.10 (Jul 13 2023 - 01:58:15 +0000)

CPU: Renesas Electronics RZ/T2H
Model: Renesas Development EVK based on r9a09g077m44
DRAM:  7.8 GiB
MMC:   mmc@92080000: 0, mmc@92090000: 1
Loading Environment from MMC... OK
In:    serial@80005000
Out:   serial@80005000
Err:   serial@80005000
Net:
Error: ethernet@92000000 address not set.
No ethernet found.

Hit any key to stop autoboot:  0
## Resetting to default environment
switch to partitions #0, OK
mmc1 is current device
20462080 bytes read in 815 ms (23.9 MiB/s)
32727 bytes read in 4 ms (7.8 MiB/s)
Moving Image from 0x204080000 to 0x204200000, end=205600000
```

```
## Flattened Device Tree blob at 205f00000
   Booting using the fdt blob at 0x205f00000
ERROR: reserving fdt memory region failed (addr=210000000 size=10000000 flags=0)
   Loading Device Tree to 0000000213ff5000, end 0000000213ffffd6 ... OK

Starting kernel ...
…
```

9.4 How to boot from eSD

This section explains how to boot Linux from eSD mode.

(1) Build Initialize

After completing step (2) of section 2.1, initialize a build using the 'oe-init-build-env' script in Poky and point TEMPLATECONF to platform conf path.

```
$ TEMPLATECONF=$PWD/meta-renesas/meta-rzt2h/docs/template/conf/ source \
poky/oe-init-build-env build
```

(2) Add a setting to the local.conf

Add below line to "~/rzt2h_n2h_bsp_<package version>/build/conf/local.conf" to create WIC file for eSD boot.

```
WKS_DEFAULT_FILE_rzt2h = "rz-image-bootpart-esd_rzt2h.wks"
```

After that, move to step (3) of section 2.1 and start a build.

(3) Create a microSD card to boot Linux for eSD boot

Create a microSD card by following "3 Preparing the SD Card".

(4) Setting the U-boot and boot with eSD boot mode

Please Set the SW14 settings as follows. This mode is used to boot Linux.

Table 12. SW14 settings

SW14-1	OFF
SW14-2	OFF
SW14-3	ON
SW14-4	OFF
SW14-5	OFF
SW14-6	ON
SW14-7	ON
SW14-8	OFF

Then, turn on the power of the board with SW16.

After booting Linux, please login as root.

10. Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Nov. 26, 2024	-	First edition issued.
1.01	Dec. 23, 2024	-	Newly add RZ/N2H.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.