RZ/T2, RZ/N2

Device Setup Guide for Flash boot

Introduction

This application note describes how to set up the device when booting a program from external flash in xSPI boot mode or 16-bit bus boot mode, and how to write JTAG Authentication ID or SCI/USB Boot Authentication ID to the OTP.

Although this document shows how to self-program to external flash on the RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L or RZ/N2H, it is just an example of how to program to external flash and can be programmed in any way you wish.

Target Device

RZ/T2M Group

RZ/T2L Group

RZ/N2L Group

RZ/T2H Group

RZ/N2H Group

Contents

1. Overview

1.1 Introduction

The RZ/T2, RZ/N2 device setup sample program is a sample program to self-program a user program to the external flash on Renesas Starter Kit+ for RZ/T2M, RZ/T2L and RZ/N2L, or Evaluation Board for RZ/T2H and RZ/N2H.

This sample program package uses the Flexible Software Package for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H. For more information about FSP, please refer to RZ/T2, RZ/N2 Getting Started with Flexible Software Package.

1.2 Features

The sample program has the following features:

- It supports QSPI flash, OSPI flash and NOR Flash as external flash and writes the user program to the external flash via SCI or USB.
- This sample program can also program to the RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H OTP memory.

1.3 Limitations

No limitations in this sample program.

1.4 Package Contents

RZ/T2, RZ/N2 Device setup sample program package contains several files with software and tools. The following table gives an overview of its structure and its content.

Table 1.1 RZ/T2, RZ/N2 Device Setup Sample Program Package Contents

No.	File Name	Classification	Remarks
1	RZT2M_RSK_DeviceSetup_Rev230.zip	Software	RZ/T2M Sample program project Note1
2	RZT2M_RSK_DeviceSetup.out.srec	Software	Pre-built sample program (S-Record format)
3	RZT2M_bsp_led.bin	Software	Reference of user program Note2
4	RZT2M_bsp_led.zip	Software	Project for RZT2M_bsp_led.bin Note1
5	RZT2L_RSK_DeviceSetup_Rev230.zip	Software	RZ/T2L Sample program project Note1
6	RZT2L_RSK_DeviceSetup_usb.out.srec	Software	Pre-built sample program for USB communication (S-Record format)
7	RZT2L_RSK_DeviceSetup_sci_without_ospi.out.srec	Software	Pre-built sample programs for SCI communication (S-Record format) OSPI flash is disabled
8	RZT2L_bsp_led.bin	Software	Reference of user program Note2
9	RZT2L_bsp_led.zip	Software	Project for RZT2L_bsp_led.bin Note1
10	RZN2L_RSK_DeviceSetup_Rev230.zip	Software	RZ/N2L Sample program project Note1
11	RZN2L_RSK_DeviceSetup_qspi.out.srec	Software	Pre-built sample program for QSPI Flash (S-Record format)
12	RZN2L_RSK_DeviceSetup_nor.out.srec	Software	Pre-built sample program for NOR Flash (S-Record format)
13	RZN2L_bsp_led.bin	Software	Reference of user program Note2
14	RZN2L_bsp_led.zip	Software	Project for RZN2L_bsp_led.bin Note1
15	RZT2H_EVB_DeviceSetup_Rev230.zip	Software	RZ/T2H Sample program project Note1
16	RZT2H_EVB_DeviceSetup.out.srec	Software	Pre-built sample program (S-Record format)
17	RZT2H_bsp_led.bin	Software	Reference of user program for Cortex®-R52 Note2
18	RZT2H_bsp_led.zip	Software	Project for RZT2H_bsp_led.bin Note1
19	RZT2H_CA55_bsp_led.bin	Software	Reference of user program for Cortex®-A55 Note2
20	RZT2H_CA55_bsp_led.zip	Software	Project for RZT2H_CA55_bsp_led.bin Note1
21	RZN2H_EVB_DeviceSetup_Rev230.zip	Software	RZ/N2H Sample program project Note1
22	RZN2H_EVB_DeviceSetup.out.srec	Software	Pre-built sample program (S-Record format)
23	RZN2H_bsp_led.bin	Software	Reference of user program for Cortex®-R52 Note2
24	RZN2H_bsp_led.zip	Software	Project for RZN2H_bsp_led.bin Note1
25	RZN2H_CA55_bsp_led.bin	Software	Reference of user program for Cortex®-A55 Note2
26	RZN2H_CA55_bsp_led.zip	Software	Project for RZT2H_CA55_bsp_led.bin Note1
27	parameter_generator.py	Tool	Generation tool for the parameter for the loader
28	device_setup.py	Tool	Command sending tool for device setup
29	r01an6471ej0240-rzt2-n2-flashboot.pdf	Document	This document
30	r01an6622ej0240-rzt2-n2-releasenote.pdf	Document	Release Note

Note1 Contains sample program projects for GCC and IAR compilers.
Note2 Generated by project for IAR compilers.

1.5 Related Documents

Table 1.2 lists documents related to this document.

Table 1.2 Related Documents

Title	Document Number
RZ/T2M Group Renesas Starter Kit+ for RZ/T2M User's Manual	R20UT4939EG****
RZ/T2M Group Renesas Starter Kit+ for RZ/T2M Quick Start Guide	R20UT4941EG****
RZ/T2, RZ/N2 Getting Started with Flexible Software Package	R01AN6434EJ****
RZ/T2M Group User's Manual: Hardware	R01UH0916EJ****
RZ/T2L Group Renesas Starter Kit+ for RZ/T2L User's Manual	R20UT5164EJ****
RZ/T2L Group Renesas Starter Kit+ for RZ/T2L Quick Start Guide	R20UT5235EJ****
RZ/T2L Group User's Manual: Hardware	R01UH0985EJ****
RZ/N2L Group Renesas Starter Kit+ for RZ/N2L User's Manual	R20UT4984EG****
RZ/N2L Group Renesas Starter Kit+ for RZ/N2L Quick Start Guide	R20UT4986EG****
RZ/N2L Group User's Manual: Hardware	R01UH0955EJ****
RZ/T2H Group RZ/T2H Evaluation Board User's Manual	R20UT5405EJ****
RZ/T2H and RZ/N2H Groups User's Manual: Hardware	R01UH1039EJ****
RZ/N2H Group RZ/N2H Evaluation Board User's Manual	R20UT5522EJ****

1.6 Explanation of Terms

The meanings of terms used in this document are indicated below.

Term Used in This Document	Meaning of Term
OTP	One-Time Programmable Memory
password	Authentication ID during JTAG Authentication or SCI/USB Authentication
OSPI flash	Octa flash

2. Quick Start

This chapter describes how to write a user program to the external Flash on the boards and how to set up the OTP in the devices using this sample program package.

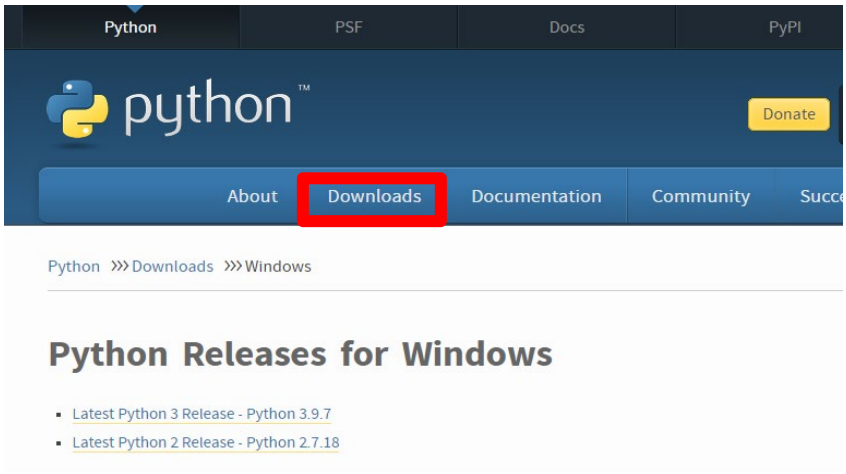
The target devices for the explanation are RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L, RZ/N2H, and the target boards are the Renesas Starter Kit+ (hereinafter referred to as RSK+) and Evaluation Board (hereinafter referred to as EVB) for these devices.

2.1 Setup

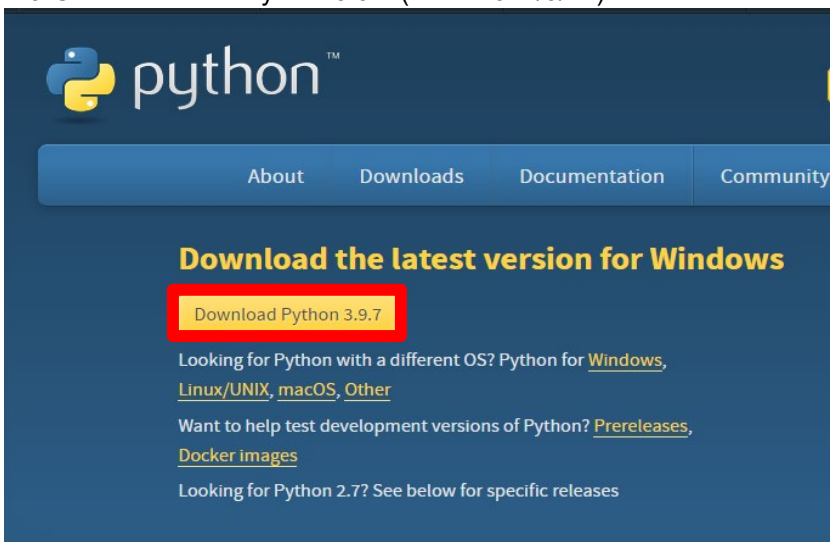
2.1.1 Install python to your PC

1. Download and install the Python 3.8 or later version
   1-1. Access https://www.python.org/downloads/windows/

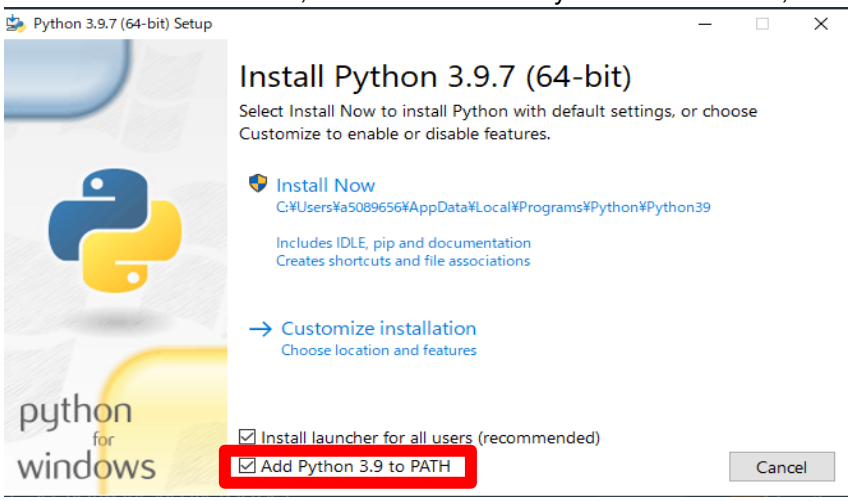
   1-2. Click Downloads.



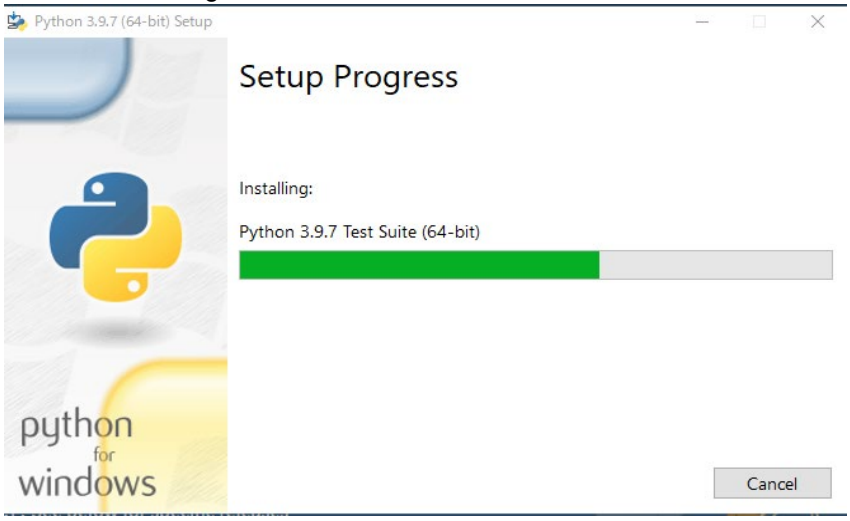
   1-3. Click Download Python 3.9.7.(As of 2021/9/17)



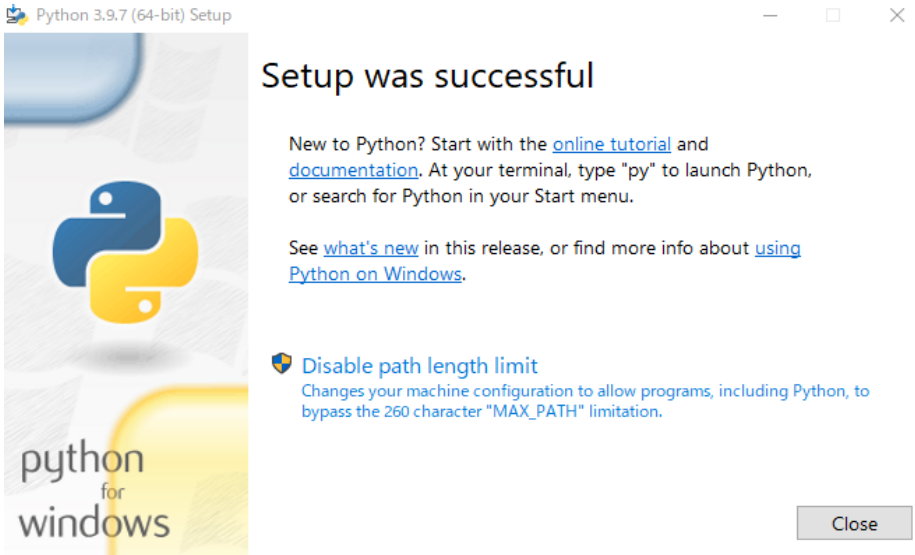
1-4. In case of new install, check box of "Add Python 3.9 to PATH", click "Install Now".



1-5. Start installing as below.



1-6. Display following message, click "Close", complete installing.



2. After installing Python, enter the following from the command prompt to install pyserial.

```
> pip install pyserial
```

2.1.2 Check the COM port on your PC

In device_setup.py, you need to specify the COM port that indicates the device for serial communication with the device on RSK+ or EVB.

Here is how to check the COM port of the device on your PC.

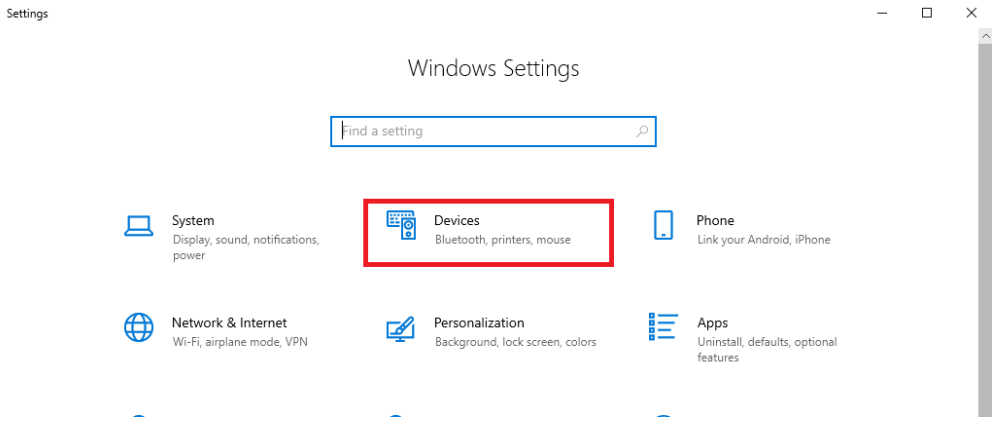
1. Set up the board.
   Refer to the following documents to supply power to the CPU board.
   - RZ/T2M Group Renesas Starter Kit+ for RZ/T2M Quick Start Guide,
   - RZ/T2L Group Renesas Starter Kit+ for RZ/T2L Quick Start Guide,
   - RZ/N2L Group Renesas Starter Kit+ for RZ/N2L Quick Start Guide,
   - RZ/T2H Group RZ/T2H Evaluation Board User's Manual,
   - RZ/N2H Group RZ/N2H Evaluation Board User's Manual.

   Refer to the following manuals for board details.
   - RZ/T2M Group Renesas Starter Kit+ for RZ/T2M User's Manual,
   - RZ/T2L Group Renesas Starter Kit+ for RZ/T2L User's Manual,
   - RZ/N2L Group Renesas Starter Kit+ for RZ/N2L User's Manual,
   - RZ/T2H Group RZ/T2H Evaluation Board User's Manual,
   - RZ/N2H Group RZ/N2H Evaluation Board User's Manual.

   Note that RSK+ for RZ/N2L requires either QSPI flash or NOR flash to be selected in the board settings. Also, with RSK+ for RZ/T2L and EVB for RZ/T2H, OSPI flash and QSPI flash can be used, but NOR flash cannot be used. Only RSK+ for RZ/T2L can use OSPI flash, but if you use it, you cannot use SCI communication, so you need to select either SCI communication or OSPI flash in the board settings.

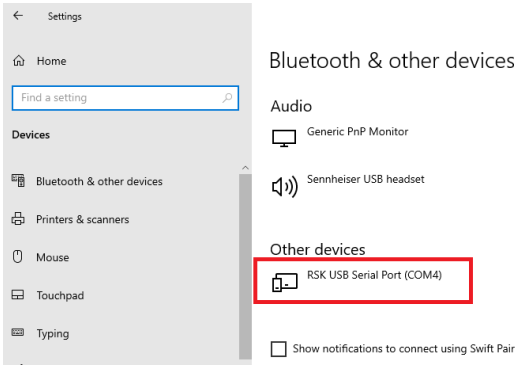
2. Connect RSK+ or EVB to your PC
   Connect RSK+ and PC as follows:
   — For setup via SCI: Connect CN16 of RSK+ to PC via USB cable.
   — For setup via USB: Connect CN11 of RSK+ to PC with USB cable.
   Connect EVB for RZ/T2H and PC as follows:
   — For setup via SCI: Connect CN34 of EVB to PC via USB cable.
   — For setup via USB: Connect CN79 of EVB to PC with USB cable.
   Connect EVB for RZ/N2H and PC as follows:
   — For setup via SCI: Connect CN27 of EVB to PC via USB cable.
   — For setup via USB: Connect CN8 of EVB to PC with USB cable.

3. How to check the COM port on your PC
   3-1. Click the Windows Start menu.
   3-2. Click Settings.
   3-3. Click the device in the setting menu.



   3-4. When the USB cable connected to the PC is connected to CN16 on RSK+, CN34 on EVB for RZ/T2H or CN27 on EVB for RZ/N2H, "RSK USB Serial Port (COMx)" or " USB Serial Port (COMx)" is displayed on other devices.

When the USB cable connected to the PC is connected to CN11 on RSK+, CN79 on EVB for RZ/T2H or CN8 on EVB for RZ/N2H, "USB Serial Device (COMx)" is displayed on other devices.



2.2 Generates binary data for programming to external flash

Use the user program included in the package (RZ*_bsp_led.bin) or follow the steps below to generate binary data.
The user program for Cortex®-R52 included in the package starts from 0x00102000, the user program for Cortex®-A55 included in the package starts from 0x10000000. If you change the starting address, replace the address description in the following chapters. The program size (binary data size) must be a multiple of 512 bytes and less than the maximum size shown in Table 2.1. If the binary data size after the user program build is not a multiple of 512 bytes, add dummy data after the binary data to adjust it to a multiple of 512 bytes. For information on how to adjust the binary data to 512-byte units by adding project settings, refer to 3.3.7.

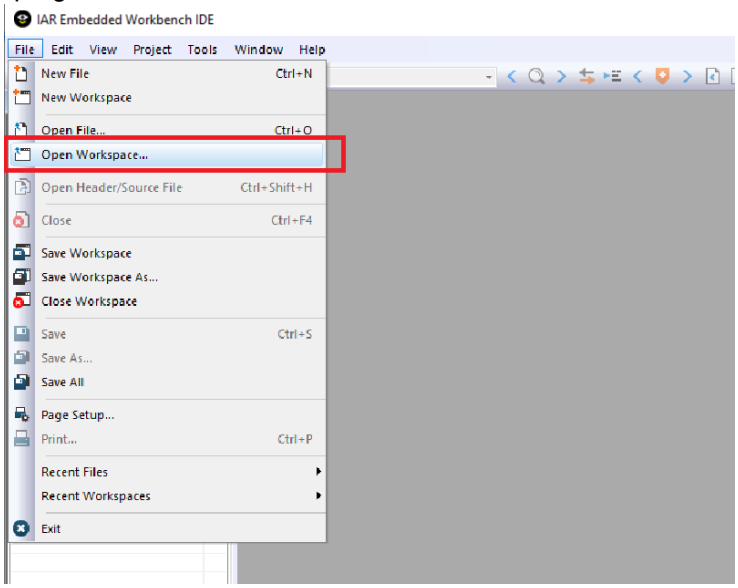
Table 2.1 Maximum user program size

Products	First booting CPU core	Maximum user program size
RZ/T2M	Cortex®-R52	56 KB
RZ/T2L	Cortex®-R52	56 KB
RZ/N2L	Cortex®-R52	120 KB
RZ/T2H and RZ/N2H	Cortex®-R52	52 KB
	Cortex®-A55	2036 KB

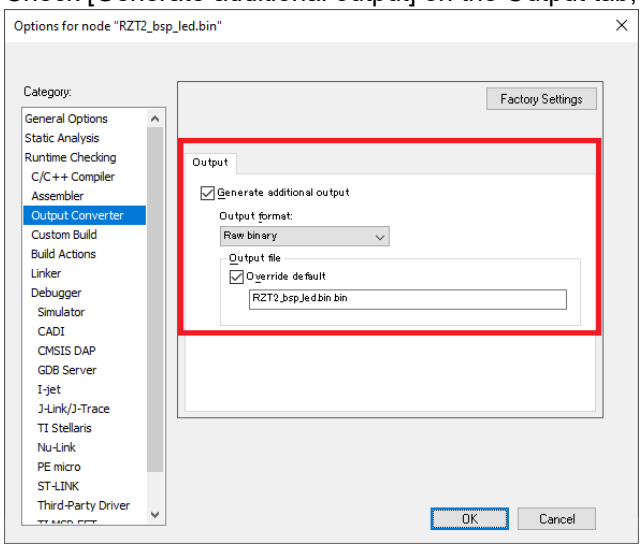
EWARM:
The following settings are also included in the user program project (RZ*_bsp_led.zip).

1. Start EWARM

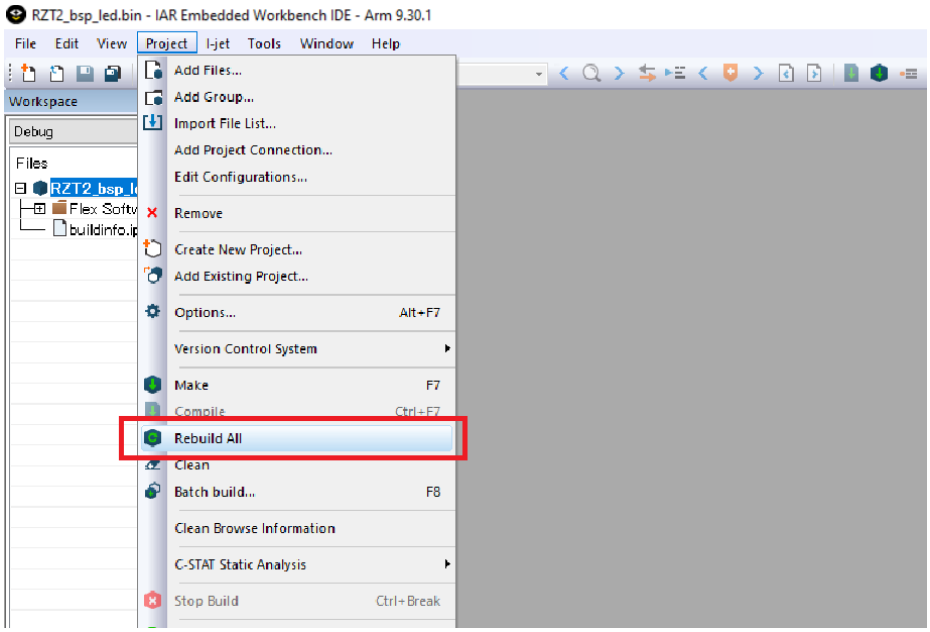
2. Select [File]-[Open Workspace] from the EWARM menu, and open project file(extension: .eww) of the program to be written.



3. Select the project option and select the output converter in the category list.
Check [Generate additional output] on the Output tab, select [Raw binary] and enter the output file name.



4. Select [Project]-[Rebuild All] from the EWARM menu.



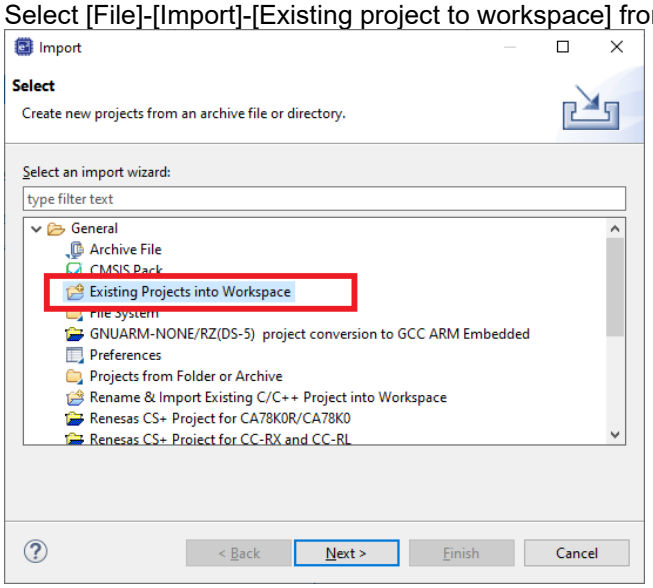
5. After the build is completed, the extension bin file is generated.
   It is assumed that the user program for Cortex®-R52 to be built will run on BTCM, the user program for Cortex®-A55 to be built will run on SYSRAM. Edit the *.icf file under the project directory to change the memory allocation.

   The program size (binary data size) must be a multiple of 512 bytes and less than the maximum size shown in Table 2.1. If the binary data size after the user program build is not a multiple of 512 bytes, add dummy data after the binary data to adjust it to a multiple of 512 bytes. For information on how to adjust the binary data to 512-byte units by adding project settings, refer to 3.3.7.

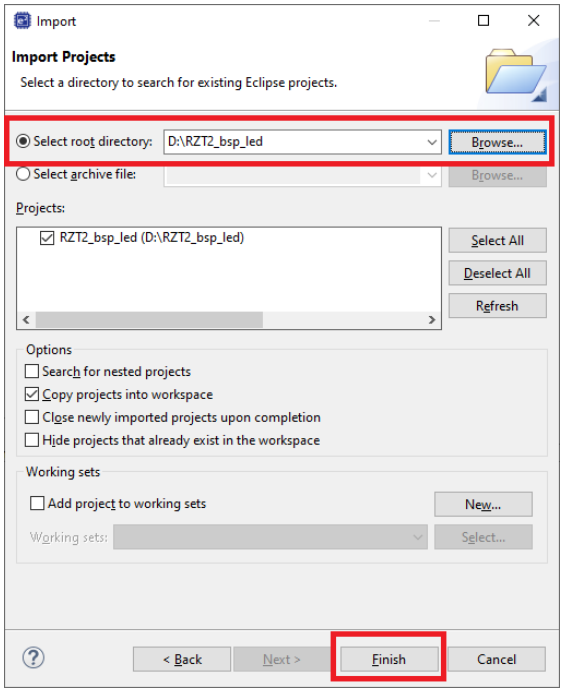
e² studio:
The following settings are also included in the user program project (RZ*_bsp_led.zip).

1. Start e² studio.

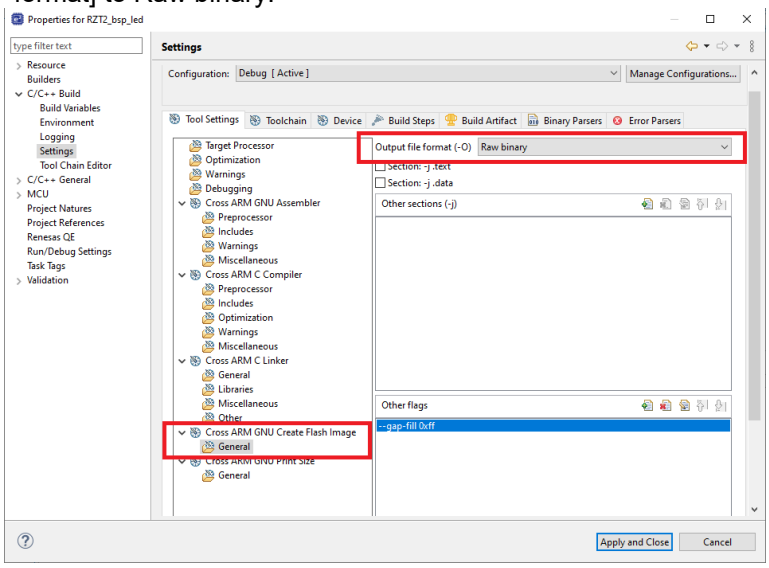
2. Select [File]-[Import]-[Existing project to workspace] from the e² studio menu.



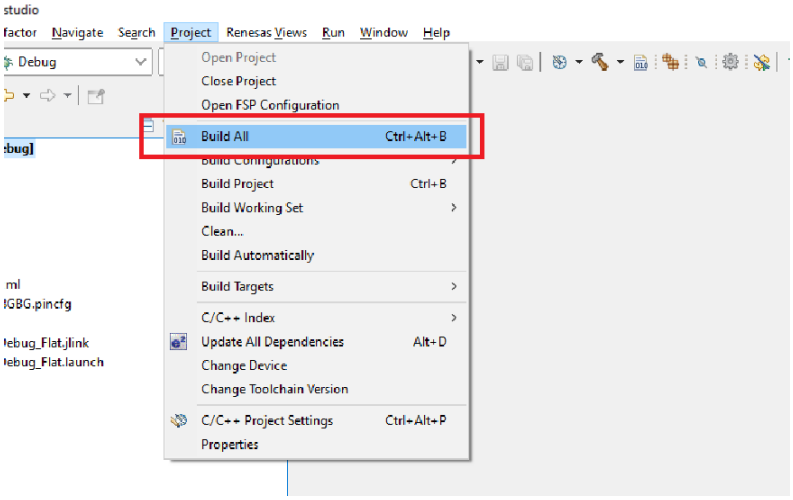
3. Select [Select Root Directory], click the [Browse] button, and select the folder of the program to be written.



4. Select [Project]-[Properties] from the e² studio menu, select [C/C ++ Build]-[Settigns] on the left side, select [Cross ARM GNU Create Flash Image]-[General] on the Tool Settings tab, and set the [Output file format] to Raw binary.



5. Select [Project]-[Build All] from the e² studio menu.



6. After the build is completed, the extension bin file is generated.
   It is assumed that the user program for Cortex®-R52 to be built will run on BTCM, the user program for Cortex®-A55 to be built will run on SYSRAM. Edit the *.ld file under the project directory to change the memory allocation.

   The program size (binary data size) must be a multiple of 512 bytes and less than the maximum size shown in Table 2.1. If the binary data size after the user program build is not a multiple of 512 bytes, add dummy data after the binary data to adjust it to a multiple of 512 bytes. For information on how to adjust the binary data to 512-byte units by adding project settings, refer to 3.3.7.

2.3 Program the user program to the external flash for RZ/T2M

The following shows how to program a user program to the external Flash on the board using this sample program package. The target boards are the Renesas Starter Kit+ for RZ/T2M (hereinafter referred to as RSK+).

This instruction uses a pre-built device setup sample program (RZT2M_RSK_DeviceSetup.out.srec) and two tools (device_setup.py, parameter_generator.py) to program a user program (RZT2M_bsp_led.bin) to flash.

If you wish to use the device setup sample program with modifications, please refer to section to build the sample program project.

Please note that if the version of the device setup sample program and the device setup tool (device_setup.py) used are different, the setup process may fail.

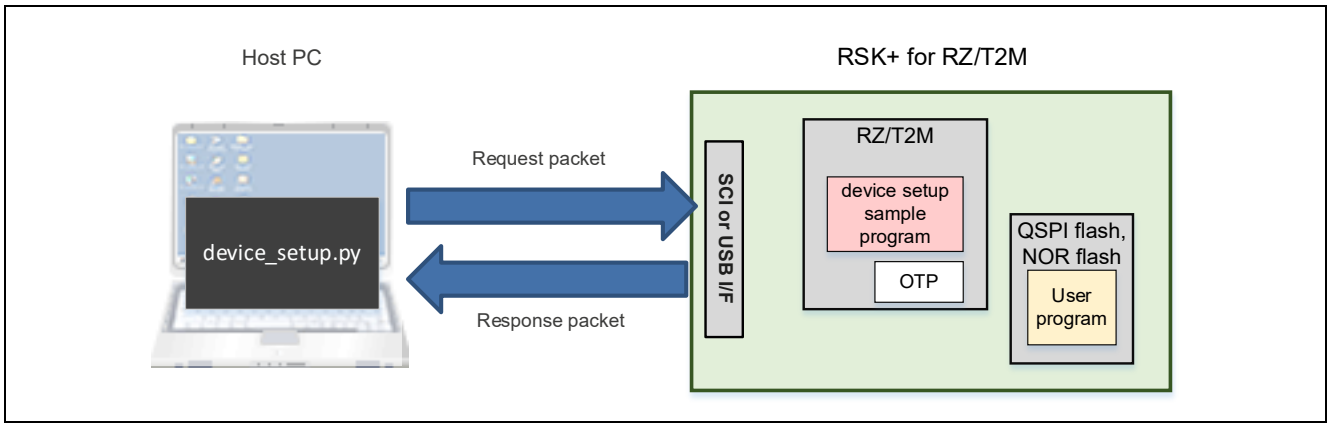


Figure 2.1 System Structure of Device Setup Sample Program

2.3.1 Generate parameters for the loader

Use parameter_generator.py to generate Parameters for the loader needed when starting user program (RZT2M_bsp_led.bin).

First, copy the user program file (*.bin) you created to the same folder as parameter_generator.py. Then start a command prompt and use parameter_generator.py to generate a file (*.bin) to program to the external flash.

The following shows an example of tool execution when xSPI0 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x60000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZT2M_bsp_led_xspi0.bin:

```
> python parameter_generator.py loader --mpu rzt2m --mode xspi0 --src_addr
60000050 --dest_addr 00102000 -i RZT2M_bsp_led.bin -o
RZT2M_bsp_led_xspi0.bin --concat_loader
```

If the --concat_loader option is specified, RZT2M_bsp_led_xspi0.bin contains parameter information plus the program itself.

If the --concat_loader option is omitted, RZT2M_bsp_led_xspi0.bin will contain only the parameter information.

The following shows an example of tool execution when external bus address space flash is specified:
External flash address where the program is stored (--src_addr): 0x70000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZT2M_bsp_led_bus.bin:

```
> python parameter_generator.py loader --mpu rzt2m --mode bus --src_addr
70000050 --dest_addr 00102000 -i RZT2M_bsp_led.bin -o RZT2M_bsp_led_bus.bin
--concat_loader
```

2.3.2 Start the device setup program

Boot the device in SCI boot mode or USB boot mode and load the device setup program (RZT2M_RSK_DeviceSetup.out.srec) into the RAM in the device.

If the program is successfully loaded, the device setup program will start.

In SCI boot mode:

a-1) Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/T2M:

SW	Setting	Description
SW4.1	OFF	SCI (UART) boot mode.
SW4.2	ON	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait

a-2) Load the device setup program (RZT2M_RSK_DeviceSetup.out.srec) into the device using the device setup tool (device_setup.py).

The following command loads RZT2M_RSK_DeviceSetup.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode sci -i
RZT2M_RSK_DeviceSetup.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

In USB boot mode:

b-1) Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/T2M:

SW	Setting	Description
SW4.1	ON	USB boot mode.
SW4.2	OFF	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait

b-2) Load the device setup program (RZT2M_RSK_DeviceSetup.out.srec) into the device using the device setup tool (device_setup.py).

The following command loads RZT2M_RSK_DeviceSetup.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode usb -i
RZT2M_RSK_DeviceSetup.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

2.3.3 Program to Flash

Program the parameter for the loader and user program into the external flash on the RSK+ using the device setup tool (device_setup.py).

If you have started the device setup program in USB boot mode, the COM port of the RSK+ may change from the boot time; refer to step 3 in 2.1.2 to check the COM port again.

The following is an example of tool execution for writing data to the following addresses.

Parameter for the loader + user program (RZT2M_bsp_led_xspi0.bin): 0x60000000-

```
> python device_setup.py writeflash --port COM9 --addr 60000000 -i
RZT2M_bsp_led_xspi0.bin
writeflash : Setup success.
```

Parameter for the loader + user program (RZT2M_bsp_led_bus.bin): 0x70000000-

```
> python device_setup.py writeflash --port COM9 --addr 70000000 -i
RZT2M_bsp_led_bus.bin
writeflash : Setup success.
```

2.3.4 Start the user program

When the device is booted in xSPI0 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the QSPI Flash is extracted to RAM and booted.

Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/T2M:

SW	Setting	Description
SW4.1	ON	xSPI0 boot mode (x1 boot Serial flash)
SW4.2	ON	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait

When the RZ/T2M is booted in 16bit bus boot mode, the device's boot function refers to the parameters for the loader and the user program written to the NOR Flash is extracted to RAM and booted.

Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/T2M:

SW	Setting	Description
SW4.1	ON	16bit bus boot mode
SW4.2	OFF	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait

2.4 Program the user program to the external flash for RZ/T2L

The following shows how to program a user program to the external Flash on the board using this sample program package. The target boards are the Renesas Starter Kit+ for RZ/T2L (hereinafter referred to as RSK+).

This instruction uses a pre-built device setup sample program (RZT2L_RSK_DeviceSetup_*.out.srec) and two tools (device_setup.py, parameter_generator.py) to program a user program (RZT2L_bsp_led.bin) to flash.

If you wish to use the device setup sample program with modifications, please refer to section 3.3.5 and 3.3.6 to build the sample program project.

Please note that if the version of the device setup sample program and the device setup tool (device_setup.py) used are different, the setup process may fail.

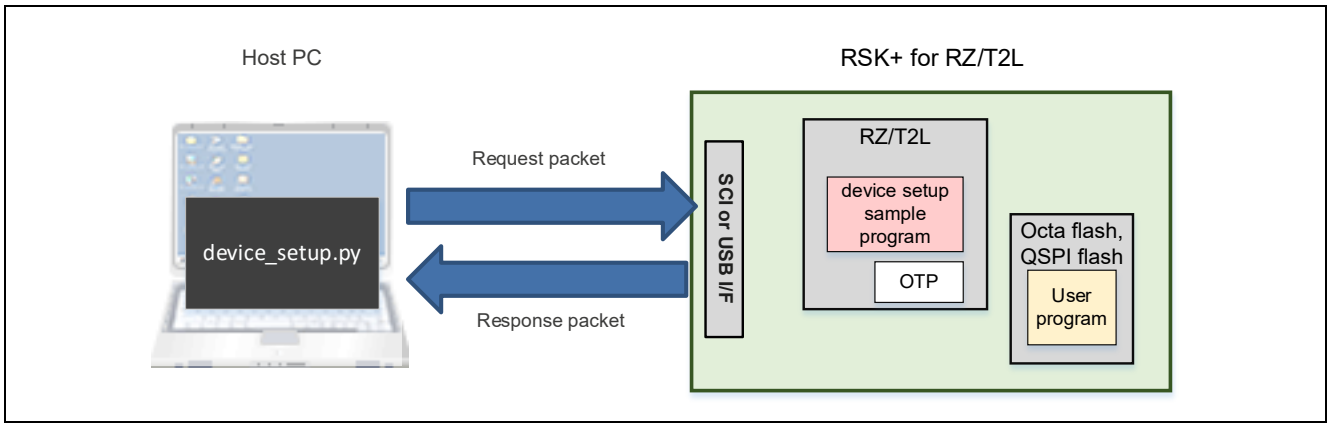


Figure 2.2 System Structure of Device Setup Sample Program

2.4.1 Generate parameters for the loader

Use parameter_generator.py to generate Parameters for the loader needed when starting user program (RZT2L_bsp_led.bin).

First, copy the user program file (*.bin) you created to the same folder as parameter_generator.py. Then start a command prompt and use parameter_generator.py to generate a file (*.bin) to program to the external flash.

The following shows an example of tool execution when xSPI0 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x60000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZT2L_bsp_led_xspi0.bin:

```
> python parameter_generator.py loader --mpu rzt2l --mode xspi0 --src_addr
60000050 --dest_addr 00102000 -i RZT2L_bsp_led.bin -o
RZT2L_bsp_led_xspi0.bin --concat_loader
```

If the --concat_loader option is specified, RZT2L_bsp_led_xspi0.bin contains parameter information plus the program itself.

If the --concat_loader option is omitted, RZT2L_bsp_led_xspi0.bin will contain only the parameter information.

The following shows an example of tool execution when xSPI1 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x68000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZT2L_bsp_led_xspi1.bin:

```
> python parameter_generator.py loader --mpu rzt2l --mode xspi1 --src_addr
68000050 --dest_addr 00102000 -i RZT2L_bsp_led.bin -o
RZT2L_bsp_led_xspi1.bin --concat_loader
```

2.4.2 Start the device setup program

Boot the device in SCI boot mode or USB boot mode and load the device setup program (RZT2L_RSK_DeviceSetup_*.out.srec) into the RAM in the device.

For RZ/T2L, you need to use a device setup program that supports the communication you want to use. The device setup program (RZT2L_RSK_DeviceSetup_usb.out.srec) dedicated to USB communication can use OSPI flash. When using SCI communication, use the device setup program (RZT2L_RSK_DeviceSetup_sci_without_ospi.out.srec) with OSPI flash disabled.

If the program is successfully loaded, the device setup program will start.

In SCI boot mode:

a-1) Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/T2L:

SW	Setting	Description
SW4.1	OFF	SCI (UART) boot mode.
SW4.2	ON	
SW4.3	OFF	
SW4.4	OFF	ATCM wait cycle = 1 wait.
SW4.5	ON	JTAG Authentication by Hash is disabled.

a-2) Load the device setup program (RZT2L_RSK_DeviceSetup_*.out.srec) into the device using the device setup tool (device_setup.py).

The following command loads RZT2L_RSK_DeviceSetup_sci_without_ospi.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode sci -i
RZT2L_RSK_DeviceSetup_sci_without_ospi.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

In USB boot mode:

b-1) Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/T2L:

SW	Setting	Description
SW4.1	ON	USB boot mode.
SW4.2	OFF	
SW4.3	OFF	
SW4.4	OFF	ATCM wait cycle = 1 wait.
SW4.5	ON	JTAG Authentication by Hash is disabled.

b-2) Load the device setup program (RZT2L_RSK_DeviceSetup_*.out.srec) into the device using the device setup tool (device_setup.py).

The following command loads RZT2L_RSK_DeviceSetup_usb.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode usb -i
RZT2L_RSK_DeviceSetup_usb.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

2.4.3 Program to Flash

Program the parameter for the loader and user program into the external flash on the RSK+ using the device setup tool (device_setup.py).

If you have started the device setup program in USB boot mode, the COM port of the RSK+ may change from the boot time; refer to step 3 in 2.1.2 to check the COM port again.

The following is an example of tool execution for writing data to the following addresses.

Parameter for the loader + user program (RZT2L_bsp_led_xspi0.bin): 0x60000000-

```
> python device_setup.py writeflash --port COM9 --addr 60000000 -i
RZT2L_bsp_led_xspi0.bin
writeflash : Setup success.
```

Parameter for the loader + user program (RZT2L_bsp_led_xspi1.bin): 0x68000000-

```
> python device_setup.py writeflash --port COM9 --addr 68000000 -i
RZT2L_bsp_led_xspi1.bin
writeflash : Setup success.
```

2.4.4 Start the user program

When the device is booted in xSPI0 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the OSPI Flash is extracted to RAM and booted.

Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/T2L:

SW	Setting	Description
SW4.1	ON	xSPI0 boot mode (x1 boot Serial flash)
SW4.2	ON	
SW4.3	ON	
SW4.4	OFF	ATCM wait cycle = 1 wait.
SW4.5	ON	JTAG Authentication by Hash is disabled.

When the RZ/T2L is booted in xSPI1 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the QSPI Flash is extracted to RAM and booted.

Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/T2L:

SW	Setting	Description
SW4.1	ON	xSPI1 boot mode (x1 boot Serial flash)
SW4.2	ON	
SW4.3	OFF	
SW4.4	OFF	ATCM wait cycle = 1 wait.
SW4.5	ON	JTAG Authentication by Hash is disabled.

2.5 Program the user program to the external flash for RZ/N2L

The following shows how to program a user program to the external Flash on the board using this sample program package. The target boards are the Renesas Starter Kit+ for RZ/N2L (hereinafter referred to as RSK+).

This instruction uses a pre-built device setup sample program (RZN2L_RSK_DeviceSetup_*.out.srec) and two tools (device_setup.py, parameter_generator.py) to program a user program (RZN2L_bsp_led.bin) to flash.

If you wish to use the device setup sample program with modifications, please refer to section 3.3.4 and 3.3.6 to build the sample program project.

Please note that if the version of the device setup sample program and the device setup tool (device_setup.py) used are different, the setup process may fail.

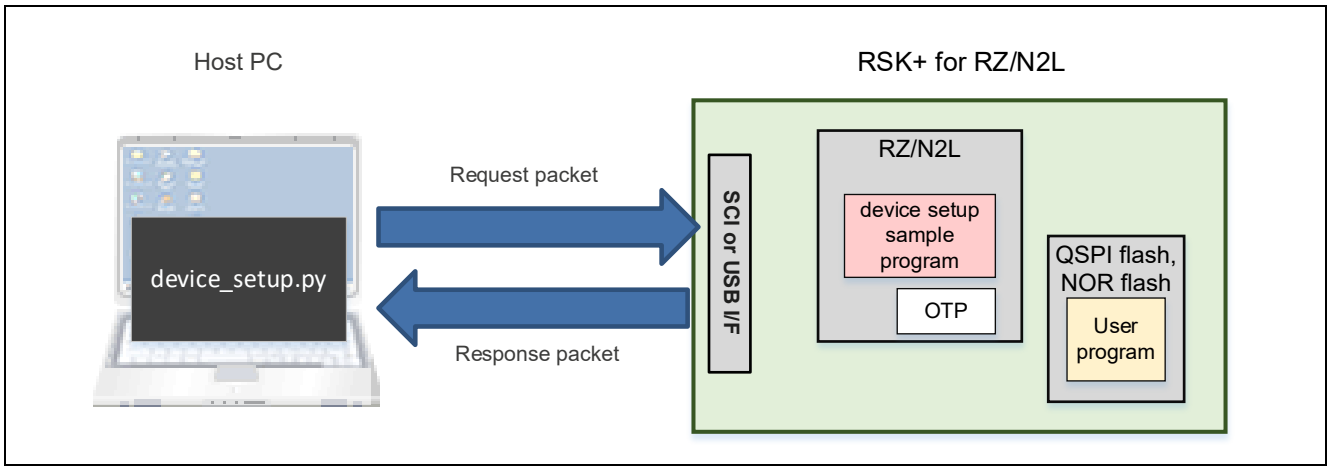


Figure 2.3 System Structure of Device Setup Sample Program

2.5.1 Generate parameters for the loader

Use parameter_generator.py to generate Parameters for the loader needed when starting user program (RZN2L_bsp_led.bin).

First, copy the user program file (*.bin) you created to the same folder as parameter_generator.py. Then start a command prompt and use parameter_generator.py to generate a file (*.bin) to program to the external flash.

The following shows an example of tool execution when xSPI0 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x60000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZN2L_bsp_led_xspi0.bin:

```
> python parameter_generator.py loader --mpu rzn2l --mode xspi0 --src_addr
60000050 --dest_addr 00102000 -i RZN2L_bsp_led.bin -o
RZN2L_bsp_led_xspi0.bin --concat_loader
```

If the --concat_loader option is specified, RZN2L_bsp_led_xspi0.bin contains parameter information plus the program itself.

If the --concat_loader option is omitted, RZN2L_bsp_led_xspi0.bin will contain only the parameter information.

The following shows an example of tool execution when external bus address space flash is specified:
External flash address where the program is stored (--src_addr): 0x70000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZN2L_bsp_led_bus.bin:

```
> python parameter_generator.py loader --mpu rzn2l --mode bus --src_addr
70000050 --dest_addr 00102000 -i RZN2L_bsp_led.bin -o RZN2L_bsp_led_bus.bin
--concat_loader
```

2.5.2 Start the device setup program

Boot the device in SCI boot mode or USB boot mode and load the device setup program (RZN2L_RSK_DeviceSetup_*.out.srec) into the RAM in the device.

For RZ/N2L, it is necessary to use the device setup program corresponding to the external flash used. The procedure assumes that the device setup program (RZN2L_RSK_DeviceSetup_qspi.out.srec) for QSPI flash is used. When using NOR flash, use the device setup program for NOR flash (RZN2L_RSK_DeviceSetup_nor.out.srec).

If the program is successfully loaded, the device setup program will start.

In SCI boot mode:

a-1) Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/N2L:

SW	Setting	Description
SW4.1	OFF	SCI (UART) boot mode.
SW4.2	ON	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.

a-2) Load the device setup program (RZN2L_RSK_DeviceSetup_*.out.srec) into the device using the device setup tool (device_setup.py).

The following command loads RZN2L_RSK_DeviceSetup_qspi.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode sci -i
RZN2L_RSK_DeviceSetup_qspi.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

In USB boot mode:

b-1) Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/N2L:

SW	Setting	Description
SW4.1	ON	USB boot mode.
SW4.2	OFF	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.

b-2) Load the device setup program (RZN2L_RSK_DeviceSetup_*.out.srec) into the device using the device setup tool (device_setup.py).

The following command loads RZN2L_RSK_DeviceSetup_qspi.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode usb -i
RZN2L_RSK_DeviceSetup_qspi.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

2.5.3 Program to Flash

Program the parameter for the loader and user program into the external flash on the RSK+ using the device setup tool (device_setup.py).

If you have started the device setup program in USB boot mode, the COM port of the RSK+ may change from the boot time; refer to step 3 in 2.1.2 to check the COM port again.

The following is an example of tool execution for writing data to the following addresses.

Parameter for the loader + user program (RZN2L_bsp_led_xspi0.bin): 0x60000000-

```
> python device_setup.py writeflash --port COM9 --addr 60000000 -i
RZN2L_bsp_led_xspi0.bin
writeflash : Setup success.
```

Parameter for the loader + user program (RZN2L_bsp_led_bus.bin): 0x70000000-

```
> python device_setup.py writeflash --port COM9 --addr 70000000 -i
RZN2L_bsp_led_bus.bin
writeflash : Setup success.
```

2.5.4 Start the user program

When the device is booted in xSPI0 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the QSPI Flash is extracted to RAM and booted.

Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/N2L:

SW	Setting	Description
SW4.1	ON	xSPI0 boot mode (x1 boot Serial flash)
SW4.2	ON	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.

When the RZ/N2L is booted in 16bit bus boot mode, the device's boot function refers to the parameters for the loader and the user program written to the NOR Flash is extracted to RAM and booted.

Set SW4[1:5] on the RSK+ board to the following and press S3 RESET.

RZ/N2L:

SW	Setting	Description
SW4.1	ON	16bit bus boot mode
SW4.2	OFF	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.

2.6 Program the user program to the external flash for RZ/T2H

The following shows how to program a user program to the external Flash on the board using this sample program package. The target boards are the RZ/T2H Evaluation Board (hereinafter referred to as EVB).

This instruction uses a pre-built device setup sample program (RZT2H_EVB_DeviceSetup.out.srec) and two tools (device_setup.py, parameter_generator.py) to program a user program (RZT2H_bsp_led.bin or RZT2H_CA55_bsp_led.bin) to flash.

If you wish to use the device setup sample program with modifications, please refer to section 3.3.6 to build the sample program project.

Please note that if the version of the device setup sample program and the device setup tool (device_setup.py) used are different, the setup process may fail.

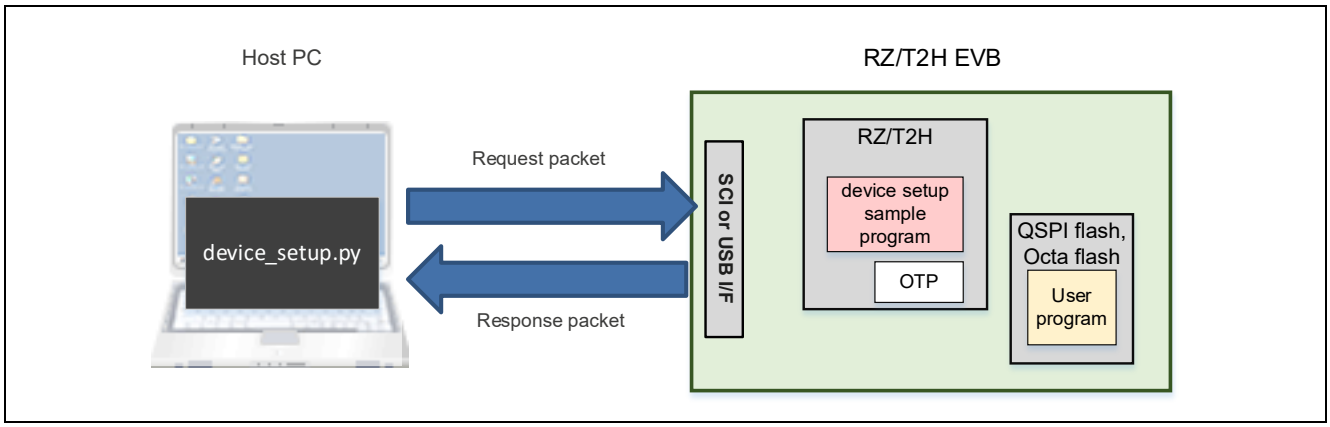


Figure 2.4 System Structure of Device Setup Sample Program

2.6.1 Generate parameters for the loader

Use parameter_generator.py to generate Parameters for the loader needed when starting user program (RZT2H_bsp_led.bin or RZT2H_CA55_bsp_led.bin).

First, copy the user program file (*.bin) you created to the same folder as parameter_generator.py. Then start a command prompt and use parameter_generator.py to generate a file (*.bin) to program to the external flash.

RZ/T2H Cortex®-R52:
The following shows an example of tool execution when xSPI0 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x40000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZT2H_bsp_led_xspi0.bin:

```
> python parameter_generator.py loader --mpu rzt2h_r52 --mode xspi0 --
src_addr 40000050 --dest_addr 00102000 -i RZT2H_bsp_led.bin -o
RZT2H_bsp_led_xspi0.bin --concat_loader
```

If the --concat_loader option is specified, RZT2H_bsp_led_xspi0.bin contains parameter information plus the program itself.

If the --concat_loader option is omitted, RZT2H_bsp_led_xspi0.bin will contain only the parameter information.

The following shows an example of tool execution when xSPI1 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x50000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZT2H_bsp_led_xspi1.bin:

```
> python parameter_generator.py loader --mpu rzt2h_r52 --mode xspi1 --
src_addr 50000050 --dest_addr 00102000 -i RZT2H_bsp_led.bin -o
RZT2H_bsp_led_xspi1.bin --concat_loader
```

RZ/T2H Cortex®-A55:
The following shows an example of tool execution when xSPI0 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x40000050
RAM address where the program is loaded (--dest_addr): 0x10000000

The following command will generate RZT2H_CA55_bsp_led_xspi0.bin:

```
> python parameter_generator.py loader --mpu rzt2h_a55 --mode xspi0 --
src_addr 40000050 --dest_addr 10000000 -i RZT2H_CA55_bsp_led.bin -o
RZT2H_CA55_bsp_led_xspi0.bin --concat_loader
```

If the --concat_loader option is specified, RZT2H_CA55_bsp_led_xspi0.bin contains parameter information plus the program itself.
If the --concat_loader option is omitted, RZT2H_CA55_bsp_led_xspi0.bin will contain only the parameter information.

The following shows an example of tool execution when xSPI1 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x50000050
RAM address where the program is loaded (--dest_addr): 0x10000000

The following command will generate RZT2H_CA55_bsp_led_xspi1.bin:

```
> python parameter_generator.py loader --mpu rzt2h_a55 --mode xspi1 --
src_addr 50000050 --dest_addr 10000000 -i RZT2H_CA55_bsp_led.bin -o
RZT2H_CA55_bsp_led_xspi1.bin --concat_loader
```

2.6.2 Start the device setup program

Boot the device in SCI boot mode or USB boot mode and load the device setup program (RZT2H_EVB_DeviceSetup.out.srec) into the RAM in the device.

If the program is successfully loaded, the device setup program will start.

In SCI boot mode:

a-1) Set SW14[1:7] on the EVB to the following and press SW13 RESET.
RZ/T2H:

SW	Setting	Description
SW14.1	OFF	SCI (UART) boot mode.
SW14.2	ON	
SW14.3	OFF	
SW14.4	OFF	CPU0 ATCM 1 wait
SW14.5	OFF	CPU1 ATCM 1 wait
SW14.6	OFF	Supply voltage of boot peripheral is 3.3 V
SW14.7	ON	JTAG Authentication by Hash is disabled.

a-2) Load the device setup program (RZT2H_EVB_DeviceSetup.out.srec) into the device using the device setup tool (device_setup.py).
The following command loads RZT2H_EVB_DeviceSetup.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode sci -i
RZT2H_EVB_DeviceSetup.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to RAM  ---------------
Send program data. (S3)
-- Start Boot Program on RAM ---------
```

In USB boot mode:

b-1) Set SW14[1:7] on the EVB to the following and press SW13 RESET.

RZ/T2H:

SW	Setting	Description
SW14.1	ON	USB boot mode.
SW14.2	OFF	
SW14.3	OFF	
SW14.4	OFF	CPU0 ATCM 1 wait
SW14.5	OFF	CPU1 ATCM 1 wait
SW14.6	OFF	Supply voltage of boot peripheral is 3.3 V
SW14.7	ON	JTAG Authentication by Hash is disabled.

b-2) Load the device setup program (RZT2H_EVB_DeviceSetup.out.srec) into the device using the device setup tool (device_setup.py).

The following command loads RZT2H_EVB_DeviceSetup.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode usb -i
RZT2H_EVB_DeviceSetup.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to RAM  ---------------
Send program data. (S3)
-- Start Boot Program on RAM ---------
```

2.6.3 Program to Flash

Program the parameter for the loader and user program into the external flash on the EVB using the device setup tool (device_setup.py).

If you have started the device setup program in USB boot mode, the COM port of the EVB may change from the boot time; refer to step 3 in 2.1.2 to check the COM port again.

The following is an example of tool execution for writing data to the following addresses.

RZ/T2H Cortex®-R52:
Parameter for the loader + user program (RZT2H_bsp_led_xspi0.bin): 0x40000000-

```
> python device_setup.py writeflash --port COM9 --addr 40000000 -i
RZT2H_bsp_led_xspi0.bin
writeflash : Setup success.
```

Parameter for the loader + user program (RZT2H_bsp_led_xspi1.bin): 0x50000000-

```
> python device_setup.py writeflash --port COM9 --addr 50000000 -i
RZT2H_bsp_led_xspi1.bin
writeflash : Setup success.
```

RZ/T2H Cortex®-A55:
Parameter for the loader + user program (RZT2H_CA55_bsp_led_xspi0.bin): 0x40000000-

```
> python device_setup.py writeflash --port COM9 --addr 40000000 -i RZT2H
_CA55_bsp_led_xspi0.bin
writeflash : Setup success.
```

Parameter for the loader + user program (RZT2H_CA55_bsp_led_xspi1.bin): 0x50000000-

```
> python device_setup.py writeflash --port COM9 --addr 50000000 -i RZT2H
_CA55_bsp_led_xspi1.bin
writeflash : Setup success.
```

2.6.4 Start the user program

When the RZ/T2H is booted in xSPI0 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the OSPI Flash is extracted to RAM and booted.

Set SW14[1:7] on the Evaluation Board to the following and press SW13 RESET.

RZ/T2H:

SW	Setting	Description
SW14.1	ON	xSPI0 boot mode (x1 boot Serial flash)
SW14.2	ON	
SW14.3	ON	
SW14.4	OFF	CPU0 ATCM 1 wait
SW14.5	OFF	CPU1 ATCM 1 wait
SW14.6	OFF	Supply voltage of boot peripheral is 3.3 V
SW14.7	ON	JTAG Authentication by Hash is disabled.

When the RZ/T2H is booted in xSPI1 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the QSPI Flash is extracted to RAM and booted.

Set SW14[1:7] on the Evaluation Board to the following and press SW13 RESET.

RZ/T2H:

SW	Setting	Description
SW14.1	ON	xSPI1 boot mode (x1 boot Serial flash)
SW14.2	OFF	
SW14.3	ON	
SW14.4	OFF	CPU0 ATCM 1 wait
SW14.5	OFF	CPU1 ATCM 1 wait
SW14.6	OFF	Supply voltage of boot peripheral is 3.3 V
SW14.7	ON	JTAG Authentication by Hash is disabled.

2.7 Program the user program to the external flash for RZ/N2H

The following shows how to program a user program to the external Flash on the board using this sample program package. The target boards are the RZ/N2H Evaluation Board (hereinafter referred to as EVB).

This instruction uses a pre-built device setup sample program (RZN2H_EVB_DeviceSetup.out.srec) and two tools (device_setup.py, parameter_generator.py) to program a user program (RZN2H_bsp_led.bin or RZN2H_CA55_bsp_led.bin) to flash.

If you wish to use the device setup sample program with modifications, please refer to section 3.3.6 to build the sample program project.

Please note that if the version of the device setup sample program and the device setup tool (device_setup.py) used are different, the setup process may fail.

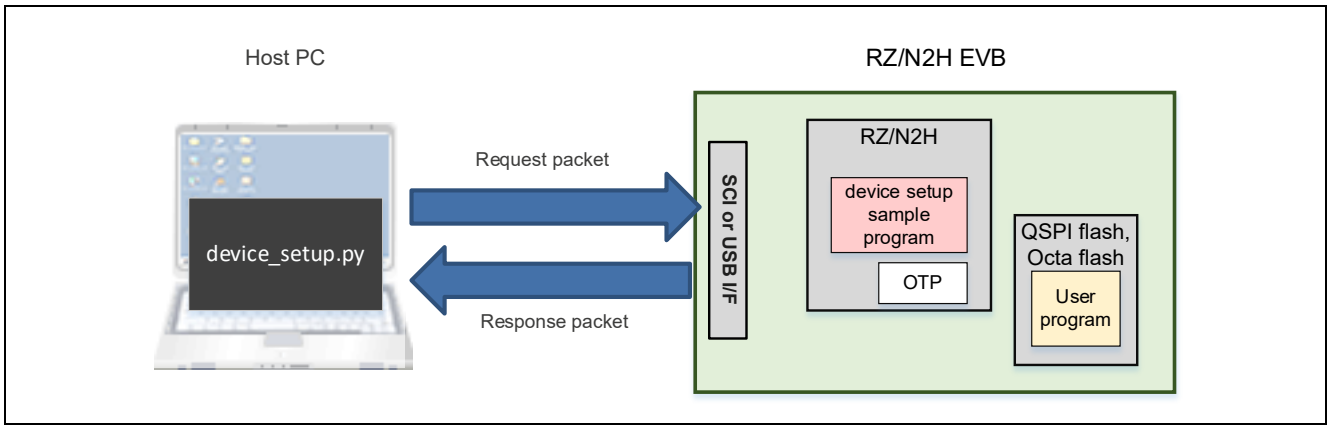


Figure 2.5 System Structure of Device Setup Sample Program

2.7.1 Generate parameters for the loader

Use parameter_generator.py to generate Parameters for the loader needed when starting user program (RZN2H_bsp_led.bin or RZN2H_CA55_bsp_led.bin).

First, copy the user program file (*.bin) you created to the same folder as parameter_generator.py. Then start a command prompt and use parameter_generator.py to generate a file (*.bin) to program to the external flash.

RZ/N2H Cortex®-R52:
The following shows an example of tool execution when xSPI0 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x40000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZN2H_bsp_led_xspi0.bin:

```
> python parameter_generator.py loader --mpu rzn2h_r52 --mode xspi0 --
src_addr 40000050 --dest_addr 00102000 -i RZN2H_bsp_led.bin -o
RZN2H_bsp_led_xspi0.bin --concat_loader
```

If the --concat_loader option is specified, RZN2H_bsp_led_xspi0.bin contains parameter information plus the program itself.

If the --concat_loader option is omitted, RZN2H_bsp_led_xspi0.bin will contain only the parameter information.

The following shows an example of tool execution when xSPI1 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x50000050
RAM address where the program is loaded (--dest_addr): 0x00102000

The following command will generate RZN2H_bsp_led_xspi1.bin:

```
> python parameter_generator.py loader --mpu rzn2h_r52 --mode xspi1 --
src_addr 50000050 --dest_addr 00102000 -i RZN2H_bsp_led.bin -o
RZN2H_bsp_led_xspi1.bin --concat_loader
```

RZ/N2H Cortex®-A55:
The following shows an example of tool execution when xSPI0 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x40000050
RAM address where the program is loaded (--dest_addr): 0x10000000

The following command will generate RZN2H_CA55_bsp_led_xspi0.bin:

```
> python parameter_generator.py loader --mpu rzn2h_a55 --mode xspi0 --
src_addr 40000050 --dest_addr 10000000 -i RZN2H_CA55_bsp_led.bin -o
RZN2H_CA55_bsp_led_xspi0.bin --concat_loader
```

If the --concat_loader option is specified, RZN2H_CA55_bsp_led_xspi0.bin contains parameter information plus the program itself.
If the --concat_loader option is omitted, RZN2H_CA55_bsp_led_xspi0.bin will contain only the parameter information.

The following shows an example of tool execution when xSPI1 address space flash is specified:
External flash address where the program is stored (--src_addr): 0x50000050
RAM address where the program is loaded (--dest_addr): 0x10000000

The following command will generate RZN2H_CA55_bsp_led_xspi1.bin:

```
> python parameter_generator.py loader --mpu rzn2h_a55 --mode xspi1 --
src_addr 50000050 --dest_addr 10000000 -i RZN2H_CA55_bsp_led.bin -o
RZN2H_CA55_bsp_led_xspi1.bin --concat_loader
```

2.7.2 Start the device setup program

Boot the device in SCI boot mode or USB boot mode and load the device setup program (RZN2H_EVB_DeviceSetup.out.srec) into the RAM in the device.

If the program is successfully loaded, the device setup program will start.

In SCI boot mode:

a-1) Set DSW3[1:7] on the EVB to the following and press SW5 RESET.
RZ/N2H:

SW	Setting	Description
DSW3.1	OFF	SCI (UART) boot mode.
DSW3.2	ON	
DSW3.3	OFF	
DSW3.4	OFF	CPU0 ATCM 1 wait
DSW3.5	OFF	CPU1 ATCM 1 wait
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG Authentication by Hash is disabled.

a-2) Load the device setup program (RZN2H_EVB_DeviceSetup.out.srec) into the device using the device setup tool (device_setup.py).
The following command loads RZN2H_EVB_DeviceSetup.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode sci -i
RZN2H_EVB_DeviceSetup.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to RAM  ---------------
Send program data. (S3)
-- Start Boot Program on RAM ---------
```

In USB boot mode:

b-1) Set DSW3[1:7] on the EVB to the following and press SW5 RESET.

RZ/N2H:

SW	Setting	Description
DSW3.1	ON	USB boot mode.
DSW3.2	OFF	
DSW3.3	OFF	
DSW3.4	OFF	CPU0 ATCM 1 wait
DSW3.5	OFF	CPU1 ATCM 1 wait
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG Authentication by Hash is disabled.

b-2) Load the device setup program (RZN2H_EVB_DeviceSetup.out.srec) into the device using the device setup tool (device_setup.py).

The following command loads RZN2H_EVB_DeviceSetup.out.srec:

```
> python device_setup.py start --port COM9 --boot_mode usb -i
RZN2H_EVB_DeviceSetup.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to RAM  ---------------
Send program data. (S3)
-- Start Boot Program on RAM ---------
```

2.7.3 Program to Flash

Program the parameter for the loader and user program into the external flash on the EVB using the device setup tool (device_setup.py).

If you have started the device setup program in USB boot mode, the COM port of the EVB may change from the boot time; refer to step 3 in 2.1.2 to check the COM port again.

The following is an example of tool execution for writing data to the following addresses.

RZ/N2H Cortex®-R52:

Parameter for the loader + user program (RZN2H_bsp_led_xspi0.bin): 0x40000000-

```
> python device_setup.py writeflash --port COM9 --addr 40000000 -i
RZN2H_bsp_led_xspi0.bin
writeflash : Setup success.
```

Parameter for the loader + user program (RZN2H_bsp_led_xspi1.bin): 0x50000000-

```
> python device_setup.py writeflash --port COM9 --addr 50000000 -i
RZN2H_bsp_led_xspi1.bin
writeflash : Setup success.
```

RZ/N2H Cortex®-A55:

Parameter for the loader + user program (RZN2H_CA55_bsp_led_xspi0.bin): 0x40000000-

```
> python device_setup.py writeflash --port COM9 --addr 40000000 -i
RZN2H_CA55_bsp_led_xspi0.bin
writeflash : Setup success.
```

Parameter for the loader + user program (RZN2H_CA55_bsp_led_xspi1.bin): 0x50000000-

```
> python device_setup.py writeflash --port COM9 --addr 50000000 -i
RZN2H_CA55_bsp_led_xspi1.bin
writeflash : Setup success.
```

2.7.4 Start the user program

When the RZ/N2H is booted in xSPI0 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the OSPI Flash is extracted to RAM and booted.

Set DSW3[1:7] on the Evaluation Board to the following and press SW5 RESET.

RZ/N2H:

SW	Setting	Description
DSW3.1	ON	xSPI0 boot mode (x1 boot Serial flash)
DSW3.2	ON	
DSW3.3	ON	
DSW3.4	OFF	CPU0 ATCM 1 wait
DSW3.5	OFF	CPU1 ATCM 1 wait
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG Authentication by Hash is disabled.

When the RZ/N2H is booted in xSPI1 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the QSPI Flash is extracted to RAM and booted.

Set DSW3[1:7] on the Evaluation Board to the following and press SW5 RESET.

RZ/N2H:

SW	Setting	Description
DSW3.1	ON	xSPI1 boot mode (x1 boot Serial flash)
DSW3.2	OFF	
DSW3.3	ON	
DSW3.4	OFF	CPU0 ATCM 1 wait
DSW3.5	OFF	CPU1 ATCM 1 wait
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG Authentication by Hash is disabled.

2.8 Program to the OTP memory

2.8.1 Start the device setup program

Refer to section 2.3.2 to boot the device in SCI boot mode or USB boot mode and load the device setup program (RZT2M_RSK_DeviceSetup.out.srec, RZT2L_RSK_DeviceSetup_*.out.srec, RZT2H_EVB_DeviceSetup.out.srec, RZN2L_RSK_DeviceSetup_*.out.srec or RZN2H_EVB_DeviceSetup.out.srec) into the RAM of the device.

If the program is successfully loaded, the device setup program will start.

2.8.2 Program to OTP

Program to the OTP in the device using the device setup tool (device_setup.py).

An example of setting a password for JTAG Authentication level 1 is shown below.

In id_plain.bin, specify 128-bit binary data to be set as the plaintext password.

```
> python device_setup.py setjauthid --port COM9 --mode authlv1 -i
id_plain.bin
> python device_setup.py setjauth --port COM9 --mode authlv1
```

An example of disabling SCI/USB boot is shown below. These commands are available on the RZ/T2M, RZ/T2L, and RZ/N2L.

Note that due to device specifications, SCI boot and USB boot cannot be disabled separately. setsciboot and setusbboot are two commands available, but both work the same.

```
> python device_setup.py setsciboot --port COM9 -disable
> python device_setup.py setusbboot --port COM9 -disable
```

An example of setting a password for SCI/USB Boot Authentication is shown below. These commands are available on the RZ/T2H and RZ/N2H.
Note that due to device specifications, SCI boot and USB boot cannot be set separately.

In id_plain.bin, specify 128-bit binary data to be set as the plaintext password.

```
> python device_setup.py setsauthid --port COM9 -i id_plain.bin
> python device_setup.py setsauth --port COM9 --mode auth
```

The following is an example of tool execution for reading the Unique ID.

```
> python device_setup.py getuid --port COM9
```

3. Sample Program

This package is provided as a set of sample program projects including source codes and tool body files in the execution format. This sample program projects and tools can be modified for each user environment.

In this section, the specifications of the tools included in the sample program package are described in Section 3.1 and 3.2, the implementation specifications of the device setup program in Section 3.3, and the communication specifications of the device setup program in Section 3.4.

3.1 parameter_generator.py

The tool parameter_generator.py is used to create the parameters for the loader.

Using parameter_generator.py, you can create the parameters for the loader by specifying the user program and its ROM/RAM location address.

The command format of parameter_generator.py is as follows:

```
python parameter_generator.py < command > < options >
```

3.1.1 Commands and Options

Table 3.1 lists the commands, and Table 3.2 and Table 3.3 lists the options corresponding to each command.

Table 3.1 Commands of parameter_generator.py

Commands	Description
loader	Create the parameters for the loader.
userapp	Create the parameters for the user application program. The parameter data created using this command can be used in the loader program included in the firmware update sample program package.

Table 3.2 Options for loader command

Option	Required/Optional	Description
--mpu [rzt2m / rzt2l / rzn2l / rzt2h_r52 / rzt2h_a55 / rzn2h_r52 / rzn2h_a55]	Required	Specify the target MPU for parameter information to be generated. Possible values are shown below. rzt2m: Generate parameter information for RZ/T2M. rzt2l: Generate parameter information for RZ/T2L. rzn2l: Generate parameter information for RZ/N2L. rzt2h_r52: Generate parameter information for RZ/T2H Cortex®-R52. rzt2h_a55: Generate parameter information for RZ/T2H Cortex®-A55. rzn2h_r52: Generate parameter information for RZ/N2H Cortex®-R52. rzn2h_a55: Generate parameter information for RZ/N2H Cortex®-A55. (The upper limit of the loader program size that can be specified is determined by specifying the MPU)
--src_addr <Address>	Required	Specify the flash storage address of the program to be set in the parameter of the loader. Specify the address in 8-digit hexadecimal number. Example: 5000004C The tool checks that the entered value is an 8-digit hexadecimal number but does not check the address range.
--dest_addr <Address>	Required	Specify the RAM address to expand the loader program file. The specified value is set as the destination RAM address included in the parameter information. Specify the address in 8-digit hexadecimal number. Example: 00102000 The tool checks that the entered value is an 8-digit hexadecimal number but does not check the address range.
--mode [xspi0 / xspi1 / bus]	Required	Specify the boot mode to be set in the parameter for the loader. Possible values are shown below. xspi0, xspi1: Generate the parameter for the loader for xSPI boot mode. bus: Generates the parameter for the loader for Bus boot mode.
-i <File>	Required	Specify the program file path for the target loader program for which parameter for the loader is to be created.
-o <File>	Required	Specify the file name of the parameter for the loader to be output.
--concat_loader	Optional	Program data concatenation flag If specified, data from the program file specified by -i is concatenated after the parameter the loader. When concatenating, the program is padded to the required size based on the values of --mode and --src_addr before concatenating the program. The size calculation for padding is as follows xspi0: [--src_addr address] - PROG_BASE_ADDR_XSPI0_RZ* xspi1: [--src_addr address] - PROG_BASE_ADDR_XSPI1_RZ* bus: [--src_addr address] - PROG_BASE_ADDR_BUS_RZ* The following values are used to fill the area when padding. PADDING_VAL
-h	Optional	Specify this option to display help on using this tool.

Table 3.3 Options for userapp command

Option	Required/Optional	Description
--src_addr <Address>	Required	Specify the flash storage address of the program to be set in the parameter of the user application program. Specify the address in 8-digit hexadecimal number. Example: 5000004C The tool checks that the entered value is an 8-digit hexadecimal number but does not check the address range.
--app_start_addr <Address>	Required	Specify the start address of the user application program. Specify the address in 8-digit hexadecimal number. Example: 00102000 The tool checks that the entered value is an 8-digit hexadecimal number but does not check the address range.
-i <File>	Required	Specify the program file path for the target user application program for which parameter for the user application program is to be created.
-o <File>	Required	Specify the file name of the parameter for the user application program to be output.
-h	Optional	Specify this option to display help on using this tool.

An example of specifying options on the command line is shown below.

Create the parameters for the loader:

```
> python parameter_generator.py loader --mpu rzt2m --src_addr 5000004C --
dest_addr 00102000 --mode xspi0 -i loader_program.bin -o loader_param.bin
```

Create the parameters for the user application program:

```
> python parameter_generator.py userapp --src_addr 50100010 --app_start_addr
00002000 -i userapp_program.bin -o userapp_param.bin
```

3.1.2 Input files

This tool does not check the data in the input file. When generating parameters for a loader, the input file is assumed to be the loader program. When generating parameters for a user application program, the input file is assumed to be the user application program to be loaded by the loader program.

The maximum size of the input file for the loader program shall be the following values defined in the tool.

MPU is determined by the value specified in the command option (--mpu).

For RZ/T2M：LOADER_SIZE_MAX_RZT2M_BTCM

For RZ/T2L：LOADER_SIZE_MAX_RZT2L_BTCM

For RZ/N2L：LOADER_SIZE_MAX_RZN2L_BTCM

For RZ/T2H Cortex®-R52 and RZ/N2H Cortex®-R52：LOADER_SIZE_MAX_RZT2H_BTCM

For RZ/T2H Cortex®-A55 and RZ/N2H Cortex®-A55:LOADER_SIZE_MAX_RZT2H_SYSRAM

If the upper limit is exceeded, an error message will be displayed, and processing will terminate.

If the size of the input file is not a multiple of the following values, a warning is displayed.

LOADER_UNIT_SIZE

3.1.3 Output files

Each parameter data is output as a binary file.

3.2 device_setup.py

The tool device_setup.py is used to send device setup commands to the device (RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L or RZ/N2H).

Using device_setup.py, you can send device setup commands to the device via SCI or USB. Afterward, the tool receives the command execute result from the device and outputs it to the console.

The command format of device_setup.py is as follows:

```
python device_setup.py < command > < options >
```

3.2.1 Commands and Options

Table 3.4 lists the commands.

Table 3.4 Commands of device_setup.py

Commands	Description
start	Send the setup program to the device and start it
writeflash	Write to Flash
writeotp	Write to OTP
readotp	Read from OTP
setjauth	Set JTAG authentication
getjauth	Get JTAG authentication settings
setjauthid	Set JTAG authentication ID
setsciboot, setusbboot	Set SCI/USB boot disabled
getsciboot, getusbboot	Get SCI/USB boot settings
getuid	Read unique ID
setsauth	Set SCI/USB boot authentication
getsauth	Get SCI/USB boot authentication settings
setsauthid	Set SCI/USB boot authentication ID

3.2.1.1 Send the setup program to the device and start it (start)

When sending the setup program, specify the setup program to be booted on the device (RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L or RZ/N2H). When setting up a device, first execute this command to start up the setup program, and then execute other setup commands.

The options for the command to start are listed below.

Table 3.5 List of options for start command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
-i <File>	Required	Specify the program file to be SCI/USB booted
--boot_mode [sci / usb]	Required	Selecting the boot mode sci: Transfer program with SCI boot protocol usb: Transfer program with USB boot protocol
--ps <File>	Optional	Specify the password file for SCI/USB boot authentication
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py start --port COM9 --boot_mode sci -i
RZT2M_DeviceSetup.srec
```

3.2.1.2 Write to Flash (writeflash)

Optionally specify the data file to be written and the write destination address. The data file to be specified is assumed to be a program file (loader program or user application program) or a parameter file generated by the parameter_generator.py.

If the file size exceeds DATA_MAX, the data is split and sent multiple times.

The options for the command to write flash are listed below.

Table 3.6 List of options for writeflash command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
--addr <Address>	Required	Specify the Flash address. Specify the address in 8-digit hexadecimal number. Example: 5000004C The tool checks that the entered value is an 8-digit hexadecimal number but does not check the address range.
-i <File>	Required	Specify the file path of the data to be written to flash.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py writeflash --port COM9 --addr 5000004C -i
loader_param.bin
```

3.2.1.3 Write to OTP（writeotp）

When writing to the OTP, specify the OTP address and data. The options for the command to write OTP are listed below.

Table 3.7 List of options for writeotp command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
--addr <Address>	Required	Specify the OTP address. Specify the address in 4-digit hexadecimal number. Example: 004C The tool checks that the entered value is a 4-digit hexadecimal number but does not check the address range.
--data <Data>	Required	Specify 16 to 32-bit write data. Specify the data in 4 to 8-digit hexadecimal number. Example: 00F8 The tool checks that the entered value is a 4 to 8-digit hexadecimal number.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py writeotp --port COM9 --addr 004C --data 00F8
```

3.2.1.4 Read from OTP (readopt)

When reading from the OTP, the OTP address to be read is specified. The options for the command to read OTP are listed below.

Table 3.8 List of options for readotp command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
--addr <Address>	Required	Specify the OTP address. Specify the address in 4-digit hexadecimal number. Example: 004C The tool checks that the entered value is a 4-digit hexadecimal number but does not check the address range.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py readotp --port COM9 --addr 004C
```

3.2.1.5 Set JTAG Authentication (setjauth)

When setting the JTAG authentication, specify the authentication mode; for JTAG authentication, see 9.3.8 in the RZ/T2M Group User's Manual: Hardware, see 9.3.8 in the RZ/T2L Group User's Manual: Hardware, or see 9.3.8 in the RZ/N2L Group User's Manual: Hardware, see 10.3.7 in the RZ/T2H and RZ/N2H Groups User's Manual: Hardware.

The options for the command to set JTAG Authentication are listed below.

Table 3.9 List of options for setjauth command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
--mode [authlv1 / authlv2 / prohibit]	Required	Select JTAG authentication mode. authlv1: Authentication by ID when connecting to JTAG (authentication level 1) authlv2: Authentication by ID when connecting to JTAG (authentication level 2) prohibit: JTAG connecting is permanently disabled.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py setjauth --port COM9 --mode authlv1
```

3.2.1.6 Get JTAG Authentication Settings (getjauth)

The options for the command to get JTAG Authentication are listed below.

Table 3.10 List of options for getjauth command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py getjauth --port COM9
```

3.2.1.7 Set JTAG Authentication ID (setjauthid)

When setting the JTAG Authentication ID, specify the JTAG authentication mode and the password file.

The options for the command to set JTAG Authentication ID are listed below.

Table 3.11 List of options for setjauthid command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
--mode [authlv1 / authlv2]	Required	Select JTAG authentication mode. authlv1: Authentication by ID when connecting to JTAG (authentication level 1) authlv2: Authentication by ID when connecting to JTAG (authentication level 2)
-i <File>	Required	Specify the file path of the password file.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py setjauthid --port COM9 --mode authlv1 -i
id_plain.bin
```

3.2.1.8 Set SCI/USB Boot Mode Disable (setsciboot, setusbboot)

When setting the SCI/USB boot mode disabled, specify the disable option; for SCI/USB boot mode disable, see 3.5.8 in the RZ/T2M Group User's Manual: Hardware, see 3.5.9 in the RZ/T2L Group User's Manual: Hardware, or see 3.5.9 in the RZ/N2L Group User's Manual: Hardware. This command is not supported by RZ/T2H and RZ/N2H.

The options for the command to set SCI/USB boot disabled are listed below.

Table 3.12 List of options for setsciboot/setusbboot command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
--disable	Required	Permanently disable SCI/USB boot.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py setsciboot --port COM9 --disable
```

3.2.1.9 Get SCI/USB Boot Mode Settings (getsciboot, getusbboot)

The options for the command to get SCI/USB boot mode settings are listed below. This command is not supported by RZ/T2H and RZ/N2H.

Table 3.13 List of options for getsciboot, getusbboot command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py getsciboot --port COM9
```

3.2.1.10 Read Unique ID (getuid)

The options for the command to get a unique ID are listed below.

Table 3.14 List of options for getuid command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py getuid --port COM9
```

3.2.1.11 Set SCI/USB Boot Authentication (setsauth)

When setting the SCI/USB boot authentication, specify the authentication mode; for SCI/USB boot authentication, see 4.6.10 in the RZ/T2H and RZ/N2H Groups User's Manual: Hardware. This command is not supported by RZ/T2M, RZ/T2L, and RZ/N2L.

The options for the command to set SCI/USB boot authentication are listed below.

Table 3.15 List of options for setsauth command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
--mode [auth / prohibit]	Required	Select SCI/USB boot authentication mode. auth: Authentication by ID when connecting to SCI/USB boot. prohibit:SCI/USB boot connecting is permanently disabled.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py setsauth --port COM9 --mode auth
```

3.2.1.12 Get SCI/USB Boot Authentication Settings (getsauth)

The options for the command to get SCI/USB boot authentication are listed below. This command is not supported by RZ/T2M, RZ/T2L, and RZ/N2L.

Table 3.16 List of options for getsauth command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py getsauth --port COM9
```

3.2.1.13 Set SCI/USB Boot Authentication ID (setsauthid)

When setting the SCI/USB boot authentication ID, specify the SCI/USB boot authentication mode and the password file. This command is not supported by RZ/T2M, RZ/T2L, and RZ/N2L.

The options for the command to set SCI/USB boot authentication ID are listed below.

Table 3.17 List of options for setsauthid command

Option	Required/Optional	Description
--port <Port>	Required	Specify the COM port to be used for communication with the device.
-i <File>	Required	Specify the file path of the password file.
-h	Optional	Show help messages.

The following is an example of tool execution.

```
> python device_setup.py setsauthid --port COM9 -i id_plain.bin
```

3.2.2 Input Files

3.2.2.1 Data to be written to flash (writeflash)

This tool does not check the data in the file specified as write data. The file is assumed to be a program file running on RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L or RZ/N2H, or a parameter file generated by parameter_generator.py.

If the file size exceeds the upper limit of DATA_MAX that can be sent at one time, the tool will split the file data and send the data.

Example: Each data size when splited into x pieces

1st block: sector_size - (write_addr - sector_size)

x - 1st block: SETUP_SEND_DATA_MAX

xth block: Remaining data (less than SETUP_SEND_DATA_MAX)

3.2.2.2 Password File (setjauthid, setsauthid)

The password is used as an authentication ID during JTAG authentication or SCI/USB boot authentication.

The authentication ID is assumed to be 128-bit plain-text binary data. The tool performs a size check of the input file, and an error is generated if the size is not 16 bytes.

3.3 Implementation Specifications of Device Setup Program

3.3.1 Development Environment

Refer to RZ/T2, RZ/N2 Getting Started with Flexible Software Package.

3.3.2 File Structure

The zip file in the package contains the Device Setup Sample program code for the GCC compiler and the IAR compiler. When you unzip the zip file, you will find folders named gcc and iccarm, which contain the projects for each compiler.

Table 3.18 lists the main files contained in the device setup program project for each compiler.

Table 3.18 File Structure of Device setup program

Folder Name	File Name	Description
RZ*_*_DeviceSetup_Rev*\		
├	*.jlink, *.launch, *project, *.eww, *.ewd, *.ewp	Project files
├	*.pincfg, *.xml, *.ipcf	Flexible Software Package Files
├	rz*_cfg.txt	
├ rz*\		
├ rz*_cfg\		
├ rz*_gen\		
├ script\	*.ld, *.icf	Memory allocation
└ src\	*.c, *.h	Source code folder

3.3.3 Memory Maps

Table 3.19 show memory maps for the device setup program for RZ/T2M. Table 3.20 show memory maps for the device setup program for RZ/T2L. Table 3.21 show memory maps for the device setup program for RZ/N2L. Table 3.22 show memory maps for the device setup program for RZ/T2H Cortex®-R52 and RZ/N2H Cortex®-R52.

Table 3.19 Memory Map for RZ/T2M

Memory Type	Address (Mirror area address)	Size	Description
ATCM	0x00000000 - 0x00002FFF	512 K byte	Device setup program area (stack/heap/bss)
	0x00003000 - 0x0007FFFF		Unused space.
BTCM	0x00100000 - 0x00101FFF	64 K byte	Boot loader area
	0x00102000 - 0x0010FFFF		Device setup program area (text/data)
System RAM	0x10000000 - 0x100FFFFF (0x10000000 - 0x100FFFFF)	1.5 M byte	Device setup program area (Transmit, receive buffers/flash write buffers, etc.)
	0x10100000 - 0x1017FFFF (0x30100000 - 0x3017FFFF)		Unused space.
External Memory (xSPI0)	0x60000000 - 0x67FFFFFF (0x40000000-0x47FFFFFF)	128 M Byte	Flash writes destination area (RSK+ has QSPI flash 64 M Byte)
External Memory (xSPI1)	0x68000000 - 0x6FFFFFFF (0x48000000-0x4FFFFFFF)	128 M Byte	-
External Memory (External bus)	0x70000000 - 0x7FFFFFFF (0x50000000-0x5FFFFFFF)	256 M Byte	Flash writes destination area (RSK+ has NOR Flash 32 M Byte)

Table 3.20 Memory Map for RZ/T2L

Memory Type	Address (Mirror area address)	Size	Description
ATCM	0x00000000 - 0x00002FFF	512 K byte	Device setup program area (stack/heap/bss)
	0x00003000 - 0x0007FFFF		Unused space.
BTCM	0x00100000 - 0x00101FFF	64 K byte	Boot loader area
	0x00102000 - 0x0010FFFF		Device setup program area (text/data)
System RAM	0x10000000 - 0x100FFFFF (0x30000000 - 0x300FFFFF)	1.0 M byte	Device setup program area (Transmit, receive buffers/flash write buffers, etc.)
External Memory (xSPI0)	0x60000000 - 0x67FFFFFF (0x40000000-0x47FFFFFF)	128 M Byte	Flash writes destination area (RSK+ has Octa flash 64 M Byte)
External Memory (xSPI1)	0x68000000 - 0x6FFFFFFF (0x48000000-0x4FFFFFFF)	128 M Byte	Flash writes destination area (RSK+ has QSPI flash 16 M Byte)
External Memory (External bus)	0x70000000 - 0x7FFFFFFF (0x50000000-0x5FFFFFFF)	256 M Byte	-

Table 3.21 Memory Map for RZ/N2L

Memory Type	Address (Mirror area address)	Size	Description
ATCM	0x00000000 - 0x00002FFF	128 K byte	Device setup program area (stack/heap/bss)
	0x00003000 - 0x0001FFFF		Unused space.
BTCM	0x00100000 - 0x00101FFF	128 K byte	Boot loader area
	0x00102000 - 0x0010FFFF		Device setup program area (Transmit, receive buffers/flash write buffers, etc.)
	0x00110000 - 0x0011FFFF		Unused space.
System RAM	0x10000000 - 0x100FFFFF (0x30000000 - 0x300FFFFF)	1.5 M byte	Device setup program area (Transmit, receive buffers/flash write buffers, etc.)
	0x10100000 - 0x1017FFFF (0x30100000 - 0x3017FFFF)		Unused space.
External Memory (xSPI0)	0x60000000 - 0x67FFFFFF (0x40000000-0x47FFFFFF)	128 M Byte	Flash writes destination area (RSK+ has QSPI flash 64 M Byte)
External Memory (xSPI1)	0x68000000 - 0x6FFFFFFF (0x48000000-0x4FFFFFFF)	128 M Byte	-
External Memory (External bus)	0x70000000 - 0x7FFFFFFF (0x50000000-0x5FFFFFFF)	256 M Byte	Flash writes destination area (RSK+ has NOR Flash 32 M Byte)

Table 3.22 Memory Map for RZ/T2H Cortex®-R52 and RZ/N2H Cortex®-R52

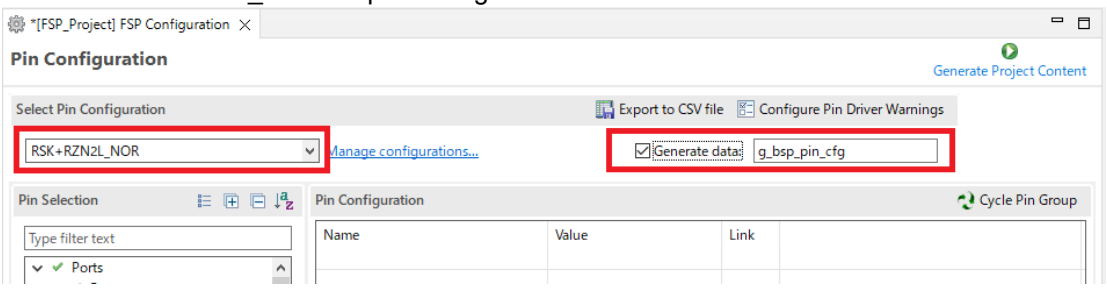
Memory Type	Address (Mirror area address)	Size	Description
ATCM	0x00000000 - 0x00002FFF	512 K byte	Device setup program area (stack/heap/bss)
	0x00003000 - 0x0007FFFF		Unused space.
BTCM	0x00100000 - 0x00101FFF	64 K byte	Boot loader area
	0x00102000 - 0x0010FFFF		Device setup program area (text/data)
System RAM	0x10000000 - 0x100FFFFF	2.0 M byte	Device setup program area (Transmit, receive buffers/flash write buffers, etc.)
	0x10100000 - 0x101FFFFF		Unused space.
External Memory (xSPI0)	0x40000000 - 0x4FFFFFFF	256 M Byte	Flash writes destination area (EVB has Octa flash 64 M Byte)
External Memory (xSPI1)	0x50000000 - 0x5FFFFFFF	256 M Byte	Flash writes destination area (EVB has QSPI flash 16 M Byte)

3.3.4 How to Use NOR Flash in the RZ/N2L Project

The device setup sample program project for RZ/N2L must be configured to use the external flash.

By default, the setting to use QSPI flash is enabled; to use NOR flash, the following settings are required.

1. Start FSP Configuration.
   For GCC version, use e^2 studio.
   For IAR version, use FSP Smart Configurator.
   For details, Refer to Getting Started with Flexible Software Package.
2. Select "RSK+RZN2L_NOR" in pin settings and enable Generate data check.



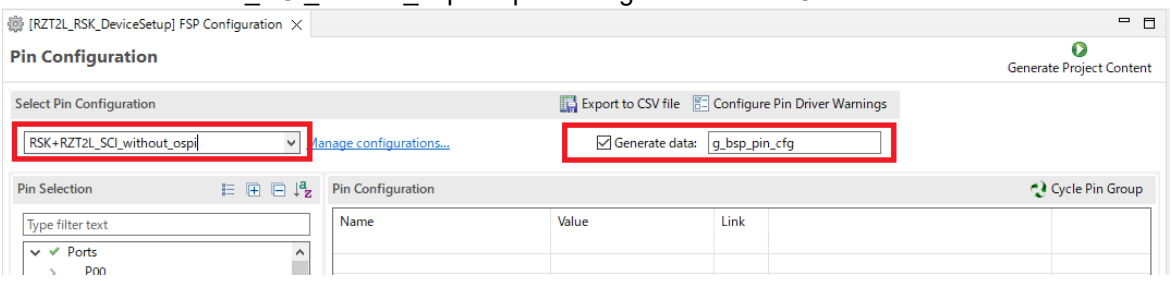
3. Click Generate Project Content (green play icon).
   Sample program code is generated that can use the NOR flash.

3.3.5 How to Use SCI Communication in the RZ/T2L Project

The device setup sample program project for RZ/T2L must be configured to use the SCI communication.

By default, the setting that only USB communication can be used is enabled. The following settings are required to use SCI communication. Also, when SCI communication is enabled, OSPI flash cannot be used.

1. Start FSP Configuration.
   For GCC version, use e^2 studio.
   For IAR version, use FSP Smart Configurator.
   For details, Refer to Getting Started with Flexible Software Package.
2. Select "RSK+RZT2L_SCI_without_ospi" in pin settings and enable Generate data check.



3. Click Generate Project Content (green play icon).
   Sample program code is generated that can use the SCI communication.

3.3.6 How to Build the Project

The following shows how to build the device setup sample program project and generate an S-Record format file that can be used in SCI boot mode and USB boot mode of the RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L or RZ/N2H.

Both projects for each compiler include command settings to generate SCI bootable/USB bootable S-Record format files (files with srec extension) after building.

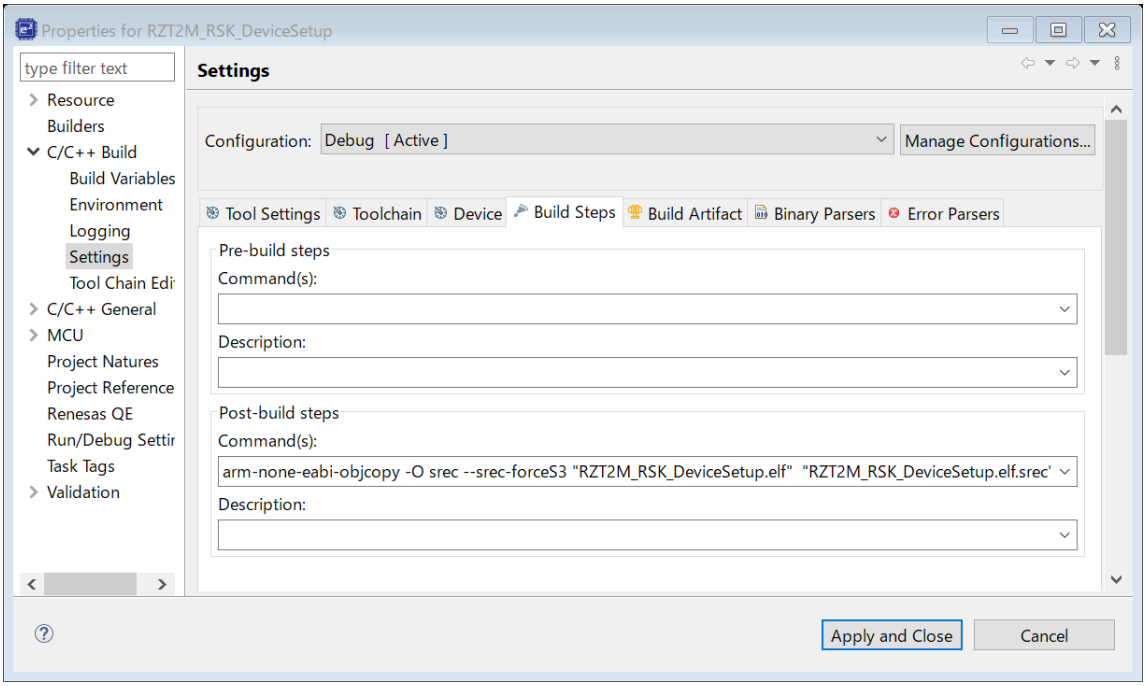
After importing the project into the IDE, build it as is.

The commands to generate S-Record format files are as follows.

In the GCC version, set the following command in Command(s) of the Post-build steps of e² studio:

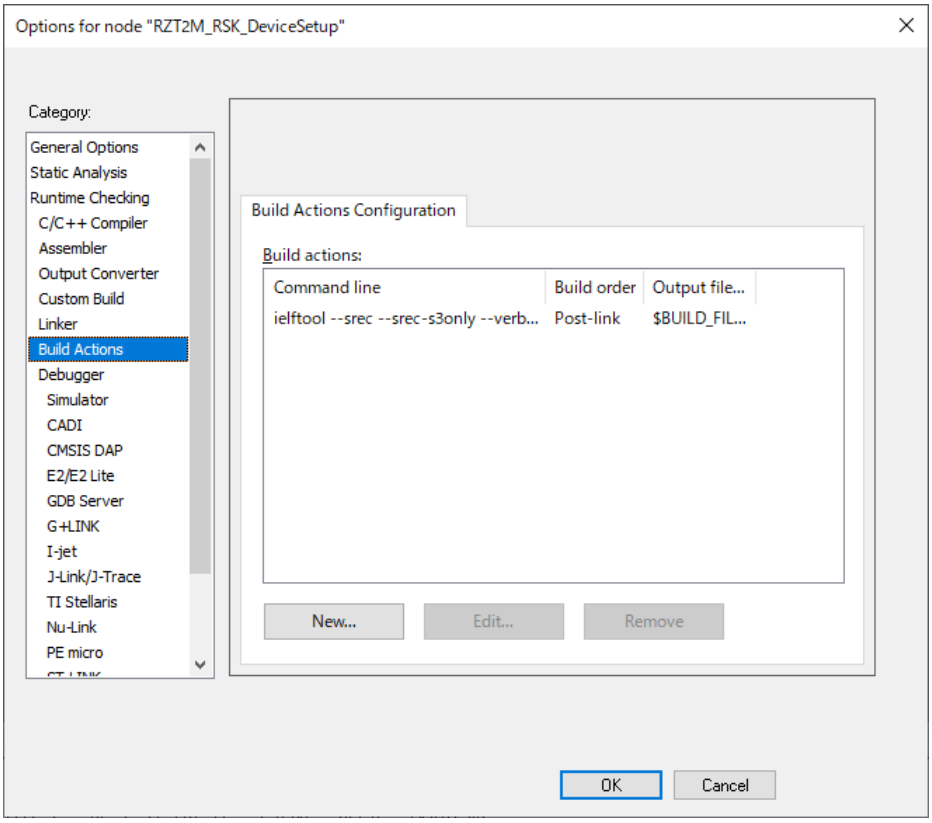
```
arm-none-eabi-objcopy -O srec --srec-forceS3 --srec-len 32
"RZT2M_RSK_DeviceSetup.elf" "RZT2M_RSK_DeviceSetup.elf.srec"
```

Please change the file names in the commands to match your project.



In the IAR version, set the build action for EWARM:

Create a new build action configuration.



Configure the commands described below.

Command line

```
ielftool --srec --srec-s3only --srec-len 32 --verbose "$TARGET_PATH$"
"$TARGET_PATH$.srec" && echo > "$BUILD_FILES_DIR$/.postbuild"
```

Output files (one per line)

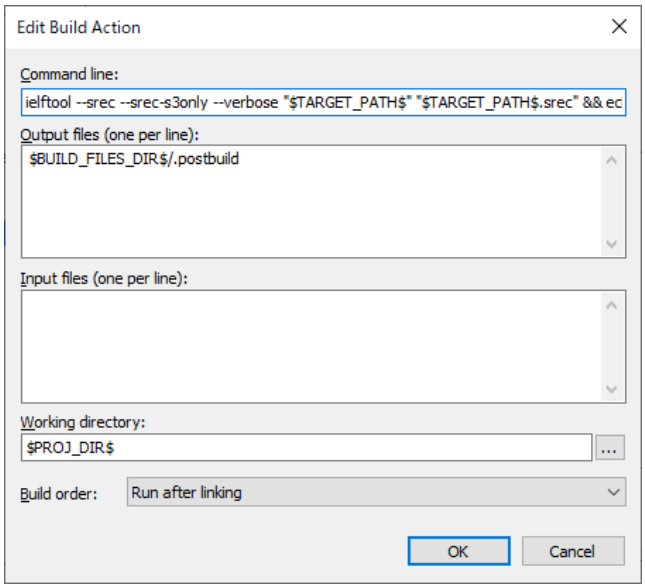
```
$BUILD_FILES_DIR$/.postbuild
```

Working directory

```
$PROJ_DIR$
```

Build order

```
Run after linking
```



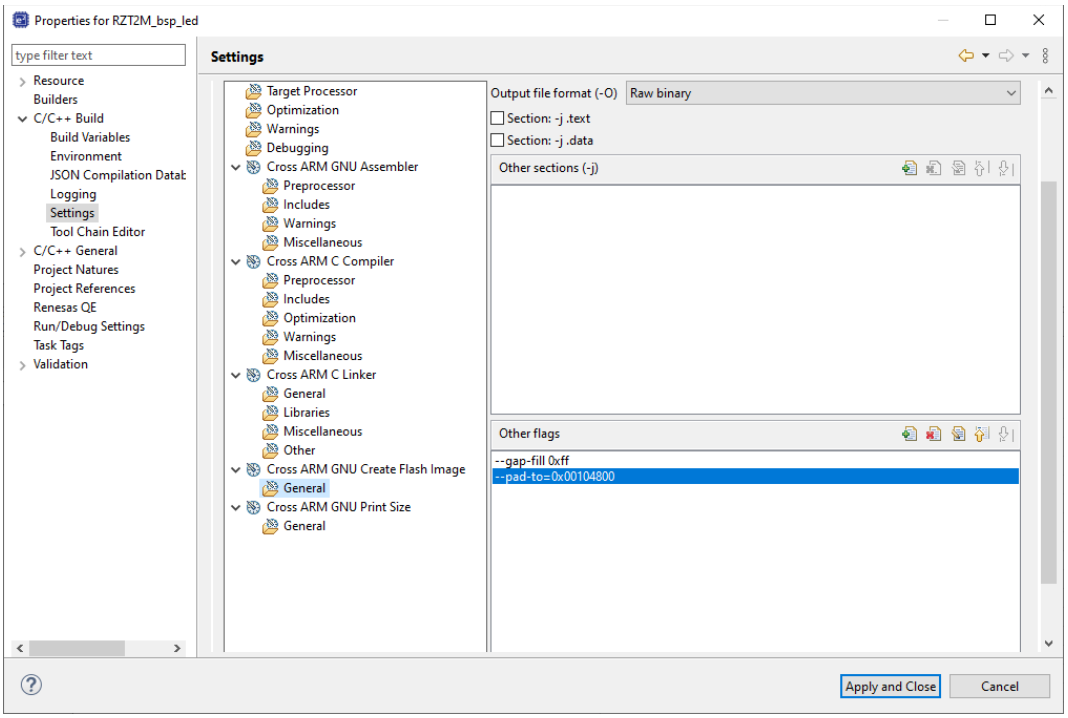
3.3.7 How to Adjust the Binary Data to 512-byte Units

In this section, introduce how to adjust the binary data to 512-byte units by adding project settings.

In the GCC version, set the following in Other flags of Cross ARM GNU Create Flash Image in the e^2 studio project properties:

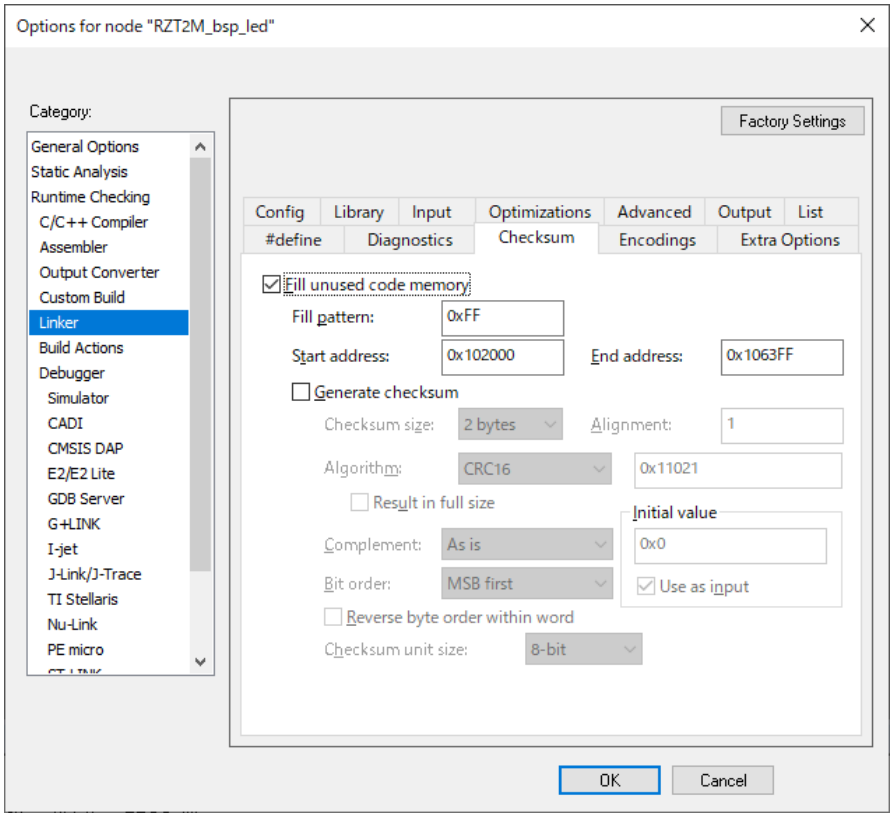
```
--pad-to=0x00104800
```

For the address value (e.g. 0x00104800), specify the range in 512-byte units that the program will fit into, starting from the start address of the program.



In the IAR version, set the Linker Checksum setting to Fill unused code memory:

For the Start address, specify the start address of the program. For the End address, specify the range from the start address of the program to the address that the program will fit into, in 512-byte units.



3.4 Communication Protocols of Device Setup Program

Figure 3.1 illustrates the communication protocols of the device setup program running on the device (RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L or RZ/N2H).

When the device setup program in the device receives a Request packet, it performs the setup process corresponding to the command code contained in the received data. After the setup process is completed, the result is sent in a Response packet.

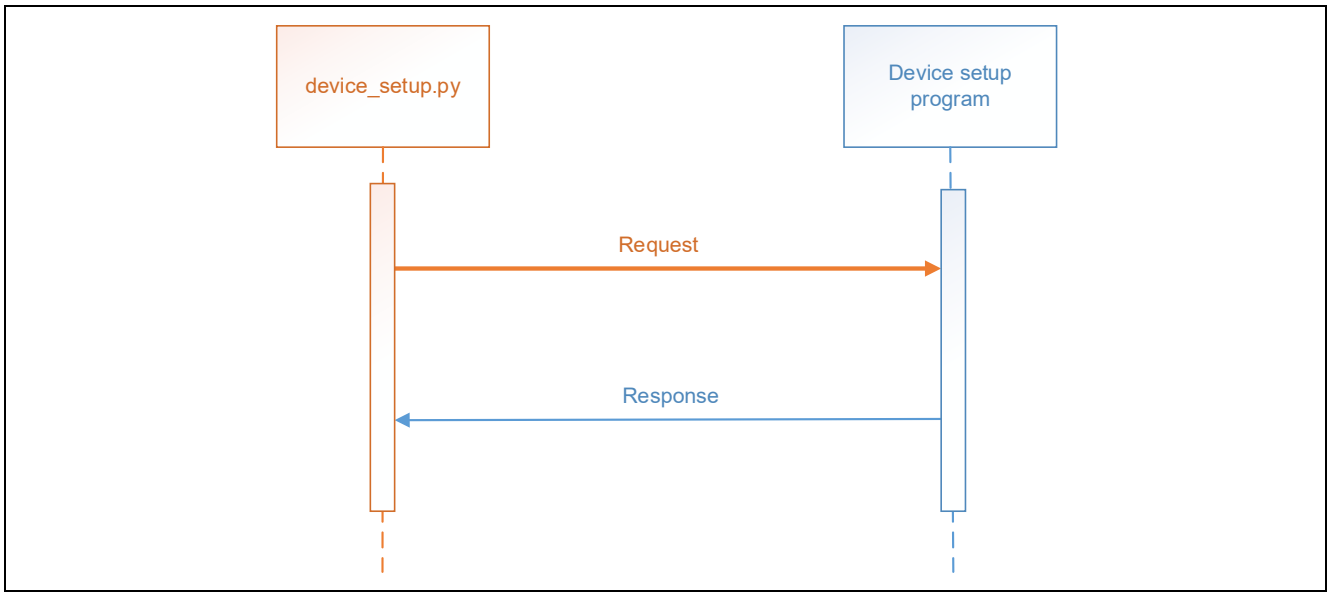


Figure 3.1 Communication protocols of device setup program

Table 3.23 shows the format of the control packets sent and received by the device setup program. Note that the unit of the Offset and Size values shown below is bytes.

Table 3.23 Control Packet Format

Offset	Field	Size	Value
0	Command Type	1	Command type
1	Command Code	1	Command code
2	Dummy	2	Dummy (For alignment adjustment)
4	Payload Size	4	Size of payload: n (little-endian)
8	Payload	n	Data of various types is stored here.
8 + n	Checksum	1	Check sum

The checksum value of 1 byte in the control packet is the sum of the data in each checksum target area added in 8-bit increments.

An example of checksum value calculation (packet sent during flash write) is shown below.

Target packet (WRITE_FLASH):

Command type = 50h

Command code = 11h

Dummy = 0000h

Payload size = 0000 0004h

Address = 5000 004Ch

Data = 1122 3344h

The checksum is calculated as follows (*0x00 is omitted):

CHECK_SUM = (50h) + (11h) + (04h) + (50h) + (4Ch) + (11h) + (22h) + (33h) + (44h) = (ABh)

Table 3.24 lists the command types and Table 3.25 lists the command codes. The contents of the packets corresponding to each command code are described in 3.4.1 to 3.4.9.

Table 3.24 Command Types

Command Types	Value	Description
Request	0x50	Request to the device
Response	0x55	Response from the device

Table 3.25 Command Codes

Command Code	Value	Description
WRITE_FLASH	0x11	Write to Flash
WRITE_OTP	0x12	Write to OTP
READ_OTP	0x13	Read from OTP
SET_JTAG_AUTH	0x14	Set JTAG authentication
GET_JTAG_AUTH	0x15	Get JTAG authentication settings
SET_JTAG_AUTHID	0x16	Set JTAG authentication ID
SET_SCIUSB	0x17	Set SCI/USB boot disabled
GET_SCIUSB	0x18	Get SCI/USB boot settings
GET_UID	0x21	Read unique ID
SET_SCIUSB_AUTH	0x22	Set SCI/USB boot authentication
GET_SCIUSB_AUTH	0x23	Get SCI/USB boot authentication settings
SET_SCIUSB_AUTHID	0x24	Set SCI/USB boot authentication ID

3.4.1 WRITE_FLASH

Table 3.26 Contents of WRITE_FLASH request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x11 (WRITE_FLASH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000004 + n
8	Address	4	Flash address to write to (8-digit)
12	Data	n	Write data
12 + n	Checksum	1	Check sum

Table 3.27 Contents of WRITE_FLASH response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x11 (WRITE_FLASH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000001
8	Result	1	Command execution result
9	Checksum	1	Check sum

3.4.2 WRITE_OTP

Table 3.28 Contents of WRITE_OTP request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x12 (WRITE_OTP)
2	Dummy	2	Dummy
4	Payload size	4	0x00000006
8	Address	2	OTP address to write to (4-digit)
10	Data	4	Write data (32 bits)
14	Checksum	1	Check sum

Table 3.29 Contents of WRITE_OTP response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x12 (WRITE_OTP)
2	Dummy	2	Dummy
4	Payload size	4	0x00000001
8	Result	1	Command execution result
9	Checksum	1	Check sum

3.4.3 READ_OTP

Table 3.30 Contents of READ_OTP request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x13 (READ_OTP)
2	Dummy	2	Dummy
4	Payload size	4	0x00000002
8	Address	2	OTP address to read to (4-digit)
10	Checksum	1	Check sum

Table 3.31 Contents of READ_OTP response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x13 (READ_OTP)
2	Dummy	2	Dummy
4	Payload size	4	0x00000001 + n
8	Result	1	Command execution result
9	Read data	n	Read data
9 + n	Checksum	1	Check sum

3.4.4 SET_JTAG_AUTH

Table 3.32 Contents of SET_JTAG_AUTH request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x14 (SET_JTAG_AUTH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000002
8	Mode	1	JATG Authentication Mode
9	Type	1	JTAG Authentication Type (Fixed 0)
10	Checksum	1	Check sum

Table 3.33 Contents of SET_JTAG_AUTH response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x14 (SET_JTAG_AUTH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000001
8	Result	1	Command execution result
9	Checksum	1	Check sum

3.4.5 GET_JTAG_AUTH

Table 3.34 Contents of GET_JTAG_AUTH request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x15 (GET_JTAG_AUTH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000000
8	Checksum	1	Check sum

Table 3.35 Contents of GET_JTAG_AUTH response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x15 (GET_JTAG_AUTH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000002
8	Result	1	Command execution result
9	Mode	1	JATG Authentication Mode
10	Checksum	1	Check sum

3.4.6 SET_JTAG_AUTHID

Table 3.36 Contents of SET_JTAG_AUTHID request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x16 (SET_JTAG_AUTHID)
2	Dummy	2	Dummy
4	Payload size	4	0x00000012
8	Mode	1	JATG Authentication Mode
9	Type	1	JTAG Authentication Type (Fixed 0)
10	ID	16	ID plain
26	Checksum	1	Check sum

Table 3.37 Contents of SET_JTAG_AUTHID response packet

Offset	Field	Size	Value
0	Type	1	0x55（Response）
1	Command Code	1	0x16（SET_JTAG_AUTHID）
2	Dummy	2	Dummy
4	Payload size	4	0x00000001
8	Result	1	Command execution result
9	Checksum	1	Checksum

3.4.7 SET_SCIUSB

Table 3.38 Contents of SET_SCIUSB request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x17 (SET_SCIUSB)
2	Dummy	2	Dummy
4	Payload size	4	0x00000002
8	Mode	1	SCI/USB Boot Mode Disable (Fixed 0x01)
9	Checksum	1	Checksum

Table 3.39 Contents of SET_SCIUSB response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x17 (SET_SCIUSB)
2	Dummy	2	Dummy
4	Payload size	4	0x00000001
8	Result	1	Command execution result
9	Checksum	1	Check sum

3.4.8 GET_SCIUSB

Table 3.40 Contents of GET_SCIUSB request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x18 (GET_SCIUSB)
2	Dummy	2	Dummy
4	Payload size	4	0x00000000
8	Checksum	1	Check sum

Table 3.41 Contents of GET_SCIUSB response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x18 (GET_SCIUSB)
2	Dummy	2	Dummy
4	Payload size	4	0x00000002
8	Result	1	Command execution result
9	Mode	1	SCI/USB Boot Mode
10	Checksum	1	Check sum

3.4.9 GET_UID

Table 3.42 Contents of GET_UID request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x21 (GET_UID)
2	Dummy	2	Dummy
4	Payload size	4	0x00000000
8	Checksum	1	Check sum

Table 3.43 Contents of GET_UID response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x21 (GET_UID)
2	Dummy	2	Dummy
4	Payload size	4	0x00000009
8	Result	1	Command execution result
9	Unique ID	8	Unique ID
17	Checksum	1	Check sum

3.4.10 SET_SCIUSB_AUTH

Table 3.44 Contents of SET_SCIUSB_AUTH request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x22 (SET_SCIUSB_AUTH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000002
8	Mode	1	SCI/USB Boot Authentication Mode
9	Type	1	SCI/USB Boot Authentication Type (Fixed 0)
10	Checksum	1	Check sum

Table 3.45 Contents of SET_SCIUSB_AUTH response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x22 (SET_SCIUSB_AUTH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000001
8	Result	1	Command execution result
9	Checksum	1	Check sum

3.4.11 GET_SCIUSB_AUTH

Table 3.46 Contents of GET_SCIUSB_AUTH request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x23 (GET_SCIUSB_AUTH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000000
8	Checksum	1	Check sum

Table 3.47 Contents of GET_SCIUSB_AUTH response packet

Offset	Field	Size	Value
0	Type	1	0x55 (Response)
1	Command Code	1	0x23 (GET_SCIUSB_AUTH)
2	Dummy	2	Dummy
4	Payload size	4	0x00000002
8	Result	1	Command execution result
9	Mode	1	SCI/USB Boot Authentication Mode
10	Checksum	1	Check sum

3.4.12 SET_SCIUSB_AUTHID

Table 3.48 Contents of SET_SCIUSB_AUTHID request packet

Offset	Field	Size	Value
0	Command type	1	0x50 (Request)
1	Command code	1	0x24 (SET_SCIUSB_AUTHID)
2	Dummy	2	Dummy
4	Payload size	4	0x00000012
8	Type	1	SCI/USB Boot Authentication Type (Fixed 0)
9	ID	16	ID plain
25	Checksum	1	Check sum

Table 3.49 Contents of SET_SCIUSB_AUTHID response packet

Offset	Field	Size	Value
0	Type	1	0x55（Response）
1	Command Code	1	0x24（SET_SCIUSB_AUTHID）
2	Dummy	2	Dummy
4	Payload size	4	0x00000001
8	Result	1	Command execution result
9	Checksum	1	Checksum

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun 30, 2022	-	First edition issued
1.01	Aug 26, 2022	p1, p14, p18, p21, p26, p29	2.3.2, 3.2, 3.4 Changed SCI/USB boot authentication
		p23, p24	3.3.4 Add How to Build the Project
1.10	Oct 7, 2022	-	RZ/N2L is supported
1.20	Apr 28, 2023	-	RZ/T2L is supported
2.00	Apr 19, 2024	-	Updated FSP for RZ/T2M and RZ/T2L sample projects to RZ/T2 FSP v2.0.0. Updated explanations of the sample programs including tools.
2.10	Jun 14, 2024	-	Updated FSP for RZ/N2L sample projects to RZ/N2 FSP v2.0.0.
2.11	Sep 13, 2024	p3, p4	The limitation of USB booting on RZ/T2M and RZ/N2L has been tentatively resolved.
2.20	Nov 15, 2024	-	RZ/T2H Cortex®-R52 core is supported. Updated FSP for RZ/T2M and RZ/T2L sample projects to RZ/T2 FSP v2.2.0.
		p10, p11, p13, p42, p43	Added information how to adjust the binary data to 512-byte units by adding project settings.
		p10-p25	The procedure for programming user programs to external flash memory has been changed to explain it for each target device.
2.30	Dec 13, 2024	-	RZ/N2H Cortex®-R52 core is supported. Updated FSP for RZ/N2L sample projects to RZ/N2 FSP v2.1.0.
2.40	Mar 31, 2025	-	RZ/T2H Cortex®-A55 and RZ/N2H Cortex®-A55 core are supported.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.