

# RZ/T2, RZ/N2

## Getting Started with Flexible Software Package

---

### Introduction

This manual describes how to use the Renesas Flexible Software Package (FSP) for writing applications for the RZ/T2, RZ/N2 microprocessor series.

### Target Device

RZ/T series: RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H

RZ/N series: RZ/N2L, RZ/N2H

### About the Video Contents

We provide videos about the development tools using the RZ/T and RZ/N FSP. Access the following links:

- How to install the development tools
  - [RZ/T RZ/N FSP Quick Start Guide - FSP Installation and Generating Your First Project for e2 studio](#)  
([English](#), [Japanese](#), [Chinese](#))
  - [RZ/T RZ/N FSP Quick Start Guide - FSP Installation & Generating Your First Project for EWARM & FSP SC](#)  
([English](#), [Japanese](#), [Chinese](#))
- Instructions and usage of each tab in FSP Configuration
  - [RZ/T RZ/N FSP Tutorial - Pin Configuration Function](#)  
([English](#), [Japanese](#), [Chinese](#))
  - [RZ/T RZ/N FSP Tutorial for FSP Configuration \(1/2\) - Introduction of Tabs](#)  
([English](#), [Japanese](#), [Chinese](#))
  - [RZ/T RZ/N FSP Tutorial for FSP Configuration \(2/2\) - How to Use](#)  
([English](#), [Japanese](#), [Chinese](#))

## Contents

1.	Introduction .....	5
1.1	Overview .....	5
1.2	Introduction to FSP .....	5
1.2.1	Purpose .....	5
1.2.2	e <sup>2</sup> studio IDE .....	5
1.2.3	FSP SC .....	5
1.2.4	FSP Documentation .....	5
1.3	Related Documentation Files .....	6
1.3.1	Evaluation Board User's Manual .....	6
1.3.2	FSP Documentation .....	6
1.4	Starting Development Introduction .....	7
2.	Starting Development Introduction .....	8
2.1	e <sup>2</sup> studio setup .....	8
2.1.1	What is e <sup>2</sup> studio .....	8
2.1.2	e <sup>2</sup> studio Prerequisites .....	8
2.1.3	e <sup>2</sup> studio installation for Windows PC .....	8
2.1.4	e <sup>2</sup> studio installation for Linux PC .....	17
2.2	FSP Setup .....	23
2.2.1	Installation of FSP using Platform Installer .....	23
2.2.2	Installation of FSP using Package Installer .....	28
2.2.3	Installation of FSP Packs using Package Zip file .....	30
3.	Set up the Evaluation Board .....	32
3.1	Obtaining an Evaluation Board .....	32
3.2	System Configuration .....	32
3.3	Supported Emulator .....	33
3.3.1	SEGGER J-Link .....	33
3.3.2	IAR I-Jet .....	33
3.4	RZ/T Series Board Setup .....	34
3.4.1	RSK+RZT2M .....	34
3.4.2	RSK+RZT2L .....	37
3.4.3	RSK+RZT2ME .....	39
3.4.4	RZ/T2H Evaluation Board .....	40
3.5	RZ/N Series Board Setup .....	43
3.5.1	RSK+RZN2L .....	43
3.5.2	RZ/N2H Evaluation Board .....	46
4.	Tutorial: Your First RZ MPU Project – Blinky .....	49
4.1	Tutorial Blinky .....	49
4.2	What Does Blinky Do? .....	49

4.3	Create a New Project for Blinky .....	49
4.3.1	Details about the Blinky Configuration .....	57
4.3.2	Configuring the Blinky Clocks.....	57
4.3.3	Configuring the Blinky Pins .....	57
4.3.4	Configuring the Parameters for Blinky Components .....	57
4.3.5	Where is main()? .....	57
4.3.6	Blinky Example Code .....	57
4.4	Build the Blinky Project.....	58
4.4.1	Build.....	58
4.4.2	Build for Multiprocessing .....	58
4.5	Debug the Blinky Project .....	59
4.5.1	Debug Prerequisites .....	59
4.5.2	Debug Steps.....	59
4.5.3	Details about the Debug Process.....	63
4.6	Run the Blinky Project.....	63
4.7	Debug and Run for Multiprocessing .....	64
4.8	Import the Project .....	66
5.	FSP SC User Guide .....	70
5.1	What is FSP SC?.....	70
5.2	Tutorial Blinky.....	70
5.3	Using FSP SC with IAR EWARM.....	71
5.3.1	Prerequisites.....	71
5.3.2	Create a New Project .....	72
5.3.3	Build the Project .....	84
5.3.4	Download & Debug the Project .....	89
5.3.5	Debug for Multiprocessing.....	93
5.4	Re-configuring Project with FSP SC .....	94
5.4.1	Launch FSP SC from IAR EWARM.....	94
5.4.2	Launch from the Command Prompt.....	95
5.5	Note when debugging in different workspaces .....	95
6.	FSP Configuration Users Guide .....	96
6.1	What is a Project? .....	96
6.2	Create a Project.....	98
6.2.1	Creating a New Project .....	98
6.2.2	Selecting a Board and Toolchain .....	99
6.2.3	Selecting a Project Template .....	101
6.2.4	Duplication of Resources .....	102
6.3	Configuring a Project.....	103
6.3.1	Summary Tab.....	103
6.3.2	Configuring the BSP.....	104

6.3.3	Configuring Clocks .....	105
6.3.4	Configuring Pins .....	106
6.4	Configuring Interrupts from the Stacks Tab .....	109
6.4.1	Creating Interrupts from the Interrupts Tab .....	109
6.4.2	Viewing Event Links .....	110
6.5	Adding and Configuring HAL Drivers .....	111
6.6	Reviewing and Adding Components .....	112
7.	Appendix.....	113
7.1	Known Issues .....	113
7.2	Tool Software Limitations .....	150
8.	Appendix.....	167
8.1	How to Debug the FSP Project with Flash Boot Mode.....	167
9.	Appendix.....	171
9.1	How to Erase Flash Memory .....	171
10.	Appendix.....	176
10.1	How to Change Boot Mode of the FSP Project.....	176
	Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio .....	179
	Multiprocessing with 2 cores .....	179
	Multiprocessing with 3 or more cores .....	186
	Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM .....	189
	Multiprocessing with 2 cores .....	189
	Multiprocessing with 3 or more cores .....	196
	Revision History .....	200

## 1. Introduction

### 1.1 Overview

This application note describes how to use the Renesas Flexible Software Package (FSP) running on the Cortex®-R52 and Cortex®-A55 (hereinafter referred to as CR52 and CA55) incorporated on RZ/T2 and RZ/N2.

### 1.2 Introduction to FSP

#### 1.2.1 Purpose

The Renesas Flexible Software Package (FSP) is an optimized software package designed to provide easy-to-use, scalable, high-quality software for embedded system design. The primary goal is to provide lightweight, efficient hardware abstraction layer (HAL) drivers and the board support package (BSP) that meet common use cases in embedded systems.

#### 1.2.2 e<sup>2</sup> studio IDE

FSP provides a host of efficiency-enhancing tools for developing projects targeting the Renesas RZ/T2, RZ/N2 series of MPU devices. The e<sup>2</sup> studio IDE provides a familiar development cockpit from which the key steps of project creation, module selection and configuration, code development, code generation, and debugging are all managed.

#### 1.2.3 FSP SC

The Renesas FSP Smart Configurator (FSP SC) is a desktop application designed to configure device hardware, such as clock setup and pin assignment, as well as initialization of FSP software components when using a 3<sup>rd</sup> party IDE and toolchain.

For creating the RZ/T2, RZ/N2 project, the FSP SC can currently be used with IAR Systems Embedded Workbench for Arm (IAR EWARM) with IAR toolchain for Arm.

#### 1.2.4 FSP Documentation

The related file, "FSP Documentation" contains HTML documentations describing the features, APIs, and usage notes regarding the BSP and HAL drivers implemented as FSP modules and interfaces. After clicking the "**index.html**" in "**FSP Documentation**" to open the introduction page on your HTML browser, the reference documents for utilizing each FSP module and interface can be read from the 'API Reference' menu.

### 1.3 Related Documentation Files

The related documentation files are shown in the following.

#### 1.3.1 Evaluation Board User's Manual

This Getting Started Guide refers to the following "Evaluation Board User's Manual".

- RZ/T series
  - RZ/T2M Group Renesas Starter Kit+ for RZ/T2M User's Manual (RZ/T2M and RZ/T2ME)
    - Document No. **R20UT4939**
  - RZ/T2L Group Renesas Starter Kit+ for RZ/T2L User's Manual
    - Document No. **R20UT5164**
  - RZ/T2H Group RZ/T2H Evaluation Board User's Manual
    - Document No. **R20UT5405**
- RZ/N series
  - RZ/N2L Group Renesas Starter Kit+ for RZ/N2L User's Manual
    - Document No. **R20UT4984**
  - RZ/N2H Group RZ/N2H Evaluation Board User's Manual
    - Document No. **R20UT5522**

These documents can be found on Renesas website by entering their **Document No.** into a search box.

- URL: <https://www.renesas.com/>

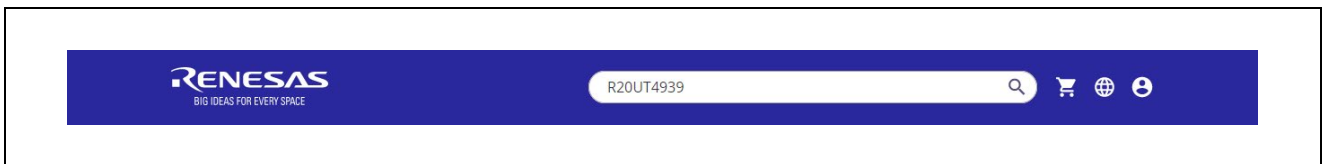


Figure 1. Search Box in Renesas Web Page

#### 1.3.2 FSP Documentation

This Getting Started Guide refers to the following "FSP Documentation". It contains notes on the use of the software modules packaged with FSP.

These documents are available in the Renesas Git repository on GitHub.

- RZ/T, N series
  - RZ Flexible Software Package Documentation
    - URL: <https://github.com/renesas/rz-fsp/releases>
      - File name: fsp\_documentation\_vx.x.x.zip

**Note:** The "vx.x.x" is the FSP version number, such as "v1.0.0".

## 1.4 Starting Development Introduction

The FSP application project can be created by e<sup>2</sup> studio or FSP SC (for IAR EWARM), and this Getting Started includes a tutorial for both tools; the chapters you should read change.

### **e<sup>2</sup> studio users should read the following chapters:**

- Chapter 2 “Starting Development Introduction”
- Chapter 3 “Set up the Evaluation Board”
- Chapter 4 “Tutorial: Your First RZ MPU Project – Blinky”
- Chapter 6 “FSP Configuration Users Guide”

### **FSP SC users (for IAR EWARM users) should read the following chapters:**

- Chapter 3 “Set up the Evaluation Board”
- Chapter 5 “FSP SC User Guide”
- Chapter 6 “FSP Configuration Users Guide”

The summary of each chapter is shown below.

- Chapter 2 “Starting Development Introduction”
  - Explains the setup of e<sup>2</sup> studio for utilizing FSP.
- Chapter 3 “Set up the Evaluation Board”
  - Explains how to set up the Evaluation Board to proceed with the tutorials in Chapter 4 and 5.
- Chapter 4 “Tutorial: Your First RZ MPU Project – Blinky”
  - Explains the tutorial with minimal steps to create, run, and debug an FSP project by using e<sup>2</sup> studio.
- Chapter 5 “FSP SC User Guide”
  - Explains the tutorial with minimal steps to create an FSP project as an IAR EWARM project by using the FSP SC, and to run and debug the created IAR EWARM project.
- Chapter 6 “FSP Configuration Users Guide”
  - Explains how to create and configure an FSP project in detail.
  - The explanation is described based on e<sup>2</sup> studio, but most of the explanations are applied to the FSP SC.

## 2. Starting Development Introduction

### 2.1 e<sup>2</sup> studio setup

#### 2.1.1 What is e<sup>2</sup> studio

Renesas e<sup>2</sup> studio is a development tool encompassing code development, build, and debug. e<sup>2</sup> studio is based on the open-source Eclipse IDE and the associated C/C++ Development Tooling (CDT).

When developing for RZ MPUs, e<sup>2</sup> studio hosts the Renesas Flexible Software Package (FSP). FSP provides a wide range of time-saving tools to simplify the selection, configuration, and management of modules and threads, to easily implement complex applications.

#### 2.1.2 e<sup>2</sup> studio Prerequisites

##### 2.1.2.1 Obtaining an RZ MPU Kit

To develop applications with RZ FSP, start with each Evaluation Board Kit / Renesas Starter Kit. The start-up guide of each Evaluation Board Kit is available below.

- RZ/N series
  - [Renesas Starter Kit+ for RZ/N2L Quick Start Guide](#)
  - [RZ/N2H Evaluation Board Kit Quick Start Guide](#)
- RZ/T series
  - [Renesas Starter Kit+ for RZ/T2M Quick Start Guide](#)
  - [Renesas Starter Kit+ for RZ/T2ME Quick Start Guide](#)
  - [Renesas Starter Kit+ for RZ/T2L Quick Start Guide](#)
  - [RZ/T2H Evaluation Board Kit Quick Start Guide](#)

##### 2.1.2.2 PC Requirements

The following are the minimum PC requirements to use e<sup>2</sup> studio:

- Windows 10 with Intel i5 or i7, or AMD A10-7850K or FX
- Memory: 8-GB DDR3 or DDR4 DRAM (16-GB DDR4/2400-MHz RAM is preferred)
- Minimum 250 GB hard disk

##### 2.1.2.3 Licensing

FSP licensing includes full source code, limited to Renesas hardware only.

### 2.1.3 e<sup>2</sup> studio installation for Windows PC

This chapter describes how to install the e<sup>2</sup> studio IDE on a Windows PC.

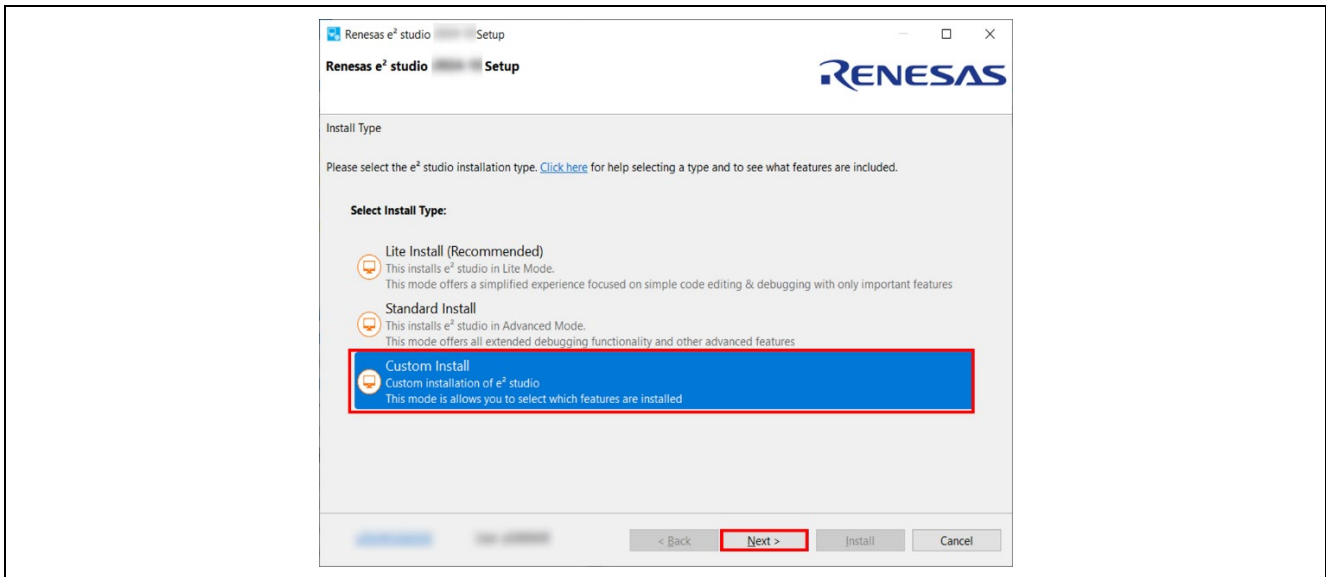
#### 2.1.3.1 Download

The latest e<sup>2</sup> studio IDE installer package can be downloaded from Renesas website for free. Please check the detailed information from: <https://www.renesas.com/e2studio>. Note that the user has to log in to the Renesas account (in the MyRenesas page) for the software download.

#### 2.1.3.2 Installation of e<sup>2</sup> studio IDE

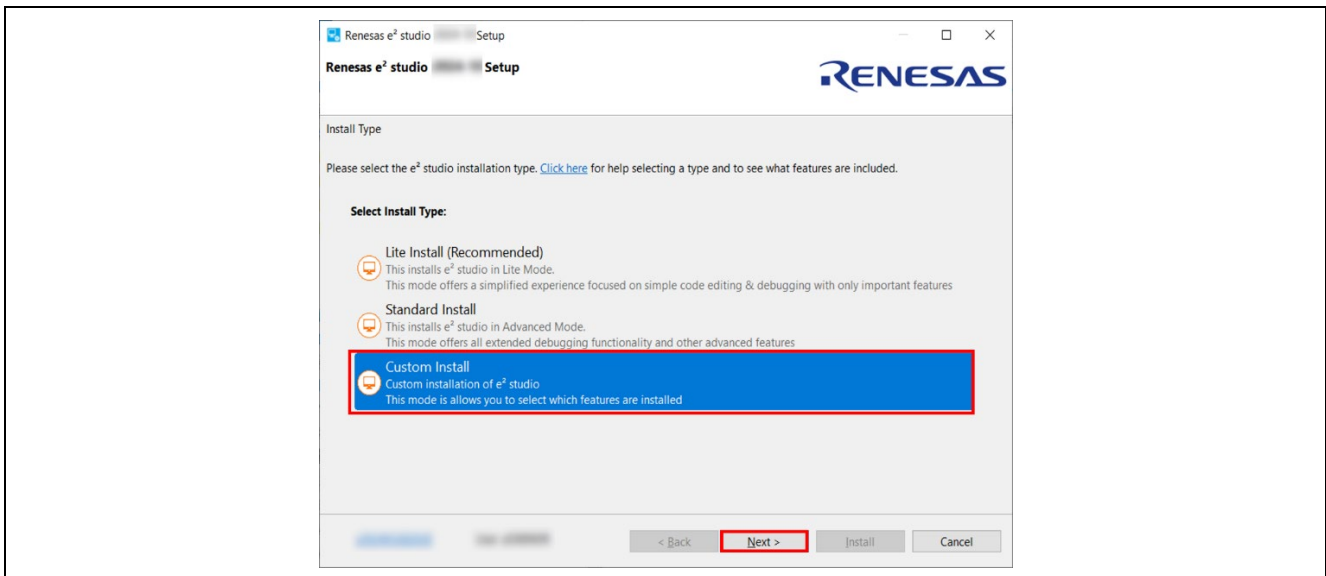
1. Double-click on the e<sup>2</sup> studio installer to invoke the e<sup>2</sup> studio installation wizard page. First, you need to select the Install Type. In this material, it is expected that Custom Install will be selected. Then, click [Next >] to continue.

**Note:** If e<sup>2</sup> studio was installed in your PC, the option to modify, remove the existing version, or install the e<sup>2</sup> studio will be displayed in a different location.



**Figure 2. e² studio installation wizard**

**Note:** If you are using a multi-user environment, you may receive a prompt to confirm whether you want to install it for the current user only or for all users.



**Figure 3. e² studio installation wizard**

## 2. Welcome page

The user can change the installation folder by clicking [Change...]. Click [Next] to continue.

**Note 1:** If you would like to have multiple versions of e² studio, please specify a new folder here.

**Note 2:** Multi-byte characters cannot be used for the e² studio installation folder name.

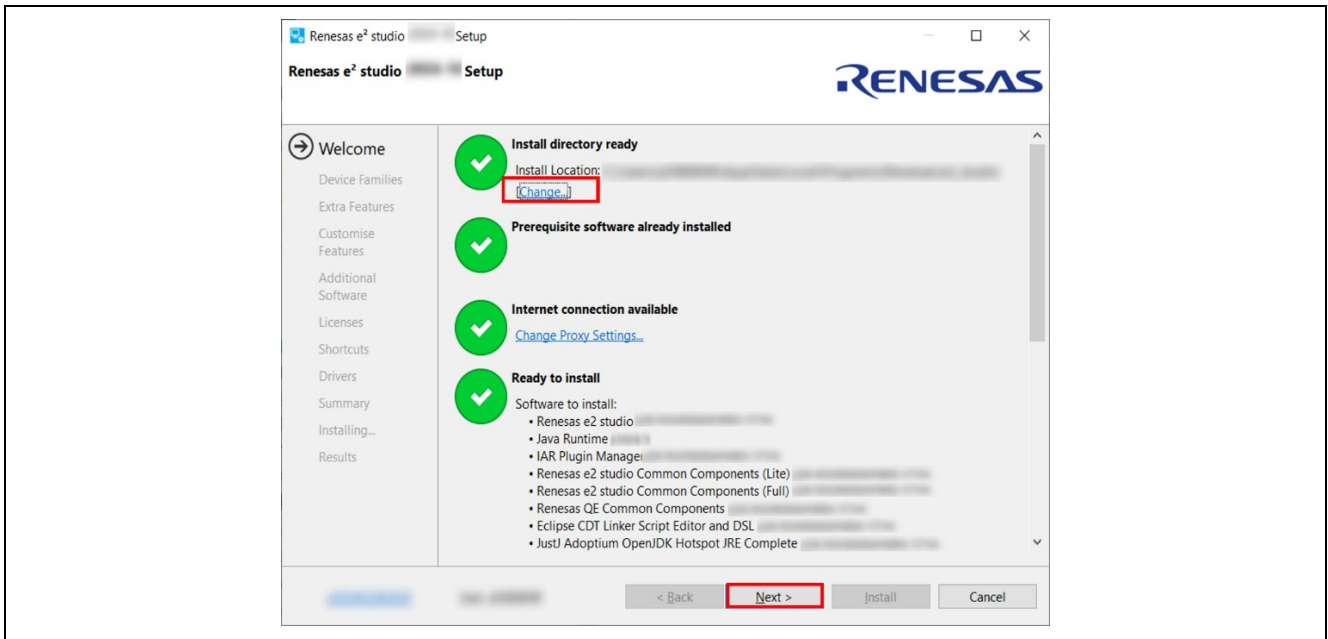


Figure 4. Installation of e<sup>2</sup> studio – Welcome page

### 3. Device Families

Select Device Families to install. Click the [Next] button to continue.

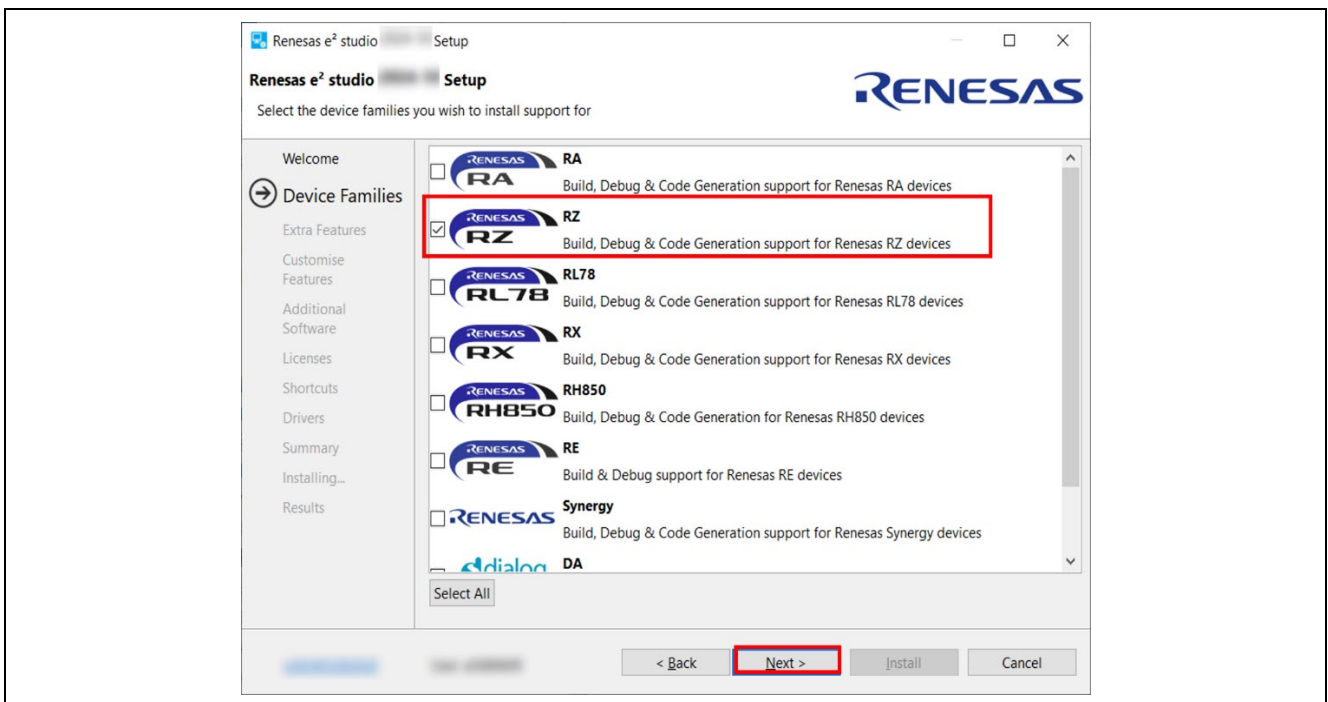


Figure 5. Installation of e<sup>2</sup> studio – Device Families

### 4. Extra Features

Select Extra Features (i.e., Language packs, SVN & Git support, RTOS support...) to be installed. For non-English language users, please select Language packs at this step if needed.

Click the [Next] button to continue.

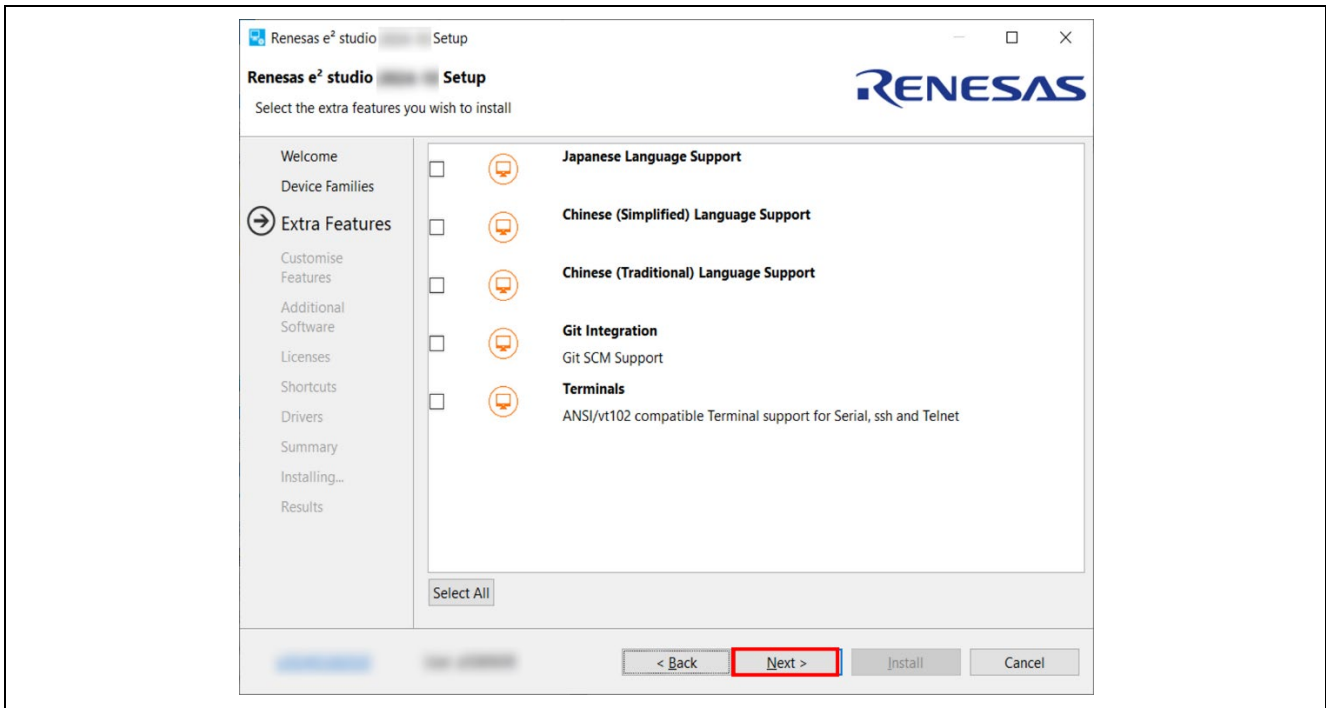


Figure 6. Installation of e² studio – Extra Features

#### 5. Customize Features

Select the components to install and click the [Next] button to continue. Be sure that Renesas FSP Smart Configurator Core and Renesas FSP Smart Configurator ARM are selected.

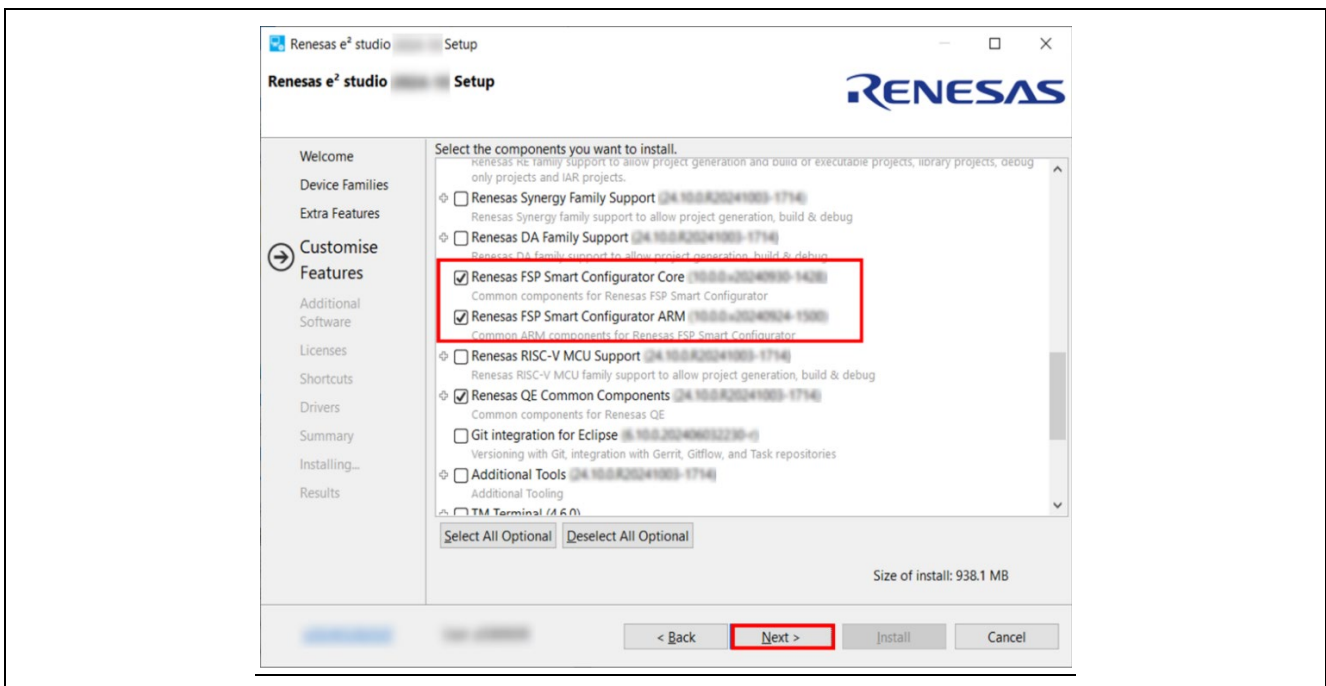
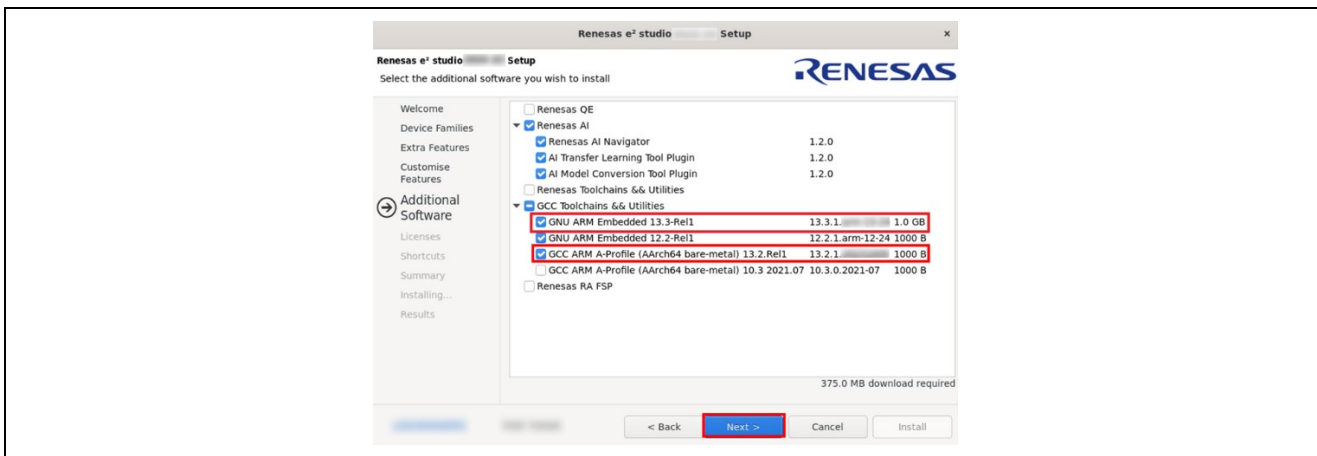


Figure 7. Installation of e² studio – Features

#### 6. Additional Software

Select additional software (i.e., compilers, utilities, QE...) to be installed. Be sure to select the following item and click [Next] to continue.

- GNU ARM Embedded 13.3-Rel1 (for Cortex-M and Cortex-R)
- GCC ARM A-Profile (AArch64 bare-metal) 13.2.Rel1 (for Cortex-A)

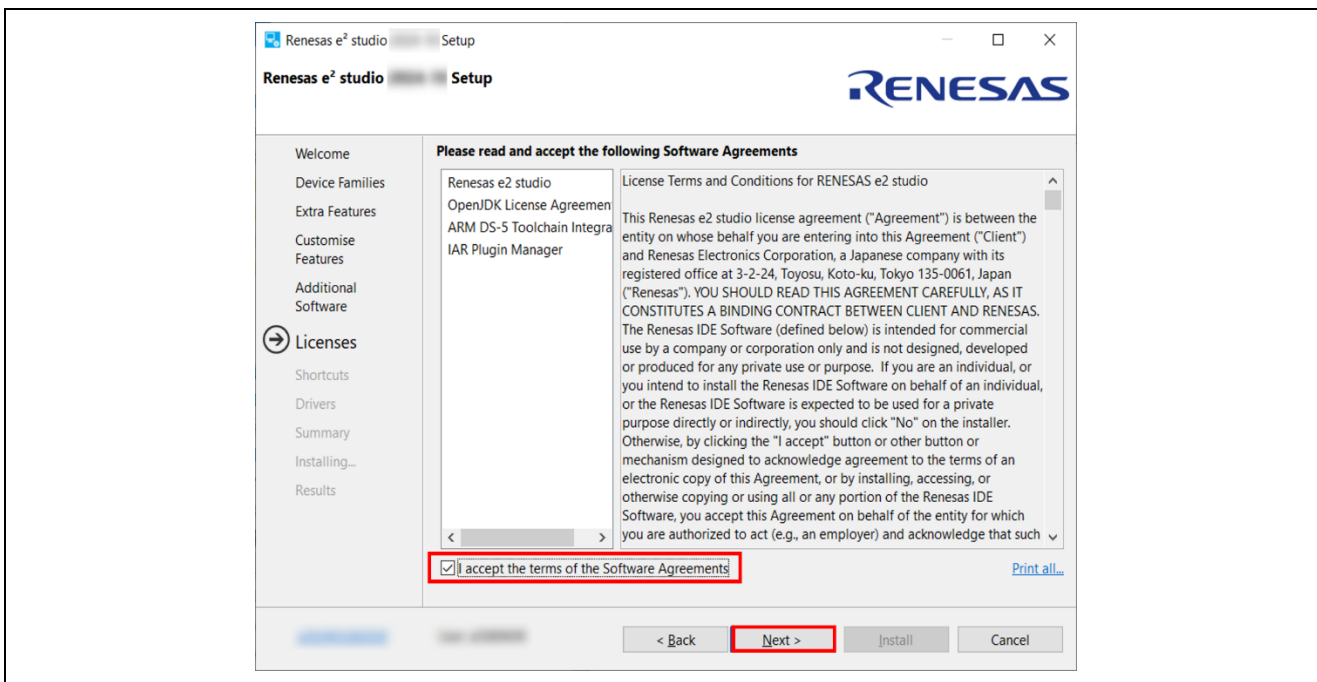


**Figure 8. Installation of e² studio – Additional Software**

For more details on the installation of Additional Software, please see section 2.1.3.3.

7. License Agreement

Read and accept the software license agreement. Click the [Next] button. Please note that the user must accept the license agreement; installation cannot continue.

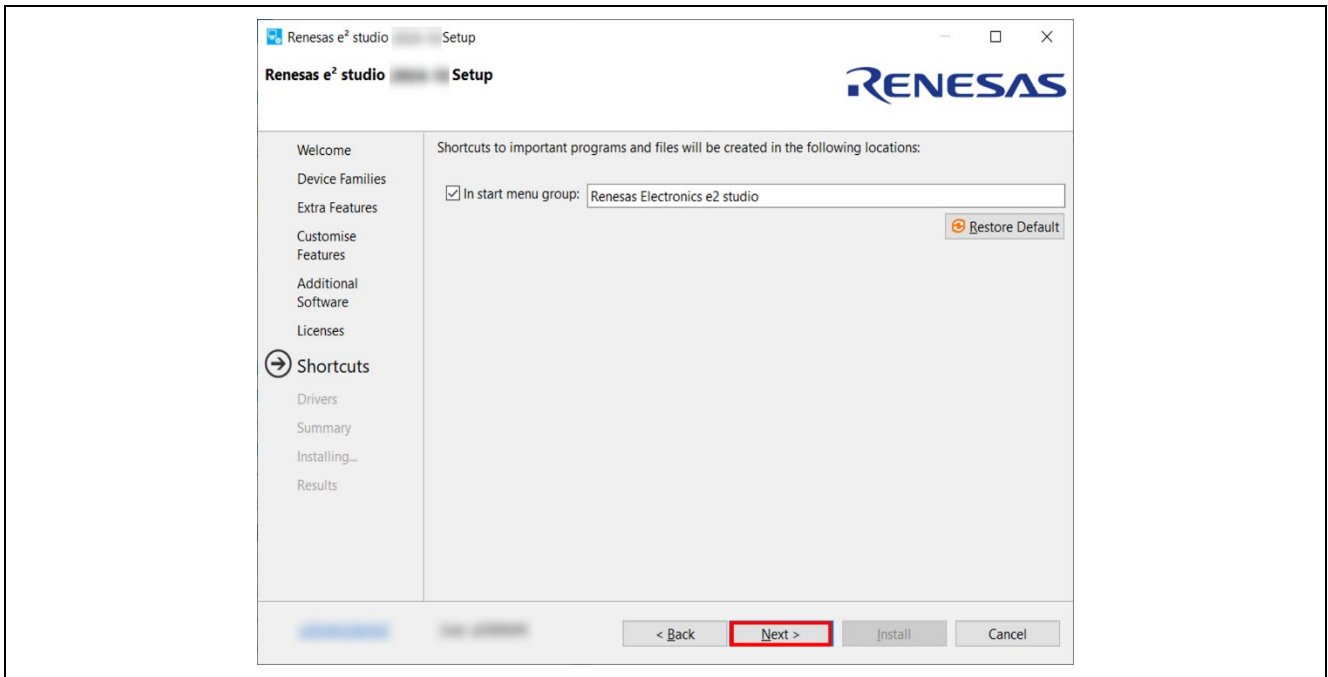


**Figure 9. Installation of e² studio – Licenses**

8. Shortcuts

Select the shortcut name for the Start menu and click the [Next] button to continue.

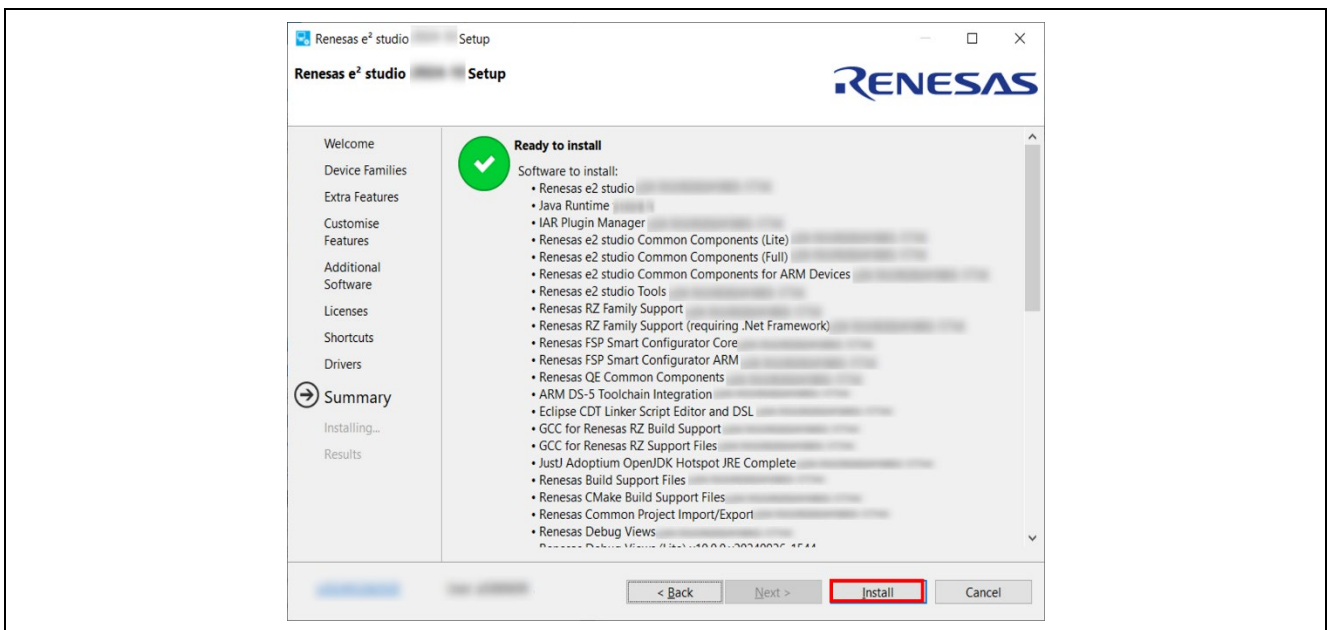
**Note:** If e² studio was installed in another location, it is recommended to rename it to distinguish it from the other e² studio(s).



**Figure 10. Installation of e² studio – Shortcuts**

9. Summary

The components to be installed are shown. Please confirm the contents and click the [Install] button to install the Renesas e² studio IDE.



**Figure 11. Installation of e² studio – Summary**

10. Installing

The installation is performed. Depending on selected items of additional software, new dialog prompts may appear during the installation process. Please see the chapter 2.1.3.3 for more detailed information.

11. Results

Click the **OK** button to complete the installation.

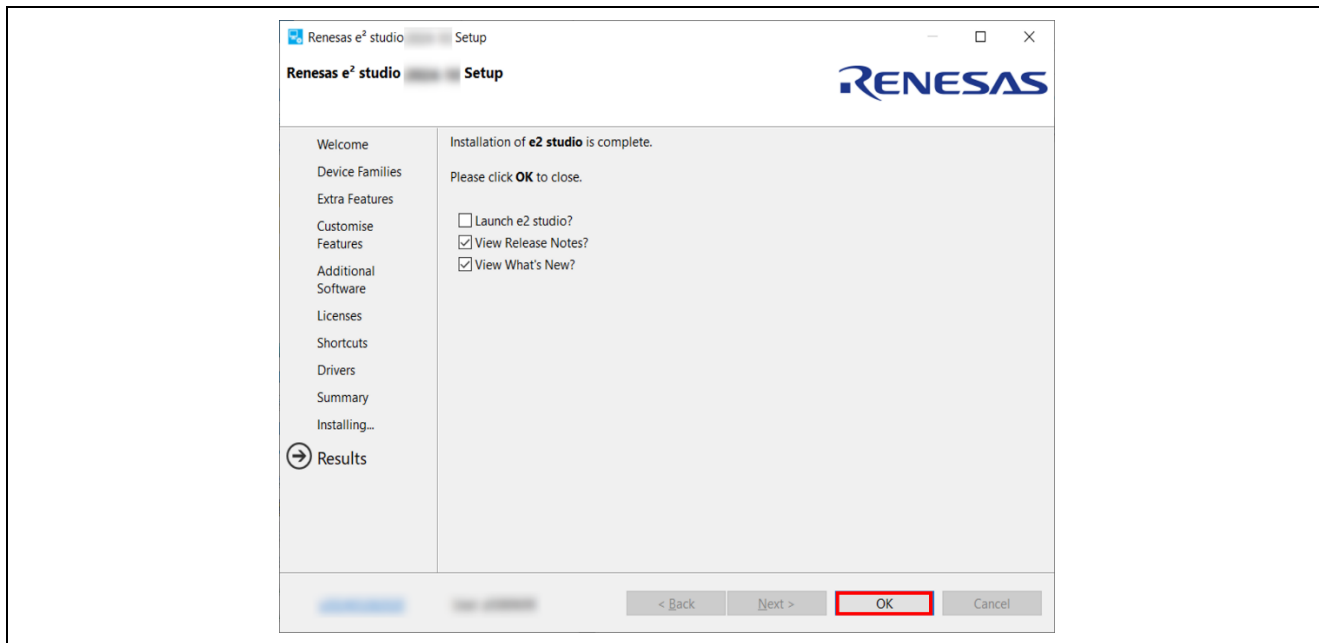


Figure 12. Summary Page

**Installation of Additional Software**

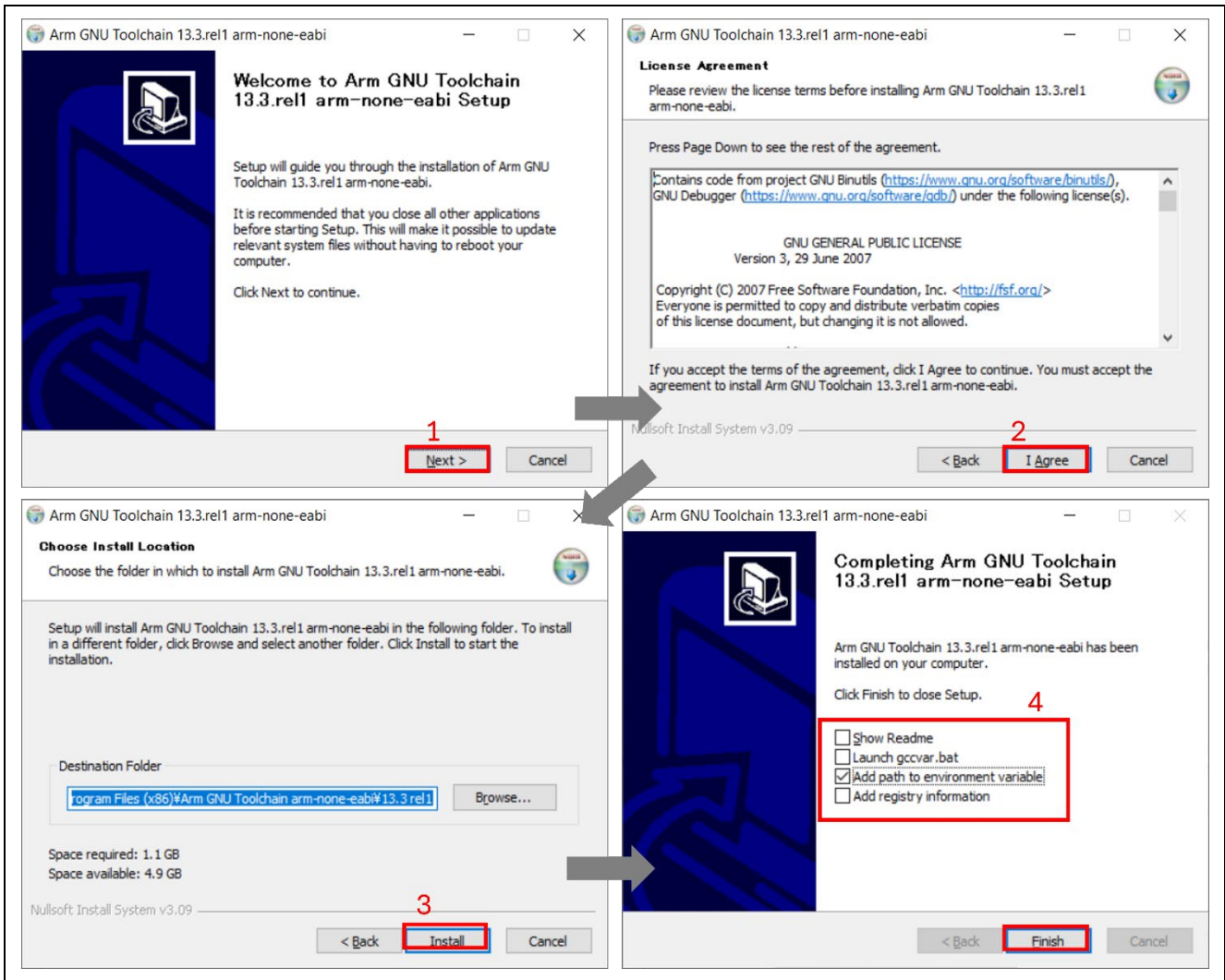
As mentioned in section 2.1.3.2, the additional software listed below is essential for RZ FSP.

- GNU ARM Embedded 13.3-Rel1.
- GNU ARM A-Profile (AArch64 bare-metal) 13.2.Rel1.

In this section, the detailed procedure for installing this tool.

• **GNU ARM Embedded Toolchain 13.3-Rel1**

If it was selected in the Additional Software pane of e<sup>2</sup> studio, you will see the installation wizard for the GNU ARM Embedded Toolchain during the installation process.



**Figure 13. Installation of GNU ARM Embedded Toolchain**

• **GNU ARM A-Profile (AArch64 bare-metal) 13.2.Rel1.**

If it was selected in the Additional Software pane of e<sup>2</sup> studio, you will see the installation wizard for the GNU ARM A-Profile Toolchain during the installation process.

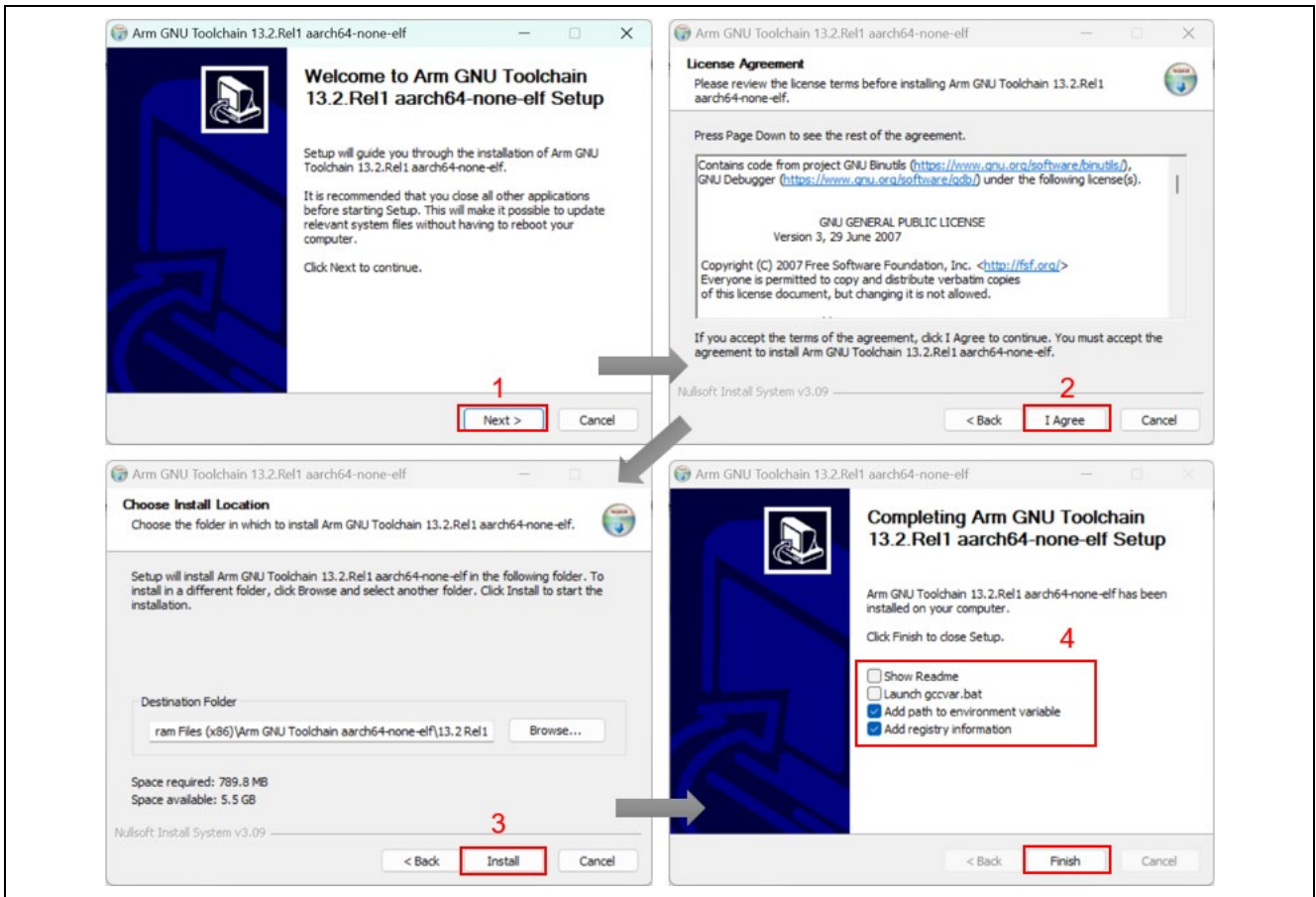


Figure 14. Installation of GNU ARM A-Profile Toolchain

## 2.1.4 e<sup>2</sup> studio installation for Linux PC

This chapter describes how to install the e<sup>2</sup> studio IDE on a Linux PC.

### 2.1.4.1 Download

The following files are required to download before installation.

SEGGER J-Link driver

Please download the driver V8.60 or later from:

<https://www.segger.com/downloads/jlink>

e<sup>2</sup> studio IDE installer

e<sup>2</sup> studio IDE installer package can be downloaded from Renesas website for free. Please check the detailed information from: <https://www.renesas.com/e2studio>.

### 2.1.4.2 Installation

This section describes the procedure for each software installation.

Filename, version number, and the file path are provided for example purposes only.

- **SEGGER J-Link driver**

1. Open a terminal window and enter the commands below.

```
sudo dpkg -i JLink_Linux_V7860_x86_64.deb
```

(If the previous install fails with unmet dependencies, retry it as follows)

```
sudo apt-get -f install
```

```
sudo dpkg -i JLink_Linux_V860_x86_64.deb
```

- **e<sup>2</sup> studio IDE**

1. Run the e<sup>2</sup> studio IDE Installer. (Before running the installer, check the execution permission of the installer.)

```
./e2studio_installer-yyyy-mm_linux_host.run
```

2. **Welcome page**

User needs to select Install Type as shown below. In this material, it is expected that Custom Install is selected. Then, click [Next >] to continue.

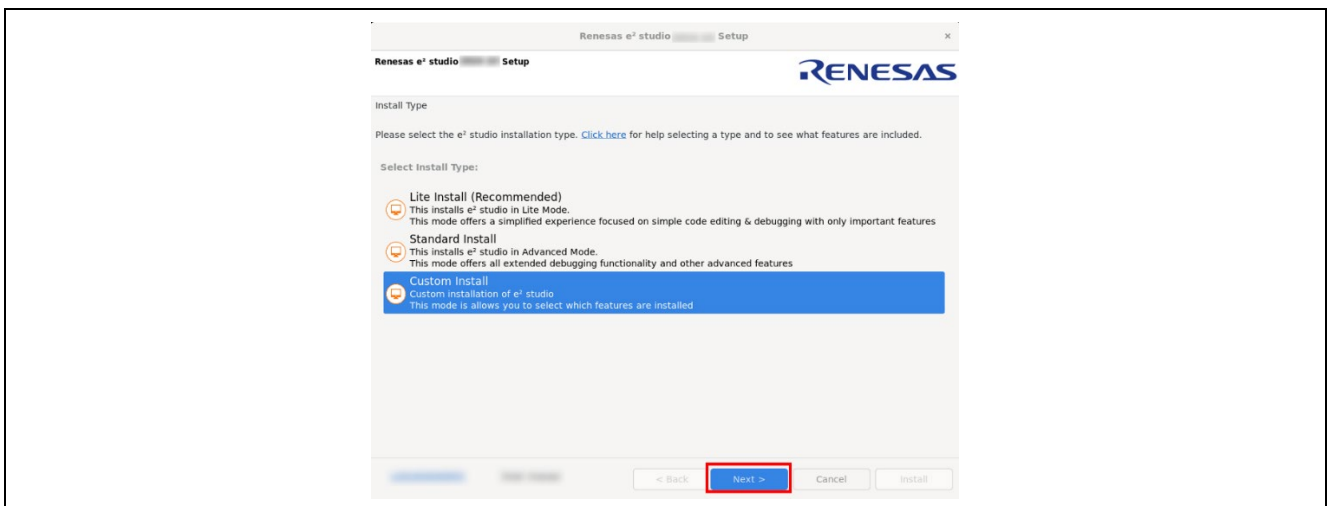


Figure 17. Installation of e<sup>2</sup> studio – Welcome page

### 3. Welcome page (Cont'd)

The user can change the installation folder by clicking [Change...].

Click [Next] to continue.

**Note 1:** If you would like to have multiple versions of e<sup>2</sup> studio, please specify a new folder here.

**Note 2:** Multi-byte characters cannot be used for the e<sup>2</sup> studio installation folder name.

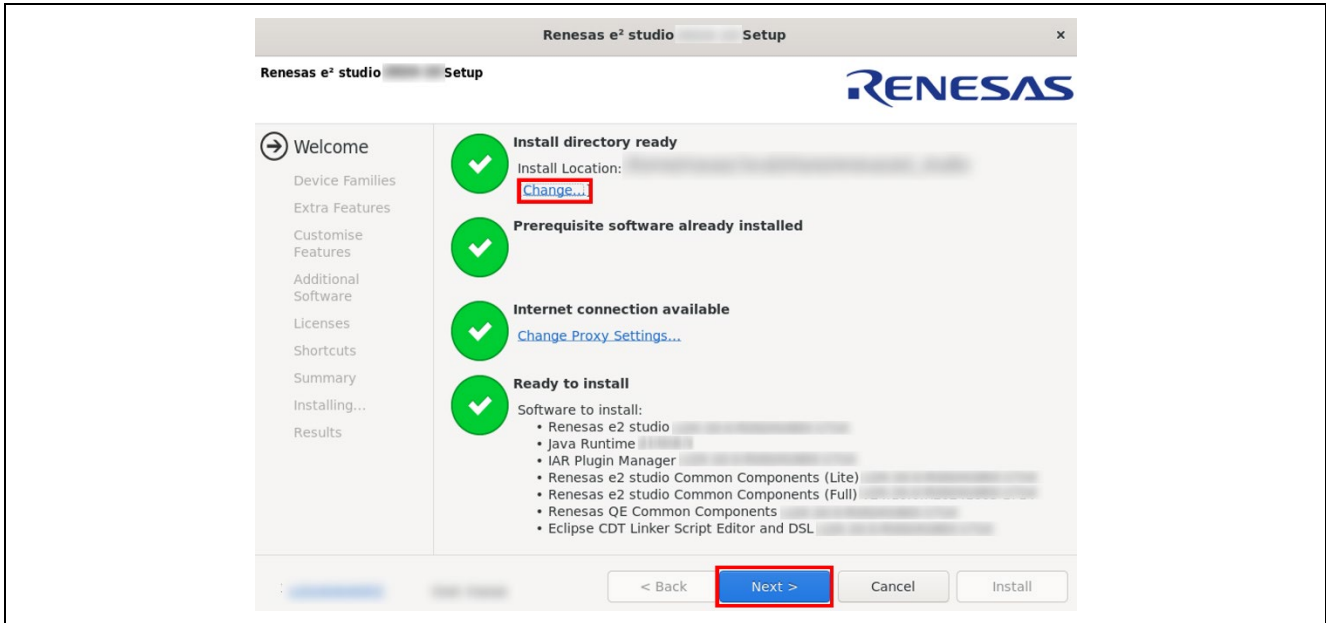


Figure 18. Installation of e<sup>2</sup> studio – Welcome page

#### 4. Device Families

Select Device Families to install. Click the [Next] button to continue.

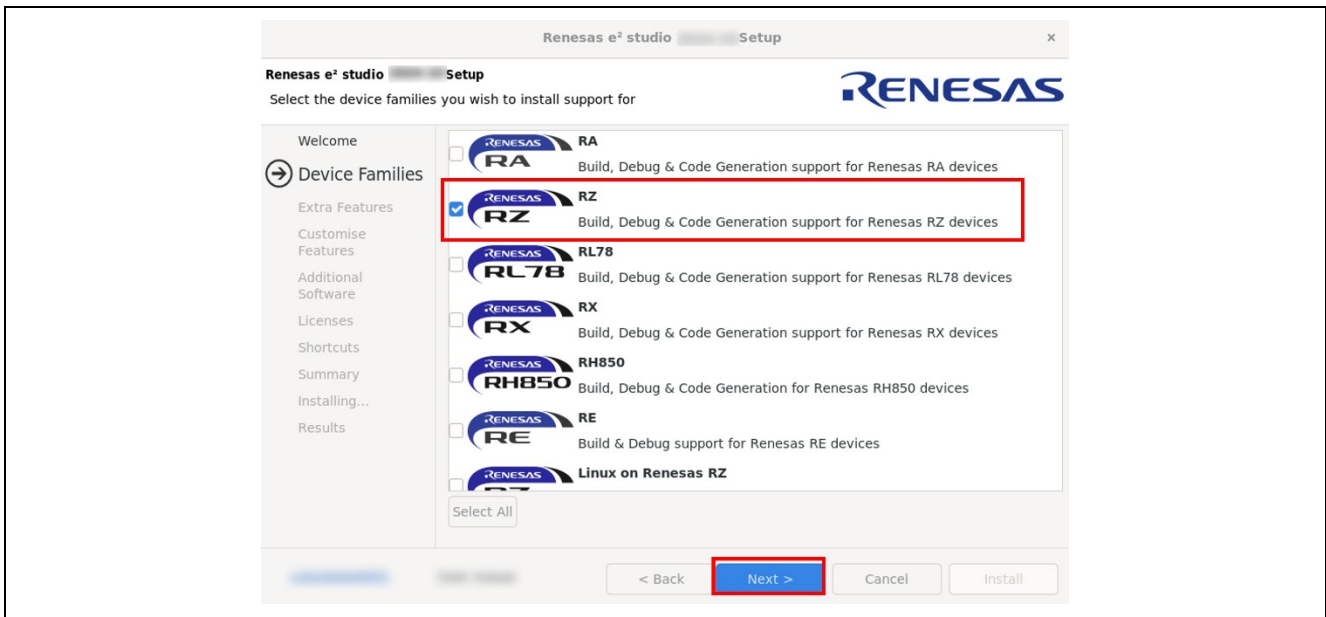


Figure 19. Installation of e² studio – Device Families

#### 5. Extra Features

Select Extra Features (i.e., Language packs, SVN & Git support, RTOS support...) to be installed. For non-English language users, please select Language packs at this step if needed. Click the [Next] button to continue.

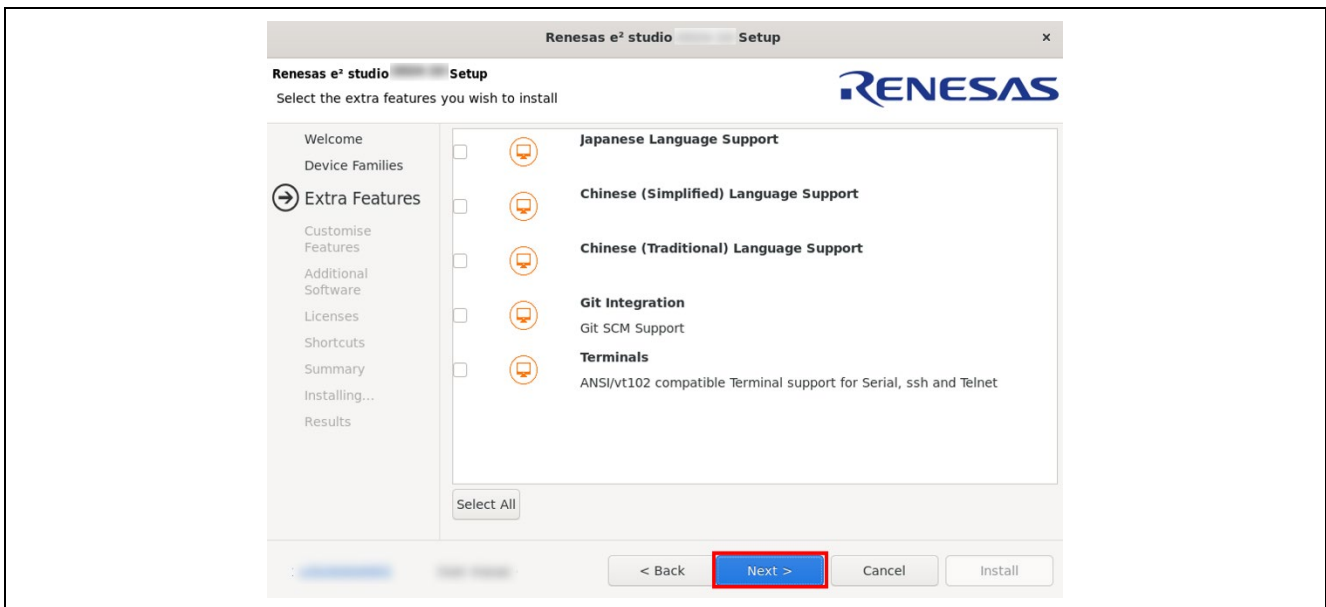


Figure 20. Installation of e² studio – Extra Features

### 6. Customize Features

Select the components to install and click the [Next] button to continue.

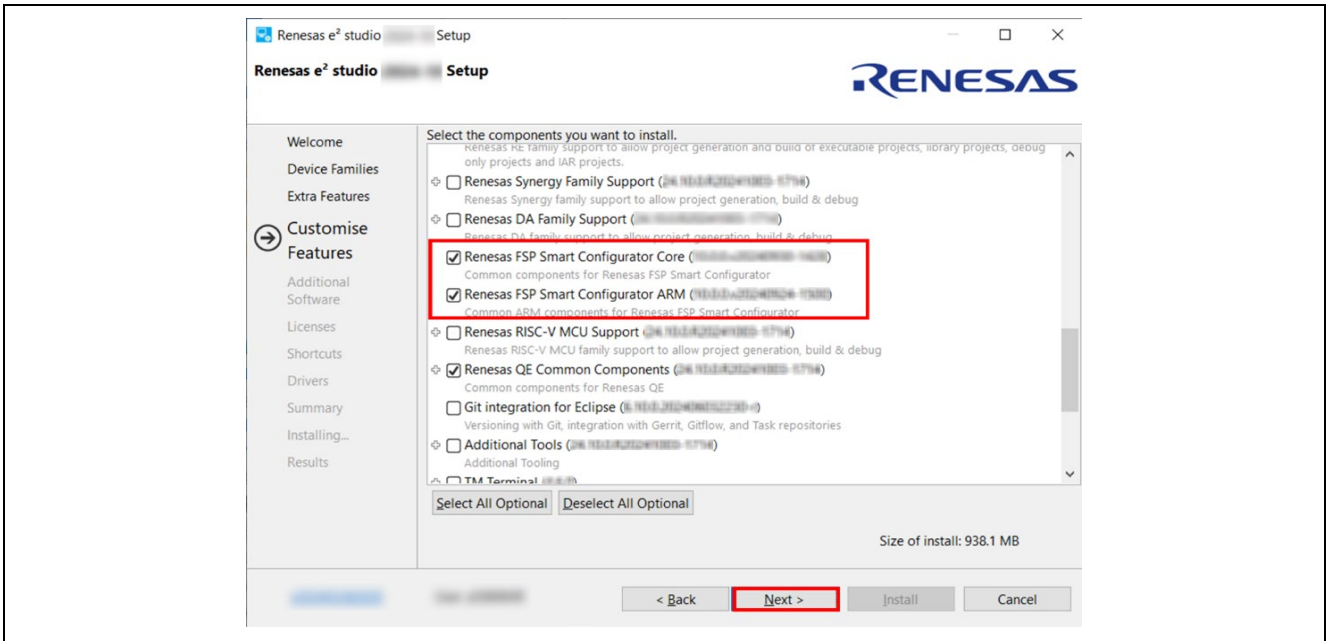


Figure 21. Installation of e2 studio – Features

### 7. Additional Software

Select additional software (i.e., compilers, utilities, QE...) to be installed. Be sure to select the “GNU ARM Embedded 13.3-Rel1” and “GCC ARM A-Profile (AArch64 bare-metal) 13.2. Rel1” and click [Next >] to continue.

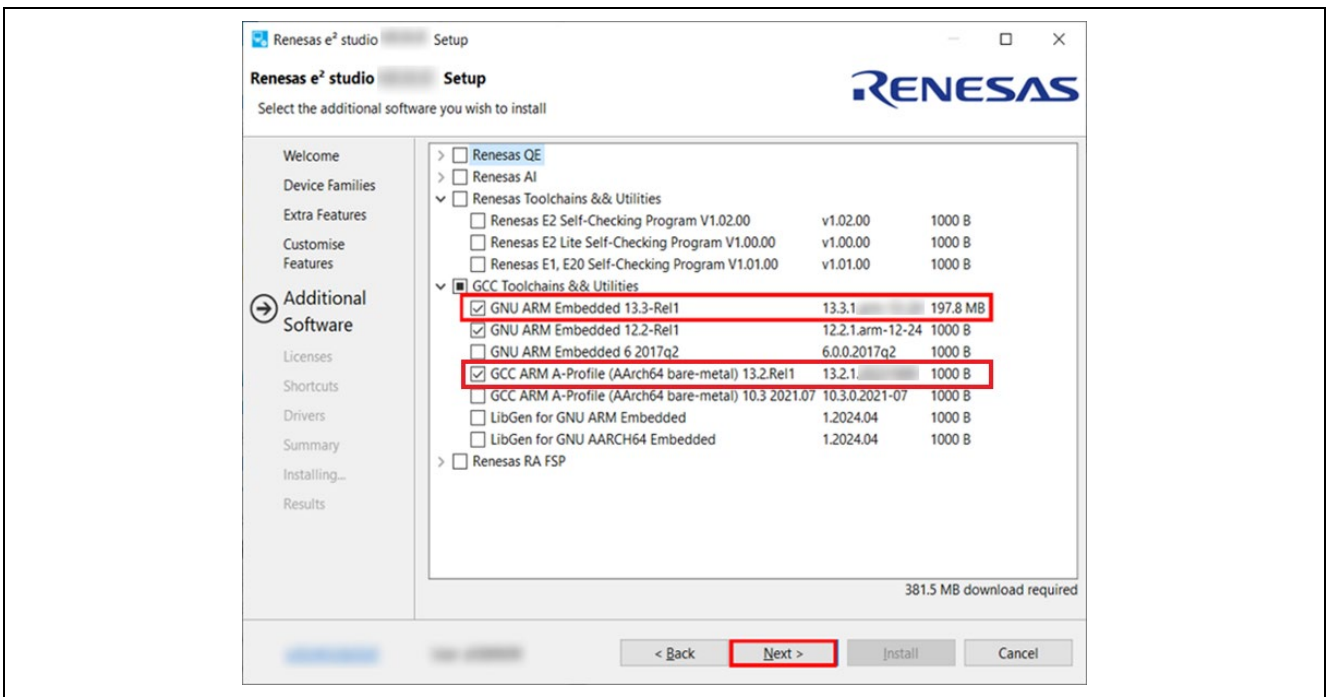


Figure 22. Installation of e2 studio – Additional Software

### 8. License Agreement

Read and accept the software license agreement. Click the [Next] button. Please note that the user must accept the license agreement; installation cannot continue.

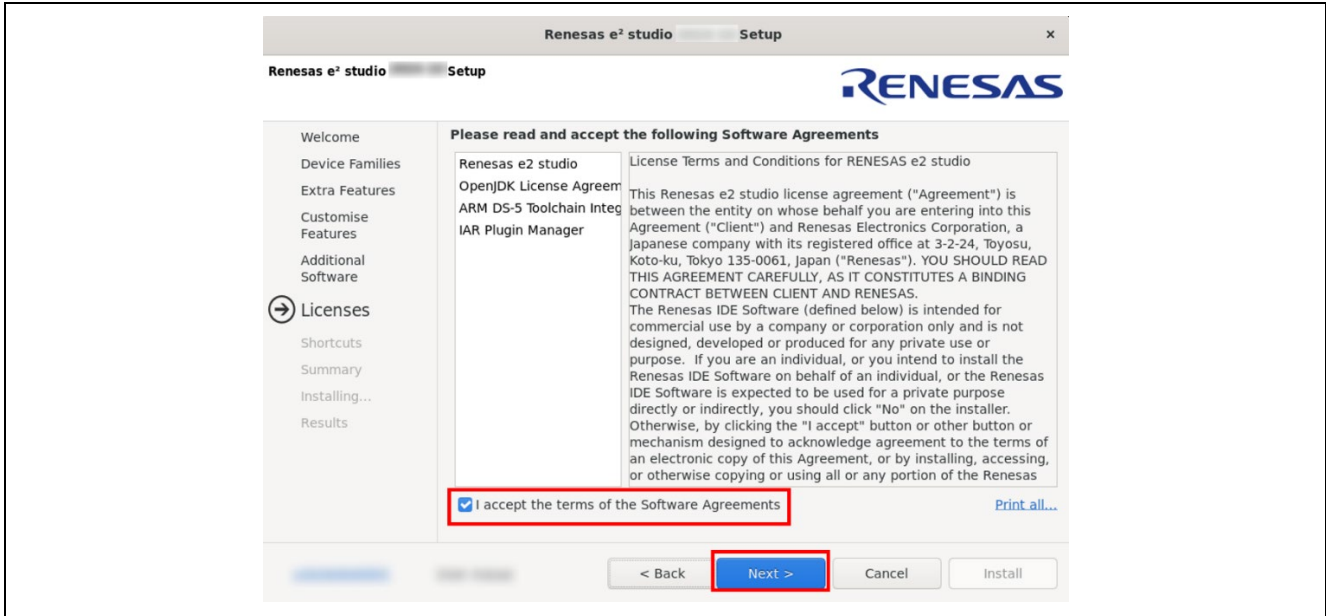


Figure 23. Installation of e² studio – Licenses

### 9. Shortcuts

Select the shortcut name for the Start menu and click the [Next] button to continue.

**Note:** If e² studio was installed in another location, it is recommended to rename it to distinguish it from the other e² studio(s).

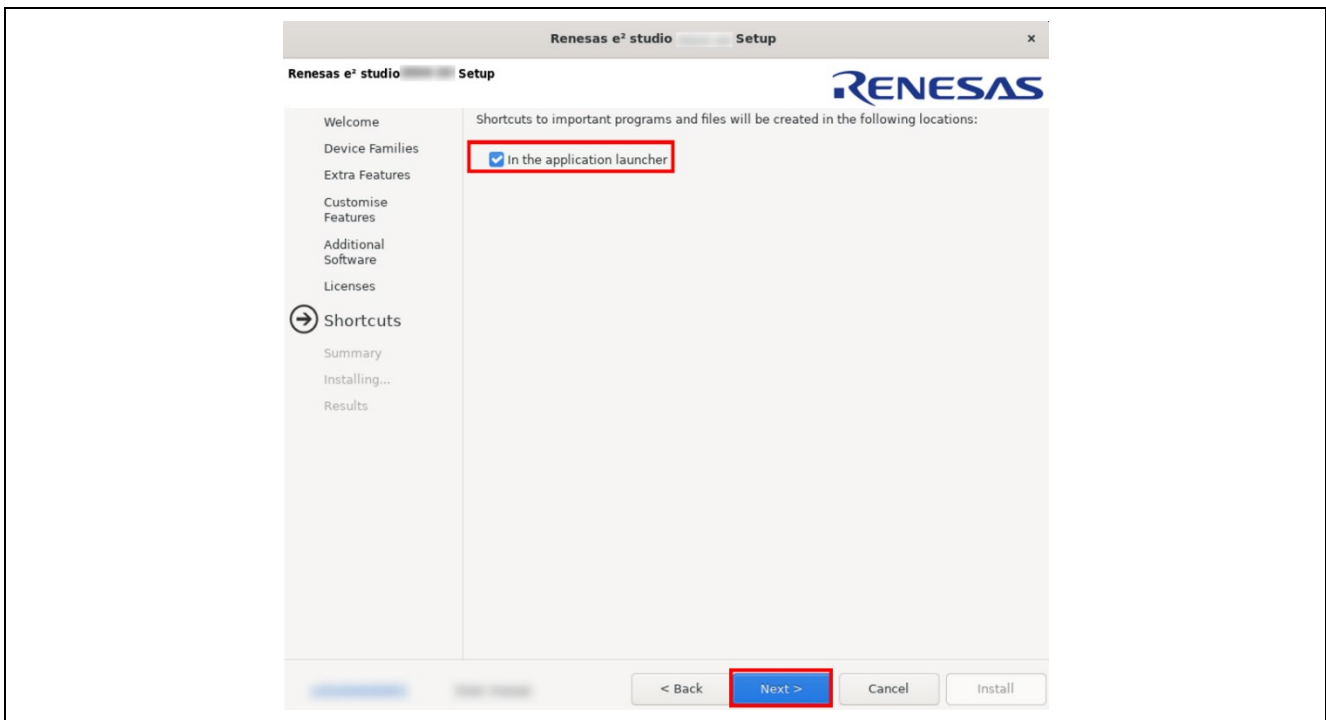


Figure 24. Installation of e² studio – Shortcuts

## 10. Summary

The components to be installed are shown below. Please confirm the contents and click the [Install] button to install the Renesas e<sup>2</sup> studio IDE.

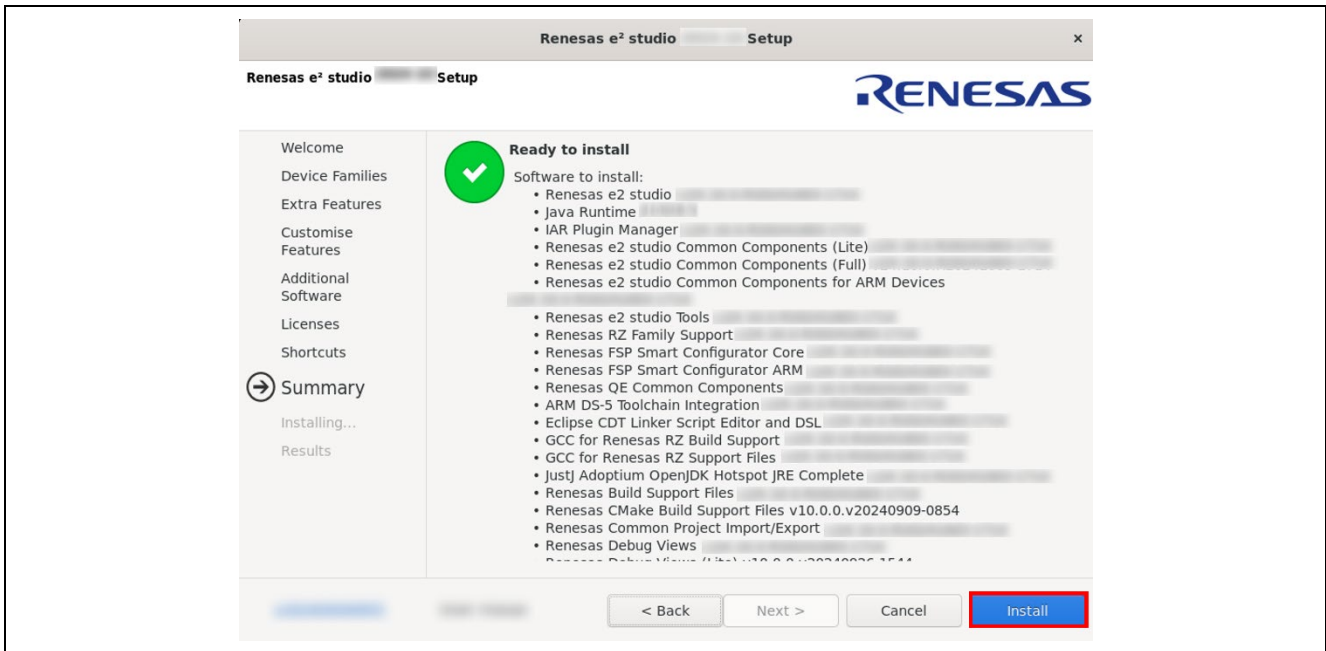


Figure 25. Installation of e<sup>2</sup> studio – Summary

## 11. Installing

The installation is performed. Depending on the selected items of additional software, new dialog prompts may appear during the installation process.

## 12. Results

Click the **OK** button to complete the installation.

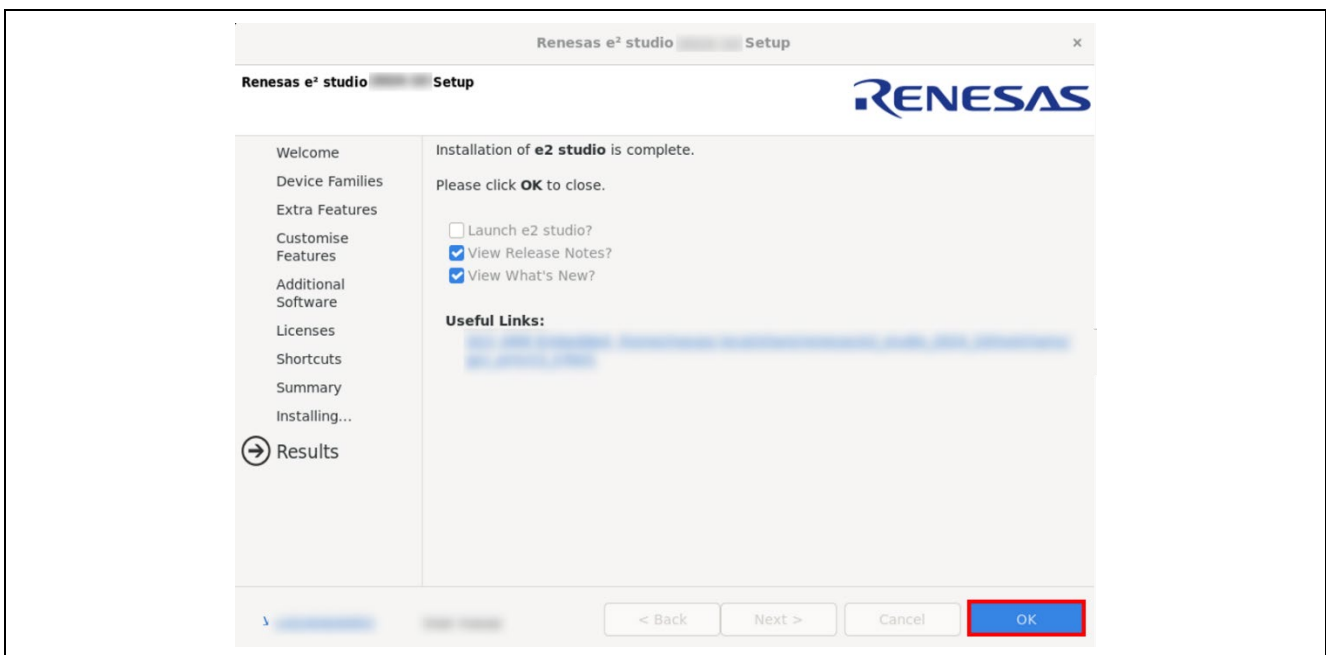


Figure 26. Summary Page

## 2.2 FSP Setup

In this section, 3 ways of FSP installation are described.

### 2.2.1 Installation of FSP using Platform Installer

This section describes how to install FSP using Platform Installer `setup_rzfsp_v4_1_0_e2s_v2026-04.1.exe` showcased [here](#).

1. Double-click `setup_rzfsp_v4_1_0_e2s_v2026-04.1.exe`, select either **[Quick Install]** or **[Custom Install]**, and click **[Next >]** when the installation wizard is shown. When you choose **[Quick Install]**, you can jump to 6. Licenses.

**Note:**

If e<sup>2</sup> studio was installed on your PC, the option to upgrade the existing version or install e<sup>2</sup> studio to a different location will be displayed.

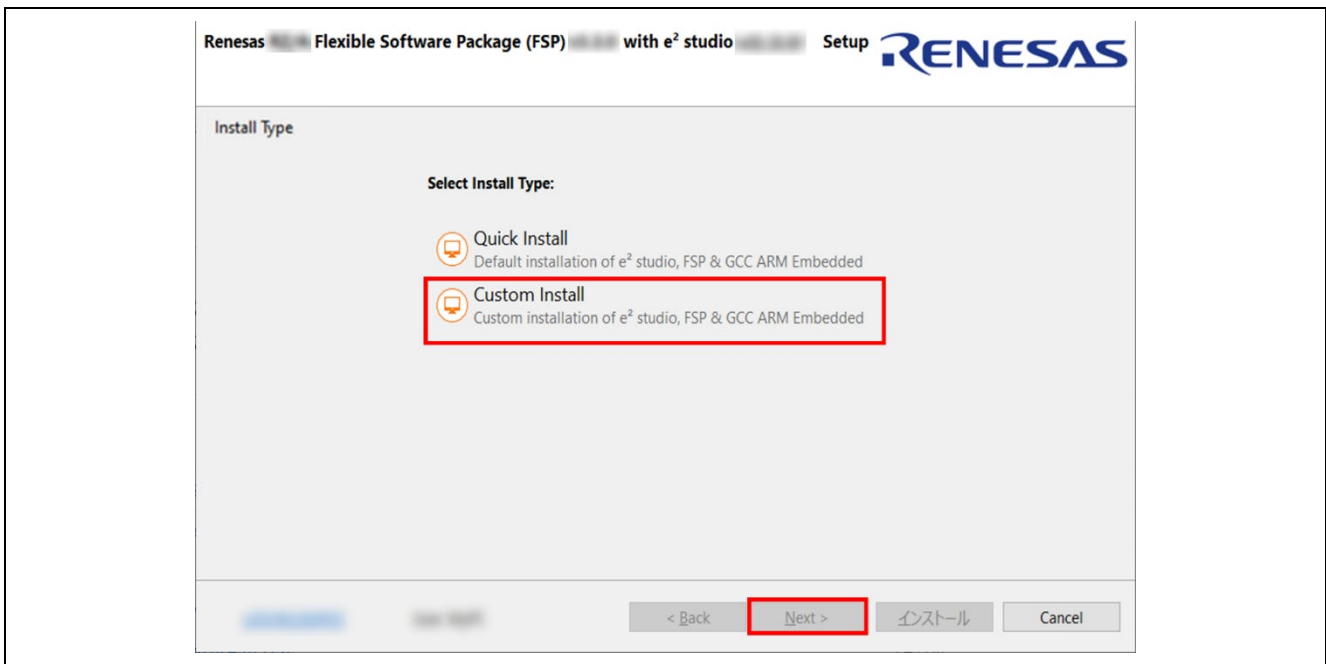


Figure 27. FSP Platform Installation Wizard

2. Welcome page

The user can change the installation folder by clicking [Change...]. Click [Next] to continue.

**Note:**

1. If you would like to have multiple versions of e<sup>2</sup> studio, please specify a new folder here.
2. Multi-byte characters cannot be used for the e<sup>2</sup> studio installation folder name.

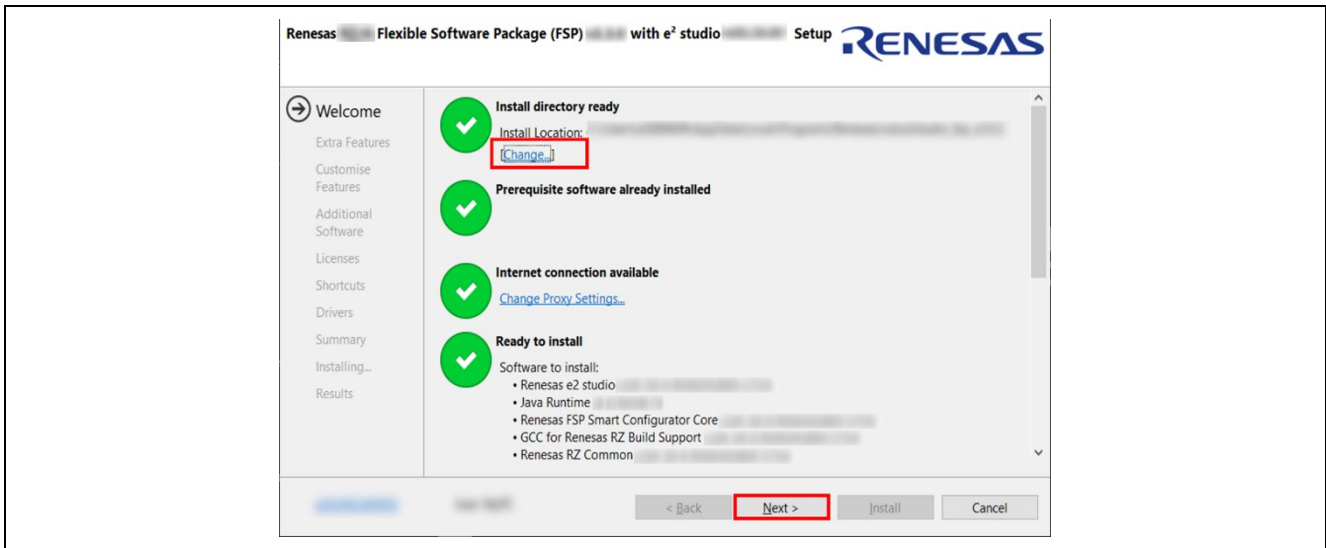


Figure 28. FSP Platform Installer – Welcome page

3. Extra Features

Select Extra Features (i.e., Language packs, SVN & Git support, RTOS support...) to be installed. For non-English language users, please select Language packs at this step if needed. Then, click the [Next] button to continue.

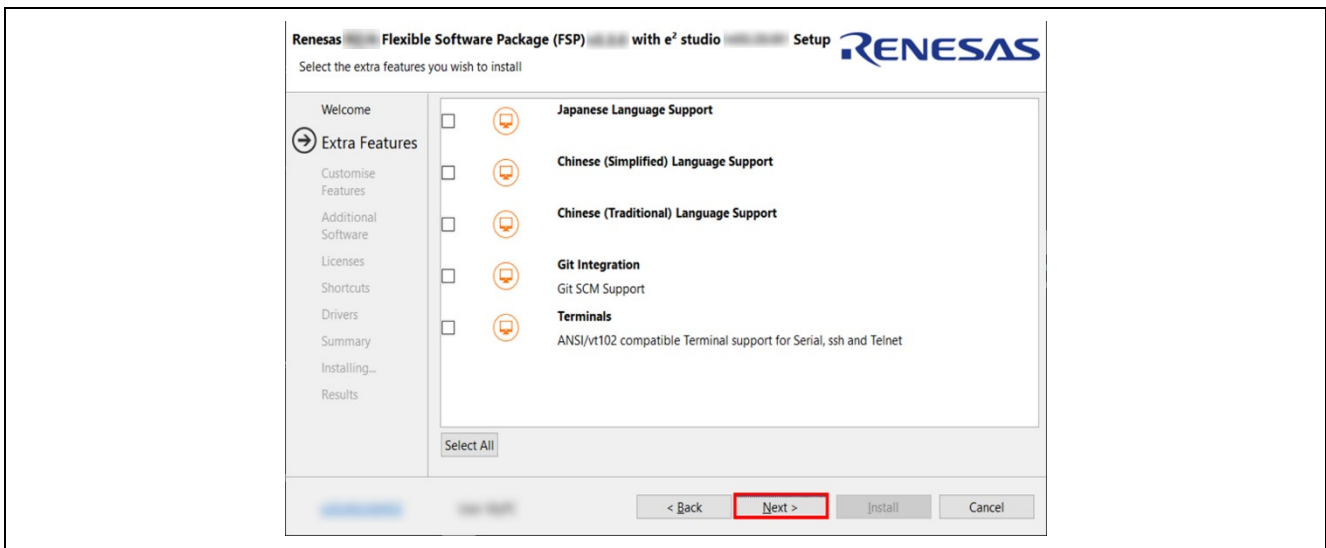


Figure 29. FSP Platform Installer – Extra Features

4. Customize Features

Essential features have already been selected. If you would like to install additional features, please check those and then click [Next >].

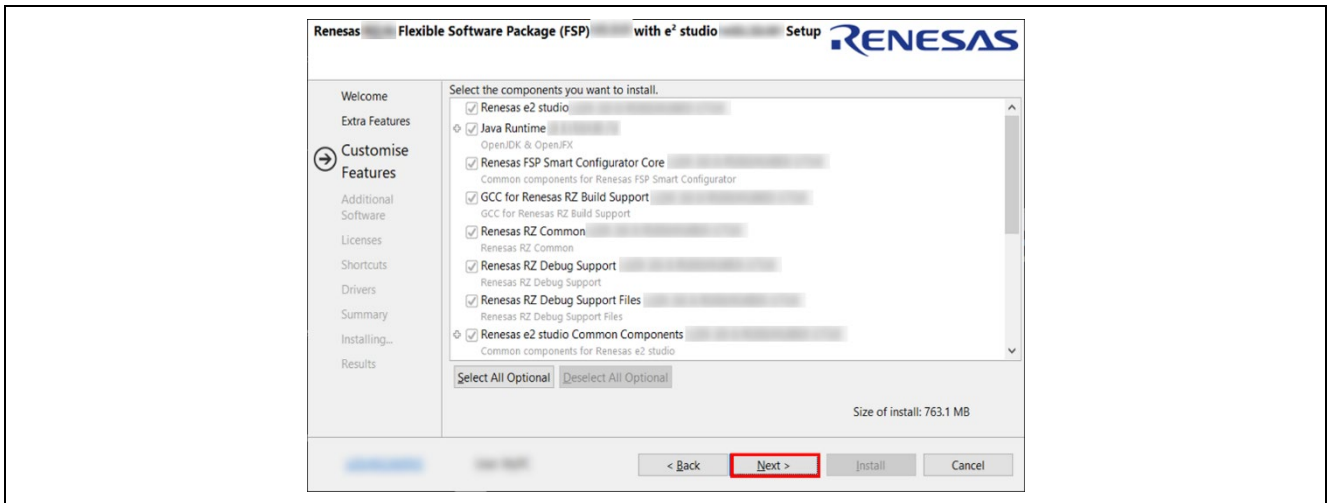


Figure 30. FSP Platform Installer – Features

5. Additional Software

All the software is selected by default. Click [Next >].

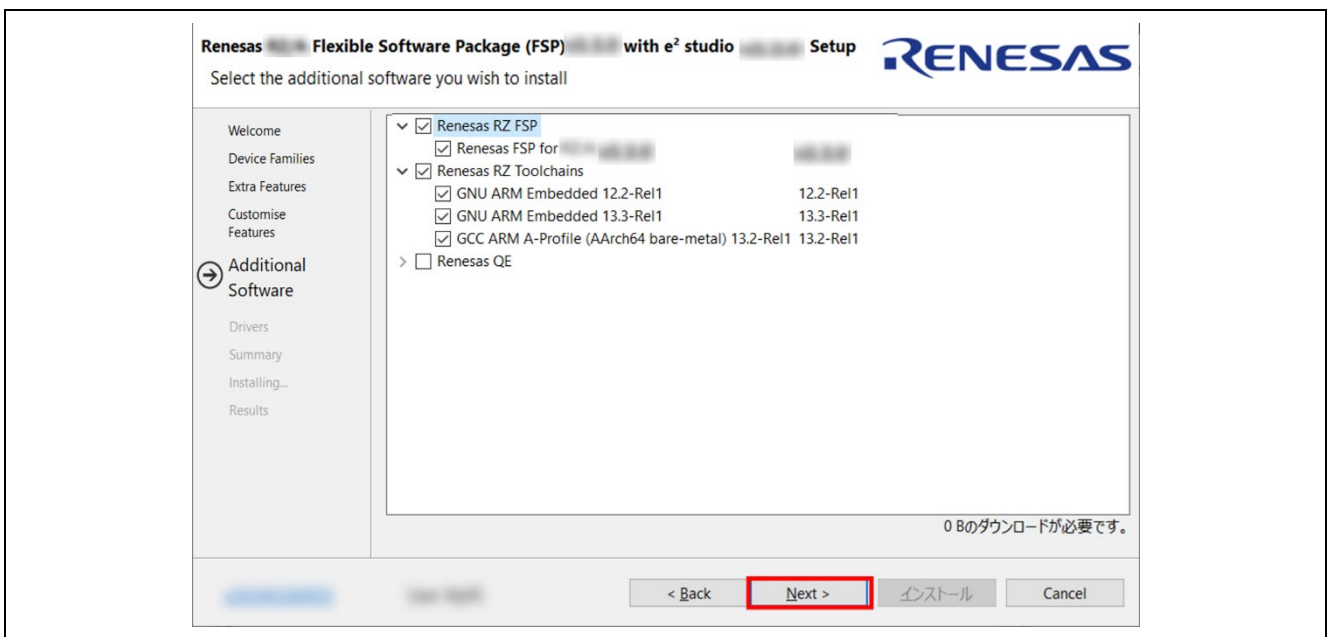


Figure 31. FSP Platform Installer – Additional Software

6. Licenses

Please read and accept the Software License Agreements to be listed and click [Next >].

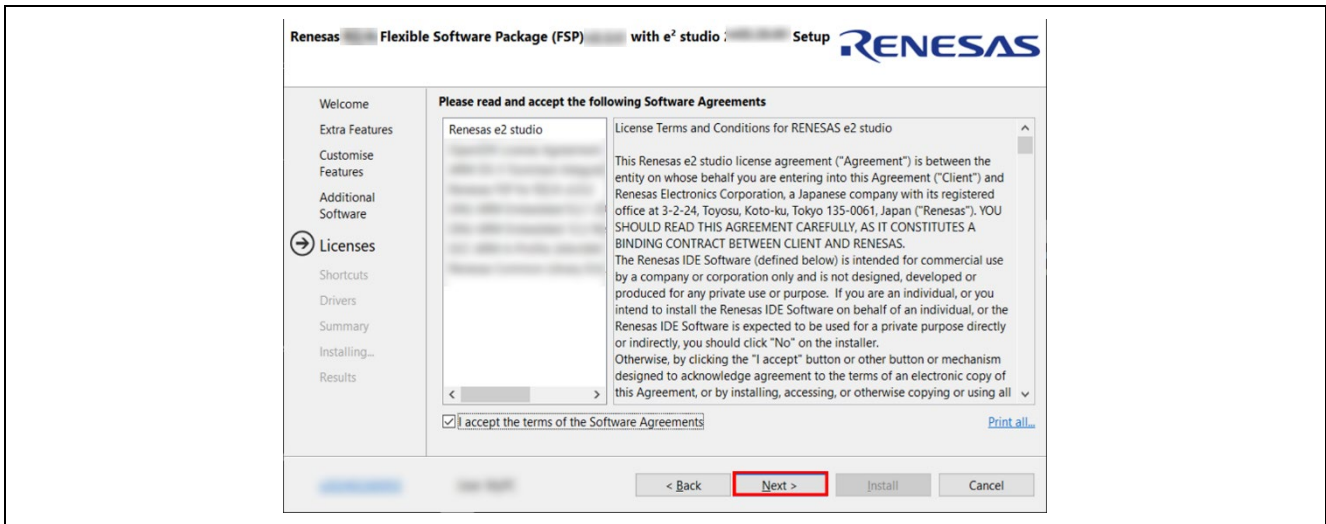


Figure 32. FSP Platform Installer – Licenses

## 7. Shortcuts

Select the shortcut name for the Start menu and click the [Next] button to continue.

**Note:** If e<sup>2</sup> studio was installed in another location, it is recommended to rename it to distinguish it from the other e<sup>2</sup> studio(s).

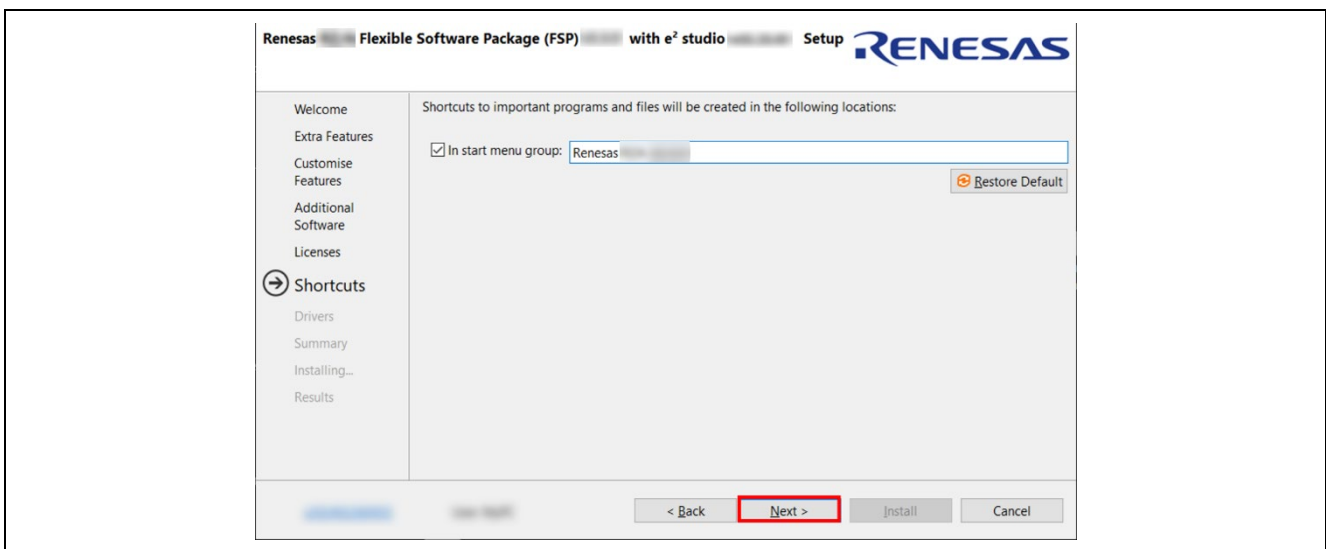


Figure 33. FSP Platform Installer – Shortcuts

## 8. Summary

The components to be installed are shown. Please confirm the contents and click the [Install] button to install the Renesas e<sup>2</sup> studio IDE.

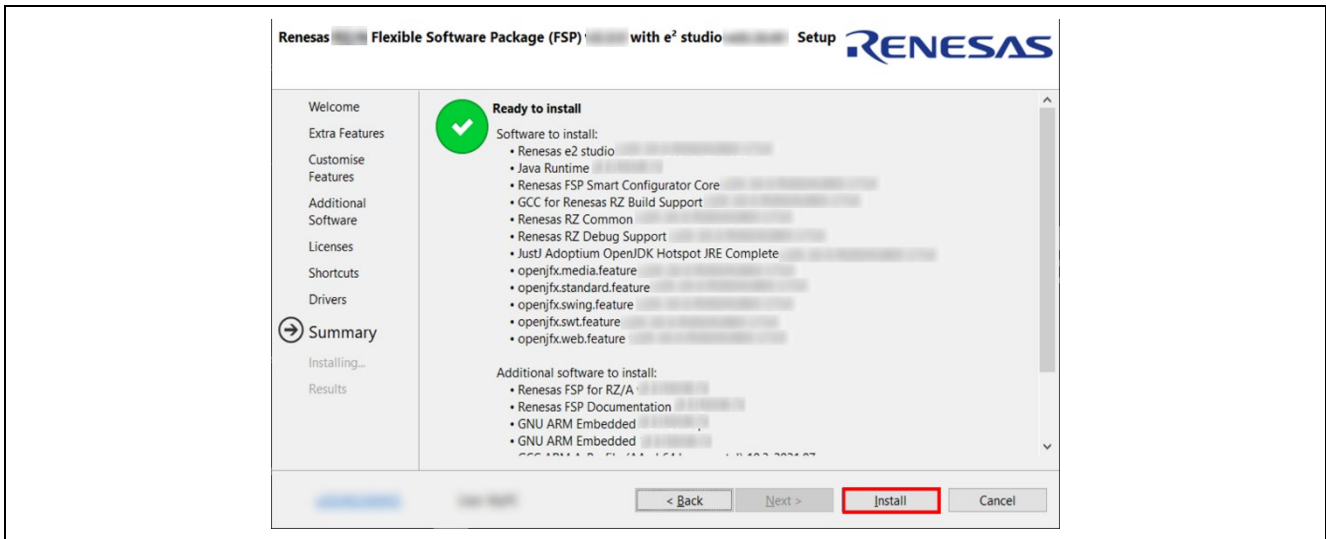


Figure 34. FSP Platform Installer – Summary

10. Installing...

The installation is performed. Depending on the selected items of additional software, new dialog prompts may appear during the installation process. At that time, please follow the instructions the installer indicates.

11. Results

If the installation is successful, you should see the following information.

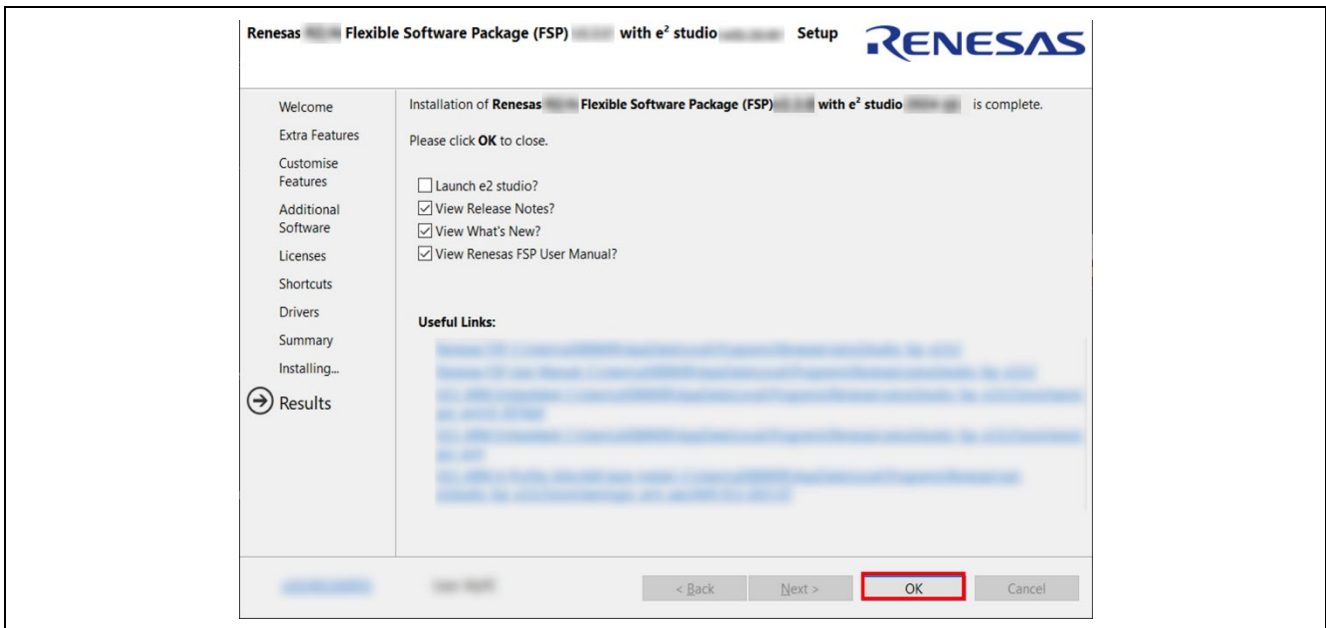


Figure 35. Installation Results of FSP Platform Installer

### 2.2.2 Installation of FSP using Package Installer

Package Installer **RZ\_FSP\_Packs\_v4.1.0.exe** is showcased [here](#). Please note that it is for the Windows Host PC only.

1. Quit e<sup>2</sup> studio.
2. Invoke **RZ\_FSP\_Packs\_v4.1.0.exe**.
3. Click [Next >] to start the installation.

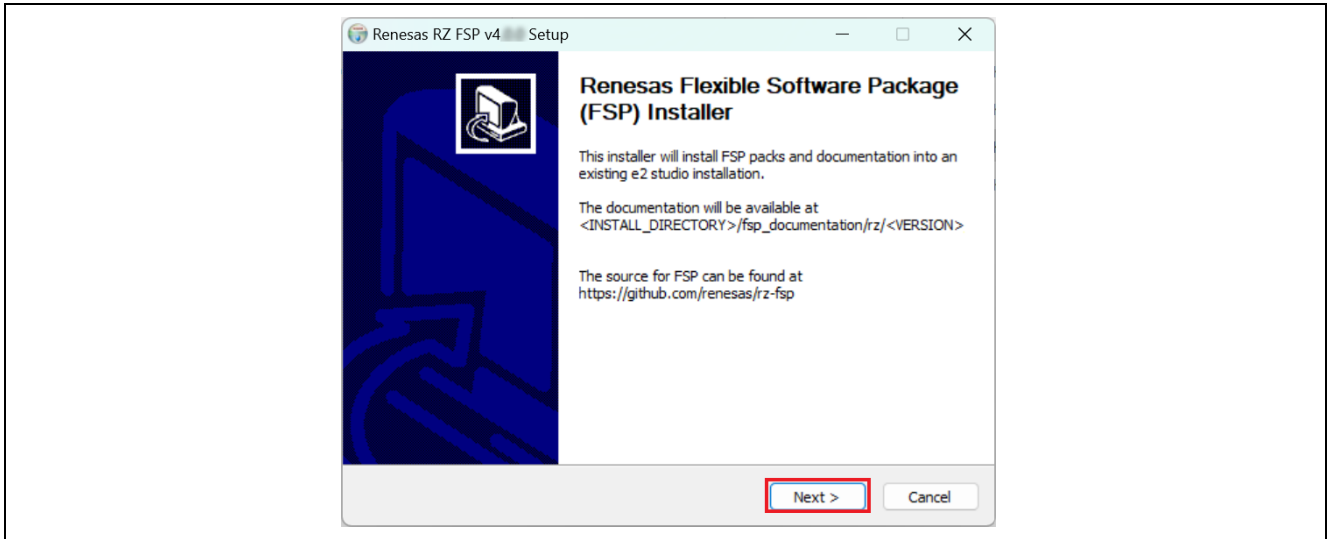


Figure 36. FSP Package Installer

4. See the license term and click the [**I Agree**] button if it is acceptable.

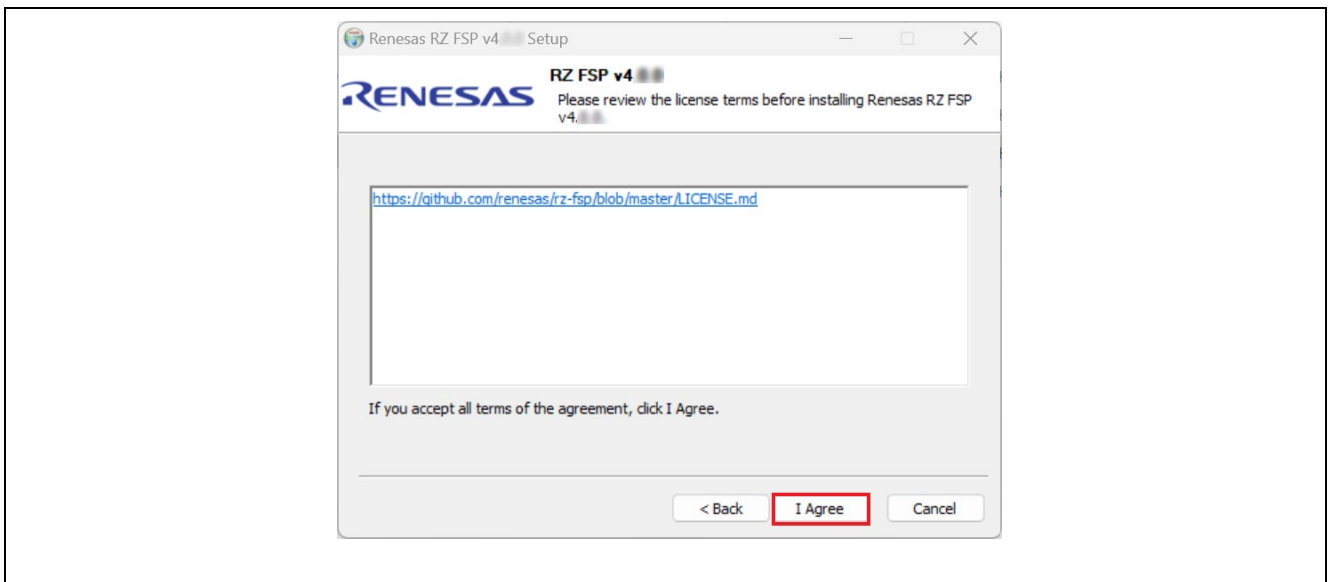
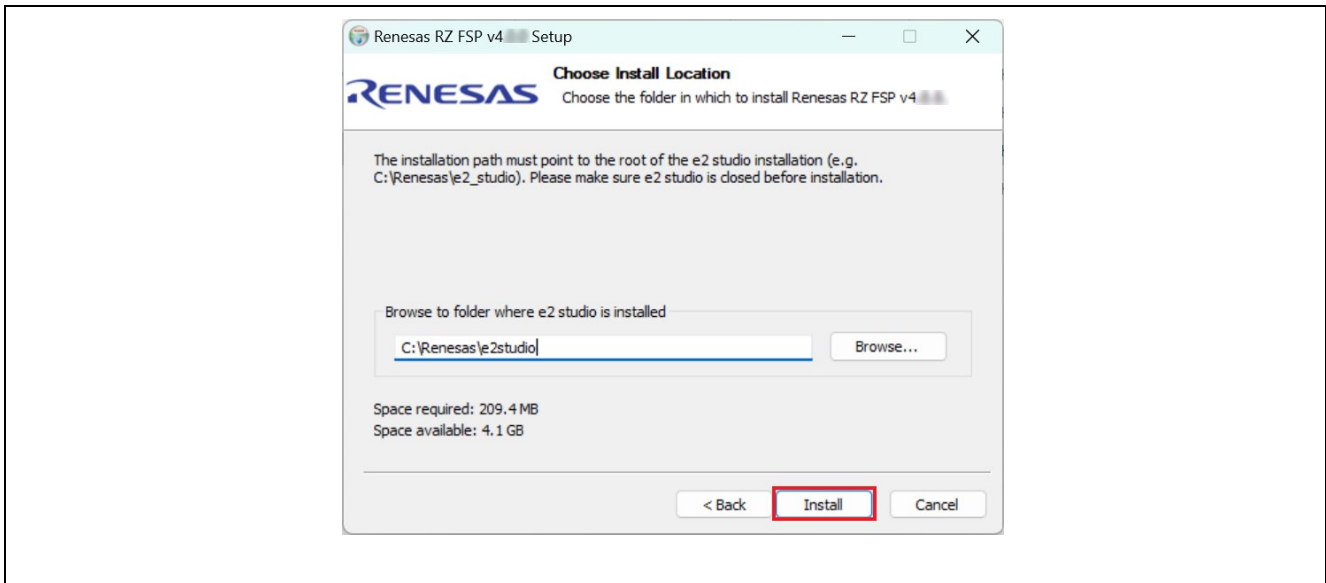


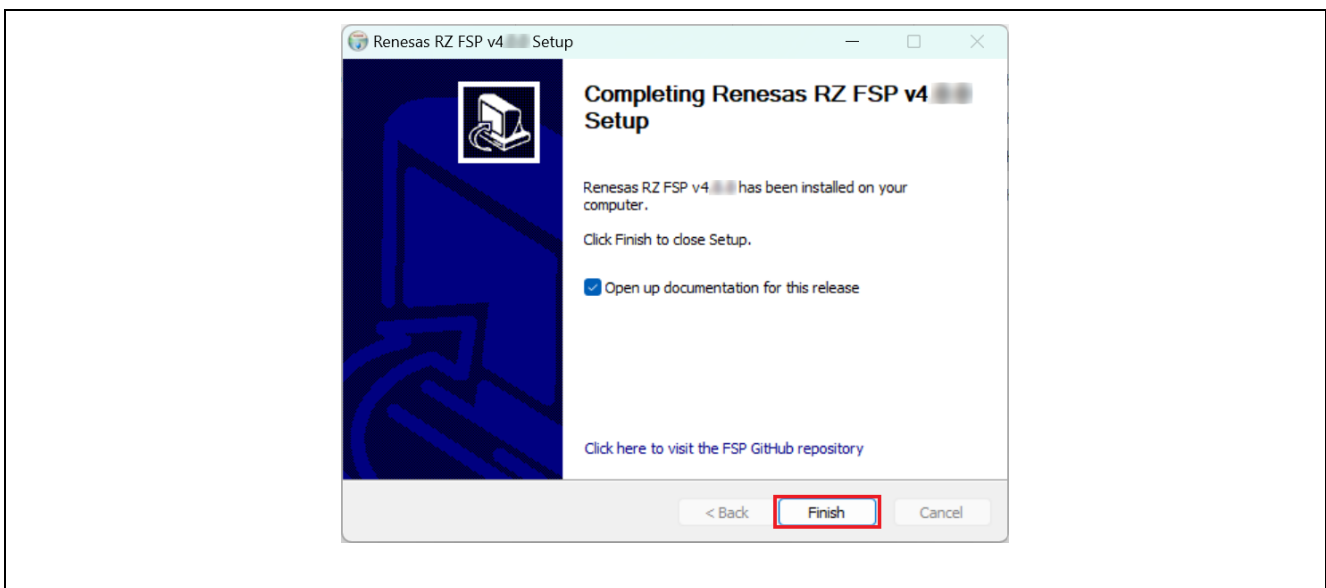
Figure 37. FSP License Term

5. Specify the e<sup>2</sup> studio installation folder (e.g., C:\Renesas\e<sup>2</sup> studio) and click the **[Install]** button.



**Figure 38. Browse to the folder where e<sup>2</sup> studio is installed**

6. Click the **[Finish]** button to complete the installation.



**Figure 39. Completion of FSP Installation**

If the box **Open up documentation for this release** is checked at that time, FSP documentation for the installed version of FSP should be opened.

### 2.2.3 Installation of FSP Packs using a Package Zip file

No Package installer is available for the Linux Host PC, and therefore, you need to install FSP Packs with **RZ\_FSP\_Packs\_v4.1.0.zip**. This section describes how to install it. Please note that the same installation procedure is valid for the Windows Host PC.

1. Download **RZ\_FSP\_Packs\_v4.1.0.zip** from [here](#).
2. Extract the zip file to the e<sup>2</sup> studio installation directory. If it's successfully extracted, **rz\_fsp/packs** should be placed at **<e2 studio installation directory>/Internal/projectgen**.

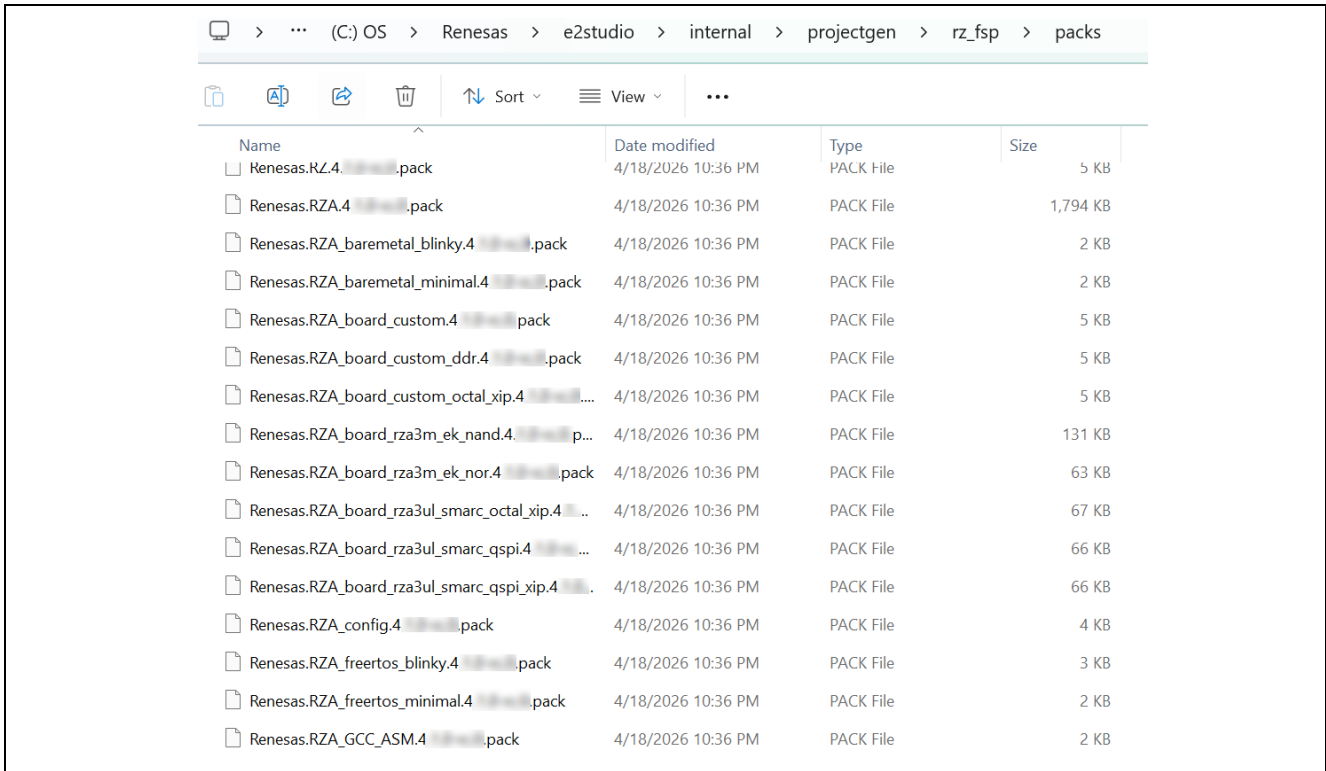


Figure 40. FSP Packs on e<sup>2</sup> studio installation directory

3. At the 1st invocation of e<sup>2</sup> studio after the extraction, FSP should be automatically installed.
4. You can check if the installation is successful by following the procedure below:
  - Click Help > CMSIS Packs Management > Renesas RZ

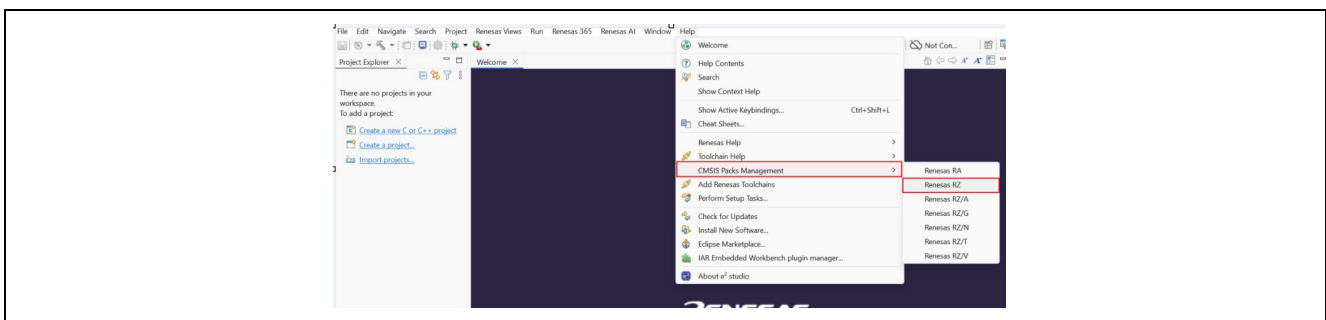


Figure 41. CMSIS Packs Management (1)

- If FSP is successfully installed, 4.1.0 should be listed under FSP as shown below:

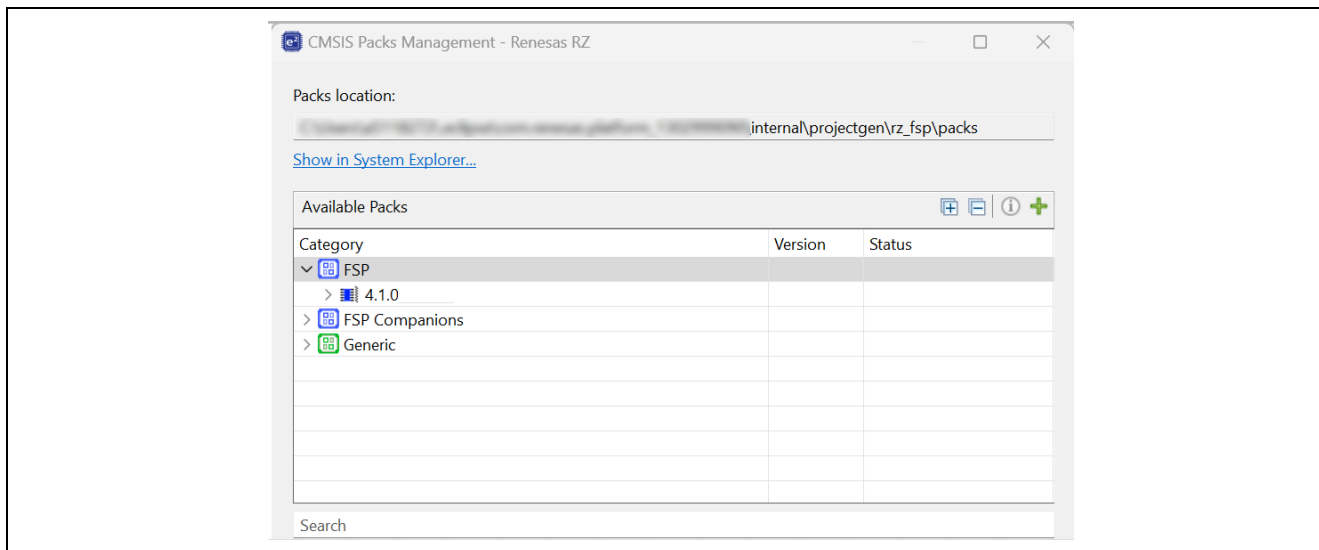


Figure 42. CMSIS Packs Management (2)

### 3. Set up the Evaluation Board

#### 3.1 Obtaining an Evaluation Board

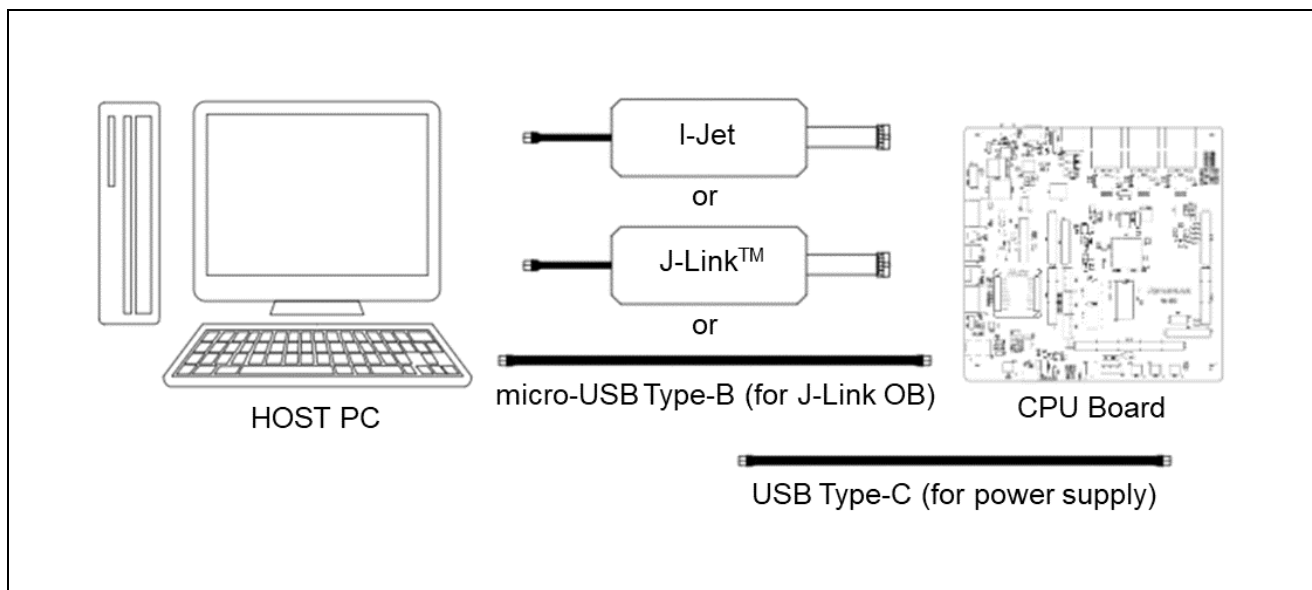
To develop applications with RZ/T2 FSP and RZ/N2 FSP, start with the Evaluation Board and the Renesas Starter Kit+ (RSK+).

The Evaluation Board and RSK+ for RZ/T2 and RZ/N2 CPU Board are designed to seamlessly integrate with the e<sup>2</sup> studio.

Ordering information, User's Manuals, and other related documents for boards are available. Please contact Renesas to get them.

#### 3.2 System Configuration

Below is an example of a typical system configuration of an evaluation board.



**Figure 43. System Configuration Example – with Evaluation Board**

For the details, please refer to the related document, “1.3.1 Evaluation Board User’s Manual”.

### 3.3 Supported Emulator

#### 3.3.1 SEGGER J-Link

SEGGER J-Link can be used on Renesas e<sup>2</sup> studio only for debugging on RZ/T2 and RZ/N2 devices.

Renesas e<sup>2</sup> studio supports the following emulators.

- J-Link EDU V11 and later
- J-Link BASE V11 and later
- J-Link PLUS V11 and later
- J-Link WiFi V1 and later
- J-Link ULTRA+ V5 and later
- J-Link PRO V5 and later
- J-Link OB-S124 V1.00

Renesas has tested debugging RZ/T2 and RZ/N2 devices with J-Link BASE V11 and J-Link OB-S124.

For the details on SEGGER J-Link, please see the SEGGER website.

Debugging the FSP Project was verified with the following software environment.

**Table 1. Verified Operating Environment**

Series	Device	FSP version	e <sup>2</sup> studio version	J-Link Software version
RZ/T	RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H	RZ FSP v4.1.0	2026-04.1	V8.60
RZ/N	RZ/N2L, RZ/N2H			

Regarding how to update J-Link firmware, please confirm the procedure described in the following link on the Renesas Knowledge Base website.

<https://en-support.renesas.com/knowledgeBase/20736714>

#### 3.3.2 IAR I-Jet

IAR I-jet can be used on IAR EWARM only for debugging on RZ/T2 and RZ/N2 devices.

For the details on I-jet, please see the IAR Systems website.

### 3.4 RZ/T Series Board Setup

#### 3.4.1 RSK+RZT2M

##### 3.4.1.1 Boot Mode

The operation mode settings for the RSK+RZT2M board are as follows.

**Note:** This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, please refer to the manual of the RSK boards listed in the chapter 1.3.1. For [the sample codes available on Renesas website](#), please refer to the documentation included with each code and implement the appropriate board settings, respectively.

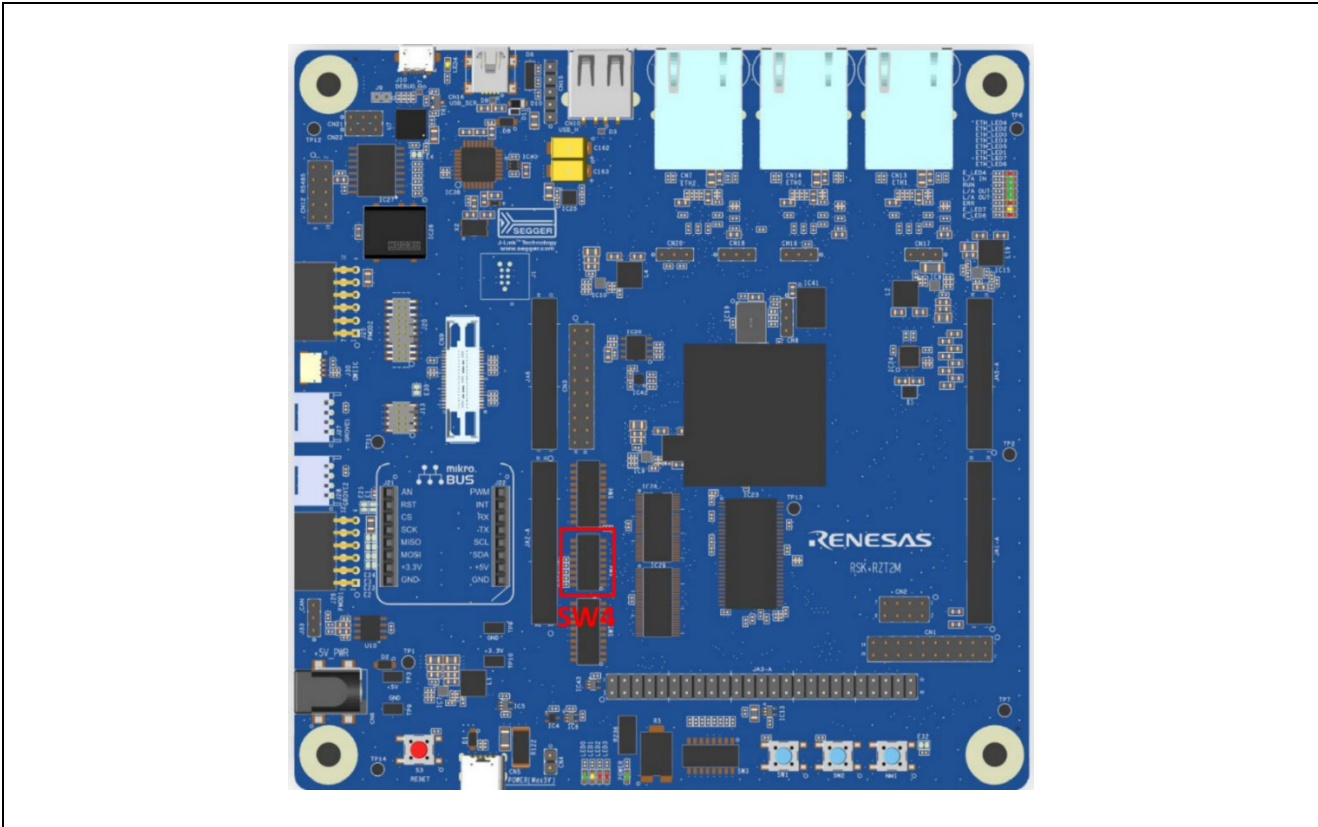


Figure 44. Switch Position of Operation Mode Settings for RSK+RZT2M

Table 2. Operation Mode Switch Settings for RSK+RZT2M

Switch	Setting	Description
SW4.1	ON	16-bit bus boot mode (NOR Flash)
SW4.2	OFF	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	ON	ATCM 0 wait Valid for CPU operating frequency equal to or less than 400MHz.

### 3.4.1.2 Debugger Connection

If you use a JTAG connection with I-Jet or J-Link,

1. Short the jumper pin (J9) for switching the debug connection so that the RSK+RZT2M board can use the emulator connected to the JTAG connector (J20).
2. Connect the emulator (J-Link or I-Jet) to a free USB port on your computer.
3. Connect the I-Jet to the RSK+RZT2M board, ensuring that it is plugged into the header "J20".

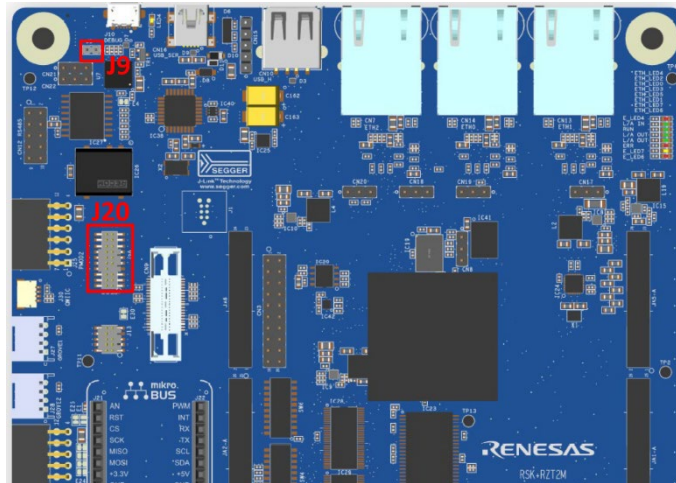


Figure 45. Jumper Position of JTAG Connection for RSK+RZT2M

If you use J-Link OB on RSK+RZT2M board,

1. Open the jumper pin (J9) for switching the debug connection so that RSK+RZT2M can use J-Link OB on the board.
2. Connect the micro-USB type-B to the J-Link OB USB connector (J10), and then LED4 is lit.

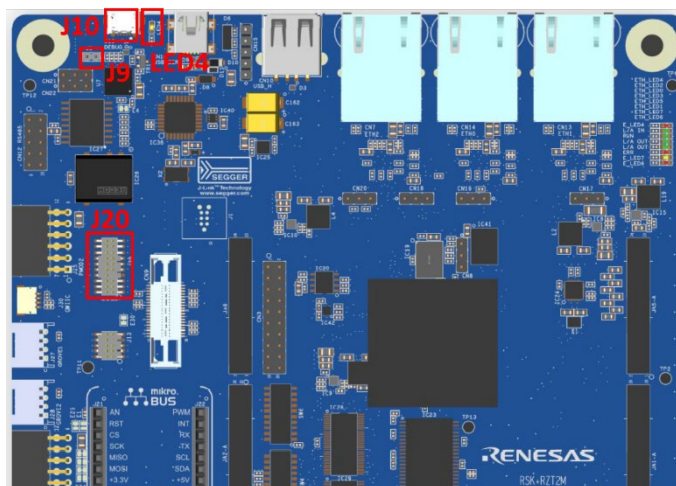


Figure 46. J-Link OB Connection Settings for RSK+RZT2M

### 3.4.1.3 Power Supply

Power is supplied using a USB-C cable or an AC / DC adapter.

- When using a USB cable (Type-C), connect it to the USB connector “CN5” of the RSK+RZT2M board.
- When connecting the AC / DC adapter, connect it to the USB connector “CN6” of the RSK+RZT2M board.

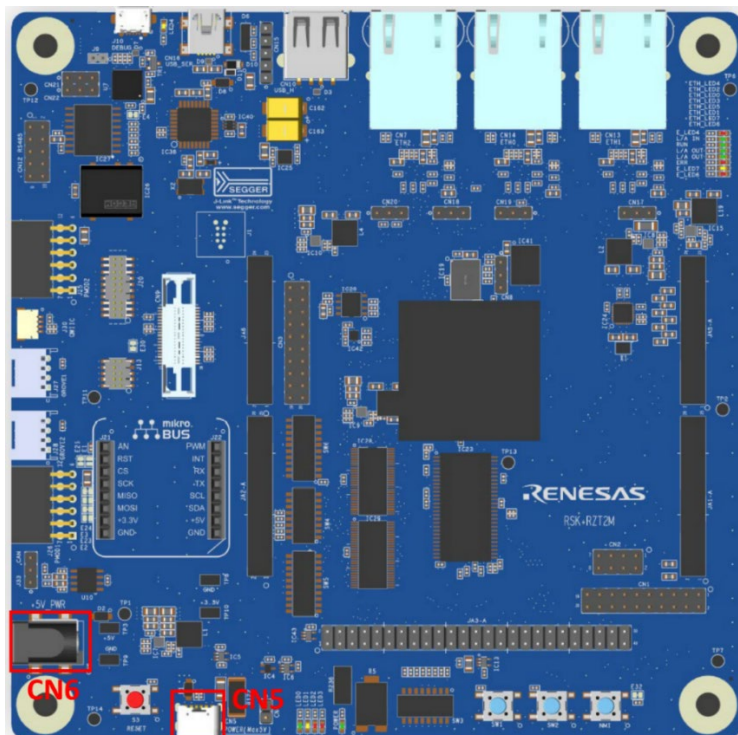


Figure 47. How to Power Supply for RSK+RZT2M

### 3.4.2 RSK+RZT2L

#### 3.4.2.1 Boot Mode

The operation mode settings for the RSK+RZT2L board are as follows.

**Note:** This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, please refer to the manual of the RSK boards listed in chapter 1.3.1. For [the sample codes available on Renesas website](#), please refer to the documentation included with each code and implement the appropriate board settings, respectively.

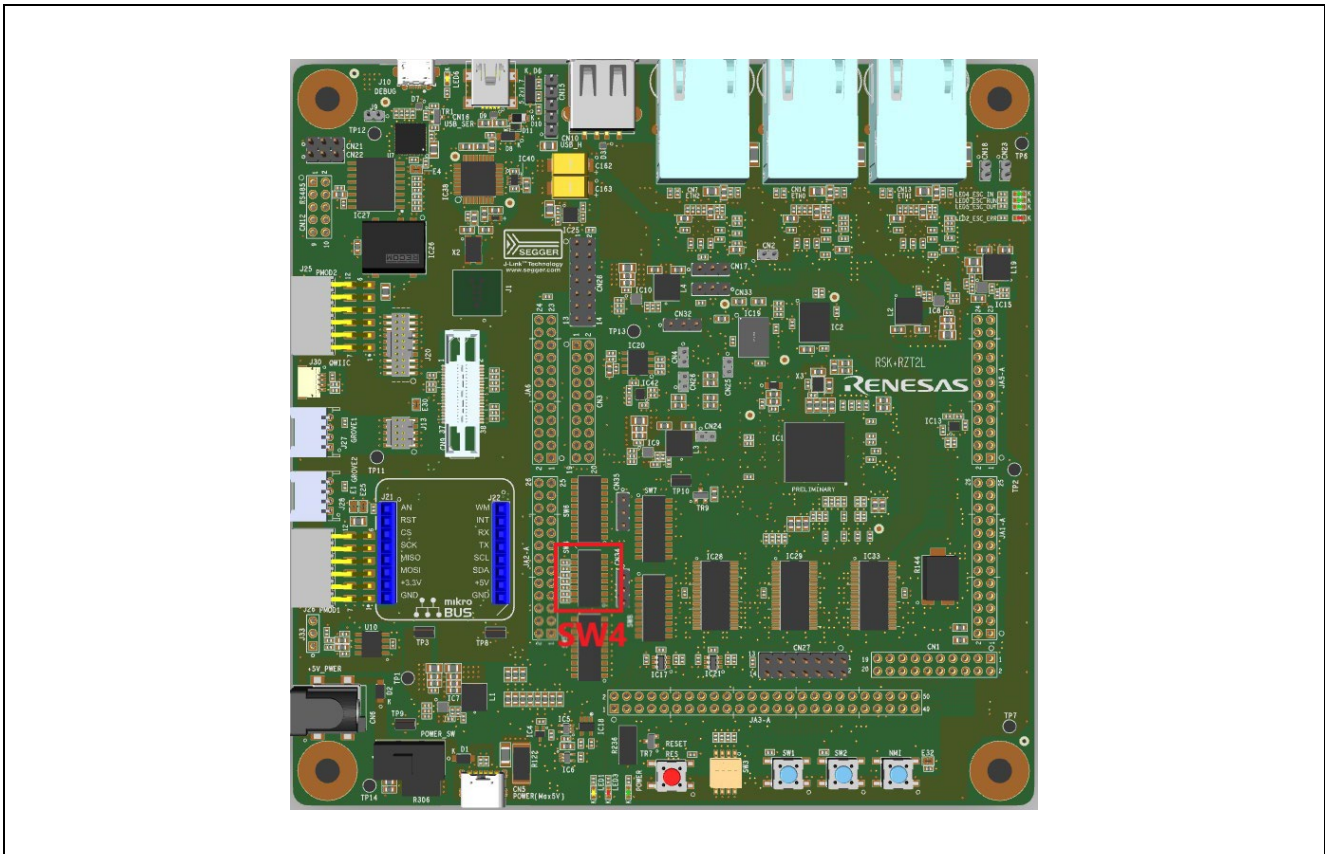


Figure 48. Switch Position of Operation Mode Settings for RSK+RZT2L

Table 3. Operation Mode Switch Settings for RSK+RZT2L

Switch	Setting	Description
SW4.1	ON	xSPI0 boot mode (x1 boot serial flash)
SW4.2	ON	
SW4.3	ON	
SW4.4	ON	ATCM wait cycle = 0 wait
SW4.5	ON	JTAG mode = Normal mode

### 3.4.2.2 Debugger Connection

If you use a JTAG connection with I-Jet or J-Link,

1. Short the jumper pin (J9) for switching the debug connection so that the RSK+RZT2L board can use the emulator connected to the JTAG connector (J20).
2. Connect the emulator (J-Link or I-Jet) to a free USB port on your computer.
3. Connect the I-Jet to the RSK+RZT2L board, ensuring that it is plugged into the header “J20”.

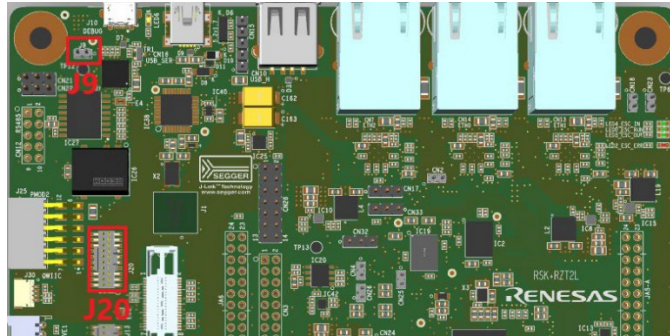


Figure 49. Jumper Position of JTAG Connection for RSK+RZT2L

If you use J-Link OB on RSK+RZT2L board,

1. Open the jumper pin (J9) for switching the debug connection so that RSK+RZT2L can use J-Link OB on the board.
2. Connect the micro-USB type-B to the J-Link OB USB connector (J10), and then the LED6 is lit.

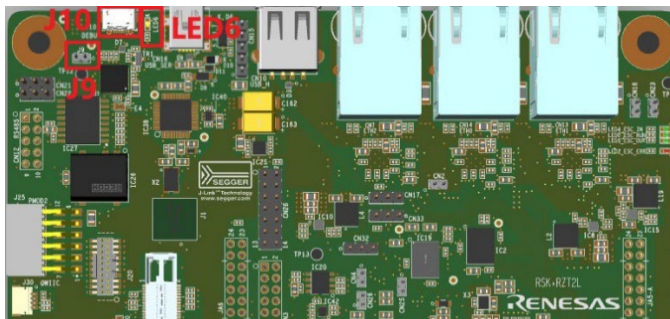


Figure 50. J-Link OB Connection Settings for RSK+RZT2L

### 3.4.2.3 Power Supply

Power is supplied using a USB-C cable or an AC / DC adapter.

- When using a USB cable (Type-C), connect it to the USB connector “CN5” of the RSK+RZT2L board.
- When connecting the AC / DC adapter, connect it to the USB connector “CN6” of the RSK+RZT2L board.
- After connecting to the power (CN5 or CN6), turn on the POWER\_SW slide switch to start power supply.

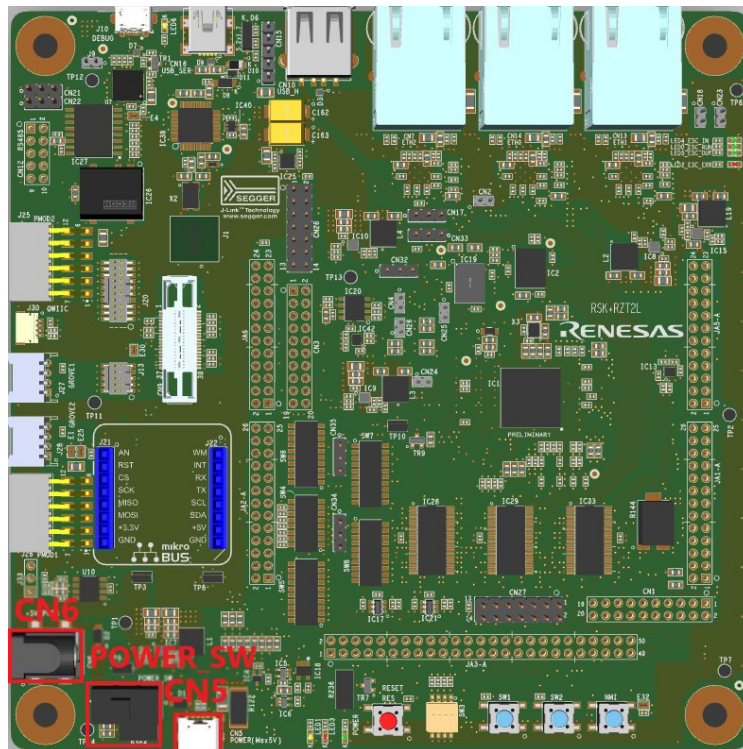


Figure 51. How to Power Supply for RSK+RZT2L

### 3.4.3 RSK+RZT2ME

For each setting, see 3.4.1 RSK+RZT2M.

### 3.4.4 RZ/T2H Evaluation Board

#### 3.4.4.1 Boot Mode

The operation mode settings for the RZ/T2H evaluation board are as follows.

**Note:** This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, please refer to the manual of the evaluation board listed in chapter 1.3.1. For [the sample codes available on Renesas website](#), please refer to the documentation included with each code and implement the appropriate board settings, respectively.

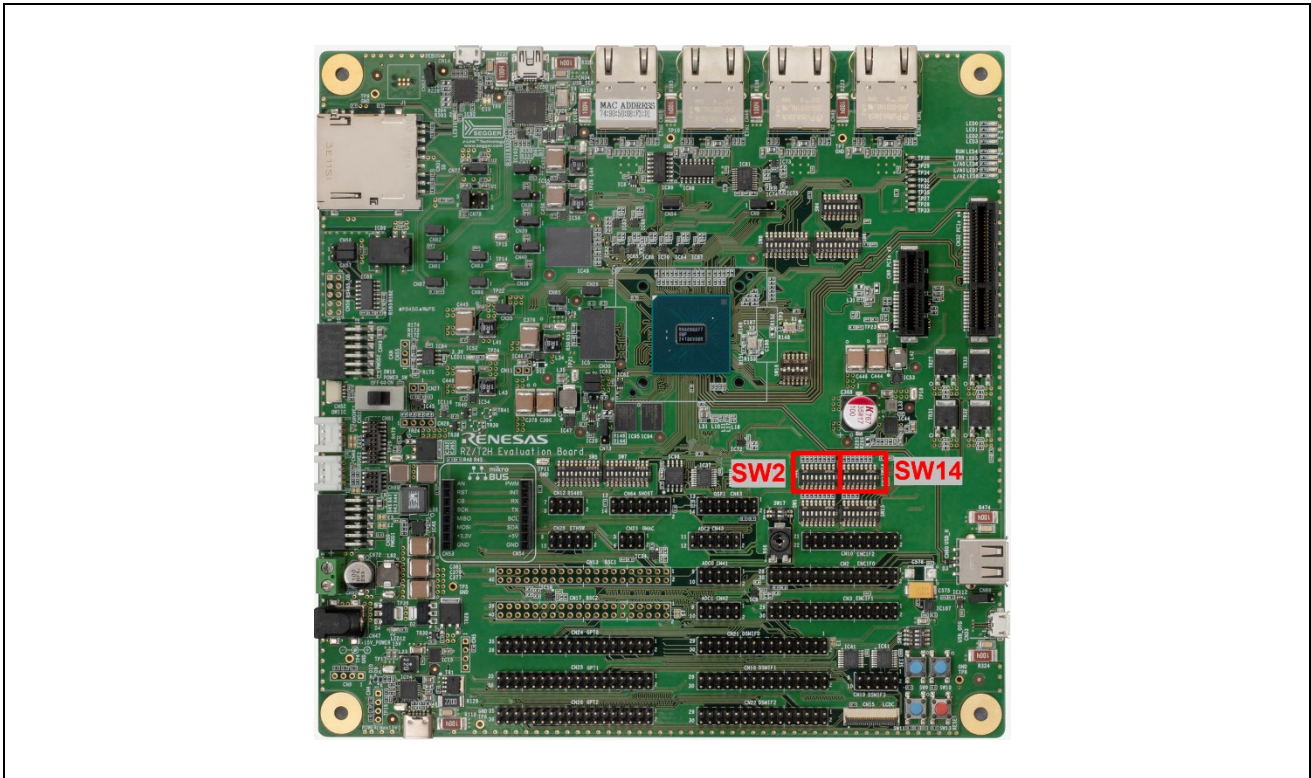


Figure 52. Switch Position of Operation Mode Settings for RZ/T2H Evaluation Board

Table 4. Operation Mode Switch Settings for RZ/T2H Evaluation Board

Switch	Setting	Description
SW14.1	ON	xSPI1 boot mode (x1 boot serial flash)
SW14.2	OFF	
SW14.3	ON	
SW14.4	ON	CPU0 ATCM 0 wait
SW14.7	ON	JTAG Authentication by Hash is disabled.
SW2.3	OFF	This is necessary to light up LED3 (corresponding to CA55 Core1 blinky operation). Note: This switch is not present on the provisional version of the board. Due to this setting, P17_4, P08_5, and P08_6 cannot be used as SD1 control terminals.

### 3.4.4.2 Debugger Connection

If you use a JTAG connection with I-Jet or J-Link,

1. Short the jumper block (CN62) for switching the debug connection so that the RZ/T2H evaluation board can use the emulator connected to the JTAG connector (CN61).
2. Connect the emulator (J-Link or I-Jet) to a free USB port on your computer.
3. Connect the emulator to the RZ/T2H evaluation board, ensuring that it is plugged into the header "CN61".

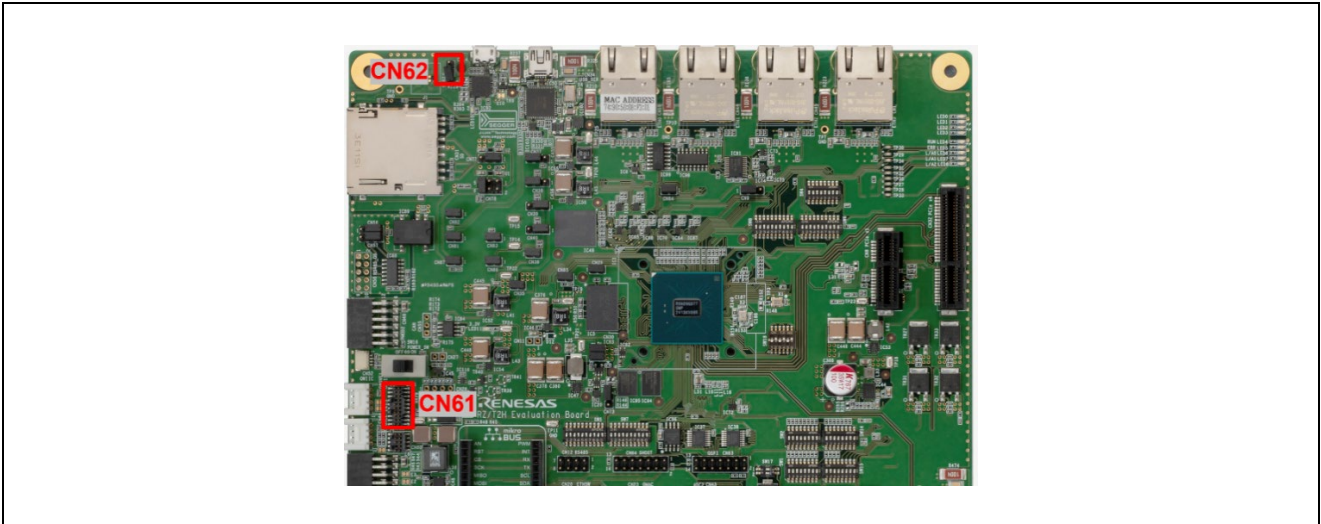


Figure 53. Jumper Position of JTAG Connection for RZ/T2H Evaluation Board

If you use J-Link OB on the RZ/T2H evaluation board,

1. Open the jumper block (CN62) for switching the debug connection so that the RZ/T2H evaluation board can use J-Link OB on the board.
2. Connect the micro-USB type-B to the J-Link OB USB connector (CN14), and then the LED10 is lit.

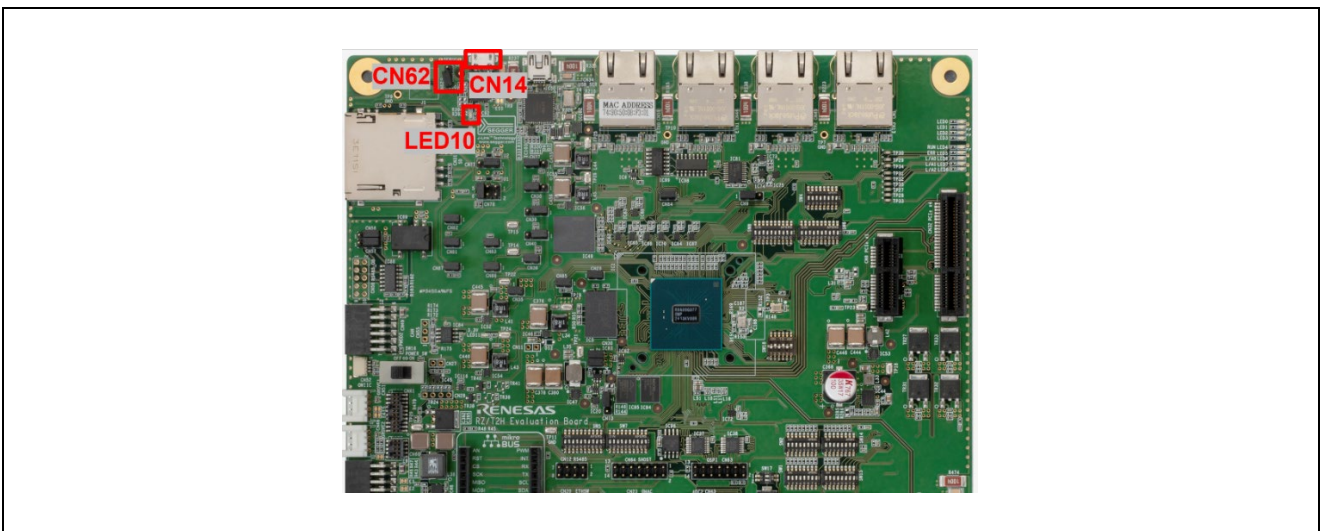


Figure 54. J-Link OB Connection Settings for RZ/T2H Evaluation Board

### 3.4.4.3 Power Supply

Power is supplied using a USB-C cable or an AC / DC adapter.

- When using a USB cable (Type-C), connect it to the USB connector “CN46” of the RZ/T2H evaluation board.
- When connecting the AC / DC adapter, connect it to the USB connector “CN47” of the RZ/T2H evaluation board.
- After connecting to the power (CN46 or CN47), turn on the POWER\_SW slide switch to start power supply.

**Note:** Some Renesas boards, such as the Renesas Starter Kit, require a 12-V or 5-V power supply; the supply of this board is 15-V / 3 A. Be careful not to accidentally connect a 12-V or 5-V power supply. When supplying power through CN47, use a stabilized power source that is capable of supplying at least 15-V / 3 A.

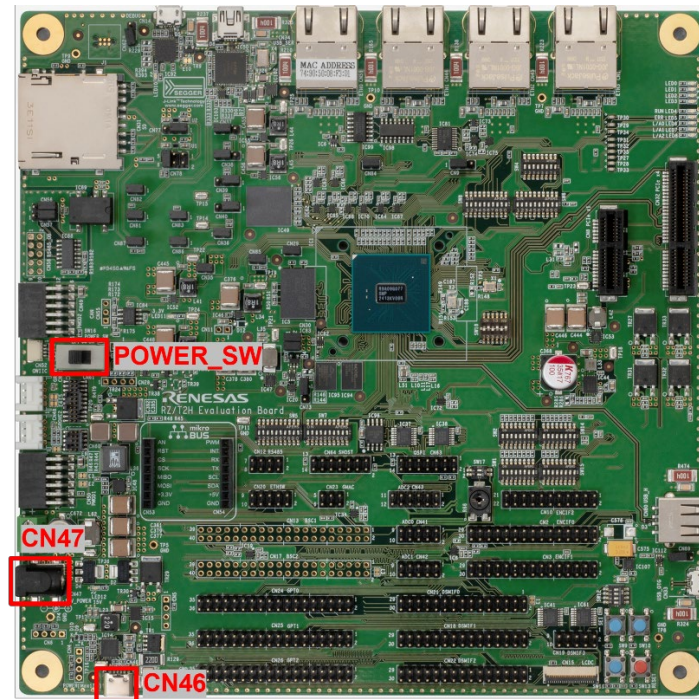


Figure 55. How to Power Supply for RZ/T2H Evaluation Board

## 3.5 RZ/N Series Board Setup

### 3.5.1 RSK+RZN2L

#### 3.5.1.1 Boot Mode

The operation mode settings for the RSK+RZN2L board are as follows.

**Note:** This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, please refer to the manual of the RSK boards listed in chapter 1.3.1. For [the sample codes available on Renesas website](#), please refer to the documentation included with each code and implement the appropriate board settings, respectively.

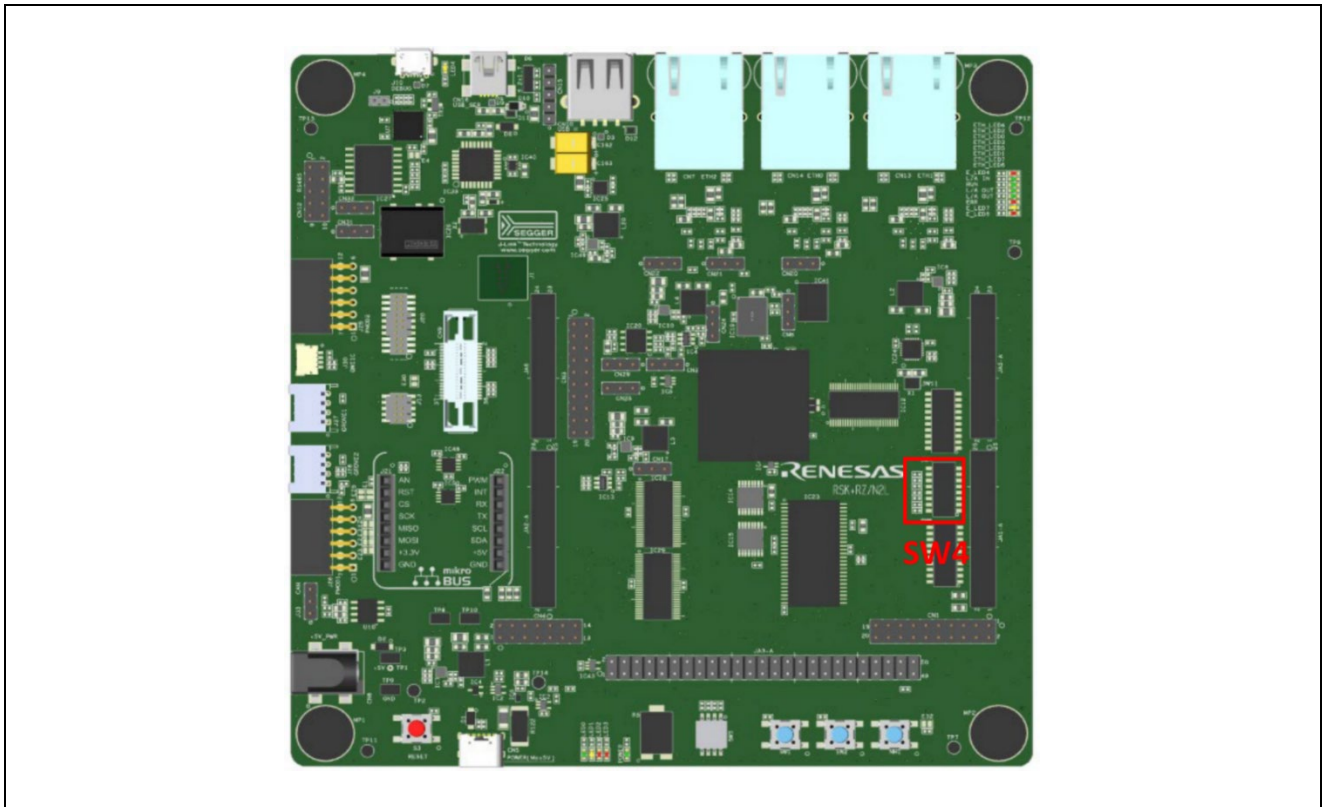


Figure 56. Switch Position of Operation Mode Settings for RSK+RZN2L

Table 5. Operation Mode Switch Settings for RSK+RZN2L

Switch	Setting	Description
SW4.1	ON	16-bit bus boot mode (NOR flash)
SW4.2	OFF	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.

### 3.5.1.2 Debugger Connection

If you use a JTAG connection with I-Jet or J-Link,

1. Short the jumper pin (J9) for switching the debug connection so that the RSK+RZN2L board can use the emulator connected to the JTAG connector (J20).
2. Connect the Emulator (J-Link or I-jet) to a free USB port on your computer.
3. Connect the I-Jet to the RSK+RZN2L board, ensuring that it is plugged into the header "J20".

The figure below shows an I-jet used as an Emulator.

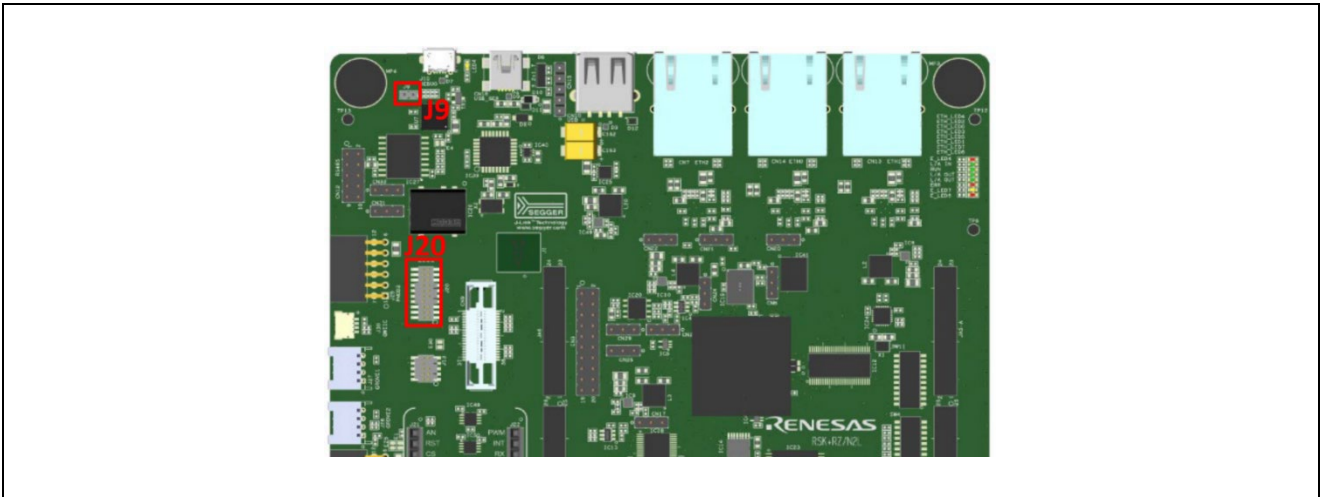


Figure 57. Jumper Position of JTAG Connection for RSK+RZN2L

If you use J-Link OB on the RSK+RZN2L board,

1. Open the jumper pin (J9) for switching the debug connection so that RSK+RZN2L can use J-Link OB on the board.
2. Connect the micro-USB type-B to the J-Link OB USB connector (J10), and then the LED4 is lit.

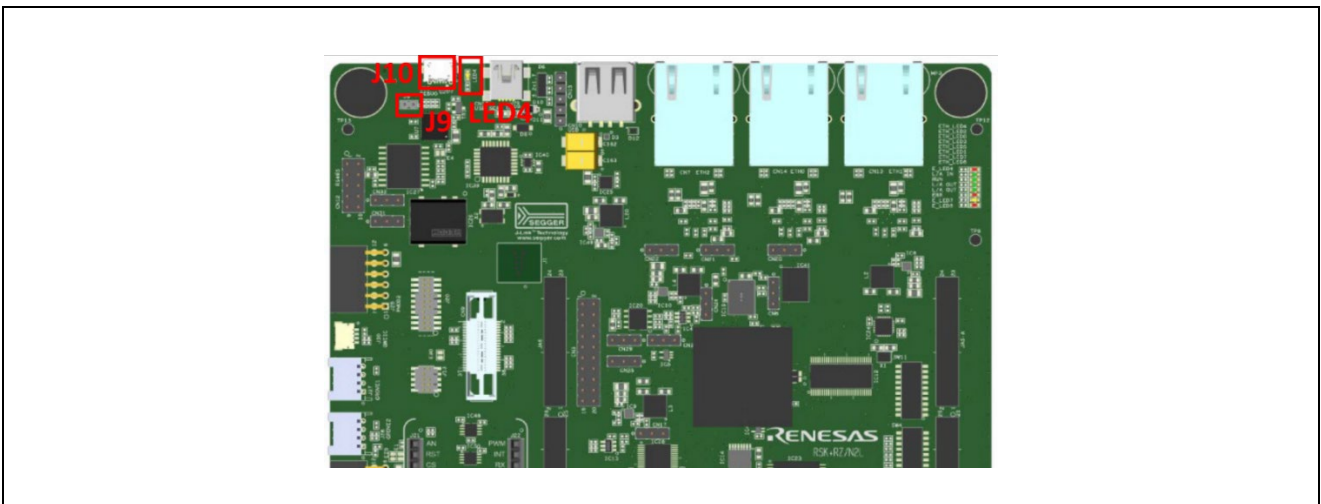


Figure 58. J-Link OB Connection Settings for RSK+RZN2L

### 3.5.1.3 Power Supply

Power is supplied using a USB-C cable or an AC / DC adapter.

- When using a USB cable (Type-C), connect it to the USB connector “CN5” of the RSK+RZN2L board.
- When connecting the AC / DC adapter, connect it to the USB connector “CN6” of the RSK+RZN2L board.

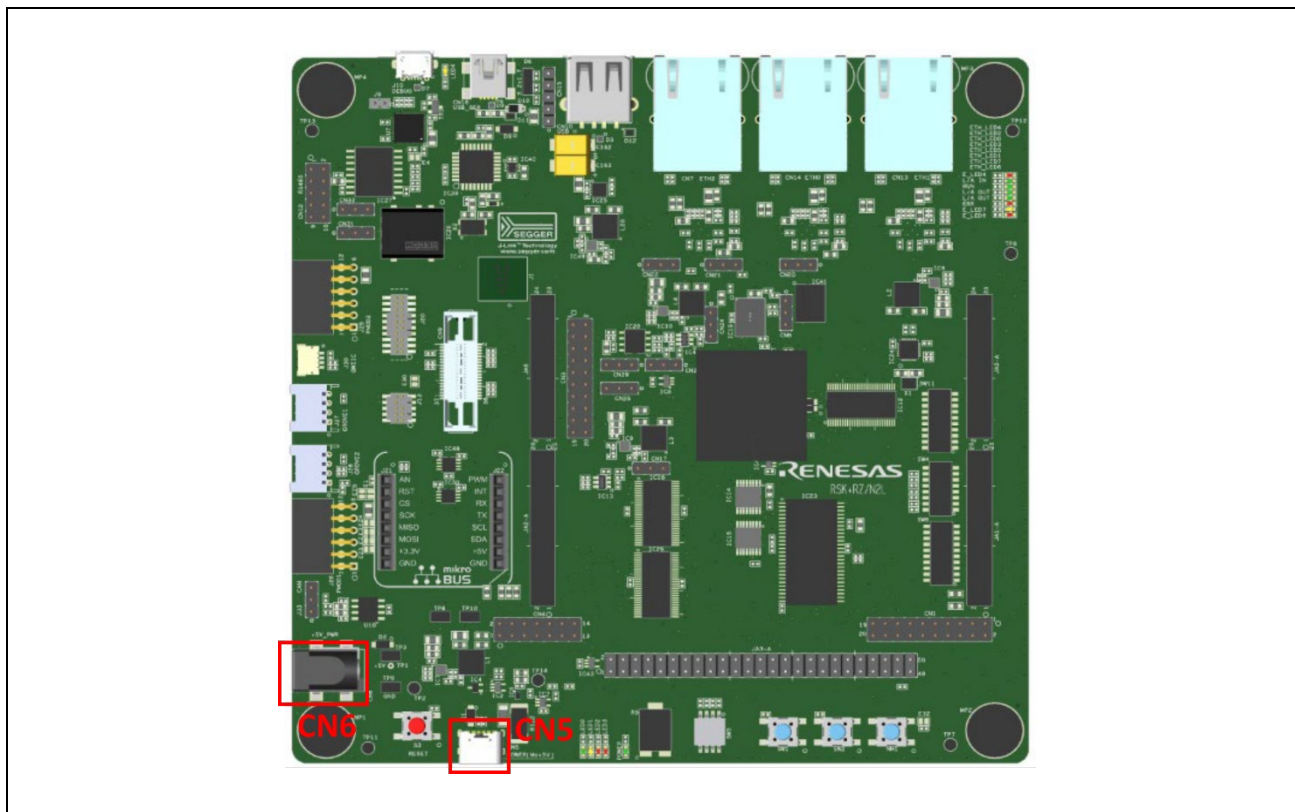


Figure 59. How to Power Supply for RSK+RZN2L

### 3.5.2 RZ/N2H Evaluation Board

#### 3.5.2.1 Boot Mode

The operation mode settings for the RZ/N2H evaluation board are as follows.

**Note:** This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, please refer to the manual of the evaluation board listed in chapter 1.3.1. For [the sample codes available on Renesas website](#), please refer to the documentation included with each code and implement the appropriate board settings, respectively.

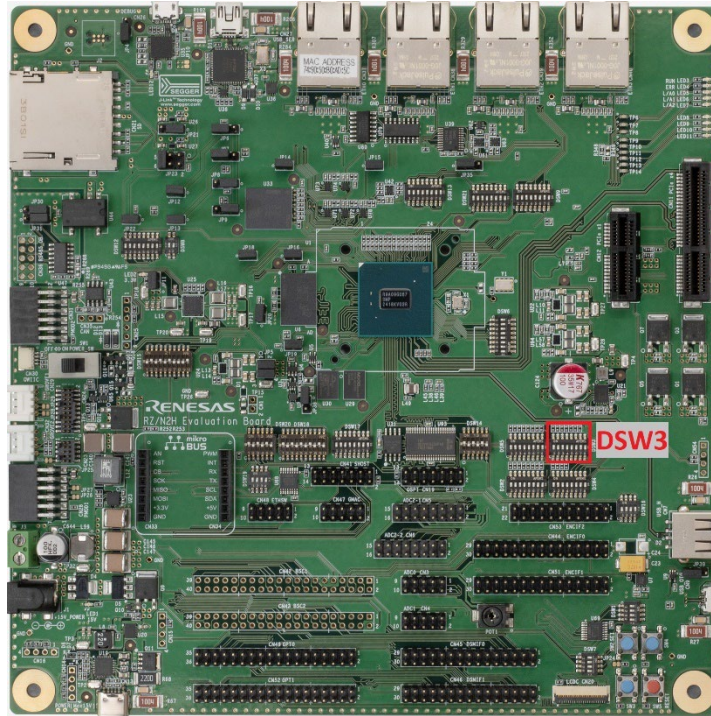


Figure 60. Switch Position of Operation Mode Settings for RZ/N2H Evaluation Board

Table 6. Operation Mode Switch Settings for RZ/N2H Evaluation Board

Switch	Setting	Description
DSW3.1	ON	xSPI1 boot mode (x1 boot serial flash)
DSW3.2	OFF	
DSW3.3	ON	
DSW3.4	ON	CPU0 ATCM 0 wait
DSW3.7	ON	JTAG Authentication by Hash is disabled.

### 3.5.2.2 Debugger Connection

If you use a JTAG connection with I-Jet or J-Link,

1. Short the jumper block (JP40) for switching the debug connection so that the RZ/N2H evaluation board can use the emulator connected to the JTAG connector (CN24).
2. Connect the emulator (J-Link or I-Jet) to a free USB port on your computer.
3. Connect the emulator to the RZ/N2H evaluation board, ensuring that it is plugged into the header "CN24".

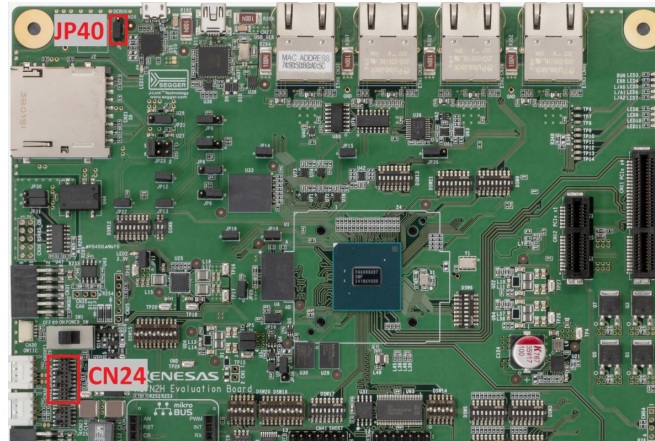


Figure 61. Jumper Position of JTAG Connection for RZ/N2H Evaluation Board

If you use J-Link OB on the RZ/N2H evaluation board,

1. Open the jumper block (JP40) for switching the debug connection so that the RZ/N2H evaluation board can use J-Link OB on the board.
2. Connect the micro-USB type-B to the J-Link OB USB connector (CN26), and then the LED12 is lit.

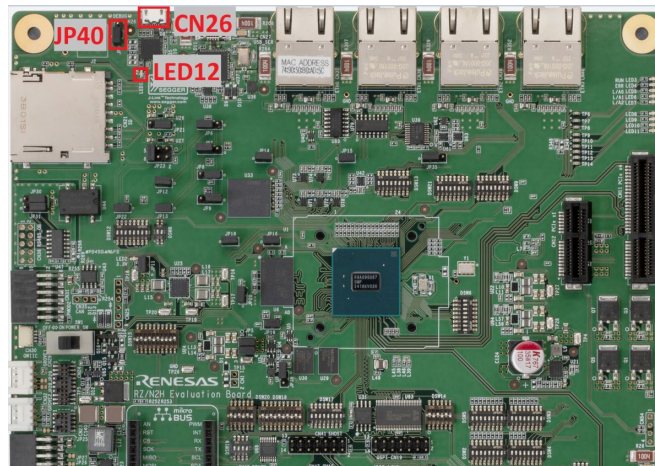


Figure 62. J-Link OB Connection Settings for RZ/N2H Evaluation Board

### 3.5.2.3 Power Supply

Power is supplied using a USB-C cable or an AC / DC adapter.

- When using a USB cable (Type-C), connect it to the USB connector “CN13” of the RZ/N2H evaluation board.
- When connecting the AC / DC adapter, connect it to the USB connector “J1” of the RZ/N2H evaluation board.
- After connecting to the power (J1 or CN13), turn on the POWER\_SW slide switch to start power supply.

**Note:** Some Renesas boards, such as the Renesas Starter Kit, require a 12-V or 5-V power supply; the supply of this board is 15-V / 3 A. Be careful not to accidentally connect a 12-V or 5-V power supply. When supplying power through J1, use a stabilized power source that is capable of supplying at least 15-V / 3 A.

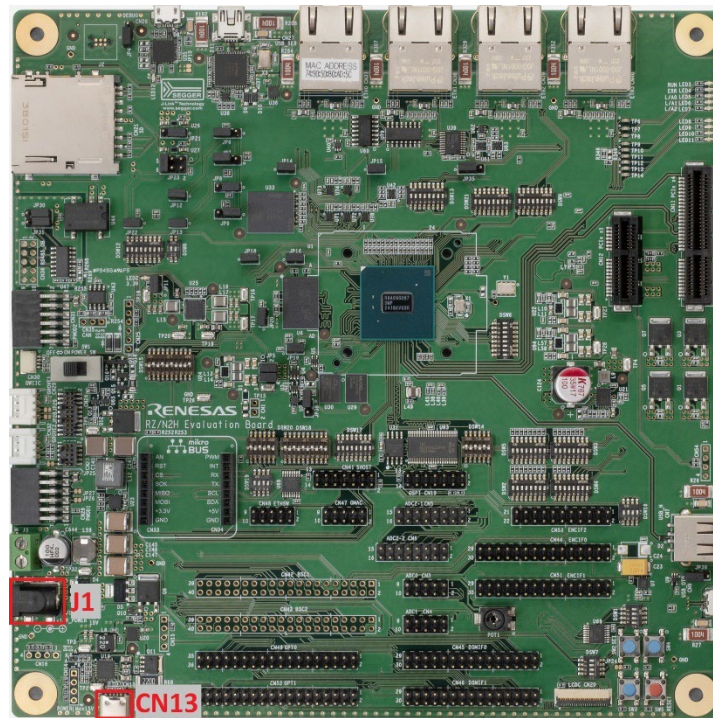


Figure 63. How to Power Supply for RZ/N2H Evaluation Board

## 4. Tutorial: Your First RZ MPU Project – Blinky

### 4.1 Tutorial Blinky

The goal of this tutorial is to quickly get acquainted with the Flexible Platform by moving through the steps of creating a simple application using e<sup>2</sup> studio and running that application on an RZ/T2, RZ/N2 evaluation board. This chapter guides you through creating projects for single-core processing and multiprocessing with RAM execution without flash memory. In this chapter, multiprocessing refers to a process in which the CR52 CPU0 core is activated first, and the second core (CR52 CPU1 or CA55 Core0) operates after the CR52 CPU0 core sets up the second core.

### 4.2 What Does Blinky Do?

The application used in this tutorial is Blinky, traditionally the first program run in a new embedded development environment.

Blinky is the “Hello World” of microprocessors. If the LED blinks, you know that:

- The toolchain is set up correctly and builds a working executable image for your chip.
- The debugger has been installed with working drivers and is properly connected to the board.
- The board is powered up, and its jumper and switch settings are probably correct.
- The microprocessor is alive, the clocks are running, and the memory is initialized.

### 4.3 Create a New Project for Blinky

The creation and configuration of an RZ/T and RZ/N C/C++ FSP Project is the first step in the creation of an application. The base RZ/T2 pack and RZ/N2 packs include a pre-written Blinky example application. The procedure from creating a project to running it varies depending on the number of cores used and boot mode. This chapter shows only some of the cases where RAM execution is used. Refer to Table 7 Project Creation Procedure (e<sup>2</sup> studio) to find out which steps are required for your application.

**Table 7. Project Creation Procedure (e<sup>2</sup> studio)**

Steps	Single-core processing		Multiprocessing	
	RAM execution	Flash boot mode	RAM execution (Combination of (CR52 CPU0, CPU1) and (CR52 CPU0, CA55 Core0) only)	RAM execution (Other combinations) Flash boot mode
Check tool limitations	Appendix Tool Software Limitations			
Erase flash memory(if needed)	Appendix How to Erase Flash Memory			
Create a project	4.3 Create a New Project for Blinky	4.3 Create a New Project for Blinky Appendix. How to Debug the FSP Project with Flash Boot Mode	4.3 Create a New Project for Blinky	Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio
Build the project	4.4.1 Build		4.4.2 Build for Multiprocessing	
Debug the project	4.5.2 Debug Steps		4.7 Debug and Run for Multiprocessing	

---

<b>Run the project</b>	4.6 Run the Blinky Project		
------------------------	----------------------------	--	--

**Note for Multiprocessing Projects:**

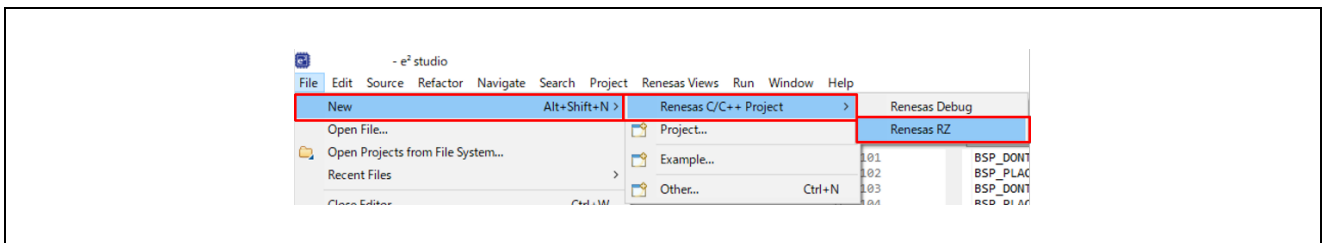
In the case of multiprocessing, two projects with different settings must be created. A project that starts first is called the primary project and the secondary project that runs after releasing reset by the primary project is called the secondary project.

The primary project and the secondary project should be created in the same workspace.

The secondary project should be created after the primary project is created in section 4.3 and built the primary project in 4.4 section.

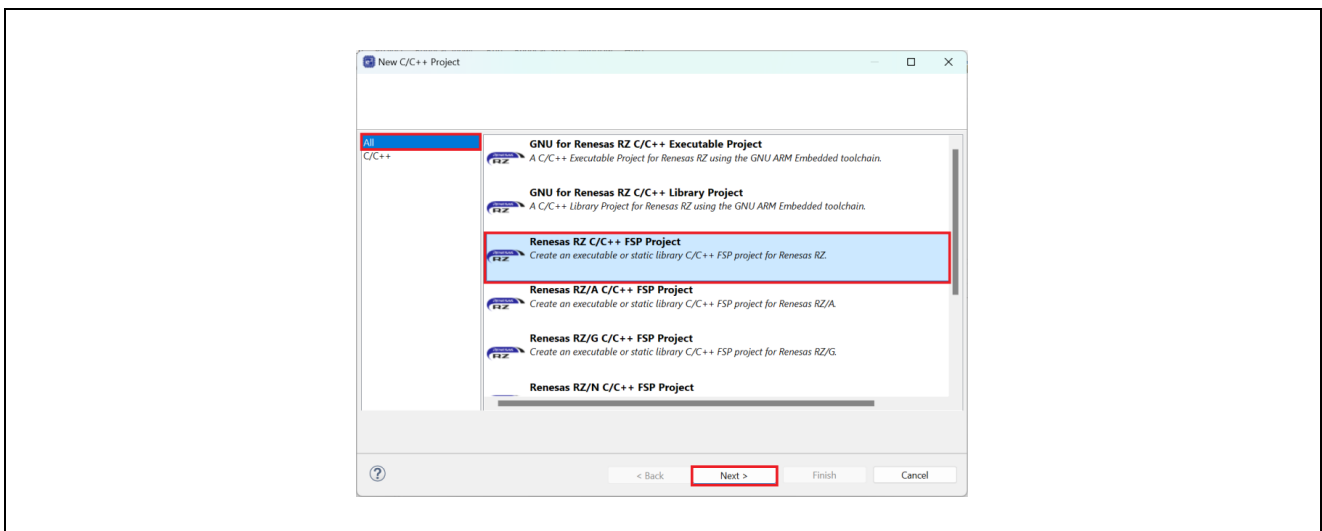
Follow these steps to create an RZ/T2, RZ/N2 MPU project:

1. In e<sup>2</sup> studio, click **File > New > Renesas C/C++ Project > Renesas RZ**.



**Figure 64. New C/C++ Project**

2. Select **All > Renesas RZ C/C++ FSP Project**.
3. Click **Next**.



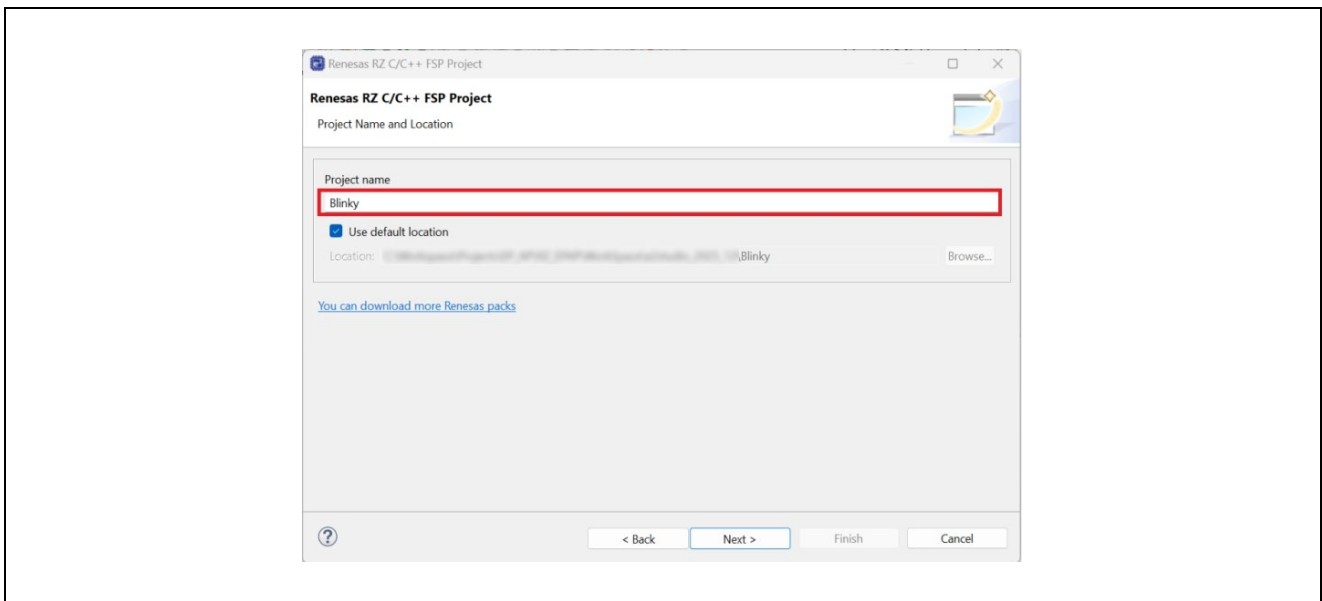
**Figure 65. Renesas RZ C/C++ FSP Project**

4. Assign a name to this new project. An example of naming is shown below.

**Table 8. e<sup>2</sup> studio Newly Created Project Settings (1)**

	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Project name	Blinky	Blinky_primary	Blinky_secondary	Blinky_primary	Blinky_secondary

5. Click **Next**. The Project Configuration window shows your selection.

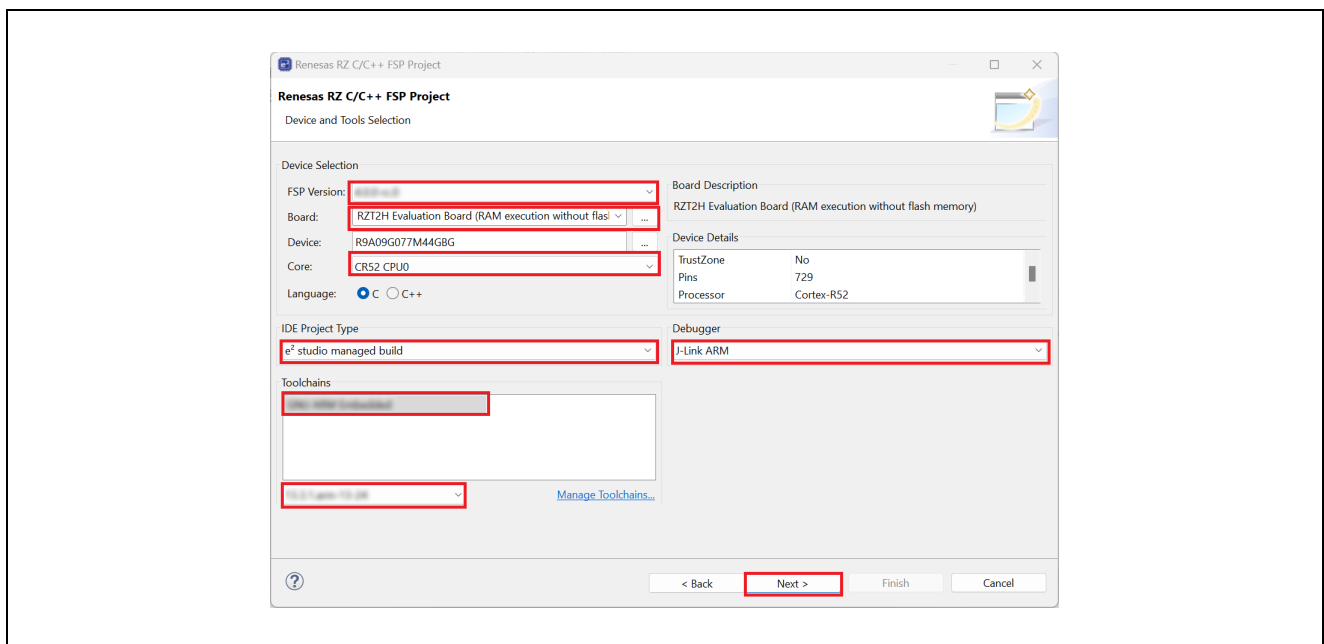


**Figure 66. e<sup>2</sup> studio Project Configuration Window (Part 1)**

6. Select the board support package by selecting the name of your board from the drop-down list.  
In this tutorial, please select either one depending on your device and board.
7. (Multicore device ONLY) Select the Core from the drop-down list.
8. Select toolchains and version, then click **Next**.
  - If there is NO target toolchain, please download the version of the toolchain from the ARM Developer website and install it.

**Table 9. e<sup>2</sup> studio Newly Created Project Settings (2)**

	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Board	RSK+RZXXX (RAM execution without flash memory) or RZXXX Evaluation Board (RAM execution without flash memory)				
Core	CR52_0 or CR52 CPU0	CR52_0 or CR52 CPU0	CR52_1 or CR52 CPU1	CR52 CPU0	CA55 Core0
IDE Project Type	e <sup>2</sup> studio managed build				
Toolchains	GNU ARM Embedded 13.3.1.arm-13-24			GNU ARM A-Profile (AArch64 bare-metal) 13.2.Rel1	
Debugger	J-Link ARM				



**Figure 67. e<sup>2</sup> studio Project Configuration Window (Part 2)**

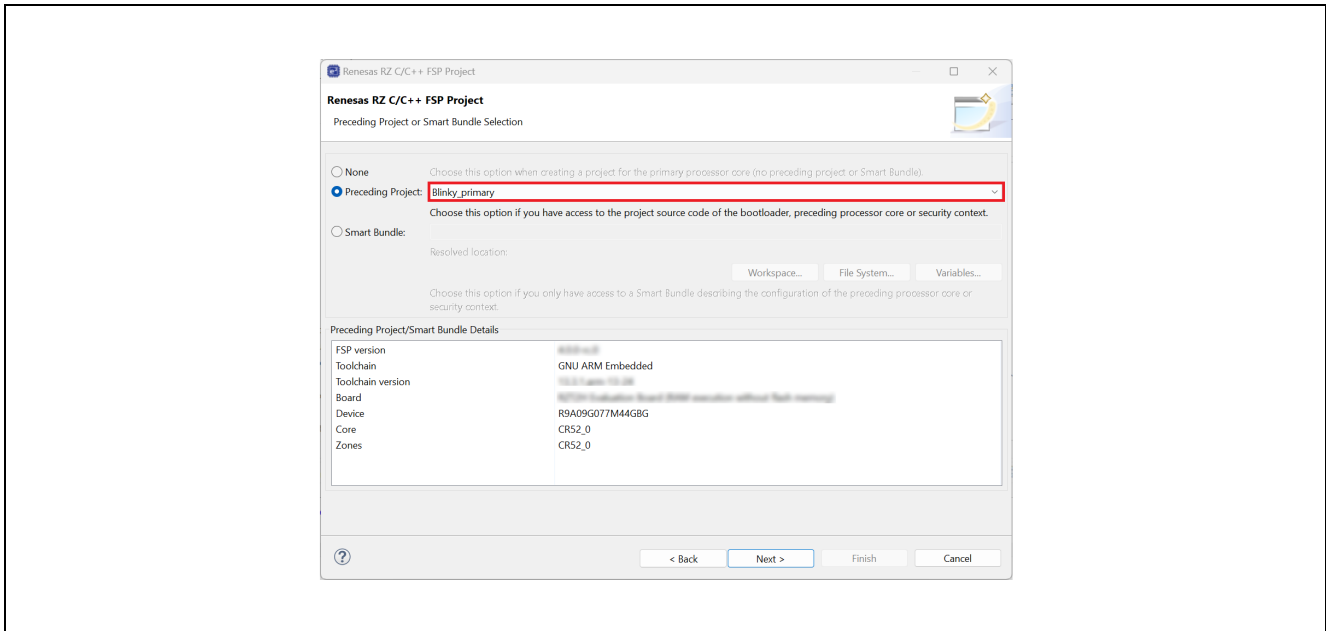
9. Select a bundle file. For the secondary project of multiprocessing, select the primary project as the Preceding Project. Built the primary project names in the same workspace appear as an option in the drop-down list.

**Table 10. e<sup>2</sup> studio Newly Created Project Settings (3)**

	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Preceding Project	None	None	The primary project	None	The primary project

**Note:** Warnings occur if the FSP version or Board (boot mode) used is different between the primary project and the secondary project. Use the same FSP version and Board (boot mode).

Warnings occur when the cores of the primary project and the secondary project are different in multiprocessing, because the Toolchain and Toolchain version do not match. Ignore the warning and proceed to the next step.



**Figure 68. e<sup>2</sup> studio Project Configuration Window (Part 3)**

10. Select the **Build artifact** and RTOS.

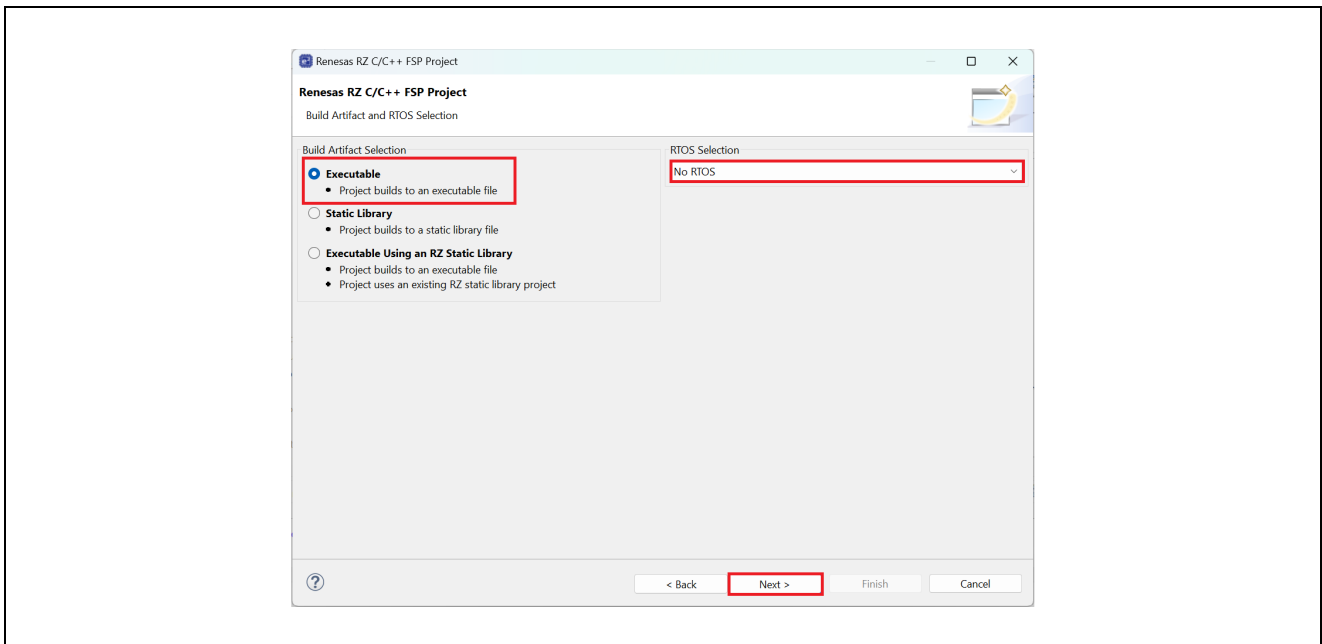


Figure 69. e<sup>2</sup> studio Project Configuration Window (Part 4)

11. Select the **Blinky** template for your board and click **Finish**.

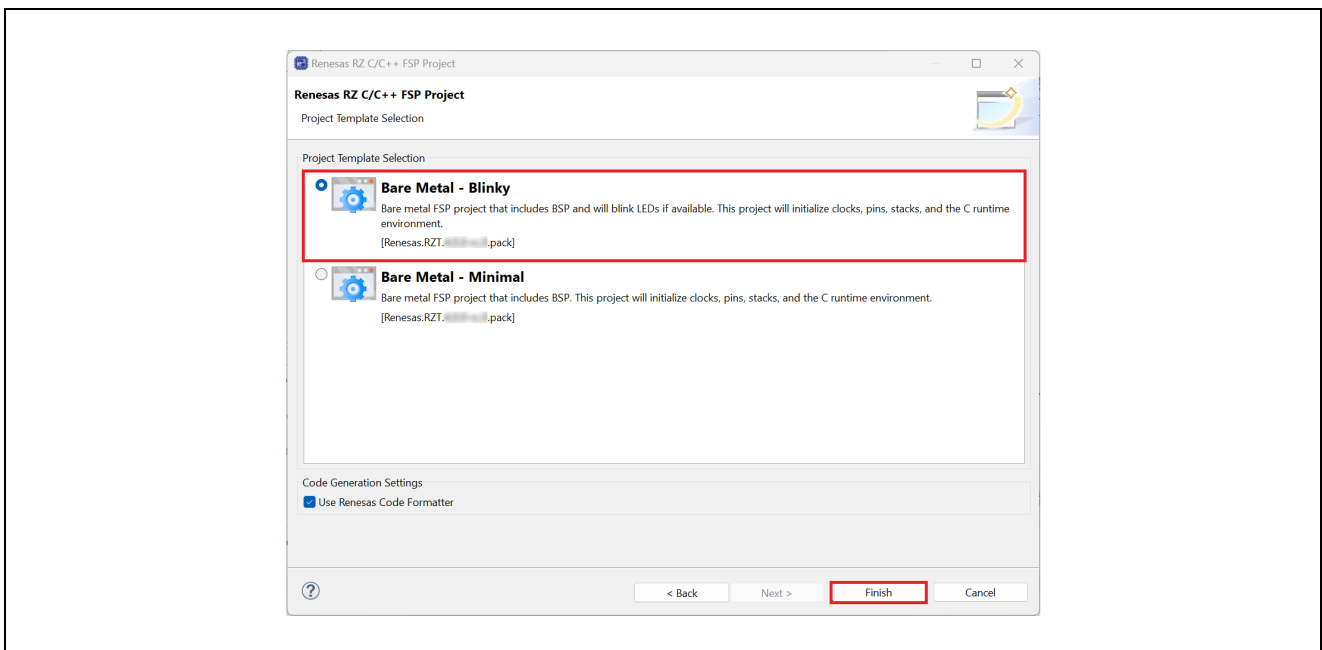
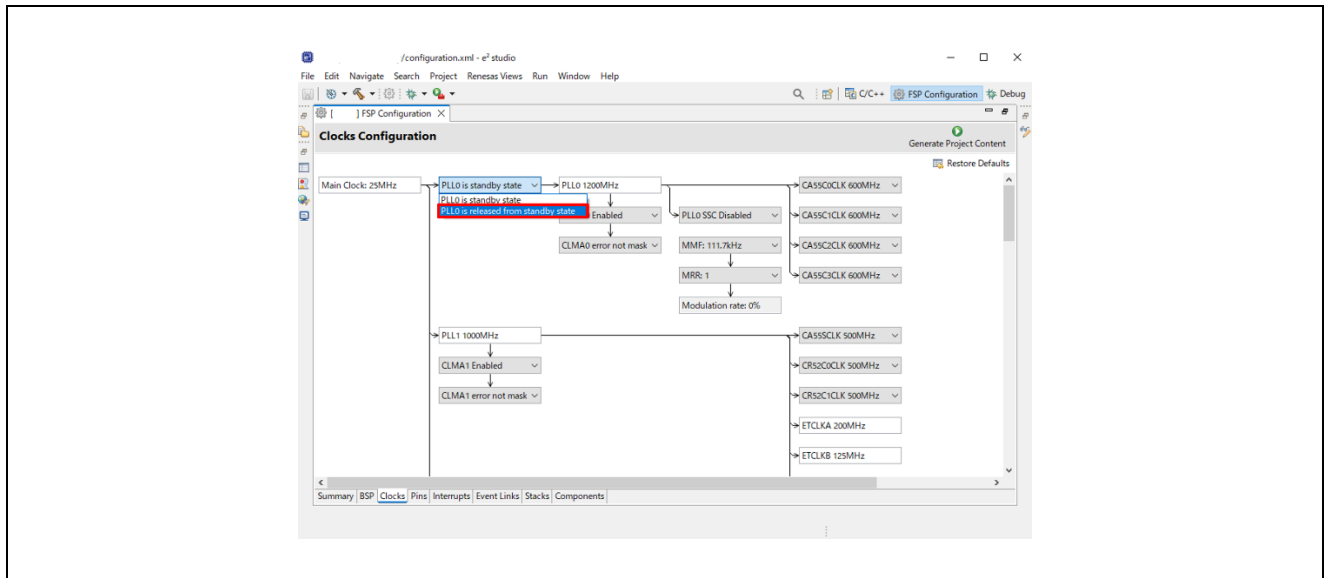


Figure 70. e<sup>2</sup> studio Project Configuration Window (Part 5)

Once the project has been created, the name of the project will show up in the **Project Explorer** window of e<sup>2</sup> studio.

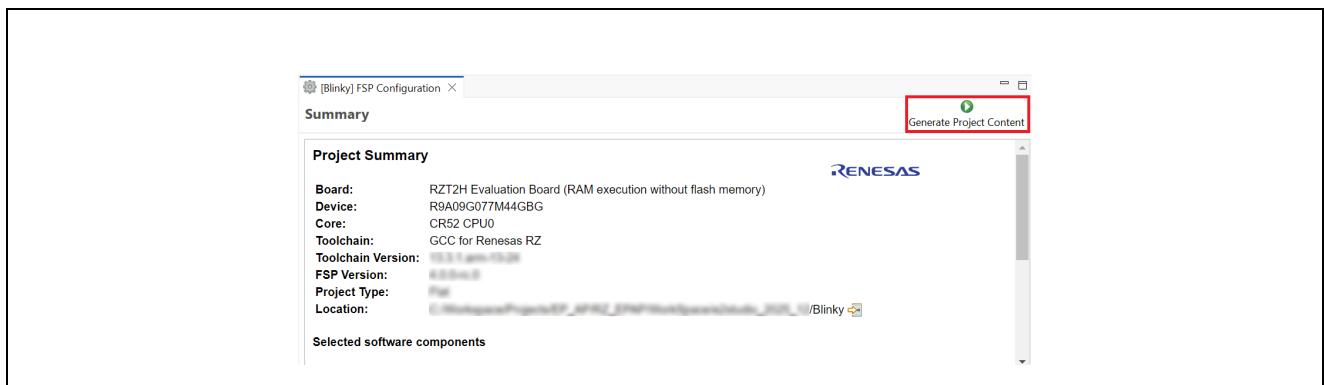
**Note for the primary project using CR52 CPU0:**

If the primary project selects CR52 CPU0 as the **Core** and the secondary or later project uses a CA55 core, you need to set "PLL0 is released from standby state" and enable PLL0 in the Clocks tab of FSP Configuration.



**Figure 71. Enable PLL0 in the Primary Project using CR52 CPU0**

Now, click the Generate Project Content button in the top right corner of the Project Configuration window to generate your board-specific files.



**Figure 72. e<sup>2</sup> studio Project Configuration Tab**

Your new project is now created, configured, and ready to build.

### 4.3.1 Details about the Blinky Configuration

The Generate Project Content button creates configuration header files, copies source files from templates, and generally configures the project based on the state of the Project Configuration screen.

For example, if you check a box next to a module in the Components tab and click the Generate Project Content button, all the files necessary for the inclusion of that module into the project will be copied or created. If that same check box is then unchecked, those files will be deleted.

### 4.3.2 Configuring the Blinky Clocks

By selecting the Blinky template, the clocks are configured by e<sup>2</sup> studio for the Blinky application.

The clock configuration tab (see 6.3.3 Configuring Clocks) shows the Blinky clock configuration. The Blinky clock configuration is stored in the BSP clock configuration file.

### 4.3.3 Configuring the Blinky Pins

By selecting the Blinky template, the GPIO pins used to toggle some of the LEDs are configured by e<sup>2</sup> studio for the Blinky application.

The pin configuration tab shows the pin configuration for the Blinky application (see 6.3.4 Configuring Pins). The Blinky pin configuration is stored in the BSP configuration file.

### 4.3.4 Configuring the Parameters for Blinky Components

The Blinky project automatically selects the following HAL components in the Components tab:

- `r_ioport`

To see the configuration parameters for any of the components, check the Properties tab in the HAL window for the respective drivers (see 6.5 Adding and Configuring HAL Drivers).

### 4.3.5 Where is main()?

The main function is located in:

- `< RZ FSP project >/rz_gen/main.c`.

It is one of the files that are generated during the project creation stage and only contains a call to `hal_entry()`. For more information on generated files, see 6.5 Adding and Configuring HAL Drivers.

### 4.3.6 Blinky Example Code

The blinky application is stored in the `hal_entry.c` file. This file is generated by e<sup>2</sup> studio when you select the Blinky Project template and is located in the project's folder `< project >/src/` folder.

The application performs the following steps:

1. Get the LED information for the selected board by the `bsp_leds_t` structure.
2. Initialize output level for LED pin to LOW using `R_BSP_PinClear((bsp_io_region_t) leds.p_leds[i][1], (bsp_io_port_pin_t) leds.p_leds[i][0])`.
3. Use `R_BSP_PinToggle ((bsp_io_region_t) leds.p_leds[i][1], (bsp_io_port_pin_t) leds.p_leds[i][0])` to set the output level to the LED pin.
4. `R_BSP_SoftwareDelay(delay, bsp_delay_units)` waits for a certain period of time. Then run #3 again.

## 4.4 Build the Blinky Project

Highlight the new project in the Project Explorer window by clicking on it and building it.  
When multiprocessing, please refer to Section 4.4.2 Build for Multiprocessing.

### 4.4.1 Build

There are three ways to build a project:

1. Click on **Project** in the menu bar and select **Build Project**.
2. Click on the hammer icon.
3. Right-click on the project and select **Build Project**.

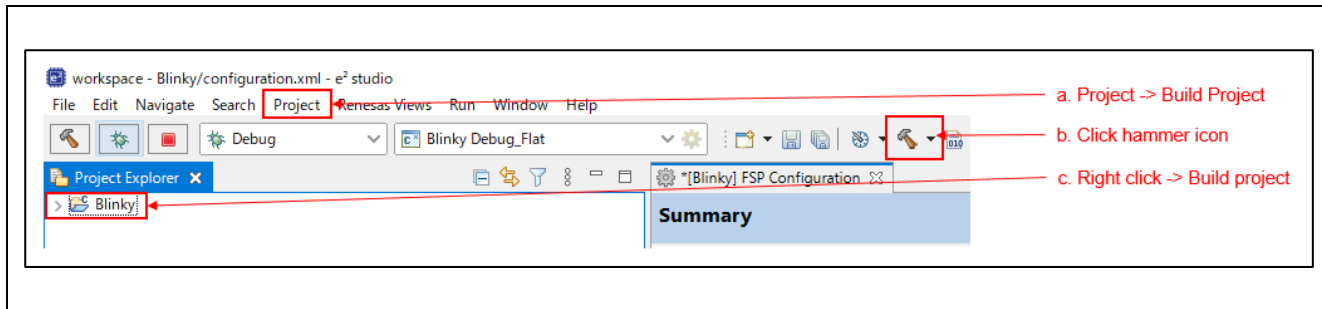


Figure 73. e<sup>2</sup> studio Project Explorer Window

Once the build is complete, a message is displayed in the build Console window that displays the final image file name and section sizes in that image.

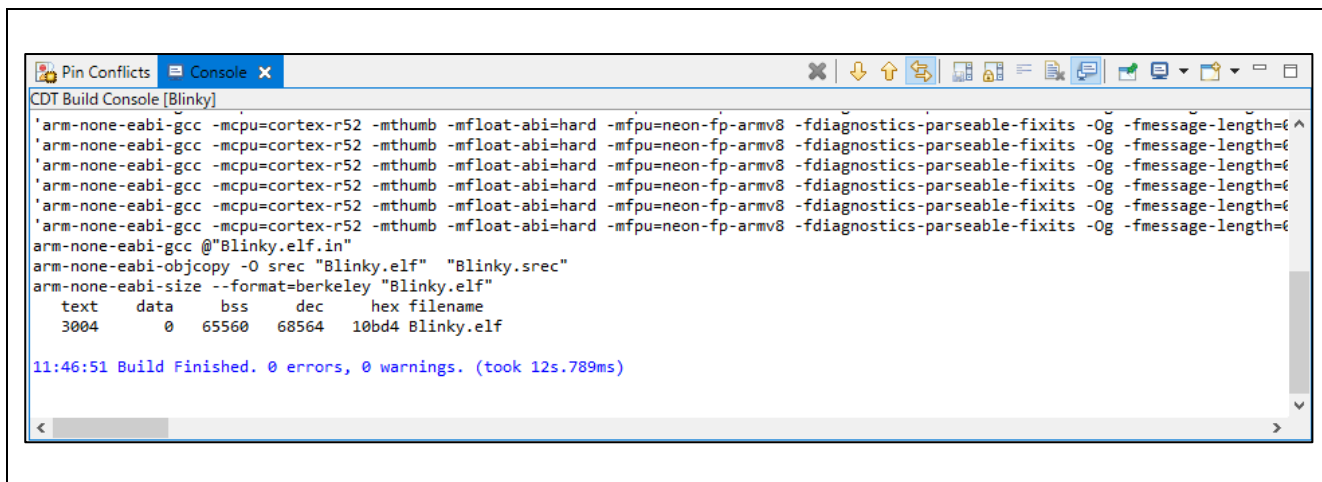


Figure 74. e<sup>2</sup> studio Project Build Console

### 4.4.2 Build for Multiprocessing

Build the projects for multiprocessing with the following steps.

1. Create and build the primary project. (1st build of the primary project)  
Refer to 4.3 Create a New Project for Blinky and 4.4.1 Build.
2. Create the secondary project and build it.
3. Build the primary project again. (2nd build of the primary project)

## 4.5 Debug the Blinky Project

### 4.5.1 Debug Prerequisites

To debug the project on a board, you need the following:

- The board is to be connected to the e<sup>2</sup> studio.
- The debugger needs to be configured to talk to the board.
- The application is to be programmed for the microprocessor.

Applications run from the internal RAM of your microprocessor. To run or debug the application, the application must first be programmed into RAM by a JTAG debugger.

The evaluation board has a JTAG header and requires an external JTAG debugger to connect to the header.

### 4.5.2 Debug Steps

When multiprocessing, please refer to Section 4.7.

Debug and Run for Multiprocessing.

#### Note:

The main chapter of this documentation describes a RAM execution without a flash memory project. When debugging a project with flash boot mode, please also refer to Appendix.

How to Debug the FSP Project with Flash Boot Mode.

To debug the Blinky application, follow these steps. If the step is preceded by (XXX), it is executed only if the condition is met.

(RAM exec): The boot mode used in the project is RAM execution without flash memory.

(CR52): The core used in the project is CR52.

(CA55): The core used in the project is CA55.

1. Configure the debugger for your project by clicking **Run > Debugger Configurations ...** or by selecting the drop-down menu next to the bug icon and selecting **Debugger Configurations ...**

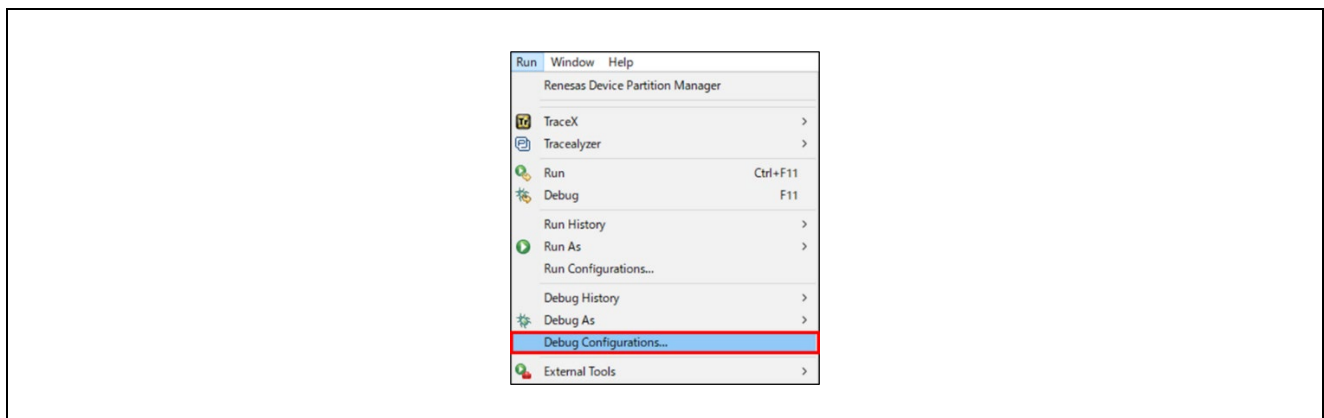


Figure 75. e<sup>2</sup> studio Debugger Configurations Selection Option

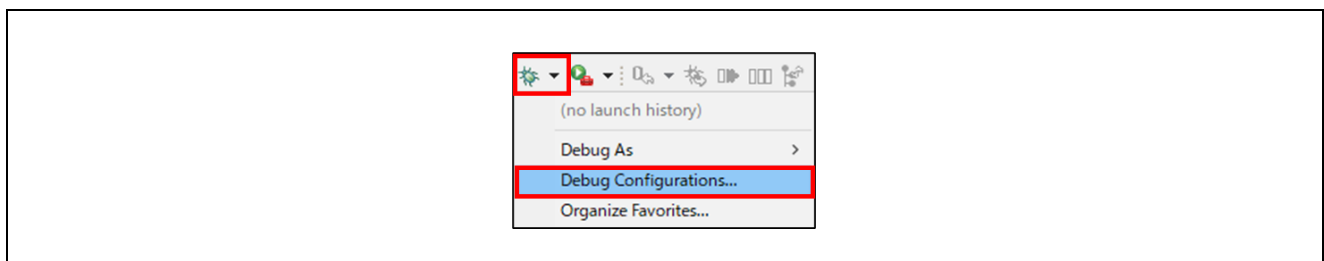
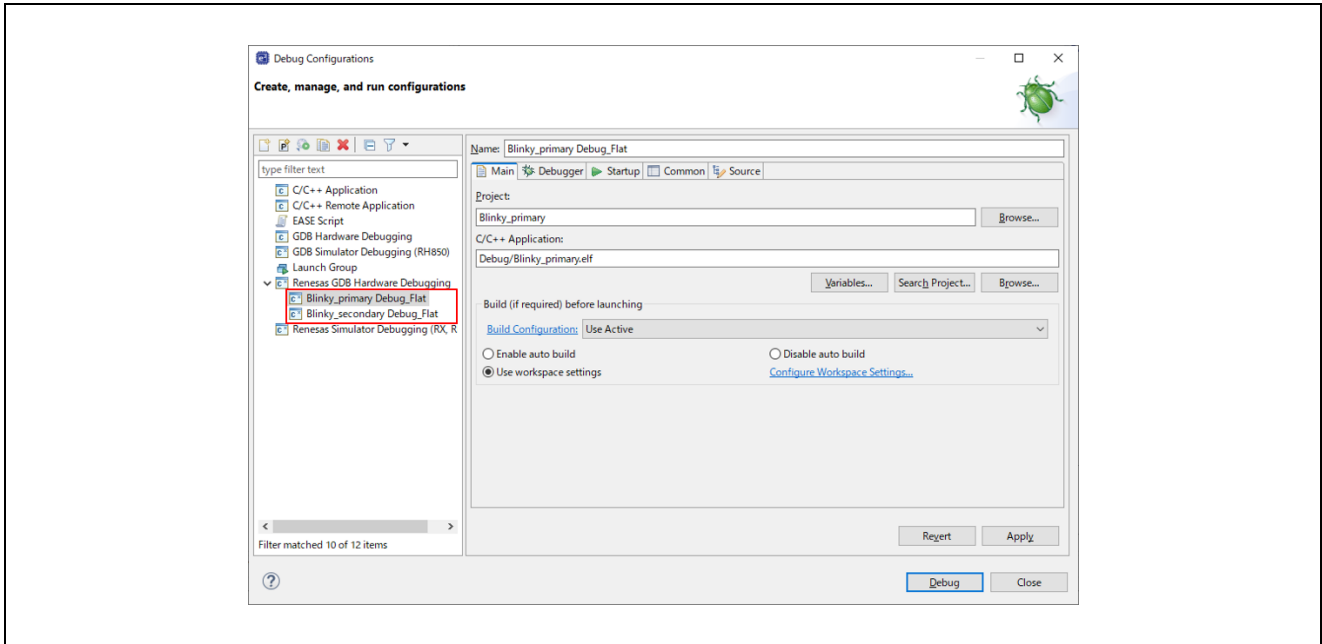


Figure 76. e<sup>2</sup> studio Debug Icon

- Select your debugger filter text configuration in the window. If it is not visible, then it must be created by clicking the New icon in the top left corner of the window. Once selected, the **Debug Configuration** window displays the **Debug configuration** for your **Blinky** project.



**Figure 77. e<sup>2</sup> studio Debugger Configurations Window with Blinky Project**

- If you use RAM execution without flash memory boot mode, you need the following configuration.
  - **Debugger > Connection Settings > Connection**
    - (RAM exec) Set **No** to **Reset after download** to avoid resetting MPU after program download
    - (CR52 CPU0) Set **Yes** to **Set CPSR(5bit) after download** to set the CPSR register value of CR52 general register before running the application.
    - (CR52 CPU1) Set **Yes** to **Prevent Releasing the Reset of the CM3 Core** for debugging.

**Table 11. e<sup>2</sup> studio Newly Created Project Debug Settings (1)**

	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Reset after download	No (default)				
Set CPSR(5bit) after download	Yes	Yes	No (default)	Yes	No (default)
Prevent Releasing the Reset of the CM3 Core	-	-	Yes (default)	-	-

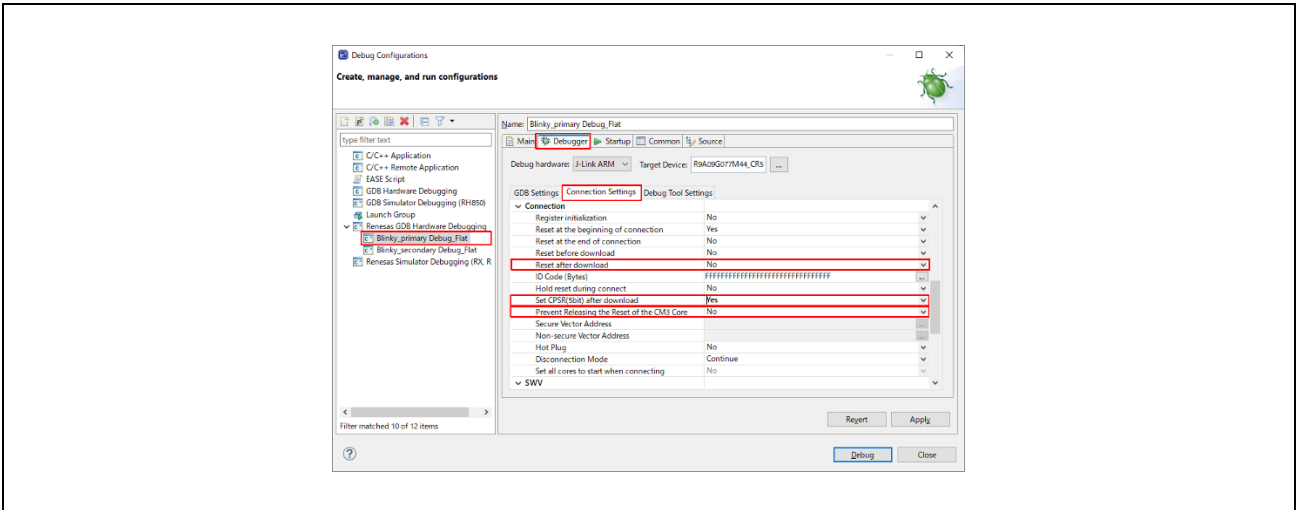


Figure 78. e<sup>2</sup> studio Debugger Configurations Window with Blinky Project (CR52 CPU0)

- (CA55) Check and modify the target device and GDB common settings of the CA55 core project to connect to debugging.

Table 12. e<sup>2</sup> studio Newly Created Tool Project Debug Settings (2)

	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Debugger > Target Device	(default)	(default)	(default)	(default)	target device
Debugger > GDB settings > GDB > GDB Command	(default)	(default)	(default)	(default)	aarch64-elf-gdb

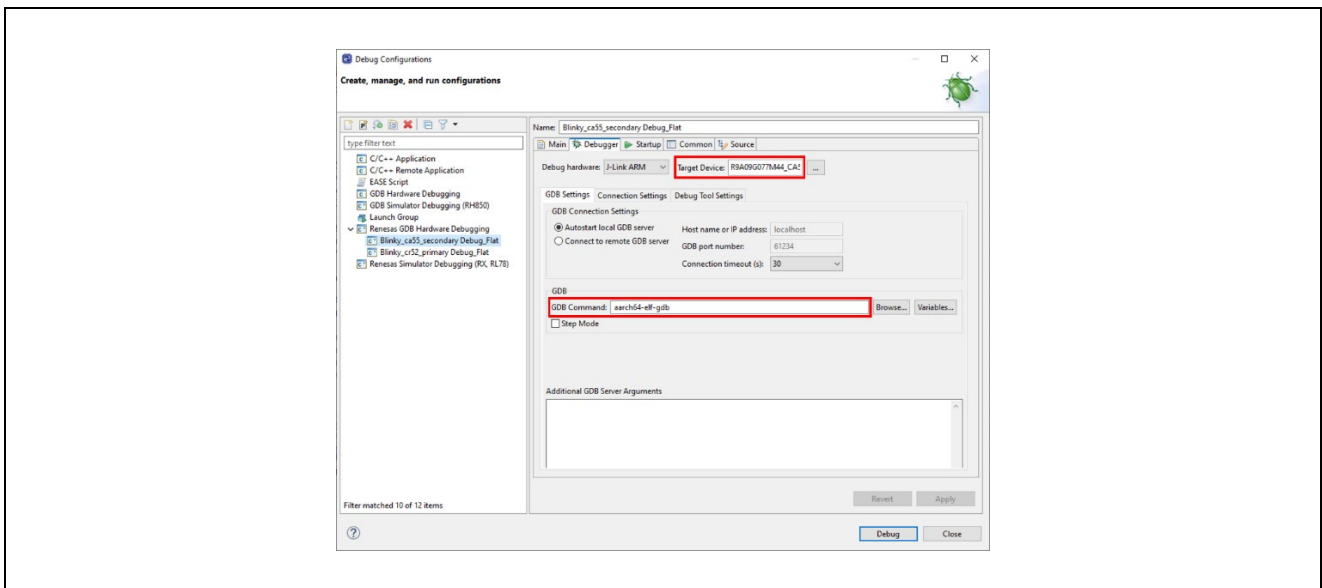
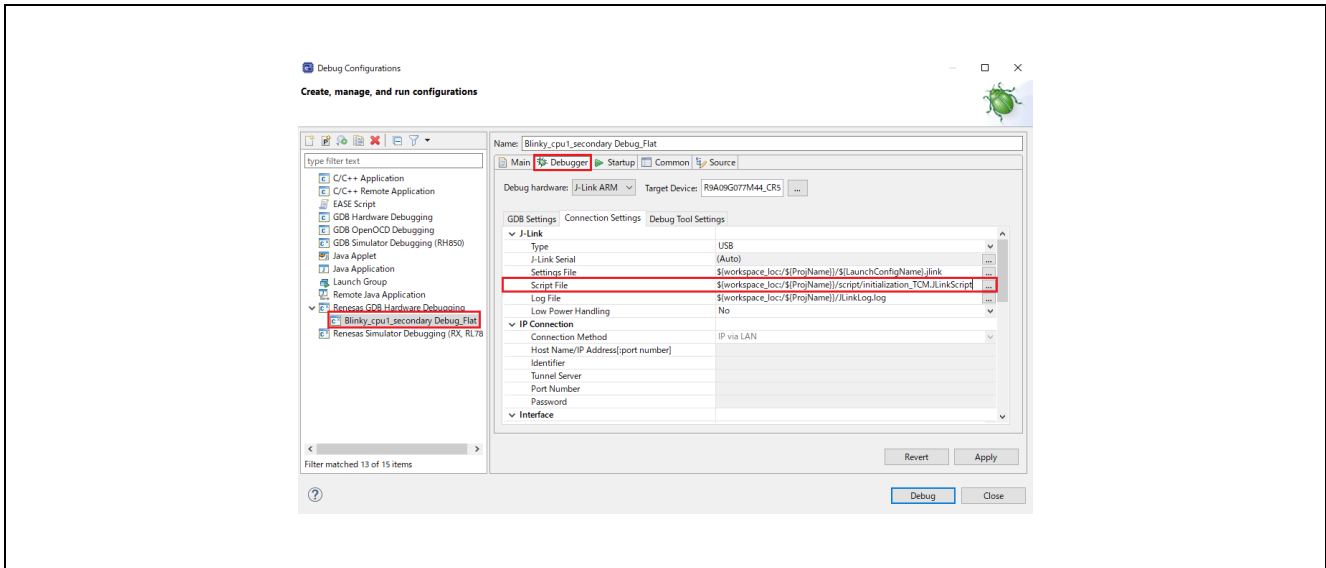


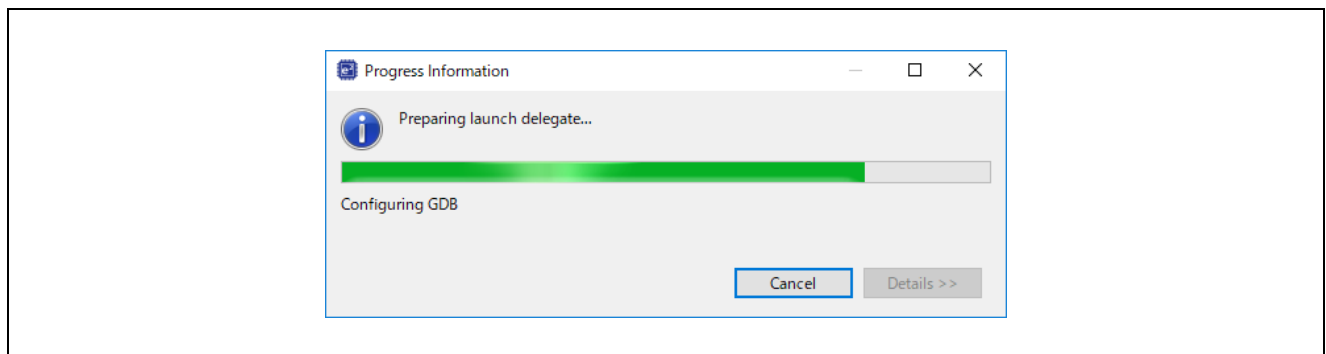
Figure 79. e<sup>2</sup> studio Debugger Configurations Window with Blinky Project (CA55)

5. (RZ/T2H CR52 CPU1 and RZ/N2H CR52 CPU1) Set the script file for TCM initialization.
  - **Debugger > Connection Settings > J-Link**
  - **Script File** : `${workspace_loc}/${ProjName}}/script/initialization_TCM.JLinkScript`



**Figure 80. e<sup>2</sup> studio Debugger Configurations Window with Blinky Project (RZ/T2H CR52 CPU1 and RZ/N2H CR52 CPU1)**

6. Click **Debug** to begin debugging the application.



**Figure 81. Start Debugging**

### 4.5.3 Details about the Debug Process

In debug mode, e<sup>2</sup> studio executes the following tasks:

1. Downloading the application image to the microprocessor and programming the image to the internal memory.
2. Setting a breakpoint at **main()**.
3. Setting the stack pointer register to the stack.
4. Loading the program counter register with the address of the **system\_init()**.
5. Displaying the startup code where the program counter points to.

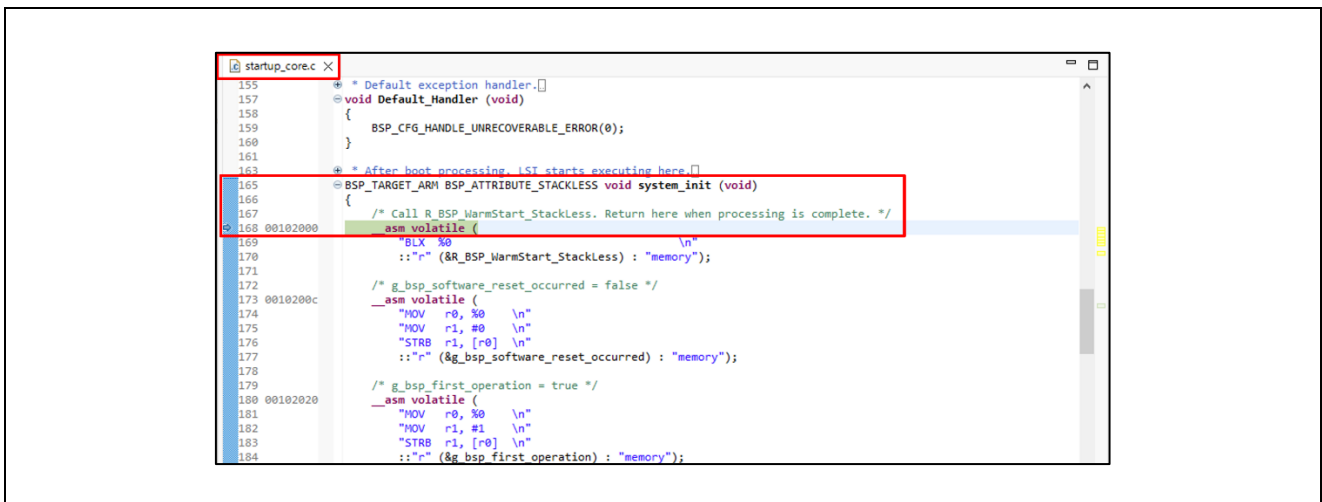


Figure 82. e<sup>2</sup> studio Debugger Memory Window

## 4.6 Run the Blinky Project

While in Debug mode, click **Run > Resume** or click on the **Play** icon twice.

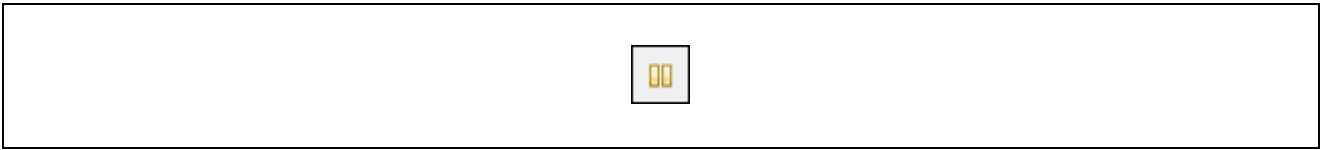


Figure 83. e<sup>2</sup> studio Debugger Play Icon

The following LEDs on the board should now be blinking.

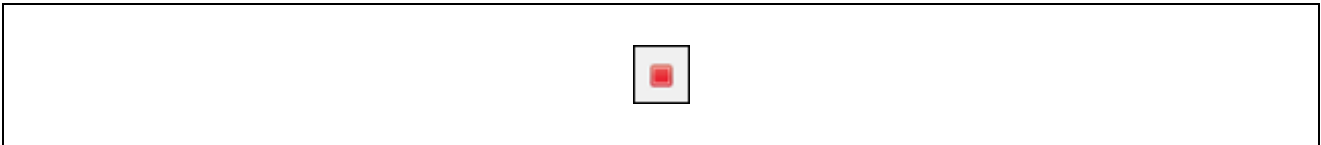
- RZ/T series
  - RSK+RZ/T2M: LED0-1 (CPU0), LED2-3 (CPU1)
  - RSK+RZ/T2L: LED0-6 (including LEDx\_ESC\_xxx)
  - RSK+RZ/T2ME: LED0-1 (CPU0), LED2-3 (CPU1)
  - RZ/T2H Evaluation Board: LED0 (CR52 CPU0), LED1 (CR52 CPU1), LED2 (CA55 Core0)
- RZ/N series
  - RSK+RZ/N2L: LED0-3
  - RZ/N2H Evaluation Board: LED3 (CR52 CPU0), LED4 (CR52 CPU1), LED8 (CA55 Core0)

To suspend program execution, click **Run > Suspend** or click on the **Pause** icon.



**Figure 84. e<sup>2</sup> studio Debugger Pause Icon**

To exit Debug mode and disconnect from the debugger, click **Run > Terminate** or click on the **Stop** icon.

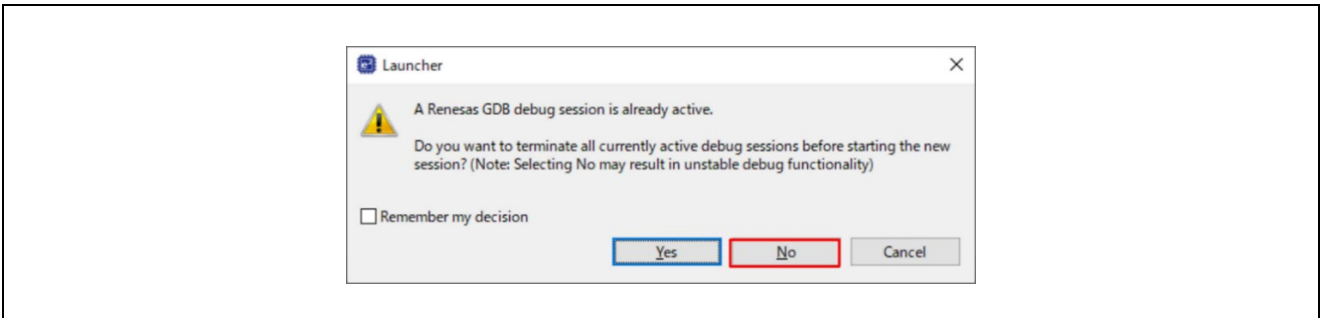


**Figure 85. e<sup>2</sup> studio Debugger Stop Icon**

## 4.7 Debug and Run for Multiprocessing

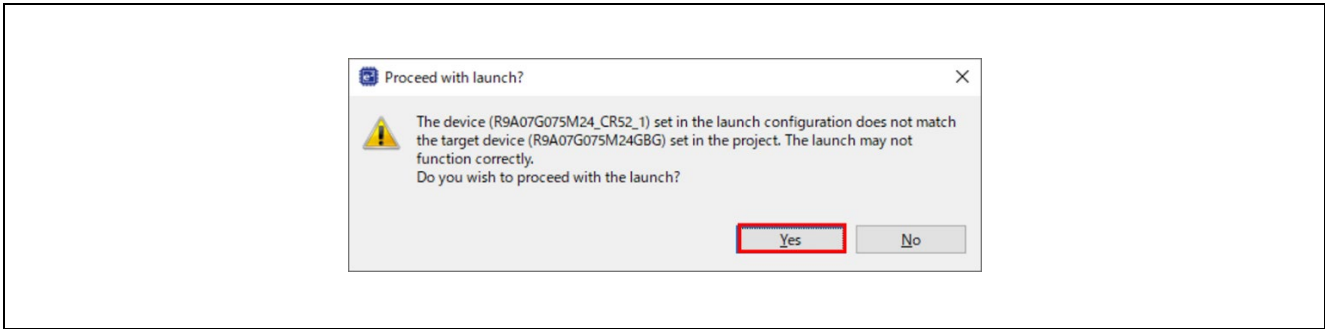
To debug the Blinky application, follow these steps:

1. Connect the debugger with the primary project using the procedure in 4.5.2 Debug Steps .
2. The primary project stays connected, and connects the debugger with the secondary project using the procedure in 4.5.2 Debug Steps.
  - (The secondary project uses CA55 Core0) In addition to the content in No. 3 of 4.5.2 Debug Steps 4.5.2. You also need to apply the following settings.
    - **Debugger > Connection Settings > Connection**
      - **Reset at the beginning of connection: No**
3. When the following dialog box is shown, please click **No** to start debugging.



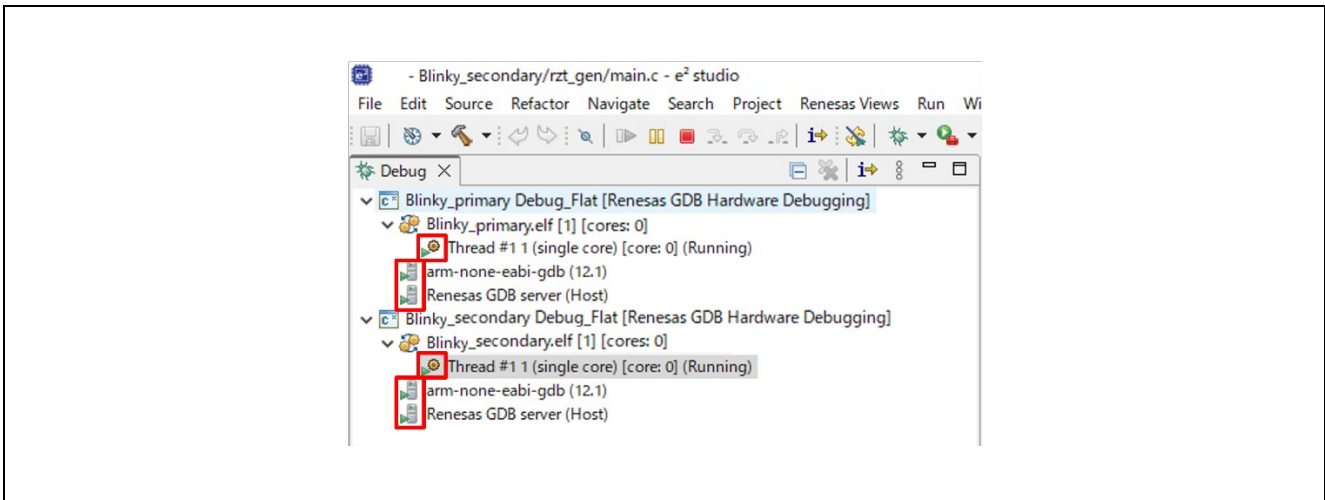
**Figure 86. Warning Window of Starting Debug Session**

4. When Figure 87 is shown, please click **Yes** to proceed with the launch.

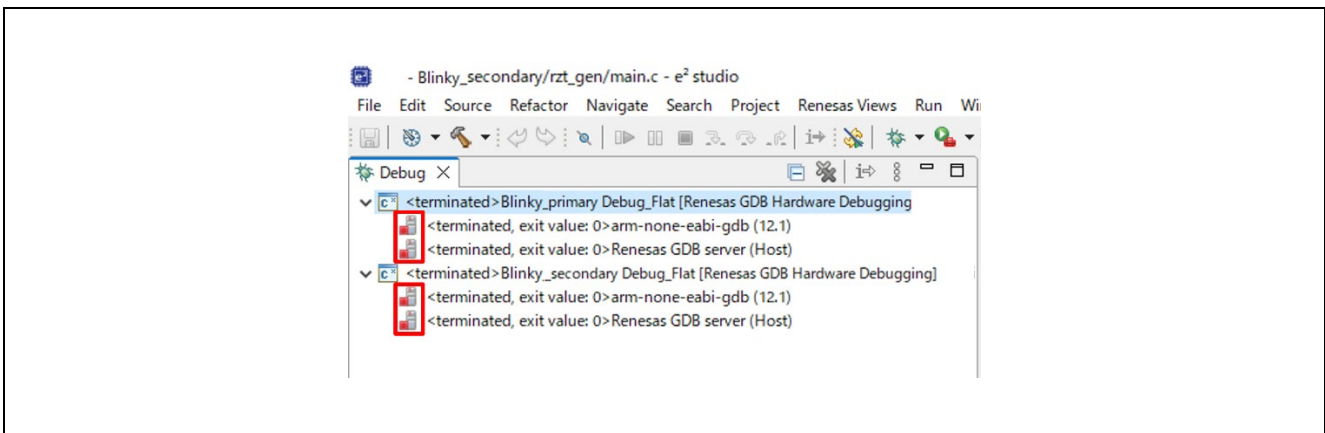


**Figure 87. Warning Window of Device Name**

5. Run the primary project with the procedure 4.6 Run the Blinky Project to copy the binaries of the secondary and subsequent projects to the internal RAM in the primary project. After the primary project reaches **hal\_entry** in **main.c**, other cores are executed. If the LEDs are blinking, proceed to the next step.
6. Run the secondary project with the procedure 4.6 Run the Blinky Project.
7. To exit Debug mode and disconnect from the debugger, terminate both projects, the primary and the secondary.



**Figure 88. During Program Execution**



**Figure 89. Terminated Program**

## 4.8 Import the Project

The project created, built, and debugged in chapters 4.3 through 4.7 can be imported and run in other workspaces.

**Note:**

Apply the same version of the FSP package used for the project to the other workspace.

To import the projects, follow these steps:

1. Click **File > Import**.

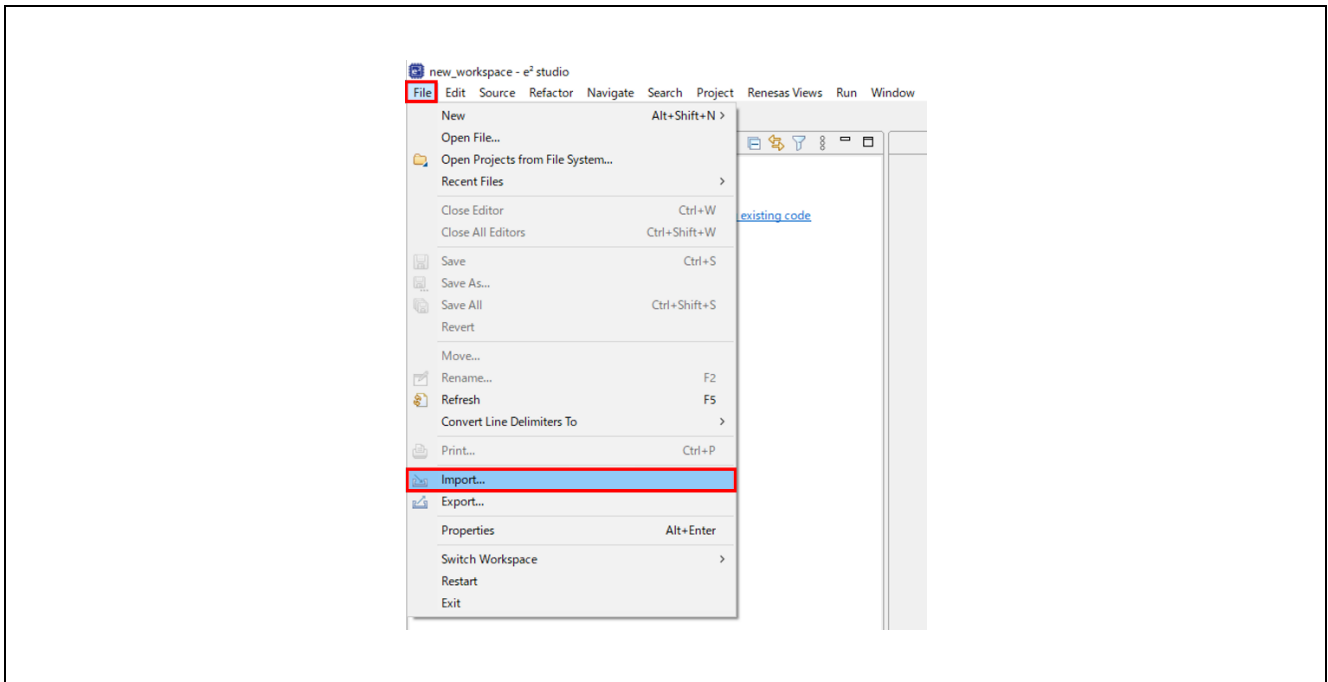


Figure 90. e<sup>2</sup> studio Import

2. Click **General > Existing Projects into Workspace**.

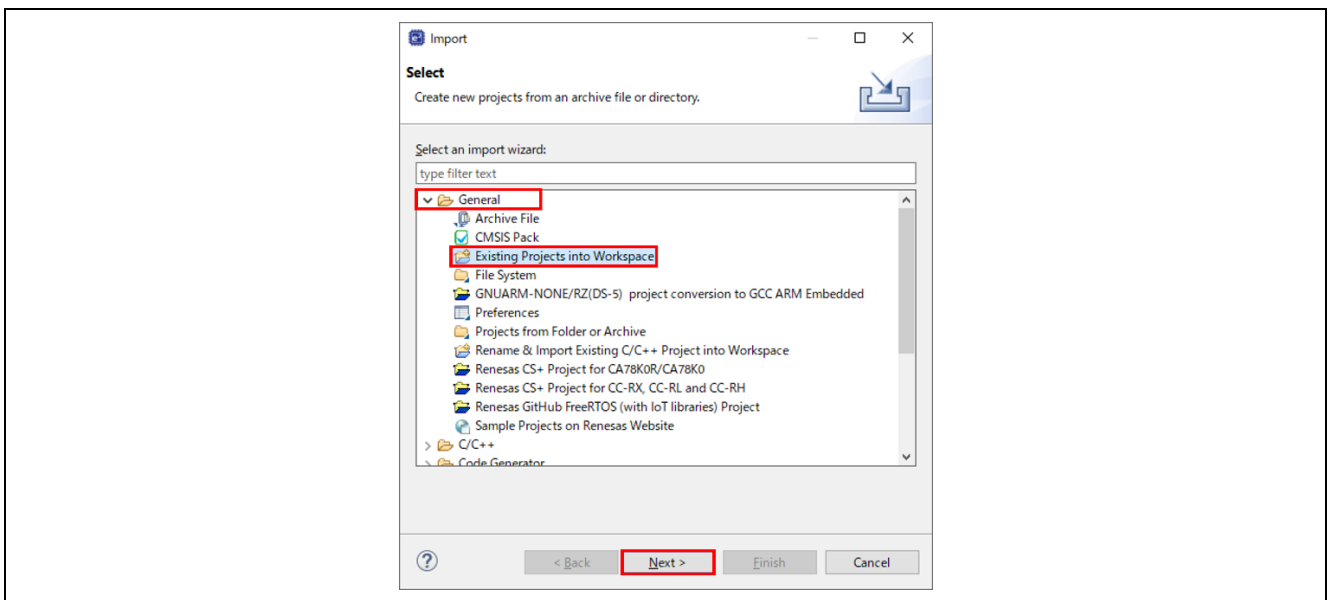


Figure 91. e<sup>2</sup> studio Select Import Type

3. **Select the root directory or select the archive file** where the project you would like to import into the other workspace resides.
4. Select projects to import into **Projects**. When using **Select root directory**, it is recommended to set **Copy projects into workspace** in **Options** to avoid updating the same project from multiple workspaces.

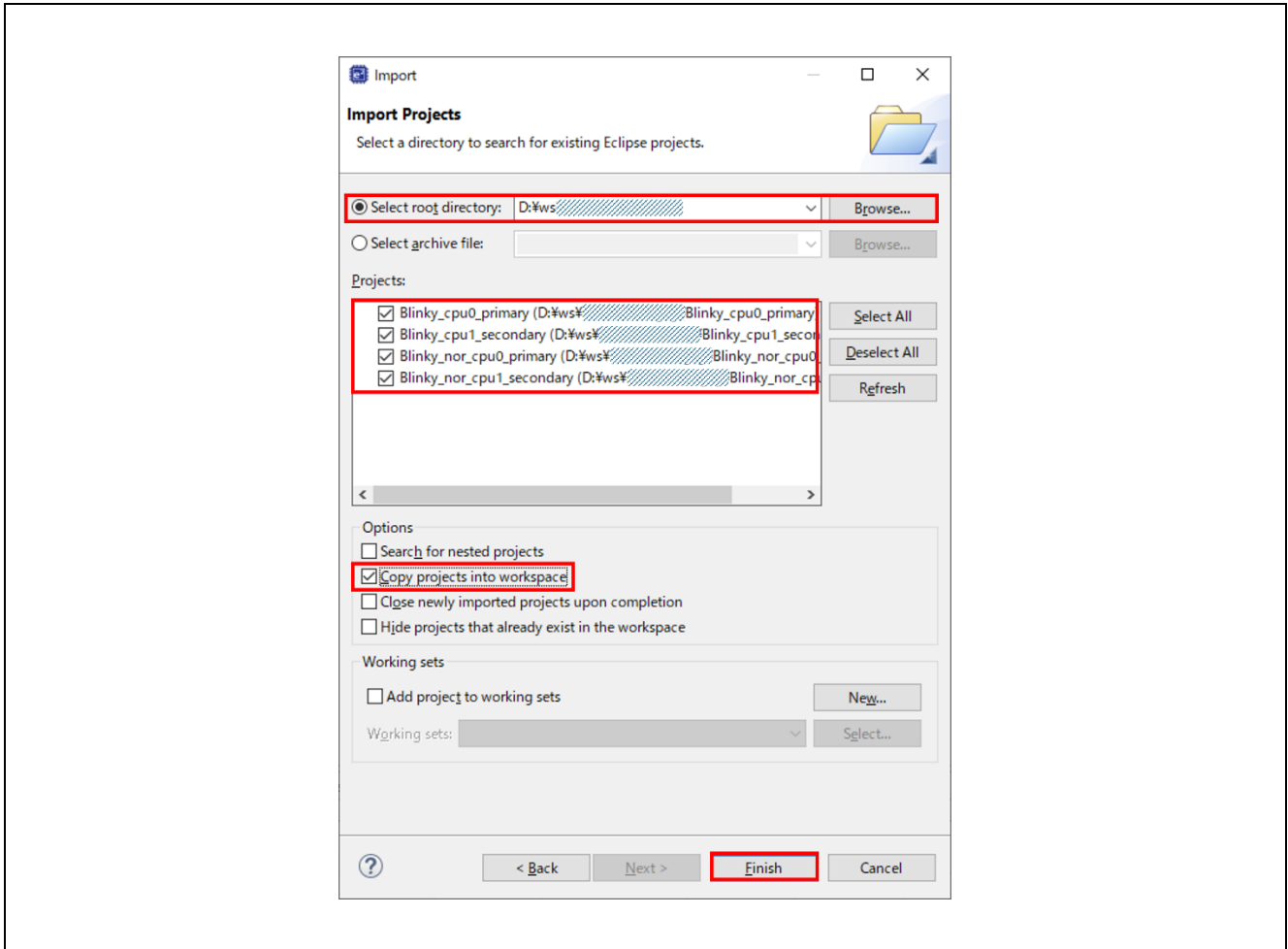


Figure 92. e<sup>2</sup> studio Select Root Directory to Import Project

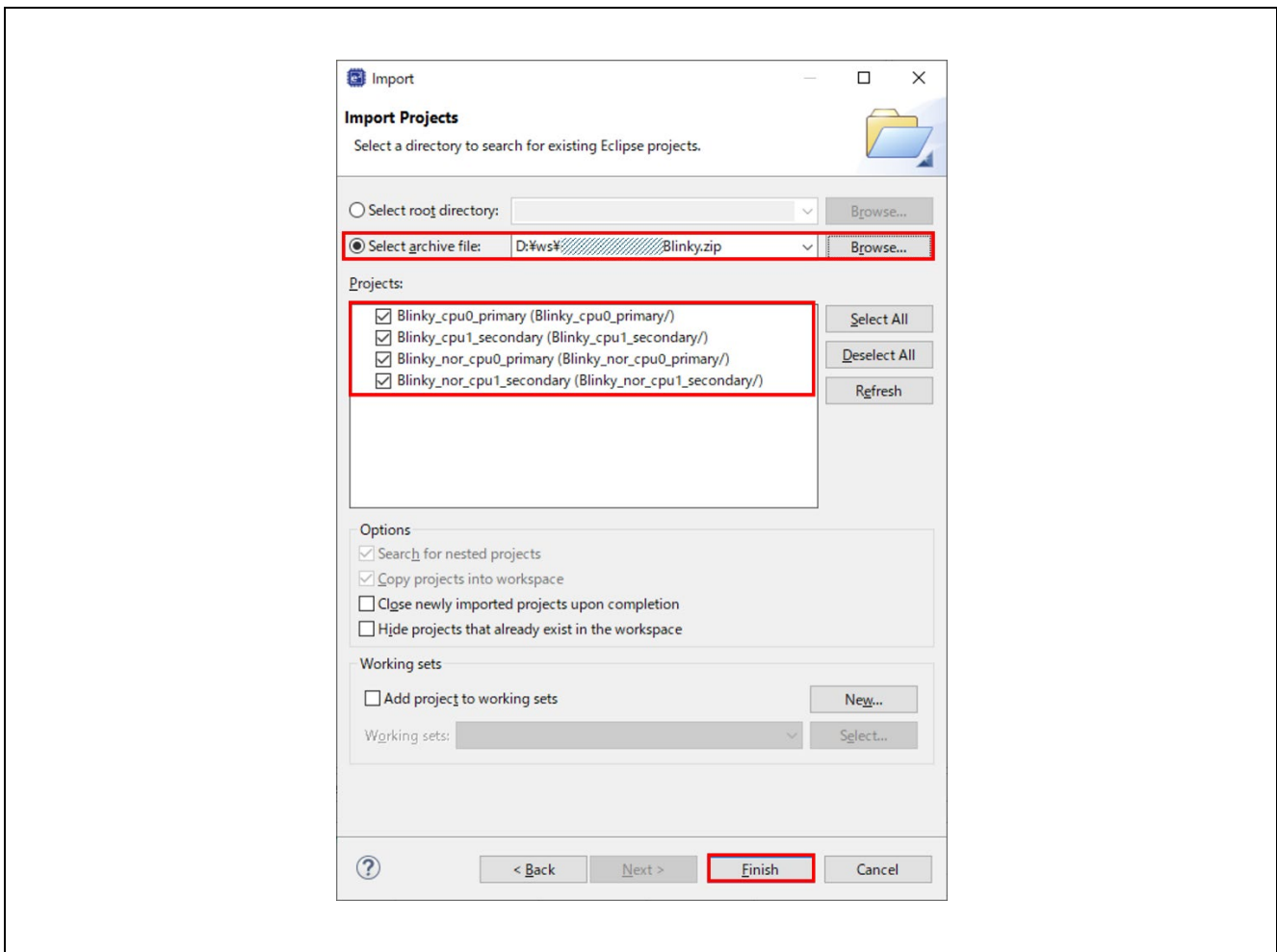


Figure 93. e<sup>2</sup> studio Select Archive File to Import Project

- The projects have been imported into the other workspace.

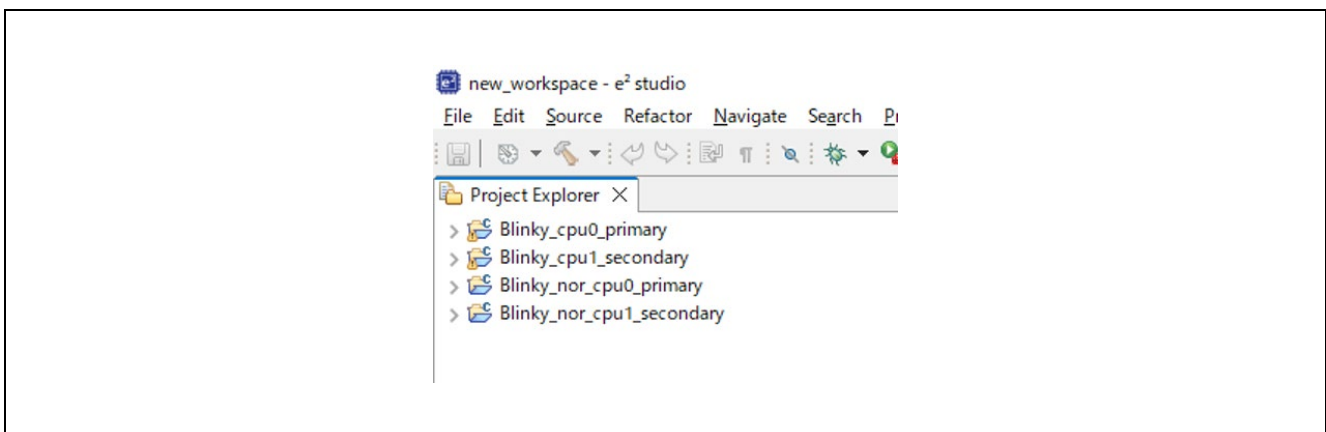


Figure 94. e<sup>2</sup> studio New Workspace

**Note:** After importing the project, the user needs to click the **Generate Project Content** button in the FSP configuration.

## 5. FSP SC User Guide

### 5.1 What is FSP SC?

The Renesas FSP Smart Configurator (FSP SC) is a desktop application designed to configure device hardware, such as clock setup and pin assignment, as well as initialization of FSP software components when using a 3<sup>rd</sup>-party IDE and toolchain.

For creating the RZ/T2 and RZ/N2 projects, the FSP SC can currently be used with

- IAR EWARM with IAR toolchain for Arm

Projects can be configured, and the project content generated in the same way as in e<sup>2</sup> studio. Please refer to 5.2 Configuring a Project section for more details.

### 5.2 Tutorial Blinky

The goal of this tutorial is to quickly get acquainted with the Flexible Platform by moving through the steps of creating a simple application using FSP SC and 3<sup>rd</sup>-party IDE and running that application on an RZ/T2, RZ/N2 MPU board. This chapter guides you through creating projects for single-core processing and multiprocessing with RAM execution without flash memory. In this chapter, multiprocessing refers to a process in which the CR52 CPU0 core is activated first, and the second core (CR52 CPU1 or CA55 Core0) operates after the CR52 CPU0 core sets up the second core.

The application used in this tutorial is Blinky, traditionally the first program run in a new embedded development environment.

Blinky is the “Hello World” of microprocessors. If the LED blinks, you know that:

- The toolchain is set up correctly and builds a working executable image for your chip.
- The debugger has been installed with working drivers and is properly connected to the board.
- The board is powered up, and its jumper and switch settings are probably correct.
- The microprocessor is alive, the clocks are running, and the memory is initialized.

### 5.3 Using FSP SC with IAR EWARM

IAR EWARM includes support for Renesas RZ/T2, RZ/N2 devices. These can be set up as bare metal designs within IAR EWARM. However, most RZ/T2, RZ/N2 developers will want to integrate RZ/T2, RZ/N2 FSP drivers and middleware into their designs. SC will facilitate this.

FSP SC generates a “Project Connection” file that can be loaded directly into IAR EWARM to update project files.

#### 5.3.1 Prerequisites

- IAR EWARM installed and licensed.
  - Please refer to the IAR Systems website regarding IAR EWARM.
- FSP SC and FSP Pack installed.
  - Please refer to Renesas website regarding FSP SC and FSP Pack.

#### Note for RZ/T2ME:

If you use the IAR EWARM 9.60.1 or 9.60.2 to debug the RZ/T2ME FSP project, please apply the following patch file.

- EWARM\_Patch\_for\_RZT2ME (EWARM\_Patch\_for\_RZT2ME\_rev1.0.zip)
  - This patch file is available at <http://www.renesas.com/rzt2me>.

To apply the patch, please read the README in the patch.

#### Note for RZ/T2H and RZ/N2H:

If you use the IAR EWARM to debug the RZ/T2H and RZ/N2H FSP project, please apply the following patch file.

- EWARM\_Patch\_for\_RZT2H\_RZN2H (EWARM\_Patch\_for\_RZT2H\_RZN2H\_rev1.0.zip)
  - This patch file is available at <http://www.renesas.com/rzt2h> and <http://www.renesas.com/rzn2h>.

To apply the patch, please read the README in the patch.

### 5.3.2 Create a New Project

The following steps are required to create a project using IAR EWARM, FSP SC, and FSP. The procedure from creating a project to running it varies depending on the number of cores used and the boot mode. This chapter shows only some of the cases where RAM execution is used. Refer to Table 13 Project Creation Procedure (IAR EWARM, FSP SC) to find out which steps are required for your application.

**Table 13. Project Creation Procedure (IAR EWARM, FSP SC)**

Step	Single-core processing		Multiprocessing	
	RAM execution	Flash boot mode	RAM execution (Combination of (CR52 CPU0, CPU1) and (CR52 CPU0, CA55 Core0) only)	RAM execution (Other combinations) Flash boot mode
Check tool limitations	Appendix Tool Software Limitations			
Erase flash memory(if needed)	Appendix How to Erase Flash Memory			
Create a project	5.3.2 Create a New Project	5.3.2 Create a New Project Appendix. How to Debug the FSP Project with Flash Boot Mode	5.3.2 Create a New Project	Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM
Build the project	5.3.3.1 Build		5.3.3.2 Build for Multiprocessing	
Debug the project	5.3.4 Download & Debug the Project		5.3.5 Debug for Multiprocessing	
Run the project				

#### Note for Multiprocessing Projects:

In the case of multiprocessing, two projects with different settings must be created. A project that starts first is called the primary project, and the second project that runs after releasing reset by the primary project is called the secondary project.

The primary project and the secondary project should be created in the same workspace.

The secondary project should be created after the primary project is created in 5.3.2 section and done 1<sup>st</sup> build of the primary project in 5.3.3 section.

#### 1. Start the FSP SC.

- FSP SC is installed in the following path by default.
  - For RZ/T, N series, it is installed in C:\Renesas\rz\sc\_vYYYY-MM\_fsp\_vX.X\ eclipse\rasc.exe

- 2. Select the **File > New > FSP Project...**
  - This step may be unnecessary depending on the old FSP SC version.

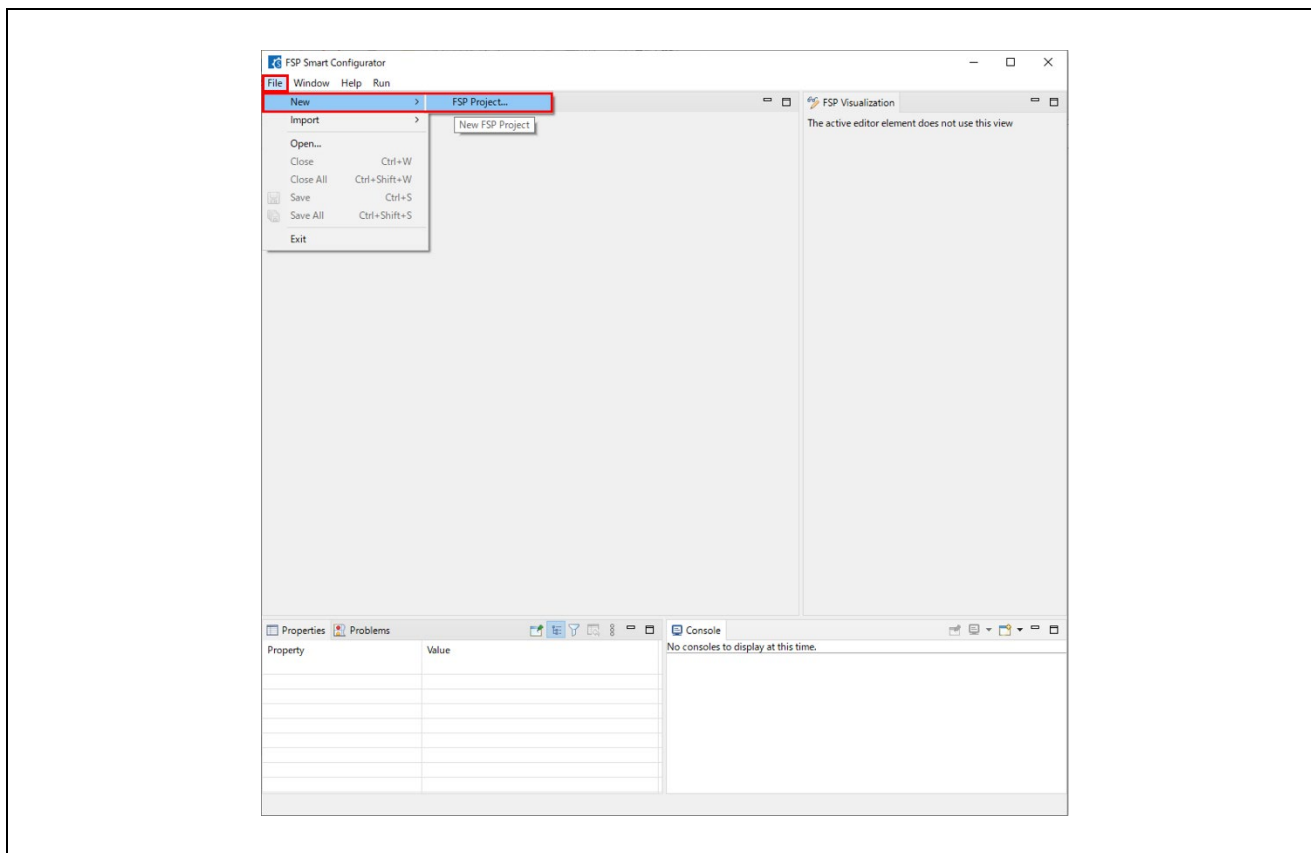
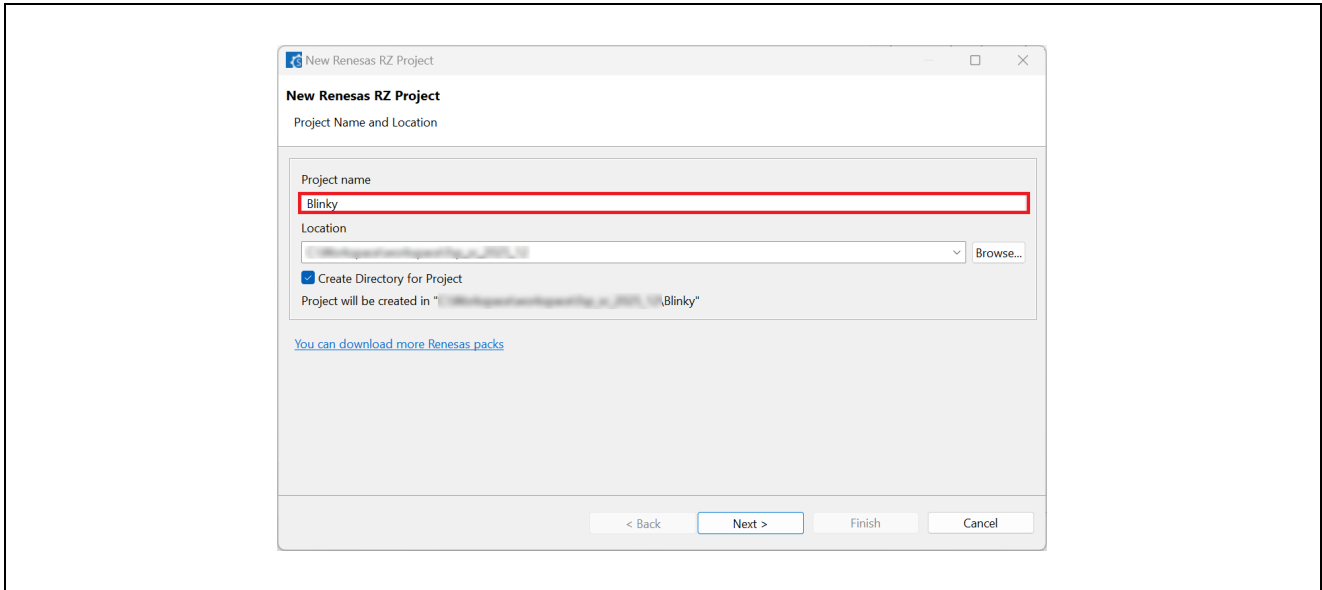


Figure 95. FSP SC New Project

3. Enter a project folder and project name. An example of naming is shown below.

**Table 14. FSP SC Newly Created Project Settings (1)**

	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Project name	Blinky	Blinky_primary	Blinky_secondary	Blinky_primary	Blinky_secondary

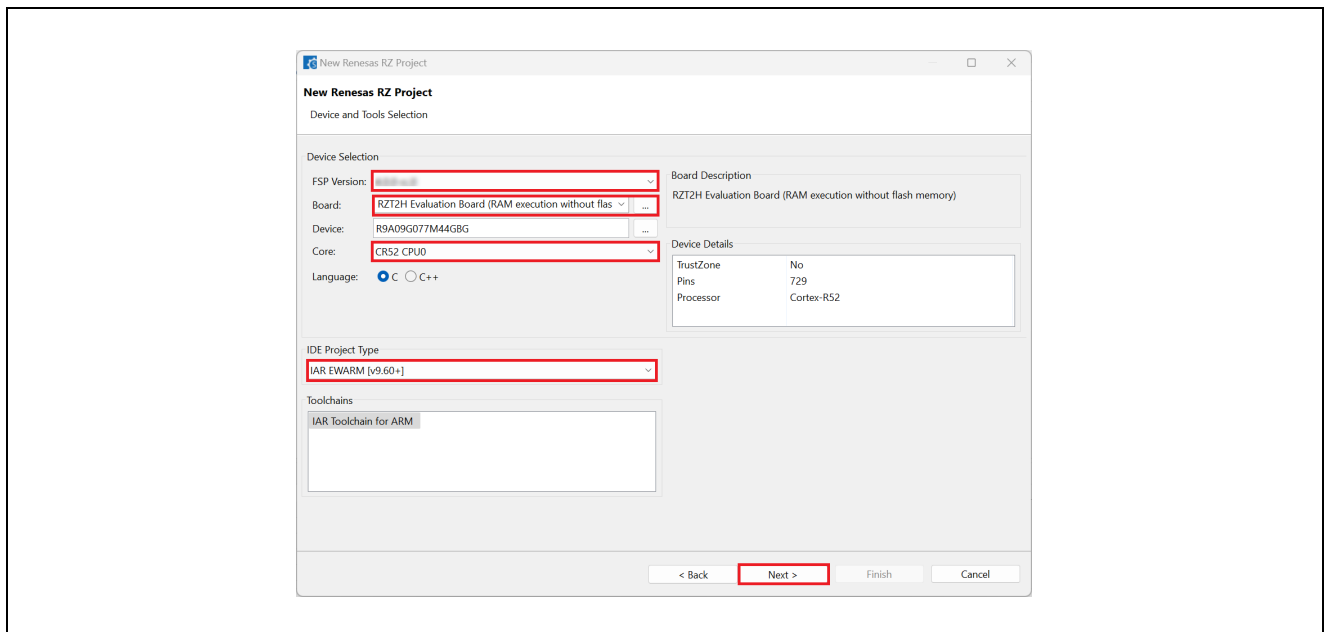


**Figure 96. FSP SC Project Settings**

4. Select the **FSP** version.
5. Select the **Board** for your application.
  - You can select an existing RZ/T2, RZ/N2 MPU Evaluation Board or select a Custom User Board for any of the RZ/T2, RZ/N2 MPU devices with your own BSP definition.
  - Here, select either of the following boards to create an FSP project for the Evaluation board.
6. (Multicore device ONLY) Select the **Core** from the drop-down list.
7. Select **IDE Project Type**.
  - As the Toolchain, the IAR Toolchain for ARM is preselected.
8. Click **Next**.

**Table 15. FSP SC Newly Created Project Settings (2)**

	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Board	RSK+RZXXX (RAM execution without flash memory) or RZXXX Evaluation Board (RAM execution without flash memory)				
Core	CR52_0 or CR52 CPU0	CR52_0 or CR52 CPU0	CR52_1 or CR52 CPU1	CR52 CPU0	CA55 Core0
IDE Project Type	<b>IAR EWARM [v9.60+]</b>				



**Figure 97. Target Device and IDE Selections**

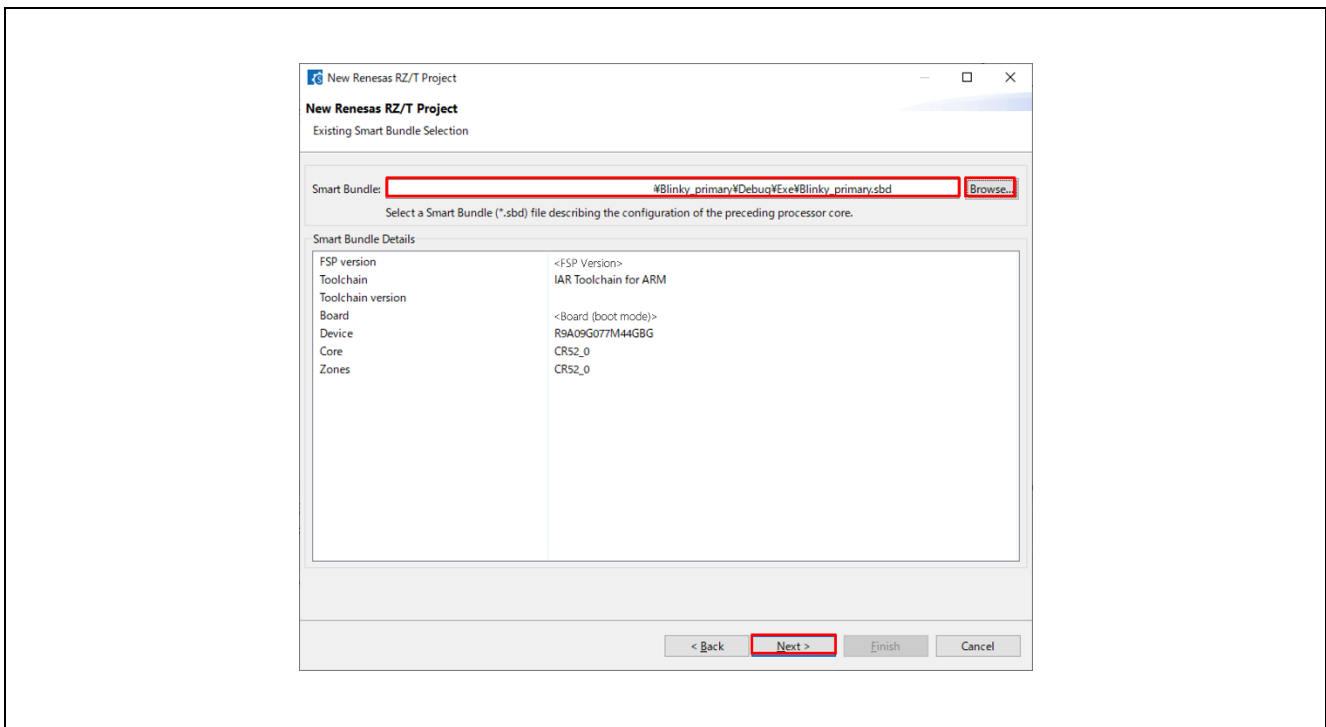
9. Select a bundle file. For the secondary project of multiprocessing, select the file of the primary project. The file is generated in the Debug/Exe directory of the project after building the project.

**Table 16. FSP SC Newly Created Project Settings (3)**

	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Use Smart Bundle	Uncheck	Uncheck	-	Uncheck	Check
Smart Bundle	-	-	.sbd file of the primary project	-	.sbd file of the primary project

**Note:**

Warnings occur if the FSP version or Board (boot mode) used is different between the primary project and the secondary project. Use the same version of the FSP and Board (boot mode).



**Figure 98. Bundle File Selection**

10. Select **RTOS**.

- Here, select **No RTOS** to proceed with the following tutorial.

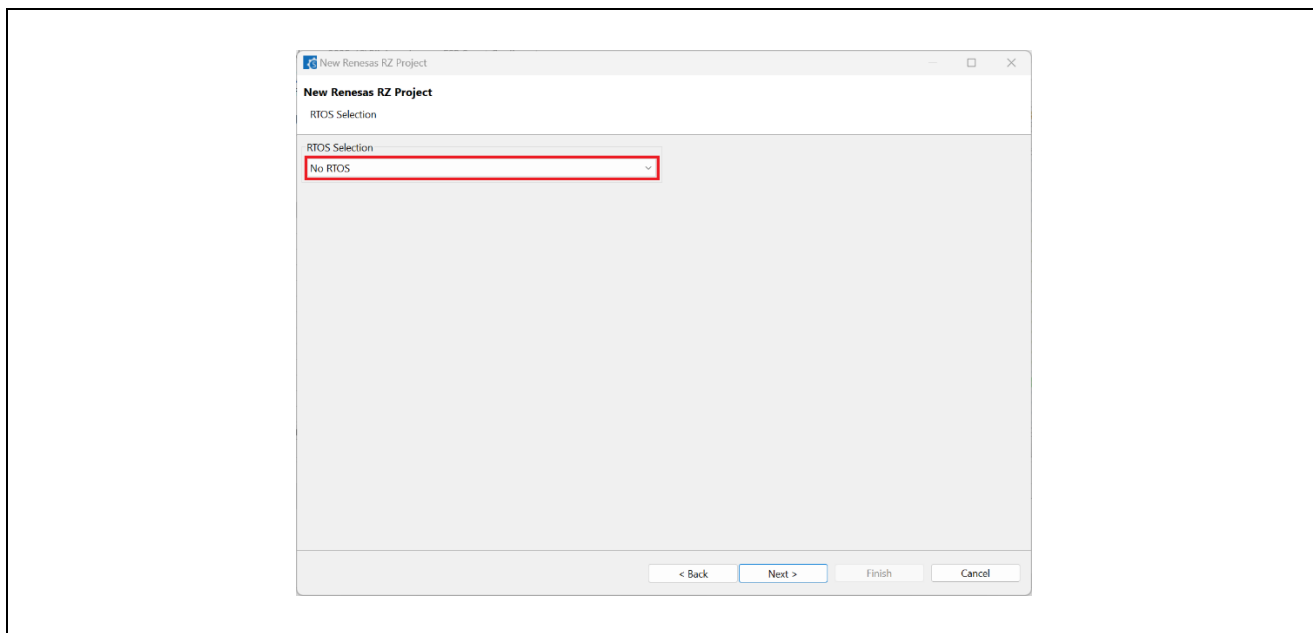
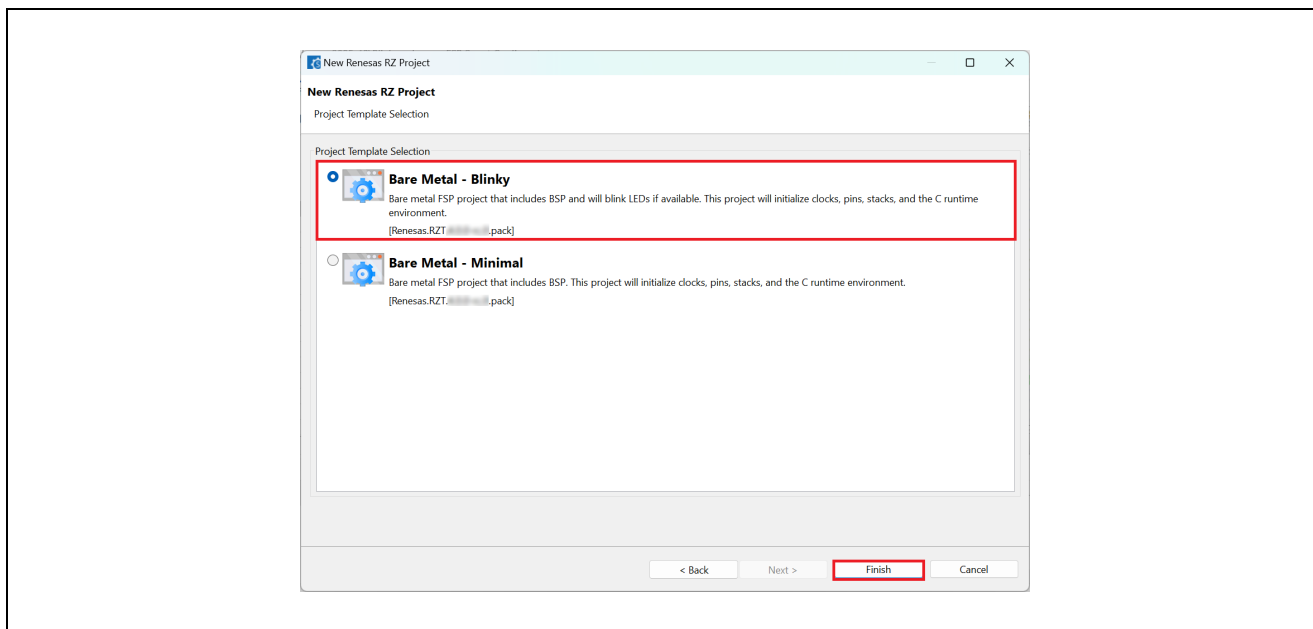
11. Click **Next**.

Figure 99. RTOS Selection

12. Select a **project template** from the list of available templates.
  - By default, this screen shows the templates that are included in your current RZ/T MPU Pack.
  - Here, select Bare Metal – Blinky to proceed with the following tutorial.
    - If you want to develop your own application, select the basic template for your board, **Bare Metal – Minimal**.
13. Click **Finish**.

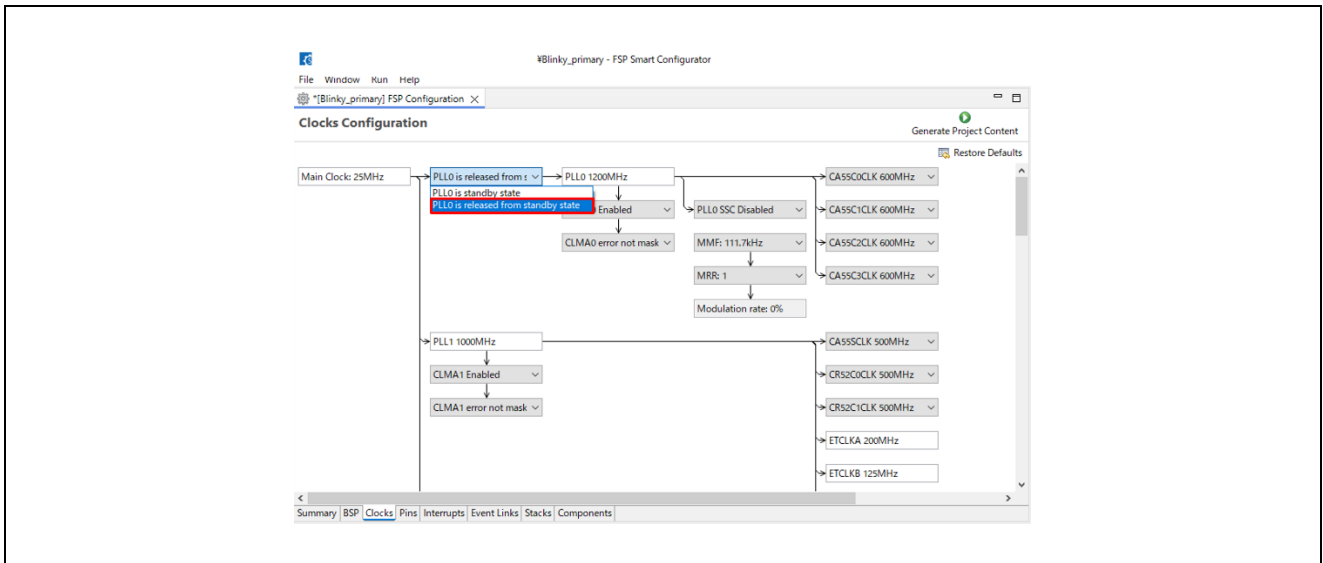


**Figure 100. Template Selection**

14. **Configure** the FSP configuration by referring to Chapter 6.3 “Configuring a Project”.
  - Here, skip this configuration step to proceed with the following tutorial.

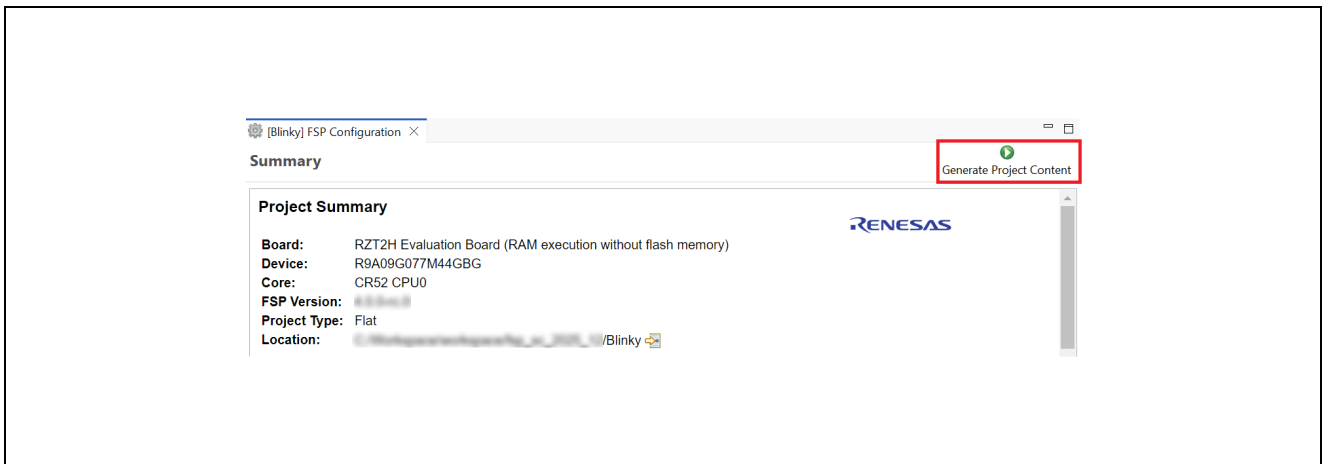
**Note for the primary project using CR52 CPU0:**

If the primary project selects CR52 CPU0 as the **Core** and the secondary or later project uses a CA55 core, you need to set "PLL0 is released from standby state" and enable PLL0 in the Clocks tab of FSP Configuration.



**Figure 101. Enable PLL0 in the Primary Project using CR52 CPU0**

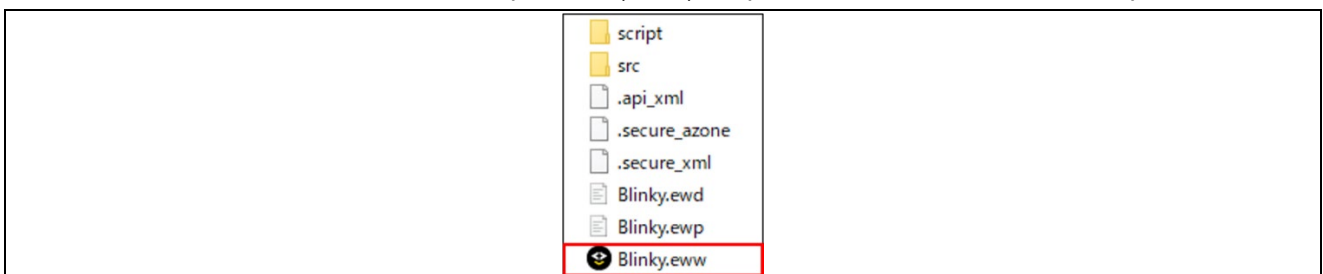
15. On completion of the FSP configuration, click **Generate Project Content**.



**Figure 102. FSP Project Configuration and Generation**

A new IAR EWARM project file will be generated in the project path.

16. Double-click the IAR EWARM Workspace file (.eww) to open IAR EWARM with the workspace.



**Figure 103. FSP Project Workspace**

### 5.3.2.1 NOTE: Configure IAR EWARM Project

1. Click on **Project** and then click on **Option...** to open the project option window.
2. Click **Debugger > Setup** and select **I-jet** of **Driver**.

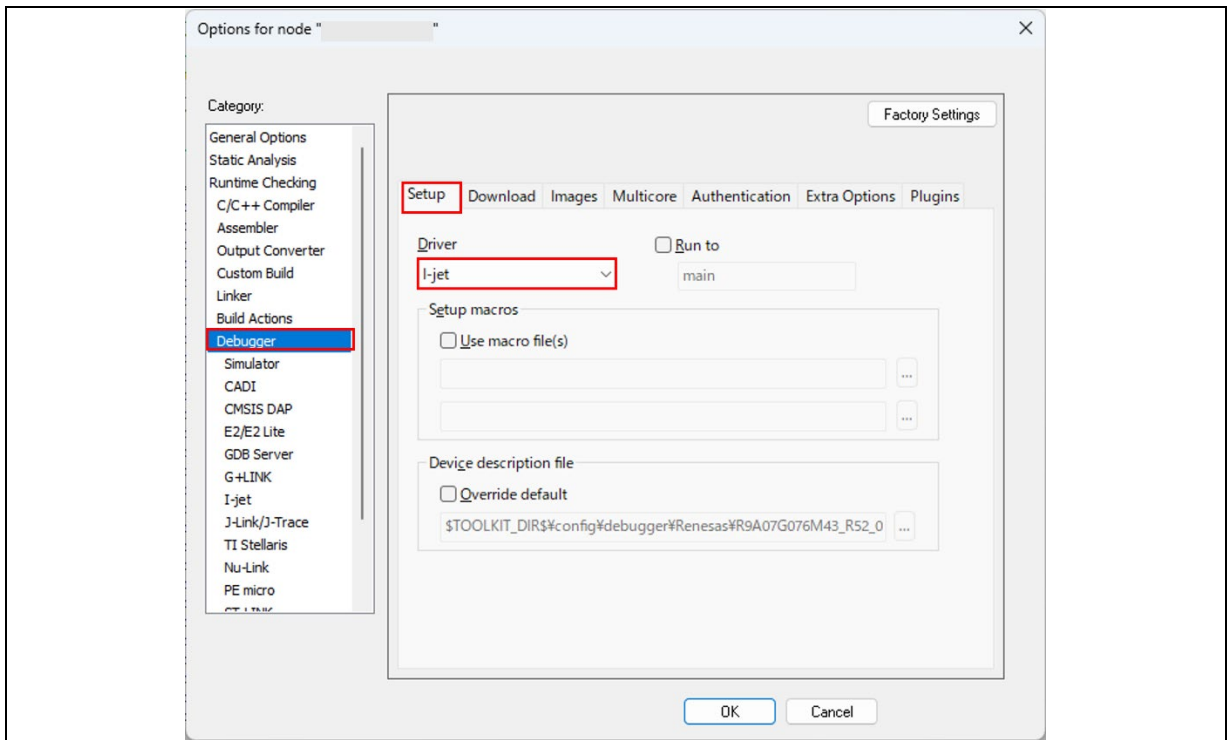


Figure 104. IAR EWARM Debugger Setup

3. Click **Linker > Extra Options** and add "--redirect Reset\_Handler=system\_init" to **Command line options (one per line)**:

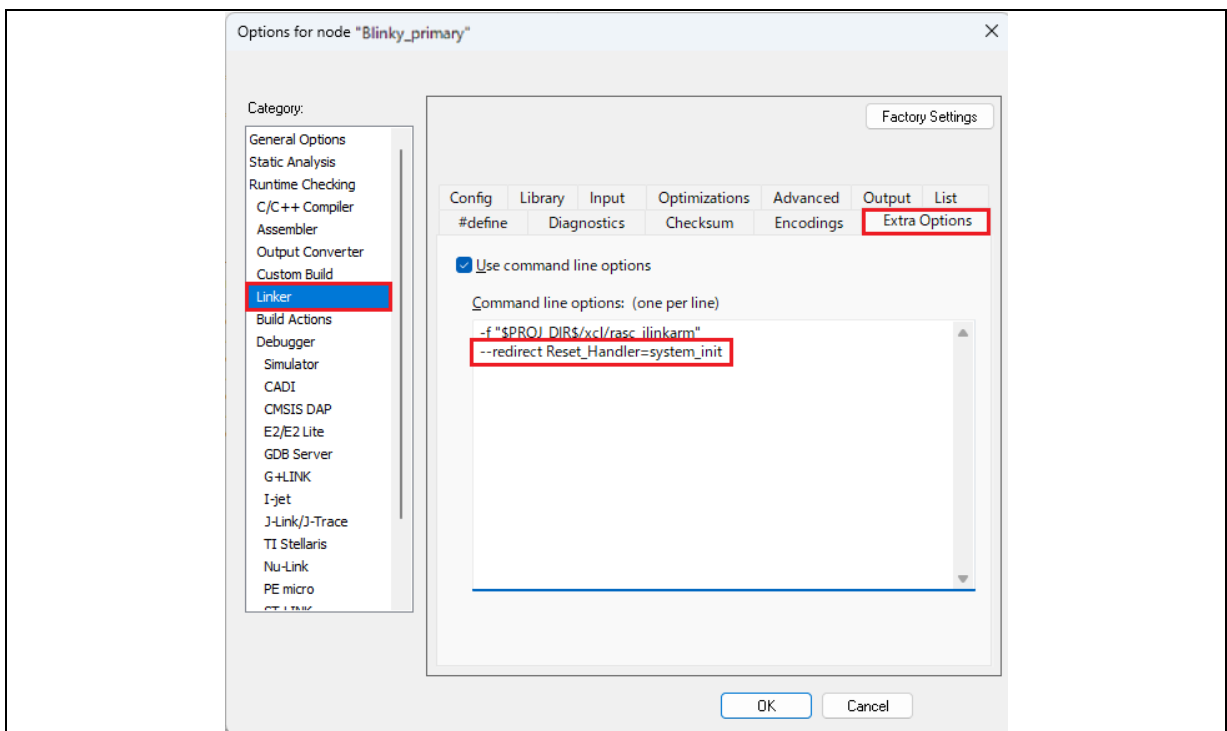


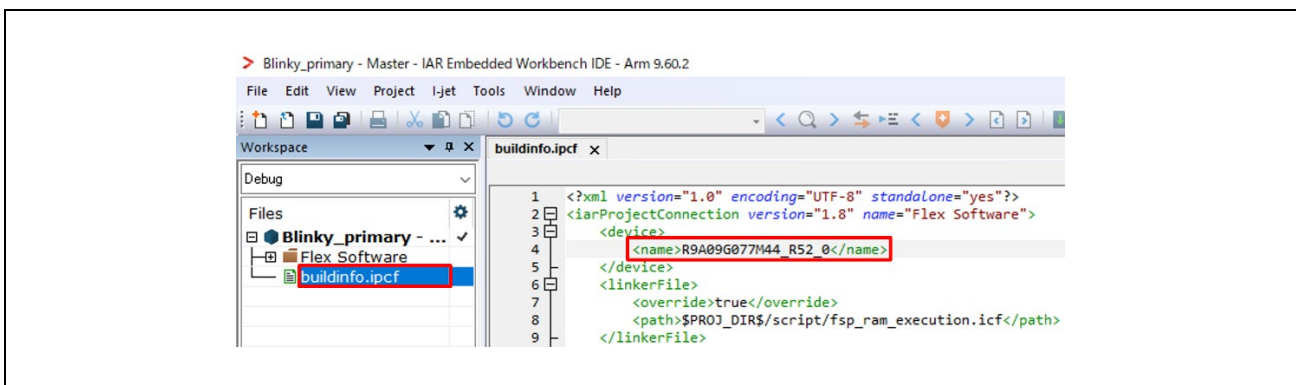
Figure 105. IAR EWARM Linker Command line options

**5.3.2.2 NOTE: Configure IAR EWARM Project [RZ/T2H and RZ/N2H Only]**

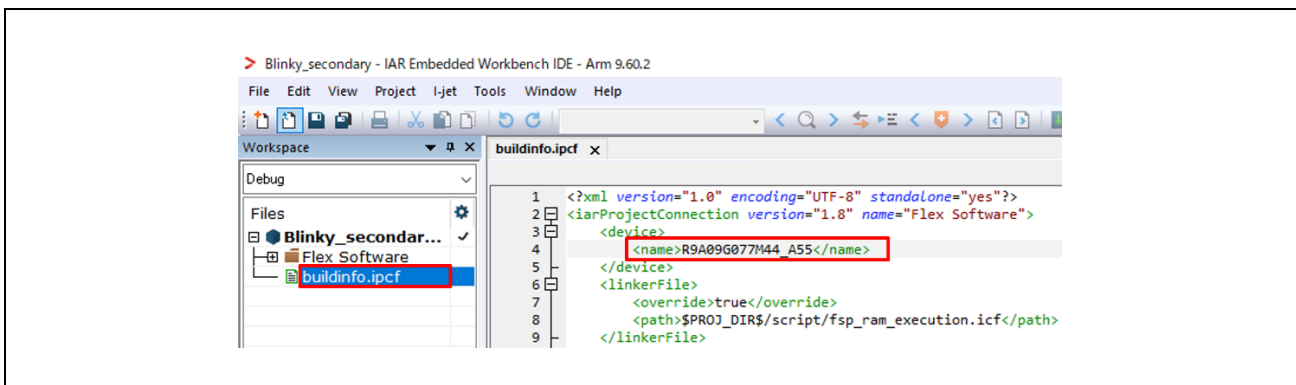
1. Change the device tag name of buildinfo.ipcf in the project and save it.

**Table 17 FSP SC Newly Created Project Debug Settings**

Device name	Single-core processing (CR52 CPU0)	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
RZ/T2H	R9A09G077M44_R52_0	R9A09G077M44_R52_0	R9A09G077M44_R52_1	R9A09G077M44_R52_0	R9A09G077M44_A55
RZ/N2H	R9A09G087M44_R52_0	R9A09G087M44_R52_0	R9A09G087M44_R52_1	R9A09G087M44_R52_0	R9A09G087M44_A55



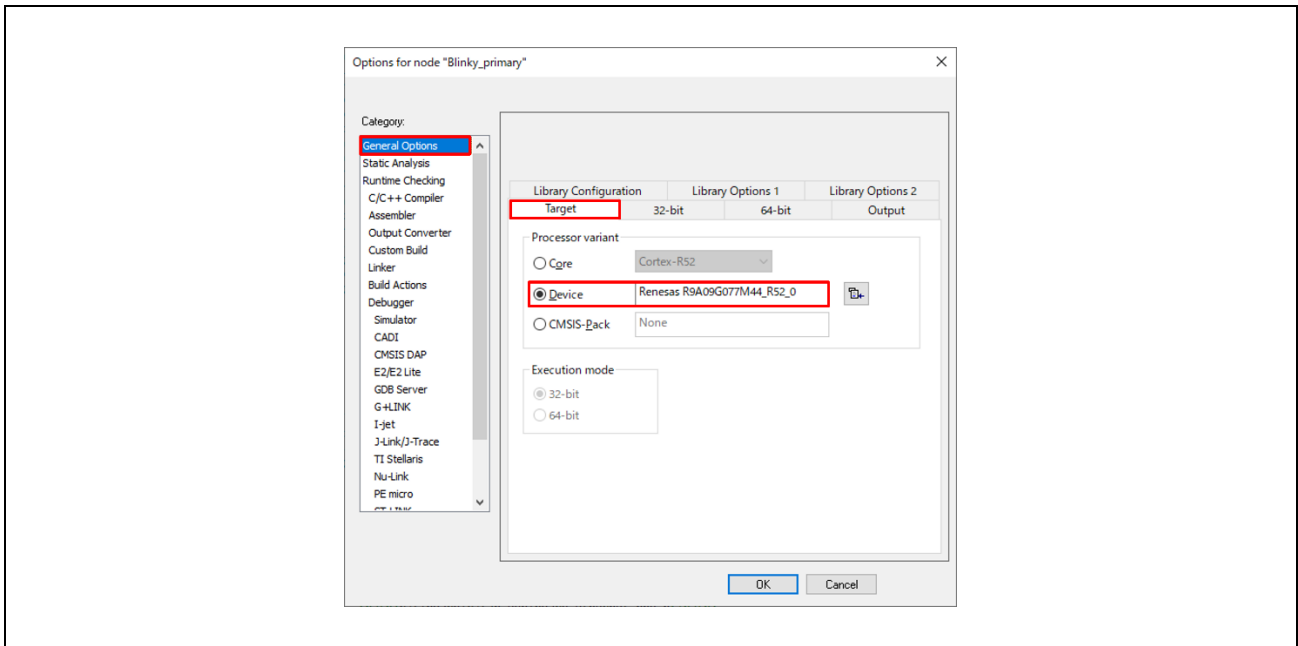
**Figure 106. IAR EWARM Project File (CR52)**



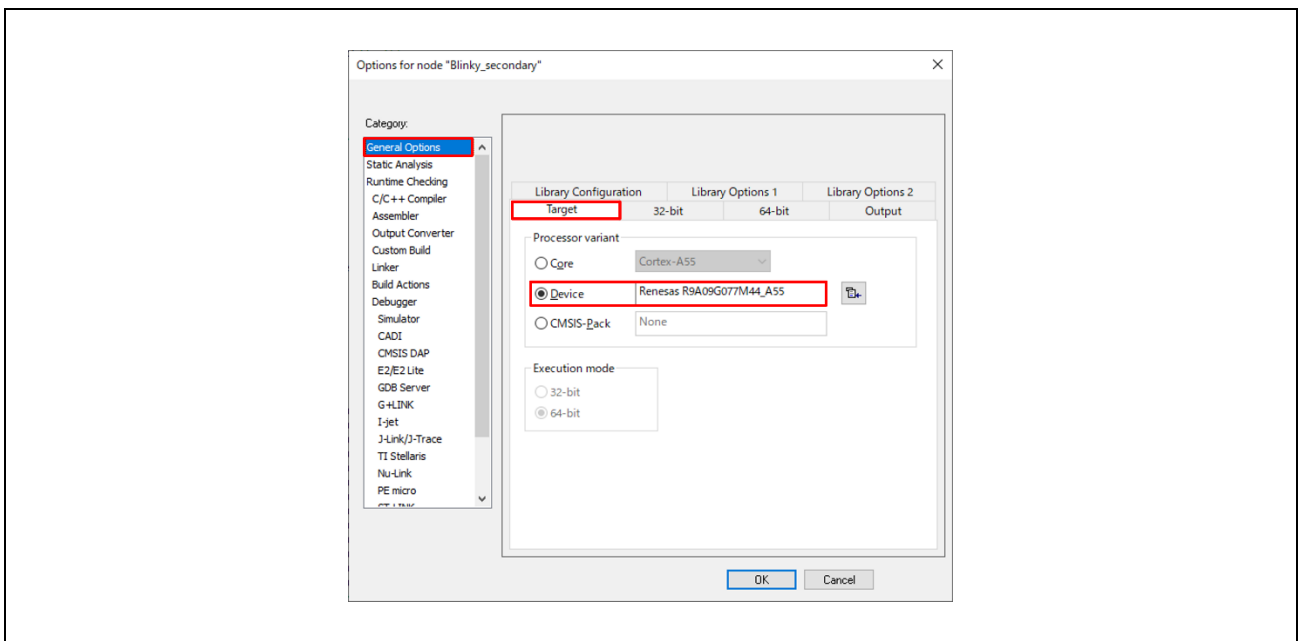
**Figure 107. IAR EWARM Project File (CA55)**

2. Click on **Project** and then click on **Option...** to open the project option window.

3. Select the **General Options** category and **Target** tab.
4. Confirm that the name changed in step 1 appears in the **device** of the **Processor variant**.



**Figure 108. Project Options – Device (CR52 CPU0)**



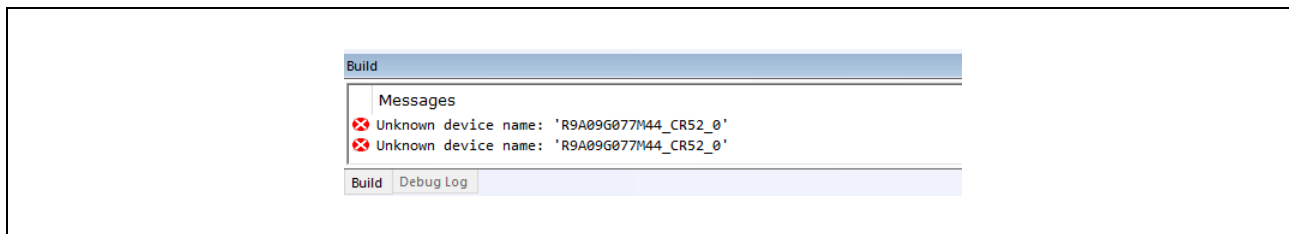
**Figure 109. Project Options – Device (CA55)**

**Note:**

If any of the following apply, the contents of buildinfo.ipcf will be overwritten, and the device name reverts to its pre-modified name.

- A project is built for the first time after creating the project.
- A project is reopened in IAR EWARM after changing the FSP configuration of it and clicking **“Generate Project Content”** in FSP SC.

This will result in an error message, but the setting of project options in step 4 is maintained and do not need to be modified again in buildinfo.ipcf. Please build the project as is.



**Figure 110. Error Message after Configuration Change**

### 5.3.3 Build the Project

When multiprocessing, please refer to Section 5.3.3.2 Build for Multiprocessing.

#### 5.3.3.1 Build

- Single-core processing  
Click on **Project** -> **Make** from the menu bar or the **Make** button on the toolbar to build.
- Multiprocessing  
Build both the primary and secondary projects.  
Click on **Project** -> **Rebuild All** from the menu bar.

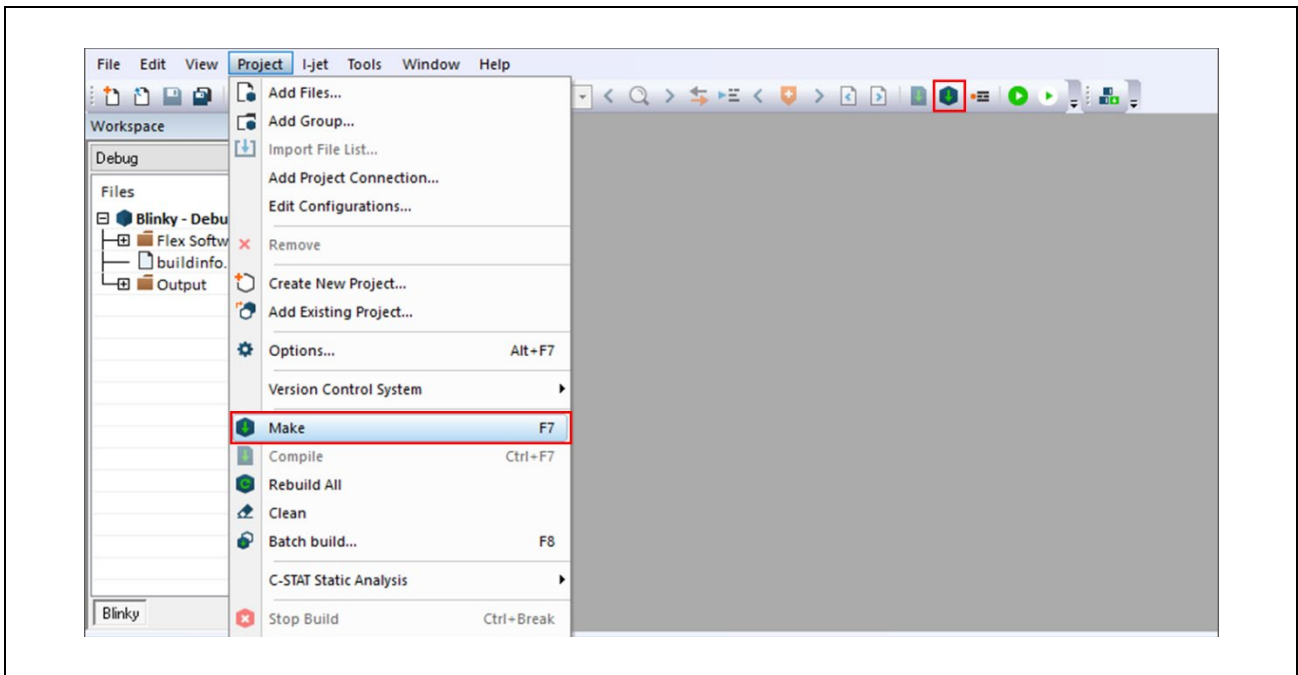


Figure 111. Make Button

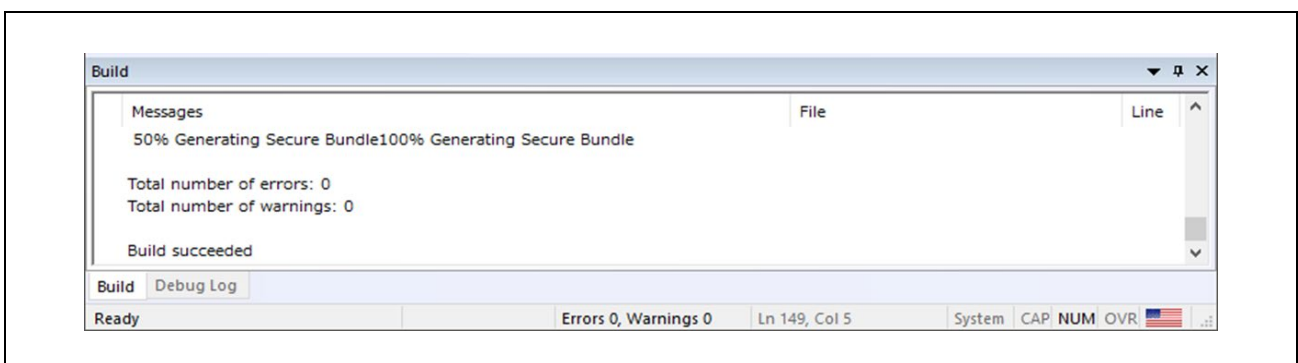


Figure 112. Build Message Console

Once the build is completed, the build message is displayed in the Build Console window, which displays compilation target files and the number of errors/warnings.

### 5.3.3.2 Build for Multiprocessing

For multiprocessing, note the build order and build settings. If the step is preceded by (XXX), it is executed only if the condition is met.

(CR52): The core used in the project is CR52.

(CA55): The core used in the project is CA55.

1. Create and build the primary project. (1st build of the primary project)
  - Set the following before building:
    - i. Click **Project > Options...**

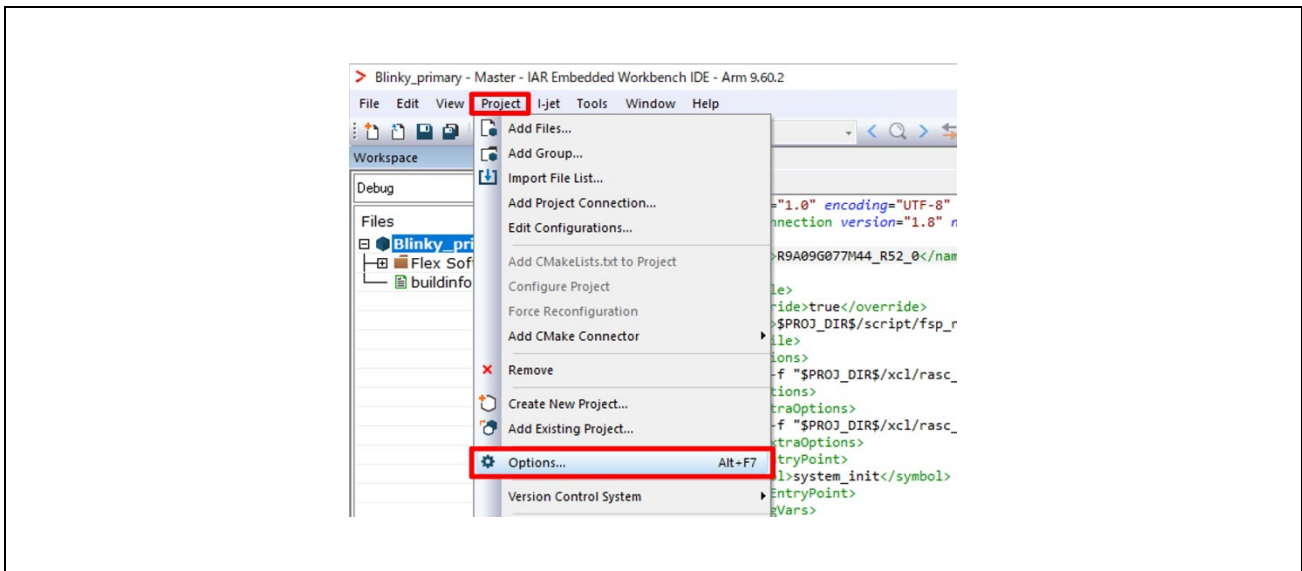


Figure 113. IAR EWARM Project Options

- ii. Click **Debugger > Setup** and uncheck "Run to".

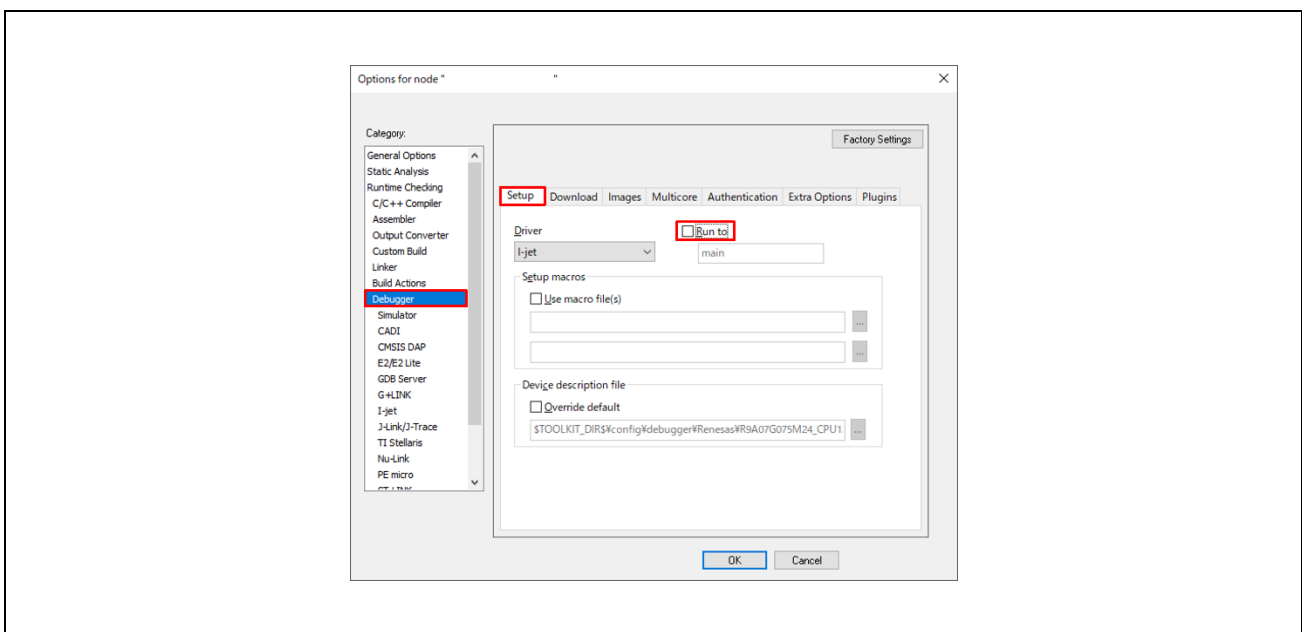


Figure 114. IAR EWARM Project Options for the Primary Project (Run to)

- iii. (CA55) Click **I-jet > Interface**, select **From file** in **Probe config**, and select the core in **CPU** of **Probe Configuration file**.

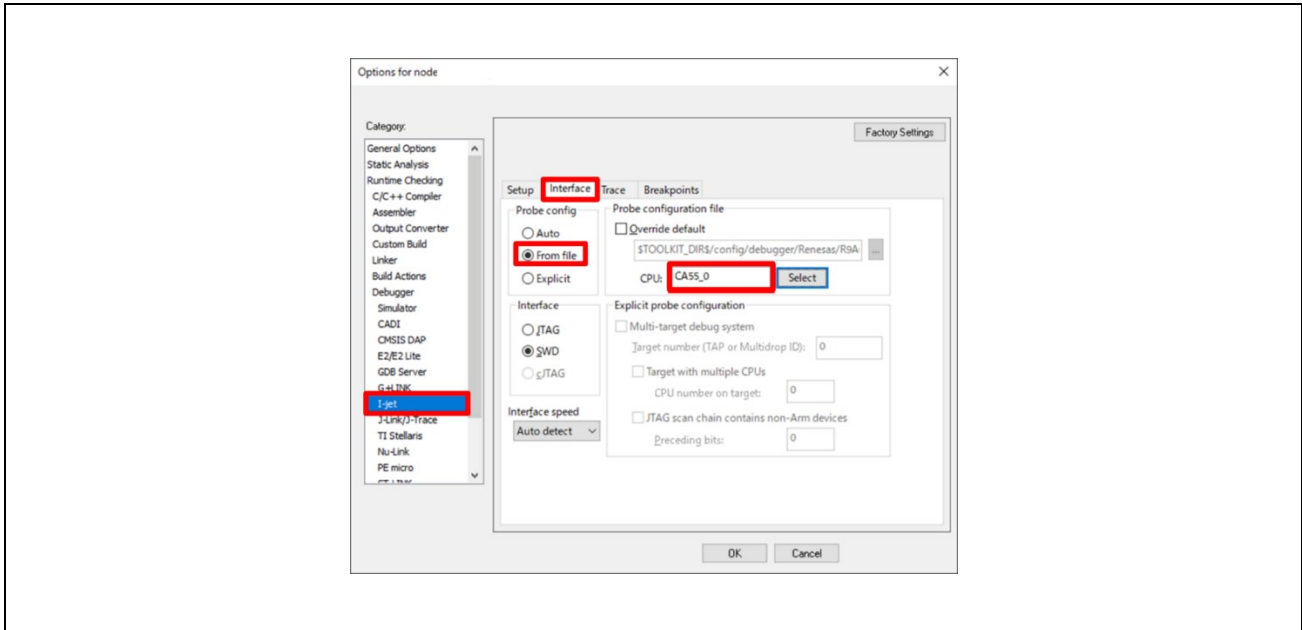


Figure 115. IAR EWARM Project Options for the Primary Project (I-jet Interface)

- iv. (CA55) Select **General Options > 64-bit** and select **LP64** of **Data model**.

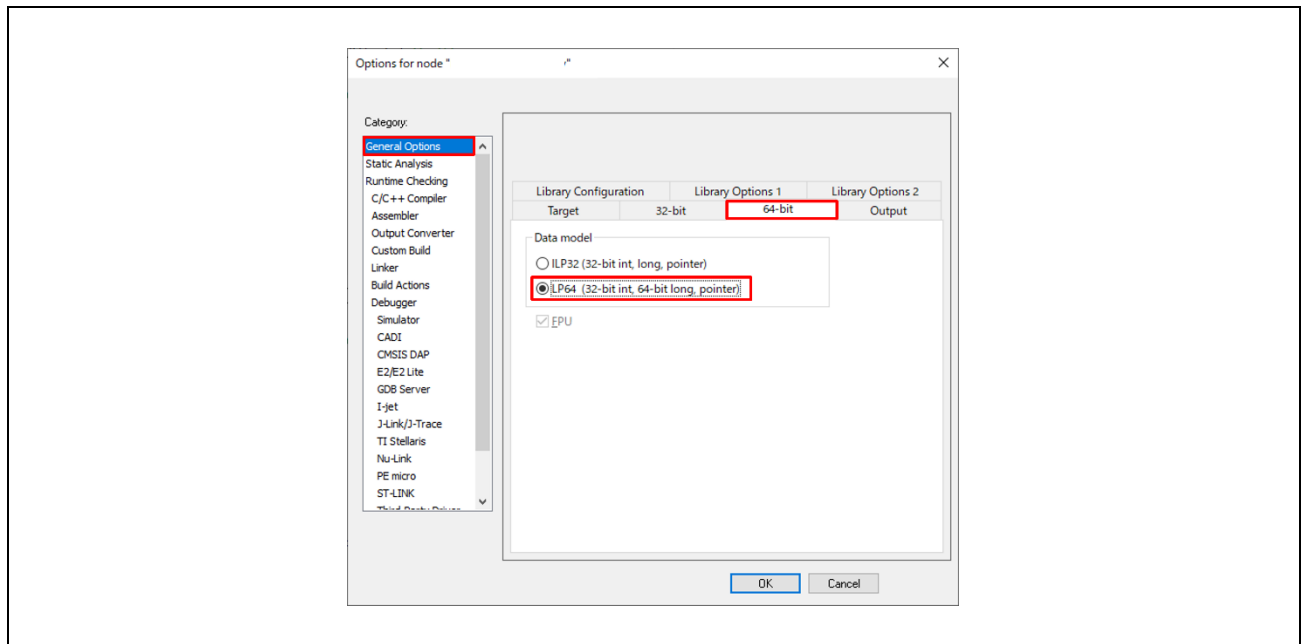


Figure 116. Project Options – Data Model

- v. Proceed to 5.3.3.1 Build.

2. Create the secondary project. Change the project options setting and build it.
  - i. Click **Project > Options...**
  - ii. Click **Debugger > Setup** and uncheck **"Run to"**.
  - iii. (RZ/T2H CR52 CPU1 and RZ/N2H CR52 CPU1) Click **Debugger > Setup**, check **Use macro file(s)** and add "\$PROJ\_DIR\$\script\initialization\_TCM.mac".

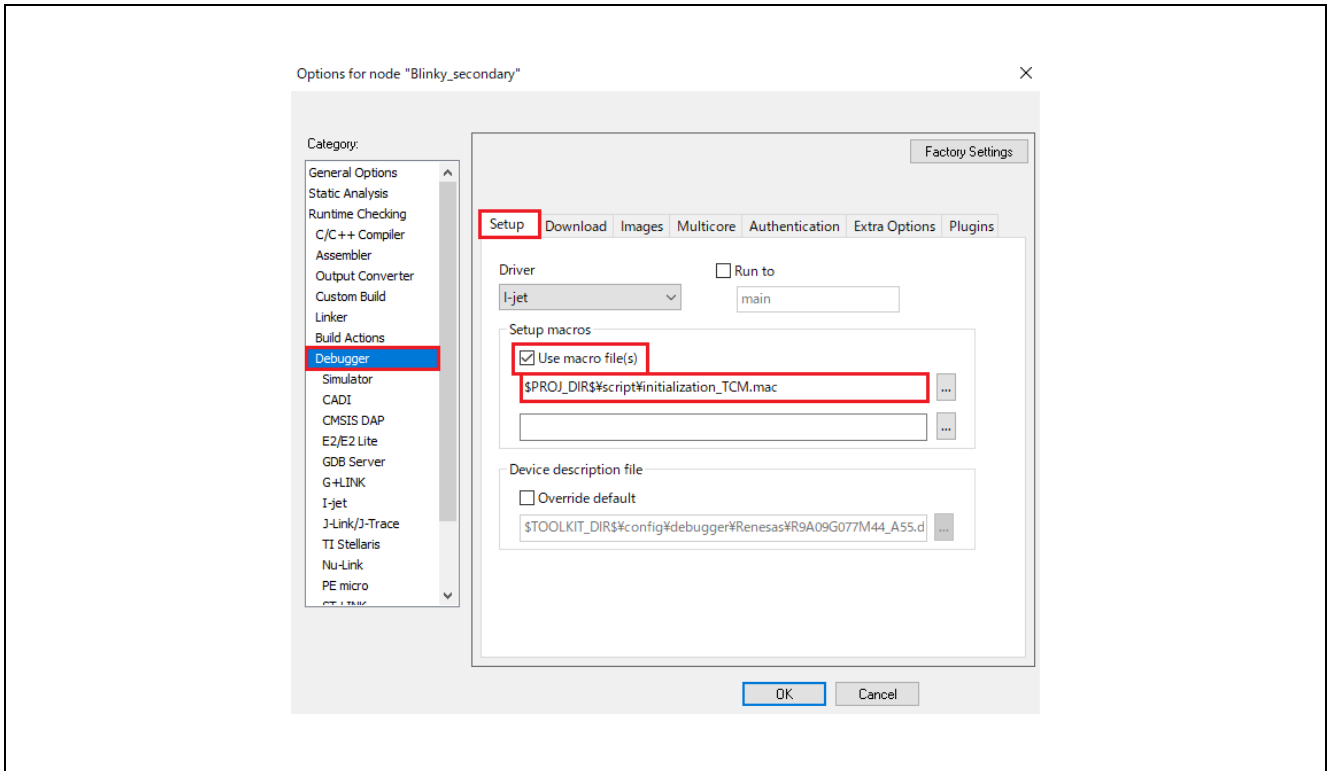


Figure 117. IAR EWARM Project Options for the Secondary Project (Setup Macros)

- iv. Click **Debugger > Extra Options** and add "--macro\_param cpu1\_enable=1" to **Command line options: (one per line)**.

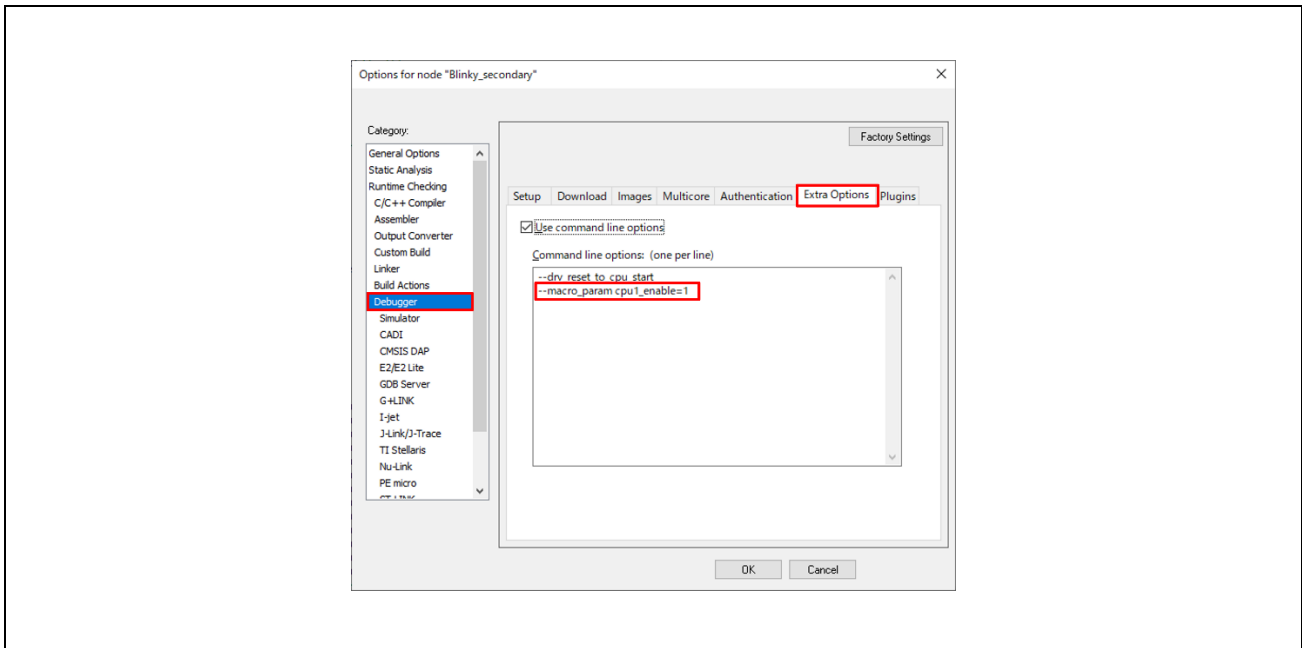


Figure 118. IAR EWARM Project Options for the Secondary Project (Debugger Extra Options)

- v. Click I-jet > Setup and select Software as Reset.

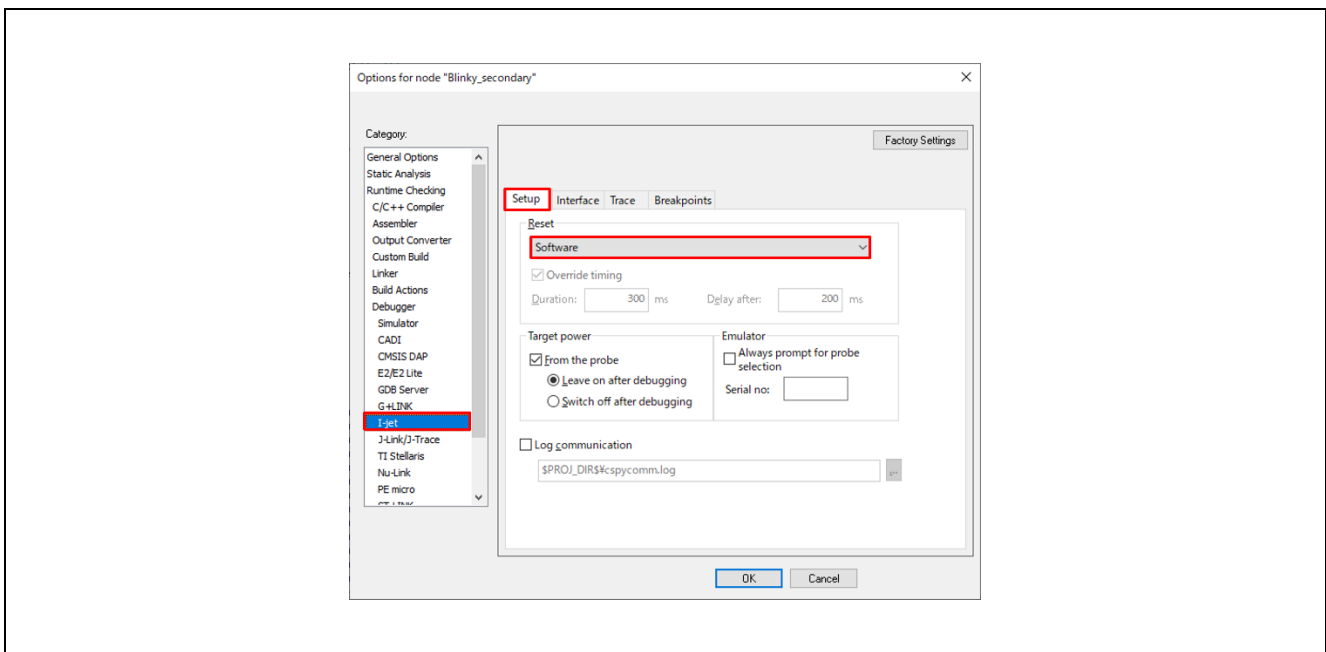


Figure 119. IAR EWARM Project Options for the Secondary Project (Reset)

- vi. (CA55) Click **I-jet > Interface**, select **From file** in **Probe config**, and select core in **CPU of Probe Configuration file**.
  - vii. (CA55) Select **General Options > 64-bit** and select **LP64** of **Data model**.
  - viii. Proceed to 5.3.3.1 Build.
  - ix. Close the secondary project.
3. Build the primary project. (2nd build of the primary project)  
No setting is required; proceed to 5.3.3.1 Build.

### 5.3.4 Download & Debug the Project

When multiprocessing, please refer to section.5.3.5 Debug for Multiprocessing

#### Note:

The main chapter of this documentation describes a RAM execution without a flash memory project. When debugging a project with flash boot mode, please also refer to Appendix. How to Debug the FSP Project with Flash Boot Mode.

Click on **Project -> Download and debug** from the menu bar or the **Download and Debug** button on the toolbar to download and debug.

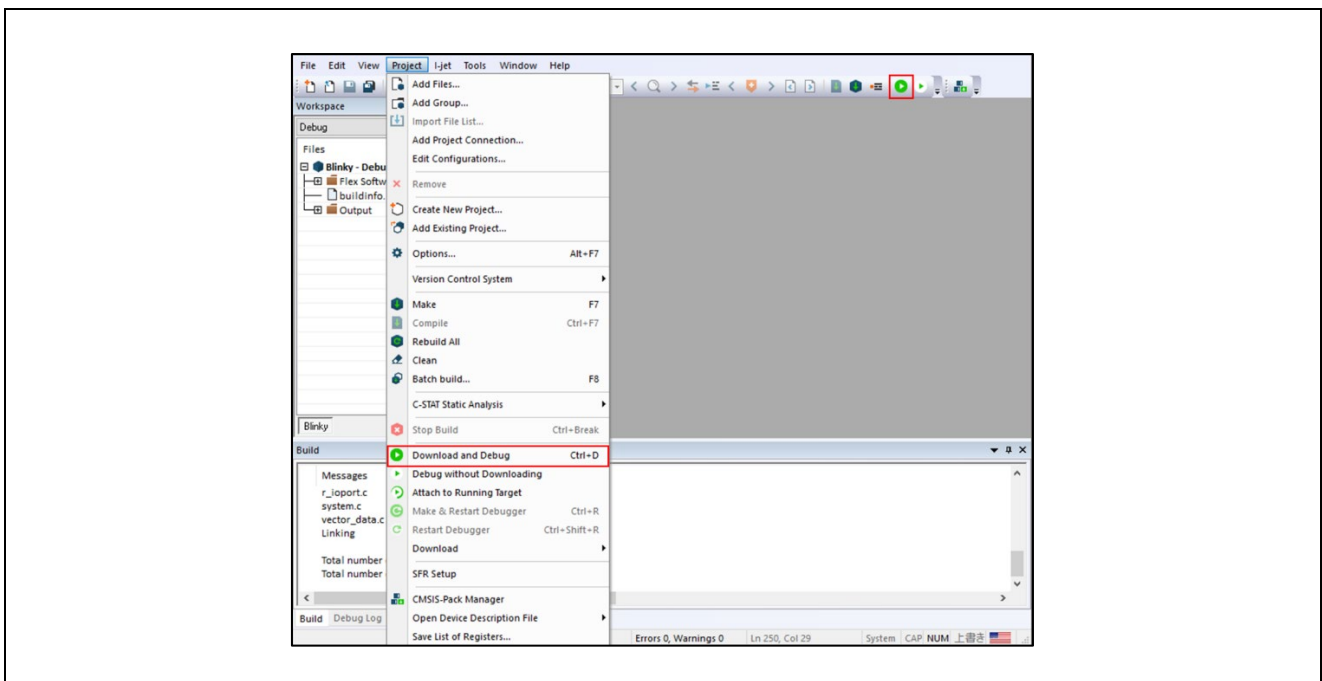


Figure 120. Download and Debug Button

Once the download is completed and the debug is started, the program breaks at the beginning of **main** in **main.c**.

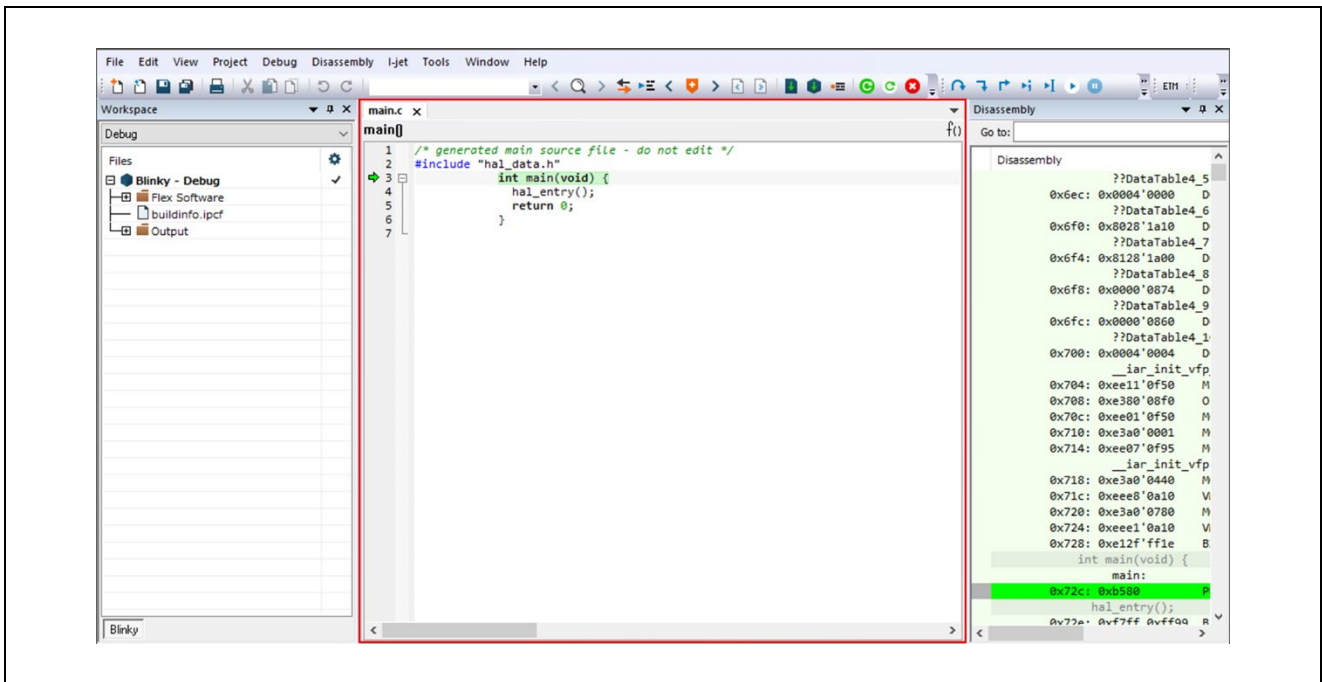


Figure 121. Starting Debug

Click on **Debug->Go** from the menu bar or the **Go** button on the toolbar to run this program.

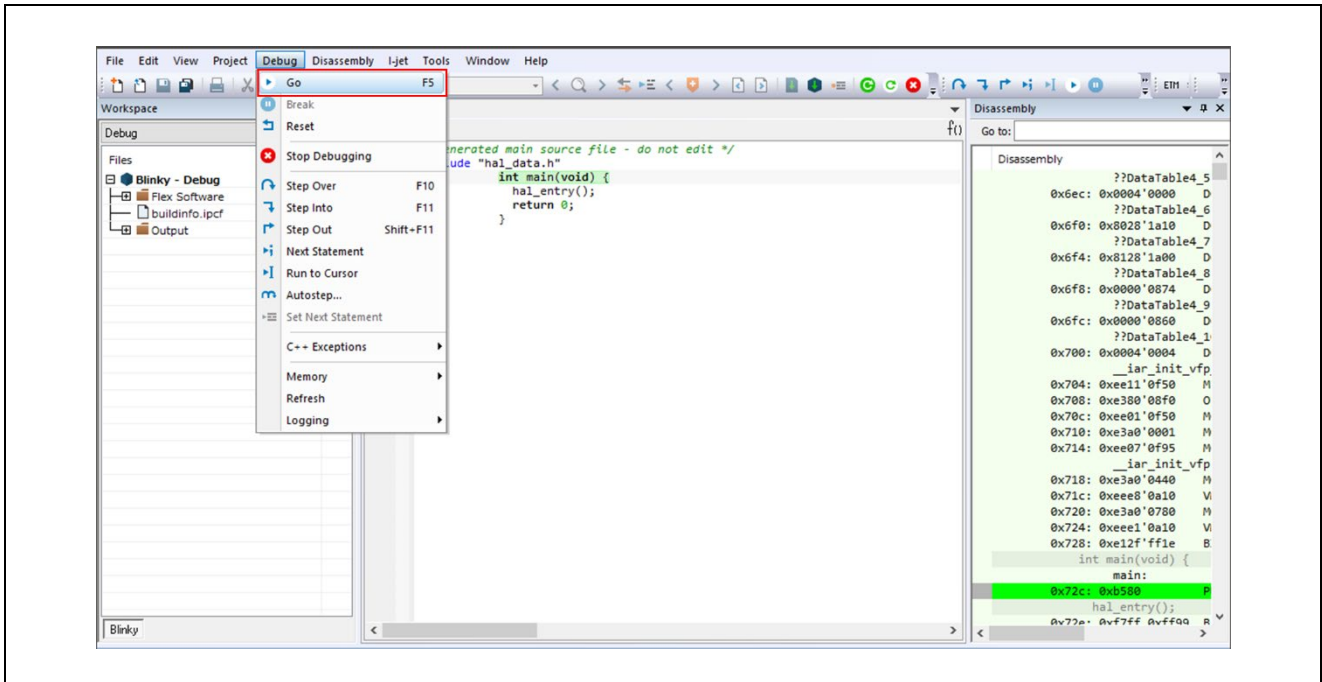


Figure 122. Go Button

The blinky application is stored in the `hal_entry.c` file. This file is generated by FSP SC when you select the Blinky Project template and is located in the project's `src/` folder. In the IAR EWARM workspace view, the `hal_entry.c` is registered **Flex Software > Program Entry**.

The application performs the following steps:

1. Get the LED information for the selected board by the `bsp_leds_t` structure.
2. Initialize output level for LED pin to LOW using `R_BSP_PinClear((bsp_io_region_t) leds.p_leds[i][1], (bsp_io_port_pin_t) leds.p_leds[i][0])`.
3. Use `R_BSP_PinToggle ((bsp_io_region_t) leds.p_leds[i][1], (bsp_io_port_pin_t) leds.p_leds[i][0])` to set the output level to the LED pin.
4. `R_BSP_SoftwareDelay(delay, bsp_delay_units)` waits for a certain period of time. Then run #3 again.

On debugging on IAR EWARM, the breakpoint can be set by clicking the left space next to the line number.

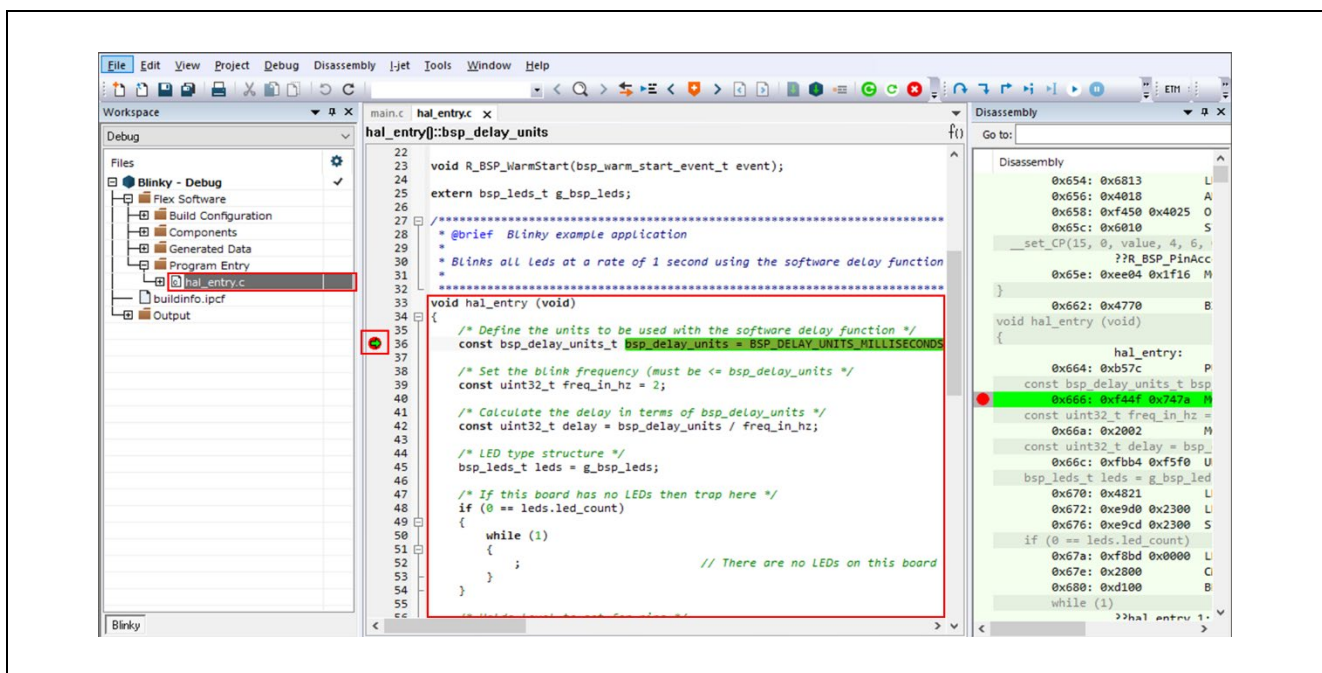


Figure 123. `hal_entry.c` and Setting Breakpoint

By using the breakpoint and the **Debug** menu or **Debug** toolbar, you can check the behavior of the Blinky application step by step.

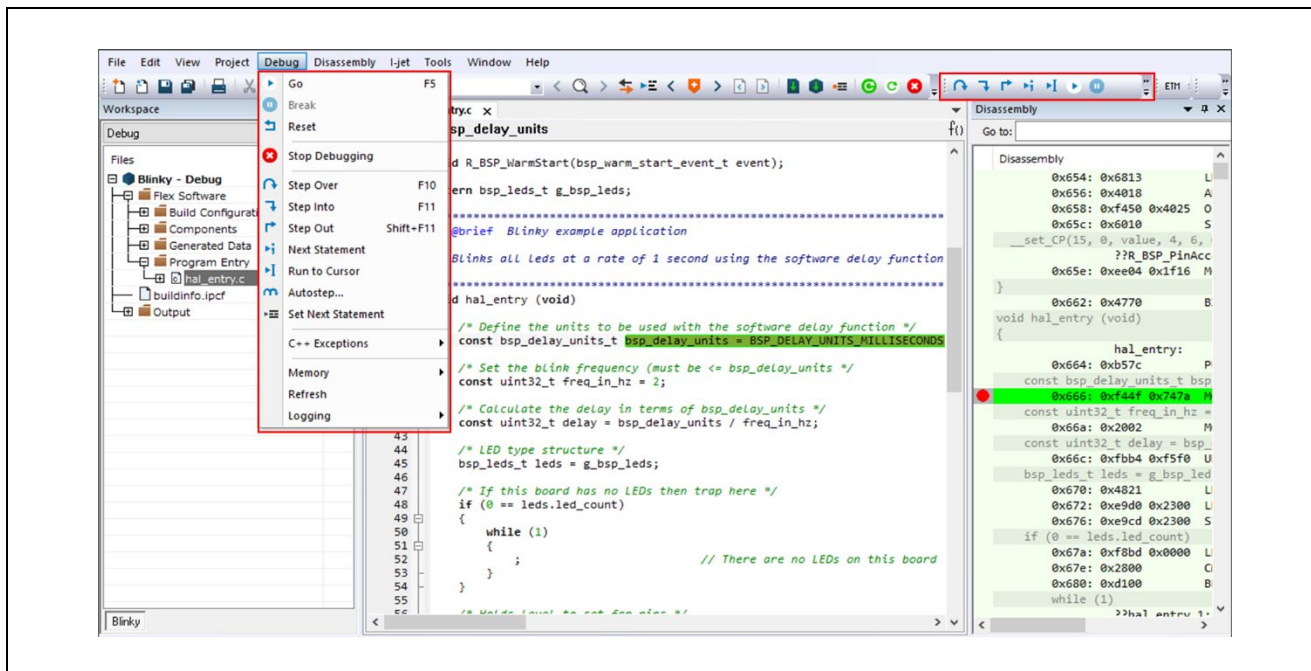


Figure 124. Debug Menu

When clicking **Go** button, the following LEDs on the board should now be blinking.

- RZ/T series
  - RSK+RZ/T2M: LED0-1 (CPU0), LED2-3 (CPU1)
  - RSK+RZ/T2L: LED0-6 (including LEDx\_ESC\_xxx)
  - RSK+RZ/T2ME: LED0-1 (CPU0), LED2-3 (CPU1)
  - RZ/T2H Evaluation Board: LED0 (CR52 CPU0), LED1 (CR52 CPU1), LED2 (CA55 Core0)
- RZ/N series
  - RSK+RZ/N2L: LED0-3
  - RZ/N2H Evaluation Board: LED3 (CR52 CPU0), LED4 (CR52 CPU1), LED8 (CA55 Core0)

To suspend program execution, click **Debug > Break** or click on the **Pause** icon.



Figure 125. IAR EWARM Debugger Pause Icon

To exit Debug and disconnect from the debugger, click **Debug > Stop Debugging** or click on the **Stop** icon.



Figure 126. IAR EWARM Debugger Stop Icon

### 5.3.5 Debug for Multiprocessing

To debug the Blinky application of multiprocessing, follow these steps:

1. Open the primary project and close the secondary project on IAR EWARM.
2. Set the following in the primary project before debugging:
  - i. Click **Project > Options...**
  - ii. Click **Debugger > Multicore** and check the setting value of **Symmetric multicore** and set the following contents in **Asymmetric multicore**.
    - Symmetric multicore
      - Number of cores: 1
    - Asymmetric multicore
      - Simple
        - ✧ Partner workspace: \$PROJ\_DIR\$\..\[the secondary project name]\[the secondary project name].eww
        - ✧ Partner project: [the secondary project name]
        - ✧ Partner configuration: Debug

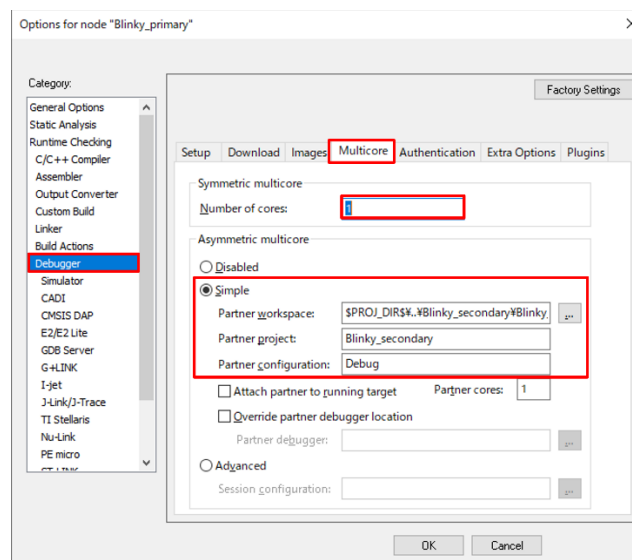


Figure 127. IAR EWARM Project Options for the Primary Project (Multicore)

- iii. Click **OK** and close the Options window.

3. Download of the primary project with procedure 5.3.4 Download & Debug the Project as shown in Figure 120.
4. The secondary project is automatically launched. Once the download is completed and the debug is started, the program breaks at the beginning of **system\_init** in **startup\_core.c**.
5. Run the program of the primary project as shown in Figure 122 to copy the binaries of the secondary and subsequent projects to the internal RAM in the primary project. After the primary project reaches **hal\_entry** in **main.c**, another core is executed. If the LEDs are blinking, proceed to the next step.
6. The primary project is in operation, and runs the program of the secondary project.
7. When exiting Debug and disconnecting from the debugger, if debugging is stopped in one of the projects, either the primary or the secondary, the other will automatically stop as well.

When changing the project and debugging it again, refer to No. 14 in the Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM.

## 5.4 Re-configuring Project with FSP SC

To proceed with the tutorial on the Blinky project, the FSP configuration steps of the Blinky project were skipped in this chapter. The FSP SC can be launched from IAR EWARM or the command prompt, and the FSP project configuration can be reconfigured by the FSP SC.

There are two ways to launch FSP SC with an existing project.

### 5.4.1 Launch FSP SC from IAR EWARM

1. Select "Tools -> Configure Tools..."
2. Select "New" and fill in the fields as follows:
  - Menu Text        FSP Smart Configurator
  - Command        \$RASC\_EXE\_PATH\$
  - Argument        --compiler IAR configuration.xml
  - Initial Directory \$PROJ\_DIR\$

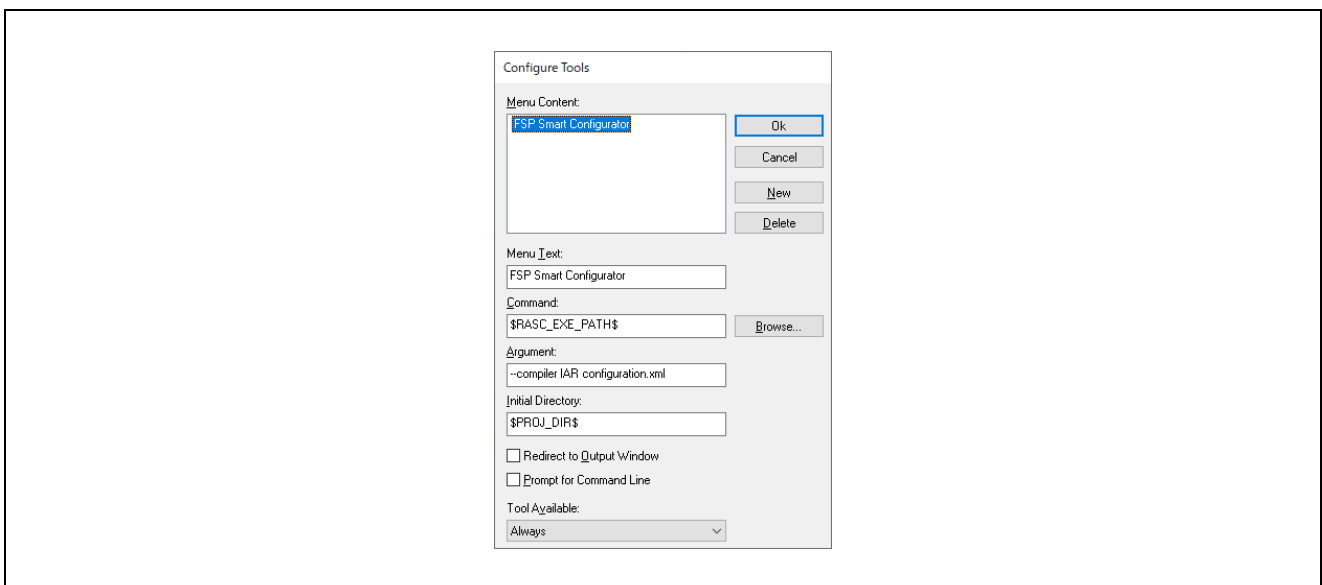


Figure 128. Settings to Launch FSP SC from IAR EWARM

### 5.4.2 Launch from the Command Prompt

1. Open a command prompt window.
2. Move to the folder where the created project is located.
3. Execute the following command.
  - {FSP SC installation folder} \ eclipse \ rasc.exe –compiler IAR configuration.xml

### 5.5 Note when debugging in different workspaces

The project was created, built, and debugged in chapters 5.3.2 through 5.3.5 can be run in other workspaces. When debugging in the other workspace, please note the following two points:

- Apply the same version of the FSP package used for the project to the FSP SC.
- After importing the project, the user needs to click the **Generate Project Content** button in the FSP configuration.

## 6. FSP Configuration Users Guide

### 6.1 What is a Project?

In e<sup>2</sup> studio, all FSP applications are organized in RZ/T2, RZ/N2 MPU projects. Setting up an RZ/T2, RZ/N2 MPU project involves:

1. Create a Project
2. Configuring a Project

These steps are described in detail in the next two sections. When you have existing projects already, after you launch e<sup>2</sup> studio and select a workspace, all projects previously saved in the selected workspace are loaded and displayed in the **Project Explorer** window. Each project has an associated configuration file named configuration.xml, which is located in the project's root directory.

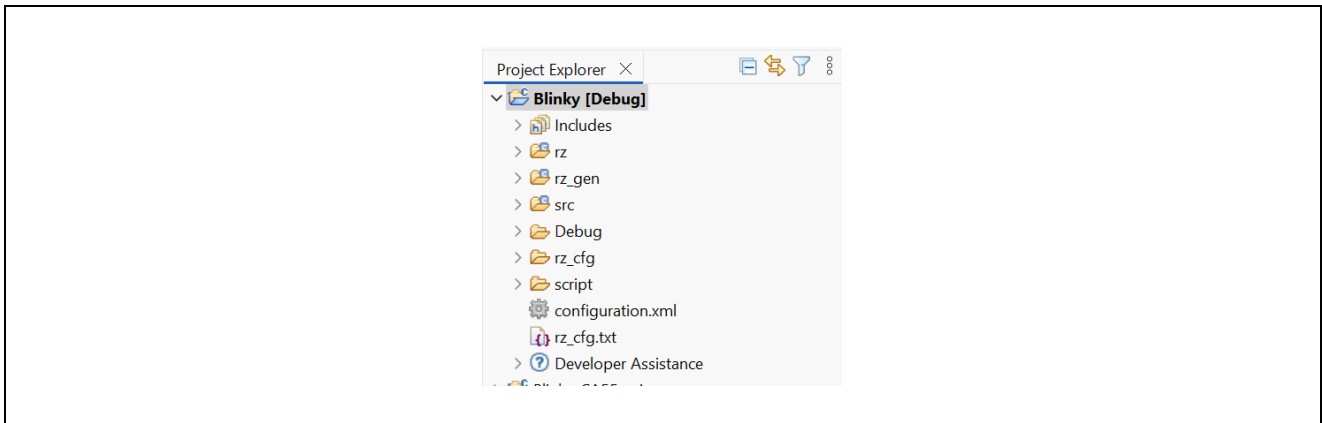


Figure 129. e<sup>2</sup> studio Project Configuration File

Double-click on the configuration.xml file to open the RZ/T2, RZ/N2 MPU Project Editor. To edit the project configuration, make sure that the **FSP Configuration** perspective is selected in the upper right-hand corner of the e<sup>2</sup> studio window. Once selected, you can use the editor to view or modify the configuration settings associated with this project.

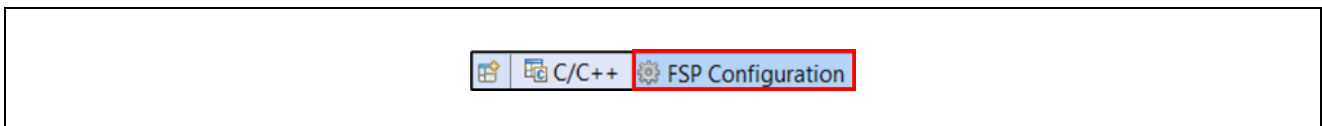
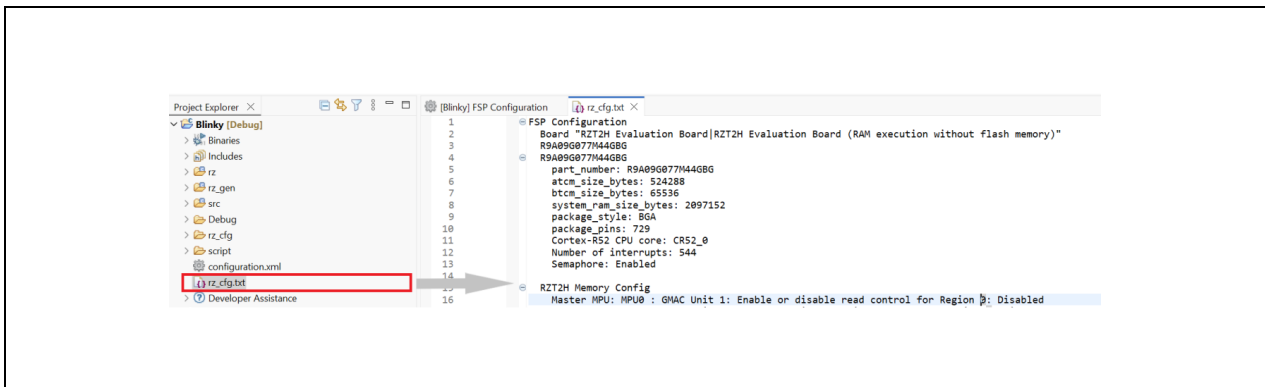


Figure 130. e<sup>2</sup> studio FSP Configuration Perspective

**Note:** Whenever the RZ/T2, RZ/N2 project configuration (that is, the configuration.xml file) is saved after configuring the project, a verbose RZ/T2, RZ/N2 Project Report file (rz\_cfg.txt) with all the project settings is generated. The format allows differences to be easily viewed using a text comparison tool. The generated file is located in the project root directory.

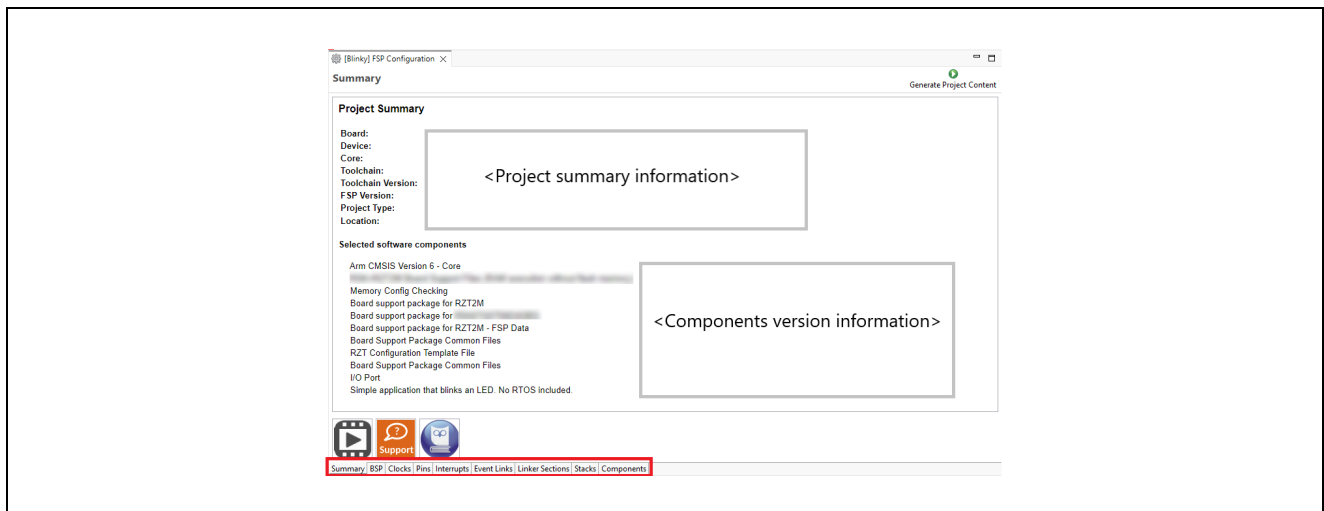


**Figure 131. RZ/T2, RZ/N2 Project Report**

The RZ/T2, RZ/N2 Project Editor has several tabs. The configuration steps and options for individual tabs are discussed in the following sections.

**Note:**

The tabs available in the RZ/T2, RZ/N2 Project Editor depend on the e<sup>2</sup> studio version, and the layout may vary slightly, the functionality should be easy to follow.



**Figure 132. RZ/T2, RZ/N2 Project Editor Tabs**

## 6.2 Create a Project

### 6.2.1 Creating a New Project

For RZ/T2, RZ/N2 MPU applications, generate a new project using the following steps:

1. Click on **File > New > Renesas C/C++ Project > Renesas RZ**.

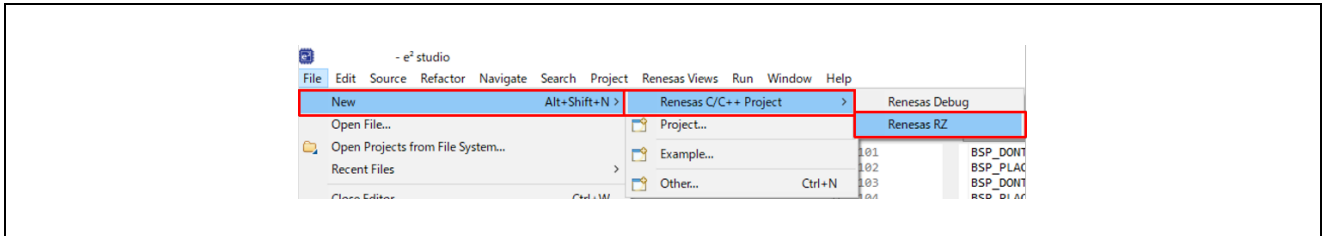


Figure 133. New RZ/T2, RZ/N2 MPU Project

2. Then click on the **Renesas C/C++ FSP Project** template for the type of project you are creating.

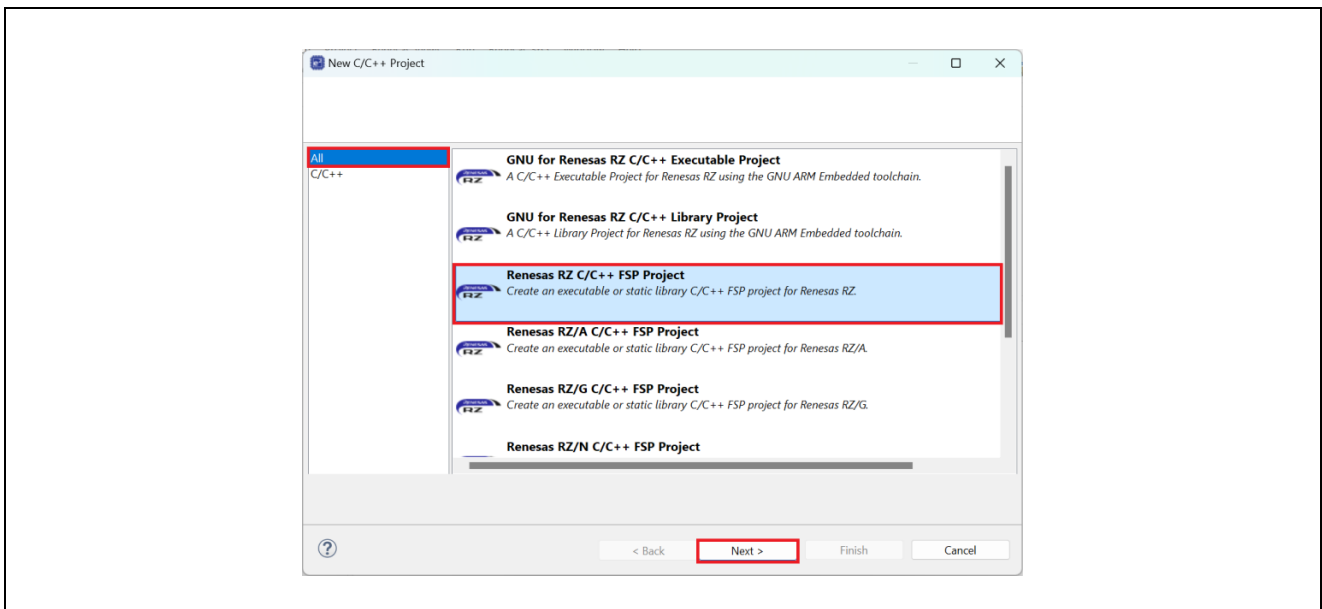


Figure 134. New Project Templates

3. Select a project name and location.
4. Click Next.

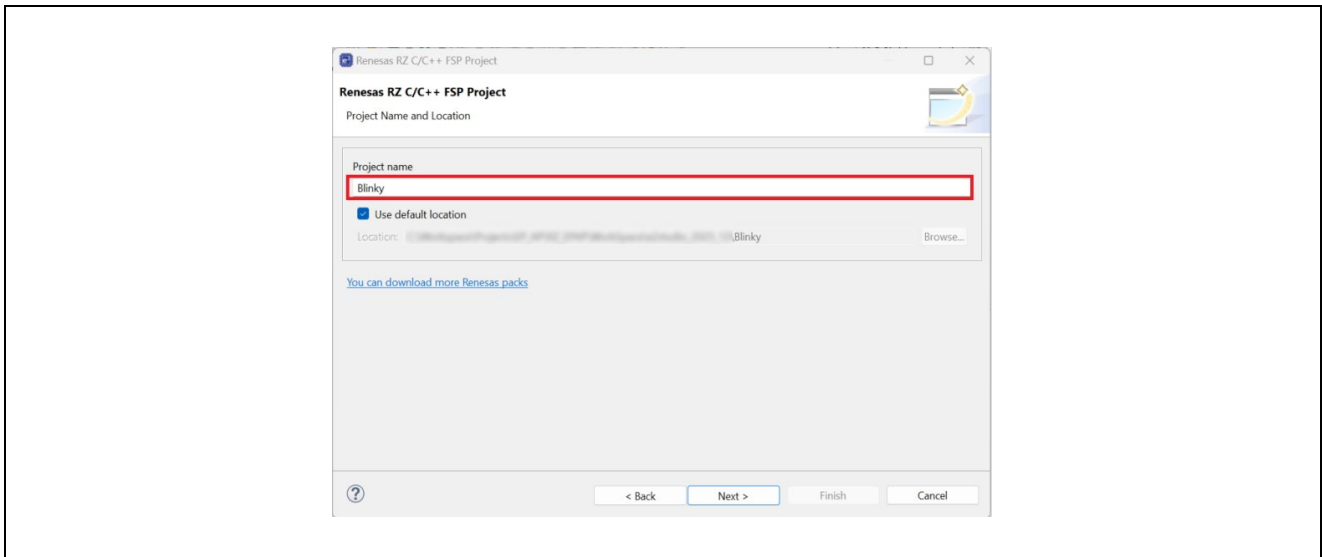


Figure 135. RZ/T2, RZ/N2 MPU Project Generator (Part 1)

### 6.2.2 Selecting a Board and Toolchain

In the Project Configuration window, select the hardware and software environment:

1. Select the **FSP version**.
2. Select the **Board** and **Device** for your application.

**Note:**

You can select an existing RZ/T2, RZ/N2 MPU Evaluation Kit (Such as RSK) or select a **Custom User Board** for any of the RZ/T2, RZ/N2 MPU devices with your own BSP definition.

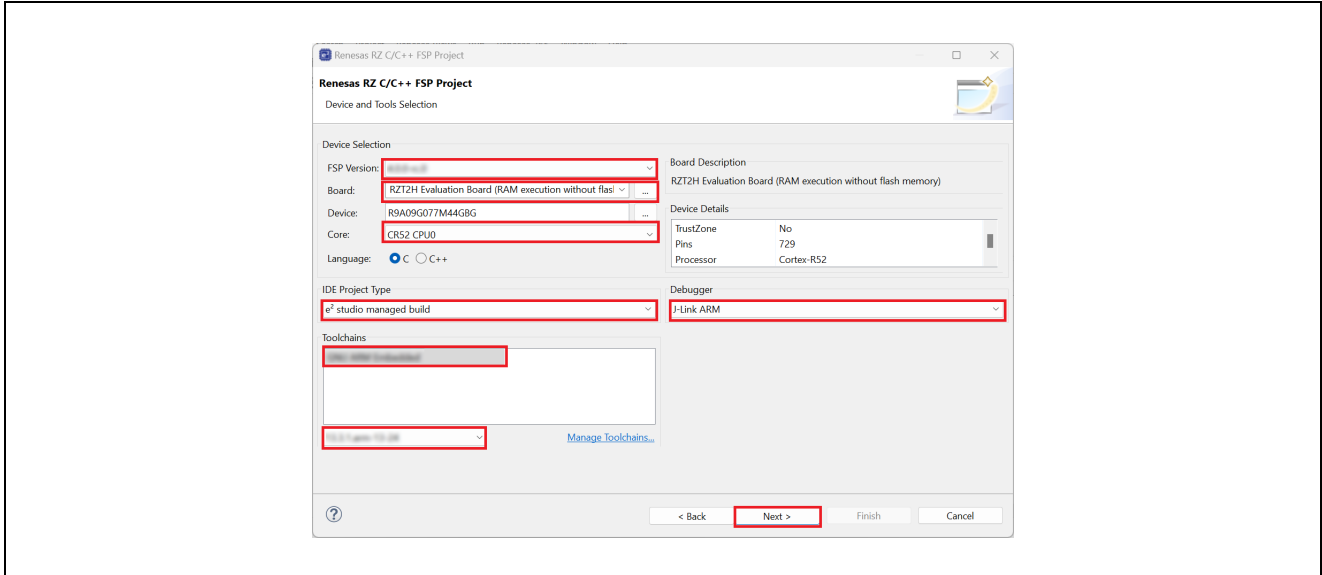
When you use the RZ/T2, RZ/N2 MPU Evaluation Kit,

- First, please set the **Board** to the Evaluation Kit and the boot mode that you use.
- In this case, please don't change the **Device, which is automatically set to the device that the RSK board uses.**

When you use a **Custom User Board**,

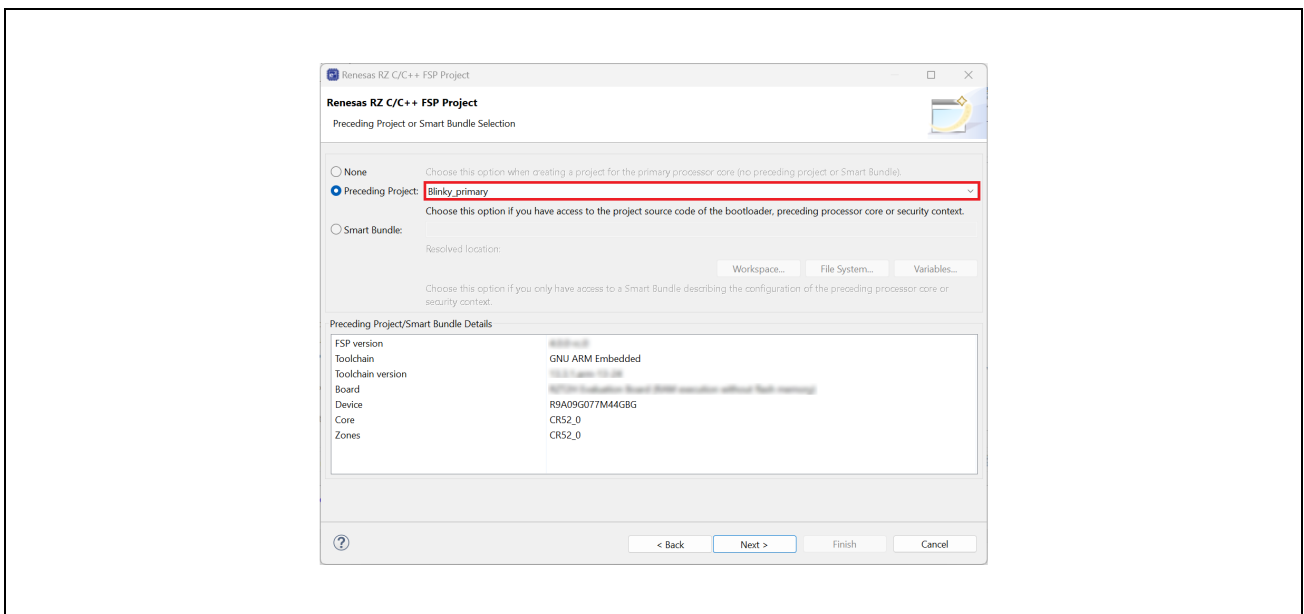
- First, please set the **Device** to your device on your board.
- Second, please set the Board to **Custom User Board** with the boot mode that you use.

3. Select the **Core**. You could select if you selected multicore device for **Device**.
4. Select the **Toolchains**.
5. Select the **Toolchain version**.
6. Select the **Debugger**. The J-Link Arm Debugger is preselected.
7. Click **Next**.



**Figure 136. RZ/T2, RZ/N2 MPU Project Generator (Part 2)**

If CR52 CPU0 is not selected for the secondary project of multiprocessing in procedure 3, you need to select the preceding project. To select the preceding project when creating the secondary project for multiprocessing, it is required to prepare CR52 CPU0 as the primary project before creating the secondary project.



**Figure 137. RZ/T2, RZ/N2 MPU Project Generator (Part 3)**

### 6.2.3 Selecting a Project Template

In the next window, select the build artifact and RTOS.

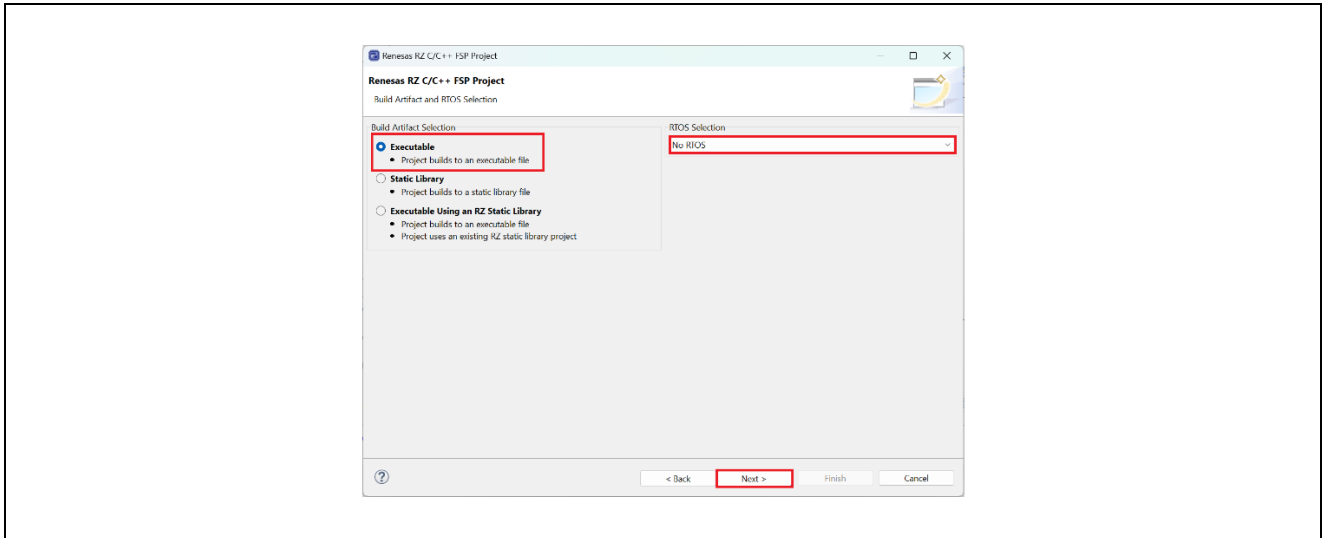


Figure 138. RZ/T2, RZ/N2 MPU Project Generator (Part 4)

In the next window, select a project template from the list of available templates. By default, this screen shows the templates that are included in your current RZ/T2, RZ/N2 MPU Pack. Once you have selected the appropriate template, click **Finish**.

**Note:**

If you want to develop your own application, select the basic template for your board, **Bare Metal – Minimal**.

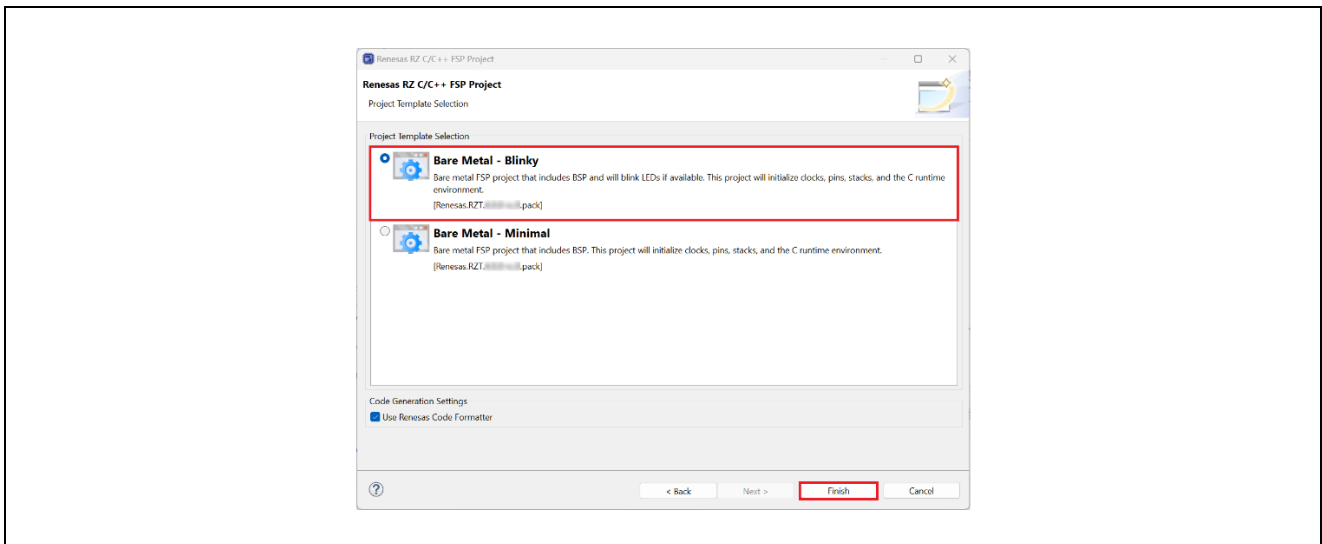
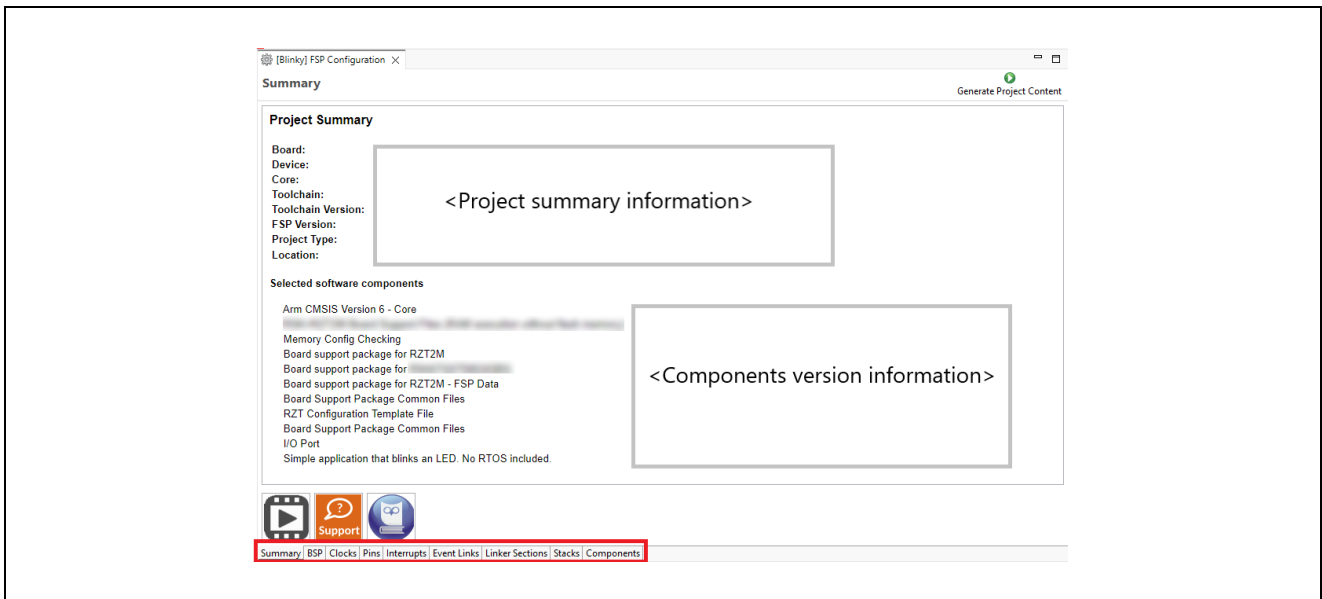


Figure 139. RZ/T2, RZ/N2 MPU Project Generator (Part 5)

When the project is created, e<sup>2</sup> studio displays a summary of the current project configuration in the RZ/T2, RZ/N2 MPU Project Editor.



**Figure 140. RZ/T2, RZ/N2 MPU Project Editor and Available Editor Tabs**

On the bottom of the RZ/T2, RZ/N2 MPU Project Editor view, you can find the tabs for configuring multiple aspects of your project:

- With the **Summary** tab, you can see all the key characteristics of the project: board, device, toolchain, and more.
- With the **BSP** tab, you can change board-specific parameters from the initial project selection.
- With the **Clocks** tab, you can configure the MPU clock settings for your project.
- With the **Pins** tab, you can configure the electrical characteristics and functions of each port pin.
- With the **Interrupts** tab, you can add new user events/interrupts.
- With the **Event Links** tab, you can configure events used by the Event Link Controller.
- With the **Stacks** tab, you can add and configure FSP modules. For each module selected in this tab, the **Properties** window provides access to the configuration parameters and interrupt selections.
- The **Components** tab provides an overview of the selected modules. Although you can also add drivers for specific FSP releases and application sample code here, this tab is normally only used for reference.

#### 6.2.4 Duplication of Resources

In the case of creating a project with a core other than CR52 CPU0 on a multicore device, duplicate resources will be grayed out or hidden in each tab of Configuration. For more details, see the Configuration section of the [Flexible Software Package Documentation](#) API Reference > BSP > MCU Board Support Package page.

## 6.3 Configuring a Project

Each of the configurable elements in an FSP project can be edited using the appropriate tab in the RZ/T2, RZ/N2 Configuration editor window. Importantly, the initial configuration of the MPU after reset and before any user code is executed is set by the configuration settings in the **BSP** tab. When you select a project template during project creation, e<sup>2</sup> studio configures default values that are appropriate for the associated board. You can change those default values as needed. The following sections detail the process of configuring each of the project elements for each of the associated tabs.

### 6.3.1 Summary Tab

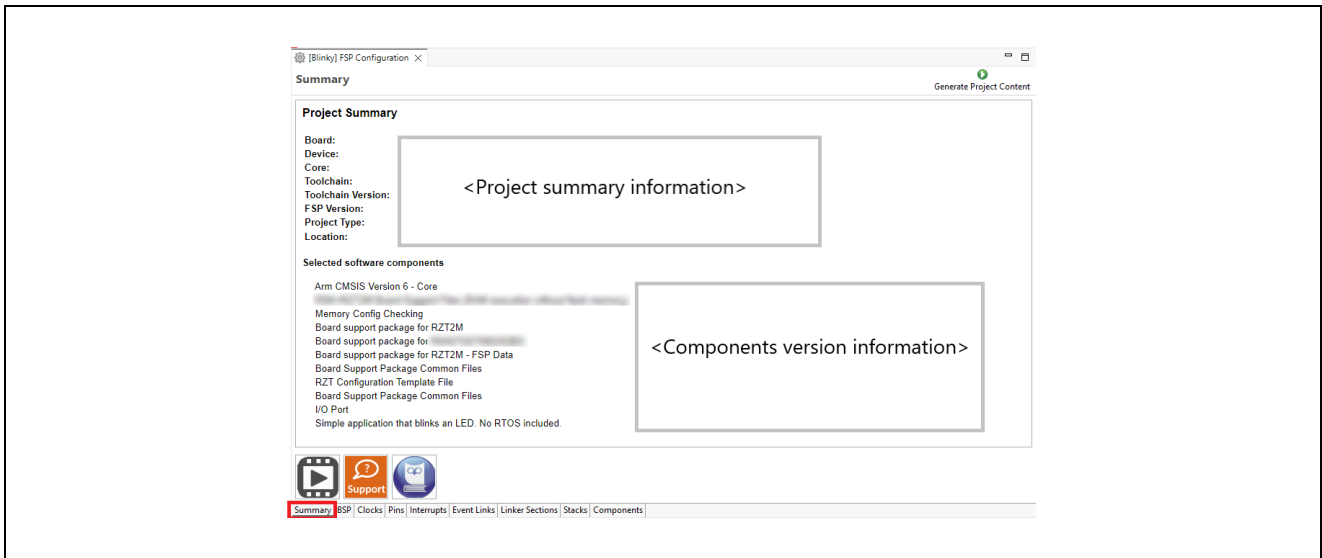


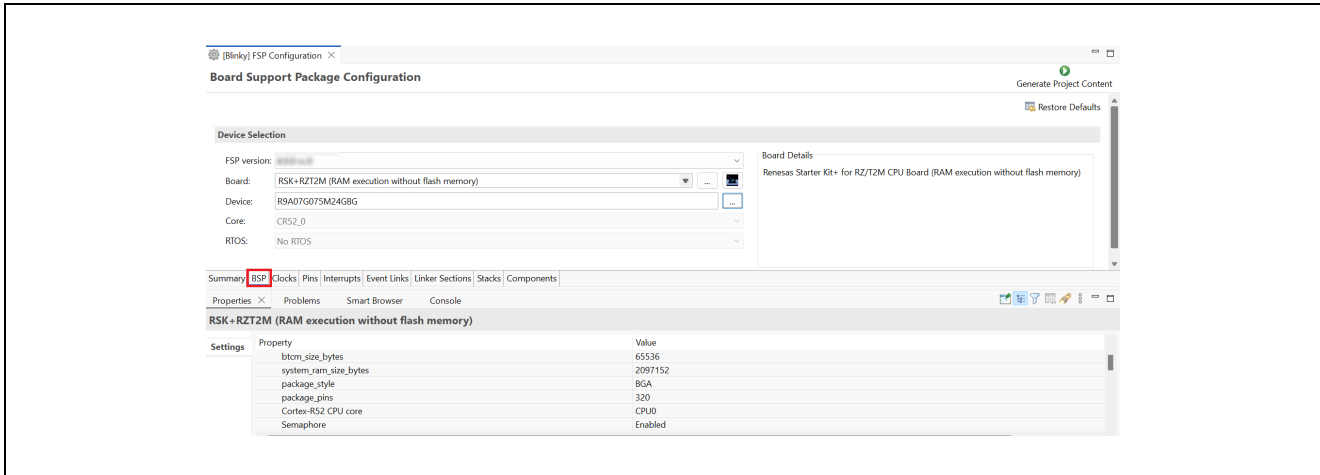
Figure 141. Configuration Summary Tab

The **Summary** tab, seen in the above figure, identifies all the key elements and components of a project. It shows the target board, the device, the toolchain, and the FSP version. Additionally, it provides a list of all the selected software components and modules used by the project. This is a more convenient summary view when compared to the **Components** tab.

### 6.3.2 Configuring the BSP

The **BSP** tab shows the currently selected board (if any) and device. The Properties view is located in the lower left of the Project Configurations view, as shown below.

**Note:** If the Properties view is not visible, click **Window > Show View > Properties** in the top menu bar.



**Figure 142. Configuration BSP Tab**

The **Properties** view shows the configurable options available for the BSP. These can be changed as required. The BSP is the FSP layer above the MPU hardware. e<sup>2</sup> studio checks the entry fields to flag invalid entries. For example, only valid numeric values can be entered for the stack size.

When you click the **Generate Project Content** button, the BSP configuration contents are written to:

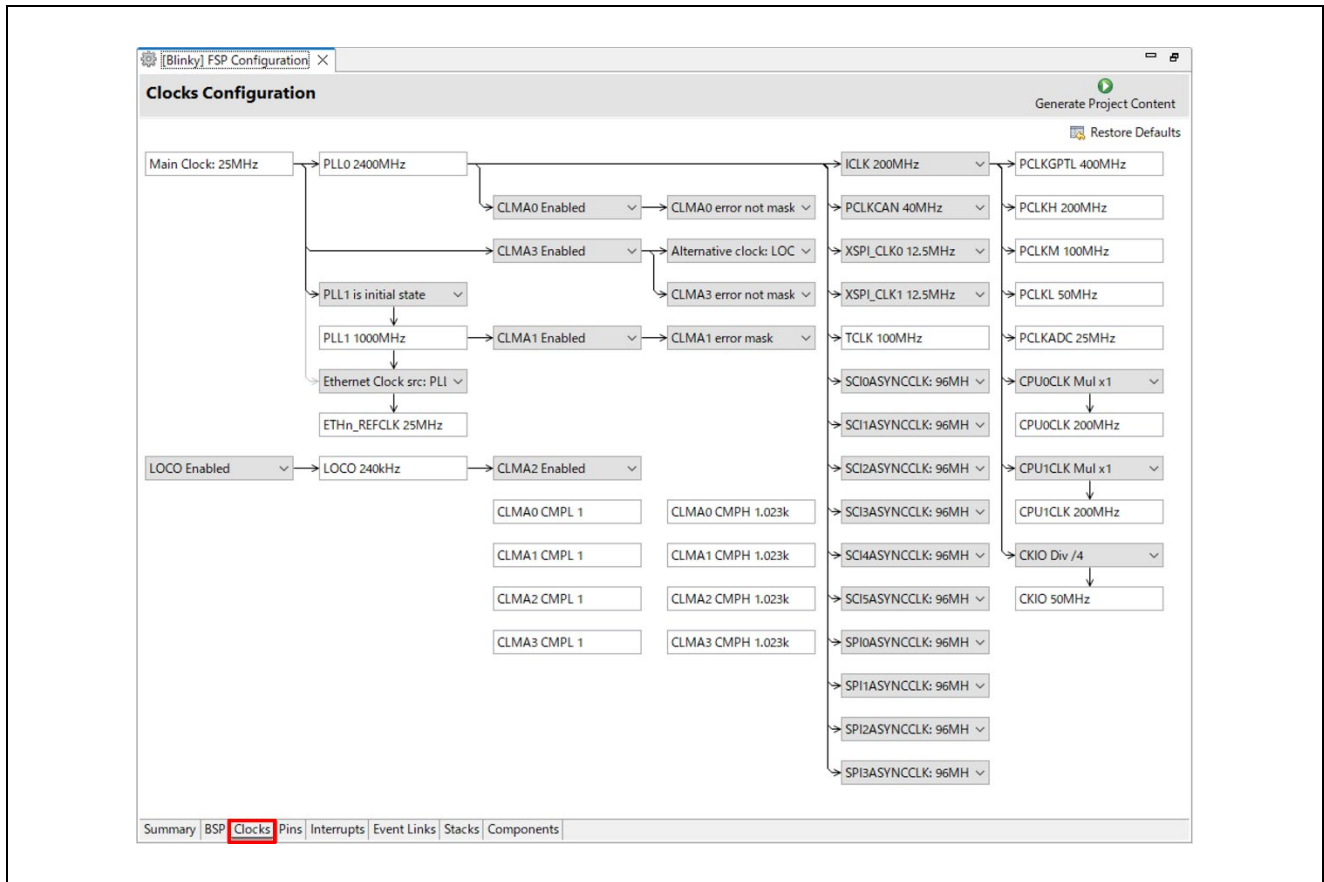
- `rz_cfg/fsp_cfg/bsp/bsp_cfg.h`

This file is created if it does not already exist.

**Warning:** Do not edit this file as it is overwritten whenever the Generate Project Content button is clicked.

### 6.3.3 Configuring Clocks

The Clocks tab presents a graphical view of the MPU's clock tree, allowing the various clock dividers and sources to be modified.



**Figure 143. Configuration Clocks Tab**

When you click the Generate Project Content button, the clock configuration contents are written to:

- `rz_gen/bsp_clock_cfg.h`

This file will be created if it does not exist.

**Warning:** Do not edit this file as it is overwritten whenever the **Generate Project Content** button is clicked.

### 6.3.4 Configuring Pins

The **Pins** tab provides a flexible configuration of the MPU's pins. As many pins are able to provide multiple functions, they can be configured on a peripheral basis. For example, selecting a serial channel via the SCI peripheral offers multiple options for the location of the receive and transmit pins for that module and channel. Once a pin is configured, it is shown as green in the **Package** view.

**Note:** If the **Package** view window is not open in e<sup>2</sup> studio, select **Window > Show View > Pin Configurator > Package** from the top menu bar to open it.

The **Pins** tab simplifies the configuration of large packages with highly multiplexed pins by highlighting errors and presenting the options for each pin or for each peripheral. If you selected a project template for a specific board, such as RSK+RZT2M, some peripherals connected to the board are preselected.

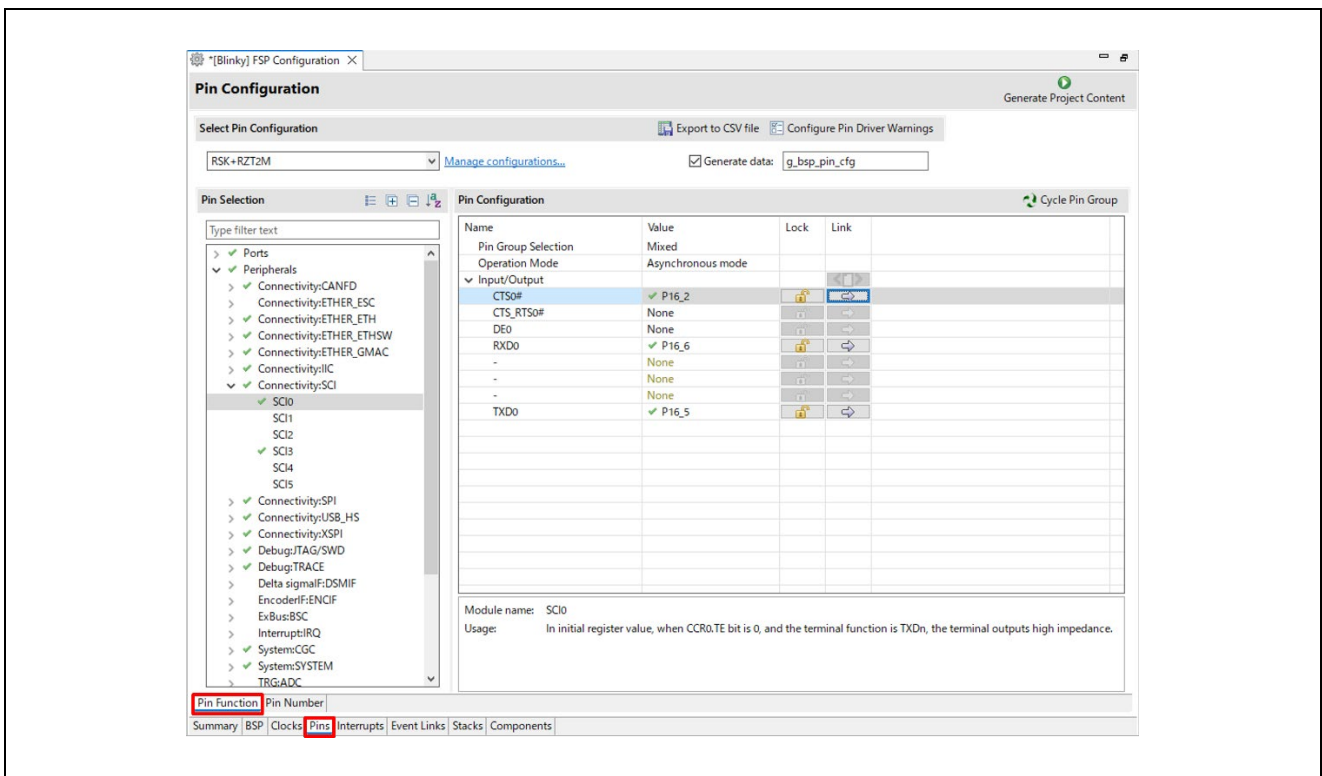


Figure 144. Pin Configuration

The pin configurator includes a built-in conflict checker, so if the same pin is allocated to another peripheral or I/O function, the pin will be shown as red in the package view and also with a white cross in a red square in the **Pin Selection** pane and **Pin Configuration** pane in the main **Pins** tab. The **Pin Conflicts** view provides a list of conflicts, so conflicts can be quickly identified and fixed.

In the example shown below, port P162 is already used by the GPIO, and the attempt to connect this port to the Serial Communications Interface (SCI) results in a dangling connection error. To fix this error, select another port from the pin drop-down list or disable the GPIO in the **Pin Selection** pane on the left side of the tab.

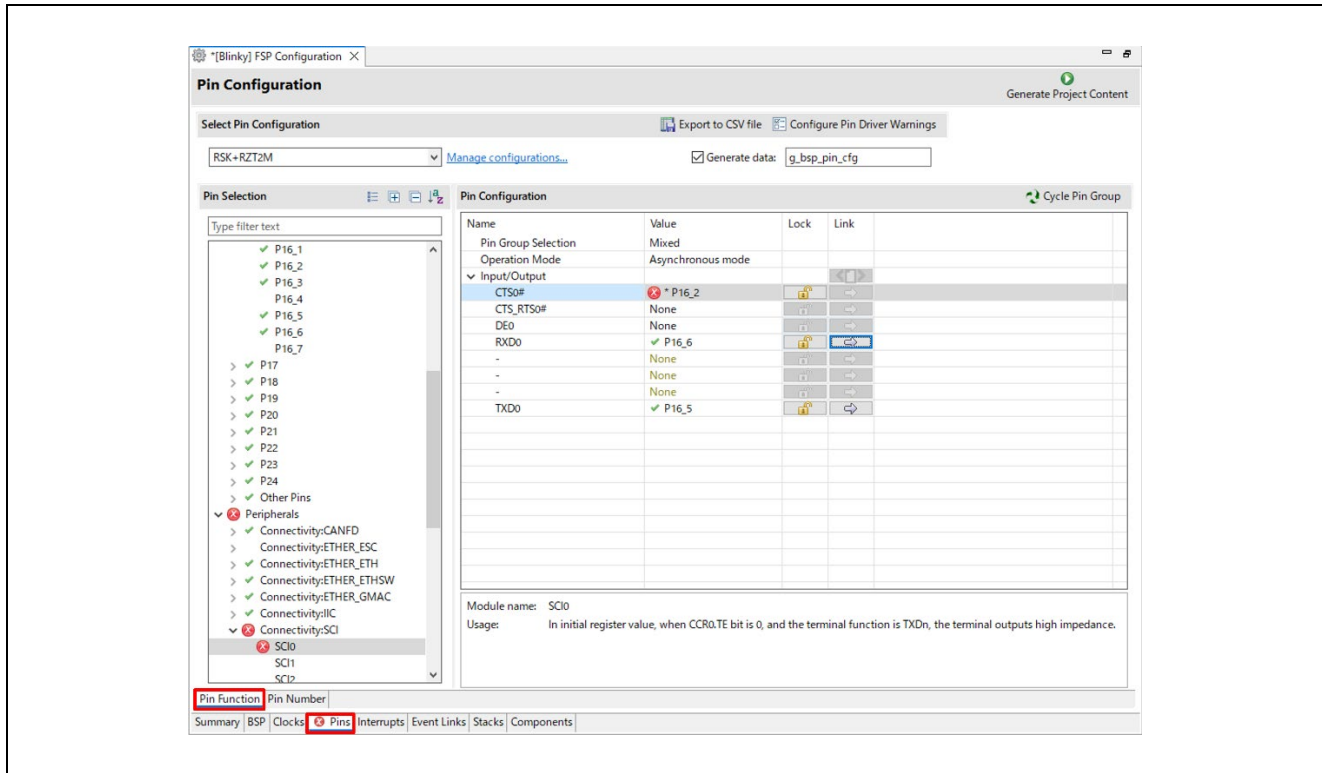


Figure 145. Conflict Checker in Pin Configuration

The pin configurator also shows a package view and the selected electrical or functional characteristics of each pin.

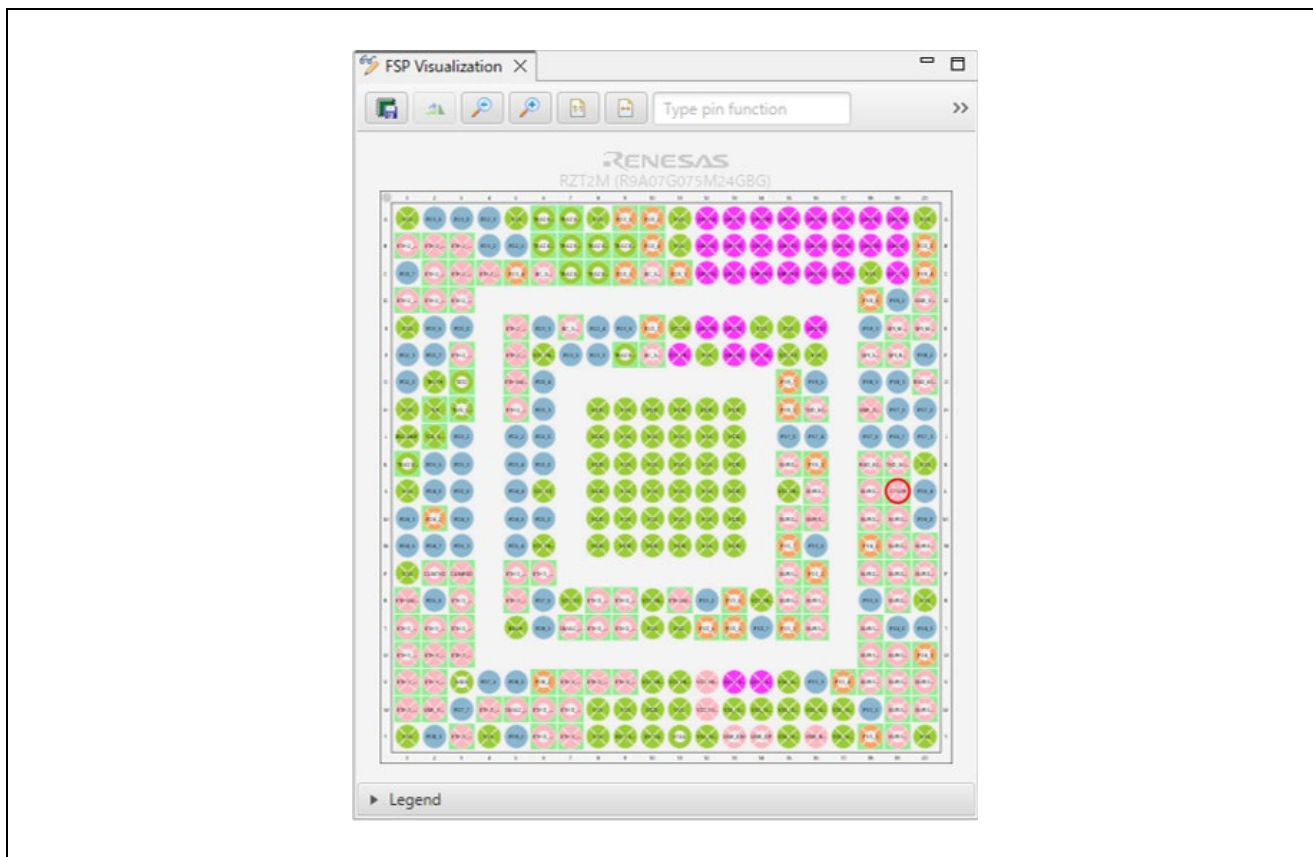


Figure 146. Pin Configurator Package View

When you click the **Generate Project Content** button, the pin configuration contents are written to:

- rz\_cfg/fsp\_cfg/bsp/bsp\_pin\_cfg.h

This file will be created if it does not already exist.

**Warning:** Do not edit this file as it is overwritten whenever the **Generate Project Content** button is clicked.

## 6.4 Configuring Interrupts from the Stacks Tab

You can use the **Properties** view in the **Stacks** tab to enable interrupts by setting the interrupt priority. Select the driver in the **Stacks** pane to view and edit its properties.

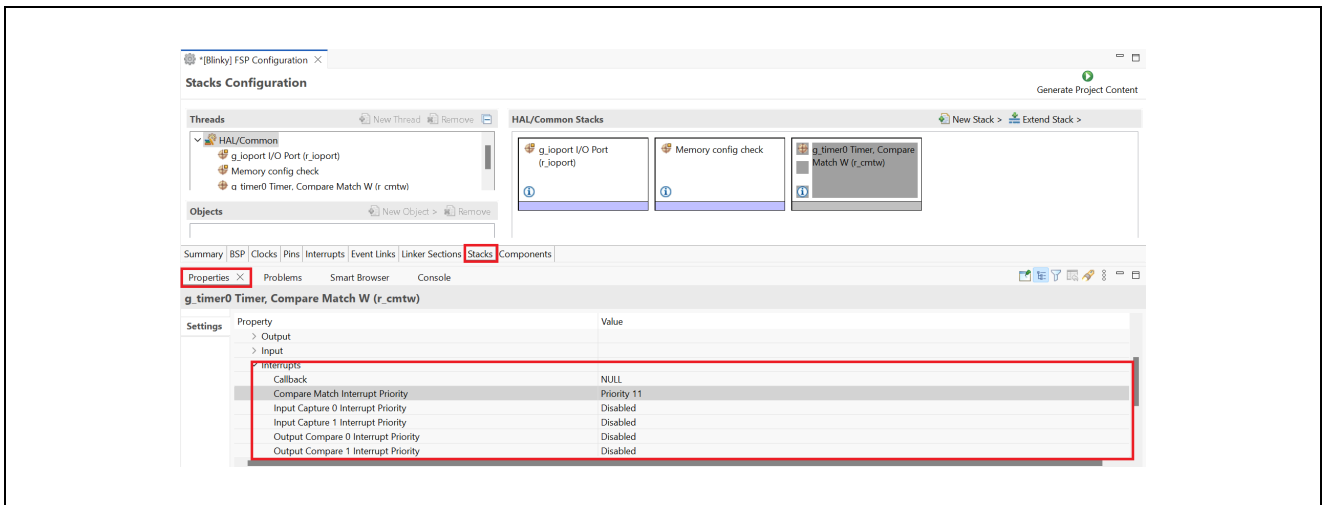


Figure 147. Configuring Interrupts in the Stacks Tab

### 6.4.1 Creating Interrupts from the Interrupts Tab

On the **Interrupts** tab, the interrupt of the driver selected in the **Stacks** tab is registered.

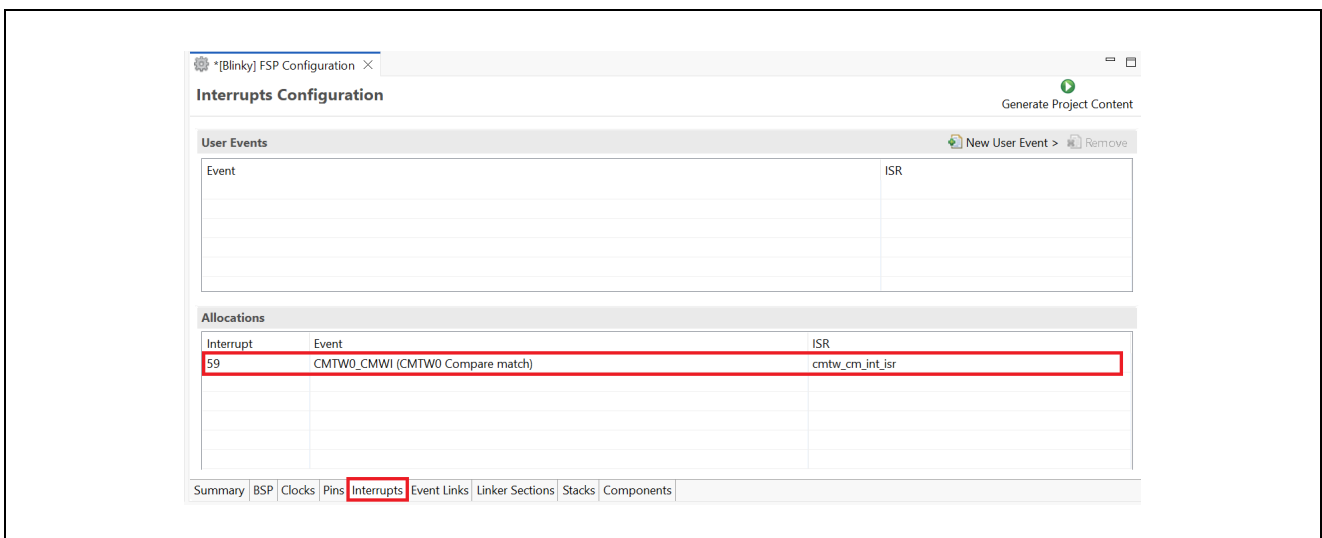


Figure 148. Configuring Interrupt in Interrupt Tab

And the user can add a peripheral interrupt created by the user. This can be done by adding a new event via the **New User Event** button.

## 6.4.2 Viewing Event Links

The Event Links tab can be used to view the Event Link Controller events. The events are sorted by peripheral to make it easy to find and verify them.

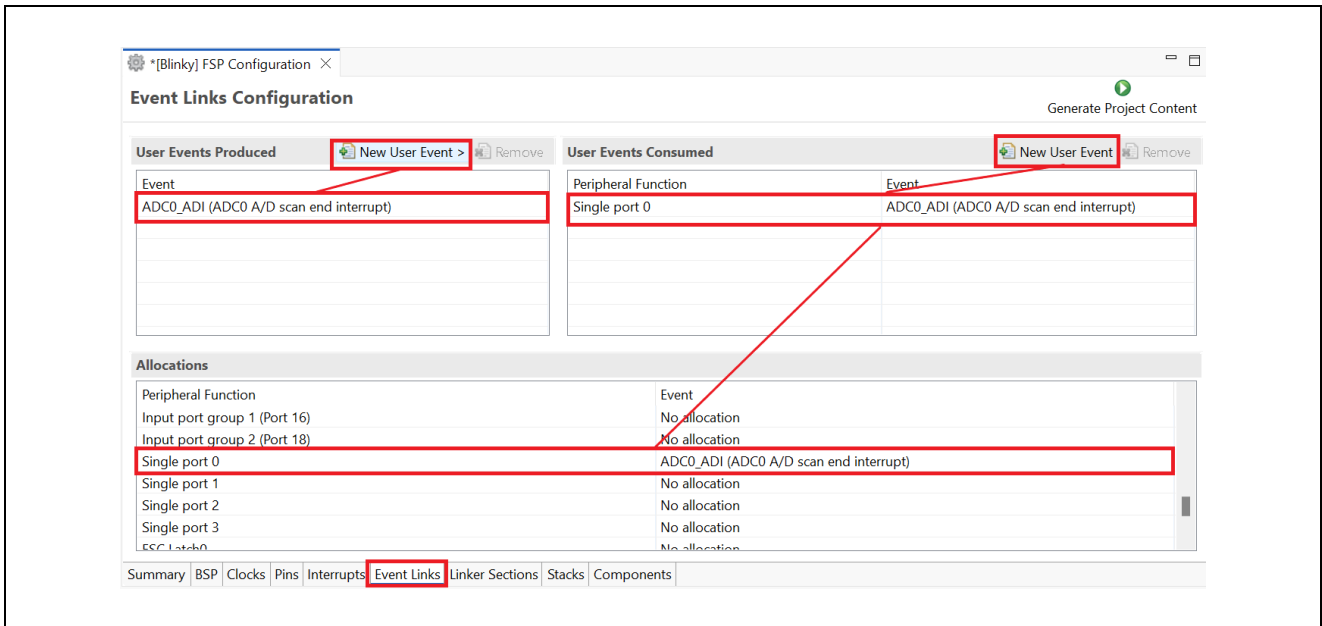


Figure 149. Viewing Event Links

Like the Interrupts tab, user-defined event sources and destinations (producers and consumers) can be defined by clicking the relevant **New User Event** button. Once a consumer is linked to a producer, the link will appear in the **Allocations** section at the bottom.

**Note:** When selecting an ELC event to receive for a module (or when manually defining an event link), only the events that are made available by the modules configured in the project will be shown.

### 6.5 Adding and Configuring HAL Drivers

For applications that run outside or without the RTOS, you can add additional HAL drivers to your application using the HAL/Common thread. To add drivers, follow these steps:

1. Click on the HAL/Common icon in the **Stacks** pane. The Modules pane changes to **HAL/Common Stacks**.
2. Click New Stack to see a drop-down list of HAL-level drivers available in the FSP.
3. Select a driver from the menu **New Stack > Driver**.

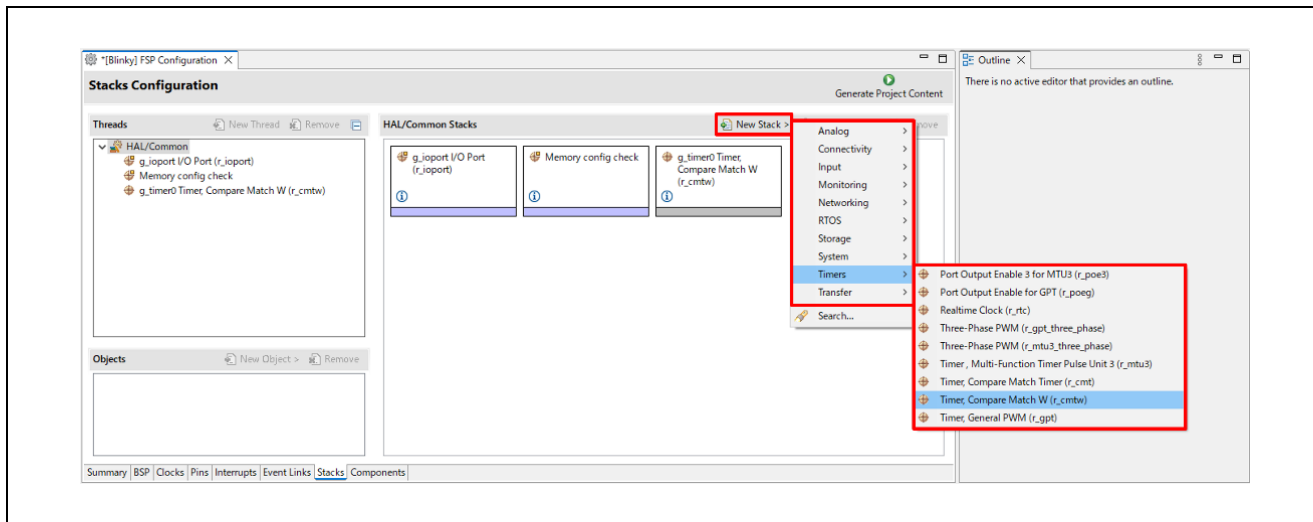


Figure 150. e2 studio Project Configurator – Adding Drivers

4. Select the driver module in the **HAL/Common Modules** pane and configure the driver properties in the **Properties** view.

e2 studio adds the following files when you click the **Generate Project Content** button:

- The selected driver module and its files to the rz/fsp directory
- The main() function and configuration structures, header files for your application are shown in the table below.

Table 18. Generate Contents on FSP Configuration

File	Contents	Overwritten by Generate Project Content?
rz_gen/main.c	Contains main() calling generated and user code. When called, the BSP has already initialized the MPU.	Yes
rz_gen/hal_data.c	Configuration structures for HAL Driver only modules.	Yes
rz_gen/hal_data.h	Header file for HAL driver only modules.	Yes
src/hal_entry.c	User entry point for HAL Driver only code. Add your code here.	No

The configuration header files for all included modules are created or overwritten in this folder:

- rz\_cfg/fsp\_cfg

## 6.6 Reviewing and Adding Components

The **Components** tab enables the individual modules required by the application to be included or excluded. Modules common to all RZ MPU projects are preselected. All modules that are necessary for the modules selected in the **Stacks** tab are included automatically. You can include or exclude additional modules by ticking the box next to the required component.

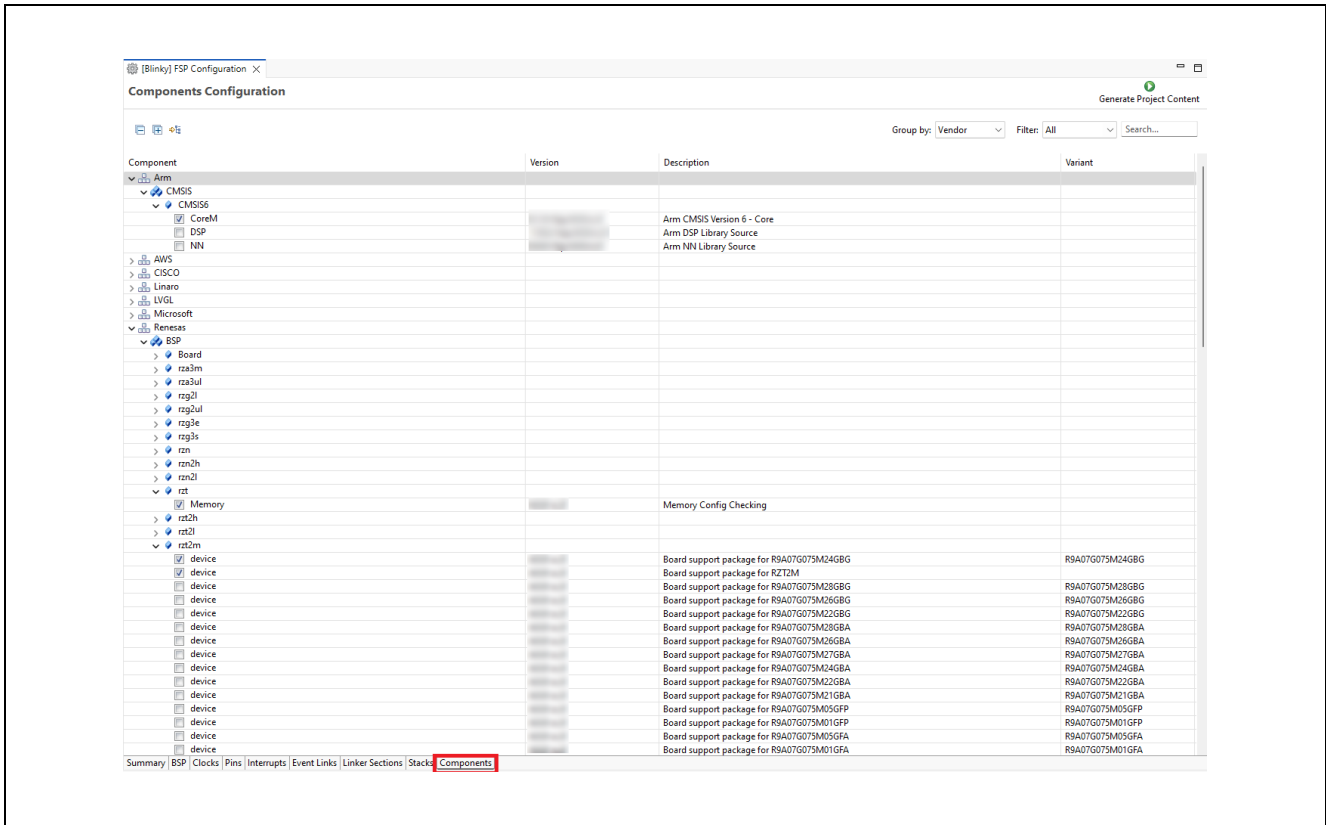


Figure 151. Components Tab

Clicking the **Generate Project Content** button copies the .c and .h files for each selected component into the following folders:

- rz/fsp/inc/api
- rz/fsp/inc/instances
- rz/fsp/src/bsp
- rz/fsp/src/<Driver\_Name>

e<sup>2</sup> studio also creates configuration files in the following folder with configuration options set in the **Stacks** tab.

- rz\_cfg/fsp\_cfg

## 7. Appendix

### 7.1 Known Issues

This chapter describes the known issues regarding the current version of FSP and related platform software. Most of the issues may require users to follow some manual operations to resolve the issues or to avoid the problems caused by the issues. Please follow the operations in the description of the issues if you use the features related to the issues. The grayed-out items have been resolved.

The known issues are categorized into two main groups, FSP Configuration and FSP Modules.

- FSP Configuration

FSP Configuration on e<sup>2</sup> studio and FSP SC have various configuration features that work on the GUI with FSP.

Regarding the overview of each configuration feature (GUI tab) provided as a part of FSP configuration on e<sup>2</sup> studio and FSP SC, please see the chapter 6. “FSP Configuration Users Guide”.

- FSP Modules

The FSP provides HAL drivers and BSP configured by FSP Configuration on e<sup>2</sup> studio and FSP SC.

Regarding their features, usage notes, and API references, please see the related file “FSP Documentation”.

**Table 19. List of Known Issues**

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
1	“r_gmac” may be shown as “r_ether” incorrectly.	✓				✓		FSP Configuration, Stacks
2	“Edge” can be selected as Transfer End Interrupt Detect Type in “r_dmac”, but it cannot be used.	✓	✓			✓		FSP Configuration, Stacks
3	When the “Device” or “Board” selection in the BSP tab is changed, the BSP properties are sometimes configured incorrectly.	✓	✓	✓	✓	✓	✓	FSP Configuration, BSP
4	(FSP SC ONLY) The device name is not output correctly depending on the selected device.	✓		✓				FSP Configuration, BSP
5	Errors occur when changing board settings.		✓			✓		FSP Configuration, BSP
6	Pin configuration error occurs in the MPX-IO 16bit operating mode of “r_bsc”.	✓	✓			✓		FSP Configuration, Pins
7	Build error when using the definition name of input/output external pins for the module.	✓	✓			✓		FSP Configuration, Pins
8	“R_SCI_UART_BaudCalculate()” of “r_sci_uart” module properly works ONLY when its clock source is SCInASYNCCLK and its frequency is 96MHz.	✓	✓			✓		FSP Modules, SCI UART

No	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
9	“R_SPI_CalculateBitrate()” of “r_spi” module properly works ONLY when its clock source is SPInASYNCCLK and its frequency is 96MHz.	✓	✓			✓		FSP Modules, SPI
10	A warning occurs when building “r_gmac” module with the gcc compiler.	✓	✓					FSP Modules, Ethernet
11	In the FSP Documentation, there is an incorrect description. “API Reference > Modules > Ethernet PHY” page.	✓				✓		FSP Modules, Ethernet PHY
12	The interrupt number cannot be successfully acquired by the R_FSP_CurrentIrqGet() when multiple interrupt occurs.					✓		FSP Modules, FreeRTOS
13	Block Media Custom Implementation can be selected as Memory Implementation for “rm_freertos_plus_fat” module, but it cannot be used.	✓	✓	✓		✓		FSP Configuration , Stacks
14	The second argument of “r_mtu3” APIs does not match the common API.	✓	✓	✓	✓	✓	✓	FSP Modules, MTU3
15	In multiprocessing, a configuration error occurs when “r_gpt” module is used for both projects for CPU0 and CPU1.	✓		✓				FSP Configuration , Stacks
16	A project build error occurs when a 32-bit bus NOR flash and xSPI0 x8 boot modes are selected on the RZT Custom User Board.	✓						FSP Configuration , BSP
17	The secondary project for multiprocessing cannot be created when xSPI1 x1 boot modes are selected on the RZT Custom User Board.	✓						FSP Configuration , BSP
18	An incorrect value is set to a pin select value for MTU0-B/MTU6/MTU7 as the MTU3 output pin.		✓					FSP Modules, POE3
19	Build error when using DSMIFn_ERR as an additional trigger for “r_poe3” module.		✓					FSP Modules, POE3
20	Control setting values for MTU3 output pins in the Stacks tab of the FSP Configuration are set to the incorrect pin.	✓	✓	✓				FSP Configuration , Stacks

No	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
21	A bug that prevented the setup of PLL1.					✓		FSP Configuration, Clocks
22	A section cannot be copied successfully when its size is not a multiple of the alignment size.					✓		FSP Modules, BSP
23	Initial values of data placed in some sections were overwritten with 0.					✓		FSP Modules, BSP
24	Some sections were not initialized in the flash boot project.					✓		FSP Modules, BSP
25	DSMIF 0/1 error 1: trigger macros are not defined.		✓					FSP Modules, POEG
26	DSMIF 0/1 error 1 status macros are not defined.		✓					FSP Modules, POEG
27	Missing constraint for the DSMIF error trigger in channel 1 and channel 2.	✓	✓	✓	✓			FSP Modules, POEG
28	FreeRTOS+FAT format process is not executed correctly.	✓	✓	✓	✓	✓	✓	FSP Modules, FreeRTOS+FAT
29	Caution when specifying program placement in linker scripts.				✓		✓	Others, Linker script
30	In the secondary project for multiprocessing, no error occurs when there is a conflict in a resource used by the preceding project.				✓		✓	FSP Configuration, Stacks
31	Errors occur when setting ELC in r_gpt module.				✓		✓	FSP Configuration, Stacks
32	CR52 CPU1 of RZ/T2H and RZ/N2H is implemented to start programs from System SRAM instead of CPU1 ATCM.				✓		✓	Others, Linker script
33	No Error Occurs when entering out-of-range values for window parameters in r_pcie_ep and r_pcie_rc module configurations.				✓		✓	FSP Configuration, Stacks
34	The address space of DDR and PCIE cannot be used in the secondary (or later) projects with flash boot mode.				✓		✓	Others, Address space

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
35	r_gmac_b module cannot use zero-copy mode.				✓		✓	FSP Modules, GMAC
36	r_adc module does not support the calibration function.				✓			FSP Modules, ADC
37	The USB driver for the CA55 project does not work.				✓		✓	FSP Modules, USB
38	No error returns when entering the virtual addresses that cannot be translated to physical addresses as arguments.				✓		✓	FSP Modules, xSPI_OSPI, xSPI_QSPI, DMAC
39	The CA55 project with noncache sections aborts when debugging with flash boot mode on IAR EWARM.				✓		✓	FSP Modules, BSP
40	When changing the duty setting in the r_gpt module, there is a possibility that the duty may unintentionally become 100%.	✓	✓	✓	✓	✓	✓	FSP Modules, GPT
41	The CPU registers save and restore process cannot be performed correctly in FIQ_Handler for CA55 projects.				✓		✓	FSP Modules, BSP, FreeRTOS
42	MTU3 callback does not occur as expectation.				✓		✓	FSP Modules, MTU3
43	An undefined error of r_gpt module occurs when building a project.				✓		✓	FSP Modules, GPT
44	Pin names according to unit and channel numbers are not displayed in r_gpt module configurations.	✓	✓	✓	✓	✓	✓	FSP Configuration, Stacks
45	Using R_GPT_DutyCycleSet() with option both pins A and B cannot work properly.	✓	✓	✓	✓			FSP Modules, GPT
46	Parameter checking of R_ETHER_SELECTOR_Open() is not working properly.	✓	✓	✓	✓	✓	✓	FSP Modules, ETHER_SELECTOR
47	Parameter checking feature of R_GMAC_CallbackSet() is not working.	✓	✓	✓	✓	✓	✓	FSP Modules, ETHER_GMAC

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
48	r_usb_hhid module is not working properly.	✓	✓	✓	✓			FSP Modules, USB_HHID
49	The secondary and later projects may not work properly when multicore with flash boot mode.	✓		✓	✓		✓	FSP Modules, BSP, FreeRTOS
50	*length_bytes is an input/output parameter of R_GMAC_Read()/R_GMAC_B_Read() and must be reset before every call.	✓	✓	✓	✓	✓	✓	FSP Modules, GMAC, GMAC_B
51	When one unit is connected to multiple CS spaces, manual commands are issued only to the most recently opened CS space.	✓	✓	✓	✓	✓	✓	FSP Modules, XSPI_HYPER, XSPI_OSPI, XSPI_QSPI
52	Limit the requested transfer size to 2048 bytes or less when using DMA transfer mode of a USB peripheral.	✓	✓	✓	✓	✓	✓	FSP Modules, USB PCDC, USB PHID, USB PMSC, USB PVND, USB (FSP Configuration)

**No. 1 Resolved**

<b>Title</b>	“r_gmac” may be shown as “r_ether” incorrectly.
<b>Target</b>	RZ/T2M, RZ/N2L
<b>Category</b>	FSP Configuration, Stacks
<b>Description</b>	In the Stacks tab, “r_gmac” may be shown as “r_ether” incorrectly.
<b>Workaround</b>	Please read the “r_ether” as “r_gmac”.

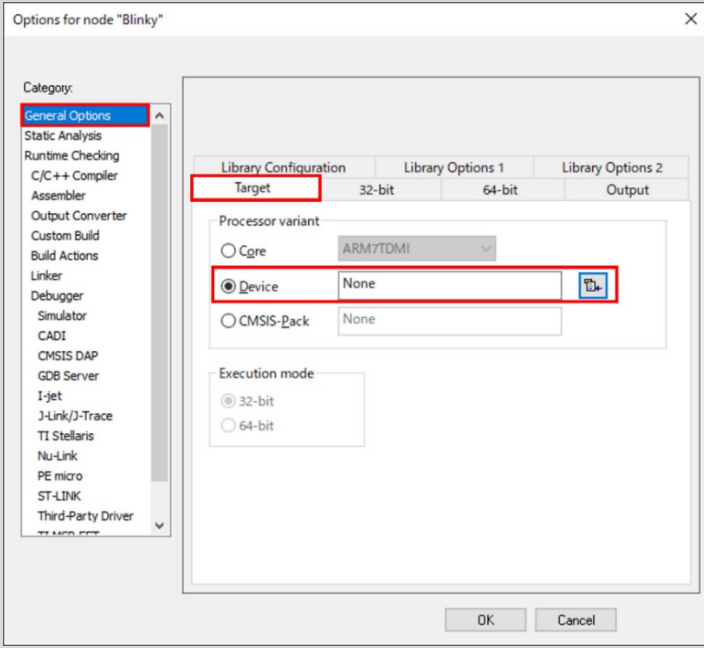
**No. 2 Resolved**

<b>Title</b>	“Edge” can be selected as Transfer End Interrupt Detect Type in “r_dmac”, but it cannot be used.
<b>Target</b>	RZ/T2M, RZ/T2L, RZ/N2L
<b>Category</b>	FSP Configuration, Stacks
<b>Description</b>	“Edge” of interrupt detect type is not available due to a change in hardware specifications.
<b>Workaround</b>	Please don't set Edge to Transfer End Interrupt Detect Type

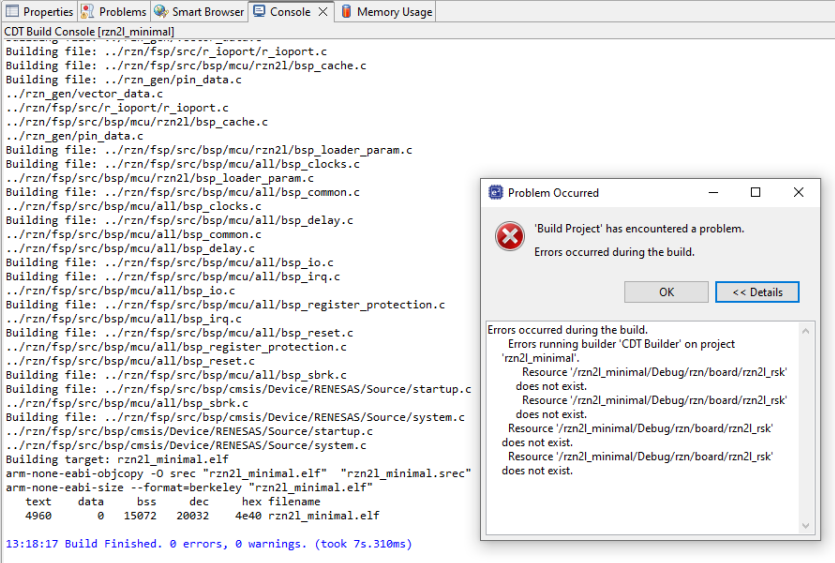
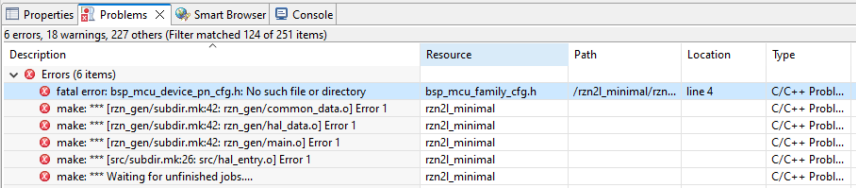
**No. 3 Resolved**

<b>Title</b>	<b>When the “Device” or “Board” selection in the BSP tab is changed, the BSP properties are sometimes configured incorrectly.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>FSP Configuration, BSP</b>
<b>Description</b>	When the “Device” or “Board” selection in the BSP tab is changed, the BSP properties are sometimes configured incorrectly. Once this issue occurs, the project cannot be fixed to the correct configuration.
<b>Workaround</b>	If changing the “Device” or “Board”, please reselect “FSP Version” from the drop-down list. If you want to change only the boot mode on the same board, please refer to Appendix. How to Change Boot Mode of the FSP Project

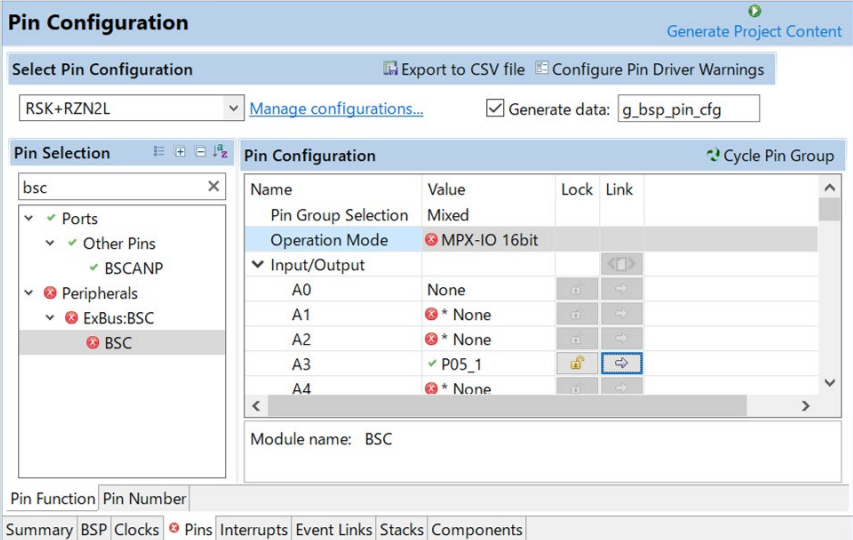
**No. 4 Resolved**

<b>Title</b>	<b>(FSP SC ONLY) The device name is not output correctly depending on the selected device.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2ME</b>
<b>Category</b>	<b>FSP Configuration, BSP</b>
<b>Description</b>	If you create a project by selecting a single core device (R9A07G075M01xxx, R9A07G075M05xxx), the device setting will be “None” when you open the project in IAR EWARM.
<b>Workaround</b>	<p>Please reselect the device name from the device list in the IAR EWARM project options. Options &gt; General Options &gt; Target &gt; Processor variant &gt; Device</p> 

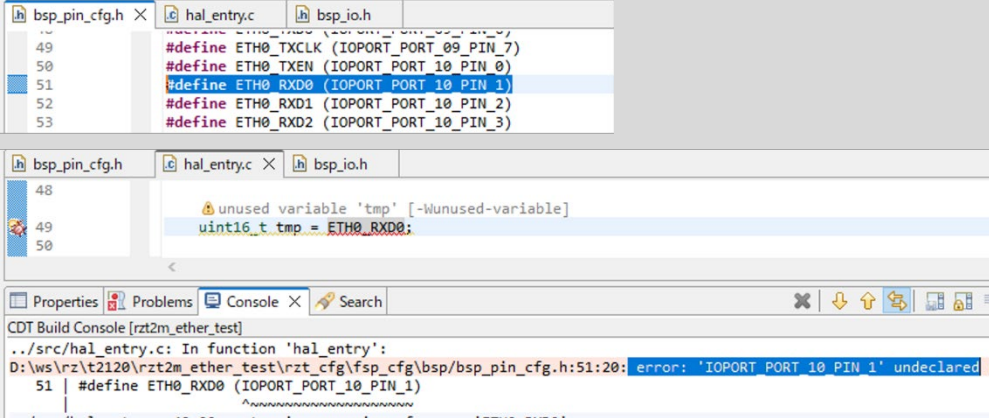
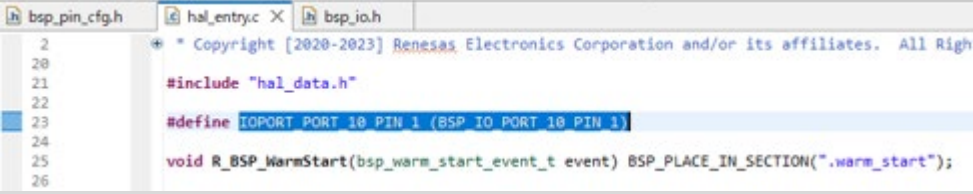
No. 5 Deleted due to one of the issues with Known Issues No. 3

<b>Title</b>	<b>Errors occur when changing board settings.</b>
<b>Target</b>	<b>RZ/T2L, RZ/N2L</b>
<b>Category</b>	<b>FSP Configuration, BSP</b>
<b>Description</b>	<p>Errors occur when changing board settings from RSK+RZN2L (RAM execution without flash memory) to RZN2L Custom User Board (RAM execution without flash memory) and from RSK+RZN2L (RAM execution without flash memory) to RZN2L Custom User Board (xSPI0 x1 boot mode).</p> <ol style="list-style-type: none"> <li>1. Changed from RSK+RZN2L (RAM execution without flash memory) to RZN2L Custom User Board (RAM execution without flash memory) In this case, the build is successful. But the following screen is displayed after the build.</li> </ol>  <ol style="list-style-type: none"> <li>2. Changed from RSK+RZN2L (RAM execution without flash memory) to RZN2L Custom User Board (xSPI0 x1 boot mode) bsp_mcu_device_pn_cfg.h is not generated, and a build error occurs.</li> </ol> 
<b>Workaround</b>	Please create a new project to change the board setting to RZN2L Custom User Board.

**No. 6 Resolved**

<b>Title</b>	<b>Pin configuration error occurs in the MPX-IO 16bit operating mode of “r_bsc”.</b>																																				
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/N2L</b>																																				
<b>Category</b>	<b>FSP Configuration, Pins</b>																																				
<b>Description</b>	<p>Pin assignment is an optional specification in the MPX-IO 16bit operation mode of “r_bsc”, but an error will occur if Input/Output A1-A13 is not set.</p>  <p>The screenshot shows the 'Pin Configuration' window for the 'bsc' module. The 'Operation Mode' is set to 'MPX-IO 16bit'. The 'Input/Output' table is as follows:</p> <table border="1" data-bbox="596 607 1193 846"> <thead> <tr> <th>Name</th> <th>Value</th> <th>Lock</th> <th>Link</th> </tr> </thead> <tbody> <tr> <td>Pin Group Selection</td> <td>Mixed</td> <td></td> <td></td> </tr> <tr> <td>Operation Mode</td> <td>MPX-IO 16bit</td> <td></td> <td></td> </tr> <tr> <td colspan="4">Input/Output</td> </tr> <tr> <td>A0</td> <td>None</td> <td></td> <td></td> </tr> <tr> <td>A1</td> <td>* None</td> <td></td> <td></td> </tr> <tr> <td>A2</td> <td>* None</td> <td></td> <td></td> </tr> <tr> <td>A3</td> <td>P05_1</td> <td></td> <td></td> </tr> <tr> <td>A4</td> <td>* None</td> <td></td> <td></td> </tr> </tbody> </table> <p>Module name: BSC</p>	Name	Value	Lock	Link	Pin Group Selection	Mixed			Operation Mode	MPX-IO 16bit			Input/Output				A0	None			A1	* None			A2	* None			A3	P05_1			A4	* None		
Name	Value	Lock	Link																																		
Pin Group Selection	Mixed																																				
Operation Mode	MPX-IO 16bit																																				
Input/Output																																					
A0	None																																				
A1	* None																																				
A2	* None																																				
A3	P05_1																																				
A4	* None																																				
<b>Workaround</b>	Please set “Custom” to “Operation Mode” of “r_bsc” in the Pins tab when you use the MPX-IO 16bit operation mode of “r_bsc”.																																				

**No. 7 Resolved**

<b>Title</b>	<b>Build error when using the definition name of input/output external pins for the module.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/N2L</b>
<b>Category</b>	<b>FSP Configuration, Pins</b>
<b>Description</b>	<p>After code generation, the definition of input/output external pins for the module is generated in fsp_cfg/bsp/bsp_pin_cfg.h, but the defined values are not defined in FSP. When using the defined name in a user application, a build error occurs.</p> 
<b>Workaround</b>	<p>Please add a definition to read IOPORT_PORT_mm_PIN_n as BSP_IO_PORT_mm_PIN_n in hal_entry. Do NOT edit file fsp_cfg/bsp/bsp_pin_cfg.h because its contents will be overwritten.</p> <p>An example of a setting: When using ETH0_RXD0 (IOPORT_PORT_10_PIN_1), add definition of #define IOPORT_PORT_10_PIN_1 (BSP_IO_PORT_10_PIN_1) in hal_entry.c.</p> 

**No. 8 Resolved**

<b>Issue</b>	<b>“R_SCI_UART_BaudCalculate()” of “r_sci_uart” module properly works ONLY when its clock source is SCInASYNCCLK and its frequency is 96MHz.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/N2L</b>
<b>Category</b>	<b>FSP Modules, Serial Communication Interface (SCI) UART</b>
<b>Description</b>	The “R_SCI_UART_BaudCalculate()” of “r_sci_uart” module works ONLY when its clock source is “SCInASYNCCLK” and its frequency is “96MHz”; therefore, when the module uses “PCLKM” as its clock source or the frequency is not 96MHz, the API function will not work properly.
<b>Workaround</b>	The clock source and frequency are limited in the Clocks and Stacks tab; therefore, you can NOT use the PCLKM clock and can NOT change the clock frequency.

**No. 9 Resolved**

<b>Issue</b>	<b>“R_SPI_CalculateBitrate()” of “r_spi” module properly works ONLY when its clock source is SPInASYNCCLK and its frequency is 96MHz.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/N2L</b>
<b>Category</b>	<b>FSP Modules, Serial Peripheral Interface</b>
<b>Description</b>	The “R_SPI_BaudCalculate()” of “r_spi” module works ONLY when its clock source is “SPInASYNCCLK” and its frequency is “96MHz”; therefore, when the module uses “PCLKM” as its clock source or the frequency is not 96MHz, the API function will not work properly.
<b>Workaround</b>	The clock source and frequency are limited in the Clocks and Stacks tab; therefore, you can NOT use the PCLKM clock and can NOT change the clock frequency.

**No. 10 Resolved**

<b>Issue</b>	<b>A warning occurs when building the “r_gmac” module with the GCC compiler.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L</b>
<b>Category</b>	<b>FSP Modules, Ethernet</b>
<b>Description</b>	The following warning occurs when building the “r_gmac” module with the GCC compiler. <pre>../rzt/fsp/src/r_gmac/r_gmac.c:2173:14: warning: the comparison will always evaluate as 'false' for the pointer operand in 'pp_phy_instance + (sizetype)(port * 12)' must not be NULL [-Waddress] 2173       if (NULL == pp_phy_instance[port])               ^~</pre>
<b>Workaround</b>	Please ignore this warning.

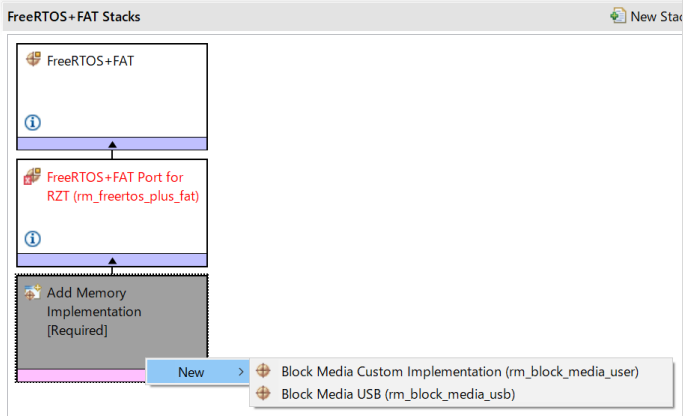
**No. 11 Resolved**

<b>Issue</b>	<b>In the FSP Documentation, there is an incorrect description. “API Reference &gt; Modules &gt; Ethernet PHY” page.</b>
<b>Target</b>	<b>RZ/T2M, RZ/N2L</b>
<b>Category</b>	<b>FSP Modules, Ethernet PHY</b>
<b>Description</b>	In the “API Reference > Modules > Ethernet PHY” page in FSP Documentation, the <b>default</b> column description of the “Select PHYs to use” configuration is incorrect.
<b>Workaround</b>	When reading the incorrect description, please replace the reading of it with the following.  [Error] <ul style="list-style-type: none"> <li>➤ config.driver.ether_phy.phy_lsi.default,config.driver.ether_phy.phy_lsi.0,config.driver.ether_phy.phy_lsi.1,config.driver.ether_phy.phy_lsi.2,config.driver.ether_phy.phy_lsi.3,config.driver.ether_phy.phy_lsi.</li> </ul> [Correction] <ul style="list-style-type: none"> <li>➤ All check boxes are enabled.</li> </ul>

## No. 12 Resolved

<b>Issue</b>	<b>The interrupt number cannot be successfully acquired by the R_FSP_CurrentIrqGet() when multiple interrupts occur.</b>
<b>Target</b>	<b>RZ/N2L</b>
<b>Category</b>	<b>FSP Modules, FreeRTOS</b>
<b>Description</b>	The interrupt number cannot be successfully acquired by the R_FSP_CurrentIrqGet() when using multiple interrupt handlers with different priority levels in FreeRTOS.
<b>Workaround</b>	<p>Please modify the following for the countermeasure against nested interrupts.</p> <p>Target File: port.c</p> <pre> void vApplicationIRQHandler (uint32_t ulCCIAR) {     #if 0         /* Re-enable interrupts. */         __asm("cpsie i");     #endif      bsp_common_interrupt_handler(ulCCIAR); } </pre> <p>Target File: bsp_irq.c</p> <pre> void bsp_common_interrupt_handler (uint32_t id) {     uint16_t gic_intid;      /* Get interrupt ID (GIC INTID). */     gic_intid = (uint16_t) (id &amp; BSP_PRV_ID_MASK);      #if VECTOR_DATA_IRQ_COUNT &gt; 0         if (BSP_CORTEX_VECTOR_TABLE_ENTRIES &lt;= gic_intid)         {             /* Remain the interrupt number */             g_current_interrupt_num[g_current_interrupt_pointer++] =                 (uint16_t) (gic_intid - BSP_CORTEX_VECTOR_TABLE_ENTRIES);              __asm volatile ("dmb");              #if 1                 /* Enable nested interrupt. */                 __asm volatile ("cpsie i");                 __asm volatile ("isb");             #endif              /* Branch to an interrupt handler. */             g_vector_table[(gic_intid - BSP_CORTEX_VECTOR_TABLE_ENTRIES)]();         }         else         #endif         {             /* Remain the interrupt number */             g_current_interrupt_num[g_current_interrupt_pointer++] = gic_intid;              __asm volatile ("dmb");              #if 1                 /* Enable nested interrupt. */                 __asm volatile ("cpsie i");                 __asm volatile ("isb");             #endif              /* Branch to an interrupt handler. */             g_sgi_ppi_vector_table[gic_intid]();         }          #if 1             /* Disable nested interrupt. */             __asm volatile ("cpsid i");             __asm volatile ("isb");         #endif          g_current_interrupt_pointer--;     } } </pre>

## No. 13 Resolved

<b>Issue</b>	<b>Block Media Custom Implementation can be selected as Memory Implementation for “rm_freertos_plus_fat” module, but it cannot be used.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/N2L</b>
<b>Category</b>	<b>FSP Configuration, Stacks</b>
<b>Description</b>	<p>In the Stacks tab of Configuration, Block Media Custom Implementation can be selected as Memory Implementation for the “rm_freertos_plus_fat” module, but it is unsupported and causes build errors.</p> 
<b>Workaround</b>	Please select Block Media USB as Memory Implementation for “rm_freertos_plus_fat” module.

## No. 14 Resolved

<b>Issue</b>	<b>The second argument of “r_mtu3” APIs does not match the common API.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>FSP Modules, MTU3</b>
<b>Description</b>	The second argument of these three APIs "R_MTU3_PeriodSet()", "R_MTU3_InfoGet()", and "R_MTU3_StatusGet()" of the "r_mtu3" module, do not match with the API in "r_timer api.h" header file
<b>Workaround</b>	<p>You cannot call these API by using the function pointer</p> <pre>g_timer0.p_api-&gt;periodSet() g_timer0.p_api-&gt; InfoGet() g_timer0.p_api-&gt; StatusGet()</pre> <p>Please use the API by calling it directly</p> <pre>R_MTU3_PeriodSet() R_MTU3_InfoGet() R_MTU3_StatusGet()</pre> <p>For reference on how to use these APIs, please refer to the <b>MTU3 Examples</b> in the <a href="#">FSP documentation</a>.</p>

## No. 15

<b>Issue</b>	<b>In multiprocessing, a configuration error occurs when the “r_gpt” module is used for both projects for CPU0 and CPU1.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2ME</b>
<b>Category</b>	<b>FSP Configuration, Stacks</b>
<b>Description</b>	When using the “r_gpt” module in the Stacks tab of both projects for CPU0 and CPU1, a configuration error occurs. The “r_gpt” module can only be used with either CPU0 or CPU1 in multiprocessing, regardless of the Unit or Channel number used.
<b>Workaround</b>	Please use the “r_gpt” module ONLY with either CPU0 or CPU1 in multiprocessing.

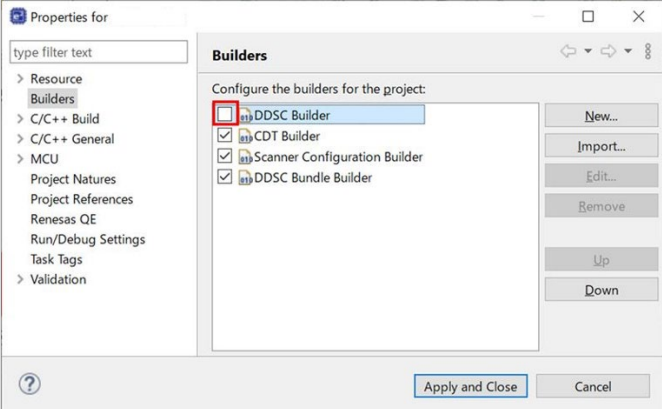
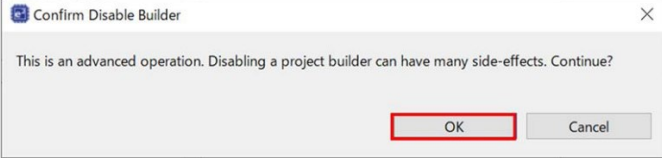
## No. 16 Resolved

<b>Issue</b>	<b>A project build error occurs when a 32-bit bus NOR flash and xSPI0 x8 boot modes are selected on the RZT Custom User Board.</b>
<b>Target</b>	<b>RZ/T2M</b>
<b>Category</b>	<b>FSP Configuration, BSP</b>
<b>Description</b>	When the following boards (boot mode) are selected, the required definitions are not generated, and a build error occurs. <ul style="list-style-type: none"> <li>• RZT Custom User Board (32-bit bus NOR flash boot mode)</li> <li>• RZT Custom User Board (xSPI0 x8 boot mode)</li> </ul>
<b>Workaround</b>	Please don't select 32-bit bus NOR flash and xSPI0 x8 boot modes on the RZT Custom User Board.

## No. 17 Resolved

<b>Issue</b>	<b>The secondary project for multiprocessing cannot be created when xSPI1 x1 boot modes are selected on the RZT Custom User Board.</b>
<b>Target</b>	<b>RZ/T2M</b>
<b>Category</b>	<b>FSP Configuration, BSP</b>
<b>Description</b>	When the following boards (boot mode) are selected for the primary project of multiprocessing, a variable required for multiprocessing is not defined, and the secondary project cannot be created. <ul style="list-style-type: none"> <li>• RZT Custom User Board (xSPI1 x1 boot mode)</li> </ul>
<b>Workaround</b>	Please don't select xSPI1 x1 boot modes on the RZT Custom User Board when multiprocessing.

No. 18 Resolved

<b>Issue</b>	<b>An incorrect value is set to a pin select value for MTU0-B/MTU6/MTU7 as the MTU3 output pin.</b>
<b>Target</b>	<b>RZ/T2L</b>
<b>Category</b>	<b>FSP Modules, POE3</b>
<b>Description</b>	Pin select value of MTU3 output pins used in FSP does not match the User's Manual: Hardware. Therefore, what you want to set up is not correctly described in the generated file of the FSP Configuration.
<b>Workaround</b>	<p>When using "r_poe3" and MTU0-B/MTU6/MTU7 as MTU3 output pin, please follow these four steps:</p> <ol style="list-style-type: none"> <li>1. Add "Port Output Enable 3 for MTU3 (r_poe3)" on the Stacks tab of FSP Configuration.</li> <li>2. Click the "Generate Project Content" button, and the "r_poe3" code is generated.</li> <li>3. Disable the code-generating function.</li> </ol> <p>After this setting, the code cannot be generated.          [For e<sup>2</sup> studio Smart Configurator]          Use the following settings to suppress the code-generating operation.          If this setting is missed, the code-generating operation is automatically executed at a clean build, and the changes made in step 4 revert to the original.</p> <p>a. Uncheck "Project Properties &gt; Builders &gt; DDSC Builder"</p>  <p>b. When unchecking it, the following message appears: click OK.</p>  <p>[For FSP SC (IAR EWARM)]          No need to set since code generation is not executed automatically.</p> <p>4. Modify definitions in rzt_gen/hal_data.c          Change the value of [module name]_pwm_pin_setting[] or [module name]_complementary_pwm_setting[1].pin_setting[X](X=0,1,2) according to the MTU3 output pins used. The tables below show the replacements required for each MTU3 output pin.</p> <p>File to be modified: rzt_gen/hal_data.c</p> <pre> /* Setting structure for pwm pin. */ static const poe3_pwm_pin_setting_t g_poe30_pwm_pin_setting[] = {   { .pwm_pin_select = POE3_PIN_SELECT_0, .hiz_output_enable = false },   { .pwm_pin_select = POE3_PIN_SELECT_0, .hiz_output_enable = false },   { .pwm_pin_select = POE3_PIN_SELECT_0, .hiz_output_enable = false },   { .pwm_pin_select = POE3_PIN_SELECT_0, .hiz_output_enable = false }, };  /* Setting structure for complementary pwm pin. */ static const poe3_complementary_pwm_setting_t g_poe30_complementary_pwm_setting[] = { </pre>

```

{ .pin_setting[0] =
{ .positive_pwm_pin_select = POE3_PIN_SELECT_0,
.....
  .pin_setting[X] =
{ .positive_pwm_pin_select = POE3_PIN_SELECT_0,
  .negative_pwm_pin_select = POE3_PIN_SELECT_0,
  .positive_pwm_pin_active_level = POE3_ACTIVE_LEVEL_SETTING_NONE,
  .negative_pwm_pin_active_level = POE3_ACTIVE_LEVEL_SETTING_NONE,
  .hiz_output_enable = false },
.....

```

## For MTU0-B

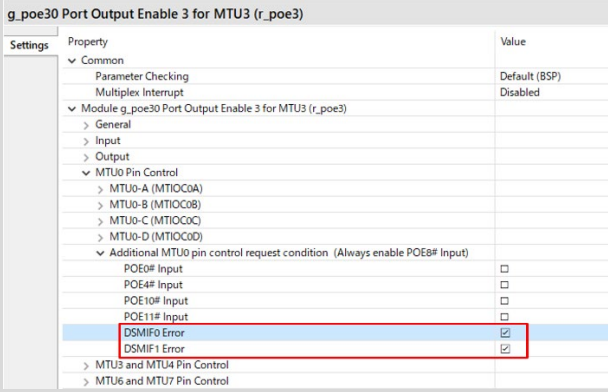
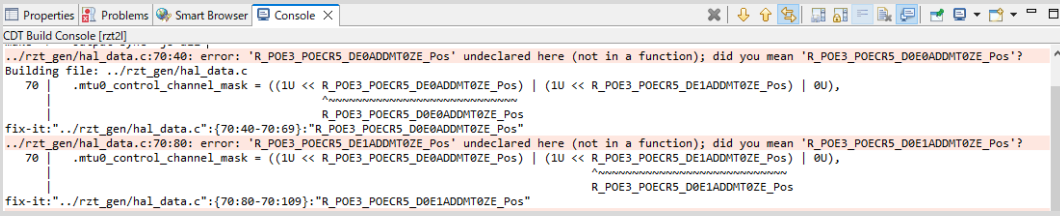
MTU3 output pin	Port	Location (Struct [module name]_pwm_pin_setting[])	Replace with
mtioc0b	P14_4	The <b>second</b> pwm_pin_select	.pwm_pin_select = <b>POE3_PIN_SELECT_1</b>
mtioc0b	P24_0	The <b>second</b> pwm_pin_select	.pwm_pin_select = <b>POE3_PIN_SELECT_2</b>
mtioc0b	P13_3	The <b>second</b> pwm_pin_select	.pwm_pin_select = <b>POE3_PIN_SELECT_3</b>

## For MTU6/MTU7

MTU3 output pin	Port	Location (Struct [module name]_complementary_pwm_setting[])	Replace with
mtioc6b	P21_2	the <b>second</b> pin_setting[0]	.positive_pwm_pin_select = <b>POE3_PIN_SELECT_1,</b>
mtioc6b	P08_5	the <b>second</b> pin_setting[0]	.positive_pwm_pin_select = <b>POE3_PIN_SELECT_2,</b>
mtioc6d	P21_4	the <b>second</b> pin_setting[0]	.negative_pwm_pin_select = <b>POE3_PIN_SELECT_1,</b>
mtioc6d	P08_7	the <b>second</b> pin_setting[0]	.negative_pwm_pin_select = <b>POE3_PIN_SELECT_2,</b>
mtioc7a	P21_5	the <b>second</b> pin_setting[1]	.positive_pwm_pin_select = <b>POE3_PIN_SELECT_1,</b>
mtioc7a	P09_0	the <b>second</b> pin_setting[1]	.positive_pwm_pin_select = <b>POE3_PIN_SELECT_2,</b>
mtioc7c	P21_7	the <b>second</b> pin_setting[1]	.negative_pwm_pin_select = <b>POE3_PIN_SELECT_1,</b>
mtioc7c	P09_2	the <b>second</b> pin_setting[1]	.negative_pwm_pin_select = <b>POE3_PIN_SELECT_2,</b>
mtioc7b	P21_6	the <b>second</b> pin_setting[2]	.positive_pwm_pin_select = <b>POE3_PIN_SELECT_1,</b>
mtioc7b	P09_1	the <b>second</b> pin_setting[2]	.positive_pwm_pin_select = <b>POE3_PIN_SELECT_2,</b>
mtioc7d	P22_0	the <b>second</b> pin_setting[2]	.negative_pwm_pin_select = <b>POE3_PIN_SELECT_1,</b>
mtioc7d	P09_3	the <b>second</b> pin_setting[2]	.negative_pwm_pin_select = <b>POE3_PIN_SELECT_2,</b>

Note: There are two pin\_setting[X] 's with the same number(X); modify the second one, not the first.

No. 19 Resolved

<b>Issue</b>	<b>Build error when using DSMIFn_ERR as an additional trigger for “r_poe3” module.</b>
<b>Target</b>	<b>RZ/T2L</b>
<b>Category</b>	<b>FSP Modules, POE3</b>
<b>Description</b>	<p>When using DSMIFn_ERR as an additional trigger in FSP Configuration, after code generation, values of the DSMIFn_ERR additional trigger for the module are generated in rzt_gen/hal_data.c. But the defined values are not defined in FSP, so a build error occurs.</p> <p>Property of r_poe3 stack</p>  <p>Build error log</p> 
<b>Workaround</b>	<p>When using “r_poe3” and DSMIFn_ERR(n=0,1) as an additional trigger, please follow these four steps:</p> <ol style="list-style-type: none"> <li>1-3. Refer to the workaround of No. 18.</li> <li>4. Modify definitions in rzt_gen/hal_data.c</li> </ol> <p>When using DSMIF 0 ERROR, DSMIF 1 ERROR for MTU0 additional trigger, modify:</p> <ul style="list-style-type: none"> <li>• R_POE3_POECR5_DE0ADDMT0ZE_Pos to R_POE3_POECR5_D0E0ADDMT0ZE_Pos</li> <li>• R_POE3_POECR5_DE1ADDMT0ZE_Pos to R_POE3_POECR5_D0E1ADDMT0ZE_Pos</li> </ul> <p>When using DSMIF 0 ERROR, DSMIF 1 ERROR for MTU3/4 additional trigger, modify:</p> <ul style="list-style-type: none"> <li>• R_POE3_POECR4_DE0ADDMT34ZE_Pos to R_POE3_POECR4_D0E0ADDMT34ZE_Pos</li> <li>• R_POE3_POECR4_DE1ADDMT34ZE_Pos to R_POE3_POECR4_D0E1ADDMT34ZE_Pos</li> </ul> <p>When using DSMIF 0 ERROR, DSMIF 1 ERROR for MTU6/7 additional trigger, modify:</p> <ul style="list-style-type: none"> <li>• R_POE3_POECR4_DE0ADDMT67ZE_Pos to R_POE3_POECR4_D0E0ADDMT67ZE_Pos</li> <li>• R_POE3_POECR4_DE1ADDMT67ZE_Pos to R_POE3_POECR4_D0E1ADDMT67ZE_Pos</li> </ul>

**No. 20 Resolved**

<b>Issue</b>	<b>Control setting values for MTU3 output pins in the Stacks tab of FSP Configuration are set to the incorrect pin.</b>		
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME</b>		
<b>Category</b>	<b>FSP Configuration, Stacks</b>		
<b>Description</b>	Settings for MTU4-B(MTIOC4B) and MTU4-D(MTIOC4D) in the Stacks tab property would be treated as settings for MTU4-A(MTIOC4A) and MTU4-C(MTIOC4C) in the generated file by Smart Configurator. All MTU3 output pins for MTU4 and MTU7 are the same as above.		
	Module [module name] Port Output Enable 3 for MTU3 (r_poe3) of Stacks property		
	Second Category	Third Category	Used for
	MTU3 and MTU4 Pin Control	MTU4-B(MTIOC4B) and MTU4-D(MTIOC4D)	MTU4-A(MTIOC4A) and MTU4-C(MTIOC4C)
	MTU3 and MTU4 Pin Control	MTU4-A(MTIOC4A) and MTU4-C(MTIOC4C)	MTU4-B(MTIOC4B) and MTU4-D(MTIOC4D)
	MTU6 and MTU7 Pin Control	MTU7-B(MTIOC7B) and MTU7-D(MTIOC7D)	MTU7-A(MTIOC7A) and MTU7-C(MTIOC7C)
MTU6 and MTU7 Pin Control	MTU7-A(MTIOC7A) and MTU7-C(MTIOC7C)	MTU7-B(MTIOC7B) and MTU7-D(MTIOC7D)	
<b>Workaround</b>	In the configuration of MTU4 and MTU7, please replace B/D and A/C.		

**No. 21 Resolved**

<b>Issue</b>	<b>A bug that prevented the setup of PLL1.</b>
<b>Target</b>	<b>RZ/N2L</b>
<b>Category</b>	<b>FSP Configuration, Clocks</b>
<b>Description</b>	The PLL1 state setting in the Clocks tab is invalid.
<b>Workaround</b>	Please don't use the PLL1 state setting.

**No. 22 Resolved**

<b>Issue</b>	<b>A section may not be copied correctly when it is not aligned and the section size is not a multiple of the alignment width.</b>
<b>Target</b>	<b>RZ/N2L</b>
<b>Category</b>	<b>FSP Modules, BSP</b>
<b>Description</b>	<p>Depending on a combination of section size and placement address, when the section is copied, some data from the following section may also be copied with it.</p> <p>A case that cannot be copied correctly:  The address of the .data_noncache section is 0x30190005, and its size is 0x23 bytes.  (Alignment size is 4 bytes.)</p> <pre>.data_noncache 0x30190005 0x23 ./src/hal_entry.o</pre>
<b>Workaround</b>	<p>All section sizes must be aligned by 4 bytes, and the data size should be a multiple of the number of bytes in the alignment. Section sizes can be found in the following files:</p> <ul style="list-style-type: none"> <li>gcc: [project name]/Debug/[project name].map</li> <li>iccar: [project name]/Debug/List/[project name].map</li> </ul>


**No. 23 Resolved**

<b>Issue</b>	<b>Initial values of data placed in some sections were overwritten with 0.</b>
<b>Target</b>	<b>RZ/N2L</b>
<b>Category</b>	<b>FSP Modules, BSP</b>
<b>Description</b>	<p>When selecting XXXXX (RAM execution without flash memory) as “Board” in the BSP tab of FSP Configuration, variables placed in the following sections are always cleared to zero.</p> <ul style="list-style-type: none"> <li>.dmac_link_mode</li> <li>.shared_noncache_buffer</li> <li>.noncache_buffer</li> </ul>
<b>Workaround</b>	Do NOT place data with initial values in the above sections.

**No. 24 Resolved**

<b>Issue</b>	<b>Some sections were not initialized in the flash boot project.</b>
<b>Target</b>	<b>RZ/N2L</b>
<b>Category</b>	<b>FSP Modules, BSP</b>
<b>Description</b>	<p>When selecting a flash boot mode as “Board” in the BSP tab of FSP Configuration, variables placed in the following sections are NOT initialized.</p> <p>Boards for flash boot mode</p> <ul style="list-style-type: none"> <li>• XXXXX (xSPI0 x1 boot mode)</li> <li>• XXXXX (16-bit bus NOR flash boot mode)</li> <li>• RZN2L Custom User Board (xSPI0 x8 boot mode)</li> <li>• RZN2L Custom User Board (xSPI1 x1 boot mode)</li> </ul> <p>Sections</p> <ul style="list-style-type: none"> <li>• .dmac_link_mode</li> <li>• .shared_noncache_buffer</li> <li>• .noncache_buffer</li> </ul>
<b>Workaround</b>	Please initialize the variables placed in the above sections in the user application.

**No. 25 Resolved**

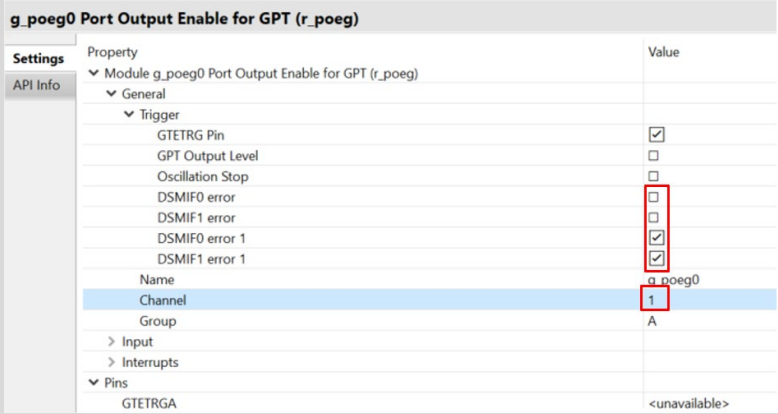
<b>Issue</b>	<b>DSMIF 0/1 error 1: Trigger macros are not defined.</b>
<b>Target</b>	<b>RZ/T2L</b>
<b>Category</b>	<b>FSP Modules, POEG</b>
<b>Description</b>	<p>In “bsp_override.h” of the rzt2l device, enum e_poeg_trigger, the definitions for DSMIF0 error 1 and DSMIF1 error 1 are missing.</p> <p>When setting it as below, build errors will occur.</p> <p>Property of r_poeg stack</p>  <p>Generated code by configurator</p> <pre>const poeg_cfg_t g_poeg0_cfg = { .trigger = (poeg_trigger_t) (POEG_TRIGGER_DERR0E_1   POEG_TRIGGER_DERR1E_1   POEG_TRIGGER_SOFTWARE), .polarity =   POEG_GTETRГ_POLARITY_ACTIVE_HIGH,</pre> <p>Build error log</p> <pre>make -r --output-sync -j8 all ../rzt_gen/hal_data.c:7:36: error: 'POEG_TRIGGER_DERR0E_1' undeclared here (not in a function); did you mean 'POEG_TRIGGER_DERR0E'? Building file: ../rzt_gen/hal_data.c 7     POEG_TRIGGER_PIN   POEG_TRIGGER_DERR0E_1   POEG_TRIGGER_DERR1E_1   POEG_TRIGGER_SOFTWARE),       ^~~~~~       POEG_TRIGGER_DERR0E fix-it: "../rzt_gen/hal_data.c":{7:36-7:57}: "POEG_TRIGGER_DERR0E" ../rzt_gen/hal_data.c:7:60: error: 'POEG_TRIGGER_DERR1E_1' undeclared here (not in a function); did you mean 'POEG_TRIGGER_DERR1E'? 7     POEG_TRIGGER_PIN   POEG_TRIGGER_DERR0E_1   POEG_TRIGGER_DERR1E_1   POEG_TRIGGER_SOFTWARE),       ^~~~~~       POEG_TRIGGER_DERR1E fix-it: "../rzt_gen/hal_data.c":{7:60-7:81}: "POEG_TRIGGER_DERR1E" make: *** [rzt_gen/subdir.mk:42: rzt_gen/hal_data.o] Error 1 make: *** Waiting for unfinished jobs....</pre>
<b>Workaround</b>	Add definition for DSMIF0 error 1 and DSMIF1 error 1 trigger in enum e_poeg_trigger.

	Location: rzt/bsp/src/bsp/mcu/rzt2l/bsp_override.h, enum e_poeg_trigger Add content: <pre>                 POEG_TRIGGER_DERR0E_1 = 1U &lt;&lt; 18,    ///&lt; Permit output disabled by DSMIF0 error 1 detection                 POEG_TRIGGER_DERR1E_1 = 1U &lt;&lt; 19,    ///&lt; Permit output disabled by DSMIF1 error 1 detection                 </pre>
--	---

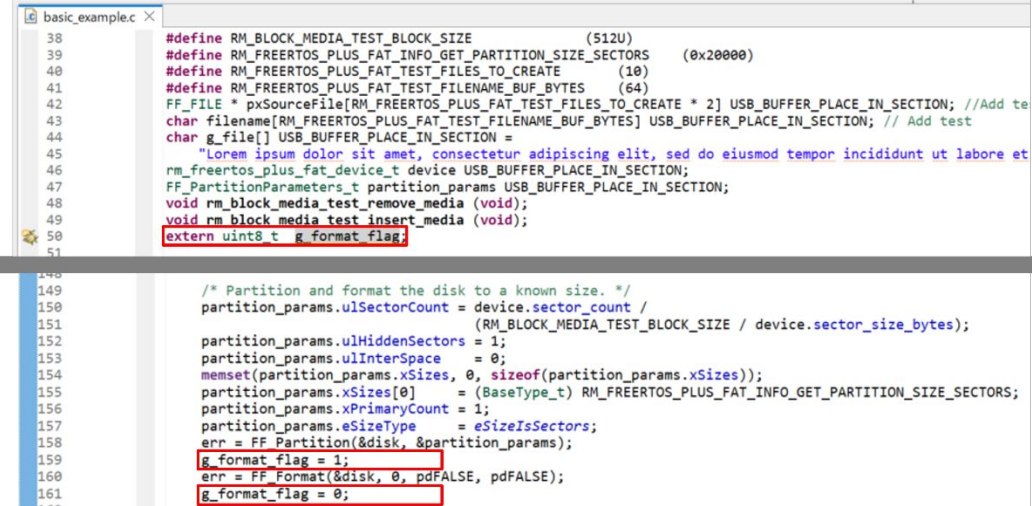
**No. 26 Resolved**

<b>Issue</b>	<b>DSMIF 0/1 error 1 status macros are not defined.</b>
<b>Target</b>	<b>RZ/T2L</b>
<b>Category</b>	<b>FSP Modules, POEG</b>
<b>Description</b>	In “bsp_override.h” of the rzt2l device, enum e_poeg_state, the definition for DSMIF0 error 1 state and DSMIF1 error 1 state are missing. When using R_POEG_StatusGet, <ul style="list-style-type: none"> <li>· If the POEG module is in state GPT output disabled due to DSMIF0 error 1, the p_status will be 0x100000 instead of POEG_STATE_DSMIF0_1_DISABLE_REQUEST.</li> <li>· If the POEG module is in state GPT output disabled due to DSMIF1 error 1, the p_status will be 0x200000 instead of POEG_STATE_DSMIF1_1_DISABLE_REQUEST.</li> </ul>
<b>Workaround</b>	When the POEG module is in the 'GPT output disabled' state due to a DSMIF0 error 1, assume that p_status = 0x100000 corresponds to POEG_STATE_DSMIF0_1_DISABLE_REQUEST. When the POEG module is in the 'GPT output disabled' state due to a DSMIF1 error 1, assume that p_status = 0x200000 corresponds to POEG_STATE_DSMIF1_1_DISABLE_REQUEST.

**No. 27 Resolved**

<b>Issue</b>	<b>Missing constraint for the DSMIF error trigger in channel 1 and channel 2.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2L, RZ/T2H</b>
<b>Category</b>	<b>FSP Modules, POEG</b>
<b>Description</b>	POEG channels 1 and 2 do not support the DSMIF error trigger in all RZT devices. But currently, there are no constraints to prevent configuring the DSMIF error trigger for channel 1 and channel 2. 
<b>Workaround</b>	Use the DSMIF error trigger for channel 0 only.

No. 28

<b>Issue</b>	<b>FreeRTOS+FAT format process is not executed correctly.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2L, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>FSP Modules, FreeRTOS+FAT</b>
<b>Description</b>	Executing the FF_Format function causes a USB AHB bus error, and the FreeRTOS+FAT format function processing is not executed correctly.
<b>Workaround</b>	<p>Please add g_format_flag before and after calling the FF_Format function in your application code.</p> <pre>extern uint8_t g_format_flag;  g_format_flag = 1; err = FF_Format(&amp;disk, 0, pdFALSE, pdFALSE) g_format_flag = 0;</pre> 

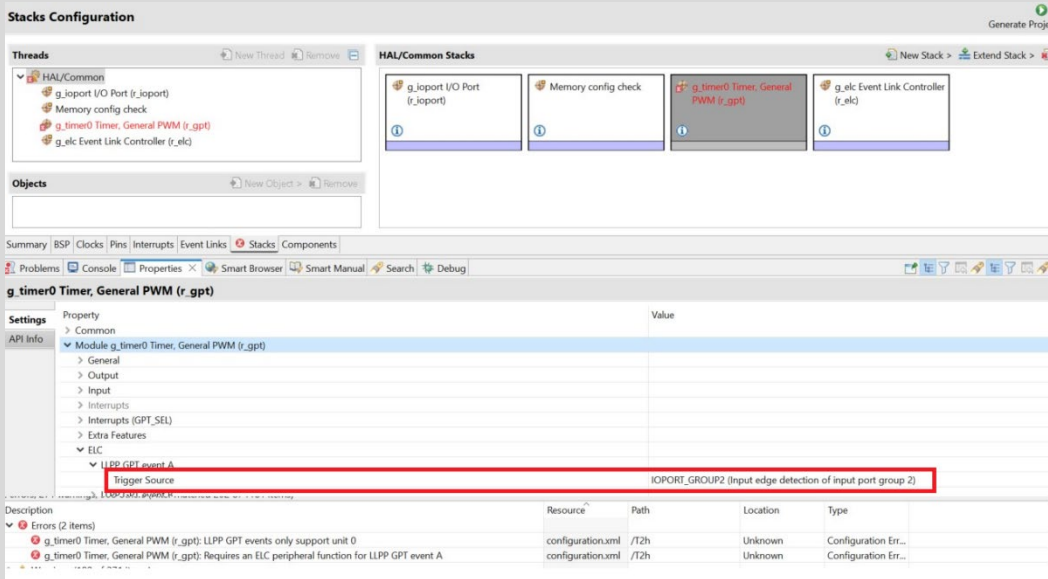
No. 29 Resolved

<b>Issue</b>	<b>Caution when specifying program placement in linker scripts</b>
<b>Target</b>	<b>RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>Others, Linker script</b>
<b>Description</b>	<p>The reserved area in the device address space cannot be used for a program placement, but no error occurs when placed there by a linker script.</p> <p>For example, the CA55 core has reserved areas ATCM and BTCM, the linker script is ready to place them there, and no error occurs when the following description exists.</p> <pre>.text TEXT_ADDRESS : AT (TEXT_ADDRESS) {     .....abbreviation..... }&gt; ATCM</pre>
<b>Workaround</b>	Do not specify that any program is to be placed at the reserved area in a linker script.

No. 30

<b>Issue</b>	<b>In the secondary project for multiprocessing, no error occurs when there is a conflict in a resource used by the preceding project.</b>
<b>Target</b>	<b>RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>FSP Configuration, Stacks</b>
<b>Description</b>	As noted in 6.2.4 Duplication of Resources, Smart Configurator has the feature to inform about resource duplication. However, it does not show an error when the secondary project for multiprocessing uses the same channel/unit of ADC, MTU3, GPT, and TSU_B as preceding projects.
<b>Workaround</b>	Please don't use the same channel/unit of ADC, MTU3, GPT, and TSU_B between the preceding project and the secondary project of RZ/T2H.

No. 31 Resolved

<b>Issue</b>	<b>Errors occur when setting ELC in the r_gpt module.</b>
<b>Target</b>	<b>RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>FSP Configuration, Stacks</b>
<b>Description</b>	<p>When setting the ELC event trigger source in the stack of the r_gpt module, errors will occur, and the value cannot be generated.</p> 
<b>Workaround</b>	<p>To use the ELC trigger source for the r_gpt module in code without configuring it in the FSP Configurator, follow these steps.</p> <ol style="list-style-type: none"> <li>1. Add the r_elc stack in the Stacks tab of FSP Configuration and include the r_elc header in your application.</li> <li>2. Call the following functions to initially set up the r_elc module.             <ol style="list-style-type: none"> <li>i. R_ELC_Open();</li> <li>ii. R_ELC_Enable();</li> </ol> </li> <li>3. Use the R_ELC_LinkSet(); function to configure the event triggers.</li> </ol> <p>Example:</p> <pre>R_ELC_Open(&amp;g_gpt_test_elc_ctrl, &amp;g_elc_cfg); R_ELC_Enable(&amp;g_gpt_test_elc_ctrl); R_ELC_LinkSet(&amp;g_gpt_test_elc_ctrl, ELC_PERIPHERAL_GPT00_A, ELC_EVENT_INTCPU0);</pre>

## No. 32 Resolved

<b>Issue</b>	<b>CR52 CPU1 of RZ/T2H and RZ/N2H is implemented to run the program from the System SRAM instead of the CPU1 ATCM.</b>
<b>Target</b>	<b>RZ/T2H(CR52), RZ/N2H(CR52)</b>
<b>Category</b>	<b>Others, Linker script</b>
<b>Description</b>	CR52 CPU1 of RZ/T2H has CPU1 ATCM and CPU1 BTCM as hardware, and when reset is released, the program runs from CPU1 ATCM. On the other hand, this FSP is implemented to run from System SRAM for RZ/T2H CR52 CPU1, like the program for RZ/T2M, and does not use CPU1 ATCM or CPU1 BTCM of RZ/T2H as the start of the program.
<b>Workaround</b>	Please consider using this implementation that uses System SRAM.

## No. 33

<b>Issue</b>	<b>No Error Occurs when entering out-of-range values for window parameters in the r_pcie_ep and r_pcie_rc module configurations.</b>																												
<b>Target</b>	<b>RZ/T2H, RZ/N2H</b>																												
<b>Category</b>	<b>FSP Configuration, Stacks</b>																												
<b>Description</b>	<p>There is a problem with the r_pcie_ep and r_pcie_rc module configuration screens, where entering unsupported values does not generate an error, and the code can be generated with unusable values.</p> <p>The configuration items for which the validity judgment of the input value does not work are as follows.</p> <table border="1"> <thead> <tr> <th>Configuration item</th> <th>r_pcie_ep</th> <th>r_pcie_rc</th> </tr> </thead> <tbody> <tr> <td>AXI Window Base</td> <td>✓</td> <td>✓</td> </tr> <tr> <td>AXI Window Mask</td> <td>✓</td> <td>✓</td> </tr> <tr> <td>AXI Window Destination</td> <td>✓</td> <td>✓</td> </tr> <tr> <td>PCIe Window Base</td> <td>✓</td> <td>✓</td> </tr> <tr> <td>PCIe Window Mask</td> <td>✓</td> <td>✓</td> </tr> <tr> <td>PCIe Window Destination</td> <td>✓</td> <td>✓</td> </tr> <tr> <td>MSI Receive Window Address</td> <td>-</td> <td>✓</td> </tr> <tr> <td>MSI Receive Window Mask</td> <td>-</td> <td>✓</td> </tr> </tbody> </table>		Configuration item	r_pcie_ep	r_pcie_rc	AXI Window Base	✓	✓	AXI Window Mask	✓	✓	AXI Window Destination	✓	✓	PCIe Window Base	✓	✓	PCIe Window Mask	✓	✓	PCIe Window Destination	✓	✓	MSI Receive Window Address	-	✓	MSI Receive Window Mask	-	✓
Configuration item	r_pcie_ep	r_pcie_rc																											
AXI Window Base	✓	✓																											
AXI Window Mask	✓	✓																											
AXI Window Destination	✓	✓																											
PCIe Window Base	✓	✓																											
PCIe Window Mask	✓	✓																											
PCIe Window Destination	✓	✓																											
MSI Receive Window Address	-	✓																											
MSI Receive Window Mask	-	✓																											
<b>Workaround</b>	<p>Please enter the values so that they meet the input value conditions for each item.</p> <p>AXI Window Base and PCIe Window Base - "greater than or equal to 0" and "address is 4Kbyte aligned"</p> <p>AXI Window Mask and PCIe Window Mask - "greater than or equal to 0", "the lower 12 bits are 1", and "the 63rd bit is 0"</p> <p>AXI Window Destination and PCIe Window Destination - "greater than or equal to 0" and "address is 4Kbyte aligned"</p> <p>MSI Receive Window Mask - "greater than or equal to 0" and "the lower 2 bits are 1"</p> <p>MSI Receive Window Address - Align according to "MSI Receive Window Mask".</p>																												

## No. 34

<b>Issue</b>	<b>DDR and PCIE0/1 memory cannot be used in secondary (or later) projects with flash boot mode.</b>
<b>Target</b>	<b>RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>Others, Address space</b>
<b>Description</b>	If you use DDR or PCIE0/1 memory in a secondary (or later) project with flash boot mode, the binary file will be of a huge size. Therefore, multicore operation is not possible.
<b>Workaround</b>	Secondary (or later) projects with flash boot mode do not use DDR or PCIE0/1 memory.

## No. 35

<b>Issue</b>	<b>r_gmac_b module cannot use zero-copy mode.</b>
<b>Target</b>	<b>RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>FSP Modules, GMAC</b>
<b>Description</b>	r_gmac_b module cannot use zero-copy mode. The "Zero-copy mode" setting in the r_gmac_b configuration cannot be changed from the default "Disable."
<b>Workaround</b>	Please use the standard buffers provided by the r_gmac_b module for transmit and receive buffers.

## No. 36 Resolved

<b>Issue</b>	<b>r_adc module does not support the calibration function.</b>
<b>Target</b>	<b>RZ/T2H</b>
<b>Category</b>	<b>FSP Modules, ADC</b>
<b>Description</b>	The 12-bit A/D converter needs to be calibrated before A/D conversion after reset is released. However, since the FSP does not support the calibration function, the accuracy shown in the electrical characteristics chapter of the device user's manual cannot be guaranteed.
<b>Workaround</b>	<p>This will be implemented in the next version of FSP.</p> <p>As a temporary measure, the calibration process must be implemented in the user application before the R_ADC_Open function is executed.</p> <p>The following is an example of implementation. (In the case of ADC Unit 2)</p> <pre>#define ADC_ADCCALCTL_SET_CAL 1U  /* Release module stop for ADC12 */ R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_LPC_RESET); R_BSP_MODULE_START(FSP_IP_ADC12, 2); R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_LPC_RESET);  R_BSP_SoftwareDelay(1, BSP_DELAY_UNITS_MICROSECONDS); /* Write ADCCALCTL.CAL bit to 1 to start calibration. */ R_ADC122-&gt;ADCCALCTL_b.CAL = ADC_ADCCALCTL_SET_CAL;  /* Poll ADCCALCTL.CAL_RDY bit until it is changed to 1. */ FSP_HARDWARE_REGISTER_WAIT(R_ADC122-&gt;ADCCALCTL_b.CAL_RDY, 1U); /* Confirm ADCCALCTL.CAL_ERR bit is 0.*/ FSP_HARDWARE_REGISTER_WAIT(R_ADC122-&gt;ADCCALCTL_b.CAL_ERR, 0U); /* Write ADCCALCTL.CAL bit to 0 */ R_ADC122-&gt;ADCCALCTL_b.CAL = 0U;  /* Initializing the ADC module */ R_ADC_Open(&amp;g_adc0_ctrl, &amp;g_adc0_cfg);</pre>

## No. 37 Resolved

<b>Issue</b>	<b>The USB driver for the CA55 project does not work.</b>
<b>Target</b>	<b>RZ/T2H(CA55), RZ/N2H(CA55)</b>
<b>Category</b>	<b>FSP Modules, USB</b>
<b>Description</b>	The R_BSP_MmuPatoVA function executed from USB driver (r_usb_hmsc, r_usb_hcdc, r_usb_hhid modules) in a CA55 project fails to perform the expected address translation, resulting in a USB transfer failure.
<b>Workaround</b>	<p>Please modify \rzt\fsp\src\r_usb_basic\src\driver\r_usb_mmu_pa_to_va.c as follows.</p> <ol style="list-style-type: none"> <li>1. Modify the r_usb_pa_to_va function.</li> </ol> <pre>uint64_t r_usb_pa_to_va (uint64_t paddr) {     uint64_t vaddr = 0;      #if defined(BSP_CFG_CORE_CA55)          /* Converts a physical address to a virtual address. */         if (FSP_SUCCESS != R_BSP_MmuPatoVa(paddr, &amp;vaddr, BSP_MMU_CONVERSION_NON_CACHE))         {             /* On error, returns the physical address without conversion. */             vaddr = paddr;         }     #else /* #if defined(BSP_CFG_CORE_CA55) */         vaddr = paddr;     #endif /* #if defined(BSP_CFG_CORE_CA55) */      return vaddr; } /* End of function r_usb_pa_to_va() */</pre> <ol style="list-style-type: none"> <li>2. Modify the r_usb_va_to_pa function.</li> </ol> <pre>uint64_t r_usb_va_to_pa (uint64_t vaddr) {     uint64_t paddr = 0;      #if defined(BSP_CFG_CORE_CA55)          /* Converts a virtual address to a physical address. */         if (FSP_SUCCESS != R_BSP_MmuVatoPa(vaddr, &amp;paddr))         {             /* On error, returns the virtual address without conversion. */             paddr = vaddr;         }     #else /* #if defined(BSP_CFG_CORE_CA55) */         paddr = vaddr;     #endif /* #if defined(BSP_CFG_CORE_CA55) */      return paddr; }</pre>

**No. 38 Resolved**

<b>Issue</b>	<b>No error returns when entering the virtual addresses that cannot be translated to physical addresses as arguments.</b>
<b>Target</b>	<b>RZ/T2H(CA55), RZ/N2H(CA55)</b>
<b>Category</b>	<b>FSP Modules, xSPI_OSPI, xSPI_QSPI, DMAC</b>
<b>Description</b>	<p>In a CA55 project, there is a problem with the "r_xspi_qspi", "r_xspi_ospi", and "r_dmac" modules that no error returns when entering virtual addresses that cause translation error in MMU as arguments.</p> <p>The following functions have a problem.</p> <ul style="list-style-type: none"> <li>-R_XSPI_QSPI_Write()</li> <li>-R_XSPI_OSPI_Write()</li> <li>-R_DMACE_Open()</li> <li>-R_DMACE_Reconfigure()</li> <li>-R_DMACE_Reload()</li> <li>-R_DMACE_LinkDescriptorSet()</li> </ul>
<b>Workaround</b>	Do not enter the virtual addresses that cannot be translated to physical addresses as arguments.

## No. 39 Resolved

<b>Issue</b>	<b>The CA55 project with non-cache sections aborts when debugging with flash boot mode on IAR EWARM</b>	
<b>Target</b>	<b>RZ/T2H(CA55), RZ/N2H(CA55)</b>	
<b>Category</b>	<b>FSP Modules, BSP</b>	
<b>Description</b>	When performing debugging of a flash boot CA55 project with non-cache sections on IAR EWARM, executing the CA55 project will abort. This is due to cache initialization.	
<b>Workaround</b>	<p>Follow the steps below</p> <ol style="list-style-type: none"> <li>1. Add "<code>__set_ICIALLU(0);</code>" and "<code>__ISB();</code>" to <code>bsp_memory_protect_setting</code> in <code>XXX/fsp/src/bsp/cmsis/Device/RENESAS/Source/ca/system_core.c</code>. (XXX= rzt, rzn) <pre>void bsp_memory_protect_setting (void) {     bsp_mmu_configure();     R_BSP_CacheInvalidateAll();      R_BSP_CacheEnableMemoryProtect();     R_BSP_CacheEnableInst();     R_BSP_CacheEnableData();      __set_ICIALLU(0);     __ISB(); }</pre> </li> <li>2. Move "<code>__set_ICIALLU(0);</code>" in <code>R_BSP_CacheInvalidateAll</code> in <code>XXX/fsp/src/bsp/mcu/all/bsp_cache.c</code> (XXX= rzt, rzn) to just before "<code>__asm volatile("ISB SY");</code>" at the end.</li> </ol>	
	<pre>15  uintptr_t ccsidr_associativity_value; 16  uintptr_t ccsidr_associativity_msb; 17  uintptr_t ccsidr_numsets; 18  uintptr_t ccsidr_numsets_total; 19 20  uintptr_t dcsiw; 21 22  __asm volatile ("DSB SY"); 23 24  __set_ICIALLU(0); 25 26  __asm volatile ("DMB SY"); 27 28  /* Reads the maximum level of cache implemented */ 29  clidr = __get_CLIDR(); 30  clidr_loc = (clidr &gt;&gt; BSP_PRIV_CLIDR_LOC_OFFSET) &amp; BSP_PRIV_CLIDR_LOC_MASK; 31 32  /* If the cache does not exist, do not process */ 33  if (0 != clidr_loc) 34  { 105     } 106     else 107     { 108         /* Do Nothing */ 109     } 110 } 111 112 __asm volatile ("DSB SY"); 113 114 __asm volatile ("ISB SY"); 115 } 116 else 117 { 118     /* Do Nothing */ 119 }</pre>	<pre>15  uintptr_t ccsidr_associativity_value; 16  uintptr_t ccsidr_associativity_msb; 17  uintptr_t ccsidr_numsets; 18  uintptr_t ccsidr_numsets_total; 19 20  uintptr_t dcsiw; 21 22  __asm volatile ("DSB SY"); 23 24  __asm volatile ("DMB SY"); 25 26  /* Reads the maximum level of cache implemented */ 27  clidr = __get_CLIDR(); 28  clidr_loc = (clidr &gt;&gt; BSP_PRIV_CLIDR_LOC_OFFSET) &amp; BSP_PRIV_CLIDR_LOC_MASK; 29 30  /* If the cache does not exist, do not process */ 31  if (0 != clidr_loc) 32  { 104     } 105     else 106     { 107         /* Do Nothing */ 108     } 109 } 110 111 __asm volatile ("DSB SY"); 112 113 __set_ICIALLU(0); 114 115 __asm volatile ("ISB SY"); 116 } 117 else 118 { 119     /* Do Nothing */ 120 }</pre>

## No. 40 Resolved

<b>Issue</b>	<b>When changing the duty setting in the r_gpt module, there is a possibility that the duty may unintentionally become 100%.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>FSP Modules, GPT</b>
<b>Description</b>	When changing the PWM period with "R_GPT_PeriodSet()" and the duty with "R_GPT_DutyCycleSet()" while the GPT is running, the duty may unintentionally become 100% depending on both setting values. The reason is that when "gpt_calculate_duty_cycle()" in "R_GPT_DutyCycleSet()" performs a comparison calculation of the duty and period, the old period value of the GTPR register is mistakenly referenced instead of the current period value of the GTPBR (buffer) register.
<b>Workaround</b>	<p>Please correct the reading of the period in the "gpt_calculate_duty_cycle()" to GTPBR instead of GTPR in XXX/fsp/src/r_gpt/r_gpt.c. (XXX= rzt, rzn)</p> <pre> static void gpt_calculate_duty_cycle (gpt_instance_ctrl_t * const p_instance_ctrl,                                      uint32_t const      duty_cycle_counts,                                      gpt_prv_duty_registers_t * p_duty_reg) {     /* Determine the current period. The actual period is one cycle longer than the register value for saw waves     * and twice the register value for triangle waves. Reference section 19.2.21 "General PWM Timer Cycle Setting     * Register (GTPR)". The setting passed to the configuration is expected to be half the desired duty cycle for     * triangle waves. */     uint32_t current_period = p_instance_ctrl-&gt;p_reg-&gt;GTPR;     #if GPT_PRV_EXTRA_FEATURES_ENABLED == GPT_CFG_OUTPUT_SUPPORT_ENABLE     if (p_instance_ctrl-&gt;p_cfg-&gt;mode &lt; TIMER_MODE_TRIANGLE_WAVE_SYMMETRIC_PWM)     #endif </pre>

## No. 41 Resolved

<b>Issue</b>	<b>CPU registers save and restore process cannot be performed correctly in the FIQ_Handler for CA55 projects.</b>
<b>Target</b>	<b>RZ/T2H(CA55), RZ/N2H(CA55)</b>
<b>Category</b>	<b>FSP Modules, BSP, FreeRTOS</b>
<b>Description</b>	Some of the CPU registers are not saved and restored in FIQ_Handler. Therefore, the value of registers may be partially corrupted before or after the FIQ interrupt occurs.
<b>Workaround</b>	<ul style="list-style-type: none"> <li>When NOT using FreeRTOS Please modify XXX/fsp/src/bsp/cmsis/Device/RENESAS/Source/ca/startup_core.c (XXX= rzt, rzn) as follows.</li> </ul> <pre> __WEAK void FIQ_Handler (void) {     __asm volatile (         "STP x30, XZR, [SP, #-0x10]! \n"         "STP x28, x29, [SP, #-0x10]! \n"         "STP x26, x27, [SP, #-0x10]! \n"         "STP x24, x25, [SP, #-0x10]! \n"         "STP x22, x23, [SP, #-0x10]! \n"         "STP x20, x21, [SP, #-0x10]! \n"         "STP x18, x19, [SP, #-0x10]! \n"         "STP x16, x17, [SP, #-0x10]! \n"         "STP x14, x15, [SP, #-0x10]! \n"         "STP x12, x13, [SP, #-0x10]! \n"         "STP x10, x11, [SP, #-0x10]! \n"         "STP x8, x9, [SP, #-0x10]! \n"         "STP x6, x7, [SP, #-0x10]! \n"         "STP x4, x5, [SP, #-0x10]! \n"         "STP x2, x3, [SP, #-0x10]! \n"         "STP x0, x1, [SP, #-0x10]! \n"         #if __FPU_USED         "MRS x0, FPCR \n"         "MRS x1, FPSR \n"         "STP x0, x1, [SP, #-0x10]! \n"         "STP q30, q31, [SP, #-0x20]! \n"         "STP q28, q29, [SP, #-0x20]! \n"         "STP q26, q27, [SP, #-0x20]! \n" </pre>

```

"STP q24, q25, [SP, #-0x20]! \n"
"STP q22, q23, [SP, #-0x20]! \n"
"STP q20, q21, [SP, #-0x20]! \n"
"STP q18, q19, [SP, #-0x20]! \n"
"STP q16, q17, [SP, #-0x20]! \n"
"STP q14, q15, [SP, #-0x20]! \n"
"STP q12, q13, [SP, #-0x20]! \n"
"STP q10, q11, [SP, #-0x20]! \n"
"STP q8, q9, [SP, #-0x20]! \n"
"STP q6, q7, [SP, #-0x20]! \n"
"STP q4, q5, [SP, #-0x20]! \n"
"STP q2, q3, [SP, #-0x20]! \n"
"STP q0, q1, [SP, #-0x20]! \n"
#endif
~~~~~
#if __FPU_USED
"LDP q0, q1, [sp], #0x20 \n"
"LDP q2, q3, [sp], #0x20 \n"
"LDP q4, q5, [sp], #0x20 \n"
"LDP q6, q7, [sp], #0x20 \n"
"LDP q8, q9, [sp], #0x20 \n"
"LDP q10, q11, [sp], #0x20 \n"
"LDP q12, q13, [sp], #0x20 \n"
"LDP q14, q15, [sp], #0x20 \n"
"LDP q16, q17, [sp], #0x20 \n"
"LDP q18, q19, [sp], #0x20 \n"
"LDP q20, q21, [sp], #0x20 \n"
"LDP q22, q23, [sp], #0x20 \n"
"LDP q24, q25, [sp], #0x20 \n"
"LDP q26, q27, [sp], #0x20 \n"
"LDP q28, q29, [sp], #0x20 \n"
"LDP q30, q31, [sp], #0x20 \n"
"LDP x0, x1, [sp], #0x10 \n"
"MSR FPCR, x0 \n"
"MSR FPSR, x1 \n"
#endif

"LDP x0, x1, [sp], #0x10 \n"
"LDP x2, x3, [sp], #0x10 \n"
"LDP x4, x5, [sp], #0x10 \n"
"LDP x6, x7, [sp], #0x10 \n"
"LDP x8, x9, [sp], #0x10 \n"
"LDP x10, x11, [sp], #0x10 \n"
"LDP x12, x13, [sp], #0x10 \n"
"LDP x14, x15, [sp], #0x10 \n"
"LDP x16, x17, [sp], #0x10 \n"
"LDP x18, x19, [sp], #0x10 \n"
"LDP x20, x21, [sp], #0x10 \n"
"LDP x22, x23, [sp], #0x10 \n"
"LDP x24, x25, [sp], #0x10 \n"
"LDP x26, x27, [sp], #0x10 \n"
"LDP x28, x29, [sp], #0x10 \n"
"LDP x30, XZR, [sp], #0x10 \n"

"ERET \n"
::: "memory");
}

```

- When using FreeRTOS  
Please modify XXX/fsp/src/rm\_freertos\_port/ca/port.c (XXX= rzt, rzn) as follows.

```

BSP_ATTRIBUTE_STACKLESS void FIQ_Handler (void)
{
/* Save volatile registers. */
asm volatile (
"STP x30, XZR, [SP, #-0x10]! \n"
"STP x28, x29, [SP, #-0x10]! \n"
"STP x26, x27, [SP, #-0x10]! \n"
"STP x24, x25, [SP, #-0x10]! \n"
"STP x22, x23, [SP, #-0x10]! \n"
"STP x20, x21, [SP, #-0x10]! \n"
"STP x18, x19, [SP, #-0x10]! \n"
"STP x16, x17, [SP, #-0x10]! \n"
"STP x14, x15, [SP, #-0x10]! \n"
"STP x12, x13, [SP, #-0x10]! \n"
"STP x10, x11, [SP, #-0x10]! \n"
"STP x8, x9, [SP, #-0x10]! \n"
"STP x6, x7, [SP, #-0x10]! \n"
"STP x4, x5, [SP, #-0x10]! \n"
"STP x2, x3, [SP, #-0x10]! \n"
"STP x0, x1, [SP, #-0x10]! \n"
#endif
#if __FPU_USED
"MRS x0, FPCR \n"

```

```

"MRS x1, FPSR                                \n"
"STP x0, x1, [SP, #-0x10]!                  \n"
"STP q30, q31, [SP, #-0x20]!                \n"
"STP q28, q29, [SP, #-0x20]!                \n"
"STP q26, q27, [SP, #-0x20]!                \n"
"STP q24, q25, [SP, #-0x20]!                \n"
"STP q22, q23, [SP, #-0x20]!                \n"
"STP q20, q21, [SP, #-0x20]!                \n"
"STP q18, q19, [SP, #-0x20]!                \n"
"STP q16, q17, [SP, #-0x20]!                \n"
"STP q14, q15, [SP, #-0x20]!                \n"
"STP q12, q13, [SP, #-0x20]!                \n"
"STP q10, q11, [SP, #-0x20]!                \n"
"STP q8, q9, [SP, #-0x20]!                  \n"
"STP q6, q7, [SP, #-0x20]!                  \n"
"STP q4, q5, [SP, #-0x20]!                  \n"
"STP q2, q3, [SP, #-0x20]!                  \n"
"STP q0, q1, [SP, #-0x20]!                  \n"
#endif
/* Save the SPSR and ELR. */
~~~~~
"DSB SY                                     \n"
"ISB SY                                     \n"
#if __FPU_USED
"LDP q0, q1, [sp], #0x20                     \n"
"LDP q2, q3, [sp], #0x20                     \n"
"LDP q4, q5, [sp], #0x20                     \n"
"LDP q6, q7, [sp], #0x20                     \n"
"LDP q8, q9, [sp], #0x20                     \n"
"LDP q10, q11, [sp], #0x20                   \n"
"LDP q12, q13, [sp], #0x20                   \n"
"LDP q14, q15, [sp], #0x20                   \n"
"LDP q16, q17, [sp], #0x20                   \n"
"LDP q18, q19, [sp], #0x20                   \n"
"LDP q20, q21, [sp], #0x20                   \n"
"LDP q22, q23, [sp], #0x20                   \n"
"LDP q24, q25, [sp], #0x20                   \n"
"LDP q26, q27, [sp], #0x20                   \n"
"LDP q28, q29, [sp], #0x20                   \n"
"LDP q30, q31, [sp], #0x20                   \n"
"LDP x0, x1, [sp], #0x10                      \n"
"MSR FPCR, x0                                 \n"
"MSR FPSR, x1                                 \n"
#endif

"LDP x0, x1, [sp], #0x10                      \n"
"LDP x2, x3, [sp], #0x10                      \n"
"LDP x4, x5, [sp], #0x10                      \n"
"LDP x6, x7, [sp], #0x10                      \n"
"LDP x8, x9, [sp], #0x10                      \n"
"LDP x10, x11, [sp], #0x10                    \n"
"LDP x12, x13, [sp], #0x10                    \n"
"LDP x14, x15, [sp], #0x10                    \n"
"LDP x16, x17, [sp], #0x10                    \n"
"LDP x18, x19, [sp], #0x10                    \n"
"LDP x20, x21, [sp], #0x10                    \n"
"LDP x22, x23, [sp], #0x10                    \n"
"LDP x24, x25, [sp], #0x10                    \n"
"LDP x26, x27, [sp], #0x10                    \n"
"LDP x28, x29, [sp], #0x10                    \n"
"LDP x30, XZR, [sp], #0x10                    \n"
::: "memory");

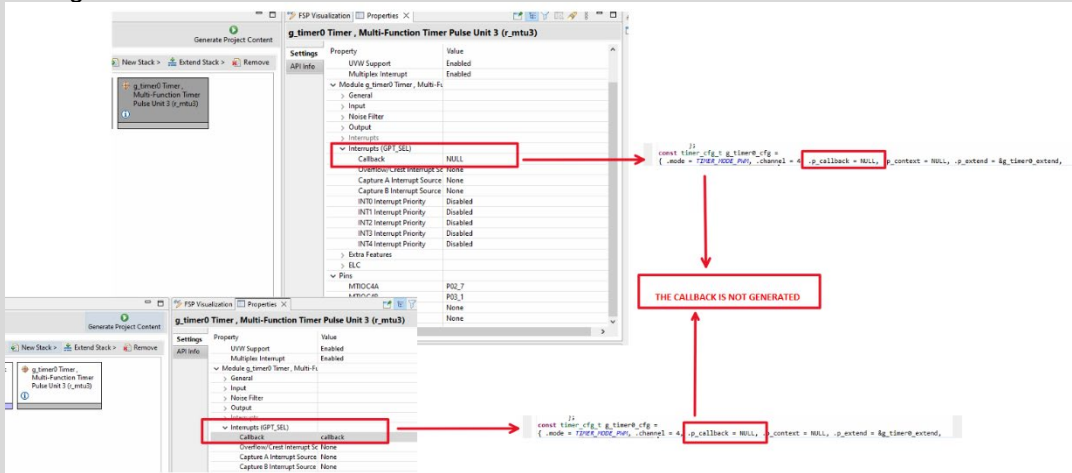
/* Save the context of the current task and select a new task to run. */
~~~~~
}

BSP_ATTRIBUTE_STACKLESS void Exit_IRQ_No_Context_Switch (void)
{
~~~~~
"DSB SY                                     \n"
"ISB SY                                     \n"
#if __FPU_USED
"LDP q0, q1, [sp], #0x20                     \n"
"LDP q2, q3, [sp], #0x20                     \n"
"LDP q4, q5, [sp], #0x20                     \n"
"LDP q6, q7, [sp], #0x20                     \n"
"LDP q8, q9, [sp], #0x20                     \n"
"LDP q10, q11, [sp], #0x20                   \n"
"LDP q12, q13, [sp], #0x20                   \n"
"LDP q14, q15, [sp], #0x20                   \n"
"LDP q16, q17, [sp], #0x20                   \n"
"LDP q18, q19, [sp], #0x20                   \n"
"LDP q20, q21, [sp], #0x20                   \n"

```

```
"LDP q22, q23, [sp], #0x20    \n"  
"LDP q24, q25, [sp], #0x20    \n"  
"LDP q26, q27, [sp], #0x20    \n"  
"LDP q28, q29, [sp], #0x20    \n"  
"LDP q30, q31, [sp], #0x20    \n"  
"LDP x0, x1, [sp], #0x10      \n"  
"MSR FPCR, x0                \n"  
"MSR FPSR, x1                \n"  
#endif  
  
"LDP x0, x1, [sp], #0x10      \n"  
"LDP x2, x3, [sp], #0x10      \n"  
"LDP x4, x5, [sp], #0x10      \n"  
"LDP x6, x7, [sp], #0x10      \n"  
"LDP x8, x9, [sp], #0x10      \n"  
"LDP x10, x11, [sp], #0x10     \n"  
"LDP x12, x13, [sp], #0x10     \n"  
"LDP x14, x15, [sp], #0x10     \n"  
"LDP x16, x17, [sp], #0x10     \n"  
"LDP x18, x19, [sp], #0x10     \n"  
"LDP x20, x21, [sp], #0x10     \n"  
"LDP x22, x23, [sp], #0x10     \n"  
"LDP x24, x25, [sp], #0x10     \n"  
"LDP x26, x27, [sp], #0x10     \n"  
"LDP x28, x29, [sp], #0x10     \n"  
"LDP x30, XZR, [sp], #0x10     \n"  
"ERET                        \n"  
::: "memory");  
}
```

No. 42 Resolved

<p><b>Issue</b></p>	<p><b>MTU3 callback does not occur as expected.</b></p>
<p><b>Target</b></p>	<p><b>RZ/T2H, RZ/N2H</b></p>
<p><b>Category</b></p>	<p><b>FSP Modules, MTU3</b></p>
<p><b>Description</b></p>	<p>Callback does not always occur, just like when you specify NULL in the configurator settings of MTU3.</p> 
<p><b>Workaround</b></p>	<p>Please use the Callback Set API from:  <b>R_MTU3_CallbackSet()</b>          For reference on how to use the API, please refer to the MTU3 Examples below:</p> <p><b>Step 1: Define the Callback Function</b></p> <pre>void mtu3_callback_set_test(timer_callback_args_t * p_args) {     //Callback function to be implemented by the user }</pre> <p><b>Step 2: Sequence call the API</b></p> <pre>/* Open the timer module */ R_MTU3_Open(&amp;g_timer0_ctrl, &amp;g_timer0_cfg);  /* Use R_MTU3_CallbackSet */  R_MTU3_CallbackSet (timer_ctrl_t * const    p_ctrl,                     void ( * p_callback)(timer_callback_args_t *),                     void const * const    p_context,                     timer_callback_args_t * const p_callback_memory)</pre> <p><b>Example:</b></p> <pre>R_MTU3_CallbackSet (     &amp;g_timer0_ctrl,           // Timer control instance     mtu3_callback_set_test, // User define Callback function     (void *)&amp;mtu3_callback_context // User Defined context     (void *)&amp;mtu3_callback_test_args // User define Callback memory ) /* Start timer */ R_MTU3_Start(&amp;g_timer0_ctrl);</pre>

**No. 43 Resolved**

<b>Issue</b>	<b>An undefined error of r_gpt module occurs when building a project.</b>										
<b>Target</b>	<b>RZ/T2H, RZ/N2H</b>										
<b>Category</b>	<b>FSP Modules, GPT</b>										
<b>Description</b>	Undefined error of "gpt_counter_underflow_isr" occurs when "Pin Output Support" is set to anything other than "Enabled with Extra Features" regardless of whether Pin Output is used or not.										
<b>Workaround</b>	<p>Please always set the "Pin Output Support" to "Enabled with Extra Features" when configuring r_gpt module.</p> <p><b>Timer, General PWM (r_gpt)</b></p> <table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Parameter Checking</td> <td>Default (BSP)</td> </tr> <tr> <td>Pin Output Support</td> <td>Enabled with Extra Features</td> </tr> <tr> <td>Write Protect Enable</td> <td>Disabled</td> </tr> <tr> <td>Multiplex Interrupt</td> <td>Disabled</td> </tr> </tbody> </table>	Property	Value	Parameter Checking	Default (BSP)	Pin Output Support	Enabled with Extra Features	Write Protect Enable	Disabled	Multiplex Interrupt	Disabled
Property	Value										
Parameter Checking	Default (BSP)										
Pin Output Support	Enabled with Extra Features										
Write Protect Enable	Disabled										
Multiplex Interrupt	Disabled										

**No. 44**

<b>Issue</b>	<b>Pin names according to unit and channel numbers are not displayed in the r_gpt module configurations.</b>																																																																														
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>																																																																														
<b>Category</b>	<b>FSP Configuration, Stacks</b>																																																																														
<b>Description</b>	<p>When setting the unit and channel number in the r_gpt module, the pin names in the Pins of stack property do not change or do not appear.</p> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid gray; padding: 5px;"> <p><b>g_timer0 Timer, General PWM (r_gpt) RZ/T2M</b></p> <table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr><td>Parameter Checking</td><td>Default (BSP)</td></tr> <tr><td>Pin Output Support</td><td>Enabled</td></tr> <tr><td>Write Protect Enable</td><td>Disabled</td></tr> <tr><td>Multiplex Interrupt</td><td>Disabled</td></tr> <tr><td>Module g_timer0 Timer, General PWM (r_gpt)</td><td></td></tr> <tr><td>  Name</td><td>g_timer0</td></tr> <tr><td>  Unit</td><td>0</td></tr> <tr><td>  Channel</td><td>1</td></tr> <tr><td>  Mode</td><td>Periodic</td></tr> <tr><td>  Period</td><td>0x10000000</td></tr> <tr><td>  Period Unit</td><td>Raw Counts</td></tr> <tr><td>  Output</td><td></td></tr> <tr><td>    Duty Cycle Percent (only applicable in PWM mode)</td><td>50</td></tr> <tr><td>    GPIOCA Output Enabled</td><td>True</td></tr> <tr><td>    GPIOCA Stop Level</td><td>Pin Level Low</td></tr> <tr><td>    GPIOCB Output Enabled</td><td>True</td></tr> <tr><td>    GPIOCB Stop Level</td><td>Pin Level Low</td></tr> <tr><td>  Input</td><td></td></tr> <tr><td>  Interrupts</td><td></td></tr> <tr><td>  Interrupts (GPT_SEL)</td><td></td></tr> <tr><td>  Extra Features</td><td></td></tr> <tr><td>  ELC</td><td></td></tr> <tr><td>  Pins</td><td></td></tr> <tr><td>    GPIOCA</td><td>&lt;unavailable&gt;</td></tr> <tr><td>    GPIOCB</td><td>&lt;unavailable&gt;</td></tr> </tbody> </table> <p>No change pin name (expected name: GTIOC1A, GTIOC1B)</p> </div> <div style="border: 1px solid gray; padding: 5px;"> <p><b>g_timer0 Timer, General PWM (r_gpt) RZ/T2H</b></p> <table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr><td>Parameter Checking</td><td>Default (BSP)</td></tr> <tr><td>Pin Output Support</td><td>Disabled</td></tr> <tr><td>Write Protect Enable</td><td>Disabled</td></tr> <tr><td>Multiplex Interrupt</td><td>Disabled</td></tr> <tr><td>Module g_timer0 Timer, General PWM (r_gpt)</td><td></td></tr> <tr><td>  General</td><td></td></tr> <tr><td>  Output</td><td></td></tr> <tr><td>  Input</td><td></td></tr> <tr><td>  Interrupts</td><td></td></tr> <tr><td>  Interrupts (GPT_SEL)</td><td></td></tr> <tr><td>  Extra Features</td><td></td></tr> <tr><td>  ELC</td><td></td></tr> </tbody> </table> <p>No pin configuration items.</p> </div> </div>	Property	Value	Parameter Checking	Default (BSP)	Pin Output Support	Enabled	Write Protect Enable	Disabled	Multiplex Interrupt	Disabled	Module g_timer0 Timer, General PWM (r_gpt)		Name	g_timer0	Unit	0	Channel	1	Mode	Periodic	Period	0x10000000	Period Unit	Raw Counts	Output		Duty Cycle Percent (only applicable in PWM mode)	50	GPIOCA Output Enabled	True	GPIOCA Stop Level	Pin Level Low	GPIOCB Output Enabled	True	GPIOCB Stop Level	Pin Level Low	Input		Interrupts		Interrupts (GPT_SEL)		Extra Features		ELC		Pins		GPIOCA	<unavailable>	GPIOCB	<unavailable>	Property	Value	Parameter Checking	Default (BSP)	Pin Output Support	Disabled	Write Protect Enable	Disabled	Multiplex Interrupt	Disabled	Module g_timer0 Timer, General PWM (r_gpt)		General		Output		Input		Interrupts		Interrupts (GPT_SEL)		Extra Features		ELC	
Property	Value																																																																														
Parameter Checking	Default (BSP)																																																																														
Pin Output Support	Enabled																																																																														
Write Protect Enable	Disabled																																																																														
Multiplex Interrupt	Disabled																																																																														
Module g_timer0 Timer, General PWM (r_gpt)																																																																															
Name	g_timer0																																																																														
Unit	0																																																																														
Channel	1																																																																														
Mode	Periodic																																																																														
Period	0x10000000																																																																														
Period Unit	Raw Counts																																																																														
Output																																																																															
Duty Cycle Percent (only applicable in PWM mode)	50																																																																														
GPIOCA Output Enabled	True																																																																														
GPIOCA Stop Level	Pin Level Low																																																																														
GPIOCB Output Enabled	True																																																																														
GPIOCB Stop Level	Pin Level Low																																																																														
Input																																																																															
Interrupts																																																																															
Interrupts (GPT_SEL)																																																																															
Extra Features																																																																															
ELC																																																																															
Pins																																																																															
GPIOCA	<unavailable>																																																																														
GPIOCB	<unavailable>																																																																														
Property	Value																																																																														
Parameter Checking	Default (BSP)																																																																														
Pin Output Support	Disabled																																																																														
Write Protect Enable	Disabled																																																																														
Multiplex Interrupt	Disabled																																																																														
Module g_timer0 Timer, General PWM (r_gpt)																																																																															
General																																																																															
Output																																																																															
Input																																																																															
Interrupts																																																																															
Interrupts (GPT_SEL)																																																																															
Extra Features																																																																															
ELC																																																																															

<b>Workaround</b>	<p>Please use the Pins tab in the FSP Configuration to configure pins for the r_gpt module.</p> <p><b>Pin Configuration</b></p> <p>Select Pin Configuration: RZT2H Evaluation Board   Manage configurations...   Export to CSV file   Configure Pin Driver Warnings   Generate data: g_bsp_pin_cfg</p> <p>Pin Selection: GPT (selected)</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> <th>Lock</th> <th>Link</th> </tr> </thead> <tbody> <tr><td>Pin Group Selection</td><td>Mixed</td><td></td><td></td></tr> <tr><td>Operation Mode</td><td>Disabled</td><td></td><td></td></tr> <tr><td>Input/Output</td><td></td><td></td><td></td></tr> <tr><td>GTIADC00_0</td><td>None</td><td></td><td></td></tr> <tr><td>GTIADC00_1</td><td>None</td><td></td><td></td></tr> <tr><td>GTIADC00_0A</td><td>None</td><td></td><td></td></tr> <tr><td>GTIADC00_0B</td><td>None</td><td></td><td></td></tr> <tr><td>GTIADC00_1A</td><td>None</td><td></td><td></td></tr> <tr><td>GTIADC00_1B</td><td>None</td><td></td><td></td></tr> </tbody> </table> <p>Module name: GPT00</p> <p>Pin Function   Pin Number   Summary   BSP   Clock   Pins   Interrupts   Event Links   Stacks   Components</p>	Name	Value	Lock	Link	Pin Group Selection	Mixed			Operation Mode	Disabled			Input/Output				GTIADC00_0	None			GTIADC00_1	None			GTIADC00_0A	None			GTIADC00_0B	None			GTIADC00_1A	None			GTIADC00_1B	None		
Name	Value	Lock	Link																																						
Pin Group Selection	Mixed																																								
Operation Mode	Disabled																																								
Input/Output																																									
GTIADC00_0	None																																								
GTIADC00_1	None																																								
GTIADC00_0A	None																																								
GTIADC00_0B	None																																								
GTIADC00_1A	None																																								
GTIADC00_1B	None																																								

**No. 45 Resolved**

<b>Issue</b>	Using R_GPT_DutyCycleSet() with the option, both pins A and B cannot work properly.
<b>Target</b>	RZ/T2H, RZ/T2M, RZ/T2ME, RZ/T2L
<b>Category</b>	FSP Modules, GPT
<b>Description</b>	When updating the duty cycle, GPT_IO_PIN_GTIOCA_AND_GTIOCB cannot be used to set both pins at the same time.
<b>Workaround</b>	To change the duty cycle during runtime, each pin must be updated separately. Example: R_GPT_DutyCycleSet(&g_timer0_ctrl, duty_cycle_counts, GPT_IO_PIN_GTIOCA); R_GPT_DutyCycleSet(&g_timer0_ctrl, duty_cycle_counts, GPT_IO_PIN_GTIOCB);

**No. 46 Resolved**

<b>Issue</b>	Parameter checking of R_ETHER_SELECTOR_Open() is not working properly.
<b>Target</b>	RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H
<b>Category</b>	FSP Modules, ETHER_SELECTOR
<b>Description</b>	When using the R_ETHER_SELECTOR_Open() with parameter checking enabled, the configuration pointer is used before being assigned. This causes parameter checking to work incorrectly, sometimes leading to exceptions.
<b>Workaround</b>	Disable parameter checking when using the Ether Selector module.

**No. 47 Resolved**

<b>Issue</b>	Parameter checking feature of R_GMAC_CallbackSet() is not working.
<b>Target</b>	RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H
<b>Category</b>	FSP Modules, ETHER_GMAC
<b>Description</b>	Parameter check feature of R_GMAC_CallbackSet() does not work because the parameter checking macro name generated by the MDF is inconsistent with the macro referenced in the actual source code. <ul style="list-style-type: none"> <li>Macro generated by MDF: GMAC_CFG_PARAM_CHECKING_ENABLE</li> <li>Macro used in source code: ETHER_CFG_PARAM_CHECKING_ENABLE</li> </ul> Note: Parameter check feature of other APIs works properly.
<b>Workaround</b>	Step 1: In project Properties->Builders, uncheck the DDSC Builder option (this step allows the user to modify the "r_ether_cfg.h" file)  Step 2: In "XXX_cfg/fsp_cfg/r_ether_cfg.h"(XXX = rzt, rzn), manually add the macro definition below. #define ETHER_CFG_PARAM_CHECKING_ENABLE ((1))

No. 48 Resolved

<b>Issue</b>	<b>r_usb_hhid module is not working properly.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H</b>
<b>Category</b>	<b>FSP Modules, USB_HHID</b>
<b>Description</b>	<p>It is possible that HOST does not respond to the received data because it cannot get the transfer size correctly, so even if the transfer is completed without error, it may not be successful.</p> <p>In the report, when debugging with a USB analyzer, errors are reported as below:</p>
<b>Workaround</b>	<p>Please modify “rz\fsp\src\r_usb_basic\r_usb_basic.c” line 1877 as below.</p> <p>Before update:</p> <pre>(*p_pipe) = (uint16_t)((*p_pipe)   (uint16_t) 1 &lt;&lt; (pipe_no - 1));/* Start timer */</pre> <p>After update:</p> <pre>if (((uint16_t) p_instance_ctrl-&gt;device_address) &lt;&lt; USB_DEVADDRBIT) ==     ((uint16_t) (g_usb_pipe_table[p_instance_ctrl-&gt;module_number][pipe_no].pipe_maxp &amp; USB_DEVSEL)) {     (*p_pipe) = (uint16_t)((*p_pipe)   (uint16_t) 1 &lt;&lt; pipe_no); }</pre>

No. 49 Resolved

<b>Issue</b>	<b>The secondary and later projects may not work properly when multicore with flash boot mode.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>FSP Modules, BSP, FreeRTOS</b>
<b>Description</b>	<p>Because the secondary and later cores are released from reset state immediately after the primary core copies the program from external flash memory to internal RAM, the secondary and later cores' processing is carried out before the startup processing that should be carried out by the primary core is completed.</p> <p>e.g., In multicore projects using four or more cores, the registers that control register write protection may be set on all cores, making it impossible to write to protected registers.</p> <p>e.g., In FreeRTOS Blinky projects using multicore, the RTOS is run on the secondary and later cores before the global system counter is initialized on the primary core, causing it to not function properly.</p>
<b>Workaround</b>	<p>In multiprocessing, insert the loop part in described in startup_core.c of the secondary and later projects with reference to Appendix.</p> <p>How to Debug the FSP Project with Flash Boot Mode No. 1.</p>

## No. 50

<b>Issue</b>	<b>*length_bytes is an input/output parameter of R_GMAC_Read()/R_GMAC_B_Read() and must be reset before every call</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2L, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>FSP Modules, GMAC, GMAC_B</b>
<b>Description</b>	When Read Buffer Size Checking is enabled in the r_gmac or r_dmac_b module configuration, the R_GMAC_Read()/R_GMAC_B_Read() uses *length_bytes as an input parameter, representing the size of the *p_buffer. The R_GMAC_Read()/R_GMAC_B_Read() also uses *length_bytes as an output parameter and overwrites it with the actual received length on return. If the caller does not reset the "length_bytes" value to the buffer capacity before each call, the buffer check may fail, and the API can return FSP_ERR_INVALID_SIZE.
<b>Workaround</b>	If Read Buffer Size Checking is enabled in r_gmac or r_dmac_b module configuration, please set *length_bytes to the capacity of p_buffer (in bytes) before each R_GMAC_Read()/R_GMAC_B_Read() call. Example: <pre>fsp_err_t err; uint8_t rx_buffer[1524]; uint32_t len = sizeof(rx_buffer); err = R_GMAC_Read(p_ctrl, rx_buffer, &amp;len); ... len = sizeof(rx_buf); // Must be set again fsp_err_t err = R_GMAC_Read(p_ctrl, rx_buffer, &amp;len);</pre>

## No. 51

<b>Issue</b>	<b>When one unit is connected to multiple CS spaces, manual commands are applied only to the most recently opened CS space.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2L, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>FSP Modules, XSPI_HYPER, XSPI_OSPI, XSPI_QSPI</b>
<b>Description</b>	When connecting multiple CS spaces within the same unit, the target of manual command issuance is fixed to the CS space that was opened last. As a result, manual commands cannot be issued to other CS spaces.
<b>Workaround</b>	Please implement processing to change the CDCTL0.CSSEL bit before calling the DirectTransfer API as below: <pre>/* Change the CDCTL0.CSSEL bit to 0 for CS0 or 1 for CS1 (The following is an example for case unit 0 and CS0 space) */ R_XSPI0-&gt;CDCTL0_b.CSSEL = 0;  /* Call the API (The following is an example for case r_xspi_ospi module) */ R_XSPI_OSPI_DirectTransfer();</pre>

## No. 52

<b>Issue</b>	Limit the requested transfer size to 2048 bytes or less when using the DMA transfer mode of a USB peripheral.
<b>Target</b>	RZ/T2M, RZ/T2ME, RZ/T2L, RZ/T2H, RZ/N2L, RZ/N2H
<b>Category</b>	FSP Modules, USB PCDC, USB PHID, USB PMSC, USB PVND, USB (FSP Configuration)
<b>Description</b>	If the requested transfer size exceeds 2048 bytes in DMA transfer mode, 1 to 3 bytes of data transmitted from the USB peripheral may be lost.
<b>Workaround</b>	Please use DMA transfers only for requested transfer sizes of 2048 bytes or less. For larger transfers, use CPU transfer.

## Appendix

**7.2 Tool Software Limitations**

This section describes the limitations regarding the tool software (e<sup>2</sup> studio, FSP SC) to create and debug FSP projects.

**Table 20. List of Tool Software Limitations**

No	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
1	When installing, please install into the default installation folder specified by the installer.	✓	✓	✓		✓		SC, FSP SC
2	Before pressing the reset button on the board, disconnect the e <sup>2</sup> studio connection first.	✓	✓			✓		e <sup>2</sup> studio
3	An error has occurred because the program download to the NOR flash area has failed. The download is successful on the second connection.	✓				✓		e <sup>2</sup> studio
4	The user program cannot be stopped immediately after the device boot process.	✓	✓	✓	✓	✓	✓	e <sup>2</sup> studio
5	When using the e <sup>2</sup> studio installer, if checking multiple check boxes such as "View Release Notes" and so on to show information on the browser, the ONLY head item of checked items is shown.	✓	✓			✓		e <sup>2</sup> studio
6	The Memory Region Usage of ATCM displayed in the Memory Usage window of e <sup>2</sup> studio is smaller than the actual size by the memory region usage of DUMMY.	✓	✓			✓		e <sup>2</sup> studio
7	When debugging RAM execution without a flash memory project with a program written to flash memory, erase the flash memory before debugging.	✓	✓	✓	✓	✓	✓	e <sup>2</sup> studio
8	Applying RZ/T2 FSP v.1.2.0 pack to a project that is already working with RZ/T2M FSP v.1.1.0 causes an error when connecting the debugger.	✓						IAR EWARM
9	The Device Memory Usage of CPU1 in the Memory Usage window does not work properly.	✓		✓	✓		✓	e <sup>2</sup> studio
10	When adding the CallbackSet function using the Developer Assistance feature, the second argument needs to be changed.	✓	✓	✓	✓	✓	✓	e <sup>2</sup> studio, SC
11	In IAR EWARM 9.60.1, an error occurs when starting to debug multiprocessing projects of RAM execution without flash memory.	✓		✓				IAR EWARM
12	Build failed when executed with a different install path.	✓	✓	✓	✓	✓	✓	FSP SC
13	Unable to debug the CA55 flash boot project with e <sup>2</sup> studio.				✓		✓	e <sup>2</sup> studio

No	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
14	Unable to restart debugging immediately after debugging ends of CA55 RAM execution without a flash memory project in e <sup>2</sup> studio.				✓		✓	e <sup>2</sup> studio
15	Unable to re-download the CA55 binary file.				✓		✓	IAR EWARM
16	Unable to access the upper 32-bit address area in memory view.				✓		✓	e <sup>2</sup> studio
17	Unable to create a CMake project using FSP SC.	✓	✓	✓	✓	✓	✓	FSP SC
18	The secondary project aborts when debugging multicore with flash boot mode.	✓		✓	✓		✓	IAR EWARM
19	Build errors occur in CA55 projects when installing e <sup>2</sup> studio as the Current user.				✓		✓	e <sup>2</sup> studio
20	Wrong core name when creating a project CA55 with FSP SC.						✓	FSP SC
21	Build is failed when adding the OpenAMP.				✓		✓	FSP SC
22	The bundle file (.sbd) may not be generated during build.	✓	✓	✓	✓	✓	✓	e <sup>2</sup> studio
23	When implementing CMT interrupts in the RZ/T2H CR52 or RZ/N2H CR52 project, an unintended source file is displayed during debugging.				✓		✓	e <sup>2</sup> studio
24	The assert function does not work properly.	✓	✓	✓	✓	✓	✓	e2 studio
25	In multicore debugging, changes to the pin settings in the primary project from the default are not reflected.	✓		✓	✓		✓	FSP SC
26	After building the project, the debug configuration file is not automatically generated.	✓		✓	✓		✓	e <sup>2</sup> studio
27	When using the RZ/T2H CR52 or RZ/N2H CR52 core as the secondary project, an unintended source file is displayed during multicore debugging.				✓		✓	e <sup>2</sup> studio
28	When multiple packs are imported, a message appears during debugging indicating that an unknown device is selected.	✓	✓	✓	✓	✓	✓	e <sup>2</sup> studio

**No. 1 Resolved**

<b>Limitation</b>	<b>When installing, please install into the default installation folder specified by the installer.</b>
<b>Target Device</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/N2L</b>
<b>Category</b>	<b>SC, FSP SC</b>

<b>Description</b>	When sharing a project between different PCs, build errors will occur if the installation folders are different.
--------------------	--

**No. 2 Resolved**

<b>Limitation</b>	<b>Before pressing the reset button on the board, disconnect the e<sup>2</sup> studio connection first.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/N2L</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	If the reset button is pressed on the board while connected with e <sup>2</sup> studio, debugging will not be able to continue.

**No. 3 Resolved**

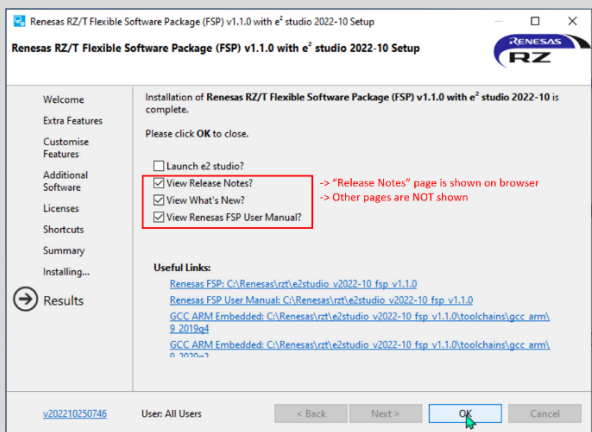
<b>Limitation</b>	<b>An error has occurred because the program download to the NOR flash area has failed. The download is successful on the second connection.</b>
<b>Target</b>	<b>RZ/T2M, RZ/N2L</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	If the following error is displayed when connecting the debugger or downloading the program, click the [OK] button to close the dialog and try connecting again. <div data-bbox="600 949 1177 1256" data-label="Image"> </div>

**No. 4**

<b>Limitation</b>	<b>The user program cannot be stopped immediately after the device boot process.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	Immediately after the device boot process (boot code), the program cannot be stopped at the beginning of the user program (loader program). When debugging, please follow the guide in Appendix. How to Debug the FSP Project with Flash Boot Mode.

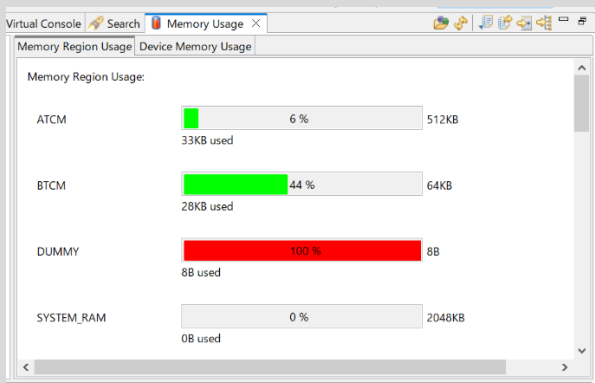
**No. 5 Invalid**

<b>Limitation</b>	When using the e <sup>2</sup> studio installer, if checking multiple check boxes, such as “View Release Notes” and so on, to show information on the browser, the ONLY head item of checked items is shown.
<b>Target</b>	RZ/T2M, RZ/T2L, RZ/N2L
<b>Category</b>	e <sup>2</sup> studio
<b>Description</b>	For example, if checking the “View Release Notes” check box and other check boxes on the following window, the ONLY “Release Notes” is shown, and the other contents are NOT shown.



**No. 6 Resolved**

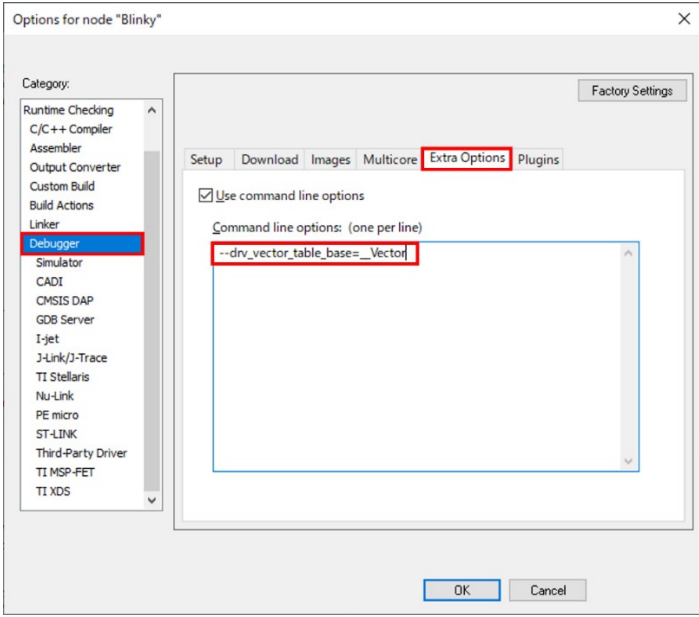
<b>Limitation</b>	The Memory Region Usage of ATCM displayed in the Memory Usage window of e <sup>2</sup> studio is smaller than the actual size by the Memory Region Usage of DUMMY.
<b>Target</b>	RZ/T2M, RZ/T2L, RZ/N2L
<b>Category</b>	e <sup>2</sup> studio
<b>Description</b>	<p>The Memory Region Usage of DUMMY shown in the Memory Usage window is the region used by the system. The DUMMY is placed in ATCM, the Memory Region Usage of ATCM does NOT include its size.</p> <p>Therefore, please note that the Memory Region Usage of ATCM displayed is smaller than the actual size by the Memory Region Usage of DUMMY.</p>



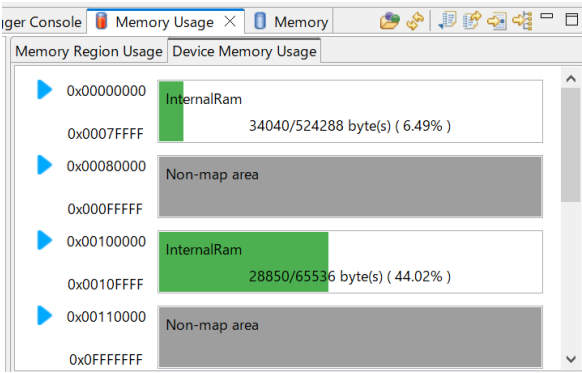
## No. 7

<b>Limitation</b>	<b>When debugging RAM execution without a flash memory project, erase the flash memory before debugging.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	If you run a RAM-only execution without flash memory and the program is written in flash memory, it may be impossible to debug the project. When erasing flash memory, please follow the guide.  Appendix How to Erase Flash Memory

## No. 8

<b>Limitation</b>	<b>Applying the RZ/T2 FSP v.1.2.0 pack to a project already using RZ/T2M FSP v.1.1.0 causes an error when connecting the debugger.</b>
<b>Target</b>	<b>RZ/T2M</b>
<b>Category</b>	<b>IAR EWARM</b>
<b>Description</b>	An error occurs when connecting to the debugger because the function name of the vector table was changed in RZ/T2 FSP v.1.2.0. Change the following command in the “ <b>command line options (one per line)</b> ” to <ul style="list-style-type: none"> <li>• (Before change) <code>--drv_vector_table_base=vector_table</code></li> <li>• (After change) <code>--drv_vector_table_base=__Vector</code></li> </ul> 

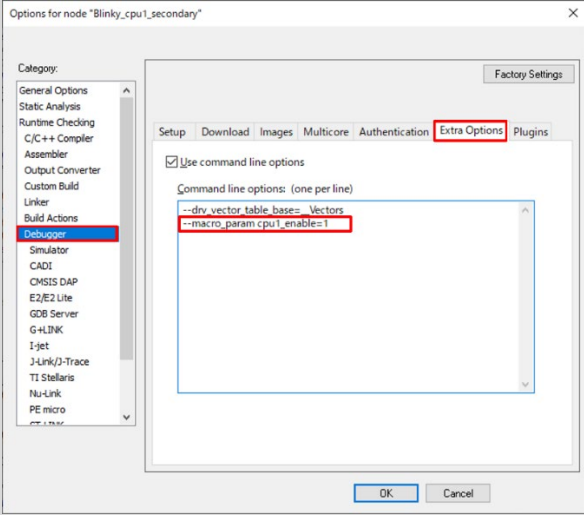
## No. 9

<b>Limitation</b>	<b>The Device Memory Usage of CPU1 in the Memory Usage window does not work properly.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	<p>The Device Memory Usage in the Memory Usage window cannot distinguish between CPU0 and CPU1.</p> <p>When debugging CPU1, the memory area available for CPU1 should be displayed, but the memory area available for CPU0 is incorrectly displayed.</p> 

## No. 10

<b>Limitation</b>	<b>When adding the CallbackSet function using the Developer Assistance feature, the second argument needs to be changed.</b>
<b>Target</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>e<sup>2</sup> studio, SC</b>
<b>Description</b>	<p>When adding the R_xxx_CallbackSet() function (xxx means any module name) using the Developer Assistance feature, the second argument does not have the correct value. Please replace the second argument with "p_callback".</p> <p>An example of the SCI_SPI module, adding CallbackSet() using the Developer Assistance, results in the following.</p> <pre>status = R_SCI_SPI_CallbackSet(&amp;g_spi0_ctrl, spi_callback_args_t, p_context, p_callback_memory);</pre> <p>It needs to replace the second argument with "p_callback".</p> <pre>status = R_SCI_SPI_CallbackSet(&amp;g_spi0_ctrl, p_callback, p_context, p_callback_memory);</pre>

## No. 11 Moved to 5.3.3.2 Build for Multiprocessing No. 2-iv

<b>Limitation</b>	In IAR EWARM 9.60.1, an error occurs when starting to debug multiprocessing projects of RAM execution without flash memory.
<b>Target</b>	RZ/T2M, RZ/T2ME
<b>Category</b>	IAR EWARM
<b>Description</b>	<p>When IAR EWARM 9.60.1 is used, there is no defined value required for debugging the CPU1 project, and an error occurs when debugging begins.</p> <p>In EWARM 9.60.1, when debugging projects of multiprocessing, it is necessary to add "--macro_param cpu1_enable=1" in the "command line options (one per line)" of the CPU1 project.</p> 

## No. 12

<b>Limitation</b>	Build failed when executed with a different install path
<b>Target</b>	RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H
<b>Category</b>	FSP SC
<b>Description</b>	<p>If the installation path of FSP SC is different between creating the project and executing the project, the build of the project fails.</p> <p>Please reselect the execution path by following the steps below.</p> <ol style="list-style-type: none"> <li>1. Launch FSP SC.</li> <li>2. Close the window to create a new project.</li> <li>3. Click on File -&gt; Open and select configuration.xml in your project.</li> <li>4. Click "Generate Project Content".</li> <li>5. Save the project and close FSP SC.</li> <li>6. Open the project with EWARM.</li> </ol>

**No. 13 Resolved**

<b>Limitation</b>	<b>Unable to debug the CA55 flash boot project with e<sup>2</sup> studio.</b>
<b>Target Device</b>	<b>RZ/T2H(CA55) , RZ/N2H(CA55)</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	When debugging with e <sup>2</sup> studio, the CA55 Core0 system reset is not performed. Therefore, the program in the external flash is not copied to the internal RAM, and it cannot be operated correctly. Please check the operation with the RAM execution without the flash memory project.

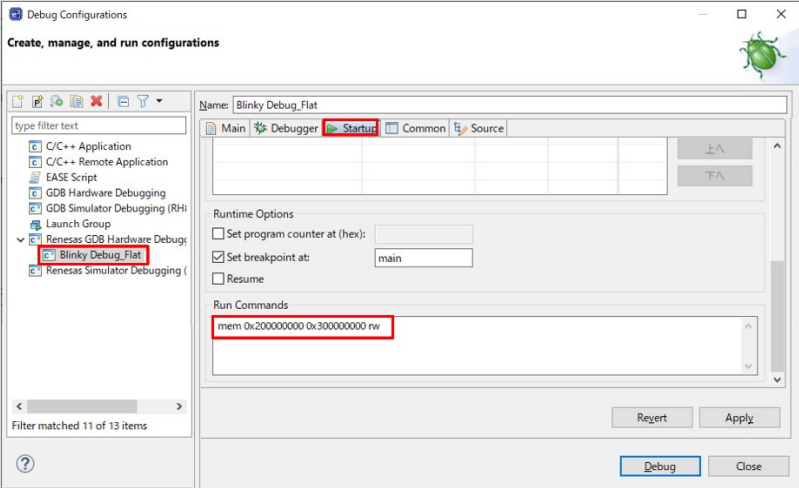
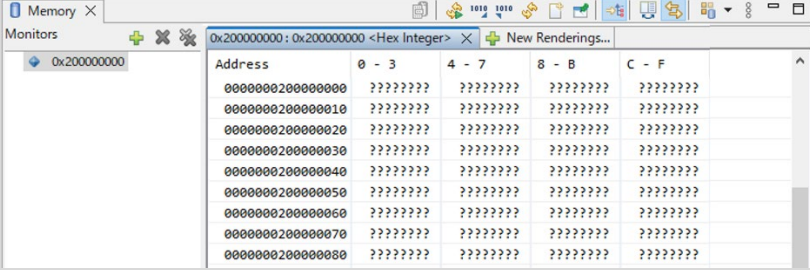
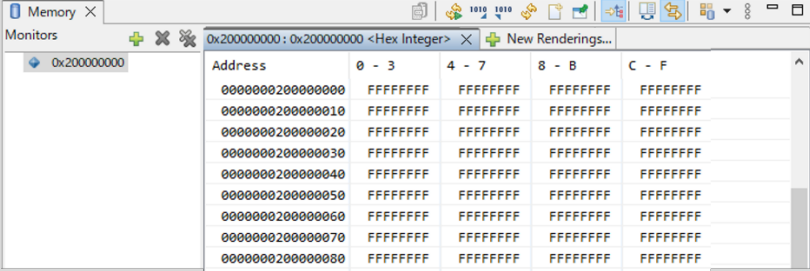
**No. 14 Resolved**

<b>Limitation</b>	<b>Unable to restart debugging immediately after debugging ends of CA55 RAM execution without a flash memory project in e<sup>2</sup> studio.</b>
<b>Target Device</b>	<b>RZ/T2H(CA55), RZ/N2H(CA55)</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	When debugging with e <sup>2</sup> studio, the CA55 Core0 system reset is not performed. Therefore, after debugging is complete, if you want to run the debug again, press the reset button (red) on the board.

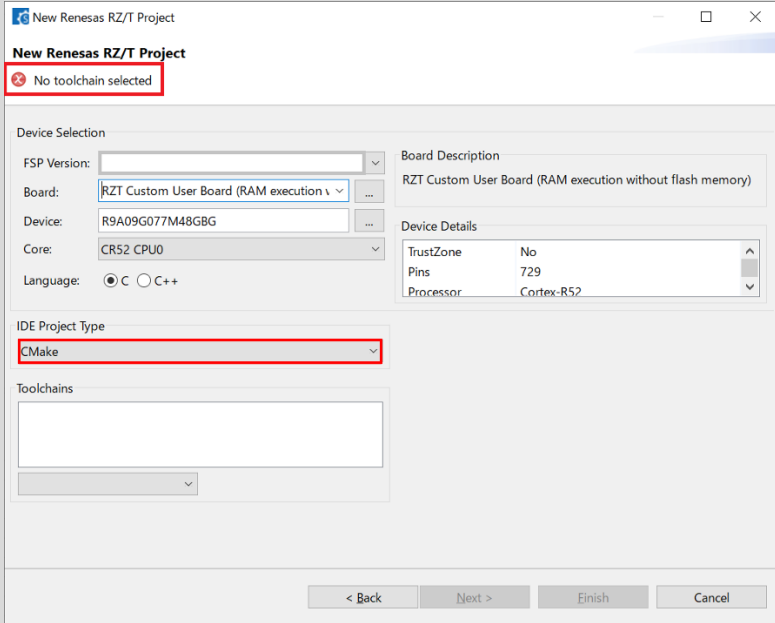
**No. 15**

<b>Limitation</b>	<b>Unable to re-download the CA55 binary file.</b>
<b>Target Device</b>	<b>RZ/T2H(CA55), RZ/N2H(CA55)</b>
<b>Category</b>	<b>IAR EWARM</b>
<b>Description</b>	If you download the CA55 binary file, then download it again, the process never finishes. To avoid this issue, you need to erase the flash from the CR52 flash boot project.

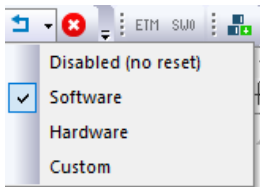
No. 16 Resolved

<b>Limitation</b>	<b>Unable to access the upper 32-bit address area in memory view.</b>
<b>Target Device</b>	<b>RZ/T2H(CA55), RZ/N2H(CA55)</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	<p>In the Memory view of e<sup>2</sup> studio, data in the upper 32-bit address area cannot be displayed.</p> <p>When debugging the CA55 project, please make the following settings when accessing the upper 32-bit address.</p> <ol style="list-style-type: none"> <li>1. Open <b>Debug Configurations</b> of the CA55 project.</li> <li>2. Select <b>Renesas GDB Hardware Debugging</b> &gt; [project name] Debug_Flat.</li> <li>3. Select <b>Startup</b> and specify the command in <b>Run Commands</b>. Referring to the following command, specify the address area you want to access, and you will be able to access the upper 32-bit address area.  "mem 0x200000000 0x300000000 rw"</li> </ol>  <p><b>Before adding the command</b></p>  <p><b>After adding the command</b></p> 

## No. 17 Resolved

<b>Limitation</b>	<b>Unable to create a CMake project using FSP SC.</b>
<b>Target Device</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>FSP SC</b>
<b>Description</b>	<p>Even if you specify CMake as the IDE Project Type when creating a project, an error message "No toolchain selected" is displayed, and you cannot proceed to the next screen for project creation.</p>  <p>There is no workaround.</p>


## No. 18

<b>Limitation</b>	<b>The secondary project aborts when debugging multicore with flash boot mode.</b>
<b>Target Device</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>IAR EWARM</b>
<b>Description</b>	<p>When performing multicore debugging of a flash boot project on IAR EWARM, after copying an application program from the primary core memory to the secondary one, executing the secondary project will abort.</p> <p>For multicore debugging with flash boot mode in IAR EWARM, follow the steps below:</p> <ol style="list-style-type: none"> <li>1. Run the primary project up to main().</li> <li>2. Software reset the secondary project. Click on the downward triangle to the right of the reset icon and select <b>Software</b>.</li> </ol>  <ol style="list-style-type: none"> <li>3. Run the secondary project.</li> </ol>

## No. 19

<b>Limitation</b>	<b>Build errors occur in CA55 projects when installing e<sup>2</sup> studio as the Current user.</b>
<b>Target Device</b>	<b>RZ/T2H(CA55), RZ/N2H(CA55)</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	If you install e <sup>2</sup> studio as the Current user, an error will occur when building a CA55 project. As a workaround, install e <sup>2</sup> studio as All Users.

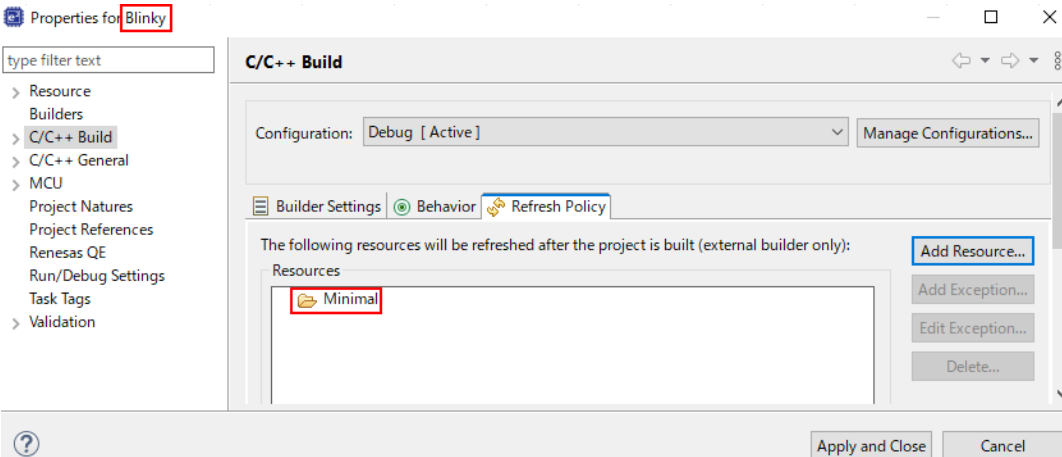
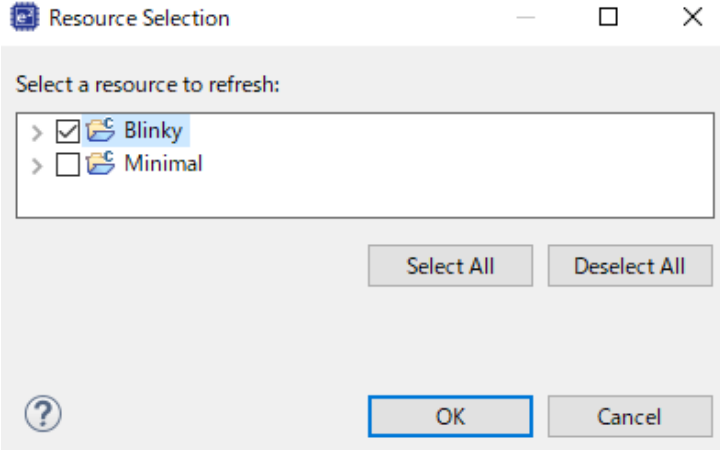
## No. 20 Resolved

<b>Limitation</b>	<b>Wrong core name when creating a project CA55 with FSP SC.</b>
<b>Target Device</b>	<b>RZ/N2H(CA55)</b>
<b>Category</b>	<b>FSP SC</b>
<b>Description</b>	When you create a project CA55 with FSP SC the device name is shown as "CR52_0" in buildinfo.ipcf even though you select CA55 core. 

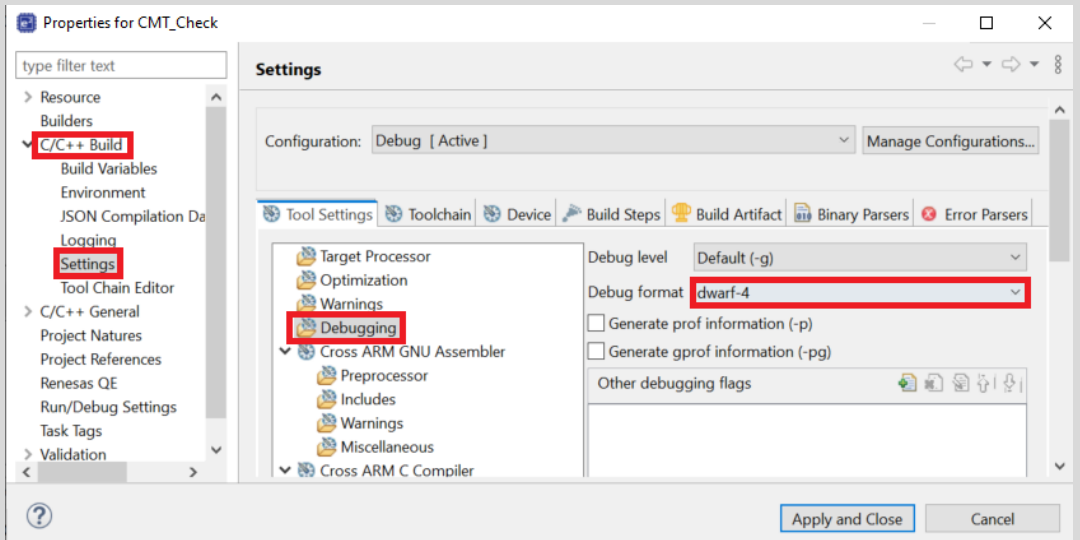
## No. 21

<b>Limitation</b>	<b>Build is failed when adding the OpenAMP.</b>
<b>Target Device</b>	<b>RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>FSP SC</b>
<b>Description</b>	When adding OpenAMP in the Stacks tab, the source browser throws an error, and the build fails due to a conflict of the same file name. After clicking the Generate Project Content button in the FSP SC, please move the files listed below from the "Component" group to another group in [project name]/buildinfo.ipcf. <pre>&lt;path&gt;rzt/linaro/libmetal/lib/device.c&lt;/path&gt; &lt;path&gt;rzt/linaro/libmetal/lib/dma.c&lt;/path&gt; &lt;path&gt;rzt/linaro/libmetal/lib/init.c&lt;/path&gt; &lt;path&gt;rzt/linaro/libmetal/lib/io.c&lt;/path&gt; &lt;path&gt;rzt/linaro/libmetal/lib/log.c&lt;/path&gt; &lt;path&gt;rzt/linaro/libmetal/lib/shmem.c&lt;/path&gt;</pre>

No. 22

<b>Limitation</b>	<b>The bundle file (.sbd) may not be generated during build.</b>
<b>Target Device</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	<p>In e<sup>2</sup> studio, when you import and build an existing project, the bundle file (.sbd) may not be generated.</p> <p>As a workaround, right-click the project in Project Explorer, click <b>Properties &gt; C/C++ Build &gt; Refresh Policy</b>, and verify that the projects displayed in Resources match the projects listed in the upper left.</p> <p>If they are different, click the project in Resources and click <b>Delete...</b></p>  <p>Next, click <b>Add Resource...</b>, check the correct project, and click <b>OK</b>.</p>  <p>Finally, click <b>Apply and Close</b>. By making this setting, an .sbd file will be generated during the build.</p>

## No. 23 Resolved

<b>Limitation</b>	<b>When implementing CMT interrupts in the RZ/T2H CR52 or RZ/N2H CR52 project, an unintended source file is displayed during debugging.</b>
<b>Target Device</b>	<b>RZ/T2H (CR52), RZ/N2H (CR52)</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	<p>When you create a Blinky project for an RZ/T2H CR52 or an RZ/N2H CR52 project, add a CMT stack, and define a CMT callback function, the command to jump to hal_entry() during debugging may not display the hal_entry.c source file, and an unintended file may be displayed.</p> <p>As a workaround, right-click the project in Project Explorer, click <b>Properties &gt; C/C++ Build &gt; Settings &gt; Tool Settings &gt; Debugging &gt; Default format</b>, and change "Toolchain default" to "dwarf-4".</p> 

## No. 24 Resolved

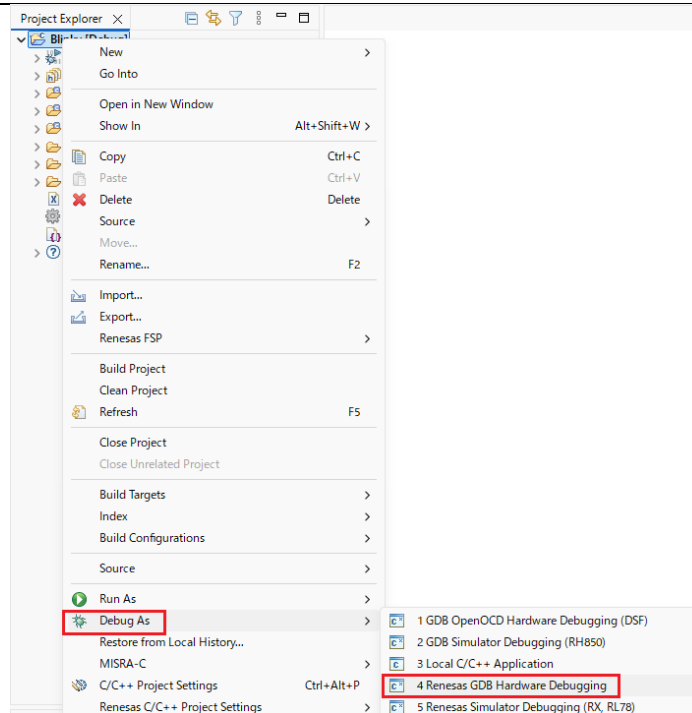
<b>Limitation</b>	<b>The assert function does not work properly.</b>
<b>Target Device</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H(CR52), RZ/N2L, RZ/N2H(CR52)</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	<p>Executing the assert function causes a transition to ARM mode. Since most FSP instructions are Thumb instructions, the following behavior occurs:</p> <ol style="list-style-type: none"> <li>1. The instruction is fetched.</li> <li>2. Prefetch_Handler is called.</li> <li>3. By default, Prefetch_Handler has no implementation, so the program resumes without stopping, returning to the address of the original instruction plus 4.</li> <li>4. It returns to step 1.</li> </ol> <p>This loop repeats infinitely. Caution is required because any ARM instructions present within the loop will be executed.</p> <p>As a countermeasure, implement the following function as a user function: </p> <pre>void __assert_func (const char * file, int line, const char * func, const char * expr) {     FSP_PARAMETER_NOT_USED(file);     FSP_PARAMETER_NOT_USED(line);     FSP_PARAMETER_NOT_USED(func);     FSP_PARAMETER_NOT_USED(expr);     while (1)     {         /* Do nothing. */     } }</pre>

## No. 25

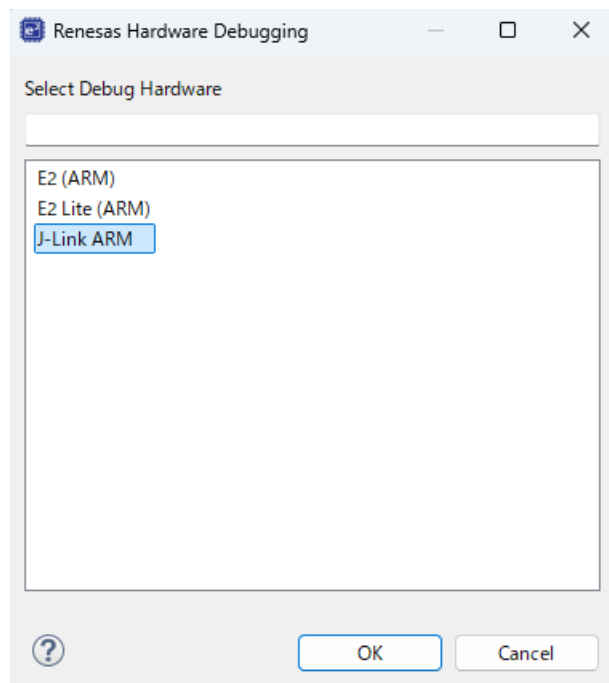
<b>Limitation</b>	<b>In multicore debugging, changes to the pin settings in the primary project from the default are not reflected.</b>
<b>Target Device</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>SC, FSP SC</b>
<b>Description</b>	Pin settings configured in the primary project are not propagated to the secondary and subsequent projects. As a result, even if you change the pin settings in the primary project, the secondary and later projects overwrite them with the default pin settings, so the changes are not reflected. If you want to change the pin settings from the default, please update the pin settings in all projects.

## No. 26

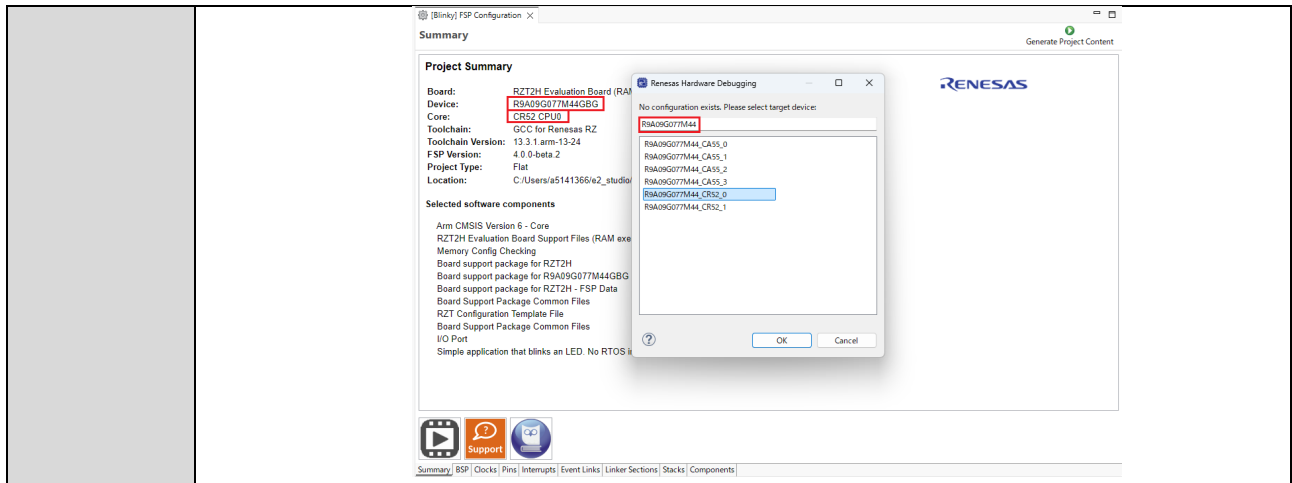
<b>Limitation</b>	<b>After building the project, the debug configuration file is not automatically generated.</b>
<b>Target Device</b>	<b>RZ/T2M, RZ/T2ME, RZ/T2H, RZ/N2H</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	<p>Even if you create a project and the build completes successfully, the debug configuration file is not generated automatically. Therefore, the project's debug configuration does not appear under Run &gt; Debug Configurations... &gt; GDB Hardware Debugging.</p> <p>Please generate the file manually using the following steps.</p> <ol style="list-style-type: none"> <li>1. Right-click on the project, select <b>Debug As</b>, and select <b>4 Renesas GDB Hardware Debugging</b>.</li> </ol>



2. Select **J-Link ARM** and click **OK**.



3. Enter the Device listed in the FSP Configuration into the search field, select the one matching the Core, and click **OK**.



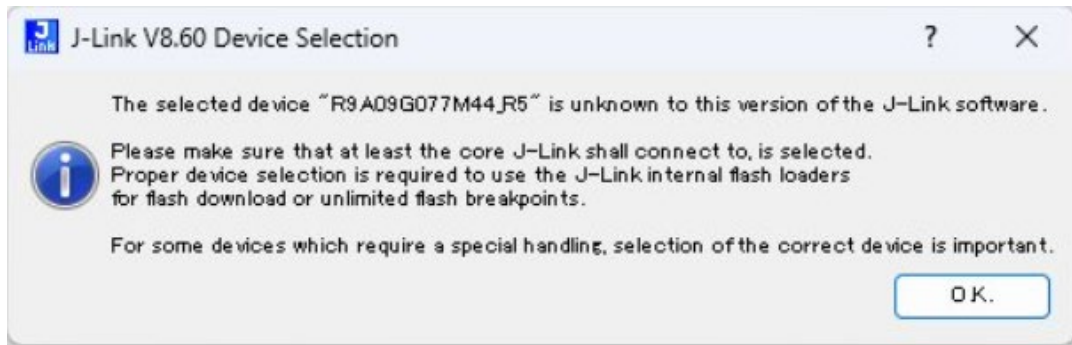
**No. 27**

<b>Limitation</b>	<b>When using the RZ/T2H CR52 or RZ/N2H CR52 core as the secondary project, an unintended source file is displayed during multicore debugging.</b>
<b>Target Device</b>	<b>RZ/T2H(CR52), RZ/N2H(CR52)</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	<p>When you create a Blinky project for the RZ/T2H CR52 or RZ/N2H CR52 core as the secondary project in multicore debugging, the command to jump to hal_entry() may not display hal_entry.c source file, and an unintended file may be displayed. As a workaround, right-click the project in Project Explorer, click <b>Properties &gt; C/C++ Build &gt; Settings &gt; Tool Settings &gt; Debugging &gt; Default format</b>, and change "Toolchain default" to "dwarf-4".</p>

**No. 28**

<b>Limitation</b>	<b>When multiple packs are imported, a message appears during debugging indicating that an unknown device is selected.</b>
<b>Target Device</b>	<b>RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H</b>
<b>Category</b>	<b>e<sup>2</sup> studio</b>
<b>Description</b>	This message appears because setting "Prevent Releasing the Reset of the CM3 Core" in the debug configuration is not enabled. It is typically enabled by default, but may become

disabled when importing RZ FSP v4.0.0 and an older RZT or RZN FSP packs. You can import both the RZ FSP pack (internal/projectgen/rz\_fsp) and the older RZT or RZN FSP pack (internal/projectgen/rz\_fsp/rzt or internal/projectgen/rz\_fsp/rzn) into the same folder. However, be aware that if you generate an RZT or RZN project using RZ FSP v4.0.0 in this state, the setting will be disabled.



## 8. Appendix.

### 8.1 How to Debug the FSP Project with Flash Boot Mode

Various flash boot modes exist depending on the device; FSP supports xSPI, NOR flash, eSD, and eMMC boot modes. The method for programming projects with xSPI or NOR flash boot mode to flash memory is the same as the RAM execution without flash memory, as explained in the main chapters, using IDE (e<sup>2</sup> studio or IAR EWARM). This chapter explains the precautions for debugging in such cases. For eSD and eMMC boot modes, tools other than the IDE are required when programming to external flash memory devices. Please refer to the FSP Documentation for details on creating image files, the programming process, and debugging procedures.

- eSD boot ([RZT](#), [RZN](#))
- eMMC boot ([RZT](#), [RZN](#))

When debugging FSP projects with flash boot modes, the program cannot be stopped at the beginning of the user program (loader program). Furthermore, there may be limitations depending on your IDE or device. Please note the following point to debug the user program from the beginning.

1. **(Both e<sup>2</sup> studio and IAR EWARM)** Insert the loop part in startup\_core.c.

When debugging is started, the debugger stops the user program (loader program) a few hundred milliseconds or more after the device boot process (boot code). If using e<sup>2</sup> studio, the PC (program counter) is replaced at the entry point (first line in **system\_init()** function) after the debugger stops; otherwise, the PC points to an address somewhere in the user program.

#### Note for Multiprocessing Projects:

Only the primary project requires the following step. No modification is required for the secondary or subsequent projects.

When debugging the program immediately after the boot process, insert the loop part in

- src/hal\_entry.c

Core	Boot mode*	Content to insert at the first line of R_BSP_WarmStart_StackLess() function
CR52	xSPI NOR flash	<pre> BSP_ATTRIBUTE_STACKLESS void R_BSP_WarmStart_StackLess (void) {   /* The very beginning of the startup process. */   #if 1 // Software loops are only needed when debugging.     __asm volatile (       " mov r0, #0          \n"       " movw r1, #0x68bf    \n"       " movt r1, #0x478     \n"       "software_loop:      \n"       " adds r0, #1         \n"       " cmp r0, r1         \n"       " bne software_loop  \n"       ::: "memory");   #endif   /* Do not delete. Required to return to system_init. */   #if (0 == BSP_LP64_SUPPORT)     __asm volatile ("BX lr");   #endif   ..... </pre>
CA55	xSPI	<pre> BSP_ATTRIBUTE_STACKLESS void R_BSP_WarmStart_StackLess (void) {   /* The very beginning of the startup process. */   #if 1 // Software loops are only needed when debugging.     __asm volatile (       " mov x1, #0          \n"       " movz x2, #0x68bf    \n"       " movk x2, #0x478, LSL #16 \n"       "software_loop:      \n"       " adds x1, x1, #1     \n"       " cmp x1, x2         \n"       " b.ne software_loop \n"       ::: "memory");   #endif   /* Do not delete. Required to return to system_init. */   #if (0 == BSP_LP64_SUPPORT) </pre>

		<code>__asm volatile ("BX lr");</code> .....
--	--	---

\* Since eSD and eMMC boot modes require different numbers of loops, please refer to FSP Documentation.

**Note:**

The required waiting time varies in proportion to the size of the executable file of the project using FSP. Therefore, when the executable file size is large, the number of loop processes added above should be adjusted.

In this process, the loop count is expressed in hexadecimal, and the 32-bit loop count is divided into the upper 16-bit and lower 16-bit and set in a general-purpose register. The following shows the procedure for changing the loop count of the CR52 core from 50000000 to 100000000, which is twice the number of loops.

1. Convert the loop count from decimal to hexadecimal.     100000000d = 0x5f5 e0ff
2. Replace the operand of movw\* to the lower 16-bit value.   movw r1, #0xe0ff
3. Replace the operand of movt\* to the upper 16-bit value.   movt r1, #0x5f5

\* In the CA55 core, opcodes are different.

## 2. (IAR EWARM) (RZ/N2H Only) Override the board files.

When RZ/N2H in xspi boot mode, it's required to override the .board files for RZ/T2H (*FlashRZT2H\_EVB\_A55.board*, *FlashRZT2H\_EVB\_R52.board*) to flash the board.

Please reselect the execution path by following the steps below:

- i. Click on **Project** and then click on **Option...** to open the project option window.
- ii. Select **Debugger** category and **Download** Tab.
- iii. Enable **“Override default .board file.”**
- iv. Select path to override the .board files for RZ/T2H
  - CR52 project: TOOLKIT\_DIR\$\config\flashloader\Renesas\FIashRZT2H\_EVB\_R52.board
  - CA55 project: TOOLKIT\_DIR\$\config\flashloader\Renesas\FIashRZT2H\_EVB\_A55.board

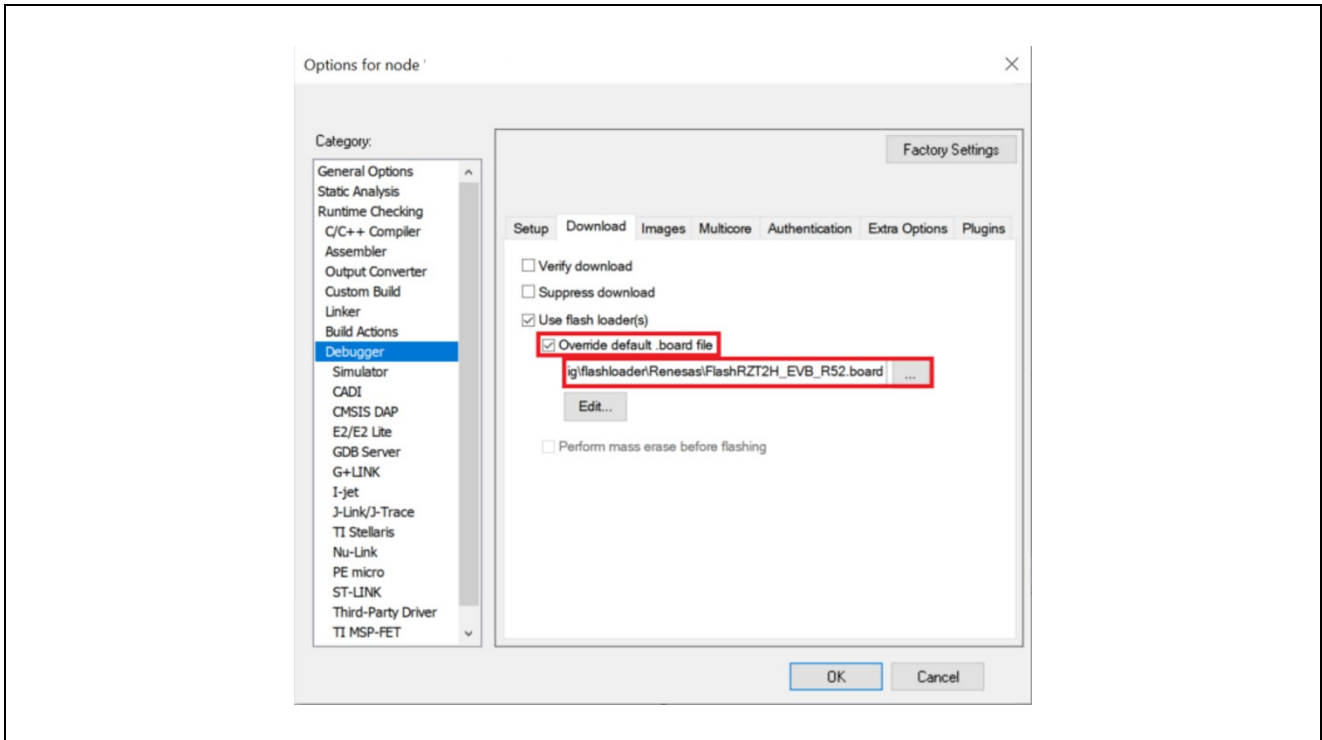


Figure 152. Project Options – Debugger (RZ/N2H Only)

## 9. Appendix

### 9.1 How to Erase Flash Memory

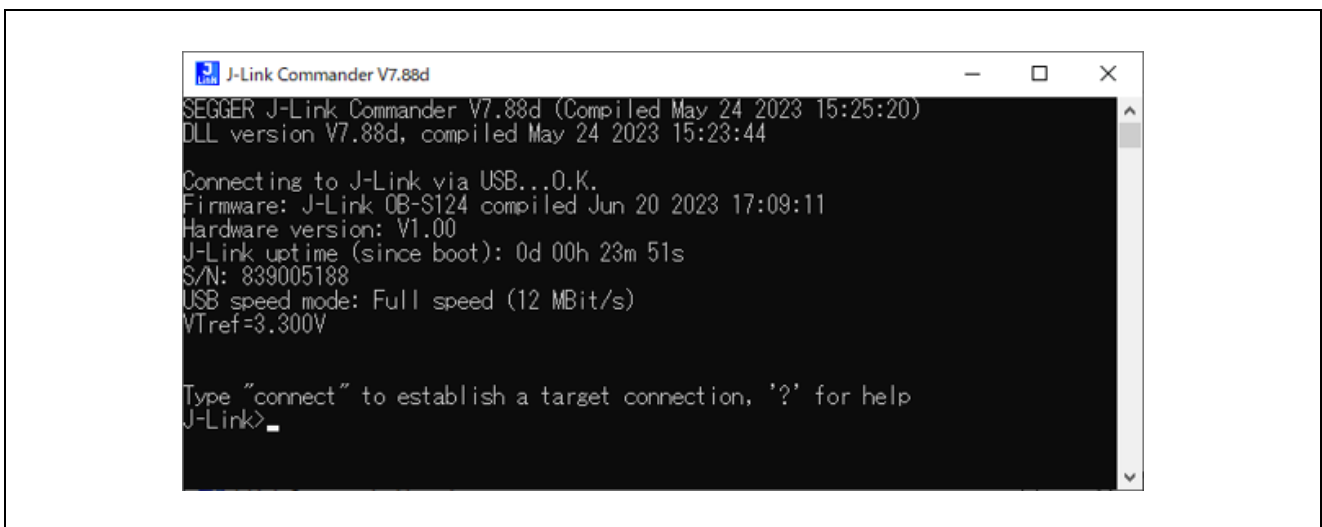
If you run RAM execution without a flash memory project with a program written in flash memory, it may be impossible to debug the project.

Please erase flash memory by following the steps depending on your IDE (e<sup>2</sup> studio or IAR EWARM) before running the project.

#### 1. e<sup>2</sup> studio

If you would like to erase the flash memory on the board using J-Link Commander, execute the following steps.

- i) Set the switch for boot mode on RSK to correspond to the area to be erased.
- ii) Open the J-Link Commander.



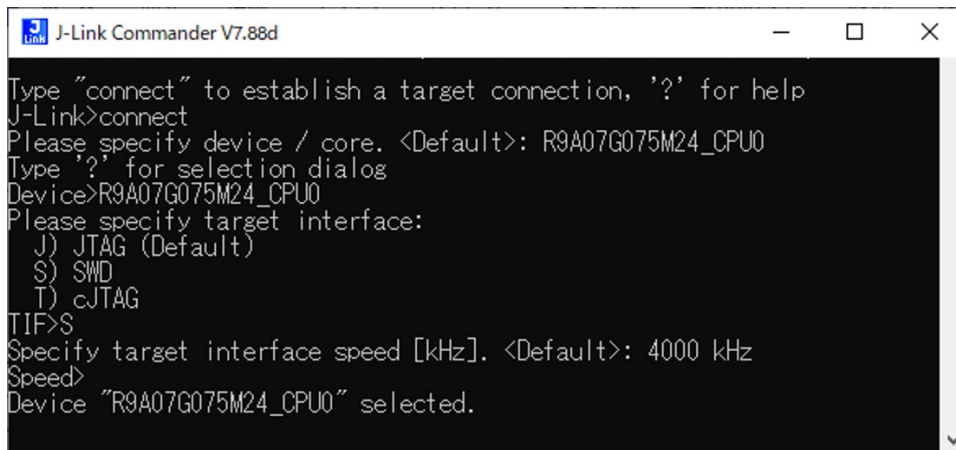
**Figure 153. Launch J-Link Commander**

- iii) First, type “connect” to establish a target connection and press enter. Next, specify the connection conditions as follows.
  - Device> (Device type name)

**Table 21. Device Type Name on Renesas Board**

Board	Device type name
RSK + RZT2M	R9A07G075M24_CPU0
RSK + RZT2L	R9A07G074M04
RSK + RZT2ME	R9A07G075M29_CPU0
RZT2H Evaluation Board	R9A09G077M44_R52_0
RSK + RZ/N2L	R9A07G084M04
RZN2H Evaluation Board	R9A09G087M44_R52_0

- TIF>S
- Speed> (Default: press enter without inputting any data)



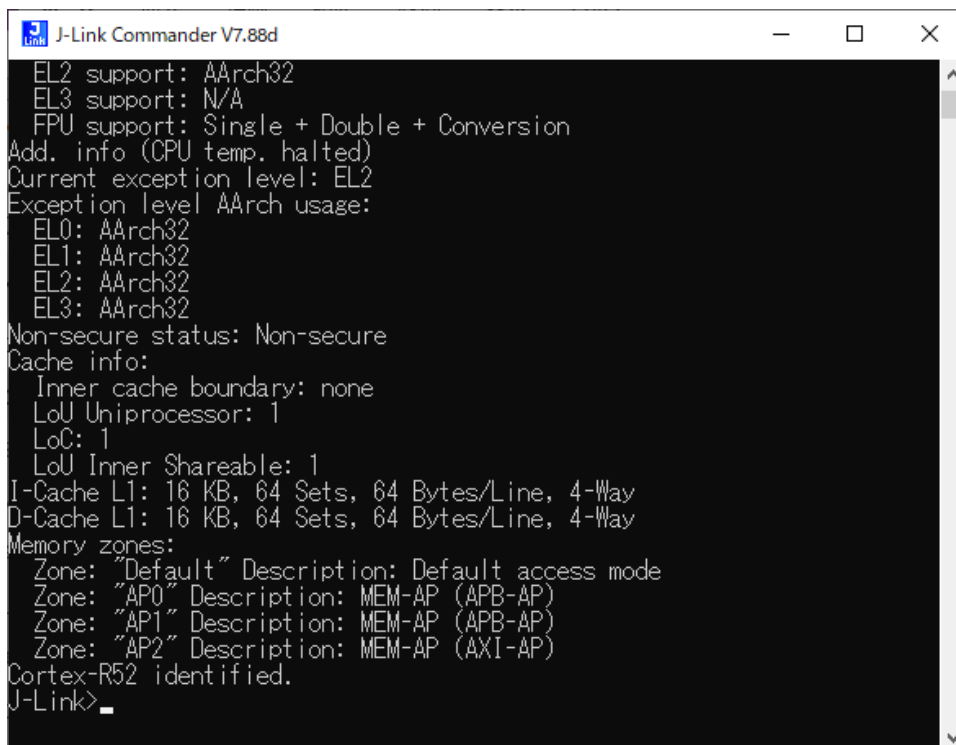
```

J-Link Commander V7.88d
Type "connect" to establish a target connection, '?' for help
J-Link>connect
Please specify device / core. <Default>: R9A07G075M24_CPU0
Type '?' for selection dialog
Device>R9A07G075M24_CPU0
Please specify target interface:
  J) JTAG (Default)
  S) SWD
  T) cJTAG
TIF>S
Specify target interface speed [kHz]. <Default>: 4000 kHz
Speed>
Device "R9A07G075M24_CPU0" selected.

```

**Figure 154. Initial Setup for Connecting to the Device**

After that, confirm that the message "Cortex-R52 identified." is displayed.



```

J-Link Commander V7.88d
EL2 support: AArch32
EL3 support: N/A
FPU support: Single + Double + Conversion
Add. info (CPU temp. halted)
Current exception level: EL2
Exception level AArch usage:
  EL0: AArch32
  EL1: AArch32
  EL2: AArch32
  EL3: AArch32
Non-secure status: Non-secure
Cache info:
  Inner cache boundary: none
  LoU Uniprocessor: 1
  LoC: 1
  LoU Inner Shareable: 1
I-Cache L1: 16 KB, 64 Sets, 64 Bytes/Line, 4-Way
D-Cache L1: 16 KB, 64 Sets, 64 Bytes/Line, 4-Way
Memory zones:
  Zone: "Default" Description: Default access mode
  Zone: "AP0" Description: MEM-AP (APB-AP)
  Zone: "AP1" Description: MEM-AP (APB-AP)
  Zone: "AP2" Description: MEM-AP (AXI-AP)
Cortex-R52 identified.
J-Link>_

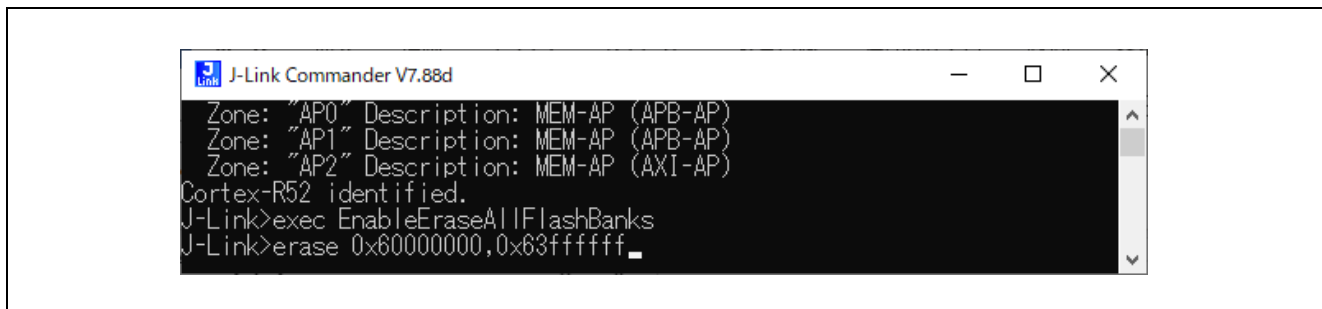
```

**Figure 155. Message of Device Core Identification**

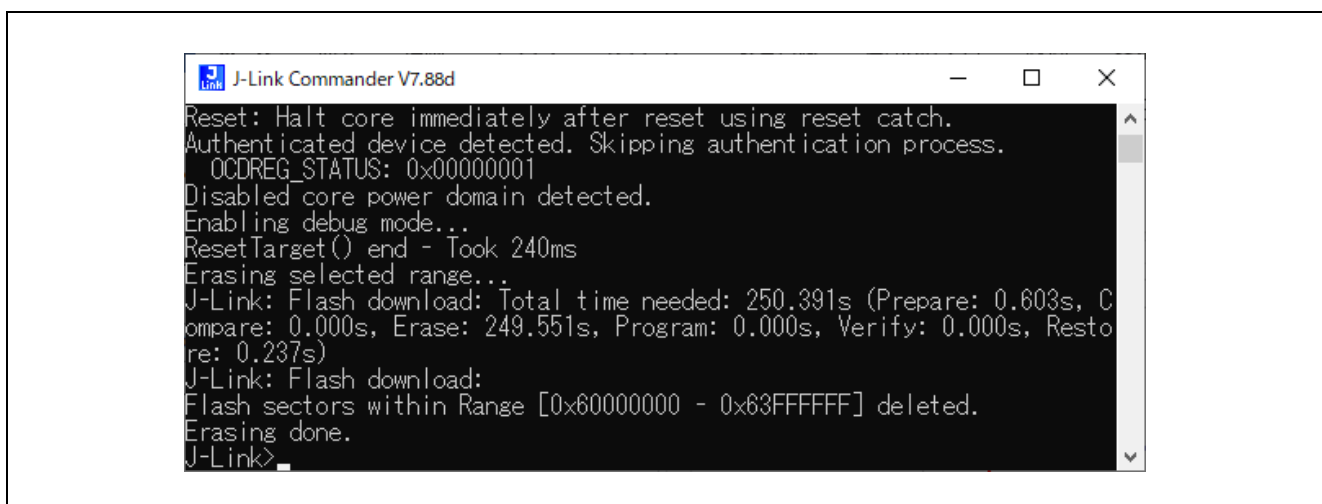
- iv) Use the commands below to enable flash erase and erase the flash memory.
- J-Link>exec EnableEraseAllFlashBanks
  - J-Link>erase (Start address), (Endaddress)

**Table 22. External Address Space to Be Used in Each Boot Mode**

Board	Boot mode	External address space to be used	Start address	End address
RSK + RZT2M, RSK + RZT2ME	xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
	16-bit bus	CS0	0x70000000	0x71FFFFFF
RSK + RZT2L	xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
	xSPI1 x1	xSPI1 CS0	0x68000000	0x68FFFFFF
RZT2H Evaluation Board	xSPI0 x1	xSPI0 CS0	0x40000000	0x47FFFFFF
	xSPI1 x1	xSPI1 CS0	0x50000000	0x57FFFFFF
RSK + RZN2L	xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
	16-bit bus	CS0	0x70000000	0x71FFFFFF
RZN2H Evaluation Board	xSPI0 x1	xSPI0 CS0	0x40000000	0x47FFFFFF
	xSPI1 x1	xSPI1 CS0	0x50000000	0x57FFFFFF

**Figure 156. Specify Erase Range**

After that, confirm that the message "Erasing done." is displayed.

**Figure 157. Message of Flash Memory Erase Complete**

- v) Enter "q" to exit J-Link Commander.

## 2. IAR EWARM

If you want to erase the flash memory on the board using IAR EWARM, execute the following steps. If the asymmetric multicore setting is enabled, the erase function cannot be used; it must be disabled.

Disable asymmetric multicore setting:

- a. Click **Project > Options....**
  - b. Click **Debugger > Multicore** and select **Disable** in Asymmetric multicore.
- i) Set the switch for boot mode on the board to correspond to the area to be erased.
  - ii) Open the workspace of a project.  
xxx.eww

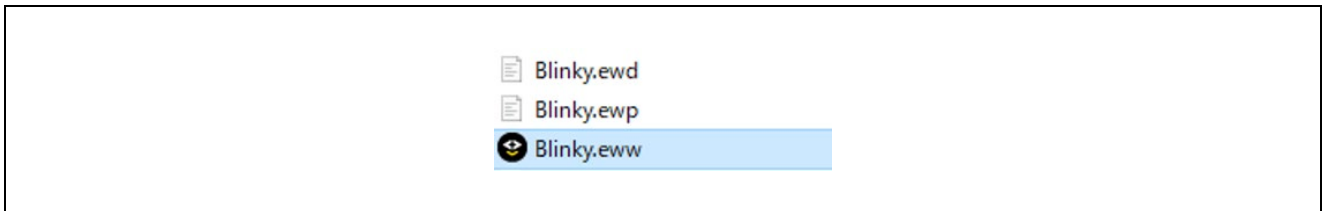


Figure 158. Open Workspace for IAR EWARM

- iii) Select "Project" -> "Download" -> "Erase memory".

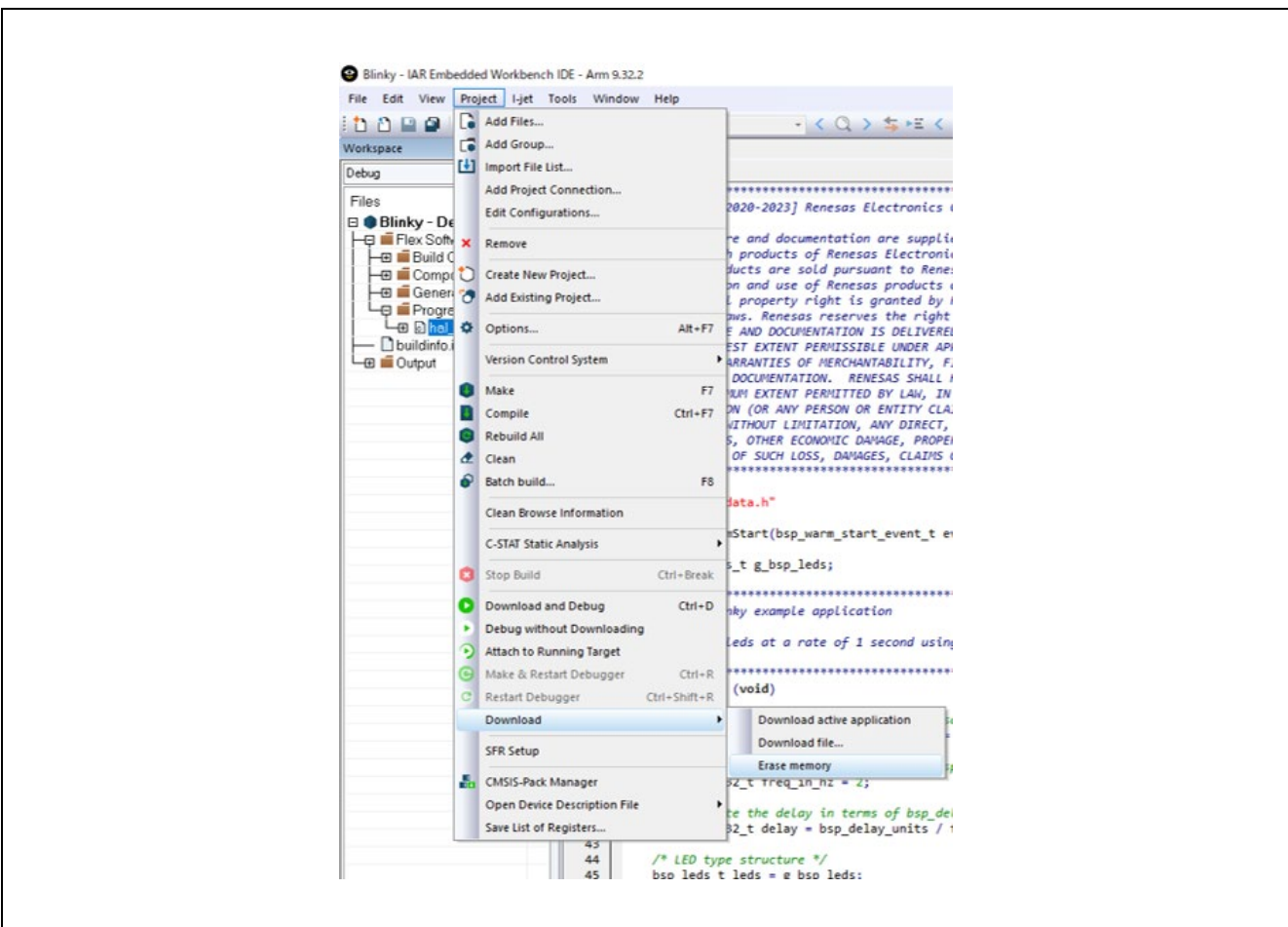


Figure 159. Select Erase Memory Command

- iv) Select erase memory space.

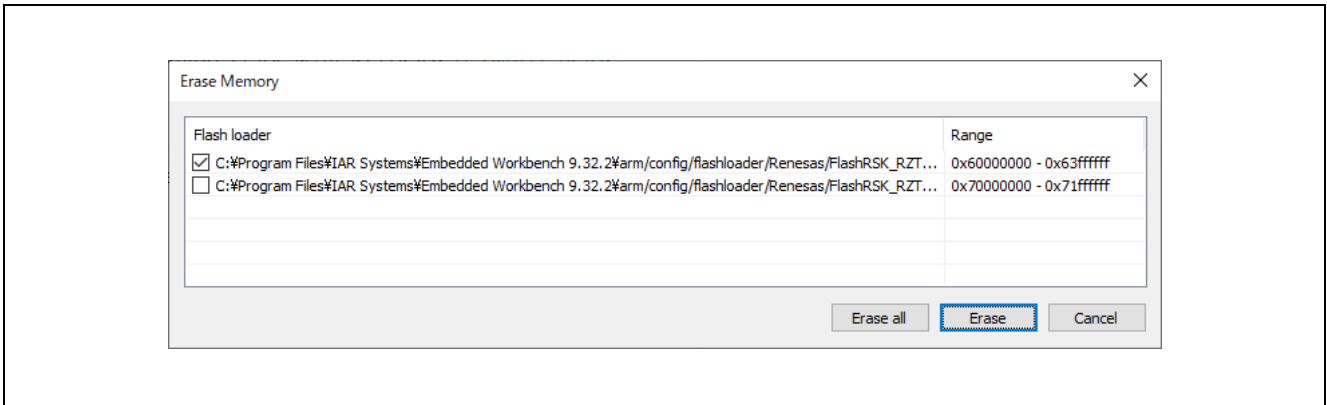


Figure 160. Select Erase Memory Space

- v) After the following dialog appears, erasing of the flash is complete if no error occurs.

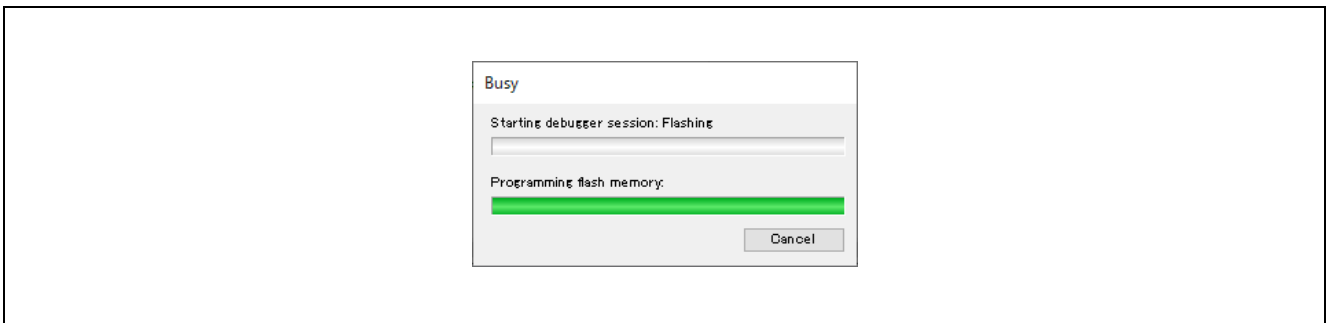


Figure 161. Screen During Erasing

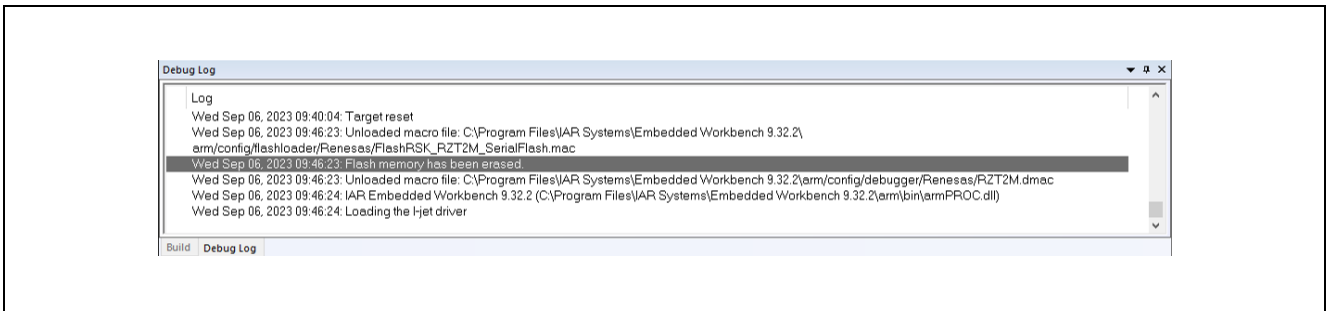


Figure 162. Message of Flash Memory Erase Complete

## 10. Appendix

### 10.1 How to Change Boot Mode of the FSP Project

When the boot mode of the project is changed, the Pin Configuration needs to be recreated.

It also needs to rename and save the pin configuration to retain the original one before changing the boot mode.

For example, one of the specific cases in which re-configuration is necessary is when a RAM execution without flash memory project is changed to flash boot mode (xSPI0 x1 boot mode and others).

Please change the boot mode by following the steps.

#### Note for FSP version earlier than v1.3.0:

If the FSP version of your project is earlier than FSP v1.3.0, change it to FSP v1.3.0 before doing the following steps.

1. Rename and save the current Pin Configuration in the **Pins** tab.
  - How to rename Pin Configuration: Click “Manage Configurations...”

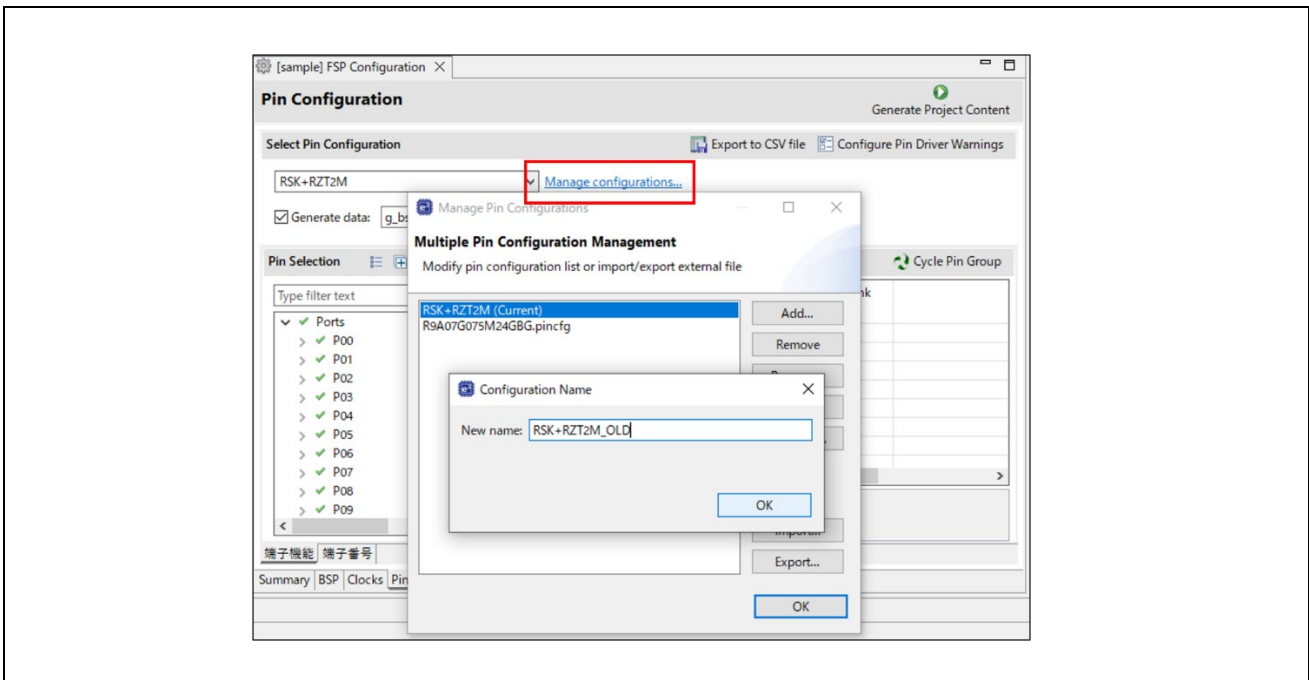
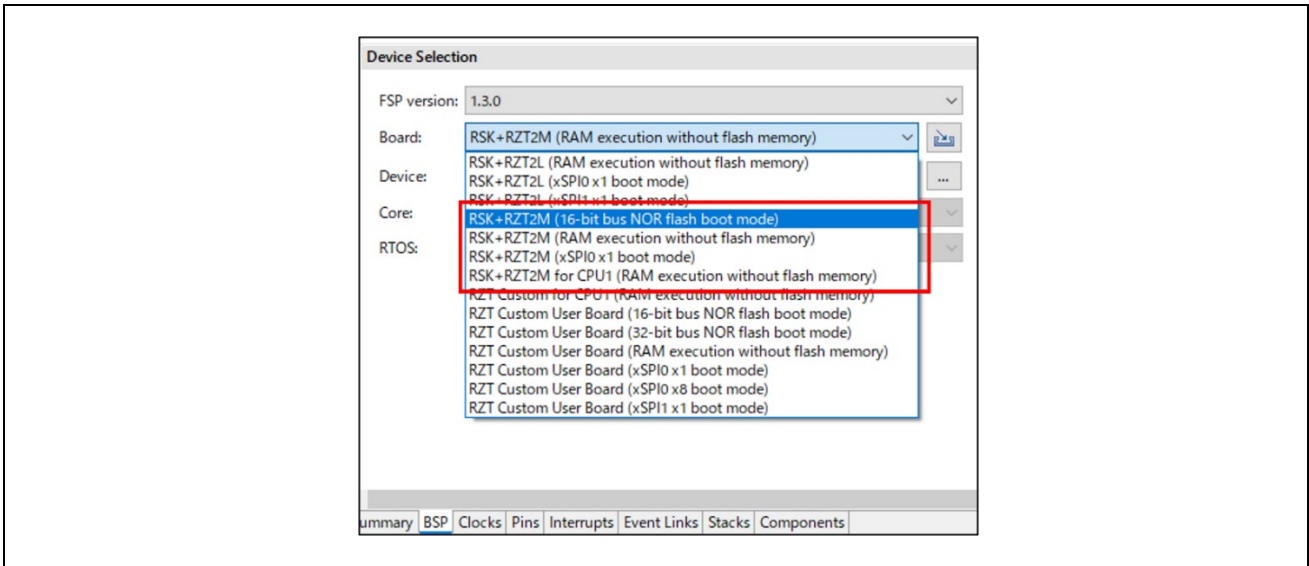


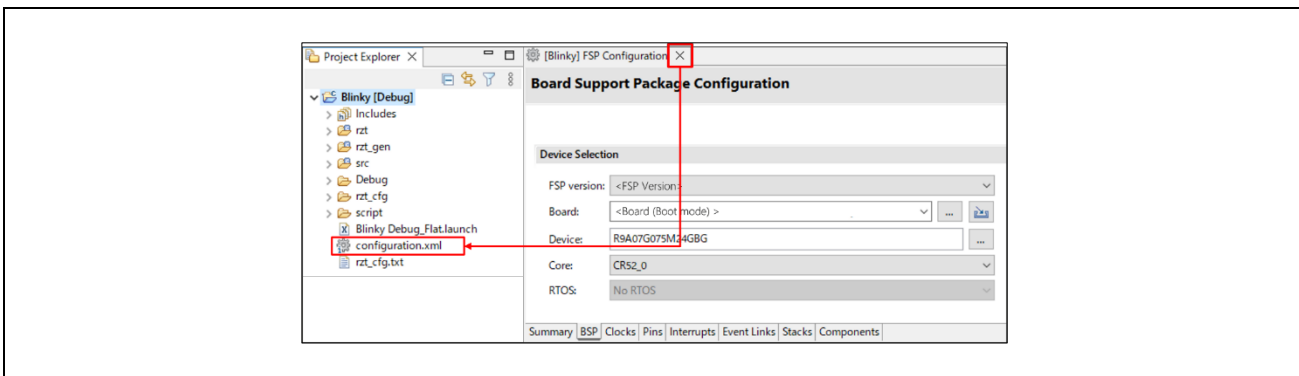
Figure 163. How to Rename Pin Configuration

2. Change the boot mode in the **BSP** tab. (The board must be the same as before the change.)



**Figure 164. Change the Boot Mode in the BSP Tab**

3. Close the configuration and reopen it.



**Figure 165. Close Configuration and reopen**

- 4. Uncheck "Generate data" in the **Pins** tab.

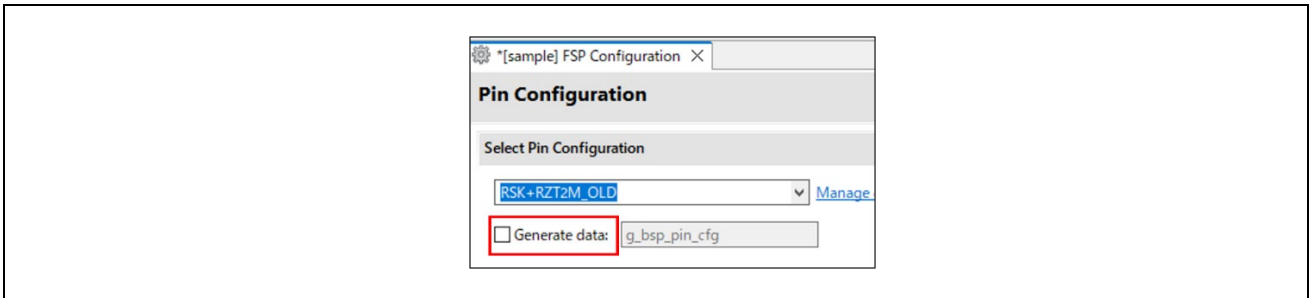


Figure 166. Uncheck Generate data in the Pins Tab

- 5. Select the regenerated configuration for the board.

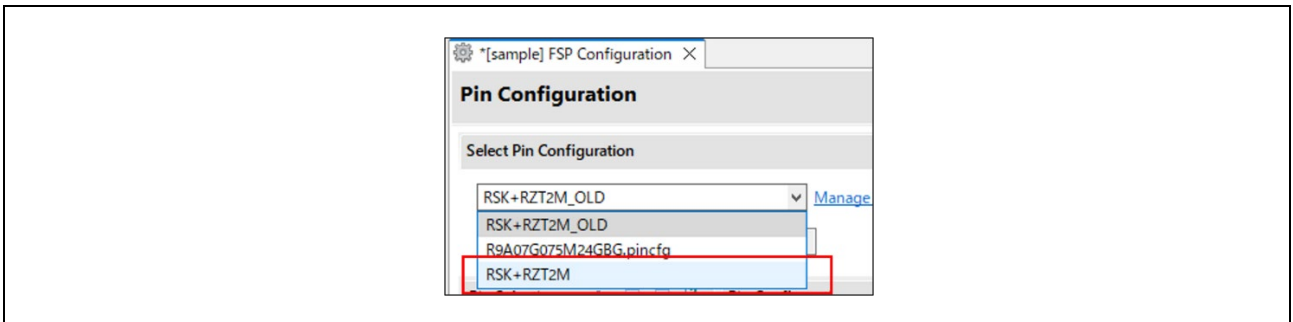


Figure 167. Select the Regenerated Configuration for the Board

- 6. Check "Generate data" again and enter "g\_bsp\_pin\_cfg" as the name.

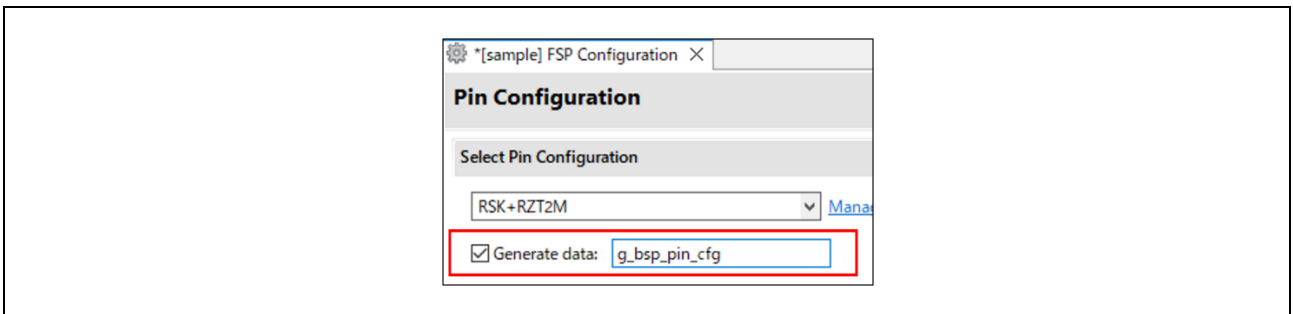


Figure 168. Check the generated data again

## Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e<sup>2</sup> studio

4 Tutorial: Your First RZ MPU Project – Blinky describes how to create and debug a project in e<sup>2</sup> studio for RAM execution of a single-core process using CR52\_0 and multiprocessing processes, where CR52\_0 is the primary core. This chapter describes project creation and debugging methods in e<sup>2</sup> studio applicable to other boot modes and core combinations.

If the procedure is preceded by (XXX), it is executed only if the condition is met.

(RAM exec): The boot mode used in the project is RAM execution without flash memory.

(Flash boot): The boot mode used in the project is the NOR flash boot mode or the xSPI flash boot mode.

(CR52): The core used in the project is CR52.

(CA55): The core used in the project is CA55.

(Different core types in N and M) In multiprocessing, the different types of cores in the N and M projects (N, M = pri(primary), sec(secondary), ter(tertiary)).

(N is CR52 and M is CA55): In multiprocessing, a CR52 core is used in N project and a CA55 core is used in M project (N, M = pri(primary), sec(secondary), ter(tertiary)).

### Multiprocessing with 2 cores

1. Create a primary project according to the procedures in 4.3 Create a New Project for Blinky.
  - (RZ/T2H CA55 Core1 and RZ/N2H CA55 Core1) To blink the LED on the RZ/T2H CA55 Core1 and RZ/N2H CA55 Core1, you need to change the pin configuration of the primary project in the Smart Configurator as follows.
    - i. After creating the primary project in 4.3 Create a New Project for Blinky No. 11, set the pins before clicking **Generate Project Content**.
    - ii. In the Pins tab of the FSP configuration, click **Pin Selection -> Peripherals -> Connectivity: SDHI -> SDHI1**
    - iii. Change the value of SD1\_PWEN to None.
    - iv. In the Pins tab of the FSP configuration, click **Pin Selection -> Ports -> P08 -> P08\_5**.
    - v. Change the value of Symbolic Name to LED3.
    - vi. Change the value of Mode to Output mode (Low & Not Into Input)
  - 2. (Flash boot) Insert the loop part in startup\_core.c of the primary project with reference to Appendix.
3. How to Debug the FSP Project with Flash Boot Mode.
4. Build the primary project according to the procedures in 4.4.1 Build.
5. Create a secondary project using the bundle file (.sbd) of the primary project according to the procedures in 4.3 Create a New Project for Blinky.
6. Build the secondary project according to the procedures in 4.4.1 Build. By configuring the following additional setting for the secondary project before building it, configuration of prebuild.sh (No. 6) for the primary project is not required.
  - (Flash boot) (Different core types in pri and sec)
    - (pri is CA55 and sec is CR52) The following additional properties must be set.
      - i. In the Properties window of the secondary project, click **C/C++ Build > Environment > Add** and set it as follows:
        - a. Name: AARCH64\_OBJCOPY
        - b. Value: C:[GNU installation folder path]\Arm GNU Toolchain aarch64-none-elf\13.2 Rel1\bin\
          - **Be careful not to forget to add the trailing backslash.**
          - This is the path to the folder where "aarch64-none-elf-ld.exe" is saved.
      - ii. Skip No. 6 to the configuration of prebuild.sh and proceed to No. 7.
    - (pri is CR52 and sec is CA55) No need to set additional properties. Skip No. 6 to the configuration of prebuild.sh and proceed to No. 7.

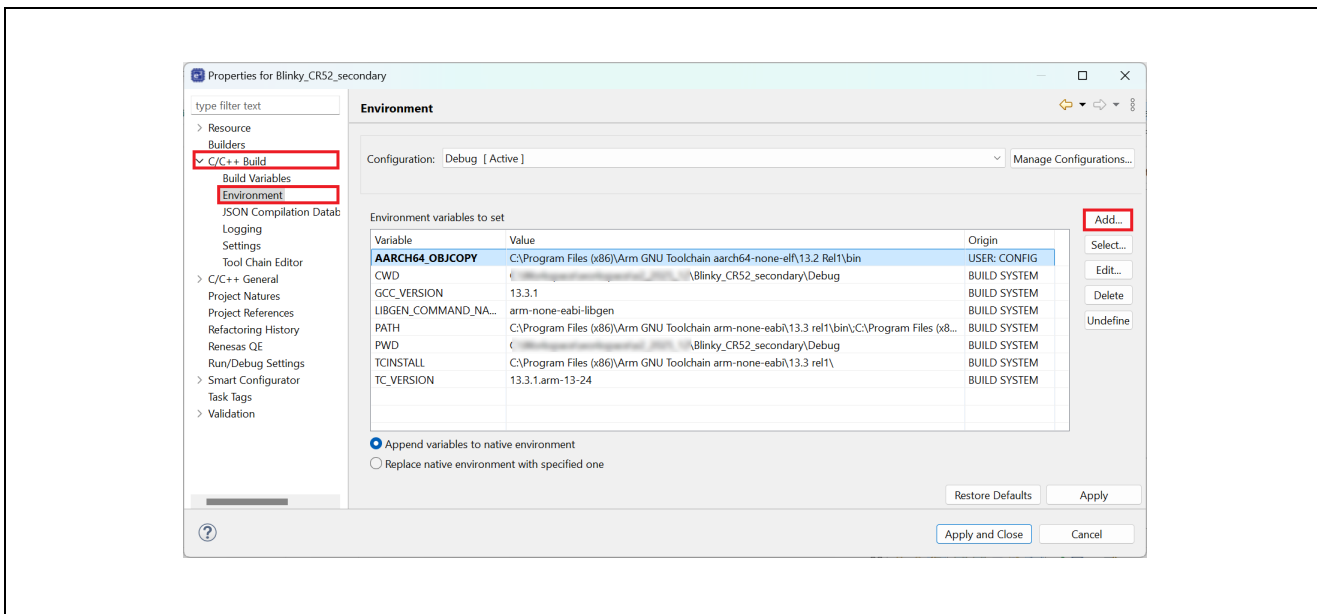


Figure 169. e2 studio Build Setting for the Secondary Project (Flash boot) (pri is CA55 and sec is CR52) (Part 1)

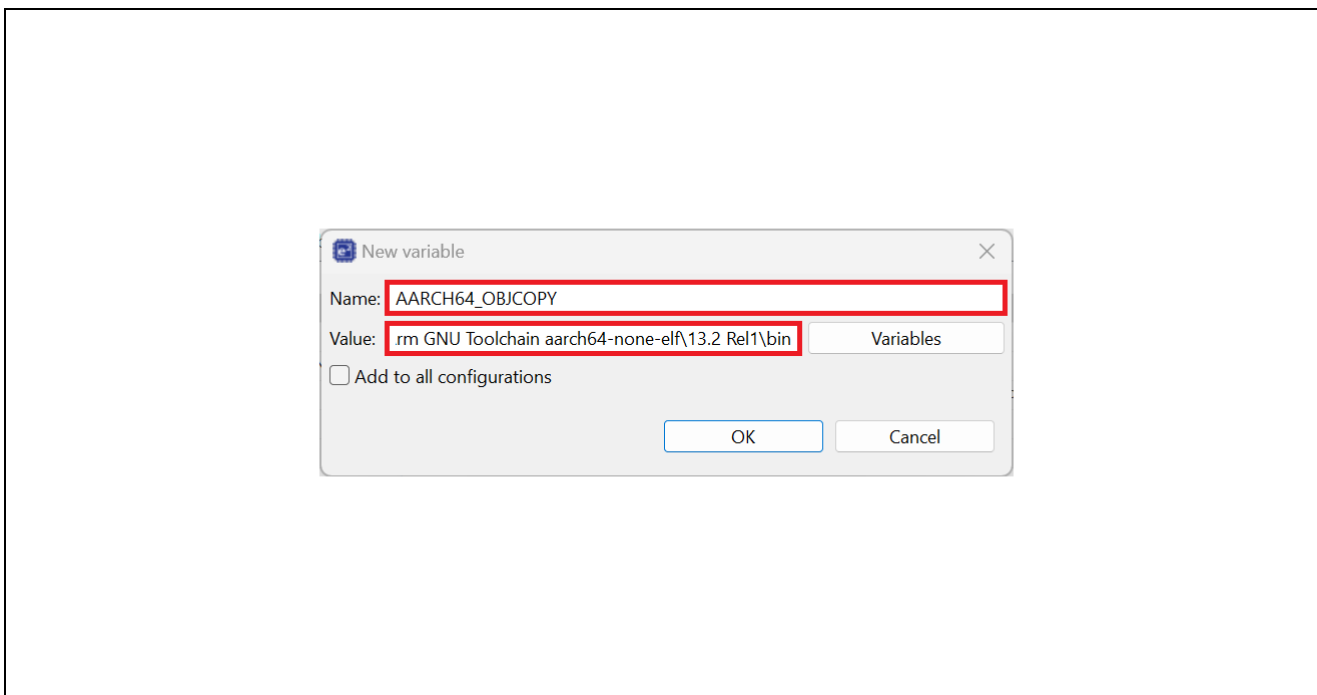


Figure 170. e2 studio Build Setting for the Secondary Project (Flash boot) (pri is CA55 and sec is CR52) (Part 2)

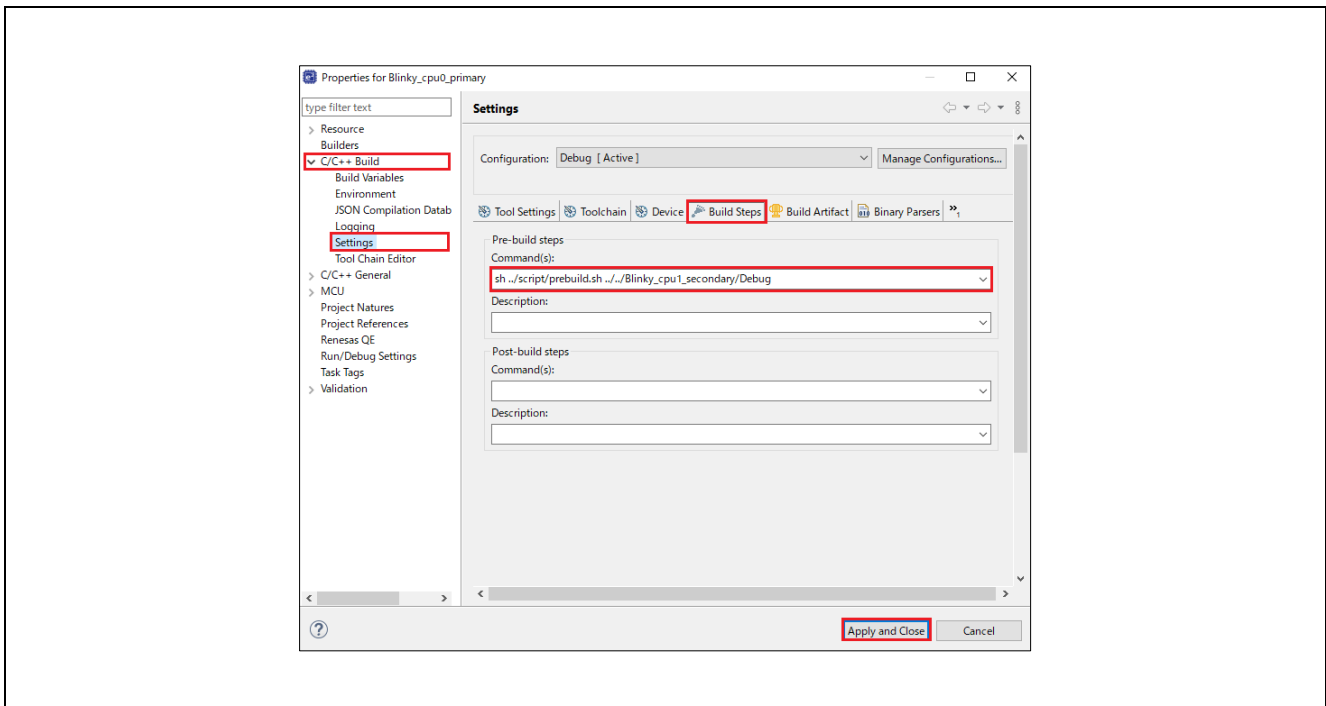
- (Flash boot) The following object files are output to the Debug folder of the secondary project.

Table 23. Object files Output to the Debug Folder of the Secondary Project

Device	Project core	Object files	Note
RZ/T2M RZ/T2ME	CR52	secondary_CR52.o	
		secondary_noncache_CR52.o	When using a non-cache section
RZ/T2H RZ/N2H	CR52	secondary_atcm_CR52_0.o	For CR52 CPU0
		secondary_btcm_CR52_0.o	For CR52 CPU0

		secondary_atcm_CR52_1.o	For CR52 CPU1
		secondary_btcm_CR52_1.o	For CR52 CPU1
		secondary_systemram_CR52.o	For a multi-core project with 3 or more cores
		secondary_CR52.o	When placing a program in System SRAM instead of TCM
		secondary_noncache_CR52.o	When using a non-cache section
	CA55	secondary_CA55.o	
		secondary_noncache_CA55.o	When using a non-cache section

7. (Flash boot) (Different core types in pri and sec) The following additional properties must be set.
  - i. In the Properties window of the primary project, click **C/C++ Build > Settings > Build Steps**.
  - ii. Add **Command(s)** at **Pre-build steps**.  
`sh ../script/prebuild.sh ../[the secondary project name]/Debug`



**Figure 171. e<sup>2</sup> studio Build Setting for the Primary Project (Flash Boot) (Different core types in pri and sec)**

8. (Flash boot) (Different core types in pri and sec) Build the primary project according to the procedures in 4.4.1 Build. The following object files are output to the Debug folder of the secondary project. If No. 6 is skipped, these files will be generated during the secondary project build in No. 5.

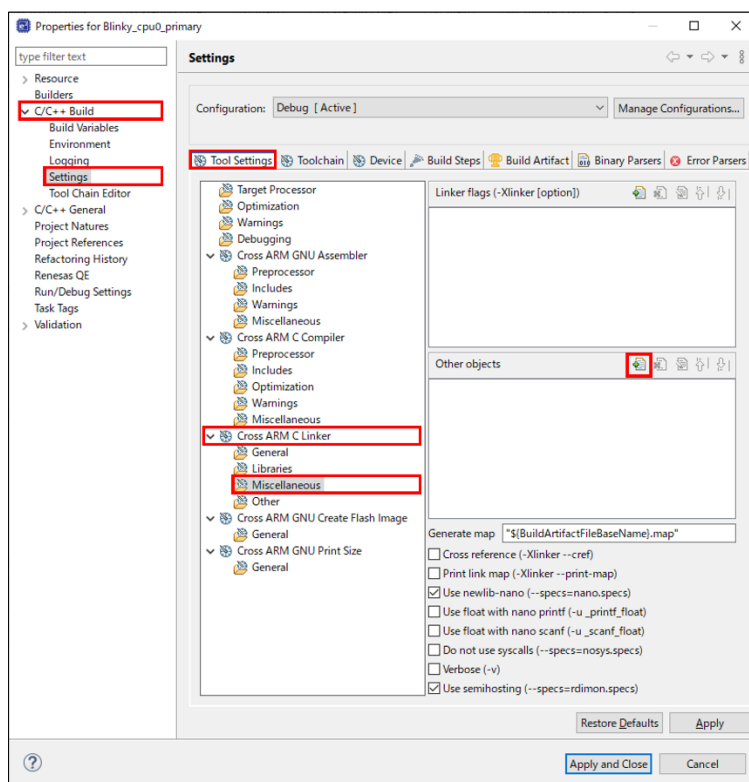
**Table 24. Object files Output to the Debug Folder of the Secondary Project**

Primary project core	Secondary project core	Object files	Note
CA55	CR52	secondary_atcm_aarch64_CR52_0.o	For CR52 CPU0
		secondary_btcm_aarch64_CR52_0.o	For CR52 CPU0
		secondary_atcm_aarch64_CR52_1.o	For CR52 CPU1
		secondary_btcm_aarch64_CR52_1.o	For CR52 CPU1
		secondary_systemram_aarch64_CR52.o	For a multi-core project with 3 or more cores
		secondary_aarch64_CR52.o	When placing a program in System SRAM instead of TCM
CR52	CA55	secondary_aarch64_noncache_CR52.o	When using a non-cache section
		secondary_noncache_aarch32_CA55.o	When using a non-cache section

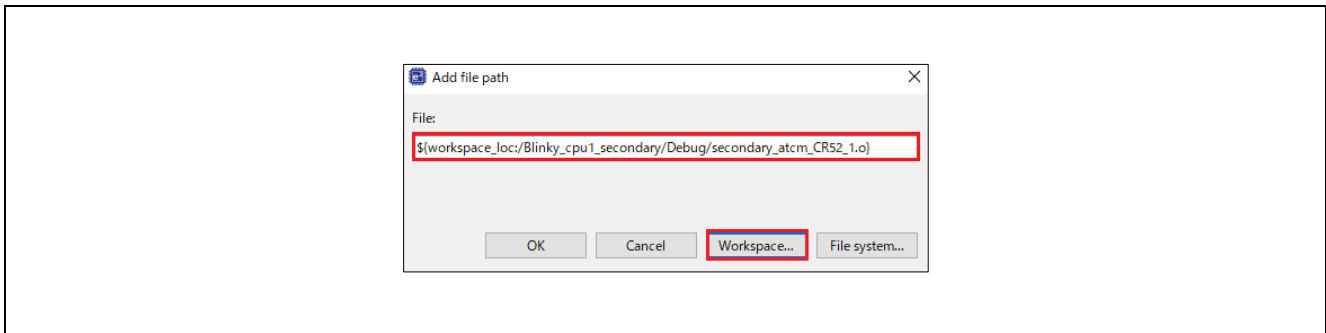
9. (Flash boot) Set the following additional properties to the primary project.
  - i. In the Properties window of the primary project, click **C/C++ Build > Settings > Tool Settings > Cross ARM C Linker > Miscellaneous**.
  - ii. Set file paths of object files in the secondary project to **Other objects**.

**Table 25. Example of File Paths to Be Set to Other Objects**

Device	Primary project core	Secondary project core	File path	Note
RZ/T2M RZ/T2ME	CR52 CPU0	CR52 CPU1	\${workspace_loc:/Blinky_cpu1_secondary/Debug/secondary_CR52.o}	
			\${workspace_loc:/Blinky_cpu1_secondary/Debug/secondary_noncache_CR52.o}	Import only if the file is the output
RZ/T2H RZ/N2H	CR52 CPU0	CR52 CPU1	\${workspace_loc:/Blinky_cpu1_secondary/Debug/secondary_atcm_CR52_1.o}	
			\${workspace_loc:/Blinky_cpu1_secondary/Debug/secondary_btcm_CR52_1.o}	
			\${workspace_loc:/Blinky_cpu1_secondary/Debug/secondary_noncache_CR52.o}	Import only if the file is the output
	CA55 Core0	CA55 Core1	\${workspace_loc:/Blinky_cpu1_secondary/Debug/secondary_CA55.o} \${workspace_loc:/Blinky_cpu1_secondary/Debug/secondary_noncache_CA55.o}	Import only if the file is the output



**Figure 172. e2 studio Build Setting for the Primary Project (Part 1)**

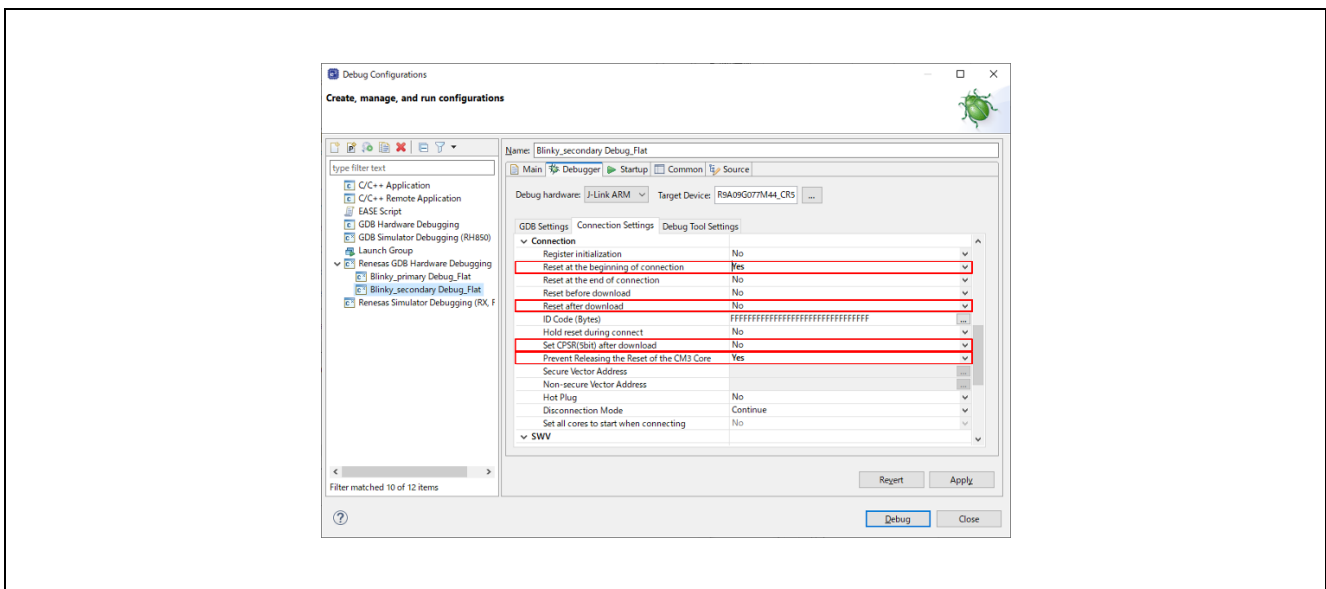


**Figure 173. e<sup>2</sup> studio Build Setting for the Primary Project (Part 2)**

- 10. (Flash boot) Build the primary project.  
Debug the projects according to the procedures in 0
- 11. Debug and Run for Multiprocessing.
  - Check the debug configuration in the No. 3 procedure of 4.5.2 Debug Steps.

**Table 26. e<sup>2</sup> studio Debug Configurations for Multiprocessing**

Item	Project core	Primary project		Secondary or later project	
		RAM exec	Flash boot	RAM exec	Flash boot
Reset at the beginning of the connection.	CR52 CPU0 CA55 Core0	Yes		No	
	Other cores		-	Yes	
Reset after download	-	No	Yes	No	
Set CPSR(5bit) after download	CR52 CPU0	Yes	No	Yes	
	Other cores	No		No	
Prevent Releasing the Reset of the CM3 Core	CR52 CPU1		-	Yes	



**Figure 174. e<sup>2</sup> studio Debug Configurations for Multiprocessing**

- (Flash boot) The flash boot mode differs from RAM execution without flash memory. Both the primary and secondary project binaries are downloaded to the device when connecting the debugger to the primary project.

12. When changing the project and debugging it again, follow these steps.
  - i. Build the primary project (No. 4).
  - ii. Build the secondary project (No. 6).
  - iii. (Flash boot) Right-click on the primary project and select **Clean Project**.
  - iv. (Flash boot) Build the primary project again (No. 10).
  - v. Debug the projects (No. 0).

## Multiprocessing with 3 or more cores

This section shows how to perform multi-core debugging using three cores. If more than 4 cores are used, add the process of creating a project, building it, and one previous project after No. 3.

1. Create and build the primary and secondary projects according to Multiprocessing with 2 cores No. 1 to No. 7.
2. Create a tertiary project using the bundle file (.sbd) of the secondary project according to the procedures in 4.3 Create a New Project for Blinky.
3. Build the tertiary project according to the procedures in 4.4.1 Build. By configuring the following additional setting for the tertiary project before building it, the configuration of the prebuild.sh (No. 4) for the tertiary project is not required.
  - (Flash boot) (Different core types in pri and ter)
    - (pri is CA55) The following additional properties must be set.
      - i. In the Properties window of the tertiary project, click **C/C++ Build > Environment > Add** and set it as follows:
        - a. Name: AARCH64\_OBJCOPY
        - b. Value: C:[GNU installation folder path]\Arm GNU Toolchain aarch64-none-elf\13.2 Rel1\bin\
          - **Be careful not to forget to add the trailing backslash**
          - This is the path to the folder where "aarch64-none-elf-ld.exe" is saved.
      - ii. Skip No. 4 to the configuration of prebuild.sh and proceed to No. 5.

**Note:** When the primary core is CA55, it is necessary to compare the core type of the primary core with the core type currently being built (third or later), and if the core types are different, properties need to be added. As an example, when building the fifth core, compare the core types of the primary and the fifth cores.

- (pri is CR52) No need to set additional properties. Skip No. 4 to the configuration of prebuild.sh and proceed to No. 5.
- (Flash boot) The following object files are output to the Debug folder of the secondary project.

**Table 27. Object files Output to the Debug Folder of the Secondary Project**

Device	Project core	Object files	Note
RZ/T2H RZ/N2H	CR52	secondary_atcm_CR52_0.o	For CR52 CPU0
		secondary_btcm_CR52_0.o	For CR52 CPU0
		secondary_atcm_CR52_1.o	For CR52 CPU1
		secondary_btcm_CR52_1.o	For CR52 CPU1
		secondary_systemram_CR52.o	For a multi-core project with 3 or more cores
		secondary_CR52.o	When placing a program in System SRAM instead of TCM
	secondary_noncache_CR52.o	When using a non-cache section	
	CA55	secondary_CA55.o	
	secondary_noncache_CA55.o	When using a non-cache section	

4. (Flash boot) (pri is CR52) (Different core types in sec and ter) or (Flash boot) (pri is CA55) (Different core types in pri and ter) The following additional properties must be set.
  - i. In the Properties window of the secondary project, click **C/C++ Build > Settings > Build Steps**.
  - ii. Add **Command(s) at Pre-build steps**.  
sh ../script/prebuild.sh ../[the tertiary project name]/Debug

5. (Flash boot) (Different core types in sec and ter) Build the secondary project according to the procedures in 4.4.1 Build. The following object files are output to the Debug folder of the tertiary project. If No. 4 is skipped, these files will be generated during the secondary project build in No. 3.

**Table 28. Object files Output to the Debug Folder of the Tertiary Project**

Secondary project core	Tertiary project core	Object files	Note
CA55	CR52	secondary_atcm_aarch64_CR52_0.o	For CR52 CPU0
		secondary_btcm_aarch64_CR52_0.o	For CR52 CPU0
		secondary_atcm_aarch64_CR52_1.o	For CR52 CPU1
		secondary_btcm_aarch64_CR52_1.o	For CR52 CPU1
		secondary_systemram_aarch64_CR52.o	For a multi-core project with 3 or more cores
		secondary_aarch64_CR52.o	When placing a program in System SRAM instead of TCM
		secondary_aarch64_noncache_CR52.o	When using a non-cache section
CR52	CA55	secondary_aarch32_CA55.o	
		secondary_noncache_aarch32_CA55.o	When using a non-cache section

6. (Flash boot) Set the following additional properties to the secondary project.
- In the Properties window of the secondary project, click **C/C++ Build > Settings > Tool Settings > Cross ARM C Linker > Miscellaneous**.
  - Set file paths of object files in the tertiary project to **Other objects**.

**Table 29. Example of File Paths to Be Set to Other Objects for the Secondary Project**

Device	Primary project core	Secondary project core	Tertiary project core	File path	Note
RZ/T2H RZ/N2H	CA55 Core0	<b>CR52 CPU0</b>	CR52 CPU1	\${workspace_loc:/Blinky_tertiary/Debug/secondary_noncache_CR52.o}	Import only if the file is the output*
	CR52 CPU0	<b>CA55 Core0</b>	CA55 Core1	\${workspace_loc:/Blinky_tertiary/Debug/secondary_CA55.o}	
				\${workspace_loc:/Blinky_tertiary/Debug/secondary_noncache_CA55.o}	Import only if the file is the output
	CR52 CPU0	<b>CR52 CPU1</b>	CA55 Core0	\${workspace_loc:/Blinky_tertiary/Debug/secondary_aarch32_CA55.o}	
				\${workspace_loc:/Blinky_tertiary/Debug/secondary_noncache_aarch32_CA55.o}	Import only if the file is the output

\* Object files for TCM, such as secondary\_atcm\_CR52\_1.o, will not work properly if they are imported into a project other than the primary project. Even object files from tertiary projects must be imported into the primary project.

7. (Flash boot) Build the secondary project.

8. (Flash boot) Set the following additional properties to the primary project.

**Table 30. Example of File Path to Be Set to Other Objects for the Primary Project**

Device	Primary project core	Secondary project core	Tertiary project core	File path	Note
RZ/T2H RZ/N2H	<b>CA55 Core0</b>	CR52 CPU0	CR52 CPU1	\${workspace_loc}/Blinky_secondary/Debug/secondary_atcm_aarch64_CR52_0.o}	
				\${workspace_loc}/Blinky_secondary/Debug/secondary_btcm_aarch64_CR52_0.o}	
				\${workspace_loc}/Blinky_secondary/Debug/secondary_noncache_aarch64_CR52.o}	Import only if file is the output.
				\${workspace_loc}/Blinky_tertiary/Debug/secondary_atcm_aarch64_CR52_1.o}	
				\${workspace_loc}/Blinky_tertiary/Debug/secondary_btcm_aarch64_CR52_1.o}	
	<b>CR52 CPU0</b>	CA55 Core0	CA55 Core1	\${workspace_loc}/Blinky_secondary/Debug/secondary_aarch32_CA55.o}	
				\${workspace_loc}/Blinky_secondary/Debug/secondary_noncache_aarch32_CA55.o}	Import only if the file is the output.
	<b>CR52 CPU0</b>	CR52 CPU1	CA55 Core0	\${workspace_loc}/Blinky_secondary/Debug/secondary_atcm_CR52_1.o}	
				\${workspace_loc}/Blinky_secondary/Debug/secondary_btcm_CR52_1.o}	
				\${workspace_loc}/Blinky_secondary/Debug/secondary_systemram_CR52.o}	
				\${workspace_loc}/Blinky_secondary/Debug/secondary_noncache_CR52.o}	Import only if the file is the output

**Note:**

If the primary project is CA55 and the secondary and tertiary projects are CR52, the following additional properties must be set.

- i. In the Properties window of the primary project, click **C/C++ Build > Settings > Build Steps**.
- ii. Add **Command(s) at Pre-build steps**.  

```
sh ../script/prebuild.sh ../[the secondary project name]/Debug &&
sh ../script/prebuild.sh ../[the tertiary project name]/Debug
```

9. (Flash boot) Build the primary project.
10. Debug the projects according to the procedures in Multiprocessing with 2 cores No. 10. Connections are made in the order primary, secondary, and tertiary. The tertiary project is connected in the same procedure as the secondary project.
11. When changing the project and debugging it again, follow these steps.
  - i. Build the primary project (Multiprocessing with 2 cores Multiprocessing with 2 cores No. 4).
  - ii. Build the secondary project (Multiprocessing with 2 cores No. 6).
  - iii. Build the tertiary project (Multiprocessing with 3 or more cores Multiprocessing with 3 or more cores No. 3).
  - iv. (Flash boot) Right-click on the secondary project and select **Clean Project**.
  - v. (Flash boot) Build the secondary project (Multiprocessing with 3 or more cores No. 7).
  - vi. (Flash boot) Right-click on the primary project and select **Clean Project**.
  - vii. (Flash boot) Build the primary project again (Multiprocessing with 3 or more cores No. 9).
  - viii. Debug the projects (Multiprocessing with 3 or more cores No. 10).

## Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM

5.3 Using FSP SC with IAR EWARM describes how to create in FSP SC and debug a project in IAR EWARM for RAM execution of a single core process using CR52\_0 and multiprocessing processes, where CR52\_0 is the primary core. This chapter describes project creation in FSP SC and debugging methods in IAR EWARM applicable to other boot modes and core combinations.

If the procedure is preceded by (XXX), it is executed only if the condition is met.

(RAM exec): The boot mode used in the project is RAM execution without flash memory.

(Flash boot): The boot mode used in the project is the NOR flash boot mode or the xSPI flash boot mode.

(CR52): The core used in the project is CR52.

(CA55): The core used in the project is CA55.

### Multiprocessing with 2 cores

1. Create projects according to the procedures in 5.3.2 Create a New Project.
  - (RZ/T2H CA55 Core1 and RZ/N2H CA55 Core1) To blink the LED on the RZ/T2H CA55 Core 1 and RZ/N2H CA55 Core1, you need to change the pin configuration of the primary project in the FSP SC as follows.
    - i. In 5.3.2 Create a New Project No. 14, set the pins before clicking **Generate Project Content**.
    - ii. In the Pins tab of FSP configuration, click **Pin Selection -> Peripherals -> Connectivity:SDHI -> SDHI1**
    - iii. Change the value of SD1\_PWEN to None.
    - iv. In the Pins tab of FSP configuration, click **Pin Selection -> Ports -> P08 -> P08\_5**.
    - v. Change the value of Symbolic Name to LED3.
    - vi. Change the value of Mode to Output mode (Low & Not Into Input)
  - 2. (Flash boot) Insert the loop part in startup\_core.c of the primary project with reference to Appendix.
3. How to Debug the FSP Project with Flash Boot Mode. For RZ/N2H, the board file override is also required.
4. Build the primary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 1.
5. Create a secondary project using the bundle file (.sbd) of the primary project according to the procedures in 5.3.2 Create a New Project.
6. The following additional properties must be set.
  - i. Click **Project > Options....**
  - ii. (Flash boot) Click **Build Actions > Build Actions Configuration > New** and set it as follows:
    - a. Command line
      - `ielftool --bin-multi=0x0-0x7FFFF;0x102000-0x10FFFF;__region_CACHE_start__ - __region_CACHE_end__;__region_NONCACHE_INIT_start__ - __region_NONCACHE_INIT_end__ $TARGET_BPATH$.out $TARGET_BPATH$.bin`
    - b. Build order
      - Run after linking

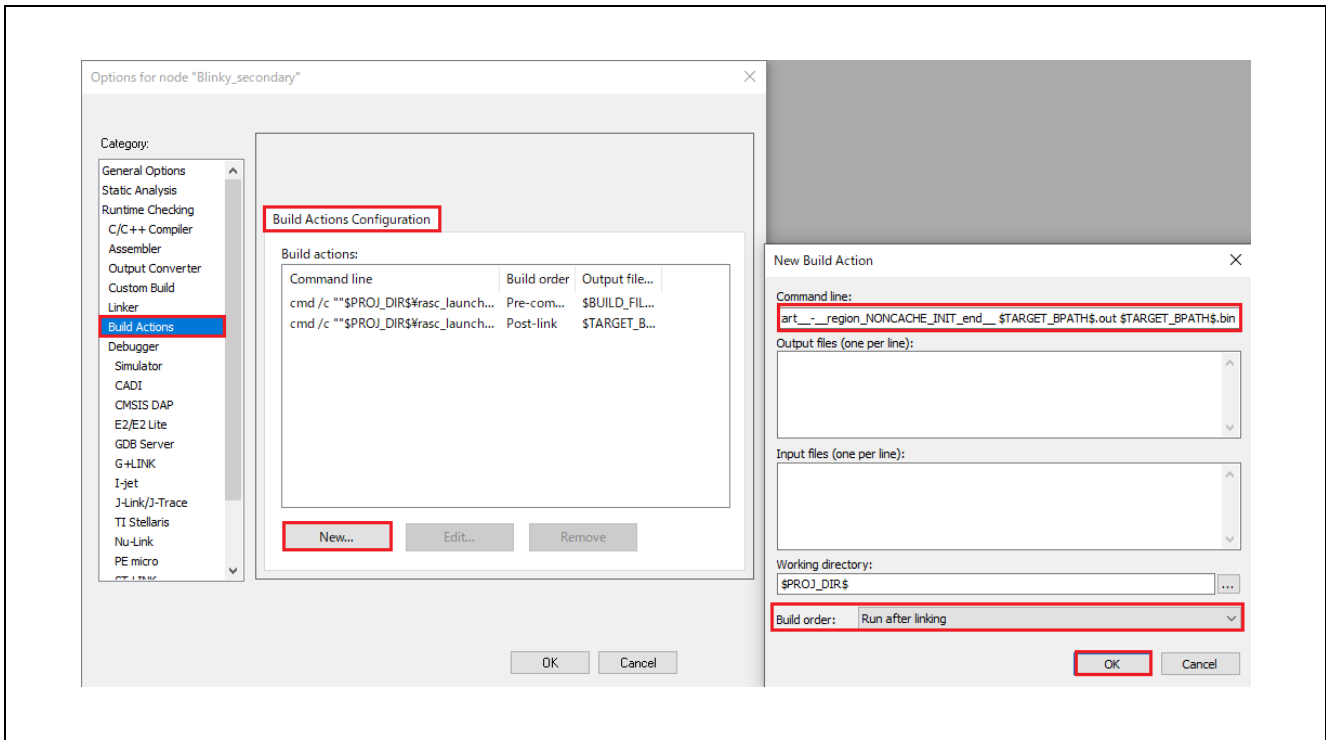


Figure 175. IAR EWARM Project Options for the Secondary Project in Flash Boot Mode

7. Build the secondary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 2.
  - (Flash boot) The following binary files are output to the Debug\Exe folder of the secondary project.

Table 31. Binary Files Output to the Debug\Exe folder of the Secondary Project

Device	Project core	Binary files	Note
RZ/T2M RZ/T2ME	CR52	xxxxx-yyyyy.bin	Binary files of programs are placed in the cache section of the System SRAM.
		xxxxx-zzzzz.bin	Binary files of programs are placed in the non-cache section of System SRAM. When using non-cache sections.
RZ/T2H RZ/N2H	CR52	xxxxx-0x0.bin	Binary files of programs are placed in ATCM.
		xxxxx-0x102000.bin	Binary files of programs are placed in BTCM.
		xxxxx-yyyyy.bin	Binary files of programs are placed in the cache section of the System SRAM.
		xxxxx-zzzzz.bin	Binary files of programs are placed in the non-cache section of System SRAM. When using non-cache sections.
	CA55	xxxxx-yyyyy.bin	Binary files of programs are placed in the cache section of the System SRAM.
		xxxxx-zzzzz.bin	Binary files of programs are placed in the non-cache section of System SRAM. When using non-cache sections.

xxxxx: the secondary project name

yyyyy: The start address of the cache section of System SRAM

zzzzz: The start address of the non-cache section of System SRAM

8. (Flash boot) Set the following additional options for the primary project.
  - i. Click **Project > Options....**

- ii. Click **Linker > Input** and set it as follows:

Table 32. Example of Set to "Keep symbols" and "Raw binary image"

Device	Primary project core	Secondary project core	Keep symbols (one per line), Symbol, Section	File	Align	Note
RZ/T2M RZ/T2ME	CR52 CPU0	CR52 CPU1	SECONDARY	\$PROJ_DIR\$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10000000.bin	8	
			SECONDARY_NONCACHE	\$PROJ_DIR\$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if file is output
RZ/T2H RZ/N2H	CR52 CPU0	CR52 CPU1	SECONDARY_ATCM_CR521	\$PROJ_DIR\$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x0.bin	8	
			SECONDARY_BTCM_CR521	\$PROJ_DIR\$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x102000.bin	8	
			SECONDARY	\$PROJ_DIR\$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10000000.bin	8	
			SECONDARY_NONCACHE	\$PROJ_DIR\$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if the file is the output
	CA55 Core0	CA55 Core1	SECONDARY	\$PROJ_DIR\$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10020000.bin	8	
		SECONDARY_NONCACHE	\$PROJ_DIR\$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if the file is the output	

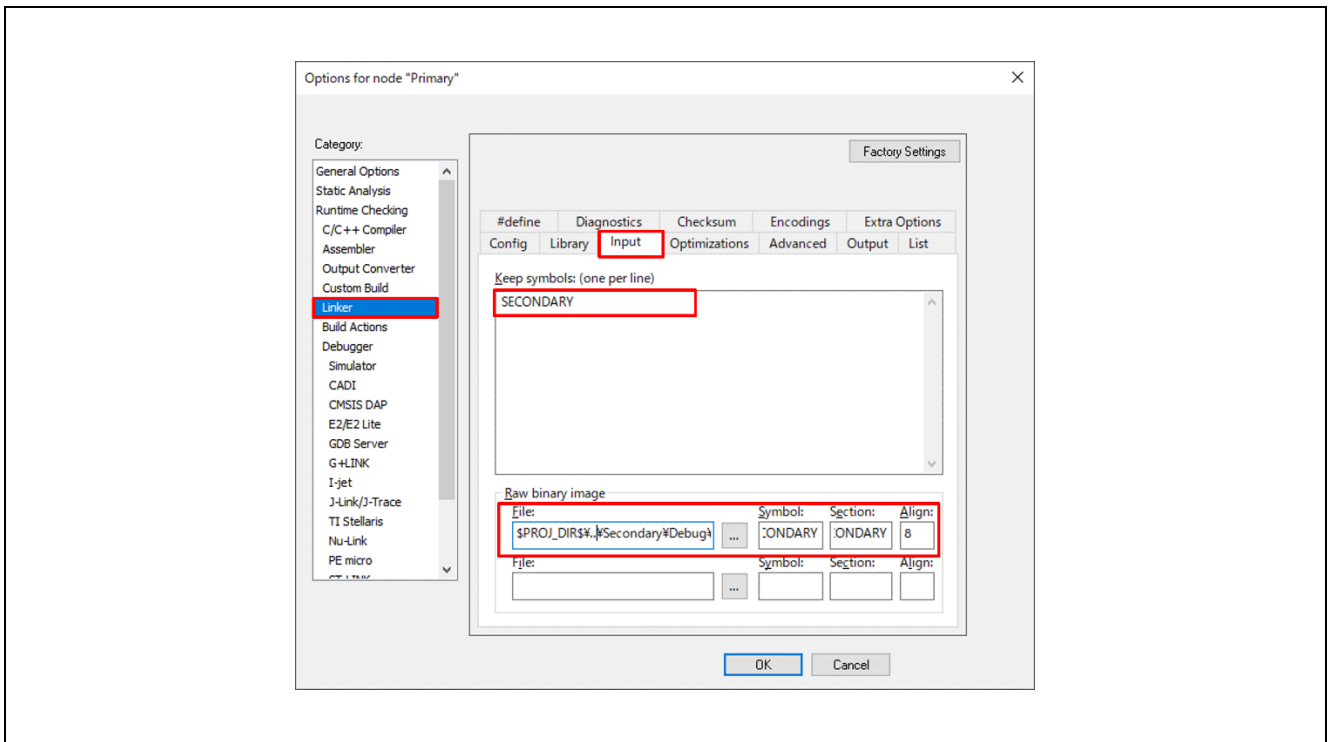


Figure 176. IAR EWARM Project Options for the Primary Project in Flash Boot Mode

**Note:**

Only two binary files can be imported with "Raw binary image". If you want to import more binary files, follow these steps.

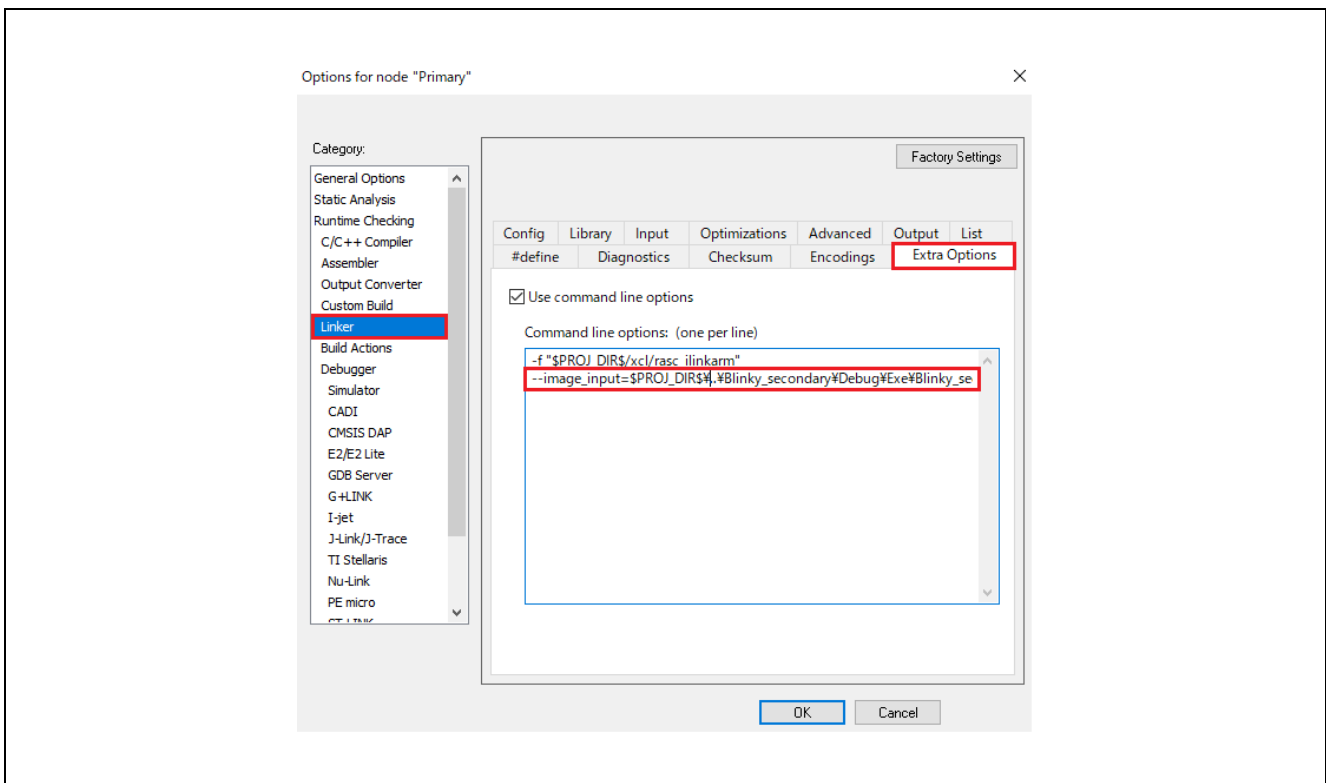
- a. Click **Linker > Extra Options** and add "--image\_input=[the secondary project path]\Debug\Exe\[the secondary project name]-[address].bin,[Symbol],[Section],[Align]" to **Command line options: (one per line)**.

e.g.

```
--image_input=$PROJ_DIR$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x0.bin,SECONDARY_ATCM_CR521,SECONDARY_ATCM_CR521,8
```

- b. Click **Linker > Input** and set it as follows:

- Keep symbols: (one per line)
  - Set the same as the **symbol** of the **Command line options** set in **Extra Options**.  
e.g. SECONDARY\_ATCM\_CR521



**Figure 177. IAR EWARM Project Options for the Primary Project in Flash Boot Mode (Extra Options)**

9. (Flash boot) Build the primary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 3.

10. (Flash boot) In the primary project, click **Project > Download > Download file...** and select the file of the primary project.  
e.g. \$PROJ\_DIR\$.\.Blinky\_primary\Debug\Exe\Blinky\_primary.out

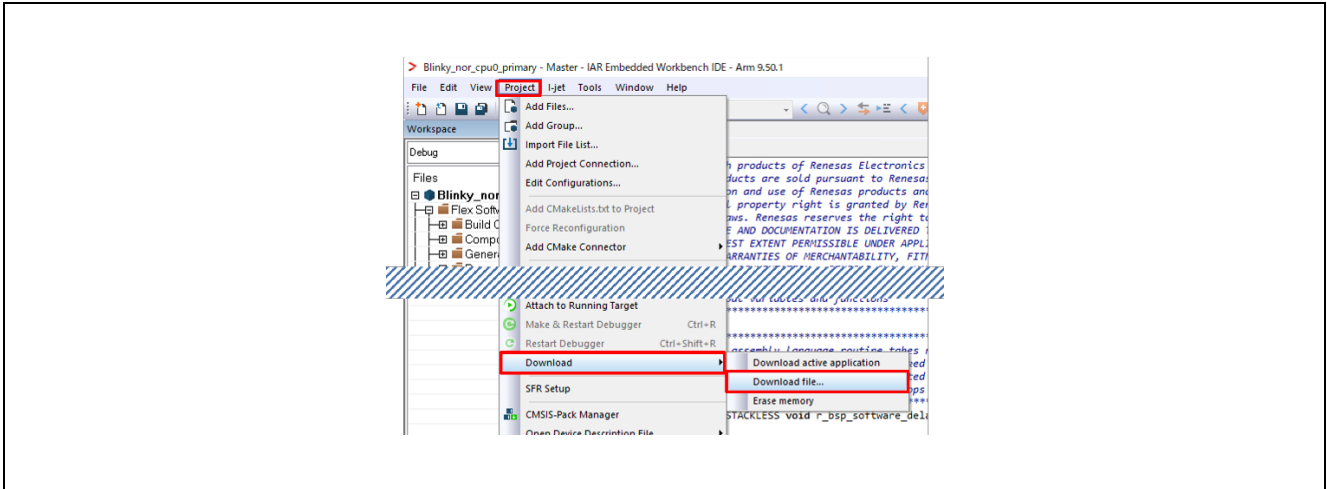


Figure 178. IAR EWARM Download File

11. Enable settings for multicore debugging in 5.3.5 Debug for Multiprocessing No. 1 and No. 2.
12. (RAM exec) Debug the projects according to the procedures in 5.3.5 Debug for Multiprocessing No. 3 and after.
13. (Flash boot) Click **Project > Debug without Downloading** on the primary project to debug. Debug the projects according to the procedures in 5.3.5 Debug for Multiprocessing No. 4 and after.

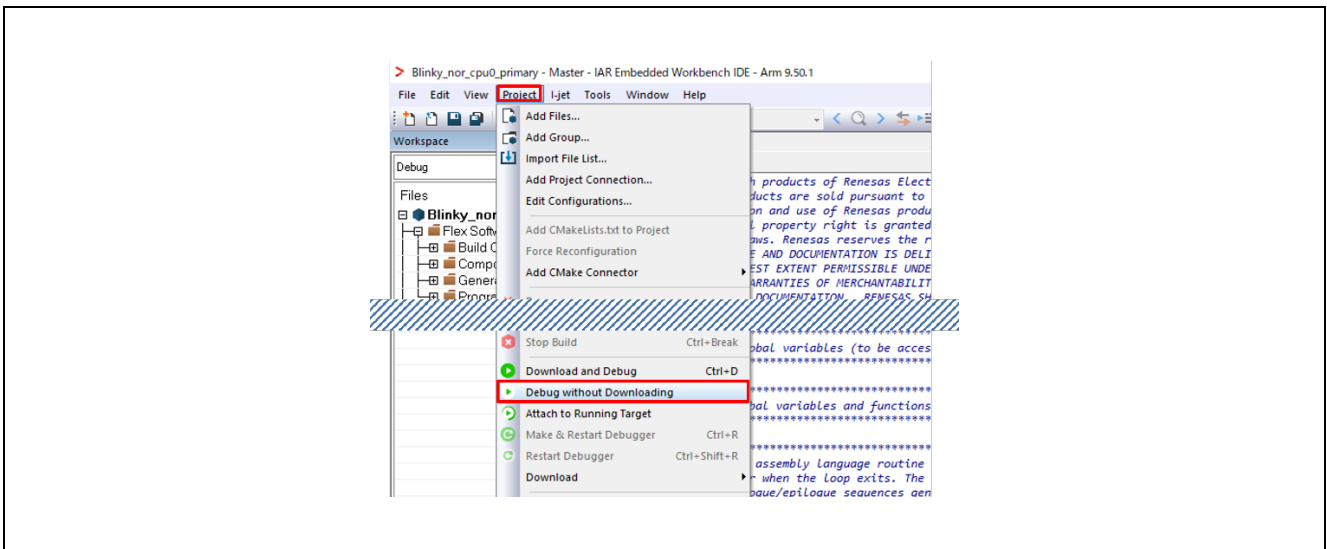


Figure 179. IAR EWARM Debug without Downloading

14. When changing the project and debugging it again, follow these steps.
  - i. (Flash boot) Disable asymmetric multicore setting.
    - a. Click **Project > Options...**
    - b. Click **Debugger > Multicore** and select **Disable** in Asymmetric multicore.
  - ii. Build the primary project (No. 4).
  - iii. Build the secondary project (No. 7).
  - iv. (Flash boot) Right-click on the primary project and select **Clean**.
  - v. (Flash boot) Build the primary project again (No. 9).
  - vi. (Flash boot) Download the file (No. 10).
  - vii. (Flash boot) Enable asymmetric multicore setting (No. 11).
  - viii. (RAM exec) Debug the projects (No. 12).
  - ix. (Flash boot) Debug the projects (No. 13).

### Multiprocessing with 3 or more cores

This section shows how to perform multi-core debugging using three cores. If more than 4 cores are used, add the process of creating a project, building it, specifying a raw binary image to build one previous project after No. 3, and add the project information in multicore\_setup.xml.

1. Create and build the primary and secondary projects according to Multiprocessing with 2 cores No. 1 to No. 7.
2. Create a tertiary project using the bundle file (.sbd) of the secondary project according to the procedures in 5.3.2 Create a New Project.
3. Build the tertiary project according to Multiprocessing with 2 cores No. 6 and No. 7. The project option settings for the tertiary project are the same as for the secondary project.
4. (Flash boot) Set the following additional options for the secondary project.
  - i. Click **Project > Options....**
  - ii. Click **Linker > Input** and set it as follows:

**Table 33 Example of Set to "Keep symbols" and "Raw binary image" for the Secondary Project**

Device	Primary project core	Secondary project core	Tertiary project core	Keep symbols (one per line), Symbol, Section	File	Align	Note
RZ/T2H RZ/N2H	CA55 Core0	<b>CR52 CPU0</b>	CR52 CPU1	SECONDARY _NONCACHE	\$PROJ_DIR\$..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x10180080.bin	8	Import only if the file is output*
	CR52 CPU0	<b>CA55 Core0</b>	CA55 Core1	SECONDARY	\$PROJ_DIR\$..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x10020000.bin	8	
				SECONDARY _NONCACHE	\$PROJ_DIR\$..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x10180080.bin	8	Import only if the file is output
	CR52 CPU0	<b>CR52 CPU1</b>	CA55 Core0	SECONDARY	\$PROJ_DIR\$..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x10000000.bin	8	
				SECONDARY _NONCACHE	\$PROJ_DIR\$..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x10180080.bin	8	Import only if the file is output

\* Binary files for TCM, such as Blinky\_secondary\_0x0.bin, will not work properly if they are imported into a project other than the primary project. Even binary files from tertiary projects must be imported into the primary project.

5. (Flash boot) Build the secondary project according to the procedures in 5.3.3.1 Build.
6. (Flash boot) Set the following additional options for the primary project according to Multiprocessing with 2 cores No. 8.
  - i. Click **Project > Options....**
  - ii. Click **Linker > Input** and set it as follows:

Table 34. Example of Set to "Keep symbols" and "Raw binary image" for the Primary Project

Device	Primary project core	Secondary project core	Tertiary project core	Keep symbols (one per line), Symbol, Section	File	Align	Note
RZ/T2H RZ/N2H	CA55 Core0	CR52 CPU0	CR52 CPU1	SECONDARY_ATCM_CR52_0	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x0.bin	8	
				SECONDARY_BTCM_CR52_0	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x102000.bin	8	
				SECONDARY_ATCM_CR52_1	\$PROJ_DIR\$..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x0.bin	8	
				SECONDARY_BTCM_CR52_1	\$PROJ_DIR\$..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x102000.bin	8	
				SECONDARY_NONCACHE	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if the file is the output
	CR52 CPU0	CA55 Core0	CA55 Core1	SECONDARY	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10000000.bin	8	
				SECONDARY_NONCACHE	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if the file is the output
	CR52 CPU0	CR52 CPU1	CA55 Core0	SECONDARY_ATCM_CR52_1	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x0.bin	8	
				SECONDARY_BTCM_CR52_1	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x102000.bin	8	
				SECONDARY	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10000000.bin	8	
				SECONDARY_NONCACHE	\$PROJ_DIR\$..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if the file is the output

7. Create `multicore_setup.xml` and store it in the primary project.

`multicore_setup.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<sessionSetup>
  <partner>
    <name>Partner0</name>
    <workspace>$WS_PATH$</workspace>
    <project>$PROJ_PATH$</project>
    <config>Debug</config>
    <numberOfCores>1</numberOfCores>
  </partner>

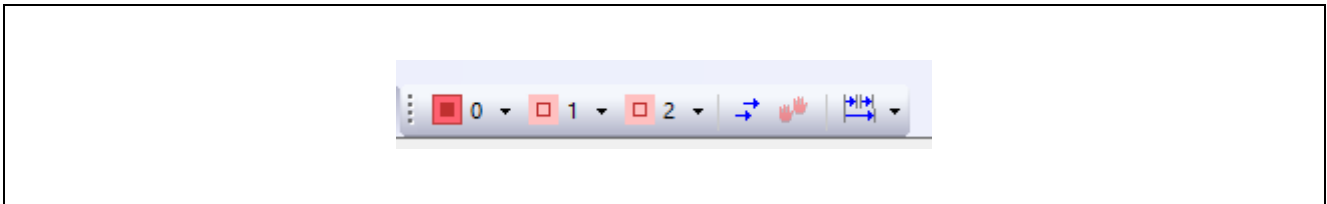
  <partner>
    <name>Partner1</name>
    <workspace>$PROJ_DIR$..\Blinky_secondary\Blinky_secondary.eww</workspace>
    <project>Blinky_secondary</project>
    <config>Debug</config>
    <numberOfCores>1</numberOfCores>
    <attachToRunningTarget>>false</attachToRunningTarget>
  </partner>

  <partner>
    <name>Partner2</name>
    <workspace>$PROJ_DIR$..\Blinky_tertiary\Blinky_tertiary.eww</workspace>
    <project>Blinky_tertiary</project>
    <config>Debug</config>
    <numberOfCores>1</numberOfCores>
    <attachToRunningTarget>>false</attachToRunningTarget>
  </partner>
</sessionSetup>
```

The project information to be debugged is described in the `<sessionSetup>` tag. The `<partner>` tag settings are as follows:

- `<name>` Arbitrary name.
  - `<workspace>` Location of workspace starting from the primary project location. The primary project only set “\$WS\_PATH\$”.
  - `<project>` Project name. The primary project only set “\$PROJ\_PATH\$”.
  - `<config>` Debug
  - `<numberOfCores>` 1
  - (Except the primary project) `<attachToRunningTarget>` false
8. (Flash boot) Build the primary project according to the procedures in 5.3.3.1 Build.
9. (Flash boot) In the primary project, click **Project > Download > Download file...** and select out file of the primary project according to Multiprocessing with 2 cores No. 10.
10. Enable settings for multicore debugging.
- i. Open the primary project and close the secondary and later projects on IAR EWARM.
  - ii. Set the following in the primary project before debugging:
    - a. Click **Project > Options....**
    - b. Click **Debugger > Multicore** and check the setting value of **Symmetric multicore**, and set the following contents in **Asymmetric multicore**.
      - Symmetric multicore
        - Number of cores: 1
      - Asymmetric multicore
        - Advanced
          - ✧ Session configuration: \$PROJ\_DIR\$\multicore\_setup.xml
11. (RAM exec) Debug the projects according to the procedures in 5.3.5 Debug for Multiprocessing No. 3 and after.

12. (Flash boot) Click **Project > Debug without Downloading** the primary project to debug. Debug the projects according to the procedures in 5.3.5 Debug for Multiprocessing No. 4 and after. As shown in Figure 180, it is according to the setting in multicore\_setup.xml. The primary project is 0, the secondary project is 1, and the tertiary project is 2.



**Figure 180. IAR EWARM Running Projects**

13. When changing the project and debugging it again, follow these steps.
  - i. (Flash boot) Disable asymmetric multicore setting.
    - a. Click **Project > Options...**
    - b. Click **Debugger > Multicore** and select **Disable** in Asymmetric multicore.
  - ii. Build the primary project (Multiprocessing with 2 cores No. 4).
  - iii. Build the secondary project (Multiprocessing with 2 cores No. 7).
  - iv. Build the tertiary project (Multiprocessing with 3 or more cores No. 3)
  - v. (Flash boot) Right-click on the secondary project and select **Clean**.
  - vi. (Flash boot) Build the secondary project again (Multiprocessing with 2 cores No. 7).
  - vii. (Flash boot) Right-click on the primary project and select **Clean**.
  - viii. (Flash boot) Build the primary project again (Multiprocessing with 2 cores No. 3).
  - ix. (Flash boot) Download the file (Multiprocessing with 2 cores No. 10).
  - x. (Flash boot) Enable asymmetric multicore setting (Multiprocessing with 3 or more cores No. 10)
  - xi. (RAM exec) Debug the projects (Multiprocessing with 3 or more cores No. 11)
  - xii. (Flash boot) Debug the projects (Multiprocessing with 3 or more cores No. 12)

**Revision History**

Rev.	Date	Description	
		Page	Summary
1.00	Jun.7.22	-	First Edition issued
1.01	Aug.9.22	-	Added the RZ/N2L device as the target device.
		All	Unified some terminologies.
		p.33	Updated “SEGGER J-Link” section. <ul style="list-style-type: none"> <li>Added the software environment on which FSP projects are verified.</li> </ul>
		p.37	Added the “3.5.1 RSK+RZN2L” section.
		p.49	Updated “” <ul style="list-style-type: none"> <li>Updated the Windows PC requirements.</li> </ul>
		p.71	Update “Prerequisites” section <ul style="list-style-type: none"> <li>Added the note regarding the patch for debugging RZ/N2L FSP project on EWARM.</li> </ul>
		p.72	Updated “Create a New Project” section. <ul style="list-style-type: none"> <li>Added installation path of FSP SC.</li> <li>Added some steps for creating an EWARM project.</li> <li>Added Note subsection for debugging RZ/N2L EWARM project.</li> </ul>
		p.99	Updated “Selecting a Board and Toolchain” section. <ul style="list-style-type: none"> <li>Added a detailed explanation of how to select the Board and Device for creating an FSP project.</li> </ul>
		p.113	Updated “Appendix Known Issues” chapter.
		p.124	Updated “Appendix Tool Software Limitations” section.
		-	Added “Appendix: How to update J-Link DLL files in e <sup>2</sup> studio” chapter.
		p.154	Added “Appendix. How to Debug the FSP Project with Flash Boot Mode”
1.02	Oct.31.22	-	Updated documentation for RZ/T2M FSP v1.1.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/T2M FSP v1.0.0</li> </ul>
		All	Updated minor issues. <ul style="list-style-type: none"> <li>Fixed minor typo.</li> <li>Adjusted page breaks.</li> </ul>
		p.33	Updated “3.3.1 SEGGER J-Link” section. <ul style="list-style-type: none"> <li>Updated the FSP version and J-Link version for RZ/T2M</li> <li>Added the notification that J-Link OB S124 requires the firmware update to debug the RZ/T2M FSP project.</li> <li>Added the link to Renesas Knowledge Base, which explains how to update J-Link DLL in e<sup>2</sup> studio.</li> </ul>
		p.45	Added “5.3.2.2 NOTE: Configure IAR EWARM Project [RZ/T2M, RZ/T2L]” section.
		p.113	Updated “Appendix Known Issues” chapter. <ul style="list-style-type: none"> <li>Remove some limitations regarding RZ/T2M</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.124	Updated “Appendix Tool Software Limitations” section. <ul style="list-style-type: none"> <li>Added new limitation of e<sup>2</sup> studio regarding J-Link OB S124 firmware version.</li> <li>Added the link to explain how to update the J-Link DLL.</li> </ul>
		-	Removed “Appendix: How to update J-Link DLL files in e <sup>2</sup> studio” chapter.
1.03	Dec.23.22	-	Updated documentation for RZ/N2L FSP v1.1.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/N2L FSP v1.0.0</li> </ul>
		All	Updated minor issues. <ul style="list-style-type: none"> <li>Fixed minor typo.</li> </ul>
		p.69	Updated “Appendix Known Issues” chapter. <ul style="list-style-type: none"> <li>Removed some limitations regarding RZ/N2L</li> </ul>
		p.73	Updated “Appendix Tool Software Limitations” section. Removed some limitations regarding RZ/N2L
		p.75	Updated “Appendix. How to Debug the FSP Project with Flash Boot Mode” chapter. <ul style="list-style-type: none"> <li>Added new procedure for RZ/N2L FSP v1.1.0.</li> </ul>
1.04	Mar.23.23	All	Updated documentation for RZ/T2 FSP v1.2.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/T2M FSP v1.1.0.</li> <li>Added contents for RZ/T2L.</li> </ul>
		p.1	Added video contents website link.
		p.9	Updated “3.3.1 SEGGER J-Link” section. <ul style="list-style-type: none"> <li>Updated the FSP version and J-Link version for RZ/T2M and RZ/T2L.</li> <li>Removed the notification that J-Link OB S124 requires the firmware update to debug RZ/T2M FSP project.</li> </ul>
		p.10-12	Updated “3.4.1 RSK+RZT2M” section. <ul style="list-style-type: none"> <li>Added explanation that other boot mode board settings refer to the RSK User’s Manual.</li> <li>Modified figure names and added a table title.</li> </ul>
		p.13-15	Added “3.4.2 RSK+RZT2L” section
		p.16-18	Updated “3.5.1 RSK+RZN2L” section. <ul style="list-style-type: none"> <li>Corrected board name.</li> <li>Added explanation that other boot mode board settings refer to the RSK User’s Manual.</li> <li>Modified figure names and added a table title.</li> </ul>
		p.23	Updated “4.3 Create a New Project for Blinky” section. <ul style="list-style-type: none"> <li>Added RSK+RZT2L Board Setting.</li> </ul>
		p.26	Updated “4.3.5 Where is main(?)” section. <ul style="list-style-type: none"> <li>Corrected product group name.</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.32	Updated “4.5.4 Change CPSR Register Value” section. <ul style="list-style-type: none"> <li>Revised description.</li> <li>Added description of how to automatically change CPSR register value.</li> </ul>
		p.33	Updated “4.6 Run the Blinky Project” section. <ul style="list-style-type: none"> <li>Removed LEDs working in CPU1 project.</li> </ul>
		p.34	Updated “5.3.1 Prerequisites” section. <ul style="list-style-type: none"> <li>Added note for RZ/T2L patch file.</li> </ul>
		p.37	Updated “5.3.2 Create a New Project” section. <ul style="list-style-type: none"> <li>Added RSK+RZT2L Board Setting.</li> </ul>
		p.50	Updated “5.3.4 Download & Debug the Project” section. <ul style="list-style-type: none"> <li>Removed LEDs working in CPU1 project.</li> </ul>
		p.69	Updated “6.5 Adding and Configuring HAL Drivers” section. <ul style="list-style-type: none"> <li>Added a table title.</li> </ul>
		p.71-73	Updated “Appendix Known Issues” section. <ul style="list-style-type: none"> <li>Added a table “List of Known Issues.”</li> <li>Numbered each issue.</li> <li>Removed the issue of adding "r_dsmif" alone</li> <li>Updated issue contents that the BSP properties are sometimes configured incorrectly</li> <li>Removed Ethernet SELECTOR issue.</li> </ul>
		p.74-77	Updated “Appendix Tool Software Limitations” section. <ul style="list-style-type: none"> <li>Added a table “List of Tool Software Limitations.”</li> <li>Numbered each limitation.</li> <li>Added new limitation of applying RZ/T2 FSP v.1.2.0 pack.</li> </ul>
		p.78	Updated “Appendix. How to Debug the FSP Project with Flash Boot Mode” section <ul style="list-style-type: none"> <li>1. (Both e<sup>2</sup> studio and EWARM) Insert the loop part in startup.c. Added e<sup>2</sup> studio 2023-01 to the table.</li> <li>3. (e<sup>2</sup> studio ONLY) Apply a macro file for RZ/N2L FSP v1.1.0 xSPI0 x1 boot mode.</li> <li>Added direct download URL of RZ/N2L patch file.</li> </ul>
1.05	Jun.30.23	All	Updated documentation for RZ/N2L FSP v1.2.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/N2L FSP v1.1.0</li> </ul>
		p.9	Updated “3.3.1 SEGGER J-Link” section. <ul style="list-style-type: none"> <li>Updated the FSP version and e<sup>2</sup> studio version for RZ/N2L.</li> </ul>
		p.30	Updated “4.5.2 Debug Steps” section. <ul style="list-style-type: none"> <li>Added description of how to automatically change CPSR register value for RZ/N2L and e<sup>2</sup> studio 2023-04.</li> </ul>
		p.33	Updated “4.5.4 NOTE: Change CPSR Register Value [RZ/T2M, RZ/T2L]” section. <ul style="list-style-type: none"> <li>Changed section title to limit the target device</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.35	Updated “5.3.1 Prerequisites” section. Removed EWARM Patch for RZ/N2L
		p.72-77	Updated “Appendix Known Issues” chapter. <ul style="list-style-type: none"> <li>Updated table “List of Known Issues” to add new issues and add N2L as the target device for No.2</li> <li>Added a new issue related to BSP configuration when changing board settings.</li> <li>Added a new issue related to the FSP module FreeRTOS issue.</li> </ul>
		p.78	Updated “Appendix Tool Software Limitations” chapter. <ul style="list-style-type: none"> <li>Removed some limitations regarding the breakpoint</li> </ul>
		p.82	Updated “Appendix. How to Debug the FSP Project with Flash Boot Mode” <ul style="list-style-type: none"> <li>Updated IDE version in the table for including e<sup>2</sup> studio 2023-04</li> </ul>
1.06	Sep.8.23	All	Updated documentation for RZ/T2 FSP v1.3.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/T2 FSP v1.2.0</li> <li>Changed GNU ARM Embedded Toolchain to version 12.2.1.arm-12-24.</li> </ul>
		p.6	Updated “1.3.2 FSP Documentation” section. <ul style="list-style-type: none"> <li>Added note for RZ/N2L FSP documentation.</li> </ul>
		p.26	Updated “4.3.6 Blinky Example Code” section. <ul style="list-style-type: none"> <li>Changed the processing of the blinky template code.</li> </ul>
		p.28	Updated “4.5.2 Debug Steps” section. <ul style="list-style-type: none"> <li>Added reset setting of debug configuration for RAM execution without flash memory.</li> </ul>
		p.43	Removed “5.3.2.2 NOTE: Configure IAR EWARM Project [RZ/T2M, RZ/T2L]” section.
		p.44	Updated “5.3.4 Download & Debug the Project” section. <ul style="list-style-type: none"> <li>Changed the processing of the blinky template code.</li> </ul>
		p.68-73	Updated “Appendix Known Issues” section. <ul style="list-style-type: none"> <li>Updated table “List of Known Issues” to add new issues.</li> <li>Added new issues related to Pins configuration.</li> <li>Added a new issue of a warning message when building “r_gmac” with the GCC compiler.</li> </ul>
		p.75	Updated “Appendix Tool Software Limitations” chapter. <ul style="list-style-type: none"> <li>Added “Smart Configurator” section.</li> <li>Added a new limitation of displaying memory region usage</li> </ul>
		p.80	Updated “Appendix. How to Debug the FSP Project with Flash Boot Mode” section. <ul style="list-style-type: none"> <li>Removed limitation related to reset when using e<sup>2</sup> studio</li> </ul>
		p.82-86	Added “Appendix: How to Erase Flash Memory” section.

Rev.	Date	Description	
		Page	Summary
1.07	Sep.29.23	All	Updated documentation for RZ/N2L FSP v1.3.0. Removed contents for RZ/N2L FSP v1.2.0
		p.9	Updated “3.3.1 SEGGER J-Link” section. Updated the FSP version and e <sup>2</sup> studio version for RZ/N2L.
		p.80	Updated “Appendix. How to Debug the FSP Project with Flash Boot Mode” section. <ul style="list-style-type: none"> <li>Change the number of items.</li> <li>Removed limitation related to RZ/N2 FSP v1.2.0 and J-link V7.80b only</li> </ul>
1.08	Jan.22.24	p.6	Corrected document numbers of RSK+RZ/T2L and RSK+RZ/N2L User's Manual
		p.68-75	Updated “Appendix Known Issues” section. <ul style="list-style-type: none"> <li>Moved the position of FSP Configurations and FSP Modules descriptions to the beginning of the chapter.</li> <li>Added column “Category” to the List</li> <li>Removed category headings (FSP Configurations, Stacks Configuration, FSP Module, BSP Configuration, ...)</li> <li>Added item “Category” to the description of each Known Issue.</li> <li>Grayed out items where issues have been resolved.</li> <li>Added description to workaround of No. 3.</li> <li>Added RZ/T2L as the target device to No. 5</li> <li>Added RZ/T2M and RZ/T2L as target device to No. 6</li> <li>Corrected instructions in the code of No. 14.</li> </ul>
		p.76-80	Updated “Appendix Tool Software Limitations” chapter. <ul style="list-style-type: none"> <li>Added column “Category” to the List</li> <li>Removed category headings (Smart Configurator, FSP Smart Configurator, e<sup>2</sup> studio, ...)</li> <li>Added item “Category” to the description of each Tool Software Limitations.</li> <li>Grayed out items where limitations have been resolved.</li> </ul>
		p.87-89	Added “Appendix: How to Change Boot Mode of FSP Project” section.
1.09	Mar.29.24	All	Updated documentation for RZ/T2 FSP v2.0.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/T2 FSP v1.3.0</li> </ul>
		p.1	List Target Device separately for each series.
		p.6	Updated “1.3 Related Documentation Files” section. <ul style="list-style-type: none"> <li>List Target Device separately for each series.</li> </ul>
		p.9	Updated “3.3.1 SEGGER J-Link” section. <ul style="list-style-type: none"> <li>List Target Device separately for each series in a table.</li> </ul>
		p.10-18	Updated “3.4 RZ/T Series Board Setup” section and added “3.5 RZ/N Series Board Setup” section. <ul style="list-style-type: none"> <li>Each series was divided into separate explanatory chapters.</li> <li>Delete unnecessary figure descriptions</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.20	Updated “4.1 Tutorial Blinky” section. Added description of a multiprocessing.
		p.21	Updated “4.3 Create a New Project for Blinky” section. Added description of a multiprocessing.
		p.26	Updated “4.3.6 Blinky Example Code” section. Updated Blinky code.
		p.27	Updated “4.4 Build the Blinky Project” section. <ul style="list-style-type: none"> <li>Added description of a multiprocessing.</li> </ul>
		p.28	Updated “4.5.2 Debug Steps” section. <ul style="list-style-type: none"> <li>Added description of a multiprocessing.</li> </ul>
		p.31	Updated “4.6 Run the Blinky Project” section. <ul style="list-style-type: none"> <li>Added LED2-3 of RSK+RZ/T2M for CPU1 core.</li> <li>Added descriptions to suspend program execution and exit debug mode.</li> </ul>
		p.32	Added “0 Debug and Run for Multiprocessing” section.
		p.33	Added “4.8 Import the Project” section.
		p.37	Updated “5.2 Tutorial Blinky” section. <ul style="list-style-type: none"> <li>Added description of a multiprocessing.</li> </ul>
		p.38	Updated “5.3.2 Create a New Project” section. <ul style="list-style-type: none"> <li>Added description of a multiprocessing.</li> </ul>
		p.43	Added “5.3.2.1 NOTE: Configure IAR EWARM Project [Only RZ/N2L]” section.
		p.43	Updated “5.3.3 Build the Project” section. <ul style="list-style-type: none"> <li>Moved text to “5.3.3.2 Build” section.</li> </ul>
		p.43	Added “5.3.3.1 NOTE: Build settings [Only Multiprocessing]” section.
		p.47	Added “5.3.3.2 Build” section.
		p.48	Updated “5.3.4 Download & Debug the Project” section. <ul style="list-style-type: none"> <li>Added description of a multiprocessing.</li> <li>Added LED2-3 of RSK+RZ/T2M for CPU1 core.</li> </ul>
		p.52	Added “5.3.5 Debug for Multiprocessing” section.
		p.53	Added “5.5 Note when debugging in different workspaces” section.
		p.72	Updated “Appendix Known Issues” chapter. <ul style="list-style-type: none"> <li>Resolved issues No. 2, No. 6, No. 8, No. 9, No. 10 and No. 11.</li> <li>Removed RZ/T2M and RZ/T2L as target device from No. 12</li> <li>Added new issue No. 13, No. 14, and No. 15.</li> </ul>
		p.89	Updated “Appendix Tool Software Limitations” chapter. <ul style="list-style-type: none"> <li>Added new limitation No. 9 and No. 10.</li> </ul>
		P.105	Added “Appendix. How to Debug FSP multiprocessing projects with Flash Boot Mode” chapter.

Rev.	Date	Description	
		Page	Summary
1.10	May.30.24	All	Updated documentation for RZ/N2L FSP v2.0.0. Removed contents for RZ/N2L FSP v1.3.0
		p.9	Updated “3.3.1 SEGGER J-Link” section. List Target Device separately for each series in a table
		p.44	Removed “5.3.2.1 NOTE: Configure IAR EWARM Project [Only RZ/N2L]” section.
		p.44-46	Updated “5.3.3.2 Build for Multiprocessing” section. <ul style="list-style-type: none"> <li>Changed the program execution start position setting.</li> <li>Corrected reset setting.</li> </ul> Added images for the settings screen.
		P.73	Updated “Appendix Known Issues” chapter. <ul style="list-style-type: none"> <li>Resolved issue No. 12</li> <li>Added RZ/N2L as target device from No.13 and No.14</li> <li>Added new issues No. 16 and No. 17.</li> </ul>
		p.84	Updated “Appendix Tool Software Limitations” chapter. <ul style="list-style-type: none"> <li>Added RZ/N2L as target device from No. 10.</li> </ul>
		p.97	Updated “Appendix How to Change Boot Mode of the FSP Project” chapter. <ul style="list-style-type: none"> <li>Added new step 4</li> </ul>
		p.99	Updated “Appendix. How to Debug FSP multiprocessing projects with Flash Boot Mode” section. <ul style="list-style-type: none"> <li>Changed the program execution start position setting.</li> <li>Modified explanation of debugging sequence.</li> </ul>
1.11	Jun.28.24	All	Updated documentation for RZ/T2 FSP v2.1.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/T2 FSP v2.0.0</li> </ul>
		All	Added the RZ/T2ME device as target device.
		p.1	Added video links of FSP Configuration.
		p.6	Updated “1.3.2 FSP Documentation” section. <ul style="list-style-type: none"> <li>Added explanation of notes when using FSP software modules.</li> </ul>
		p.10	Updated “3.4.1.1 Boot Mode” section. <ul style="list-style-type: none"> <li>Added note for board setting.</li> </ul>
		p.13	Updated “3.4.2.1 Boot Mode” section. <ul style="list-style-type: none"> <li>Added note for board setting.</li> </ul>
		p.15	Added “3.4.3 RSK+RZT2ME” section.
		p.16	Updated “3.5.1.1 Boot Mode” section. <ul style="list-style-type: none"> <li>Added note for board setting.</li> </ul>
		p.38	Updated “5.3.2 Create a New Project” section. <ul style="list-style-type: none"> <li>Added IDE Project Type setting for newer versions of FSP SC.</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.43	Updated “5.3.3.2 Build for Multiprocessing” section. <ul style="list-style-type: none"> <li>Removed Build Actions setting.</li> </ul> Added Make before debugging setting.
		p.54	Updated “6 FSP Configuration Users Guide” section. Updated figures with new tool screens.
		P.74	Updated “Appendix Known Issues” chapter. <ul style="list-style-type: none"> <li>Resolved issues No. 16 and No. 17</li> </ul> Added new issues No. 18 to No. 24.
		p.85	Updated “Appendix Tool Software Limitations” chapter. <ul style="list-style-type: none"> <li>Removed RZ/T2M and RZ/T2L as target device from No. 6.</li> </ul> Added new issue No. 11.
		p.102	Updated “Appendix How to Change Boot Mode of the FSP Project” section. <ul style="list-style-type: none"> <li>Added note on FSP version changes</li> </ul>
1.12	Nov.26.24	All	Updated documentation for RZ/T2 FSP v2.2.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/T2 FSP v2.1.0</li> </ul>
		All	Added the RZ/T2H as target device and CA55 core support.
		p.6	Updated “1.3.2 FSP Documentation” section. <ul style="list-style-type: none"> <li>Removed Note for RZ/N2L about the documentation issue.</li> </ul>
		p.8	Updated “Set up the Evaluation Board” chapter. <ul style="list-style-type: none"> <li>Changed the boards designations.</li> </ul>
		p.9	Updated “3.3.1 SEGGER J-Link” section. <ul style="list-style-type: none"> <li>Added how to update J-Link firmware.</li> </ul>
		p.16	Added “3.4.4 RZ/T2H Evaluation Board” section.
		p.24	Updated “4.3 Create a New Project for Blinky” section. <ul style="list-style-type: none"> <li>Modified to a generic description that does not specify which cores are used in multiprocessing.</li> <li>Updated tool screen images.</li> <li>Added tables describing the settings for each project.</li> </ul>
		p.31	Added “4.4.2 Build for Multiprocessing” section.
		p.32	Updated “4.5.2 Debug Steps” section. <ul style="list-style-type: none"> <li>Added tables describing the settings for each project.</li> <li>Added a note for debugging flash boot project.</li> </ul>
		p.37	Updated “Debug and Run for Multiprocessing” section. <ul style="list-style-type: none"> <li>Clarified explanation of step 2.</li> <li>Added supplemental information on behavior to step 5.</li> </ul>
		p.44	Updated “5.3.2 Create a New Project” section. <ul style="list-style-type: none"> <li>Modified to a generic description that does not specify which cores are used in multiprocessing.</li> <li>Updated tool screen images.</li> <li>Added tables describing the settings for each project.</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.51	Added "5.3.2.2 NOTE: Configure IAR EWARM Project [RZ/T2H]" section.
		p.54	Updated "5.3.3 Build the Project" section. <ul style="list-style-type: none"> <li>Swapped the order of "5.3.3.1 NOTE: Build settings [Only Multiprocessing]" and "5.3.3.2 Build" chapters.</li> <li>Added how to build for multiprocessing in "5.3.3.1 Build" section.</li> </ul> Renamed "5.3.3.2 Build for Multiprocessing" section.
		p.55	Updated "5.3.3.2 Build for Multiprocessing" section. <ul style="list-style-type: none"> <li>Added the running setting to step 1.</li> <li>Added the extra options to step 2.</li> <li>Removed the running setting and tools options from step 3.</li> </ul> Moved multicore debugging setting from step 3 to "5.3.5 Debug for Multiprocessing" section.
		p.58	Updated "5.3.4 Download & Debug the Project" section. <ul style="list-style-type: none"> <li>Added a note for debugging flash boot project.</li> </ul>
		p.62	Updated "5.3.5 Debug for Multiprocessing" section. <ul style="list-style-type: none"> <li>Added multicore debugging setting as step 2.</li> <li>Added supplemental information on behavior to step 5.</li> <li>Added how to debug when changing the project.</li> </ul>
		p.67	Updated "6.2 Create a Project" section. <ul style="list-style-type: none"> <li>Updated tool screen images and menu names.</li> </ul>
		p.71	Added "6.2.4 Duplication of Resources" section.
		p.82	Updated "Appendix Known Issues" section. <ul style="list-style-type: none"> <li>Added RZ/T2H as target device of No. 3, No. 14, No. 27 and No. 28.</li> <li>Added new issues No. 29 to No. 38.</li> <li>Resolved issues No. 4, No. 13, No. 18, No. 19 and No. 20.</li> </ul>
		p.106	Updated "Appendix Tool Software Limitations" chapter. <ul style="list-style-type: none"> <li>Added RZ/T2H as target device of No. 4, No. 7, No. 9 and No. 10.</li> <li>Added new issues No. 12 to No. 19.</li> <li>Resolved issues No. 1 and No. 11.</li> </ul>
		p.117	Updated "Appendix. How to Debug the FSP Project with Flash Boot Mode" section. <ul style="list-style-type: none"> <li>Add Note for multiprocessing projects.</li> <li>Corrected boot mode name from xSPI0 to xSPI.</li> <li>Updated path description.</li> <li>Add a column for cores to the table.</li> </ul>
		-	Removed "Appendix. How to Debug FSP multiprocessing projects with Flash Boot Mode" section.
		p.127	Added "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio" section.
		p.132	Added "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM" section.

Rev.	Date	Description	
		Page	Summary
1.13	Dec.23.24	All	Updated documentation for RZ/N2 FSP v2.1.0. Removed contents for RZ/N2 FSP v2.0.0.
		All	Added the RZ/N2H as target device.
		p.22	Added "3.5.2 RZ/N2H Evaluation Board" section.
		p.57	Renamed "NOTE: Configure IAR EWARM Project [RZ/T2H and RZ/N2H Only]" section.
		p.86	Updated "Appendix Known Issues" section. <ul style="list-style-type: none"> <li>Resolved issues No. 21, No. 22, No. 23, No. 24.</li> <li>Added RZ/N2H as target device of No. 3, No. 14, No. 27 to No. 36.</li> <li>Added RZ/N2L as target device of No. 27, No. 28.</li> <li>Added new issues No. 38 to No. 43.</li> </ul> Replaced Smart Configuration with SC.
		p.113	Updated "Appendix Tool Software Limitations" chapter. <ul style="list-style-type: none"> <li>Added RZ/N2H as target device of No. 4, No. 7, No. 9, No. 10 and No. 12 to No. 19.</li> <li>Added new issues No. 20, No. 21.</li> <li>Resolved issues No. 6.</li> <li>Updated the description of issues No. 12.</li> <li>Replaced Smart Configuration with SC.</li> </ul>
		p.125	Updated "Appendix. How to Debug the FSP Project with Flash Boot Mode" section. <ul style="list-style-type: none"> <li>Moved Note for multiprocessing projects to No. 1 in the same chapter.</li> <li>Removed CA55 NOR flash boot from the table in No. 1.</li> <li>Added No. 2 for RZ/N2H.</li> <li>Added note for debug FSP project.</li> </ul>
		p.127	Updated "Appendix How to Erase Flash Memory" <ul style="list-style-type: none"> <li>Added Device Type Name for RZ/N2H.</li> <li>Added External Address Space for RZ/N2H.</li> </ul>
1.14	Feb.28.25	All	Updated documentation for RZ/T2 FSP v2.3.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/T2 FSP v2.2.0</li> <li>Unified wording for Smart Configurator(SC) and IAR I-jet.</li> </ul>
		p.25	Updated "3.2.1 Windows PC Requirements" section. <ul style="list-style-type: none"> <li>Updated Windows PC requirements to use e<sup>2</sup> studio.</li> </ul>
		p.25	Updated "3.2.3 Choosing a Toolchain" section. <ul style="list-style-type: none"> <li>Added a table of toolchain version for each FSP.</li> </ul>
		p.26	Updated "4.3 Create a New Project for Blinky" section. <ul style="list-style-type: none"> <li>Added a table of the project creation procedure.</li> <li>Added a table of selecting a bundle file to procedure No. 9.</li> </ul>
		p.47	Updated "5.3.2 Create a New Project" section. <ul style="list-style-type: none"> <li>Added a table of the project creation procedure.</li> <li>Updated a description of selecting a bundle file in procedure No. 9.</li> <li>Added a table of selecting a bundle file to procedure No. 9.</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.55	Updated “5.3.2.2 NOTE: Configure IAR EWARM Project [RZ/T2H and RZ/N2H Only]” section. <ul style="list-style-type: none"> <li>Updated a description of error conditions.</li> </ul> Removed step 5 to set Build Actions.
		p.86	Updated “Appendix Known Issues” chapter. <ul style="list-style-type: none"> <li>Resolved issues No. 3, No. 25 to 27, No. 29, No. 31, No. 36 to 43 for RZ/T series devices.</li> </ul> Added new issue No. 44.
		p.118	Updated “Appendix Tool Software Limitations” chapter. <ul style="list-style-type: none"> <li>Resolved limitation No. 17 for RZ/T series devices.</li> </ul> Added new limitation No. 22.
		p.141	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio” chapter. <ul style="list-style-type: none"> <li>Added “Multiprocessing with 3 or more cores” section.</li> </ul>
		p.146	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM” chapter. Added “ <ul style="list-style-type: none"> <li>Multiprocessing with 3 or more cores” section.</li> </ul>
1.15	Apr.25.25	All	Updated documentation for RZ/N2 FSP v2.2.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/N2 FSP v2.1.0</li> </ul>
		p.86	Updated “Appendix Known Issues” chapter. <ul style="list-style-type: none"> <li>Resolved issues No. 3, No. 27, No. 29, No. 31, No. 37 to 43.</li> <li>Added new issue No. 45, No. 46, No. 47, No. 48.</li> </ul>
		p.118	Updated “Appendix Tool Software Limitations” chapter. <ul style="list-style-type: none"> <li>Removed RZ/N2H as target device from No. 13, No. 14.</li> <li>Resolved limitation No. 17.</li> <li>Added new limitation No. 23.</li> </ul>
		p.131	Updated “Appendix. How to Debug the FSP Project with Flash Boot Mode” section. <ul style="list-style-type: none"> <li>Updated a description of the CA55 xSPI boot from the table in No. 1</li> </ul>
		p.147	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio” chapter. <ul style="list-style-type: none"> <li>Updated “Multiprocessing with 3 or more cores” section for flash boot.</li> </ul>
1.16	Jul.9.25	All	Updated documentation for RZ/T2 FSP v3.0.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/T2 FSP v2.3.0</li> </ul>
		p.25	Updated “3.2.3 Choosing a Toolchain” section. <ul style="list-style-type: none"> <li>Added version name displayed in arm Developer website.</li> </ul>
		p.35	Updated “4.5.2 Debug Steps” section. <ul style="list-style-type: none"> <li>Added script file setting for TCM initialization.</li> </ul>
		p.60	Updated “5.3.3.2 Build for Multiprocessing” section. <ul style="list-style-type: none"> <li>Added macro file setting for TCM initialization.</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.88	Updated "Appendix Known Issues" chapter. <ul style="list-style-type: none"> <li>Resolved issues No. 14, No. 32, No. 46, No. 47 for RZ/T series devices.</li> </ul> Resolved issues No.45, No. 48.
		p.122	Updated "Appendix Tool Software Limitations" chapter. Resolved limitations No. 13, No. 14.
		p.135	Updated "Appendix. How to Debug the FSP Project with Flash Boot Mode" chapter. <ul style="list-style-type: none"> <li>Removed column e<sup>2</sup> studio 2022-04 2024-07.</li> <li>Updated black text code in system_init().</li> </ul> Changed the number of loops for waiting to 1.5x.
		p.145	Updated "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio" chapter. Added procedure for RZ/T2 FSP v3.0.0 to meet that specification.
		p.158	Updated "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM" chapter. <ul style="list-style-type: none"> <li>Added procedure for RZ/T2 FSP v3.0.0 to meet that specification.</li> </ul>
1.17	Aug.29.25	All	Updated documentation for RZ/N2 FSP v3.0.0. <ul style="list-style-type: none"> <li>Removed contents for RZ/N2 FSP v2.2.0.</li> </ul>
		p.35	Updated "4.5.2 Debug Steps" section. <ul style="list-style-type: none"> <li>Added the RZ/N2H as target device.</li> </ul>
		p.60	Updated "5.3.3.2 Build for Multiprocessing" section. <ul style="list-style-type: none"> <li>Added the RZ/N2H as target device in step 2.</li> </ul>
		p.88	Updated "Appendix Known Issues" chapter. <ul style="list-style-type: none"> <li>Resolved issues No. 14, No. 32, No. 46, No. 47.</li> <li>Added new issue No. 49.</li> </ul>
		p.122	Updated "Appendix Tool Software Limitations" chapter. <ul style="list-style-type: none"> <li>Resolved limitations No. 20.</li> </ul>
		p.135	Updated "Appendix. How to Debug the FSP Project with Flash Boot Mode" chapter. <ul style="list-style-type: none"> <li>Removed Note for multiprocessing projects in No. 1.</li> </ul>
		p.143	Updated "Appendix How to Change Boot Mode of the FSP Project" chapter. <ul style="list-style-type: none"> <li>Inserted the description in step 3.</li> </ul>
		p.145	Updated "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio" chapter. <ul style="list-style-type: none"> <li>Removed "For RZ/N2 FSP v2.2.0".</li> <li>Updated Multiprocessing with 2 cores for RZ/N2H as target device.</li> <li>Updated Multiprocessing with 3 or more cores for RZ/N2H as target device.</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.153	Updated "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM" chapter. <ul style="list-style-type: none"> <li>Removed "For RZ/N2 FSP v2.2.0"</li> <li>Updated Multiprocessing with 2 cores for RZ/N2H as target device.</li> </ul> Updated Multiprocessing with 3 or more cores for RZ/N2H as target device.
1.18	Oct.31.25	All	Updated documentation for RZ/T2 FSP v3.1.0. Removed contents for RZ/T2 FSP v3.0.0.
		p.35	Update "4.5.2 Debug Steps" section. Added Prevent Releasing the Reset of the CM3 Core setting for CR52 CPU1 to step 2.
		p.87	Updated "Appendix Known Issues" chapter. Resolved issue No. 49 for RZ/T series devices.
		p.121	Updated "Appendix Tool Software Limitations" chapter. Resolved limitations No. 16 and No. 23 for RZ/T series devices.
		p.134	Updated "Appendix. How to Debug the FSP Project with Flash Boot Mode" chapter. <ul style="list-style-type: none"> <li>Add the explanation of eSD and eMMC boot modes.</li> <li>Add Note for multiprocessing projects in No. 1.</li> <li>Corrected the user program stop time.</li> <li>The function to add the loop part was changed to R_BSP_WarmStart_StackLess() for RZ/T2 FSP v3.1.0.</li> </ul>
		p.145	Updated "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio" chapter. <ul style="list-style-type: none"> <li>Added Prevent Releasing the Reset of the CM3 Core setting for CR52 CPU1 to No. 10 of Multiprocessing with 2 cores.</li> <li>Added project clean to the procedure for modifying and re-running existing projects to No. 11 of Multiprocessing with 2 cores and Multiprocessing with 3 cores or more cores.</li> <li>Added setting properties for RZT2 CR52 secondary project to No. 5 of Multiprocessing with 2 cores and No. 3 of Multiprocessing with 3 cores or more cores.</li> </ul>
		p.154	Updated "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM" chapter. <ul style="list-style-type: none"> <li>Added project clean to the procedure for modifying and re-running existing projects to No. 13 of Multiprocessing with 2 cores and Multiprocessing with 3 cores or more cores.</li> </ul>
1.19	Nov.28.25	All	Updated documentation for RZ/N2 FSP v3.1.0. Removed contents for RZ/N2 FSP v3.0.0.
		p.87	Updated "Appendix Known Issues" chapter. <ul style="list-style-type: none"> <li>Resolved issue No. 49.</li> </ul>
		p.121	Updated "Appendix Tool Software Limitations" chapter. <ul style="list-style-type: none"> <li>Resolved limitations No. 16, No. 23.</li> <li>Added new issue No. 24.</li> </ul>

Rev.	Date	Description	
		Page	Summary
		p.134	Updated "Appendix. How to Debug the FSP Project with Flash Boot Mode" chapter. <ul style="list-style-type: none"> <li>Removed "For RZ/N2 FSP v3.0.0".</li> </ul> The function to add the loop part was changed to R_BSP_WarmStart_StackLess() for RZ/N2 as target device.
		p.145	Updated "Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e <sup>2</sup> studio" chapter. Updated setting properties for RZN2 CR52 secondary project to No. 5 of Multiprocessing with 2 cores and No. 3 of Multiprocessing with 3 cores or more cores.
4.00	Feb.27.26	p.9	Updated SEGGER J-Link version.
		p.25	Updated Tool chain and FSP versions.
		p.40	Added debug configuration for CA55 secondary project to section 4.7 "Debug and Run for Multiprocessing".
		p.56	<ul style="list-style-type: none"> <li>Added new section "5.3.2.1 NOTE: Configure IAR EWARM Project"</li> </ul>
		p.90	Updated "Appendix Known Issues" chapter. <ul style="list-style-type: none"> <li>Added new issue No. 50.</li> </ul>
		p.126	Updated "Appendix Tool Software Limitations" chapter. Added new issues No. 25, 26, 27, 28.
		p.159	Set "Primary project" to "-" at "Reset at the beginning of connection" in case of "Other cores" at table 27.
4.10	May.20.26	p.8	Added e <sup>2</sup> studio installation steps into section 2.1.
		p.25	Added new section "2.2 FSP Setup".
		p.34	Swap chapter number between "Set up the Evaluation Board" and "Starting Development Introduction".
		p.62	Corrected default value of "Prevent Releasing the Reset of the CM3 Core" setting as Yes at step 3.
		p.120 p.151 p.152	Updated "Appendix Known Issues" chapter. Added new issues No. 51, 52.
		p.154	Updated "Appendix Tool Software Limitations" chapter. Resolved issue No. 24.

## 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](http://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/](http://www.renesas.com/contact/).