RZ/T, RZ/N Series

Getting Started with RZ Flexible Software Package

Introduction

This application note describes how to use the Renesas Flexible Software Package (FSP) for writing applications for the RZ/T, RZ/N microprocessor series.

Target Device

RZ/T series: RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/T2N
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 e2studio
    (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 e²studio 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 e²studio setup 8
2.1.1 What is e²studio 8
2.1.2 e²studio Prerequisites 8
2.1.3 e²studio installation for Windows PC 8
2.1.4 e²studio 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 a 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 Emulators 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
3.4.5 RZ/T2N Evaluation Board 43
3.5 RZ/N Series Board Setup 46
3.5.1 RSK+RZN2L 46
3.5.2 RZ/N2H Evaluation Board 49

4. Tutorial: Your First RZ MPU Project – Blinky 52
4.1 Tutorial Blinky 52

4.2 What Does Blinky Do? ... 52
4.3 Create a New Project for Blinky ... 52
4.3.1 Details about the Blinky Configuration ... 60
4.3.2 Configuring the Blinky Clocks ... 60
4.3.3 Configuring the Blinky Pins ... 60
4.3.4 Configuring the Parameters for Blinky Components ... 60
4.3.5 Where is main()? ... 60
4.3.6 Blinky Example Code ... 60
4.4 Build the Blinky Project ... 61
4.5 Debug the Blinky Project ... 62
4.5.1 Debug Prerequisites ... 62
4.5.2 Debug Steps ... 62
4.5.3 Details about the Debug Process ... 65
4.6 Run the Blinky Project ... 65
4.7 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 ... 86
5.3.4 Download & Debug the Project ... 87
5.4 Re-configuring Project with FSP SC ... 90
5.4.1 Launch FSP SC from IAR EWARM ... 90
5.4.2 Launch from the Command Prompt ... 91
5.5 Note when debugging in different workspaces ... 91

6. FSP Configuration Users Guide ... 92
6.1 What is a Project? ... 92
6.2 Create a Project ... 94
6.2.1 Creating a New Project ... 94
6.2.2 Selecting a Board and Toolchain ... 95
6.2.3 Selecting a Project Template ... 96
6.2.4 Duplication of Resources ... 98
6.3 Configuring a Project ... 99
6.3.1 Summary Tab ... 99
6.3.2 Configuring the BSP ... 100
6.3.3 Configuring Clocks ... 101
6.3.4 Configuring Pins ... 102
6.4 Configuring Interrupts from the Stacks Tab ... 105

6.4.1 Creating Interrupts from the Interrupts Tab 105
6.4.2 Viewing Event Links 106
6.5 Adding and Configuring HAL Drivers 107
6.6 Reviewing and Adding Components 108

7. Appendix 109
7.1 Known Issues 109

8. Appendix 109
8.1 How to Erase Flash Memory 109
8.2 How to Program an ELF File to Flash Memory 115

9. Appendix 116
9.1 How to Change Boot Mode of the FSP Project 116

10. Appendix. How to Create and Debug FSP Projects for Multiprocessing for e^{2}studio 119
10.1 Debug Primary CR52 Core0 or CA55 Core0 (All Devices) 119
10.2 Standalone Primary CR52 Core0 (RZ/T2M or RZ/T2ME) 121
10.3 Standalone Primary CR52 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H) 123
10.4 Standalone Primary CA55 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H) 126

11. Appendix. How to Create and Debug FSP Projects for Multiprocessing for IAR EWARM 130
11.1 Debug Primary CR52 Core0 or CA55 Core0 (All Devices) 131
11.2 Standalone Primary CR52 Core0 (RZ/T2M or RZ/T2ME) 136
11.3 Standalone Primary CR52 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H) 138
11.4 Standalone Primary CA55 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H) 142

12. Appendix. How to Create a CMake Project using e^{2}studio 147
12.1 Create a New Project 147
12.2 Link Object Files 147
12.3 Debug the Project 148

13. Appendix. How to Create a CMake Project using FSP SC 149
13.1 Create a New Project 149
13.2 Setup Visual studio Code 149
13.3 Build the Project 150
13.4 Link Binary Files 152
13.5 Debug the Project 152

Revision History 153

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/T and RZ/N MPU devices.

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 e²studio IDE

FSP provides a host of efficiency-enhancing tools for developing projects targeting the Renesas RZ/T, RZ/N series of MPU devices. The e²studio 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 third-party IDE and toolchain.

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

1.2.4 FSP Documentation

The related file, “FSP Documentation” contains HTML documentation describing the features, APIs, and usage notes regarding the BSP and HAL drivers implemented as FSP modules and interfaces. After clicking “**index.html**” in “**FSP Documentation**” to open the introduction page in 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/T2N Group RZ/T2N Evaluation Board User's Manual
    - Document No. **R20UT5574**
- 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 the Renesas website by entering their **Document Number** into a search box.

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

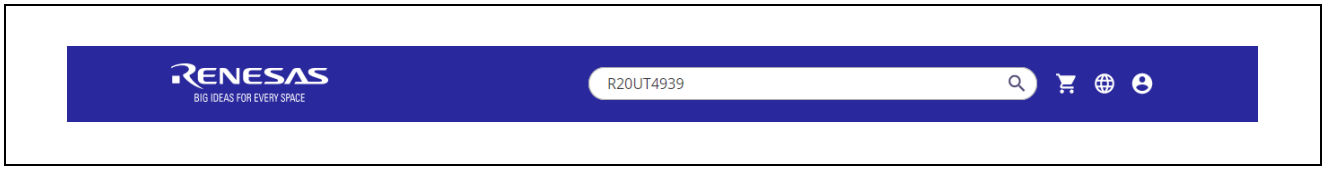


Figure 1. Search Box in Renesas Web Page

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, RZ/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 using e²studio or FSP SC (for IAR EWARM). This Getting Started guide includes tutorials for both tools, and the chapters you should read depend on the tool you use.

e²studio 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 the e²studio 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 e²studio.
- **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 based on e²studio, but most of it also applies to FSP SC.

2. Starting Development Introduction

2.1 e²studio setup

2.1.1 What is e²studio

Renesas e²studio is a development tool encompassing code development, build, and debug. e²studio is based on the open-source Eclipse IDE and the associated C/C++ Development Tooling (CDT).

When developing for RZ MPUs, e²studio 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 e²studio 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
  - RZ/T2N Evaluation Board Kit Quick Start Guide

2.1.2.2 PC Requirements

The following are the minimum PC requirements to use e²studio:

- Windows 11 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 e²studio installation for Windows PC

This chapter describes how to install the e²studio IDE on a Windows PC.

2.1.3.1 Download

The latest e²studio IDE installer package can be downloaded from the Renesas website for free. Check the detailed information from: https://www.renesas.com/e2studio. Note that the user has to log in to the Renesas account (on the MyRenesas page) for the software download.

2.1.3.2 Installation of e²studio IDE

1. Double-click the e²studio installer to open the e²studio installation wizard. First, select the install type. In this material, Custom Install is expected to be selected. Then, click [Next >] to continue.

Note: If e²studio is already installed on your PC, the option to modify, remove the existing version, or install the e²studio will be displayed in a different location.

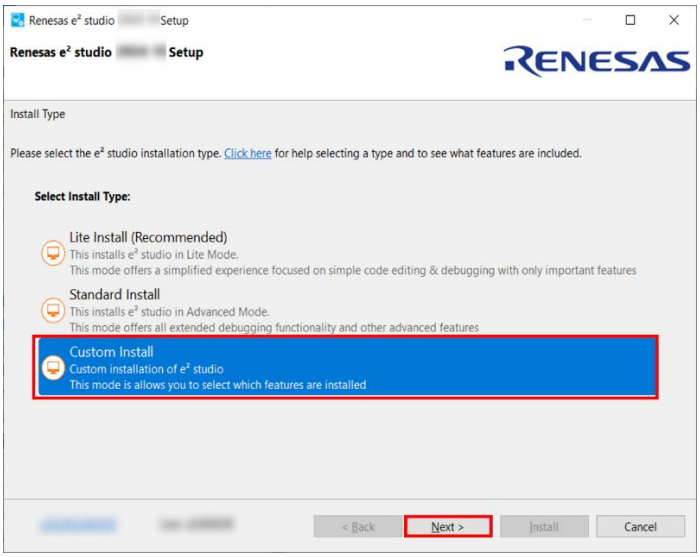


Figure 2. e²studio installation wizard

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.

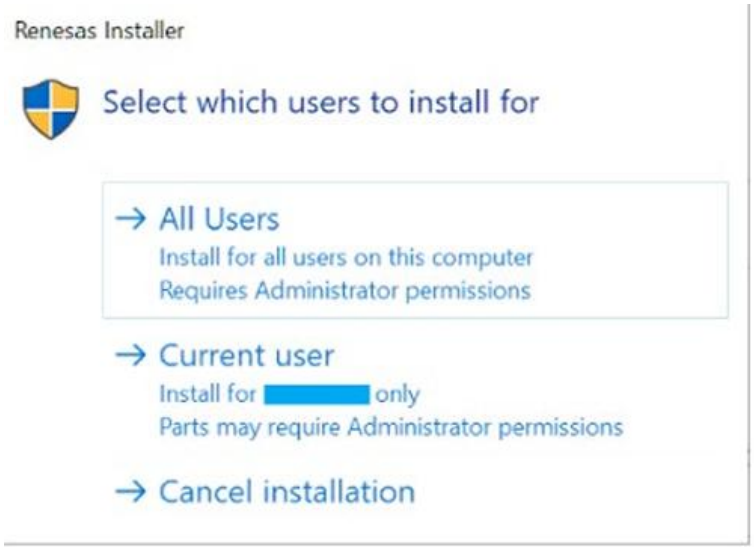


Figure 3. Select User for Installation

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

Notes:
1. If you would like to have multiple versions of e²studio, specify a new folder here.
2. Multi-byte characters cannot be used for the e²studio installation folder name.

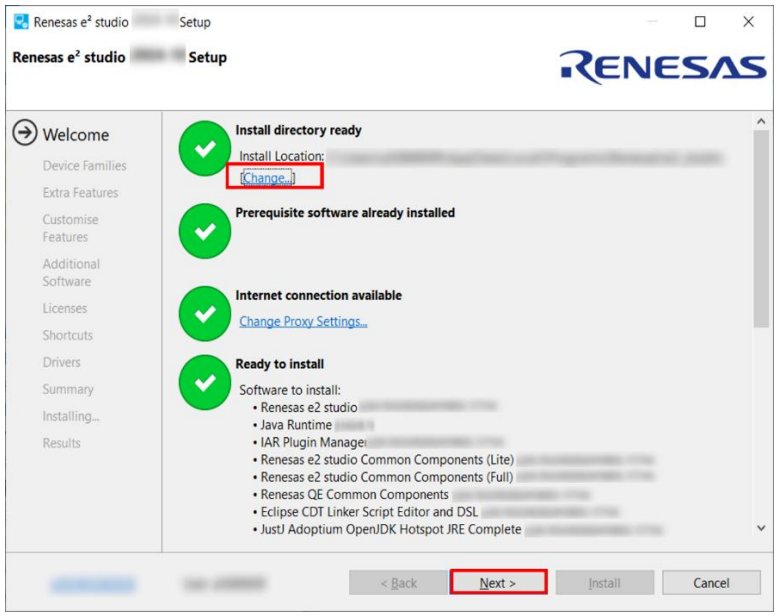


Figure 4. Installation of e²studio – Welcome page

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

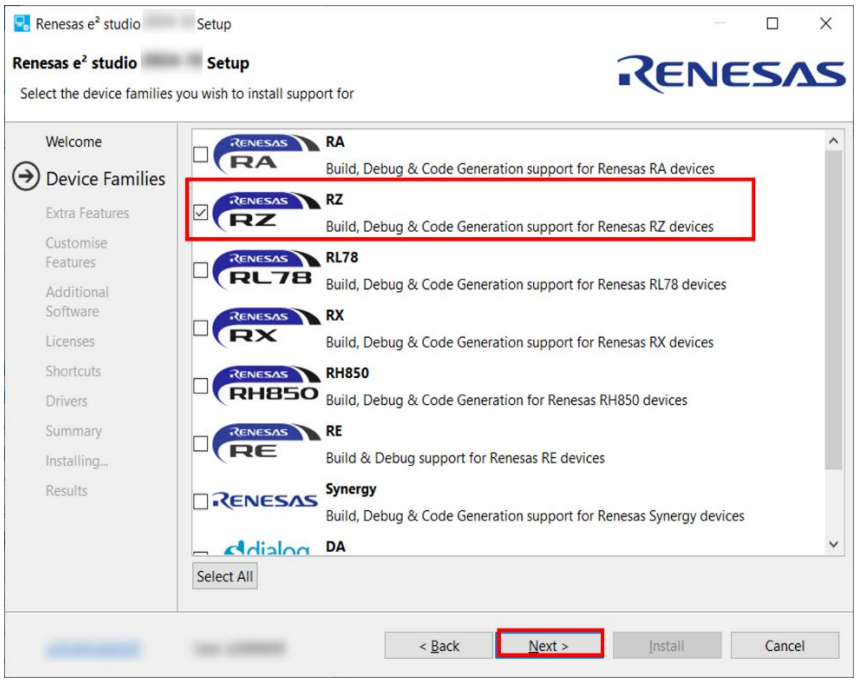


Figure 5. Installation of e²studio – Device Families

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

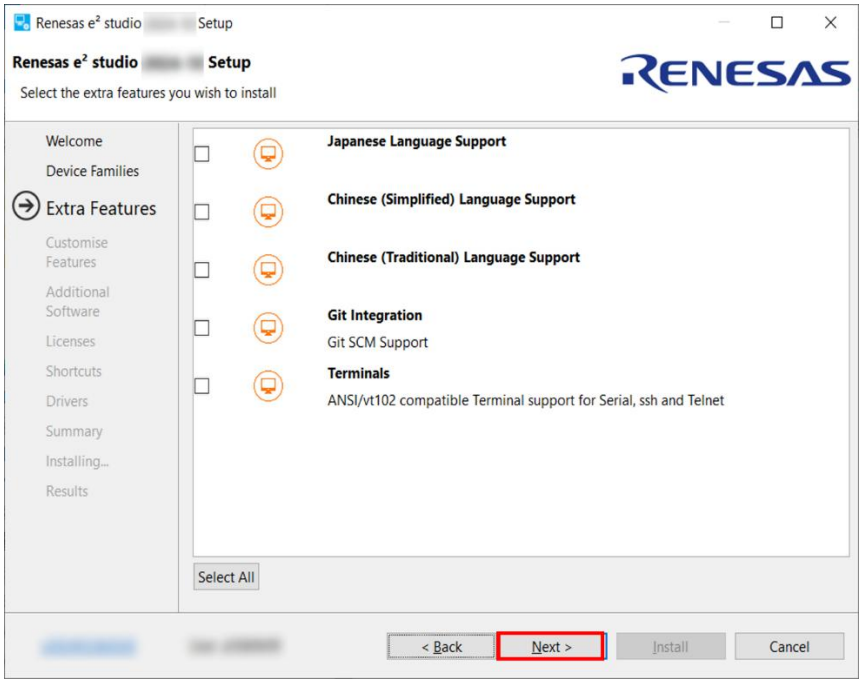


Figure 6. Installation of e²studio – Extra Features

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.

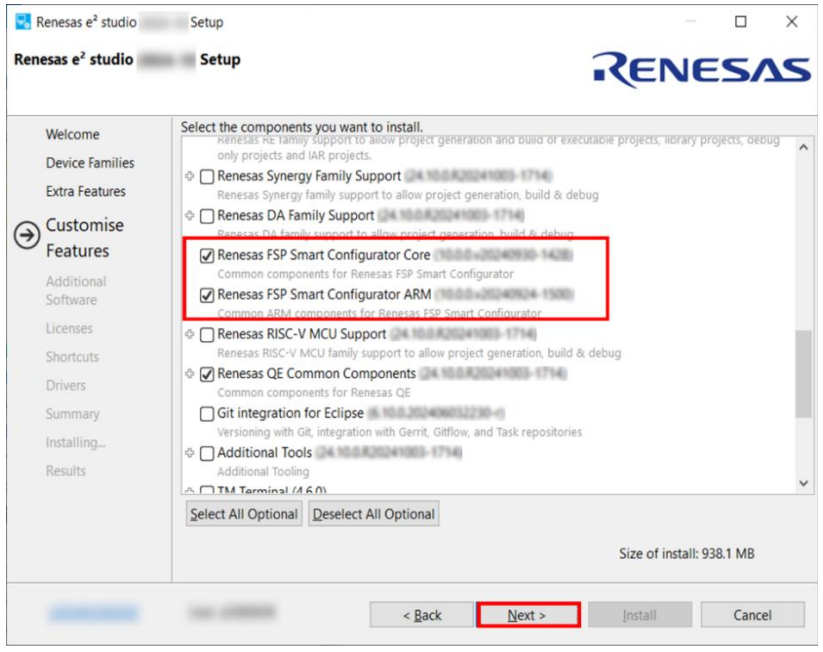


Figure 7. Installation of e²studio – Features

6. Additional Software
   Select additional software (i.e., compilers, utilities, QE…) to be installed. Be sure to select the following items 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)

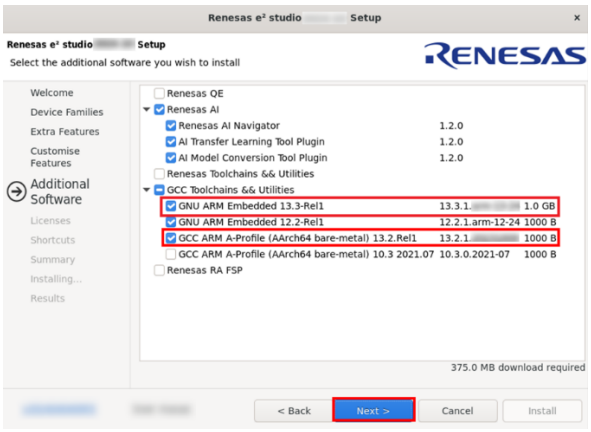


Figure 8. Installation of e²studio – Additional Software

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

7. License Agreement
   Read and accept the software license agreement. Click the [Next] button. Note that the user must accept the license agreement; otherwise, installation cannot continue.

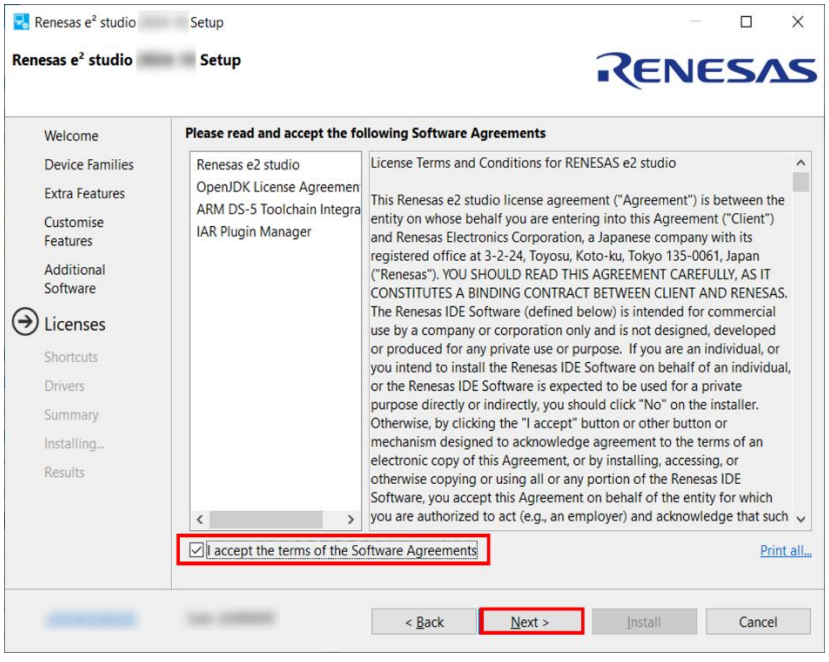


Figure 9. Installation of e²studio – Licenses

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

Note: If e²studio was installed in another location, it is recommended to rename it to distinguish it from the other e²studio(s).

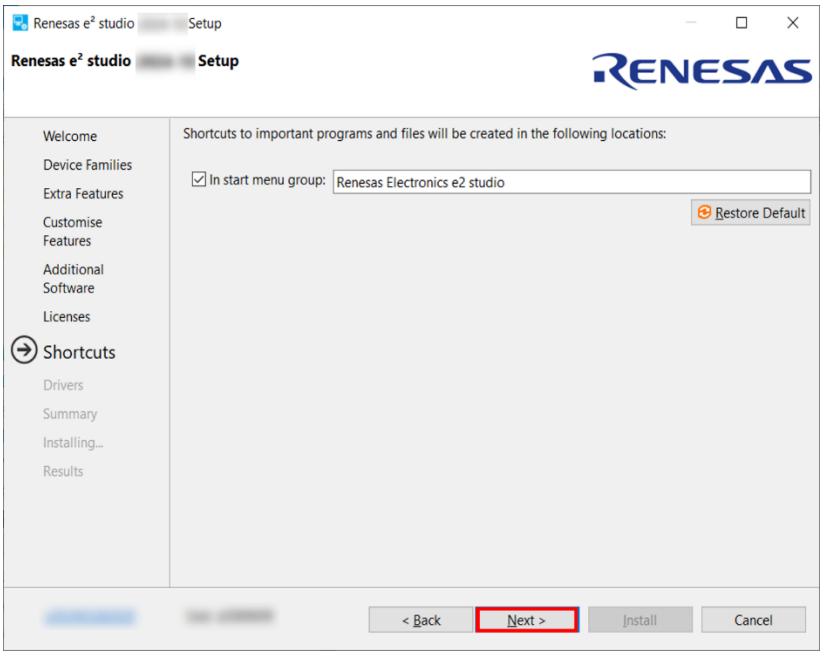


Figure 10. Installation of e²studio – Shortcuts

9. Summary
   The components to be installed are shown. Confirm the contents and click the [Install] button to install the Renesas e²studio IDE.

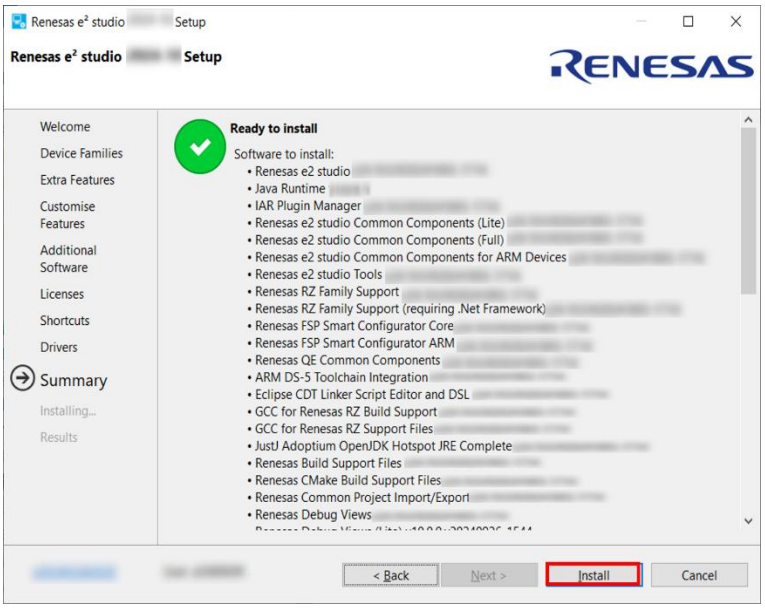


Figure 11. Installation of e²studio – Summary

10. Installing
    The installation is being completed. Depending on the selected additional software, new dialog prompts may appear during the installation process. See the section *2.1.3.3 Installation of Additional Software* for more detailed information.

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

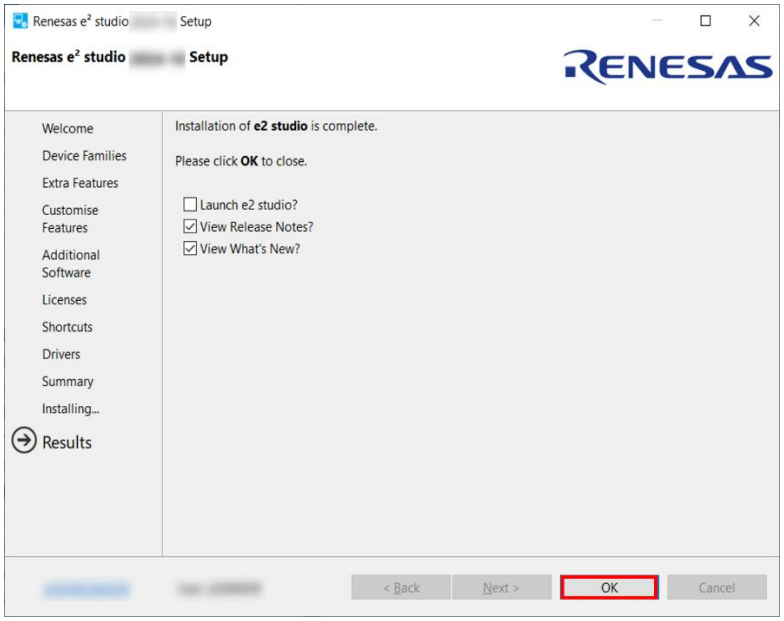


Figure 12. Summary Page

2.1.3.3 Installation of Additional Software

As mentioned in section *2.1.3.2 Installation of e²studio IDE*, 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.

This section describes the detailed procedure for installing these tools.

- **GNU ARM Embedded Toolchain 13.3-Rel1**
  If it was selected in the Additional Software pane of e²studio, you will see the installation wizard for the GNU ARM Embedded Toolchain during installation.

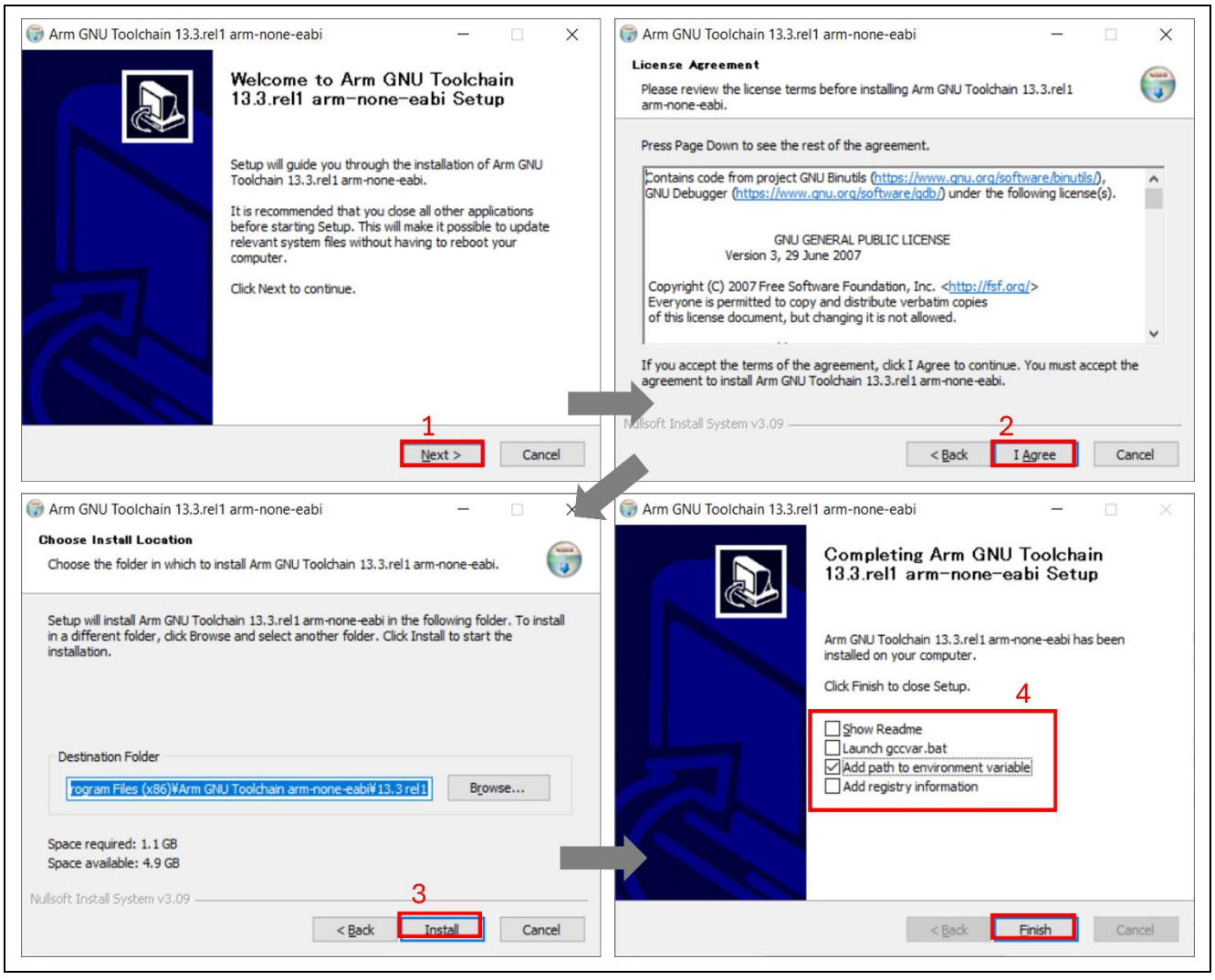


Figure 13. Installation of GNU ARM Embedded Toolchain

- **GNU ARM A-Profile (AArch64 bare-metal) 13.2.Rel1.**
  If it was selected in the Additional Software pane of e²studio, you will see the installation wizard for the GNU ARM A-Profile Toolchain during the installation process.

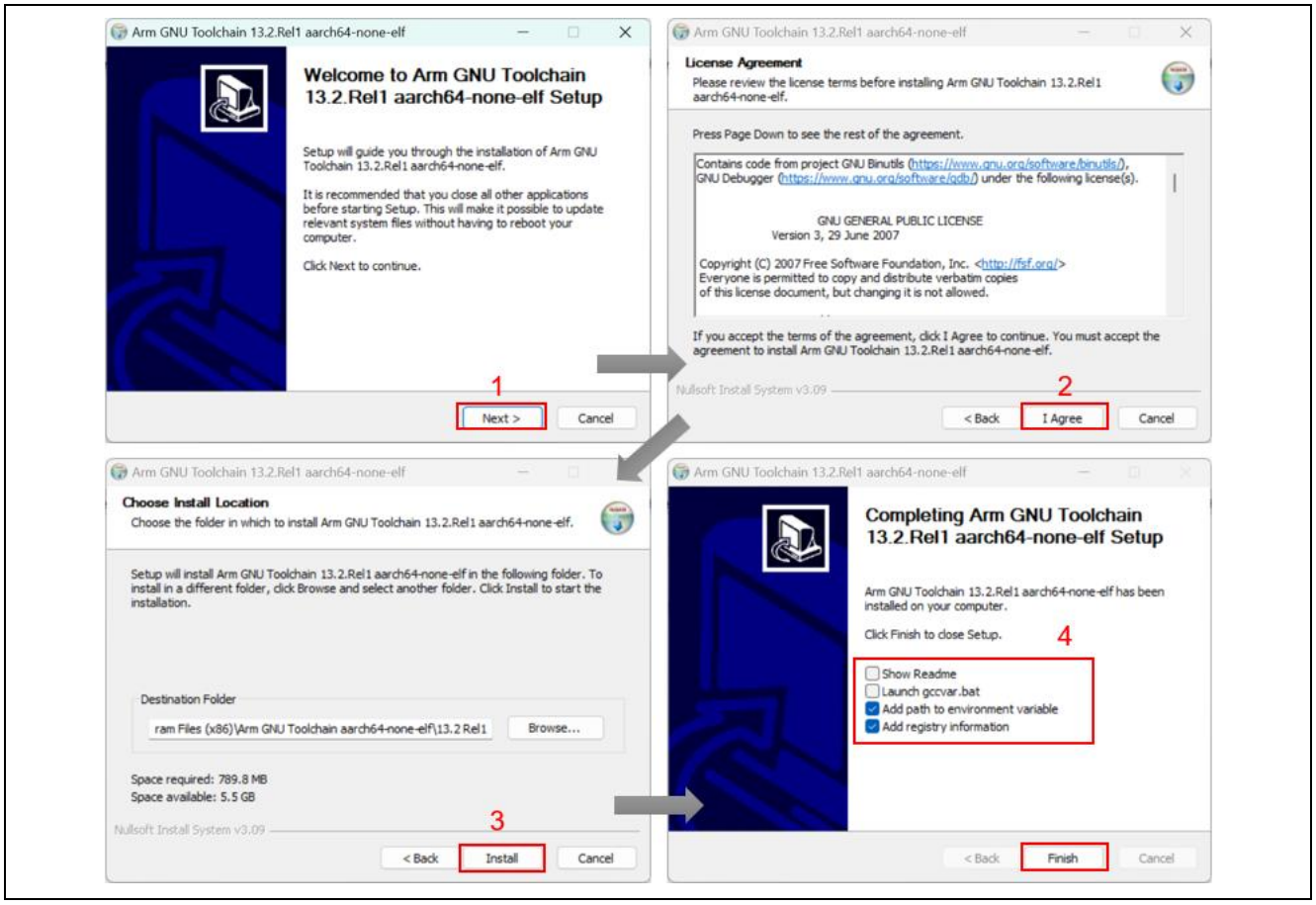


Figure 14. Installation of GNU ARM A-Profile Toolchain

2.1.4 e²studio installation for Linux PC

This chapter describes how to install the e²studio IDE on a Linux PC.

2.1.4.1 Download

The following files must be downloaded before installation.

SEGGER J-Link driver
Download the driver V9.70 or later from:
https://www.segger.com/downloads/jlink

e²studio IDE installer
e²studio IDE installer package can be downloaded from the Renesas website for free. 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_V970_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_V970_x86_64.deb

- **e²studio IDE**
  1. Run the e²studio IDE Installer. (Before running the installer, check the installer's execution permissions.)
     ./e2studio_installer-yyyy-mm_linux_host.run
  2. **Welcome page**
     The user needs to select the install type as shown below. In this material, Custom Install is expected to be selected. Then, click [Next >] to continue.

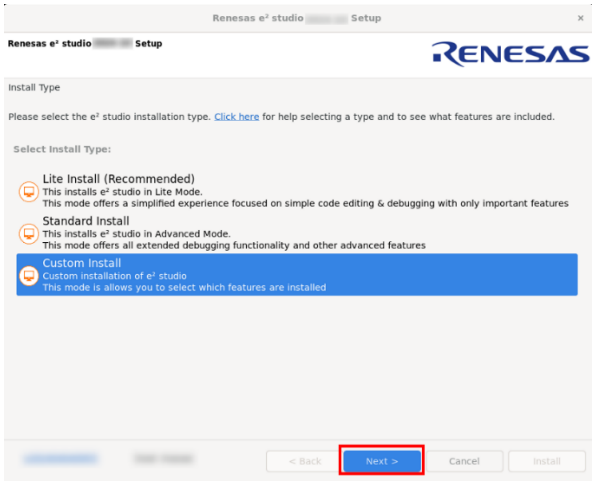


Figure 17. Installation of e²studio – Welcome page

3. **Welcome page (Cont'd)**
   The user can change the installation folder by clicking [Change…].
   Click [Next] to continue.

Notes:
1. If you would like to have multiple versions of e²studio, specify a new folder here.
2. Multi-byte characters cannot be used for the e²studio installation folder name.

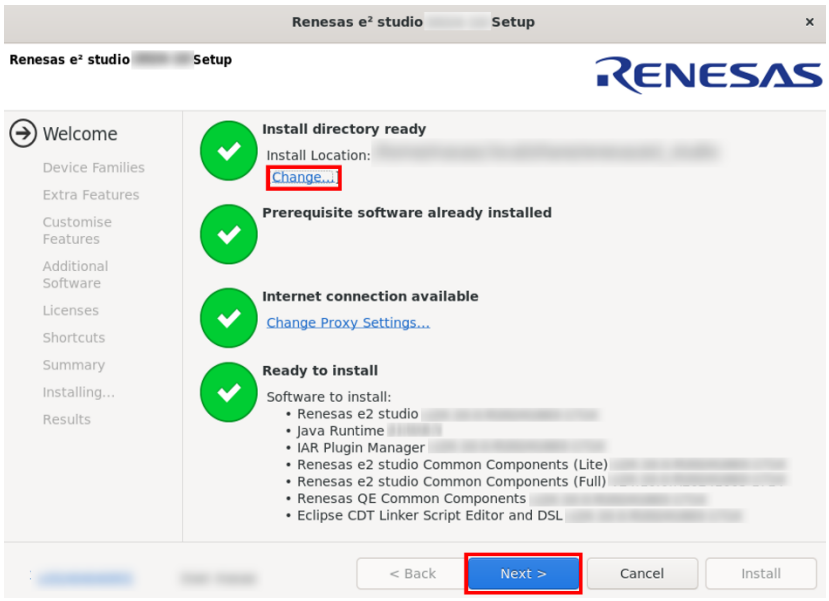


Figure 18. Installation of e²studio – Welcome page

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

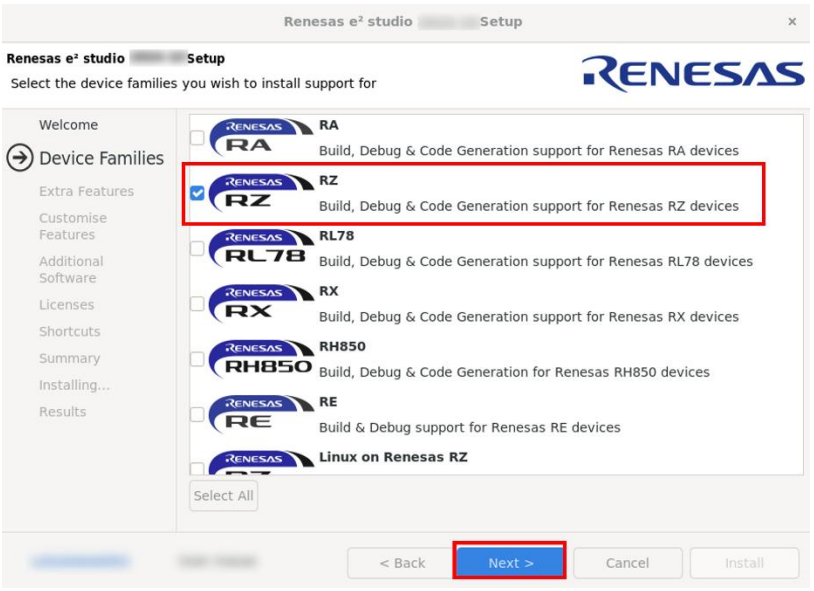


Figure 19. Installation of e²studio – Device Families

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

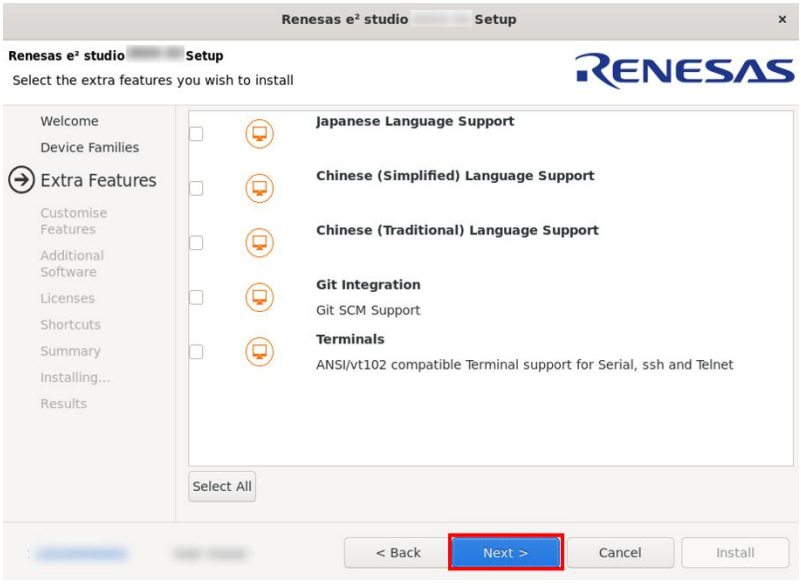


Figure 20. Installation of e²studio – Extra Features

6. **Customize Features**
   Select the components to install and click the [Next] button to continue.

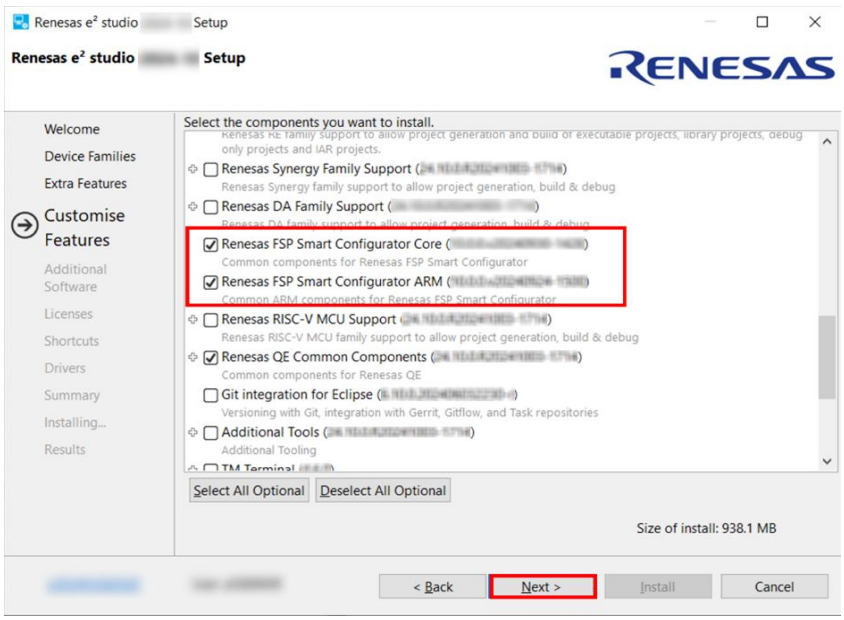


Figure 21. Installation of e²studio – Features

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.

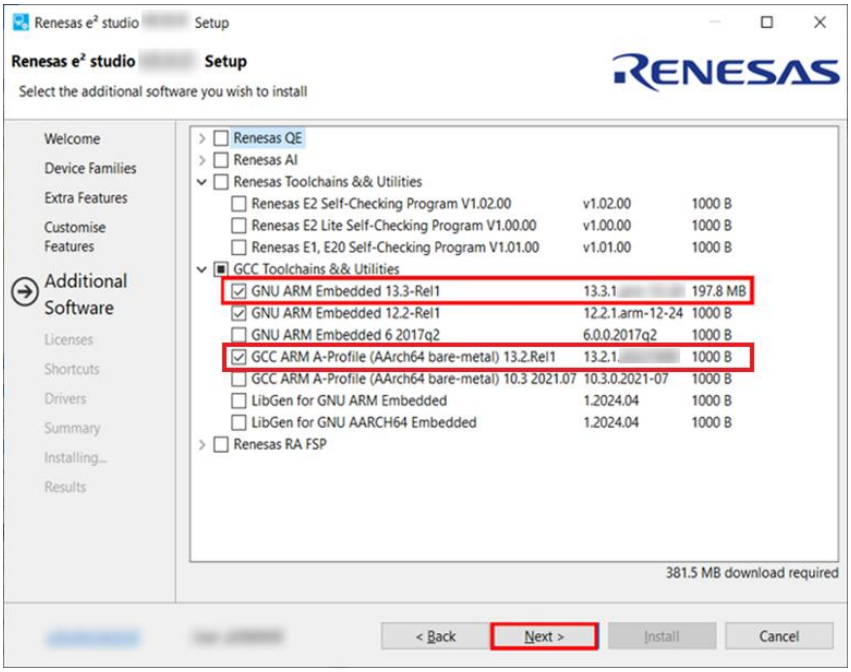


Figure 22. Installation of e²studio – Additional Software

8. **License Agreement**
   Read and accept the software license agreement. Click the [Next] button. Note that the user must accept the license agreement; otherwise, installation cannot continue.

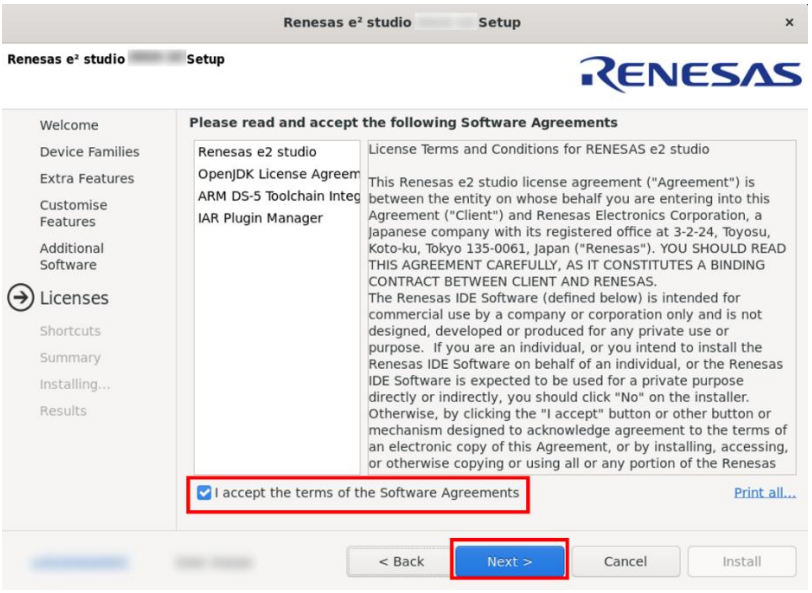


Figure 23. Installation of e²studio – Licenses

9. **Shortcuts**
   Select the shortcut name for the Application Launcher and click the [Next] button to continue.

Note: If e²studio was installed in another location, it is recommended to rename it to distinguish it from the other e²studio(s).

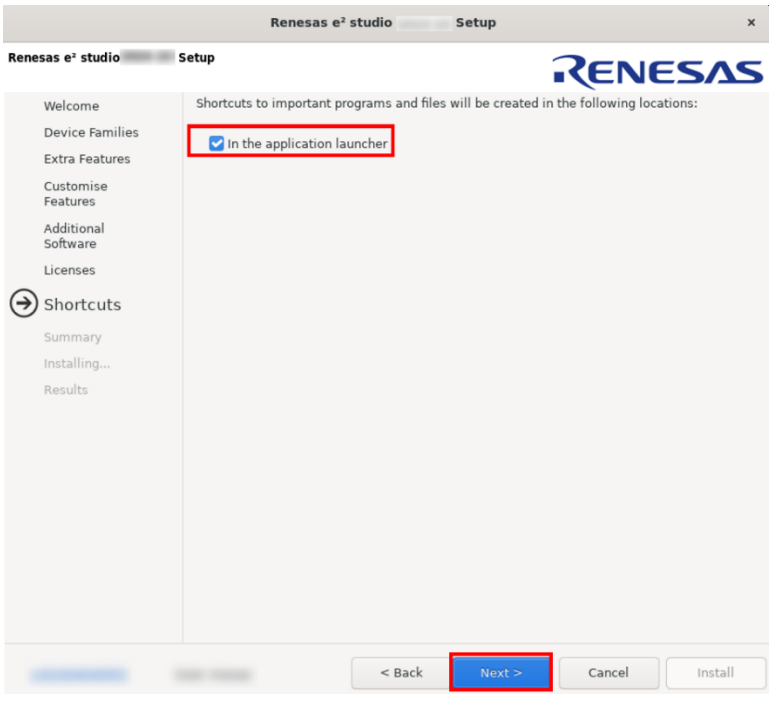


Figure 24. Installation of e²studio – Shortcuts

10. Summary

The components to be installed are shown below. Confirm the contents and click the [Install] button to install the Renesas e²studio IDE.

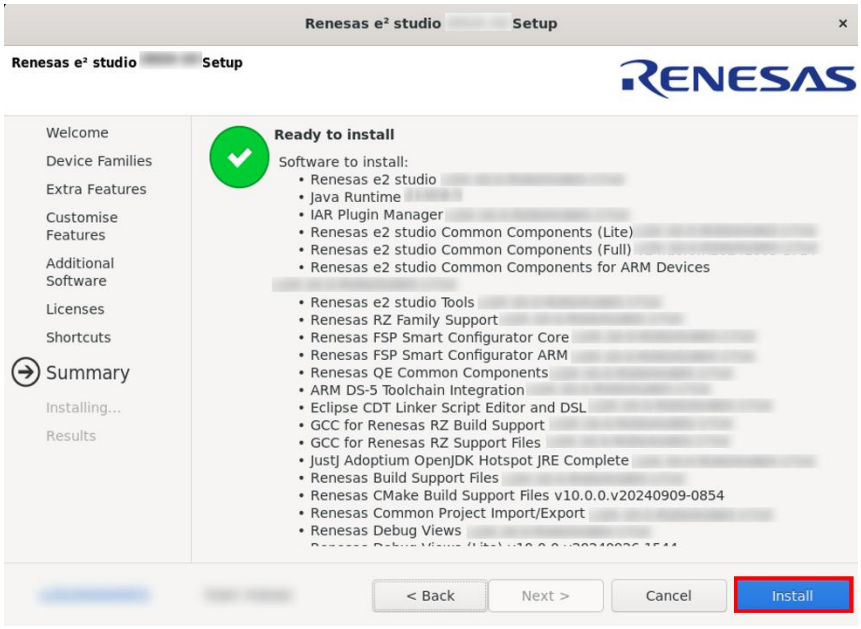


Figure 25. Installation of e²studio – Summary

11. Installing

The installation is being completed. 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.

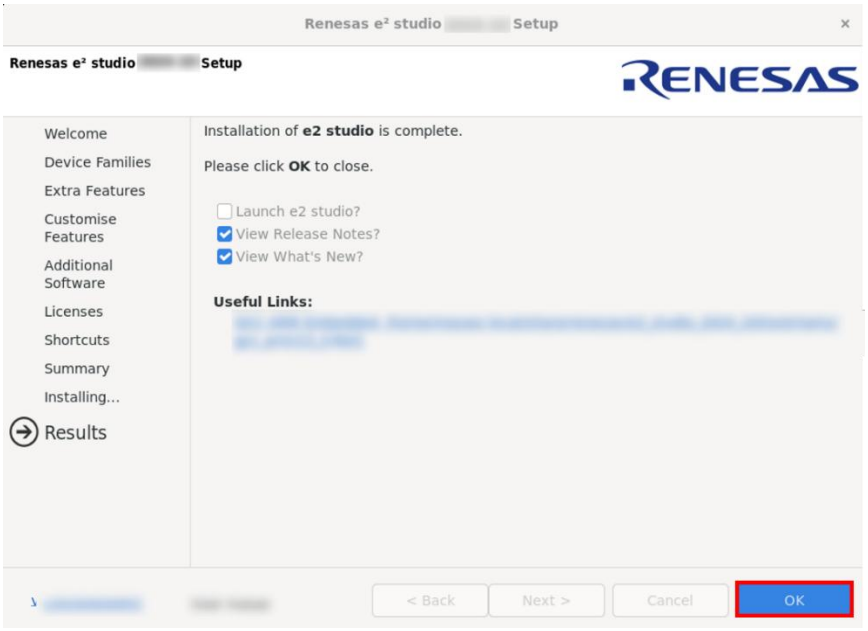


Figure 26. Summary Page

2.2 FSP Setup

This section describes three ways to install FSP.

2.2.1 Installation of FSP using Platform Installer

This section describes how to install FSP using Platform Installer **setup_rzfsp_v4_3_0_e2s_v2026-10.exe**, as shown here.

1. Double-click **setup_rzfsp_v4_3_0_e2s_v2026-10.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 e²studio was installed on your PC, the option to upgrade the existing version or install e²studio to a different location will be displayed.

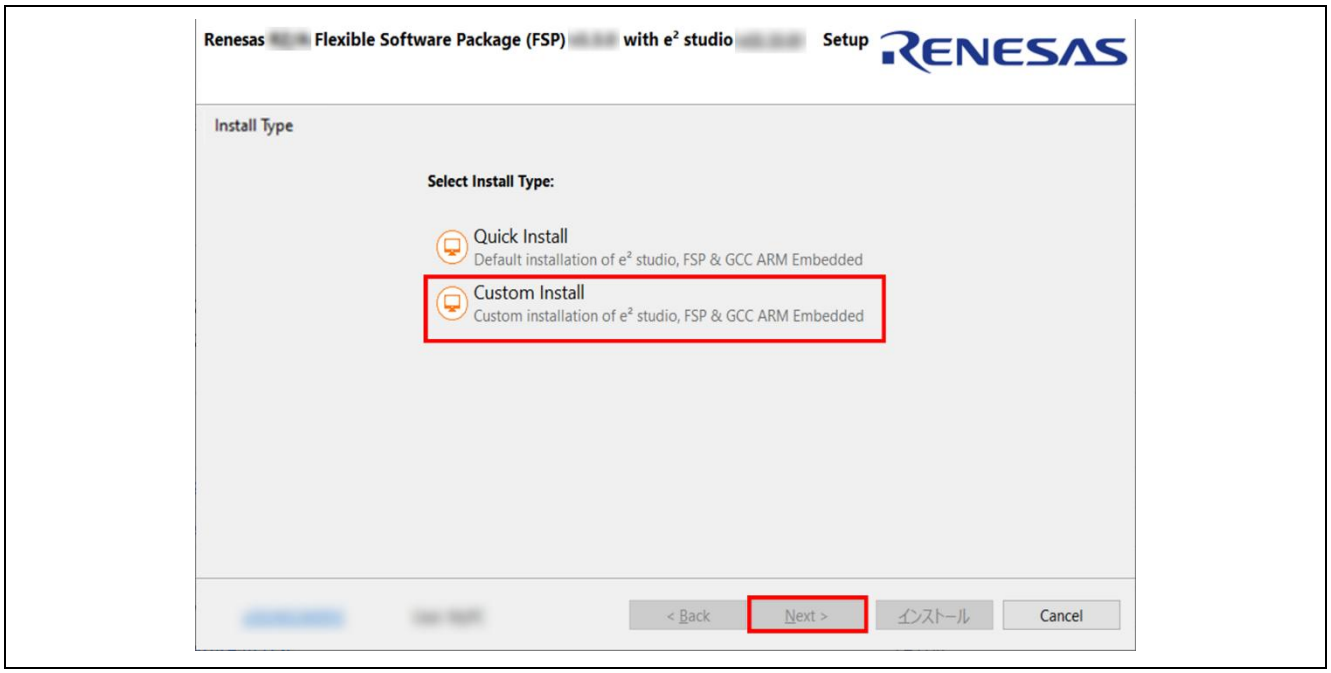


Figure 27. FSP Platform Installation Wizard

2. **Welcome page**
   The user can change the installation folder by clicking [Change…]. Click [Next] to continue.

Notes:

1. If you would like to have multiple versions of e²studio, specify a new folder here.
2. Multi-byte characters cannot be used for the e²studio installation folder name.

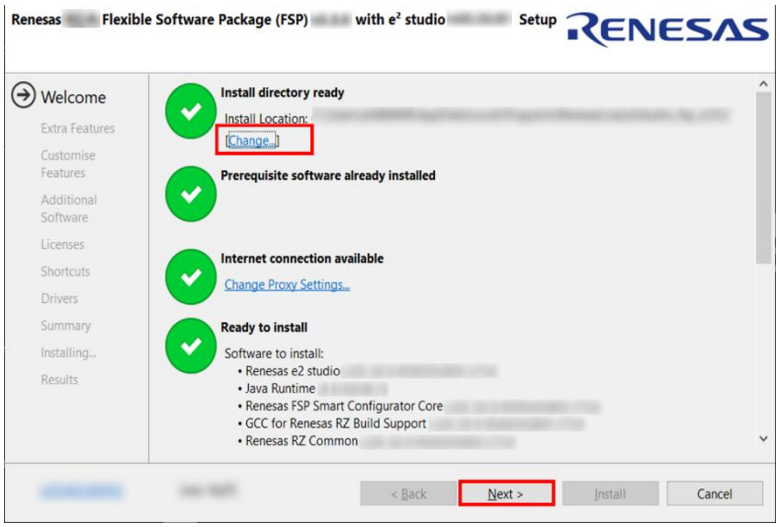


Figure 28. FSP Platform Installer – Welcome page

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

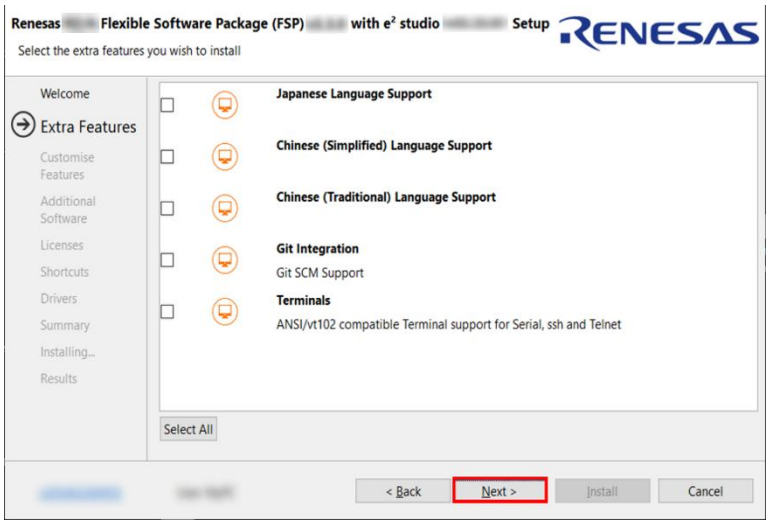


Figure 29. FSP Platform Installer – Extra Features

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

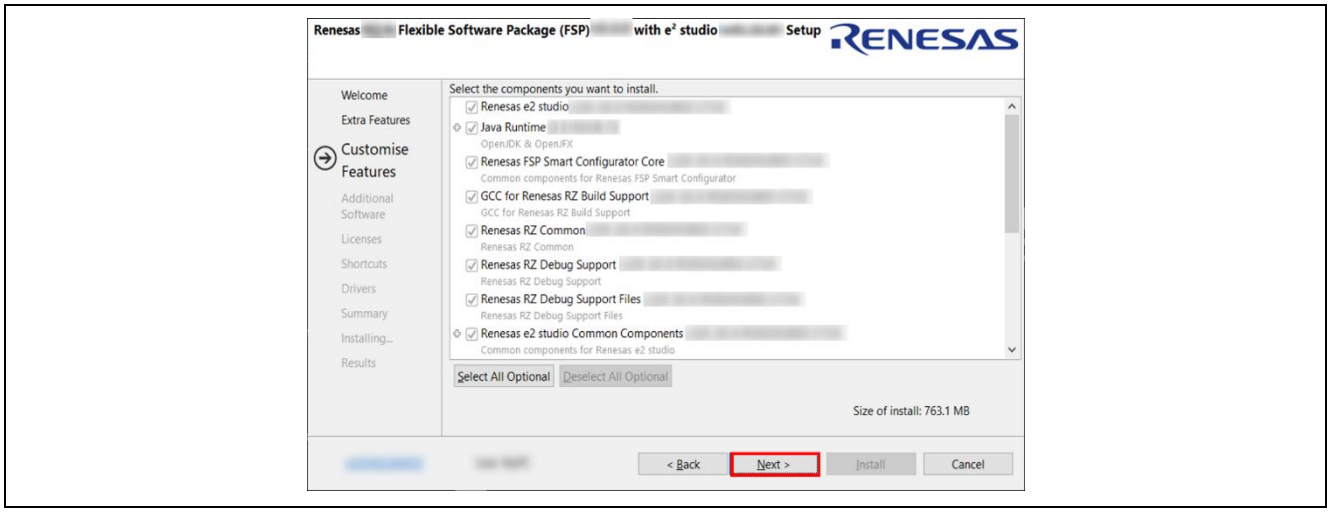


Figure 30. FSP Platform Installer – Features

5. **Additional Software**
   All the software is selected by default. Click [Next >].

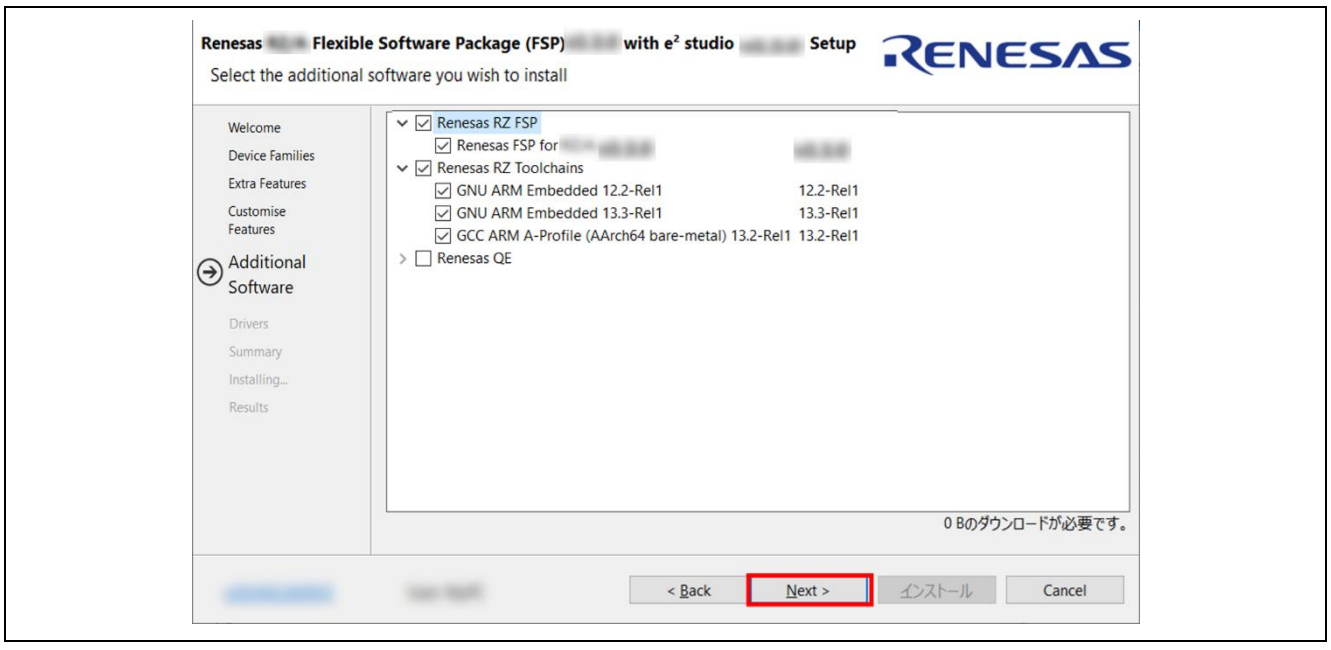


Figure 31. FSP Platform Installer – Additional Software

6. **Licenses**
   Read the listed Software License Agreements, accept them, and click [Next >].

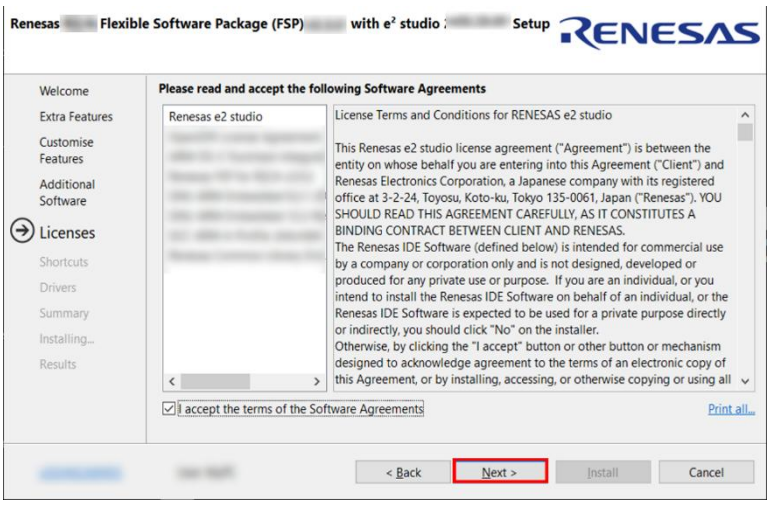


Figure 32. FSP Platform Installer – Licenses

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

Note: If e²studio was installed in another location, it is recommended to rename it to distinguish it from the other e²studio(s).

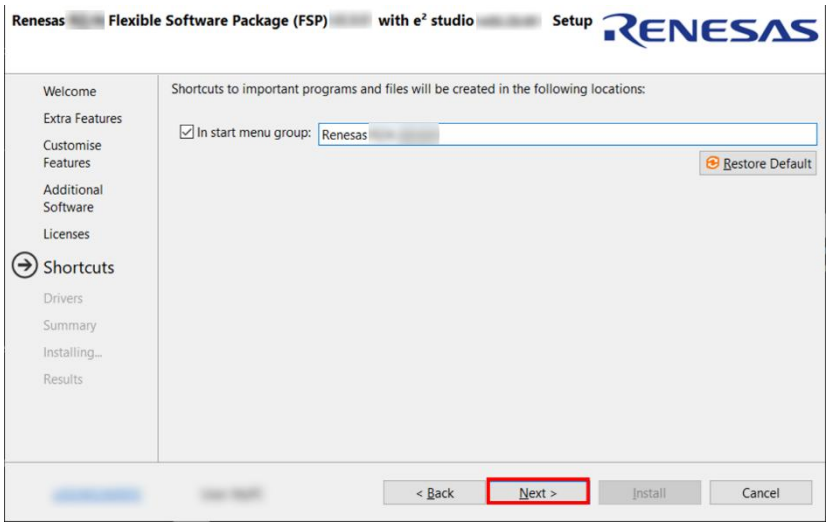


Figure 33. FSP Platform Installer – Shortcuts

8. **Summary**
   The components to be installed are shown. Confirm the contents and click the [Install] button to install the Renesas e^{2}studio IDE.

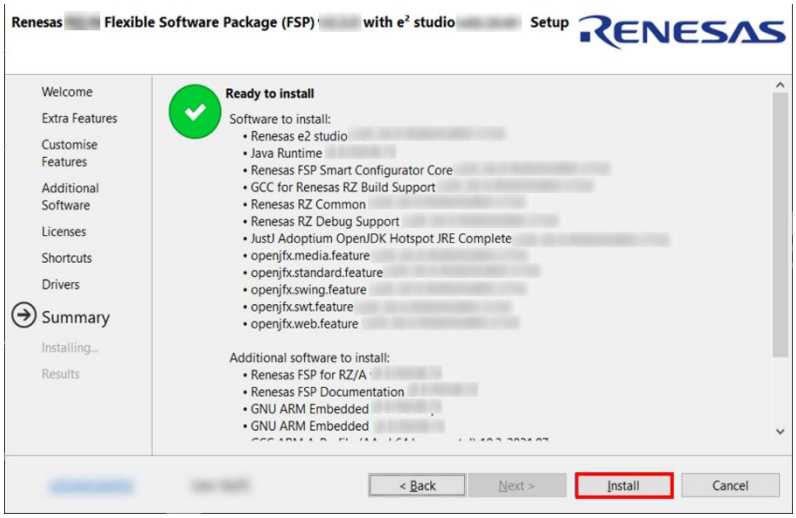


Figure 34. FSP Platform Installer – Summary

9. **Installing**
   The installation is being completed. Depending on the selected items of additional software, new dialog prompts may appear during the installation process. At that time, follow the instructions the installer indicates.

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

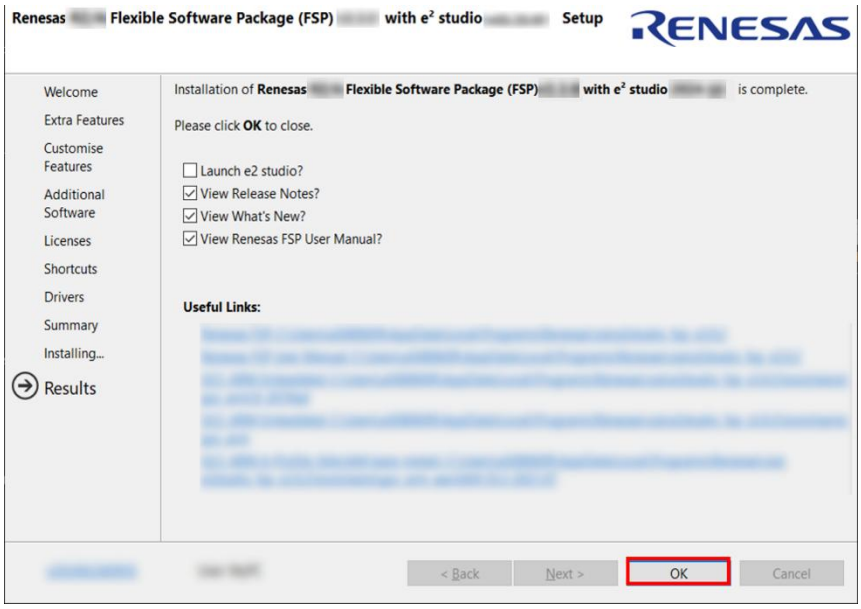


Figure 35. Installation Results of FSP Platform Installer

2.2.2 Installation of FSP using Package Installer

Package Installer **RZ_FSP_Packs_v4.3.0.exe** is showcased here. Note that it is for the Windows Host PC only.

1. Quit e²studio.
2. Invoke **RZ_FSP_Packs_v4.3.0.exe**.
3. Click [Next >] to start the installation.

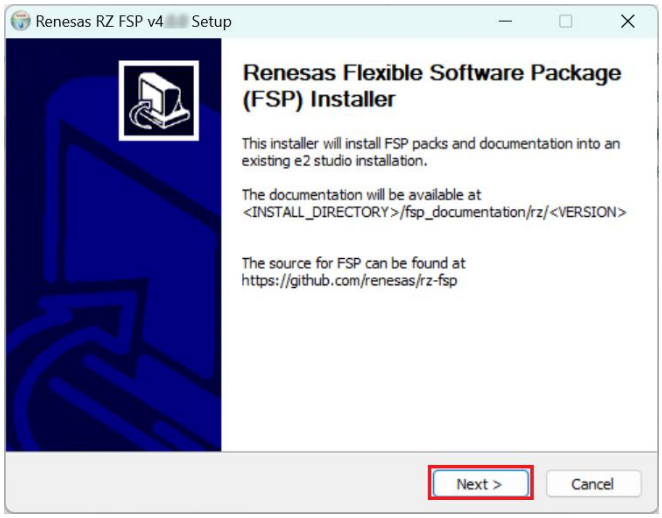


Figure 36. FSP Package Installer

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

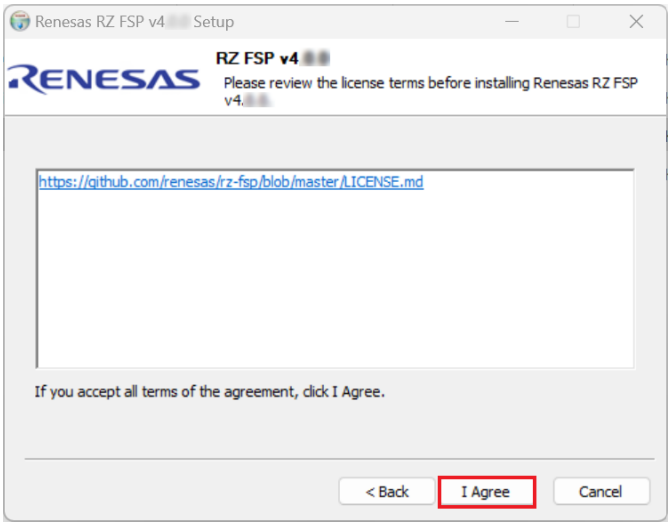


Figure 37. FSP License Term

5. Specify the e²studio installation folder (e.g., C:\Renesas\e2studio) and click the **[Install]** button.

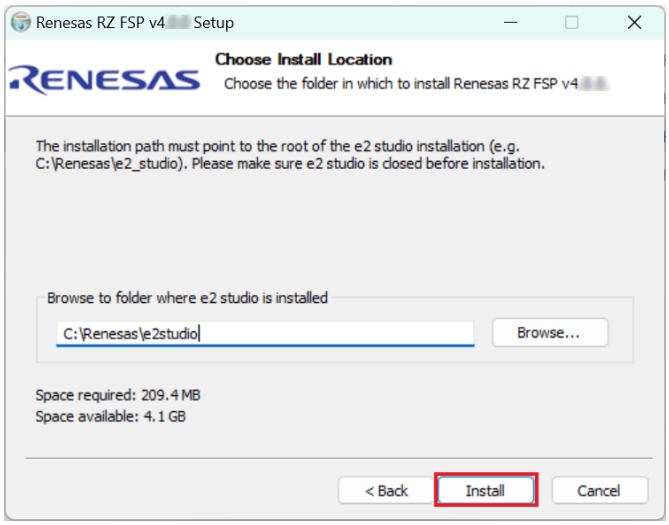


Figure 38. Browse to the folder where e²studio is installed

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

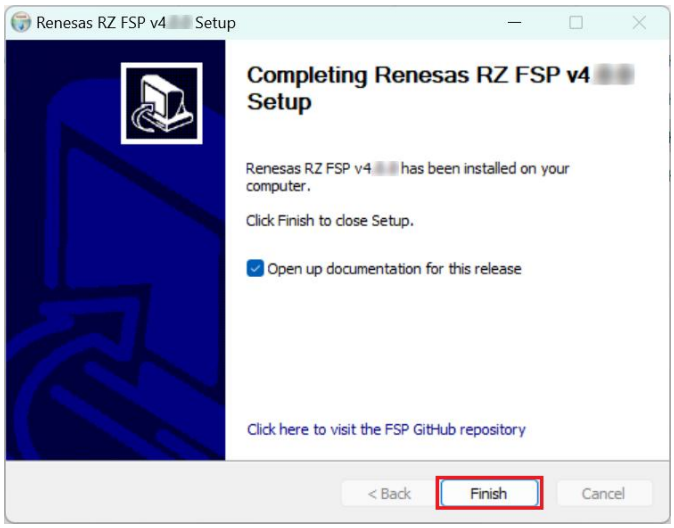


Figure 39. Completion of FSP Installation

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, the user needs to install FSP Packs with **RZ_FSP_Packs_v4.3.0.zip**. This section describes how to install it. Note that the same installation procedure is valid for the Windows Host PC.

1. Download **RZ_FSP_Packs_v4.3.0.zip** from here.
2. Extract the zip file to the e²studio installation directory. If it's successfully extracted, **rz_fsp/packs** should be placed at **<e2studio installation directory>/Internal/projectgen**.

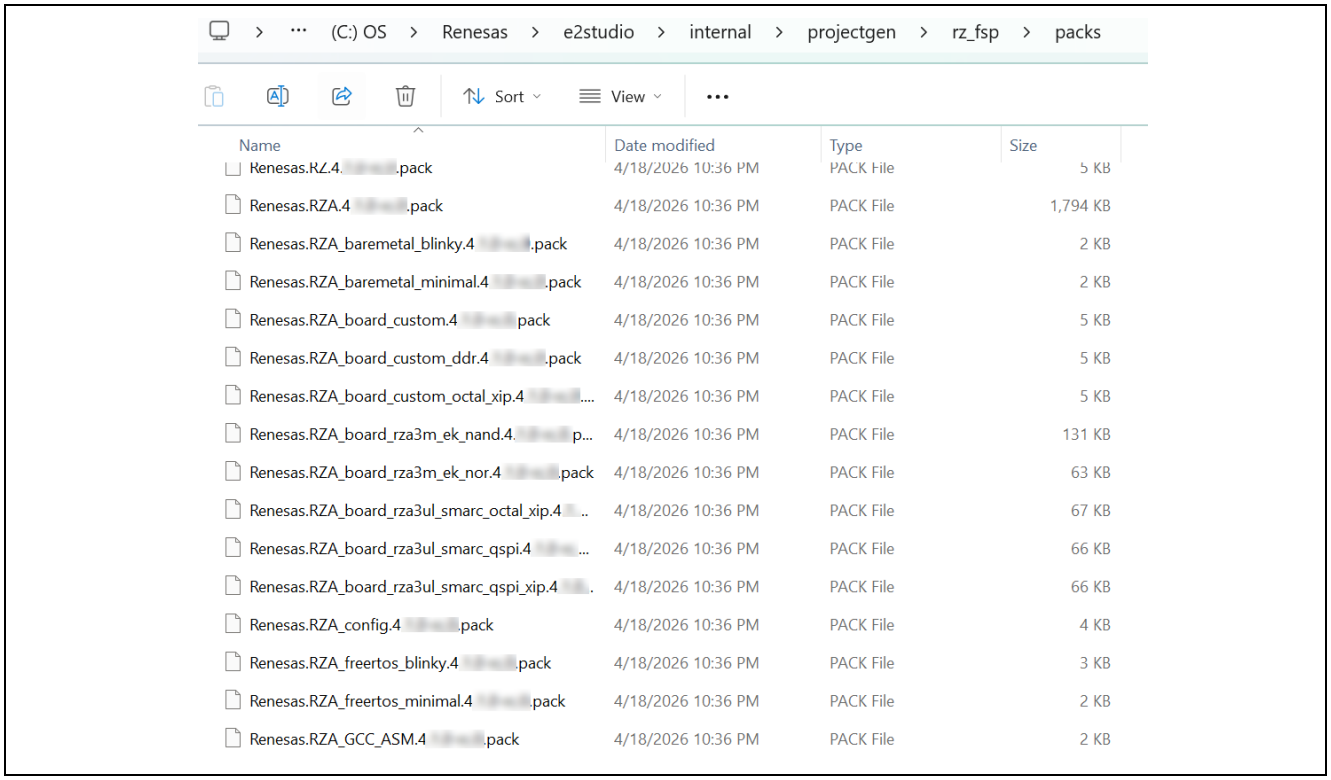


Figure 40. FSP Packs on e²studio installation directory

3. At the 1st invocation of e²studio 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***

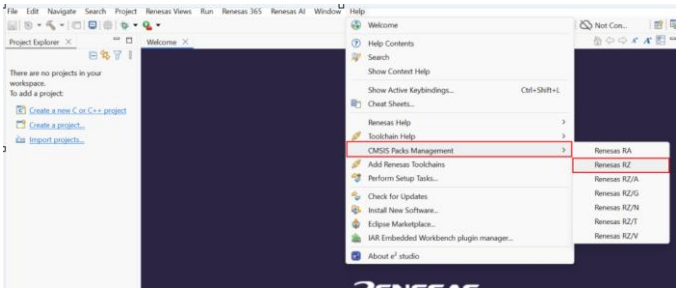


Figure 41. CMSIS Packs Management (1/2)

- If FSP is successfully installed, 4.3.0 should be listed under FSP as shown below:

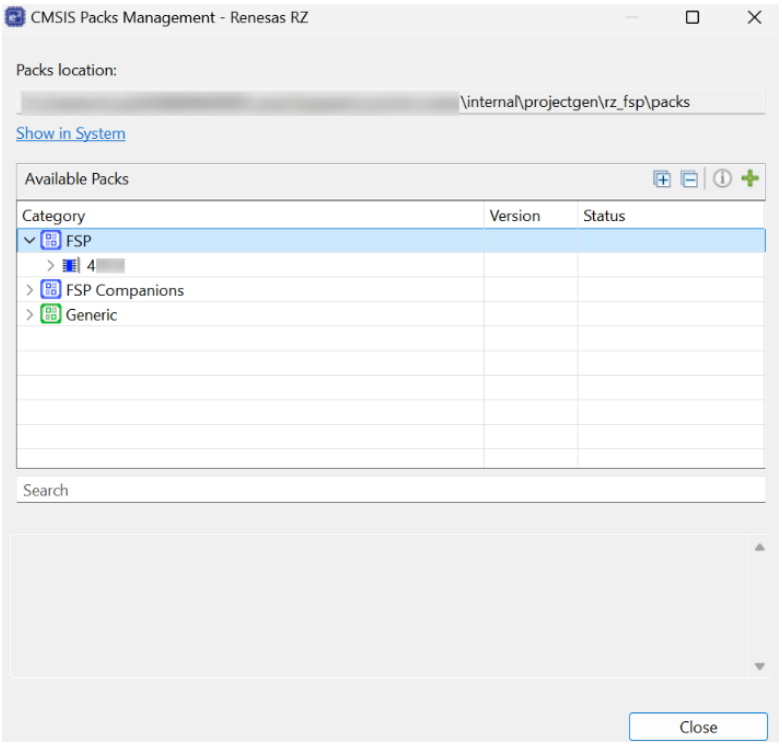


Figure 42. CMSIS Packs Management (2/2)

3. Set up the Evaluation Board

3.1 Obtaining an Evaluation Board

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

The Evaluation Board and RSK+ for RZ/T and RZ/N CPU Boards are designed to seamlessly integrate with the e^{2}studio.

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

3.2 System Configuration

Below is an example of a typical system configuration of an Evaluation Board.

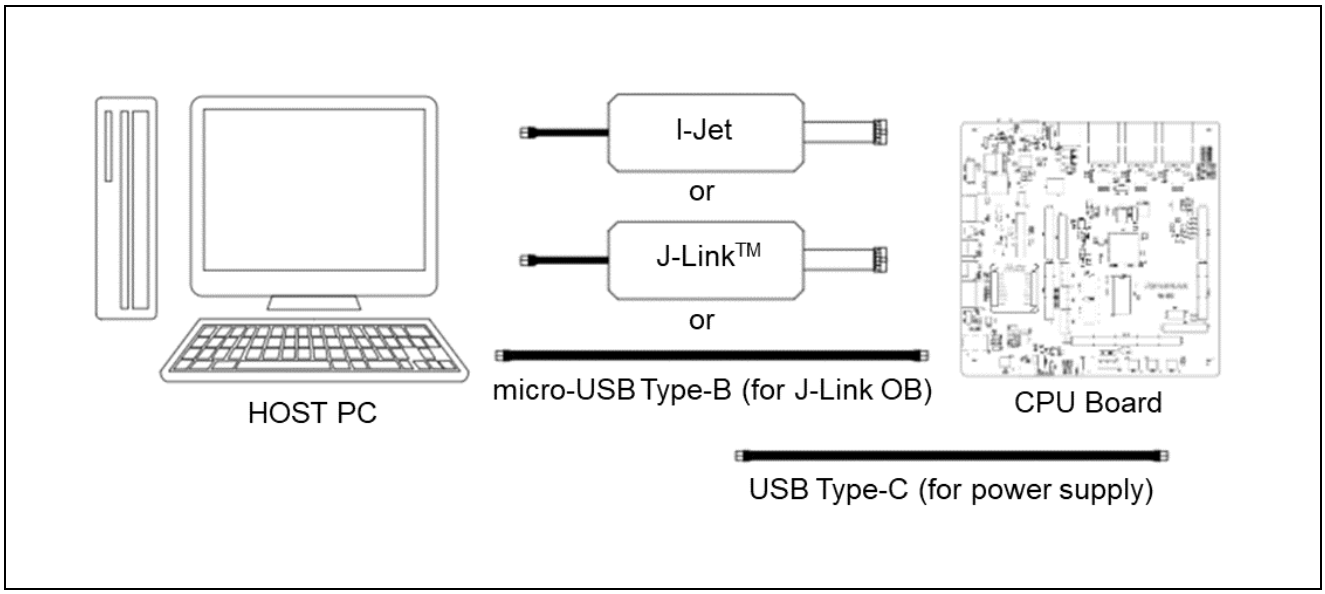


Figure 43. System Configuration Example – with Evaluation Board

For the details, refer to the section *1.3.1 Evaluation Board User's Manual*.

3.3 Supported Emulators

3.3.1 SEGGER J-Link

SEGGER J-Link can be used on Renesas e^{2}studio only for debugging on RZ/T and RZ/N devices.
Renesas e^{2}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/T and RZ/N devices with J-Link BASE V11 and J-Link OB-S124.
For details on SEGGER J-Link, 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^{2}studio version	J-Link Software version
RZ/T	RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/T2N	RZ FSP v4.3.0	2026-10	V9.70
RZ/N	RZ/N2L, RZ/N2H			

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

3.3.2 IAR I-Jet

IAR I-jet can be used on IAR EWARM only for debugging on RZ/T and RZ/N devices.
For details on I-jet, 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, refer to the manual of the RSK boards listed in the section *1.3.1 Evaluation Board User's Manual*. For the sample codes available on the Renesas website, refer to the documentation included with each code and implement the appropriate board settings, respectively.

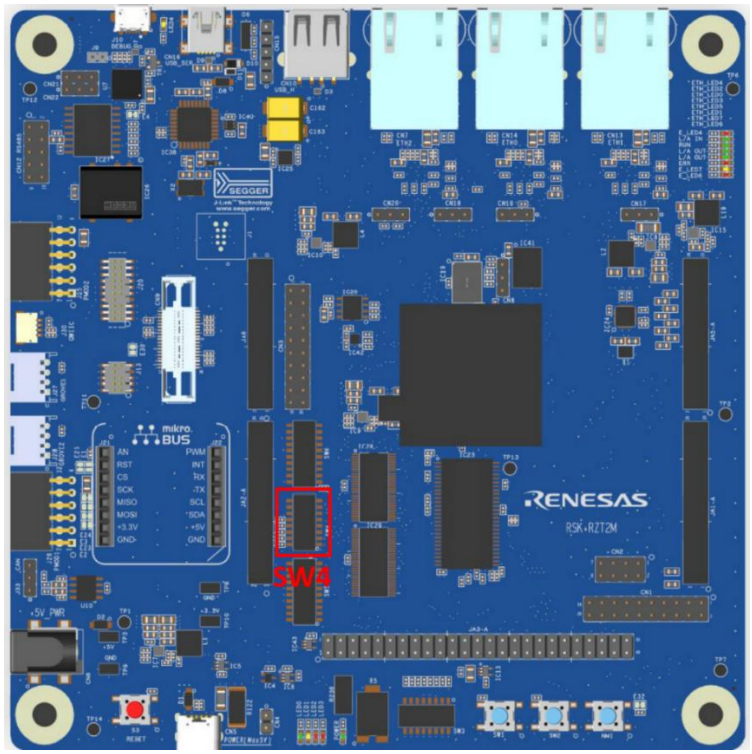


Figure 44. Switch Position of Operation Mode Settings for RSK+RZT2M

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) to switch 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”.

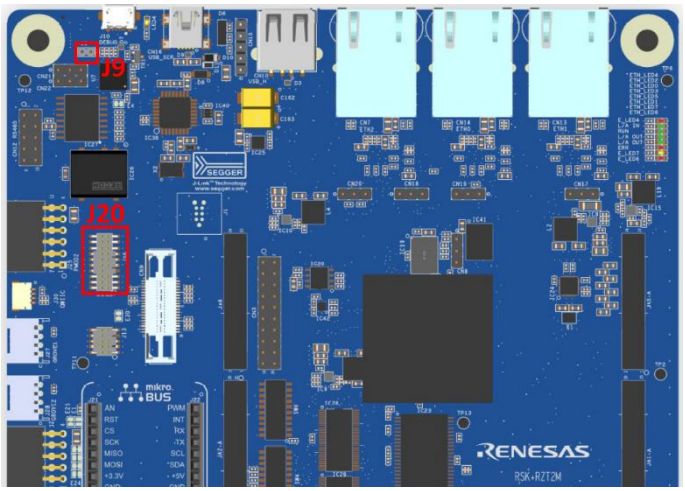


Figure 45. Jumper Position of JTAG Connection for RSK+RZT2M

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

1. Open the jumper pin (J9) to switch 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). LED4 then lights up.

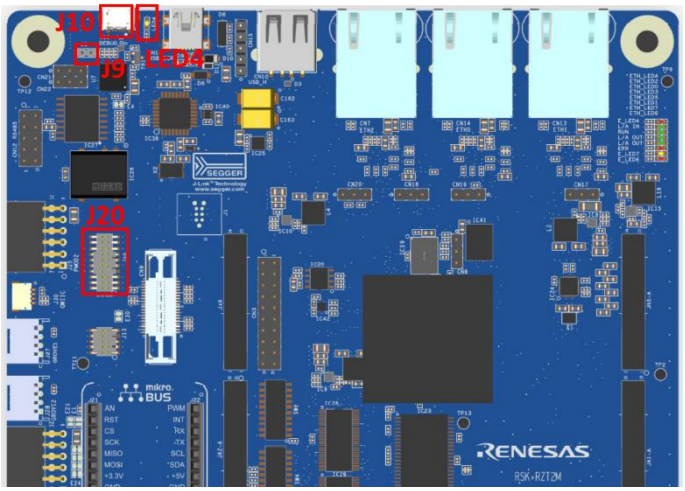


Figure 46. J-Link OB Connection Settings for RSK+RZT2M

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.

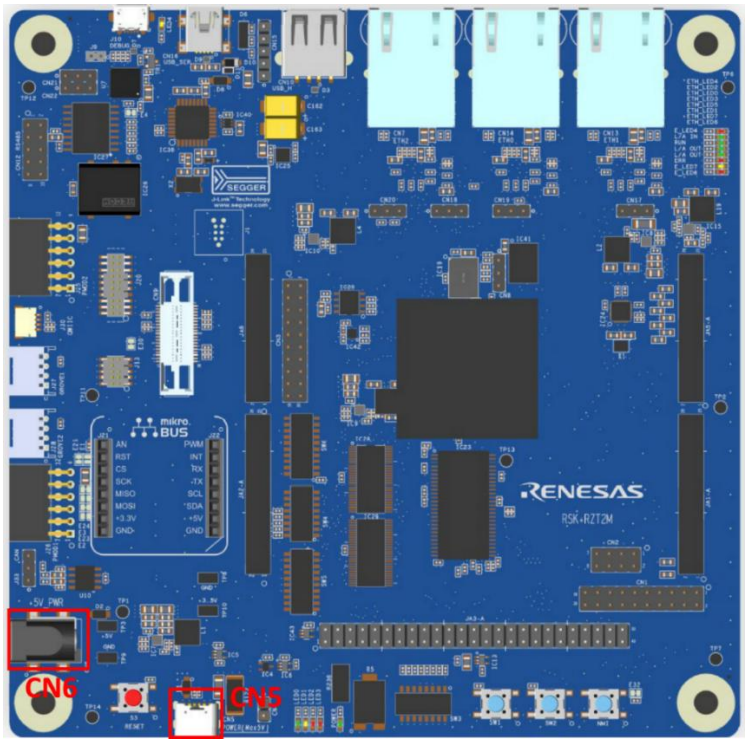


Figure 47. Power Supply for RSK+RZT2M

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, refer to the manual of the RSK boards listed in section *1.3.1 Evaluation Board User's Manual*. For the sample codes available on the Renesas website, refer to the documentation included with each code and implement the appropriate board settings, respectively.

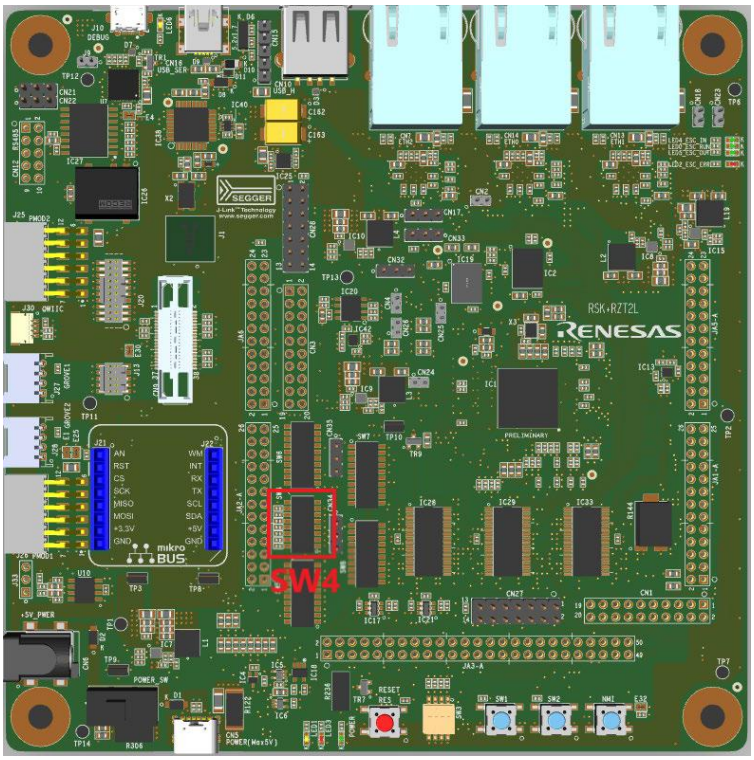


Figure 48. Switch Position of Operation Mode Settings for RSK+RZT2L

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) to switch 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”.

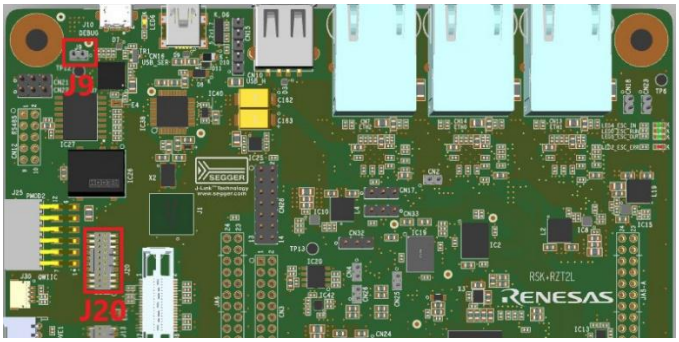


Figure 49. Jumper Position of JTAG Connection for RSK+RZT2L

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

1. Open the jumper pin (J9) to switch 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). LED6 lights up.



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.

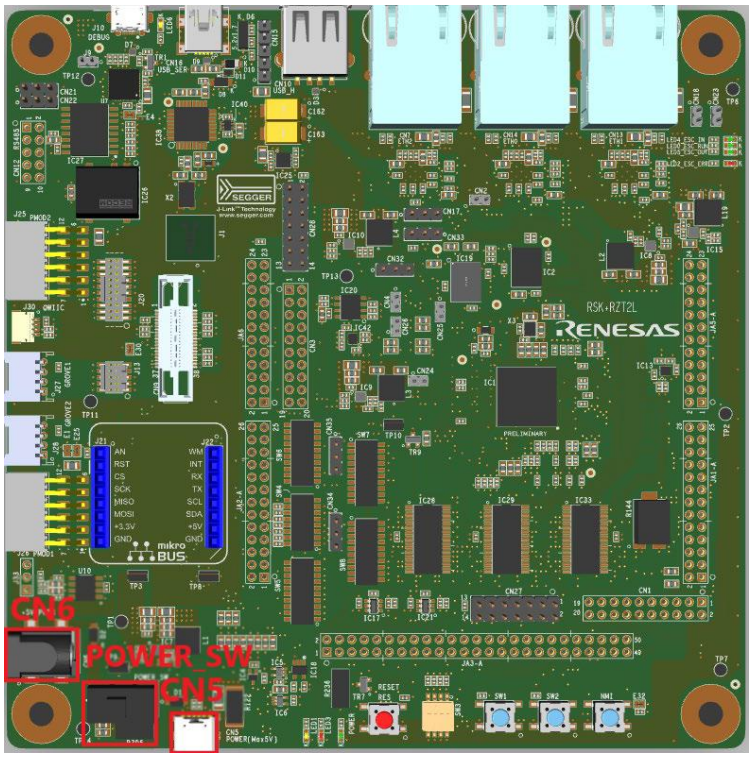


Figure 51. Power Supply for RSK+RZT2L

3.4.3 RSK+RZT2ME

For each setting, see section *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, refer to the manual of the Evaluation Board listed in section *1.3.1 Evaluation Board User's Manual*. For the sample codes available on the Renesas website, refer to the documentation included with each code and implement the appropriate board settings, respectively.

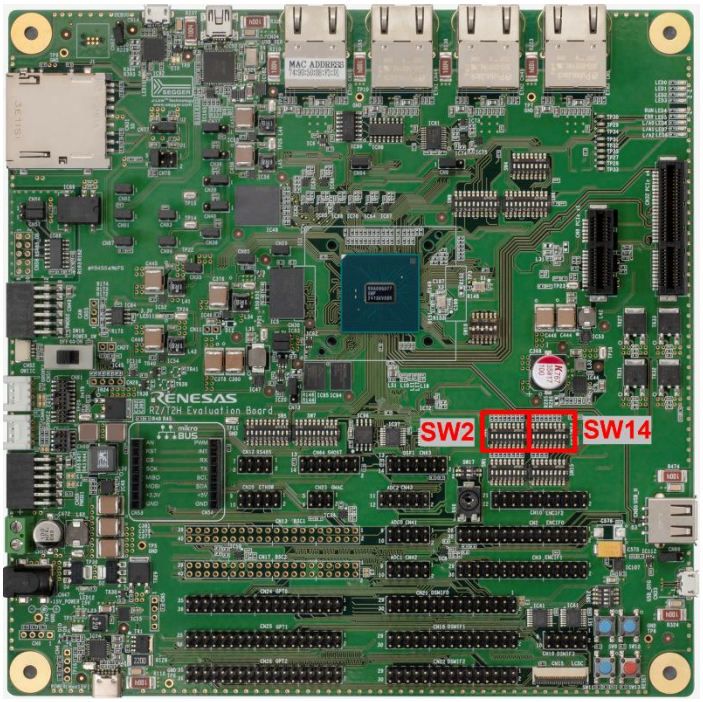


Figure 52. Switch Position of Operation Mode Settings for RZ/T2H Evaluation Board

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	Core0 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) to switch 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”.

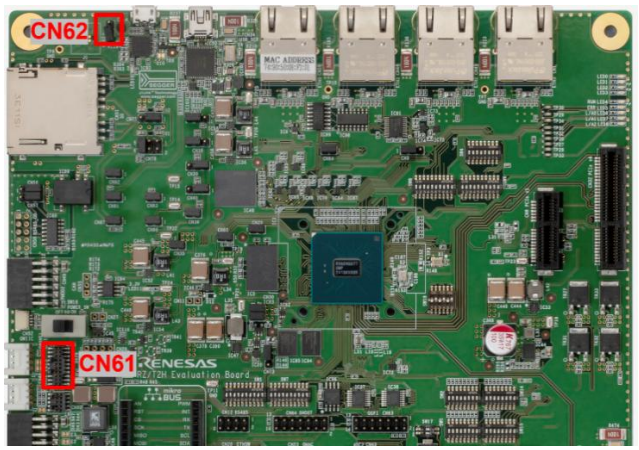


Figure 53. Jumper Position of JTAG Connection for RZ/T2H Evaluation Board

If you use J-Link OB on the RZ/T2H Evaluation Board,

1. Open the jumper block (CN62) to switch 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 LED10 is lit.

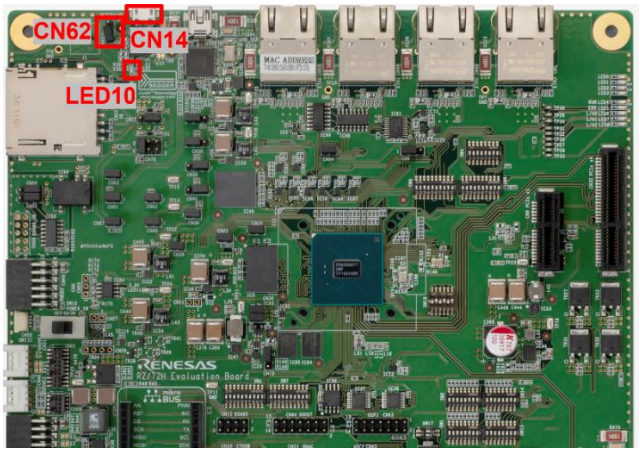


Figure 54. J-Link OB Connection Settings for RZ/T2H Evaluation Board

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 12V or 5V power supply; the supply of this board is 15V/3A. Do NOT accidentally connect a 12V or 5V power supply. When supplying power through CN47, use a stabilized power source that is capable of supplying at least 15V/3A.

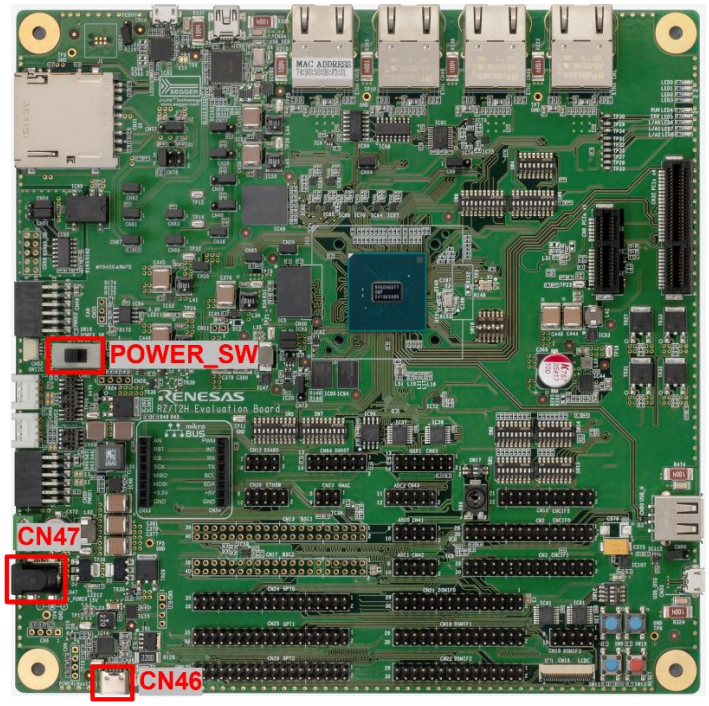


Figure 55. How to Power Supply for RZ/T2H Evaluation Board

3.4.5 RZ/T2N Evaluation Board

3.4.5.1 Boot Mode

The operation mode settings for the RZ/T2N 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, refer to the manual of the Evaluation Board listed in section *1.3.1 Evaluation Board User's Manual*.

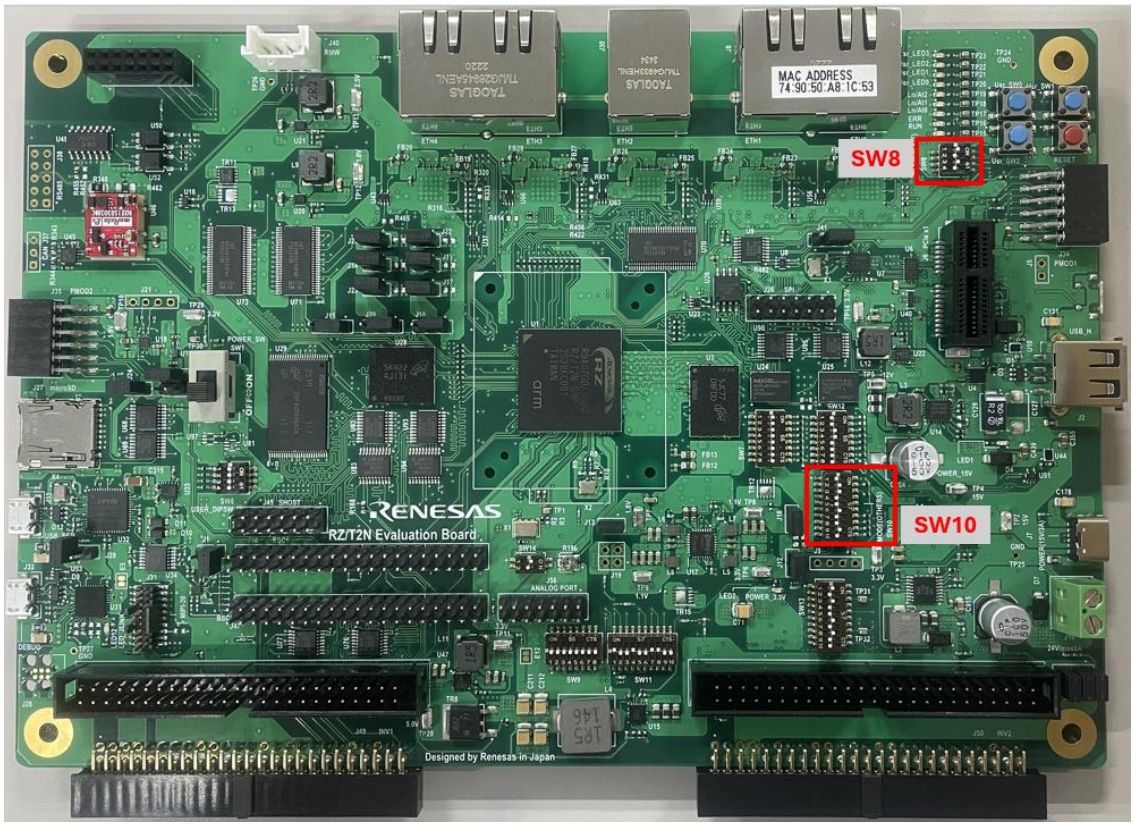


Figure 56. Switch Position of Operation Mode Settings for RZ/T2N Evaluation Board

Table 5. Operation Mode Switch Settings for RZ/T2N Evaluation Board

Switch	Setting	Description
SW8.1	ON	xSPI1 boot mode (x1 boot serial flash).
SW8.2	OFF	
SW8.3	ON	
SW10.7	ON	CR52 Core0 ATCM wait cycle = 0 wait
SW10.8	ON	CR52 Core1 ATCM wait cycle = 0 wait
SW10.9	OFF	JTAG Authentication by Hash is disabled.

3.4.5.2 Debugger Connection

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

1. Short the jumper block (J29) to switch the debug connection so that the RZ/T2N Evaluation Board can use the emulator connected to the JTAG connector (J31).
2. Connect the emulator (J-Link or I-Jet) to a free USB port on your computer.
3. Connect the emulator to the RZ/T2N Evaluation Board, ensuring that it is plugged into the header “J31”.

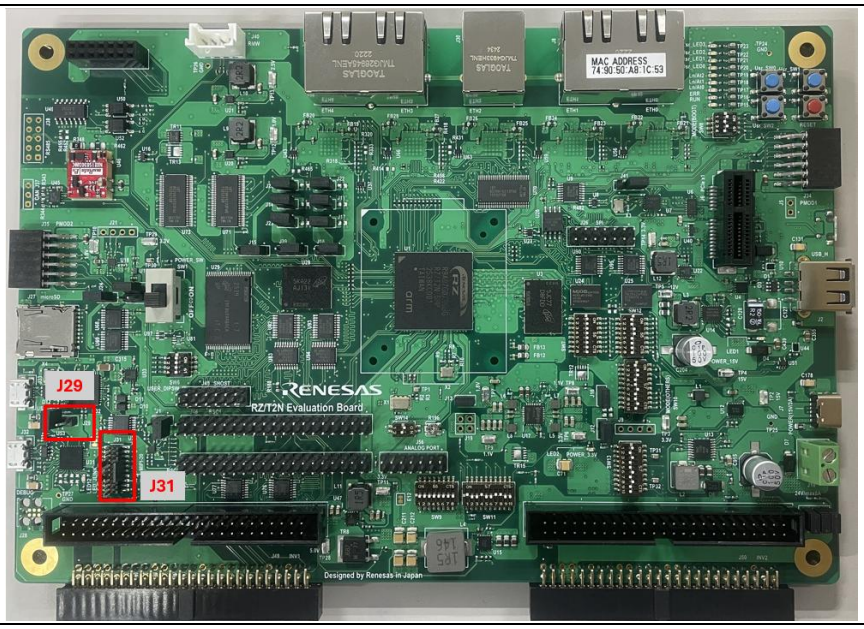


Figure 57. Jumper Position of JTAG Connection for RZ/T2N Evaluation Board

If you use J-Link OB on the RZ/T2N Evaluation Board,

1. Open the jumper block (J29) to switch the debug connection so that the RZ/T2N Evaluation Board can use J-Link OB on the board.
2. Connect the micro-USB Type-B to the J-Link OB USB connector (J32). LED12 then lights up.

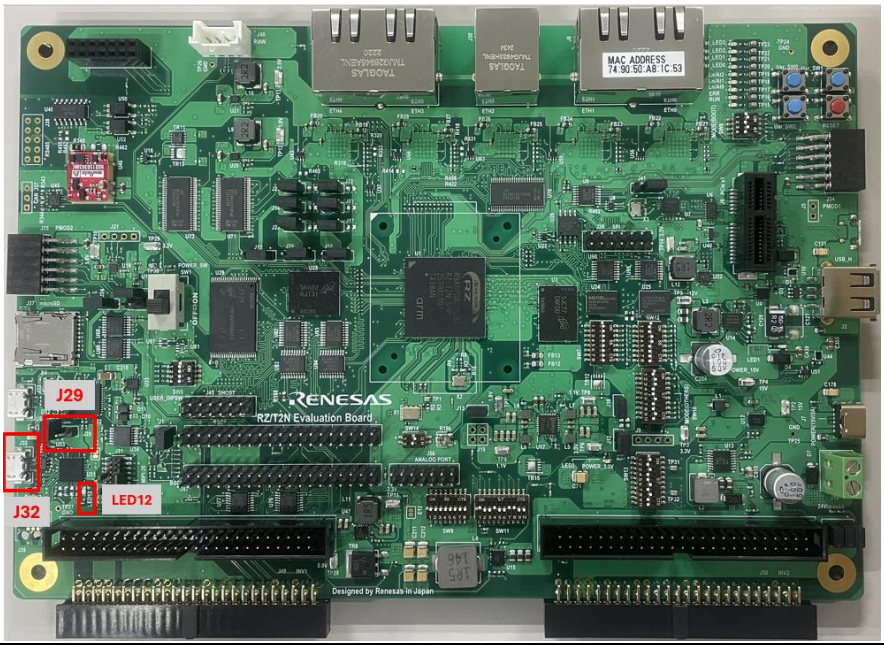


Figure 58. J-Link OB Connection Settings for RZ/T2N Evaluation Board

3.4.5.3 Power Supply

Power is supplied using a USB-C cable; connect it to the USB-C connector (J7) of the RZ/T2N Evaluation Board. After connecting to the power (J7), turn on the POWER_SW (SW1) slide switch to start the power supply.

Note: When supplying power through J7, always use an AC adapter for USB supporting 15V/3A.

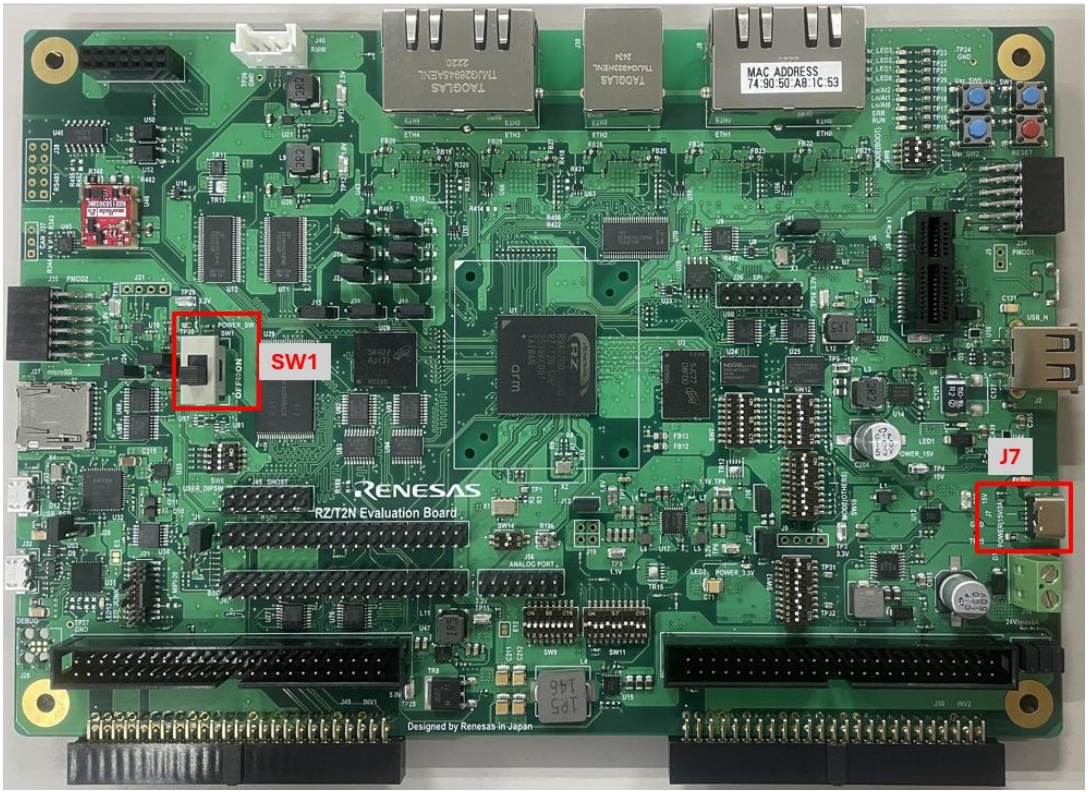


Figure 59. How to Power Supply for RZ/T2N Evaluation Board

3.5 RZ/N Series Board Setup

3.5.1 RSK+RZN2L

3.5.1.1 Boot Mode

The following section describes the operation mode settings for the RSK + RZN2L Board.

Note: This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, refer to the manual of the RSK boards listed in section *1.3.1 Evaluation Board User's Manual*. For the sample codes available on the Renesas website, refer to the documentation included with each code and implement the appropriate board settings, respectively.

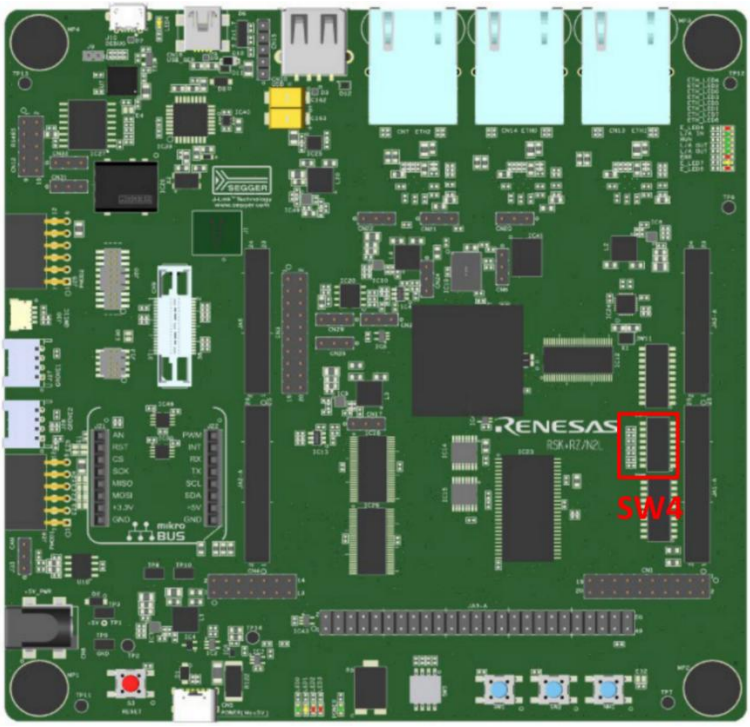


Figure 60. Switch Position of Operation Mode Settings for RSK+RZN2L

Table 6. 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) to switch 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.

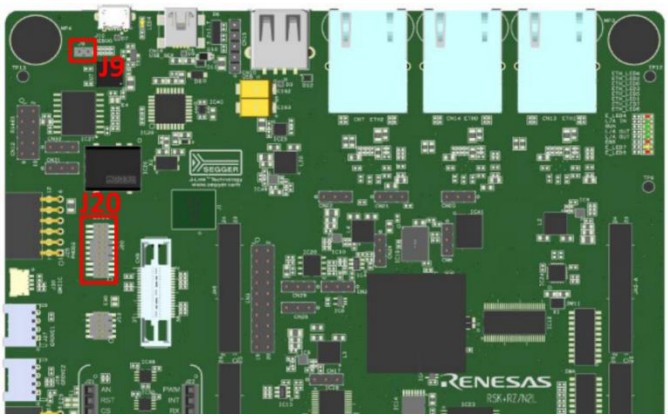


Figure 61. Jumper Position of JTAG Connection for RSK+RZN2L

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

1. Open the jumper pin (J9) to switch 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). LED4 then lights up.

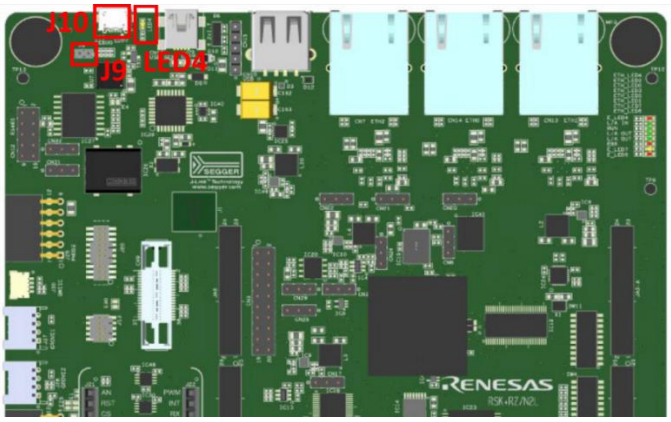


Figure 62. J-Link OB Connection Settings for RSK+RZN2L

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.

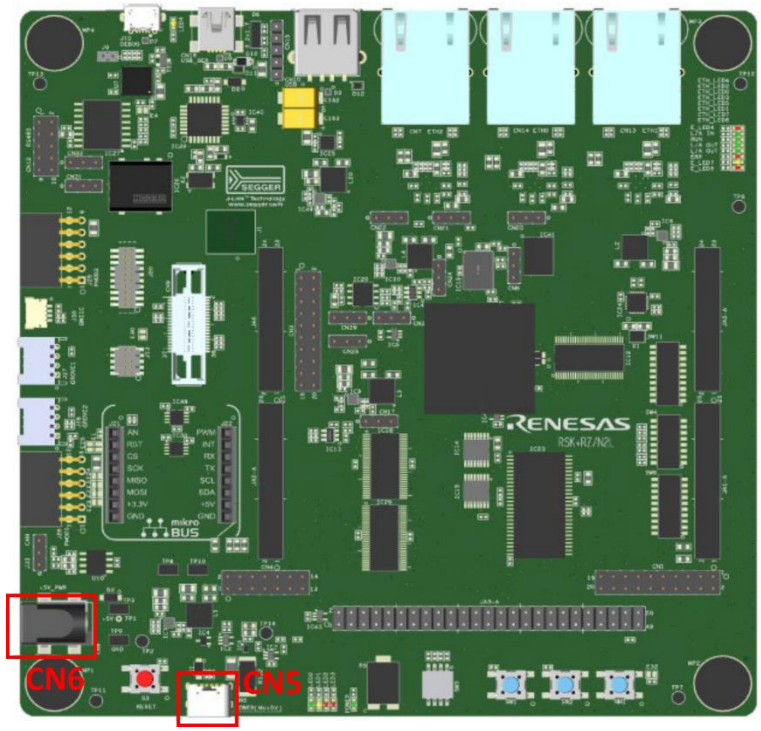


Figure 63. How to Power Supply for RSK+RZN2L

3.5.2 RZ/N2H Evaluation Board

3.5.2.1 Boot Mode

The following section describes the operation mode settings for the RZ/N2H Evaluation Board.

Note: This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, refer to the manual of the Evaluation Board listed in section *1.3.1 Evaluation Board User's Manual*. For the sample codes available on the Renesas website, refer to the documentation included with each code and implement the appropriate board settings, respectively.

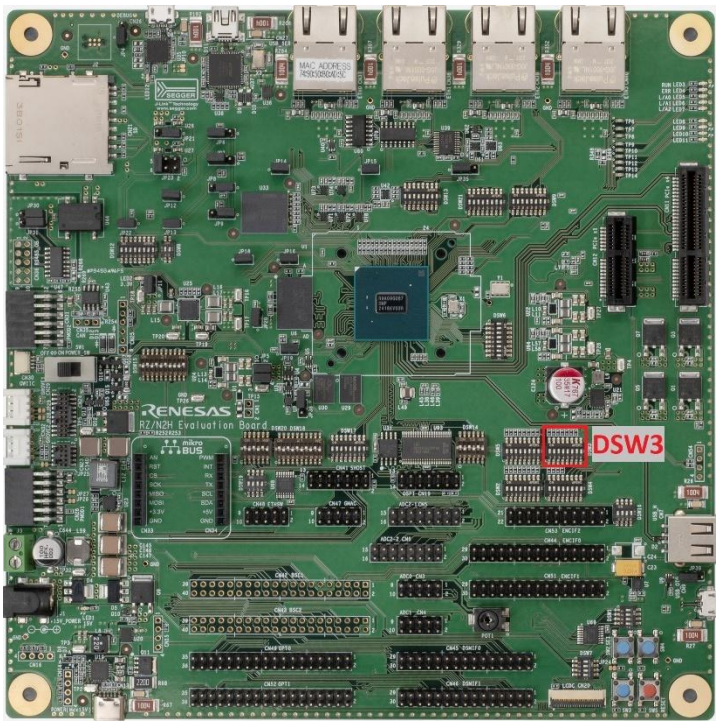


Figure 64. Switch Position of Operation Mode Settings for RZ/N2H Evaluation Board

Table 7. 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	Core0 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) to switch 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”.

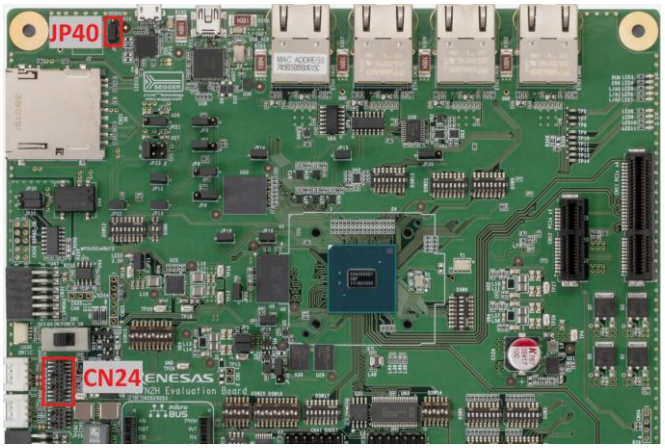


Figure 65. Jumper Position of JTAG Connection for RZ/N2H Evaluation Board

If you use J-Link OB on the RZ/N2H Evaluation Board,

1. Open the jumper block (JP40) to switch 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). LED12 then lights up.

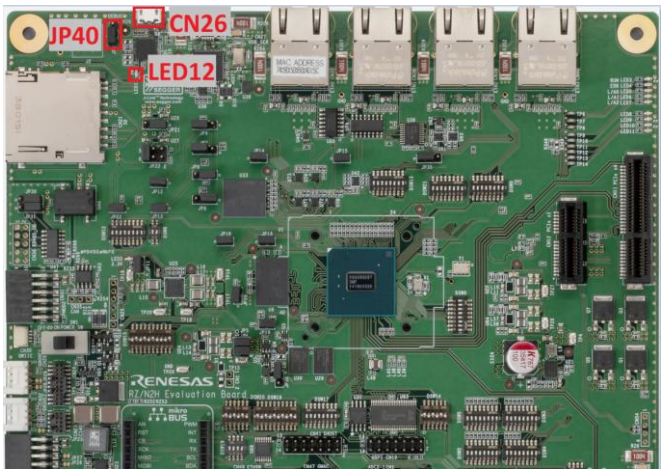


Figure 66. J-Link OB Connection Settings for RZ/N2H Evaluation Board

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 12V or 5V power supply; the supply of this board is 15V/3A. Do NOT accidentally connect a 12V or 5V power supply. When supplying power through J1, use a stabilized power source that is capable of supplying at least 15V/3A.

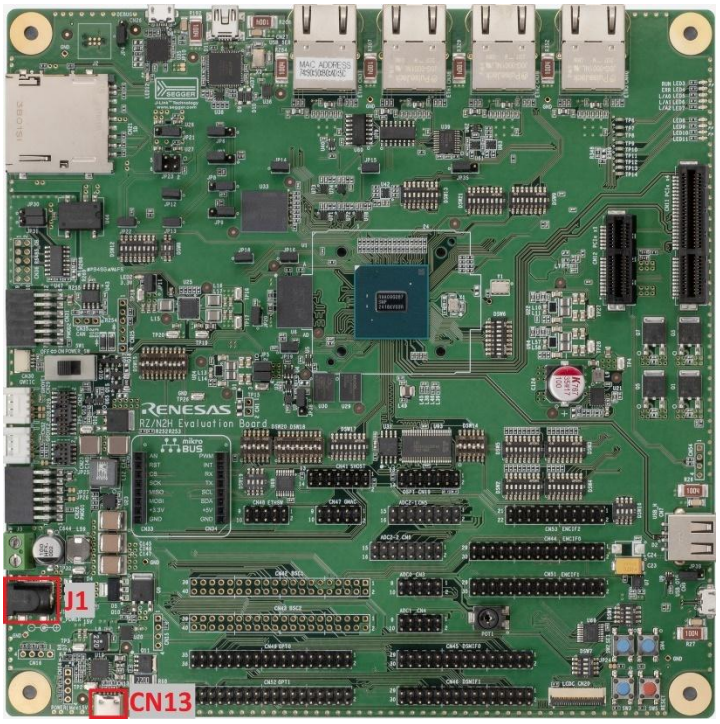


Figure 67. How to Power Supply for RZ/N2H Evaluation Board

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 e²studio and running that application on an RZ/T, RZ/N Evaluation Board. This chapter guides you through creating projects for single-core processing.

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 C/C++ FSP Project is the first step in the creation of an application. The RZ FSP includes 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 examples of projects that run on a single core. Refer to *Table 8 Project Creation Procedure (e2studio)* to find out which steps are required for your application.

Table 8. Project Creation Procedure (e²studio)

<table>
<tr><td rowspan="3">Steps</td><td colspan="3">Single-core processing</td><td rowspan="3">Multiprocessing</td></tr>
<tr><td rowspan="2">RAM execution</td><td colspan="2">Flash boot mode</td></tr>
<tr><td>xSPI
NOR flash</td><td>eSD
eMMC</td></tr>
<tr><td>Check tool limitations</td><td colspan="4">エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。</td></tr>
<tr><td>Erase flash memory (if needed)</td><td colspan="4">8.1 How to Erase Flash Memory</td></tr>
<tr><td>Create a project</td><td colspan="3">4.3 Create a New Project for Blinky</td><td rowspan="4">10 Appendix. How to Create and Debug FSP Projects for Multiprocessing for e2studio</td></tr>
<tr><td>Build the project</td><td colspan="3">4.4 Build the Blinky Project</td></tr>
<tr><td>Debug the project</td><td colspan="2">4.5.2 Debug Steps</td><td rowspan="2">RZ FSP Documentation
eSD boot (RZT, RZN)
eMMC boot (RZT, RZN)</td></tr>
<tr><td>Run the project</td><td colspan="2">4.6 Run the Blinky Project</td></tr>
</table>

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 this section, using the IDE (e²studio or IAR EWARM). For eSD and eMMC boot modes, tools other than the IDE are required when programming to external flash memory devices. Refer to the FSP Documentation for details on creating image files, the programming process, and debugging procedures.

Follow these steps to create an RZ/T, RZ/N MPU project:

1. In e²studio, click **File** > **New** > **Renesas C/C++ Project > Renesas RZ**.

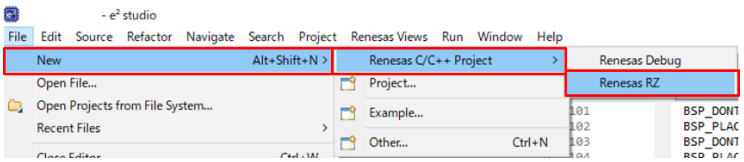


Figure 68. New C/C++ Project

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

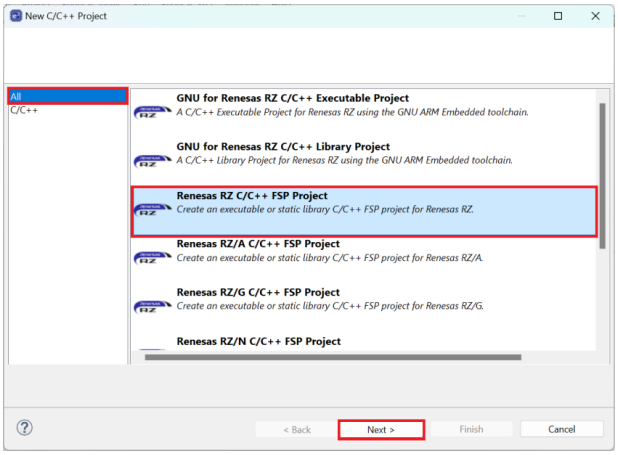


Figure 69. Renesas RZ C/C++ FSP Project

4. Assign a name for this new project. Blinky is a good name to use for this tutorial.
5. Click **Next**. The Project Configuration window shows your selection.

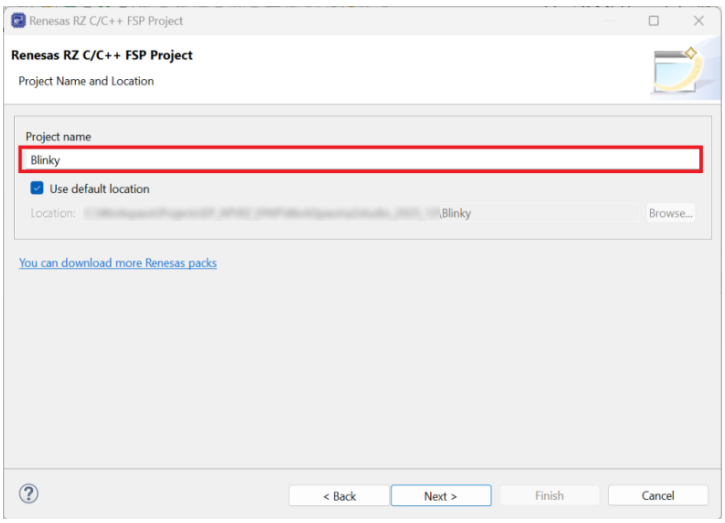


Figure 70. e²studio Project Configuration Window (1/6)

6. Select the **FSP Version** from the drop-down list.
7. Select the **Board** for your application from the drop-down list.
   - In this tutorial, select either one depending on your device and board.
8. Select the **Core** from the drop-down list.
   - Only **Cortex-R52 Core0** or **Cortex-A55 Core0** can be designated as the Primary project.
9. Select the **IDE Project Type.**
10. Select **Toolchains** and **Version** from the drop-down list.
    - If there is NO target toolchain, download the version of the toolchain from the ARM Developer website and install it.
11. Select the **Debugger** from the drop-down list, then click **Next**.

Table 9. e²studio Newly Created Project Settings

Setting	Value	
	CR52	CA55
FSP Version	(Select the FSP version to be used.)	
Board	(Select the combination of the evaluation board and boot mode to be used.)	
Core	Cortex-R52 Core0 or Cortex-R52 Core1	Cortex-A55 Core0, Cortex-A55 Core1, Cortex-A55 Core2 or Cortex-A55 Core3
IDE Project Type	e²studio managed build	
Toolchains	GNU ARM Embedded	GNU ARM A-Profile (AArch64 bare-metal)
Toolchains Version	13.3.1.arm-13-24	13.2.1.20231009
Debugger	J-Link ARM	

Note:

When creating a CMake project, specify **"e²studio CMake"** as the **IDE Project Type**.

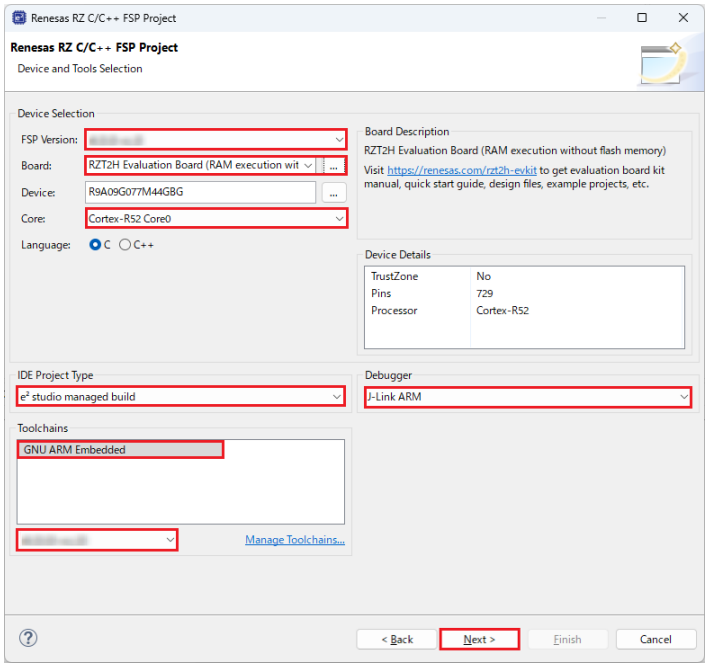


Figure 71. e²studio Project Configuration Window (2/6)

12. A bundle file selection screen will appear. Make the following settings:
    - (single-core processing or Multiprocessing Primary) Select **None** and click **Next.**

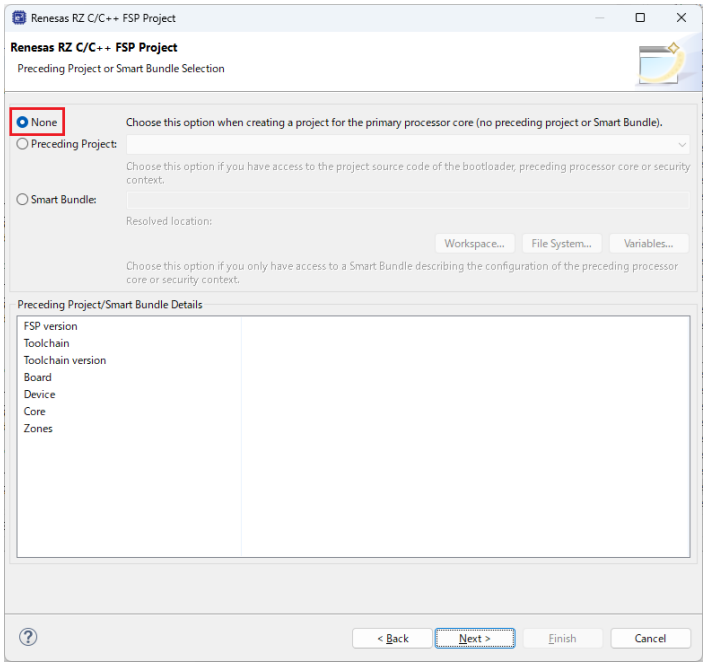


Figure 72. e²studio Project Configuration Window (3/6)

- (Multiprocessing Secondary) Select a bundle file. For the secondary or later project of multiprocessing, select the primary project as the **Preceding Project**. Built primary project names in the same workspace appear as an option in the drop-down list.

Notes:

1. 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).
2. 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.

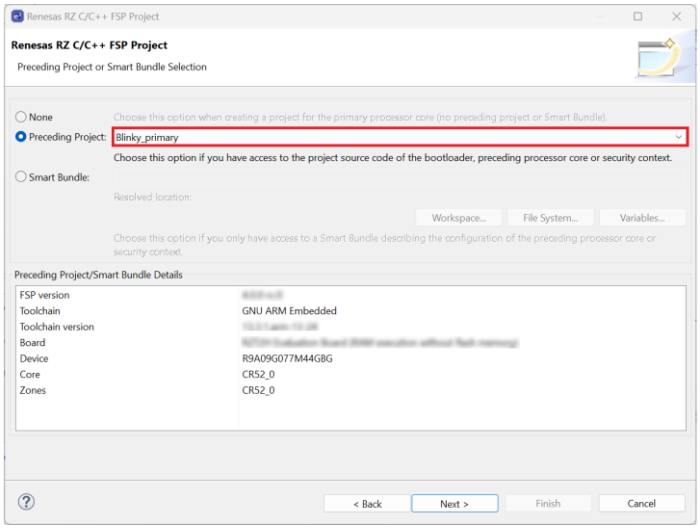


Figure 73. e²studio Project Configuration Window (4/6)

13. Select the **Build Artifact** and **RTOS**.

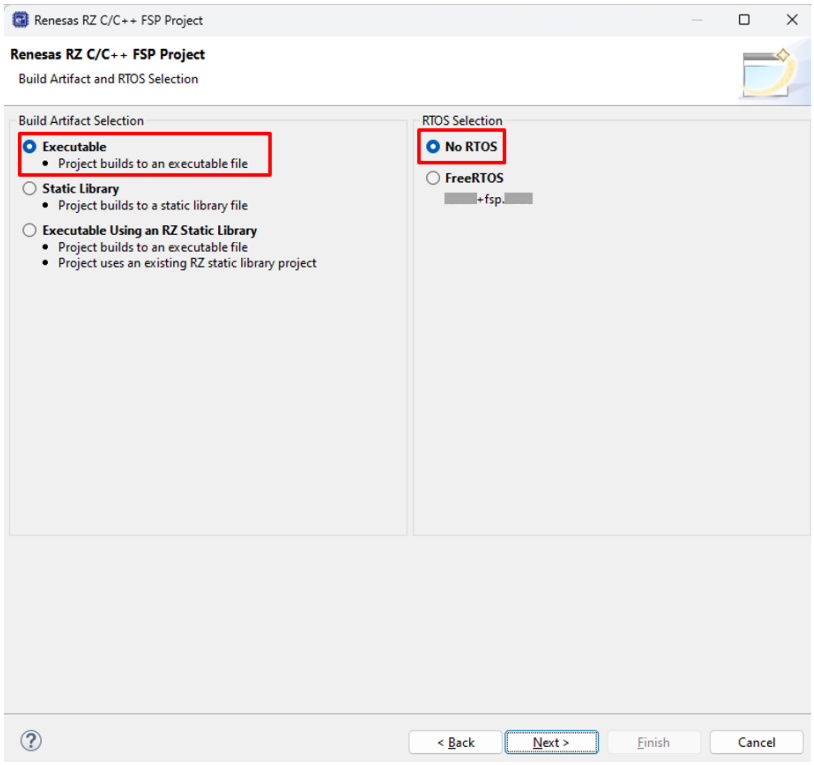


Figure 74. e²studio Project Configuration Window (5/6)

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

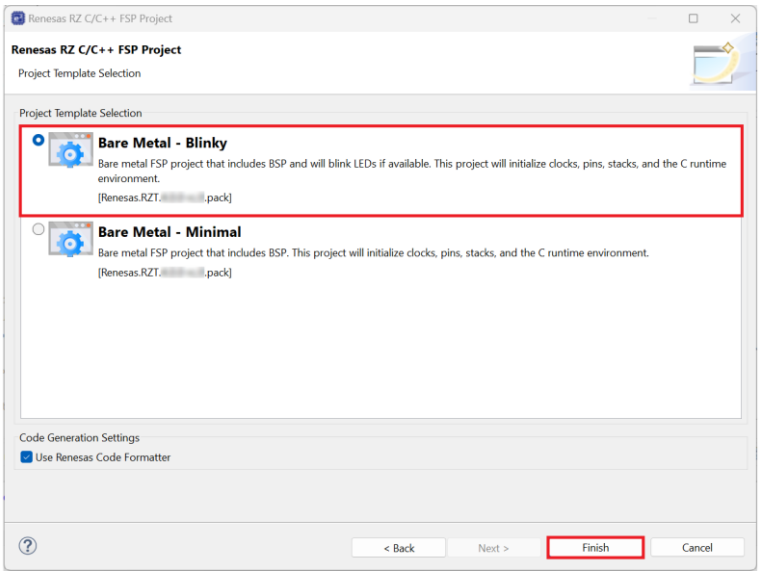


Figure 75. e²studio Project Configuration Window (6/6)

Once the project has been created, the name of the project will show up in the **Project Explorer** window of e²studio.

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

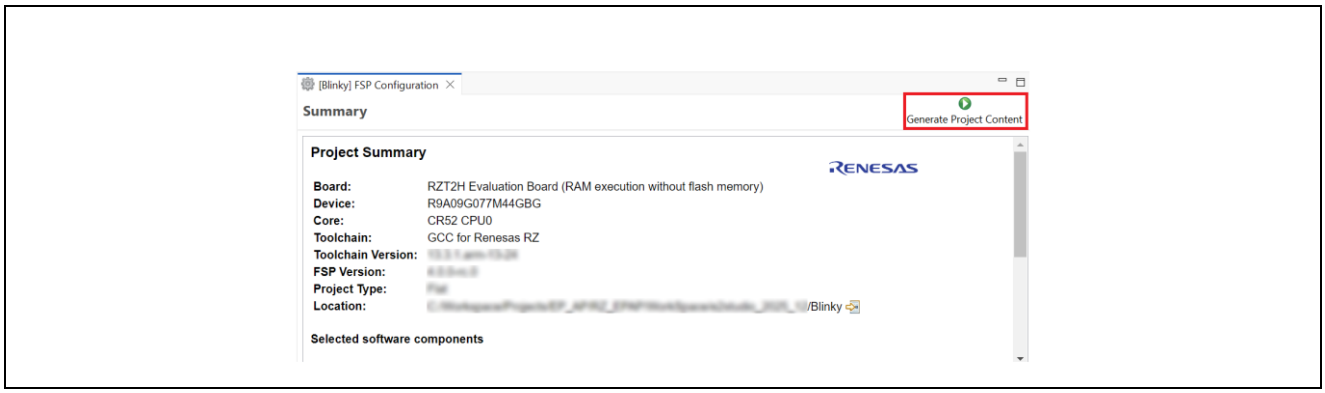


Figure 76. e²studio Project Configuration Tab

16. (xSPI, NOR flash, eSD, eMMC boot modes) Insert the loop part in *hal_entry.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 e²studio, 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 the R_BSP_WarmStart_StackLess() function
CR52	xSPI NOR flash	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"); ……
CA55	xSPI	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) __asm volatile ("BX lr"); ……
-	eSD eMMC	Since eSD and eMMC boot modes require different numbers of loops, 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 the default 75,000,000 (0x047868bf) to 150,000,000, which is twice the number of loops

1. Convert the loop count from decimal to hexadecimal: 150000000d − 1 = 149999999d = 0x08f0 d17f
2. Replace the operand of movw* with the lower 16-bit value: movw r1, #0xd17f
3. Replace the operand of movt* with the upper 16-bit value: movt r1, #0x8f0

* In the CA55 core, opcodes are different.

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^{2}studio for the Blinky application.

The clock configuration tab (see section *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^{2}studio for the Blinky application.

The pin configuration tab shows the pin configuration for the Blinky application (see section *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 section *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 section *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^{2}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 execute the **R_BSP_PinToggle**() API in #3 again.

4.4 Build the Blinky Project

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**.

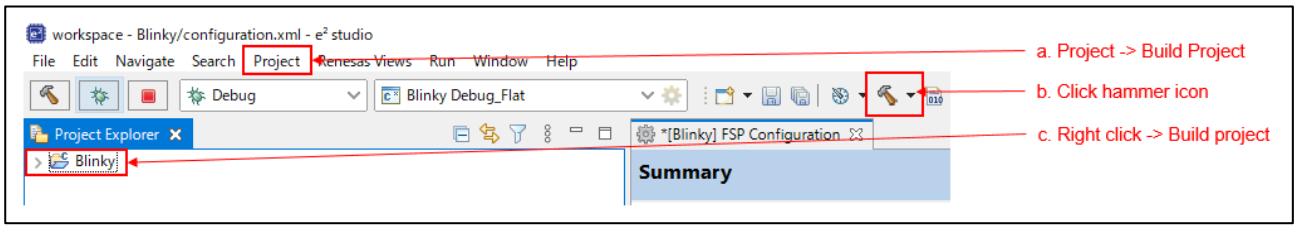


Figure 77. e²studio Project Explorer Window

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

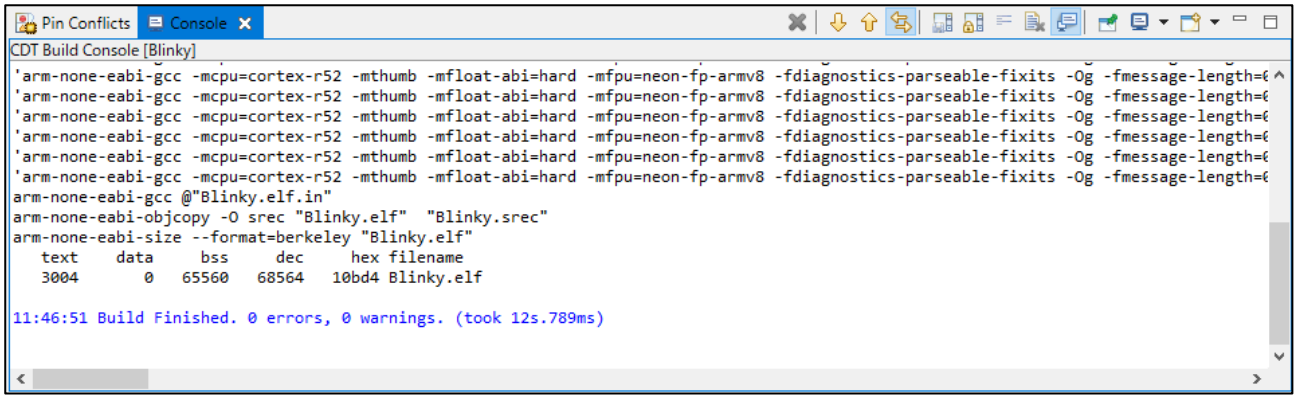


Figure 78. e²studio Project Build Console

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 e²studio.
- 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 debugger.

The Evaluation Board supports debugging via the on-board J-Link OB (connected through a USB cable) or via an external debugger (such as J-Link or I-Jet) connected to the JTAG header.

4.5.2 Debug Steps

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

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

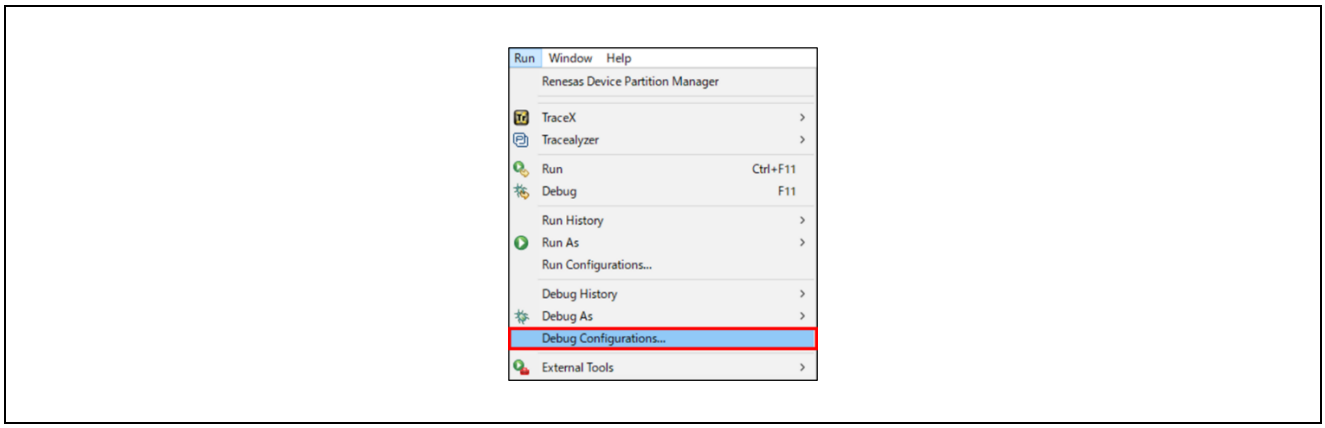


Figure 79. e²studio Debug Configurations Selection Option

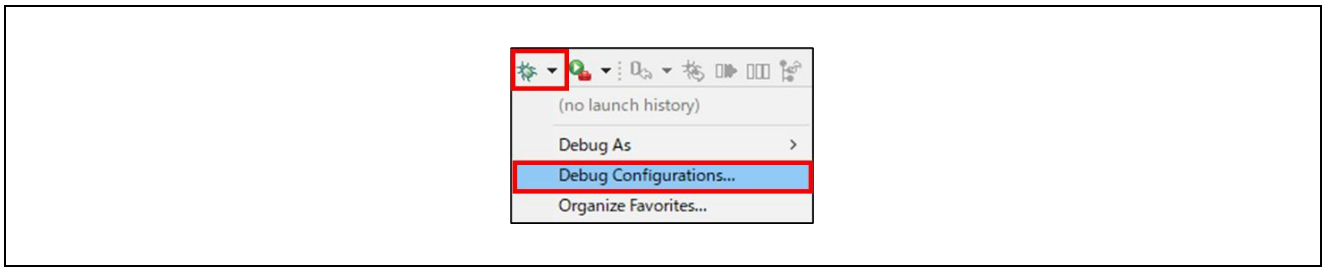


Figure 80. e²studio Debug Icon

2. Select your debugger 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.

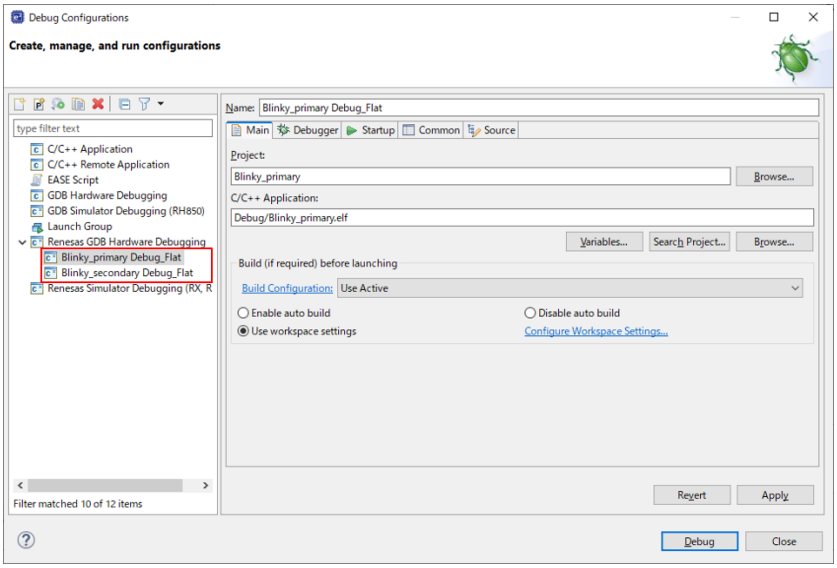


Figure 81. e²studio Debug Configurations Window with Blinky Project

3. Configuring the following additional debug settings by clicking **Debugger** > **Connection Settings** > **Connection**.

Table 10. e²studio Newly Created Project Debug Settings

Setting	Primary			Secondary or later
	RAM execution		Flash boot mode	
	CR52	CA55	All Cores	All Cores
Reset at the beginning of connection	Yes	Yes	Yes	**No**
Reset after download	No	No	**Yes**	No
Set CPSR(5bit) after download	**Yes**	No	No	No

Note:

Each setting item represents the following:

- Reset at the beginning of connection: Configures whether to reset when a debugger is connected.
- Reset after download: Configures whether to reset after the program download.
- Set CPSR(5bit) after download: Configures whether to set the 5th bit of CPSR after the program download.

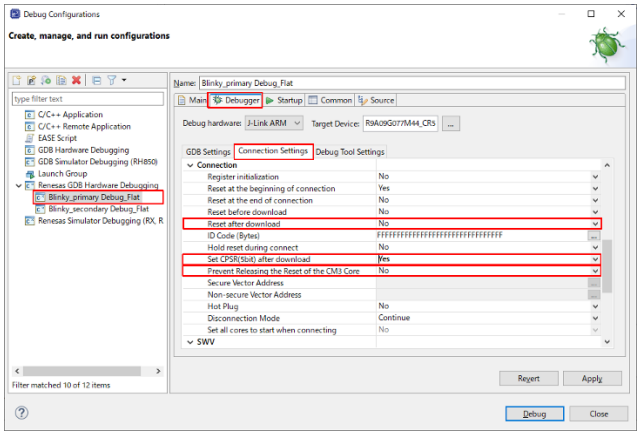


Figure 82. e²studio Debug Configurations Window with Blinky Project

4. (RZ/T2H CR52 Core1, RZ/T2N CR52 Core1, or RZ/N2H CR52 Core1) Set the script file for TCM initialization to clear ATCM and BTCM to 0. and enable them.
    1. **Debugger** > **Connection Settings** > **J-Link**
    2. **Script File** : ${workspace_loc:/${ProjName}}/script/initialization_TCM.JLinkScript

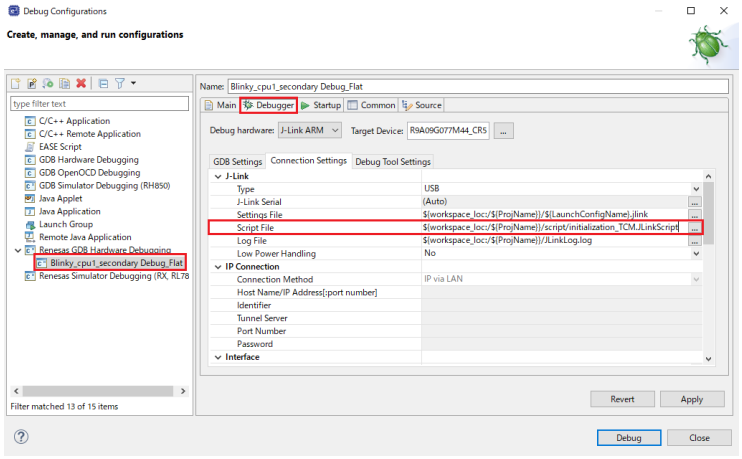


Figure 83. e²studio Debug Configurations Window with Blinky Project (RZ/T2H CR52 Core1, RZ/T2N CR52 Core1, or RZ/N2H CR52 Core1)

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

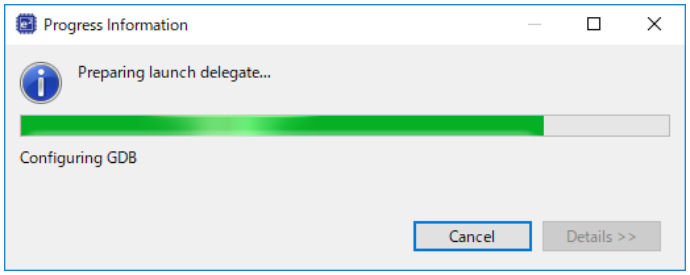


Figure 84. Start Debugging

4.5.3 Details about the Debug Process

In debug mode, e²studio 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 **system_init()**.
5. Displaying the startup code where the program counter points to.

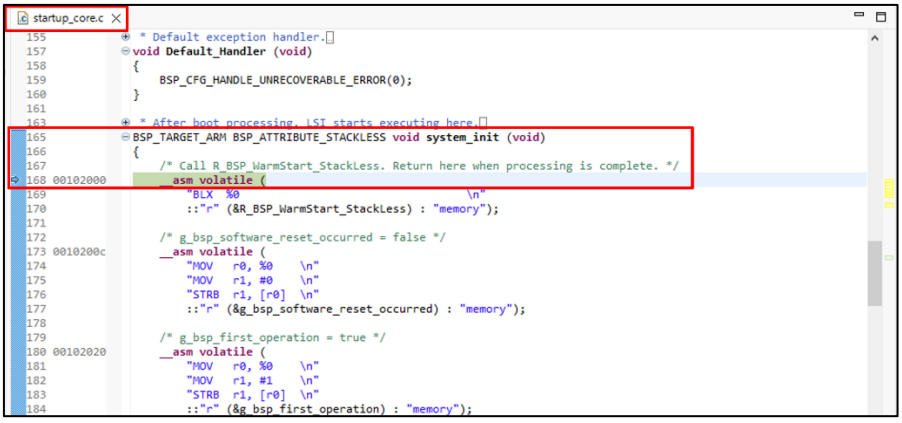


Figure 85. e²studio Debugger Memory Window

4.6 Run the Blinky Project

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



Figure 86. e²studio Debugger Play Icon

The following LEDs on the board should now be blinking.

- RZ/T series
  - RSK+RZ/T2M: LED0-1 (Core0), LED2-3 (Core1)
  - RSK+RZ/T2L: LED0-6 (including LEDx_ESC_xxx)
  - RSK+RZ/T2ME: LED0-1 (Core0), LED2-3 (Core1)
  - RZ/T2H Evaluation Board: LED0 (CR52 Core0), LED1 (CR52 Core1), LED2 (CA55 Core0), LED3 (CA55 Core1), LED4 (CA55 Core2), LED5 (CA55 Core3)
  - RZ/T2N Evaluation Board: LED0 (CR52 Core0), LED1 (CR52 Core1), LED2 (CA55 Core0), LED3 (CA55 Core1)
- RZ/N series
  - RSK+RZ/N2L: LED0-3
  - RZ/N2H Evaluation Board: LED3 (CR52 Core0), LED4 (CR52 Core1), LED8 (CA55 Core0), LED9 (CA55 Core1), LED10 (CA55 Core2), LED11 (CA55 Core3)

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

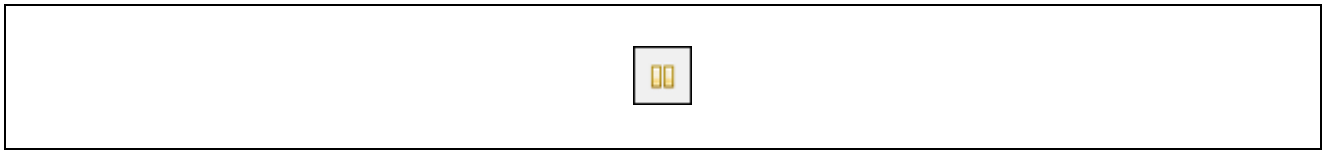


Figure 87. e²studio Debugger Pause Icon

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

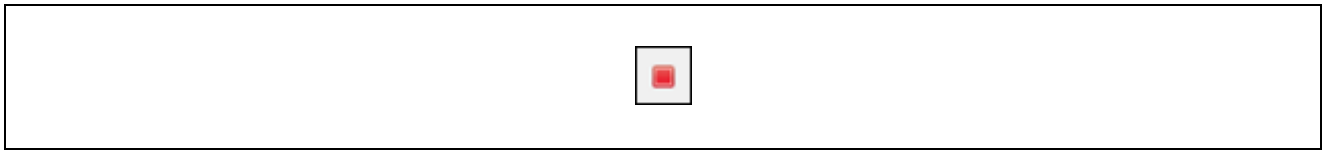


Figure 88. e²studio Debugger Stop Icon

4.7 Import the Project

The project created, built, and debugged in section *4.3 Create a New Project for Blinky* through section *4.6 Run the Blinky Project* 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**.

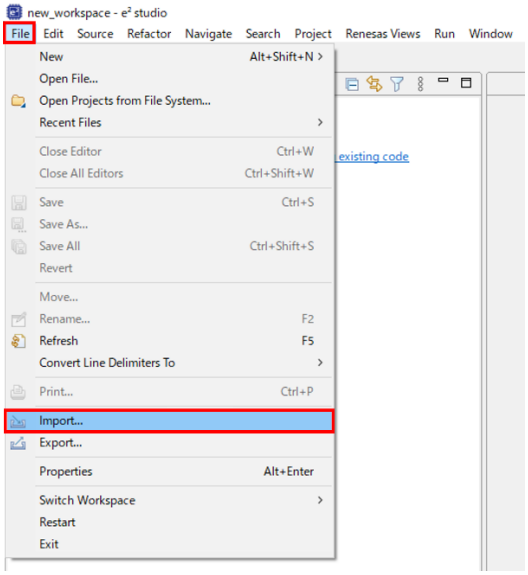


Figure 89. e²studio Import

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

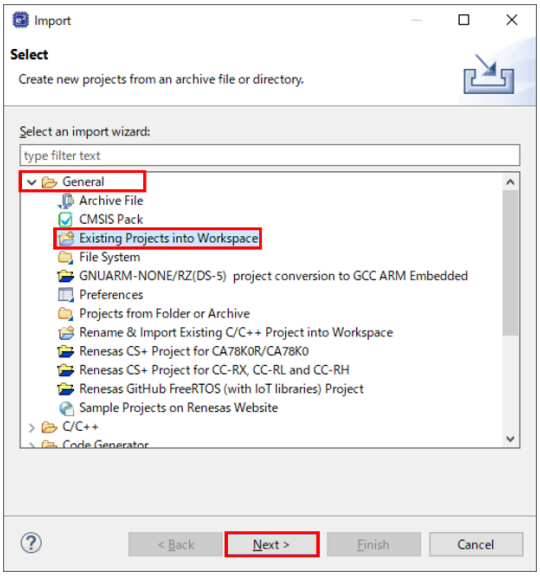


Figure 90. e²studio Select Import Type

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.

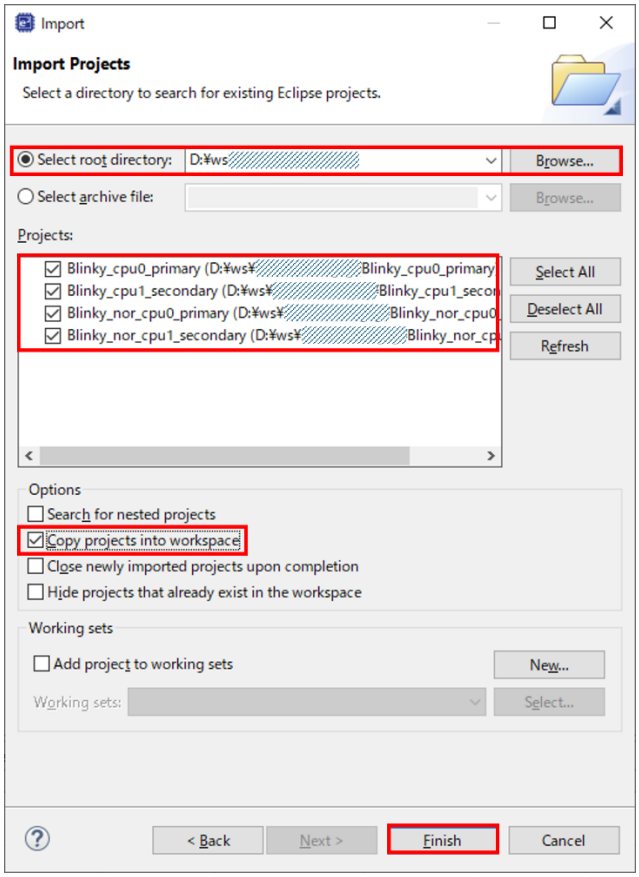


Figure 91. e²studio Select Root Directory to Import Project

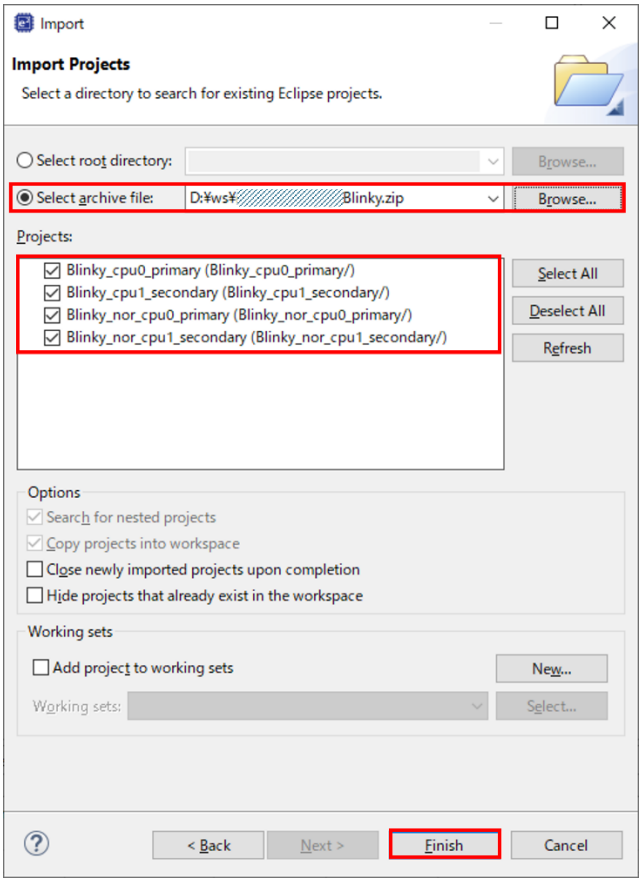


Figure 92. e²studio Select Archive File to Import Project

5. The projects have been imported into the other workspace.

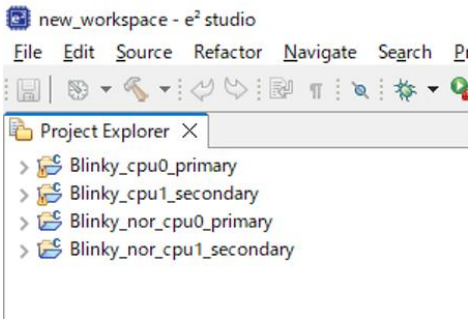


Figure 93. e²studio New Workspace

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 third-party IDE and toolchain.

For creating the RZ/T and RZ/N 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^{2}studio. Refer to section *6.3 Configuring a Project* 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 a third-party IDE and running that application on an RZ/T, RZ/N MPU board. This chapter guides you through creating projects for single-core processing.

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/T, RZ/N devices. These can be set up as bare metal designs within IAR EWARM. However, most RZ/T, RZ/N developers will want to integrate RZ/T, RZ/N 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.
  - Refer to the IAR Systems website regarding IAR EWARM.
- FSP SC and FSP Pack installed.
  - Refer to the Renesas website regarding FSP SC and FSP Pack.

Note for RZ/T2ME:

If the user uses the IAR EWARM 9.60.1 or 9.60.2 to debug the RZ/T2ME FSP project, 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, 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, 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, read the README in the patch.

Note for RZ/T2N:

If you use IAR EWARM to debug an RZ/T2N FSP project, please apply the following patch file.

- EWARM_Patch_for_RZT2N (EWARM_Patch_for_RZT2N_rev1.0.zip)
  - This patch file is available in http://www.renesas.com/rzt2n.

Regarding how to apply the patch, please read the readme file in patch file.

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 examples of projects that run on a single core. Refer to *Table 11. Project Creation Procedure (IAR EWARM, FSP SC)* to find out which steps are required for your application.

Table 11. Project Creation Procedure (IAR EWARM, FSP SC)

<table>
<tr><th rowspan="3">Step</th><th colspan="3">Single-core processing</th><th rowspan="3">Multiprocessing</th></tr>
<tr><th rowspan="2">RAM execution</th><th colspan="2">Flash boot mode</th></tr>
<tr><th>xSPI
NOR flash</th><th>eSD
eMMC</th></tr>
<tr><td>Check tool limitations</td><td colspan="4">エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。</td></tr>
<tr><td>Erase flash memory (if needed)</td><td colspan="4">8.1 Appendix
How to Erase Flash Memory</td></tr>
<tr><td>Create a project</td><td colspan="3">5.3.2 Create a New Project</td><td rowspan="4">11 Appendix. How to Create and Debug FSP Projects for Multiprocessing for IAR EWARM</td></tr>
<tr><td>Build the project</td><td colspan="3">5.3.3 Build the Project</td></tr>
<tr><td>Debug the project</td><td rowspan="2">5.3.4 Download & Debug the Project</td><td rowspan="2">RZ FSP Documentation
eSD boot (RZT, RZN)
eMMC boot (RZT, RZN)</td></tr>
<tr><td>Run the project</td></tr>
</table>

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 this section, using the IDE (e^{2}studio or IAR EWARM). For eSD and eMMC boot modes, tools other than the IDE are required when programming to external flash memory devices. Refer to the FSP Documentation for details on creating image files, the programming process, and debugging procedures.

1. Start the FSP SC.
   - FSP SC is installed in the following path by default.
     - For RZ/T, RZ/N series, it is installed in C:\Renesas\rz\sc_vYYYY-MM_fsp_vX.X.X\eclipse\rasc.exe

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

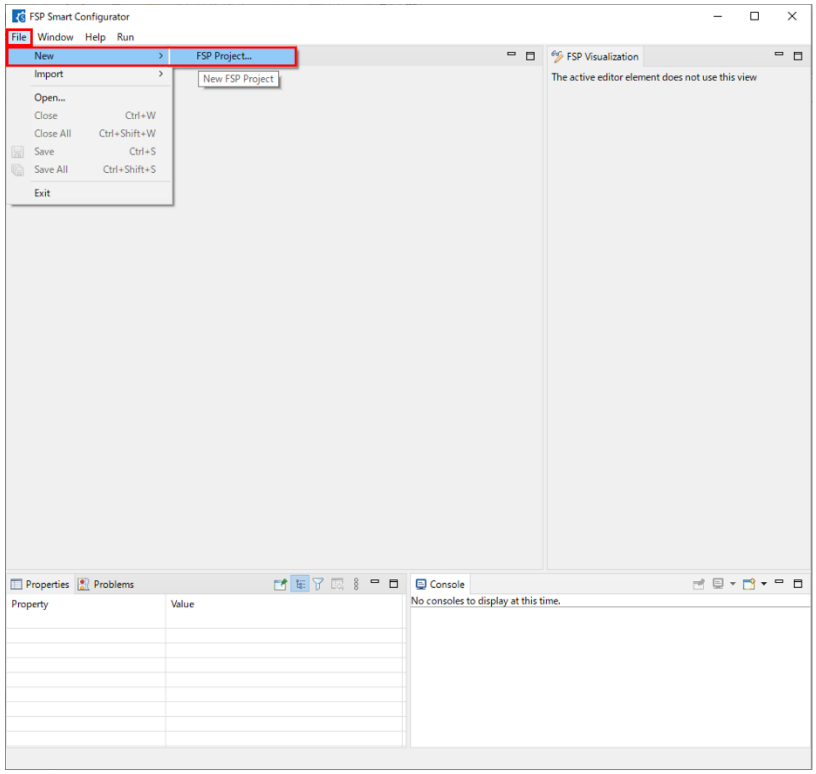


Figure 94. FSP SC New Project

3. Enter a project folder and project name. Blinky is a good name to use for this tutorial.
4. Click **Next**. The Project Configuration window shows your selection.

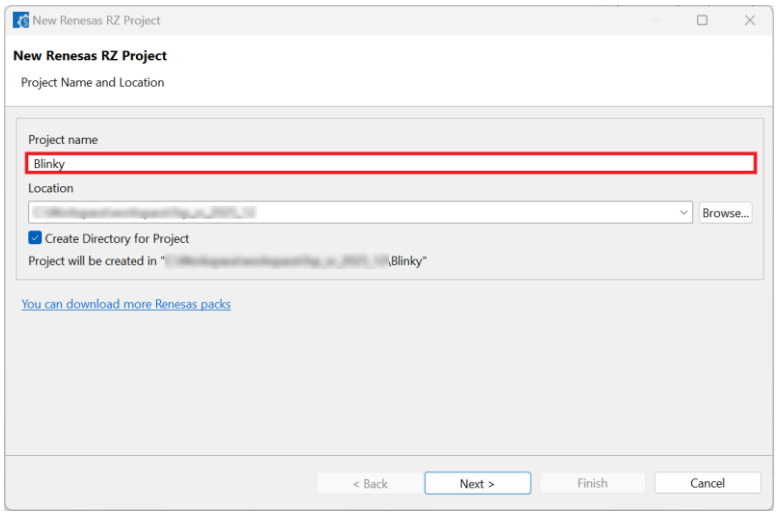


Figure 95. FSP SC Project Settings

5. Select the **FSP** version from the drop-down list.
6. Select the **Board** for your application from the drop-down list.
   - You can select an existing RZ/T, RZ/N MPU Evaluation Board or select a Custom User Board for any of the RZ/T, RZ/N MPU devices with your own BSP definition.
   - Here, select either of the following boards to create an FSP project for the Evaluation board.
7. Select the **Core** from the drop-down list.
   - Only **Cortex-R52 Core0** or **Cortex-A55 Core0** can be designated as the Primary project.
8. Select **IDE Project Type** from the drop-down list.
   - As the Toolchain, the IAR Toolchain for ARM is preselected.
9. Select the **Toolchains**, then click **Next**.

Table 12. FSP SC Newly Created Project Settings

Setting	Value
FSP Version	(Select the FSP version to be used.)
Board	(Select the combination of the evaluation board and boot mode to be used.)
Core	Cortex-R52 Core0, Cortex-R52 Core1, Cortex-A55 Core0, Cortex-A55 Core1, Cortex-A55 Core2 or Cortex-A55 Core3
IDE Project Type	**IAR EWARM [v9.60-9.70]**
Toolchains	IAR toolchain for ARM

Note:

When creating a CMake project, change the settings as follows:

- **IDE Project Type**: CMake
- **Toolchains**:
  - IAR toolchain for ARM
  - (Cortex-R52) GCC toolchain for ARM
  - (Cortex-A55) GCC ARM A-Profile Toolchain for ARM

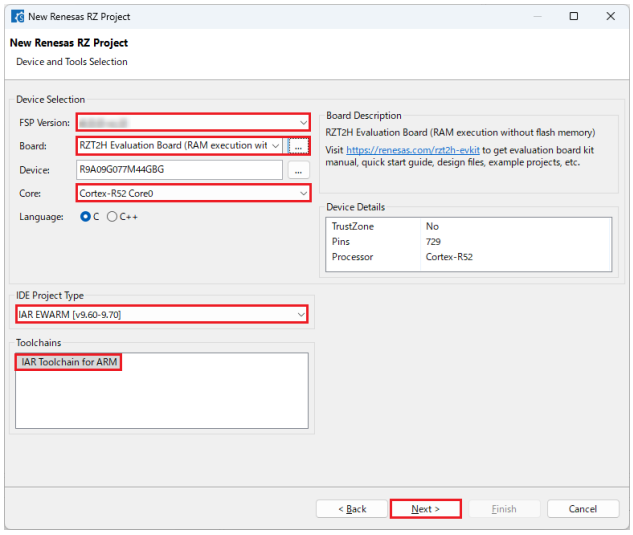


Figure 96. Target Device and IDE Selections

10. A bundle file selection screen will appear. Make the following settings:
    - (single-core processing or Multiprocessing Primary) Select **None** and click **Next**.

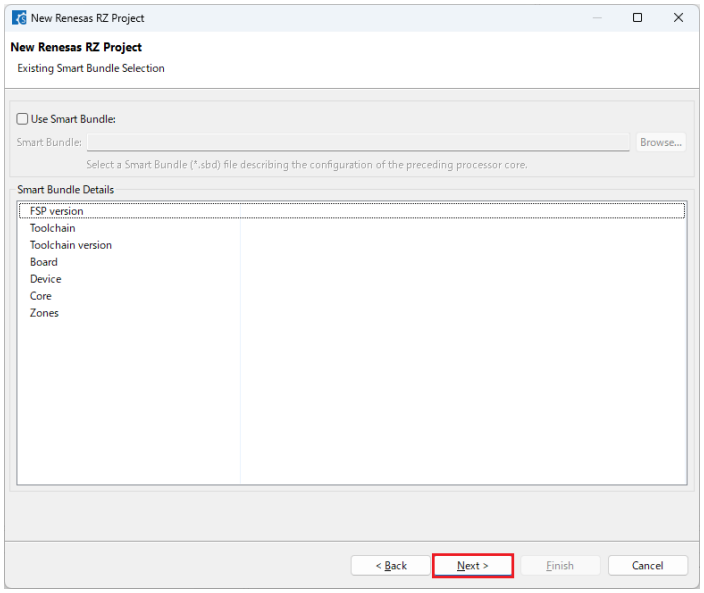


Figure 97. Bundle File Selection (not Use Smart Bundle)

- (Multiprocessing Secondary) Select a bundle file. For the secondary or later project of multiprocessing, select the file of the primary project. The file is generated in the [Build Configuration]/Exe directory of the project after building the 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).

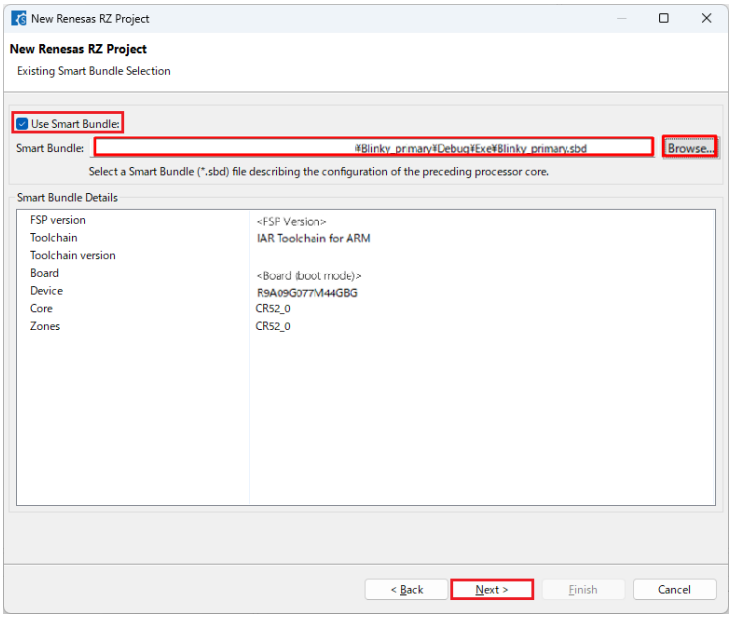


Figure 98. Bundle File Selection (Use Smart Bundle)

11. Select **RTOS**.
    - Here, select **No RTOS** to proceed with the following tutorial.
12. Click **Next**.

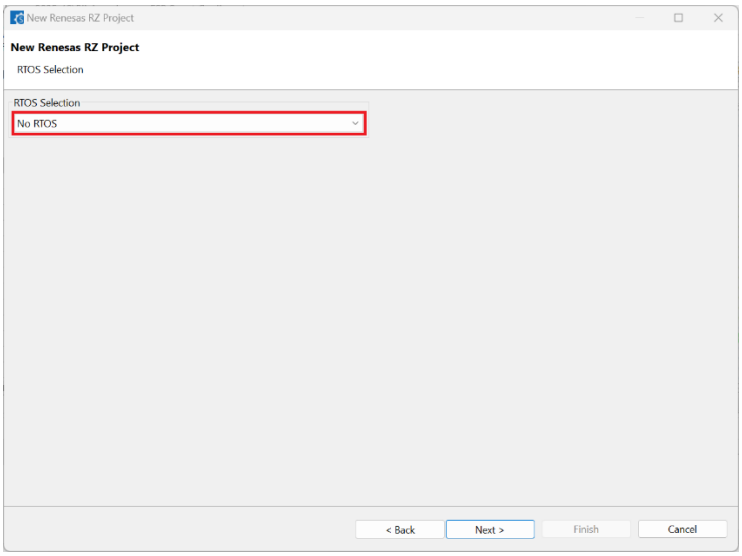


Figure 99. RTOS Selection

13. Select a **project template** from the list of available templates.
    - By default, this screen shows the templates that are included in your current RZ 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**.
14. Click **Finish.**

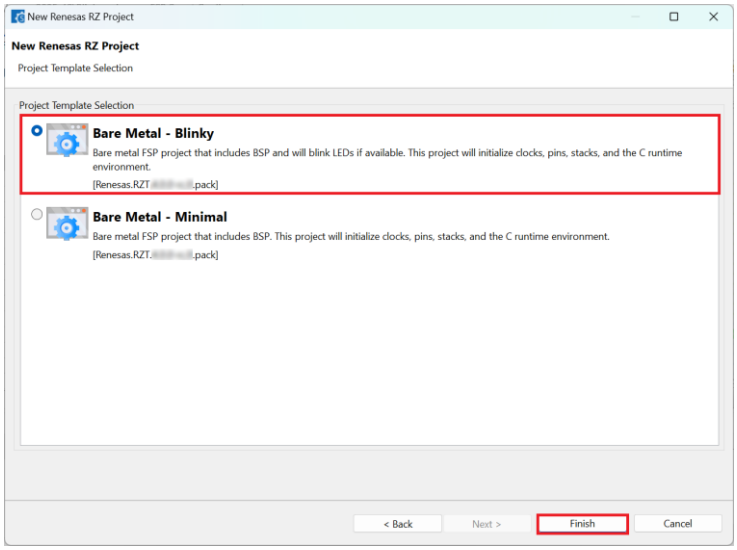


Figure 100. Template Selection

15. **Configure** the FSP configuration by referring to section *6.3 Configuring a Project*.
    - Here, skip this configuration step to proceed with the following tutorial.
16. On completion of the FSP configuration, click **Generate Project Content**.
    - A new IAR EWARM project file will be generated in the project path.

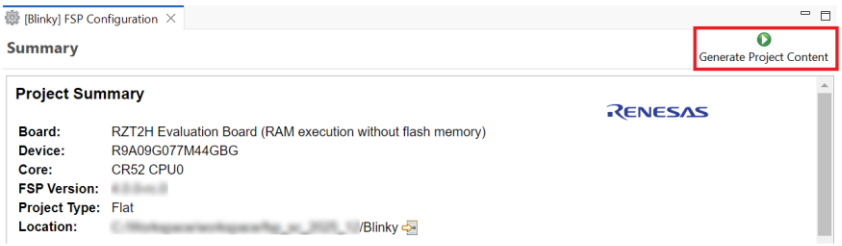


Figure 101. FSP Project Configuration and Generation

17. (xspi, NOR flash, eSD, eMMC boot modes) Insert the loop part in *hal_entry.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^{2}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 the R_BSP_WarmStart_StackLess() function
CR52	xSPI NOR flash	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"); ……
CA55	xSPI	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) __asm volatile ("BX lr"); ……
-	eSD eMMC	Since eSD and eMMC boot modes require different numbers of loops, refer to FSP Documentation.

Note:

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 the default 75,000,000 (0x047868bf) to 150,000,000, which is twice the number of loops

1. Convert the loop count from decimal to hexadecimal: 150000000d − 1 = 149999999d = 0x08f0 d17f
2. Replace the operand of movw* with the lower 16-bit value: movw r1, #0xd17f
3. Replace the operand of movt* with the upper 16-bit value: movt r1, #0x8f0

* In the CA55 core, opcodes are different.

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

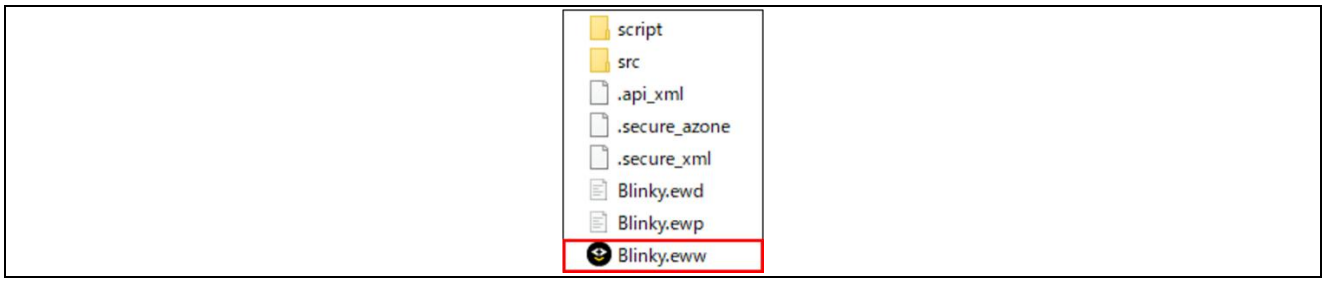


Figure 102. FSP Project Workspace

5.3.2.1 Configure IAR EWARM Project

1. Click on **Project** and then click on **Option… to open the project options** window.
2. Click **Debugger** > **Setup** and select **I-Jet** from the dropdown menu under **Driver**.

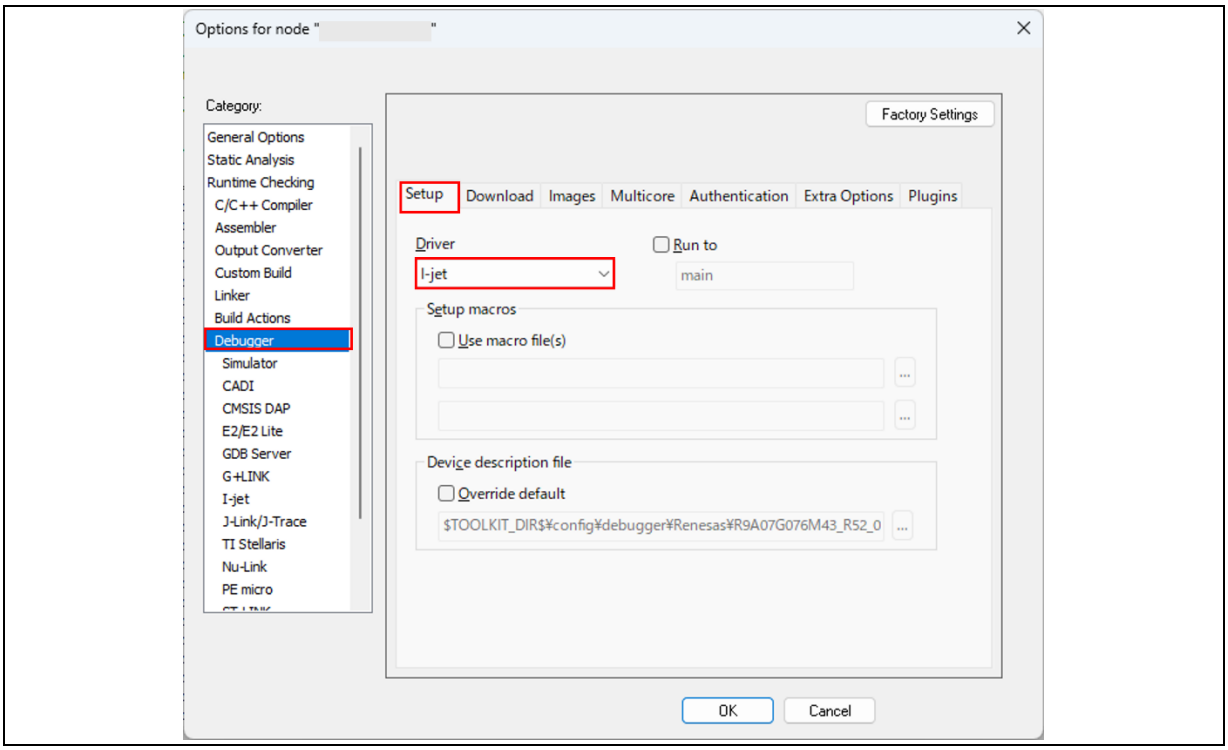


Figure 103. IAR EWARM Debugger Setup

3. Click **Debugger** > **Setup** and uncheck **“Run to"**.

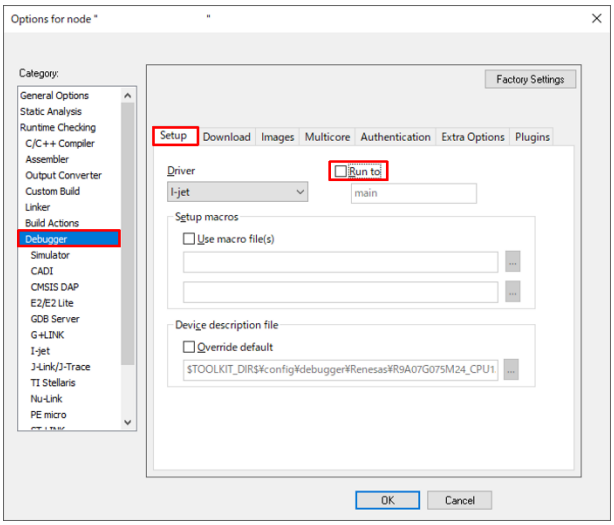


Figure 104. IAR EWARM Project Options (Run to)

5.3.2.2 Configure IAR EWARM Project [RZ/T2H, RZ/T2N and RZ/N2H Only]

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

Table 13 FSP SC Newly Created Project Debug Settings

Device name	CR52 Core0	CR52 Core1	CA55 Core0, 1, 2, 3
RZ/T2H	R9A09G077M44_R52_0	R9A09G077M44_R52_1	R9A09G077M44_A55
RZ/N2H	R9A09G087M44_R52_0	R9A09G087M44_R52_1	R9A09G087M44_A55
RZ/T2N	R9A07G076M48_R52_0	R9A07G076M48_R52_1	R9A07G076M48_A55

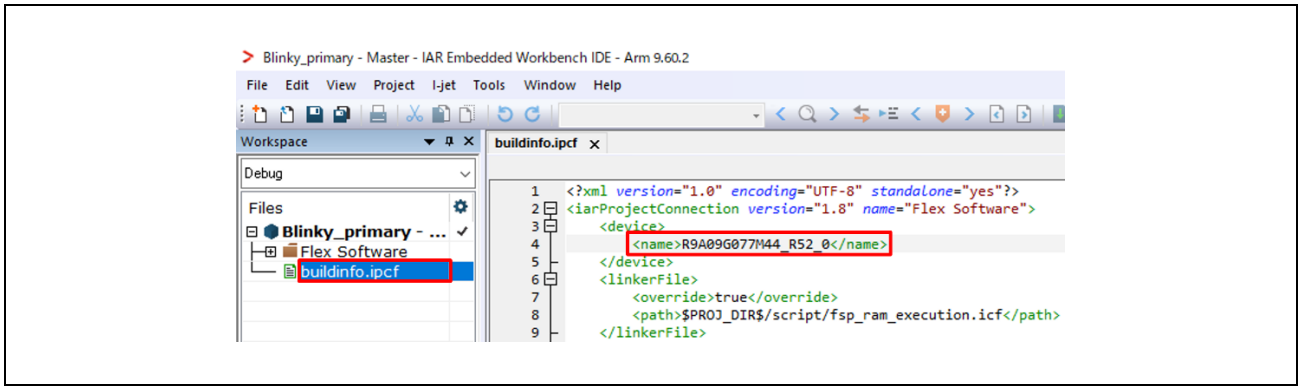


Figure 105. IAR EWARM Project File (CR52)

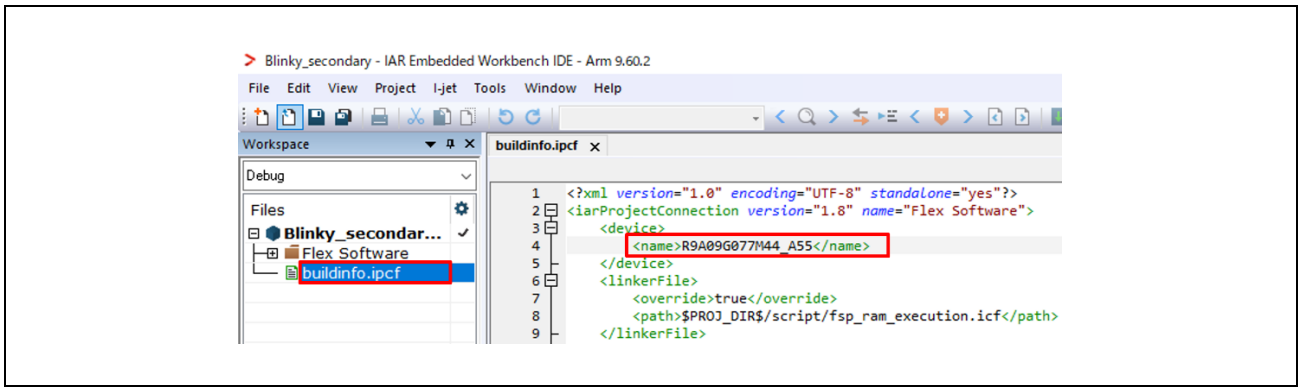


Figure 106. IAR EWARM Project File (CA55)

2. Click on **Project** and then click on **Option…** to open the project option window.
3. Click **General Options** > Target and confirm that the name changed in step 1 appears in the Device of the Processor variant.

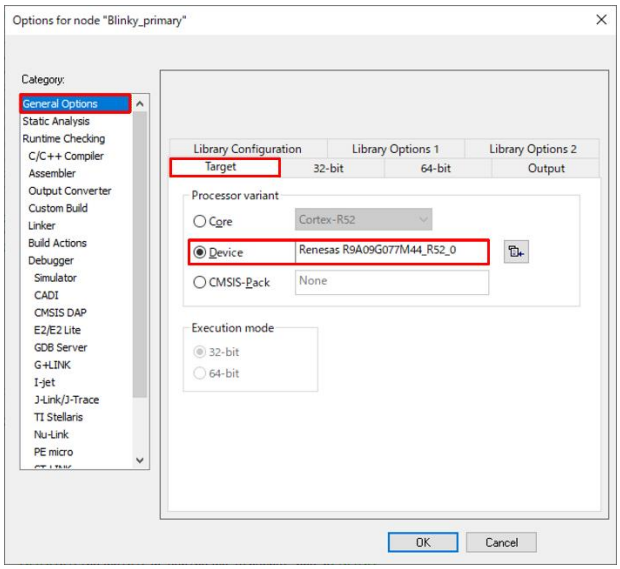


Figure 107. Project Options – Device (CR52 Core0)

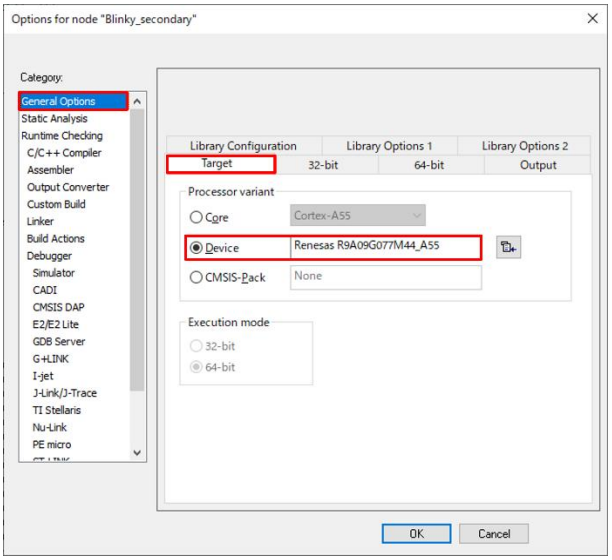


Figure 108. Project Options – Device (CA55)

Notes:

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 3 is maintained and does not need to be modified again in *buildinfo.ipcf*. Build the project as is.

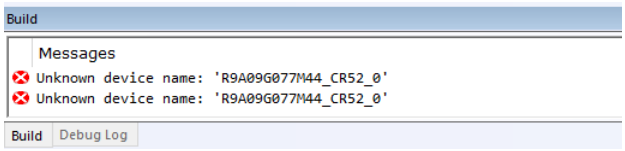


Figure 109. Error Message after Configuration Change

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

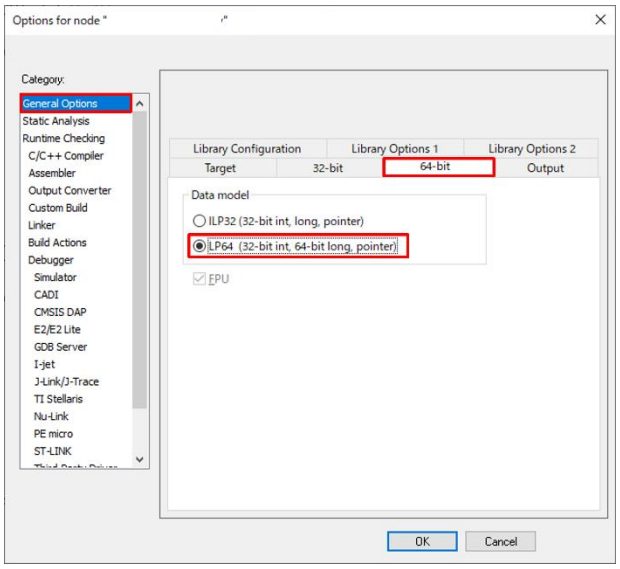


Figure 110. Project Options – Data Model

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

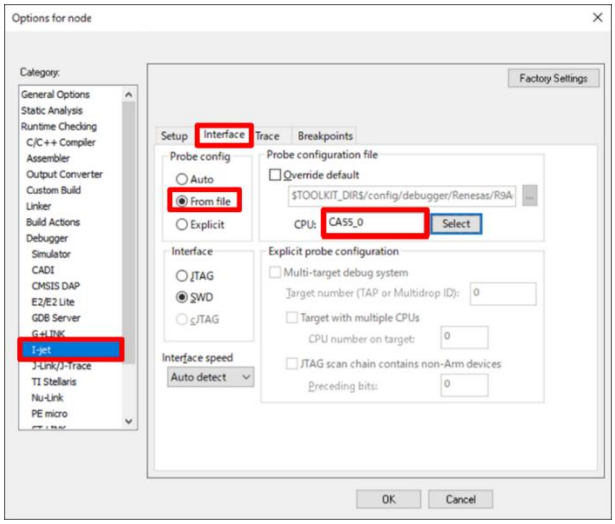


Figure 111. IAR EWARM Project Options for the Primary Project (I-Jet Interface)

6. (RZ/T2H and xspi boot mode) Override the board files.

   When RZ/T2H is 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.

   Reselect the execution path by following the steps below:

   i. Click on **Project** and then click on **Options…** to open the project options window.
   ii. Select the **Debugger category** and the **Download** Tab.
   iii. Enable "**Override default .board file.**"
   iv. Select the path to override the .board files for RZ/T2H
   - CR52 project:
     $TOOLKIT_DIR$\config\flashloader\Renesas\FlashRZT2H_EVB_R52.board
   - CA55 project:
     $TOOLKIT_DIR$\config\flashloader\Renesas\FlashRZT2H_EVB_A55.board

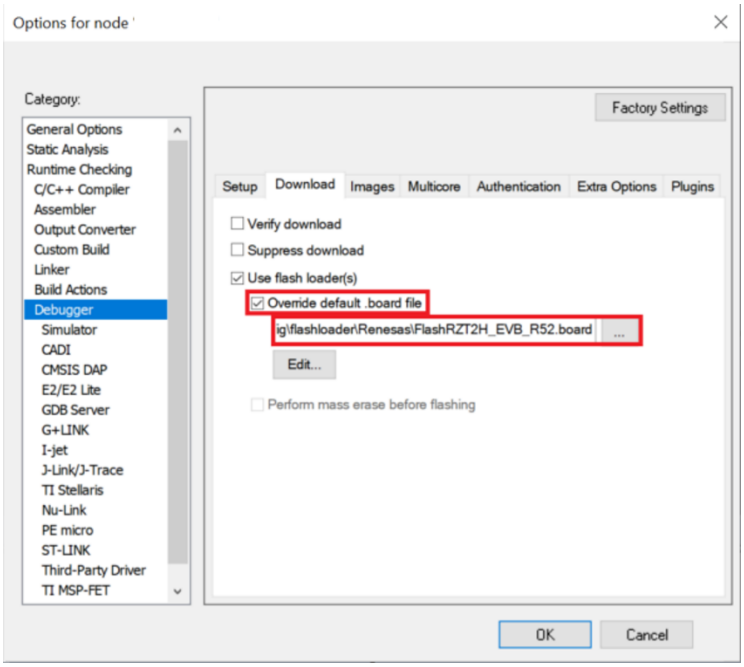


Figure 112. Project Options – Debugger (RZ/T2H Only)

5.3.3 Build the Project

- Single-core processing
  Click on **Project** -> **Make** from the menu bar or the **Make** button on the toolbar to build.

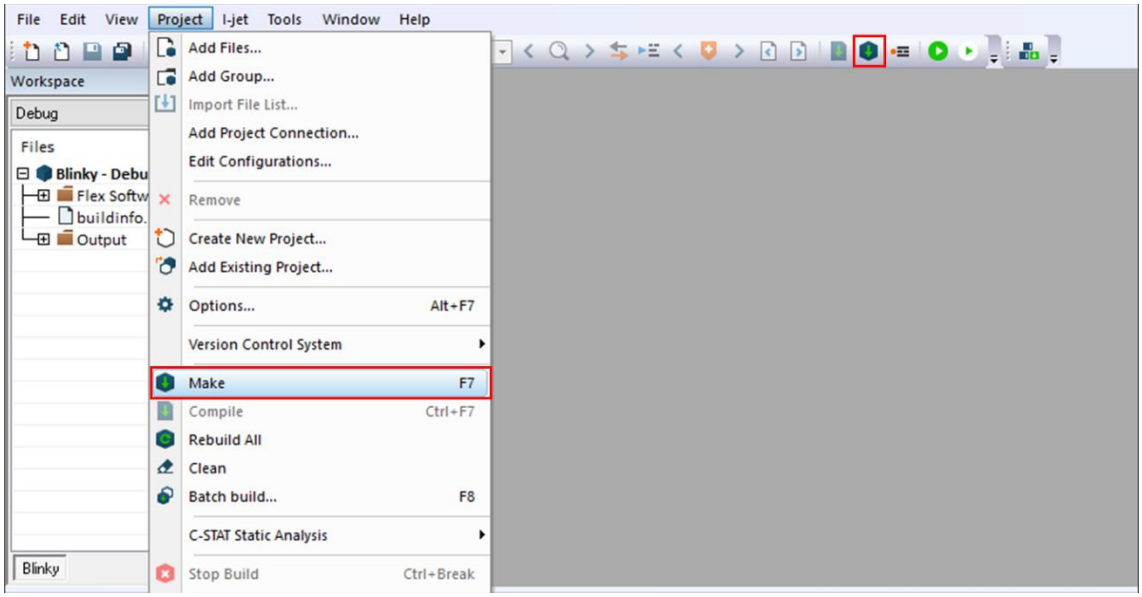


Figure 113. Make Button

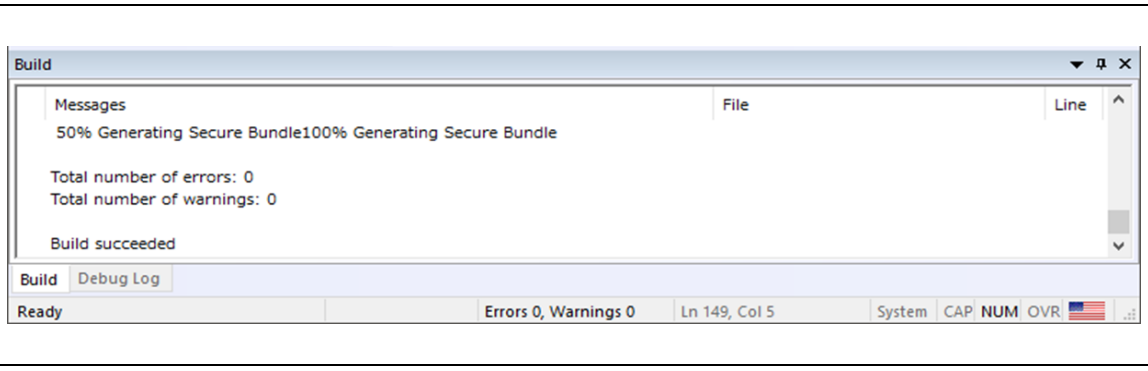


Figure 114. Build Message Console

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.

5.3.4 Download & Debug the Project

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

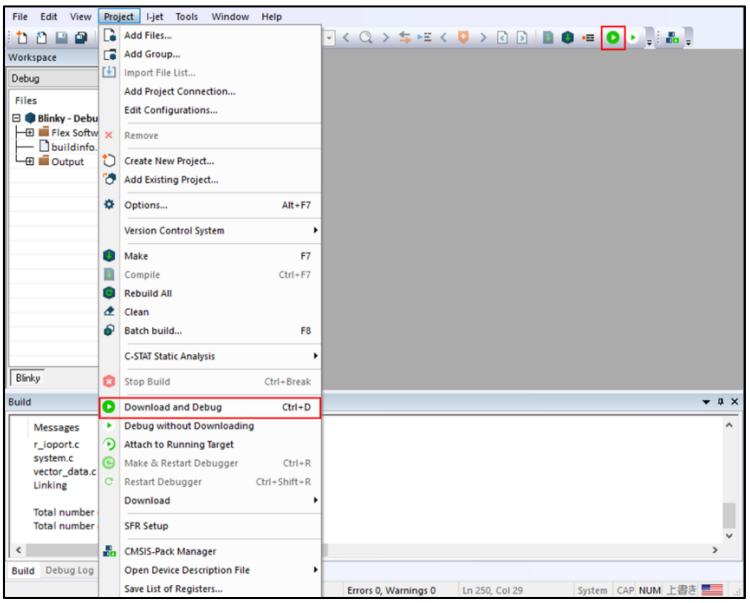


Figure 115. Download and Debug Button

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

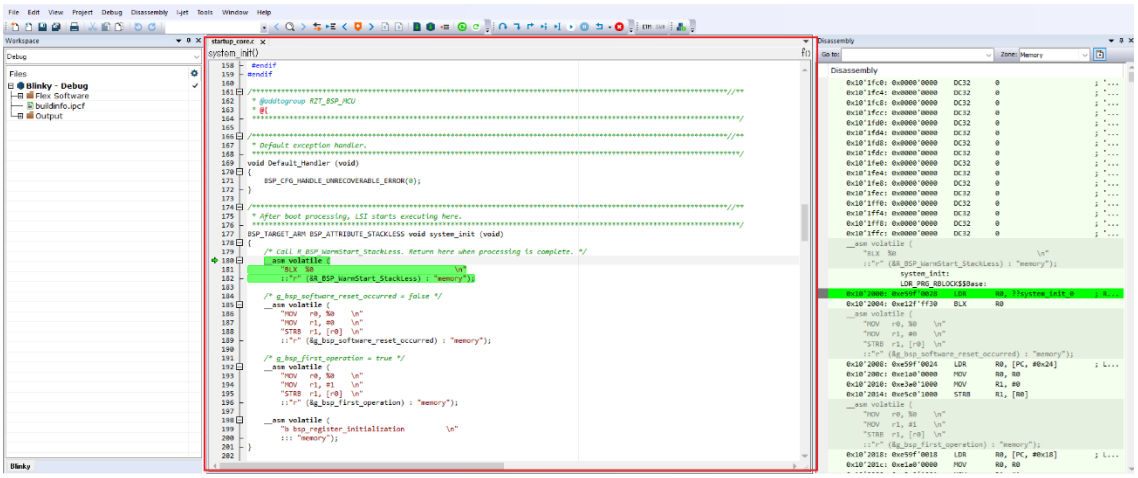


Figure 116. Starting Debug

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

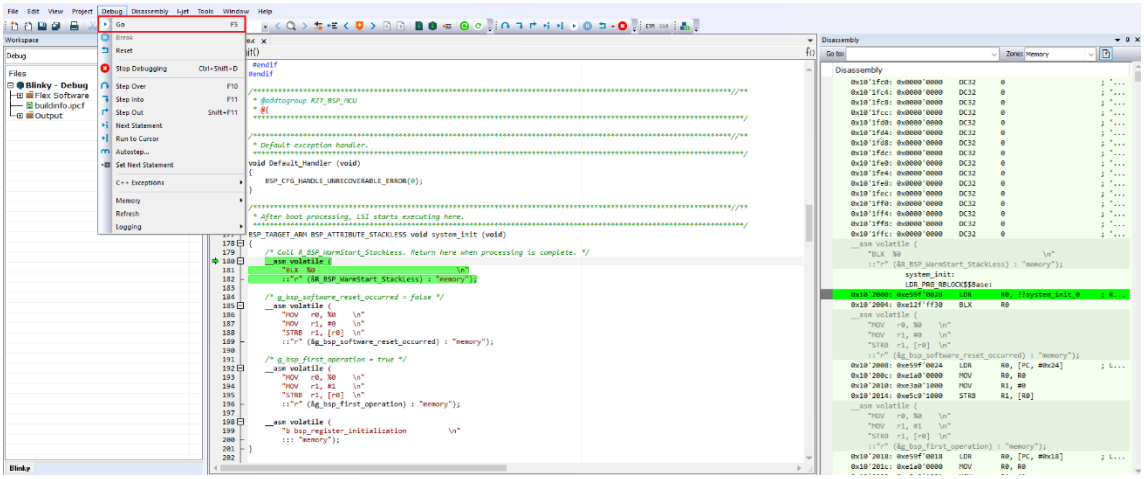


Figure 117. Go Button

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, **hal_entry.c** is registered under **Flex Software > Program Entry**.

The application performs the following steps:

1. Get the LED information for the selected board from the **bsp_leds_t** structure.
2. Initialize the output level for the 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 execute the **R_BSP_PinToggle**() API in #3 again.

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

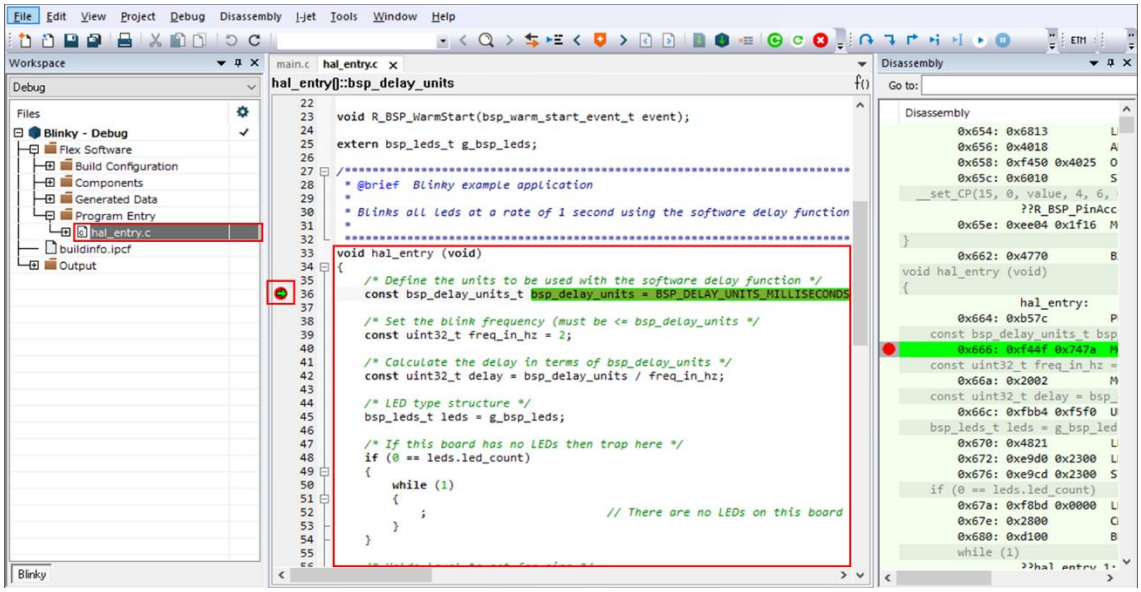


Figure 118. hal_entry.c and Setting Breakpoint

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

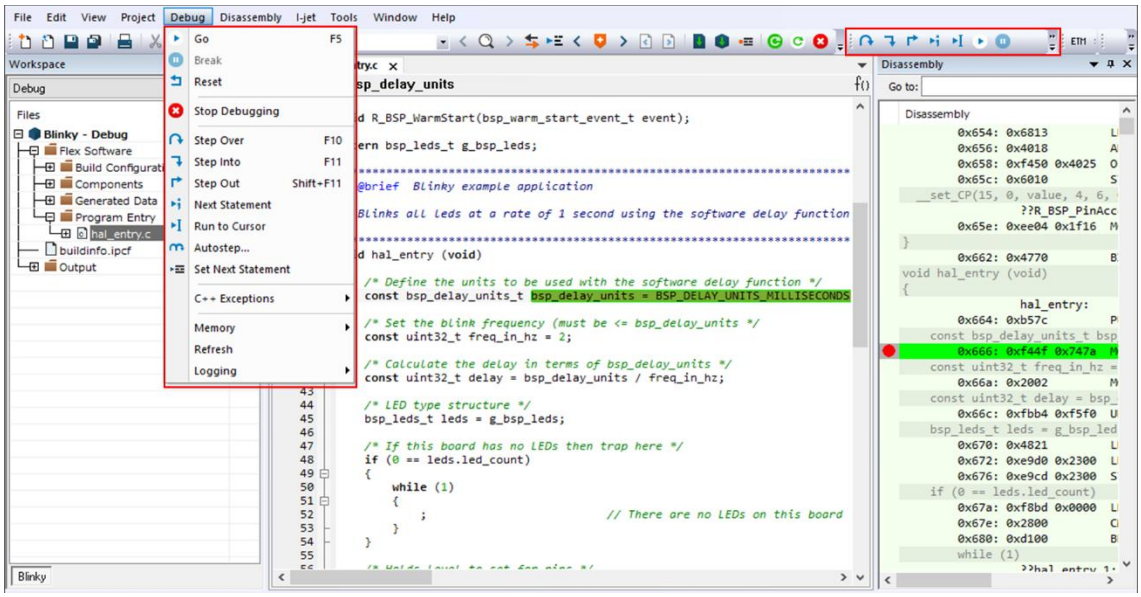


Figure 119. Debug Menu

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

RZ/T series:

- RSK+RZ/T2M: LED0-1 (CR52 Core0), LED2-3 (CR52 Core1)
- RSK+RZ/T2L: LED0-6 (including LEDx_ESC_xxx)
- RSK+RZ/T2ME: LED0-1 (CR52 Core0), LED2-3 (CR52 Core1)
- RZ/T2H Evaluation Board: LED0 (CR52 Core0), LED1 (CR52 Core1), LED2 (CA55 Core0), LED3 (CA55 Core1), LED4 (CA55 Core2), LED5 (CA55 Core3)
- RZ/T2N Evaluation Board: LED0 (CR52 Core0), LED1 (CR52 Core1), LED2 (CA55 Core0), LED3 (CA55 Core1)

RZ/N series:

- RSK+RZ/N2L: LED0-3
- RZ/N2H Evaluation Board: LED3 (CR52 Core0), LED4 (CR52 Core1), LED8 (CA55 Core0), LED9 (CA55 Core1), LED10 (CA55 Core2), LED11 (CA55 Core3)

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

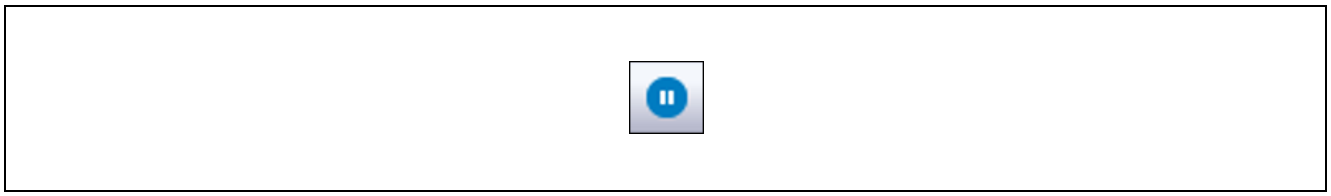


Figure 120. IAR EWARM Debugger Pause Icon

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

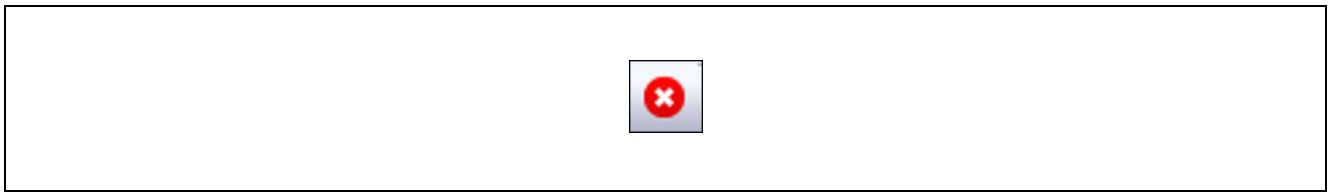


Figure 121. IAR EWARM Debugger Stop Icon

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: $PROJ_DIR$

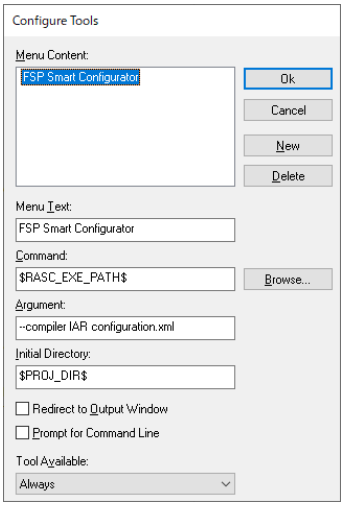


Figure 122. Settings to Launch FSP SC from IAR EWARM

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 created, built, and debugged in section *5.3.2 Create a New Project* through 5.3.4 Download & Debug the Project *エラー! 参照元が見つかりません。* can be run in other workspaces. When debugging in the other workspace, 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 e²studio, all FSP applications are organized in RZ/T, RZ/N MPU projects. Setting up an RZ/T, RZ/N 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 e²studio 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.

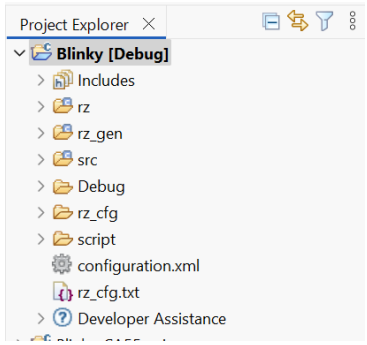


Figure 123. e²studio Project Configuration File

Double-click on the configuration.xml file to open the RZ/T, RZ/N 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 e²studio window. Once selected, you can use the editor to view or modify the configuration settings associated with this project.



Figure 124. e²studio FSP Configuration Perspective

Note: Whenever the RZ/T, RZ/N project configuration (that is, the configuration.xml file) is saved after configuring the project, a verbose RZ/T, RZ/N 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.

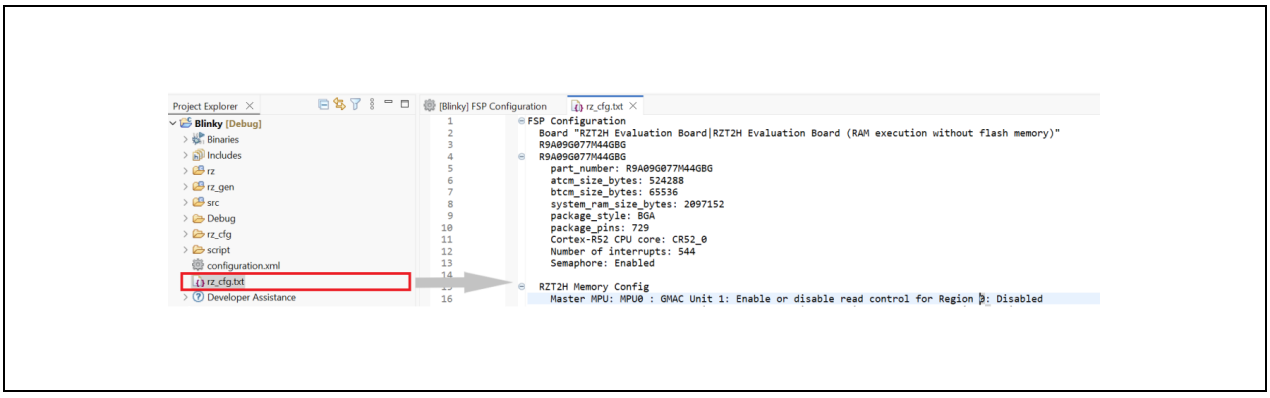


Figure 125. RZ/T, RZ/N Project Report

The RZ/T, RZ/N 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/T, RZ/N Project Editor depend on the e^{2}studio version, and the layout may vary slightly, the functionality should be easy to follow.

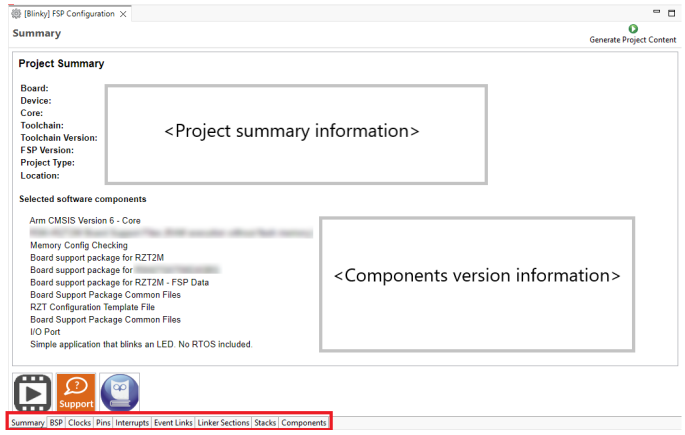


Figure 126. RZ/T, RZ/N Project Editor Tabs

6.2 Create a Project

6.2.1 Creating a New Project

For RZ/T, RZ/N MPU applications, generate a new project using the following steps:

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

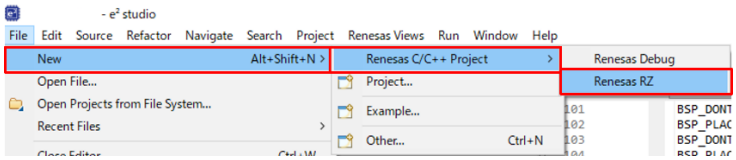


Figure 127. New RZ/T, RZ/N MPU Project

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

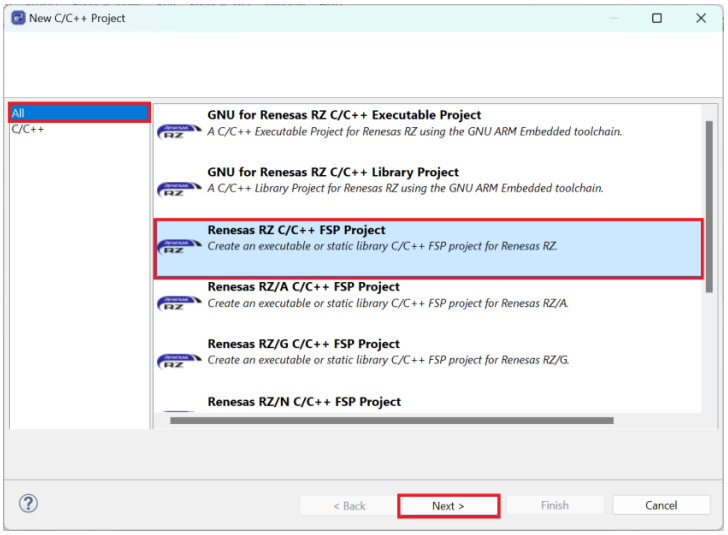


Figure 128. New Project Templates

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

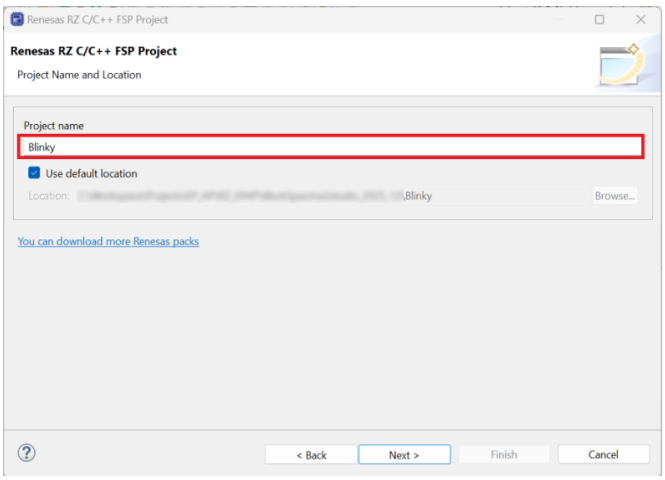


Figure 129. RZ/T, RZ/N MPU Project Generator (1/4)

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/T, RZ/N MPU Evaluation Kit (Such as RSK) or select a **Custom User Board** for any of the RZ/T, RZ/N MPU devices with your own BSP definition.

When you use the RZ/T, RZ/N MPU Evaluation Kit,

- First, set the **Board** to the Evaluation Kit and the boot mode that you use.
- In this case, 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, set the **Device** to your device on your board.
- Second, set the Board to **Custom User Board** with the boot mode that you use.

3. Select the **Core**. The option will be available if you selected multicore device under **Device**.
4. Select the **Toolchains**.
5. Select the **Toolchain version**.
6. Select the **Debugger**. The J-Link Arm is preselected.
7. Click **Next**.

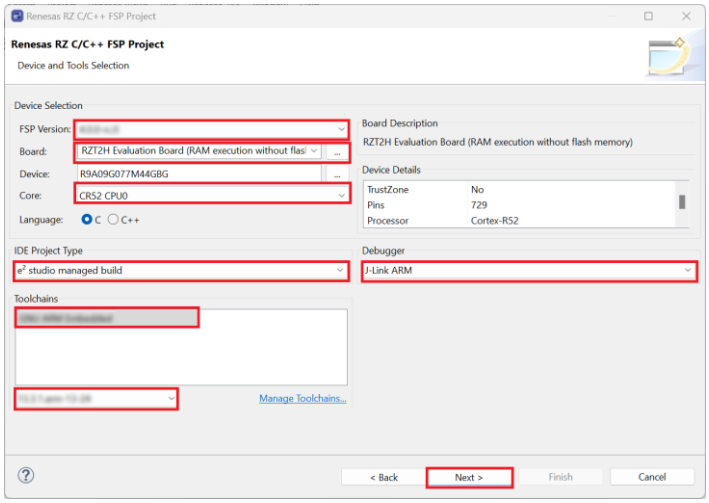


Figure 130. RZ/T, RZ/N MPU Project Generator (2/4)

6.2.3 Selecting a Project Template

In the next window, select the **Build Artifact** and **RTOS**.

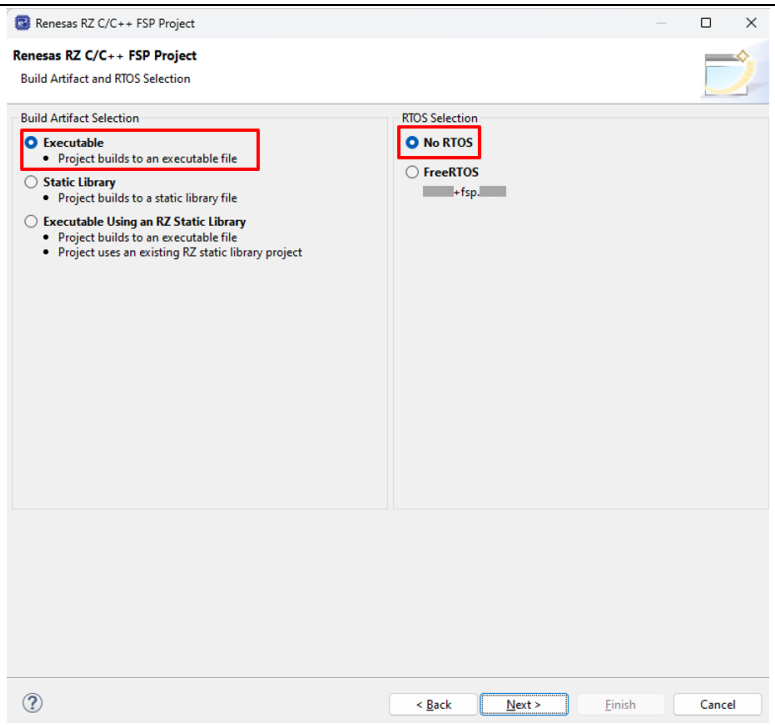


Figure 131. RZ/T, RZ/N MPU Project Generator (3/4)

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/T, RZ/N 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**.

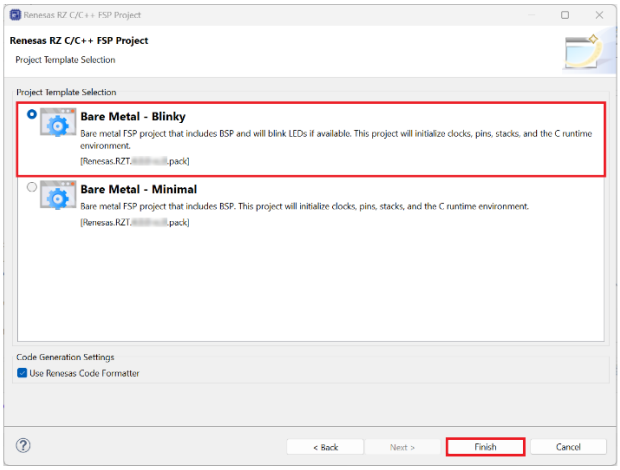


Figure 132. RZ/T, RZ/N MPU Project Generator (4/4)

When the project is created, e^{2}studio displays a summary of the current project configuration in the RZ/T, RZ/N MPU Project Editor.

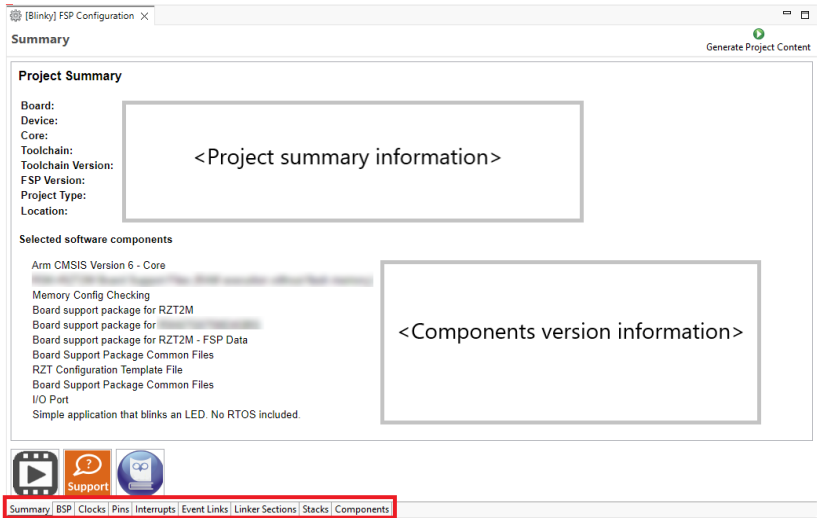


Figure 133. RZ/T, RZ/N MPU Project Editor and Available Editor Tabs

On the bottom of the RZ/T, RZ/N 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 Core0 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 > RZT(RZN) > BSP > MPU 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/T, RZ/N 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^{2}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

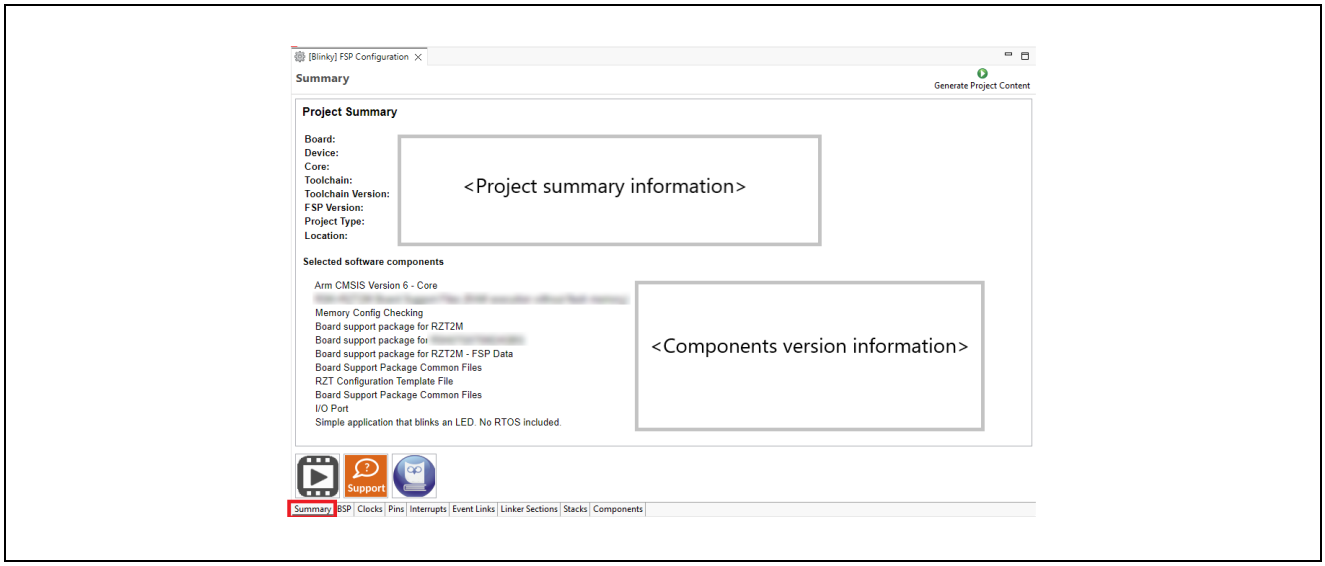


Figure 134. Configuration Summary Tab

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.

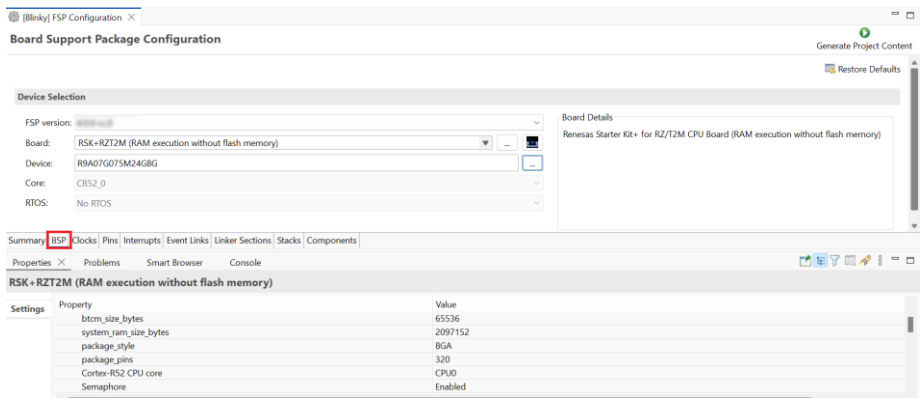


Figure 135. Configuration BSP Tab

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^{2}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.

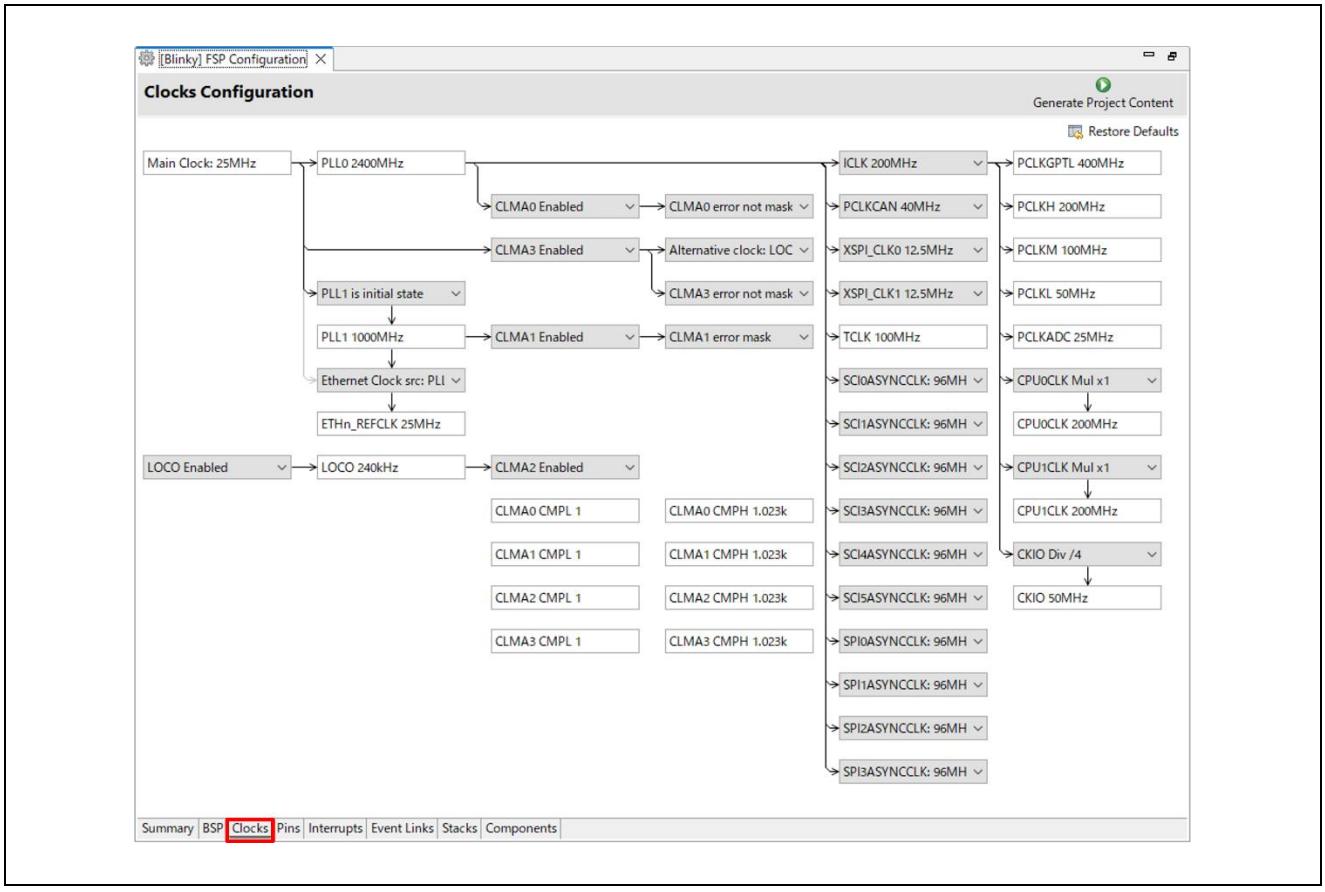


Figure 136. Configuration Clocks Tab

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 receiver 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^{2}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.

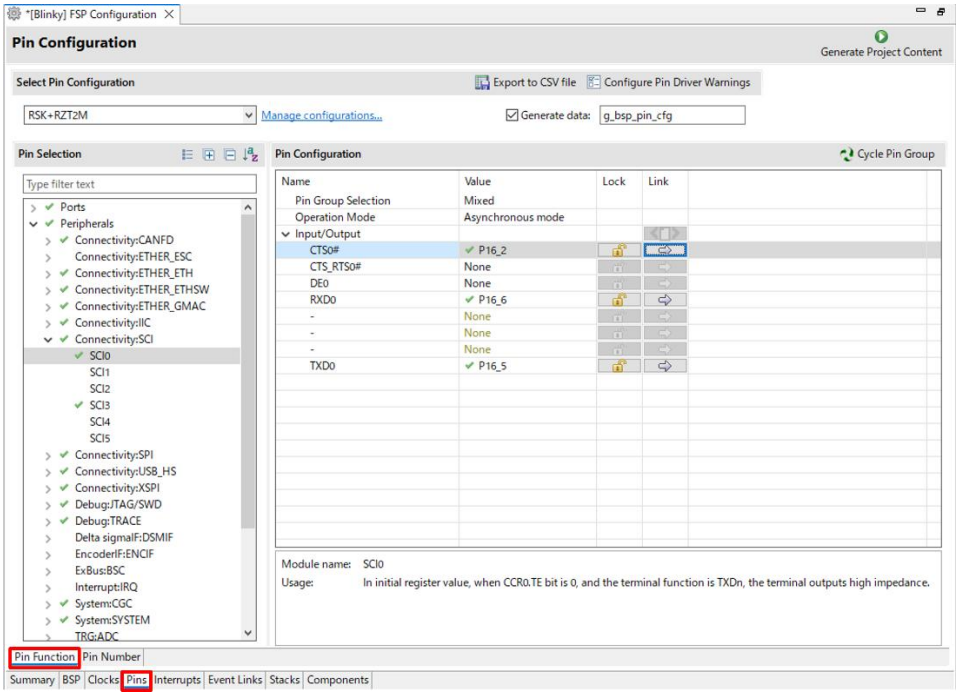


Figure 137. Pin Configuration

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 in 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, pin P16_2 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.

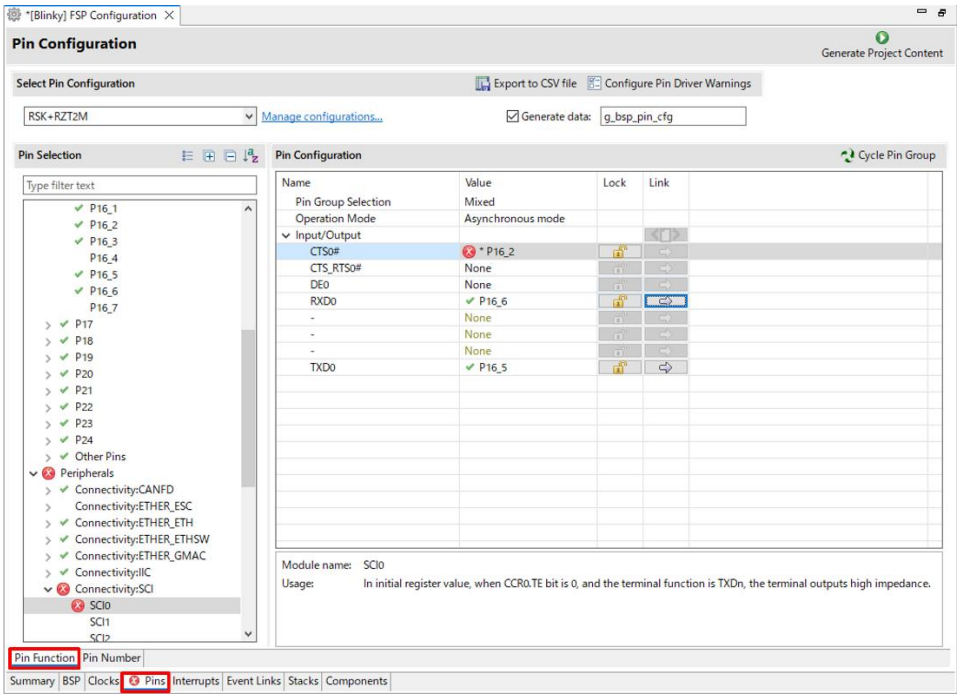


Figure 138. Conflict Checker in Pin Configuration

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

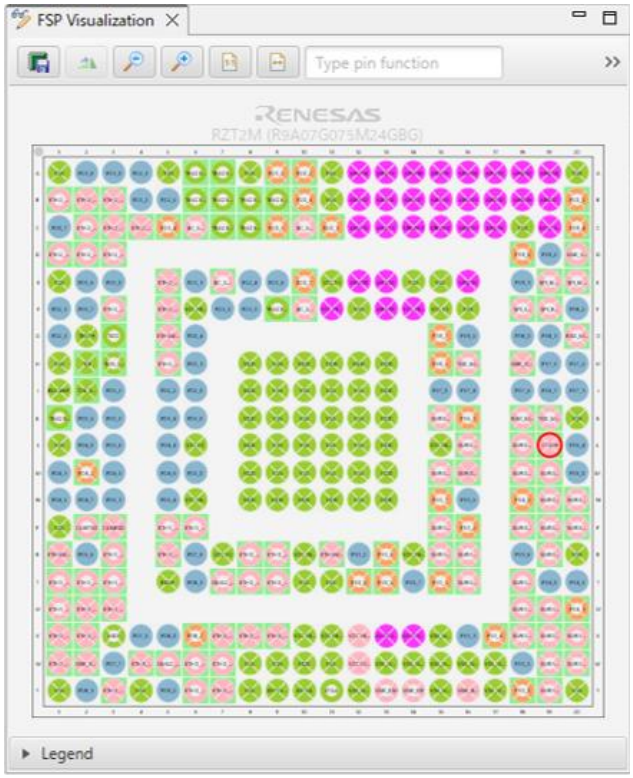


Figure 139. Pin Configurator Package View

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.

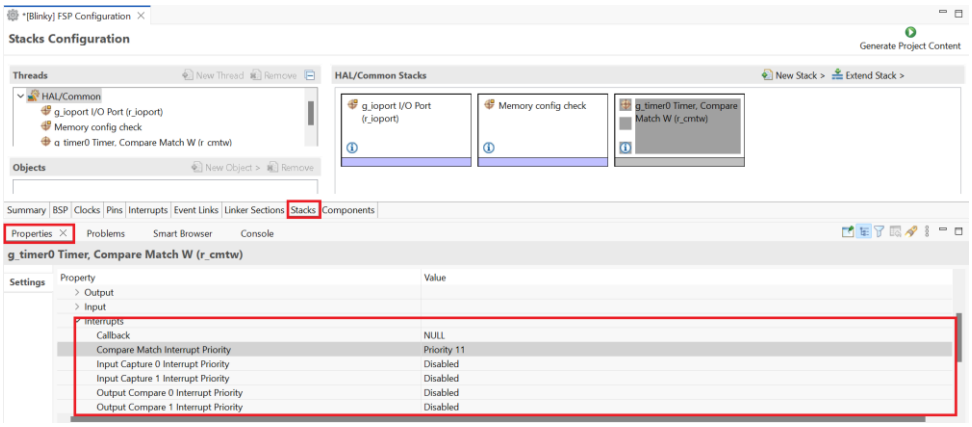


Figure 140. Configuring Interrupts in the Stacks Tab

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.

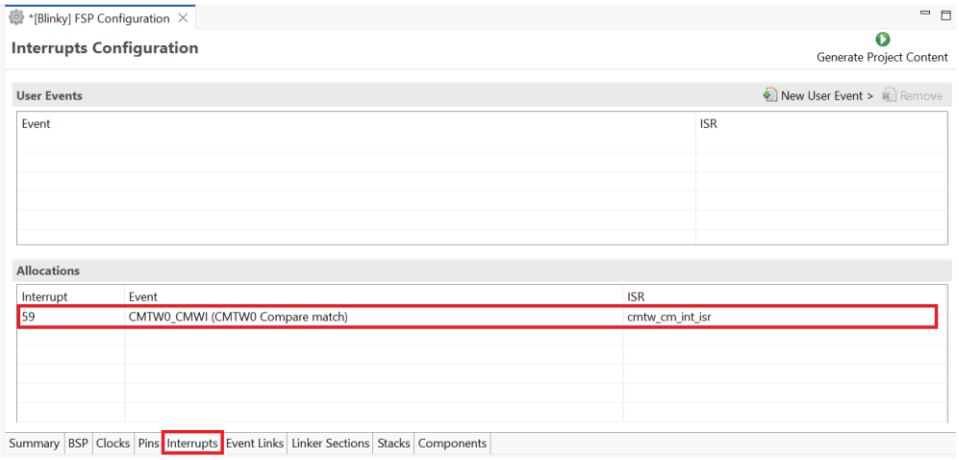


Figure 141. Configuring Interrupt in Interrupt Tab

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 (ELC)** events. The events are sorted by peripheral to make it easy to find and verify them.

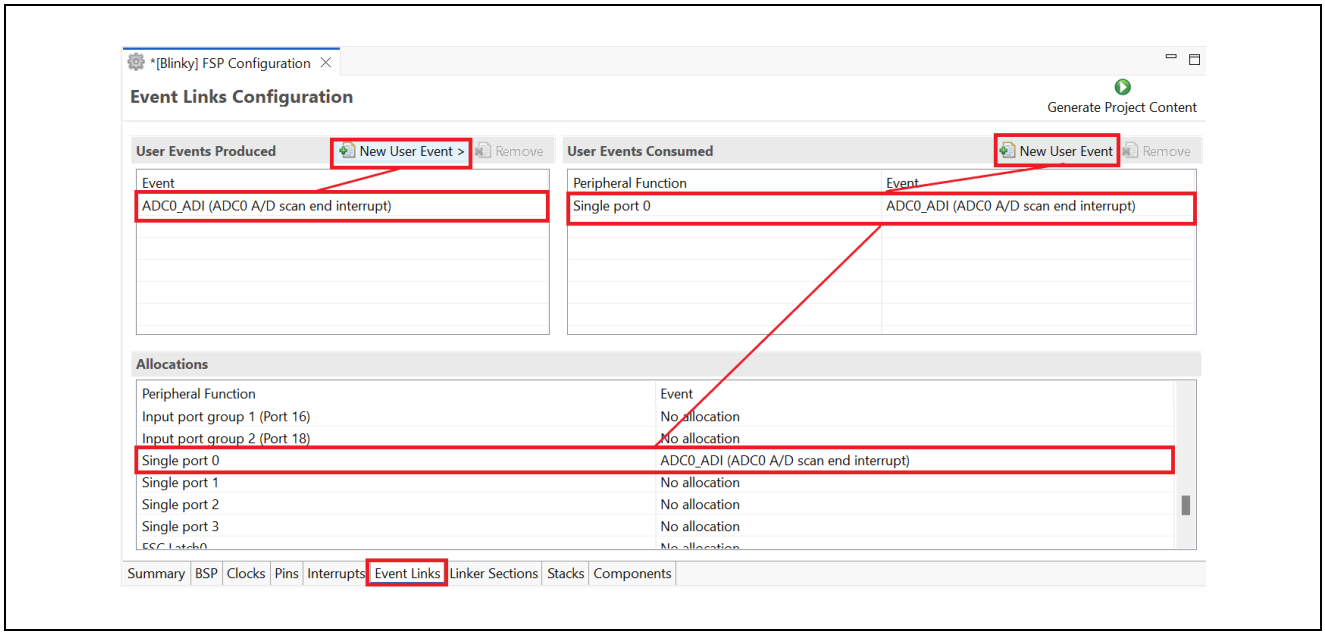


Figure 142. Viewing Event Links

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:

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

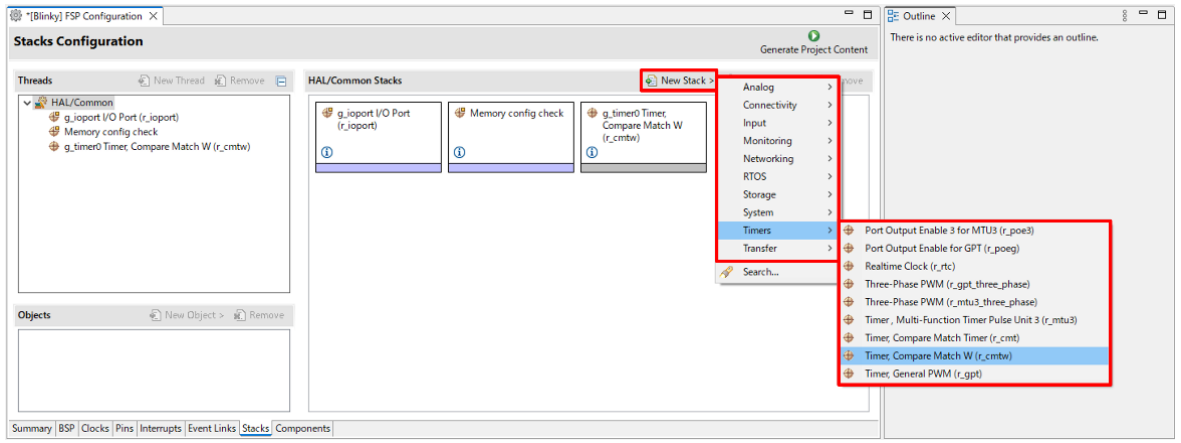


Figure 143. e²studio Project Configurator – Adding Drivers

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

e²studio 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 14. 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.

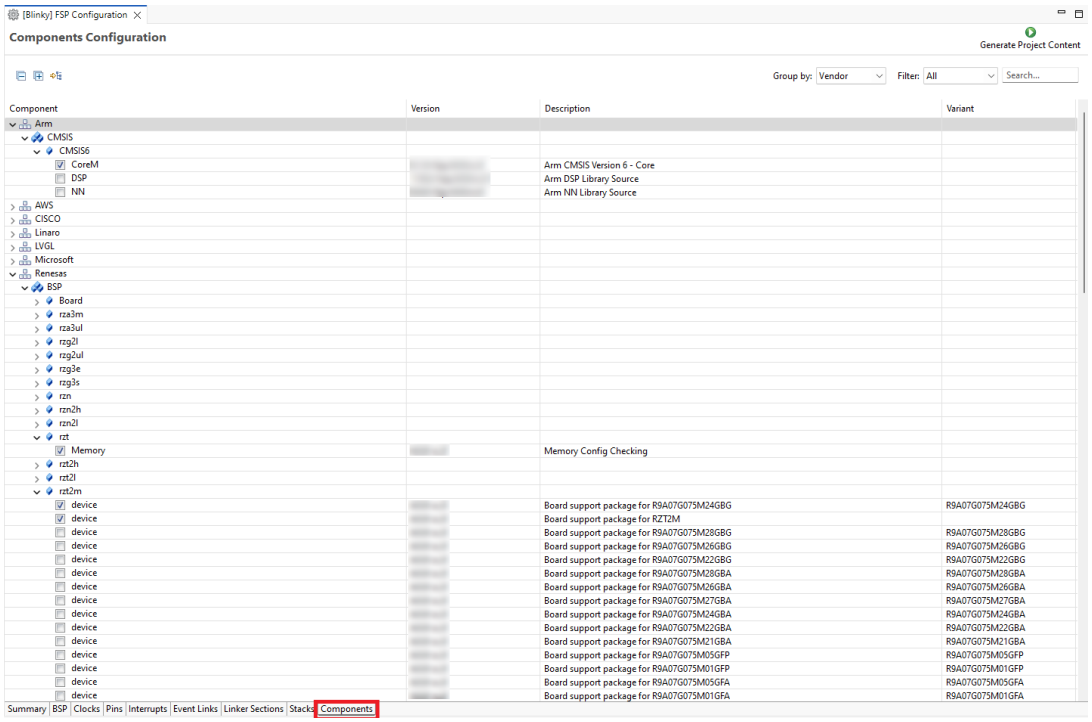


Figure 144. Components Tab

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>

e²studio 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

Visit GitHub Issues for the known issues regarding the current version of FSP and related platform software.

https://github.com/renesas/rz-fsp/issues

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. Follow the operations in the description of the issues if you use the features related to the issues.

8. Appendix

8.1 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.

Erase flash memory by following the steps depending on your IDE (e²studio or IAR EWARM) before running the project.

1. e²studio

   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 J-Link Commander.

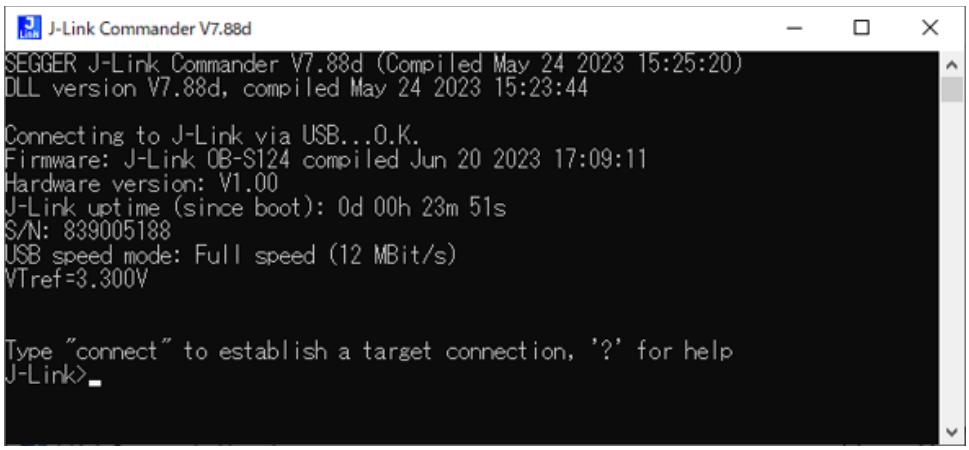


Figure 145. Launch J-Link Commander

iii) First, type “connect” to establish a target connection and press Enter.

Next, specify the connection conditions as follows.

- Device> (Device type name)

Table 15. 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

RZT2N Evaluation Board	R9A07G076M48_R52_0
RSK + RZ/N2L	R9A07G084M04
RZN2H Evaluation Board	R9A09G087M44_R52_0

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

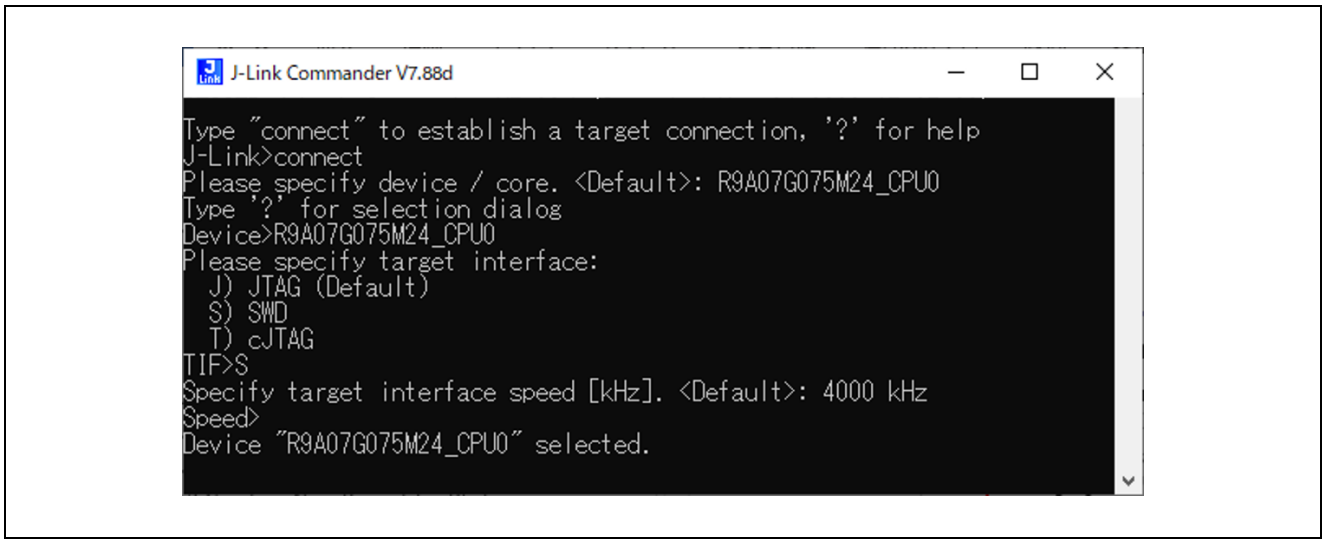


Figure 146. Initial Setup for Connecting to the Device

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

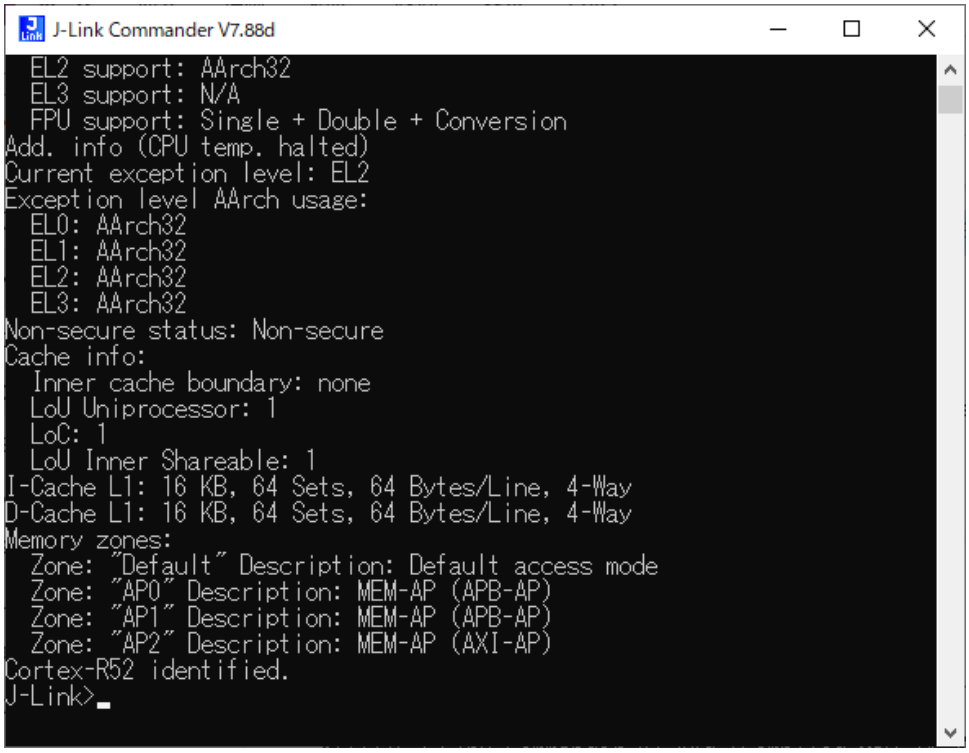


Figure 147. Message of Device Core Identification

iv) Use the commands below to enable flash erase and erase the flash memory.

- J-Link>exec EnableEraseAllFlashBanks
- J-Link>erase (Start address), (End address)

Table 16. 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	0x43FFFFFF
	xSPI1 x1	xSPI1 CS0	0x50000000	0x50FFFFFF
RZT2N Evaluation Board	xSPI0 x1	xSPI0 CS0	0x40000000	0x43FFFFFF
	xSPI1 x1	xSPI1 CS0	0x50000000	0x50FFFFFF
RSK + RZN2L	xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
	16-bit bus	CS0	0x70000000	0x71FFFFFF
RZN2H Evaluation Board	xSPI0 x1	xSPI0 CS0	0x40000000	0x43FFFFFF
	xSPI1 x1	xSPI1 CS0	0x50000000	0x50FFFFFF



Figure 148. Specify Erase Range

After that, confirm that the message “Erasing done” is displayed.

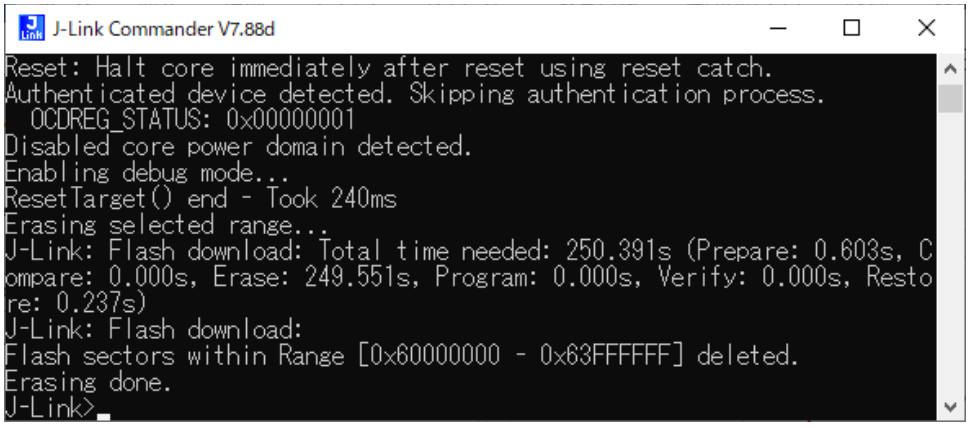


Figure 149. Message of Flash Memory Erase Complete

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

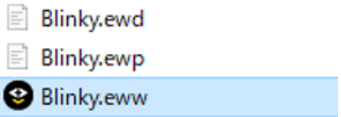


Figure 150. Open Workspace for IAR EWARM

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

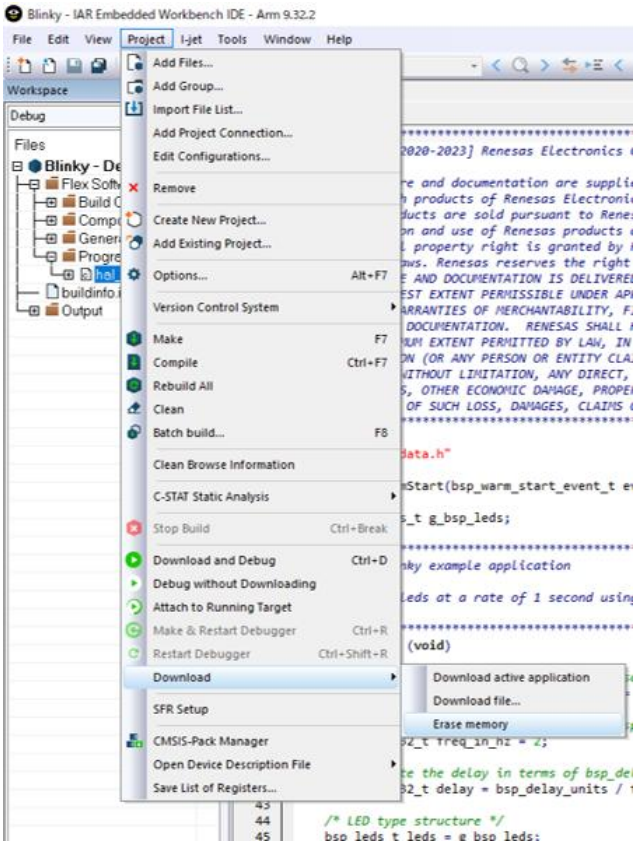


Figure 151. Select Erase Memory Command

iv) Select erase memory space.

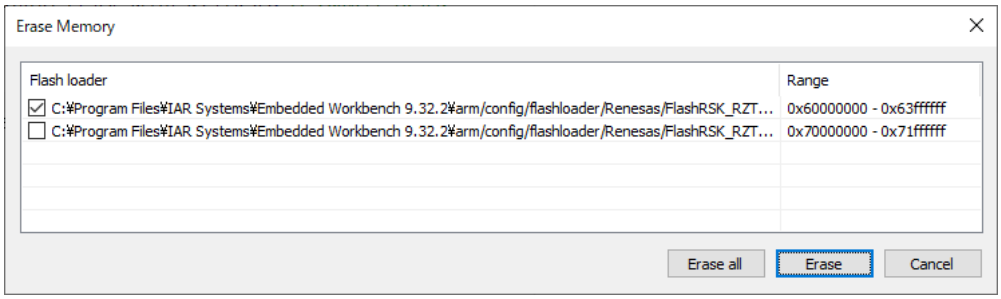


Figure 152. Select Erase Memory Space

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

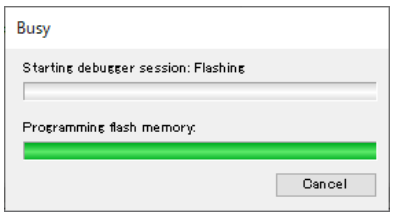


Figure 153. Screen During Erasing

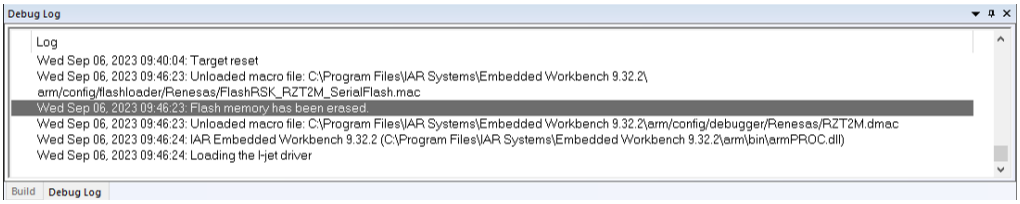


Figure 154. Message of Flash Memory Erase Complete

8.2 How to Program an ELF File to Flash Memory

e²studio and Visual studio Code do not support debugging CMake projects. Therefore, you can program the ELF file to flash memory and verify standalone operation without connecting a debugger.

This section describes how to program an ELF file to flash memory.

1. Perform steps i, ii, iii of 8.1 How to Erase Flash Memory No. 1.
2. Use the commands below to program the ELF file.
   - J-Link>loadfile [Path of ELF file]

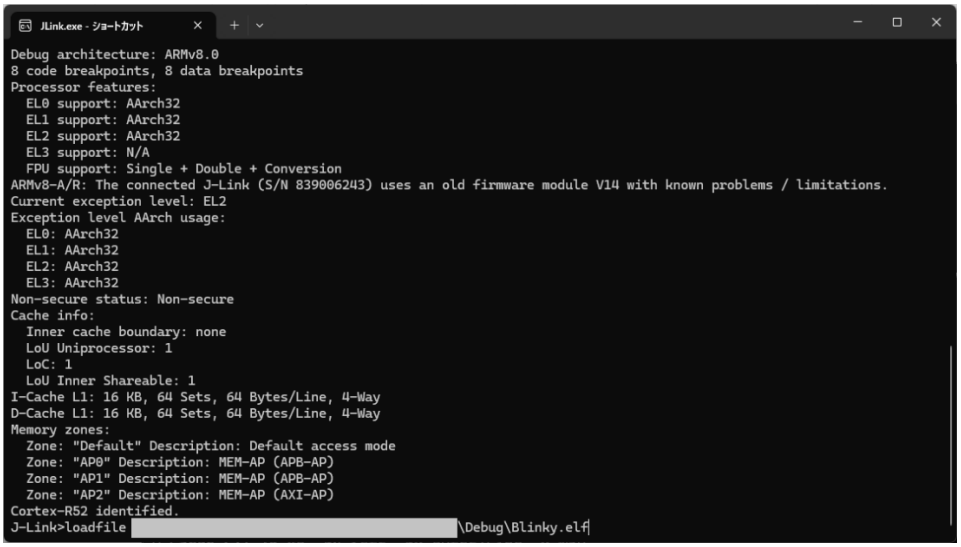


Figure 155. Command to Program ELF File

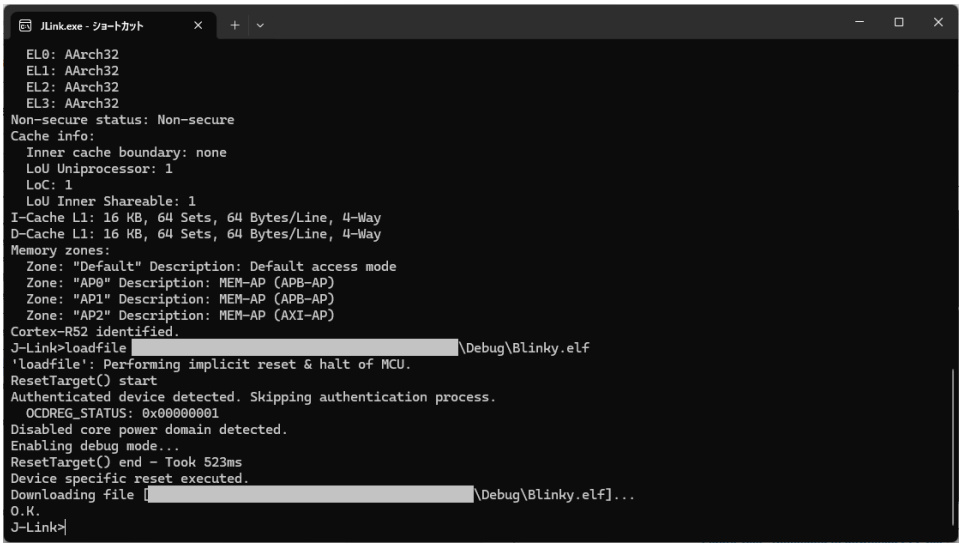


Figure 156. Message of Program Complete

3. Enter “q” to exit J-Link Commander.

9. Appendix

9.1 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).

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…**"

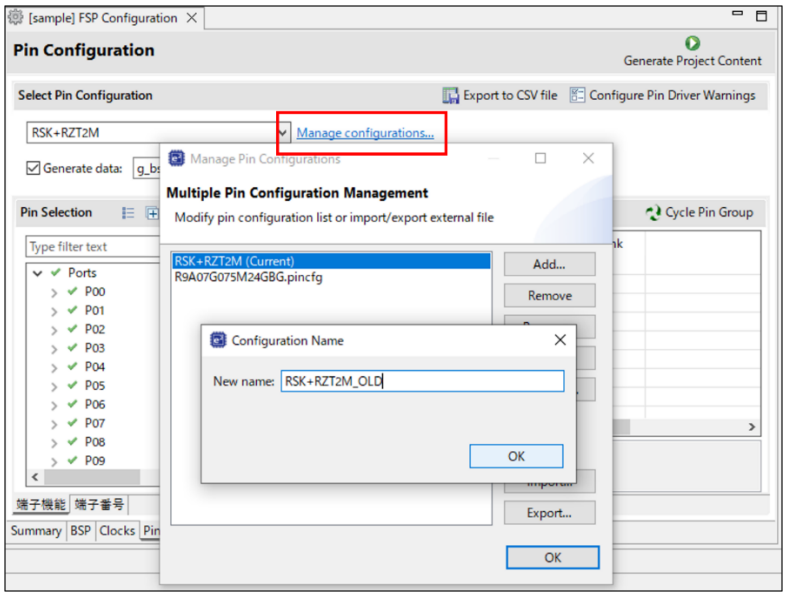


Figure 157. How to Rename Pin Configuration

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

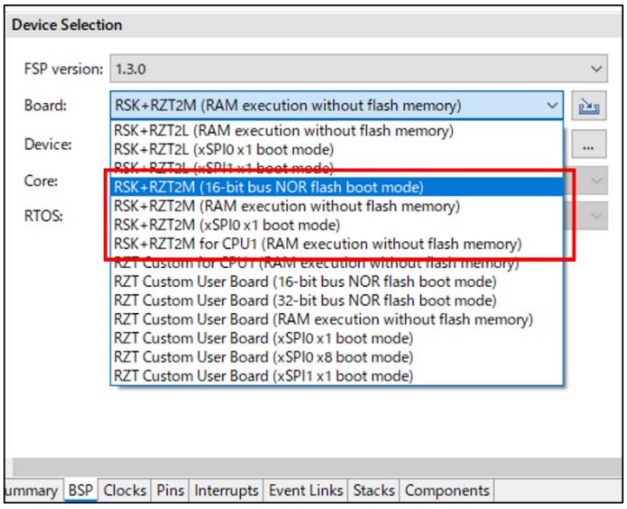


Figure 158. Change the Boot Mode in the BSP Tab

3. Close the configuration and reopen it.

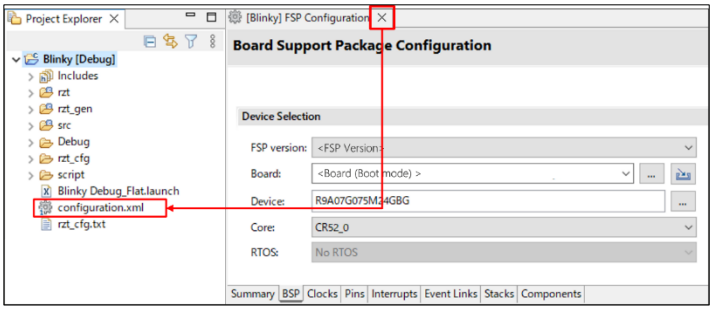


Figure 159. Close Configuration and reopen

4. Uncheck “**Generate data**” in the **Pins** tab.

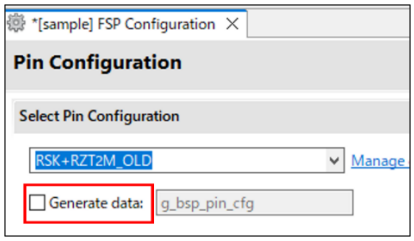


Figure 160. Uncheck Generate data in the Pins Tab

5. Select the regenerated configuration for the board.

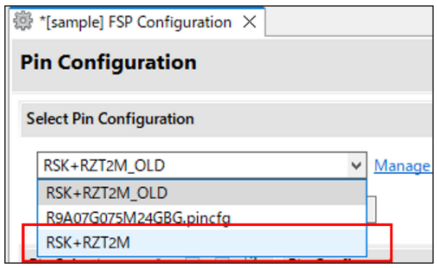


Figure 161. Select the Regenerated Configuration for the Board

6. Check "**Generate data**" again and enter "**g_bsp_pin_cfg**" as the name.

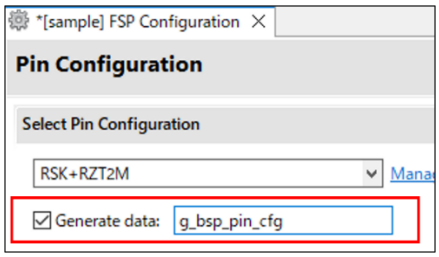


Figure 162. Check the generated data again

10. Appendix. How to Create and Debug FSP Projects for Multiprocessing for e²studio

This chapter describes project creation and debugging procedures in e²studio applicable to device, configuration and core combinations. Since the procedure varies depending on the combination, refer to Table 17. Select Your Case for Multiprocessing and then follow the appropriate section.

Table 17. Select Your Case for Multiprocessing

Use Case	Device	Primary Core	Section
Debug	All Devices	CR52 Core0 or CA55 Core0	10.1
Standalone *	RZ/T2M, RZ/T2ME	CR52 Core0	10.2
	RZ/T2H, RZ/T2N, RZ/N2H	CR52 Core0	10.3
		CA55 Core0	10.4

* Standalone means operation using a single ELF file containing the binaries for all cores.

Important Note for Standalone Multiprocessing:
- Standalone multiprocessing operation fundamentally requires Flash boot mode (including xSPI, NOR flash, eSD, and eMMC boot modes). In standalone mode, the Primary core's bootloader must unpack and copy the Secondary core images from non-volatile memory to their designated execution RAM areas before releasing the Secondary cores from reset.
- RAM execution without flash memory is NOT supported for Standalone Multiprocessing.
- When creating the Primary project and all subsequent projects, you must select the appropriate Flash Boot Mode board configuration for your evaluation board (see Table 16. External Address Space to Be Used in Each Boot Mode).

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

Note:

In the case of multiprocessing, two or more projects with different settings must be created. A project that starts first is called the primary project, and a project that runs after being reset by the primary project is called the secondary project (subsequent projects are referred to as tertiary, quaternary, quinary, and senary).

All projects should be created in the same workspace.

The build order is required because build artifacts generated by each project are used by the other project.

- **Primary project build (1st build)**
  Generates bundle information, .sbd files, and project metadata required by the Secondary project.
- **Secondary project build**
  Generates the Secondary project binaries and object files.
- **Primary project build (2nd build)**
  Incorporates the latest Secondary project image into the final Primary project image.

10.1 Debug Primary CR52 Core0 or CA55 Core0 (All Devices)

1. Create a primary project (see section 4.3 Create a New Project for Blinky).
   i. (primary: CR52 and (secondary or later: CA55)) Set "PLL0 is released from standby state" and enable PLL0 in the Clocks tab of FSP Configuration.

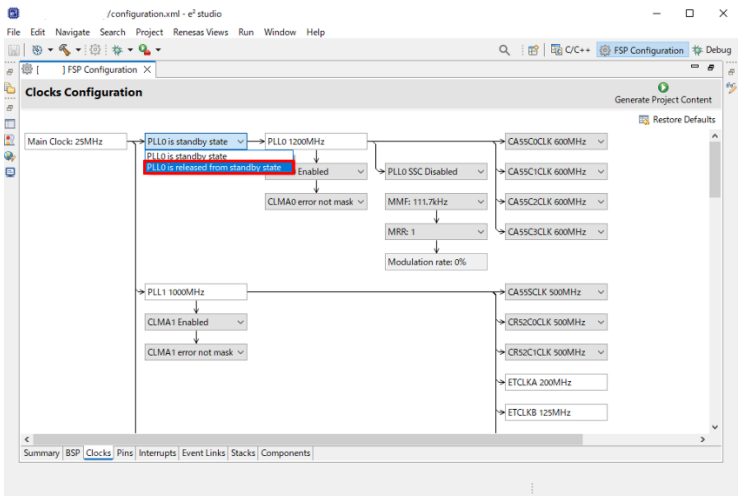


Figure 163. Enable PLL0 in the Primary Project using CR52 Core0

ii. (RZ/T2H and (secondary or later: CA55 Core1)) To blink the LED on the RZ/T2H CA55 Core1, you need to change the pin configuration of the primary project in the FSP Configuration as follows.

- In the Pins tab of the FSP configuration, click **Pin Selection** -> **Peripherals** -> **Connectivity: SDHI** -> **SDHI1**
- Change the value of SD1_PWEN to None.
- In the Pins tab of the FSP configuration, click **Pin Selection** -> **Ports** -> **P08** -> **P08_5**.
- Change the value of Symbolic Name to LED3.
- Change the value of Mode to Output mode (Low & Not Into Input)

iii. Click the **Generate Project Content** button in the top right corner of the Project Configuration window to generate your board-specific files.

2. Build the primary project (see section 4.4 Build the Blinky Project).
3. Create and build the subsequent project
   - Repeat the following steps until all required projects have been created.
     i. Create a subsequent project (see section 4.3 Create a New Project for Blinky and select the bundle file (.sbd) of the preceding project in No. 12).
     ii. Build the subsequent project (see section 4.4 Build the Blinky Project).
4. Configure the debug configuration for all projects (see section 4.5.2 Debug Steps No.1 to No.4).
5. Click **Debug** in Debug Configuration of the primary project to begin debugging the application (see section 4.5.2 Debug Steps No.5).
6. Click **Debug** in Debug Configuration of all subsequent projects to begin debugging the application (see section 4.5.2 Debug Steps No.5).
   - When the following dialog box is shown, Click **No** to start debugging.

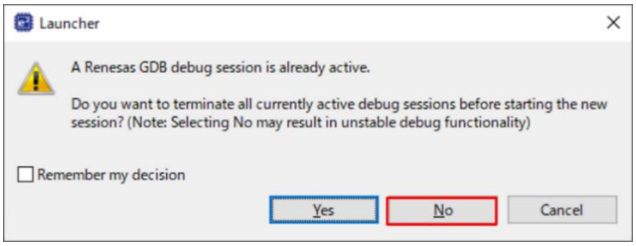


Figure 164. Warning Window of Starting Debug Session

- When the following dialog box is shown, Click **Yes** to proceed with the launch.

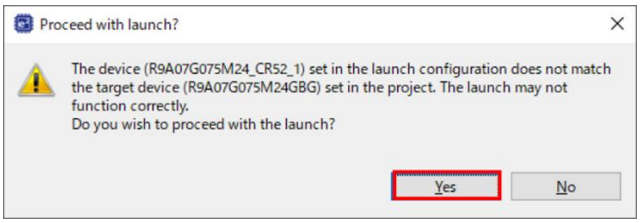


Figure 165. Warning Window of Device Name

7. Run the primary project (see section 4.6 Run the Blinky Project).
8. Run all subsequent projects (see section 4.6 Run the Blinky Project).
9. To exit Debug mode and disconnect from the debugger, click on the Terminate all icon



Figure 166. e²studio Debugger Terminate all Icon

10.2 Standalone Primary CR52 Core0 (RZ/T2M or RZ/T2ME)

1. Create a primary project (see section 4.3 Create a New Project for Blinky).
2. Build the primary project (see section 4.4 Build the Blinky Project).
3. Create and build the subsequent project
   i. Create a subsequent project (see section 4.3 Create a New Project for Blinky and select the bundle file (.sbd) of the preceding project in No. 12).
   ii. Build the subsequent project (see section 4.4 Build the Blinky Project).
4. Link the object files generated in project to the preceding project.
   i. In the Properties window of the preceding project, click **C/C++ Build** > **Settings** > **Tool Settings** > **Cross ARM C Linker** > **Miscellaneous**.
   ii. Set the file paths of the following object files generated by the subsequent project.

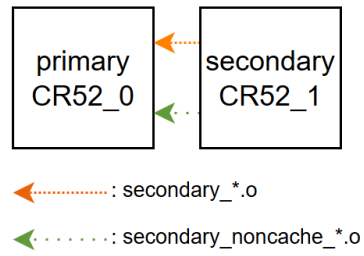


Figure 167. Project Structure Example

Table 18. Object files Output to the Debug Folder of the Subsequent Project

Core	Object files	Note
CR52 Core1	secondary_CR52.o	
	secondary_noncache_CR52.o	When using a non-cache section

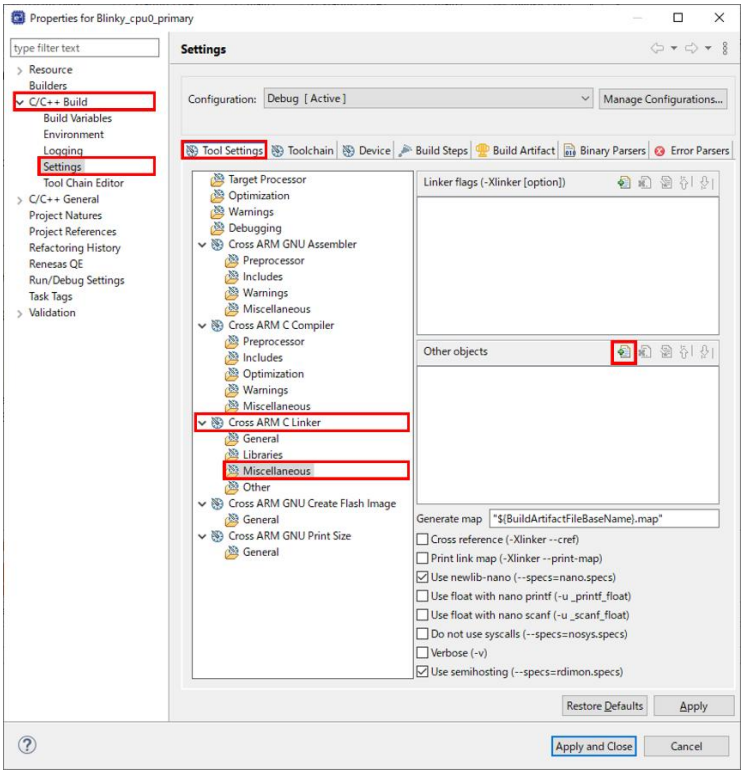


Figure 168. e²studio Build Setting for the Preceding Project (1/2)

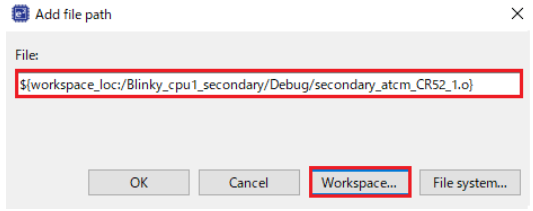


Figure 169. e²studio Build Setting for the Preceding Project (2/2)

iii. Build the preceding project (see section 4.4 Build the Blinky Project).

Note:

If you make changes to any project, repeat the following steps for each project, starting with the project you modified and ending with the primary project.

1. Right-click on the project and select **Clean Project**.
2. Build the project (see section 4.4 Build the Blinky Project).

10.3 Standalone Primary CR52 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H)

1. Create a primary project (see section 4.3 Create a New Project for Blinky).
   i. (primary: CR52 and (secondary or later: CA55)) Set "PLL0 is released from standby state" and enable PLL0 in the Clocks tab of FSP Configuration.

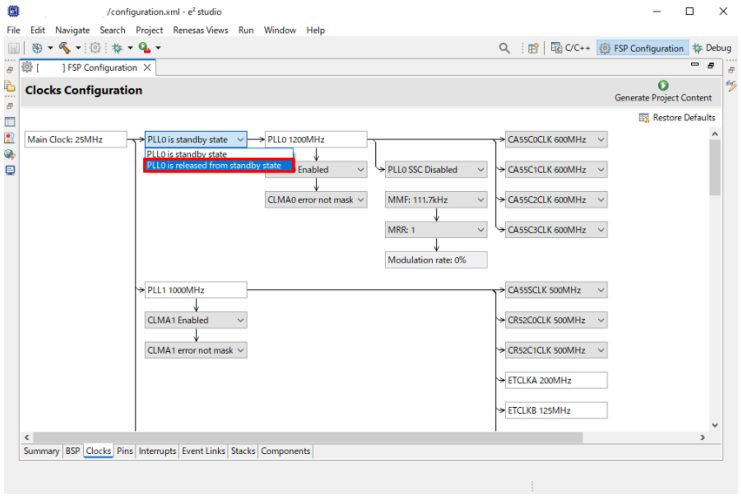


Figure 170. Enable PLL0 in the Primary Project using CR52 Core0

ii. (RZ/T2H and (secondary or later: CA55 Core1)) To blink the LED on the RZ/T2H CA55 Core1, you need to change the pin configuration of the primary project in the FSP Configuration as follows.

   a. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Peripherals** -> **Connectivity: SDHI** -> **SDHI1**
   b. Change the value of SD1_PWEN to None.
   c. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Ports** -> **P08** -> **P08_5**.

d. Change the value of Symbolic Name to LED3.
e. Change the value of Mode to Output mode (Low & Not Into Input)

iii. Click the **Generate Project Content** button in the top right corner of the Project Configuration window to generate your board-specific files.

2. Build the primary project (see section 4.4 Build the Blinky Project).
3. Create and build the subsequent project
   - Repeat the following steps until all required projects have been created.
     i. Create a subsequent project (see section 4.3 Create a New Project for Blinky and select the bundle file (.sbd) of the preceding project in No. 12).
     ii. Build the subsequent project (see section 4.4 Build the Blinky Project).
4. Link the object files generated by the subsequent project to the preceding project.
   - Repeat the following steps for each project, starting with the second-to-last project and ending with the primary project.
     i. In the Properties window of the preceding project, click **C/C++ Build** > **Settings** > **Tool Settings** > **Cross ARM C Linker** > **Miscellaneous**.
     ii. Set the file paths of the following object files generated by the subsequent project.

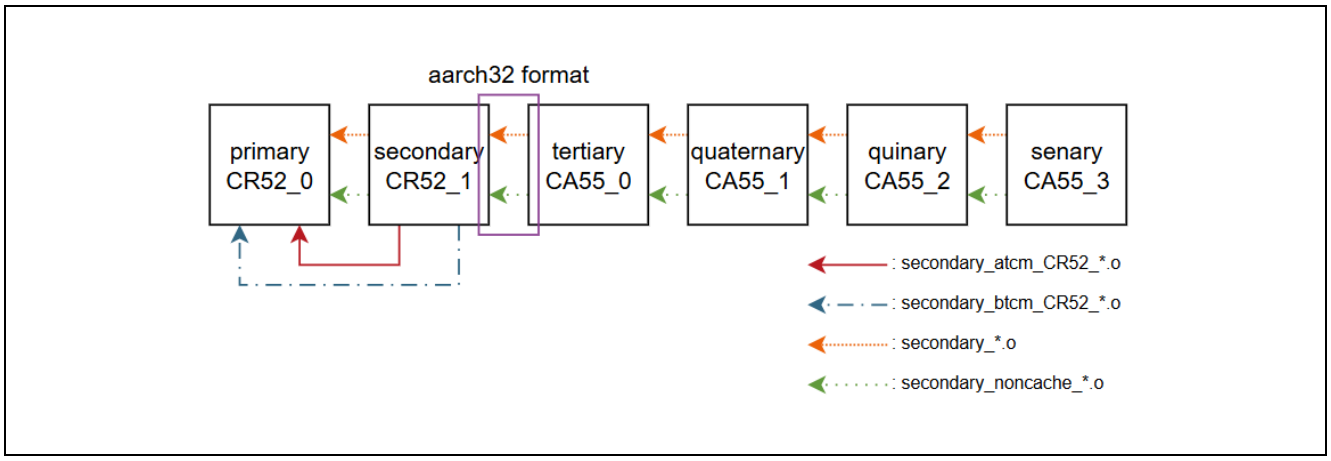


Figure 171. Project Structure Example

Note:

The diagram above is an example. If you are using a different project configuration, follow these rules.

1. The object files for ATCM and BTCM are linked to the primary project.
2. The object files for System SRAM (cache and non-cache) are linked to the preceding project.
3. Since object files in aarch64 format (CA55) cannot be linked to CR52 projects, object files in aarch32 format should be linked instead.

Table 19. Object files Output to the Debug Folder of the Subsequent Project

Core	Object files	Note
CR52 Core1	secondary_atcm_CR52_1.o	
	secondary_btcm_CR52_1.o	
	secondary_systemram_CR52.o	When using a System RAM section
	secondary_noncache_CR52.o	When using a non-cache section
CA55 Core0	secondary_aarch32_CA55.o	
	secondary_noncache_aarch32_CA55.o	When using a non-cache section

CA55 Core1, 2 or 3	secondary_CA55.o	
	secondary_noncache_CA55.o	When using a non-cache section

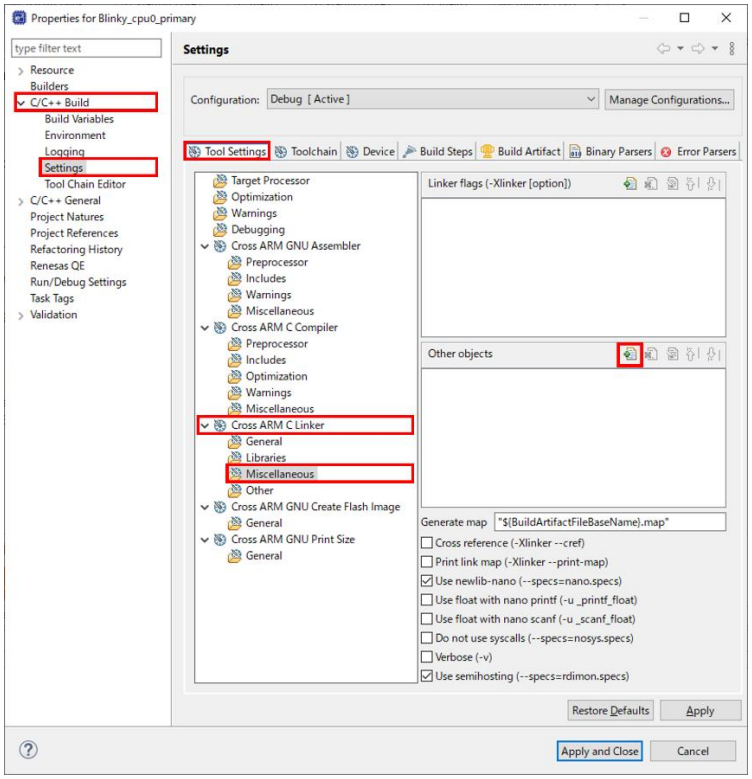


Figure 172. e²studio Build Setting for the Preceding Project (1/2)

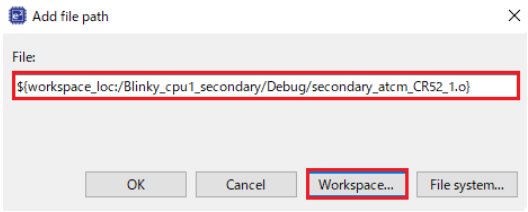


Figure 173. e²studio Build Setting for the Preceding Project (2/2)

iii. Build the preceding project (see section 4.4 Build the Blinky Project).

Note:

If you make changes to any project, repeat the following steps for each project, starting with the project you modified and ending with the primary project.

1. Right-click on the project and select **Clean Project**.
2. Build the project (see section 4.4 Build the Blinky Project).

10.4 Standalone Primary CA55 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H)

1. Create a primary project (see section 4.3 Create a New Project for Blinky).
   i. (RZ/T2H and (secondary or later: CA55 Core1)) To blink the LED on the RZ/T2H CA55 Core1, you need to change the pin configuration of the primary project in the FSP Configuration as follows.
      a. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Peripherals** -> **Connectivity: SDHI** -> **SDHI1**
      b. Change the value of SD1_PWEN to None.
      c. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Ports** -> **P08** -> **P08_5**.
      d. Change the value of Symbolic Name to LED3.
      e. Change the value of Mode to Output mode (Low & Not Into Input)
   ii. Click the **Generate Project Content** button in the top right corner of the Project Configuration window to generate your board-specific files.
2. Build the primary project (see section 4.4 Build the Blinky Project).
3. Create and build the subsequent project
   - Repeat the following steps until all required projects have been created.
   i. Create a subsequent project (see section 4.3 Create a New Project for Blinky and select the bundle file (.sbd) of the preceding project in No. 12).
   ii. (secondary or later: CR52) In the Properties window of the project, click **C/C++ Build** > **Environment** > Add and set it as follows to convert CR52 object files to aarch64 format.
      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.

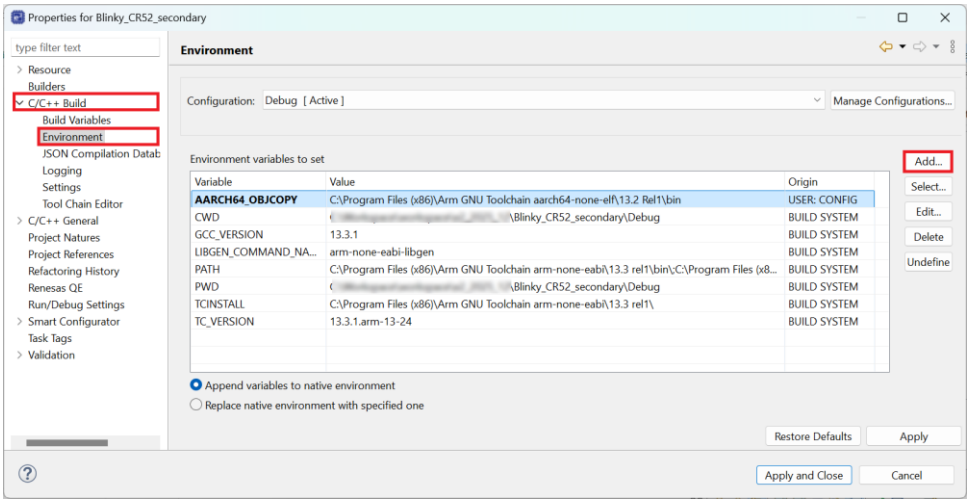


Figure 174. e²studio Build Setting for Generating aarch64 Object Files (1/2)

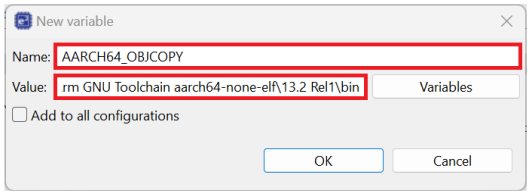


Figure 175. e²studio Build Setting for Generating aarch64 Object Files (2/2)

iii. Build the subsequent project (see section 4.4 Build the Blinky Project).

4. Link the object files generated by the subsequent project to the preceding project.

- Repeat the following steps for each project, starting with the second-to-last project and ending with the primary project.

i. In the Properties window of the preceding project, click **C/C++ Build** > **Settings** > **Tool Settings** > **Cross ARM C Linker** > **Miscellaneous**.

ii. Set the file paths of the following object files generated by the subsequent project.

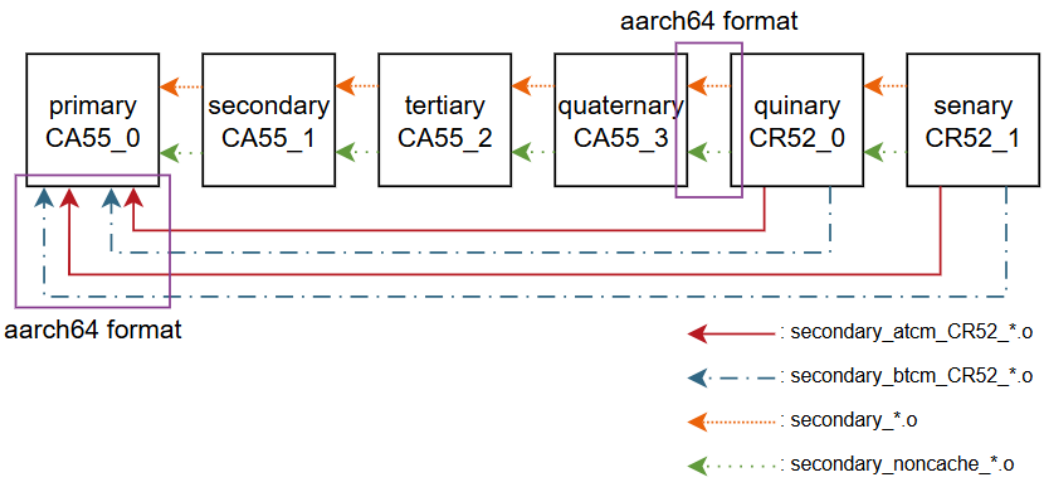


Figure 176. Project Structure Example for e²studio

Note:

The diagram above is an example. If you are using a different project configuration, follow these rules.

1. The object files for ATCM and BTCM are linked to the primary project.
2. The object files for System SRAM (cache and non-cache) are linked to the preceding project.
3. Since object files in aarch32 format (CR52) cannot be linked to CA55 projects, object files in aarch64 format should be linked instead.

Table 20. Object Files Output to the Debug Folder of the Subsequent Project

Core	Object files	Note
CA55 Core1, 2 or 3	secondary_CA55.o	
	secondary_noncache_CA55.o	When using a non-cache section
CR52 Core0	secondary_atcm_aarch64_CR52_0.o	

	secondary_btcm_aarch64_CR52_0.o	
	secondary_systemram_aarch64_CR52.o	When using a System RAM section
	secondary_noncache_aarch64_CR52.o	When using a non-cache section
CR52 Core1	secondary_atcm_aarch64_CR52_1.o	
	secondary_btcm_aarch64_CR52_1.o	
	secondary_systemram_CR52.o	When using a System RAM section
	secondary_noncache_CR52.o	When using a non-cache section

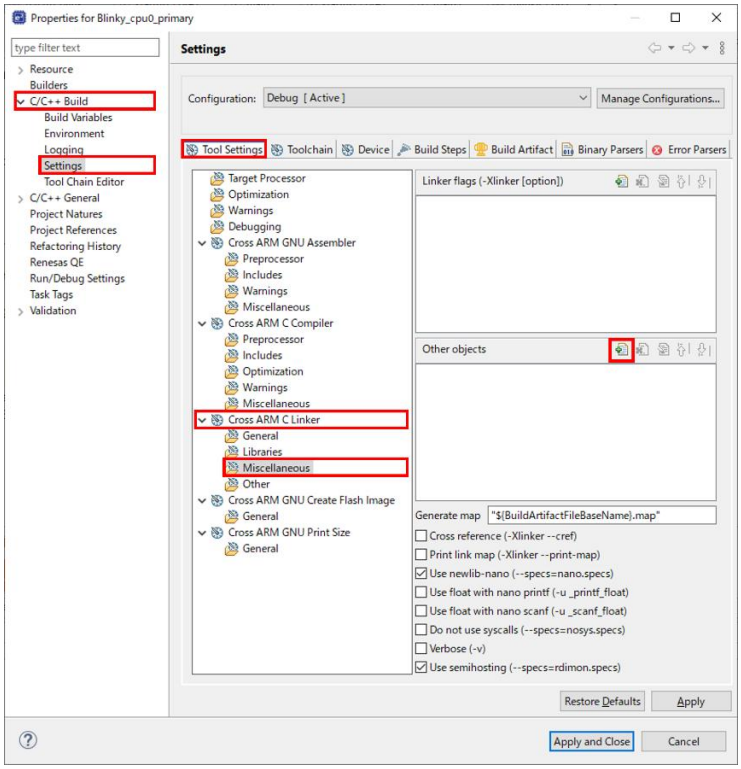


Figure 177. e²studio Build Setting for the Preceding Project (1/2)

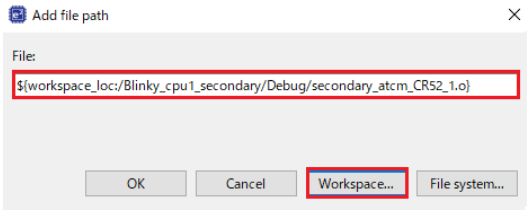


Figure 178. e²studio Build Setting for the Preceding Project (2/2)

iii. Build the preceding project (see section 4.4 Build the Blinky Project).

Note:

If you make changes to any project, repeat the following steps for each project, starting with the project you modified and ending with the primary project.

1. Right-click on the project and select **Clean Project**.
2. Build the project (see section 4.4 Build the Blinky Project).

Note:

(DEPRECATED) If you are using prebuild.sh to generate object files in aarch64 format, add the following settings to the project that will link the object files:

1. In the Properties window of the primary project, click **C/C++ Build** > **Settings** > **Build Steps**.
2. Add **Command(s)** at **Pre-build steps**.
   - sh ../script/prebuild.sh ../../[CR52 project name]/Debug
   - (When there are multiple projects) sh ../script/prebuild.sh ../../[CR52 project name]/Debug && sh ../script/prebuild.sh ../../[CR52 project name]/Debug

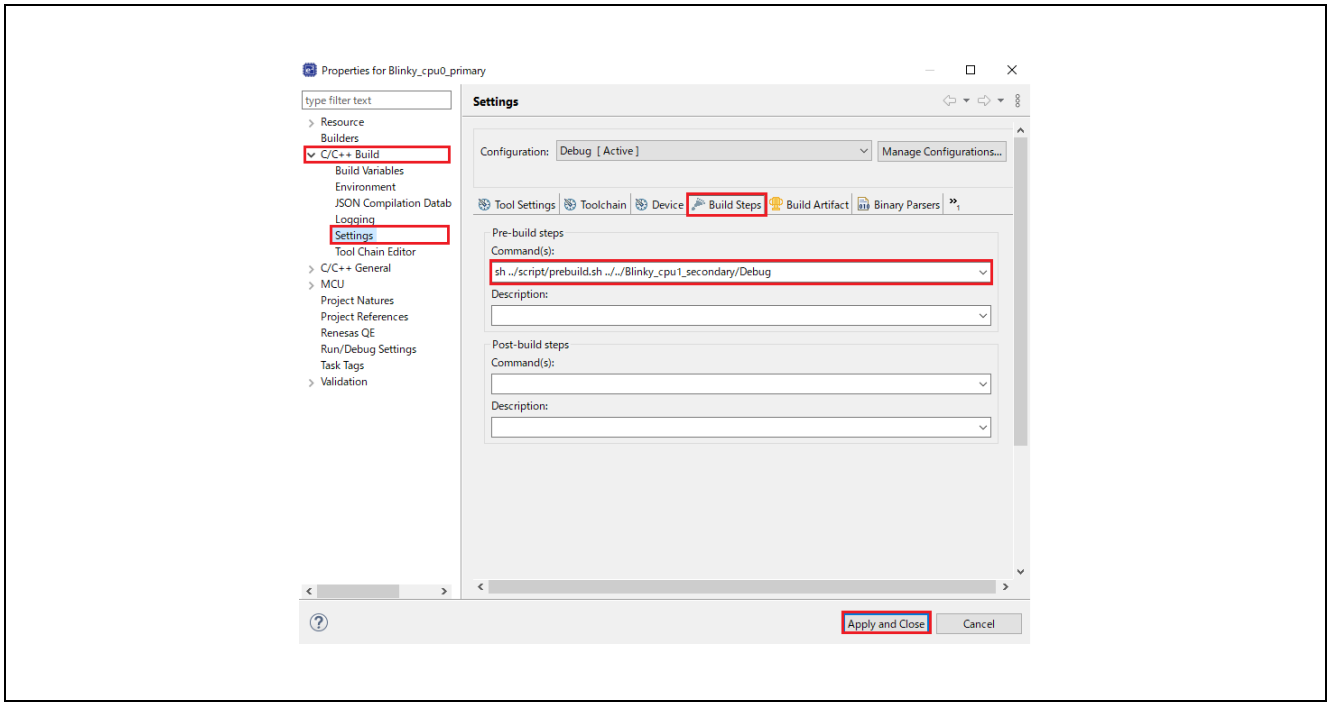


Figure 179. e²studio Build Setting for Generating aarch64 Object Files using prebuild.sh

11. Appendix. How to Create and Debug FSP Projects for Multiprocessing for IAR EWARM

This chapter describes project creation in FSP SC and debugging methods in IAR EWARM applicable to device, configuration and core combinations.

Note for Multiprocessing Projects:

In the case of multiprocessing, two or more projects with different settings must be created. A project that starts first is called the primary project, and a project that runs after being reset by the primary project is called the secondary project (subsequent projects are referred to as tertiary, quaternary, quinary, and senary).

All projects should be created in the same workspace.

The build order is required because build artifacts generated by each project are used by the other project.

- **Primary project build (1st build)**
  Generates bundle information, .sbd files, and project metadata required by the Secondary project.
- **Secondary project build**
  Generates the Secondary project binaries and object files.
- **Primary project build (2nd build)**
  Incorporates the latest Secondary project image into the final Primary project image.

Note:

If you configure your system to build at the start of debugging, it will build all projects during multi-core debugging, which will take an enormous amount of time.
To prevent this, the following settings are recommended.

1. Click **Tools** in the menu bar, then click **Options...**
2. Click **Project** and set **Make before debugging** to Never

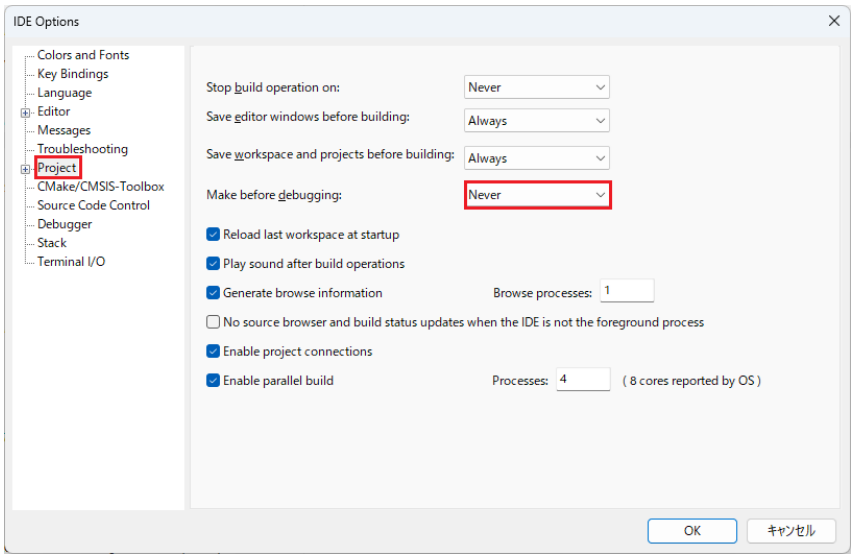


Figure 180. IDE Options (Make before debugging: Never)

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

Table 21. Select Your Case for Multiprocessing

Use Case	Devices	Primary Core	Section
Debug	All Device	CR52 Core0 or CA55 Core0	11.1
Standalone *	RZ/T2M, RZ/T2ME	CR52 Core0	11.2
	RZ/T2H, RZ/T2N, RZ/N2H	CR52 Core0	11.3
		CA55 Core0	11.4

* Standalone means operation using a single OUT file containing the binaries for all cores.

11.1 Debug Primary CR52 Core0 or CA55 Core0 (All Devices)

1. Create a primary project (see section 5.3.2 Create a New Project).
   i. (primary: CR52 and (secondary or later: CA55)) Set "PLL0 is released from standby state" and enable PLL0 in the Clocks tab of FSP Configuration.

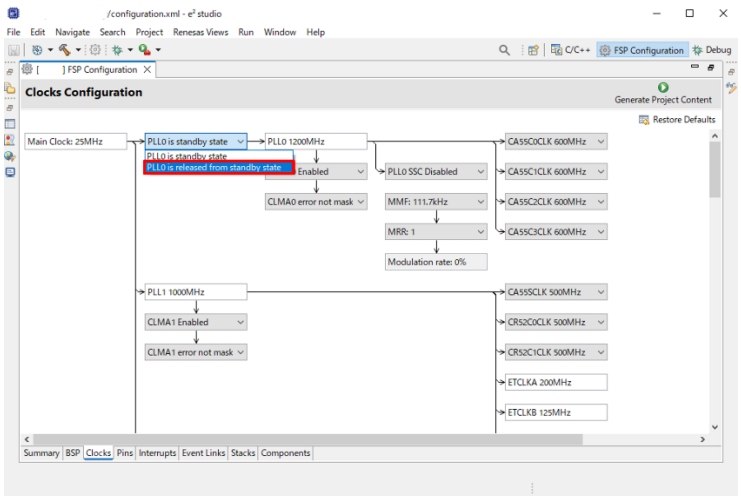


Figure 181. Enable PLL0 in the Primary Project using CR52 Core0

ii. (RZ/T2H and (secondary or later: CA55 Core1)) To blink the LED on the RZ/T2H CA55 Core1, you need to change the pin configuration of the primary project in the FSP Configuration as follows.
   a. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Peripherals** -> **Connectivity: SDHI** -> **SDHI1**
   b. Change the value of SD1_PWEN to None.
   c. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Ports** -> **P08** -> **P08_5**.
   d. Change the value of Symbolic Name to LED3.
   e. Change the value of Mode to Output mode (Low & Not Into Input)

iii. Click the **Generate Project Content** button in the top right corner of the Project Configuration window to generate your board-specific files.

2. Build the primary project. Click on **Project** -> **Rebuild All** from the menu bar.
3. Create and build the subsequent project
   - Repeat the following steps until all required projects have been created.
     i. Create a subsequent project (see section 5.3.2 Create a New Project and select the bundle file (.sbd) of the preceding project in No. 12).
     ii. Configure the debug configuration.
        a. Click **Project** > **Options….**
        b. Click **I-Jet** > **Setup** and select **Software as Reset.**

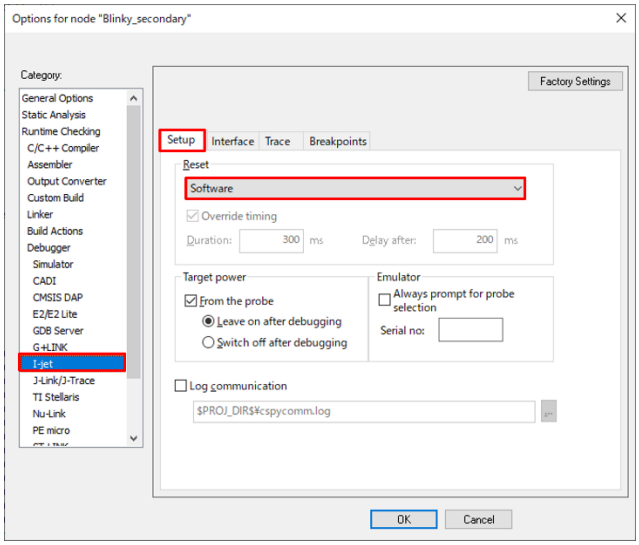


Figure 182. IAR EWARM Project Options for the Secondary or later Project (Reset)

c. ((RZ/T2M or RZ/T2ME) and (secondary or later: CR52 Core1)) Click **Debugger** > **Extra Options** and add **"--macro_param cpu1_enable=1"** to **Command line options: (one per line)**

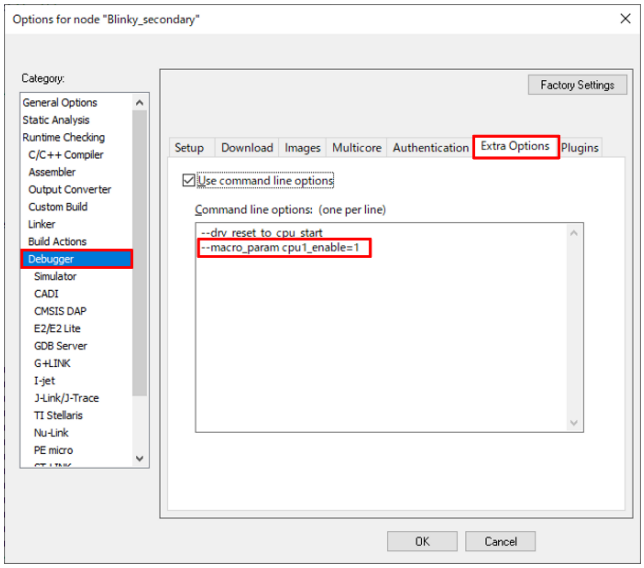


Figure 183. IAR EWARM Project Options for the Secondary Project (Setup Macros)

d. ((RZ/T2H, RZT2N or RZ/N2H) and (secondary or later: CR52 Core1)) Click **Debugger** > **Setup** and check **Use macro file(s)** and add **"$PROJ_DIR$\script\initialization_TCM.mac"** to clear ATCM and BTCM to 0. and enable them.

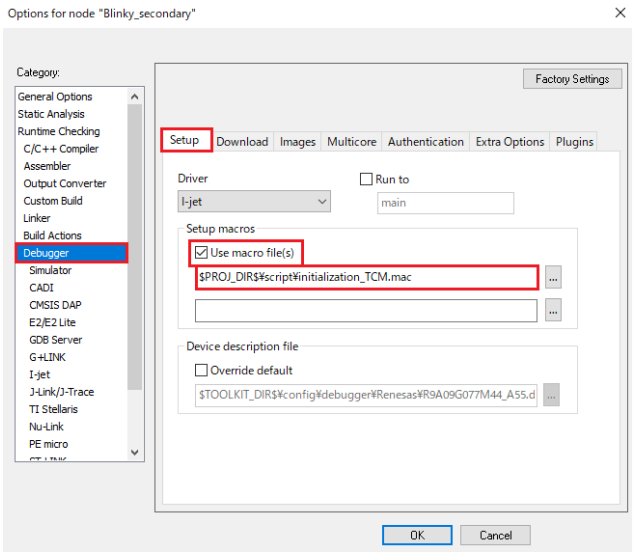


Figure 184. IAR EWARM Project Options for the Secondary Project (Setup Macros)

iii. Build the subsequent project. Click on **Project** -> **Rebuild All** from the menu bar.

4. Configure the debug configuration for the primary project.

i. Click **Project** > **Options….** > 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

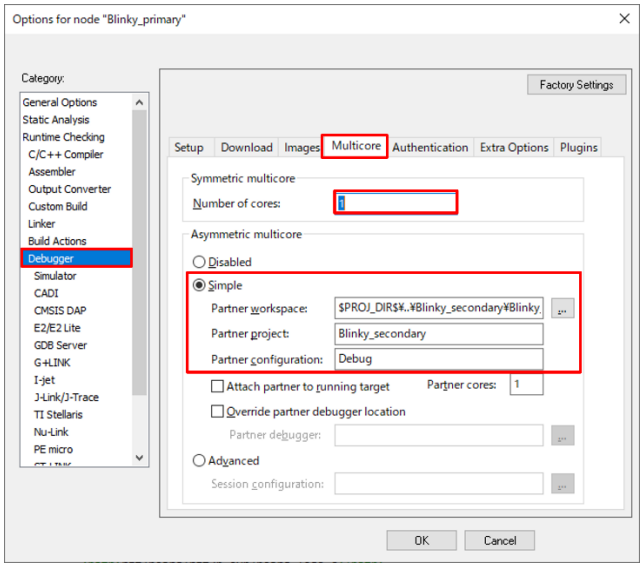


Figure 185. IAR EWARM Project Options for the Primary Project (Multicore)

Note:

For projects with 3 or more cores, configure the following settings.

1. Create multicore_setup.xml and store it in the primary project.
2. Edit multicore_setup.xml as follows:

```
<?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> Relative path from the primary project to the target workspace. 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

3. Click **Project** > **Options….** > **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

ii. Click **OK** and close the Options window.

iii. Click File tab at the top > **Save**

5. Close projects other than the primary project.
6. Debug the primary project (see section 5.3.4 Download & Debug the Project No.1)
   - During debugging, you can Run and Pause the project linked with the following icons.



Figure 186. IAR EWARM Running Projects

7. Run the primary project (see section 5.3.4 Download & Debug the Project No.3).
8. Run the all subsequent projects (see section 5.3.4 Download & Debug the Project No.3).
9. To exit Debug and disconnect from the debugger, click **Debug** > **Stop Debugging** or click on the **Stop** icon.

11.2 Standalone Primary CR52 Core0 (RZ/T2M or RZ/T2ME)

1. Create a primary project (see section 5.3.2 Create a New Project).
2. Build the primary project. Click on **Project** -> **Rebuild All** from the menu bar.
3. Create and build the subsequent project

i. Create a subsequent project (see section 5.3.2 Create a New Project and select the bundle file (.sbd) of the preceding project in No. 12).

ii. Click **Project** > **Options….** > **Build Actions** > **Build Actions Configuration** > **New** and set it as follows:

- 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
- Build order
  - Run after linking

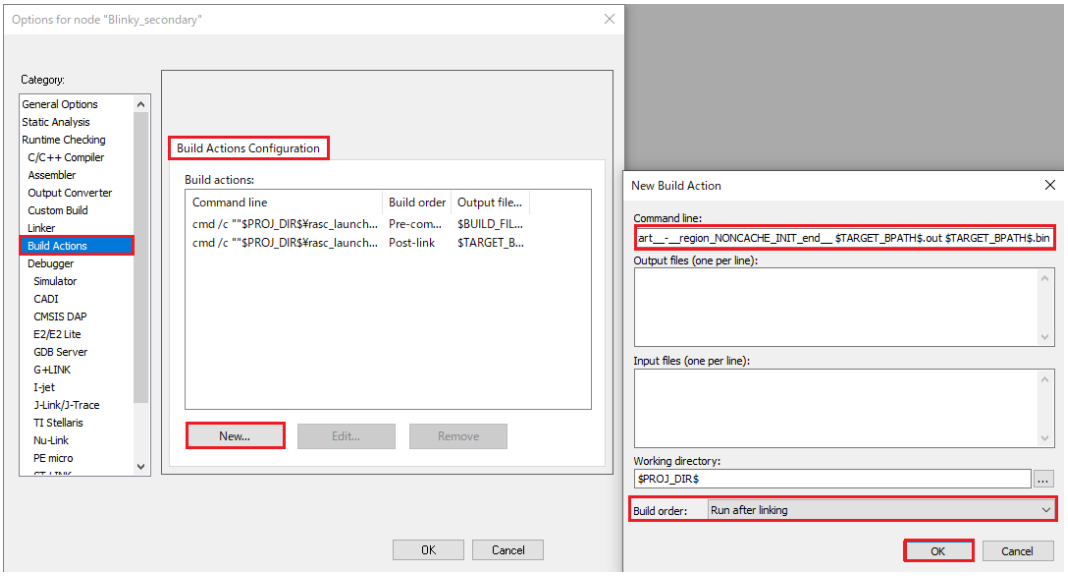


Figure 187. IAR EWARM Project Options for the subsequent Project

iii. Build the subsequent project. Click on **Project** -> **Rebuild All** from the menu bar.

4. Link the object files generated in project to the preceding project.

i. Click **Project** > **Options….** > **Linker** > **Input** and set the file paths of the following object files generated by the subsequent project.

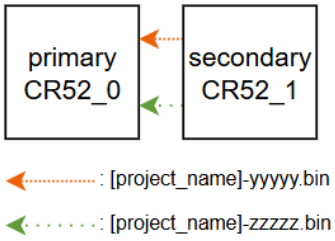


Figure 188. Project Structure Example

Note:

The address portion of the file name varies depending on the project.

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

Table 22. Object files Output to the Debug Folder of the Subsequent Project

Core	Keep symbols (one per line), Symbol, Section	File	Align	Note
CR52 Core1	SECONDARY	$PROJ_DIR$\..\[Project_Name]\Debug\Exe\ [Project_Name]-yyyyy.bin	8	
	SECONDARY _NONCACHE	$PROJ_DIR$\..\[Project_Name]\Debug\Exe\ [Project_Name]-zzzzz.bin	8	When using a non-cache section

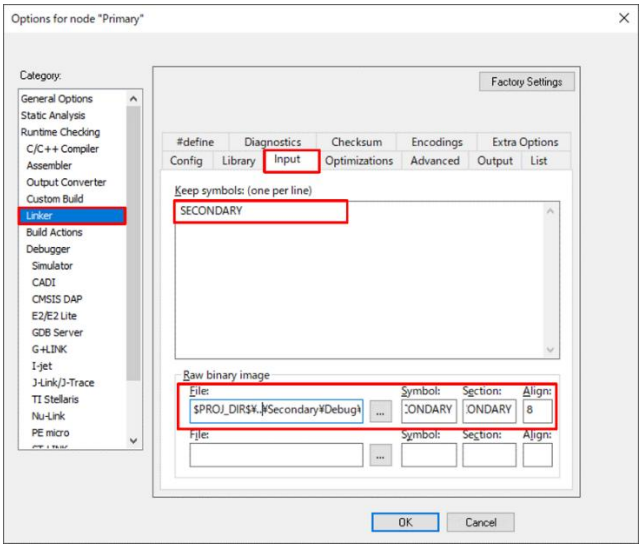


Figure 189. e2studio Build Setting for the Preceding Project

ii. Build the preceding project. Click on **Project** -> **Rebuild All** from the menu bar.

Note:

If you make changes to any project, repeat the following steps for each project, starting with the project you modified and ending with the primary project.

1. Right-click on the project and select **Clean Project**.
2. Build the project. Click on **Project** -> **Rebuild All** from the menu bar.

11.3 Standalone Primary CR52 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H)

1. Create a primary project (see section 5.3.2 Create a New Project).
   i. (primary: CR52 and (secondary or later: CA55)) Set "PLL0 is released from standby state" and enable PLL0 in the Clocks tab of FSP Configuration.

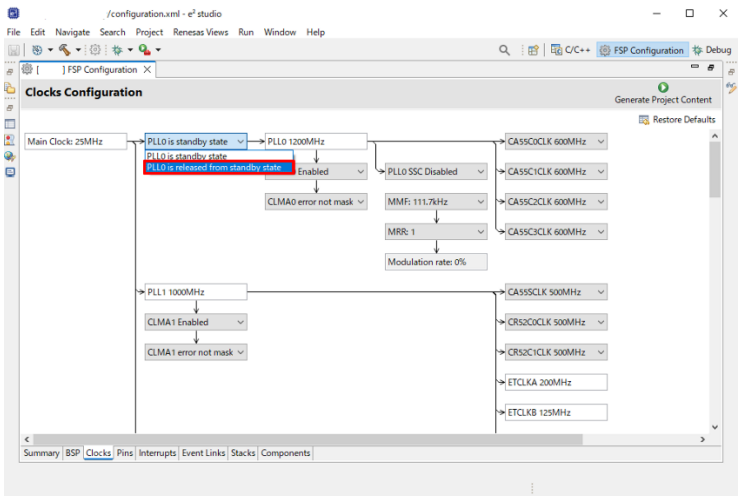


Figure 190. Enable PLL0 in the Primary Project using CR52 Core0

ii. (RZ/T2H and (secondary or later: CA55 Core1)) To blink the LED on the RZ/T2H CA55 Core1, you need to change the pin configuration of the primary project in the FSP Configuration as follows.

a. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Peripherals** -> **Connectivity: SDHI** -> **SDHI1**

b. Change the value of SD1_PWEN to None.

c. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Ports** -> **P08** -> **P08_5**.

d. Change the value of Symbolic Name to LED3.

e. Change the value of Mode to Output mode (Low & Not Into Input)

iii. Click the **Generate Project Content** button in the top right corner of the Project Configuration window to generate your board-specific files.

2. Build the primary project. Click on **Project** -> **Rebuild All** from the menu bar.
3. Create and build the subsequent project

- Repeat the following steps until all required projects have been created.

i. Create a subsequent project (see section 5.3.2 Create a New Project and select the bundle file (.sbd) of the preceding project in No. 12).

ii. Click **Project** > **Options….** > **Build Actions** > **Build Actions Configuration** > **New** and set it as follows:

- 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
- Build order
  - Run after linking

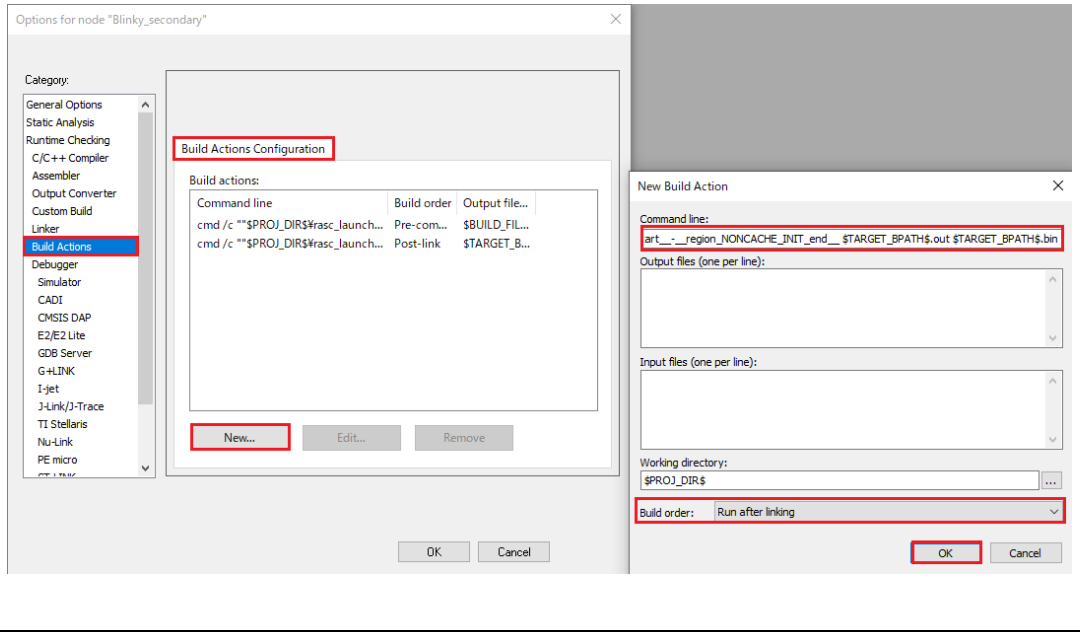


Figure 191. IAR EWARM Project Options for the subsequent Project

iii. Build the subsequent project. Click on **Project** -> **Rebuild All** from the menu bar.

4. Link the object files generated by the subsequent project to the preceding project.

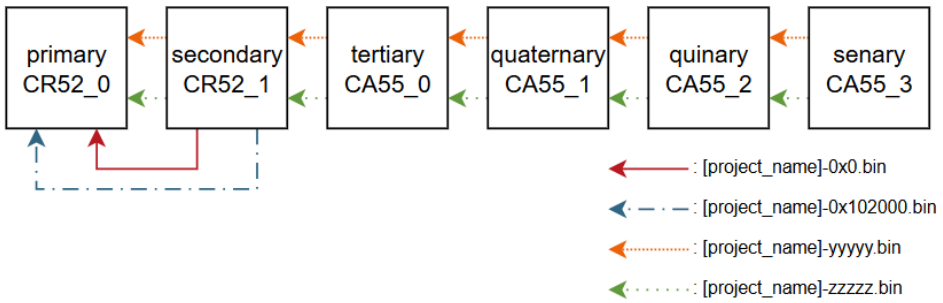


Figure 192. Project Structure Example

Note:

The address portion of the file name varies depending on the project.

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

- Repeat the following steps for each project, starting with the second-to-last project and ending with the primary project.

i. Click **Project** > **Options….** > **Linker** > **Input** and set **Keep symbols** for the sections to be used.

Table 23. Object files Output to the Debug Folder of the Subsequent Project (1/2)

Keep symbols (one per line)	Note
SECONDARY_ATCM_CR521	When using a CR52 Core1 ATCM section
SECONDARY_BTCM_CR521	When using a CR52 Core1 BTCM section
SECONDARY	When using a System RAM section
SECONDARY_NONCACHE	When using a non-cache section

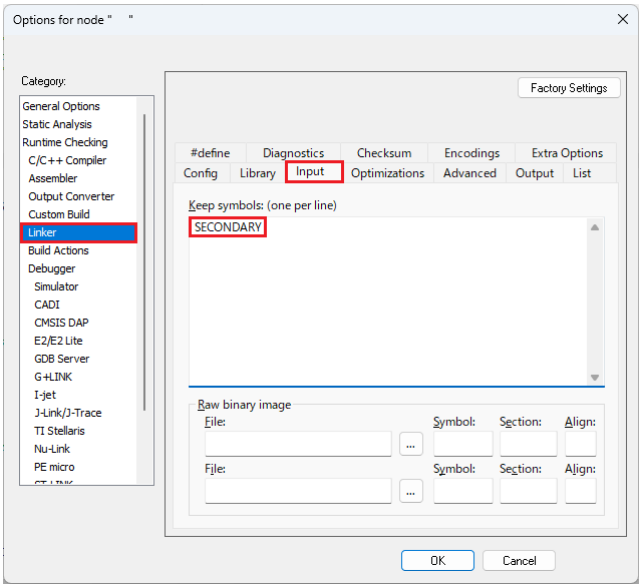


Figure 193. IAR EWARM Project Options for the preceding Project (Input)

ii. Click **Extra Options** and set the file paths of the following object files generated by the subsequent project.

Command line options: (one per line):

--image_input=$PROJ_DIR$\..\[Project_Name]\$CONFIG_NAME$\Exe\[File Name],[Symbol],[Section],[Align]

Table 24. Object files Output to the Debug Folder of the Subsequent Project (2/2)

Core	File Name	Symbol, Section	Align	Note
CR52 Core1	[Project_Name]-0x0.bin	SECONDARY_ATCM_CR521	8	
	[Project_Name]-0x102000.bin	SECONDARY_BTCM_CR521	8	
	[Project_Name]-yyyyy.bin	SECONDARY	8	When using a System RAM section
	[Project_Name]-zzzzz.bin	SECONDARY_NONCACHE	8	When using a non-cache section
CA55 Core0, 1, 2 or 3	[Project_Name]-yyyyy.bin	SECONDARY	8	
	[Project_Name]-zzzzz.bin	SECONDARY_NONCACHE	8	When using a non-cache section

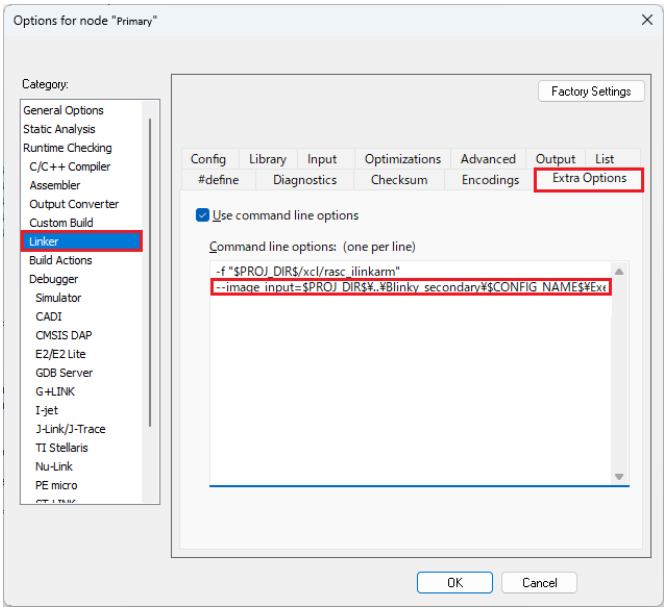


Figure 194. IAR EWARM Project Options for the Preceding Project (Extra Options)

iii. Build the preceding project. Click on **Project** -> **Rebuild All** from the menu bar.

Note:

If you make changes to any project, repeat the following steps for each project, starting with the project you modified and ending with the primary project.

3. Right-click on the project and select **Clean Project**.
4. Build the project. Click on **Project** -> **Rebuild All** from the menu bar.

11.4 Standalone Primary CA55 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H)

1. Create a primary project (see section 5.3.2 Create a New Project).
   i. (RZ/T2H and (secondary or later: CA55 Core1)) To blink the LED on the RZ/T2H CA55 Core1, you need to change the pin configuration of the primary project in the FSP Configuration as follows.
      a. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Peripherals** -> **Connectivity: SDHI** -> **SDHI1**
      b. Change the value of SD1_PWEN to None.
      c. In the Pins tab of the FSP configuration, click **Pin Selection** -> **Ports** -> **P08** -> **P08_5**.
      d. Change the value of Symbolic Name to LED3.
      e. Change the value of Mode to Output mode (Low & Not Into Input)
   ii. Click the **Generate Project Content** button in the top right corner of the Project Configuration window to generate your board-specific files.
2. Build the primary project. Click on **Project** -> **Rebuild All** from the menu bar.
3. Create and build the subsequent project
   - Repeat the following steps until all required projects have been created.
     i. Create a subsequent project (see section 5.3.2 Create a New Project and select the bundle file (.sbd) of the preceding project in No. 12).

ii. Click **Project** > **Options….** > **Build Actions** > **Build Actions Configuration** > **New** and set it as follows:

- 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
- Build order
  - Run after linking

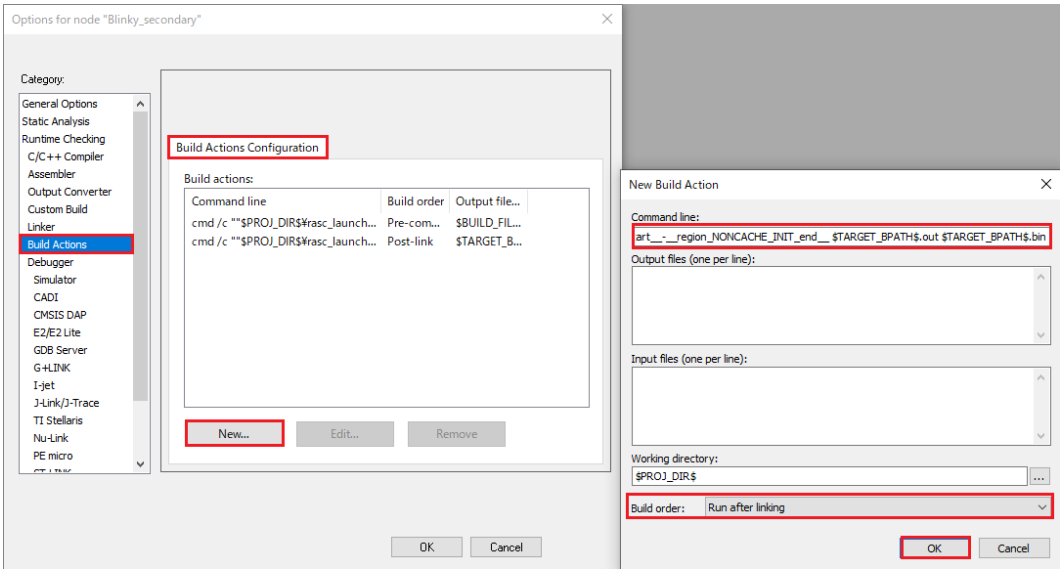


Figure 195. IAR EWARM Project Options for the Succeeding Project

iii. Build the subsequent project. Click on **Project** -> **Rebuild All** from the menu bar.

4. Link the object files generated by the subsequent project to the preceding project.

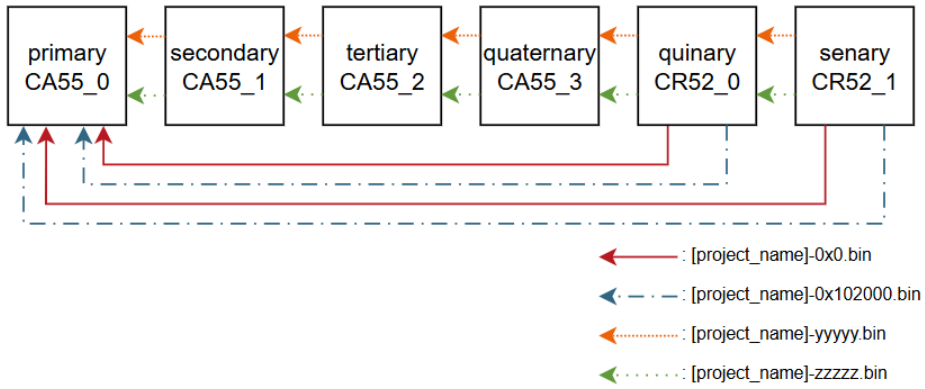


Figure 196. Project Structure Example

Note:

The address portion of the file name varies depending on the project.

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

- Repeat the following steps for each project, starting with the second-to-last project and ending with the primary project.

  i. Click **Project** > **Options….** > **Linker** > **Input** and set **Keep symbols** for the sections to be used.

Table 25. Object Files Output to the Debug Folder of the Subsequent Project (1/2)

Keep symbols (one per line)	Note
SECONDARY_ATCM_CR521	When using a CR52 Core1 ATCM section
SECONDARY_BTCM_CR521	When using a CR52 Core1 BTCM section
SECONDARY	When using a System RAM section
SECONDARY_NONCACHE	When using a non-cache section

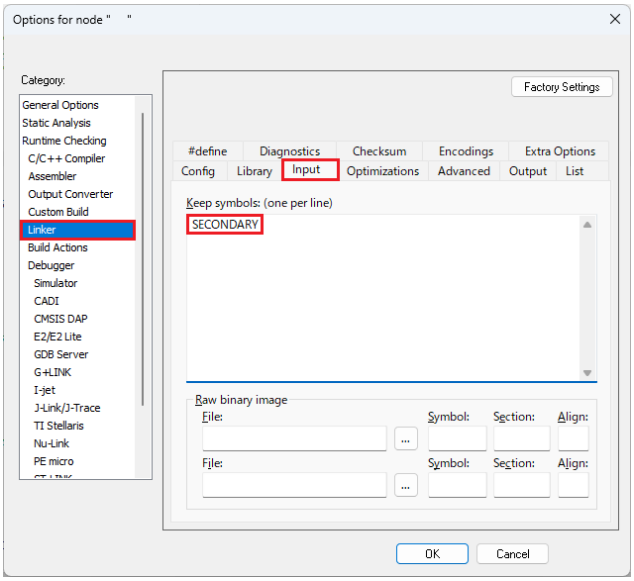


Figure 197. IAR EWARM Project Options for the Preceding Project (Input)

ii. Click **Extra Options** and set the file paths of the following object files generated by the subsequent project.

Command line options: (one per line):

--image_input=$PROJ_DIR$\..\[Project_Name]\$CONFIG_NAME$\Exe\[File Name],[Symbol],[Section],[Align]

Table 26. Object files Output to the Debug Folder of the Subsequent Project (2/2)

Core	File Name	Symbol, Section	Align	Note
CA55 Core1, 2 or 3	[Project_Name]-yyyyy.bin	SECONDARY	8	
	[Project_Name]-zzzzz.bin	SECONDARY_NONCACHE	8	When using a non-cache section
CR52 Core0	[Project_Name]-0x0.bin	SECONDARY_ATCM_CR520	8	
	[Project_Name]-0x102000.bin	SECONDARY_BTCM_CR520	8	
	[Project_Name]-yyyyy.bin	SECONDARY	8	When using a System RAM section
	[Project_Name]-zzzzz.bin	SECONDARY_NONCACHE	8	When using a non-cache section
CR52 Core1	[Project_Name]-0x0.bin	SECONDARY_ATCM_CR521	8	
	[Project_Name]-0x102000.bin	SECONDARY_BTCM_CR521	8	
	[Project_Name]-yyyyy.bin	SECONDARY	8	When using a System RAM section
	[Project_Name]-zzzzz.bin	SECONDARY_NONCACHE	8	When using a non-cache section

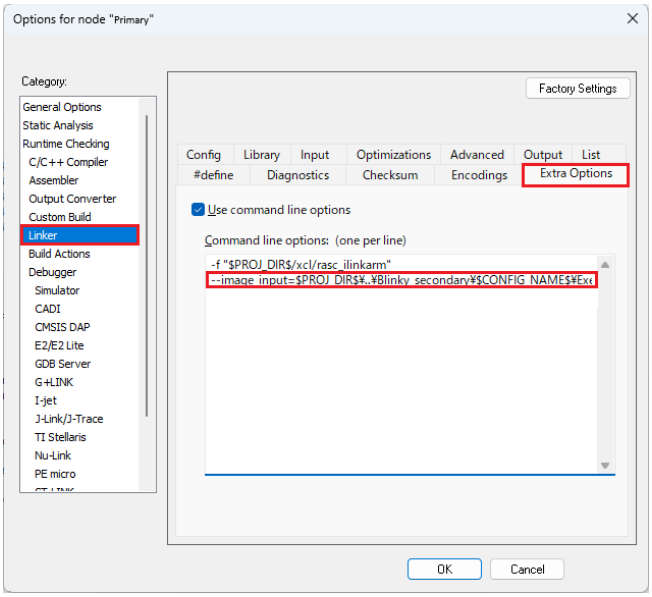


Figure 198. IAR EWARM Project Options for the Preceding Project (Extra Options)

iv. Build the preceding project. Click on **Project** -> **Rebuild All** from the menu bar.

Note:

If you make changes to any project, repeat the following steps for each project, starting with the project you modified and ending with the primary project.

1. Right-click on the project and select **Clean Project**.
2. Build the project. Click on **Project** -> **Rebuild All** from the menu bar.

12. Appendix. How to Create a CMake Project using e²studio

This section describes the steps to create a CMake project using e²studio.

12.1 Create a New Project

1. Create a project (see section 4.3 Create a New Project for Blinky and select "**e²studio CMake**" as **IDE Project Type in No.9**).
2. For the project you created, make the following additional settings:
   i. Open cmake/GeneratedCfg.cmake.
   ii. Copy the value of RASC_TARGET_DEVICE.
   - For example, if the following setting exists: "set(RASC_TARGET_DEVICE R9A09G077M44_CR52_0)", copy "R9A09G077M44_CR52_0".

   iii. Open CMakeLists.txt in the project root folder.
   iv. On line 7, immediately after "set(CMAKE_SYSTEM_PROCESSOR ...)", add the following:
   - Set(DEVICE [Value copied in No.2.ii])

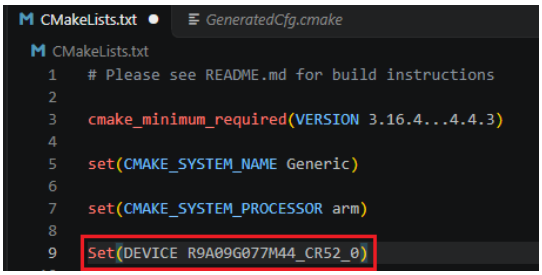


Figure 199. Additional Settings for CMakeList.txt (DEVICE)

3. Build the project (see section 4.4 Build the Blinky Project).

12.2 Link Object Files

Refer to 10 Appendix. How to Create and Debug FSP Projects for Multiprocessing for e2studio for instructions on creating a multiprocessing project.

Link the object files generated by the subsequent project using the following procedure.

1. Open CMakeLists.txt in the project root folder.
2. Second to last line, immediately after "# If required user can amend RASC ...", add the following:
   - list(APPEND SECONDARY_OBJECTS "${CMAKE_CURRENT_SOURCE_DIR}/../[Subsequent Project Name]/build/${CMAKE_BUILD_TYPE}/[Object File Name]")

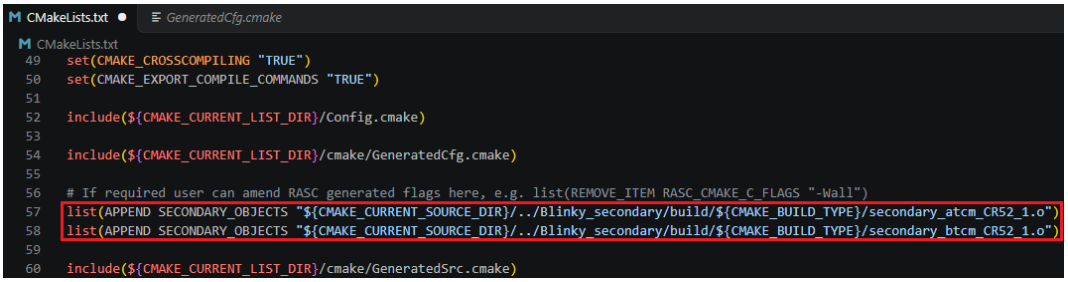


Figure 200. Additional Settings for CMakeList.txt (SECONDARY_OBJECTS)

3. Right-click on the project and select **Clean Project**.
4. Build the project (see section 4.4 Build the Blinky Project).

12.3 Debug the Project

Debugging is not supported in e²studio.

You can download the generated ELF file to flash memory and verify its operation. (See 8.2 How to Program an ELF File)

13. Appendix. How to Create a CMake Project using FSP SC

This section describes the steps to create a CMake project using FSP SC and build it using Visual studio Code. If the procedure is preceded by (XXX), it is executed only if the condition is met.

13.1 Create a New Project

Create a project (see section 5.3.2 Create a New Project and select "**CMake"** as **IDE Project Type in No.9**).

13.2 Setup Visual studio Code

The following describes the procedure for Windows.

1. Install Visual studio Code.
   - URL: https://code.visualstudio.com/download?_exp_download=d53503e735
2. Download Ninja (ninja-win.zip) and unzip to any path.
   - URL: https://github.com/ninja-build/ninja/releases
3. Download CMake and install (or unzip) it to any path.
   - URL: https://cmake.org/download/
4. Set the following in the Windows Environment Variables.
   - ARM_GCC_TOOLCHAIN_PATH: [aarch32 toolchain path] (e.g: ***\Arm GNU Toolchain arm-none-eabi\13.3 rel1\bin)
   - ARM64_GCC_TOOLCHAIN_PATH: [aarch64 toolchain path] (e.g: ***\Arm GNU Toolchain aarch64-none-elf\13.2 Rel1\bin)
   - ARM_IAR_TOOLCHAIN_PATH: [IAR EWARM toolchain path] (e.g: ***\IAR Systems\Embedded Workbench 9.60.3\arm\bin)
   - RASC_EXE_PATH: [FSP SC path] (e.g: ***\eclipse\rasc.exe)
   - Path:
     - [Path to Ninja.exe] (e.g: ***\ninja-win)
     - [Path to CMake] (e.g: ***\CMake\bin)

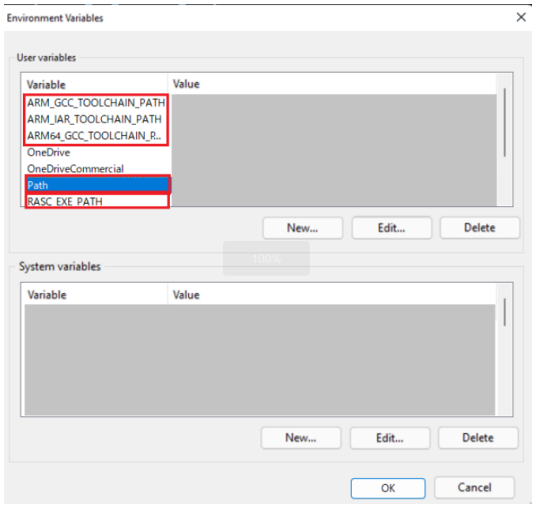


Figure 201. Set System Environment Variables

5. Open Visual studio Code.
6. Click **Extensions** in the Activity Bar, find CMake Tools, and install it.

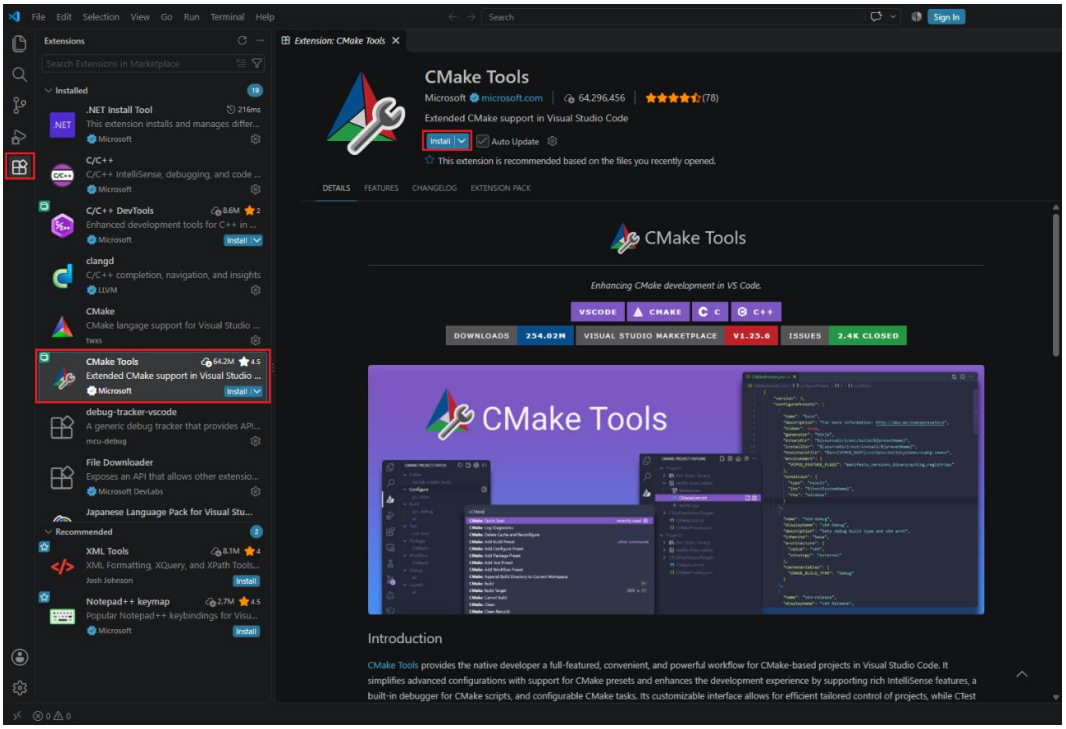


Figure 202. Install CMake Tools in Visual studio Code

13.3 Build the Project

1. Open the project you created in Visual studio Code.
   - Click **File** in the menu bar, then click **Open Workspace from File...** and select the .code-workspace file of the project you created.

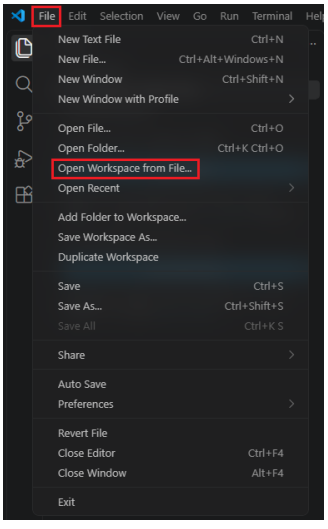


Figure 203. Open Workspace in Visual studio Code

2. (Toolchains is IAR toolchain for ARM) Open CMakeLists.txt in the project root folder and make the following additional settings:

- Second to last line, immediately after “# If required user can amend RASC ...”, add the following:
  - list(APPEND RASC_CMAKE_EXE_LINKER_FLAGS "--entry=system_init")
  - (CA55) list(REMOVE_ITEM RASC_CMAKE_C_FLAGS "--thumb")



Figure 204. Additional Settings for CMakeList.txt

3. Click **CMake** in the Activity Bar, then click **Select a Kit** in Configure and select **ARM IAR - Ninja**. (When Toolchains is GCC toolchain for ARM or GCC ARM A-Profile Toolchain for ARM, select **ARM GCC - Ninja**)

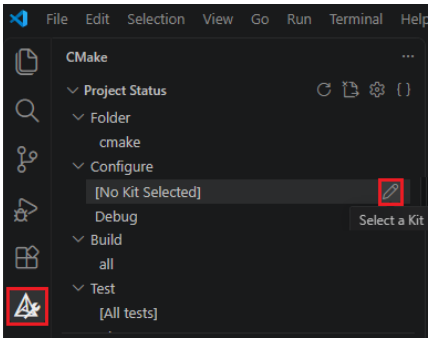


Figure 205. Select a Kit (1/2)

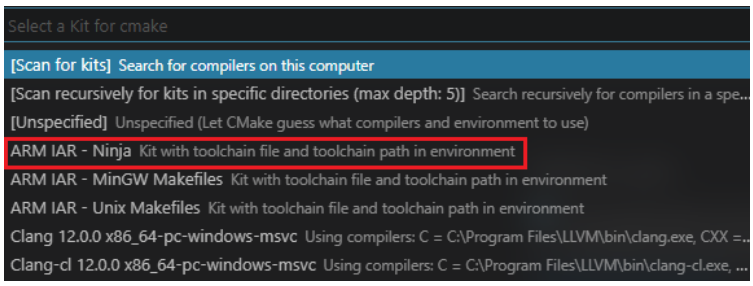


Figure 206. Select a Kit (2/2)

4. Click **Build** and build the project.

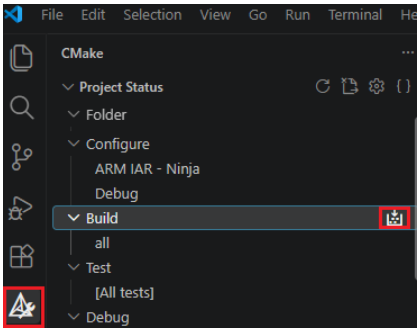


Figure 207. Build the Project

13.4 Link Binary Files

Refer to 11 Appendix. How to Create and Debug FSP Projects for Multiprocessing for IAR EWARM for instructions on creating a multiprocessing project.

Link the object files generated from the subsequent project using the following procedure.

1. Open CMakeLists.txt in the project root folder.
2. Second to last line, immediately after “# If required user can amend RASC ...”, add the following:
   - list(APPEND SECONDARY_OBJECTS "[Section]|${CMAKE_CURRENT_SOURCE_DIR}/../[Subsequent Project Name]/build/${CMAKE_BUILD_TYPE}/[Binary File Name]")

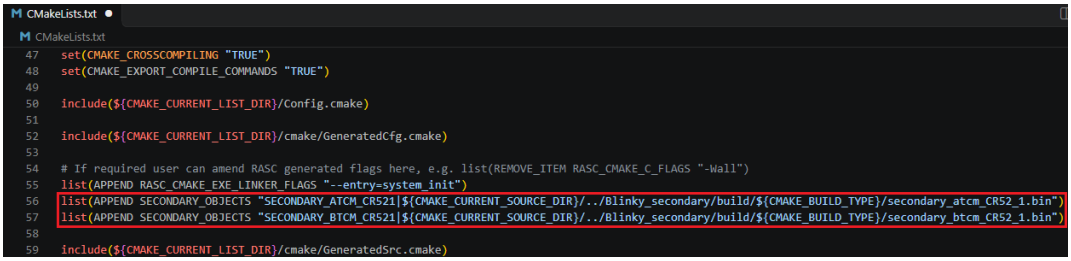


Figure 208. Additional Settings for CMakeList.txt (SECONDARY_OBJECTS)

3. Build the project (see section 13.3 Build the Project No.4).

13.5 Debug the Project

Debugging is not supported in Visual studio Code.

You can download the generated ELF file to flash memory and verify its operation. (See 8.2 How to Program an ELF File)

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. • Added the software environment on which FSP projects are verified.
		p.37	Added the “3.5.1 RSK+RZN2L” section.
		p.49	Updated the Windows PC requirements.
		p.74	Update “Prerequisites” section • Added the note regarding the patch for debugging RZ/N2L FSP project on EWARM.
		p.75	Updated “Create a New Project” section. • Added installation path of FSP SC. • Added some steps for creating an EWARM project. • Added Note subsection for debugging RZ/N2L EWARM project.
		p.101	Updated “Selecting a Board and Toolchain” section. • Added a detailed explanation of how to select the Board and Device for creating an FSP project.
		p.115	Updated “7.1 Known Issues” chapter.
		p.127	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section.
		-	Added “Appendix: How to update J-Link DLL files in e²studio” chapter.
		p.158	Added “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。”
1.02	Oct.31.22	-	Updated documentation for RZ/T2M FSP v1.1.0. • Removed contents for RZ/T2M FSP v1.0.0
		All	Updated minor issues. • Fixed minor typo. • Adjusted page breaks.
		p.33	Updated “3.3.1 SEGGER J-Link” section. • Updated the FSP version and J-Link version for RZ/T2M • Added the notification that J-Link OB S124 requires the firmware update to debug the RZ/T2M FSP project. • Added the link to Renesas Knowledge Base, which explains how to update J-Link DLL in e²studio.
		p.45	Added “5.3.2.2 NOTE: Configure IAR EWARM Project [RZ/T2M, RZ/T2L]” section.
		p.115	Updated “7.1 Known Issues” chapter. • Remove some limitations regarding RZ/T2M

Rev.	Date	Description	
		Page	Summary
		p.127	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. • Added a new limitation of e²studio regarding J-Link OB S124 firmware version. • Added the link to explain how to update the J-Link DLL.
		-	Removed “Appendix: How to update J-Link DLL files in e²studio” chapter.
1.03	Dec.23.22	-	Updated documentation for RZ/N2L FSP v1.1.0. • Removed contents for RZ/N2L FSP v1.0.0
		All	Updated minor issues. • Fixed minor typo.
		p.69	Updated “7.1 Known Issues” chapter. • Removed some limitations regarding RZ/N2L
		p.73	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. Removed some limitations regarding RZ/N2L
		p.75	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Added new procedure for RZ/N2L FSP v1.1.0.
1.04	Mar.23.23	All	Updated documentation for RZ/T2 FSP v1.2.0. • Removed contents for RZ/T2M FSP v1.1.0. • Added contents for RZ/T2L.
		p.1	Added video contents website link.
		p.9	Updated “3.3.1 SEGGER J-Link” section. • Updated the FSP version and J-Link version for RZ/T2M and RZ/T2L. • Removed the notification that J-Link OB S124 requires the firmware update to debug RZ/T2M FSP project.
		p.10-12	Updated “3.4.1 RSK+RZT2M” section. • Added explanation that other boot mode board settings refer to the RSK User’s Manual. • Modified figure names and added a table title.
		p.13-15	Added “3.4.2 RSK+RZT2L” section
		p.16-18	Updated “3.5.1 RSK+RZN2L” section. • Corrected board name. • Added explanation that other boot mode board settings refer to the RSK User’s Manual. • Modified figure names and added a table title.
		p.23	Updated “4.3 Create a New Project for Blinky” section. • Added RSK+RZT2L Board Setting.
		p.26	Updated “4.3.5 Where is main()?” section. • Corrected product group name.

Rev.	Date	Description	
		Page	Summary
		p.32	Updated “4.5.4 Change CPSR Register Value” section. • Revised description. • Added description of how to automatically change CPSR register value.
		p.33	Updated “4.6 Run the Blinky Project” section. • Removed LEDs working in CPU1 project.
		p.34	Updated “5.3.1 Prerequisites” section. • Added note for RZ/T2L patch file.
		p.37	Updated “5.3.2 Create a New Project” section. • Added RSK+RZT2L Board Setting.
		p.50	Updated “5.3.4 Download & Debug the Project” section. • Removed LEDs working in CPU1 project.
		p.69	Updated “6.5 Adding and Configuring HAL Drivers” section. • Added a table title.
		p.71-73	Updated “7.1 Known Issues” section. • Added a table “List of Known Issues.” • Numbered each issue. • Removed the issue of adding "r_dsmif" alone • Updated issue contents that the BSP properties are sometimes configured incorrectly • Removed Ethernet SELECTOR issue.
		p.74-77	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. • Added a table “List of Tool Software Limitations.” • Numbered each limitation. • Added new limitation of applying RZ/T2 FSP v.1.2.0 pack.
		p.78	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section • 1. (Both e²studio and EWARM) Insert the loop part in startup.c. Added e²studio 2023-01 to the table. • 3. (e²studio ONLY) Apply a macro file for RZ/N2L FSP v1.1.0 xSPI0 x1 boot mode. • Added direct download URL of RZ/N2L patch file.
1.05	Jun.30.23	All	Updated documentation for RZ/N2L FSP v1.2.0. • Removed contents for RZ/N2L FSP v1.1.0
		p.9	Updated “3.3.1 SEGGER J-Link” section. • Updated the FSP version and e²studio version for RZ/N2L.
		p.30	Updated “4.5.2 Debug Steps” section. • Added description of how to automatically change CPSR register value for RZ/N2L and e²studio 2023-04.
		p.33	Updated “4.5.4 NOTE: Change CPSR Register Value [RZ/T2M, RZ/T2L]” section. • Changed section title to limit the target device

Rev.	Date	Description	
		Page	Summary
		p.35	Updated “5.3.1 Prerequisites” section. Removed EWARM Patch for RZ/N2L
		p.72-77	Updated “7.1 Known Issues” chapter. • Updated table “List of Known Issues” to add new issues and add N2L as the target device for No.2 • Added a new issue related to BSP configuration when changing board settings. • Added a new issue related to the FSP module FreeRTOS issue.
		p.78	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Removed some limitations regarding the breakpoint
		p.82	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” • Updated IDE version in the table for including e^{2}studio 2023-04
1.06	Sep.8.23	All	Updated documentation for RZ/T2 FSP v1.3.0. • Removed contents for RZ/T2 FSP v1.2.0 • Changed GNU ARM Embedded Toolchain to version 12.2.1.arm-12-24.
		p.6	Updated “1.3.2 FSP Documentation” section. • Added note for RZ/N2L FSP documentation.
		p.26	Updated “4.3.6 Blinky Example Code” section. • Changed the processing of the blinky template code.
		p.28	Updated “4.5.2 Debug Steps” section. • Added reset setting of debug configuration for RAM execution without flash memory.
		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. • Changed the processing of the blinky template code.
		p.68-73	Updated “7.1 Known Issues” section. • Updated table “List of Known Issues” to add new issues. • Added new issues related to Pins configuration. • Added a new issue of a warning message when building “r_gmac” with the GCC compiler.
		p.75	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Added “Smart Configurator” section. • Added a new limitation of displaying memory region usage
		p.80	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. • Removed limitation related to reset when using e^{2}studio
		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 e²studio version for RZ/N2L.
		p.80	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. • Change the number of items. • Removed limitation related to RZ/N2 FSP v1.2.0 and J-Link V7.80b only
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 “7.1 Known Issues” section. • Moved the position of FSP Configurations and FSP Modules descriptions to the beginning of the chapter. • Added column “Category” to the List • Removed category headings (FSP Configurations, Stacks Configuration, FSP Module, BSP Configuration, …) • Added item “Category” to the description of each Known Issue. • Grayed out items where issues have been resolved. • Added description to workaround of No. 3. • Added RZ/T2L as the target device to No. 5 • Added RZ/T2M and RZ/T2L as target device to No. 6 • Corrected instructions in the code of No. 14.
		p.76-80	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Added column “Category” to the List • Removed category headings (Smart Configurator, FSP Smart Configurator, e²studio, …) • Added item “Category” to the description of each Tool Software Limitations. • Grayed out items where limitations have been resolved.
		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. • Removed contents for RZ/T2 FSP v1.3.0
		p.1	List Target Device separately for each series.
		p.6	Updated “1.3 Related Documentation Files” section. • List Target Device separately for each series.
		p.9	Updated “3.3.1 SEGGER J-Link” section. • List Target Device separately for each series in a table.
		p.10-18	Updated “3.4 RZ/T Series Board Setup” section and added “3.5 RZ/N Series Board Setup” section. • Each series was divided into separate explanatory chapters. • Delete unnecessary figure descriptions

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. • Added description of a multiprocessing.
		p.28	Updated “4.5.2 Debug Steps” section. • Added description of a multiprocessing.
		p.31	Updated “4.6 Run the Blinky Project” section. • Added LED2-3 of RSK+RZ/T2M for CPU1 core. • Added descriptions to suspend program execution and exit debug mode.
		p.32	Added “4.7 Debug and Run for Multiprocessing” section.
		p.33	Added “4.8 Import the Project” section.
		p.37	Updated “5.2 Tutorial Blinky” section. • Added description of a multiprocessing.
		p.38	Updated “5.3.2 Create a New Project” section. • Added description of a multiprocessing.
		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. • Moved text to “5.3.3.2 エラー! 参照元が見つかりません。” section.
		p.43	Added “5.3.3.1 NOTE: Build settings [Only Multiprocessing]” section.
		p.47	Added “5.3.3.2 エラー! 参照元が見つかりません。” section.
		p.48	Updated “5.3.4 Download & Debug the Project” section. • Added description of a multiprocessing. • Added LED2-3 of RSK+RZ/T2M for CPU1 core.
		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 “7.1 Known Issues” chapter. • Resolved issues No. 2, No. 6, No. 8, No. 9, No. 10 and No. 11. • Removed RZ/T2M and RZ/T2L as target device from No. 12 • Added new issue No. 13, No. 14, and No. 15.
		p.89	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Added new limitation No. 9 and No. 10.
		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. • Changed the program execution start position setting. • Corrected reset setting. Added images for the settings screen.
		P.73	Updated “7.1 Known Issues” chapter. • Resolved issue No. 12 • Added RZ/N2L as target device from No.13 and No.14 • Added new issues No. 16 and No. 17.
		p.84	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Added RZ/N2L as target device from エラー! 参照元が見つかりません。.
		p.97	Updated “Appendix How to Change Boot Mode of the FSP Project” chapter. • Added new step 4
		p.99	Updated “Appendix. How to Debug FSP multiprocessing projects with Flash Boot Mode” section. • Changed the program execution start position setting. • Modified explanation of debugging sequence.
1.11	Jun.28.24	All	Updated documentation for RZ/T2 FSP v2.1.0. • Removed contents for RZ/T2 FSP v2.0.0
		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. • Added explanation of notes when using FSP software modules.
		p.10	Updated “3.4.1.1 Boot Mode” section. • Added note for board setting.
		p.13	Updated “3.4.2.1 Boot ModeBoot Mode” section. • Added note for board setting.
		p.15	Added “3.4.3 RSK+RZT2ME” section.
		p.16	Updated “3.5.1.1 Boot ModeBoot Mode” section. • Added note for board setting.
		p.38	Updated “5.3.2 Create a New Project” section. • Added IDE Project Type setting for newer versions of FSP SC.

Rev.	Date	Description	
		Page	Summary
		p.43	Updated “5.3.3.2 Build for Multiprocessing” section. • Removed Build Actions setting. Added Make before debugging setting.
		p.54	Updated “6 FSP Configuration Users Guide” section. Updated figures with new tool screens.
		P.74	Updated “7.1 Known Issues” chapter. • Resolved issues No. 16 and No. 17 Added new issues No. 18 to No. 24.
		p.85	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Removed RZ/T2M and RZ/T2L as target device from No. 6. Added new issue No. 11.
		p.102	Updated “Appendix How to Change Boot Mode of the FSP Project” section. • Added note on FSP version changes
1.12	Nov.26.24	All	Updated documentation for RZ/T2 FSP v2.2.0. • Removed contents for RZ/T2 FSP v2.1.0
		All	Added the RZ/T2H as target device and CA55 core support.
		p.6	Updated “1.3.2 FSP Documentation” section. • Removed Note for RZ/N2L about the documentation issue.
		p.8	Updated “Set up the Evaluation Board” chapter. • Changed the boards designations.
		p.9	Updated “3.3.1 SEGGER J-Link” section. • Added how to update J-Link firmware.
		p.16	Added “3.4.4 RZ/T2H Evaluation Board” section.
		p.24	Updated “4.3 Create a New Project for Blinky” section. • Modified to a generic description that does not specify which cores are used in multiprocessing. • Updated tool screen images. • Added tables describing the settings for each project.
		p.31	Added “4.4.2 Build for Multiprocessing” section.
		p.32	Updated “4.5.2 Debug Steps” section. • Added tables describing the settings for each project. • Added a note for debugging flash boot project.
		p.37	Updated “4.7 Debug and Run for Multiprocessing ” section. • Clarified explanation of step 2. • Added supplemental information on behavior to step 5.
		p.44	Updated “5.3.2 Create a New Project” section. • Modified to a generic description that does not specify which cores are used in multiprocessing. • Updated tool screen images. • Added tables describing the settings for each project.

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. • Swapped the order of “5.3.3.1 NOTE: Build settings [Only Multiprocessing]” and “5.3.3.2 Build” chapters. • Added how to build for multiprocessing in “5.3.3.1 Build” section. • Renamed “5.3.3.2 Build for Multiprocessing” section.
		p.55	Updated ““5.3.3.2 Build for Multiprocessing” section. • Added the running setting to step 1. • Added the extra options to step 2. • Removed the running setting and tools options from step 3. 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. • Added a note for debugging flash boot project.
		p.62	Updated ““5.3.5 Debug for Multiprocessing エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. • Added multicore debugging setting as step 2. • Added supplemental information on behavior to step 5. • Added how to debug when changing the project.
		p.67	Updated “6.2 Create a Project” section. • Updated tool screen images and menu names.
		p.71	Added “6.2.4 Duplication of Resources” section.
		p.82	Updated “7.1 Known Issues” section. • Added RZ/T2H as target device of No. 3, No. 14, No. 27, No. 28. • Added new issues No. 29 to No. 38. • Resolved issues No. 4, No. 13, No. 18, No. 19 and No. 20.
		p.106	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Added RZ/T2H as target device of No. 4, No. 7, No. 9 and No. 10. • Added new issues No. 12 to No. 19. • Resolved issues No. 1 and No. 11.
		p.117	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. • Add Note for multiprocessing projects. • Corrected boot mode name from xSPI0 to xSPI. • Updated path description. • Add a column for cores to the table.
		-	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 e2studio” 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 “Configure IAR EWARM Project [RZ/T2H, RZ/T2N and RZ/N2H Only]” section.
		p.86	Updated “7.1 Known Issues” section. • Resolved issues No. 21, No. 22, No. 23, No. 24. • Added RZ/N2H as target device of No. 3, No. 14, No. 27 to No. 36. • Added RZ/N2L as target device of No. 27, No. 28. • Added new issues No. 38 to No. 43. Replaced Smart Configuration with SC.
		p.113	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Added RZ/N2H as target device of No. 4, No. 7, No. 9, No. 10 and No. 12 to No. 19. • Added new issues No. 20, No. 21. • Resolved issues No. 6. • Updated the description of issues No. 12. • Replaced Smart Configuration with SC.
		p.125	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. • Moved Note for multiprocessing projects to No. 1 in the same chapter. • Removed CA55 NOR flash boot from the table in No. 1. • Added No. 2 for RZ/N2H. • Added note for debug FSP project.
		p.127	Updated “Appendix. How to Erase Flash Memory” • Added Device Type Name for RZ/N2H. • Added External Address Space for RZ/N2H.
1.14	Feb.28.25	All	Updated documentation for RZ/T2 FSP v2.3.0. • Removed contents for RZ/T2 FSP v2.2.0 • Unified wording for Smart Configurator (SC) and IAR I-jet.
		p.25	Updated “3.2.1 Windows PC Requirements” section. • Updated Windows PC requirements to use e^{2}studio.
		p.25	Updated “3.2.3 Choosing a Toolchain” section. • Added a table of toolchain version for each FSP.
		p.26	Updated “4.3 Create a New Project for Blinky” section. • Added a table of the project creation procedure. • Added a table of selecting a bundle file to procedure No. 9.

Rev.	Date	Description	
		Page	Summary
		p.47	Updated “5.3.2 Create a New Project” section. • Added a table of the project creation procedure. • Updated a description of selecting a bundle file in procedure No. 9. • Added a table of selecting a bundle file to procedure No. 9.
		p.55	Updated “5.3.2.2 Configure IAR EWARM Project [RZ/T2H, RZ/T2N and RZ/N2H Only]” section. • Updated a description of error conditions. Removed step 5 to set Build Actions.
		p.86	Updated “7.1 Known Issues” chapter. • Resolved issues No. 3, No. 25 to 27, No. 29, No. 31, No. 36 to 43 for RZ/T series devices. Added new issue No. 44.
		p.118	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Resolved limitation No. 17 for RZ/T series devices. Added new limitation No. 22.
		p.141	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e2studio” chapter. • Added “Multiprocessing with 3 or more cores エラー! 参照元が見つかりません。” section.
		p.146	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM” chapter. Added “Multiprocessing with 3 or more cores” section.
1.15	Apr.25.25	All	Updated documentation for RZ/N2 FSP v2.2.0. • Removed contents for RZ/N2 FSP v2.1.0
		p.86	Updated “7.1 Known Issues” chapter. • Resolved issues No. 3, No. 27, No. 29, No. 31, No. 37 to 43. • Added new issue No. 45, No. 46, No. 47, No. 48.
		p.118	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Removed RZ/N2H as target device from No. 13, No. 14. • Resolved limitation No. 17. • Added new limitation No. 23.
		p.131	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” section. • Updated a description of the CA55 xSPI boot from the table in No. 1
		p.147	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e2studio” chapter. • Updated “Multiprocessing with 3 or more cores” section for flash boot.
1.16	Jul.9.25	All	Updated documentation for RZ/T2 FSP v3.0.0. • Removed contents for RZ/T2 FSP v2.3.0
		p.25	Updated “3.2.3 Choosing a Toolchain” section. • Added version name displayed in arm Developer website.

Rev.	Date	Description	
		Page	Summary
		p.35	Updated “4.5.2 Debug Steps” section. • Added script file setting for TCM initialization.
		p.60	Updated “5.3.3.2 Build for Multiprocessing” section. • Added macro file setting for TCM initialization.
		p.88	Updated “7.1 Known Issues” chapter. • Resolved issues No. 14, No. 32, No. 46, No. 47 for RZ/T series devices. Resolved issues No.45, No. 48.
		p.122	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. Resolved limitations No. 13, No. 14.
		p.135	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Removed column e²studio 2022-04 2024-07. • Updated black text code in system_init(). 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 e2studio” 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. • Added procedure for RZ/T2 FSP v3.0.0 to meet that specification.
1.17	Aug.29.25	All	Updated documentation for RZ/N2 FSP v3.0.0. • Removed contents for RZ/N2 FSP v2.2.0.
		p.35	Updated “4.5.2 Debug Steps” section. • Added the RZ/N2H as target device.
		p.60	Updated “5.3.3.2 Build for Multiprocessing” section. • Added the RZ/N2H as target device in step 2.
		p.88	Updated “7.1 Known Issues” chapter. • Resolved issues No. 14, No. 32, No. 46, No. 47. • Added new issue No. 49.
		p.122	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Resolved limitations No. 20.

Rev.	Date	Description	
		Page	Summary
		p.135	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Removed Note for multiprocessing projects in No. 1.
		p.143	Updated “9.1 How to Change Boot Mode of the FSP Project” chapter. • Inserted the description in step 3.
		p.145	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e2studio” chapter. • Removed “For RZ/N2 FSP v2.2.0”. • Updated Multiprocessing with 2 cores for RZ/N2H as target device. • Updated Multiprocessing with 3 or more cores for RZ/N2H as target device.
		p.153	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM” chapter. • Removed “For RZ/N2 FSP v2.2.0” • Updated Multiprocessing with 2 cores for RZ/N2H as target device. 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 “7.1 Known Issues” chapter. Resolved issue No. 49 for RZ/T series devices.
		p.121	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. Resolved limitations No. 16 and No. 23 for RZ/T series devices.
		p.134	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Add the explanation of eSD and eMMC boot modes. • Add Note for multiprocessing projects in No. 1. • Corrected the user program stop time. The function to add the loop part was changed to R_BSP_WarmStart_StackLess() for RZ/T2 FSP v3.1.0.

Rev.	Date	Description	
		Page	Summary
		p.145	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e2studio” chapter. • Added Prevent Releasing the Reset of the CM3 Core setting for CR52 CPU1 to No. 10 of Multiprocessing with 2 cores. • 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. 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.
		p.154	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM” chapter. 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.
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 “7.1 Known Issues” chapter. Resolved issue No. 49.
		p.121	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. • Resolved limitations No. 16, No. 23. Added new issue No. 24.
		p.134	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. Removed “For RZ/N2 FSP v3.0.0”. 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 e2studio” 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	Added new section “5.3.2.1 NOTE: Configure IAR EWARM Project”
		p.90	Updated “7.1 Known Issues” chapter. Added new issue No. 50.

Rev.	Date	Description	
		Page	Summary

		p.126	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” 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 e²studio 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 “7.1 Known Issues” chapter. Added new issues No. 51, 52.
		p.154	Updated “エラー! 参照元が見つかりません。 エラー! 参照元が見つかりません。” chapter. Resolved issue No.24.
4.20	Aug.11.26	All	Updated naming from “RZ/T2, RZ/N2” to “RZ/T, RZ/N”.
		p.1	Added new device RZ/T2N.
		p.6	Added new RZ/T2N board User’s Manual in Section 1.3.1 Evaluation Board User’s Manual.
		p.8	Added information for RZ/T2N in Section 2.1.2.1 Obtaining an RZ MPU Kit. Updated Windows version to Windows 11 in Section 2.1.2.2 PC Requirements.
		p.17	Updated J-Link version to v9.44 in Section 2.1.4 e²studio installation for Linux PC.
		p.23, p.28, p.30	Updated RZ FSP version to v4.2.0 in Section 2.2.1 Installation of FSP using Platform Installer, Section 2.2.2 Installation of FSP using Package Installer Installation of FSP using Package Installer and 2.2.3 Installation of FSP Packs using a Package Zip file.
		p.33	Updated RZ FSP version to v4.2.0, e²studio version to 2026-07, J-Link version to v9.44 in Table 1.
		p.43- p.45	Added new Section 3.4.5 RZ/T2N Evaluation Board.
		p.66	Added new information for RZ/T2N Evaluation Board in Section 4.6 Run the Blinky Project.
		p.74	Added new note for RZ/T2N in Section 5.3.1 Prerequisites.
		p.94	Added new information for RZ/T2N Evaluation Board.
		p.115- p.119	Added new column for RZ/T2N and updated related known issues in Table 20 エラー! 参照元が見つかりません。.
		p.121	Added new known issues No. 53, No. 54 in Table 20. Resolved known issues No. 33, No. 51 in Table 20.

Rev.	Date	Description	
		Page	Summary
		p.152-p.153	Added new column for RZ/T2N and updated related software limitations in Table 21.
		p.155, p.157	Added new software limitations エラー! 参照元が見つかりません。, エラー! 参照元が見つかりません。 in Table 21. Resolved software limitation No. 26 in Table 21.
		p.170	Correct typo from "*2. (IAR EWARM) (RZ/N2H Only)*" to "*2. (IAR EWARM) (RZ/T2H Only)*".
		p.171, p.173	Added new information for board RZ/T2N Evaluation Board in Table 22, Table 23.
		p.178	Corrected the End Address for RZT2H Evaluation Board, RZN2H Evaluation Board in Table 23.
		p.181, p.183, p.187-p.189, p.191-p.192, p.196-p.197	Added device RZ/T2N in Table 24, Table 26, Table 28, Table 30, Table 31, Table 32, Table 33, Table 34, Table 35.
4.30	Nov.13.26	All	Changed the CR52 core number notation from CPU0/CPU1 to Core0/Core1 to align with the notation used when creating a project.
		p.17	Updated J-Link version in Section 2.1.4 e^{2}studio installation for Linux PC.
		p.23, p.28, p.30	Updated RZ FSP version to v4.3.0 in Section 2.2.1 Installation of FSP using Platform Installer, Section 2.2.2 Installation of FSP using Package Installer, and Section 2.2.3 Installation of FSP Packs using a Package Zip file.
		p.33	Updated RZ FSP version to v4.3.0, e^{2}studio version to 2026-10, J-Link version to v9.70 in Table 1.
		p.52	Removed information about multicore debugging in Section 4.3 Create a New Project for Blinky. Moved information about Flash Boot mode from the Appendix to Section 4.3 Create a New Project for Blinky.
		p.61	Removed Section 4.4.1 Build and moved the content to Section 4.4 Build the Blinky Project. Removed Section 4.4.2 Build for Multiprocessing and moved the content to Section 10. Appendix. How to Create and Debug FSP Projects for Multiprocessing for e^{2}studio.
		p.65	Removed Section 4.7 Debug and Run for Multiprocessing and moved the content to Section 10. Appendix. How to Create and Debug FSP Projects for Multiprocessing for e^{2}studio.
		p.71	Updated the note for RZ/T2N in Section 5.3.1 Prerequisites.
		p.72	Removed information about multicore debugging in Section 5.3.2 Create a New Project. Moved information about Flash Boot mode from the Appendix to Section 5.3.2 Create a New Project.

Rev.	Date	Description	
		Page	Summary
		p.86	Removed Section 5.3.3.1 Build and moved the content to Section 5.3.3 Build the Project. Deleted Section 5.3.3.2 Build for Multiprocessing and moved the content to Section 11. Appendix. How to Create and Debug FSP Projects for Multiprocessing for IAR EWARM.
		p.90	Removed Section 5.3.5 Debug for Multiprocessing and moved the content to Section 11. Appendix. How to Create and Debug FSP Projects for Multiprocessing for IAR EWARM.
		p.109	Removed the individual Known Issues entries and added a link to GitHub Issues. Updated Section 7.2 Tool Software Limitations. • Resolved issue No.15 and No. 25. • Added new limitation No. 31.
		p.109	Removed Section 8.1 How to Debug the FSP Project with Flash Boot Mode and moved the content to Section 4.3 Create a New Project for Blinky and Section 5.3.2 Create a New Project.
		p.115	Added new section 8.2 How to Program an ELF File to Flash Memory.
		p.119	Added Sections 10.1 Debug Primary CR52 Core0 or CA55 Core0 (All Devices) through Section 10.4 Standalone Primary CA55 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H) and divided the procedures by use case. Added a note explaining the build order for multiprocessing projects in Section 10.2 Standalone Primary CR52 Core0 (RZ/T2M or RZ/T2ME).
		p.131	Added Sections 11.1 Debug Primary CR52 Core0 or CA55 Core0 (All Devices) through Section 11.4 Standalone Primary CA55 Core0 (RZ/T2H, RZ/T2N, or RZ/N2H) and divided the procedures by use case. Added a note explaining the build order for multiprocessing projects in Section 11.2 Standalone Primary CR52 Core0 (RZ/T2M or RZ/T2ME).
		p.147	Added new chapter 12. Appendix. How to Create a CMake Project using e^{2}studio.
		p.149	Added new chapter 13. Appendix. How to Create a CMake Project using FSP SC.

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

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

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

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

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

Contact information

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

© 2026 Renesas Electronics Corporation. All rights reserved.