RZ/T2, RZ/N2

Getting Started with Flexible Software Package

Introduction

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

Target Device

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

RZ/N series: RZ/N2L

About the video contents

We provide videos on how to install the development tools that will enable you to start developing applications for the RZ/T and RZ/N series of MPUs. Access the following links:

RZ/T RZ/N FSP Quick Start Guide - FSP Installation and Generating Your First Project for e2 studio

RZ/T RZ/N FSP Quick Start Guide - FSP Installation & Generating Your First Project for EWARM & FSP SC

Contents

Contents

1. Introduction 5
1.1 Overview 5
1.2 Introduction to FSP 5
1.2.1 Purpose 5
1.2.2 e² studio IDE 5
1.2.3 FSP Smart Configurator 5
1.2.4 FSP Documentation 5
1.3 Related Documentation Files 6
1.3.1 Renesas Starter Kit+ User's Manual 6
1.3.2 FSP Documentation 6
1.4 Starting Development Introduction 7

2. Set up Renesas Starter kit+ for RZ/T2, RZ/N2 CPU Board 8
2.1 Obtaining an Renesas Starter Kit+ 8
2.2 System Configuration 8
2.3 Supported Emulator 9
2.3.1 SEGGER J-Link 9
2.3.2 IAR I-Jet 9
2.4 RZ/T Series Board Setup 10
2.4.1 RSK+RZT2M 10
2.4.2 RSK+RZT2L 13
2.5 RZ/N Series Board Setup 16
2.5.1 RSK+RZN2L 16

3. e² studio Setup 19
3.1 What is e² studio? 19
3.2 e² studio Prerequisites 19
3.2.1 Windows PC Requirements 19
3.2.2 Installing e² studio, Platform Installer and FSP Package 19
3.2.3 Choosing a Toolchain 19
3.2.4 Licensing 19

4. Tutorial: Your First RZ/T2, RZ/N2 MPU Project – Blinky 20
4.1 Tutorial Blinky 20
4.2 What Does Blinky Do? 20
4.3 Create a New Project for Blinky 21
4.3.1 Details about the Blinky Configuration 26
4.3.2 Configuring the Blinky Clocks 26
4.3.3 Configuring the Blinky Pins 26

4.3.4 Configuring the Parameters for Blinky Components 26
4.3.5 Where is main()? 26
4.3.6 Blinky Example Code 26
4.4 Build the Blinky Project 27
4.4.1 Build 27
4.5 Debug the Blinky Project 28
4.5.1 Debug Prerequisites 28
4.5.2 Debug Steps 28
4.5.3 Details about the Debug Process 31
4.6 Run the Blinky Project 32
4.7 Debug and Run for Multiprocessing 32
4.8 Import the Project 34

5. FSP Smart Configurator User Guide 37
5.1 What is FSP Smart Configurator? 37
5.2 Tutorial Blinky 37
5.3 Using Smart Configurator with IAR EWARM 37
5.3.1 Prerequisites 37
5.3.2 Create a New Project 38
5.3.3 Build the Project 43
5.3.4 Download & Debug the Project 48
5.3.5 Debug for Multiprocessing 52
5.4 Re-configuring Project with FSP SC 52
5.4.1 Launch FSP Smart Configurator from IAR EWARM 52
5.4.2 Launch from the Command Prompt. 53
5.5 Note when debugging in different workspaces 53

6. FSP Configuration Users Guide 54
6.1 What is a Project? 54
6.2 Create a Project 56
6.2.1 Creating a New Project 56
6.2.2 Selecting a Board and Toolchain 58
6.2.3 Selecting a Project Template 59
6.3 Configuring a Project 62
6.3.1 Summary Tab 62
6.3.2 Configuring the BSP 63
6.3.3 Configuring Clocks 64
6.3.4 Configuring Pins 65
6.4 Configuring Interrupts from the Stacks Tab 68
6.4.1 Creating Interrupts from the Interrupts Tab 69
6.4.2 Viewing Event Links 70
6.5 Adding and Configuring HAL Drivers 71

6.6 Reviewing and Adding Components 72

Appendix. Known Issues 73

Appendix. Tool Software Limitations 84

Appendix. How to Debug FSP Project with Flash Boot Mode 90

Appendix. How to Erase Flash Memory 91

Appendix. How to Change Boot Mode of FSP Project 96

Appendix. How to Debug FSP multiprocessing projects with Flash Boot Mode 99

Revision History 107

1. Introduction

1.1 Overview

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

1.2 Introduction to FSP

1.2.1 Purpose

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

1.2.2 e² studio IDE

FSP provides a host of efficiency enhancing tools for developing projects targeting the Renesas RZ/T2, RZ/N2 series of MPU devices. The e² studio IDE provides a familiar development cockpit from which the key steps of project creation, module selection and configuration, code development, code generation, and debugging are all managed.

1.2.3 FSP Smart Configurator

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

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

- IAR Systems Embedded Workbench for Arm (IAR EWARM) with IAR toolchain for Arm

1.2.4 FSP Documentation

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

1.3 Related Documentation Files

The related documentation files are shown in the following.

1.3.1 Renesas Starter Kit+ User's Manual

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

- RZ/T series
  - RZ/T2M Group Renesas Starter Kit+ for RZ/T2M User's Manual
    - Document No. **R20UT4939**
  - RZ/T2L Group Renesas Starter Kit+ for RZ/T2L User's Manual
    - Document No. **R20UT5164**
- RZ/N series
  - RZ/N2L Group Renesas Starter Kit+ for RZ/N2L User's Manual
    - Document No. **R20UT4984**

These documents can be found on Renesas web site by inputting their **Document No.** into search box.

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

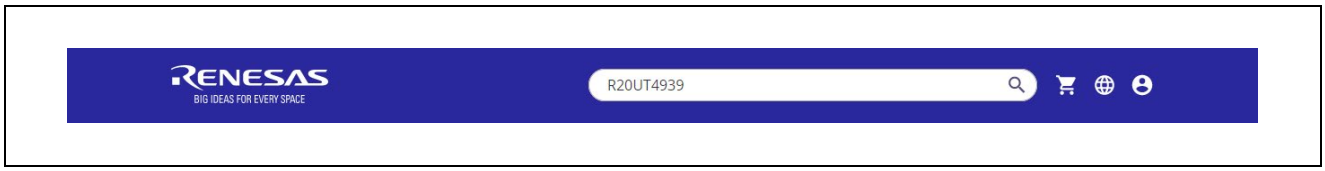


Figure 1: Search Box in Renesas Web Page

1.3.2 FSP Documentation

This Getting Started Guide refers to the following "FSP Documentation".
These documents are available in Renesas Git repository in GitHub.

- RZ/T series
  - RZ/T2 Flexible Software Package Documentation
    - URL: https://github.com/renesas/rzt-fsp/releases
      - File name: fsp_documentation_vx.x.x.zip
- RZ/N series
  - RZ/N2 Flexible Software Package Documentation
    - URL: https://github.com/renesas/rzn-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".

Note for RZ/N2L:

The RZ/N2L FSP documentation can also be viewed in a web browser by accessing the following link, but it is RZ/N2L FSP v1.0.0 content.

- URL: https://renesas.github.io/rzn-fsp/

The latest version of the document should be obtained as a zip file as described above.

1.4 Starting Development Introduction

FSP application project can be created by e^2 studio or FSP SC (for IAR EWARM), and this Getting Started includes tutorial for both tools; the chapters you should read changes.

e^2 studio users should read the following chapters:

- Chapter 2 ”Set up Renesas Starter kit+ for RZ/T2, RZ/N2 CPU Board”
- Chapter 3 “e^2 studio Setup”
- Chapter 4 “Tutorial: Your First RZ/T2, RZ/N2 MPU Project – Blinky”
- Chapter 6 “FSP Configuration Users Guide”

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

- Chapter 2 ”Set up Renesas Starter kit+ for RZ/T2, RZ/N2 CPU Board”
- Chapter 5 “FSP Smart Configurator User Guide”
- Chapter 6 “FSP Configuration Users Guide”

The summary of each chapter is shown below.

- Chapter 2 ”Set up Renesas Starter kit+ for RZ/T2, RZ/N2 CPU Board”
  - Explains how to setup Renesas Starter kit+ for RZT2, RZ/N2 CPU Board to proceed the tutorials in Chapter 4 and 5.
- Chapter 3 “e^2 studio Setup”
  - Explains the setup of e^2 studio for utilizing FSP.
- Chapter 4 “Tutorial: Your First RZ/T2, RZ/N2 MPU Project – Blinky”
  - Explains the tutorial with minimal steps to create, run, and debug a FSP project by using e^2 studio.
- Chapter 5 “FSP Smart Configurator User Guide”
  - Explains the tutorial with minimal steps to create an FSP project as IAR EWARM project by using the FSP Smart Configurator and to run and debug the created IAR EWARM project.
- Chapter 6 “FSP Configuration Users Guide”
  - Explains how to create and configure an FSP project in detail.
  - The explanation is described based on e^2 studio, but most of the explanations are applied to the FSP smart configurator.

2. Set up Renesas Starter kit+ for RZ/T2, RZ/N2 CPU Board

2.1 Obtaining an Renesas Starter Kit+

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

The Renesas Starter Kit+ for RZ/T2 and RZ/N2 CPU Board (RSK+RZT2M, RSK+RZ/T2L, and RSK+RZN2L) are designed to seamlessly integrate with the e^2 studio.

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

2.2 System Configuration

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

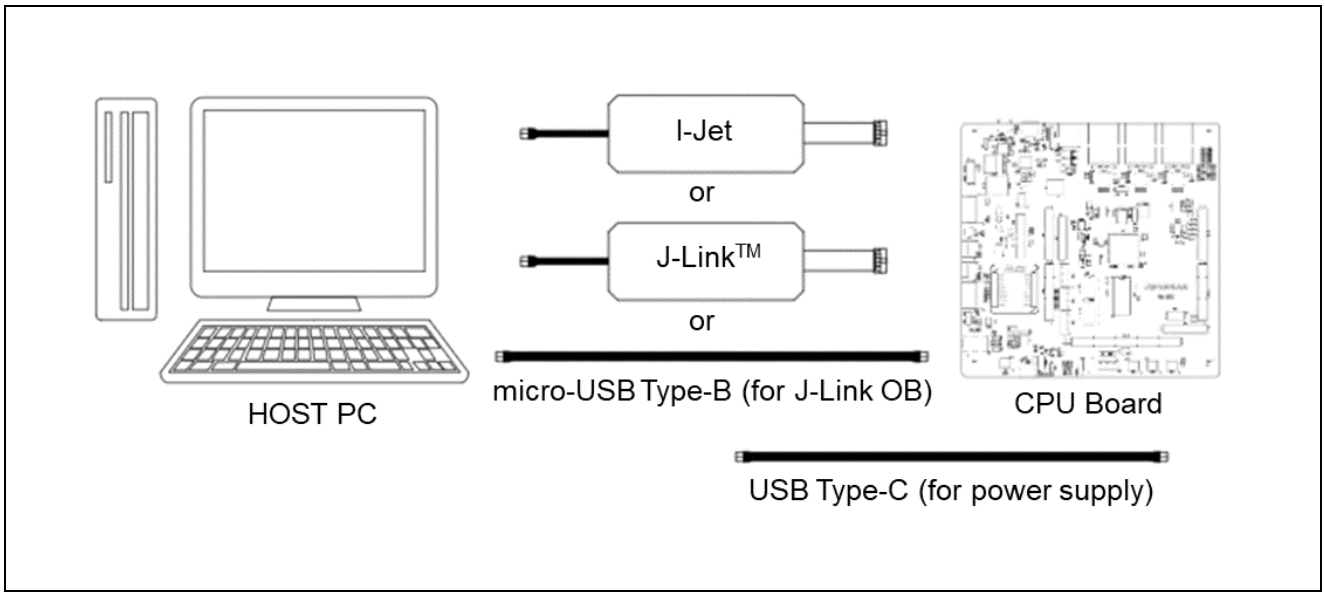


Figure 2: System Configuration Example – with RSK Board

For the details, please refer to the related document “Renesas Starter Kit+ User’s Manual”.

2.3 Supported Emulator

2.3.1 SEGGER J-Link

SEGGER J-Link can be used on Renesas e^2 studio only for debugging on RZ/T2 and RZ/N2 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/T2 and RZ/N2 devices with J-Link BASE V11 and J-Link OB-S124.
For the details on SEGGER J-Link, please see SEGGER website.

Debugging 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/T2 FSP v2.0.0	2024-01.1	V7.94h
RZ/N	RZ/N2L	RZ/N2L FSP v2.0.0	2024-01.1	V7.94h

2.3.2 IAR I-Jet

IAR I-jet can be used on IAR EWARM only for debugging on RZ/T2 and RZ/N2 devices.
For the details on IAR I-jet, please see IAR Systems website.

2.4 RZ/T Series Board Setup

2.4.1 RSK+RZT2M

2.4.1.1 Boot Mode

The operation mode settings for the RSK+RZT2M board are as follows. This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, please refer to the manual of the RSK boards listed in chapter 1.3.1.

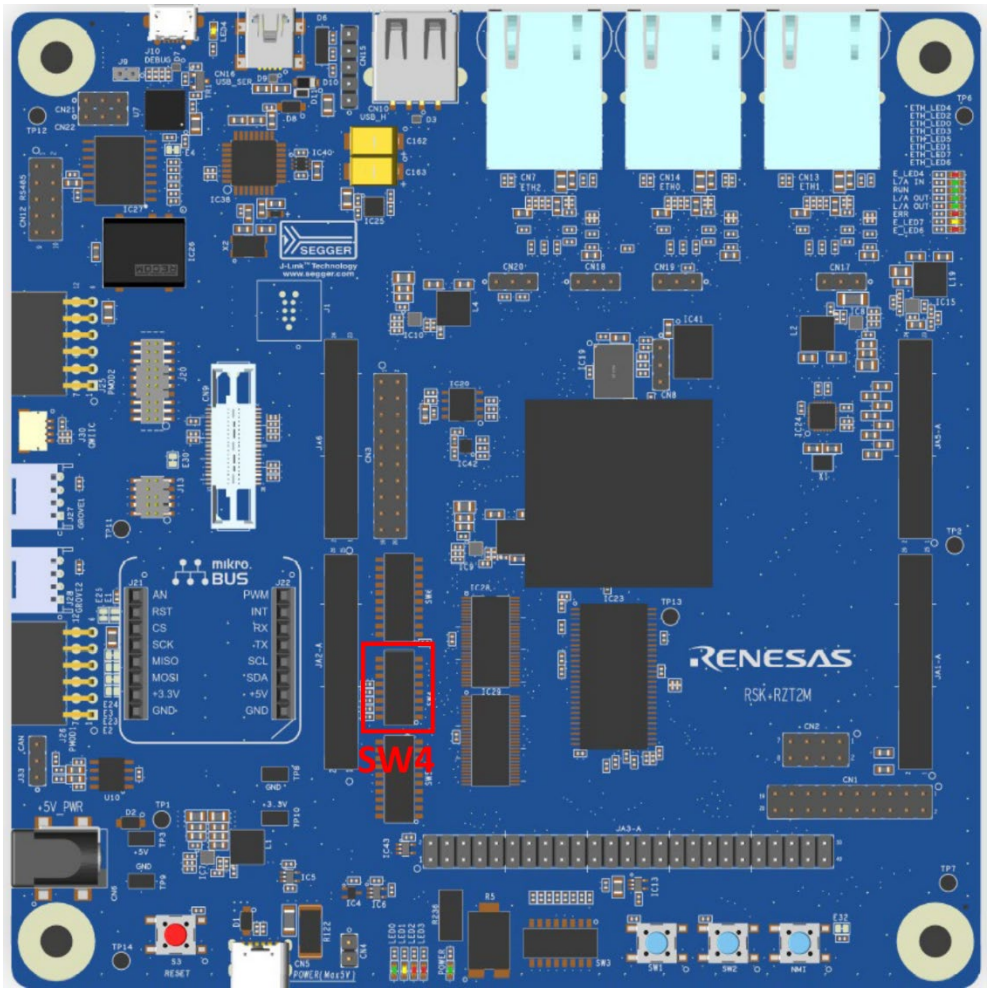


Figure 3: Switch Position of Operation Mode Settings for RSK+RZT2M

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.

2.4.1.2 Debugger Connection

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

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

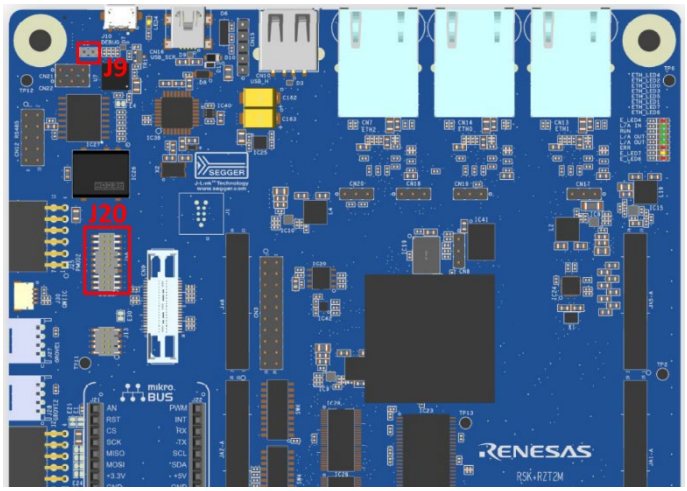


Figure 4: Jumper Position of JTAG connection for RSK+RZT2M

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

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

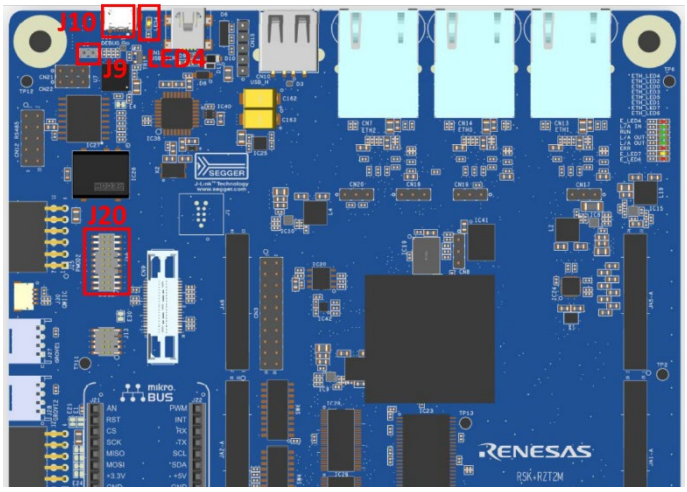


Figure 5: J-Link OB Connection Settings for RSK+RZT2M

2.4.1.3 Power Supply

Power is supplied using a USB cable (Type-C) 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.

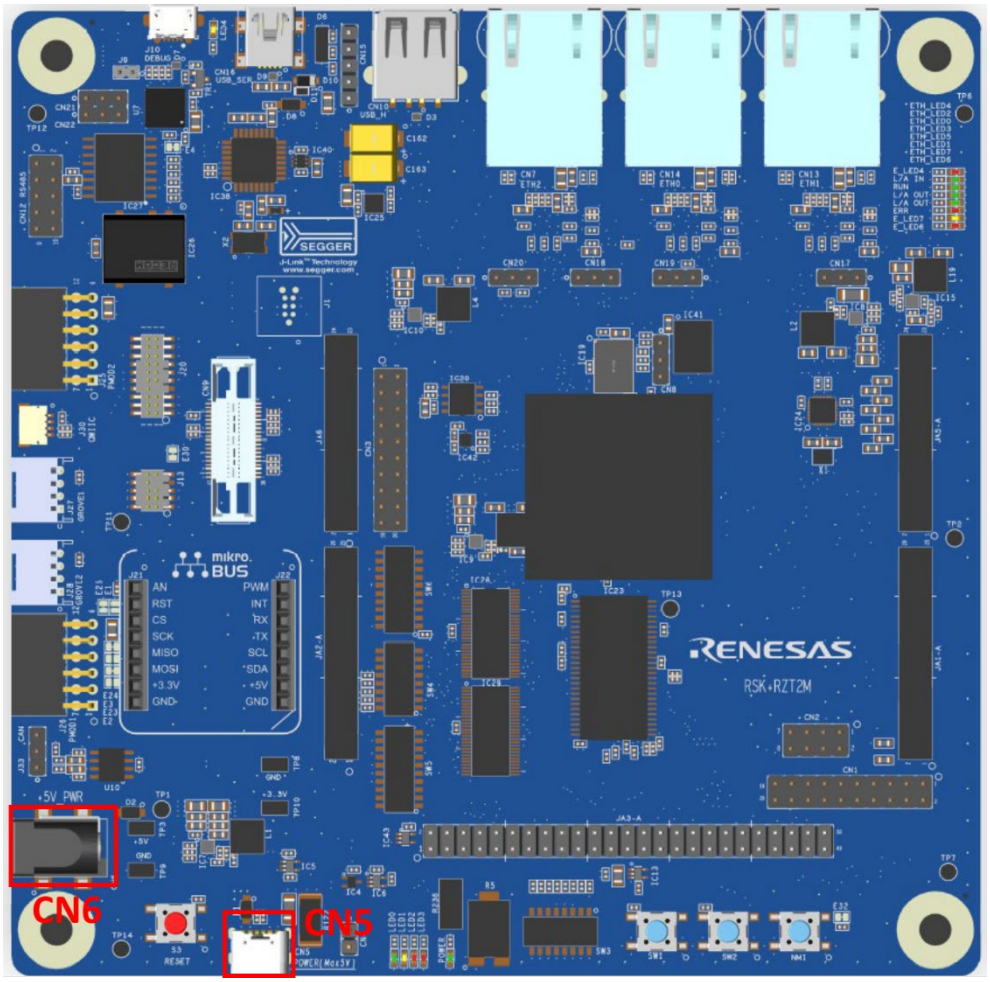


Figure 6: How to Power Supply for RSK+RZT2M

2.4.2 RSK+RZT2L

2.4.2.1 Boot Mode

The operation mode settings for the RSK+RZT2L board are as follows. This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, please refer to the manual of the RSK boards listed in chapter 1.3.1.

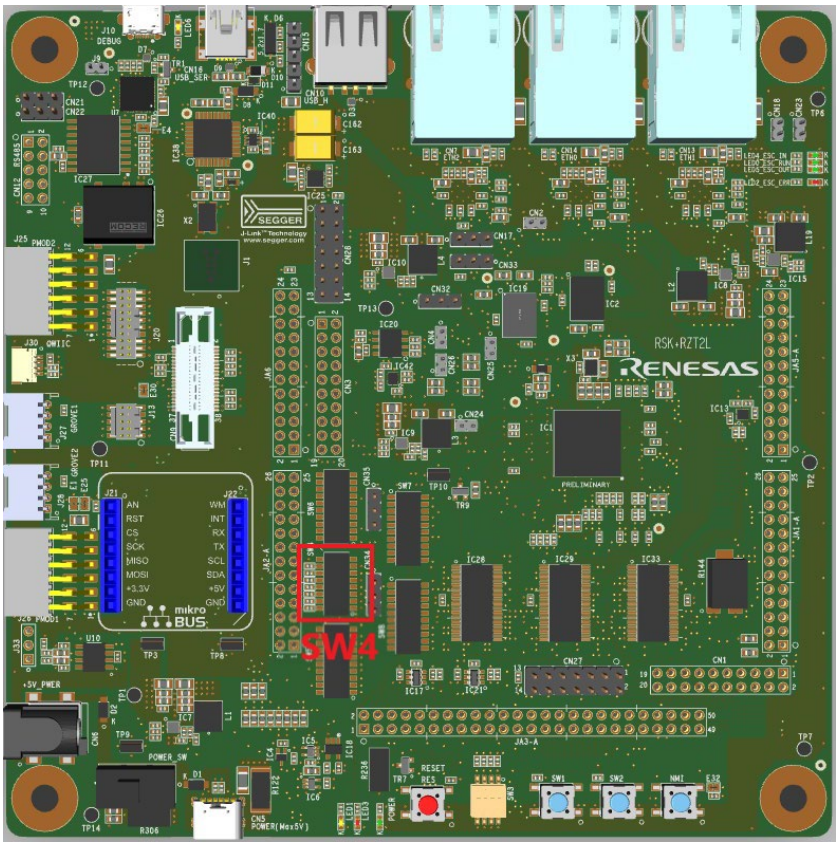


Figure 7: Switch Position of Operation Mode Settings for RSK+RZT2L

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

2.4.2.2 Debugger Connection

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

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

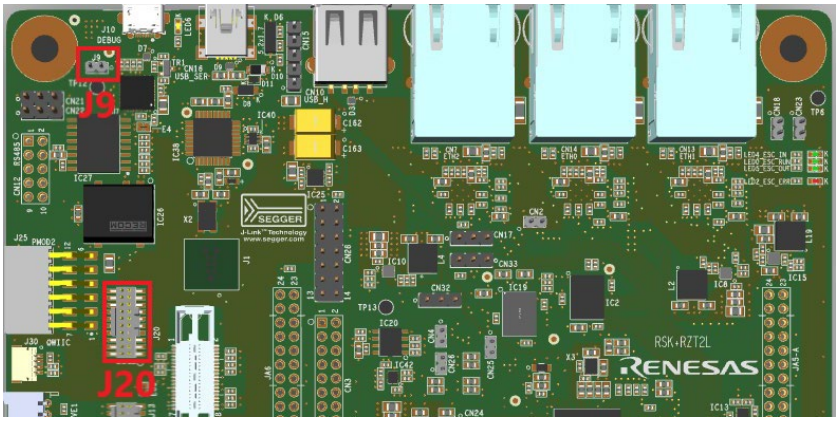


Figure 8: Jumper Position of JTAG connection for RSK+RZT2L

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

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

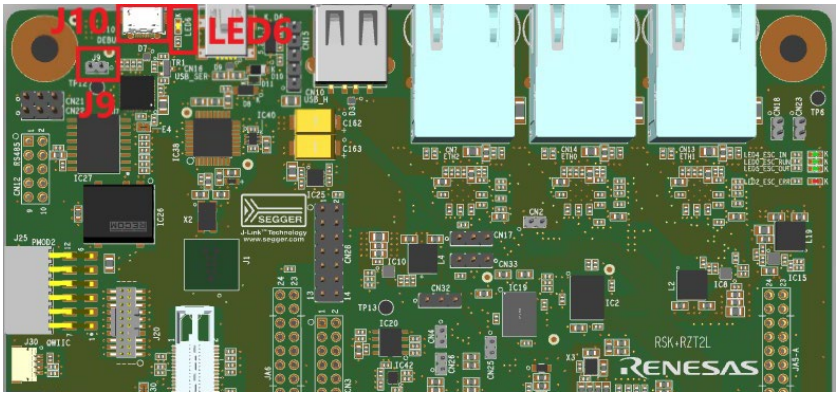


Figure 9: J-Link OB Connection Settings for RSK+RZT2L

2.4.2.3 Power Supply

Power is supplied using a USB cable (Type-C) 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.

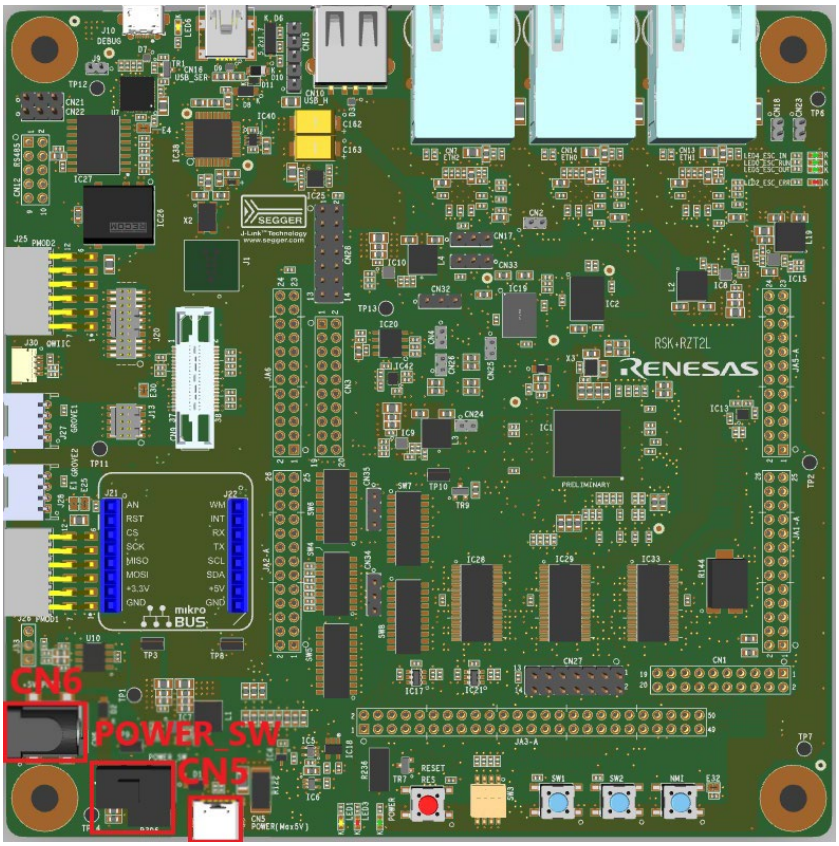


Figure 10: How to Power Supply for RSK+RZT2L

2.5 RZ/N Series Board Setup

2.5.1 RSK+RZN2L

2.5.1.1 Boot Mode

The operation mode settings for the RSK+RZN2L board are as follows. This section shows the settings for running on RAM without external flash memory. For settings to run in other boot modes, please refer to the manual of the RSK boards listed in chapter 1.3.1.

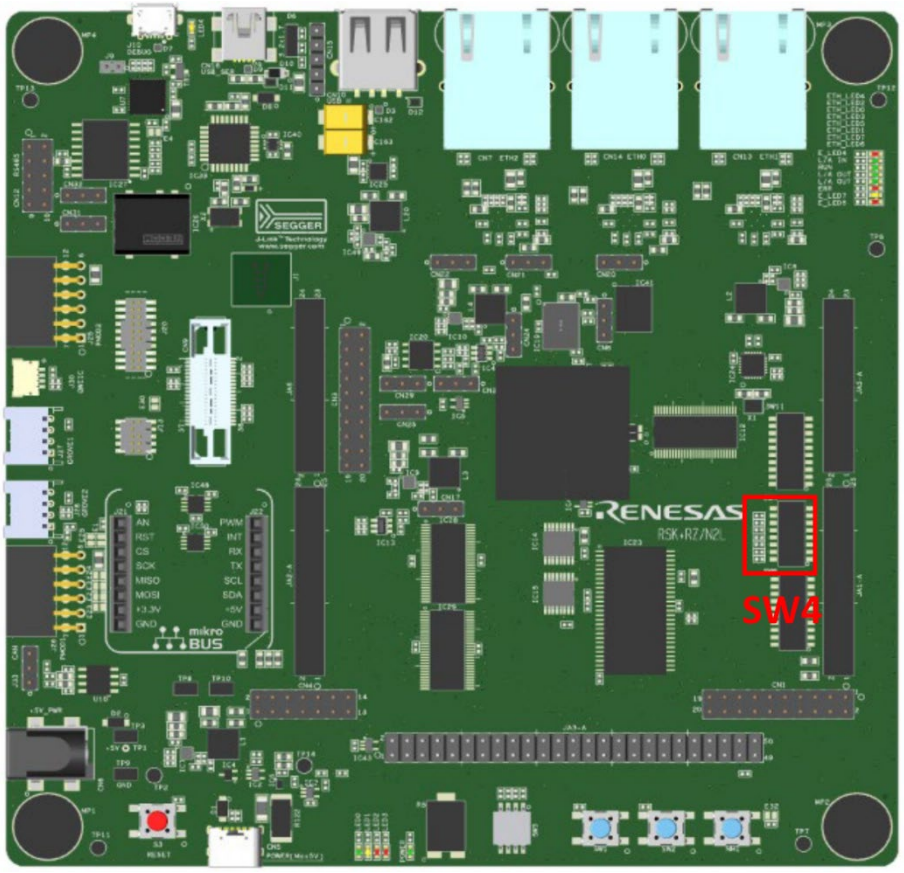


Figure 11: Switch Position of Operation Mode Settings for RSK+RZN2L

Table 4 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.

2.5.1.2 Debugger Connection

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

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

The figure below is when a I-jet is used as Emulator.

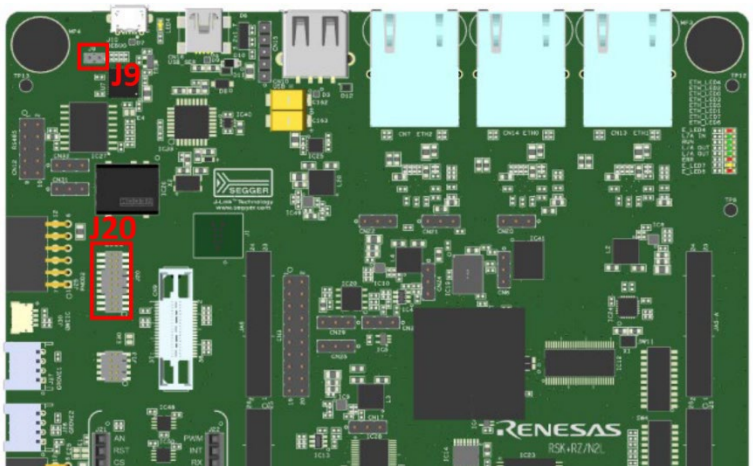


Figure 12: Jumper Position of JTAG Connection for RSK+RZN2L

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

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

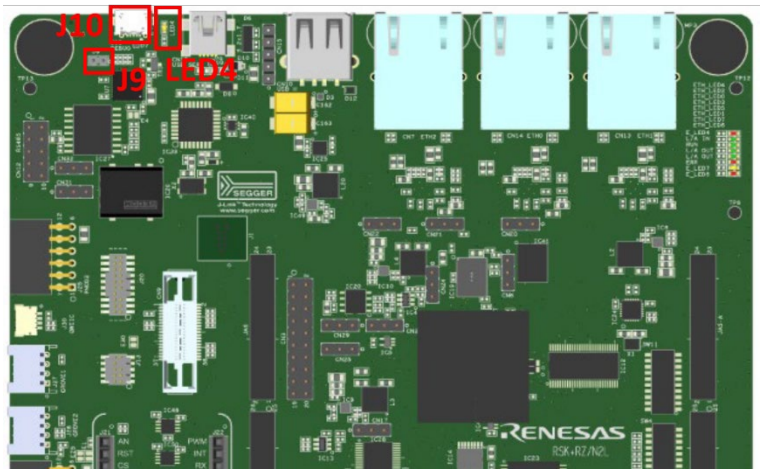


Figure 13: J-Link OB Connection Settings for RSK+RZN2L

2.5.1.3 Power Supply

Power is supplied using a USB cable (Type-C) 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.

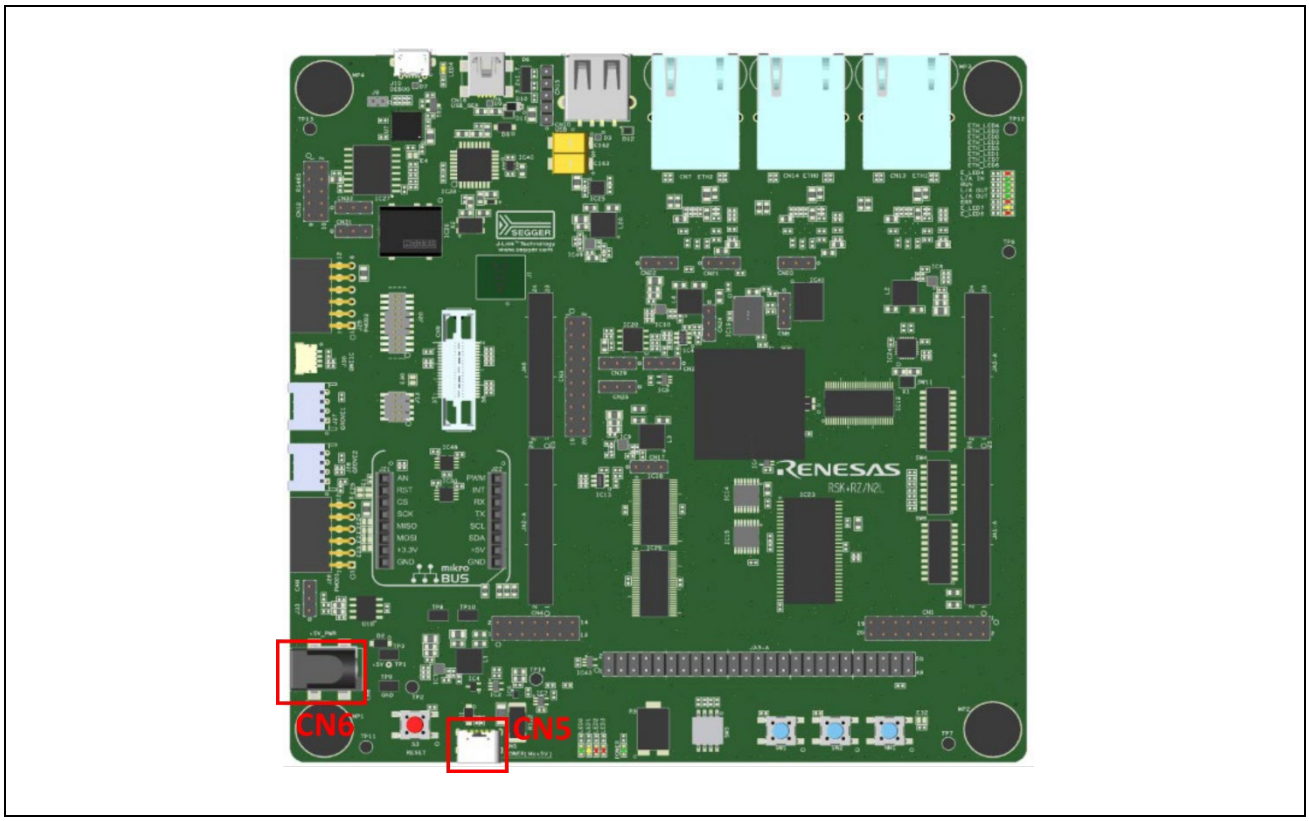


Figure 14: How to Power Supply for RSK+RZN2L

3. e² studio Setup

3.1 What is e² studio?

Renesas e² studio is a development tool encompassing code development, build, and debug. e² studio is based on the open-source Eclipse IDE and the associated C/C++ Development Tooling (CDT).

When developing for RZ/T2, RZ/N2 MPUs, e² studio hosts the Renesas Flexible Software Package (FSP). FSP provides a wide range of time saving tools to simplify the selection, configuration, and management of modules and threads, to easily implement complex applications.

3.2 e² studio Prerequisites

3.2.1 Windows PC Requirements

The following are the Windows PC requirements to use e² studio:

For Windows 64-bit version

- Windows® 11 (64-bit version)
- Windows® 10 (64-bit version)
- Windows® 8.1 (64-bit version)
- Memory capacity: We recommend 8 GB or more. At least 4 GB.
- Capacity of hard disk: At least 2 GB of free space.
- Display: Graphics resolution should be at least 1024 x 768, and the mode should display at least 65,536 colors.
- Interface: USB 2.0
- Microsoft Visual C++ 2010 SP1 runtime library *1
- Microsoft Visual C++ 2015-2019 runtime library *1

*1. This software will be installed at the same time as the e² studio.

3.2.2 Installing e² studio, Platform Installer and FSP Package

Detailed installation instructions for the e² studio and the FSP are available on the Renesas website. Review the release notes for e² studio to ensure that the e² studio version supports the selected FSP version. The starting version of the installer includes all features of the RZ/T2, RZ/N2 MPUs.

3.2.3 Choosing a Toolchain

The GNU ARM Embedded Toolchain (version 12.2.1.arm-12-24) is required.

If the version of the toolchain has not been installed, please download the toolchain from ARM Developer website, and install it.

3.2.4 Licensing

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

4. Tutorial: Your First RZ/T2, RZ/N2 MPU Project – Blinky

4.1 Tutorial Blinky

The goal of this tutorial is to quickly get acquainted with the Flexible Platform by moving through the steps of creating a simple application using e^2 studio and running that application on an RZ/T2, RZ/N2 MPU board. This chapter guides you through creating projects for a single-core processing and a multiprocessing. Here, the multiprocessing refers to a process in which CR52 CPU0 core is activated first and CR52 CPU1 core operates after CR52 CPU0 core sets up for CR52 CPU1 core.

4.2 What Does Blinky Do?

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

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

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

4.3 Create a New Project for Blinky

The creation and configuration of an RZ/T and RZ/N C/C++ FSP Project is the first step in the creation of an application. The base RZ/T2 pack and RZ/N2 packs include a pre-written Blinky example application. In the case of multiprocessing, two projects with different settings must be created. The CR52 CPU0 core project that starts first is called the primary project and the CR52 CPU1 core project is called the secondary project.

Note for multiprocessing projects:

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

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

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

1. In e² studio, click **File** > **New** > **C/C++ Project**.

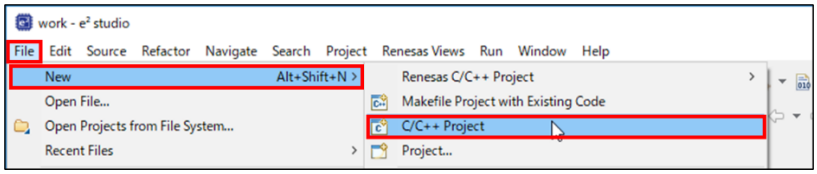


Figure 15: New C/C++ Project

2. Select either one depending on your RZ/T2, RZ/N2 MPU and RSK board.
   - **Renesas RZ > Renesas RZ/T C/C++ FSP Project**
   - **Renesas RZ > Renesas RZ/N C/C++ FSP Project**
3. Click **Next**.

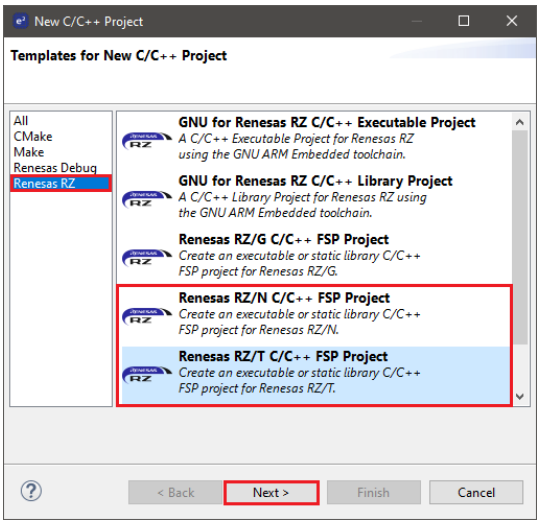


Figure 16: Renesas RZ C/C++ FSP Project

(Continued on next page)

4. Assign a name to this new project. An example of naming is shown below.
   - Single-core processing
     - Blinky
   - Multiprocessing
     - The primary project: Blinky_cpu0_primary
     - The secondary project: Blinky_cpu1_secondary
5. Click **Next**. The Project Configuration window shows your selection.

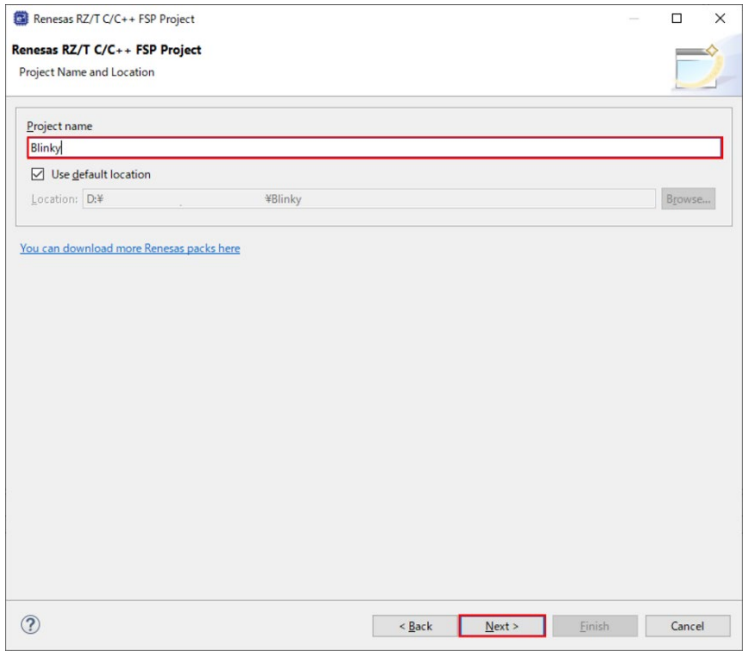


Figure 17 : e² studio Project Configuration Window (Part 1)

(Continued on next page)

6. Select the board support package by selecting the name of your board from the drop-down list.
   In this tutorial, please select either one depending on your device and RSK board.
   - RZ/T series
     - **RSK+RZT2M (RAM execution without flash memory)**
     - **RSK+RZT2L (RAM execution without flash memory)**
   - RZ/N series
     - **RSK+RZN2L (RAM execution without flash memory)**
7. (RZ/T2M ONLY) Select the Core from the drop-down list.
   - Single-core processing
     - CR52_0
   - Multiprocessing
     - The primary project: CR52_0
     - The secondary project: CR52_1
8. Select **GNU ARM Embedded** in Toolchains and version is **12.2.1.arm-12-24** as its toolchain version, and click **Next**.
   - If there is NOT the 12.2.1.arm-12-24 version of GNU ARM Embedded Toolchain, please download the version of the toolchain from ARM Developer website and install it.

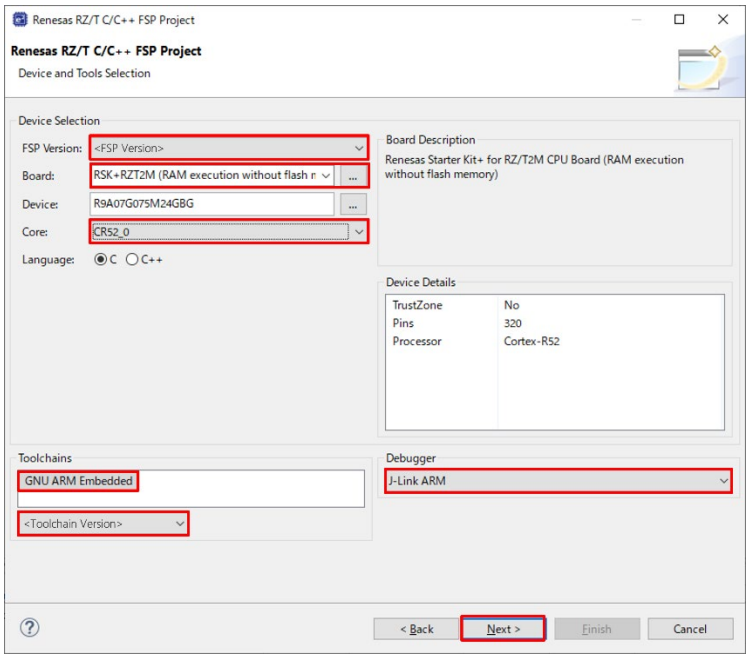


Figure 18 : e² studio Project Configuration Window (Part 2)

9. (The secondary project of multiprocessing ONLY) Select a bundle file of the primary project. Built CPU0 project names in the same workspace appear as an option in the drop-down list.

Note:

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

(Continued on next page)

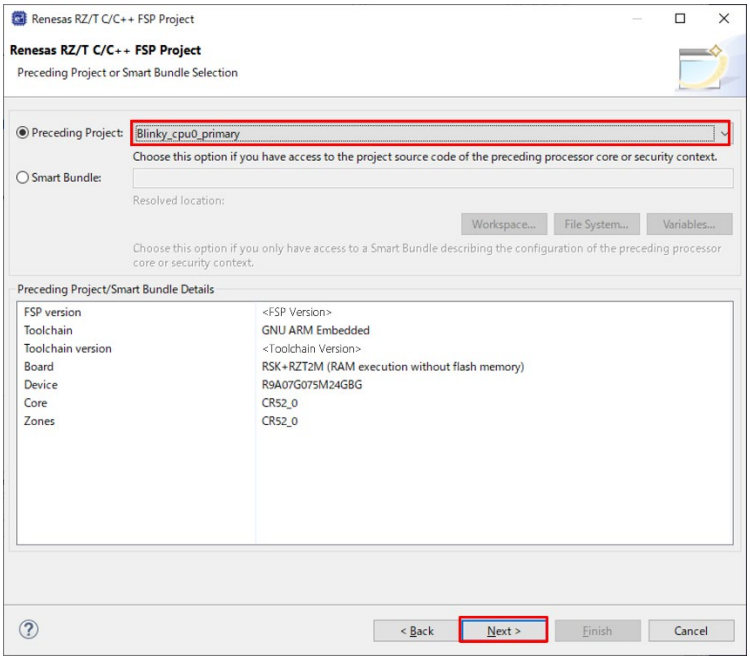


Figure 19 e² studio Project Configuration Window (Part 3)

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

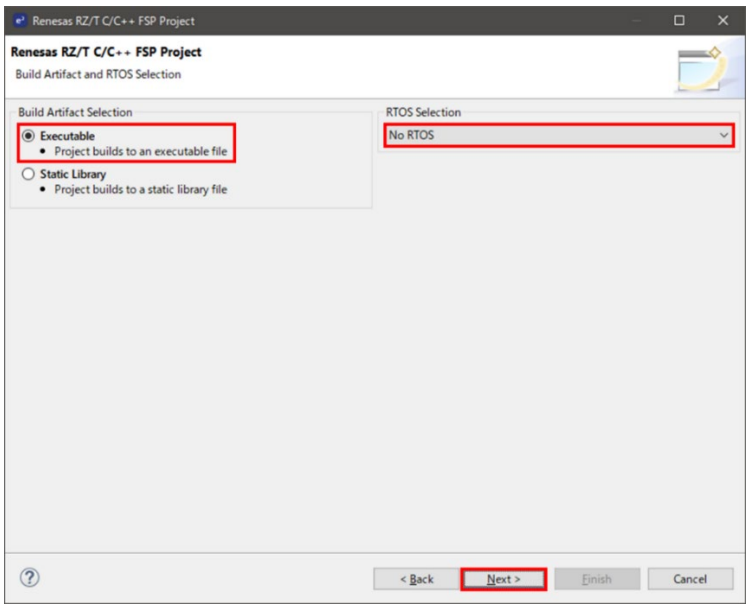


Figure 20 : e² studio Project Configuration Window (Part 4)

(Continued on next page)

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

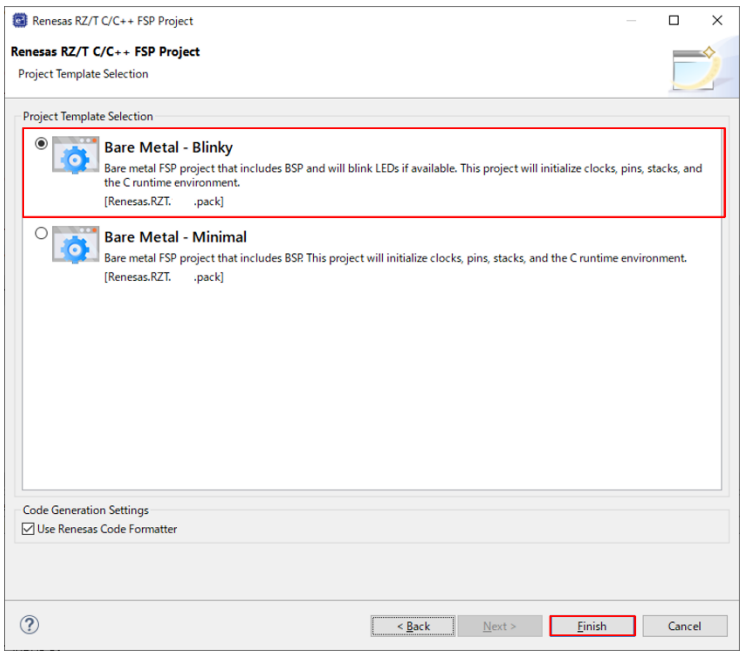


Figure 21 : e² studio Project Configuration Window (Part 5)

Once the project has been created, the name of the project will show up in the **Project Explorer** window of e² studio. Now click the **Generate Project Content** button in the top right corner of the **Project Configuration** window to generate your board specific files.

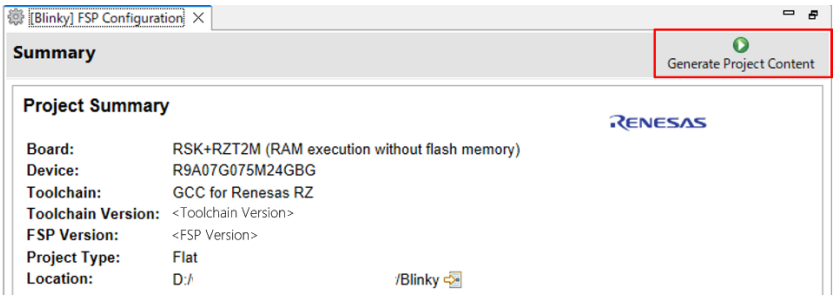


Figure 22 : e² studio Project Configuration Tab

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

4.3.1 Details about the Blinky Configuration

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

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

4.3.2 Configuring the Blinky Clocks

By selecting the Blinky template, the clocks are configured by e^2 studio for the Blinky application.

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

4.3.3 Configuring the Blinky Pins

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

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

4.3.4 Configuring the Parameters for Blinky Components

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

- r_ioport

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

4.3.5 Where is main()?

The main function is located in:

- **< RZT2 FSP project >/rzt_gen/main.c.**
- **< RZN2 FSP project >/rzn_gen/main.c.**

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

4.3.6 Blinky Example Code

The blinky application is stored in the hal_entry.c file. This file is generated by e^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 **bsp_leds_t** structure.
2. Initialize output level for LED pin to LOW using **R_BSP_PinClear((bsp_io_region_t) leds.p_leds[i][1], (bsp_io_port_pin_t) leds.p_leds[i][0])**.
3. Use **R_BSP_PinToggle ((bsp_io_region_t) leds.p_leds[i][1], (bsp_io_port_pin_t) leds.p_leds[i][0])** to set the output level to the LED pin.
4. **R_BSP_SoftwareDelay(delay, bsp_delay_units)** waits for a certain period of time. Then run #3 again.

4.4 Build the Blinky Project

Highlight the new project in the Project Explorer window by clicking on it and build it.
When multiprocessing, build both the primary and secondary projects.

4.4.1 Build

There are three ways to build a project:

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

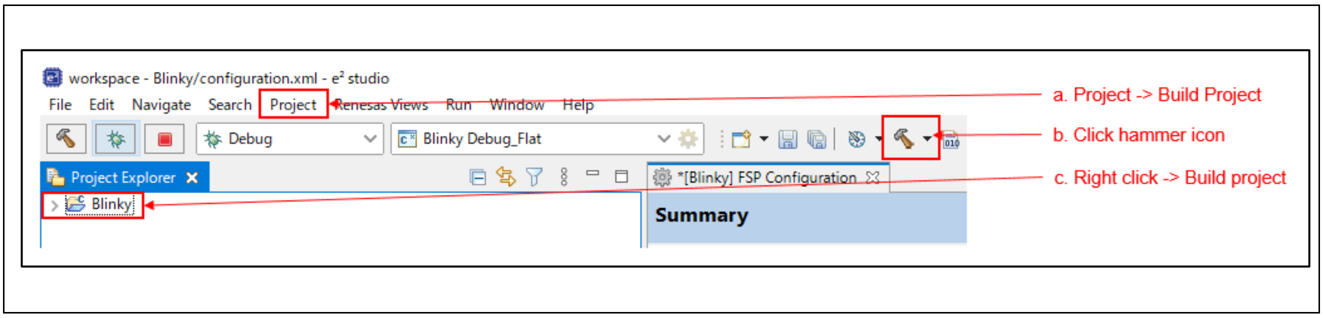


Figure 23 : e² studio Project Explorer Window

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

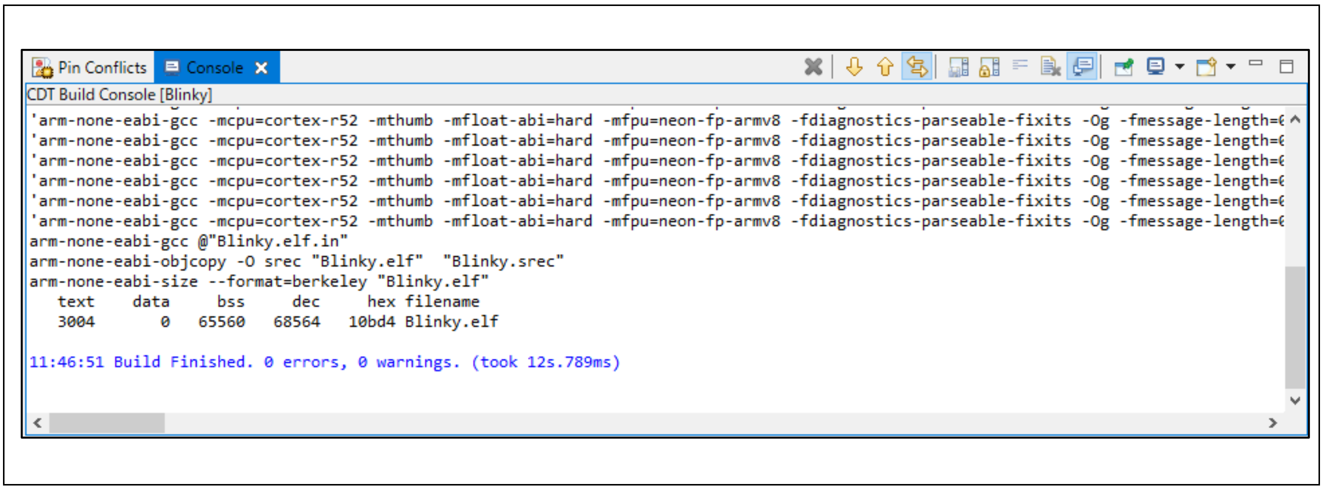


Figure 24 : e² studio Project Build Console

4.5 Debug the Blinky Project

4.5.1 Debug Prerequisites

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

- The board to be connected to e² studio.
- The debugger to be configured to talk to the board.
- The application to be programmed to the microprocessor.

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

RSK board has a JTAG header and requires an external JTAG debugger to the header.

4.5.2 Debug Steps

When multiprocessing, please refer to Section 4.7 Debug and Run for Multiprocessing.

To debug the Blinky application, follow these steps:

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

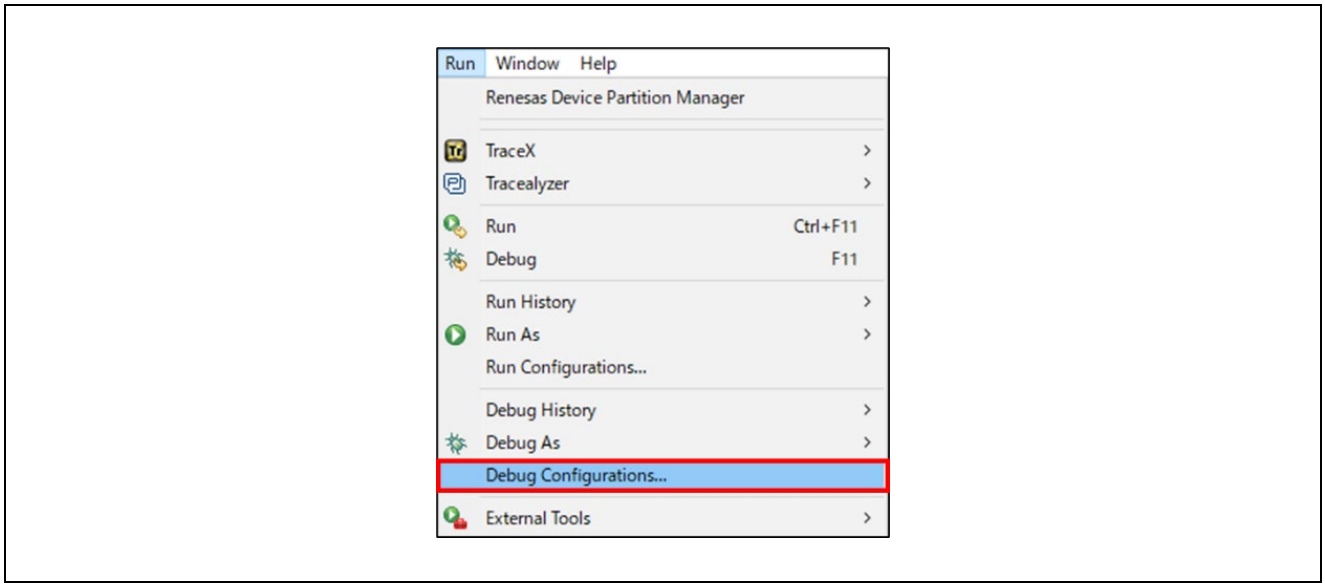


Figure 25 : e² studio Debug Icon

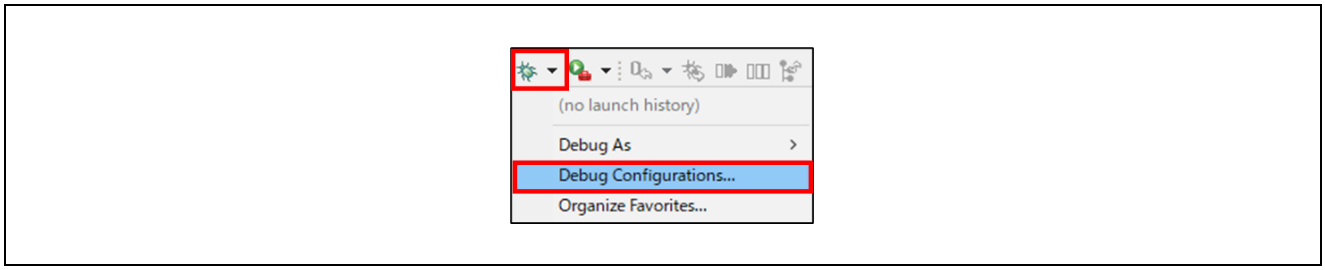


Figure 26 : e² studio Debugger Configurations Selection Option

(Continued on next page)

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.

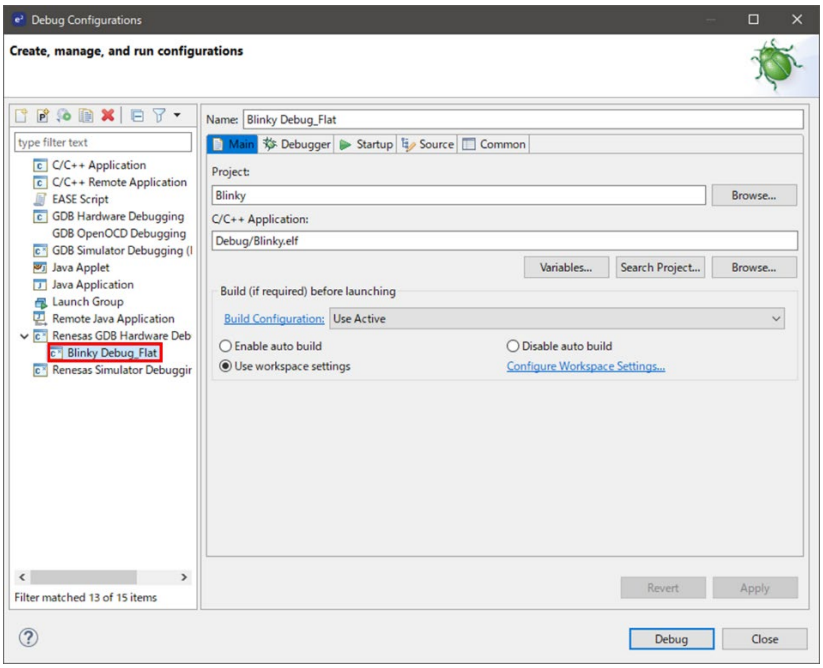


Figure 27 : e² studio Debugger Configurations Window with Blinky Project

(Continued on next page)

3. If you use RAM execution without flash memory boot mode, it needs following configuration.
   - **Debugger** > **Connection Settings** > **Connection**
     - Set **No** to **Reset after download (**Available in e^2 studio 2023-07 and later versions**)**
       to avoid resetting MPU after program download
     - (CPU0(CR52_0) ONLY) Set **Yes** to **Set CPSR(5bit) after download (**Available in e^2 studio 2023-04 and later versions**)**
       to set the CPSR register value of CR52 general register before running the application.

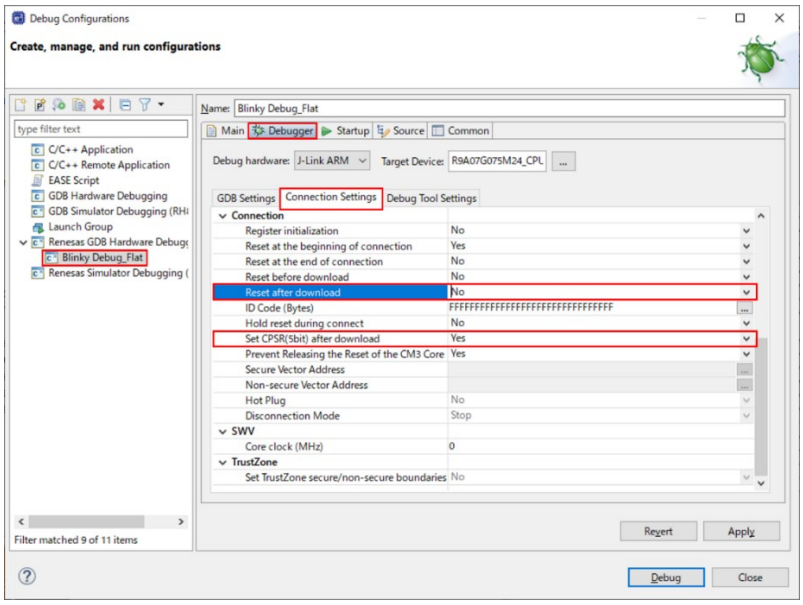


Figure 28 : e^2 studio Debugger Configurations Window with Blinky Project

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

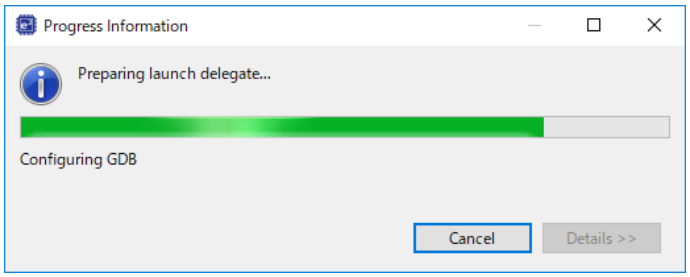


Figure 29: Start Debugging

4.5.3 Details about the Debug Process

In debug mode, e^2 studio executes the following tasks:

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

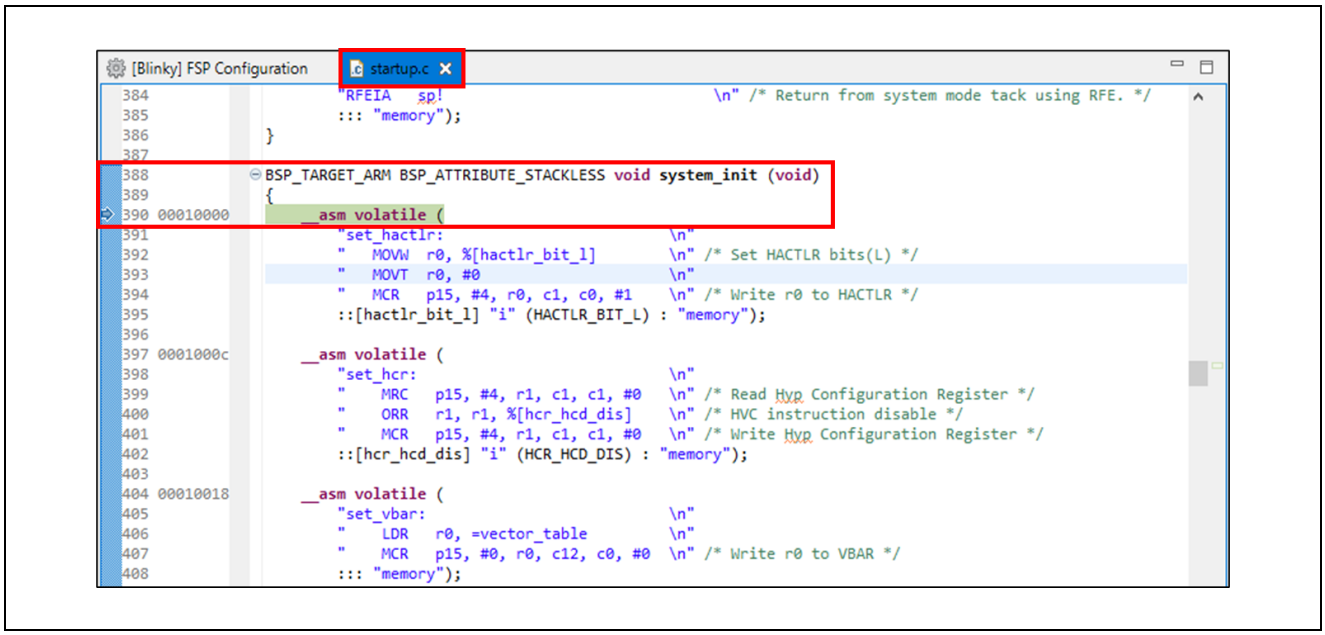


Figure 30 : e^2 studio Debugger Memory Window

4.6 Run the Blinky Project

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

Figure 31 : e² studio Debugger Play Icon

The following LEDs on the board should now be blinking.

- RZ/T series
  - RSK+RZ/T2M: LED0-1 (CPU0), LED2-3 (CPU1)
  - RSK+RZ/T2L: LED0-6 (including LEDx_ESC_xxx)
- RZ/N series
  - RSK+RZ/N2L: LED0-3

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

Figure 32 : e² studio Debugger Pause Icon

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

Figure 33 : e² studio Debugger Stop Icon

4.7 Debug and Run for Multiprocessing

To debug the Blinky application, follow these steps:

1. Connect the debugger with the primary project using the procedure in 4.5.2 Debug Steps.
2. The primary project in operation, connect the debugger with the secondary project using the procedure in 4.5.2 Debug Steps.
3. When the following dialog box is shown, please click **No** to start debugging.

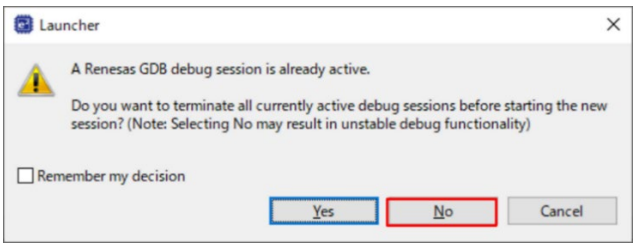


Figure 34 Warning Window of Starting Debug Session

4. When Figure 35 is shown, please click **Yes** to proceed the launch.

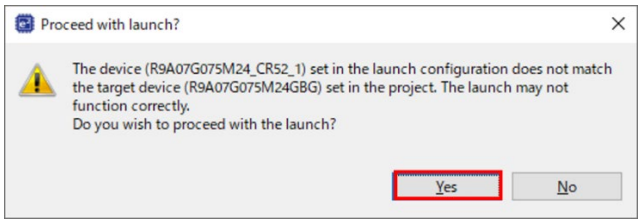


Figure 35 Warning Window of Device Name

5. Run the primary project with procedure 4.6 Run the Blinky Project. If the LED0 and LED1 are blinking, proceed to the next step.
6. Run the secondary project with procedure 4.6 Run the Blinky Project.
7. When exiting Debug mode and disconnect from the debugger, terminate both projects, the primary and the secondary.

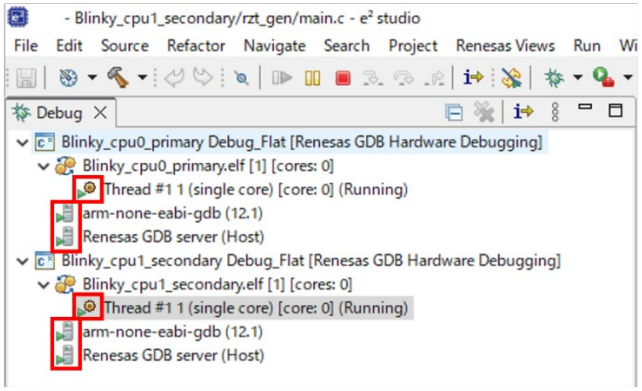


Figure 36 During Program Execution

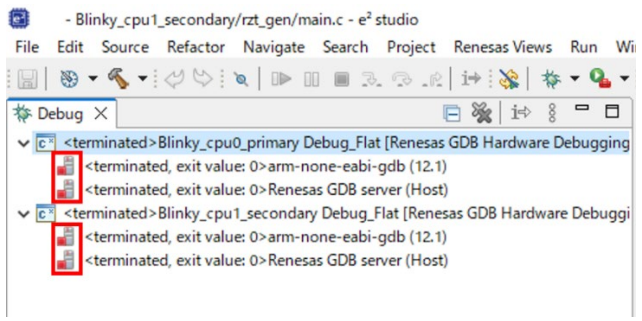


Figure 37 Terminated Program

4.8 Import the Project

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

Note:

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

To import the projects, follow these steps:

1. Click **File** > **Import**.

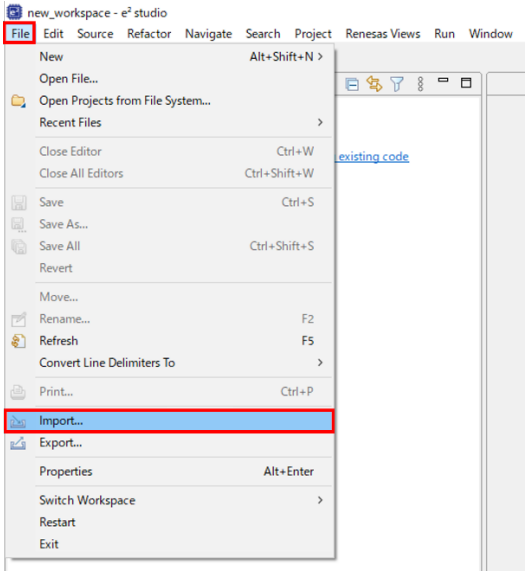


Figure 38 e² studio Import

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

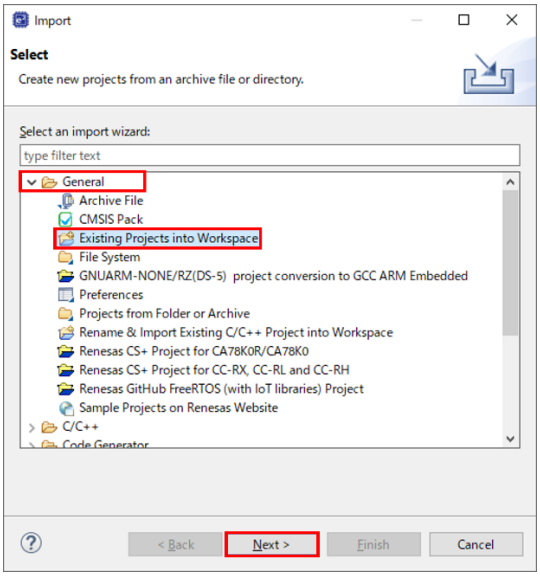


Figure 39 e² studio Select Import Type

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

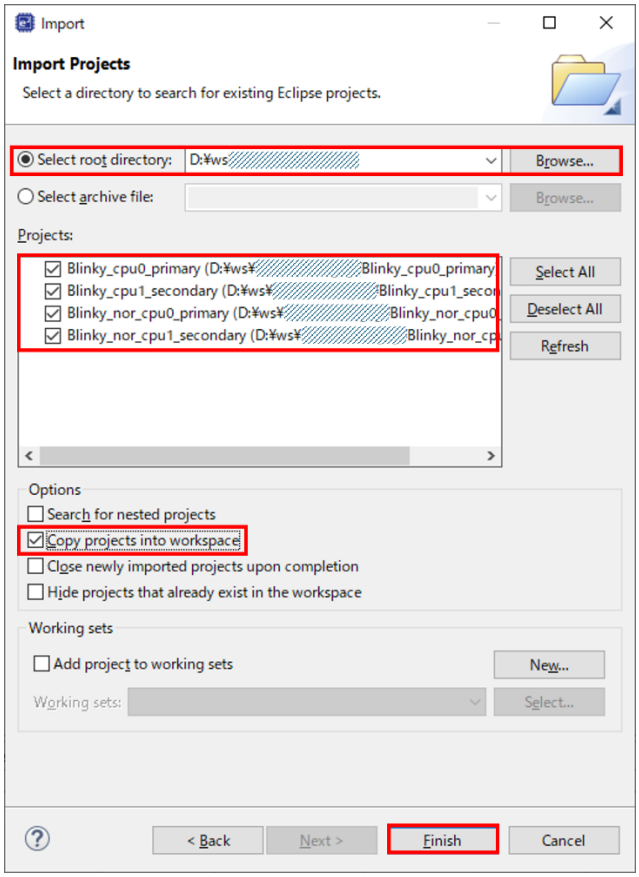


Figure 40 e² studio Select Root Directory to Import Project

(Continued on next page)

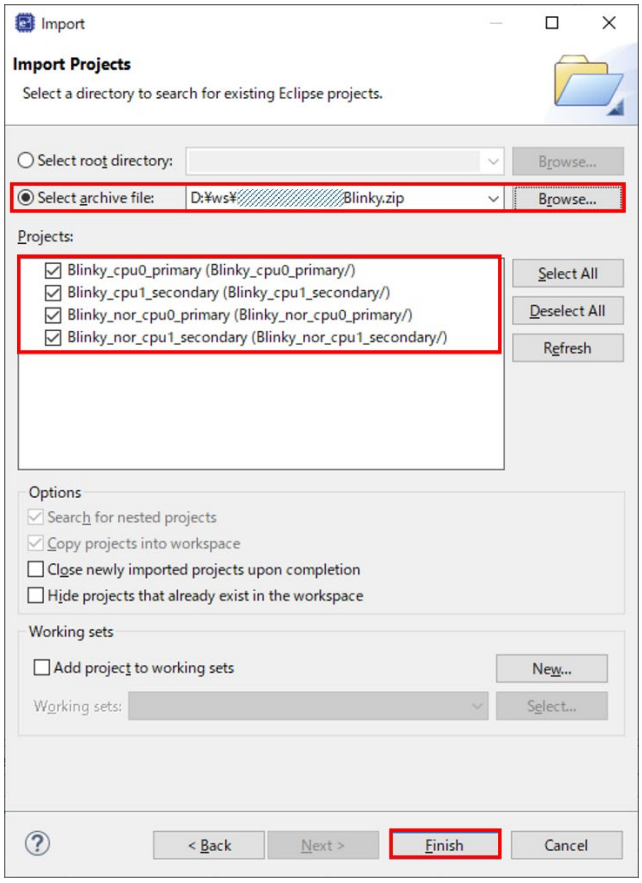


Figure 41 e² studio Select Archive File to Import Project

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

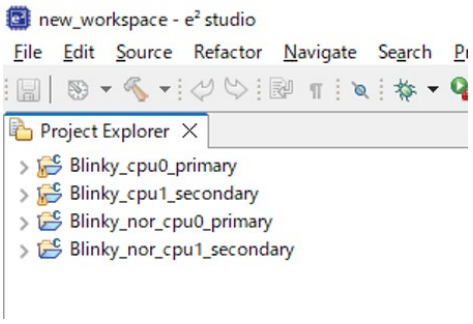


Figure 42 e² studio new workspace

Note:

The imported project must be clicked the **Generate Project Content** button and built before debugging.

5. FSP Smart Configurator User Guide

5.1 What is FSP Smart Configurator?

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

For creating RZ/T2 and RZ/N2 project, 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 e2 studio. Please refer to 5.2 Configuring a Project section for more details.

5.2 Tutorial Blinky

The goal of this tutorial is to quickly get acquainted with the Flexible Platform by moving through the steps of creating a simple application using FSP SC and 3rd-party IDE and running that application on an RZ/T2, RZ/N2 MPU board. This chapter guides you through creating projects for a single-core processing and a multiprocessing. Here, the multiprocessing refers to a process in which CR52 CPU0 core is activated first and CR52 CPU1 core operates after CR52 CPU0 core sets up for CR52 CPU1 core.

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

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

- The toolchain is setup correctly and builds a working executable image for your chip.
- The debugger has 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 Smart Configurator with IAR EWARM

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

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

5.3.1 Prerequisites

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

Note for RZ/T2L:

If you use the IAR EWARM 9.32.1 to debug RZ/T2L FSP project, please apply the following patch file.

- RZ/T2L: EWARM_Patch_for_RZT2L_rev1.0.zip
  - This patch file is available in http://www.renesas.com/rzt2l.

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. In the case of multiprocessing, two projects with different settings must be created. The CR52 CPU0 core project that starts first is called the primary project and the CR52 CPU1 core project is called the secondary project.

Note for multiprocessing projects:

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

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

1. Start the FSP Smart Configurator.
   - FSP Smart Configurator is installed in the following path as default.
     - For RZ/T series, it is installed in C:\Renesas\rzt\sc_vYYYY-MM_fsp_vX.X.X\eclipse\rasc.exe
     - For RZ/N series, it is installed in C:\Renesas\rzn\sc_vYYYY-MM_fsp_vX.X.X\eclipse\rasc.exe

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

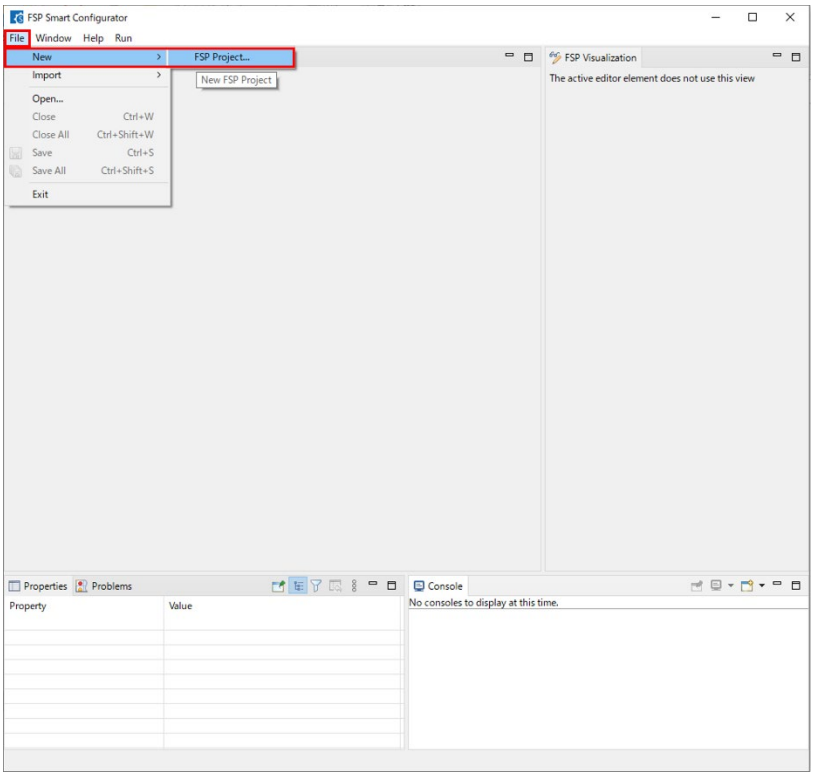


Figure 43 : FSP SC New Project

(Continued on next page)

3. Enter a project folder and project name. An example of naming is shown below.
   - Single-core processing
     - Blinky
   - Multiprocessing
     - The primary project: Blinky_cpu0_primary
     - The secondary project: Blinky_cpu1_secondary

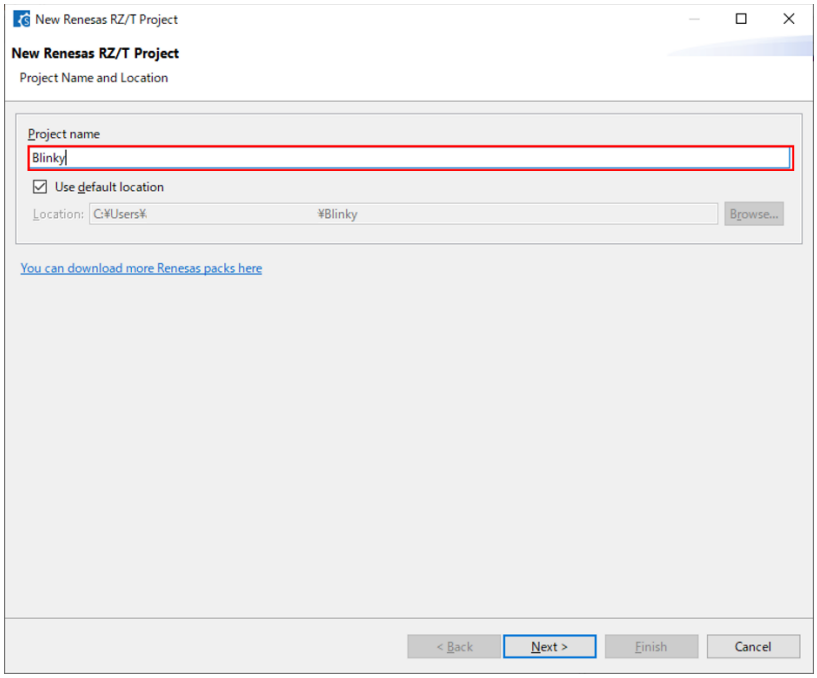


Figure 44 : FSP SC Project Settings

4. Select the **FSP** version.
5. Select the **Board** for your application.
   - You can select an existing RZ/T2, RZ/N2 MPU Evaluation Kit or select Custom User Board for any of the RZ/T2, RZ/N2 MPU devices with your own BSP definition.
   - Here, select either of following boards to create a FSP project for RSK board.
     - RZ/T series
       - **RSK+RZT2M (RAM execution without flash memory)**
       - **RSK+RZT2L (RAM execution without flash memory)**
     - RZ/N series
       - **RSK+RZN2L (RAM execution without flash memory)**
6. (RZ/T2M ONLY) Select the Core from the drop-down list.
   - Single-core processing
     - CR52_0
   - Multiprocessing
     - The primary project: CR52_0
     - The secondary project: CR52_1

(Continued on next page)

7. Select **IDE Project Type**.
   - Here, select **IAR EWARM**.
     - As the Toolchain, IAR Toolchain for ARM is preselected.
8. Click **Next**.

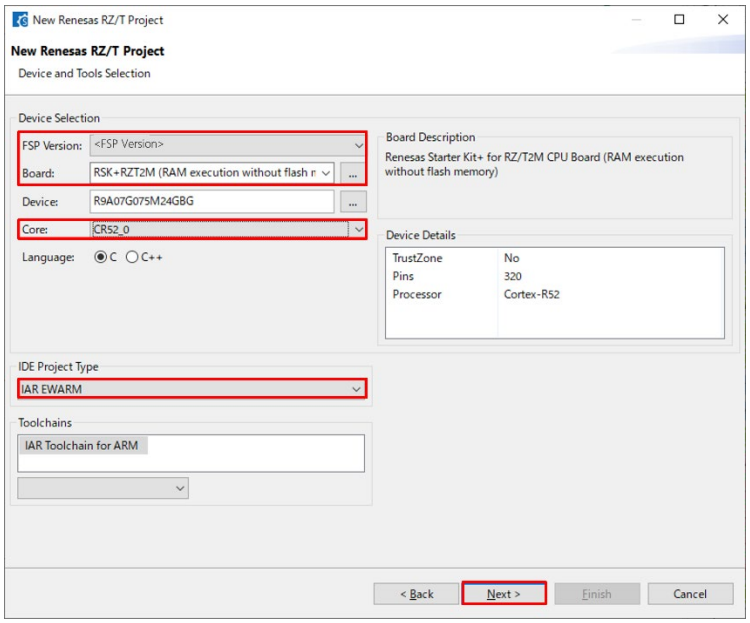


Figure 45 : Target Device and IDE Selections

9. (The secondary project of multiprocessing ONLY) Select a bundle file of the primary project. Built CPU0 project names in the same workspace appear as an option in the drop-down list.

Note:

Warnings occur if the FSP version used is different between the primary project and the secondary project. Use the FSP same version.

(Continued on next page)

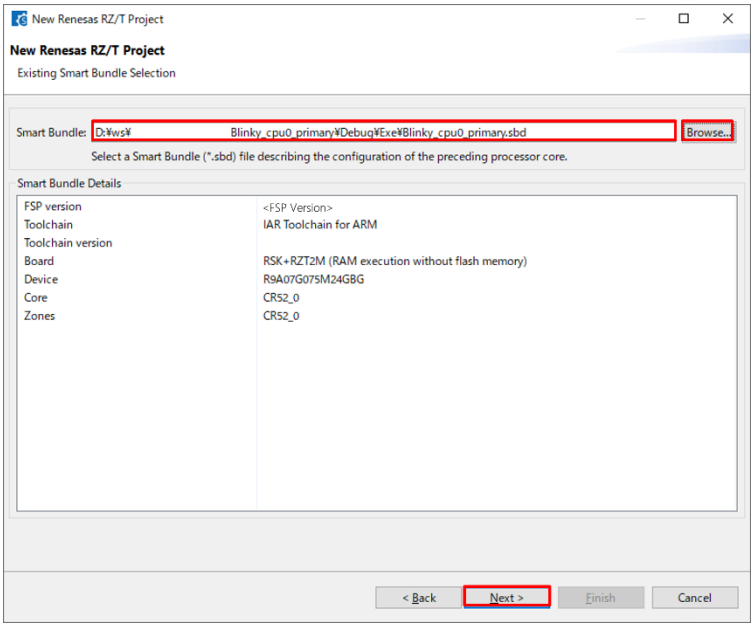


Figure 46 e² studio Project Configuration Window (Part 3)

10. Select **RTOS**.
    - Here, select **No RTOS** for proceeding the following tutorial.
11. Click **Next**.

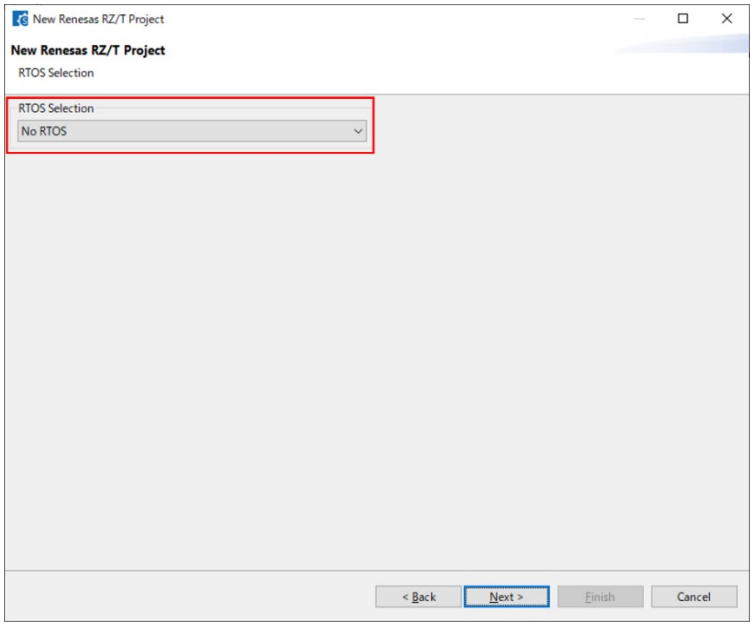


Figure 47 : RTOS Selection

(Continued on next page)

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

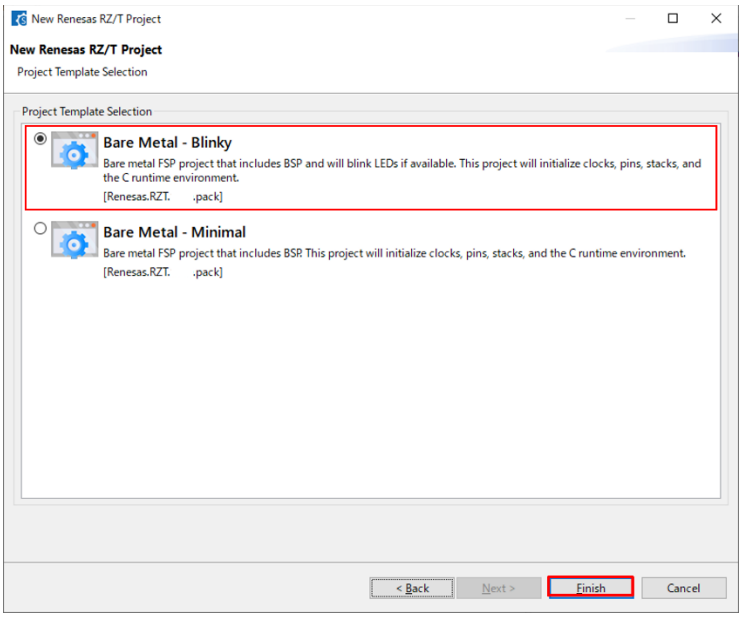


Figure 48 : Template Selection

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

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

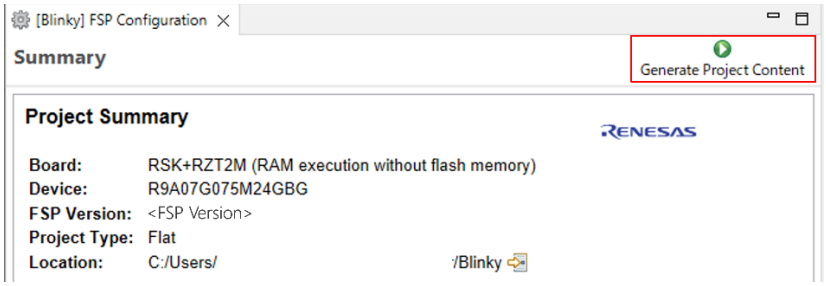


Figure 49 : FSP Project Configuration and Generation

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

(Continued on next page)

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

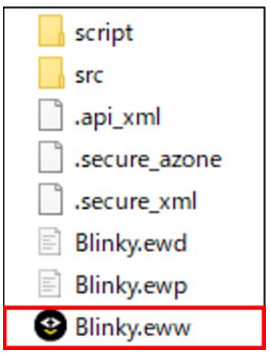


Figure 50 : FSP Project Workspace

5.3.3 Build the Project

5.3.3.1 NOTE: Build settings [Only Multiprocessing]

For multiprocessing, note the build order and build settings.

1. Create and build the primary project. (1st build of the primary project)
   No setting is required, proceed to5.3.3.2 Build.
2. Create the secondary project. Change the project options setting and build it.
   Set the following before building:
   i) Click **Project** > **Options…**.

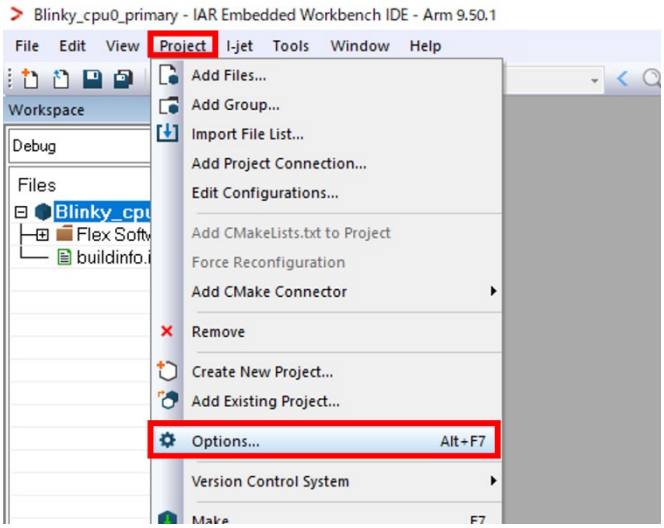


Figure 51 IAR EWARM project options

(Continued on next page)

ii) Click **Build Actions**.

iii) Remove **Build order**: Pre-compile from **Build actions**.

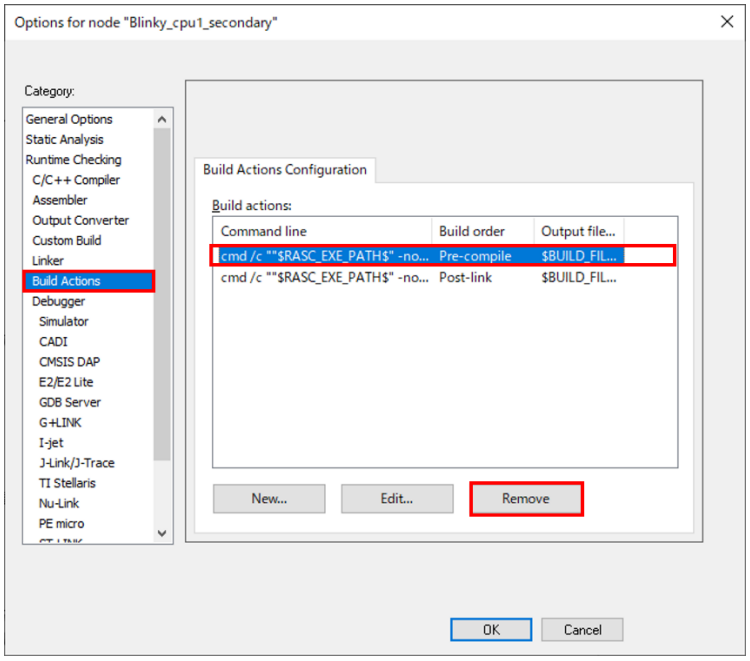


Figure 52 IAR EWARM project options for the secondary project (Build actions)

iv) Click **Debugger** > **Setup**.

v) Uncheck “Run to".

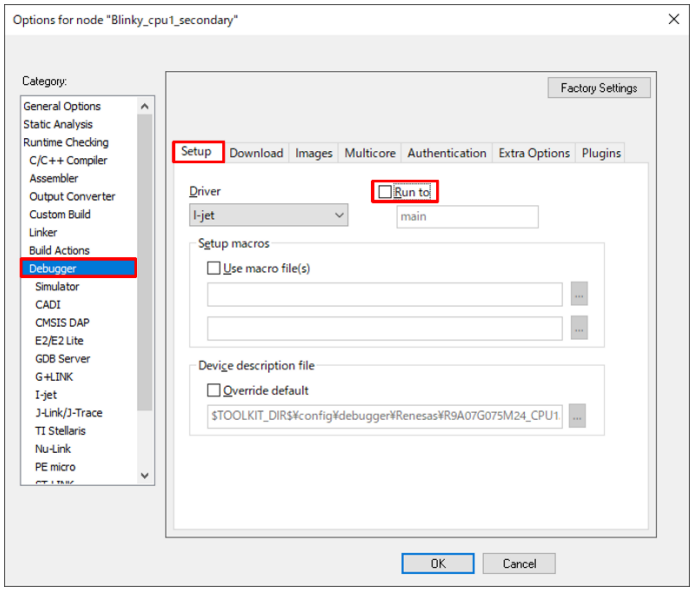


Figure 53 IAR EWARM project options for the secondary project (Run to)

vi) Click **I-jet** > **Setup**.

vii) Select **Software** as **Reset** and click **OK**.

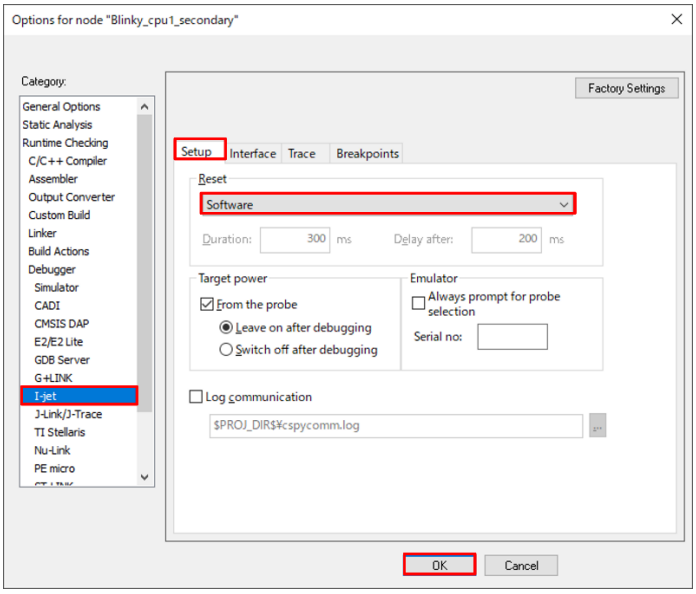


Figure 54 IAR EWARM project options for the secondary project (Reset)

viii) Proceed to 5.3.3.2 Build.

ix) Close the secondary project.

(Continued on next page)

3. Change the project options setting of the primary project and build it. (2nd build of the primary project)
   Set the following before building:
   i) Click **Project** > **Options…**.
   ii) Click **Debugger** > **Setup**.
   iii) Uncheck “Run to".
   iv) Click **Debugger** >**Multicore**.
   v) Set the following contents in **Asymmetric multicore**.
   - Simple
     - Partner workspace: the path of Blinky_cpu1_secondary.eww
     - Partner project: Blinky_cpu1_secondary
     - Partner configuration: Debug

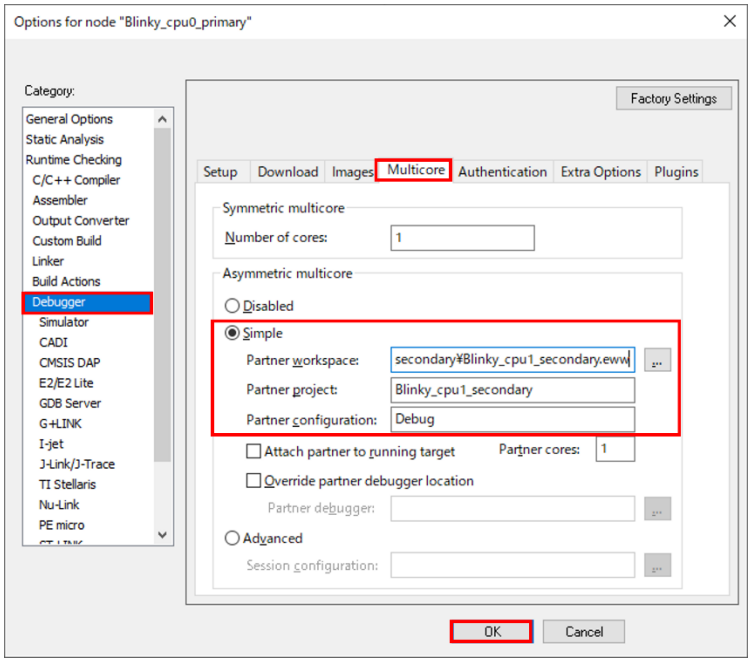


Figure 55 IAR EWARM project options for the primary project

vi) Proceed to 4.4.1 Build.

5.3.3.2 Build

Click on **Project** -> **Make** from menu bar or **Make** button on tool bar to build.

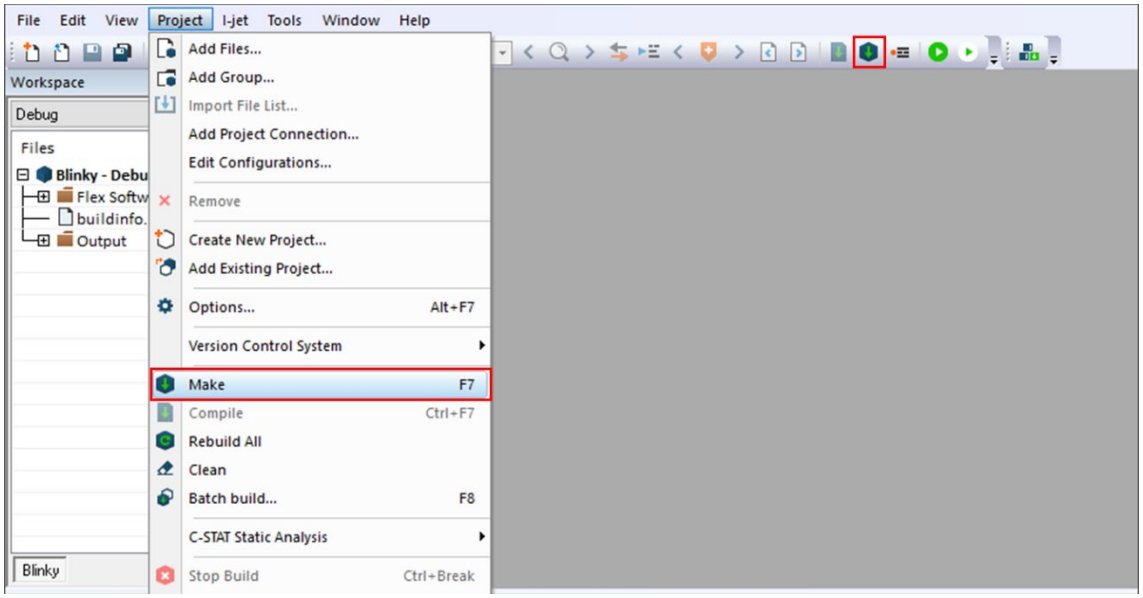


Figure 56 : Make Button

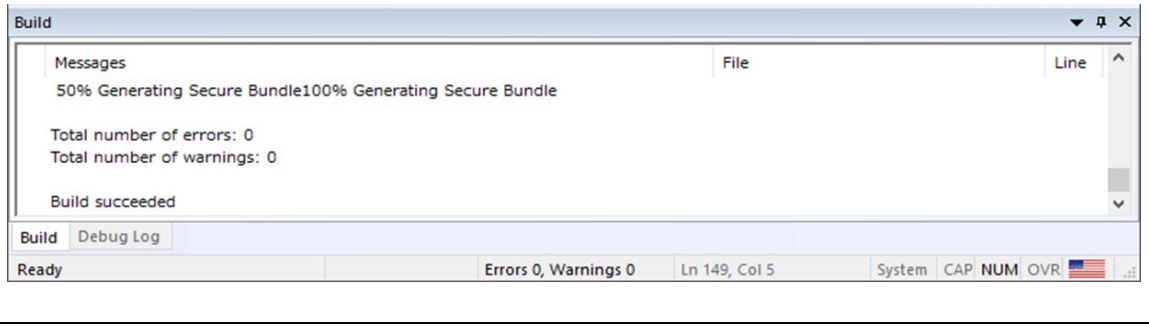


Figure 57 : Build Message Console

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

5.3.4 Download & Debug the Project

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

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

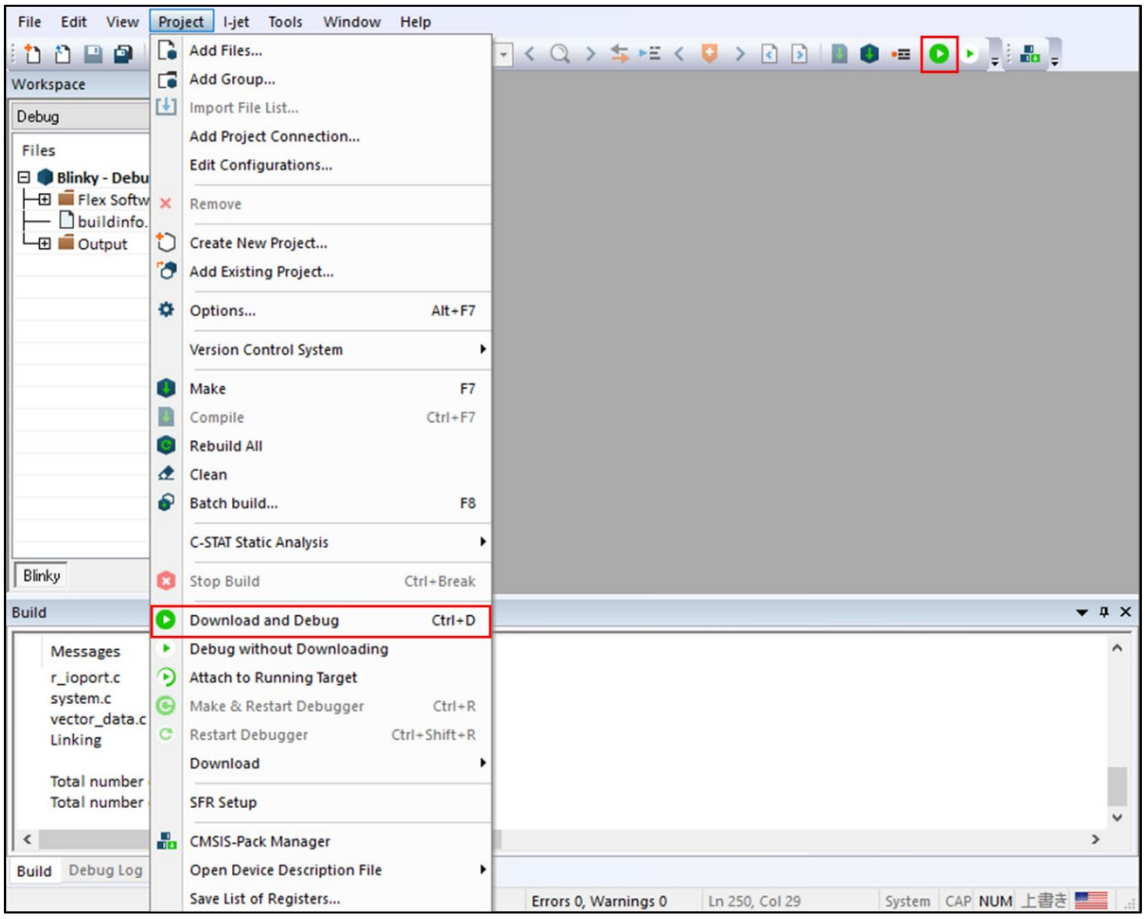


Figure 58 : Download and Debug Button

(Continued on next page)

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

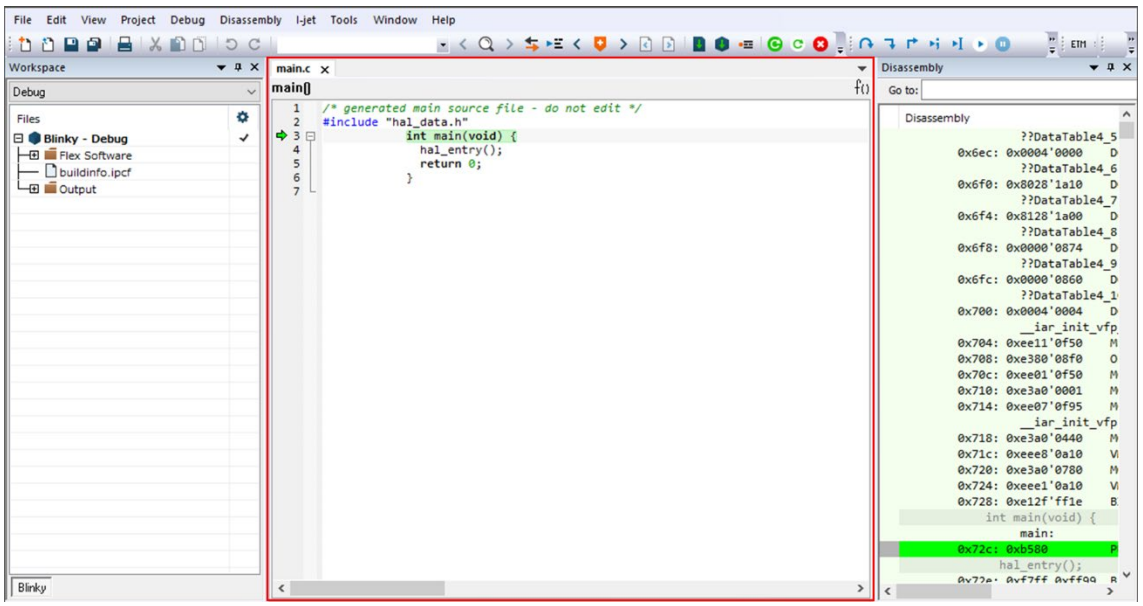


Figure 59 : Starting Debug

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

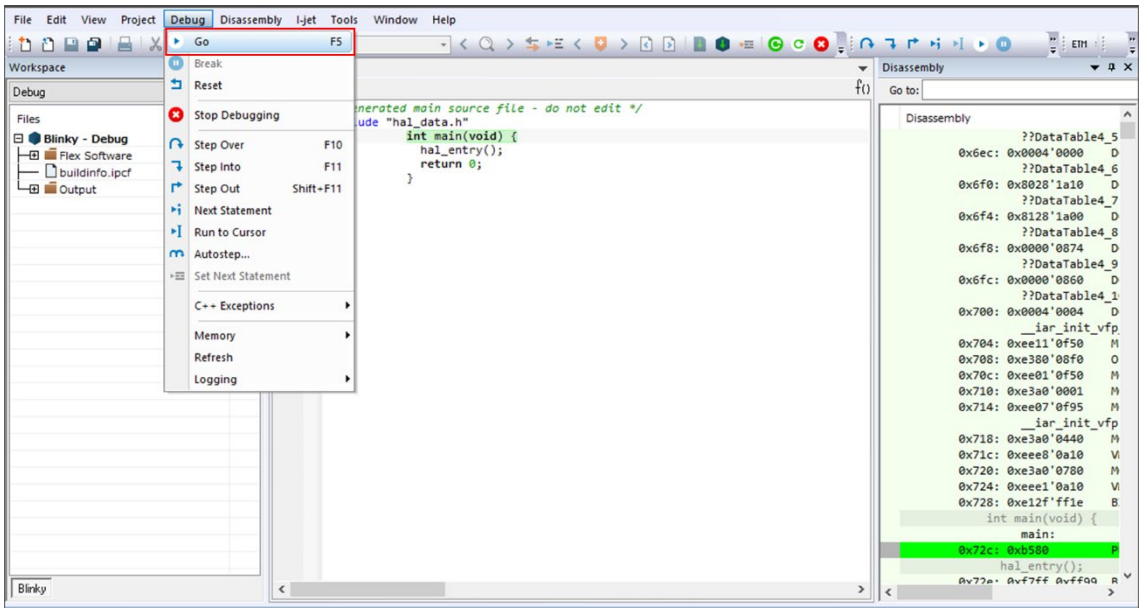


Figure 60 : Go Button

(Continued on next page)

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 IAR EWARM workspace view, the **hal_entry.c** is registered **Flex Software > Program Entry**.

The application performs the following steps:

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

On debugging on IAR EWARM, the break point can be set by click the left space next to line number.

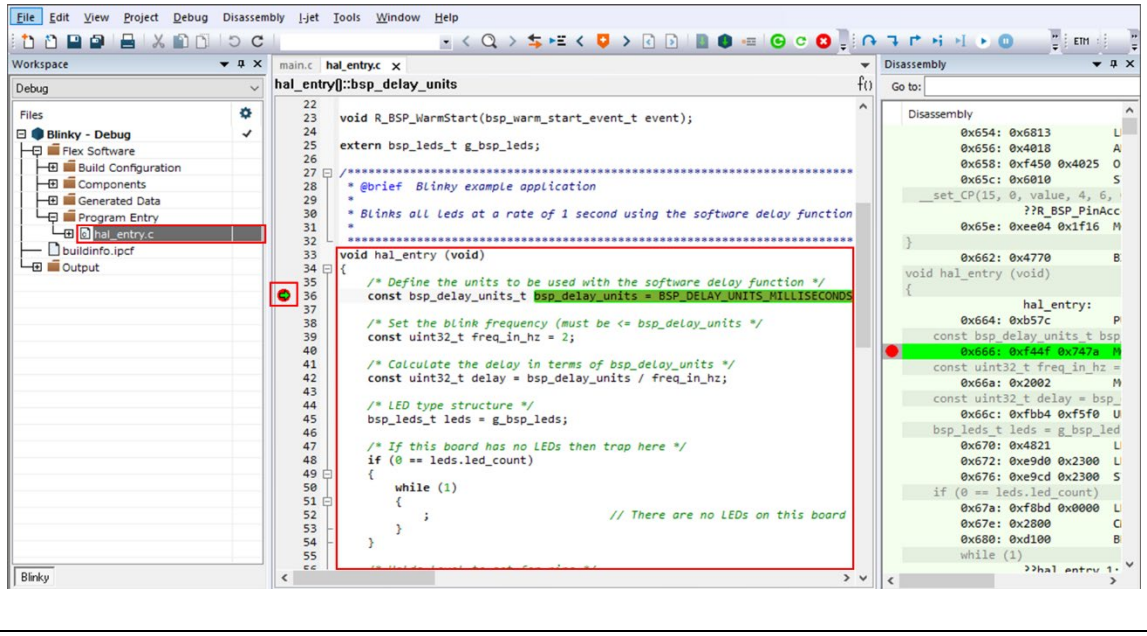


Figure 61 : hal_entry.c and Setting Breakpoint

(Continued on next page)

By using the break point and the **Debug** menu or **Debug** tool bar, you can check the behavior of the Blinky application step by step.

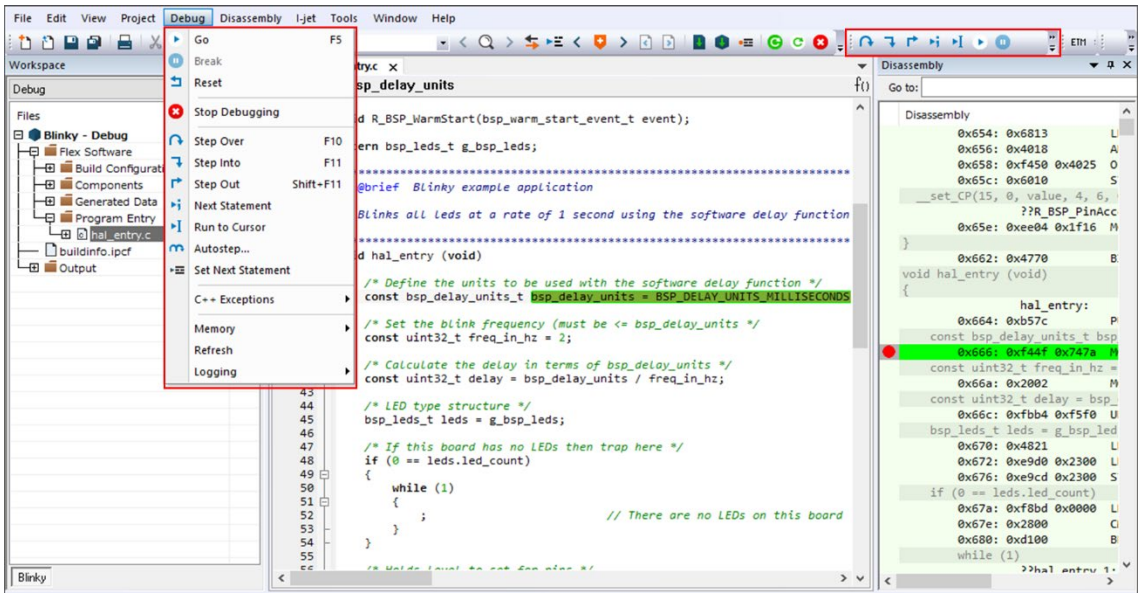


Figure 62 : Debug Menu

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

- RSK+RZ/T2M: LED0-1 (CPU0), LED2-3 (CPU1)
- RSK+RZ/T2L: LED0-6 (including LEDx_ESC_xxx)
- RSK+RZ/N2L: LED0-3

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

Figure 63 IAR EWARM Debugger Pause Icon

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

Figure 64 IAR EWARM Debugger Stop Icon

5.3.5 Debug for Multiprocessing

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

1. Open the primary project and close the secondary project on IAR EWARM.
2. Download of the primary project with procedure 5.3.4 Download & Debug the Project as shown in Figure 58.
3. The secondary project is automatically launched.
4. Run the program of primary project as shown in Figure 60. If the LED0 and LED1 are blinking, proceed to the next step.
5. The primary project in operation, run the program of secondary project.
6. When exiting Debug and disconnect from the debugger, if debugging is stopped in one of the projects, either the primary or the secondary, the other will automatically stop as well.

5.4 Re-configuring Project with FSP SC

For proceeding the tutorial with Blinky project, the FSP configuration steps of the Blinky project was skipped in this chapter. The FSP SC can be launched from IAR EWARM or command prompt, and the FSP project configuration can be re-configured by FSP SC.

There are two ways to launch FSP Smart Configurator with an exciting project.

5.4.1 Launch FSP Smart Configurator 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 $PROJ_DIR$

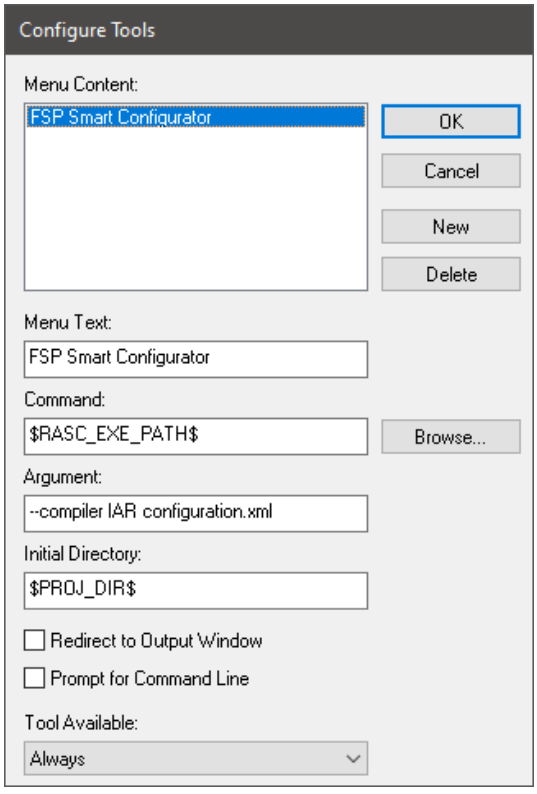


Figure 65 : Settings to Launch FSP SC from IAR EWARM

5.4.2 Launch from the Command Prompt.

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

5.5 Note when debugging in different workspaces

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

- Apply the same version of FSP package used for the project to the FSP SC.
- The project must be clicked the **Generate Project Content** button and built before debugging.

6. FSP Configuration Users Guide

6.1 What is a Project?

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

1. **Create a Project**
2. **Configuring a Project**

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

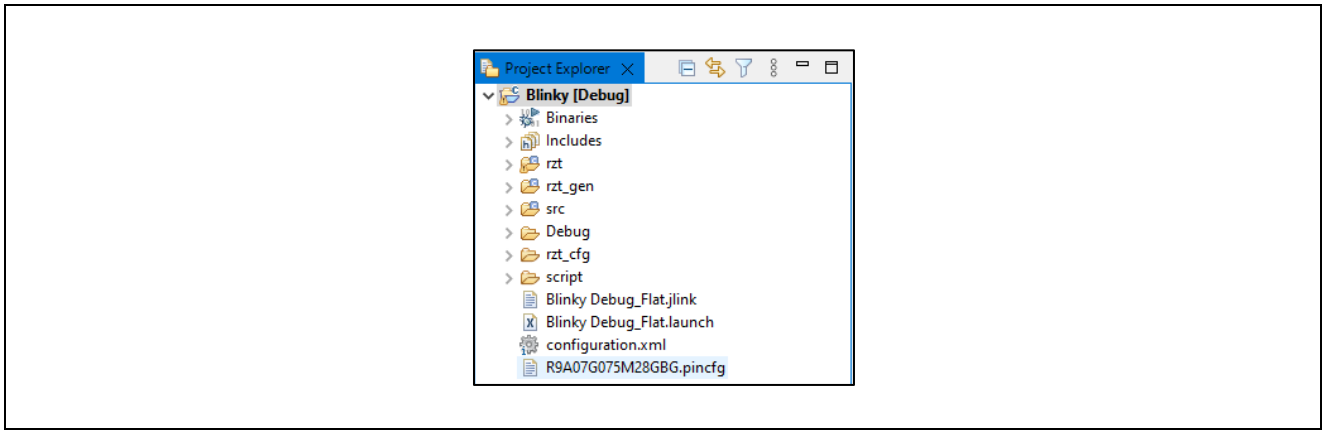


Figure 66 : e^2 studio Project Configuration File

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

Figure 67 : e^2 studio FSP Configuration Perspective

(Continued on next page)

Note:

Whenever the RZ/T2, RZ/N2 project configuration (that is, the configuration.xml file) is saved after configuring the project, a verbose RZ/T2, RZ/N2 Project Report file (rzt_cfg.txt, or rzn_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.

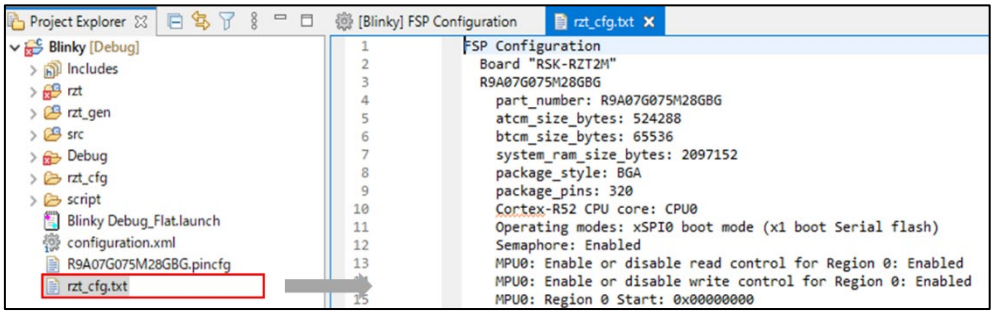


Figure 68 : RZ/T2, RZ/N2 Project Report

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

Note:

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

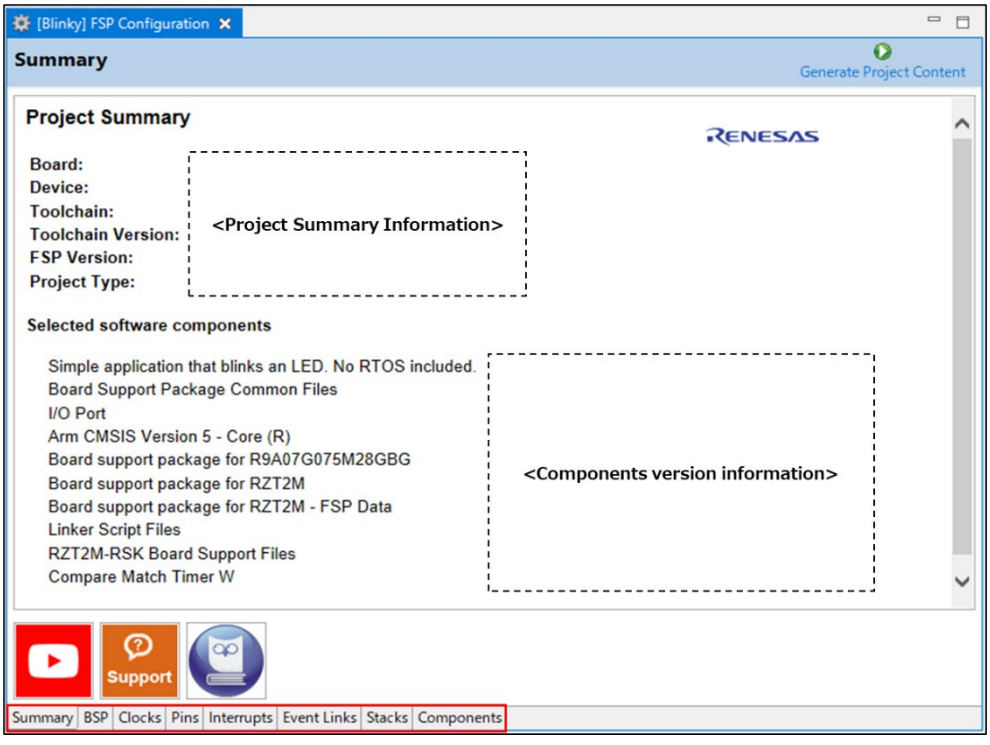


Figure 69 : RZ/T2, RZ/N2 Project Editor Tabs

6.2 Create a Project

6.2.1 Creating a New Project

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

1. Click on File > New > C/C++ Project.

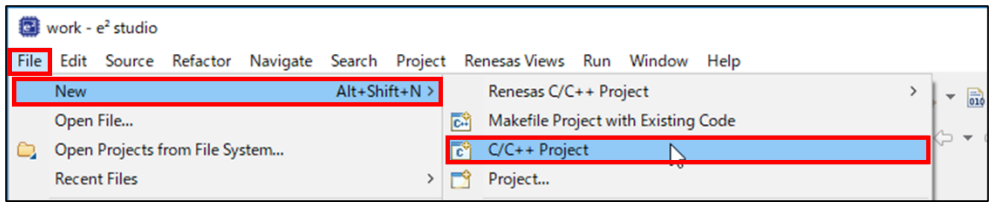


Figure 70 : New RZ/T2, RZ/N2 MPU Project

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

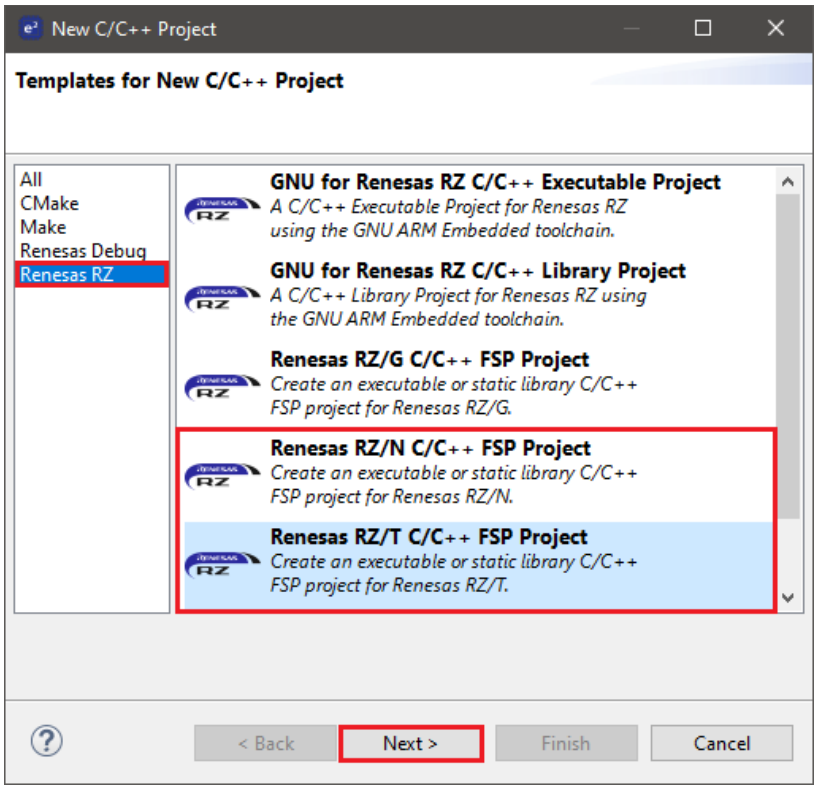


Figure 71 : New Project Templates

(Continued on next page)

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

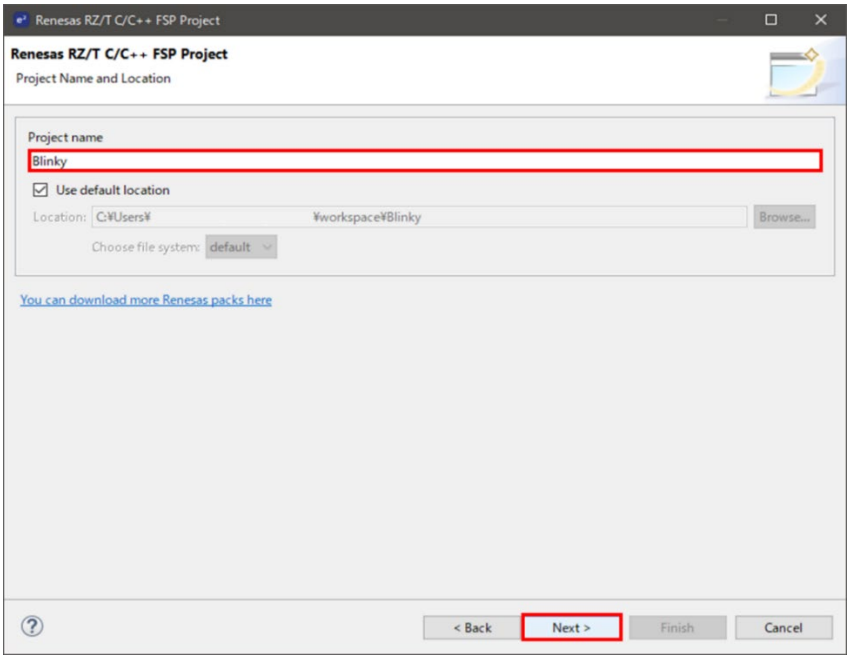


Figure 72 : RZ/T2, RZ/N2 MPU Project Generator (Part 1)

6.2.2 Selecting a Board and Toolchain

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

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

Note:

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

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

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

When you use **Custom User Board,**

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

3. The **Toolchain** selection defaults to **GNU Arm Embedded**.
   - Select the **Toolchain version**.
4. This should default to the installed toolchain version.
   - Select the **Debugger**.
5. The J-Link Arm Debugger is preselected.
6. Click **Next**.

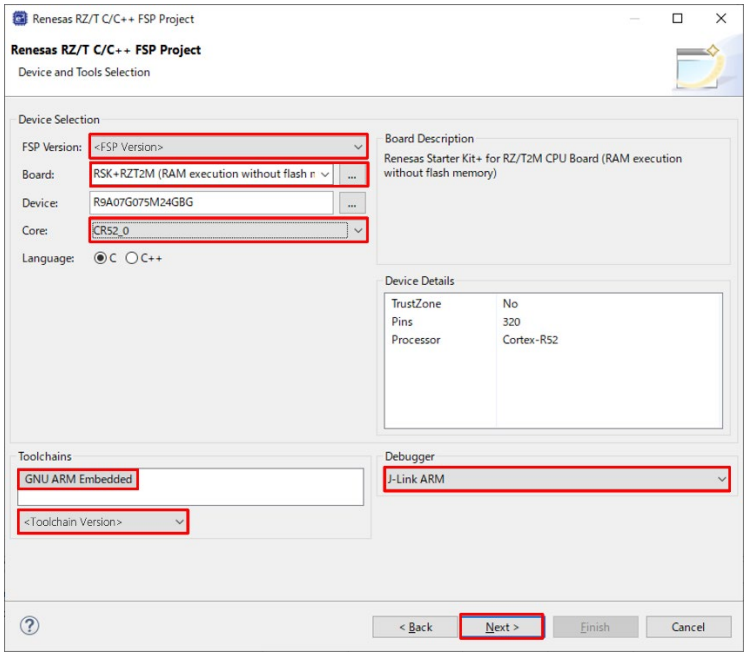


Figure 73 : RZ/T2, RZ/N2 MPU Project Generator (Part 2)

6.2.3 Selecting a Project Template

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

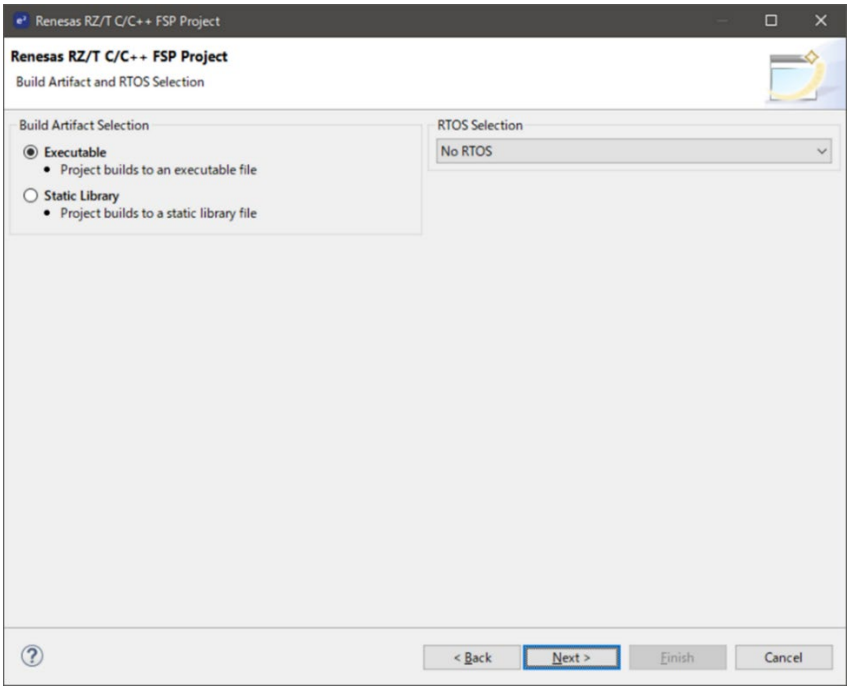


Figure 74 : RZ/T2, RZ/N2 MPU Project Generator (Part 3)

(Continued on next page)

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

Note:

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

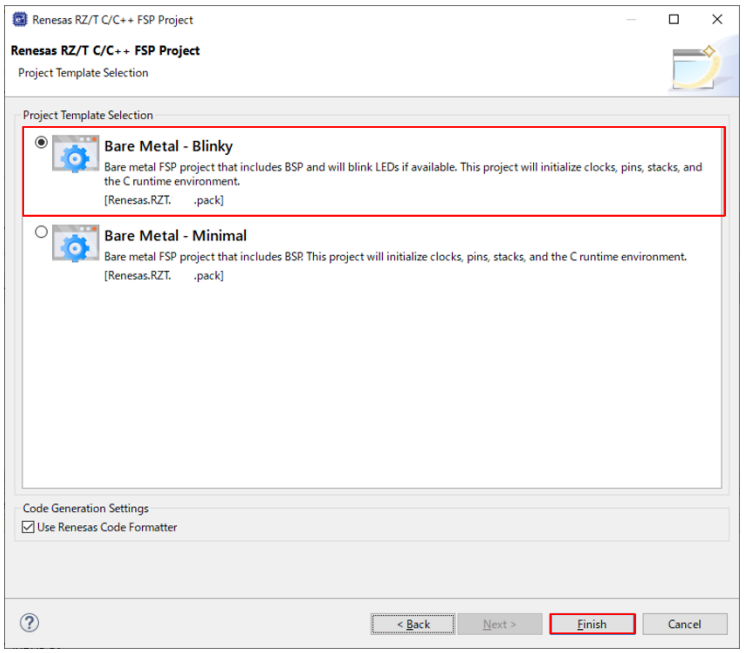


Figure 75 : RZ/T2, RZ/N2 MPU Project Generator (Part 4)

(Continued on next page)

When the project is created, e² studio displays a summary of the current project configuration in the RZ/T2, RZ/N2 MPU Project Editor.

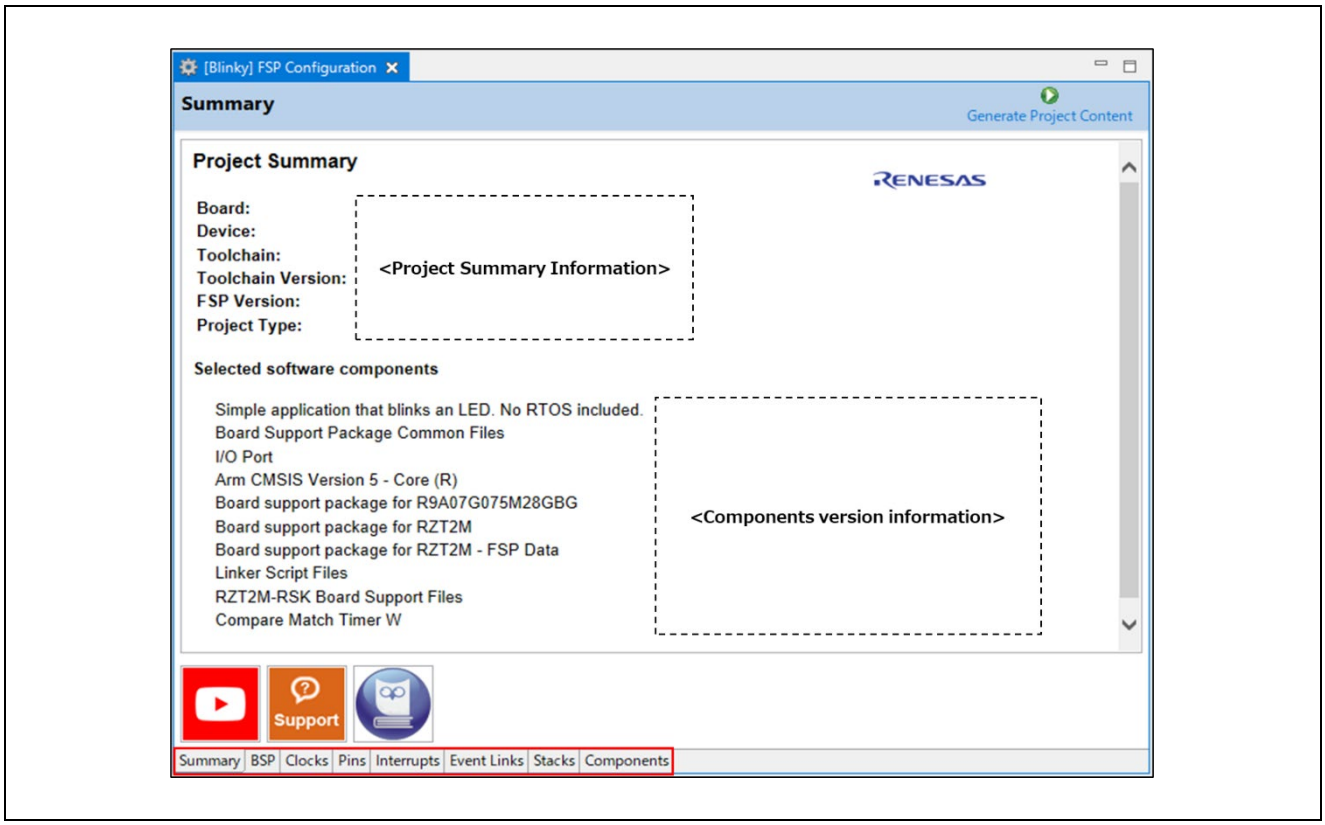


Figure 76 : RZ/T2, RZ/N2 MPU Project Editor and Available Editor Tabs

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

- With the **Summary** tab, you can see all they 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, 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.

6.3 Configuring a Project

Each of the configurable elements in an FSP project can be edited using the appropriate tab in the RZ/T2, RZ/N2 Configuration editor window. Importantly, the initial configuration of the MPU after reset and before any user code is executed is set by the configuration settings in the **BSP** tab. When you select a project template during project creation, e^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

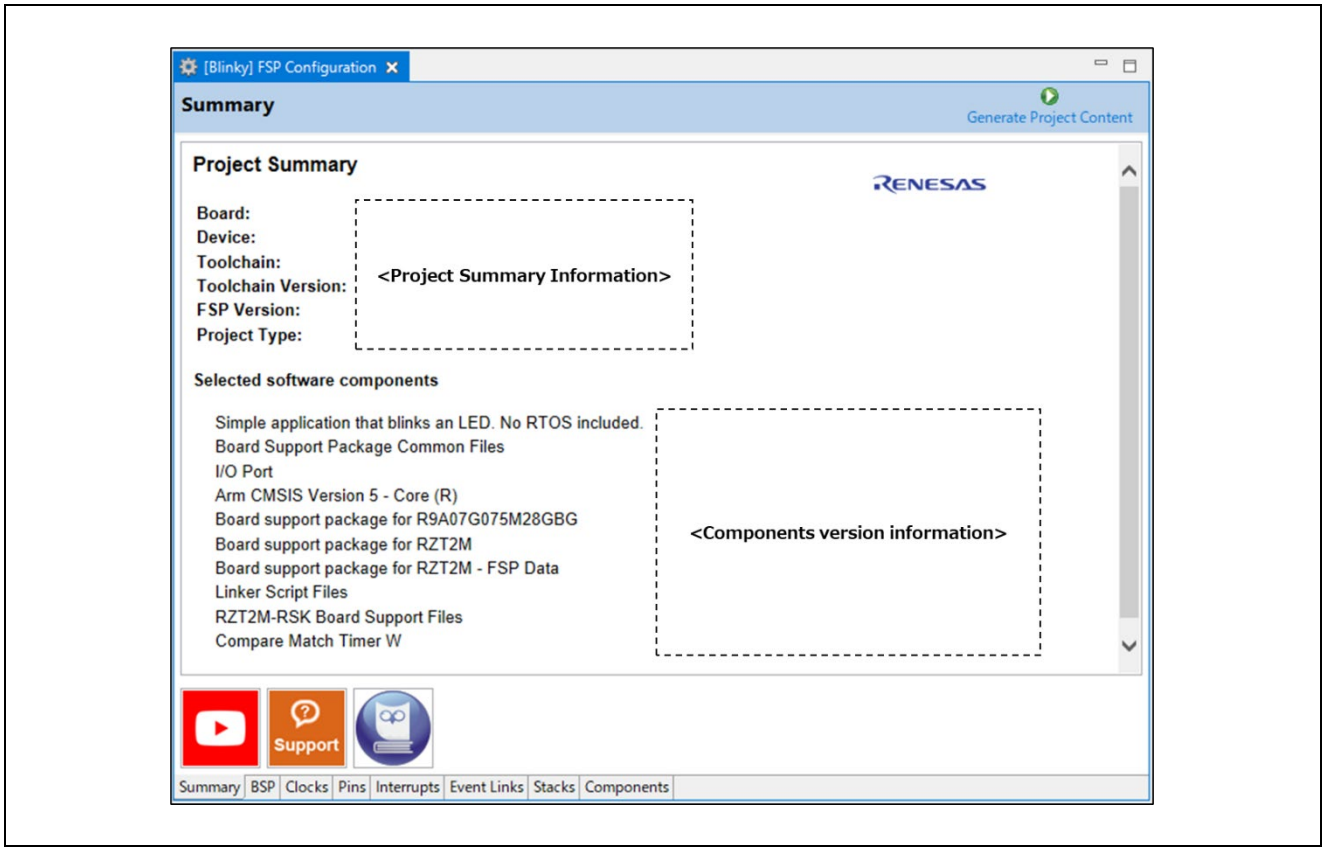


Figure 77 : Configuration Summary Tab

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, toolchain and 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.

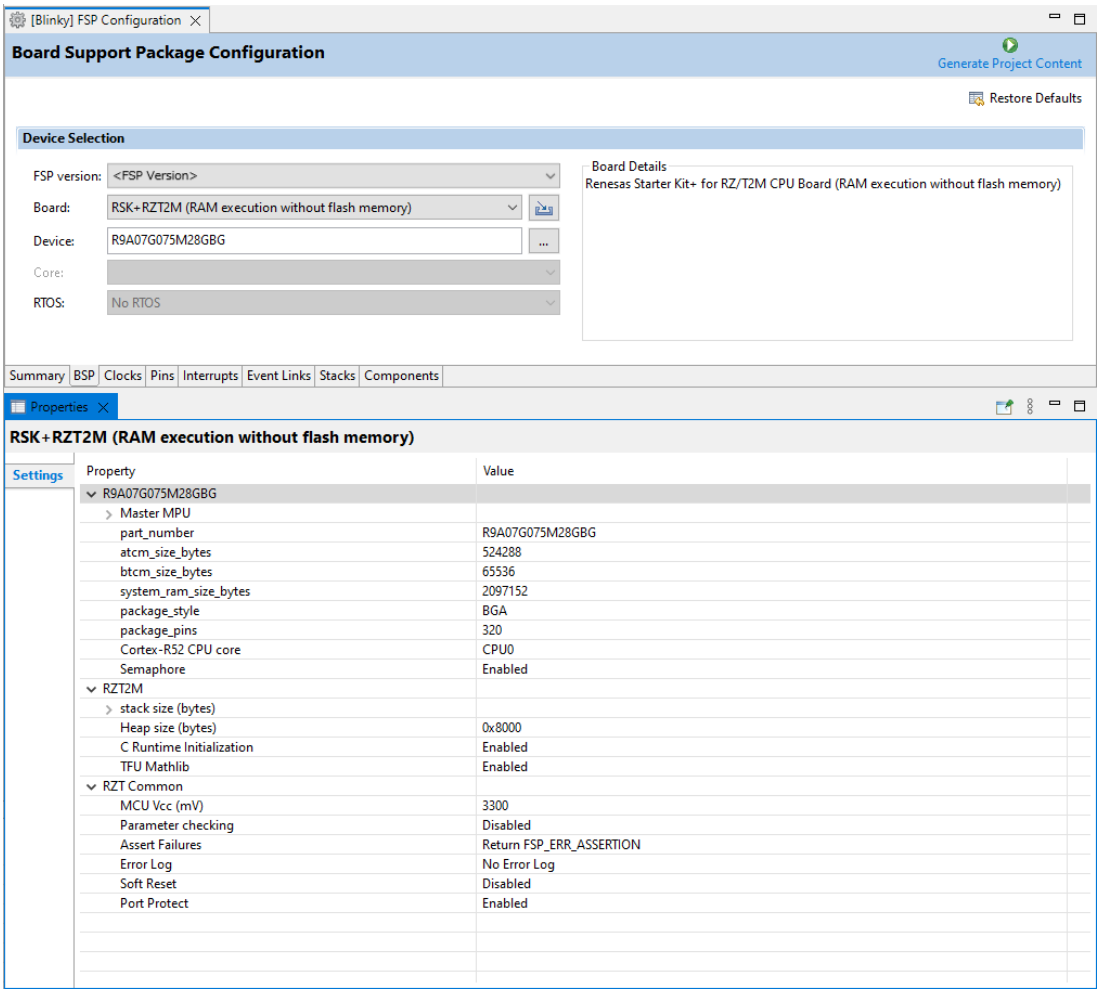


Figure 78 : Configuration BSP Tab

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:

- rzt_cfg/fsp_cfg/bsp/bsp_cfg.h, or
- rzn_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.

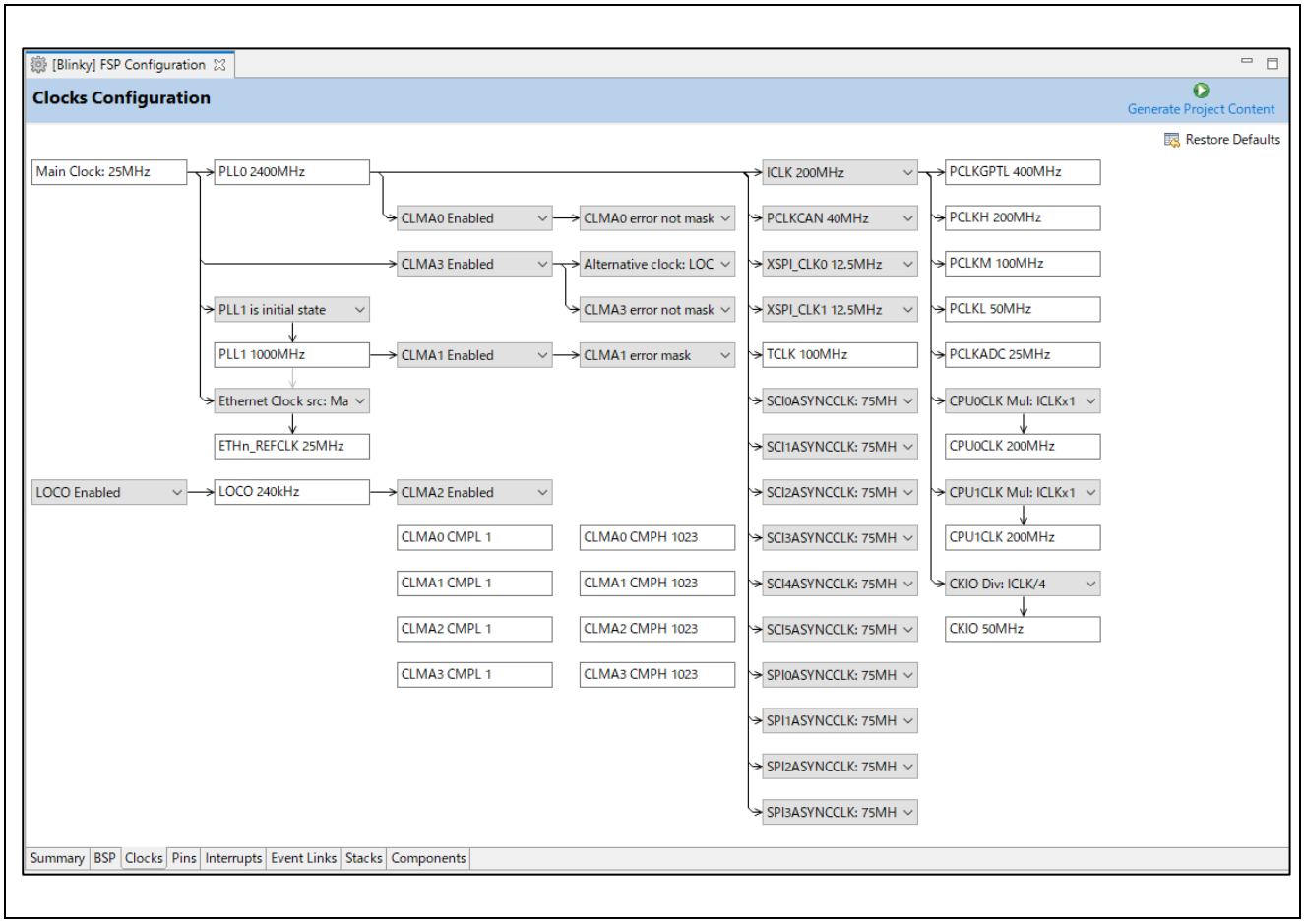


Figure 79 : Configuration Clocks Tab

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

- rzt_gen/bsp_clock_cfg.h, or
- rzn_gen/bsp_clock_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.

6.3.4 Configuring Pins

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

Note:

If the **Package** view window is not open in e^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 on the board are preselected.

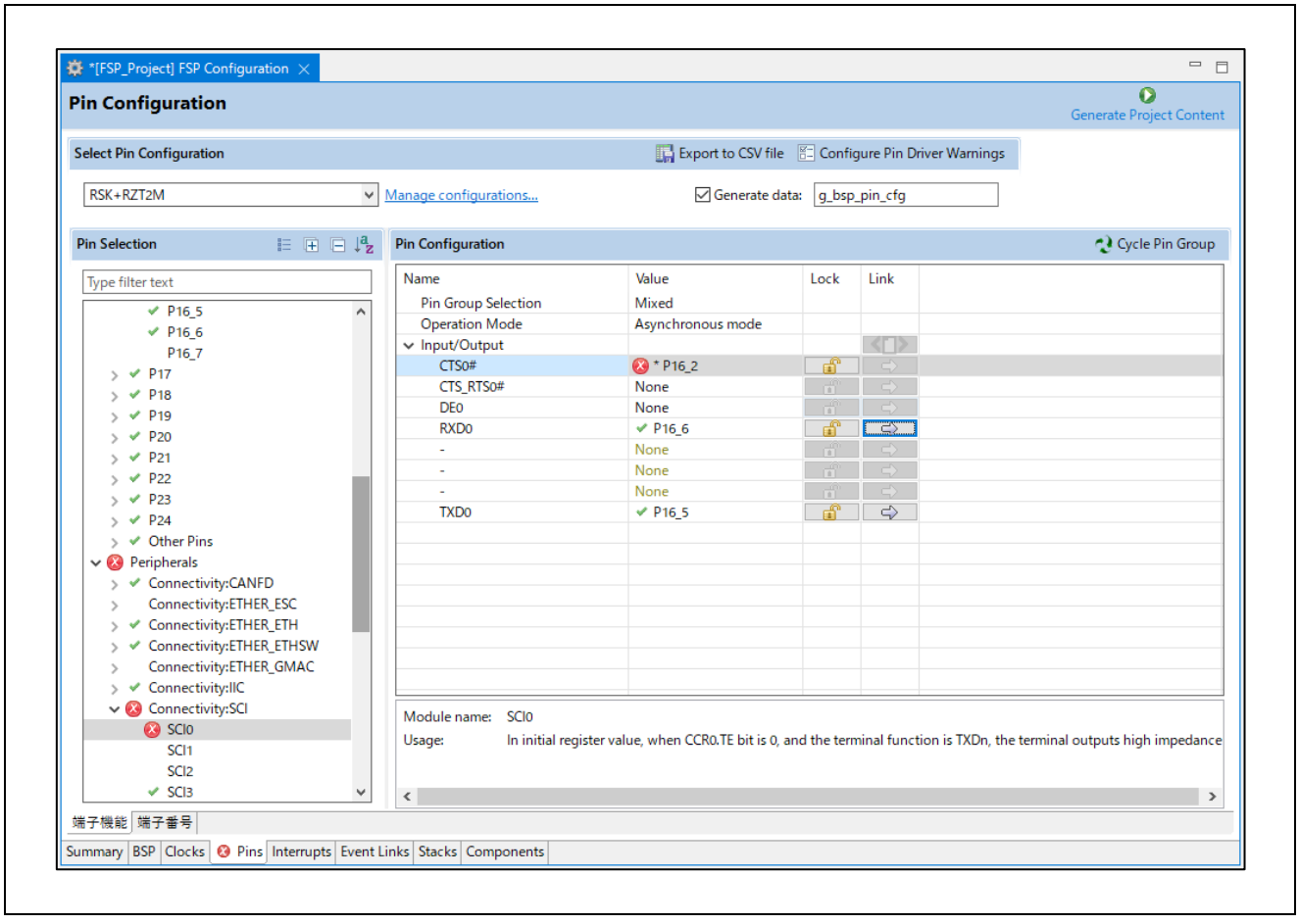


Figure 80: Pin Configuration

(Continued on next page)

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

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

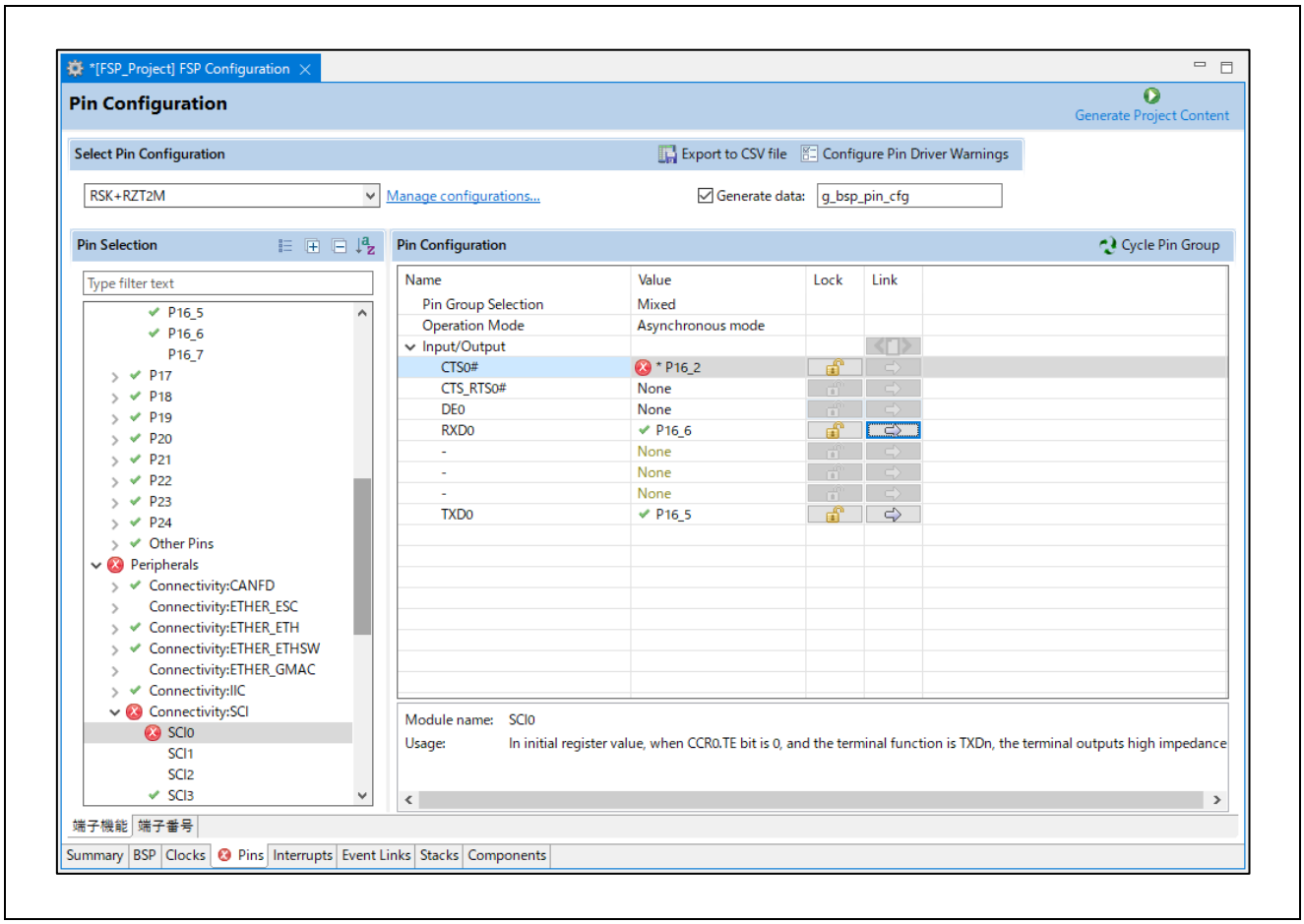


Figure 81: Conflict Checker in Pin Configuration

(Continued on next page)

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

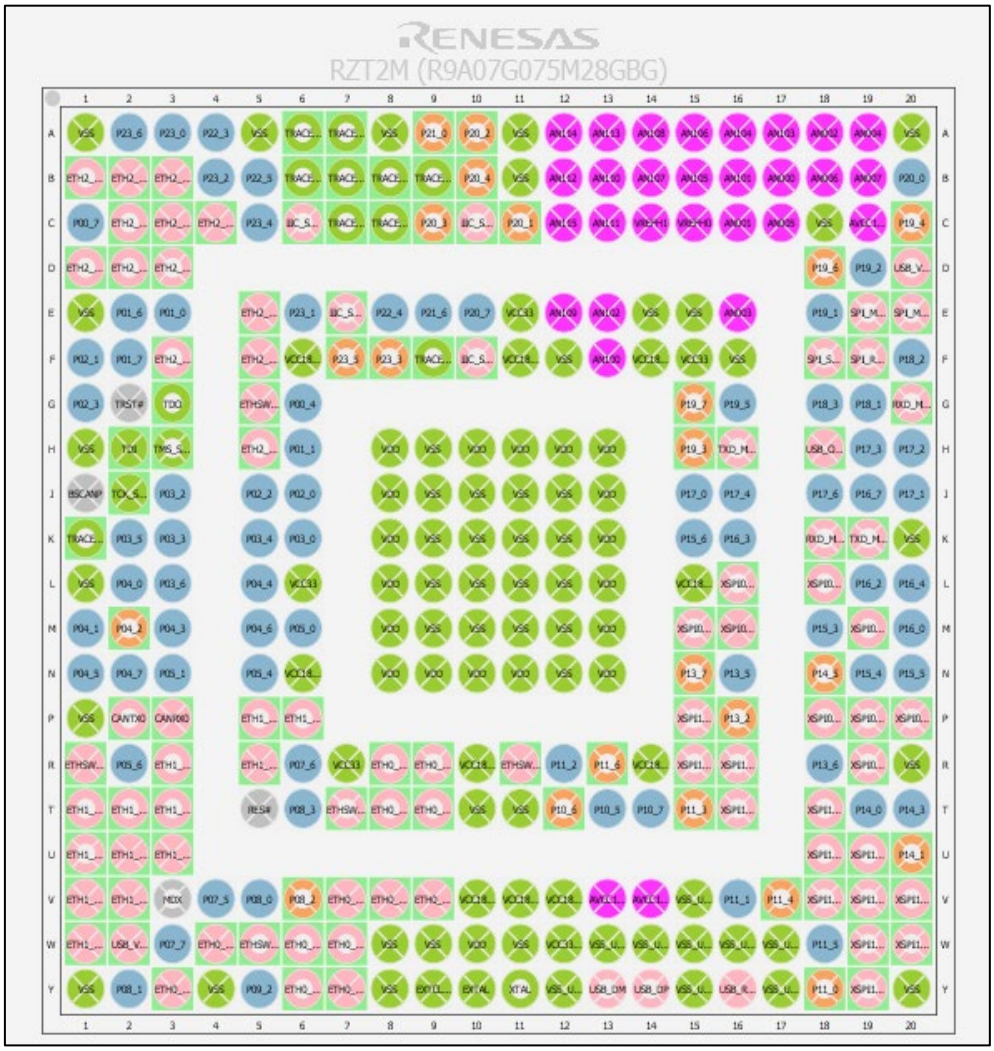


Figure 82: Pin Configurator Package View

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

- rzt_gen\bsp_pin_cfg.h, or
- rzn_gen\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.

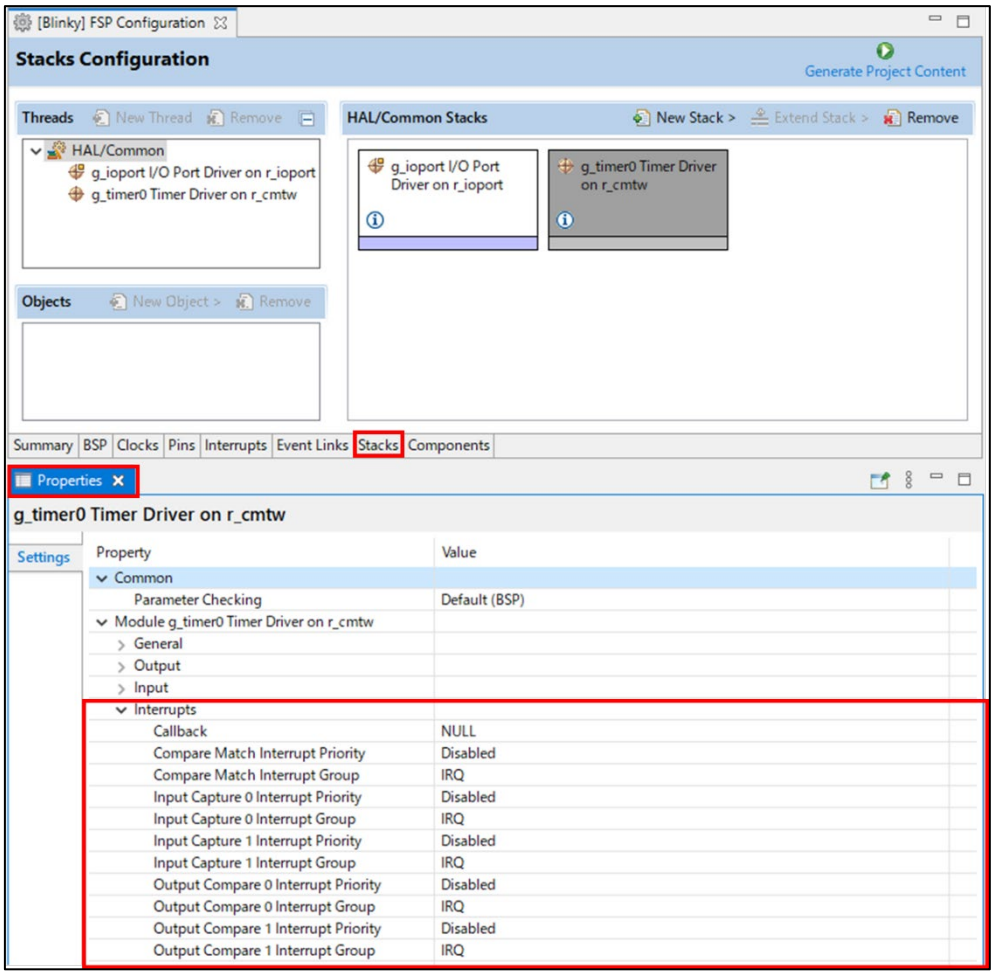


Figure 83 : Configuring Interrupts in the Stacks Tab

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.

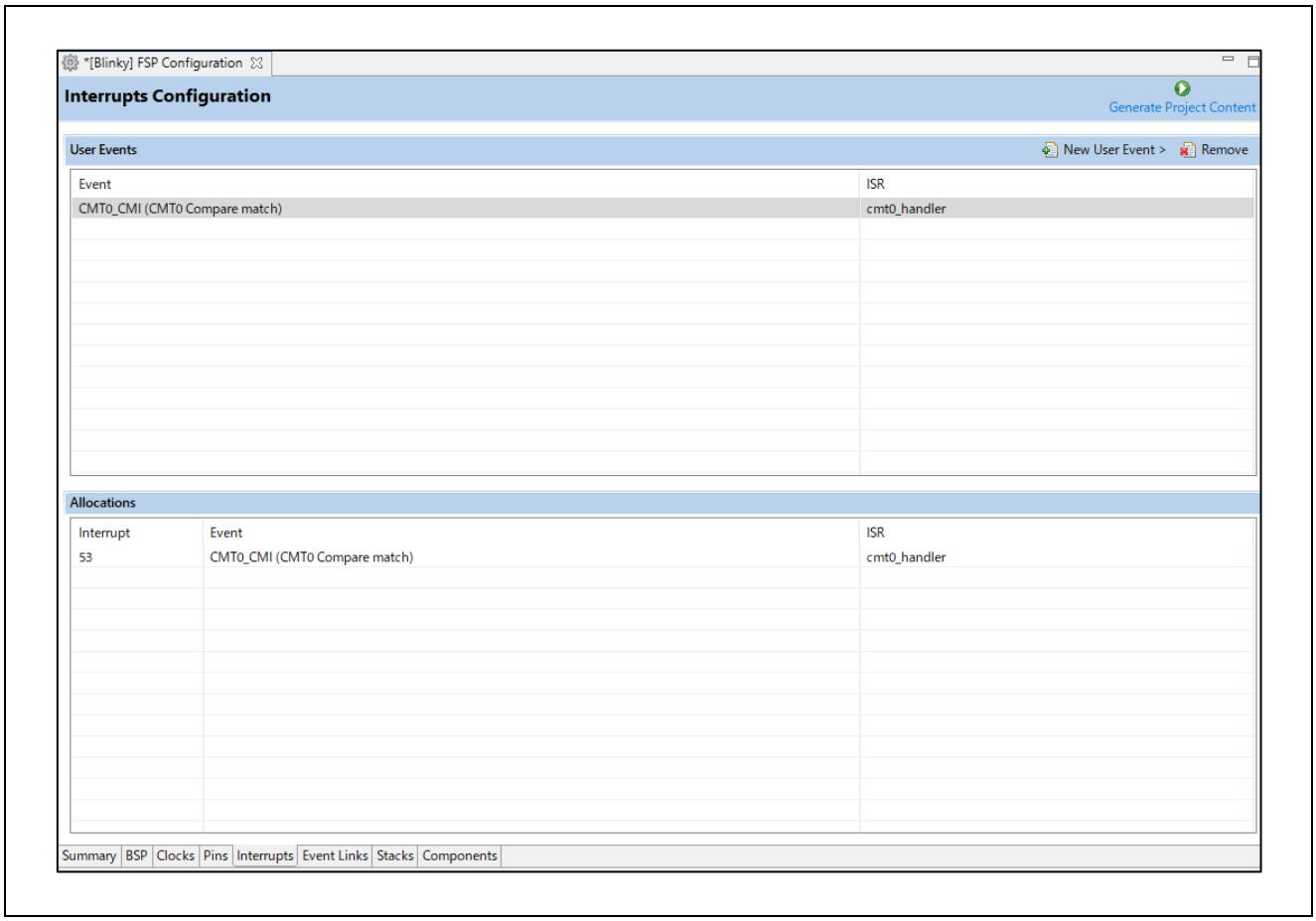


Figure 84 : Configuring Interrupt in Interrupt Tab

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

6.4.2 Viewing Event Links

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

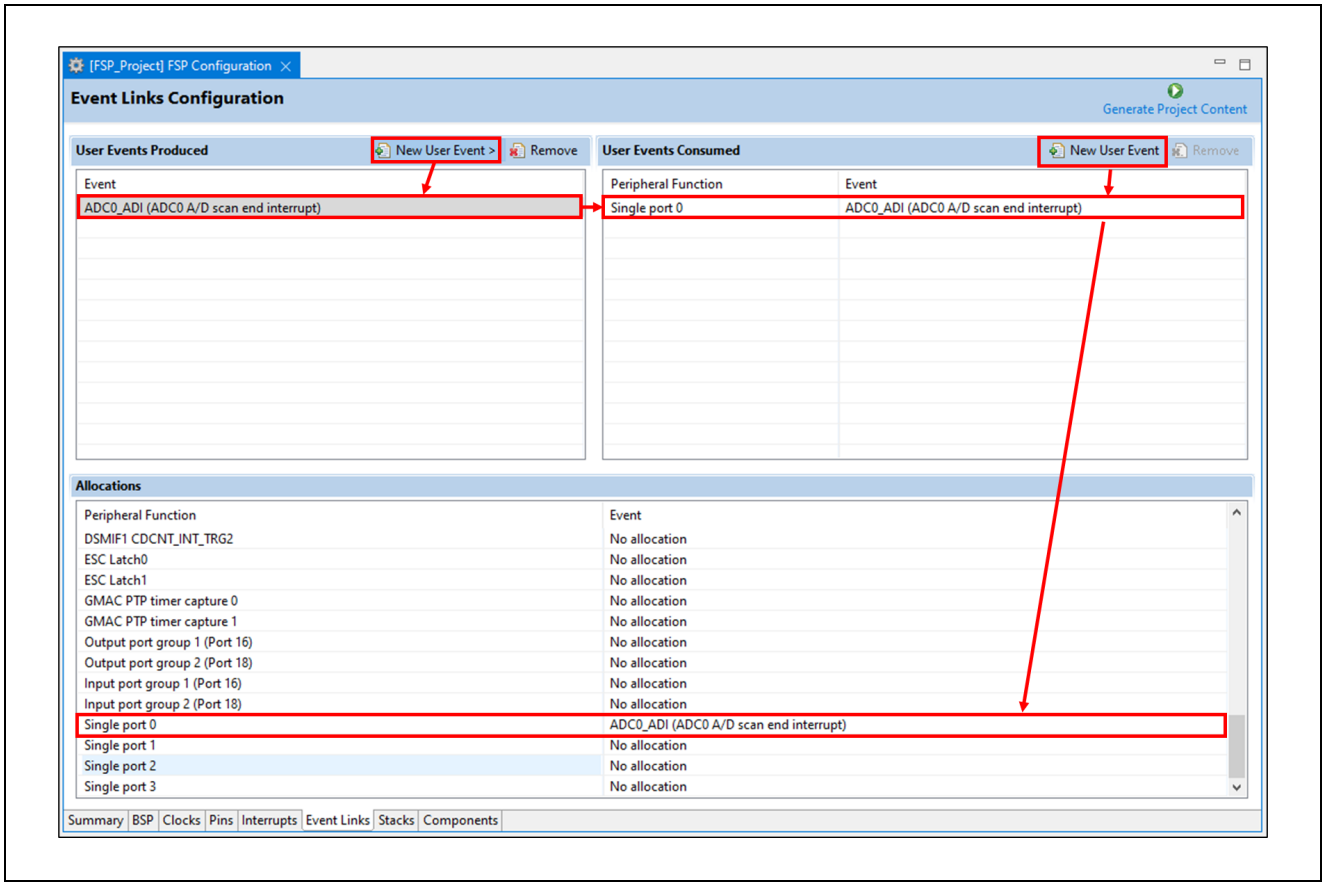


Figure 85 : Viewing Event Links

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

Note:

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

6.5 Adding and Configuring HAL Drivers

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

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

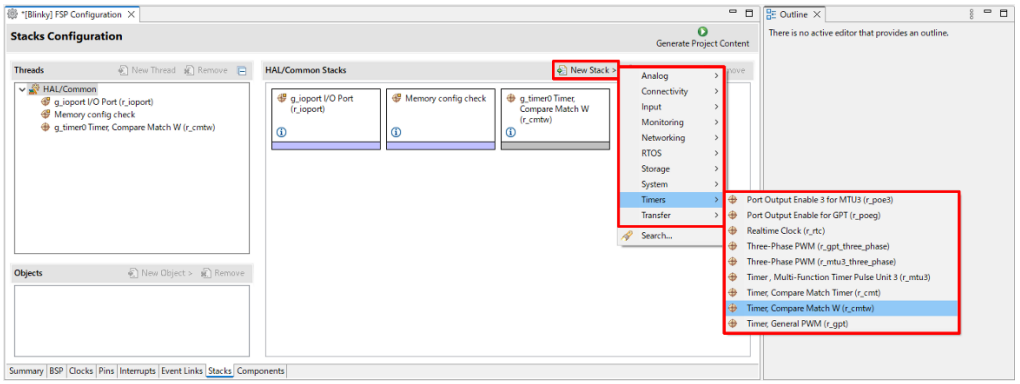


Figure 86 : e² studio Project Configurator – Adding Drivers

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

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

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

Table 5 Generate Contents on Smart Configurator

File	Contents	Overwritten by Generate Project Content?
rzt_gen/main.c or rzn_gen/main.c	Contains main() calling generated and user code. When called, the BSP already has Initialized the MPU.	Yes
rzt_gen/hal_data.c or rzn_gen/had_data.c	Configuration structures for HAL Driver only modules.	Yes
rzt_gen/hal_data.h or rzn_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:

- rzt_cfg/fsp_cfg or
- rzn_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.

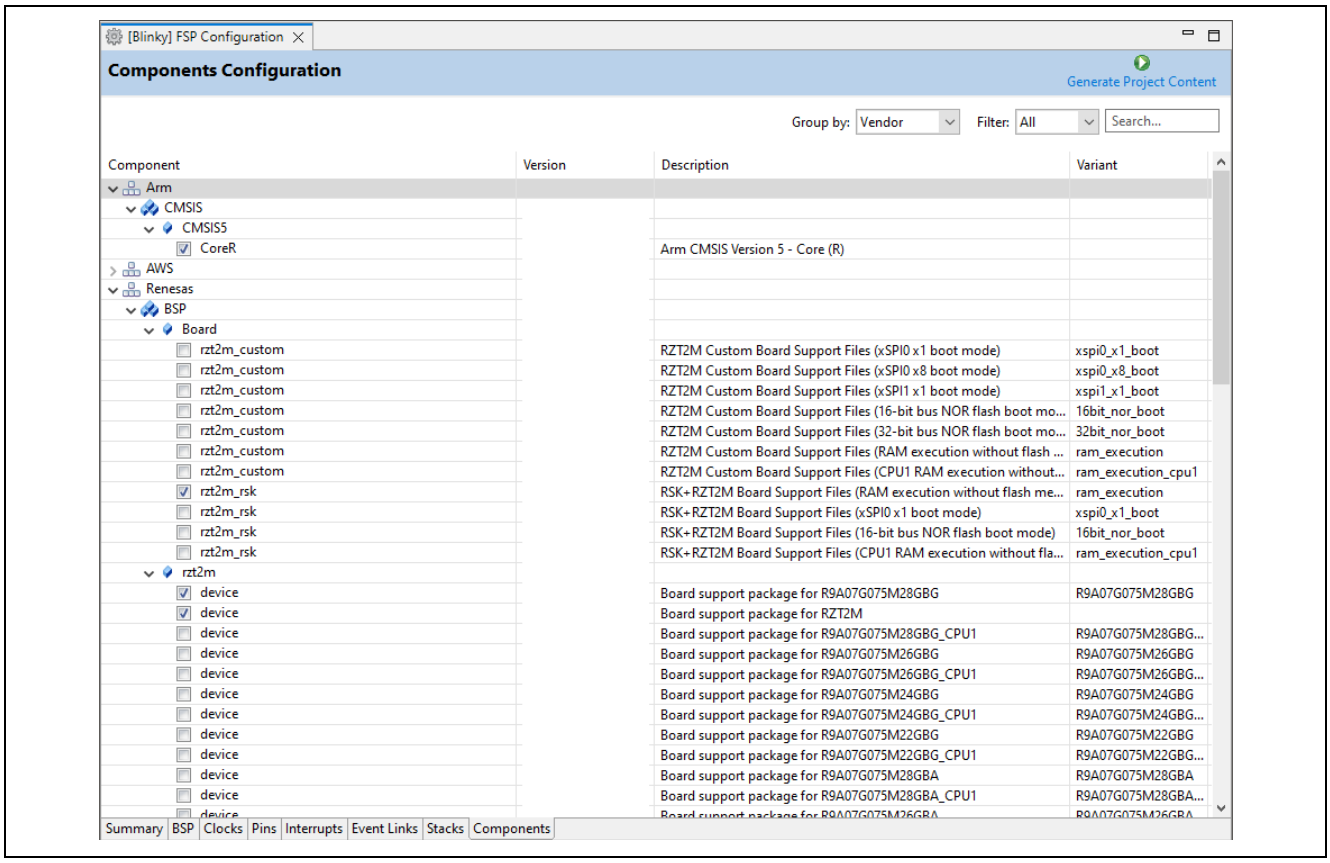


Figure 87 : Components Tab

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

- rzt/fsp/inc/api
- rzt/fsp/inc/instances
- rzt/fsp/src/bsp
- rzt/fsp/src/<Driver_Name>

or

- rzn/fsp/inc/api
- rzn/fsp/inc/instances
- rzn/fsp/src/bsp
- rzn/fsp/src/<Driver_Name>

e^2 studio also creates configuration files in the following folder with configuration options set in the **Stacks** tab.

- rzt_cfg/fsp_cfg
- rzn_cfg/fsp_cfg

Appendix. Known Issues

This chapter describes the known issues regarding the current version of FSP and related platform software.

Most of the issues may require users to follow some manual operations to resolve the issues or to avoid the problems caused by the issues. Please follow the operations in the description of the issues if you use the features related to the issues. The grayed-out items have been resolved.

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

- FSP Configuration

  e^2 studio and FSP SC have various configuration features worked on GUI with FSP.

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

- FSP Modules

  The FSP provides HAL drivers and BSP configured by FSP Configuration on e^2 studio and FSP SC.

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

Table 6 List of Known Issues

No.	Title	Target Device			Category
		T2M	T2L	N2L	
1	“r_gmac” may be showed as “r_ether” incorrectly.	✓		✓	FSP Configuration, Stacks
2	“Edge” can be selected as Transfer End Interrupt Detect Type in “r_dmac”, but it cannot be used.	✓	✓	✓	FSP Configuration, Stacks
3	When the “Device” or “Board” selection in BSP tab is changed, the BSP properties are sometimes configured to incorrect configuration.	✓	✓	✓	FSP Configuration, BSP
4	(FSP SC ONLY) Device name is not output correctly depending on the selected device.	✓			FSP Configuration, BSP
5	Errors occur when changing board settings.		✓	✓	FSP Configuration, BSP
6	Pin configuration error occurs in MPX-IO 16bit operating mode of “r_bsc”.	✓	✓	✓	FSP Configuration, Pins
7	Build error when using definition name of input/output external pins for module.	✓	✓	✓	FSP Configuration, Pins
8	“R_SCI_UART_BaudCalculate()” of “r_sci_uart” module properly works ONLY when its clock source is SCInASYNCCLK and its frequency is 96MHz.	✓	✓	✓	FSP Modules, SCI UART
9	“R_SPI_CalculateBitrate()” of “r_spi” module properly works ONLY when its clock source is SPInASYNCCLK and its frequency is 96MHz.	✓	✓	✓	FSP Modules, SPI
10	A warning occurs when building “r_gmac” module with the gcc compiler.	✓	✓		FSP Modules, Ethernet
11	In FSP Documentation, there is incorrect description in. “API Reference > Modules > Ethernet PHY” page.	✓		✓	FSP Modules, Ethernet PHY
12	The interrupt number cannot be successfully acquired by the R_FSP_CurrentIrqGet() when multiple interrupt occurs.			✓	FSP Modules, FreeRTOS
13	Block Media Custom Implementation can be selected as Memory Implementation for “rm_freertos_plus_fat” module, but it cannot be used.	✓	✓	✓	FSP Configuration, Stacks

14	The second argument of “r_mtu3” APIs do not match with common API.	✓	✓	✓	FSP Modules, MTU3
15	In multiprocessing, a configuration error occurs when “r_gpt” module is used for both projects for CPU0 and CPU1.	✓			FSP Configuration, Stacks
16	Project build error occur when 32-bit bus NOR flash and xSPI0 x8 boot modes are selected on RZT Custom User Board.	✓			FSP Configuration, BSP
17	The secondary project for multiprocessing cannot be created when xSPI1 x1 boot modes are selected on RZT Custom User Board.	✓			FSP Configuration, BSP

No. 1 Resolved

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

No. 2 Resolved

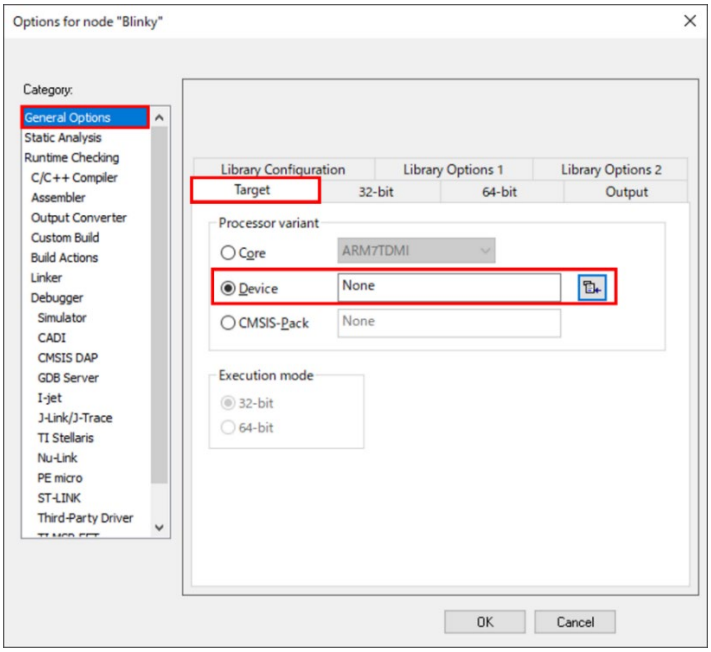
Title	**“Edge” can be selected as Transfer End Interrupt Detect Type in “r_dmac”, but it cannot be used.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**FSP Configuration, Stacks**
Description	“Edge” of interrupt detect type is not available due to a change in hardware specifications.
Workaround	Please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

No. 3

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

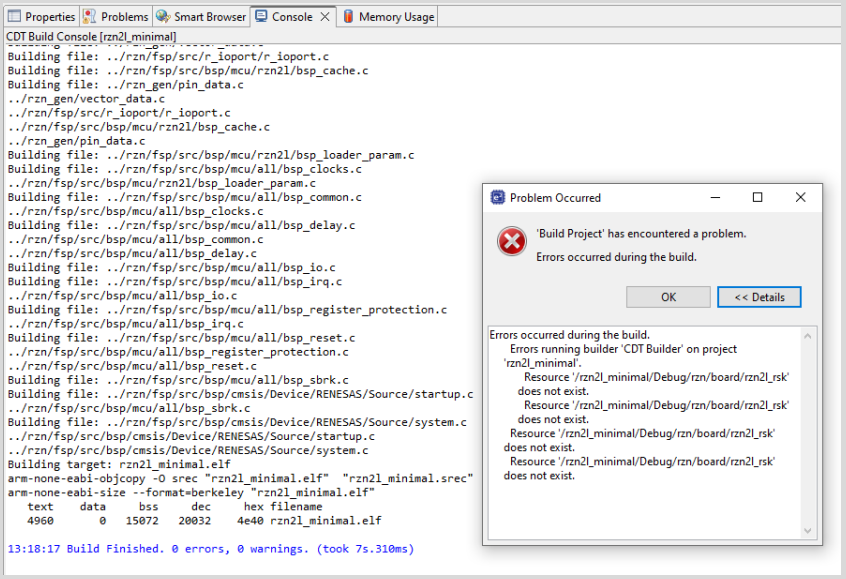
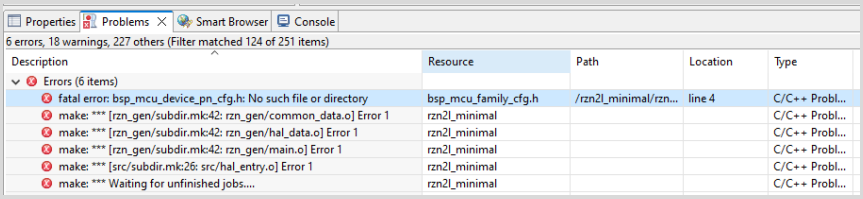
(Continued on next page)

No. 4

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

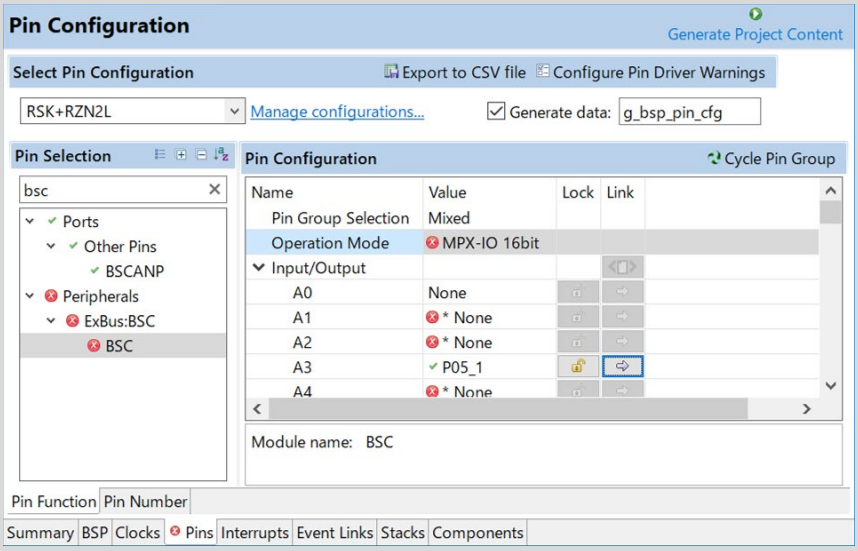
(Continued on next page)

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

Title	**Errors occur when changing board settings.**
Target	**RZ/T2L, RZ/N2L**
Category	**FSP Configuration, BSP**
Description	Errors occur when changing board settings from RSK+RZN2L (RAM execution without flash memory) to RZN2L Custom User Board (RAM execution without flash memory) and from RSK+RZN2L (RAM execution without flash memory) to RZN2L Custom User Board (xSPI0 x1 boot mode). 1. Changed from RSK+RZN2L (RAM execution without flash memory) to RZN2L Custom User Board (RAM execution without flash memory) In this case, the build is successful. But the following screen is displayed after the build.  2. Changed from RSK+RZN2L (RAM execution without flash memory) to RZN2L Custom User Board (xSPI0 x1 boot mode) bsp_mcu_device_pn_cfg.h is not generated and builds error occurs. 
Workaround	Please create a new project to change the board setting to RZN2L Custom User Board.

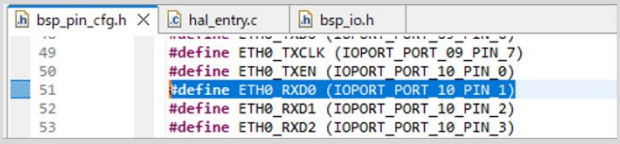
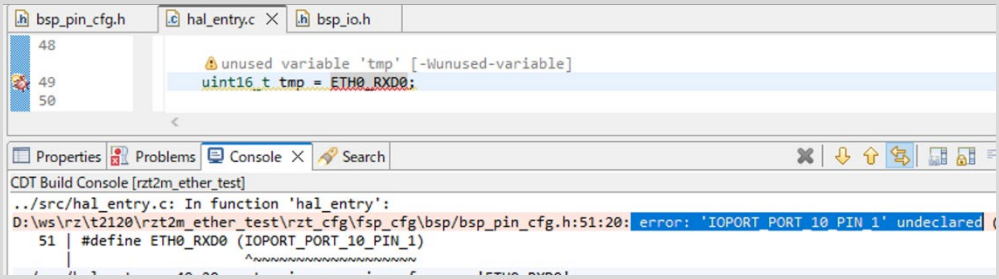
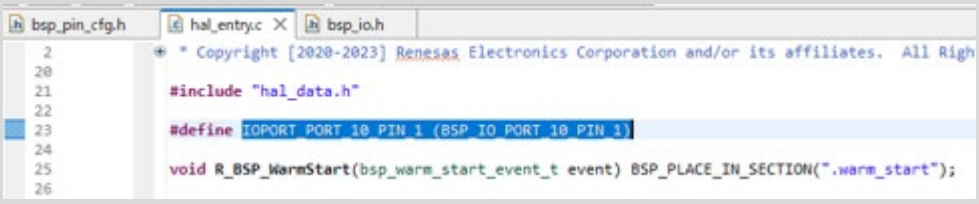
(Continued on next page)

No. 6 Resolved

Title	**Pin configuration error occurs in MPX-IO 16bit operating mode of “r_bsc”.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**FSP Configuration, Pins**
Description	Pin assignment is an optional specification in MPX-IO 16bit operation mode of “r_bsc”, but an error will occur if Input/Output A1-A13 is not set. 
Workaround	Please set “Custom” to “Operation Mode” of “r_bsc” in Pins tab when you use MPX-IO 16bit operation mode of “r_bsc”.

(Continued on next page)

No. 7 Resolved

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

No. 8 Resolved

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

(Continued on next page)

No. 9 Resolved

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

No. 10 Resolved

Issue	**A warning occurs when building "r_gmac" module with the gcc compiler.**
Target	**RZ/T2M, RZ/T2L**
Category	**FSP Modules, Ethernet**

Description	The following warning occurs when building "r_gmac" module with the gcc compiler.
	`../rzt/fsp/src/r_gmac/r_gmac.c:2173:14: warning: the comparison will always evaluate as 'false' for the pointer operand in 'pp_phy_instance + (sizetype)(port * 12)' must not be NULL [-Waddress]`
	`2173 |  if (NULL == pp_phy_instance[port])`
	`|  ^~`

Workaround	Please ignore this warning.

No. 11 Resolved

Issue	**In FSP Documentation, there is incorrect description in. "API Reference > Modules > Ethernet PHY" page.**
Target	**RZ/T2M, RZ/N2L**
Category	**FSP Modules, Ethernet PHY**
Description	In the "API Reference > Modules > Ethernet PHY" page in FSP Documentation, the **default** column description of "Select PHYs to use" configuration is incorrect.
Workaround	When reading the incorrect description, please replace the reading of it with follows. [Error] ➢ config.driver.ether_phy.phy_lsi.default,config.driver.ether_phy.phy_lsi.0,config.driver.ether_phy.phy_lsi.1,config.driver.ether_phy.phy_lsi.2,config.driver.ether_phy.phy_lsi.3,config.driver.ether_phy.phy_lsi. [Correction] ➢ All check boxes are enabled.

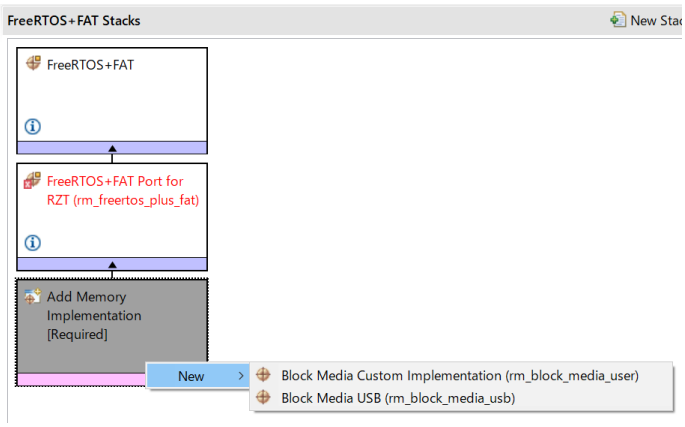
(Continued on next page)

No. 12

Issue	**The interrupt number cannot be successfully acquired by the R_FSP_CurrentIrqGet() when multiple interrupt occurs.**
Target	**RZ/N2L**
Category	**FSP Modules, FreeRTOS**
Description	The interrupt number cannot be successfully acquired by the R_FSP_CurrentIrqGet() when using multiple interrupt handlers with different priority levels in FreeRTOS.

Workaround	Please modify the followings for the countermeasure against nested interrupts.
	Target File: port.c
	`void vApplicationIRQHandler (uint32_t ulICCIAR)`
	`{`
	`#if 0`
	`  /* Re-enable interrupts. */`
	`  __asm(“cpsie i”);`
	`#endif`
	`  bsp_common_interrupt_handler(ulICCIAR);`
	`}`
	
	Target File: bsp_irq.c
	`void bsp_common_interrupt_handler (uint32_t id)`
	`{`
	`  uint16_t gic_intid;`
	`  /* Get interruot ID (GIC INTID). */`
	`  gic_intid = (uint16_t) (id & BSP_PRV_ID_MASK);`
	`#if VECTOR_DATA_IRQ_COUNT > 0`
	`  if (BSP_CORTEX_VECTOR_TABLE_ENTRIES <= gic_intid)`
	`  {`
	`  /* Remain the interrupt number */`
	`  g_current_interrupt_num[g_current_interrupt_pointer++] =`
	`  (uint16_t) (gic_intid-- BSP_CORTEX_VECTOR_TABLE_ENTRIES);`
	`  __asm volatile (“dmb”);`
	`#if 1`
	`  /* Enable nested interrupt. */`
	`  __asm volatile (“cpsie i”);`
	`  __asm volatile (“isb”);`
	`#endif`
	`  /* Branch to an interrupt handler. */`
	`  g_vector_table[(gic_intid-- BSP_CORTEX_VECTOR_TABLE_ENTRIES)]();`
	`  }`
	`  else`
	`#endif`
	`  {`
	`  /* Remain the interrupt number */`
	`  g_current_interrupt_num[g_current_interrupt_pointer++] = gic_intid;`
	`  __asm volatile (“dmb”);`
	`#if 1`
	`  /* Enable nested interrupt. */`
	`  __asm volatile (”cpsie i”);`
	`  __asm volatile (“isb”);`
	`#endif`
	`  /* Branch to an interrupt handler. */`
	`  g_sgi_ppi_vector_table[gic_intid]();`
	`  }`
	`#if 1`
	`  /* Disable nested interrupt. */`
	`  __asm volatile (”cpsid i”);`
	`  __asm volatile (”isb”);`
	`#endif`
	`  g_current_interrupt_pointer--;`
	`}`

No. 13

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

No. 14

Issue	The second argument of “r_mtu3” APIs do not match with common API.
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**FSP Modules, MTU3**
Description	The second argument of these three APIs "R_MTU3_PeriodSet()", "R_MTU3_InfoGet()", and "R_MTU3_StatusGet()" of the "r_mtu3" module, do not match with the API in “r_timer api.h” header file

Workaround	You cannot call these API by using function pointer
	`g_timer0.p_api->periodSet()`
	`g_timer0.p_api-> InfoGet()`
	`g_timer0.p_api-> StatusGet()`
	Please use API by calling them directly
	`R_MTU3_PeriodSet()`
	`R_MTU3_InfoGet()`
	`R_MTU3_StatusGet()`
	
	For reference how to use these APIs, please refer to **MTU3 Examples** in FSP documentation.

No. 15

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

Workaround	Please use “r_gpt” module ONLY with either CPU0 or CPU1 in multiprocessing.

No. 16

Issue	Project build error occur when 32-bit bus NOR flash and xSPI0 x8 boot modes are selected on RZT Custom User Board.
Target	**RZ/T2M**
Category	FSP Configuration, BSP
Description	When the following boards (boot mode) are selected, the required definitions are not generated and a build error occurs. • RZT Custom User Board (32-bit bus NOR flash boot mode) • RZT Custom User Board (xSPI0 x8 boot mode)
Workaround	Please don’t select 32-bit bus NOR flash and xSPI0 x8 boot modes on RZT Custom User Board.

No. 17

Issue	The secondary project for multiprocessing cannot be created when xSPI1 x1 boot modes are selected on RZT Custom User Board.
Target	**RZ/T2M**
Category	FSP Configuration, BSP
Description	When the following boards (boot mode) are selected for the primary project of multiprocessing, a variable required for multiprocessing is not defined and the secondary project cannot be created. • RZT Custom User Board (xSPI1 x1 boot mode)
Workaround	Please don’t select xSPI1 x1 boot modes on RZT Custom User Board when multiprocessing.

Appendix. Tool Software Limitations

This section describes the limitations regarding the tool software (e² studio, FSP SC) to create and debug FSP projects.

Table 7 List of Tool Software Limitations

No.	Title	Target Device			Category
		T2M	T2L	N2L	
1	When installing, please install into the default installation folder specified by installer.	✓	✓	✓	SC, FSP SC
2	Before pressing the reset button on the board, disconnect the e² studio connection first.	✓	✓	✓	e² studio
3	An error has occurred because the program download to the NOR flash area has failed. The download is successful on the second connection.	✓		✓	e² studio
4	The user program cannot be stopped immediately after the device boot process.	✓	✓	✓	e² studio
5	When using e² studio installer, if checking the multiple check boxes such as “View Release Notes” and so on to show information on browser, the ONLY head item of checked items is shown.	✓	✓	✓	e² studio
6	The Memory Region Usage of ATCM displayed in the Memory Usage window of e² studio is smaller than the actual size by memory region usage of DUMMY.	✓	✓	✓	e² studio
7	When debugging RAM execution without flash memory project with program written to flash memory, erase flash memory before debugging.	✓	✓	✓	e² studio
8	Applying RZ/T2 FSP v.1.2.0 pack to a project that is already working with RZ/T2M FSP v.1.1.0 causes an error when connecting the debugger.	✓			IAR EWARM
9	The Device Memory Usage of CPU1 in the Memory Usage window does not work properly.	✓			e² studio
10	When adding the CallbackSet function using the Developer Assistance feature, the second argument needs to be changed.	✓	✓	✓	e² studio, SC

No. 1

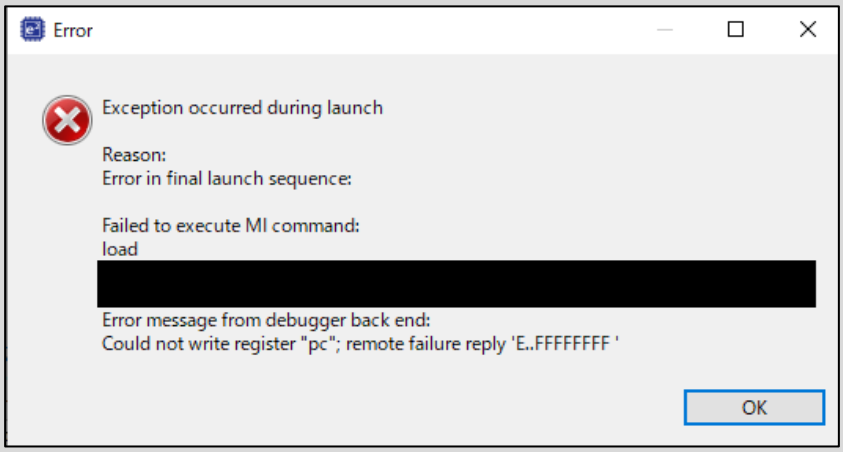
Limitation	**When installing, please install into the default installation folder specified by installer.**
Target Device	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**Smart Configurator, FSP Smart Configurator**
Description	When sharing a project between different PCs, build errors will occur if the installation folders are different.

No. 2 Resolved

Limitation	**Before pressing the reset button on the board, disconnect the e² studio connection first.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**e² studio**
Description	If the reset button is pressed on the board while connected with e² studio, debugging will not be able to continue.

(Continued on next page)

No. 3 Resolved

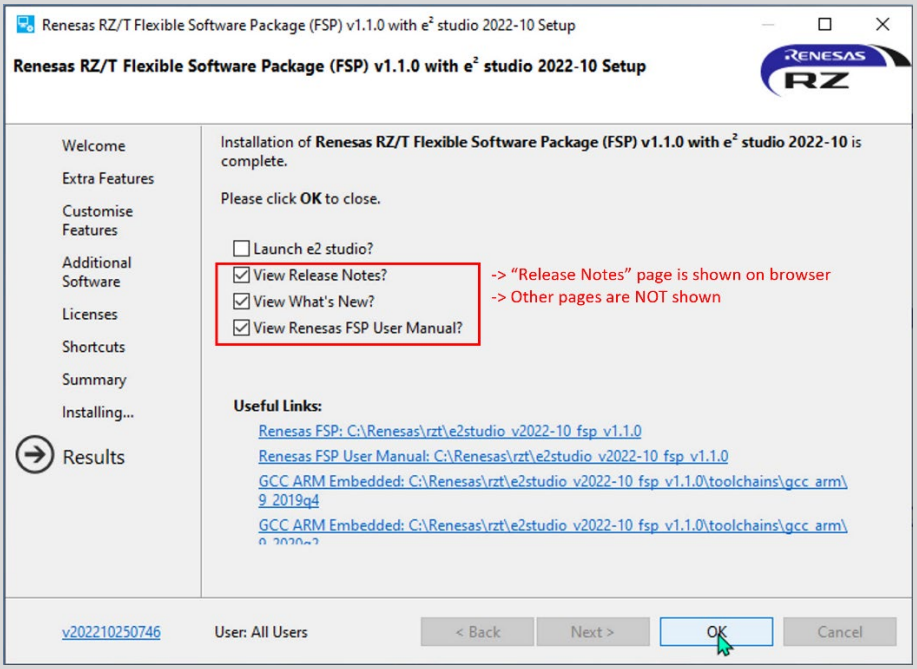
Limitation	**An error has occurred because the program download to the NOR flash area has failed. The download is successful on the second connection.**
Target	**RZ/T2M, RZ/N2L**
Category	**e² studio**
Description	If the following error is displayed when connecting the debugger or when downloading the program, click the [OK] button to close the dialog and try connecting again. 

No. 4

Limitation	**The user program cannot be stopped immediately after the device boot process.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**e² studio**
Description	Immediately after the device boot process (boot code), the program cannot be stopped at the beginning of the user program (loader program). When debugging, please follows the guide in Appendix. How to Debug FSP Project with Flash Boot Mode**.**

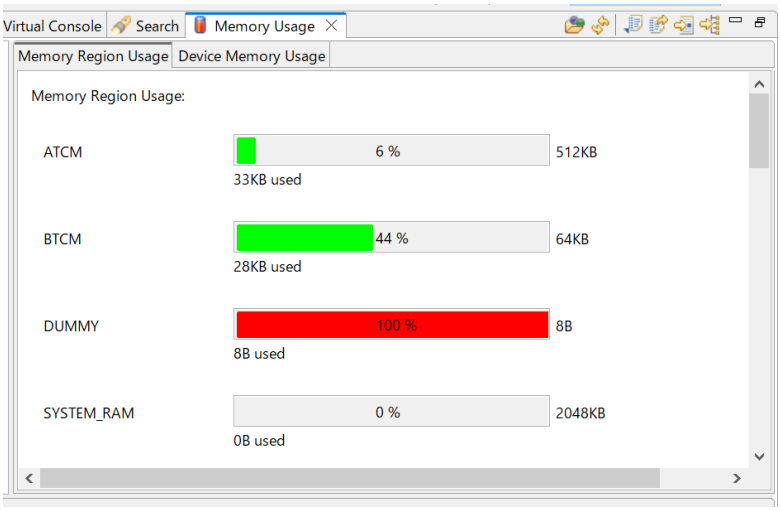
(Continued on next page)

No. 5 Invalid

Limitation	**When using e² studio installer, if checking the multiple check boxes such as "View Release Notes" and so on to show information on browser, the ONLY head item of checked items is shown.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**e² studio**
Description	For example, if checking "View Release Notes" check box and other check boxes on the following window, the ONLY "Release Notes" is shown, and the other contents are NOT shown. 

(Continued on next page)

No. 6

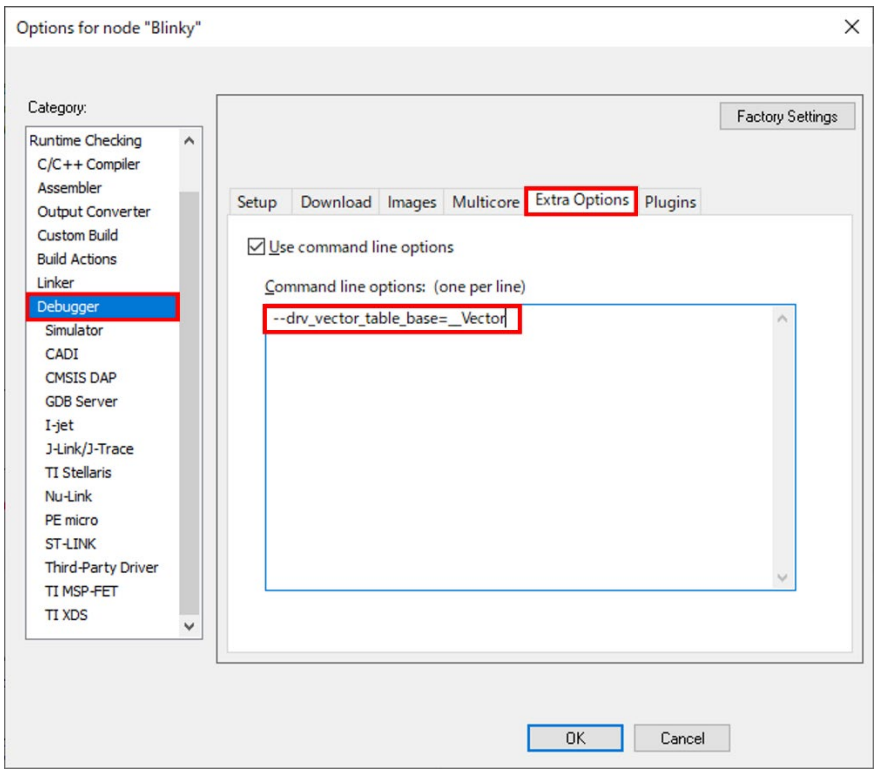
Limitation	**The Memory Region Usage of ATCM displayed in the Memory Usage window of e² studio is smaller than the actual size by Memory Region Usage of DUMMY.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**e² studio**
Description	The Memory Region Usage of DUMMY shown in the Memory Usage window is the region used by the system. The DUMMY is placed in ATCM, however Memory Region Usage of ATCM does NOT include its size. Therefore, please note that the Memory Region Usage of ATCM displayed is smaller than the actual size by the Memory Region Usage of DUMMY. 

No. 7

Limitation	**When debugging RAM execution without flash memory project with program written to flash memory, erase flash memory before debugging.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**e² studio**
Description	If you run a RAM execution without flash memory project with a program written in flash memory, it may be impossible to debug the project. When erasing flash memory, please follows the guide in Appendix. How to Erase Flash Memory

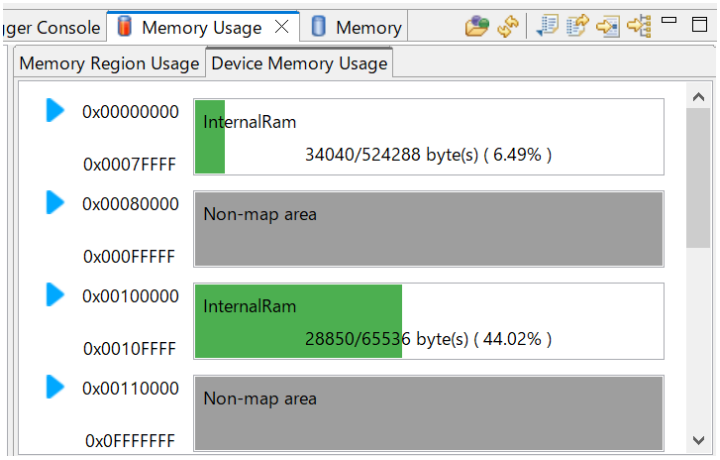
(Continued on next page)

No. 8

Limitation	**Applying RZ/T2 FSP v.1.2.0 pack to a project that is already working with RZ/T2M FSP v.1.1.0 causes an error when connecting the debugger.**
Target	**RZ/T2M**
Category	**IAR EWARM**
Description	An error occurs when connecting to the debugger because the function name of vector table was changed in RZ/T2 FSP v.1.2.0. Change the following command in the **"command line options (one per line)"** to • (Before change) **–drv_vector_table_base=vector_table** • (After change) **–drv_vector_table_base=__Vector** 

(Continued on next page)

No. 9

Limitation	**The Device Memory Usage of CPU1 in the Memory Usage window does not work properly.**
Target	**RZ/T2M**
Category	**e² studio**
Description	The Device Memory Usage in the Memory Usage window, it cannot distinguish between CPU0 and CPU1. When debugging CPU1, the memory area available for CPU1 should be displayed, but the memory area available for CPU0 is incorrectly displayed. 

No. 10

Limitation	**When adding the CallbackSet function using the Developer Assistance feature, the second argument needs to be changed.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**e² studio, Smart Configurator**

Description	When adding the R_xxx_CallbackSet() function (xxx means any module name) using the Developer Assistance feature, the second argument does not have the correct value. Please replace the second argument with "p_callback".
	
	An example of SCI_SPI module, adding CallbackSet() using the Developer Assistance results in the following.
	`status = R_SCI_SPI_CallbackSet(&g_spi0_ctrl, spi_callback_args_t, p_context, p_callback_memory);`
	It needs to replace the second argument with “p_callback”.
	`status = R_SCI_SPI_CallbackSet(&g_spi0_ctrl, p_callback, p_context, p_callback_memory);`

(Continued on next page)

Appendix. How to Debug FSP Project with Flash Boot Mode

When debugging FSP project with flash boot mode (xSPI0 boot, NOR flash boot), the program cannot be stopped at the beginning of the user program (loader program). Another note on the use of flash debug boot mode is also included in this section.

Please note the following point depending on your IDE (e^2 studio or IAR EWARM) to debug the user program from its beginning.

1. **(Both e^2 studio and IAR EWARM)** Insert the loop part in startup.c.

When debugging is started, the debugger stops the user program (loader program) about 100ms after the device boot process (boot code). If using e^2 studio 2022-10 or later, the PC (program counter) is replaced at the entry point (first line in **system_init()** function) after the debugger stops, otherwise, the PC points the address of somewhere in the user program.

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

- /rzt/fsp/src/bsp/cmsis/Device/RENESAS/Source/cr/startup_core.c, or
- /rzn/fsp/src/bsp/cmsis/Device/RENESAS/Source/startup.c

The detailed position, at which the loop part should be inserted, depends on the IDE(Debugger) and Boot mode.

<table>
<tr><th>IDE</th><th>Boot Mode</th><th>Position at which the loop part should be inserted.</th></tr>
<tr><td rowspan="2">E^2 studio
2022-04
2022-07</td><td>xSPI0 boot</td><td>Line after static_constructor_init in mpu_cache_init() function.

```
#if BSP_CFG_C_RUNTIME_INIT
    /* Initialize static constructors */
    __asm volatile (
        "static_constructor_init:                          \n"
        "    ldr r0, =bsp_static_constructor_init          \n"
        "    blx r0                                        \n"
        );
#endif

#if 1 // Software loops are only needed when debugging.
    __asm volatile (
        "    mov   r0, #0                                  \n"
        "    movw  r1, #0xf07f                             \n"
        "    movt  r1, #0x2fa                              \n"
        "software_loop:                                    \n"
        "    adds  r0, #1                                  \n"
        "    cmp   r0, r1                                  \n"
        "    bne   software_loop                           \n"
        ::: "memory");
#endif
```

</td></tr>
<tr><td>NOR flash boot</td><td>First line in system_init() function.

```
BSP_TARGET_ARM BSP_ATTRIBUTE_STACKLESS void system_init (void)
{
#if 1 // Software loops are only needed when debugging.
    __asm volatile (
        "   mov   r0, #0                          \n"
        "   movw  r1, #0xf07f                     \n"
        "   movt  r1, #0x2fa                      \n"
        "software_loop:                           \n"
        "   adds  r0, #1                          \n"
        "   cmp   r0, r1                          \n"
        "   bne   software_loop                   \n"
        ::: "memory");
#endif
    __asm volatile (
        "set_hactlr:                              \n"
        "   MOVW  r0, %[bsp_hactlr_bit_l]         \n" /* Set HACTLR bits(L) */
        "   MOVT  r0, #0                          \n"
        "   MCR   p15, #4, r0, c1, c0, #1         \n" /* Write r0 to HACTLR */
        ::[bsp_hactlr_bit_l] "i" (BSP_HACTLR_BIT_L) : "memory");
```

</td></tr>
<tr><td>e^2 studio
2022-10
or later

IAR
EWARM</td><td>xSPI0 boot</td></tr>
<tr><td>NOR flash boot</td></tr>
</table>

Appendix. How to Erase Flash Memory

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

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

1. e^2 studio

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

   i) Set the switch for boot mode on RSK to correspond to the area to be erased.

   ii) Open the J-Link Commander.

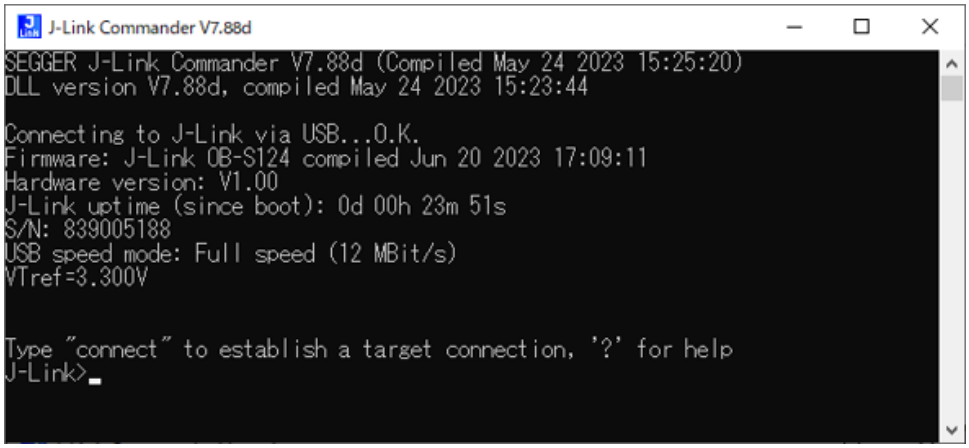


Figure 88 : Launch J-Link Commander

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

Next, specify the connection conditions as follows.

- Device> (Device type name)

Table 8 Device type name on Renesas Starter Kit+

RSK	Device type name
RSK + RZ/T2M	R9A07G075M24_CPU0
RSK + RZ/T2L	R9A07G074M04
RSK + RZ/N2L	R9A07G084M04

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

(Continued on next page)

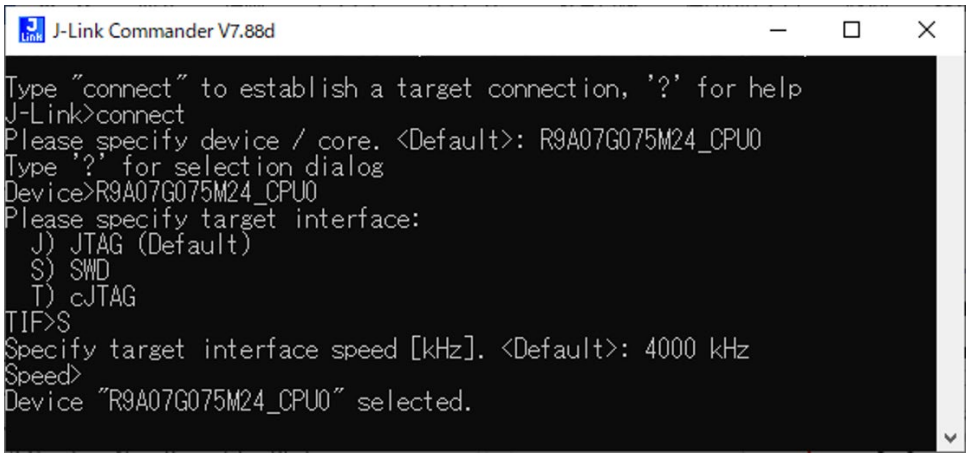


Figure 89 : Initial setup for connecting to the device

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

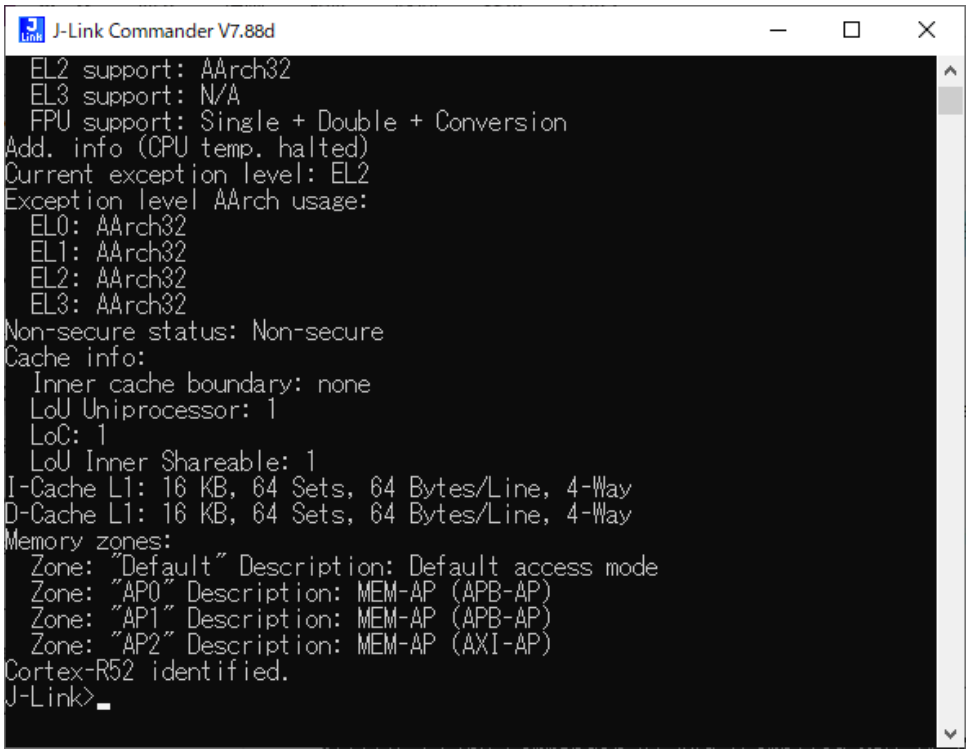


Figure 90 : Message of device core identification

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

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

(Continued on next page)

Table 9 External address space to be used in each boot mode

RSK	Boot mode	External address space to be used	Start address	End address
RSK + RZ/T2M	xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
	16-bit bus	CS0	0x70000000	0x71FFFFFF
RSK + RZ/T2L	xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
	xSPI1 x1	xSPI1 CS0	0x68000000	0x68FFFFFF
RSK + RZ/N2L	xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
	16-bit bus	CS0	0x70000000	0x71FFFFFF

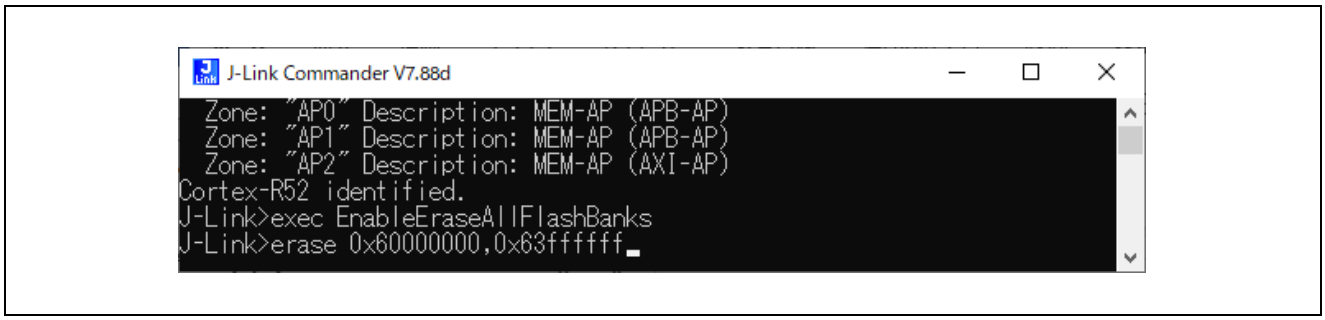


Figure 91 : Specify erase range

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

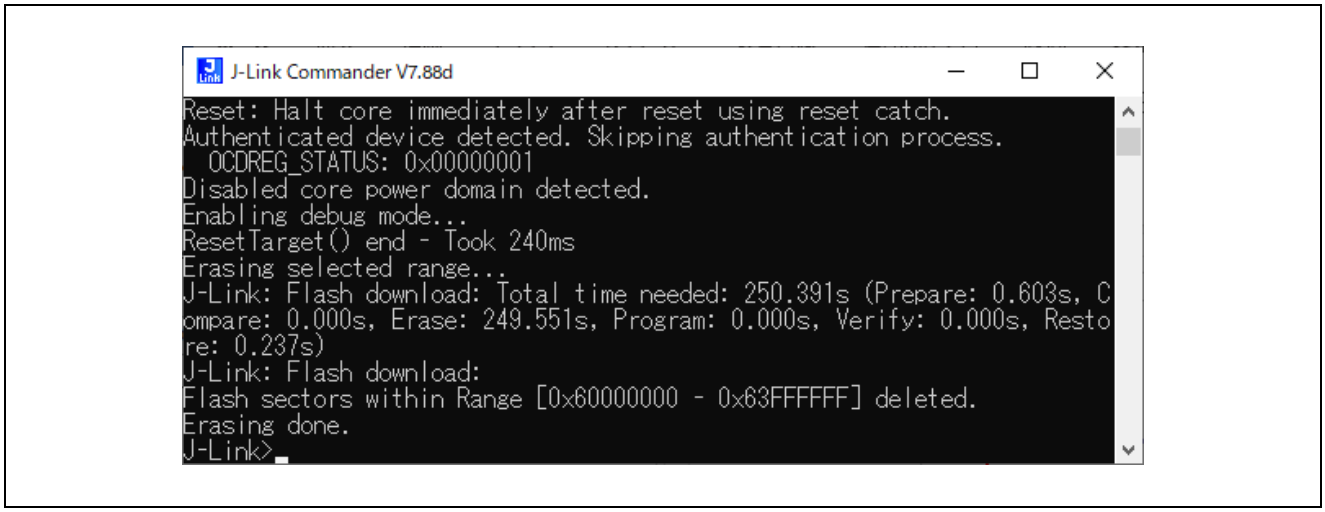


Figure 92: Message of flash memory erase complete

v) Enter “q” to exit J-Link Commander.

(Continued on next page)

2. IAR EWARM

If you want to erase the flash memory on RSK using IAR EWARM, execute the following steps.

i) Set the switch for boot mode on RSK to correspond to the area to be erased.

ii) Open the workspace of a project.

xxx.eww

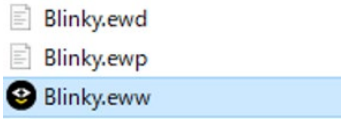


Figure 93 : Open workspace for IAR EWARM

iii) Select “Project” -> “Download” -> “Erase memory”.

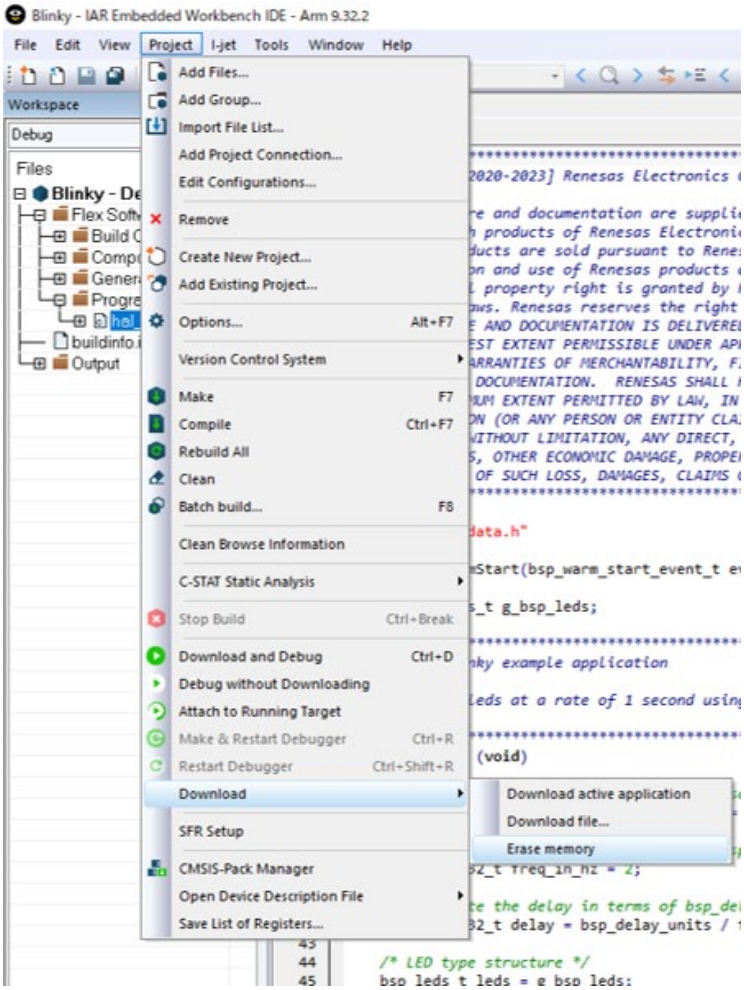


Figure 94 : Select erase memory command

(Continued on next page)

iv) Select erase memory space.

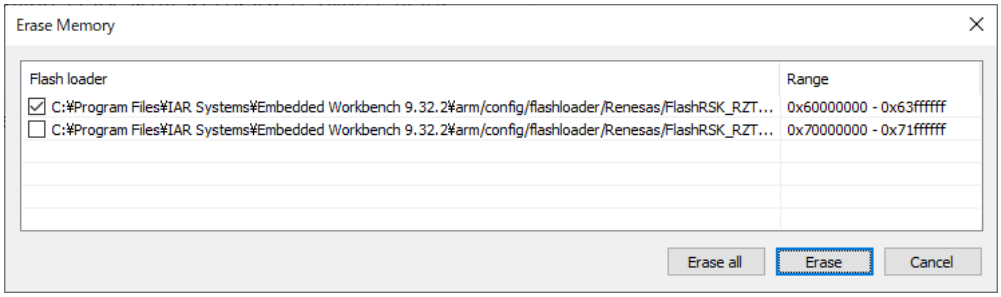


Figure 95 : Select erase memory space

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

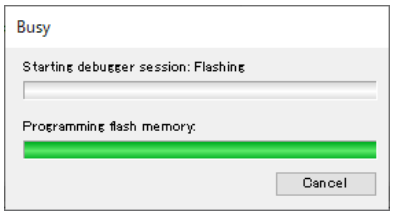


Figure 96 : Screen during erasing

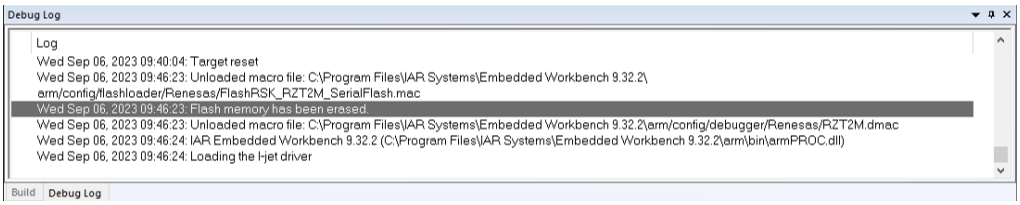


Figure 97 : Message of flash memory erase complete

Appendix. How to Change Boot Mode of 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 specific cases in which re-configure is necessary is when a RAM execution without flash memory project is changed to flash boot mode (xSPI0 x1 boot mode and others).

Please change the boot mode by following steps.

1. If the FSP version of your project is earlier than FSP v1.3.0, change it to FSP v1.3.0 or later.
2. Rename and save the current Pin Configuration in the **Pins** tab.
   - How to rename Pin Configuration: Click “Manage Configurations”…"

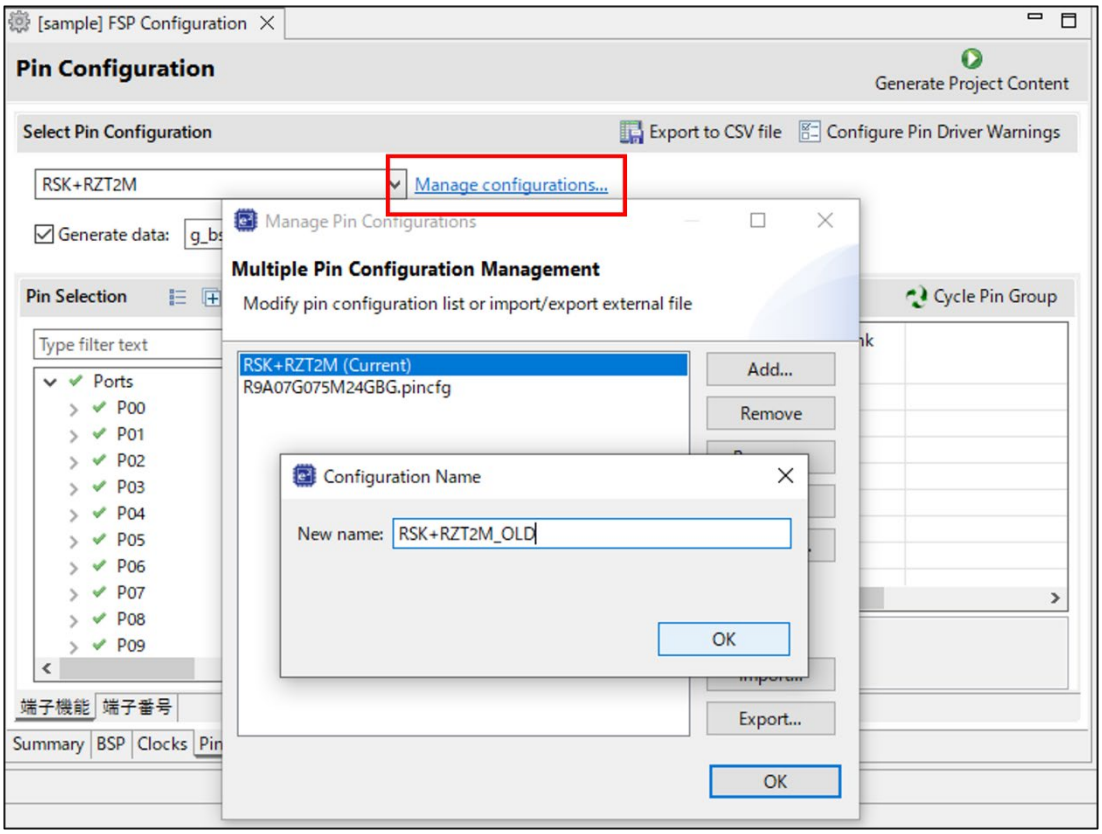


Figure 98: How to rename Pin Configuration

(Continued on next page)

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

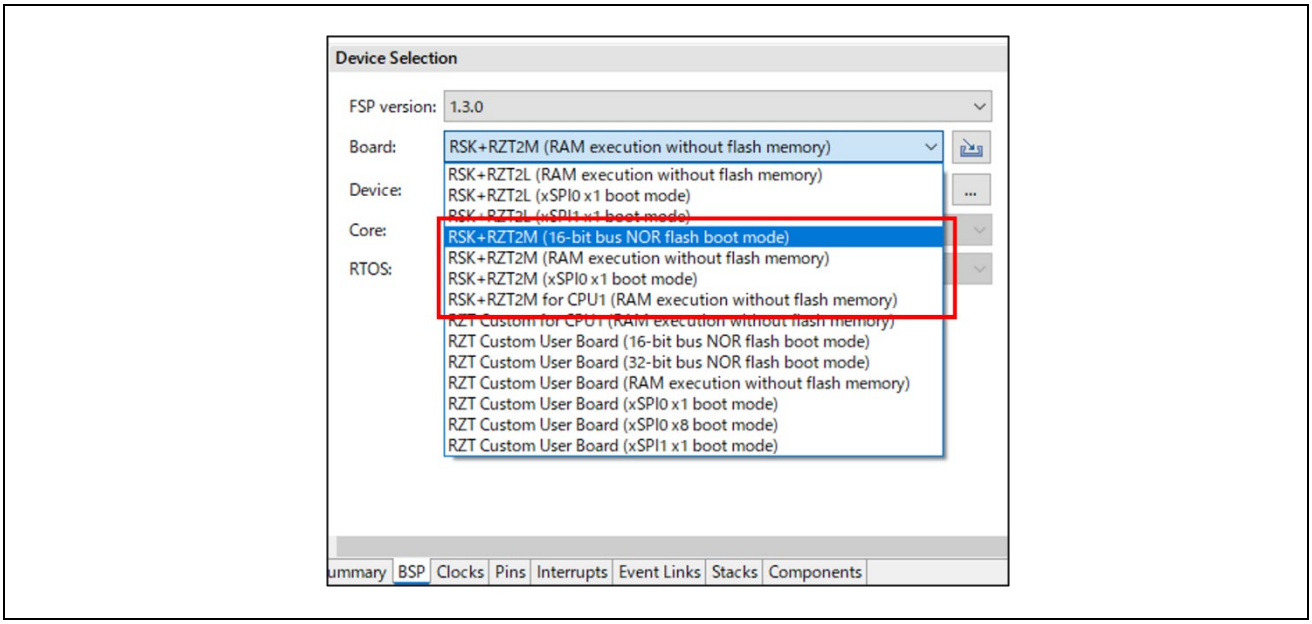


Figure 99: Change the boot mode in the BSP tab

4. Reselect “FSP Version” from the drop-down list.

(This operation is necessary even if there is only one version in the list.)

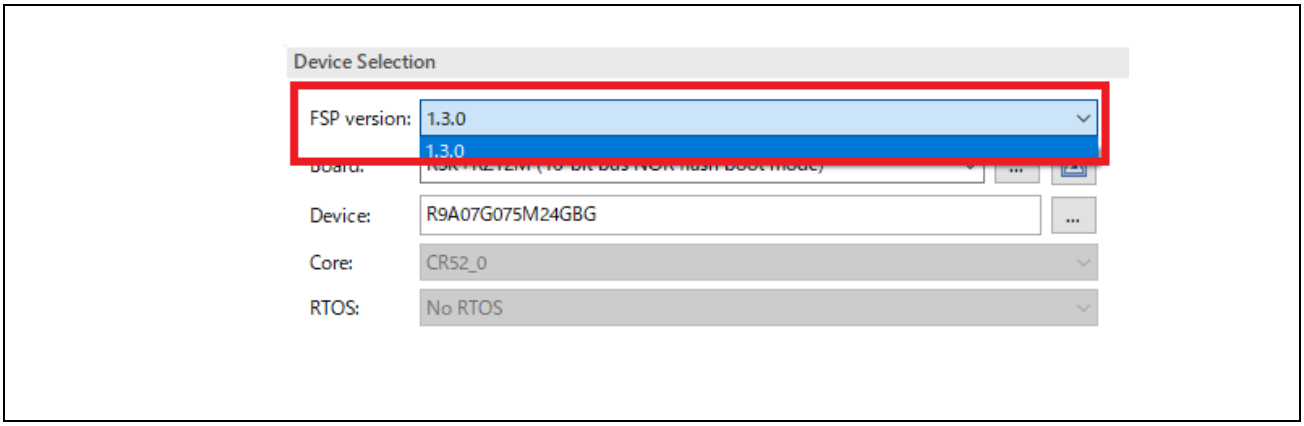


Figure 100: Reselect “FSP Version” from the drop-down list

(Continued on next page)

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

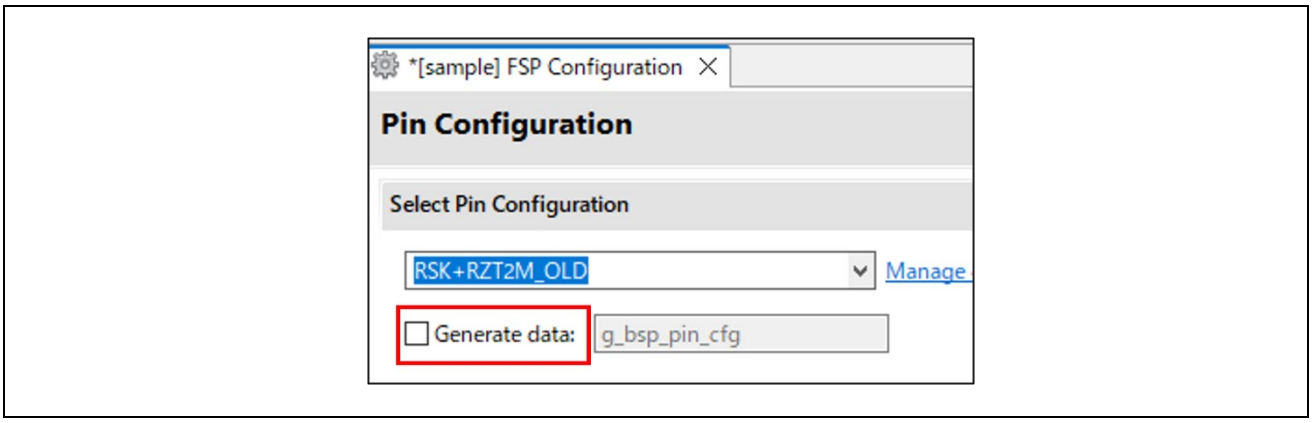


Figure 101: Uncheck Generate data in the Pins tab

6. Select the regenerated configuration for the board.

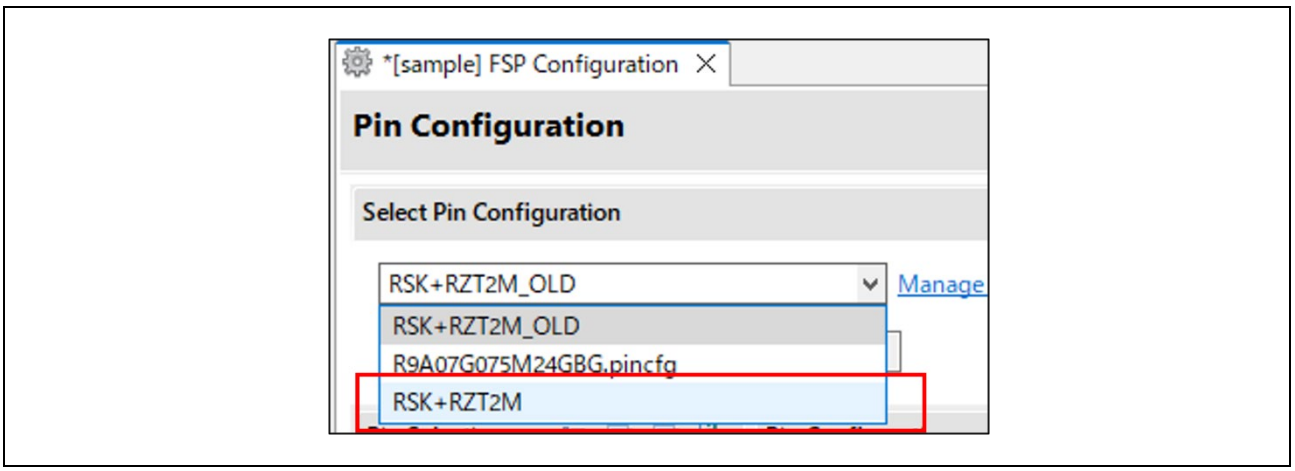


Figure 102: Select the regenerated configuration for the board

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

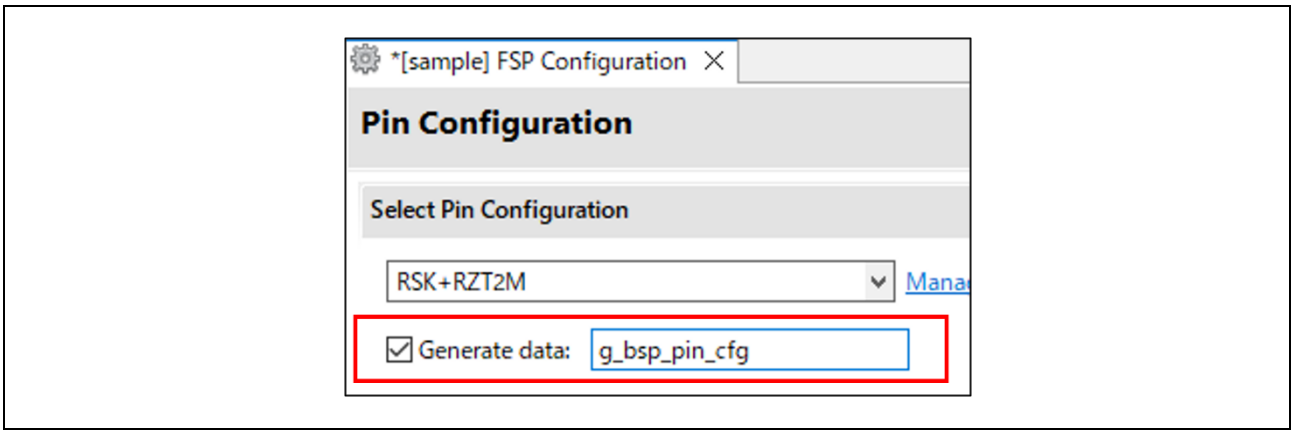


Figure 103: Check Generate data again

Appendix. How to Debug FSP multiprocessing projects with Flash Boot Mode

In the case of multi-processing with flash boot mode(xSPI0 boot, NOR flash boot), the debugging procedure is different from RAM execution without flash memory described in 4.7 Debug and Run for Multiprocessing and 5.3.5 Debug for Multiprocessing. Please create and debug FSP multiprocessing projects with flash boot mode by following steps.

1. e^2 studio
   i) Create projects according to the procedures in 4.3 Create a New Project for Blinky. However, select the flash boot mode to be used in the No. 6 procedure.
      - RZ/T series
        - **RSK+RZT2M (16-bit bus NOR flash boot mode)** or **RSK+RZT2M (xSPI0 x1 boot mode)**

      Insert the loop part in startup_core.c with reference to Appendix. How to Debug FSP Project with Flash Boot Mode
   ii) Build the projects using the following steps.
      a. Create and build the primary project. (1st build of the primary project)
         No build setting is required, proceed to 4.4.1 Build.
      b. Create the secondary project. Change the project property setting and build it.
         Set the following before building:
         b-1. Click **Project** > **Properties**.

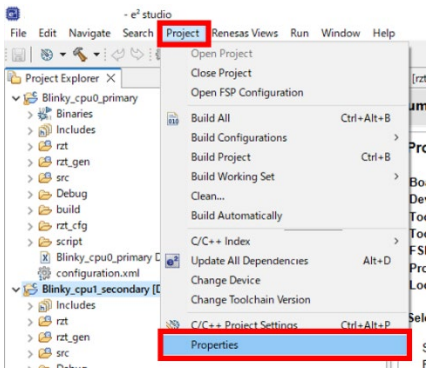


Figure 104 e^2 studio Project Properties

(Continued on next page)

b-2. Click **C/C++ Build** > **Settings** > **Build Steps**.

b-3. Add command at **Post-build steps**.

arm-none-eabi-objcopy -I elf32-littlearm -O binary ${ProjName}.elf secondary.bin && arm-none-eabi-objcopy -I binary -O elf32-littlearm -B arm --rename-section .data=.secondary,alloc,data,readonly,load,contents secondary.bin secondary.o

b-4. Proceed to 4.4.1 Build.

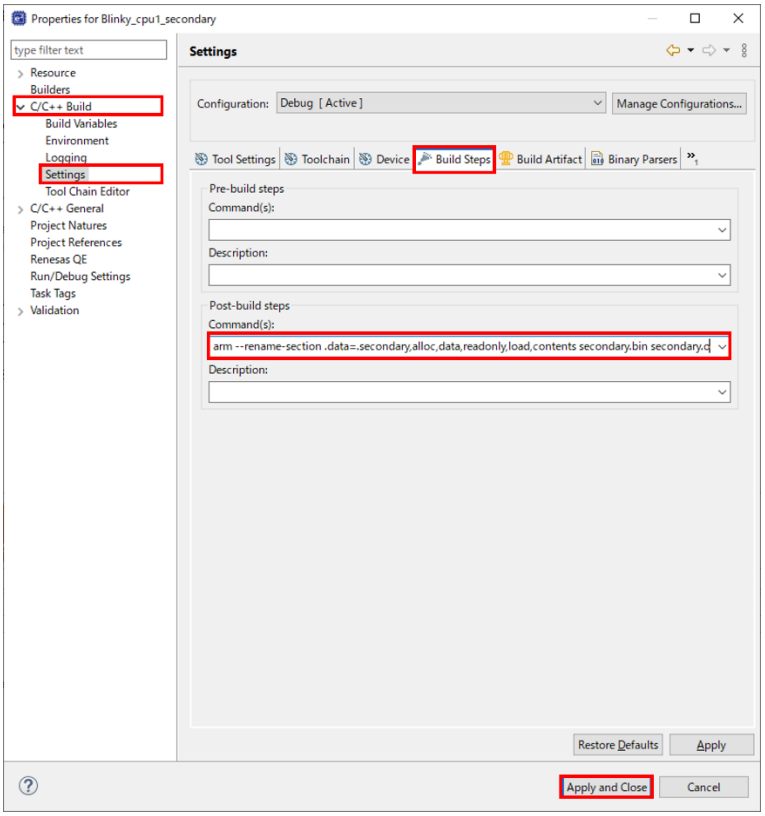


Figure 105 e² studio Build Setting for the Secondary Project

(Continued on next page)

c. Change the project property setting of the primary project and build it. (2nd build of the primary project)

Set the following before building:

c-1. Click **Project** > **Properties**.

c-2. Click **C/C++ Build** > **Settings** > **Tool Settings > Cross ARM C Linker > Miscellaneous**.

c-3. Set a file pass of secondary.o in the secondary project to **Other objects**.

${workspace_loc:/Blinky_cpu1_secondary/Debug/secondary.o}

c-4. Click **Apply and Close**.

d. Proceed to 4.4.1 Build.

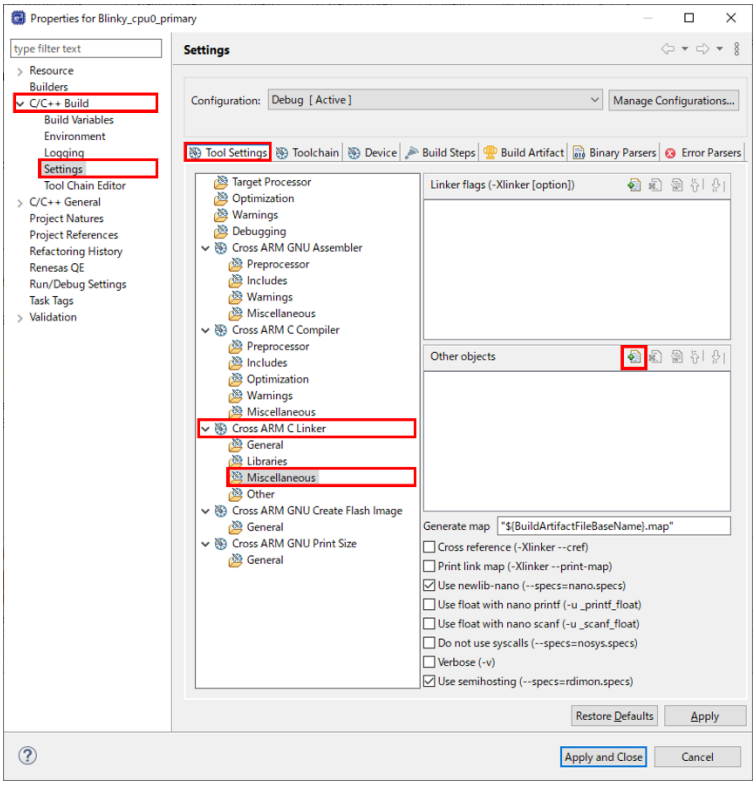


Figure 106 e² studio Build Setting for the Primary Project (Part 1)

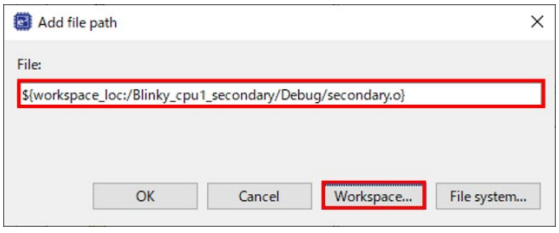


Figure 107 e² studio Build Setting for the Primary Project (Part 2)

iii) Debug the projects according to the procedures in 4.7 Debug and Run for Multiprocessing.
The flash boot mode differs from RAM execution without flash memory in two points.

- Both the primary and secondary project binaries are downloaded to the device when connecting debugger with the primary project.
- No need to change Debug Configuration from the default settings in the No. 3 procedure in 4.5.2 Debug Steps.
  - ➢ **Debugger > Connection Settings > Connection**
    - **Reset after download: Yes**
    - **Set CPSR(5bit) after download: No**

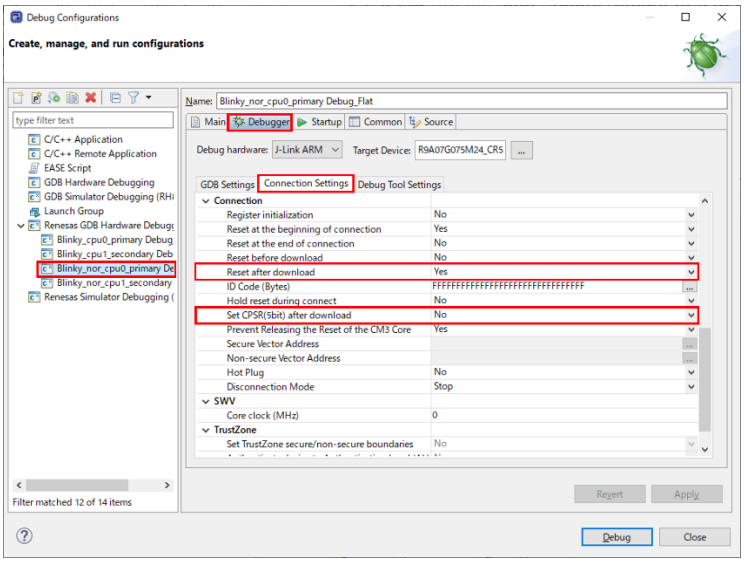


Figure 108 e² studio Debug Configuration Settings for the project in flash boot mode

iv) When changing the project and debugging it again, follow these steps.
a. Build the primary project.
b. Build the secondary project.
c. Build the primary project again.
d. Debug the projects as the procedure iii.

2. IAR EWARM

i) Create projects according to the procedures in 5.3.2 Create a New Project. However, select the flash boot mode to be used in the No. 4 procedure.

- RZ/T series
  - ➢ **RSK+RZT2M (16-bit bus NOR flash boot mode)** or **RSK+RZT2M (xSPI0 x1 boot mode)**

Insert the loop part in startup_core.c with reference to Appendix. How to Debug FSP Project with Flash Boot Mode

(Continued on next page)

ii) Build the projects by the following order.

a. Create and build the primary project. (1st build of the primary project)
No setting is required, proceed to 5.3.3.2 Build.

b. Create the secondary project. Change the project options setting and build it.
Set the following before building:

b-1. Click **Project** > **Options…**.

b-2. Click **Debugger** > **Setup**.

b-3. Uncheck “Run to".

b-4. Click **Output Converter** > **Output**.

b-5. Set **Raw binary** to **Output format**.

b-6. proceed to 5.3.3.2 Build.

b-7. Close the secondary project.

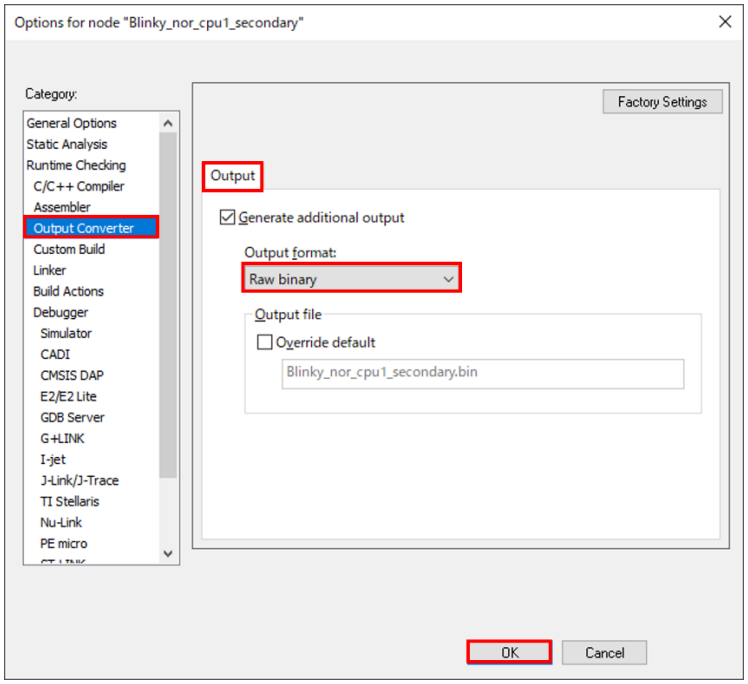


Figure 109 IAR EWARM project options for the secondary project in flash boot mode

(Continued on next page)

c. Change the project options setting of the primary project and build it. (2nd build of the primary project)

Set the following before building:

c-1. Click **Project** > **Options…**.

c-2. Click **Debugger** > **Setup**.

c-3. Uncheck “Run to".

c-4. Click **Linker** > **Input**.

c-5. Set the following contents:

- Keep symbols: (one per line)
  - SECONDARY
- Raw binary image
  - File: the path of binary file in the secondary project.

    e.g. $PROJ_DIR$\..\Blinky_nor_cpu1_secondary\Debug\Exe\Blinky_nor_cpu1_secondary.bin
  - Symbol: SECONDARY
  - Section: SECONDARY
  - Align: 4

c-6. Proceed to 5.3.3.2 Build.

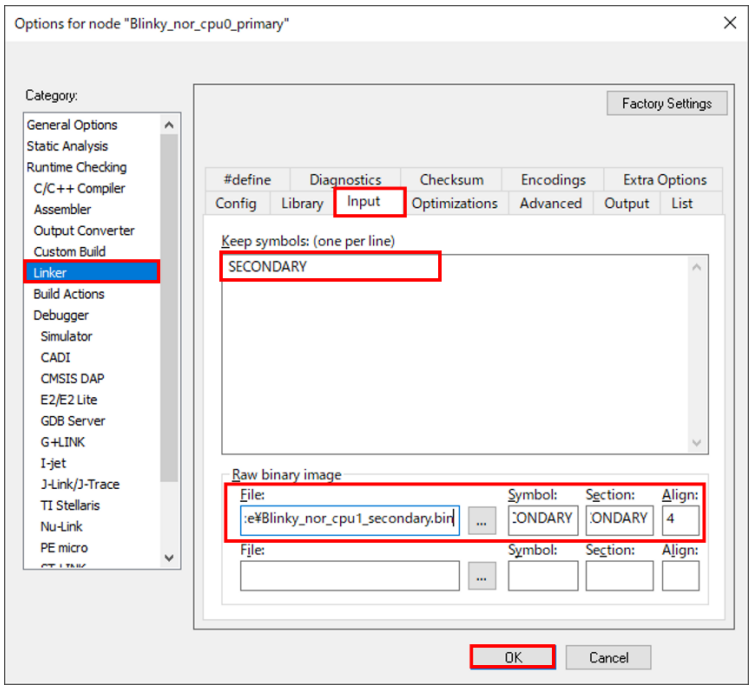


Figure 110 IAR EWARM project options for the primary project in flash boot mode

(Continued on next page)

iii) Debug the projects by the following order.

a. Open the primary project and close the secondary project on IAR EWARM.

b. Click **Project** > **Download** > **Download file…**.

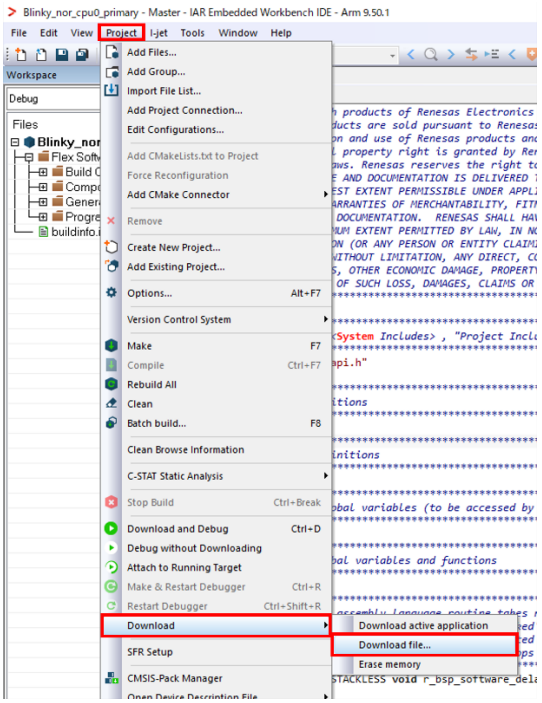


Figure 111 IAR EWARM Download File

c. Select out file of the primary project.
e.g. $PROJ_DIR$\..\Blinky_nor_cpu0_primary\Debug\Exe\Blinky_nor_cpu0_primary.out

d. Change the project options setting of the primary project.
d-1. Click **Project** > **Options…**.
d-2. Click **Debugger** > **Multicore**.
d-3. Set the following contents in Asymmetric multicore.

- Simple
  - Partner workspace: the eww file path of the secondary project.
    e.g. $PROJ_DIR$\..\Blinky_nor_cpu1_secondary\Blinky_nor_cpu1_secondary.eww
  - Partner project: Blinky_nor_cpu1_secondary
  - Partner configuration: Debug

(Continued on next page)

e. Click **Project** > **Debug without Downloading** of the primary project.

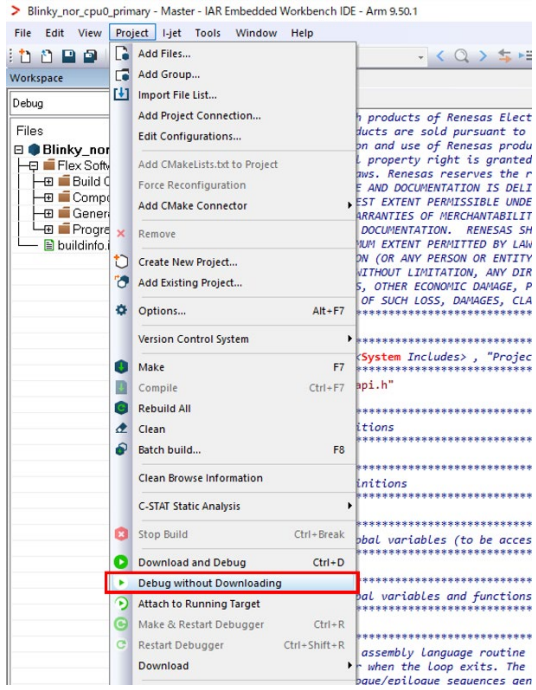


Figure 112 IAR EWARM Debug without Downloading

f. The secondary project is automatically launched.

g. Run the program of primary project. If the LED0 and LED1 are blinking, proceed to the next step.

h. The primary project in operation, run the program of secondary project.

i. When exiting Debug and disconnect from the debugger, if debugging is stopped in one of the projects, either the primary or the secondary, the other will automatically stop as well.

iv) When changing the project and debugging it again, follow these steps.

a. Disable asymmetric multicore setting.

a-1. Click **Project** > **Options…**.

a-2. Click **Debugger** > **Multicore**.

a-3. Select **Disable** in Asymmetric multicore.

b. Build the primary project.

c. Build the secondary project.

d. Build the primary project again.

e. Debug the projects as the procedure iii.

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 target device.
		All	Unified some terminologies.
		p.9	Updated “SEGGER J-Link” section. • Added the software environment on which FSP projects are verified.
		p.13	Added the “2.5.1 RSK+RZN2L” section.
		p.19	Updated “e^2 studio Prerequisites” • Updated the Windows PC requirements.
		p.37	Update “Prerequisites” section • Added the note regarding the patch for debugging RZ/N2L FSP project on EWARM.
		p.38	Updated “Create a New Project” section. • Added installation path of FSP SC. • Added some steps for creating a EWARM project. • Added Note subsection for debugging RZ/N2L EWARM project.
		p.58	Updated “Selecting a Board and Toolchain” section. • Added the detailed explanation how to select Board and Device for creating a FSP project.
		p.73	Updated “Appendix. Known Issues” chapter.
		p.82	Updated “Appendix. Tool Software Limitations” section.
		-	Added “Appendix. How to update J-Link DLL files in e^2 studio” chapter.
		p.88	Added “Appendix. How to Debug FSP Project with Flash Boot Mode”
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.9	Updated “2.3.1 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 RZ/T2M FSP project. • Added the link to Renesas Knowledge Base which explains how to update J-Link DLL in e^2 studio.
		p.45	Added “5.3.2.2 NOTE: Configure IAR EWARM Project [RZ/T2M, RZ/T2L]” section.
		p.73	Updated “Appendix. Known Issues” chapter. • Remove some limitations regarding RZ/T2M

Rev.	Date	Description	
		Page	**Summary**
		p.82	Updated “Appendix. Tool Software Limitations” section. • Added new limitation of e^2 studio regarding J-Link OB S124 firmware version. • Added the link to explain how to update J-Link DLL.
		-	Removed “Appendix. How to update J-Link DLL files in e^2 studio” 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 “Appendix. Known Issues” chapter. • Removed some limitations regarding RZ/N2L
		p.73	Updated “Appendix. Tool Software Limitations” section. Removed some limitations regarding RZ/N2L
		p.75	Updated “Appendix. How to Debug FSP Project with Flash Boot Mode” 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 “2.3.1 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 “2.4.1 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 “2.4.2 RSK+RZT2L” section
		p.16-18	Updated “2.5.1 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 “Appendix. Known Issues” section. • Added a table “List of Known Issues” • Numbered each issue. • Removed issue of adding "r_dsmif" alone • Updated issue contents that the BSP properties are sometimes configured to incorrect configuration • Removed Ethernet SELECTOR issue.
		p.74-77	Updated “Appendix. Tool Software Limitations” 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 “Appendix. How to Debug FSP Project with Flash Boot Mode” section • 1. (Both e^2 studio and EWARM) Insert the loop part in startup.c. Added e^2 studio 2023-01 to the table. • 3. (e2 studio 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 “2.3.1 SEGGER J-Link” section. • Updated the FSP version and e^2 studio 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 e^2 studio 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 “Appendix. Known Issues” chapter. • Updated table “List of Known Issues” to add new issues and add N2L as target device for No.2 • Added new issue related to BSP configuration when changing board setting. • Added new issue related to FSP module FreeRTOS issue.
		p.78	Updated “Appendix. Tool Software Limitations” chapter. • Removed some limitations regarding breakpoint
		p.82	Updated “Appendix. How to Debug FSP Project with Flash Boot Mode” • 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 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 blinky template code.
		p.68-73	Updated “Appendix. Known Issues” section. • Updated table “List of Known Issues” to add new issues. • Added new issues related to Pins configuration. • Added new issue of warning message when building “r_gmac” with gcc compiler.
		p.75	Updated “Appendix. Tool Software Limitations” chapter. • Added “Smart Configurator” section. • Added new limitation of displaying memory region usage
		p.80	Updated “Appendix. How to Debug FSP Project with Flash Boot Mode” section. • Removed limitation related to reset when using e^2 studio
		p.82-86	Added “Appendix. How to Erase Flash Memory” section.
1.07	Sep.29.23	All	Updated documentation for RZ/N2L FSP v1.3.0. • Removed contents for RZ/N2L FSP v1.2.0

Rev.	Date	Description	
		Page	Summary
		p.9	Updated “2.3.1 SEGGER J-Link” section. Updated the FSP version and e^2 studio version for RZ/N2L.
		p.80	Updated “Appendix. How to Debug FSP Project with Flash Boot Mode” section. • Change the number of items. • Removed limitation related to RZ/N2 FSP v1.2.0 and J-ink 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 “Appendix. 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 description of each Known Issues. • Grayed out items where issues have been resolved. • Added description to workaround of No. 3. • Added RZ/T2L as 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 “Appendix. Tool Software Limitations” chapter. • Added column “Category” to the List • Removed category headings (Smart Configurator, FSP Smart Configurator, e^2 studio, …) • Added item “Category” to 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 v1.3.0. • Removed contents for RZ/T2 FSP v2.0.0
		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 “2.3.1 SEGGER J-Link” section. • List Target Device separately for each series in a table.
		p. 10-18	Updated “2.4 RZ/T Series Board Setup” section and added “2.5 RZ/N Series Board Setup” section. • Each series was divided into separate explanatory chapters. • Delete unnecessary figure descriptions
		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.

Rev.	Date	Description	
		Page	**Summary**
		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. 32	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. 34	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. 44	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 Build” section.
		p. 43	Added “5.3.3.1 NOTE: Build settings [Only Multiprocessing]” section.
		p. 47	Added “5.3.3.2 Build” section.
		p. 48	Updated “5.3.4 Download & Debug the Project” section. • 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. 73	Updated “Appendix. 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. 84	Updated “Appendix. Tool Software Limitations” chapter. • Added new limitation No. 9 and No. 10.
		p. 99	Added “Appendix. How to Debug FSP multiprocessing projects with Flash Boot Mode” chapter.
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 “2.3.1 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.

Rev.	Date	Description	
		Page	Summary
		p.44-46	Updated “5.3.3.1 NOTE: Build settings [Only Multiprocessing]” section. • Changed the program execution start position setting. • Corrected reset setting. Added images for the settings screen.
		P.73	Updated “Appendix. Known Issues” chapter. • Resolved issues No. 2 • Added RZ/N2L as target device from No.13 and No.14 • Added new issues No. 16 and No. 17.
		p.84	Updated “Appendix. Tool Software Limitations” chapter. • Added RZ/N2L as target device from No. 10.
		p.97	Updated “Appendix. How to Change Boot Mode of 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.

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/.