RZ/T2, RZ/N2

Getting Started with Flexible Software Package

Introduction

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

Target Device

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

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

About the video contents

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

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

Contents

1. Introduction 5
1.1 Overview 5
1.2 Introduction to FSP 5
1.2.1 Purpose 5
1.2.2 e² studio IDE 5
1.2.3 FSP SC 5
1.2.4 FSP Documentation 5
1.3 Related Documentation Files 6
1.3.1 Evaluation Board User's Manual 6
1.3.2 FSP Documentation 6
1.4 Starting Development Introduction 7

2. Set up Evaluation Board 8
2.1 Obtaining an Evaluation Board 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
2.4.3 RSK+RZT2ME 15
2.4.4 RZ/T2H Evaluation Board 16
2.5 RZ/N Series Board Setup 19
2.5.1 RSK+RZN2L 19
2.5.2 RZ/N2H Evaluation Board 22

3. e² studio Setup 25
3.1 What is e² studio? 25
3.2 e² studio Prerequisites 25
3.2.1 Windows PC Requirements 25
3.2.2 Installing e² studio, Platform Installer and FSP Package 25
3.2.3 Choosing a Toolchain 25
3.2.4 Licensing 25

4. Tutorial: Your First RZ/T2, RZ/N2 MPU Project – Blinky 26
4.1 Tutorial Blinky 26
4.2 What Does Blinky Do? 26
4.3 Create a New Project for Blinky 26
4.3.1 Details about the Blinky Configuration 33

4.3.2 Configuring the Blinky Clocks....33
4.3.3 Configuring the Blinky Pins....33
4.3.4 Configuring the Parameters for Blinky Components....33
4.3.5 Where is main()?....33
4.3.6 Blinky Example Code....33
4.4 Build the Blinky Project....34
4.4.1 Build....34
4.4.2 Build for Multiprocessing....34
4.5 Debug the Blinky Project....35
4.5.1 Debug Prerequisites....35
4.5.2 Debug Steps....35
4.5.3 Details about the Debug Process....40
4.6 Run the Blinky Project....40
4.7 Debug and Run for Multiprocessing....41
4.8 Import the Project....42

5. FSP SC User Guide....46
5.1 What is FSP SC?....46
5.2 Tutorial Blinky....46
5.3 Using FSP SC with IAR EWARM....47
5.3.1 Prerequisites....47
5.3.2 Create a New Project....48
5.3.3 Build the Project....59
5.3.4 Download & Debug the Project....64
5.3.5 Debug for Multiprocessing....68
5.4 Re-configuring Project with FSP SC....69
5.4.1 Launch FSP SC from IAR EWARM....69
5.4.2 Launch from the Command Prompt....70
5.5 Note when debugging in different workspaces....70

6. FSP Configuration Users Guide....71
6.1 What is a Project?....71
6.2 Create a Project....73
6.2.1 Creating a New Project....73
6.2.2 Selecting a Board and Toolchain....74
6.2.3 Selecting a Project Template....76
6.2.4 Duplication of Resources....77
6.3 Configuring a Project....78
6.3.1 Summary Tab....78
6.3.2 Configuring the BSP....79
6.3.3 Configuring Clocks....80
6.3.4 Configuring Pins....81

6.4 Configuring Interrupts from the Stacks Tab 84
6.4.1 Creating Interrupts from the Interrupts Tab 84
6.4.2 Viewing Event Links 85
6.5 Adding and Configuring HAL Drivers 86
6.6 Reviewing and Adding Components 87

Appendix. Known Issues 88

Appendix. Tool Software Limitations 122

Appendix. How to Debug FSP Project with Flash Boot Mode 135

Appendix. How to Erase Flash Memory 137

Appendix. How to Change Boot Mode of FSP Project 142

Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e^2 studio 145
For RZ/T2 FSP v3.0.0 145
Multiprocessing with 2 cores for RZ/T devices 145
Multiprocessing with 3 or more cores for RZ/T devices 150
For RZ/N2 FSP v2.2.0 153
Multiprocessing with 2 cores for RZ/N devices 153
Multiprocessing with 3 or more cores for RZ/N devices 156

Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM ... 158
For RZ/T2 FSP v3.0.0 158
Multiprocessing with 2 cores for RZ/T devices 158
Multiprocessing with 3 or more cores for RZ/T devices 164
For RZ/N2 FSP v2.2.0 168
Multiprocessing with 2 cores for RZ/N devices 168
Multiprocessing with 3 or more cores for RZ/N devices 170

Revision History 173

1. Introduction

1.1 Overview

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

1.2 Introduction to FSP

1.2.1 Purpose

The Renesas Flexible Software Package (FSP) is an optimized software package designed to provide easy to use, scalable, high quality software for embedded system design. The primary goal is to provide lightweight, efficient 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 SC

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 Evaluation Board User's Manual

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

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

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

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

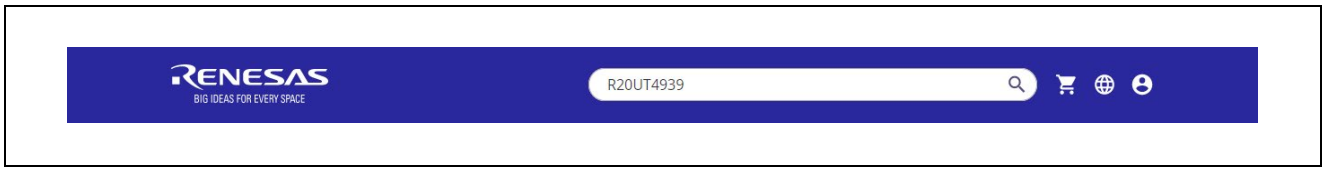


Figure 1: Search Box in Renesas Web Page

1.3.2 FSP Documentation

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

These documents are available in 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".

1.4 Starting Development Introduction

FSP application project can be created by e² studio or FSP SC (for IAR EWARM), and this Getting Started includes tutorial for both tools; the chapters you should read changes.

e² studio users should read the following chapters:

- Chapter 2 ”Set up Evaluation Board”
- Chapter 3 “e² 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 Evaluation Board”
- Chapter 5 “FSP SC User Guide”
- Chapter 6 “FSP Configuration Users Guide”

The summary of each chapter is shown below.

- Chapter 2 ”Set up Evaluation Board”
  - Explains how to setup Evaluation Board to proceed the tutorials in Chapter 4 and 5.
- Chapter 3 “e² studio Setup”
  - Explains the setup of e² 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 e² studio.
- Chapter 5 “FSP SC User Guide”
  - Explains the tutorial with minimal steps to create an FSP project as IAR EWARM project by using the FSP SC and to run and debug the created IAR EWARM project.
- Chapter 6 “FSP Configuration Users Guide”
  - Explains how to create and configure an FSP project in detail.
  - The explanation is described based on e² studio, but most of the explanations are applied to the FSP SC.

2. Set up Evaluation Board

2.1 Obtaining an Evaluation Board

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

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

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

2.2 System Configuration

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

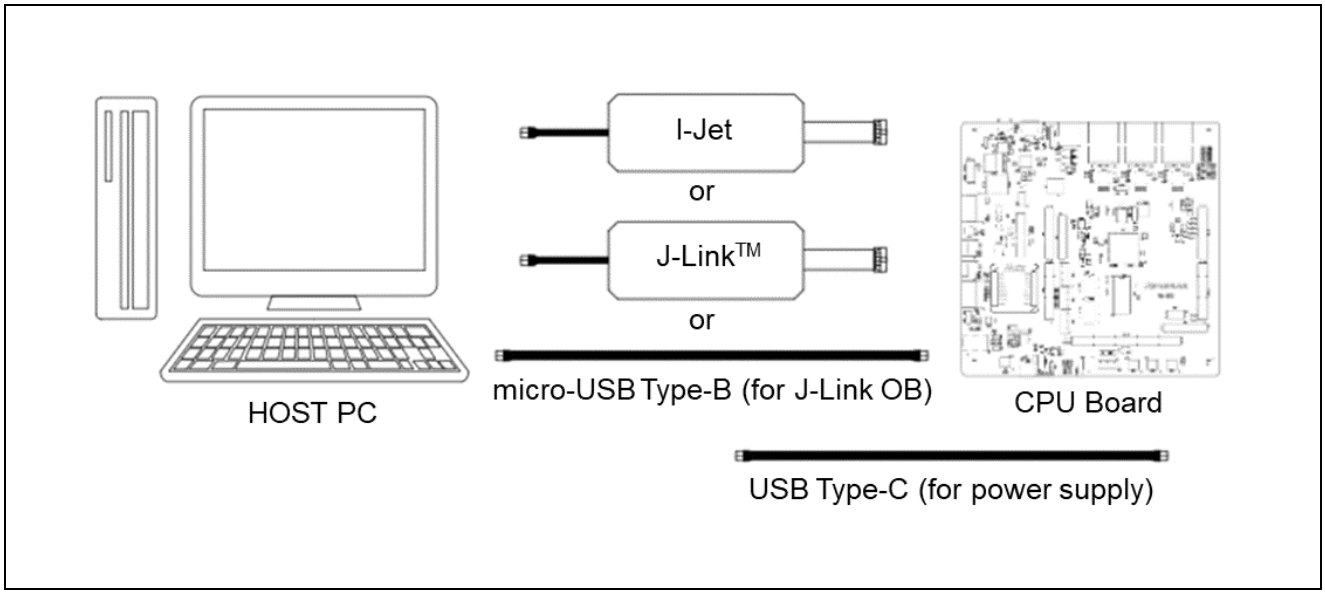


Figure 2: System Configuration Example – with Evaluation Board

For the details, please refer to the related document "1.3.1 Evaluation Board 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/T2ME, RZ/T2H	RZ/T2 FSP v3.0.0	2025-04.1	V8.30
RZ/N	RZ/N2L, RZ/N2H	RZ/N2 FSP v2.2.0	2025-01	V8.12e

Regarding how to update J-Link firmware, please confirm the procedure described in the following link into Renesas Knowledge Base web site.
https://en-support.renesas.com/knowledgeBase/20736714

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 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.

Note:

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

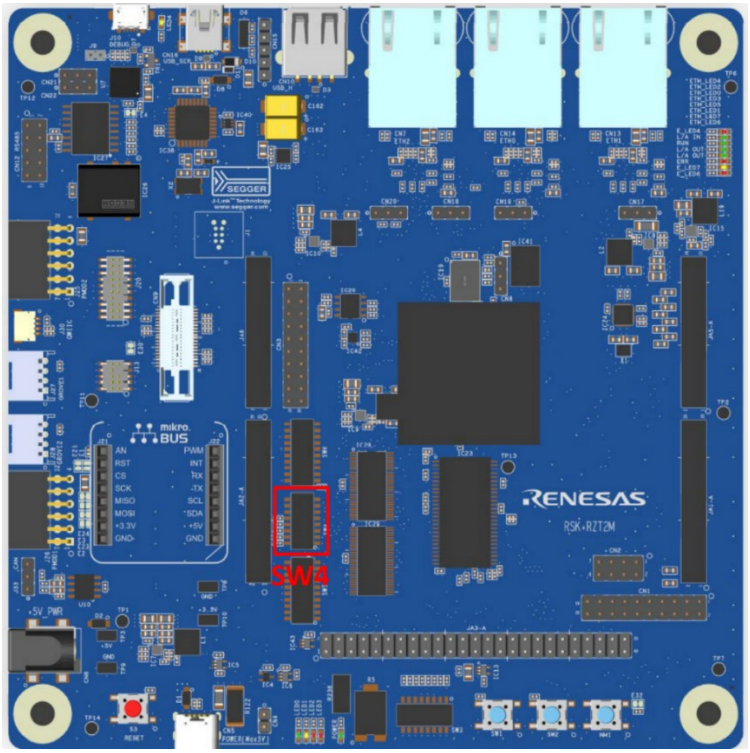


Figure 3: Switch Position of Operation Mode Settings for RSK+RZT2M

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 I-Jet to the RSK+RZT2M board ensuring that it is plugged in to the header “J20”.

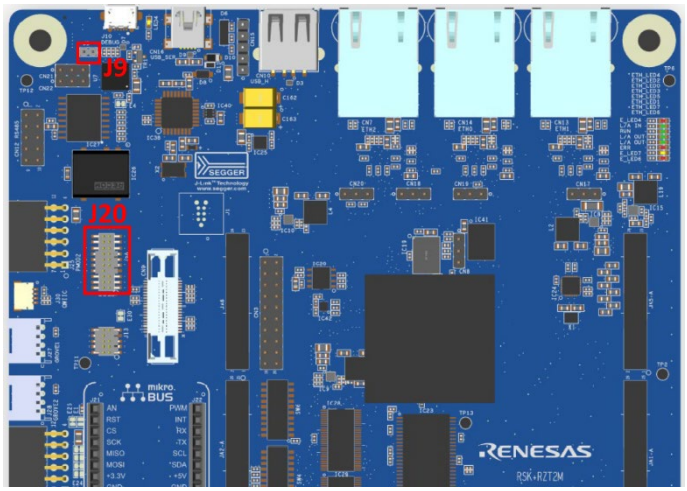


Figure 4: Jumper Position of JTAG Connection for RSK+RZT2M

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.

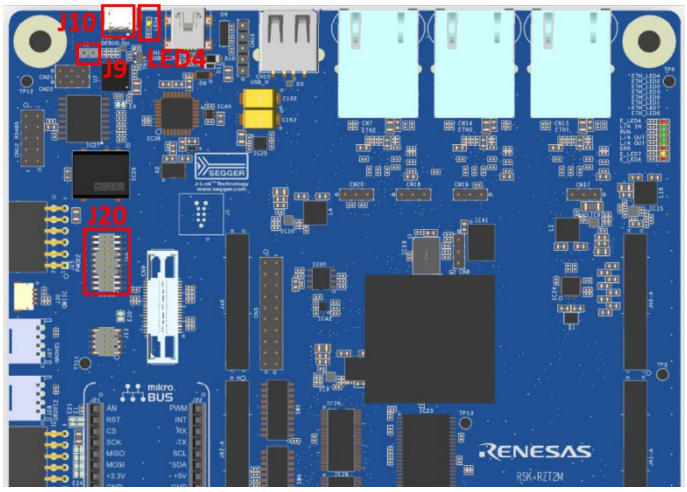


Figure 5: J-Link OB Connection Settings for RSK+RZT2M

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.

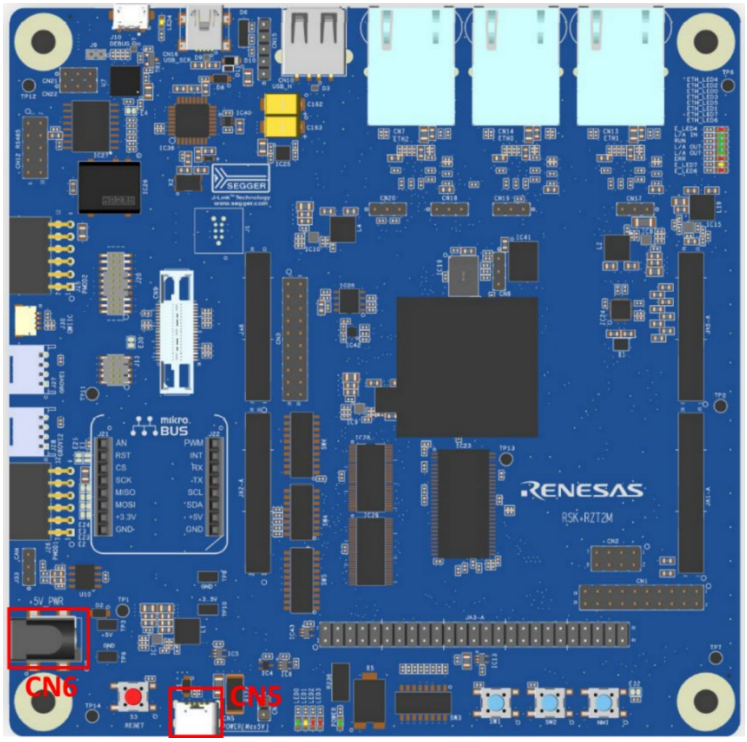


Figure 6: How to Power Supply for RSK+RZT2M

2.4.2 RSK+RZT2L

2.4.2.1 Boot Mode

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

Note:

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

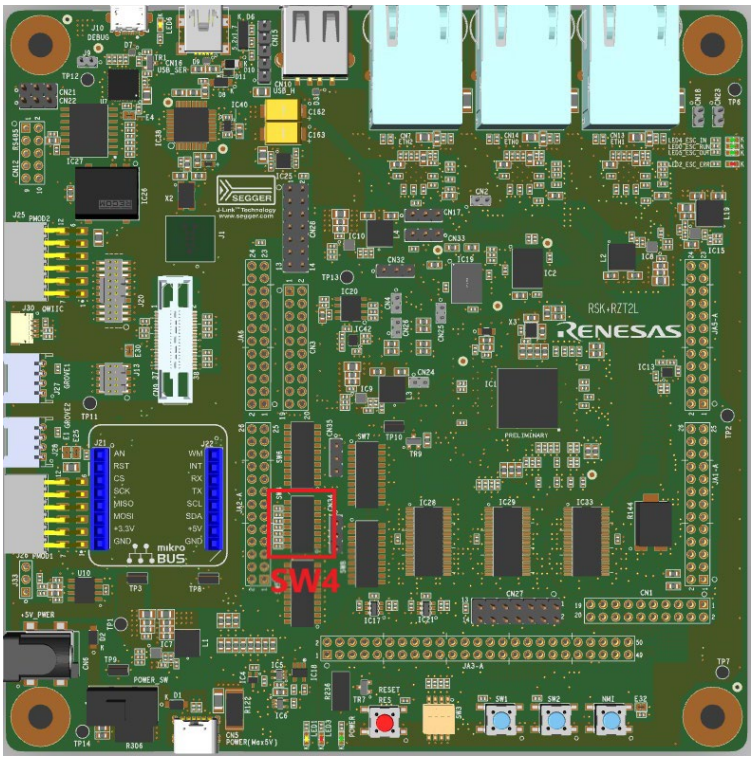


Figure 7: Switch Position of Operation Mode Settings for RSK+RZT2L

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 I-Jet to the RSK+RZT2L board ensuring that it is plugged in to the header “J20”.

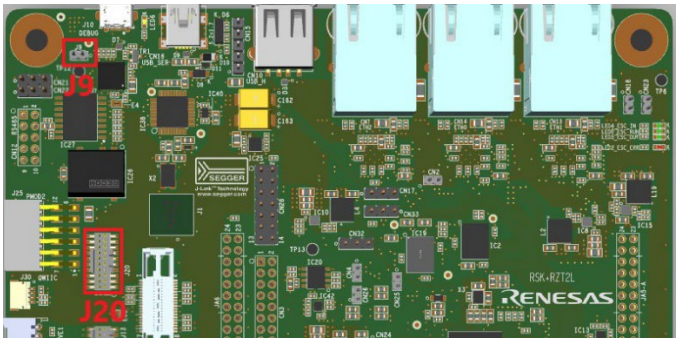


Figure 8: Jumper Position of JTAG Connection for RSK+RZT2L

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.

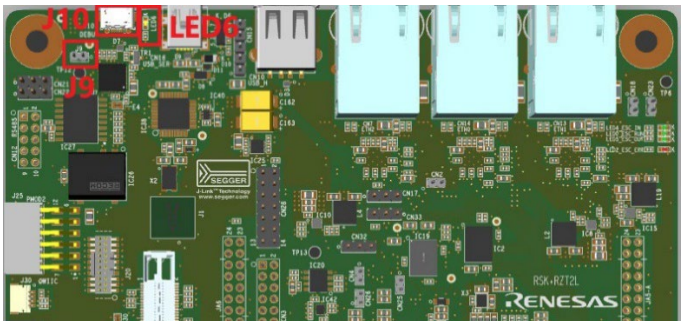


Figure 9: J-Link OB Connection Settings for RSK+RZT2L

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.

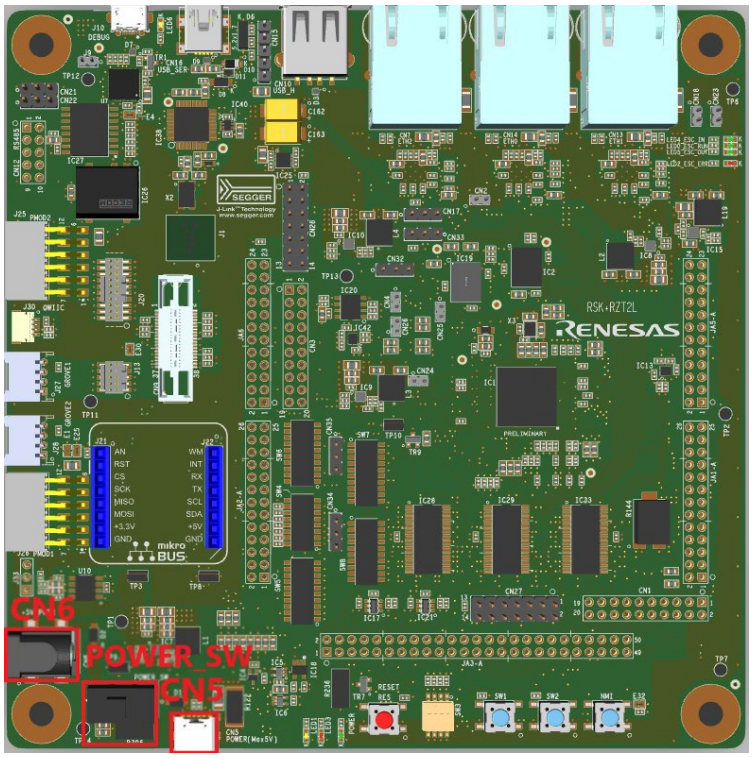


Figure 10: How to Power Supply for RSK+RZT2L

2.4.3 RSK+RZT2ME

For each setting, see 2.4.1 RSK+RZT2M.

2.4.4 RZ/T2H Evaluation Board

2.4.4.1 Boot Mode

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

Note:

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

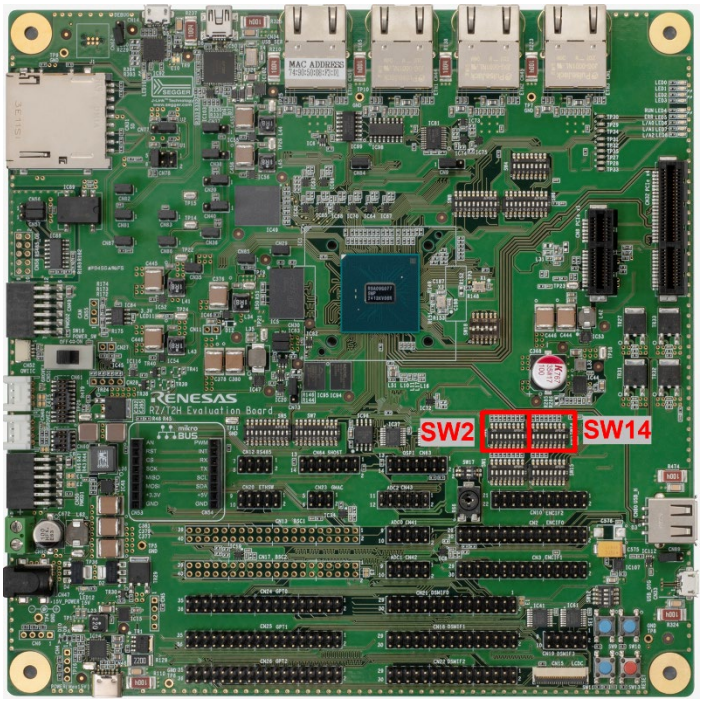


Figure 11: Switch Position of Operation Mode Settings for RZ/T2H Evaluation Board

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

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

2.4.4.2 Debugger Connection

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

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

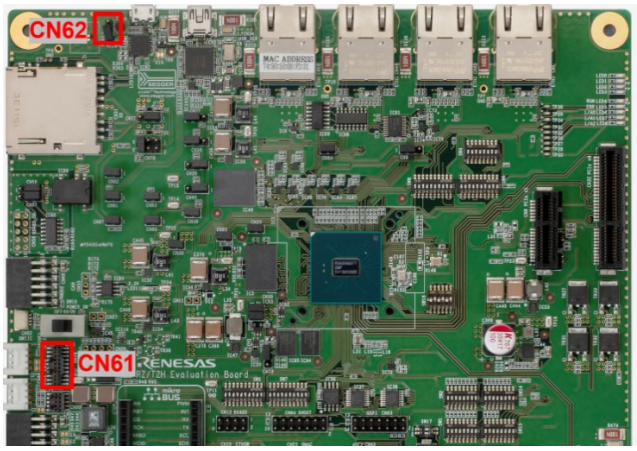


Figure 12: Jumper Position of JTAG Connection for RZ/T2H Evaluation Board

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

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

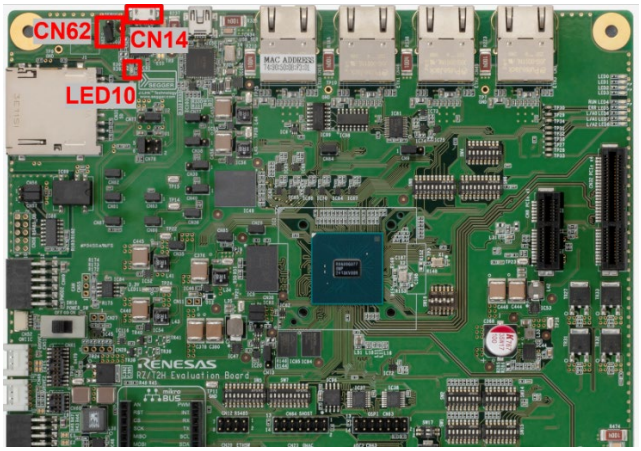


Figure 13: J-Link OB Connection Settings for RZ/T2H Evaluation Board

2.4.4.3 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 “CN46” of the RZ/T2H evaluation board.
- When connecting the AC / DC adapter, connect it to the USB connector “CN47” of the RZ/T2H evaluation board.
- After connecting to the power (CN46 or CN47), turn on the POWER_SW slide switch to start power supply.

Note:

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

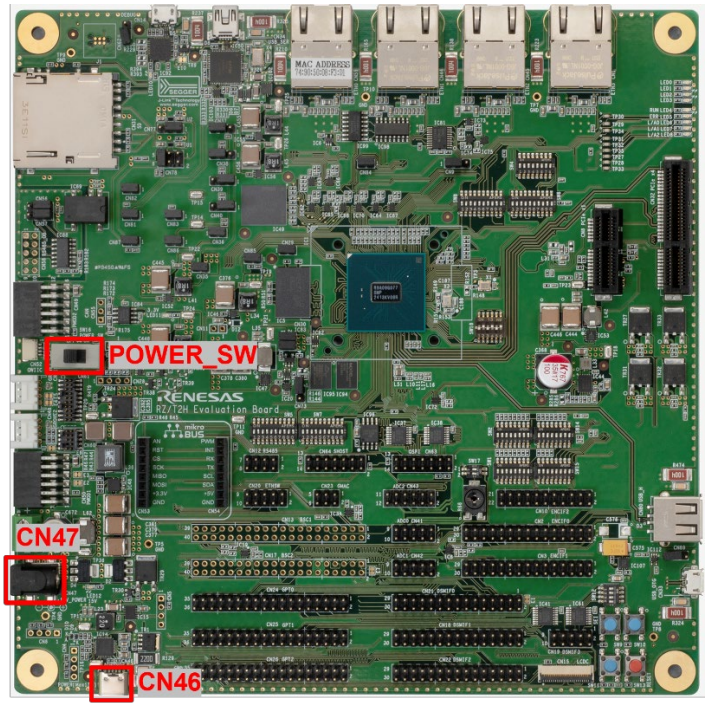


Figure 14: How to Power Supply for RZ/T2H Evaluation Board

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.

Note:

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

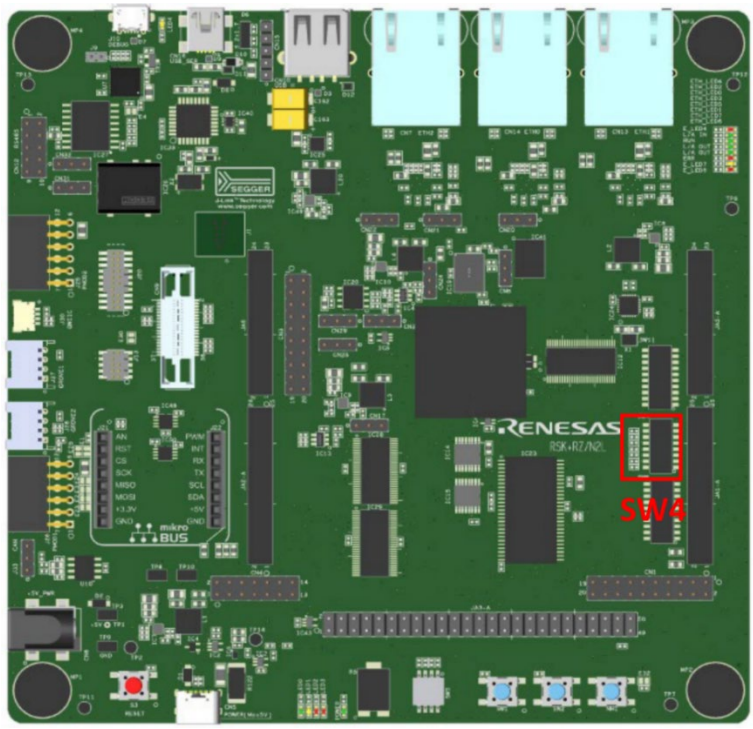


Figure 15: Switch Position of Operation Mode Settings for RSK+RZN2L

Table 5 Operation Mode Switch Settings for RSK+RZN2L

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

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 I-Jet to the RSK+RZN2L board ensuring that it is plugged in to the header “J20”.

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

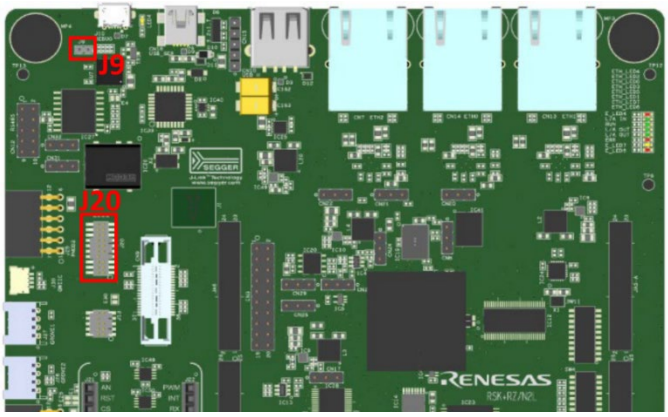


Figure 16: Jumper Position of JTAG Connection for RSK+RZN2L

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.

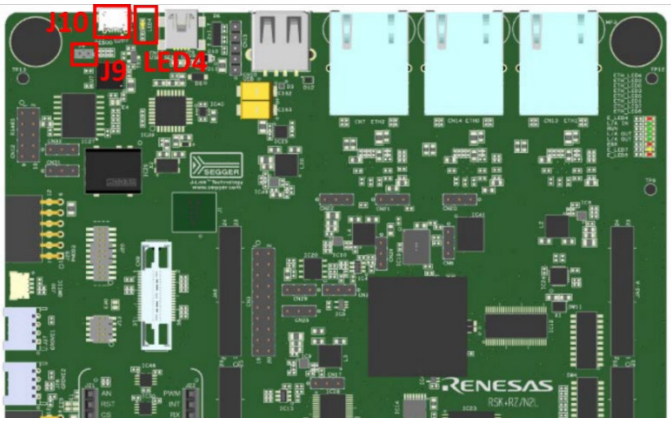


Figure 17: J-Link OB Connection Settings for RSK+RZN2L

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.

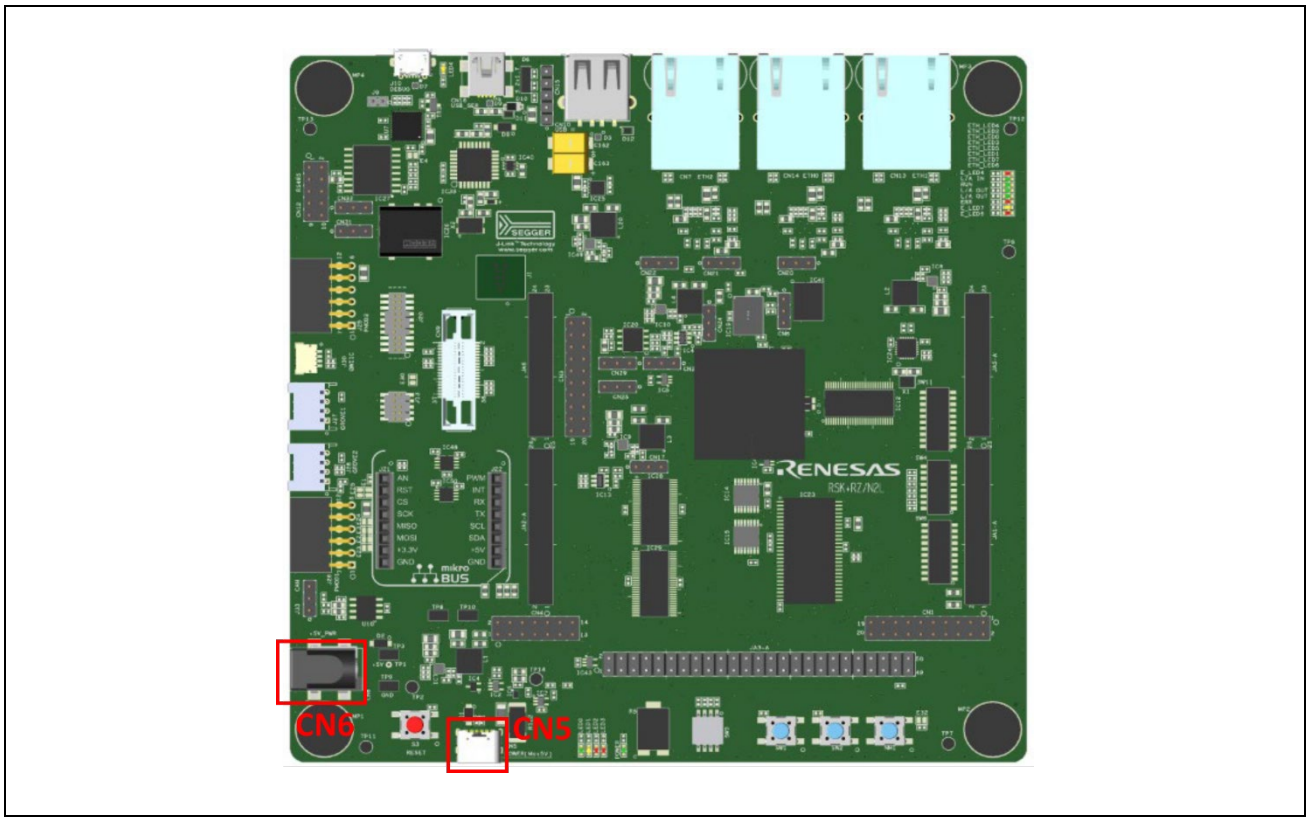


Figure 18: How to Power Supply for RSK+RZN2L

2.5.2 RZ/N2H Evaluation Board

2.5.2.1 Boot Mode

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

Note:

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

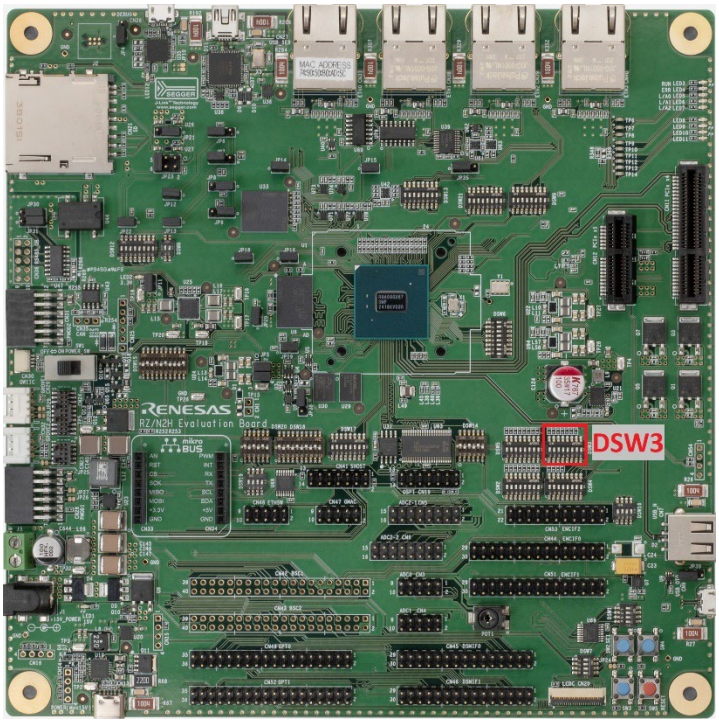


Figure 19: Switch Position of Operation Mode Settings for RZ/N2H Evaluation Board

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

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

2.5.2.2 Debugger Connection

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

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

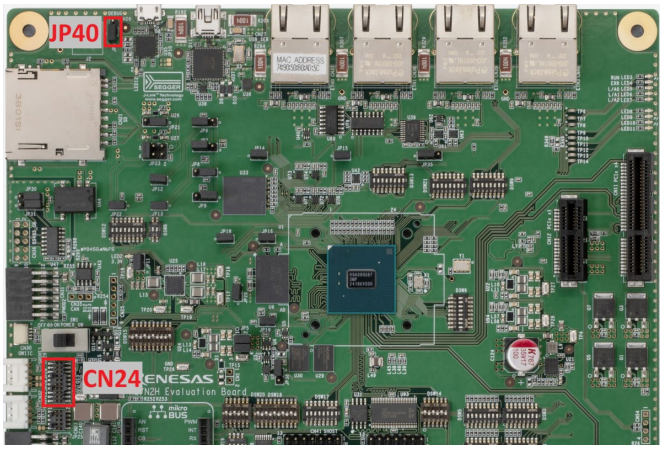


Figure 20: Jumper Position of JTAG connection for RZ/N2H Evaluation Board

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

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

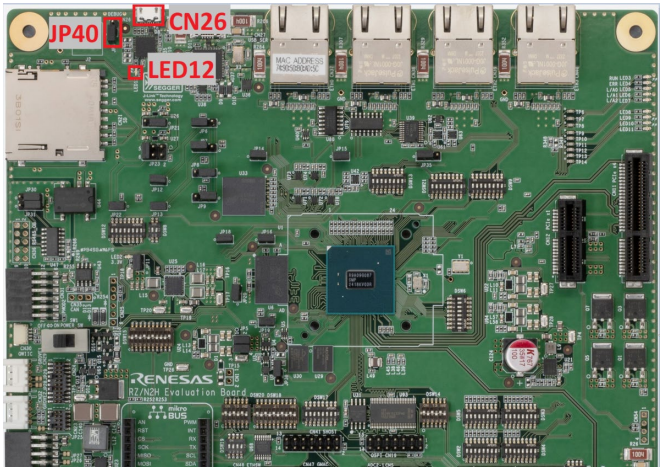


Figure 21: J-Link OB Connection Settings for RZ/N2H Evaluation Board

2.5.2.3 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 “CN13” of the RZ/N2H evaluation board.
- When connecting the AC / DC adapter, connect it to the USB connector “J1” of the RZ/N2H evaluation board.
- After connecting to the power (J1 or CN13), turn on the POWER_SW slide switch to start power supply.

Note:

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

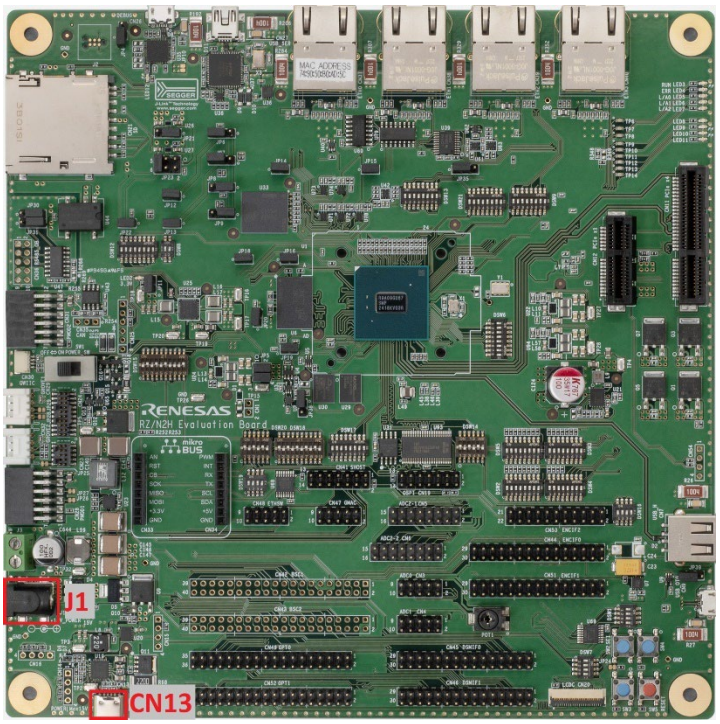


Figure 22: How to Power Supply for RZ/N2H Evaluation Board

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

- System: x64 based processor, 2 GHz or faster, CPU has dual cores or more
  - Windows® 11 (64-bit version)
  - Windows® 10 (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 following toolchains are required.

Table 7 Toolchain version for each FSP

FSP version	Core	Toolchain	Toolchain version
RZ/T2 FSP v3.0.0	CR52	GNU ARM Embedded Toolchain	13.3.Rel1 (13.3.1.arm-13-24)
	CA55	GNU ARM A-Profile (AArch64 bare-metal)	10.3-2021.07 (10.3.1.20210621)
RZ/N2 FSP v2.2.0	CR52	GNU ARM Embedded Toolchain	13.3.Rel1 (13.3.1.arm-13-24)
	CA55	GNU ARM A-Profile (AArch64 bare-metal)	10.3-2021.07 (10.3.1.20210621)

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 evaluation board. This chapter guides you through creating projects for a single-core processing and a multiprocessing with RAM execution without flash memory. In this chapter, the multiprocessing refers to a process in which CR52 CPU0 core is activated first and second core (CR52 CPU1 or CA55 Core0) operates after CR52 CPU0 core sets up for second core.

4.2 What Does Blinky Do?

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

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

- The toolchain is 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. The procedure from creating a project to running it varies depending on the number of cores used and boot mode. This chapter shows only some of the cases where RAM execution is used. Refer to Table 8 Project Creation Procedure (e^2 studio) to find out which steps are required for your application.

Table 8 Project Creation Procedure (e^2 studio)

<table>
<tr><th rowspan="2">Steps</th><th colspan="2">Single-core processing</th><th colspan="2">Multiprocessing</th></tr>
<tr><th>RAM execution</th><th>Flash boot mode</th><th>RAM execution
(Combination of (CR52 CPU0, CPU1) and (CR52 CPU0, CA55 Core0) only)</th><th>RAM execution
(Other combinations)
Flash boot mode</th></tr>
<tr><td>Check tool limitations</td><td colspan="4">Appendix. Tool Software Limitations</td></tr>
<tr><td>Erase flash memory(if needed)</td><td colspan="4">Appendix. How to Erase Flash Memory</td></tr>
<tr><td>Create a project</td><td>4.3 Create a New Project for Blinky</td><td>4.3 Create a New Project for Blinky
Appendix. How to Debug FSP Project with Flash Boot Mode</td><td>4.3 Create a New Project for Blinky</td><td rowspan="4">Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e2 studio</td></tr>
<tr><td>Build the project</td><td colspan="2">4.4.1 Build</td><td>4.4.2 Build for Multiprocessing</td></tr>
<tr><td>Debug the project</td><td colspan="2">4.5.2 Debug Steps</td><td rowspan="2">4.7 Debug and Run for Multiprocessing</td></tr>
<tr><td>Run the project</td><td colspan="2">4.6 Run the Blinky Project</td></tr>
</table>

(Continued on next page)

Note for multiprocessing projects:

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

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

The secondary project should be created after the primary project is created in 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 e^2 studio, click **File** > **New** > **Renesas C/C++ Project > Renesas RZ**.

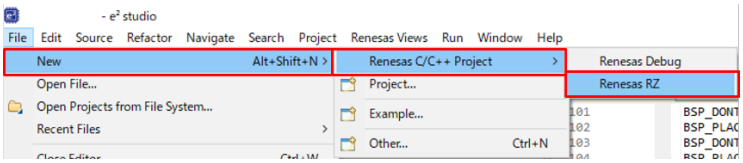


Figure 23: New C/C++ Project

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

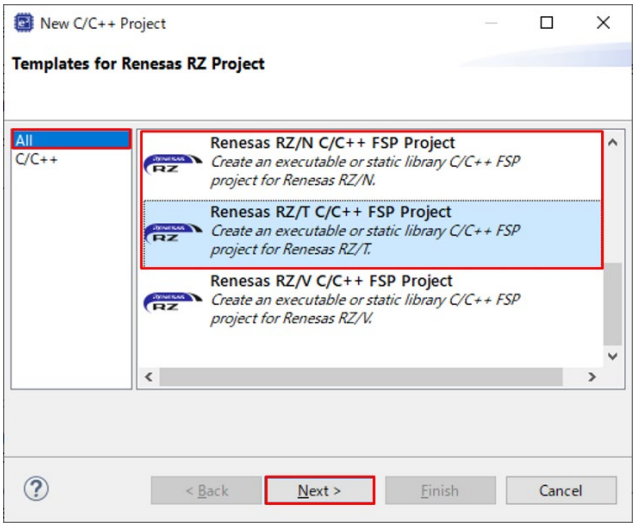


Figure 24: Renesas RZ C/C++ FSP Project

(Continued on next page)

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

Table 9 e² studio Newly Created Project Settings (1)

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

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

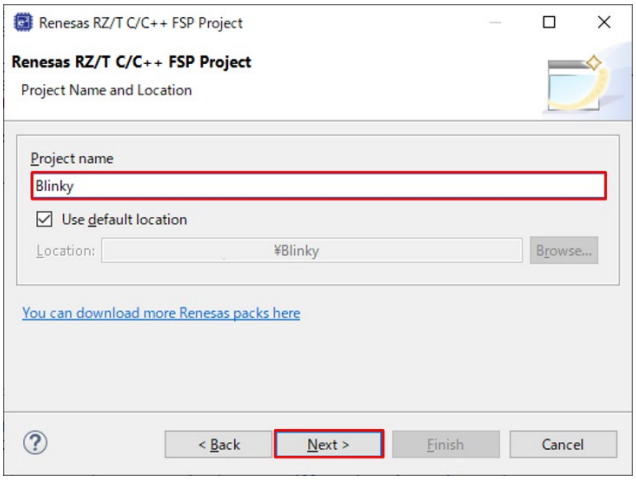


Figure 25 : e² studio Project Configuration Window (Part 1)

(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 board.
7. (Multicore device ONLY) Select the Core from the drop-down list.
8. Select toolchains and version, then click **Next**.
   - If there is NOT the target toolchain, please download the version of the toolchain from ARM Developer website and install it.

Table 10 e² studio Newly Created Project Settings (2)

	Single-core processing	Multiprocessing (CR52 CPU0, CR52 CPU1)		Multiprocessing (CR52 CPU0, CA55 Core0)	
		Primary	Secondary	Primary	Secondary
Board	RSK+RZXXX (RAM execution without flash memory) or RZXXX Evaluation Board (RAM execution without flash memory)				
Core	CR52_0 or CR52 CPU0	CR52_0 or CR52 CPU0	CR52_1 or CR52 CPU1	CR52 CPU0	CA55 Core0
IDE Project Type	e² studio managed build				
Toolchains	GNU ARM Embedded 13.3.1.arm-13-24				GNU ARM A-Profile (AArch64 bare-metal) and 10.3.1.20210621
Debbuger	J-Link ARM				

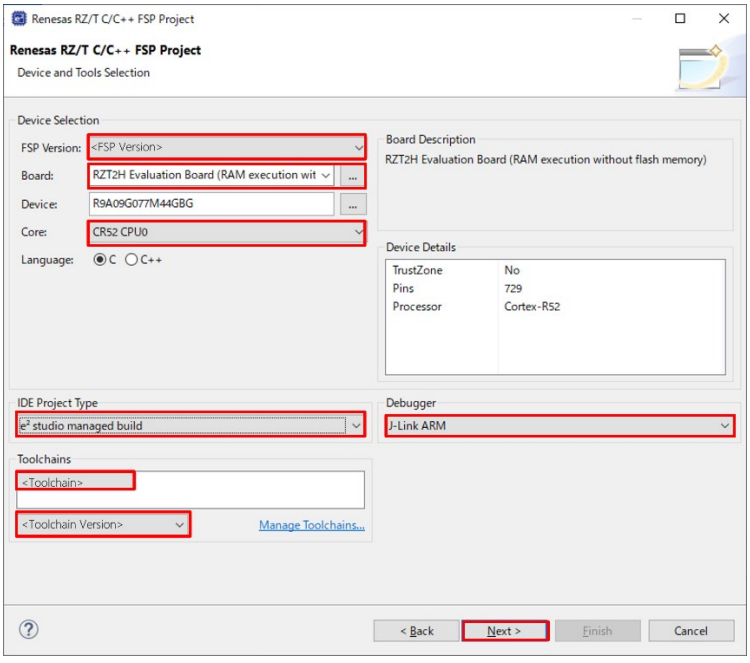


Figure 26 : e² studio Project Configuration Window (Part 2)

(Continued on next page)

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

Table 11 e² studio Newly Created Project Settings (3)

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

Note:

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

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

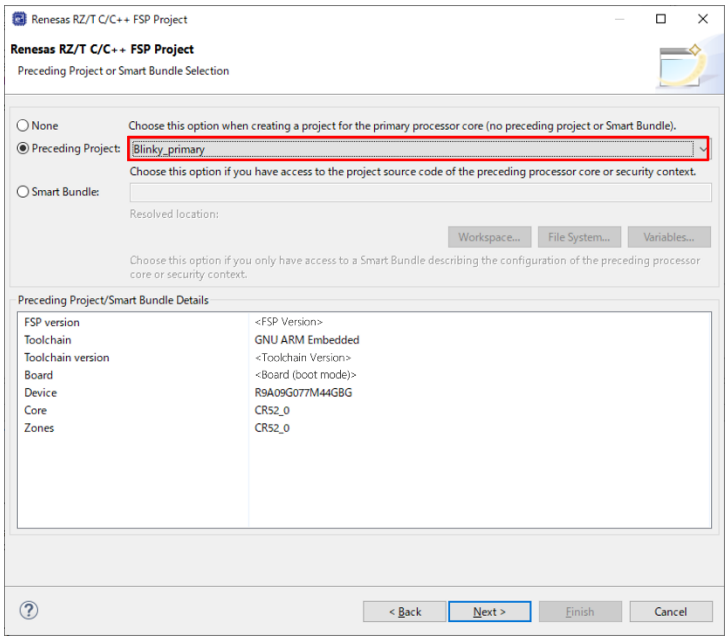


Figure 27 e² studio Project Configuration Window (Part 3)

(Continued on next page)

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

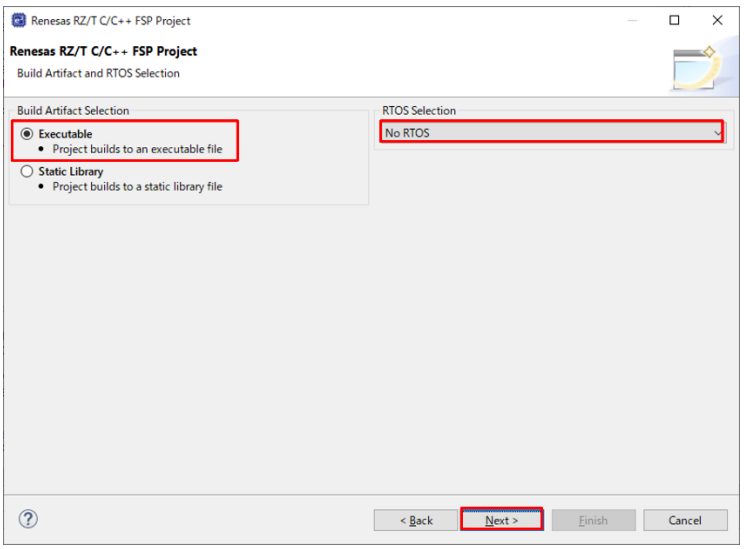


Figure 28 : e² studio Project Configuration Window (Part 4)

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

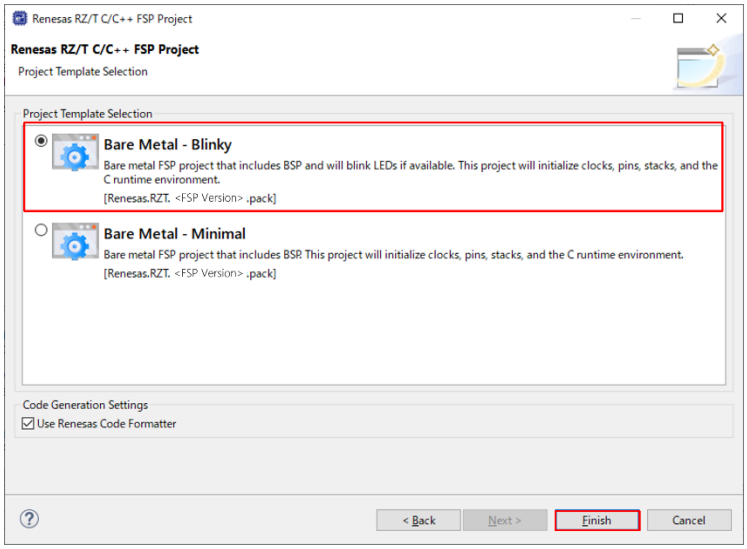


Figure 29 : e² studio Project Configuration Window (Part 5)

(Continued on next page)

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

Note for the primary project using CR52 CPU0:

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

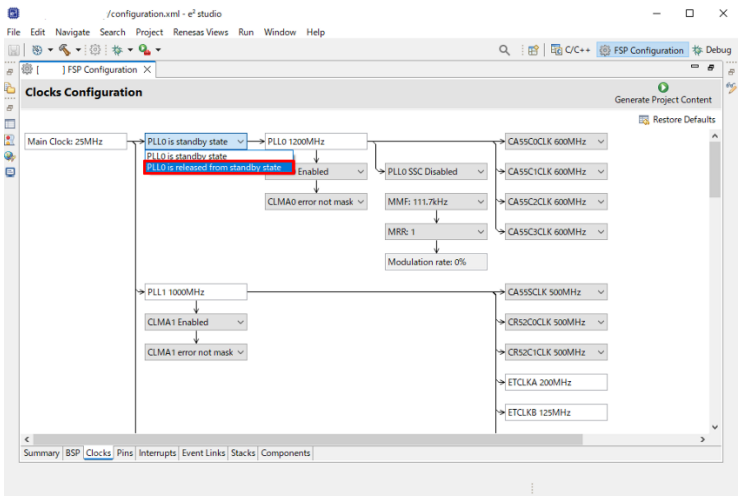


Figure 30 : Enable PLL0 in the Primary Project using CR52 CPU0

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

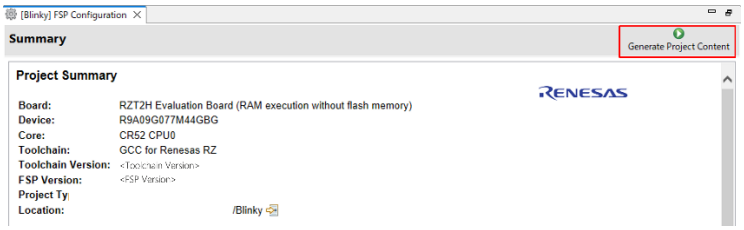


Figure 31 : e^2 studio Project Configuration Tab

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, please refer to Section 4.4.2 Build for Multiprocessing.

4.4.1 Build

There are three ways to build a project:

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

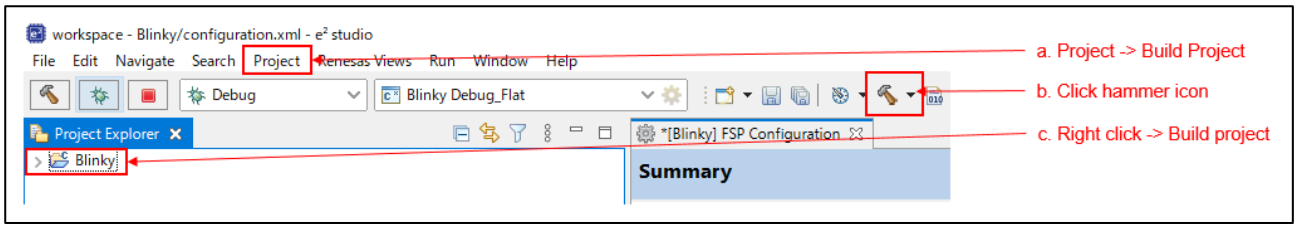


Figure 32 : e² studio Project Explorer Window

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.

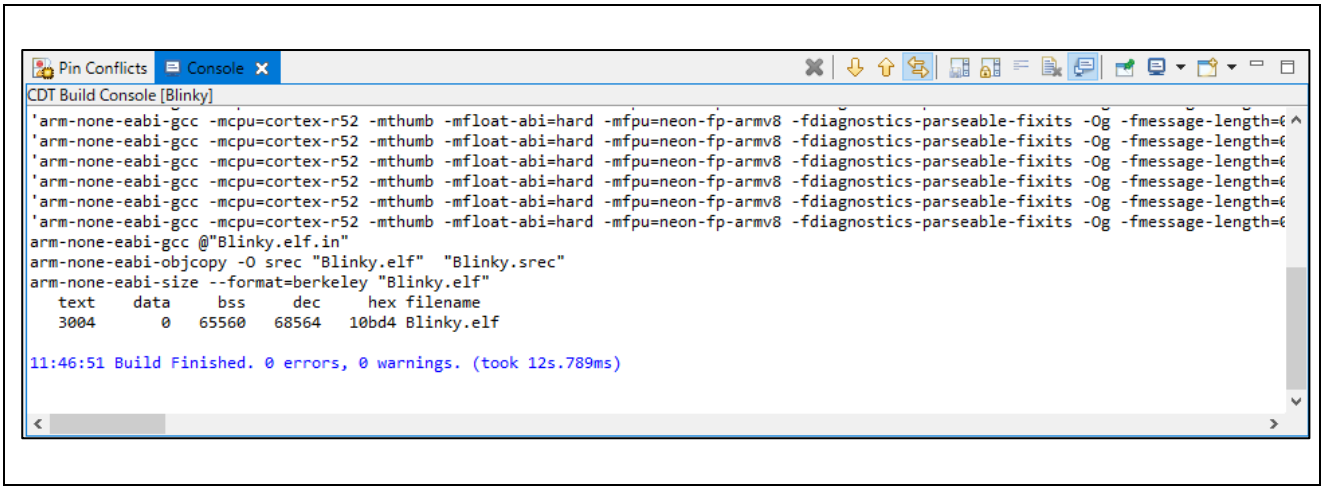


Figure 33 : e² studio Project Build Console

4.4.2 Build for Multiprocessing

Build the projects for multiprocessing with the following steps.

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

4.5 Debug the Blinky Project

4.5.1 Debug Prerequisites

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

- The board to be connected to e^2 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.

Evaluation 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.

Note:

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

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

(RAM exec): The boot mode used in the project is RAM execution without flash memory.
(CR52): The core used in the project is CR52.
(CA55): The core used in the project is CA55.

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

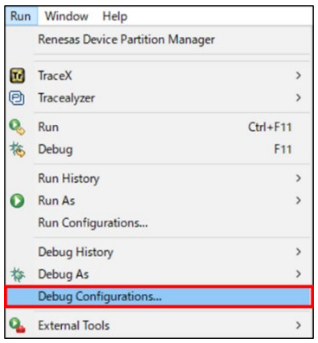


Figure 34 : e^2 studio Debugger Configurations Selection Option

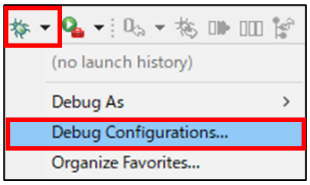


Figure 35 : e^2 studio Debug Icon

(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.

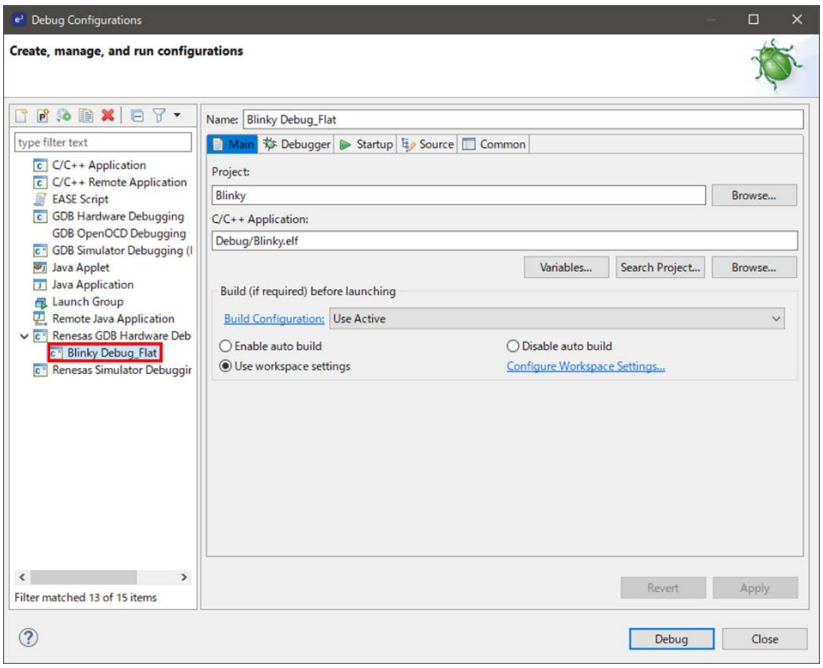


Figure 36 : e² studio Debugger Configurations Window with Blinky Project

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

Table 12 e² studio Newly Created Project Debug Settings (1)

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

(Continued on next page)

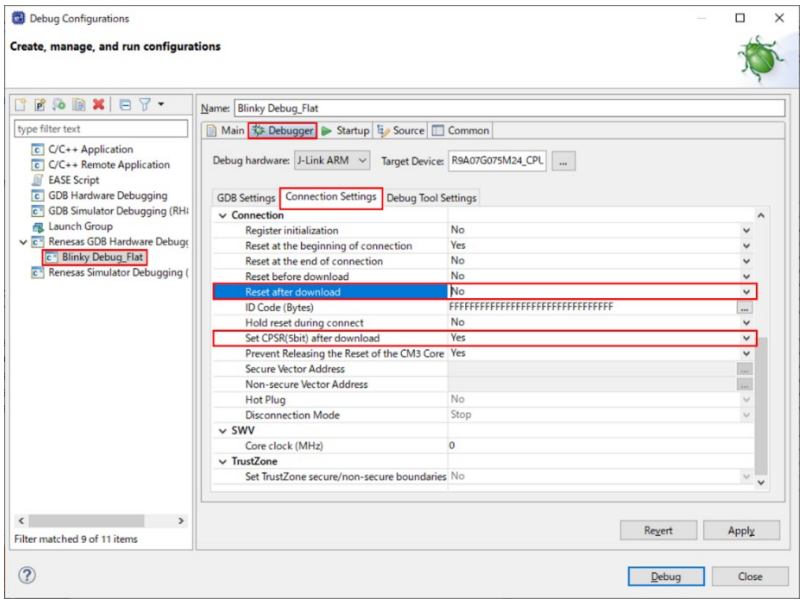


Figure 37 : e² studio Debugger Configurations Window with Blinky Project (CR52 CPU0)

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

Table 13 e² studio Newly Created Project Debug Settings (2)

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

(Continued on next page)

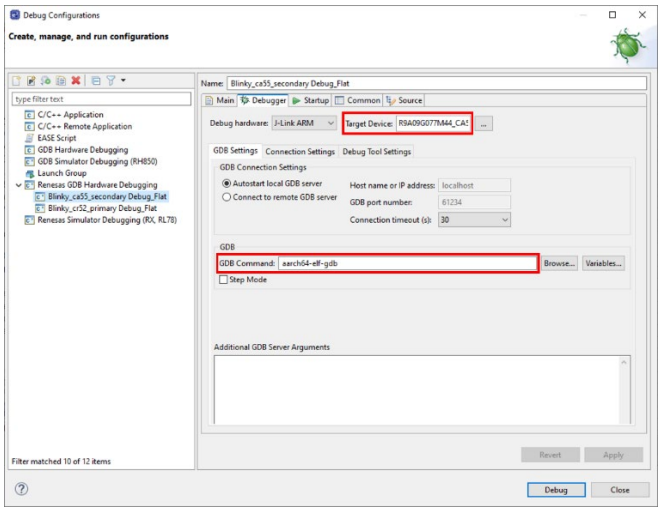


Figure 38 : e² studio Debugger Configurations Window with Blinky Project (CA55)

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

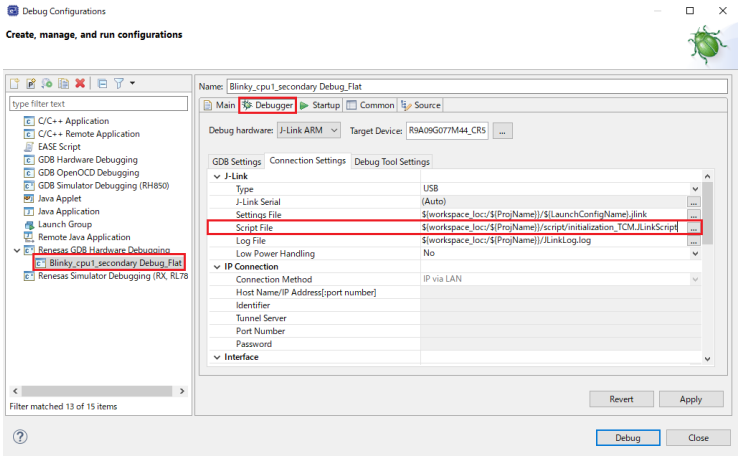


Figure 39 e² studio Debugger Configurations Window with Blinky Project (RZ/T2H CR52 CPU1)

(Continued on next page)

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

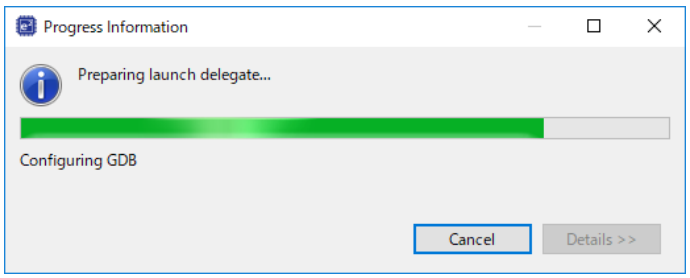


Figure 40: Start Debugging

4.5.3 Details about the Debug Process

In debug mode, e² 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.

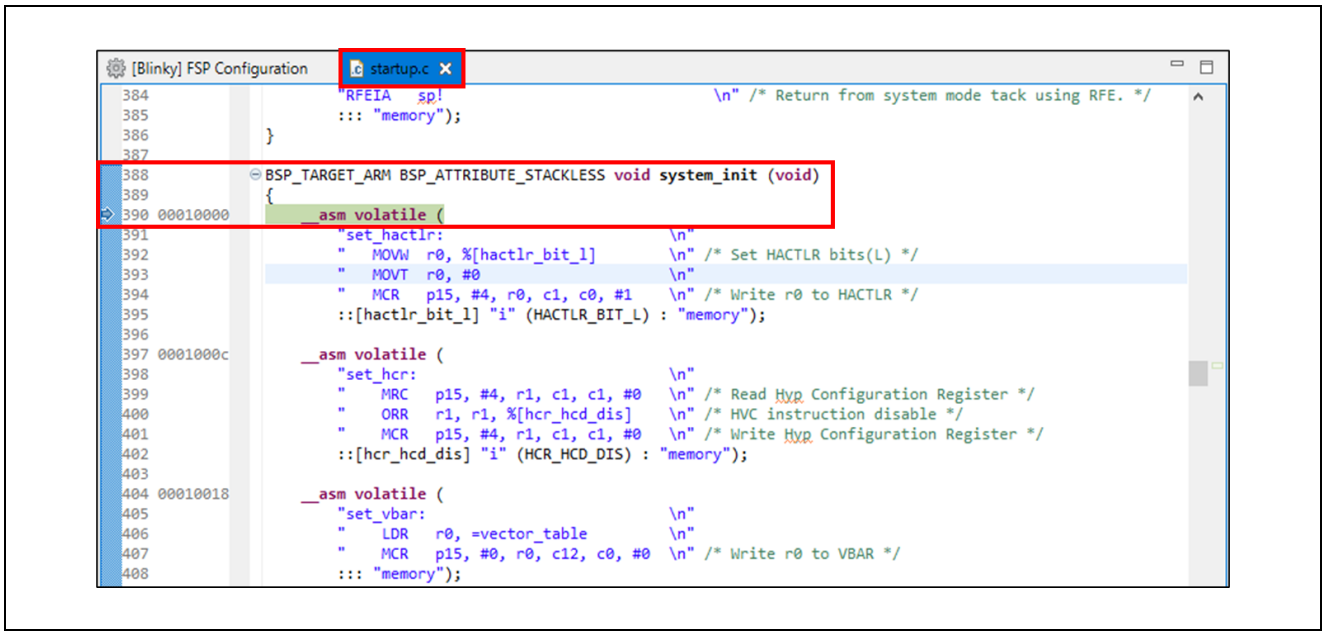


Figure 41 : e² studio Debugger Memory Window

4.6 Run the Blinky Project

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

Figure 42 : 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)
  - RSK+RZ/T2ME: LED0-1 (CPU0), LED2-3 (CPU1)
  - RZ/T2H Evaluation Board: LED0 (CR52 CPU0), LED1 (CR52 CPU1), LED2 (CA55 Core0)
- RZ/N series
  - RSK+RZ/N2L: LED0-3
  - RZ/N2H Evaluation Board: LED3 (CR52 CPU0), LED4 (CR52 CPU1), LED8 (CA55 Core0)

(Continued on next page)

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

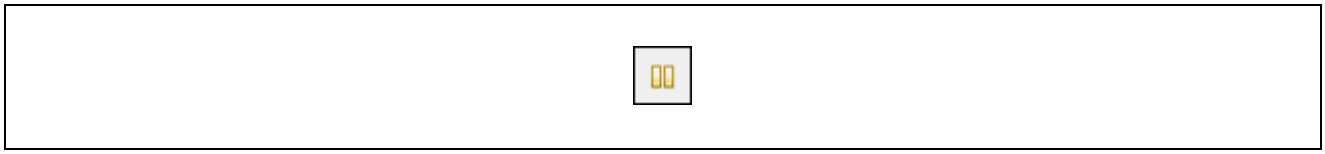


Figure 43 : e² studio Debugger Pause Icon

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

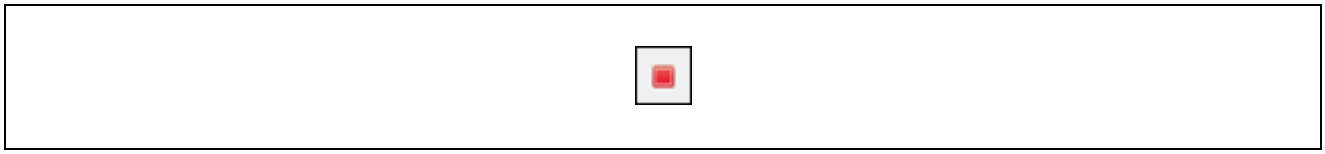


Figure 44 : e² studio Debugger Stop Icon

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 stays connected, 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.

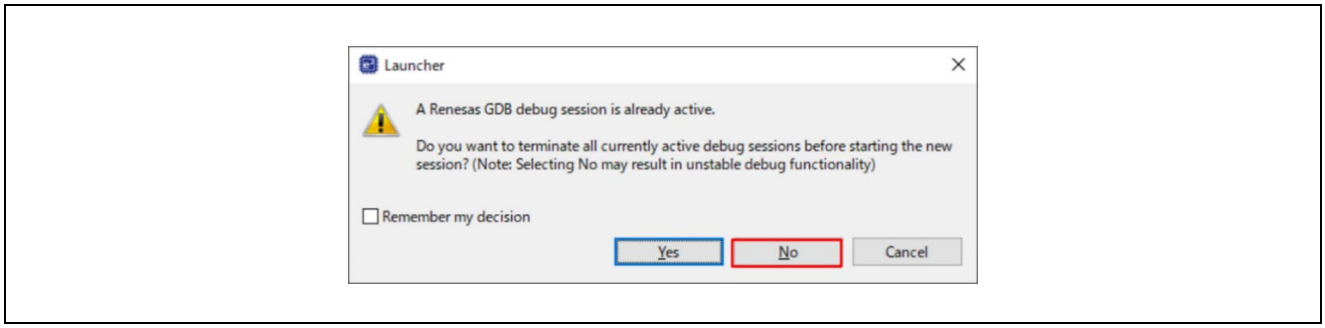


Figure 45 Warning Window of Starting Debug Session

4. When Figure 45 is shown, please click **Yes** to proceed the launch.

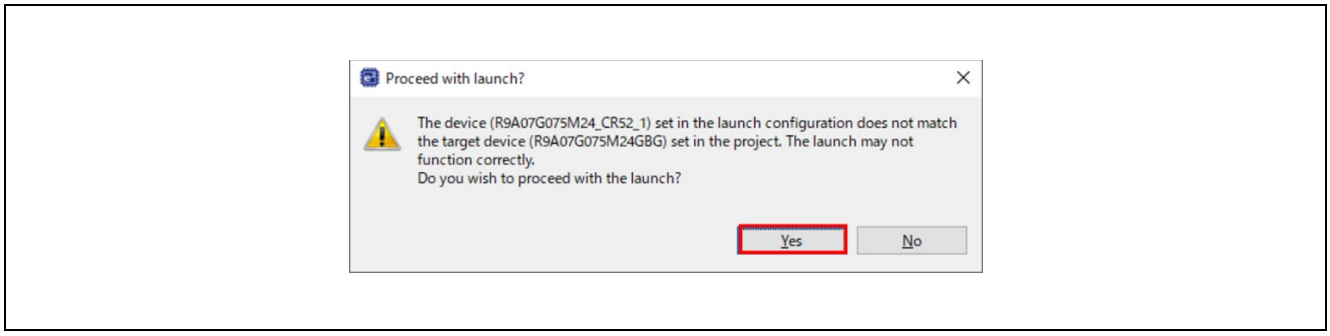


Figure 46 Warning Window of Device Name

(Continued on next page)

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

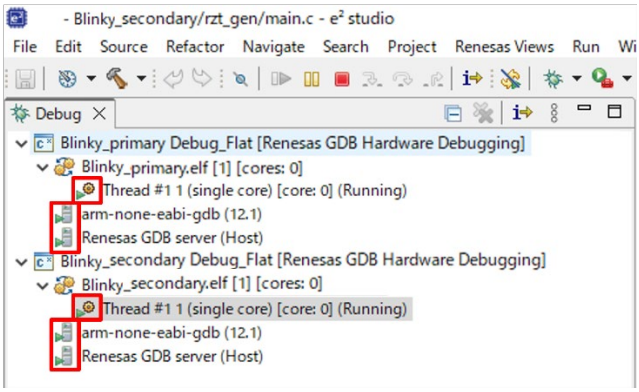


Figure 47 During Program Execution

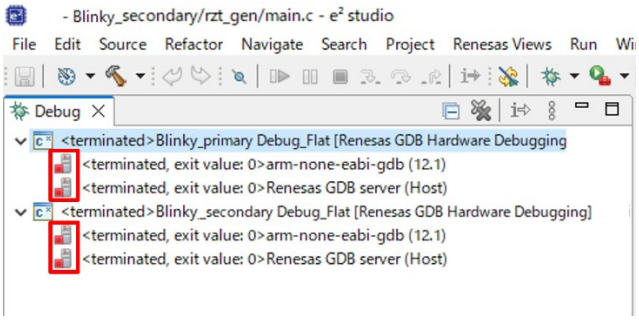


Figure 48 Terminated Program

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.

(Continued on next page)

To import the projects, follow these steps:

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

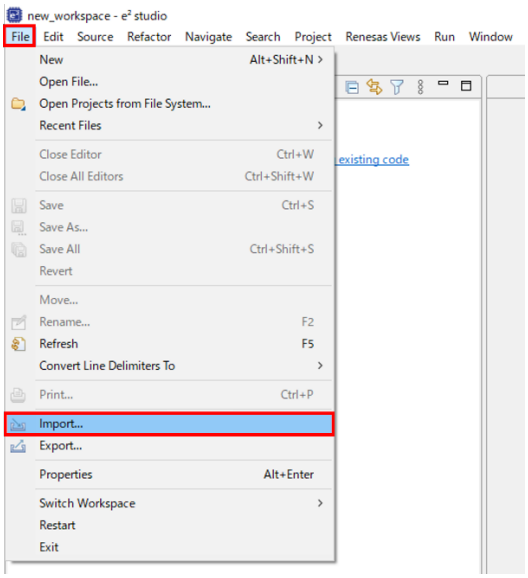


Figure 49 e² studio Import

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

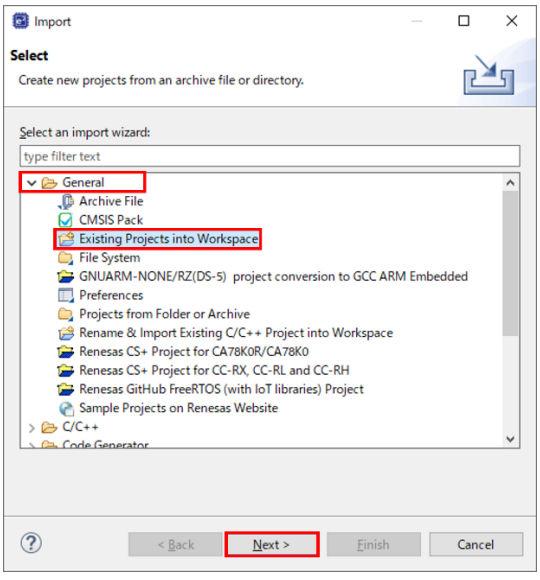


Figure 50 e² studio Select Import Type

(Continued on next page)

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 recommended to set **Copy projects into workspace** in **Options** to avoid updating the same project from multiple workspaces.

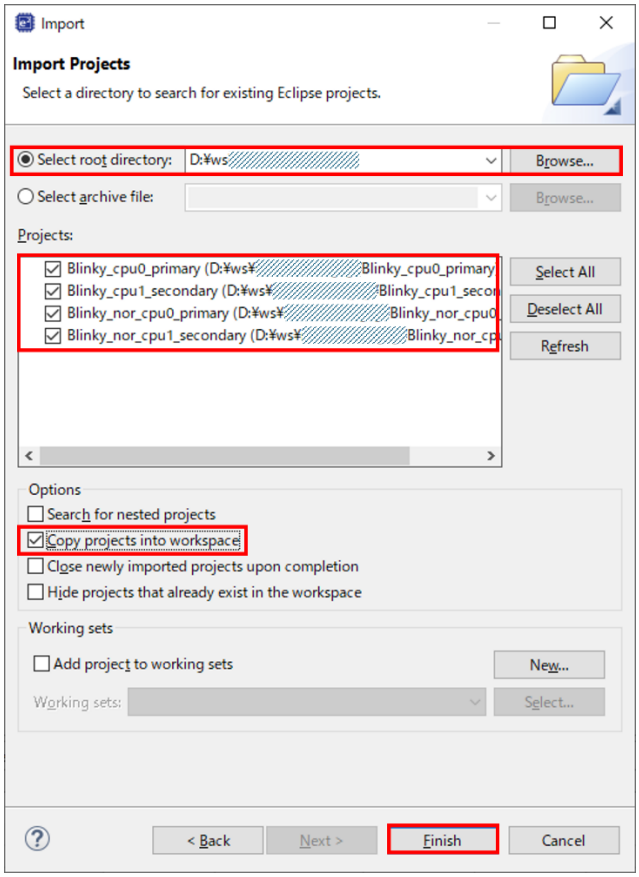


Figure 51 e² studio Select Root Directory to Import Project

(Continued on next page)

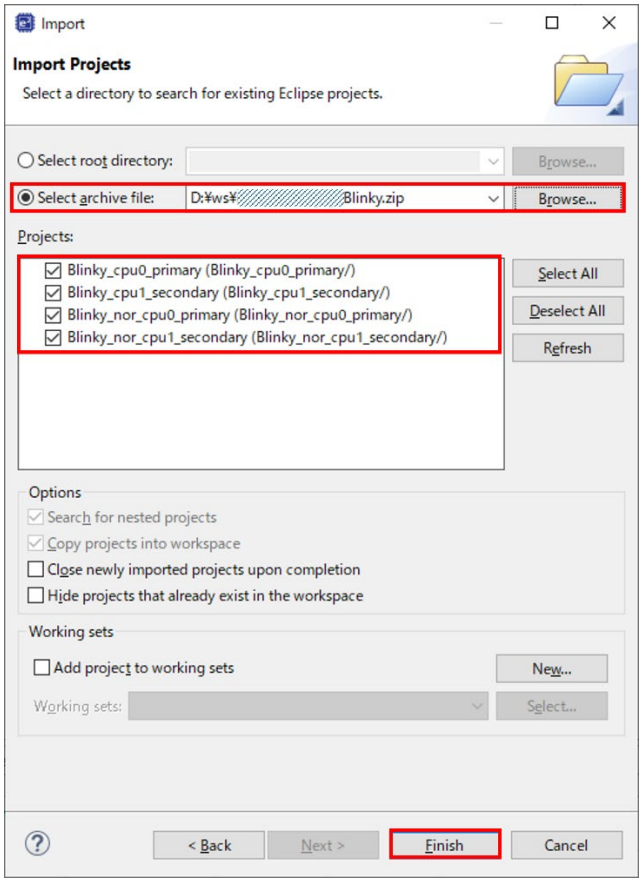


Figure 52 e² studio Select Archive File to Import Project

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

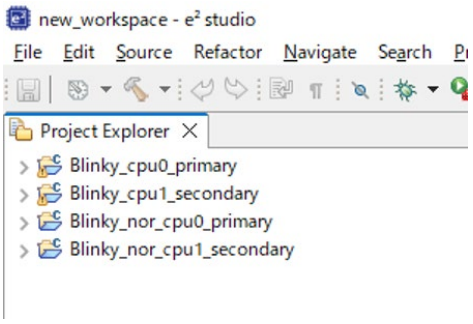


Figure 53 e² studio New Workspace

Note:

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

5. FSP SC User Guide

5.1 What is FSP SC?

The Renesas FSP Smart Configurator (FSP SC) is a desktop application designed to configure device hardware such as clock 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 with RAM execution without flash memory. In this chapter, the multiprocessing refers to a process in which CR52 CPU0 core is activated first and second core(CR52 CPU1 or CA55 Core0) operates after CR52 CPU0 core sets up for second core.

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

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

- The toolchain is 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 FSP SC with IAR EWARM

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

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

5.3.1 Prerequisites

- IAR EWARM installed and licensed.
  - Please refer to 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/T2ME:

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

- EWARM_Patch_for_RZT2ME (EWARM_Patch_for_RZT2ME_rev1.0.zip)
  - This patch file is available in http://www.renesas.com/rzt2me.

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

Note for RZ/T2H and RZ/N2H:

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

- EWARM_Patch_for_RZT2H_RZN2H (EWARM_Patch_for_RZT2H_RZN2H_rev1.0.zip)
  - This patch file is available in http://www.renesas.com/rzt2h and http://www.renesas.com/rzn2h.

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

5.3.2 Create a New Project

The following steps are required to create a project using IAR EWARM, FSP SC and FSP. The procedure from creating a project to running it varies depending on the number of cores used and boot mode. This chapter shows only some of the cases where RAM execution is used. Refer to Table 14 Project Creation Procedure (IAR EWARM, FSP SC) to find out which steps are required for your application.

Table 14 Project Creation Procedure (IAR EWARM, FSP SC)

<table>
<tr><th rowspan="2">Step</th><th colspan="2">Single-core processing</th><th colspan="2">Multiprocessing</th></tr>
<tr><th>RAM execution</th><th>Flash boot mode</th><th>RAM execution
(Combination of (CR52 CPU0, CPU1) and (CR52 CPU0, CA55 Core0) only)</th><th>RAM execution
(Other combinations)
Flash boot mode</th></tr>
<tr><td>Check tool limitations</td><td colspan="4">Appendix. Tool Software Limitations</td></tr>
<tr><td>Erase flash memory(if needed)</td><td colspan="4">Appendix. How to Erase Flash Memory</td></tr>
<tr><td>Create a project</td><td>5.3.2 Create a New Project</td><td>5.3.2 Create a New Project
Appendix. How to Debug FSP Project with Flash Boot Mode</td><td>5.3.2 Create a New Project</td><td rowspan="4">Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM</td></tr>
<tr><td>Build the project</td><td colspan="2">5.3.3.1 Build</td><td>5.3.3.2 Build for Multiprocessing</td></tr>
<tr><td>Debug the project</td><td colspan="2" rowspan="2">5.3.4 Download & Debug the Project</td><td rowspan="2">5.3.5 Debug for Multiprocessing</td></tr>
<tr><td>Run the project</td></tr>
</table>

Note for multiprocessing projects:

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

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

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

1. Start the FSP SC.
   - FSP SC 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

(Continued on next page)

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

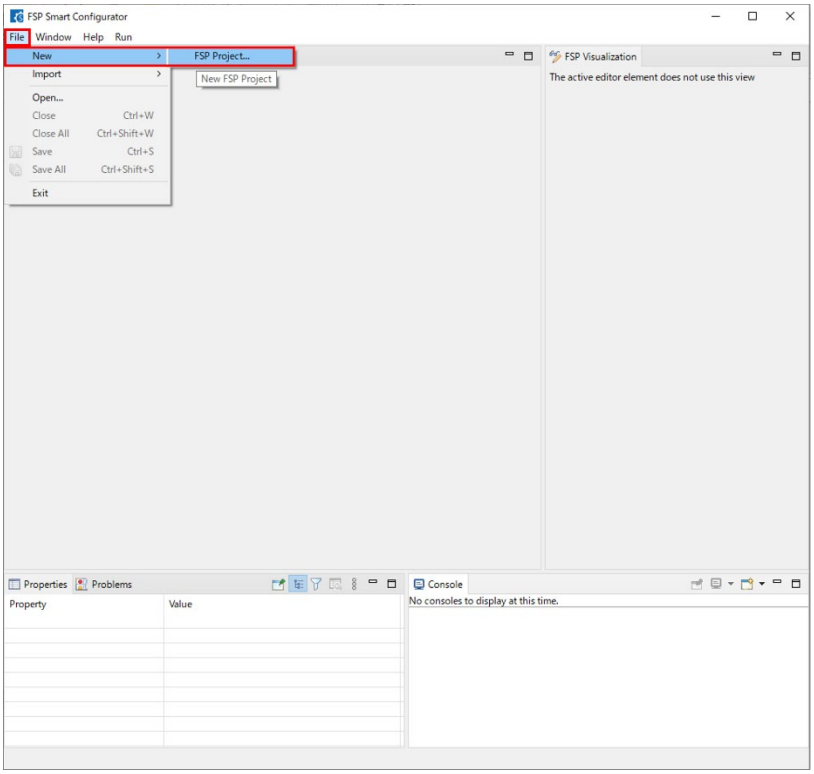


Figure 54 : FSP SC New Project

(Continued on next page)

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

Table 15 FSP SC Newly Created Project Settings (1)

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

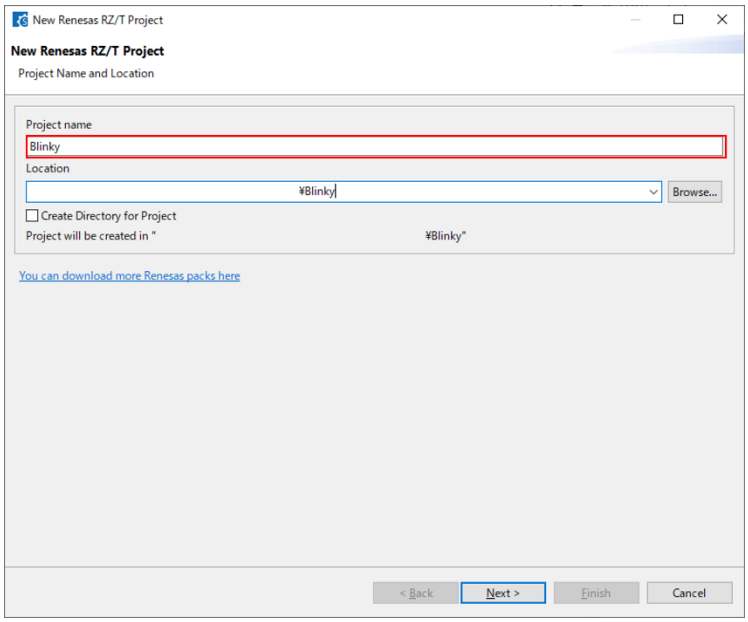


Figure 55 : FSP SC Project Settings

(Continued on next page)

4. Select the **FSP** version.
5. Select the **Board** for your application.
   - You can select an existing RZ/T2, RZ/N2 MPU Evaluation Board or select 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 Evaluation board.
6. (Multicore device ONLY) Select the **Core** from the drop-down list.
7. Select **IDE Project Type**.
   - As the Toolchain, IAR Toolchain for ARM is preselected.
8. Click **Next**.

Table 16 FSP SC Newly Created Project Settings (2)

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

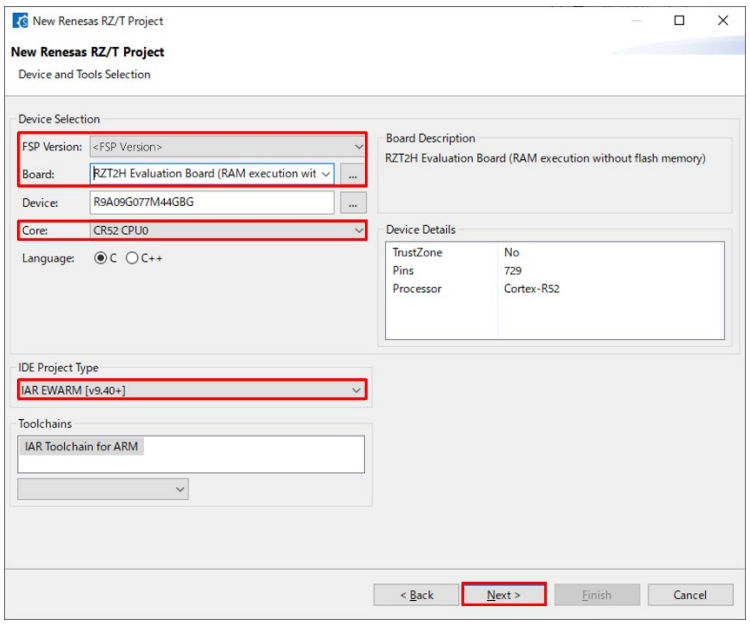


Figure 56 : Target Device and IDE Selections

(Continued on next page)

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

Table 17 FSP SC Newly Created Project Settings (3)

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

Note:

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

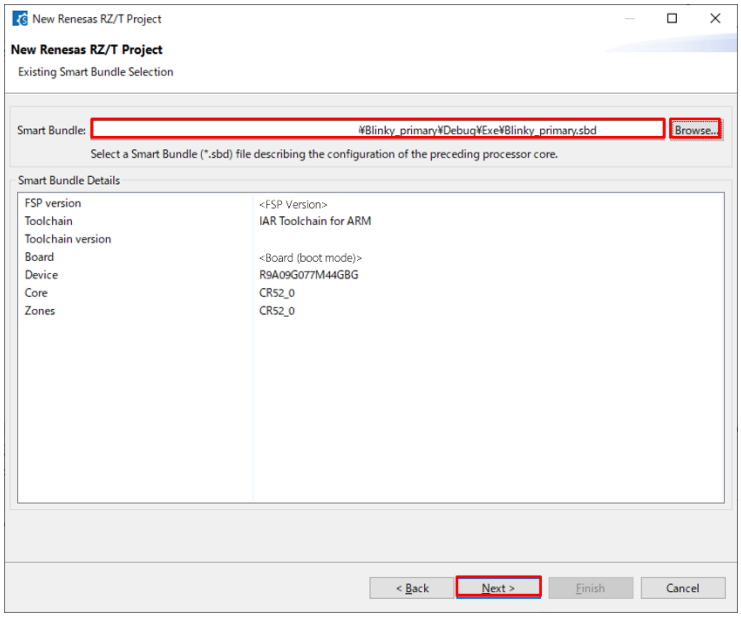


Figure 57 Bundle File Selection

(Continued on next page)

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

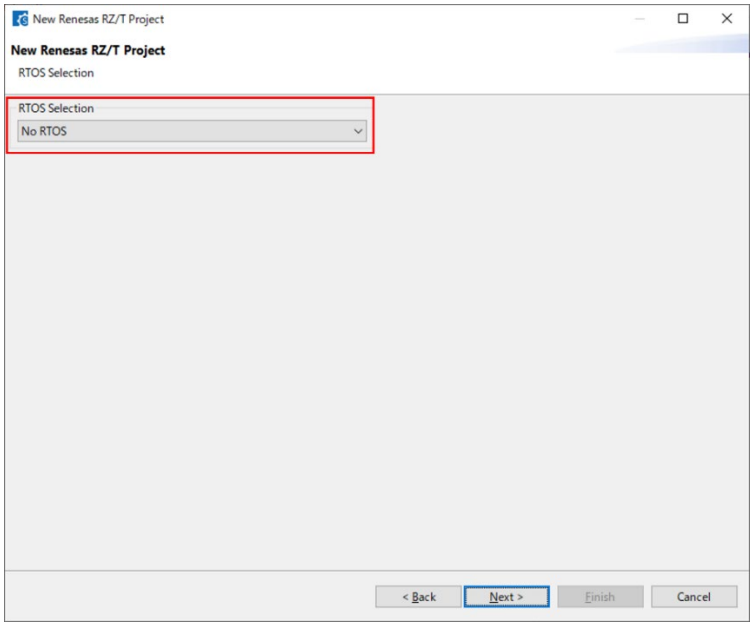


Figure 58 : RTOS Selection

(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.**

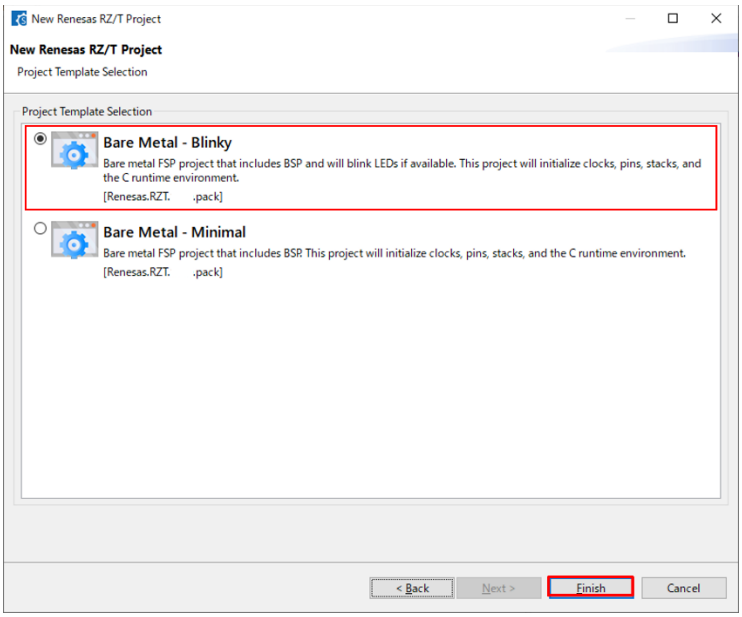


Figure 59 : Template Selection

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

(Continued on next page)

Note for the primary project using CR52 CPU0:

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

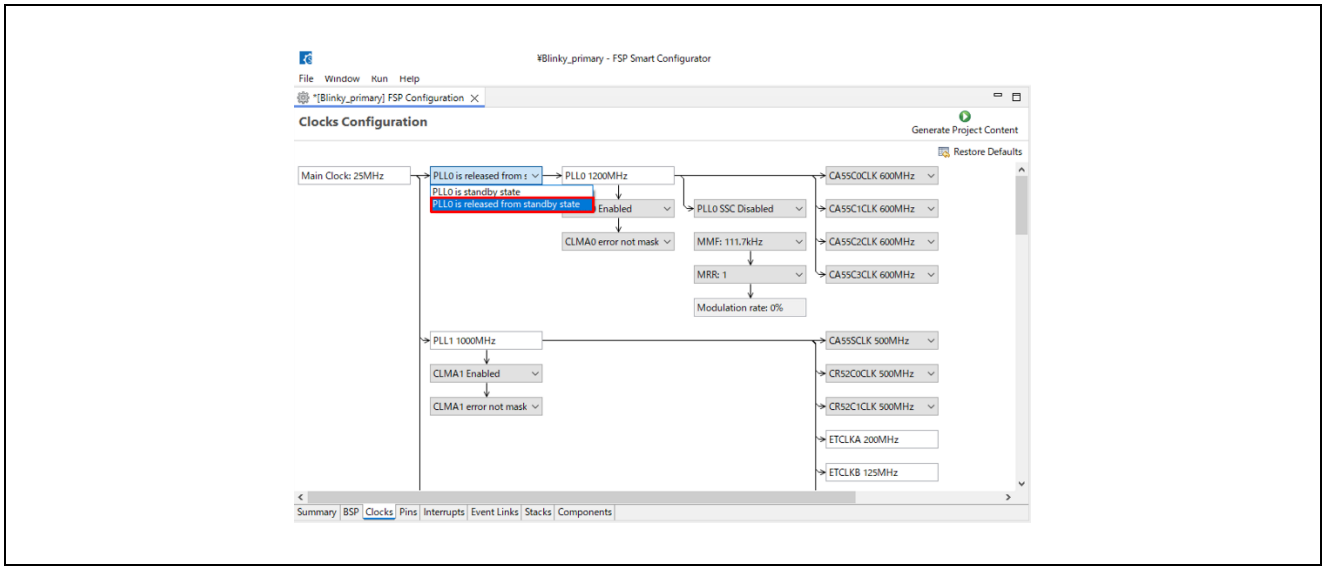


Figure 60 : Enable PLL0 in the Primary Project using CR52 CPU0

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

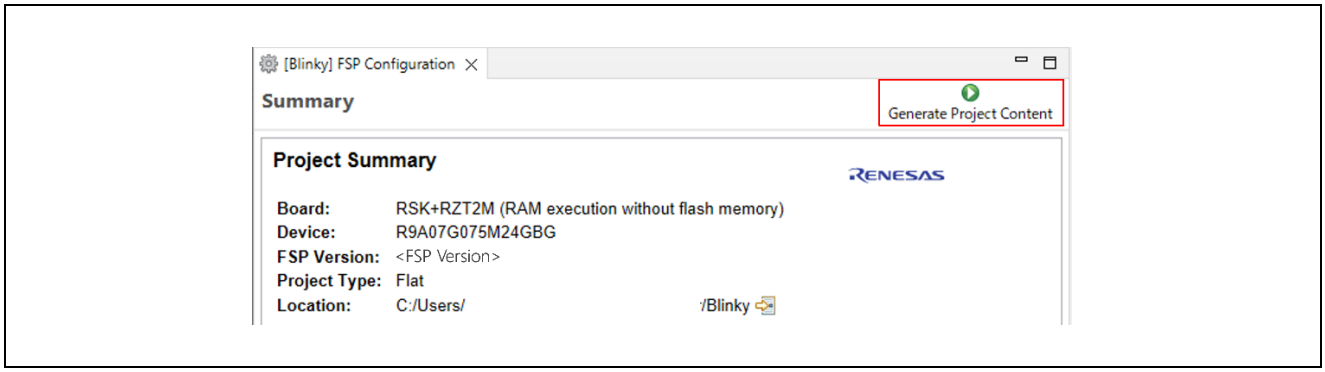


Figure 61 : FSP Project Configuration and Generation

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

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

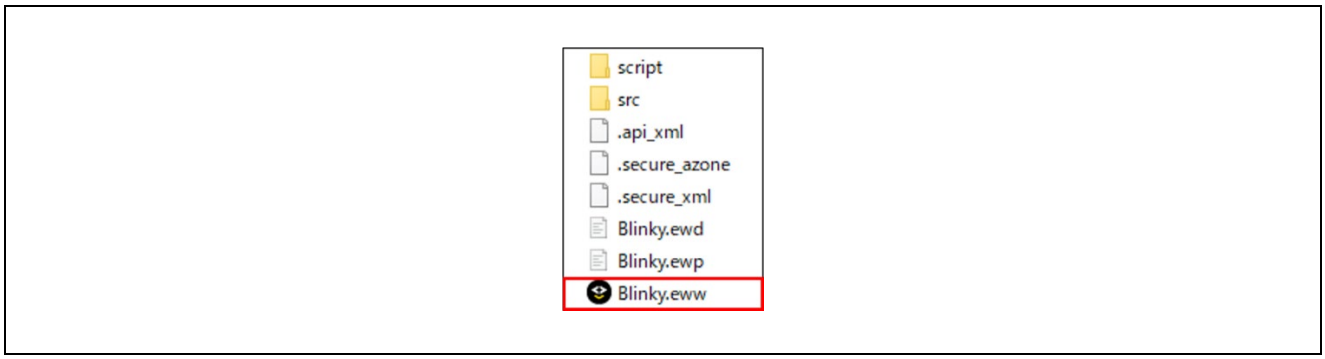


Figure 62 : FSP Project Workspace

5.3.2.1 NOTE: Configure IAR EWARM Project [RZ/T2H and RZ/N2H]

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

Table 18 FSP SC Newly Created Project Debug Settings

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

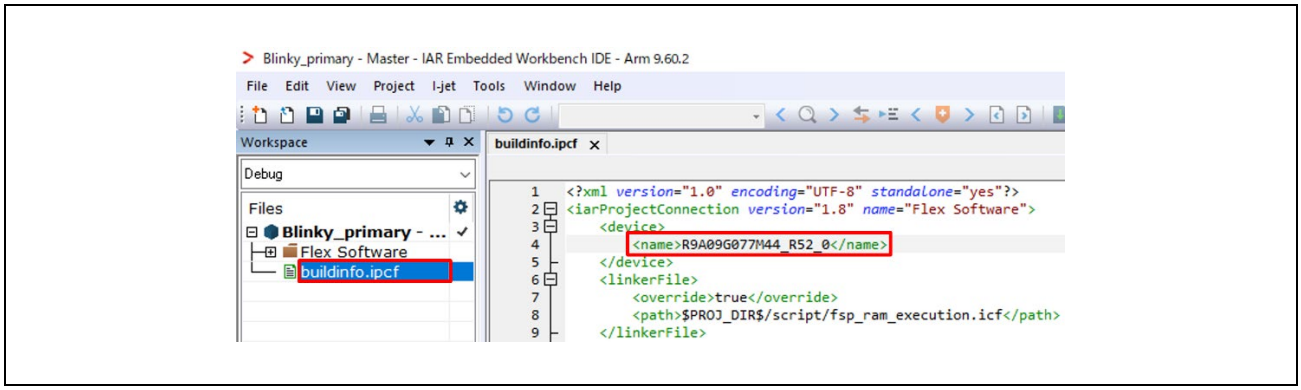


Figure 63 : IAR EWARM Project File (CR52)

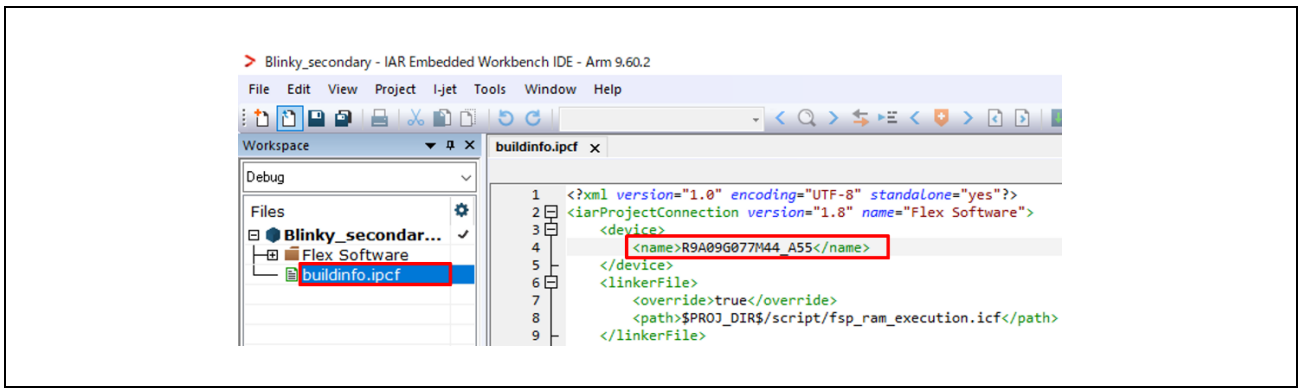


Figure 64 : IAR EWARM Project File (CA55)

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

(Continued on next page)

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

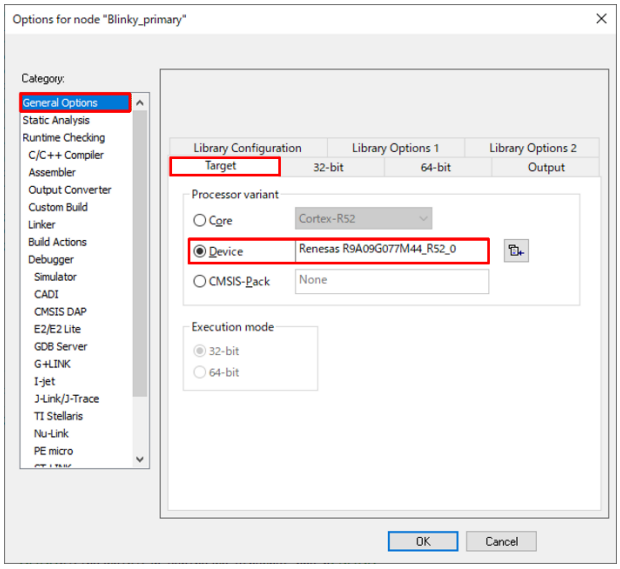


Figure 65 : Project Options – Device (CR52 CPU0)

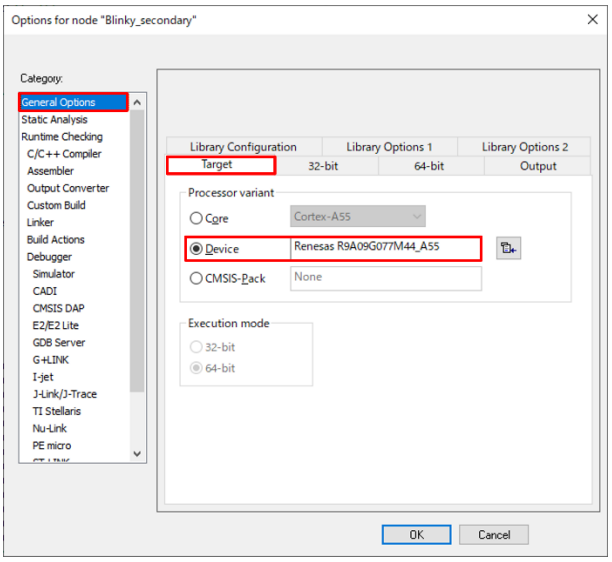


Figure 66 : Project Options – Device (CA55)

(Continued on next page)

Note:

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

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

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

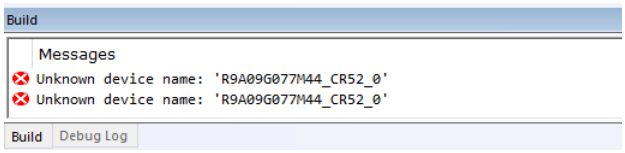


Figure 67 : Error Message after Configuration Change

5.3.3 Build the Project

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

5.3.3.1 Build

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

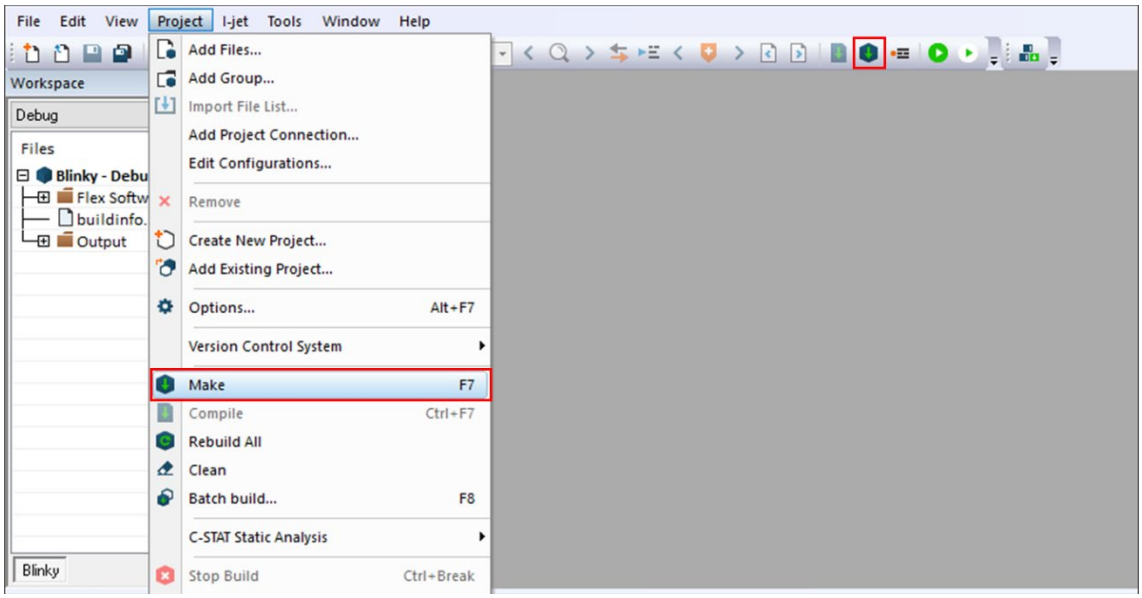


Figure 68 : Make Button

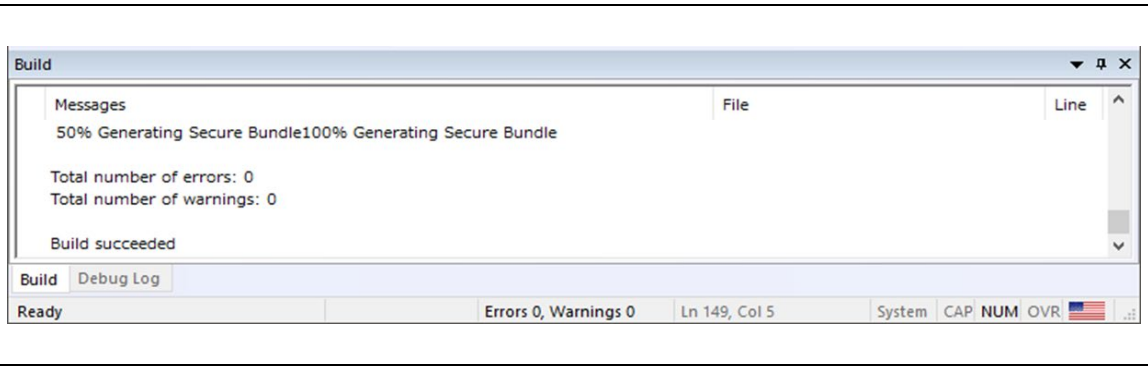


Figure 69 : Build Message Console

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.

5.3.3.2 Build for Multiprocessing

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

(CR52): The core used in the project is CR52.
(CA55): The core used in the project is CA55.

1. Create and build the primary project. (1st build of the primary project)

   Set the following before building:

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

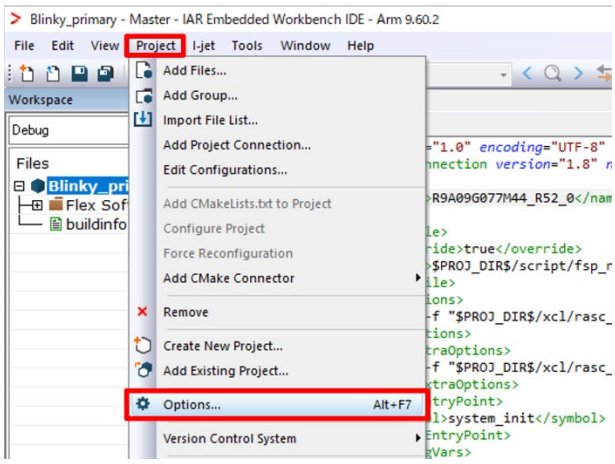


Figure 70 IAR EWARM Project Options

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

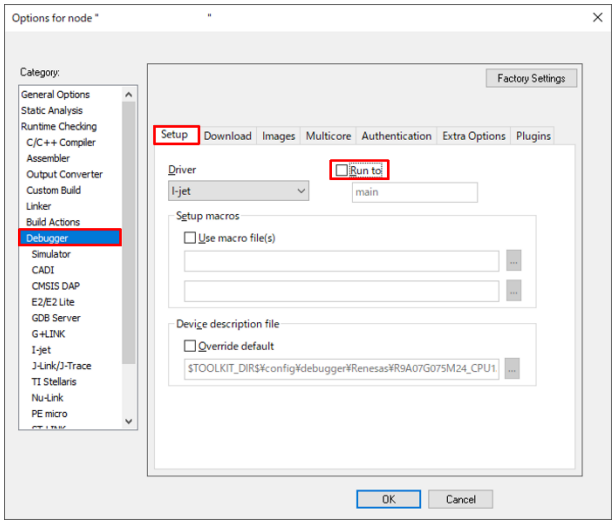


Figure 71 IAR EWARM Project Options for the Primary Project (Run to)

(Continued on next page)

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

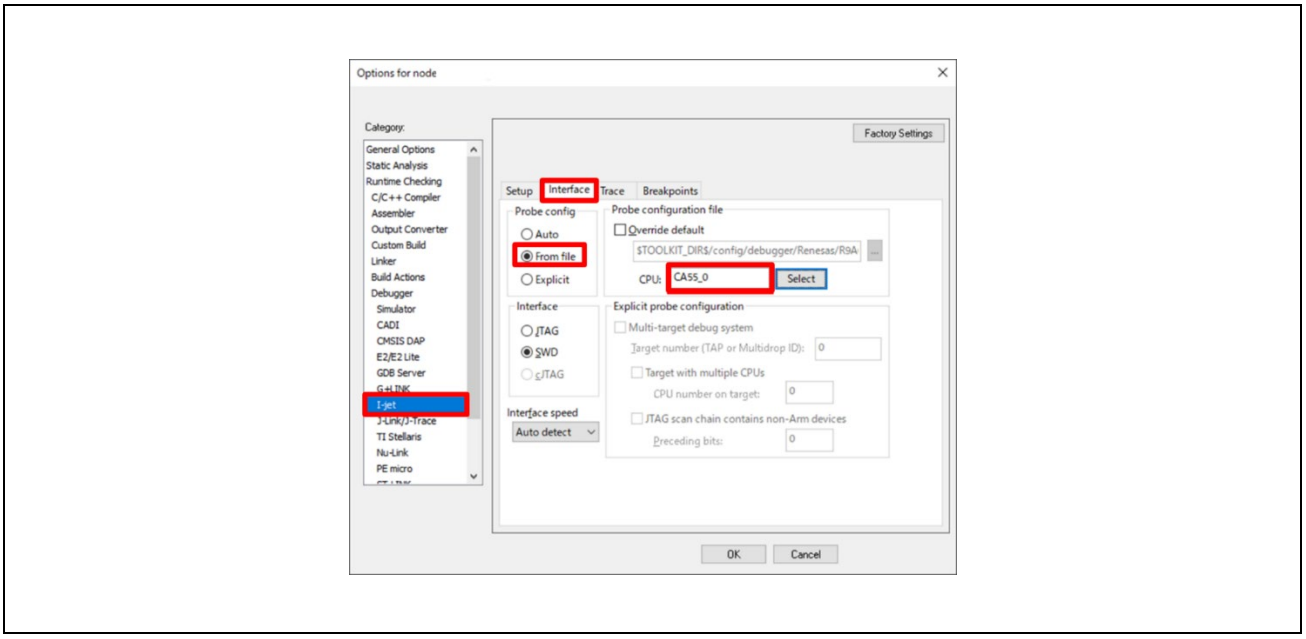


Figure 72 IAR EWARM Project Options for the Primary Project (I-jet Interface)

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

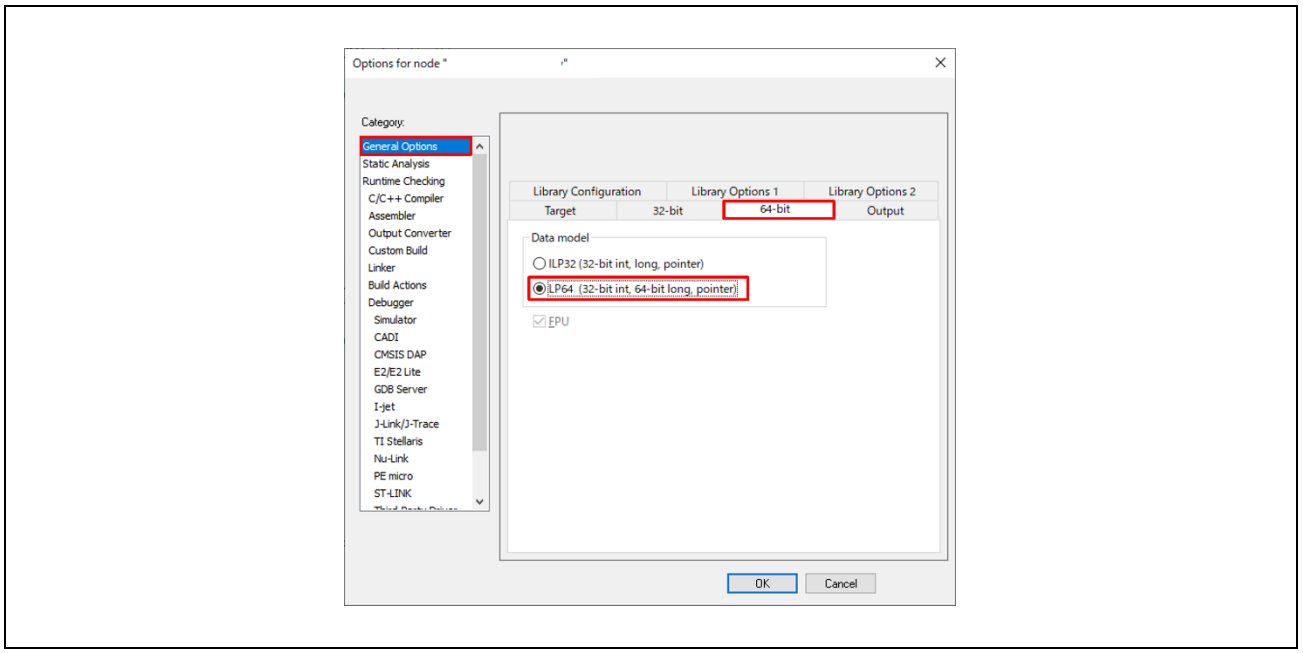


Figure 73 : Project Options – Data Model

v. Proceed to 5.3.3.1 Build.

(Continued on next page)

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

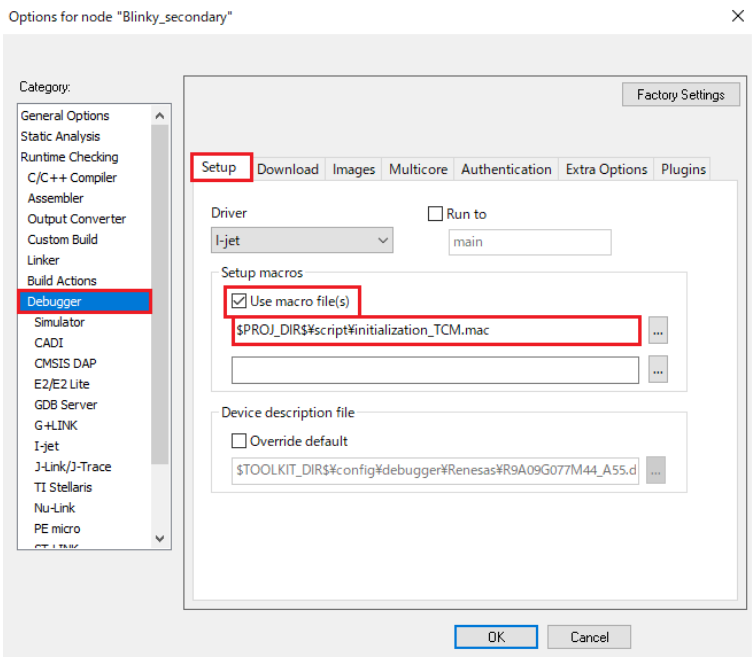


Figure 74 IAR EWARM Project Options for the Secondary Project (Setup Macros)

   iv. Click **Debugger** > **Extra Options** and add "--macro_param cpu1_enable=1" to **Command line options: (one per line)**.

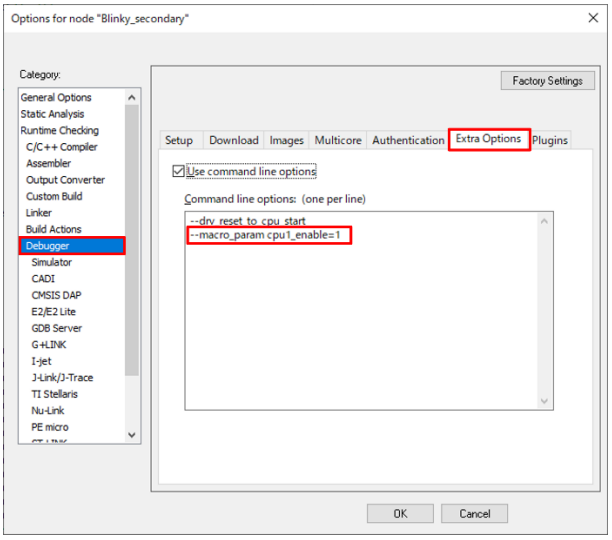


Figure 75 IAR EWARM Project Options for the Secondary Project (Debugger Extra Options)

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

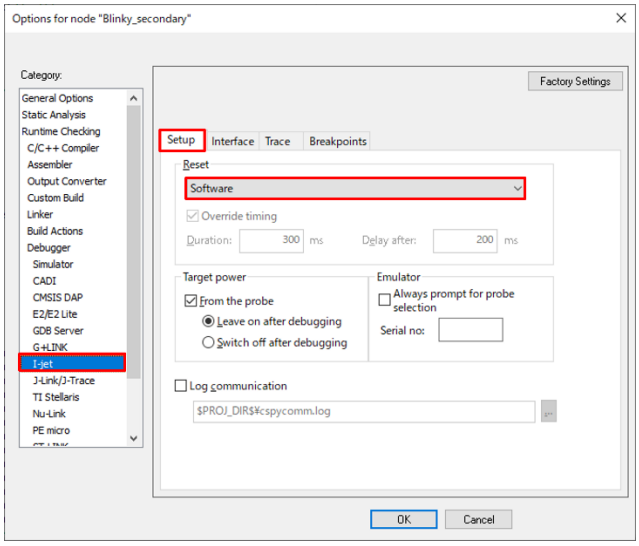


Figure 76 IAR EWARM Project Options for the Secondary Project (Reset)

(Continued on next page)

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

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

viii. Proceed to 5.3.3.1 Build.

ix. Close the secondary project.

3. Build the primary project. (2nd build of the primary project)

No setting is required, proceed to 5.3.3.1 Build.

5.3.4 Download & Debug the Project

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

Note:

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

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

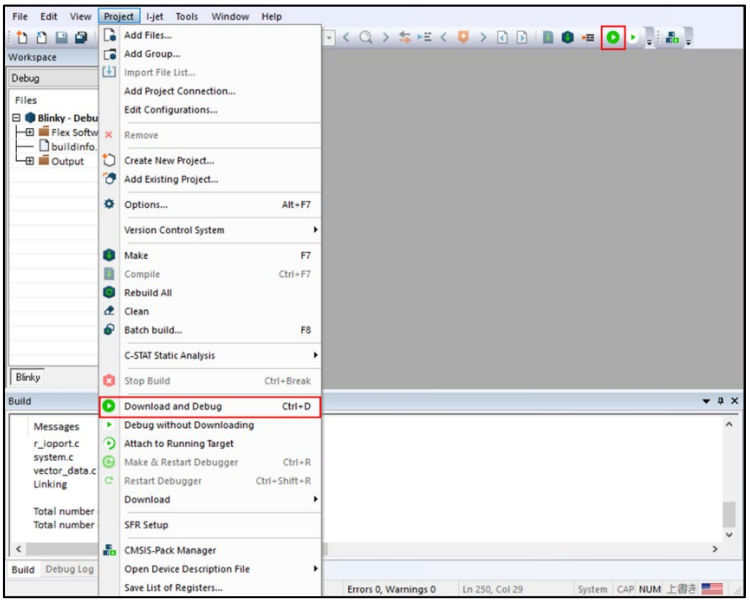


Figure 77 : Download and Debug Button

(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**.

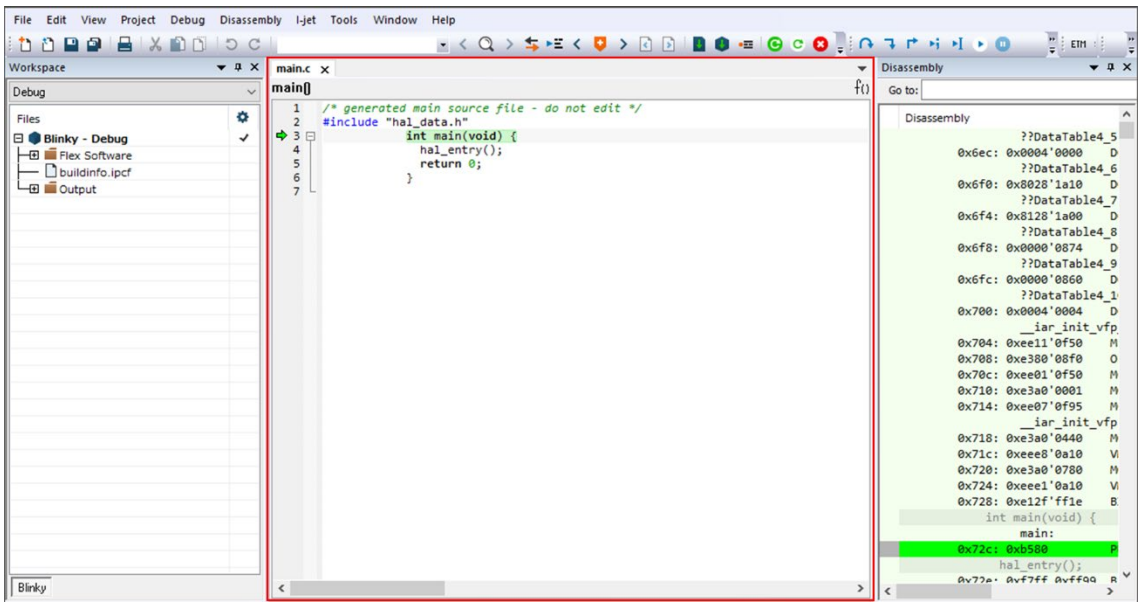


Figure 78 : Starting Debug

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

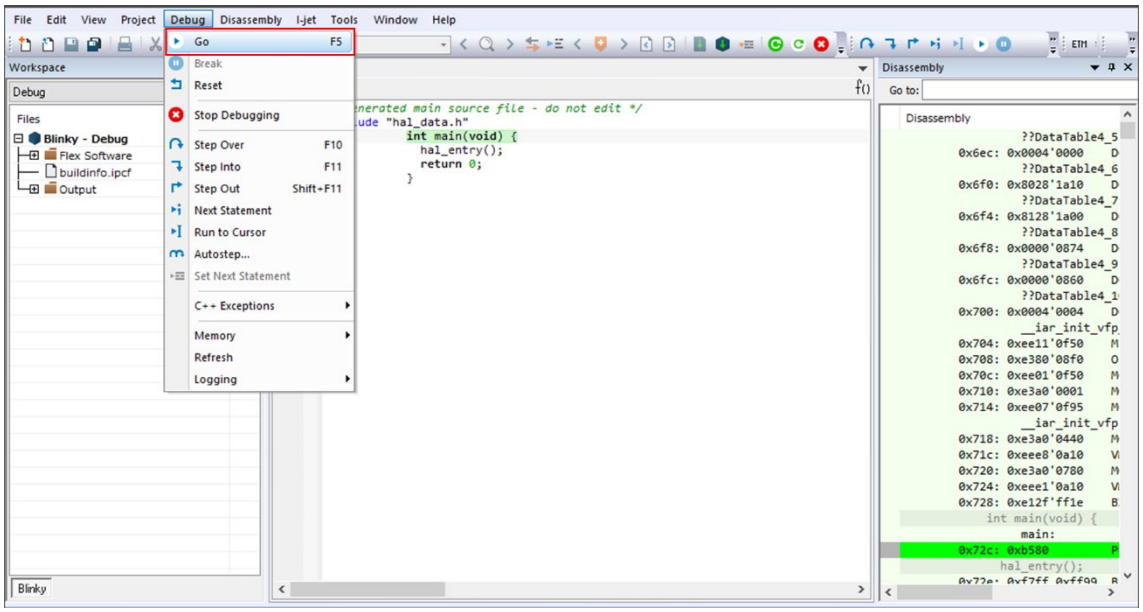


Figure 79 : Go Button

(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.

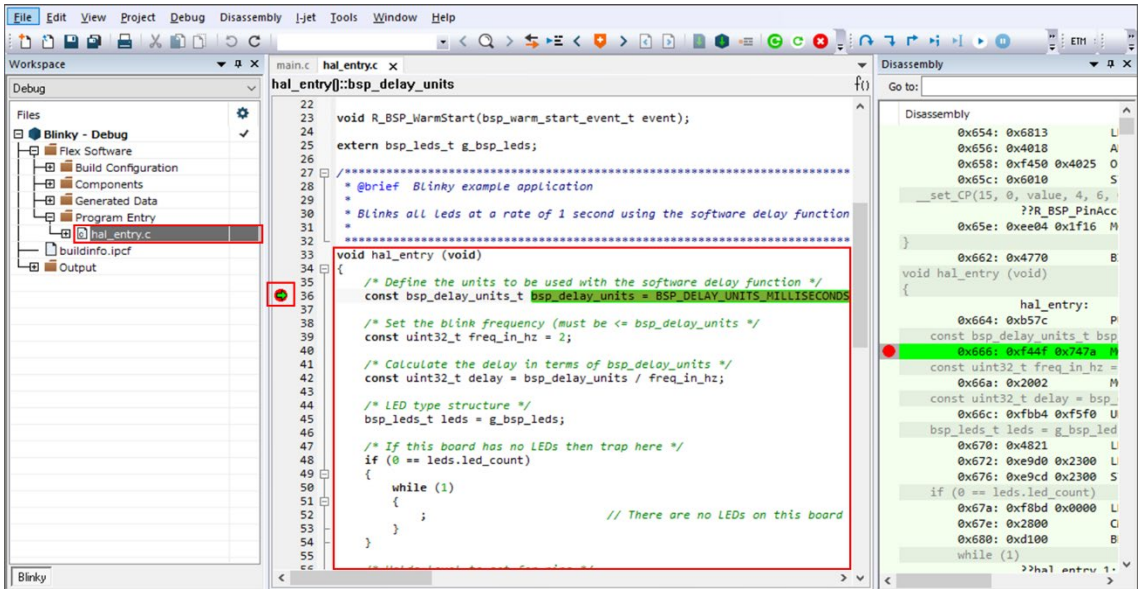


Figure 80 : hal_entry.c and Setting Breakpoint

(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.

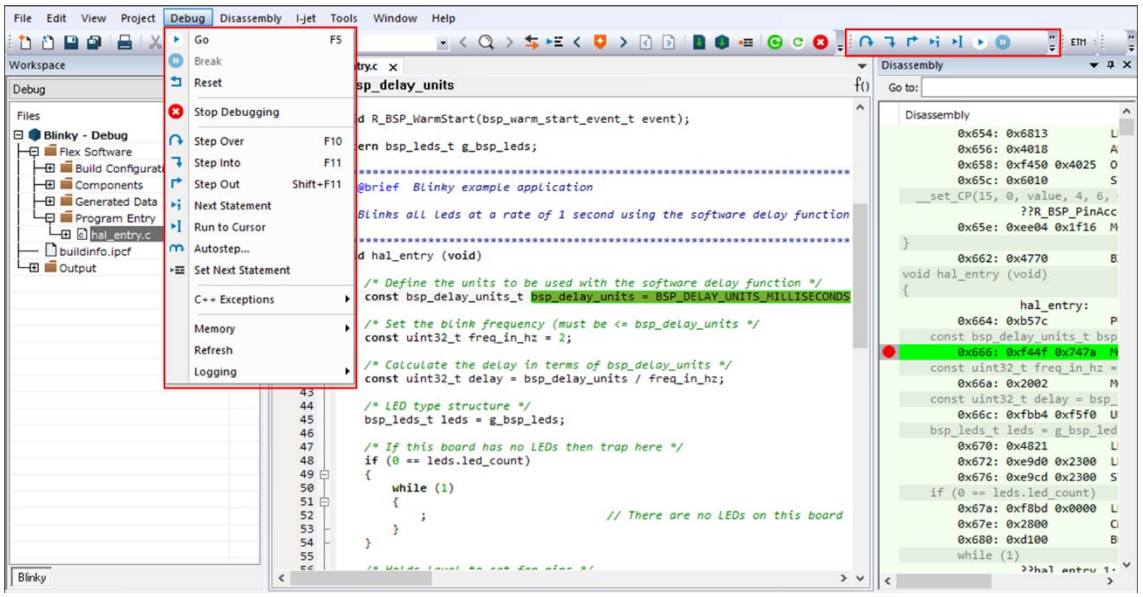


Figure 81 : Debug Menu

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

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

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

Figure 82 IAR EWARM Debugger Pause Icon

(Continued on next page)

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

Figure 83 IAR EWARM Debugger Stop Icon

5.3.5 Debug for Multiprocessing

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

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

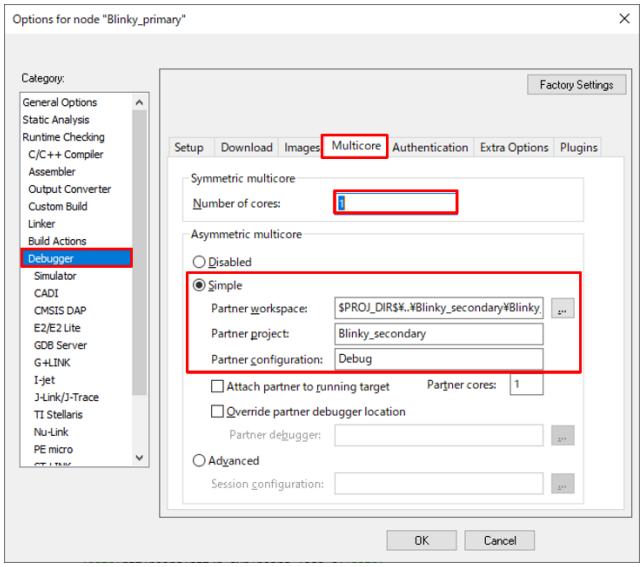


Figure 84 IAR EWARM Project Options for the Primary Project (Multicore)

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

(Continued on next page)

3. Download of the primary project with procedure 5.3.4 Download & Debug the Project as shown in Figure 75.
4. The secondary project is automatically launched. Once the download is completed and the debug is started, the program breaks at the beginning of **system_init** in **startup_core.c**.
5. Run the program of primary project as shown in Figure 77 to copy the binaries of the secondary and subsequent projects to the internal RAM in the primary project. After the primary project reaches **hal_entry** in **main.c**, another core is executed. If the LEDs are blinking, proceed to the next step.
6. The primary project in operation, run the program of secondary project.
7. 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.

When changing the project and debugging it again, refer No. 13 in the Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM.

5.4 Re-configuring Project with FSP SC

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 SC with an existing project.

5.4.1 Launch FSP SC from IAR EWARM

1. Select “Tools -> Configure Tools…”
2. Select “New” and fill in the fields as follows:
   - Menu Text FSP Smart Configurator
   - Command $RASC_EXE_PATH$
   - Argument --compiler IAR configuration.xml
   - Initial Directory $PROJ_DIR$

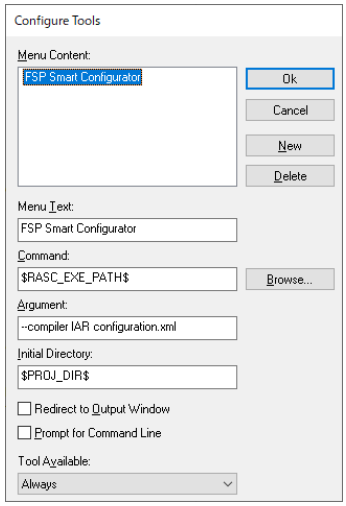


Figure 85 : Settings to Launch FSP SC from IAR EWARM

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 SC installation folder} \ eclipse \ rasc.exe –compiler IAR configuration.xml

5.5 Note when debugging in different workspaces

The project created, built, and debugged in 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 e² 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 e² 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.

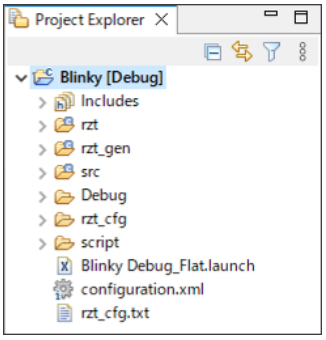


Figure 86 : e² studio Project Configuration File

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 e² studio window. Once selected, you can use the editor to view or modify the configuration settings associated with this project.

Figure 87 : e² 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.

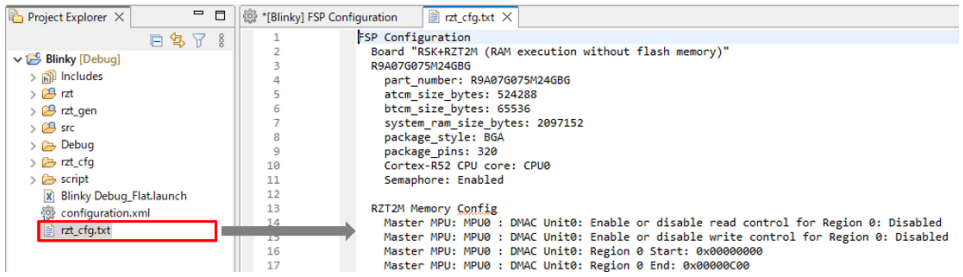


Figure 88 : RZ/T2, RZ/N2 Project Report

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.

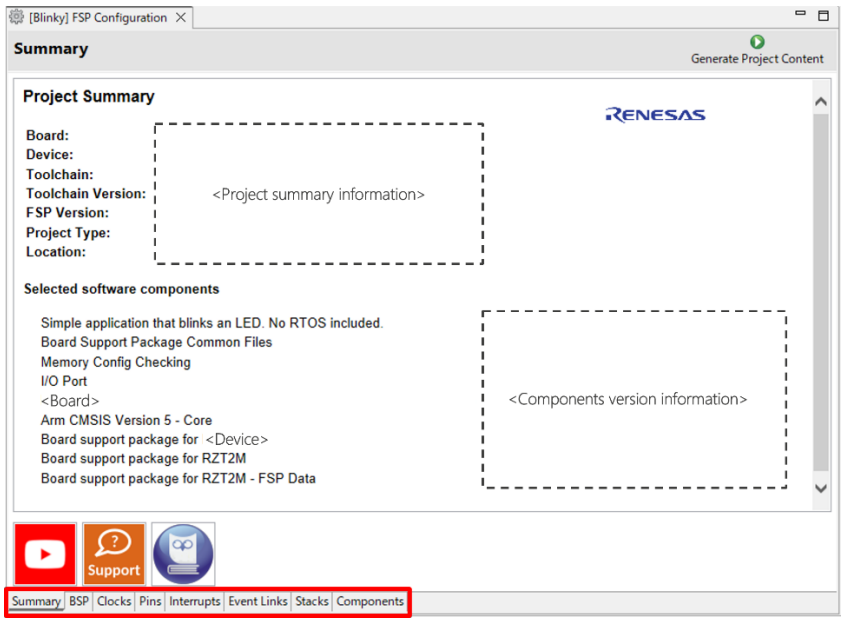


Figure 89 : RZ/T2, RZ/N2 Project Editor Tabs

6.2 Create a Project

6.2.1 Creating a New Project

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

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

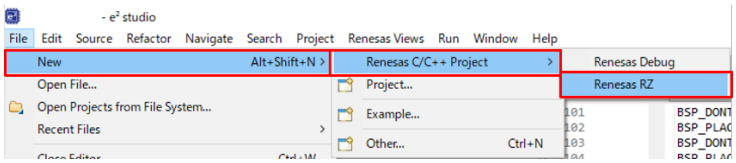


Figure 90 : New RZ/T2, RZ/N2 MPU Project

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

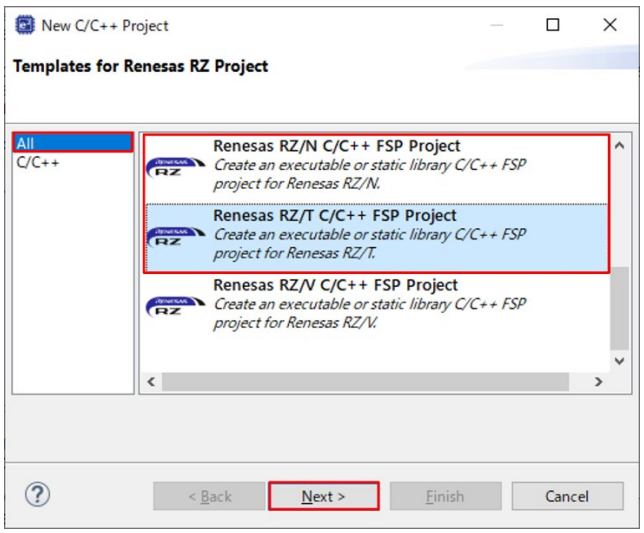


Figure 91 : New Project Templates

(Continued on next page)

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

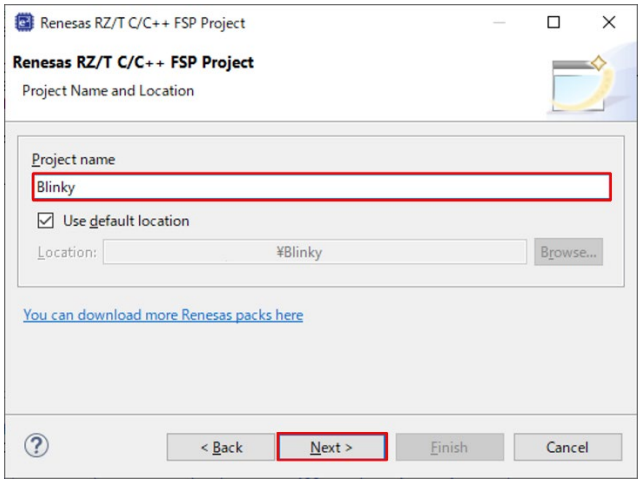


Figure 92 : RZ/T2, RZ/N2 MPU Project Generator (Part 1)

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.

(Continued on next page)

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

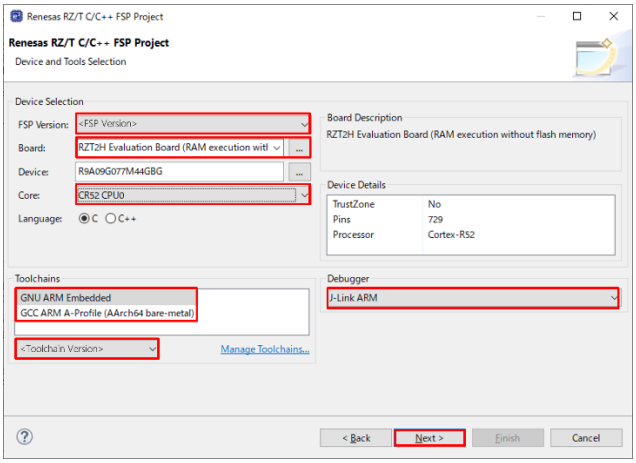


Figure 93 : RZ/T2, RZ/N2 MPU Project Generator (Part 2)

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

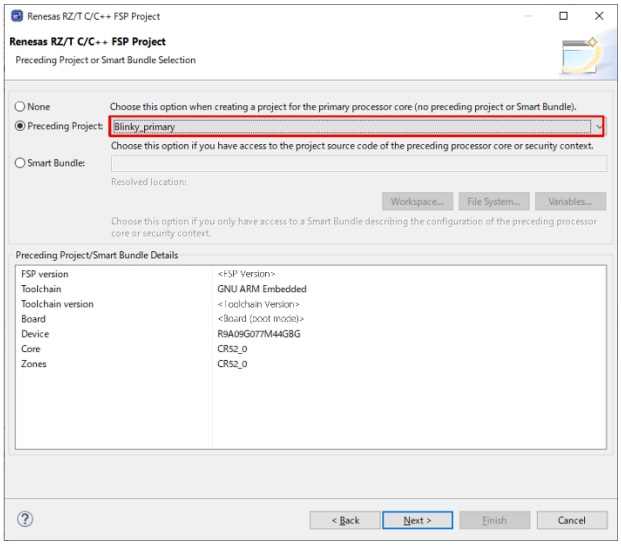


Figure 94 RZ/T2, RZ/N2 MPU Project Generator (Part 3)

6.2.3 Selecting a Project Template

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

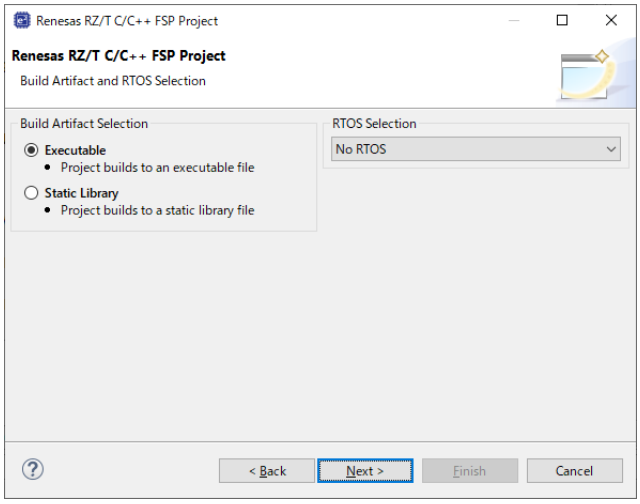


Figure 95 : RZ/T2, RZ/N2 MPU Project Generator (Part 4)

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

Note:

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

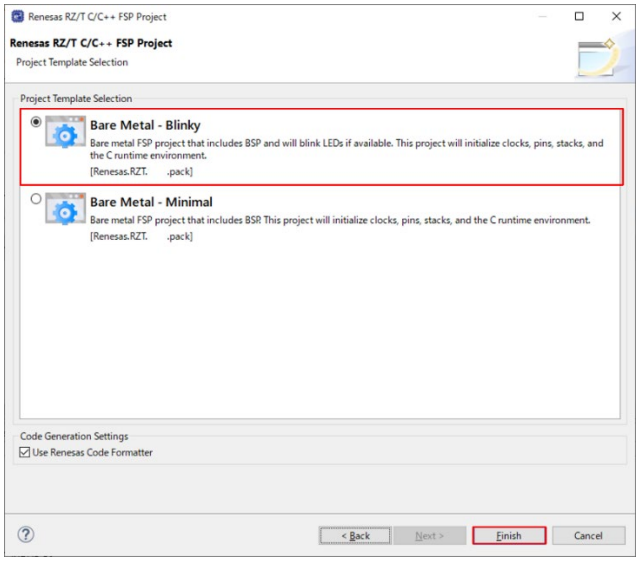


Figure 96 : RZ/T2, RZ/N2 MPU Project Generator (Part 5)

(Continued on next page)

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

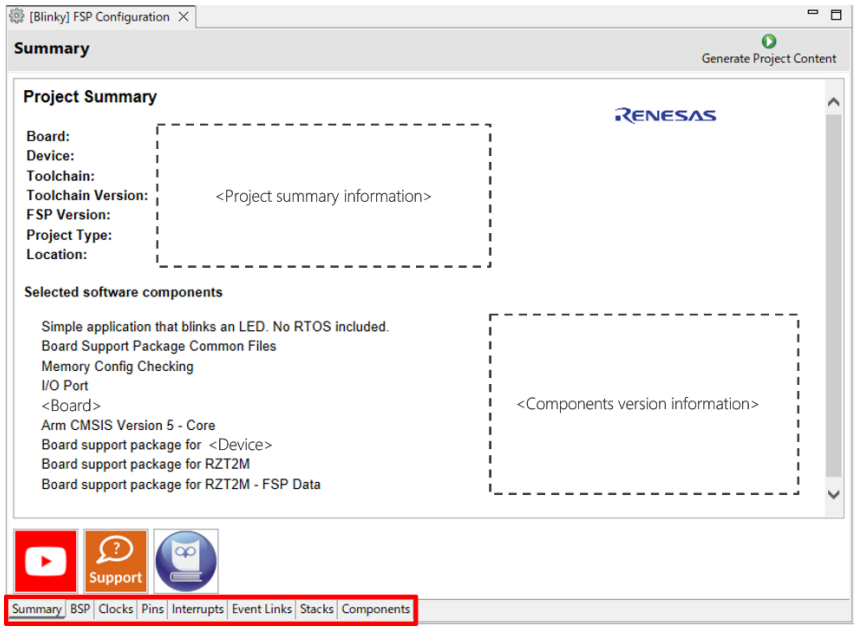


Figure 97 : RZ/T2, RZ/N2 MPU Project Editor and Available Editor Tabs

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.

6.2.4 Duplication of Resources

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

6.3 Configuring a Project

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

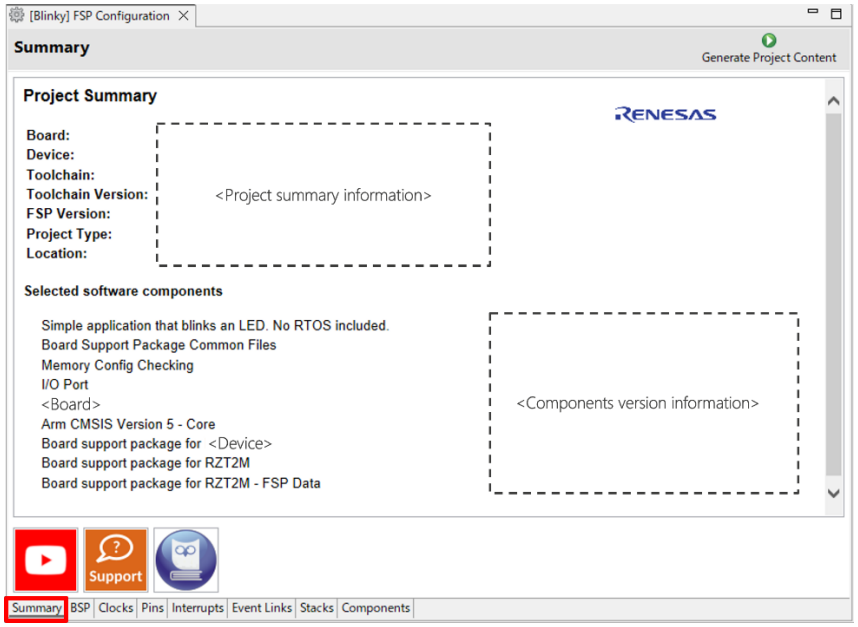


Figure 98 : Configuration Summary Tab

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.

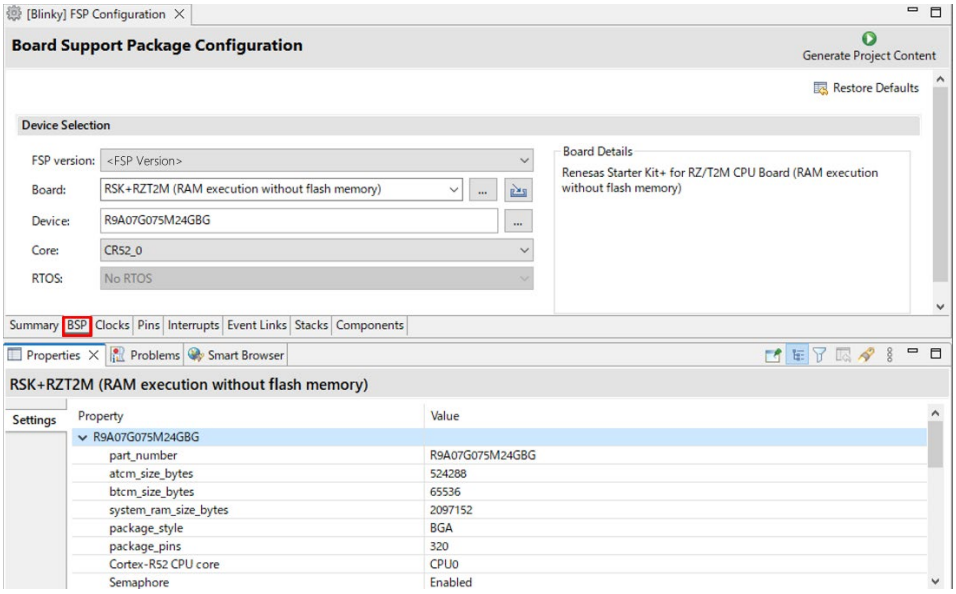


Figure 99 : Configuration BSP Tab

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.

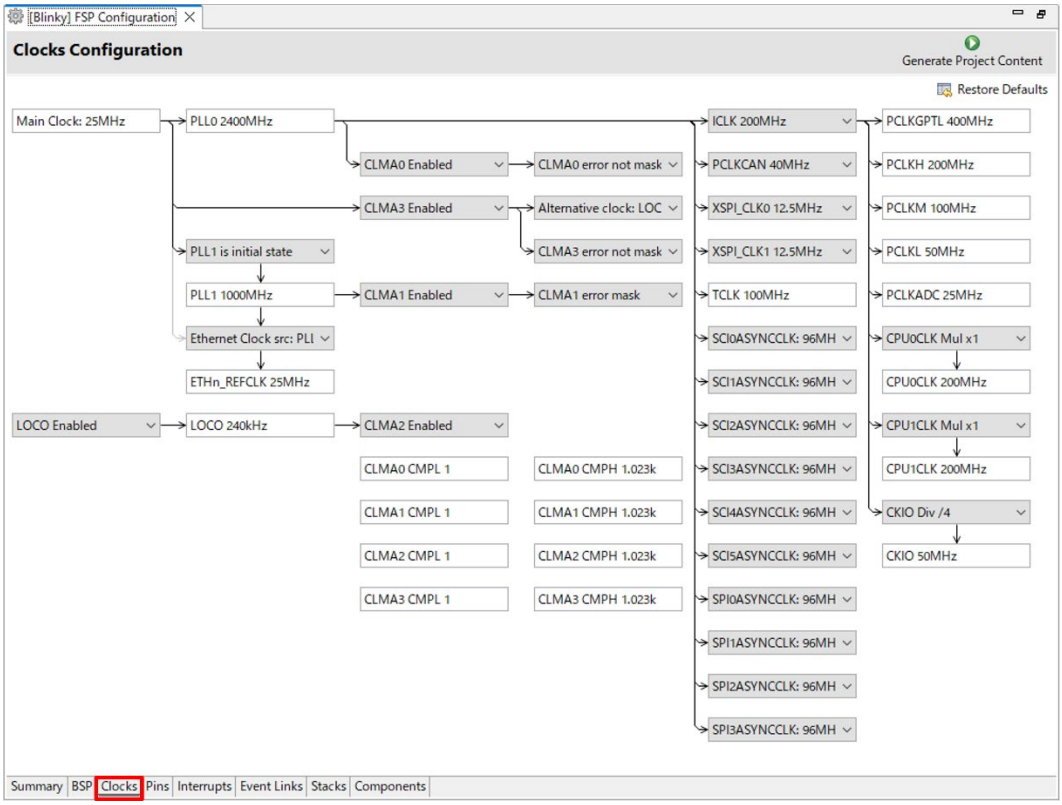


Figure 100 : Configuration Clocks Tab

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.

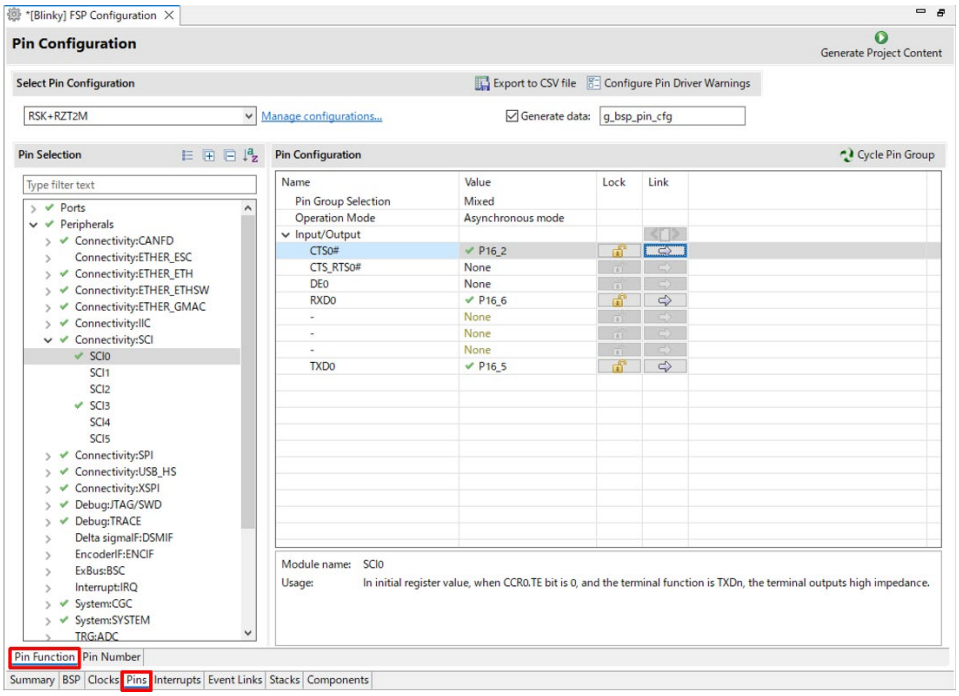


Figure 101: Pin Configuration

(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.

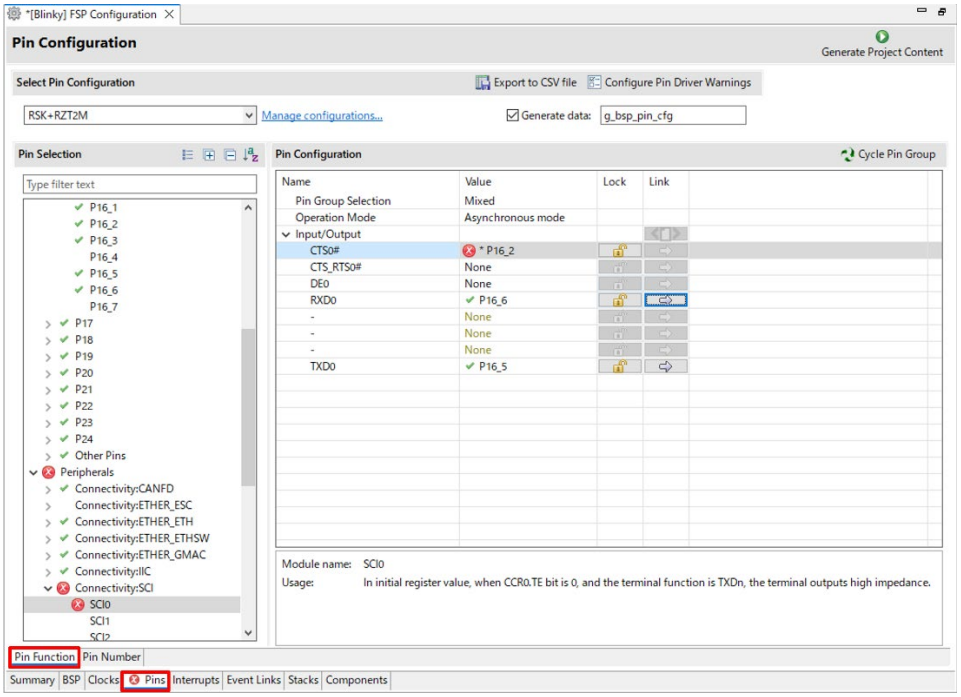


Figure 102: Conflict Checker in Pin Configuration

(Continued on next page)

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

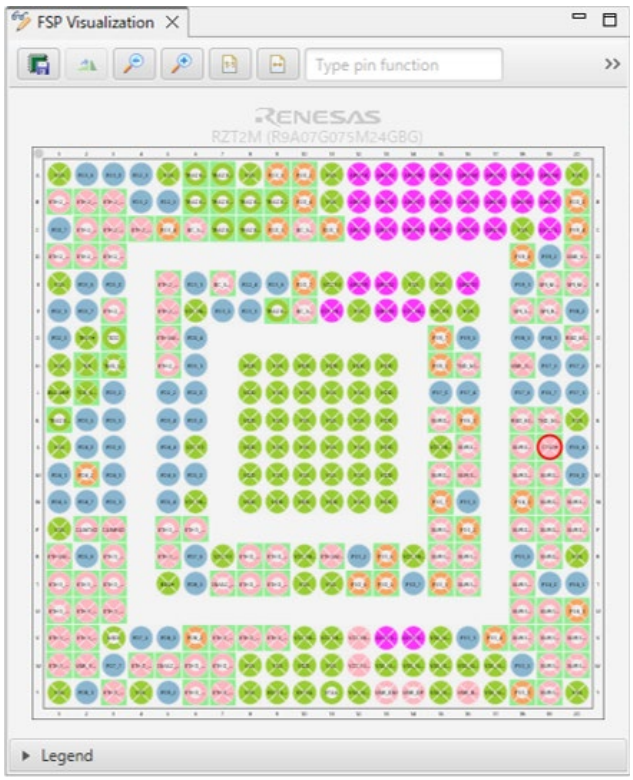


Figure 103: Pin Configurator Package View

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.

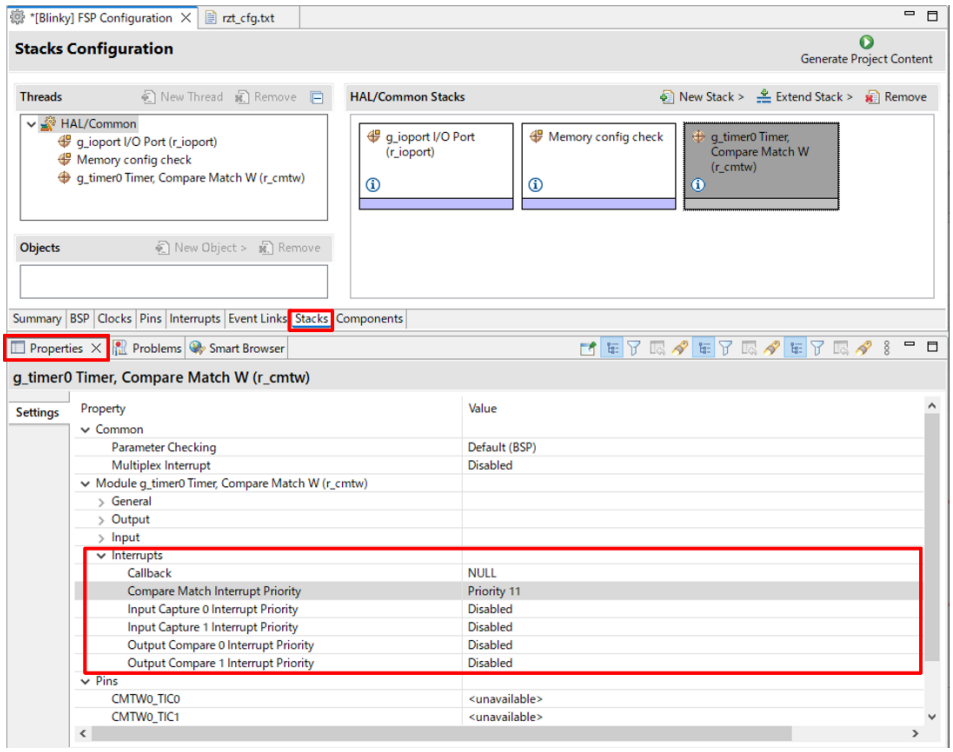


Figure 104 : Configuring Interrupts in the Stacks Tab

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.

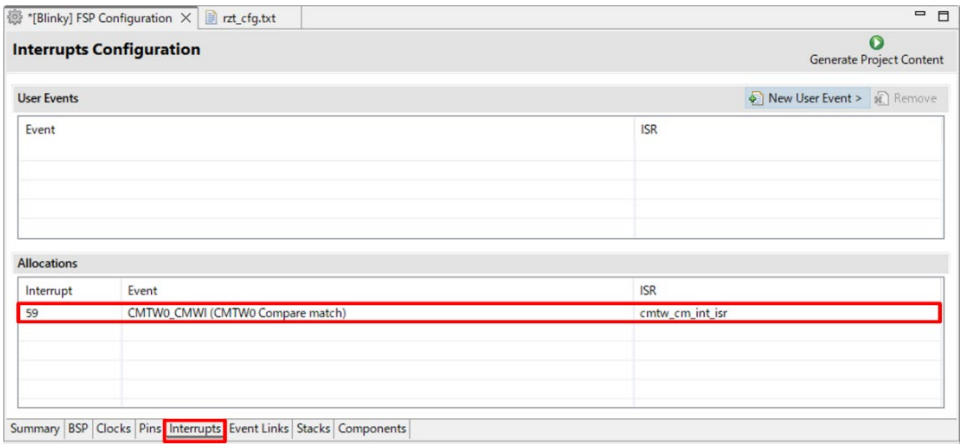


Figure 105 : Configuring Interrupt in Interrupt Tab

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.

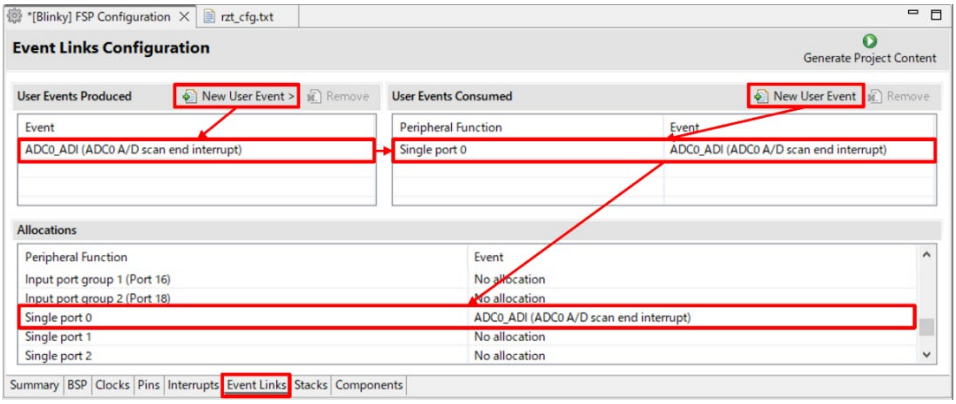


Figure 106 : Viewing Event Links

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**.

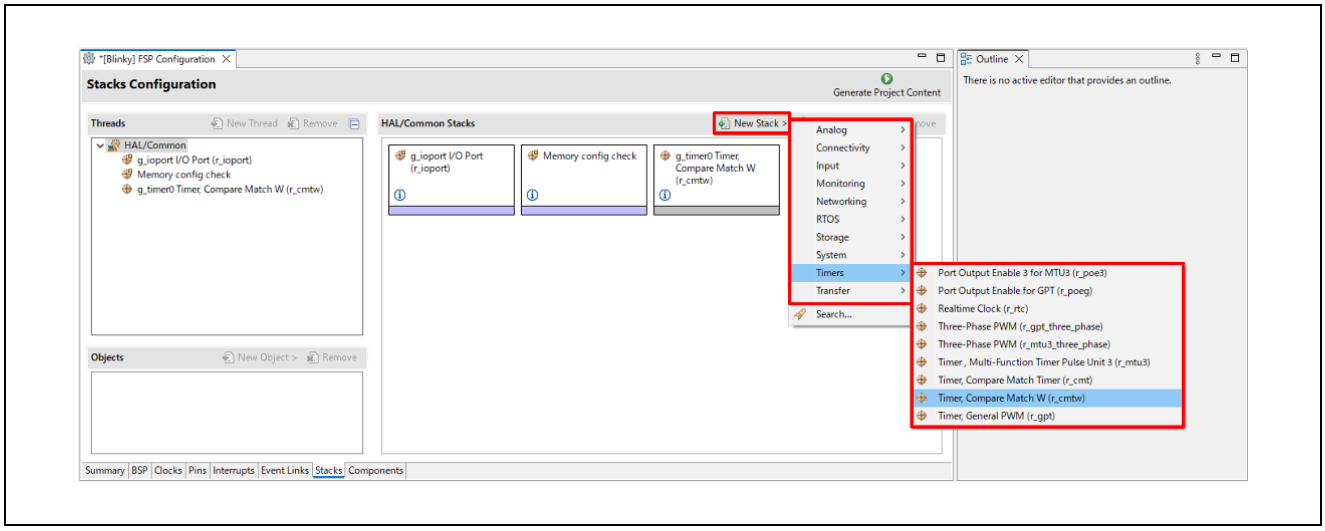


Figure 107 : e² studio Project Configurator – Adding Drivers

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 19 Generate Contents on FSP Configuration

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.

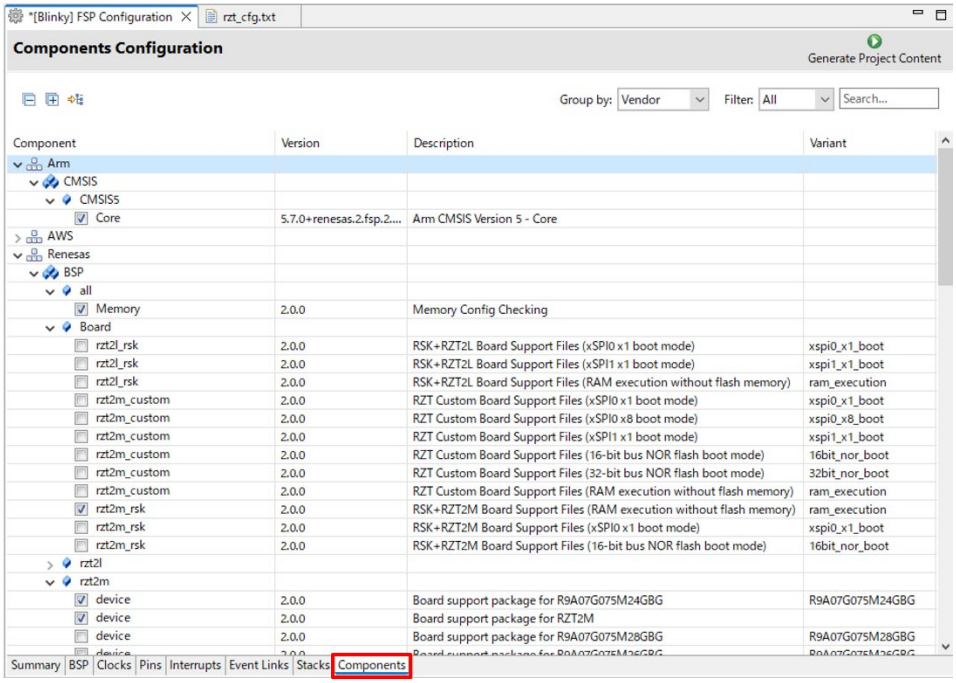


Figure 108 : Components Tab

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

  FSP Configuration on 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 on 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 20 List of Known Issues

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
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

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
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
18	An incorrect value is set to a pin select value for MTU0-B/MTU6/MTU7 as MTU3 output pin.		✓					FSP Modules, POE3
19	Build error when using DSMIFn_ERR as an additional trigger for “r_poe3” module.		✓					FSP Modules, POE3
20	Control setting values for MTU3 output pins in Stacks tab of FSP Configuration are set to the incorrect pin.	✓	✓	✓				FSP Configuration, Stacks
21	A bug that prevented the setup of PLL1.					✓		FSP Configuration, Clocks
22	A section cannot be copied successfully when its size is not a multiple of the alignment size.					✓		FSP Modules, BSP
23	Initial values of data placed in some sections were overwritten with 0.					✓		FSP Modules, BSP
24	Some sections were not initialized in the flash boot project.					✓		FSP Modules, BSP

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
25	DSMIF 0/1 error 1 trigger macros are not defined.		✓					FSP Modules, POEG
26	DSMIF 0/1 error 1 status macros are not defined.		✓					FSP Modules, POEG
27	Missing constraint for DSMIF error trigger in channel 1 and channel 2.	✓	✓	✓	✓			FSP Modules, POEG
28	FreeRTOS+FAT format process is not executed correctly.	✓	✓	✓	✓	✓	✓	FSP Modules, FreeRTOS+FAT
29	Caution when specifying program placement in linker scripts.				✓		✓	Others, Linker script
30	In the secondary project for multiprocessing, no error occurs when there is a conflict in a resource used with the preceding project.				✓		✓	FSP Configuration, Stacks
31	Errors occur when setting ELC in r_gpt module.				✓		✓	FSP Configuration, Stacks
32	CR52 CPU1 of RZ/T2H and RZ/N2H is implemented to start programs from System SRAM instead of CPU1 ATCM.				✓		✓	Others, Linker script
33	No Error Occurs when entering out-of-range values for window parameters in r_pcie_ep and r_pcie_rc module configurations.				✓		✓	FSP Configuration, Stacks
34	Address space of DDR and PCIE cannot be used in the secondary (or later) projects with flash boot mode.				✓		✓	Others, Address space
35	r_gmac_b module cannot use zero-copy mode.				✓		✓	FSP Modules, GMAC
36	r_adc module does not support the calibration function.				✓			FSP Modules, ADC
37	The USB driver for CA55 project does not work.				✓		✓	FSP Modules, USB
38	No error returns when entering the virtual addresses that cannot be translated to physical addresses as arguments.				✓		✓	FSP Modules, xSPI_OSPI, xSPI_QSPI, DMAC
39	The CA55 project with noncache sections aborts when debugging with flash boot mode on IAR EWARM.				✓		✓	FSP Modules, BSP
40	When changing the duty setting in r_gpt module, there is a possibility the duty may unintentionally become 100%.	✓	✓	✓	✓	✓	✓	FSP Modules, GPT

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
41	CPU registers save and restore process cannot be performed correctly in FIQ_Handler for CA55 projects.				✓		✓	FSP Modules, BSP, FreeRTOS
42	MTU3 callback does not occur as expectation.				✓		✓	FSP Modules, MTU3
43	An undefined error of r_gpt module occurs when building a project.				✓		✓	FSP Modules, GPT
44	Pin names according to unit and channel numbers are not displayed in r_gpt module configurations.	✓	✓	✓	✓	✓	✓	FSP Configuration, Stacks
45	Using R_GPT_DutyCycleSet() with option both pins A and B cannot work properly.	✓	✓	✓	✓			FSP Modules, GPT
46	Parameter checking of R_ETHER_SELECTOR_Open() is not working properly.	✓	✓	✓	✓	✓	✓	FSP Modules, ETHER_SELECTOR
47	Parameter checking feature of R_GMAC_CallbackSet() is not working.	✓	✓	✓	✓	✓	✓	FSP Modules, ETHER_GMAC
48	r_usb_hhid module is not working properly.	✓	✓	✓	✓			FSP Modules, USB_HHID

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

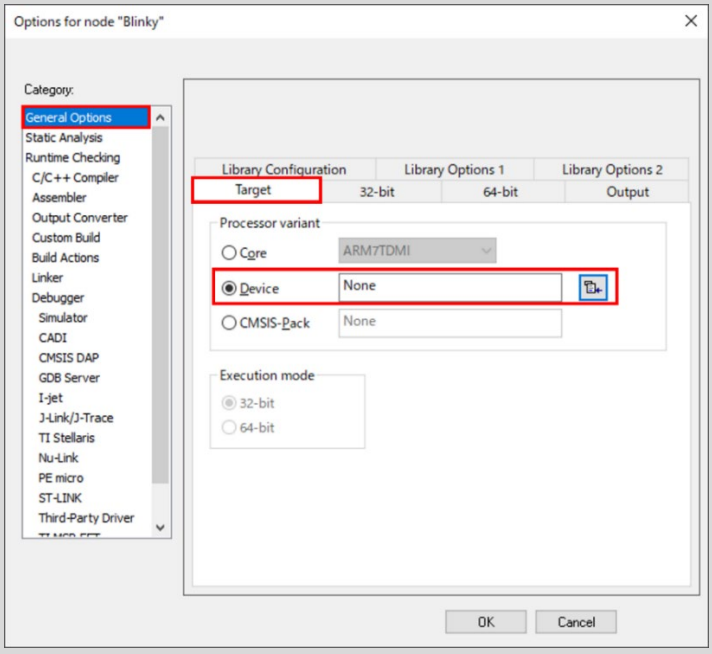
Title	**“Edge” can be selected as Transfer End Interrupt Detect Type in “r_dmac”, but it cannot be used.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**FSP Configuration, Stacks**
Description	“Edge” of interrupt detect type is not available due to a change in hardware specifications.
Workaround	Please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

(Continued on next page)

No. 3 Resolved

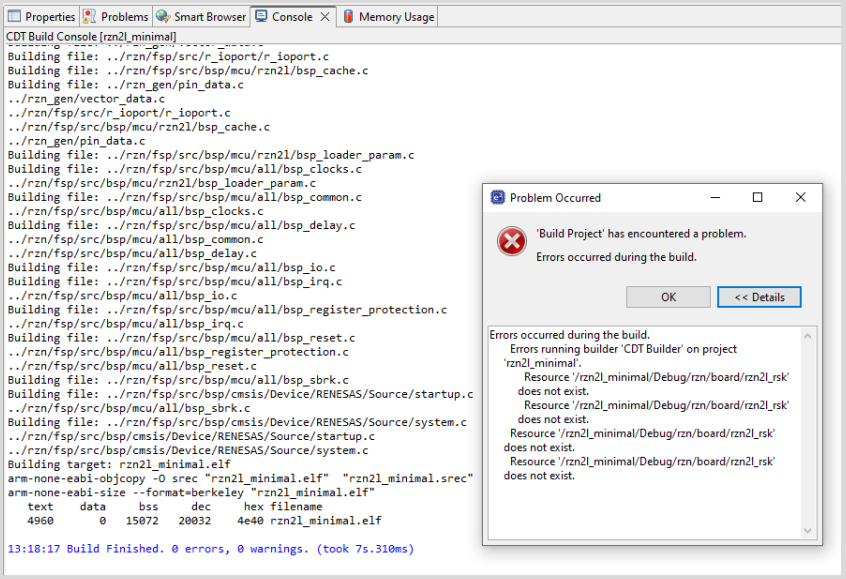
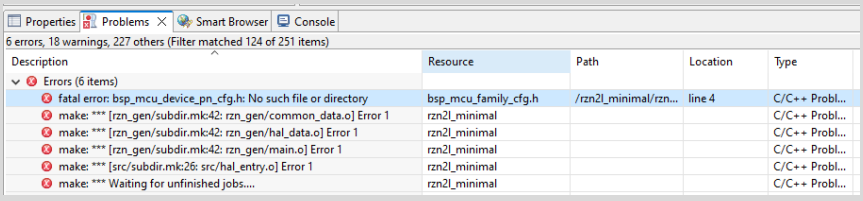
Title	**When the “Device” or “Board” selection in BSP tab is changed, the BSP properties are sometimes configured to incorrect configuration.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**FSP Configuration, BSP**
Description	When the “Device” or “Board” selection in BSP tab 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

No. 4 Resolved

Title	**(FSP SC ONLY) Device name is not output correctly depending on the selected device.**
Target	**RZ/T2M, RZ/T2ME**
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 

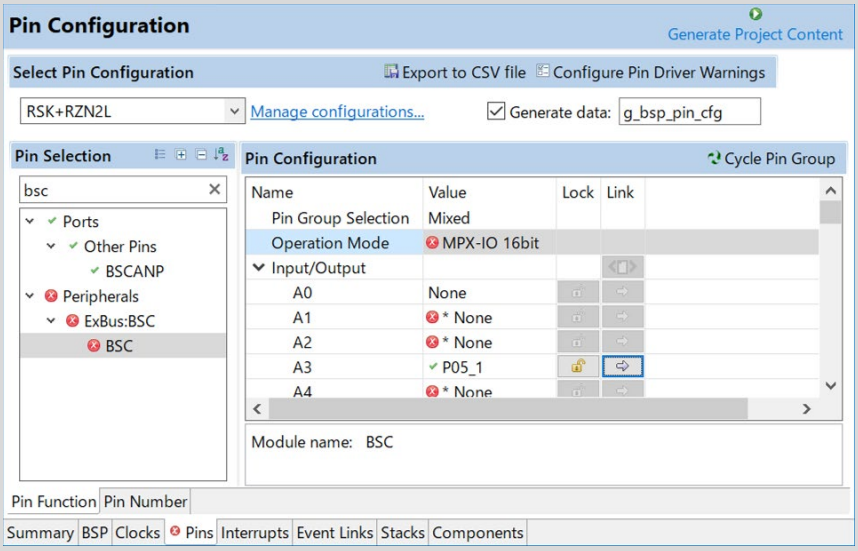
(Continued on next page)

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.

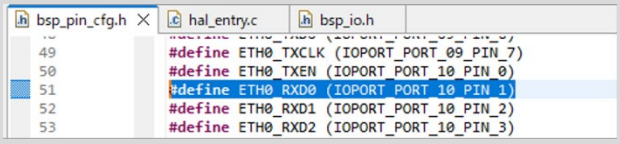
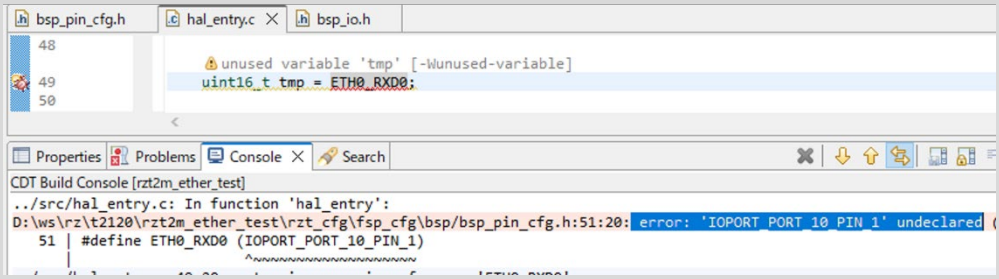
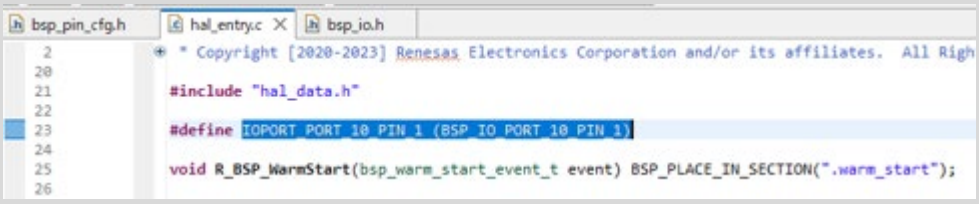
(Continued on next page)

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.

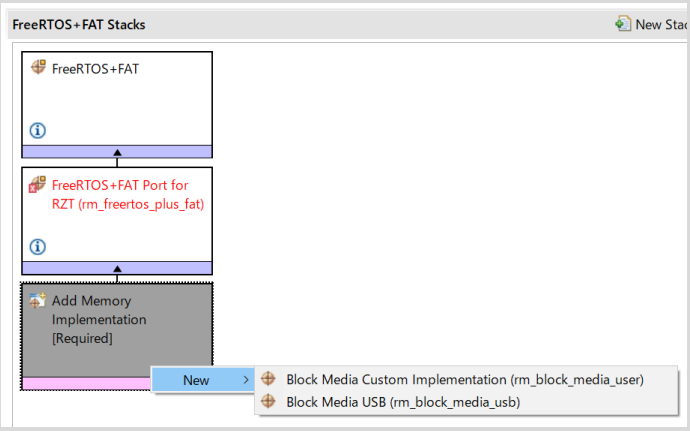
(Continued on next page)

No. 12 Resolved

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 Resolved

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/T2ME, 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 Resolved for RZ/T series devices in RZT FSP v3.0.0

Issue	**The second argument of "r_mtu3" APIs do not match with common API.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
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.

(Continued on next page)

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, RZ/T2ME**
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 Resolved

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.

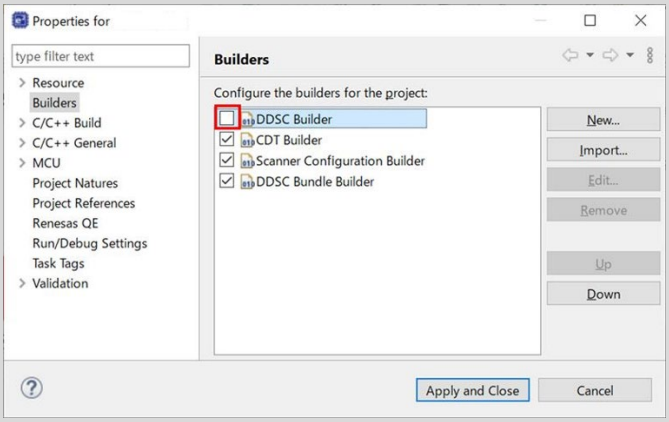
No. 17 Resolved

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.

(Continued on next page)

No. 18 Resolved

Issue	**An incorrect value is set to a pin select value for MTU0-B/MTU6/MTU7 as MTU3 output pin.**
Target	**RZ/T2L**
Category	**FSP Modules, POE3**
Description	Pin select value of MTU3 output pins used in FSP do not match with User's Manual: Hardware. Therefore, what you want to set up is not correctly described in the generated file of FSP Configuration.

Workaround	When using "r_poe3" and MTU0-B/MTU6/MTU7 as MTU3 output pin, please follow these four steps:
	1. Add "Port Output Enable 3 for MTU3 (r_poe3)" on Stacks tab of FSP Configuration.
	2. Click Generate Project Content button and "r_poe3" code is generated.
	3. Disable code generating function.
	After this setting, the code cannot be generated.
	[For e^2 studio Smart Configurator]
	Use the following settings to suppress the code generating operation.
	If this setting is missed, code generating operation is automatically executed at clean build, and the changes made in step 4 revert to the original.
	a. Uncheck "Project Properties > Builders > DDSC Builder"
	 
	
	b. When unchecking it, the following message appears: click OK.
	 
	
	[For FSP SC (IAR EWARM)]
	No need setting since code generation is not executed automatically.
	4. Modify definitions in rzt_gen/hal_data.c
	Change the value of [module name]_ pwm_pin_setting[] or [module name]_complementary_pwm_setting[1].pin_setting[X](X=0,1,2) according to the MTU3 output pins used. The tables below show the replacements required for each MTU3 output pins.
	
	File to be modified: rzt_gen/hal_data.c
	`/* Setting structure for pwm pin. */`
	`static const poe3_pwm_pin_setting_t g_poe30_pwm_pin_setting[] =`
	`{`
	`{ .pwm_pin_select = POE3_PIN_SELECT_0, .hiz_output_enable = false },`
	`{ .pwm_pin_select = POE3_PIN_SELECT_0, .hiz_output_enable = false },`
	`{ .pwm_pin_select = POE3_PIN_SELECT_0, .hiz_output_enable = false },`
	`{ .pwm_pin_select = POE3_PIN_SELECT_0, .hiz_output_enable = false }, };`
	
	`/* Setting structure for complementary pwm pin. */`
	`static const poe3_complementary_pwm_setting_t g_poe30_complementary_pwm_setting[] =`
	`{`
	`{ .pin_setting[0] =`

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

For MTU0-B

MTU3 output pin	Port	Location (Struct [module name]_ pwm_pin_setting[])	Replace with
mtioc0b	P14_4	the **second** pwm_pin_select	.pwm_pin_select = POE3_PIN_SELECT_1
mtioc0b	P24_0	the **second** pwm_pin_select	.pwm_pin_select = POE3_PIN_SELECT_2
mtioc0b	P13_3	the **second** pwm_pin_select	.pwm_pin_select = POE3_PIN_SELECT_3

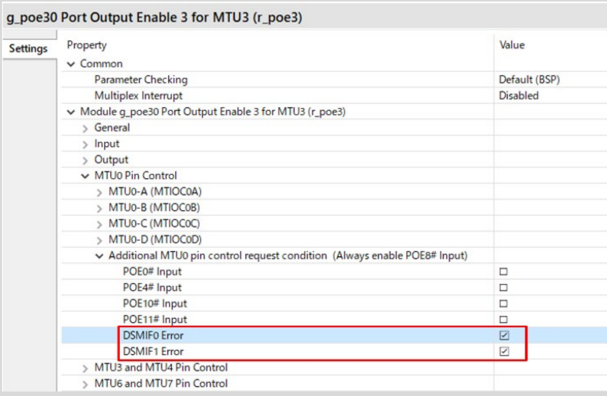
For MTU6/MTU7

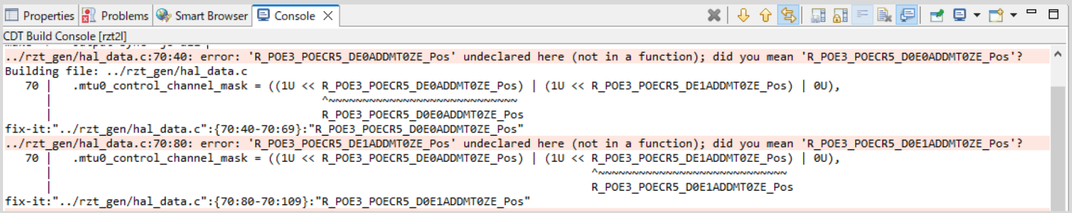
MTU3 output pin	Port	Location (Struct [module name]_complementary_pwm_setting[])	Replace with
mtioc6b	P21_2	the **second** pin_setting[0]	.positive_pwm_pin_select = POE3_PIN_SELECT_1,
mtioc6b	P08_5	the **second** pin_setting[0]	.positive_pwm_pin_select = POE3_PIN_SELECT_2,
mtioc6d	P21_4	the **second**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

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

(Continued on next page)

No. 19 Resolved

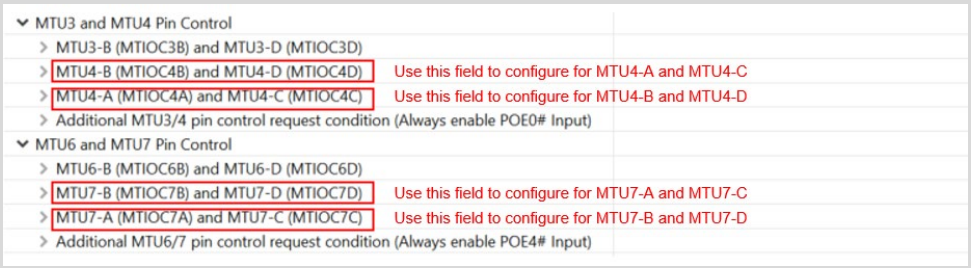
<table>
<tr><td>Issue</td><td>Build error when using DSMIFn_ERR as an additional trigger for “r_poe3” module.</td></tr>
<tr><td>Target</td><td>RZ/T2L</td></tr>
<tr><td>Category</td><td>FSP Modules, POE3</td></tr>
<tr><td>Description</td><td>When using DSMIFn_ERR as an additional trigger in FSP Configuration, after code generation, values of DSMIFn_ERR additional trigger for the module are generated in rzt_gen/hal_data.c. But the defined values are not defined in FSP, so a build error occurs.
Property of r_poe3 stack
 

Build error log
 
</td></tr>
<tr><td>Workaround</td><td>When using “r_poe3” and DSMIFn_ERR(n=0,1) as an additional trigger, please follow these four steps:
1-3. Refer to the workaround of No. 18.
4. Modify definitions in rzt_gen/hal_data.c
When using DSMIF 0 ERROR, DSMIF 1 ERROR for MTU0 additional trigger, modify:
R_POE3_POECR5_DE0ADDMT0ZE_Pos to R_POE3_POECR5_D0E0ADDMT0ZE_Pos
R_POE3_POECR5_DE1ADDMT0ZE_Pos to R_POE3_POECR5_D0E1ADDMT0ZE_Pos
When using DSMIF 0 ERROR, DSMIF 1 ERROR for MTU3/4 additional trigger, modify:
R_POE3_POECR4_DE0ADDMT34ZE_Pos
to R_POE3_POECR4_D0E0ADDMT34ZE_Pos
R_POE3_POECR4_DE1ADDMT34ZE_Pos to
R_POE3_POECR4_D0E1ADDMT34ZE_Pos
When using DSMIF 0 ERROR, DSMIF 1 ERROR for MTU6/7 additional trigger, modify:
R_POE3_POECR4_DE0ADDMT67ZE_Pos to
R_POE3_POECR4_D0E0ADDMT67ZE_Pos
R_POE3_POECR4_DE1ADDMT67ZE_Pos to
R_POE3_POECR4_D0E1ADDMT67ZE_Pos</td></tr>
</table>

(Continued on next page)

No. 20 Resolved

<table>
<tr><td>Issue</td><td>Control setting values for MTU3 output pins in Stacks tab of FSP Configuration are set to the incorrect pin.</td></tr>
<tr><td>Target</td><td>RZ/T2M, RZ/T2L, RZ/T2ME</td></tr>
<tr><td>Category</td><td>FSP Configuration, Stacks</td></tr>
<tr><td>Description</td><td>Settings for MTU4-B(MTIOC4B) and MTU4-D(MTIOC4D) in the Stacks tab property would be treated as settings for MTU4-A(MTIOC4A) and MTU4-C(MTIOC4C) in the generated file by Smart Configurator. All MTU3 output pins for MTU4 and MTU7, are the same as above.

Module [module name] Port Output Enable 3 for MTU3 (r_poe3) of Stacks property
<table>
<tr><td>Second Category</td><td>Third Category</td><td>Used for</td></tr>
<tr><td>MTU3 and MTU4 Pin Control</td><td>MTU4-B(MTIOC4B) and MTU4-D(MTIOC4D)</td><td>MTU4-A(MTIOC4A) and MTU4-C(MTIOC4C)</td></tr>
<tr><td>MTU3 and MTU4 Pin Control</td><td>MTU4-A(MTIOC4A) and MTU4-C(MTIOC4C)</td><td>MTU4-B(MTIOC4B) and MTU4-D(MTIOC4D)</td></tr>
<tr><td>MTU6 and MTU7 Pin Control</td><td>MTU7-B(MTIOC7B) and MTU7-D(MTIOC7D)</td><td>MTU7-A(MTIOC7A) and MTU7-C(MTIOC7C)</td></tr>
<tr><td>MTU6 and MTU7 Pin Control</td><td>MTU7-A(MTIOC7A) and MTU7-C(MTIOC7C)</td><td>MTU7-B(MTIOC7B) and MTU7-D(MTIOC7D)</td></tr>
</table></td></tr>
<tr><td>Workaround</td><td>In configuration of MTU4 and MTU7, please replace B/D and A/C.
 
</td></tr>
</table>

No. 21 Resolved

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

(Continued on next page)

No. 22 Resolved

Issue	**A section may not be copied correctly when it is not aligned and the section size is not a multiple of the alignment width.**
Target	**RZ/N2L**
Category	**FSP Modules, BSP**
Description	Depending on a combination of section size and placement address, when the section is copied, some data from the following section may also be copied with it. A case that cannot be copied correctly: The address of .data_noncache section is 0x30190005 and its size is 0x23 bytes. (Alignment size is 4 bytes.) `.data_noncache 0x30190005 0x23 ./src/hal_entry.o`
Workaround	All section sizes must be aligned by 4 bytes and the data size should be a multiple of the number of bytes in the alignment. Section sizes can be found in the following files: • gcc: [project name]/Debug/[project name].map • iccarm: [project name]/Debug/List/[project name].map

No. 23 Resolved

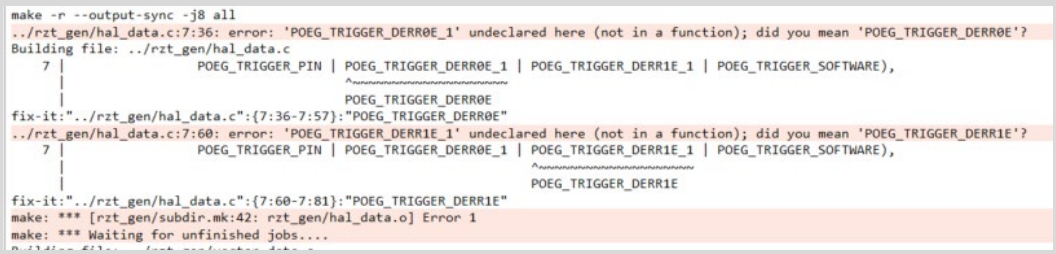
Issue	**Initial values of data placed in some sections were overwritten with 0.**
Target	**RZ/N2L**
Category	**FSP Modules, BSP**
Description	When selecting XXXXX (RAM execution without flash memory) as “Board” in BSP tab of FSP Configuration, variables placed in the following sections are always cleared to zero. • .dmac_link_mode • .shared_noncache_buffer • .noncache_buffer
Workaround	Do NOT place data with initial values in the above sections.

(Continued on next page)

No. 24 Resolved

Issue	**Some sections were not initialized in the flash boot project.**
Target	**RZ/N2L**
Category	**FSP Modules, BSP**
Description	When selecting a flash boot mode as “Board” in BSP tab of FSP Configuration, variables placed in the following sections are NOT initialized. Boards for flash boot mode • XXXXX (xSPI0 x1 boot mode) • XXXXX (16-bit bus NOR flash boot mode) • RZN2L Custom User Board (xSPI0 x8 boot mode) • RZN2L Custom User Board (xSPI1 x1 boot mode) Sections • .dmac_link_mode • .shared_noncache_buffer • .noncache_buffer
Workaround	Please initialize the variables placed in the above sections in the user application.

No. 25 Resolved

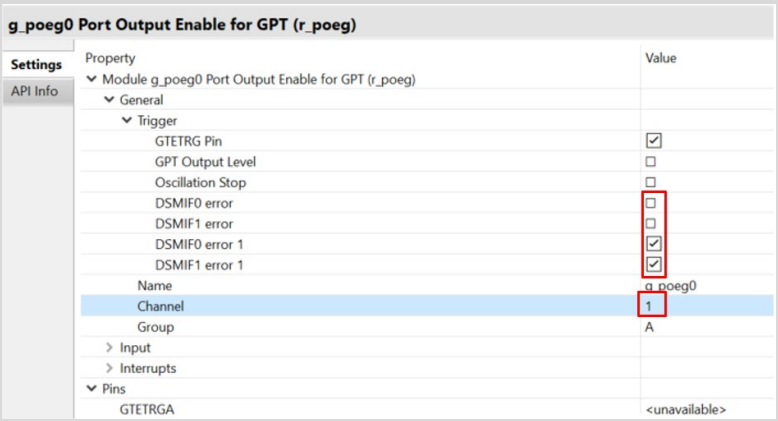
Issue	**DSMIF 0/1 error 1 trigger macros are not defined.**
Target	**RZ/T2L**
Category	**FSP Modules, POEG**
Description	In “bsp_override.h” of rzt2l device, enum e_poeg_trigger, the definition for DSMIF0 error 1 and DSMIF1 error 1 are missing. When setting as below, build errors will occur. Property of r_poeg stack  Generated code by configurator  Build error log 

Workaround	Add definition for DSMIF0 error 1 and DSMIF1 error 1 trigger in enum e_poeg_trigger.
	Location: rzt/fsp/src/bsp/mcu/rzt2l/bsp_override.h, enum e_poeg_trigger
	Add content:
	`POEG_TRIGGER_DERR0E_1 = 1U << 18,  ///< Permit output disabled by DSMIF0 error 1 detection`
	`POEG_TRIGGER_DERR1E_1 = 1U << 19,  ///< Permit output disabled by DSMIF1 error 1 detection`

No. 26 Resolved

Issue	**DSMIF 0/1 error 1 status macros are not defined.**
Target	**RZ/T2L**
Category	**FSP Modules, POEG**
Description	In “bsp_override.h” of rzt2l device, enum e_poeg_state, the definition for DSMIF0 error 1 state and DSMIF1 error 1 state are missing. When using R_POEG_StatusGet, · If POEG module is in state GPT output disabled due to DSMIF0 error 1, the p_status will be 0x100000 instead of POEG_STATE_DSMIF0_1_DISABLE_REQUEST. · If POEG module is in state GPT output disabled due to DSMIF1 error 1, the p_status will be 0x200000 instead of POEG_STATE_DSMIF1_1_DISABLE_REQUEST.
Workaround	When the POEG module is in the 'GPT output disabled' state due to a DSMIF0 error 1, assume that p_status = 0x100000 corresponds to POEG_STATE_DSMIF0_1_DISABLE_REQUEST. When the POEG module is in the 'GPT output disabled' state due to a DSMIF1 error 1, assume that p_status = 0x200000 corresponds to POEG_STATE_DSMIF1_1_DISABLE_REQUEST.

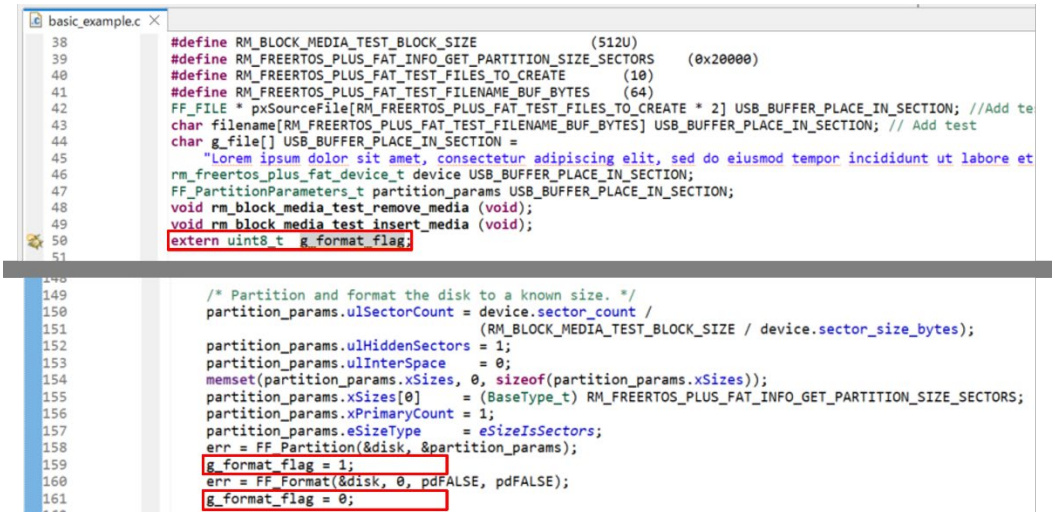
No. 27 Resolved

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

(Continued on next page)

No. 28

Issue	**FreeRTOS+FAT format process is not executed correctly.**
Target	**RZ/T2M, RZ/T2ME, RZ/T2L, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**FSP Modules, FreeRTOS+FAT**
Description	Executing the FF_Format function causes a USB AHB bus error and the FreeRTOS+FAT format function processing is not executed correctly.

Workaround	Please add g_format_flag before and after calling the FF_Format function in your application code.
	
	`extern uint8_t g_format_flag;`
	
	`g_format_flag = 1;`
	`err = FF_Format(&disk, 0, pdFALSE, pdFALSE)`
	`g_format_flag = 0;`
	
	 

No. 29 Resolved

Issue	**Caution when specifying program placement in linker scripts**
Target	**RZ/T2H, RZ/N2H**
Category	**Others, Linker script**

Description	The reserved area in the device address space cannot be used for a program placement, but no error occurs when placed there by a linker script.
	For example, CA55 core has reserved areas ATCM and BTCM, however the linker script is ready to place them there, and no error occurs when the following description exists.
	
	`.text TEXT_ADDRESS : AT (TEXT_ADDRESS)`
	`{`
	`  ......abbreviation......`
	`} > ATCM`

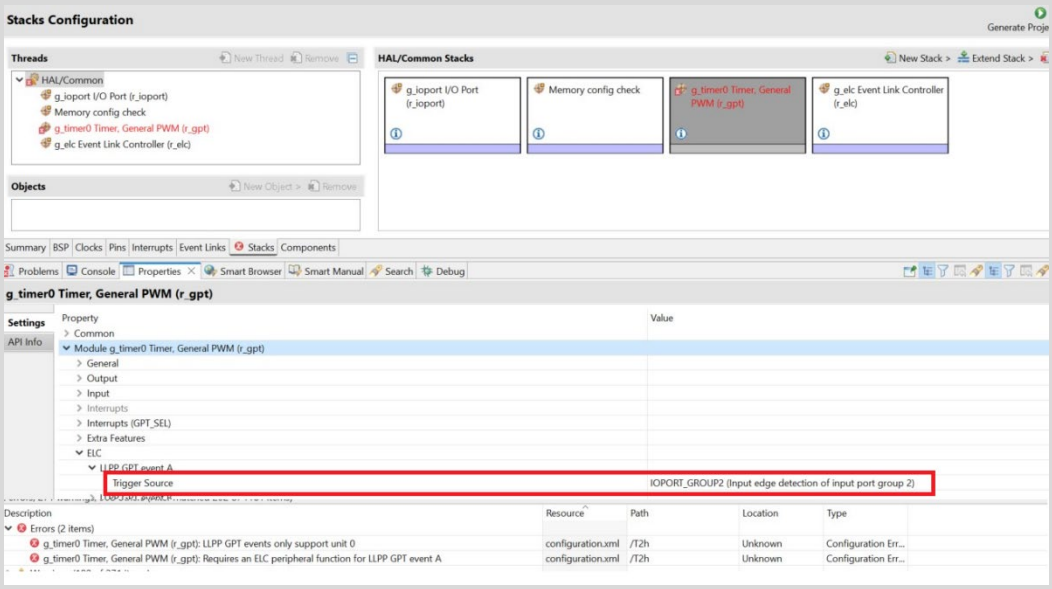
Workaround	Do not specify that any program is to be placed at the reserved area in a linker script.

(Continued on next page)

No. 30

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

No. 31 Resolved

Issue	**Errors occur when setting ELC in r_gpt module.**
Target	**RZ/T2H, RZ/N2H**
Category	**FSP Configuration, Stacks**
Description	When setting ELC event trigger source in the stack of r_gpt module, errors will occur and cannot generate the value. 

Workaround	To use the ELC trigger source for r_gpt module in code without configuring it in the FSP Configurator, follow these steps.
	1. Add the r_elc stack in the Stacks tab of FSP Configuration and include the r_elc header in your application.
	2. Call the following functions to initially set up the r_elc module.
	i. R_ELC_Open();
	ii. R_ELC_Enable();
	3. Use the R_ELC_LinkSet(); function to configure the event triggers.
	
	Example:
	`R_ELC_Open(&g_gpt_test_elc_ctrl, &g_elc_cfg);`
	`R_ELC_Enable(&g_gpt_test_elc_ctrl);`
	`R_ELC_LinkSet(&g_gpt_test_elc_ctrl, ELC_PERIPHERAL_GPT00_A, ELC_EVENT_INTCPU0);`

No. 32 Resolved for RZ/T series devices in RZT FSP v3.0.0

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

No. 33

Issue	**No Error Occurs when entering out-of-range values for window parameters in r_pcie_ep and r_pcie_rc module configurations.**
Target	**RZ/T2H, RZ/N2H**
Category	**FSP Configuration, Stacks**
Description	There is a problem with the r_pcie_ep and r_pcie_rc module configuration screens where entering unsupported values does not generate an error and the code can be generated with unusable values. The configuration items for which the validity judgment of input value does not work are as follows.

Configuration item	r_pcie_ep	r_pcie_rc
AXI Window Base	✓	✓
AXI Window Mask	✓	✓
AXI Window Destination	✓	✓
PCIe Window Base	✓	✓
PCIe Window Mask	✓	✓
PCIe Window Destination	✓	✓
MSI Receive Window Address	-	✓
MSI Receive Window Mask	-	✓

Workaround	Please enter the values so that they meet the input value conditions for each item. AXI Window Base and PCIe Window Base - “greater than or equal to 0” and ”address is 4Kbyte aligned” AXI Window Mask and PCIe Window Mask - “greater than or equal to 0”, ‘the lower 12 bits are 1’, and ”the 63rd bit is 0” AXI Window Destination and PCIe Window Destination - “greater than or equal to 0” and ”address is 4Kbyte aligned” MSI Receive Window Mask - “greater than or equal to 0” and ”the lower 2 bits are 1” MSI Receive Window Address - Align according to “MSI Receive Window Mask”.

(Continued on next page)

No. 34

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

No. 35

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

No. 36 Resolved

Issue	**r_adc module does not support the calibration function.**
Target	**RZ/T2H**
Category	**FSP Modules, ADC**
Description	The 12-bit A/D converter needs to be calibrated before A/D conversion after reset is released. However, since the FSP does not support the calibration function, the accuracy shown in the electrical characteristics chapter of device user's manual cannot be guaranteed.

Workaround	This will be implemented in the next version of FSP.
	As a temporary measure, the calibration process must be implemented in the user application before the R_ADC_Open function is executed.
	The following is an example of implementation. (In the case of ADC Unit2)
	`#define ADC_ADCALCTL_SET_CAL 1U`
	
	`/* Release module stop for ADC12 */`
	`R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_LPC_RESET);`
	`R_BSP_MODULE_START(FSP_IP_ADC12, 2);`
	`R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_LPC_RESET);`
	
	`R_BSP_SoftwareDelay(1, BSP_DELAY_UNITS_MICROSECONDS);`
	`/* Write ADCCALCTL.CAL bit to 1 to start calibration. */`
	`R_ADC122->ADCALCTL_b.CAL = ADC_ADCALCTL_SET_CAL;`
	
	`/* Poll ADCCALCTL.CAL_RDY bit until it is changed to 1. */`
	`FSP_HARDWARE_REGISTER_WAIT(R_ADC122->ADCALCTL_b.CAL_RDY, 1U);`
	`/* Confirm ADCCALCTL.CAL_ERR bit is 0.*/`
	`FSP_HARDWARE_REGISTER_WAIT(R_ADC122->ADCALCTL_b.CAL_ERR, 0U);`
	`/* Write ADCCALCTL.CAL bit to 0 */`
	`R_ADC122->ADCALCTL_b.CAL = 0U;`
	
	`/* Initializing the ADC module */`
	`R_ADC_Open(&g_adc0_ctrl,&g_adc0_cfg );`

No. 37 Resolved

Issue	**The USB driver for CA55 project does not work.**
Target	**RZ/T2H(CA55), RZ/N2H(CA55)**
Category	**FSP Modules, USB**
Description	The R_BSP_MmuPatoVA function executed from USB driver (r_usb_hmsc, r_usb_hcdc, r_usb_hhid modules) in a CA55 project fails to perform the expected address translation, resulting in a USB transfer failure.
Workaround	Please modify \rzt\fsp\src\r_usb_basic\src\driver\r_usb_mmu_pa_to_va.c as follows. 1. Modify the r_usb_pa_to_va function. 2. Modify the r_usb_va_to_pa function.

```
uint64_t r_usb_pa_to_va (uint64_t paddr)
{
    uint64_t vaddr = 0;

#if defined(BSP_CFG_CORE_CA55)

    /* Converts a physical address to a virtual address.  */
    if (FSP_SUCCESS != R_BSP_MmuPatoVa(paddr, &vaddr, BSP_MMU_CONVERSION_NON_CACHE))
    {
        /* On error, returns the physical address without conversion. */
        vaddr = paddr;
    }
#else /* #if defined(BSP_CFG_CORE_CA55) */
    vaddr = paddr;
#endif /* #if defined(BSP_CFG_CORE_CA55) */

    return vaddr;
}                                       /* End of function r_usb_pa_to_va() */
```

```
uint64_t r_usb_va_to_pa (uint64_t vaddr)
{
    uint64_t paddr = 0;

#if defined(BSP_CFG_CORE_CA55)

    /* Converts a virtual address to a physical address.  */
    if (FSP_SUCCESS != R_BSP_MmuVatoPa(vaddr, &paddr))
    {
        /* On error, returns the virtual address without conversion. */
        paddr = vaddr;
    }
#else /* #if defined(BSP_CFG_CORE_CA55) */
    paddr = vaddr;
#endif /* #if defined(BSP_CFG_CORE_CA55) */

    return paddr;
}
```

No. 38 Resolved

Issue	**No error returns when entering the virtual addresses that cannot be translated to physical addresses as arguments.**
Target	**RZ/T2H(CA55), RZ/N2H(CA55)**
Category	**FSP Modules, xSPI_OSPI, xSPI_QSPI, DMAC**
Description	In a CA55 project, there is a problem with the "r_xspi_qspi", "r_xspi_ospi", and "r_dmac" modules that no error returns when entering virtual addresses that cause translation error in MMU as arguments. The following functions have the problem. -R_XSPI_QSPI_Write() -R_XSPI_OSPI_Write() -R_DMAC_Open() -R_DMAC_Reconfigure() -R_DMAC_Reload() -R_DMAC_LinkDescriptorSet()
Workaround	Do not enter the virtual addresses that cannot be translated to physical addresses as arguments.

No. 39 Resolved

Issue	**The CA55 project with noncache sections aborts when debugging with flash boot mode on IAR EWARM**
Target	**RZ/T2H(CA55), RZ/N2H(CA55)**
Category	**FSP Modules, BSP**
Description	When performing debugging of a flash boot CA55 project with noncache sections on IAR EWARM, executing the CA55 project will abort. This is due to cache initialization.
Workaround	Follow the steps below 1. Add " set_ICIALLU(0);" and "__ISB();" to bsp_memory_protect_setting in XXX/fsp/src/bsp/cmsis/Device/RENESAS/Source/ca/system_core.c. (XXX= rzt, rzn) 2. Move "__set_ICIALLU(0);" in R_BSP_CacheInvalidateAll in XXX/fsp/src/bsp/mcu/all/bsp_cache.c (XXX= rzt, rzn) to just before "__asm volatile("ISB SY");" at the end.

```
void bsp_memory_protect_setting (void)
{
    bsp_mmu_configure();
    R_BSP_CacheInvalidateAll();

    R_BSP_CacheEnableMemoryProtect();
    R_BSP_CacheEnableInst();
    R_BSP_CacheEnableData();

    __set_ICIALLU(0);
    __ISB();
}
```

```
    uintptr_t ccsidr_associativity_value;
    uintptr_t ccsidr_associativity_msb;
    uintptr_t ccsidr_numsets;
    uintptr_t ccsidr_numsets_total;

    uintptr_t dcisw;

    __asm volatile ("DSB SY");

    __set_ICIALLU(0);

    __asm volatile ("DMB SY");

    /* Reads the maximum level of cache implemented */
    clidr     = __get_CLIDR();
    clidr_loc = (clidr >> BSP_PRV_CLIDR_LOC_OFFSET) & BSP_PRV_CLIDR_LOC_MASK;

    /* If the cache does not exist, do not process */
    if (0 != clidr_loc)
    {
                }
            }
            else
            {
                /* Do Nothing */
            }
        }

        __asm volatile ("DSB SY");

        __asm volatile ("ISB SY");
    }
    else
    {
        /* Do Nothing */
    }
```

```
    uintptr_t ccsidr_associativity_value;
    uintptr_t ccsidr_associativity_msb;
    uintptr_t ccsidr_numsets;
    uintptr_t ccsidr_numsets_total;

    uintptr_t dcisw;

    __asm volatile ("DSB SY");


    __asm volatile ("DMB SY");

    /* Reads the maximum level of cache implemented */
    clidr     = __get_CLIDR();
    clidr_loc = (clidr >> BSP_PRV_CLIDR_LOC_OFFSET) & BSP_PRV_CLIDR_LOC_MASK;

    /* If the cache does not exist, do not process */
    if (0 != clidr_loc)
    {
                }
            }
            else
            {
                /* Do Nothing */
            }
        }

        __asm volatile ("DSB SY");
        __set_ICIALLU(0);

        __asm volatile ("ISB SY");
    }
    else
    {
        /* Do Nothing */
    }
```

No. 40 Resolved

Issue	**When changing the duty setting in r_gpt module, there is a possibility the duty may unintentionally become 100%.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**FSP Modules, GPT**
Description	When changing the PWM period with "R_GPT_PeriodSet()" and the duty with "R_GPT_DutyCycleSet()" while the GPT is running, the duty may unintentionally become 100% depending on the both setting values. The reason is that when "gpt_calculate_duty_cycle()" in "R_GPT_DutyCycleSet()" performs a comparison calculation of the duty and period, the old period value of GTPR register is mistakenly referenced instead of the current period value of GTPBR (buffer) register.

Workaround	Please correct the reading of the period in the "gpt_calculate_duty_cycle()" to GTPBR instead of GTPR in XXX/fsp/src/r_gpt/r_gpt.c. (XXX= rzt, rzn)
	
	`static void gpt_calculate_duty_cycle (gpt_instance_ctrl_t * const p_instance_ctrl,`
	`  uint32_t const  duty_cycle_counts,`
	`  gpt_prv_duty_registers_t  * p_duty_reg)`
	`{`
	`  /* Determine the current period. The actual period is one cycle longer than the register value for saw waves`
	`  * and twice the register value for triangle waves. Reference section 19.2.21 "General PWM Timer Cycle Setting`
	`  * Register (GTPR)". The setting passed to the configuration is expected to be half the desired duty cycle for`
	`  * triangle waves. */`
	`  uint32_t current_period = p_instance_ctrl->p_reg->GTPR;`
	`#if GPT_PRV_EXTRA_FEATURES_ENABLED == GPT_CFG_OUTPUT_SUPPORT_ENABLE`
	`  if (p_instance_ctrl->p_cfg->mode < TIMER_MODE_TRIANGLE_WAVE_SYMMETRIC_PWM)`
	`#endif`

No. 41 Resolved

Issue	**CPU registers save and restore process cannot be performed correctly in FIQ_Handler for CA55 projects.**
Target	**RZ/T2H(CA55), RZ/N2H(CA55)**
Category	**FSP Modules, BSP, FreeRTOS**
Description	Some of CPU registers are not saved and restored in FIQ_Handler. Therefore, the value of registers may be partially corrupted before or after the FIQ interrupt occurs.

Workaround	• When NOT using FreeRTOS
	Please modify XXX/fsp/src/bsp/cmsis/Device/RENESAS/Source/ca/startup_core.c (XXX= rzt, rzn) as follows.
	
	`__WEAK void FIQ_Handler (void)`
	`{`
	`  __asm volatile (`
	`  "STP  x30, XZR, [SP, #-0x10]!  \n"`
	`  "STP  x28, x29, [SP, #-0x10]!  \n"`
	`  "STP  x26, x27, [SP, #-0x10]!  \n"`
	`  "STP  x24, x25, [SP, #-0x10]!  \n"`
	`  "STP  x22, x23, [SP, #-0x10]!  \n"`
	`  "STP  x20, x21, [SP, #-0x10]!  \n"`
	`  "STP  x18, x19, [SP, #-0x10]!  \n"`
	`  "STP  x16, x17, [SP, #-0x10]!  \n"`
	`  "STP  x14, x15, [SP, #-0x10]!  \n"`
	`  "STP  x12, x13, [SP, #-0x10]!  \n"`
	`  "STP  x10, x11, [SP, #-0x10]!  \n"`
	`  "STP  x8, x9, [SP, #-0x10]!  \n"`
	`  "STP  x6, x7, [SP, #-0x10]!  \n"`
	`  "STP  x4, x5, [SP, #-0x10]!  \n"`
	`  "STP  x2, x3, [SP, #-0x10]!  \n"`
	`  "STP  x0, x1, [SP, #-0x10]!  \n"`
	
	`#if __FPU_USED`

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

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

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

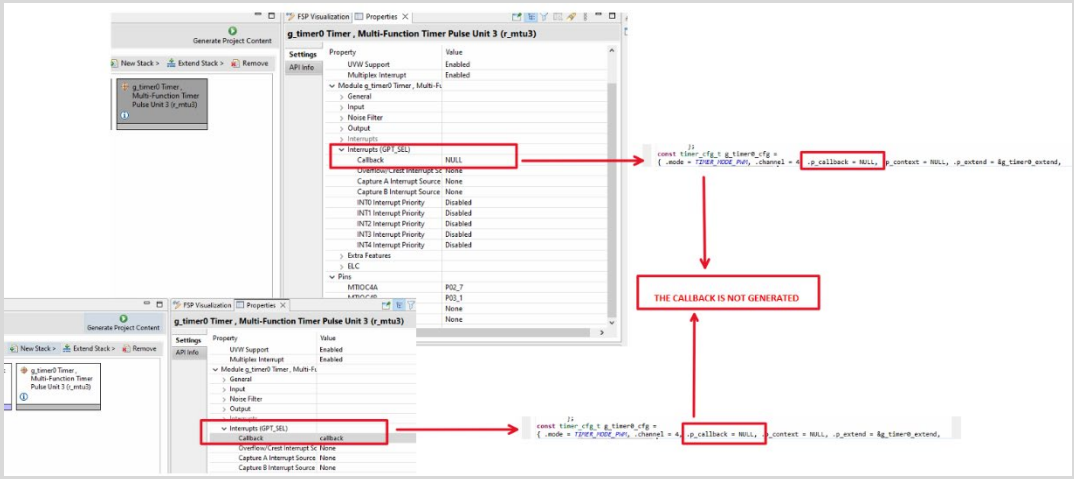
- When using FreeRTOS
  Please modify XXX/fsp/src/rm_freertos_port/ca/port.c (XXX= rzt, rzn) as follows.

```
BSP_ATTRIBUTE_STACKLESS void FIQ_Handler (void)
{
    /* Save volatile registers. */
    __asm volatile (
        "STP  x30, XZR, [SP, #-0x10]!           \n"
        "STP  x28, x29, [SP, #-0x10]!           \n"
        "STP  x26, x27, [SP, #-0x10]!           \n"
        "STP  x24, x25, [SP, #-0x10]!           \n"
        "STP  x22, x23, [SP, #-0x10]!           \n"
        "STP  x20, x21, [SP, #-0x10]!           \n"
        "STP  x18, x19, [SP, #-0x10]!           \n"
        "STP  x16, x17, [SP, #-0x10]!           \n"
        "STP  x14, x15, [SP, #-0x10]!           \n"
        "STP  x12, x13, [SP, #-0x10]!           \n"
        "STP  x10, x11, [SP, #-0x10]!           \n"
        "STP  x8, x9, [SP, #-0x10]!             \n"
```

```
        "STP  x6, x7, [SP, #-0x10]!             \n"
        "STP  x4, x5, [SP, #-0x10]!             \n"
        "STP  x2, x3, [SP, #-0x10]!             \n"
        "STP  x0, x1, [SP, #-0x10]!             \n"

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

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

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

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

BSP_ATTRIBUTE_STACKLESS void Exit_IRQ_No_Context_Switch (void)
{
~~~~~~~~~~~~

        "DSB    SY                                  \n"
        "ISB    SY                                  \n"

#if __FPU_USED
        "LDP  q0, q1, [sp], #0x20               \n"
        "LDP  q2, q3, [sp], #0x20               \n"
```

```
        "LDP  q4, q5, [sp], #0x20               \n"
        "LDP  q6, q7, [sp], #0x20               \n"
        "LDP  q8, q9, [sp], #0x20               \n"
        "LDP  q10, q11, [sp], #0x20             \n"
        "LDP  q12, q13, [sp], #0x20             \n"
        "LDP  q14, q15, [sp], #0x20             \n"
        "LDP  q16, q17, [sp], #0x20             \n"
        "LDP  q18, q19, [sp], #0x20             \n"
        "LDP  q20, q21, [sp], #0x20             \n"
        "LDP  q22, q23, [sp], #0x20             \n"
        "LDP  q24, q25, [sp], #0x20             \n"
        "LDP  q26, q27, [sp], #0x20             \n"
        "LDP  q28, q29, [sp], #0x20             \n"
        "LDP  q30, q31, [sp], #0x20             \n"
        "LDP  x0, x1, [sp], #0x10               \n"
        "MSR  FPCR, x0                          \n"
        "MSR  FPSR, x1                          \n"
#endif

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

No. 42 Resolved

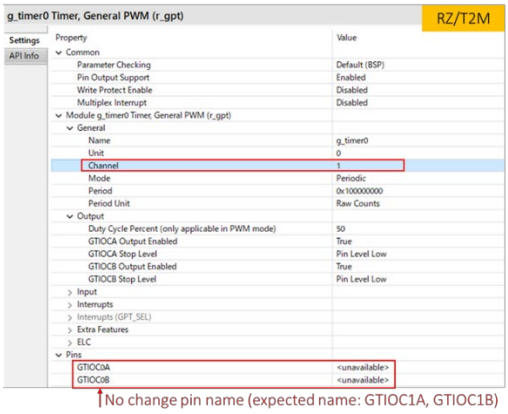
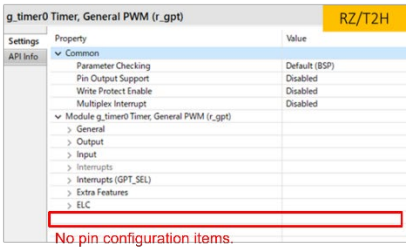
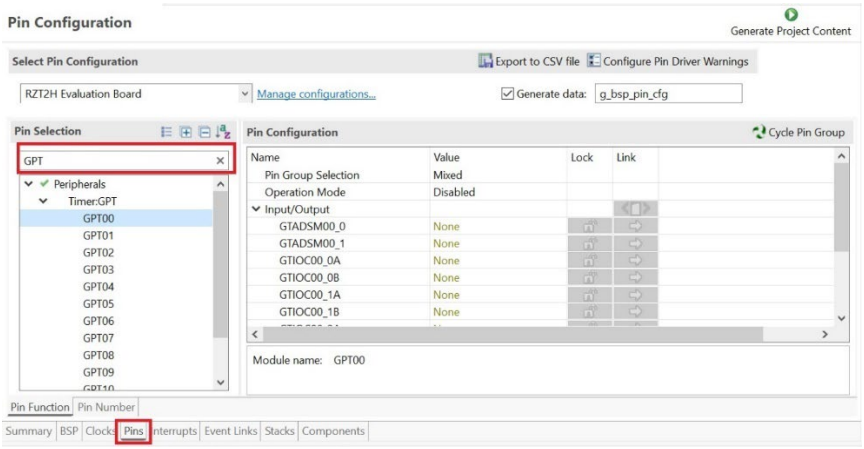
Issue	**MTU3 callback does not occur as expectation.**
Target	**RZ/T2H, RZ/N2H**
Category	**FSP Modules, MTU3**
Description	Callback does not always occur just like when you specify NULL in the configurator settings of MTU3. 

Workaround	Please use the Callback Set API from:
	R_MTU3_CallbackSet()
	For reference on how to use the API, please refer to MTU3 Examples below:
	
	Step1: Define the Callback Function
	
	`void mtu3_callback_set_test(timer_callback_args_t * p_args)`
	`{`
	`  //Callback function to be implemented by the user`
	`}`
	
	Step2: Sequence call the API
	
	`/* Open the timer module */`
	`R_MTU3_Open(&g_timer0_ctrl, &g_timer0_cfg);`
	
	`/* Use R_MTU3_CallbackSet */`
	
	`R_MTU3_CallbackSet (timer_ctrl_t * const p_ctrl,`
	`  void ( * p_callback)(timer_callback_args_t *),`
	`  void const * const p_context,`
	`  timer_callback_args_t * const p_callback_memory)`
	Example:
	`R_MTU3_CallbackSet (`
	`  &g_timer0_ctrl, // Timer control instance`
	`  mtu3_callback_set_test, // User define Callback function`
	`  (void )&mtu3_callback_context // User Defined context`
	`  (void )&mtu3_callback_test_args // User define Callback memory`
	`)`
	`/* Start timer */`
	`R_MTU3_Start(&g_timer0_ctrl);`

No. 43 Resolved

Issue	**An undefined error of r_gpt module occurs when building a project.**
Target	**RZ/T2H, RZ/N2H**
Category	**FSP Modules, GPT**
Description	Undefined error of "gpt_counter_underflow_isr" occurs when "Pin Output Support" is set to anything other than "Enabled with Extra Features" regardless of whether Pin Output is used or not.
Workaround	Please always set the “Pin Output Support” to "Enabled with Extra Features" when configuring r_gpt module. 

No. 44

Issue	**Pin names according to unit and channel numbers are not displayed in r_gpt module configurations.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**FSP Configuration, Stacks**
Description	When setting unit and channel number in r_gpt module, pin names in the Pins of stack property do not change or not appear.  
Workaround	Please use the Pins tab in the FSP Configuration to config pins for r_gpt module. 

No. 45 Resolved

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

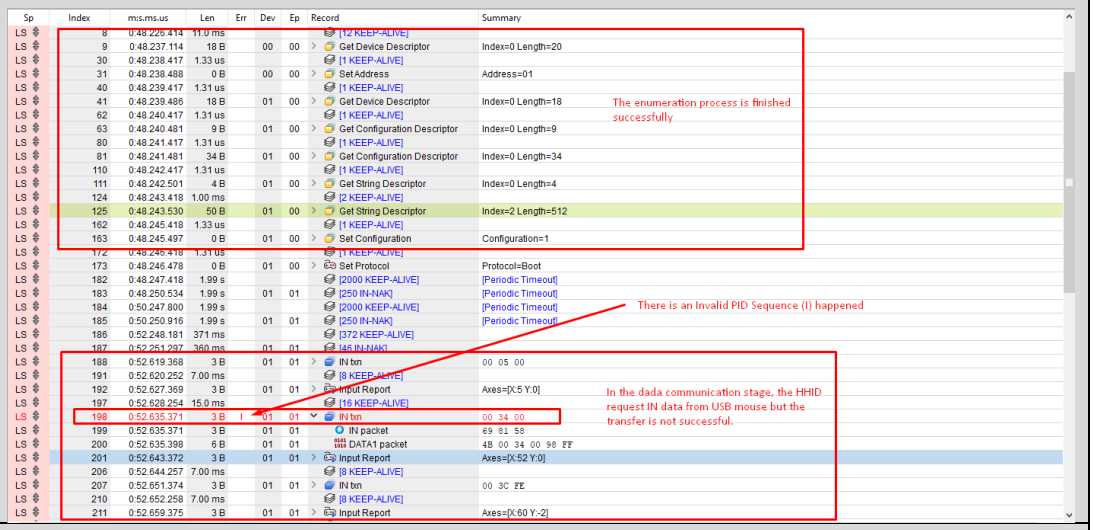
No. 46 Resolved for RZ/T series devices in RZT FSP v3.0.0

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

No. 47 Resolved for RZ/T series devices in RZT FSP v3.0.0

Issue	**Parameter checking feature of R_GMAC_CallbackSet() is not working.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**FSP Modules, ETHER_GMAC**
Description	Parameter check feature of R_GMAC_CallbackSet() does not work because the parameter checking macro name generated by the MDF is inconsistent with the macro referenced in the actual source code. • Macro generated by MDF: GMAC_CFG_PARAM_CHECKING_ENABLE • Macro used in source code: ETHER_CFG_PARAM_CHECKING_ENABLE Note: Parameter check feature of other APIs work properly.
Workaround	Step 1: In project Properties->Builders, uncheck DDSC Builder option (this step make user can modify "r_ether_cfg.h" file) Step 2: In "XXX_cfg/fsp_cfg/r_ether_cfg.h"(XXX = rzt, rzn), manually add the macro definition below. #define ETHER_CFG_PARAM_CHECKING_ENABLE ((1))

No. 48 Resolved

Issue	**r_usb_hhid module is not working properly.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H**
Category	**FSP Modules, USB_HHID**
Description	It is possible that HOST does not respond to the received data because it cannot get the transfer size correctly, so even if the transfer is completed without error, it may not be successful. In the report, when debuging with a usb analyzer, errors are reported as belows: 

Workaround	Please modify “rzt\fsp\src\r_usb_basic\r_usb_basic.c” line 1877 as below.
	Before update:
	`(*p_pipe) = (uint16_t) ((*p_pipe) | (uint16_t) 1 << (pipe_no - 1));/* Start timer */`
	After update:
	`if ((((uint16_t) p_instance_ctrl->device_address) << USB_DEVADDRBIT) ==`
	`    (uint16_t) (g_usb_pipe_table[p_instance_ctrl->module_number][pipe_no].pipe_maxp & USB_DEVSEL))`
	`{`
	`    (*p_pipe) = (uint16_t) ((*p_pipe) | (uint16_t) 1 << pipe_no);`
	`}`

Appendix. Tool Software Limitations

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

Table 21 List of Tool Software Limitations

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
1	When installing, please install into the default installation folder specified by 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
11	In IAR EWARM 9.60.1, an error occurs when starting to debug multiprocessing projects of RAM execution without flash memory.	✓		✓				IAR EWARM
12	Build is failed when executed with different install path.	✓	✓	✓	✓	✓	✓	FSP SC
13	Unable to debug CA55 flash boot project with e² studio.				✓		✓	e² studio
14	Unable to restart debugging immediately after debugging ends of CA55 RAM execution without flash memory project in e² studio.				✓		✓	e² studio
15	Unable to re-download CA55 binary file.				✓		✓	IAR EWARM

No.	Title	Target Device						Category
		T2M	T2L	T2ME	T2H	N2L	N2H	
16	Unable to access the upper 32-bit address area in memory view.				✓		✓	e² studio
17	Unable to create a CMake project using FSP SC.	✓	✓	✓	✓	✓	✓	FSP SC
18	The secondary project aborts when debugging multicore with flash boot mode.	✓		✓	✓		✓	IAR EWARM
19	Build errors occur in CA55 projects when install e² studio as the Current user.				✓		✓	e² studio
20	Wrong core name when create a project CA55 with FSP SC.						✓	FSP SC
21	Build is failed when adding the OpenAMP.				✓		✓	FSP SC
22	The bundle file (.sbd) may not be generated during build.	✓	✓	✓	✓	✓	✓	e² studio
23	When implementing CMT interrupts in RZ/T2H CR52 or RZ/N2H CR52 project, an unintended source file is displayed during debugging.				✓		✓	e² studio

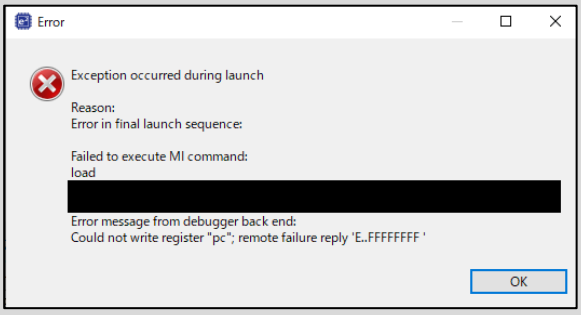
No. 1 Resolved

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

No. 3 Resolved

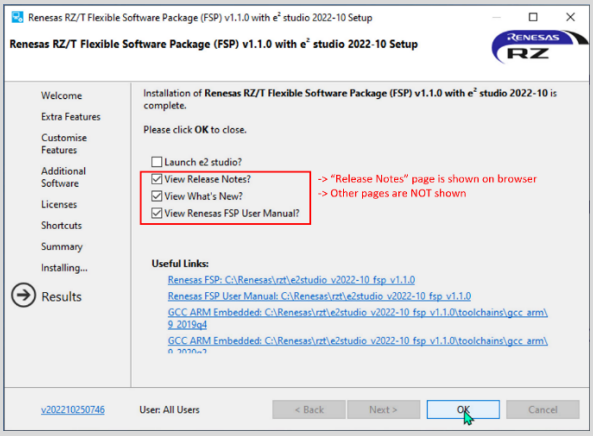
Limitation	**An error has occurred because the program download to the NOR flash area has failed. The download is successful on the second connection.**
Target	**RZ/T2M, RZ/N2L**
Category	**e² studio**
Description	If the following error is displayed when connecting the debugger or when downloading the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

No. 4

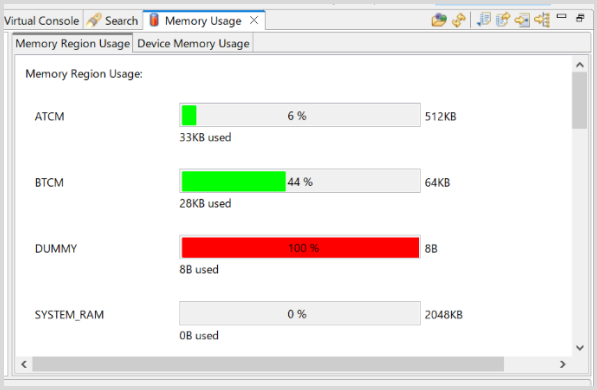
Limitation	**The user program cannot be stopped immediately after the device boot process.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**e² studio**
Description	Immediately after the device boot process (boot code), the program cannot be stopped at the beginning of the user 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. 

No. 6 Resolved

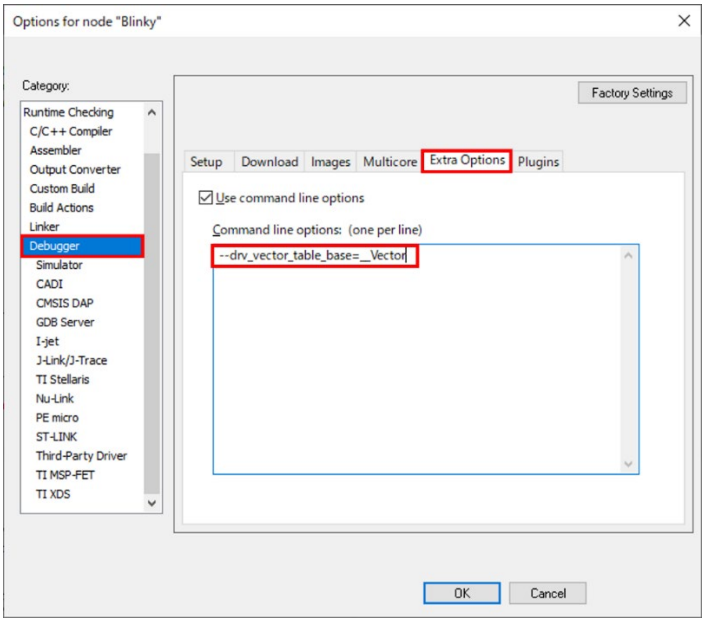
Limitation	**The Memory Region Usage of ATCM displayed in the Memory Usage window of e² studio is smaller than the actual size by Memory Region Usage of DUMMY.**
Target	**RZ/T2M, RZ/T2L, RZ/N2L**
Category	**e² studio**
Description	The Memory Region Usage of DUMMY shown in the Memory Usage window is the 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. 

(Continued on next page)

No. 7

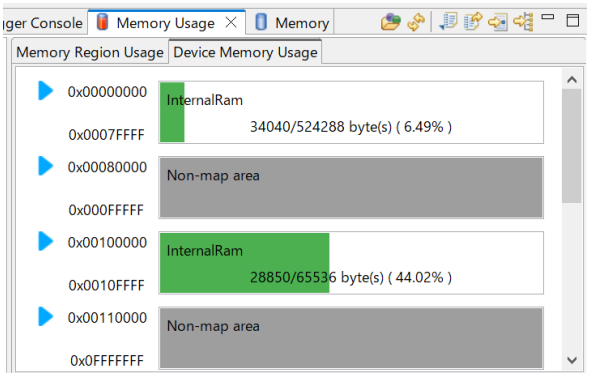
Limitation	**When debugging RAM execution without flash memory project with program written to flash memory, erase flash memory before debugging.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**e² studio**
Description	If you run an RAM execution without flash memory project with a program written in flash memory, it may be impossible to debug the project. When erasing flash memory, please follow the guide in Appendix. How to Erase Flash Memory

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, RZ/T2ME, RZ/T2H, RZ/N2H**
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. 

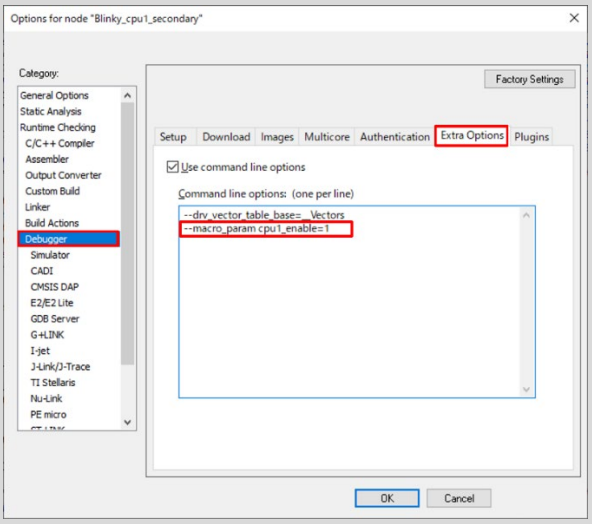
No. 10

Limitation	**When adding the CallbackSet function using the Developer Assistance feature, the second argument needs to be changed.**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**e² studio, SC**

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)

No. 11 Moved to 5.3.3.2 Build for Multiprocessing No. 2-iii

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

No. 12

Limitation	**Build is failed when executed with different install path**
Target	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**FSP SC**
Description	If install path of FSP SC is different between creating project and executing project, build of a project is failed. Please reselect the execution path by following the steps below. 1. Launch FSP SC. 2. Close the window to create a new project. 3. Click on File -> Open and select configuration.xml in your project. 4. Click "Generate Project Content". 5. Save the project and close FSP SC. 6. Open the project with EWARM.

(Continued on next page)

No. 13 Resolved

Limitation	**Unable to debug CA55 flash boot project with e² studio.**
Target Device	**RZ/T2H(CA55) , RZ/N2H(CA55)**
Category	**e² studio**
Description	When debugging with e² studio, the CA55 Core0 system reset is not performed. Therefore, the program in the external flash is not copied to the internal RAM, and it cannot be operated correctly. Please check the operation with the RAM execution without flash memory project.

No. 14 Resolved

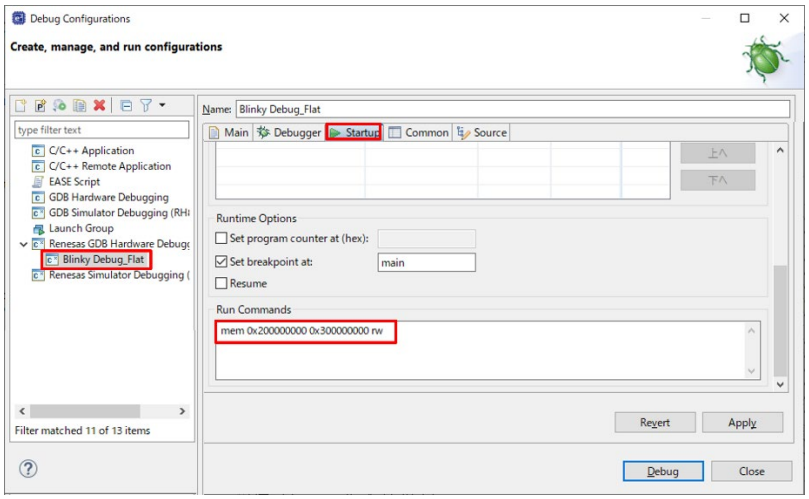
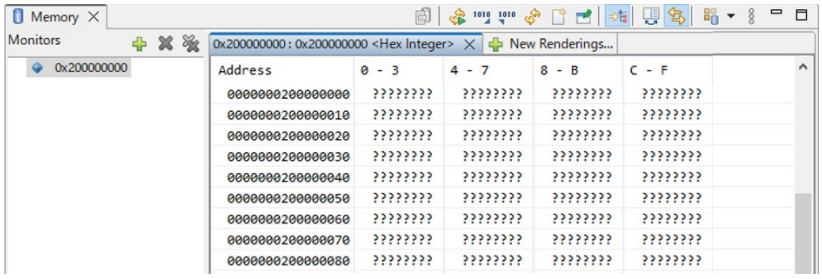
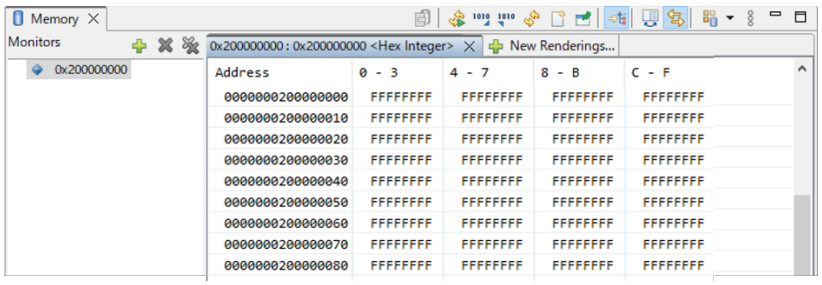
Limitation	**Unable to restart debugging immediately after debugging ends of CA55 RAM execution without flash memory project in e2 studio.**
Target Device	**RZ/T2H(CA55), RZ/N2H(CA55)**
Category	**e² studio**
Description	When debugging with e² studio, the CA55 Core0 system reset is not performed. Therefore, after debugging is complete, if you want to run the debug again, press the reset button (red) on the board.

No. 15

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

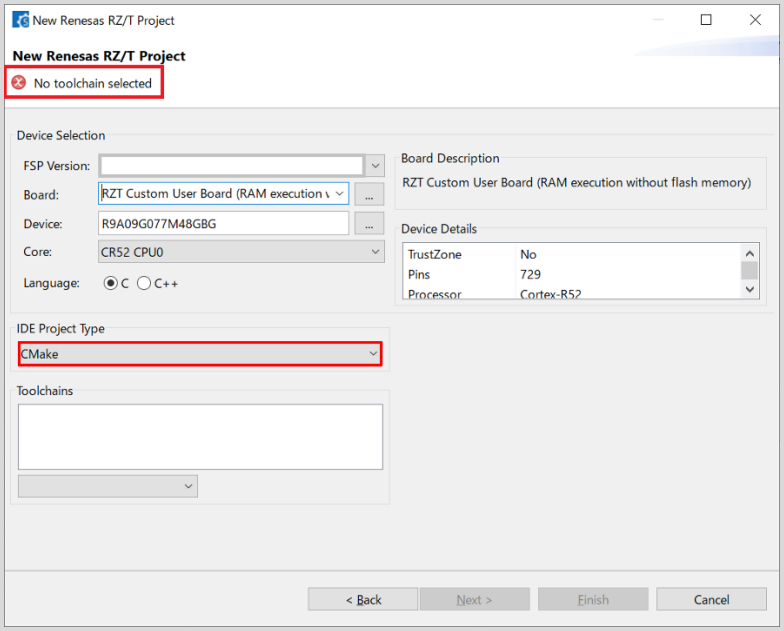
(Continued on next page)

No. 16

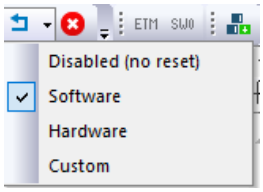
Limitation	**Unable to access the upper 32-bit address area in memory view.**
Target Device	**RZ/T2H(CA55), RZ/N2H(CA55)**
Category	**e² studio**
Description	In the Memory view of e² studio, data in the upper 32-bit address area cannot be displayed. When debugging the CA55 project, please make the following settings when accessing the upper 32-bit address. 1. Open **Debug Configurations** of CA55 project. 2. Select **Renesas GDB Hardware Debugging** > [project name] Debug_Flat. 3. Select **Startup** and specify the command in **Run Commands**. Referring to the following command, specify the address area you want to access, and you will be able to access the upper 32-bit address area. "mem 0x200000000 0x300000000 rw"  Before adding the command  After adding the command 

(Continued on next page)

No. 17 Resolved

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

No. 18

Limitation	**The secondary project aborts when debugging multicore with flash boot mode.**
Target Device	**RZ/T2M, RZ/T2ME, RZ/T2H, RZ/N2H**
Category	**IAR EWARM**
Description	When performing multicore debugging of a flash boot project on IAR EWARM, after copying an application program from the primary core memory to the secondary one, executing the secondary project will abort. For the multicore debugging with flash boot mode in IAR EWARM, follow the steps below: 1. Run the primary project up to main(). 2. Software reset the secondary project. Click on the downward triangle to the right of the reset icon and select **Software**.  3. Run the secondary project.

(Continued on next page)

No. 19

Limitation	**Build errors occur in CA55 projects when install e² studio as the Current user.**
Target Device	**RZ/T2H(CA55), RZ/N2H(CA55)**
Category	**e² studio**
Description	If you install e² studio as the Current user, an error will occur when building a CA55 project. As a workaround, install e² studio as All Users.

No. 20

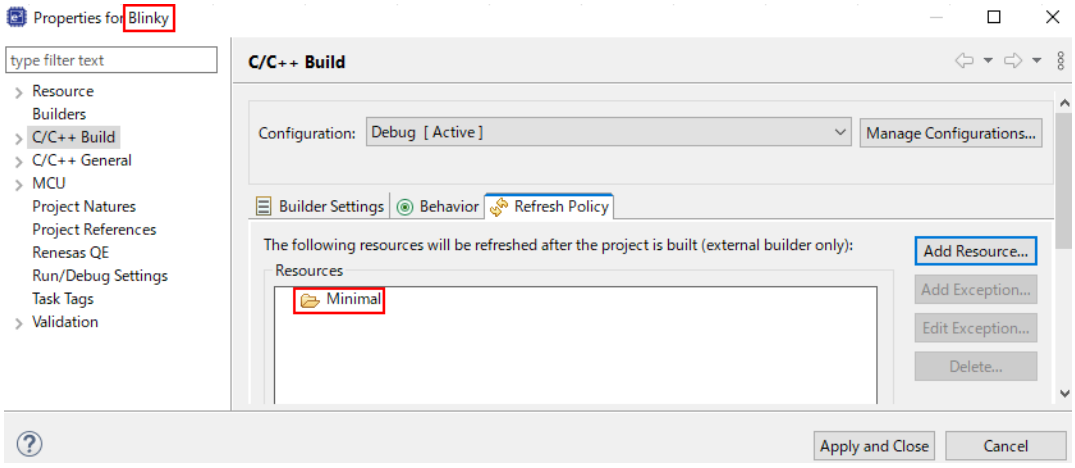
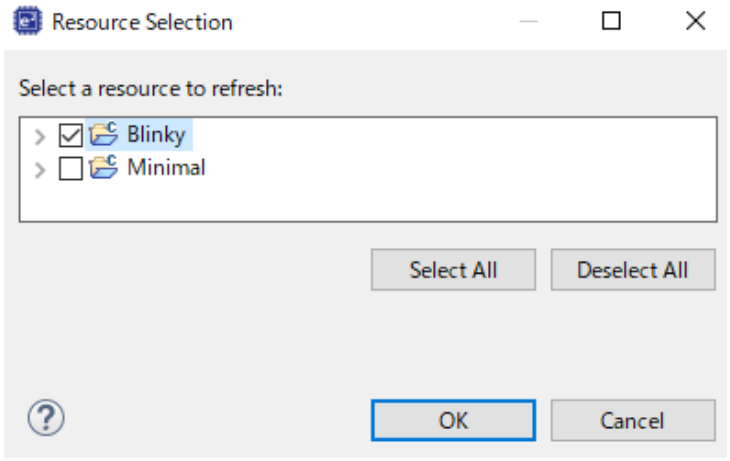
Limitation	**Wrong core name when create a project CA55 with FSP SC.**
Target Device	**RZ/N2H(CA55)**
Category	**FSP SC**
Description	When you create a project CA55 with FSP SC the device name is shown as "**CR52_0**" in buildinfo.ipcf even though you select CA55 core. 

No. 21

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

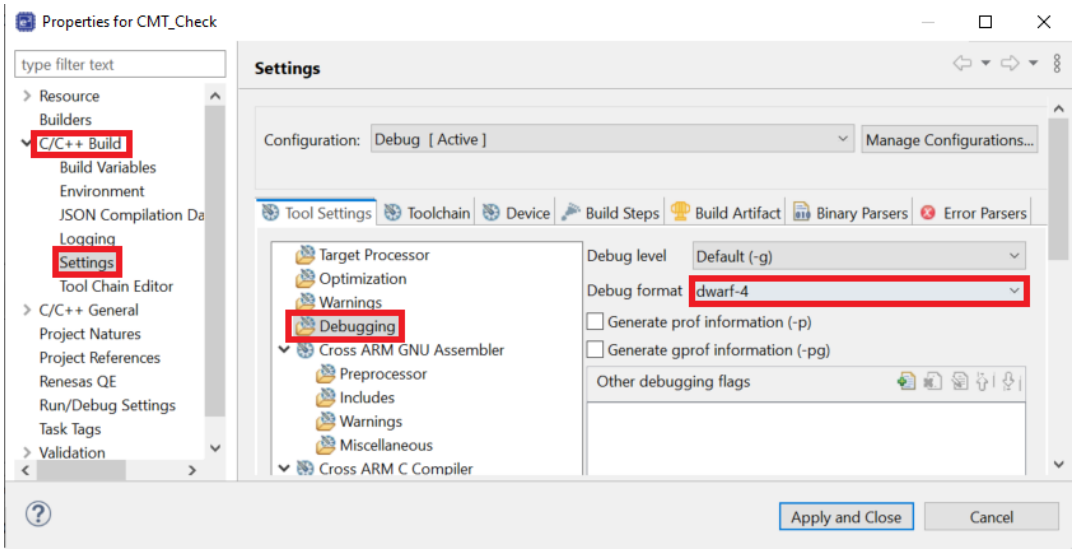
(Continued on next page)

No. 22

Limitation	**The bundle file (.sbd) may not be generated during build.**
Target Device	**RZ/T2M, RZ/T2L, RZ/T2ME, RZ/T2H, RZ/N2L, RZ/N2H**
Category	**e² studio**
Description	In e² studio, when you import and build an existing project, the bundle file (.sbd) may not be generated. As a workaround, right-click the project in Project Explorer, click **Properties** > **C/C++ Build** > **Refresh Policy**, and verify that the projects displayed in Resources match the projects listed in the upper left. If they are different, click the project in Resources and click **Delete...**.  Next, click **Add Resource...**, check the correct project, and click **OK**.  Finally, click **Apply and Close**. By making this setting, an .sbd file will be generated during the build.

(Continued on next page)

No. 23

Limitation	**When implementing CMT interrupts in RZ/T2H CR52 or RZ/N2H CR52 project, an unintended source file is displayed during debugging.**
Target Device	**RZ/T2H (CR52), RZ/N2H (CR52)**
Category	**e² studio**
Description	When you create a Blinky project for RZ/T2H CR52 or RZ/N2H CR52 project, add a CMT stack, and define a CMT callback function, the command to jump to hal_entry() during debugging may not display the hal_entry.c source file, and an unintended file may be displayed. As a workaround, right-click the project in Project Explorer, click **Properties** > **C/C++ Build** >**Settings** > **Tool Settings** > **Debugging** > **Default format**, and change "Toolchain default" to "dwarf-4". 

Appendix. How to Debug FSP Project with Flash Boot Mode

When debugging FSP project with flash boot mode (xSPI boot, NOR flash boot), the program cannot be stopped at the beginning of the user program (loader program).

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_core.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, 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

- /XXX/fsp/src/bsp/cmsis/Device/RENESAS/Source/YY/startup_core.c (XXX = rzt, rzn, YY = cr, ca)

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

Note for multiprocessing projects:

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

<table>
<tr><th>IDE</th><th>Core</th><th>Boot Mode</th><th>Position at which the loop part should be inserted.</th></tr>
<tr><td rowspan="2">e^2 studio

IAR EWARM</td><td rowspan="2">CR52</td><td>xSPI boot</td><td rowspan="2">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, #0x68bf                   \n"
        "   movt  r1, #0x478                    \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>NOR flash boot</td></tr>
<tr><td>CA55</td><td>xSPI boot</td><td>First line in system_init() function.

```
BSP_ATTRIBUTE_STACKLESS void system_init (void)
{
#if 1 // Software loops are only needed when debugging.
    __asm volatile (
        "    mov   x1, #0                       \n"
        "    movz  x2, #0x68bf                  \n"
        "    movk  x2, #0x478, LSL #16          \n"
        "software_loop:                         \n"
        "    adds  x1, x1, #1                   \n"
        "    cmp   x1, x2                       \n"
        "    b.ne   software_loop               \n"
        ::: "memory");
#endif
    /* g_bsp_software_reset_occurred = false */
    __asm volatile (
        "MOV  x0, %0    \n"
        "MOV  w1, #0    \n"
        "STRB  w1, [x0] \n"
        ::"r" (&g_bsp_software_reset_occurred) : "memory");
```

</td></tr>
</table>

Note:

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

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

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

* In CA55 core, opcodes are different.

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

When RZ/N2H in xspi boot mode, it's required to override the .board files for RZ/T2H *(FlashRZT2H_EVB_A55.board, FlashRZT2H_EVB_R52.board)* to flash the board.

Please reselect the execution path by following the steps bellow:

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

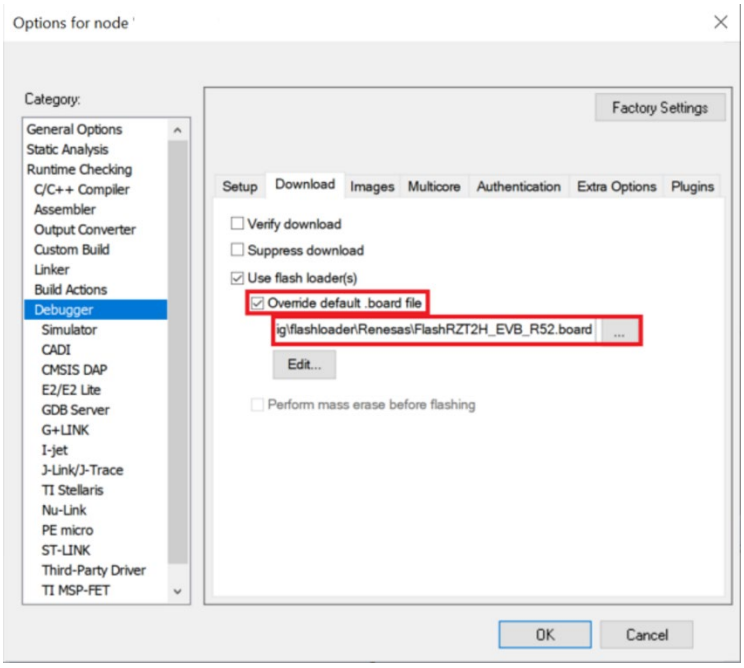


Figure 109 : Project Options – Debugger (RZ/N2H Only)

Appendix. How to Erase Flash Memory

If you run 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 the board using J-Link Commander, execute the following steps.

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

   ii) Open the J-Link Commander.

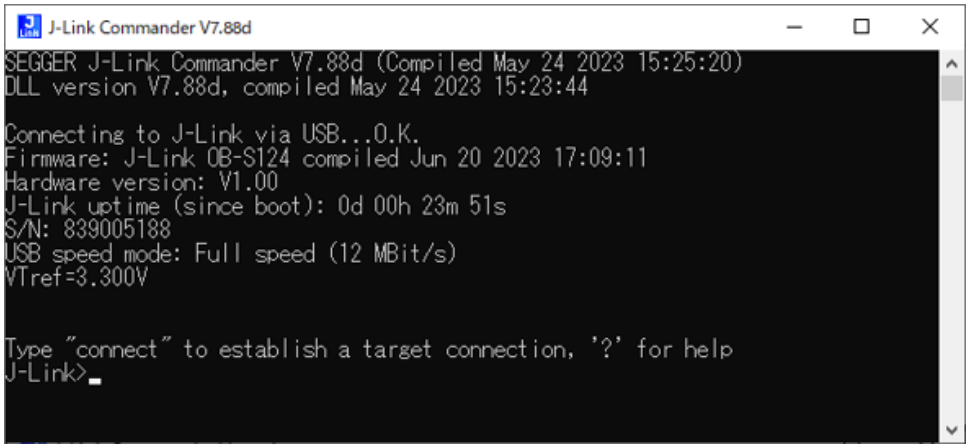


Figure 110 : Launch J-Link Commander

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

Next, specify the connection conditions as follows.

- Device> (Device type name)

Table 22 Device Type Name on Renesas Board

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

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

(Continued on next page)

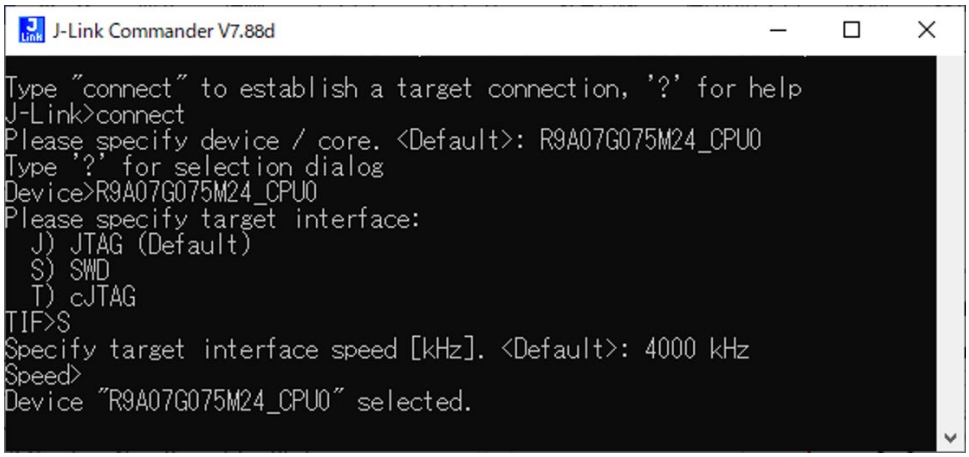


Figure 111 : Initial Setup for Connecting to the Device

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

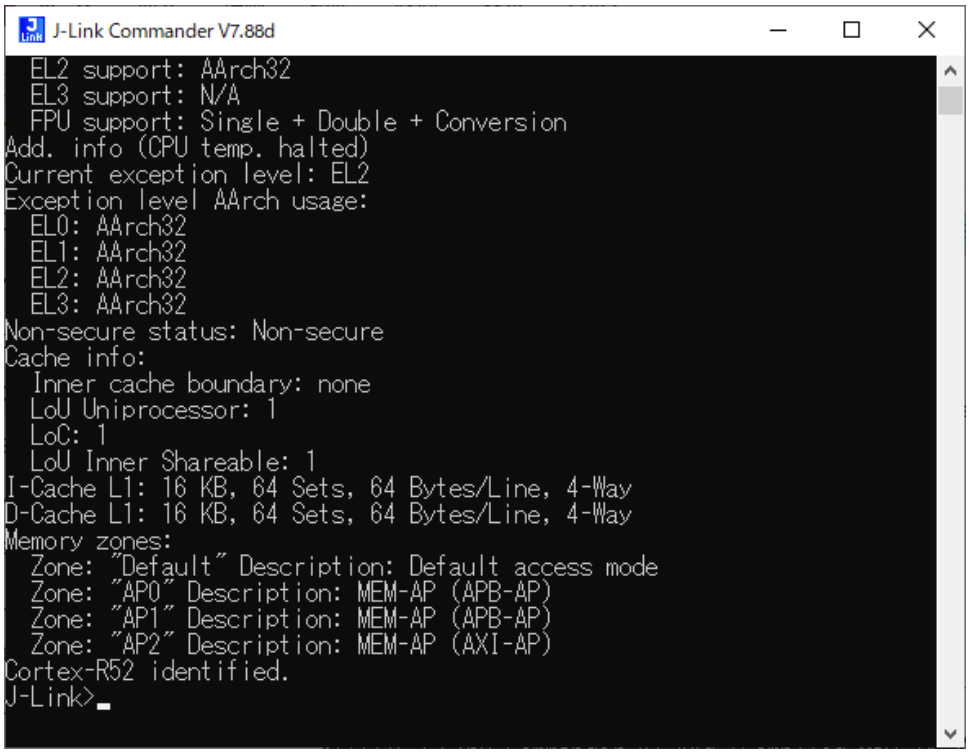


Figure 112 : Message of Device Core Identification

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 23 External Address Space to Be Used in Each Boot Mode

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

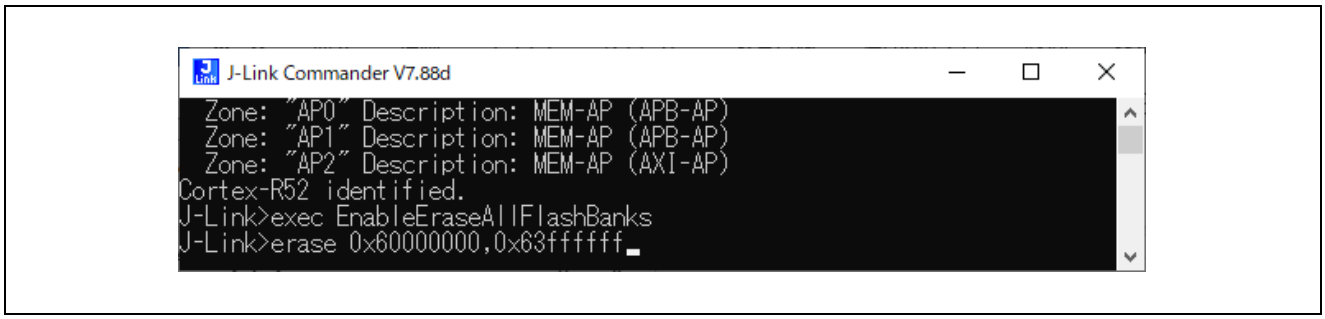


Figure 113 : Specify Erase Range

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

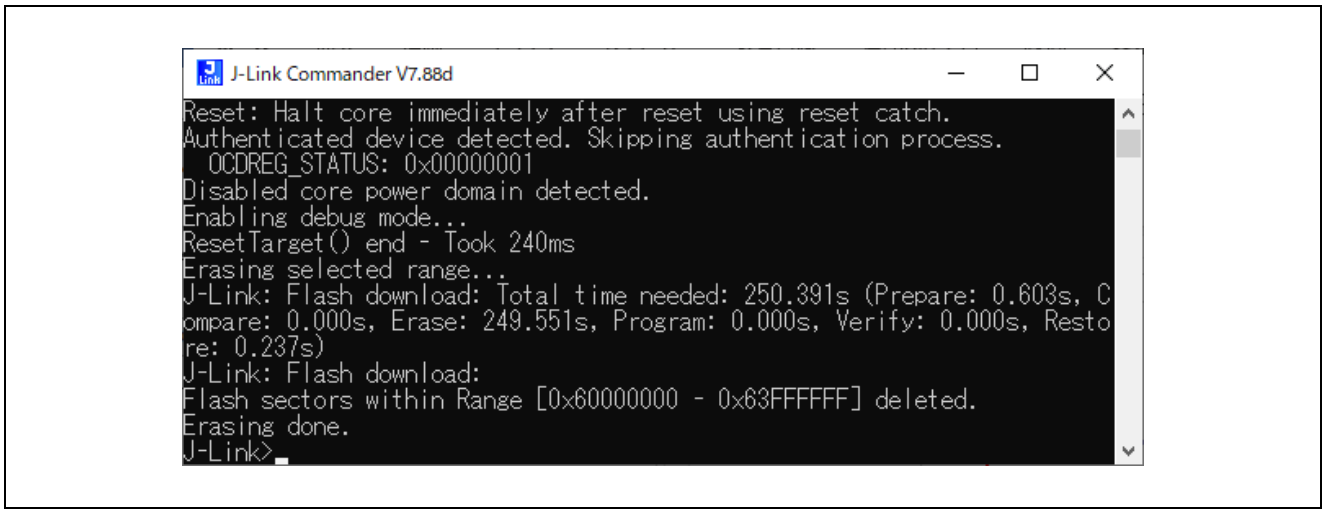


Figure 114: Message of Flash Memory Erase Complete

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

2. IAR EWARM

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

Disable asymmetric multicore setting:

a. Click **Project** > **Options…**.

b. Click **Debugger** > **Multicore** and select **Disable** in Asymmetric multicore.

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

ii) Open the workspace of a project.

xxx.eww

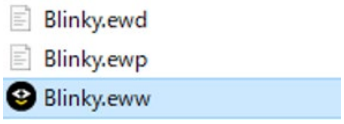


Figure 115 : Open Workspace for IAR EWARM

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

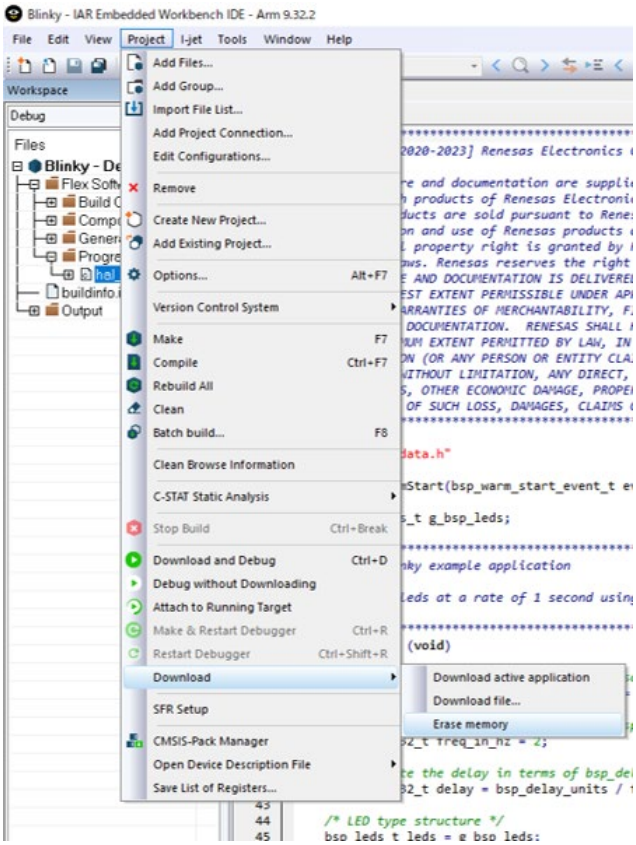


Figure 116 : Select Erase memory Command

iv) Select erase memory space.

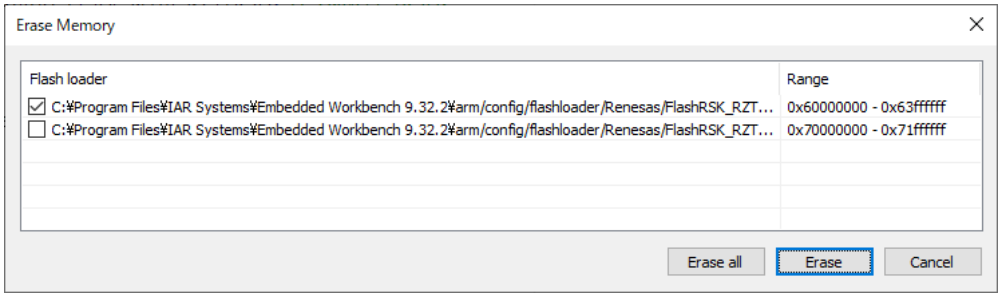


Figure 117 : Select Erase Memory Space

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

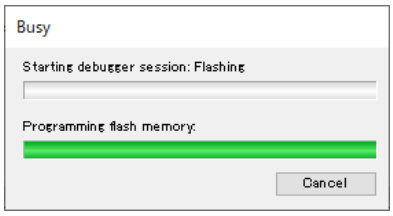


Figure 118 : Screen During Erasing

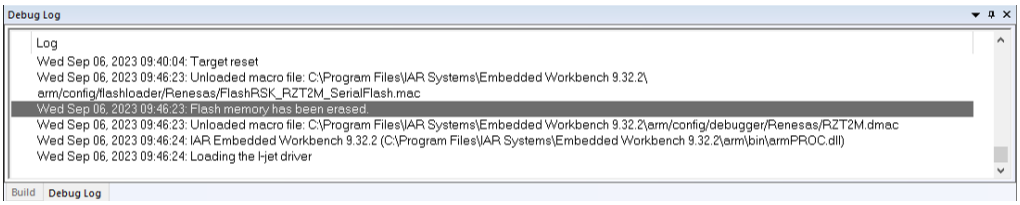


Figure 119 : Message of Flash Memory Erase Complete

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.

Note for FSP version earlier than v1.3.0:

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

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

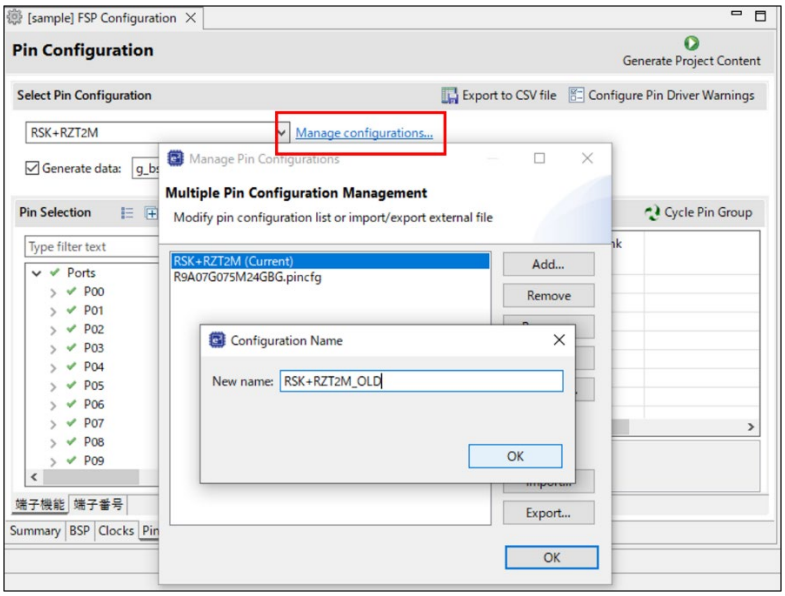


Figure 120: How to Rename Pin Configuration

(Continued on next page)

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

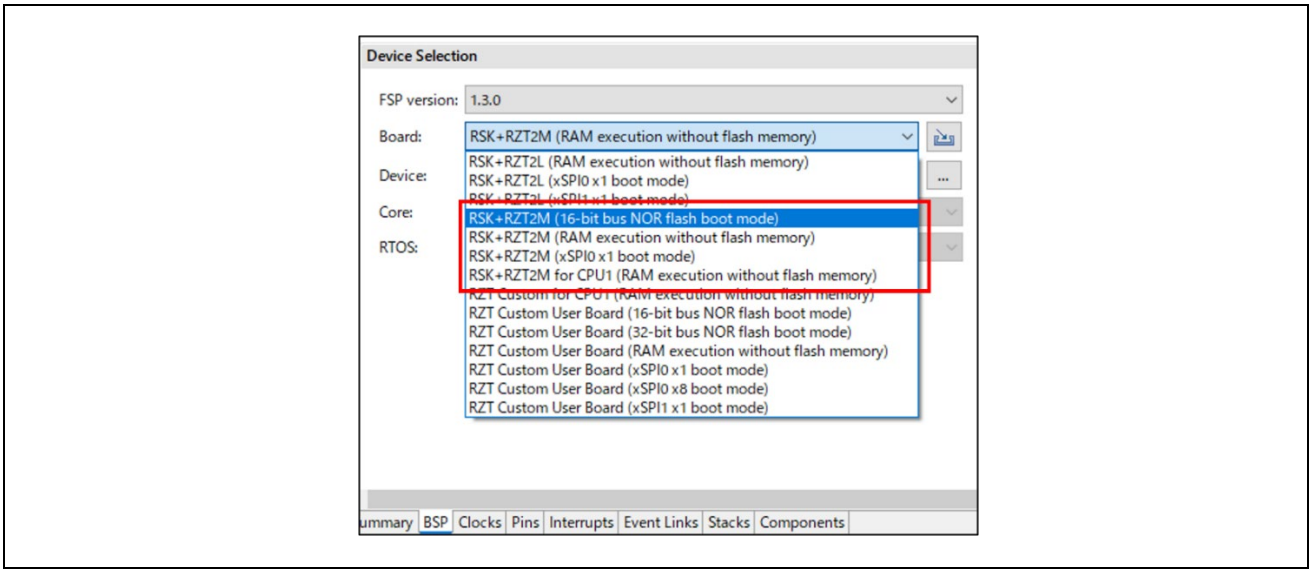


Figure 121: Change the Boot Mode in the BSP Tab

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

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

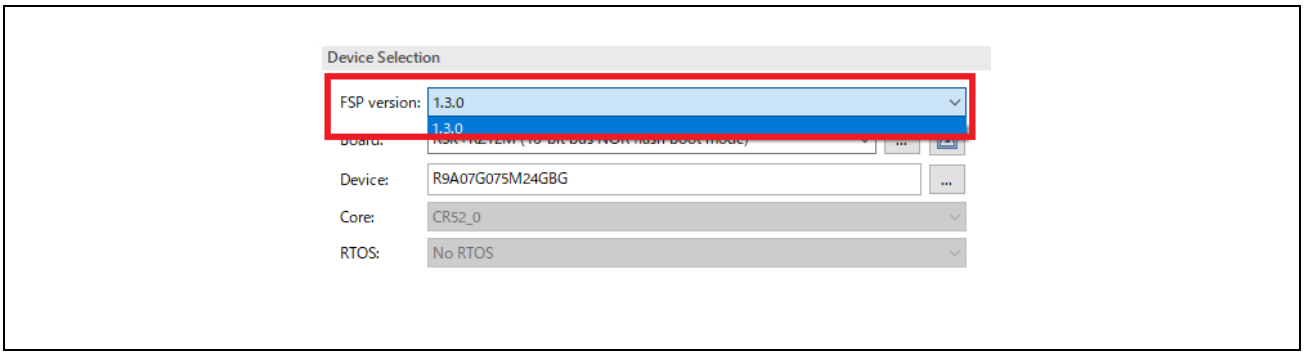


Figure 122: Reselect “FSP Version” from the Drop-down List

(Continued on next page)

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

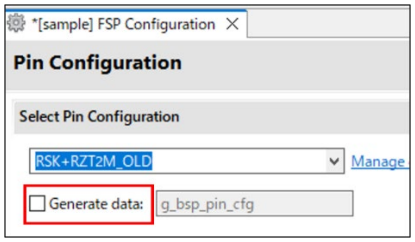


Figure 123: Uncheck Generate data in the Pins Tab

5. Select the regenerated configuration for the board.

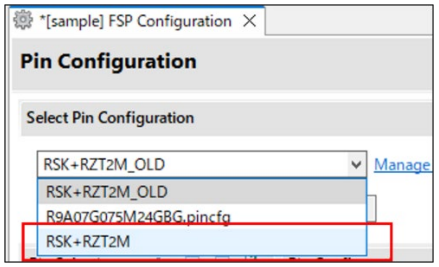


Figure 124: Select the Regenerated Configuration for the Board

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

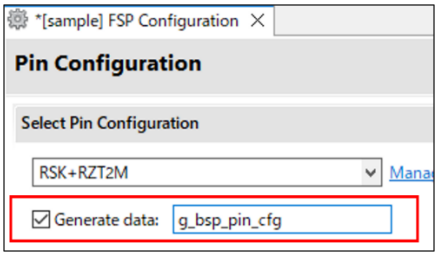


Figure 125: Check Generate data again

Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e² studio

4 Tutorial: Your First RZ/T2, RZ/N2 MPU Project – Blinky describes how to create and debug a project in e² studio for RAM execution of a single core process using CR52_0 and multiprocessing processes where CR52_0 is the primary core. This chapter describes project creation and debugging methods in e² studio applicable to other boot modes and core combinations.

If the procedure is preceded by (XXX), it is executed only if the condition is met.
(RAM exec): The boot mode used in the project is RAM execution without flash memory.
(Flash boot): The boot mode used in the project is NOR flash boot mode or xSPI flash boot mode.
(CR52): The core used in the project is CR52.
(CA55): The core used in the project is CA55.
(Same core type in N and M): In multiprocessing, the same type of cores in the N and M projects(N, M = pri(primary), sec(secondary), ter(tertiary)).
(Different core types in N and M): In multiprocessing, the different type of cores in the N and M projects(N, M = pri(primary), sec(secondary), ter(tertiary)).

For RZ/T2 FSP v3.0.0

Multiprocessing with 2 cores for RZ/T devices

1. Create a primary project according to the procedures in 4.3 Create a New Project for Blinky.
   - (RZ/T2H CA55 Core1) To blink the LED on the RZ/T2H CA55 Core1, you need to change the pin configuration of the primary project in the Smart Configurator as follows.
     i. After creating the primary project in 4.3 Create a New Project for Blinky No. 11, set the pins before clicking **Generate Project Content**.
     ii. In Pins tab of FSP configuration, click **Pin Selection** -> **Peripherals** -> **Connectivity:SDHI** -> **SDHI1**
     iii. Change the value of SD1_PWEN to None.
     iv. In Pins tab of FSP configuration, click **Pin Selection** -> **Ports** -> **P08** -> **P08_5**.
     v. Change the value of Symbolic Name to LED3.
     vi. Change the value of Mode to Output mode (Low & Not Into Input)
2. (Flash boot) Insert the loop part in startup_core.c of the primary project with reference to Appendix. How to Debug FSP Project with Flash Boot Mode.
3. Build the primary project according to the procedures in 4.4.1 Build.
4. Create a secondary project using the bundle file (.sbd) of the primary project according to the procedures in 4.3 Create a New Project for Blinky.
5. Build the secondary project according to the procedures in 4.4.1 Build.
   - (Flash boot) The following object files are output to the Debug folder of the secondary project.

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

Device	Project core	Object files	Note
RZ/T2M RZ/T2ME	CR52	secondary_CR52.o	
		secondary_noncache_CR52.o	When using a noncache sections
RZ/T2H	CR52	secondary_atcm_CR52_0.o	For CR52 CPU0
		secondary_btcm_CR52_0.o	For CR52 CPU0
		secondary_atcm_CR52_1.o	For CR52 CPU1
		secondary_btcm_CR52_1.o	For CR52 CPU1
		secondary_systemram_CR52.o	For a multi-core project with 3 or more cores
		secondary_CR52.o	When placing a program in System SRAM instead of TCM
		secondary_noncache_CR52.o	When using a noncache sections
	CA55	secondary_CA55.o	
		secondary_noncache_CA55.o	When using a noncache sections

(Continued on next page)

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

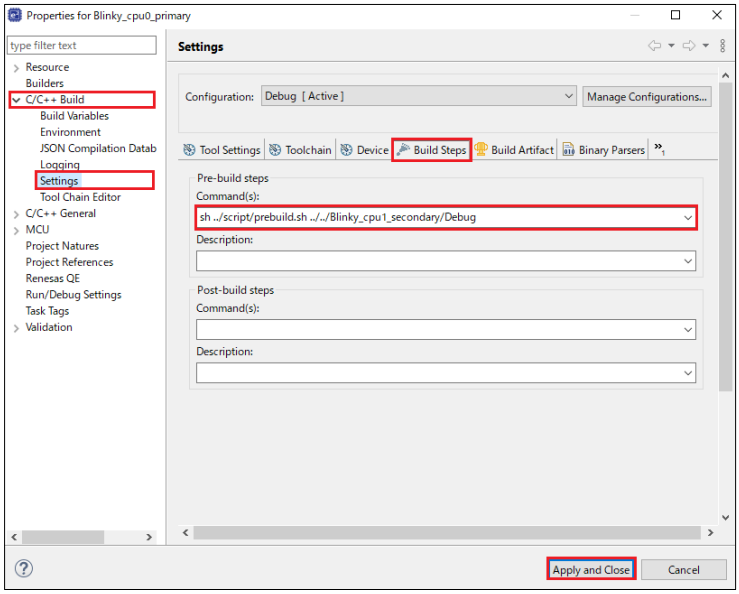


Figure 126 e² studio Build Setting for the Primary Project (Flash Boot) (Different core types in pri and sec)

7. (Flash boot) (Different core types in pri and sec) Build the primary project according to the procedures in 4.4.1 Build. The following object files are output to the Debug folder of the secondary project.

Table 25 Object files Output to the Debug Folder of the Secondary Project

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

(Continued on next page)

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

Table 26 Example of File Paths to Be Set to Other Objects

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

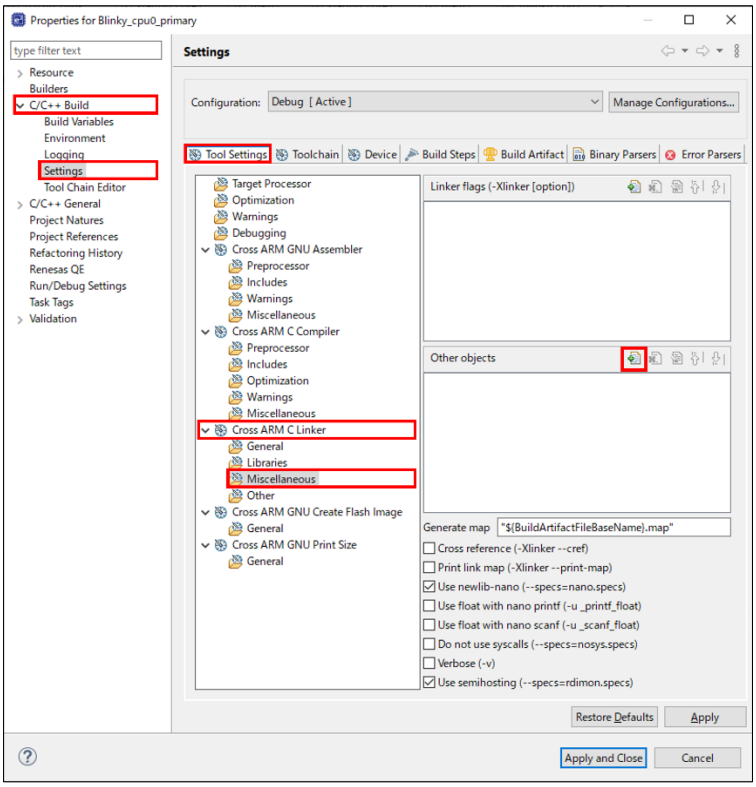


Figure 127 e² studio Build Setting for the Primary Project (Part 1)

(Continued on next page)

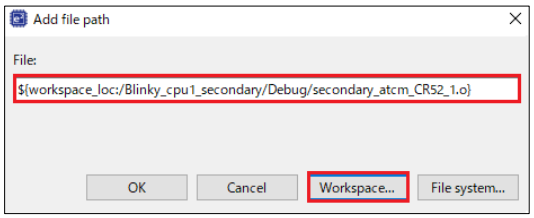


Figure 128 e² studio Build Setting for the Primary Project (Part 2)

9. (Flash boot) Build the primary project.
10. Debug the projects according to the procedures in 4.7 Debug and Run for Multiprocessing.
    - Check the debug configuration in the No. 3 procedure of 4.5.2 Debug Steps.

Table 27 e² studio Debug Configurations for Multiprocessing

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

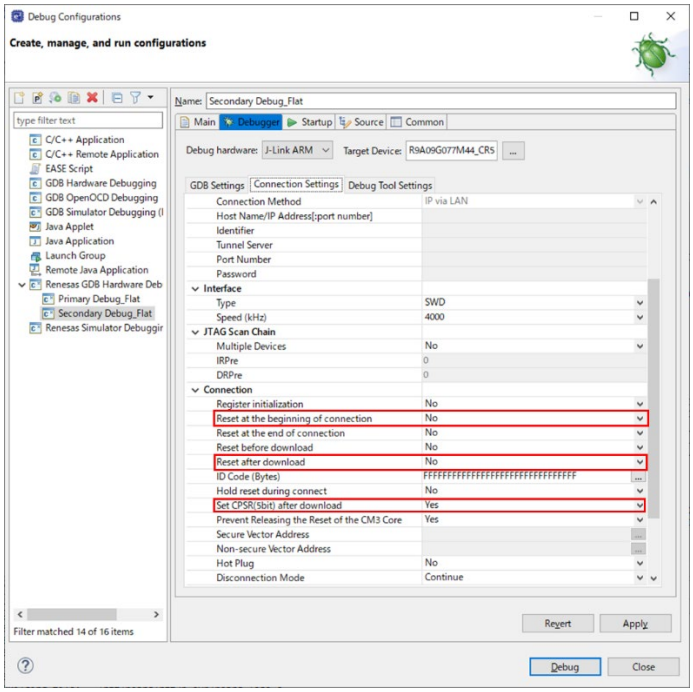


Figure 129 e² studio Debug Configurations for Multiprocessing

(Continued on next page)

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

11. When changing the project and debugging it again, follow these steps.
    i. Build the primary project (No. 3).
    ii. Build the secondary project (No. 5).
    iii. (Flash boot) Build the primary project again (No. 9).
    iv. Debug the projects (No. 10).

Multiprocessing with 3 or more cores for RZ/T devices

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

1. Create and build the primary and secondary projects according to Multiprocessing with 2 cores for RZ/T devices No. 1 to No. 6.
2. Create a tertiary project using the bundle file (.sbd) of the secondary project according to the procedures in 4.3 Create a New Project for Blinky.
3. Build the tertiary project according to the procedures in 4.4.1 Build.
   - (Flash boot) The following object files are output to the Debug folder of the secondary project.

Table 28 Object files Output to the Debug Folder of the Secondary Project

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

4. (Flash boot) (Different core types in sec and ter) The following additional properties must be set.
   i. In the Properties window of the secondary project, click **C/C++ Build** > **Settings** > **Build Steps**.
   ii. Add **Command(s)** at **Pre-build steps**.
   sh ../script/prebuild.sh ../../[the tertiary project name]/Debug
5. (Flash boot) (Different core types in sec and ter) Build the secondary project according to the procedures in 4.4.1 Build. The following object files are output to the Debug folder of the tertiary project.

Table 29 Object files Output to the Debug Folder of the Tertiary Project

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

(Continued on next page)

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

Table 30 Example of File Paths to Be Set to Other Objects for the Secondary Project

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

* Object files for TCM such as secondary_atcm_CR52_1.o will not work properly if they are imported into a project other than the primary project. Even object files from tertiary projects must be imported into the primary project.

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

Table 31 Example of File Path to Be Set to Other Objects for the Primary Project

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

(Continued on next page)

Note:

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

i. In the Properties window of the primary project, click **C/C++ Build** > **Settings** > **Build Steps**.
ii. Add **Command(s)** at **Pre-build steps**.
sh ../script/prebuild.sh ../../[the secondary project name]/Debug && sh ../script/prebuild.sh ../../[the tertiary project name]/Debug

9. (Flash boot) Build the primary project.
10. Debug the projects according to the procedures in Multiprocessing with 2 cores for RZ/T devices No. 10. Connections are made in the order primary, secondary, tertiary. The tertiary project is connected in the same procedure as secondary project.
11. When changing the project and debugging it again, follow these steps.
    i. Build the primary project (Multiprocessing with 2 cores for RZ/T devices No. 3).
    ii. Build the secondary project (Multiprocessing with 2 cores for RZ/T devices No. 5).
    iii. Build the tertiary project (Multiprocessing with 3 or more cores for RZ/T devices No. 3).
    iv. (Flash boot) Build the secondary project (Multiprocessing with 3 or more cores for RZ/T devices No. 7).
    v. (Flash boot) Build the primary project again (Multiprocessing with 3 or more cores for RZ/T devices No. 9).
    vi. Debug the projects (Multiprocessing with 3 or more cores for RZ/T devices No. 10).

For RZ/N2 FSP v2.2.0

Multiprocessing with 2 cores for RZ/N devices

1. Create a primary project according to the procedures in 4.3 Create a New Project for Blinky.
2. (Flash boot) Insert the loop part in startup_core.c of the primary project with reference to Appendix. How to Debug FSP Project with Flash Boot Mode.
3. Build the primary project according to the procedures in 4.4.1 Build.
4. Create a secondary project using the bundle file (.sbd) of the primary project according to the procedures in 4.3 Create a New Project for Blinky.
5. (RAM exec) Build the secondary project according to the procedures in 4.4.1 Build.
6. (Flash boot) (Same core type in pri and sec) Build the secondary project according to the procedures in 4.4.2 Build for Multiprocessing No. 2. The following additional properties must be set.
   i. In the Properties window, click **C/C++ Build** > **Settings** > **Build Steps**.
   ii. Add **Command(s)** at **Post-build steps**.
   - (CR52)
     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
   - (CA55)
     aarch64-none-elf-objcopy -I elf64-littleaarch64 -O binary ${ProjName}.elf secondary.bin && aarch64-none-elf-objcopy -I binary -O elf64-littleaarch64 -B aarch64 --rename-section .data=.secondary,alloc,data,readonly,load,contents secondary.bin secondary.o

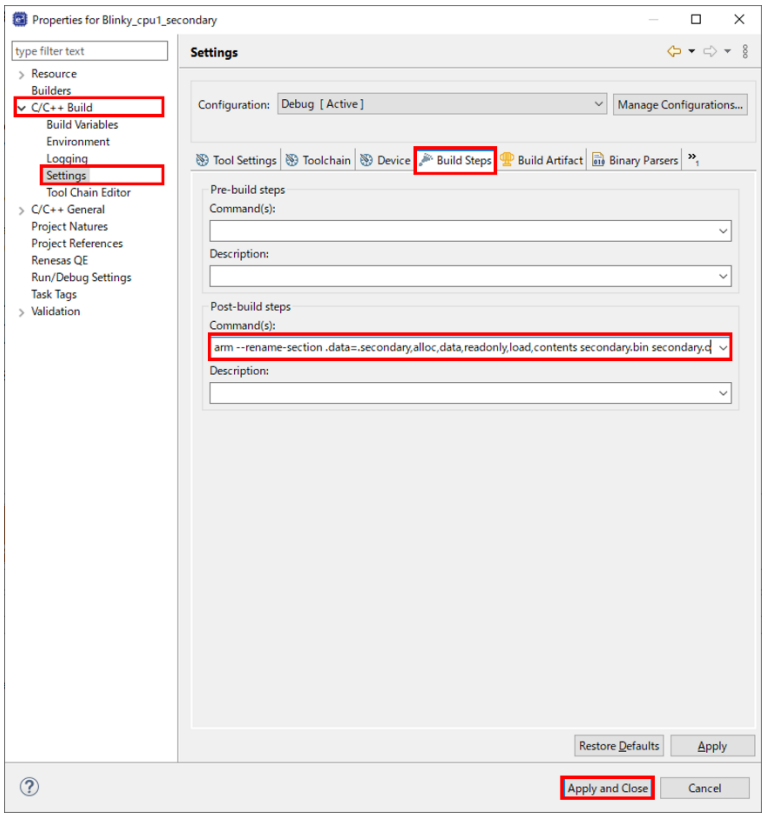


Figure 130 e² studio Build Setting for the Secondary Project (Flash Boot) (Same core type in pri and sec)

(Continued on next page)

7. (Flash boot) (Different core types in pri and sec) Create an object file and set it to the secondary project.
   i. Build the secondary project according to the procedures in 4.4.2 Build for Multiprocessing No. 2. The following additional properties must be set.
      a. In the Properties window of the secondary project, click **C/C++ Build** > **Settings** > **Build Steps**.
      b. Add **Command(s)** at **Post-build steps**.
         - (The primary project uses CR52, and the secondary project uses CA55)
           aarch64-none-elf-objcopy -I elf64-littleaarch64 -O binary ${ProjName}.elf secondary.bin
         - (The primary project uses CA55, and the secondary project uses CR52)
           arm-none-eabi-objcopy -I elf32-littlearm -O binary ${ProjName}.elf secondary.bin
   ii. Since secondary.bin is output to the Debug folder of the secondary project, move it to the Debug folder of the primary project.

   iii. Build the primary project with the following additional properties.
      a. In the Properties window of the primary project, click **C/C++ Build** > **Settings** > **Build Steps**.
      b. Add **Command(s)** at **Post-build steps**.
         - (The primary project uses CR52, and the secondary project uses CA55)
           arm-none-eabi-objcopy -I binary -O elf32-littlearm -B arm --rename-section .data=.secondary,alloc,data,readonly,load,contents secondary.bin secondary.o
         - (The primary project uses CA55, and the secondary project uses CR52)
           aarch64-none-elf-objcopy -I binary -O elf64-littleaarch64 -B aarch64 --rename-section .data=.secondary,alloc,data,readonly,load,contents secondary.bin secondary.o
   iv. Since secondary.o is output to the Debug folder of the primary project, move it to the Debug folder of the secondary project.
8. (Flash boot) Set the following additional properties to the primary project.
   i. In the Properties window of the primary project, click **C/C++ Build** > **Settings** > **Tool Settings > Cross ARM C Linker > Miscellaneous**.
   ii. Set a file path of secondary.o in the secondary project to **Other objects**.
   e.g. ${workspace_loc:/Blinky_secondary/Debug/secondary.o}

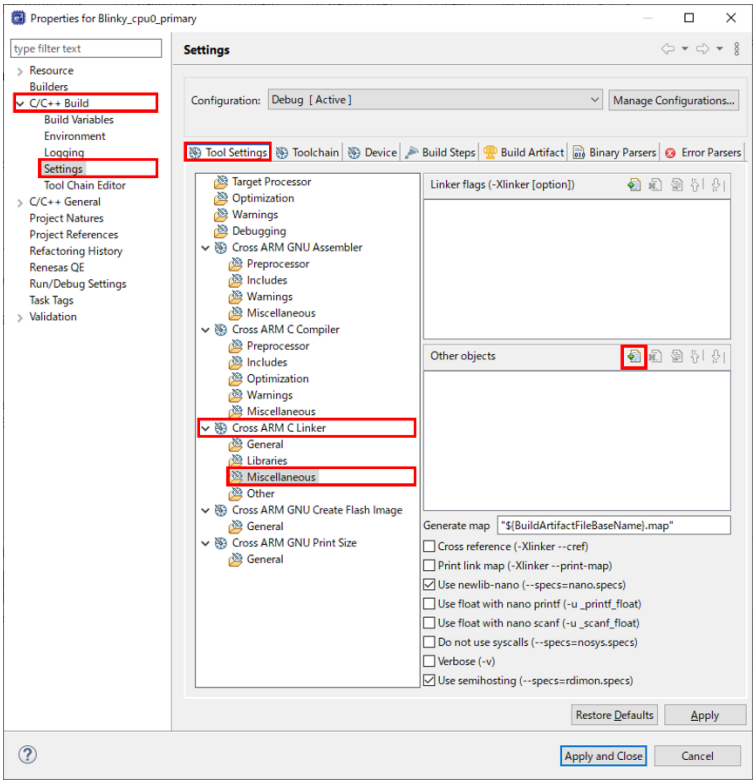


Figure 131 e² studio Build Setting for the Primary Project (Part 1)

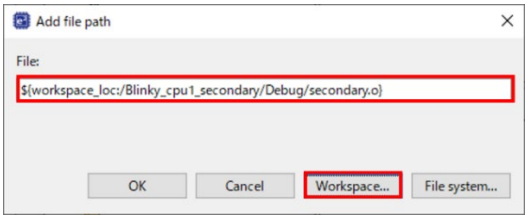


Figure 132 e² studio Build Setting for the Primary Project (Part 2)

9. Build the primary project.
10. Debug the projects according to the procedures in 4.7 Debug and Run for Multiprocessing.
    - (CR52 CPU0) When using the CR52 CPU0 core for the secondary or later project, set the additional debug connection settings:
      - **Debugger > Connection Settings > Connection**
        - **Reset at the beginning of connection : No**
        - **Set CPSR(5bit) after download: Yes**

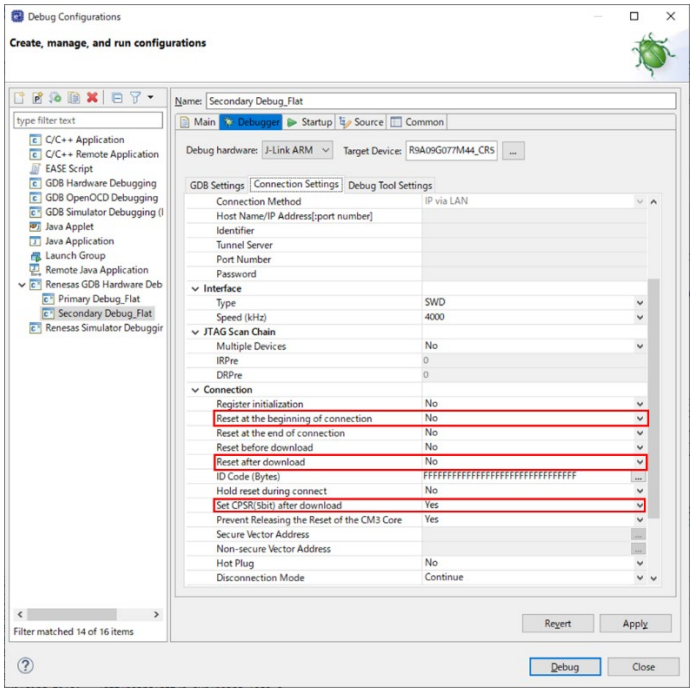


Figure 133 e² studio Debug Configuration for the Secondary or later Project (CR52 CPU0)

(Continued on next page)

- (Flash boot) 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.
  - Change Debug Configuration settings in the No. 3 procedure of 4.5.2 Debug Steps.
    - **Debugger** > **Connection Settings** > **Connection**
      - **(Primary project) Reset after download: Yes**
      - **(Secondary project) Reset after download: No (default)**
      - **Set CPSR(5bit) after download: No (default)**

11. When changing the project and debugging it again, follow these steps.
    i. Build the primary project (No. 3).
    ii. (RAM exec) Build the secondary project (No. 5).
    iii. (Flash boot) (Same core type in pri and sec) Build the secondary project (No. 6).
    iv. (Flash boot) (Different core types in pri and sec) Create an object file and set it to the secondary project (No. 7).
    v. Build the primary project again (No. 9).
    vi. Debug the projects (No. 10).

Multiprocessing with 3 or more cores for RZ/N devices

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

1. Create and build the primary and secondary projects according to Multiprocessing with 2 cores for RZ/N devices No. 1 to No. 7.
2. Create a tertiary project using the bundle file (.sbd) of the secondary project according to the procedures in 4.3 Create a New Project for Blinky.
3. (RAM exec) Build the tertiary project according to the procedures in 4.4.1 Build.
4. (Flash boot) (Same core type in pri and ter) Build the tertiary project according to the procedures in 4.4.2 Build for Multiprocessing No. 2. The following additional properties must be set.
   i. In the Properties window, click **C/C++ Build > Settings > Build Steps**.
   ii. Add **Command(s)** at **Post-build** steps.
   - (CR52)

     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
   - (CA55)

     aarch64-none-elf-objcopy -I elf64-littleaarch64 -O binary ${ProjName}.elf secondary.bin && aarch64-none-elf-objcopy -I binary -O elf64-littleaarch64 -B aarch64 --rename-section .data=.secondary,alloc,data,readonly,load,contents secondary.bin secondary.o
5. (Flash boot) (Different core types in pri and ter) Create an object file and set it to the tertiary project.
   i. Build the tertiary project according to the procedures in 4.4.2 Build for Multiprocessing No. 2. The following additional properties must be set.
      a. In the Properties window of the tertiary project, click **C/C++ Build > Settings > Build Steps**.
      b. Add **Command(s)** at **Post-build** steps.
      - (The primary project uses CR52, and the tertiary project uses CA55)

        aarch64-none-elf-objcopy -I elf64-littleaarch64 -O binary ${ProjName}.elf secondary.bin
      - (The primary project uses CA55, and the tertiary project uses CR52)

        arm-none-eabi-objcopy -I elf32-littlearm -O binary ${ProjName}.elf secondary.bin
   ii. Since secondary.bin is output to the Debug folder of the tertiary project, move it to the Debug folder of the primary project.
   iii. Build the primary project with the following additional properties.
      a. In the Properties window of the primary project, click **C/C++ Build > Settings > Build Steps**.

b. Add **Command(s)** at **Post-build** steps.

- (The primary project uses CR52, and the tertiary project uses CA55)

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

- (The primary project uses CA55, and the tertiary project uses CR52)

  aarch64-none-elf-objcopy -I binary -O elf64-littleaarch64 -B aarch64 --rename-section .data=.secondary,alloc,data,readonly,load,contents secondary.bin secondary.o

iv. Since secondary.o is output to the Debug folder of the primary project, move it to the Debug folder of the tertiary project.

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

   e.g. ${workspace_loc:/Blinky_tertiary/Debug/secondary.o}
7. Build the secondary project.
8. (Flash boot) Set the following additional properties to the primary project (Multiprocessing with 2 cores for RZ/N devices No. 8)
9. Build the primary project.
10. Debug the projects according to the procedures in Multiprocessing with 2 cores for RZ/N devices No. 10. Connections are made in the order primary, secondary, tertialy. The tertialy project is connected in the same procedure as secondary project.
11. When changing the project and debugging it again, follow these steps.
    i. Build the primary project (Multiprocessing with 2 cores for RZ/N devices No. 3).
    ii. (RAM exec) Build the secondary project (Multiprocessing with 2 cores for RZ/N devices No. 5).
    iii. (RAM exec) Build the tertiary project (Multiprocessing with 3 or more cores for RZ/N devices No. 3).
    iv. (Flash boot) (Same core type in pri and ter) Build the tertiary project (Multiprocessing with 3 or more cores for RZ/N devices No. 4).
    v. (Flash boot) (Different core types in pri and ter) Create an object file and set it to the tertiary project (Multiprocessing with 3 or more cores for RZ/N devices No. 5).
    vi. Build the secondary project (Multiprocessing with 3 or more cores for RZ/N devices No. 7).
    vii. Build the primary project again (Multiprocessing with 3 or more cores for RZ/N devices No. 9).
    viii. Debug the project (Multiprocessing with 3 or more cores for RZ/N devices No. 10)

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

5.3 Using FSP SC with IAR EWARM describes how to create in FSP SC and debug a project in IAR EWARM for RAM execution of a single core process using CR52_0 and multiprocessing processes where CR52_0 is the primary core. This chapter describes project creation in FSP SC and debugging methods in IAR EWARM applicable to other boot modes and core combinations.

If the procedure is preceded by (XXX), it is executed only if the condition is met.
(RAM exec): The boot mode used in the project is RAM execution without flash memory.
(Flash boot): The boot mode used in the project is NOR flash boot mode or xSPI flash boot mode.
(CR52): The core used in the project is CR52.
(CA55): The core used in the project is CA55.

For RZ/T2 FSP v3.0.0

Multiprocessing with 2 cores for RZ/T devices

1. Create projects according to the procedures in 5.3.2 Create a New Project.
   - (RZ/T2H CA55 Core1) To blink the LED on the RZ/T2H CA55 Core1, you need to change the pin configuration of the primary project in the FSP SC as follows.
     i. In 5.3.2 Create a New Project No. 14, set the pins before clicking **Generate Project Content**.
     ii. In Pins tab of FSP configuration, click **Pin Selection** -> **Peripherals** -> **Connectivity:SDHI** -> **SDHI1**
     iii. Change the value of SD1_PWEN to None.
     iv. In Pins tab of FSP configuration, click **Pin Selection** -> **Ports** -> **P08** -> **P08_5**.
     v. Change the value of Symbolic Name to LED3.
     vi. Change the value of Mode to Output mode (Low & Not Into Input)
2. (Flash boot) Insert the loop part in startup_core.c of the primary project with reference to Appendix. How to Debug FSP Project with Flash Boot Mode.
3. Build the primary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 1.
4. Create a secondary project using the bundle file (.sbd) of the primary project according to the procedures in 5.3.2 Create a New Project.
5. The following additional properties must be set.
   i. Click **Project** > **Options…**.
   ii. (Flash boot) Click **Build Actions** > **Build Actions Configuration** > **New** and set it as follows:
      a. Command line
         - ielftool --bin-multi=0x0-0x7FFF;0x102000-0x10FFFF;__region_CACHE_start__-__region_CACHE_end__;__region_NONCACHE_INIT_start__-__region_NONCACHE_INIT_end__ $TARGET_BPATH$.out $TARGET_BPATH$.bin
      b. Build order
         - Run after linking

(Continued on next page)

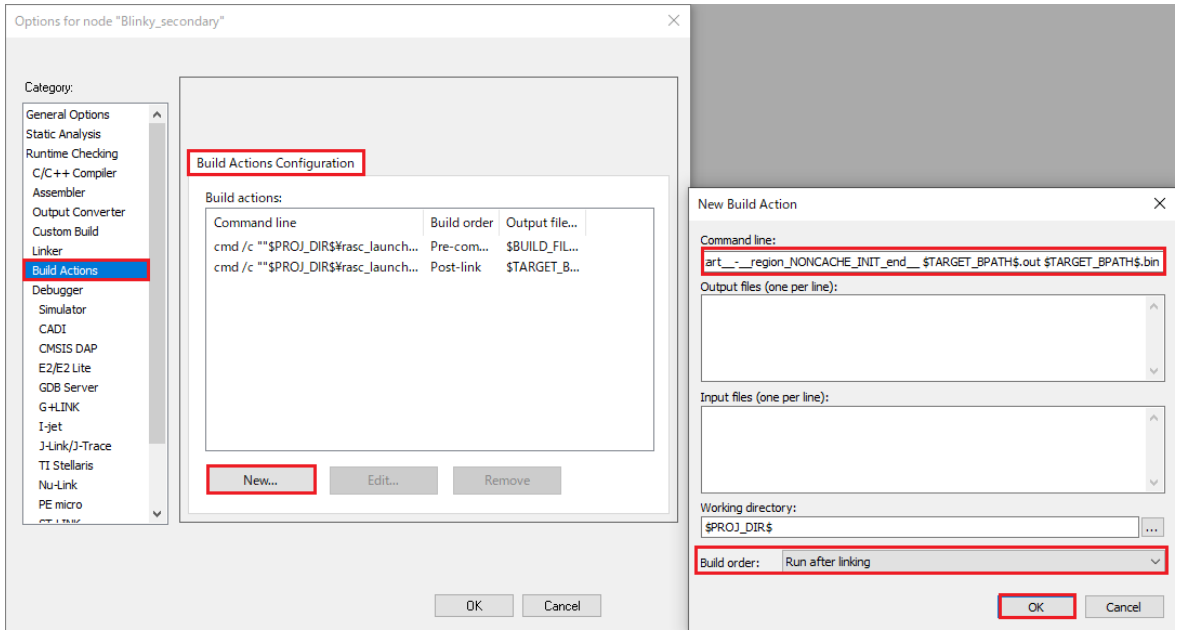


Figure 134 IAR EWARM Project Options for the Secondary Project in Flash Boot Mode

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

Table 32 Binary Files Output to the Debug\Exe folder of the Secondary Project

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

xxxxx: the secondary project name

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

zzzzz: The start address of the noncache section of System SRAM

7. (Flash boot) Set the following additional options to the primary project.
   i. Click **Project** > **Options…**.
   ii. Click **Linker** > **Input** and set it as follows:

Table 33 Example of Set to "Keep symbols" and "Raw binary image"

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

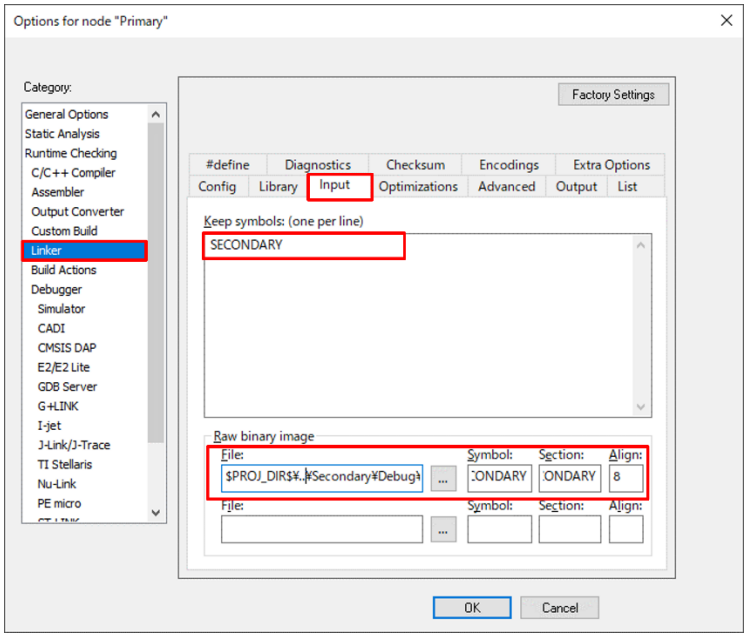


Figure 135 IAR EWARM Project Options for the Primary Project in Flash Boot Mode

Note:

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

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

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

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

- Keep symbols: (one per line)
  - Set the same as **symbol** of the **Command line options** set in **Extra Options**.
    e.g. SECONDARY_ATCM_CR521

(Continued on next page)

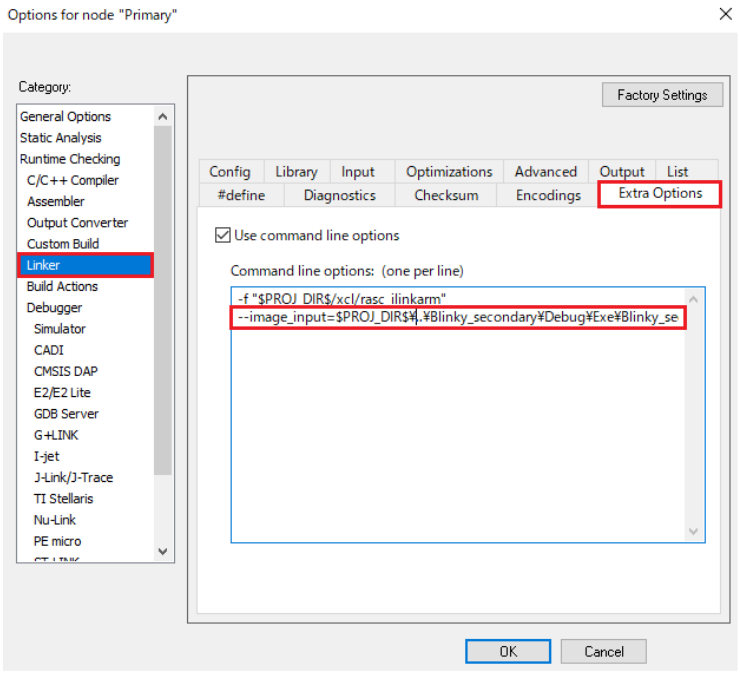


Figure 136 IAR EWARM Project Options for the Primary Project in Flash Boot Mode (Extra Options)

8. (Flash boot) Build the primary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 3.
9. (Flash boot) In the primary project, click **Project** > **Download** > **Download file…** and select out file of the primary project.
   e.g. $PROJ_DIR$\..\Blinky_primary\Debug\Exe\Blinky_primary.out

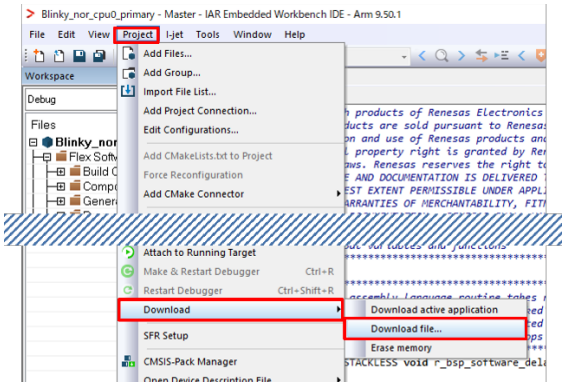


Figure 137 IAR EWARM Download File

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

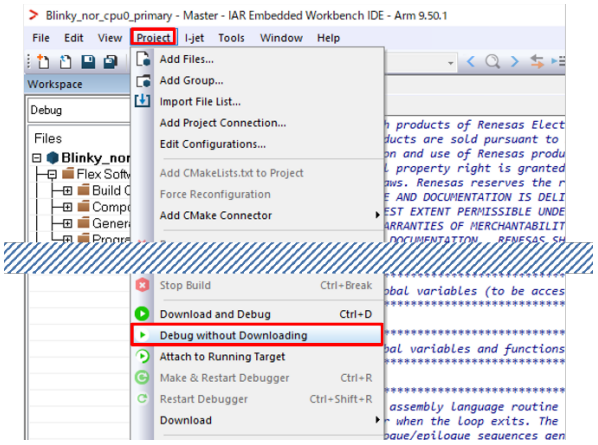


Figure 138 IAR EWARM Debug without Downloading

13. When changing the project and debugging it again, follow these steps.
    i. (Flash boot) Disable asymmetric multicore setting.
        a. Click **Project** > **Options…**.
        b. Click **Debugger** > **Multicore** and select **Disable** in Asymmetric multicore.
    ii. Build the primary project (No. 3).
    iii. Build the secondary project (No. 6).
    iv. (Flash boot) Build the primary project again (No. 8).
    v. (Flash boot) Download the file (No. 9).
    vi. (Flash boot) Enable asymmetric multicore setting (No. 10).
    vii. (RAM exec) Debug the projects (No. 11).
    viii. (Flash boot) Debug the projects (No. 12).

Multiprocessing with 3 or more cores for RZ/T devices

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

1. Create and build the primary and secondary projects according to Multiprocessing with 2 cores for RZ/T devices No. 1 to No. 6.
2. Create a tertiary project using the bundle file (.sbd) of the secondary project according to the procedures in 5.3.2 Create a New Project.
3. Build the tertiary project according to Multiprocessing with 2 cores for RZ/T device No. 5 and No. 6. The project option settings for the tertiary project are the same as for the secondary project.
4. (Flash boot) Set the following additional options to the secondary project.
   i. Click **Project** > **Options…**.
   ii. Click **Linker** > **Input** and set it as follows:

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

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

***** Binary files for TCM such as Blinky_secondary_0x0.bin will not work properly if they are imported into a project other than the primary project. Even binary files from tertiary projects must be imported into the primary project.

5. (Flash boot) Build the secondary project according to the procedures in 5.3.3.1 Build.
6. (Flash boot) Set the following additional options to the primary project according to Multiprocessing with 2 cores for RZ/T device No. 7.
   i. Click **Project** > **Options…**.
   ii. Click **Linker** > **Input** and set it as follows:

(Continued on next page)

Table 35 Example of Set to "Keep symbols" and "Raw binary image" for the Primary Project

Device	Primary project core	Secondary project core	Tertiary project core	Keep symbols (one per line), Symbol, Section	File	Align	Note
RZ/T2H	**CA55 Core0**	CR52 CPU0	CR52 CPU1	SECONDARY_ATCM_CR520	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x0.bin	8	
				SECONDARY_BTCM_CR520	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x102000.bin	8	
				SECONDARY_ATCM_CR521	$PROJ_DIR$\..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x0.bin	8	
				SECONDARY_BTCM_CR521	$PROJ_DIR$\..\Blinky_tertiary\Debug\Exe\Blinky_tertiary-0x102000.bin	8	
				SECONDARY_NONCACHE	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if file is output
	CR52 CPU0	CA55 Core0	CA55 Core1	SECONDARY	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10000000.bin	8	
				SECONDARY_NONCACHE	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if file is output
	CR52 CPU0	CR52 CPU1	CA55 Core0	SECONDARY_ATCM_CR521	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x0.bin	8	
				SECONDARY_BTCM_CR521	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x102000.bin	8	
				SECONDARY	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10000000.bin	8	
				SECONDARY_NONCACHE	$PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary-0x10180040.bin	8	Import only if file is output

(Continued on next page)

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

   multicore_setup.xml

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

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

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

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

   - <name> Arbitrary name.
   - <workspace> Location of workspace starting from the primary project location. The primary project only set “WS_PATH”.
   - <project> Project name. The primary project only set “$PROJ_PATH$”.
   - <config> Debug
   - <numberOfCores> 1
   - (Except the primary project) <attachToRunningTarget> false

8. (Flash boot) Build the primary project according to the procedures in 5.3.3.1 Build.
9. (Flash boot) In the primary project, click **Project** > **Download** > **Download file…** and select out file of the primary project according to Multiprocessing with 2 cores for RZ/T device No. 9.
10. Enable settings for multicore debugging.
    i. Open the primary project and close the secondary and later projects on IAR EWARM.
    ii. Set the following in the primary project before debugging:
        a. Click **Project** > **Options…**.
        b. Click **Debugger** > **Multicore** and check the setting value of **Symmetric multicore** and set the following contents in **Asymmetric multicore**.
           - Symmetric multicore
             - Number of cores: 1
           - Asymmetric multicore
             - Advanced
               - Session configuration: $PROJ_DIR$\multicore_setup.xml
11. (RAM exec) Debug the projects according to the procedures in 5.3.5 Debug for Multiprocessing No. 3 and after.

12. (Flash boot) Click **Project** > **Debug without Downloading** of the primary project to debug. Debug the projects according to the procedures in 5.3.5 Debug for Multiprocessing No. 4 and after. As shown in Figure 139, it is according to the setting in multicore_setup.xml. The primary project is 0, the secondary project is 1, and the tertiary project is 2.

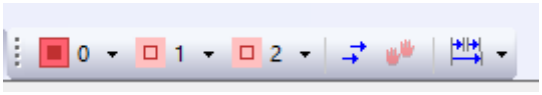


Figure 139 IAR EWARM Running Projects

13. When changing the project and debugging it again, follow these steps.
    i. (Flash boot) Disable asymmetric multicore setting.
        a. Click **Project** > **Options…**.
        b. Click **Debugger** > **Multicore** and select **Disable** in Asymmetric multicore.
    ii. Build the primary project (Multiprocessing with 2 cores for RZ/T device No. 3).
    iii. Build the secondary project (Multiprocessing with 2 cores for RZ/T device No. 6).
    iv. Build the tertiary project (Multiprocessing with 3 or more cores for RZ/T devices No. 1)
    iii. (Flash boot) Build the secondary project again (Multiprocessing with 2 cores for RZ/T device No. 6).
    iv. (Flash boot) Build the primary project again (Multiprocessing with 2 cores for RZ/T device No. 1).
    v. (Flash boot) Download the file (Multiprocessing with 2 cores for RZ/T device No. 9).
    vi. (Flash boot) Enable asymmetric multicore setting (Multiprocessing with 3 or more cores for RZ/T devices No. 10)
    vii. (RAM exec) Debug the projects (Multiprocessing with 3 or more cores for RZ/T devices No. 11)
    viii. (Flash boot) Debug the projects (Multiprocessing with 3 or more cores for RZ/T devices No. 12)

For RZ/N2 FSP v2.2.0

Multiprocessing with 2 cores for RZ/N devices

1. Create projects according to the procedures in 5.3.2 Create a New Project.
2. (Flash boot) Insert the loop part in startup_core.c of the primary project with reference to Appendix. How to Debug FSP Project with Flash Boot Mode. For RZ/N2H, the board file override is also required.
3. Build the primary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 1.
4. Create a secondary project using the bundle file (.sbd) of the primary project according to the procedures in 5.3.2 Create a New Project.
5. (RAM exec) Build the secondary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 2.
6. (Flash boot) Build the secondary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 2. The following additional properties must be set.
   i. Click **Project** > **Options…**.
   ii. Click **Output Converter** > **Output** and set **Raw binary** to **Output format**.

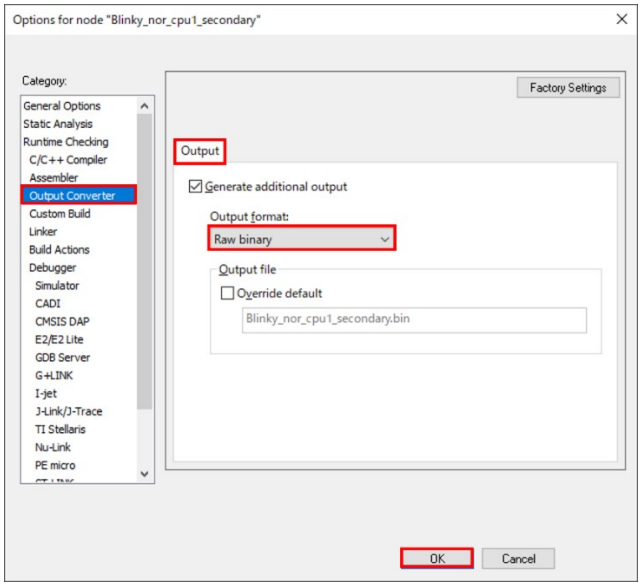


Figure 140 IAR EWARM Project Options for the Secondary Project in Flash Boot Mode

7. (Flash boot) Set the following additional options to the primary project.
   i. Click **Project** > **Options…**.
   ii. Click **Linker** > **Input** and set it as follows:
   - Keep symbols: (one per line)
     - SECONDARY
   - Raw binary image
     - File: the path of binary file in the secondary project.
       e.g. $PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary.bin
     - Symbol: SECONDARY
     - Section: SECONDARY
     - Align: 8

(Continued on next page)

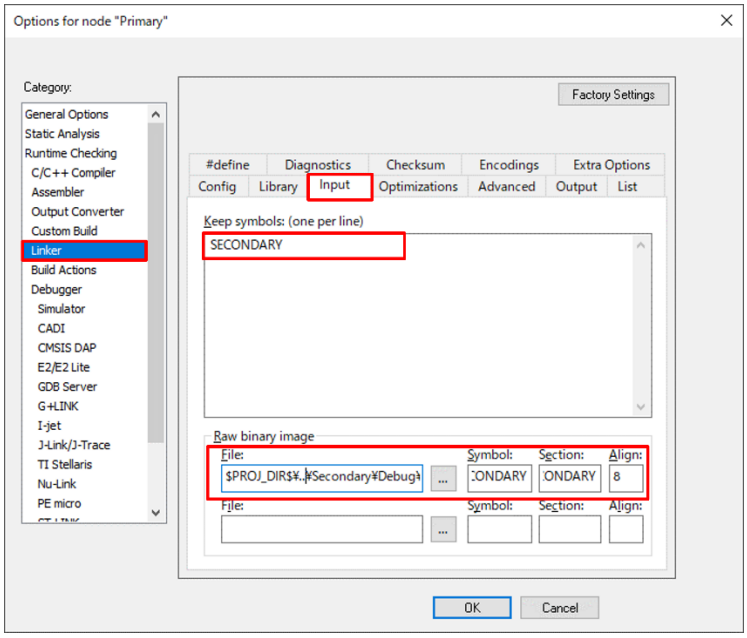


Figure 141 IAR EWARM Project Options for the Primary Project in Flash Boot Mode

8. Build the primary project according to the procedures in 5.3.3.2 Build for Multiprocessing No. 3.
9. (Flash boot) In the primary project, click **Project** > **Download** > **Download file…** and select out file of the primary project.
   e.g. $PROJ_DIR$\..\Blinky_primary\Debug\Exe\Blinky_primary.out

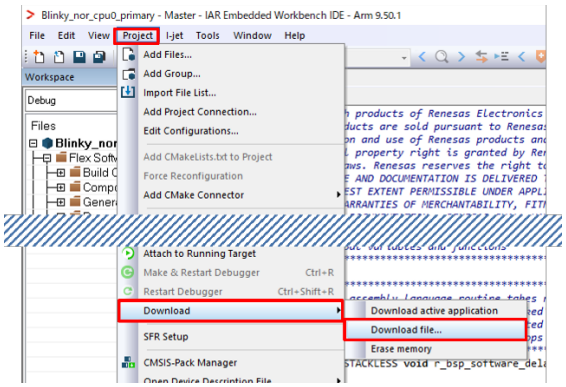


Figure 142 IAR EWARM Download File

10. (RAM exec) Debug the projects according to the procedures in 5.3.5 Debug for Multiprocessing.
11. (Flash boot) Enable settings for multicore debugging in 5.3.5 Debug for Multiprocessing No. 2.

(Continued on next page)

12. (Flash boot) Click **Project** > **Debug without Downloading** of the primary project to debug. Debug the projects according to the procedures in 5.3.5 Debug for Multiprocessing No. 4 and after.

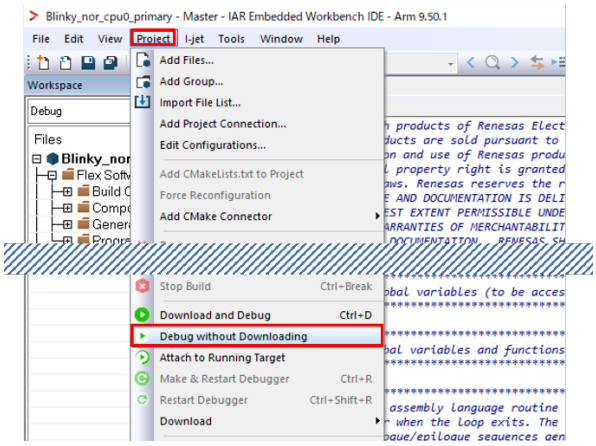


Figure 143 IAR EWARM Debug without Downloading

13. When changing the project and debugging it again, follow these steps.
    i. (Flash boot) Disable asymmetric multicore setting.
        a. Click **Project** > **Options…**.
        b. Click **Debugger** > **Multicore** and select **Disable** in Asymmetric multicore.
    ii. Build the primary project (No. 3).
    iii. (RAM exec) Build the secondary project (No. 5).
    iv. (Flash boot) Build the secondary project (No. 6).
    v. Build the primary project again (No. 8).
    vi. (Flash boot) Download the file (No. 9).
    vii. (RAM exec) Debug the projects (No. 10).
    viii. (Flash boot) Enable asymmetric multicore setting (No. 11).
    ix. (Flash boot) Debug the projects (No. 12).

Multiprocessing with 3 or more cores for RZ/N devices

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

1. Create and build the primary and secondary projects according to Multiprocessing with 2 cores for RZ/N devices No. 1 to No. 6.
2. Create a tertiary project using the bundle file (.sbd) of the secondary project according to the procedures in 5.3.2 Create a New Project.
3. Build the tertiary project according to Multiprocessing with 2 cores for RZ/N devices No. 5 and No. 6. The project option settings for the tertiary project are the same as for the secondary project.

(Continued on next page)

4. (Flash boot) Set the following additional options to the secondary project.
   i. Click **Project** > **Options…**.
   ii. Click **Linker** > **Input** and set it as follows:
   - Keep symbols: (one per line)
     - SECONDARY
   - Raw binary image
     - File: the path of binary file in the tertiary project.
       e.g. $PROJ_DIR$\..\Blinky_tertiary\Debug\Exe\Blinky_tertiary.bin
     - Symbol: SECONDARY
     - Section: SECONDARY
     - Align: 8
5. Build the secondary project according to the procedures in 5.3.3.1 Build.
6. (Flash boot) Set the following additional options to the primary project according to Multiprocessing with 2 cores for RZ/N devices No. 7.
   i. Click **Project** > **Options…**.
   ii. Click **Linker** > **Input** and set it as follows:
   - Keep symbols: (one per line)
     - SECONDARY
   - Raw binary image
     - File: the path of binary file in the secondary project.
       e.g. $PROJ_DIR$\..\Blinky_secondary\Debug\Exe\Blinky_secondary.bin
     - Symbol: SECONDARY
     - Section: SECONDARY
     - Align: 8
7. Create multicore_setup.xml and store it in the primary project.

   multicore_setup.xml

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

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

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

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

- <name> Arbitrary name.
- <workspace> Location of workspace starting from the primary project location. The primary project only set "WS_PATH".
- <project> Project name. The primary project only set "$PROJ_PATH$".
- <config> Debug
- <numberOfCores> 1
- (Except the primary project) <attachToRunningTarget> false

8. Build the primary project according to the procedures in 5.3.3.1 Build.
9. Debug the projects
   i. Open the primary project and close the secondary and later projects on IAR EWARM.
   ii. Set the following in the primary project before debugging:
      a. Click **Project** > **Options…**.
      b. Click **Debugger** > **Multicore** and check the setting value of **Symmetric multicore** and set the following contents in **Asymmetric multicore**.
         - Symmetric multicore
           - Number of cores: 1
         - Asymmetric multicore
           - Advanced
             - Session configuration: $PROJ_DIR$\multicore_setup.xml
   iii. Follow steps according to 5.3.5 Debug for Multiprocessing No. 3 to No. 7. As shown in Figure 139, it is according to the setting in multicore_setup.xml. The primary project is 0, the secondary project is 1, and the tertiary project is 2.

Figure 144 IAR EWARM Running Projects

10. When changing the project and debugging it again, follow these steps.
    i. (Flash boot) Disable asymmetric multicore setting.
       a. Click **Project** > **Options…**.
       b. Click **Debugger** > **Multicore** and select **Disable** in Asymmetric multicore.
    ii. Build the primary and secondary projects (Multiprocessing with 2 cores for RZ/N devices No. 3)
    iii. Build the secondary project (Multiprocessing with 2 cores for RZ/N devices No. 5 and No. 6).
    iv. Build the tertiary project (Multiprocessing with 3 or more cores for RZ/N devices No. 3).
    v. Build the secondary project again (Multiprocessing with 2 cores for RZ/N devices No. 5 and No. 6).
    vi. Build the primary project again (Multiprocessing with 2 cores for RZ/N devices No. 3).
    vii. (Flash boot) Debug the projects (Multiprocessing with 3 or more cores for RZ/N devices No. 9).

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.25	Updated “e^2 studio Prerequisites” • Updated the Windows PC requirements.
		p.47	Update “Prerequisites” section • Added the note regarding the patch for debugging RZ/N2L FSP project on EWARM.
		p.48	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.74	Updated “Selecting a Board and Toolchain” section. • Added the detailed explanation how to select Board and Device for creating a FSP project.
		p.88	Updated “Appendix. Known Issues” chapter.
		p.98	Updated “Appendix. Tool Software Limitations” section.
		-	Added “Appendix. How to update J-Link DLL files in e^2 studio” chapter.
		p.126	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.88	Updated “Appendix. Known Issues” chapter. • Remove some limitations regarding RZ/T2M

Rev.	Date	Description	
		Page	**Summary**
		p.98	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 v2.0.0. • Removed contents for RZ/T2 FSP v1.3.0
		p.1	List Target Device separately for each series.
		p.6	Updated “1.3 Related Documentation Files” section. • List Target Device separately for each series.
		p.9	Updated “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.31	Updated “4.6 Run the Blinky Project” section. • Added LED2-3 of RSK+RZ/T2M for CPU1 core. • Added descriptions to suspend program execution and exit debug mode.
		p.32	Added “4.7 Debug and Run for Multiprocessing” section.
		p.33	Added “4.8 Import the Project” section.
		p.37	Updated “5.2 Tutorial Blinky” section. • Added description of a multiprocessing.
		p.38	Updated “5.3.2 Create a New Project” section. • Added description of a multiprocessing.
		p.43	Added “5.3.2.1 NOTE: Configure IAR EWARM Project [Only RZ/N2L]” section.
		p.43	Updated “5.3.3 Build the Project” section. • Moved text to “5.3.3.2 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.72	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.89	Updated “Appendix. Tool Software Limitations” chapter. • Added new limitation No. 9 and No. 10.
		P.105	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 “5.3.3.2 Build for 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 issue No. 12 • Added RZ/N2L as target device from No.13 and No.14 • Added new issues No. 16 and No. 17.
		p.84	Updated “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.
1.11	Jun.28.24	All	Updated documentation for RZ/T2 FSP v2.1.0. • Removed contents for RZ/T2 FSP v2.0.0
		All	Added the RZ/T2ME device as target device.
		p.1	Added video links of FSP Configuration.
		p.6	Updated “1.3.2 FSP Documentation” section. • Added explanation of notes when using FSP software modules.
		p.10	Updated “2.4.1.1 Boot Mode” section. • Added note for board setting.
		p.13	Updated “2.4.2.1 Boot Mode” section. • Added note for board setting.
		p.15	Added “2.4.3 RSK+RZT2ME” section.
		p.16	Updated “2.5.1.1 Boot Mode” section. • Added note for board setting.
		p.38	Updated “5.3.2 Create a New Project” section. • Added IDE Project Type setting for newer versions of FSP SC.
		p.43	Updated “5.3.3.2 Build for Multiprocessing” section. • Removed Build Actions setting. • Added Make before debugging setting.
		p.54	Updated “6 FSP Configuration Users Guide” section. • Updated figures with new tool screens.
		P.74	Updated “Appendix. Known Issues” chapter. • Resolved issues No. 16 and No. 17 • Added new issues No. 18 to No. 24.
		p.85	Updated “Appendix. Tool Software Limitations” chapter. • Removed RZ/T2M and RZ/T2L as target device from No. 6. • Added new issue No. 11.

Rev.	Date	Description	
		Page	**Summary**
		p.102	Updated “Appendix. How to Change Boot Mode of FSP Project” section. • Added note on FSP version changes
1.12	Nov.26.24	All	Updated documentation for RZ/T2 FSP v2.2.0. • Removed contents for RZ/T2 FSP v2.1.0
		All	Added the RZ/T2H as target device and CA55 core support.
		p.6	Updated “1.3.2 FSP Documentation” section. • Removed Note for RZ/N2L about the documentation issue.
		p.8	Updated “2 Set up Evaluation Board” chapter. • Changed the boards designations.
		p.9	Updated “2.3.1 SEGGER J-Link” section. • Added how to update J-Link firmware.
		p.16	Added “2.4.4 RZ/T2H Evaluation Board” section.
		p.24	Updated “4.3 Create a New Project for Blinky” section. • Modified to a generic description that does not specify which cores are used in multiprocessing. • Updated tool screen images. • Added tables describing the settings for each project.
		p.31	Added “4.4.2 Build for Multiprocessing” section.
		p.32	Updated “4.5.2 Debug Steps” section. • Added tables describing the settings for each project. • Added a note for debugging flash boot project.
		p.37	Updated “4.7 Debug and Run for Multiprocessing” section. • Clarified explanation of step 2. • Added supplemental information on behavior to step 5.
		p.44	Updated “5.3.2 Create a New Project” section. • Modified to a generic description that does not specify which cores are used in multiprocessing. • Updated tool screen images. • Added tables describing the settings for each project.
		p.51	Added “5.3.2.1 NOTE: Configure IAR EWARM Project [RZ/T2H]” section.
		p.54	Updated “5.3.3 Build the Project” section. • Swapped the order of “5.3.3.1 NOTE: Build settings [Only Multiprocessing]” and “5.3.3.2 Build” chapters. • Added how to build for multiprocessing in “5.3.3.1 Build” section. • Renamed “5.3.3.2 Build for Multiprocessing” section.
		p.55	Updated “5.3.3.2 Build for Multiprocessing” section. • Added the running setting to step 1. • Added the extra options to step 2. • Removed the running setting and tools options from step 3. • Moved multicore debugging setting from step 3 to “5.3.5 Debug for Multiprocessing” section.

Rev.	Date	Description	
		Page	**Summary**
		p.58	Updated “5.3.4 Download & Debug the Project” section. • Added a note for debugging flash boot project.
		p.62	Updated “5.3.5 Debug for Multiprocessing” section. • Added multicore debugging setting as step 2. • Added supplemental information on behavior to step 5. • Added how to debug when changing the project.
		p.67	Updated “6.2 Create a Project” section. • Updated tool screen images and menu names.
		p.71	Added “6.2.4 Duplication of Resources” section.
		p.82	Updated “Appendix. Known Issues” section. • Added RZ/T2H as target device of No. 3, No. 14, No. 27 and No. 28. • Added new issues No. 29 to No. 38. • Resolved issues No. 4, No. 13, No. 18, No. 19 and No. 20.
		p.106	Updated “Appendix. Tool Software Limitations” chapter. • Added RZ/T2H as target device of No. 4, No. 7, No. 9 and No. 10. • Added new issues No. 12 to No. 19. • Resolved issues No. 1 and No. 11.
		p.117	Updated “Appendix. How to Debug FSP Project with Flash Boot Mode” section. • Add Note for multiprocessing projects. • Corrected boot mode name from xSPI0 to xSPI. • Updated path description. • Add a column for cores to the table.
		-	Removed “Appendix. How to Debug FSP multiprocessing projects with Flash Boot Mode” section.
		p.127	Added “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e^2 studio” section.
		p.132	Added “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM” section.
1.13	Dec.23.24	All	Updated documentation for RZ/N2 FSP v2.1.0. • Removed contents for RZ/N2 FSP v2.0.0.
		All	Added the RZ/N2H as target device.
		p.22	Added “2.5.2 RZ/N2H Evaluation Board” section.
		p.57	Renamed “NOTE: Configure IAR EWARM Project [RZ/T2H and RZ/N2H]” section.
		p.86	Updated “Appendix. Known Issues” section. • Resolved issues No. 21, No. 22, No. 23, No. 24. • Added RZ/N2H as target device of No. 3, No. 14, No. 27 to No. 36. • Added RZ/N2L as target device of No. 27, No. 28. • Added new issues No. 38 to No. 43. • Replaced Smart Configuration with SC.

Rev.	Date	Description	
		Page	**Summary**
		p.113	Updated “Appendix. Tool Software Limitations” chapter. • Added RZ/N2H as target device of No. 4, No. 7, No. 9, No. 10 and No. 12 to No. 19. • Added new issues No. 20, No. 21. • Resolved issues No. 6. • Updated the description of issues No. 12. • Replaced Smart Configuration with SC.
		p.125	Updated “Appendix. How to Debug FSP Project with Flash Boot Mode” section. • Moved Note for multiprocessing projects to No. 1 in the same chapter. • Removed CA55 NOR flash boot from the table in No. 1. • Added No. 2 for RZ/N2H. • Added note for debug FSP project.
		p.127	Updated “Appendix. How to Erase Flash Memory” • Added Device Type Name for RZ/N2H. • Added External Address Space for RZ/N2H.
1.14	Feb.28.25	All	Updated documentation for RZ/T2 FSP v2.3.0. • Removed contents for RZ/T2 FSP v2.2.0 • Unified wording for Smart Configurator(SC) and IAR I-jet.
		p.25	Updated “3.2.1 Windows PC Requirements” section. • Updated Windows PC requirements to use e^2 studio.
		p.25	Updated “3.2.3 Choosing a Toolchain” section. • Added a table of toolchain version for each FSP.
		p.26	Updated “4.3 Create a New Project for Blinky” section. • Added a table of the project creation procedure. • Added a table of selecting a bundle file to procedure No. 9.
		p.47	Updated “5.3.2 Create a New Project” section. • Added a table of the project creation procedure. • Updated a description of selecting a bundle file in procedure No. 9. • Added a table of selecting a bundle file to procedure No. 9.
		p.55	Updated “5.3.2.1 NOTE: Configure IAR EWARM Project [RZ/T2H and RZ/N2H]” section. • Updated a description of error conditions. • Removed step 5 to set Build Actions.
		p.86	Updated “Appendix. Known Issues” chapter. • Resolved issues No. 3, No. 25 to 27, No. 29, No. 31, No. 36 to 43 for RZ/T series devices. • Added new issue No. 44.
		p.118	Updated “Appendix. Tool Software Limitations” chapter. • Resolved limitation No. 17 for RZ/T series devices. • Added new limitation No. 22.

Rev.	Date	Description	
		Page	**Summary**
		p.141	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e^2 studio” chapter. • Added “Multiprocessing with 3 or more cores” section.
		p.146	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM” chapter. Added “ • Multiprocessing with 3 or more cores” section.
1.15	Apr.25.25	All	Updated documentation for RZ/N2 FSP v2.2.0. • Removed contents for RZ/N2 FSP v2.1.0
		p.86	Updated “Appendix. Known Issues” chapter. • Resolved issues No. 3, No. 27, No. 29, No. 31, No. 37 to 43. • Added new issue No. 45, No. 46, No. 47, No. 48.
		p.118	Updated “Appendix. Tool Software Limitations” chapter. • Removed RZ/N2H as target device from No. 13, No. 14. • Resolved limitation No. 17. • Added new limitation No. 23.
		p.131	Updated “Appendix. How to Debug FSP Project with Flash Boot Mode” section. • Updated a description of CA55 xSPI boot from the table in No. 1
		p.147	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e^2 studio” chapter. • Updated “Multiprocessing with 3 or more cores” section for flash boot.
1.16	Jul.9.25	All	Updated documentation for RZ/T2 FSP v3.0.0. • Removed contents for RZ/T2 FSP v2.3.0
		p.25	Updated “3.2.3 Choosing a Toolchain” section. • Added version name displayed in arm Developer website.
		p.35	Updated “4.5.2 Debug Steps” section. • Added script file setting for TCM initialization.
		p.60	Updated “5.3.3.2 Build for Multiprocessing” section. • Added macro file setting for TCM initialization.
		p.88	Updated “Appendix. Known Issues” chapter. • Resolved issues No. 14, No. 32, No. 46, No. 47 for RZ/T series devices. • Resolved issues No.45, No. 48.
		p.122	Updated “Appendix. Tool Software Limitations” chapter. • Resolved limitations No. 13, No. 14.
		p.135	Updated “Appendix. How to Debug FSP Project with Flash Boot Mode” chapter. • Removed column e^2 studio 2022-04 2024-07. • Updated black text code in system_init(). • Changed the number of loops for waiting to 1.5x.
		p.145	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for e^2 studio” chapter. • Added procedure for RZ/T2 FSP v3.0.0 to meet that specification.

Rev.	Date	Description	
		Page	Summary
		p.158	Updated “Appendix. How to Create and Debug FSP Projects for Multiprocessing in All Cases for IAR EWARM” chapter. • Added procedure for RZ/T2 FSP v3.0.0 to meet that specification.

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/.