RZ/A3UL, RZ/A3M

Getting Started with RZ Flexible Software Package

Introduction

This manual describes how to use the RZ Flexible Software Package (FSP) for developing applications for the RZ microprocessor series.

Target Device

RZ/A3UL

RZ/A3M

Contents

1. Introduction

1.1 Overview

This application note describes how to use the Renesas RZ Flexible Software Package (FSP) running on the Cortex®-A55 (hereinafter referred to as CA55) incorporated on RZ/A3UL and RZ/A3M.

1.2 Introduction to FSP

1.2.1 Purpose

The Renesas RZ 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 drivers that meet common use cases in embedded systems.

1.2.2 e^2 studio IDE

FSP provides a host of efficiency-enhancing tools for developing projects targeting the Renesas RZ series of MPU devices. The e^2 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.

1.3 Limitations

1.3.1 Hardware Initial Setup

RZ FSP expects the initial setup of hardware to be carried out beforehand by RZ Initial Program Loader (hereinafter referred to as IPL). For details on IPL, please refer to the Document of IPL.

2. Starting Development Introduction

2.1 e^2 studio setup

2.1.1 What is e^2 studio?

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

When developing for RZ MPUs, e^2 studio hosts the RZ FSP. The FSP provides a wide range of time-saving tools to simplify the selection, configuration, and management of modules and threads, to easily implement complex applications.

2.1.2 e^2 studio Prerequisites

2.1.2.1 Obtaining an RZ MPU Kit

To develop applications with RZ FSP, start with the RZ/A3UL Evaluation Board Kit and the EK-RZ/A3M.

RZ/A3UL Evaluation Board Kit-related information is available at RZ/A3UL Evaluation Board Kit.

EK-RZ/A3M related information is available at the EK-RZ/A3M Evaluation Kit for RZ/A3M MPU.

The relationship between the board type and the board name on e^2 studio is as follows.

	Board name on the GUI screen	**Note**
RZ/A3UL Evaluation Board Kit QSPI Edition (RTK9763U02S01000BE)	RZ/A3UL Evaluation Board Kit QSPI Edition (Exec with DDR SDRAM)	If you select this, the initial program loader will transfer the entire program, including the code area, to DDR4.
	RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place)	If you select this, only the data area will be transferred to DDR4 by the initial program loader. The code area on the flash ROM is referenced during execution.
RZ/A3UL Evaluation Board Kit Octal-SPI Edition (RTK9763U02S01001BE)	RZ/A3UL Evaluation board Kit OCTAL Edition (eXecute-In-Place)	If you select this, only the data area will be transferred to OctaRAM by the initial program loader. The code area on the flash ROM is referenced during execution.
EK-RZ/A3M (RTK9EKZA3MS10001BE)	EK-RZ/A3M NOR Boot (Exec with DDR SDRAM)	If you select this, the initial program loader will transfer the entire program, including the code area, to Built-in DDR3.
EK-RZ/A3M (RTK9EKZA3MS10001BE)	EK-RZ/A3M NAND Boot (Exec with DDR SDRAM)	If you select this, the initial program loader will transfer the entire program, including the code area, to Built-in DDR3.

2.1.2.2 PC Requirements

The following are the minimum PC requirements to use e^2 studio:

- Windows 10 or Ubuntu 22.04 LTS Desktop(64-bit) with Intel i5 or i7, or AMD A10-7850K or FX
- Memory: 8-GB DDR3 or DDR4 DRAM (16-GB DDR4/2400-MHz RAM is preferred)
- Minimum 250 GB hard disk

2.1.2.3 Licensing

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

2.1.3 e^2 studio installation for Windows PC

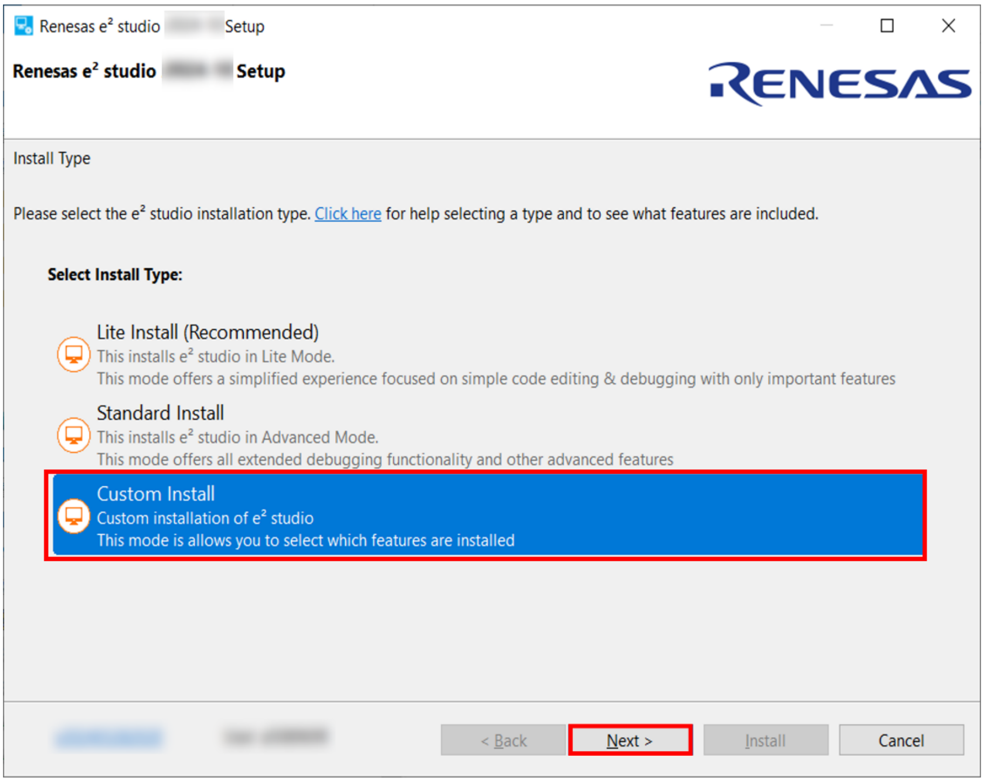
This chapter describes how to install the e^2 studio IDE on a Windows PC. If you would like to install e^2 studio and FSP at the same time, please jump to 2.2.1.

2.1.3.1 Download

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

2.1.3.2 Installation of e^2 studio IDE

1. Double-click the e^2 studio installer to launch the e^2 studio installation wizard. Then, select the [Custom Install] option and click the [Next] button.



Renesas e² studio [redacted] Setup

Renesas e² studio [redacted] Setup

Install Type

Please select the e² studio installation type. Click here for help selecting a type and to see what features are included.

Select Install Type:

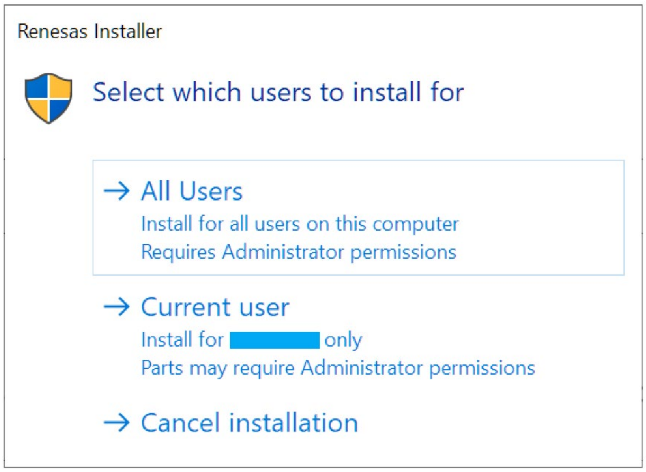
Lite Install (Recommended)
This installs e² studio in Lite Mode.
This mode offers a simplified experience focused on simple code editing & debugging with only important features

Standard Install
This installs e² studio in Advanced Mode.
This mode offers all extended debugging functionality and other advanced features

Custom Install
Custom installation of e² studio
This mode is allows you to select which features are installed

Figure 1. e^2 studio installation wizard

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



Renesas Installer

Select which users to install for

→ All Users
Install for all users on this computer
Requires Administrator permissions

→ Current user
Install for [redacted] only
Parts may require Administrator permissions

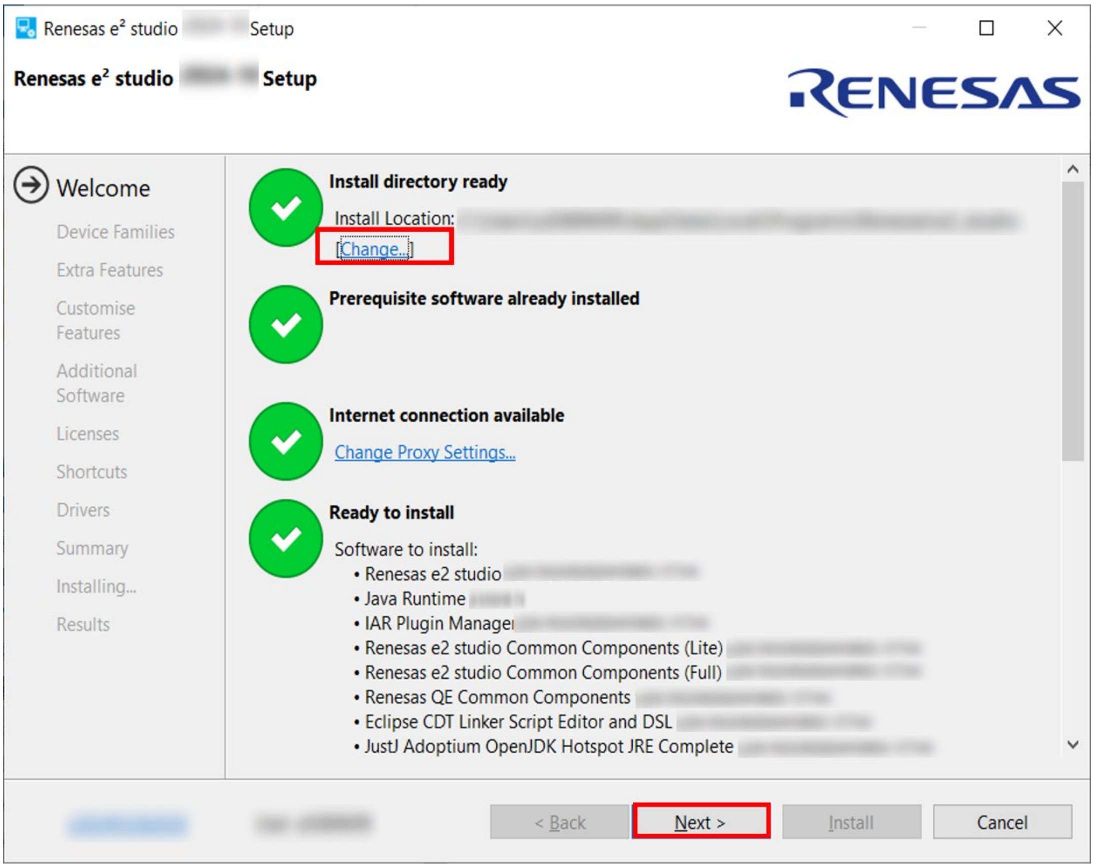
→ Cancel installation

Figure 2. Select User for Installation

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

Note:

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



Install directory ready

Install Location:

Prerequisite software already installed

Internet connection available

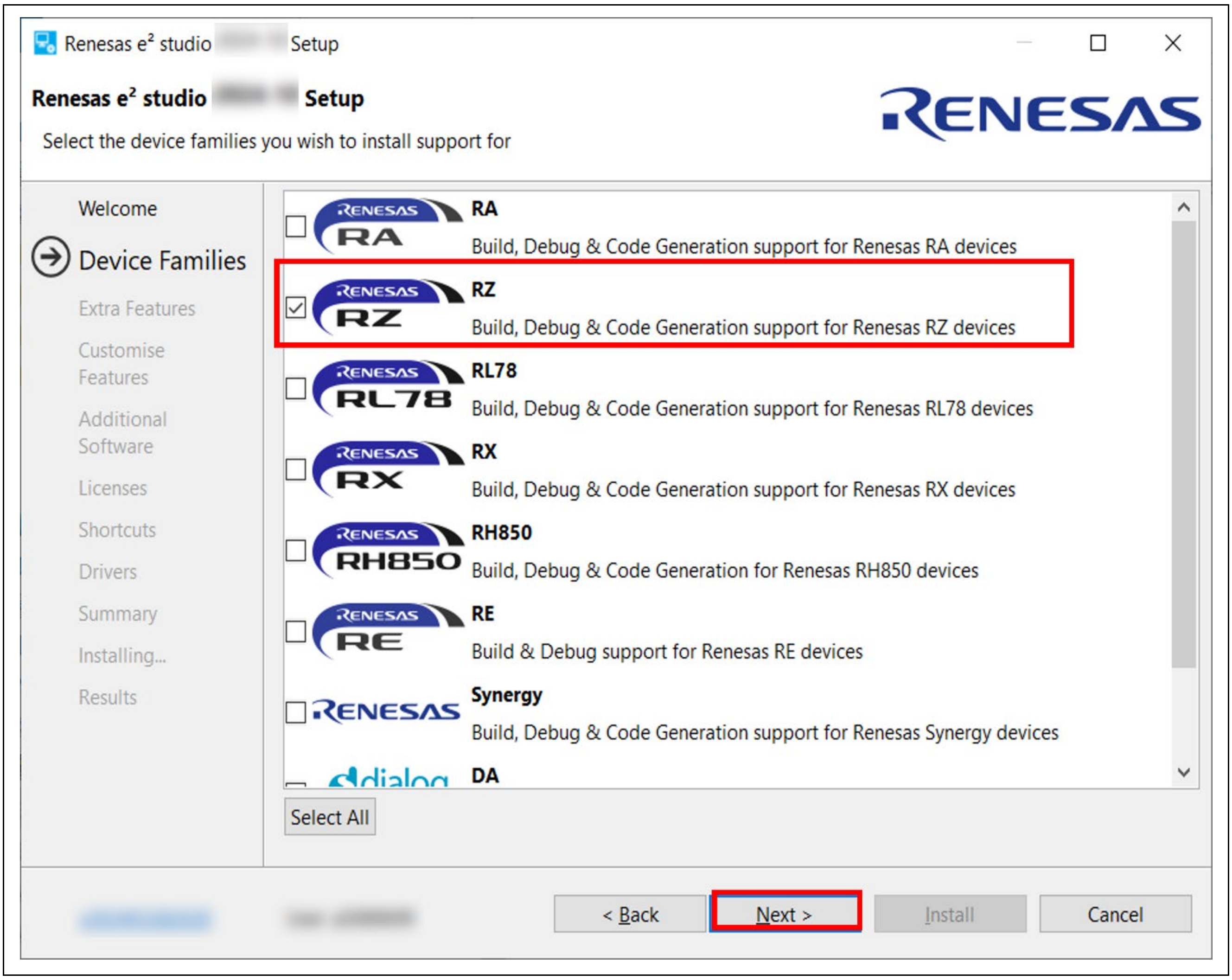
Ready to install

Software to install:

- Renesas e2 studio
- Java Runtime
- IAR Plugin Manager
- Renesas e2 studio Common Components (Lite)
- Renesas e2 studio Common Components (Full)
- Renesas QE Common Components
- Eclipse CDT Linker Script Editor and DSL
- JustJ Adoptium OpenJDK Hotspot JRE Complete

Figure 3. Installation of e^2 studio – Welcome page

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



Renesas e² studio Setup

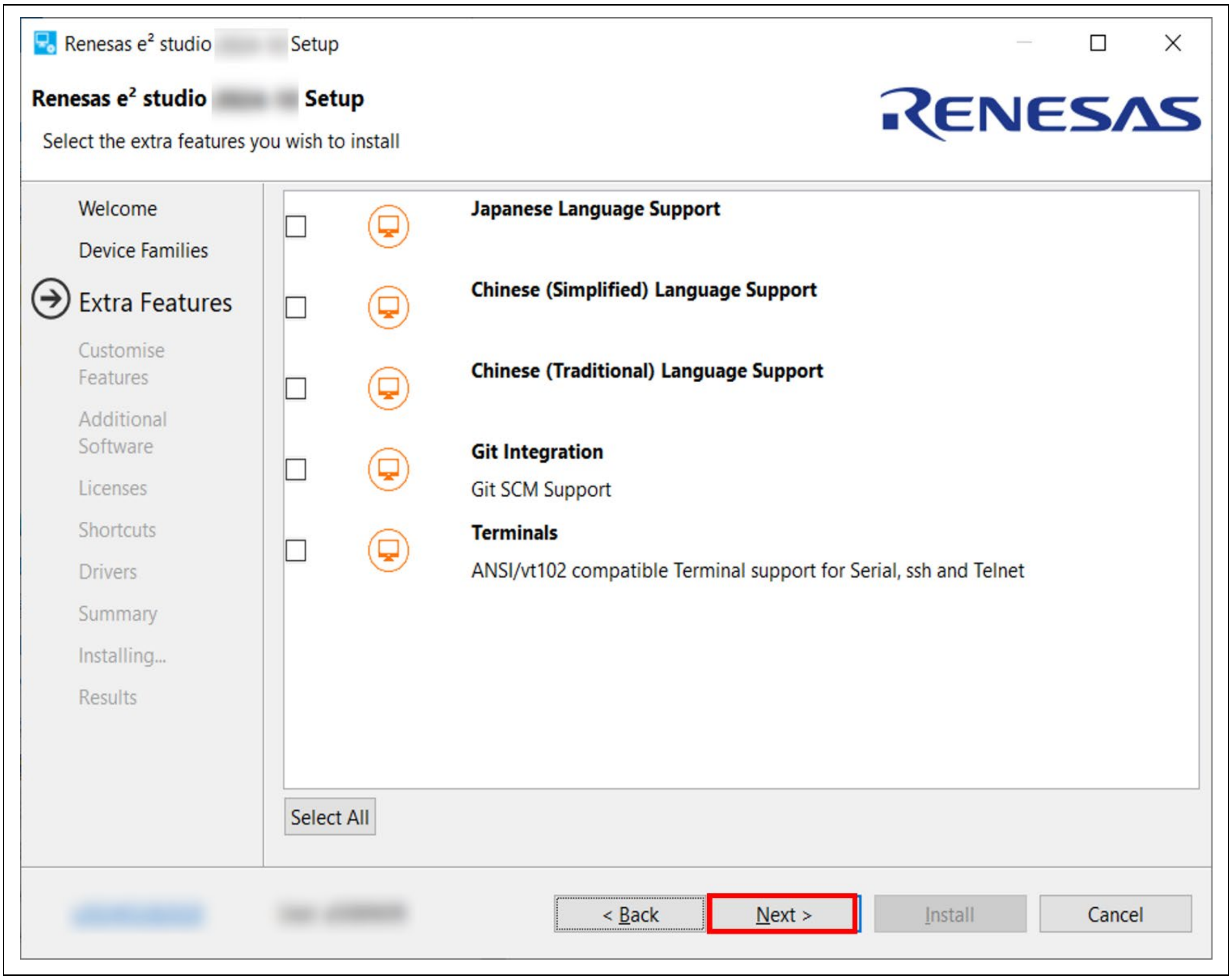
Renesas e² studio Setup

Select the device families you wish to install support for

- ☐ **RA** Build, Debug & Code Generation support for Renesas RA devices
- ☑ **RZ** Build, Debug & Code Generation support for Renesas RZ devices
- ☐ **RL78** Build, Debug & Code Generation support for Renesas RL78 devices
- ☐ **RX** Build, Debug & Code Generation support for Renesas RX devices
- ☐ **RH850** Build, Debug & Code Generation for Renesas RH850 devices
- ☐ **RE** Build & Debug support for Renesas RE devices
- ☐ **Synergy** Build, Debug & Code Generation support for Renesas Synergy devices
- **DA**

Figure 4. Installation of e² studio – Device Families

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



Renesas e² studio Setup

Renesas e² studio Setup

Select the extra features you wish to install

☐ **Japanese Language Support**

☐ **Chinese (Simplified) Language Support**

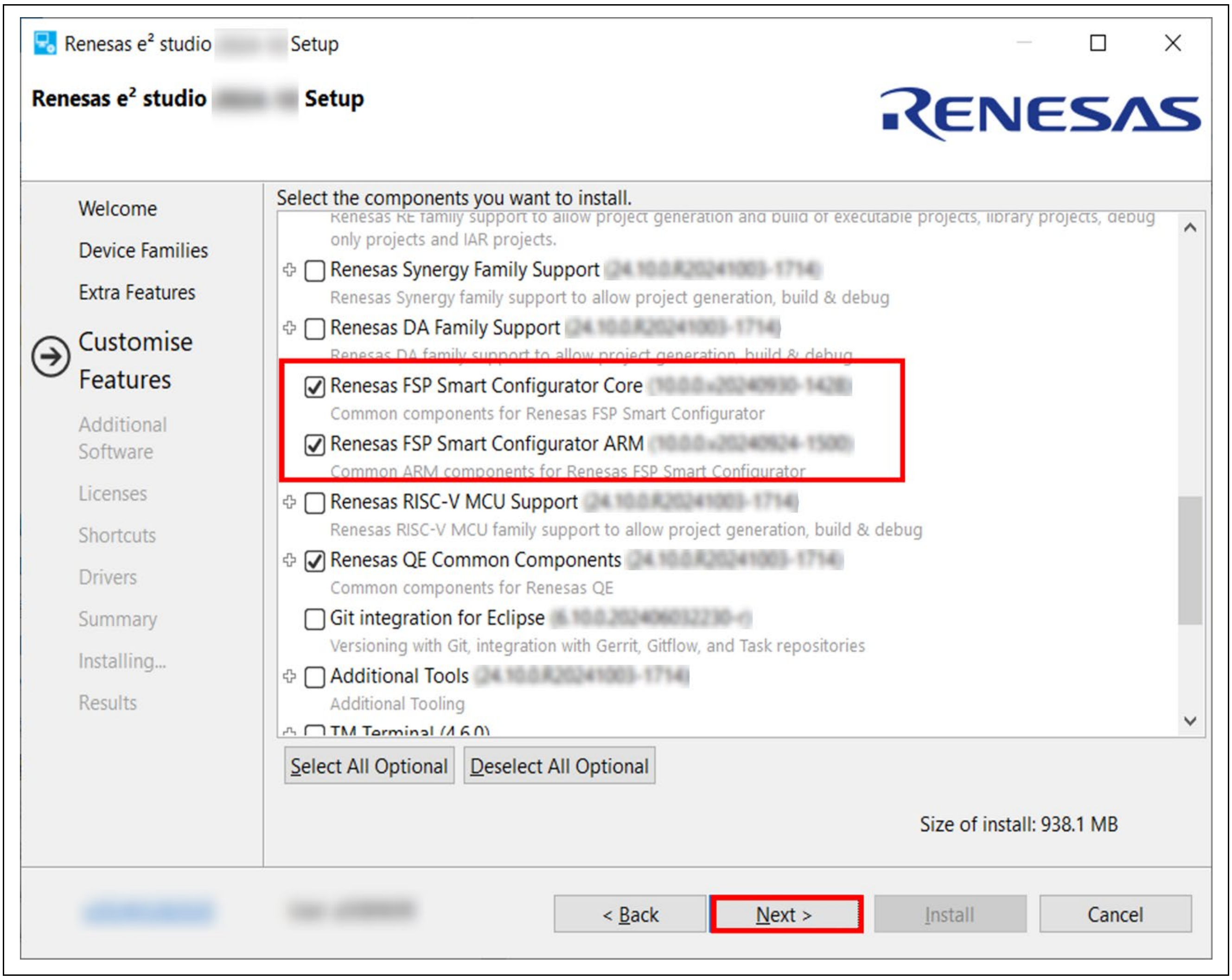
☐ **Chinese (Traditional) Language Support**

☐ **Git Integration**
Git SCM Support

☐ **Terminals**
ANSI/vt102 compatible Terminal support for Serial, ssh and Telnet

Figure 5. Installation of e² studio – Extra Features

5. Customize Features
   Select the components to install and click the [Next] button to continue. Be sure to choose “Renesas FSP Smart Configurator Core”. Otherwise, FSP won’t be built on e^2 studio successfully.



Select the components you want to install.

Renesas RE family support to allow project generation and build of executable projects, library projects, debug only projects and IAR projects.

☐ Renesas Synergy Family Support
Renesas Synergy family support to allow project generation, build & debug

☐ Renesas DA Family Support
Renesas DA family support to allow project generation, build & debug

☑ Renesas FSP Smart Configurator Core
Common components for Renesas FSP Smart Configurator

☑ Renesas FSP Smart Configurator ARM
Common ARM components for Renesas FSP Smart Configurator

☐ Renesas RISC-V MCU Support
Renesas RISC-V MCU family support to allow project generation, build & debug

☑ Renesas QE Common Components
Common components for Renesas QE

☐ Git integration for Eclipse
Versioning with Git, integration with Gerrit, Gitflow, and Task repositories

☐ Additional Tools
Additional Tooling

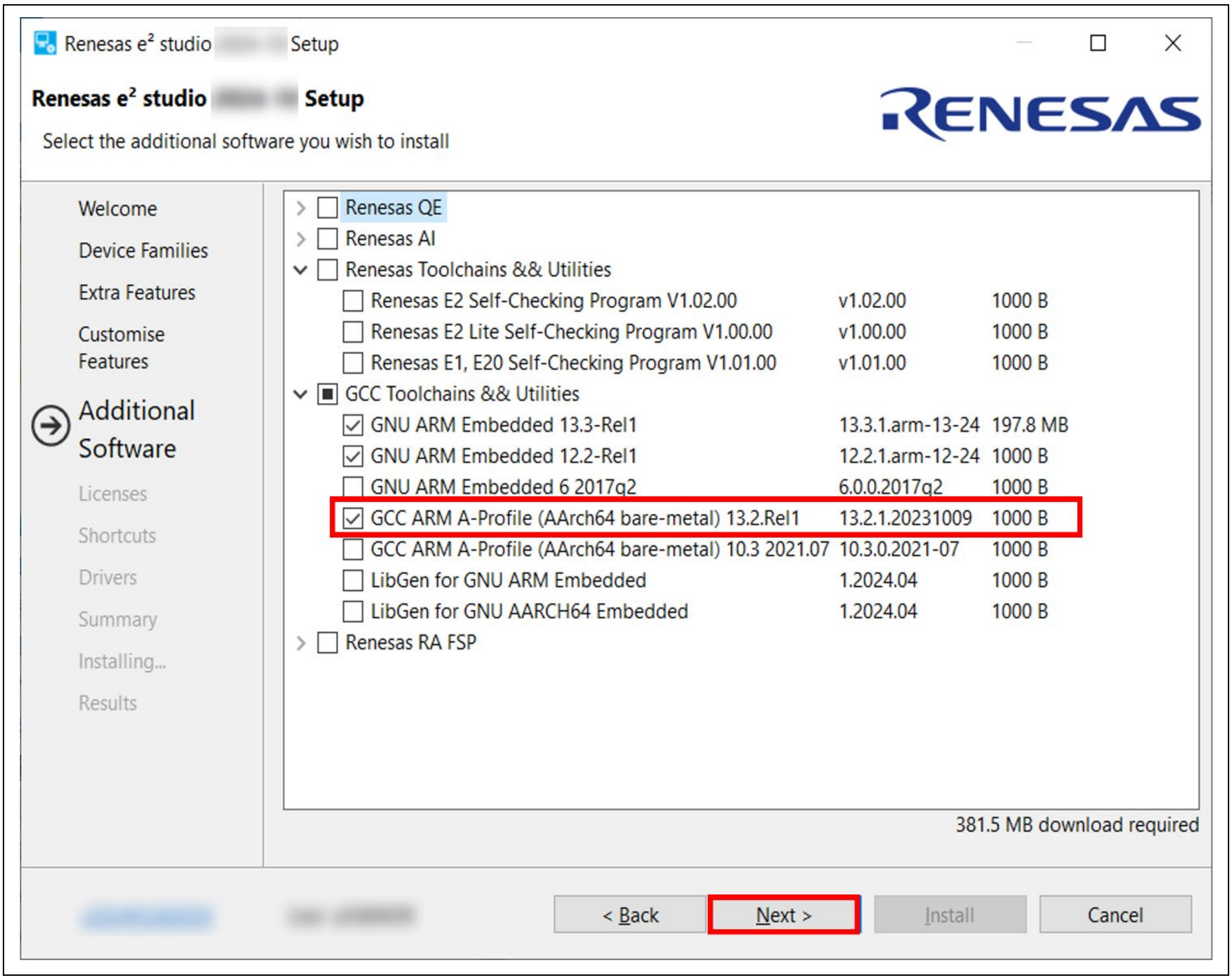
☐ TM Terminal (4.6.0)

Size of install: 938.1 MB

Figure 6. Installation of e^2 studio – Features

6. Additional Software
   Select additional software (i.e., compilers, utilities, QE…) to be installed. Be sure that you select the “GCC toolchains & Utilities” tab, choose the following items, and click [Next] to continue.

   - GCC ARM A-Profile (AArch64 bare-metal) 13.2 rel1



Renesas e² studio [redacted] Setup

Renesas e² studio [redacted] Setup

Select the additional software you wish to install

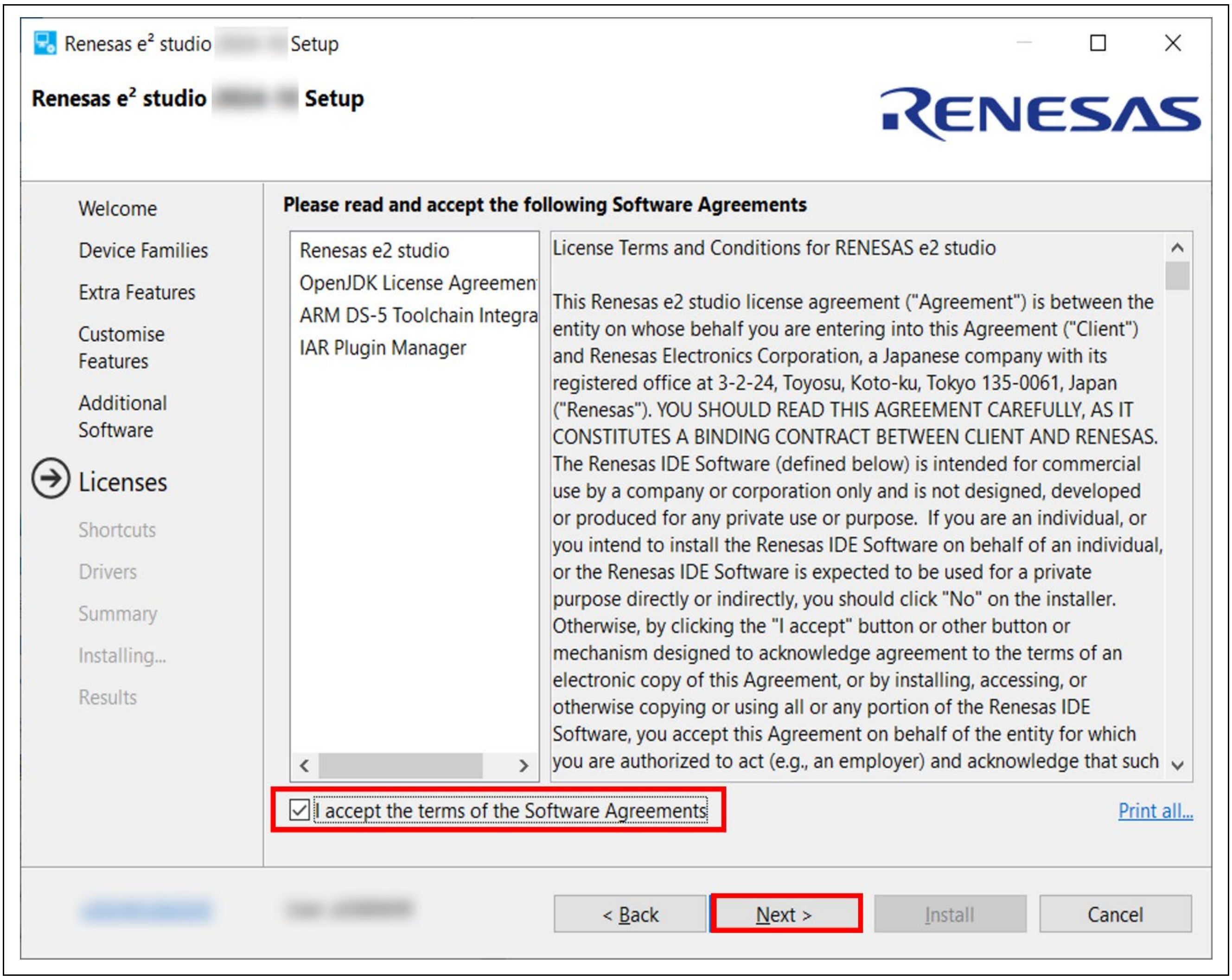
☐ Renesas QE		
☐ Renesas AI		
☐ Renesas Toolchains && Utilities		
☐ Renesas E2 Self-Checking Program V1.02.00	v1.02.00	1000 B
☐ Renesas E2 Lite Self-Checking Program V1.00.00	v1.00.00	1000 B
☐ Renesas E1, E20 Self-Checking Program V1.01.00	v1.01.00	1000 B
■ GCC Toolchains && Utilities		
☑ GNU ARM Embedded 13.3-Rel1	13.3.1.arm-13-24	197.8 MB
☑ GNU ARM Embedded 12.2-Rel1	12.2.1.arm-12-24	1000 B
☐ GNU ARM Embedded 6 2017q2	6.0.0.2017q2	1000 B
☑ GCC ARM A-Profile (AArch64 bare-metal) 13.2.Rel1	13.2.1.20231009	1000 B
☐ GCC ARM A-Profile (AArch64 bare-metal) 10.3 2021.07	10.3.0.2021-07	1000 B
☐ LibGen for GNU ARM Embedded	1.2024.04	1000 B
☐ LibGen for GNU AARCH64 Embedded	1.2024.04	1000 B
☐ Renesas RA FSP		

381.5 MB download required

[redacted] [redacted]

Figure 7. Installation of e² studio – Additional Software

7. Licenses Agreement
   Please read and accept the software license agreement, then click the [Next] button. Note that acceptance of the license agreement is mandatory; without it, the installation process cannot proceed.



Please read and accept the following Software Agreements

License Terms and Conditions for RENESAS e2 studio

This Renesas e2 studio license agreement ("Agreement") is between the entity on whose behalf you are entering into this Agreement ("Client") and Renesas Electronics Corporation, a Japanese company with its registered office at 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan ("Renesas"). YOU SHOULD READ THIS AGREEMENT CAREFULLY, AS IT CONSTITUTES A BINDING CONTRACT BETWEEN CLIENT AND RENESAS. The Renesas IDE Software (defined below) is intended for commercial use by a company or corporation only and is not designed, developed or produced for any private use or purpose. If you are an individual, or you intend to install the Renesas IDE Software on behalf of an individual, or the Renesas IDE Software is expected to be used for a private purpose directly or indirectly, you should click "No" on the installer. Otherwise, by clicking the "I accept" button or other button or mechanism designed to acknowledge agreement to the terms of an electronic copy of this Agreement, or by installing, accessing, or otherwise copying or using all or any portion of the Renesas IDE Software, you accept this Agreement on behalf of the entity for which you are authorized to act (e.g., an employer) and acknowledge that such

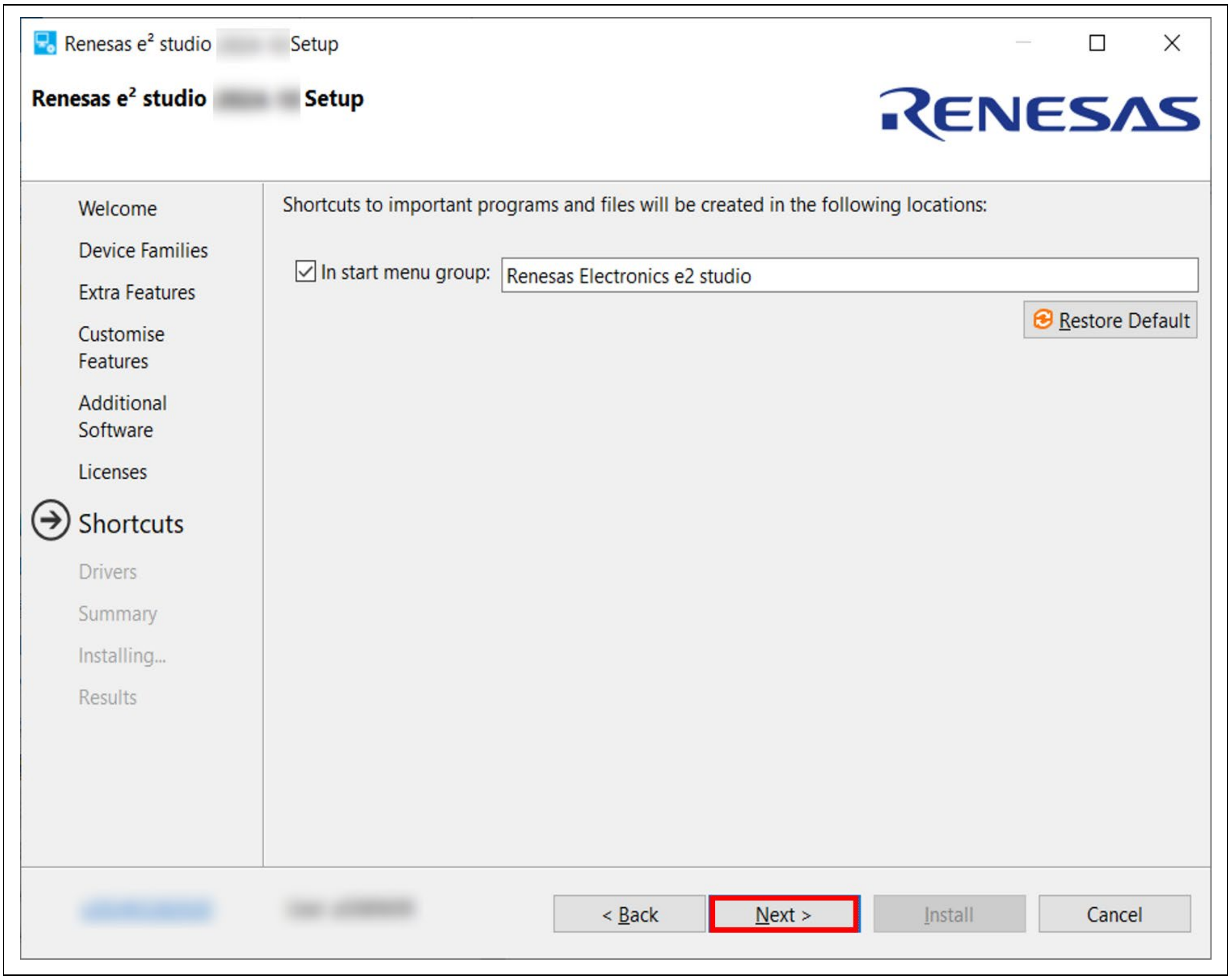
☑ I accept the terms of the Software Agreements

Figure 8. Installation of e² studio – Licenses

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

Note:

If e^2 studio was installed in another location, it is recommended to rename it to distinguish it from the other e^2 studio(s).

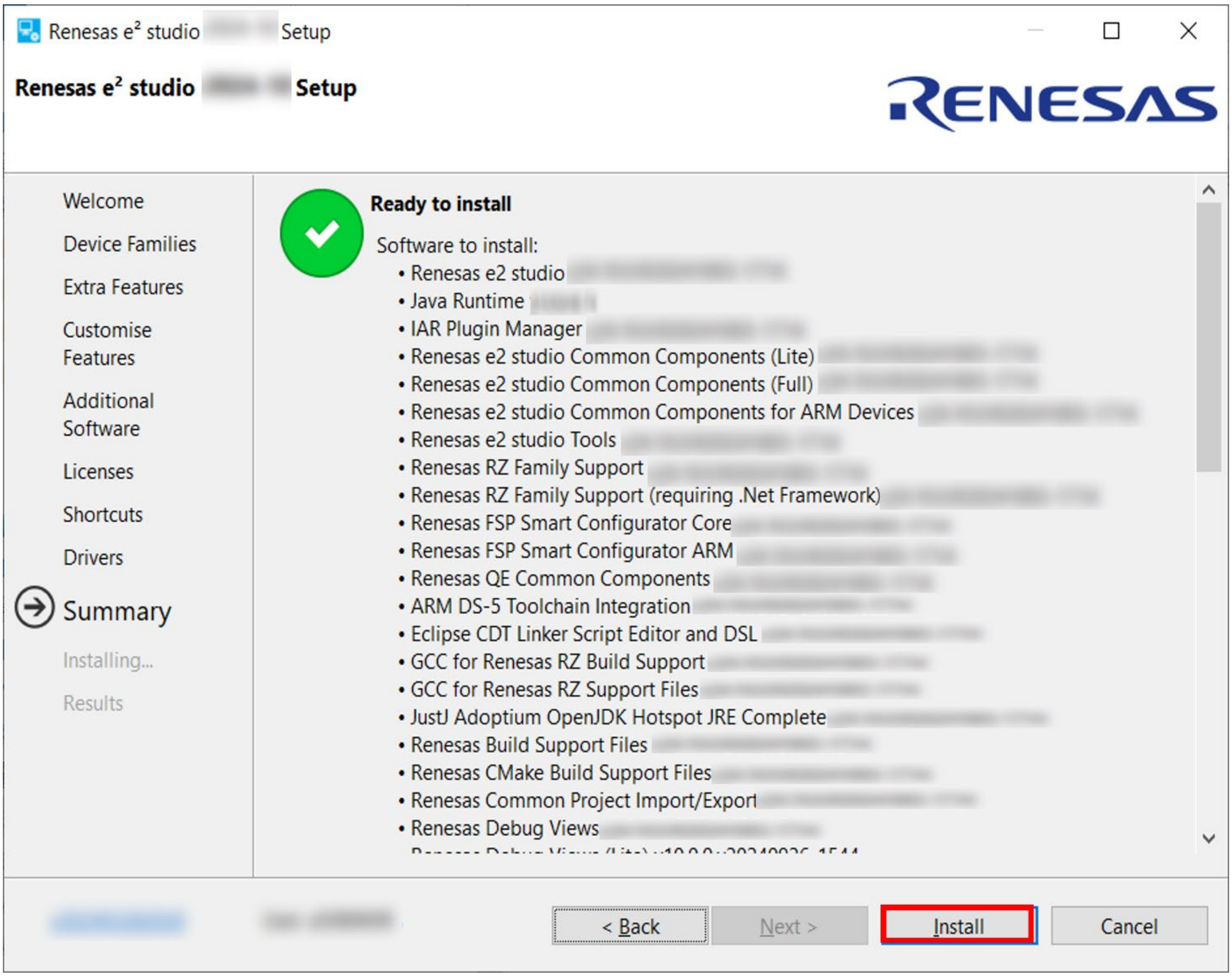


Shortcuts to important programs and files will be created in the following locations:

☑ In start menu group: Renesas Electronics e2 studio

Figure 9. Installation of e^2 studio – Shortcuts

9. Summary
   On the summary page, a list of components to be installed will be displayed. Please review the contents and click the [Install] button to proceed with the installation of the Renesas e^2 studio IDE.



Renesas e² studio [redacted] Setup

Ready to install

Software to install:
- Renesas e2 studio [redacted]
- Java Runtime [redacted]
- IAR Plugin Manager [redacted]
- Renesas e2 studio Common Components (Lite) [redacted]
- Renesas e2 studio Common Components (Full) [redacted]
- Renesas e2 studio Common Components for ARM Devices [redacted]
- Renesas e2 studio Tools [redacted]
- Renesas RZ Family Support [redacted]
- Renesas RZ Family Support (requiring .Net Framework) [redacted]
- Renesas FSP Smart Configurator Core [redacted]
- Renesas FSP Smart Configurator ARM [redacted]
- Renesas QE Common Components [redacted]
- ARM DS-5 Toolchain Integration [redacted]
- Eclipse CDT Linker Script Editor and DSL [redacted]
- GCC for Renesas RZ Build Support [redacted]
- GCC for Renesas RZ Support Files [redacted]
- JustJ Adoptium OpenJDK Hotspot JRE Complete [redacted]
- Renesas Build Support Files [redacted]
- Renesas CMake Build Support Files [redacted]
- Renesas Common Project Import/Export [redacted]
- Renesas Debug Views [redacted]

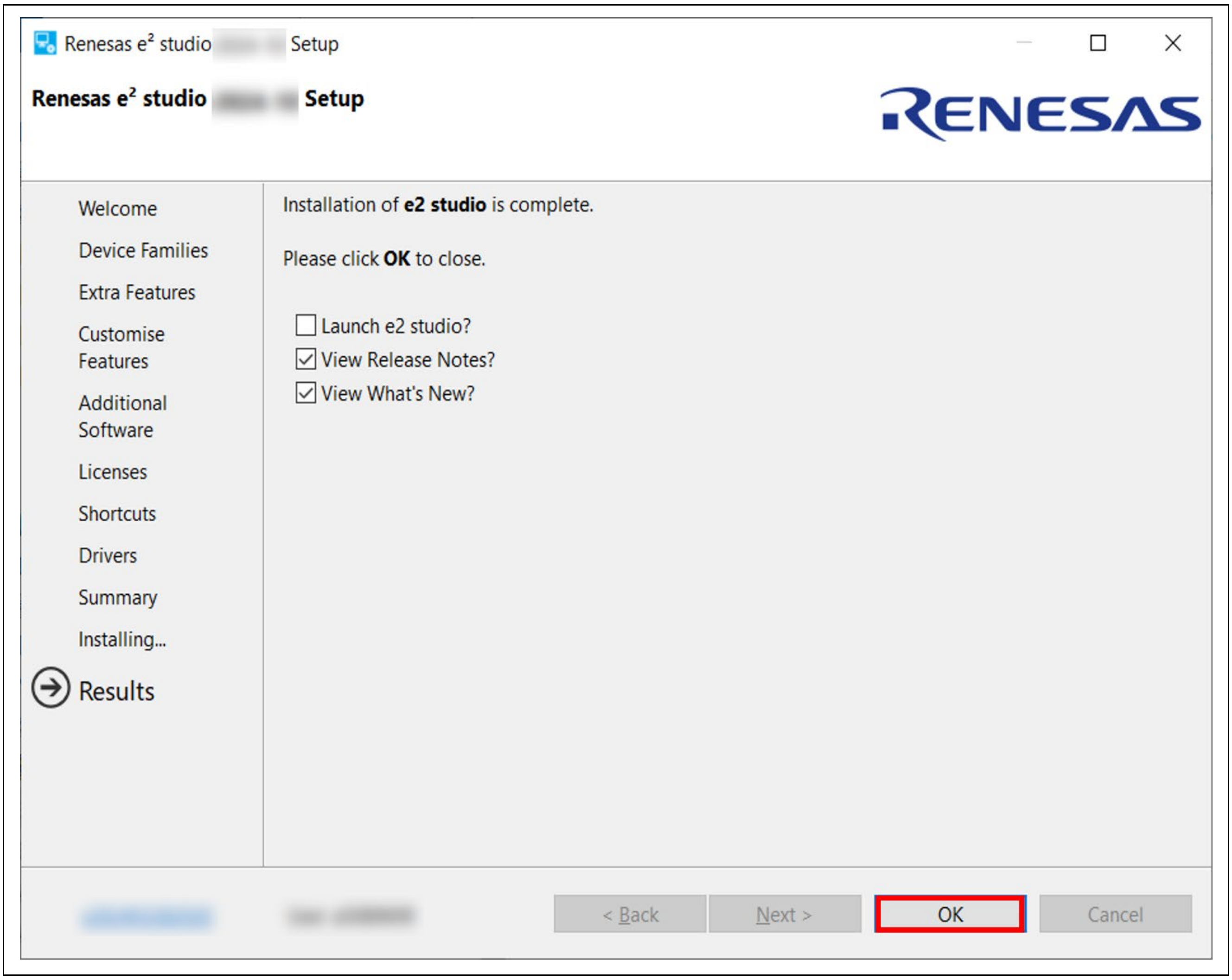
[redacted] [redacted]

Figure 10. Installation of e^2 studio – Summary

10. Installing…
    The installation will proceed. Depending on the selected items of the additional software, new dialog prompts may appear during the installation process. Please follow the instructions provided by the installer when this occurs.

11. Results

If the installation has been successfully completed, you should see the following information.



Renesas e² studio Setup

Renesas e² studio Setup

Installation of **e2 studio** is complete.

Please click **OK** to close.

☐ Launch e2 studio?
☑ View Release Notes?
☑ View What's New?

Figure 11. Results Page

2.1.4 e² studio installation for Linux PC

This chapter describes how to install the e² studio IDE on a Linux PC.

2.1.4.1 Prerequisite

Please download the development tool-related stuff:

- **SEGGER J-Link driver**
  Please download the driver V8.60:
  https://www.segger.com/downloads/jlink

- **e² studio IDE installer**
  The latest e² studio IDE installer package can be downloaded from Renesas website for free. Please check the detailed information from: https://www.renesas.com/e2studio.

2.1.4.2 Installation

This section describes the procedure of each software installation. Filename, version number, and the file path are just examples. Please replace those in accordance with your environment.

- **Segger J-Link driver**
  Open a terminal window and enter the commands stated below:

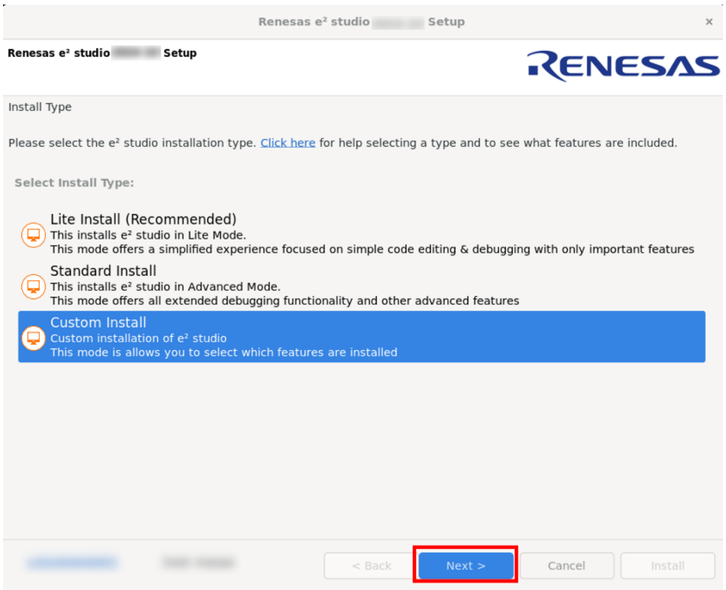
```
$ sudo dpkg -i JLink_Linux_V860_x86_64.deb
```

If the previous installation fails with unmet dependencies, please retry as follows:

```
$ sudo apt-get -f install
$ sudo dpkg -i JLink_Linux_V860_x86_64.deb
```

- **e² studio**

1. Run the e² studio IDE Installer "./e2studio_installer-yyyy-mm_linux_host.run". (Before running the installer, check the execution permission of the installer.
2. The user needs to select the Install Type as shown below. In this material, it is expected that Custom Install will be selected. Then, click [Next >] to continue.



Renesas e² studio [redacted] Setup

Renesas e² studio [redacted] Setup

Install Type

Please select the e² studio installation type. Click here for help selecting a type and to see what features are included.

Select Install Type:

○ Lite Install (Recommended)
This installs e² studio in Lite Mode.
This mode offers a simplified experience focused on simple code editing & debugging with only important features

○ Standard Install
This installs e² studio in Advanced Mode.
This mode offers all extended debugging functionality and other advanced features

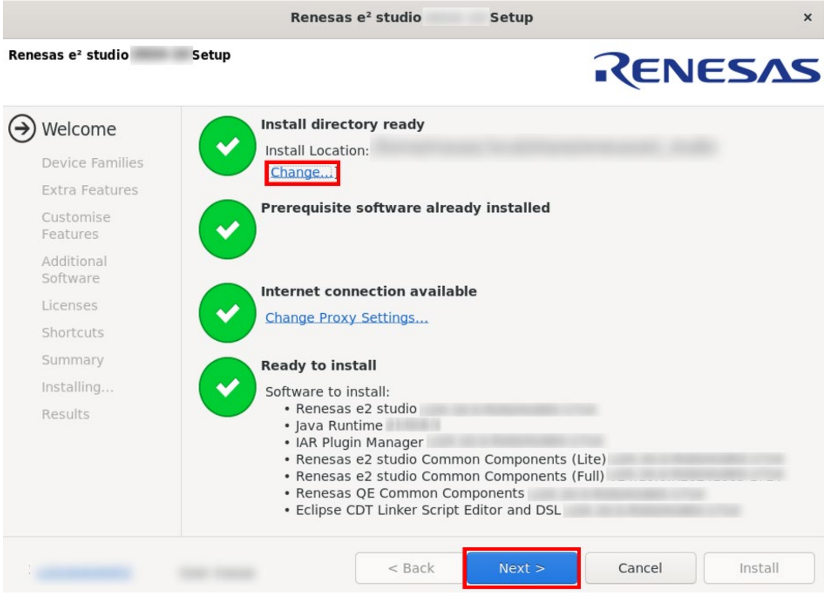
● Custom Install
Custom installation of e² studio
This mode is allows you to select which features are installed

Figure 12. Selection of Install Type

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

Note:

1. If you would like to have multiple versions of e² studio, please specify a new folder here.
2. Multi-byte characters cannot be used for the e² studio installation folder name.



Renesas e² studio Setup

Install directory ready

Install Location:

Prerequisite software already installed

Internet connection available

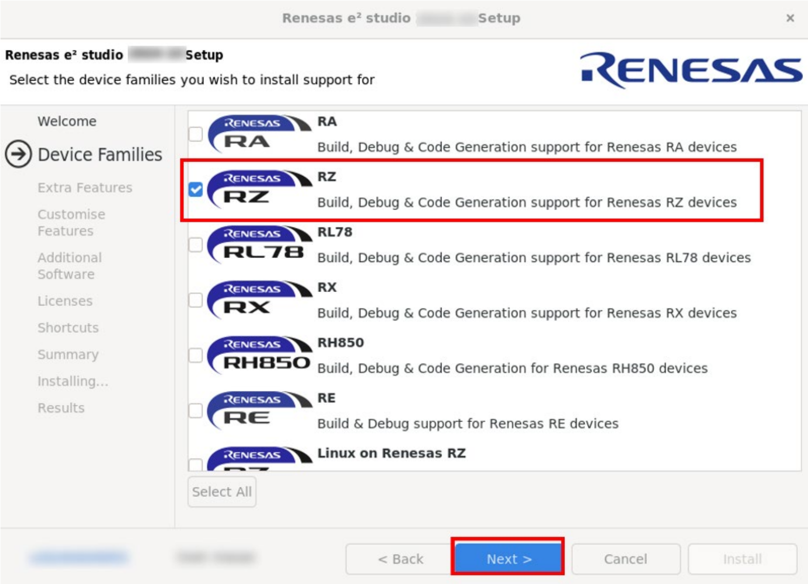
Ready to install

Software to install:

- Renesas e2 studio
- Java Runtime
- IAR Plugin Manager
- Renesas e2 studio Common Components (Lite)
- Renesas e2 studio Common Components (Full)
- Renesas QE Common Components
- Eclipse CDT Linker Script Editor and DSL

Figure 13. Installation of e² studio – Welcome page

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



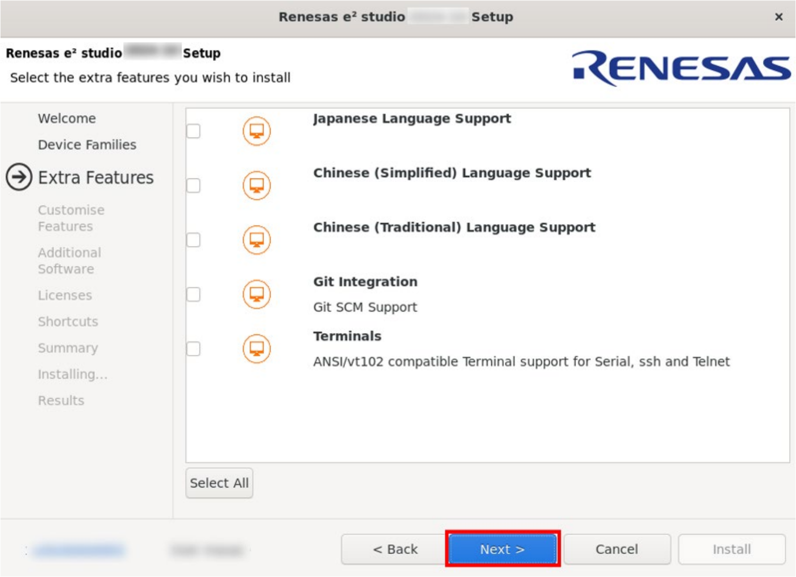
Renesas e² studio Setup

Select the device families you wish to install support for

- ☐ **RA** Build, Debug & Code Generation support for Renesas RA devices
- ☑ **RZ** Build, Debug & Code Generation support for Renesas RZ devices
- ☐ **RL78** Build, Debug & Code Generation support for Renesas RL78 devices
- ☐ **RX** Build, Debug & Code Generation support for Renesas RX devices
- ☐ **RH850** Build, Debug & Code Generation for Renesas RH850 devices
- ☐ **RE** Build & Debug support for Renesas RE devices
- ☐ **Linux on Renesas RZ**

Figure 14. Installation of e² studio – Device Families

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


Renesas e² studio Setup

Renesas e² studio Setup

Select the extra features you wish to install

☐ **Japanese Language Support**

☐ **Chinese (Simplified) Language Support**

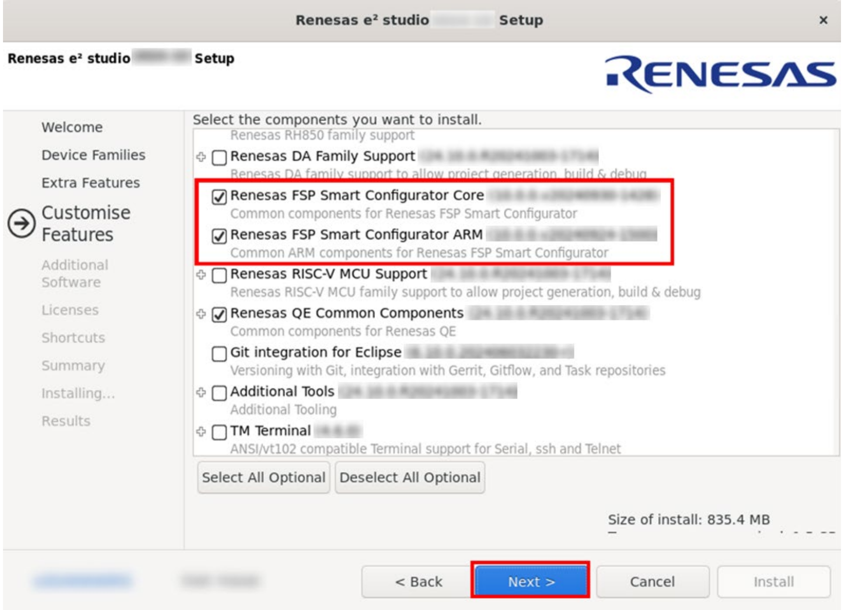
☐ **Chinese (Traditional) Language Support**

☐ **Git Integration**
Git SCM Support

☐ **Terminals**
ANSI/vt102 compatible Terminal support for Serial, ssh and Telnet

Figure 15. Installation of e^2 studio – Extra Features

5. Customize Features
   Select the components to install and click the [Next] button to continue. Be sure that “Renesas FSP Smart Configurator Core” is certainly selected.


Renesas e² studio Setup

Renesas e² studio Setup

Select the components you want to install.

Renesas RH850 family support
☐ Renesas DA Family Support
Renesas DA family support to allow project generation, build & debug
☑ Renesas FSP Smart Configurator Core
Common components for Renesas FSP Smart Configurator
☑ Renesas FSP Smart Configurator ARM
Common ARM components for Renesas FSP Smart Configurator
☐ Renesas RISC-V MCU Support
Renesas RISC-V MCU family support to allow project generation, build & debug
☑ Renesas QE Common Components
Common components for Renesas QE
☐ Git integration for Eclipse
Versioning with Git, integration with Gerrit, Gitflow, and Task repositories
☐ Additional Tools
Additional Tooling
☐ TM Terminal
ANSI/vt102 compatible Terminal support for Serial, ssh and Telnet

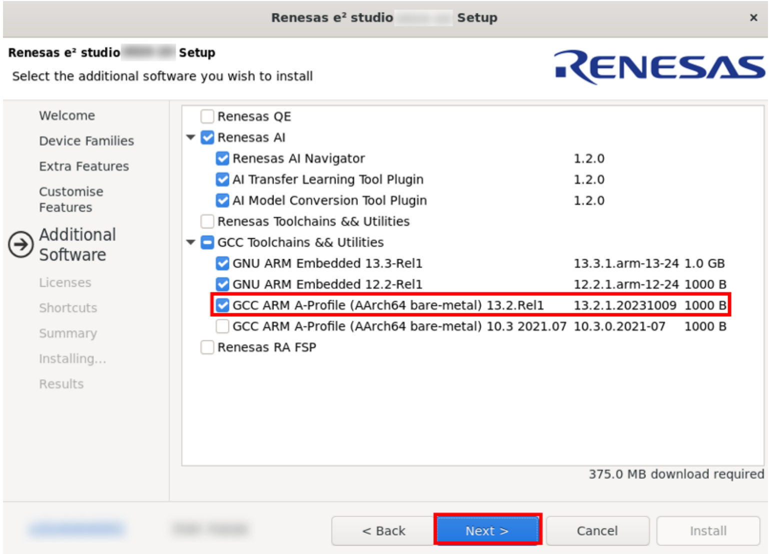
Size of install: 835.4 MB

Figure 16. Installation of e^2 studio – Features

6. Customize Features

   Select additional software (i.e., compilers, utilities, QE…) to be installed. Be sure that you select the "GCC toolchains & Utilities" tab, choose the following items, and click [Next] to continue.

   - GCC ARM A-Profile (AArch64 bare-metal) 13.2 rel1



Renesas e² studio [redacted] Setup

Select the additional software you wish to install

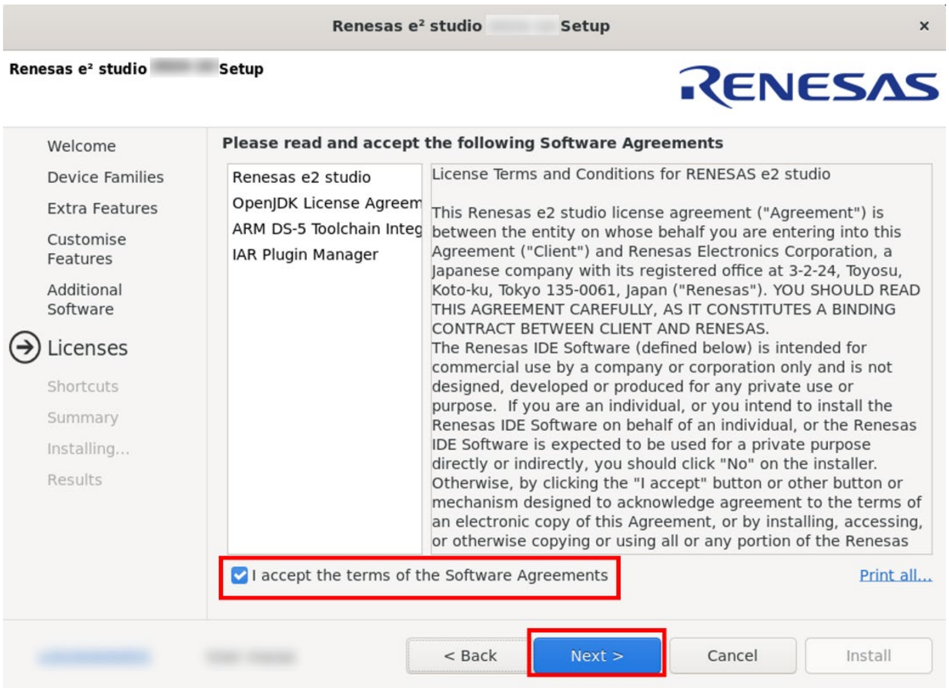
- ☐ Renesas QE
- ☑ Renesas AI
  - ☑ Renesas AI Navigator 1.2.0
  - ☑ AI Transfer Learning Tool Plugin 1.2.0
  - ☑ AI Model Conversion Tool Plugin 1.2.0
- ☐ Renesas Toolchains && Utilities
- ☑ GCC Toolchains && Utilities
  - ☑ GNU ARM Embedded 13.3-Rel1 13.3.1.arm-13-24 1.0 GB
  - ☑ GNU ARM Embedded 12.2-Rel1 12.2.1.arm-12-24 1000 B
  - ☑ GCC ARM A-Profile (AArch64 bare-metal) 13.2.Rel1 13.2.1.20231009 1000 B
  - ☐ GCC ARM A-Profile (AArch64 bare-metal) 10.3 2021.07 10.3.0.2021-07 1000 B
- ☐ Renesas RA FSP

375.0 MB download required

Figure 17. Installation of e^2 studio – Additional Software

7. License Agreement

   Read and accept the software license agreement. Click the [Next] button. Please note that the user must accept the license agreement for the installation to continue.



Please read and accept the following Software Agreements

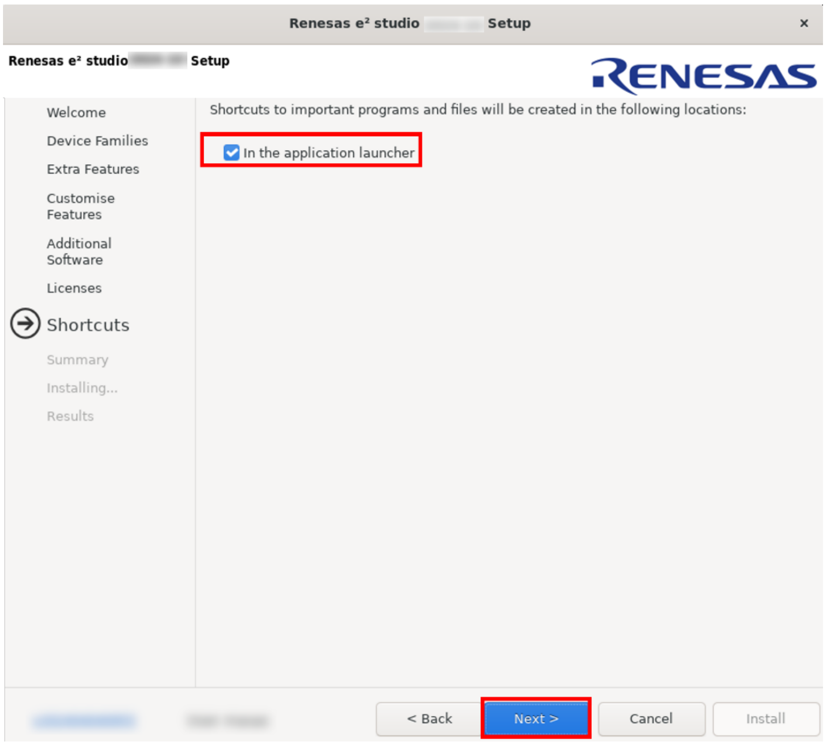
License Terms and Conditions for RENESAS e2 studio

This Renesas e2 studio license agreement ("Agreement") is between the entity on whose behalf you are entering into this Agreement ("Client") and Renesas Electronics Corporation, a Japanese company with its registered office at 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan ("Renesas"). YOU SHOULD READ THIS AGREEMENT CAREFULLY, AS IT CONSTITUTES A BINDING CONTRACT BETWEEN CLIENT AND RENESAS.
The Renesas IDE Software (defined below) is intended for commercial use by a company or corporation only and is not designed, developed or produced for any private use or purpose. If you are an individual, or you intend to install the Renesas IDE Software on behalf of an individual, or the Renesas IDE Software is expected to be used for a private purpose directly or indirectly, you should click "No" on the installer. Otherwise, by clicking the "I accept" button or other button or mechanism designed to acknowledge agreement to the terms of an electronic copy of this Agreement, or by installing, accessing, or otherwise copying or using all or any portion of the Renesas

☑ I accept the terms of the Software Agreements

Figure 18. Installation of e^2 studio – Licenses

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

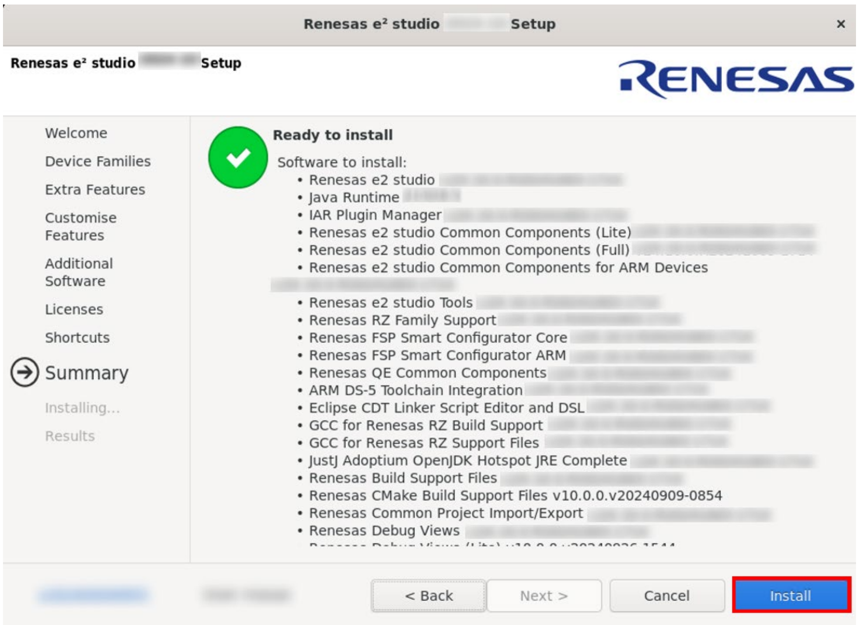


Shortcuts to important programs and files will be created in the following locations:

☑ In the application launcher

Figure 19. Installation of e^2 studio – Shortcuts

9. Summary
   The components to be installed are shown. Please confirm the contents and click the [Install] button to install the Renesas e^2 studio IDE.



Ready to install

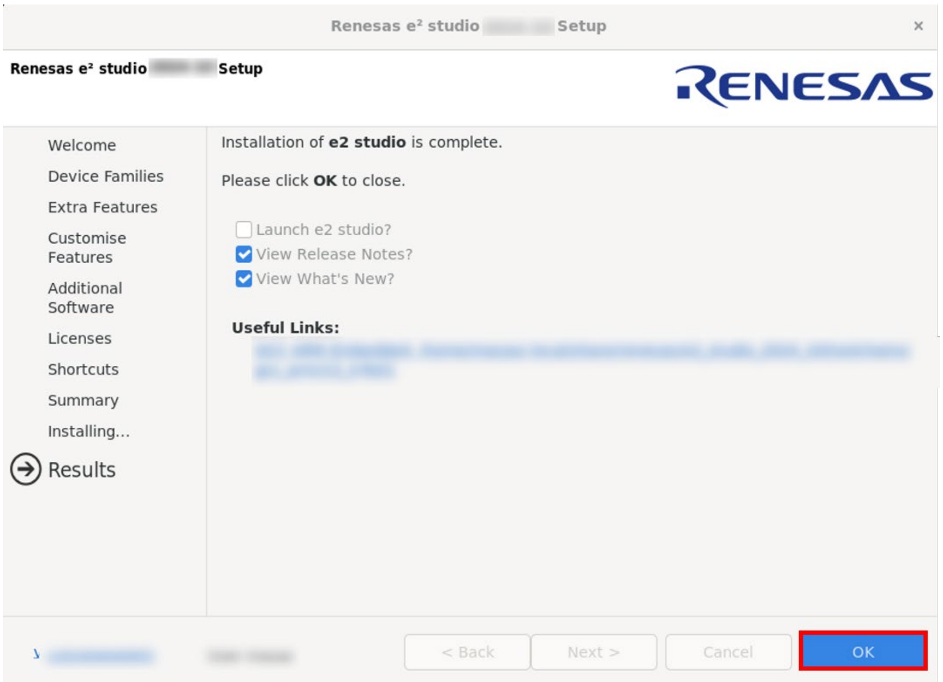
Software to install:

- Renesas e2 studio
- Java Runtime
- IAR Plugin Manager
- Renesas e2 studio Common Components (Lite)
- Renesas e2 studio Common Components (Full)
- Renesas e2 studio Common Components for ARM Devices
- Renesas e2 studio Tools
- Renesas RZ Family Support
- Renesas FSP Smart Configurator Core
- Renesas FSP Smart Configurator ARM
- Renesas QE Common Components
- ARM DS-5 Toolchain Integration
- Eclipse CDT Linker Script Editor and DSL
- GCC for Renesas RZ Build Support
- GCC for Renesas RZ Support Files
- JustJ Adoptium OpenJDK Hotspot JRE Complete
- Renesas Build Support Files
- Renesas CMake Build Support Files v10.0.0.v20240909-0854
- Renesas Common Project Import/Export
- Renesas Debug Views

Figure 20. Installation of e^2 studio – Summary

10. Installing...
    The installation is performed. Depending on the selected items of additional software, new dialog prompts may appear during the installation process. Please see the chapter 2.1.3.2 for more detailed information.

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



Installation of **e2 studio** is complete.

Please click **OK** to close.

☐ Launch e2 studio?
☑ View Release Notes?
☑ View What's New?

Useful Links:

Figure 21. Summary Page

2.2 FSP setup

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

2.2.1 Installation of FSP Packs using Platform Installer

This section describes how to install FSP using Platform Installer **setup_rzfsp_v4_0_0_e2s_v2025-12.exe** showcased here.

1. Double-click **setup_rzfsp_v4_0_0_e2s_v2025-12.exe**, select either **[Quick Install]** or **[Custom Install],** and click **[Next >]** when the installation wizard is shown. When you choose **[Quick Install]**, you can jump to 6. Licenses.

Note:

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

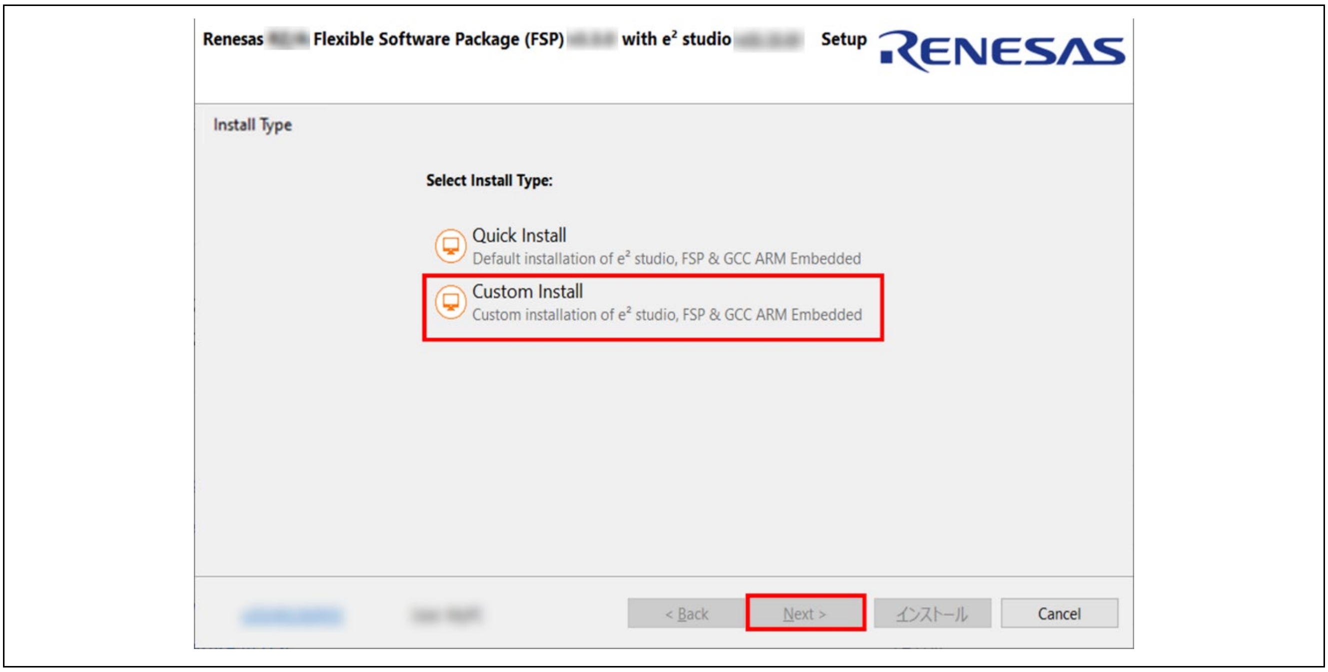
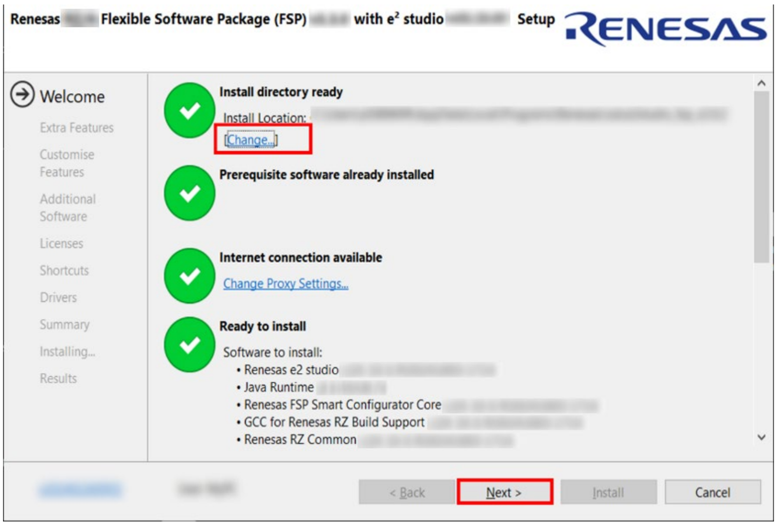


Figure 22. FSP Platform Installation Wizard

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

Note:

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



Install directory ready

Install Location:

[Change...]

Prerequisite software already installed

Internet connection available

Change Proxy Settings...

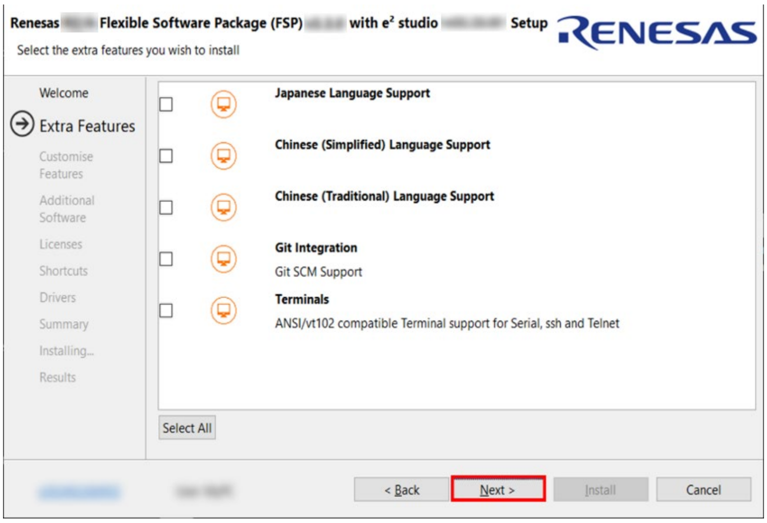
Ready to install

Software to install:

- Renesas e2 studio
- Java Runtime
- Renesas FSP Smart Configurator Core
- GCC for Renesas RZ Build Support
- Renesas RZ Common

Figure 23. FSP Platform Installer – Welcome page

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



Select the extra features you wish to install

☐ **Japanese Language Support**

☐ **Chinese (Simplified) Language Support**

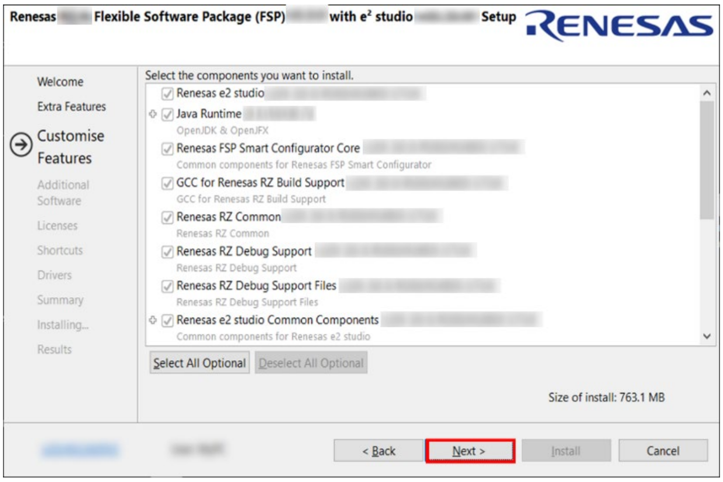
☐ **Chinese (Traditional) Language Support**

☐ **Git Integration**
Git SCM Support

☐ **Terminals**
ANSI/vt102 compatible Terminal support for Serial, ssh and Telnet

Figure 24. FSP Platform Installer – Extra Features

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


Renesas [redacted] Flexible Software Package (FSP) [redacted] with e² studio [redacted] Setup

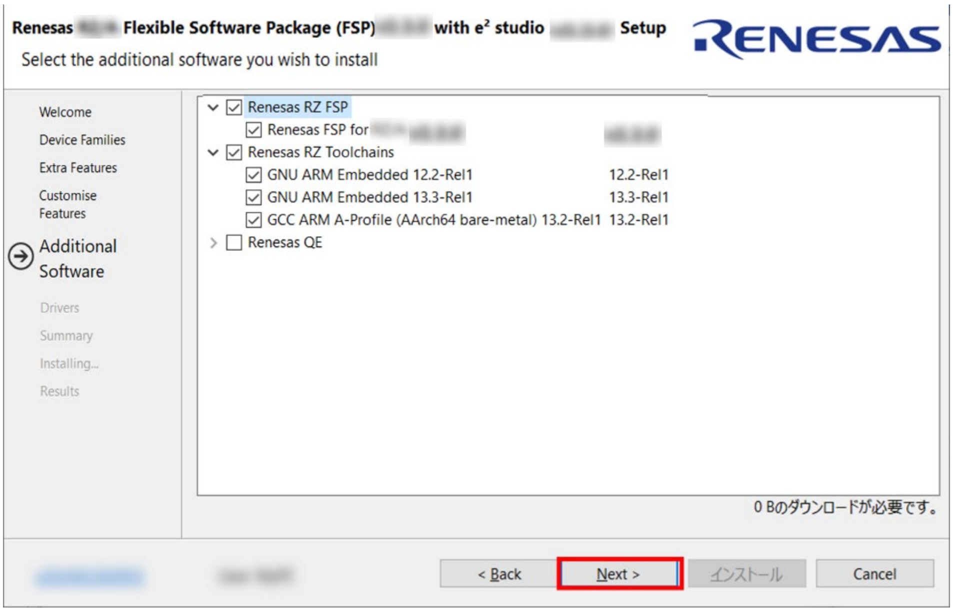
Select the components you want to install.

- ☑ Renesas e2 studio [redacted]
- ☑ Java Runtime [redacted]
  OpenJDK & OpenJFX
- ☑ Renesas FSP Smart Configurator Core [redacted]
  Common components for Renesas FSP Smart Configurator
- ☑ GCC for Renesas RZ Build Support [redacted]
  GCC for Renesas RZ Build Support
- ☑ Renesas RZ Common [redacted]
  Renesas RZ Common
- ☑ Renesas RZ Debug Support [redacted]
  Renesas RZ Debug Support
- ☑ Renesas RZ Debug Support Files [redacted]
  Renesas RZ Debug Support Files
- ☑ Renesas e2 studio Common Components [redacted]
  Common components for Renesas e2 studio

Size of install: 763.1 MB

Figure 25. FSP Platform Installer – Features

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



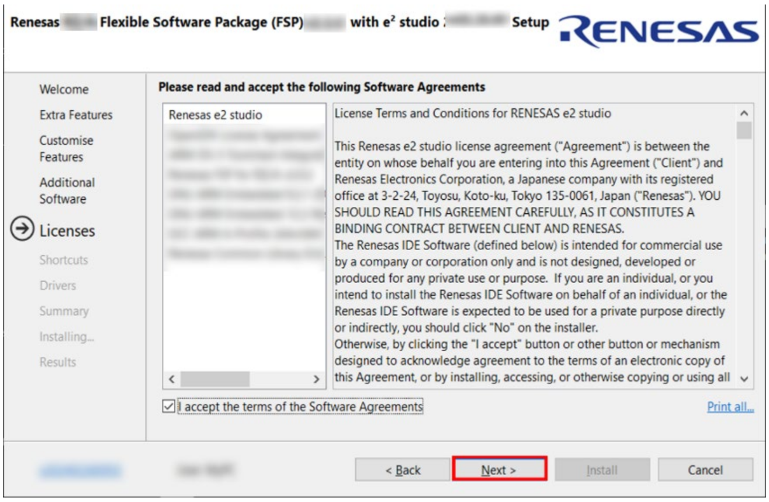
Select the additional software you wish to install

- ☑ Renesas RZ FSP
  - ☑ Renesas FSP for [redacted] [redacted]
- ☑ Renesas RZ Toolchains
  - ☑ GNU ARM Embedded 12.2-Rel1 12.2-Rel1
  - ☑ GNU ARM Embedded 13.3-Rel1 13.3-Rel1
  - ☑ GCC ARM A-Profile (AArch64 bare-metal) 13.2-Rel1 13.2-Rel1
- ☐ Renesas QE

0 Bのダウンロードが必要です。

Figure 26. FSP Platform Installer – Additional Software

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



Please read and accept the following Software Agreements

License Terms and Conditions for RENESAS e2 studio

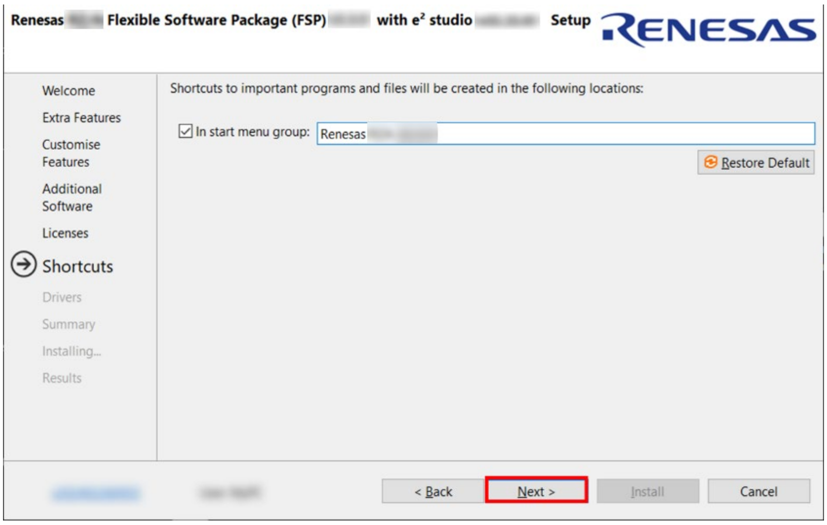
This Renesas e2 studio license agreement ("Agreement") is between the entity on whose behalf you are entering into this Agreement ("Client") and Renesas Electronics Corporation, a Japanese company with its registered office at 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan ("Renesas"). YOU SHOULD READ THIS AGREEMENT CAREFULLY, AS IT CONSTITUTES A BINDING CONTRACT BETWEEN CLIENT AND RENESAS.
The Renesas IDE Software (defined below) is intended for commercial use by a company or corporation only and is not designed, developed or produced for any private use or purpose. If you are an individual, or you intend to install the Renesas IDE Software on behalf of an individual, or the Renesas IDE Software is expected to be used for a private purpose directly or indirectly, you should click "No" on the installer.
Otherwise, by clicking the "I accept" button or other button or mechanism designed to acknowledge agreement to the terms of an electronic copy of this Agreement, or by installing, accessing, or otherwise copying or using all

☑ I accept the terms of the Software Agreements

Figure 27. FSP Platform Installer – Licenses

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

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

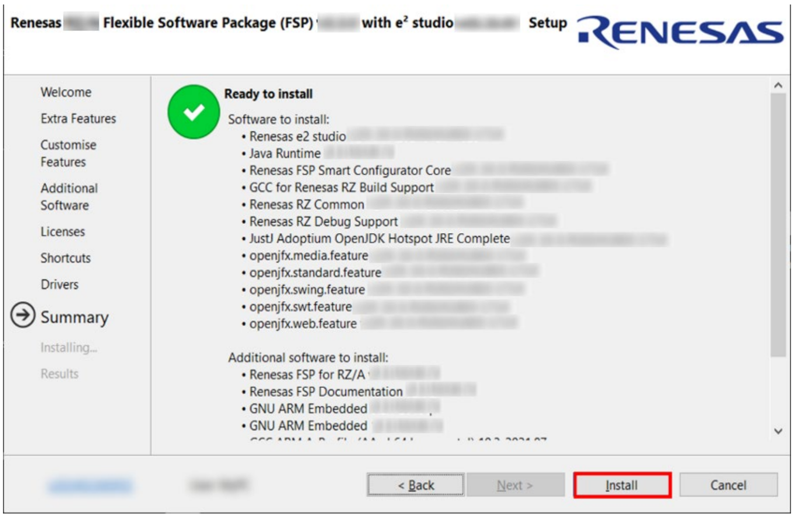


Shortcuts to important programs and files will be created in the following locations:

☑ In start menu group: Renesas [redacted]

Figure 28. FSP Platform Installer – Shortcuts

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



Ready to install

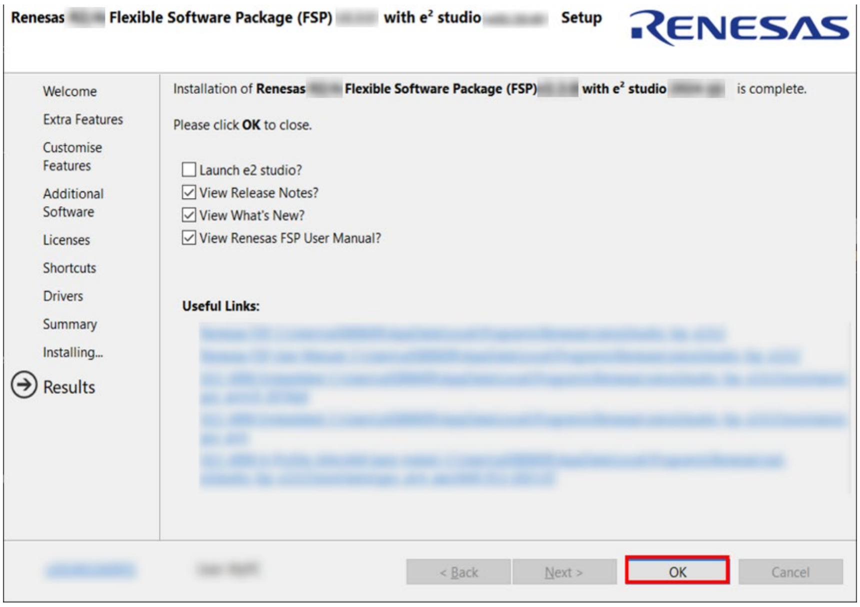
Software to install:
- Renesas e2 studio [redacted]
- Java Runtime [redacted]
- Renesas FSP Smart Configurator Core [redacted]
- GCC for Renesas RZ Build Support [redacted]
- Renesas RZ Common [redacted]
- Renesas RZ Debug Support [redacted]
- JustJ Adoptium OpenJDK Hotspot JRE Complete [redacted]
- openjfx.media.feature [redacted]
- openjfx.standard.feature [redacted]
- openjfx.swing.feature [redacted]
- openjfx.swt.feature [redacted]
- openjfx.web.feature [redacted]

Additional software to install:
- Renesas FSP for RZ/A [redacted]
- Renesas FSP Documentation [redacted]
- GNU ARM Embedded [redacted]
- GNU ARM Embedded [redacted]

Figure 29. FSP Platform Installer – Summary

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

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



Installation of **Renesas** **Flexible Software Package (FSP)** **with e² studio** is complete.

Please click **OK** to close.

☐ Launch e2 studio?
☑ View Release Notes?
☑ View What's New?
☑ View Renesas FSP User Manual?

Useful Links:

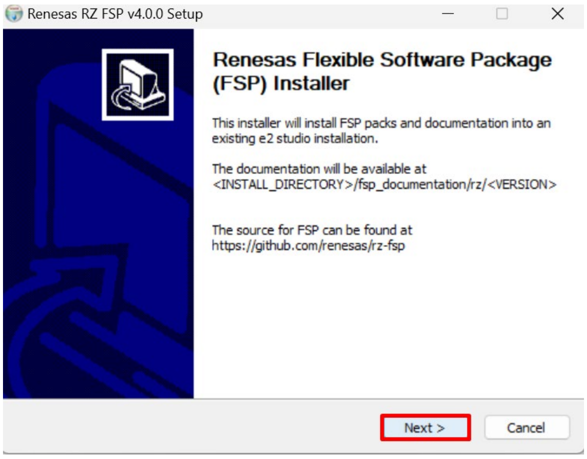
Figure 30. Installation Results of FSP Platform Installer

2.2.2 Installation of FSP Packs using Package Installer

Package Installer **RZ_FSP_Packs_v4.0.0.exe** is showcased here. Please note that it's for the Windows Host PC only.

Here is the procedure:

1. Exit e² studio.
2. Invoke **RZ_FSP_Packs_v4.0.0.exe**.
3. Click [Next >] to start the installation.



Renesas Flexible Software Package (FSP) Installer

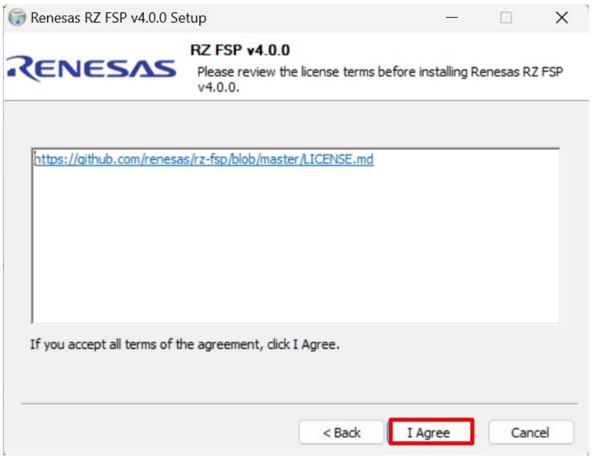
This installer will install FSP packs and documentation into an existing e2 studio installation.

The documentation will be available at <INSTALL_DIRECTORY>/fsp_documentation/rz/<VERSION>

The source for FSP can be found at https://github.com/renesas/rz-fsp

Figure 31. FSP Package Installer

4. See the license term and click [I Agree] if it's acceptable



RZ FSP v4.0.0

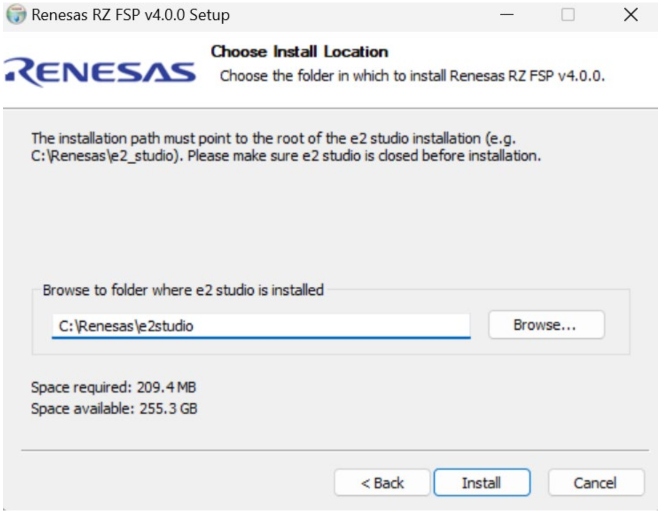
Please review the license terms before installing Renesas RZ FSP v4.0.0.

https://github.com/renesas/rz-fsp/blob/master/LICENSE.md

If you accept all terms of the agreement, click I Agree.

Figure 32. FSP License Term

5. Specify the e^2 studio installation folder (e.g., C:\Renesas\e2studio) and click [Install].



Choose Install Location

Choose the folder in which to install Renesas RZ FSP v4.0.0.

The installation path must point to the root of the e2 studio installation (e.g. C:\Renesas\e2_studio). Please make sure e2 studio is closed before installation.

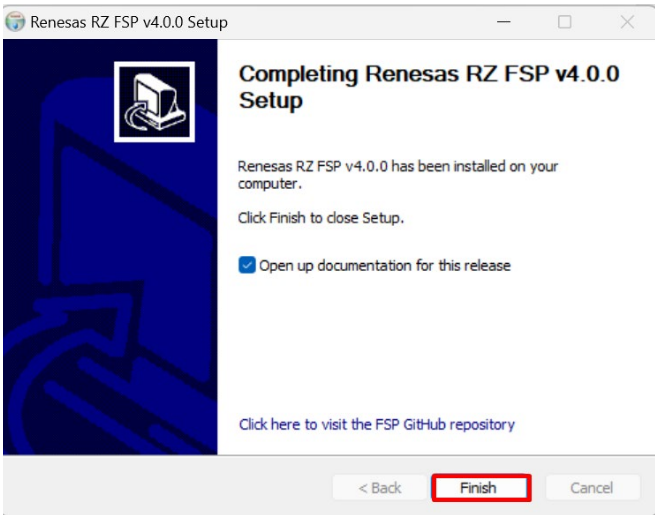
Browse to folder where e2 studio is installed

C:\Renesas\e2studio

Space required: 209.4 MB
Space available: 255.3 GB

Figure 33. FSP Installation

6. Click [Finish] to complete the installation.



Renesas RZ FSP v4.0.0 Setup

Completing Renesas RZ FSP v4.0.0 Setup

Renesas RZ FSP v4.0.0 has been installed on your computer.

Click Finish to close Setup.

☑ Open up documentation for this release

Click here to visit the FSP GitHub repository

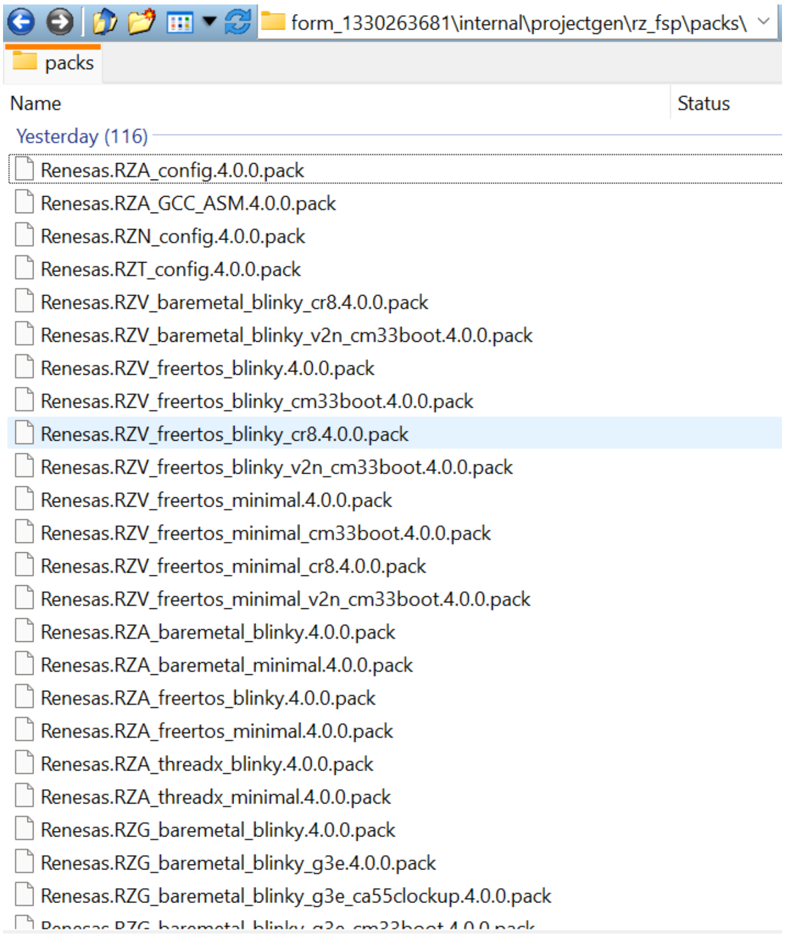
Figure 34. Completion of FSP Installation

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

2.2.3 Installation of FSP Packs using a Package Zip file

No Package installer is available for the Linux Host PC, and therefore, you need to install FSP Packs with **RZ_FSP_Packs_v4.0.0.zip**. This section describes how to install it. Please note that the same installation procedure is valid for the Windows Host PC.

1. Download **RZ_FSP_Packs_v4.0.0.zip** from here.
2. Extract the zip file to e^2 studio installation directory.
   If the FSP Packs are successfully extracted, **rz_fsp/packs** directory is placed at the location below:
   - **<e^2 studio installation directory>/internal/projectgen**

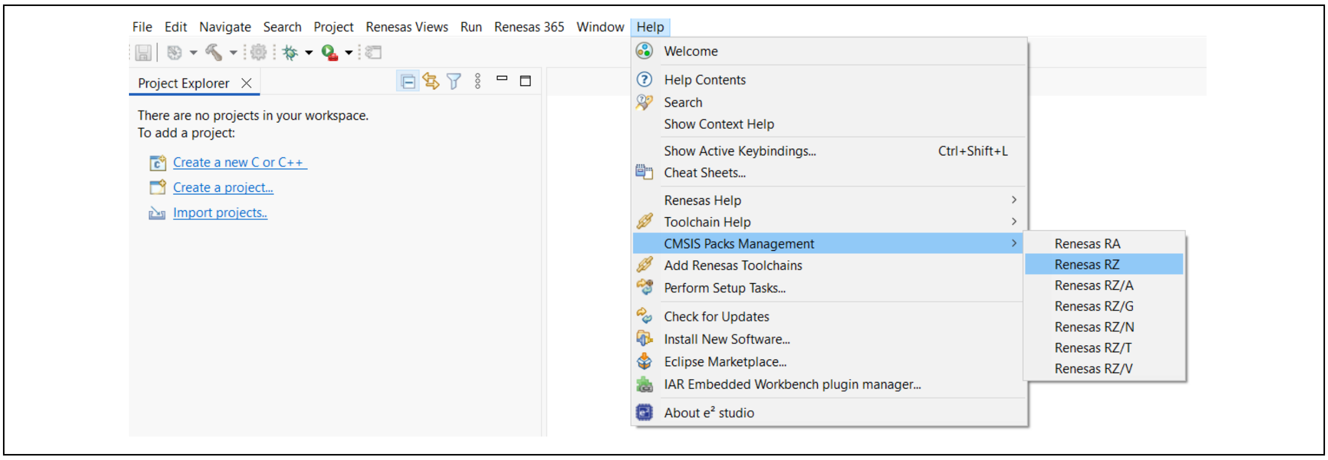


Renesas.RZA_config.4.0.0.pack
Renesas.RZA_GCC_ASM.4.0.0.pack
Renesas.RZN_config.4.0.0.pack
Renesas.RZT_config.4.0.0.pack
Renesas.RZV_baremetal_blinky_cr8.4.0.0.pack
Renesas.RZV_baremetal_blinky_v2n_cm33boot.4.0.0.pack
Renesas.RZV_freertos_blinky.4.0.0.pack
Renesas.RZV_freertos_blinky_cm33boot.4.0.0.pack
Renesas.RZV_freertos_blinky_cr8.4.0.0.pack
Renesas.RZV_freertos_blinky_v2n_cm33boot.4.0.0.pack
Renesas.RZV_freertos_minimal.4.0.0.pack
Renesas.RZV_freertos_minimal_cm33boot.4.0.0.pack
Renesas.RZV_freertos_minimal_cr8.4.0.0.pack
Renesas.RZV_freertos_minimal_v2n_cm33boot.4.0.0.pack
Renesas.RZA_baremetal_blinky.4.0.0.pack
Renesas.RZA_baremetal_minimal.4.0.0.pack
Renesas.RZA_freertos_blinky.4.0.0.pack
Renesas.RZA_freertos_minimal.4.0.0.pack
Renesas.RZA_threadx_blinky.4.0.0.pack
Renesas.RZA_threadx_minimal.4.0.0.pack
Renesas.RZG_baremetal_blinky.4.0.0.pack
Renesas.RZG_baremetal_blinky_g3e.4.0.0.pack
Renesas.RZG_baremetal_blinky_g3e_ca55clockup.4.0.0.pack
Renesas.RZG_baremetal_blinky_g3e_cm33boot.4.0.0.pack

Figure 35. FSP Packs in the e^2 studio installation directory

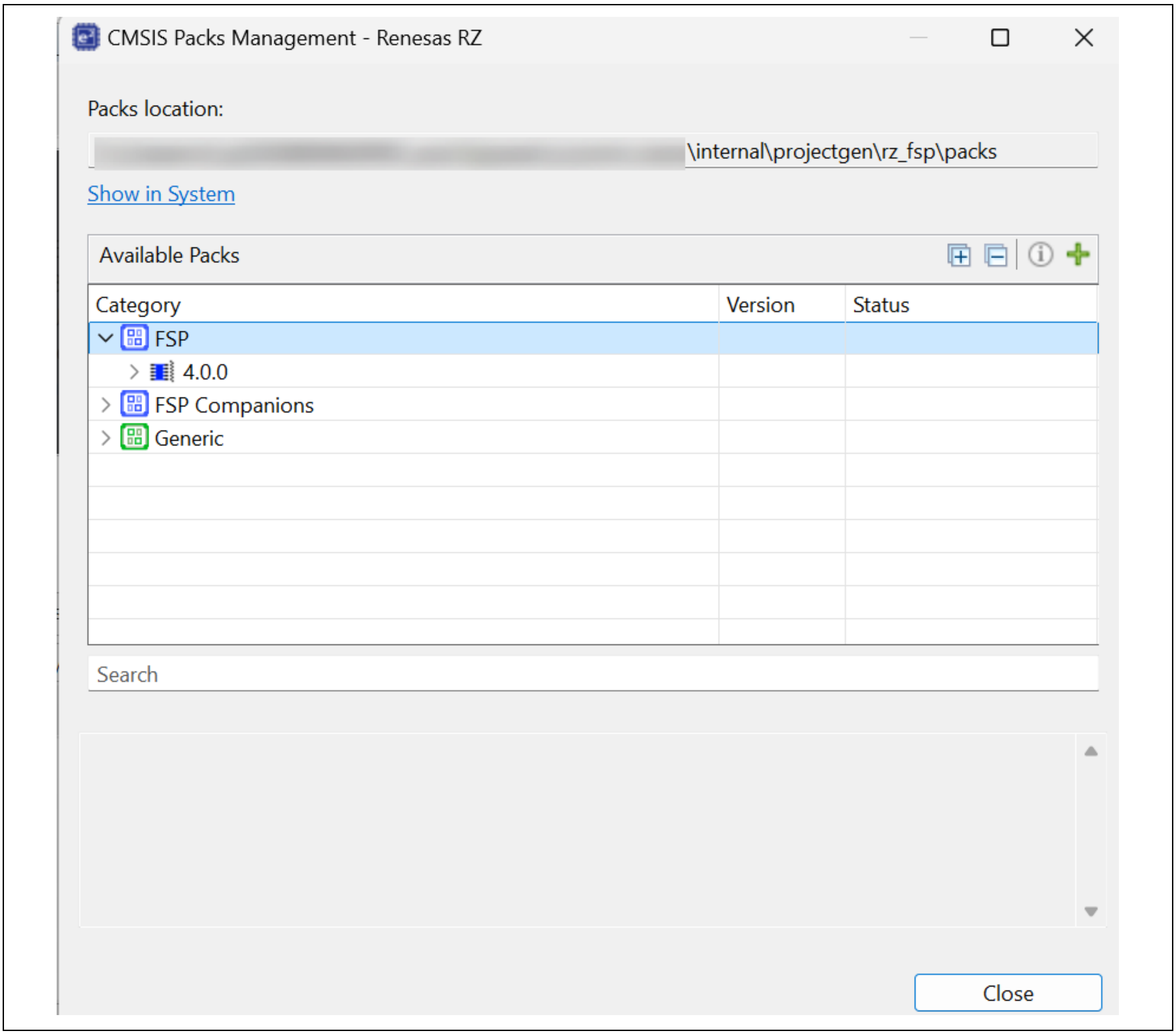
3. At the 1st invocation of e^2 studio after you carry out the above procedure, FSP Packs should be installed automatically.

4. You can check if the installation is successful by following the procedure:
   Click **Help** > **CMSIS Packs Management** > **Renesas RZ**.



There are no projects in your workspace.
To add a project:

If FSP is successfully installed, 4.0.0 should be listed under FSP as shown below:



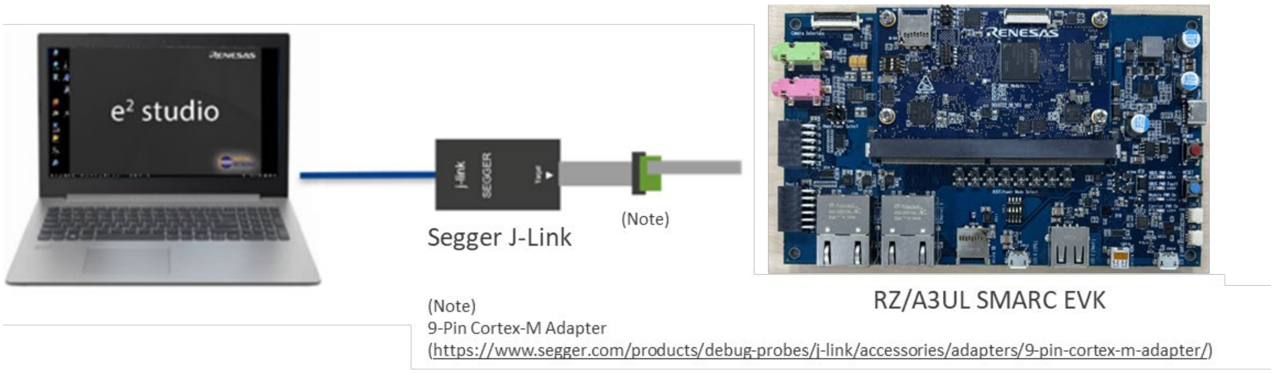
Packs location:

[redacted]\internal\projectgen\rz_fsp\packs

Category	Version	Status
FSP		
4.0.0		
FSP Companions		
Generic		

3. Set Up a Target Board

Below is an example of a typical system configuration.



(Note)
9-Pin Cortex-M Adapter
(https://www.segger.com/products/debug-probes/j-link/accessories/adapters/9-pin-cortex-m-adapter/)

Figure 36. System Configuration Example – SMARC EVK

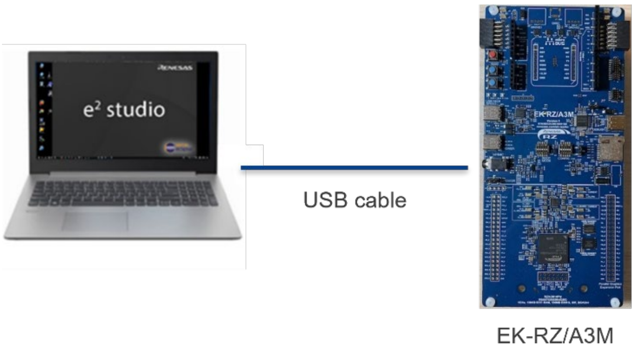


Figure 37. System Configuration Example – EK-RZ/A3M

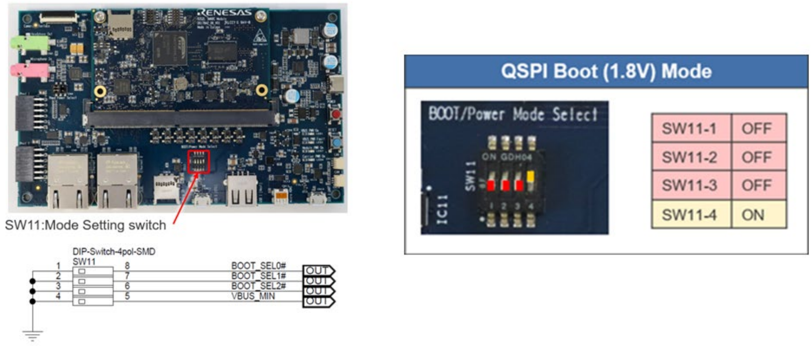
3.1 Supported Debugger

- SEGGER J-Link
  For details on SEGGER J-Link, please see J-Link Debug Probes by SEGGER – the Embedded Experts.
  In the case of EK-RZ/A3M, the on-board debug functionality is provided using Renesas RA4M2 Debug MPU and SEGGER J-Link® firmware as J-Link OB. For details, please refer to 3.2.2.

3.2 Board Setup

3.2.1 Boot Mode

For SMARC EVK, set up the SW11 as follows to configure Boot Mode 3 (QSPI or OCTA Boot (1.8V) Mode).



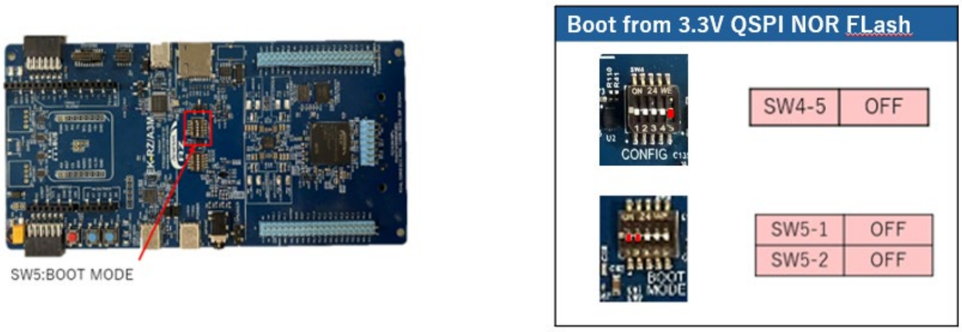
QSPI Boot (1.8V) Mode

SW11-1	OFF
SW11-2	OFF
SW11-3	OFF
SW11-4	ON

Figure 38. Boot MODE – SMARC EVK

For EK-RZ/A3M,

Set up the SW4 and SW5 as follows to configure Boot Mode 4 (Boot from 3.3V QSPI NOR flash).



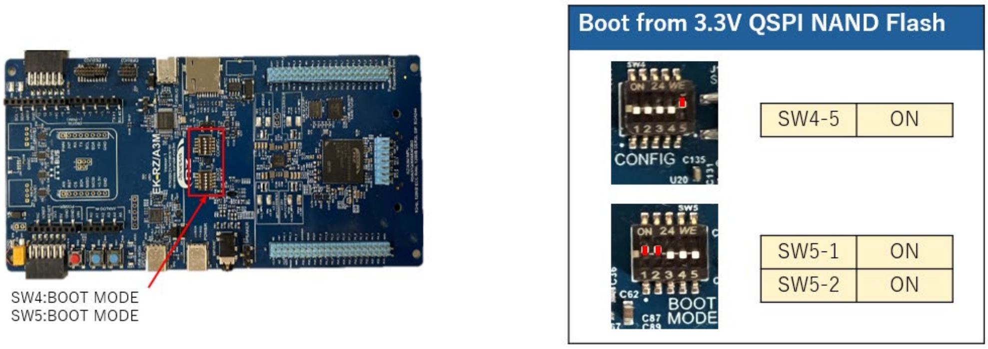
Boot from 3.3V QSPI NOR FLash

SW4-5	OFF

SW5-1	OFF
SW5-2	OFF

Figure 39: Boot MODE 4 – EK-RZ/A3M

Set up the SW4 and SW5 as follows to configure Boot Mode 7 (Boot from 3.3V QSPI NAND flash).



Boot from 3.3V QSPI NAND Flash

SW4-5	ON

SW5-1	ON
SW5-2	ON

Figure 40. Boot MODE 7 – EK-RZ/A3M

(Additional) Serial NAND Flash may degrade due to the increased frequency of memory access, which can result in the occurrence of bad blocks (blocks that cannot be accessed correctly). To address this issue, in **Boot Mode 7** the system checks whether loading the user-written Loader Program from Serial NAND Flash is successful. If the load fails, it will repeatedly attempt to read the Loader Program data from the next block.

Therefore, users need to prepare for possible read failures by writing the same Loader Program into the **first four blocks** of the Serial NAND Flash (see Figure 41). If any bad blocks are included among the first four blocks, writing to those blocks should be skipped (see Figure 42). Please note that the application image must always be written starting from the fourth physical block.

For detailed information on the boot procedure, refer to the RZ/A3M Group User's Manual: Hardware – Chapter 4, "Boot Mode".

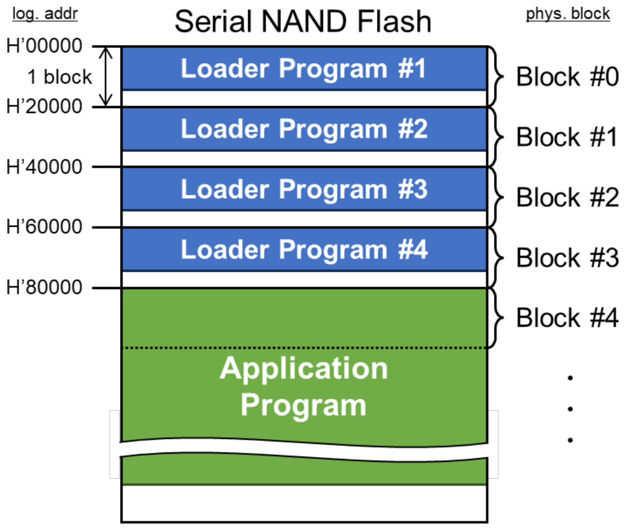


Figure 41. Serial NAND Flash Write Image – Case where no bad blocks are present in the first four blocks

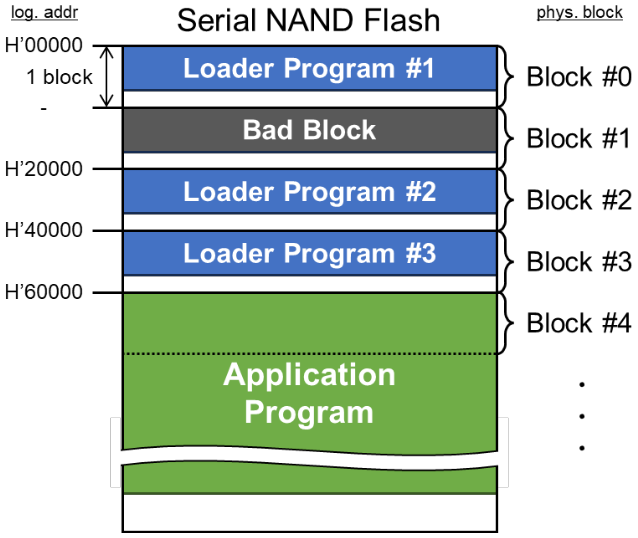
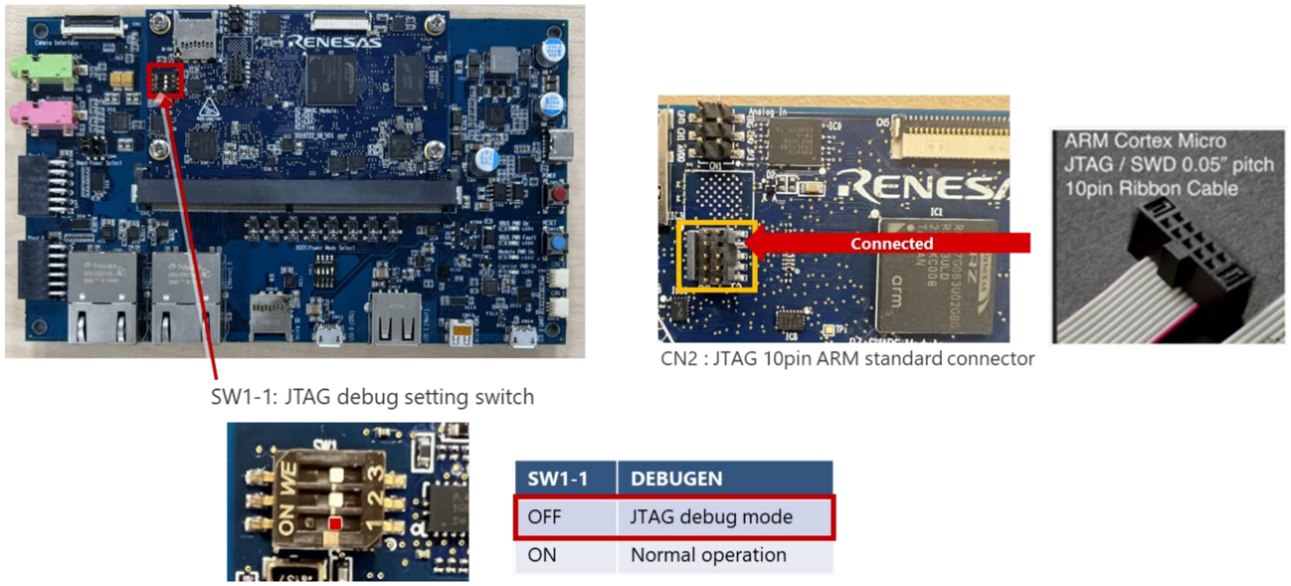


Figure 42. Serial NAND Flash Write Image – Case where one bad block is present in the first four blocks

3.2.2 JTAG connection

When connecting JTAG to SMARC EVK, you must set the DIP SW1 settings as follows:



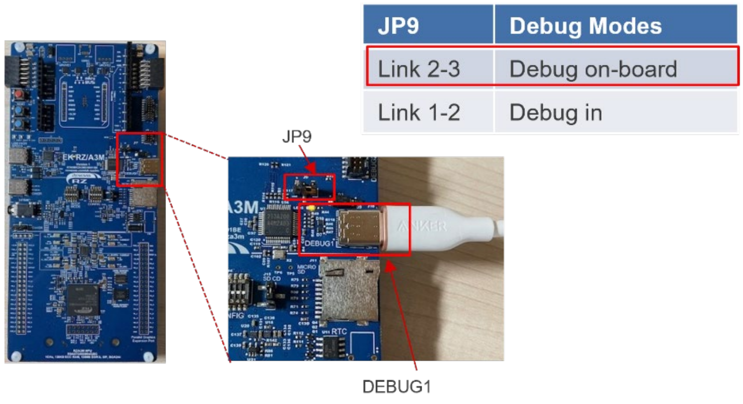
SW1-1	DEBUGEN
OFF	JTAG debug mode
ON	Normal operation

Figure 43. JTAG connection – SMARC EVK

Please note that RZ/A3UL SMARC EVK has a CoreSight 10 connector and therefore, the following adapter is needed to connect to Segger J-Link.

https://www.segger.com/products/debug-probes/j-link/accessories/adapters/9-pin-cortex-m-adapter/

In the case of EK-RZ/A3M, the on-board debug functionality is supported. The user can debug by connecting the DEBUG1 connector on the EK-RZ/A3M with the PC. When using on-board debug functionality, you must set the J9 as follows.



JP9	Debug Modes
Link 2-3	Debug on-board
Link 1-2	Debug in

Figure 44. JTAG connection – EK-RZ/A3M

3.2.3 Debug Serial (console output)

For SMARC EVK, debug serial uses CN14. The baud rate is 115200bps.

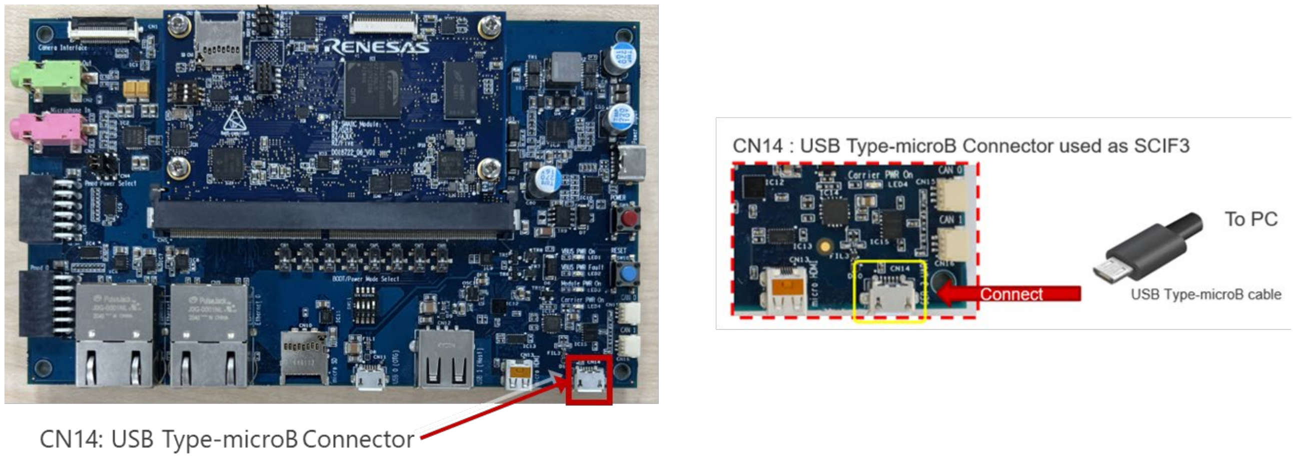
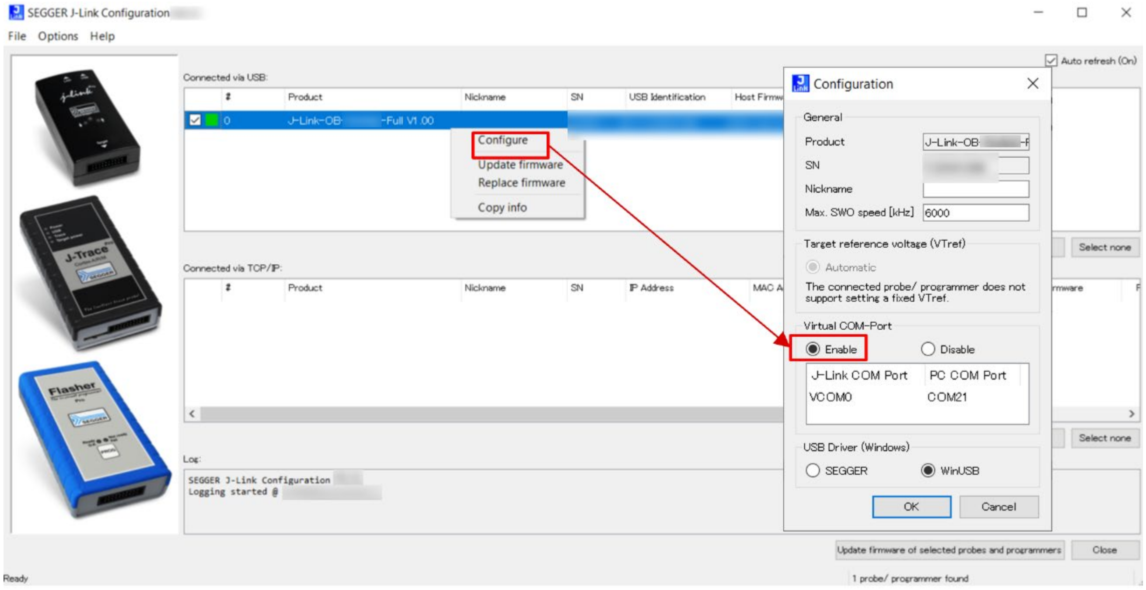


Figure 45. Debug Serial (console output) – SMARC EVK

For EK-RZ/A3M, on-board debug functionality supports debug serial output. You must configure the J-Link OB to enable the Virtual COM-Port after connecting the board to your PC. And the baud rate is 115200bps.



Connected via USB: 0 J-Link-OB-[redacted]-Full V1.00

Configuration

General
- Product: J-Link-OB-[redacted]
- SN: [redacted]
- Nickname:
- Max. SWO speed [kHz]: 6000

Target reference voltage (VTref)
- ● Automatic

The connected probe/ programmer does not support setting a fixed VTref.

Virtual COM-Port
- ● Enable ○ Disable

J-Link COM Port	PC COM Port
VCOM0	COM21

USB Driver (Windows)
- ○ SEGGER ● WinUSB

```
SEGGER J-Link Configuration [redacted]
Logging started @ [redacted]
```

1 probe/ programmer found

J-Link Configurator

Figure 46. Debug Serial (console output) – EK-RZ/A3M

3.2.4 Power Supply

Here are the proven power supply-related goods to be used in Renesas' development. Please prepare for the equivalent ones for your development.

- USB Type-C cable CB-CD23BK (manufactured by Aukey)
- USB PD Charger Anker PowerPort III 65W Pod (manufactured by Anker)

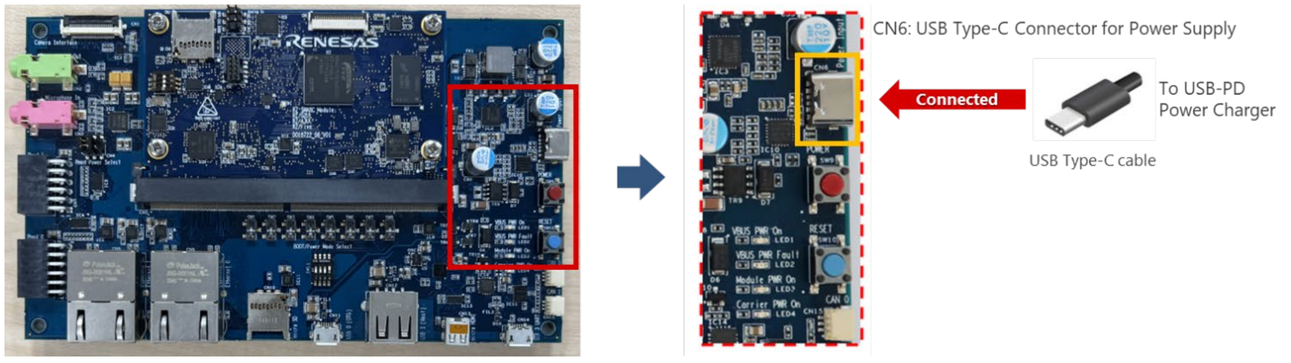


Figure 47. Power Supply

For the power supply, please follow the following procedure:

SMARC EVK

1. Connect USB-PD Power Charger to USB Type-C Connector (CN6).
   Once the USB-PD Power Charger is connected to the CN6, LED1 (VBUS PWR ON) and LED3 (Module PWR ON) should light up.
2. Press the power button (SW9) to turn on the power
   When turning on the power, you need to press and hold the power button for 1 second. Also, the power button should be pressed and held for 2 seconds to turn off the power.
3. If the power supply is successful, LED4 (Carrier PWR On) should light up.

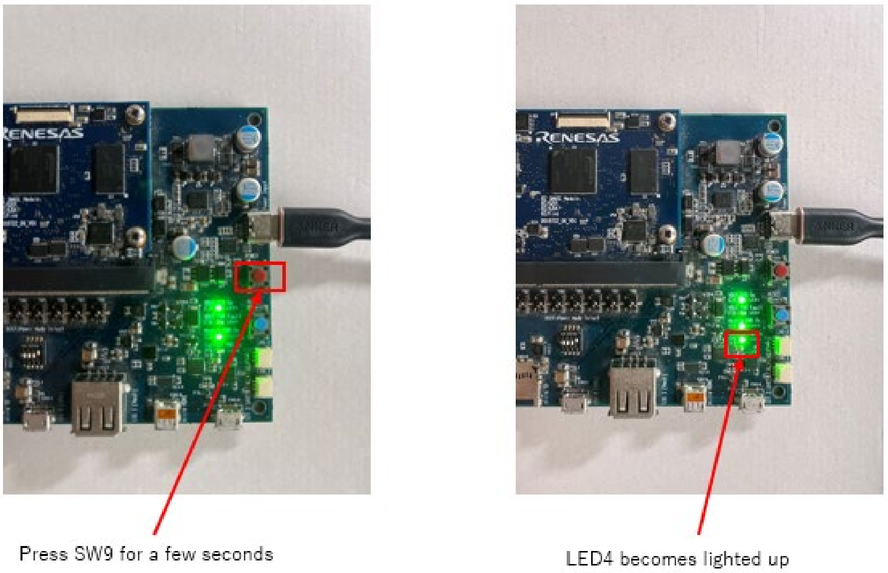


Figure 48. LED Status after Turning on EVK – SMARC EVK

EK-RZ/A3M

1. Connect DEBUG1(J10) connector of EK-RZ/A3M with PC.
   When powered, the white LED near the center of the board (the “dash” in the EK-RZ/A3M name) will illuminate.

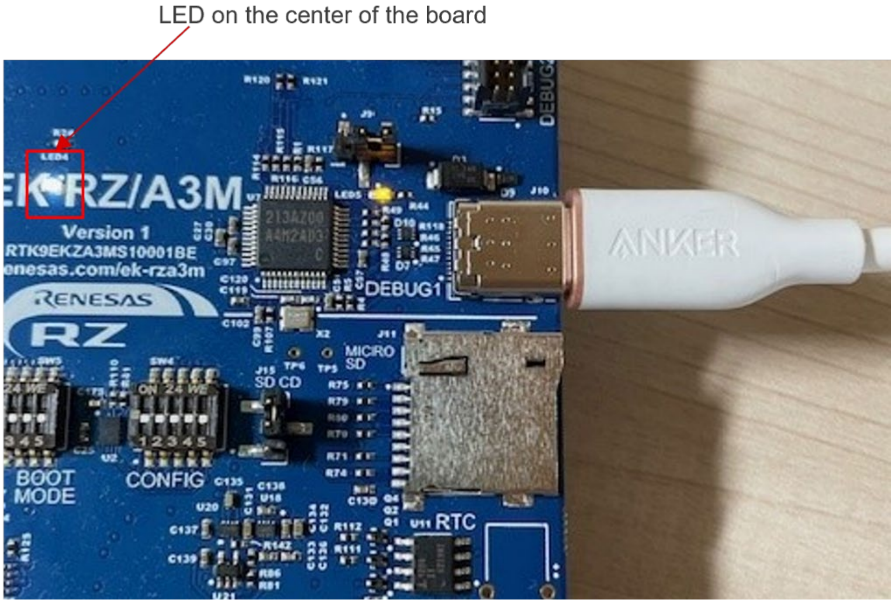
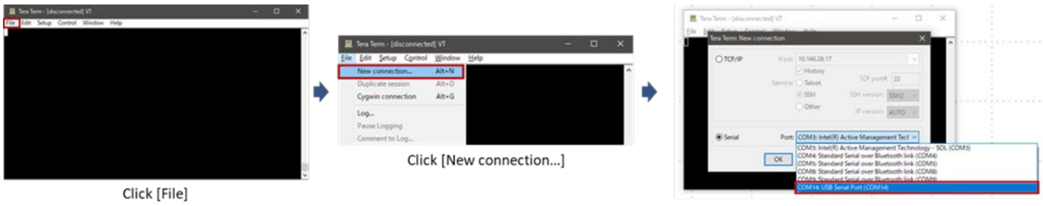


Figure 49. LED Status after Turning on – EK-RZ/A3M

3.2.5 How to check if your board is operational

This section describes how to check if your board is operational.

1. Connect the board to your development PC as described in 3.2.3.
2. Turn on the board as described in 3.2.4.
3. Launch Terminal Software (e.g., Tera Term).
4. Establish the connection between the board and the development PC as shown in the figure below:



Click [File]

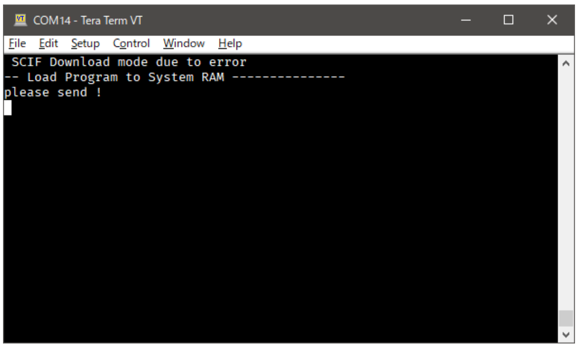
Click [New connection...]

Select "COM ##: USB Serial Port (COM ##) (note)

(note) The number of COM Port depends on your development PC.

Figure 50. Establishment of a connection between the Target board and the Development PC

5. You should see the following message on your Terminal Software. You can ignore the keyword "error" since the cause of the error is that nothing is programmed to Flash memory by default.



```
 SCIF Download mode due to error
-- Load Program to System RAM ---------------
please send !
```

Figure 51. Message on your terminal at the 1st power-up

4. Tutorial: Your First RZ MPU Project - Blinky

4.1 Tutorial Blinky

The goal of this tutorial is to quickly get acquainted with the Flexible Platform by moving through the steps of creating a simple application using e^2 studio and running that application on an RZ MPU board.

4.2 What Does Blinky Do?

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

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

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

Note:

THE SRMAC EVK board does not have any LEDs. Thus, the Blinky sample application used in this tutorial is designed to use the Pmod module described below, alternatively:

- Pmod LED (Four High-brightness LEDs): https://reference.digilentinc.com/pmod/pmodled/start

Figure 52. Connection Pmod LED module (410-076)

This module is not included on the SRMAC EVK board, and so, please prepare it beforehand.

In the case of EK-RZ/A3M, there are User LEDs on the board.

4.3 Creating a New Project for Blinky

The creation and configuration of an RZ C/C++ FSP Project is the first step in the creation of an application. The base RZ pack includes a pre-written Blinky example application.

Follow these steps to create an RZ MPU project:

1. In e^2 studio, click [File] > [New] > [C/C++ Project].

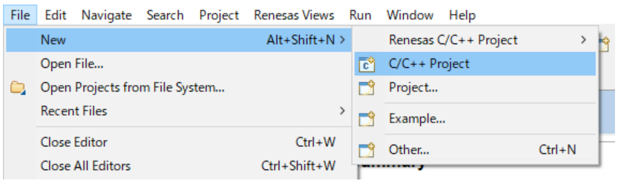
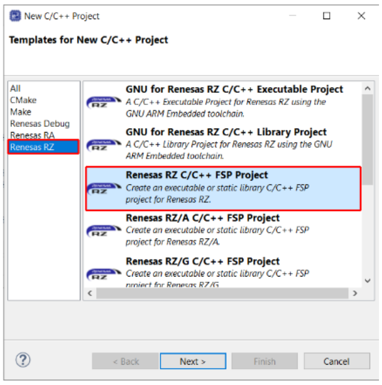


Figure 53. New C/C++ Project

2. Select [Renesas RZ] > [Renesas RZ C/C++ FSP Project] and Click Next.



GNU for Renesas RZ C/C++ Executable Project
A C/C++ Executable Project for Renesas RZ using the GNU ARM Embedded toolchain.

GNU for Renesas RZ C/C++ Library Project
A C/C++ Library Project for Renesas RZ using the GNU ARM Embedded toolchain.

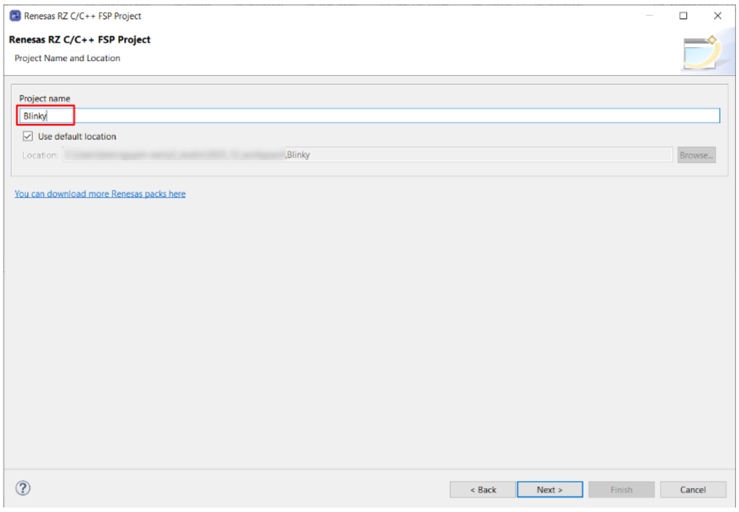
Renesas RZ C/C++ FSP Project
Create an executable or static library C/C++ FSP project for Renesas RZ.

Renesas RZ/A C/C++ FSP Project
Create an executable or static library C/C++ FSP project for Renesas RZ/A.

Renesas RZ/G C/C++ FSP Project
Create an executable or static library C/C++ FSP project for Renesas RZ/G.

Figure 54. Renesas RZ C/C++ FSP Project

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



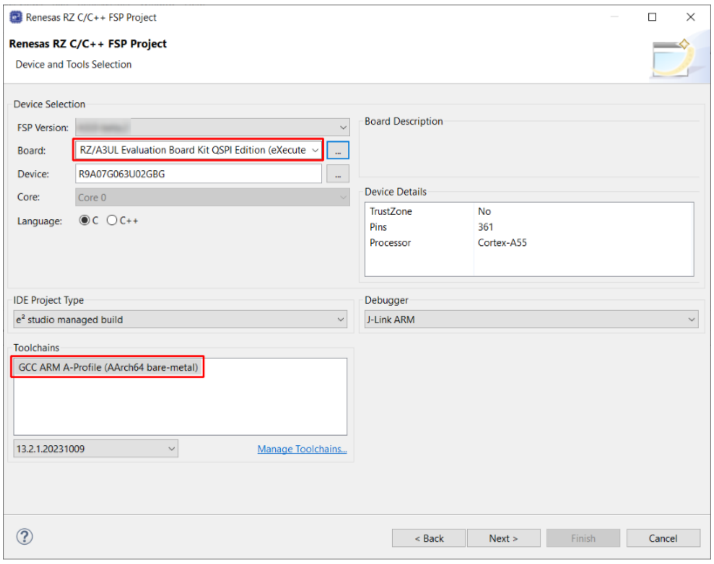
Project name: Blinky

☑ Use default location

Location: [redacted]Blinky

Figure 55. e² studio Project Configuration window (part 1)

5. Select the board support package corresponding to the package you would like to use, GCC ARM A-Profile (AArch64 bare-metal), and 13.2.1 from the Device Selection drop-down list, Toolchains and Version Selection drop-down list, respectively. Then, Click [Next].



FSP Version: [redacted]

Board: RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute

Device: R9A07G063U02GBG

Core: Core 0

Language: ● C ○ C++

TrustZone	No
Pins	361
Processor	Cortex-A55

IDE Project Type: e² studio managed build

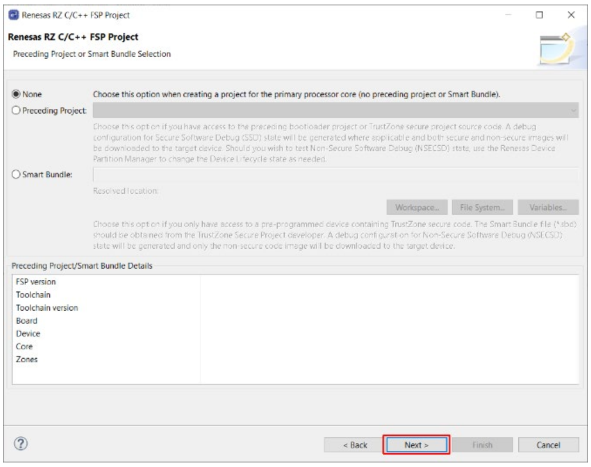
Debugger: J-Link ARM

Toolchains: GCC ARM A-Profile (AArch64 bare-metal)

13.2.1.20231009

Figure 56. e² studio Project Configuration window (part 2)

6. Choose this option when creating a project for the primary processor core. For a new project, simply click [Next].



Renesas RZ C/C++ FSP Project

Renesas RZ C/C++ FSP Project

Preceding Project or Smart Bundle Selection

● None — Choose this option when creating a project for the primary processor core (no preceding project or Smart Bundle).

○ Preceding Project:

Choose this option if you have access to the preceding bootloader project or TrustZone secure project source code. A debug configuration for Secure Software Debug (SSD) state will be generated where applicable and both secure and non-secure images will be downloaded to the target device. Should you wish to test Non-Secure Software Debug (NSECSD) state, use the Renesas Device Partition Manager to change the Device Lifecycle state as needed.

○ Smart Bundle:

Resolved location:

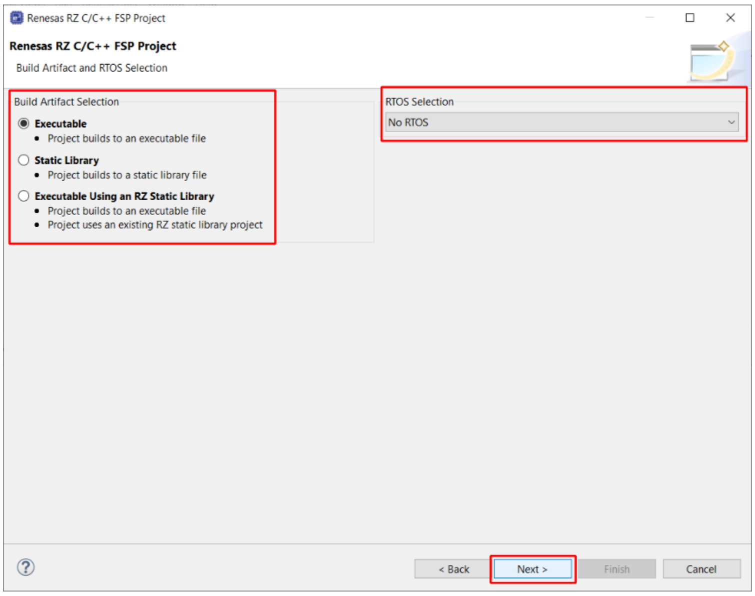
Choose this option if you only have access to a pre-programmed device containing TrustZone secure code. The Smart Bundle file (*.sbd) should be obtained from the TrustZone Secure Project developer. A debug configuration for Non-Secure Software Debug (NSECSD) state will be generated and only the non-secure code image will be downloaded to the target device.

Preceding Project/Smart Bundle Details

FSP version
Toolchain
Toolchain version
Board
Device
Core
Zones

Figure 57. e^2 studio Project Configuration window (part 3)

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



Renesas RZ C/C++ FSP Project

Renesas RZ C/C++ FSP Project

Build Artifact and RTOS Selection

Build Artifact Selection

● **Executable**
  - Project builds to an executable file

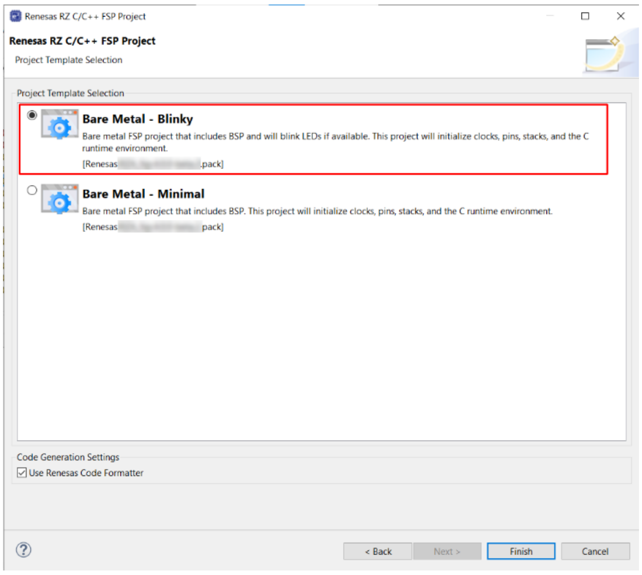
○ **Static Library**
  - Project builds to a static library file

○ **Executable Using an RZ Static Library**
  - Project builds to an executable file
  - Project uses an existing RZ static library project

RTOS Selection: No RTOS

Figure 58. e^2 studio Project Configuration window (part 4)

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



Renesas RZ C/C++ FSP Project

Renesas RZ C/C++ FSP Project

Project Template Selection

Project Template Selection

● **Bare Metal - Blinky**
Bare metal FSP project that includes BSP and will blink LEDs if available. This project will initialize clocks, pins, stacks, and the C runtime environment.
[Renesas[redacted].pack]

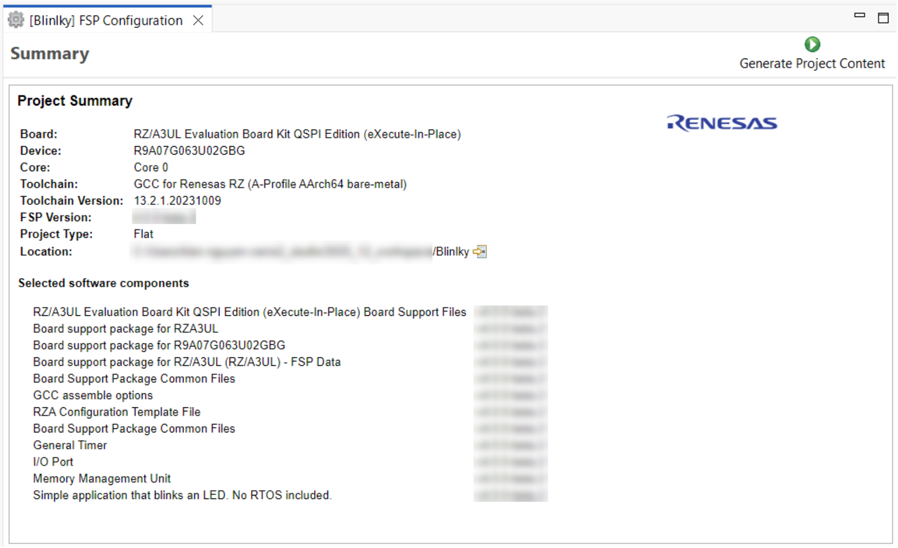
○ **Bare Metal - Minimal**
Bare metal FSP project that includes BSP. This project will initialize clocks, pins, stacks, and the C runtime environment.
[Renesas[redacted]pack]

Code Generation Settings

☑ Use Renesas Code Formatter

Figure 59. e^2 studio Project Configuration window (part 5)

Once the project has been created, the name of the project will show up in the **Project Explorer** window of e^2 studio. Now, click the Generate Project Content button in the top right corner of the Project Configuration window to generate your board-specific files.



[Blinlky] FSP Configuration

Summary

Project Summary

Board:	RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place)
Device:	R9A07G063U02GBG
Core:	Core 0
Toolchain:	GCC for Renesas RZ (A-Profile AArch64 bare-metal)
Toolchain Version:	13.2.1.20231009
FSP Version:	[redacted]
Project Type:	Flat
Location:	[redacted]/Blinlky

Selected software components

RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place) Board Support Files
Board support package for RZA3UL
Board support package for R9A07G063U02GBG
Board support package for RZ/A3UL (RZ/A3UL) - FSP Data
Board Support Package Common Files
GCC assemble options
RZA Configuration Template File
Board Support Package Common Files
General Timer
I/O Port
Memory Management Unit
Simple application that blinks an LED. No RTOS included.

[redacted]

Figure 60. 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 5.2.3. 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 the LED1 are configured by e^2 studio for the Blinky application. The pin configuration tab shows the pin configuration for the Blinky application (see 5.2.4.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_gtm
- r_ioport
- r_mmu

To see the configuration parameters for any of the components, check the Properties tab in the HAL window for the respective driver (see 5.2.9.Adding and Configuring HAL Drivers).

4.3.5 Where is main()?

The main function is located in <project>/rz_gen/main.c. It is one of the files that are generated during the project creation stage and only contains a call to hal_entry(). For more information on generated files, see 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 src/ folder.

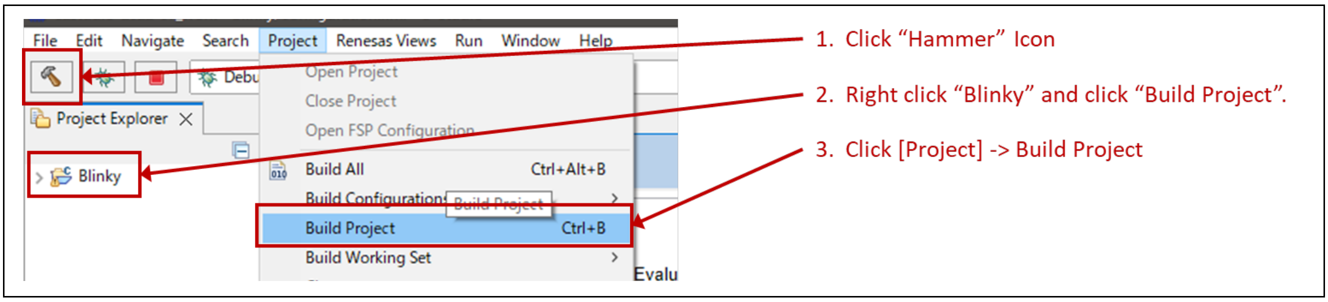
The application performs the following steps:

1. Get the LED information for the selected board from the bsp_leds_t structure.
2. Set the configuration of Timer (GTM) and the callback function that is called when interrupt is fired.
3. Define the output level HIGH for the GPIO pins controlling the LEDs for the selected board.
4. Toggle the LEDs by calling “R_BSP_PinWrite((bsp_io_port_pin_t) pin, pin_level)” for writing to the GPIO pin in the callback function of GTM that is called with the specified interval.

4.4 Build the Blinky Project

Highlight the new project in the Project Explorer window by clicking on it and building it. 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.



1. Click “Hammer” Icon
2. Right click “Blinky” and click “Build Project”.
3. Click [Project] -> Build Project

Figure 61. e^2 studio Project Explorer window

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

```
Building file: ../rz/fsp/src/bsp/mcu/all/bsp_security.c
Building file: ../rz/fsp/src/bsp/cmsis/Device/RENESAS/Source/startup.asm
Building file: ../rz/fsp/src/bsp/cmsis/Device/RENESAS/Source/default_handler.c
Building file: ../rz/fsp/src/bsp/cmsis/Device/RENESAS/Source/vector_table.asm
Building file: ../rz/dummy.c
Building file: ../rz/fsp/src/bsp/cmsis/Device/RENESAS/Source/system.c
Building file: ../rz/board/rza3ul_smarc_qspi_xip/board_init.c
Building file: ../rz/board/rza3ul_smarc_qspi_xip/board_leds.c
Building target: Blinlky.elf
aarch64-none-elf-size --format=berkeley "Blinlky.elf"
   text    data     bss     dec     hex filename
  14852    6224 2165120 2186196  215bd4 Blinlky.elf
aarch64-none-elf-objcopy -O srec "Blinlky.elf"  "Blinlky.srec"

13:34:30 Build Finished. 0 errors, 0 warnings. (took 3s.664ms)

**** Post-build Processing for project Blinlky ****
Post build script started
Post build script complete
Processing finished
```

Figure 62. e^2 studio Project Build console

4.5 Debug the Blinky Project

4.5.1 Debug prerequisites

To debug the project on a board, you need

- The board to be connected to e² studio
- The debugger needs to be configured to talk to the board
- The application is to be programmed for the microprocessor

Applications run from the internal ram or external ram of your microprocessor. To run or debug the application, the application must first be programmed to ram by a JTAG debugger. SMARC EVK board has a JTAG header and requires an external JTAG debugger to connect to the header.

In the case of choosing the “EK-RZ/A3M NAND Boot (Exec with DDR SDRAM)” for the board when creating the project, Flash Writer is required to download the program.
Please download the Flash Writer from:
https://www.renesas.com/products/rz-a3m

4.5.2 Debug steps for NOR Flash memory and Octa Flash memory environment

This chapter explains the debugging procedure in an environment for boards equipped with NOR Flash memory or Octa Flash memory.
If you chose the following board when creating a project in 4.3, please debug using this procedure.

- RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place)
- RZ/A3UL Evaluation Board Kit QSPI Edition (Exec with DDR SDRAM)
- RZ/A3UL Evaluation Board Kit Octal Edition (eXecute-In-Place)
- EK-RZ/A3M NOR Boot (Exec with DDR SDRAM)

1. Configure the debugger for your project by clicking [Run] > [Debugger Configurations...].

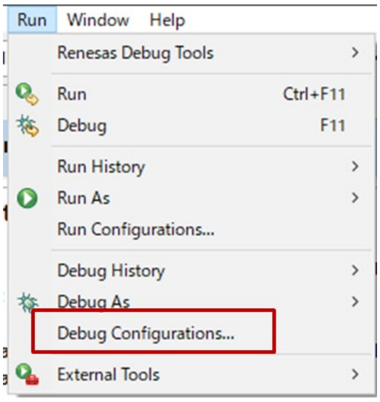


Figure 63. e² studio Debug icon

or by selecting the drop-down menu next to the bug icon and selecting [Debug Configurations...].

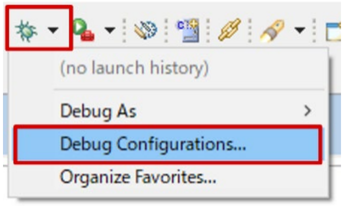
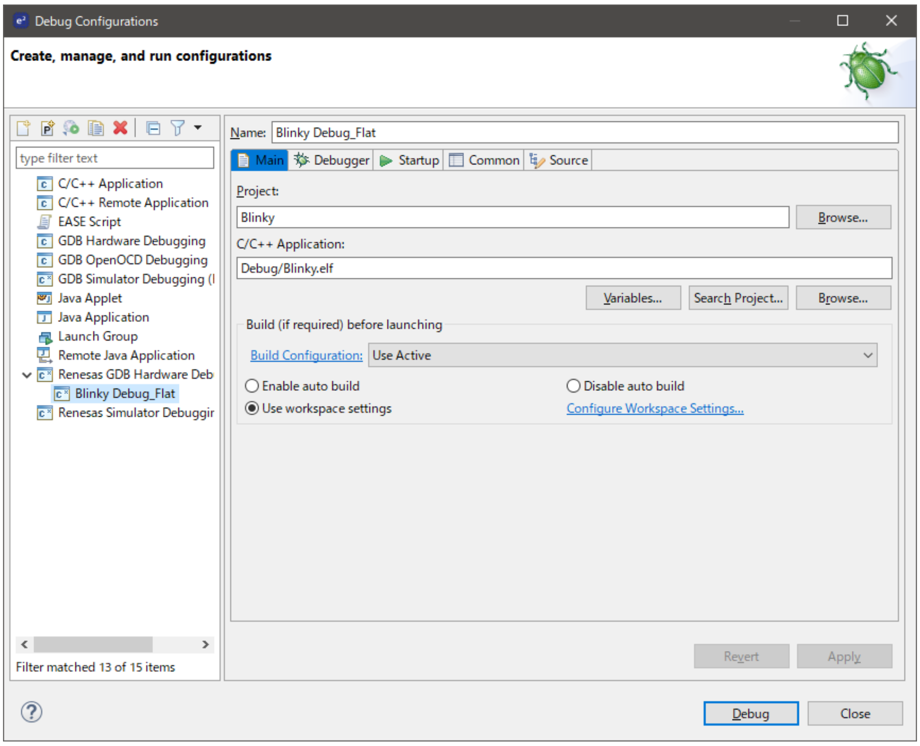


Figure 64. e² studio Debugger Configurations selection option

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** for your **Blinky** project should be displayed.



Debug Configurations

Create, manage, and run configurations

Name: Blinky Debug_Flat

Project: Blinky

C/C++ Application: Debug/Blinky.elf

Build (if required) before launching

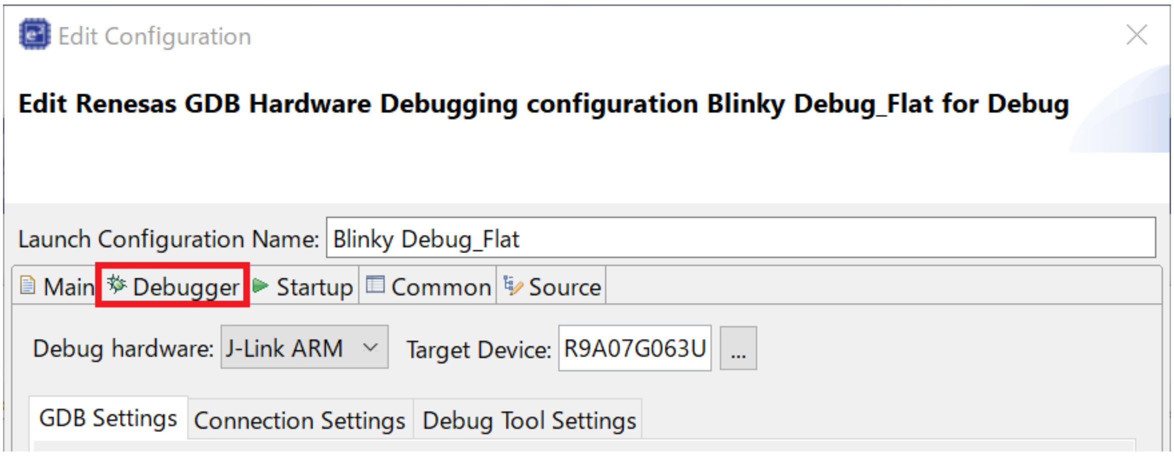
Build Configuration: Use Active

○ Enable auto build
○ Disable auto build
● Use workspace settings

Filter matched 13 of 15 items

Figure 65. e^2 studio Debugger Configurations window with Blinky project (1)

3. Select the debug configuration for the generated project and select the **Debugger** tab.



Edit Configuration

Edit Renesas GDB Hardware Debugging configuration Blinky Debug_Flat for Debug

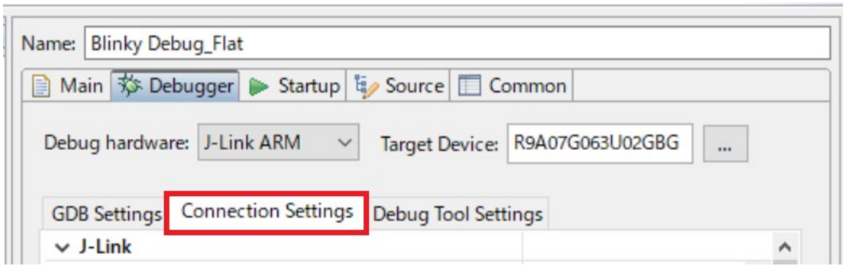
Launch Configuration Name: Blinky Debug_Flat

Debug hardware: J-Link ARM

Target Device: R9A07G063U

Figure 66. e^2 studio Debugger Configurations window with Blinky project (2)

4. Select the **Connection Settings tab** inside the **Debugger** tab.



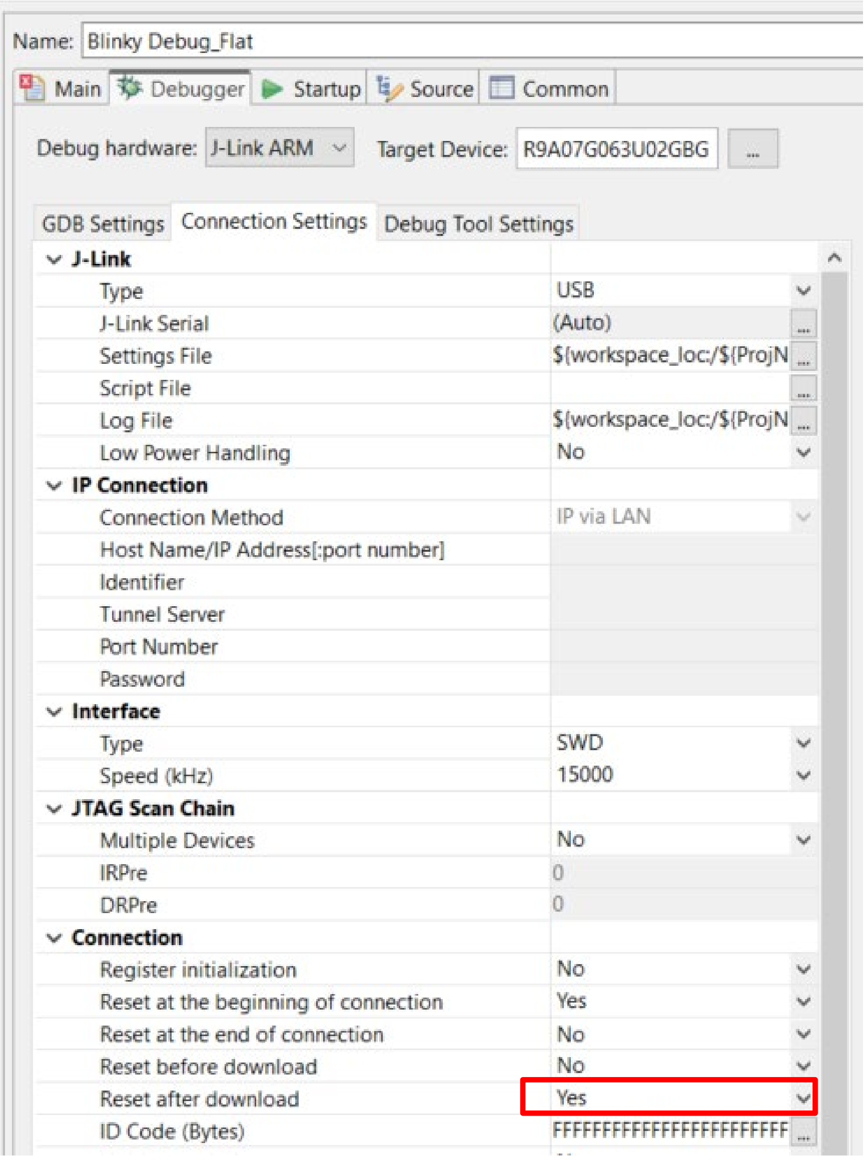
Name: Blinky Debug_Flat

Debug hardware: J-Link ARM

Target Device: R9A07G063U02GBG

Figure 67. e^2 studio Debugger Configurations window with Blinky project (3)

5. Change **Reset after download** to **Yes**.



Name: Blinky Debug_Flat

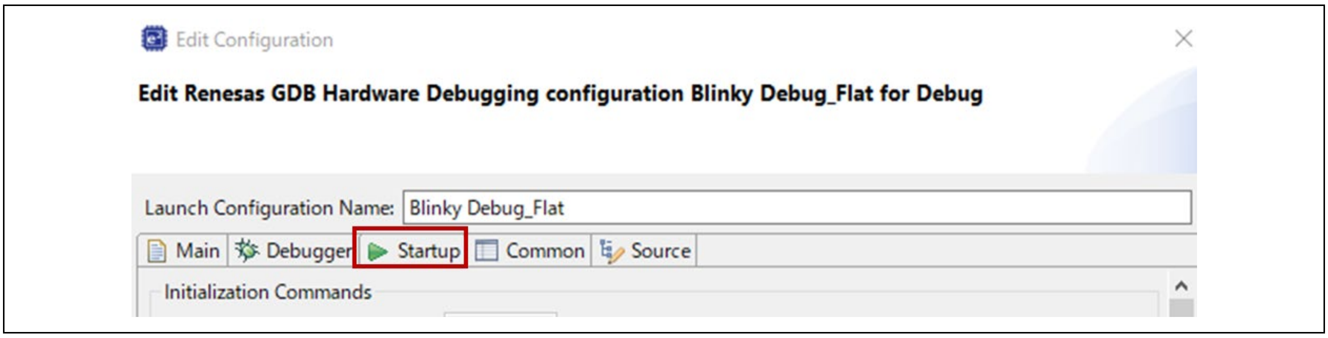
Debug hardware: J-Link ARM

Target Device: R9A07G063U02GBG

J-Link	
Type	USB
J-Link Serial	(Auto)
Settings File	${workspace_loc:/${ProjN
Script File	
Log File	${workspace_loc:/${ProjN
Low Power Handling	No
IP Connection	
Connection Method	IP via LAN
Host Name/IP Address[:port number]	
Identifier	
Tunnel Server	
Port Number	
Password	
Interface	
Type	SWD
Speed (kHz)	15000
JTAG Scan Chain	
Multiple Devices	No
IRPre	0
DRPre	0
Connection	
Register initialization	No
Reset at the beginning of connection	Yes
Reset at the end of connection	No
Reset before download	No
Reset after download	Yes
ID Code (Bytes)	FFFFFFFFFFFFFFFFFFFFFFFF

Figure 68. e^2 studio Debugger Configurations window with Blinky project (4)

6. Select the debug configuration for the generated project and select the **Startup** tab.



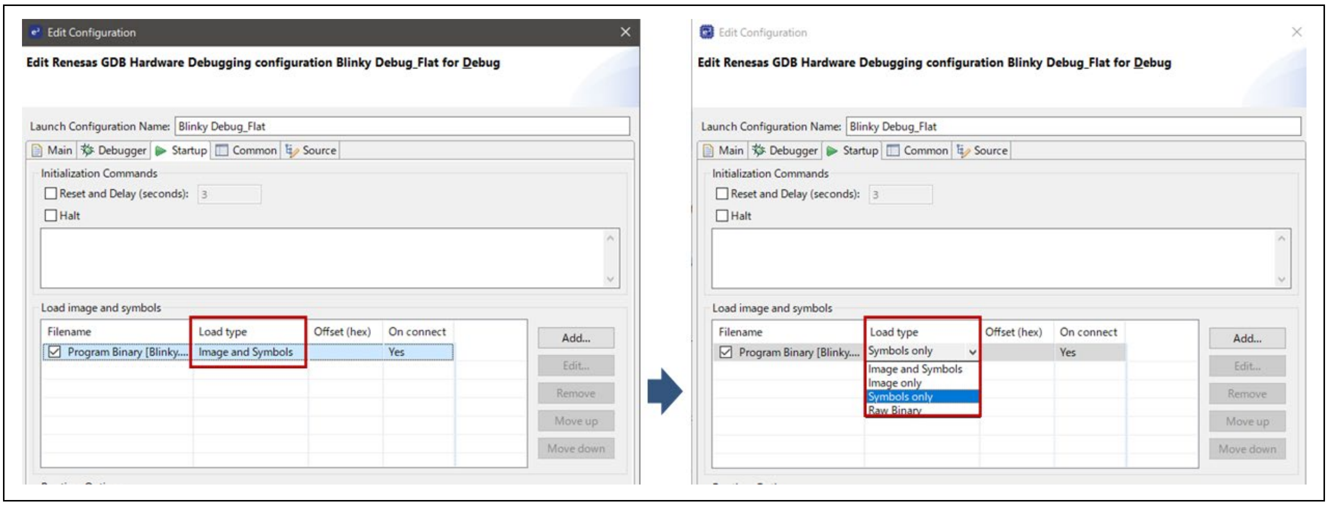
Edit Configuration

Edit Renesas GDB Hardware Debugging configuration Blinky Debug_Flat for Debug

Launch Configuration Name: Blinky Debug_Flat

Figure 69. e^2 studio Debugger Configurations window with Blinky project (5)

7. Be sure to change the setting in the **Load type** field of **Program Binary [Blinky...** raw from **Image and Symbols** to **Symbols only**.



Edit Renesas GDB Hardware Debugging configuration Blinky Debug_Flat for Debug

Launch Configuration Name: Blinky Debug_Flat

Initialization Commands

☐ Reset and Delay (seconds): 3

☐ Halt

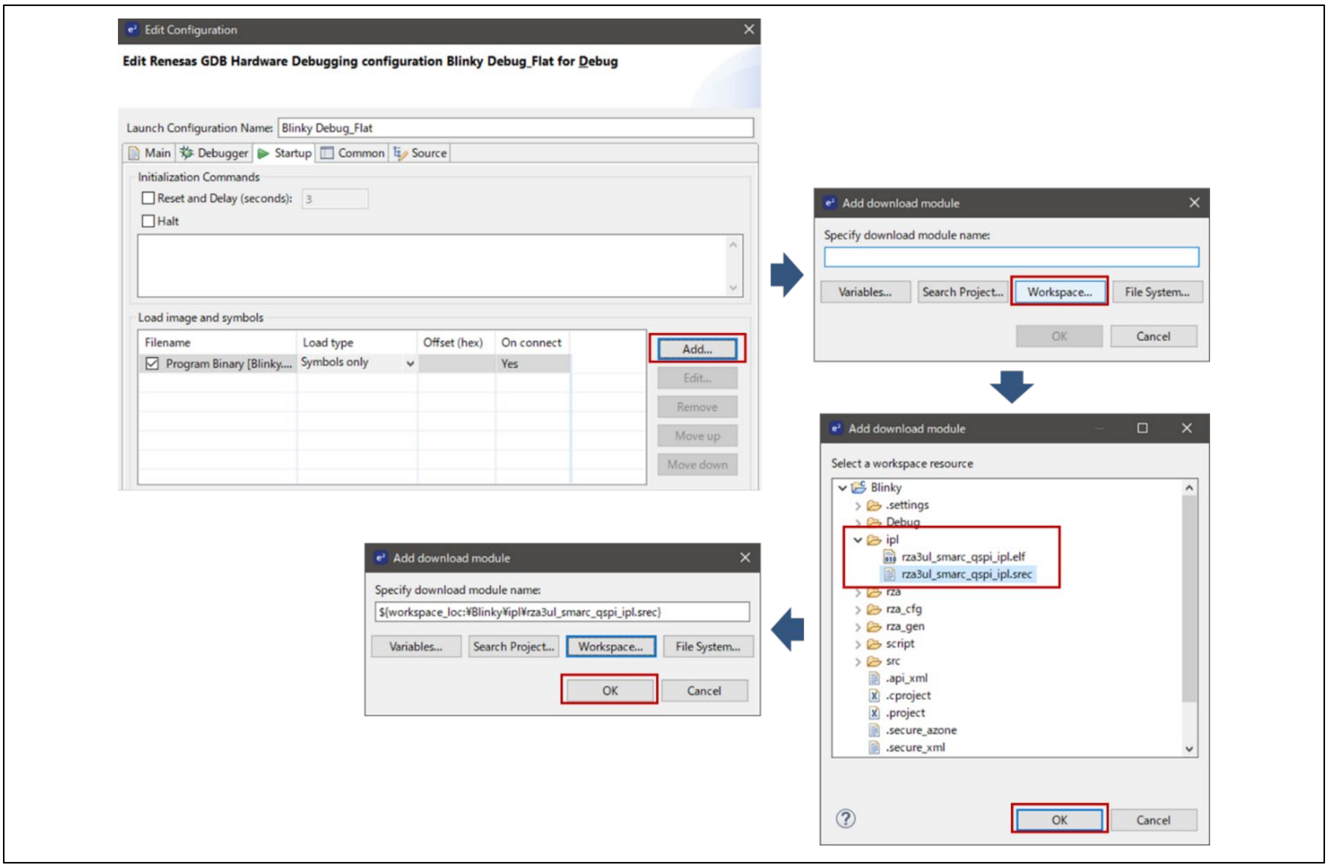
Load image and symbols

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [Blinky...	Image and Symbols		Yes

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [Blinky...	Symbols only		Yes

Figure 70. e^2 studio Debugger Configurations window with Blinky project (6)

8. Click on **Add...** to launch the **Add download module** window. Then click **Workspace...**, choose **rza3ul_smarc_qspi_ipl.srec(*)** as module to be downloaded, and finally click on **OK**.
   * If you have selected "RZ/A3UL Evaluation Board Kit OCTAL Edition (eXecute-In-Place)" in board selection, use "**rza3ul_smarc_octal_ipl.srec**" instead.



Launch Configuration Name: Blinky Debug_Flat

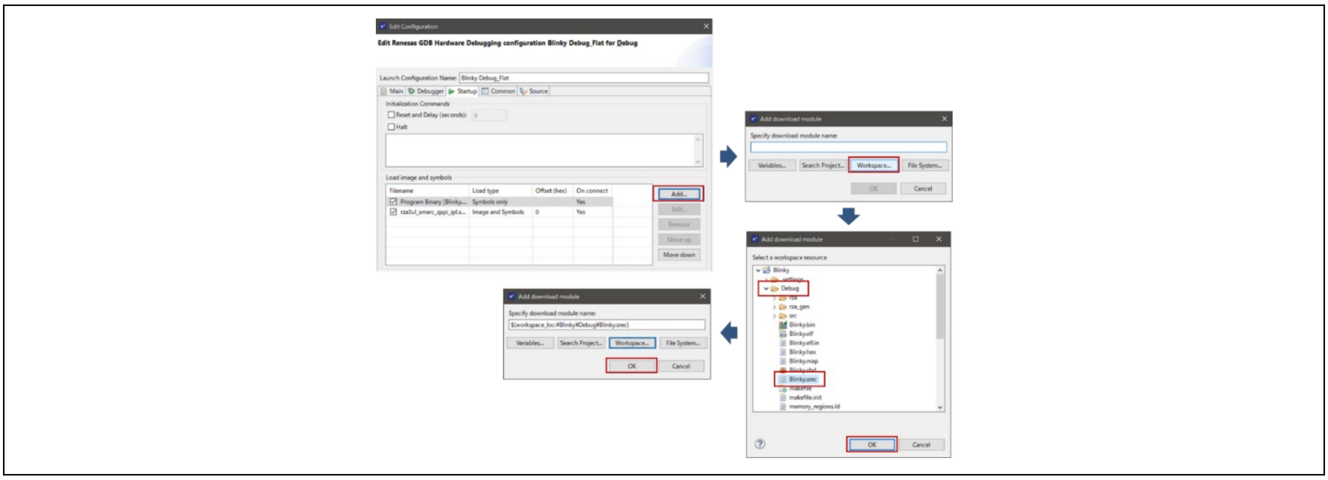
☐ Reset and Delay (seconds): 3

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [Blinky...	Symbols only		Yes

Specify download module name: ${workspace_loc:¥Blinky¥ipl¥rza3ul_smarc_qspi_ipl.srec}

Figure 71. e^2 studio Debugger Configurations window with Blinky project (7)

9. Again, click on **Add...** to launch the **Add download module** window. Then click **Workspace...**, choose **BLINKY.srec** as the module to be downloaded, and finally click on **OK**.



Launch Configuration Name: Blinky Debug_Flat

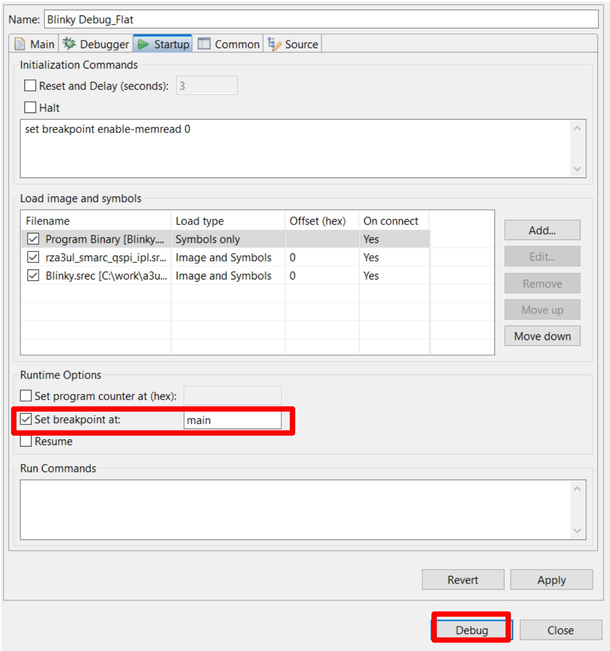
☐ Reset and Delay (seconds): 3

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [Blinky...	Symbols only		Yes
☑ rza3ul_smarc_qspi_ipl.s...	Image and Symbols	0	Yes

Specify download module name: ${workspace_loc:¥Blinky¥Debug¥Blinky.srec}

Figure 72. e^2 studio Debugger Configurations window with Blinky project (8)

10. Check the **Set breakpoint at:** check box, then click the **Debug** button.



Name: Blinky Debug_Flat

Initialization Commands

☐ Reset and Delay (seconds): 3

☐ Halt

```
set breakpoint enable-memread 0
```

Load image and symbols

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [Blinky...	Symbols only		Yes
☑ rza3ul_smarc_qspi_ipl.sr...	Image and Symbols	0	Yes
☑ Blinky.srec [C:\work\a3u...	Image and Symbols	0	Yes

Runtime Options

☐ Set program counter at (hex):

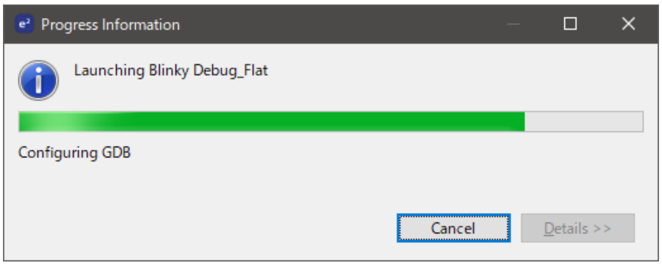
☑ Set breakpoint at: main

☐ Resume

Run Commands

Figure 73. e² studio Debugger Configurations window with Blinky project (10)

11. Debug session has now started.

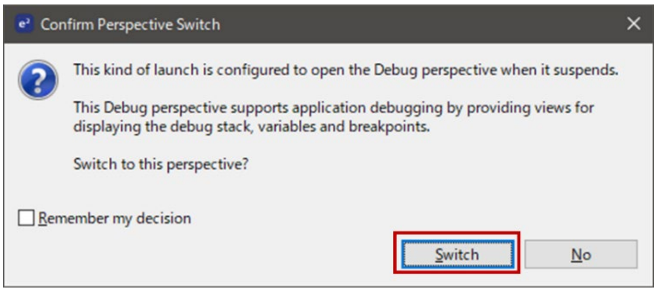


Launching Blinky Debug_Flat

Configuring GDB

Figure 74. e² studio Debugger Configurations window with Blinky project (11)

12. If you see the following window, please click **Switch** to continue.



This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective supports application debugging by providing views for displaying the debug stack, variables and breakpoints.

Switch to this perspective?

☐ Remember my decision

Figure 75. e² studio Debugger Configurations window with Blinky project (12)

4.5.3 Debug steps for NAND Flash memory

This chapter explains the debugging procedure in an environment for boards equipped with NAND Flash.
If you choose the following board when creating a project in 4.3, please debug using this procedure.

- EK-RZ/A3M NAND Boot (Exec with DDR SDRAM)

There are two boot modes for IPL in NAND Flash memory environments: Only init device mode and Normal boot mode.

Only Init device mode:

In this mode, the IPL only initializes the device and does not load the application.
The application is downloaded directly to DDR SDRAM using e^2 studio.

It is recommended that you use this mode when debugging.
NAND Flash memory tends to deteriorate if it is written frequently.
However, in Normal boot mode, the NAND Flash memory must be rewritten every time the application is changed.

Normal boot mode:

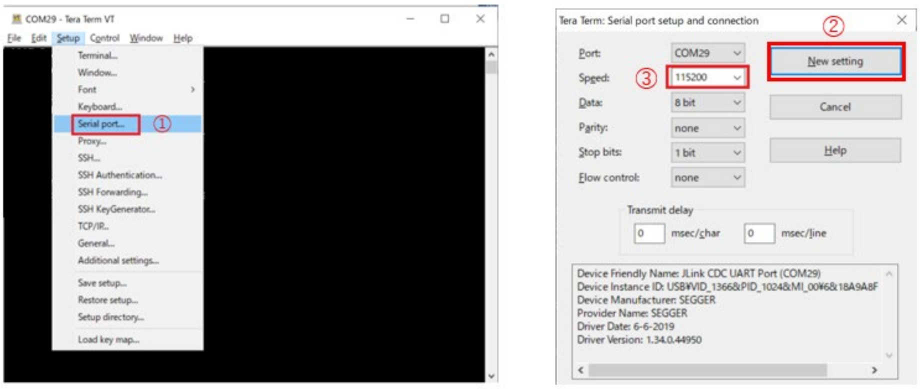
In this mode, the IPL initializes the device and loads the application program from the Serial NAND Flash to the internal RAM and jumps to the application.

4.5.3.1 Debug steps for Only init device mode

To debug in Init Device Only mode, follow the steps below.

1. To write the IPL to NAND Flash, you need the transmission file **rza3m_ek_nand_ipl_debug.srec**. When operating in Only Init Device mode, you do not need the transmission file **(Project Name).srec**. Please refer to Chapter 3.3 of “RZ/A3M How to use Flash Writer to boot from Serial NAND Flash” (R01AN8012EJ0101).

2. Click **Setup -> Serial port -> New setting**, and change the Speed to **[115200]**.

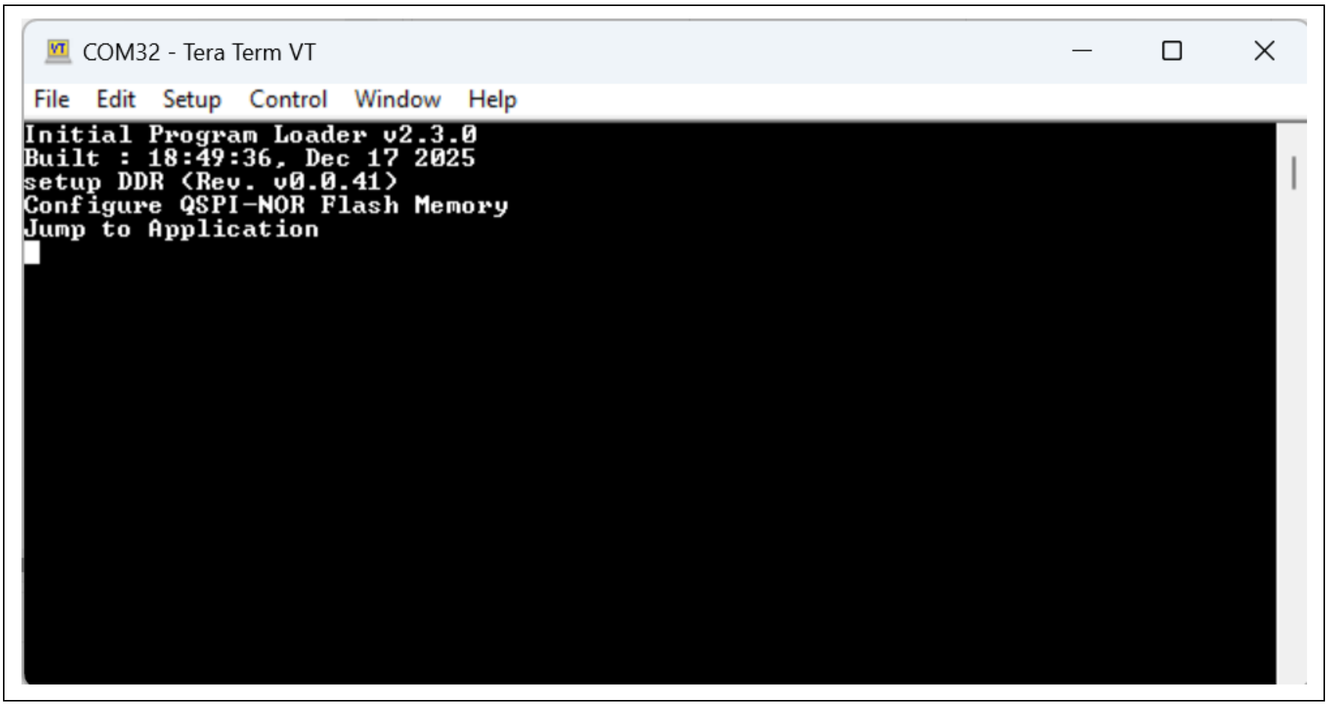


Port: COM29
Speed: 115200
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none
Transmit delay
0 msec/char 0 msec/line

Device Friendly Name: JLink CDC UART Port (COM29)
Device Instance ID: USB¥VID_1366&PID_1024&MI_00¥6&18A9A8F
Device Manufacturer: SEGGER
Provider Name: SEGGER
Driver Date: 6-6-2019
Driver Version: 1.34.0.44950

Figure 76. Select speed

3. Set the EK-RZ/A3M main board operating mode to Boot Mode7 (3.3-V Serial NAND Flash Memory) (SW4-5 ON, SW5-1 ON, SW5-2 ON) and reset the power. At this time, IPL startup messages are output to the serial terminal.



```
Initial Program Loader v2.3.0
Built : 18:49:36, Dec 17 2025
setup DDR (Rev. v0.0.41)
Configure QSPI-NOR Flash Memory
Jump to Application
```

Figure 77. IPL startup messages

4. If you haven't started e^2 studio yet, start it.
Then configure the debugger for your project by clicking **[Run]** > **[Debugger Configurations...]**.

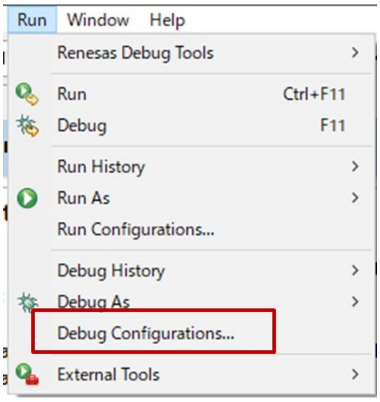


Figure 78. e^2 studio Debug icon

or by selecting the drop-down menu next to the bug icon and selecting **[Debug Configurations...]**.

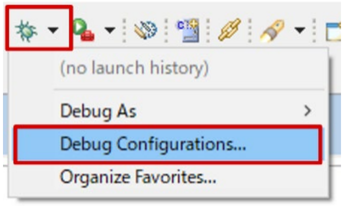
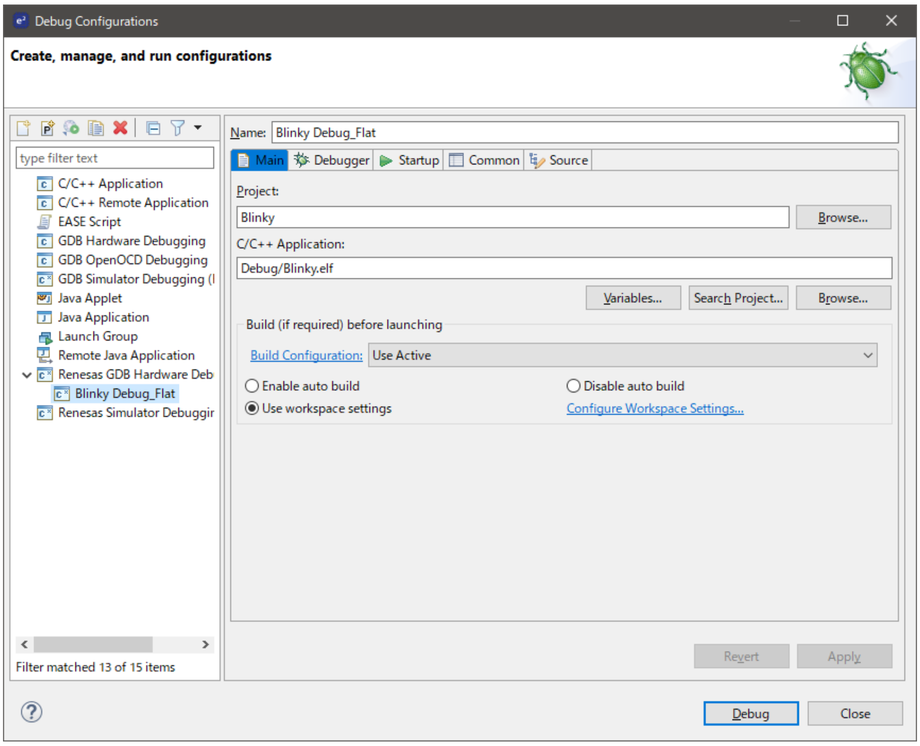


Figure 79. e^2 studio Debugger Configurations selection option

5. 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** for your **Blinky** project should be displayed.



Debug Configurations

Create, manage, and run configurations

Name: Blinky Debug_Flat

Project: Blinky

C/C++ Application: Debug/Blinky.elf

Build (if required) before launching

Build Configuration: Use Active

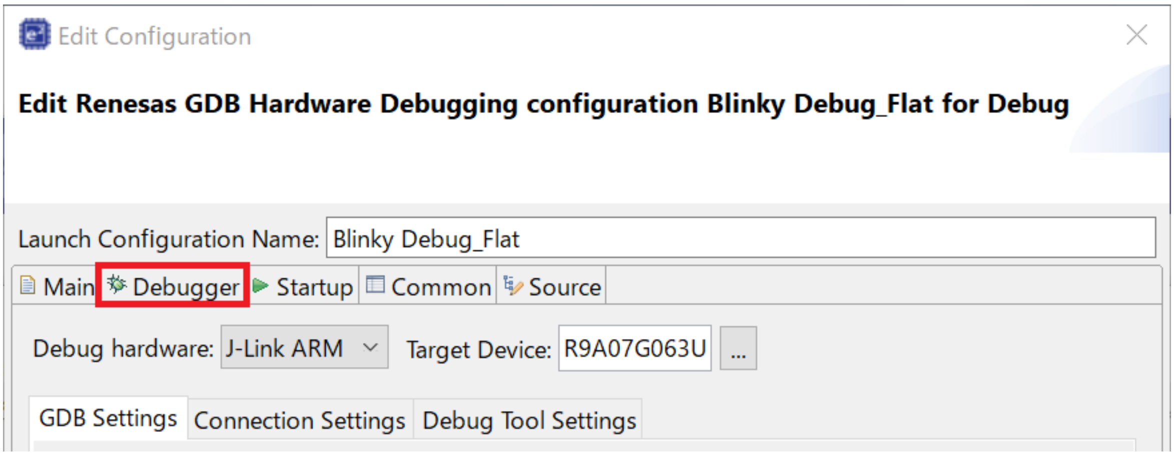
○ Enable auto build ○ Disable auto build

● Use workspace settings

Filter matched 13 of 15 items

Figure 80. e^2 studio Debugger Configurations window with Blinky project (1)

6. Select the debug configuration for the generated project and select the **Debugger** tab.



Edit Configuration

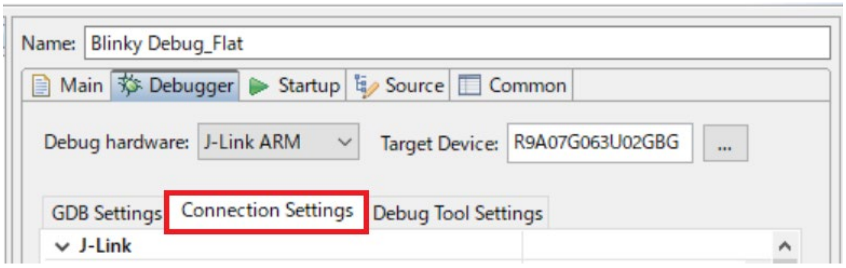
Edit Renesas GDB Hardware Debugging configuration Blinky Debug_Flat for Debug

Launch Configuration Name: Blinky Debug_Flat

Debug hardware: J-Link ARM Target Device: R9A07G063U

Figure 81. e^2 studio Debugger Configurations window with Blinky project (2)

7. Select the **Connection Settings tab** inside the **Debugger** tab.



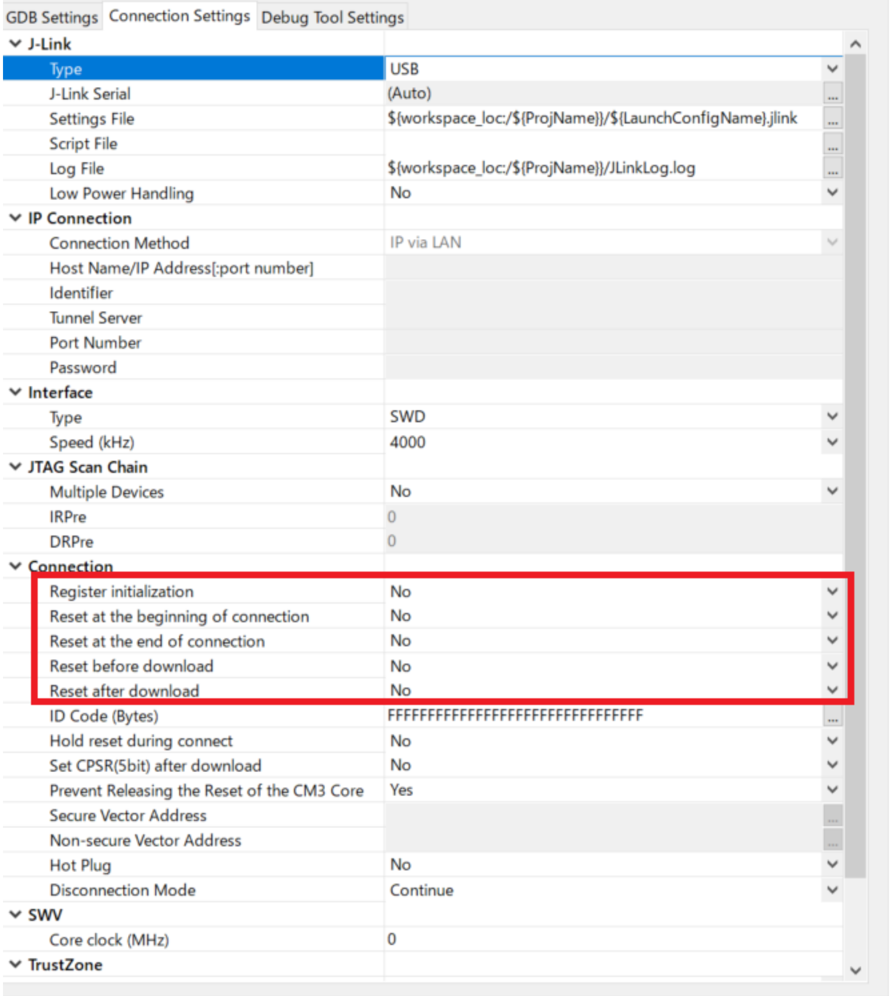
Name: Blinky Debug_Flat

Debug hardware: J-Link ARM

Target Device: R9A07G063U02GBG

Figure 82. e^2 studio Debugger Configurations window with Blinky project (3)

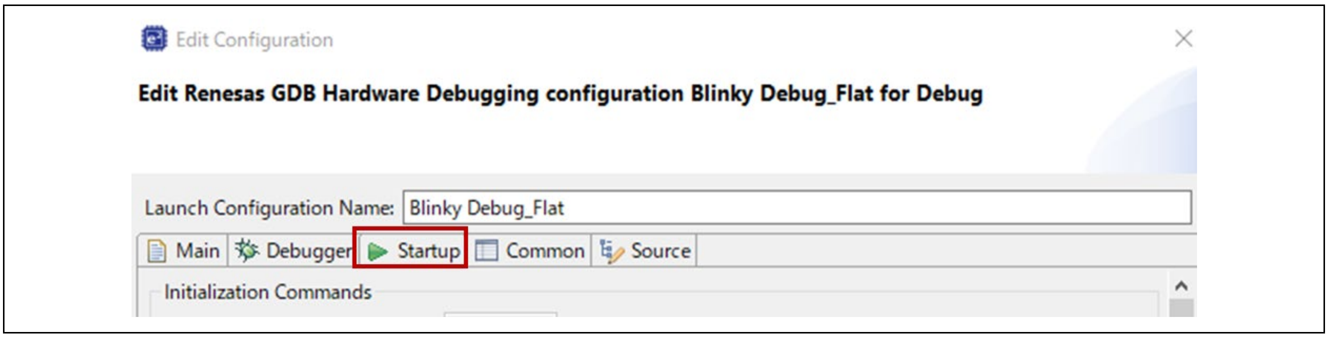
8. Change all **Reset settings** to **No**.



J-Link	
Type	USB
J-Link Serial	(Auto)
Settings File	${workspace_loc:/${ProjName}}/${LaunchConfigName}.jlink
Script File	
Log File	${workspace_loc:/${ProjName}}/JLinkLog.log
Low Power Handling	No
IP Connection	
Connection Method	IP via LAN
Host Name/IP Address[:port number]	
Identifier	
Tunnel Server	
Port Number	
Password	
Interface	
Type	SWD
Speed (kHz)	4000
JTAG Scan Chain	
Multiple Devices	No
IRPre	0
DRPre	0
Connection	
Register initialization	No
Reset at the beginning of connection	No
Reset at the end of connection	No
Reset before download	No
Reset after download	No
ID Code (Bytes)	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Hold reset during connect	No
Set CPSR(5bit) after download	No
Prevent Releasing the Reset of the CM3 Core	Yes
Secure Vector Address	
Non-secure Vector Address	
Hot Plug	No
Disconnection Mode	Continue
SWV	
Core clock (MHz)	0
TrustZone	

Figure 83. e^2 studio Debugger Configurations window with Blinky project (4)

9. Select the debug configuration for the generated project and select the **Startup** tab.

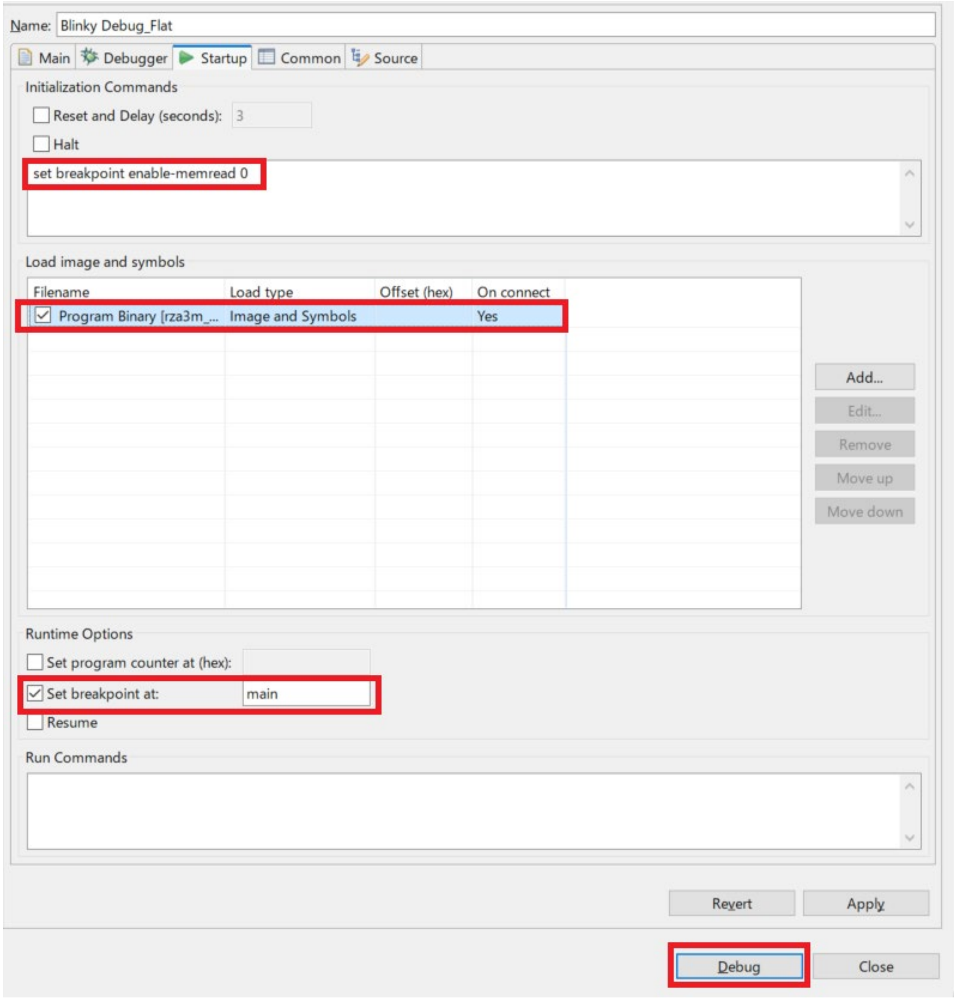


Edit Renesas GDB Hardware Debugging configuration Blinky Debug_Flat for Debug

Launch Configuration Name: Blinky Debug_Flat

Figure 84. e^2 studio Debugger Configurations window with Blinky project (5)

10. Make sure that only **Program Binary** is checked and that the Load type is set to **Image and Symbol**. Moreover, check the **Set breakpoint at:** check box, and add the “set breakpoint enable-memread 0” to the Initialization Commands.
Then click [Debug] to load the application to DDR SDRAM and launch it.



Name: Blinky Debug_Flat

☐ Reset and Delay (seconds): 3
☐ Halt

```
set breakpoint enable-memread 0
```

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [rza3m_...	Image and Symbols		Yes

☐ Set program counter at (hex):
☑ Set breakpoint at: main
☐ Resume

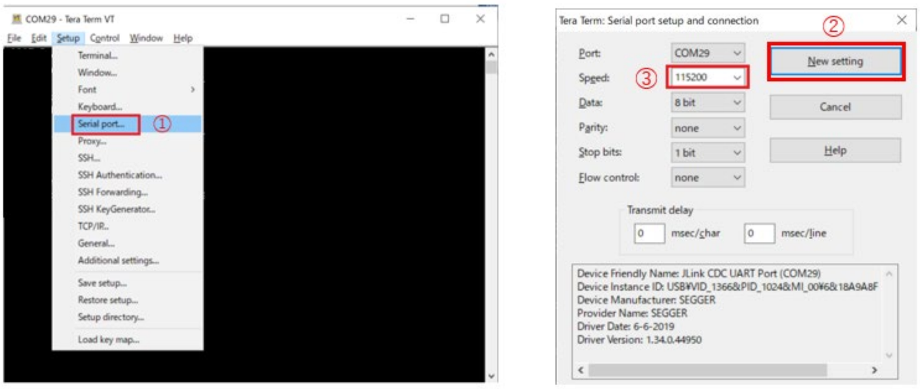
Figure 85. e^2 studio Debugger Configurations window with Blinky project (6)

4.5.3.2 Debug steps for Normal boot mode

To debug in Normal boot mode, follow the steps below.

1. To write IPL and application to NAND Flash, you need the transmission file **rza3m_ek_nand_ipl.srec** and **(Project Name).srec**. Please refer to Chapter 3.3 of “RZ/A3M How to use Flash Writer to boot from Serial NAND Flash” (R01AN8012EJ0101).

2. Click **Setup -> Serial port -> New setting**, and change the Speed to **[115200]**.

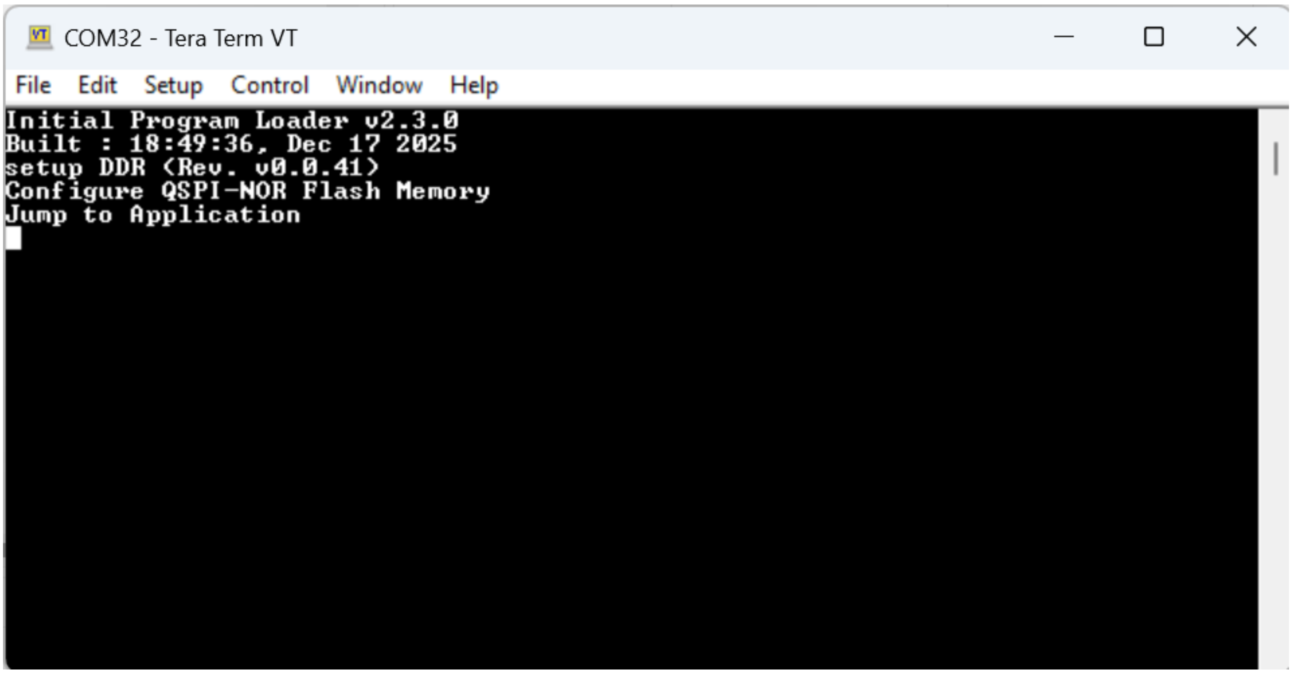


Port: COM29
Speed: 115200
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none
Transmit delay
0 msec/char 0 msec/line

Device Friendly Name: JLink CDC UART Port (COM29)
Device Instance ID: USB¥VID_1366&PID_1024&MI_00¥6&18A9A8F
Device Manufacturer: SEGGER
Provider Name: SEGGER
Driver Date: 6-6-2019
Driver Version: 1.34.0.44950

Figure 86. Select speed

3. Set the EK-RZ/A3M main board operating mode to Boot Mode7 (3.3-V Serial NAND Flash Memory) (SW4-5 ON, SW5-1 ON, SW5-2 ON) and reset the power. At this time, IPL startup messages are output to the serial terminal.



```
Initial Program Loader v2.3.0
Built : 18:49:36, Dec 17 2025
setup DDR (Rev. v0.0.41)
Configure QSPI-NOR Flash Memory
Jump to Application
```

Figure 87. IPL startup messages

4. If you haven't started e^2 studio yet, start it.
   Then Configure the debugger for your project by clicking **[Run]** > **[Debugger Configurations...]**.

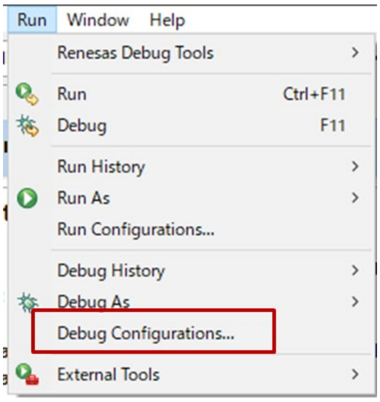


Figure 88. e^2 studio Debug icon

or by selecting the drop-down menu next to the bug icon and selecting **[Debug Configurations...]**.

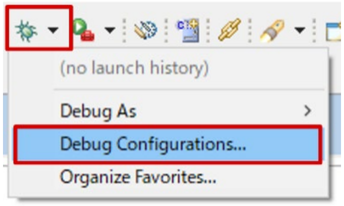
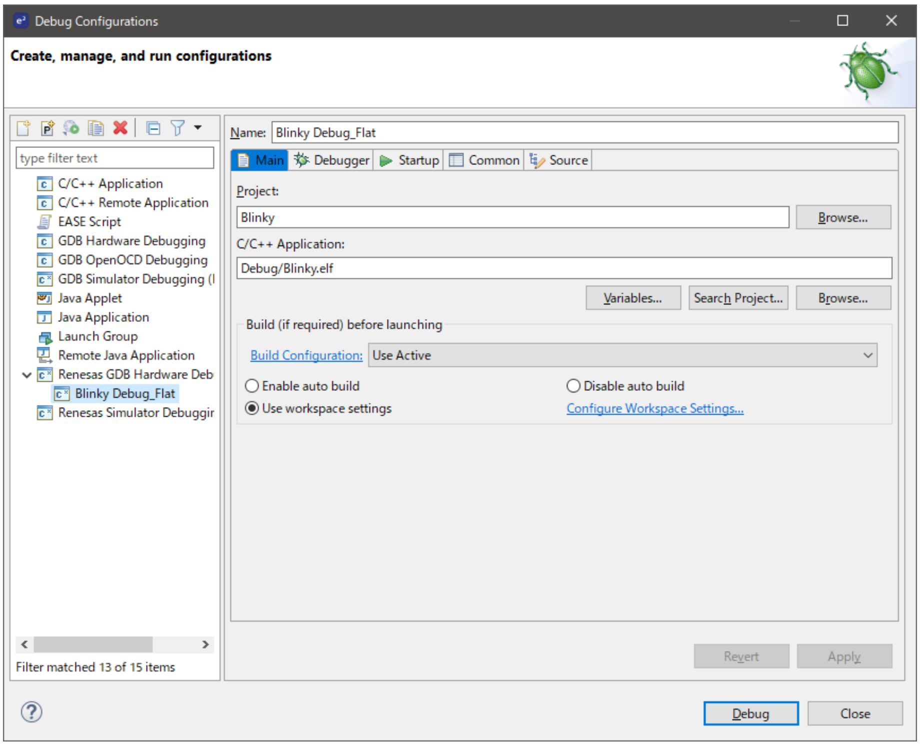


Figure 89. e^2 studio Debugger Configurations selection option

5. 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** for your **Blinky** project should be displayed.



Create, manage, and run configurations

Name: Blinky Debug_Flat

Project: Blinky

C/C++ Application: Debug/Blinky.elf

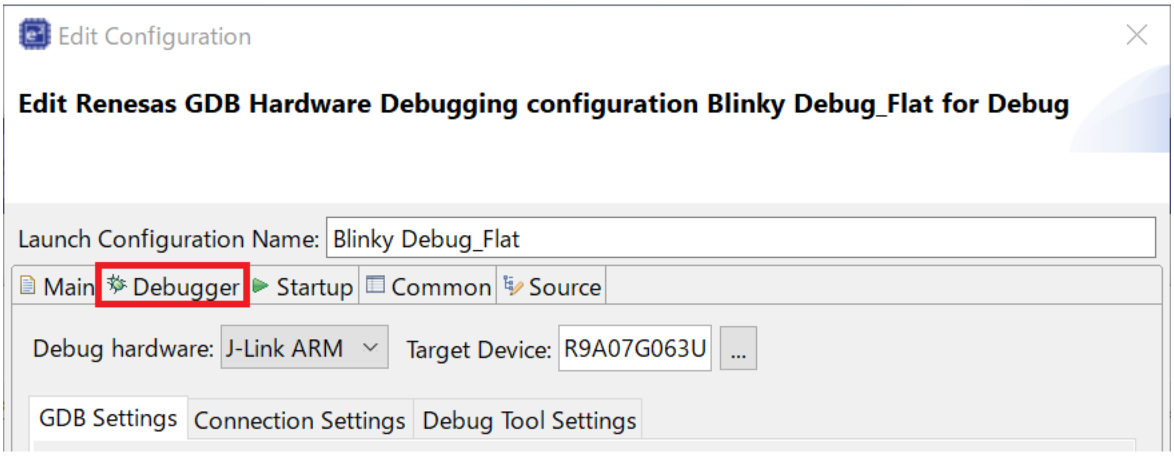
Build (if required) before launching

Build Configuration: Use Active

○ Enable auto build
○ Disable auto build
● Use workspace settings

Figure 90. e^2 studio Debugger Configurations window with Blinky project (1)

6. Select the debug configuration for the generated project and select the **Debugger** tab.



Edit Renesas GDB Hardware Debugging configuration Blinky Debug_Flat for Debug

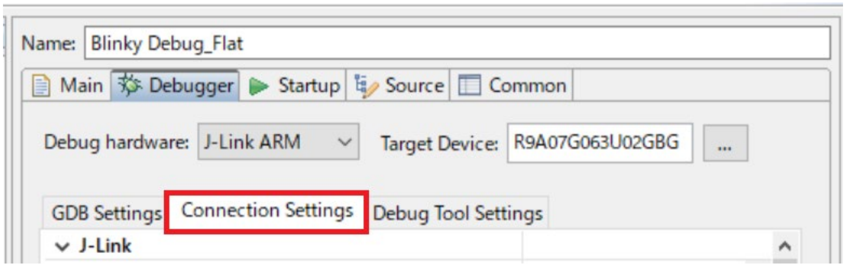
Launch Configuration Name: Blinky Debug_Flat

Debug hardware: J-Link ARM

Target Device: R9A07G063U

Figure 91. e^2 studio Debugger Configurations window with Blinky project (2)

7. Select the **Connection Settings tab** inside the **Debugger** tab.



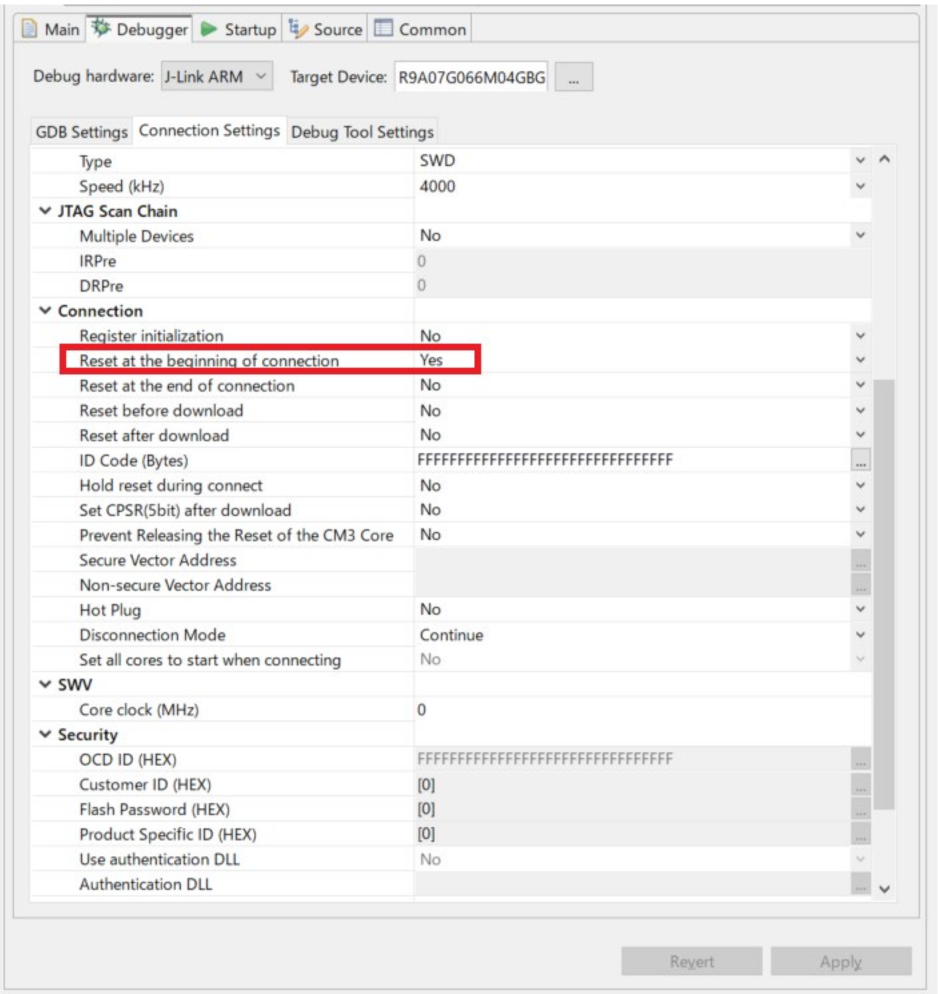
Name: Blinky Debug_Flat

Debug hardware: J-Link ARM

Target Device: R9A07G063U02GBG

Figure 92. e^2 studio Debugger Configurations window with Blinky project (3)

8. Change **Reset at the beginning of connection** to **Yes**.



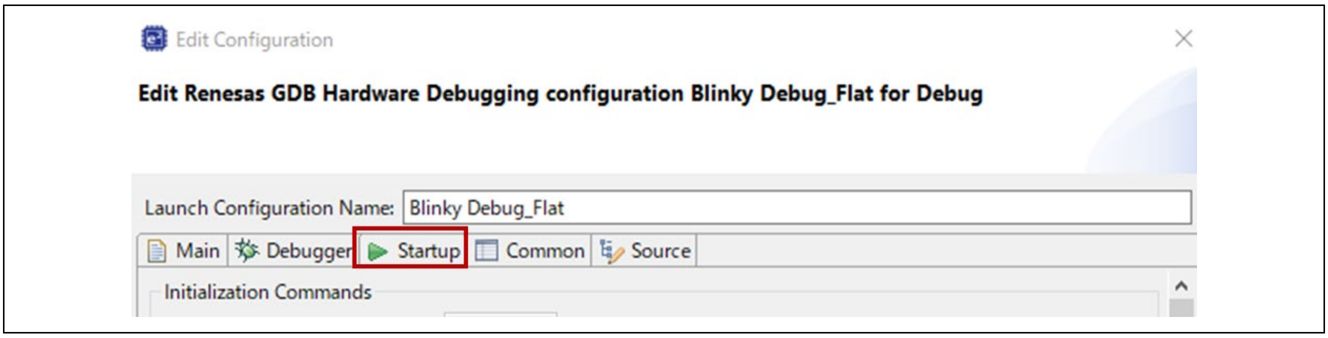
Debug hardware: J-Link ARM

Target Device: R9A07G066M04GBG

Type	SWD
Speed (kHz)	4000
JTAG Scan Chain	
Multiple Devices	No
IRPre	0
DRPre	0
Connection	
Register initialization	No
Reset at the beginning of connection	Yes
Reset at the end of connection	No
Reset before download	No
Reset after download	No
ID Code (Bytes)	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Hold reset during connect	No
Set CPSR(5bit) after download	No
Prevent Releasing the Reset of the CM3 Core	No
Secure Vector Address	
Non-secure Vector Address	
Hot Plug	No
Disconnection Mode	Continue
Set all cores to start when connecting	No
SWV	
Core clock (MHz)	0
Security	
OCD ID (HEX)	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Customer ID (HEX)	[0]
Flash Password (HEX)	[0]
Product Specific ID (HEX)	[0]
Use authentication DLL	No
Authentication DLL	

Figure 93. e^2 studio Debugger Configurations window with Blinky project (4)

9. Select the debug configuration for the generated project and select the **Startup** tab.



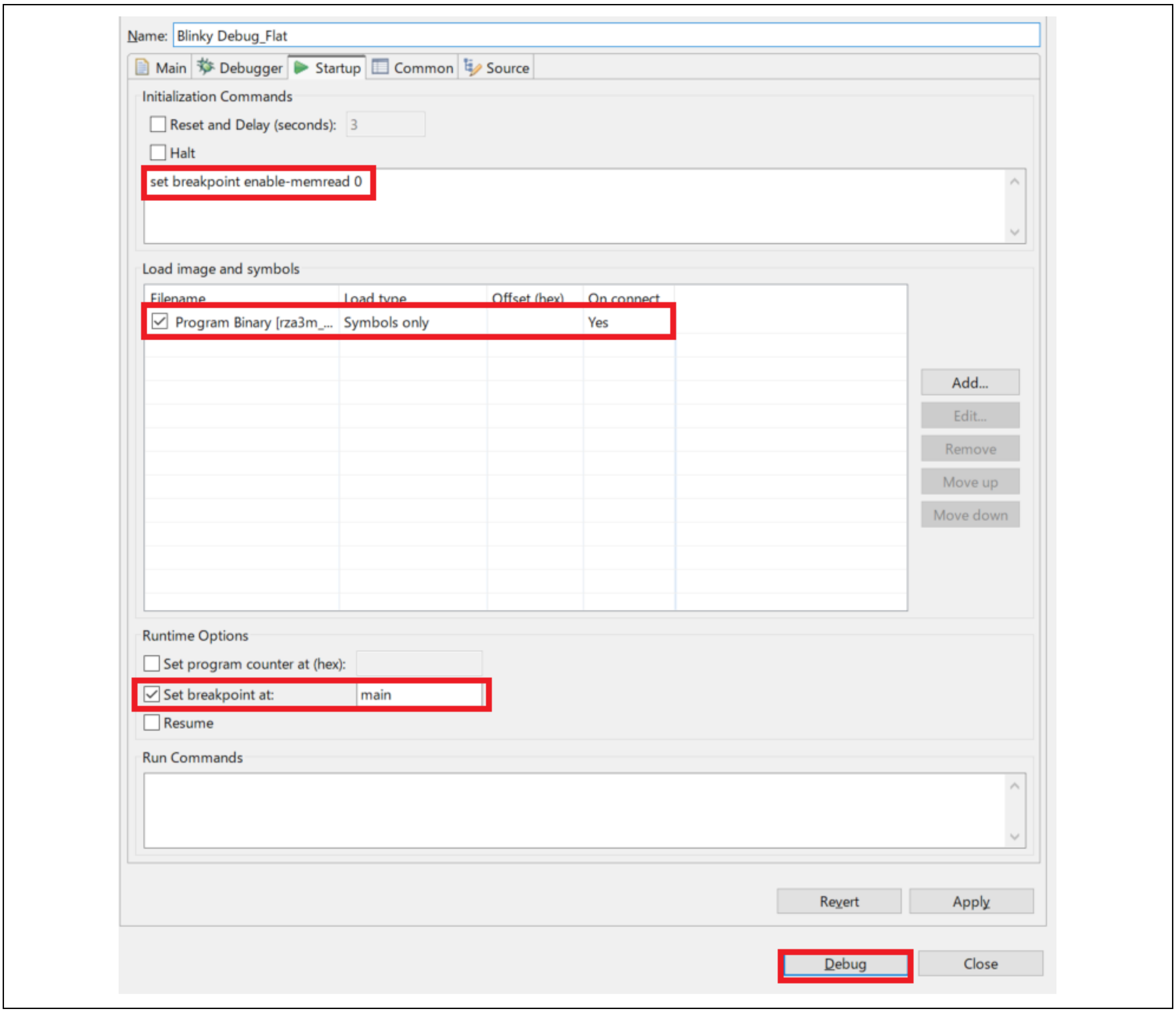
Edit Configuration

Edit Renesas GDB Hardware Debugging configuration Blinky Debug_Flat for Debug

Launch Configuration Name: Blinky Debug_Flat

Figure 94. e^2 studio Debugger Configurations window with Blinky project (5)

10. Make sure that only **Program Binary** is checked and that the Load type is set to **Symbols only**. Moreover, check the **Set breakpoint at:** check box, and add the “set breakpoint enable-memread 0” to the Initialization Commands.
    Then click [Debug] to load the application to DDR SDRAM and launch it.



Name: Blinky Debug_Flat

Initialization Commands

☐ Reset and Delay (seconds): 3

☐ Halt

```
set breakpoint enable-memread 0
```

Load image and symbols

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [rza3m_...	Symbols only		Yes

Runtime Options

☐ Set program counter at (hex):

☑ Set breakpoint at: main

☐ Resume

Run Commands

Figure 95. e^2 studio Debugger Configurations window with Blinky project (6)

4.6 Details about the Debug Process

In debug mode, e² studio executes the following tasks:

1. Downloading the application image to QSPI/OctaFlash ROM or DDR SDRAM.
2. Setting a breakpoint at main().
3. Setting the stack pointer register to the stack.

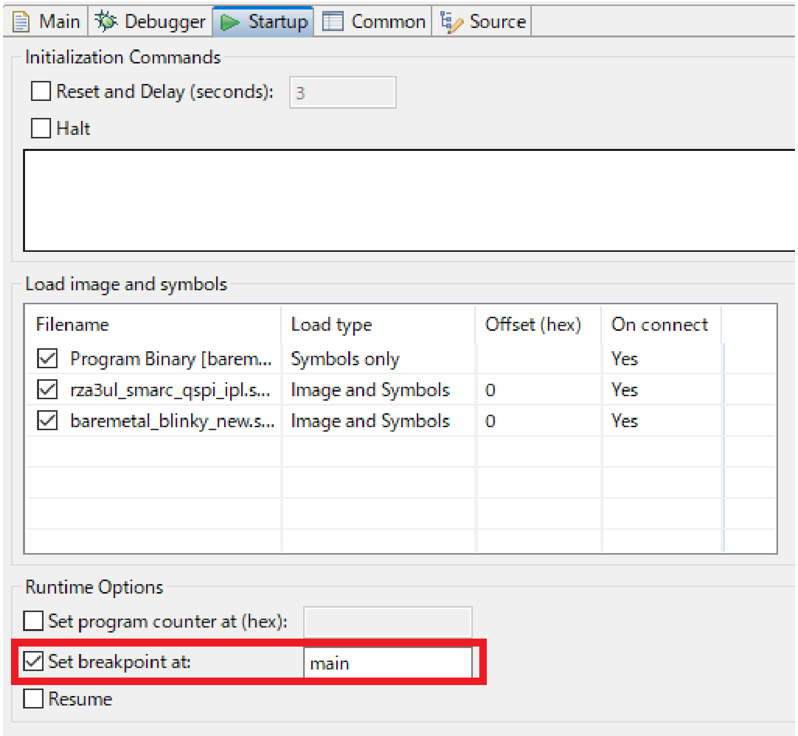
This section describes the details of the debug process of the Blinky Project.

4.6.1 Run the Blinky Project

Click [Run] > [Resume] or click on the Play icon shown below:

Figure 96. e² studio Debugger Play icon

Make sure the box **Set breakpoint at:** is checked and specify **main** as its value. The Program Counter should be stopped at the main() function.



Initialization Commands

☐ Reset and Delay (seconds): 3

☐ Halt

Load image and symbols

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [barem...	Symbols only		Yes
☑ rza3ul_smarc_qspi_ipl.s...	Image and Symbols	0	Yes
☑ baremetal_blinky_new.s...	Image and Symbols	0	Yes

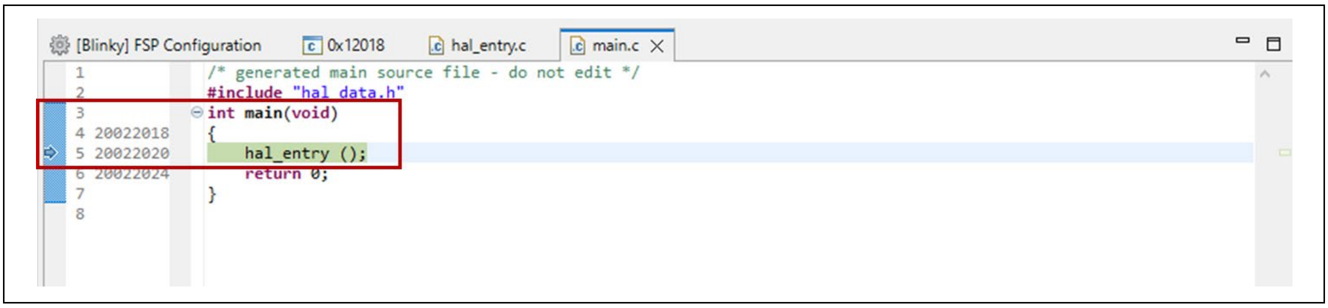
Runtime Options

☐ Set program counter at (hex):

☑ Set breakpoint at: main

☐ Resume

Figure 97. Set breakpoint at: option



```
/* generated main source file - do not edit */
#include "hal_data.h"
int main(void)
{
    hal_entry ();
    return 0;
}
```

Figure 98. Blinky project in Debug Mode

After that, the LED should start to blink when clicking [Run] > [Resume] or the Play icon again.

5. FSP Application Launch with e² studio

5.1 Create a Project

5.1.1 What is a Project?

In e² studio, all FSP applications are organized in RZ MPU projects. Setting up an RZ 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 in the project's root directory.

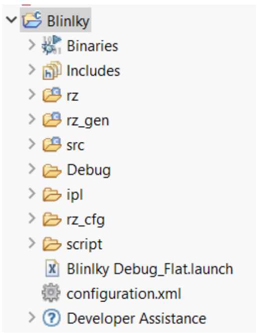
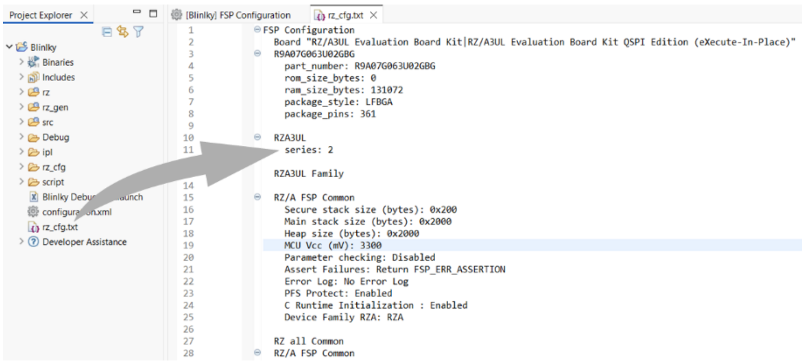


Figure 99. e² studio FSP Configuration Perspective

Double-click on the configuration.xml to open the RZ MPU Project Editor. To edit the project configuration, make sure that the **FSP Configuration** perspective shown below 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 100. e² studio FSP Configuration Perspective

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

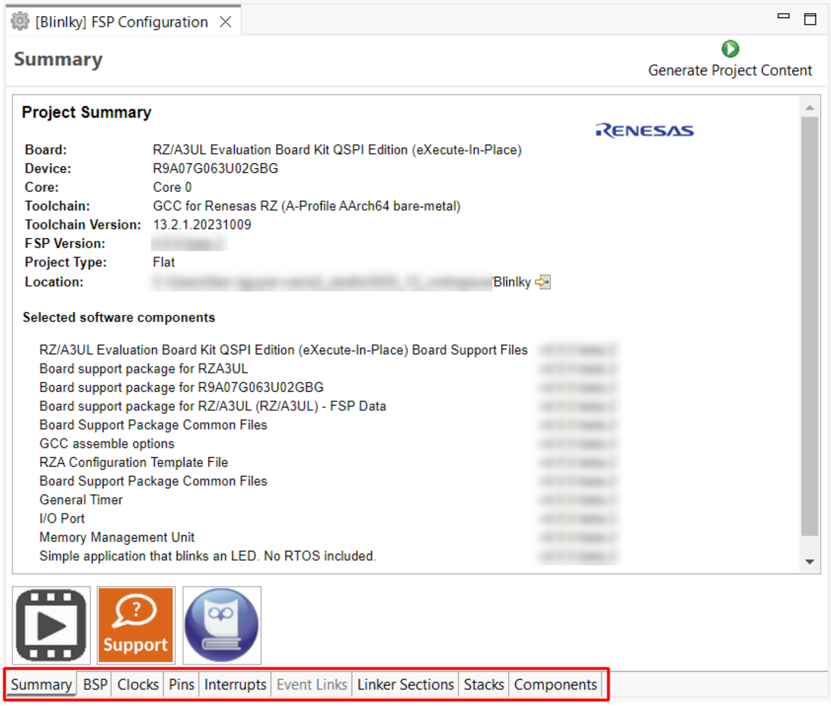


```
FSP Configuration
  Board "RZ/A3UL Evaluation Board Kit|RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place)"
  R9A07G063U02GBG
    part_number: R9A07G063U02GBG
    rom_size_bytes: 0
    ram_size_bytes: 131072
    package_style: LFBGA
    package_pins: 361

  RZA3UL
    series: 2

  RZA3UL Family

  RZ/A FSP Common
    Secure stack size (bytes): 0x200
    Main stack size (bytes): 0x2000
    Heap size (bytes): 0x2000
    MCU Vcc (mV): 3300
    Parameter checking: Disabled
    Assert Failures: Return FSP_ERR_ASSERTION
    Error Log: No Error Log
    PFS Protect: Enabled
    C Runtime Initialization : Enabled
    Device Family RZA: RZA

  RZ all Common
  RZ/A FSP Common
```

Figure 101. RZ Project Report

The RZ 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 Project Editor depend on the e² studio version, and the layout may vary slightly, the functionality should be easy to follow.



Project Summary

Board:	RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place)
Device:	R9A07G063U02GBG
Core:	Core 0
Toolchain:	GCC for Renesas RZ (A-Profile AArch64 bare-metal)
Toolchain Version:	13.2.1.20231009
FSP Version:	[redacted]
Project Type:	Flat
Location:	[redacted]Blinlky

Selected software components

RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place) Board Support Files	[redacted]
Board support package for RZA3UL	[redacted]
Board support package for R9A07G063U02GBG	[redacted]
Board support package for RZ/A3UL (RZ/A3UL) - FSP Data	[redacted]
Board Support Package Common Files	[redacted]
GCC assemble options	[redacted]
RZA Configuration Template File	[redacted]
Board Support Package Common Files	[redacted]
General Timer	[redacted]
I/O Port	[redacted]
Memory Management Unit	[redacted]
Simple application that blinks an LED. No RTOS included.	[redacted]

Figure 102. RZ Project Editor tabs

5.1.2 Creating a New Project

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

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

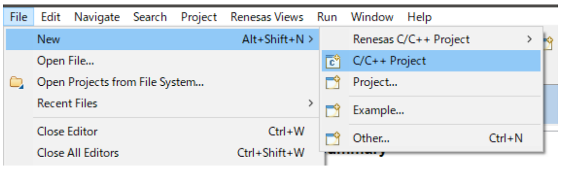
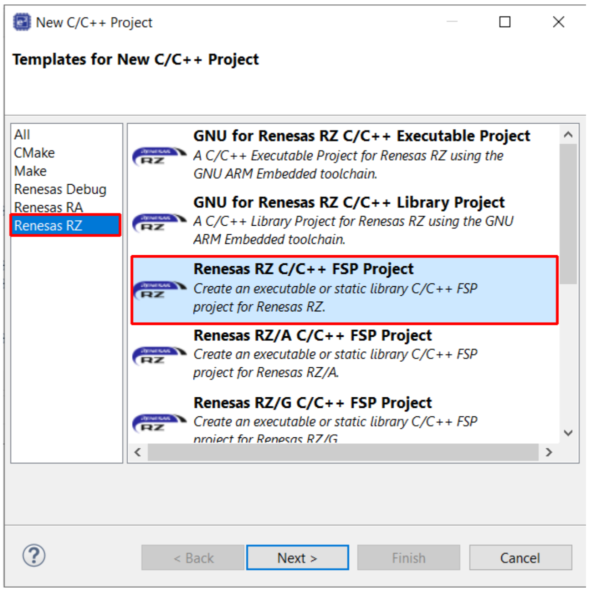


Figure 103. New RZ MPU Project

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



GNU for Renesas RZ C/C++ Executable Project
A C/C++ Executable Project for Renesas RZ using the GNU ARM Embedded toolchain.

GNU for Renesas RZ C/C++ Library Project
A C/C++ Library Project for Renesas RZ using the GNU ARM Embedded toolchain.

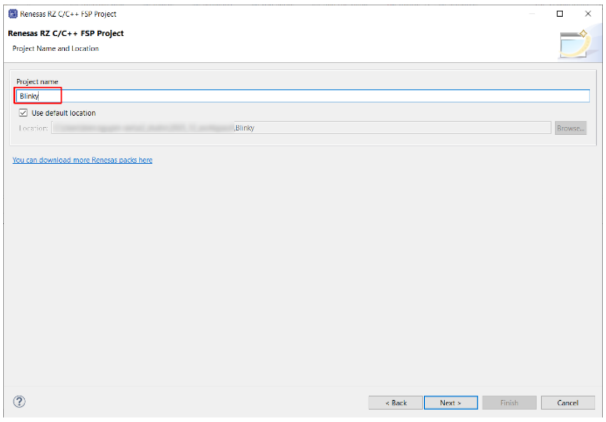
Renesas RZ C/C++ FSP Project
Create an executable or static library C/C++ FSP project for Renesas RZ.

Renesas RZ/A C/C++ FSP Project
Create an executable or static library C/C++ FSP project for Renesas RZ/A.

Renesas RZ/G C/C++ FSP Project
Create an executable or static library C/C++ FSP project for Renesas RZ/G.

Figure 104. New Project Templates

3. Select a project name and location.



Project name: Blinky

☑ Use default location

Location: [redacted]Blinky

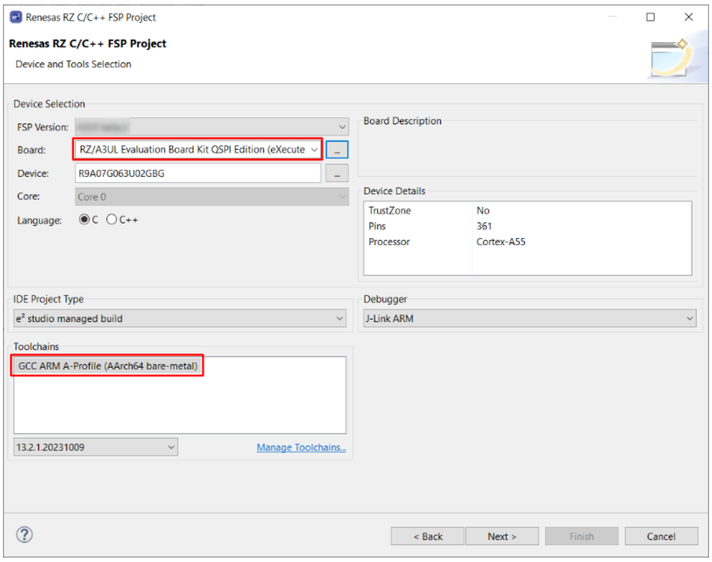
Figure 105. RZ MPU Project Generator (Screen 1)

4. Click [Next].

5.1.2.1 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** for your application. You can select an existing RZ MPU Evaluation Kit or **Custom User Board** for any of the RZ MPU devices with your own BSP definition. (Please refer 2.1.2.1 for more information about the RZ MPU Evaluation Kit.)
3. Select the **Device**. The **Device** is automatically populated based on the **Board** selection. Only change the **Device** when using the **Custom User Board QSPI Boot (eXecute-In-Place)** board selection.
4. The **Toolchain** selection defaults to **GCC ARM A-Profile (AArch64 bare-metal)**.
5. Select the **Toolchain version**. This should default to the installed toolchain version.
6. Select the **Debugger**. The J-Link Arm Debugger is preselected.
7. Click **Next**.



FSP Version: [redacted]

Board: RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute

Device: R9A07G063U02GBG

Core: Core 0

Language: ● C ○ C++

TrustZone	No
Pins	361
Processor	Cortex-A55

IDE Project Type: e^2 studio managed build

Debugger: J-Link ARM

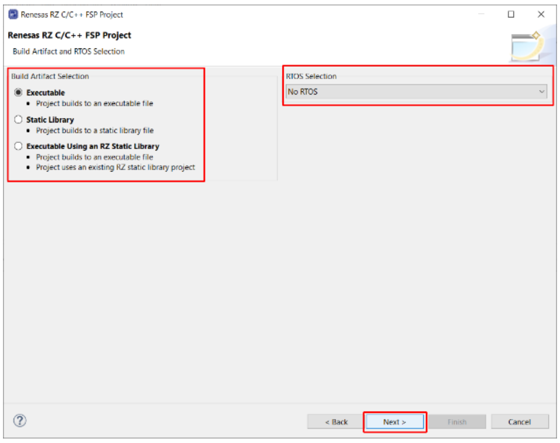
Toolchains: GCC ARM A-Profile (AArch64 bare-metal)

13.2.1.20231009

Figure 106. RZ MPU Project Generator (Screen 2)

5.1.2.2 Selecting a Project Template

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



● **Executable**
- Project builds to an executable file

○ **Static Library**
- Project builds to a static library file

○ **Executable Using an RZ Static Library**
- Project builds to an executable file
- Project uses an existing RZ static library project

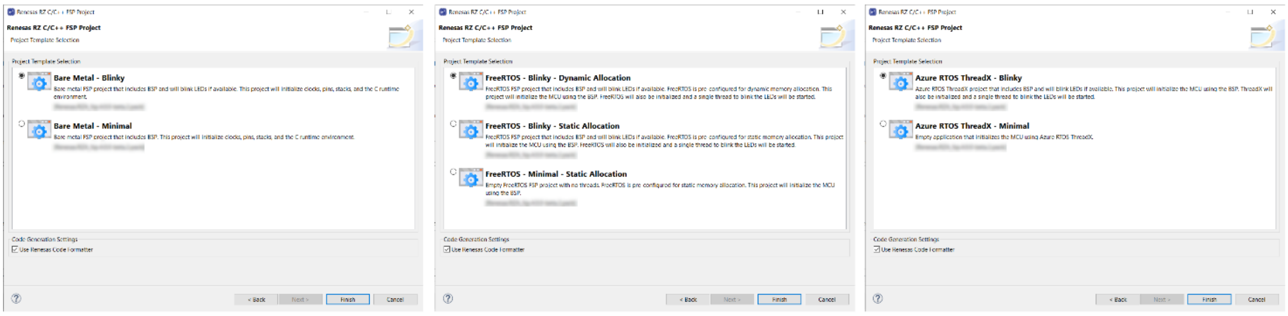
RTOS Selection: No RTOS

Figure 107. RZ MPU Project Generator (Screen 3)

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 MPU Pack. To add threads, select **RTOS**, or **No RTOS** if an RTOS is not being used. Once you have selected the appropriate template, click **Finish**.

Note:

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



● **Bare Metal - Blinky**
Bare metal FSP project that includes BSP and will blink LEDs if available. This project will initialize clocks, pins, stacks, and the C runtime environment.
[redacted]

○ **Bare Metal - Minimal**
Bare metal FSP project that includes BSP. This project will initialize clocks, pins, stacks, and the C runtime environment.
[redacted]

☑ Use Renesas Code Formatter

● **FreeRTOS - Blinky - Dynamic Allocation**
FreeRTOS FSP project that includes BSP and will blink LEDs if available. FreeRTOS is pre-configured for dynamic memory allocation. This project will initialize the MCU using the BSP. FreeRTOS will also be initialized and a single thread to blink the LEDs will be started.
[redacted]

○ **FreeRTOS - Blinky - Static Allocation**
FreeRTOS FSP project that includes BSP and will blink LEDs if available. FreeRTOS is pre-configured for static memory allocation. This project will initialize the MCU using the BSP. FreeRTOS will also be initialized and a single thread to blink the LEDs will be started.
[redacted]

○ **FreeRTOS - Minimal - Static Allocation**
Empty FreeRTOS FSP project with no threads. FreeRTOS is pre-configured for static memory allocation. This project will initialize the MCU using the BSP.
[redacted]

☑ Use Renesas Code Formatter

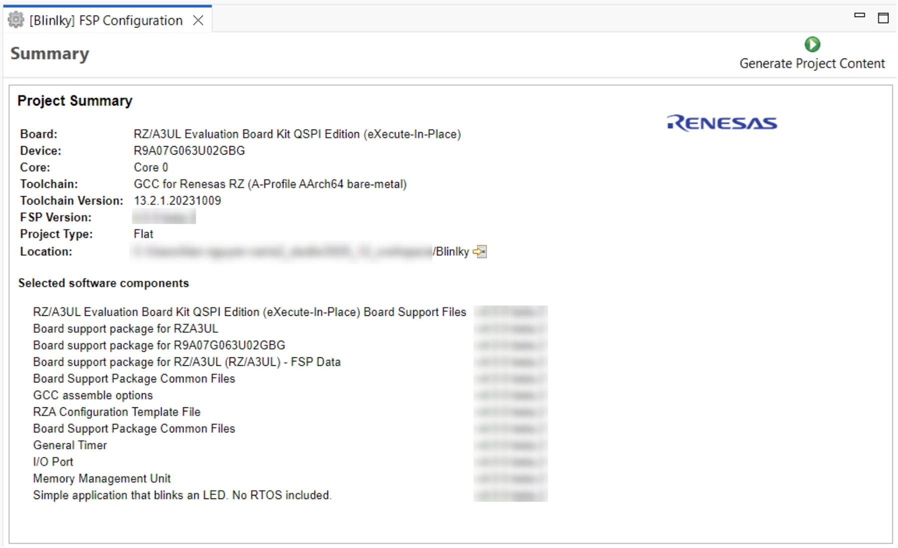
● **Azure RTOS ThreadX - Blinky**
Azure RTOS ThreadX project that includes BSP and will blink LEDs if available. This project will initialize the MCU using the BSP. ThreadX will also be initialized and a single thread to blink the LEDs will be started.
[redacted]

○ **Azure RTOS ThreadX - Minimal**
Empty application that initializes the MCU using Azure RTOS ThreadX.
[redacted]

☑ Use Renesas Code Formatter

Figure 108. RZ MPU Project Generator (Screen 4)

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



Project Summary

Board: RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place)
Device: R9A07G063U02GBG
Core: Core 0
Toolchain: GCC for Renesas RZ (A-Profile AArch64 bare-metal)
Toolchain Version: 13.2.1.20231009
FSP Version: [redacted]
Project Type: Flat
Location: [redacted]/Blinlky

Selected software components

RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place) Board Support Files	[redacted]
Board support package for RZA3UL	[redacted]
Board support package for R9A07G063U02GBG	[redacted]
Board support package for RZ/A3UL (RZ/A3UL) - FSP Data	[redacted]
Board Support Package Common Files	[redacted]
GCC assemble options	[redacted]
RZA Configuration Template File	[redacted]
Board Support Package Common Files	[redacted]
General Timer	[redacted]
I/O Port	[redacted]
Memory Management Unit	[redacted]
Simple application that blinks an LED. No RTOS included.	[redacted]

Figure 109. RZ MPU Project Editor and available editor tabs

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

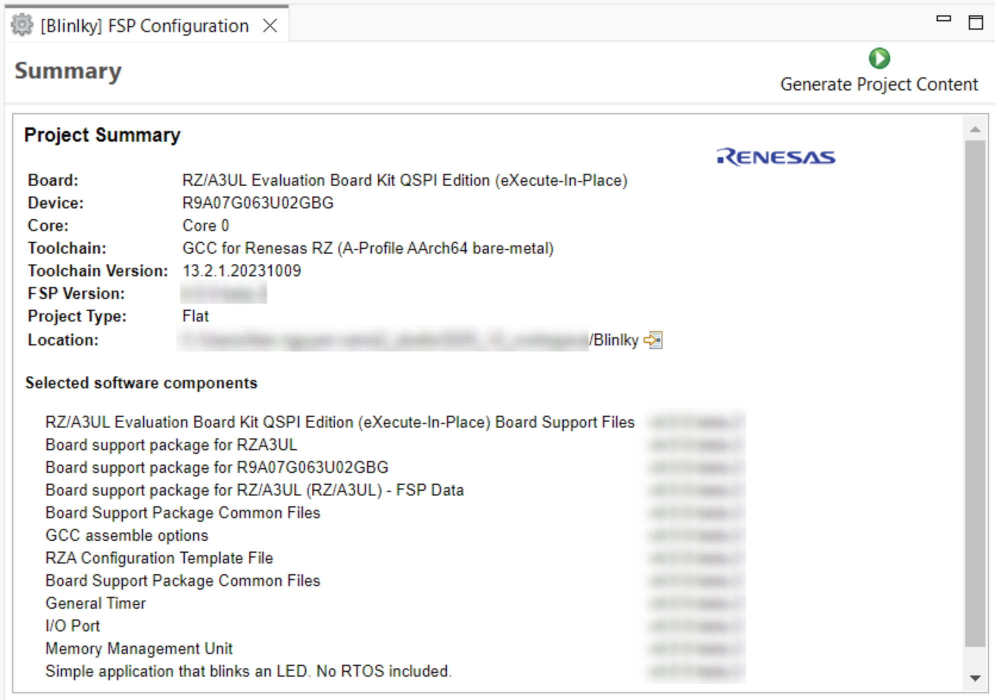
The functions and use of each of the supported tabs are explained in detail in the next section.

Please note that RZ FSP doesn't support the **Event Links** tab, and so, those tabs are grayed out as shown above.

5.2 Configuring a Project

Each of the configurable elements in an FSP project can be edited using the appropriate tab in the RZ 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.

5.2.1 Summary Tab



Project Summary

Board:	RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place)
Device:	R9A07G063U02GBG
Core:	Core 0
Toolchain:	GCC for Renesas RZ (A-Profile AArch64 bare-metal)
Toolchain Version:	13.2.1.20231009
FSP Version:	[redacted]
Project Type:	Flat
Location:	[redacted]/Blinlky

Selected software components

RZ/A3UL Evaluation Board Kit QSPI Edition (eXecute-In-Place) Board Support Files	[redacted]
Board support package for RZA3UL	[redacted]
Board support package for R9A07G063U02GBG	[redacted]
Board support package for RZ/A3UL (RZ/A3UL) - FSP Data	[redacted]
Board Support Package Common Files	[redacted]
GCC assemble options	[redacted]
RZA Configuration Template File	[redacted]
Board Support Package Common Files	[redacted]
General Timer	[redacted]
I/O Port	[redacted]
Memory Management Unit	[redacted]
Simple application that blinks an LED. No RTOS included.	[redacted]

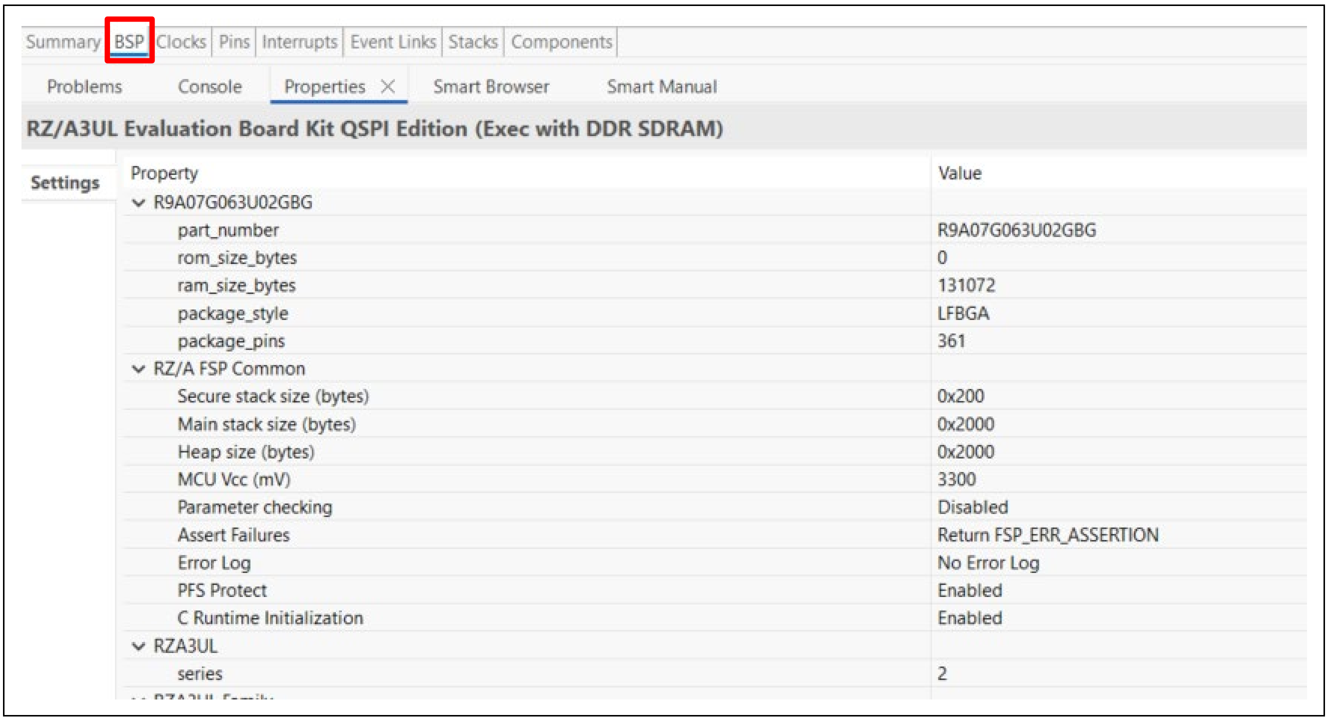
Figure 110. Configuration Summary tab

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

5.2.2 Configuring the BSP

The **BSP** tab shows the currently selected board (if any) and device. The Properties view is 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.



RZ/A3UL Evaluation Board Kit QSPI Edition (Exec with DDR SDRAM)

Property	Value
∨ R9A07G063U02GBG	
part_number	R9A07G063U02GBG
rom_size_bytes	0
ram_size_bytes	131072
package_style	LFBGA
package_pins	361
∨ RZ/A FSP Common	
Secure stack size (bytes)	0x200
Main stack size (bytes)	0x2000
Heap size (bytes)	0x2000
MCU Vcc (mV)	3300
Parameter checking	Disabled
Assert Failures	Return FSP_ERR_ASSERTION
Error Log	No Error Log
PFS Protect	Enabled
C Runtime Initialization	Enabled
∨ RZA3UL	
series	2

Figure 111. 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.

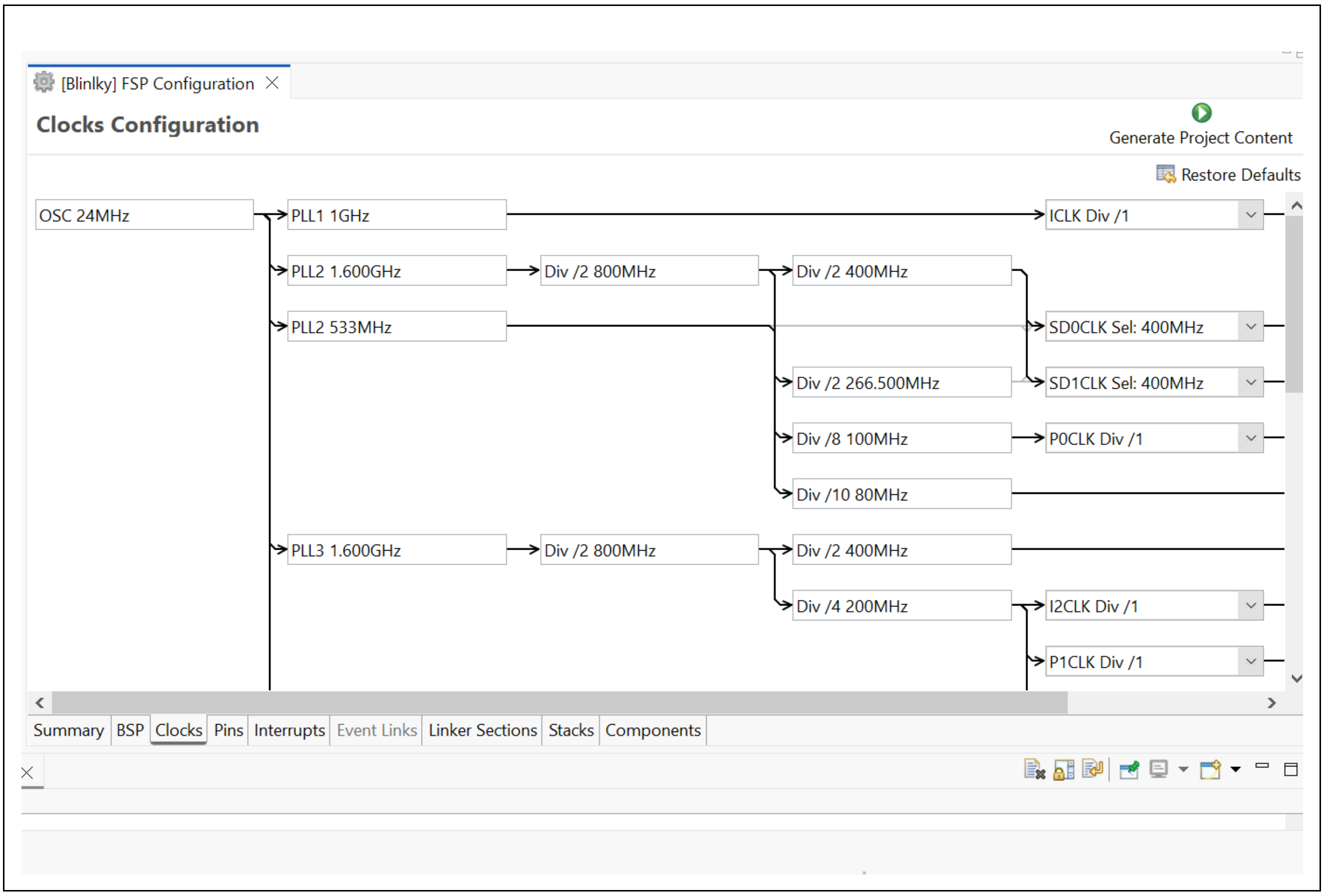
When you click the **Generate Project Content** button, the BSP configuration contents are written to rz_cfg/fsp_cfg/bsp/bsp_cfg.h. This file is created if it does not already exist.

Warning:

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

5.2.3 Configuring Clocks

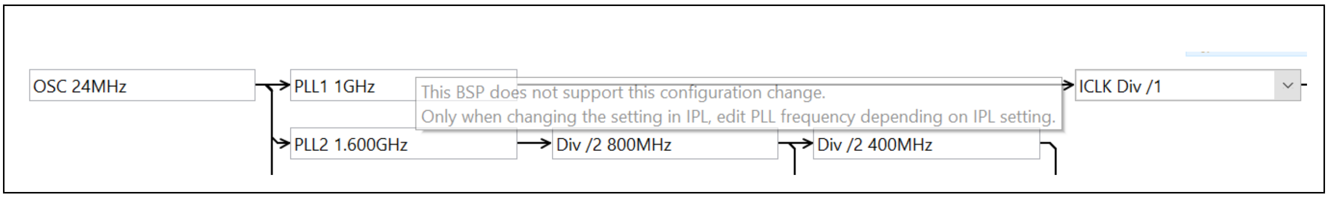
The **Clocks** tab presents a graphical view of the MPU's clock tree, and each HAL driver uses the settings for dedicated numerical calculation. For example, the scif_uart driver calculates the communication rate from the settings in the Clocks tab. Please note that PLLs should be configured by IPL and, therefore, PLL settings should be consistent with those in IPL.



OSC 24MHz
PLL1 1GHz
PLL2 1.600GHz
PLL2 533MHz
PLL3 1.600GHz
Div /2 800MHz
Div /2 400MHz
Div /2 266.500MHz
Div /8 100MHz
Div /10 80MHz
Div /2 800MHz
Div /2 400MHz
Div /4 200MHz
ICLK Div /1
SD0CLK Sel: 400MHz
SD1CLK Sel: 400MHz
P0CLK Div /1
I2CLK Div /1
P1CLK Div /1

Figure 112. Configuration Clocks tab

When mousing over the blocks of PLLs on the clocks tab, you should see the pop-up message describing this precaution.



This BSP does not support this configuration change.
Only when changing the setting in IPL, edit PLL frequency depending on IPL setting.

Figure 113. Precautions for PLL settings

When you click the **Generate Project Content** button, the clock configuration contents are written to: rz_gen/bsp_clock_cfg.h

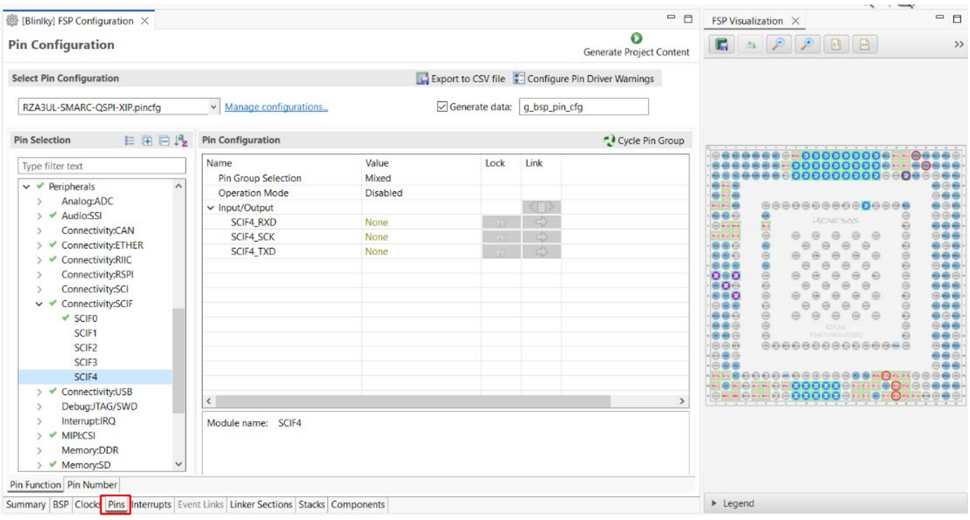
This file will be created if it does not already exist.

Warning:

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

5.2.4 Configuring Pins

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



RZA3UL-SMARC-QSPI-XIP.pincfg

☑ Generate data: g_bsp_pin_cfg

Pin Configuration

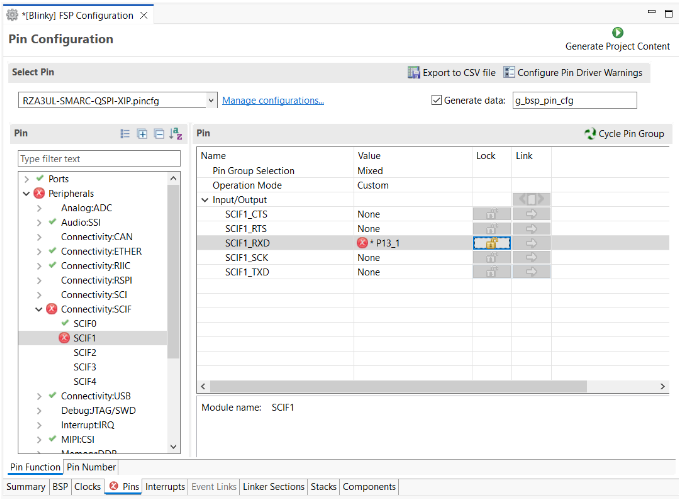
Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Disabled		
Input/Output			
SCIF4_RXD	None		
SCIF4_SCK	None		
SCIF4_TXD	None		

Module name: SCIF4

Figure 114. Pin Configuration

The pin configurator includes a built-in conflict checker. So, if the same pin is allocated to another peripheral or I/O function, the pin will be shown as red in the **FSP Visualization** view and with a white cross in a red square in the **Pin Selection** pane and **Pin Configuration** pane in the main **Pins** tab.

In the example shown below, port P13_1 is already used by the Display, and the attempt to connect to this pin to the Serial Communication Interface with FIFO (SCIF) results in a dangling connection error. To fix this error, select another port from the pin drop-down list or disable the Display.



RZA3UL-SMARC-QSPI-XIP.pincfg

☑ Generate data: g_bsp_pin_cfg

Pin

Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Custom		
Input/Output			
SCIF1_CTS	None		
SCIF1_RTS	None		
SCIF1_RXD	☒ * P13_1		
SCIF1_SCK	None		
SCIF1_TXD	None		

Module name: SCIF1

Figure 115. e^2 studio Pin Configurator

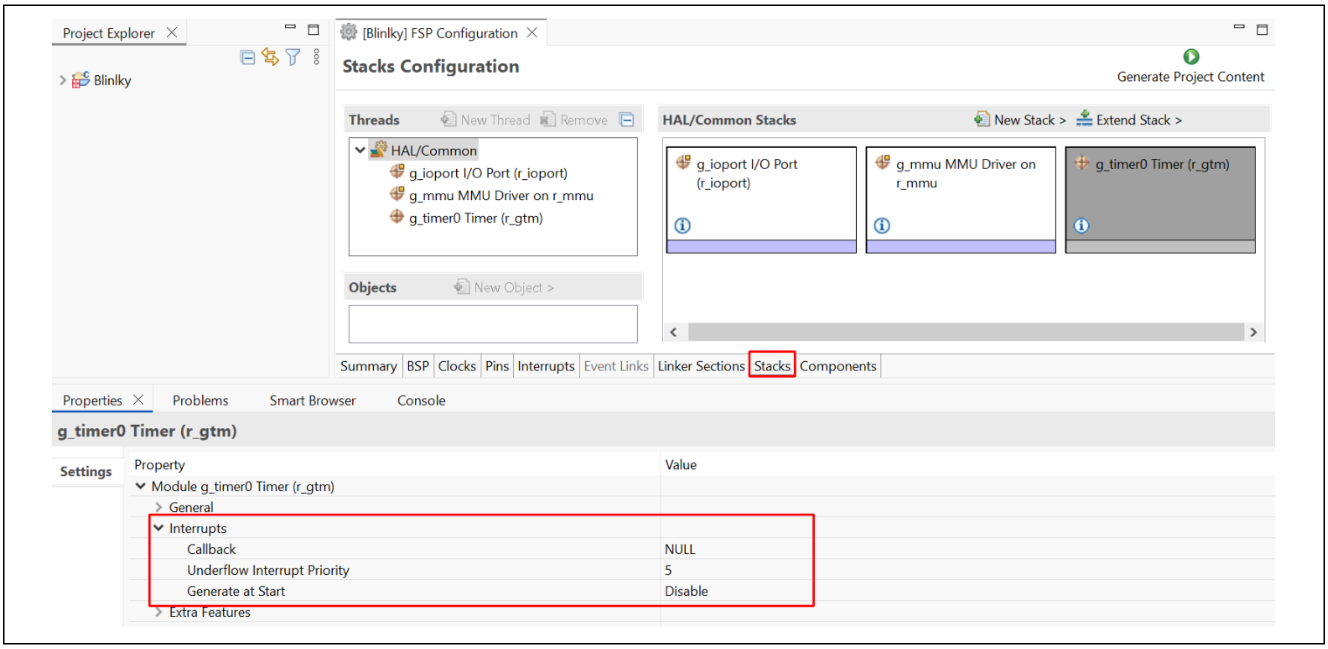
When you click the **Generate Project Content** button, the pin configuration contents are written to: rz_gen\pin_data.c This file will be created if it does not already exist.

Warning:

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

5.2.5 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.

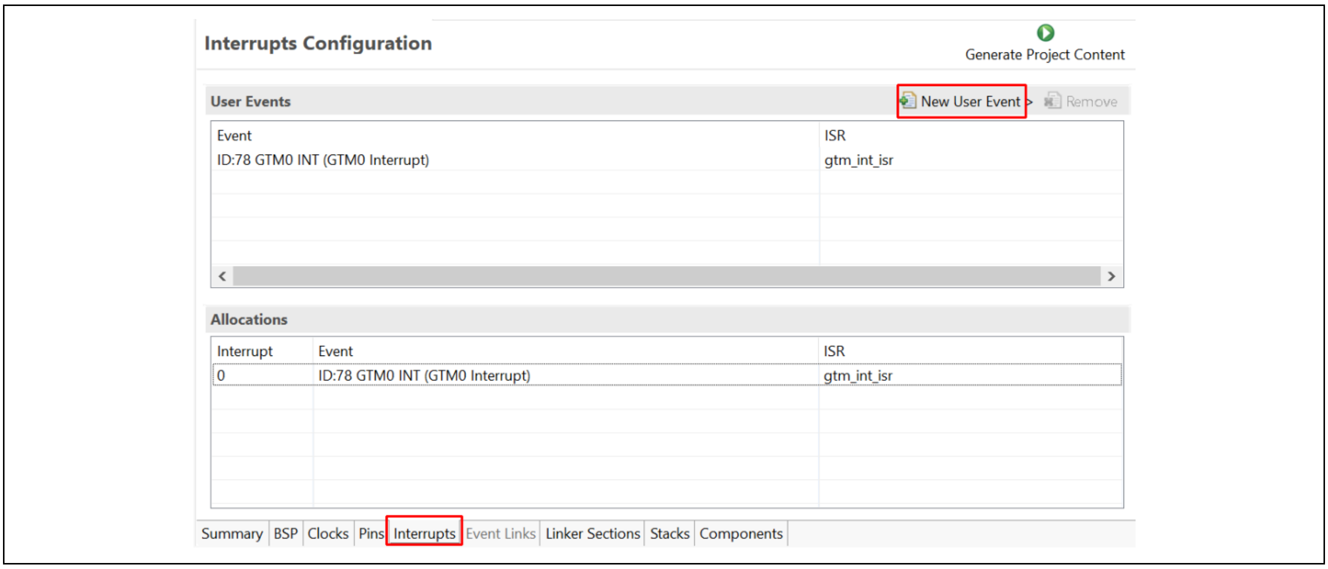


Property	Value
Module g_timer0 Timer (r_gtm)	
General	
Interrupts	
Callback	NULL
Underflow Interrupt Priority	5
Generate at Start	Disable
Extra Features	

Figure 116. Configuring Interrupts in the Stacks tab

5.2.6 Creating Interrupts from the Interrupts Tab

On the **Interrupts** tab, the user can bypass a peripheral interrupt set by the FSP by setting a user-defined ISR. This can be done by adding a new event via the New User Event button.



User Events

Event	ISR
ID:78 GTM0 INT (GTM0 Interrupt)	gtm_int_isr

Allocations

Interrupt	Event	ISR
0	ID:78 GTM0 INT (GTM0 Interrupt)	gtm_int_isr

Figure 117. Configuring interrupt in Interrupt Tab

5.2.7 Viewing Event Links

RZ FSP doesn't support the **Event Links** tab, and it is grayed out.

5.2.8 Adding Threads and Drivers

Every RTOS-based RZ FSP Project includes at least one RTOS Thread and a stack of FSP module running in that thread. The **Stacks** tab is a graphical user interface that helps you add the right modules to a thread and configure the properties of both the threads and the modules associated with each thread. Once you have configured the thread, e^2 studio automatically generates the code reflecting your configuration choices.

For any driver, or, more generally, any module that you add to a thread, e^2 studio automatically resolves all dependencies with other modules and creates the appropriate stack. This stack is displayed in the **Stacks** pane, which e^2 studio populates with the selected modules and module options for the selected thread.

The default view of the **Stacks** tab includes a Common Thread called **HAL/Common**. This thread includes the driver for I/O control (IOPORT). The default stack is shown in the **HAL/Common Stacks** pane. The default modules added to the HAL/Common driver are special in that the FSP only requires a single instance of each, which e^2 studio then includes in every user-defined thread by default.

In applications that do not use an RTOS or run outside of the RTOS, the HAL/Common thread becomes the default location where you can add additional drivers to your application.

For a detailed description of how to add and configure modules and stacks, see the following sections:

- **Adding and Configuring HAL Drivers**
- **Adding Drivers to a Thread and Configuring the Drivers**

Only you have added a module either to HAL/Common or to a new thread; you can access the driver's configuration options in the **Properties** view. If you added thread objects, you were able to access the objects' configuration options in the **Properties** view in the same way.

5.2.9 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 on the FSP.
3. Select a driver from the menu **New Stack > Driver**.

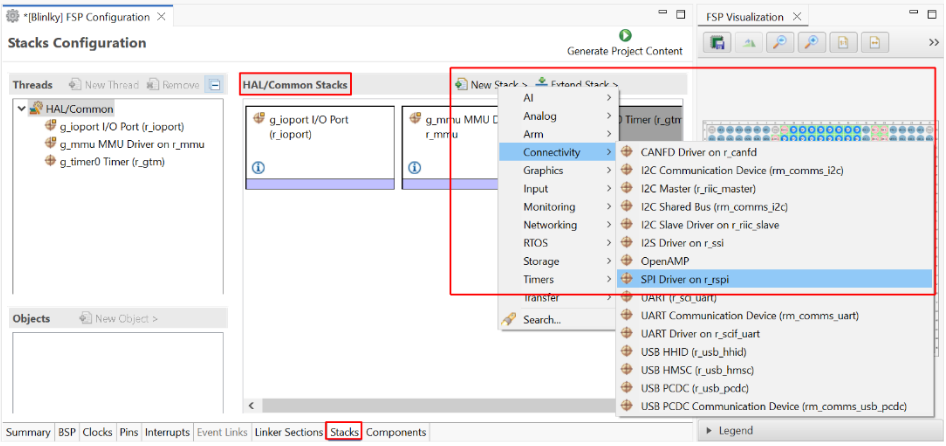


Figure 118. e^2 studio Project configurator - Adding drivers

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

e^2 studio adds the following files when you click the **Generate Project Content** button:

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

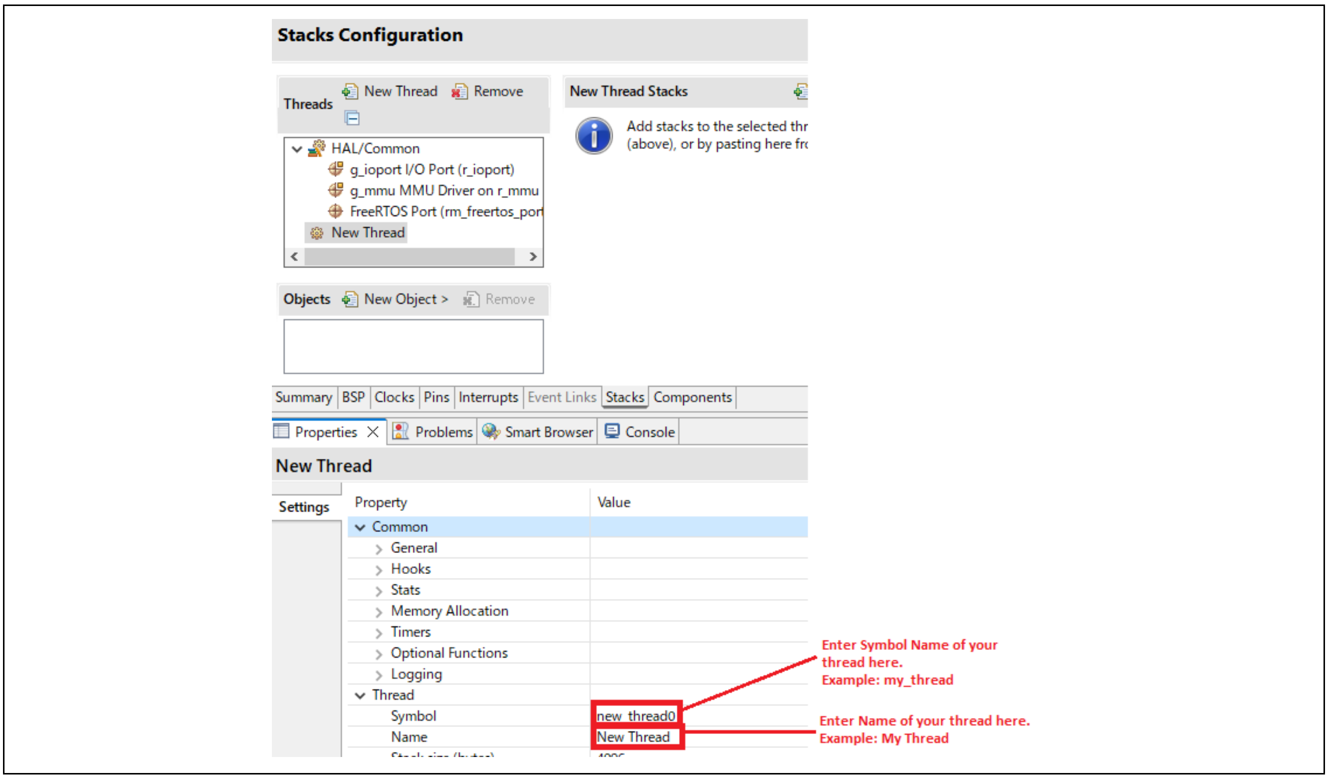
File	Contents	Overwritten by Generate Project Content?
rz_gen/main.c	Contains main() calling generated and user code. When called, the BSP has already initialized the MPU	Yes
rz_gen/hal_data.c	Configuration structures for HAL Driver only modules	Yes
rz_gen/hal_data.h	Header file for HAL driver only modules	Yes
src/hal_entry.c	User entry point for HAL Driver only code. Add your code here	No
src/mmu_page_table.c	Virtual memory page table settings	No
src/sections.c	Rules for section transfer from ROM to RAM	No
src/syscalls.c	Low-level processing stub for file I/O functions	No

The configuration header files for all included modules are created or overwritten in the folder “rz_cfg/fsp_cfg”.

5.2.10 Adding Drivers to a Thread and Configuring the Drivers

For an application that uses the RTOS, you can add one or more threads, and for each thread at least one module that runs in the thread. You can select modules from the Driver dropdown menu. To add modules to a thread, follow these steps:

1. In the **Threads** pane, click **New Thread** to add a Thread.



Symbol new_thread0

Name New Thread

Enter Symbol Name of your thread here.
Example: my_thread

Enter Name of your thread here.
Example: My Thread

Figure 119. Adding a new RTOS Thread on the Stack tab

2. In the properties view, click on the Name and Symbol entries and enter a distinctive name and symbol for the new thread.

Note: e^2 studio updates the name of the thread stacks pane to **My Thread Stacks**.

3. In the **My Thread Stacks** pane, click on **New Stack** to see a list of modules and drivers.

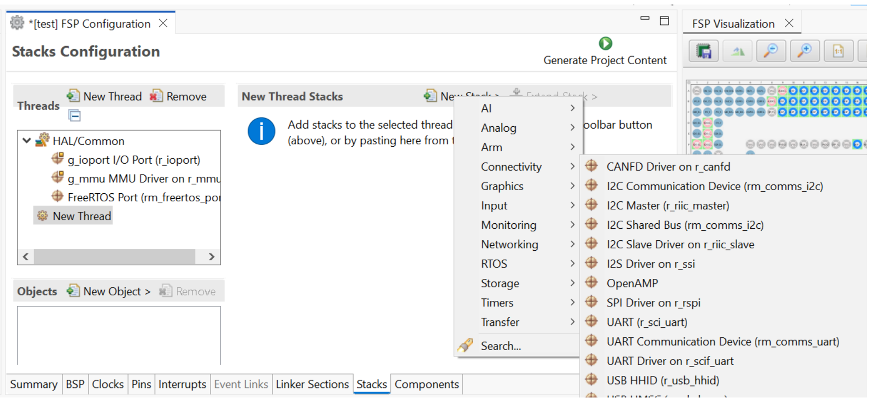
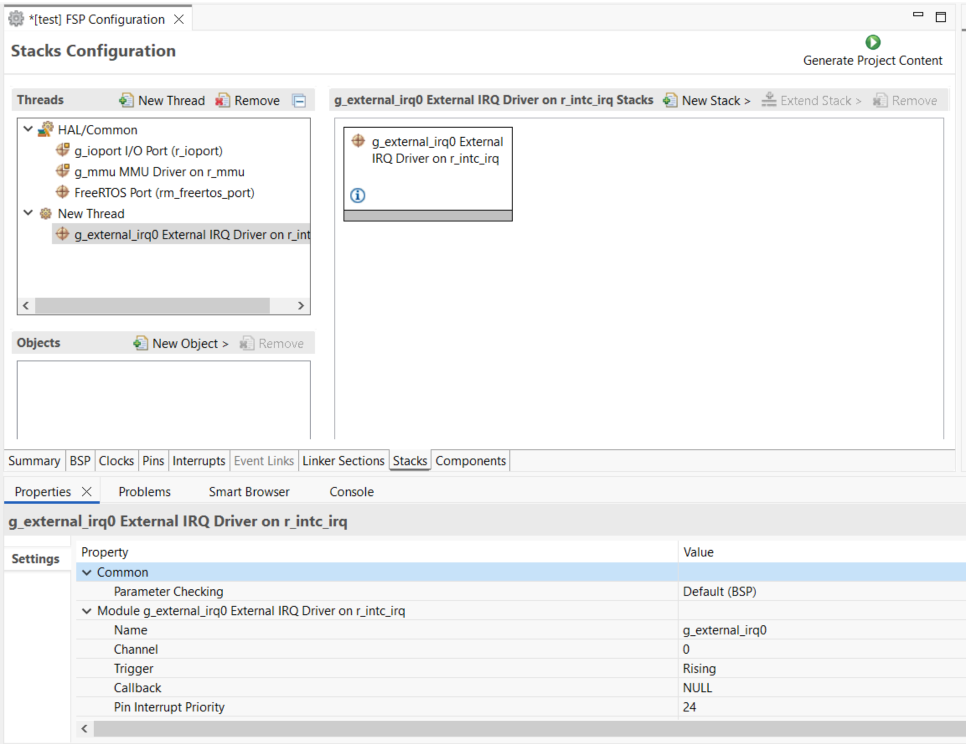


Figure 120. Adding Modules and Drivers to a thread

4. Select a module or driver from the list.
5. Click on the added driver and configure the driver as required by the application by updating the configuration parameters in the **Properties** view. To see the selected module or driver and be able to edit its properties, make sure the Thread containing the driver is highlighted in the **Threads** pane.



Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_external_irq0 External IRQ Driver on r_intc_irq	
Name	g_external_irq0
Channel	0
Trigger	Rising
Callback	NULL
Pin Interrupt Priority	24

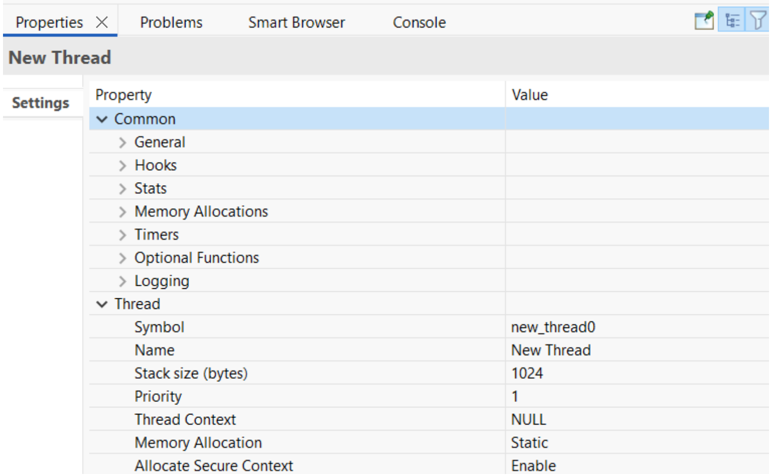
Figure 121. Configuring Module or Driver properties

6. When you press the Generate Project Content button for the example above, e^2 studio creates the files as shown in the following table:

File	Contents	Overwritten by Generate Project Content?
rz_gen/main.c	Contains main() calling generated and user code. When called, the BSP will have initialized the MPU.	Yes
rz_gen/my_thread.c	Generated thread “my_thread” and configuration structures for modules added to this thread.	Yes
rz_gen/my_thread.h	Header file for thread “my_thread”	Yes
rz_gen/hal_data.c	Configuration structures for HAL Driver only modules.	Yes
rz_gen/hal_data.h	Header file for HAL driver only modules.	Yes
src/hal_entry.c	User entry point for HAL Driver only code. Add your code here.	No
src/my_thread_entry.c	User entry point for thread “my_thread”. Add your code here.	No

5.2.11 Configuring Threads

If the application uses RTOS, the Stacks tab can be used to simplify the creation of RTOS threads, semaphores, mutexes, and event flags. The components of each thread can be configured from the **Properties** view, as shown below:



New Thread

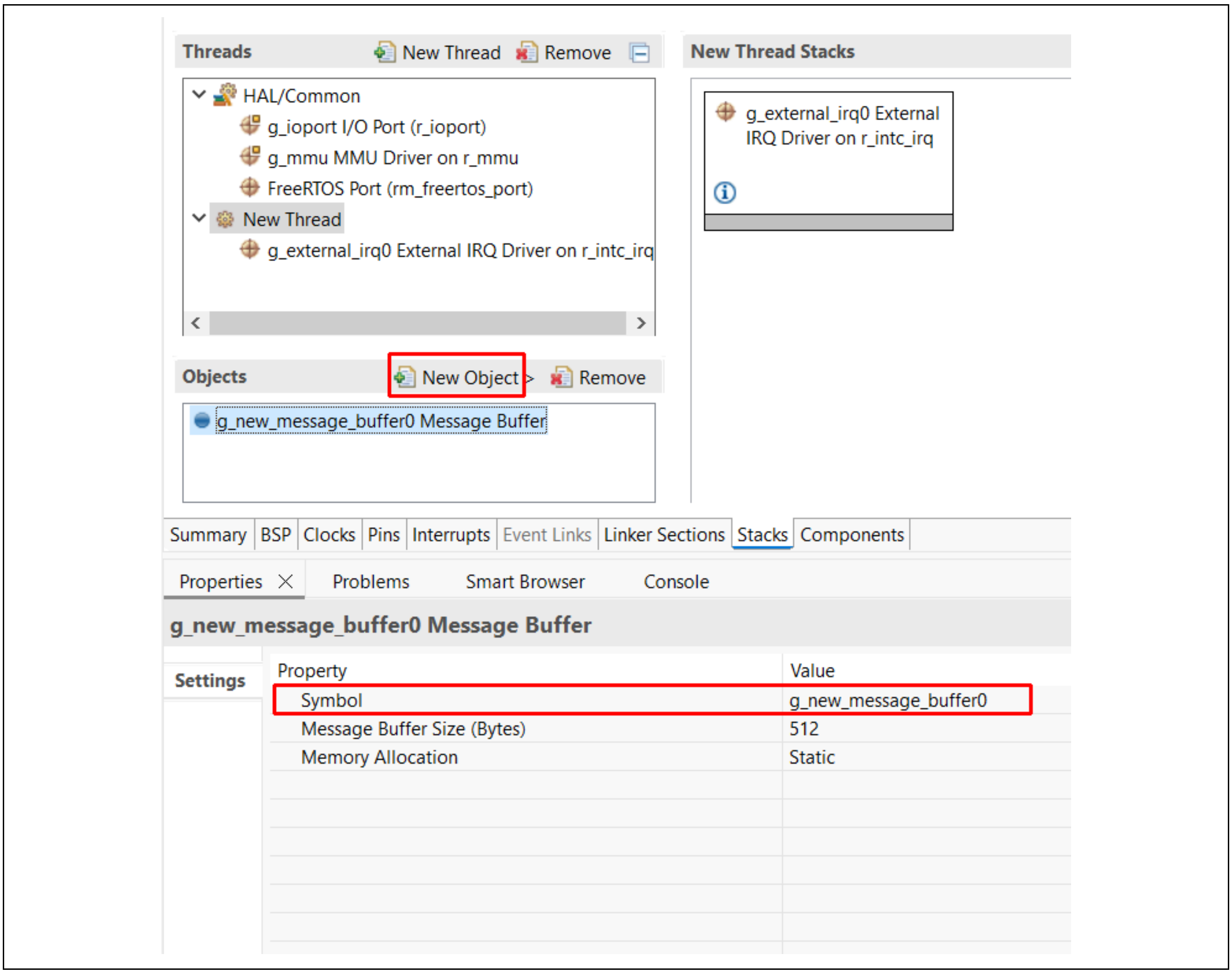
Property	Value
Common	
General	
Hooks	
Stats	
Memory Allocations	
Timers	
Optional Functions	
Logging	
Thread	
Symbol	new_thread0
Name	New Thread
Stack size (bytes)	1024
Priority	1
Thread Context	NULL
Memory Allocation	Static
Allocate Secure Context	Enable

Figure 122. New Thread Properties

The Properties view contains settings that are common for all Threads (**Common**) and settings for this particular thread (**Thread**).

For this thread instance, the thread's name and properties (such as priority level or stack size) can be easily configured. e^2 studio checks that the entries in the property field are valid. For example, it will verify that the field **Priority**, which requires an integer value, only contains numeric values between 0 and 9.

To add RTOS resources to a Thread, select a thread and click on **New Object** in the Thread Objects pane. The pane takes on the name of the selected thread, in this case, **My Thread Objects**.



g_new_message_buffer0 Message Buffer

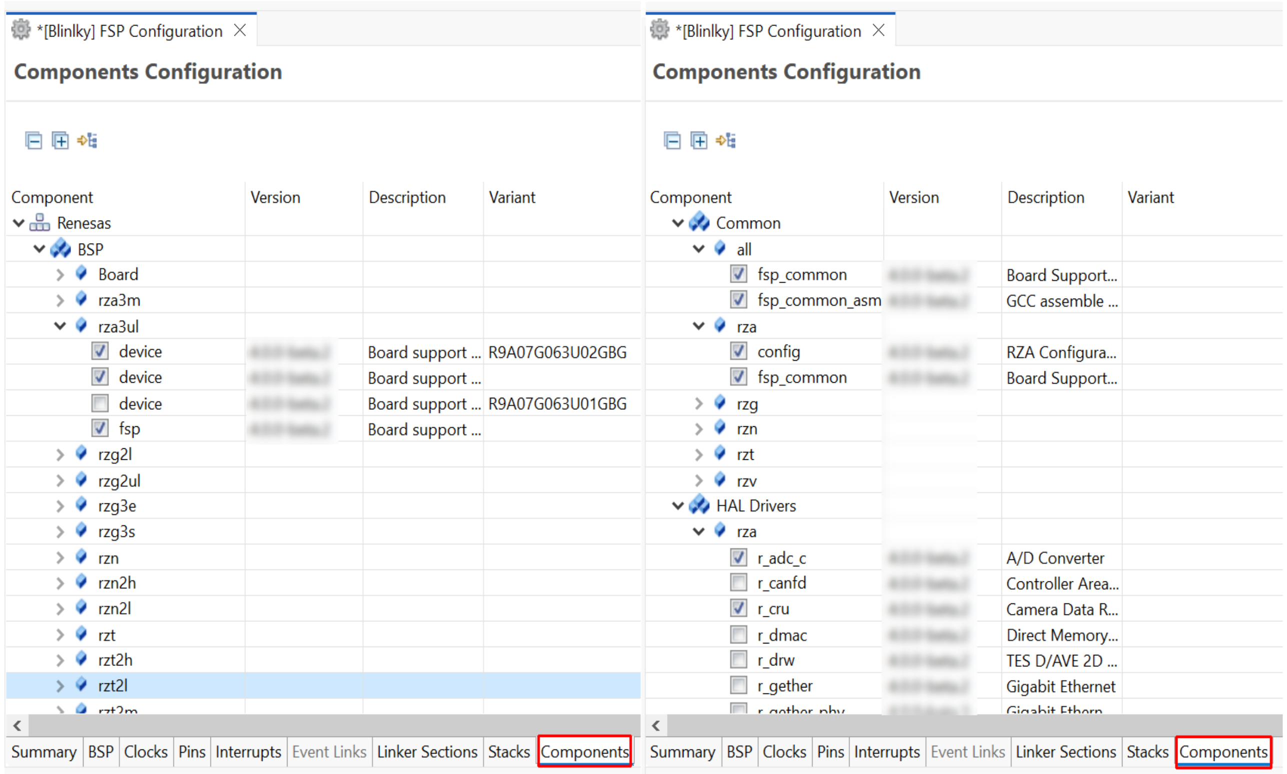
Property	Value
Symbol	g_new_message_buffer0
Message Buffer Size (Bytes)	512
Memory Allocation	Static

Figure 123. Configuring Thread Object Properties

Make sure to give each thread object a unique symbol by updating the **Symbol** entries in the **Properties** view.

5.3 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.



Component	Version	Description	Variant
Renesas			
BSP			
Board			
rza3m			
rza3ul			
☑ device	[redacted]	Board support ...	R9A07G063U02GBG
☑ device	[redacted]	Board support ...	
☐ device	[redacted]	Board support ...	R9A07G063U01GBG
☑ fsp	[redacted]	Board support ...	
rzg2l			
rzg2ul			
rzg3e			
rzg3s			
rzn			
rzn2h			
rzn2l			
rzt			
rzt2h			
rzt2l			

Component	Version	Description	Variant
Common			
all			
☑ fsp_common	[redacted]	Board Support...	
☑ fsp_common_asm	[redacted]	GCC assemble ...	
rza			
☑ config	[redacted]	RZA Configura...	
☑ fsp_common	[redacted]	Board Support...	
rzg			
rzn			
rzt			
rzv			
HAL Drivers			
rza			
☑ r_adc_c	[redacted]	A/D Converter	
☐ r_canfd	[redacted]	Controller Area...	
☑ r_cru	[redacted]	Camera Data R...	
☐ r_dmac	[redacted]	Direct Memory...	
☐ r_drw	[redacted]	TES D/AVE 2D ...	
☐ r_gether	[redacted]	Gigabit Ethernet	

Figure 124. Components Tab

By clicking the **Generate Project Content** button, the .c and .h files for each selected component are copied into the following folders:

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

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

5.4 Debugging the Project

Once your project builds without errors, you can use the Debugger to download your application to the board and execute it.

To debug an application, follow these steps:

1. On the drop-down list next to the debug icon, select **Debug Configurations**.

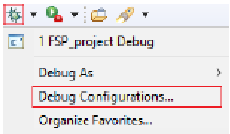
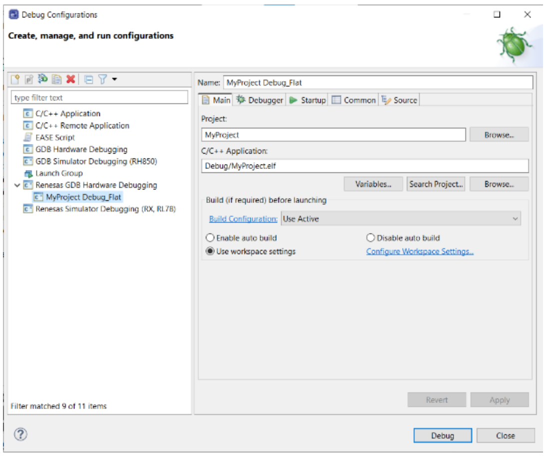


Figure 125. Select Debug Configurations

2. In the **Debug Configurations** view, click on your project listed as **MyProject Debug_Flat**.



Name: MyProject Debug_Flat

Project: MyProject

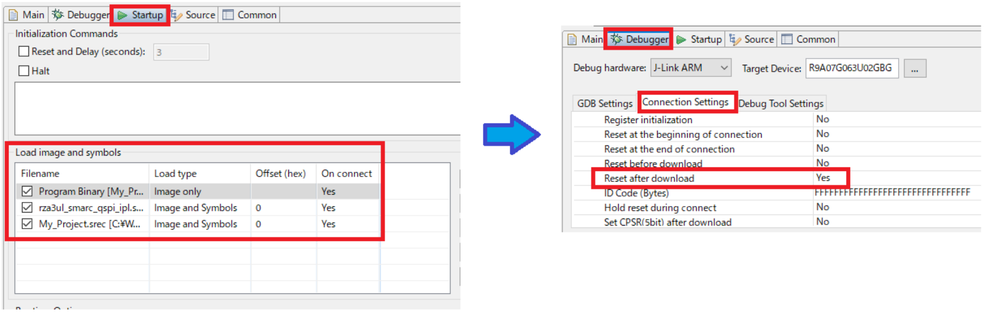
C/C++ Application: Debug/MyProject.elf

Build Configuration: Use Active

○ Enable auto build
○ Disable auto build
● Use workspace settings

Figure 126. Debug Configuration Window

3. Please set load images and set **Reset after download** setting to **Yes,** as shown below:



☐ Reset and Delay (seconds): 3

☐ Halt

	Filename	Load type	Offset (hex)	On connect
☑	Program Binary [My_Pr...	Image only		Yes
☑	rza3ul_smarc_qspi_ipl.s...	Image and Symbols	0	Yes
☑	My_Project.srec [C:¥W...	Image and Symbols	0	Yes

Debug hardware: J-Link ARM   Target Device: R9A07G063U02GBG

Register initialization	No
Reset at the beginning of connection	No
Reset at the end of connection	No
Reset before download	No
Reset after download	Yes
ID Code (Bytes)	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Hold reset during connect	No
Set CPSR(5bit) after download	No

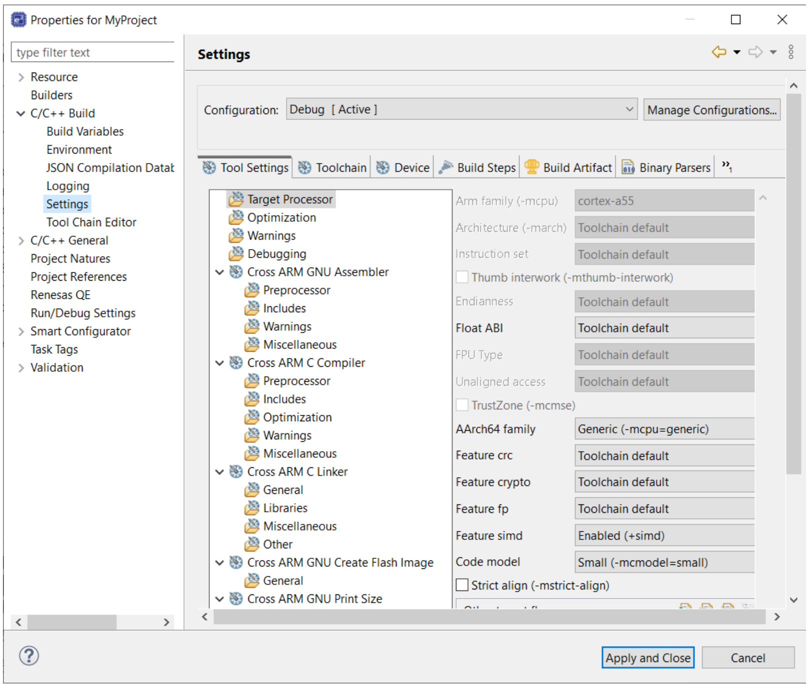
Figure 127. Debug Setting

4. Connect the board to your PC via a standalone Segger J-Link debugger and click **Debug**.

Note: For details on using J-Link and connecting the board to the PC, see 3.2.2.JTAG connection.

5.5 Modifying Toolchain Settings

There are instances where it may be necessary to make changes to the toolchain being used (for example, to change the optimization level of the compiler or add a library to the linker). Such modifications can be made within e^2 studio through the menu **Project > Properties > Settings** when the project is selected. The following screenshot shows the settings dialog for the GNU Arm toolchain. This dialog will look slightly different depending on the toolchain being used.



Configuration: Debug [Active]

Arm family (-mcpu): cortex-a55
Architecture (-march): Toolchain default
Instruction set: Toolchain default
☐ Thumb interwork (-mthumb-interwork)
Endianness: Toolchain default
Float ABI: Toolchain default
FPU Type: Toolchain default
Unaligned access: Toolchain default
☐ TrustZone (-mcmse)
AArch64 family: Generic (-mcpu=generic)
Feature crc: Toolchain default
Feature crypto: Toolchain default
Feature fp: Toolchain default
Feature simd: Enabled (+simd)
Code model: Small (-mcmodel=small)
☐ Strict align (-mstrict-align)

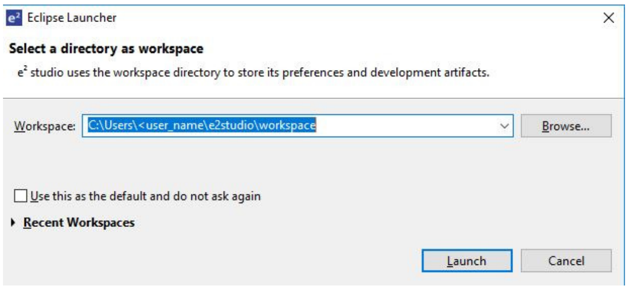
Figure 128. e^2 studio Project toolchain settings

The scope for the settings is project scope, which means that the settings are valid only for the project being modified.

The settings for the linker, which controls the location of the various memory sections, are contained in a script file specific to the device being used. This script file is included in the project when it is created and is found in the created project. (for example, script/rza3ul_smarc_qspi_xip.ld).

5.6 Importing an Existing Project into e^2 studio

1. Launch e^2 studio.
2. Open an existing Workspace to import the project and skip to step d. If the workspace does not exist, proceed with the following steps:
    a. At the end of e^2 studio startup, you will see the Workspace Launcher Dialog box as shown in the following figure.



Select a directory as workspace

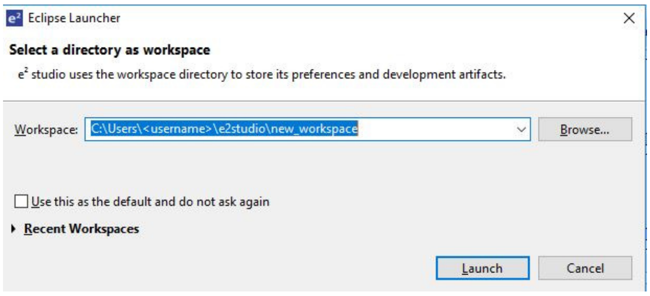
e^2 studio uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\<user_name\e2studio\workspace

☐ Use this as the default and do not ask again

Figure 129. Workspace Launcher dialog

b. Enter a new workspace name in the Workspace Launcher Dialog as shown in the following figure. e^2 studio creates a new workspace with this name.



Select a directory as workspace

e^2 studio uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\<username>\e2studio\new_workspace

☐ Use this as the default and do not ask again

Figure 130. Workspace Launcher dialog - Select Workspace

c. Click [Launch].

d. When the workspace is open, you may see the Welcome Window. Click on the **Workbench** arrow button to proceed past the Welcome Screen, as seen in the following figure.

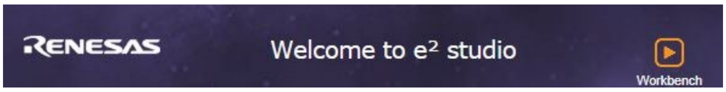


Figure 131. Workbench arrow button

3. You are now in the workspace that you want to import the project into. Click the **File** menu in the menu bar, as shown in the following figure:

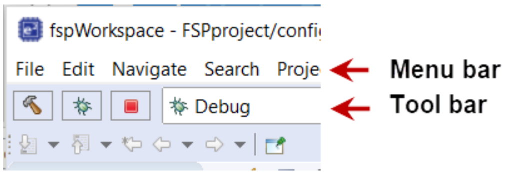
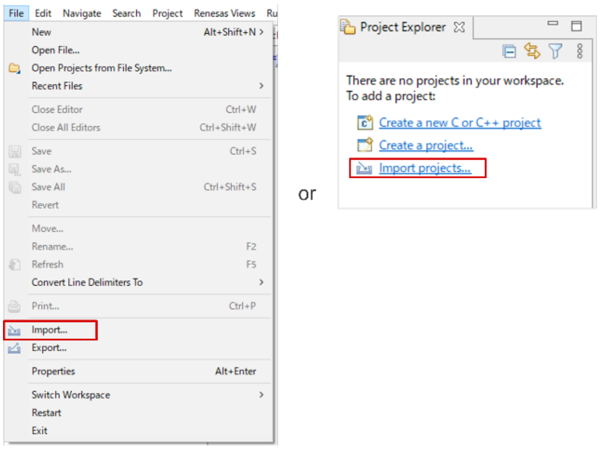


Figure 132. Workbench arrow button

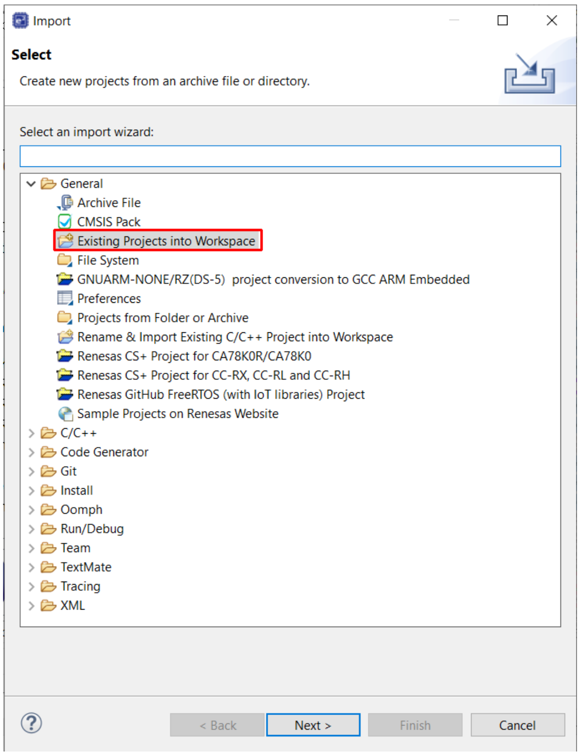
4. Click [Import] on the [File] menu or “Import project” on Project Explorer, as shown in the following figure:



There are no projects in your workspace.
To add a project:

Figure 133. File drop-down menu

5. In the **Import** dialog box, as shown in the following figure, choose the **General** option, then **Existing Projects into Workspace**, to import the project into the current workspace.


Import

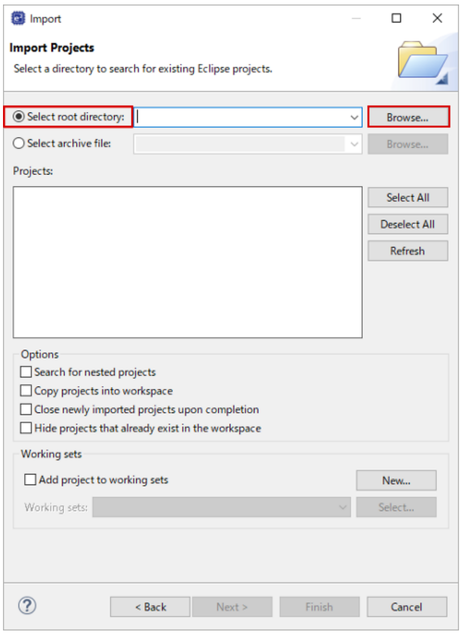
Select

Create new projects from an archive file or directory.

Select an import wizard:

Figure 134. Project Import dialog with "Existing Projects into Workspace" option selected

6. Click [Next >]
7. To import the project, use either **Select archive file** or **Select root directory**.
   First, choose **Select root directory** as shown below:


Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

☐ Search for nested projects

☐ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

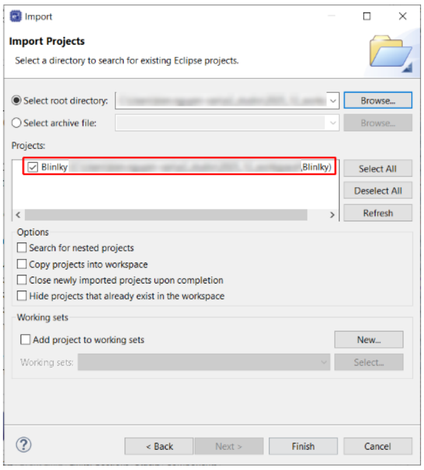
Working sets

☐ Add project to working sets

Working sets:

Figure 135. Import Existing Project dialog 1 - Select root directory

8. Click [Browse...].
9. Choose the directory of the project you would like to import to specify the directory as the **root directory**.
10. Select the project for import.
11. Click [Open].
12. Select the project to import from the list of [Projects:] as shown below:



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: [redacted]

○ Select archive file:

Projects:

☑ Blinky [redacted] Blinky)

Options

☐ Search for nested projects

☐ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

Figure 136. Import Existing Project dialog 2 – Select the project to be imported

13. Click [Finish] to import the project.

6. Notes on development

6.1 Getting the USB Hub to work with USBX

When using a USB Hub with USBX, please follow the pro shown below:

1. Copy **rz/microsoft/azure-rtos/usbx/common/core/inc/ux_user_sample.h** to the directory **rz/fsp/src/rm_usbx_port** and rename it to **ux_user.h**.
2. Add the definitions listed below to ux_user.h.

```
#define UX_MAX_CLASS_DRIVER    3
#define UX_MAX_ED              80
#define UX_MAX_TD              128
#define UX_MAX_ISO_TD          128
```

3. Add the following definition to **rz_azurertos_sample/rz/fsp/src/rm_usbx_port/ux_port.h**.

```
#define UX_INCLUDE_USER_DEFINE_FILE
```

Revision History

Rev.	Date	Description	
		Page	**Summary**
4.0.0	Jan.30.26	4, 17 to 32, 41 to 44, 65 to 81, 87	Updated the description and figure based on the latest development environment.
3.6.0	Sep.19 2025	5	Added explanation for “EK-RZ/A3M NAND Boot (Exec with DDR SDRAM)”, which is newly supported.
		34	Added a description for switching between NOR boot and NAND boot on the RZ/A3M.
		35	Added content to address the issue of NAND Flash performance degradation.
		47	Updated the debug prerequisites for using NAND Flash memory.
		53 to 62	Added debug steps for the NAND Flash memory environment.
3.5.0	May.08.2025	1, 5	Added RZ/A3M as a supported device.
		22, 27, 30, 46 to 50	Updated the description and figure based on the latest development environment.
		47	Removed the debug step that sets Flash Bus Type and Flash Memory Type since it is not required in this version.
		32 to 37, 39	Added the information of EK-RZ/A3M.
3.4.0	Feb.21.2025	6 to 31, 37 to 39, 43, 48, 52, 54, 56, 57	Updated the description and figure based on the latest development environment.
3.3.0	Dec.20.2024	6 to 31, 37 to 39, 52, 54, 56	Updated the description and figure based on the latest development environment.
3.2.0	Sep.30.2024	22 to 31, 37, 39	Updated the description and figure based on the latest development environment.
3.1.0	Jul.31.2024	6 to 29, 34, 37, 44 to 45, 50, 52, 54 to 55	Updated the description and figure based on the latest development environment.
		40 to 42	Corrected the debug step when using the OCTAL Edition of the SMARC EVK board.
3.0.0	Apr.26.2024	6 to 29, 32 to 36, 47 to 53	Updated the description and figure based on the latest development environment.
2.0.2	Feb.29.2024	15, 36, 72, 73	Added 2.2 to install the Arm GNU toolchain. Added Steps 4 and 5 to 4.5.2. Removed 6.1 Unexpected termination of GDB connection. Removed 6.3 to describe the way to fix the building error.
2.0.1	Sep.30.2023	-	Updated the versions.
2.0.0	Jun.30.2023	63, 67	Added 6.3 to describe the way to fix the building error. Added 6.4 to describe the way to use USBX.
1.21	Apr.07.2023	22, 23	Added 2.2.3 to describe the way to install FSP with the zipped Packs.
1.20	Dec.26.2022	-	Added the instructions for installing FSP using Platform Installer.
1.10	Sep.30.2022	-	Added the info on “RZ/A3UL Evaluation Board Kit Octal-SPI Edition.
1.00	Jul.28.2022	-	First edition issued

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/.