RZ/V2L, RZ/V2H

Getting Started with Flexible Software Package

Introduction

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

Target Device

RZ/V2L, RZ/V2H

Contents

1. Introduction

1.1 Overview

This application note describes how to use the Renesas Flexible Software Package (FSP) running on the Cortex®-M33 (hereinafter referred to as CM33) and Cortex®-R8(*)(hereinafter referred to as CR8) incorporated on RZ/V2L, and RZ/V2H.
(*: CR8 is for RZ/V2H only.)

1.2 Introduction to FSP

1.2.1 Purpose

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

1.2.2 e² studio IDE

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

1.3 Limitations

1.3.1 Peripherals and pins assignment

RZ/V2L has a multi-core configuration of Cortex-A55 (hereinafter referred to as CA55) and CM33. Also, RZ/V2H has a multi-core configuration of CA55, CM33 and CR8. It is possible to use each peripheral and GPIO from each core. This package provides drivers for each peripheral for CM33 and CR8, but each driver can operate on the assumption that it is not used in CA55.

1.3.2 RAM Initialization

Initialization of DDR SDRAM is always carried out in CA55 bootstrap regardless of the selection of boot CPU, meanwhile Internal SRAM is initialized in the bootstrap of boot CPU.

2. Starting Development Introduction

2.1 e² studio setup

2.1.1 What is e² studio?

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

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

2.1.2 e² studio Prerequisites

2.1.2.1 Obtaining an RZ MPU Kit

To develop applications with RZ/V FSP, start with each Evaluation Board Kit.
Start-up guide of RZ/V2L Evaluation Board Kit is available at RZ/V2L SMARC EVK Start-up Guide.
Also, Getting Started of RZ/V2H Evaluation Board Kit is available at RZ/V2H EVK Getting Started.

2.1.2.2 PC Requirements

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

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

2.1.2.3 Licensing

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

2.1.3 e² studio installation for Windows PC

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

2.1.3.1 Download

The latest e² studio IDE installer package can be downloaded from Renesas website for free. Please check detailed information from: https://www.renesas.com/e2studio. Note that user has to login to the Renesas account (in MyRenesas page) for the software download.

2.1.3.2 Installation of e² studio IDE

1. Double-click on e² studio installer to invoke the e² studio installation wizard page. First, you need to select Install Type. In this material, it is expected that Custom Install is selected. Then, click [Next >] to continue.

Note: If e² studio was installed in your PC, the option to modify, remove the existing version or install e² studio to a different location will be displayed

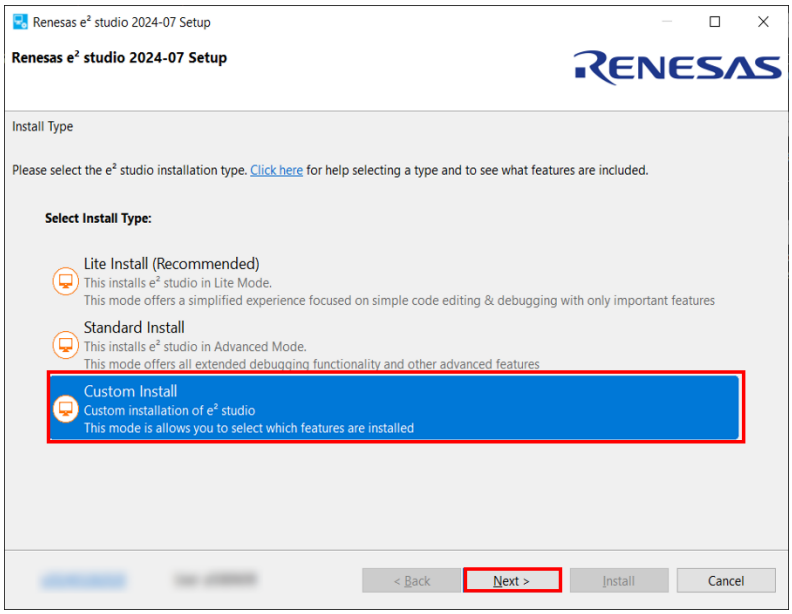


Figure 1: e² studio installation wizard

2. Welcome page
   User can change the install folder by clicking [Change…].Click [Next] to continue.

Note1: If you would like to have multiple versions of e² studio, please specify new folder here.
Note2: Multi-byte characters cannot be used for e² studio installation folder name.

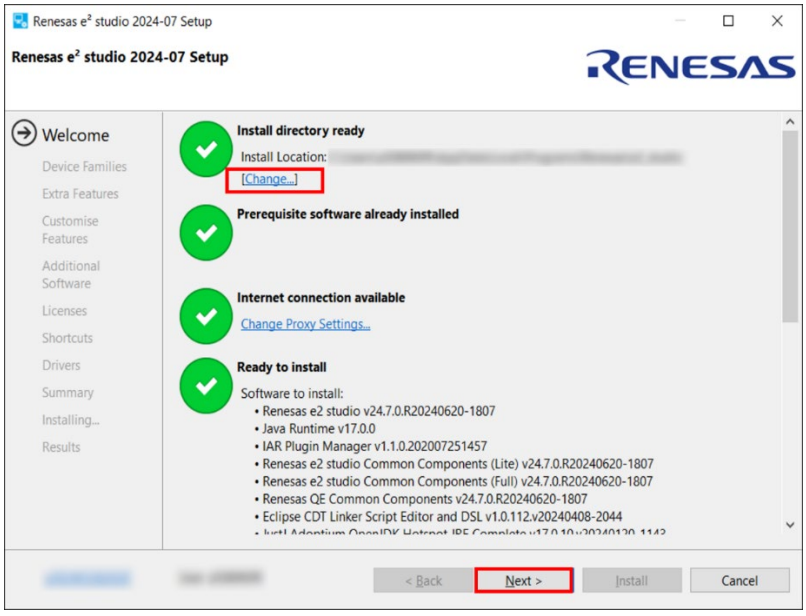


Figure 2: Installation of e² studio – Welcome page

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

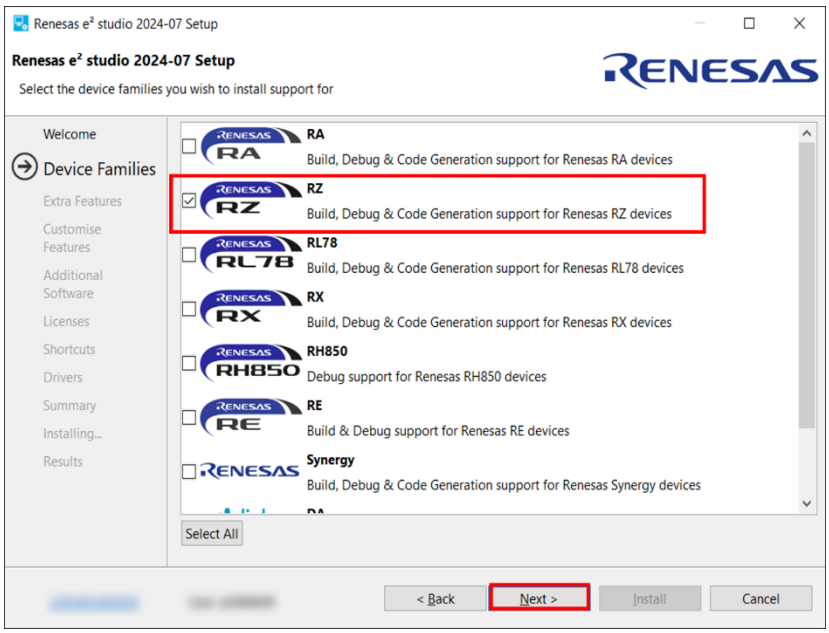


Figure 3: Installation of e² studio – Device Families

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.
   Click the [Next] button to continue.

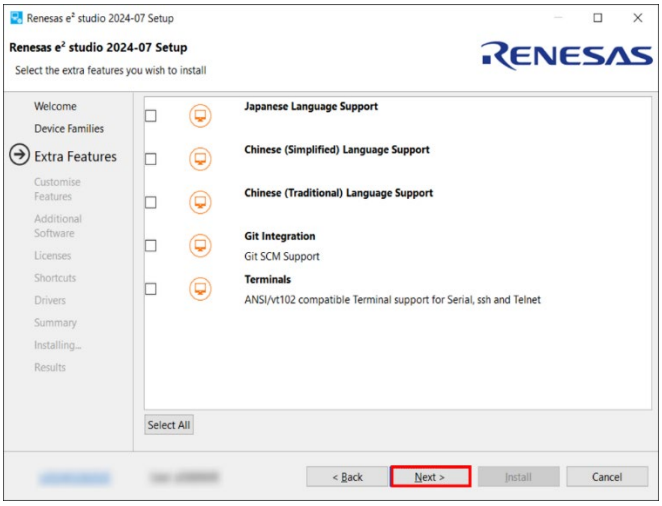


Figure 4: Installation of e² studio – Extra Features

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

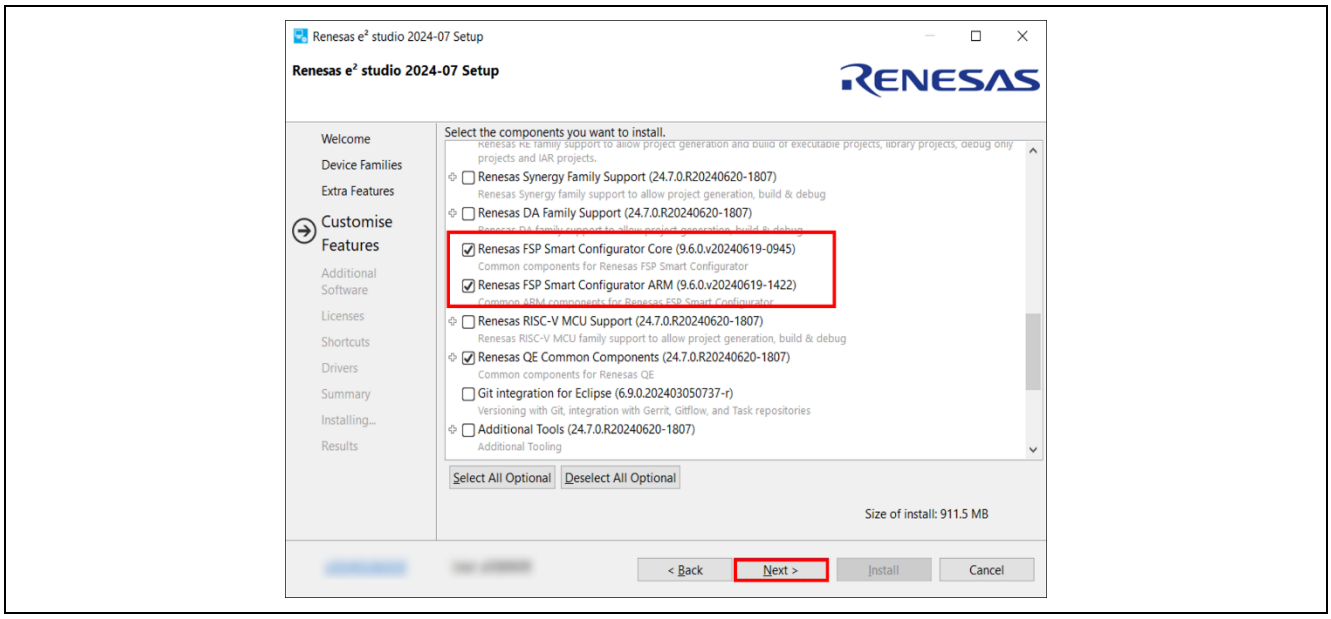


Figure 5: Installation of e² studio – Features

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

   GNU ARM Embedded 12.2-Rel1

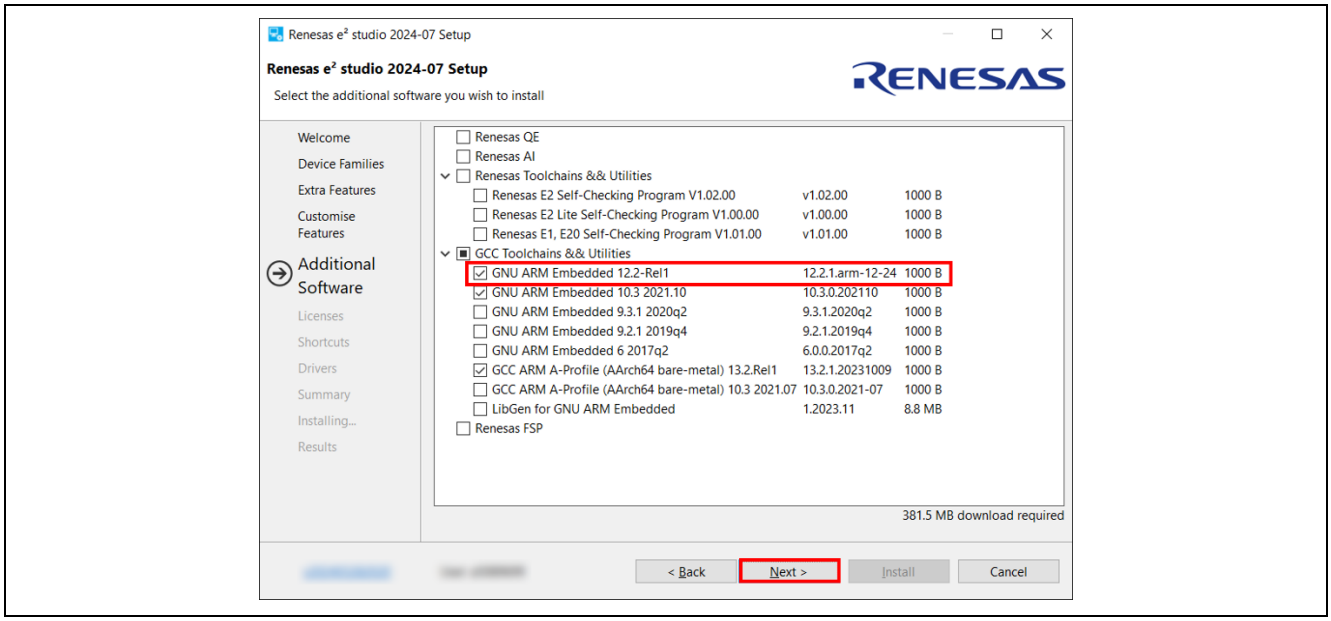


Figure 6: Installation of e² studio – Additional Software

For more details on the installation of Additional Software, please see the section 2.1.4.

7. License Agreement
   Read and accept the software license agreement. Click the [Next] button. Please note that user must accept the license agreement, otherwise installation cannot be continued.

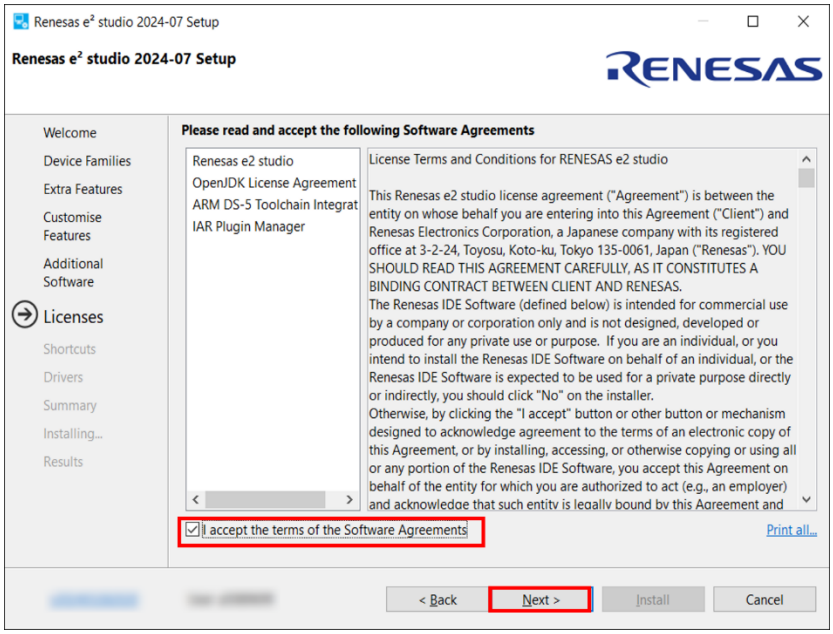


Figure 7: Installation of e² studio – Licenses

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

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

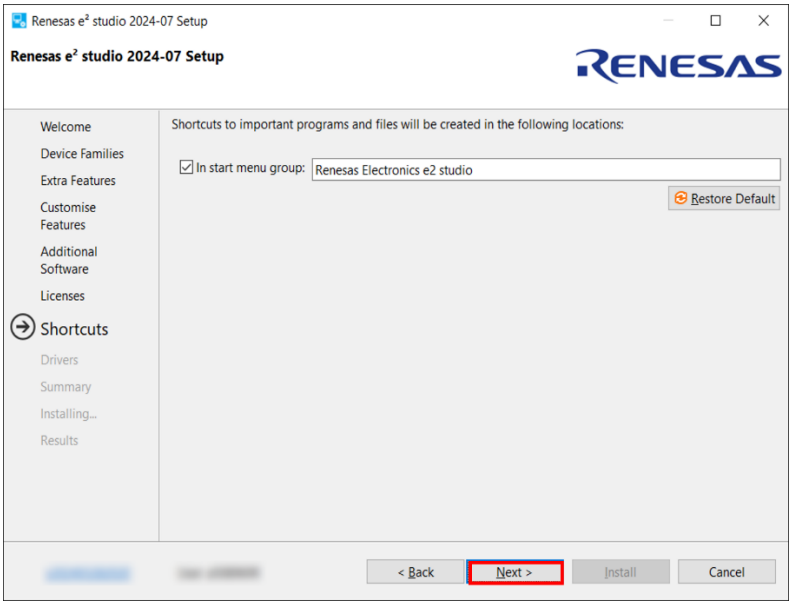


Figure 8: Installation of e² studio – Shortcuts

9. Summary
   Components list to be installed is shown. Please confirm the contents and click the [Install] button to install the Renesas e² studio IDE.

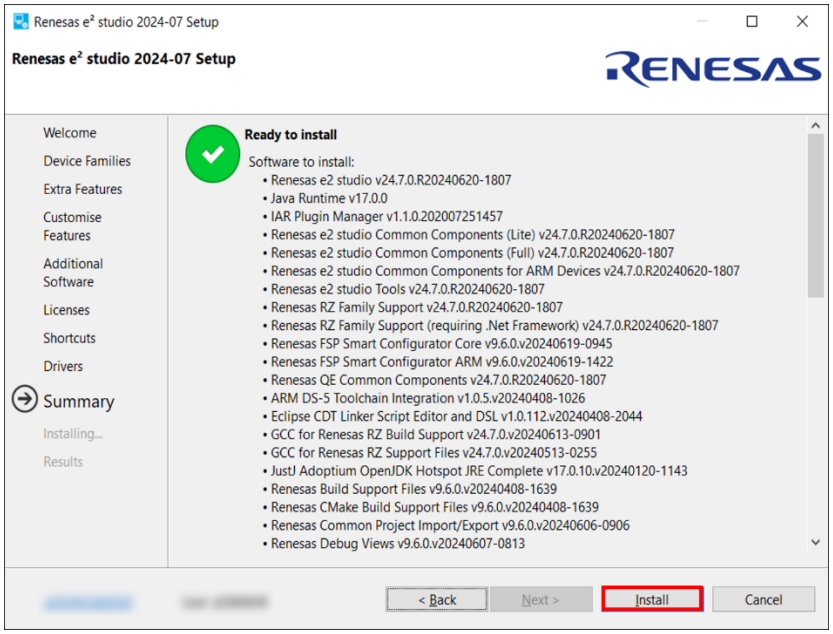


Figure 9: Installation of e² studio – Summary

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

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

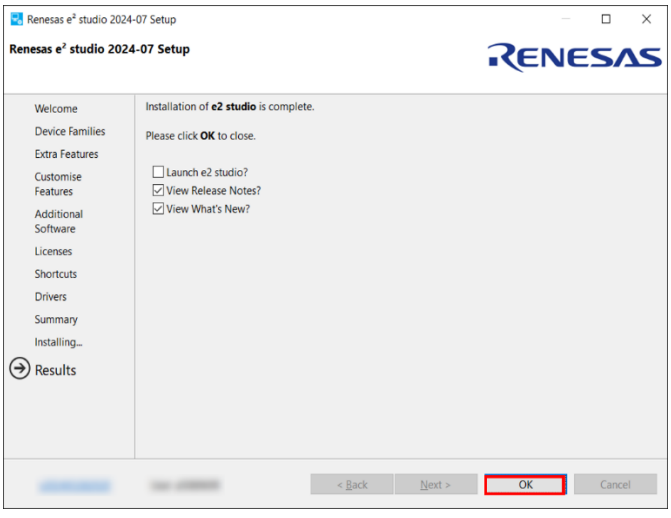


Figure 10: Summary Page

2.1.3.3 Installation of Additional Software

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

GNU ARM Embedded 12.2-Rel1.

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

- **GNU ARM Embedded Toolchain 12.2-Rel1**
  If it was selected in the Additional Software pane of e² studio, you will see the installation wizard for the GNU ARM Embedded Toolchain during the installation process.

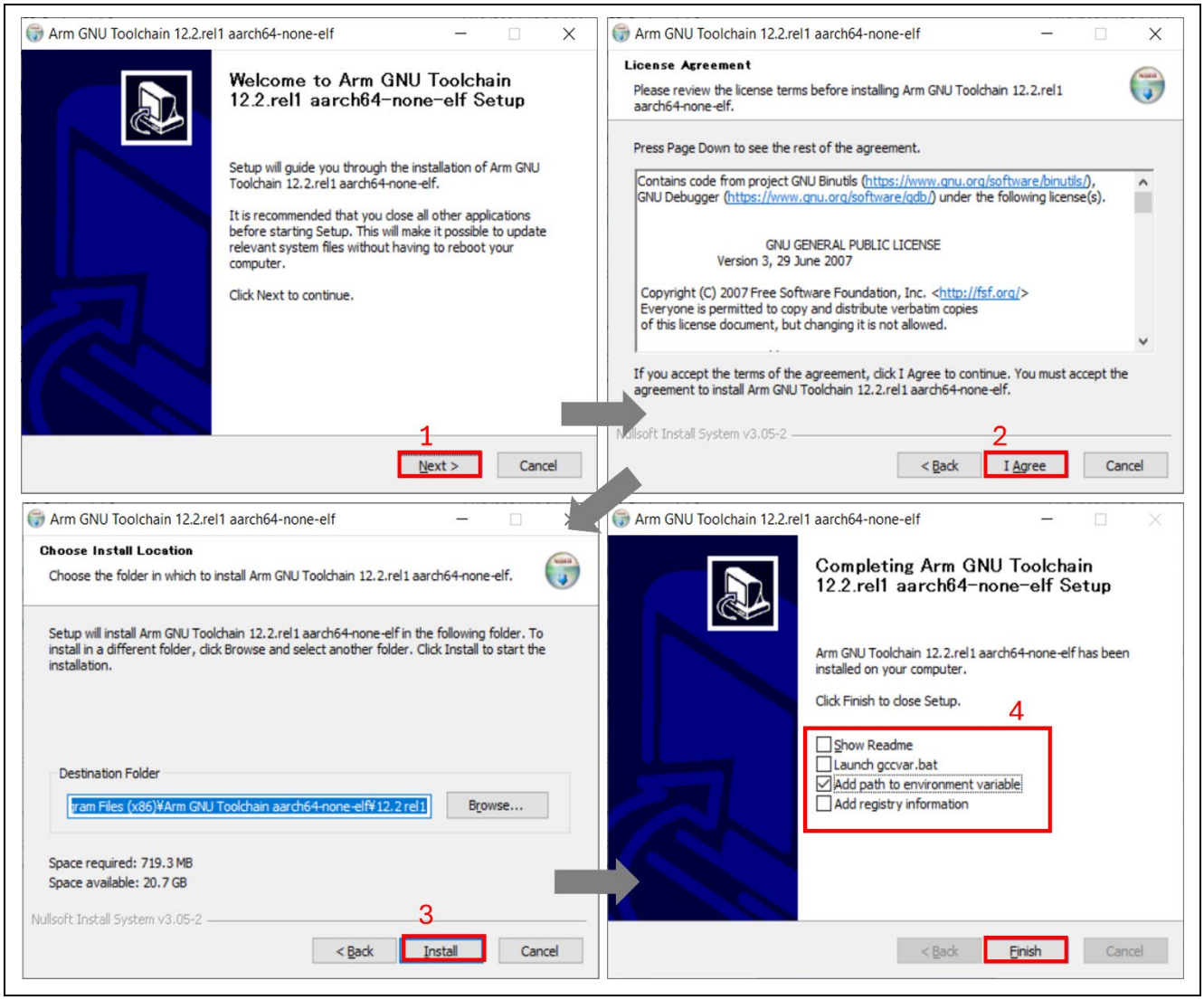


Figure 11: Installation of GNU ARM Embedded Toolchain

2.1.4 e² studio installation for Linux PC

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

2.1.4.1 Download

The following files are required to download before installation.

SEGGER J-Link driver
Please download the driver V7.96e or later from:
https://www.segger.com/downloads/jlink/JLink_Linux_V796e_x86_64.deb

e² studio IDE installer
e2 studio IDE installer package can be downloaded from Renesas website for free. Please check 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 is provided for example purpose only.

- **SEGGER J-Link driver**
  1. Open a terminal window and enter below commands.
     sudo dpkg -i JLink_Linux_V796e_x86_64.deb

     (If the previous install fails with unmet dependencies, retry it as follows)
     sudo apt-get -f install
     sudo dpkg -i JLink_Linux_V796e_x86_64.deb

- **e² studio IDE**
  1. Run the e² studio IDE Installer. (Before running the installer, check the execution permission of the installer.)
     ./e2studio_installer-2024-07_linux_host.run
  2. **Welcome page**
     User needs to select Install Type as shown below. In this material, it is expected that Custom Install is selected. Then, click [Next >] to continue.

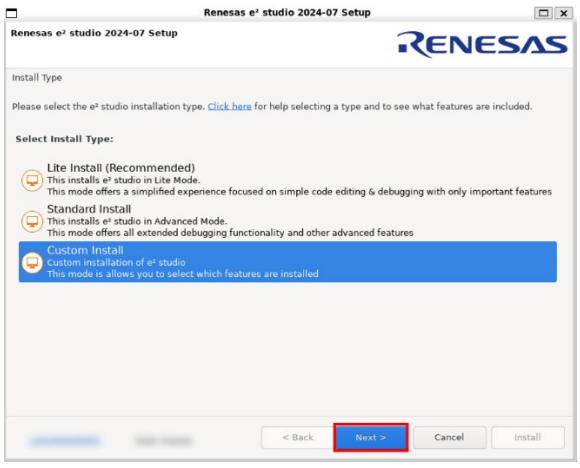


Figure 14: Installation of e² studio – Welcome page

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

Note1: If you would like to have multiple versions of e² studio, please specify new folder here.
Note2: Multi-byte characters cannot be used for e² studio installation folder name.

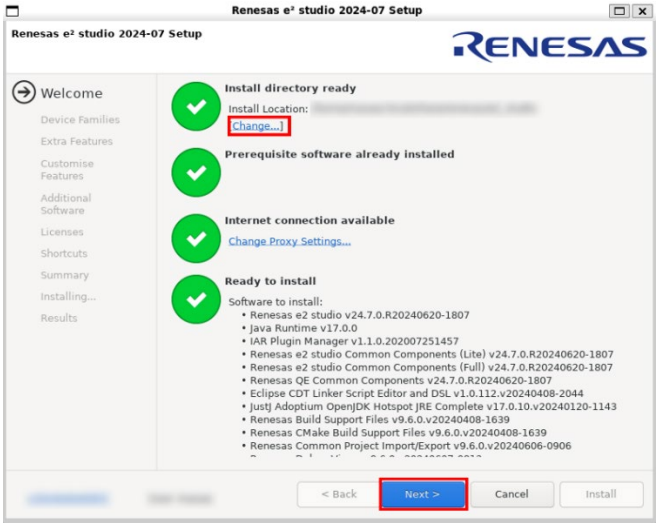


Figure 15: Installation of e² studio – Welcome page

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

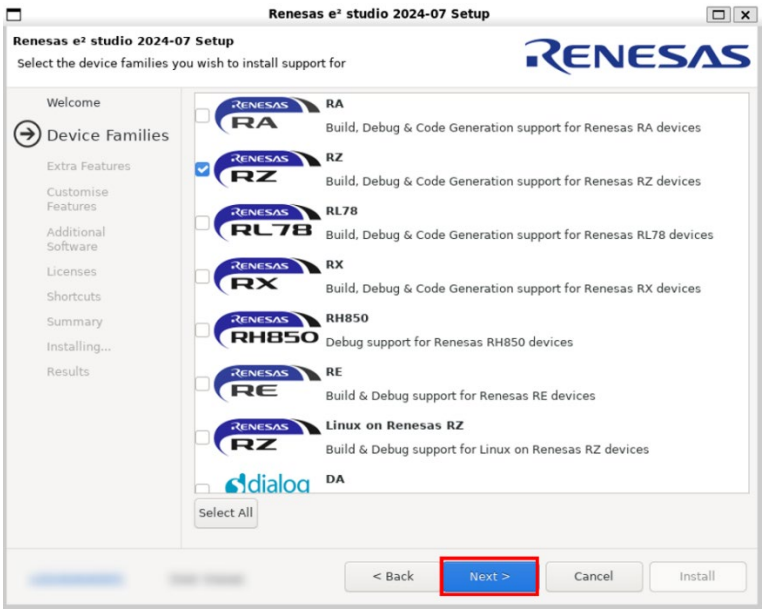


Figure 16: Installation of e² studio – Device Families

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

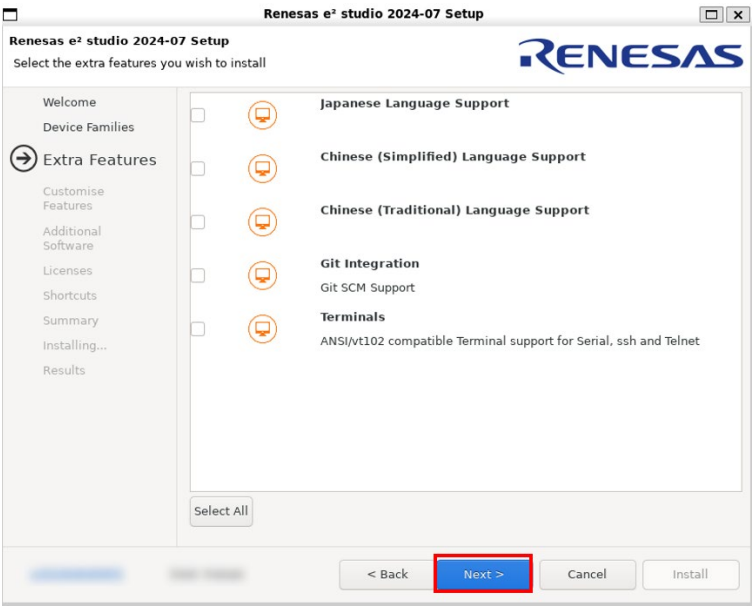


Figure 17: Installation of e² studio – Extra Features

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

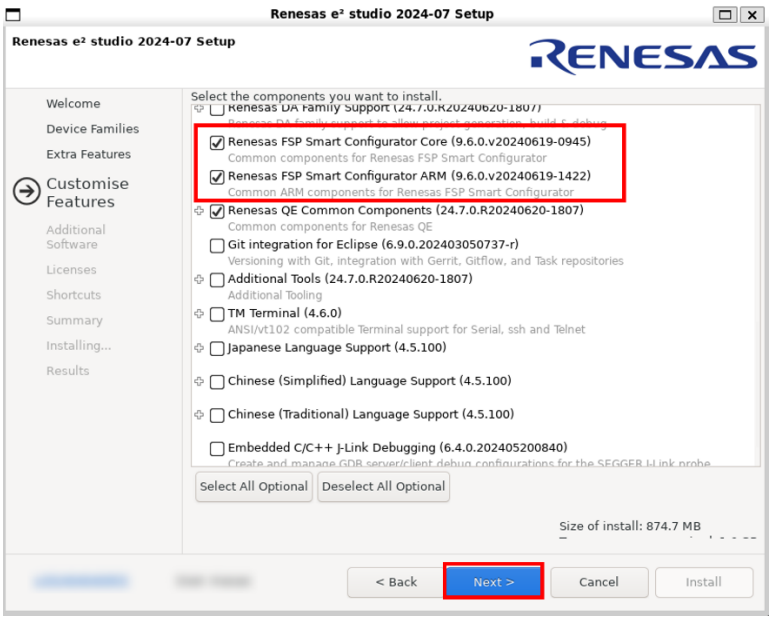


Figure 18: Installation of e² studio – Features

7. **Additional Software**
   Select additional software (i.e., compilers, utilities, QE…) to be installed. Be sure to select the “GNU ARM Embedded 12.2-Rel1” and click [Next >] to continue.

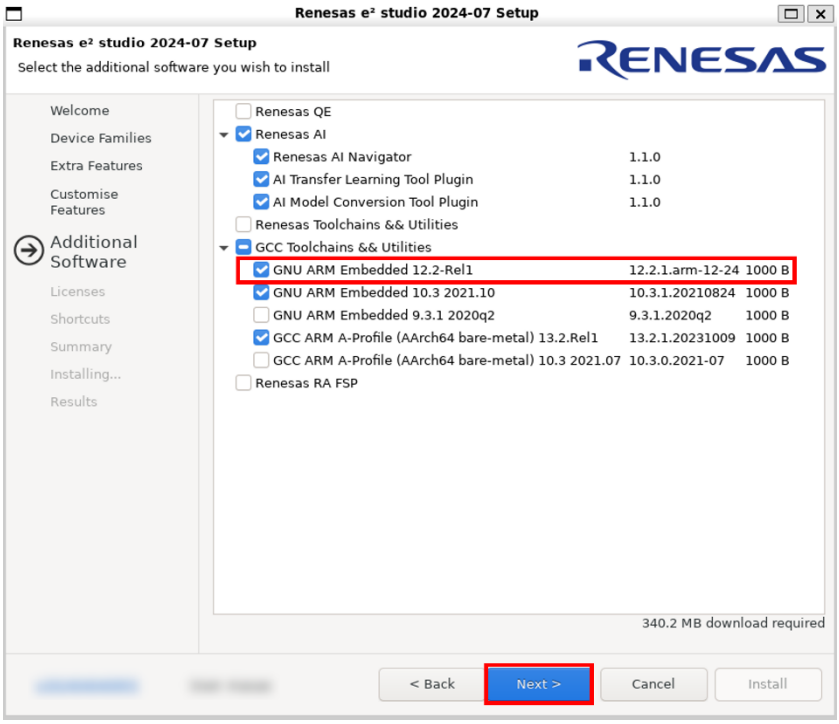


Figure 19: Installation of e² studio – Additional Software

8. **License Agreement**
   Read and accept the software license agreement. Click the [Next] button. Please note that user must accept the license agreement, otherwise installation cannot be continued.

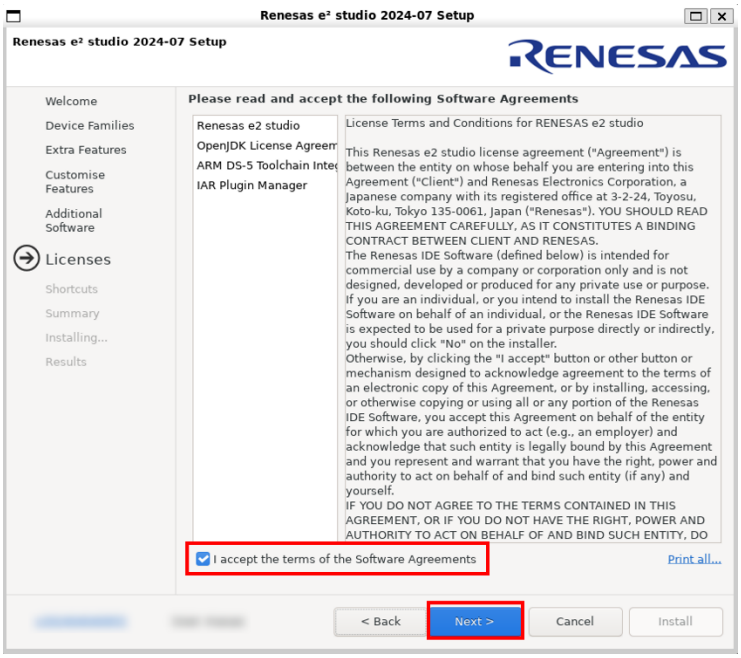


Figure 20: Installation of e² studio – Licenses

9. **Shortcuts**
   Select shortcut name for start menu and click [Next] button to continue.

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

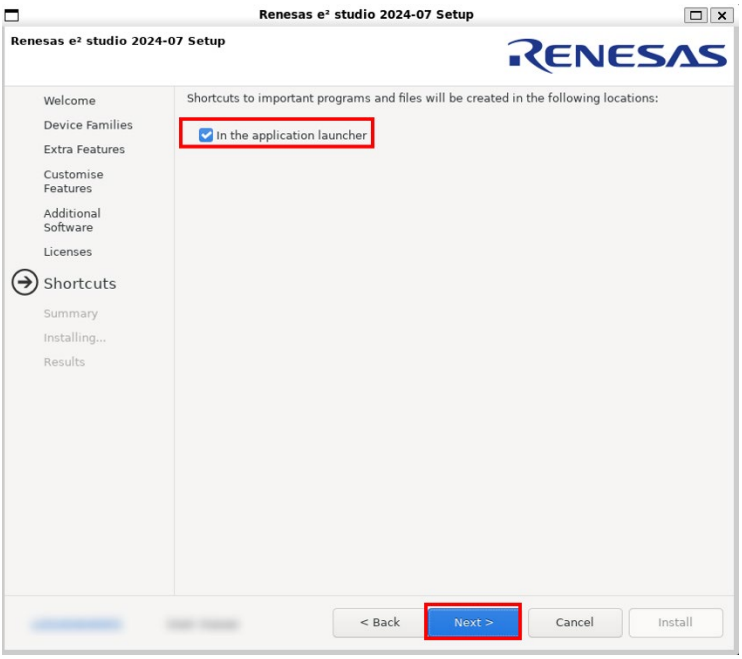


Figure 21: Installation of e² studio – Shortcuts

10. Summary
    Components list to be installed is shown. Please confirm the contents and click the [Install] button to install the Renesas e² studio IDE.

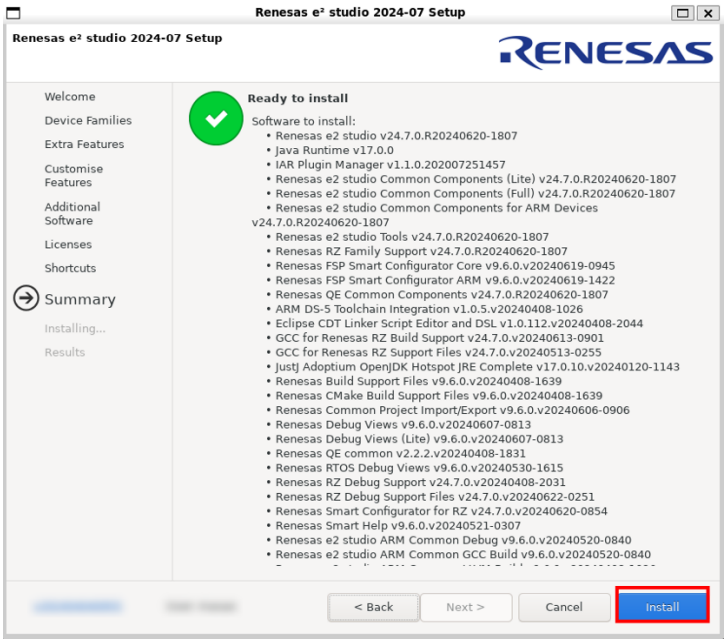


Figure 22: Installation of e² studio – Summary

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

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

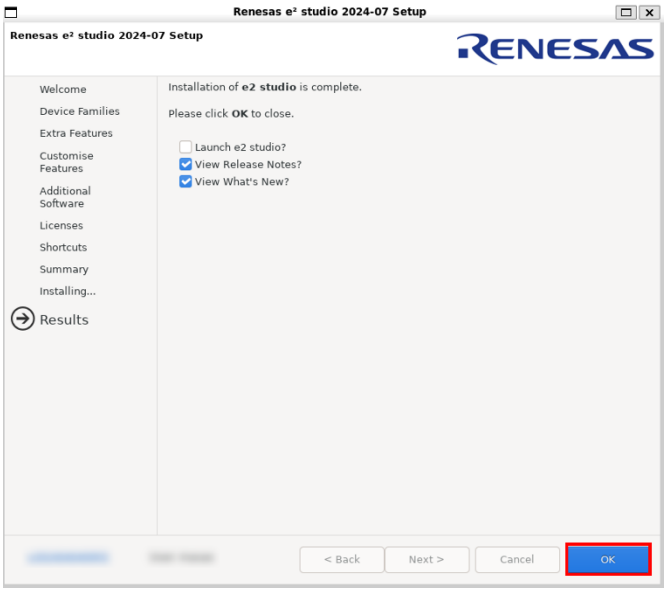


Figure 23: Summary Page

2.2 FSP setup

2.2.1 Installation of FSP using Package Installer

Package Installer **RZV_FSP_Packs_v2.0.1.exe** is showcased at here. This section describes the procedure for installation. Note that it's for Windows Host PC only.

1. Quit e2 studio.
2. Invoke **RZV_FSP_Packs_v2.0.1.exe**.
3. Click [Next >] to start the installation.

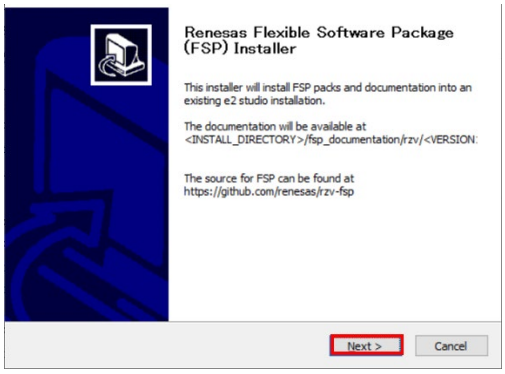


Figure 24: FSP Package Installer

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

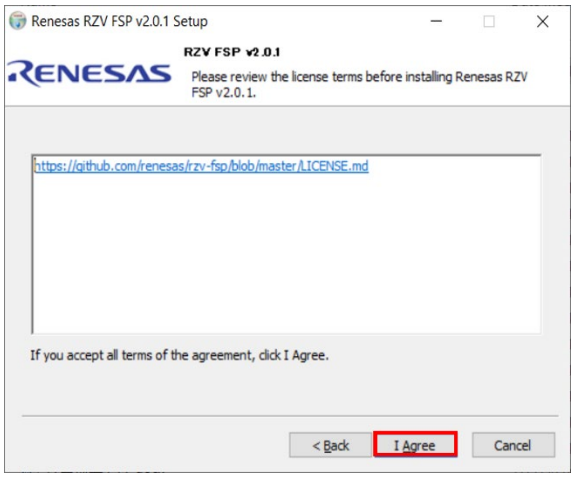


Figure 25: FSP License Term

5. Specify e2 studio installation folder (e.g., C:\Renesas\e2studio) and click [Install].

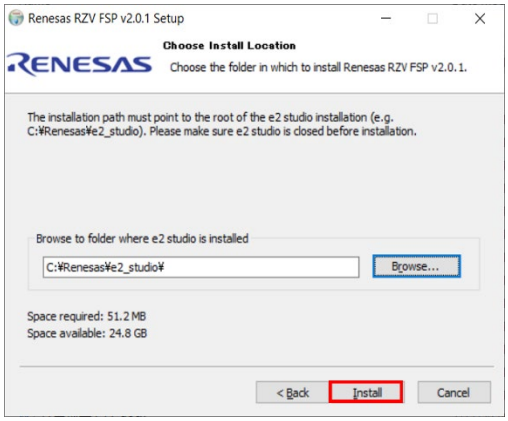


Figure 26: Browse to the folder where e2 studio is installed

6. Click [Finish] to complete the installation.

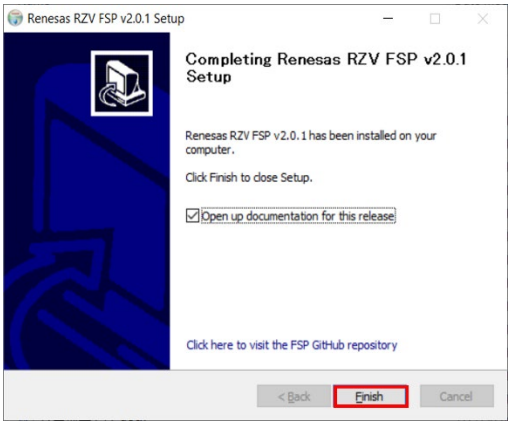


Figure 27: Completion of FSP Installation

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.

2.2.2 Installation of FSP Pack using Package Zip file

No package installer is available for Linux Host PC. Thus, you need to install FSP with the zip file **RZV_FSP_Packs_v2.0.1.zip**. This section describes the procedure for installation.

1. Download RZV_FSP_Packs_v2.0.1.zip from here.
2. Extract the zip file to e2 studio installation directory. If it's successfully extracted, **rz_fsp/rzv/packs** should be placed at **<e2 stduio installation directory>/Internal/projectgen**.

Figure 28: FSP Packs on e2studio installation directory

3. At the 1st invocation of e2 studio after the extraction, FSP should be automatically installed.
4. You can check if the installation is successfully done by the procedure below:
   - Click **Help** > **CMSIS Packs Management** > **Renesas RZ/V**

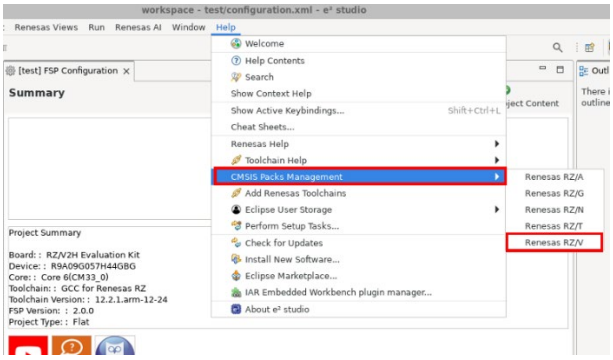


Figure 29: CMSIS Packs Management (1)

- If FSP is successfully installed, 2.0.1 should be listed under FSP as shown below:

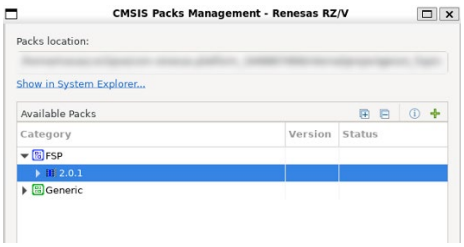


Figure 30: CMSIS Packs Management (2)

3. Set up an SMARC EVK

3.1 Set up an RZ/V2L SMARC EVK

Below is an example of a typical system configuration.

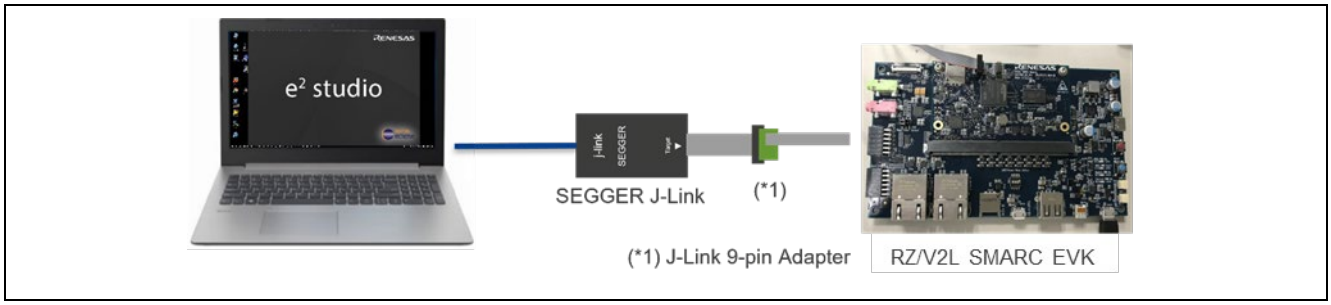


Figure 31: System Configuration Example – RZ/V2L SMARC EVK

3.1.1 Supported Emulator

- SEGGER J-Link

For details on SEGGER J-Link, please see J-Link Debug Probes by SEGGER – the Embedded Experts.

3.1.2 Board Setup

3.1.2.1 Boot MODE

To set the board to Boot mode 3(QSPI Boot(1.8V) Mode), set the SW11 as below.

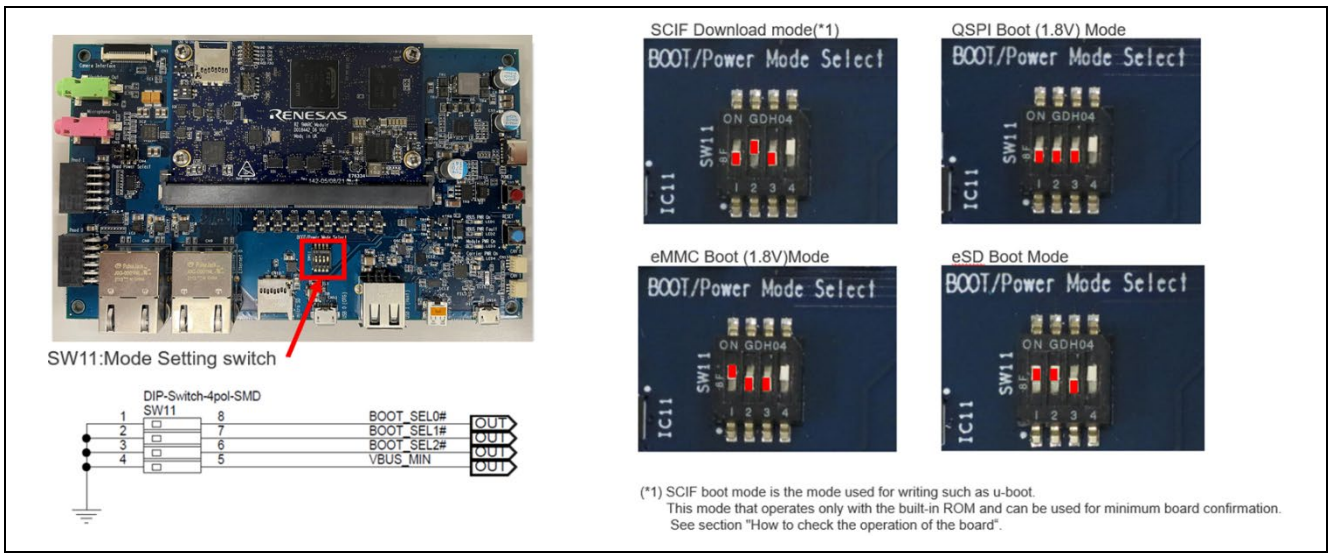


Figure 32: Boot MODE

3.1.2.2 JTAG connection

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

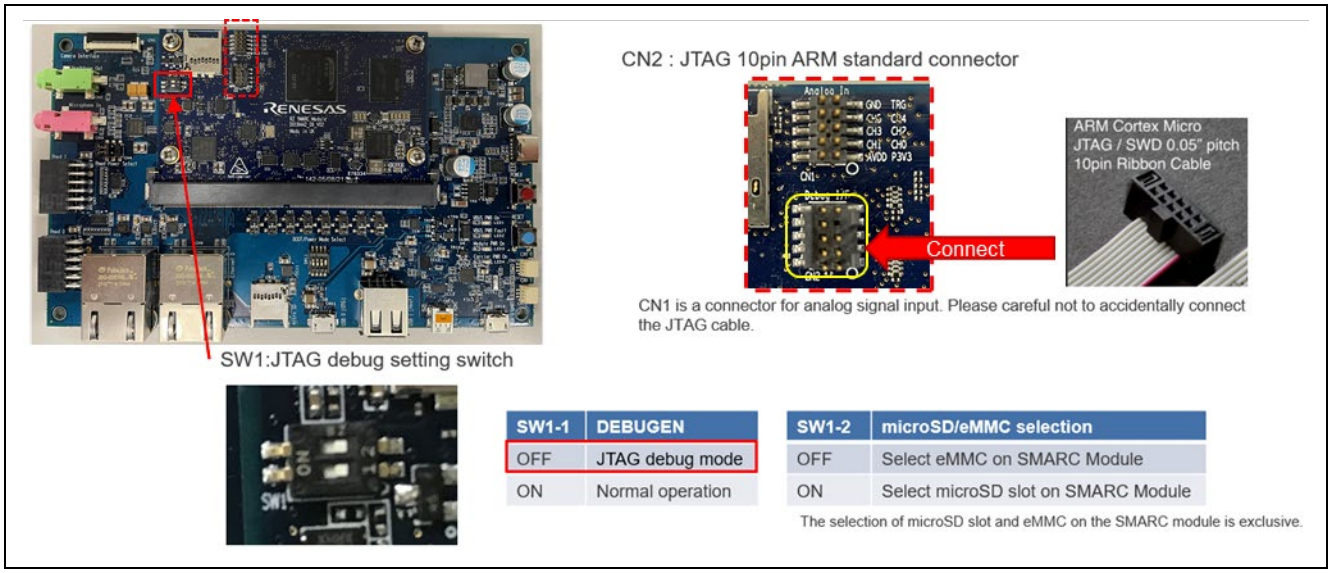


Figure 33: JTAG connection

Please note that RZ/V2L SMARC EVK has CoreSight 10 connector and therefore, the following adapter must be needed to connect Segger J-Link.

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

3.1.2.3 Debug Serial (console output)

Debug serial uses CN14. The baud rate is 115200bps.

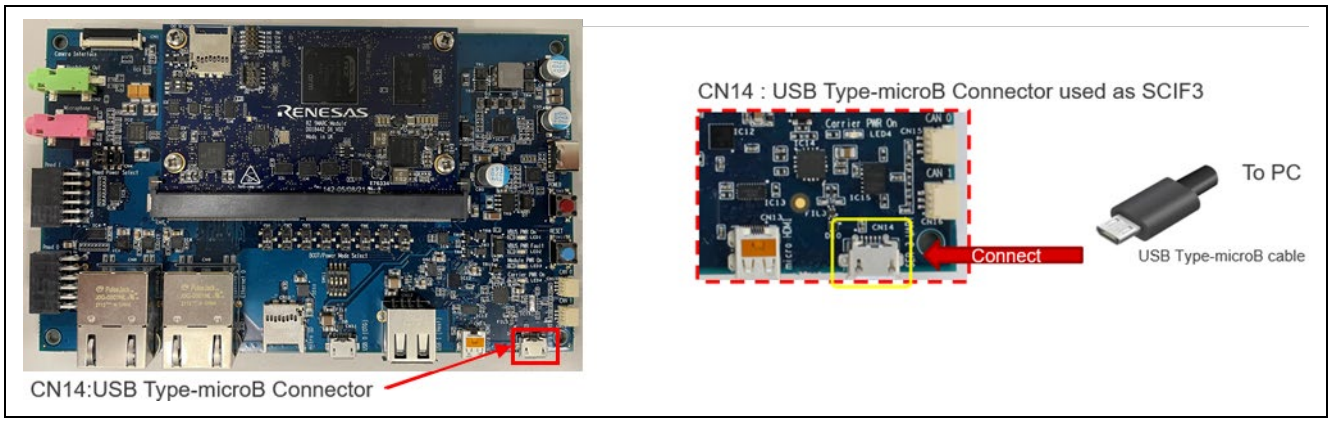


Figure 34: Debug Serial(console output)

3.1.2.4 Power Supply

Here are the 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)

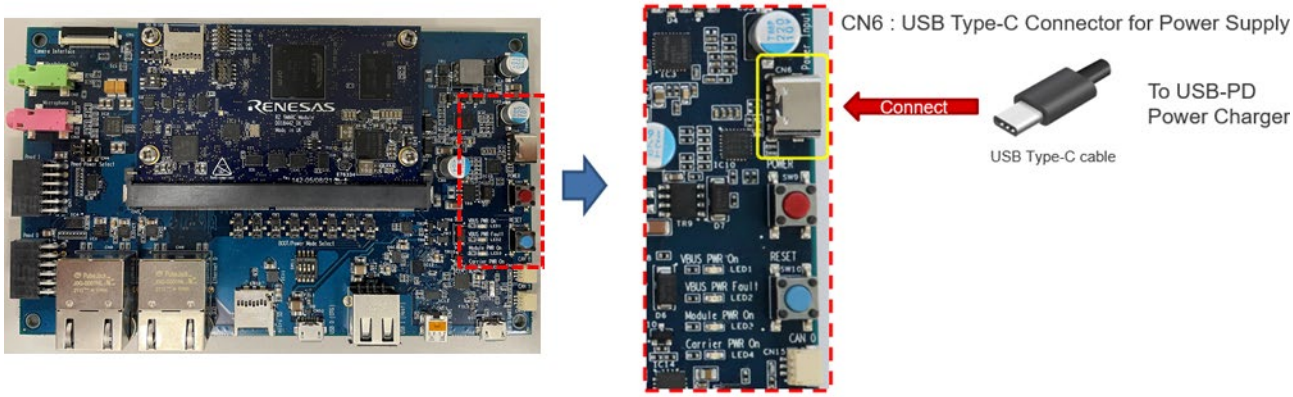


Figure 35: Power Supply

Connect USB-PD Power Charger to USB Type-C Connector. Then LED1(VBUS PWR On) and LED3 (Module PWR On) lights up. Press SW9 to turn on the power. Then LED4(Carrier PWR On) lights up.

Note: When turn on the power, press and hold the power button for 1 second.
When turn off the power, press and hold the power button for 2 seconds

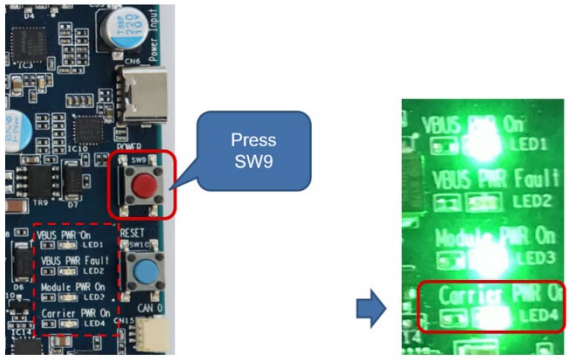


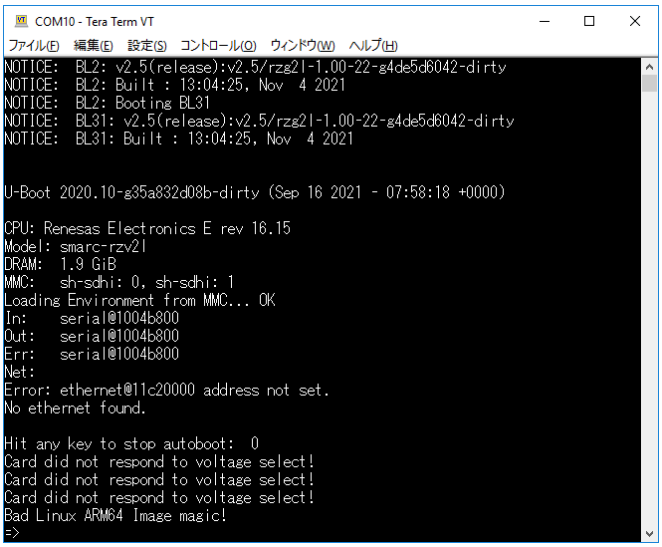
Figure 36: LED Status after Turning on EVK

3.1.2.5 How to check the operation of the board

First, check the board for problems. There are two ways to do this. Please check with either.

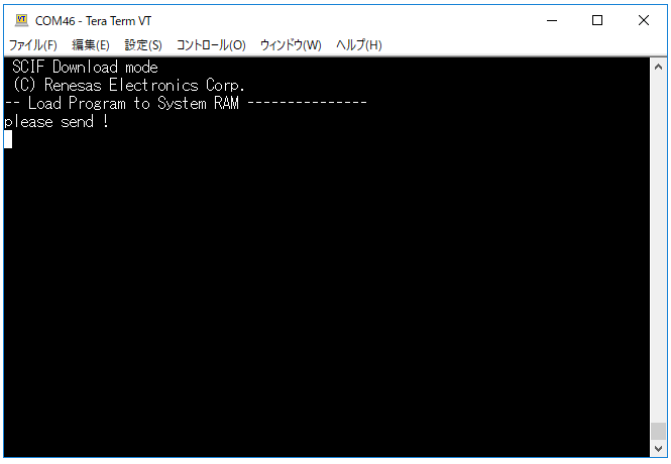
BOOT MODE: QSPI Boot(1.8V) Mode

If u-boot is written to the serial flash, when the power is turned on, the following will be output to the console(CN14).



BOOT MODE: SCIF Download Mode

When the power is turned on, the following will be output to the console (CN14).



3.2 Set up an RZ/V2H EVK

Below is an example of a typical system configuration.

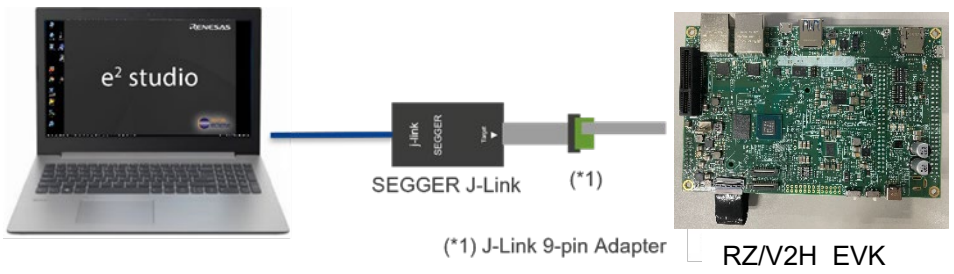


Figure 37: System Configuration Example – RZ/V2L SMARC EVK

3.2.1 Supported Emulator

- SEGGER J-Link

For details on SEGGER J-Link, please see J-Link Debug Probes by SEGGER – the Embedded Experts.

3.2.2 Board Setup

3.2.2.1 Boot MODE

Set the boot mode using the DSW1 shown in the figure below.

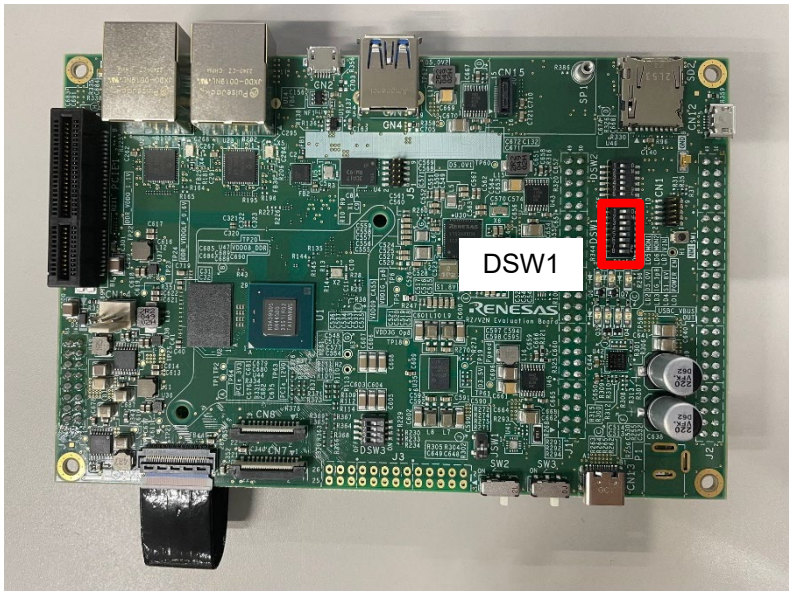


Figure 38: Boot MODE

Select the boot mode with the following settings.

Switch No.	Function
1	Select the cold boot CPU OFF: CM33 / ON: CA55
2 3	Input the CA55 frequency at the CA55 cold boot. [SW2 : SW3] = [OFF : OFF] : 1.6GHz = [OFF : ON] : 1.7GHz [default] = [ON : OFF] : 1.1GHz = [ON : ON] : 1.5GHz
4 5	Input the boot mode select signal. [SW4 : SW5] = [OFF : OFF] : xSPI = [OFF : ON] : SCIF = [ON : OFF] : SD = [ON : ON] : eMMC
6	OFF: SSCG OFF / ON: SSCG ON
7	OFF: Normal mode / ON: Debug mode
8	Fix OFF

3.2.2.2 JTAG connection

When connecting to JTAG, you must set the DSW1 Switch No.7 settings as ON.

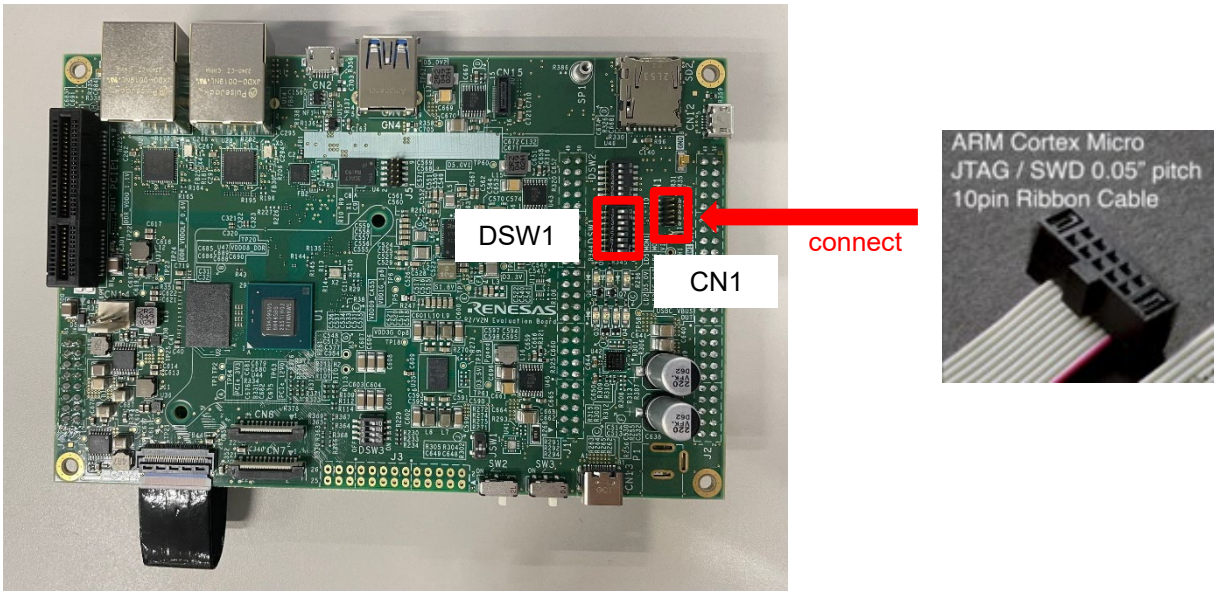


Figure 39: JTAG connection

Please note that RZ/V2H EVK has CoreSight 10 connector and therefore, the following adapter must be needed to connect Segger J-Link.

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

3.2.2.3 Debug Serial (console output)

Debug serial uses CN12. The baud rate is 115200bps.

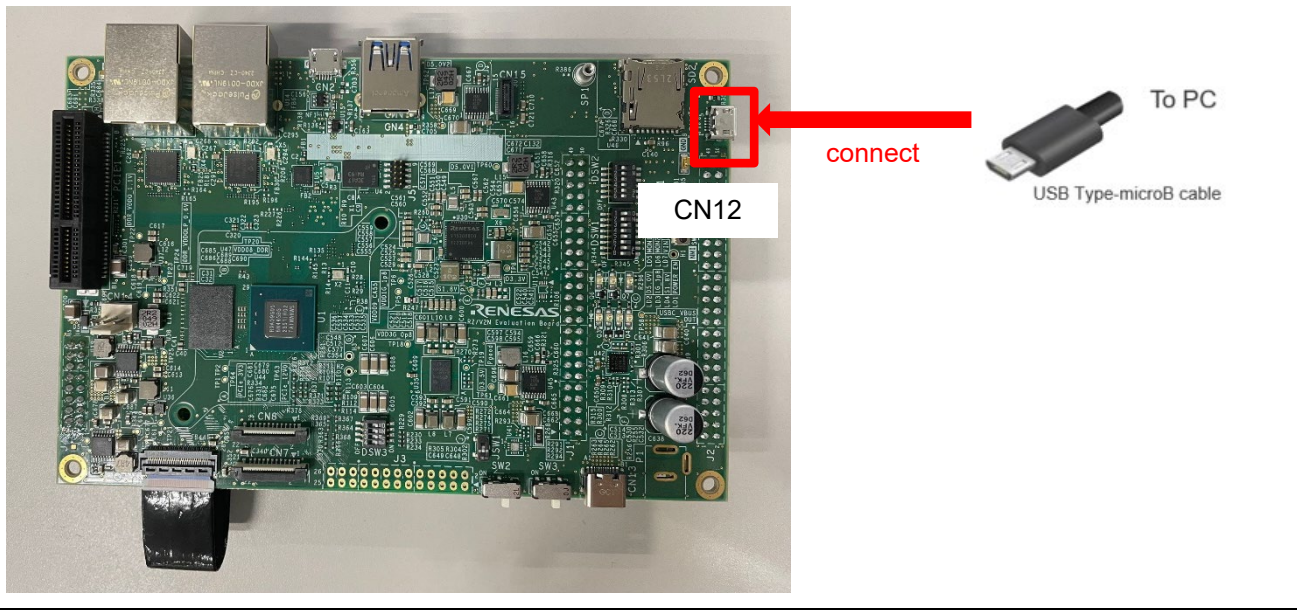


Figure 40: Debug Serial(console output)

3.2.2.4 Power Supply

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

- USB PD Charger MAGCUBE PD 100W (AOC-C005) (manufactured by AOHI)
- USB Type-C cable included with MAGCUBE PD 100W (AOC-C005) (manufactured by AOHI)

Check that the power slide switch SW2 and SW3 are turned OFF.

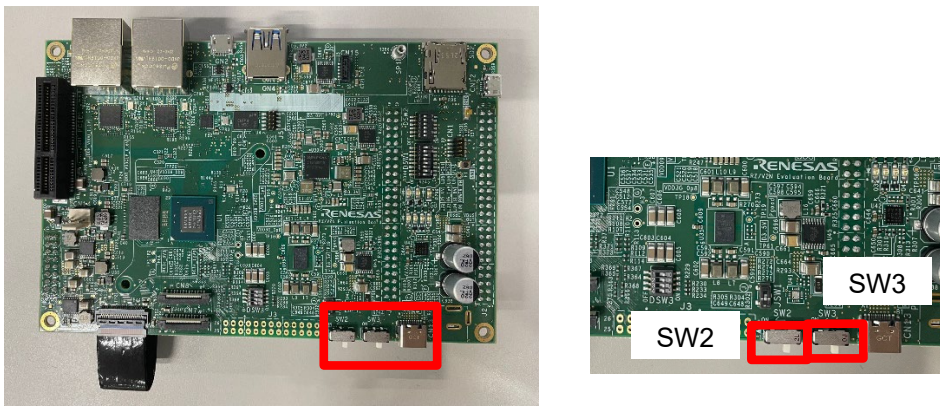


Figure 41: Power Supply

Connect USB-PD Power Charger to CN13.

Turn the SW3 ON, then LD2 and LD7 light up.

Turn the SW2 ON, then LD1, LD3 and LD4 light up.

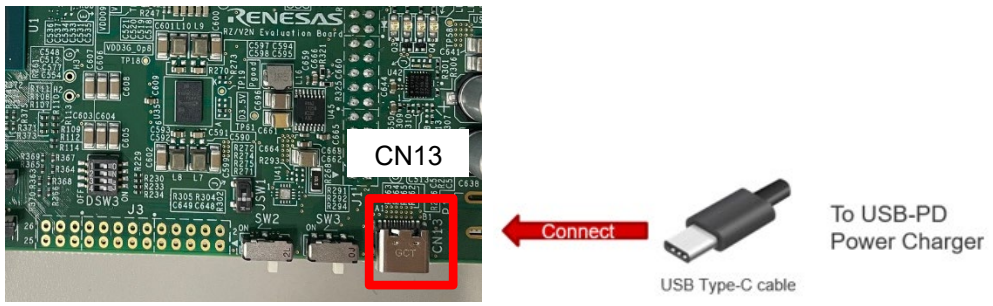


Figure 42: Power Supply

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 e² 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 setup correctly and builds a working executable image for your chip.
- The debugger has installed with working drivers and is properly connected to the board.
- The board is powered up and its jumper and switch settings are probably correct.
- The microprocessor is alive, the clocks are running, and the memory is initialized.
- Timer (GTM) interrupt is intentionally fired and GPIO is properly controlled.

Note: RZ/V2H SMARC EVK has on-board LED but RZ/V2L SRMAC EVK board does not have any LED. Thus, Blinky sample application for RZ/V2L SRMAC EVK is designed to use the Pmod module described below alternatively:

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

This module is not included on the RZ/V2L SRMAC EVK board and so, please prepare it beforehand.

Figure 43: Connection Pmod LED module (410-076)

In the case of RZ/V2H EVK, the on-board LED is placed as below.

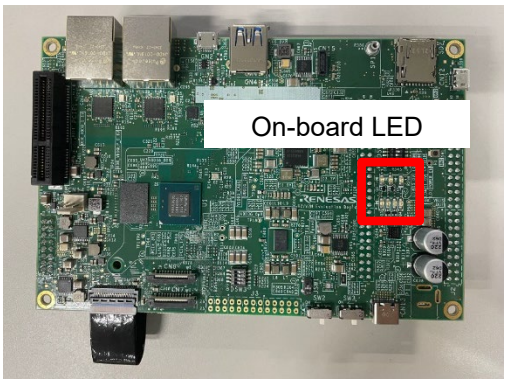


Figure 44: On-board LED of RZ/V2H EVK

4.3 Create a New Project for Blinky

The creation and configuration of an RZ/V C/C++ FSP Project is the first step in the creation of an application.

The base RZ/V pack includes a pre-written Blinky example application.

Follow these steps to create an RZ MPU project:

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

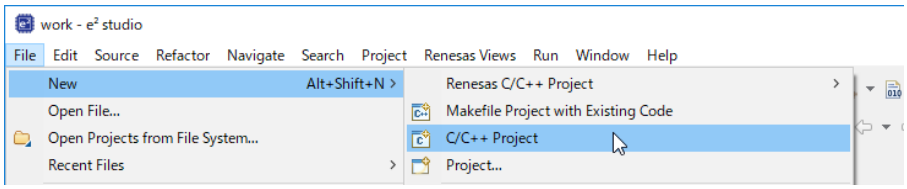


Figure 45: New C/C++ Project

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

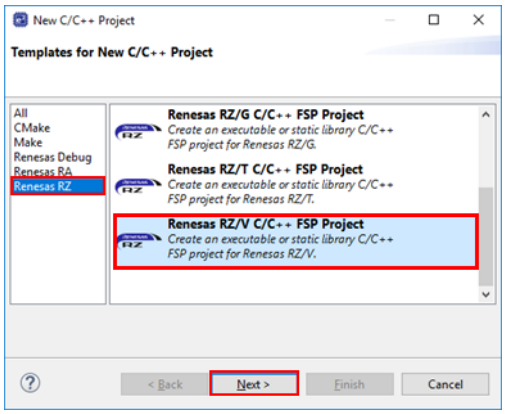


Figure 46: Renesas RZ/V C/C++ FSP Project

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

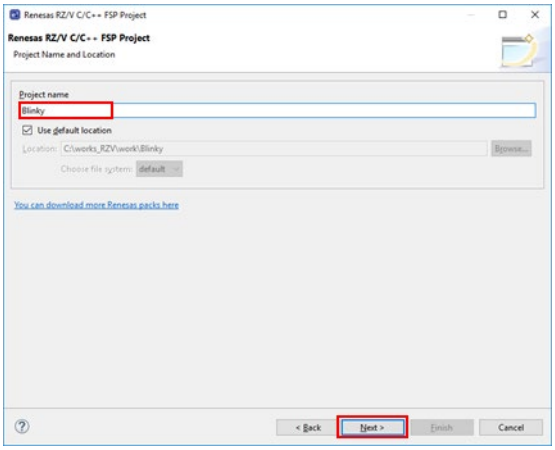


Figure 47 : e² studio Project Configuration window (part 1)

5. Select the board support package by selecting the name of your board from the Device Selection drop-down list. Select **GNU ARM Embedded** in Toolchains and version is **12.2.1.arm-12-24** and Click **Next**.

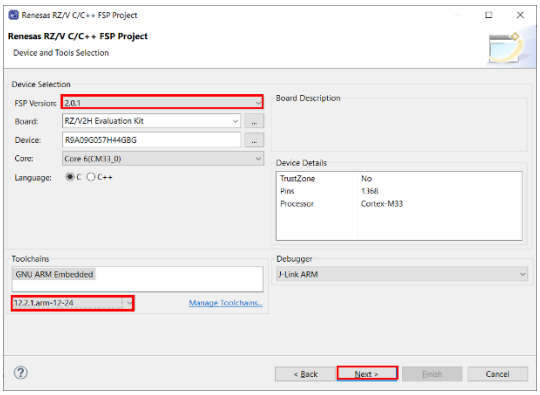


Figure 48 : e² studio Project Configuration window (part 2)

6. Select the **build artifact** and **RTOS**. Be sure that on the current version, **Secure** should always be chosen at the **Sub-core start state**. Otherwise, the created project can't be built successfully.

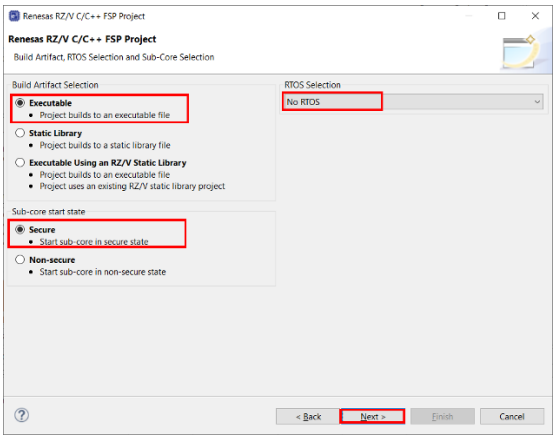
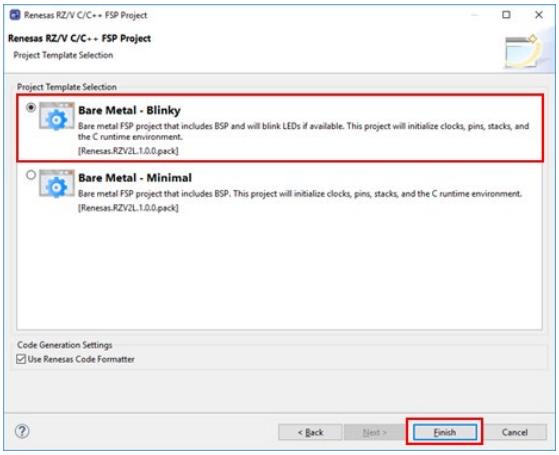
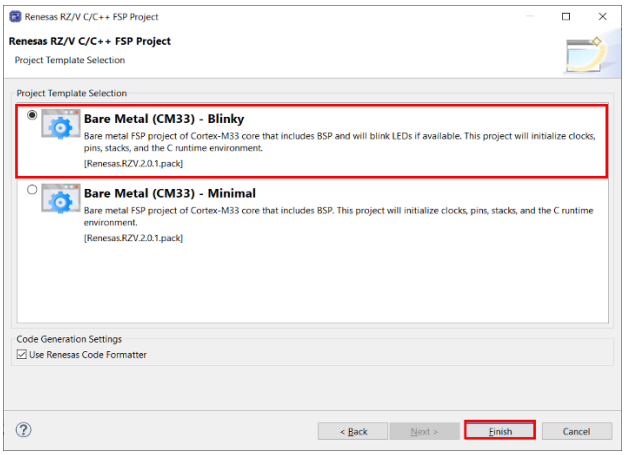


Figure 49 : e² studio Project Configuration window (part 3)

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



Templates for RZ/V2L



Templates for RZ/V2H

Figure 50 : e² studio Project Configuration window (part 4)

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

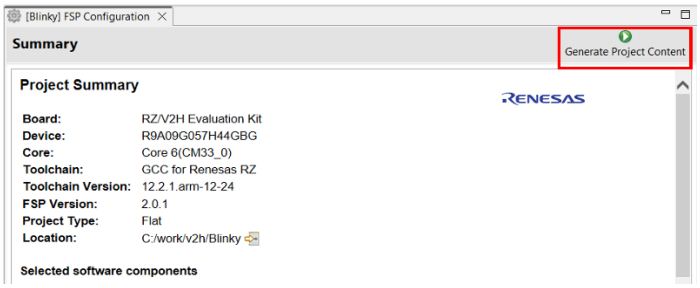


Figure 51 : e² studio Project Configuration tab

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

4.3.1 Details about the Blinky Configuration

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

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

4.3.2 Configuring the Blinky Clocks

By selecting the Blinky template, the clocks are configured by e² 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 e² 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

To see the configuration parameters for any of the components, check the Properties tab in the HAL window for the respective driver (see 5.2.8.Adding and Configuring HAL Drivers).

4.3.5 Where is main()?

The main function is located in < project >/rzv_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 e² 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 by 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 writing to the GPIO pin with “R_BSP_PinWrite((bsp_io_port_pin_t) pin, pin_level)” in 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 build 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.

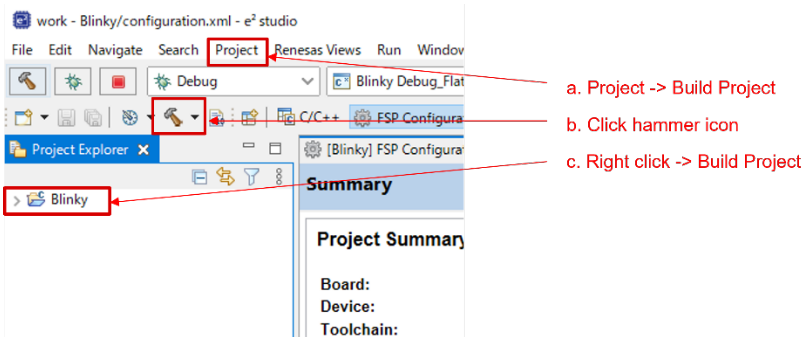


Figure 52 : e² studio Project Explorer window

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

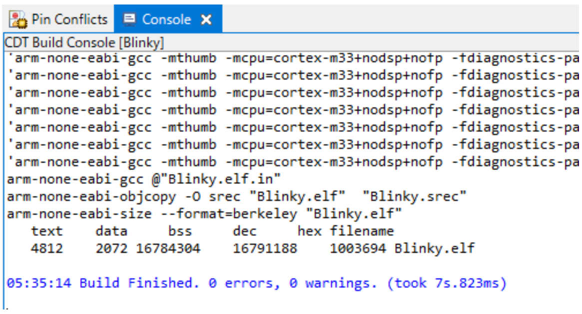


Figure 53 : e² studio Project Build console

4.5 Debug the Blinky Project

4.5.1 Debug prerequisites

To debug the project on a board, you need

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

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

4.5.2 Debug steps

To debug the Blinky application, follow these steps:

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

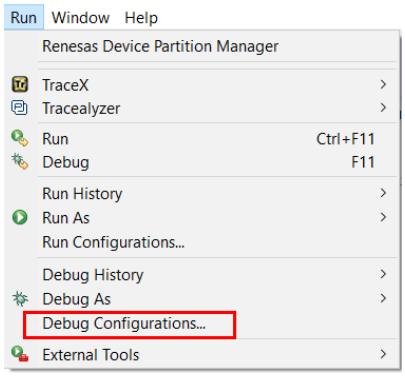


Figure 54 : e² studio Debug icon

or by selecting the drop-down menu next to the bug icon and selecting **Debugger Configuration**s ...

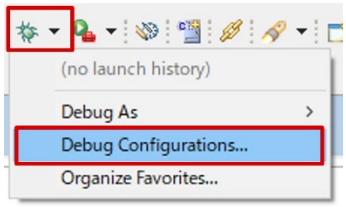


Figure 55 : e² studio Debugger Configurations selection option

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** window displays the **Debug configuration** for your **Blinky** project.

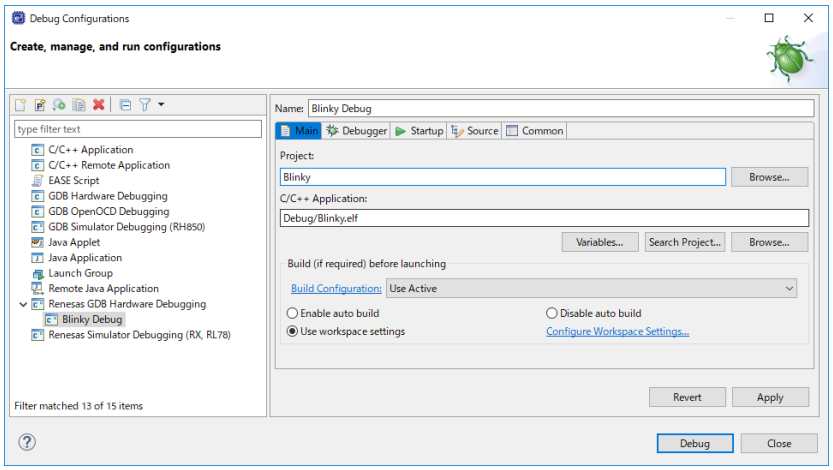
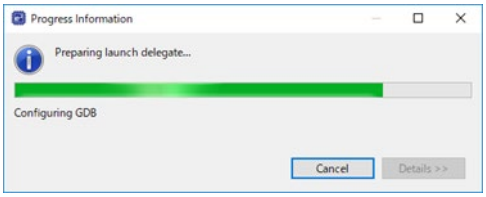


Figure 56 : e² studio Debugger Configurations window with Blinky project (1)

3. Select the debug configuration for the generated project and select the **Debugger** tab.
4. Click **Debug** to begin debugging the application.
5. Extracting **RZ Debug**.



4.5.3 Details about the Debug Process

In debug mode, e² studio executes the following tasks:

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

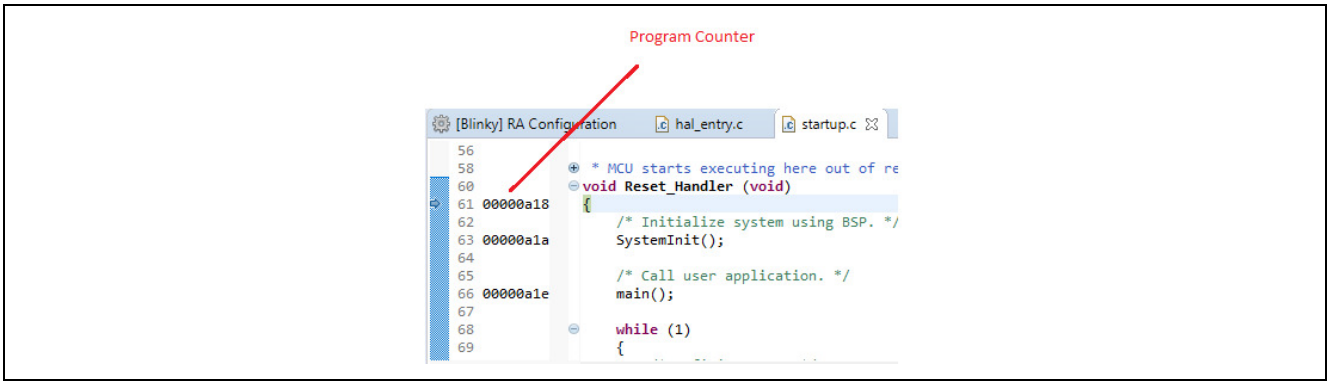


Figure 57 : e² studio Debugger memory window

4.5.4 Run the Blinky Project

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

Figure 58 : e² studio Debugger Play icon

The LED (RZ/V2L SRMAC EVK: Pmod LED, RZ/V2H SRMAC EVK: On bord LED) should now be blinking.

5. FSP application launch with e² studio

This section describes how to create project and debug in single core environment. Please refer to the section 6 for the multi-core project creation and debug in RZ/V2H.

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 located in the project's root directory.

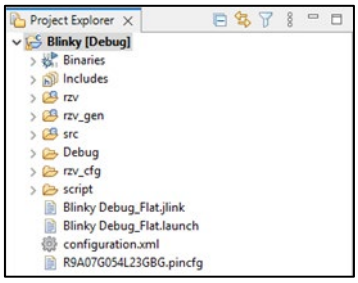


Figure 59 : e² studio Project Configuration file

Double-click on the configuration.xml file to open the RZ MPU Project Editor. To edit the project configuration, make sure that the **FSP Configuration** perspective is selected in the upper right-hand corner of the e² studio window. Once selected, you can use the editor to view or modify the configuration settings associated with this project.

Figure 60 : 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 (rzv_cfg.txt) with all the project settings is generated. The format allows differences to be easily viewed using a text comparison tool. The generated file is located in the project root directory.

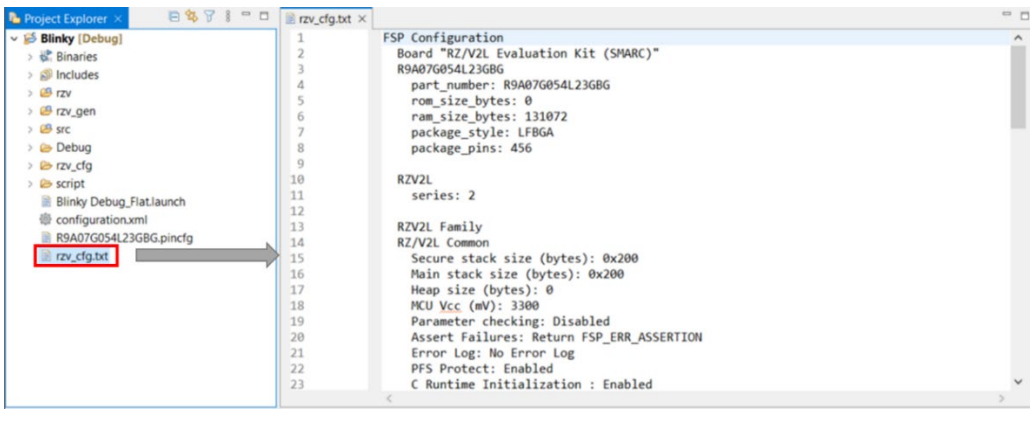


Figure 61 : RZ Project Report

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, however the functionality should be easy to follow.

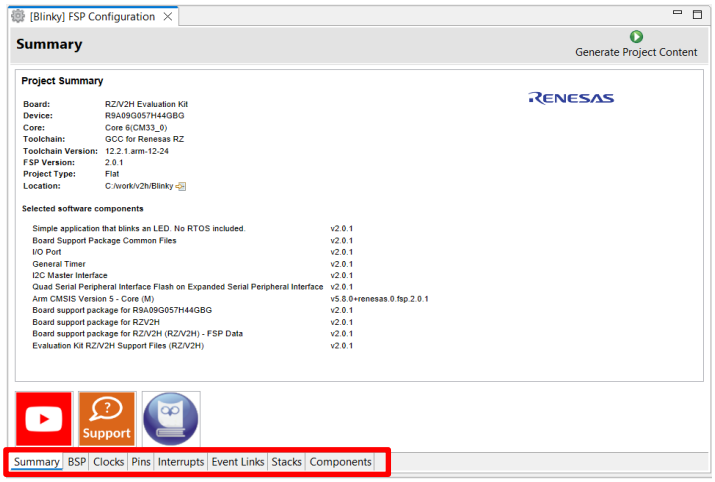


Figure 62 : RZ Project Editor tabs

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**.

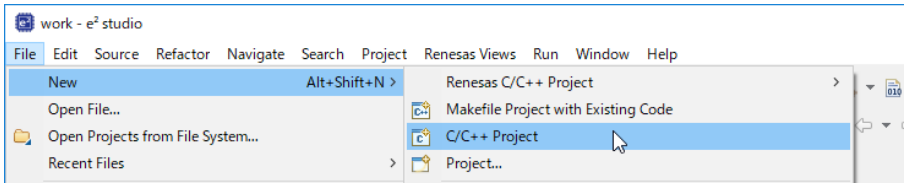


Figure 63 : New RZ MPU Project

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

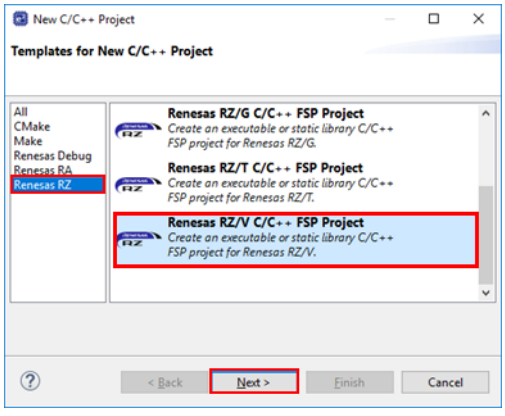


Figure 64 : New Project Templates

3. Select a project name and location.

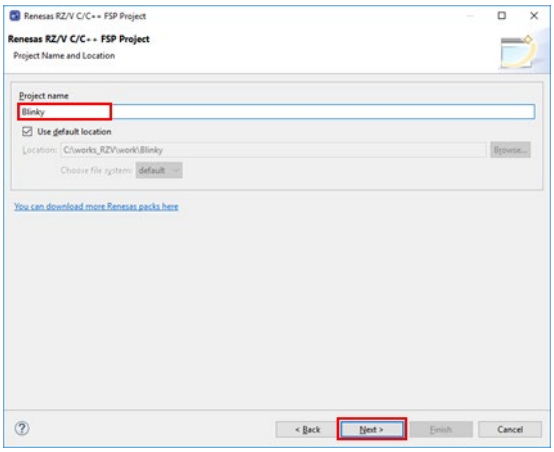


Figure 65 : RZ MPU Project Generator (Screen 1)

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 select **Custom User Board** for any of the RZ MPU devices with your own BSP definition.
3. Select the **Device**. The **Device** is automatically populated based on the **Board** selection. Only change the **Device** when using the **Custom User Board (Any Device)** board selection.
4. Select the **Core**. You can select Core 6(CM33_0), Core 4(CR8_0) or Core 5(CR8_1) if you selected RZ/V2H for the **Device**.
5. To add threads, select **RTOS**, or **No RTOS** if an RTOS is not being used.
6. The **Toolchain** selection defaults to **GNU Arm Embedded.**
7. Select the **Toolchain version**. This should default to the installed toolchain version.
8. Select the **Debugger**. The J-Link Arm Debugger is preselected.
9. Click **Next**.

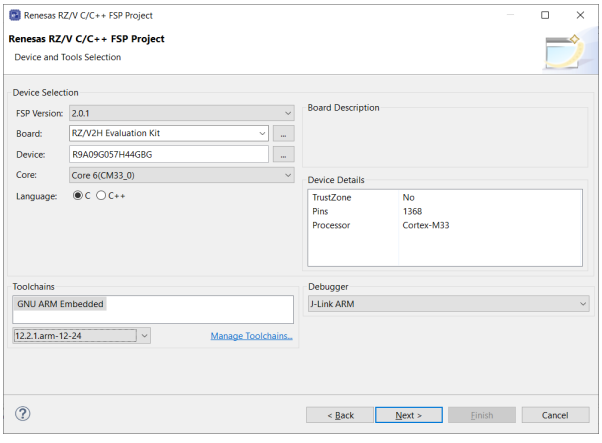


Figure 66 : RZ MPU Project Generator (Screen 2-1)

If Core 4(CR8_0) or Core 5(CR8_1) is selected in procedure 4, you need to select the preceding project. To select the preceding project when creating the Core 4(CR8_0) or Core 5(CR8_1) project, it is required to prepare Core 6(CM33_0) before Core 4(CR8_0) or Core 5(CR8_1) project creation.

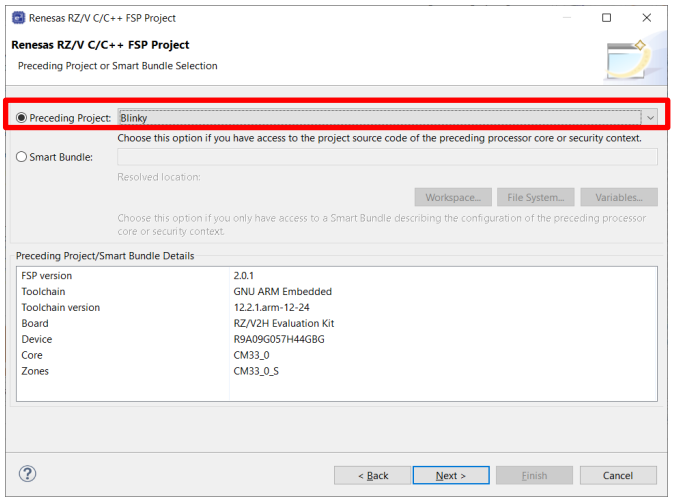


Figure 67 : RZ MPU Project Generator (Screen 2-2)

5.1.2.2 Selecting a Project Template

In the next window, select the build artifact, **Sub-core start state** and **RTOS**. Be sure that you select **Secure** as **Sub-core start state** in the current version.

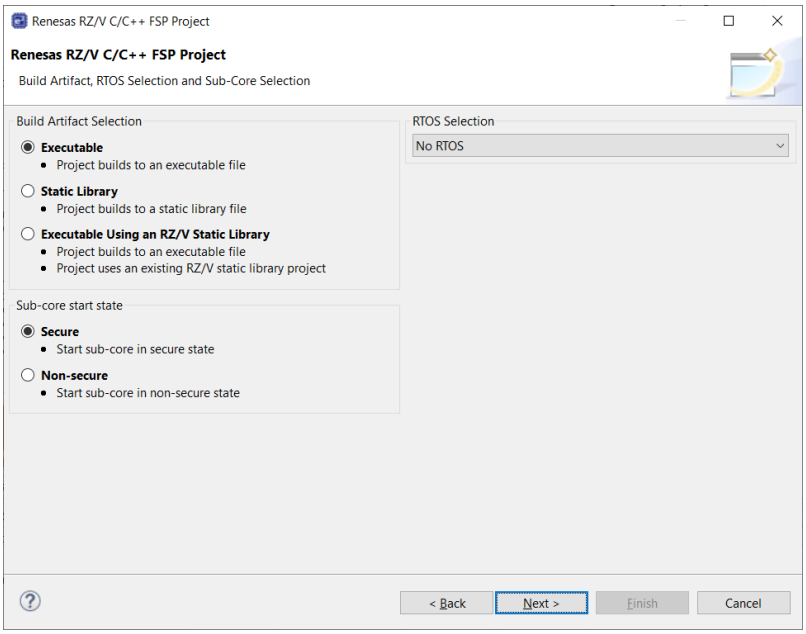


Figure 68 : RZ MPU Project Generator (Screen 3)

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/V MPU Pack. Once you have selected the appropriate template, click **Finish**.

Note: If you want to develop your own application, select the basic template for your board, **Bare Metal - Minimal** or **FreeRTOS - Blinky**.

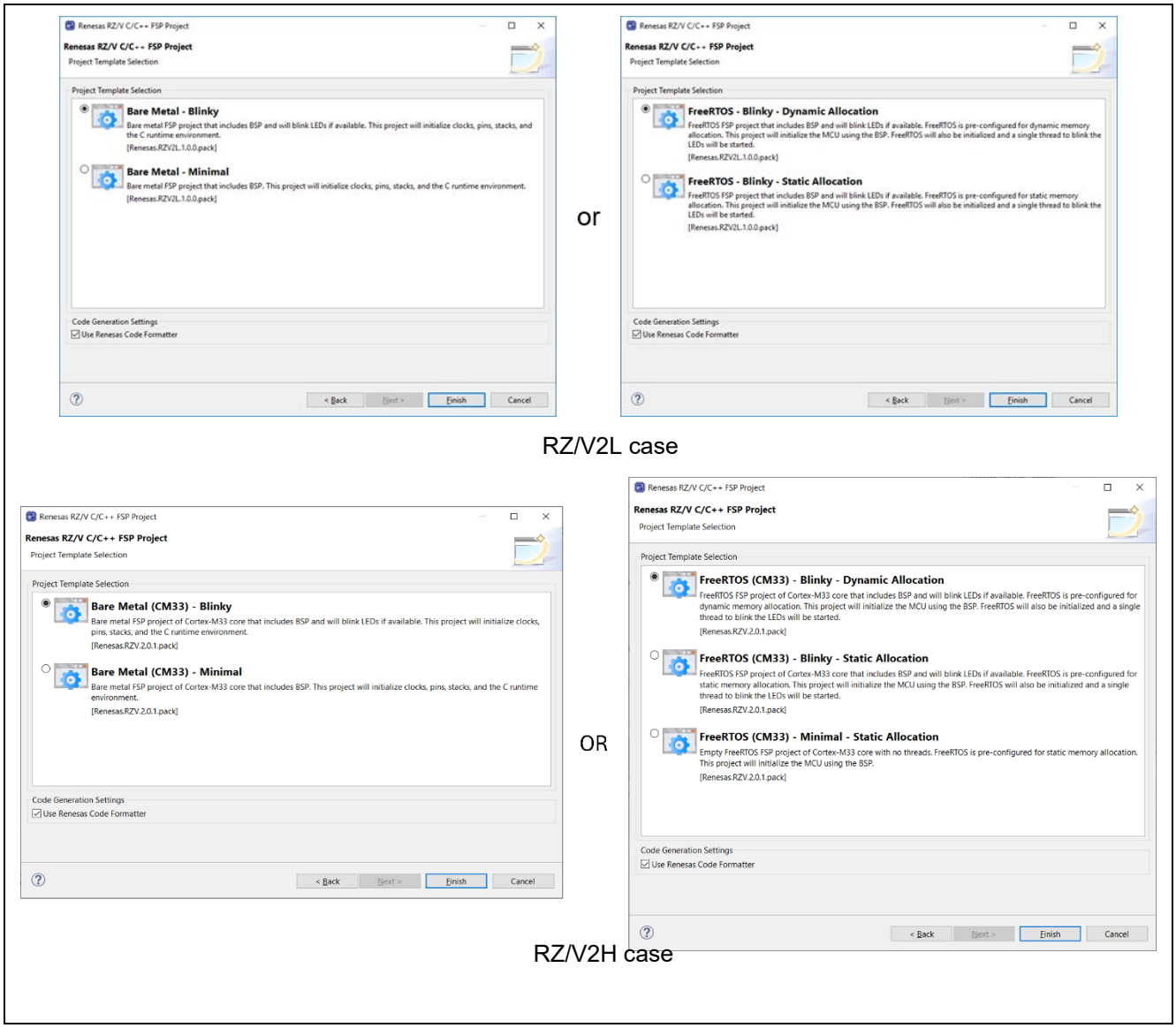


Figure 69 : RZ MPU Project Generator (Screen 4)

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

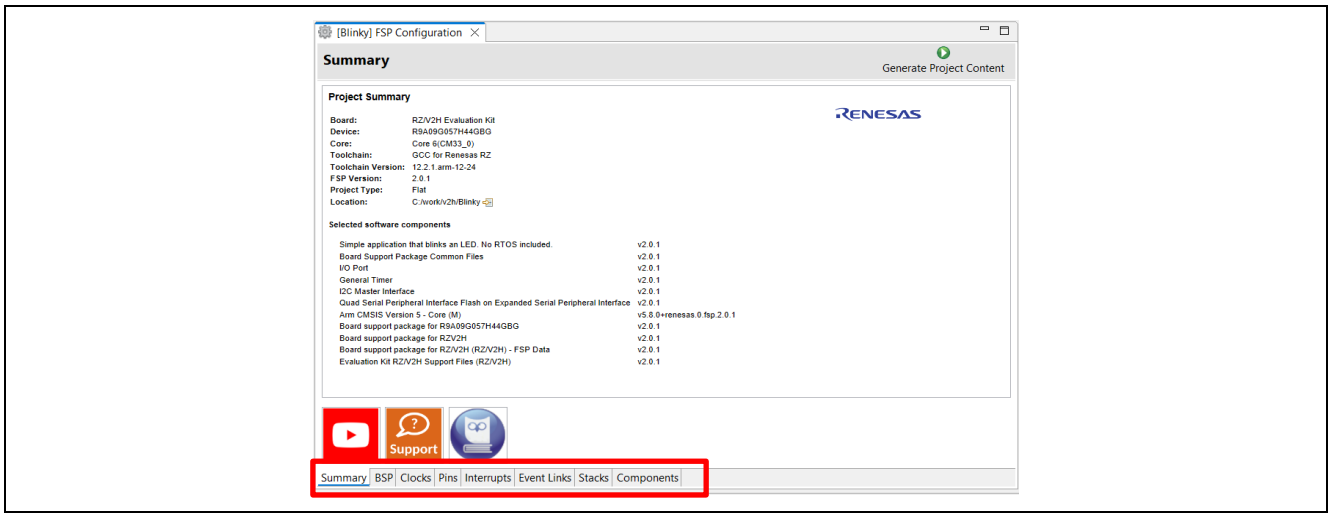


Figure 70 : RZ MPU Project Editor and available editor tabs

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

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

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

5.1.3 Duplication of Resources

In the case of RZ/V2H Core 4(CR8_0) or Core 5(CR8_1) project, duplicate resources are indicated as red character in **Stacks** tab when using resources that are used in the linked Core 6(CM33_0) project.

The following image is an example of both Core 6(CM33_0) and Core 4(CR8_0) projects using same resource. The duplication of r_gtm is indicated in **Stacks** tab. To avoid this duplication, please change the channel resource in **Properties** of r_gtm.

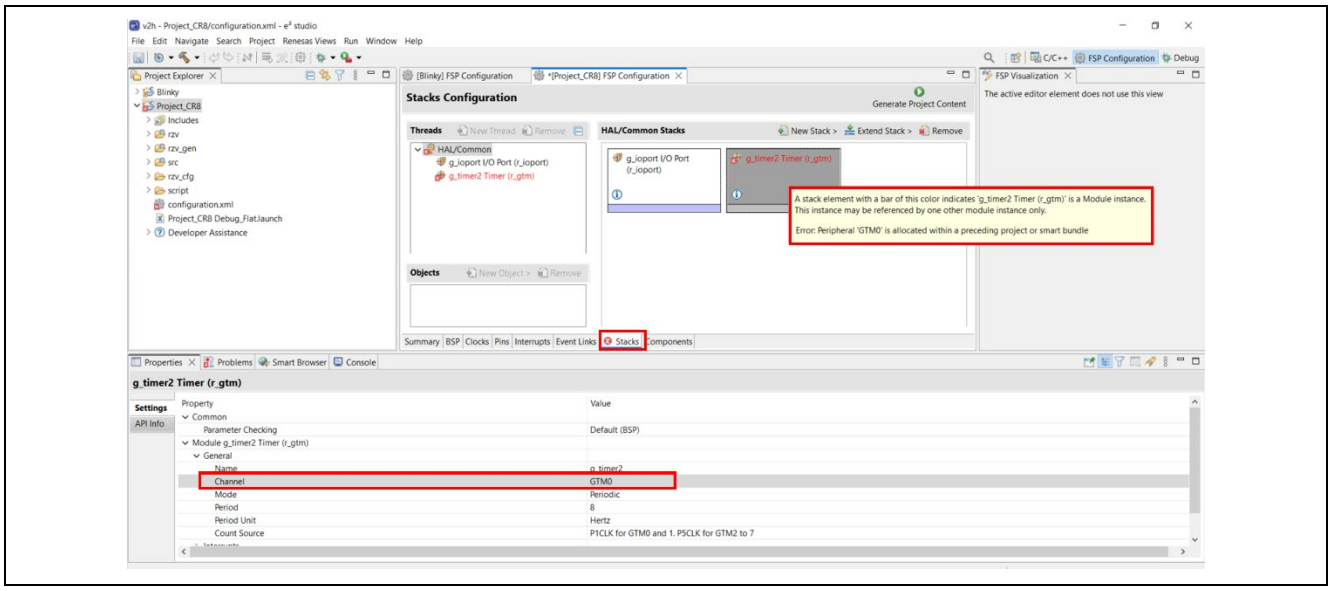


Figure 71 : Duplication of resource between Core 6(CM33_0) and Core 4(CR8_0) projects

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, e² 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

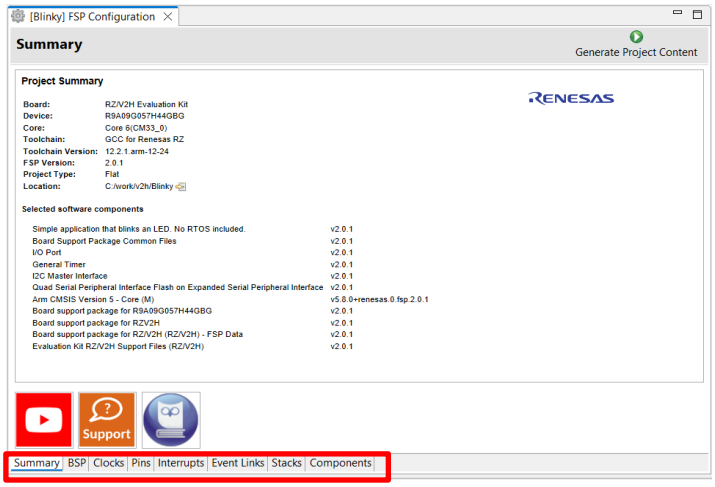


Figure 72 : Configuration Summary tab

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

5.2.2 Configuring the BSP

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

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

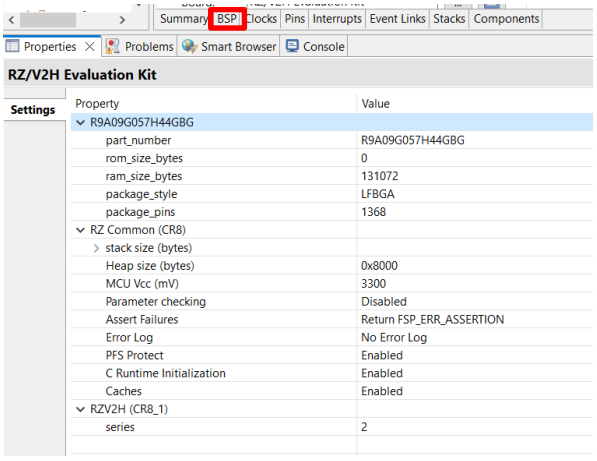


Figure 73 : Configuration BSP tab

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. e² studio checks the entry fields to flag invalid entries. For example, only valid numeric values can be entered for the stack size.

When you click the **Generate Project Content** button, the BSP configuration contents are written to rzv_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, scif_uart driver calculates the communication rate from the settings in Clocks tab. Please note that the clock configuration is carried out on the main core (CA55) in advance when CM33 work as sub core. Thus, clocks configuration here must align with the settings on CA55.

In the case of CM33 cold boot, BSP will configure each clock setting in start-up process according to content of **Clocks** tab.

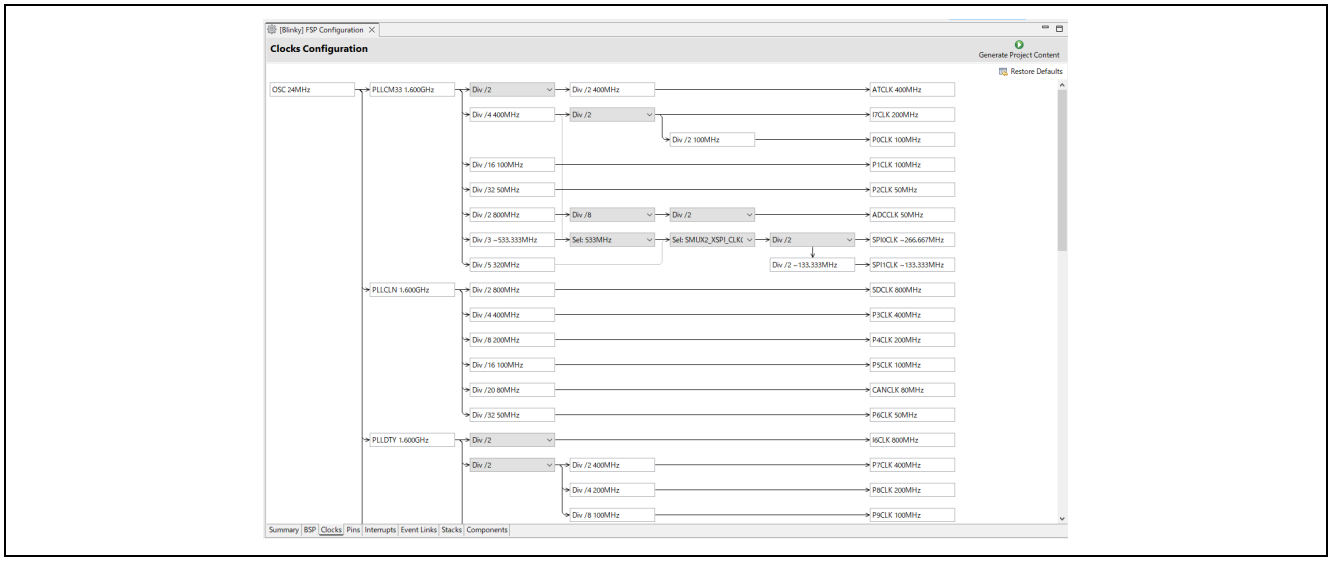


Figure 74 : Configuration Clocks tab

When you click the **Generate Project Content** button, the clock configuration contents are written to: rzv_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 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. The location and function of the pins are shown in the **FSP Visualization** view. For more information on the function and color coding of the pins, please check the Legend in the **FSP Visualization** view.

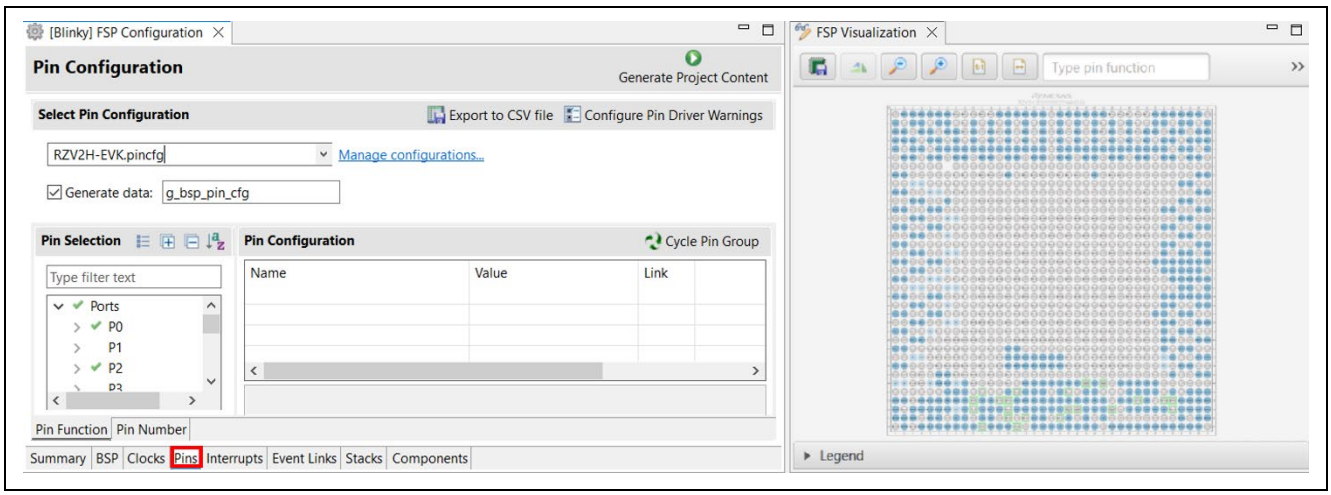


Figure 75 : Pin Configuration

The pin configurator includes 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 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 P70 is already used by the IRQ_IRQ0, and the attempt to connect to this pin to the GPT results in dangling connection error. To fix this error, select another port from the pin drop-down list or disable the IRQ_IRQ0.

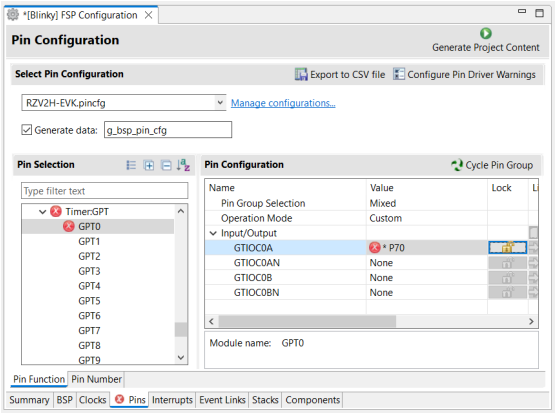


Figure 76 : e2 studio Pin Configurator

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

In the case of versions earlier than RZ/V FSP v2.0.0, It does not support **Pins** tab and If user would like to use I/O port, I/O Port setting should be applied to “src/pin_data.c” manually. For details on I/O Port setting and, please refer to Setting GPIO with Flexible Software Package.

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.

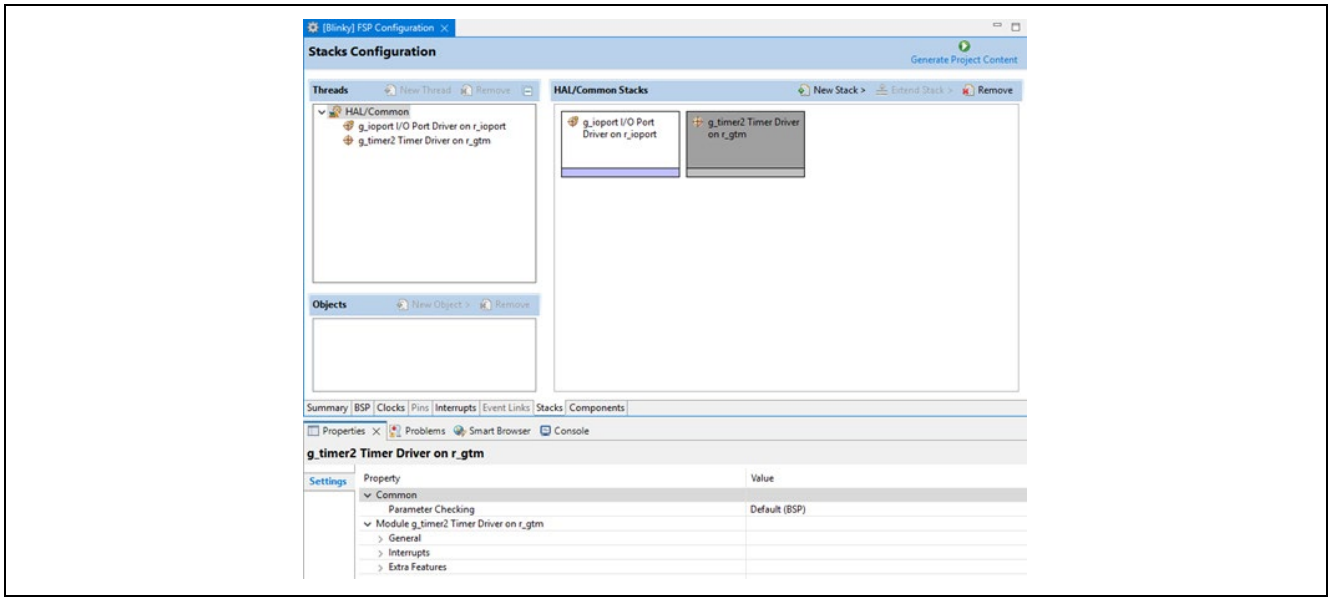


Figure 77 : Configuring Interrupts in the Stacks tab

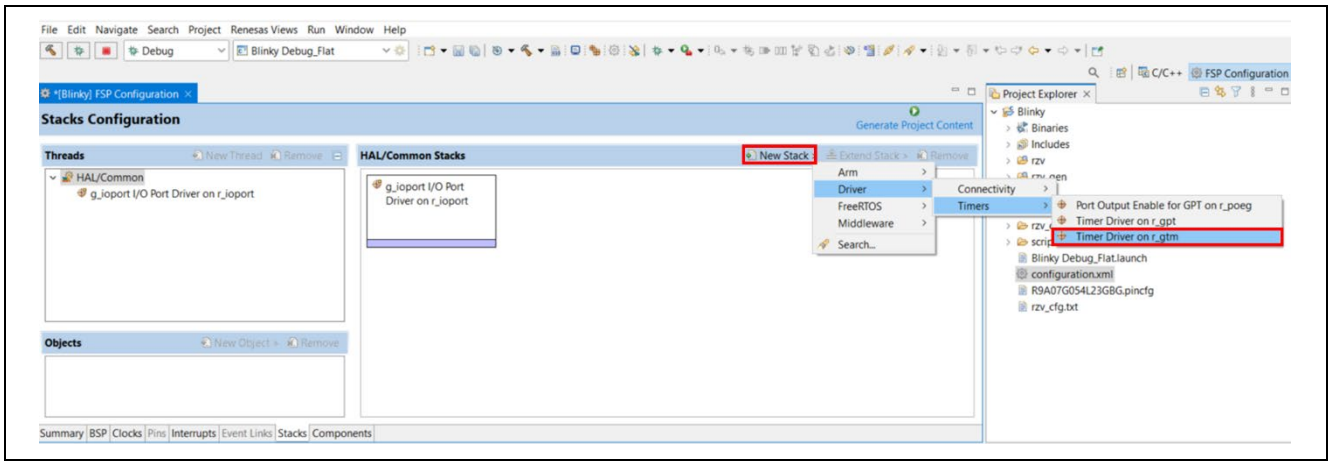


Figure 78: Add new stack Timer (GTM)

5.2.6 Creating Interrupts from the Interrupts Tab

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

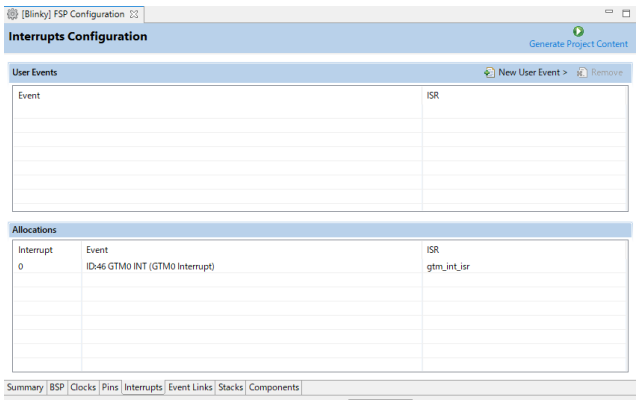


Figure 79 : Configuring interrupt in Interrupt Tab

And on the Interrupts tab, the user can add a peripheral interrupt create by user.
This can be done by adding a new event via the **New User Event** button.

5.2.7 Viewing Event Links

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

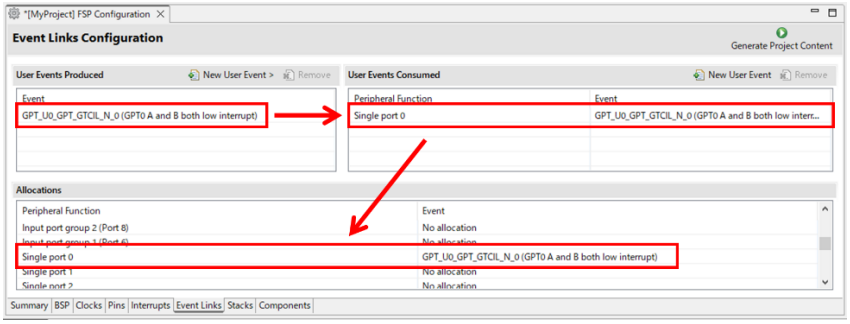


Figure 80 : e2 studio Project configurator – Viewing Event Links

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

Note1: When selecting an ELC event to receive for a module (or when manually defining an event link), only the events that are made available by the modules configured in the project will be shown.

Note2: On devices that do not have ELC, this tab is not available and it is grayed out.

5.2.8 Adding and Configuring HAL Drivers

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

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

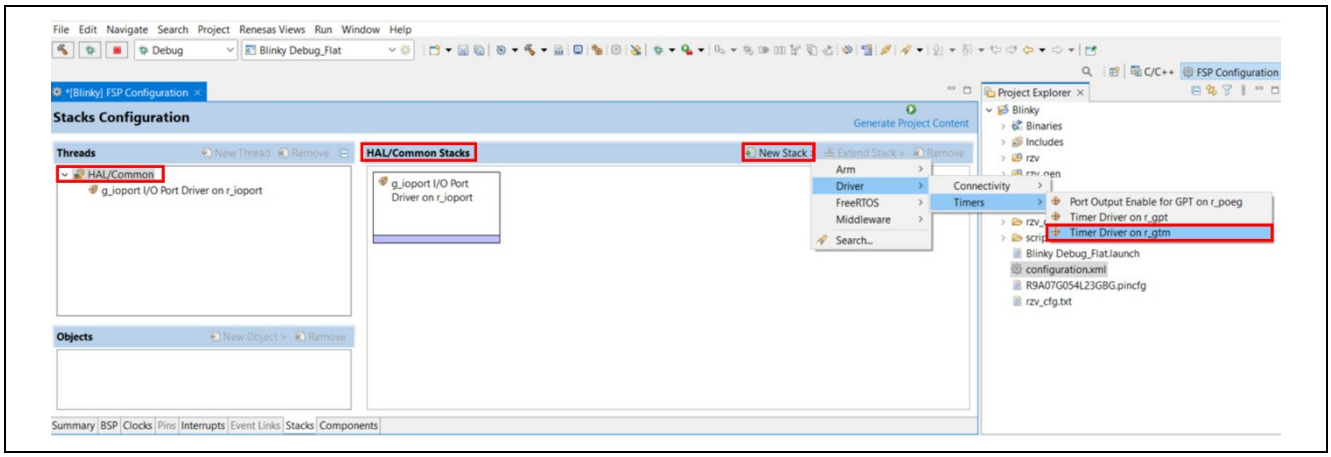


Figure 81 : e² studio Project configurator - Adding drivers

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

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

- The selected driver module and its files to the rzv/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?
rzv_gen/main.c	Contains main() calling generated and user code. When called, the BSP already has Initialized the MPU.	Yes
rzv_gen/hal_data.c	Configuration structures for HAL Driver only modules.	Yes
rzv_gen/hal_data.h	Header file for HAL driver only modules.	Yes
src/hal_entry.c	User entry point for HAL Driver only code. Add your code here.	No

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

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/V MPU projects are preselected. All modules that are necessary for the modules selected in the **Stacks** tab are included automatically. You can include or exclude additional modules by ticking the box next to the required component.

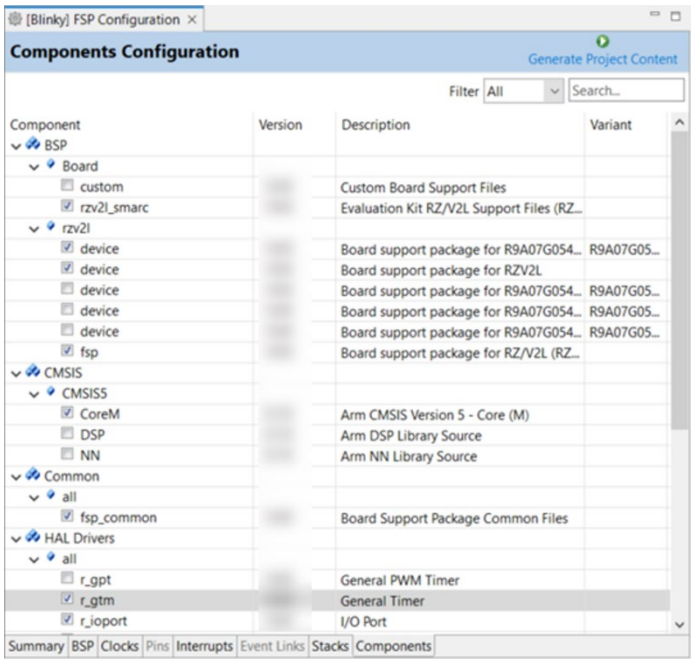


Figure 82 : Components Tab

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

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

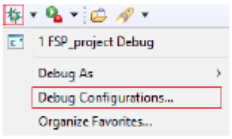
e² studio also creates configuration files in the rzv_cfg/fsp_cfg folder with configuration options set in the **Stacks** tab.

5.4 Debugging the Project

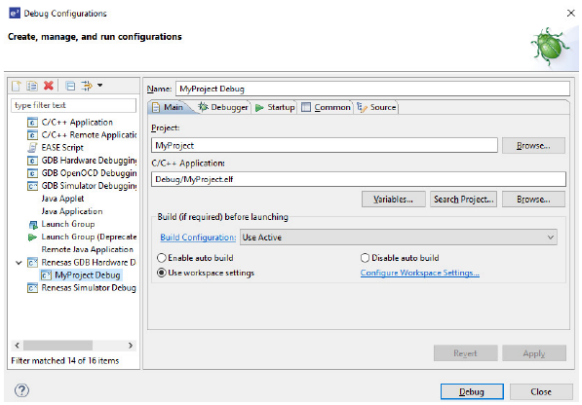
Once your project builds without errors, you can use the Debugger to download your application to the board and execute it.

To debug an application, follow these steps:

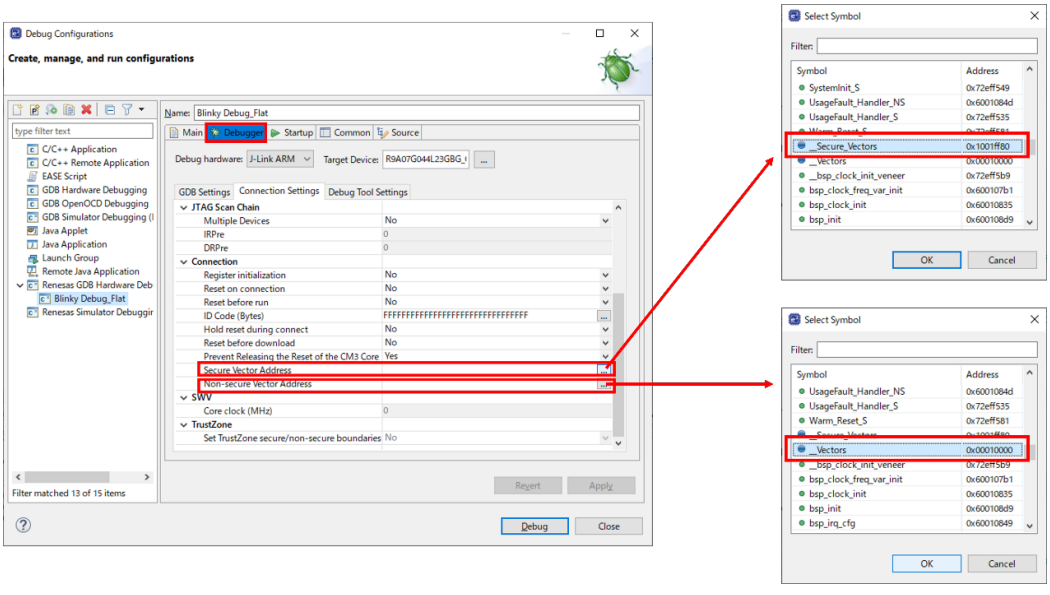
1. On the drop-down list next to the debug icon, select **Debug Configurations**.



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



3. Secure and Non-secure Vector Address are configured in the **Connection Settings** tab of the **Debugger** tab. The settings in below image are for setting the address of Secure and Non-secure Vector Offset (*) mapped in Blinky project. Please note that these addresses vary in accordance with linker settings.
(*: In the case of RZ/V2H project, Non-secure Vector is empty since it is not used.)



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 3.1.2.2.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 optimization level of the compiler or add a library to the linker). Such modifications can be made from within e² 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 upon the toolchain being used.

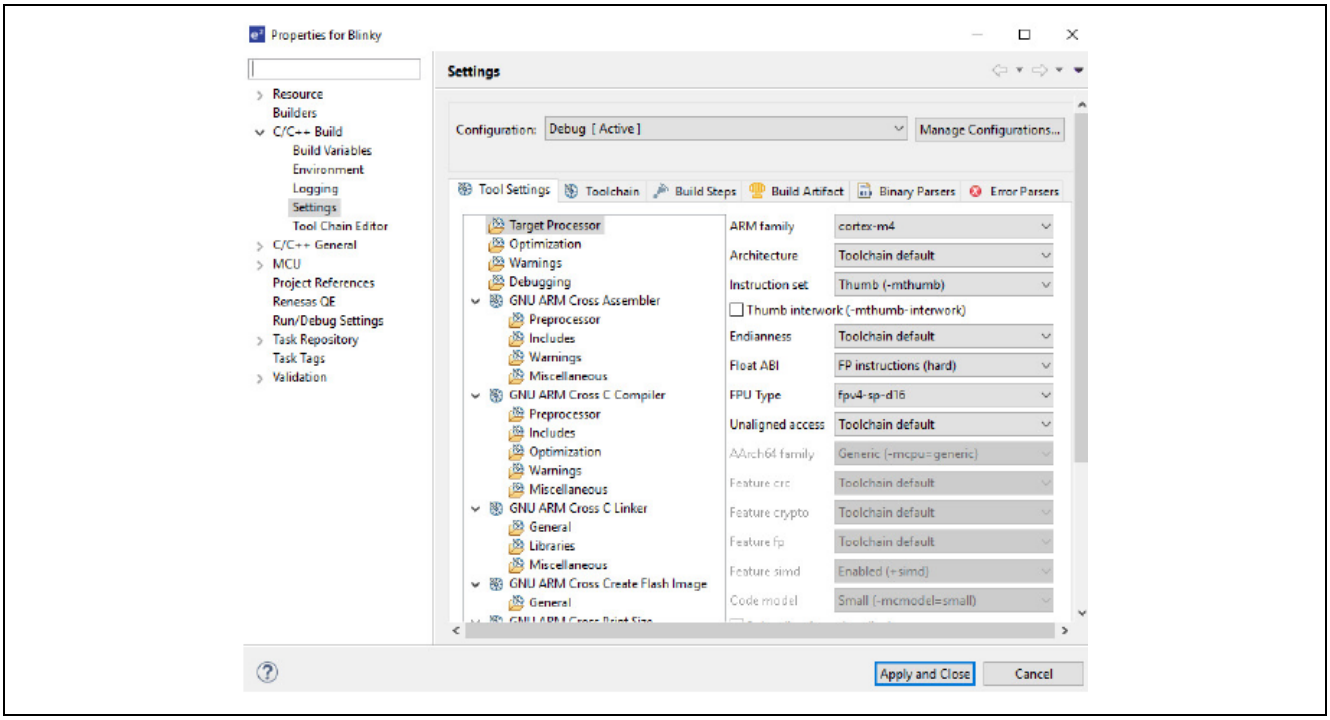


Figure 83 : e² studio Project toolchain settings

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 control the location of the various memory sections are contained in a script file specific for the device being used. This script file is included in the project when it is created and is found in the script folder (for example, /script/fsp.ld).

5.6 Importing an Existing Project into e² studio

1. Start by opening e² 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 e² studio startup, you will see the Workspace Launcher Dialog box as shown in the following figure.

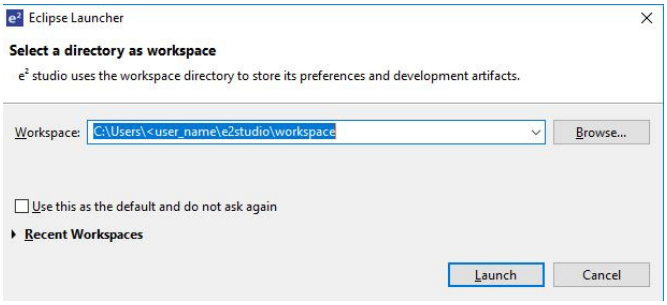


Figure 84 : Workspace Launcher dialog

   b. Enter a new workspace name in the Workspace Launcher Dialog as shown in the following figure. e² studio creates a new workspace with this name.

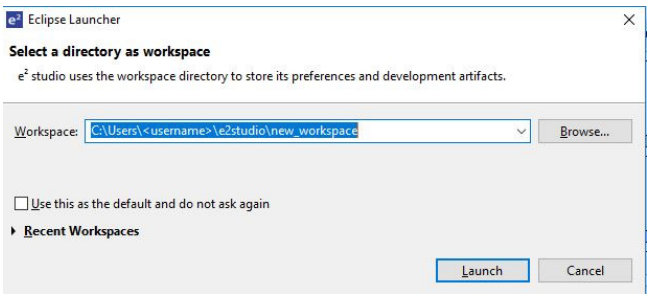


Figure 85 : Workspace Launcher dialog - Select Workspace

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

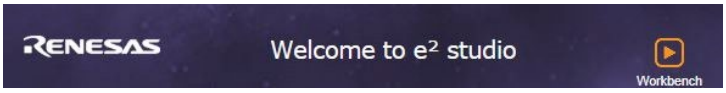


Figure 86 : Workbench arrow button

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.

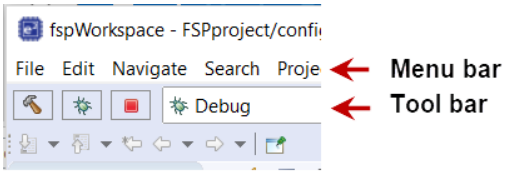


Figure 87 : Menu and tool bar

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

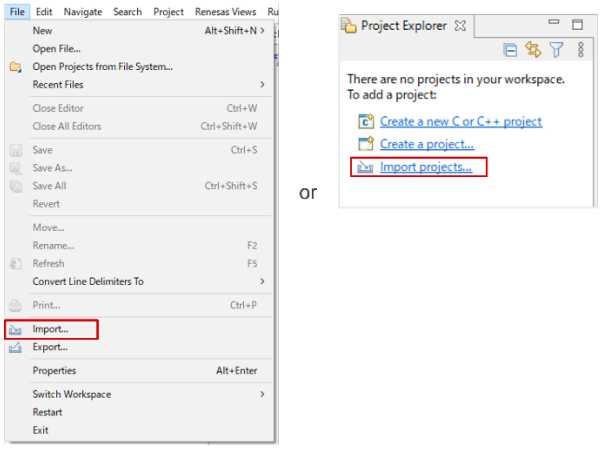


Figure 88 : File drop-down menu

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.

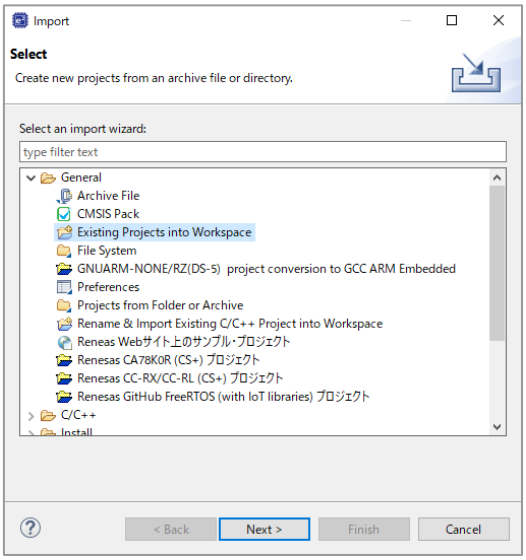


Figure 89 : Project Import dialog with "Existing Projects into Workspace" option selected

6. Click **Next**.
7. To import the project, use either **Select archive file** or **Select root directory**.

a. Click **Select root directory** file as shown in the following figure.

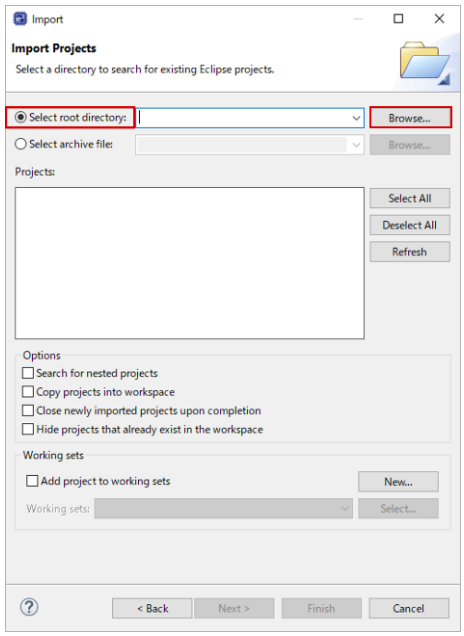


Figure 90 : Import Existing Project dialog 1 - Select root directory

8. Click **Browse**.
9. For **Select root directory**, browse to the project folder that you want to import.
10. Select the file for import.
11. Click **Open**.
12. Select the project to import from the list of **Projects**, as shown in the following figure.

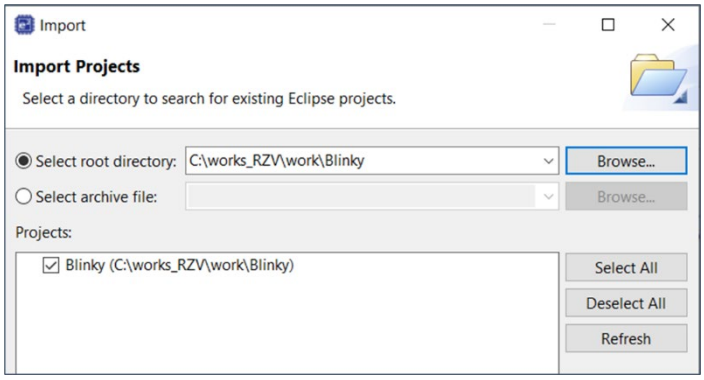


Figure 91 : Import Existing Project dialog 2

13. Click **Finish** to import the project.

6. Multi-Core Debug

In the case of RZ/V2H, FSP supports multi-core (CM33 core, CR8 core0 and CR8 core1) operation.

This section describes how to debug multi-core environment by running Blinky project for each core.

6.1 Project Creation for each Core

1. Create workspace for each core.

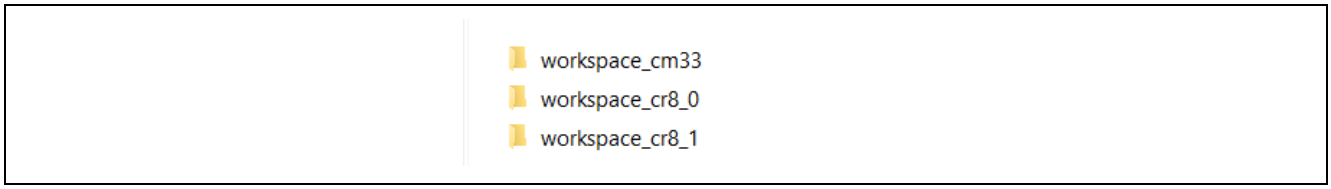


Figure 92 : Workspace creation

2. Launch e2studio and specify workspace for CM33 core.

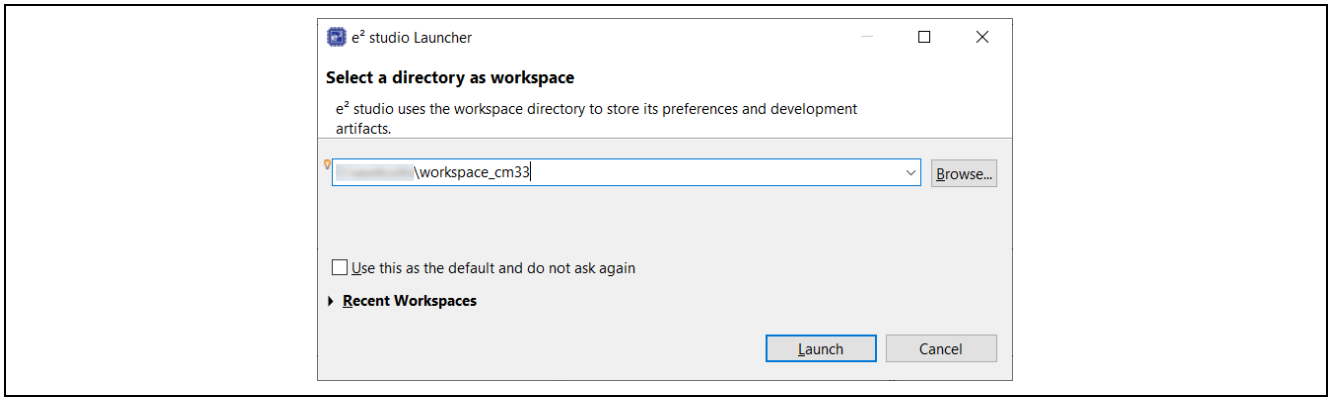


Figure 93 : Launching e2studio for CM33 Core Workspace

3. Generate new project for CM33 core in accordance with section 4.3.
   In the case of CM33 core project, ensure to select **Core** and **Project Template Selection** as below**.**

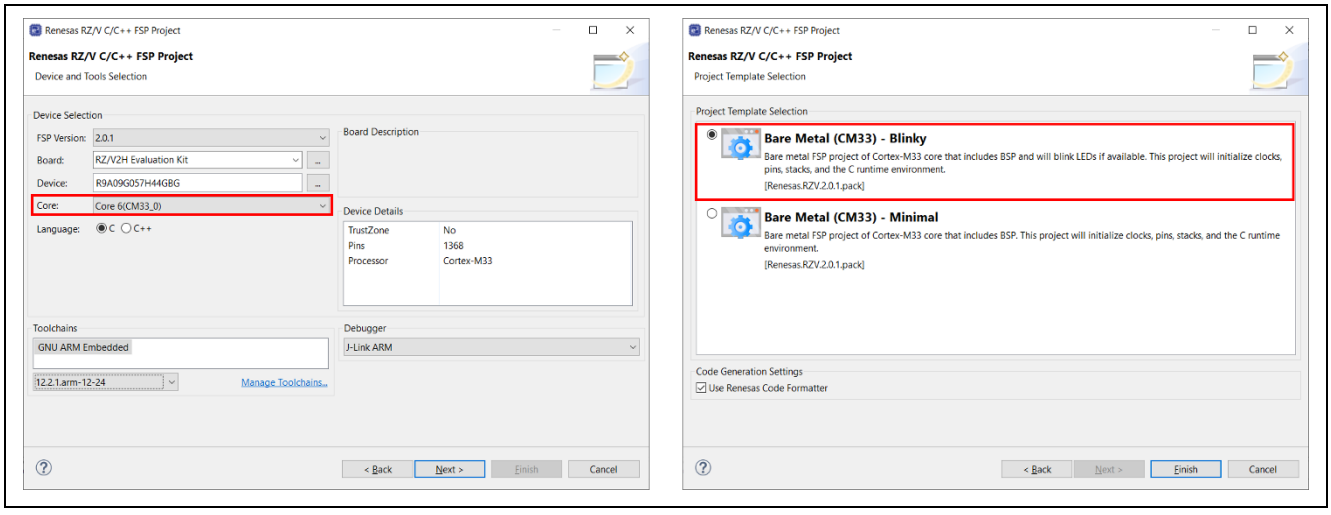


Figure 94 : Setting of CM33 core project

4. Build the Blinky project for CM33 core.

5. Generate new project for CR8 core0 in accordance with section 4.3.
   In the case of CR8 core0 project, ensure to select **Core, Preceding Project,** and **Project Template Selection** as below**.**

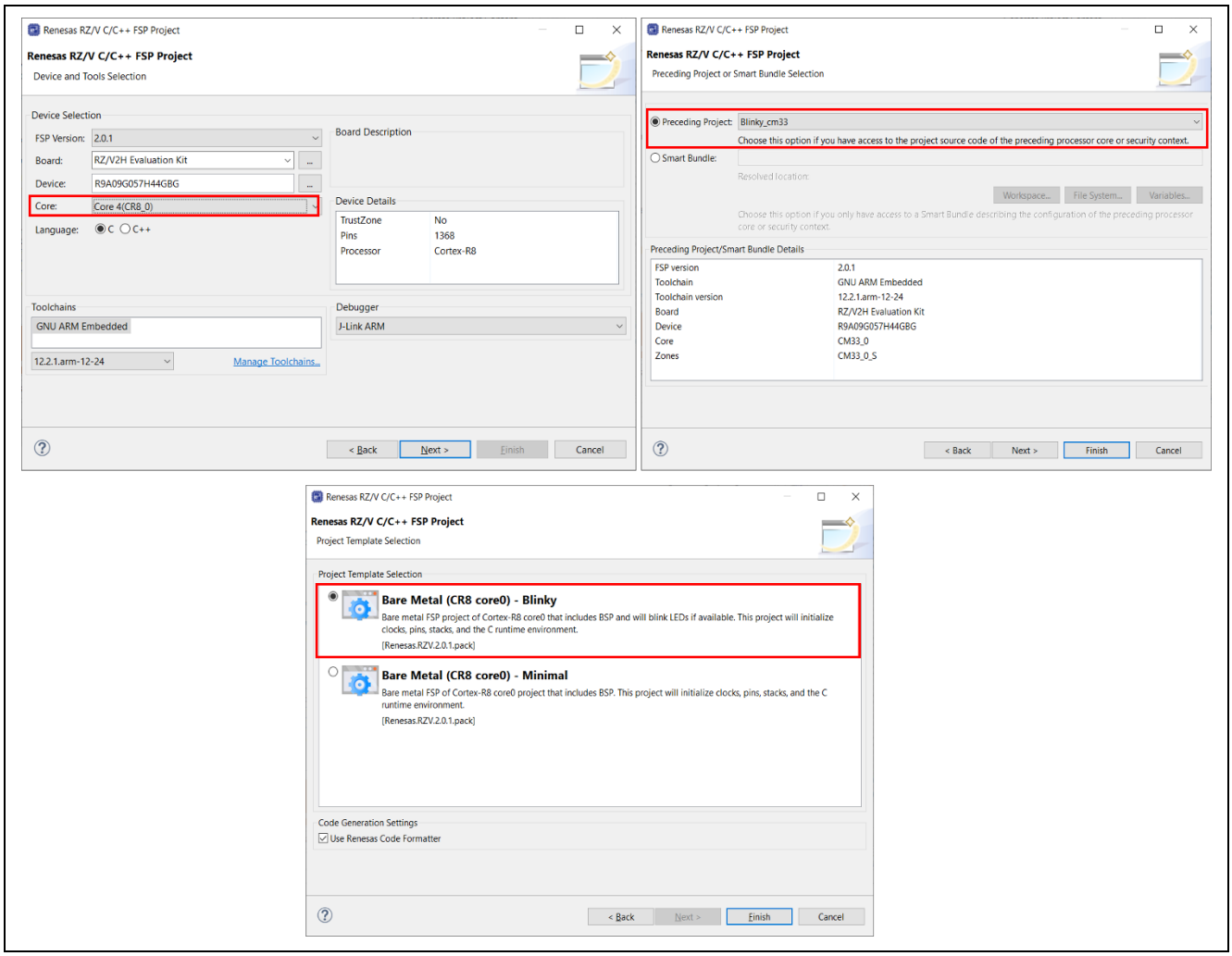


Figure 95 : Setting of CR8 core0 project

6. Build the Blinky project for CR8 core0.

7. Generate new project for CR8 core1 in accordance with section 4.3.
   In the case of CR8 core1 project, ensure to select **Core, Preceding Project,** and **Project Template Selection** as below**.**

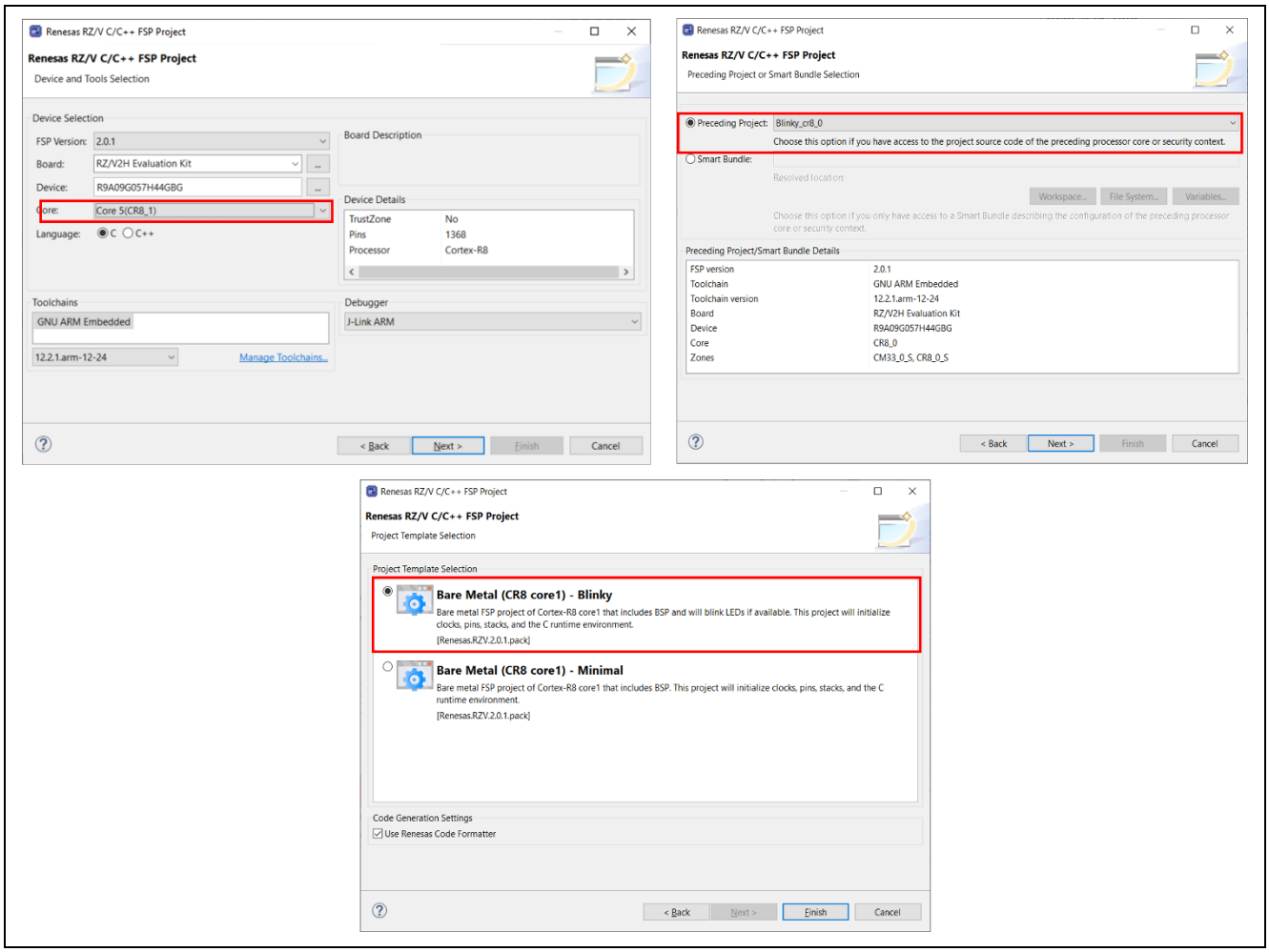


Figure 96 : Setting of CR8 core1 project

6. Build the Blinky project for CR8 core1.
7. 3 projects are created in CM33 core workspace.
   Copy all projects to CR8 core0 workspace and CR8 core1 workspace.

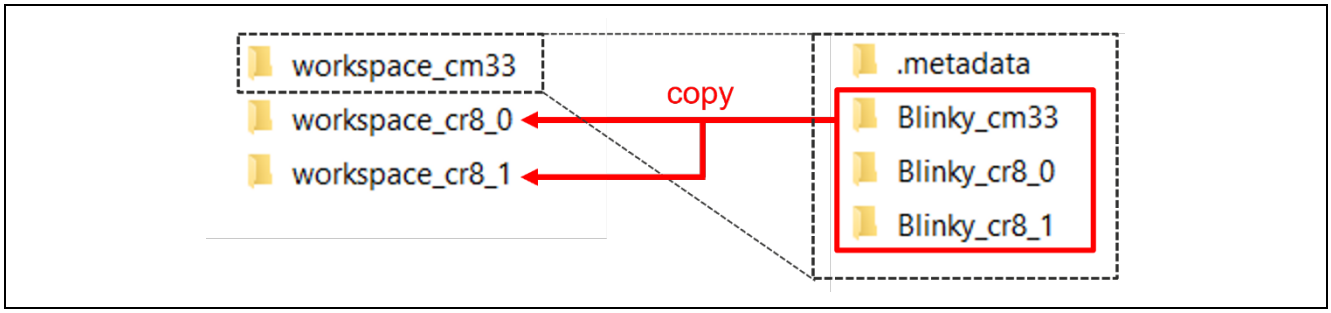


Figure 97 : Copy the project

6.2 Debugging the Project for each Core

1. Select the **Debug Configuration** on e2studio of CM33 core workspace and click **Debug**.

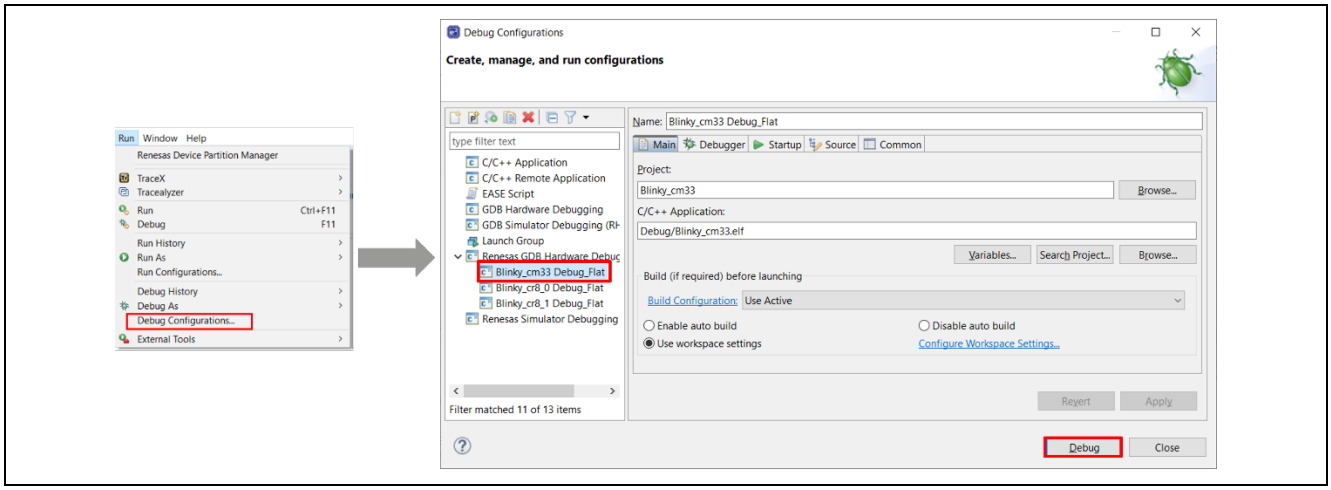


Figure 98 : CM33 Core Debug Configuration

2. Launch new e2stuio for CR8 core0 workspace.

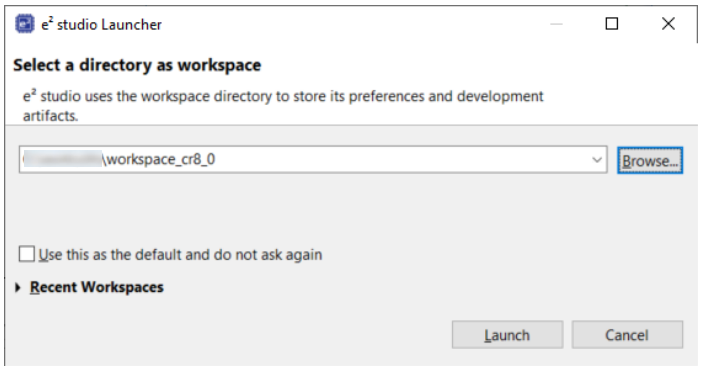


Figure 99 : Launching e2studio for CR8 Core0 Workspace

3. Select the **Debug Configuration** on e2studio of CR8 core0 workspace and click **Debug**.

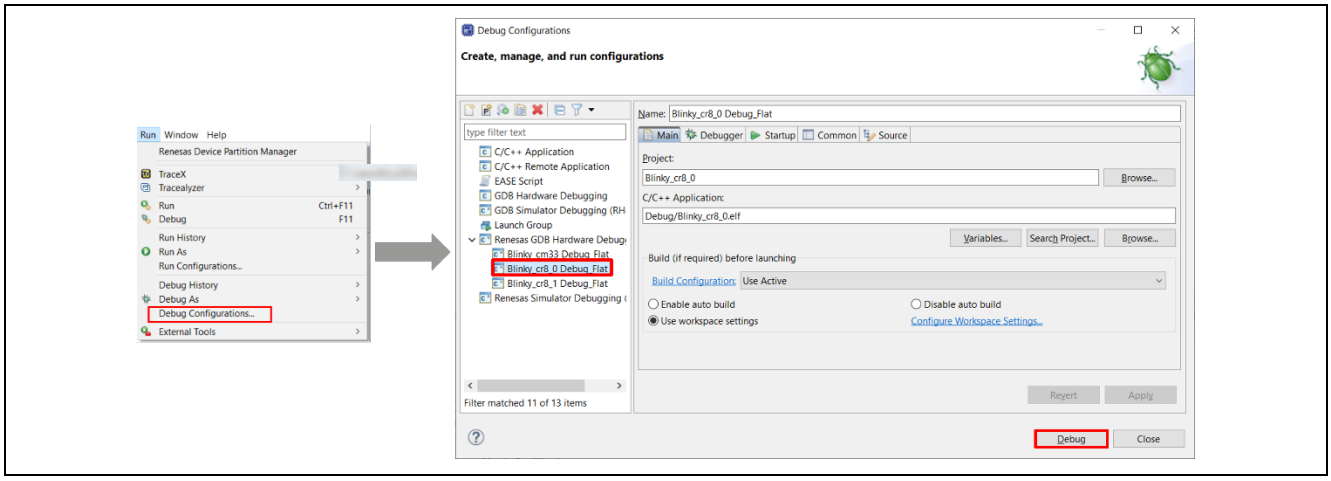


Figure 100 : CR8 Core0 Debug Configuration

4. Launch new e2stuio for CR8 core1 workspace.

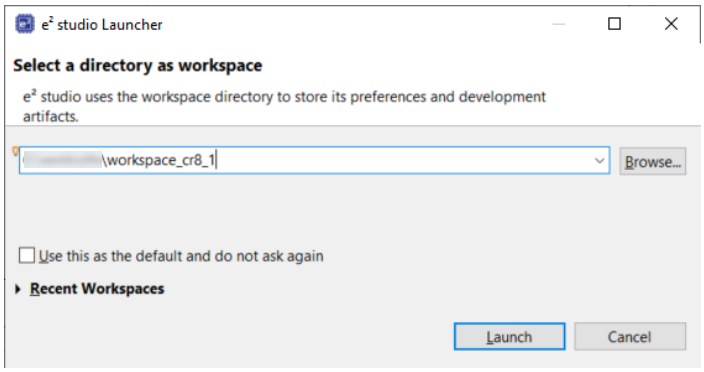


Figure 101 : Launching e2studio for CR8 Core1 Workspace

5. Select the **Debug Configuration** on e2studio of CR8 core1 workspace and click **Debug**.

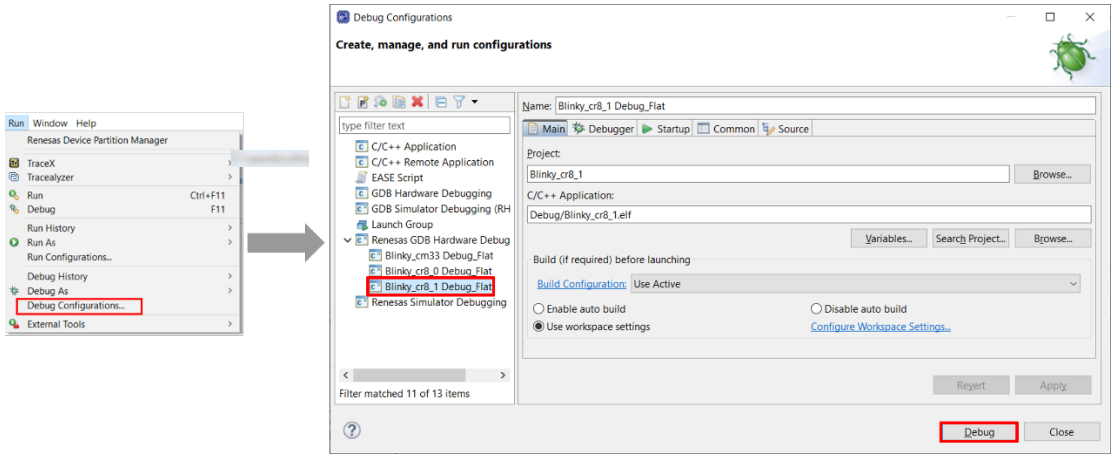


Figure 102 : CR8 Core1 Debug Configuration

6. You can debug each core project.

Revision History

Rev.	Date	Description	
		Page	Summary
2.01	Aug, 07 .24	6 to 20 29 to 32 39 to 43 45, 49, 57 to 59	Updated the description and figure based on the latest development environment.
2.00	May,31 .24	1	Added RZ/V2H to the target device.
		4 to 20 29 to 31 41 to 43 46	Updated the description and figure based on the latest development environment.
		25 to 28	Added description and figure for RZ/V2H EVK.
		40	Updated the description of project creation.
		44	Added the section of how to avoid resource duplication description in the case of RZ/V2H project creation.
		47 to 48	Added description about the Pins tab.
		58 to 62	Added description about the multi-core debugging of RZ/A1H.
1.10	Jan,31.23	5 to 10	Updated e2sutio version to install for Windows PC and changed images of e2studio installation
		16 to 20	Updated e2studio version to install for Linux PC and changed images of e2studio installation
		23 to 24	Updated pack version to install
		34 to 36	Changed Chapter 4.6 to 4.9 to a sub-chapter of Chapter 4.5 (These have been changed to Chapters 4.5.1 to 4.5.4)
		35, 49	Updated the method of specifying Secure Vector Address and Non-secure Vector Address
1.00	Jan.14.22	-	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/.