User's Manual

Renesas e^2 studio 2021-07 or Higher

User's Manual: Quick Start Guide

Renesas MCU
RE Family

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.1.00 Jul 2021

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/.

© 2021 Renesas Electronics Corporation. All rights reserved.

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.

Table of Contents

1. Overview 1
1.1 System Configuration 2
1.2 System Requirements 2
1.2.1 Hardware Environment 2
1.2.2 Operating Environment 2
1.3 Supported Toolchain 2
1.4 Supported Emulator 2

2. Installation 3
2.1 Installing e² studio and the Software Package 4
2.1.1 Installing e² studio 4
2.1.2 Installing GNU Arm Embedded Toolchain 14
2.1.3 Installing the Software Package for RE 14
2.2 Un-installation of e² studio IDE 14
2.3 Updating e² studio 15
2.4 Updating Software Package 15

3. Project generation 16
3.1 Generating new Project with Smart Configurator 17
3.2 Generating new Executable Project without Smart Configurator 22
3.3 Generating and Using a RE Static Library 26
3.3.1 Creating a Static Library Project 26
3.3.2 Creating an Executable Project using the existing Static Library 30
3.4 Import Existing Projects into Workspace 36
3.5 Configuration Editor 41
3.5.1 Summary Page 42
3.5.2 BSP Page 43
3.5.3 Clocks Page 44
3.5.4 Pins Page 46
3.5.5 Stacks Page 49
3.5.6 Interrupts Page 56
3.5.7 Components Configuration Page 60

4. Building 61
4.1 Build Option Settings 61
4.1.1 Recommended Build settings 65
4.2 Build a Sample Project 67
4.3 Export Build Configuration Settings 69

5. Debug 70
5.1 Change Existing Debug Configurations 70
5.2 Create New Debug Configurations 76
5.3 Launch Bar 78
5.4 Basic Debugging Features 79
5.4.1 Breakpoints View 80
5.4.2 Expressions View 82
5.4.3 Registers View 84
5.4.4 Memory View 85
5.4.5 Disassembly View 87
5.4.6 Variables View 88
5.4.7 Eventpoints View 89
5.4.8 IO Registers View 92

5.4.9 Trace View 93
5.4.10 Memory Usage View 96

6. Help 99

1. Overview

Renesas e² studio is the Integrated Development Environment for Renesas embedded microcontrollers. e² studio is based on the industry-standard open-source Eclipse IDE framework and the C/C++ Development Tooling (CDT) project, covering build (editor, compiler, and linker control) and debug phases with an extended GNU Debug (GDB) interface support.

e² studio IDE includes a configurator for Software Package which is an optimized software package designed to provide easy to use, scalable, high-quality software for embedded system design.

e² studio IDE with the RE configurator provides multiple Graphical User Interface (GUI) wizards to auto-generate code, configure drivers, configure build and debug options, and run the applications created. Driver documentation is integrated in the form of tooltips, which are available in the code editor view.

This chapter describes the system configuration and operating environment for e² studio IDE to develop applications for the RE family series microcontrollers.

The outline of this document is shown in the following table. It is recommended to read part of chapter 3, 4 and 5 to easily create or import a project and run it.

In this document, Software Development Kit is referred to as SDK. The SDK can be configured using the FSP Configuration of e² studio 2021-07.

Table 1-1. Content of this document

Chapter	What user would learn	Guide to create and run a project
1. Overview	Development environment requirement	Δ
2. Installation	Steps to install IDE, especially RE development environment and toolchain	Δ
3. Project generation	Creating project with/without Software Package and configuring Software Package for RE MCU family	O (sections 3.1 and 3.4) Δ (Other sections)
4. Building	Configuring build options and building the project	O (sections 4.2) Δ (Other sections)
5. Debug	Using basic debugging features of e² studio	O (section 5.1) Δ (Other sections)
6. Help	Using some important items of Help menu	Δ

O: Recommended reading.

Δ: Optional reading.

1.1 System Configuration

Below is an example of a typical system configuration.

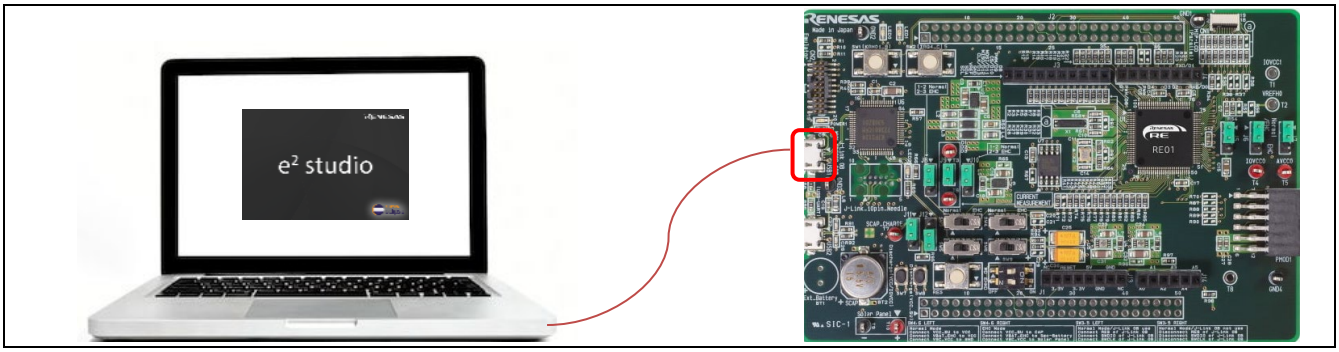


Figure 1-1 System Configuration

1.2 System Requirements

1.2.1 Hardware Environment

Processor:	At least 1GHz (support hyper-threading/multi-core CPU)
Main Memory:	At least 2GB of free memory space
Harddisk Capacity:	At least 2GB of free space
Display:	Resolution at least 1,024 x 768; at least 65,536 colors
Interface:	USB 2.0 (High-speed/Full-speed). High-speed is recommended.

1.2.2 Operating Environment

Architecture	Windows	e² studio
64-bit version	Windows 8.1, Windows 10	2021-07

1.3 Supported Toolchain

GNU Arm Embedded Toolchain (version: GCC V.6 GNU 6-2017-q2-update)

IAR C/C++ Compiler for ARM (version: 8.50.x or higher).

1.4 Supported Emulator

Segger J-Link, Segger J-Link OB, Renesas E2, Renesas E2 Lite

2. Installation

The table below shows the outline of this chapter. To set up the environment, it is recommended to read chapter 2.1. Please read the other sections when you need to update or uninstall the environment.

Table 2-1. Outline of this chapter

Chapter	Guide for Installation
2.1 Installing e² studio and the Software Package	O
2.2 Un-installation of e² studio IDE	Δ
2.3 Updating e² studio	Δ
2.4 Updating Software Package	Δ

O: Recommended reading,

Δ: Optional reading.

2.1 Installing e² studio and the Software Package

This section describes the installation of the following components.

- e² studio 2021-07
- GCC ARM embedded compiler
- Software Package for RE.

2.1.1 Installing e² studio

1. Download e²studio 2021-07 offline installer from https://www.renesas.com/e2studio
2. Run the e² studio installer to invoke the e² studio installation wizard page.
3. If e² studio was installed on your PC, the options to modify, remove the existing version and install e² studio to a different location will be shown. It is possible to install multiple versions of e² studio by selecting Install to a different location. Click the [Next] button to continue.

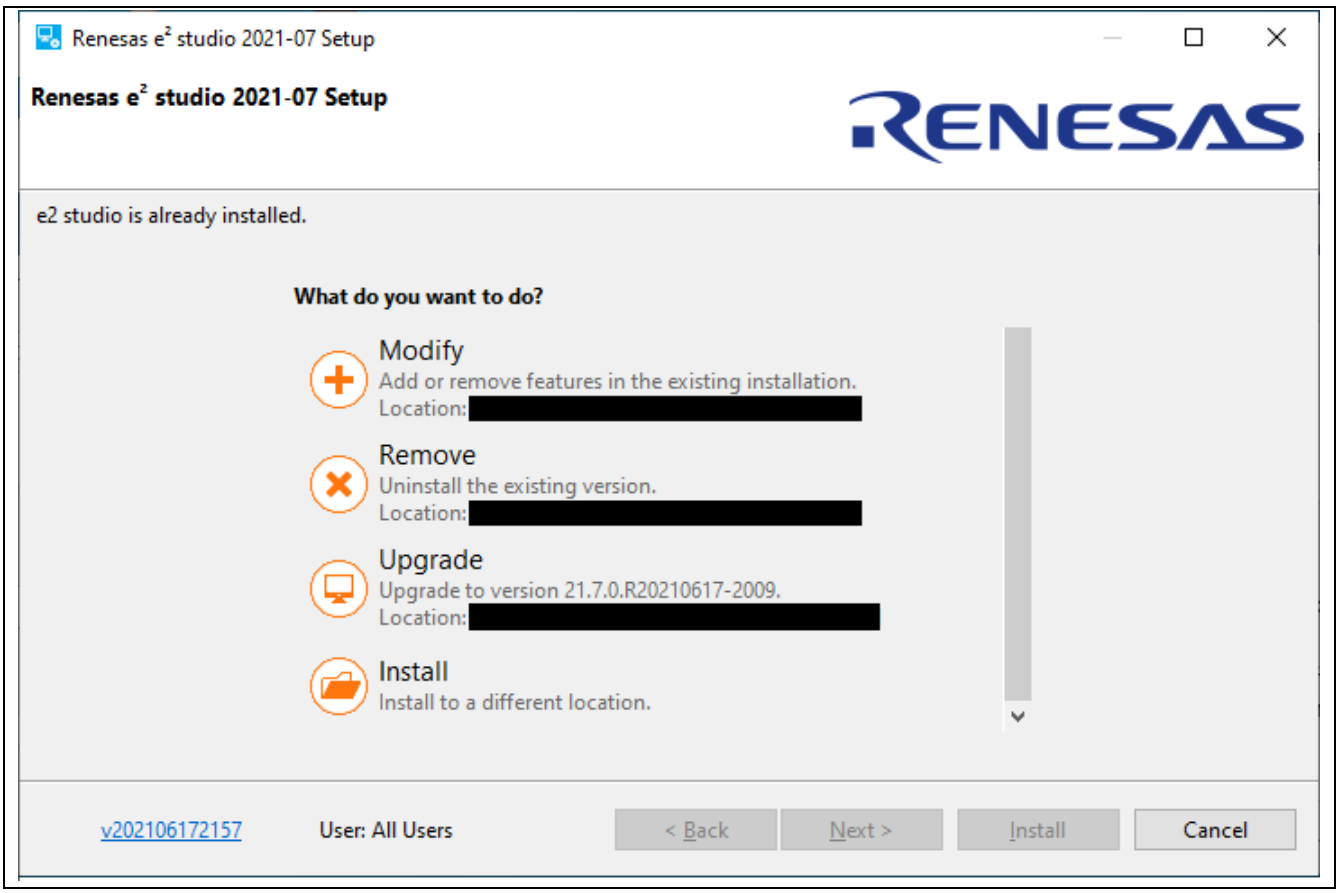


Figure 2-1 Installation of e² studio – Install Type

4. Welcome page:

   Users may use the default folder or change the folder by clicking [Change…]. Click [Next] to continue.

   Note: Multi-byte characters cannot be used for e² studio installation folder name, project name and its folder, and source file name

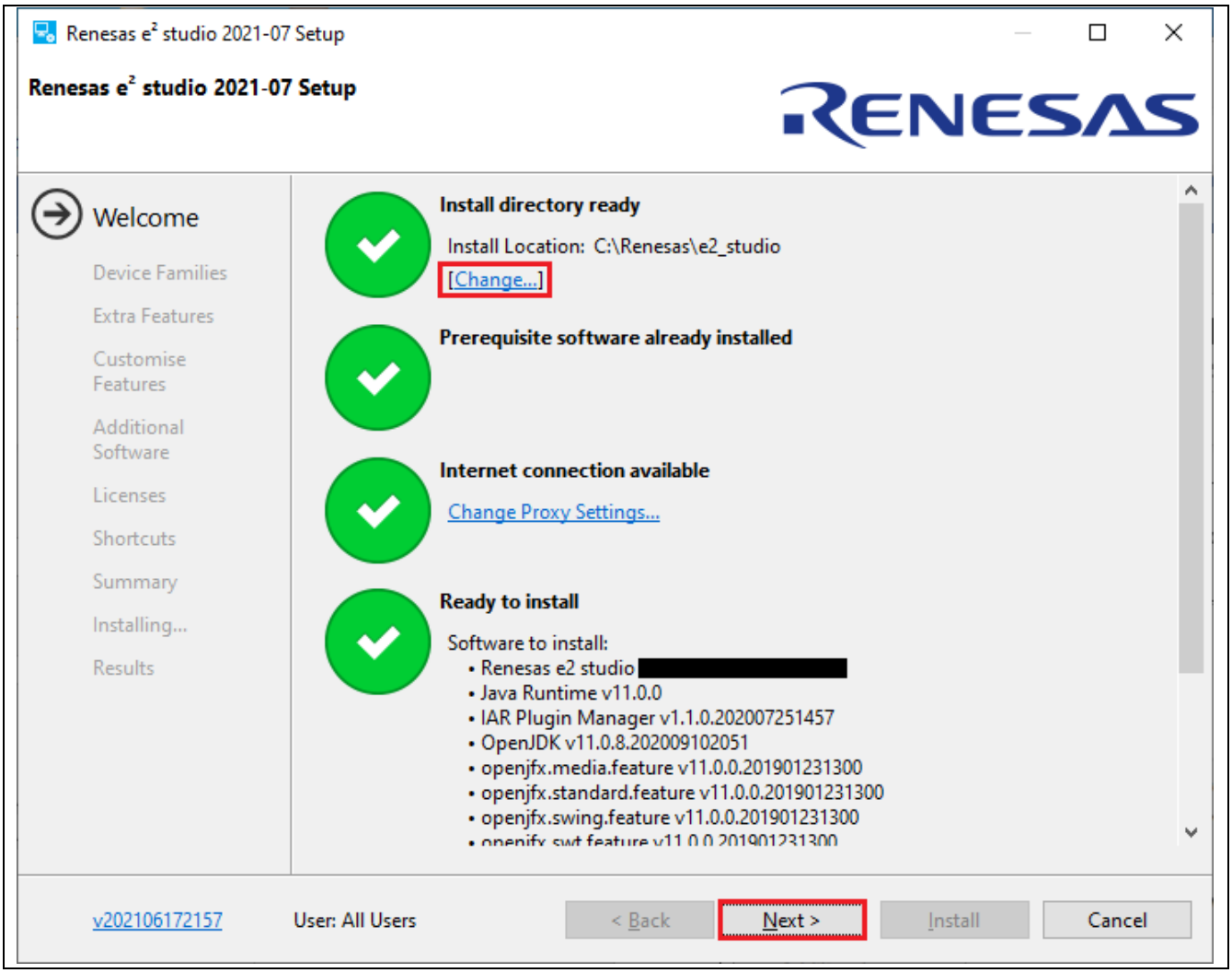


Figure 2-2 Installation of e² studio – Welcome page

5. Device Families page:

Select the checkbox for “RE”. Checkboxes of other device families are optional. Click [Next] to continue.

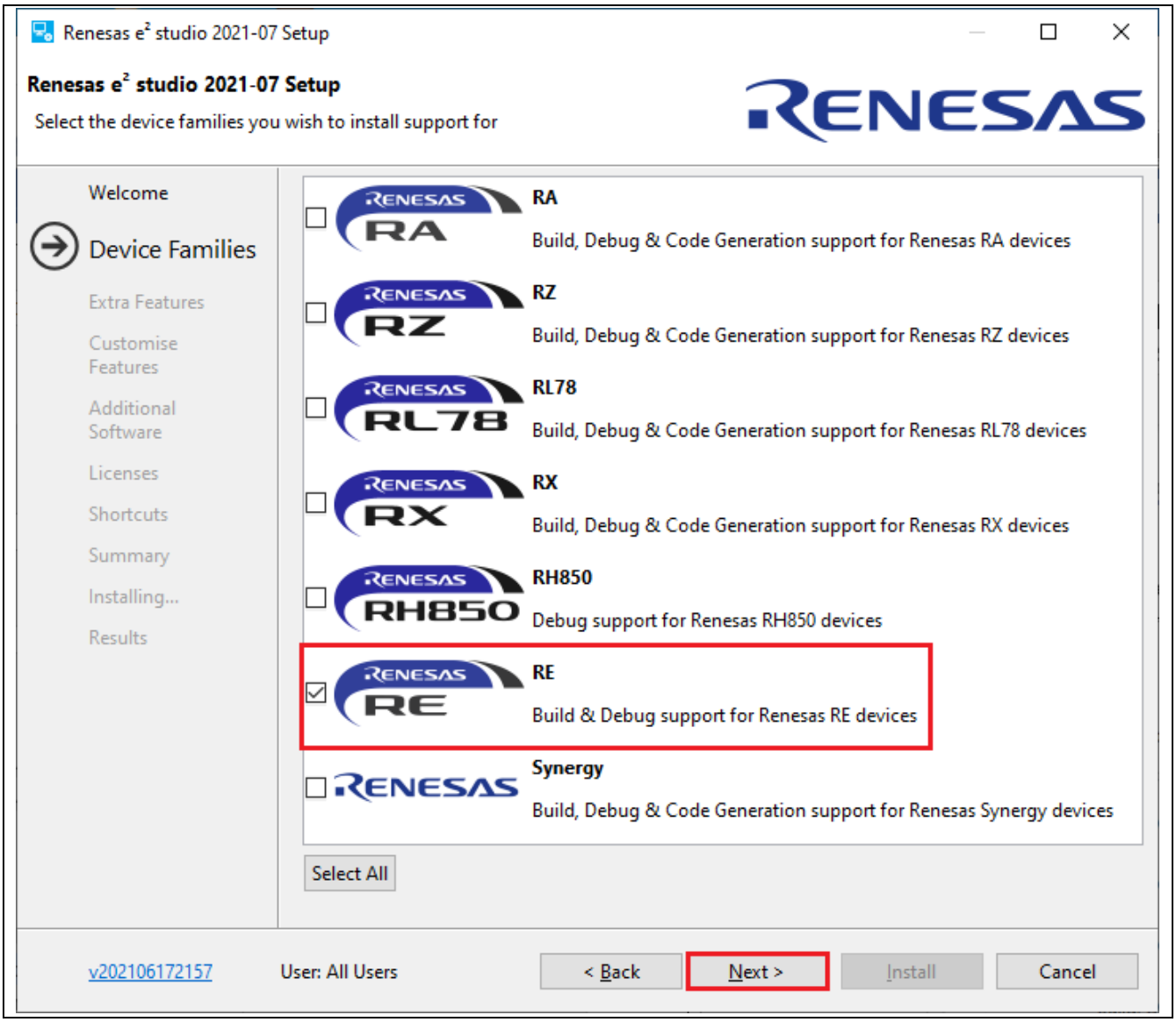


Figure 2-3 Installation of e² studio – Device Families page

6. Extra Features page:

   Select Extra Features (i.e. Language packs, SVN & Git support, RTOS support…) to install.
   For the non-English language menu, please select Language packs at this step.

   Click [Next] to continue.

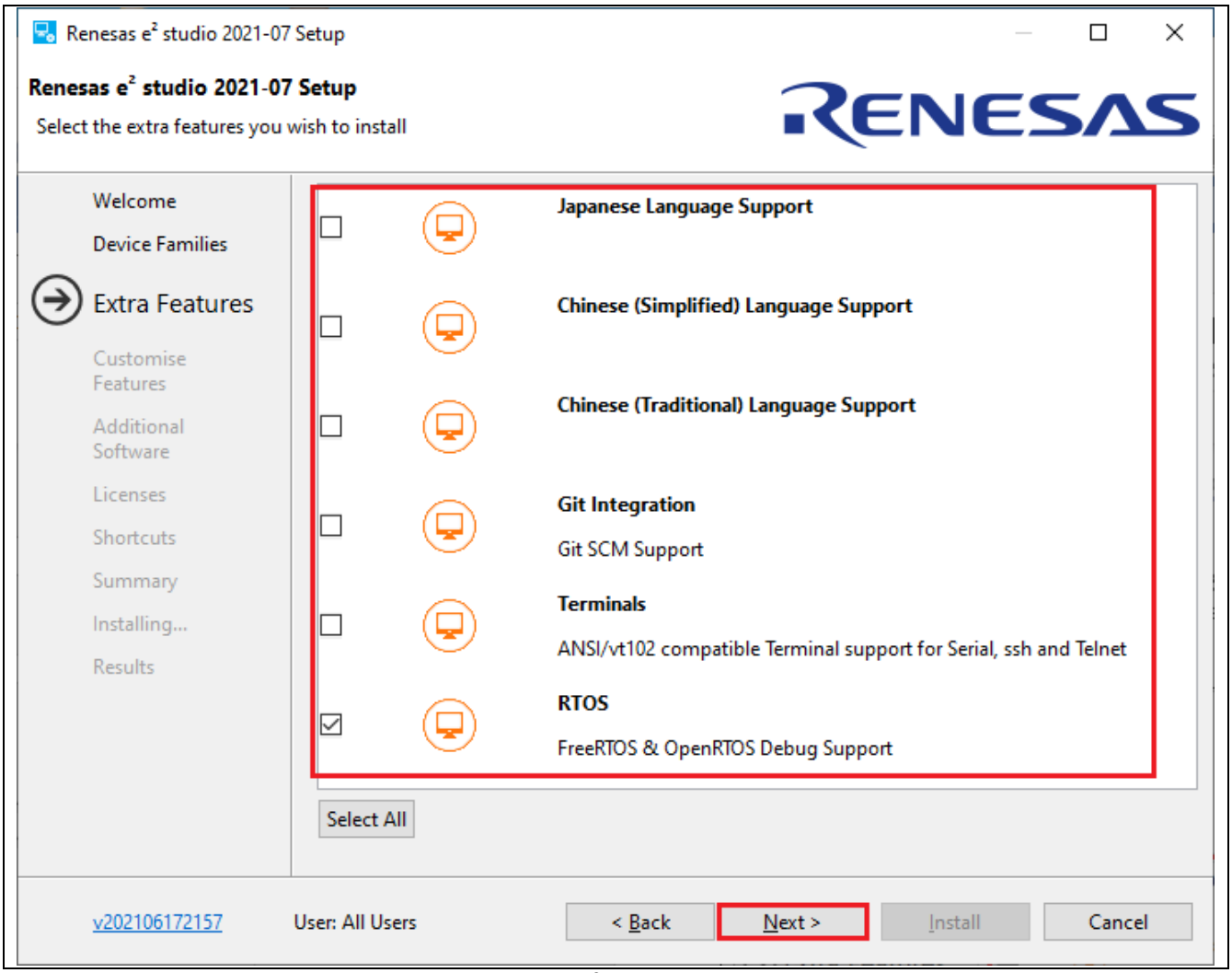


Figure 2-4 Installation of e² studio – Extra Features page

7. Customise Features page:

Ensure that “Renesas RE Family Support” and “Renesas FSP Smart Configurator” is checked. Click [Next] to continue.

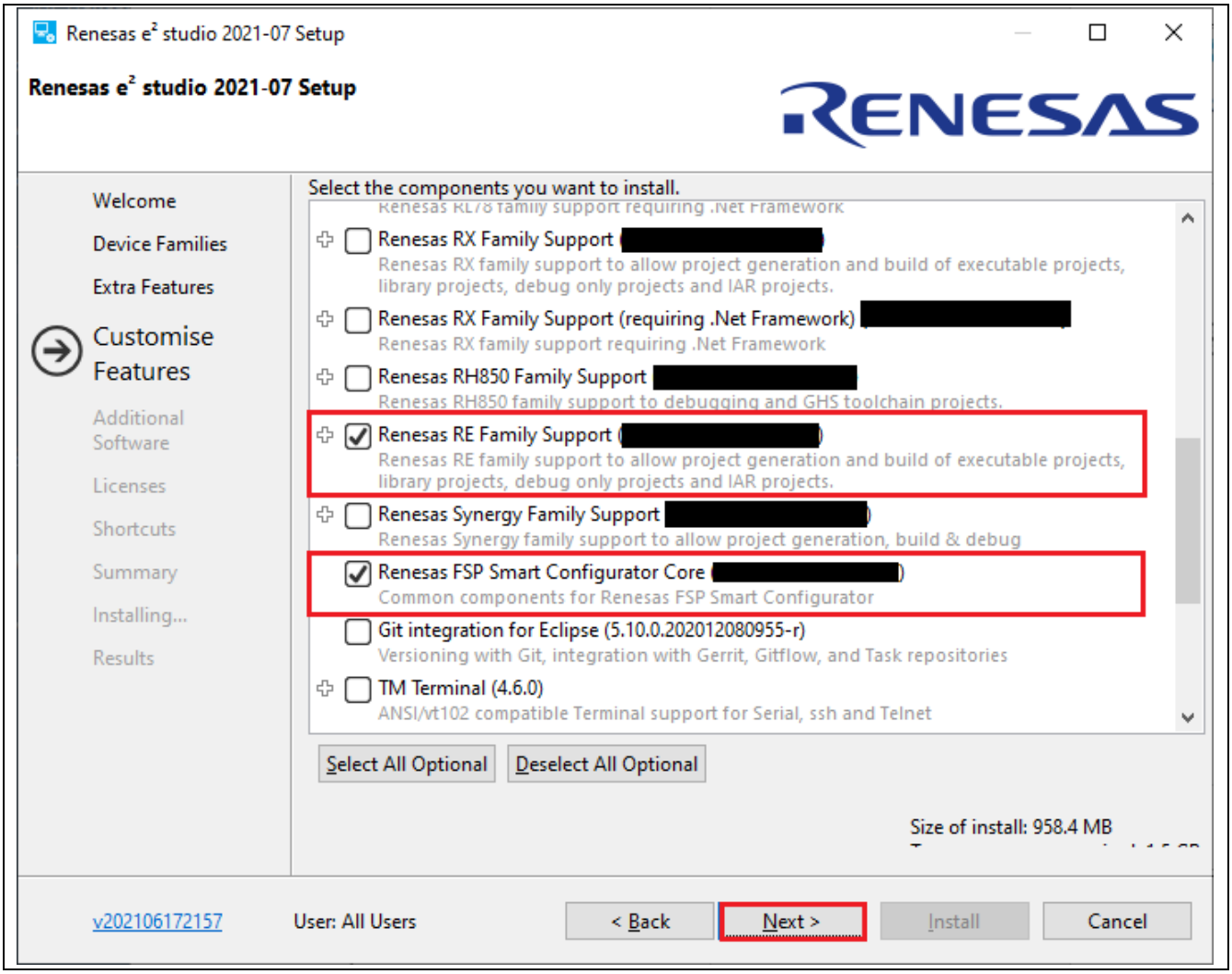


Figure 2-5 Installation of e² studio – Customize Features page

8. Additional Software page:

Select additional software (i.e. compilers, utilities, QE…) and click [Next] to continue.

Note: With no Internet access available, additional software installation can be skipped because the software catalog cannot be downloaded. The additional software can be installed later.

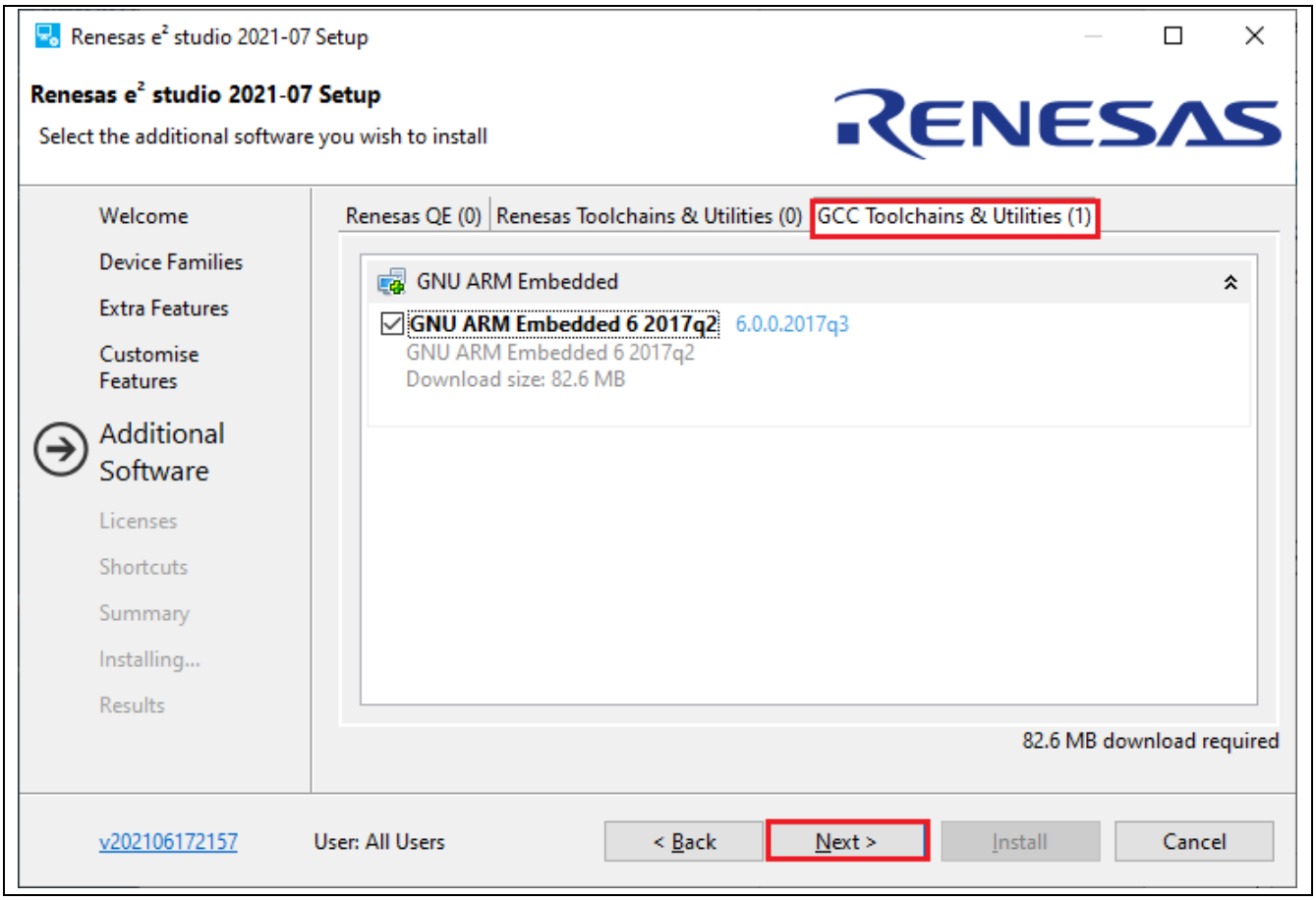


Figure 2-6 Installation of e² studio – Additional Software page

9. License page:

   Read and accept the software license agreement. Click [Next] to continue.
   Please note that user must accept the license agreement, otherwise installation cannot proceed.

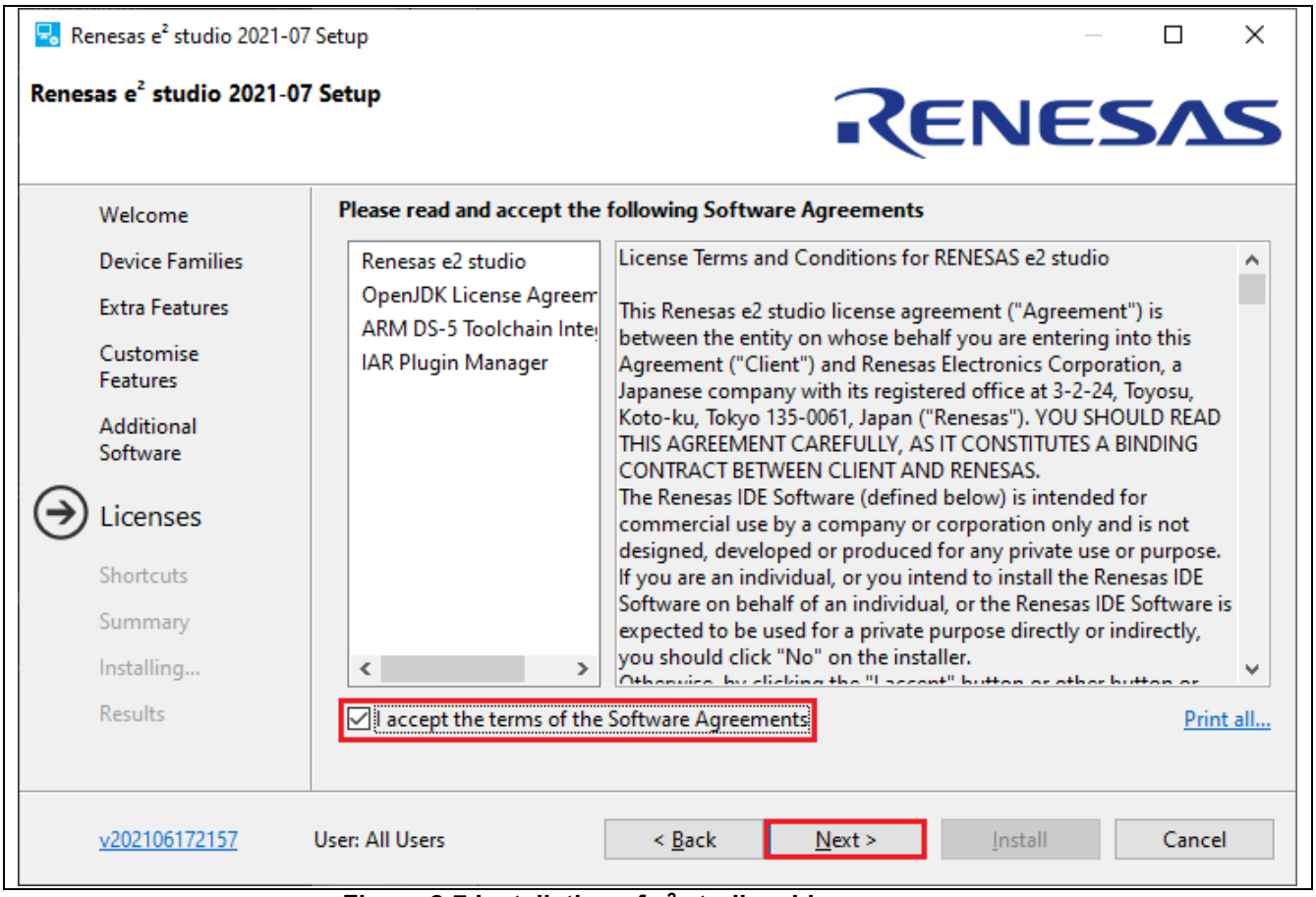


Figure 2-7 Installation of e² studio – Licenses page

7. Shortcuts

   Select shortcut name for the start menu and click [Next] 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).

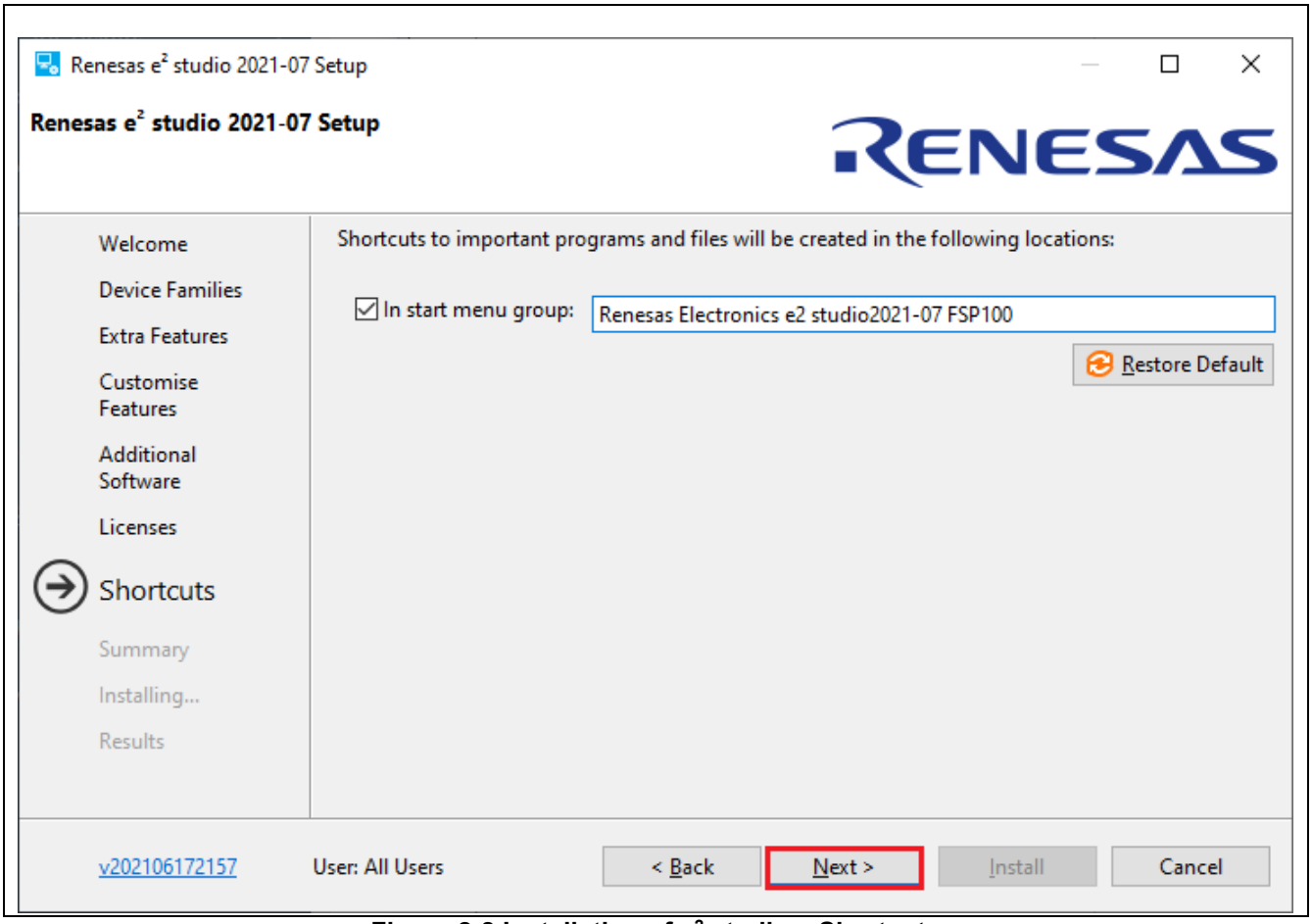


Figure 2-8 Installation of e² studio – Shortcuts

8. Summary page:

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

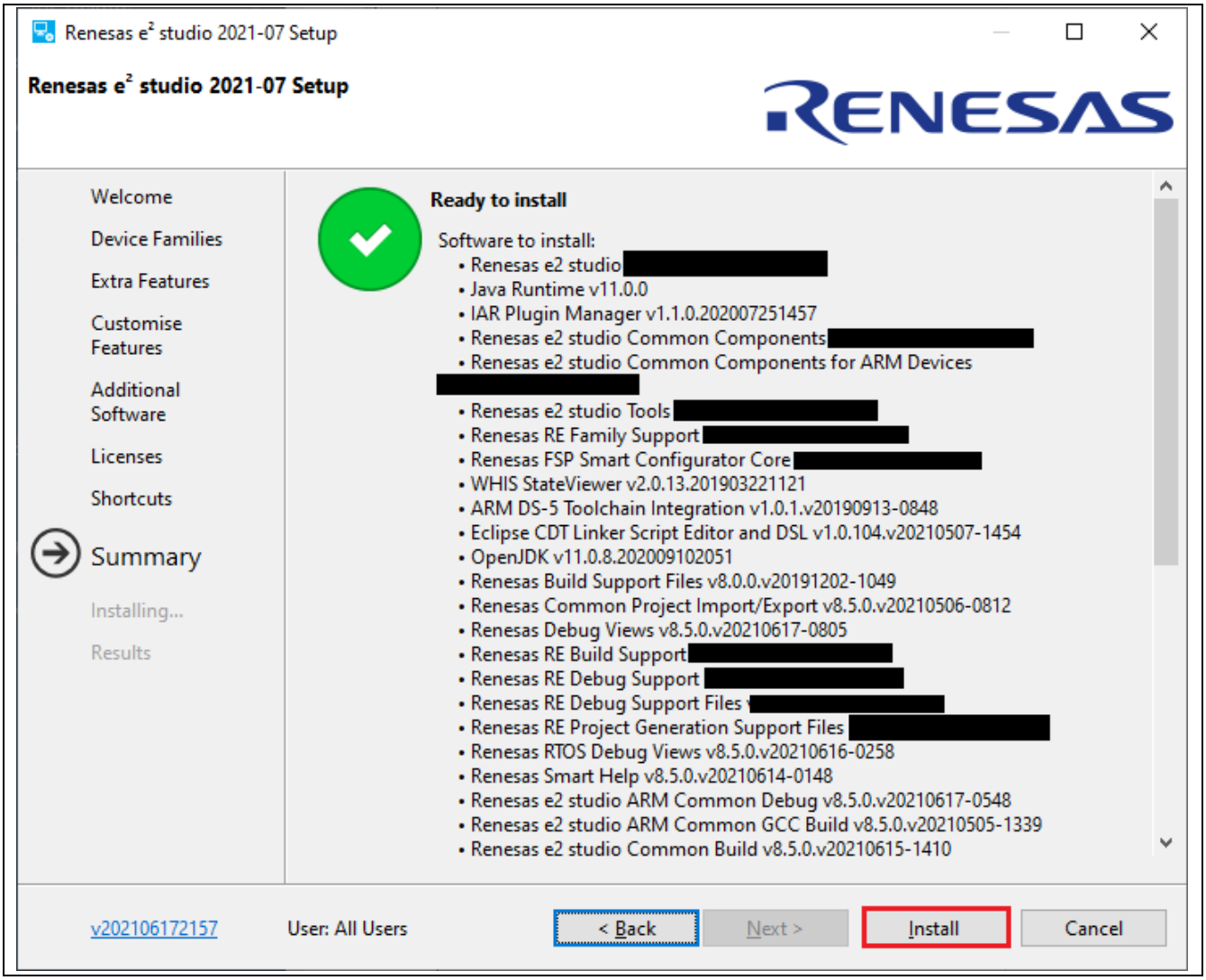


Figure 2-9 Installation of e² studio – Summary page

9. Installing… page:

The installation is now performed. Based on the selected items of Additional Software, new dialogs are opened to proceed with installation for these software items.

10. Results page:

Installation results are listed here. Please note if any errors are shown.

Click [OK] to complete the installation.

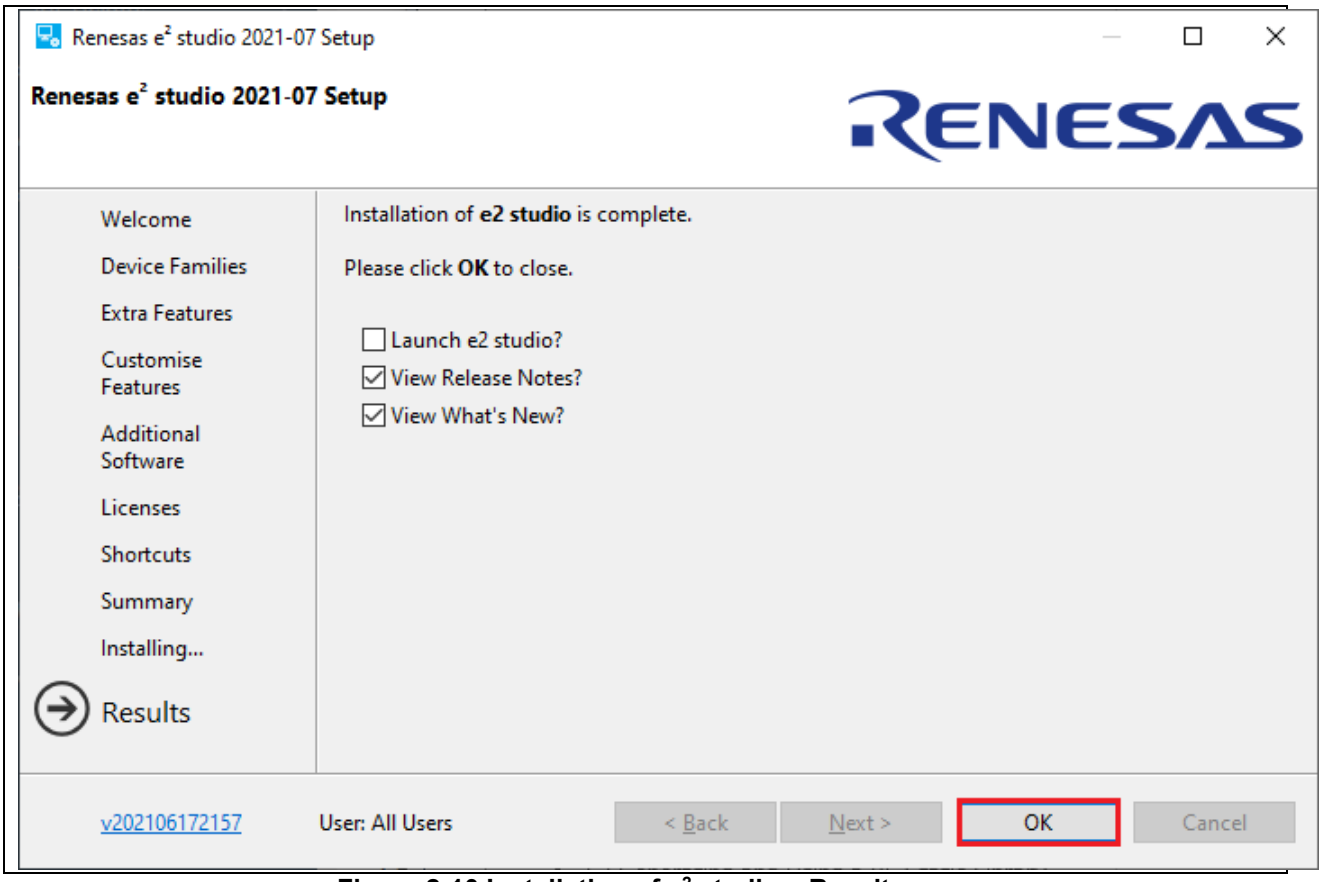


Figure 2-10 Installation of e² studio – Results

2.1.2 Installing GNU Arm Embedded Toolchain

The GNU Arm Embedded Toolchain can be installed during e² studio installation. Or after e² studio has been installed, the GNU Arm Embedded Toolchain can be installed separately.

To install GNU Arm Embedded Toolchain, follow these steps:

1. Download version 6-2017-q2-update of the GNU Arm Embedded Toolchain supported by Renesas RE (gcc-arm-none-eabi-6-2017-q2-update-win32.exe) from https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads
2. Run the installer to install the GNU Arm Embedded Toolchain on the host machine.
3. Select the installation language. Click [Yes] in the installation confirmation dialog.
4. Keep all default settings in the installation wizard.
5. When the “Install wizard Complete” dialog appears, check the box “Add path to environment variable”, “Add registry information” and “Launch gccvar.bat”. Then click [Finish] to complete the installation.

2.1.3 Installing the Software Package for RE

The Software Package can be installed during e² studio installation. Alternatively, after e² studio has been installed, the Software Package can be installed separately.

To install the Software Package, follow these steps:

(1) Visit the https://www.renesas.com/software-tool/re-software-development-kit
(2) Download the latest SDK (e.g. “r01an5970xx0110-re-sdk.zip ”). Unzip this file to get the latest software pack installer (e.g. “RE_SDK_Packs_<version>.exe”). The Software Package Installer includes the driver, HTML User's Manual.
(3) Make sure that a compatible e² studio was installed and closed during this installation.
(4) Run the Software packs installer and click [Next] to continue.
(5) Click [I Agree] to accept the agreement.
(6) Browse to the folder where e² studio is installed (e.g. C:\Renesas\e2_studio) and click [Install].
(7) Click [Finish] to finish the installation.

2.2 Un-installation of e² studio IDE

Users can uninstall e² studio program following the typical steps to uninstall a program in Window OS.

1. Click on [Start] → [Control Panel] → [Programs] → [Programs and Features]
2. From the currently installed programs list, right-click on “Renesas e² studio” and select the [Uninstall] button.
3. Click [Uninstall] to confirm the deletion in the “Uninstall” dialog.

At the end of the un-installation, e² studio IDE will be deleted from the installed location and Windows short-cuts menu are removed.

Note: If you have installed e² studio at multiple locations, you may not be able to find the uninstaller in “Program and Features” of Control Panel. In such cases, launch e² studio uninstaller located at: {e2 studio installed folder}/uninstall/uninstall.exe

2.3 Updating e² studio

To update e² studio, run the new version of e² studio installer (either *Software Package with e² studio* installer or standard e² studio installer). Please download the installer according to the instructions in Chapter 2.

Please note that you should not overwrite an existing installation. Before the IDE upgrade, users must uninstall the old version of e² studio. However, to keep both old and new e² studio versions, users can install the new e² studio version to a different location.

2.4 Updating Software Package

To update Software Package, run the new version of the Software Package installer. Download the installer according to Chapter 2.1.3.

3. Project generation

This chapter describes the creation of a new RE project. e² studio includes a wizard to help the quick creation of a new RE project. This is achieved by the ability of the wizard to match the project to a particular RE device and board.

The project generator can set up the pin configurations, interrupts, clock configurations and the necessary driver software.

As a pre-requisite, the Software Package and the toolchain must be installed on the host machine as described in chapter 2.

The following table shows the outline of this chapter. It is recommended to read chapters 3.1 and 3.4 to create or import a RE project in e² studio. Please read the other sections when you need information about creating project with Smart Configurator, creating a project without Smart Configurator and creating a static library project.

Table 3-1. Outline of this chapter

Chapter	What user would learn	Guide for project generation
3.1. Generating new Project with Smart Configurator	Creating a new project with Smart Configurator	O
3.2. Generating new Executable Project without Smart Configurator	Creating a new project without Smart Configurator	Δ
3.3. Generating and Using a RE Static Library	Creating a static library project with Smart Configurator and using it from an executable project	Δ
3.4. Import Existing Projects into Workspace	How to import an existing project into e² studio	O
3.5. Configuration Editor	Configuring the project with Smart Configurator	Δ

O: Recommended reading.

Δ: Optional reading.

3.1 Generating new Project with Smart Configurator

1. Click [File] → [New] → [Renesas C/C++ Project] → [Renesas RE] to open new project creation wizard.

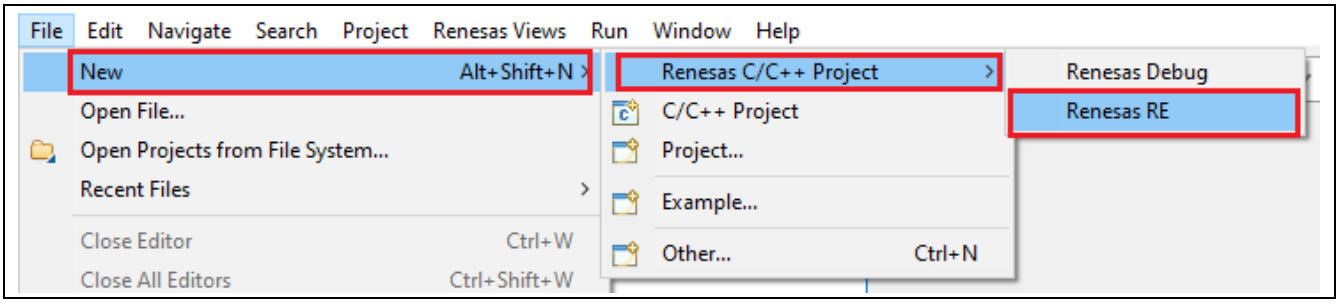


Figure 3-1 Open new project creation wizard

2. Select “All” “Renesas RE C/C++ SDK Project” template. Click [Next] to continue.

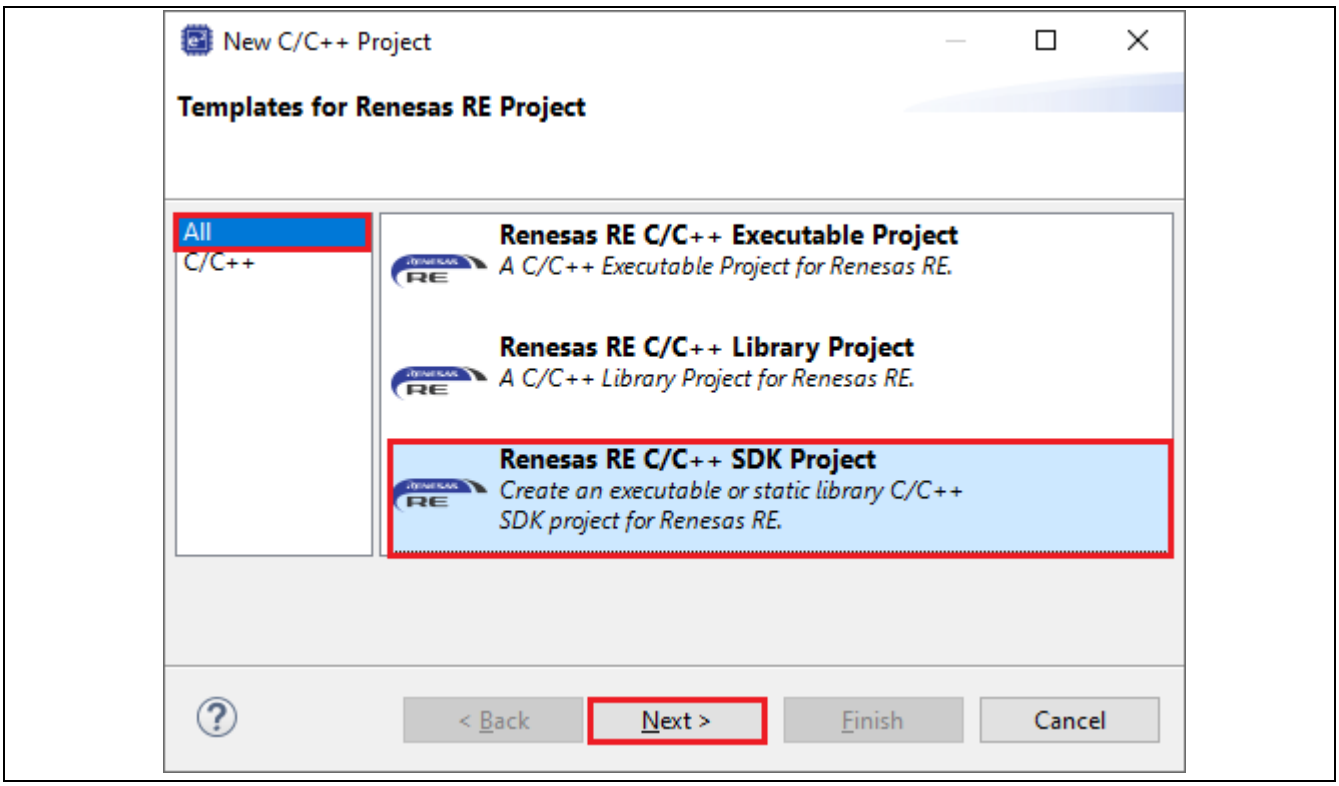


Figure 3-2 Project generation – Select template

3. Enter the project name. Click [Next] to continue.

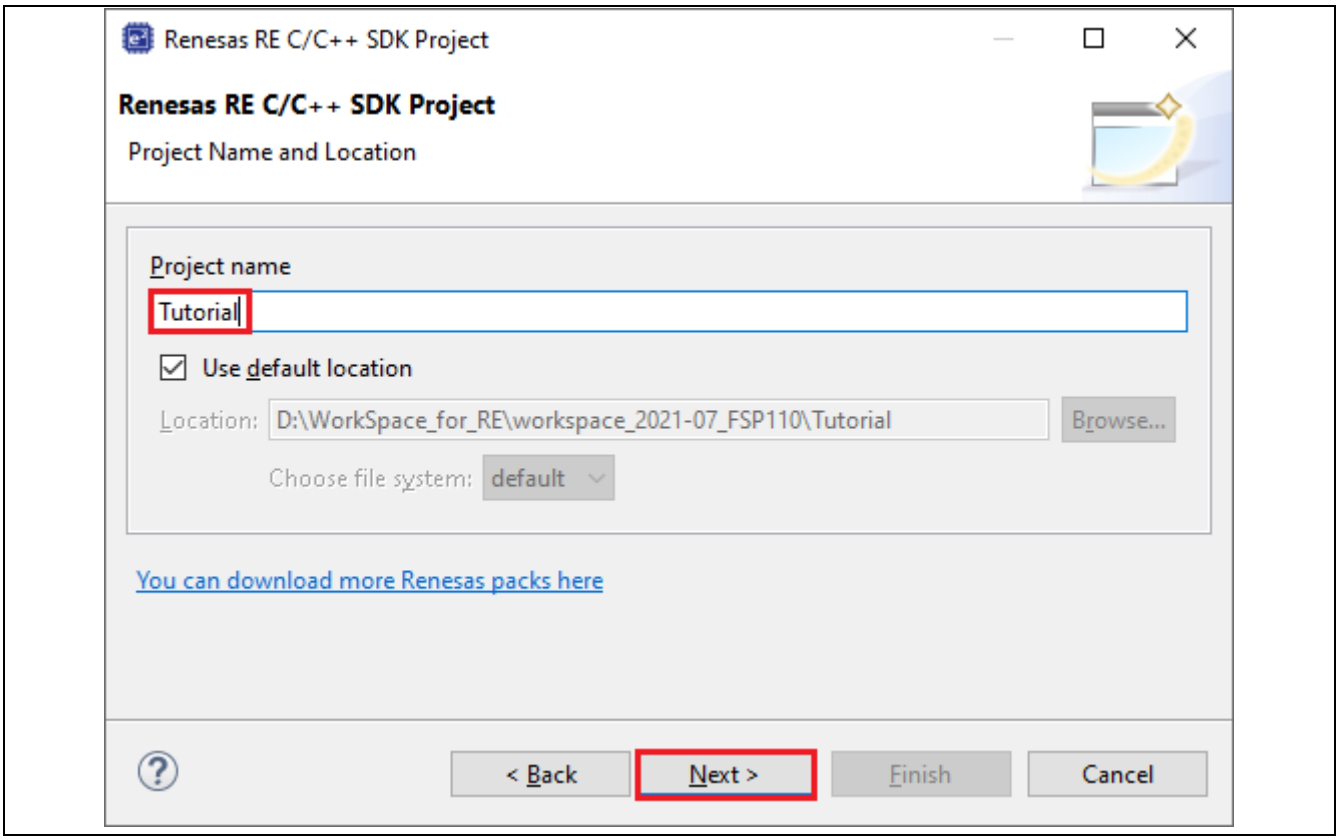


Figure 3-3 Project generation – Specify project name

4. In the device selection dialog, enter device and tool information:

   - Board: EK-RE01 1500KB
   - Toolchain version: Latest GNU Arm Embedded Toolchain approved for use with Renesas RE (e.g. GCC ARM Embedded 6.3.1.20170620)

   Keep other fields as default. Click [Next] to continue.

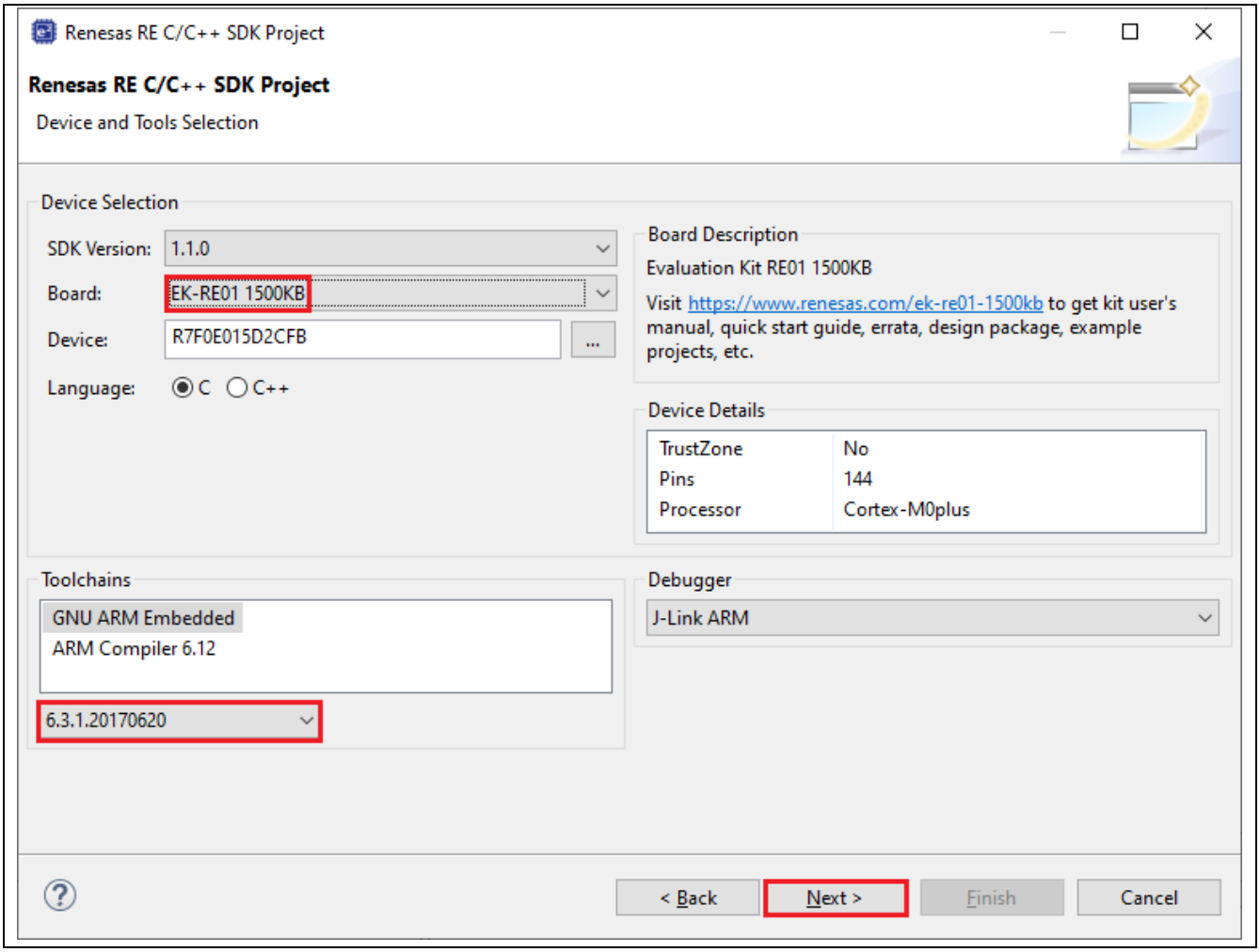


Figure 3-4 Project generation – Select device and tools

5. In the build artifact dialog, select the build artifact: executable or library. Click [Next] to continue.

Note: FreeRTOS is not available for RE now

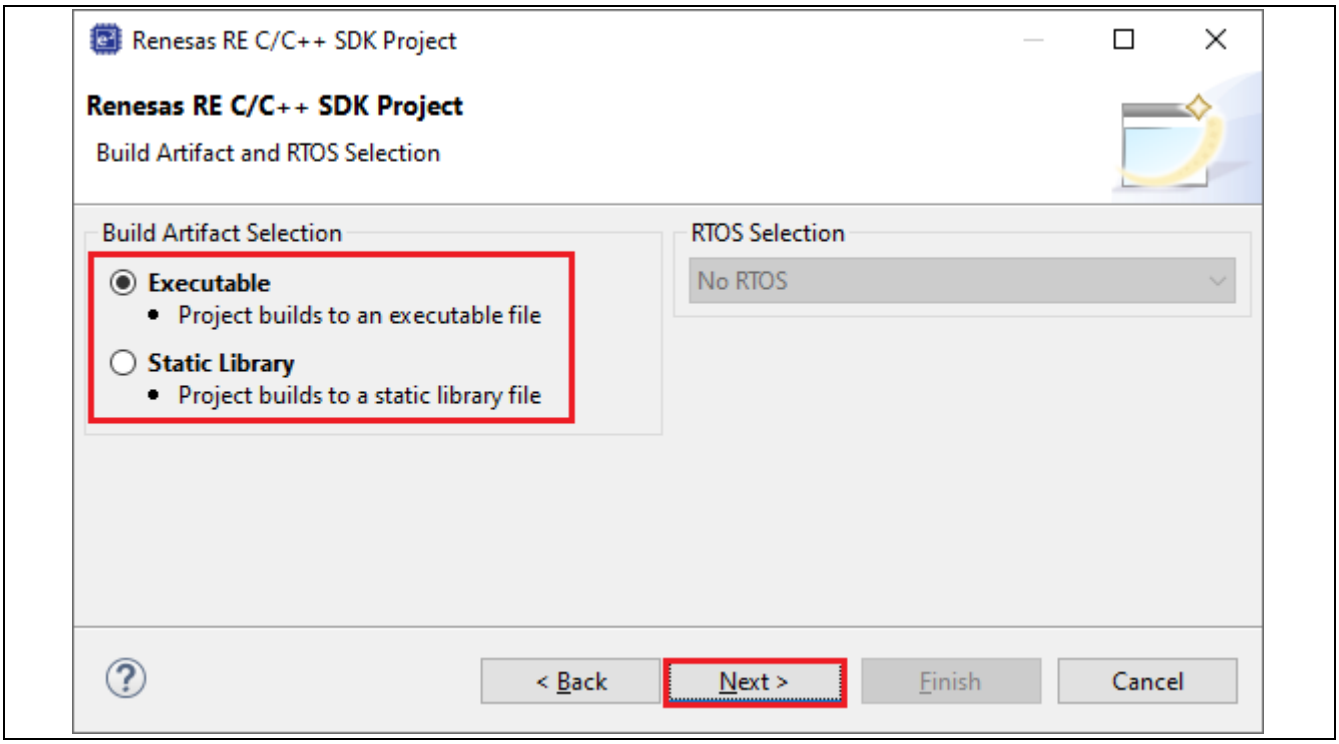


Figure 3-5 Project generation – Select build artifact

6. In the project template selection dialog, select the project template to be created:
   - Bare Metal – Blinky: RE project with BSP and LED blinking sample program.
   - Bare Metal – Minimal: RE project with BSP and no sample program.

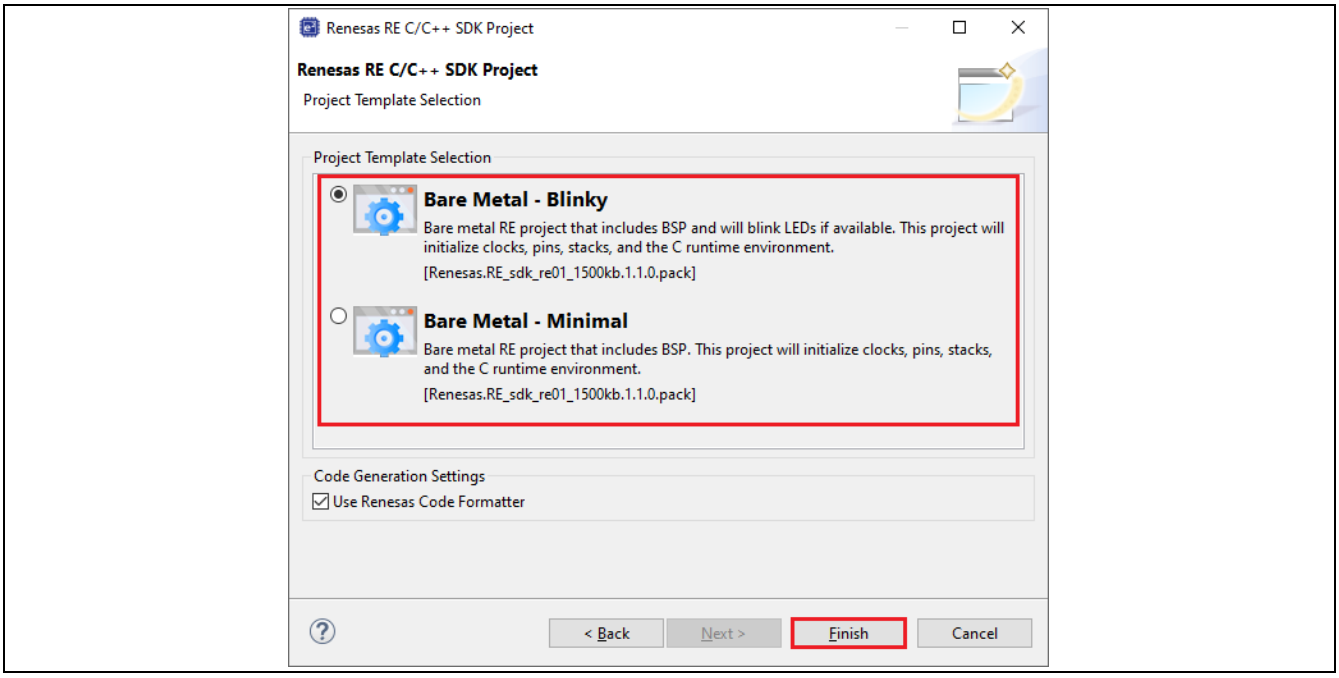


Figure 3-6 Project generation – Select project template

7. Click [Finish] to create the project.

You may be prompted to open the FSP Configuration perspective. Click [Yes] to open the perspective.

(In Eclipse, a 'perspective' is a predetermined arrangement of panes and views.)

e² studio creates a new project with various views, among them are the Project Explorer view, the Configuration editor and the Package view.

note: You must select the Pins tab in the Configuration Editor to see the package view.

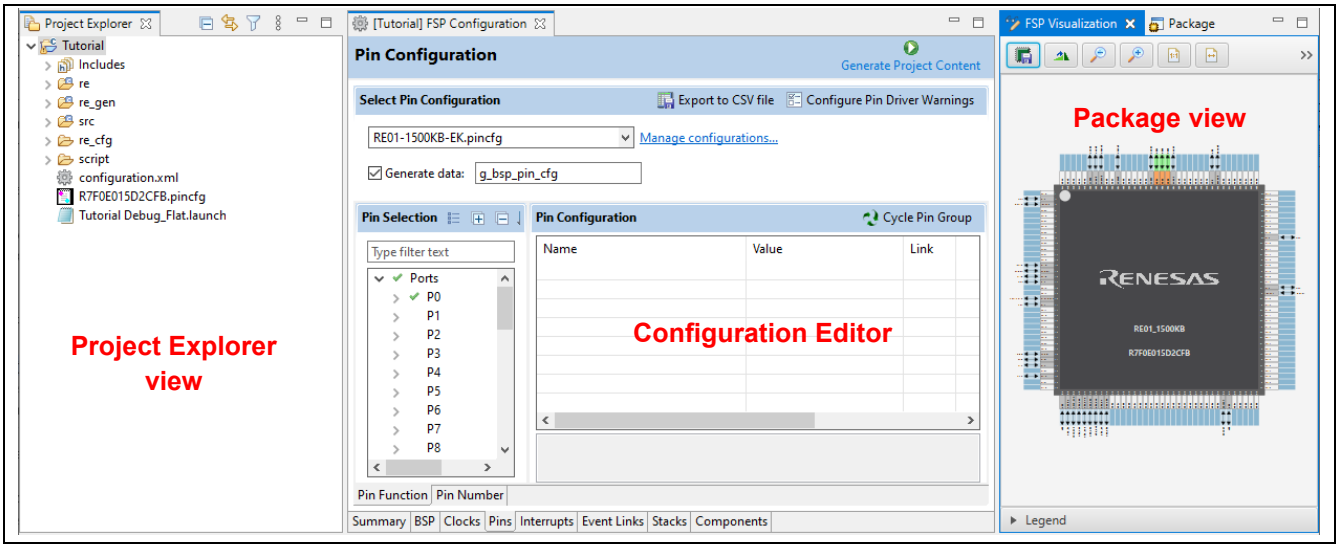


Figure 3-7 Project Generation – New Project Creation View

3.2 Generating new Executable Project without Smart Configurator

Note: It is recommended to use Software Package for projects using RE MCU Family. Smart Configurator enable easy project configuration and avoid the potential problems typically encountered during customized or manual configuration of the device.

Start the e² studio application and choose a workspace folder in the Workspace Launcher. To configure a new RE project, follow these steps:

1. Select [File] → [New] → [Renesas C/C++ Project] → [Renesas RE] to open the New Project creation wizard.

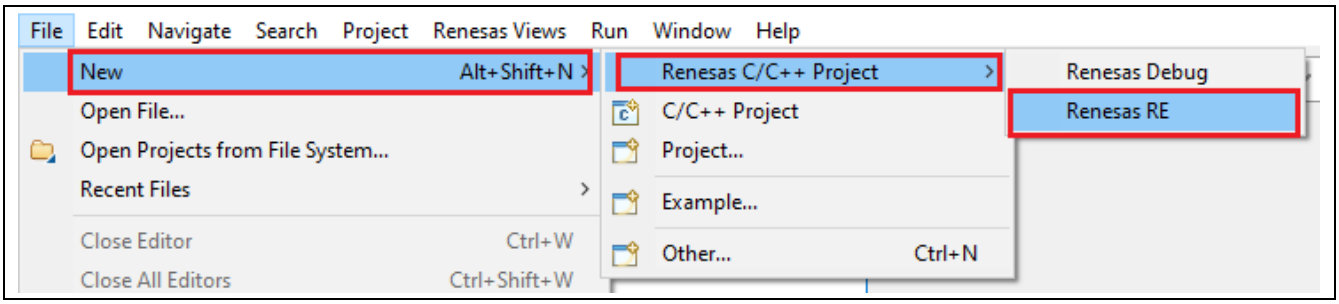


Figure 3-8 Open the New Project creation wizard

2. Select “All” “Renesas RE C/C++ Executable Project” template. Click [Next] to continue.

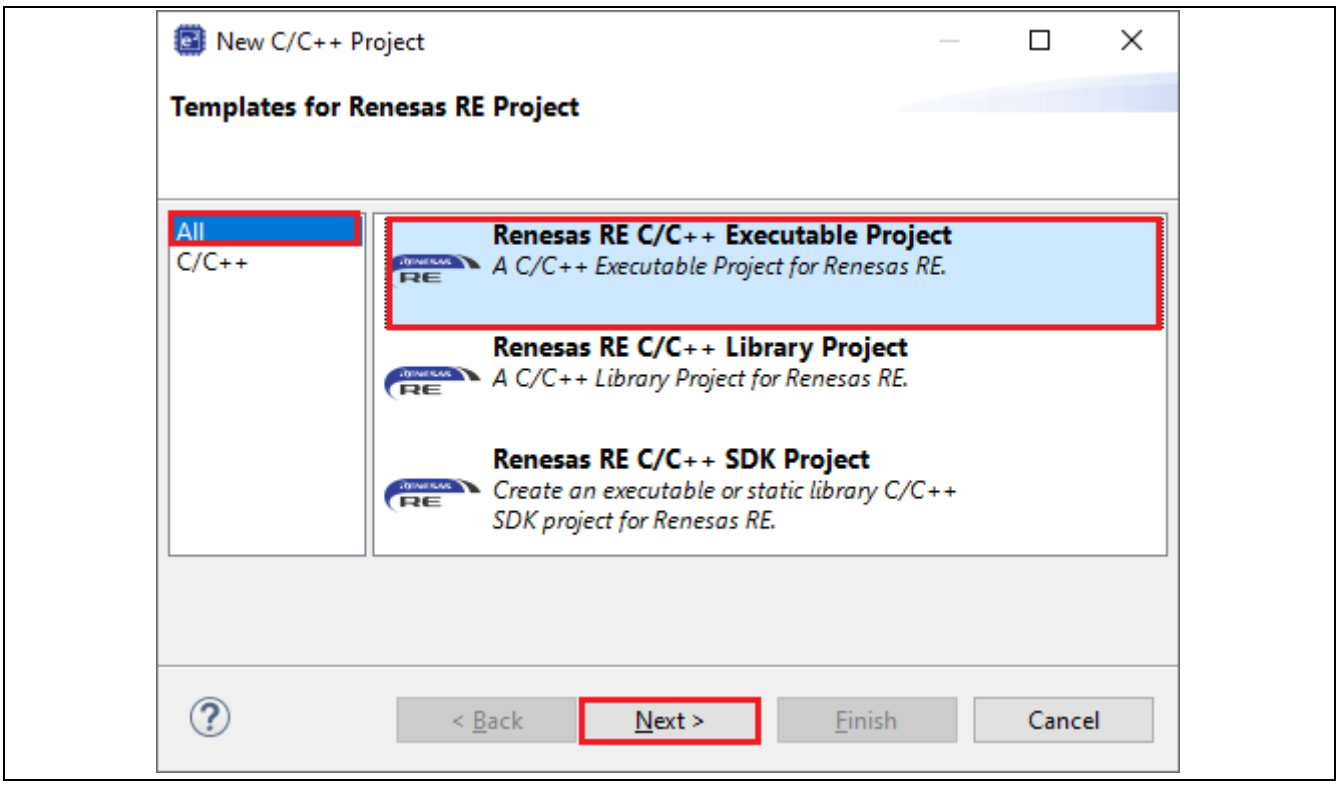


Figure 3-9 New Project Creation Wizard (1/6)

3. Enter the project name (e.g. Tutorial). Click [Next] to proceed.

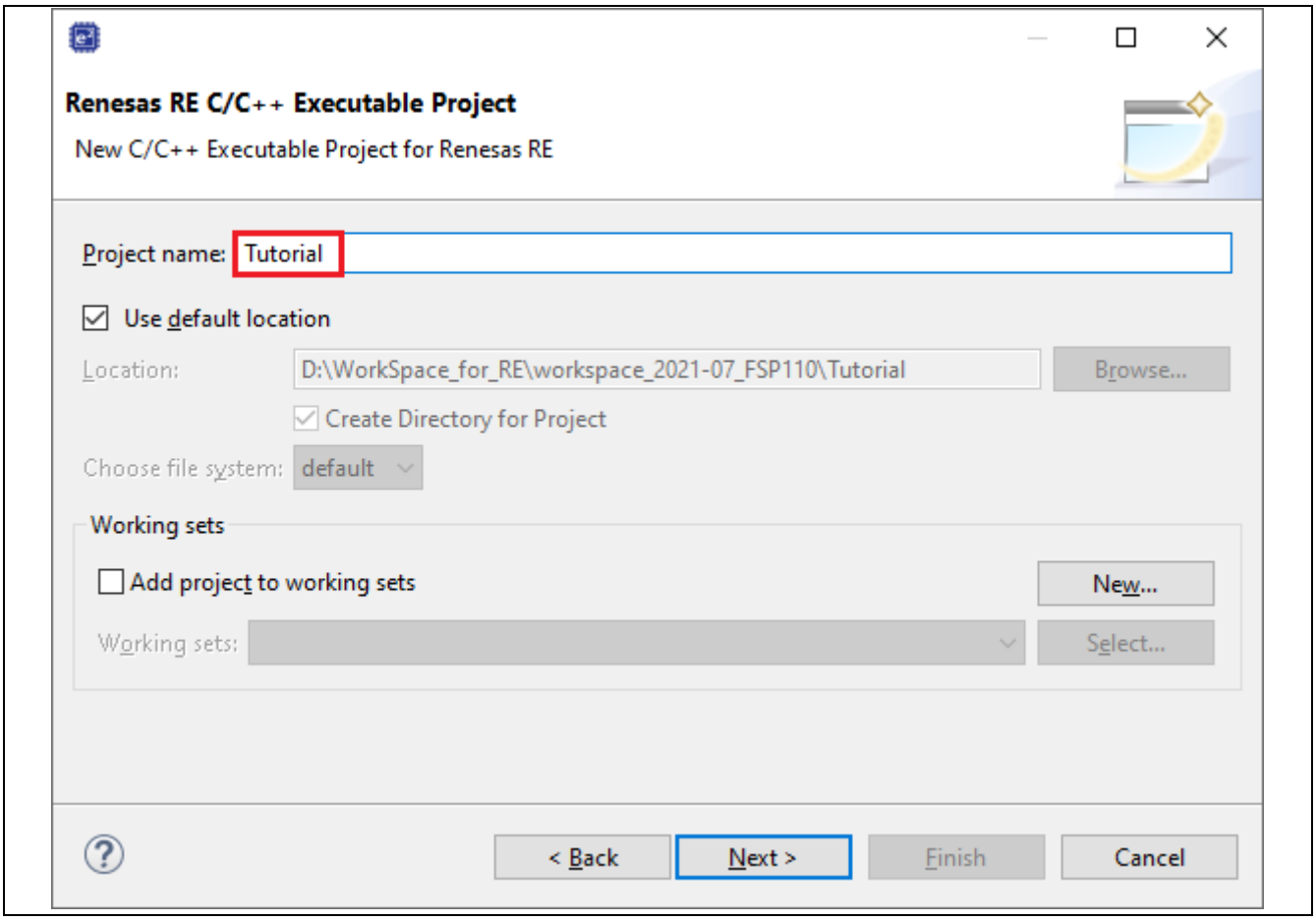


Figure 3-10 New Project Creation Wizard (2/6)

4. Enter device and tool information:

   - Target device: R7F0E015D2CFB (e.g.)
   - Toolchain version: Latest GNU Arm Embedded Toolchain approved for use with Renesas RE (e.g. GCC ARM Embedded 6.3.1.20170620)
   - Debugger: J-Link (ARM)
   - Keep all other fields as default and click [Finish] to generate the project

   Note: "E2" or “E2 Lite” can be selected in the same way as “J-Link” in the Hardware Debug Configuration pull-down menu.

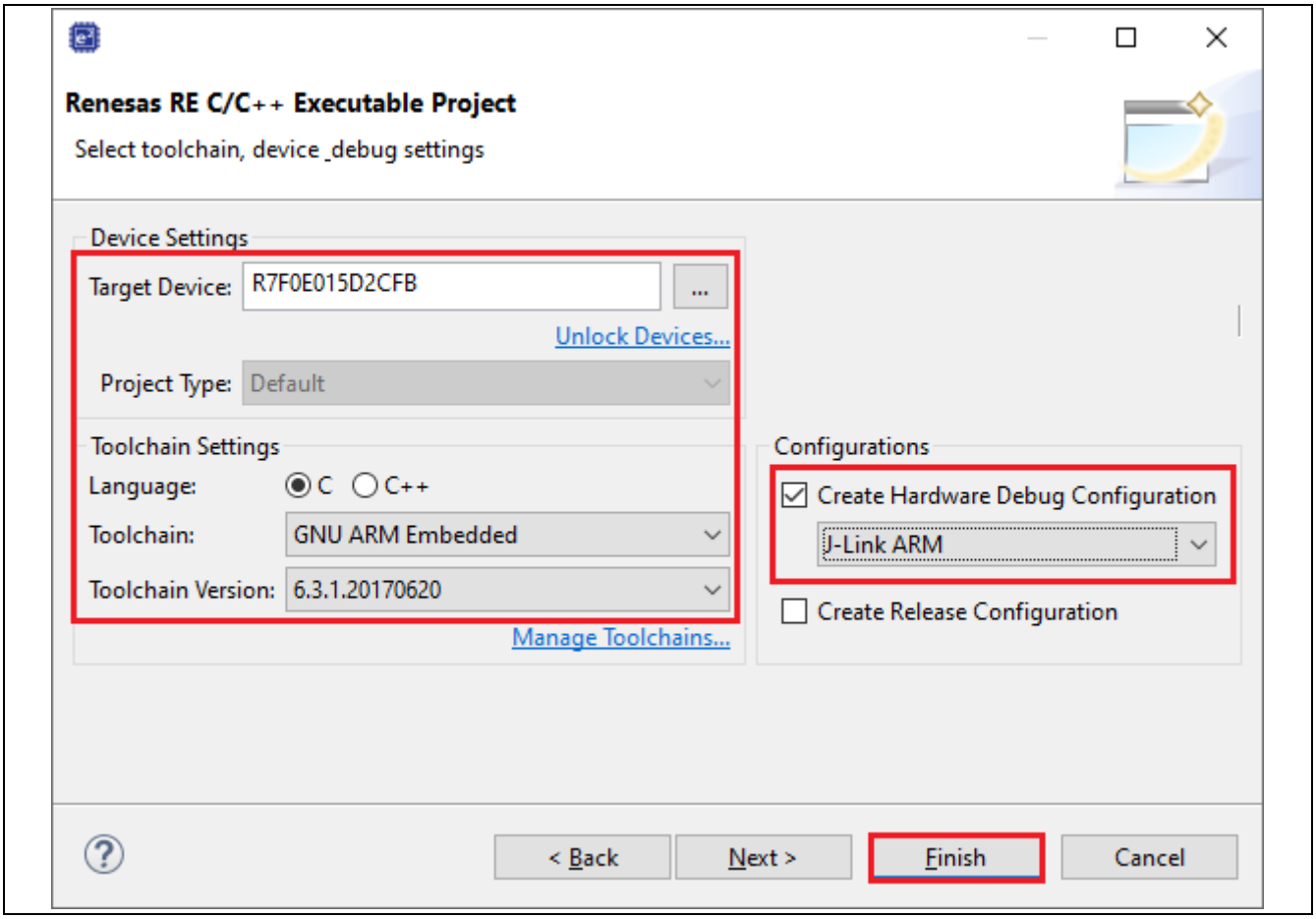


Figure 3-11 New Project Creation Wizard (3/6)

5. A new C project named “Tutorial” is created.

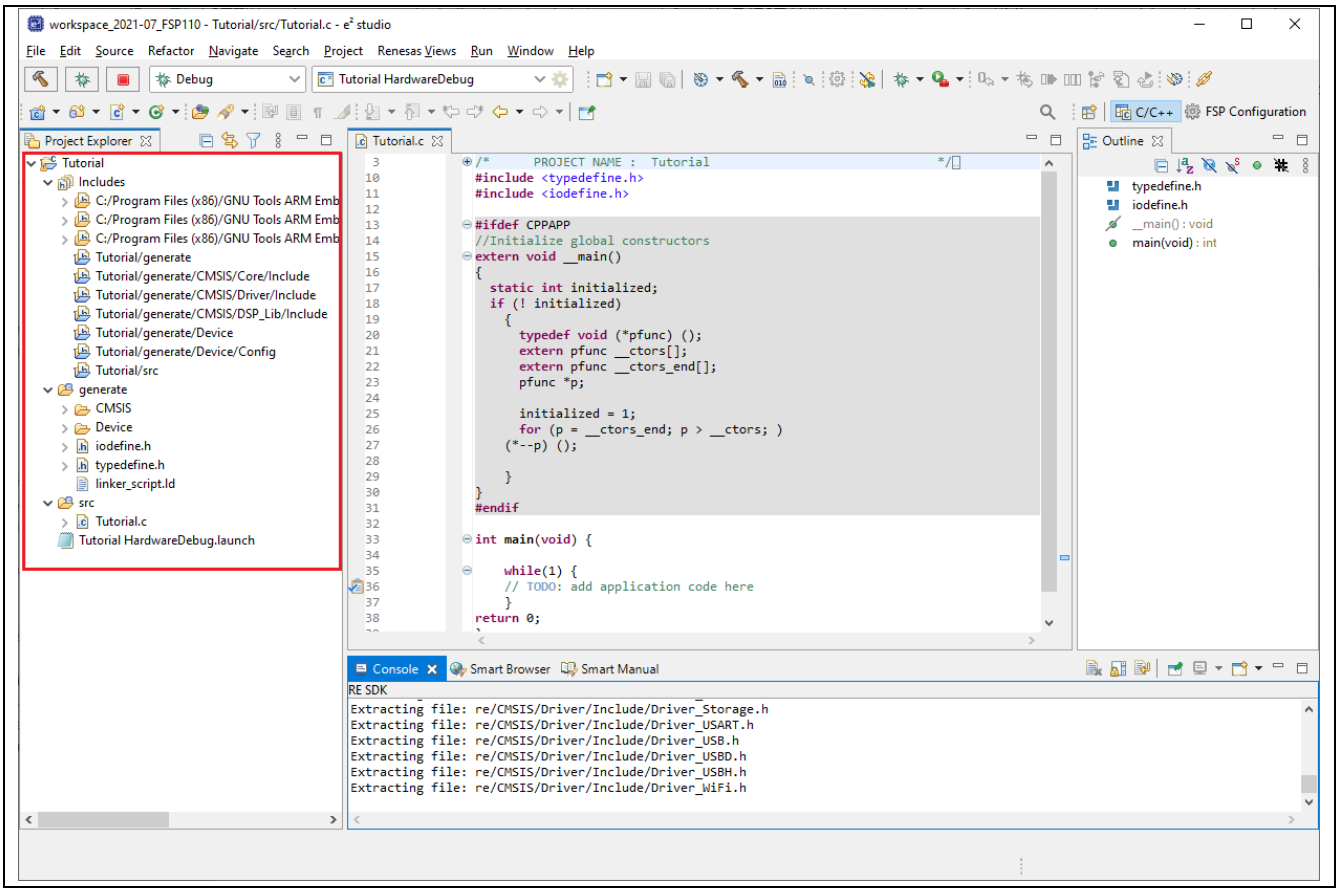


Figure 3-12 New C Project Created

3.3 Generating and Using a RE Static Library

This section describes how to generate a RE static library project and an executable project that references to the library project.

3.3.1 Creating a Static Library Project

The following steps show an example of how to create a RE static library project.

1. Generate a new SDK project as described in chapter 3.1 and use the following option:
   - Name the project (e.g. “RE_Lib”) at step 3,
   - Select build artifact as “Static Library” at step 5,

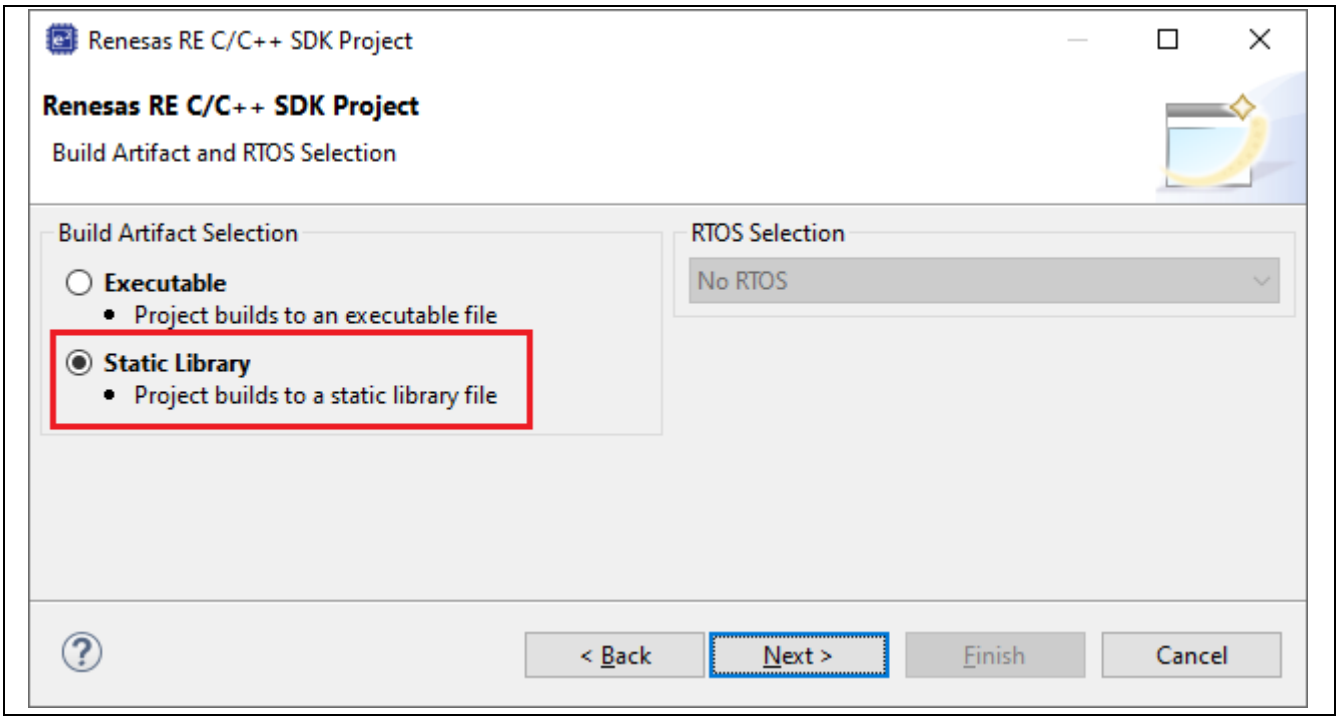


Figure 3-13 Select “Static Library” as Build Artifact

   - Select project template as “Bare Metal – Blinky” at step 6.

2. After the project is created, click [Generate Project Content] in the Configuration editor.

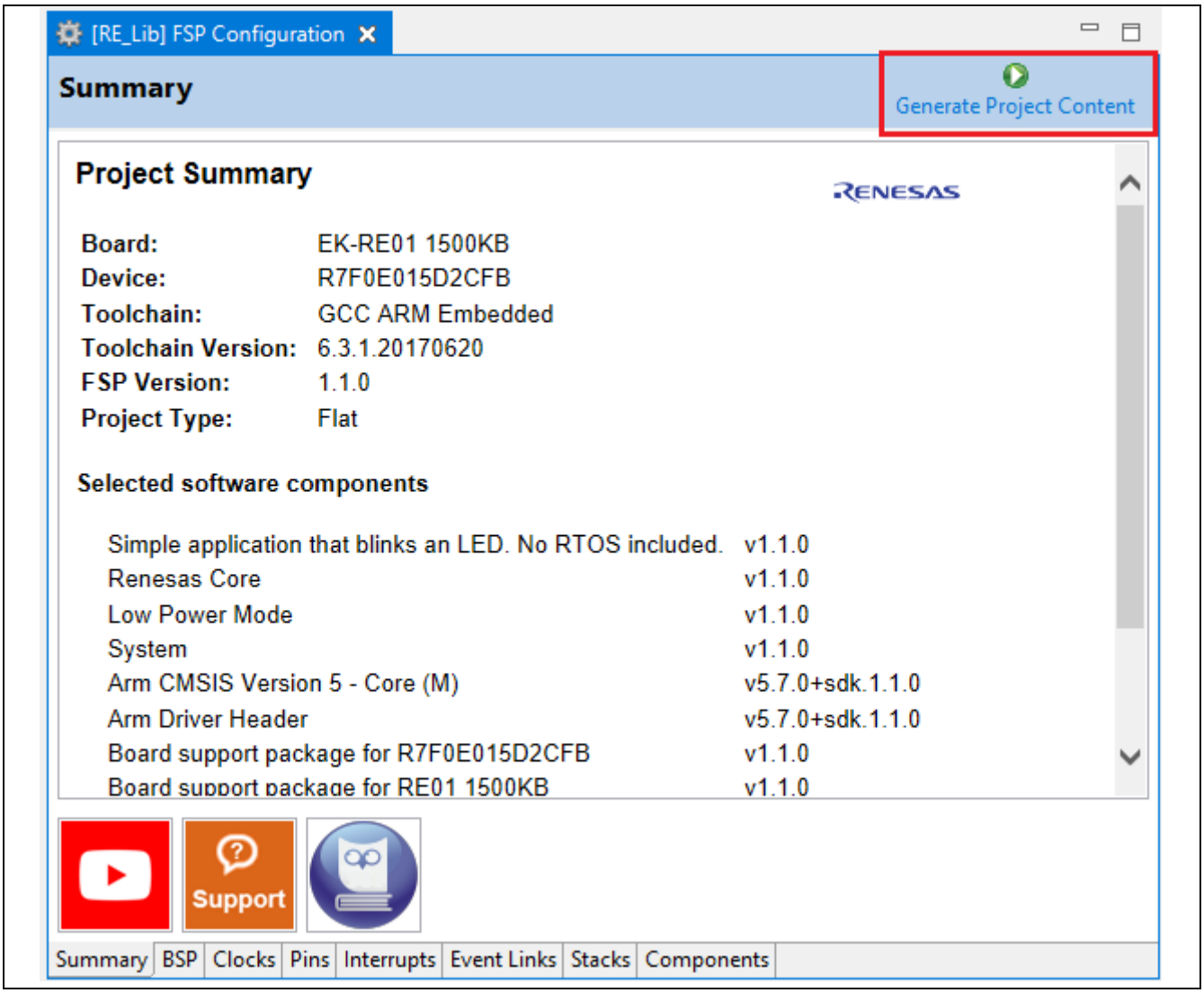


Figure 3-14 Generate library project content

3. From the project explorer window, open “hal_entry.c” under RE_Lib\src\

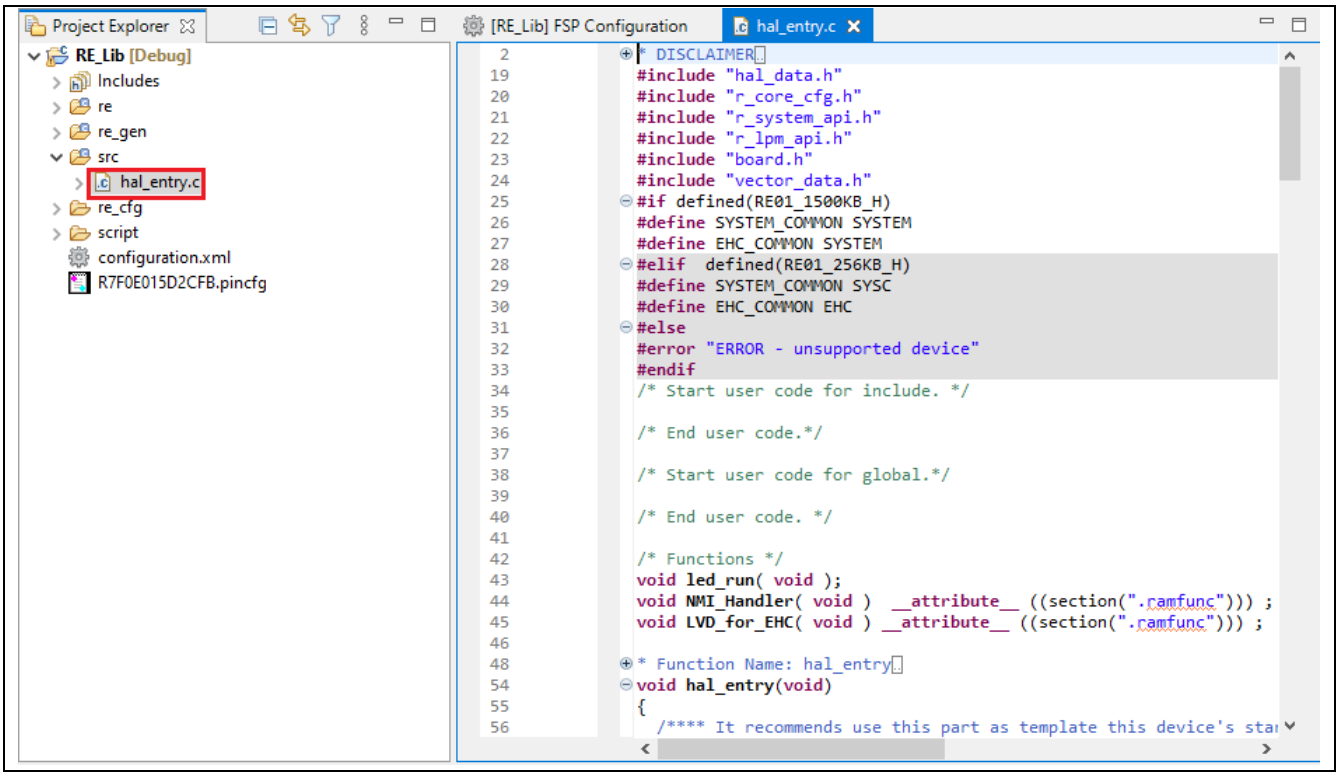


Figure 3-15 Original “hal_entry.c”

4. Rename the function hal_entry() to hal_entry_lib(), and add a declaration for hal_entry_lib().

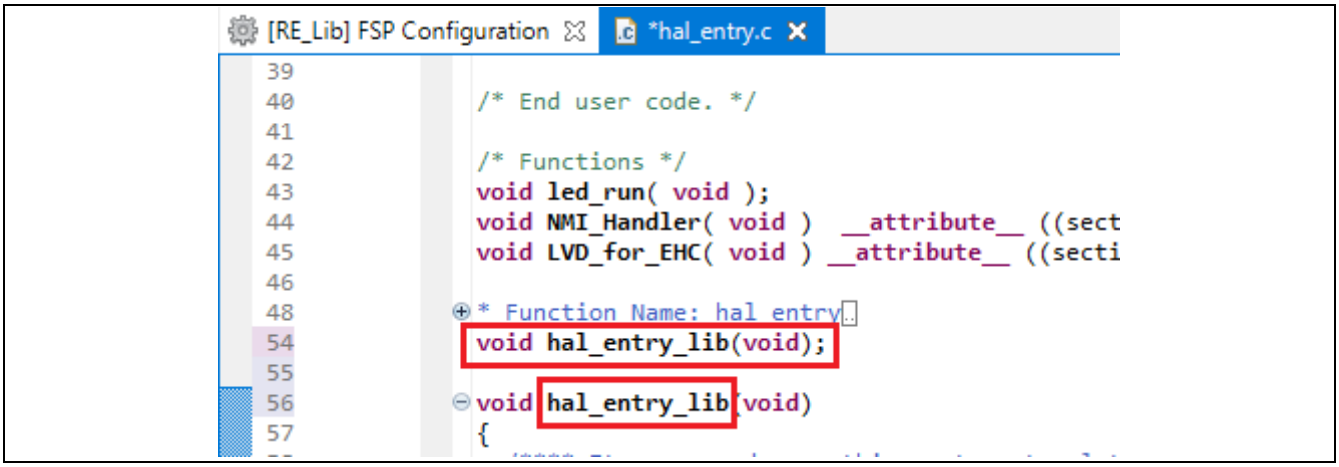


Figure 3-16 Modified “hal_entry.c”

5. Build the Library Project. The build outputs a static library file RE_Lib\Debug\“libRE_Lib.a”.

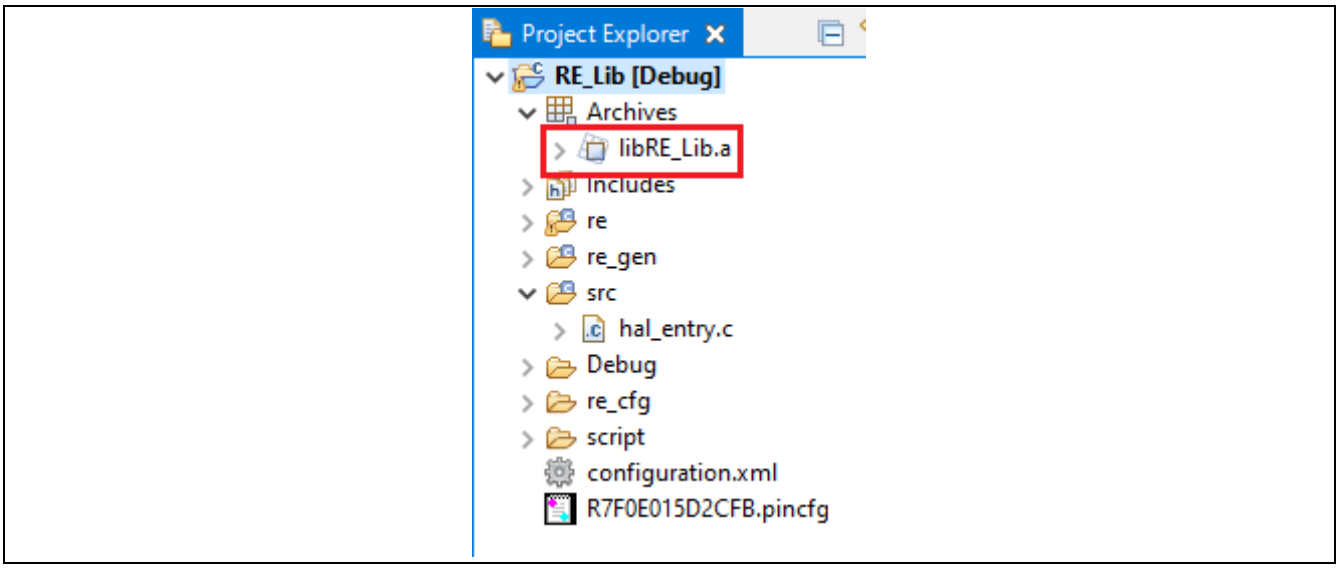


Figure 3-17 The built static library

3.3.2 Creating an Executable Project using the existing Static Library

This chapter shows how to use the static library created in the previous chapter (3.3.1) in a RE executable project by performing the following steps,

- Create a RE executable project
- Modify the source code to call a function hal_entry_lib() declared in the static library project
- Modify the build settings to add the static library
- Build the RE executable project.

Detailed steps are described below.

1. Create an executable project as described in chapter 3.2 and name the project (e.g. “RE_App”).
2. From project explorer window, open “RE_App.c” under RE_App\src\.
3. Add a declaration for library function “hal_entry_lib()”, and call this function in main() function.

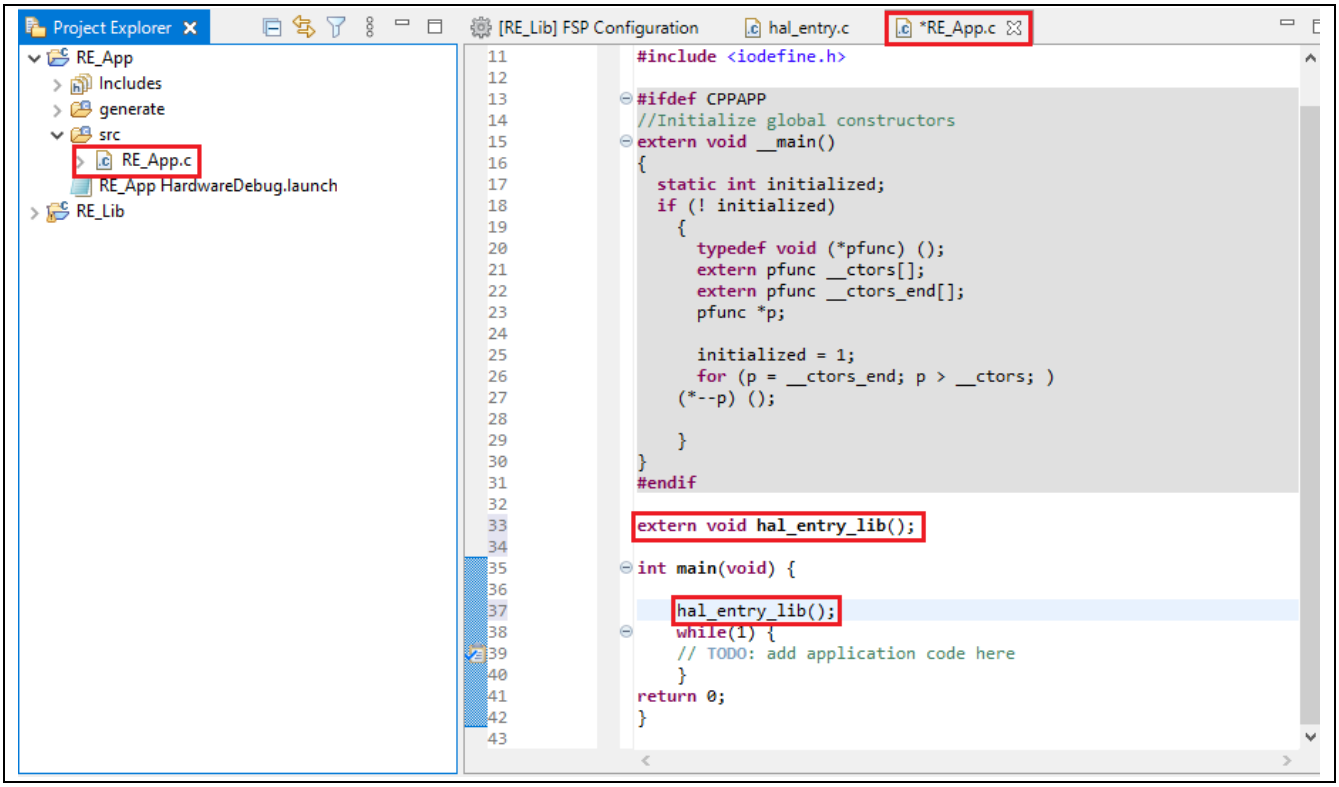


Figure 3-18 Update “RE_App.c”

4. Right-click on “RE_App” project and select [C/C++ Project Settings].

5. In the Project Properties dialog, select [C/C++ Build] → [Settings], then select [Tool Settings] tab. Select [GNU ARM Cross C Linker] → [Libraries] and click [Add…] at “User defined archive (library) files (-l)” to add “RE_Lib” as the library file.

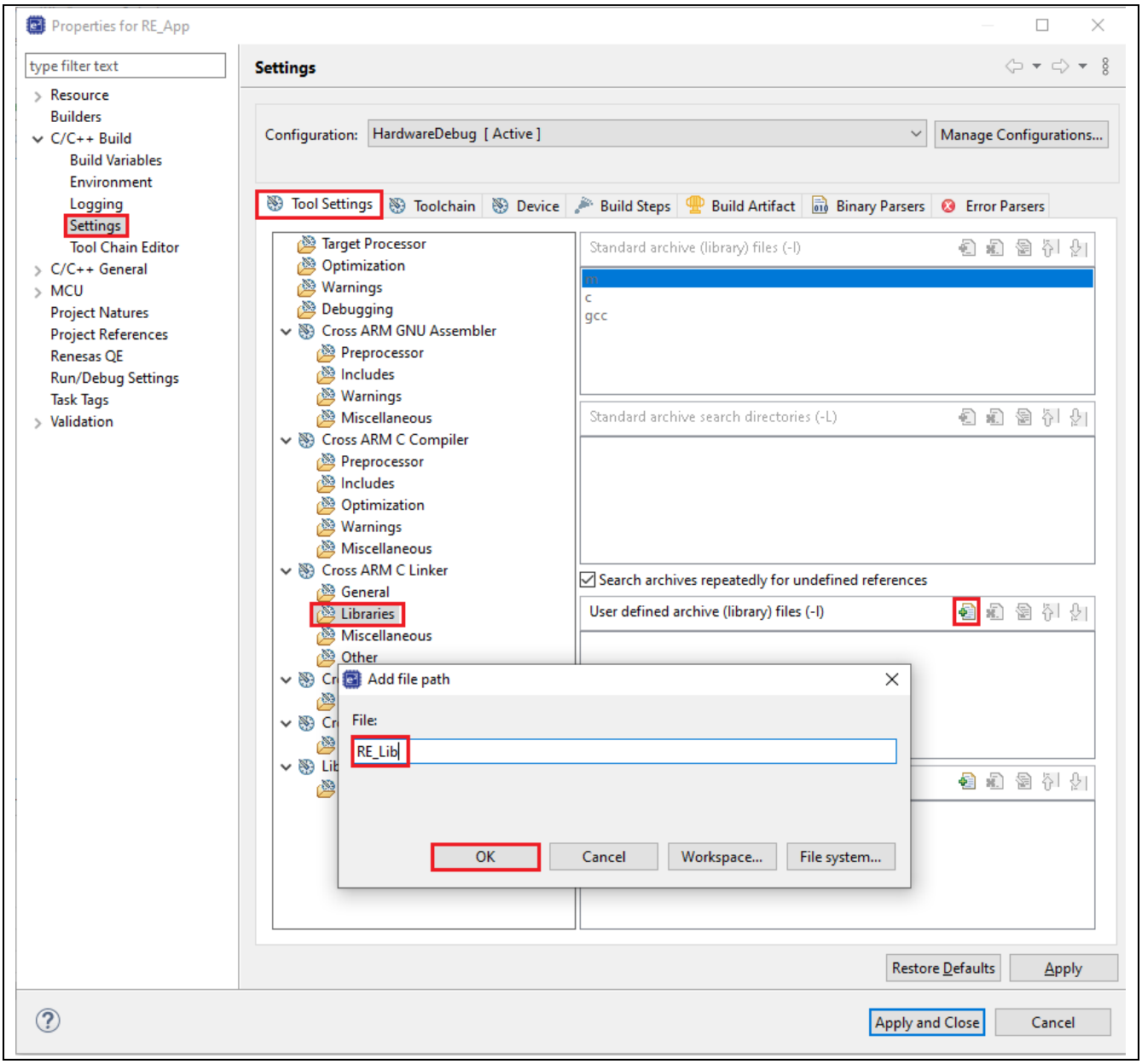


Figure 3-19 Add reference to library file

6. Click [Add…] at “User defined archive search directories (-L)” to add "${workspace_loc:/RE_Lib/Debug}"

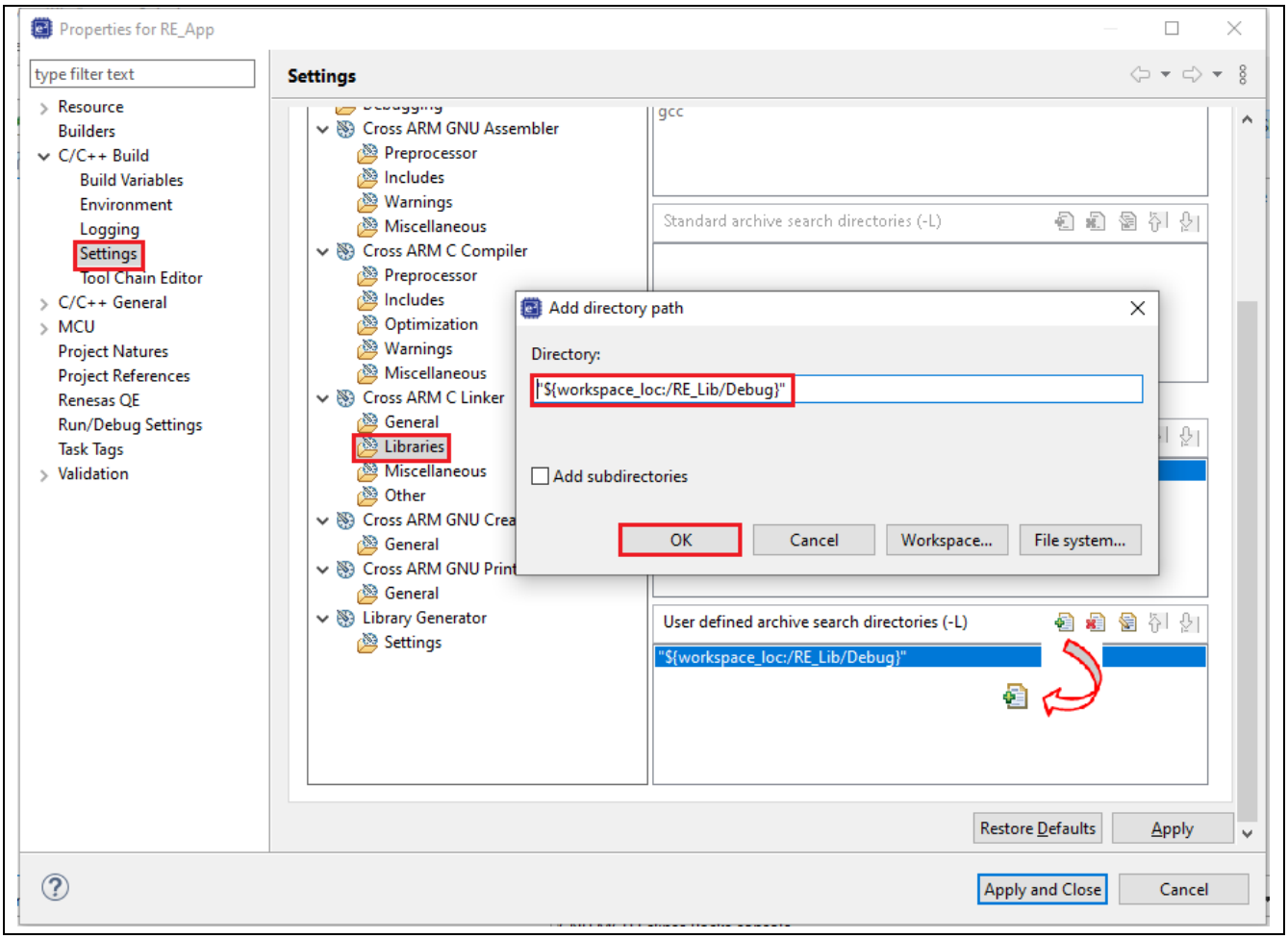


Figure 3-20 Add reference to library location

7. Verify that the new path has been updated. Go to [Cross ARM C Linker] → [Libraries] to check.

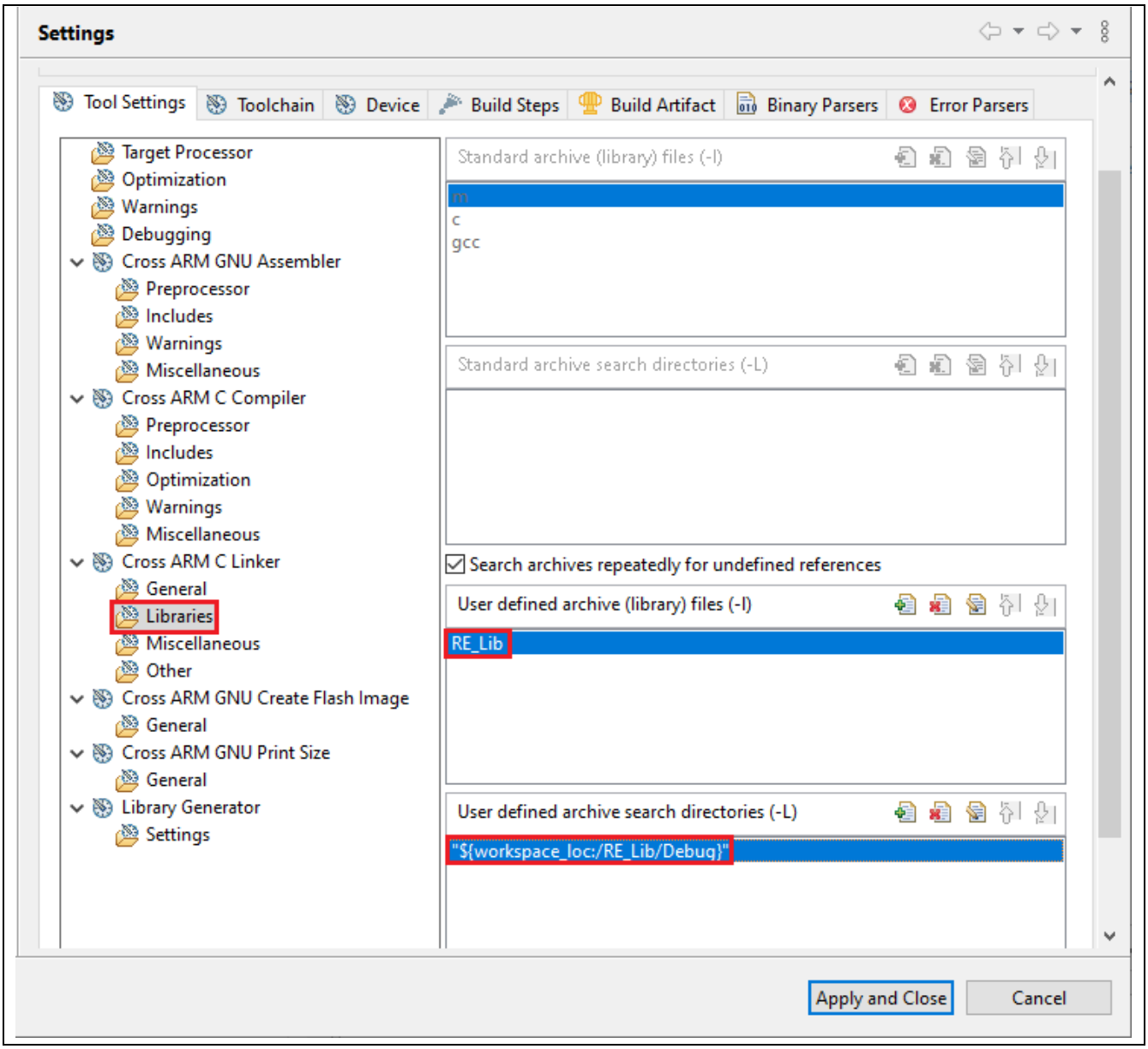


Figure 3-21 Cross ARM C Linker settings are already updated

8. In the Properties dialog, select [Project References] in the left pane, then tick on “RE_Lib” to mark the executable project as depending on the static library project. Click [Apply and Close].

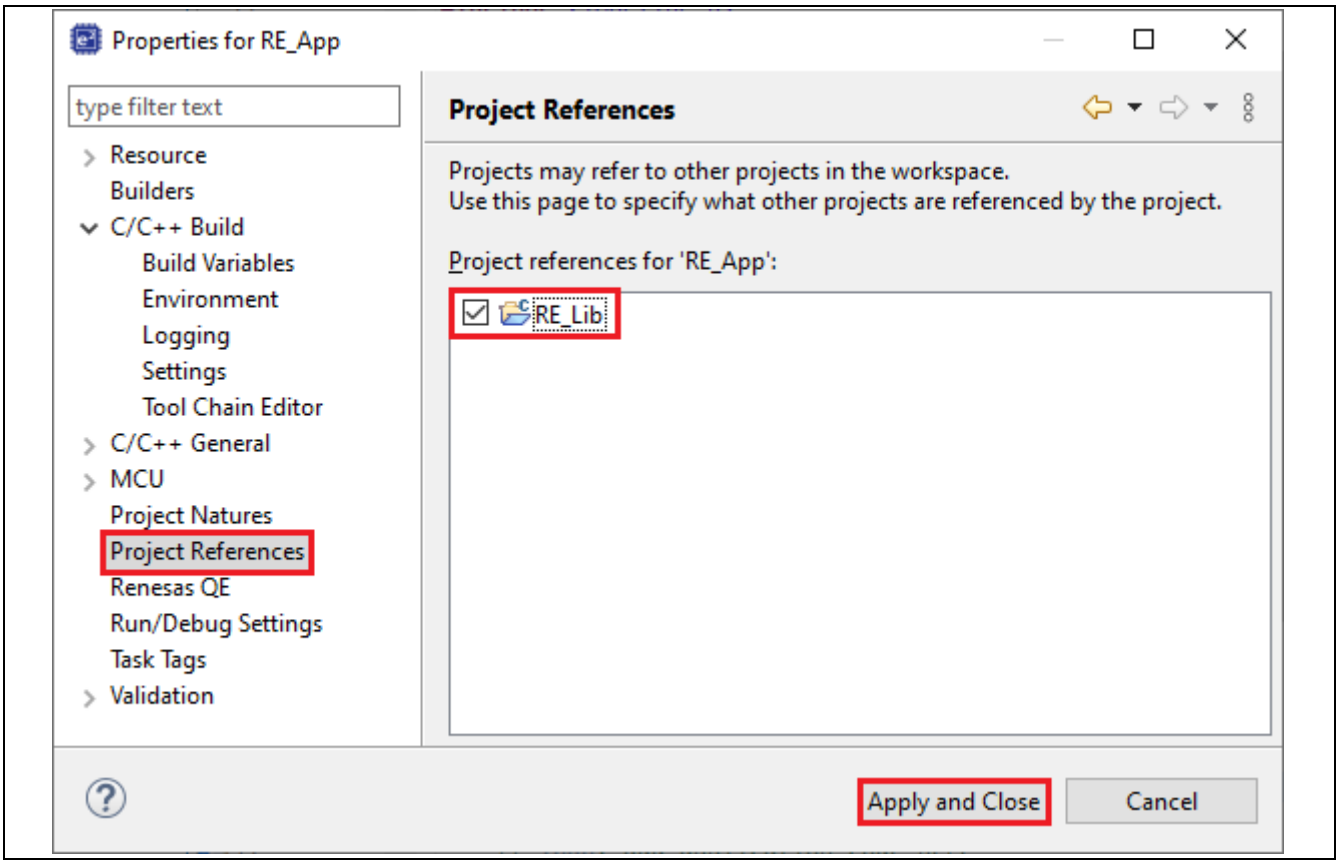


Figure 3-22 Mark executable project as depending on the static library project

9. Build the application project.

10. Set a breakpoint where the library function hal_entry_lib() is called. Run RE_App project.

11. When the program stops at the breakpoint, resume it. Confirm that the library function which blinks the LEDs (e.g. hal_entry_lib()) is executed.

Figure 3-23 Application project invokes library function

3.4 Import Existing Projects into Workspace

To import an existing e² studio project to the current workspace, follow the steps below. These steps import a sample project from Renesas website and will be used for demonstrating debugging features in section 5.4.

1. Download the LED Blinker sample code for RE01 from Renesas website: https://www.renesas.com/sg/en/document/scd/led-blinker-sample-code-re01-1500kb-and-256kb-group-application-note-sample-code.

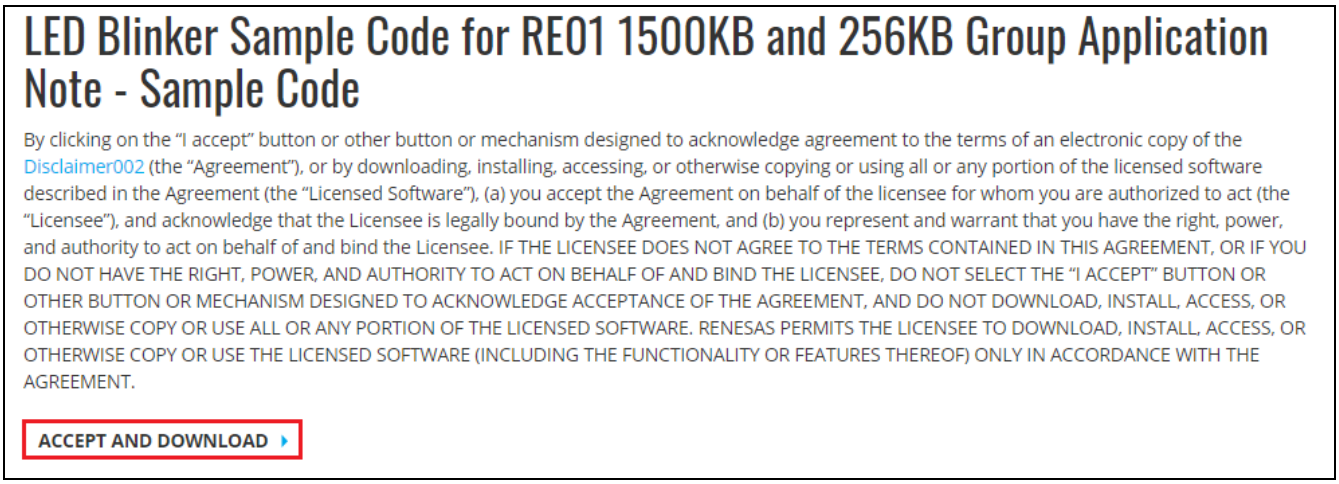


Figure 3-24 Download the Sample Code

2. After downloading the package, extract the zip file.

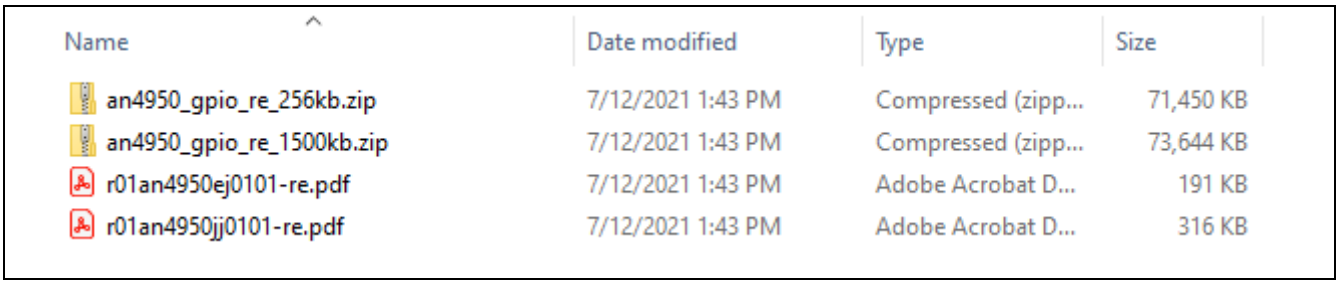


Figure 3-25 The Sample code package

3. In e² studio, select [File] → [Import]

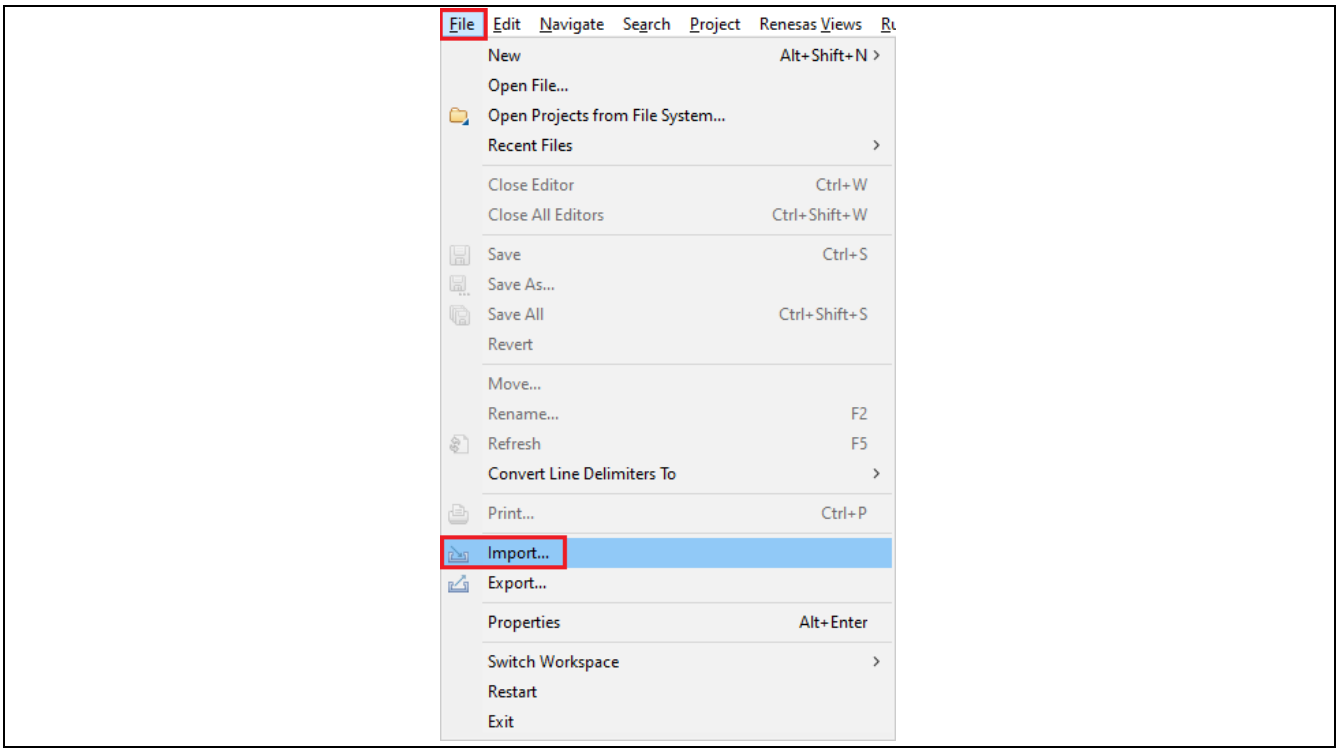


Figure 3-26 Import the sample project

4. In the [Import] dialog, select [General] → [Existing Projects into Workspace]. Click [Next]

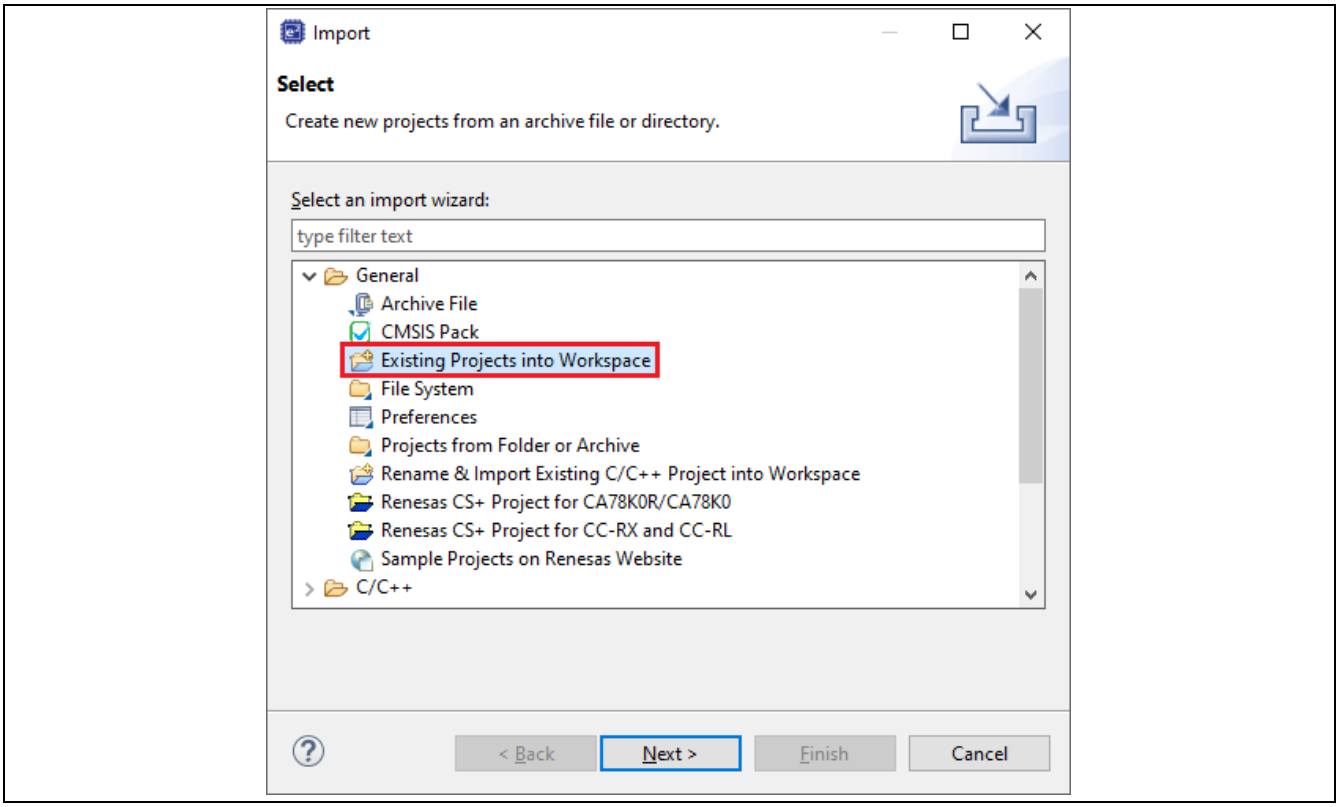


Figure 3-27 Select import wizard

5. In the [Import Projects] dialog, select “Select archive file”. Click [Browse] then select the zip file named “an4950_gpio_re_1500kb.zip” in the sample code package. Select the project listed in “Projects:” and click [Finish].

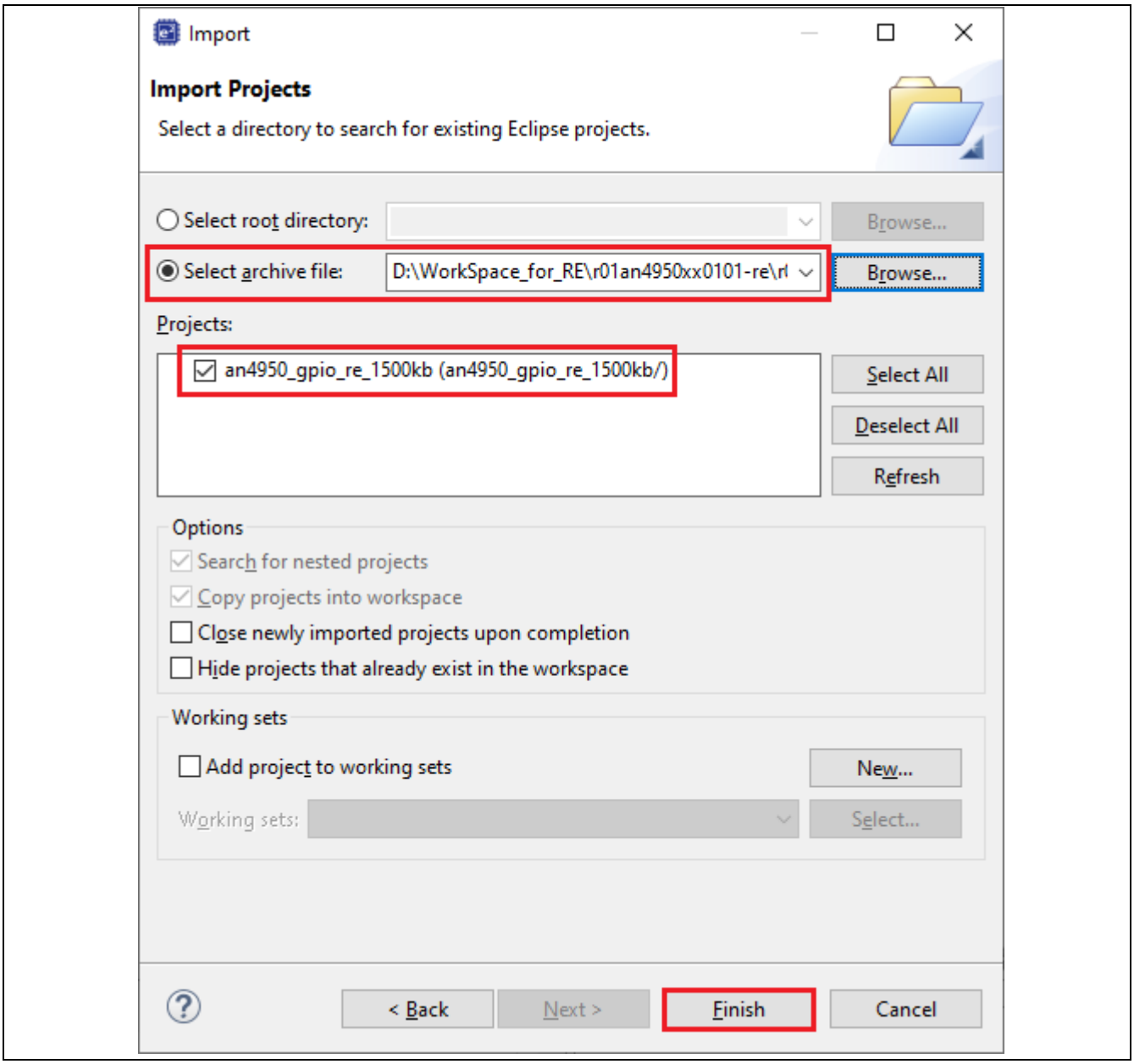


Figure 3-28 Select project to import

6. Open the project properties, select [C/C++ Build] → [Settings] in the left pane. Select tab [Toolchain] and select the correct toolchain for the project. Click [Apply and Close].

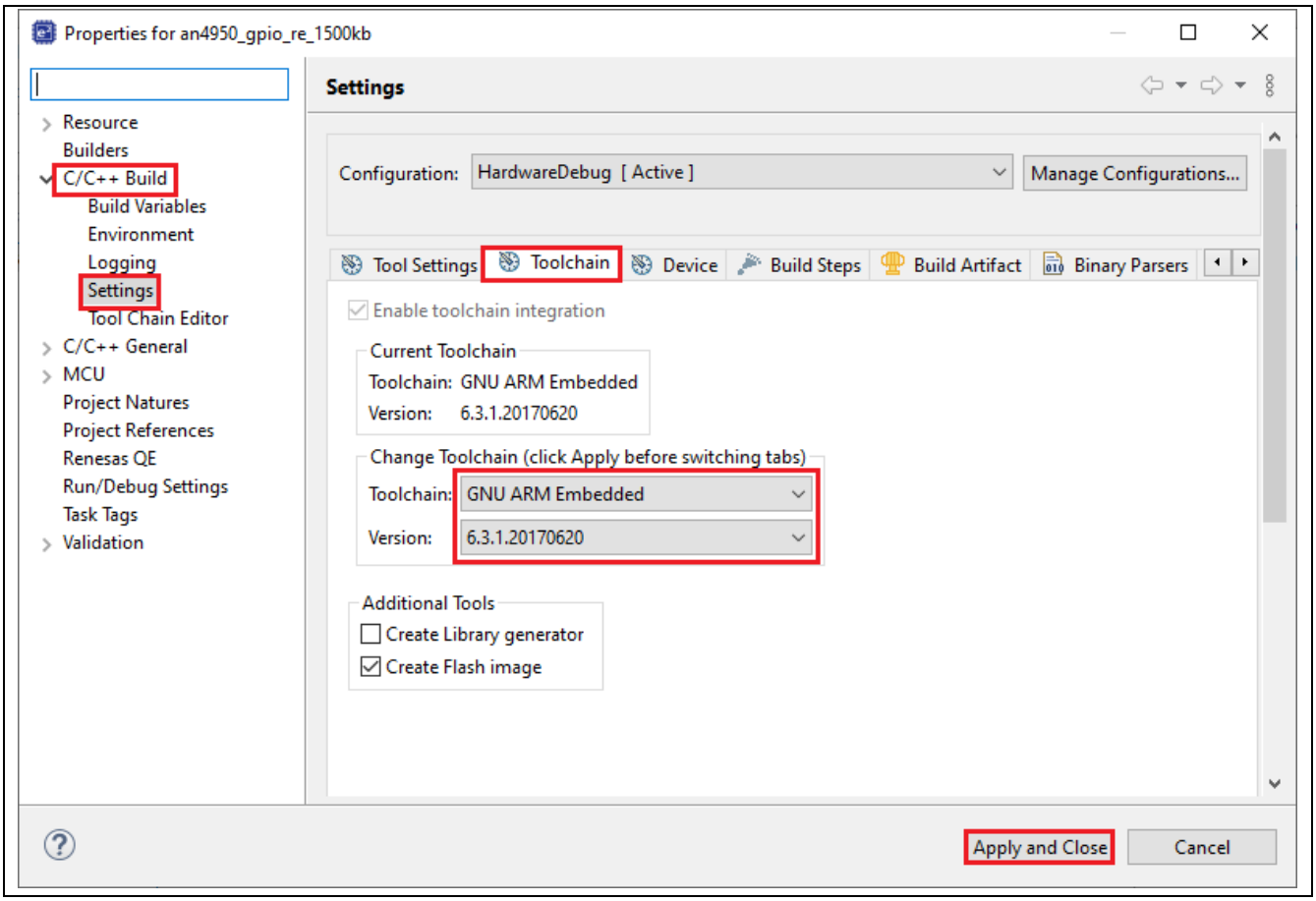


Figure 3-29 Update project toolchain (without Smart Configurator)

If the imported project uses Smart Configurator, the setting GUI may be different.

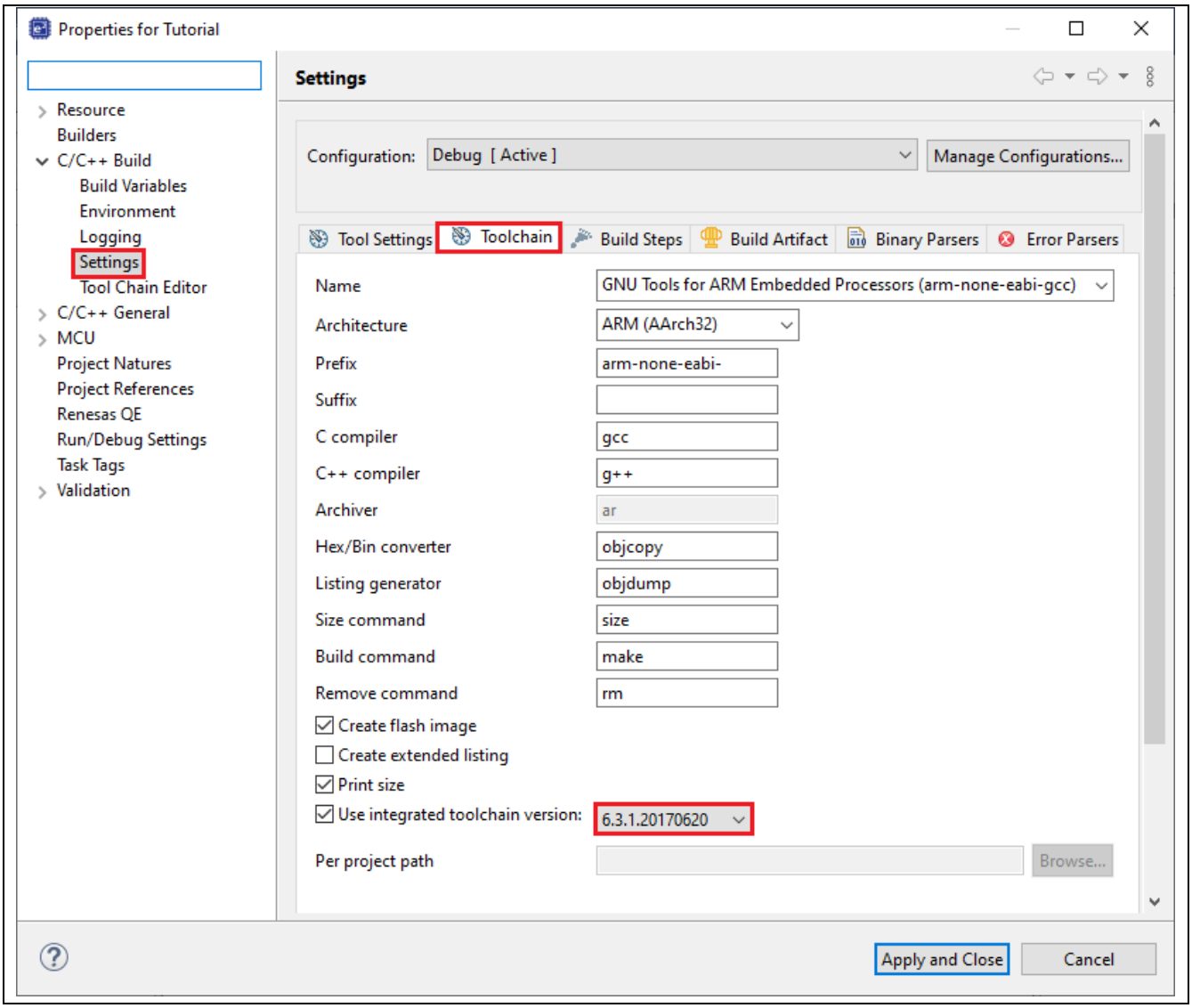


Figure 3-30 Update project toolchain (with Smart Configurator)

7. Build the project and make sure that it is successful.

3.5 Configuration Editor

The Configuration editor view displays the current project configuration settings. The settings are saved in the file 'configuration.xml'. The project configuration settings are grouped into multiple pages that allow you to set several configurable aspects of the project, such as how pins and clocks are set up, and which drivers are included.

To edit the project configuration, make sure that:

- RE Configuration perspective is open by clicking [Window] → [Perspective] → [Open Perspective] → [Other…] → [FSP Configuration] and,
- The 'configuration.xml' file is opened.

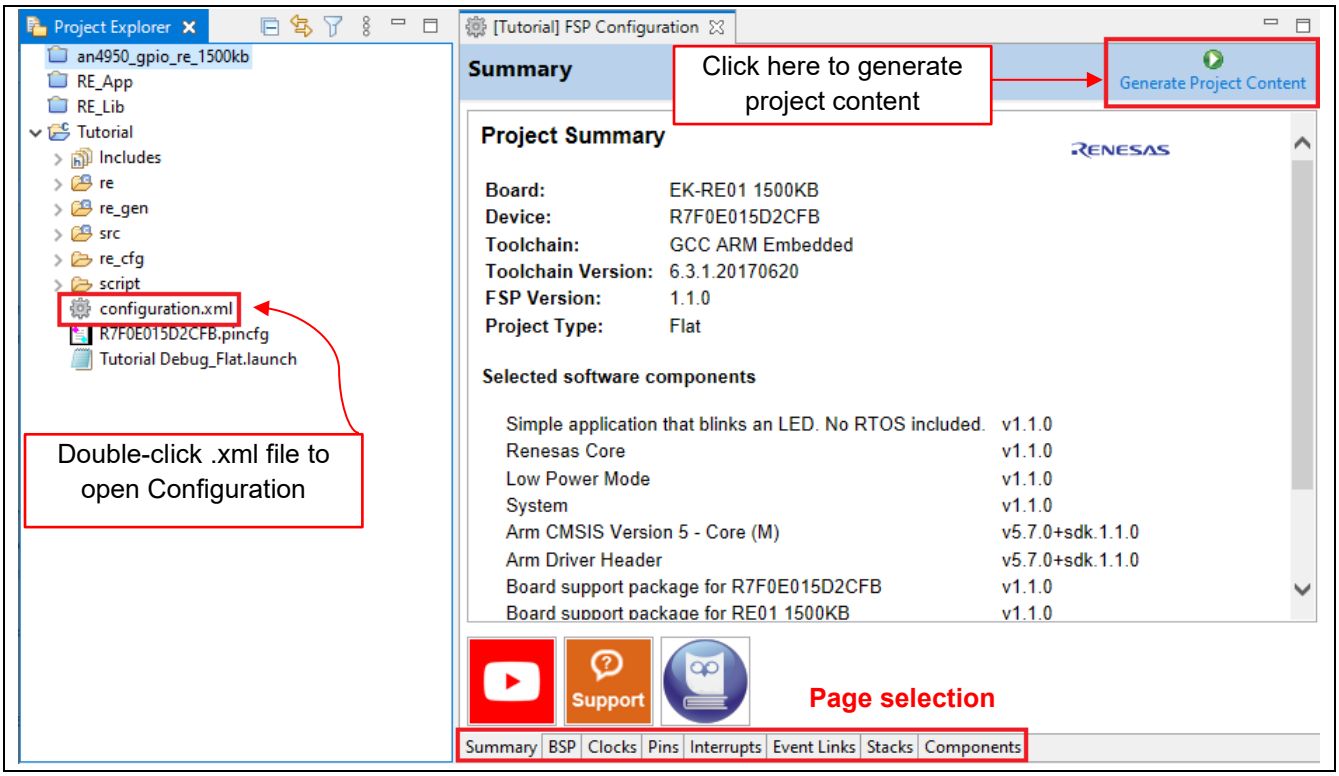


Figure 3-31 Configuration Editor view

There are 8 pages (or tabs) in the Configuration editor.

The Summary page contains project-specific summary information.

The BSP page allows users to select the FSP version, the type of RE board, and the device.

The configuration steps and options for the Clocks, Pins, Interrupts, Event Links, Stacks and Components pages are discussed in the following chapters.

3.5.1 Summary Page

The summary page contains a project-specific summary which includes details of the currently selected device, board and RE software components, etc. There are also useful links to the 'Renesas Presents' YouTube channel and the Software Package's user manual.

If user adds new threads and modules/objects to a thread, this information will be also shown on the Summary page.

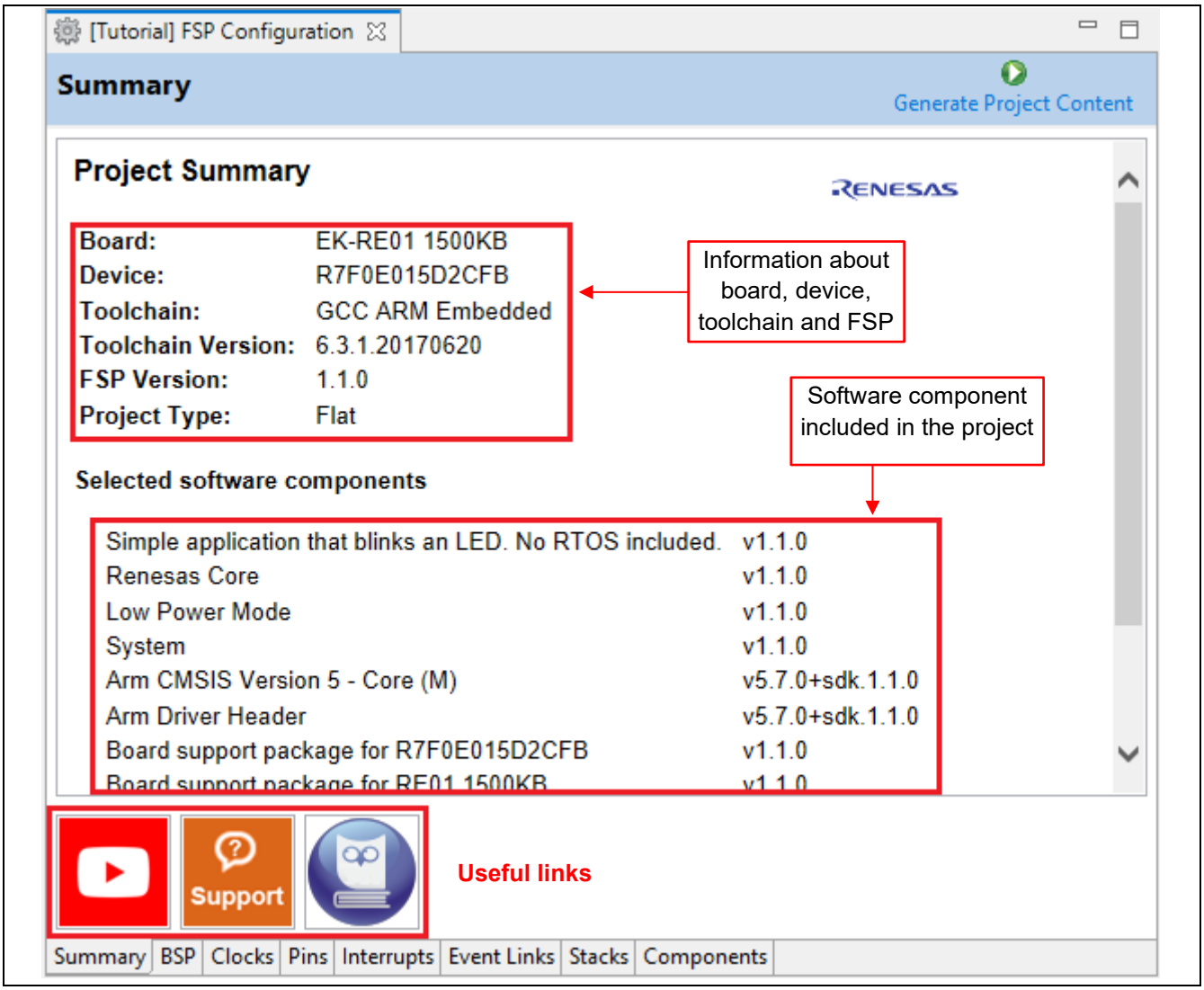


Figure 3-32 Summary page

3.5.2 BSP Page

The BSP Page allows users to select the FSP version, board and device. Users can also import the CMSIS pack from this page.

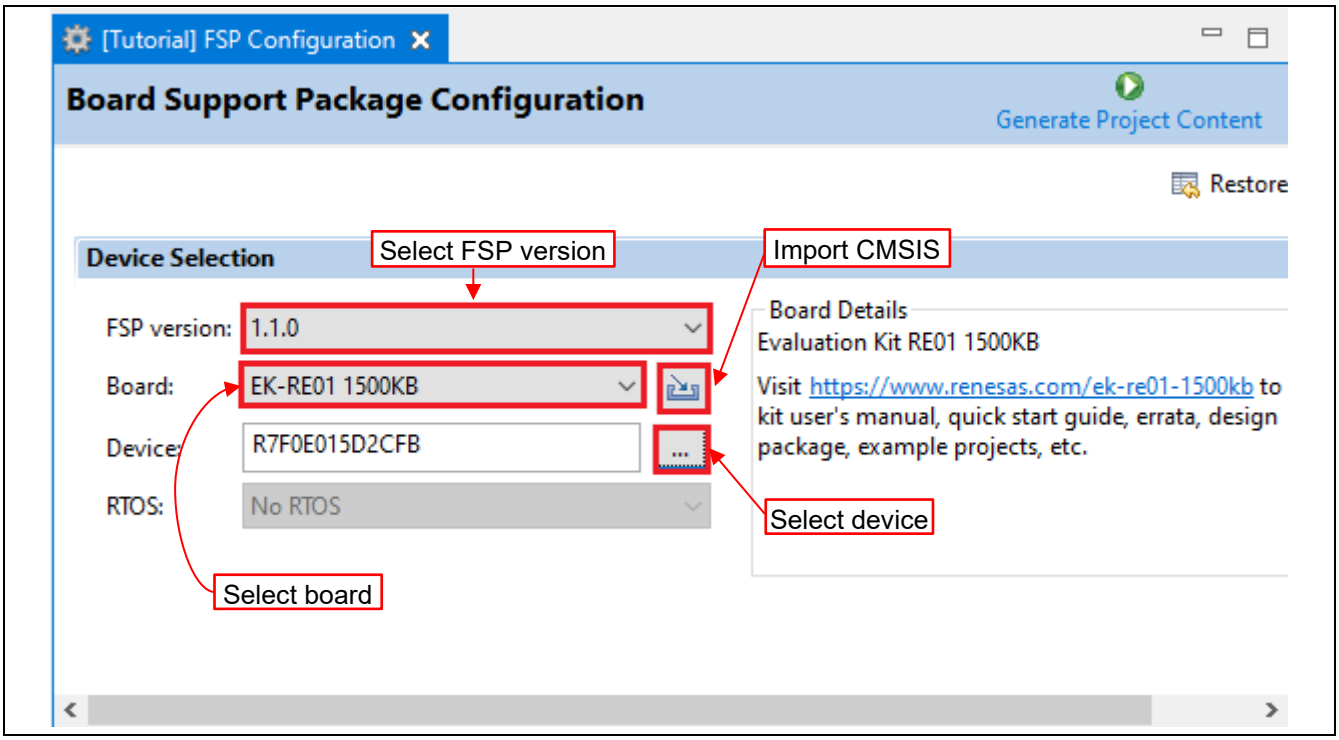


Figure 3-33 BSP page

3.5.3 Clocks Page

The Clocks page sets up the initial clocking for the application. Clock sources, PLL settings, and clock divider settings can be selected for each of the output clocks.

For details on the Clock Generation Circuit (CGC), see the RE hardware user's manual. To update the project, follow these steps:

1. Select a value in the drop-down list for the clock setting on GUI.

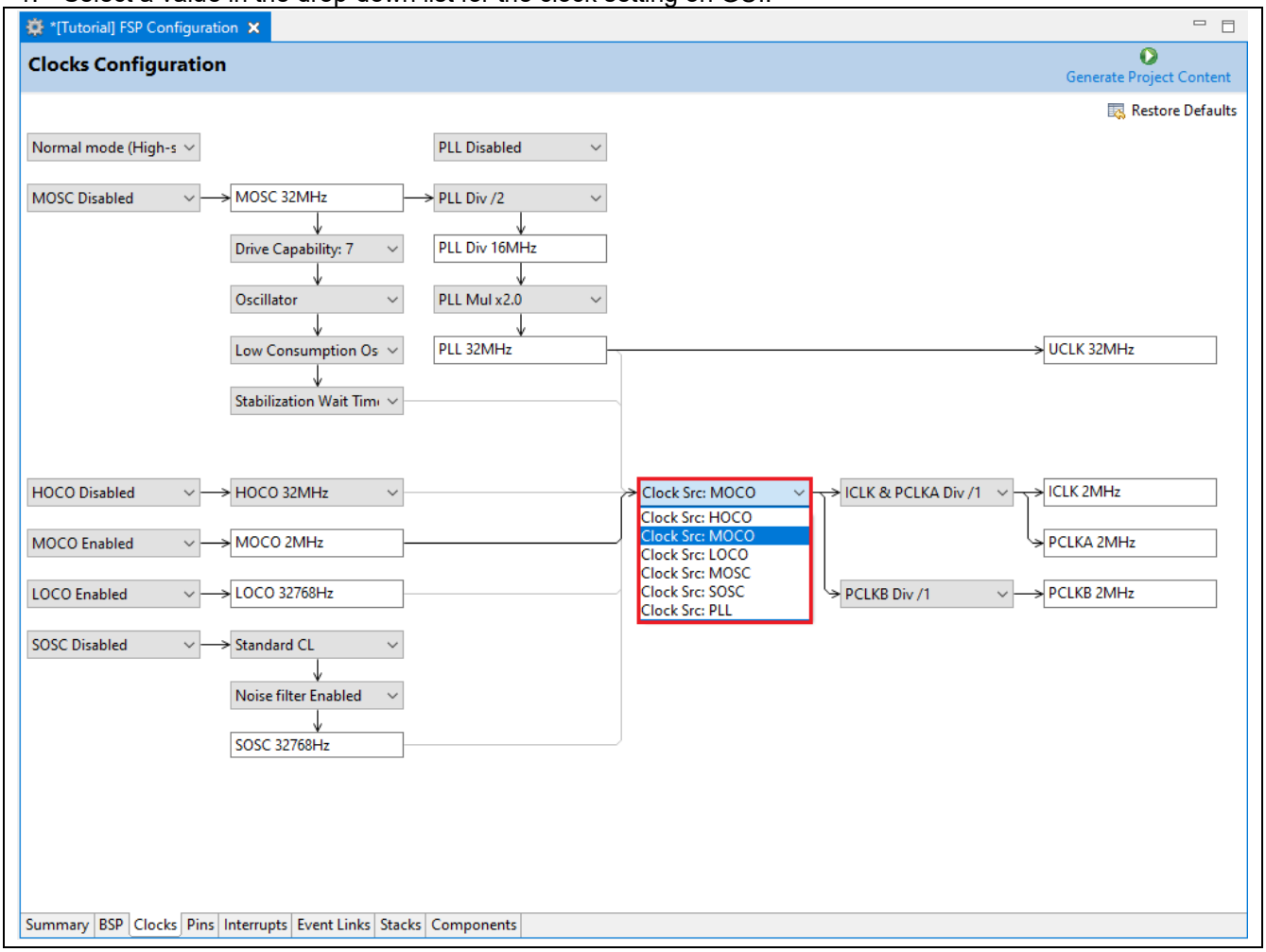
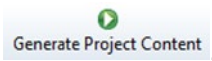


Figure 3-34 Clock configuration page

2. Save the Project Configuration Settings, for example by hitting Ctrl-S.

3. Click the Generate Project Content button 

.

4. The file bsp_clock_cfg.h is updated with the selected clock configuration.

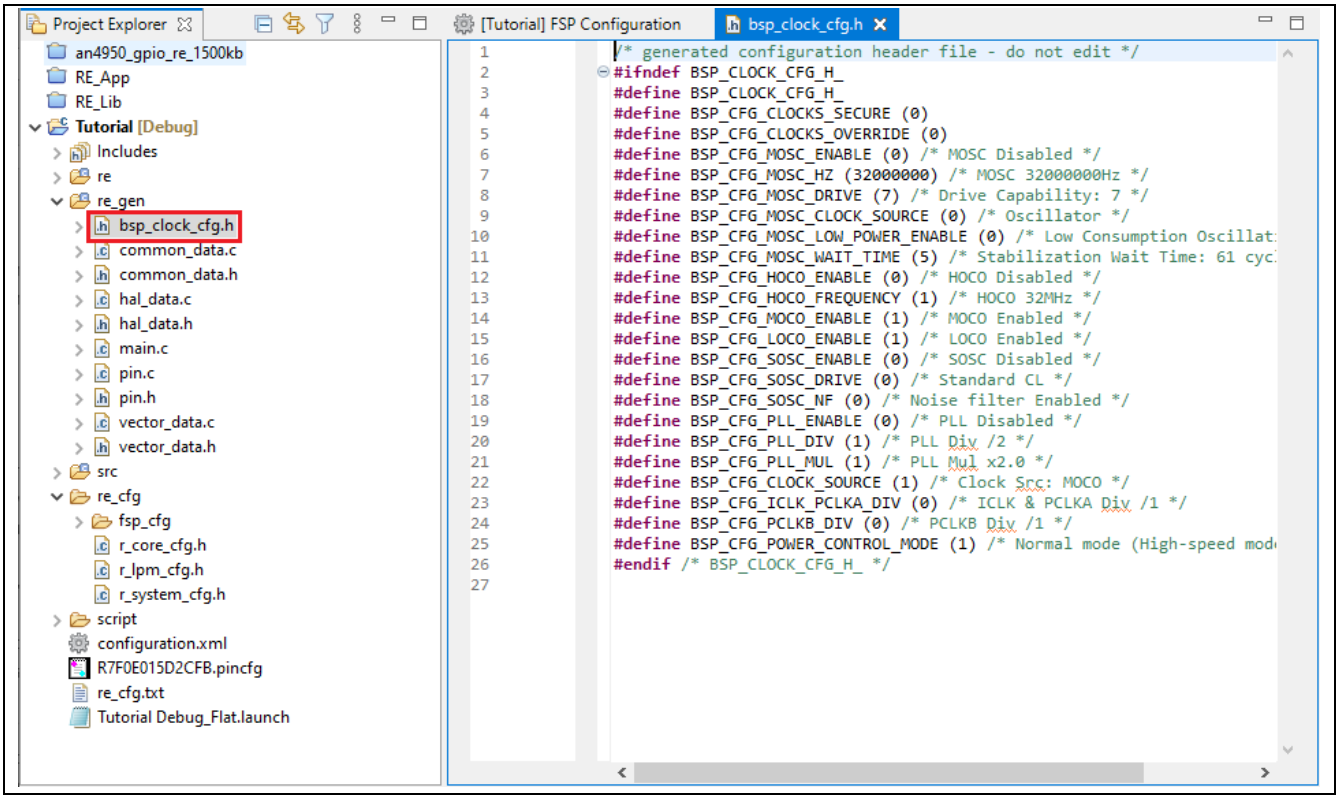


Figure 3-35 bsp_clock_cfg.h is updated

3.5.4 Pins Page

The Pins page provides a graphical user interface for generating the pin configuration settings for the project.

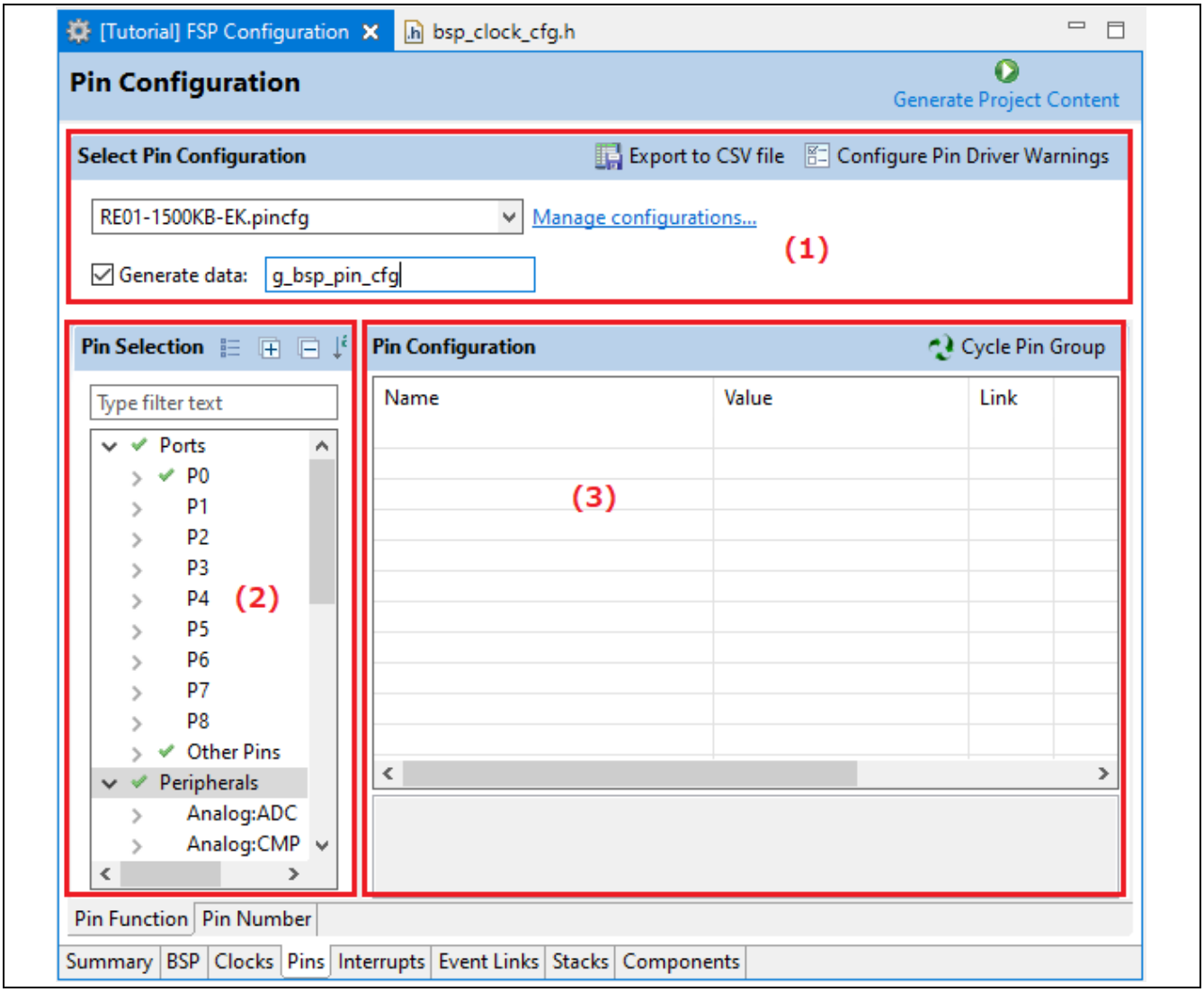


Figure 3-36 Pin configuration GUI

he Pin Configuration window consists of 3 parts:

1. Select Pin Configuration: Selects pin-configuration file and specifies the name for the associated data structure. Multiple pin configurations can be set as follows:
   - Create a new .pincfg file (e.g. NewName.pincfg) in Project Explorer by copying an existing one.
   - Select the new .pincfg file (e.g. NewName.pincfg) in the “Select Pin Configuration” dialog box.
   - Check the "Generate data" checkbox and give the new pin configuration a unique data structure name in the text field.
   - The multiple pin configurations will be created in different data structures.
2. Pin Selection: Selects pin or peripheral that will be set up.
3. Pin Configuration: Set up for function/property of the selected pin/peripheral.

Follow below steps to configure pins of the peripherals to be used in the project:

1. Select a peripheral in the “Pin Selection” pane, e.g. [Connectivity: SCI] → [SCI4]. The configuration for this peripheral will be shown in the “Pin Configuration” pane.

2. Select an Operation Mode for the peripheral, e.g. “Simple SPI”.

3. Select the pins you would like to use for the Input/Output functions of the selected peripheral in the selected mode.

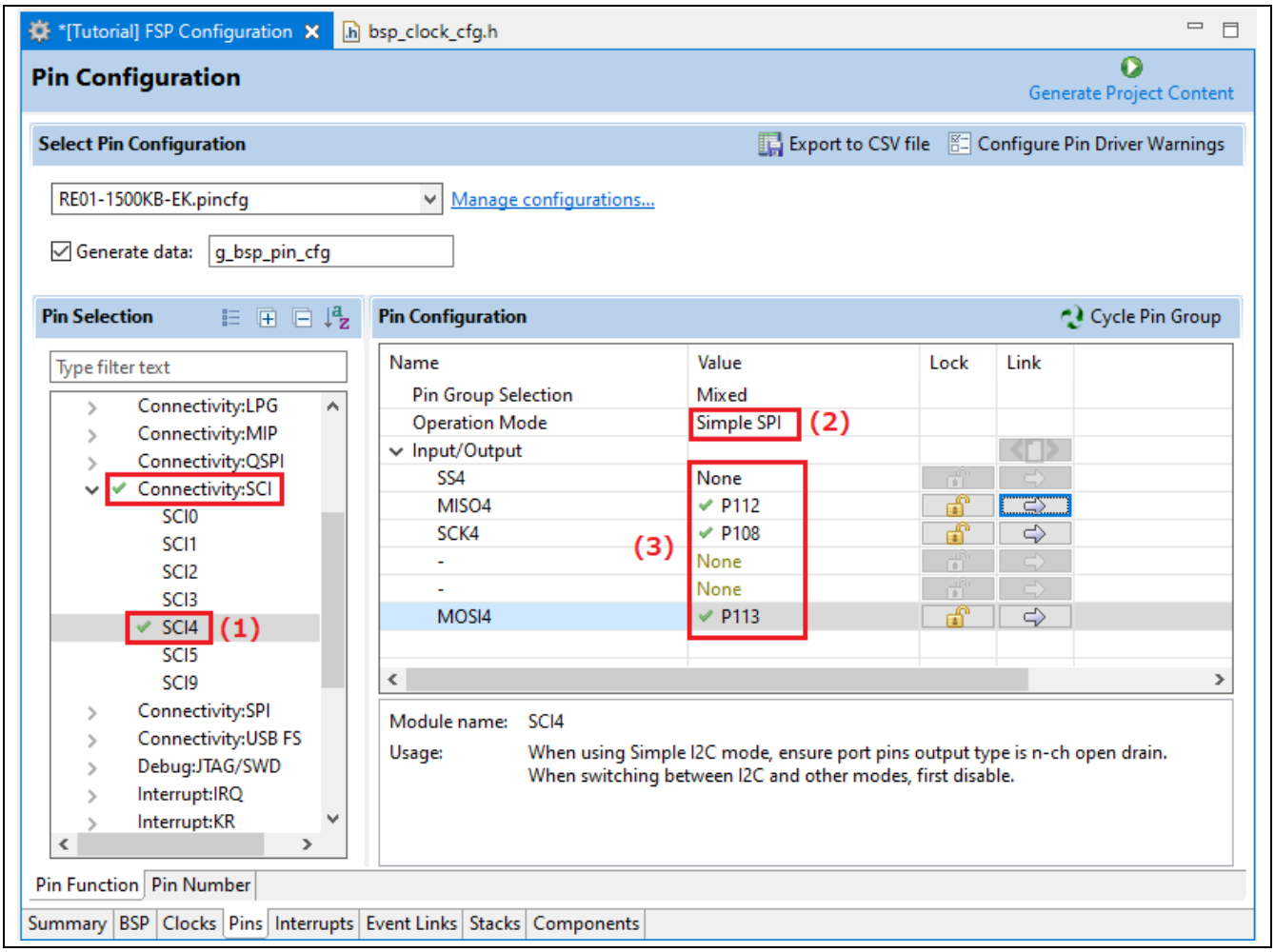


Figure 3-37 Pin Configuration Setting (by Peripheral)

A single pin can also be set up following the steps below:

1. Select a pin in the “Pin Selection” pane, e.g. [Ports] → [P0] → [P003]. The configuration for this pin will be shown in the “Pin Configuration” pane.
2. Enter properties for this pin as an example in the figure below.

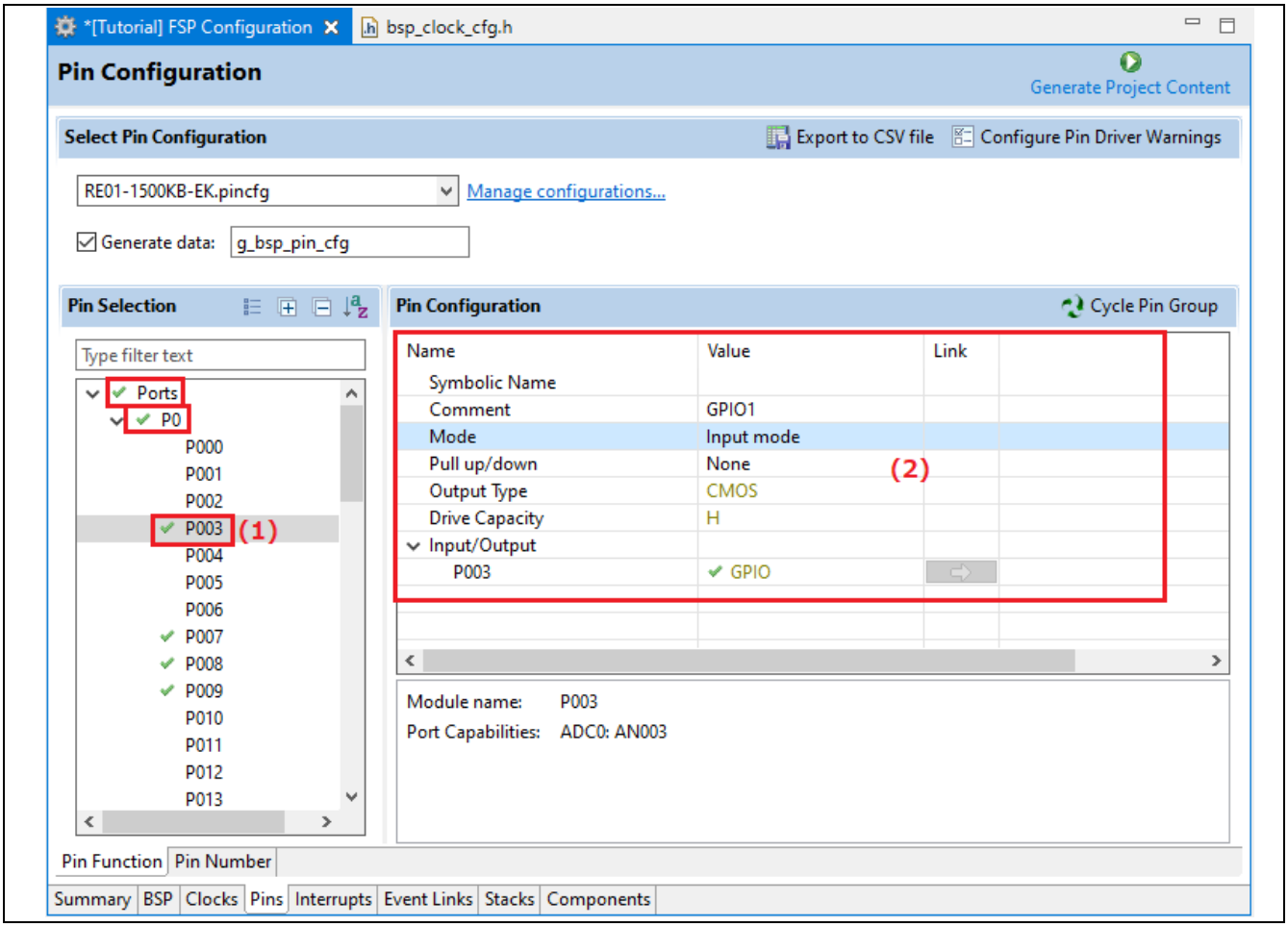


Figure 3-38 Pin Configuration Setting (by single pin)

3. The MCU Package view shows this pin change.

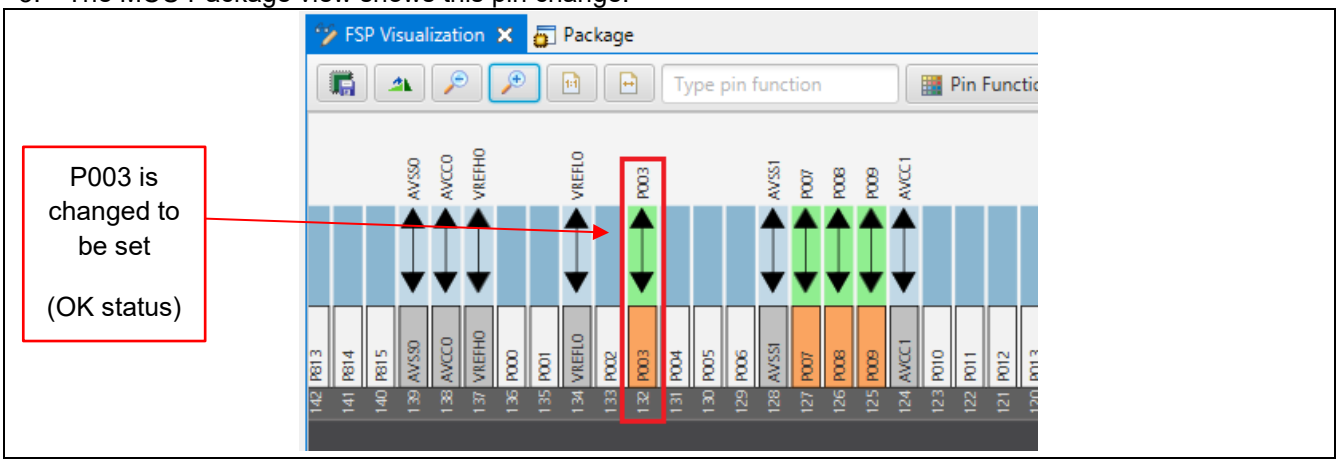


Figure 3-39 MCU Package View (Connection Status)

3.5.5 Stacks Page

The Stack page allows users to:

- Configure threads within a RE project.
- Add RE modules and objects to a thread.
- Modify module and object properties in the Properties View.

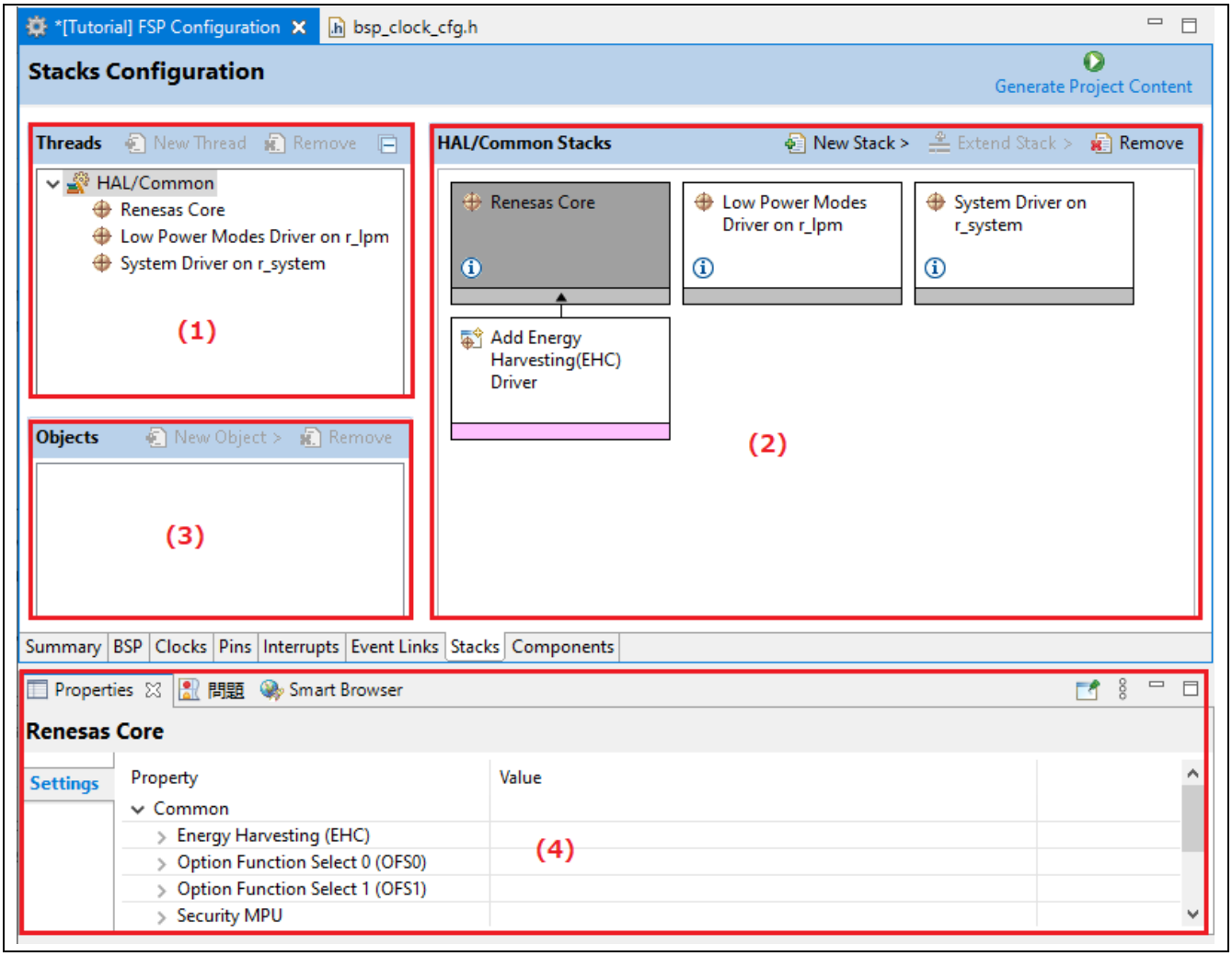


Figure 3-40 Stacks Configuration GUI

The Stacks page consists of 3 panes:

1. Threads pane: Add/remove threads.
2. Stacks pane: Add/remove Software module instances, i.e. IO port, SCI, UART, etc.
3. Objects pane: Add/remove kernel objects.

In addition, the Properties view supports the Threads Configuration and is used to modify module/object properties.

A module can be added to the existing project following the steps below:

1. Select a thread, i.e. HAL/Common. The modules and objects in this thread are shown.
2. In the Stacks pane, click “New Stack” to add a module to the thread, i.e. “New Stack” → [Driver] → [Connectivity] → [CMSIS Driver for USART on r_usart ch(SCI)].
3. Click the Generate Project Content button to generate the source code content.
4. Users can change the properties of the selected module in the Properties view according to their requirements.

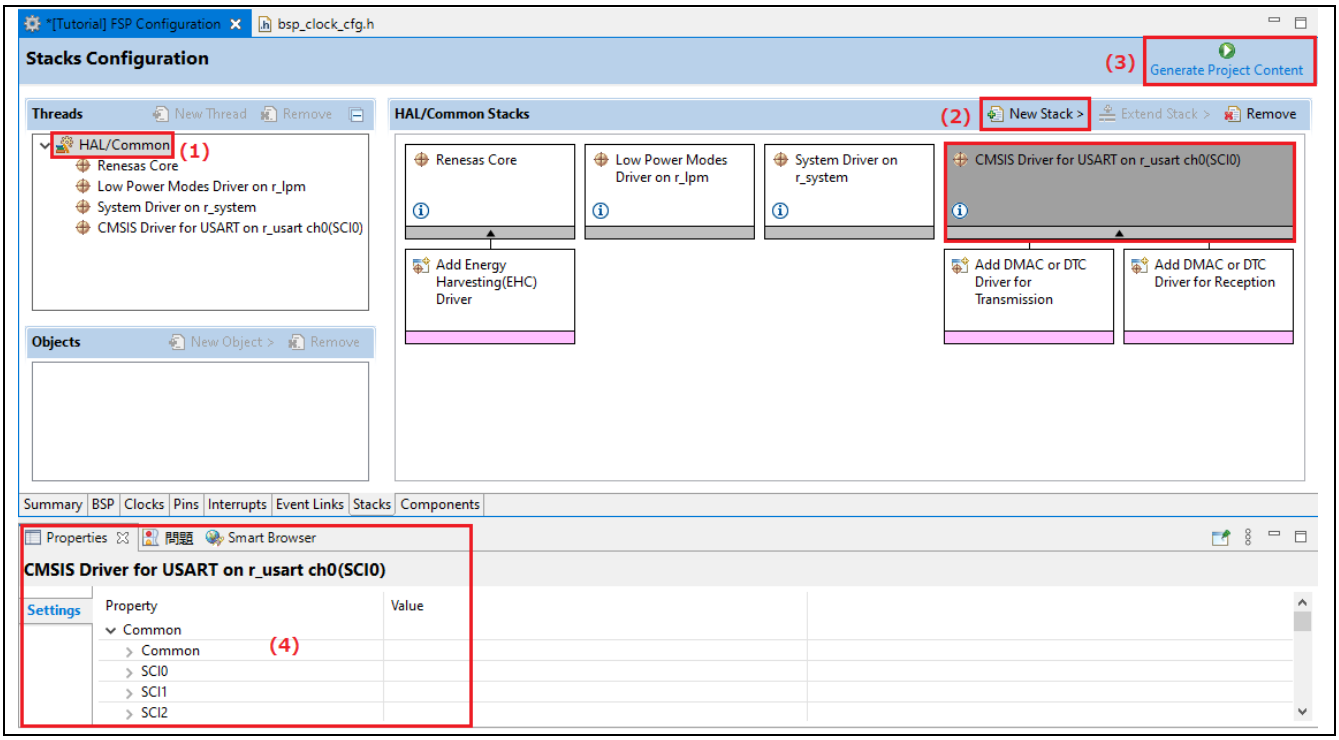


Figure 3-41 Add New Module to Thread

An added module may require dependent modules or configuration settings. Necessary dependent modules will be added automatically. Optional dependent modules are suggested to be added manually by the user. In this case, users should click on the suggested modules to add and configure their properties.

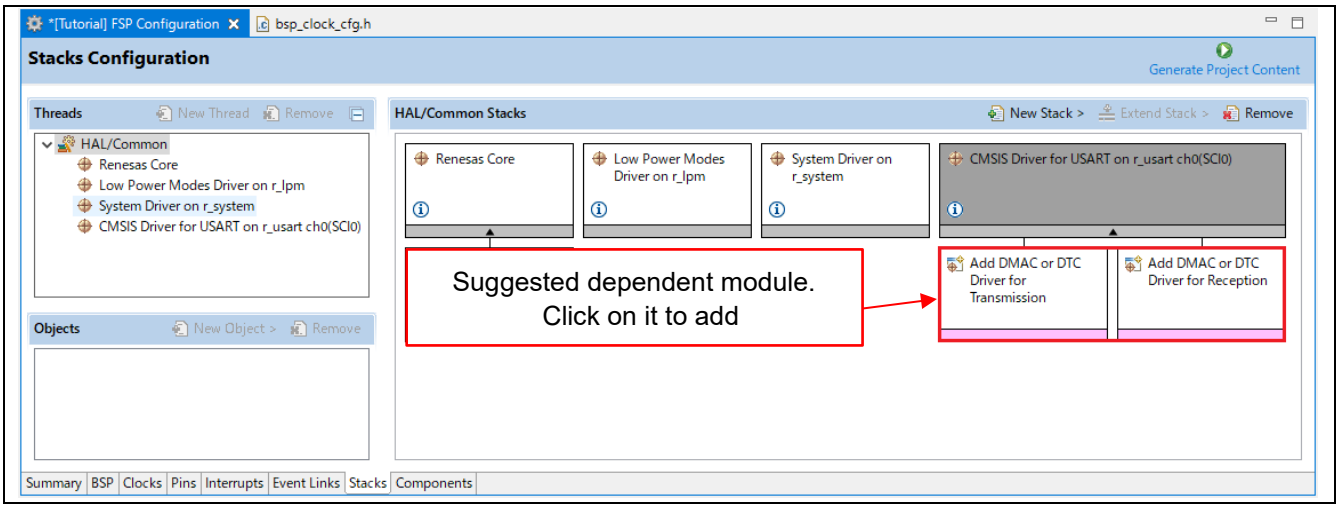


Figure 3-42 Dependent Modules

A module or a module stack can also be added by performing a “copy and paste” operation on the Threads page. Right-click on a module and select “Copy” to copy it. Right-click in the stack pane of the same or a different thread in the same project and select “Paste”.

A “cut and paste” operation is also available.

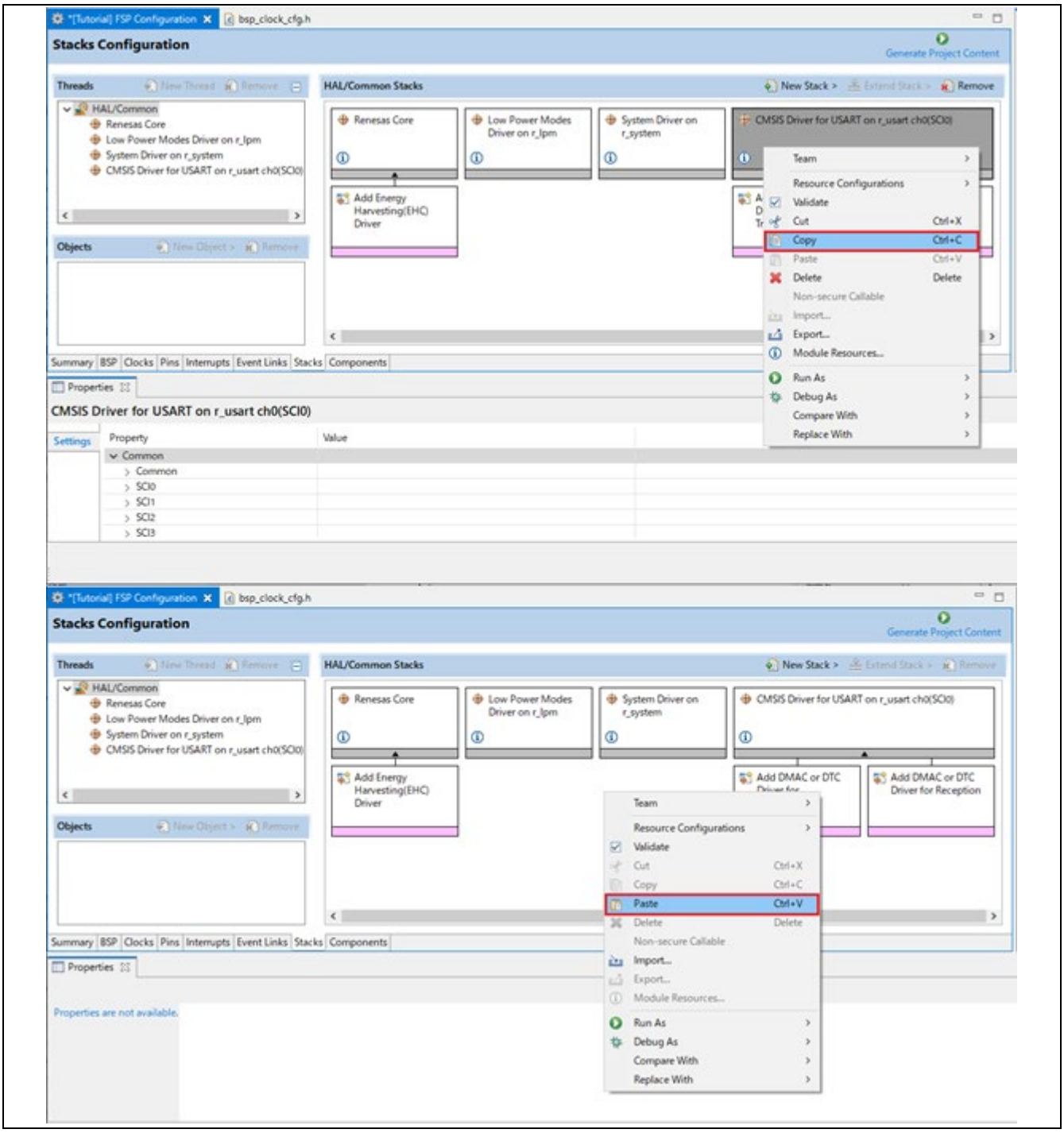


Figure 3-43 Copy & Paste operation

There will be a name conflict between the old module instance and the new one. Renaming one of the module instances will solve the problem. If there is hardware conflict, one of the modules should change to a different hardware channel.

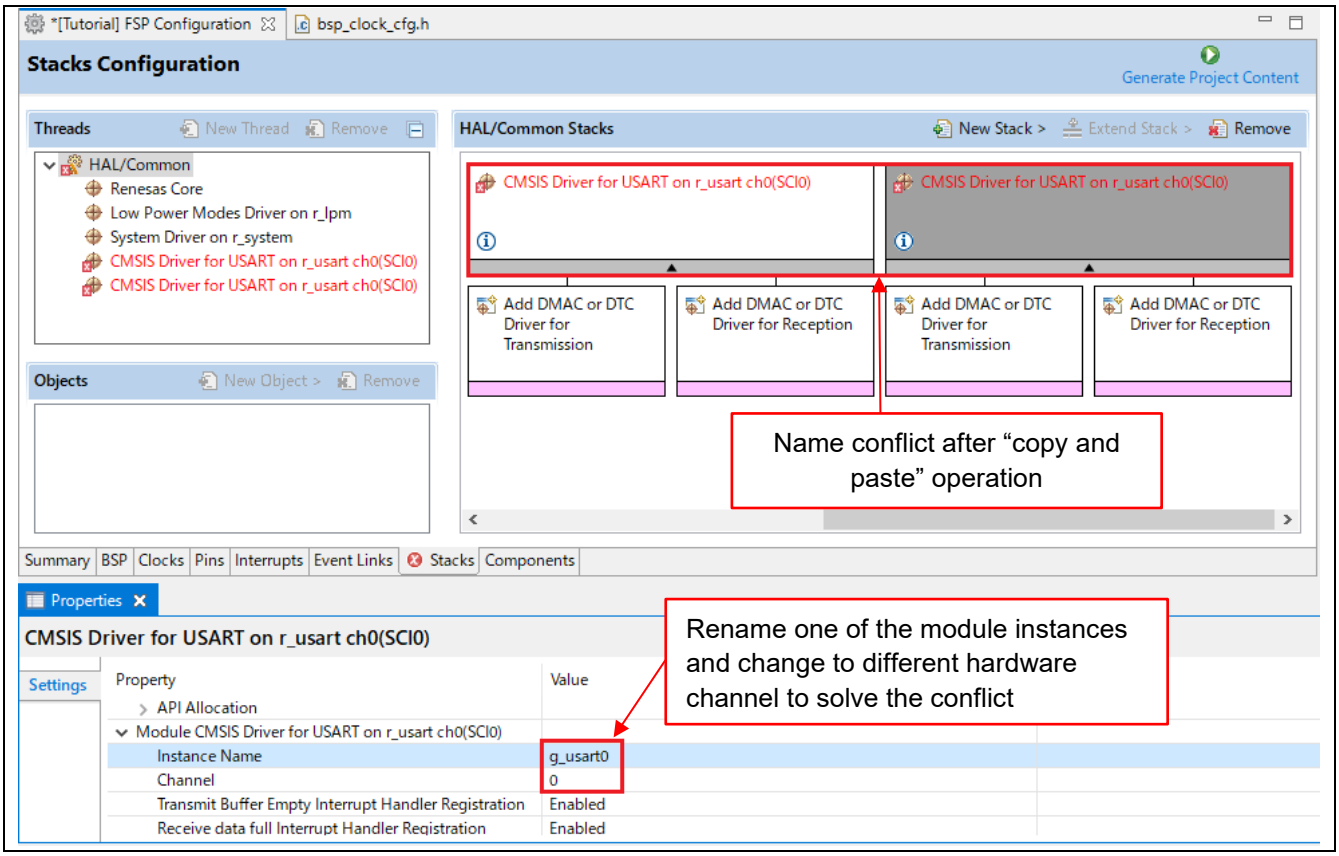


Figure 3-44 Module Instance name conflict

A module or a module stack can also be added by performing the export and import operation on the Stacks page. Right-click on a module and select “Export…” to export the configuration of the module to an XML file. Right-click in the stack pane of the same or a different thread in the same project and select “Import…” to import the configuration from the exported XML file.

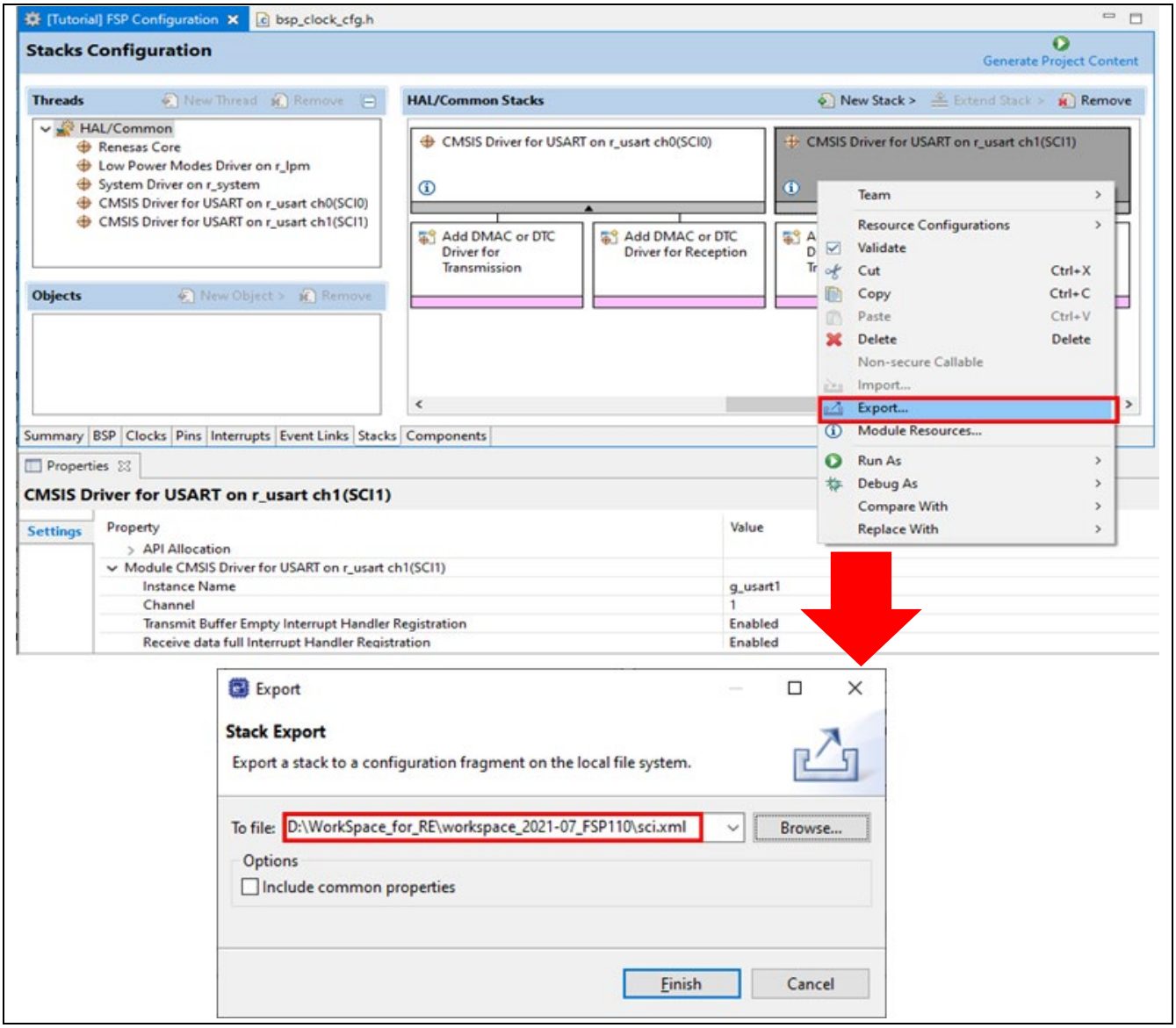


Figure 3-45 Export the RE stack

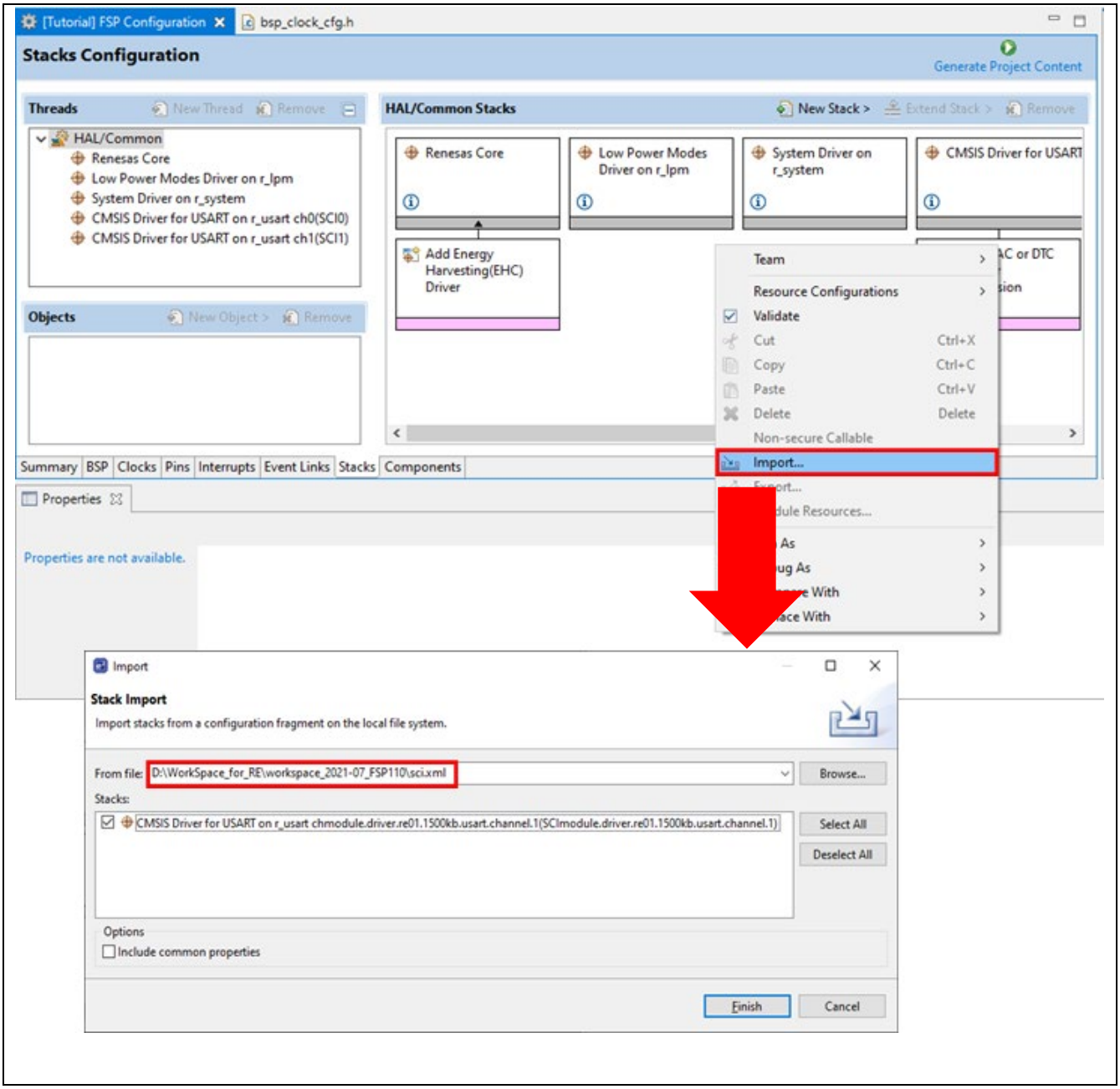


Figure 3-46 Import the RE stack

3.5.6 Interrupts Page

The Interrupt page allows the management of Event (interrupts) and ISR (Interrupt Service Routines) for use with the RE interrupt framework.

The interrupt Page consists of 2 panes:

1. The “User Events” pane shows a list of events that have been created manually by user.
2. The “Allocations” pane shows a list of events that have been provided by instantiated RE modules on the Stacks Configuration page.

   In each pane, the “Event” column contains event names. The “ISR” column contains subscriber for the corresponding event in the “Event” column.

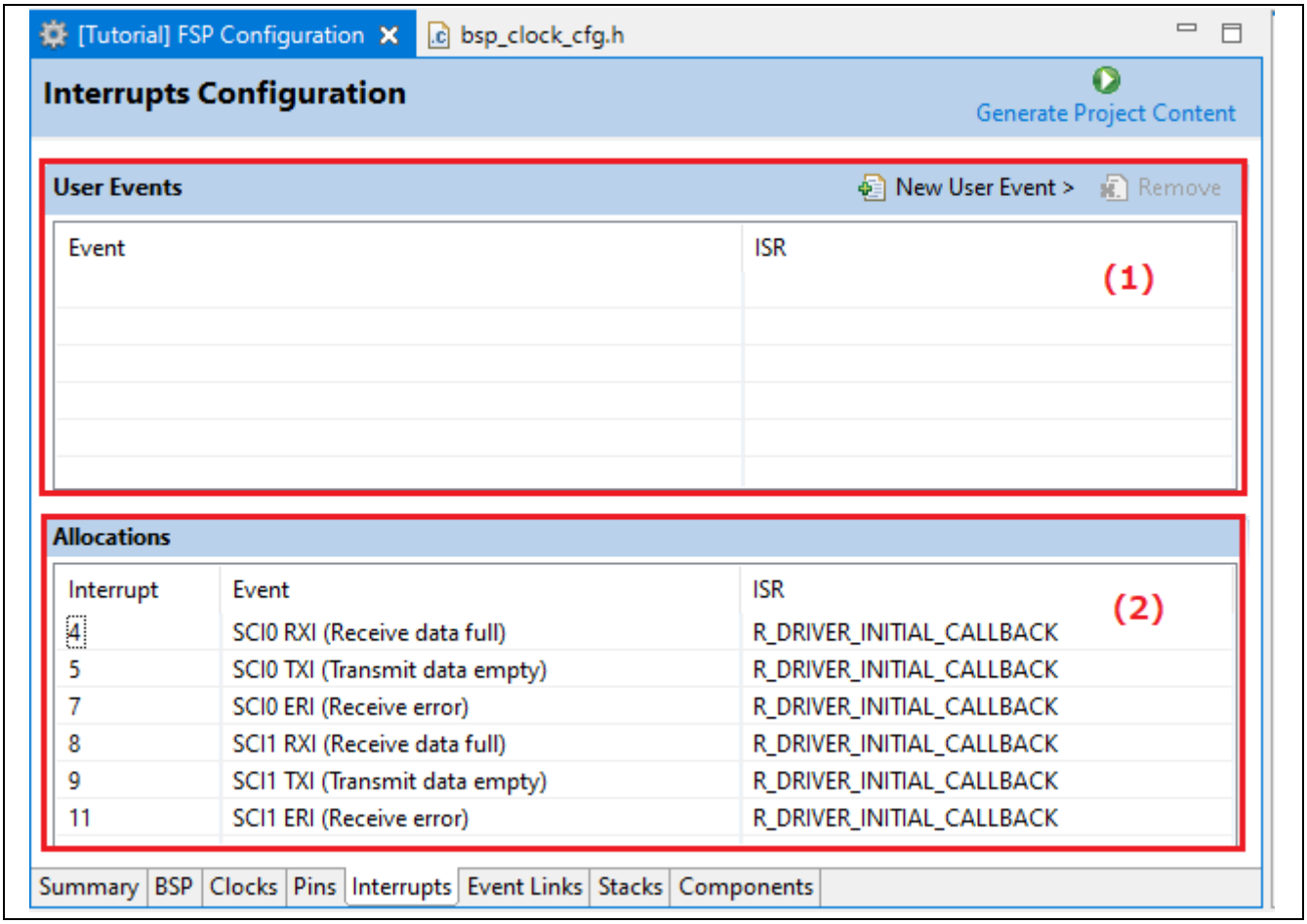


Figure 3-47 Interrupts Page

A user event and its ISR can be created manually by clicking on the button [New User Event], then select an event to create.

Note: ISR registration is required even when using interrupts as triggers for DTC or DMAC. Prepare a dummy function for ISR

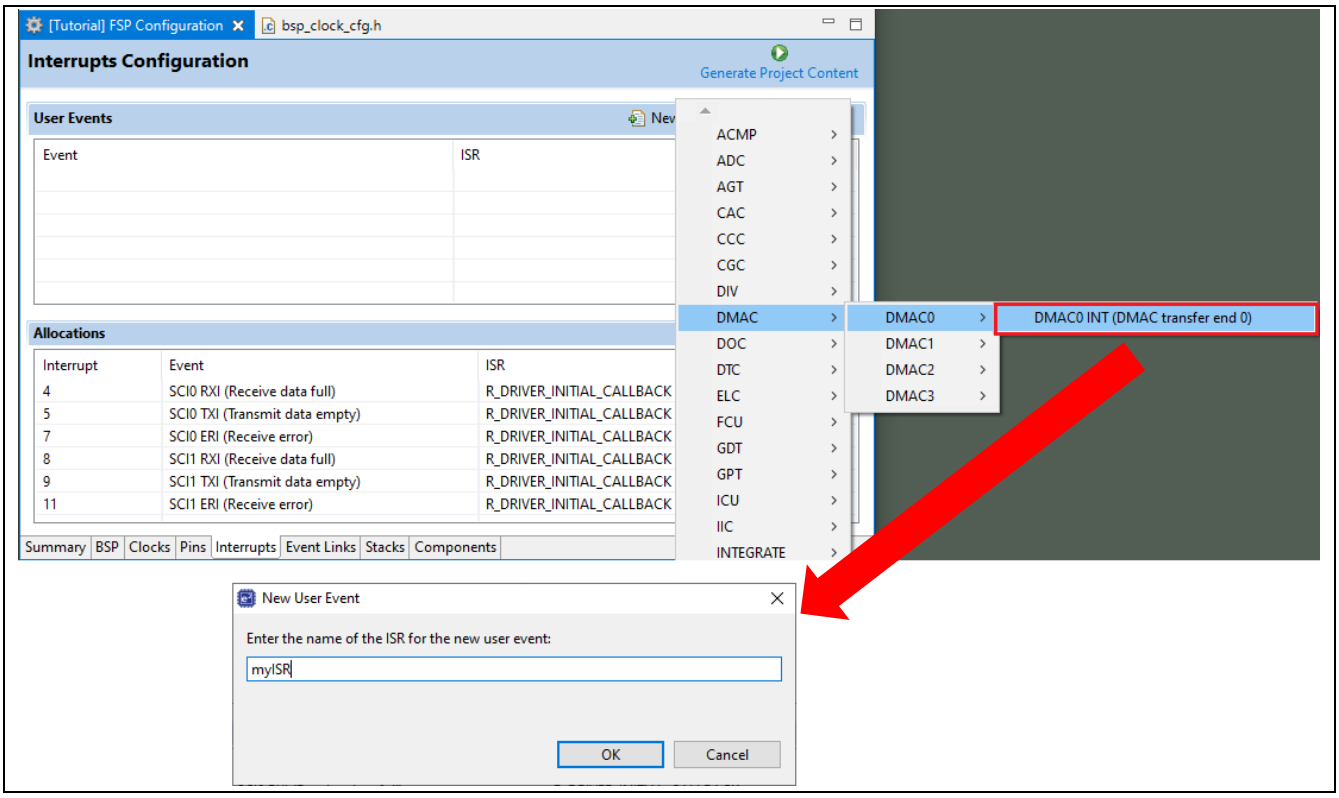


Figure 3-48 Adding a new user event

The newly created event will be displayed in the “User Events” pane.

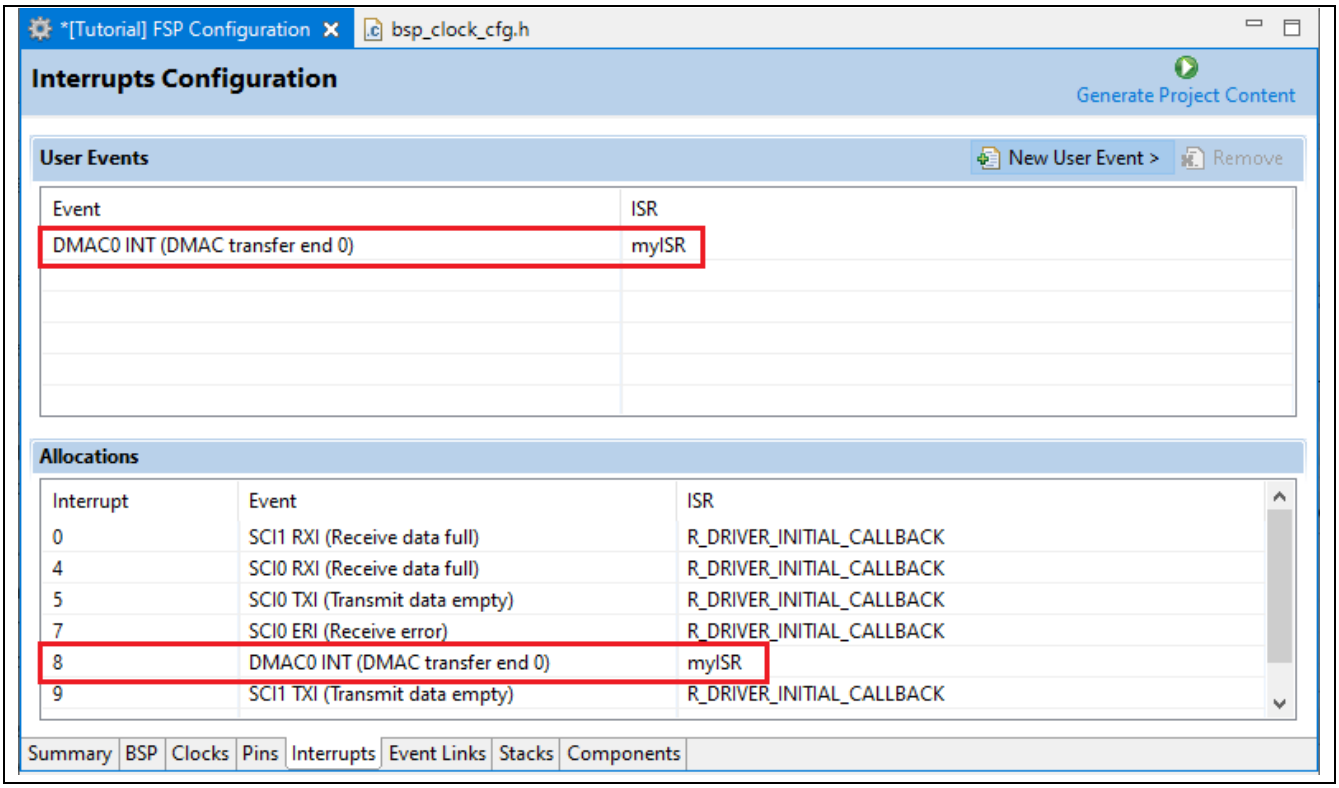


Figure 3-49 User event created

To remove a user event, select the event and click button in the “User Events” pane (events added by instantiated RE modules in the “Allocations” pane cannot be removed).

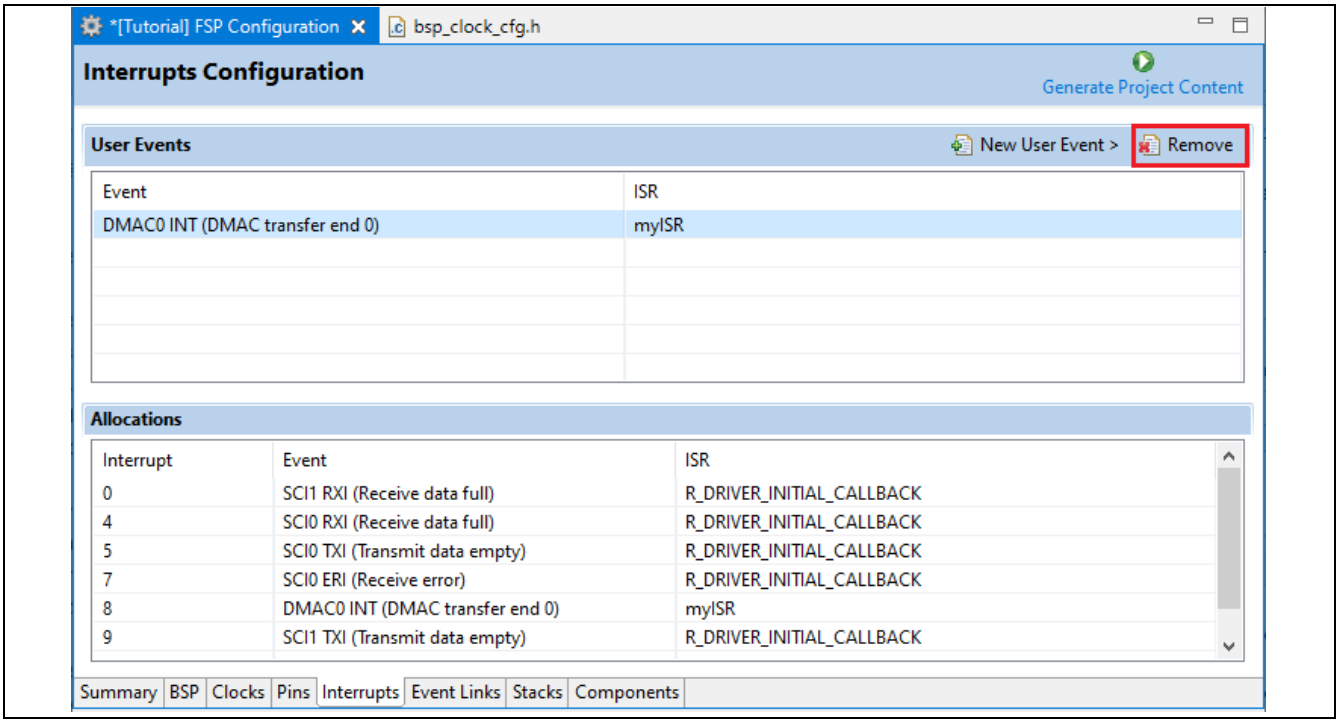


Figure 3-50 Remove user events

Removing a user event can also be done by specifying “Disabled” for it in the Properties window.

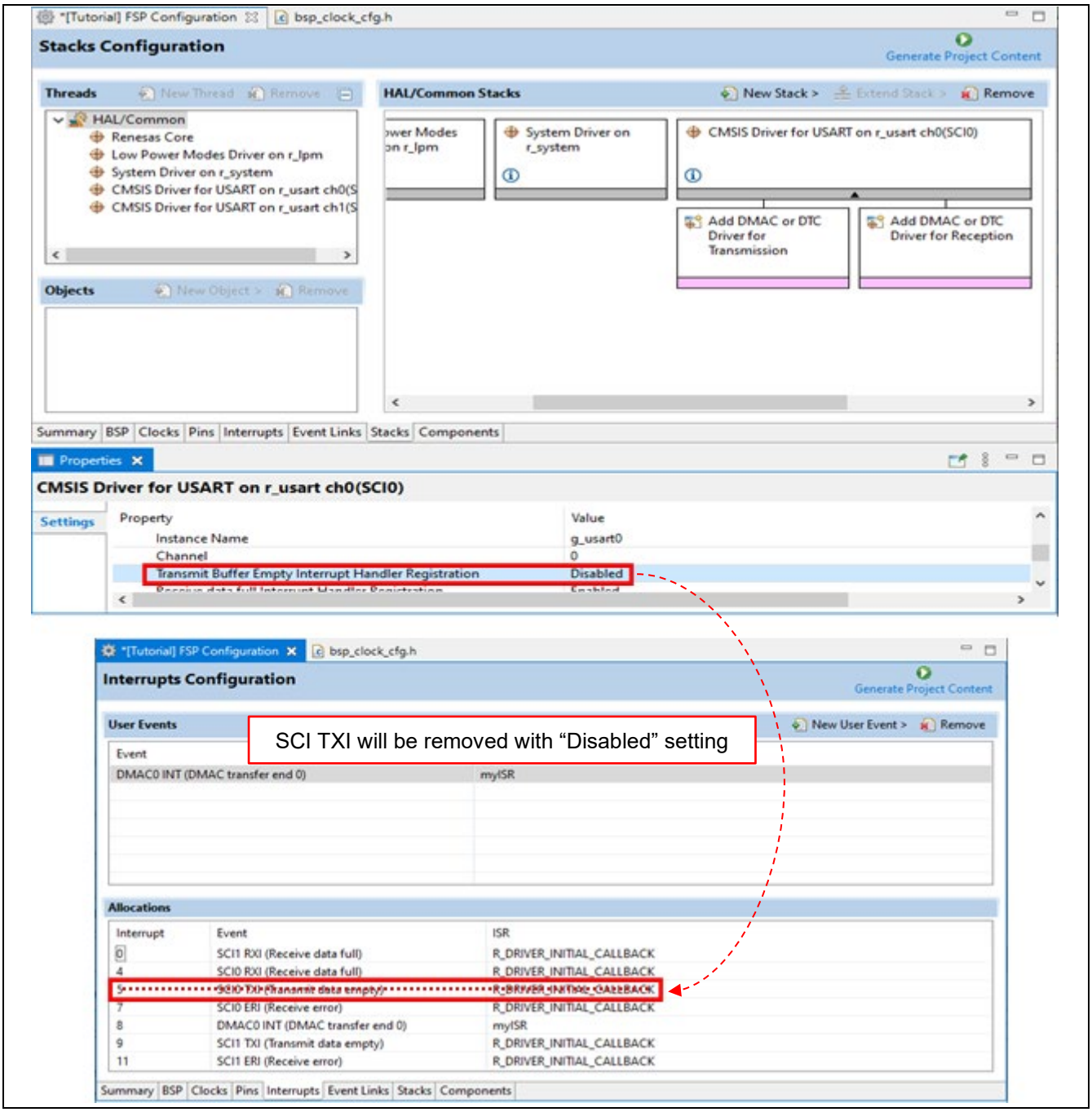


Figure 3-51 Remove user event in Properties window

3.5.7 Components Configuration Page

The Components Configuration page enables the individual modules required by the application to be included or excluded.

Modules common to all RE projects are preselected (for example HAL Drivers → re01_1500kb → r_adc).

All modules that are necessary for the drivers selected in the Stacks page are included automatically. Users can include or exclude additional modules by checking the box next to the required component.

Note: The primary way of adding modules to an application is by using the Stacks page. The Components page is primarily used as a list of components available in the installed Software Package.

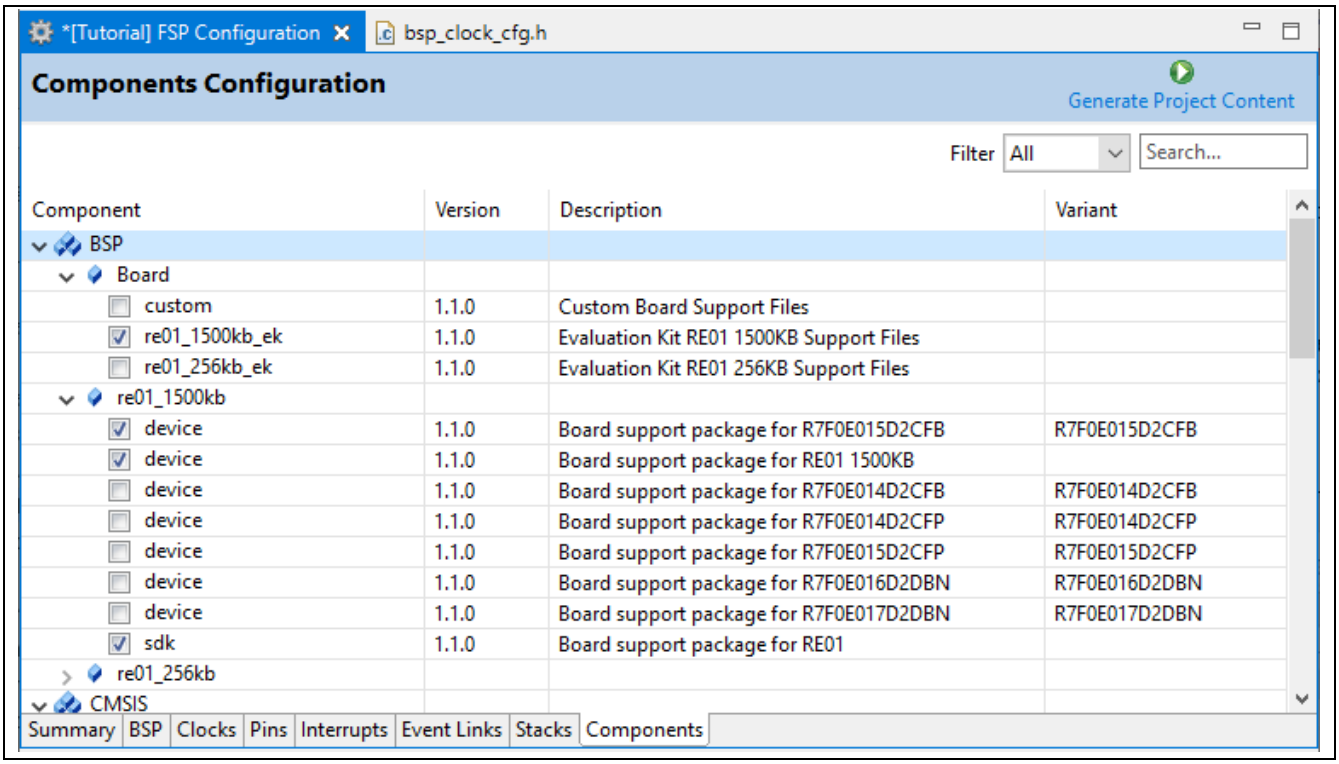


Figure 3-52 Components page

Note: Coding for User Application

User must NOT add user's application code in main.c. Please note that main.c is overwritten each time the [Generate] button is clicked in the configuration window. This is the characteristic of the Smart Configurator. As such, please implement user's application code in hal_entry.c or other *.c files appropriately.

4.Building

This chapter describes the build configurations and key build features for e² studio IDE.

4.1 Build Option Settings

A new project built with the default option can work properly. However, if user would like to change build options (e.g. toolchain version, optimization options, etc.), please follow the following steps before building the project.

1. Right-click on the sample project (e.g. “an4950_gpio_re_1500kb”) and select [Properties] or use shortcut keys [Alt] + [Enter] to open the Properties window.

   Properties window is supported at workspace, project and source level. Properties window for project supports more configurations that apply across all the files within the same project workspace.

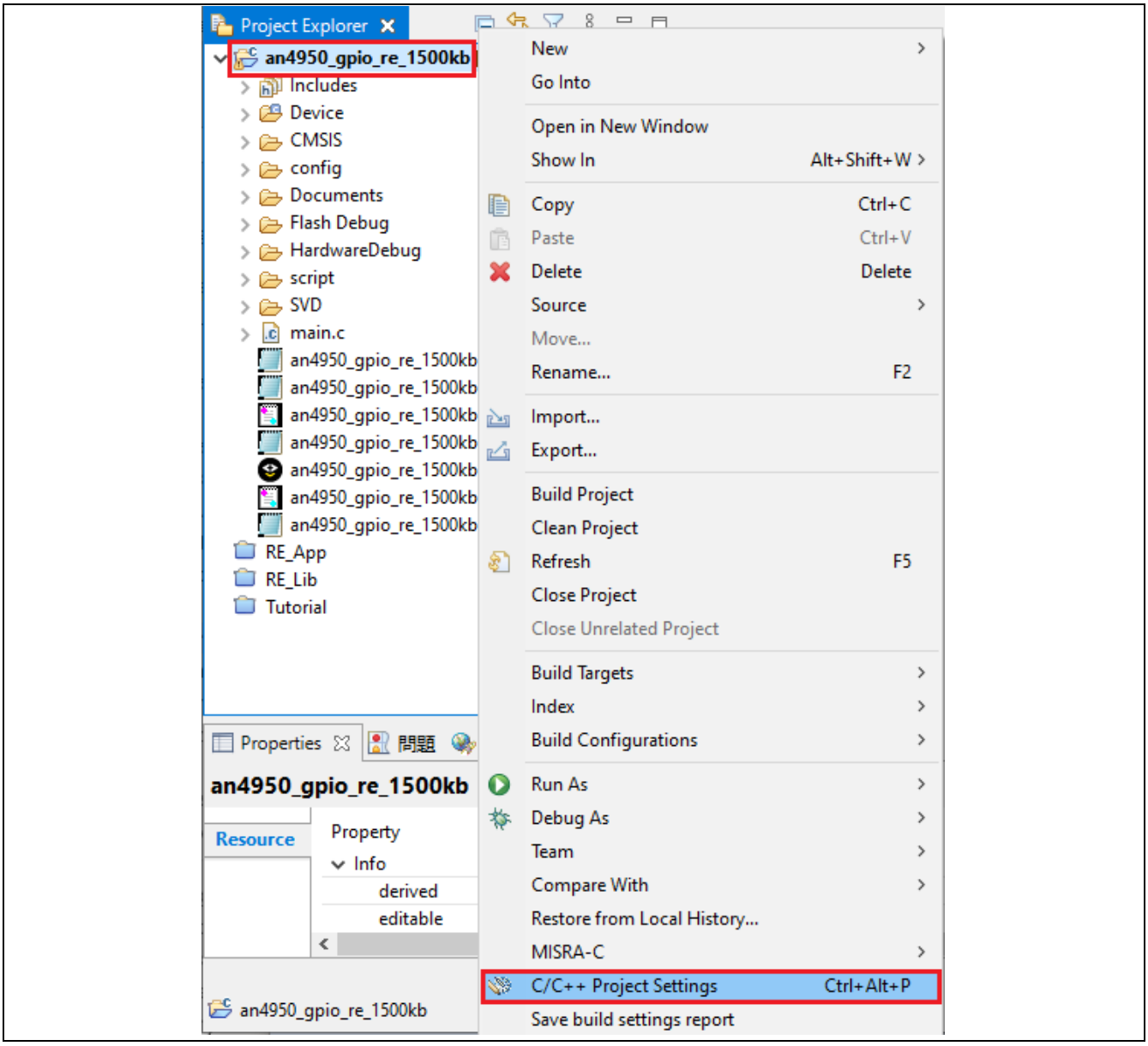


Figure 4-1 Open the Properties window

2. Click [C/C++ Build] → [Settings] → [Toolchain] tab to view or change toolchain version.
   Refer to figure 4-2, the current version is 6.3.1.20170620 and click the “Versions” option to change the toolchain version if necessary.

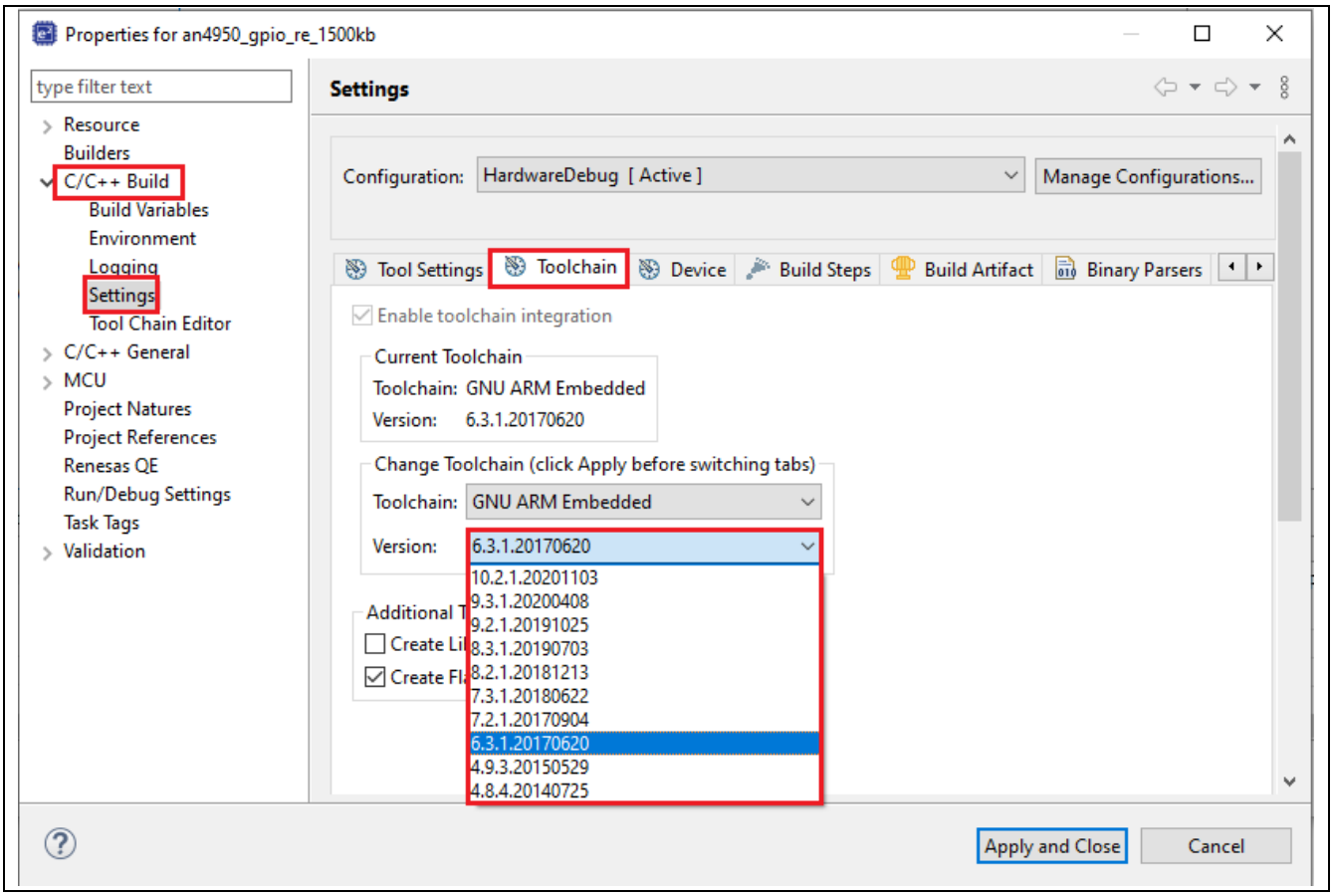


Figure 4-2 Change Toolchain Version

3. Click [C/C++ Build] → [Environment] to set build option and add or edit the environment variables.

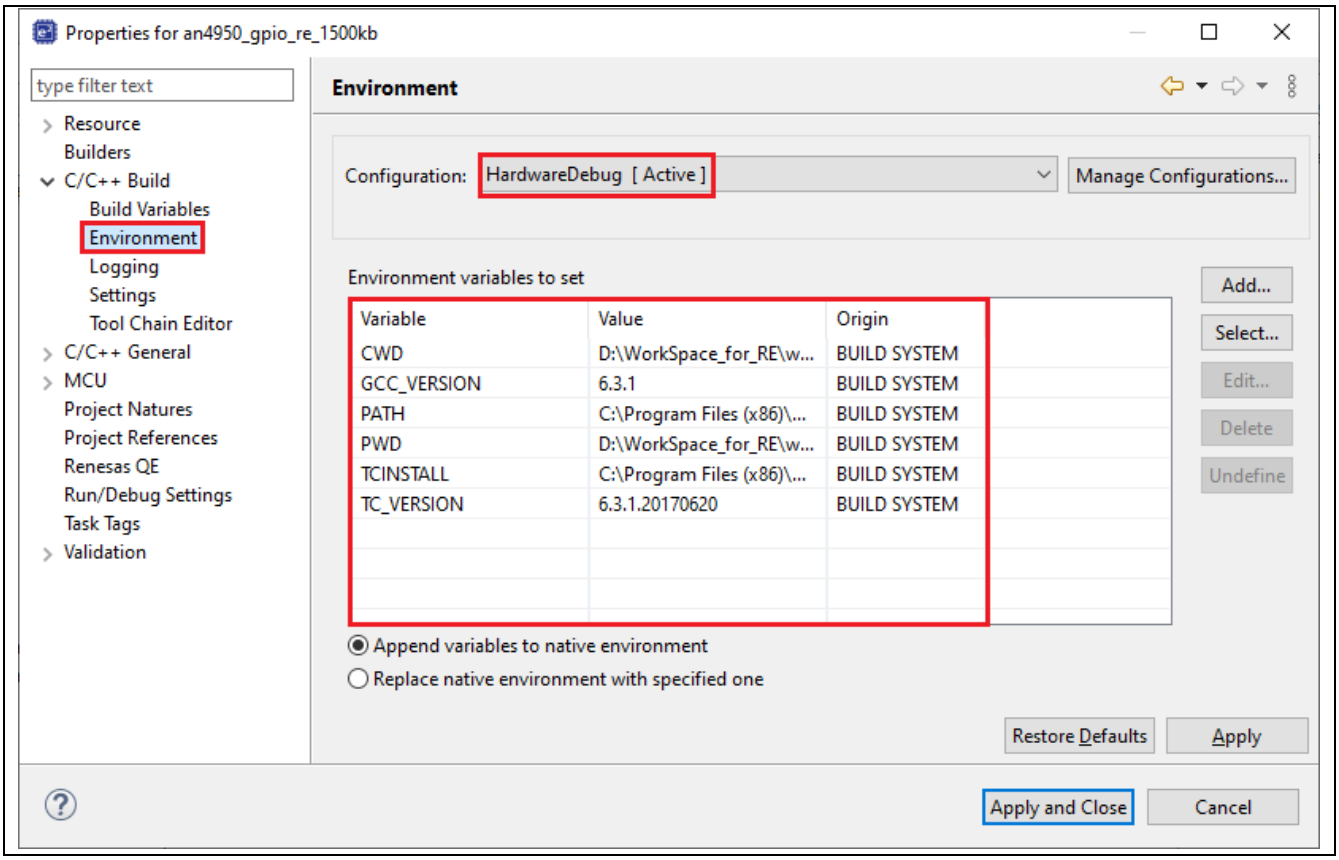


Figure 4-3 Build Environment Settings

4. Setting Build Options

Build options for compiler and linker, etc. can be set on “C/C ++ Build” → “Settings” → “Tool Setting” tab.

All settings are summarized in “All options:” which shows the commands use in the build process.

The “Build configuration” can be switched via the “Configuration:” dropdown list at the top of the window. Each build configuration manages a set of build options.

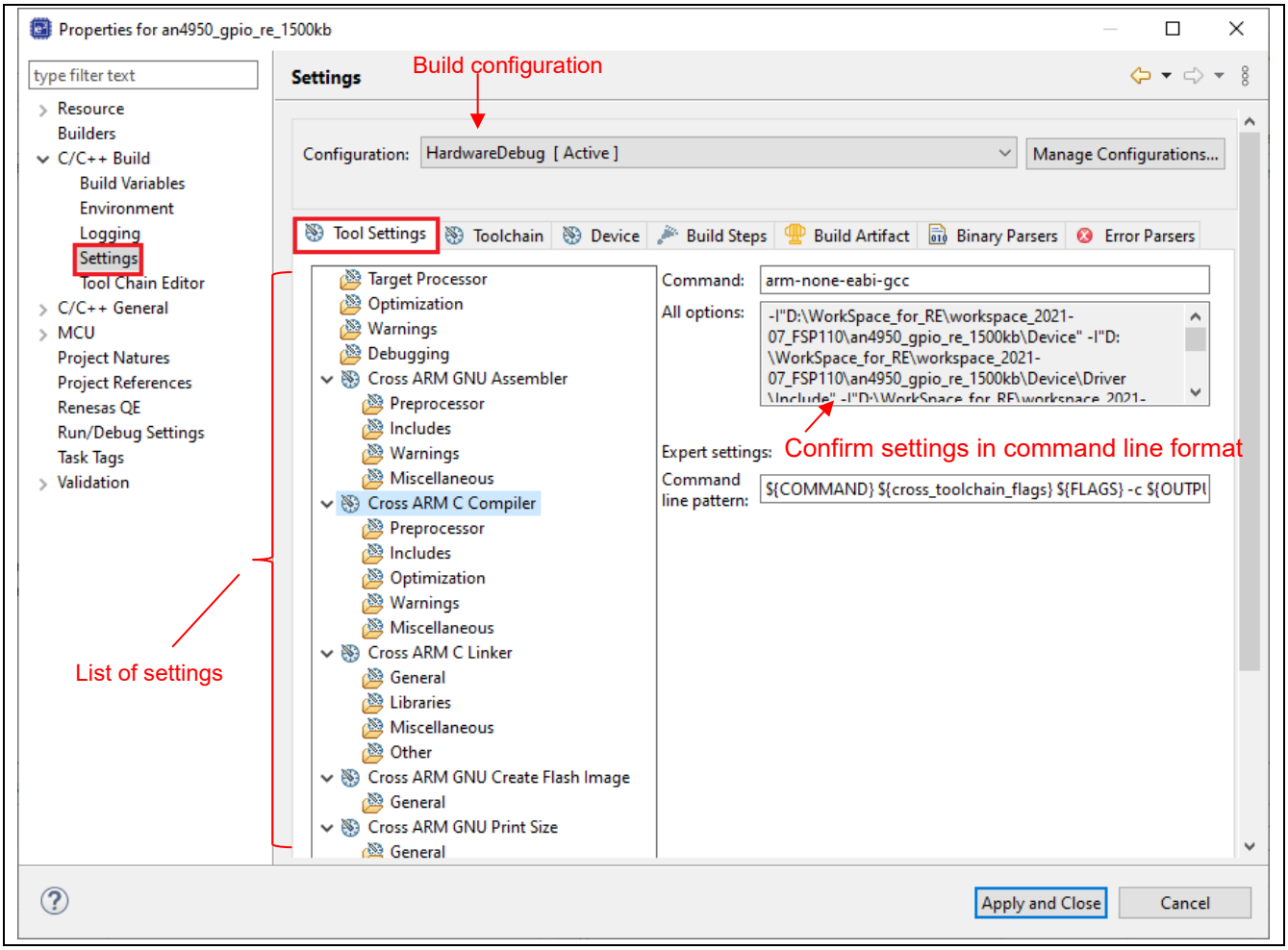


Figure 4-4 Build Option Settings

The detail of build option is described in the compiler user manual which is stored at “{Compiler installation directory}\share\doc”. For example, it can be found in “C:\Program Files (x86)\GNU Tools ARM Embedded\6 2017-q2-update\share\doc".

4.1.1 Recommended Build settings

- As you see, there is "*Toolchain Editor*" under "C/C++ Build", **please do not change the configuration**. The Toolchain editor is used for toolchains that are NOT supported by Renesas build support plugins.
- Please remove all checked options in “Tool Settings” → “Warnings” for now.

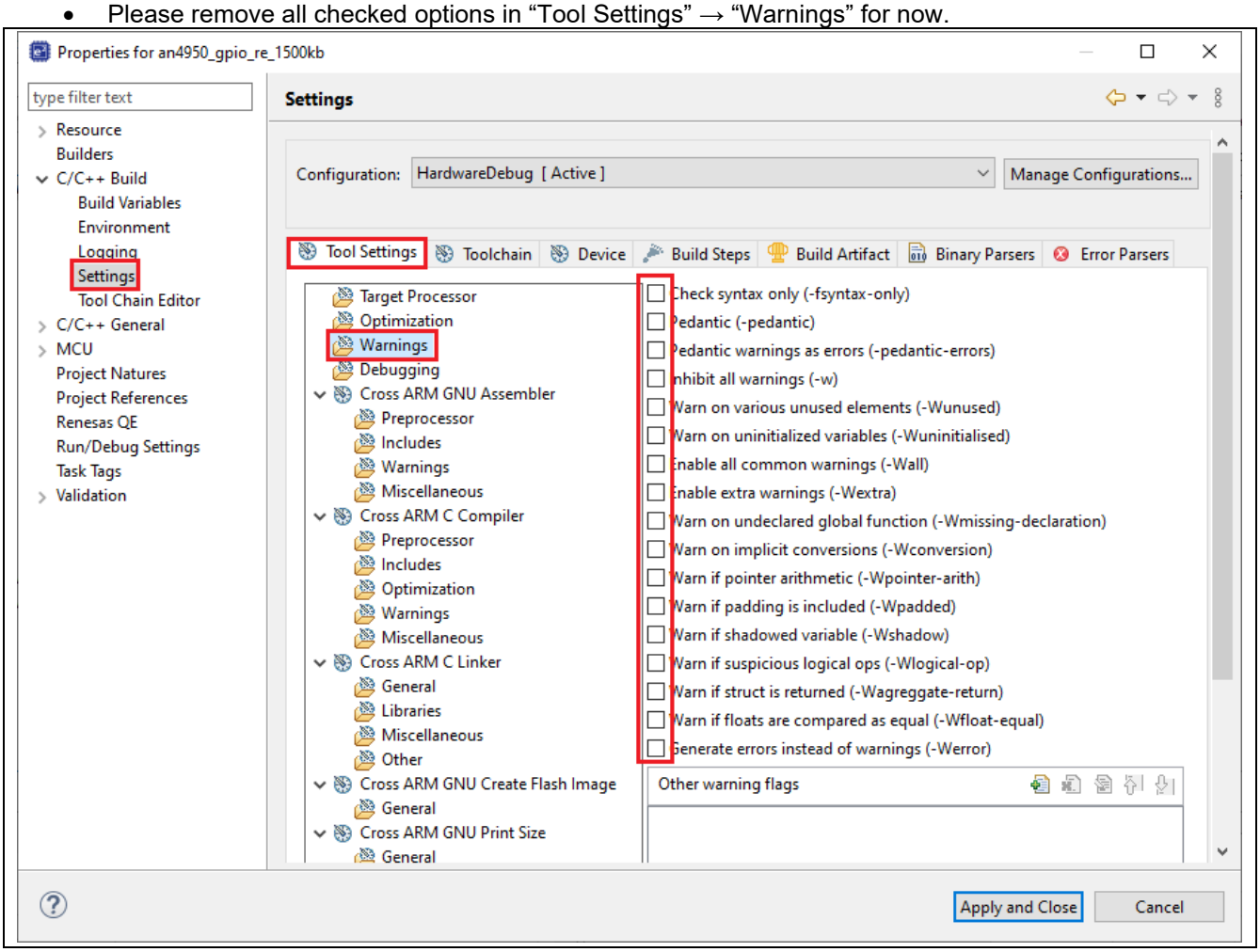


Figure 4-5 All “Warning” options should be unchecked

- This driver has been evaluated without “newlib-nano” option. If there is an issue with using the standard library, please disable this option (it is enabled by default when project is created).

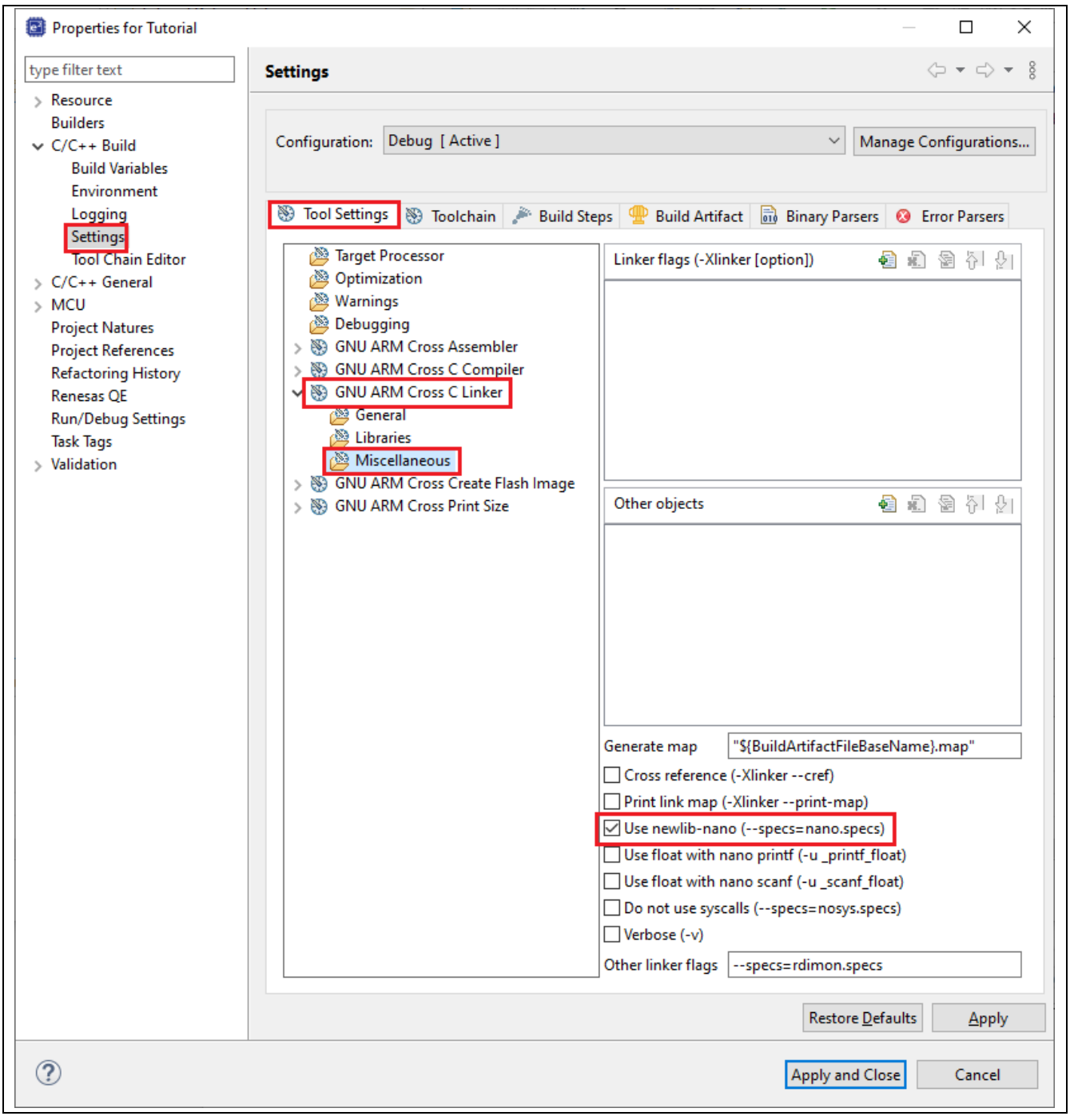


Figure 4-6 “newlib-nano” option

4.2 Build a Sample Project

A project can be built by steps below:

1. Right-click on the project and select [Build Project]

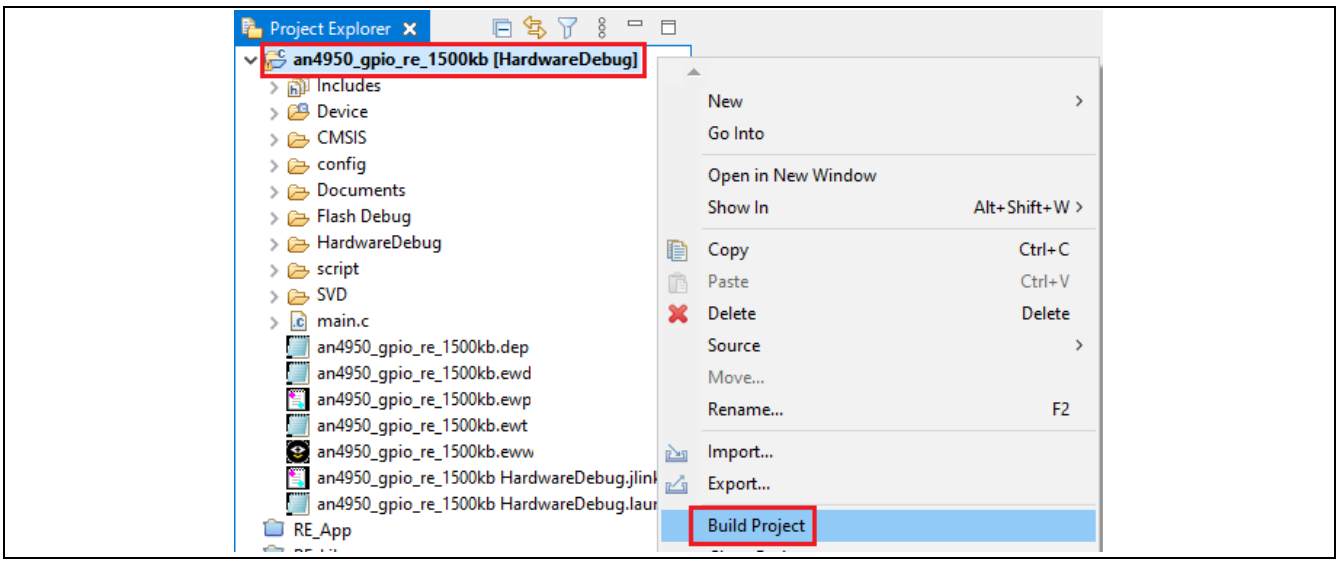


Figure 4-7 Build a Sample Project

2. The [Console] pane shows 'Build complete.' message to indicate a successful build.

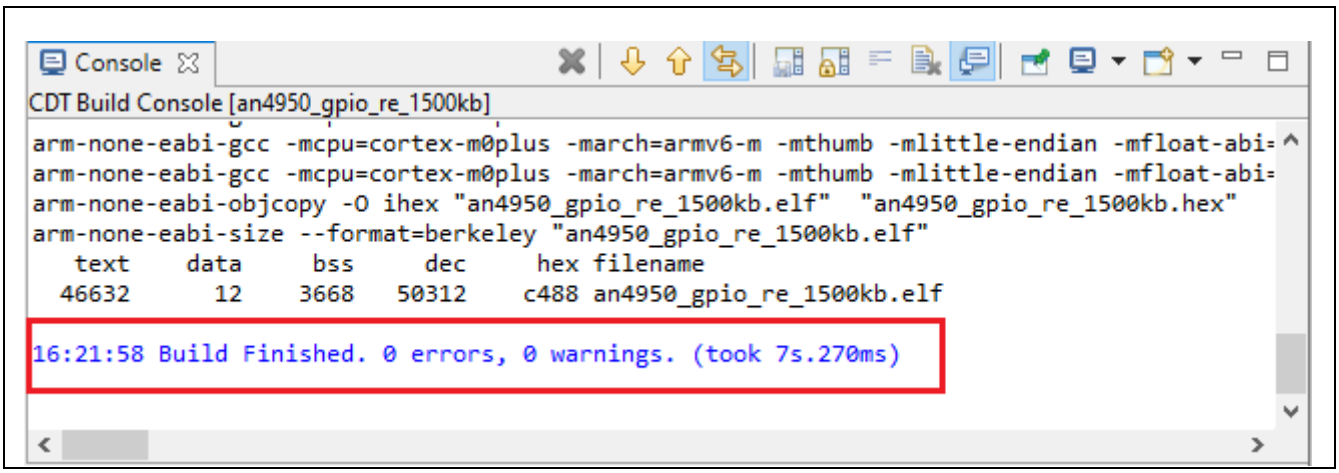


Figure 4-8 Project is built successfully.

3. If the build is unsuccessful, the console window will show error messages which contain details about the errors (location where the error occurs (file, column, and line), reason…). Please check these errors and revise the source code or configuration and rebuild the project.

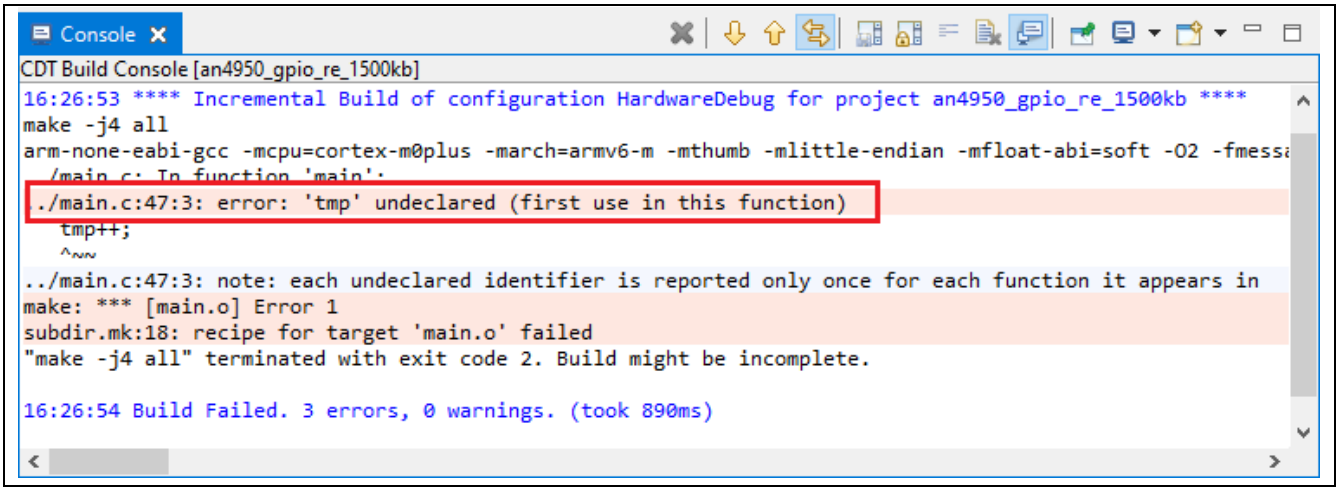


Figure 4-9 Unsuccessful build shows error messages

4.3 Export Build Configuration Settings

Project build settings in e² studio IDE can be saved to a file using the Project Reporter feature.

1. Right-click at [Project Explorer] to pop up the context menu
2. Select [Save build settings report] to save build settings report

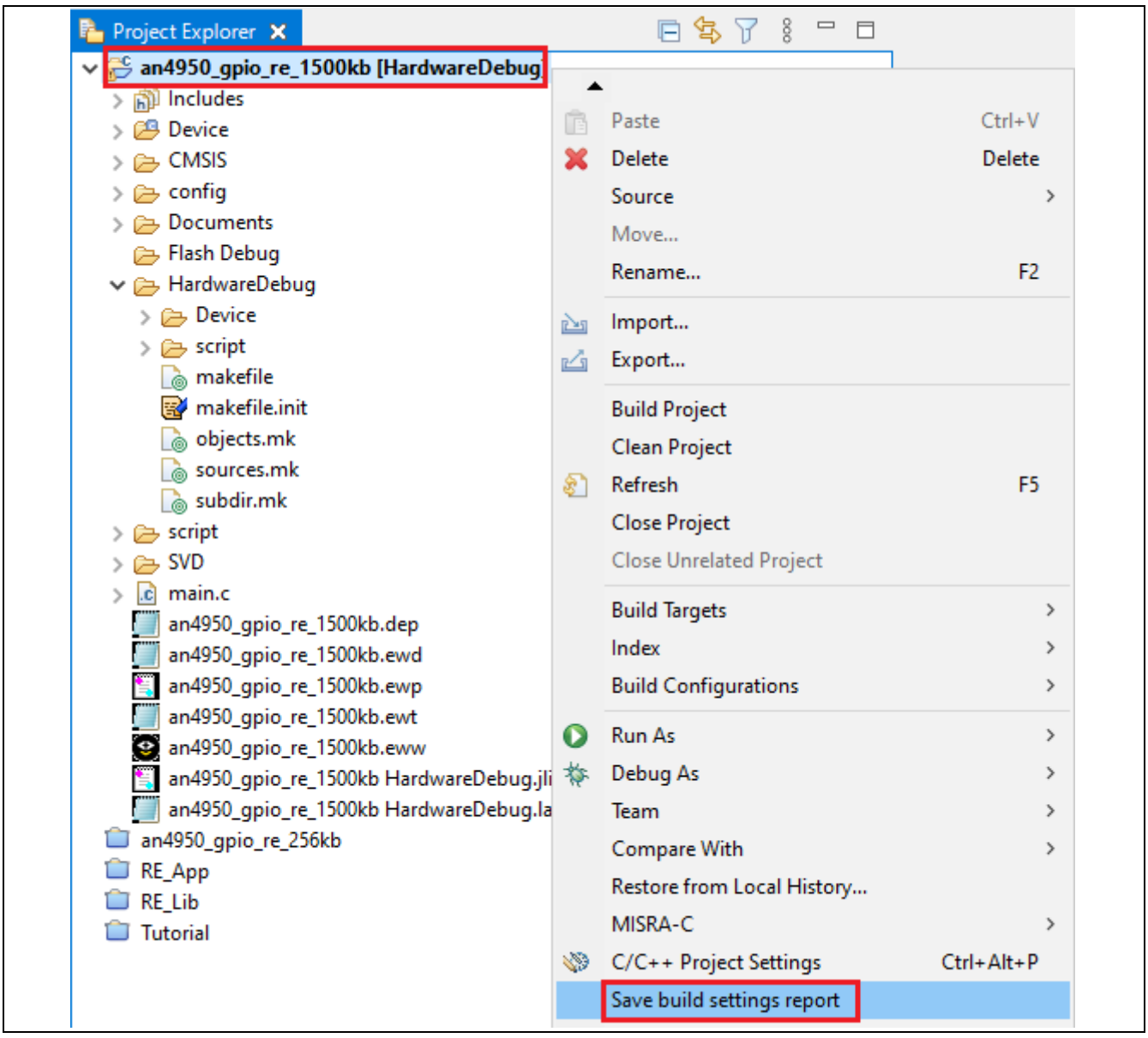


Figure 4-10 Project Reporter

5. Debug

This chapter describes the usage of debug configuration and key debugging features for e² studio. The following illustration refers to the sample project which is downloaded and imported in chapter 3.4 (e.g. “r01an4950ej0101”, hereinafter called “sample project”) and based on hardware configuration: J-Link ARM and EK-RE01 1500KB board.

1. Open the sample project workspace in e² studio IDE and click [Debug] perspective.

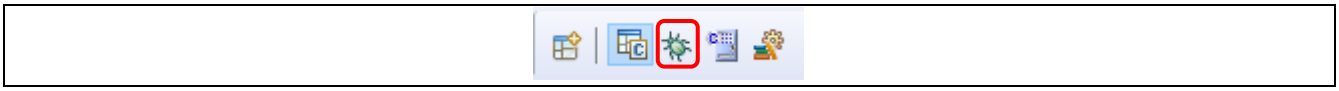


Figure 5-1 Switch to [Debug] Perspective

Perspective defines the layout views (related to development tools) in the Workbench window. Each perspective consists of a combination of views, menus and toolbars that enable users to perform specific tasks. For instance,

- The [Debug] perspective has views that enable the user to debug the program
- The [RE Configuration] perspective together with “configuration.xml” in the editor window will open the RE configuration, as well as the Package and Properties views for project configuration settings
- The [C/C++] perspective has views that help the user to develop C/C++ programs.

If a user attempts to connect the debugger when not in the [Debug] perspective, e² studio will prompt the user to switch to the [Debug] perspective. One or more perspectives can exist in a single Workbench setup. Users can customize them or add new perspectives.

5.1 Change Existing Debug Configurations

The debug configuration must be configured when debugging for the first time and it just needs to be done once. An existing debug configuration can be changed as follows.

1. Click the sample project in the [Project Explorer] pane to set focus.

   Click [Run] → [Debug Configurations…] or icon (downward arrow) → [Debug Configurations…] to open the “Debug Configurations” window.

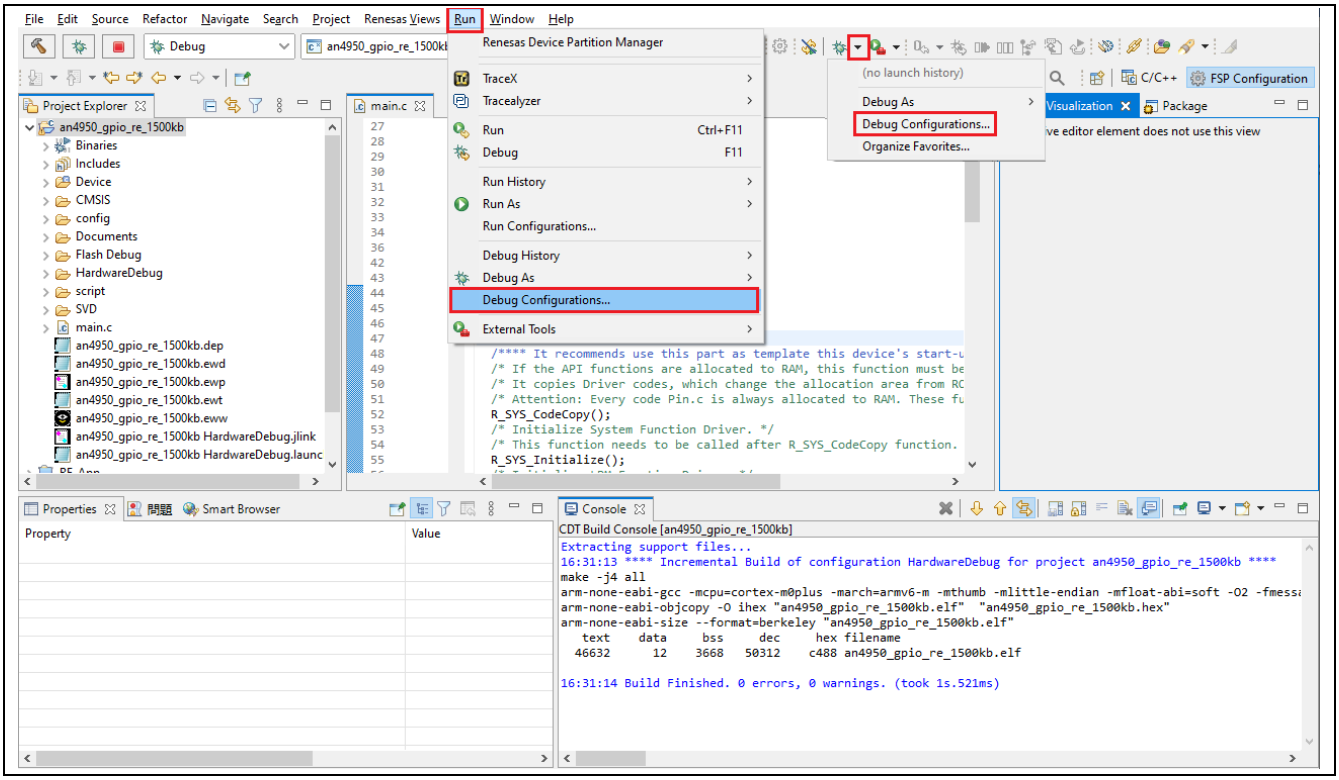


Figure 5-2 Open Debug Configurations Window

2. In “Debug Configurations” windows, go to [Renesas GDB Hardware Debugging] → [an4950_gpio_re_1500kb.elf HardwareDebug]. Click on the [Main] tab to ensure the load module is “an4950_gpio_re_1500kb.elf”.

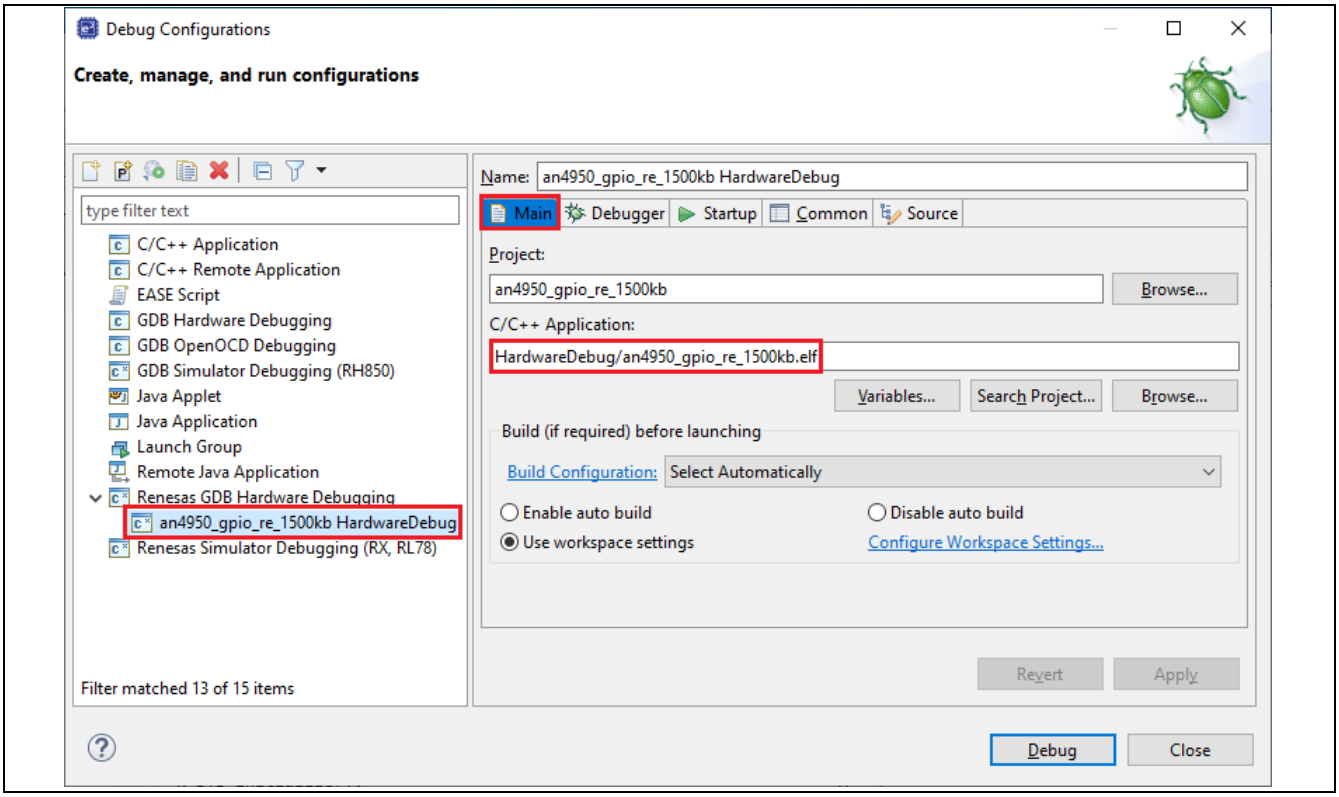


Figure 5-3 Select Load Module

3. Switch to the [Debugger] tab, set “J-Link ARM” as the debug hardware and “R7F0E015D2CFB” as the target device.

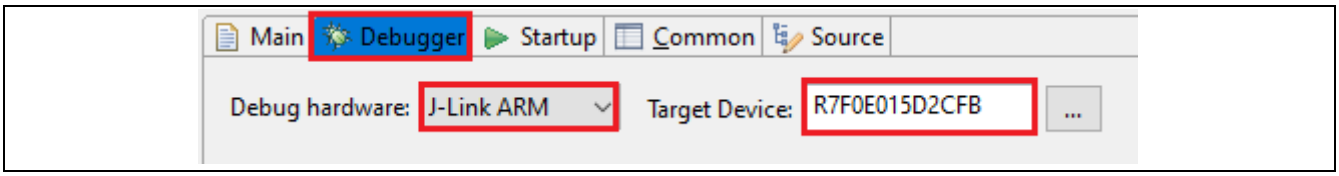


Figure 5-4 Select Target Device

4. Under the [Debugger] tab, go to the [Connection Settings] sub-tab which is related to emulator connection. The following example is based on the environment with J-Link ARM emulator and RSK RE01 board:

   - J-Link
     - Type = “USB”.
     - Low Power Handling = “Yes”
   - Interface
     - Type = **“SWD”**
     - Speed = “**320**”
   - Connection
     - Reset before run = “**Yes**”
     - Reset before download = “**Yes**”

   Note: This debug configuration in Figure 5-5 is shown as an example. The wrong settings may cause malfunction or damage to the hardware. So, be cautious to verify the board and emulator settings before connection.

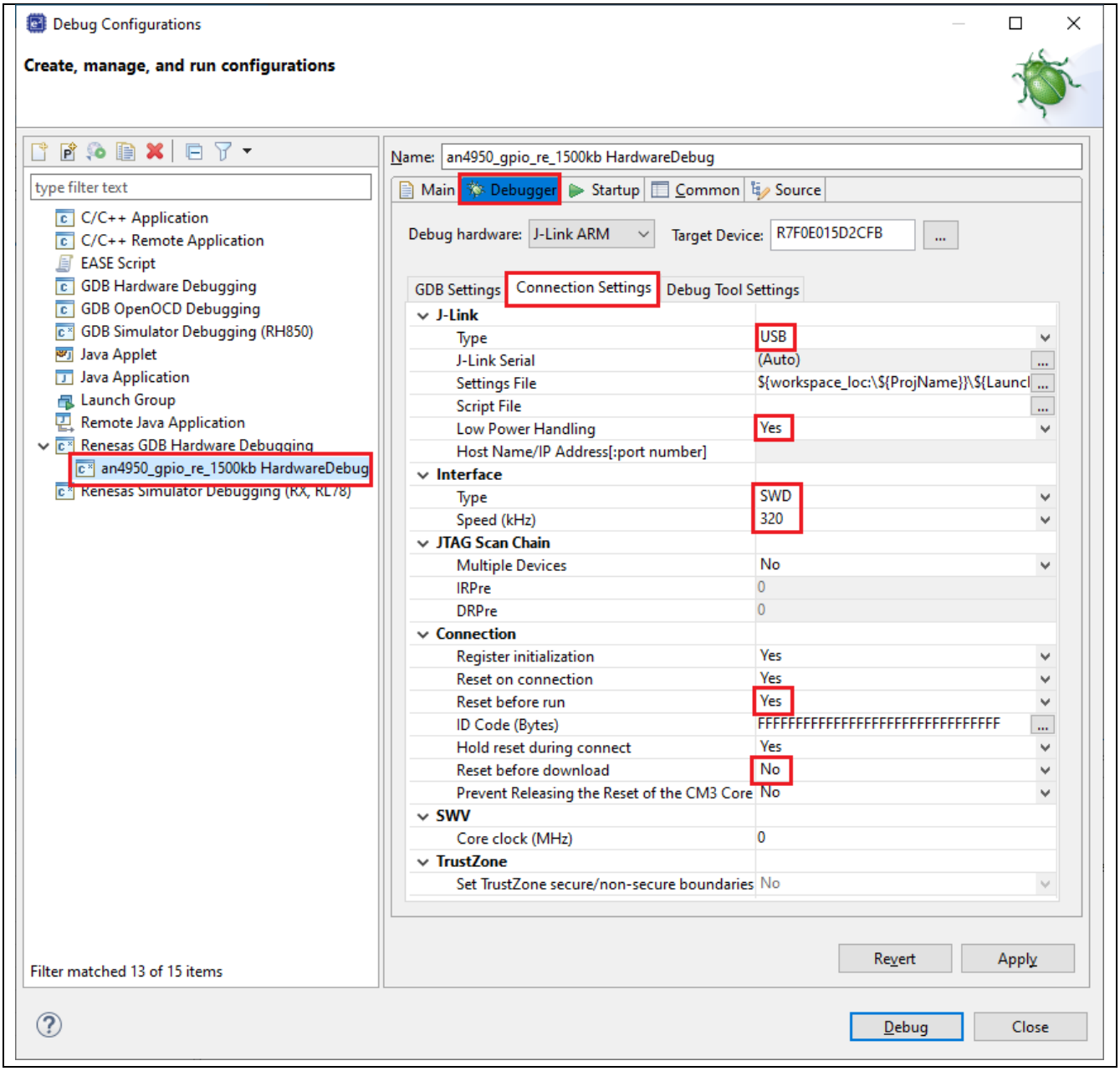


Figure 5-5 Change Connection Settings

5. Switch to [Debug Tool Settings] sub-tab which is related to debugger behavior. Please refer to the e² studio Help content at "e² studio User Guide" → "Debugging Projects" for the details.

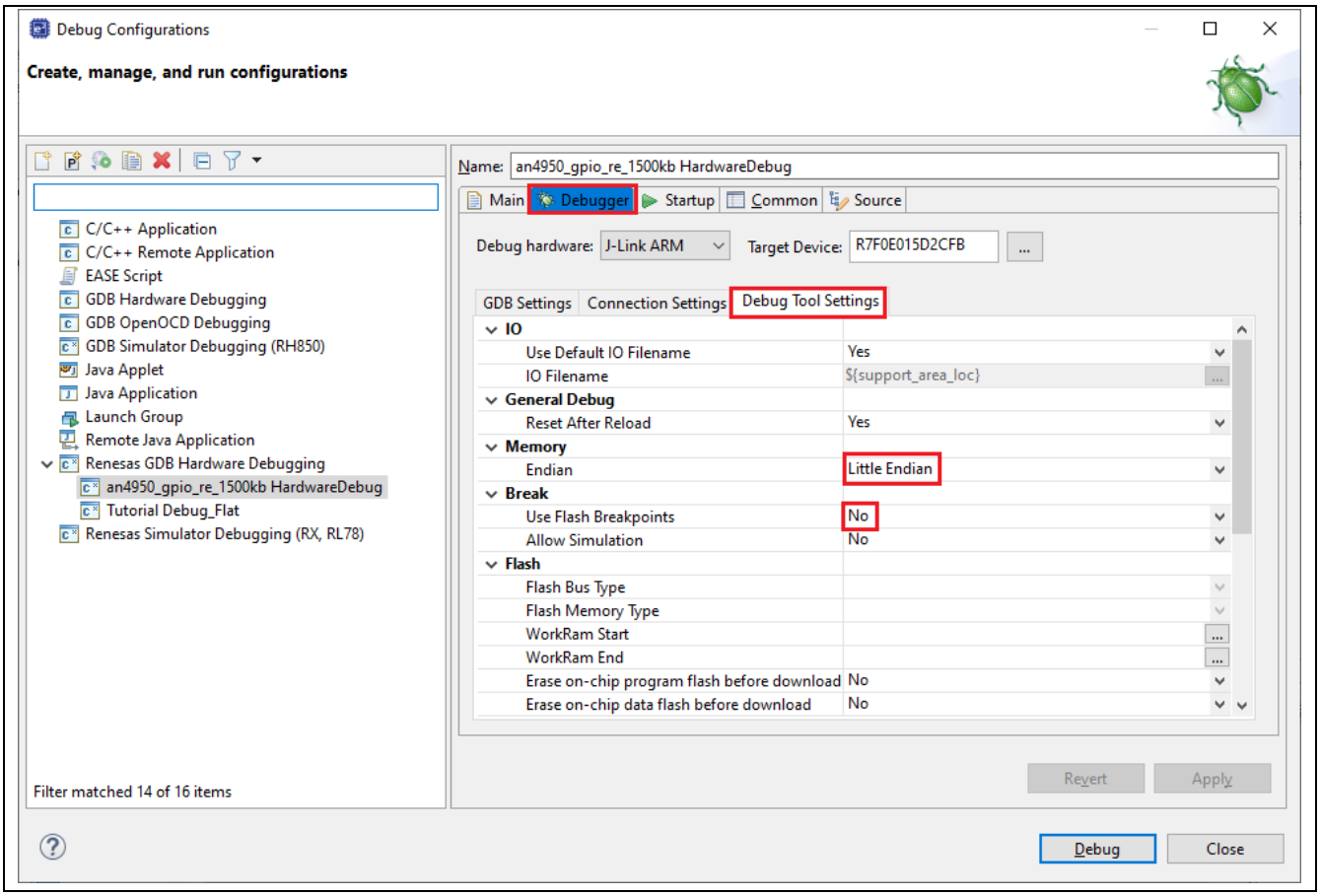


Figure 5-6 Change Debug Tool Settings

- Memory

  — Endian = “Little Endian”

  Endian setting of debugger memory reference. This configuration does not affect the target program behavior.

- Break

  - Use Flash Breakpoints = “No”

  Setting software breakpoints to flash memory area is restricted, so the hardware breakpoints must be used.

6. Click [Apply] button to confirm the settings. Then click [Debug] to launch the debugger.

7. For a successful connection, [Debug] view to show target debugging information in a tree hierarchy. The program entry point is set at “Reset_Handler() in “startup_RE01_1500KB.c”.

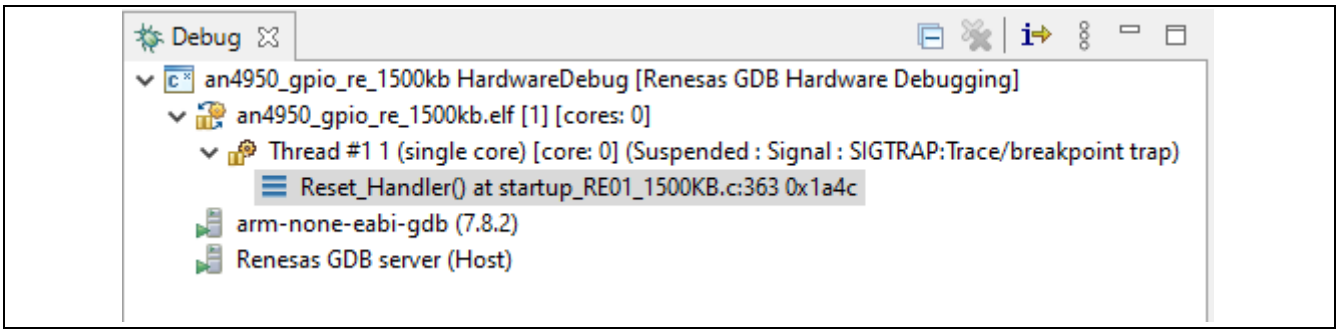


Figure 5-7 User Target Connection in the [Debug] View

5.2 Create New Debug Configurations

The simplest way to create a new debug configuration is by duplicating an existing one. It can be done by the following steps.

1. Repeat step 1 in section 5.1 to open the “Debug Configurations” window.

2. Select a debug configuration (e.g. “an4950_gpio_re_1500kb HardwareDebug”) and then click icon (to duplicates the currently selected launch configuration). A new debug launch configuration (e.g. “an4950_gpio_re_1500kb HardwareDebug (1)”) is created. User can rename it to identify the settings by typing in the “Name” textbox then click [Apply] button.

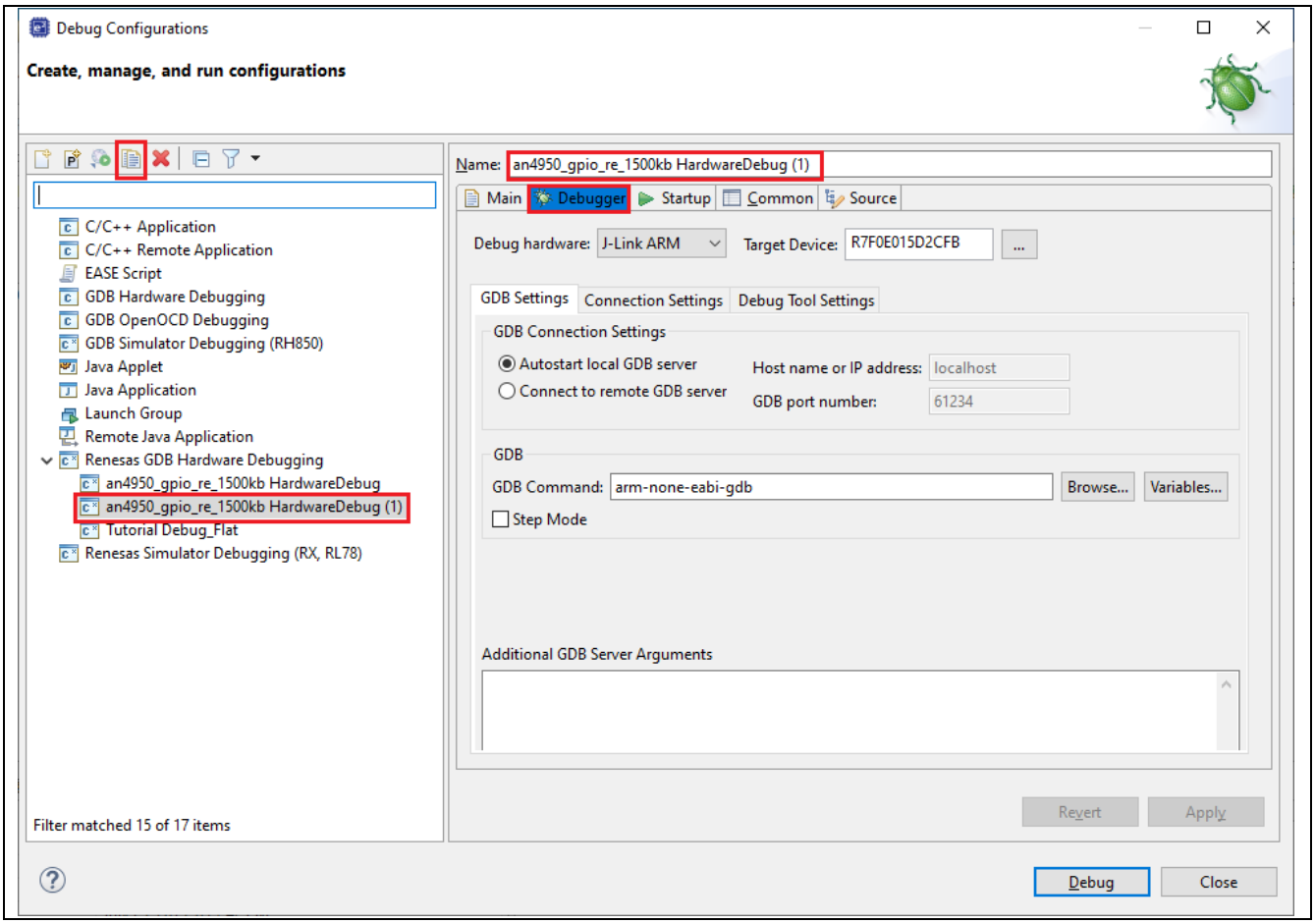


Figure 5-8 Duplicate A Selected Debug Launch Configuration

3. The debug launch configuration can be configured as described in chapter 5.1. For example, change the Debug Hardware to “E2 Lite (ARM)”.

4. If the launch configuration was added with [local] and * (red star) marker, it is not yet attached to any project. Then please specify the project name in the Common tab.

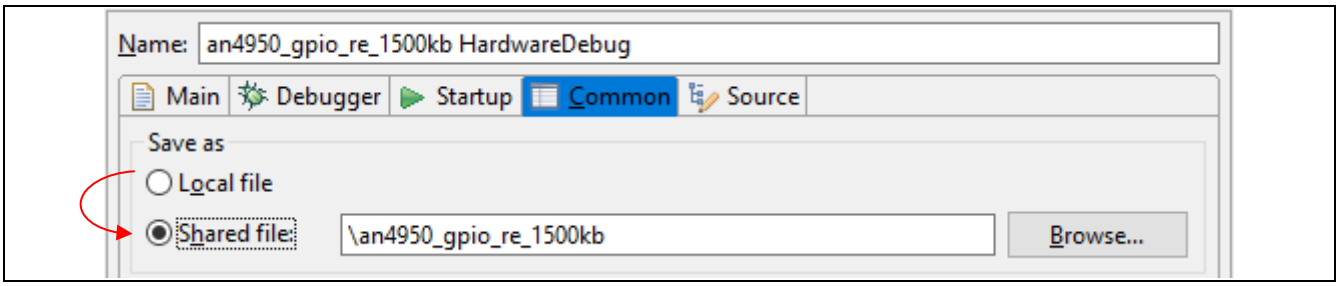


Figure 5-9 Attach Launch Configuration to Specific Project

5.3 Launch Bar

This section explains the usage of 'Launch Bar', which is supported from V6.0.0 or later version.
Launch Bar is located in the toolbar area of e² studio main window.
The interface is very simple as shown below to build and debug for the selected launch target.

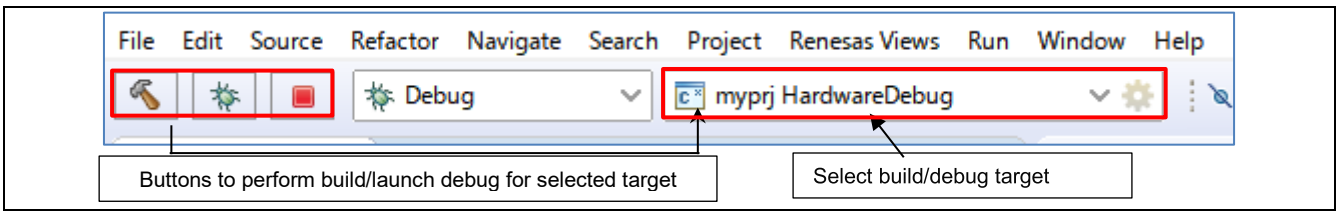


Figure 5-10 Launch Bar interface

Launch Bar buttons behave as follows (please select the build/ debug target in advance):

- button builds the load module of the selected launch configuration.

  Note: There is another build button in the "File toolbar" that builds the active build configuration of Project Explorer, while the launch bar does not reflect the active state in Project Explorer.

- buttons are triggers of debugger launch and terminate the selected launch target.

Launch Bar and build button can be hidden through the following dialog.

- Click [Window] menu → [Preferences], then click [Run/Debug] → [Launching] → [Launch Bar].

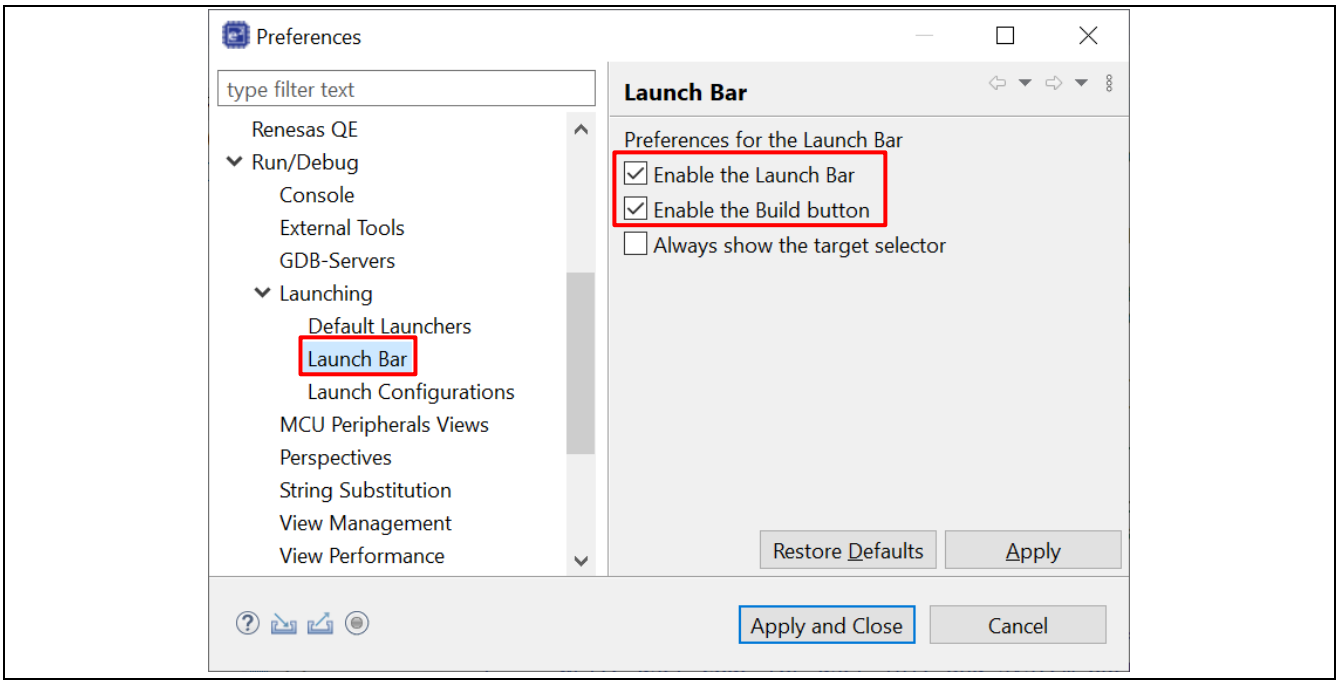


Figure 5-11 Show/hide Launch Bar

5.4 Basic Debugging Features

This section explains the typical Debug views supported in e² studio IDE.

- Standard GDB Debug (supported by Eclipse IDE framework): Breakpoints, Expressions, Registers, Memory, Disassembly and Variables
- Renesas Extension to Standard GDB Debug: Eventpoints, IO Registers and Trace.

The following are some useful buttons in the [Debug] view:

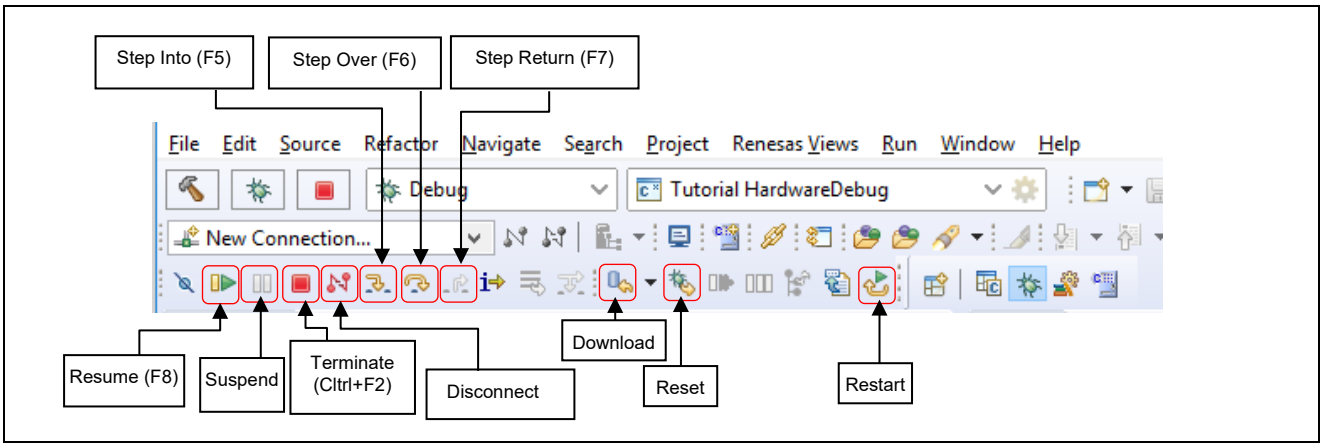


Figure 5-12 Useful Toolbars in Debug Views

The program is run by clicking button or pressing [F8].

The program can be paused by breakpoint or by clicking button. When the program is paused, user can perform the following operations:

- button or [F5] can be used for stepping into the next method call at the currently executing line of code.
- button or [F6] can be used for stepping over the next method call (executing but without entering it) at the currently executing line of code.
- button can be clicked again to resume running.
- To stop the debugging process, button is clicked to end the selected debug session and/or process or button is clicked to disconnect the debugger from the selected process.

The other operations are as following:

- button can be clicked to start new debug session.
- button can be clicked to reset the program to entry point at the PowerOn Reset.
- button is used for re-downloading the binary file to target system.

Note: To demonstrate the features in the following section, please use the sample code for RE01 from Renesas website as instruction in chapter 3.4.

5.4.1 Breakpoints View

The Breakpoints view stores the breakpoints that were set on executable lines of a program. If a breakpoint is enabled during debugging, the execution suspends before that line of code executes. e² studio allows software and hardware breakpoints to be set explicitly in the IDE. Any breakpoints added via double click on the marker bar are by default hardware breakpoints. If the hardware resources are not there then the breakpoint setting will fail. In case of a hardware breakpoint setting failure, an error message will prompt the user to switch to a software breakpoint.

To select a default Hardware or Software breakpoint type:

- Right-click on the marker bar to pop up the context menu. For a hardware breakpoint, select [Breakpoint Types] → [e² studio Breakpoint]. For a software breakpoint, select [Breakpoint Types] → [C/C++ Breakpoints].

Note: *Software breakpoint is prohibited in flash memory area, hardware breakpoint is not supported in RAM area.*

To set breakpoints in flash memory area, please use hardware breakpoint.

To set breakpoints in RAM area, please use software breakpoint.

To set a breakpoint:

1. Open “main.c”, double-click on the marker bar located in the left margin of the [C/C++ Editor] pane to set a breakpoint. A dot (Hardware breakpoint) or (Software breakpoint) is displayed in the marker bar depending on the [Breakpoint Type] selected. [Breakpoint Type] is hardware breakpoint by default.

2. Alternatively, right-click at the marker bar to choose [Toggle Hardware Breakpoint] or [Toggle Software Breakpoint] to set a hardware breakpoint or a software breakpoint .

3. Click [Windows] → [Show View] → [Breakpoints] or icon (or use shortcut key [Alt] + [Shift] + [Q], [B]) to open the [Breakpoints] view to view the corresponding software breakpoints set. Software breakpoints can be enabled and disabled in the [Breakpoints] view.

To disable breakpoints, users can choose to disable specific breakpoints or to skip all breakpoints:

1. To disable a specific breakpoint, right-click on the Software breakpoint or Hardware breakpoint located in the left margin of the [C/C++ Editor] pane and select [Disable Breakpoint], or uncheck the related line in the Breakpoints view. A disabled breakpoint is displayed as a white dot (or).

2. To skip all breakpoints, click on the icon in the Breakpoints view. A blue dot with a backslash will appear in the editor pane as well as in the Breakpoints view.

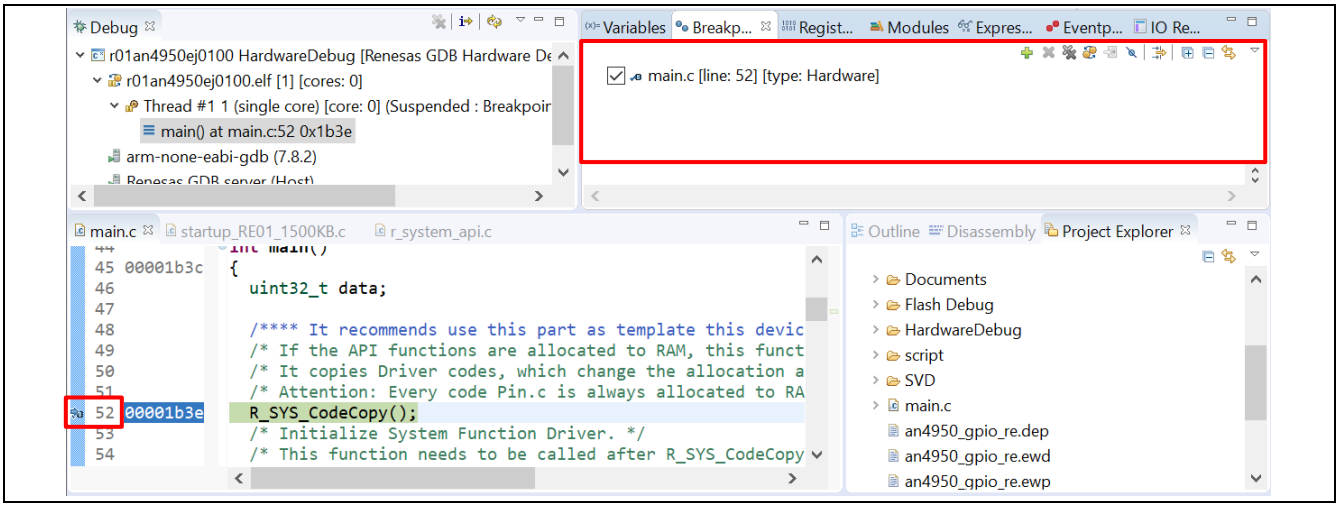


Figure 5-13 [Breakpoints] view

5.4.2 Expressions View

Expressions view monitors the value of global variable, static variable or local variable during debugging. These variables (including the local variables in scope) can be set for real-time refresh.

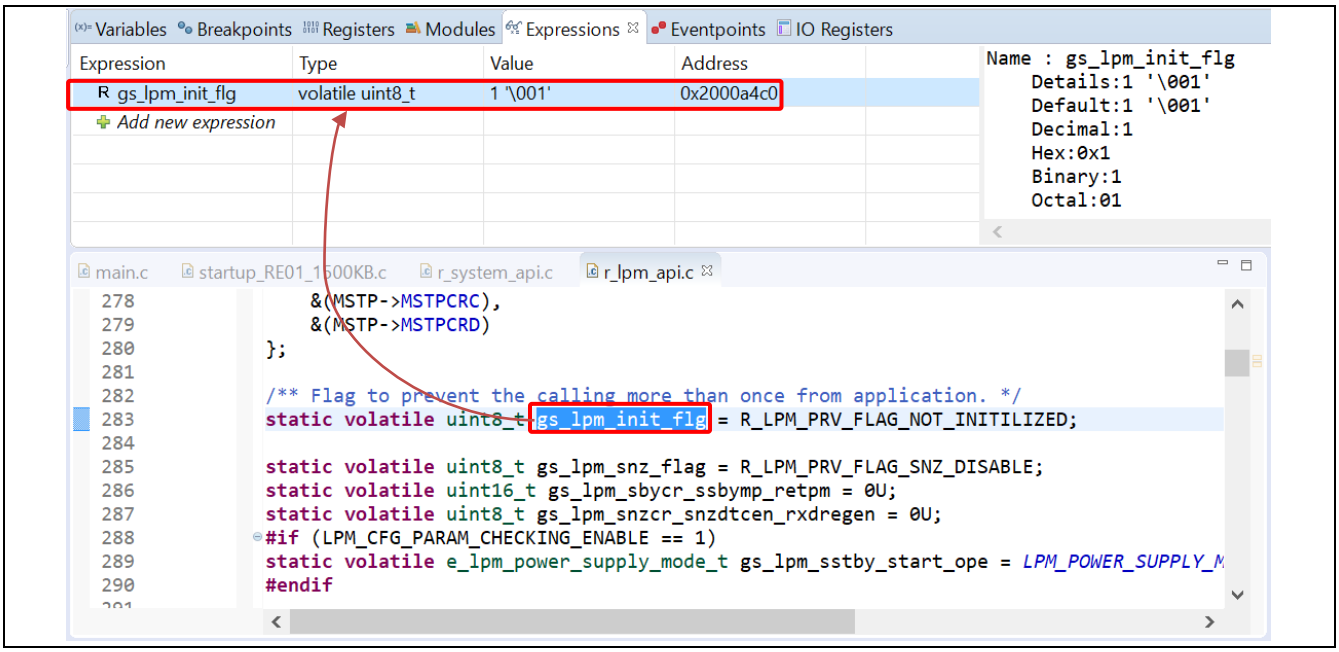


Figure 5-14 [Expressions] View

To watch a global variable,

1. Click [Window] → [Show View] → [Expressions] or icon to open the [Expressions] view
2. Drag and drop a global variable over to the [Expressions] view. (Alternatively, right-click at the global variable to select “Add Watch Expression…” menu item to add it to the [Expressions] view).
3. In the [Expressions] view, right-click to select “Real-time Refresh” menu item. This refreshes the expression value in real-time when the program is running. The character “R” indicates that this global variable will be updated in real-time.
4. To disable the “Real-time Refresh”, simply right-click to select “Disable Real-time Refresh” menu item.

Local variables can be added in the same way. However, the watch is not available when the program is running out of the scope of the variable.

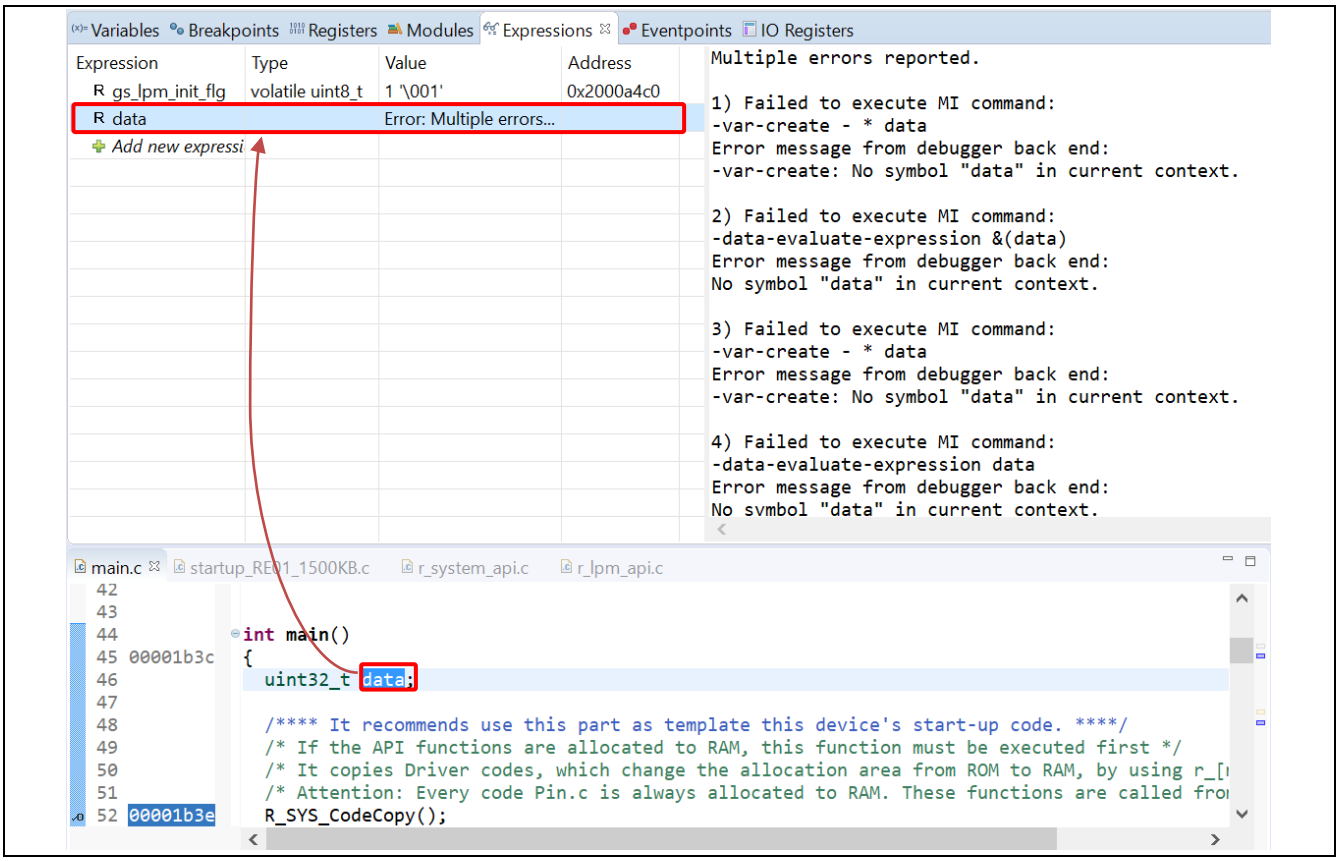


Figure 5-15 Add local variable to Expression view

5.4.3 Registers View

Registers view lists the information about the general registers of the target device.

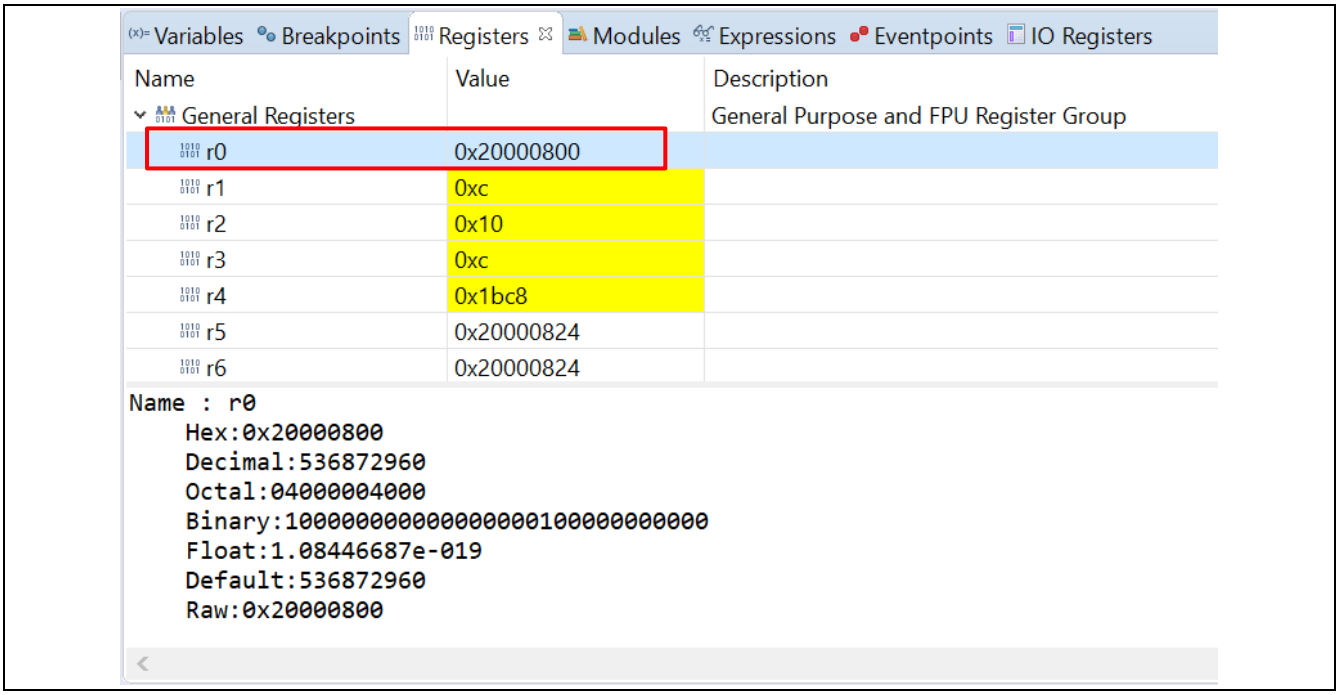


Figure 5-16 [Registers] View

To view the general register “r0”,

1. Click [Window] → [Show View] → [Registers] or icon to open the [Registers] view.
2. Click “r0” to view the values in different radix format.

Values that have been changed are highlighted (e.g. in yellow) in the [Registers] view when the program stops.

5.4.4 Memory View

Memory view allows users to view and edit the memory presented in “memory monitors”. Each monitor represents a section of memory specified by its location called “base address”. The memory data in each memory monitor can be presented in different “memory renderings”, which are the predefined data formats (e.g. Hex integer, signed integer, unsigned integer, ASCII, image, etc.).

To view the memory of a variable (e.g. “gs_lpm_init_flg”),

1. Click [Window] → [Show View] → [Memory] or icon to open the [Memory] view.

2. Click the icon to open [Monitor Memory] dialog box. Enter the address of the variable “gs_lpm_init_flg”.

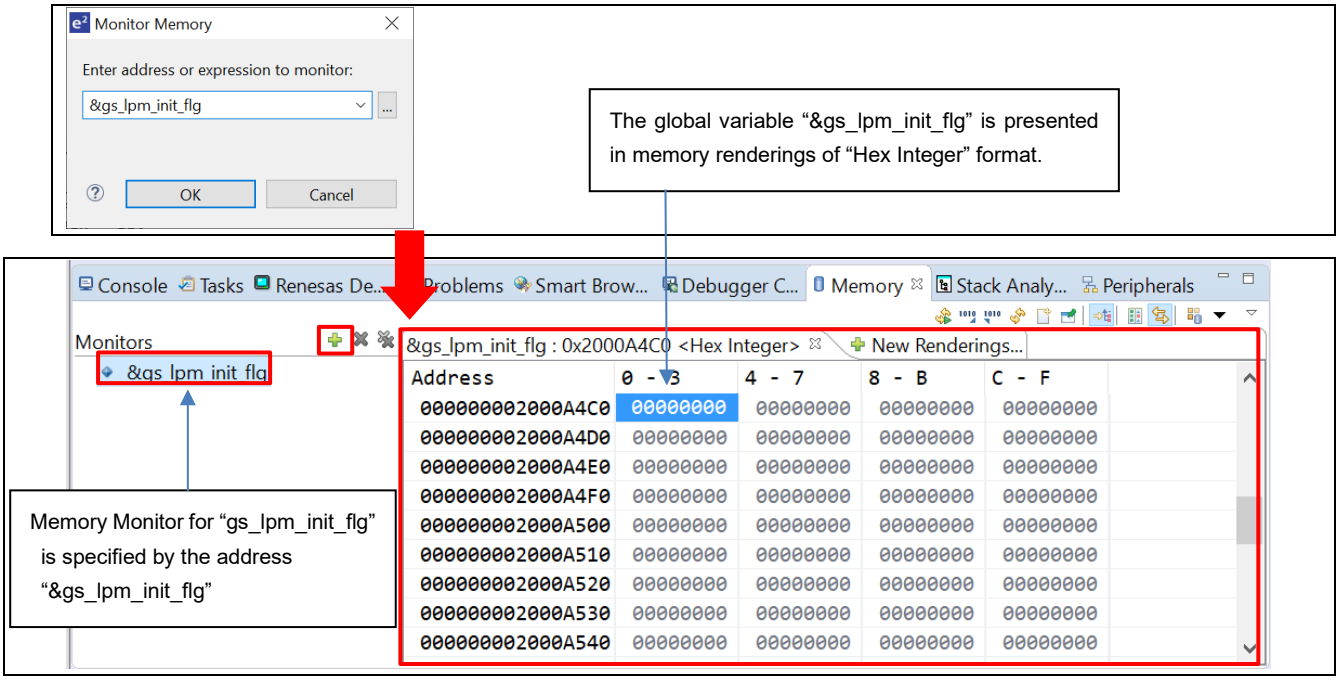
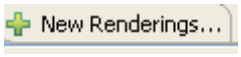


Figure 5-17 [Memory] View (1/2)

To add new renderings format (e.g. Raw Hex) for the variable “gs_lpm_init_flg”,

1. Click the tab 

to select “Floating Point” to add the rendering

   This creates a new tab named “&gs_lpm_init_flg <Floating Point>” next to the tab “&gs_lpm_init_flg <Hex Integer>”.

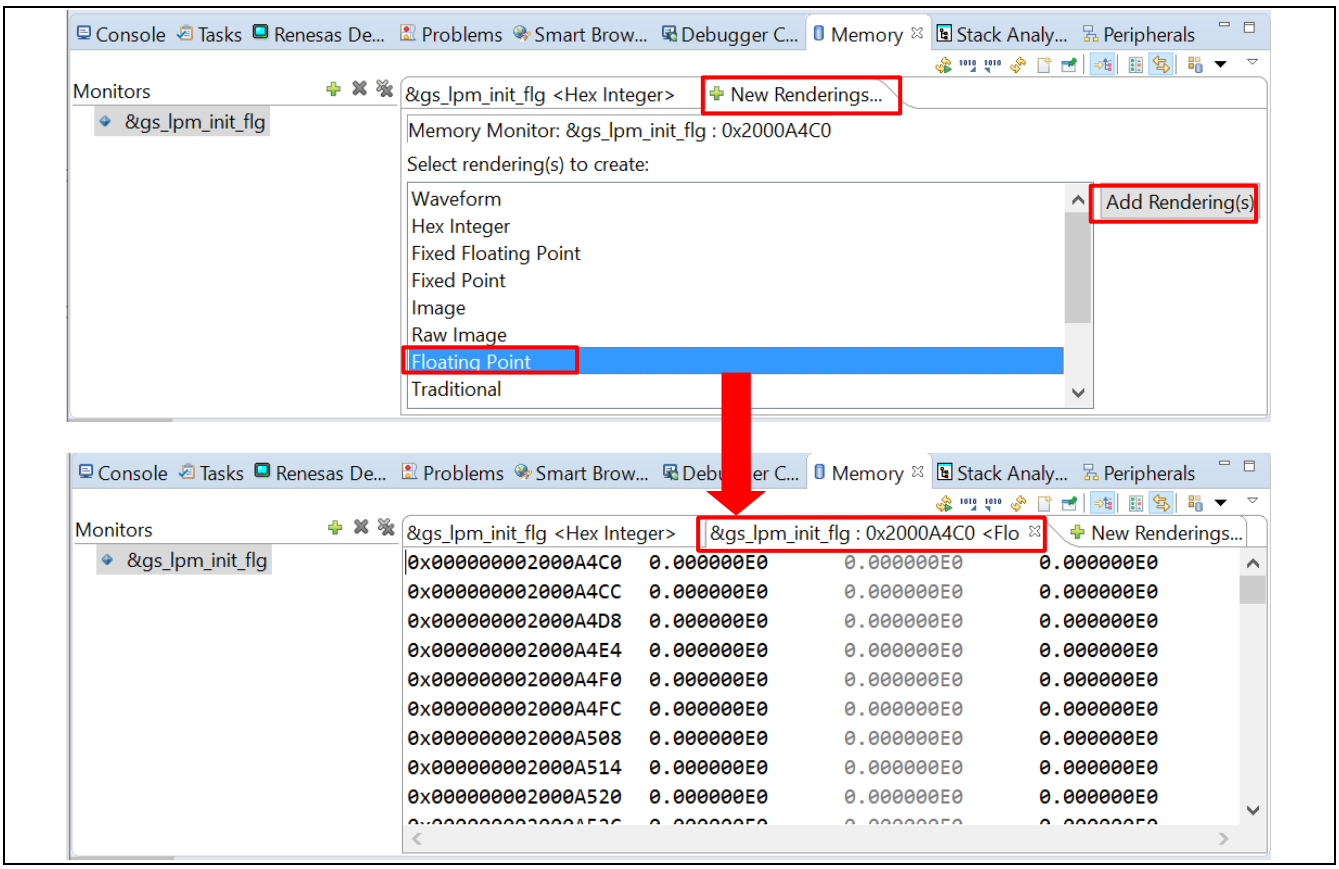


Figure 5-18 [Memory] View (2/2)

5.4.5 Disassembly View

Disassembly view shows the loaded program as assembler instructions mixed with the source code for the comparison. Current executing line is highlighted by an arrow marker in the view. In the [Disassembly] view, user can set breakpoints at the assembler instruction, enable or disable these breakpoints, step through the disassembly instructions and even jump to specific instruction in the program.

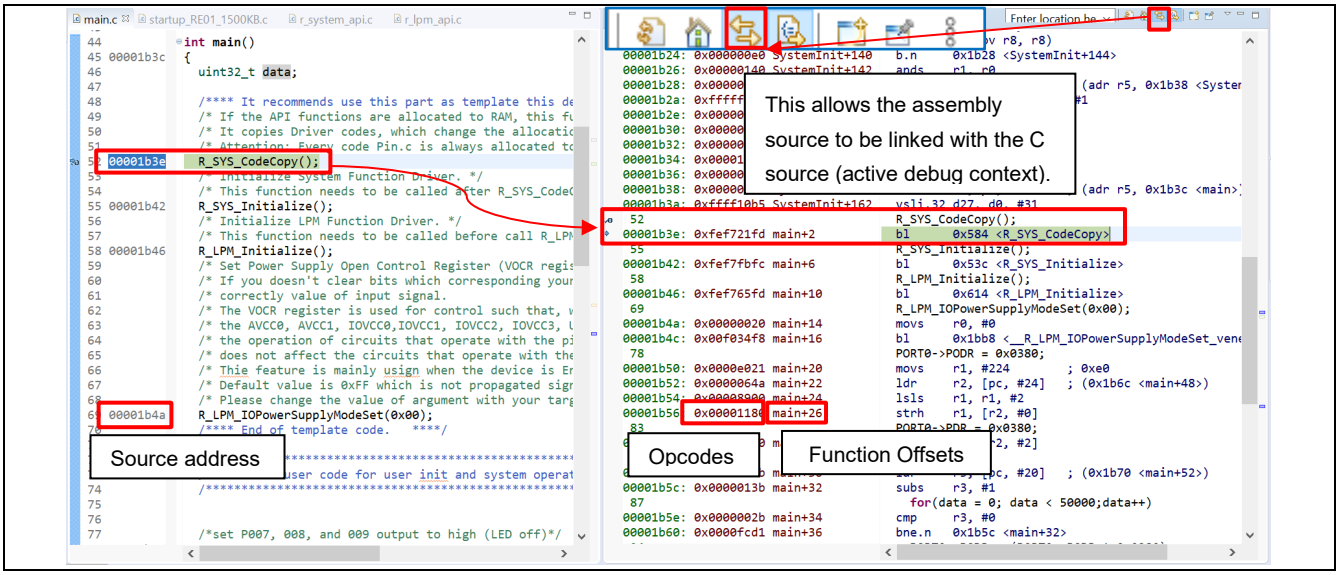


Figure 5-19 [Disassembly] View

To view both C and assembly codes in a mixed mode,

1. Click [Window] → [Show View] → [Disassembly] or icon to open the [Disassembly] view
2. Click icon to enable the synchronization between assembly source and the C source (active debug context).
3. In [Disassembly] view, right-click at the address column to select “Show Opcodes” and “Show Function Offsets”.
4. You can enable source addresses within the editor using the context menu.

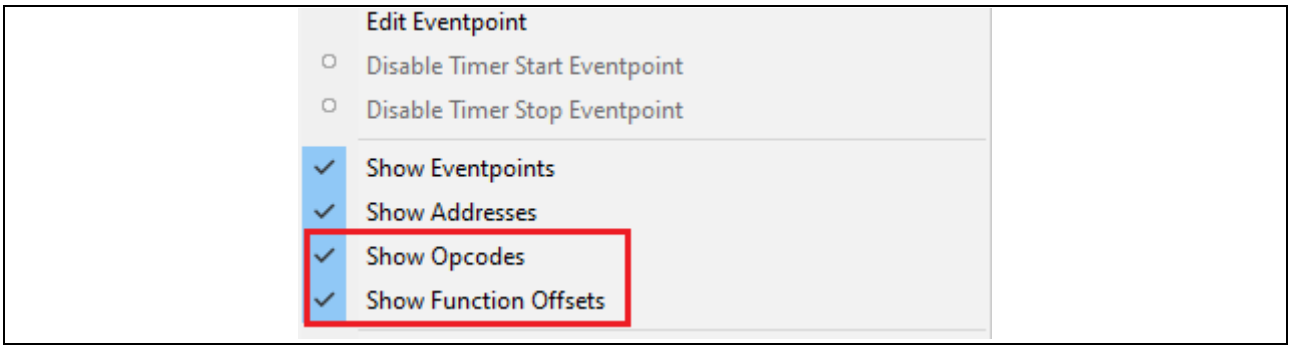


Figure 5-20 Source Addresses Menu

5.4.6 Variables View

Variables view displays all the valid local variables in the current program scope.

Please refer to the ‘Expressions’ view to watch global variables or external variables out of current program scope.

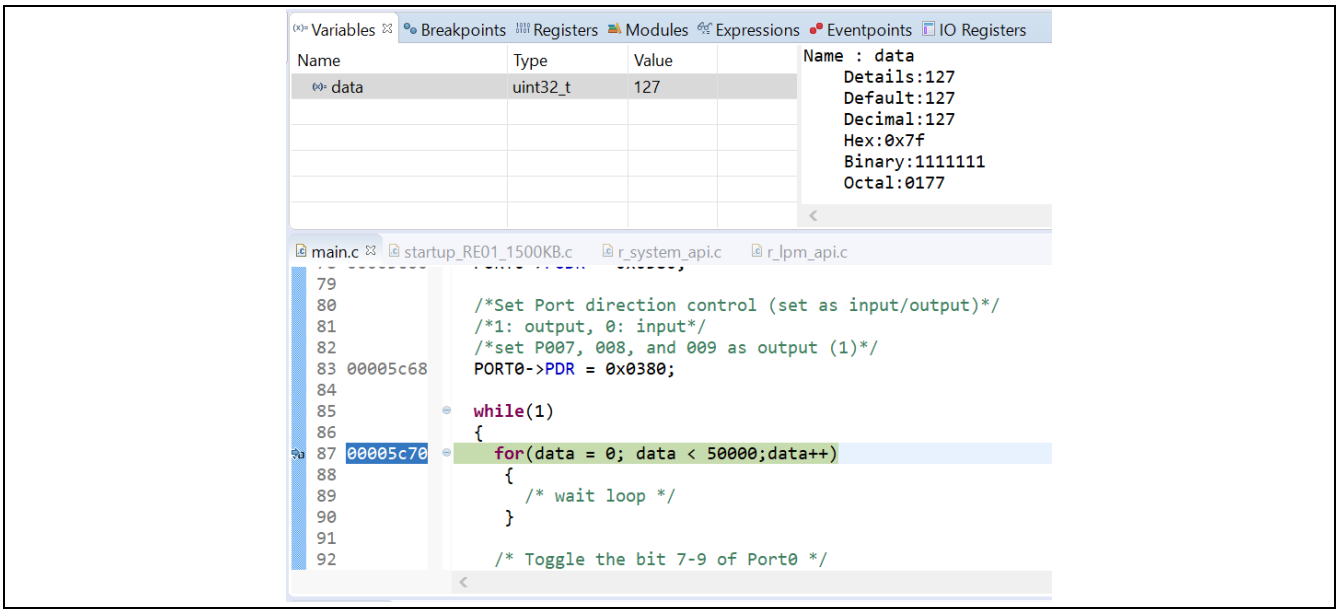


Figure 5-21 [Variables] View

To observe a local variable (e.g. “data” in function “main()”),

1. Click [Window] → [Show View] → [Variables] or icon (x)= to open the [Variables] view.
2. Step into the function “main()” to view the value of local variable “data”.

Note:
The variables which are optimized out or temporarily allocated to accumulator registers may not appear in this view. Please refer to the Disassembly view if necessary.

By disabling optimization, variables will become visible in most cases. However, this will give up all the benefits of optimization such as memory efficiency, code size reduction and performance improvement.

5.4.7 Eventpoints View

An event refers to a combination of conditions set for executing break or trace features during program execution. [Eventpoints] view enables user to set up or view defined events of different category e.g. trace start, trace stop, trace record, event break, before PC, performance (timer) start and performance (timer) stop.

The number of events that can be set and the setting conditions differs with each MCU. These are two (2) types of events:

- Execution address: The emulator detects execution of the instruction at the specified address by the CPU. It can be a “before PC” break (e.g. with event condition is satisfied immediately before execution of the instruction at the specified address) or other events (e.g. with event condition is satisfied immediately after execution of the instruction at the specified address).
- Data access: The emulator detects access under a specified condition to specified address or specified address range. This allows to setup complex address and data matching criteria.

Event combination (e.g. OR, AND (cumulative) and Sequential) can be applied to two (2) or more events.

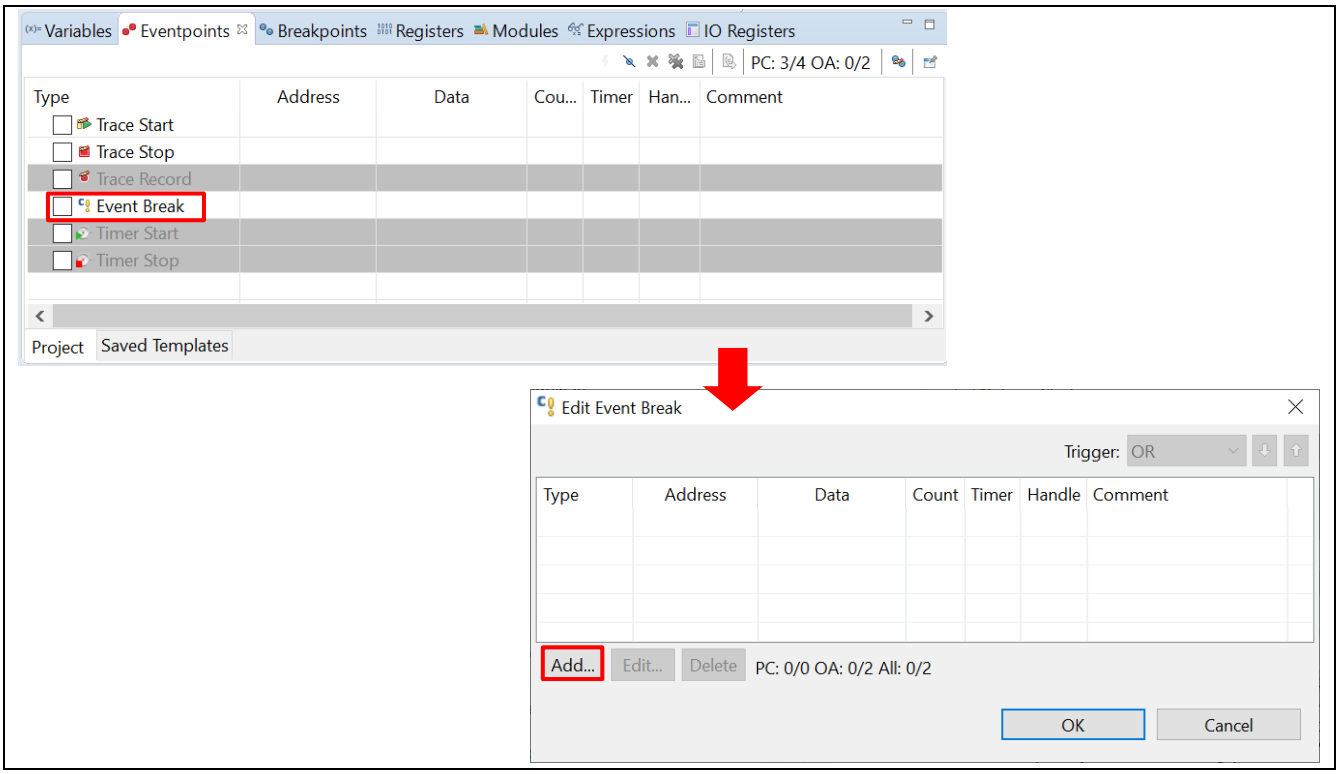


Figure 5-22 [Eventpoints] View (1/2)

To set an event break for a global variable when address/data is matched (e.g. when gs_lpm_init_flg is written),

1. Click [Window] → [Show View] → [Eventpoints] or icon to open the [Eventpoints] view.
2. Double-click on the “Event Break” option to open [Edit Event Break] dialog box
3. Click [Add…] button to continue.

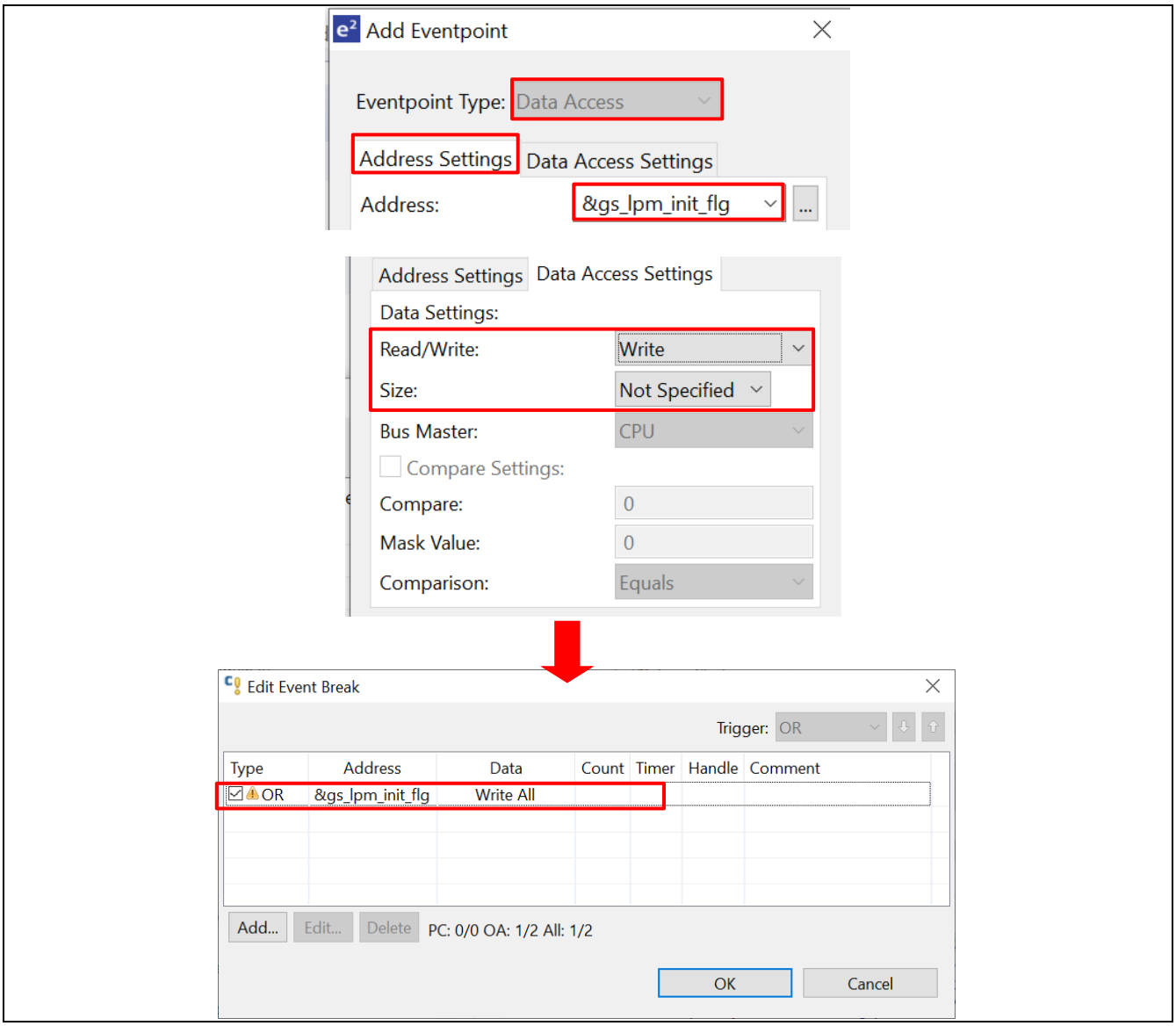


Figure 5-23 [Eventpoints] View (2/2)

4. Select “Data Access” as the eventpoint type.
5. Go to the [Address Settings] tab, click the icon to browse for the symbol “gs_lpm_init_flg”. (The address of this global variable is “&gs_lpm_init_flg”)
6. Next, switch to the [Data Access Settings] tab, select “Write” from “Read/Write” dropdown list. Click [OK] to proceed.

7. Ensure that the event break for “gs_lpm_init_flg Write All” is set and enabled in the [Eventpoints] view. Reset to execute the program from the start.

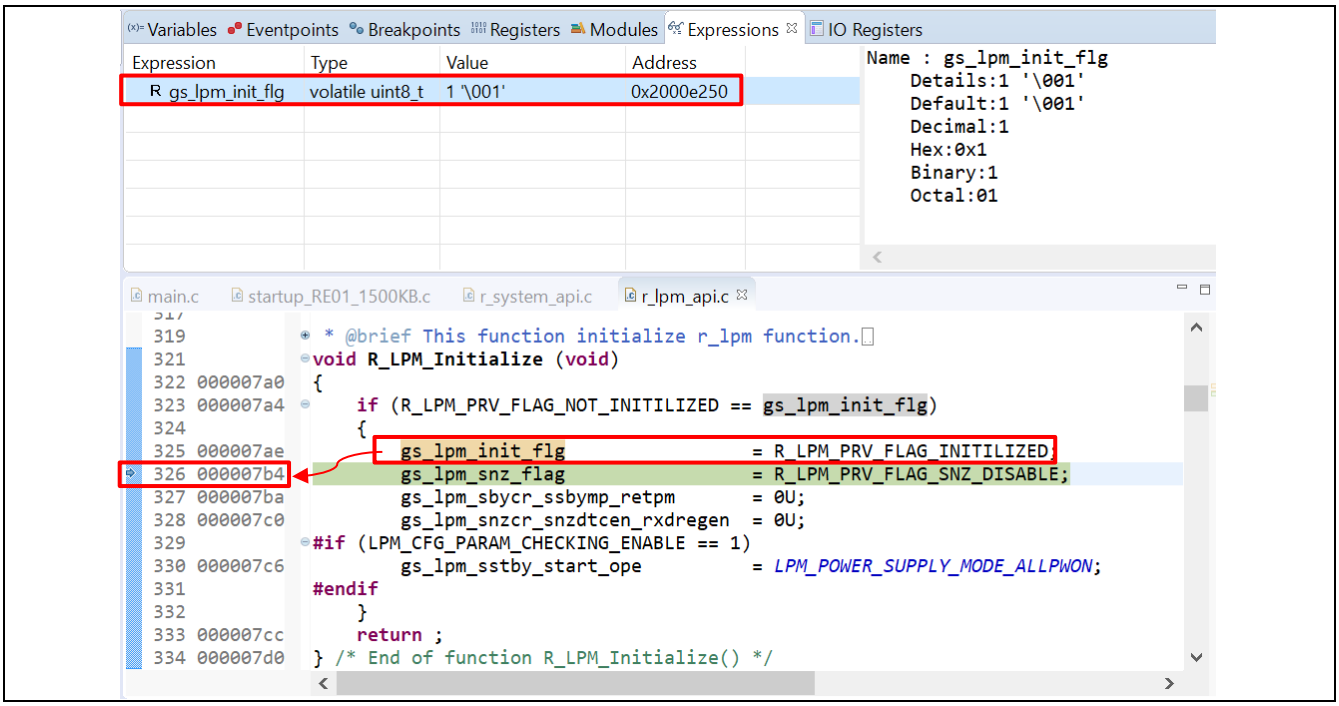


Figure 5-24 Execution of Event Break

Figure 5-24 shows that when gs_lpm_init_flg is assigned with value R_LPM_PRV_FLAG_INITILIZED, the program stops at code line No.326 (right after the line of code writing to gs_lpm_init_flg).

5.4.8 IO Registers View

IO Registers are also known as the Special Function Registers (SFR). The [IO Register] view displays all the registers set defined in a target-specific IO file, including their address, hex and binary value. User can further customize their own [IO registers] view by adding IO registers selectively to the [Selected Registers] pane.

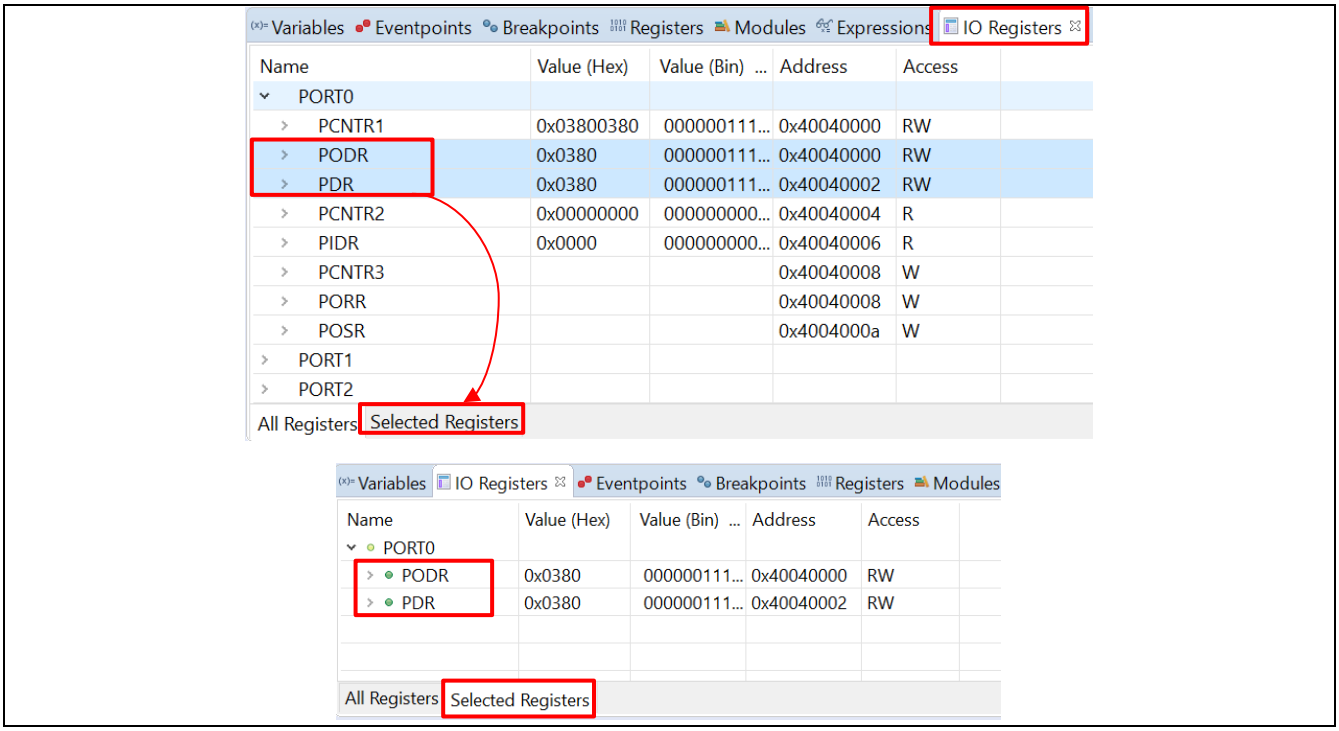


Figure 5-25 [IO Registers] View

To view selected IO registers (e.g. PODR and PDR in PORT0),

1. Click [Windows] → [Show View] → [Others…]. In “Show View” dialog, click [IO Registers] under [Debug] or icon to open the [IO Registers] view

2. Under the [All Registers] tab, locate [PORT0] in the [IO Registers] view. Expand the PORT0 IO register list. You could also use Search button in the IO Register toolbar to quickly search by name.

3. Drag and drop the “PODR” and “PDR” to the [Selected Registers] pane. A green dot beside the IO register indicates the status of being the selected register(s).

4. Switch to the [Selected Registers] tab to view “PODR” and “PCR” of the “PORT0” IO register.

The expanded IO register list may take a longer time to load in the [All Registers] pane. Hence, it is advisable to customize and view multiple selected IO registers from the [Selected Registers] pane.

5.4.9 Trace View

Tracing means the acquisition of bus information per cycle from the trace memory during user program execution. The acquired trace information is displayed in the [Trace] view. This helps users to track the program execution flow to search for and examine the points in the program where problems arise.

The trace buffer is limited, the oldest trace data is overwritten with the new data after the buffer has become full.

To set a trace until the program is suspended, users can do as following:

1. Click [Renesas Views] → [Debug] → [Trace] or icon to open the [Trace] view.
2. Turn on the Trace view by selecting the icon.

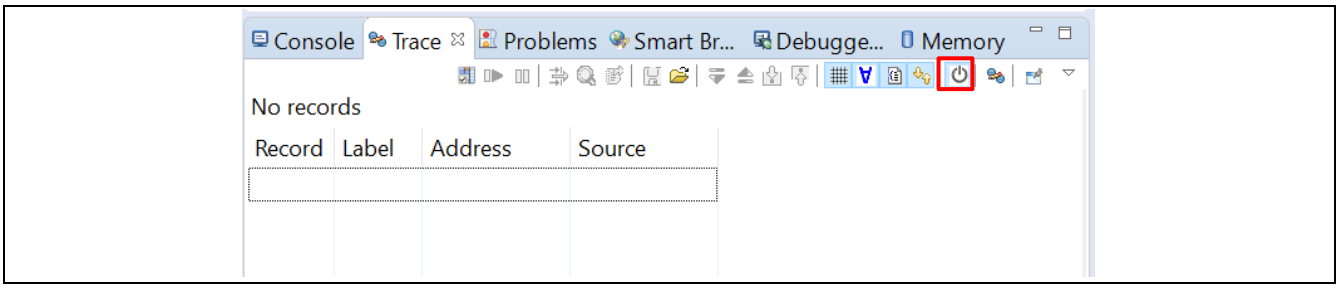


Figure 5-26 Turn on Trace view

3. Execute the program and stop program execution by using a breakpoint or by pressing the [Suspend] button on the Debug Toolbar. The content stored in trace memory at that point in time is displayed as trace result.

4. Select the display mode by clicking on the corresponding button.

The below figure shows the trace result before the main() function is executed.

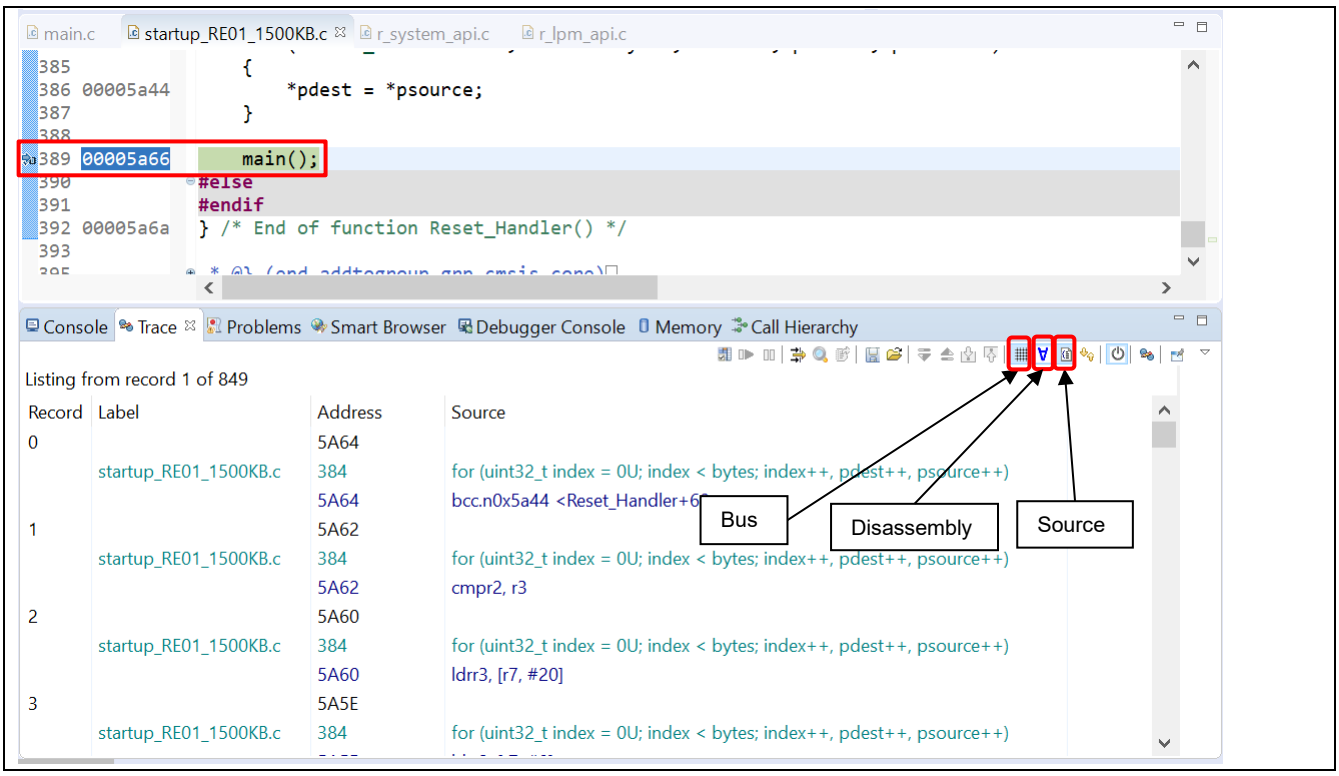


Figure 5-27 Select display mode in Trace view

5. The trace records are displayed from oldest data to latest data by default. The display order can be changed by clicking button.

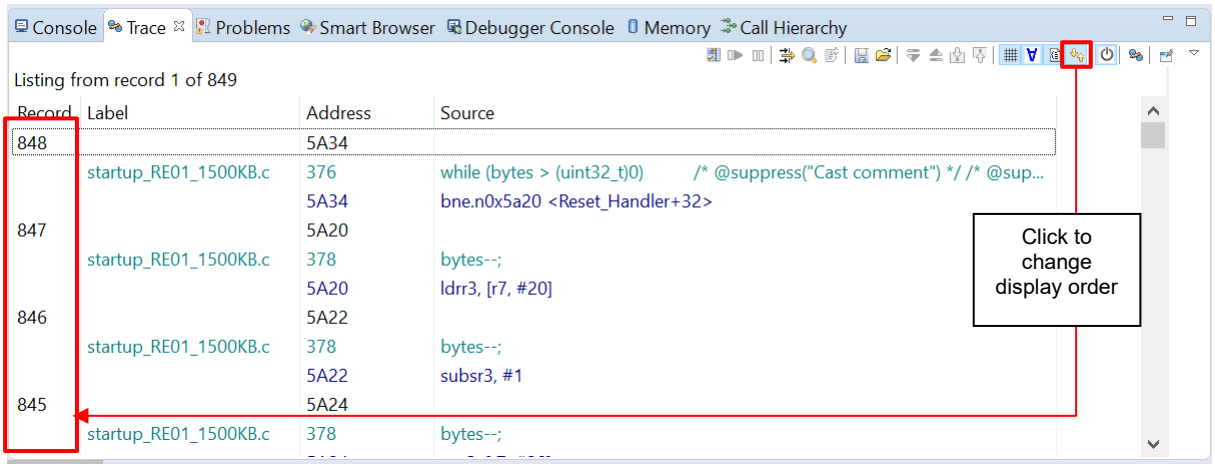


Figure 5-28 The display order is changed

6. The trace result can be filtered by clicking on button. Users can select to filter by “Record” and/or “Address”.

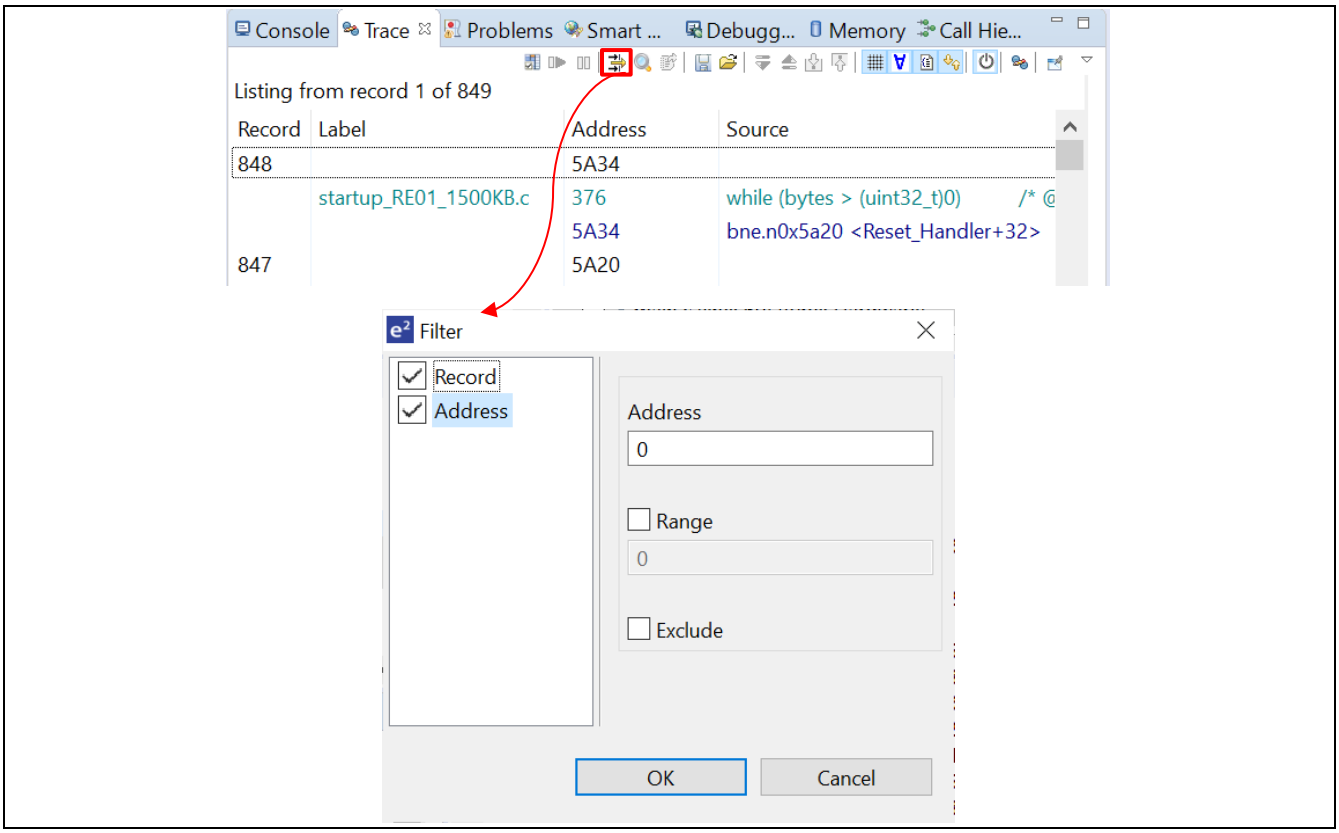


Figure 5-29 Filter trace result

7. Trace result can be saved to a .csv file (with the inclusion of bus, assembly and source information). Trace view also allows to load trace results from a .csv file.

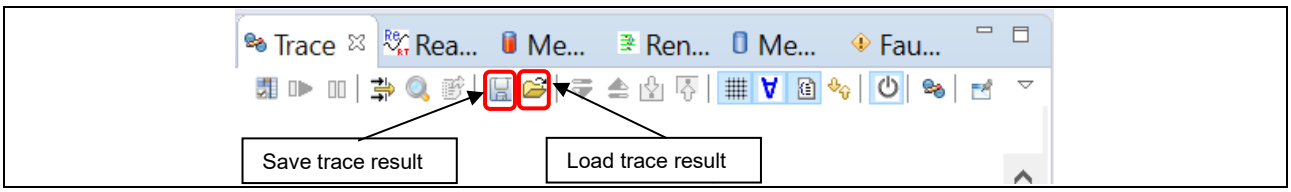


Figure 5-30 Save and load trace result

5.4.10 Memory Usage View

Memory view allows users to view the total memory size, usage of ROM and RAM ratio and detailed information of sections, objects, symbols, module, vector and cross-reference used in the project.

To view the memory usage of a project,

1. Click [Window] → [Show View] → [Other…] → [Debug] → [Memory Usage] to open the memory usage view.

2. The default display of the Memory Usage view for executable project (which uses GNU ARM Embedded Toolchain) has 3 regions: (1) Group size region, (2) Memory Region Usage/ Device Memory Usage region, and (3) Detailed table region.

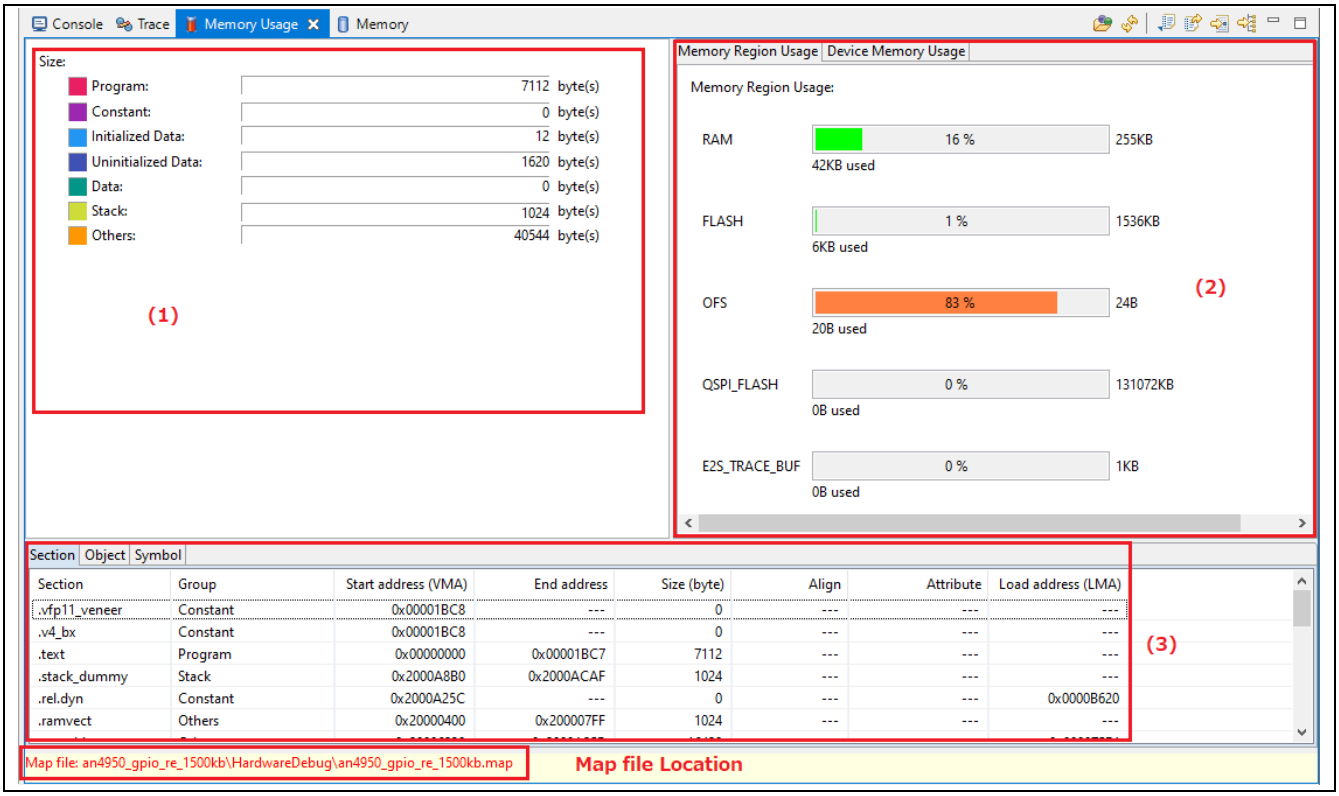


Figure 5-31 Memory Usage view

(1) Group size region (for executable project):

Displays the total size of Program, Constants, Initialized Data, Uninitialized Data, Data, Stack and Other according to the selected map file.

Note: This view only displays for executable project of supported toolchains.

(2) Memory Region Usage and Device Memory Usage region:

Memory Region Usage shows percentage of RAM/ROM/flash usage by numerical value and status of bar. Color of the bar is based on the percentage value.

- If percentage < 75%: Green.
- If percentage >= 75% and percentage < 90%: Orange.
- If percentage >= 90%: Red.

Device Memory Usage region shows device memory of the selected project's device. Each memory area shows the name, start address, end address, used size and size.

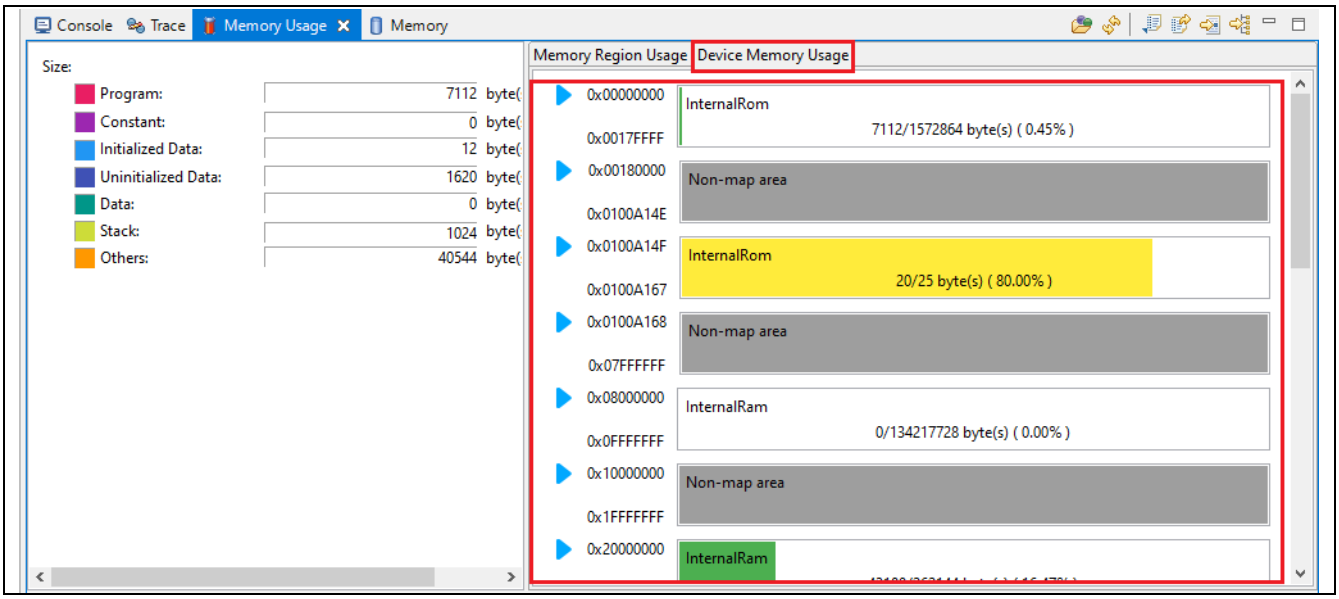


Figure 5-32 Device memory usage region

Expand memory area to see section inside. Color of sections corresponds with that of Group Size region.

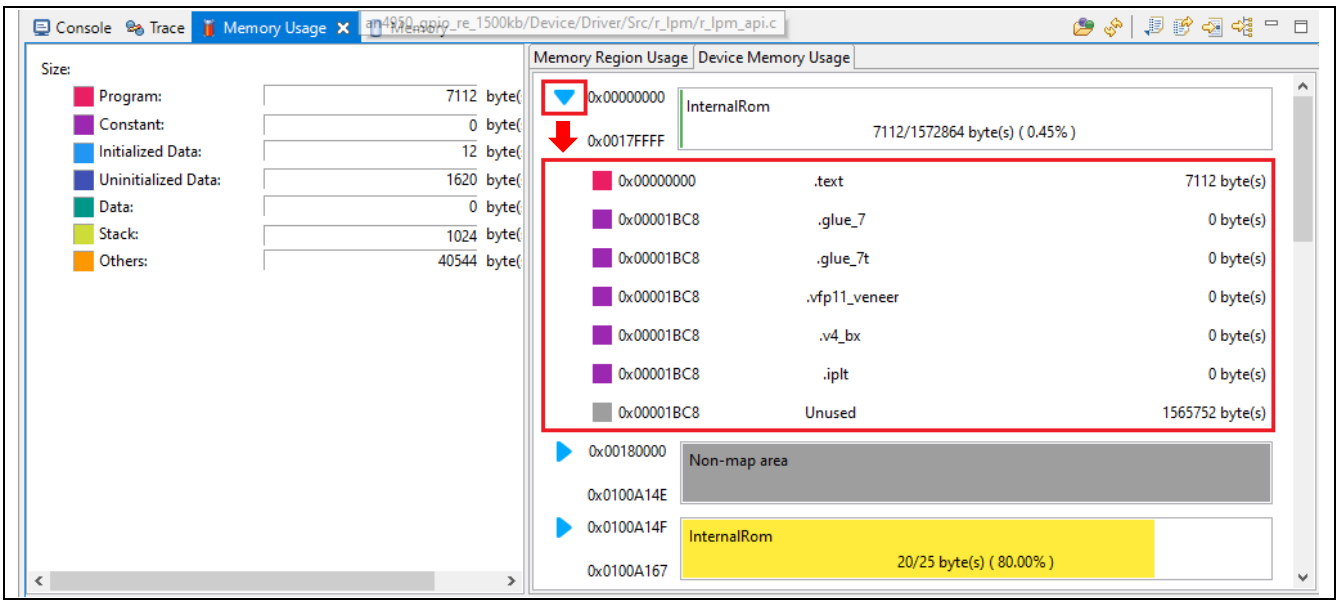


Figure 5-33 Expand memory area

(3) Detail table region:

Display the map file information of an active project or the opened map file.

- “Section” tab: contains “Linkage map” table which displays the list of Sections analyzed from map file and its detailed information.
- “Object” tab: contains “Object” table which displays the list of Objects analyzed from map file and its detailed information.
- “Symbol” tab: contains “Symbol” table which displays the list of Symbols analyzed from map file and its detailed information.
- “Cross Reference” tab: displays the cross-reference information that is retrieved from map file. This tab is only available for executable project.

Map file location:

Memory Usage will display the information of (*.map) file or library list file (*.lbp) from the project. The user can see the relative path of the selected map file or library list file at the bottom of Memory Usage view.

6.Help

The help system allows users to browse, search, bookmark and print help documentation from a separate Help window or Help view within the workbench. Users can also access online forum dedicated to e² studio from here.

Click on [Help] tap to pull down the Help menu.

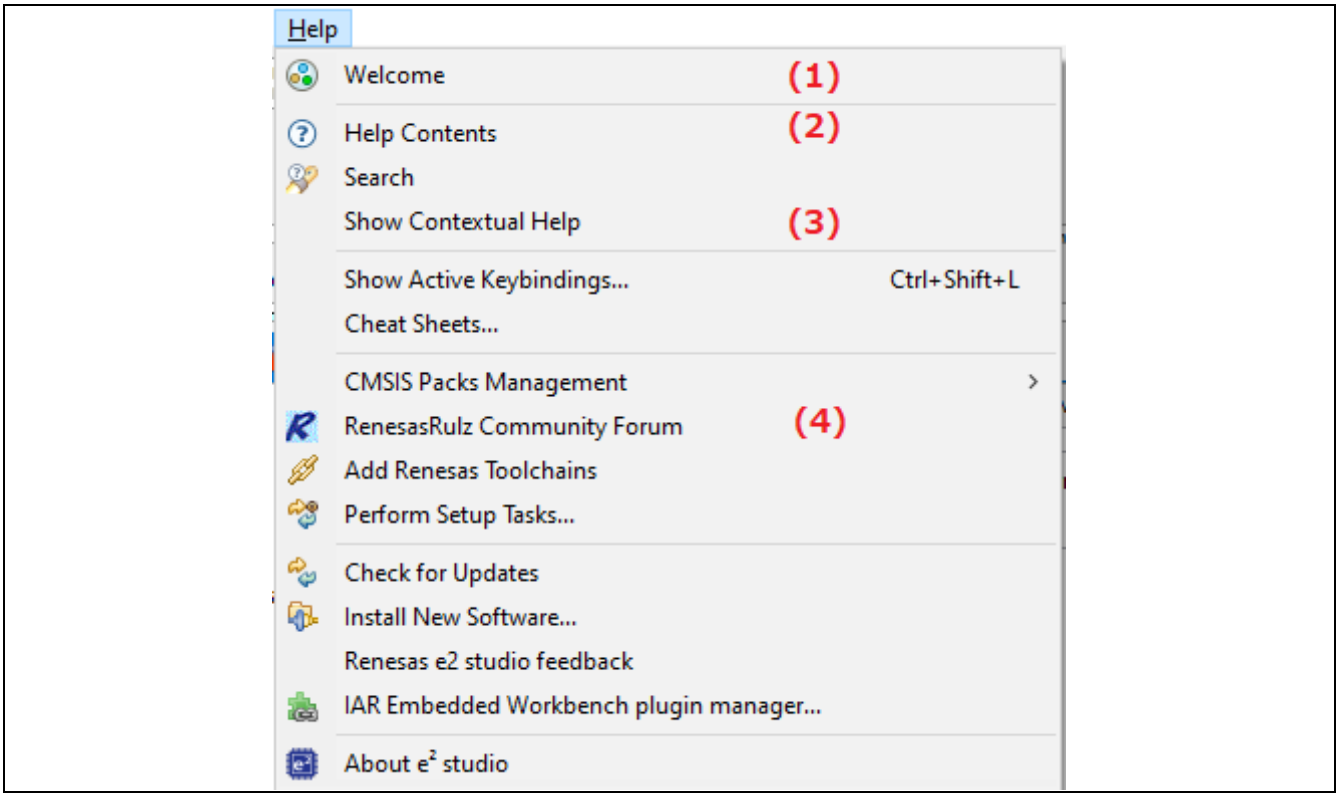


Figure 6-1 Help Menu

Quick Help Tips

(1) Click [Welcome] for Overview of e² studio, link to access IDE tutorial and sample, and to view Release Notes.

(2) Click [Help Contents] to open a separate Help window with search function.

(3) Click [Show Contextual Help] to open Help view inside the workbench.

(4) Click [RenesasRulz Community Forum] to go online forum that is dedicated to topics and discussions related to e² studio IDE. Internet connection is required.

Revision History	Renesas e^2 studio 2021-07 or Higher User's Manual : Quick Start Guide

Rev.	Date	Description	
		Page	Summary
1.00	Jul.20.2021	-	Initial release.

Renesas e^2 studio 2021-07 or higher
User's Manual: Quick Start Guide

Publication Date: Rev.1.00 Jul 20, 2021

Published by: Renesas Electronics Corporation

Renesas e^2 studio 2021-07 or Higher

User's Manual: Quick Start Guide

Renesas Electronics Corporation

R20UT5034EJ0100