User's Manual

Renesas e^2 studio v2021-10 or higher
User's Manual: Quick Start Guide

Renesas Synergy™

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.1.10 Apr 2022

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

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/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

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

Renesas e² studio is the Integrated Development Environment for Renesas Synergy™ 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.

The e² studio ISDE (Integrated Solution Development Environment) provides support for the Renesas Synergy™ Software Package (SSP), including Frameworks, Hardware Abstraction Layer (HAL) drivers, and Board Support Package (BSP) drivers for Renesas Synergy™ projects. The SSP provides a complete driver library for developing Renesas Synergy™ applications in the e² studio.

The e² studio IDE includes multiple Graphical User Interface (GUI) wizards for auto-generating code, including configuring existing drivers, configuring build and debug options, and running the applications you create. Driver documentation is integrated in the form of tooltips, which are available in the code editor view.

Renesas Synergy™ support is included in release 4.1 and higher of the Renesas e² studio. Multiple views and editors are available to specifically support Renesas Synergy™ ARM® Cortex®-M-based microcontrollers and the open-source GNU ARM toolchain.

The Renesas Synergy™-specific add-ons provide easy-to-navigate wizards for configuring hardware and for managing the extensive Renesas Synergy™ software library.

Note: “Azure RTOS ThreadX” is described in ThreadX. “Azure RTOS TraceX” is described in TraceX.

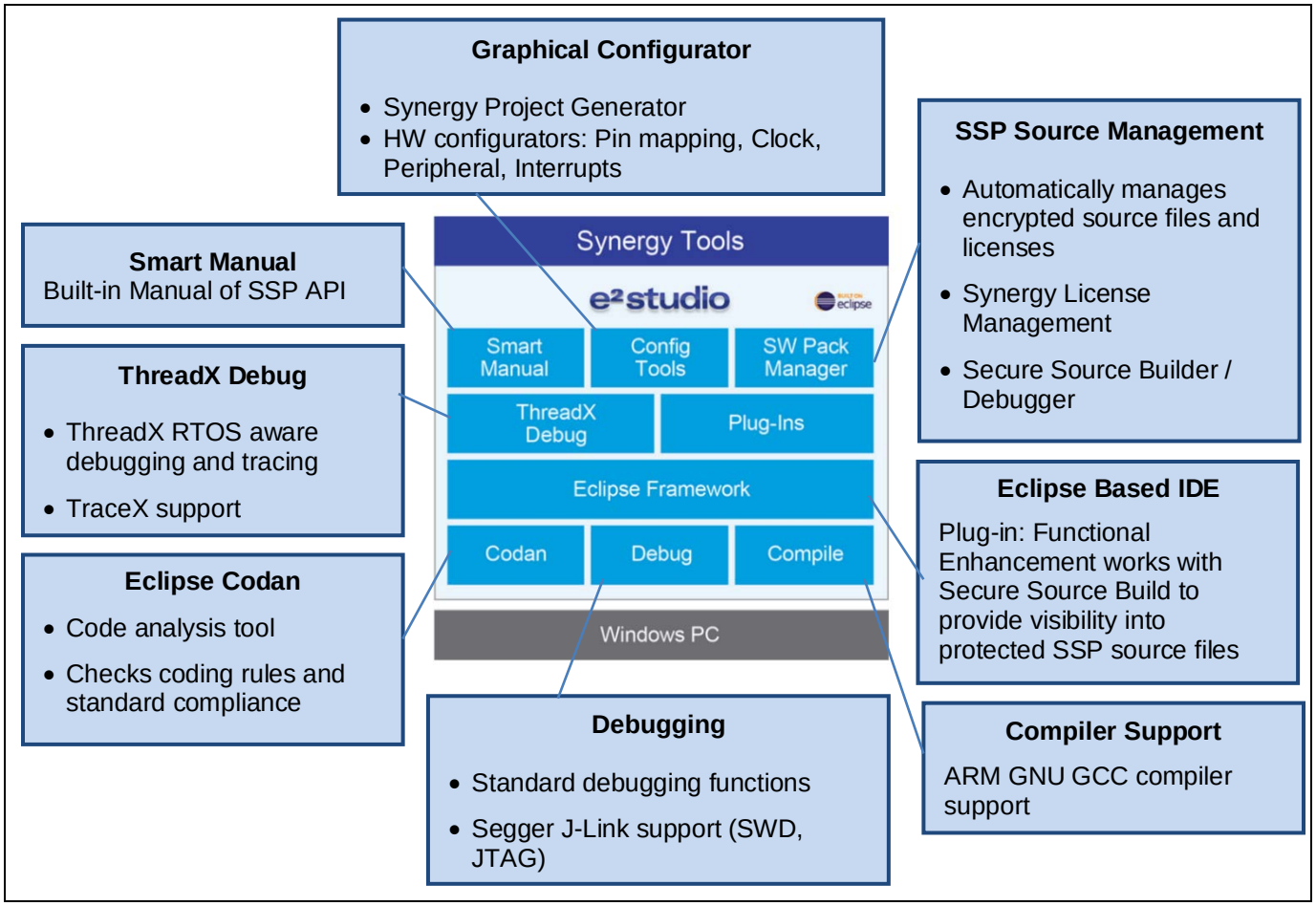


Figure 1-1. Renesas Synergy™ in e² studio

Most e² studio features are common to all supported Renesas product lines. Specific to Synergy products are the GCC ARM Embedded and IAR for ARM toolchains support, the Renesas Synergy™ Project Generator, and the Renesas Synergy™ Configuration Editor.

Table 1-1. e² studio Features Comparison

Feature	Renesas Synergy™	RX / RL78 / RZ / RH
IDE framework and C/C++ support	Eclipse + CDT	Eclipse + CDT
Code-generating tools	Synergy Project Generator Synergy Configuration Editor	Code Generator/Smart Configurator
Toolchain	GCC ARM Embedded IAR toolchain for ARM	RX family (GNURX-ELF, Renesas CC-RX and IAR build plug-ins) RL family (GNURL78-ELF, Renesas CCRL and IAR build plug-ins) RZ family (GNUARM-NONE-EABI)
HEW/CS+ project import	Not Supported	Supported for MCUs supporting HEW/CS+ IDE
Target Debuggers	Segger J-Link	E1, E2, E2 Lite, E20, IECUBE, E10A-USB, Segger J-Link
Smart Manual tooltips	Supported (for SSP API)	Supported
Code Analysis (CODAN)	Supported	Supported
Simulator	Not Supported	Supported for selected RX and RL family devices
Debugger	GDB with trace and real-time memory access	GDB with trace and real-time memory access
RTOS	Azure RTOS ThreadX	Various operating systems
ThreadX Configuration	Supported	Not Supported
ThreadX Debug	Supported	Not Supported
Memory Usage view	Supported	Supported
Visual Expressions view	Supported	Supported

1.1 System Configuration

A typical system configuration includes a host machine and a target board as shown below.

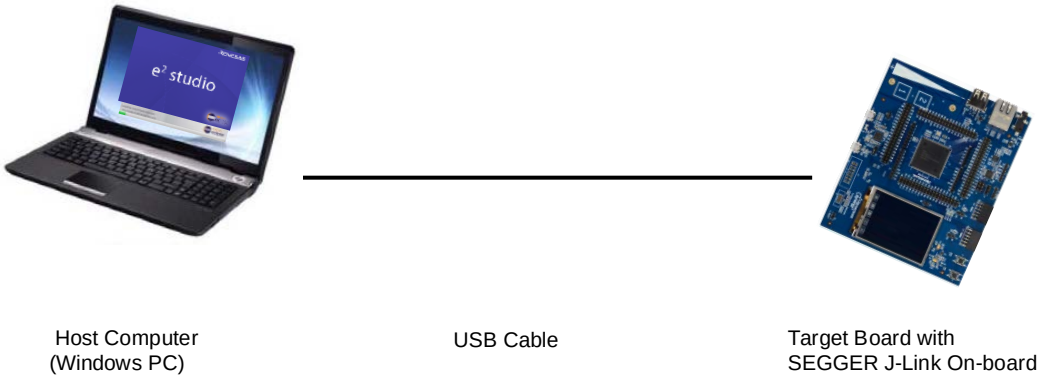


Figure 1-2. System configuration

1.2 System Requirements

- Host Computer:
  - Processor: At least 2GHz (with Intel® Core™ family processor)
  - Memory capacity: At least 4GB (8GB or larger is recommended)
  - Hard disk capacity: Minimum 2 GB
  - 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.
- Operating System:
  The following operating systems on the host computer are supported:
  Windows 8.1 (64-bit OS) and Windows 10 (64-bit OS).

1.3 Supported Toolchains

GNU ARM® compiler (version: GCC v9.2.1 and GCC v7.2.1)

Note: IAR Embedded Workbench for Synergy is no longer available for download from renesas.com. Contact IAR for access to the Embedded Workbench for Synergy. Licenses for EWSYN are issued by IAR, users can find additional information on licensing Embedded Workbench for Synergy and request a new or updated license key from IAR.

1.4 Supported Emulator Device

Segger J-Link.

1.5 Outline of a Synergy Project Development

This document provides detailed instructions on how to start developing with Renesas Synergy™. The main steps are outlined below. By understanding the main steps below, readers can relate better to the procedures described in Chapter 3 and Chapter 4.

1. Generating a Synergy project
2. Configuring the Synergy project to fit hardware specifications such as clock, ICU, pin functions
3. Configuring the Azure RTOS ThreadX
4. Configuring the BSP (selecting HAL driver models)
5. Adding user code

6. Building the project
7. Configuring the debugger and launching debugging

2. Installation

The development tools can be installed using either the Platform Installer or Standalone Installer.

The latest version of installer package can be downloaded from the Solutions Gallery of the Synergy Platform website https://www.renesas.com/products/synergy.html.

2.1 Installing with the Platform Installer

The Platform Installer includes the Synergy Software Package (SSP), e² studio ISDE, GCC ARM embedded compiler and J-Link Drivers. To download and install the Platform Installer, follow the steps below:

1. Visit the Solutions Gallery of the Synergy Platform website. Select **Download SSP** under **Synergy Software-> Synergy Software Package (SSP)** to go to the SSP page. Select **Download** under **Synergy Download** to log in to your My Renesas account and then download the platform installer.

Figure 2-1. Installation – Download the Platform Installer

2. Select e² studio as the development environment.

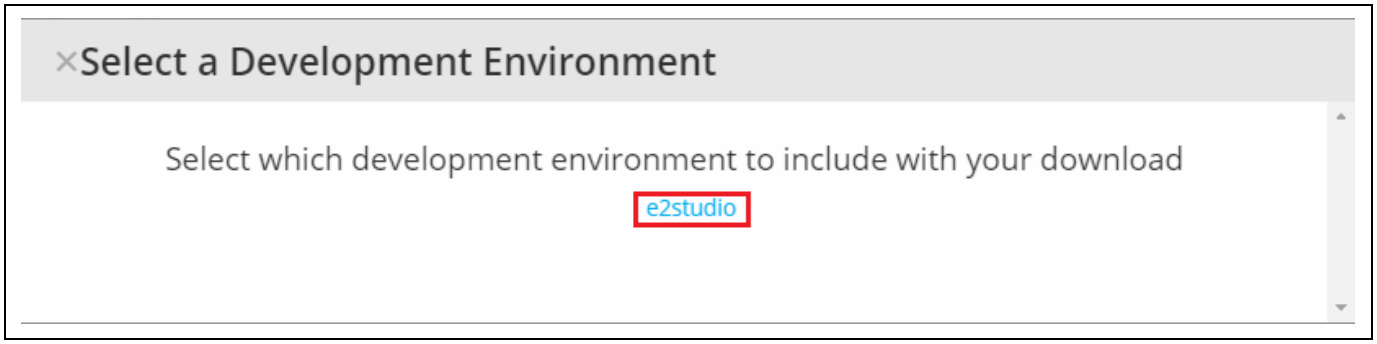


Figure 2-2. Installation – Select e² studio Development Environment

Click **I AGREE** in the License Agreement, and the installation file (for example, `setup_ssp<version>_e2s_<version >.zip`) will be downloaded.

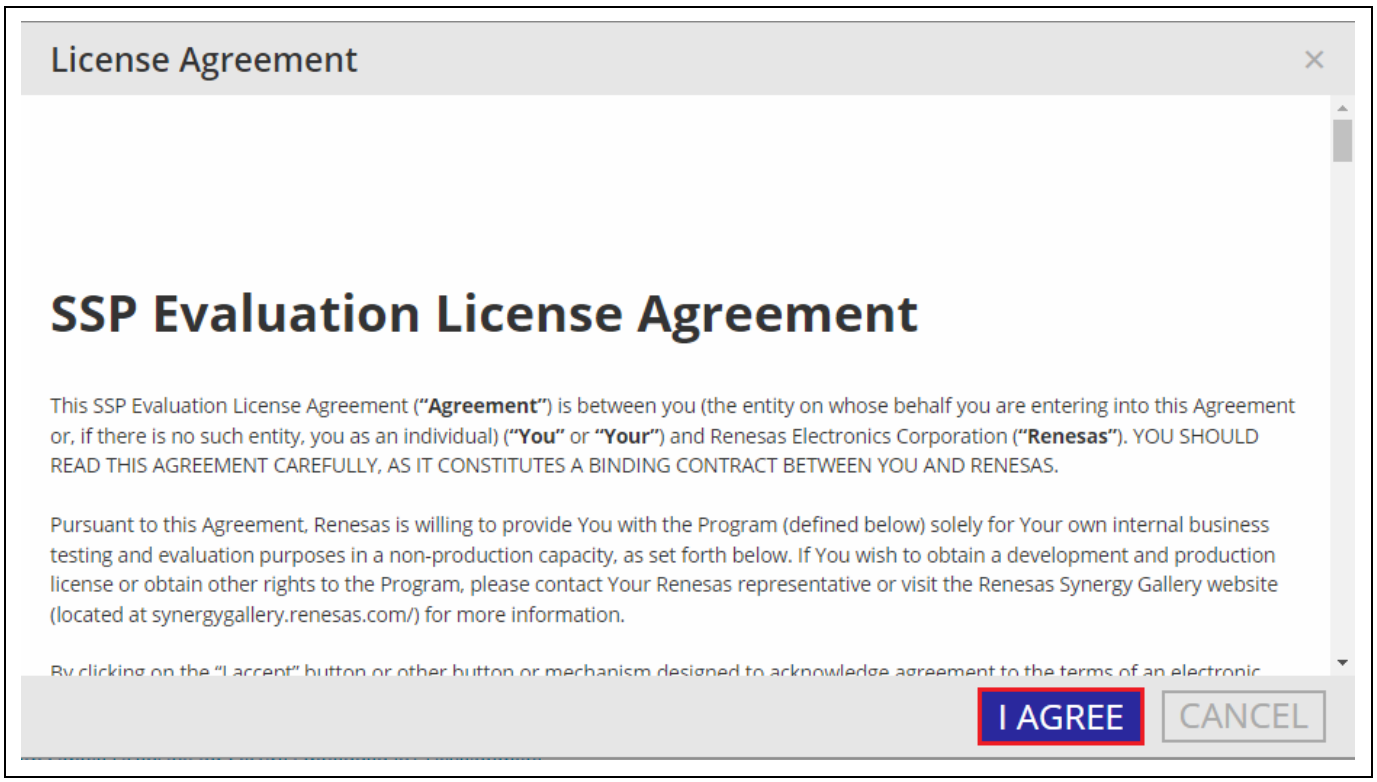


Figure 2-3. Installation – Accept the License Agreement

3. Unzip and run the installation file.
4. In the Select Install Type page, if users would like to customize the components to be installed, choose "Custom Install" then click Next.

We recommend that new users select the **Quick Install** option to minimize the configuration steps. This option will install e² studio, SSP and GCC ARM Embedded by default. If a user selects **Quick Install**, step (1) will not be shown.

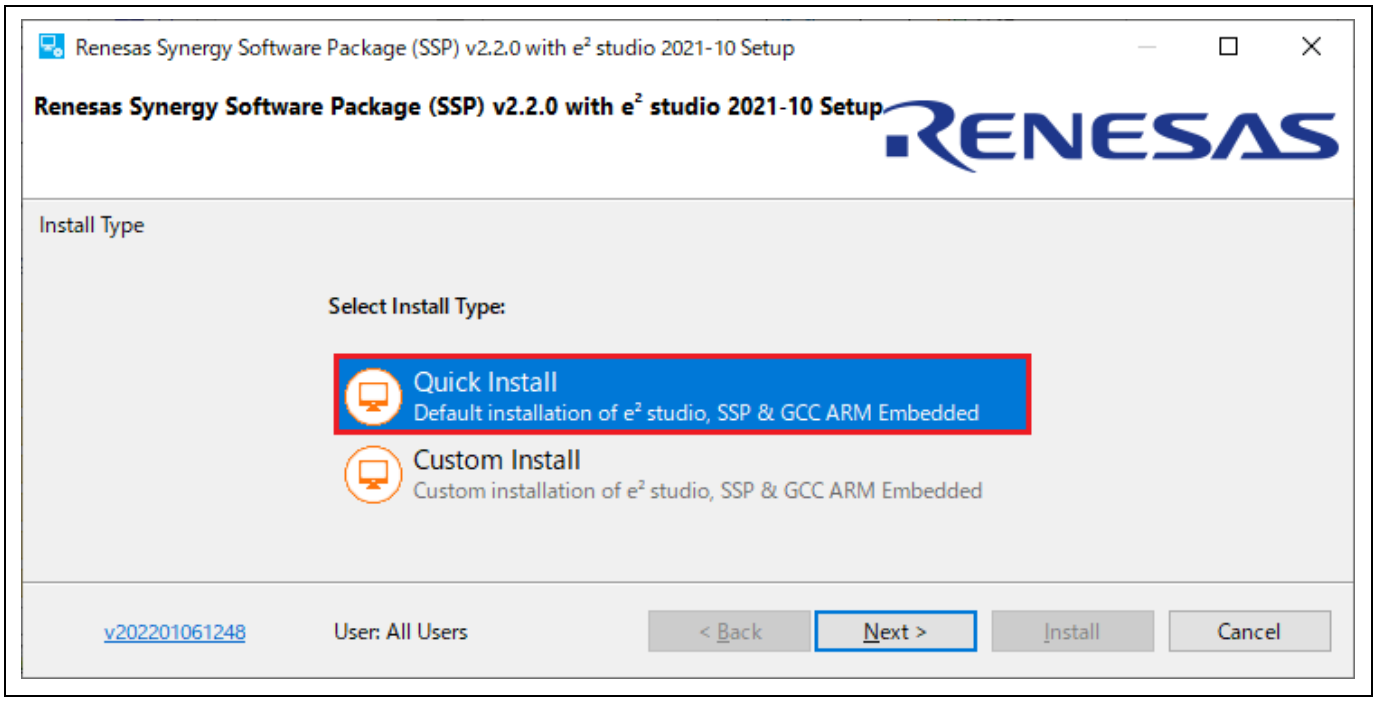


Figure 2-4. Installation – Select Install Type

1. In the welcome page, you may use the default folder or change it by clicking **Change…**. Click **Next** to continue.

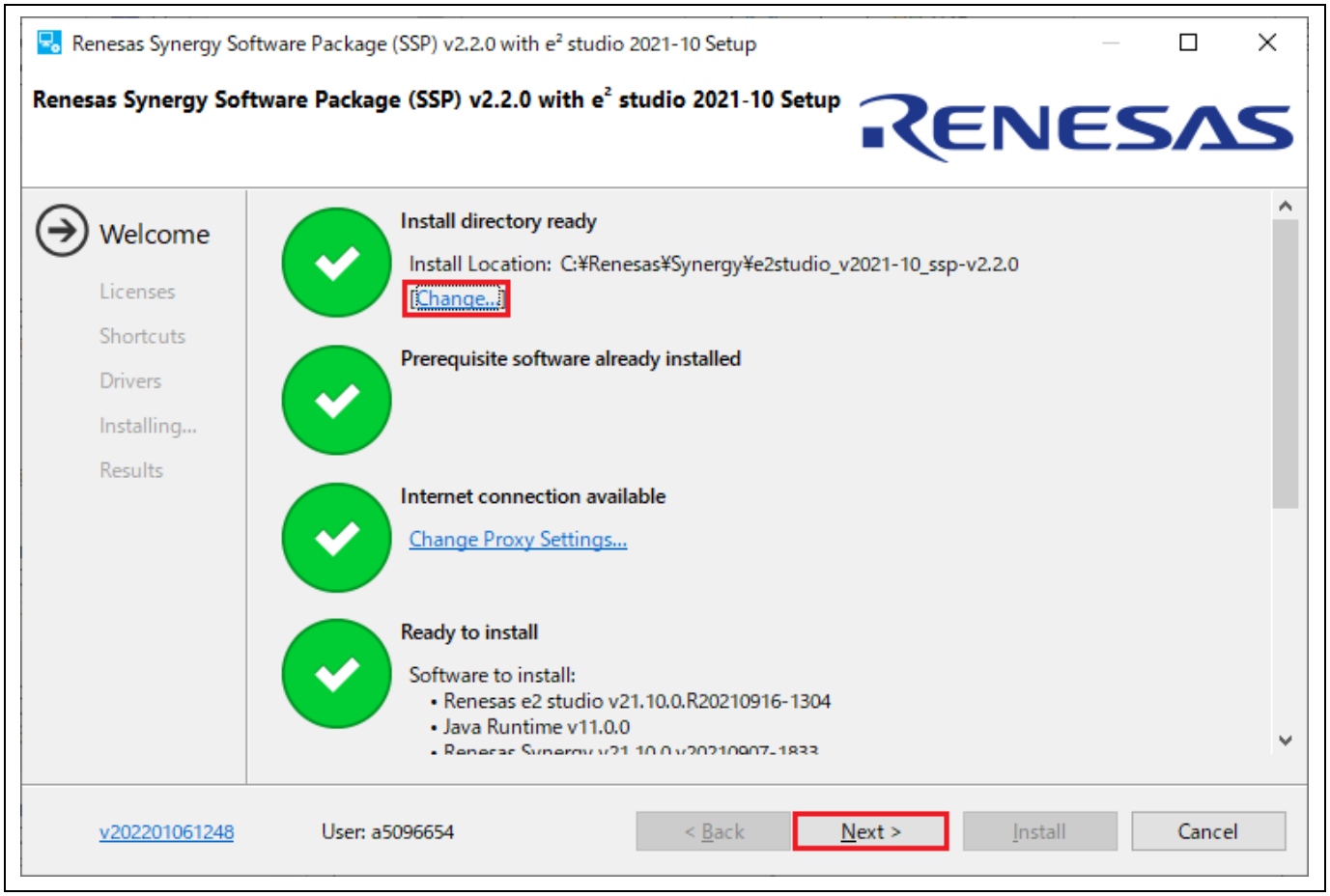


Figure 2-5. Installation – Welcome Page

2. If you specify a custom installation, configure the settings from Figure 2-11 to Figure 2-13
3. Tick the checkbox to accept the license agreement, and click **Install** to continue.

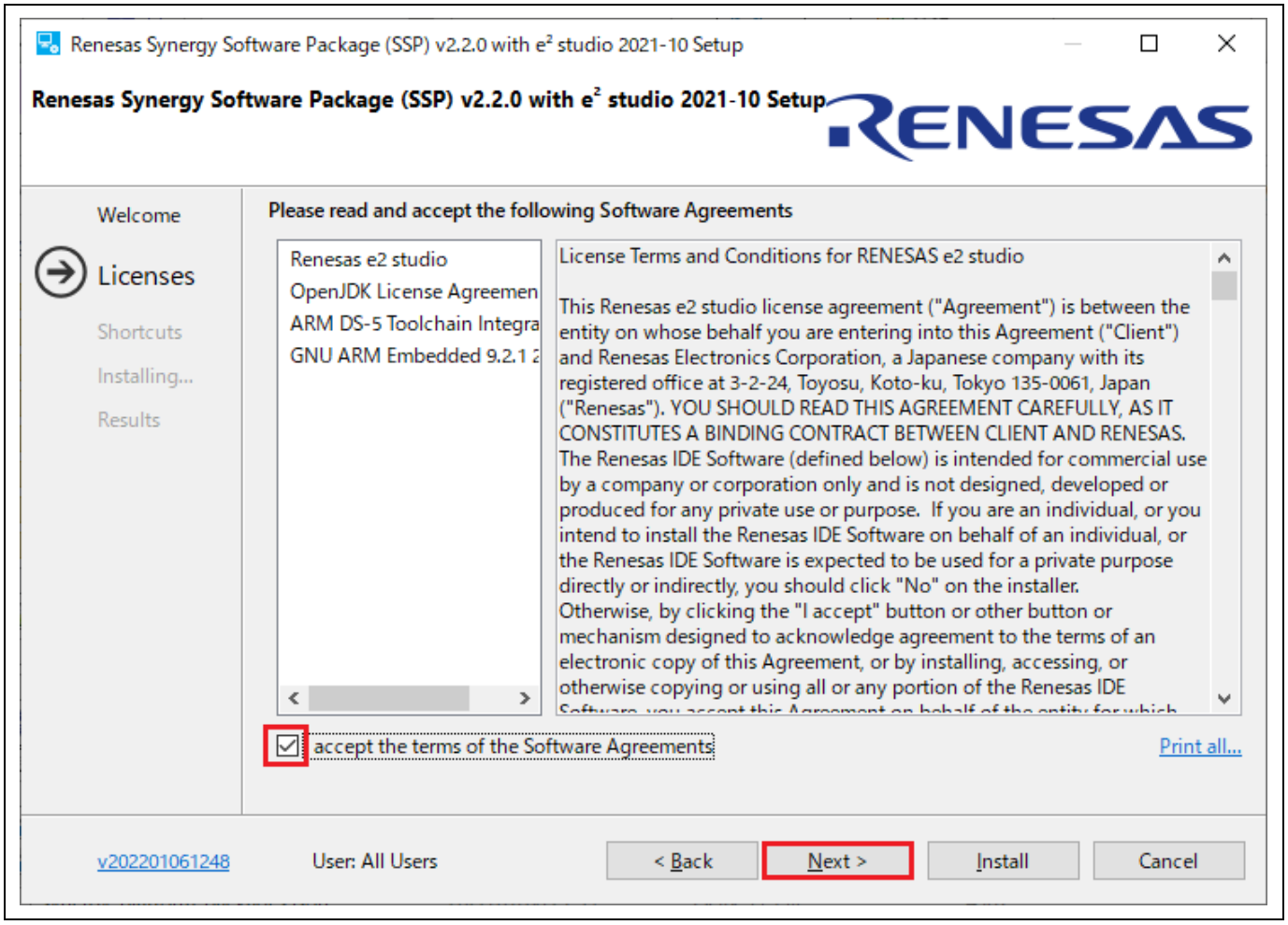


Figure 2-6. Installation – Software Agreements

4. Click **OK** to finish the installation.

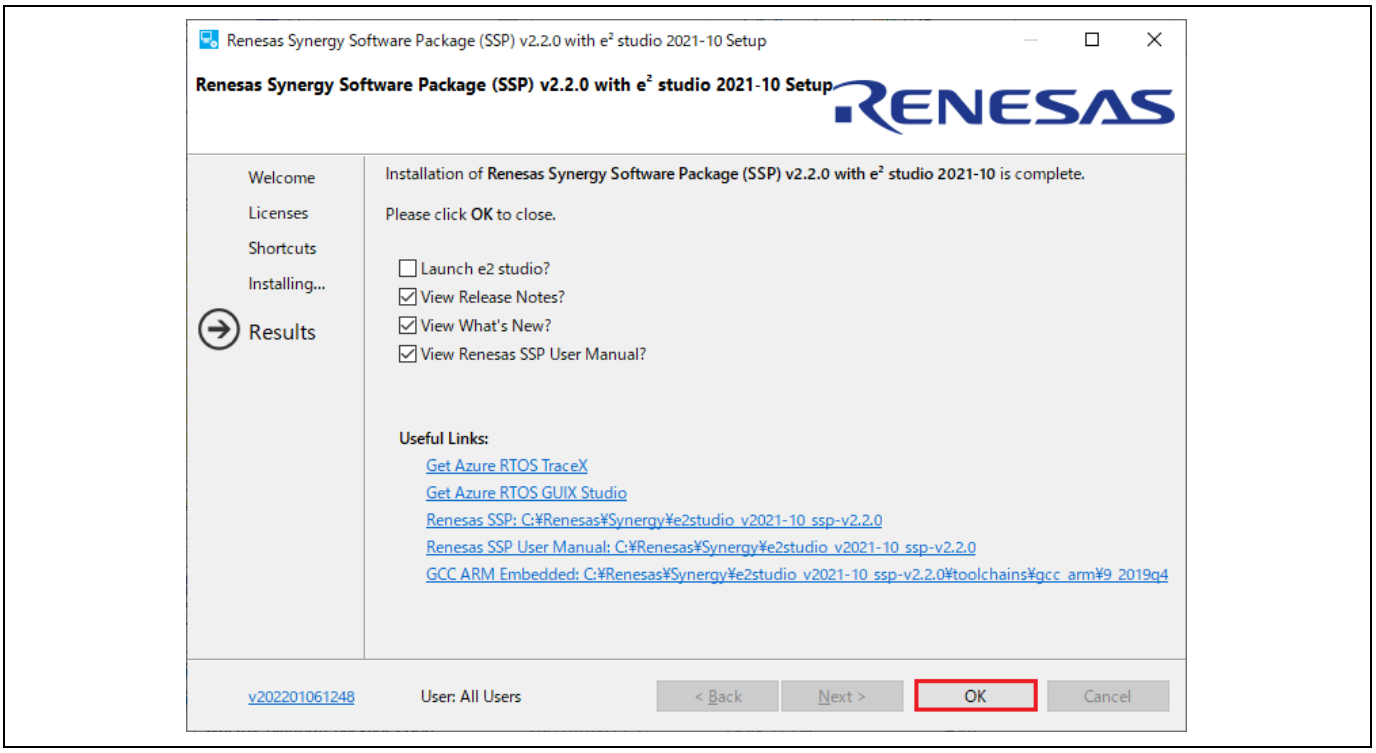


Figure 2-7. Installation – Complete Installation

2.2 Installing e² studio and SSP Independently

This section describes installation of the following components independently.

- e² studio IDE
- GCC ARM Embedded Compiler
- Renesas Synergy Software Package (SSP)

2.2.1 Installing e² studio

To install e² studio for Synergy, follow these steps:

1. Download the offline installer for e² studio (64-bit version) of the version compatible with your SSP from https://www.renesas.com/e2studio.
2. Unzip the download file and run the e² studio installer to invoke the e² studio installation wizard page.
3. If e² studio was installed in 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.

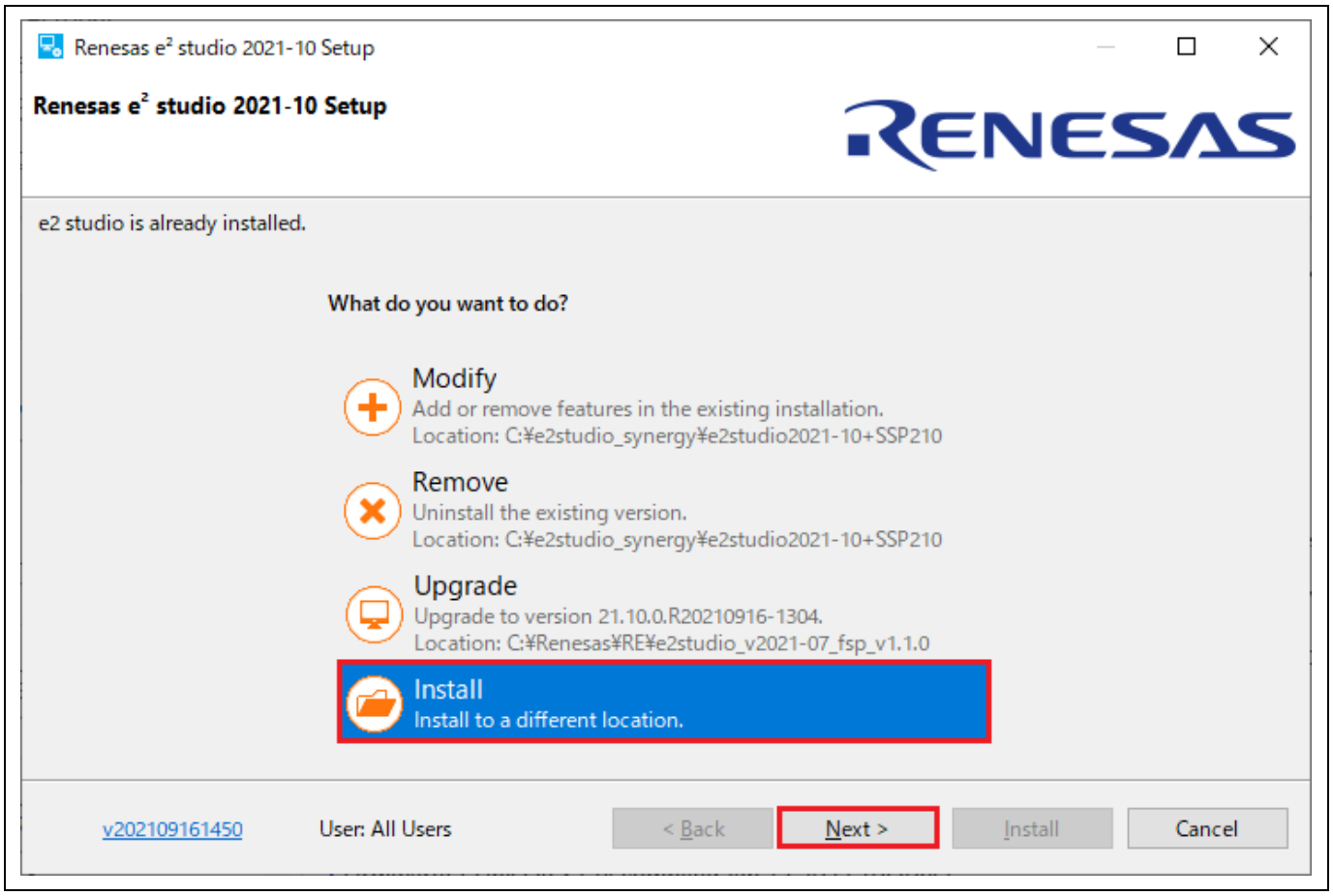


Figure 2-8. Install Multiple Versions of e² Studio

4. In the **Welcome** page, the default installation location is set to: `C:\Renesas\e2_studio`. You can click **Change…** to modify it. Click the **Next** button to continue.

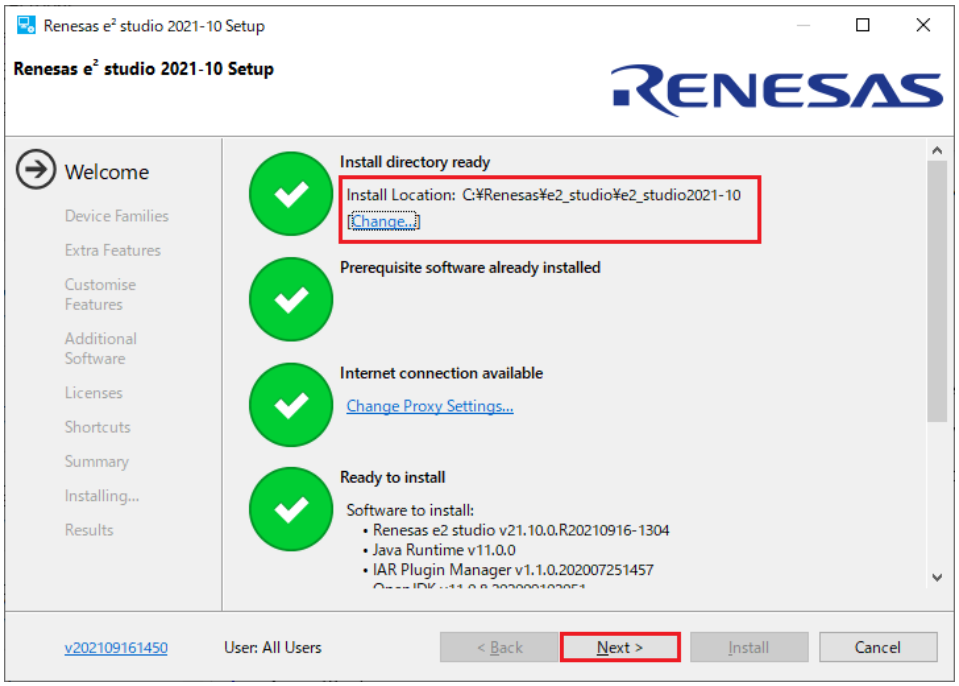


Figure 2-9 Installation – Welcome Page

5. Device Families page:
   Check the checkbox for **Renesas Synergy**. Checkboxes of other device families are optional.
   Click the Next button to continue.

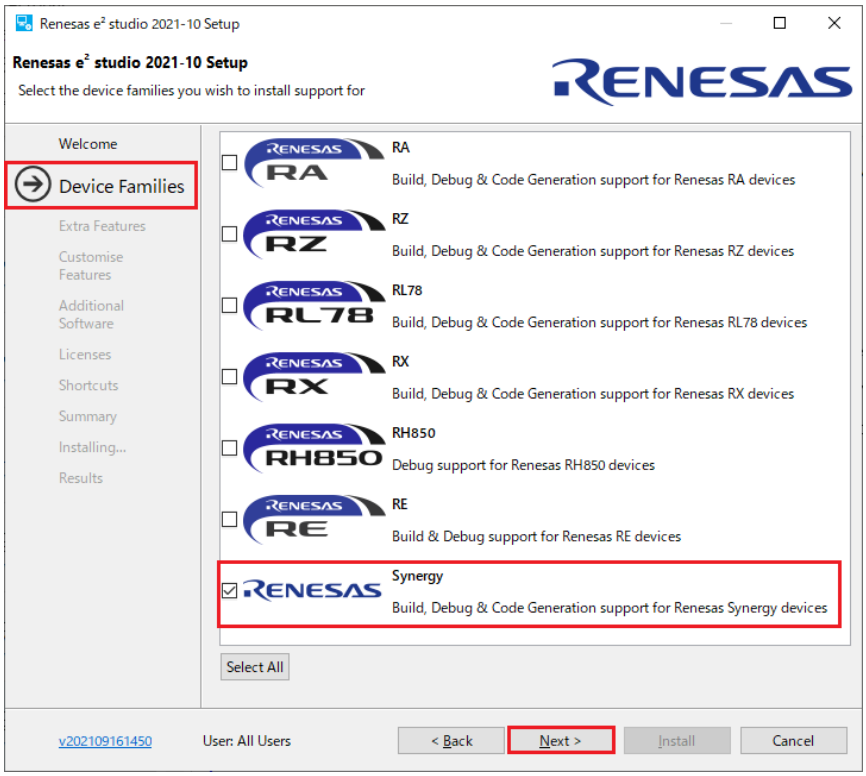


Figure 2-10. Installation – Device Families

6. **Extra Features** page:
   Select **Extra Features** (that is, Language support, Git Integration, RTOS support, and so on) to install.
   For non-English language users, select the language to support at this step.
   Click the **Next** button to continue.

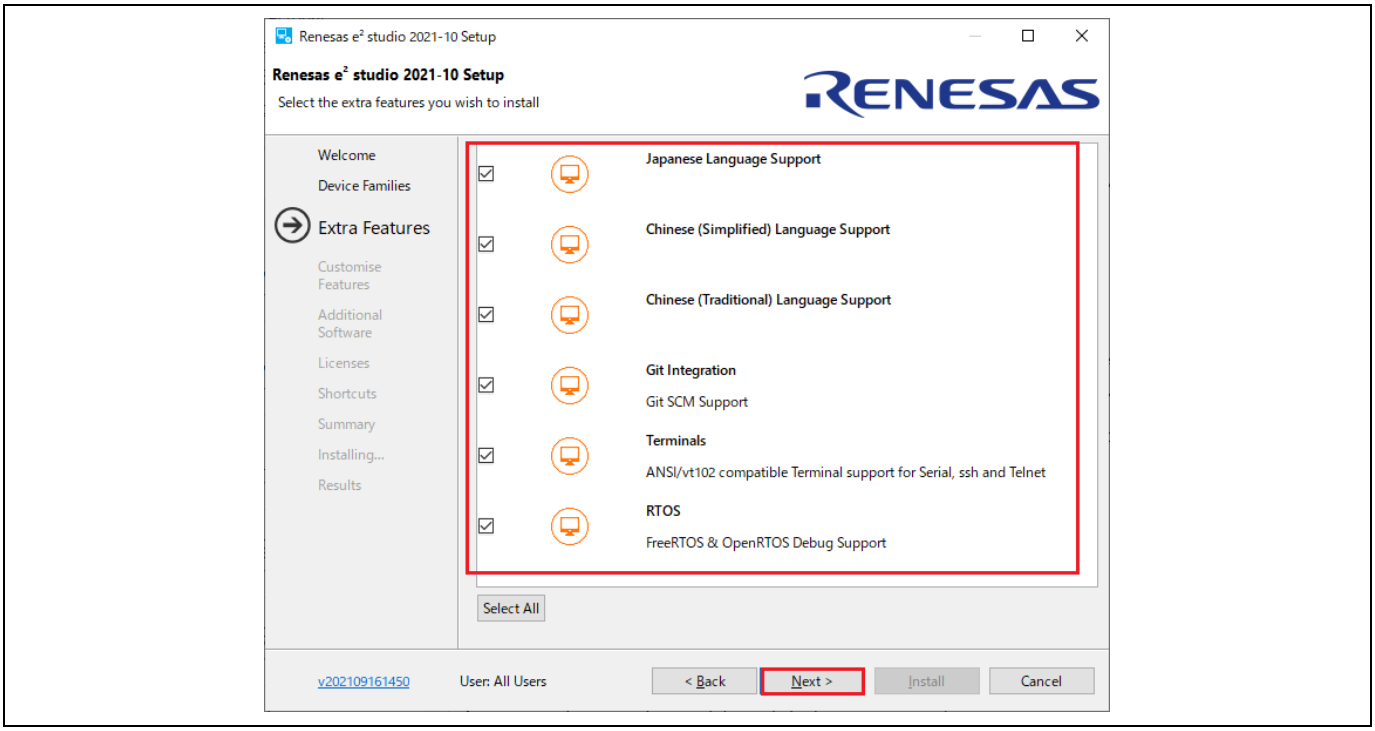


Figure 2-11. Installation – Extra Features Page

7. **Customize Features** page:
   Ensure that **Renesas RA Family Support** is checked.
   Click the N**ext** button to continue.

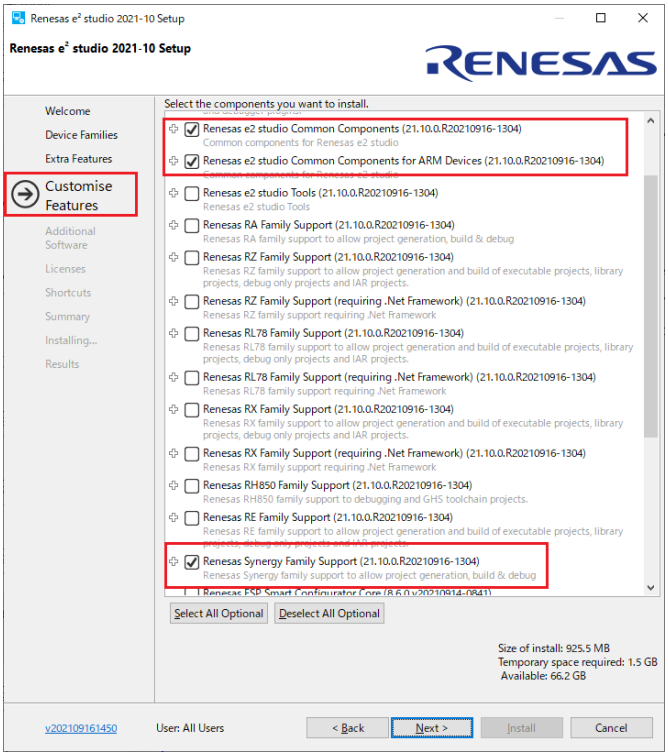


Figure 2-12. Installation – Components

5. Additional Software
   Check the checkbox for the latest version of GCC ARM Embedded in the Additional Software dialog (for example GNU ARM Embedded 9.2.1 or GNU ARM Embedded 7.2). The older versions are optional.
   Click the **Next** button to continue.

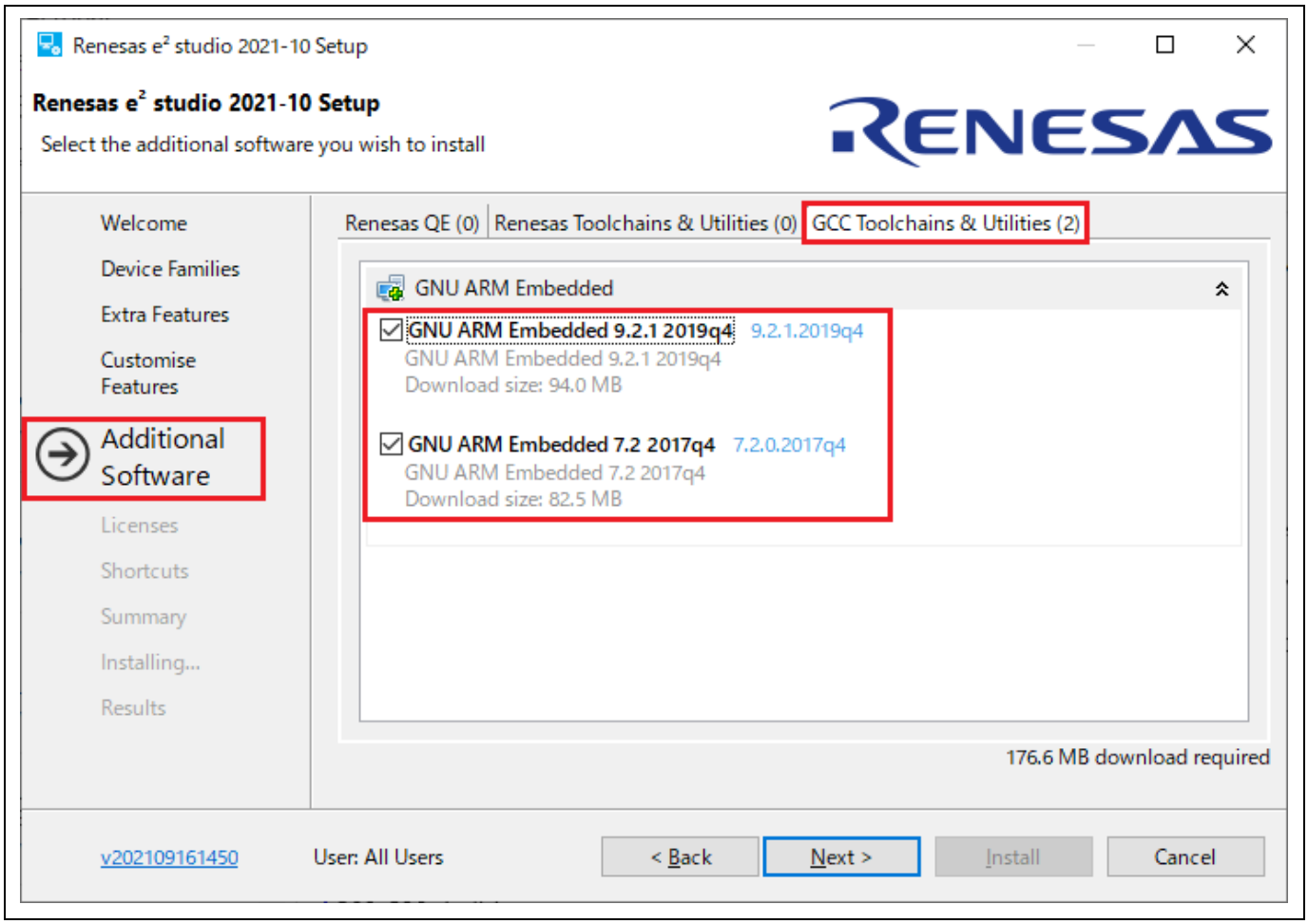


Figure 2-13. Installation – Additional Software

6. Licenses
   Read and accept the software license agreement to proceed with the **Next** button.
   Please note that users must accept the license agreement, otherwise installation cannot proceed.
7. Shortcuts
   Select the shortcut name for the start menu and click **Next** button to continue.
8. Summary
   Click the **Install** button to install Renesas e² studio.
9. Installing…
   The installation will start. Depending on the items selected in the **Additionak Software** dialog, new dialogs may open to proceed with the installation of these software packages.

2.2.2 Setting Up the GNU ARM Compiler

The GNU ARM toolchain can be installed during e² studio installation. To install the GNU ARM compiler separately, follow these steps:

1. Download the latest version of the GNU ARM compiler supported by Renesas Synergy™ (for example, v9.2.1) 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 compiler 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**. Click **Finish** to complete the installation.

2.2.3 Installing the Renesas Synergy™ Software Package (SSP)

The e² studio installer does NOT include the Renesas Synergy™ Software Package (SSP). The SSP must be installed separately unless the Platform Installer is used. The SSP Package Installer includes the driver library, an evaluation license for SSP, HTML User's Manual, and a readme file.

To install the SSP, follow these steps:

1. Visit the Solutions Gallery of the Synergy Platform website.
2. Select **Download SSP** under **Synergy Software-> Synergy Software Package (SSP)** to go to the SSP page.
3. Select **Download** under **Synergy Download** to log in to your My Renesas account and then download the standalone installer.
4. In the Synergy Software Package page, select **Download Standalone Installer** to download the file `SSP_Distribution_<SSP-version>.zip`. (You must sign in to My Renesas account to enable the '**Download Standalone Installer'** option)

The release note and User Manual of SSP can also be downloaded from the Synergy Software Package page.

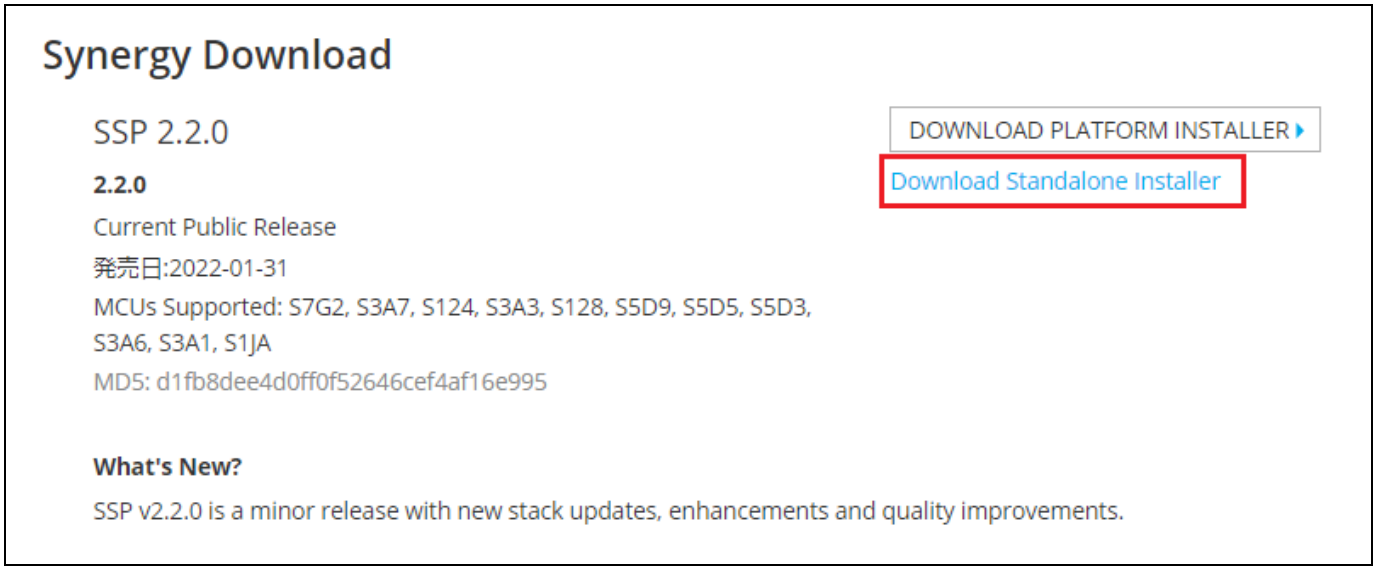


Figure 2-14. Installation – Download the standalone SSP package

1. Make sure that a compatible e² studio was installed and closed during this installation.
2. Unzip the package and run the `SSP_Distribution_<SSP-version>.exe` installer.
3. Click **Next** on the installation wizard dialog.
4. Read the License Agreement and click **I Agree** to continue the installation process.

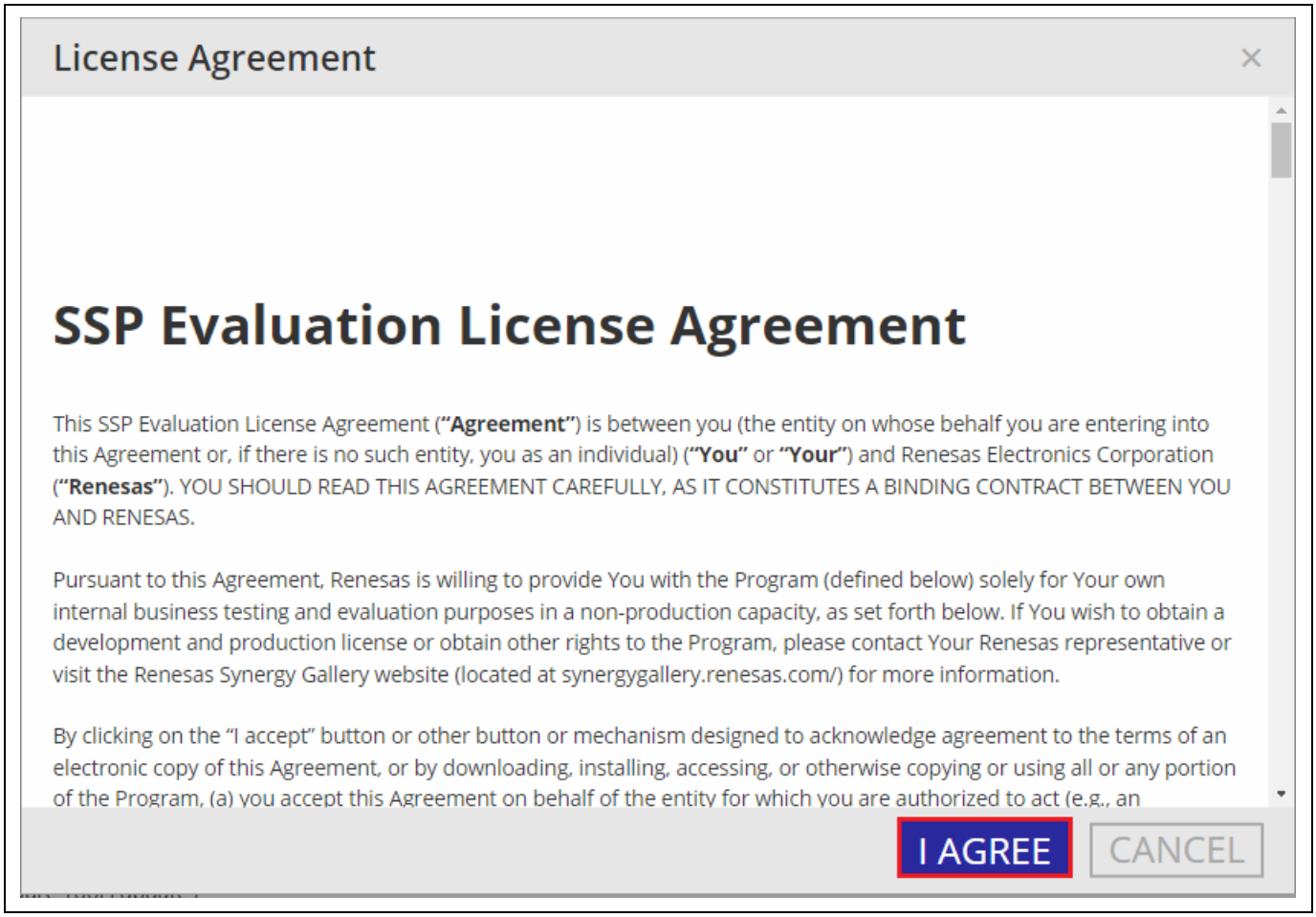


Figure 2-15. Installation - License Agreement on Renesas Web

Install the SSP in the root folder (default root folder is `C:\Renesas\e2_studio`) of e² studio. The default installation folder for the SSP is `C:\Renesas\e2_studio`. Click **Install** to start the installation.

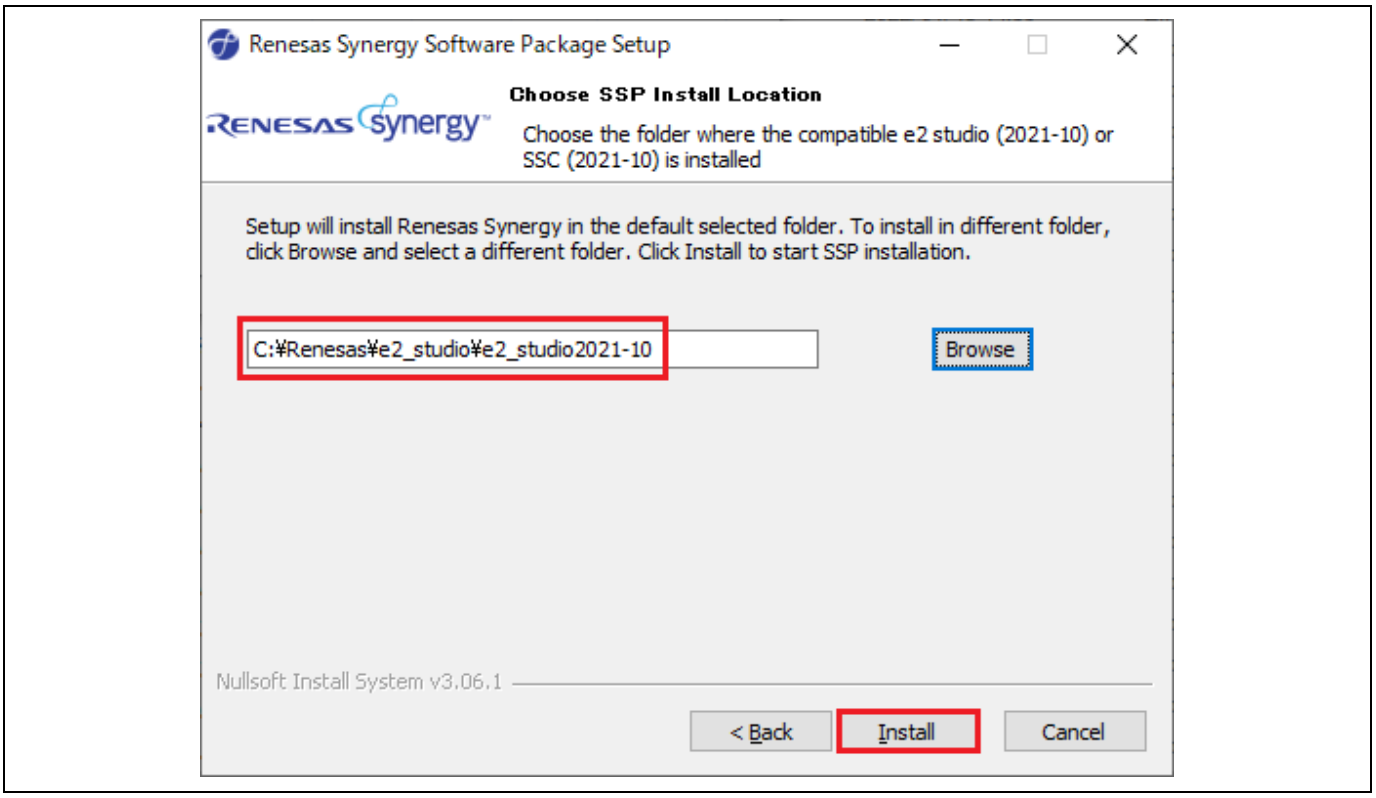


Figure 2-16. Installation – SSP Installation Folder Selection

6. Click the **Close** button to close the installation wizard when installation is done.
   After the SSP is installed, the evaluation license file can be found in the directory `<e2_studio_base_dir>/internal/projectgen/arm/licenses/`.

2.3 Uninstalling e² studio

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

1. Click on **Start** → **Control Panel** → **Programs and Features**
2. From the currently installed programs list, choose **e² studio** and click the **Uninstall** button.
3. Click **Uninstall** to confirm the deletion in the Uninstall dialog.

At the end of the uninstallation, e² studio will be deleted from the installed location and the Windows shortcut menu is removed.

3. Project Generation

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

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

As a prerequisite, the SSP and the toolchain must be installed on the host machine as described in chapter 2.

3.1 Generating a New Synergy Project

e² studio provides a simple wizard for creating projects. You can create a new Synergy project by specifying the project name, corresponding devices and boards, project type, output object type, and project template.

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

1. Select **File** → **New** → **Synergy C/C++ Project**.

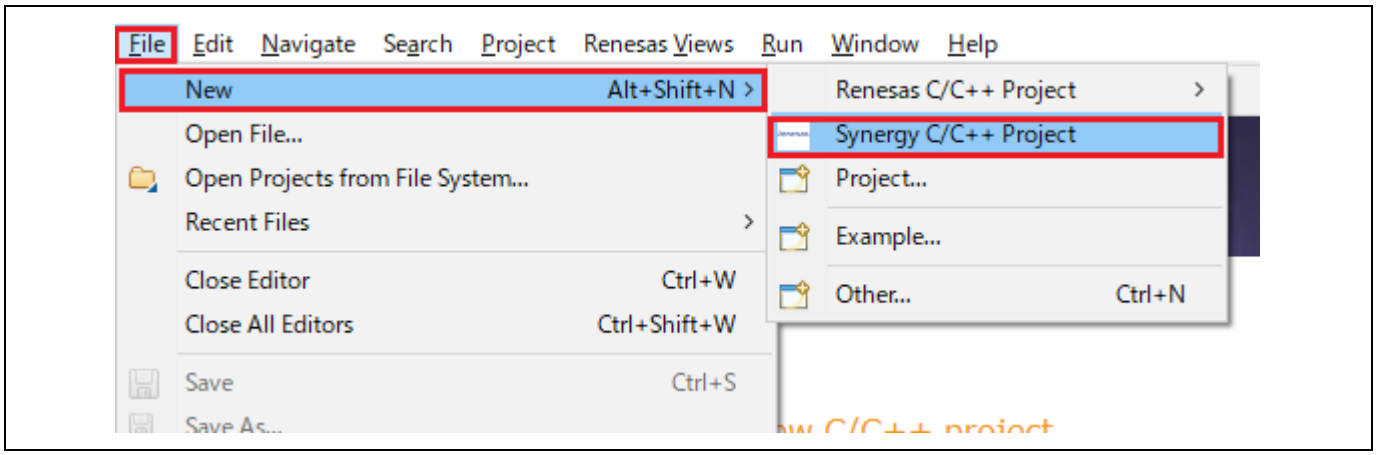


Figure 3-1. Project Generation – New Project Creation

2. Select the **Renesas Synergy C Executable Project** template. Click **Next** to continue.

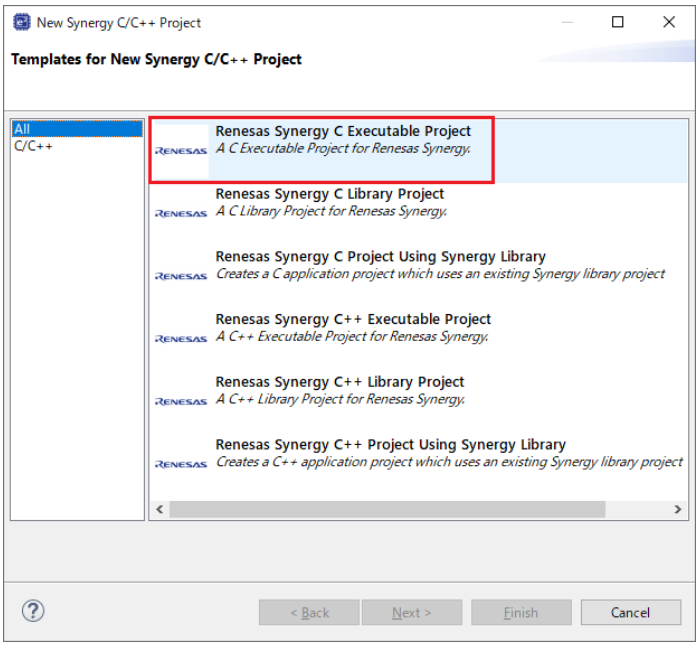


Figure 3-2. Project Generation – Select executable project template

3. In the device selection dialog, enter device and tool information:
   - Board (for example, S7G2 SK)
   - Toolchain version: Latest GNU compiler approved for use with Renesas Synergy™ (for example, GCC ARM Embedded 9.2.1.20191025)

     Note: IAR Embedded Workbench for Synergy is no longer available for download from renesas.com. Contact IAR for access to the Embedded Workbench for Synergy. Licenses for EWSYN are issued by IAR, users can find additional information on licensing Embedded Workbench for Synergy and request a new or updated license key from IAR.
   - Keep all other fields as default.
   - Click **Next** to continue.

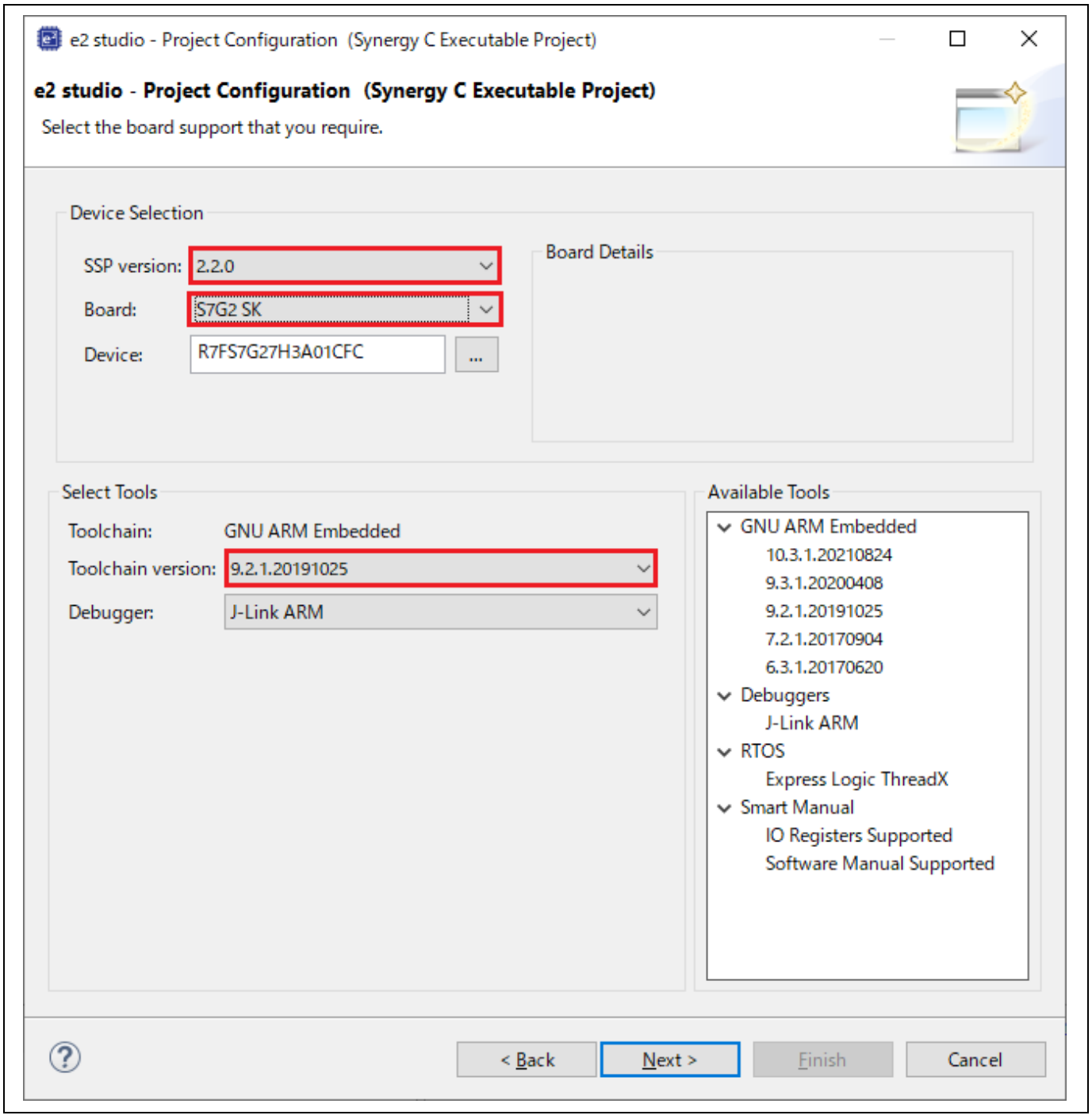


Figure 3-3. Project Generation – Device Selection

4. In the project template dialog, select a project template, Blinky. Click the **Finish** button to create a new project.

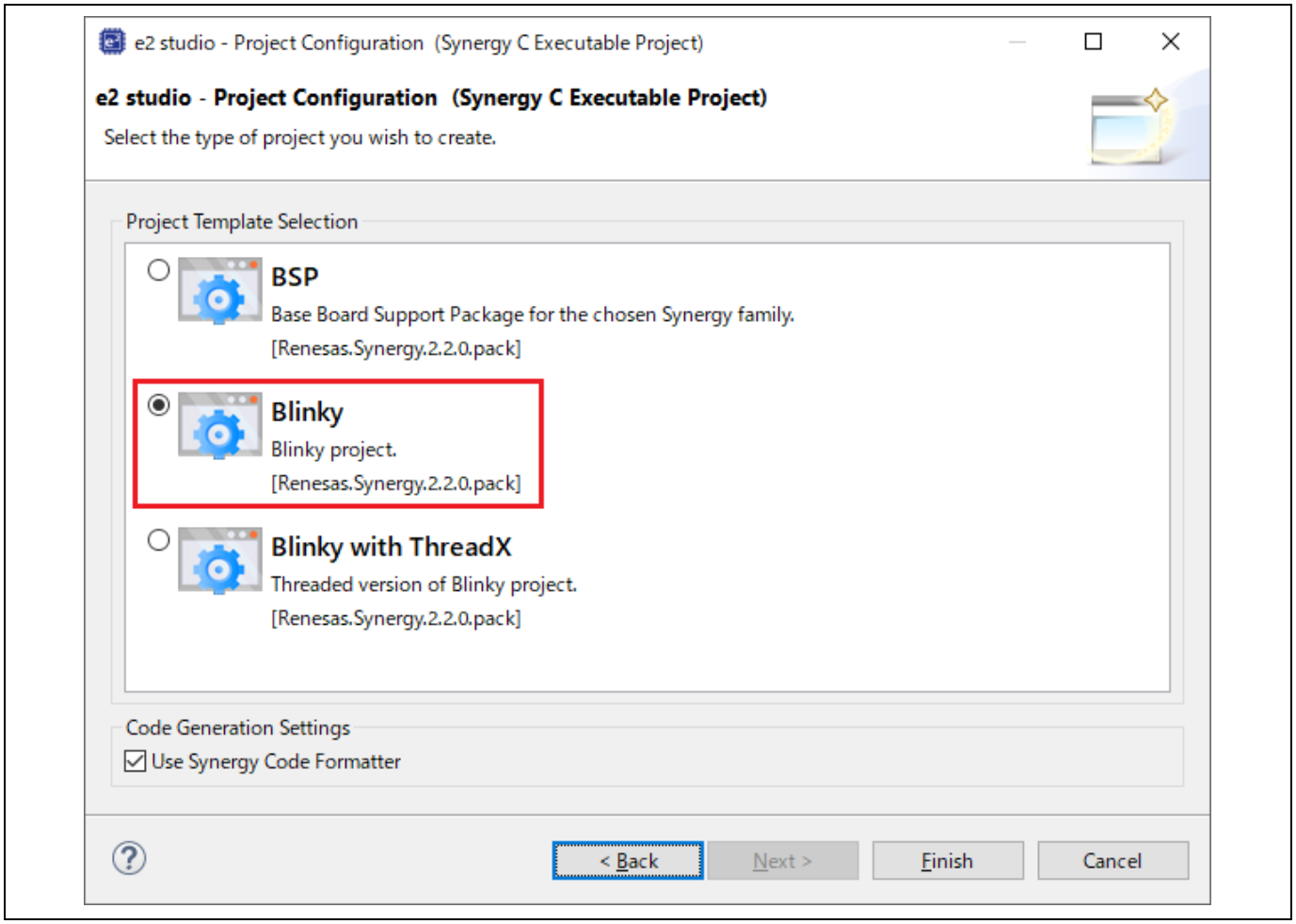


Figure 3-4. Project Generation – Project Template

5. You may be prompted to open the **Synergy 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 **Synergy Project Configuration** editor, and the **Package** view.

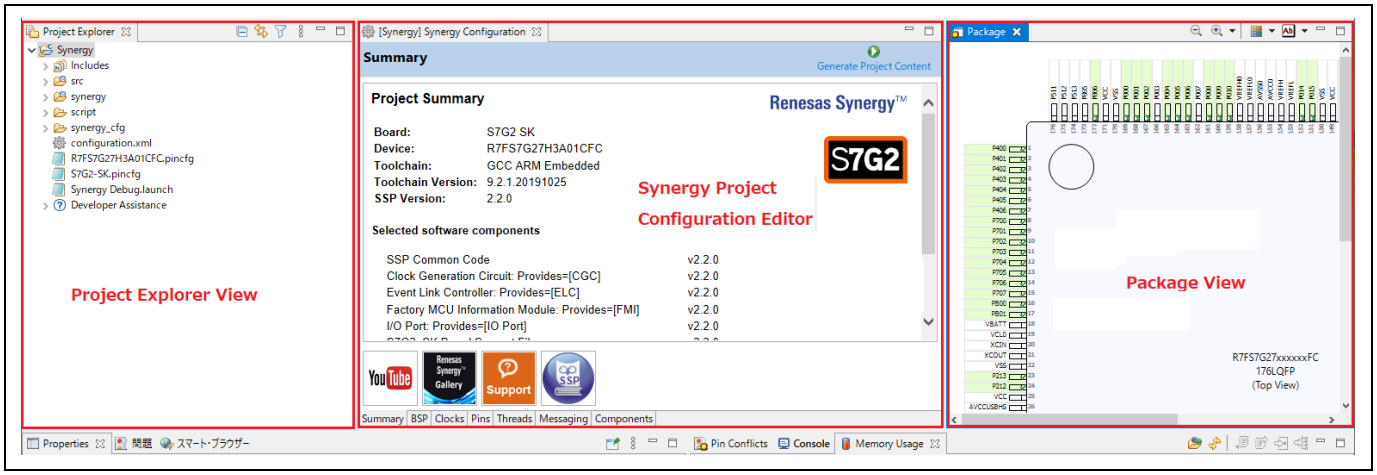


Figure 3-5. Project Generation – New Project Creation View

3.2 Importing an Existing Synergy Project

To import an existing Synergy Project, please follow the steps below:

1. Click **File → Import…**

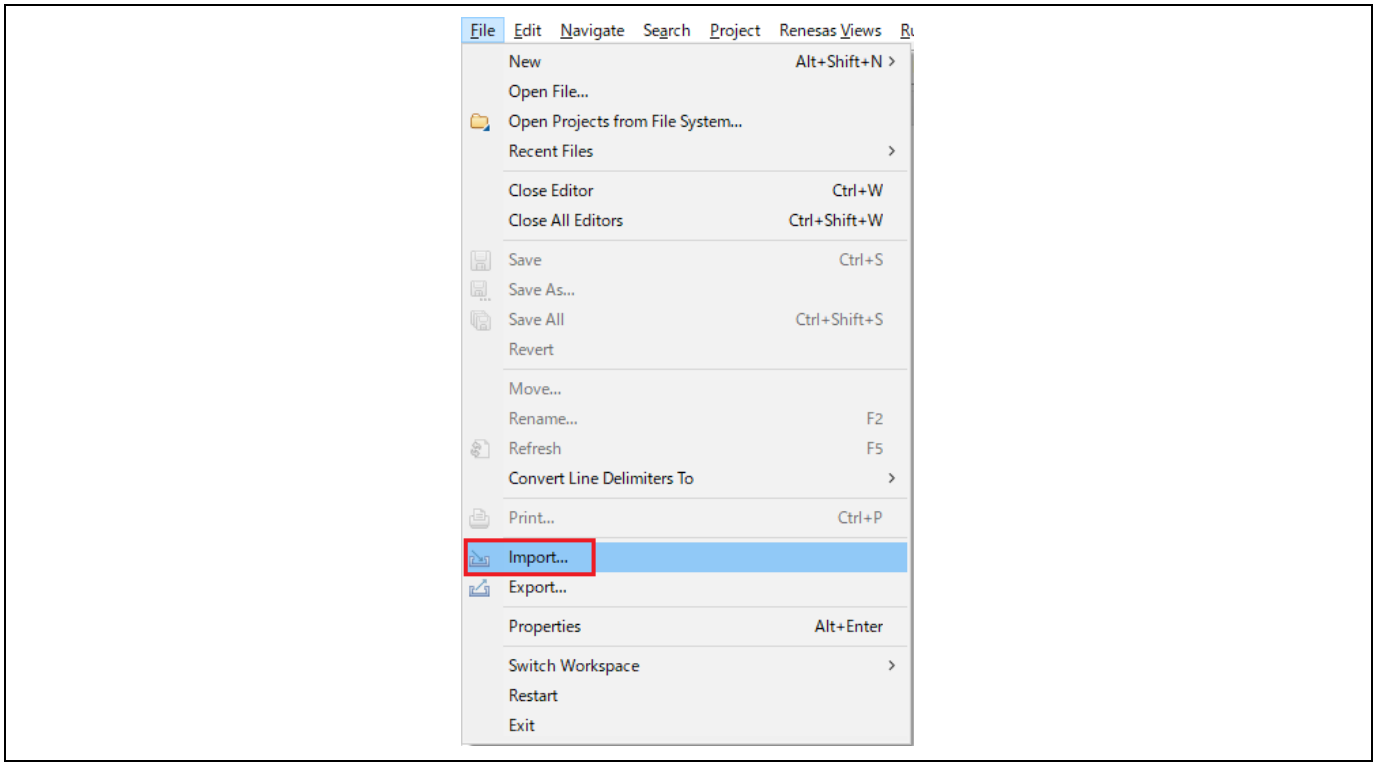


Figure 3-6. Import project

2. In the Import dialog, select **General → Existing Projects into Workspace**. Click **Next**.
   Note: To rename the project to be imported, select **General → Rename & Import Existing Projects into Workspace** instead.

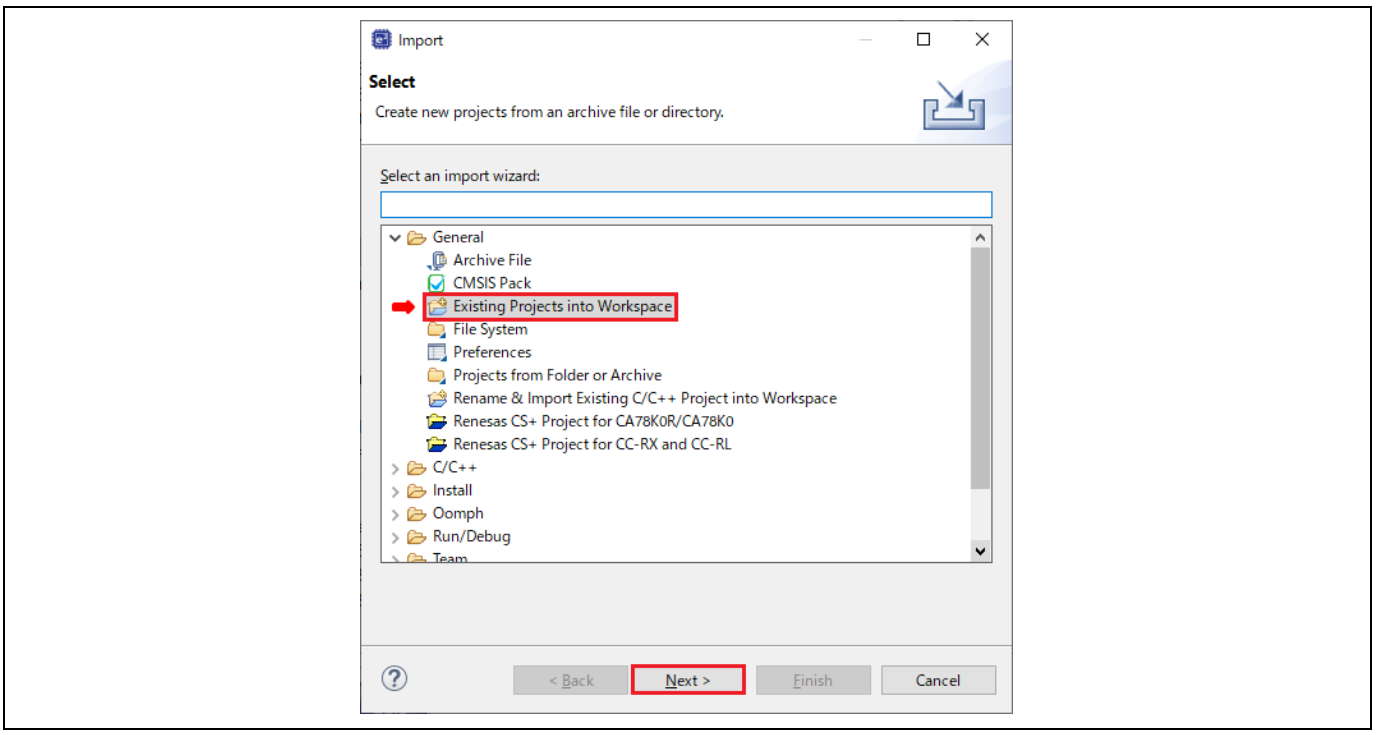


Figure 3-7. Select type of import

3. In the **Import Projects** dialog, select **Select archive file:** then **Browse…** to browse to the compressed file (.zip) containing the project.
   If the existing project is stored in a folder, then **Select root directory:** should be selected.
4. Select the project to import and click **Finish**.

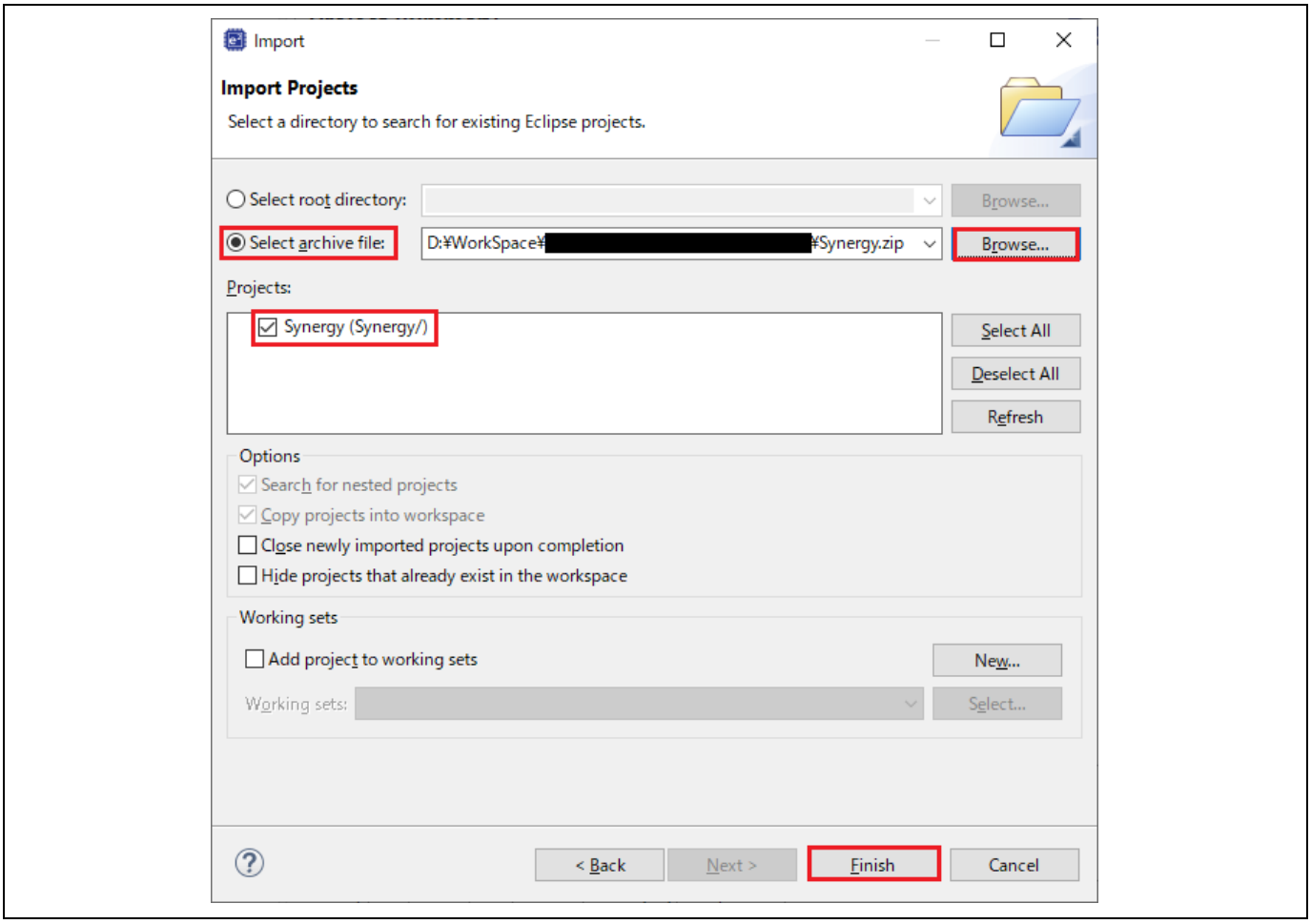


Figure 3-8. Select the project in the compressed file

5. The project will be imported to e² studio.

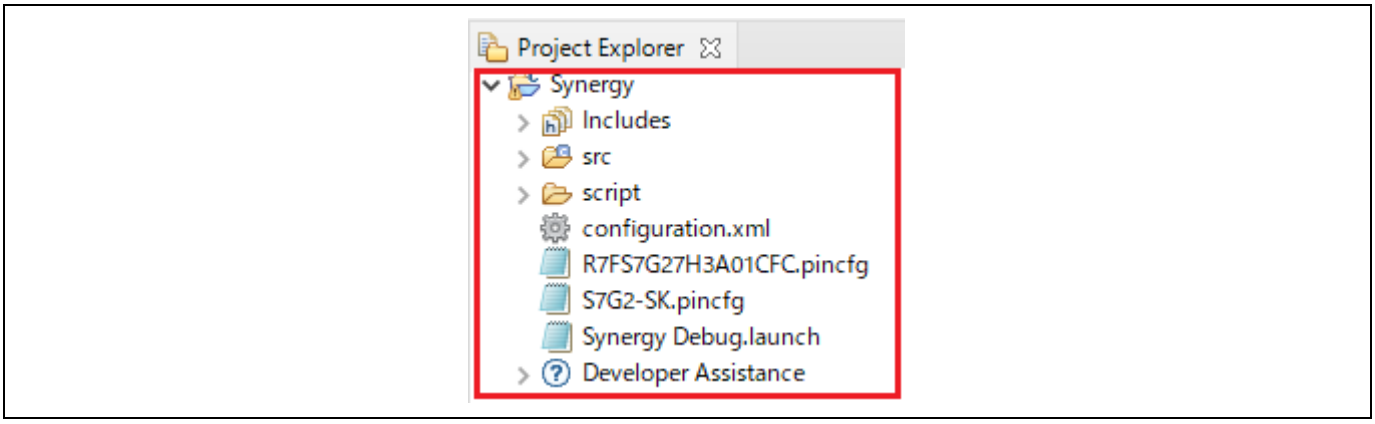


Figure 3-9. The imported project

3.3 Generating and Using a Synergy Static Library

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

3.3.1 Creating the Static Library Project

The following steps show an example of how to create a Synergy static library project:

1. Select **File** → **New** → **Synergy C/C++ Project**.
2. Select the **Renesas Synergy C Library Project** template. Click **Next** to continue.

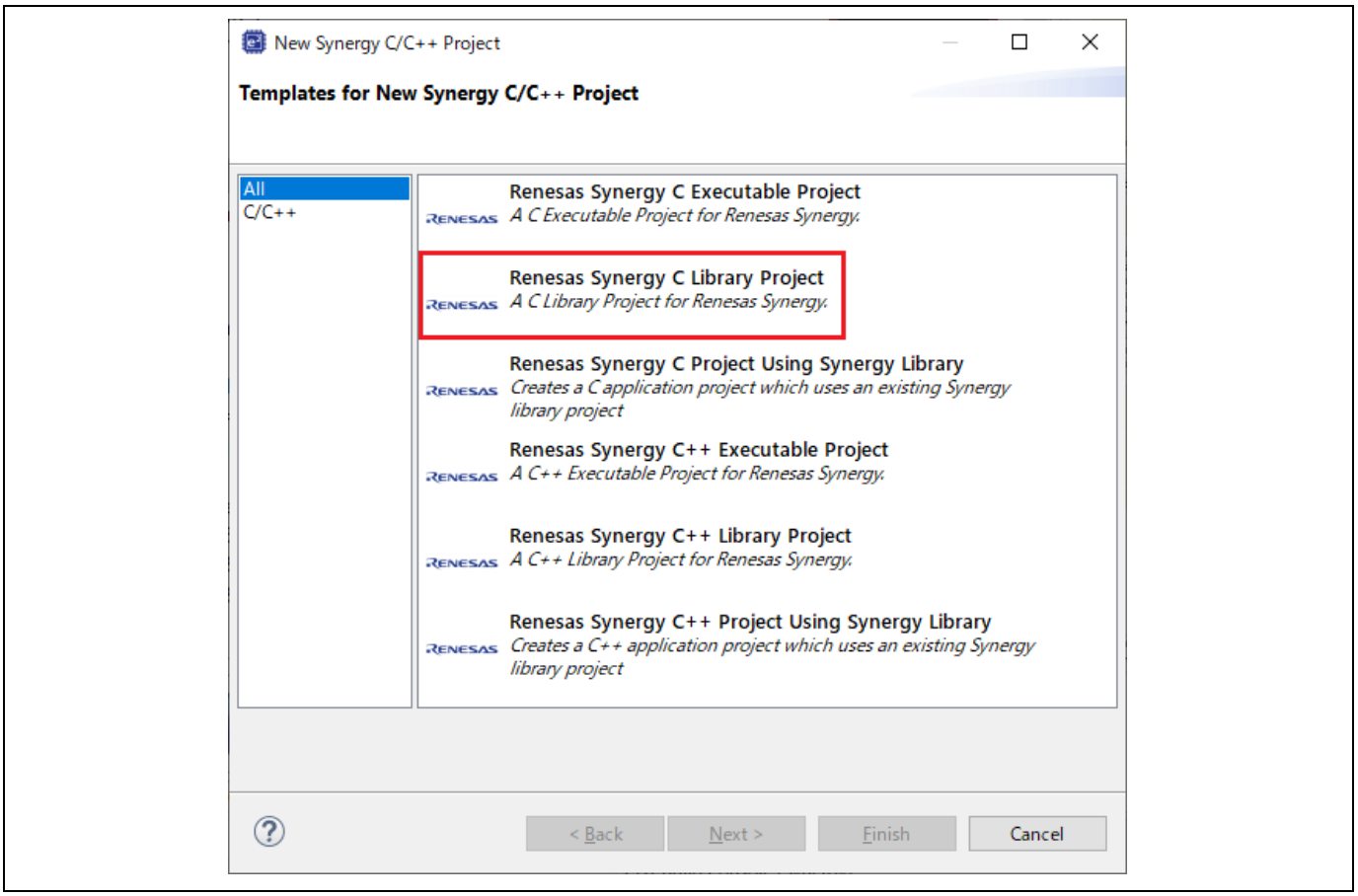


Figure 3-10. Project Generation – Select library project template

3. In the **Device and Tool Selection** dialog, select the same device and toolchain as your executable project and click **Next**.

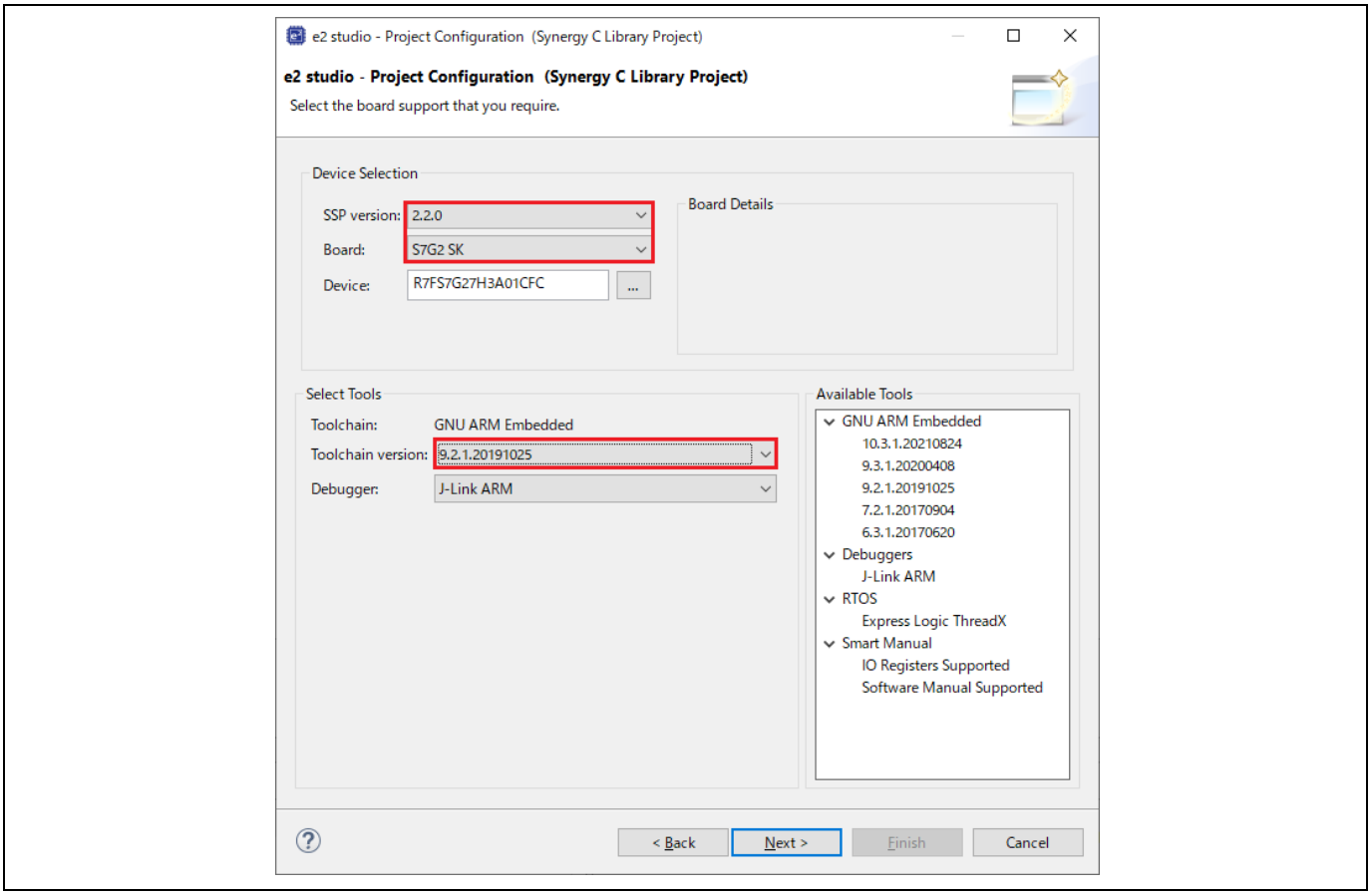


Figure 3-11. Select device and toolchain

4. In the project template dialog, select **Blinky**, then click **Finish** to create the project.

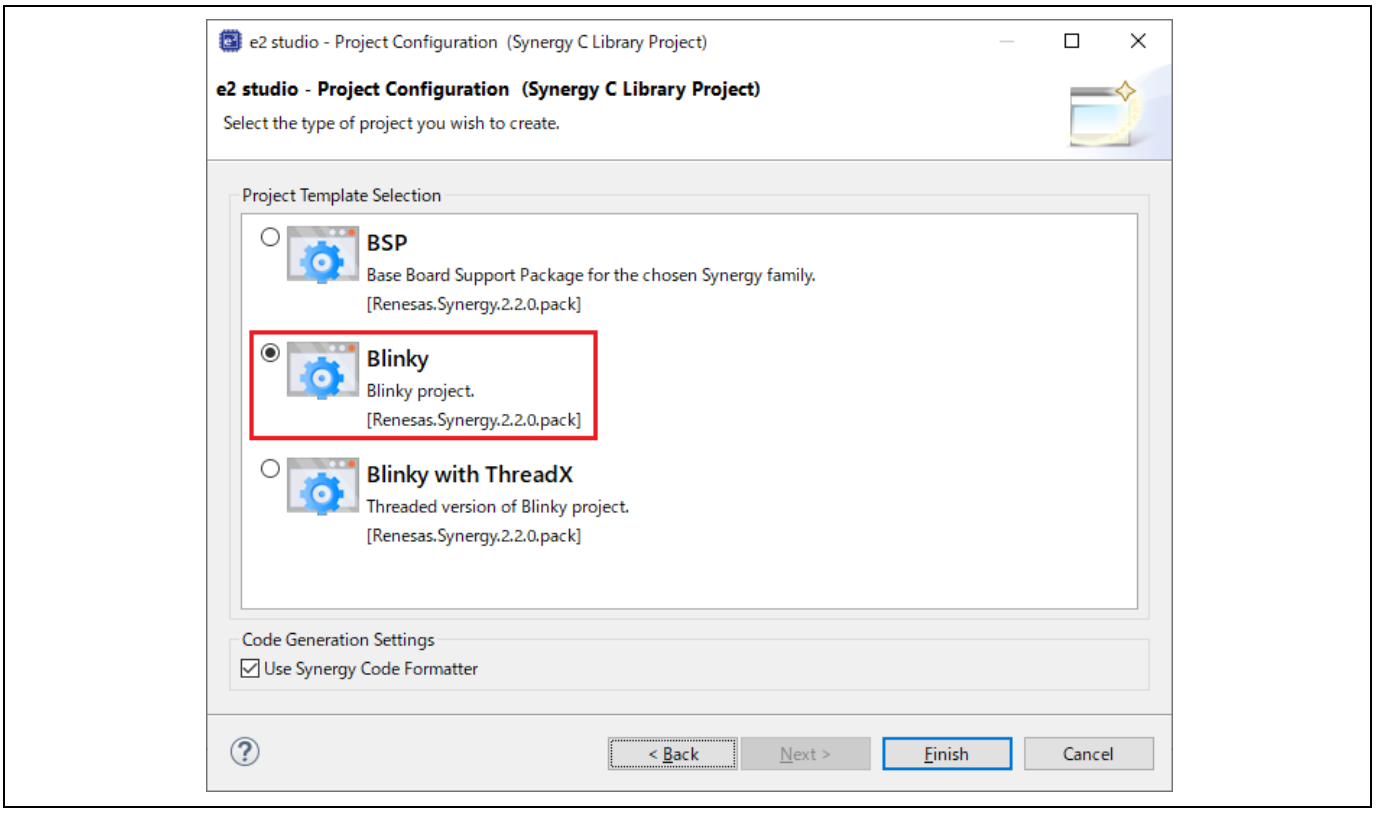


Figure 3-12. Select project template for library

5. The e² studio may prompt user to switch to Synergy perspective. Click **Yes** to open it.
6. Click **Generate Project Content**.

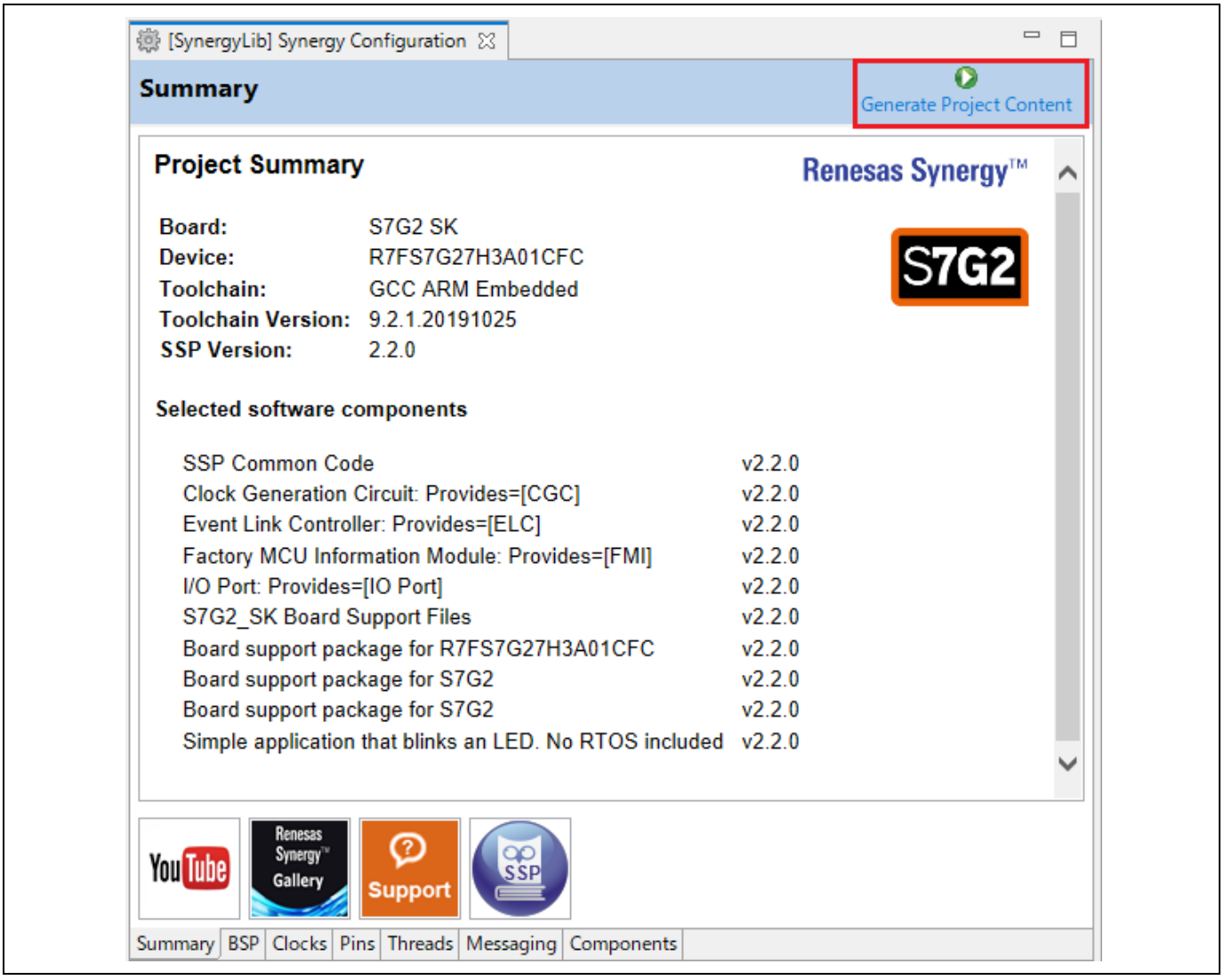


Figure 3-13. Generate library project content

7. From the project explorer window, open `hal_entry.c` under `SynergyLib\src\`.

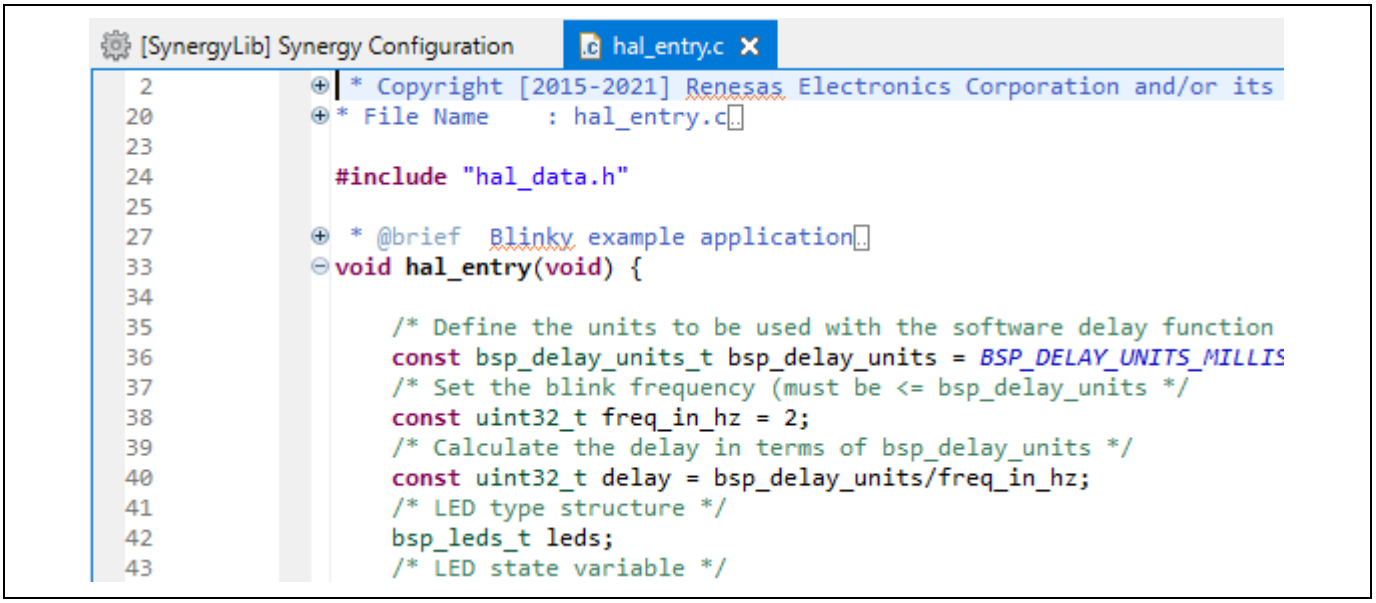


Figure 3-14. Old “hal_entry.c”

8. Then rename the function `hal_entry()` to `hal_entry_lib()` and add a declaration for `hal_entry_lib()`.

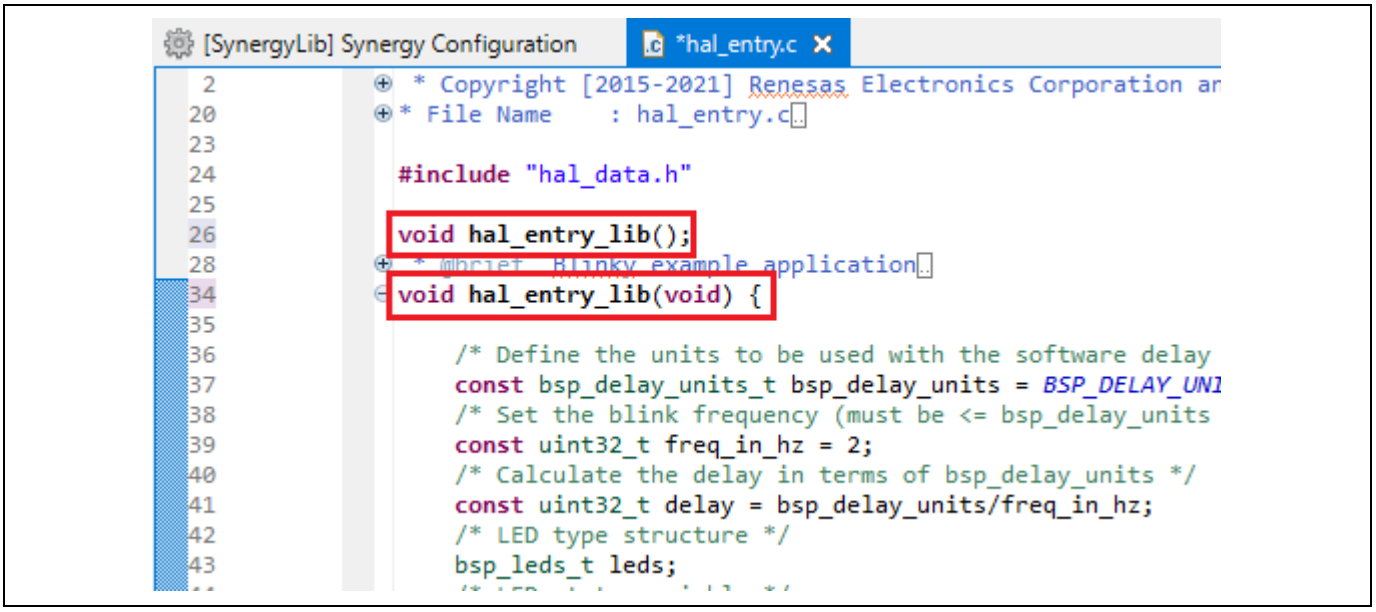


Figure 3-15. New “hal_entry.c”

9. Build the Library Project. The build outputs a static library file `SynergyLib\Debug\libSynergyLib.a`.

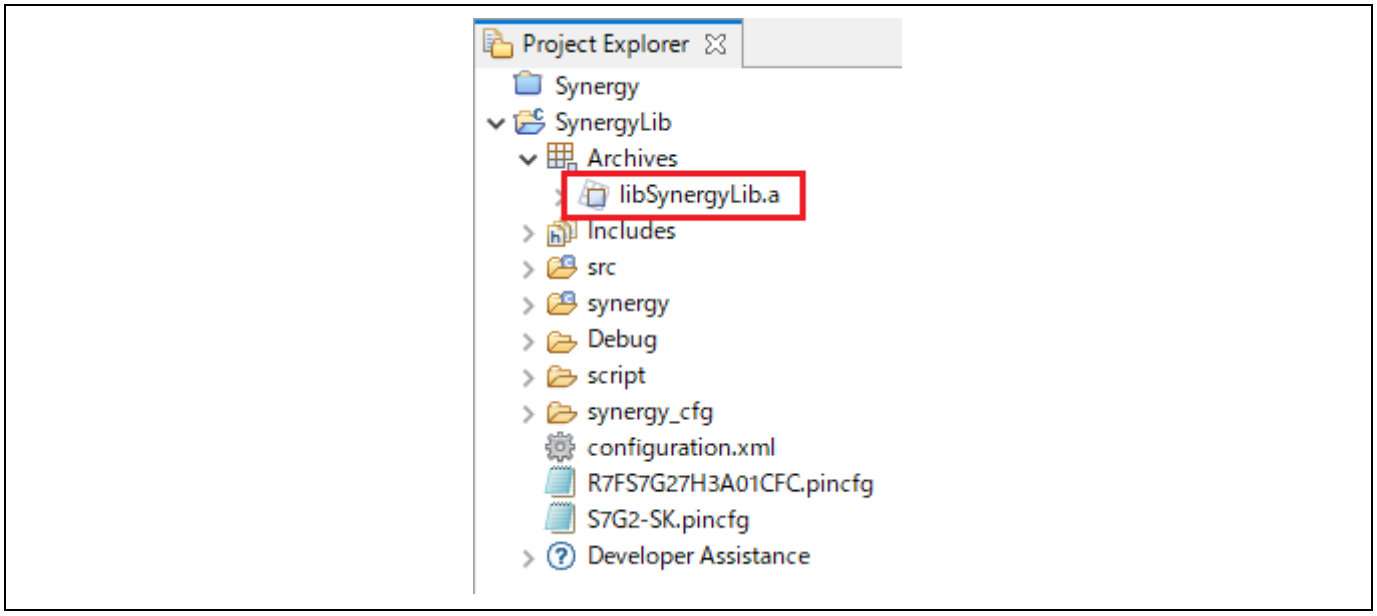


Figure 3-16. The built static library

3.3.2 Using Static Library in Executable Project

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

- Create a Synergy 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 Synergy executable project

Follow the following steps:

1. Create an executable project with template **Renesas Synergy C Project Using Synergy Library**.

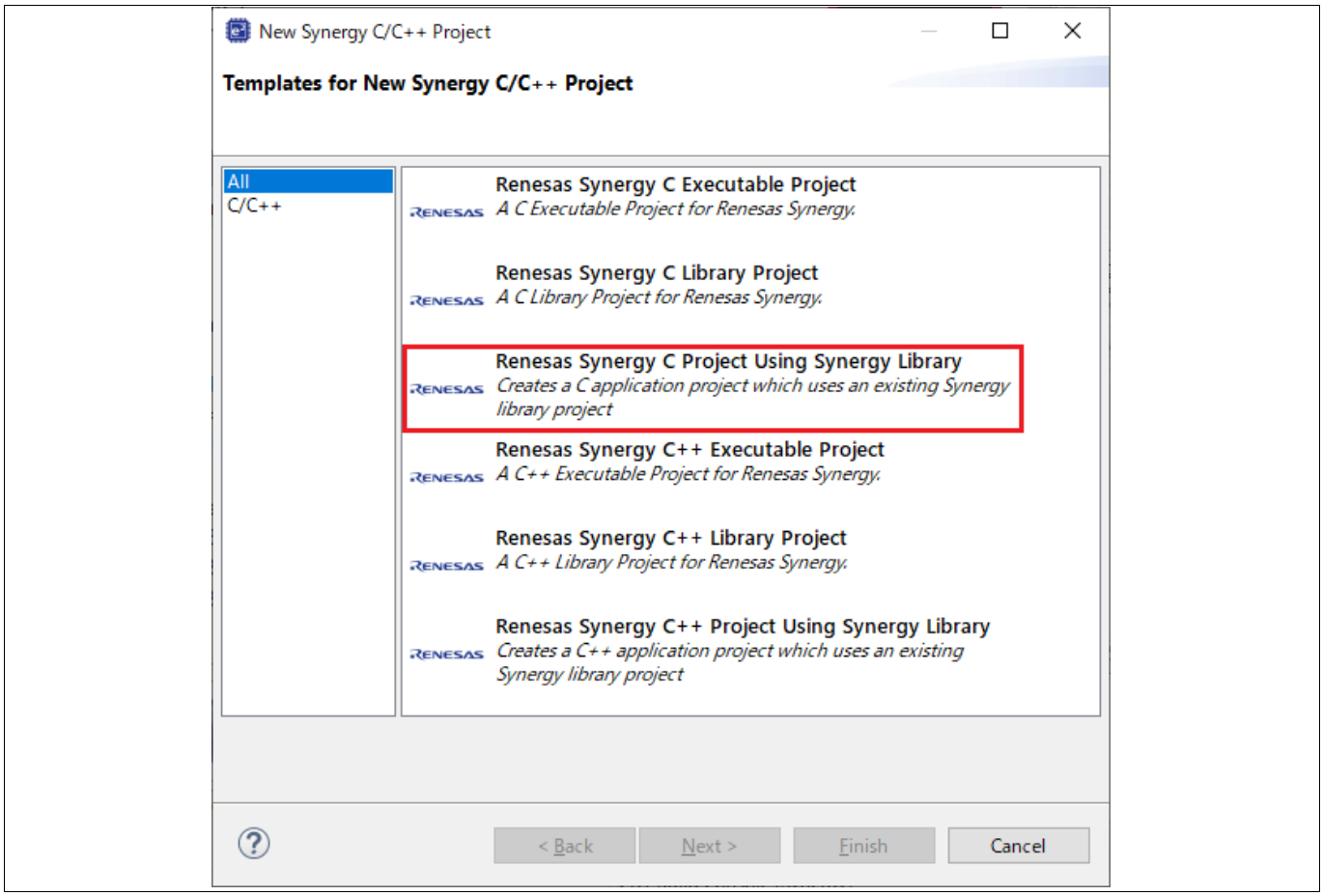


Figure 3-17. Select template for executable project using Synergy library

2. Name the project **synergyapp**.

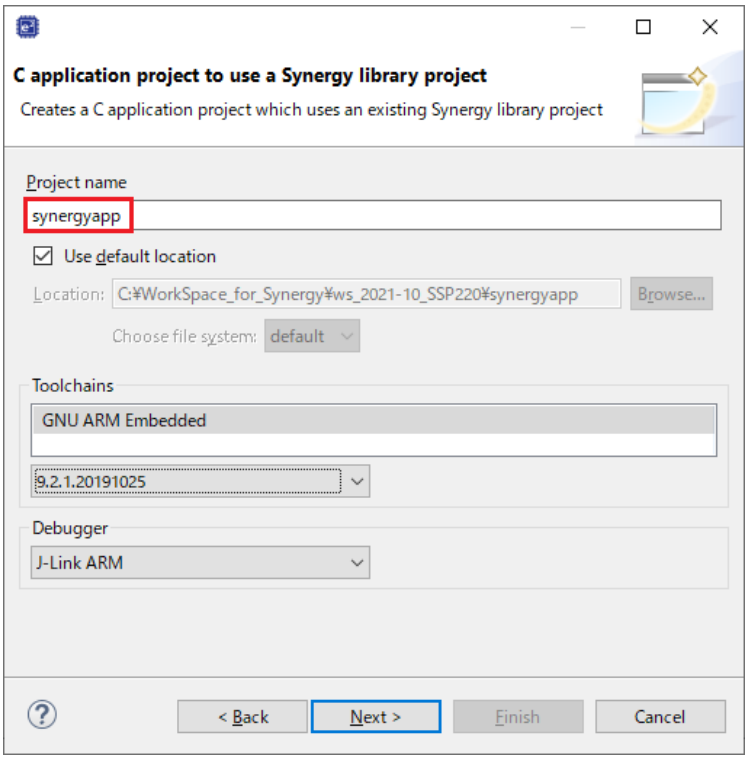


Figure 3-18. Specify project name

3. Select the library project. Click **Finish** to create the project.

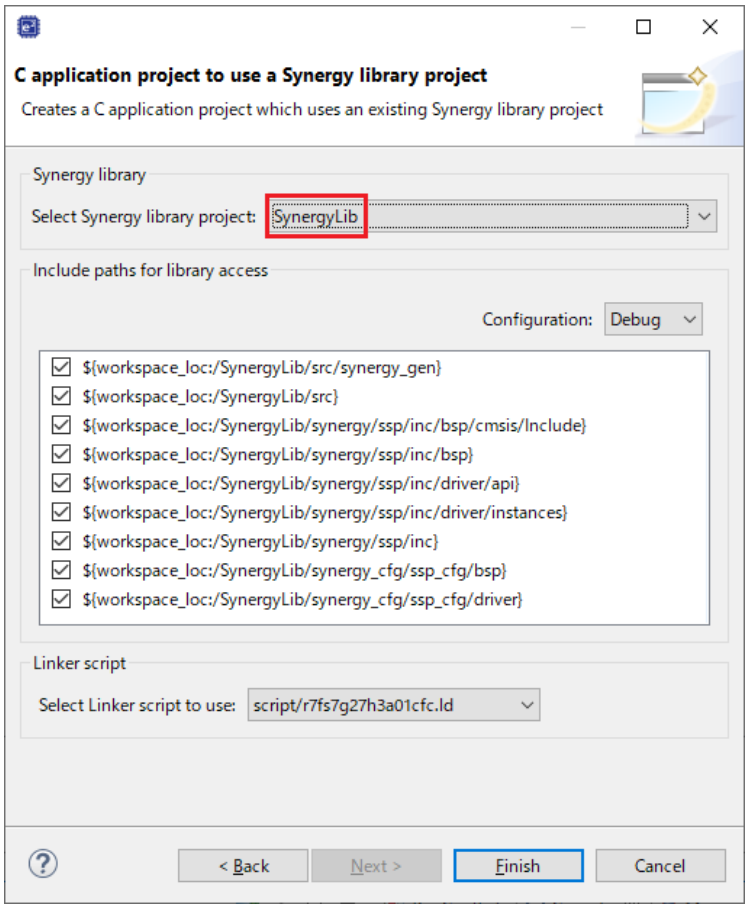


Figure 3-19. Select the library project

4. After the executable project is created, from project explorer window, open `hal_entry.c` under `synergyapp\src\`.

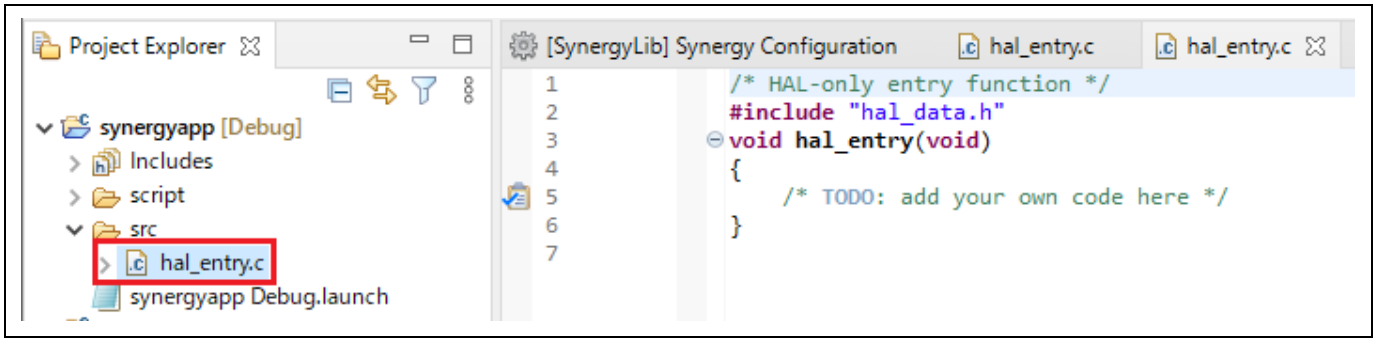


Figure 3-20. Old hal_entry.c

5. Add codes to call the LED blinking library function `hal_entry_lib()` in the `hal_entry()` function and add a declaration for the library function.

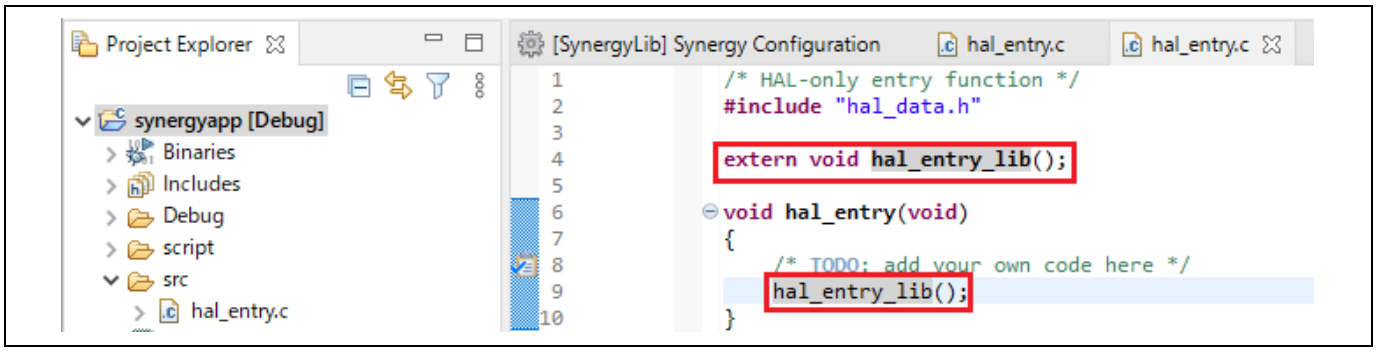


Figure 3-21. New hal_entry.c

6. Build the application project.
7. Set a breakpoint where the library function `hal_entry_lib()` is called. Run the **synergyapp** project.
8. When the program stops at the breakpoint, resume it. Confirm that the library function which blinks the LEDs (for example, `hal_entry_lib()`) is executed.

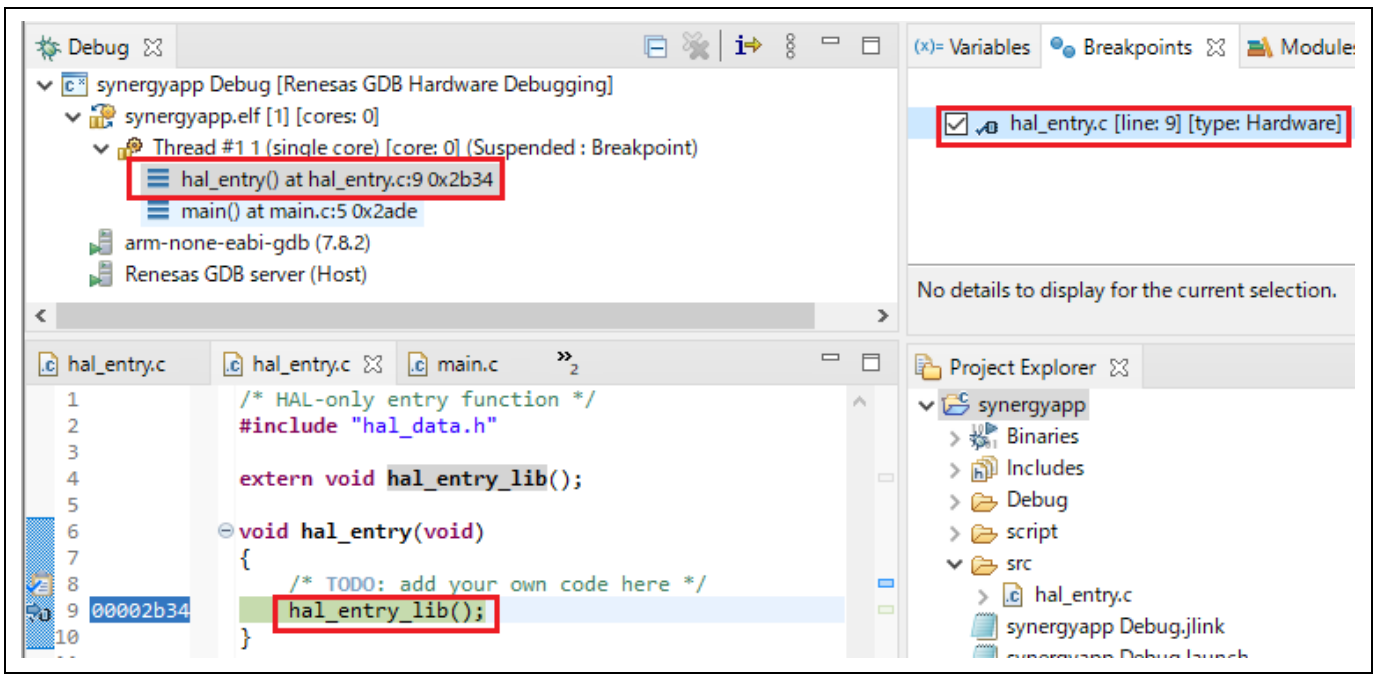


Figure 3-22. Application project executing library function

3.4 Synergy Project Configuration Editor

The Synergy Project 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. Drivers can range from simple hardware-level drivers to RTOS aware applications. Multi-thread specific components like mutexes, semaphores, and events can be configured.

To edit the project configuration, make sure that:

- Synergy Configuration perspective is selected in the upper right-hand corner of the e² studio window or click **Window → Perspective → Open Perspective → Other… → Synergy Configuration**
- The `configuration.xml` file is opened.

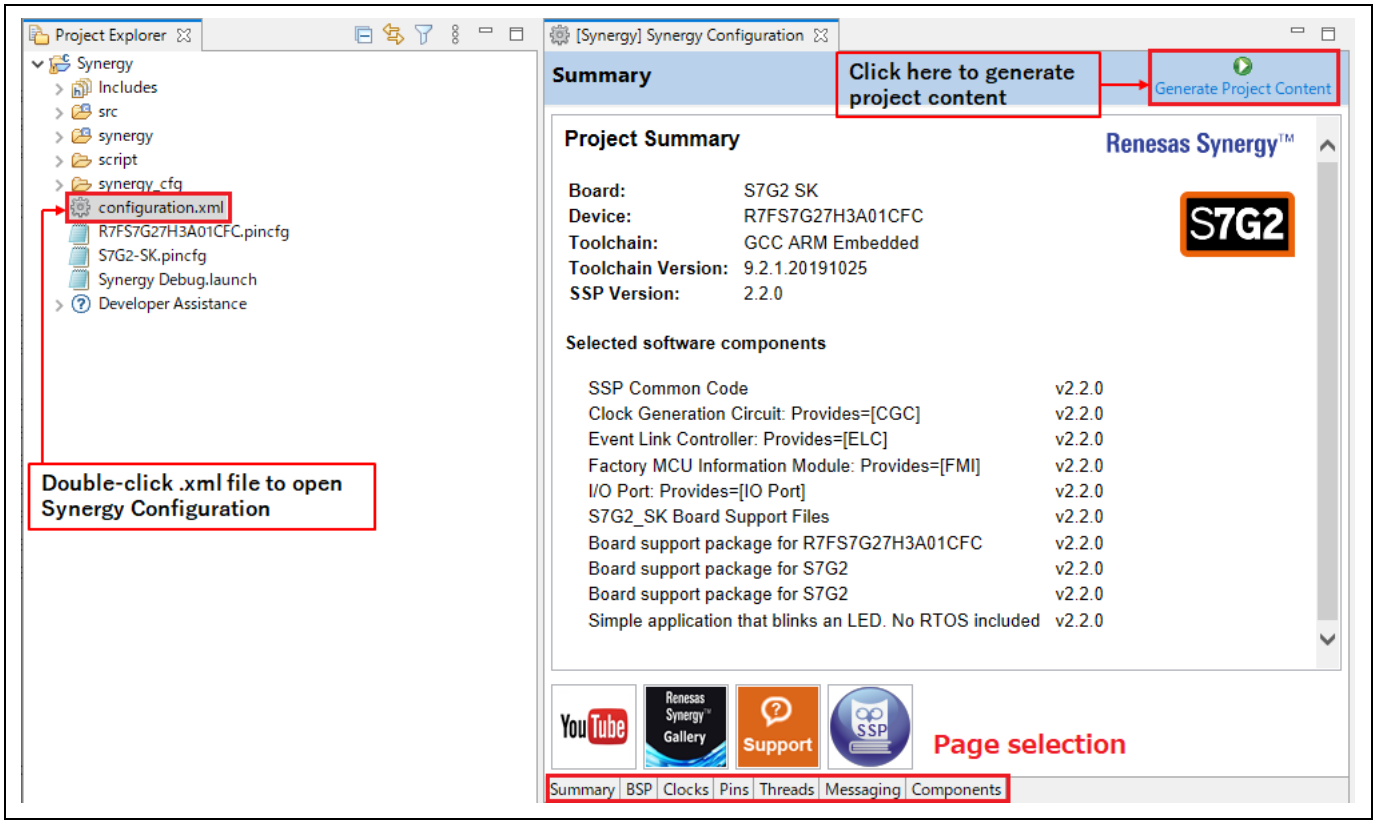


Figure 3-23. Synergy Project Configuration – Synergy Project Configuration View

There are 7 pages (or tabs) in the **Synergy Project Configuration** editor.

The **Summary** page contains a project-specific summary information.

The **BSP** tab allows users to select the SSP version, the type of Synergy board, and the device.

The configuration steps and options for the **Clocks**, **Pins**, **Threads**, **Messaging**, and **Components** pages are discussed in the following chapters.

3.4.1 Summary Page

The **Summary** page contains a project-specific summary which includes details of the currently selected device, board, and Synergy software components, and so on. There are also useful links to the Synergy Platform website, the 'Renesas Presents' YouTube channel, and the SSP user manual.

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

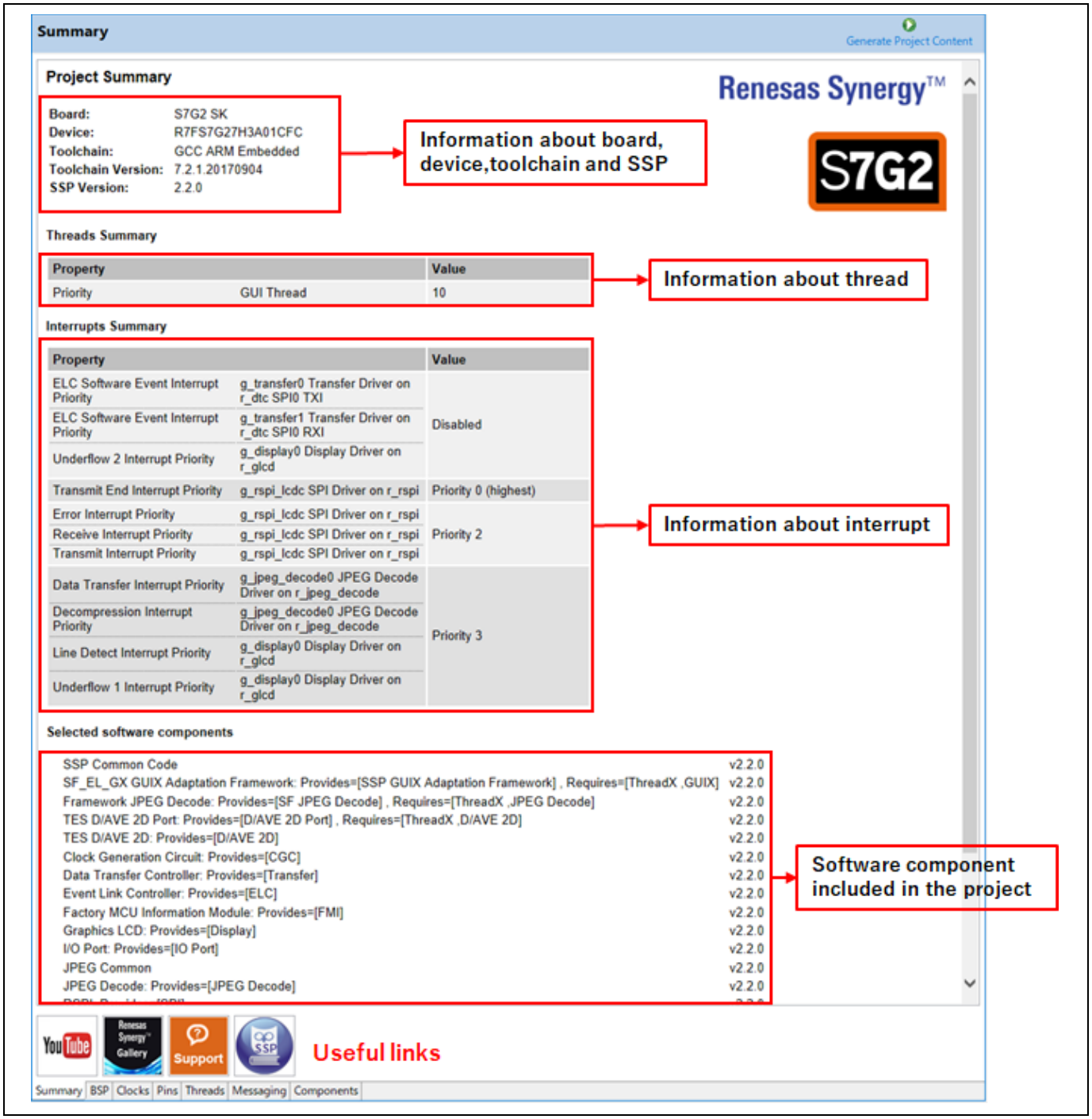


Figure 3-24. Summary Page

3.4.2 BSP Page

The **BSP** tab allows the user to select the SSP version, board, and device. The user can also import the CMSIS pack from this page.

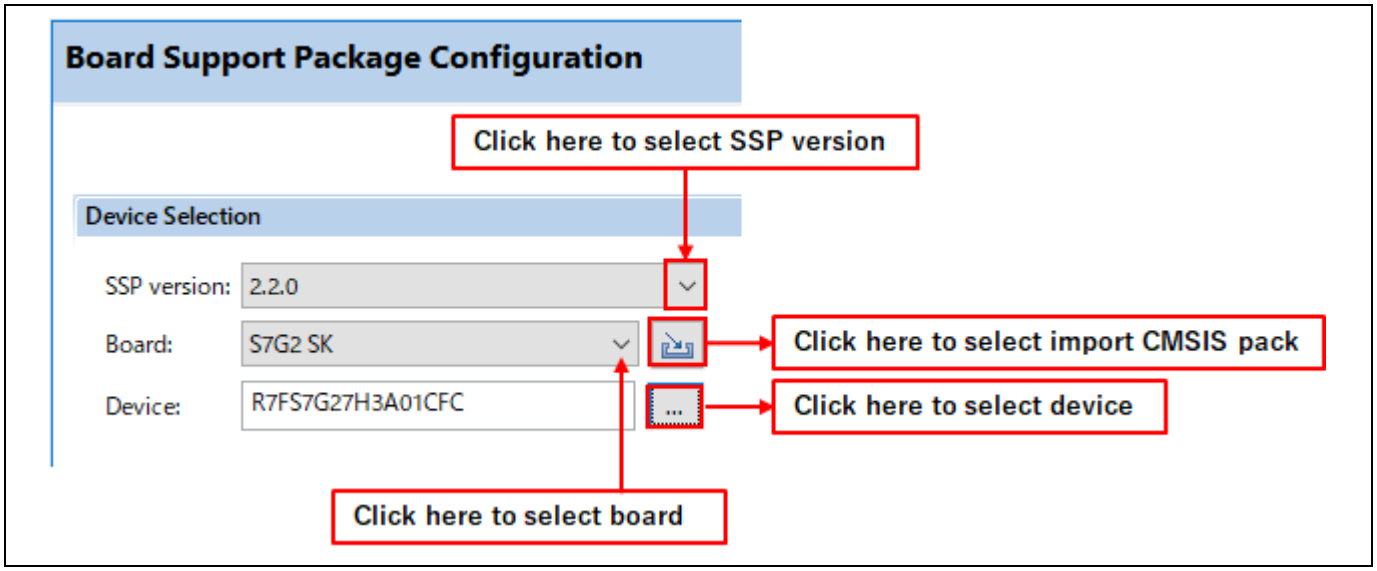


Figure 3-25. Synergy Project Configuration – BSP Page

3.4.3 Clocks Configuration Page

The **Clocks Configuration** 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 Synergy 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 the GUI.

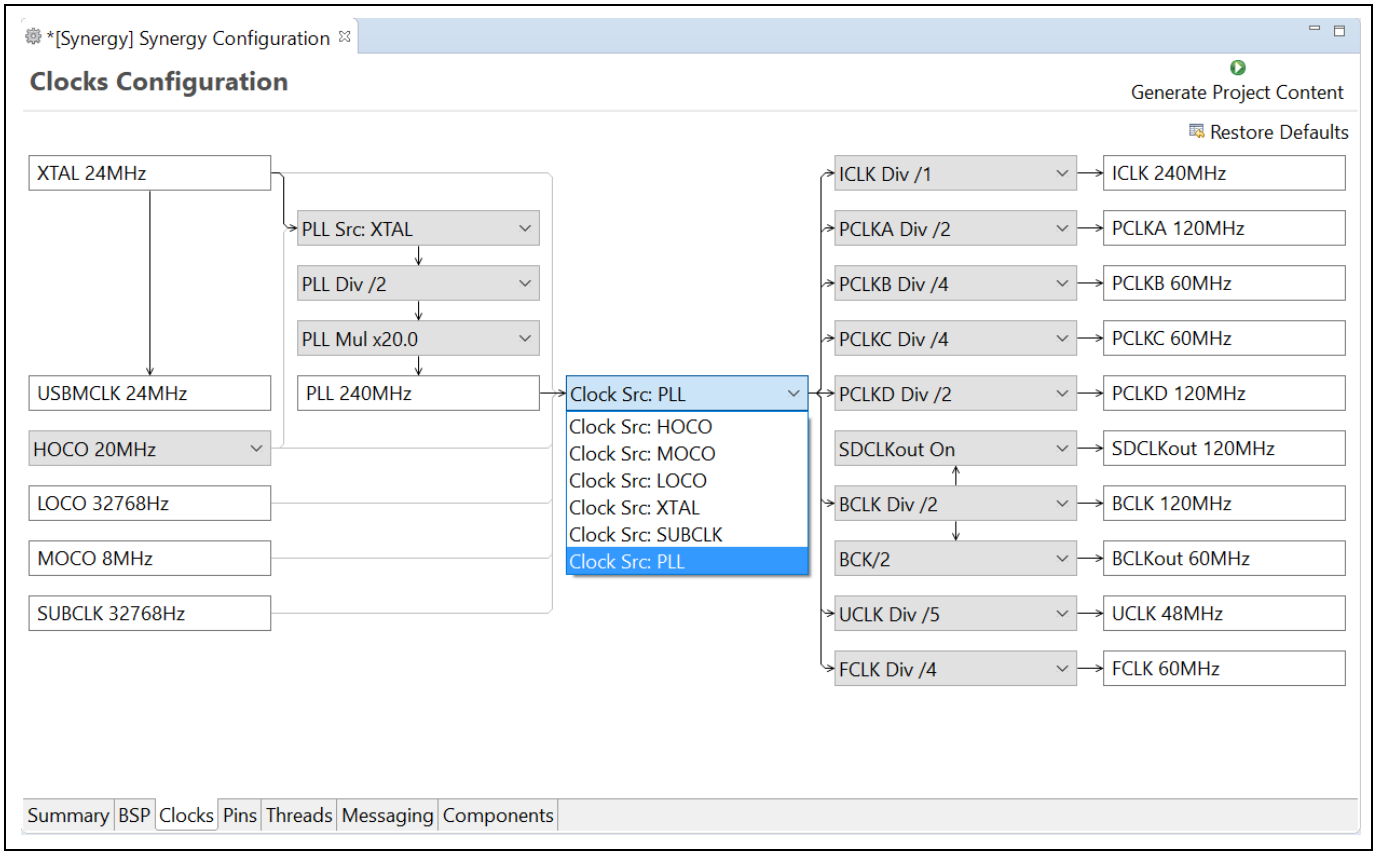
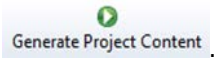


Figure 3-26. Synergy Project Configuration – Clocks Configuration

2. Save the Project Configuration Settings, for example by using the **Ctrl-S** shortcut.
3. Click the **Generate Project Content** button 

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

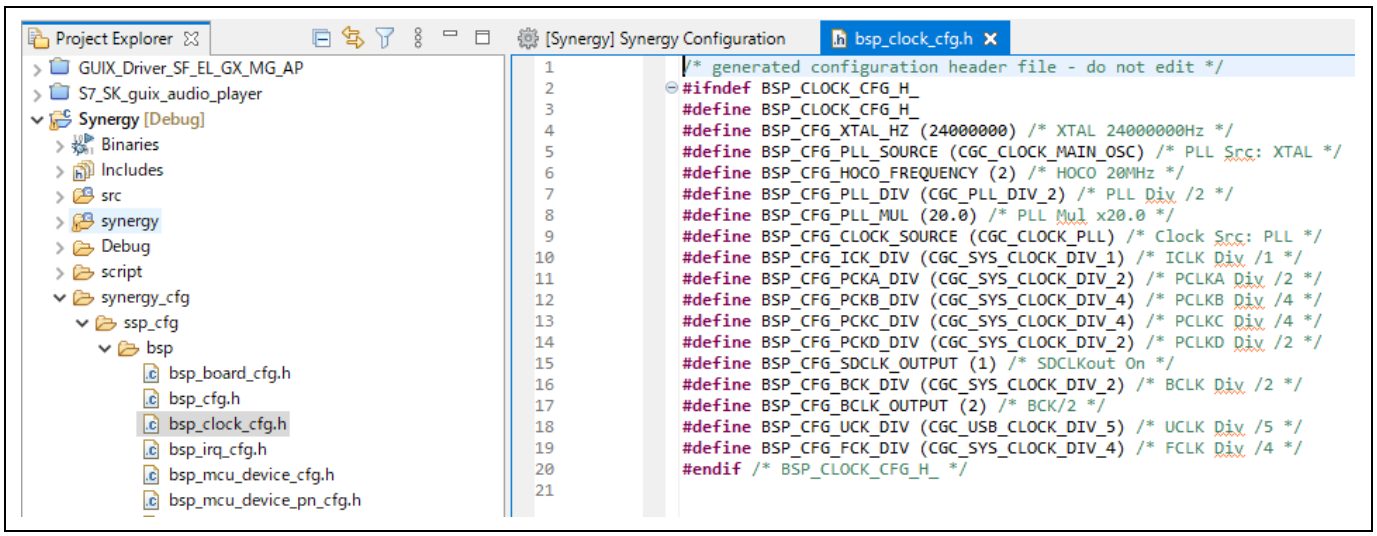


Figure 3-27. bsp_clock_cfg.h is updated

3.4.4 Pins Configuration Page

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

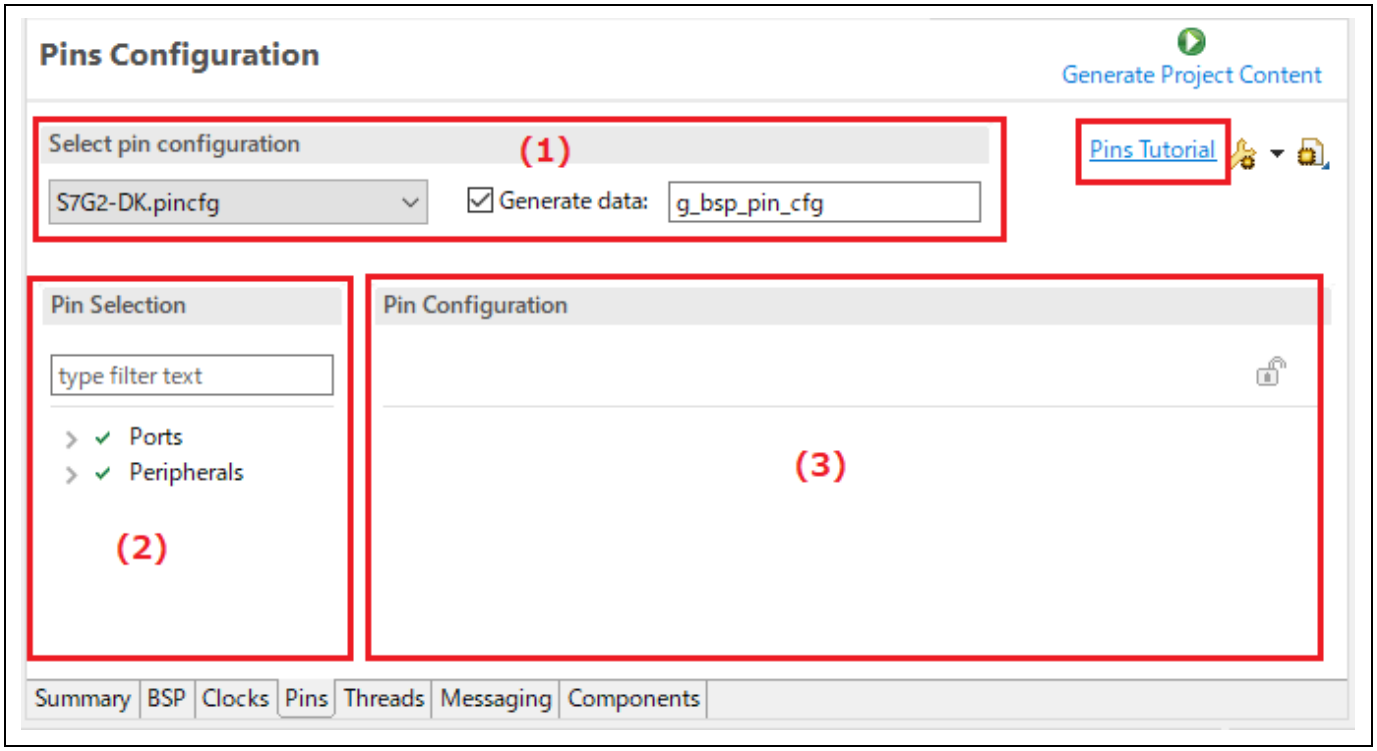


Figure 3-28. Synergy Project Configuration –Pins Configuration GUI

The 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:
   a. Create a new `.pincfg` file (for example, `NewName.pincfg`) in **Project Explorer** by copying an existing one.
   b. Select the new `.pincfg` file (for example, `NewName.pincfg`) in the **Select Pin Configuration** dialog box.
   c. Check the **Generate data** checkbox and give the new pin configuration a unique data structure name in the text field.
   d. 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.

Refer to the tutorial video for pin configuration on YouTube by clicking **Pin Tutorial**.

The best way to configure pins is to configure the peripherals to be used in the project using the steps below:

1. Select a peripheral in the **Pin Selection** pane, for example, **Connectivity:SCI → SCI1**. The configuration for this peripheral will be shown in the **Pin Configuration** pane.
2. Select an **Operation Mode** for the peripheral, for example, **Simple SPI**.
3. Select the pins you would like to use for the **Input/Output** functions of the selected peripheral in the selected mode.

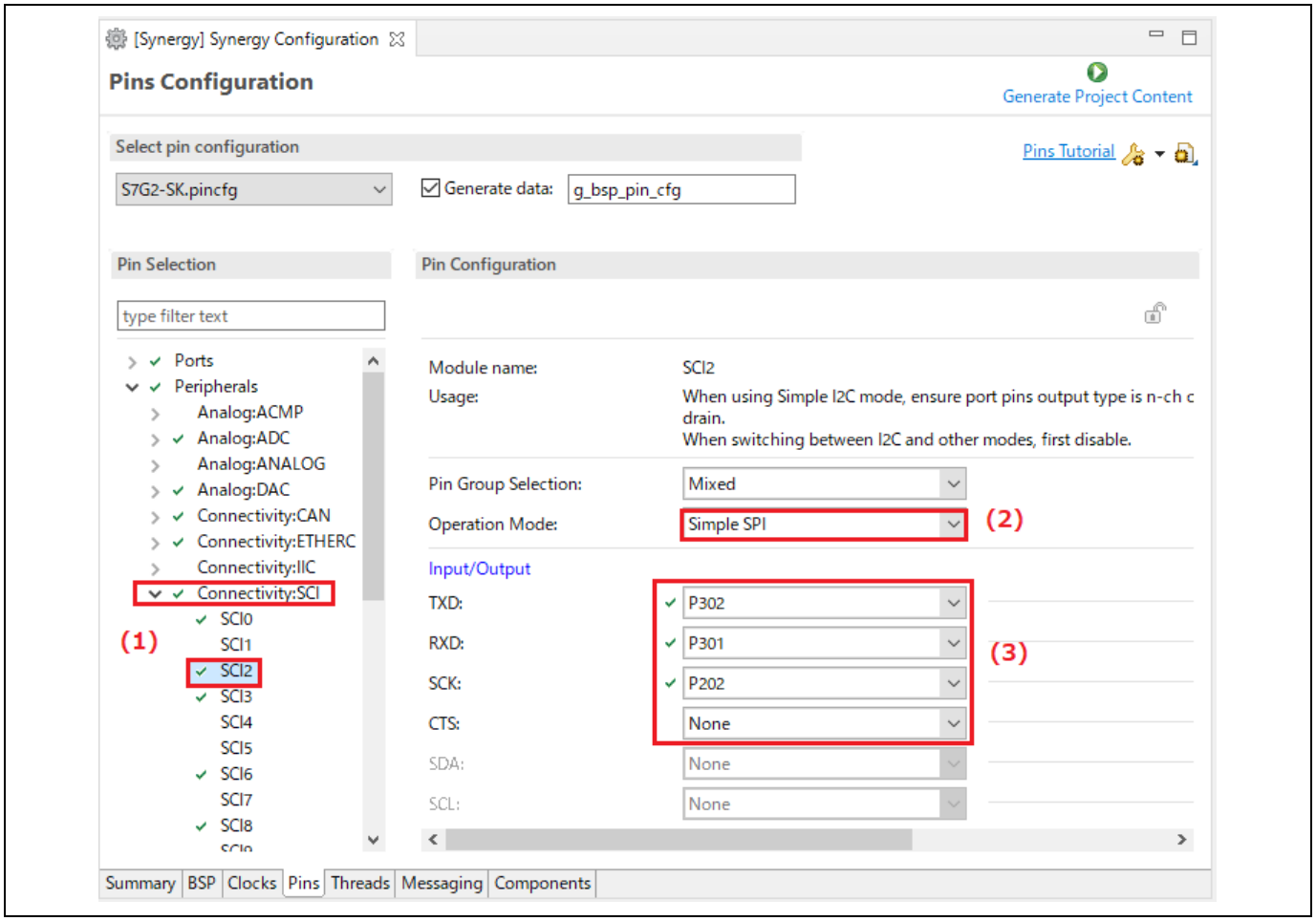


Figure 3-29. Synergy Project Configuration – Pin Configuration Setting (by Peripheral)

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

1. Select a pin in the **Pin Selection** pane, for example, **Ports** → **P0** → **P003**. The configuration for this pin will be shown in the **Pin Configuration** pane.
2. Enter properties for this pin, for example:

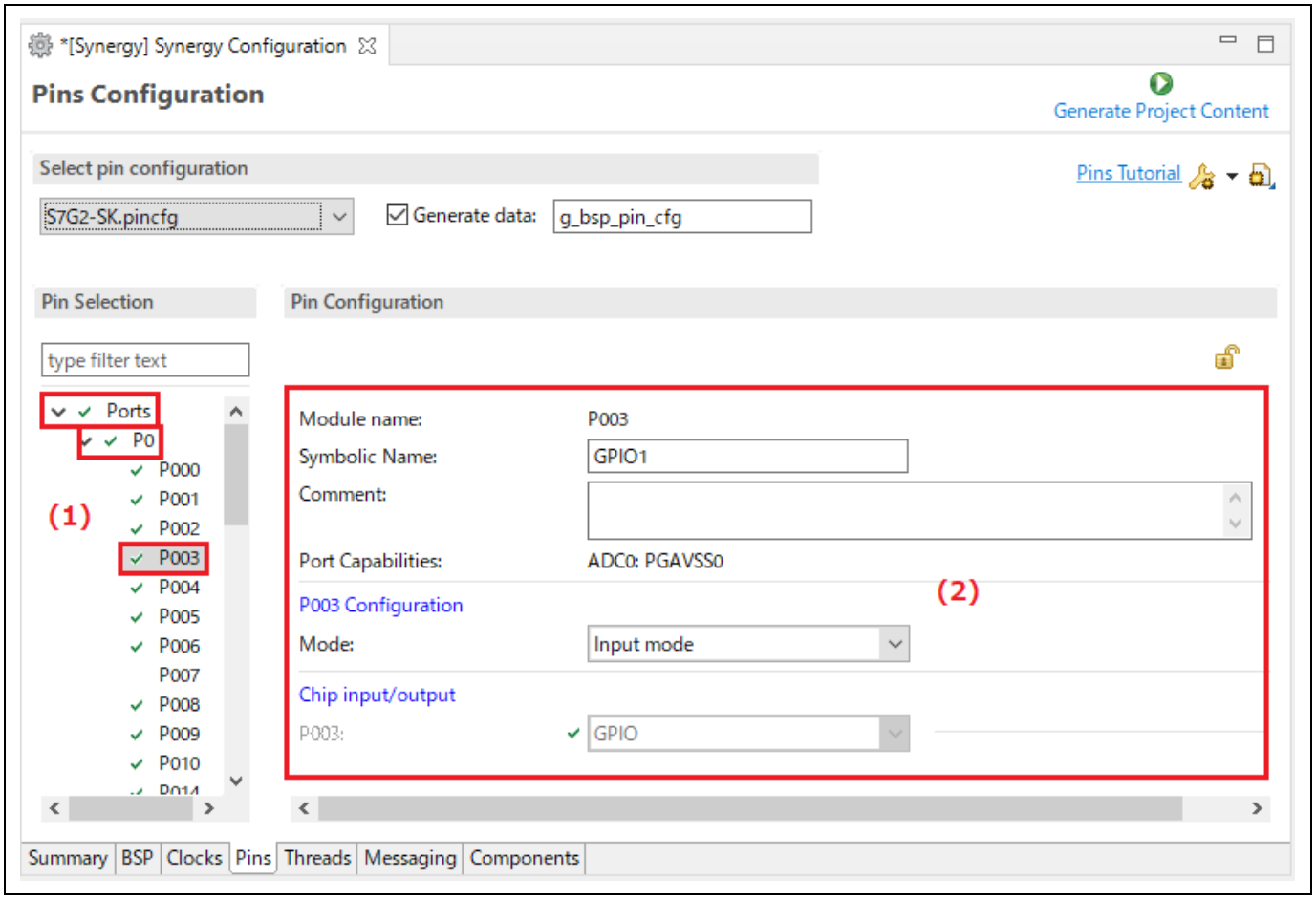


Figure 3-30. Synergy Project Configuration – Pin Configuration Setting (by single pin)

3. The **Package** view shows this pin change.

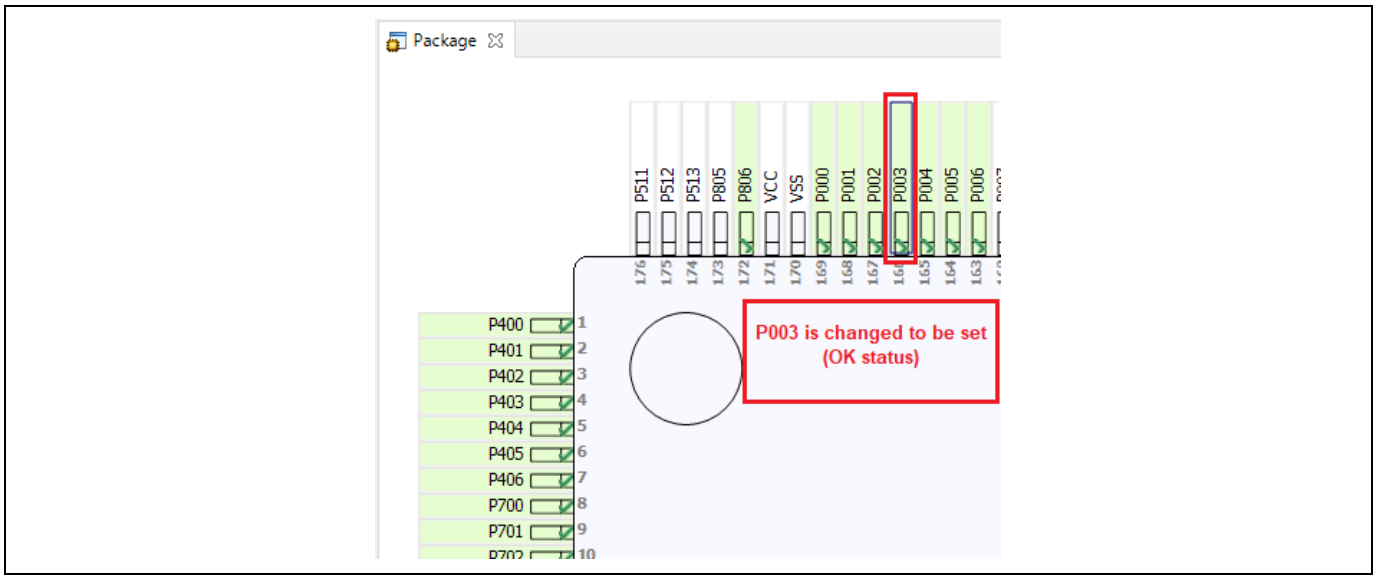


Figure 3-31. Synergy Project Configuration – Package View (Connection Status)

It is possible to migrate a pin configuration from one device to another device on this page. Use the **Import a pin configuration** button on the toolbar to perform this migration. This function allows migration of the pin configuration to the new device while retaining user setup.

To import an existing pin configuration to the current project, click **Import a pin configuration** and select the pin configuration file to import.

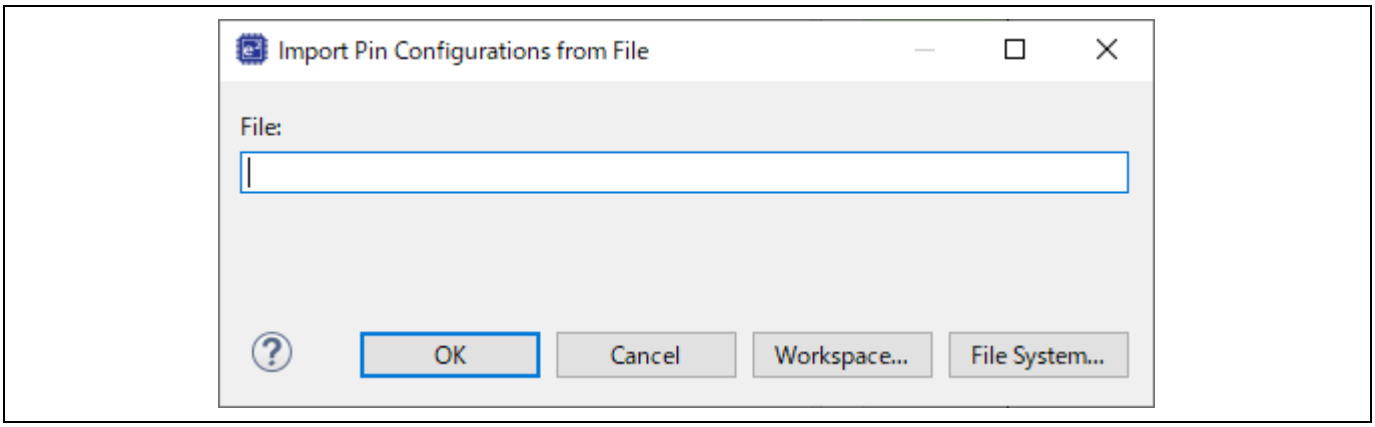


Figure 3-32. Import an existing pin configuration to the current project

The import function might point out conflicts and provide the following options for the user:

1. Cancel the import operation.
2. Ignore the conflicts and import the conflicting settings anyway.
3. Continue the import operation without importing the conflicting settings.

Note: For pin information, a `.csv` file is created in the project folder `\synergy_cfg\ssp_cfg\bsp`. No other information is output.

3.4.5 Threads Configuration Page

The **Threads Configuration** page allows users to:

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

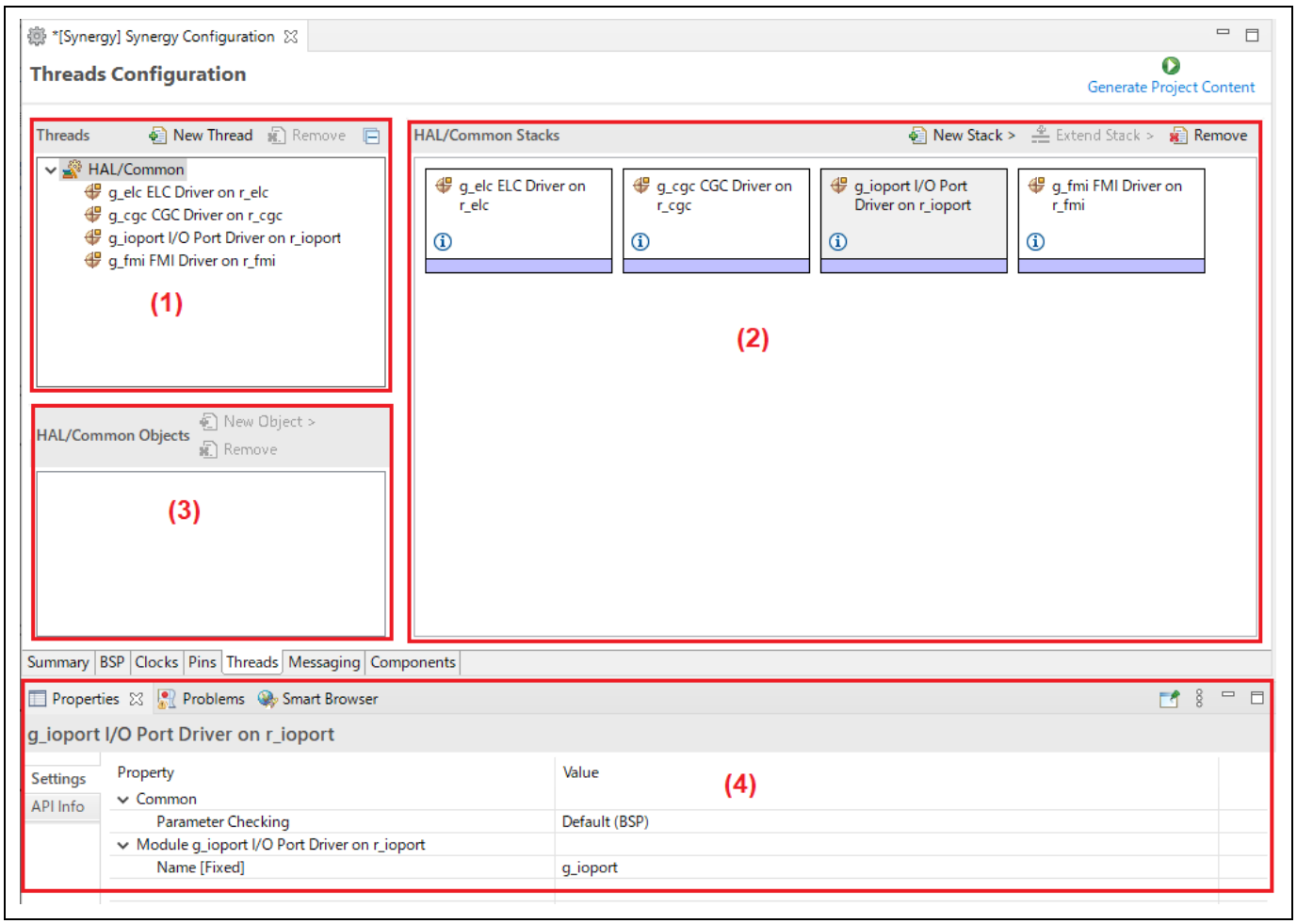


Figure 3-33. Synergy Project Configuration – Threads Configuration GUI

The **Threads Configuration** page consists of 3 panes:

1. **Threads** pane: Add/remove threads. More details are explained in Chapter 6.
2. **Stacks** pane: Add/remove SSP module instances, i.e. I/O port, SCI, UART, etc.
3. **Objects** pane: Add/remove kernel objects. More details are explained in Chapter 6.

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, such as 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, that is, **New Stack → Driver → Monitoring → Clock Accuracy Circuit Driver on r_cac**.
3. Click the **Generate Project Content** button to generate the source code content.
4. The **Properties** view shows the properties of the selected module. Users can change them according to their requirements.

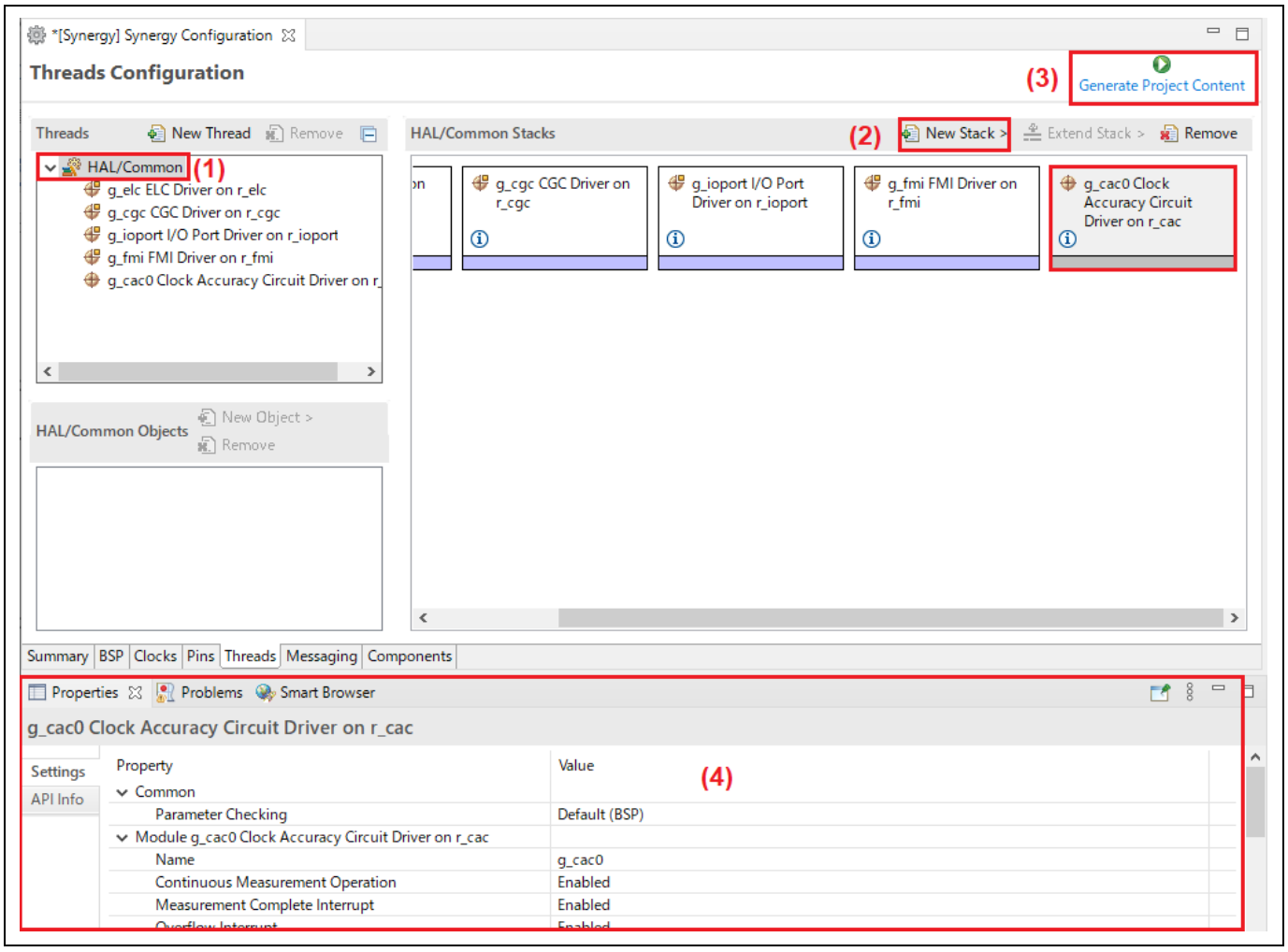


Figure 3-34. Synergy Project Configuration – Add New Module to Thread

Note: HAL/Common is not a thread. It is used to demonstrate how to add a module to the existing project in this example only. For other example, refer to chapter 6.1, General Purpose Timer Example in ThreadX. This chapter describes the procedure to add GPT module to the Blinky Thread.

An added module (for example, UART Driver on r_sci_uart) 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 to configure its properties.

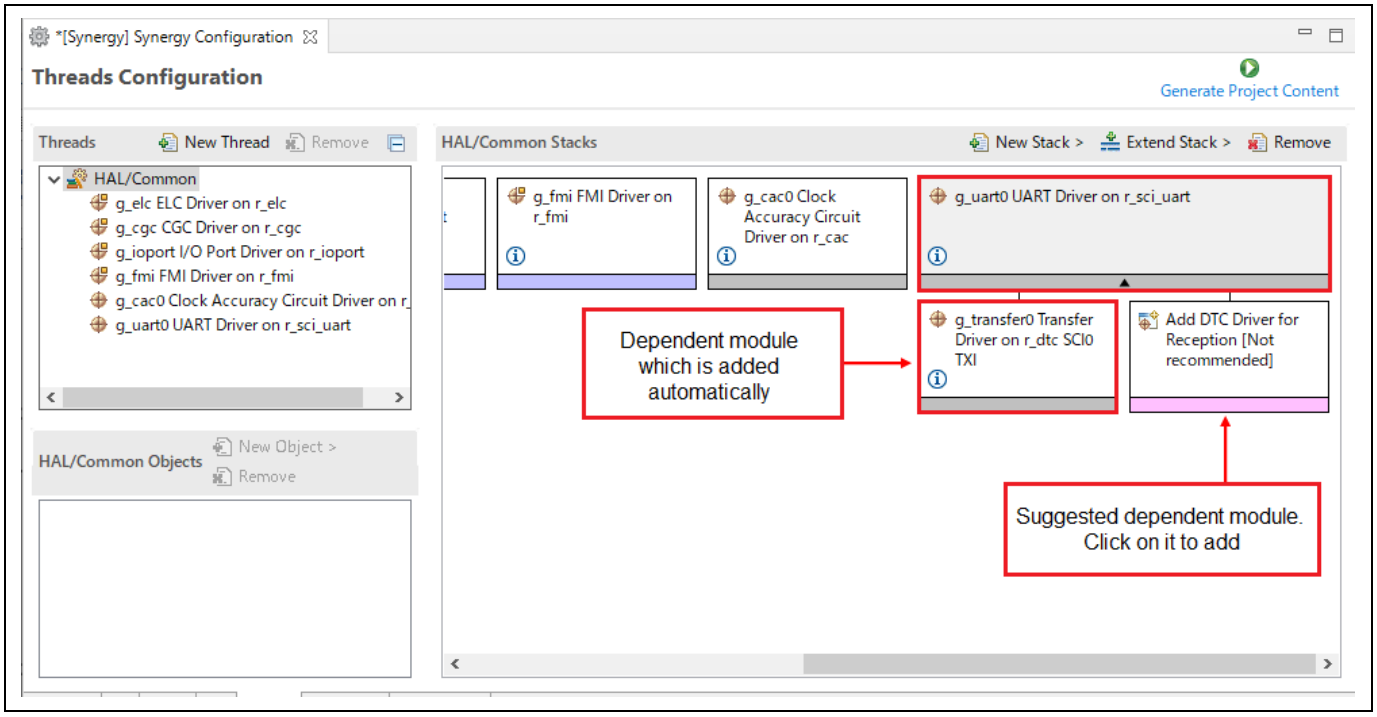


Figure 3-35. Synergy Project Configuration – Problem of Added Module

A module or a module stack can also be added by performing a copy and paste operation in the **Threads Configuration** 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.

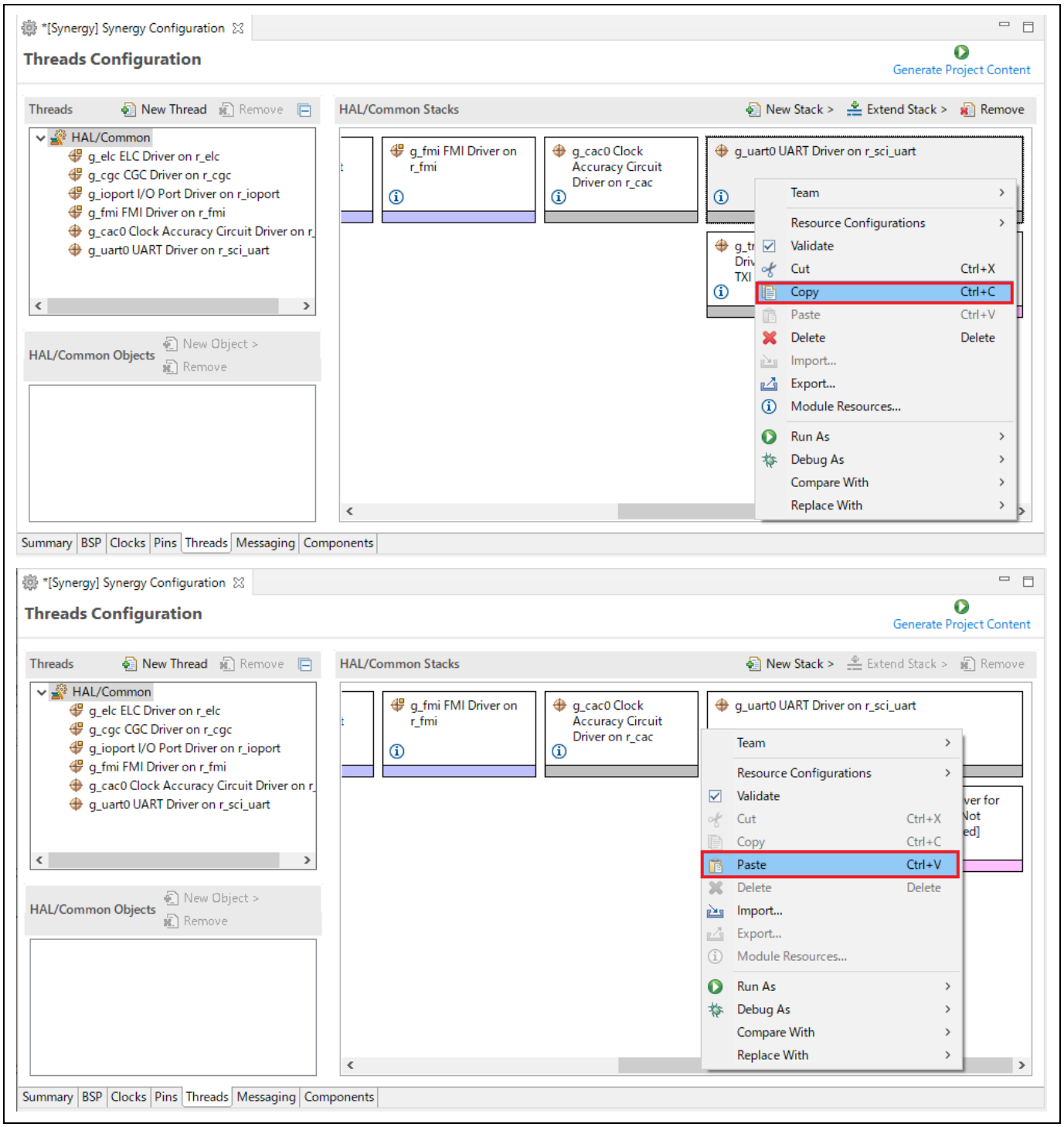


Figure 3-36. Copy and Paste operation

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.

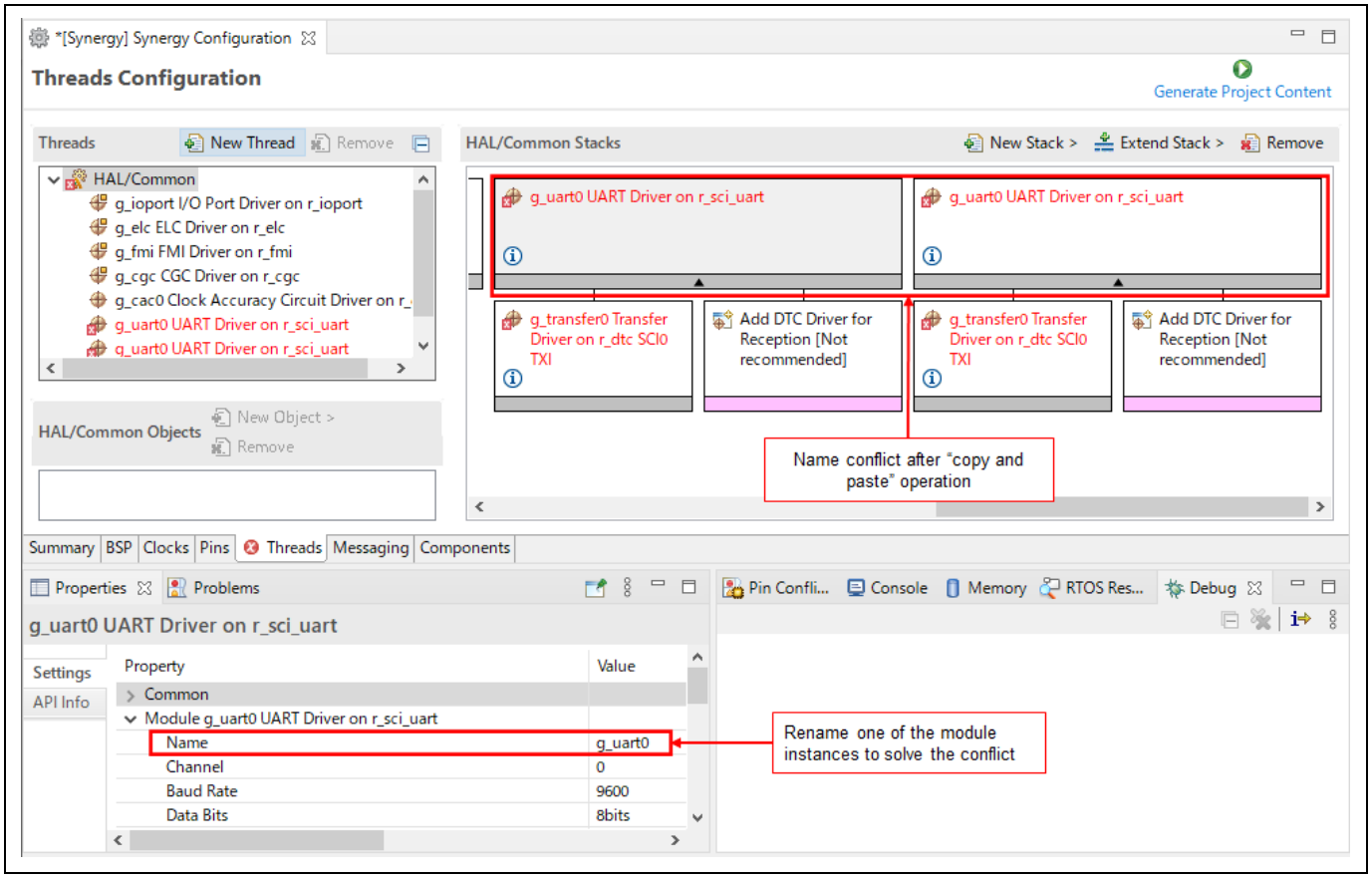


Figure 3-37. Module Instance name conflict

A module or a module stack can also be added by performing the export and import operation in the **Threads Configuration** 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. The name conflict can be solved by renaming one of the module instances.

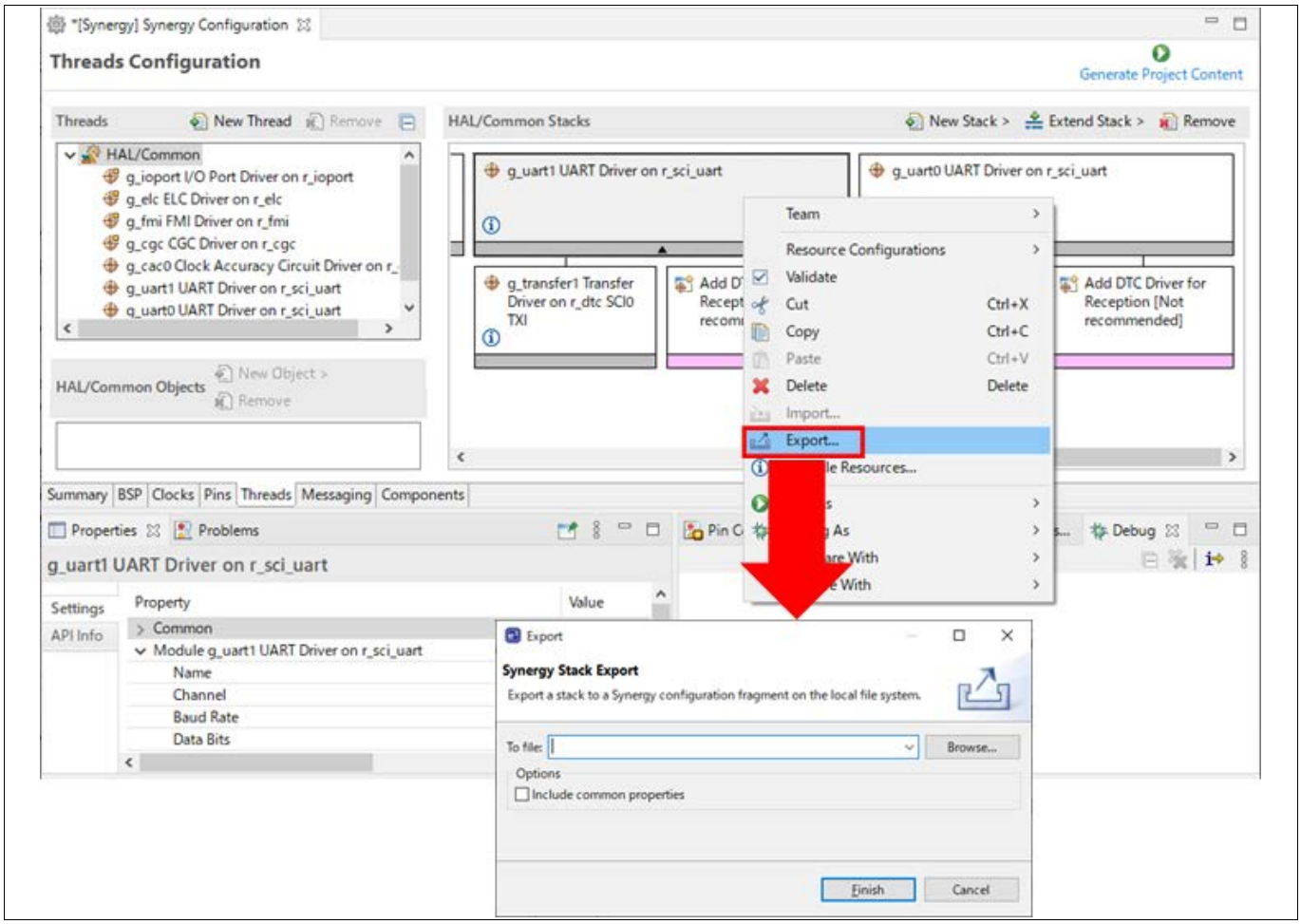


Figure 3-38. Export the Synergy stack

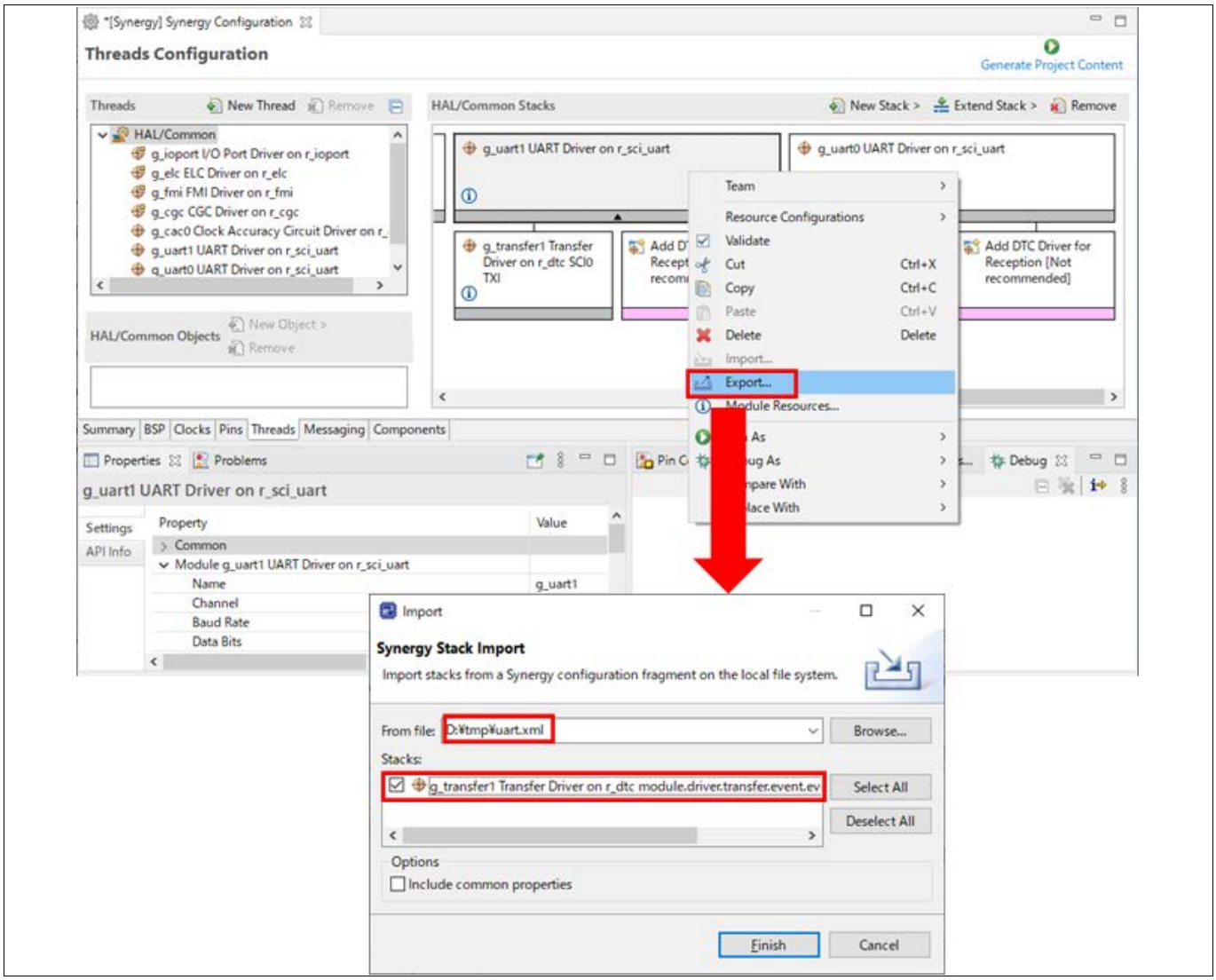


Figure 3-39. Import the Synergy stack

3.4.6 Messaging Configuration Page

The **Messaging Configuration** page allows the creation of event classes, events, and subscribers for use with the Synergy messaging framework.

The **Messaging Configuration** page consists of 3 panes:

1. The **Event Classes** pane shows a list of event classes that have been provided by instantiated Synergy modules or created manually.
2. The **Events** pane provides the events that have been provided by instantiated Synergy modules or created manually.
3. The **Subscribers** pane provides a list of subscribers that have been created. The checkbox alongside each subscriber entry indicates whether the subscriber will receive messages for the currently selected event class. A subscriber may be enabled/disabled to receive messages for the currently selected event class by checking/unchecking the checkbox.

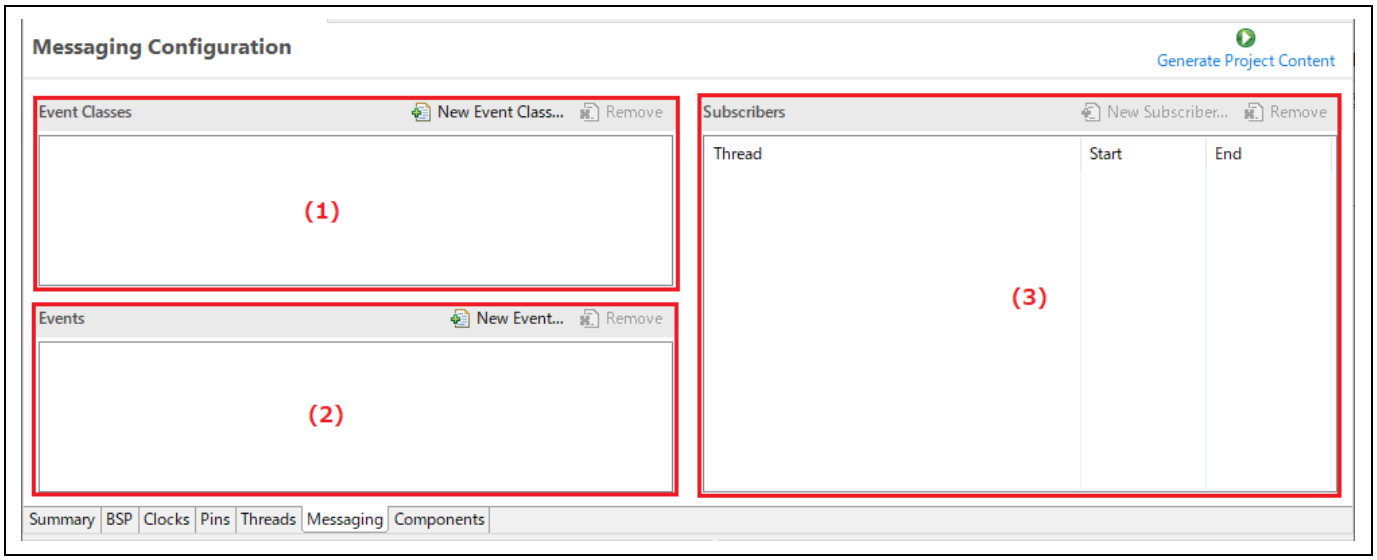


Figure 3-40. Synergy Project Configuration – Messaging Configuration Page

An event class, an event, or a subscriber can be created manually by clicking on the button of the corresponding section.

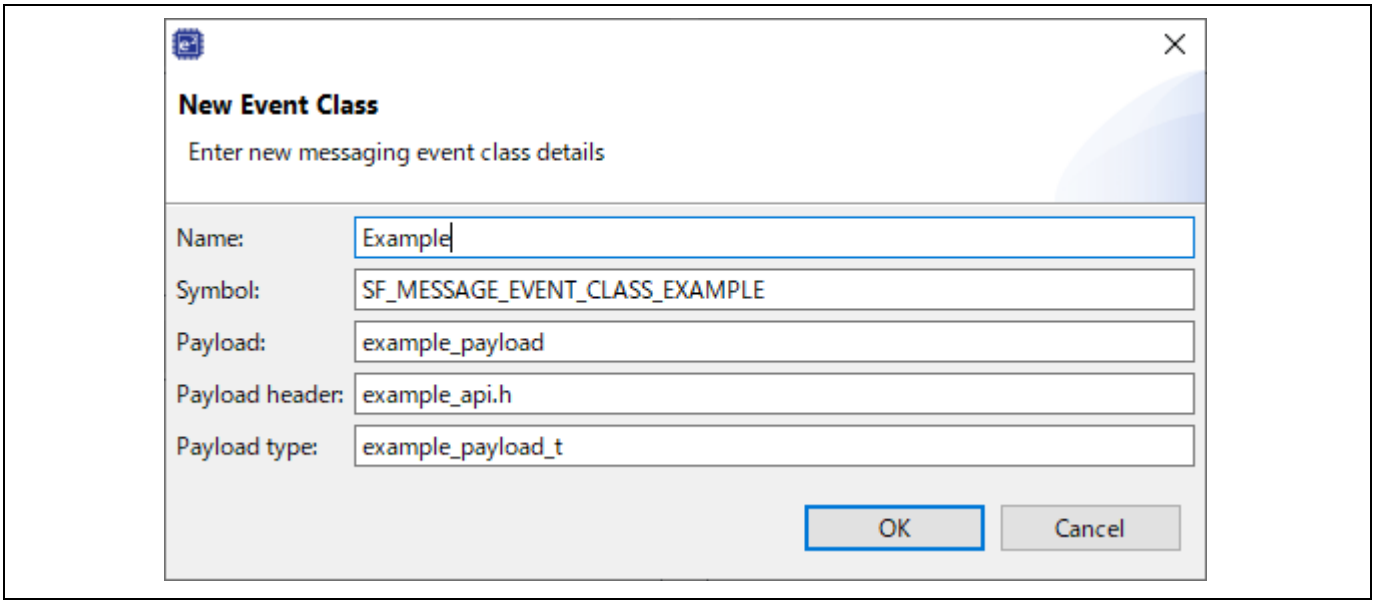


Figure 3-41. Messaging Configuration Page – Adding a new event class

To remove the item created manually, select the item and click in the corresponding section (items added by instantiated Synergy modules cannot be removed).

When a user selects an item, the e² studio **Properties** view displays the properties associated with the currently selected event class, event, or subscriber.

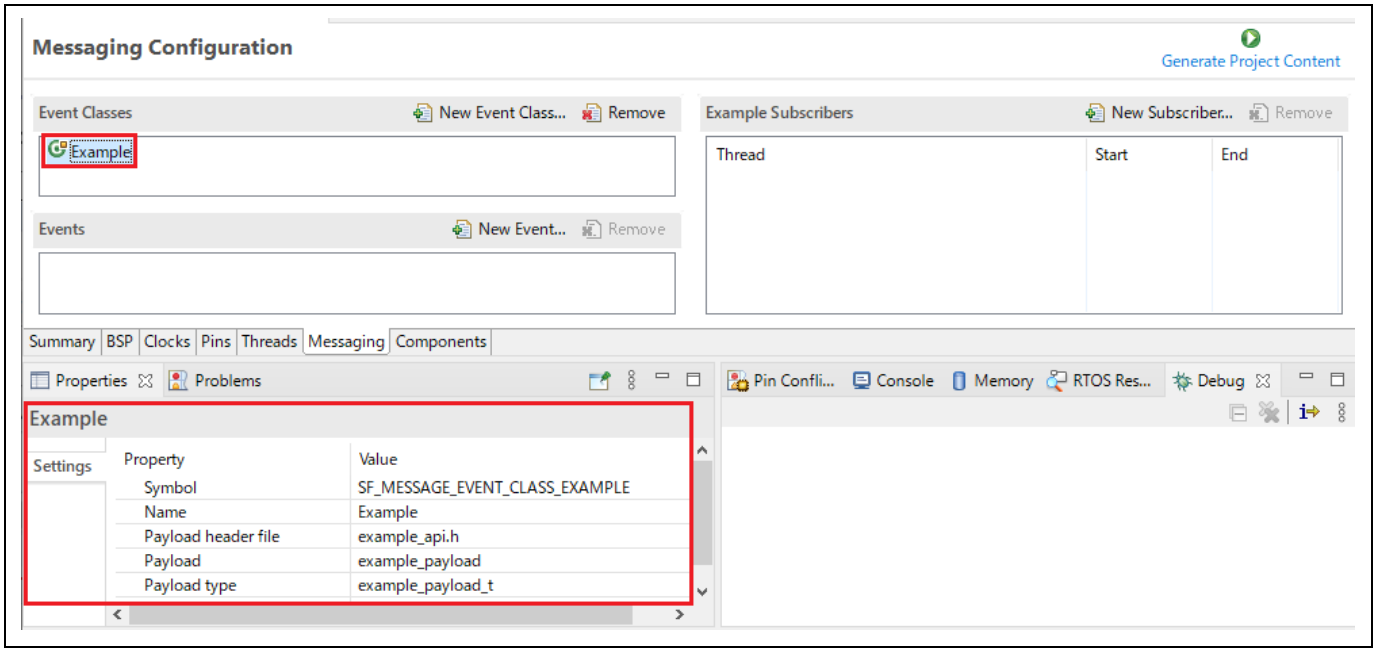


Figure 3-42. Messaging Properties

3.4.7 Components Configuration Page

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

Modules common to all Synergy projects are preselected (for example **HAL Drivers → all → r_cgc**).

All modules that are necessary for the drivers selected in the Threads 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 **Threads** page. The **Components** page is primarily used as a list of components available in the installed SSPs.

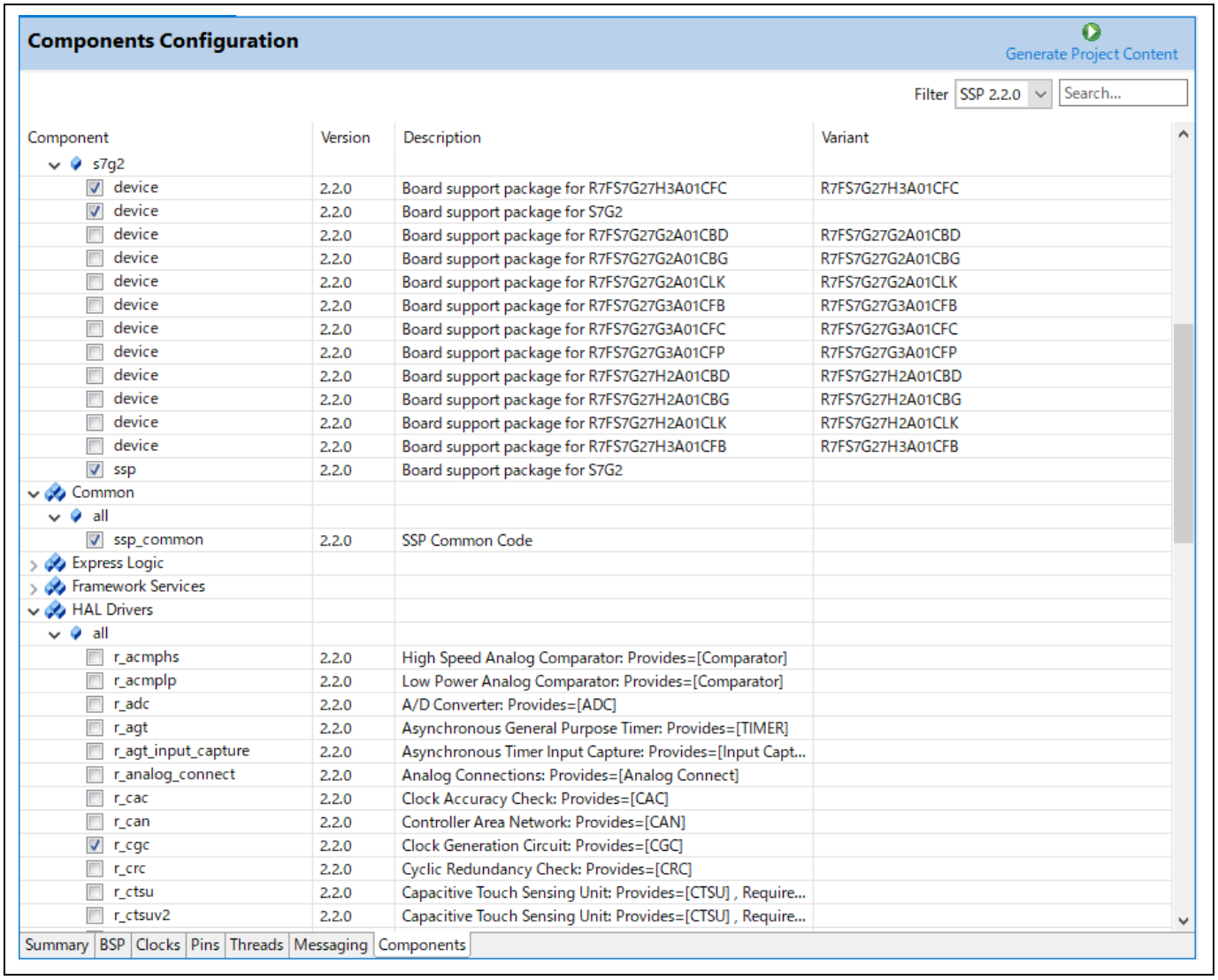


Figure 3-43. Synergy Project Configuration – Components Configuration Page

3.5 Editor hover

e² studio supports hovers in textual editor. This function can be enabled/disabled via **Window → Preferences → C/C++ → Editor → Hovers**.

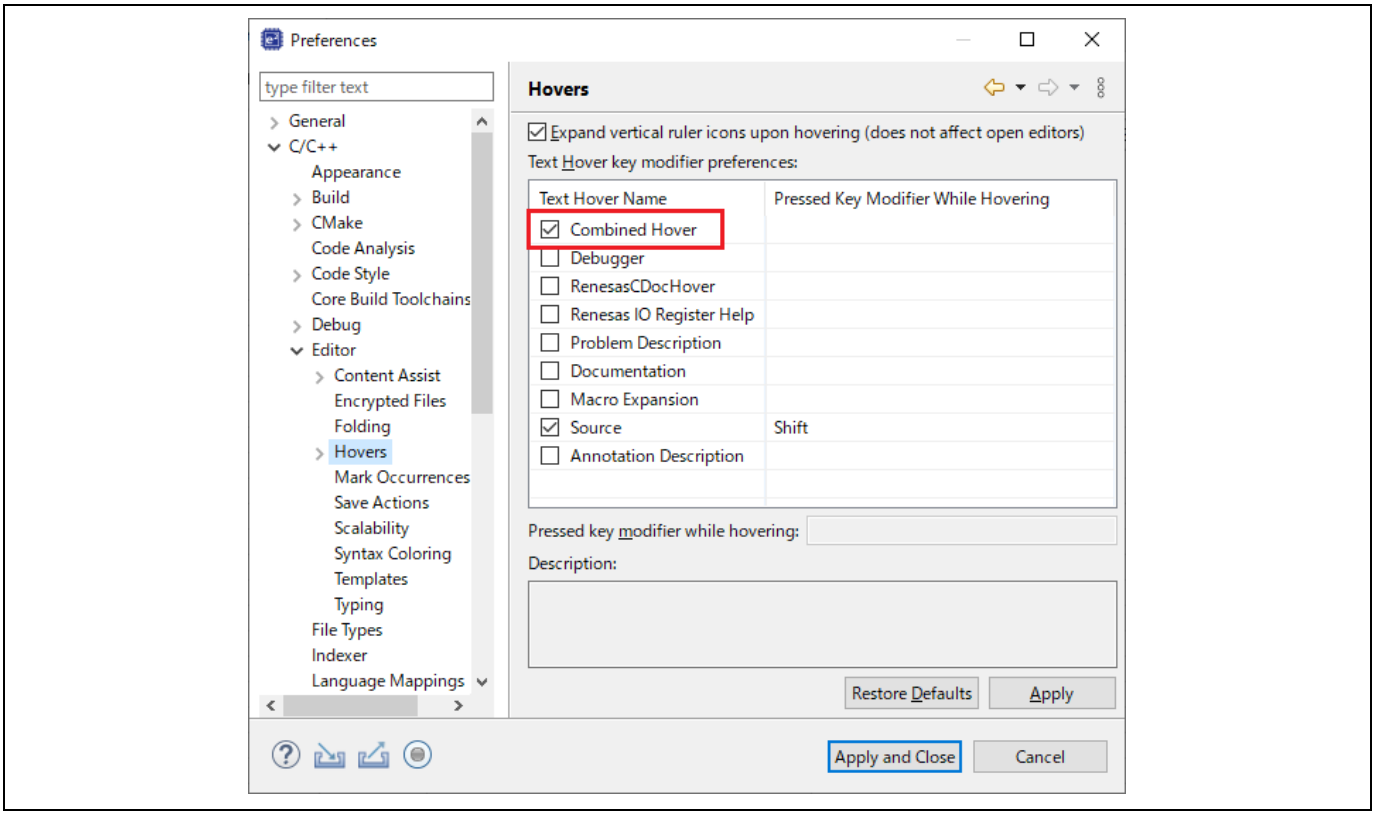


Figure 3-44. Hover settings

To enable hover, check **Combined Hover**. To disable, uncheck it. This function is enabled by default.

Hover function allows the user to view detailed information about any identifiers in the source code. Hover the mouse over an identifier and check the pop-up.

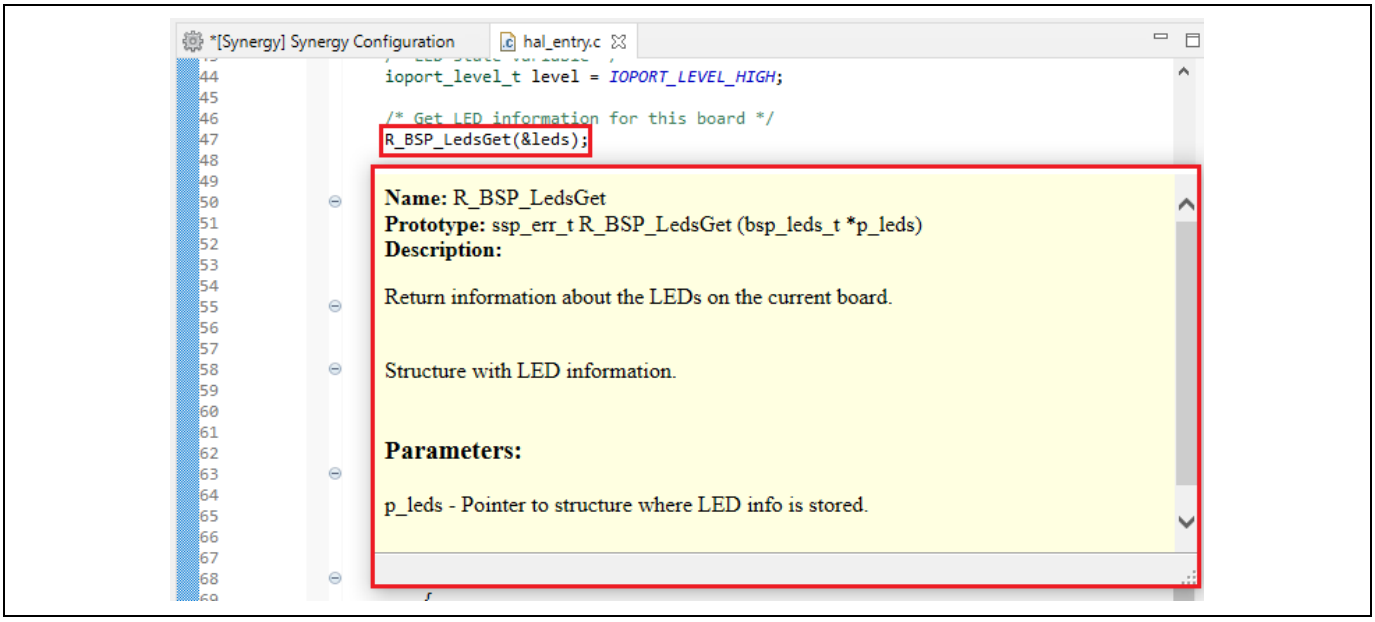


Figure 3-45. Information from hover function

3.6 Developer Assistance

Synergy Developer Assistance provides developers with module and API (Application Programming Interface) reference documentation in e² studio. After you have configured the threads and software stacks for a Synergy project with the Synergy Configuration Editor, Developer Assistance helps you quickly get started writing C/C++ application code for the project using the configured stack modules.

1. Expand the project explorer to view Developer Assistance.

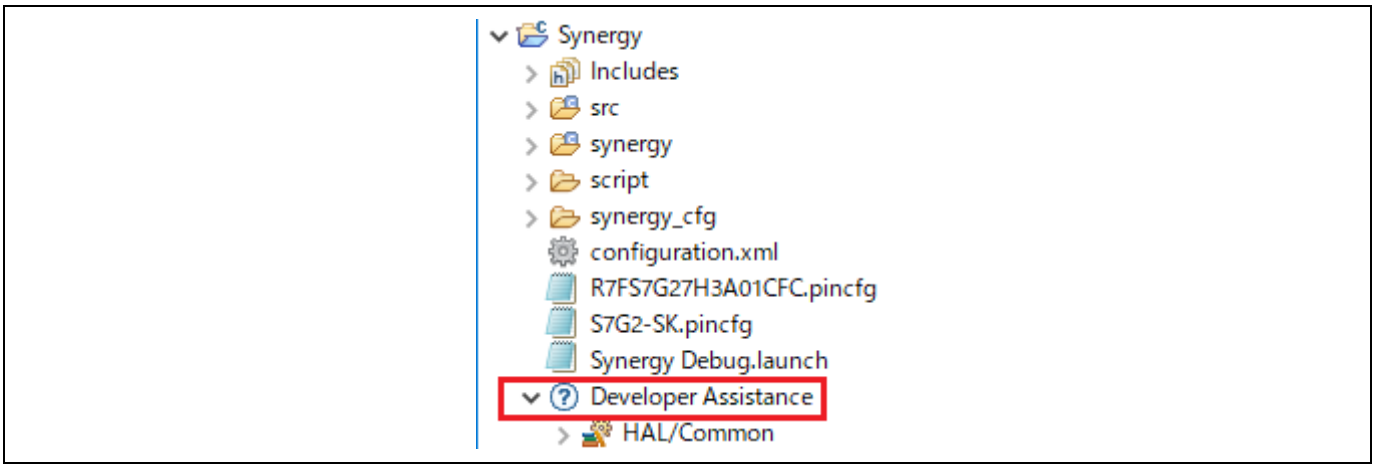


Figure 3-46. Synergy Developer Assistance

2. Expand a stack module to show its APIs.

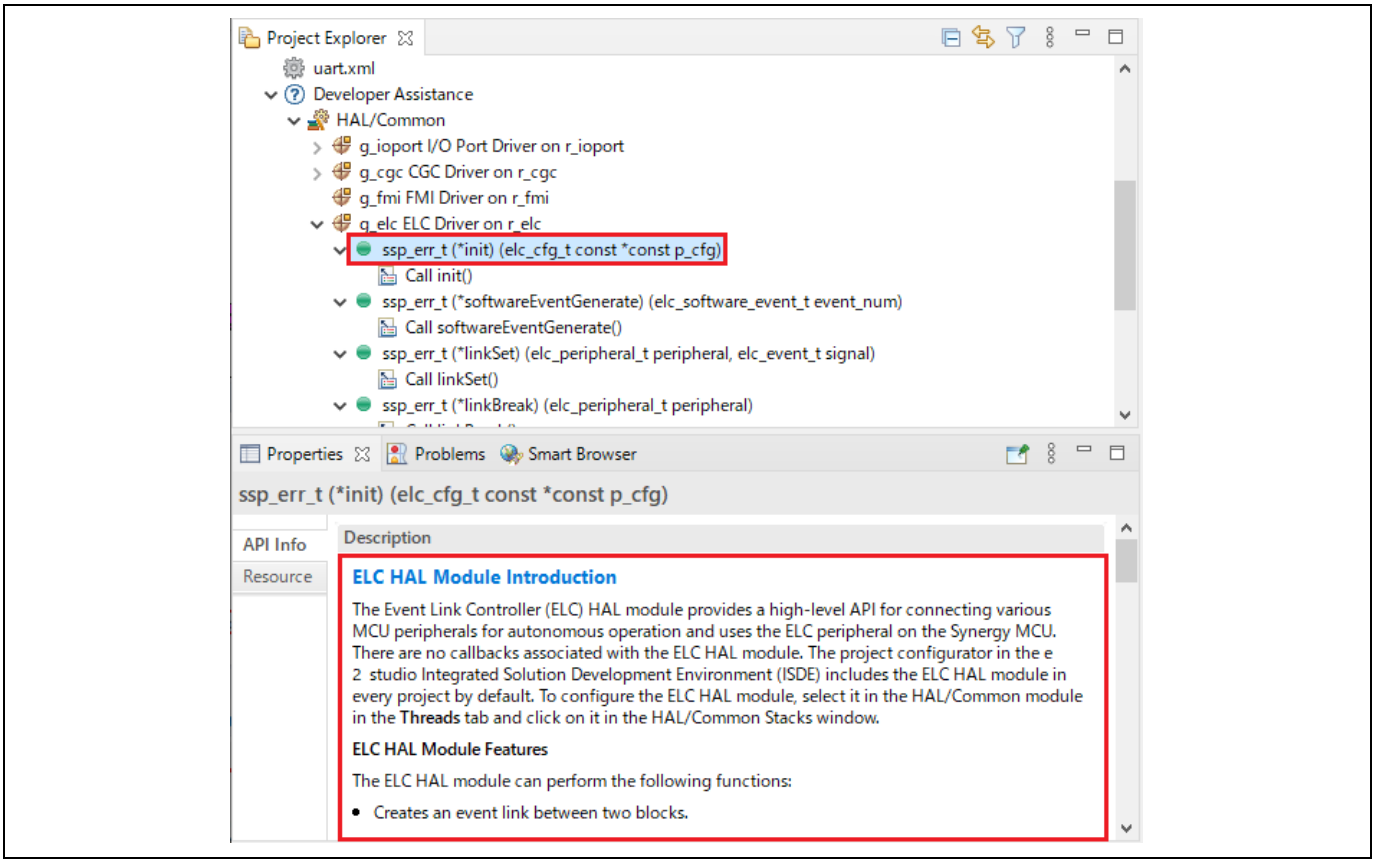


Figure 3-47. API information

3. Drag and drop an API from Develop Assistance to source file helps to write source code quickly.

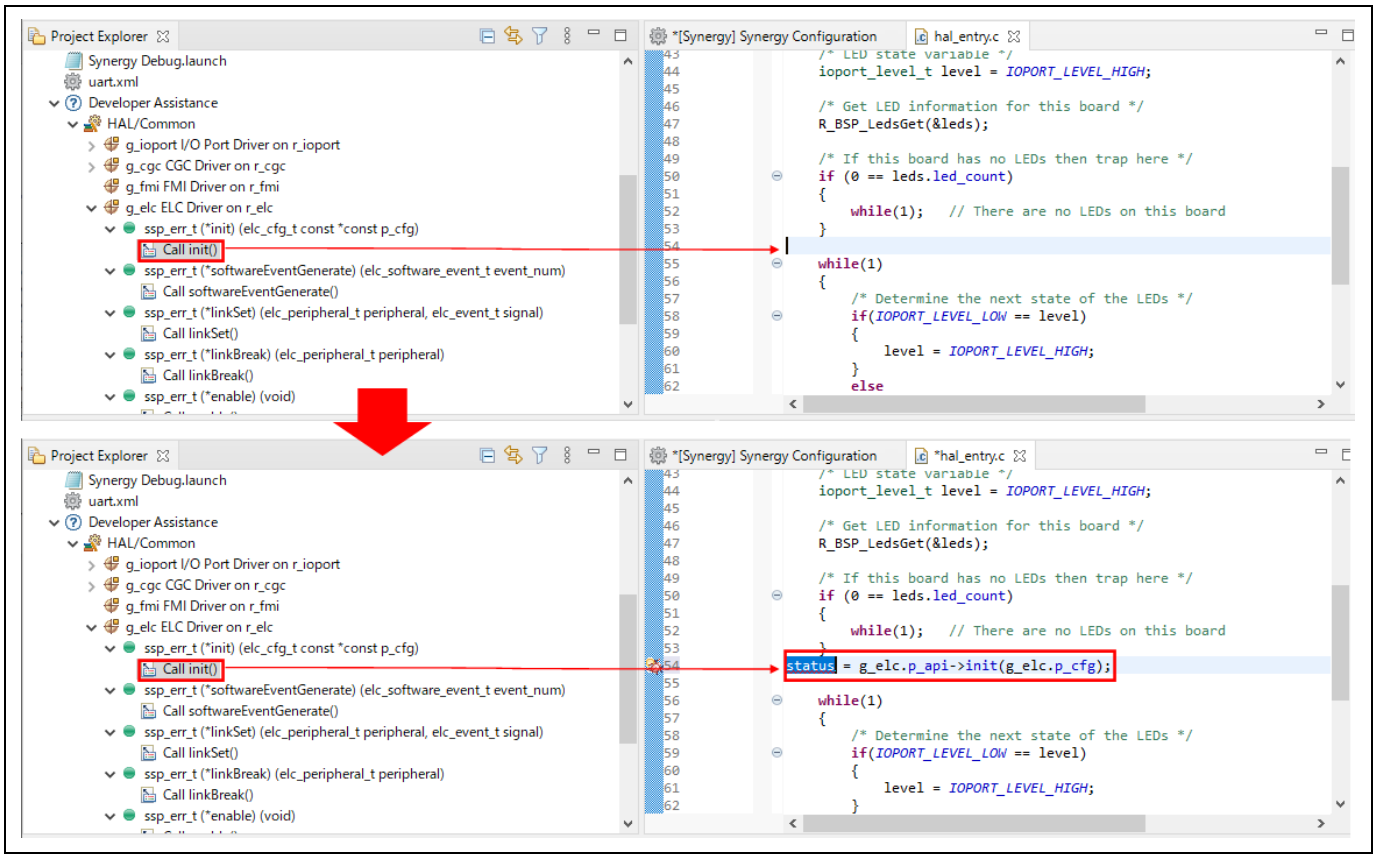


Figure 3-48. Developer Assistance – Drag and drop an API

4. Building

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

4.1 Build Configurations

The default build option is generated when a project is created, and it can usually be used to build the project.

However, if changing build options is necessary (for example, toolchain version or optimization options), please follow the following steps before building the project.

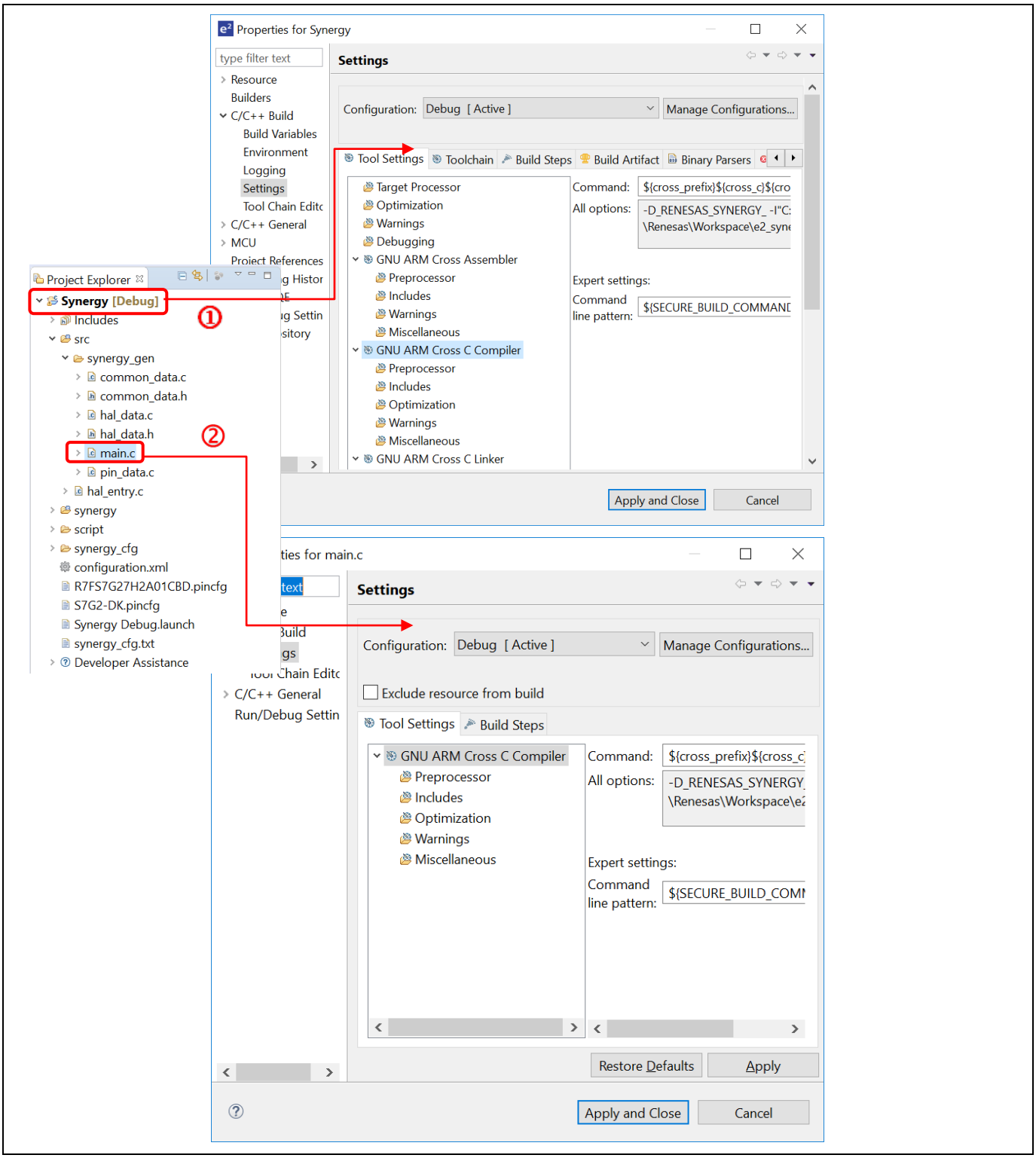


Figure 4-1. Build – Properties for Synergy Project and main.c Source File

Build options can be accessed in the properties window of a project or a source file.

1. ① Set the focus at the project name or ② set the focus at the source file name.
2. Right-click to select **Properties** or use shortcut keys **Alt+Enter** to open the properties dialog.
3. Click the **C/C++ Build** option to view or edit the configuration settings.

The **Properties** window is supported at the project and source level. The **Properties** window for projects supports more configurations which apply across all the files within the same project.

4.2 Building a Sample Project

Follow the steps below to build the project.

1. In Project Explorer, click the Synergy project to bring it into focus.
2. Click Project → Build Project or the  icon to build this project.
3. Confirm that there are no errors after build is finished.

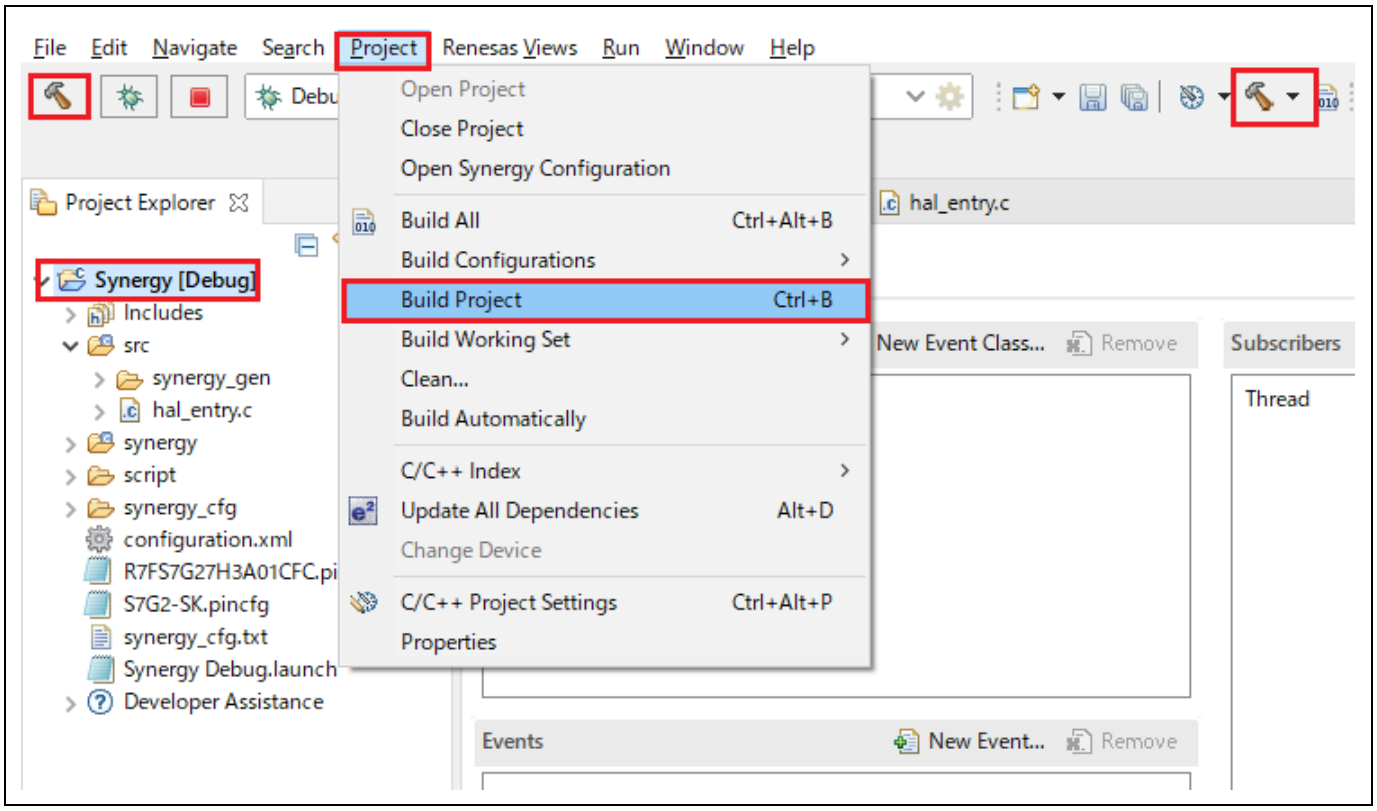


Figure 4-2. Build – Building a Sample Project

4.3 Saving the Build Settings Report

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

1. Right-click on the project in the **Project Explorer** view to pop up the context menu.
2. Select **Renesas C/C++ Project Settings-Save build settings report** to save the build settings report.

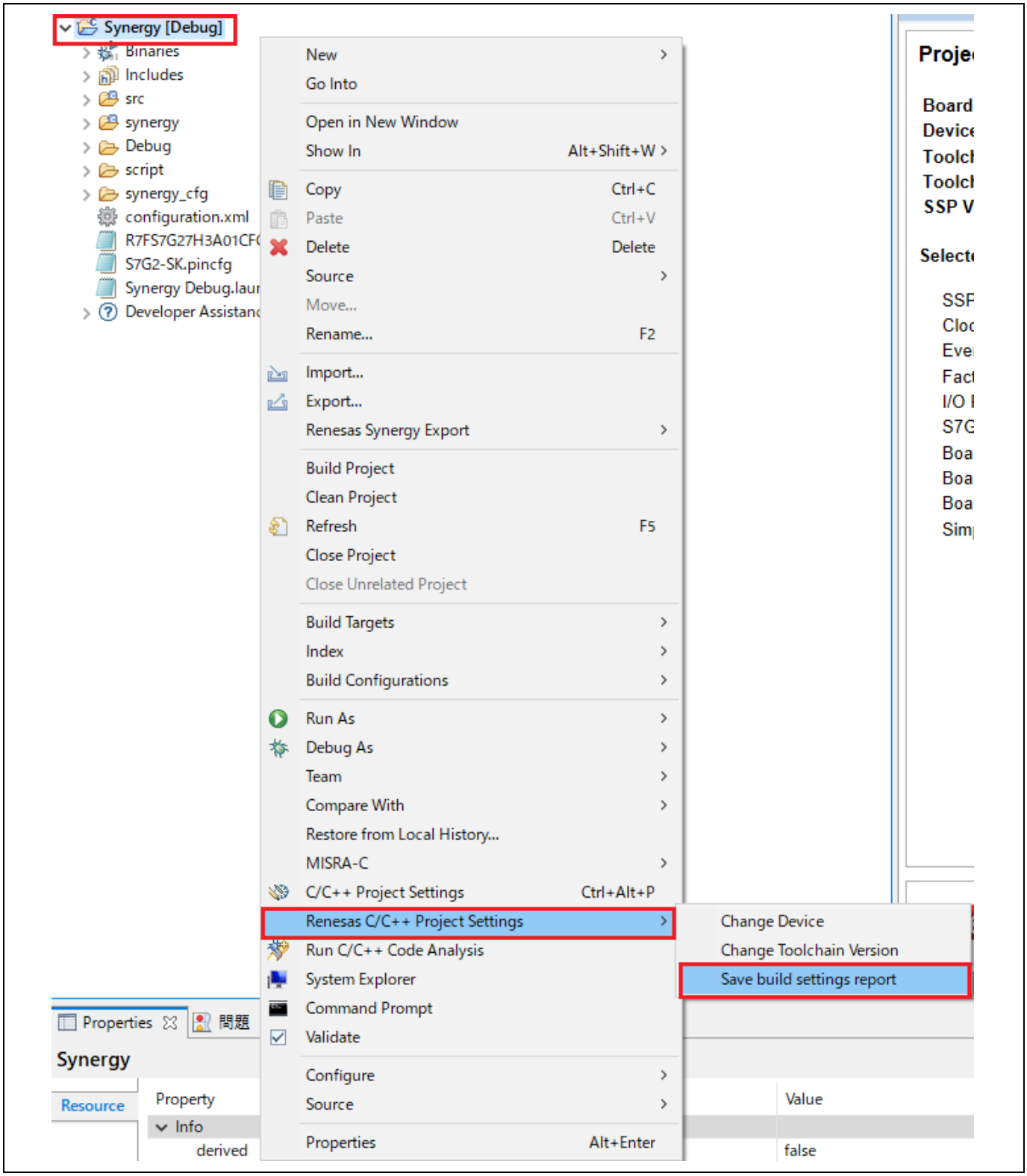


Figure 4-3. Build – Saving the build settings report

5. Debugging

This chapter describes the usage of debug configuration and key debugging features for e² studio. The following illustration refers to the “Synergy” project built in section 4.2,

Building a Sample Project, and based on the following hardware configuration: J-link ARM emulator and Synergy S7G2 DK board.

Debugging of ThreadX projects using TraceX is not included in this chapter. Please refer to the TraceX User’s Manual available in Microsoft website https://docs.microsoft.com/en-us/azure/rtos/tracex/about-this-guide for information on debugging ThreadX based projects using TraceX.

Right-click on any perspective icon, select **Show Text** to show the name of each icon.

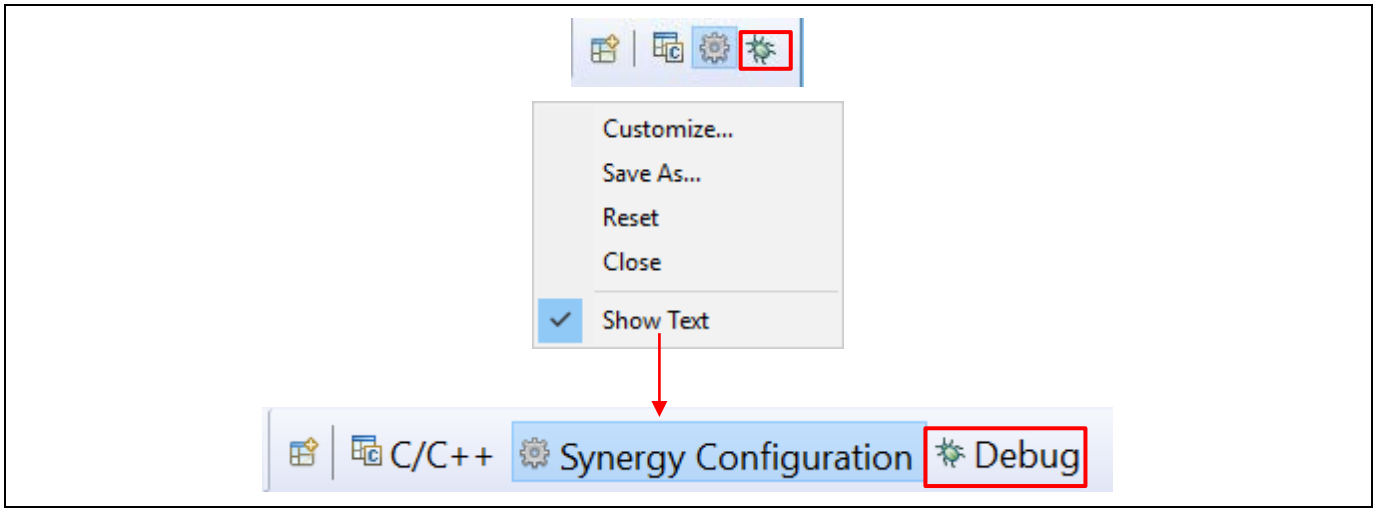


Figure 5-1. Debug – Switch to Debug Perspective

Open the Synergy project in e² studio and click **Debug** to switch to the **Debug** perspective.

As discussed earlier, a Perspective in Eclipse defines the layout of panes and views in the **Workbench** window. Each perspective consists of a combination of views, menus and toolbars that enable the user to perform a specific task. For instance,

- The **Debug** perspective has views that enable the user to debug the program.
- The **Synergy Configuration** perspective together with `configuration.xml` in the editor window will open the Synergy 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. The user can customize them or add new perspectives.

5.1 Changing an Existing Debug Configuration

A default debug configuration is automatically created the first time a specific Synergy project is built. An existing debug configuration can be changed as follows.

1. Click the project name in the **Project Explorer** view to set focus.
2. Click **Run** → **Debug Configurations…** or the  icon (downward arrow) → **Debug Configurations…** to open the **Debug Configurations** window.

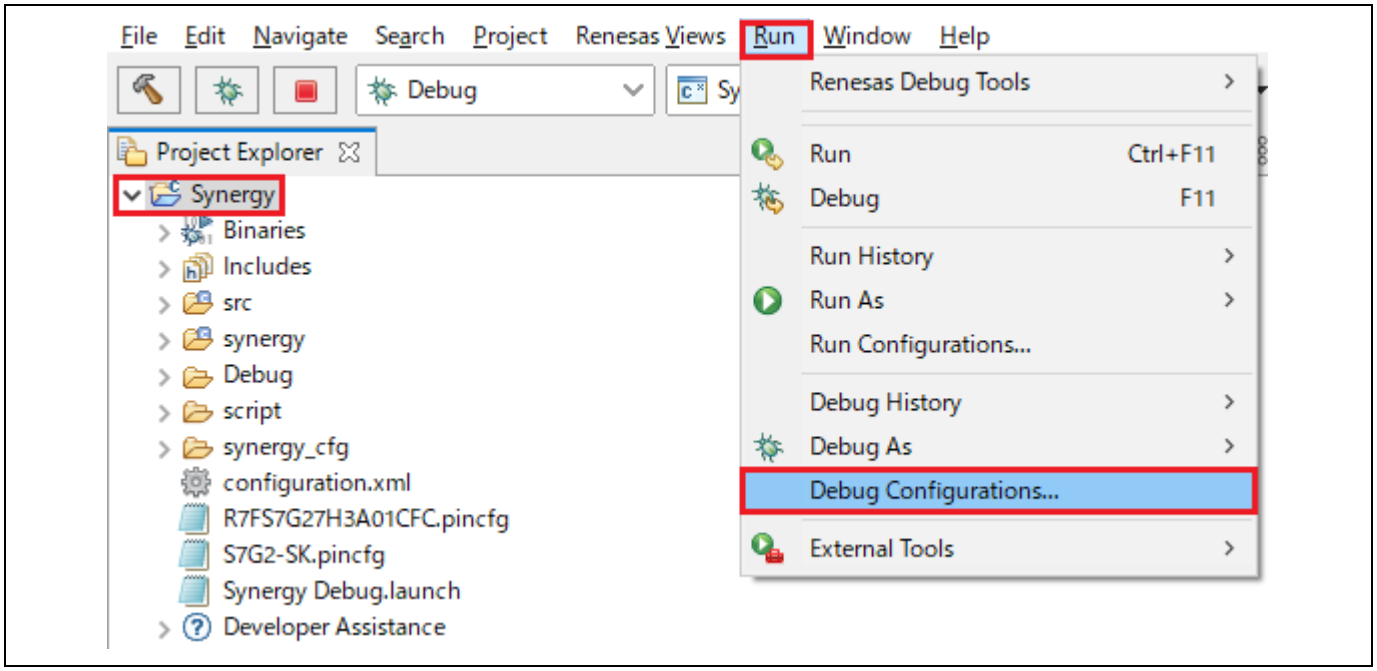


Figure 5-2. Debug – Opening the Debug Configurations Window

3. In the **Debug Configurations** windows, expand the **Renesas GDB Hardware Debugging** debug configuration and click on the existing debug configuration (for example, **Synergy Debug**).
4. Go to the **Main** tab and browse to add the load module (that is, `Synergy.elf`) located in the project build folder.

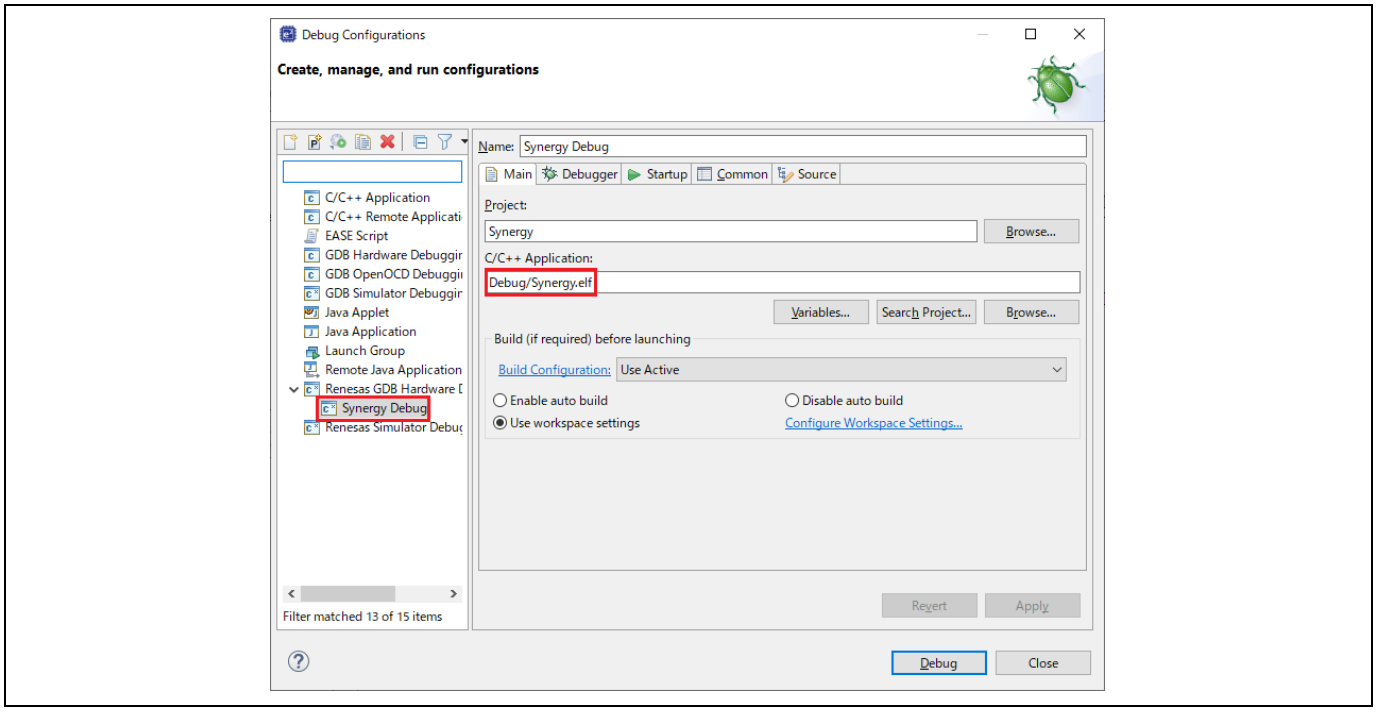


Figure 5-3. Debug – Selecting the Load Module

5. Switch to the **Debugger** tab, set **J-Link ARM** and **R7FS7G2** as the target device.
   - Debug Hardware: "**J-link ARM**"
   - Target Device: "**R7FS7G27H**"
6. Click the **Apply** button to confirm the settings.
7. Click the **Debug** button to execute the debug launch configuration to connect to the J-Link and the Synergy board.

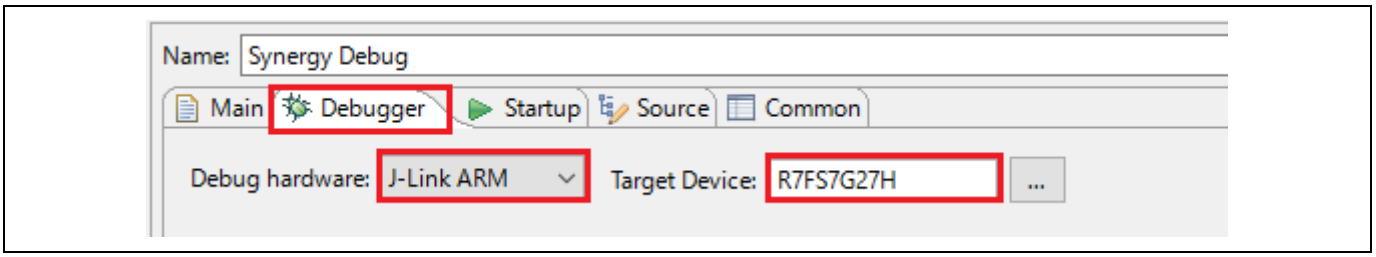


Figure 5-4. Debug – Changing the Connection Settings

8. For a successful connection, the **Debug** view shows the target debugging information in a tree hierarchy. The program entry point is set at `Reset_Handler()` in `startup_S7G2.c`.

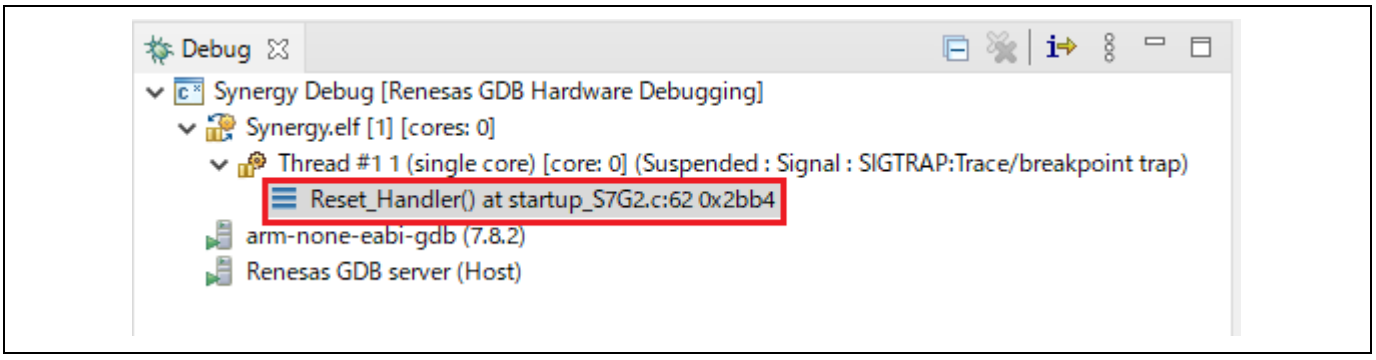


Figure 5-5. Debug – User Target Connection in the Debug view

5.2 Creating a New Debug Configurations

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

1. Open the **Debug Configurations** window (refer to Figure 5-2. Debug – Opening the Debug Configurations Window).
2. In the **Debug Configurations** window, select a debug configuration (for example, **Synergy Debug**) and click the  icon (which duplicates the currently selected launch configuration). A new debug launch configuration (for example, **Synergy Debug (1)**) is created.

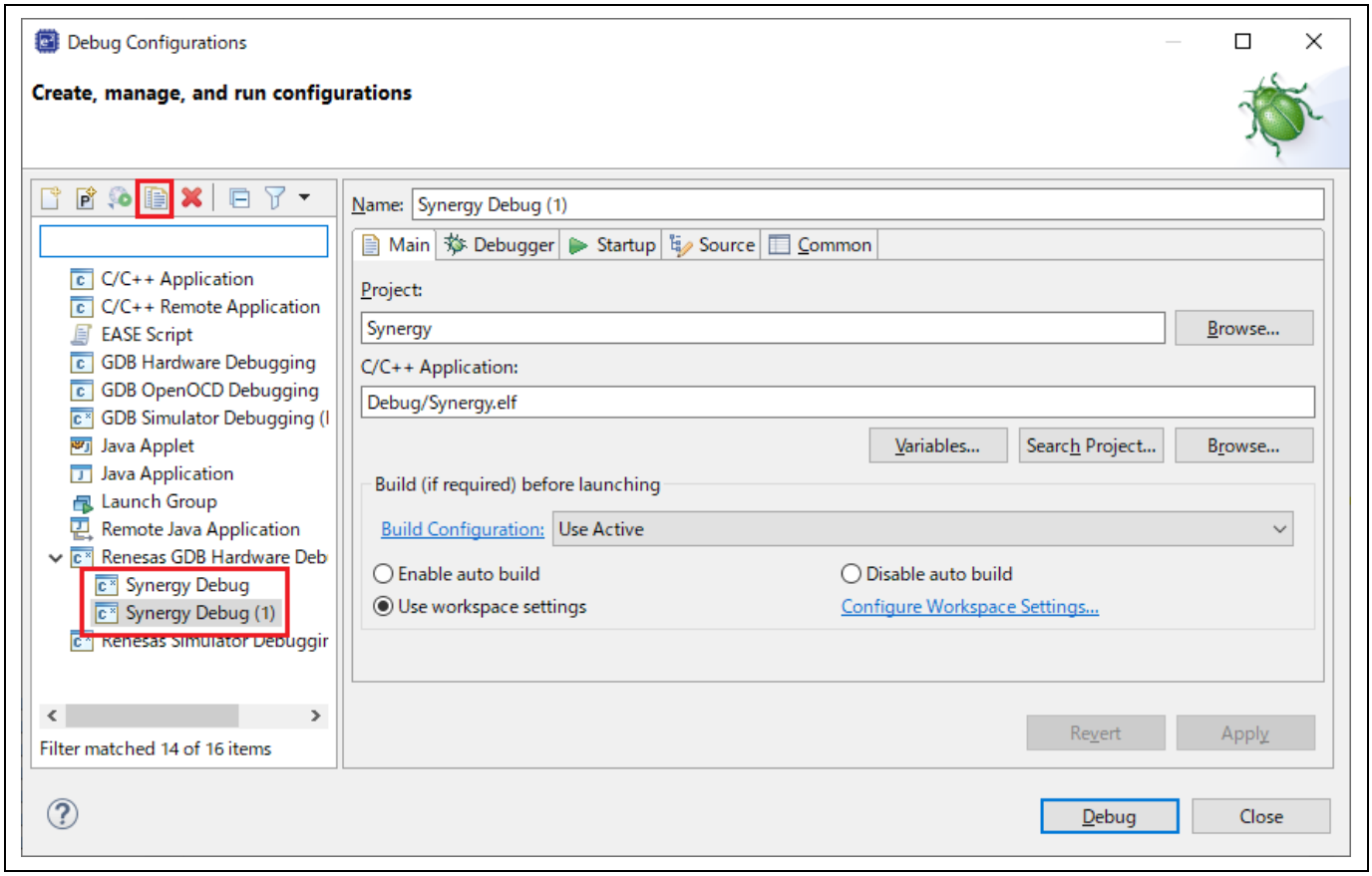


Figure 5-6. Debug – Duplicating a Selected Debug Launch Configuration

3. The new debug configuration can be configured as described in chapter 0.

5.3 Basic Debugging Features

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

- Standard GDB Debug features (supported by the Eclipse IDE framework): Breakpoints, Expressions, Registers, Memory, Disassembly and Variables (MMU view is not supported in Synergy).
- Renesas Extension to Standard GDB Debug: IO Registers, Eventpoints, Trace, and Fault Status.

To open the **Debug Toolbar**, click the pull-down menu button and click on **Show Debug Toolbar**. The following are some useful toolbars in the **Debug** view:

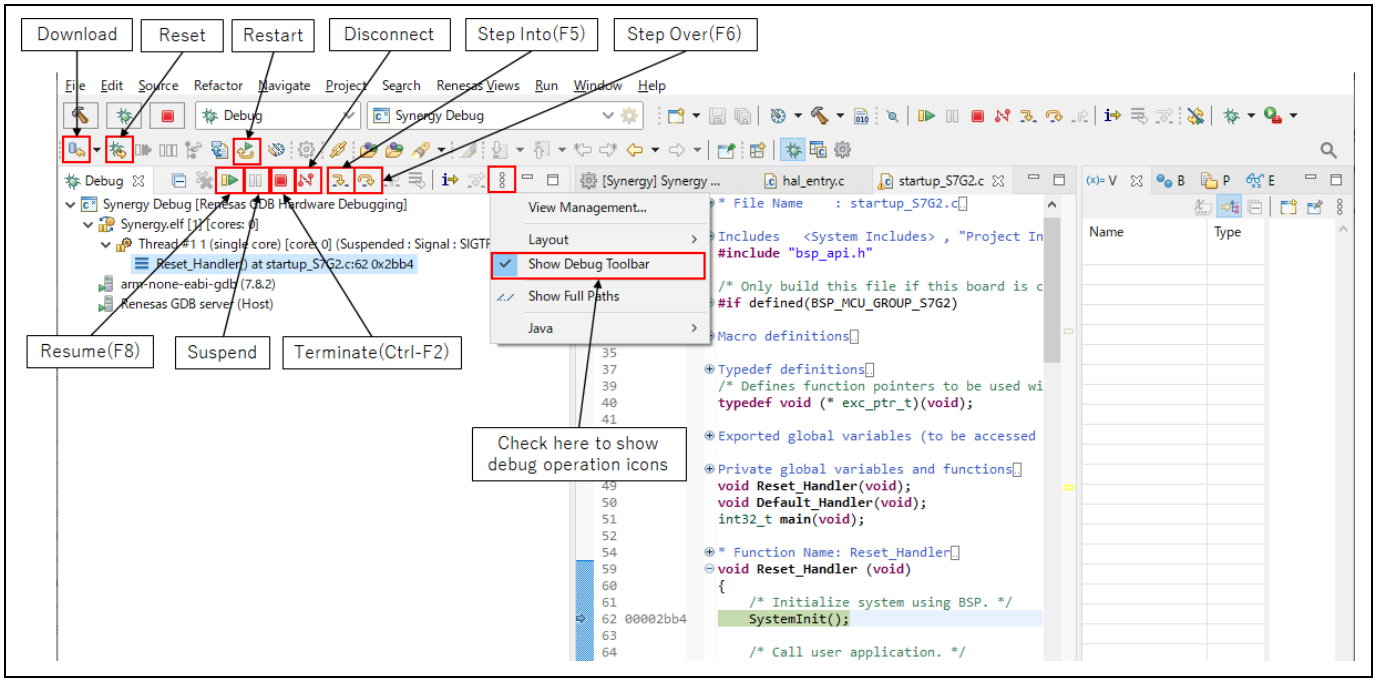


Figure 5-7. Debug – Useful Toolbars in Debug Views

The program is run by clicking the button or pressing F8.

Program execution can be suspended by breakpoint or by clicking the button. When program execution is suspended, the 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 program execution.

To stop the debugging process, click the button to end the selected debug session and/or process or click the button to disconnect the debugger from the selected process.

The other operations are as follows:

- The button can be clicked to reset and run the program. It may stop at `main()` if the breakpoint is configured in the **Debug** configuration.
- The button can be clicked to reset the program to its entry point at the PowerOn Reset.
- The button re-downloads the binary file to the target system.

5.3.1 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 hardware or software breakpoint:

1. 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**.

To set a breakpoint:

1. As an example, in "`startup_S7G2.c`" at line 62, 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 breakpoints set. 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.

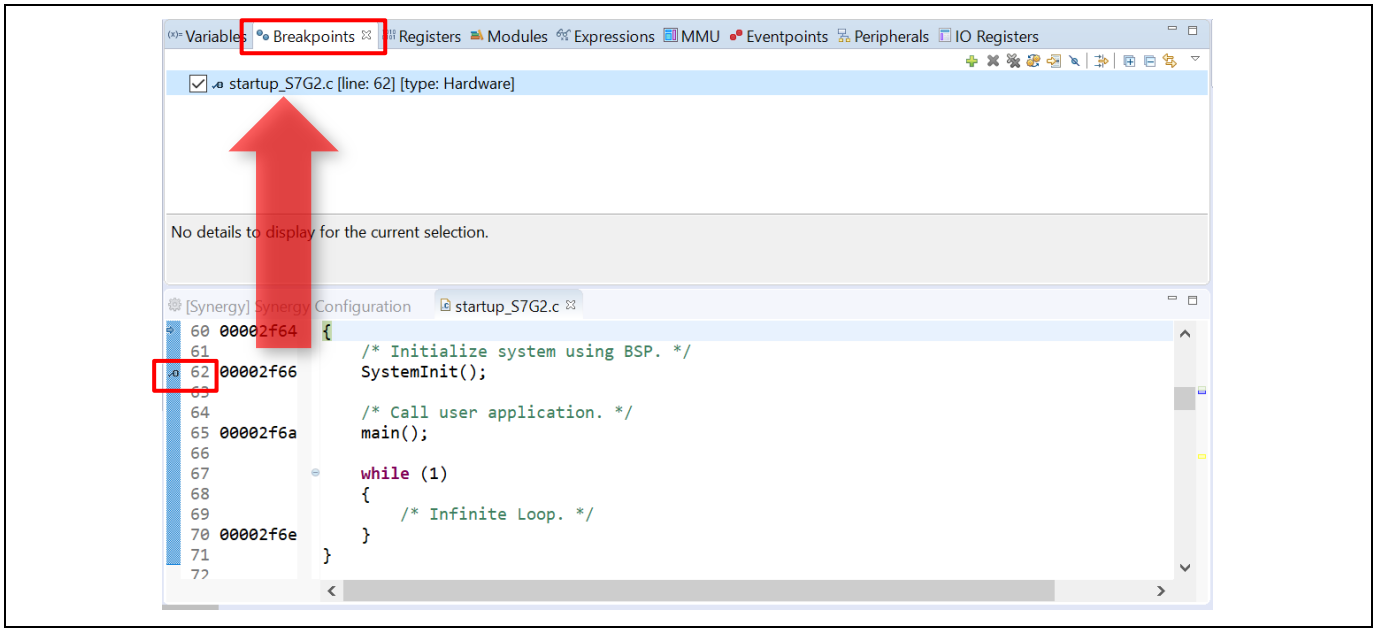


Figure. 5-8 Debug – Breakpoints view

5.3.2 Expressions View

The **Expressions** view monitors the value of global variables, static variables, or local variables during debugging.

Follow the steps below to watch a variable:

1. Click **Window** → **Show View** → **Expressions** or the icon to open the **Expressions** view.
2. Drag and drop a variable (for example, `g_cgc_version` in `r_cgc.c`) to the **Expressions** view.
   (Alternatively, right-click the variable to select the **Add Watch Expression…** menu item to add it to the **Expressions** view).

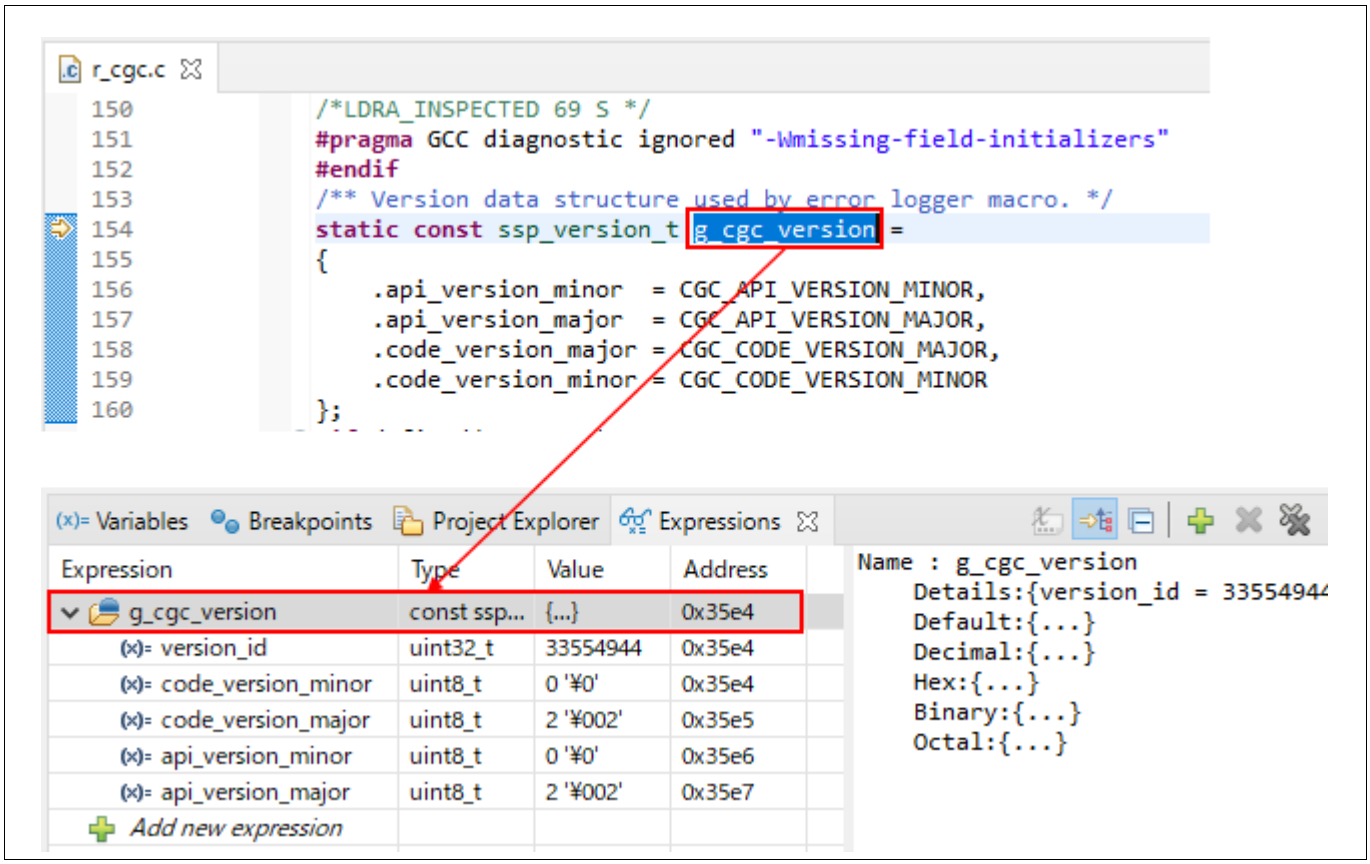


Figure 5-9. Debug - Expressions view

5.3.3 Registers View

The **Registers** view lists the information about the general registers in Synergy. Changed values are highlighted when the program stops.

1. Click **Window** → **Show View** → **Registers** or icon to open the Registers view.
2. Click a register to view the values in a different radix format.

Values that have been changed are highlighted (for example, in yellow) in the **Registers** view when the program stops.

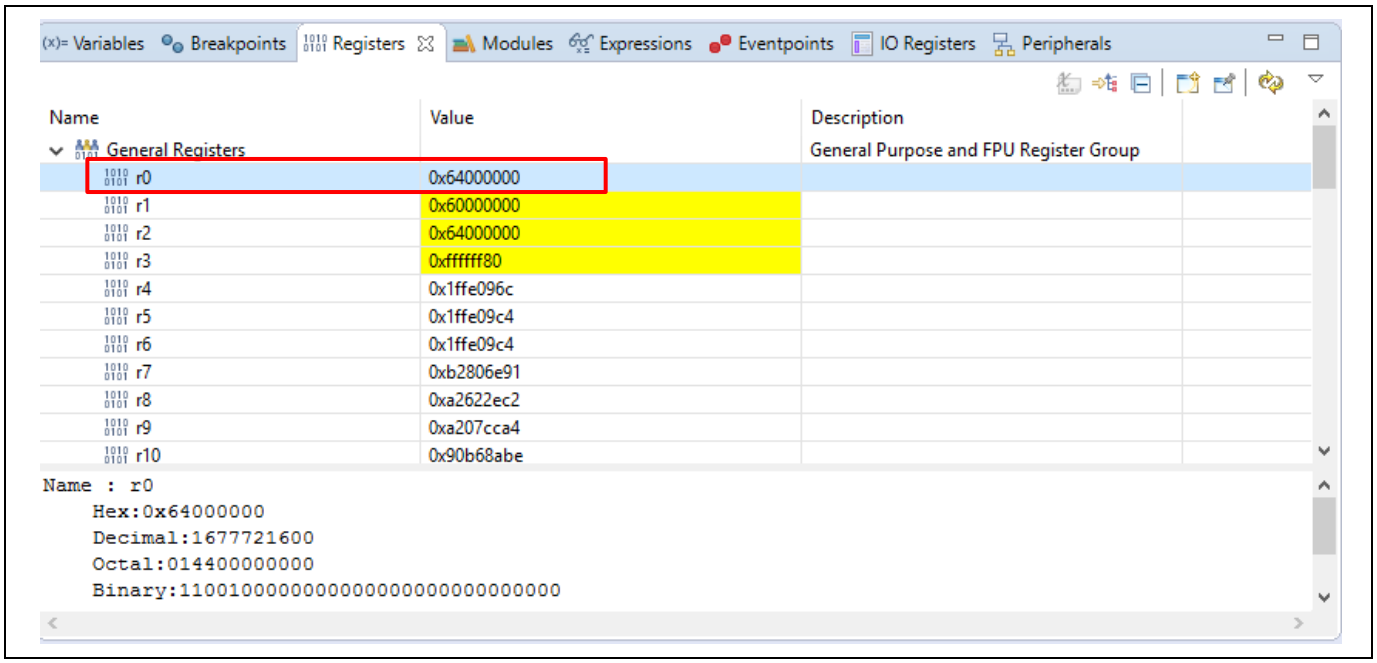


Figure 5-10. Debug – Registers view

5.3.4 Memory View

The **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 (for example, Hex integer, signed integer, unsigned integer, ASCII, and image).

To view the memory of a variable (for example, `g_ssp_version_build_string`):

1. Click **Window** → **Show View** → **Memory** or icon to open the **Memory** view.
2. Click the icon to open the **Monitor Memory** dialog box. Enter the address of the variable `&g_ssp_version_build_string`.

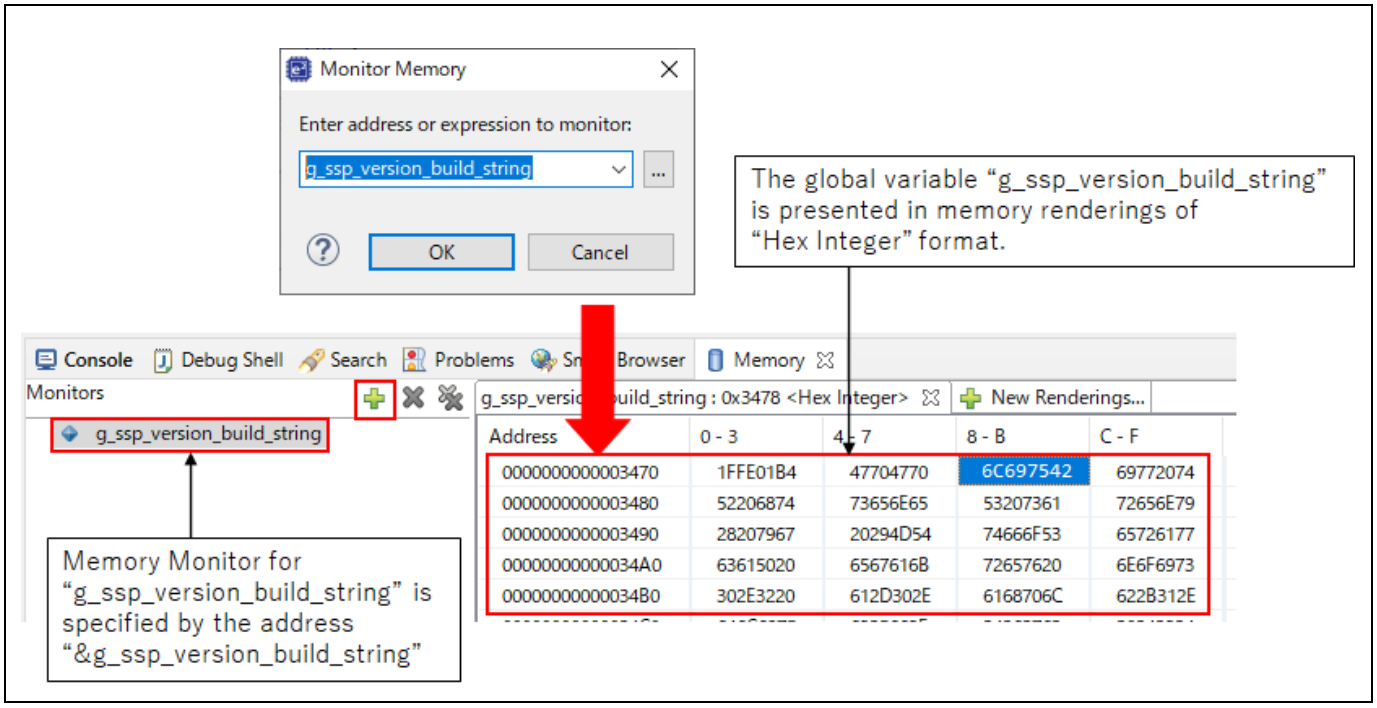


Figure 5-11. Debug – Memory view

To add a new rendering format (for example, ASCII) for the variable `g_ssp_version_build_string`:

1. Click the tab New Renderings... to select **ASCII** to add the rendering. This creates a new tab named **&g_ssp_version_build_string <ASCII>** next to the tab **&g_ssp_version_build_string <Hex Integer>**.

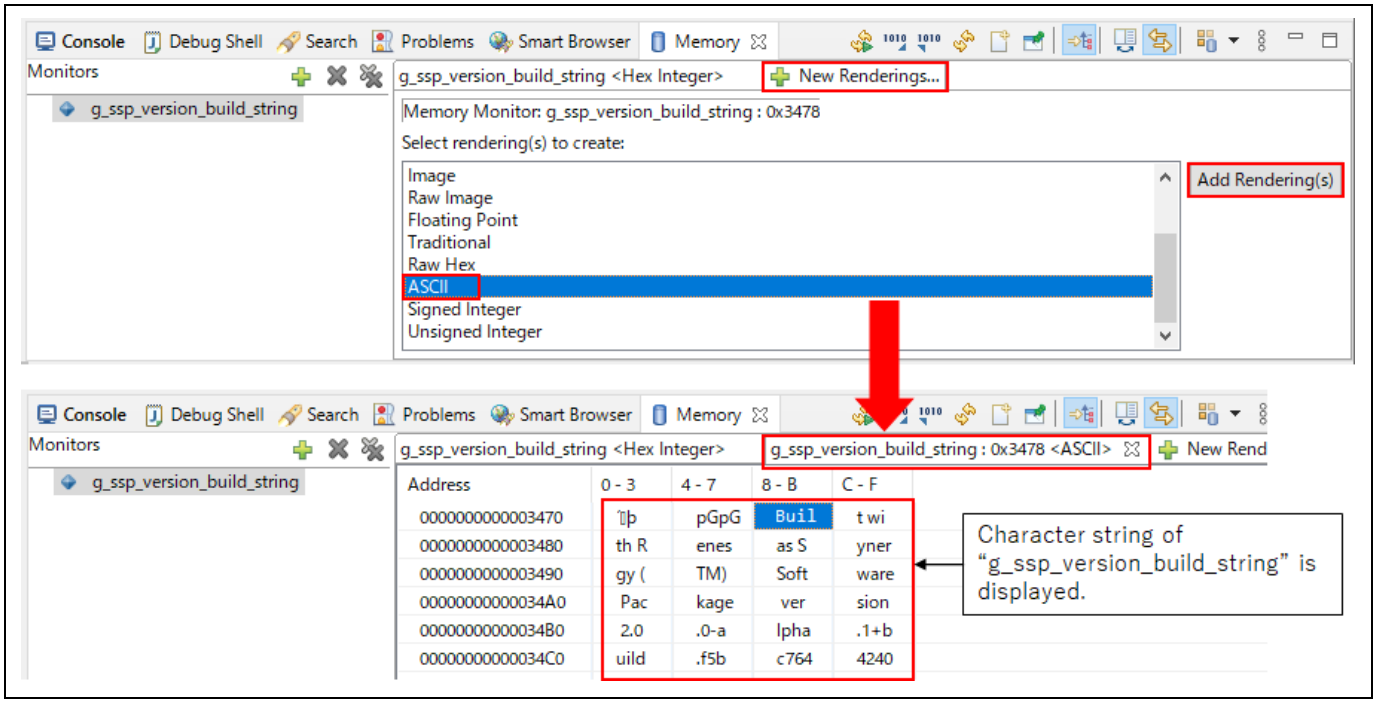


Figure 5-12. Debug – New Rendering In Memory view

5.3.5 Memory Usage view

Memory Usage will be used to get the information of `*.map` files or library list files (`*.lbp`) from a project. It will list out the total memory size, usage of ROM and RAM ratio, and detailed information of sections, objects, symbols, module, vector, and cross references used in project.

From version 7.3, e² studio supports the graphical view to show usage in the ROM and RAM memory areas.

To show the **Memory Usage** view, click on the menu **Window → Show View → Others…** to open the **Show View** dialog and click **C/C++**, select "**Memory Usage**" and click [**Open**.

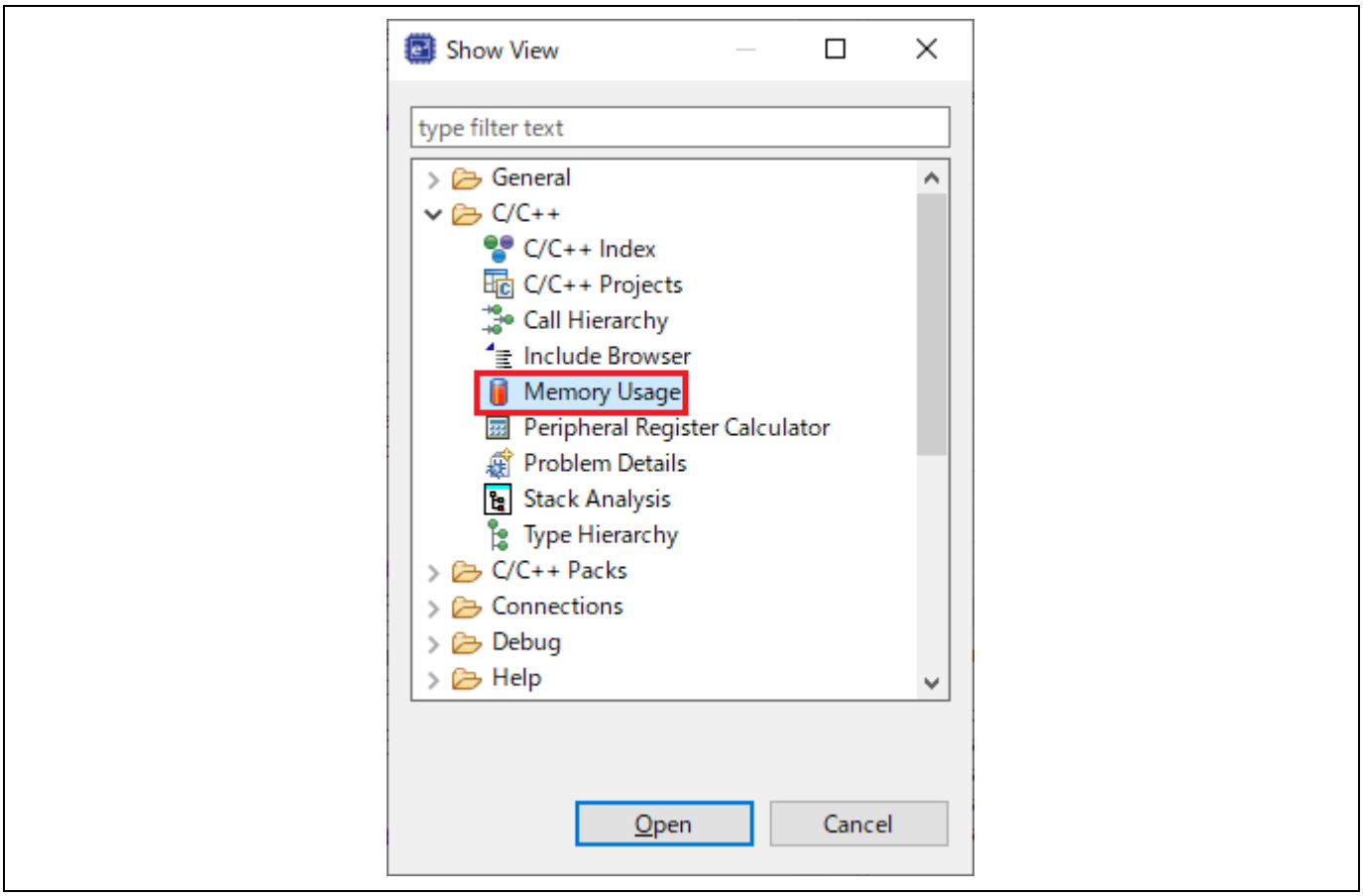


Figure 5-13. Show Memory Usage view

The **Memory Usage** view has 3 regions: (1) Group size region, (2) Memory Region Usage region (Device Memory Usage region is not supported yet), (3) Detail table region.

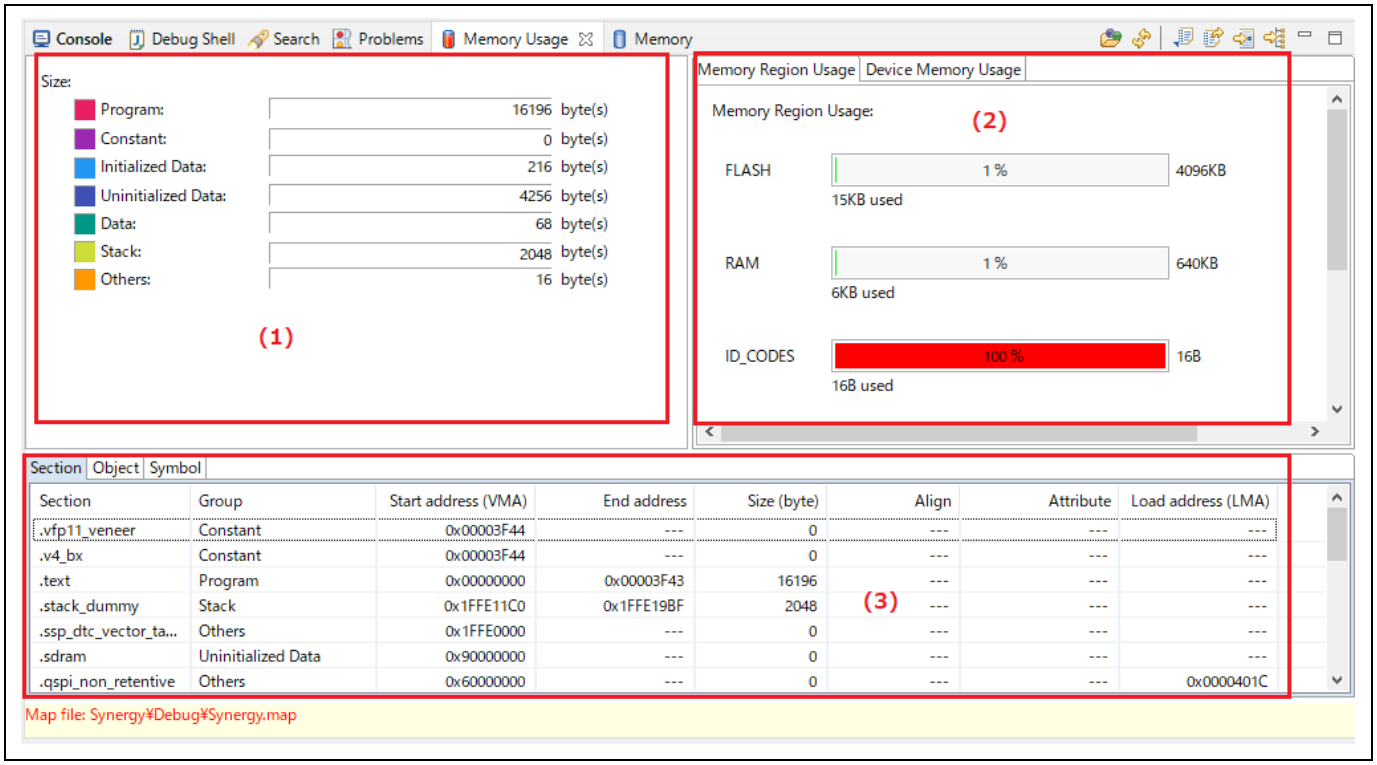


Figure 5-14. Regions of Memory Usage views

Following operations are supported in **Memory Usage** view:

: Choose a map or library list file for **Memory Usage** display.

: Refresh all information of **Memory Usage** view.

: Open a `*.map` or `*.lbp` file in the Editor (there is no library list file in the Synergy library project).

: Open Map file output page of selected project.

: Open Section page of selected project.

5.3.6 Disassembly View

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

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

1. Click **Window** → **Show View** → **Disassembly** or the icon to open the **Disassembly** view.
2. Click the 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 Sh**ow Function Offsets**.

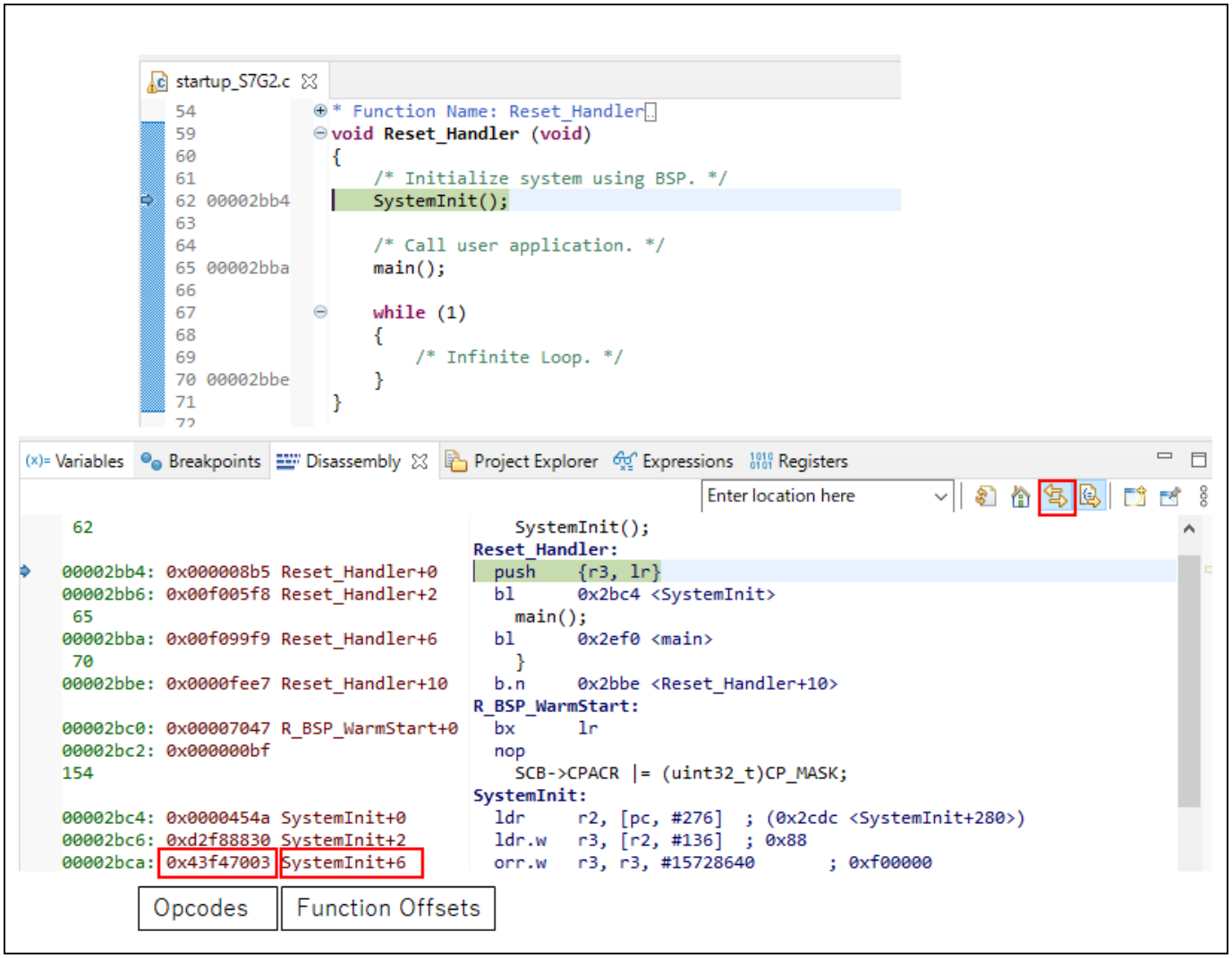


Figure 5-15. Debug – Disassembly view

5.3.7 Variables View

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

To observe a local variable (for example, `timeout` for function `R_CGC_Init()`):

1. Click **Window** → **Show View** → **Variables** or the icon to open the **Variables** view.
2. Step into the function `R_CGC_Init ()` to view the local variable `timeout` value.

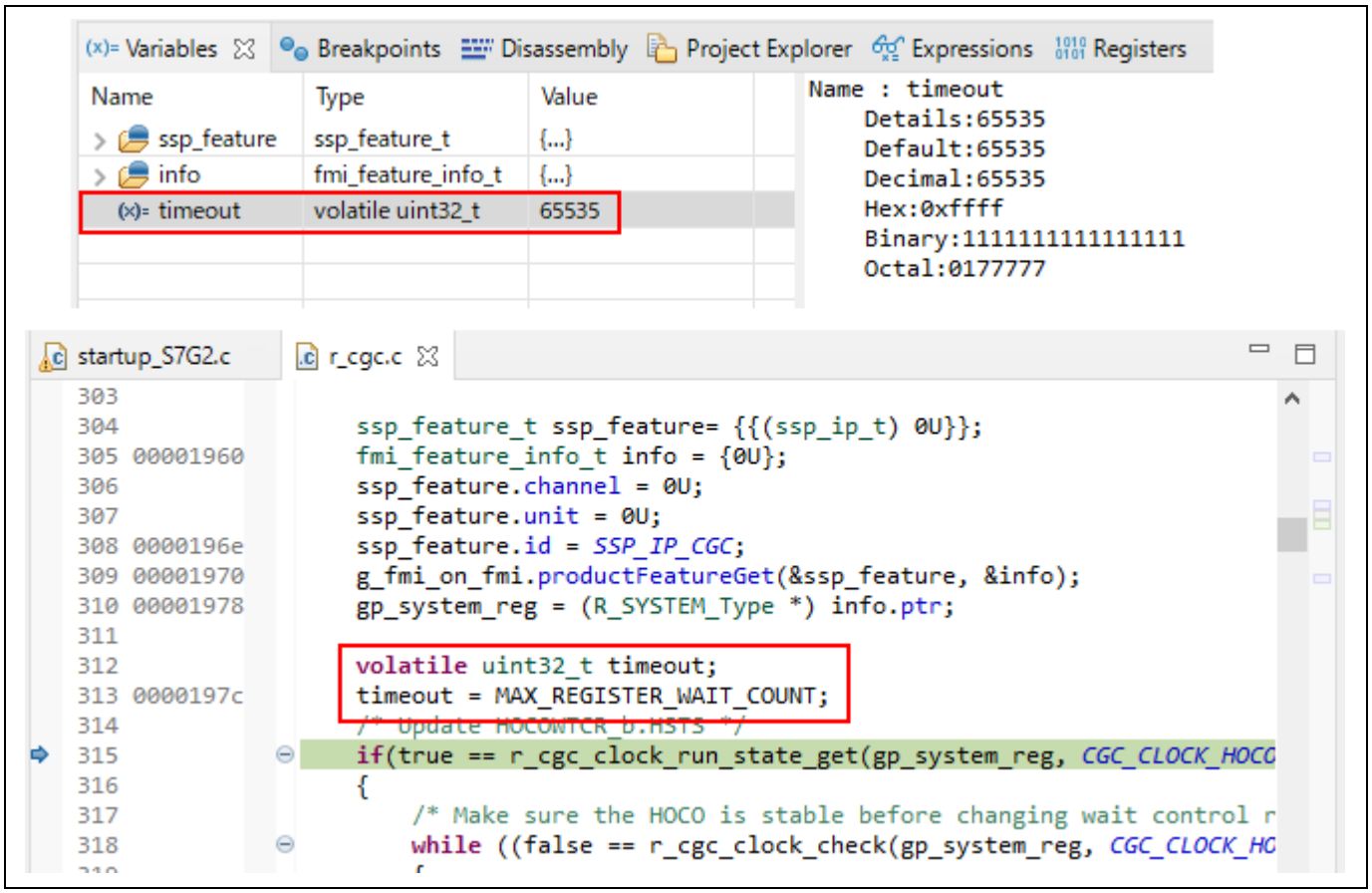


Figure 5-16. Debug – Variables view

5.3.8 IO Registers View

The IO registers are also known as the Special Function Registers (SFRs). The **IO Registers** view displays all the registers defined in a target-specific IO file. Users can further customize the IO Registers view by adding specific IO registers to the Selected Registers pane.

To view selected IO registers:

1. Click **Renesas Views** → **Debug** → **IO Registers** or icon to open the **IO Registers** view.
2. Under the **All Registers** tab, locate a module (for example, CAC) in the **IO Registers** view. Expand its IO register list.
3. Drag and drop its registers (for example, **CAICR** and **CASTR**) to the **Selected Registers** pane. A green dot next to the IO register indicates the status of being a selected register.
4. Switch to the **Selected Registers** tab to view the selected IO Registers.

The expanded IO register list may take more 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.

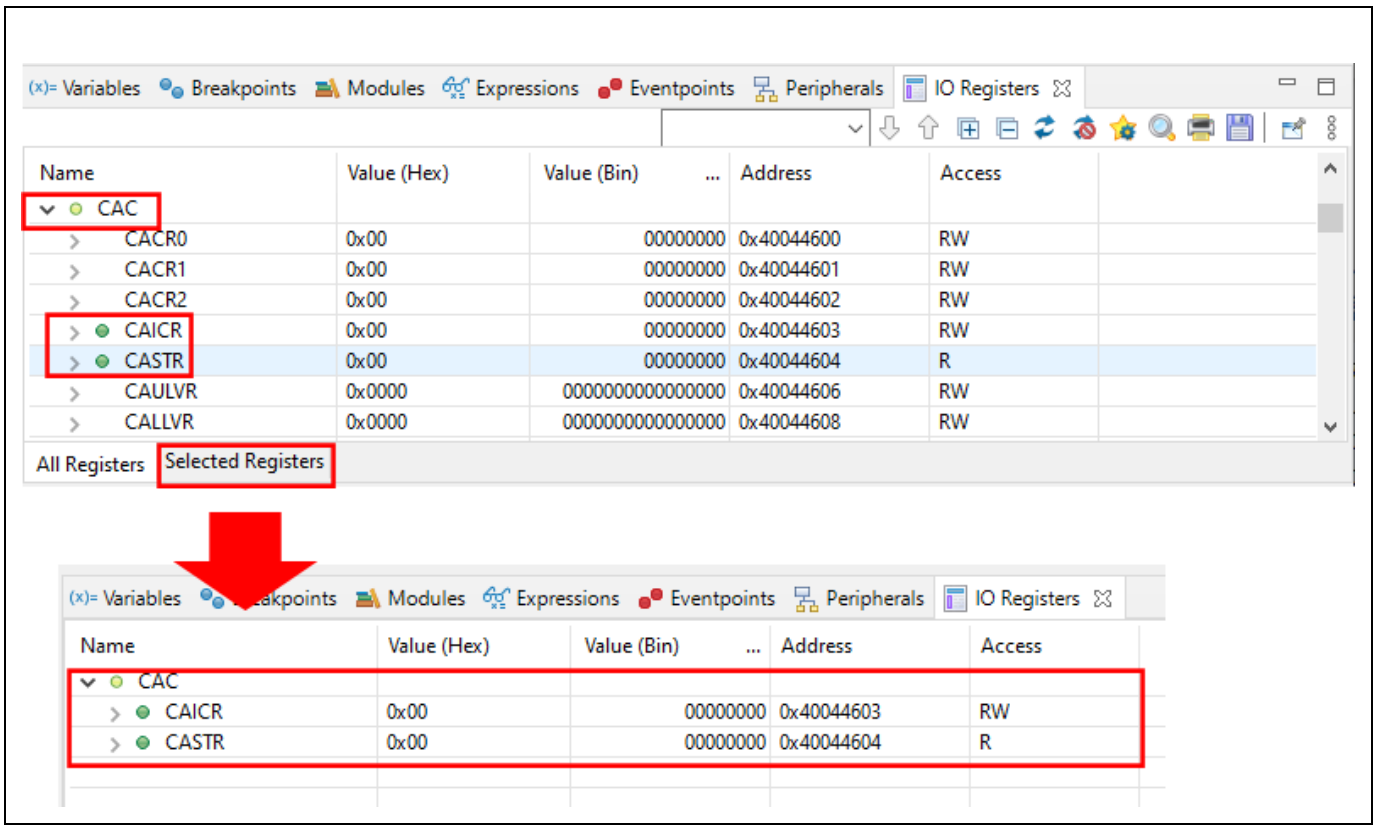


Figure 5-17. Debug – IO Registers view

To find IO registers:

1. Enter the name of the IO register to search in the search box and click the down arrow button .
2. The searched register is displayed in the **IO Register** view.

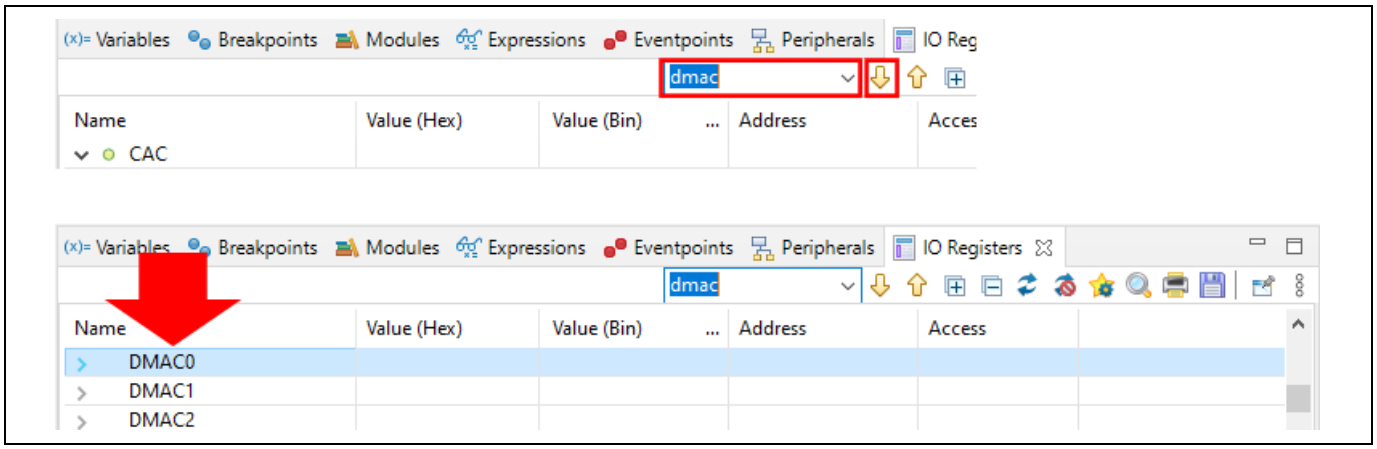


Figure 5-18. Debug – IO Registers view(Find IO registers)

5.3.9 Eventpoints View

An 'event' refers to a combination of conditions set for executing break or trace features during program execution. The **Eventpoints** view enables users to set up or view defined events of different categories, such as trace start, trace stop, or event break.

Data access event break is supported for Synergy projects. The emulator detects access under a specified condition to a specified address or a specified address range. This allows complex address and data matching criteria to be set up.

Event combinations (for example, OR, AND (cumulative) and Sequential) can be applied to two or more events.

Table 5-1. Event combinations

Event combination	Explanation
OR	The condition is met when any one of the specified events occurs.
AND (cumulative)	The condition is met when all of the specified events occur regardless of the timing.
Sequential	The condition is met when the specified events occur in a specified order.

To set an event break for a global variable when address/data is matched (for example, when `g_bsp_leds` is accessed):

1. Click **Renesas Views** → **Debug** → **Eventpoints** or the icon to open the **Eventpoints** view.
2. Double-click the **Event Break** option to open the **Edit Event Break** dialog box.
3. Click the **Add…** button to continue.

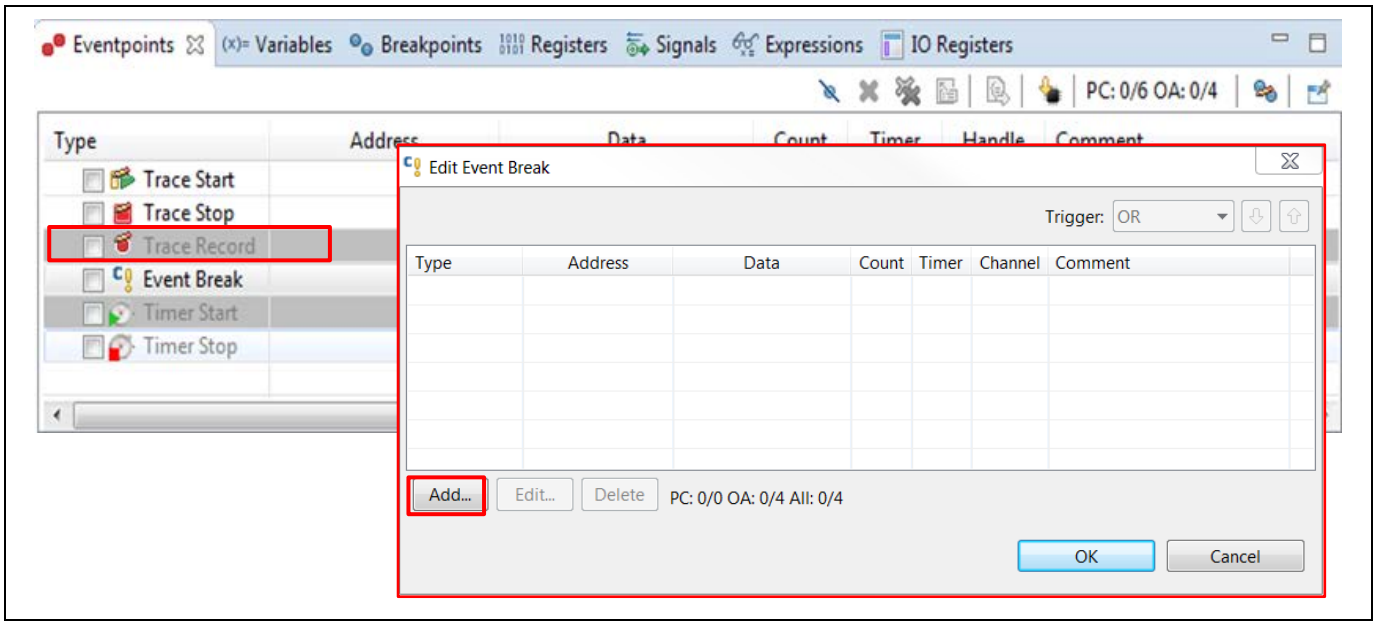


Figure 5-19. Debug – Eventpoints view (1/2)

1. The **Data Access** Eventpoint Type is selected by default.
2. Go to the **Address Settings** tab and click the **…** icon to browse for the symbol `g_bsp_leds`. (The address of this global variable is `&g_bsp_leds`.)
3. Next, switch to the **Data Access Settings** tab and set the **Read/Write** selection to **Read**.
4. Click **OK** to proceed.

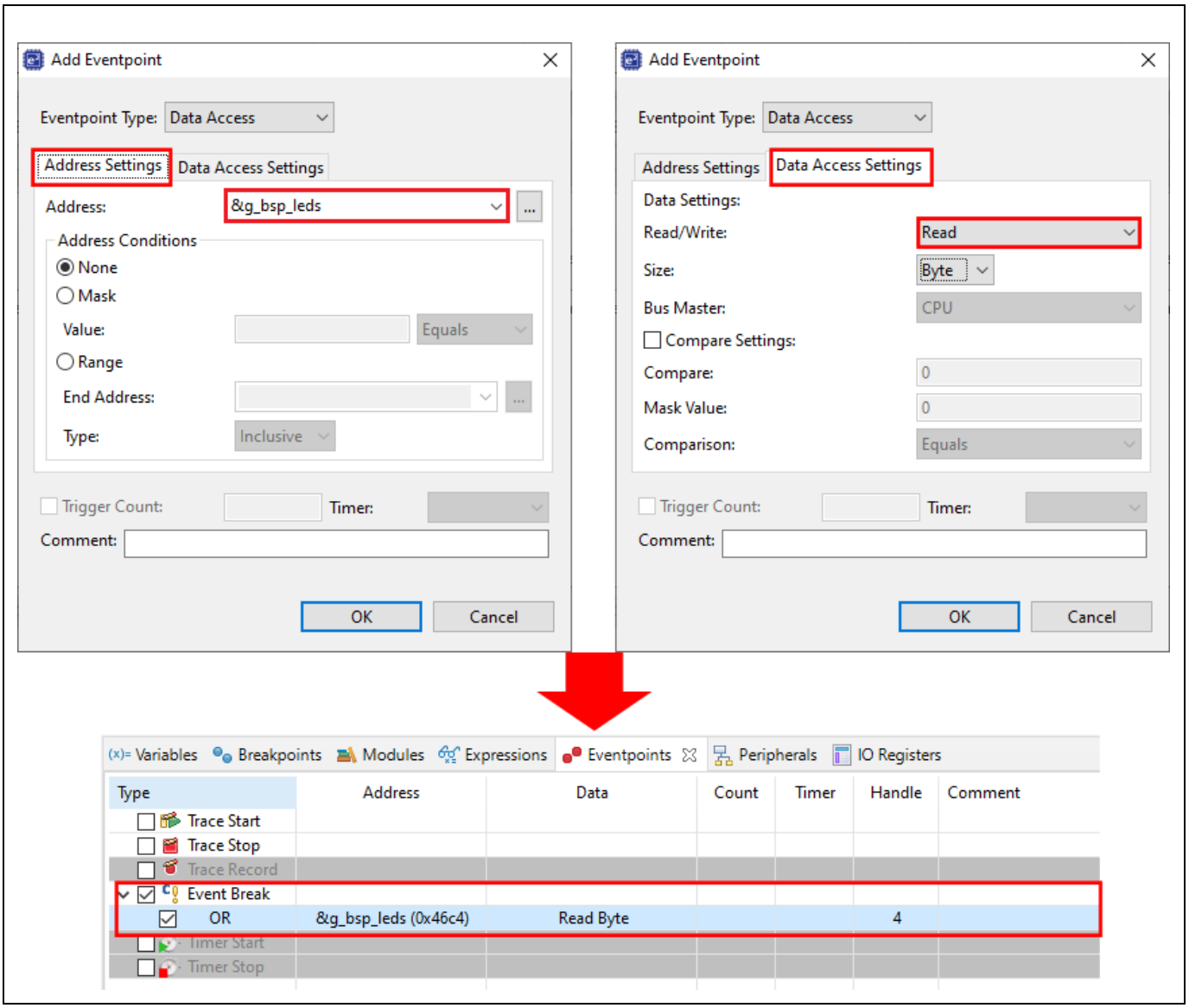


Figure 5-20. Debug – Eventpoints view (2/2)

5. Perform a reset to execute the program from the start.
6. Figure 5-21 shows that when the variable `g_bsp_leds` is accessed (read), the program stops.

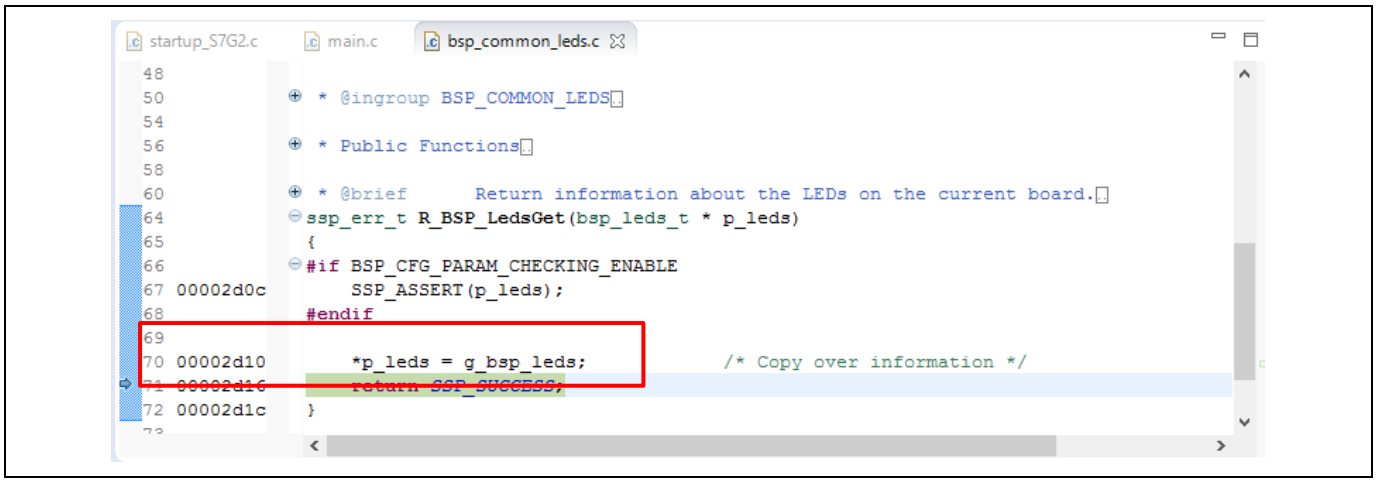


Figure 5-21. Debug – Execution of Event Break

5.3.10 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. It helps users to track the program execution flow to search for and examine the points where problems arise.

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

To set a trace until the program is suspended:

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

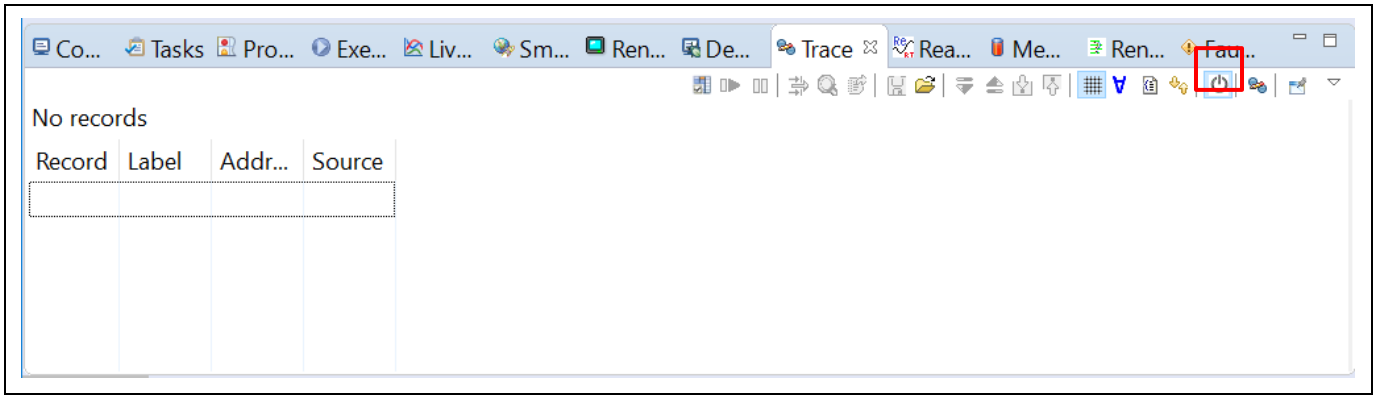


Figure 5-22. Debug – Turn on Trace view

1. 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.
2. Select the display mode by clicking on the corresponding button.

Figure 5-23 shows the trace result before the `main()` function is executed.

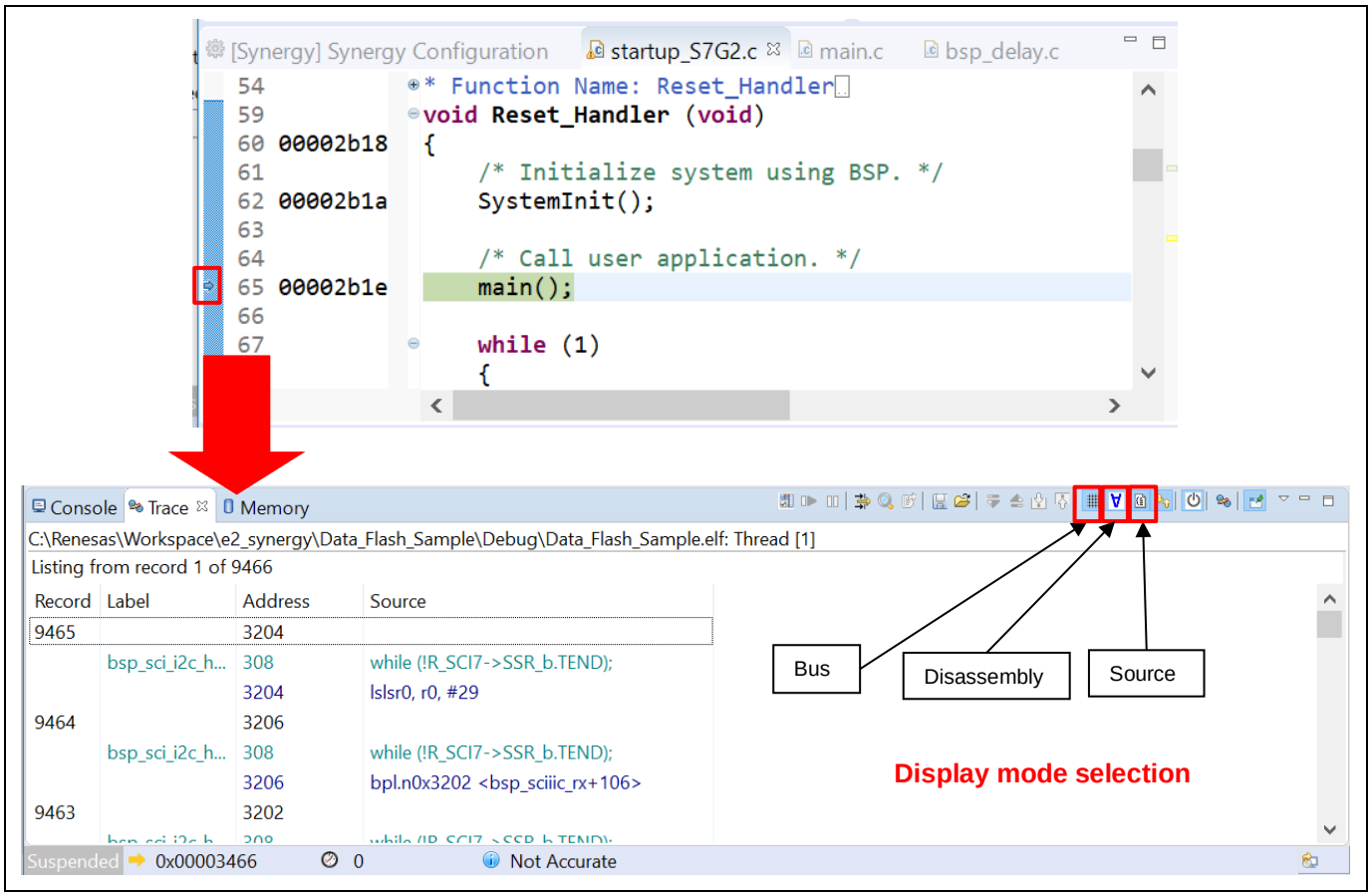


Figure 5-23. Debug – Select display mode in Trace view

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

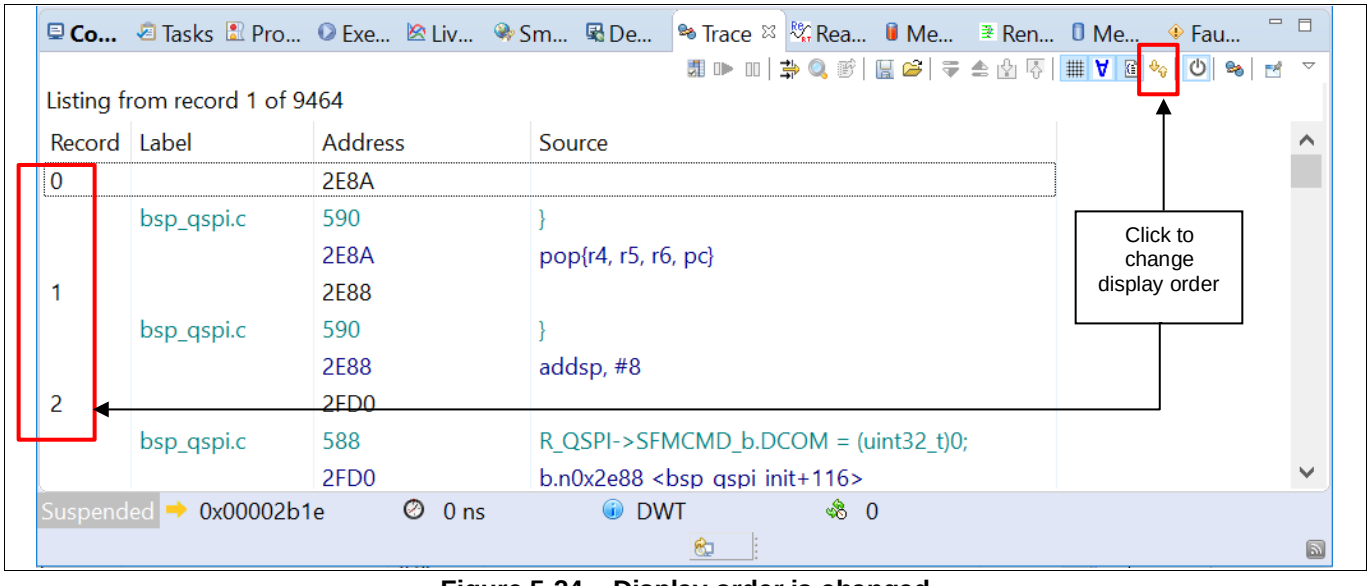


Figure 5-24. Display order is changed

2. The trace result can be filtered by clicking on the button. User can select to filter by **Record** and/or **Address**.

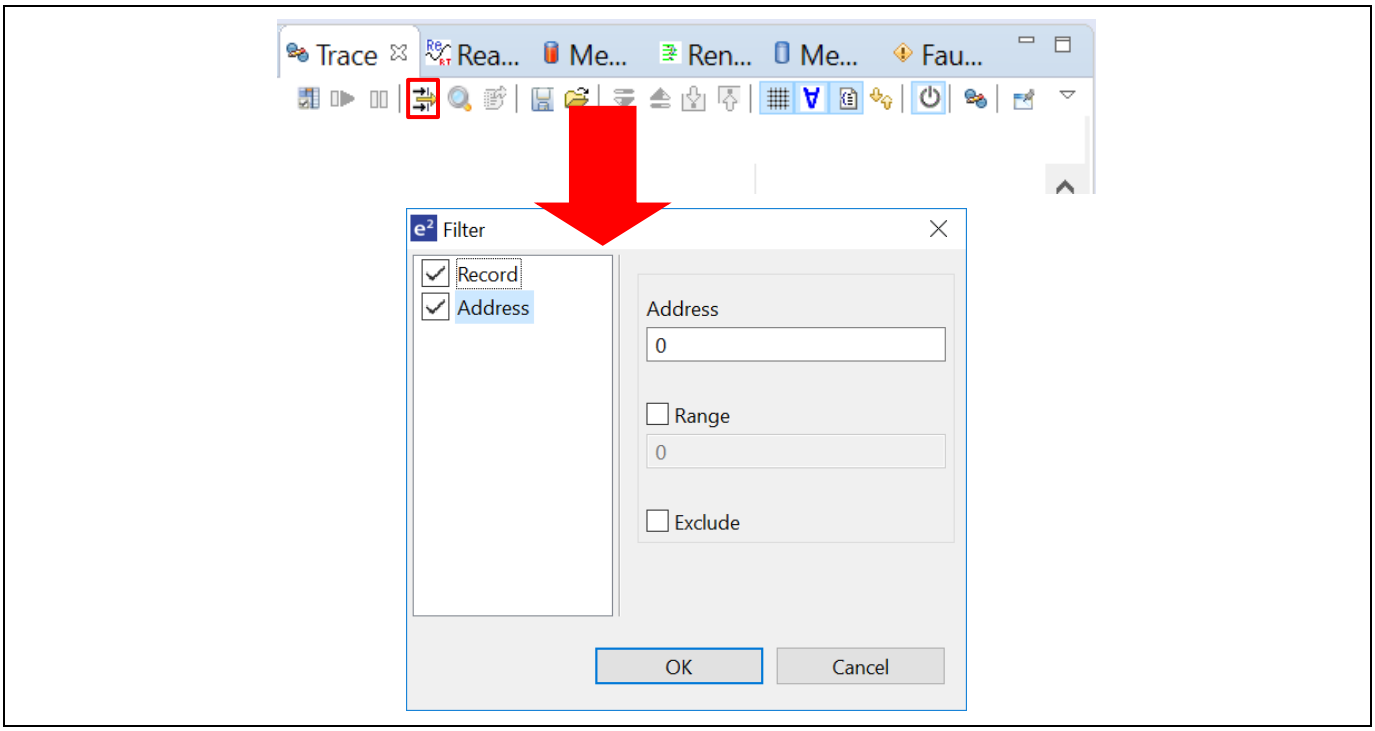


Figure 5-25. Debug – Filter trace result

3. 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 result from a `.csv` file.

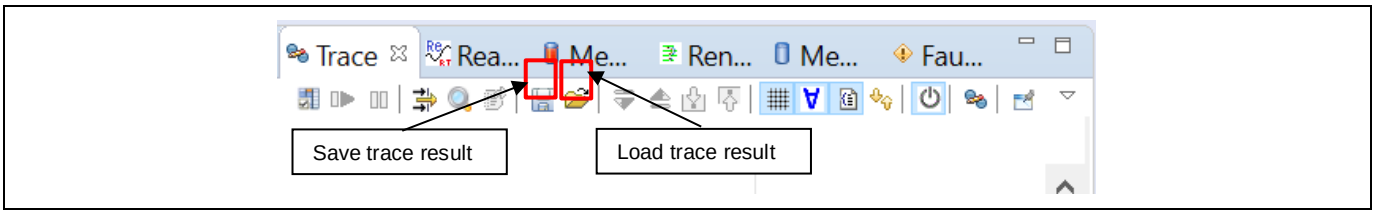


Figure 5-26. Debug – Save and load trace result

5.3.11 Fault Status View

The **Fault Status** view shows the bit status of several fault status registers and the value of the key register to the user when a hardware fault crash occurs. When a hardware fault occurs, the bits of the register related to the cause of the fault are checked and the r0, r1, r2, r3, r12, lr, pc, and psr register values are displayed. This is shown in Figure 5-27 below. This function is available in e² studio v5.2 and above.

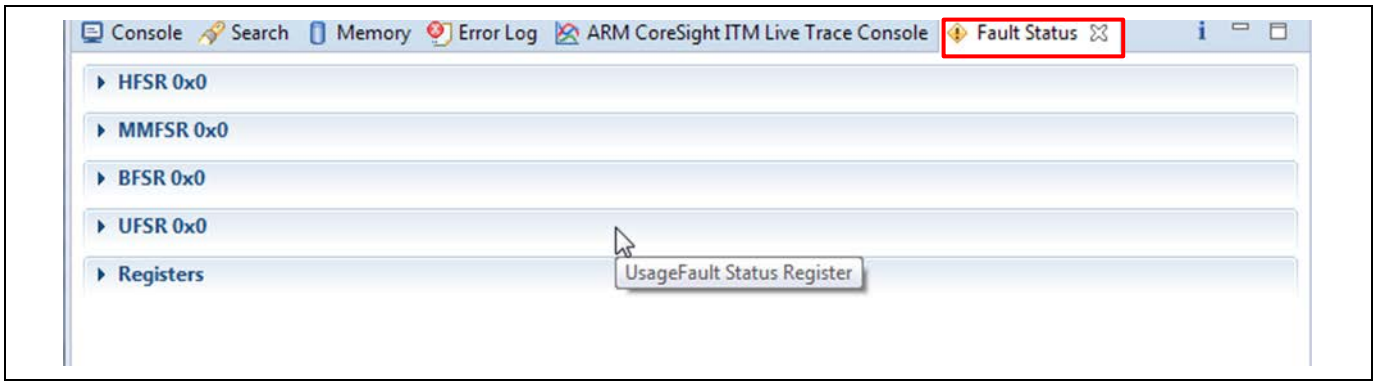


Figure 5-27. Fault Status No hardware fault

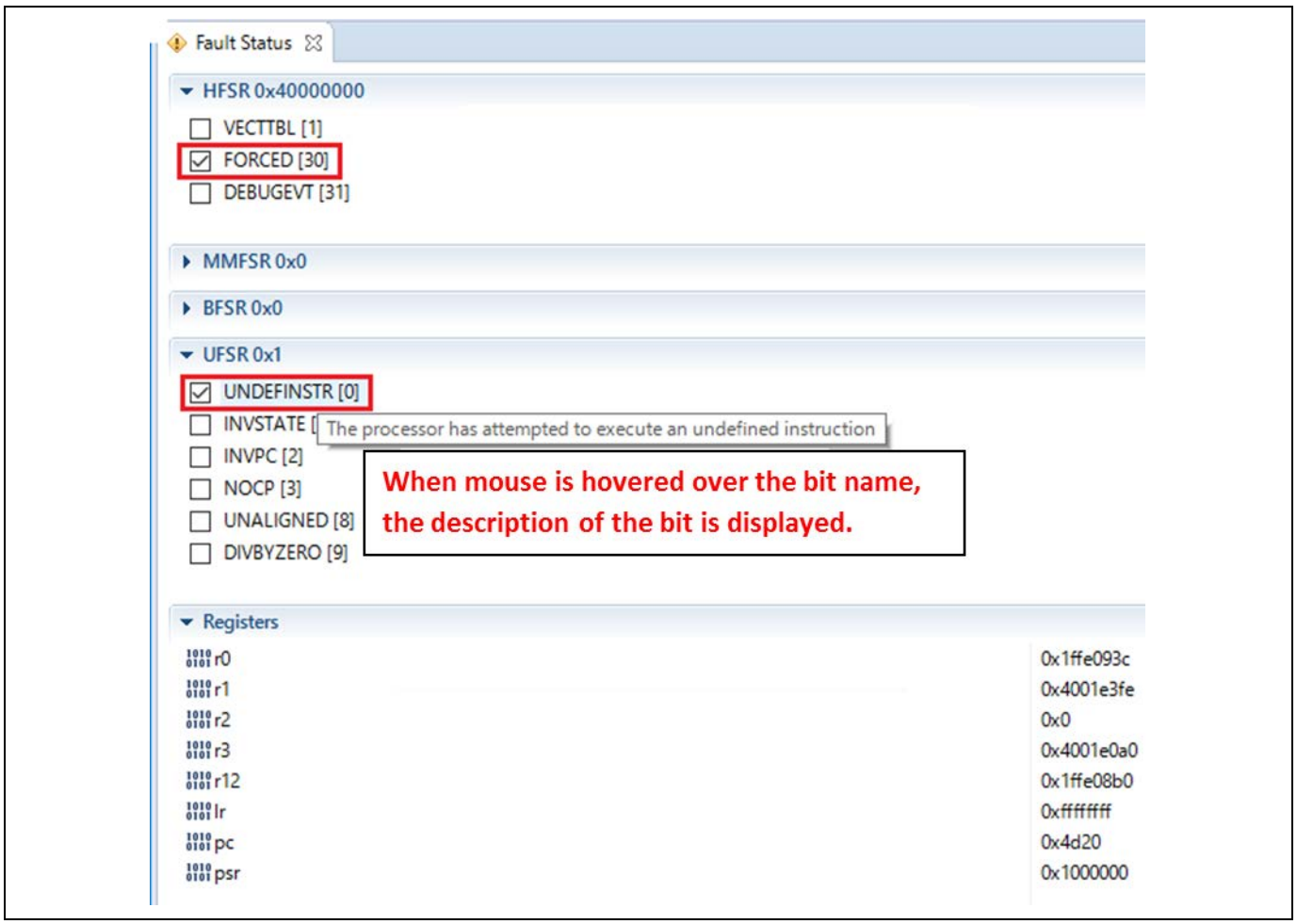


Figure 5-28. Fault Status Hardware fault occurred

5.3.12 Run Break Timer

The Run Break Timer feature allows the user to see the last execution performance on the status bar. When the program is suspended, user can check the current program counter (PC), the last execution timing either in time or CPU cycles and the accuracy or measurement method used.

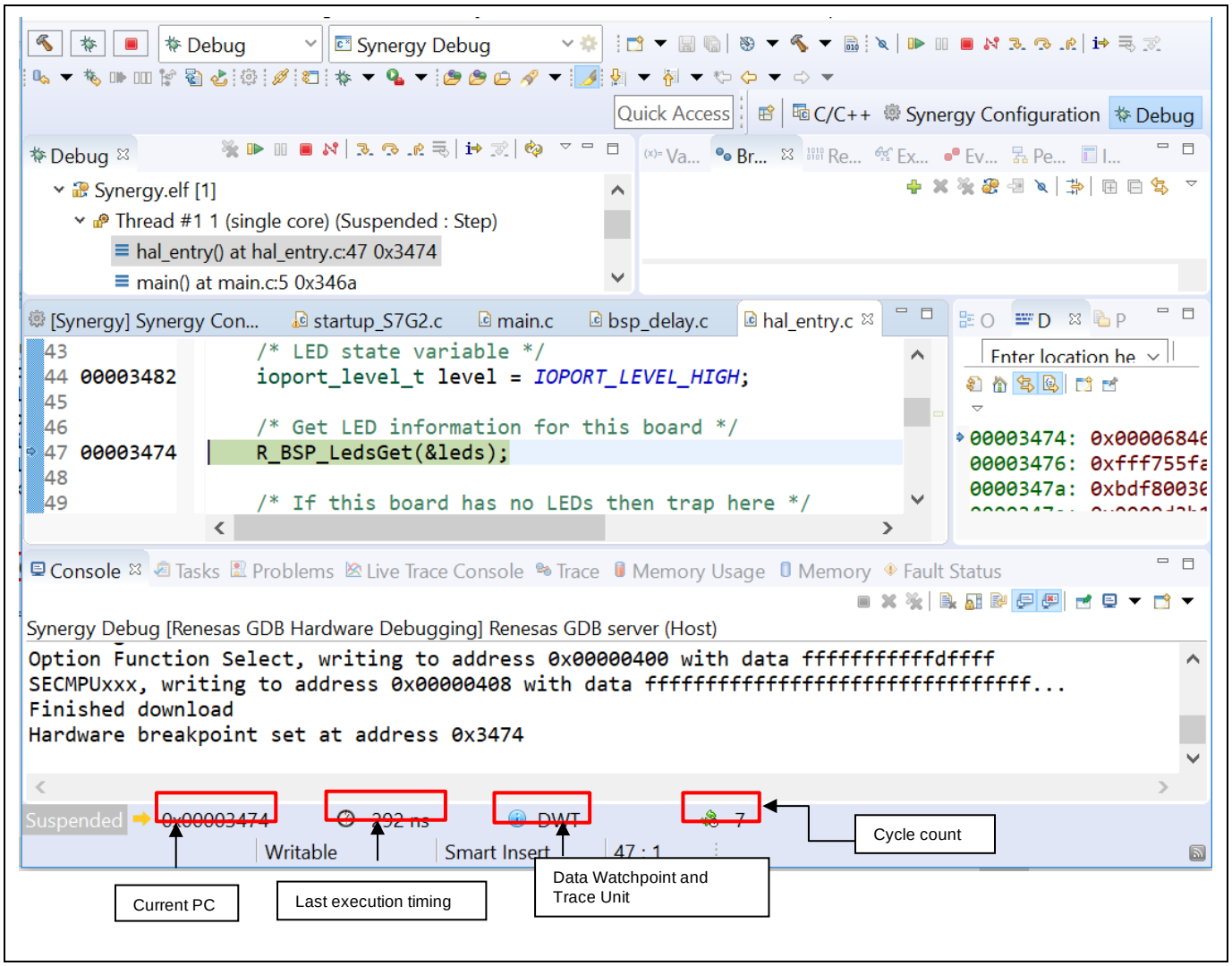


Figure 5-29. Run Break Timer shows the last execution performance

Table 5-2 shows the support of Run Break Timer feature available for various Synergy devices.

Table 5-2 Support for Run Break Timer

Device	Debugger	Support
Synergy S1 Series (Cortex M0+/M23)	J-Link	System Time
Synergy S3, S5, S7, Series	J-Link	Data Watchpoint and Trace Unit (DWT) – Cycle Count and number of overflows calculated using the System Time

The Run Break Timer feature is supported in e² studio v7.3.0 and higher version. For updates in the specification, refer to e² studio release note in this link. https://www.renesas.com/us/en/software-tool/e-studio

6. Setting Up a ThreadX Application

This example demonstrates how to generate and build a Synergy project to include ThreadX objects and General-Purpose Timer (GPT) module using the project template “Blinky with ThreadX”.

6.1 General Purpose Timer Example in ThreadX

In the “Blinky with ThreadX” Synergy project from Project Template Selection, LEDs are caused to blink by putting the task to sleep for a while before toggling the LEDs state.

In this example, instead of a sleep delay, the Blinky Thread waits for a semaphore and a timer interrupt (generated by GPT) which puts out this semaphore every 1 second so that thread can resume.

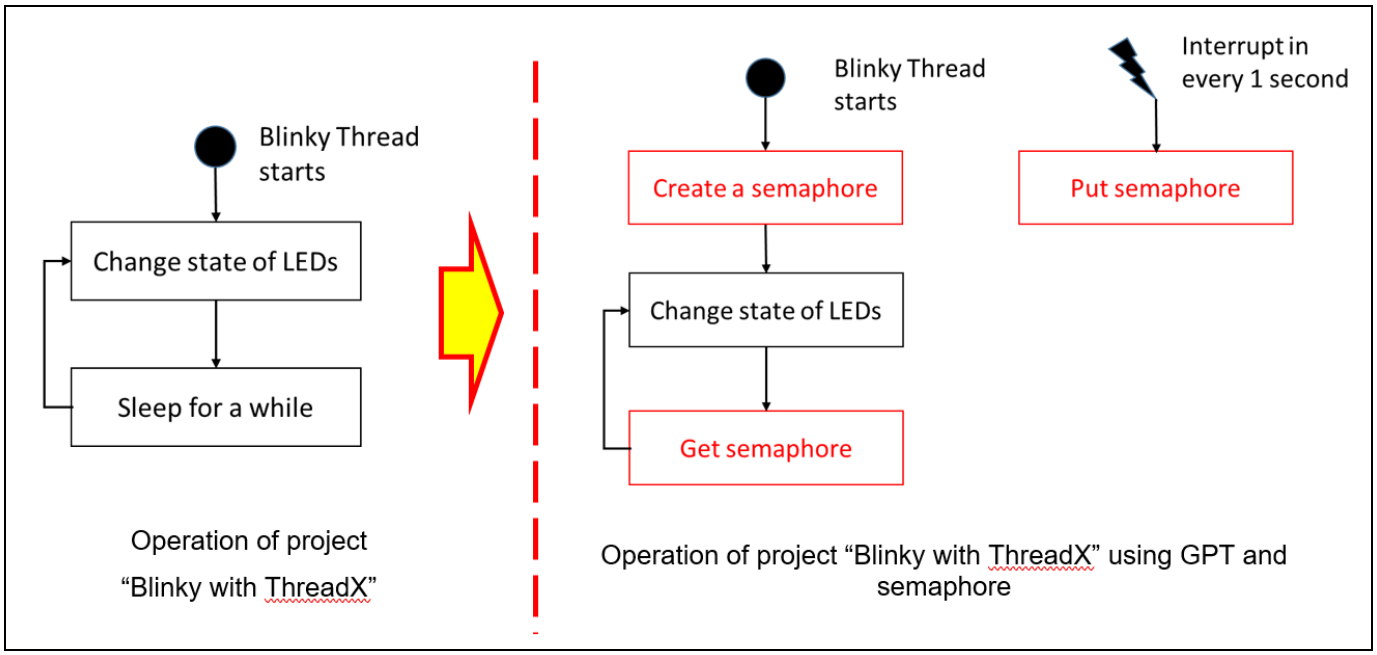


Figure 6-1. Setting up a ThreadX Application – Introduction

6.2 Creating the Sample Project

To create a sample ThreadX project with GPT and semaphore, configure the Synergy project as follows:

1. Invoke the **New Project** editor and follow the steps in Chapter 3.1 (Generating a New Synergy Project) to generate a new project. However, in the last dialog (**Project Template** dialog), select **Blinky with ThreadX**.

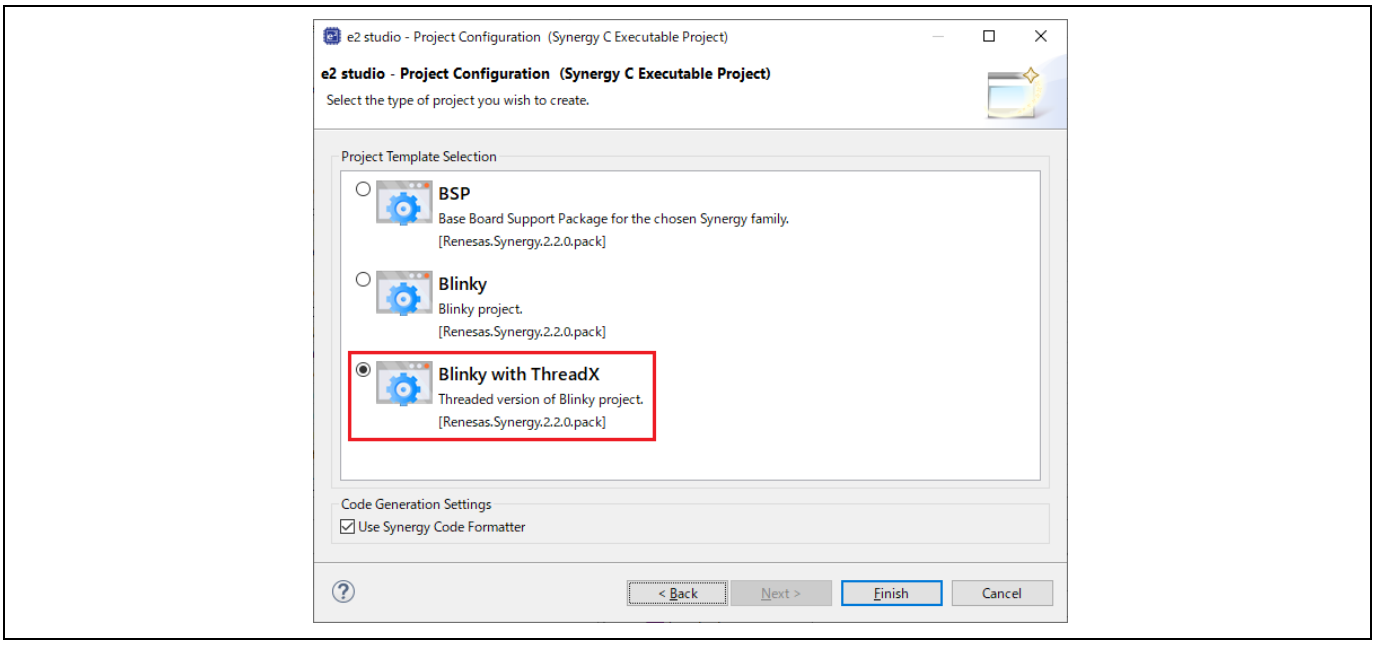


Figure 6-2. Setting up a ThreadX Application – Blinky with ThreadX template selection

2. Open the **Threads Configuration** page in the **Synergy Project Configuration**. Please refer to Chapter 3.4.5.
3. Add the GPT module to the Blinky Thread by selecting **Blinky Thread** in the **Threads** panel and selecting **New Stack → Driver → Timers → Timer Driver on r_gpt** in the **Stacks** panel.

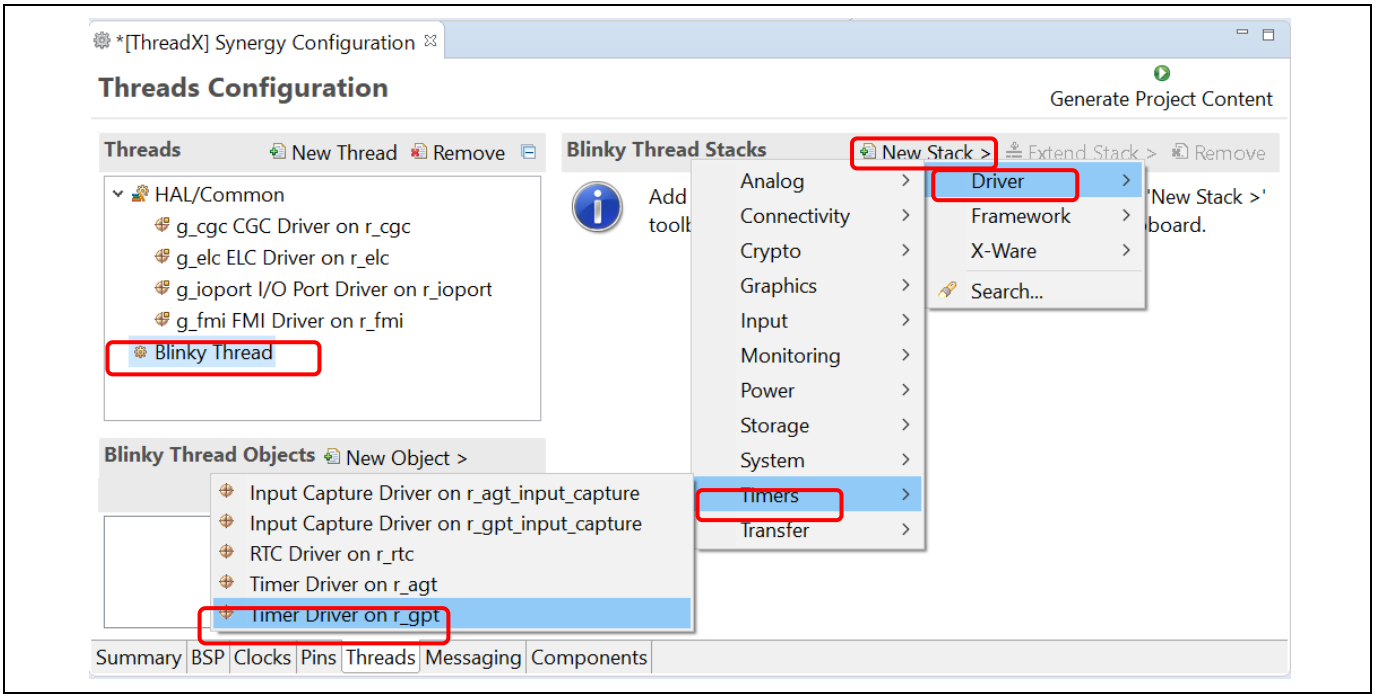


Figure 6-3. Setting up a ThreadX Application – Adding the GPT module

1. Configure the GPT module as follows.

- Name: g_timer
- Mode: Periodic
- Period Value: 1
- Period Unit: Seconds
- Callback: gpt_callback
- Overflow Interrupt priority: 2

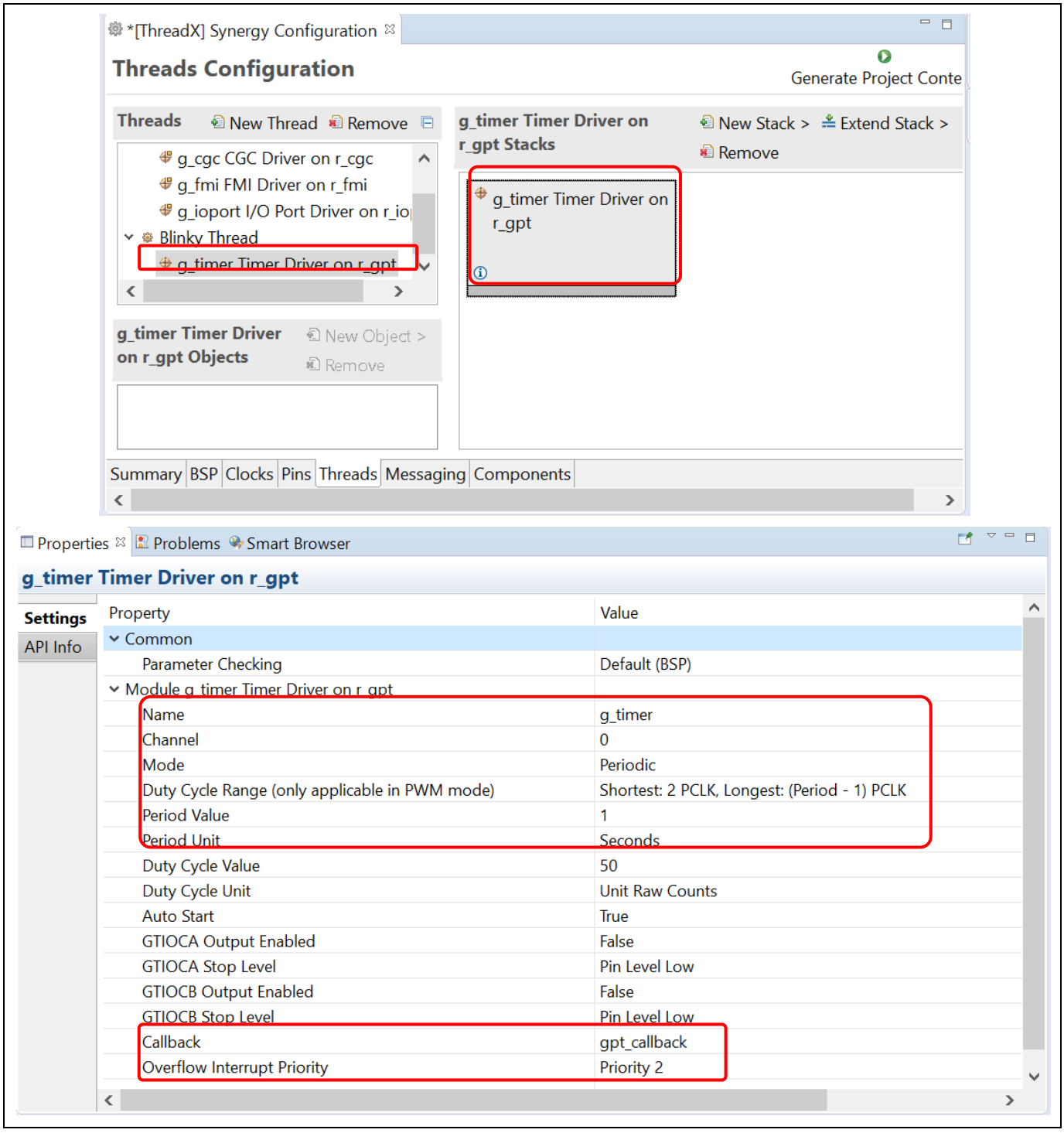


Figure 6-4. Setting up a ThreadX Application – GPT module configuration

2. Add a semaphore object to the **Blinky Thread** by selecting the **Blinky Thread** in the **Threads** panel and select → **Semaphore** in the **Objects** panel.

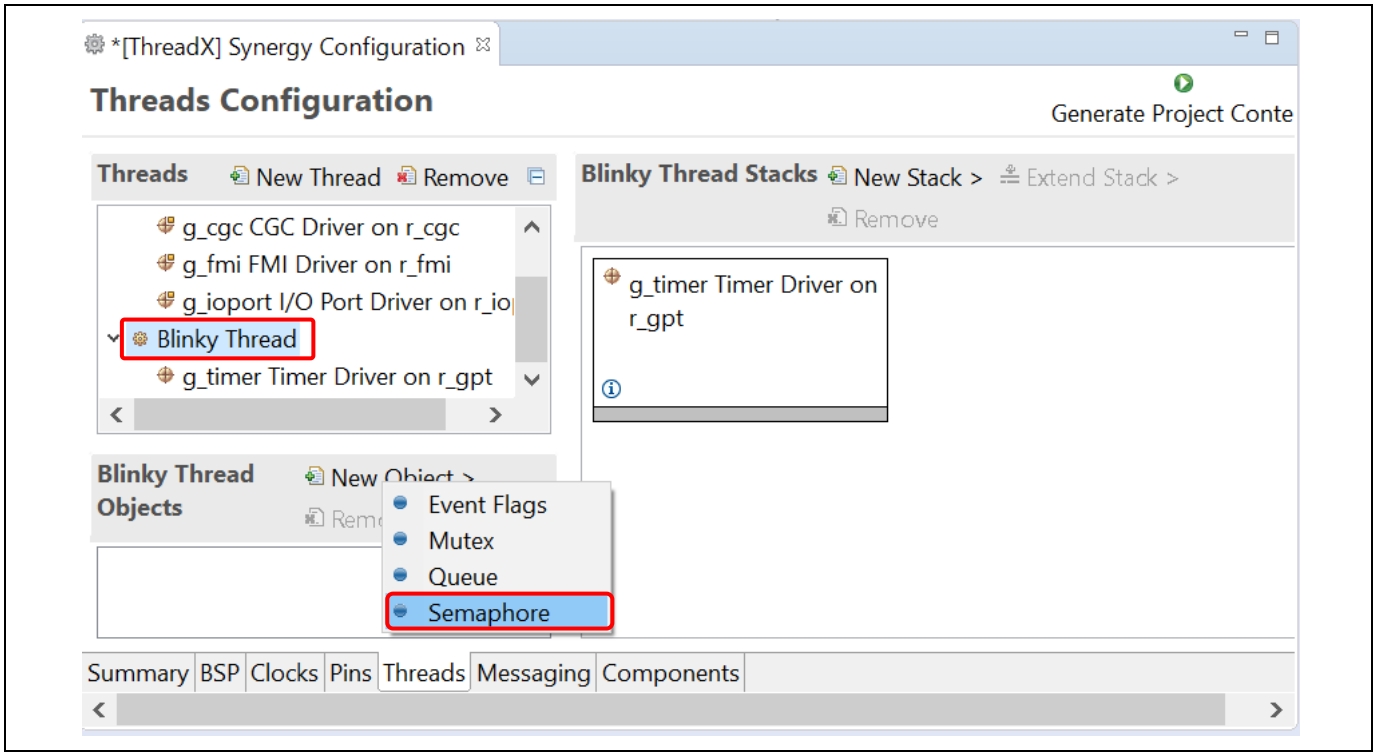


Figure 6-5. Setting up a ThreadX Application – Adding a Semaphore Object

3. Configure this newly created semaphore as follows:
   - Name: Blinky Semaphore
   - Symbol: g_blinky_semaphore
   - Initial count: 0

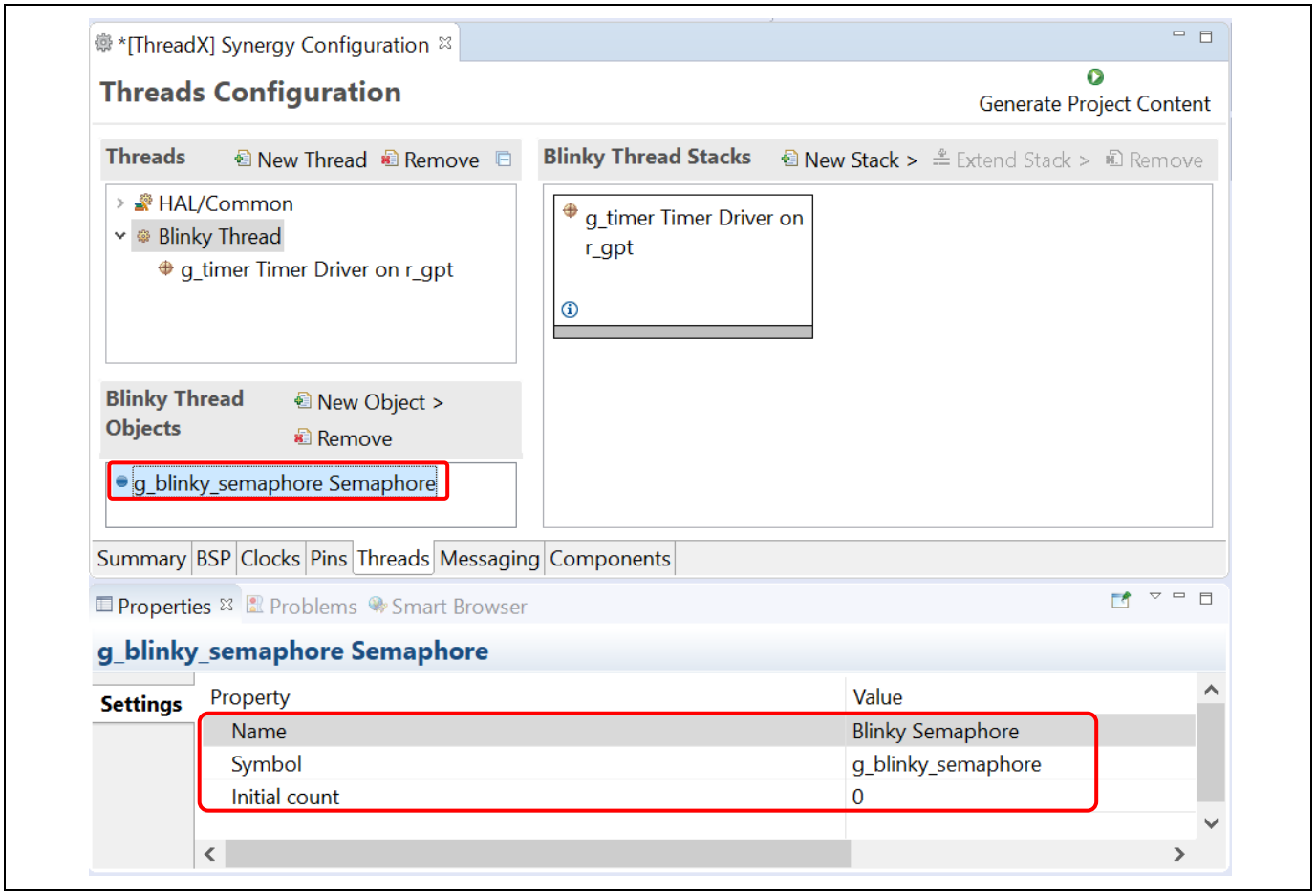


Figure 6-6. Setting up a ThreadX Application – Semaphore Object Configuration

Press **Ctrl+S** to save the setting and click the **Generate Project Content** Generate Project Content button to generate source code content.

4. Open `blinky_thread_entry.c` and implement the following contents:
   - Add source code to initialize the GPT module before the `while(1)` loop in `blinky_thread_entry()`.

     ```
     g_timer.p_api->open(g_timer.p_ctrl, g_timer.p_cfg);
     ```
   - Delete the thread sleep instruction and add code to wait for the semaphore in `blinky_thread_entry()`.

     ```
     tx_semaphore_get(&g_blinky_semaphore, TX_WAIT_FOREVER);
     ```
   - Implement the `gpt_callback()` function to signal the semaphore for the Blinky thread.

     ```
     void gpt_callback(timer_callback_args_t * p_args){
        tx_semaphore_put(&g_blinky_semaphore);
     }
     ```

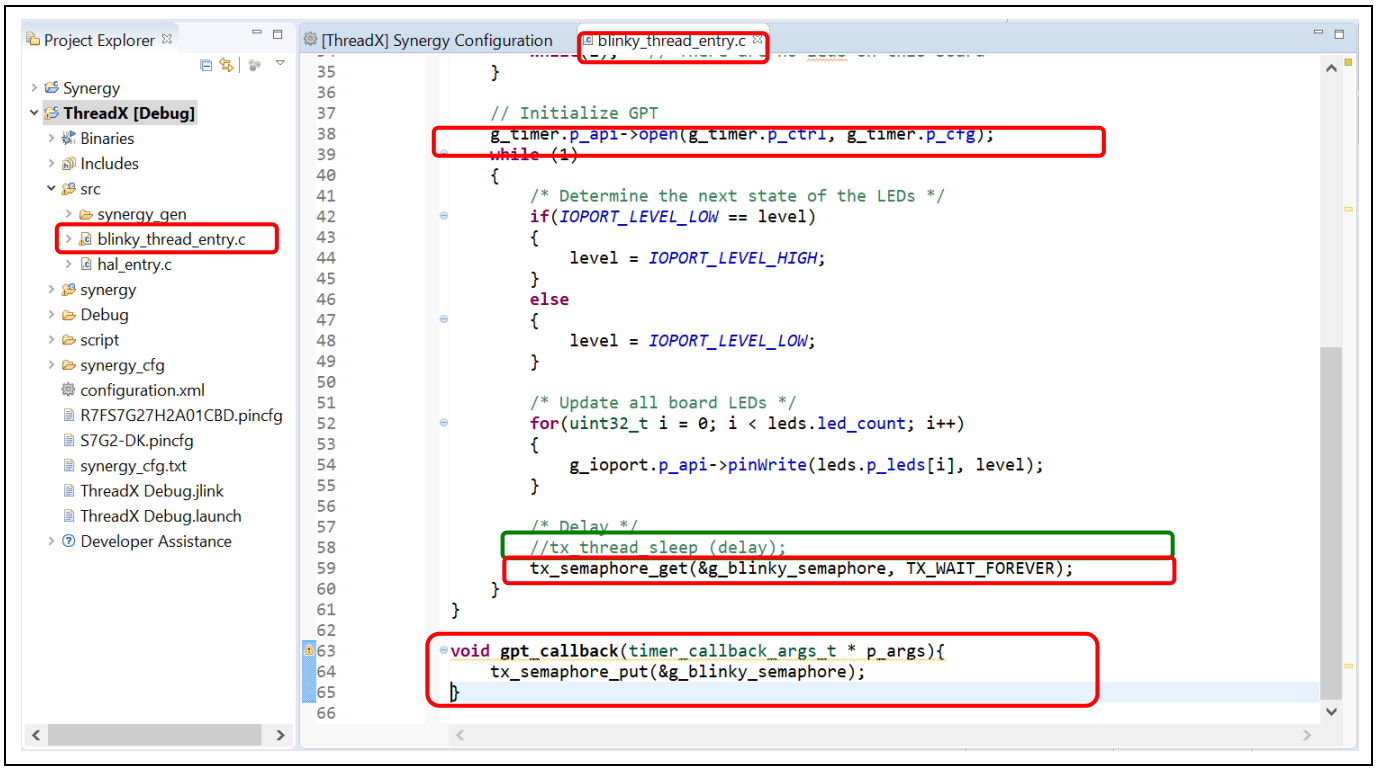


Figure 6-7. Setting up a ThreadX Application – Adding User Source Code

5. Build and run the project on the Synergy SK-S7G2 board. Confirm that the LEDs are turned ON/OFF every 1 second.

7. 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 an online forum dedicated to the e² studio from here.

Click on **Help** tab to open the **Help** menu.

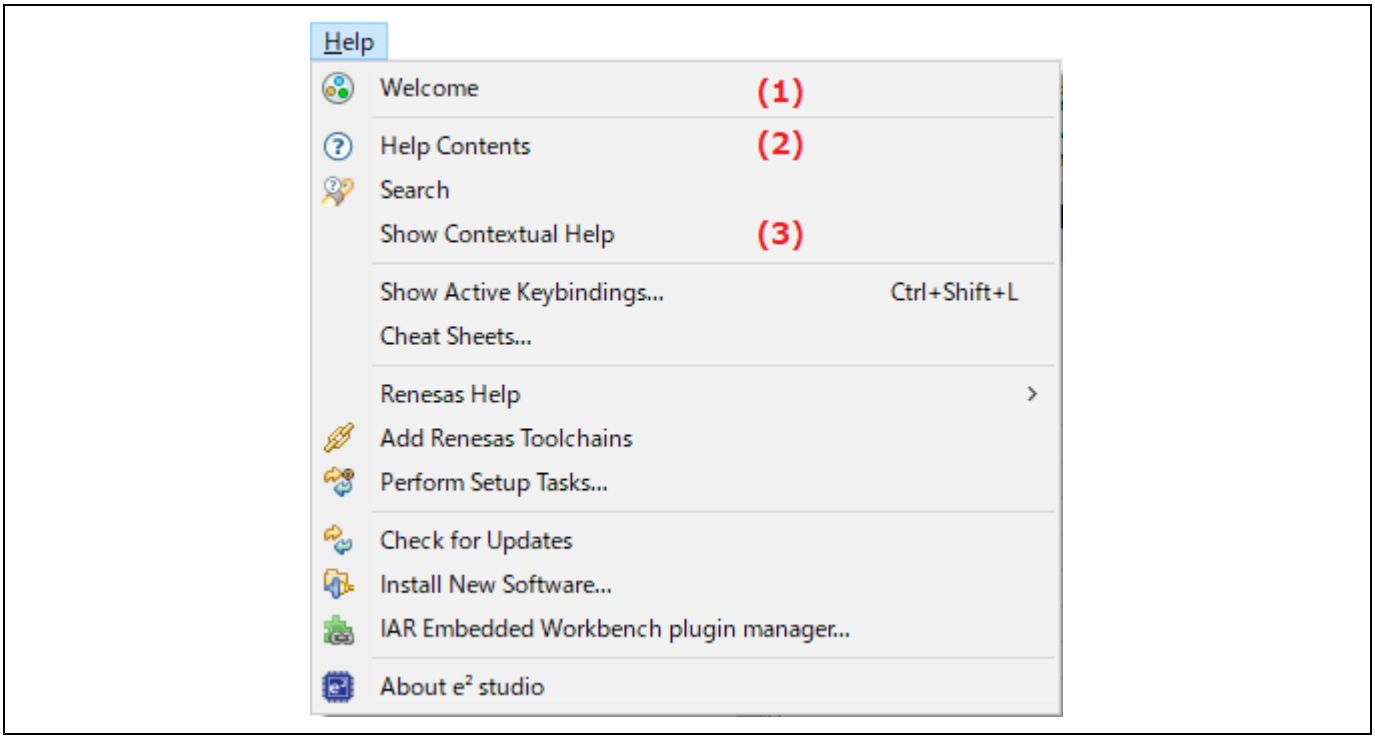


Figure 7-1. Help – Help Menu

Quick Help Tips:

(1) Click **Welcome** for an overview of the e² studio and to view Release Notes.

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

(3) Click **Show Contextual Help** to open the **Help** view within the workbench.

Under the **Help Contents** window, there are many useful topics such as:

- The "e² studio Debug Help" topic which provides useful information such as debug configuration, supported number of breakpoints, etc.
  It can be launched by clicking on the **Help** menu → **Help Contents** → **e² studio Debug Help**.
- The "Synergy Contents" topic which provides information about Synergy project creation, using the Synergy Configuration Editor and FAQs.
  It can be launched by clicking on the **Help** menu → **Help Contents** → **Synergy Contents**.

Revision History	Renesas Synergy™ e² studio User's Manual: Quick Start Guide

Rev.	Date	Description	
		Page	Summary
1.00	Jul.20.2021	-	Initial release
1.10	Apr.29.2022		Fixed by version upgrade of e² studio and SSP.
		-	Removed "2.4 Confirmation of Synergy license" (SSP 2.2.0 no longer requires license keys)
		16	"3.1 Generating a New Synergy Project" changed due to e2 studio updates.
		84	Help menu changed due to e2 studio updates.

Renesas Synergy™ e^2 studio User's Manual: Quick Start Guide

Publication Date: Rev.1.10 Apr.29.2022

Published by: Renesas Electronics Corporation

Renesas Synergy™

Renesas Electronics Corporation

R20UT5036EJ0110