User's Manual

Renesas Synergy™ e^2 studio v7.3 or Higher Getting Started Guide

User's Manual

Renesas Synergy™ Platform

Synergy Tools & Kits

Integrated Solution Development Environment (ISDE)

Renesas Electronics
www.renesas.com

Rev.1.00 Aug 2019

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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. 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.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Renesas Synergy™ Platform

e² studio Getting Started Guide

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 IDE 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 and 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 e² studio. Multiple views and editors are available to support specifically 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

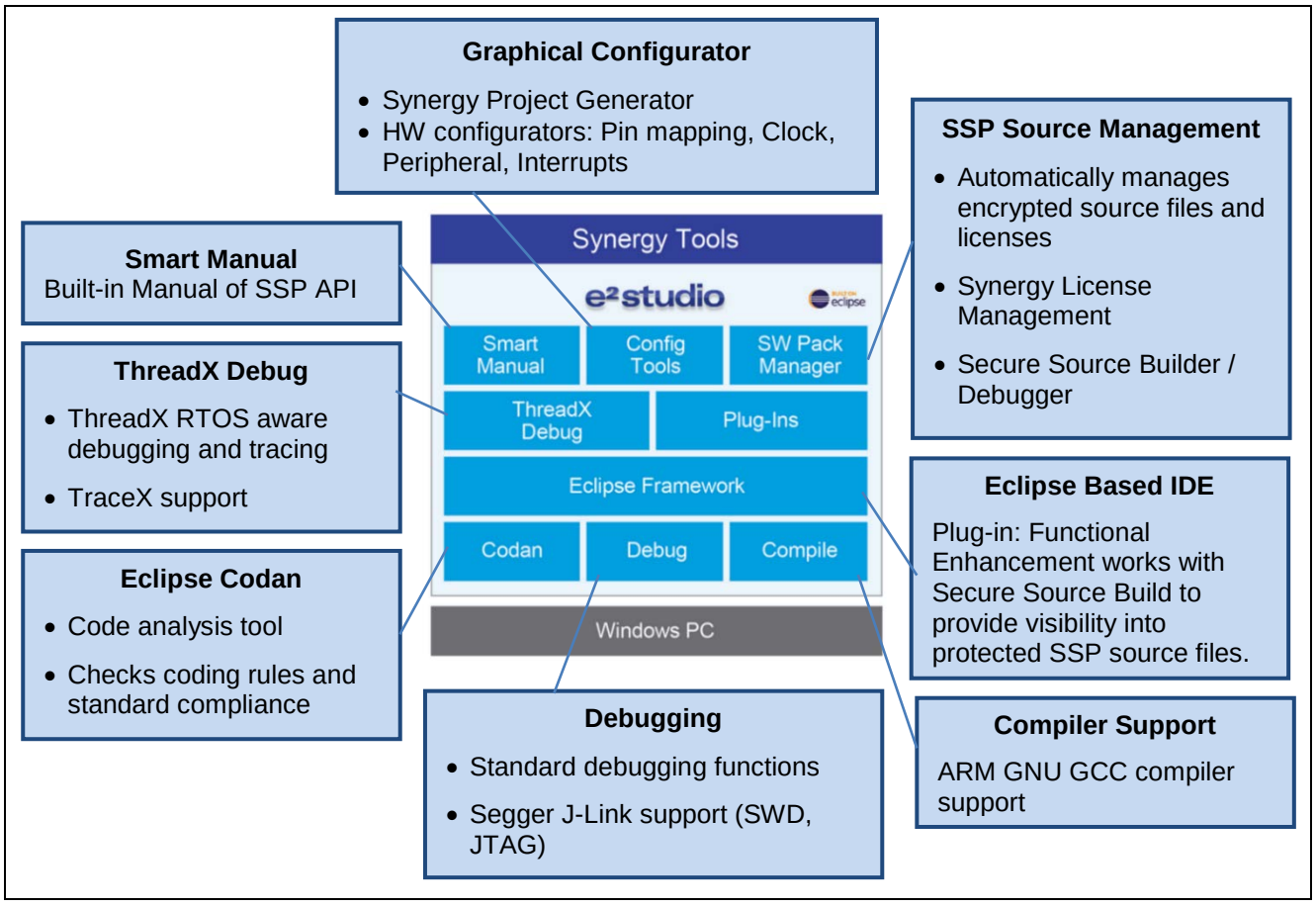


Figure 1. Renesas Synergy™ in e² studio

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

Table 1. e² studio Features Comparison

Feature	Renesas Synergy™	RX / RL78 / RZ / RH850
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	RX family (GNURX-ELF and Renesas CC-RX build plug-ins) RL family (GNURL78-ELF, Renesas CCRL build plug-in) RZ family (GNUARM-NONE-EABI) RH850 family GHS (*supported Debug only)
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	Express Logic 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.

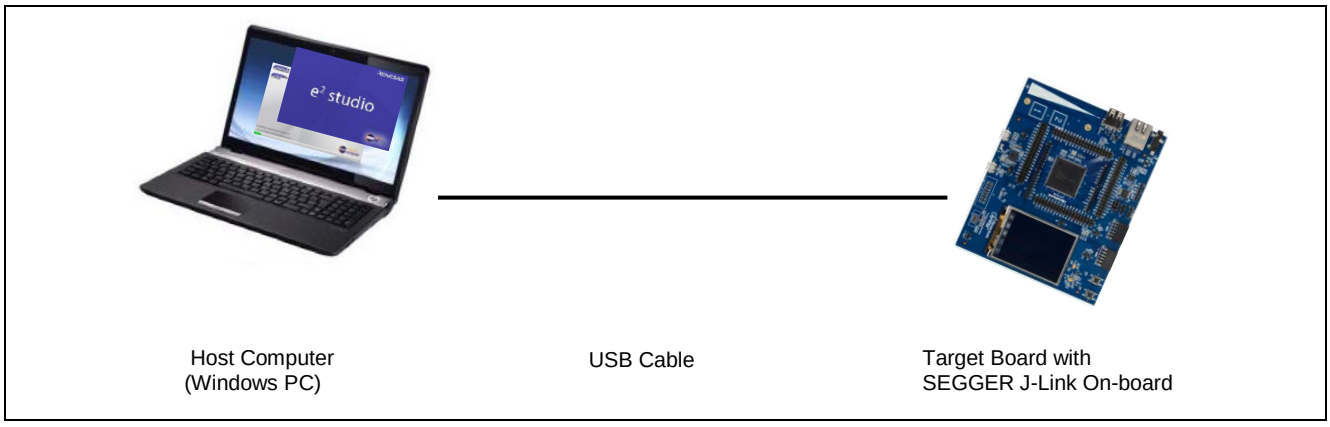


Figure 2. System configuration

1.2 Operating Environment

Following are the system requirements for this product.

1.2.1 System Requirements

Host Computer	• Processor: At least 2 GHz (with Intel® Core™ family processor) • Memory capacity: At least 1 GB (8 GB or larger is recommended) • Hard disk capacity: Minimum 250 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 7 (32/64-bit OS) • Windows 8.1 (32/64-bit OS) • Windows 10 (32/64-bit OS).

1.2.2 Supported Toolchains

GNU Arm® compiler (version: GCC_4.9_2015q3 and GCC_7.2_2017q4)

1.2.3 Supported Emulator Device

Segger J-Link

1.3 Outline of a Synergy Project Development

This document provides detailed instructions on how to start developing with Renesas Synergy™ platform. The main steps are outlined as follows. By understanding the main steps that follow, you can relate better to the procedures described in section 3 and section 4.

1. Generating a Synergy project.
2. Configuring the Synergy project to fit hardware specifications, such as clock, ICU, pin functions, and so forth.
3. Configuring the ThreadX OS.
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 *Solutions Gallery* of the Synergy Platform website https://www.renesas.com/products/synergy.html

2.1 Installing the Platform Installer

The Platform Installer includes the following:

- Synergy Software Package (SSP),
- e² studio or the IAR Embedded Workshop® for Renesas Synergy™ (IAR EW for Synergy) ISDE,
- GCC Arm embedded compiler, and
- SEGGER J-Link Drivers.

To download and install the Platform Installer, use the following steps:

1. Visit the *Solutions Gallery* page at the Synergy Platform website. Navigate to **Software** -> **Synergy Software Package** or **Software Tools** -> **e² studio** and select the option **Download Platform Installer** (login to My Renesas account is required).

Figure 3. Installation – Download the Platform Installer

2. Select e² studio as the development environment.

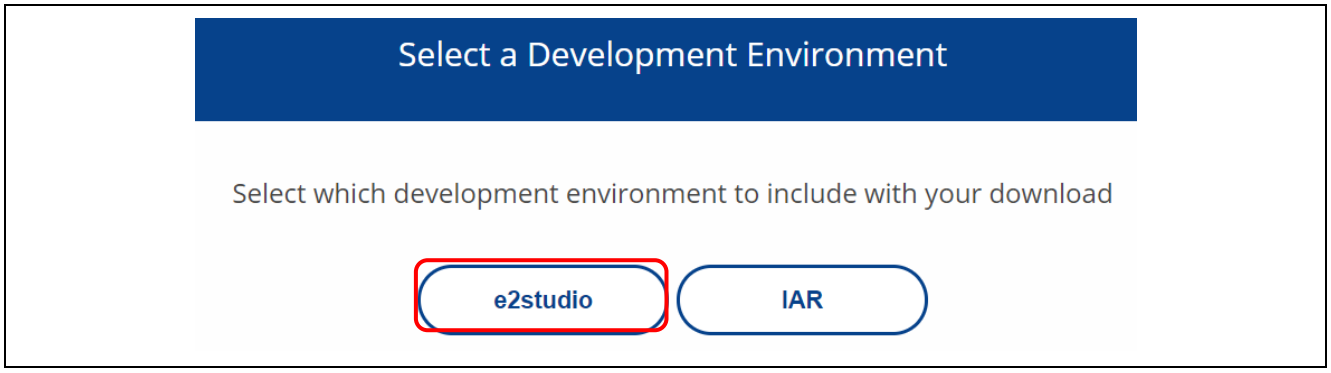


Figure 4. Installation – Select e² studio Development Environment

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

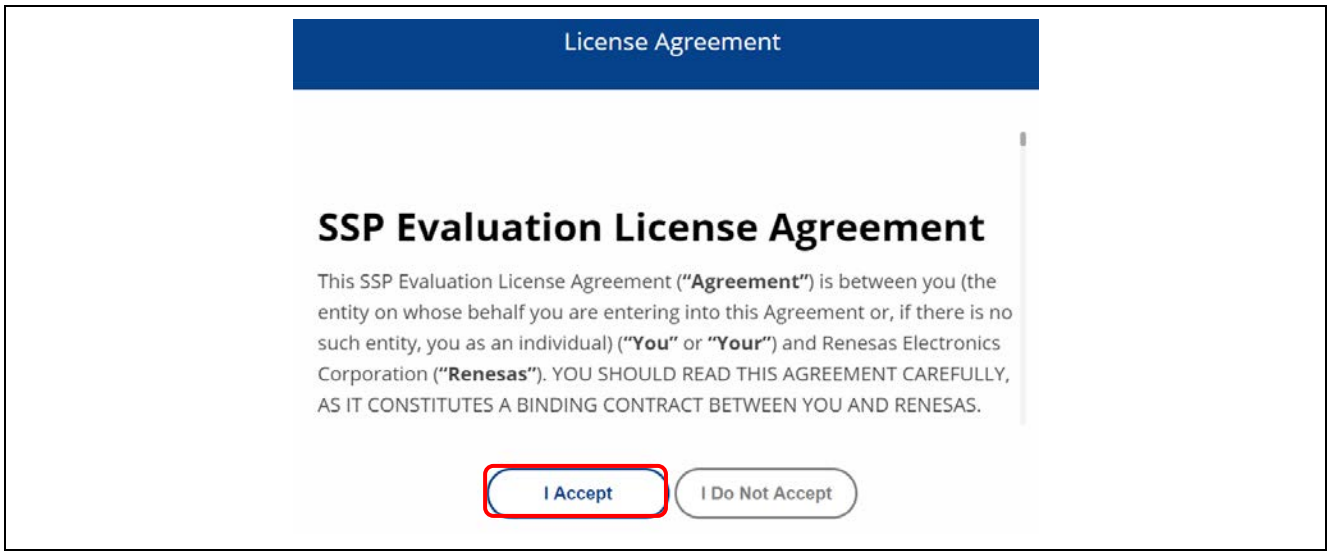


Figure 5. Installation – Accept the License Agreement

4. Unzip and run the installation file.
5. In the **Select Install Type** page, choose **Quick Install**, and click **Next**. If users would like to customize the components to be installed, choose **Custom Install** then click **Next**.
   New users are recommended to select the **Quick Install** option to minimize the configuration steps needed. This option will install e² studio, SSP, and GCC Arm Embedded by default. If the user selects **Quick Install**, then step 7 is not shown.

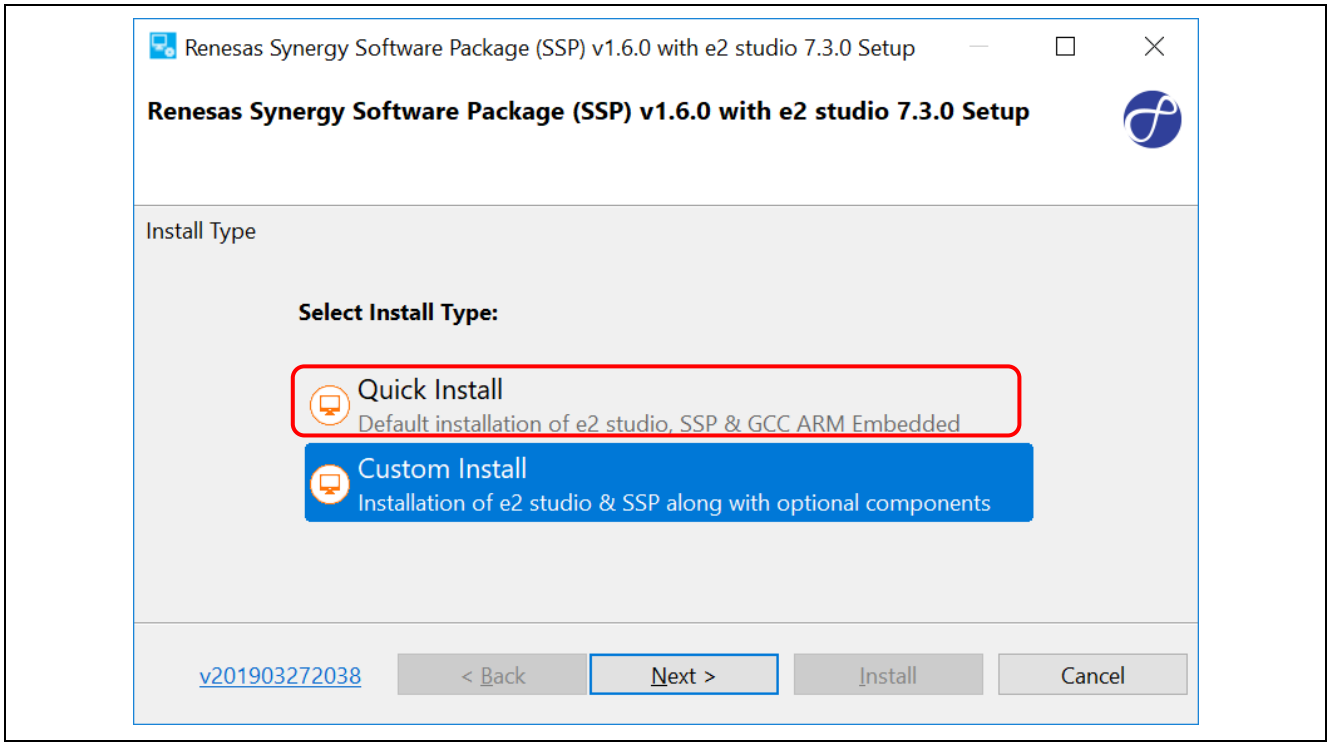


Figure 6. Installation – Select Install Type

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

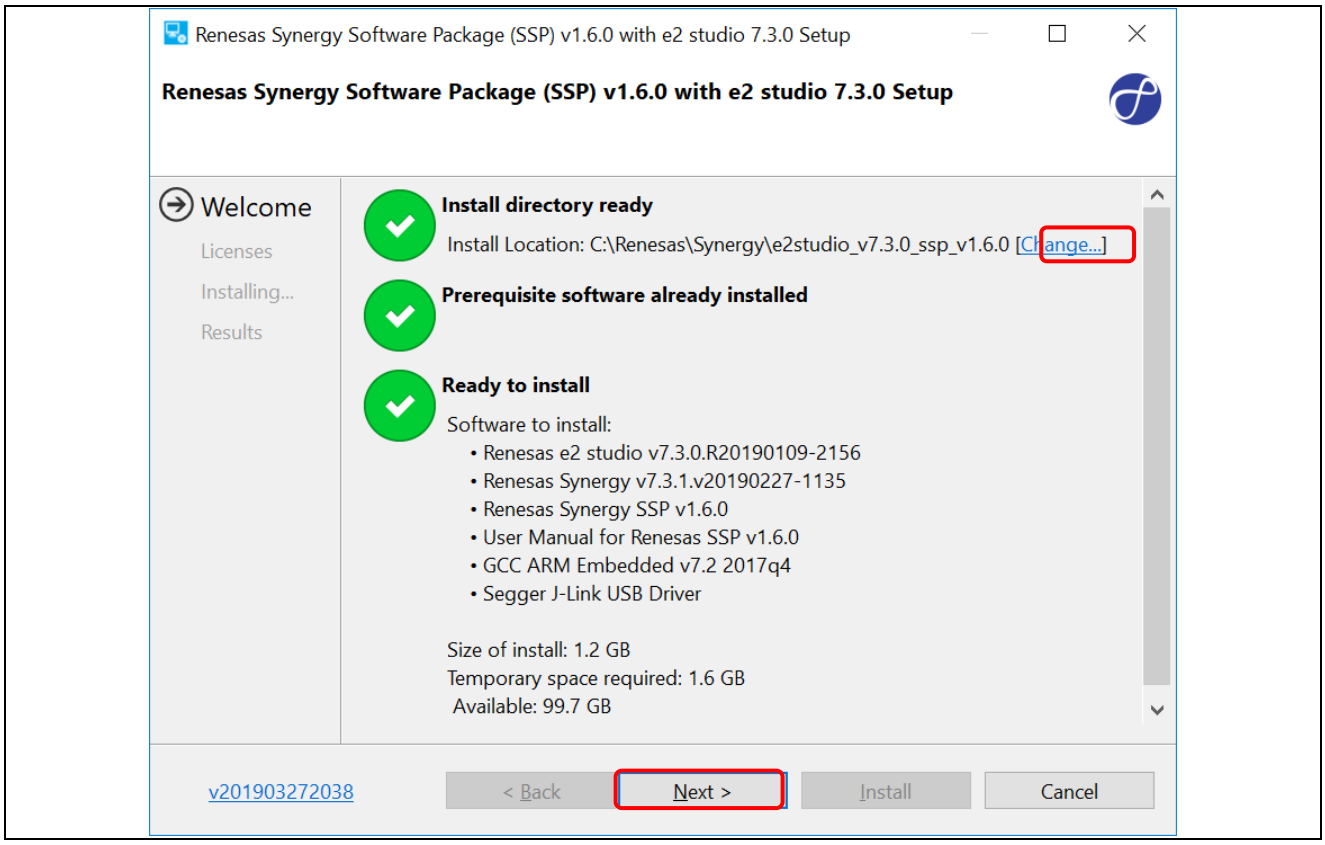


Figure 7. Installation – Welcome Page

7. In the **Optional Components** page, select the components to be installed, then click **Next**. This page will not be shown if the user selects **Quick Install**.

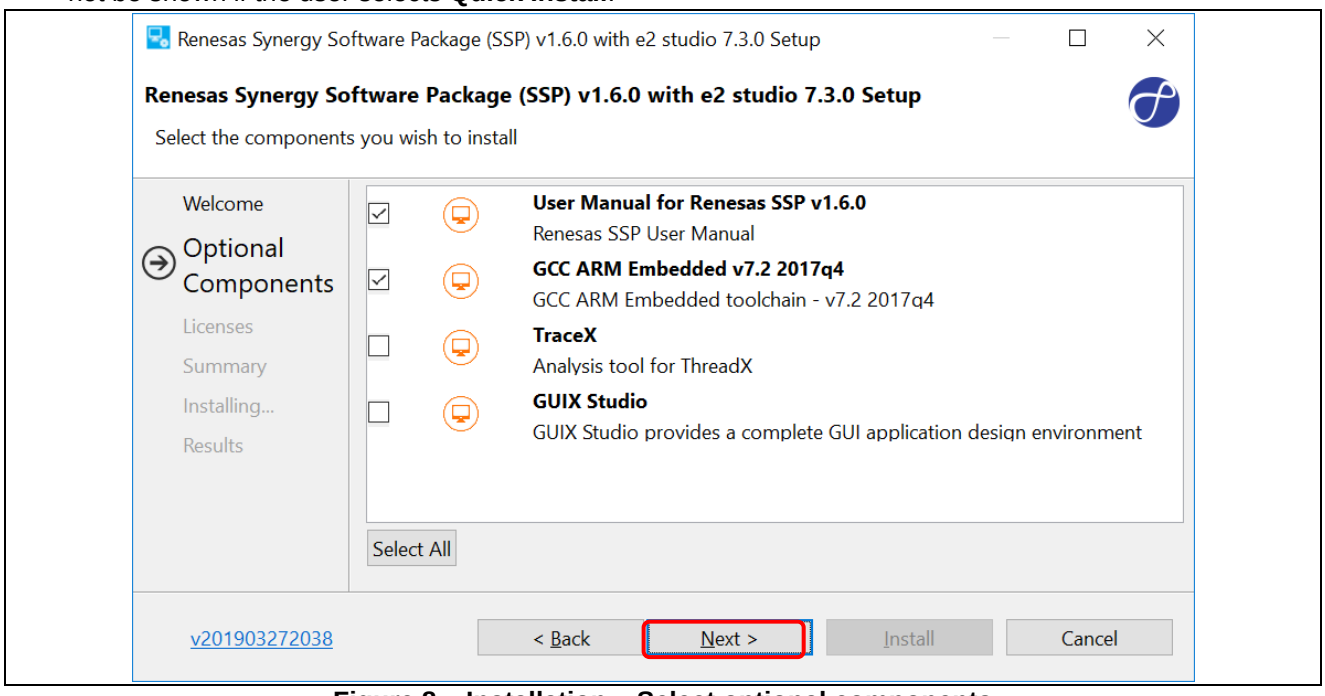


Figure 8. Installation – Select optional components

8. Tick the checkbox to accept the license agreement, click **Install** to continue.

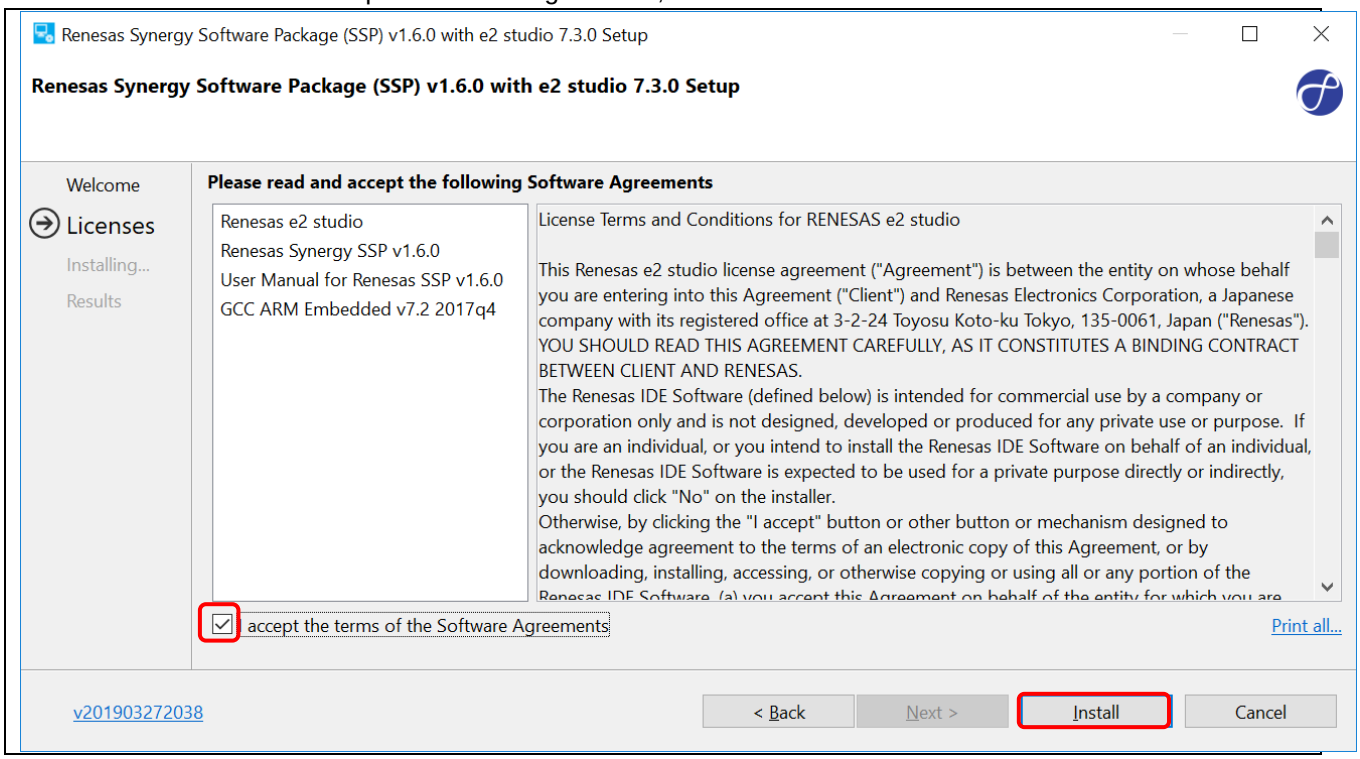


Figure 9. Installation – Software Agreements

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

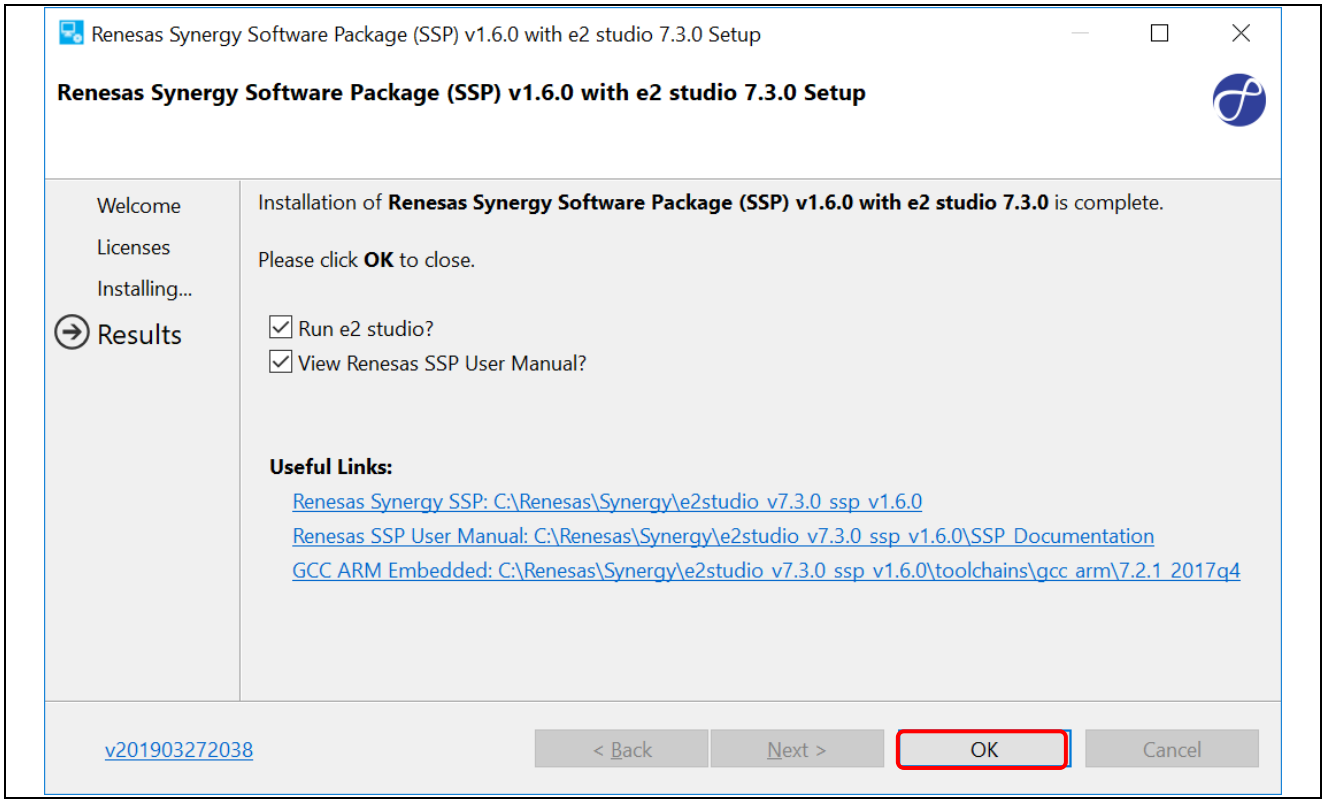


Figure 10. Installation – Complete Installation

2.2 Installing the Standalone Installer

This section describes installation of the following development tools separately using the standalone installer.

- e² studio ISDE
- GCC Arm Embedded Compiler
- Renesas Synergy™ Software Package (SSP)

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

2.2.1 Installing e² studio

To install e² studio for Synergy, follow these steps:

1. Click **Solution Gallery** on the Synergy Platform website https://www.renesas.com/products/synergy.html.
2. In the Solution Gallery page, navigate to **Software** and select **Development Tools**, then e² studio.
3. In the e² studio page, select Download Standalone Installer.

Figure 11. Installation – Download the Standalone Installer

4. Click **I Accept** in License Agreement to download a standalone e² studio Installer **setup_e2studio_<version>.zip**. (you may need to sign in MyRenesas account to enable the 'Download Standalone Installer').

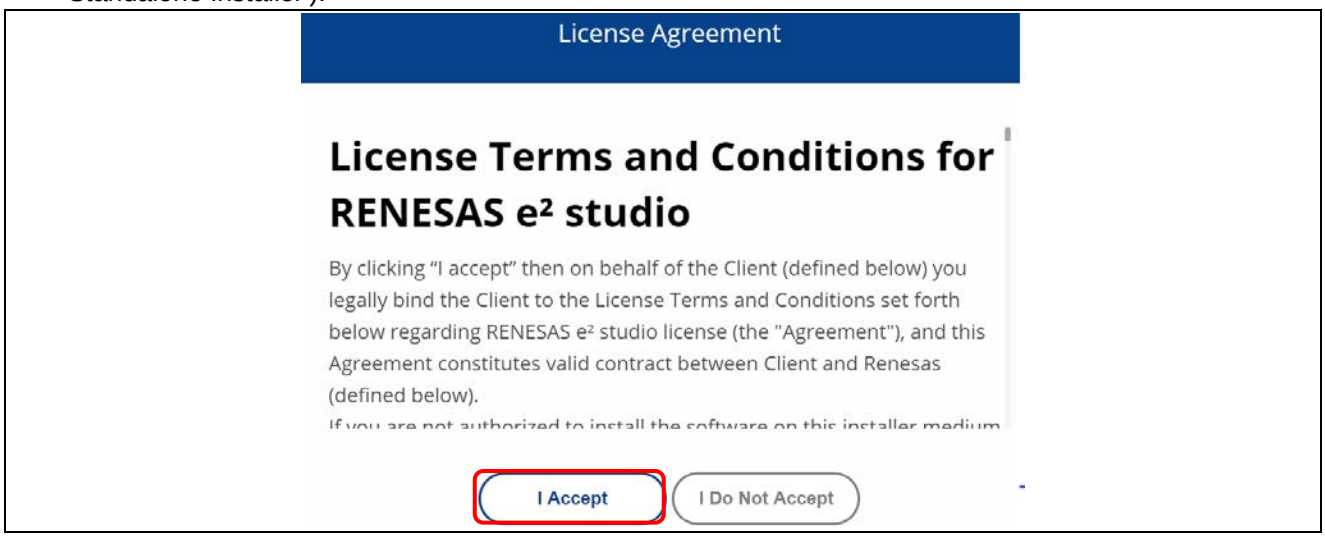


Figure 12. Installation – Accept the License Agreement

5. Unzip the download file and run the e² studio installer to invoke the e² studio installation wizard page. Click the **Next** button to continue.

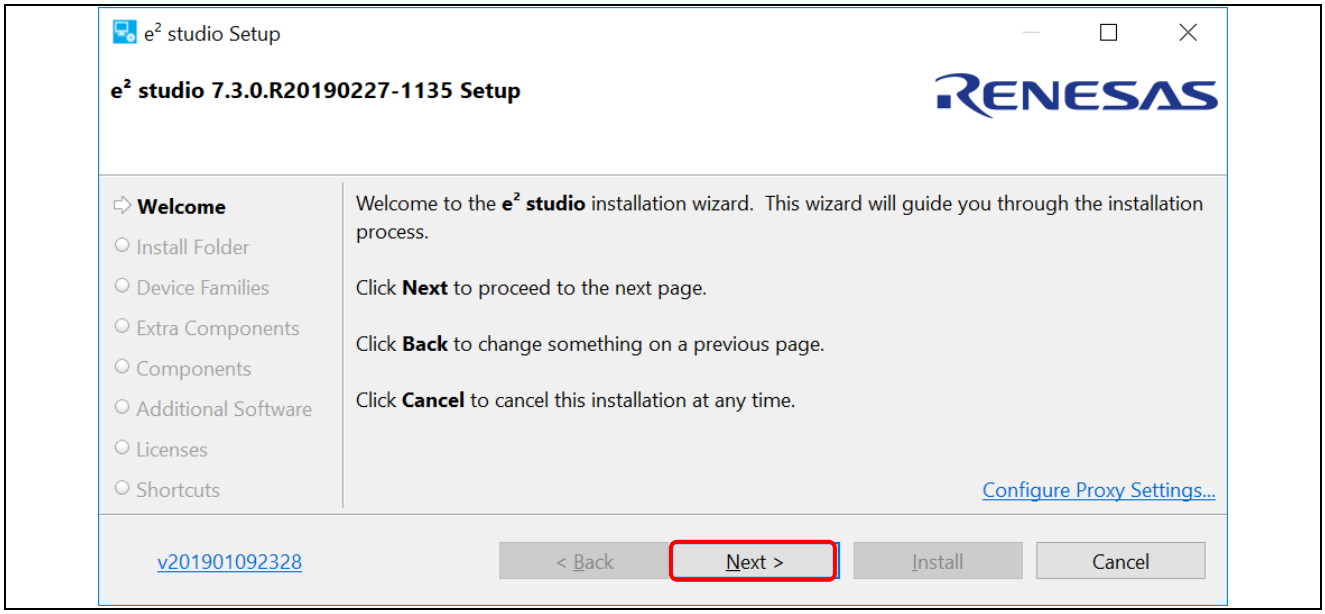


Figure 13. Installation – Welcome page

6. **Install Folder**:
   The default installation location is set to: `C:\Renesas\e2_studio`. Input install folder directly to the text box or click **Browse…** button to modify it.

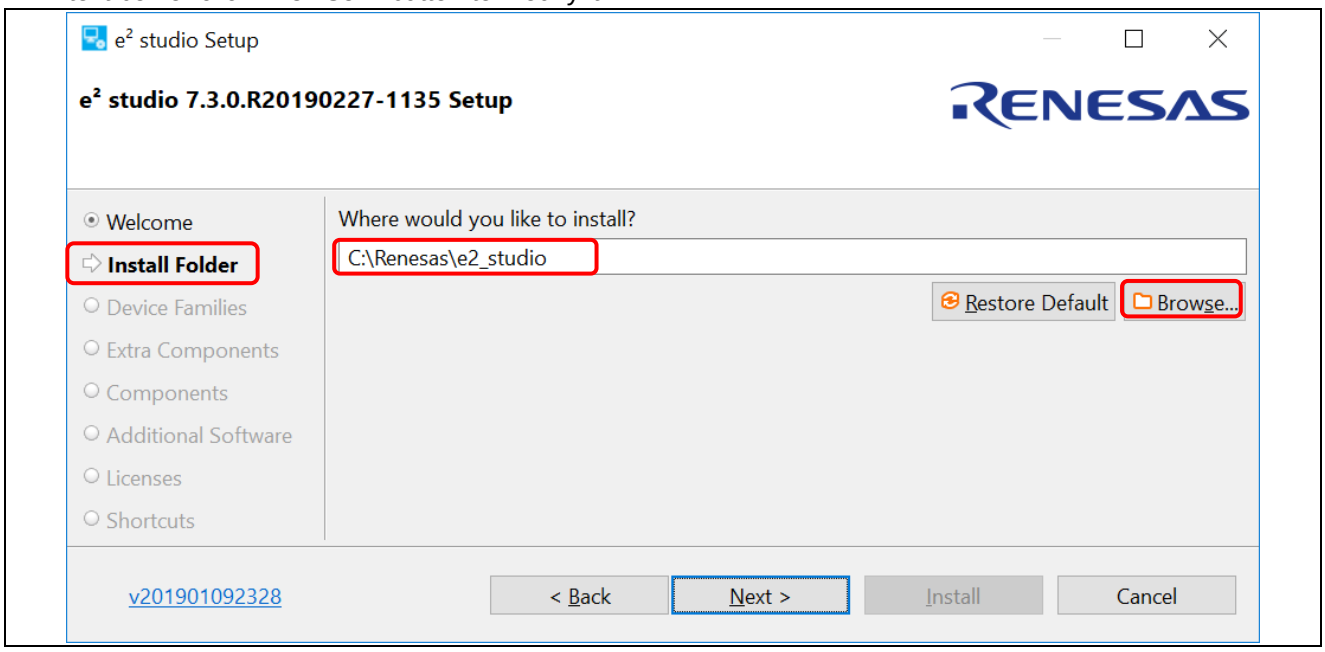


Figure 14. Installation – Install Folder

7. **Device Families**:
   Check the box for Renesas. Checkboxes of other device families are optional.
   Click the **Next** button to continue.

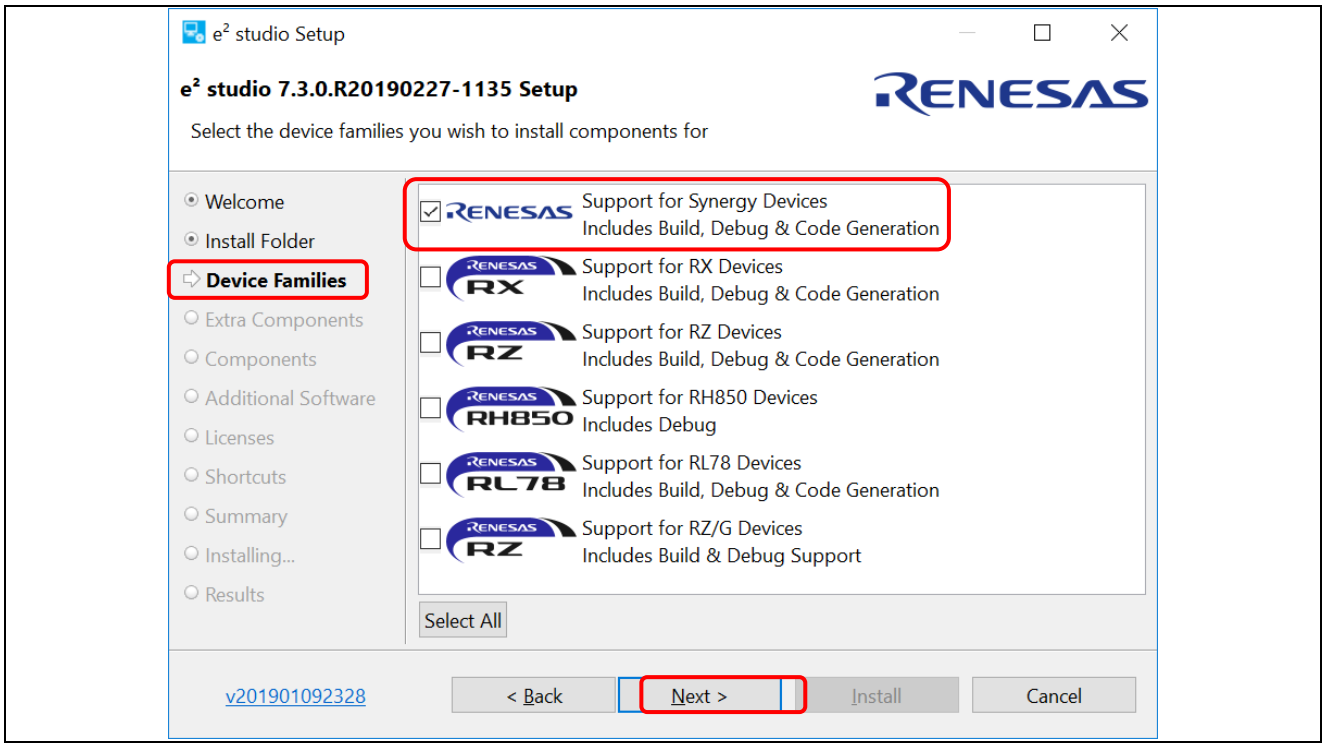


Figure 15. Installation – Device Families

8. **Extra Components**
   Select Extra Components (that is, language pack, SVN & Git support, RTOS support) to install. These components are optional. Click the **Next** button to continue.

9. **Components**
   Ensure that **Renesas Synergy** and **Renesas Synergy Build Support Files**, **Renesas Synergy Debug Support Files** are checked.
   Click the **Next** button to continue.

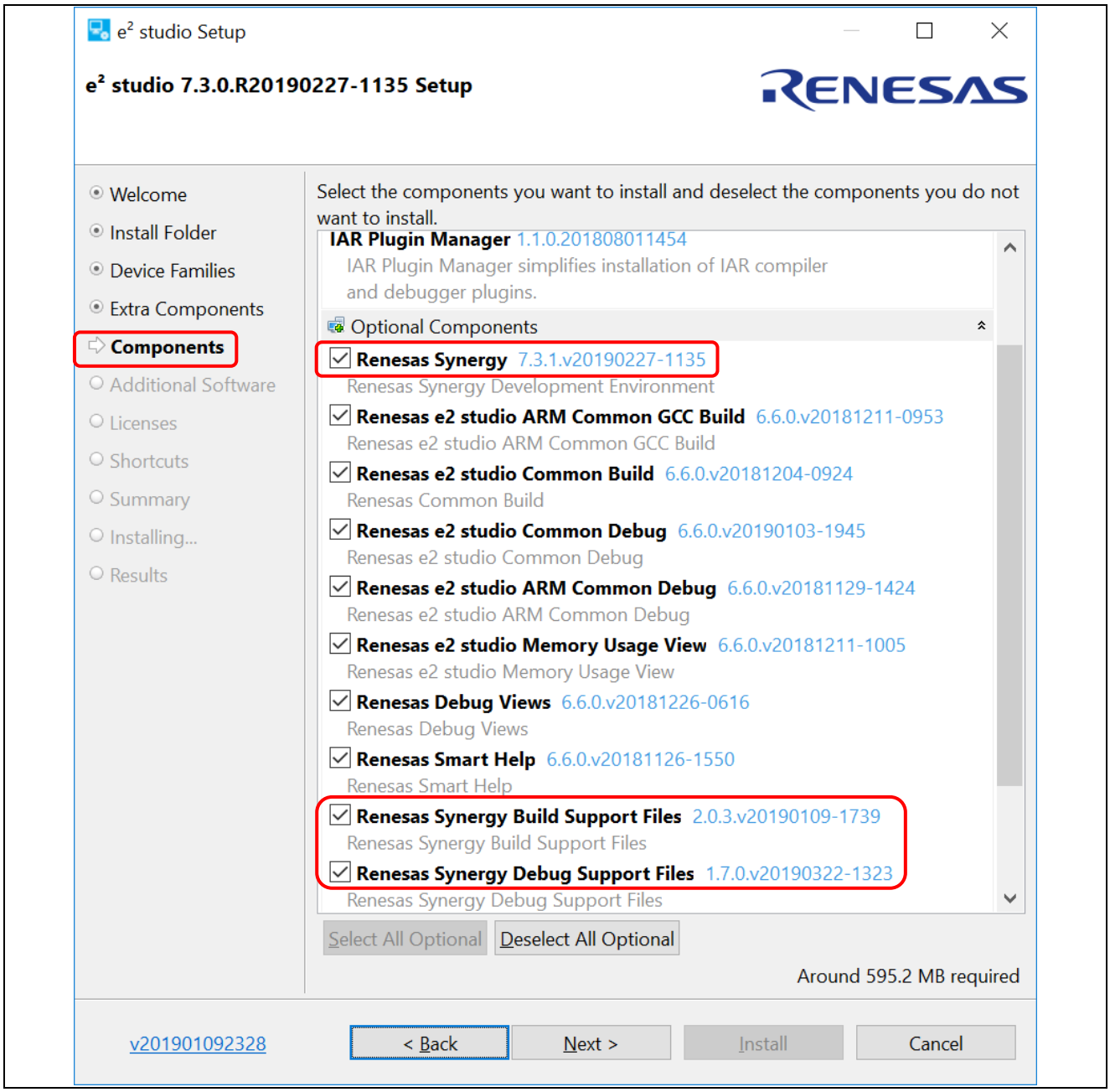


Figure 16. Installation – Components

10. **Additional Software**

Check the checkbox for the latest version of GCC Arm Embedded in the Additional Software dialog (the latest versions of GCC Arm Embedded supported by SSP v1.6 are 4.9 2015q3 and 7.2 2017q4). The older versions are optional.

Click the Next button to continue.

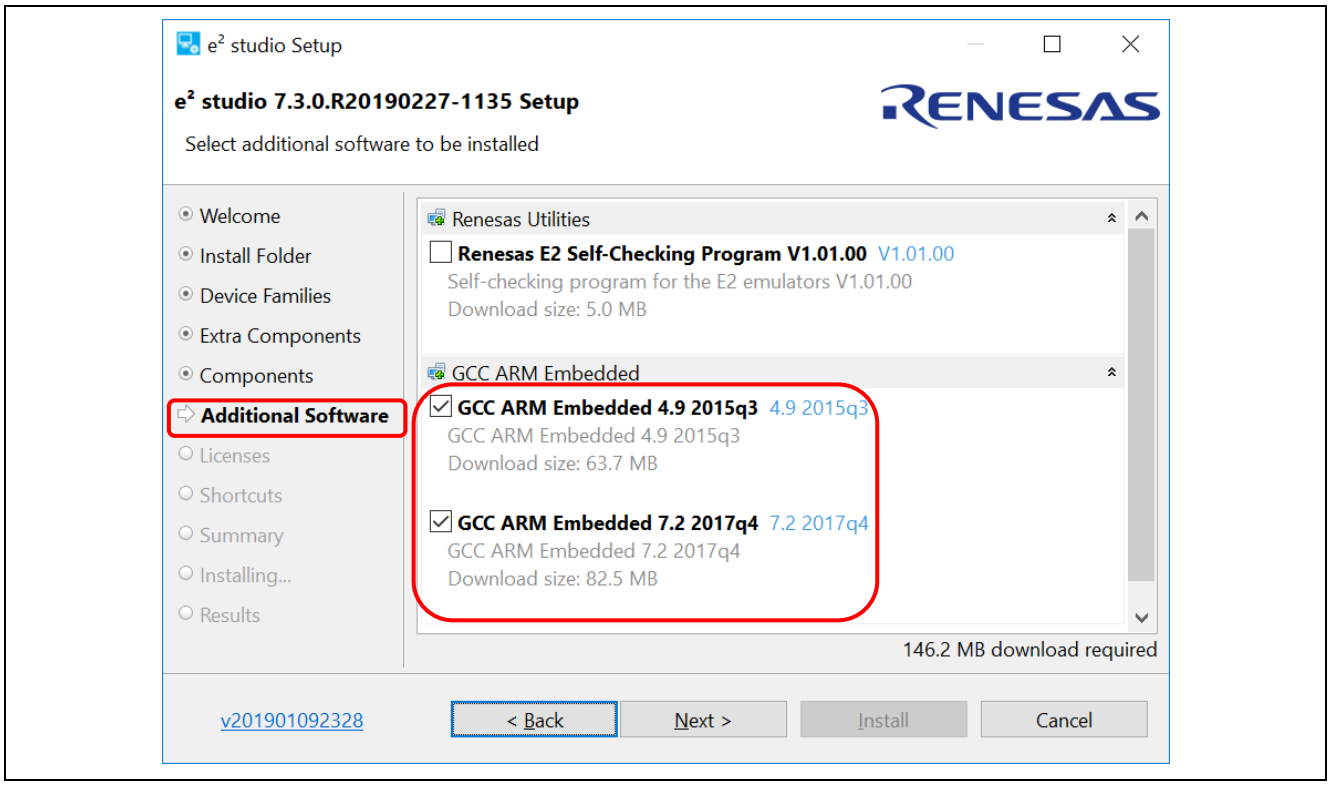


Figure 17. Installation – Additional Software

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

The **GCC Arm Embedded…** will be installed. Keep all default settings throughout the installation and check **Add path to environment variable** at the final dialog.

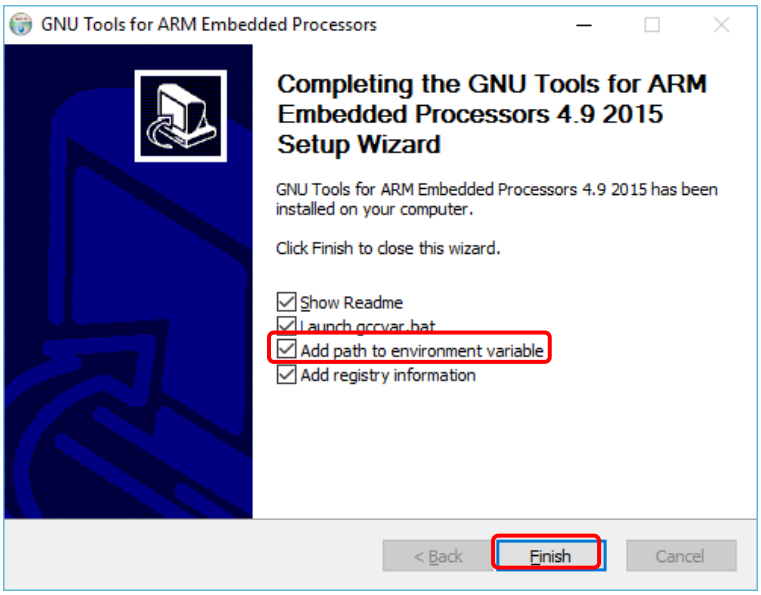


Figure 18. Installation – GCC Arm Embedded Compiler Installation

2.2.2 Setting Up the GNUARM Compiler

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

1. Download the latest version of the GNU Arm compiler supported by Renesas Synergy™ (currently v7.2.1) 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. Click **Solution Gallery** in Synergy Platform website www.renesas.com/synergy.
2. In the Solution Gallery page, select **Software** and select **Synergy Software Package**
3. In the Synergy Software Package page, select **Download Standalone Installer** to download the file `SSP_Distribution_<SSP-version>.zip`. (Sign in to MyRenesas account is necessary to enable the **Download Standalone Installer** option).

The Release Note and SSP User Manual can also be downloaded from the Synergy Software Package page.

Figure 19. Installation – Download the standalone SSP package

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

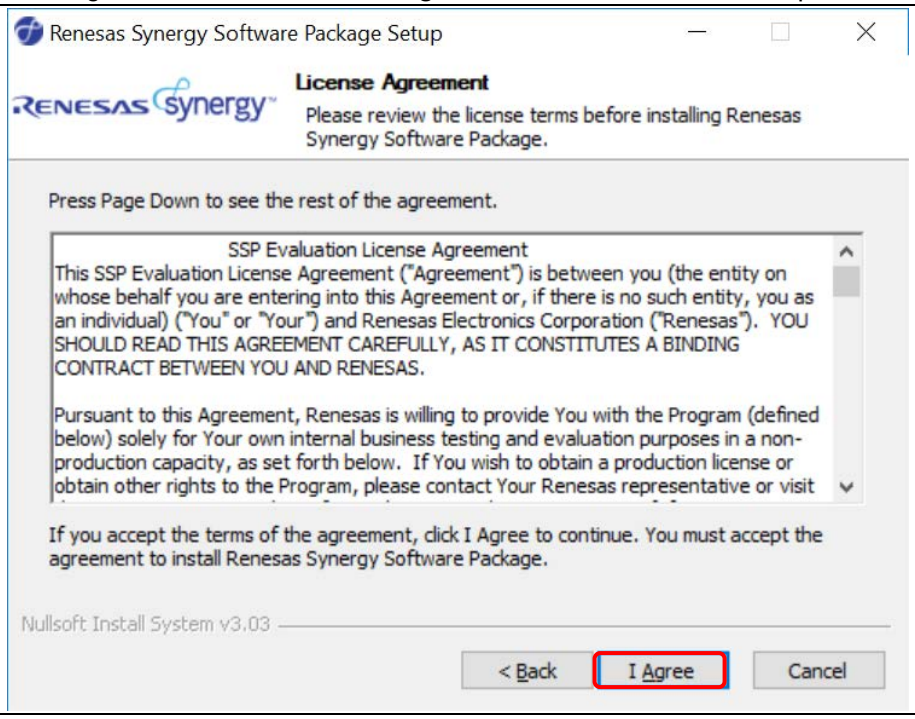


Figure 20. Installation - License Agreement

8. Use the default setting in the **Choose Components** dialog and click the **Next** button to continue.

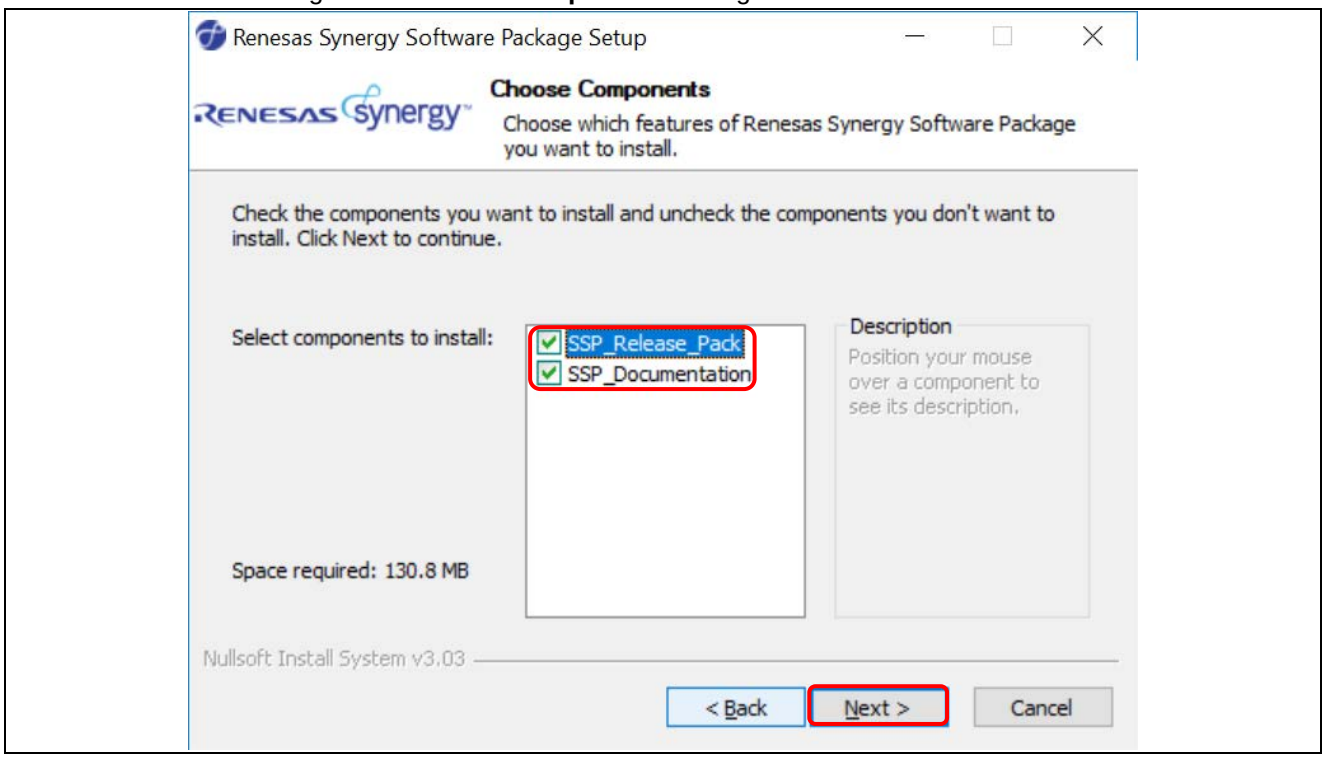


Figure 21. Installation – Choose components

9. 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 the **Install** button to start the installation.

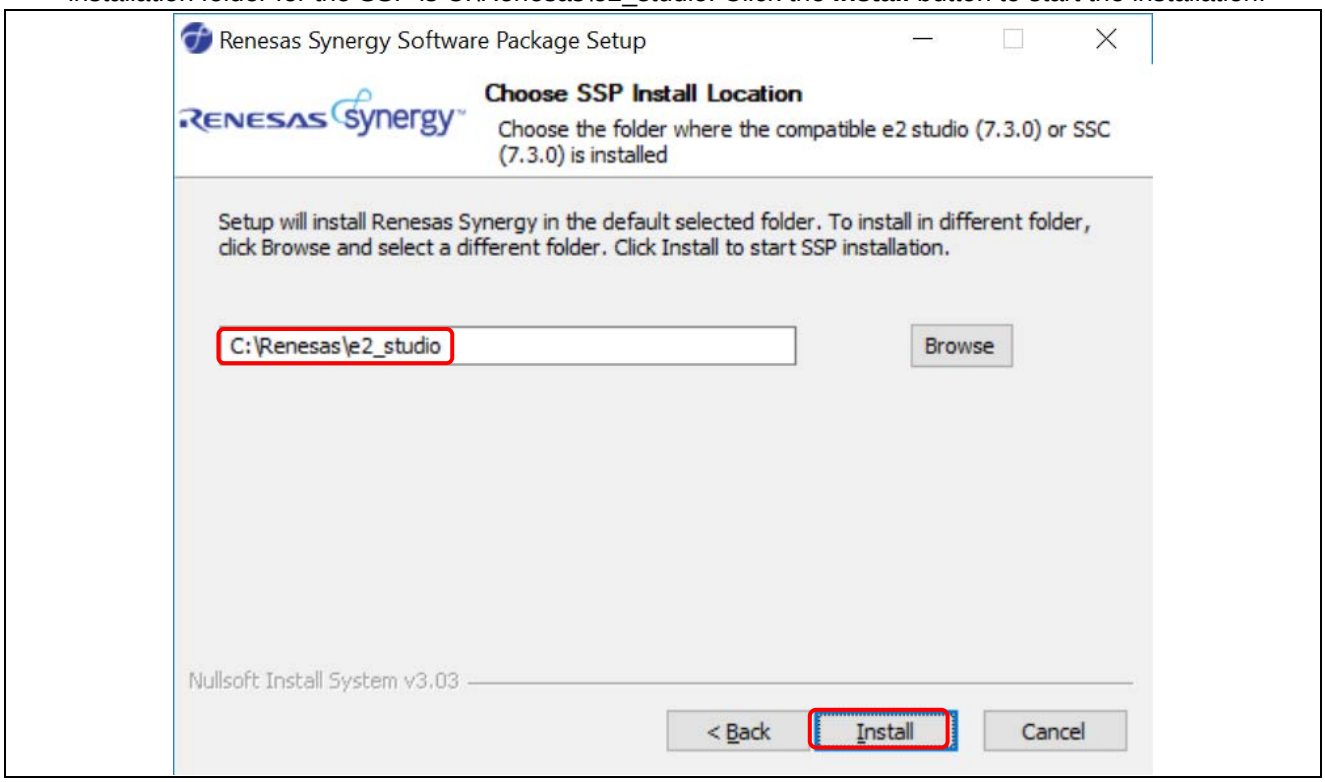


Figure 22. Installation – SSP Installation Folder Selection

10. During the SSP installation process, the SSP Documentation Installation wizard will prompt for the document installation. The default location for document installation is `C:\Renesas\Synergy`. Users can change the default location. Click the **Install** button to start the installation.

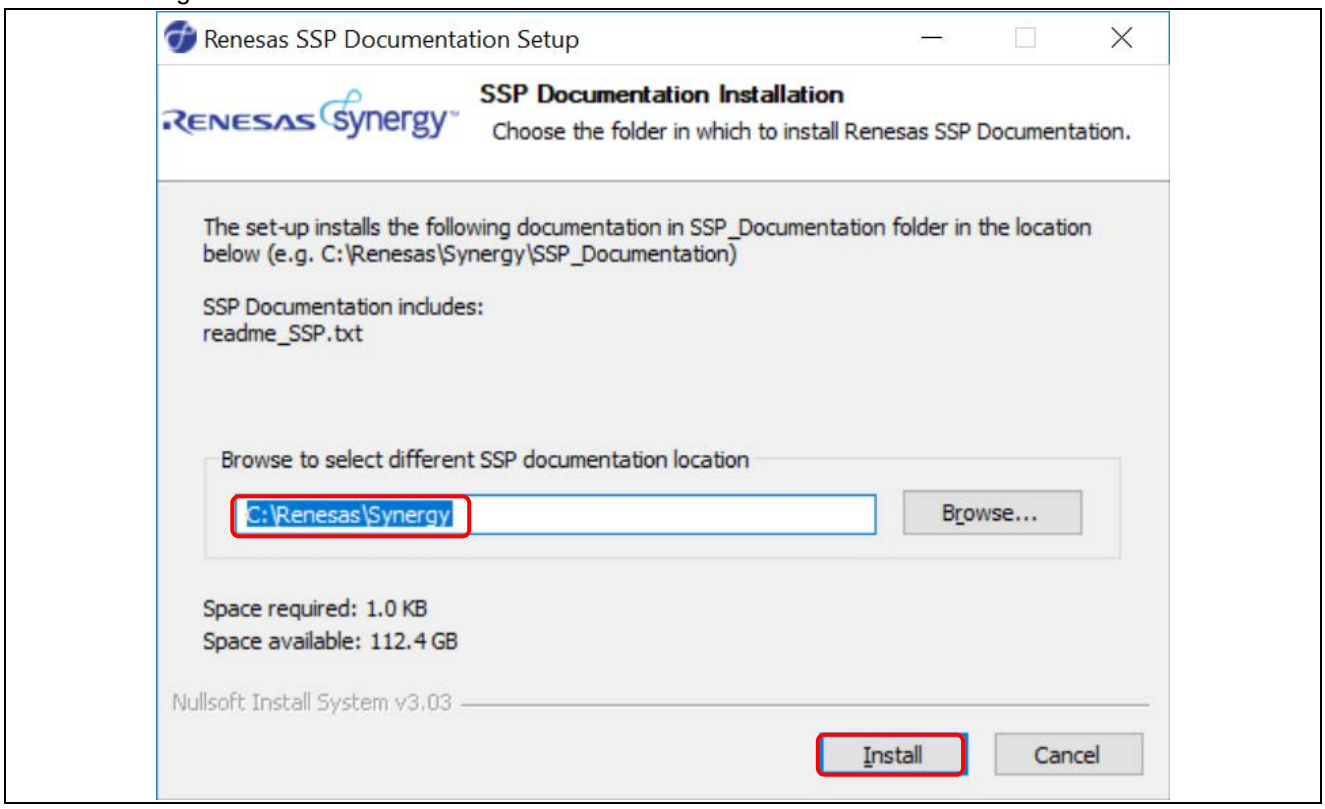


Figure 23. Installation – SSP Documentation Installation

11. 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 **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 un-installation, e² studio will be deleted from the installed location and the Windows shortcut menu is removed.

2.4 Updating e² studio

Updating to a new version requires installer download from Solutions Gallery of the Synergy Platform website https://www.renesas.com/products/synergy.html either platform installer or standalone installer).

Note that you should not overwrite an existing installation. Prior to the IDE upgrade, users must uninstall the old version of e² studio. However, to keep both old and new e² studio versions, users can create a new folder as installation destination for the new e² studio version.

2.5 Register Synergy license

Building and running a project requires a Synergy license (in XML format) to be registered in the e² studio. The registration information is stored in the workspace folder. Therefore, whenever a user switches to a new empty workspace, the registration needs to be repeated.

Follow the steps below to register a Synergy license.

1. On the menu bar, click **Help** → **Synergy License**.

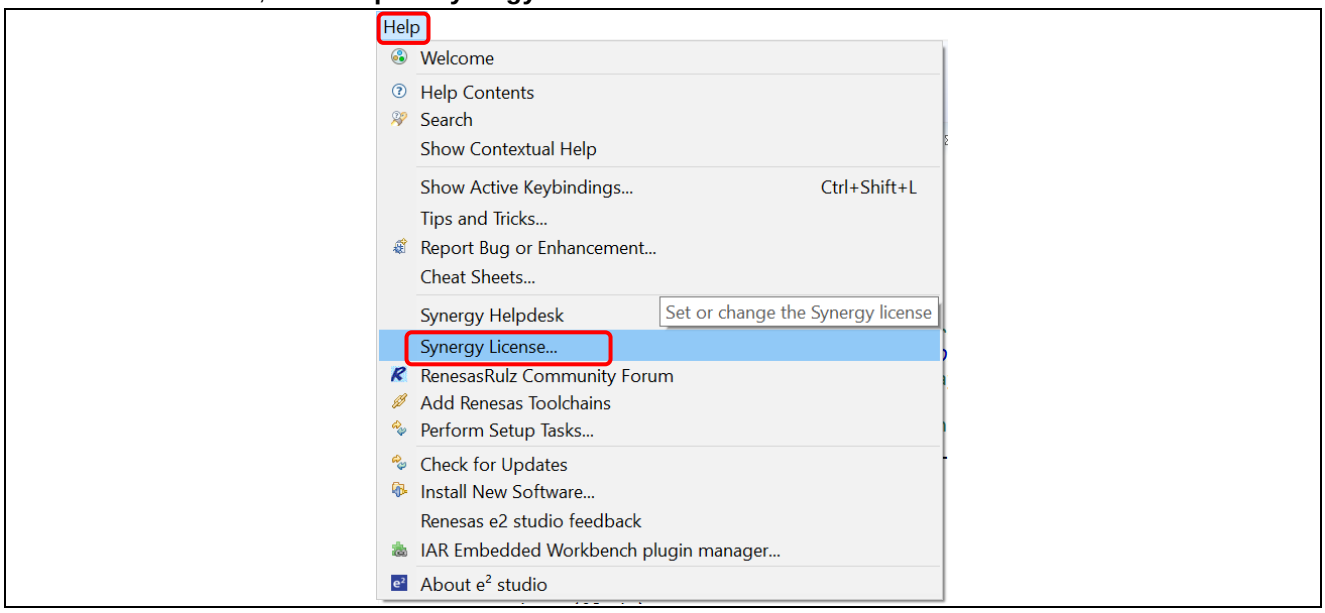


Figure 24. Register Synergy license

2. Click **…** then click **Browse** to browse to the license file. The default evaluation license file stored under `<e2_studio_base_dir>\internal\projectgen\arm\licenses\` is automatically pre-selected. Select the license file, click **Open** and click **OK** twice to register the license.

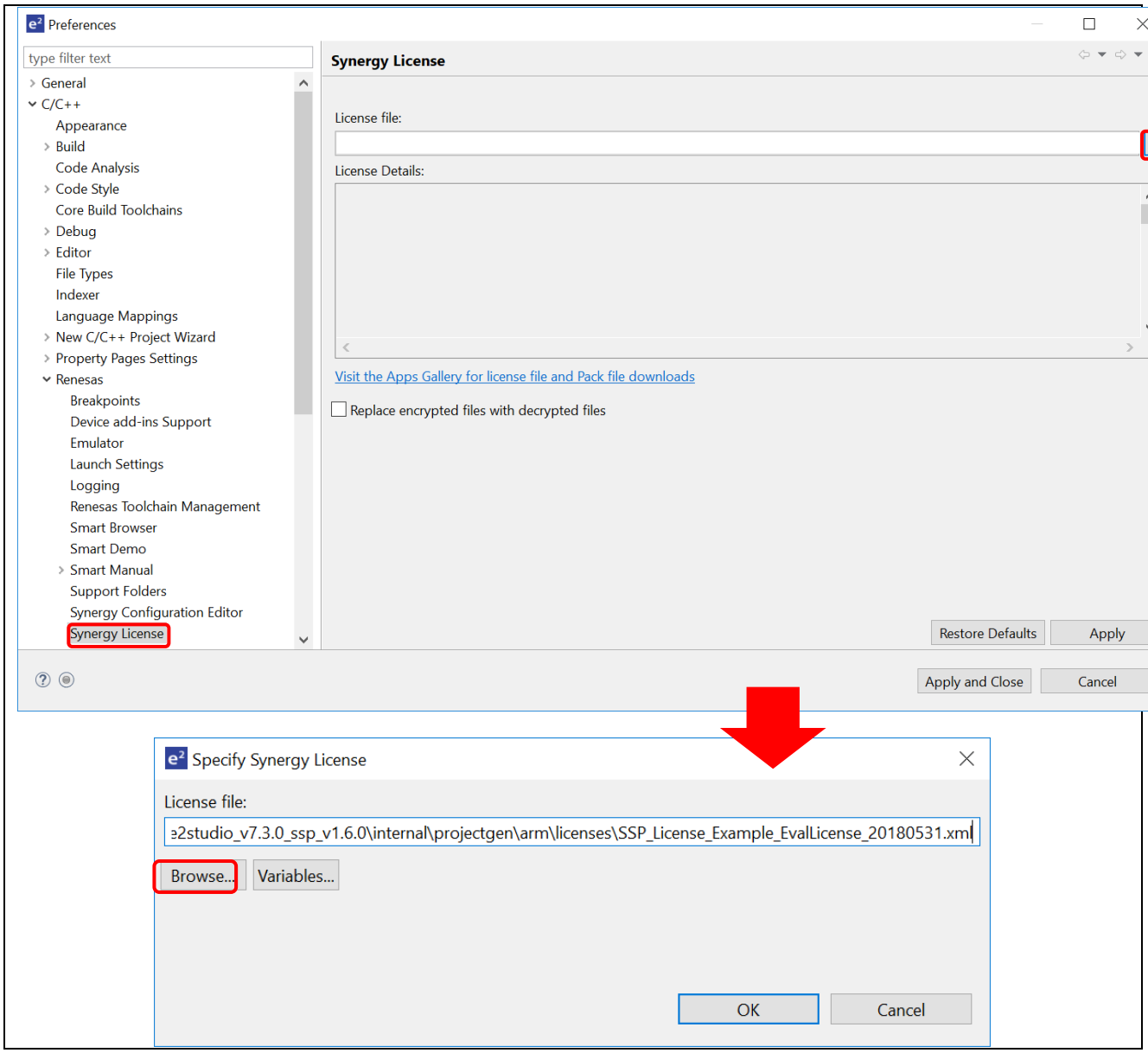


Figure 25. Specify the location of Synergy license file

3. Project Generation

This section 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 pre-requisite, the SSP and the toolchain must be installed on the host machine (see section 2).

3.1 Generating a New Synergy Project

A simple project generation wizard is available in e² studio to generate a new Synergy project with a project name and the associated device and board, including board-level drivers.

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**.

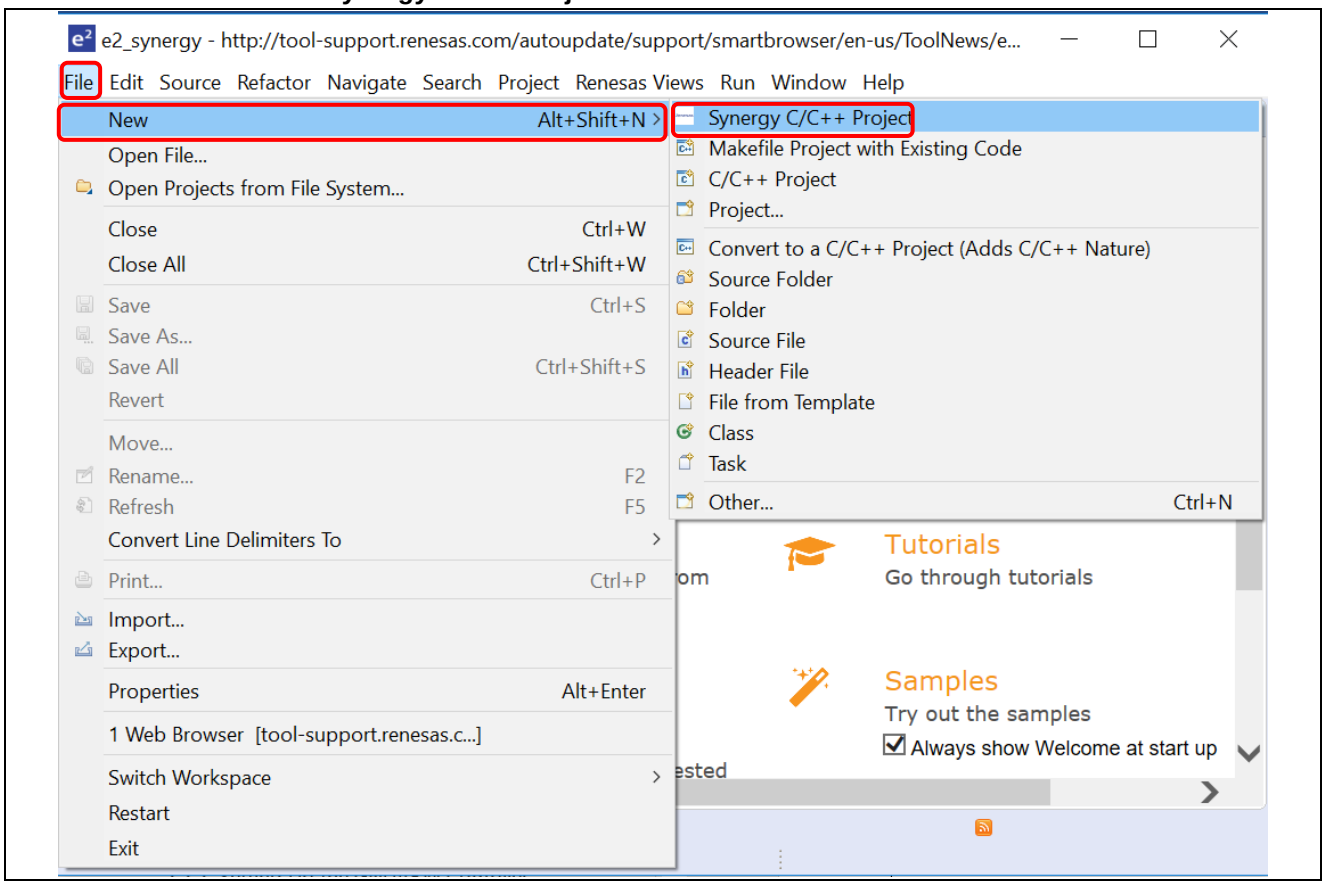


Figure 26. Project Generation – New Project Creation

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

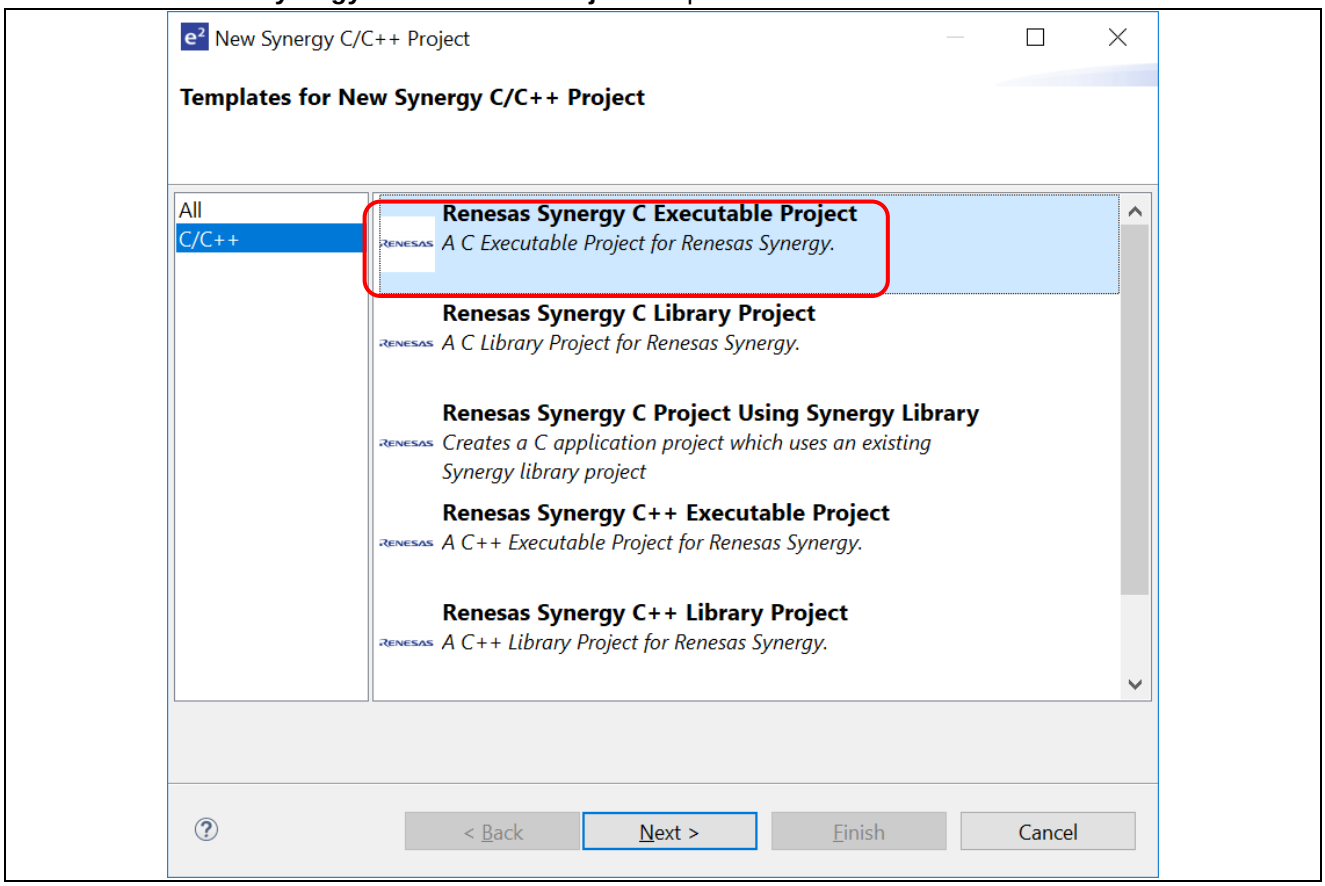


Figure 27. Project Generation – Select executable project template

3. In the project generation wizard, enter the following project information:
   - **Project name**: enter a name, such as **Synergy**.
   - Click the checkbox, **Use default location**. If users want to create a project in a different location, they can uncheck the box and enter a new location.
   - **Toolchain**: GCC Arm Embedded
   - **License**: In case a Synergy license file is not yet registered in the current workspace, click **Change license file** to open the Synergy License dialog box. Click the **…** button and browse to the Synergy license file of your choice. The default location for the Synergy license is:
     {e² studio installed folder}`\internal\projectgen\arm\Licenses`.
     This license file is available only after the SSP has been installed.
   - Click the **Next** button to continue.

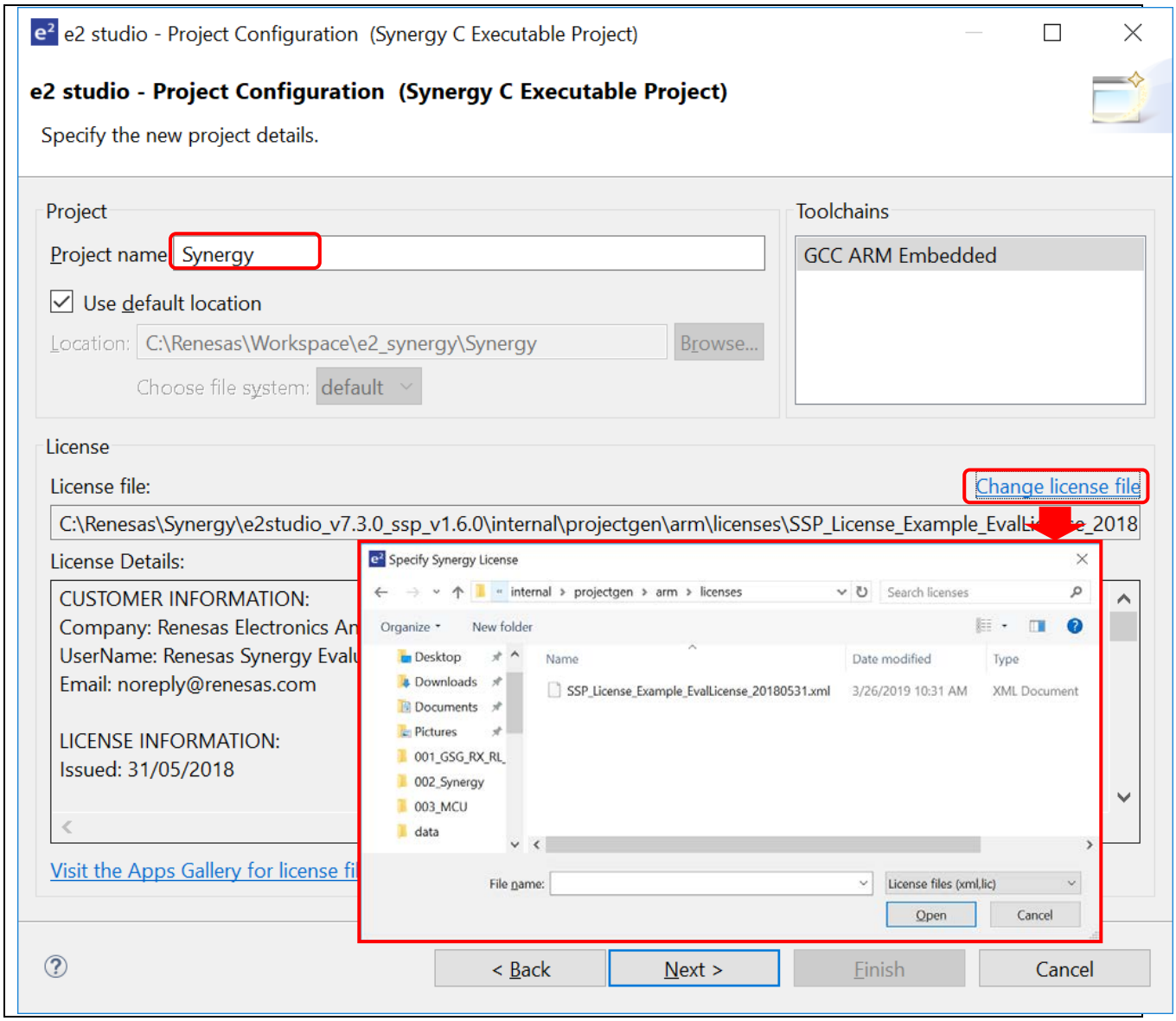


Figure 28. Project Generation – New Synergy Project Generation Wizard

4. In the device selection dialog, enter device and tool information:
   - Board (such as S7G2 DK)
   - **Toolchain version**: Latest GNU compiler approved for use with Renesas Synergy™ (for example, GCC Arm Embedded 7.2.1.20170904)
   - Keep all other fields as default.
   - Click **Next** to continue.

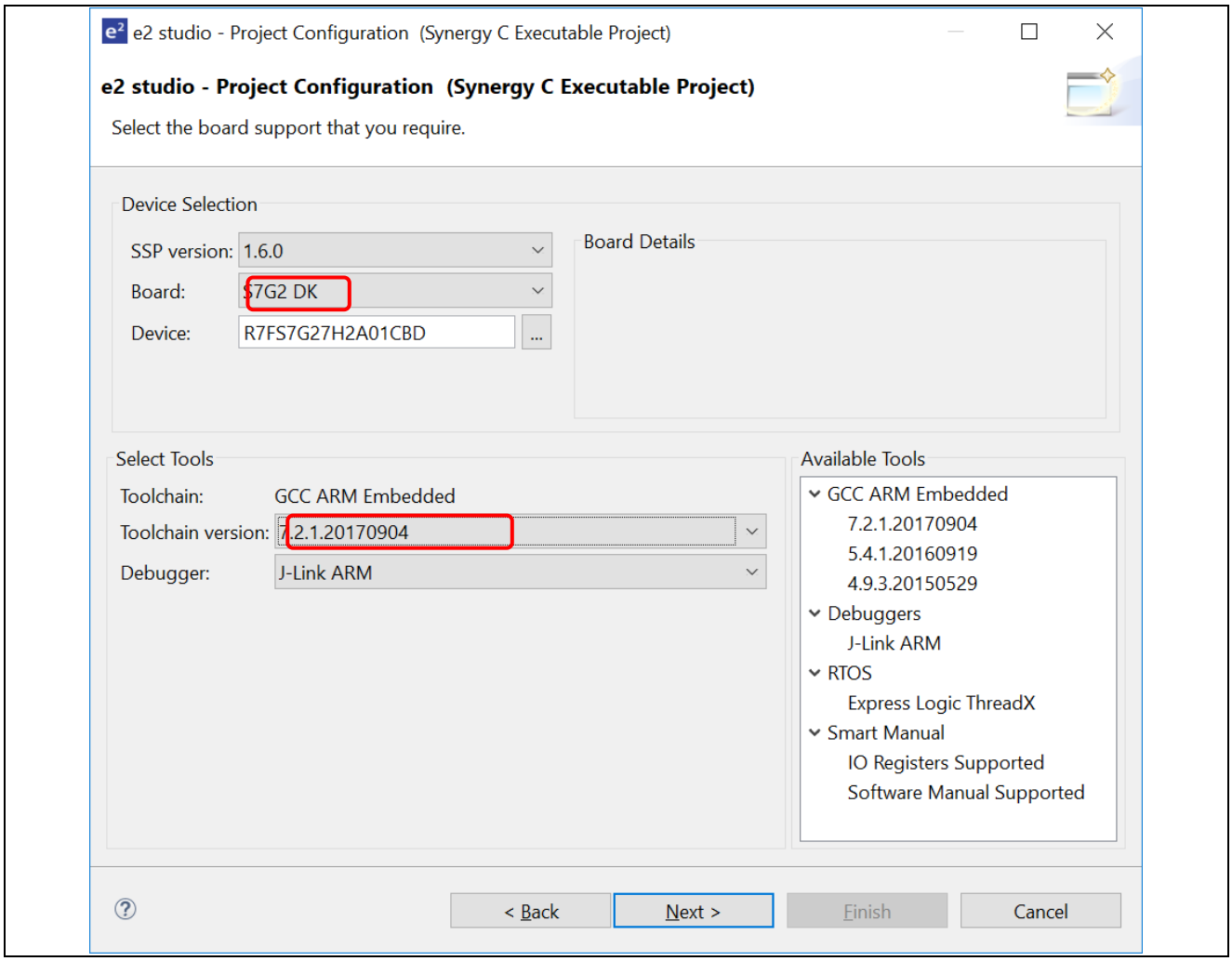


Figure 29. Project Generation – Device Selection

5. In the project template dialog, select a project template, such as Blinky.

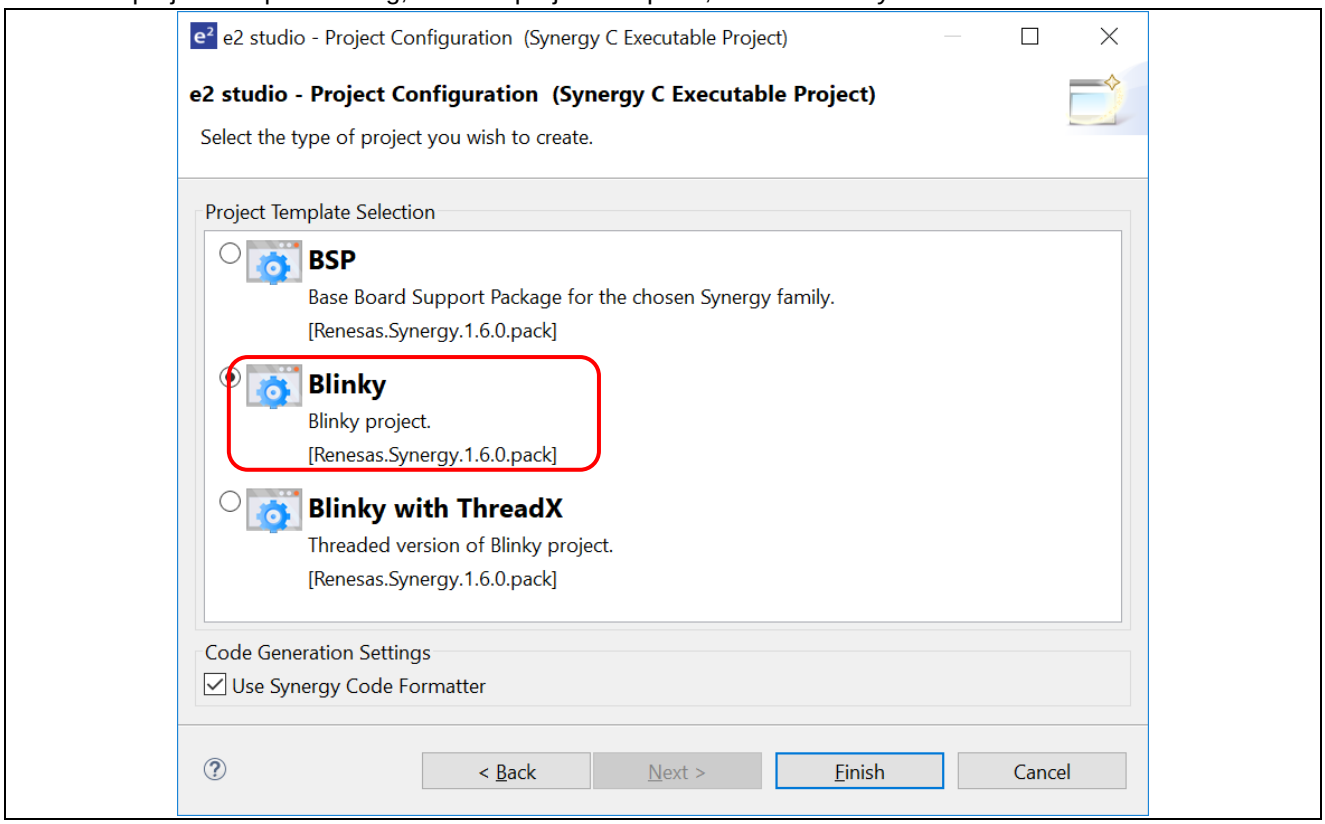


Figure 30. Project Generation – Project Template

6. Click the **Finish** button to create a new project.
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.

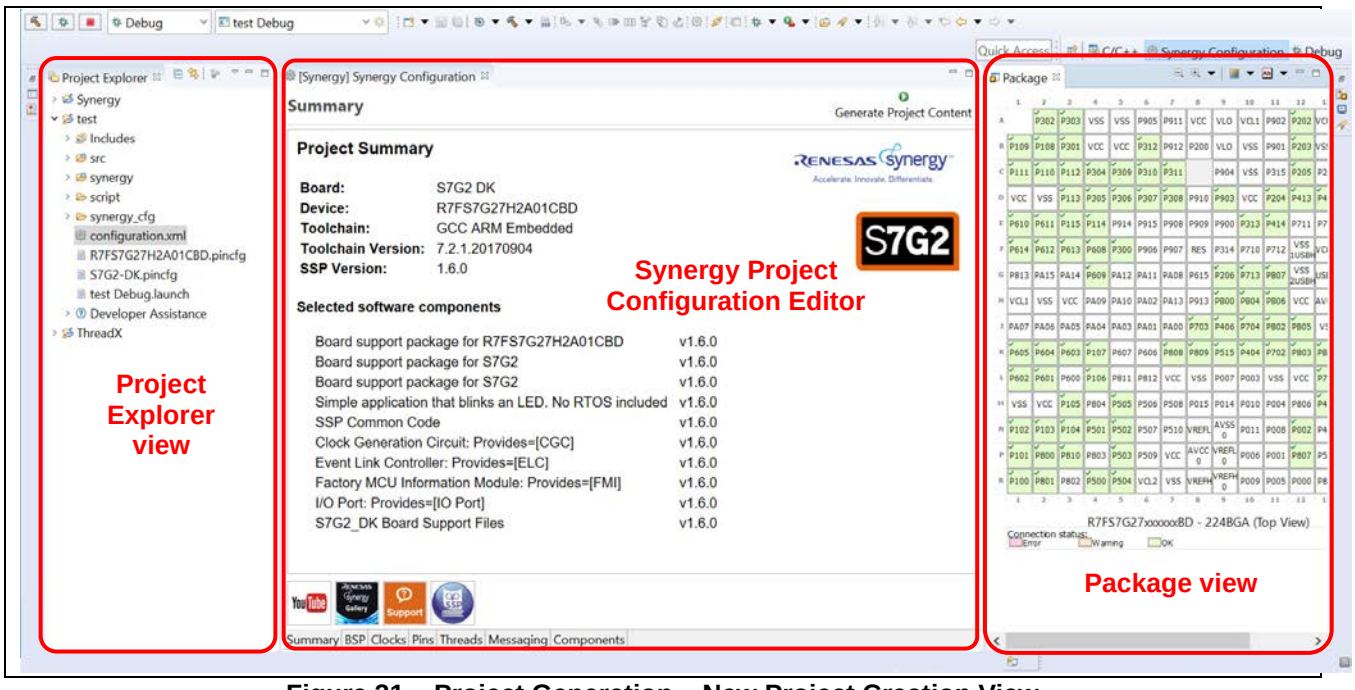


Figure 31. Project Generation – New Project Creation View

3.2 Importing an Existing Synergy Project

To import an existing Synergy Project, follow these steps:

1. Click **File** → **Import**.

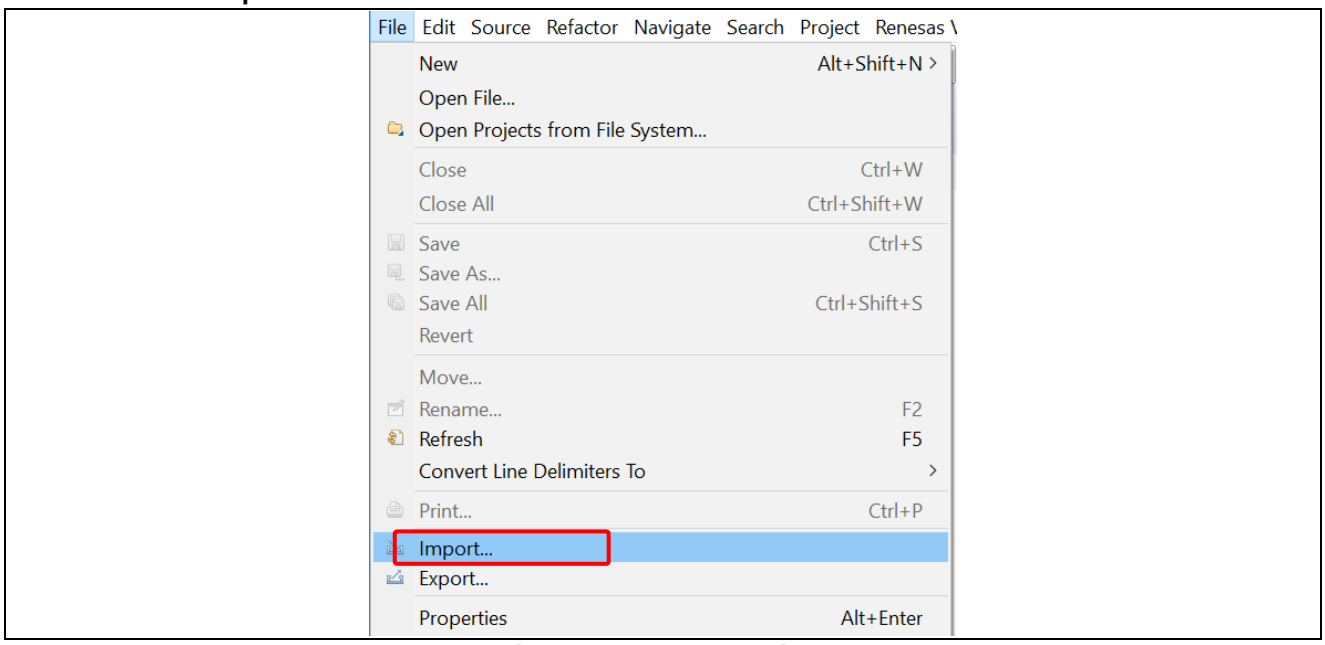


Figure 32. Import project

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.

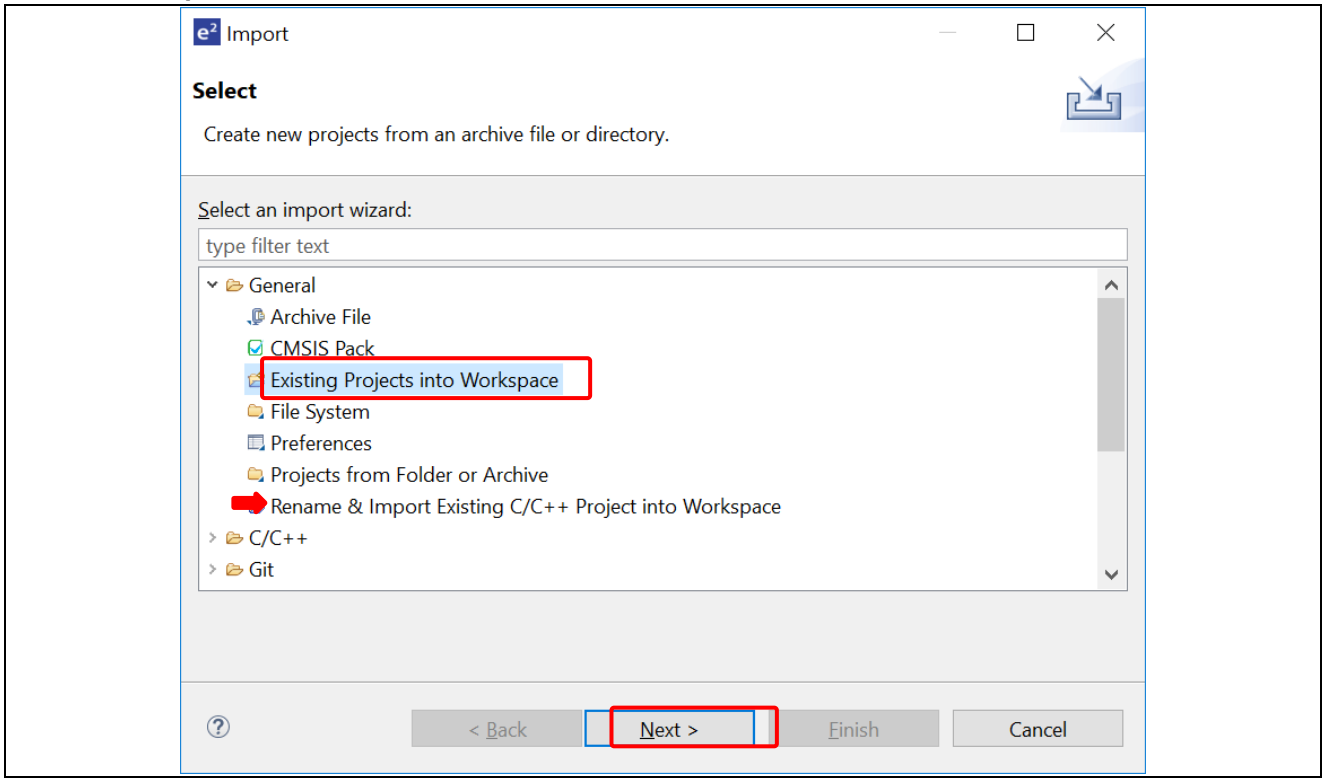


Figure 33. Select type of import

3. In the Import Projects dialog, click the **Select archive file** option, then **Browse**… to locate the compressed file (.zip) containing the project.
   If the existing project is stored in a folder, choose the **Select root directory** option.

4. Select the project to import and click **Finish**.

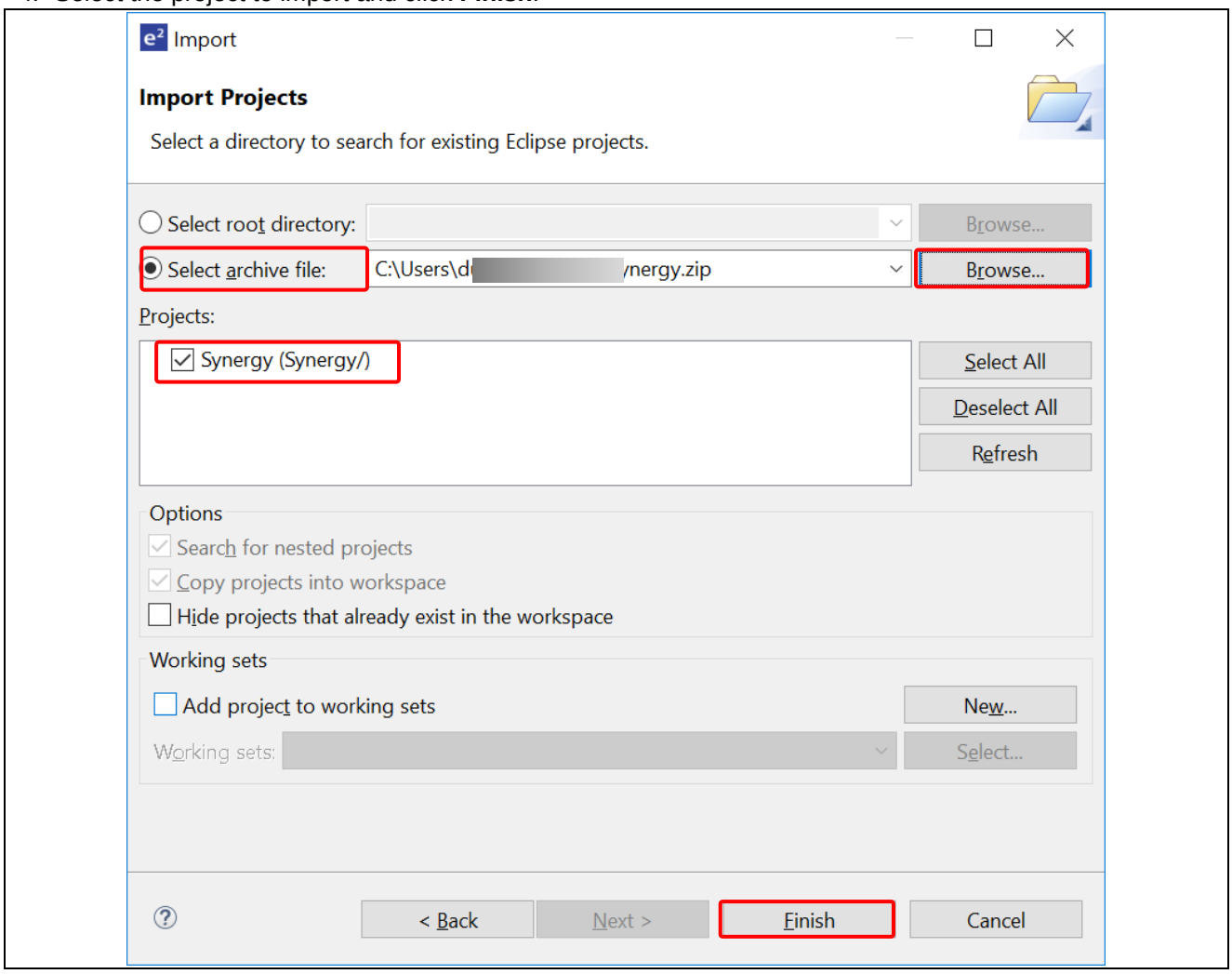


Figure 34. Select the project in the compressed file

5. The project imports into e² studio.

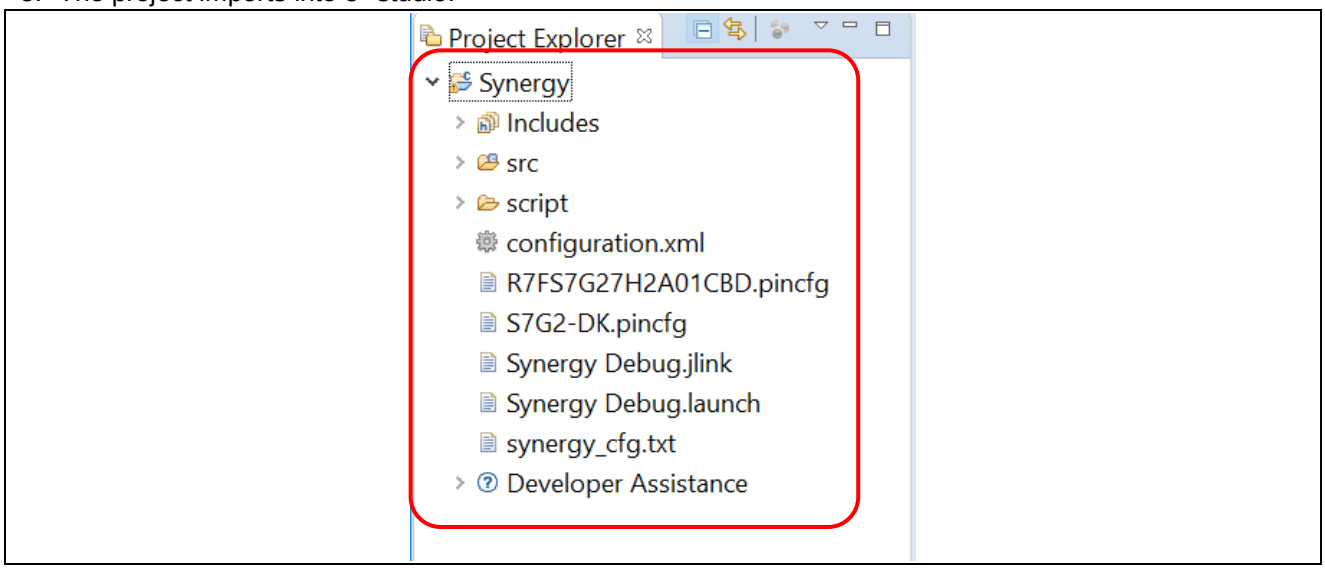


Figure 35. The imported project

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 **Renesas Synergy C Library Project** template. Click **Next** to continue.

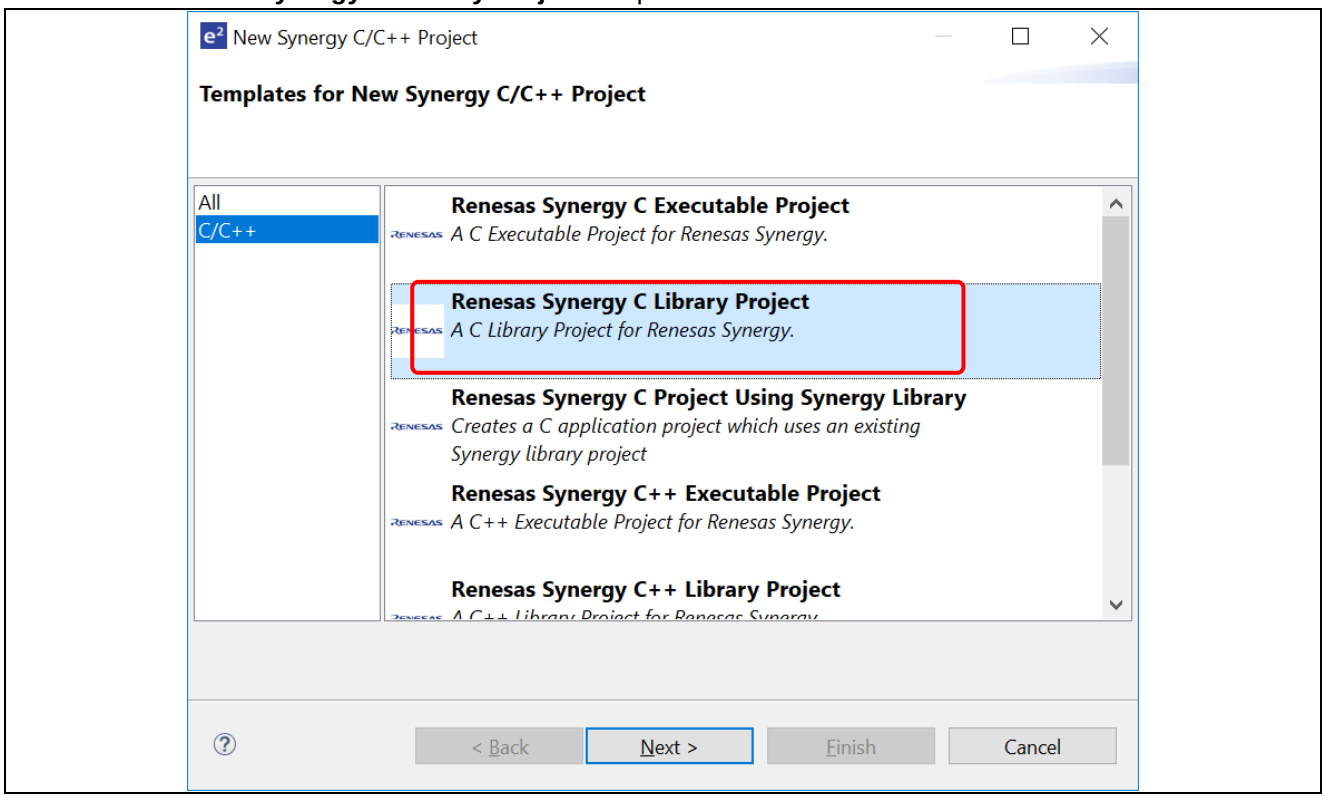


Figure 36. Project Generation – Select library project template

3. On the project details page, enter a name for the static lib project (such as, synergylib) and click **Next**.

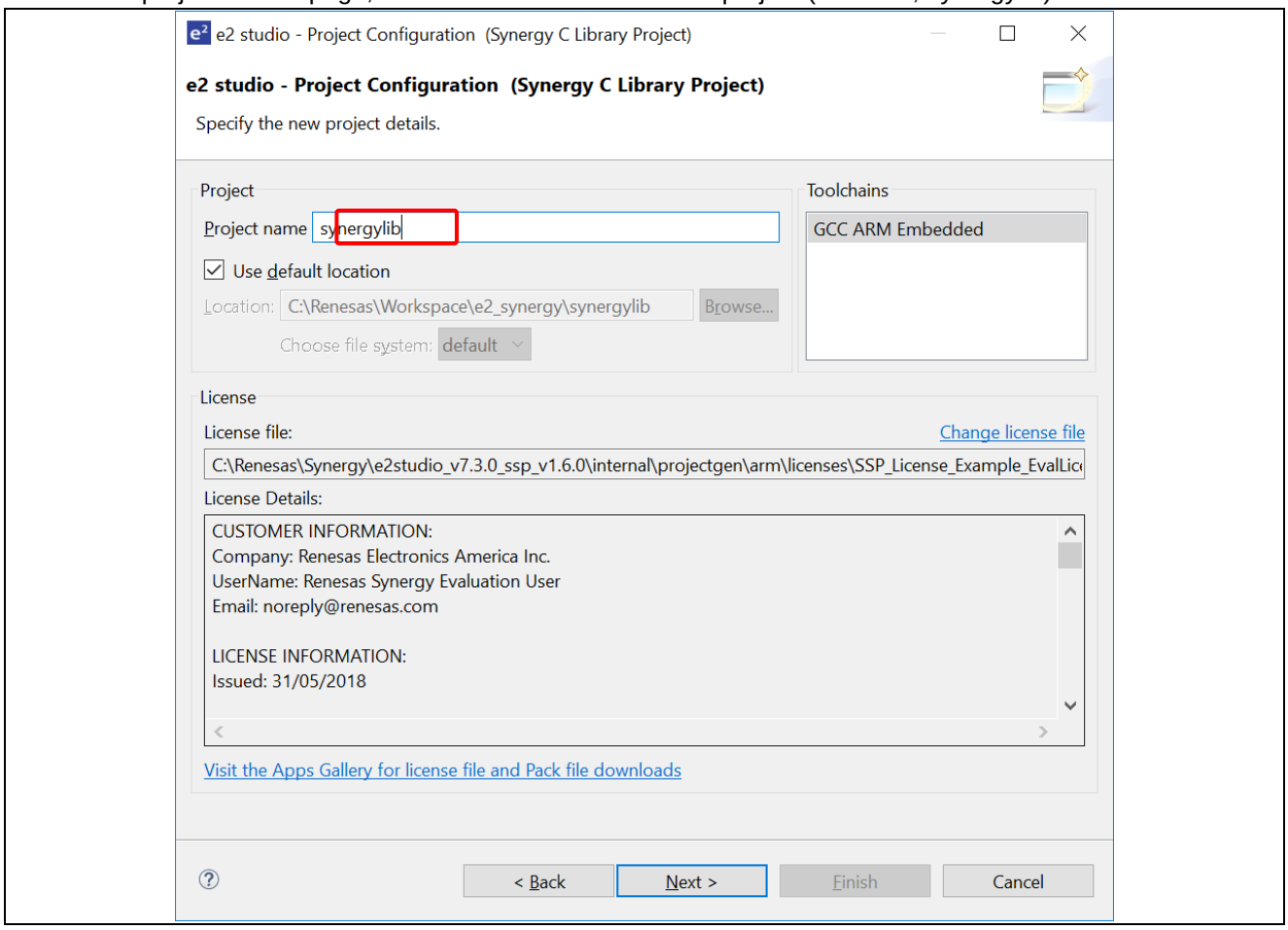


Figure 37. Library Project configuration

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

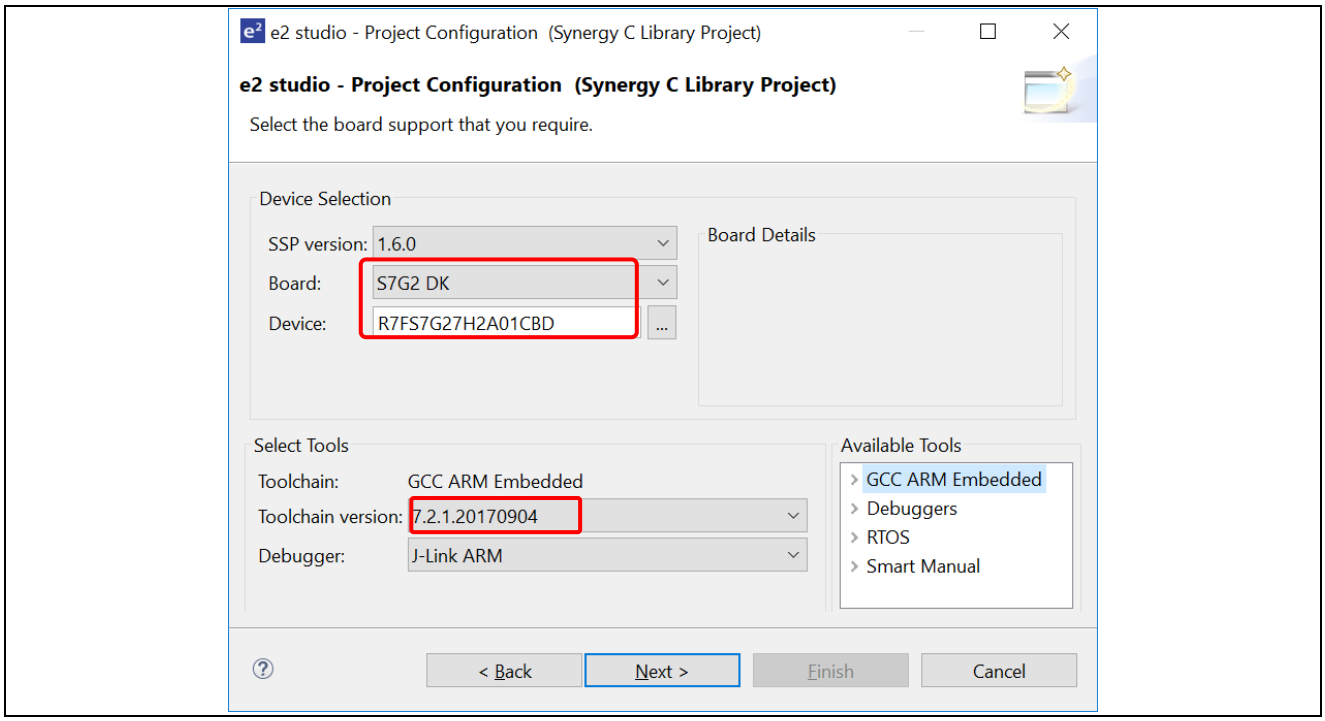


Figure 38. Select device and toolchain

5. In the project template dialog, select **Blinky**. Click **Finish** to create the project.

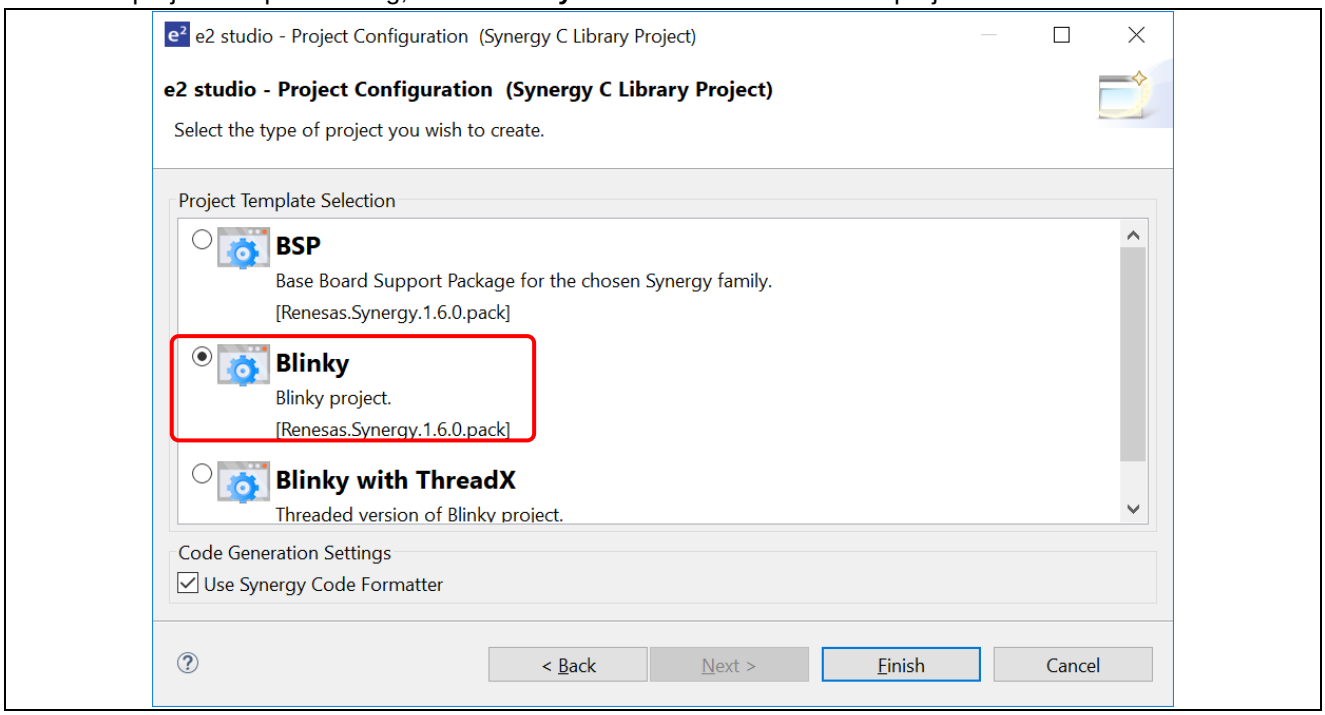


Figure 39. Select project template for library

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

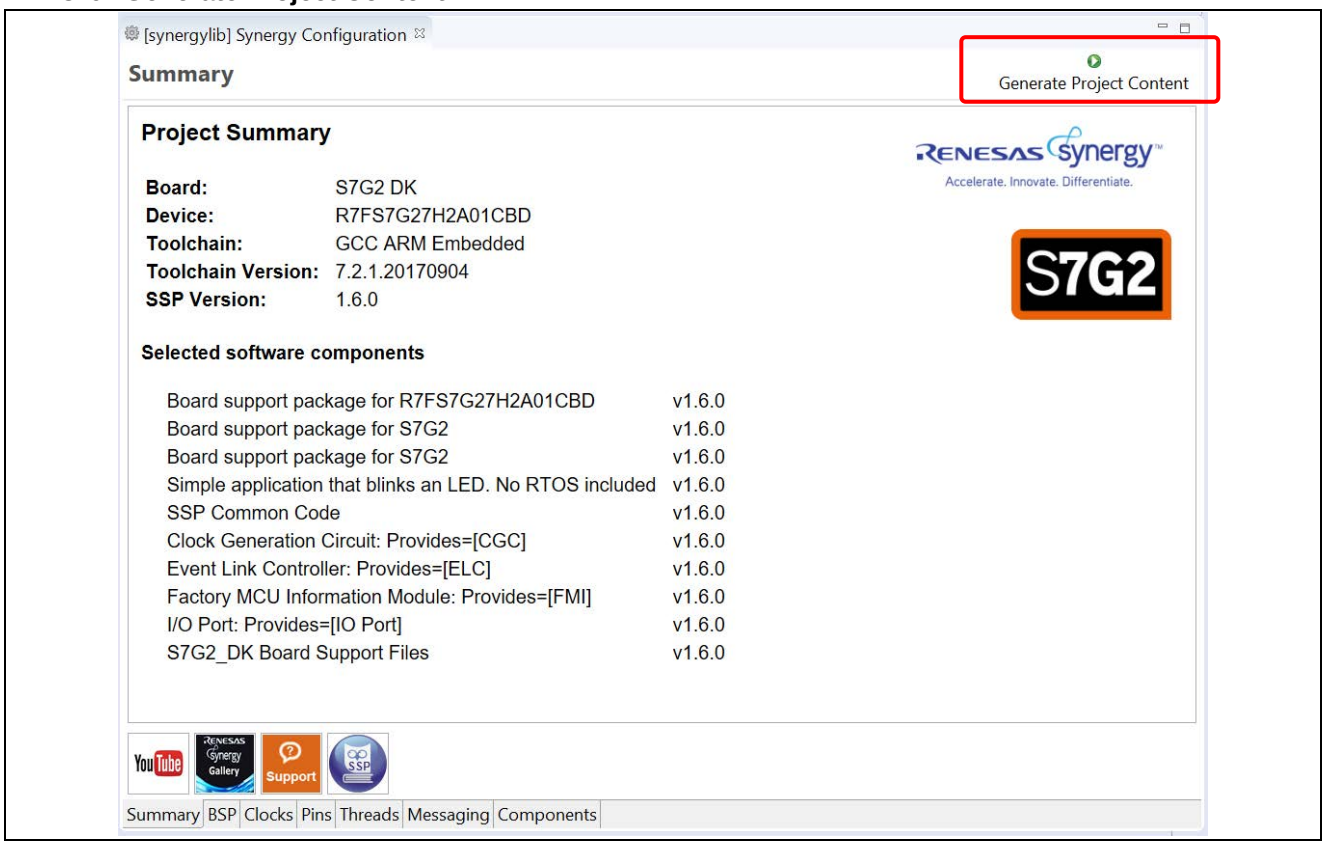


Figure 40. Generate library project content

8. From project explorer window, open **`hal_entry.c`** under `synergylib\src\`.

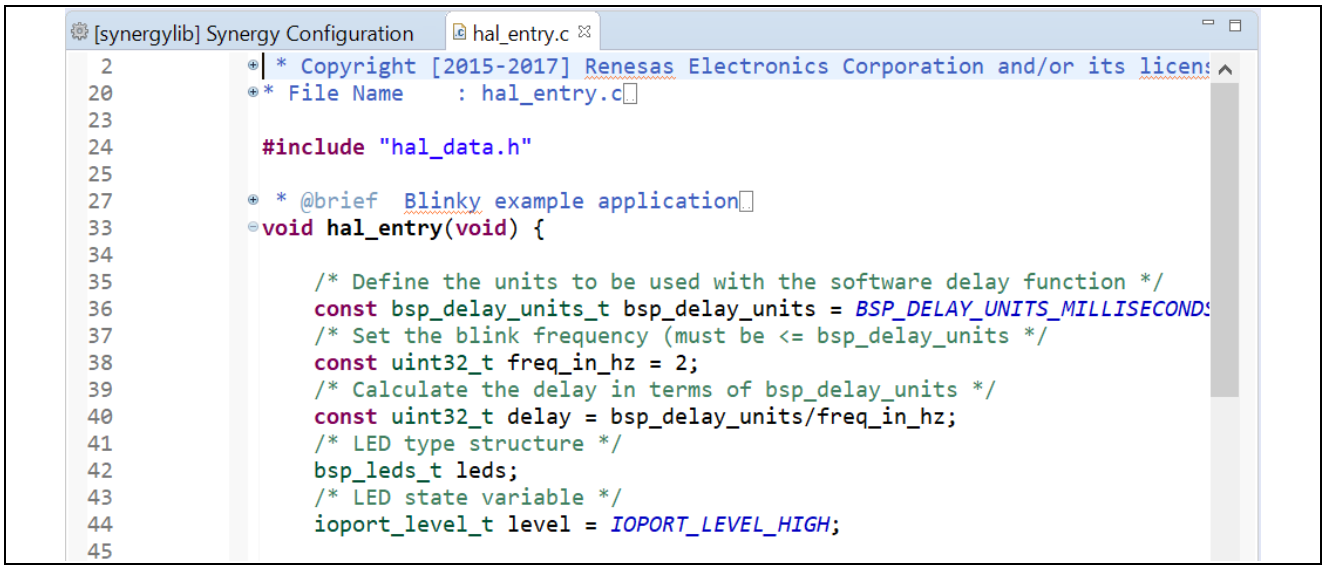


Figure 41. Old hal_entry.c

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

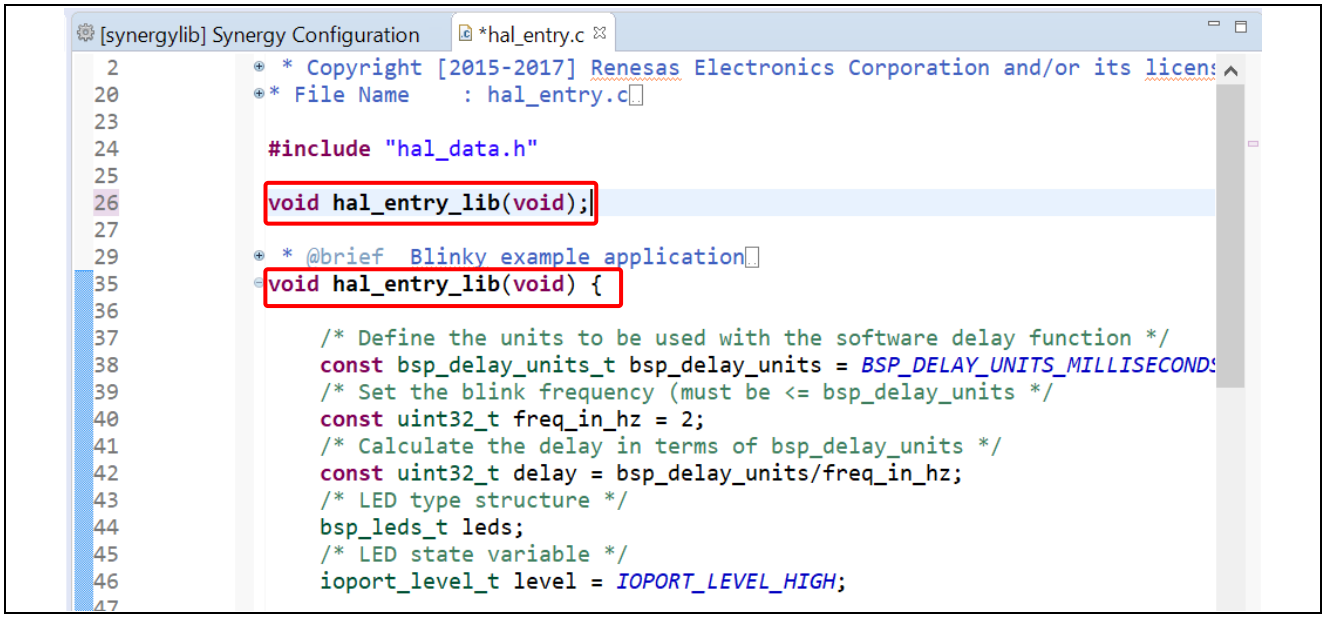


Figure 42. New hal_entry.c

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

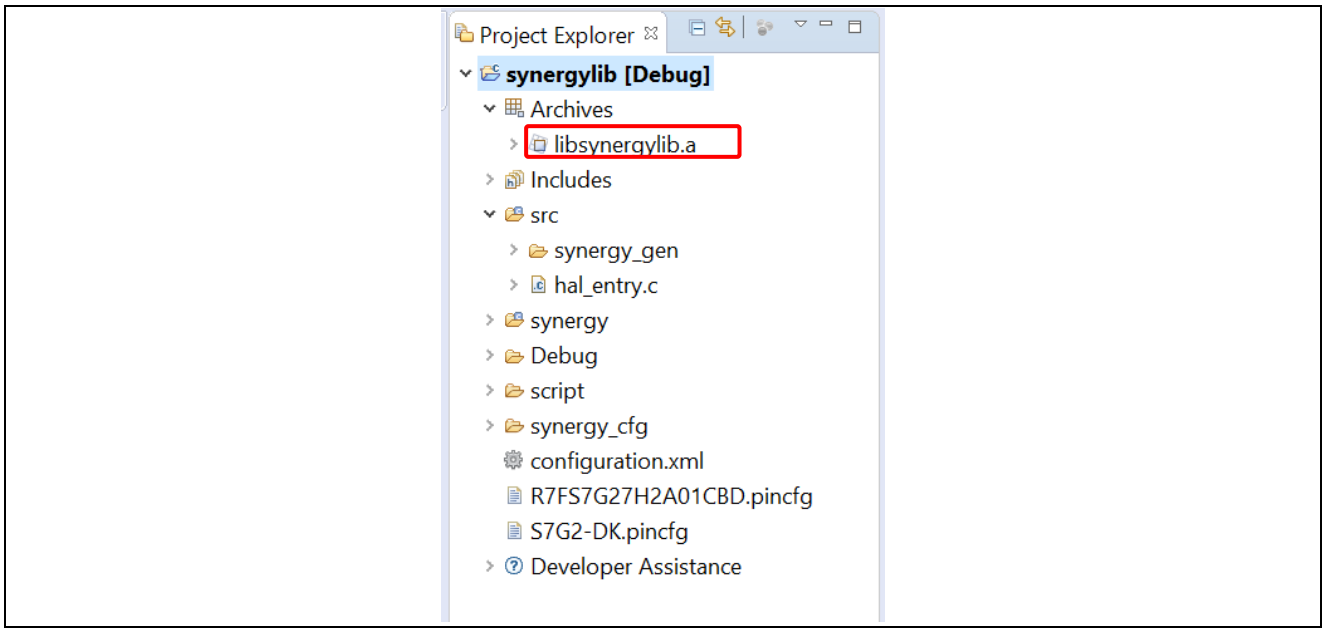


Figure 43. The built static library

3.3.2 Using the Static Library in the Executable Project

This section shows how to use the static library created in section 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.

Apply the following steps:

1. Create an executable project named **synergyapp** and select the **BSP** template. Click **Finish** to create the project. For details on creating an executable project, see section 3.1.

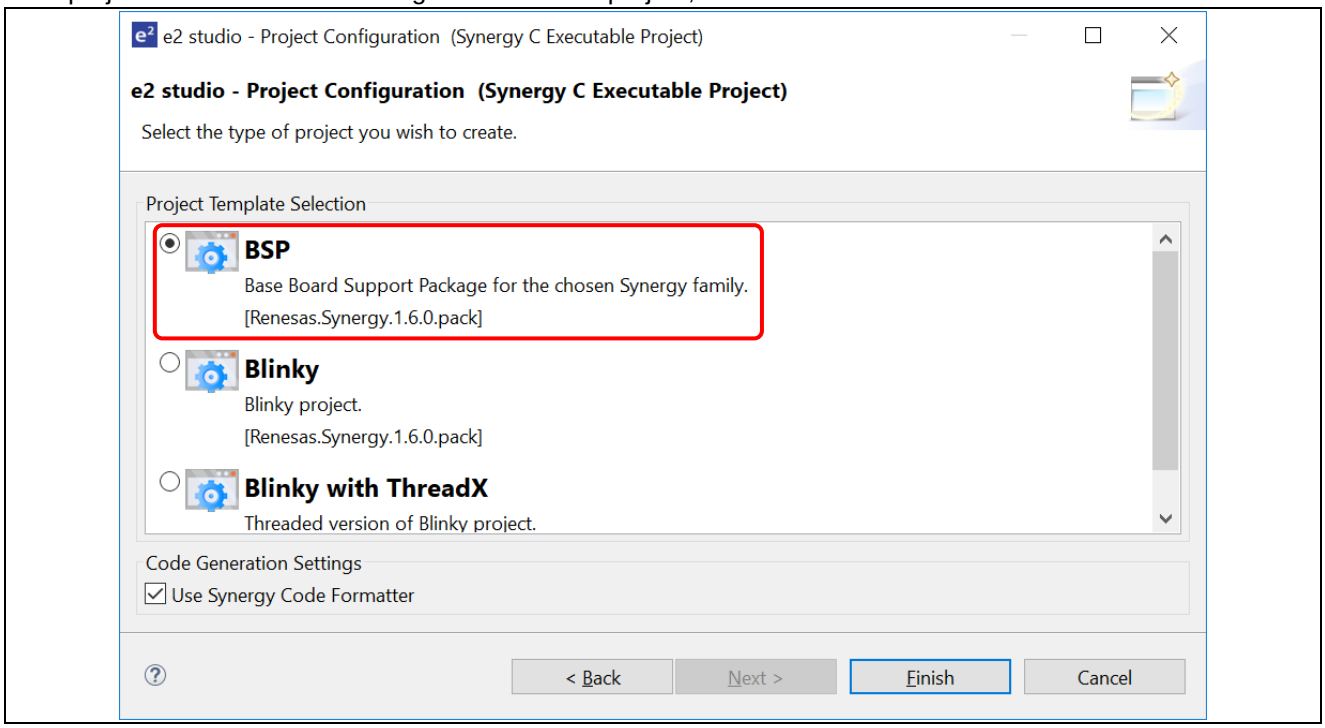


Figure 44. Select project template for executable project referring to the library

2. After the executable project is created, click **Generate Project Content**.
3. From project explorer window, open `hal_entry.c` under `synergylib\src\`.

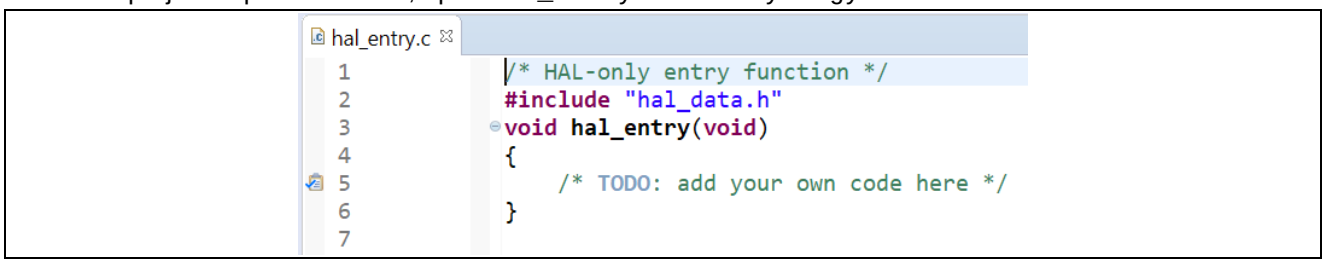


Figure 45. Old hal_entry.c

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

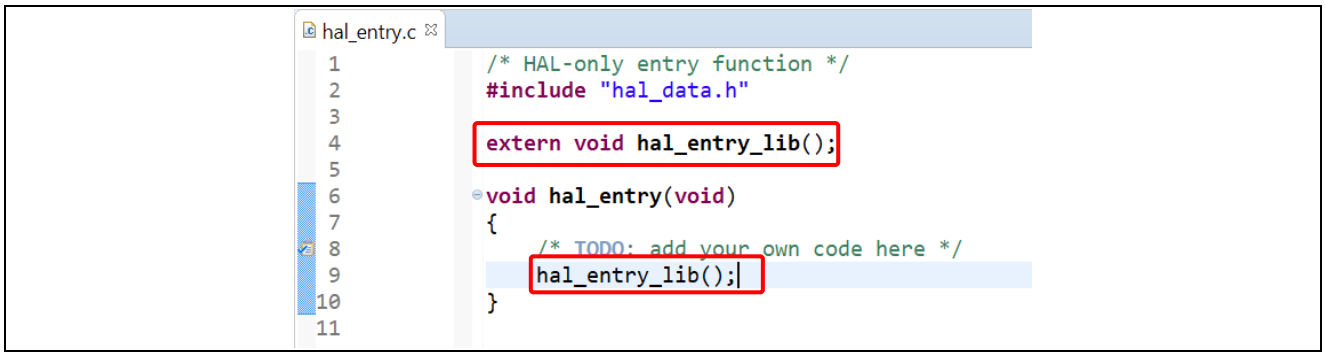


Figure 46. New hal_entry.c

4. Right-click on **synergyapp** project, then select **Properties**.
5. In the Project Properties dialog, select **C/C++ Build** → **Settings**, then select **Tool Settings** tab. Select **GNU ARM Cross C Linker** → **Libraries**, then add **synergylib** as a library

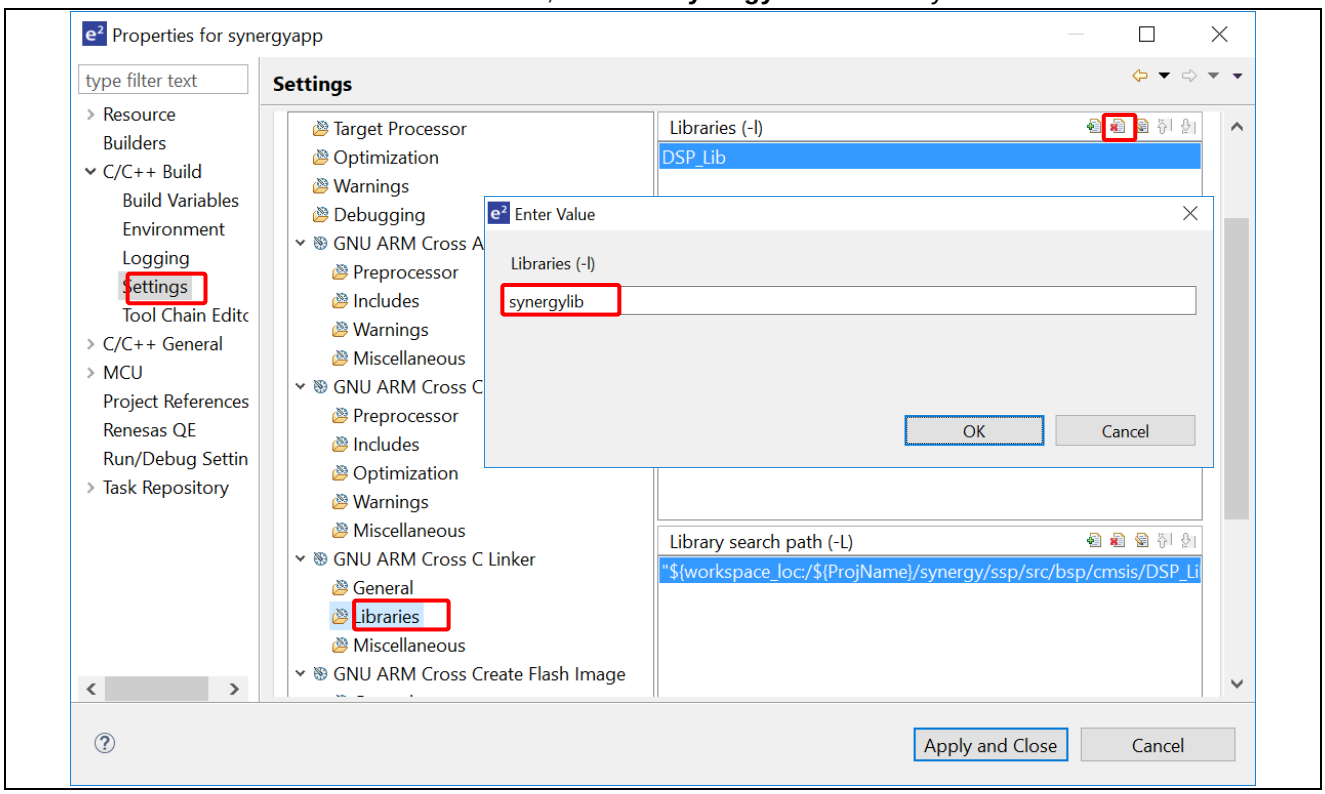


Figure 47. Add library project as a library for executable project

6. Add new library search path `${workspace_loc:/synergylib/Debug}` for **GNU ARM Cross C Linker → Libraries**.

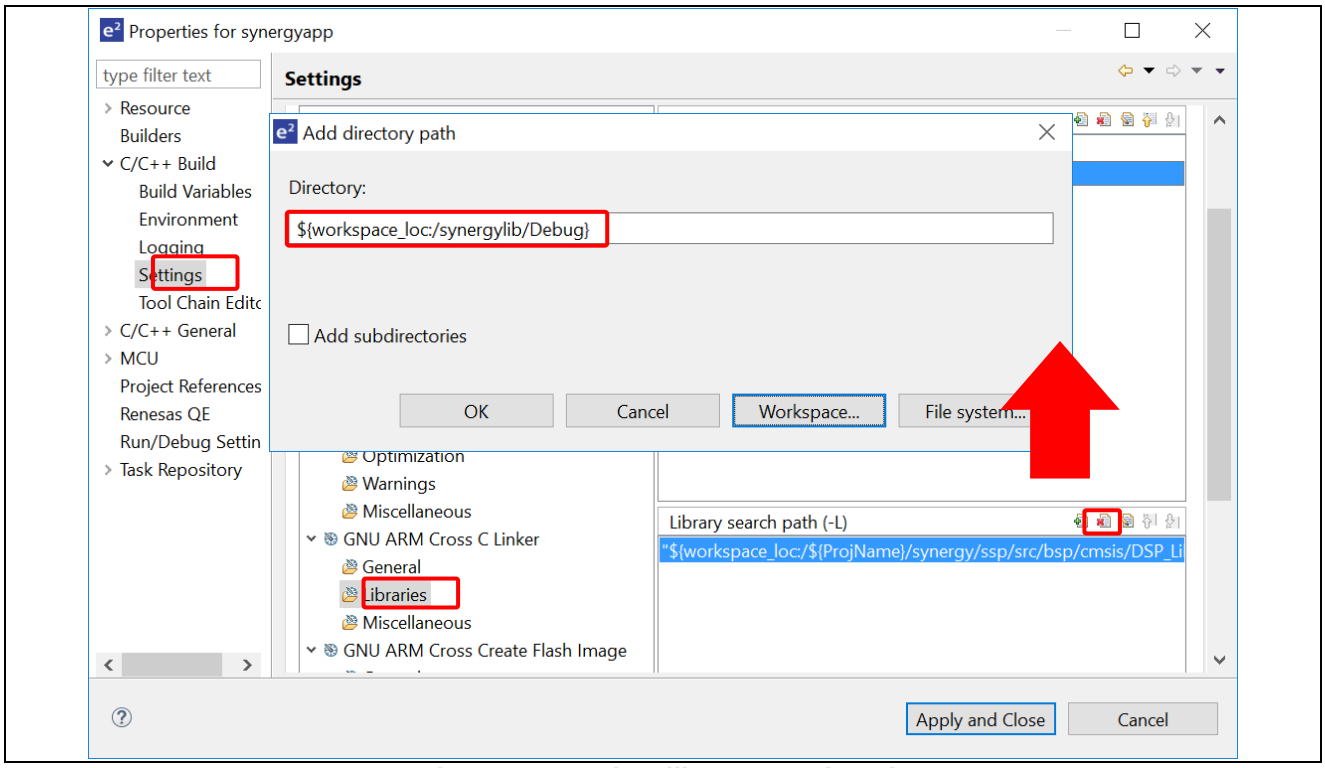


Figure 48. Update library search path

7. Under **GNU ARM Cross C Linker → Libraries** check the new path updated.

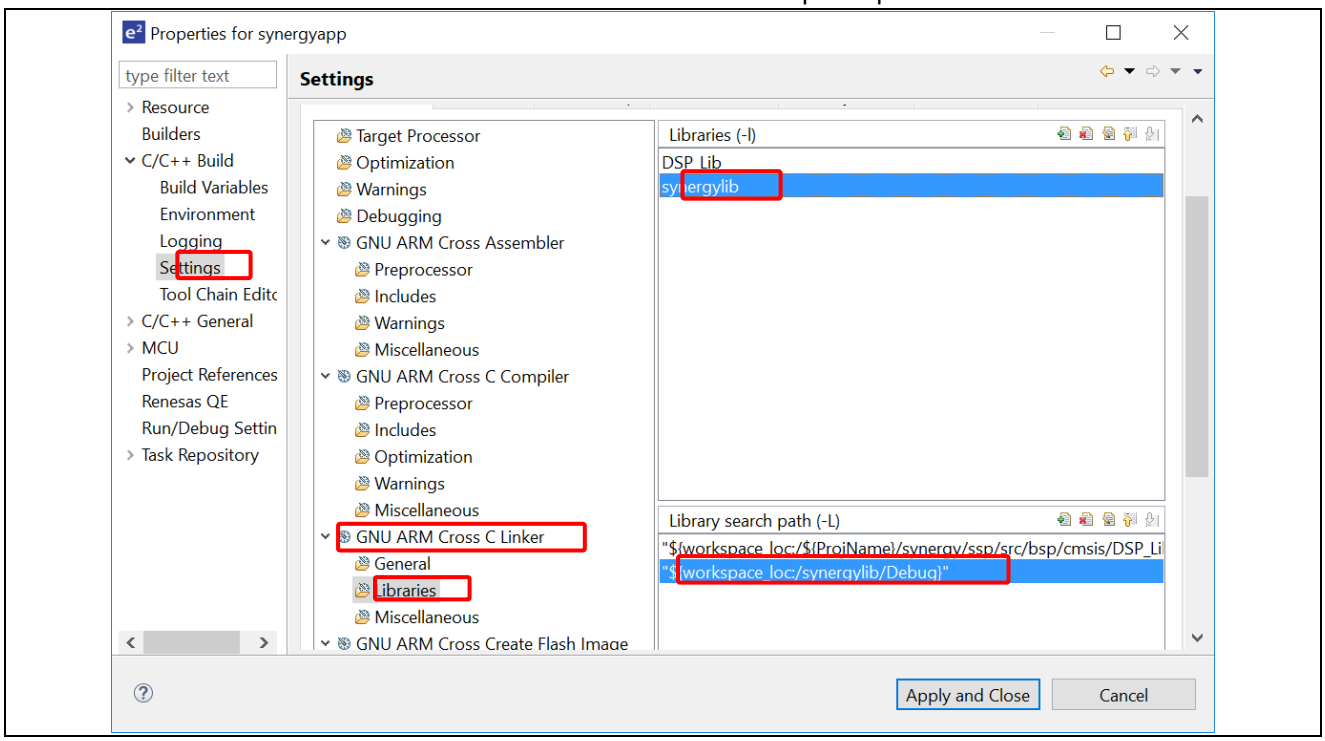


Figure 49. GNU ARM Cross C Linker libraries are already updated

Note: The `Libraries(-l)` option shown in the preceding figure is used to add the library file with file name format **libxxx.a** only. If the format of library name is not **libxxx.a**, add it using **GNU ARM Cross C Linker → Miscellaneous** settings.

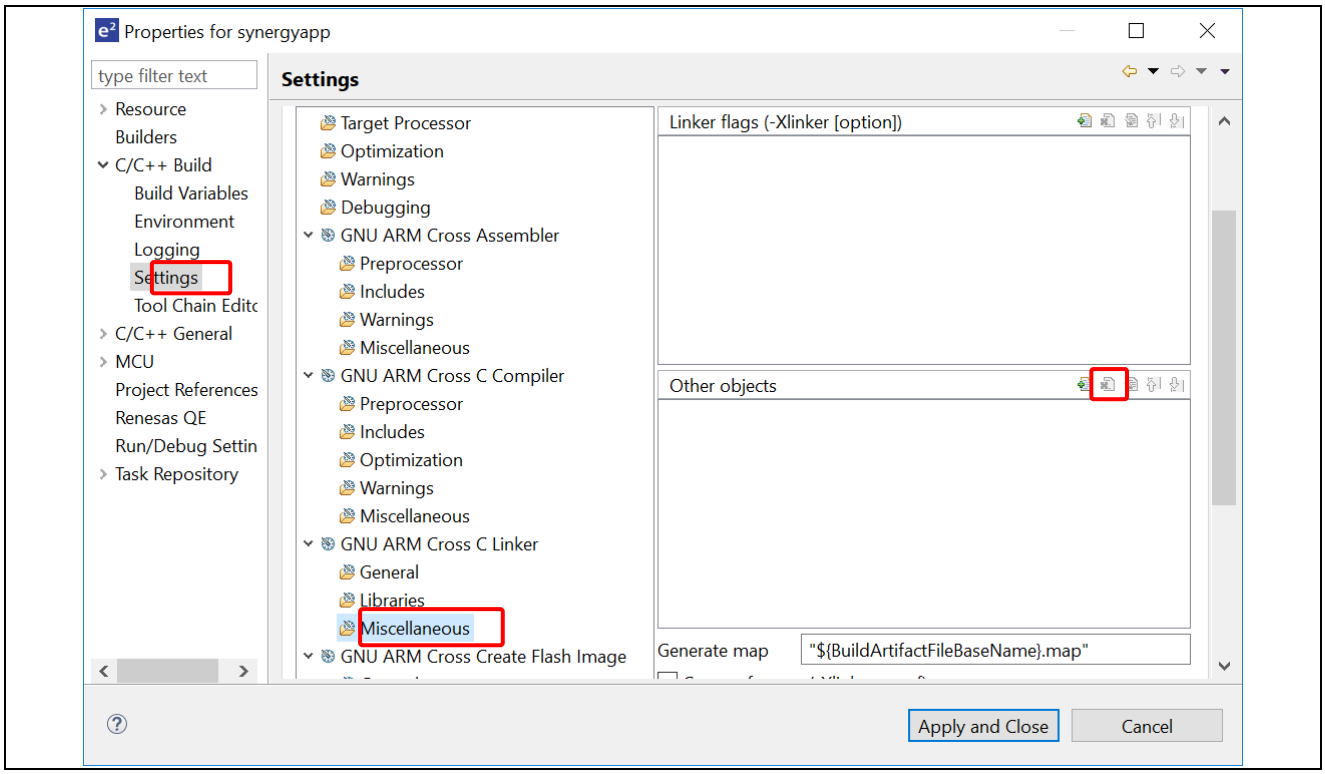


Figure 50. Add library which name does not use the format libxxx.a

8. In the Properties dialog, select **Project References** in the left pane, then check the **synergylib** box to mark the executable project as depending on the static library project. Click **Apply and Close**.

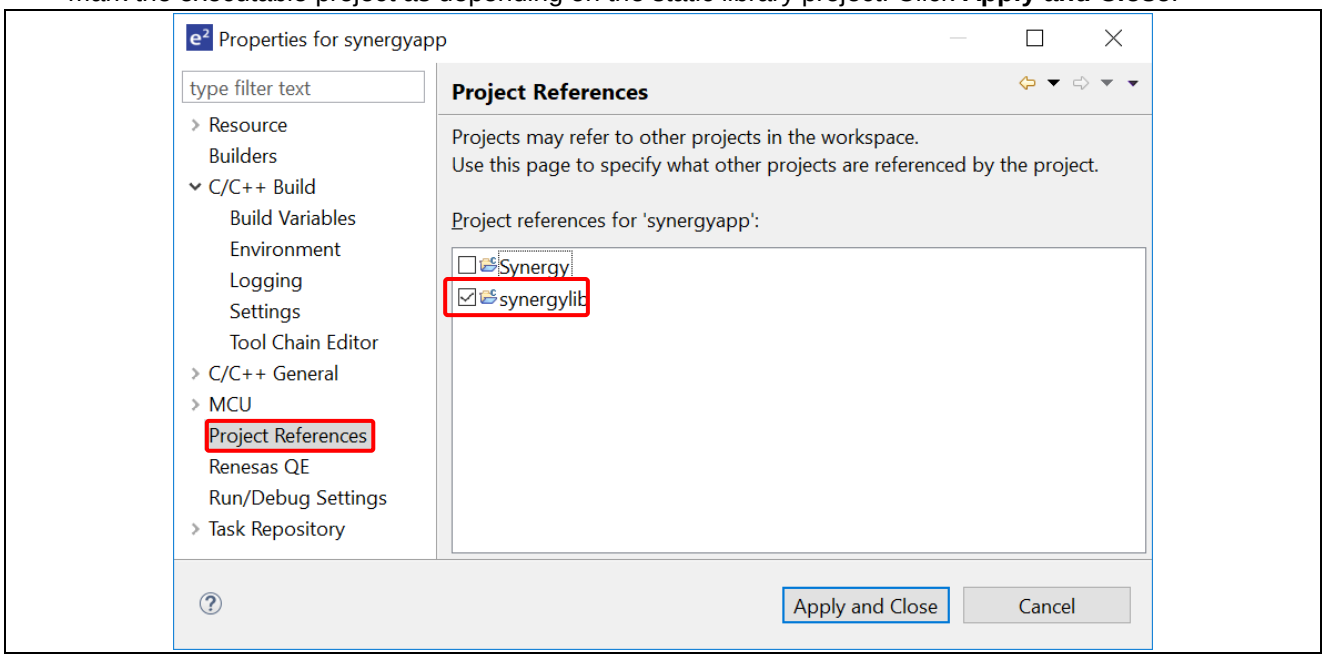


Figure 51. Mark the executable project as depending on the static library project

9. Build the application project.
10. Set a breakpoint where the library function `hal_entry_lib()` is called. Run **synergyapp** project.
11. When the program stops at the breakpoint, resume it. Confirm that the library function that blinks the LEDs (`hal_entry_lib()`) is executed.

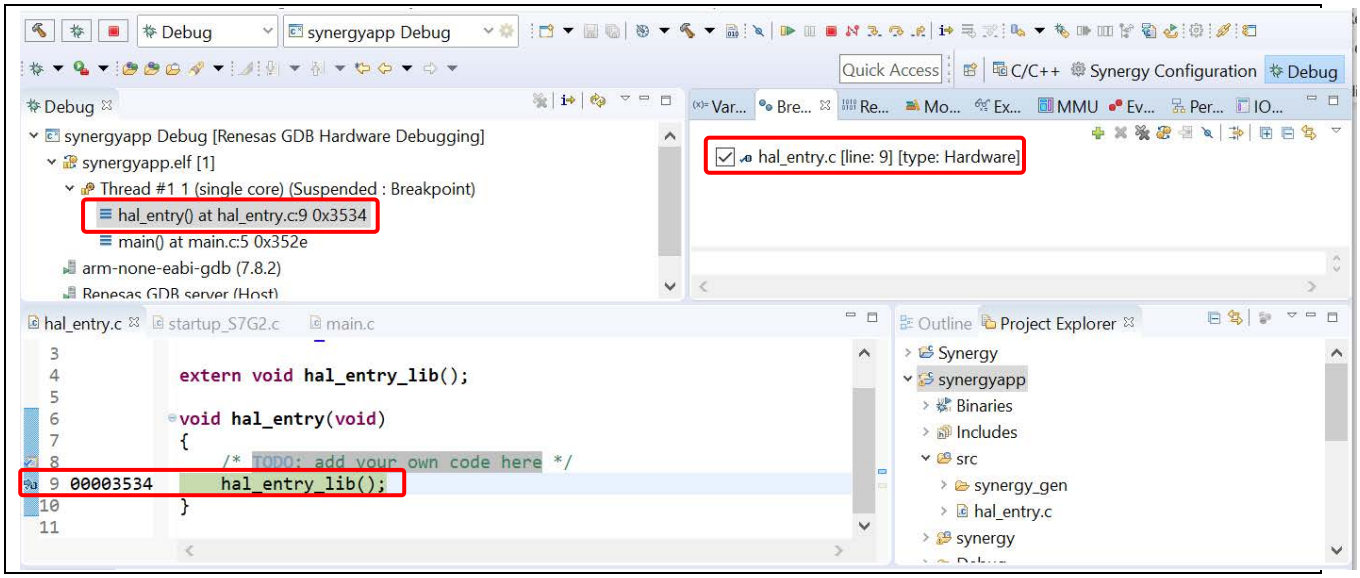


Figure 52. Application project executing library function

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, allowing 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, be sure to:

- Select the Synergy Configuration perspective in the upper right-hand corner of the e² studio window, or click **Window → Perspective → Open Perspective → Other… → Synergy Configuration**.
- Open the 'configuration.xml' file.

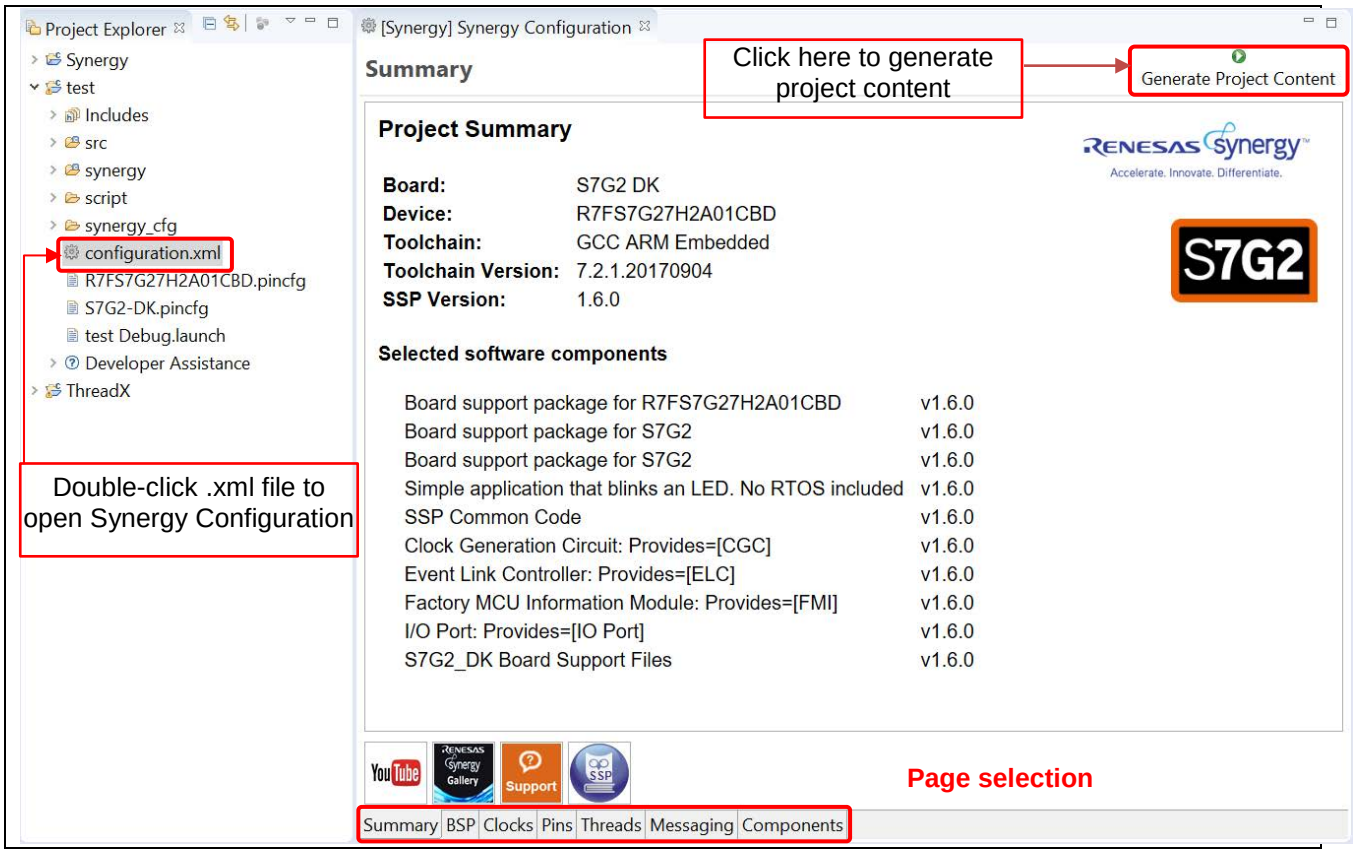


Figure 53. Synergy Project Configuration – Synergy Project Configuration View

There are seven pages (or tabs) in Synergy Project Configuration editor.

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

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

In addition, Clocks, Pins, Threads, Messaging and Components pages are discussed in the following sections, including configuration steps and options.

3.4.1 Summary Page

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

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

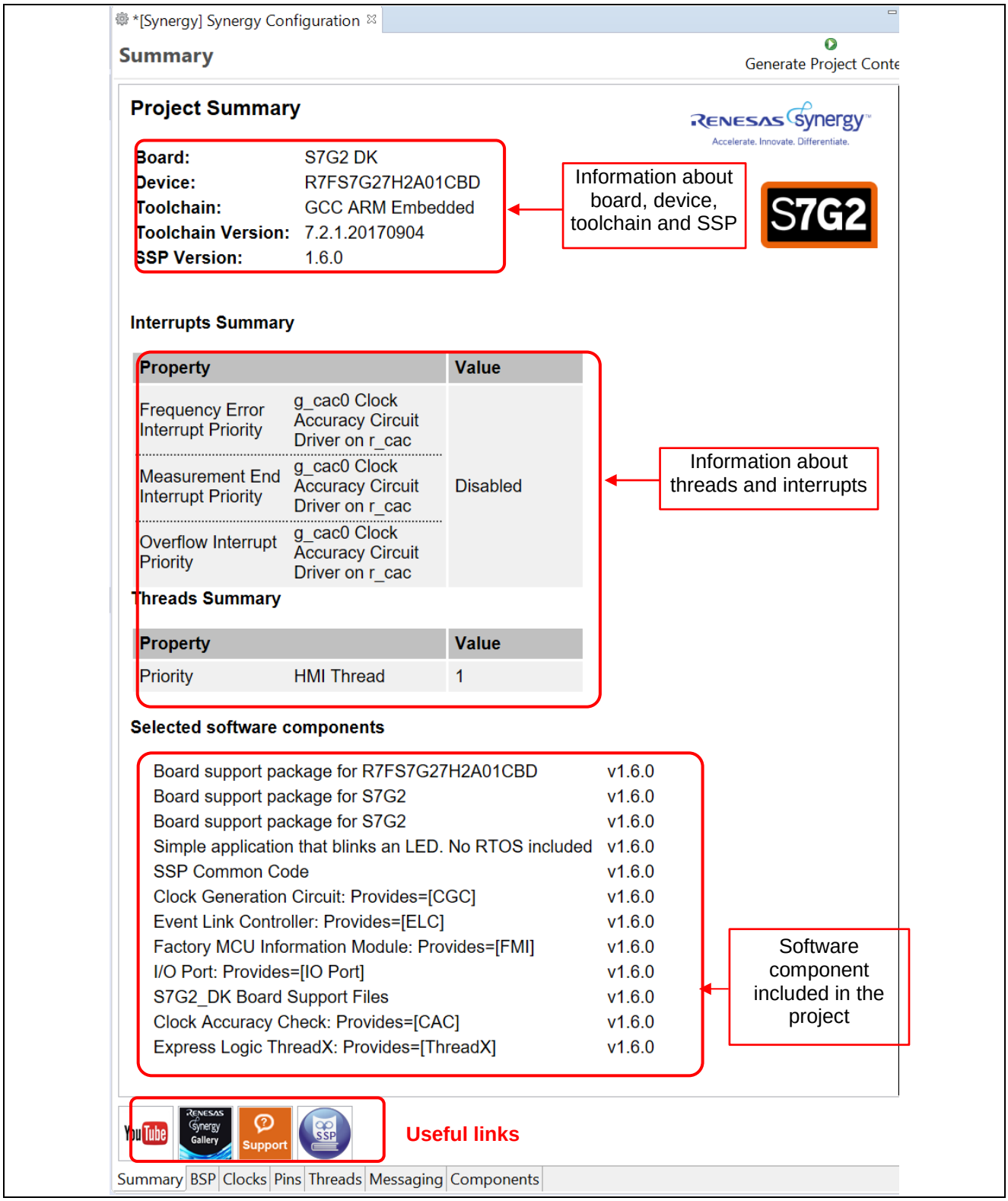


Figure 54. Summary Page

3.4.2 BSP Page

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

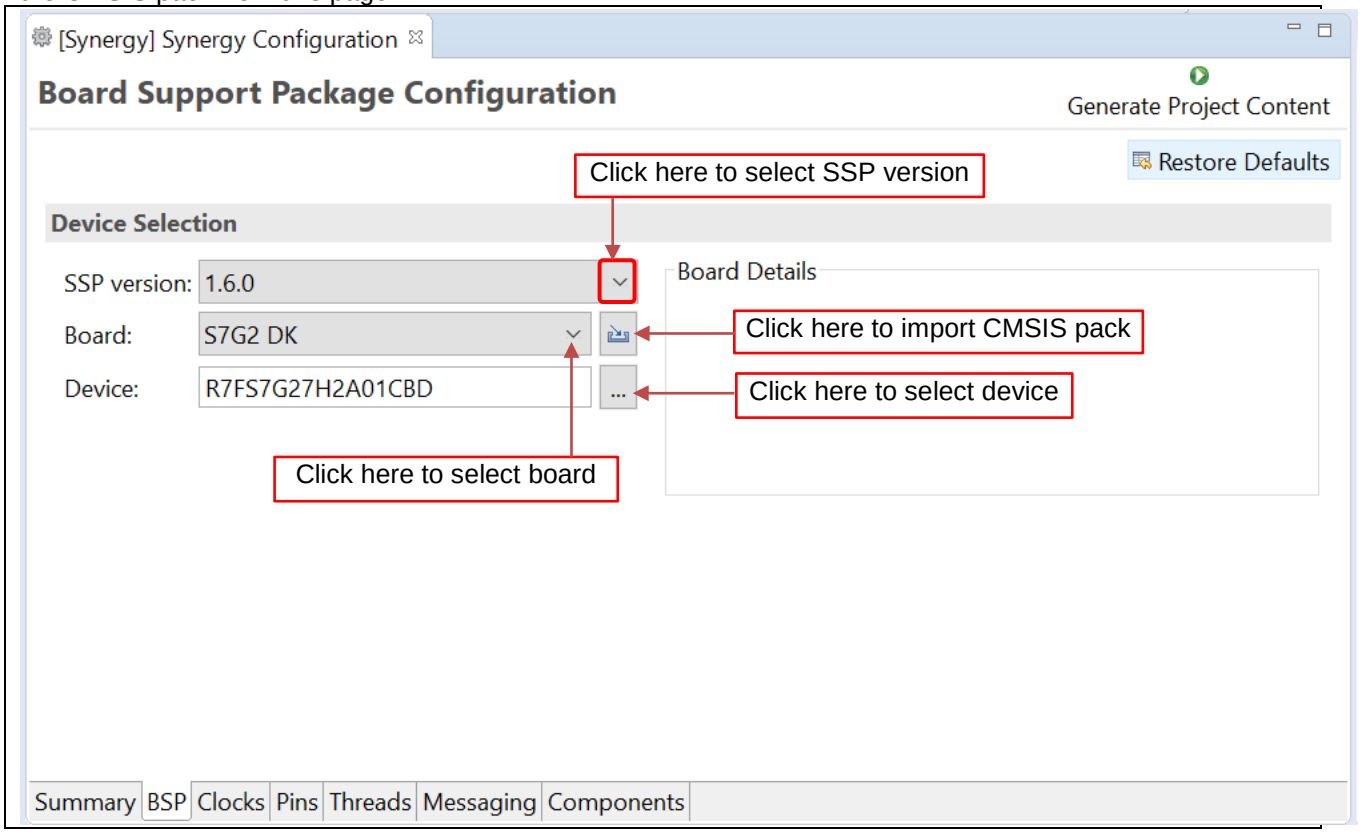


Figure 55. Synergy Project Configuration – BSP Page

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 MCU User's Manual*. To update the project, perform the following steps:

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

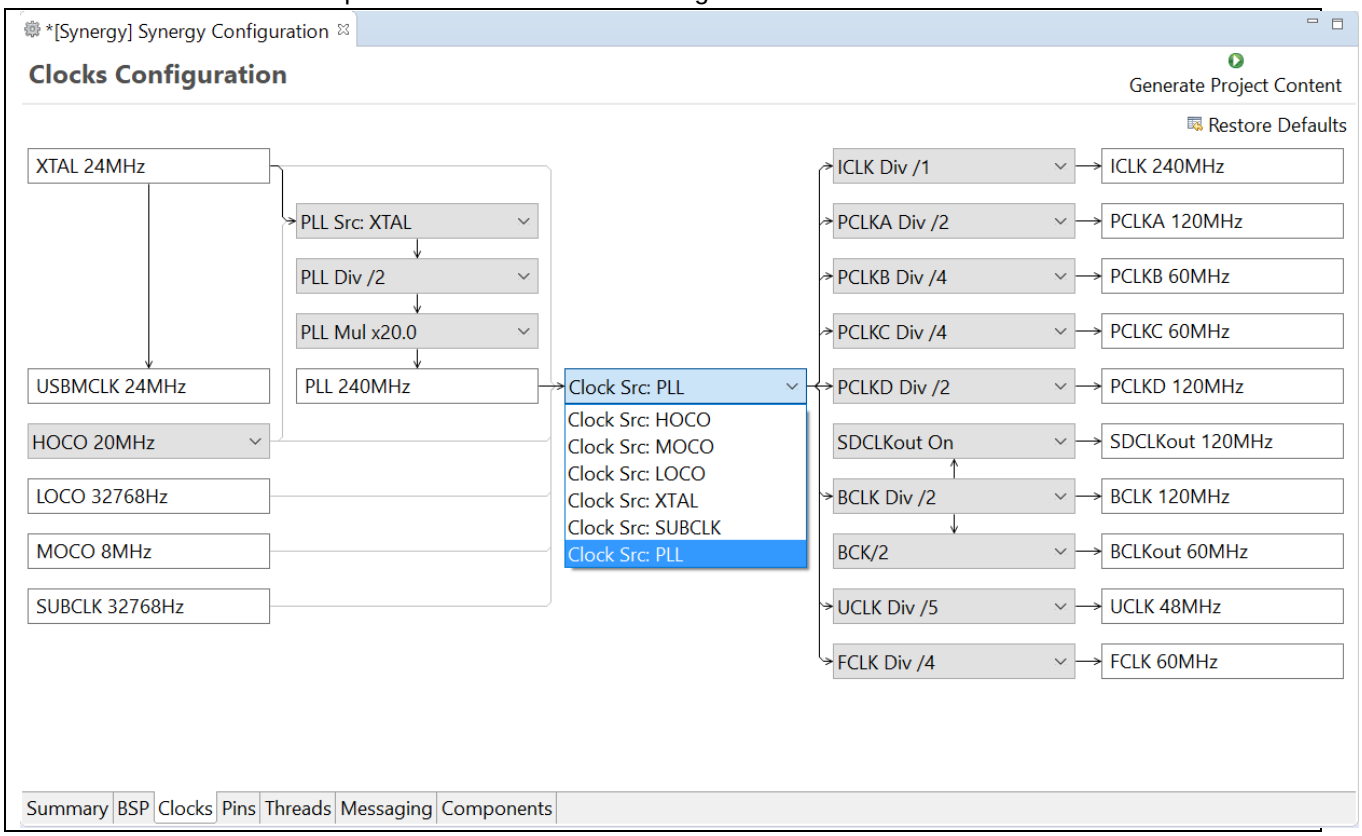
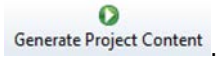


Figure 56. Synergy Project Configuration – Clocks Configuration

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.

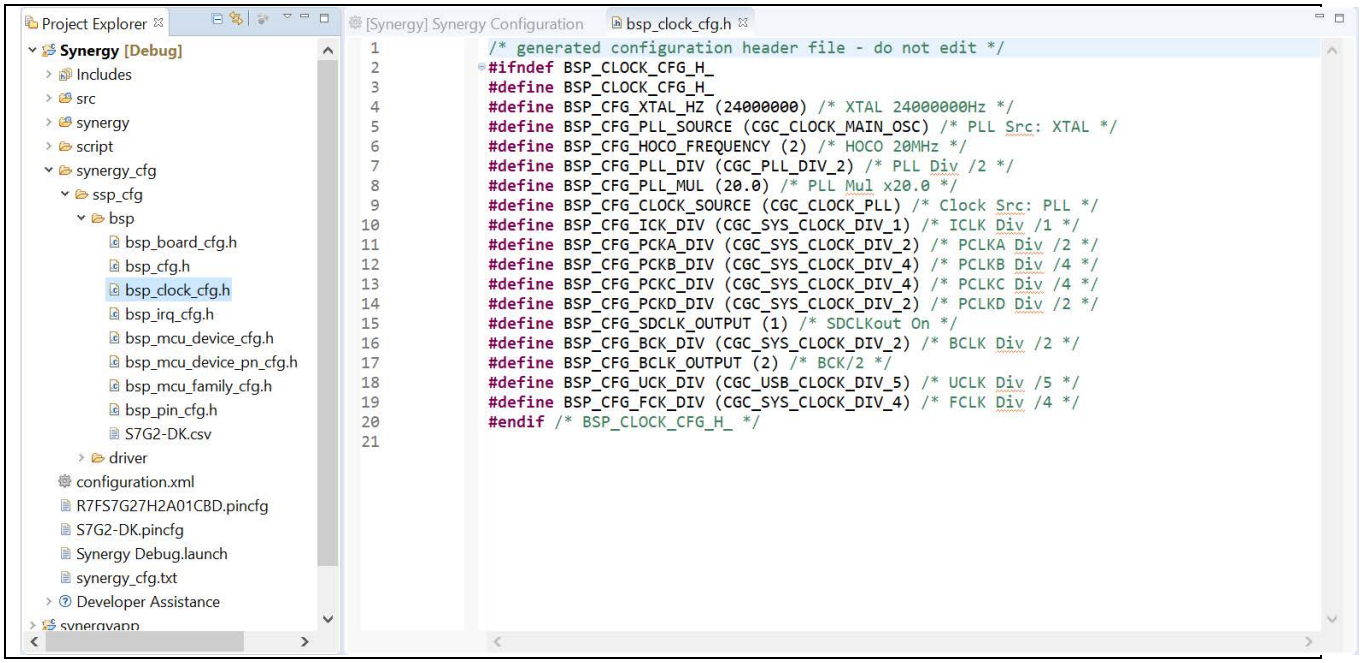


Figure 57. bsp_clock_cfg.h is updated

3.4.4 Pin Configuration Page

The Pin Configuration page provides a graphical user interface to generate the pin configuration settings for the project.

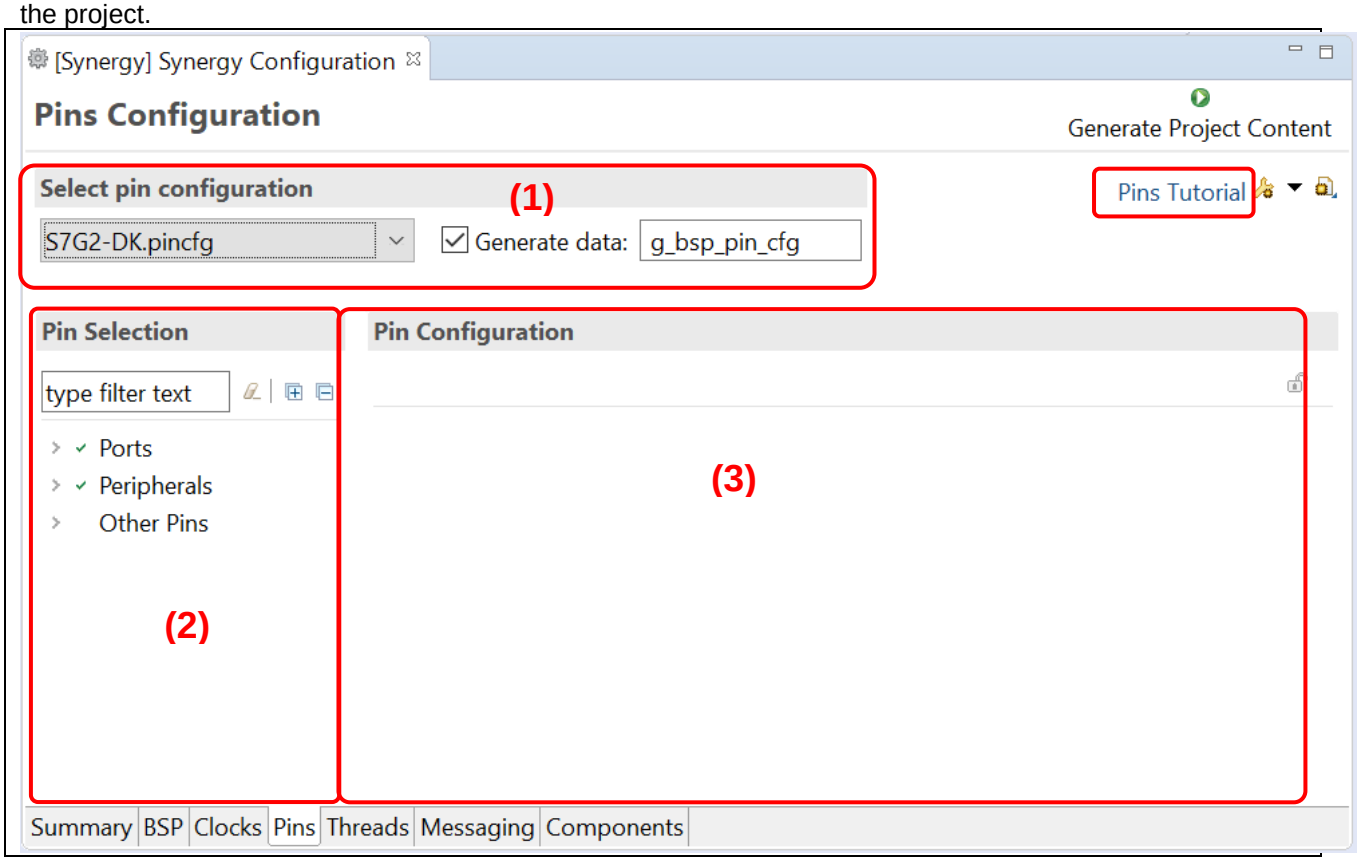


Figure 58. Synergy Project Configuration –Pin Configuration GUI

The Pin Configuration window has three 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** box 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.
   The user may 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.

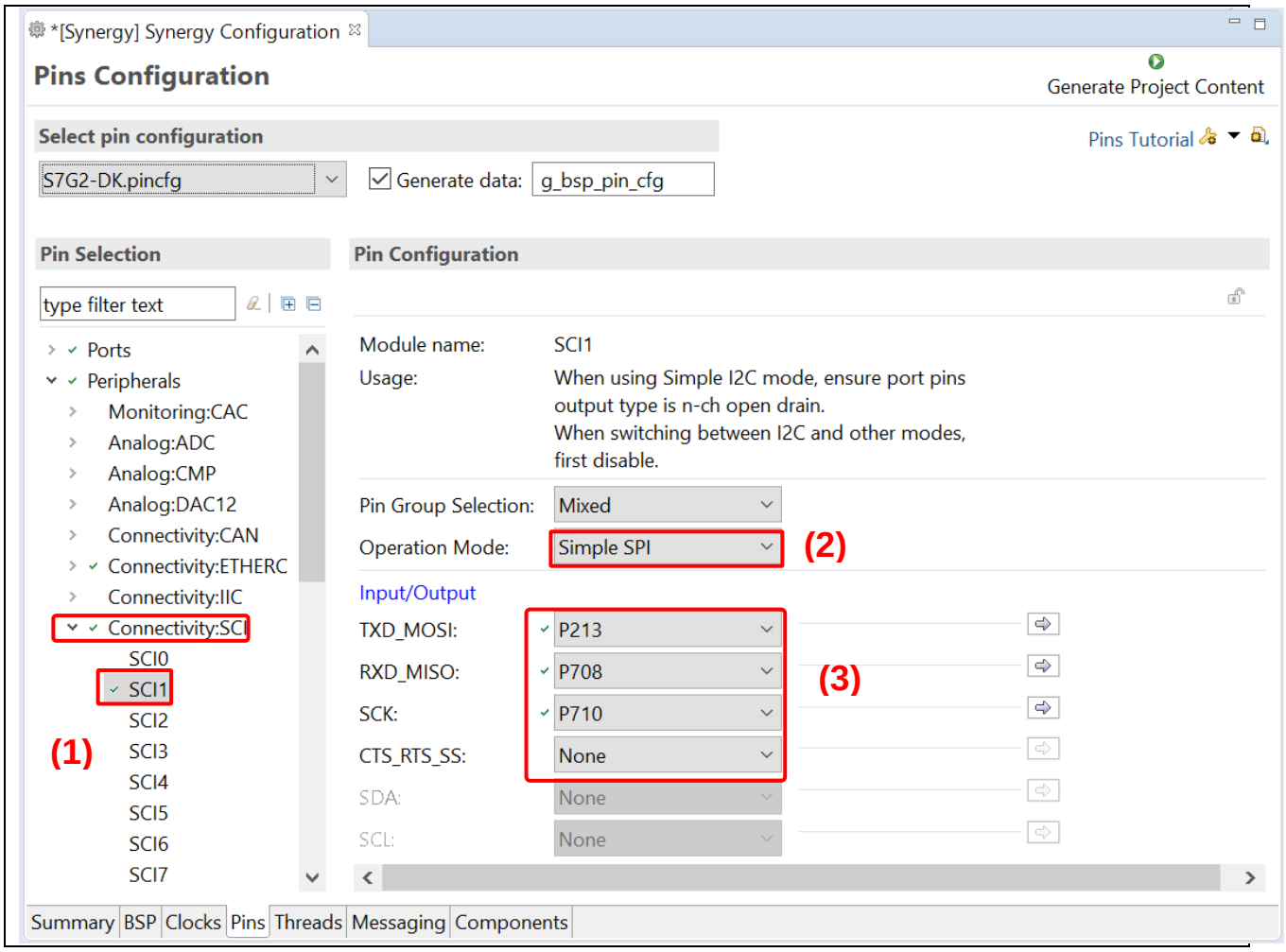


Figure 59. Synergy Project Configuration – Pin Configuration Setting (by Peripheral)

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

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

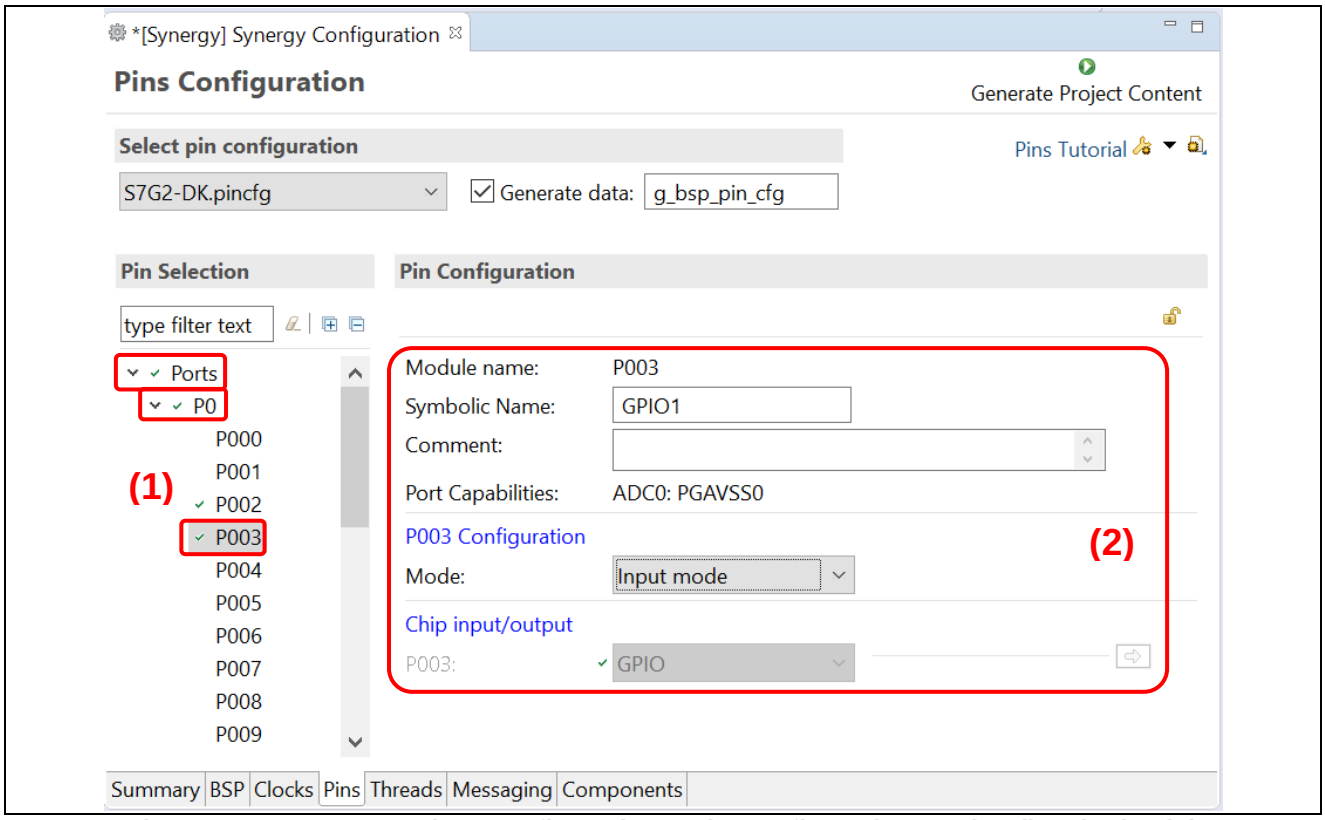


Figure 60. Synergy Project Configuration – Pin Configuration Setting (by single pin)

3. The Package view shows this pin change.

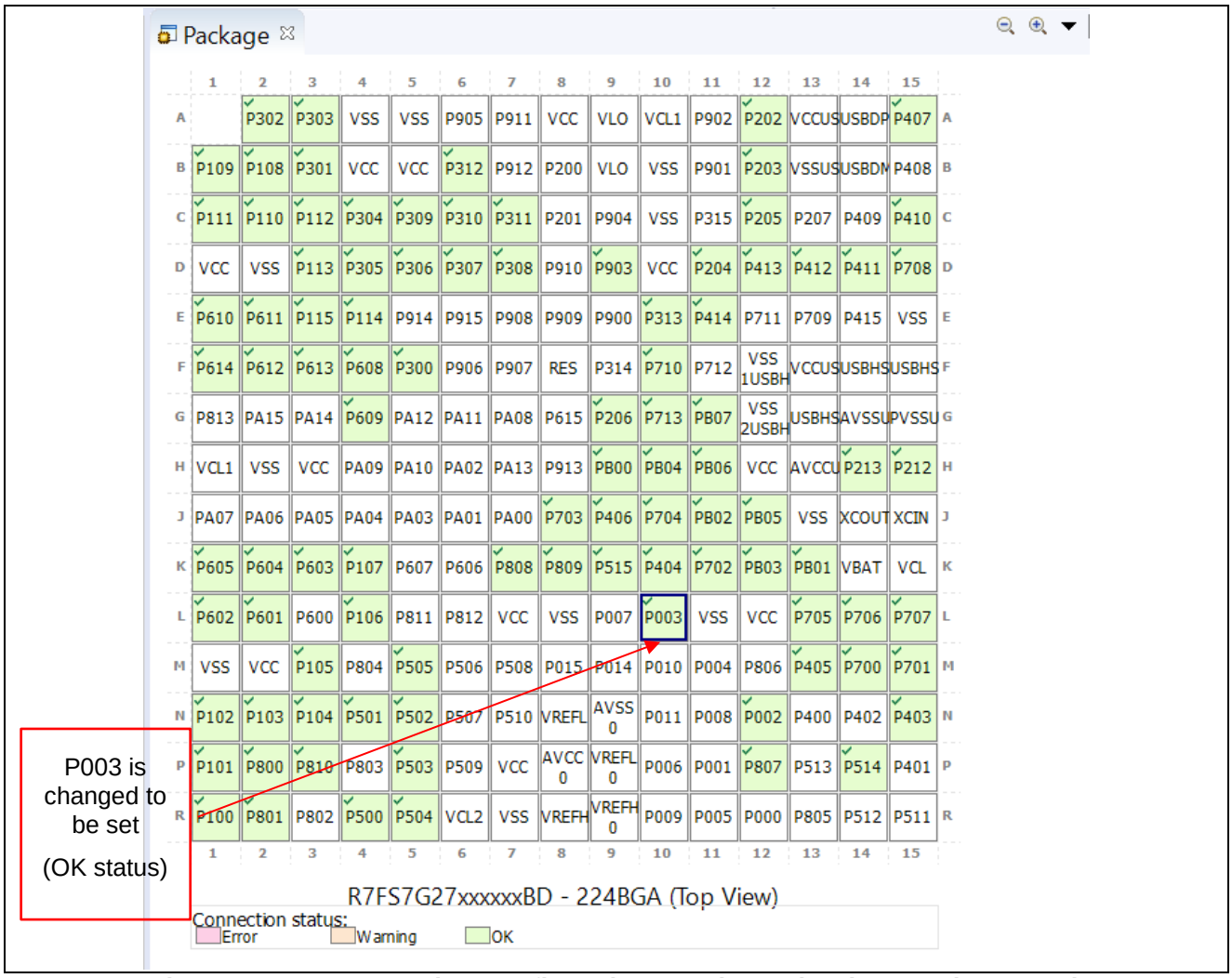


Figure 61. Synergy Project Configuration – Package View (Connection Status)

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.

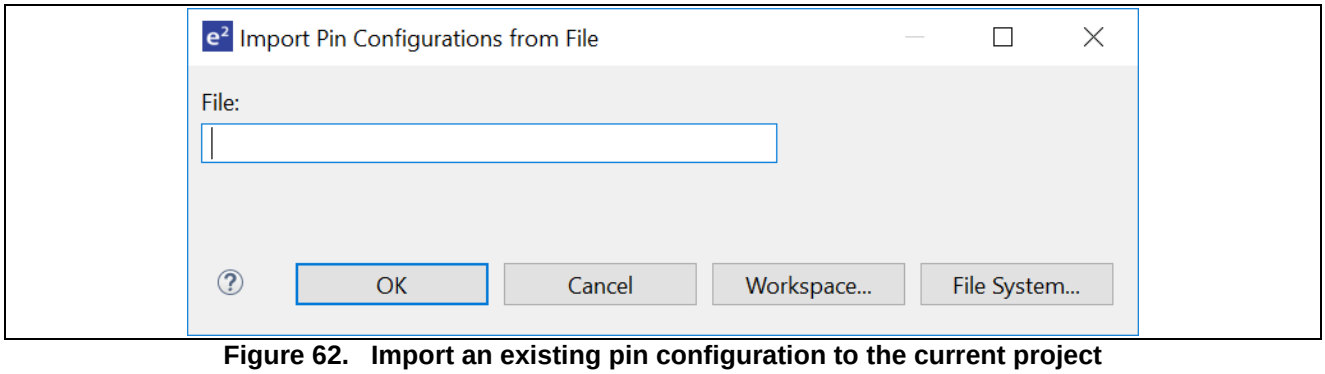


Figure 62. Import an existing pin configuration to the current project

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.

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.

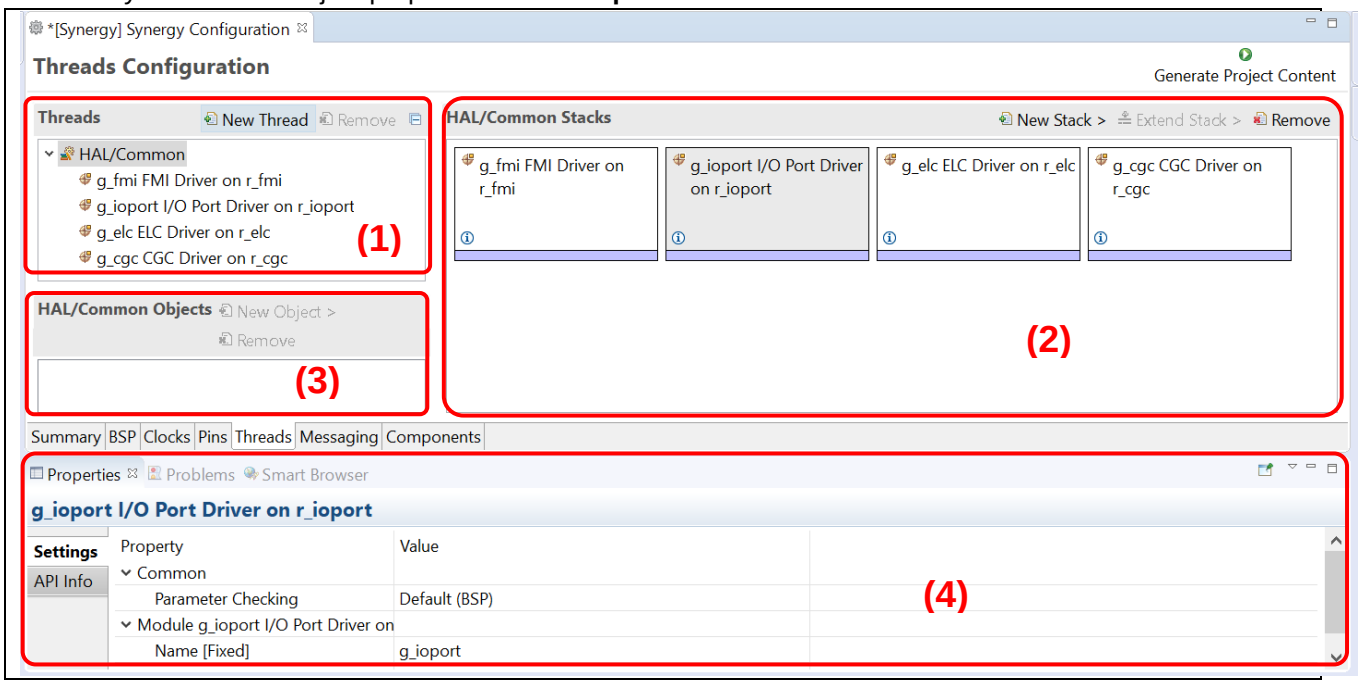


Figure 63. Synergy Project Configuration – Threads Configuration GUI

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

1. **Threads** pane: Add/remove threads. For details, see section 6.
2. **Stacks** pane: Add/remove SSP module instances, such as IO port, SCI, UART, and so on.
3. **Objects** pane: Add/remove kernel objects. For details, see section 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 using the following steps:

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.

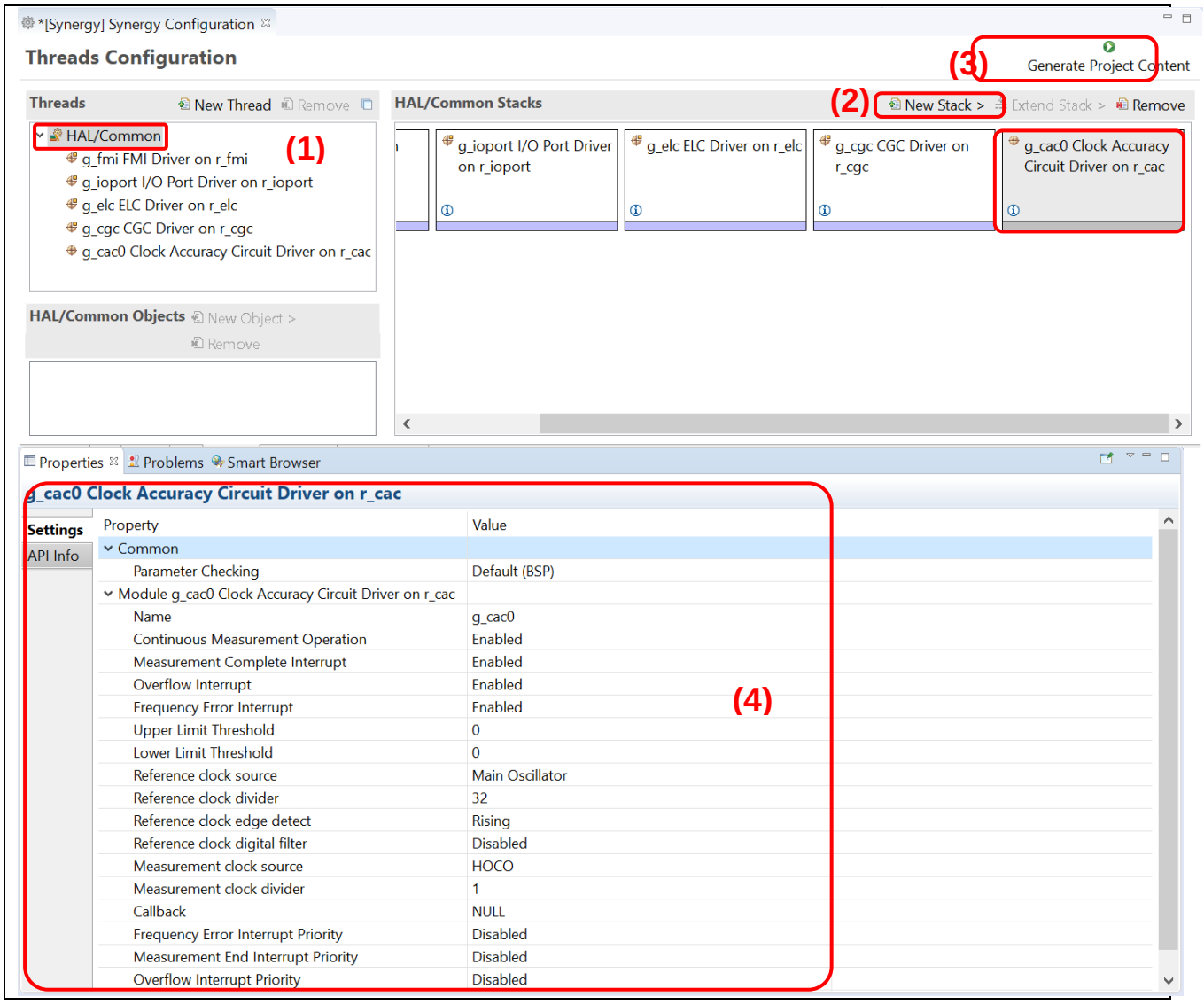


Figure 64. Synergy Project Configuration – Add New Module to Thread

Note: For another example, see section 6.1, General Purpose Timer Example in ThreadX. This section 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 are 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.

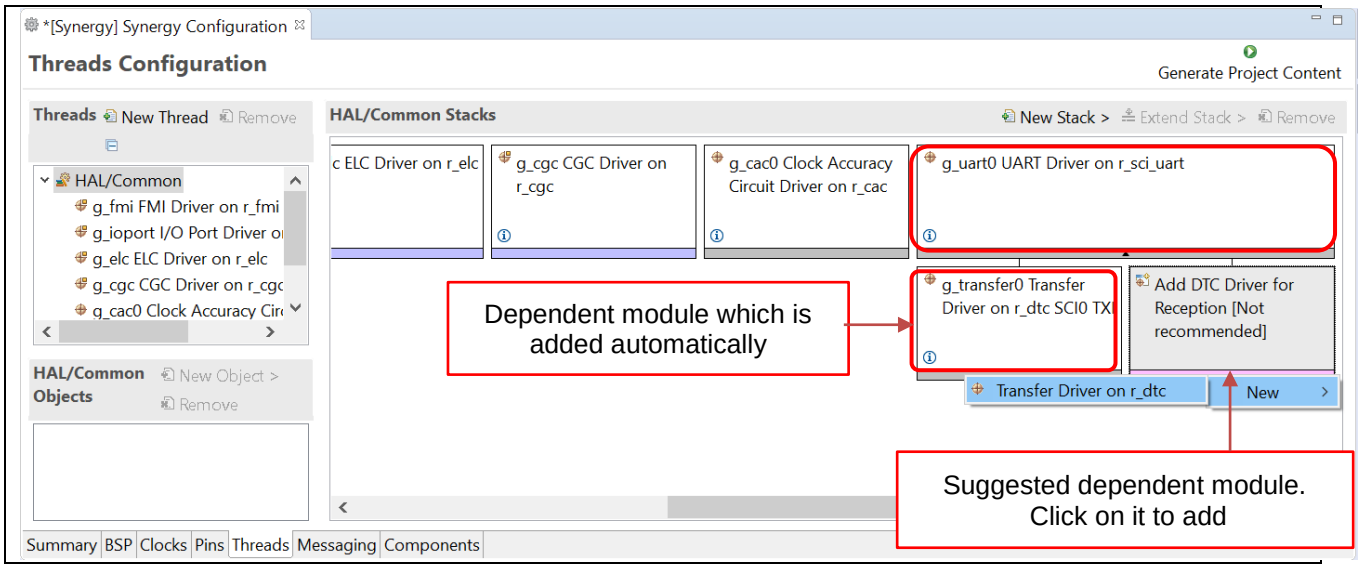


Figure 65. Synergy Project Configuration – Problem of Added Module

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

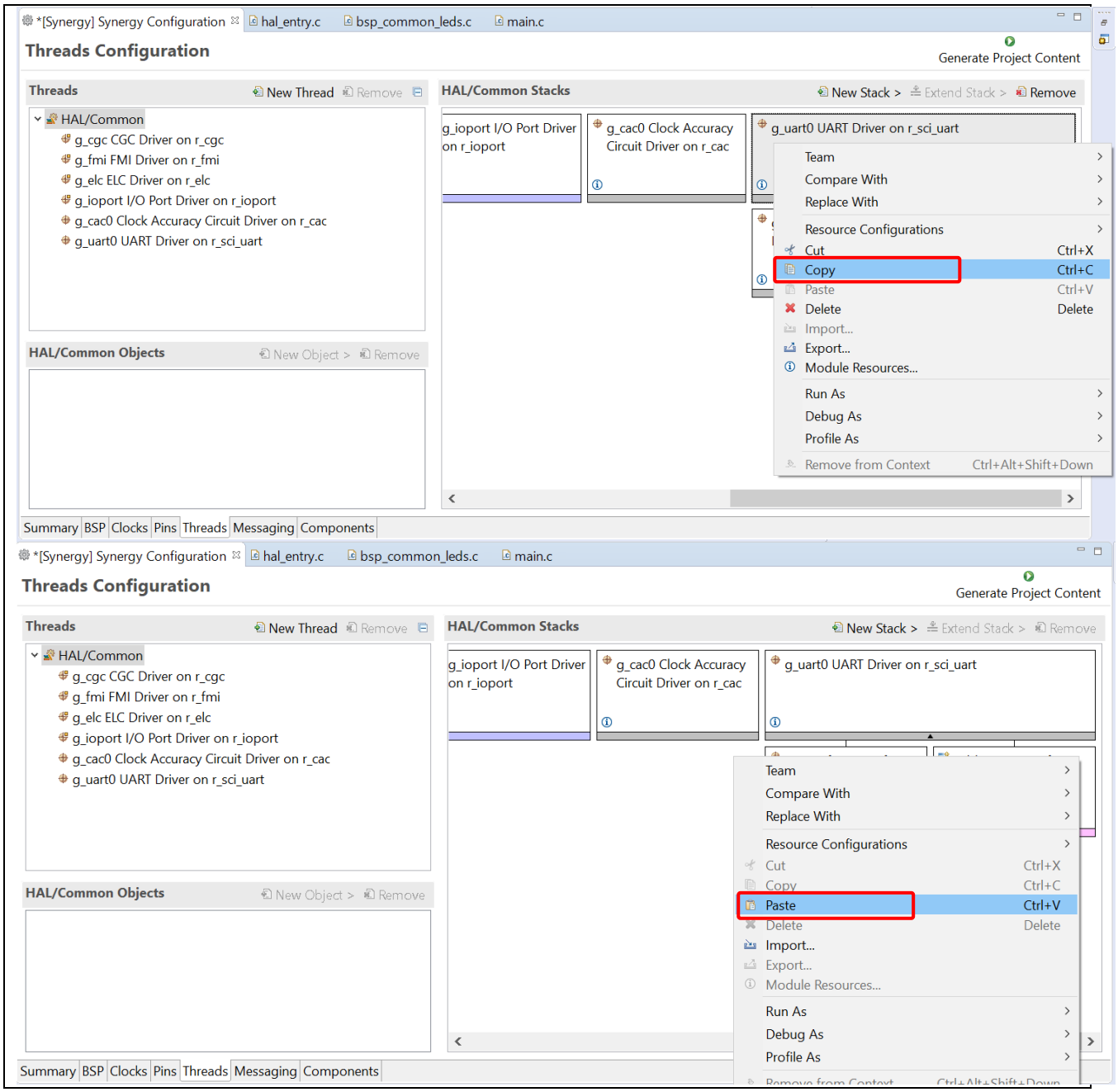


Figure 66. Copy and Paste operation

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.

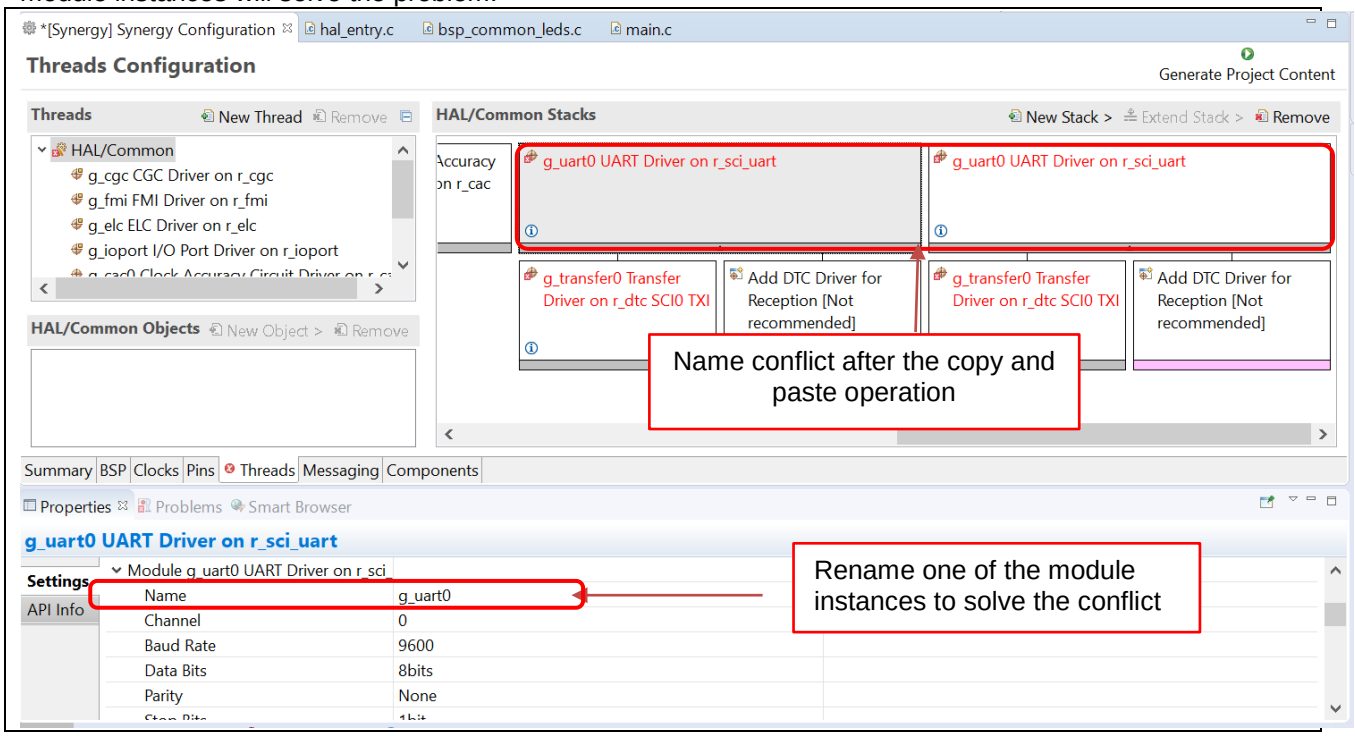
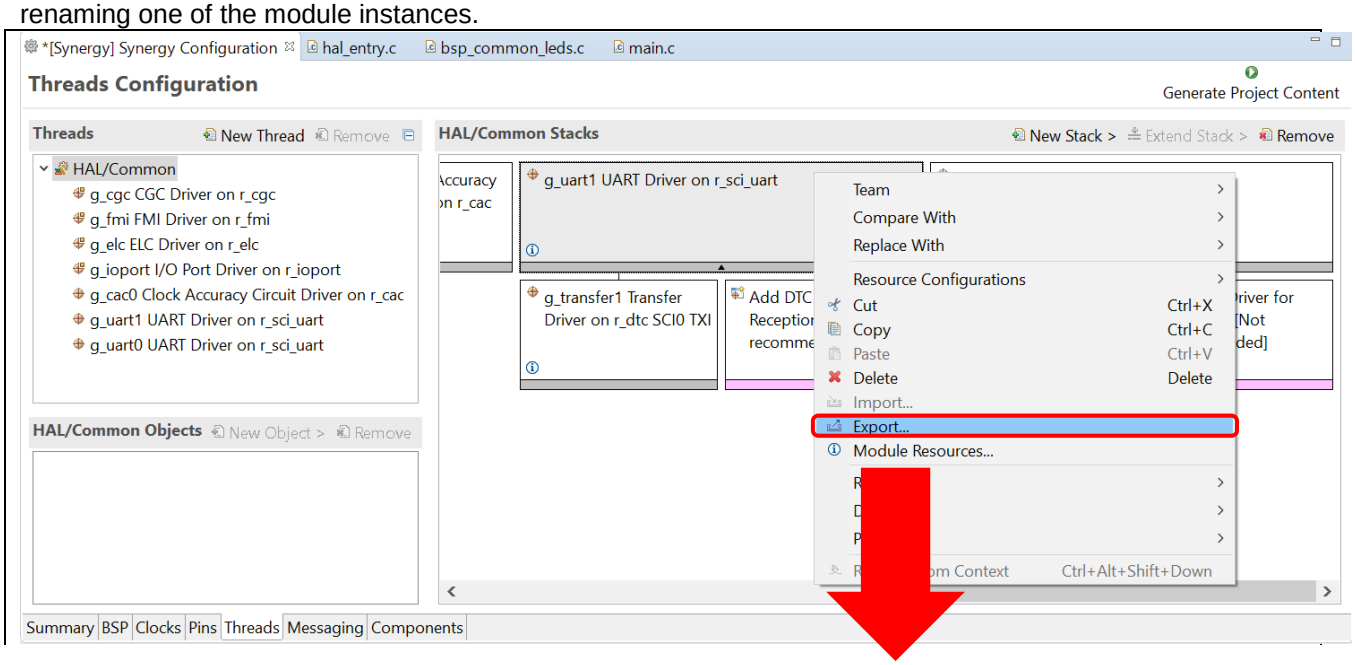


Figure 67. Module Instance name conflict

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.



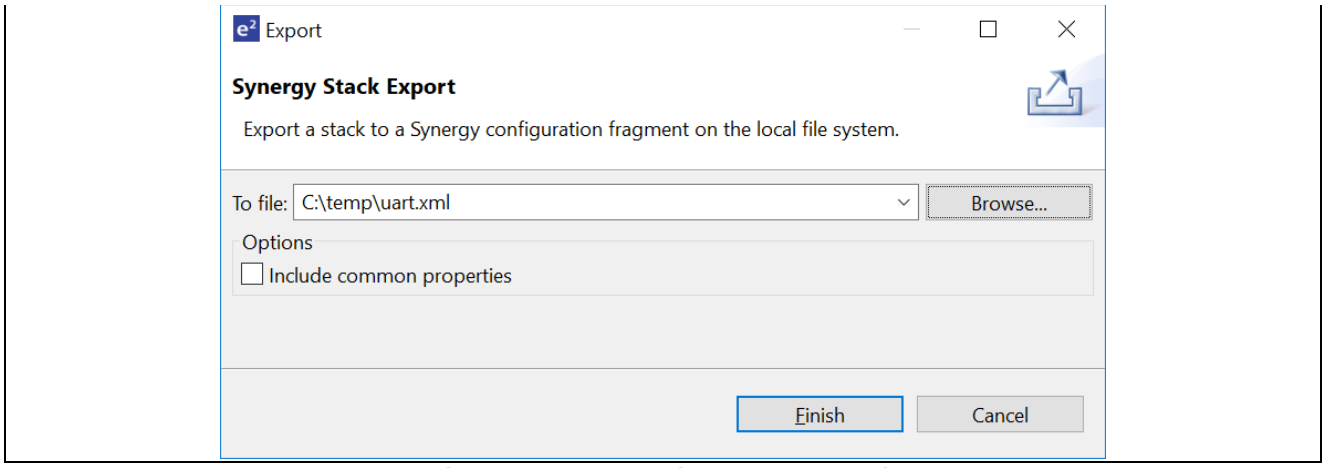


Figure 68. Export the Synergy stack

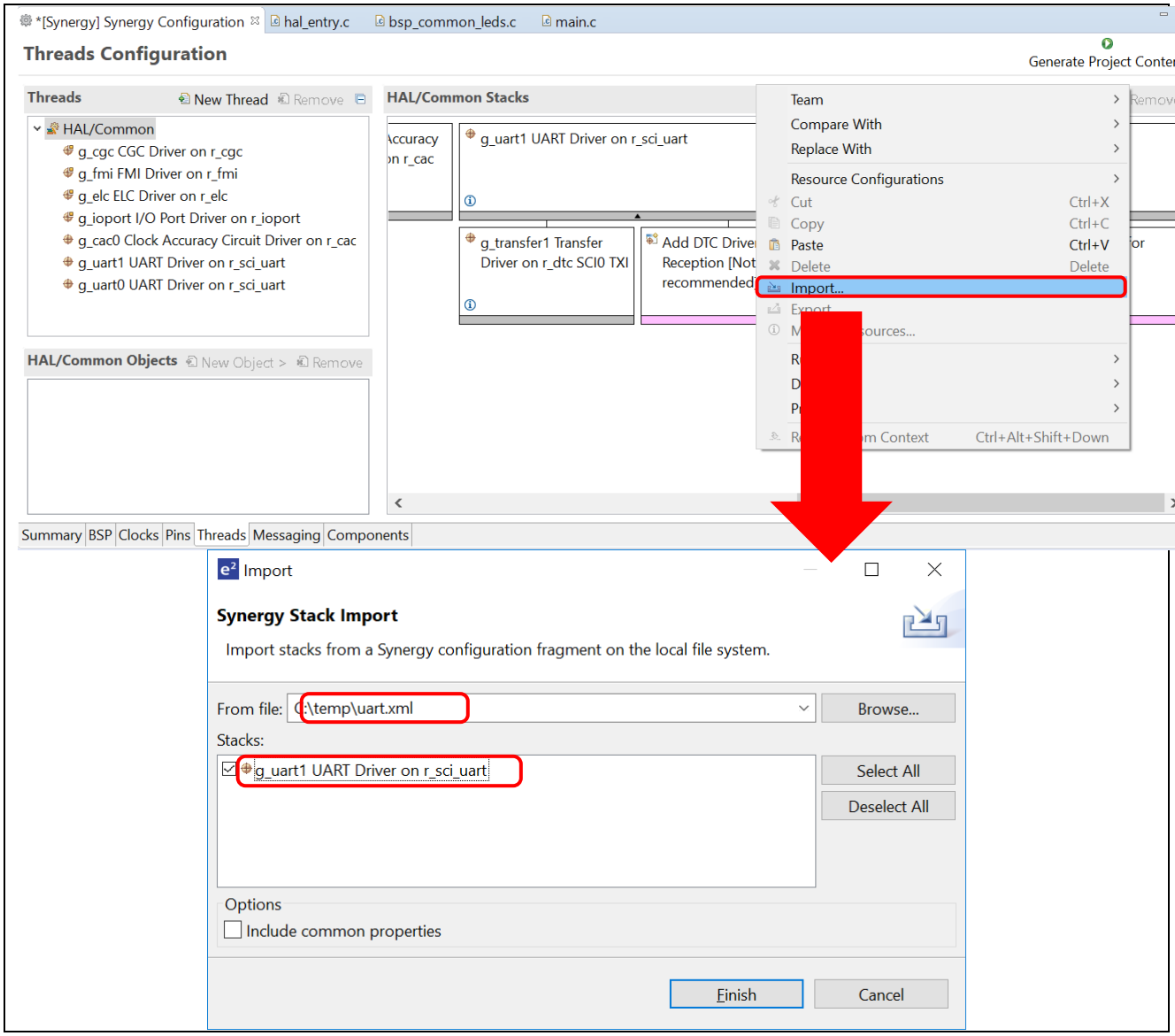


Figure 69. Import the Synergy stack

3.4.6 Messaging Page

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

The Messaging Page consists of three 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 receives 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.

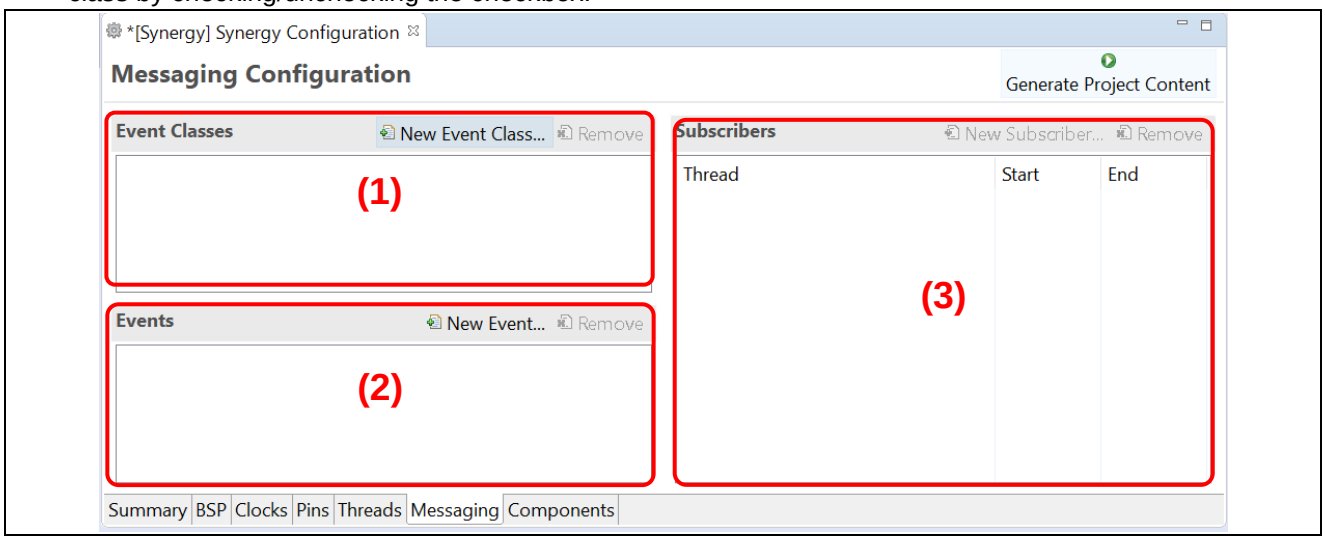


Figure 70. Synergy Project Configuration – Messaging Page

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

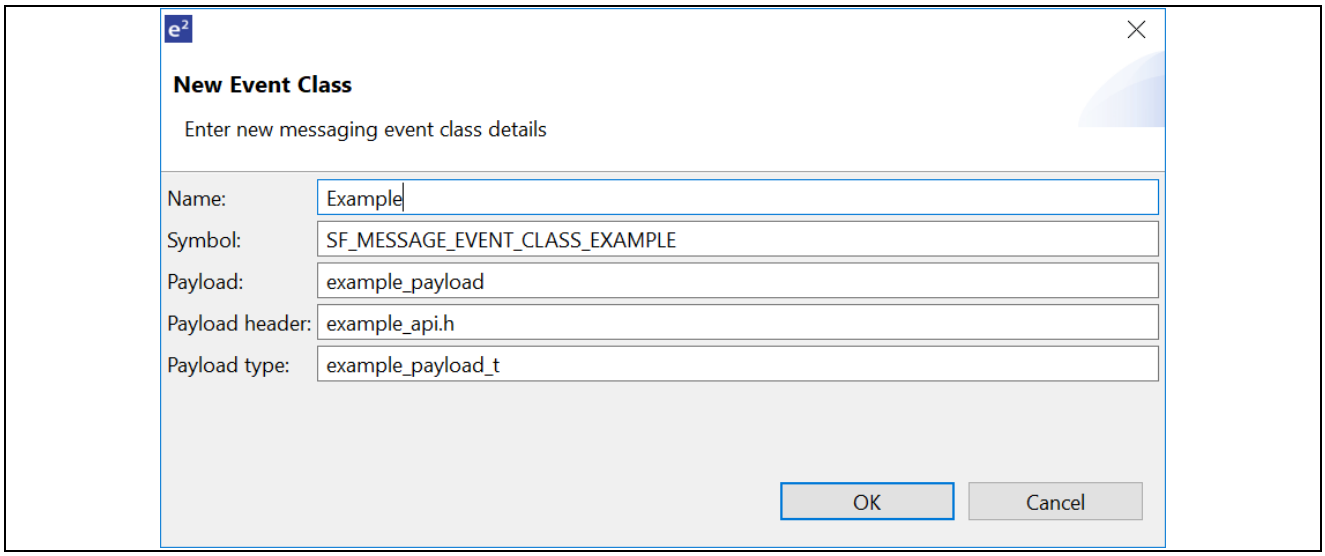


Figure 71. Messaging Page – Adding a new event class

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.

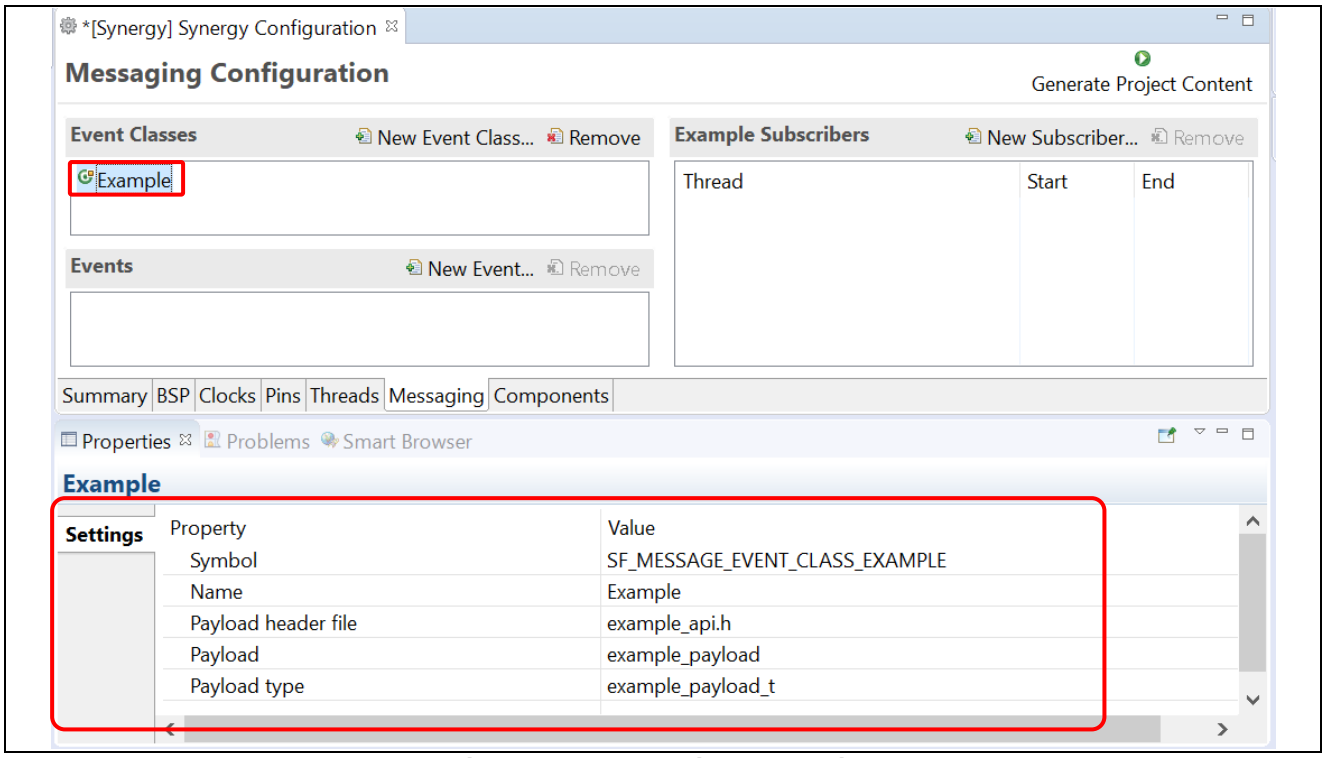


Figure 72. Messaging Properties

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 using the **Threads** page. The **Components** page is primarily used as a list of components available in the installed SSPs.

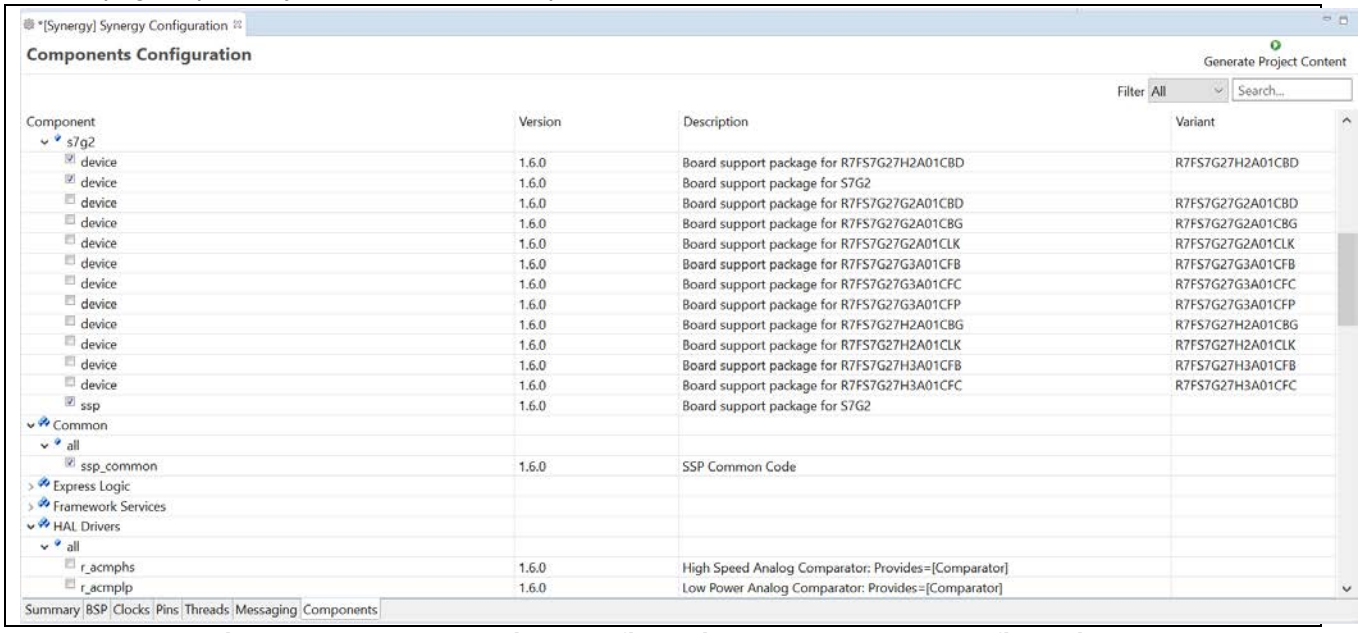


Figure 73. Synergy Project Configuration – Components Configuration

3.5 Editor hover

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

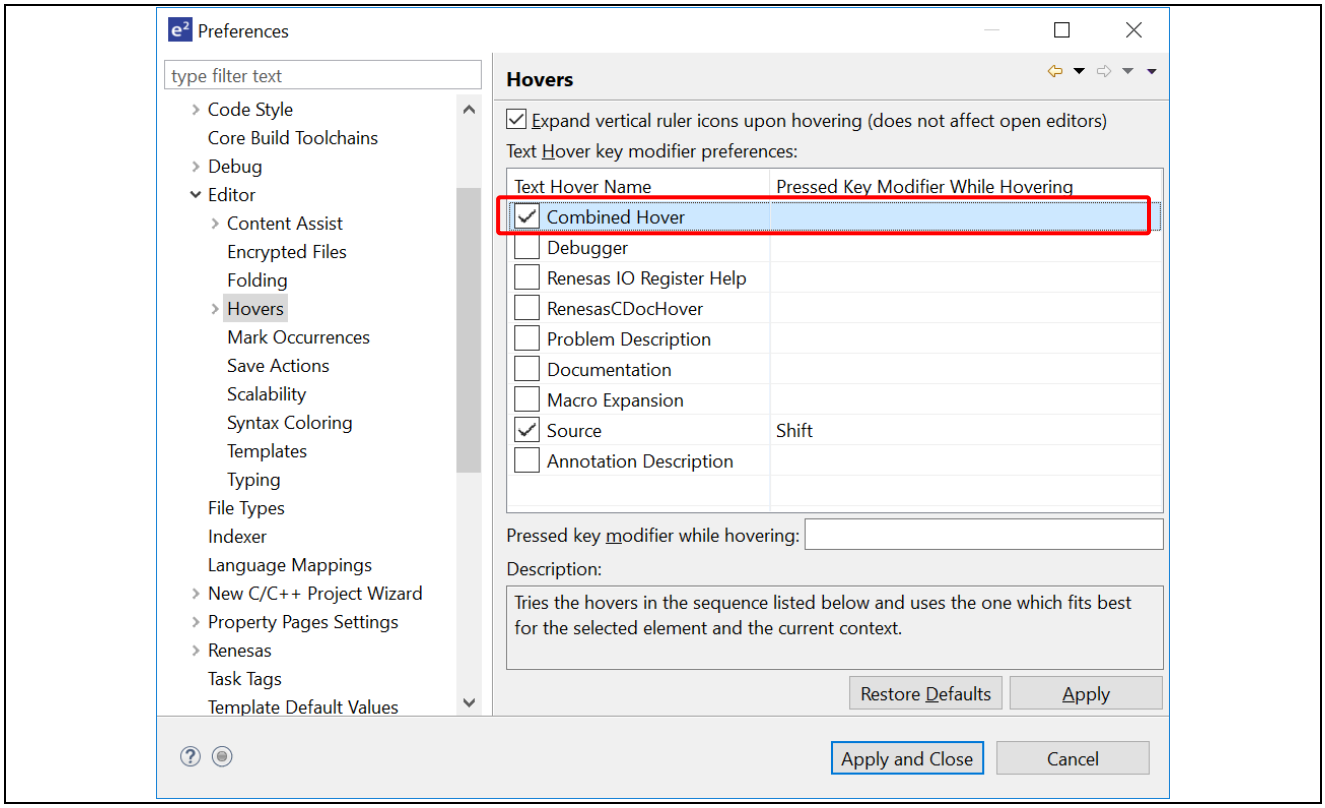


Figure 74. Hover settings

To enable hover, check **Combined Hover** box. To disable it, uncheck this box. By default, it is enabled.

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

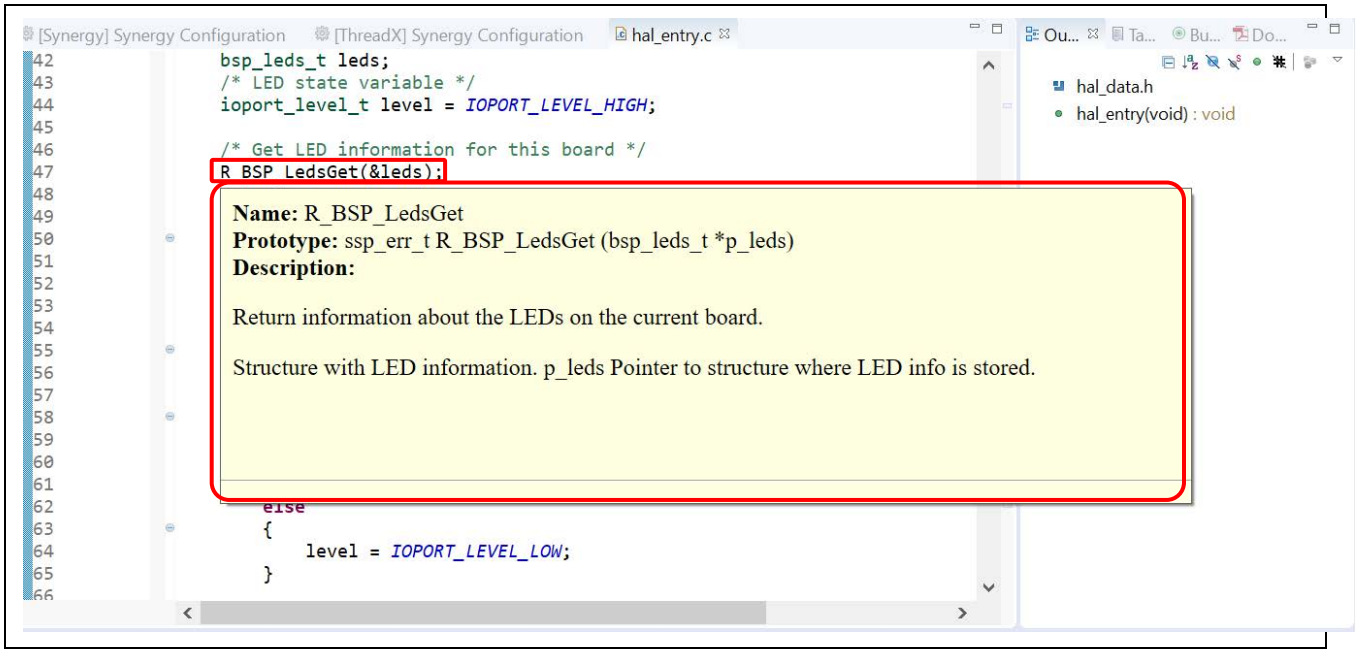


Figure 75. Information from hover function

3.6 Developer Assistance

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

1. Expand the project explorer to view Developer Assistance

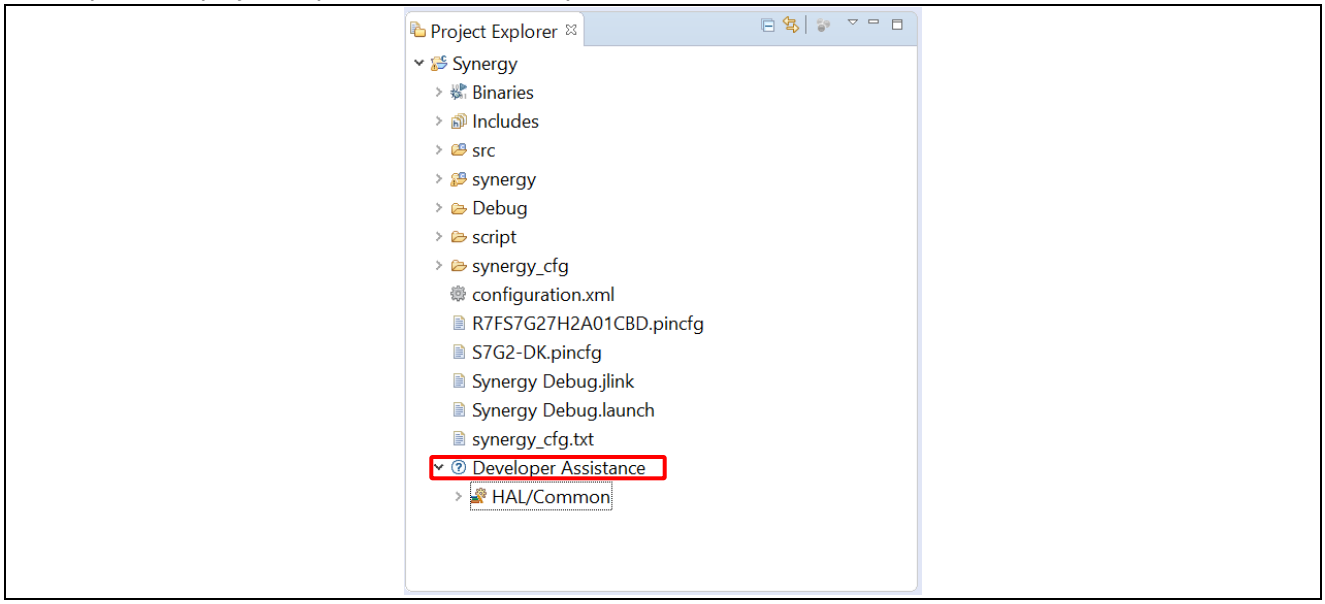


Figure 76. Synergy Developer Assistance

2. Expand a stack module to show its APIs

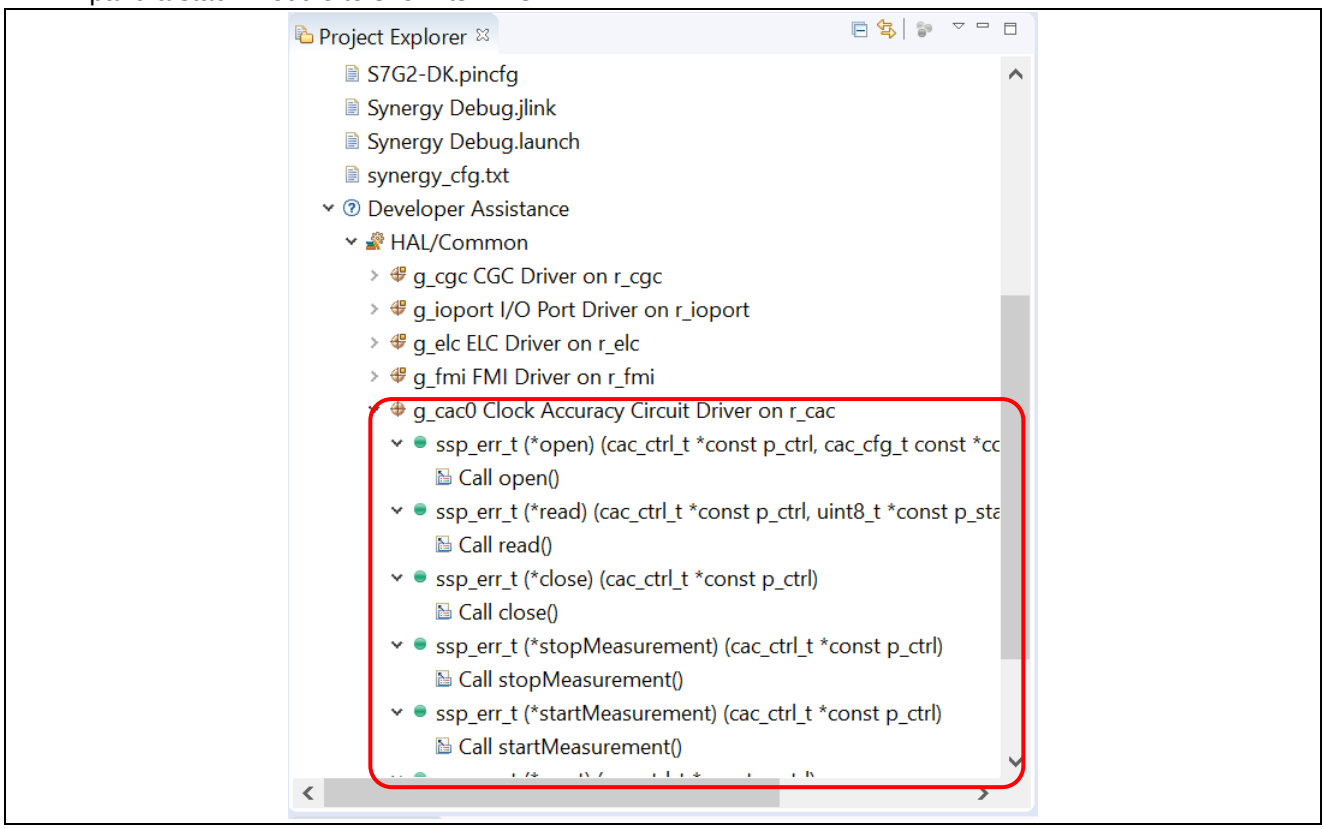


Figure 77. API information

3. Dragging and dropping an API from Develop Assistance to source file helps to write source code quickly.

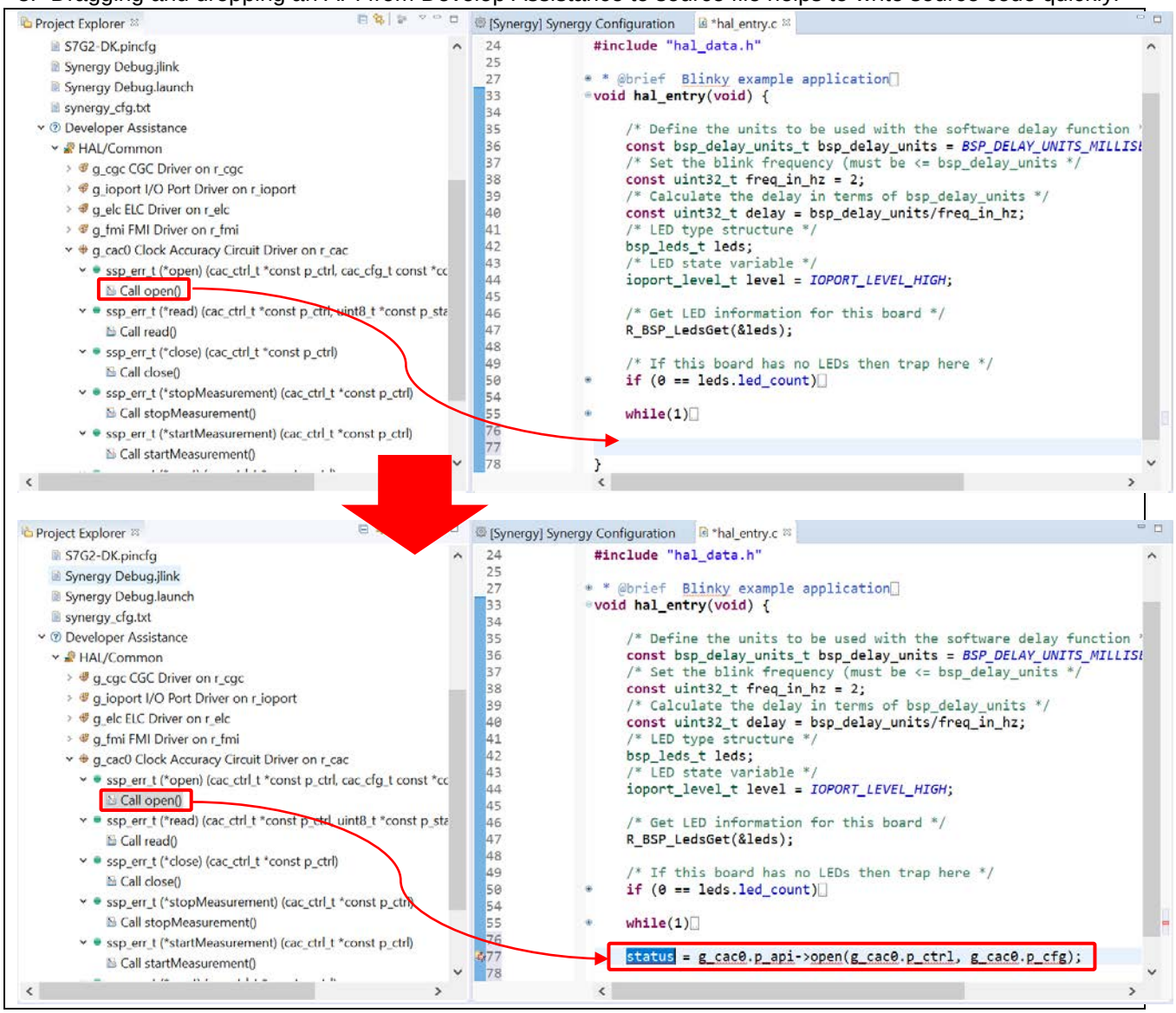


Figure 78. Developer Assistance – Drag and drop an API

4. Building

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

4.1 Build Configurations

The default build option generates when a project is created; it can usually be used to build the project.

However, if changing build options is necessary (such as the toolchain version, optimization options, and so on), do the following steps before building the project.

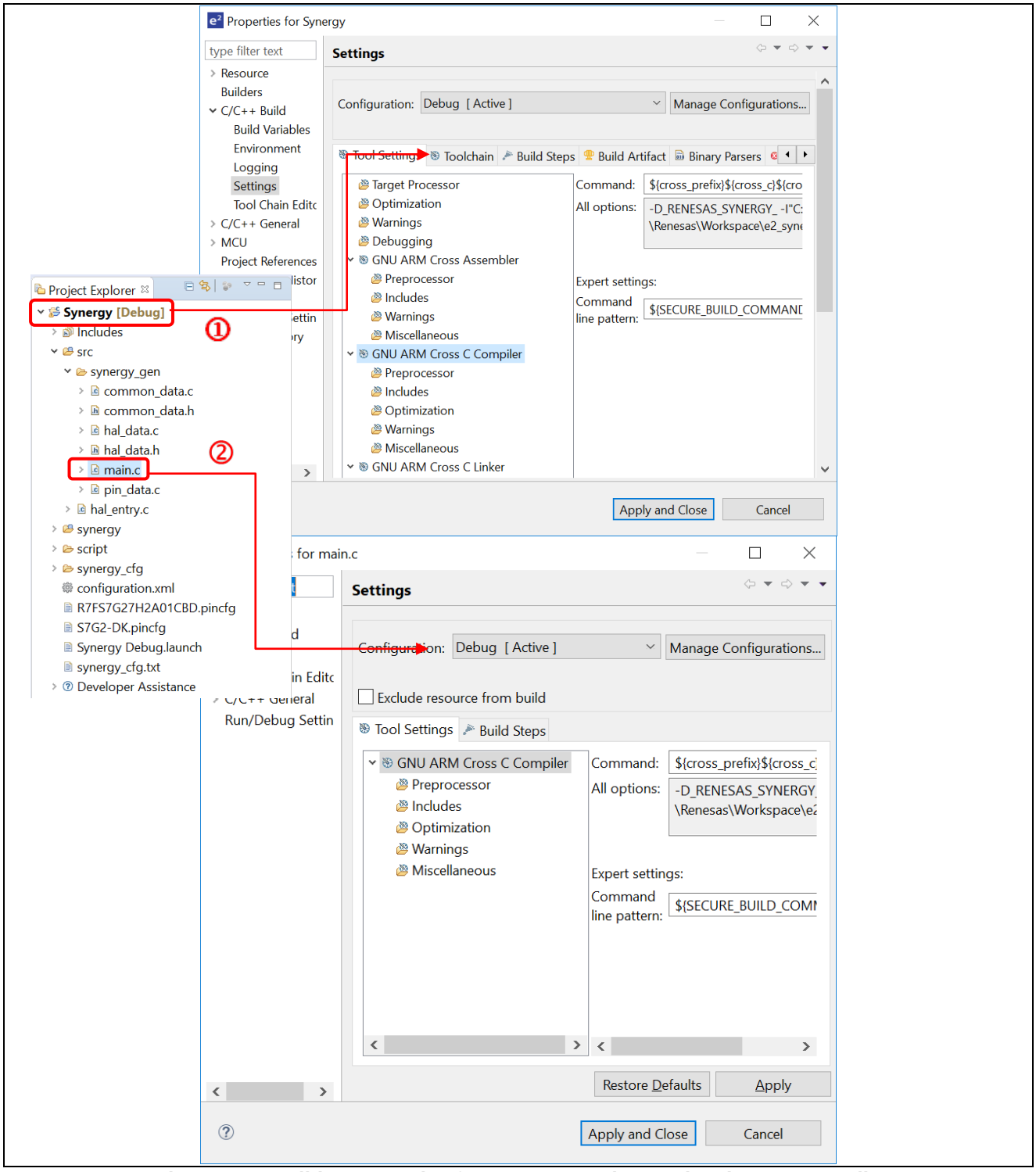


Figure 79. Build – Properties for Synergy Project and main.c Source File

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 **C/C++ Build** option to view or edit the configuration settings.

The Properties window is supported at 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.

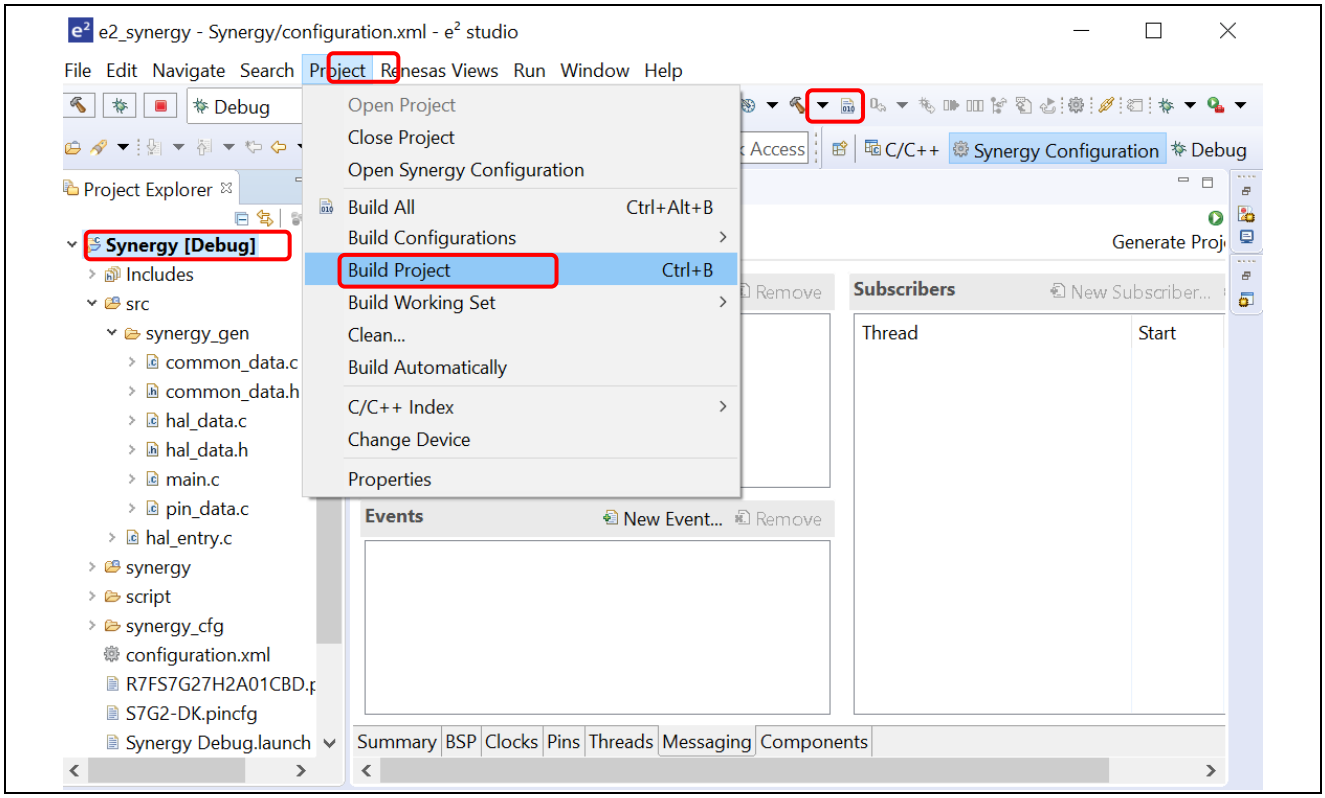


Figure 80. Build – Building a Sample Project

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 in the **Project Explorer** view to pop up the context menu.
2. Select **Save build settings report** to save the build settings report.

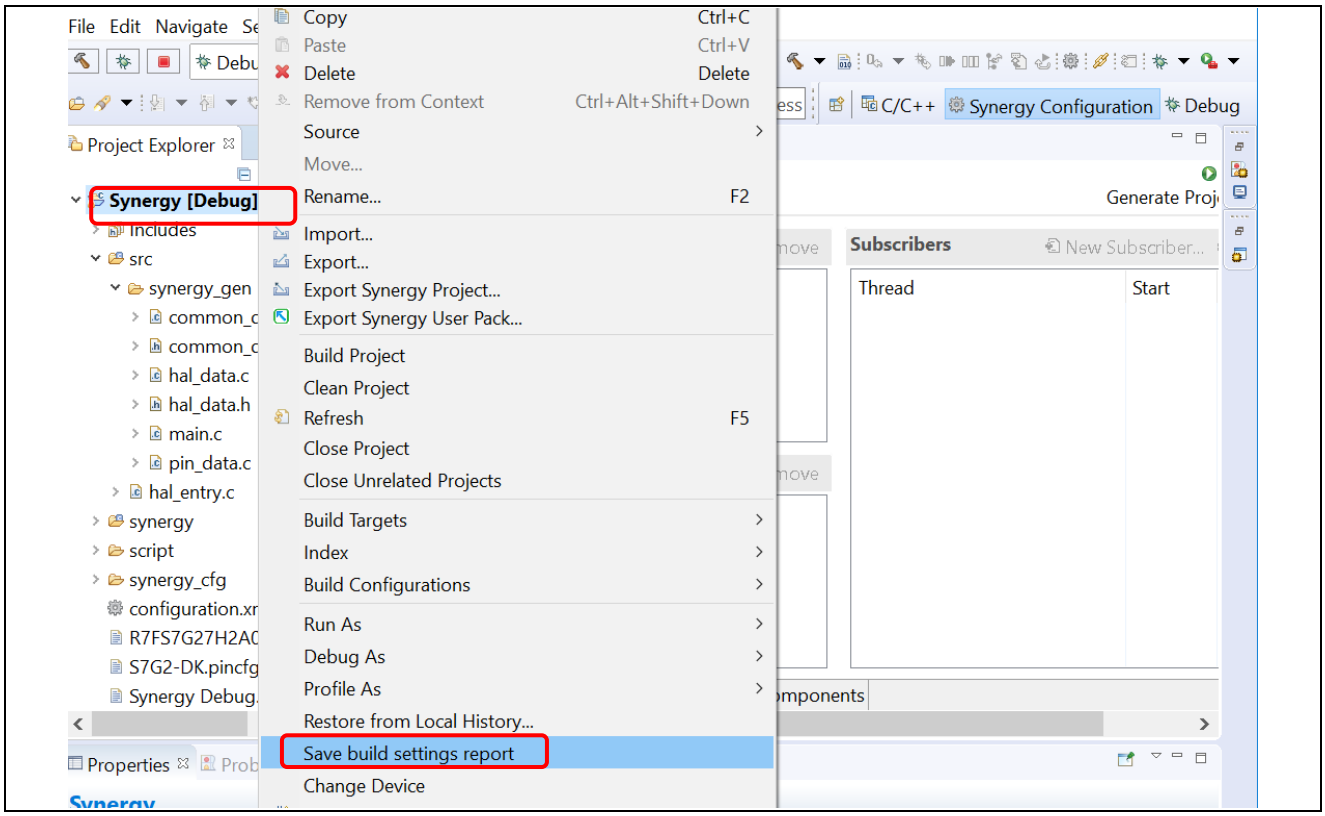


Figure 81. Build – Saving the build settings report

5. Debugging

This section describes the debug configuration usage 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 is based on the following hardware configuration: J-link Arm emulator and the Synergy DK-S7G2 board.

See the *TraceX® User's Manual* available in Synergy Platform website (www.renesas.com/synergy) to debug ThreadX-based projects using TraceX®. Debugging ThreadX projects using TraceX is not covered in this document.

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

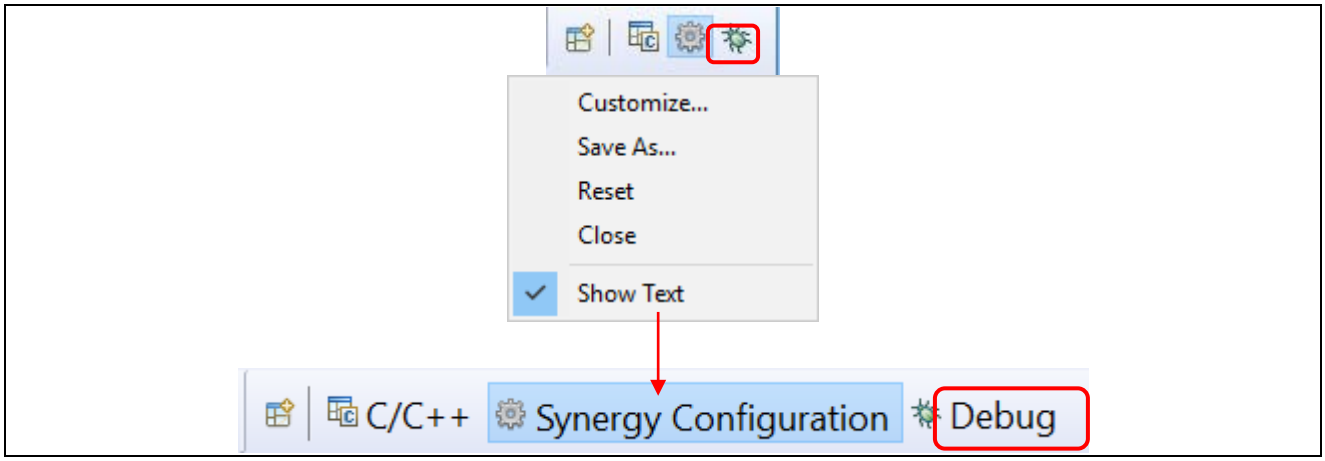


Figure 82. Debug – Switch to Debug Perspective

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, along with `configuration.xml` in the editor window, opens the Synergy configuration, as well as the Package and Properties views for project configuration settings.
- The **C/C++** perspective has views to help the user develop C/C++ programs.

If a user attempts to connect the debugger when not in the **Debug** perspective, e² studio prompts the user to switch to the **Debug** perspective.

One or more perspectives can exist in a single Workbench setup. Users can customize perspectives or add new ones.

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.

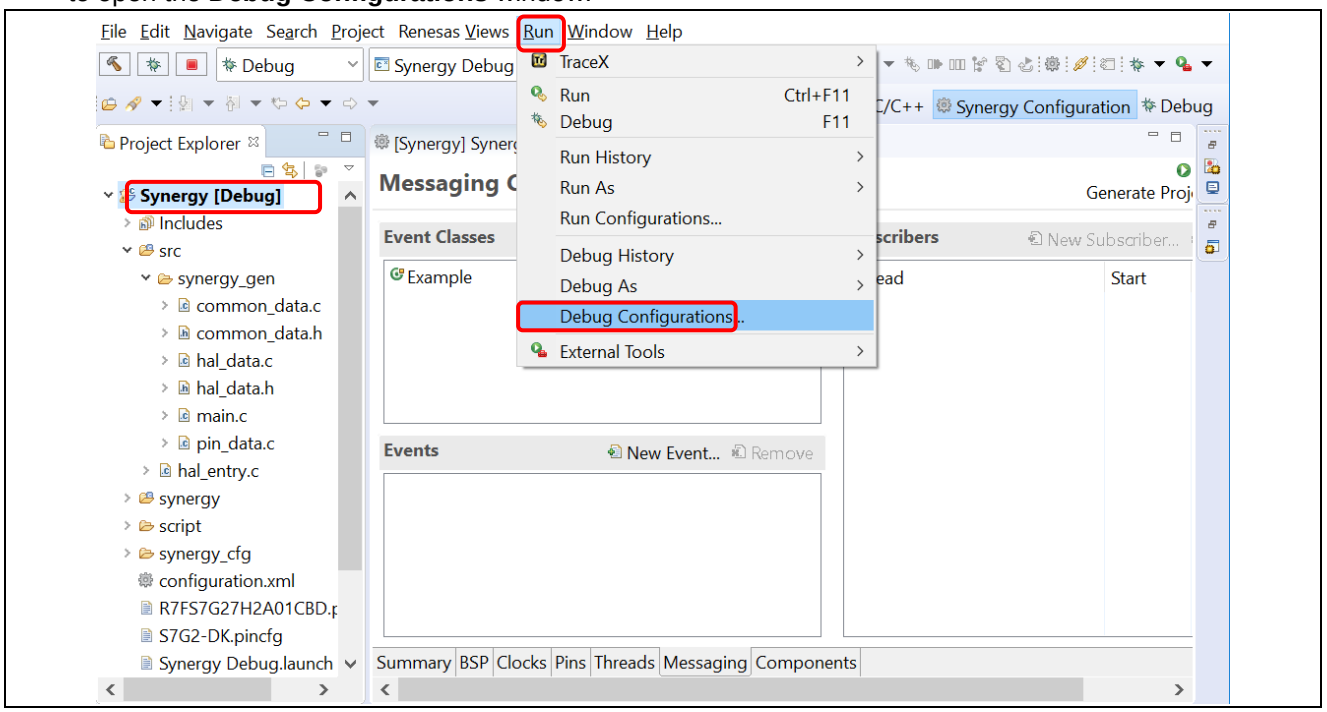


Figure 83. Debug – Opening the Debug Configurations Window

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`) in the project build folder.

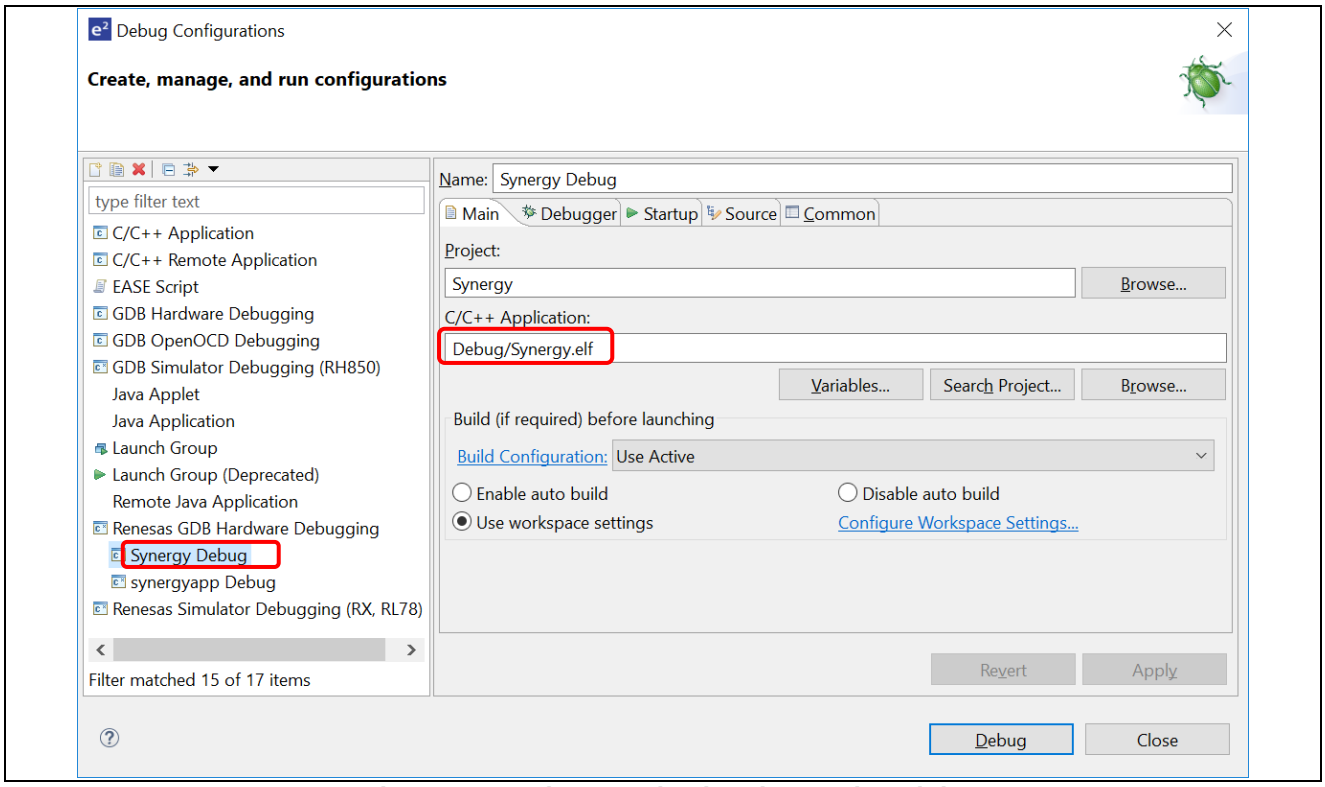


Figure 84. Debug – Selecting the Load Module

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.

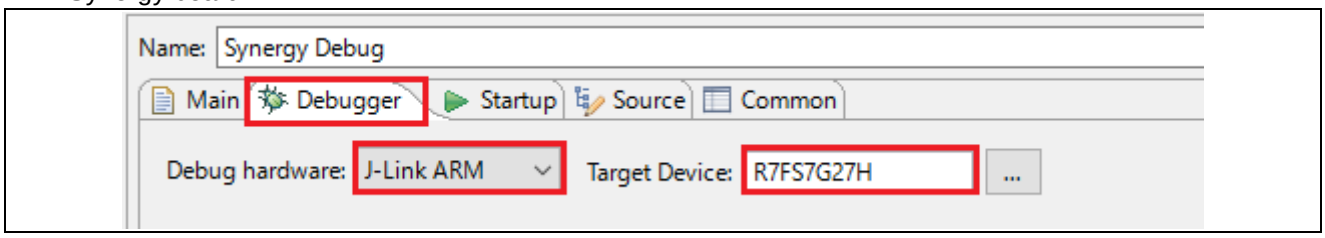


Figure 85. Debug – Changing the Connection Settings

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`.

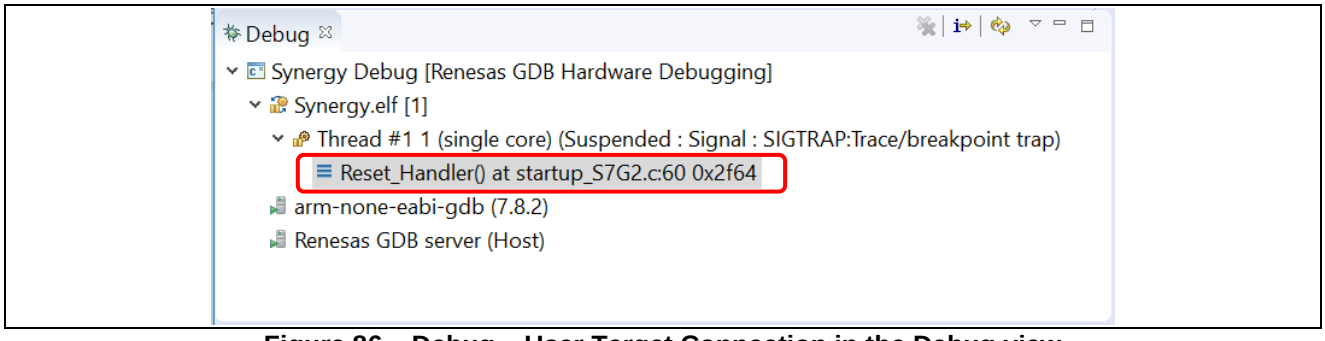


Figure 86. Debug – User Target Connection in the Debug view

5.2 Creating a New Debug Configuration

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

1. Open the Debug Configuration window (see Figure 83).
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.
3. The new debug configuration can be configured per section 5.1.

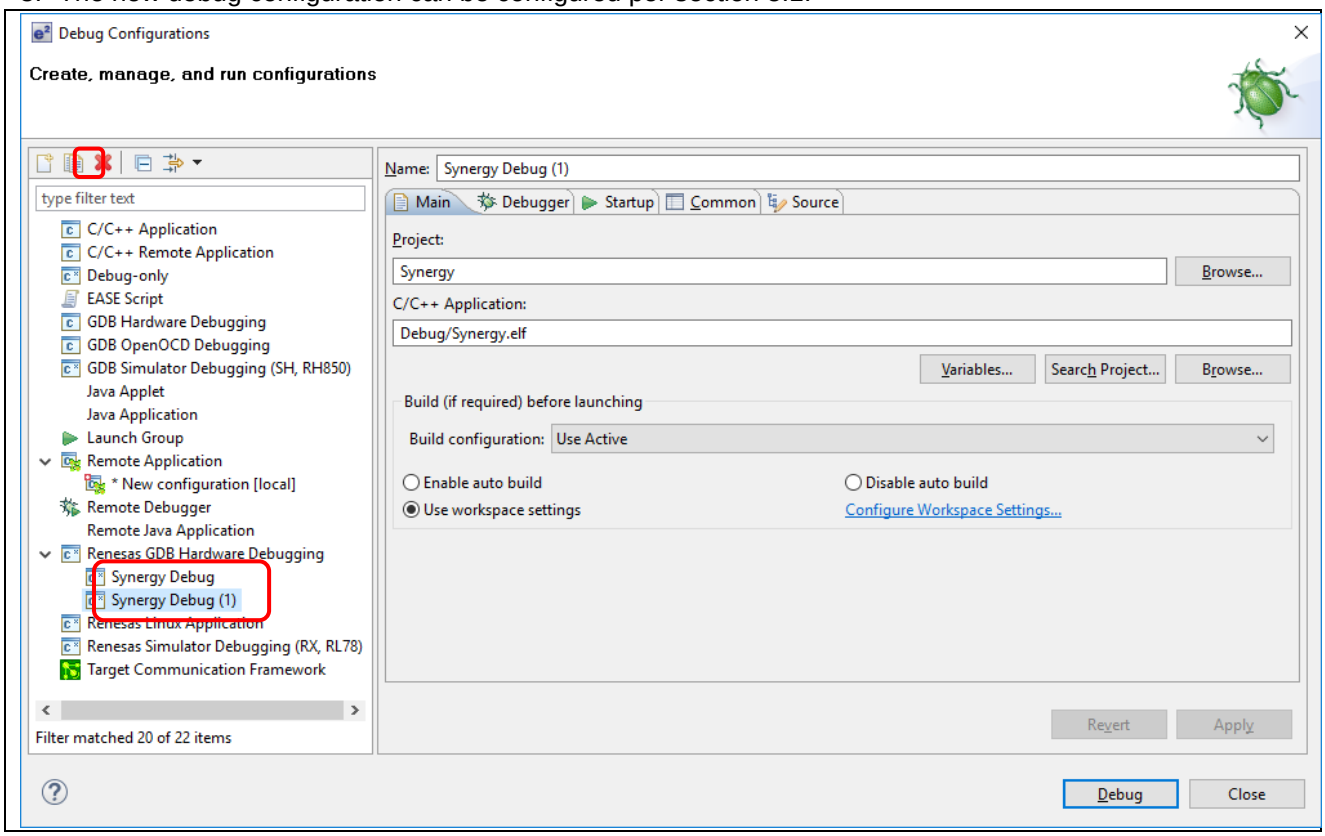


Figure 87. Debug – Duplicating a Selected Debug Launch Configuration

5.3 Basic Debugging Features

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

- Standard GDB Debug (supported by Eclipse IDE framework): Breakpoints, Expressions, Registers, Memory, Disassembly and Variables. Note that 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 check the **Show Debug Toolbar** option. The following screenshot identifies some useful toolbars in the **Debug** view.

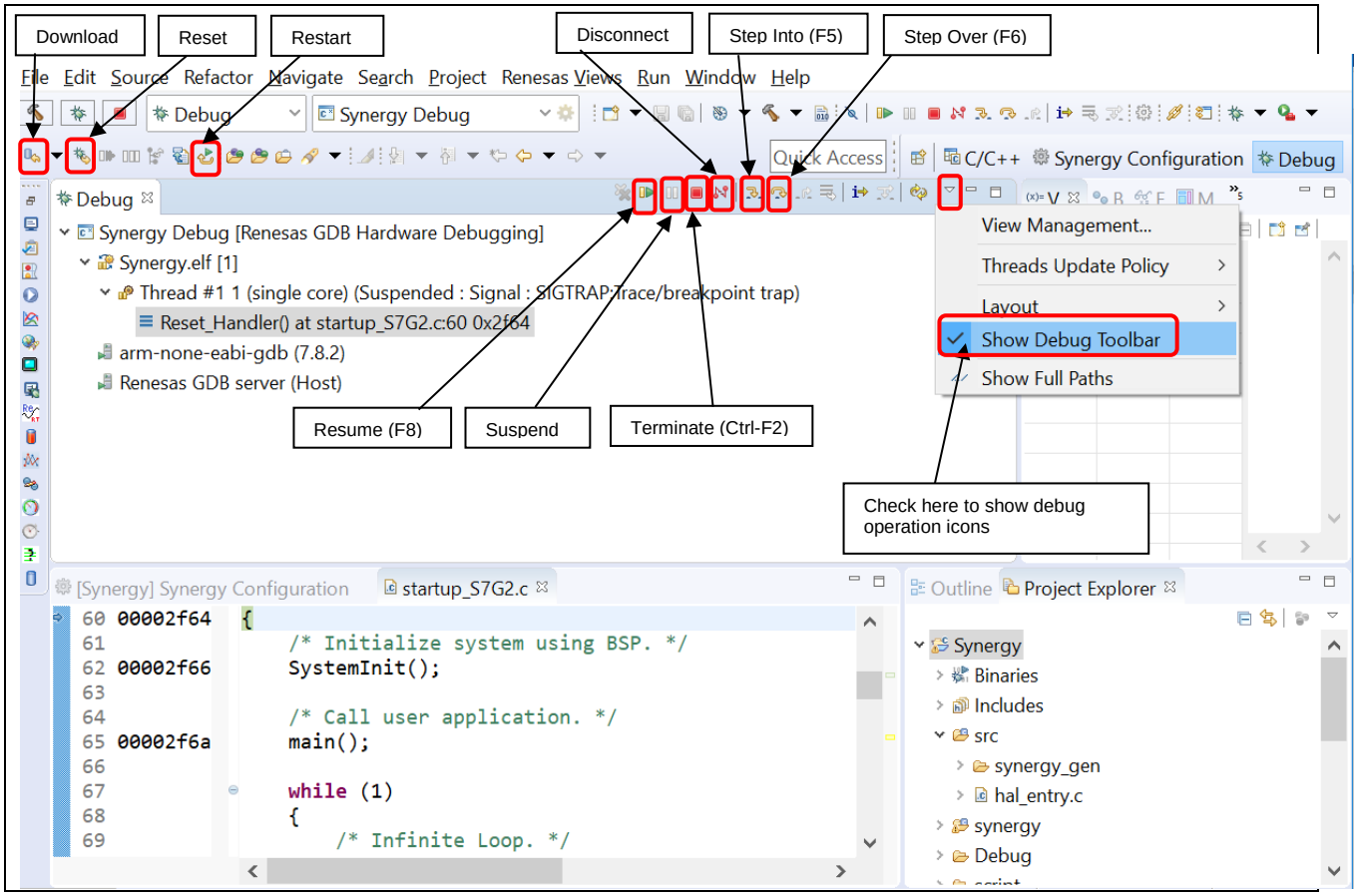


Figure 88. Debug – Useful Toolbars in Debug Views

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:

- Clicking the button or pressing **F5** can be used to step into the next method call at the current line of code being executed.
- Clicking the button or pressing **F6** can be used to step over the next method call (executing, without entering it) at the current line of code being executed.
- Clicking the button again resumes program execution.

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

The other operations available are as follows:

- Clicking the button resets and runs the program. It may stop at main() if the breakpoint is configured in the **Debug** configuration.
- Clicking the button resets the program to its entry point at the PowerOn Reset.
- The button is used for re-downloading 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.
2. For a hardware breakpoint, select **Breakpoint Types** → **e² studio Breakpoint**.
3. 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 the 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 appears in the editor pane, as well as in the Breakpoints view.

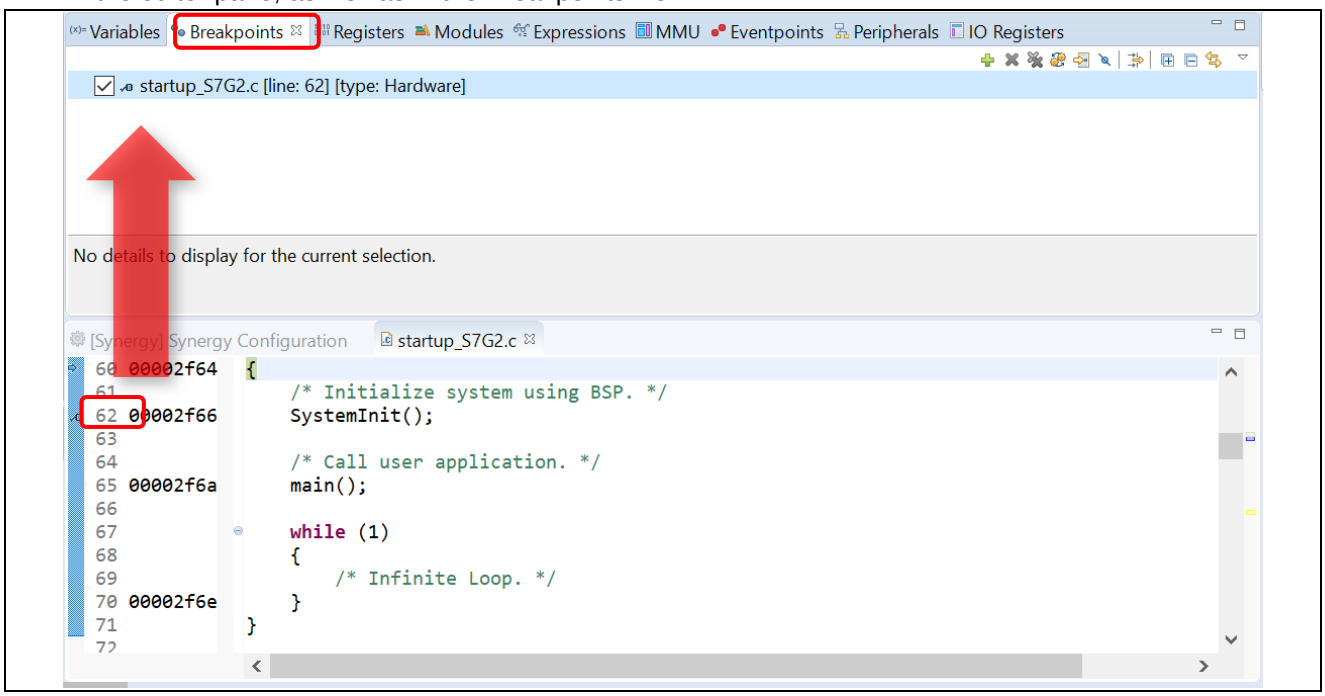


Figure 89. Debug – Breakpoints view

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 **Windows** → **Show View** → **Expressions** or 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.
3. Alternatively, right-click the variable to select the **Add Watch Expression…** menu item to add it to the **Expressions** view.

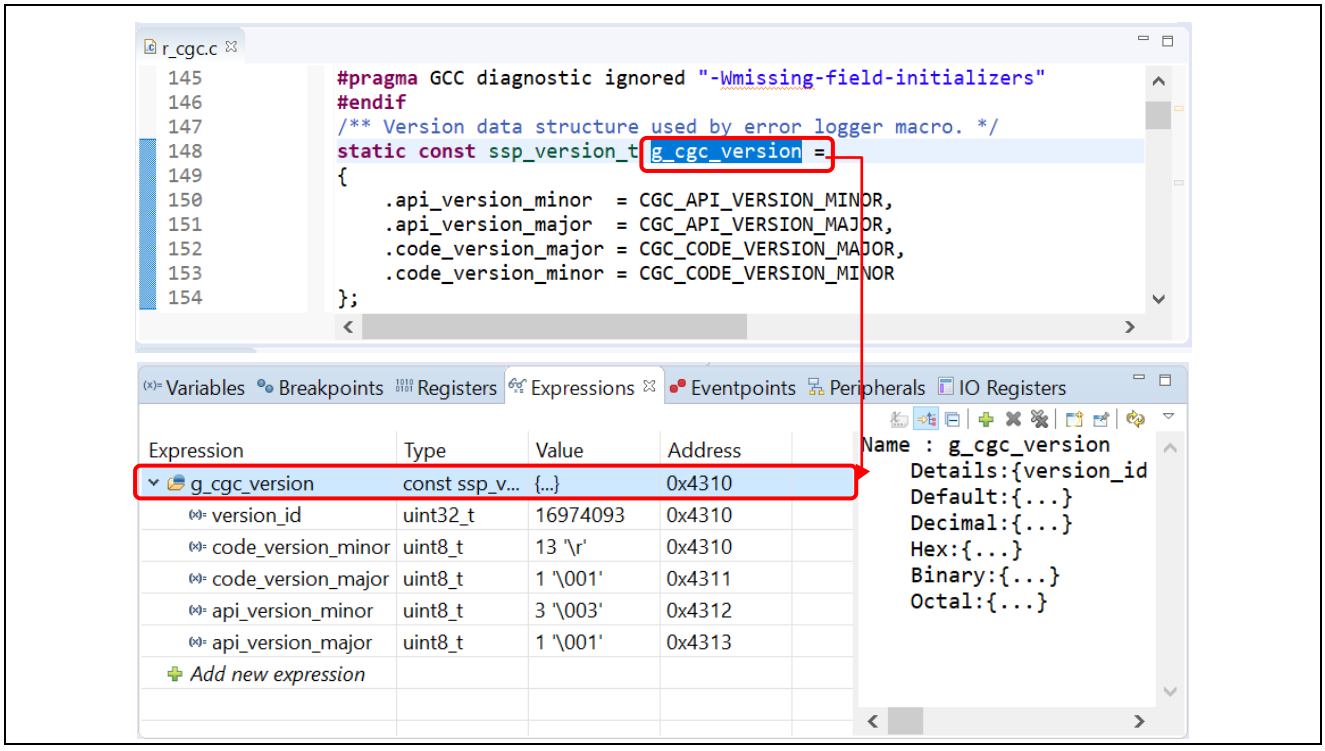


Figure 90. Debug - Expressions view

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

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

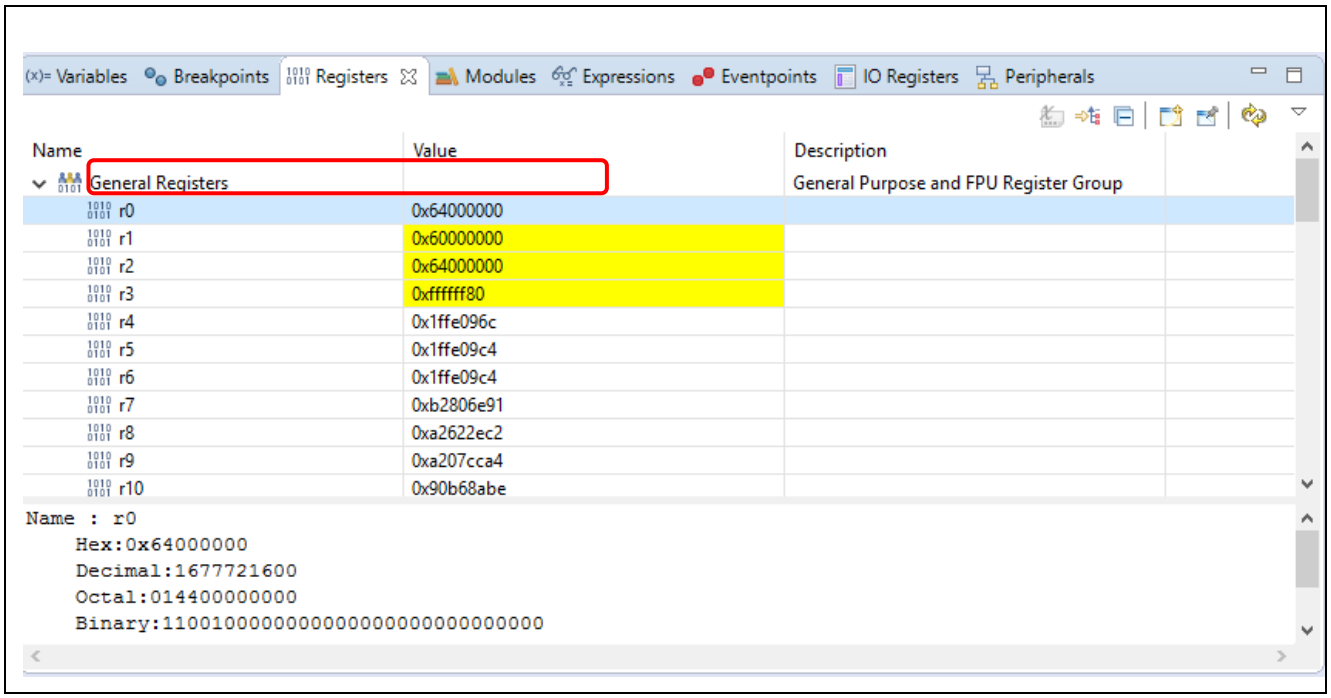


Figure 91. Debug – Registers view

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, image, and so on.).

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

1. Click **Windows** → **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`.

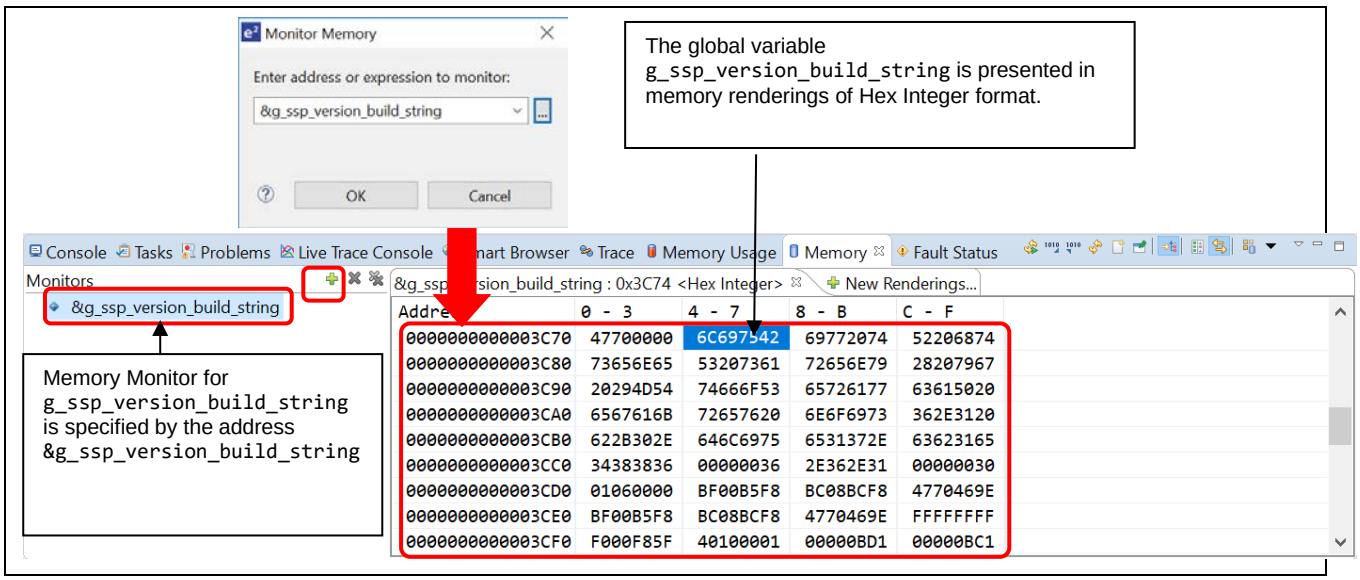


Figure 92. Debug – Memory view

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>`.

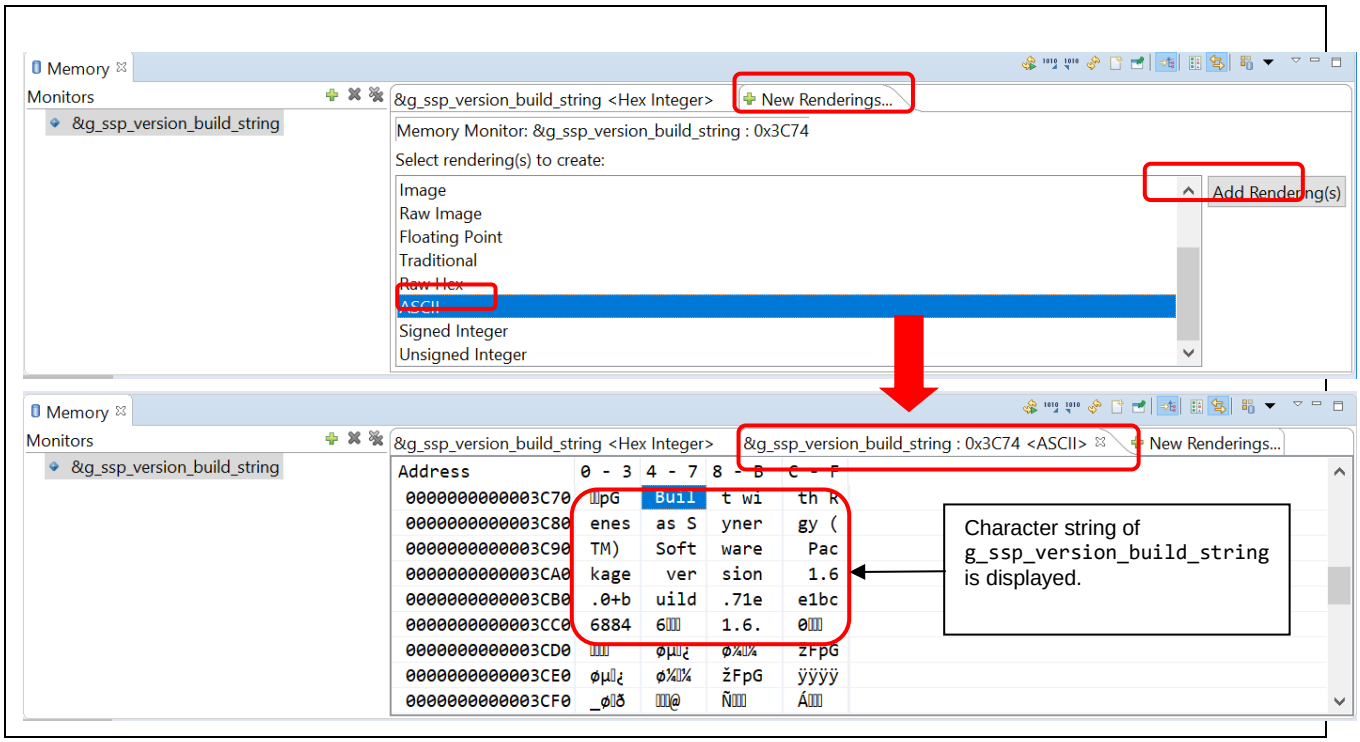


Figure 93. Debug – New Rendering In Memory view

5.3.5 Memory Usage View

Memory Usage can be used to get the information of (*.map) file or library list file (*.lbp) from project. This view lists out the total memory size, usage of ROM and RAM ratio, and detailed information of sections, objects, symbols, module, vector, and cross reference used in the project.

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

To show **Memory Usage** view, click menu **Window** → **Show view** → **Others….** In the **Show View** dialog, select **Memory Usage** and click **Open**.

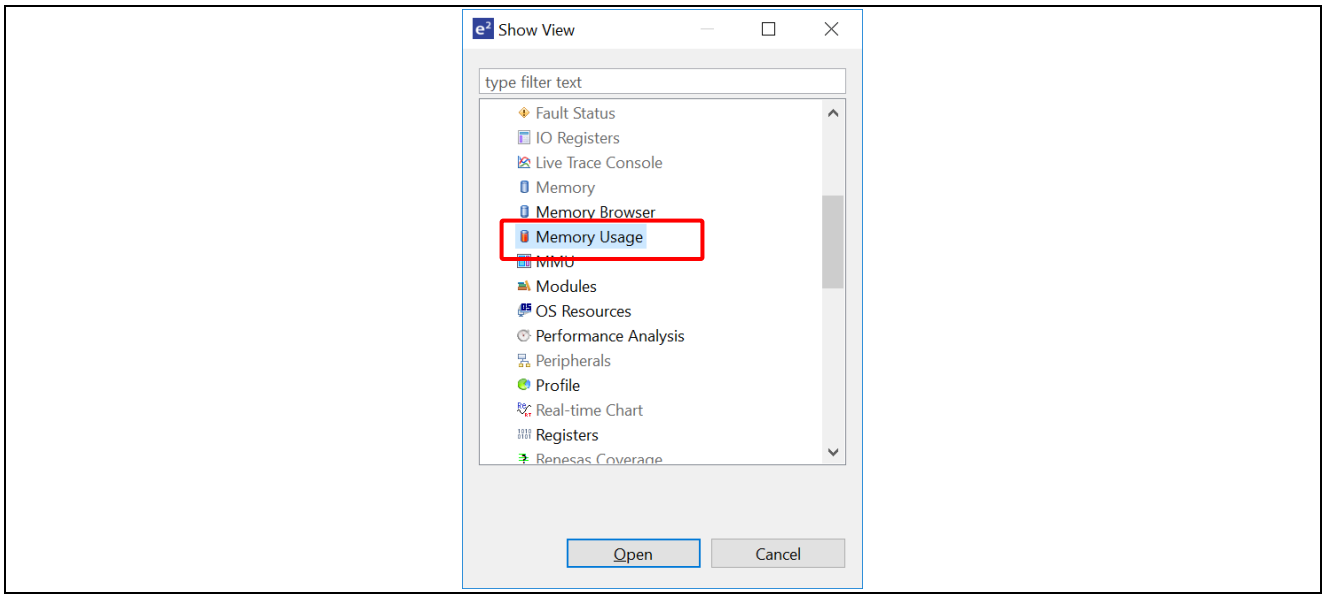


Figure 94. Show Memory Usage view

The Memory Usage view has three regions:

1. Group size region,
2. Memory Region Usage region (Device Memory Usage region is not supported yet),
3. Detail table region.

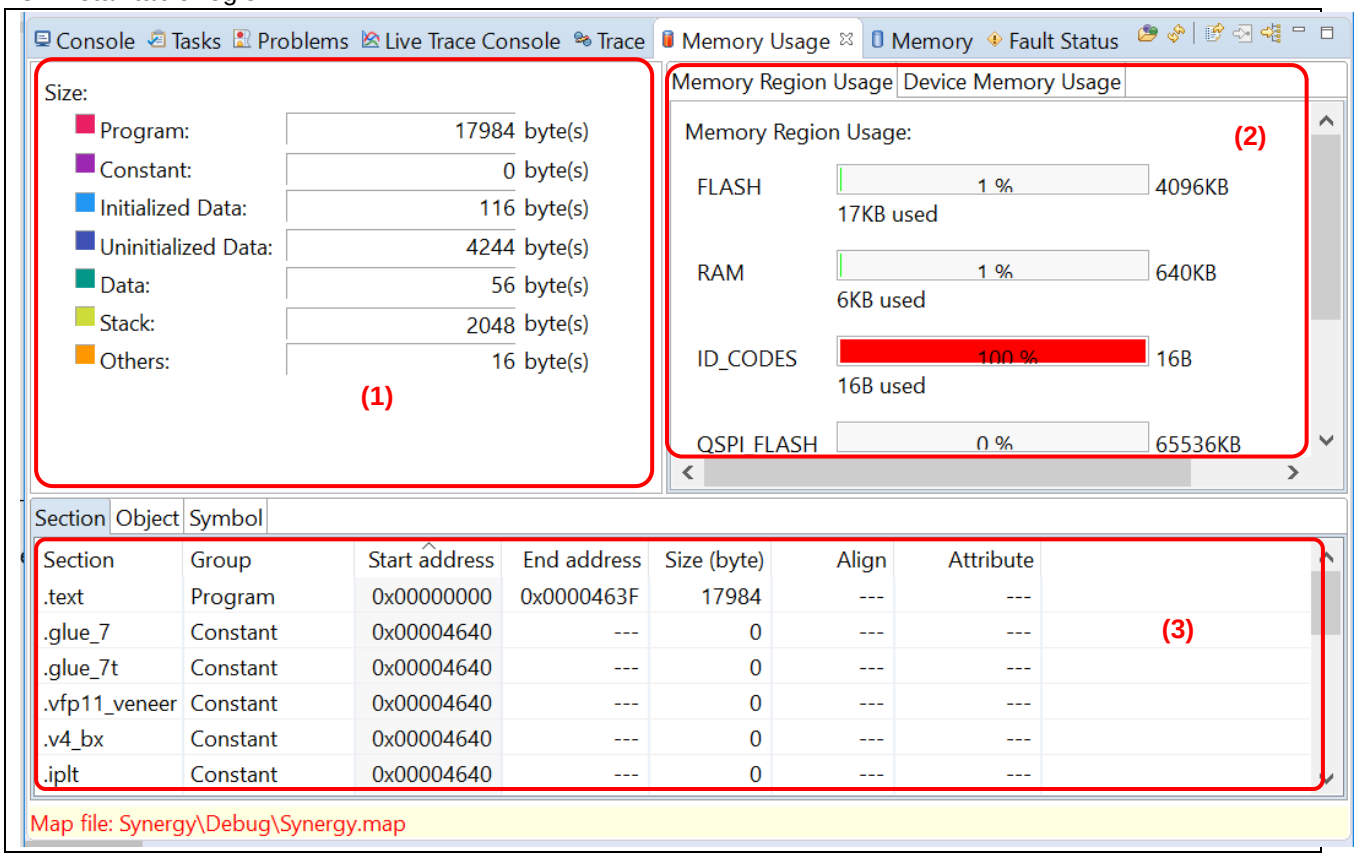


Figure 95. Regions of Memory Usage views

Memory Usage view supports the following operations:

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

: Refresh all information of Memory Usage view.

: Open *.map or *.lbp file in 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 **Windows** → **Show View** → **Disassembly** or icon to open the **Disassembly** view.
2. Click icon to enable the synchronization between assembly source and the C source (active debug context).
3. In Disassembly view, right-click at the address column to select **Show Opcodes** and **Show Function Offsets**.

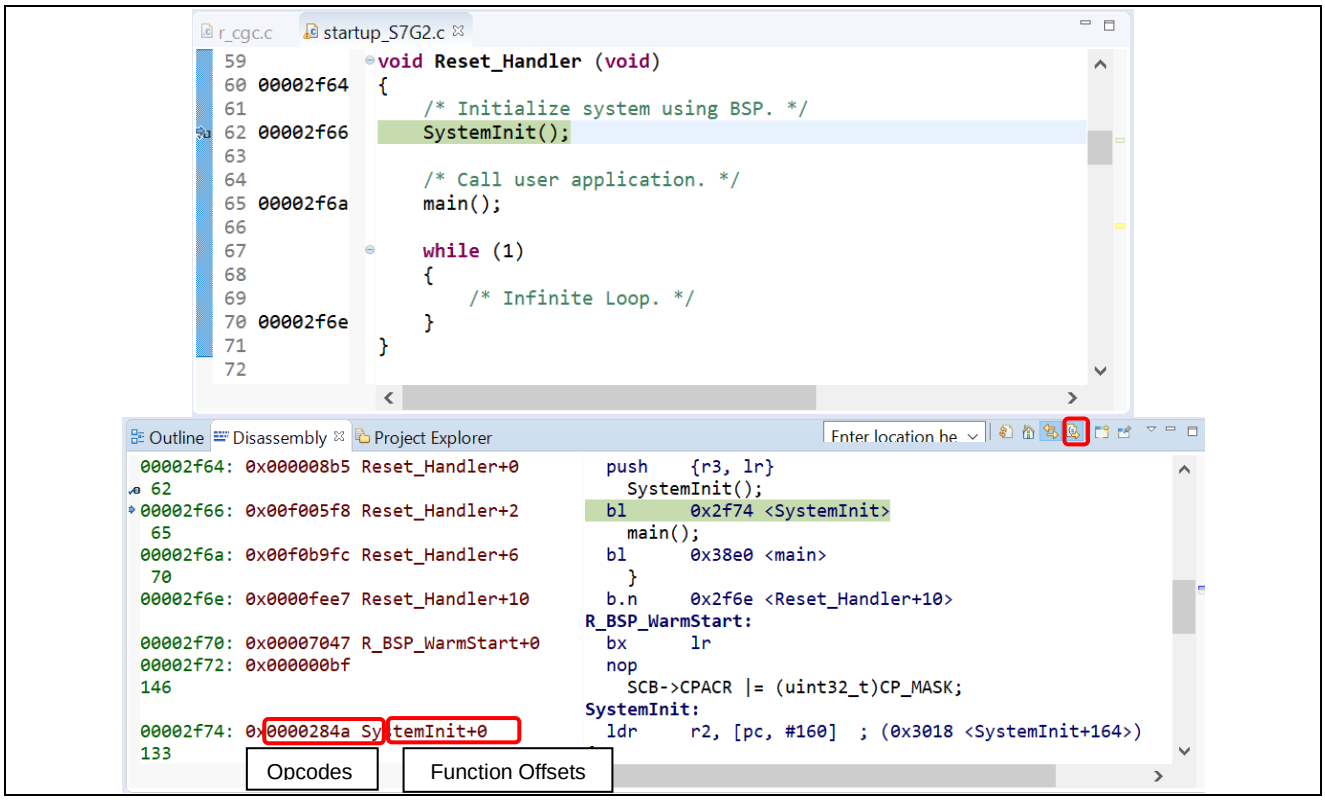


Figure 96. Debug – Disassembly view

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

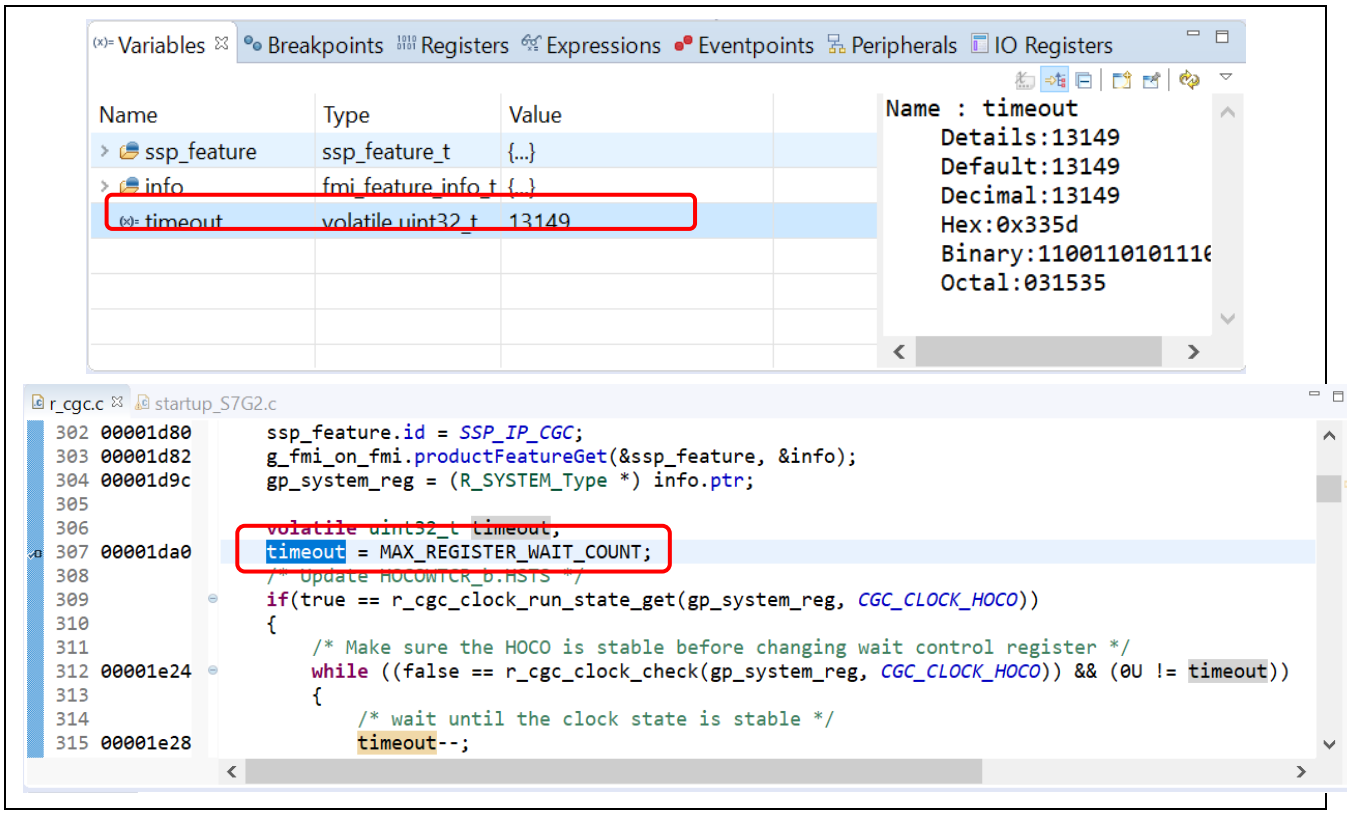


Figure 97. Debug – Variables view

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 (such as CAC) in the **IO Registers** view. Expand its IO register list.
3. Drag and drop its registers (such as 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. It is advisable to customize and view multiple selected IO registers from the **Selected Registers** pane.

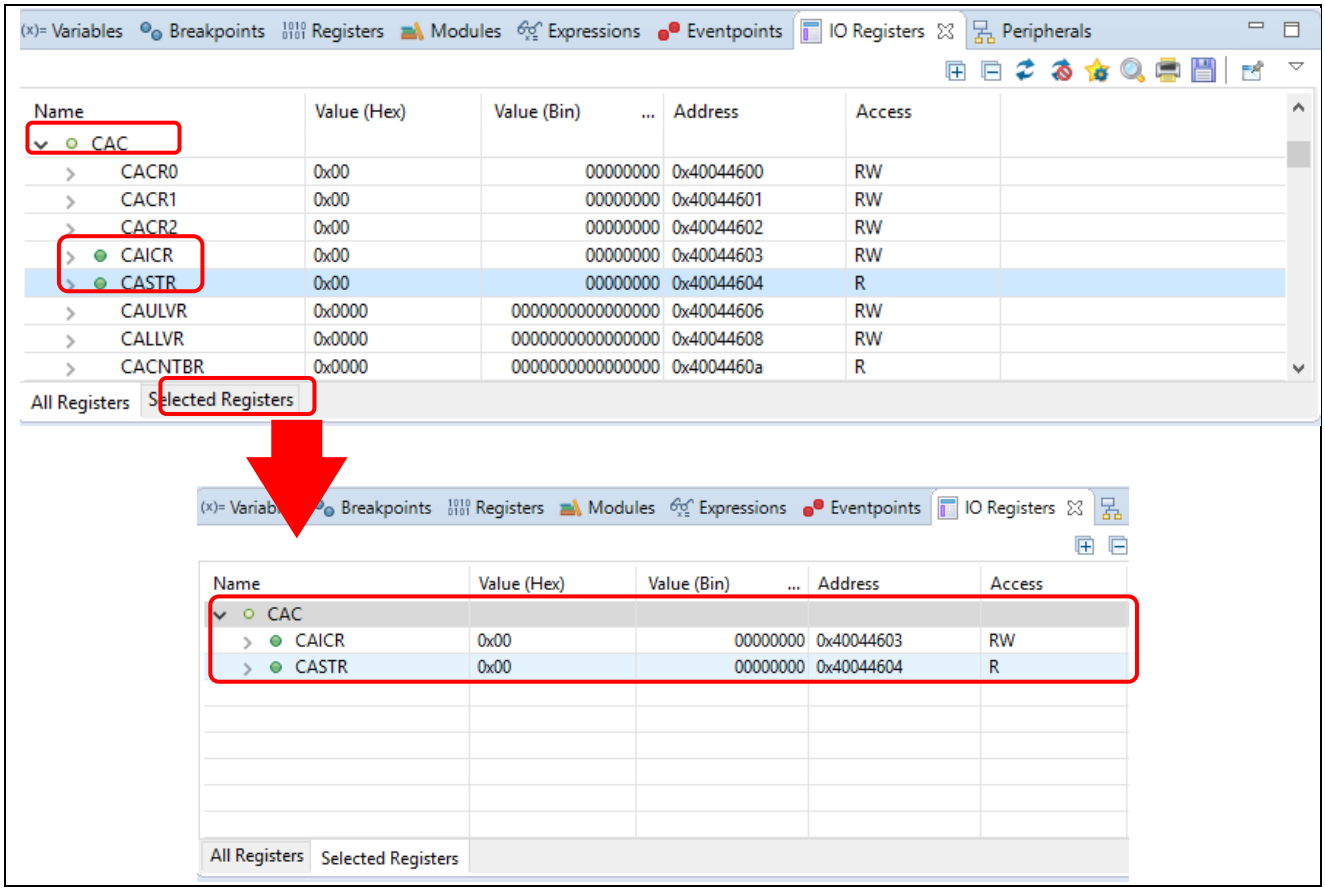


Figure 98. Debug – IO Registers view

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; for example, 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.

An event combination (for example, OR, AND (cumulative) and Sequential) can be applied to two or more events.

Table 2. Event combination explanation

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.

1. To set an event break for a global variable when an address/data is matched (for example, when `g_bsp_leds` is accessed), do the following:
   A. Click **Renesas Views** → **Debug** → **Eventpoints** or icon to open the **Eventpoints** view.
   B. Double-click the **Event Break** option to open the **Edit Event Break** dialog box.
   C. Click the **Add…** button to continue.
   The **Data Access** eventpoint type is selected by default.

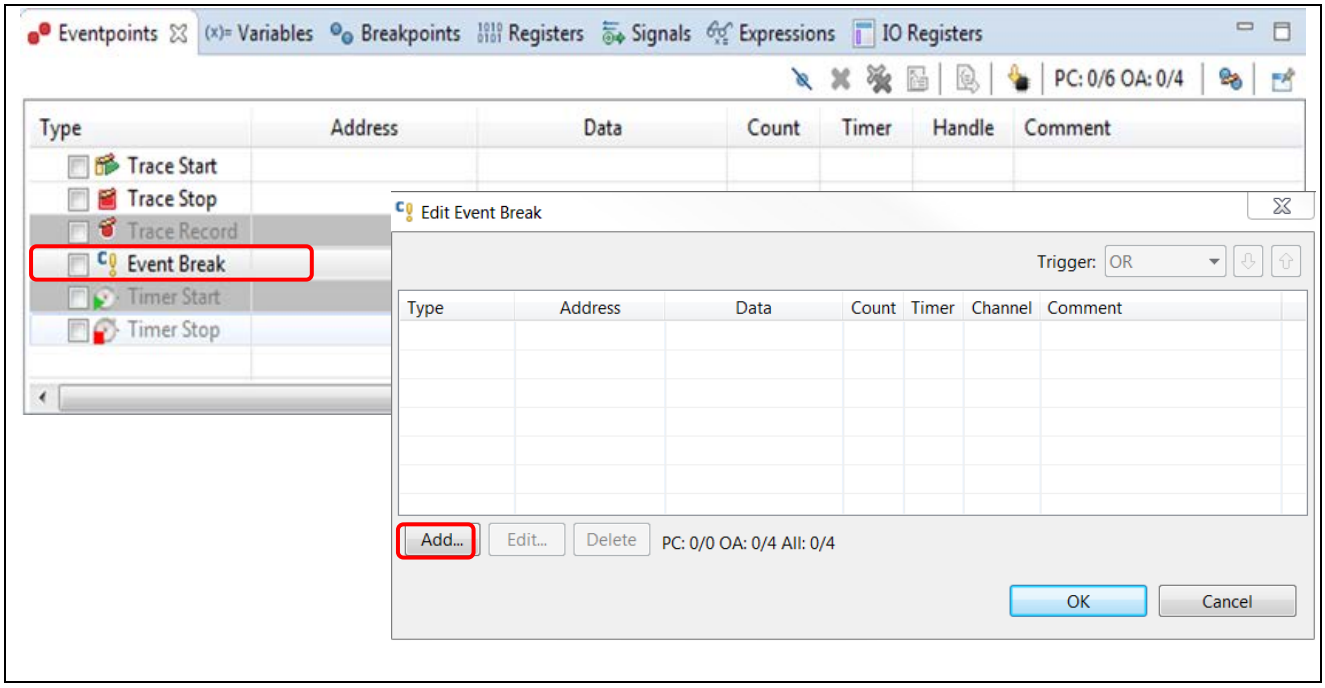


Figure 99. Debug – Eventpoints view (1/2)

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.

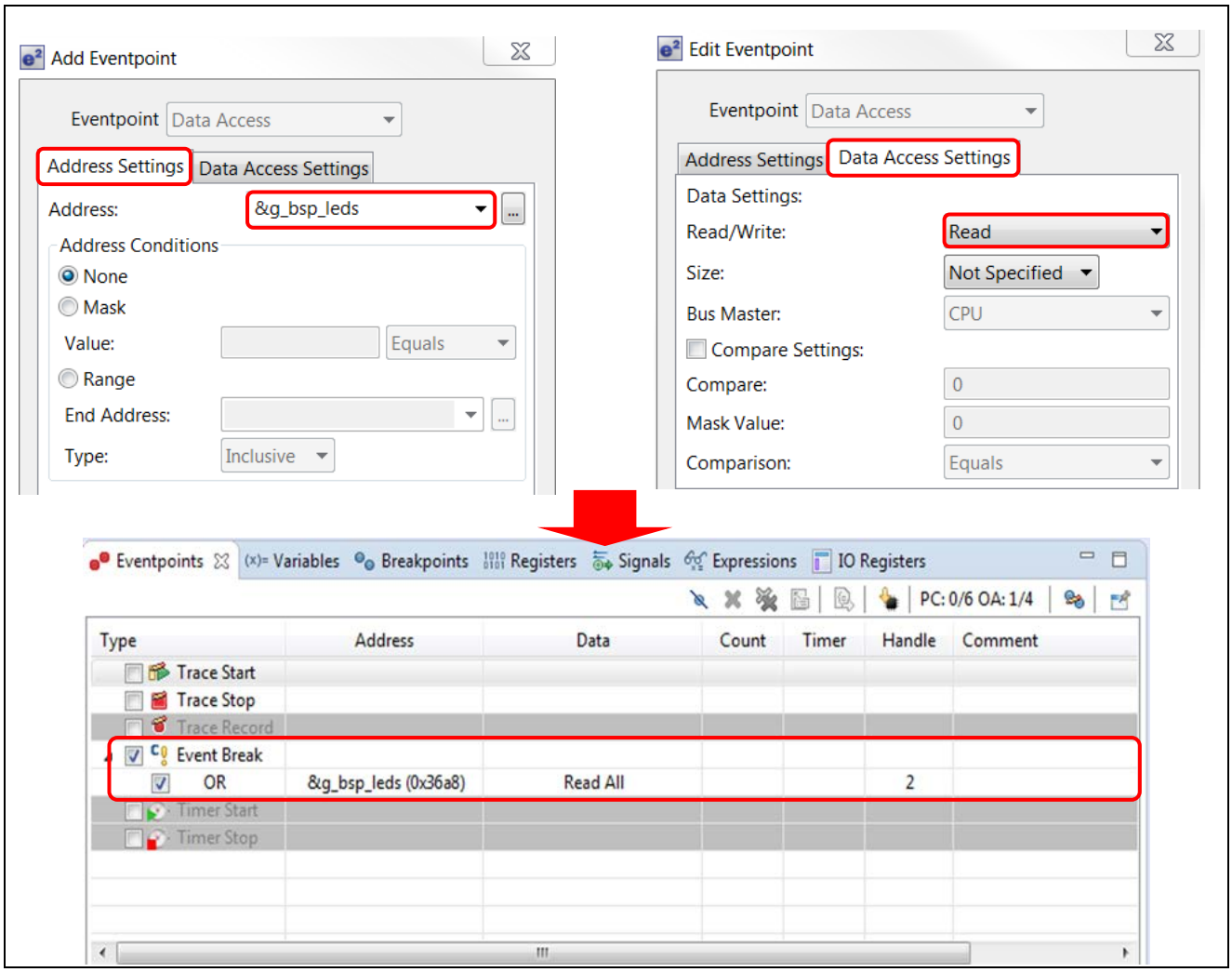


Figure 100. Debug – Eventpoints view (2/2)

5. Perform a Reset to execute the program from the start.
6. The following figure shows that when the variable `g_bsp_leds` is accessed (read), the program stops.

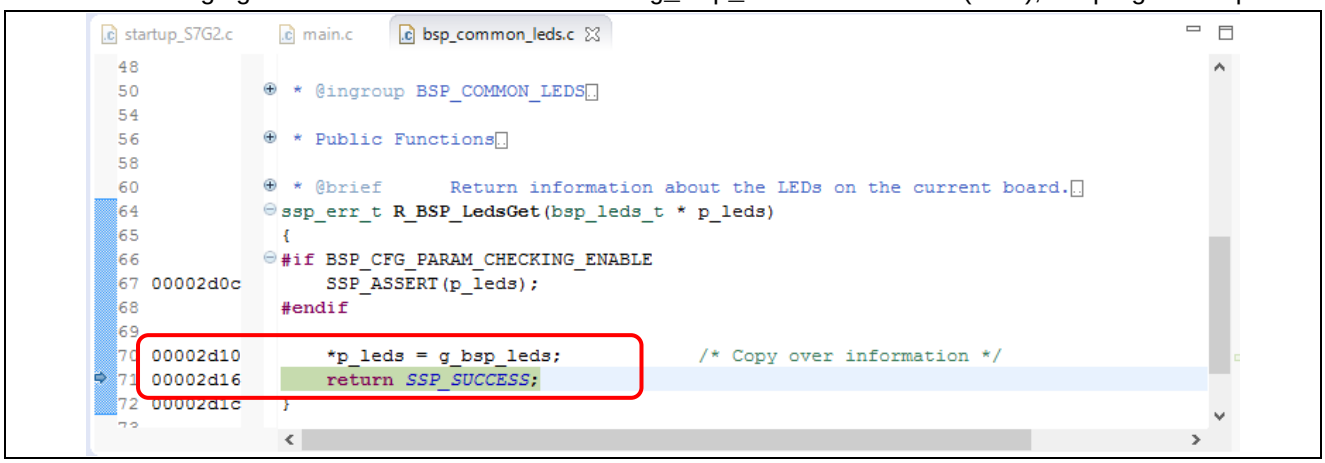


Figure 101. Debug – Execution of Event Break

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, users can do as following:

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

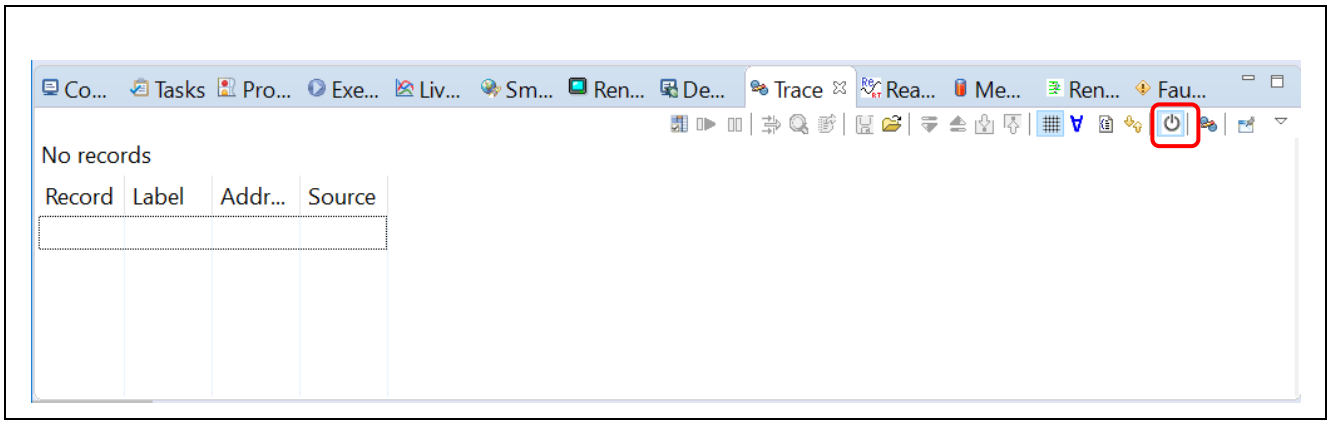


Figure 102. Debug – Turn on Trace view

3. Execute the program and stop program execution by using a breakpoint or by pressing the **Suspend** button on the Debug Toolbar. The content stored in trace memory at that point in time is displayed as trace result.
4. Select the display mode by clicking on the corresponding button.

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

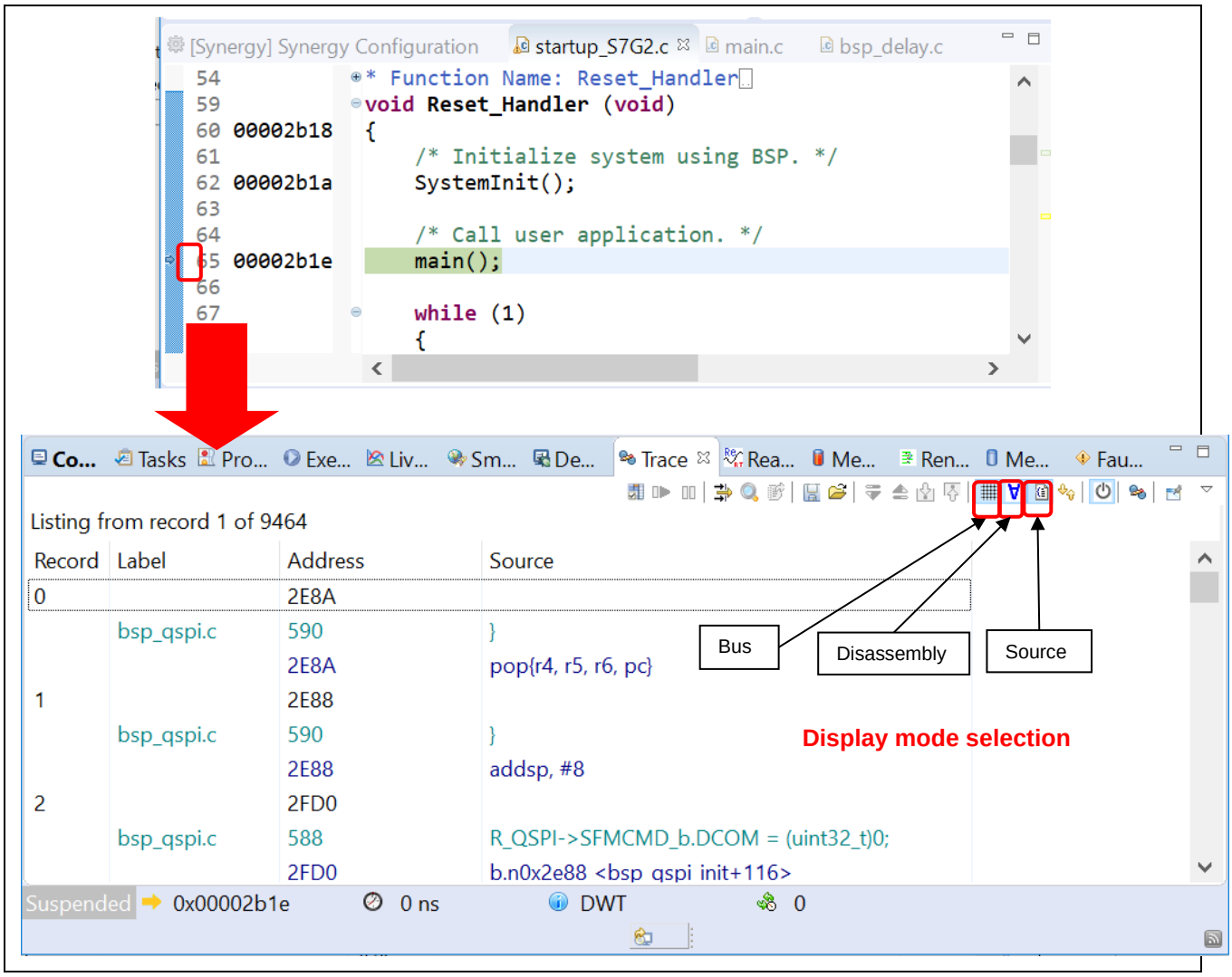


Figure 103. Debug – Select display mode in Trace view

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

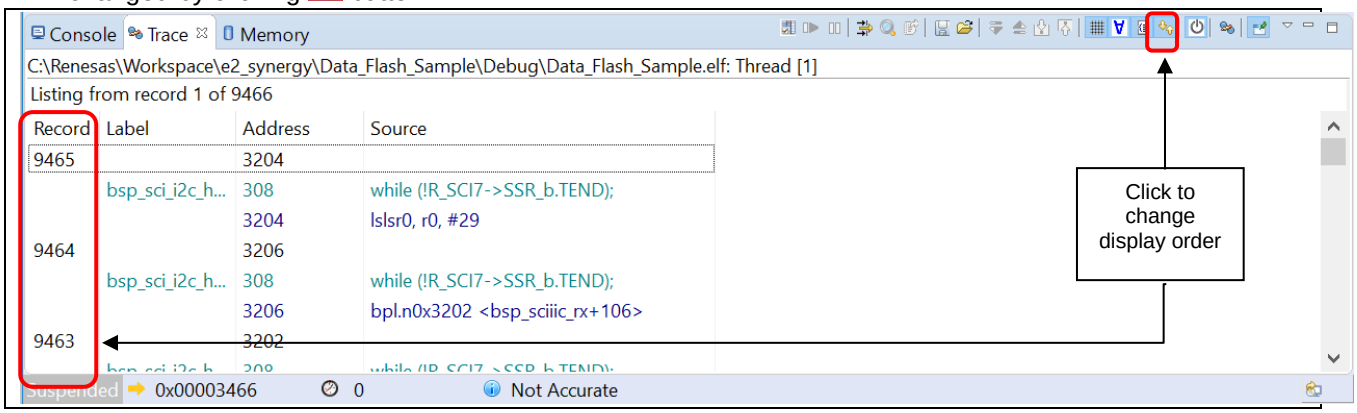


Figure 104. The display order is changed

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

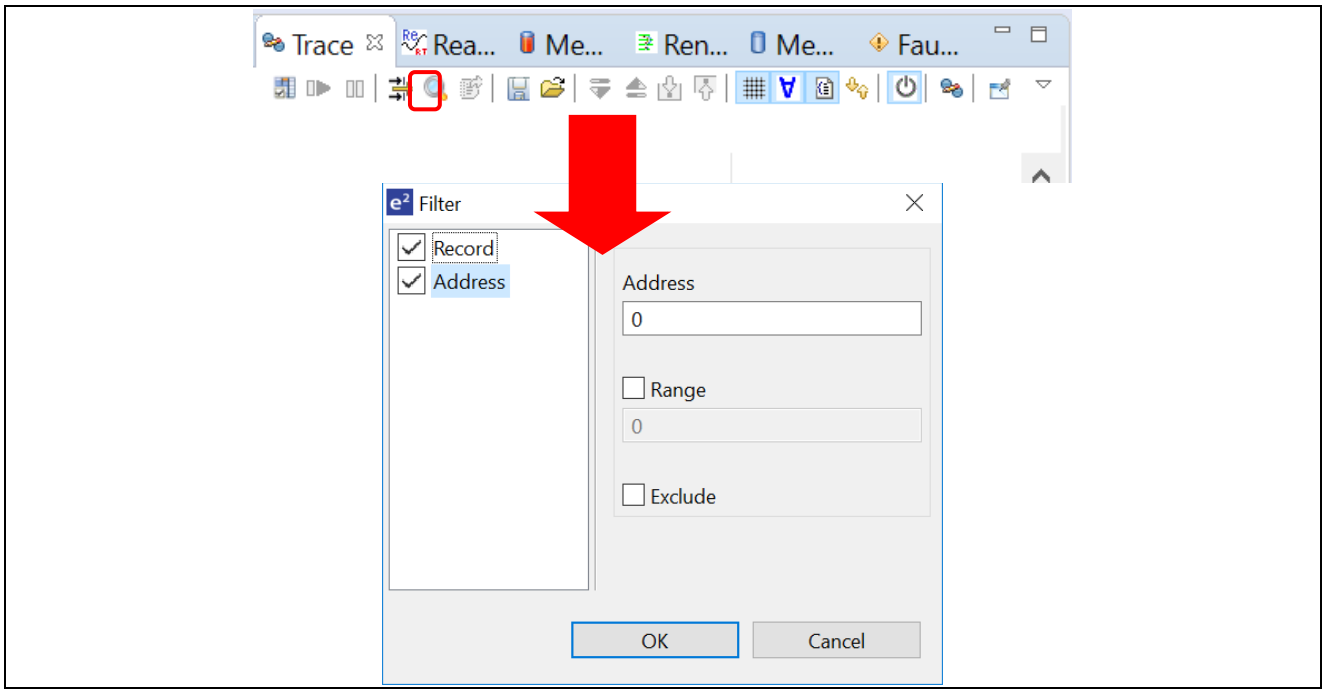


Figure 105. Debug – Filter trace result

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

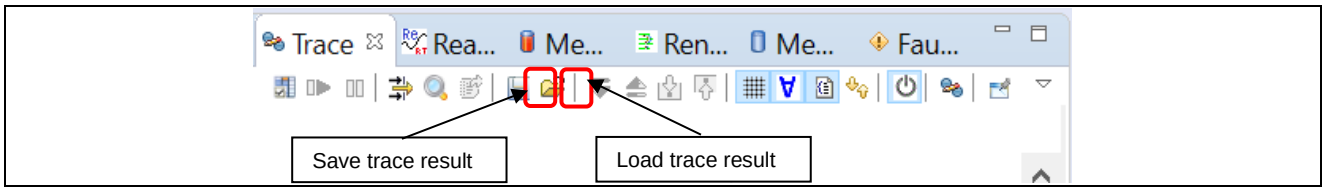


Figure 106. Debug – Save and load trace result

5.3.11 Fault Status View

This 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 following figure. This function is available in e² studio v5.2 and above.

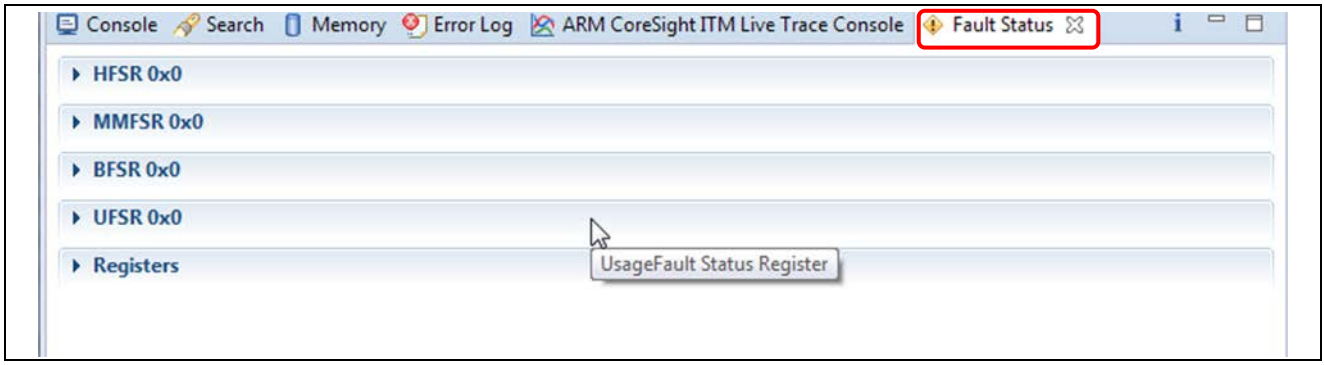


Figure 107. Fault Status No hardware fault

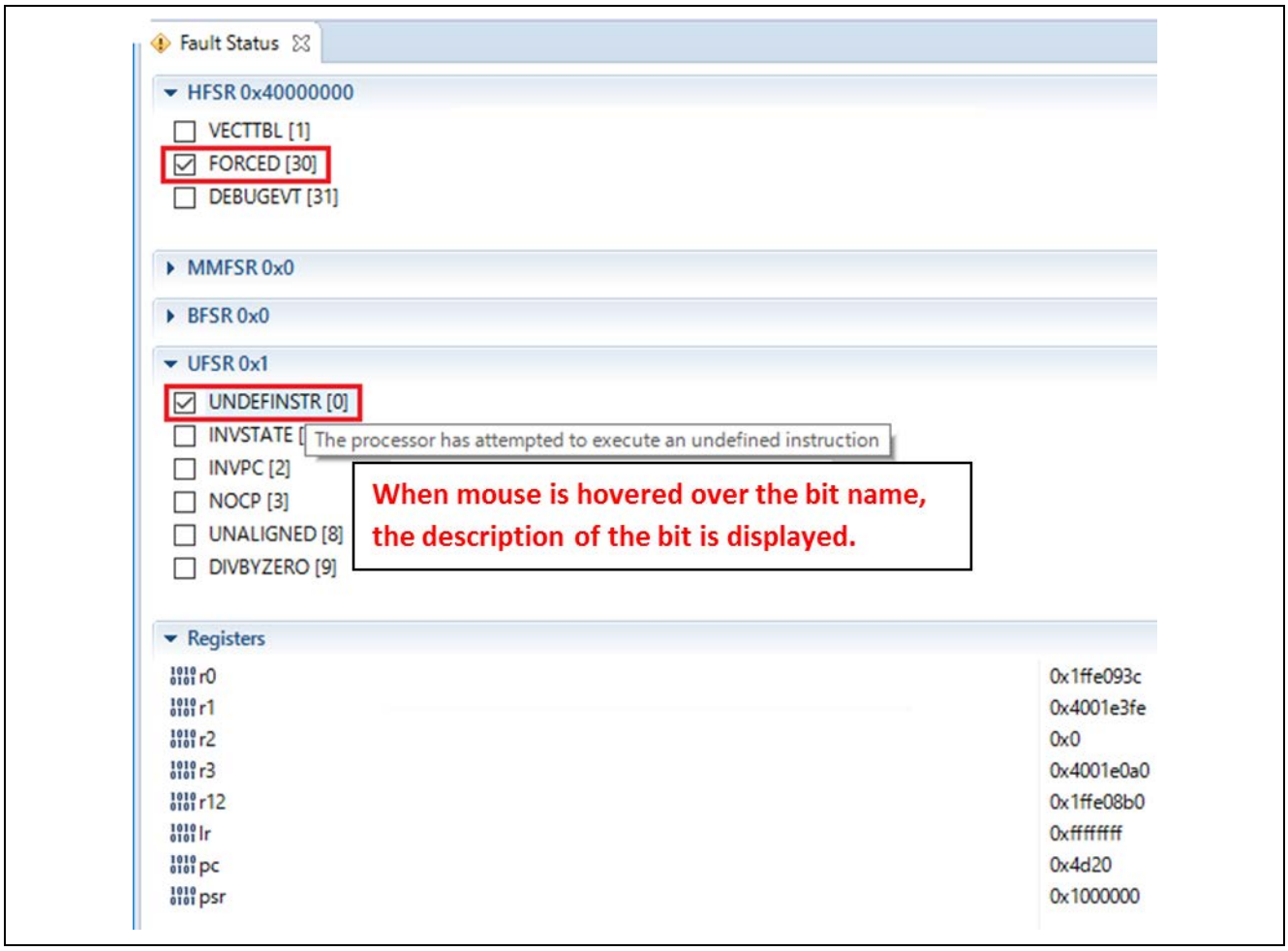


Figure 108. Fault Status Hardware fault occurred

5.3.12 Run Break Timer

The Run Break Timer feature allows 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.

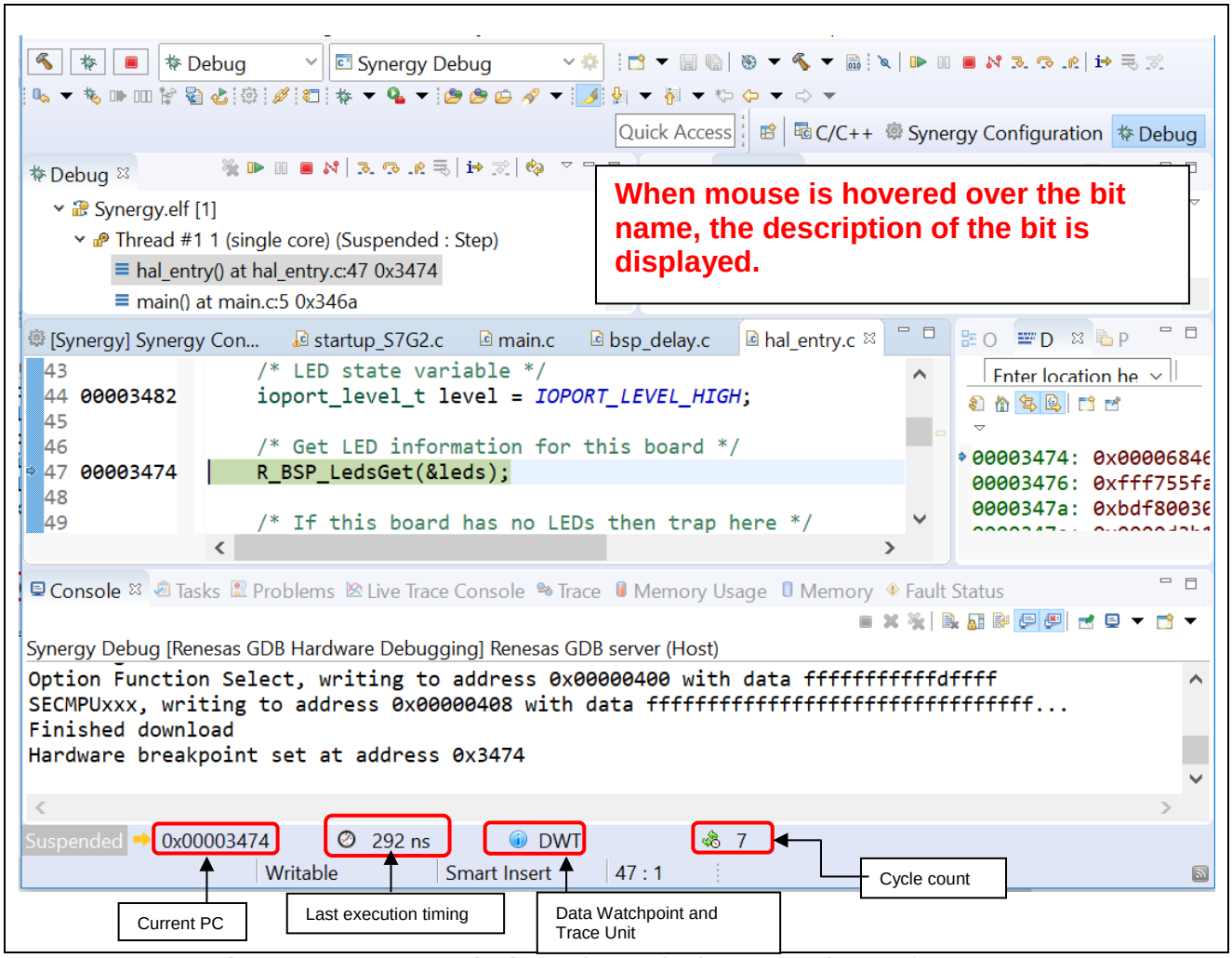


Figure 109. Run Break Timer shows the last execution performance

The following table shows the support of Run Break Timer feature available for various Synergy devices.

Table 3. 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, visit www.renesas.com/synergy/e2studio for the e² studio release note.

6. Setting up a ThreadX Application

This example demonstrates how to generate and build a Synergy project to include ThreadX objects and the 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, the LEDs are blinked 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 this semaphore every 1 second so that thread can resume.

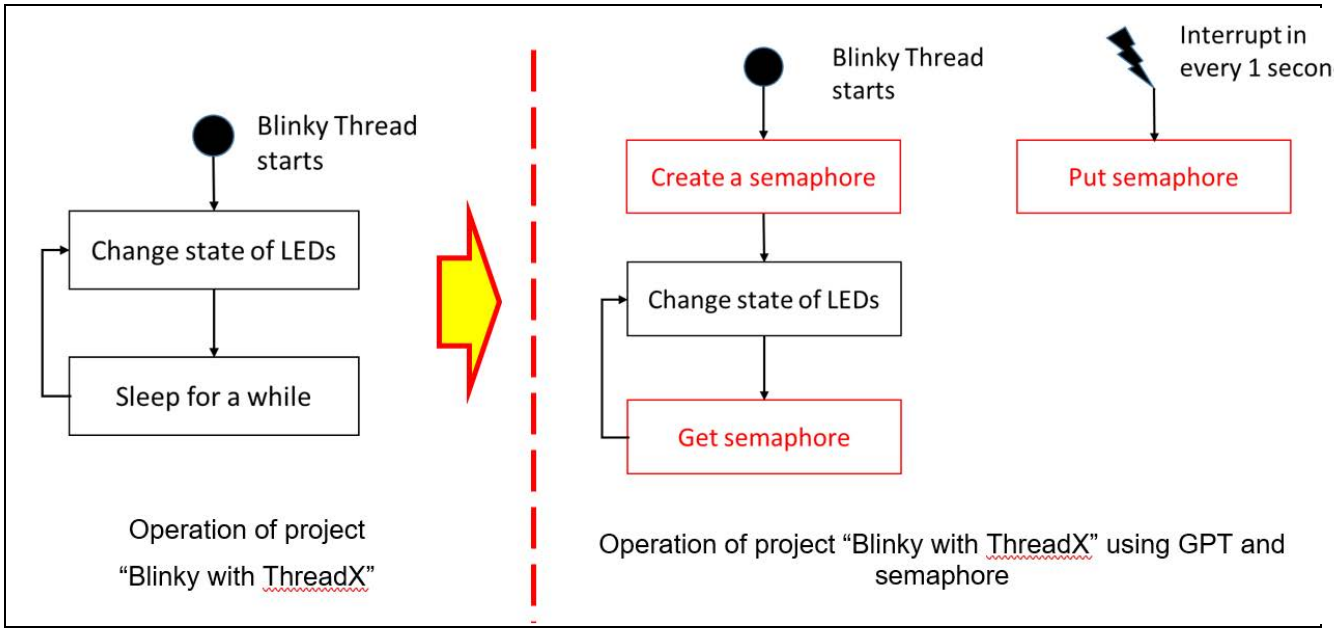


Figure 110. Setting up a ThreadX Application – Introduction

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 section 3.1,Generating a New Synergy Project to generate a new project. However, in the last dialog (the Project Template dialog), select **Blinky with ThreadX**.

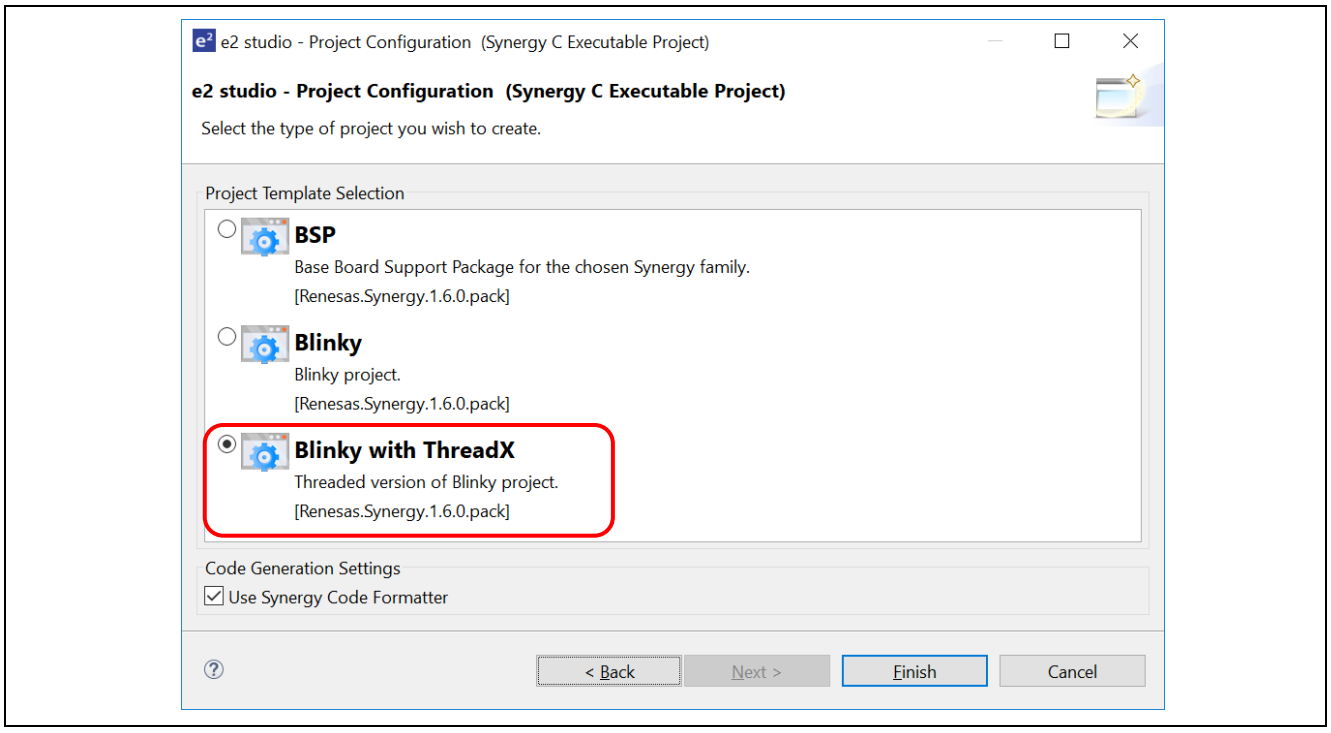


Figure 111. Setting up a ThreadX Application – Blinky with ThreadX template selection

2. Open the **Threads Configuration** page in the **Synergy Project Configuration**. See section 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.

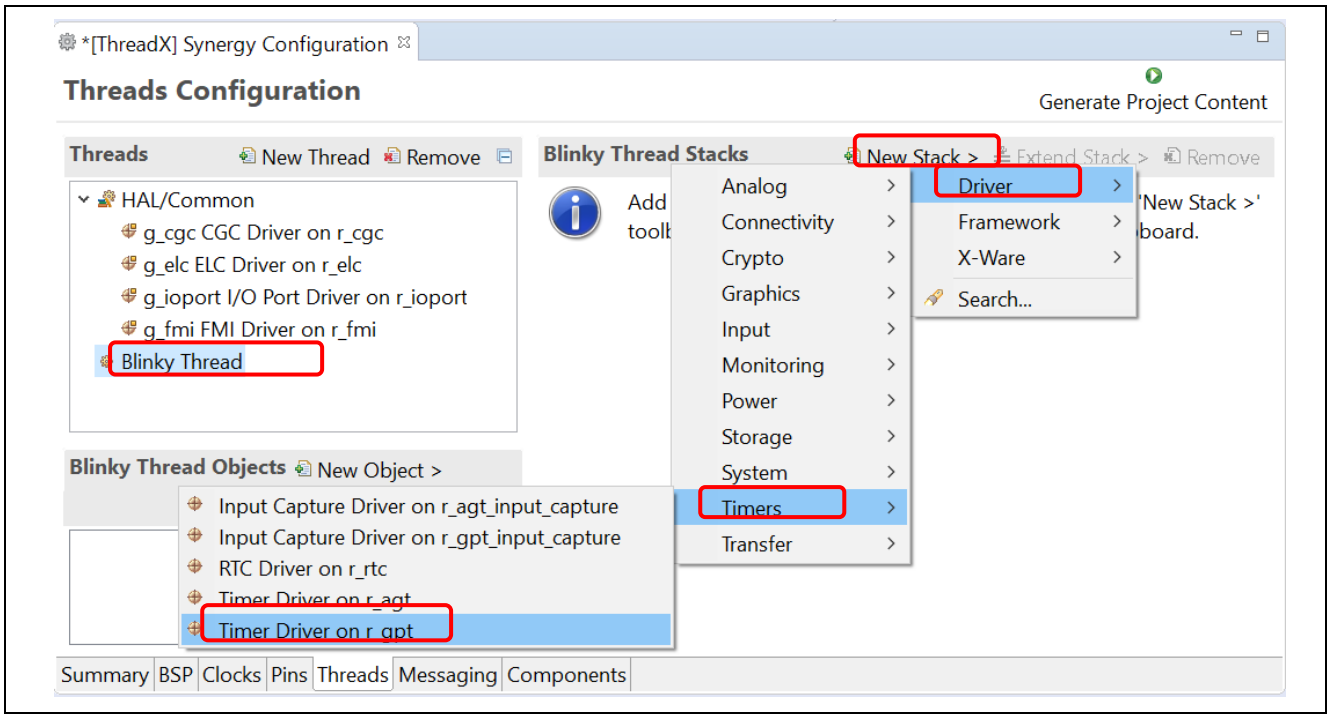


Figure 112. Setting up a ThreadX Application – Adding the GPT module

4. Configure the GPT module as follows.
   - Name: g_timer
   - Mode: Periodic
   - Period Value: 1
   - Period Unit: Seconds
   - Callback: gpt_callback
   - Overflow Interrupt priority: 2

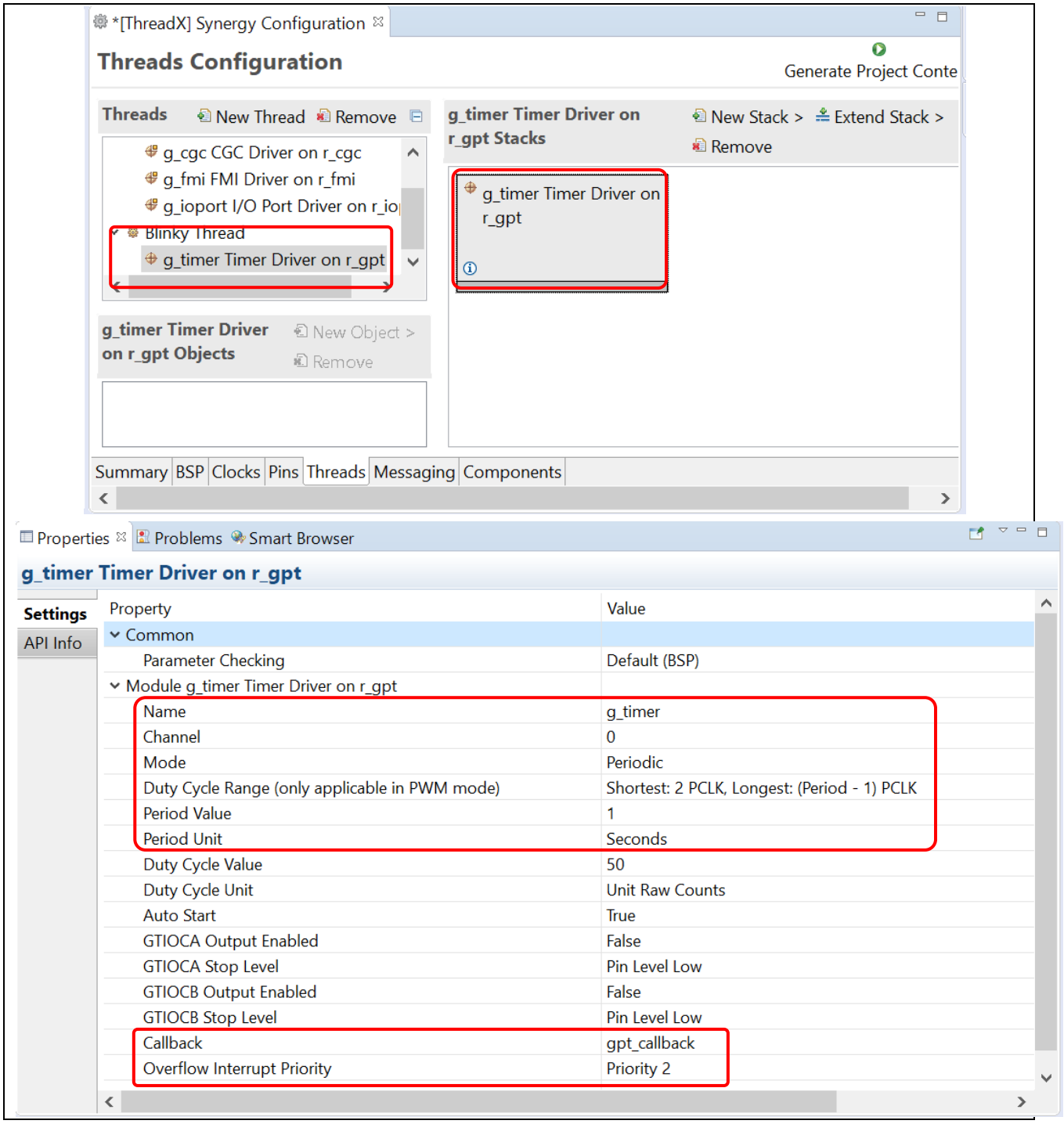


Figure 113. Setting up a ThreadX Application – GPT module configuration

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

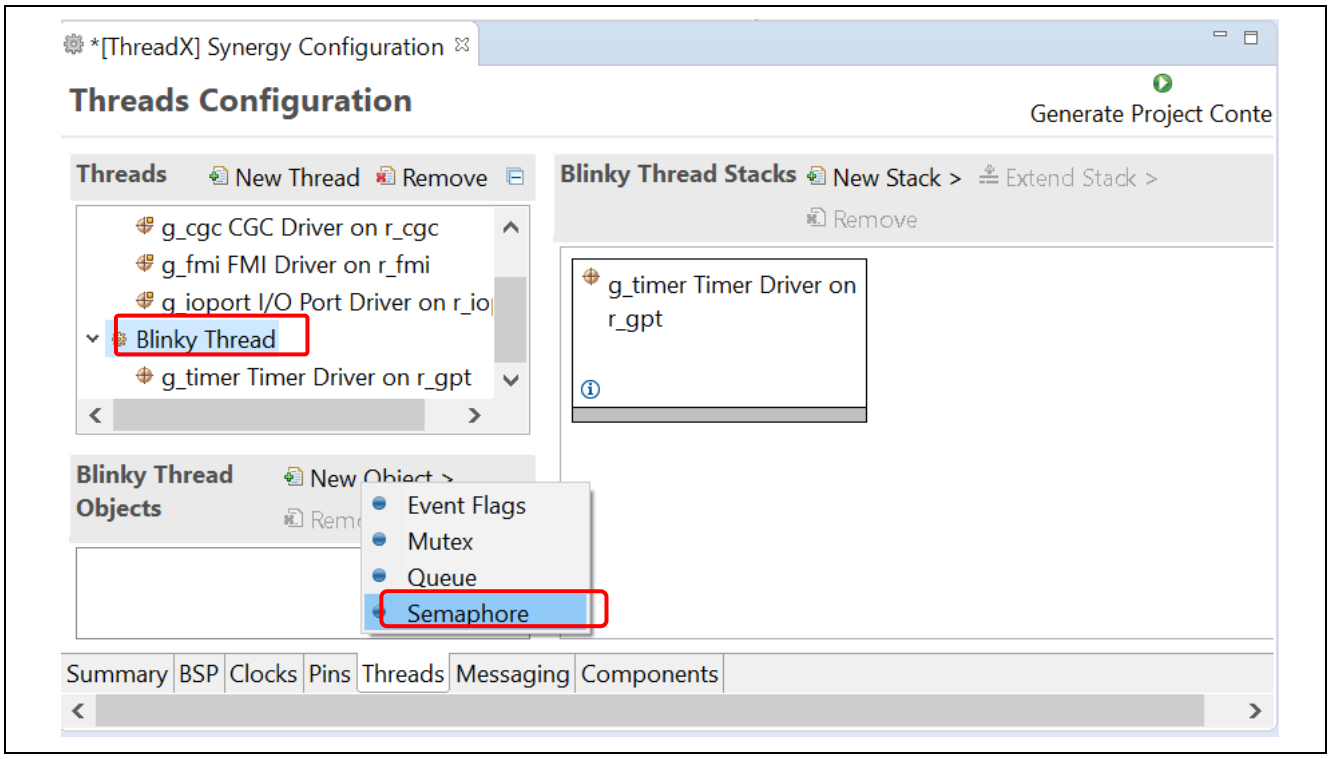


Figure 114. Setting up a ThreadX Application – Adding a Semaphore Object

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

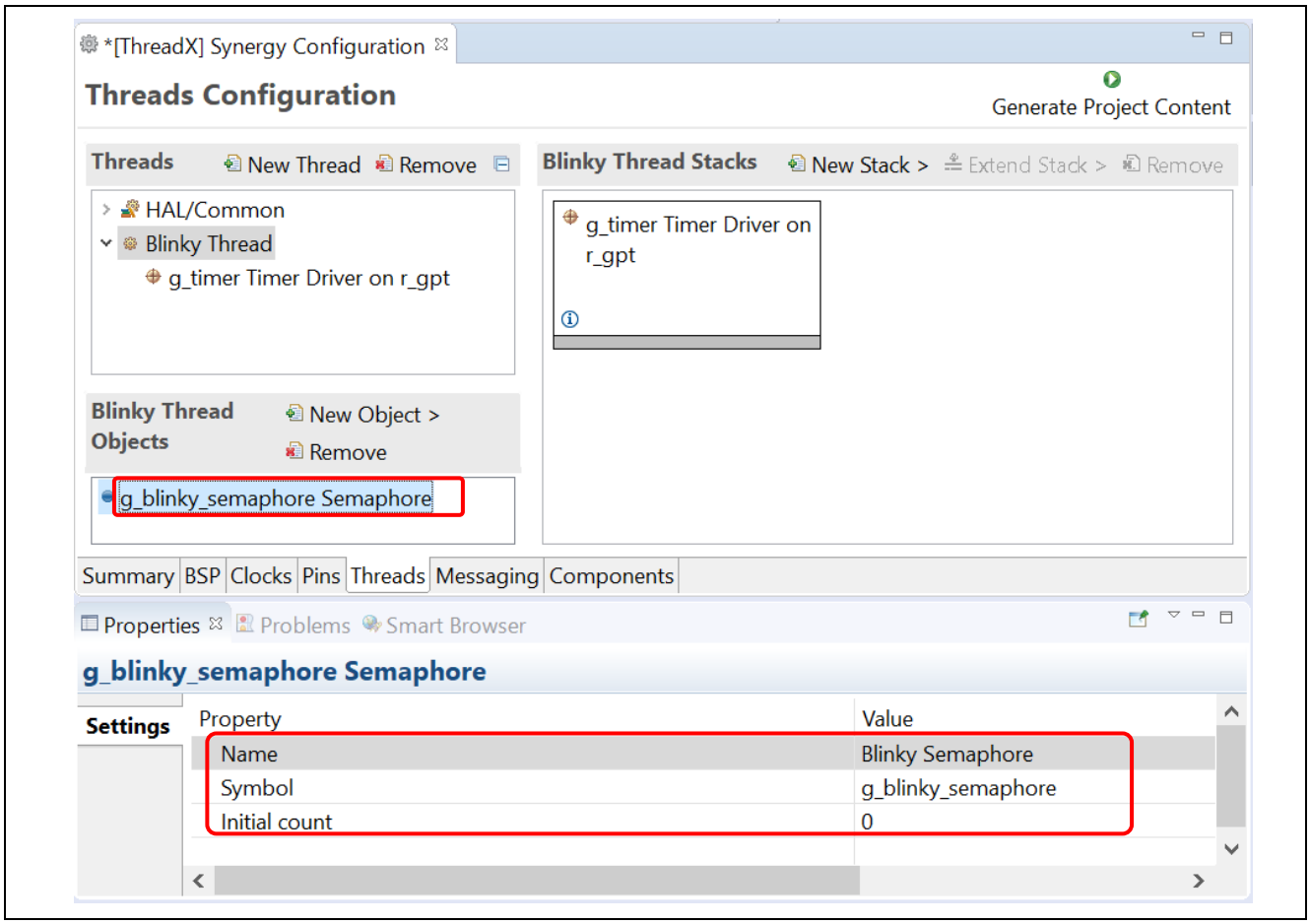


Figure 115. Setting up a ThreadX Application – Semaphore Object Configuration

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

7. 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);
     }
     ```

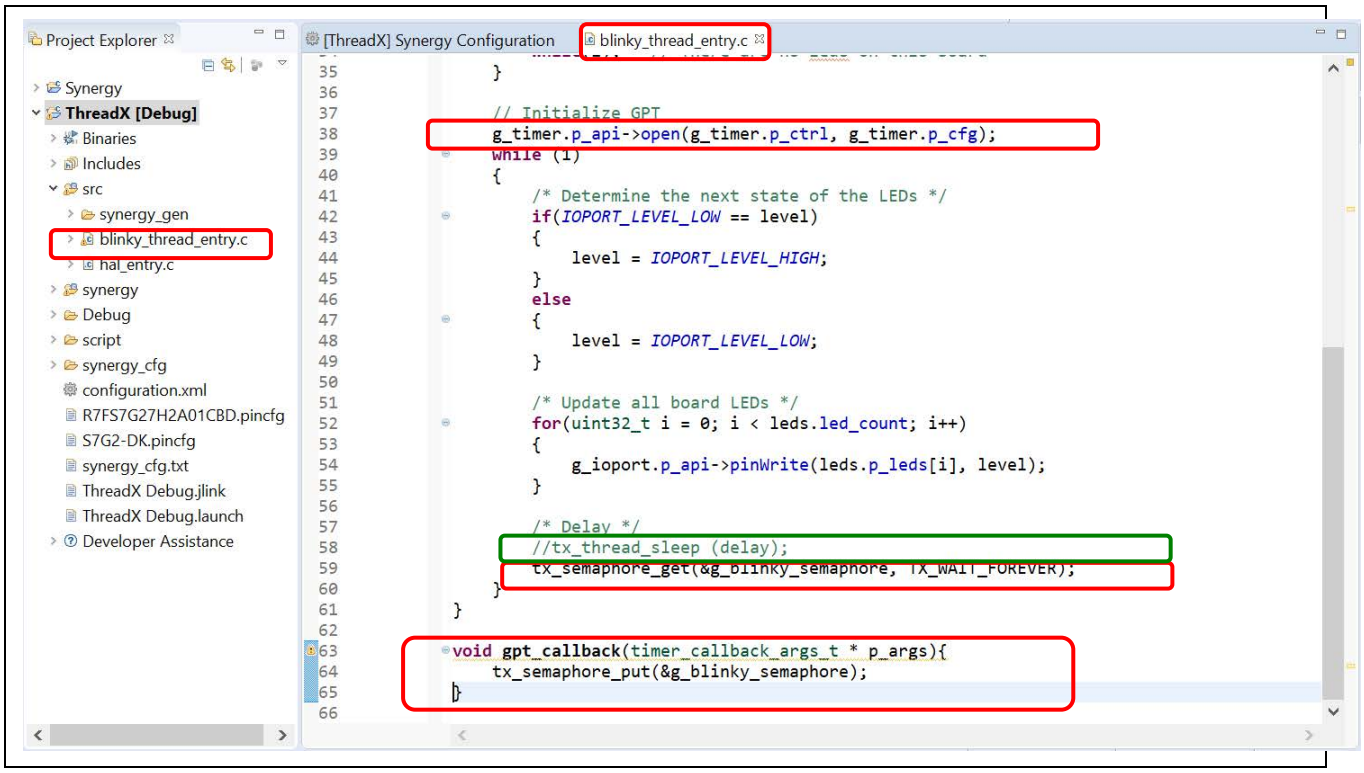


Figure 116. Setting up a ThreadX Application – Adding User Source Code

8. Build and run the project on the Synergy DK-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 the Help tab to open the Help menu.

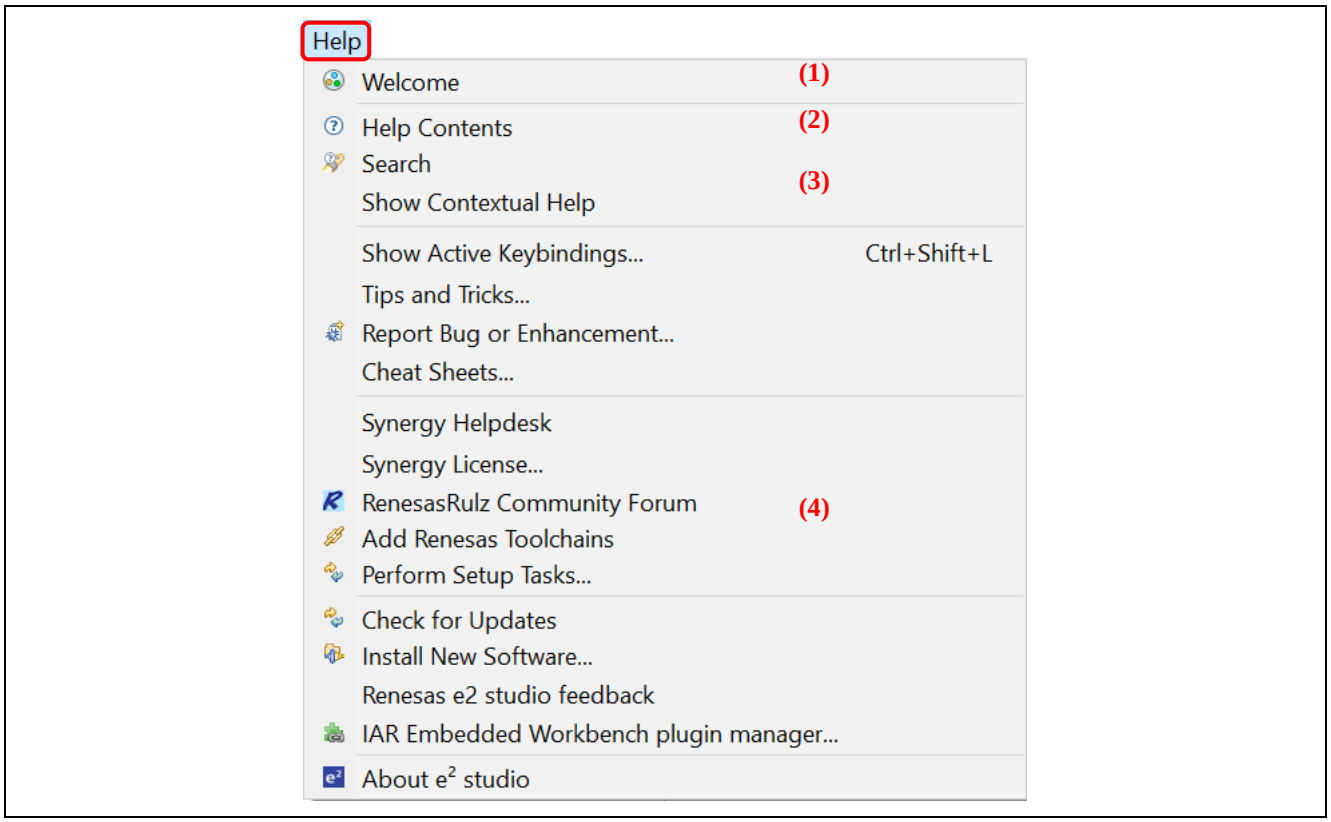


Figure 117. Help – Help Menu

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

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, and so on.
  It can be launched by clicking on the **Help** menu → **Help Contents** → **e² studio Debug Help**.
- The **Synergy Contents** topic provides information about Synergy project creation, using the Synergy Configuration Editor and FAQs.

  This topic can be launched by clicking on the **Help** menu → **Help Contents** → **Synergy Contents**.

Website and Support

Visit the following vanity URLs to learn about key elements of the Synergy Platform, download components and related documentation, and get support.

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb.13.18	—	Initial document release for e2 studio v5.2 or higher
1.01	Jan.09.19	—	Updated for e2 studio v6.2 or higher
1.20	Aug.08.19	—	Updated for e² studio v7.3.0 and SSP v1.6.0

e2 studio Getting Started Guide

Publication Date: Aug.08.19

Published by: Renesas Electronics Corporation

Renesas Synergy™ Platform
e² studio Getting Started Guide

Renesas Electronics Corporation

R20UT4204EU0120