Renesas Synergy™ Platform

Renesas Synergy™ Project Import Guide

Introduction

This application note describes how to import a Renesas Synergy Project into the IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy) or e² studio integrated solutions development environment (ISDE), and then build and run the project application. The procedure in this application note applies to all Renesas Synergy devices and development boards, and all software listed in the "Required Resources" section.

Note: The NetX™ DNS project is used an example in this application note. Substitute your desired project, as needed.

Goals and Objectives

The goal of this application note is to help you import an existing Renesas Synergy Project, such as an example application, into IAR EW for Synergy or e² studio.

Prerequisites

As the reader of this application note, you are assumed to have some experience with the IAR EW for Synergy or Renesas e² studio ISDE and the SSP. For example, before you perform the procedure in this application note, you should follow the procedure in your board's Quick Start Guide to build and run the Blinky project. By doing so, you will become familiar with IAR EW for Synergy or e² studio and the SSP, and ensure that the debug connection to your board is functioning properly.

Required Resources

To perform the procedure to import a project, you will need a PC running Microsoft® Windows® 7 or 10 with the following Renesas software installed:

- A Renesas Synergy development board (for example, DK-S7G2)
- A PC running Microsoft® Windows® 7 or 10 with the following Renesas software installed:
    — IAR EW for Synergy v8.21.1 or greater or e² studio ISDE v6.2.0 or greater
    — Synergy Software Package (SSP) v1.4.0 or greater
    — Renesas Synergy™ Standalone Configurator (SSC) v6.2.0 or greater (only for IAR EW for Synergy).

You can download the required Renesas software from the Renesas Synergy Gallery (https://synergygallery.renesas.com).

Contents

## 1. Importing and Building Projects with IAR EW for Synergy

1.1 Importing an Existing Project into IAR EW for Synergy

1. Start by opening IAR EW for Synergy.
2. Unzip the example project, **NETX_DNS_DK-S7G2.zip**, to a known destination folder, for example: `...\Desktop\NETX_DNS_DK-S7G2`.
3. Open the IAR EW workspace file (.eww) by clicking **File >Open Workspace**.

    Navigate to the folder where the **NETX_DNS_DK-S7G2.zip** or a similar zip project has been extracted.
4. Select the **NETX_DUO_DNS.eww** workspace file (.eww) and click **Open**.
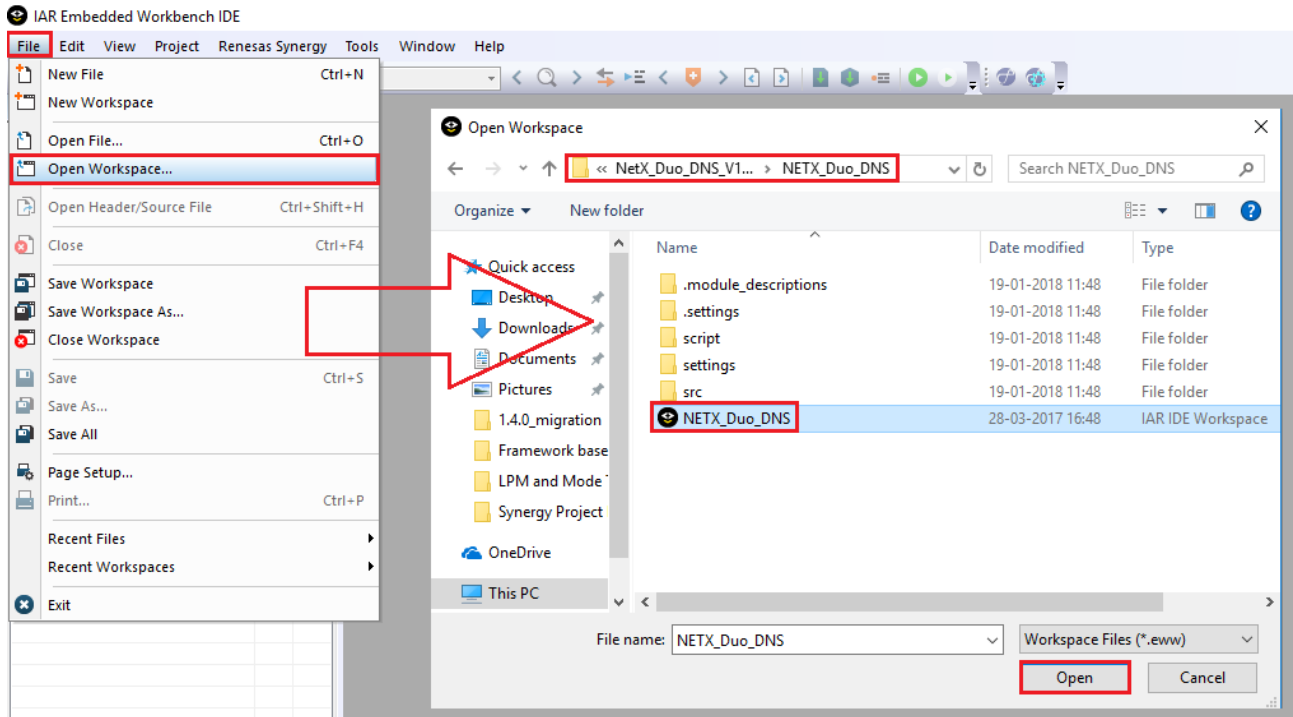
Figure 1.1   Open Workspace in IAR EW for Synergy

5. After opening the project, you should see the project structure (Figure 1.2) in the IAR EW for Synergy ISDE.
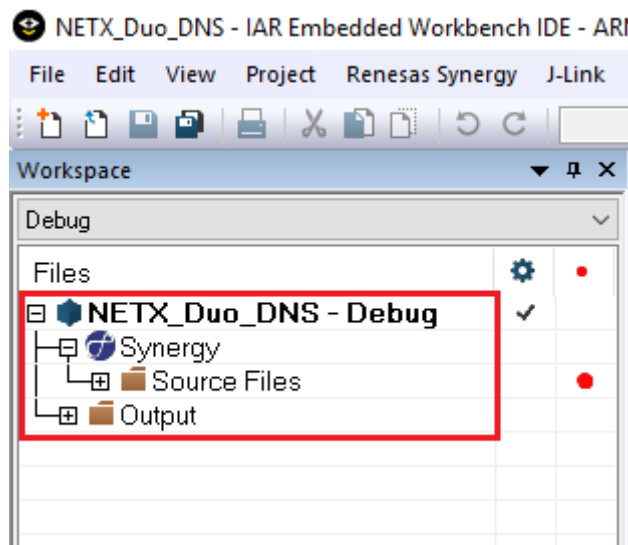
Figure 1.2   Project loaded in IAR EW for Synergy

1.2   Generating the Project Files in the IAR EW for Synergy

Now that the project has been successfully loaded, yheadleou can start configuring the project for your hardware.

Before starting the file generation, it is necessary to set the path to the Renesas Synergy™ Standalone Configurator (SSC) and Synergy Software Package (SSP). This action is requested by default if the next steps are not followed.

To generate the project files:

1.   Click **Renesas Synergy >Settings** to open the **Renesas Synergy Settings**. If the License file and the SSC/SSP folder are already configured, the License area and SSC/SSP location of the form displays (Figure 1.3). If these settings are shown, skip to step 3.
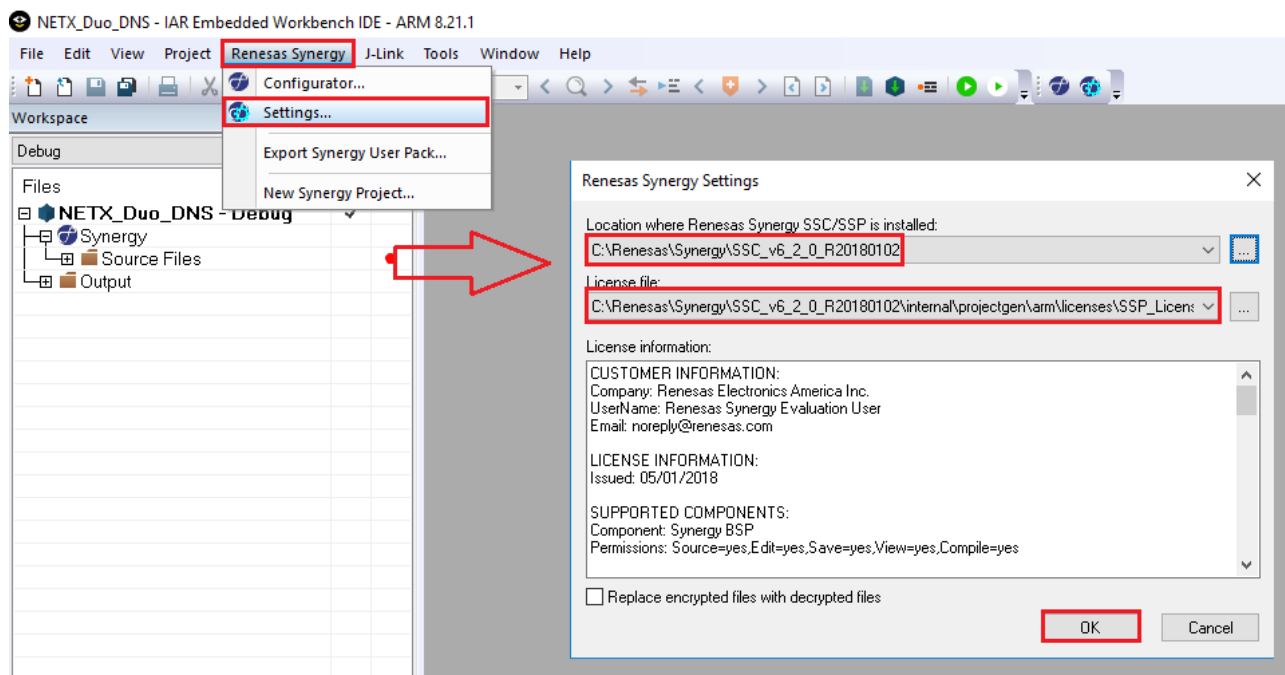

Figure 1.3   SSC/SSP settings

2.   If the License area and SSC/SSP location of the form are empty or not pointing to **C:\Renesas\Synergy\SSC**, perform steps A to F. The settings in these steps only need to be done once.
  A.   Click the browse **<…>** button for the SSC/SSP location. The IAR EW for Synergy IDE displays the Open Dialog box.

Note: If you installed the SSC/SSP to the default location, then SSC/SSP folder is located in the **C:\Renesas\Synergy\SSC** directory.
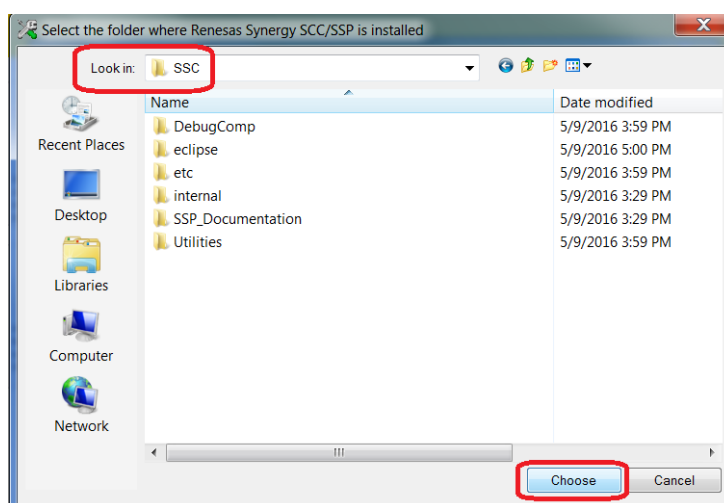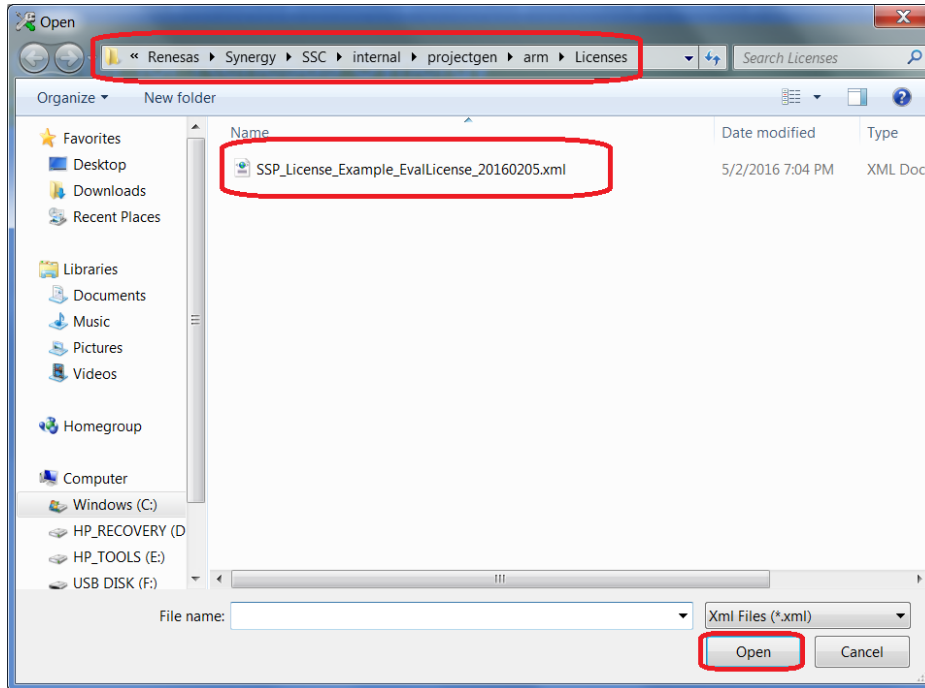

Figure 1.4   SSC folder

    B.   Click **Choose** to set the SSC/SSP location.
    C.   Click the Browse <**…**> button for the license file.
        The IAR EW for Synergy IDE displays the Open Dialog box.

Note: The SSP license is located in **C:\Renesas\Synergy\SSC\intenal\projectgen\arm\Licenses directory**.

    D.   In the directory, select **SSP_License_Example_EvalLicence_*.xml** or
   S**SP_Development_and_Production_License_*.xml**.

Figure 1.5   XML SSP License file

    E.   Click **Open** to set the License file and confirm the configuration window by clicking **OK**.
        Shortcuts are also available for the Renesas Synergy Settings and the Synergy Configurator in the IDE.

Figure 1.6   Shortcuts for SCC/SSP settings

3.   Open the Synergy™ Standalone Configurator, by clicking **Renesas Synergy** > **Configurator**. See Figure 1.7.
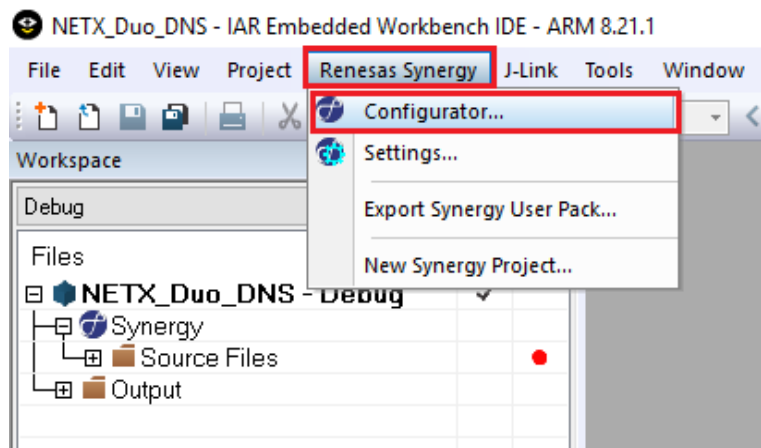
Figure 1.7   Synergy™ Standalone Configurator

Note: At this point, the **synergy** and **synergy_cfg** folders have not been created. These two folders contain the SSP generated files. The following step generates those files.

4. In the Synergy Standalone Configuration window (Figure 1.8), click the **Generate Project Content** button.
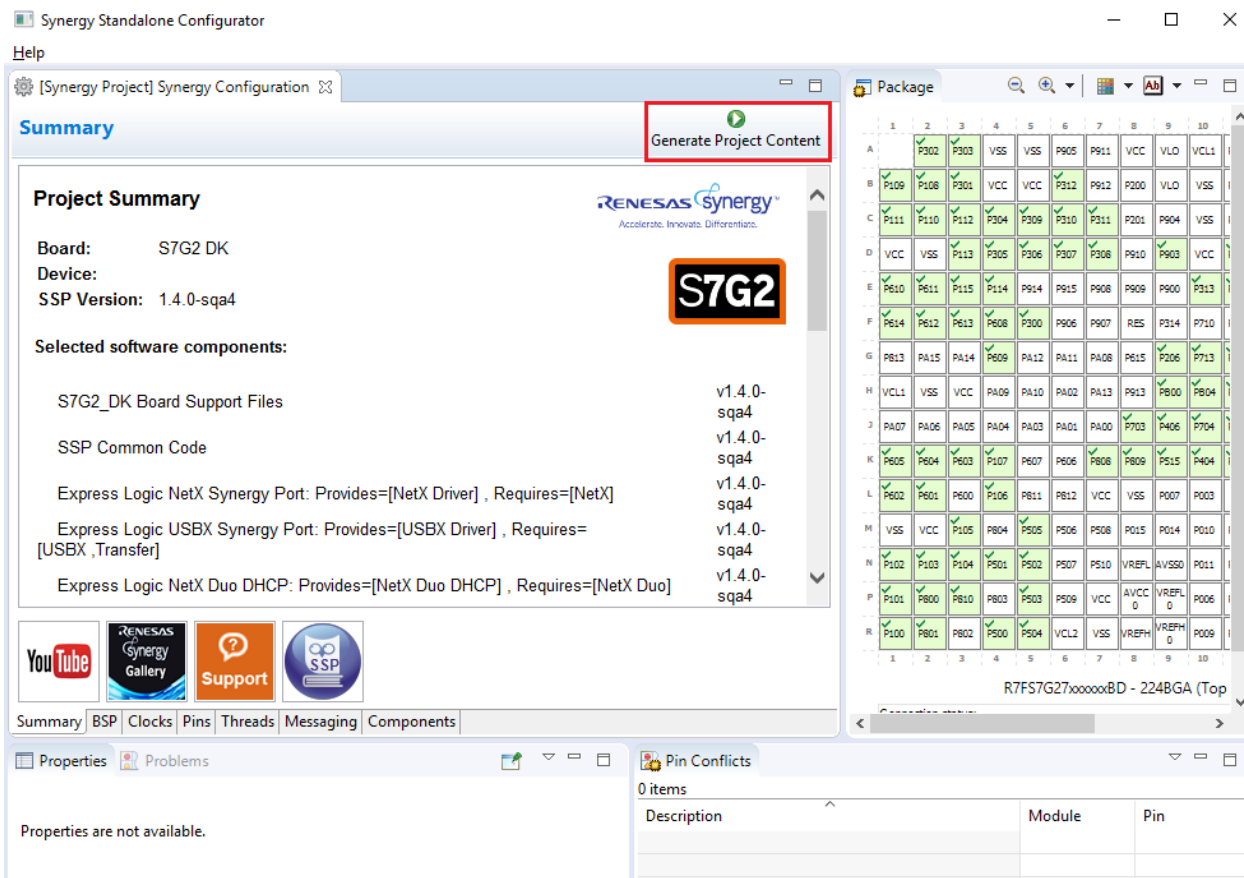

Figure 1.8   Generate Project Content button
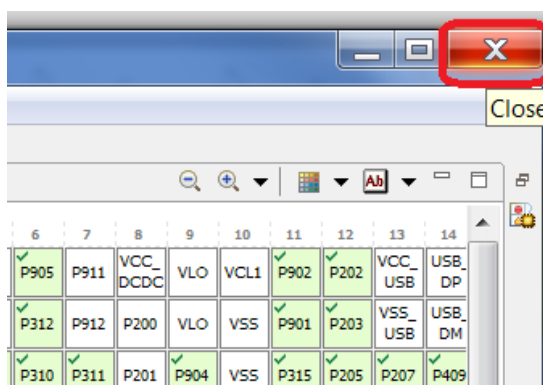
5. Close the Synergy Standalone Configuration.

Figure 1.9   Close the SCC window
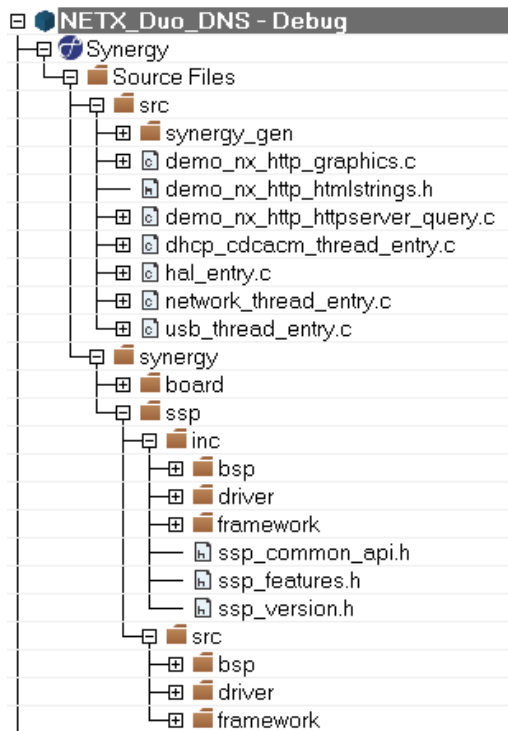
6. The project should resemble the folder structure in Figure 1.10.

Figure 1.10   Synergy Configuration tabs

1.3   Building the Application

1.   Build the project by clicking the **Make** icon in the menu bar. You can also use the **F7** shortcut key.
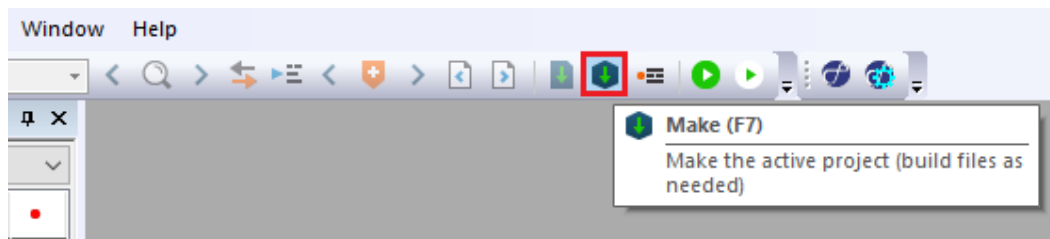

Figure 1.11   Build button

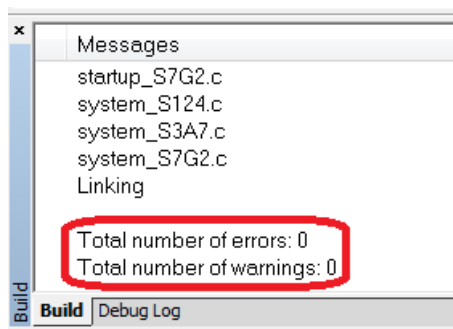2.   A successful build produces an output similar to Figure 1.12.

Figure 1.12   Error Free build

Note: For third party code used in the application project, there can be warnings at compile time. As long as these
         warnings don't affect the functionality of the project, they can be ignored.

1.4   Building from the command line

To build the project from the command line, use the IAR Command Line Build Utility (`iarbuild.exe`) located in the common\bin directory. Typically, this can be useful for automating your testing for continuous integration.

As input you use the project file, and the invocation syntax is:

```
iarbuild project.ewp [ -clean | -build | -make | -cstat_analyze |
-cstat_clean] config[,config1,config2,...]|*[-log
errors|warnings|info|all][-parallel number][-varfile filename]
```

These are the possible parameters:

Parameter	Description
`-build`	Rebuilds and relinks all files in the specified build configuration(s).
`-make`	Brings the specified build configuration(s) up to date by compiling, assembling, and linking only the files that have changed since the last build.
`-cstat_analyze`	Analyzes the project using C-STAT and generates information about the number of messages. For more information, see the *C-STAT® Static Analysis Guide*.
`-cstat_clean`	Cleans the C-STAT message database for the project. For more information, see the *C-STAT® Static Analysis Guide*.
`config|*`	Specifies config, the name of a configuration you want to build, which can be either one of the predefined configurations Debug or Release, or a name that you define yourself. For more information about build configurations, see the *IDE Project Management and Building Guide*, Projects and build configurations, page 94. * (wildcard character), the -clean, -build, and -make commands will a process all configurations defined in the project.
`-log errors`	Displays build error messages.
`-log warnings`	Displays build warning and error messages.
`-log info`	Displays build warning and error messages, and messages issued by the #pragma message preprocessor directive.
`-log all`	Displays all messages generated from the build, for example, the compiler sign-on information and the full command line.
`-parallel number`	Specifies the number of parallel processes to run the compiler in to make better use of the cores in the CPU.
`-varfile filename`	Makes custom-defined argument variables become defined in a workspace scope available to the build engine by specifying the file to use. See the *IDE Project Management and Building Guide*, Configure Custom Argument Variables dialog box, page 87.

If you run the application from a command shell without specifying a project file, you will get a sign-on message describing available parameters and their syntax.

1.4.1   GUIX Studio Command Line

GUIX Studio provides some command line options, based on the command line arguments will not start the GUI interface, but instead just load the `.gxp` project and generate the requested output files.

Command Line Usage

Usage: `guix_studio [OPTION] [ARGUMENT]`

1.   Open `.gxp` project.
2.   Load specified project and generate specified output files.

Examples:

Command line: `demo.gxp`
Open `demo.gxp` project.

Command line: `guix_studio.exe -p demo.gxp`
Open `demo.gxp` project.

Command line: `guix_studio.exe -n -p demo.gxp`
Generate all output files for `demo.gxp` project.

Command line: `guix_studio.exe -n -r -p demo.gxp`
Generate resource files for `demo.gxp` project

Table 1.1 Command Line Options

Option	Description
`-n` `--nogui`	The "No GUI" option. Tells the Win32 version of the guix_studio.exe to just run the command line, and not to start the Studio UI interface.
`-o pathname` `--log`	Log option. Specify a log file.
`-b` `--binary`	Binary resource option. Produces a binary resource file rather than a C file.
`-d display1, display2`	GUIX Studio 67.
`--display`	Display names option. If this option is used then only the specified display names are included in any generated resource or specification files. If this option is not used then all displays are included.
`-t theme1, theme2` `--theme`	Theme name(s) option. If this option is used, then only the specified display names are included in any generated resource or specification files. If this option is not used then all displays are included.
`-l langage1, language2` `--language`	Language name(s) option. If this option is used, then the specified language names are included in the generated resource or specification files. Otherwise all language names are included.
`-r [filename]` `--resource`	The resource option. Specifies that Studio should produce a resource file for previously designated display(s), theme(s), and language(s).
`-s [filename]` `--specification`	The specification option, specify that studio should produce a specification file for designated display(s), theme(s), and language(s).
`-p project_pathname` `--project`	Project pathname option, specify the `.gxp` project to be loaded.

1.5   Running the Application

The application is now ready run on the target hardware. The project settings are all generated and the default debug probe is the J-Link ARM. You can also make use of the IAR I-jet or I-jet Trace when debugging, if you change the debug driver.

1.   Verify the debug probe (Figure 1.13) by clicking **Project** > **Options** > **Debugger** > **Setup** > **Driver**.

RENESAS
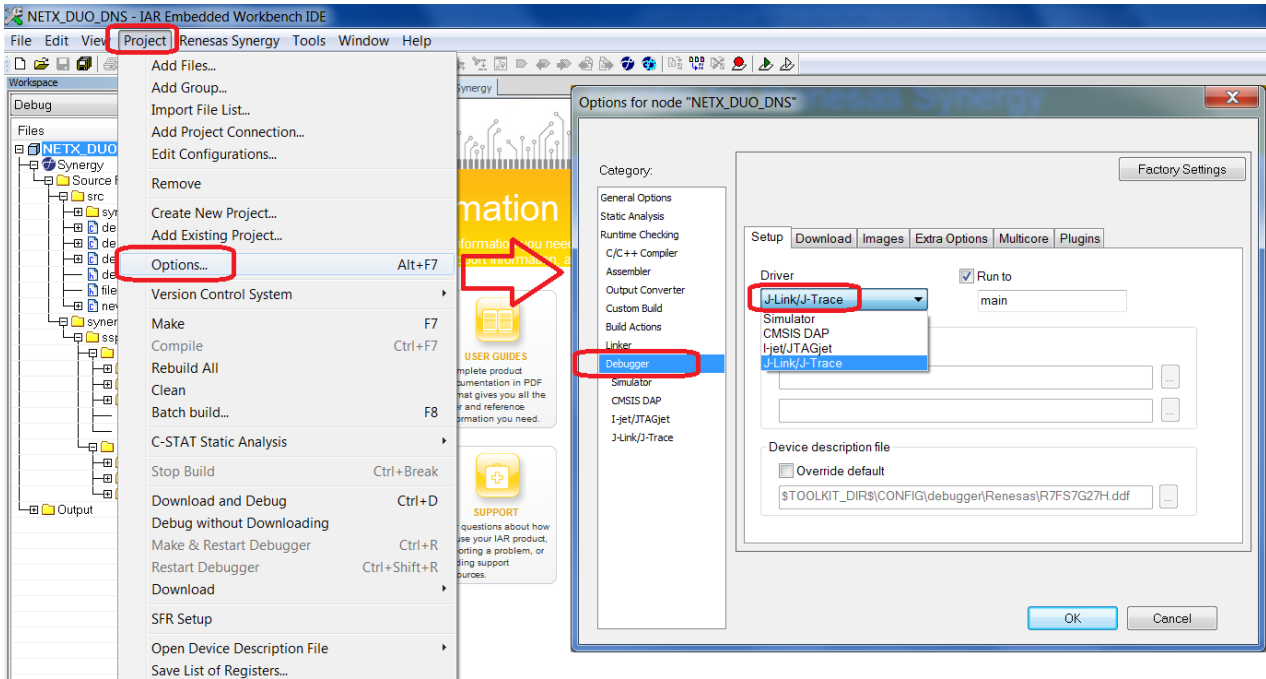
Figure 1.13   Debugger Setup 1

2.   Click **OK** to confirm the **J-Link** or **IAR I-jet** driver.

3.   Press the **CTRL+D** or **Download and Debug** button (Figure 1.14) to start debugging.
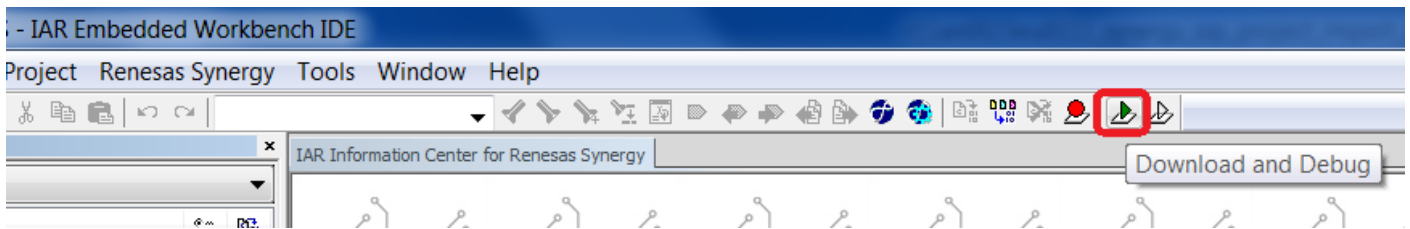
Figure 1.14   Debugger Setup 2

4.   Press **F5** or the **Go** button (Figure 1.15) to start the application.

Figure 1.15   Run button

Note: The application is now running on the hardware. You can pause, stop, and resume the application using the debug controls (Figure 1.16).
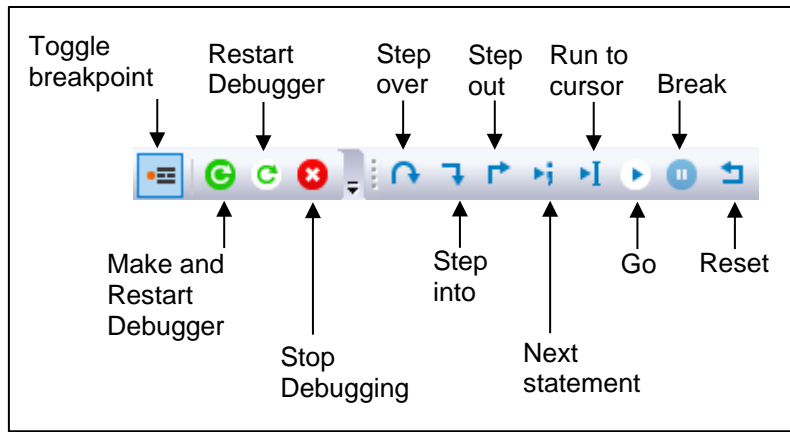
Figure 1.16 Debug control

5. Press **Ctrl** + **Shift** +**D** or the **Stop** button to end the debug session.

2. Importing and Building Projects with e^2 studio ISDE

2.1 Importing an Existing Project in e^2 studio ISDE

1. Start by opening e^2 studio.
2. Open the Workspace that you want to import the file into and skip to step D. Otherwise, proceed with the following steps:

    A. At the end of e^2 studio startup, you see the Workspace Launcher Dialog box shown in Figure 2.1.
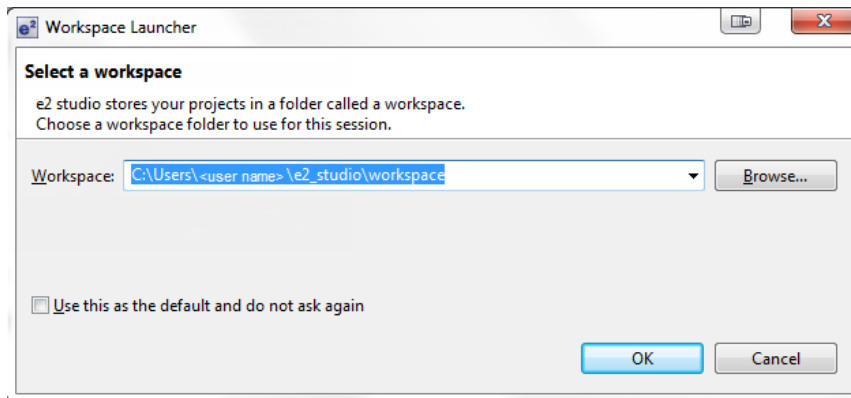
Figure 2.1 Workspace Launcher dialog

    B. If you did not see this dialog box, you might have turned off it off. If this is the case, open your desired project and skip to step D. Otherwise, continue with the following steps.

    C. Enter a new workspace name in the Workspace Launcher dialog (Figure 2.2).

       e^2 studio creates a new workspace with the name entered.
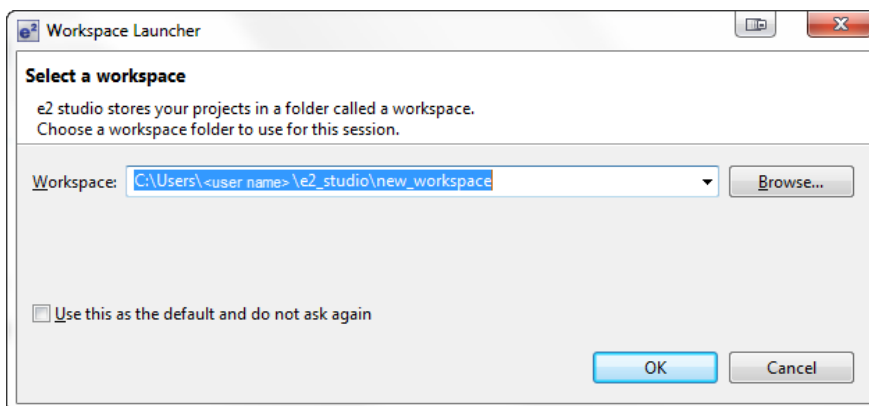
Figure 2.2 Workspace Launcher dialog

D. Click **OK**.
E. When the workspace is opened, you may see the Welcome Window. If that happens, click the Workbench arrow to proceed past the Welcome Screen (Figure 2.3).
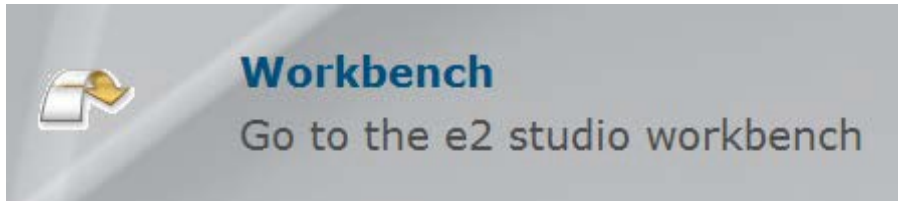
Figure 2.3   Workbench arrow

3. You are now in the workspace that you want to import the project into. Click **File** in the menu bar (Figure 2.4).
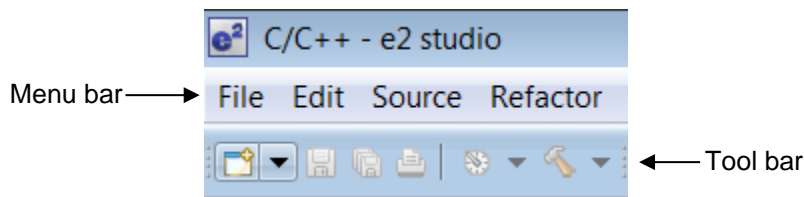
Figure 2.4   Menu and tool bar

4. Click **Import** on the **File** drop-down menu (Figure 2.5).
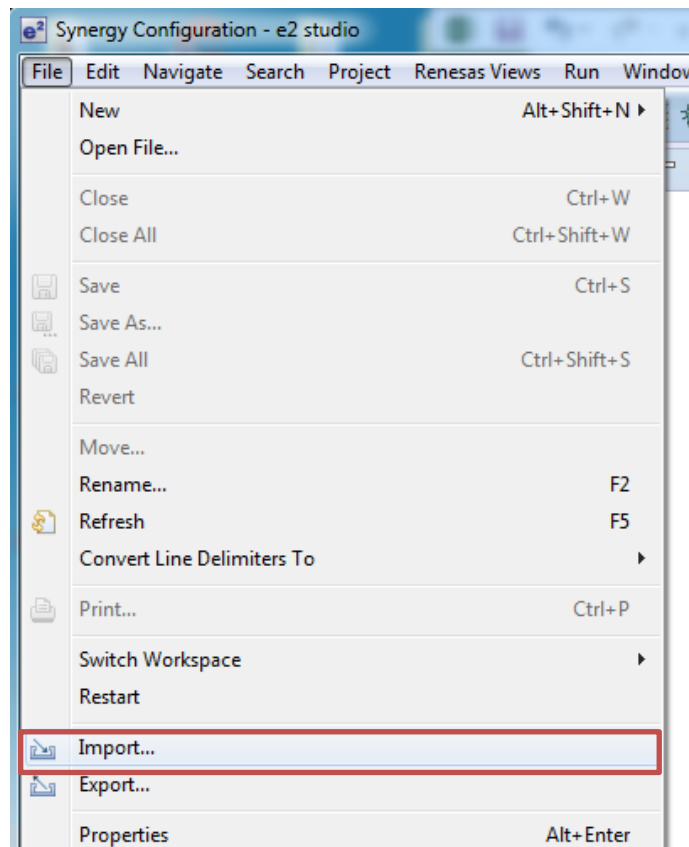
Figure 2.5   File drop-down menu

5. In the Import dialog box shown in Figure 2.6, select the **General** option, and then select **Existing Projects into Workspace** to import the project into the current workspace.
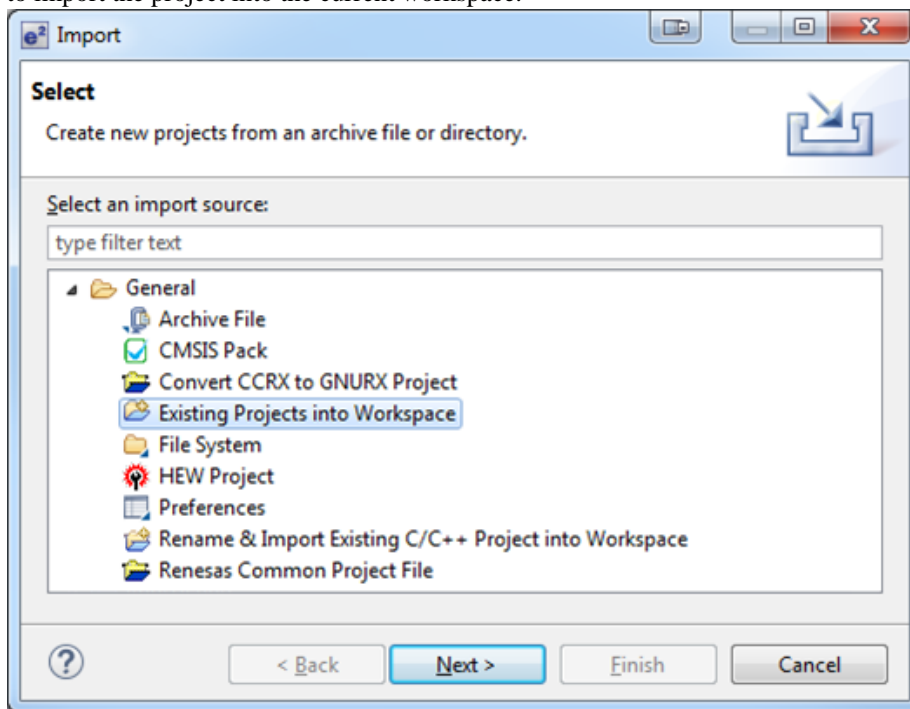
Figure 2.6   Project Import dialog with "Existing Projects into Workspace" option selected

6. Click **Next**.
7. Click **Select archive file** (Figure 2.7).
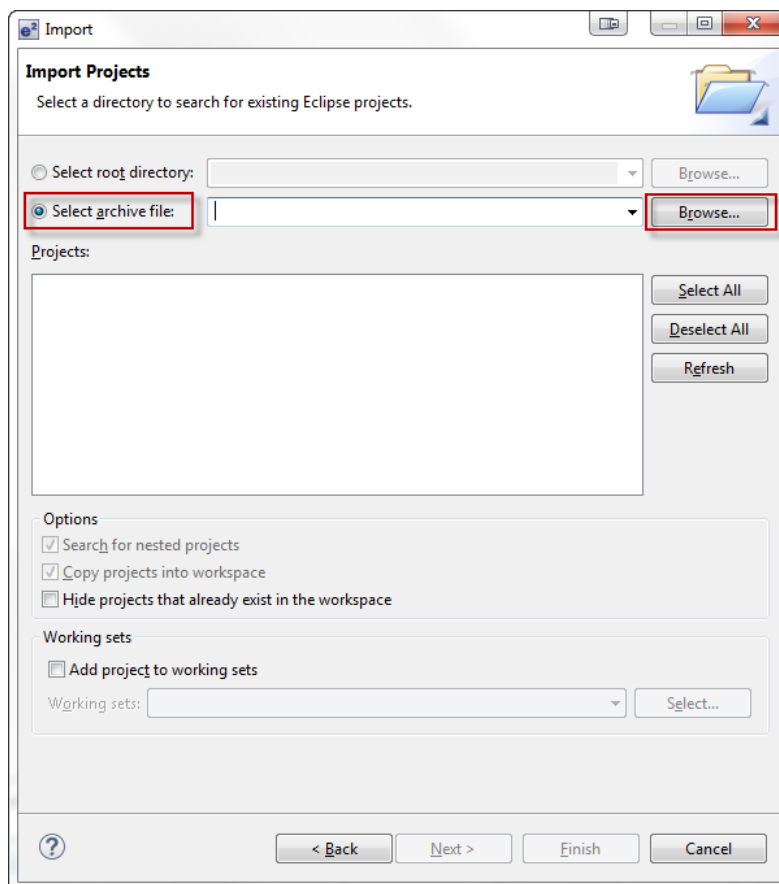
Figure 2.7   Import Existing Project dialog 1

8. Click **Browse**.

9. Browse to the folder where the zip file for the project you want to import is located.

10. Select the file for import. In our example, it is **NETX_DNS_DK-S7G2.zip**.

11. Click **Open**.

12. Select the Project to import from the list of Projects (Figure 2.8).
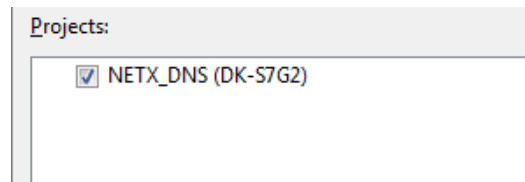
Figure 2.8   Import Existing Project dialog 2

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

2.2   Installing the Synergy License

Building and running example applications requires a Synergy license to be installed in e^2 studio. If this license is not installed, a yellow box is displayed in the lower right hand corner of the ISDE after you have imported your example application.
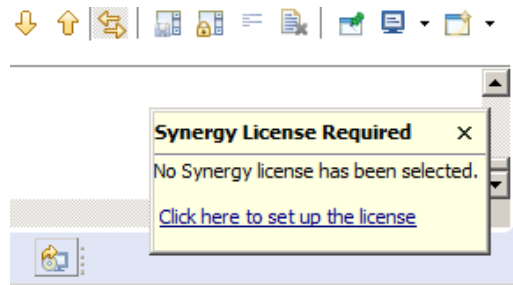
Figure 2.9   Synergy License Required prompt

2.   To install the license, select the **Click here to set up the license** link. This takes you to the Synergy License setup window.
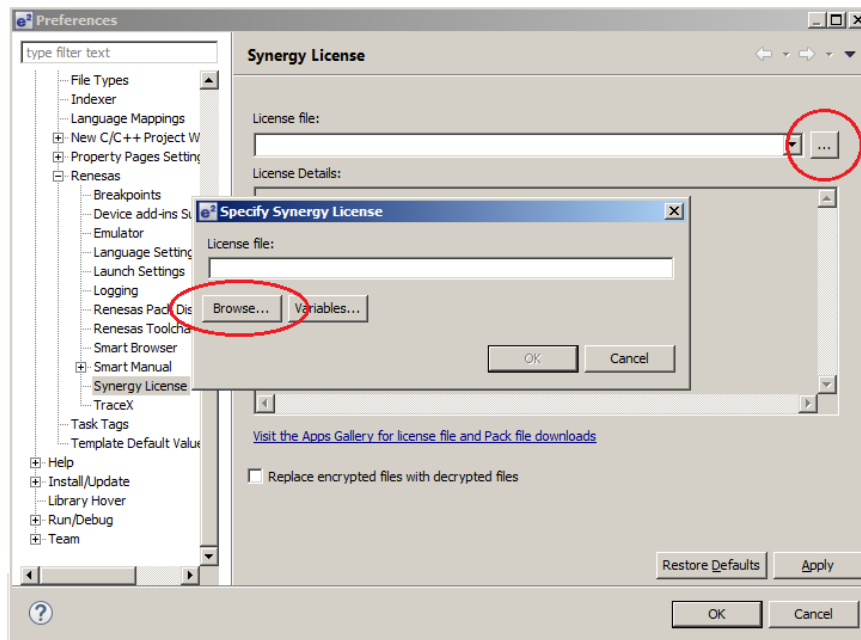
Figure 2.10   Synergy License file

Pressing the two **Browse** buttons takes you to the folder where your default Synergy License is stored. Select this file to install the license. Once this is done, the yellow window disappears and building can begin.

2.3   Generating the Project Files in the ISDE

Now that the project has been successfully imported, you can start configuring the project for the hardware.

1.   If the Project Explorer looks like Figure 2.11, click the arrow to the left to expand the project.

Figure 2.11   Collapsed Project Explorer

2.   Open the Synergy Configuration, if not already open, by double-clicking the **configuration.xml** file in the Project Explorer (Figure 2.12).
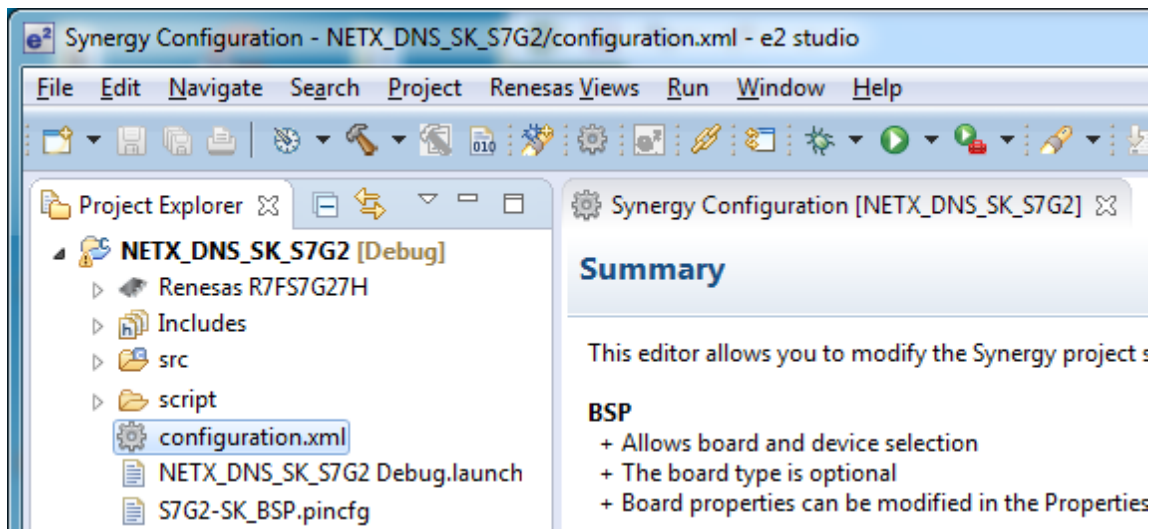
Figure 2.12   Project Explorer

Note: At this point, the **synergy** and **synergy_cfg** folders have not been created. These two folders contain files generated by e^2 studio and the SSP. The next step generates these files.

3.   In the Synergy Configuration window (Figure 2.13), click the **Generate Project Content** button.
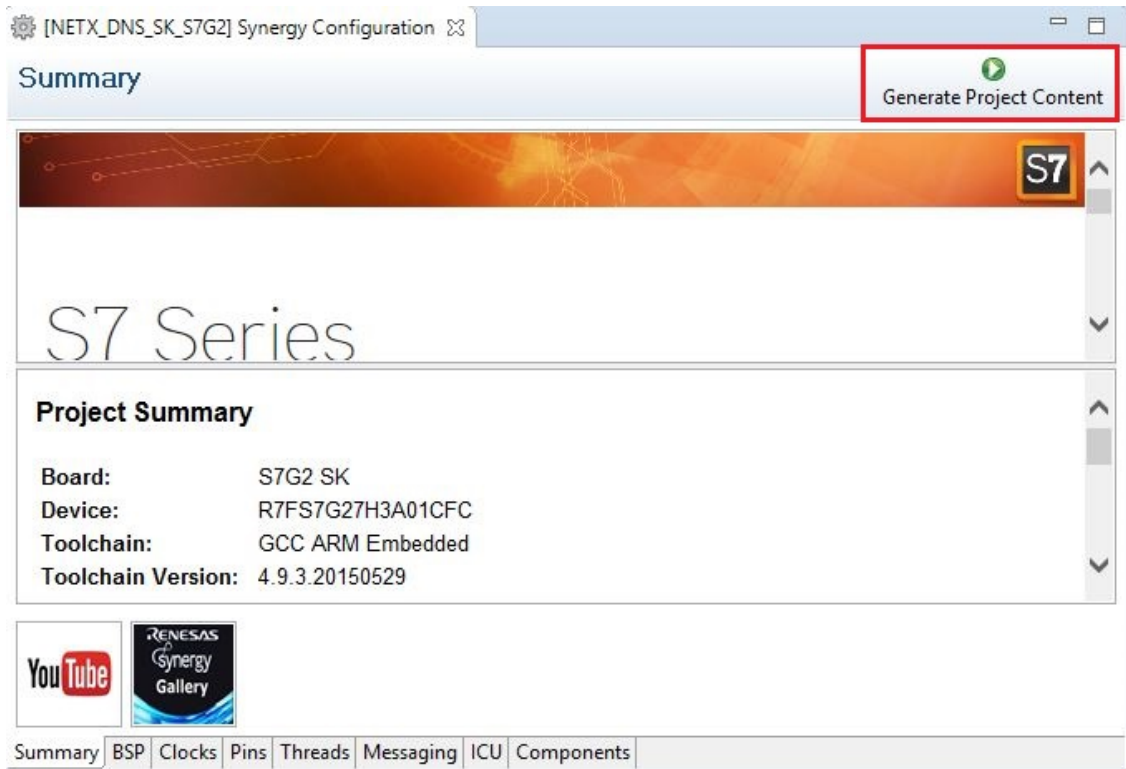
Figure 2.13 Generate Project Content button

4. The project should resemble the folder structure seen in Figure 2.14.
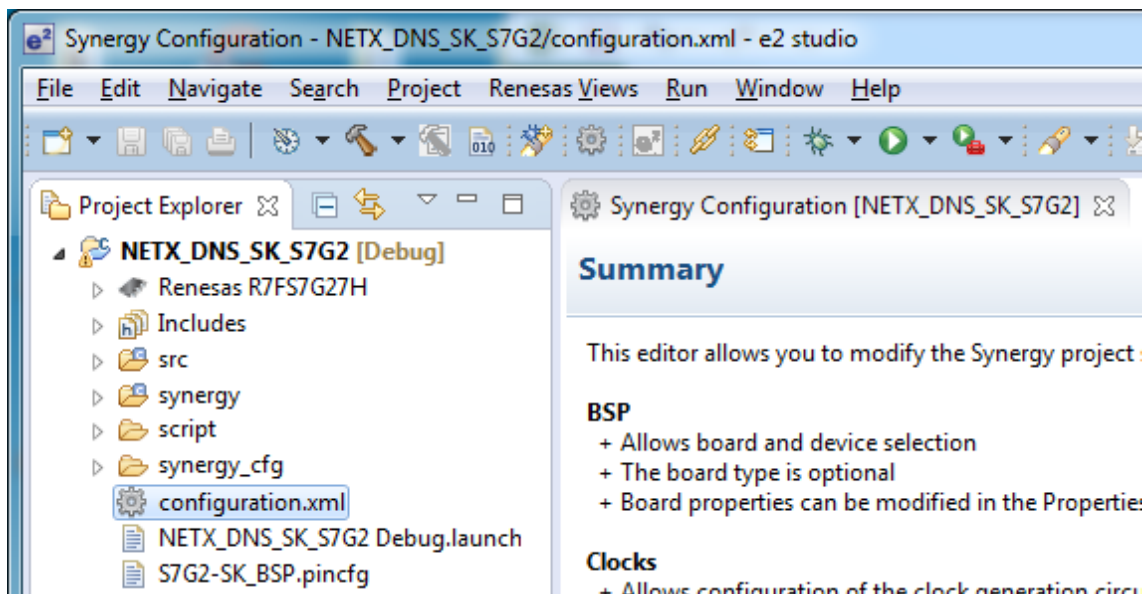
Figure 2.14 Synergy Configuration tabs

2.4 Building the Application

Build the project by clicking the hammer icon (Figure 2.15) on the tool bar.
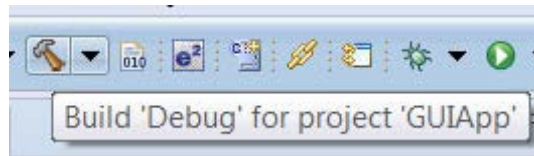
Figure 2.15   Build button

A successful build produces an output similar to Figure 2.16.
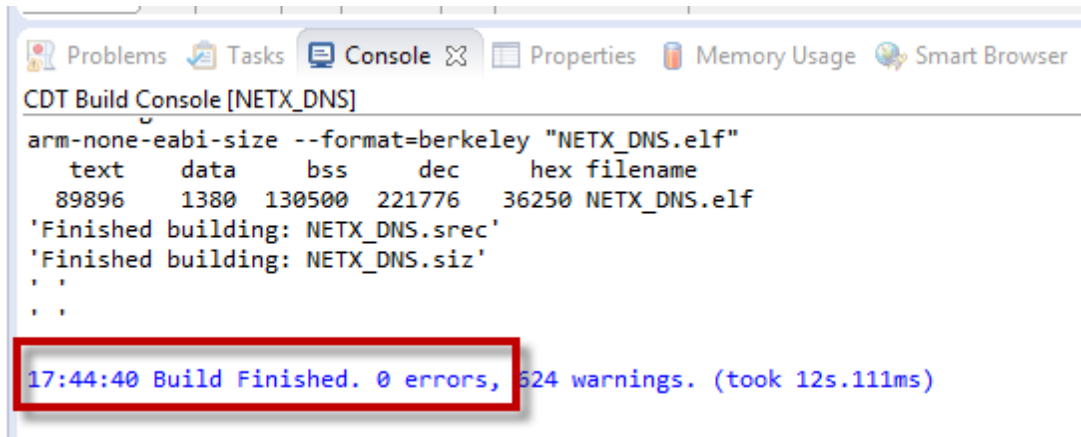
Figure 2.16   Error free build

2.5 Building from the command line

Headless build essentially means building projects from the command line without the use of the e^2 studio UI. This is useful if you want to automate your builds using continuous integration tools like Hudson or Jenkins.

2.5.1 How to build projects

To build your projects using a headless build you will need to the command line version of eclipse called `eclipsec.exe`.

First, you will need to import and build your projects into a workspace to allow headless build to build them. Use the following command to do this:

```
eclipsec.exe –nosplash –debug –consolelog –application
org.eclipse.cdt.managedbuilder.core.headlessbuild -data c:\work\project\test -
import [project location] –build all
```

This command will not only import your project into a workspace called test, but also build for all configurations. If you want to import all projects from a workspace then swap the switch "`-import`" and use "`-importAll`" and just define the path to where all the projects exist.

All the build output, for example, the compilation output and make output, will be sent to the console because of the "`-consolelog`" switch. Please see Table 2.1 for a list of the available switches for headless build.

Table 2.1 Headless build options explained

Option	Description
`-nosplash`	Will supress the e2 studio/eclipse splash screen
`-consolelog`	Any log output is sent to System.out which is normally the command shell itself
`-debug`	Will print any information, warning or error messages to the console
`-application`	Defines the identifier of the application to run, in this case it will be the cdt managedbuilder.
`-import {[uri:/]/path/to/project}`	Import the project defined in the path
`-importall {[uri:/]/path/to/projectTreeURI}`	Import all projects under URI
`-build {project_name_reg_ex{/config_reg_ex}	all}`	Build the project defined based on the selected configuration
`-cleanBuild {project_name_reg_ex{/config_reg_ex}	all}`	Clean and build the defined project based on the configuration
`-I {include_path}`	Additional include_path to add to tools
`-include {include_file}`	Additional include_file to pass to tools
`-D {prepoc_define}`	Addition pre-processor defines to pass to the tools
`-E {var=value}`	Replace/add value to environment variable when running all tools
`-Ea {var=value}`	append value to environment variable when running all tools
`-Er {var}`	Remove/unset the given environment variable
`-Ep {var=value}`	Prepend value to environment variable when running all tools
`-T {toolid} {optionid=value}`*****	Replace a tool option value in each configuration built
`-Ta {toolid} {optionid=value}`*****	Append to a tool option value in each configuration built
`-Tp {toolid} {optionid=value}`*****	Prepend to a tool option value in each configuration built
`-Tr {toolid} {optionid=value}`*****	Remove a tool option value in each configuration built

Note: * Tool option values are parsed as a string, comma separated list of strings or a Boolean based on the option's type.

How to add Headless build into Hudson/Jenkins

To add headless building to your Hudson or Jenkins build system you need to add a build step called "Execute Windows batch command". Ensure you have "-consolelog" enabled so that all the output will appear in the Hudson or Jenkins build output where you can customise by parsing and colour code the output using one of the many plugins available in Hudson/Jenkins.

Figure 2.17 shows a new build step has been added to a Hudson 3.1.0 job. Using environment variables within the job the build step can be customised further.

Build Environment

☐ Configure release build
☑ Set environment variables
Variable Name/Value
```
PROJTOBUILD=${WORKSPACE}/project
```
Please write name/value pairs like name=value, separating each pair with a newline, in this text area.

Enter environment variable key/value pairs in this textarea, separated by newlines.

```
AAA=BBB
PRJHOME=${WORKSPACE}/project
```

Build

▦ **Execute Windows batch command**

Command
```
eclipsec.exe -nosplash -debug -consolelog -application org.eclipse.cdt.managedbuilder.core.headlessbuild -data c:\work\project\test -import
${PROJTOBUILD} -build all
```

See the list of available environment variables

Delete

Add build step ▾

Figure 2.17   Defining a headless build job as a build step in Hudson

GUIX Studio Command Line

For information on the GUIXStudio command line, see section 1.4 Building from the command line)

2.6   Running the Application

The application is now ready to run on the target hardware.

1. Click the drop-down menu for the debug icon (Figure 2.18).
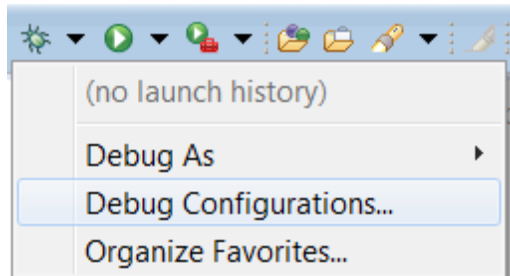
Figure 2.18   Debug options

2. Select the **Debug Configurations...** option.
3. Under the Renesas GDB Hardware Debug section, select the name of the project, which in this case is **NETX_DNS_SK_S7G2 Debug**.
4. Make sure that the ".elf" file name matches the one generated by the project inside of the debug folder.
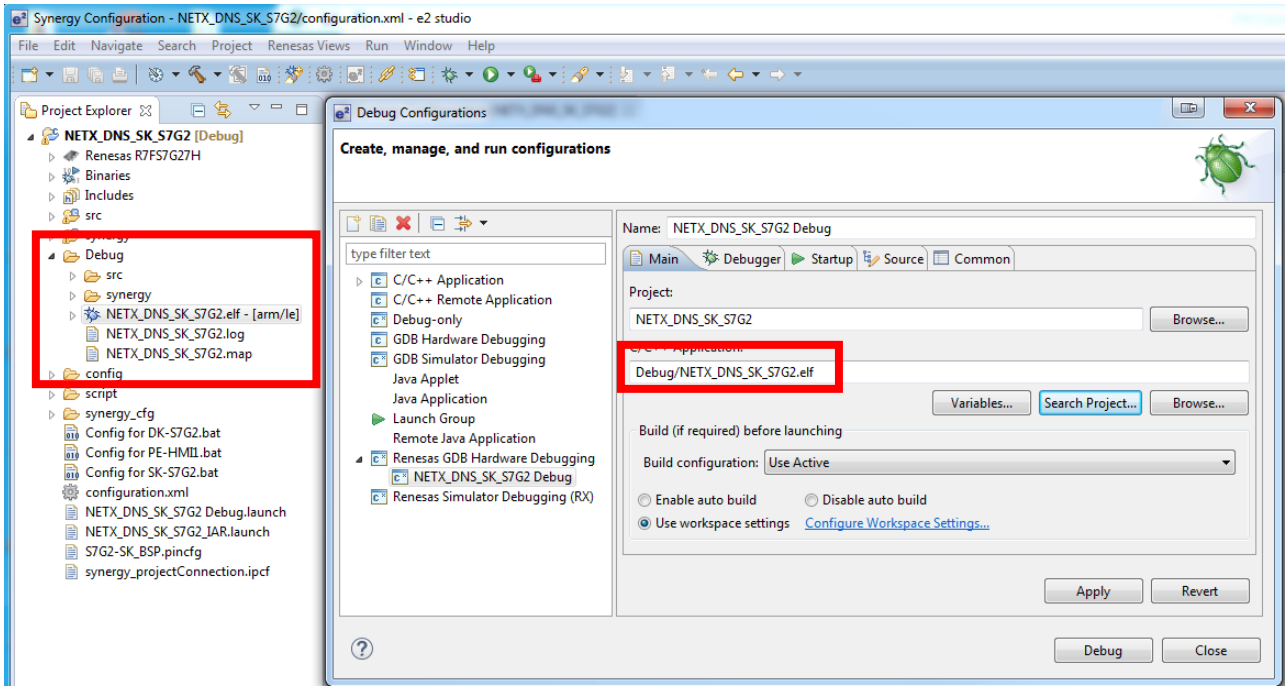
Figure 2.19   Debug Configurations window

5.   Click the **Debugger** tab on the right side of the dialog box.

6.   Ensure the **Debug hardware** setting is set to **J-Link ARM**. If not, change it using the drop-down menu.

7.   Ensure the **Target Device** setting matches the target hardware. If it does not match, click the "**…**" button to select the correct target device from the Synergy device list.
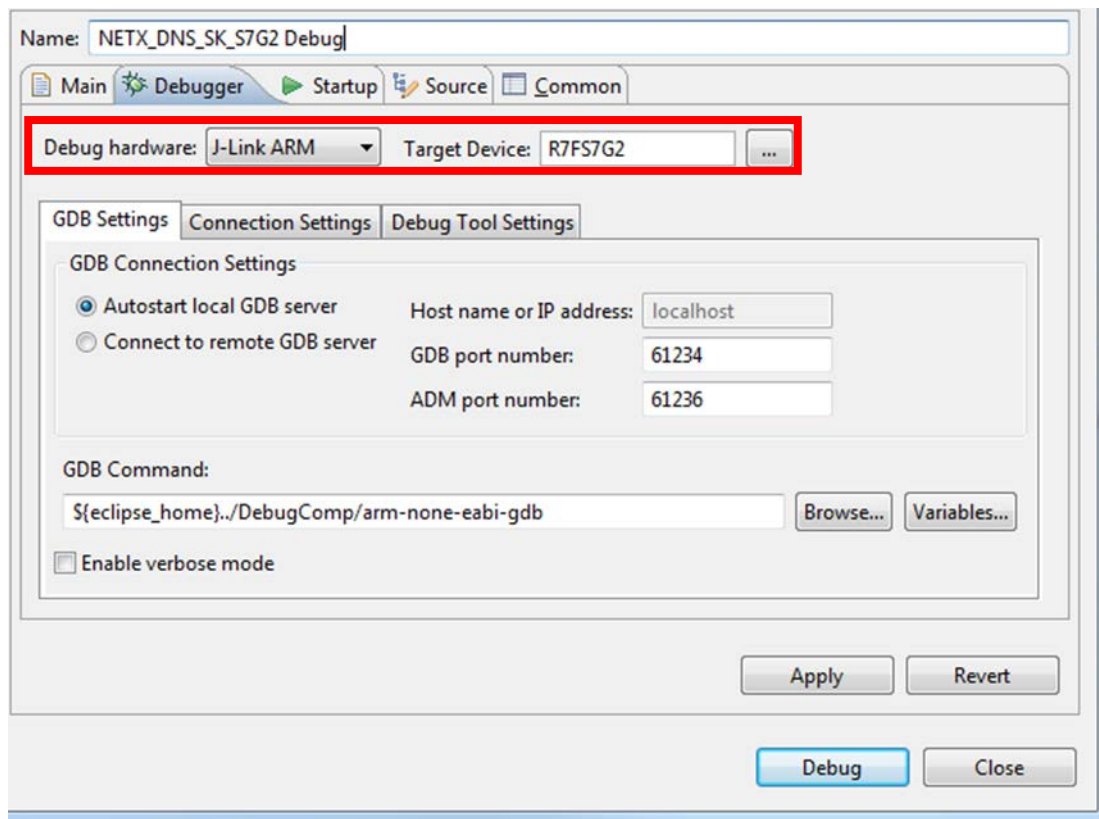
Figure 2.20   Debug Configurations Debugger setup

8. Press the **Debug** button to start debugging.

Figure 2.21   Debug button

9. Select **Yes** to open the Debug Perspective.
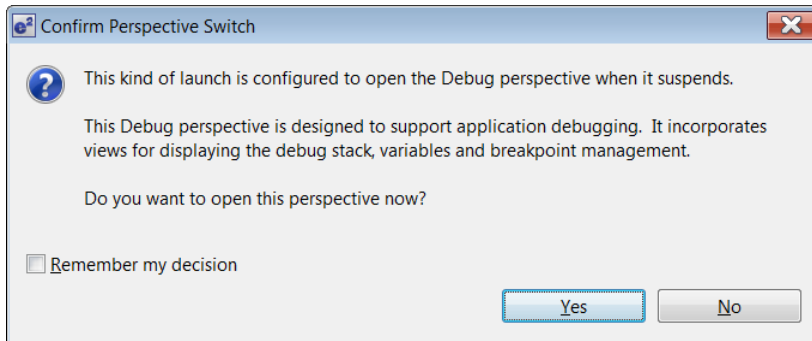
Figure 2.22   Perspective Switch dialog

10. Press **F8** or the **Resume** button to start the application.

Figure 2.23   Resume button

11. Press **F8** or the **Resume** button to continue.

Note: The application is now running on the hardware. You can pause, stop, and resume the application using the debug controls (Figure 2.24).

12. Press **Ctrl + F2** or the **Stop** button to end the debug session.
13. Press the **Synergy Configuration** button to return to the Synergy Perspective.
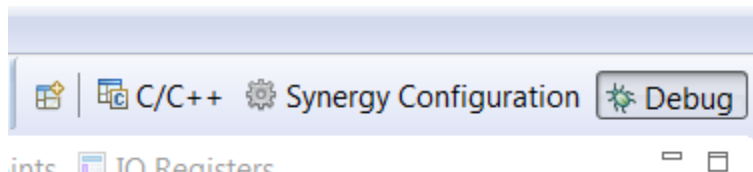
Figure 2.24   Perspective options

3.   Configuring e² studio to build with the IAR compiler

The e² studio ISDE builds with the GNU compiler by default. However, it is possible to configure e² studio to use the IAR compiler instead. Only the compiler that comes bundled with Embedded Workbench for Synergy can be used.

These are the versions of the two IDEs that work together:

e² studio	IAR EW for Synergy
v6.2.0	v7.71.3 / v8.21.1
v6.2.1	V8.23.1

This is how to configure e² studio to build with the IAR compiler:

1. Download and install IAR EWSYN

2. Open e2 studio go to help and select IAR Embedded workbench plugin manager

3. Install the IAR plugins for Synergy
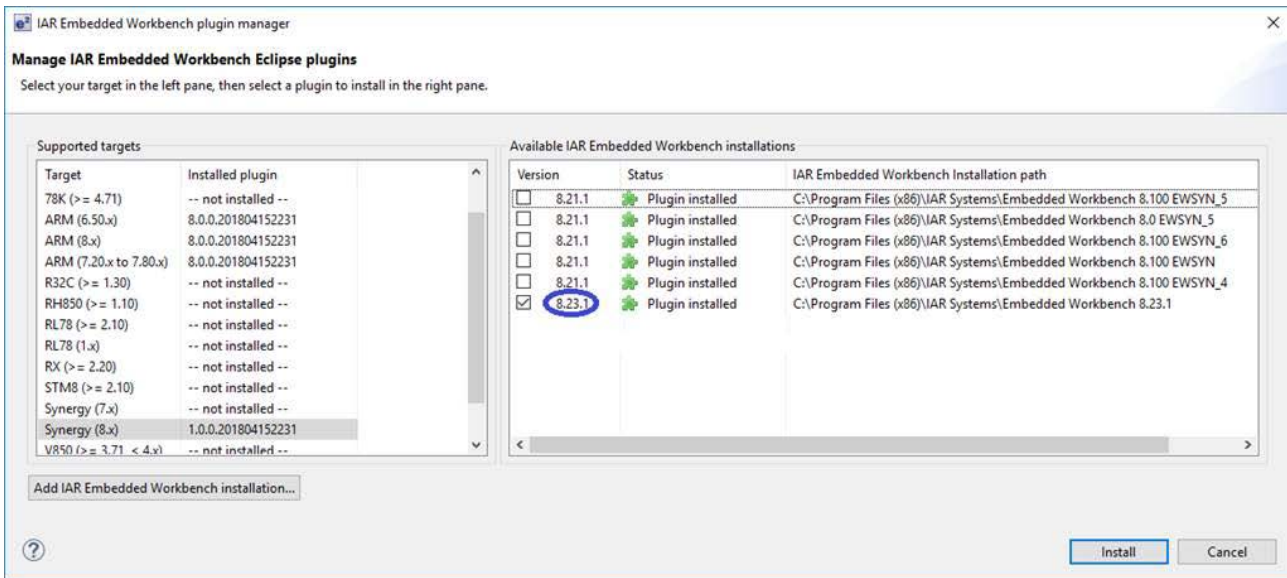
4. Set the new path to IAR EWSYN in Window ➔ Preferences ➔ IAR Embedded Workbench Setup and click 'Apply':
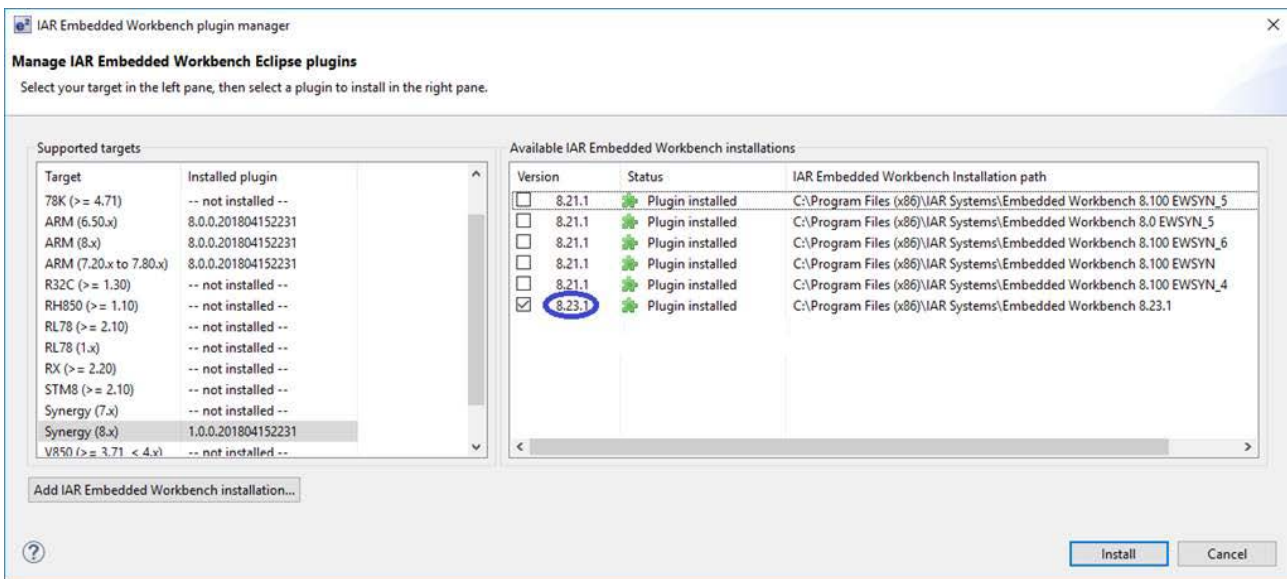
e² studio will need to restart:
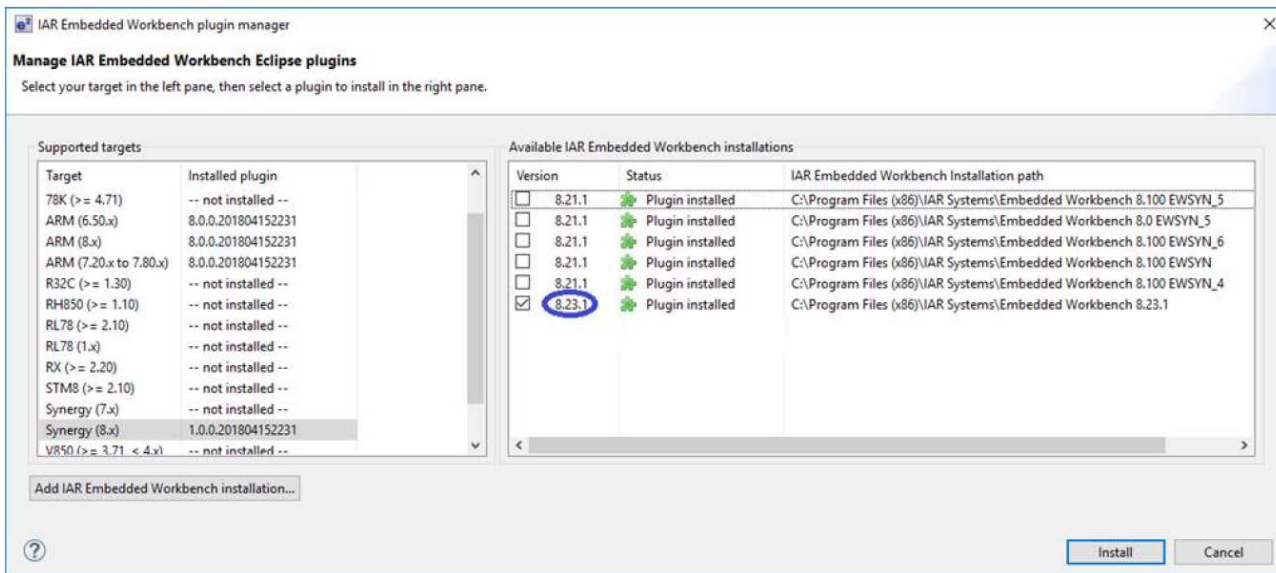
5. Now, when you create a new Synergy project, you can select the IAR compiler from the list of available toolchains and your project will build with IAR compiler:
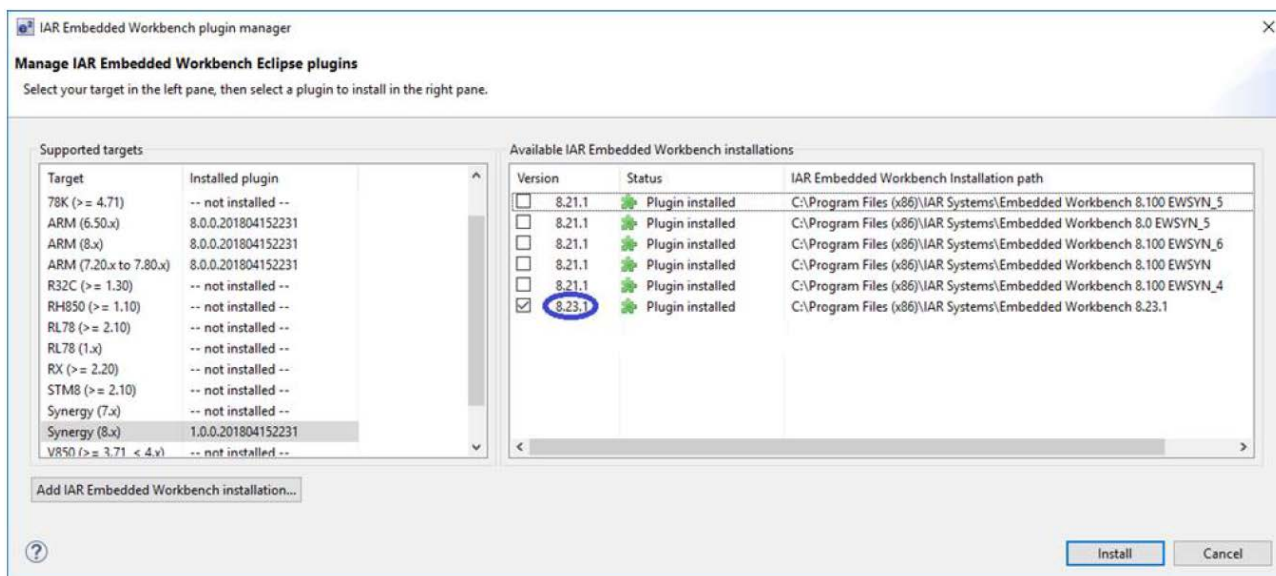
Website and Support

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

Synergy Software   renesassynergy.com/software
    Synergy Software Package   renesassynergy.com/ssp
    Software add-ons   renesassynergy.com/addons
    Software glossary   renesassynergy.com/softwareglossary
    Development tools   renesassynergy.com/tools

Synergy Hardware   renesassynergy.com/hardware
    Microcontrollers   renesassynergy.com/mcus
    MCU glossary   renesassynergy.com/mcuglossary
    Parametric search   renesassynergy.com/parametric
    Kits   renesassynergy.com/kits

Synergy Solutions Gallery   renesassynergy.com/solutionsgallery
    Partner projects   renesassynergy.com/partnerprojects
    Application projects   renesassynergy.com/applicationprojects

Self-service support resources:
    Documentation   renesassynergy.com/docs
    Knowledgebase   renesassynergy.com/knowledgebase
    Forums   renesassynergy.com/forum
    Training   renesassynergy.com/training
    Videos   renesassynergy.com/videos
    Chat and web ticket   renesassynergy.com/support

Revision History

Rev.	Date	Description Page	Summary
1.00	Jan 1, 2018	—	First release document
1.10	Mar 30, 2016	10	Removed appendix "Fixing the license path" and all references to the appendix.
1.11	May 25, 2016	All	Minor formatting and editing changes.
1.12	Jun 30, 2016	All	Added the importing information for the IAR EW for Synergy
1.13	Aug 30, 2016	All	Minor format changes
1.14	Nov 28, 2016	All	Updated for SSP v1.2.0-b1. Minor changes to title, format
1.15	Nov 29, 2016	1	Specified software version numbers for e^2 studio, IAR EW and SSC.
1.16	Feb 15, 2017	All	Updated for SSP v1.2.0. Minor changes to title, format
1.17	Jun 21, 2017	All	Updated for SSP v1.3.0
1.18	Nov 2, 2017	1	Updated software version numbers for e^2 studio, IAR EW, SSC, and SSP.
1.19	Feb 6, 2018	1	Updated software version numbers and screens for e^2 studio, IAR EW, SSC, and SSP.
1.20	Jul 11, 2018	8-9, 17-19, 21-23	Add support for building on the command line
1.21	Oct 22, 2018	1	Fixed broken link for https://synergygallery.renesas.com.

All trademarks and registered trademarks are the property of their respective owners.

RENESAS

Renesas Electronics Corporation

http://www.renesas.com