

# BASICS OF THE RENESAS SYNERGY™ PLATFORM

Richard Oed



# CHAPTER 5

## WORKING WITH THE DEVELOPMENT ENVIRONMENTS FOR THE RENESAS SYNERGY™ PLATFORM

### CONTENTS

<b>5 WORKING WITH THE DEVELOPMENT ENVIRONMENTS FOR THE RENESAS SYNERGY™ PLATFORM</b>	<b>03</b>
5.1 The Eclipse™ Based e <sup>2</sup> studio	03
5.1.1 Short Introduction to the Philosophy of Eclipse	03
5.1.2 Configurators: A Short Introduction	06
5.1.3 Importing and Exporting Synergy Projects	08
5.1.4 Team Collaboration and File Handling	10
5.2 IAR Embedded Workbench®	10
5.2.1 Short Introduction to IAR EW for Renesas Synergy™	11
5.2.2 The Synergy Standalone Configurator: A Short Introduction	13
5.2.3 Importing and Exporting Synergy Projects	16
5.2.4 Team Collaboration and File Handling	16
Disclaimer	17

# 5 WORKING WITH THE DEVELOPMENT ENVIRONMENTS FOR THE RENESAS SYNERGY™ PLATFORM

## What you will learn in this chapter:

- The difference between Perspectives, Views and Editors
- The main elements of the IAR Embedded Workbench® for Renesas Synergy™
- What the different Configurators are for and how they are used.
- How to import and export projects.

Now that we know the details of the Renesas Synergy™ Software Package (SSP) and how to use its APIs, it is time to look into the different development environments available for the Synergy Platform. At the time of writing, there are two environments available: The Eclipse-based e<sup>2</sup> studio from Renesas, which will be covered in chapter 5.1 and the IAR Embedded Workbench® for Renesas Synergy™, covered in [chapter 5.2](#).

## 5.1 The Eclipse™ Based e<sup>2</sup> studio

Renesas' own development environment, e<sup>2</sup> studio, is based on Eclipse™, a popular and widespread Integrated Development Environment (IDE) for different programming languages and target platforms. It can easily be customized and extended and is therefore the development environment of choice for thousands of developers worldwide and a de facto standard. And, it will get updated regularly to use the latest and greatest Eclipse SDK and CDT tools.

e<sup>2</sup> studio leverages all the benefits of Eclipse and includes additional views and configurator perspectives to support all the features of the Synergy Platform. It contains every tool necessary to create, compile and debug projects of any size and complexity and guides the developer through the three phases of a software design: preparation, build and debug.

The following parts of this chapter will talk about the details of the Eclipse workbench and how to use the different elements of it. While we will cover some ground, not everything will be explained in here. More details can be found in the Workbench User Guide by going to *Help* → *Help Contents* inside e<sup>2</sup> studio.

### 5.1.1 Short Introduction to the Philosophy of Eclipse

The main window of e<sup>2</sup> studio, called the workbench, is created of a few basic user interface elements, like perspectives, views, editors, menu- and tool bars.

Once e<sup>2</sup> studio is started, it opens the perspective(s) last used. A perspective is a set of views, editors and toolbars, together with their arrangement inside the workbench. If you re-arrange windows, toolbars or views, these changes will be saved in the current perspective and are available once you open it again the next time. In e<sup>2</sup> studio, there are several pre-defined perspectives, and you can have multiple of them open at the same time (like the C/C++ perspective and the resource perspective). Changing from one perspective to another can be done by clicking on the squared icon besides the perspective list at the far right side of the toolbar, which will bring up a pop-up window listing all the available perspectives, or by selecting *Window* → *Perspective* from the main menu bar. You can close a perspective using the same menu entry or by right clicking on the perspective in the list.

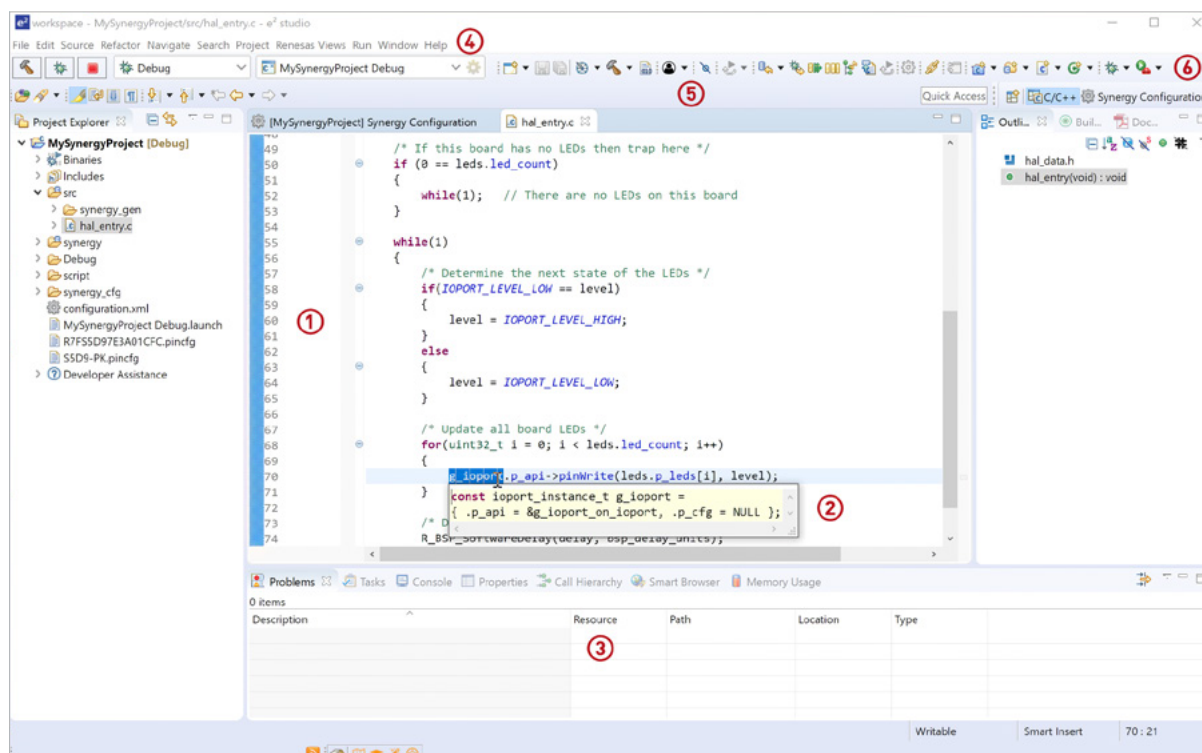


Figure 5-1: The workbench of e<sup>2</sup> studio consisting of editors (1), Smart Manuals (2), views (3), menu bar (4), tool bars (5) and perspectives (6)

## PERSPECTIVES IN e<sup>2</sup> studio

- C/C++:** This is the standard perspective of e<sup>2</sup> studio and is used for developing programs and editing source code. By default, it opens the editor, the Project Explorer view to the left, where you can manage your projects, the Outline view, showing all variables and definitions of the source file active in the editor and at the bottom a tabbed notebook with a couple of stacked views like Problems (e.g. any syntax- or compilation errors), Tasks (shows Doxygen tasks) or Properties.
- Debug:** This perspective is used for running programs and to diagnose and debug problems that occur at runtime. e<sup>2</sup> studio opens the following views by default: The Debug view (yes, the Debug perspective includes a Debug view), with the Debug Tool Bar, editor window(s), the Outline view (as with the C/C++ perspective) to the right and some stacked views for analysis and visualization at the bottom and a tabbed notebook on the top for watching variables, breakpoints, registers, and others.
- Resource:** By default, this perspective includes most of the views from the C/C++ perspective, besides the window at the bottom showing only the Tasks view.
- Synergy Configuration:** This is one of the perspectives, where you will spend a lot of time in at the beginning. The largest part of the workbench is occupied by the Synergy Configuration perspective (remember, a perspective can include other perspectives as well), with several views for the different Synergy Configurators, like the Clock Configurator or the Pin Configurator. It also includes the Properties view, where the settings for the different peripherals, threads and drivers can be made, and the Package view, where you see a visual outline of the package chosen for the microcontroller showing the current pin configuration together with a status showing whether a configuration problem has been detected or not. If needed, the Synergy Configurator can be accessed from the C/C++ perspective and the Debug perspective as well.
- Team Synchronization:** In this perspective, you can merge changes you had made with those of the other team members. [Chapter 5.1.4](#) will deal with that more in detail.
- Tracing:** This perspective opens Statistics and Histogram views, as well as the Debug Call Flow view.

If you change your perspective and you do not like the change, you can always reset it back to the default by right clicking on the perspectives name in the main tool bar and selecting Reset. This also helps to clean up the mess you created after an intensive coding or debugging session, with lots of views and editors open and undocked! No screen is big enough to keep them nicely arranged, so in the end, you will end up with a very cluttered workbench.

## VIEWS

A view is a window where you can examine something, like a set of registers, properties or a list of files. A view can also have alternative representations, like a real time chart of running threads or gauges and controllers to be connected to variables. Views can have their own menu- or toolbar. Actions triggered by a view's menu- or toolbar will only affect items within that view, but not in other views, even if they reside in the same perspective.

Multiple views can be stacked together in the same window, which is then called tabbed notebook. You can also move views from one notebook to another or undock them totally. It's one of the nice features of e<sup>2</sup> studio that you can arrange everything to match your workflow and that last layout of your perspective will automatically be saved for you.

e<sup>2</sup> studio offers a lot of different views. Renesas even created several additional views like the Memory Usage View, the Fault Status view or the RTOS Resources view, making the Integrated Solution Development Environment (ISDE) more versatile. All of them can be found on the main menu bar under *Renesas Views*.

## EDITORS

This is the view (the editor is considered a special type of view) every software developer will spend most of his or her time with. Besides the usual and expected features like code completion or keyword highlighting, the editor built into e<sup>2</sup> studio includes also a special feature, called Smart Manual. It eliminates the need to study 1000's of pages of documentation, as you get context-aware help on the SSP and on the microcontroller itself.

Hovering with the mouse over any of the thousands of highlighted words within e<sup>2</sup> studio will bring up a window displaying relevant information. For variables, the declaration will be displayed, for structures and enumerations all the members and for functions, the description, prototype, and parameter details. If a variable is associated with a register of the microcontroller, detailed information about the bit definitions will be displayed as well. The Smart Manual even pulls in relevant application notes and media-rich instructional material if available and displays them in the Smart Browser view at the bottom of the C/C++ perspective.

This feature is very cool and saves a lot of time, as you will not have to switch back and forth between the SSP and Synergy Microcontroller (MCU) manuals and e<sup>2</sup> studio. You get the relevant information where you need it: Right at the point of your mouse-cursor.

```

/* get the message from the queue */
tx_queue_receive(&g_cdc_queue, rx_msg, TX_WAIT_FOREVER);

/* send the message over the USB port */
g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl,
    /** Write data to communications driver. This call will return after all bytes are written or if a timeout occurs
     * while waiting for access to the driver.
     * @param[in] p_ctrl Pointer to device control block initialized in Open call for communications driver.
     * @param[in] p_src Source address to read data out from
     * @param[in] bytes Write data length
     * @param[in] timeout ThreadX timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF),
     * and timeout value (0x00000001 through 0xFFFFFFFF) in ThreadX tick counts.
     */
    ssp_err_t (* write)(sf_comms_ctrl_t * const p_ctrl,
                        uint8_t const * const p_src,
                        uint32_t const bytes,
                        UINT const timeout);

```

Figure 5-2: The Smart Manual feature of e<sup>2</sup> studio will display all necessary information directly where it is needed

Code completion is another of those features of the ISDE saving a lot of time during code development. Pressing `<ctrl>-<space>` after a variable will bring up a window with the available options, like the members of an API-structure. Clicking on one of the members will insert them into your code. Again, no need to look into the manual, just point-and-click!

## 5.1.2 Configurators: A Short Introduction

The configurators inside e<sup>2</sup> studio graphically guide the software developer through specific device options of the Synergy MCU and advises on their use. Examples of that are pin options or warnings of conflicts. The configurators also generate start-up code or place SSP software components within the project.

### PROJECT CONFIGURATOR

There are several configurators available to help the designer and the first one most users will experience is the Project Configurator, which leads through the process of creating a new Synergy project from scratch or from a template provided by the configurator. It allows to choose the settings for the project, like the toolchain to be used, which device and board to use or if an example project should be created.

At the end of this process, the project and all necessary files will be automatically created and added to the active project in the workbench of e<sup>2</sup> studio. There is no need to create anything manually; no looking up of make-file parameters and settings, no research which header files to include, and how to advise the compiler to use a specific series and device out of the large Synergy microcontroller family. All is done for you once you click on *Finish* on the last configurator screen.

With code generation complete, the Project Configurator will switch to the Synergy Configuration perspective, where the settings for the different SSP components can be made.

## SYNERGY CONFIGURATOR

The Synergy Configurator first displays a summary of the current project and gives an explanation of the different tabs available and how to use them. The next tab is the *BSP* (Board Support Package) tab, allowing to view and edit aspects of the board setup, like device or board selection. In the associated *Properties* view, additional settings for the BSP can be made, for example the size of the main stack (the stack used outside of a thread context) or the time period of the watchdog timer.

The next tab, named *Clocks*, is intended for setting the initial clock configuration. A graphical representation of the on-chip clock system is shown and changes can be made to the clocking tree. Hovering with the mouse over the items will bring up a short description of them. If incompatible settings are made, the respective member will be highlighted in red and an explanation of the problem will be given. Also, the tab itself changes and displays a small exclamation mark, indicating the presence of a problem.

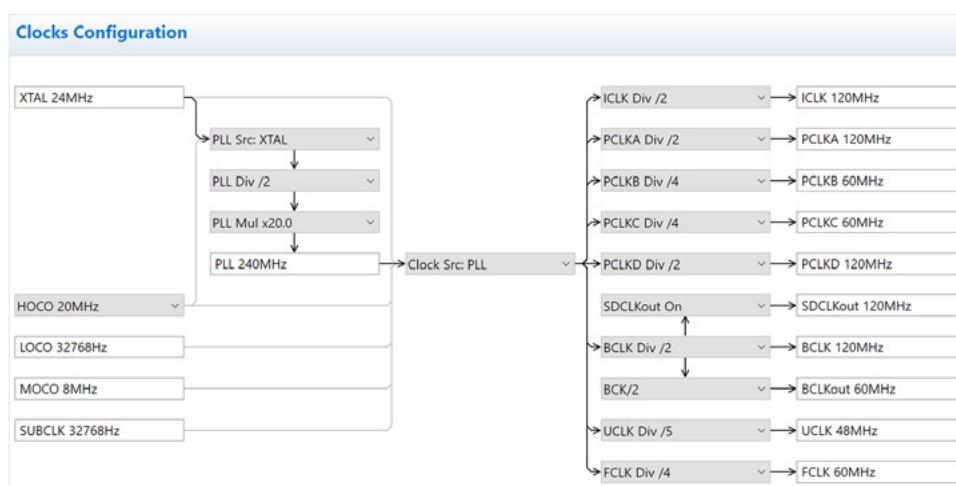


Figure 5-3: The Clocks view of the Synergy Configurator

The *Pins* tab is covering the initial pin setup of the Synergy MCU. Pins can either be listed either based on ports or peripherals. A package view to the right of the configurator shows the package of the device highlighting configured pins and marking errors, if there are conflicts or incomplete settings. These will also be listed in the *Problems*, as well as in the *Pin Conflicts* view.

The tab following, *Threads*, allows you to add and configure the threads within a Synergy project. Different modules and objects can be added to the individual threads and the properties for all of them modified in the *Properties* view. The *Threads* view displays the stacks of the different threads, allowing for a graphical configuration of the modules. New threads can be added easily, with all modules necessary added automatically down to a level, where an intervention of the developer is required. If this is the case, the module will be marked in red and a description of the necessary settings or problems given, once the mouse is hovering over the module. If resolved, the module turns back to its standard colour.

The messaging framework is configured in the *Messaging* tab, allowing you to create classes of application data messages, and define a list of threads subscribing to each message of interest.

The final tab in the Synergy Configuration perspective is named *Components*, allowing you to display and select the different Synergy components. Modifications like adding or removing modules from the current project shouldn't be made in this view, but in the *Threads* tab, as the latter one will allow the configuration of the modules.

Once all settings are made in the Synergy Configuration, the related source code can be generated or extracted from the SSP and added to the project. This is done by clicking on the *Generate Project Content* symbol at the top of the perspective. If you forgot to click on it, don't worry! This will also happen if changes to the configuration or generated files are detected.

The Synergy Configurator is a great tool, as it guides you through all the steps needed to prepare a project and to make the initial settings for it. As with the Project Configurator, there is no need to browse through the thousands of pages of documentation and to study them deeply, as the configurator will provide the necessary information on an abstract level and will assure that all settings are correct and plausible. If you remember how much time you spent during your last project not using the Synergy Configurator on just getting, for example, all the pin-routings and interrupt settings right, dealing with a lot of interdependent hardware registers, you will surely appreciate all the effort Renesas put into these new tools from Renesas. It is much faster this way!

### 5.1.3 Importing and Exporting Synergy Projects

From time to time, you will have a need to import or export projects. Maybe you want to use one of the numerous example projects from the website of Renesas website, or that you want to share your latest and greatest development with one of your peers. No matter if you want to import or export, it can be conveniently done from inside e<sup>2</sup> studio using either the Import Wizard or the Synergy Export Wizard.

## IMPORTING PROJECTS

There are two ways to import projects. Both start at the *File* → *Import* menu. Once the window of the import wizard comes up, expand the *General* entry and you will then have the choice to either select *Existing Projects into Workspace* or *Rename & Import Existing C/C++ Project into Workspace*.

In either case, you will be presented with the choice of either selecting a source directory, where the project to import resides, or selecting an archive file. If one or more projects are found, you can select them for import. If you have chosen the *Rename & Import* version, you also have to give the imported project a new name. Click on *Finish* and the project(s) will be imported into your workspace. You will have to recompile your project before you can run it.

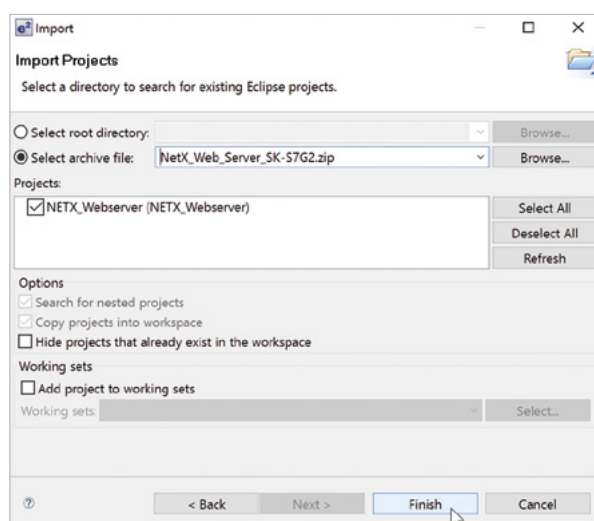


Figure 5-4: The projects found at a given location are listed for import



## EXPORTING PROJECTS

Exporting projects is as easy as importing them. Again, there are two ways to achieve the task. The first one is to go to the menu of e<sup>2</sup> studio and select *File* → *Export*. The second one is to right-click on the project and select *Export Synergy Project*. If using the first version, there are two additional steps to perform in the export window showing up: The first is to expand the *General* entry and the second to select the *Renesas Synergy Archive File* entry from the list appearing.

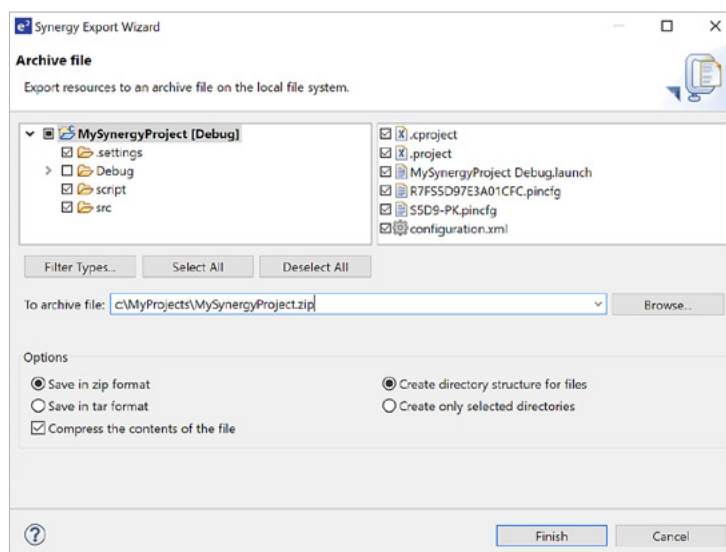


Figure 5-5: The Export Wizard in e<sup>2</sup> studio

Once the Synergy Export Wizard appears, the projects and files to be exported can be selected, along with the compression format and the desired directory structure. Clicking on *Finish* will finalize the export process.

Do not attempt to export your project by just copying the directory of your workspace, as this will violate the license agreement of the Synergy Software Package. You are not allowed to re-distribute distribution of any content from the SSP. And, it is also not necessary to include the SSP files in your export: as the *configuration.xml* file is part of the export and any SSP content can be recreated by opening the file (which will cause the Synergy Configurator to start) and clicking on *Create Project Content* again. This will restore the files omitted during export. A direct copy is also unlikely to work due to multiple hard-references to the project name and location. The Synergy Export Wizard will remove them and the Import Wizard will restore them on import.

## 5.1.4 Team Collaboration and File Handling

Team collaboration inside e<sup>2</sup> studio is possible and is easy, as it provides an interface for version control, so you can check-in and check-out files from inside the ISDE. But which parts of the project need to be checked in? The rule of thumb is: Anything created by the Synergy Configurator does not need to be checked in.

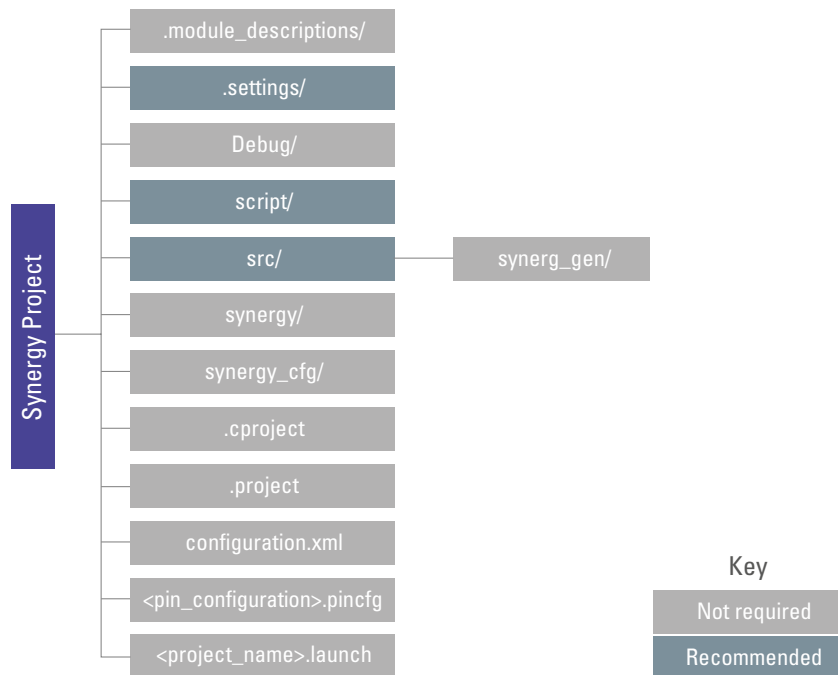


Figure 5-6: There is no need to check-in items in grey

## 5.2 IAR Embedded Workbench®

IAR's Embedded Workbench can be considered as the world's most widely used embedded development environment and is completely integrated with the Renesas Synergy™ Platform. Through the use of the Synergy Standalone Configurator (SSC), all the features of the Synergy Software Package (SSP), ThreadX® and the XWare™ middleware are easily accessible from inside the Embedded Workbench.

As all Renesas architectures are supported by IAR Systems® tools, you can easily reuse your knowledge about this development environment and/or code you have previously written for the other architectures. In addition to that, the unique code analysis tools C-STAT and C-RUN are helping you to take full control of your code. And, Renesas and IAR Systems® have worked together for many years, ensuring the excellent code quality supported by the tools.

The following parts of this chapter will talk about some of the details of the IAR Embedded Workbench and how to use the different elements. While we cover some ground, not everything will be explained in here. More details can be found in the *Getting Started with IAR Embedded Workbench® for Renesas Synergy™* guide and in the *IAR Embedded Workbench® IDE Project Management and Building Guide*. Both of them are either accessible from the *IAR Information Center* or from the *Help* menu inside the IDE.

## 5.2.1 Short Introduction to IAR EW for Renesas Synergy™

Once you launched the IAR Embedded Workbench®, the IDE will show its different elements and the *IAR Information Center*, from where you can access different user and reference guides, as well as support information.

Each window in the IDE has its default location depending on the other currently open windows. You are free to rearrange the windows and create a layout which suits your needs and preferences best. Windows can be docked or floating and if docked, can be organized in tab groups. The current layout is always saved to your workspace once you close the development environment and will be reloaded if you open your workspace again.

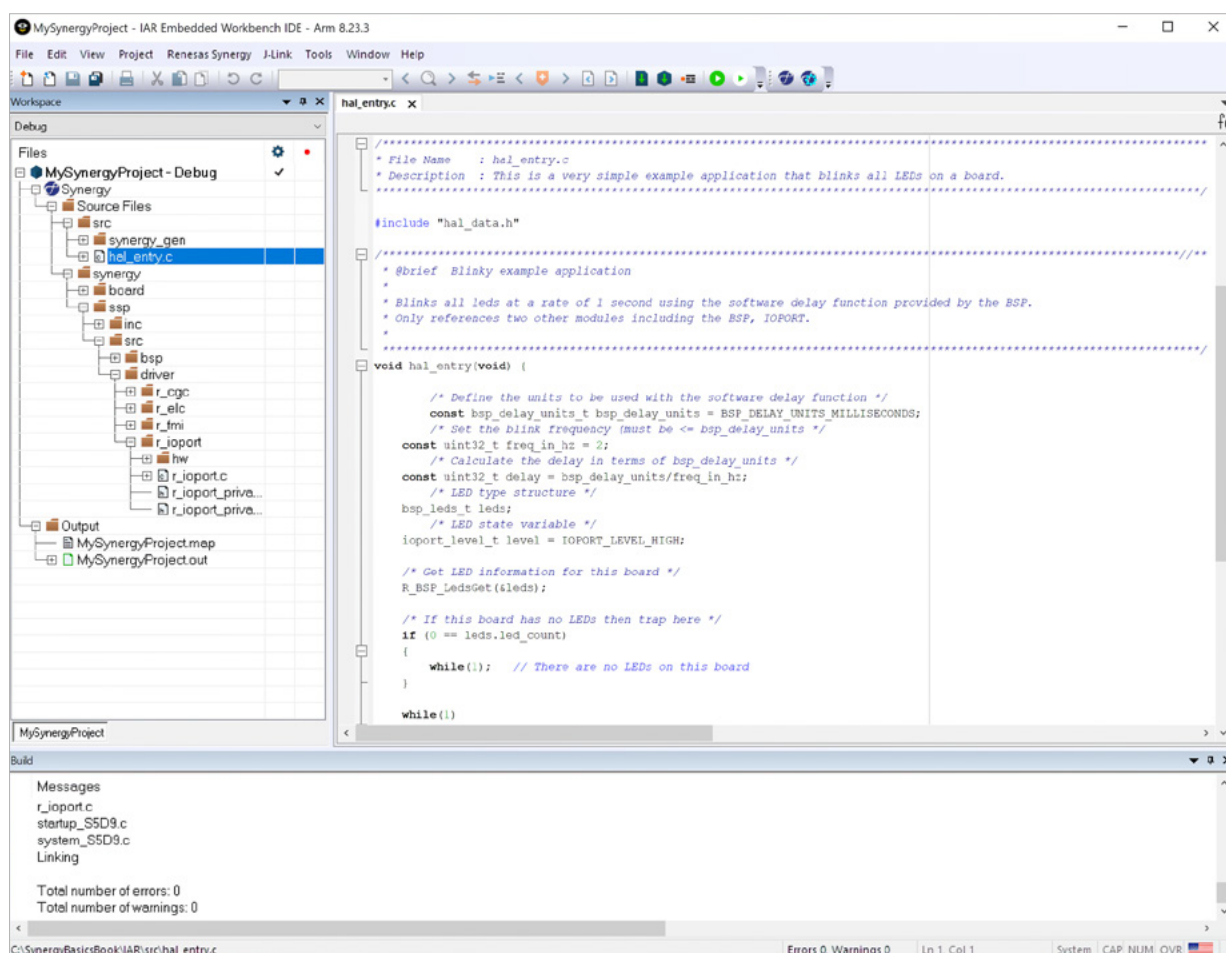


Figure 5-7: The main window of the IAR Embedded Workbench® combines all necessary elements

Now have a look at the main elements of the IDE:

### MENU BAR

The *menu bar* at the top of the window gives you access to commands for handling source and project files, editing source files and enabling breakpoints, opening windows and toolbars, as well as project related operations and access to the Renesas Synergy™ configurator and other tools.

## TOOLBAR

The *toolbar*, which can be made visible by clicking on *Window* → *Toolbar* → *Main*, if not already visible, provides buttons for the most useful commands on the IDE menu and a *quick search* text box. Hovering with the mouse over any of the buttons will reveal a short description of it.

## STATUS BAR

The *status bar* at the bottom of the main window displays information about the progress of any action currently performed, the number of errors and warnings generated during a build, the current position of the cursor in the editor window and a lot of other information regarding the editor.

## WORKSPACE WINDOW

The *workspace window* at the left lets you access your projects and files during the development of your application. At the top is the selector for your build configurations, followed by the projects file list, organized as tree. From here you can access your source files, as well as log and map files once generated. Files are opened by double-clicking on them. Right-clicking on a filename will open a context sensitive menu. A red star in the right hand column beside a filename indicates that this file was changed since the last build of the project, reminding you that a rebuild might be necessary before debugging a program. If you created multiple projects in your workspace, you can select them by clicking at the tab at the bottom.

## MESSAGE WINDOWS

There are several message windows available in the IDE and all can be accessed through the *View* menu. The window you will see most probably first is the *Build* window displaying information about the last build of a project. Other message windows are *Tool Output*, *Debug Log* and *Find in Files*.

## EDITOR WINDOW

This is the window every software developer will spend most of his or her time with. The integrated editor allows to edit multiple files in parallel and provides the developer not only with basic editing features, but also with functions specific to software development. This includes automatic word and code completion, parenthesis and bracket matching or context sensitive help for DLIB library functions and language extensions. An external editor can be used as well if needed.

```

else
{
    level = IOPORT_LEVEL_LOW;
}

/* Update all board LEDs */
for(uint32_t i = 0; i < leds.led_count; i++)
{
    g_ioport.p_api->

```

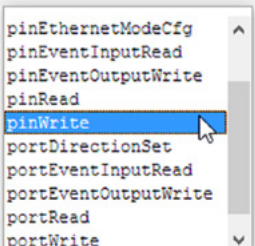


Figure 5-8: The editor offers enhanced features like autocomplete

## 5.2.2 The Synergy Standalone Configurator: A Short Introduction

The Synergy Standalone Configurator (SSC) graphically guides the software developer through specific device options of the Synergy microcontroller and advises on their use. Examples of that are pin options or warnings of conflicts. The configurator also generates start-up code or places SSP software components within the project.

There are several ways to open the SSC. First, it is automatically called once a new Synergy project is created. Other possibilities include the *Synergy Configuration* icon on the toolbar, selecting *Renesas Synergy* → *Configurator* in the menu or by right-clicking on the Synergy entry in the project tree and selecting *Open Renesas Synergy Configurator*.

### PROJECT CONFIGURATOR

If a new project is created, the SSC will lead the user to create a new project from scratch or from a template. It will start with a screen asking for the desired version of the SSP, the board type and the device being intended for use. Clicking on next will bring up the *Project Template Selection* screen (see Figure 5-10) where the user can select one of the templates provided by Renesas. Clicking on *Finish* will start the process to create the project and its associated files.

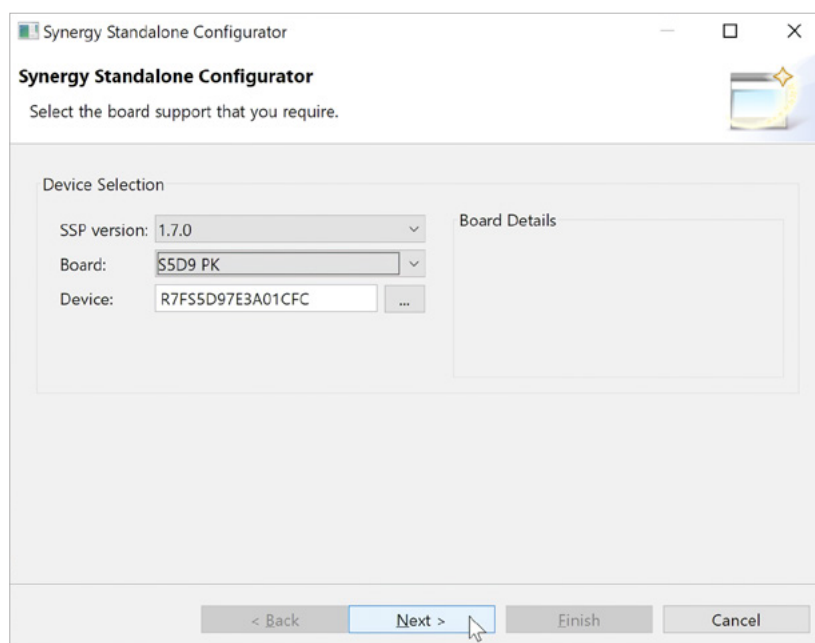


Figure 5-9: The first screen in the SSC lets you select the project relevant parameters

At the end of this process, the project and all necessary files have been automatically created and added to the active project in the workspace of IAR EW for Synergy. There is no need to create anything manually; no looking up of makefile parameters and settings, no research which header files to include, and how to advise the compiler to use a specific series and device from the large Synergy microcontroller family. All is done for you once you click *Finish* on the last configurator screen. With code generation complete, the Configurator will switch to the Synergy Configuration, where the settings for the different SSP components can be made.

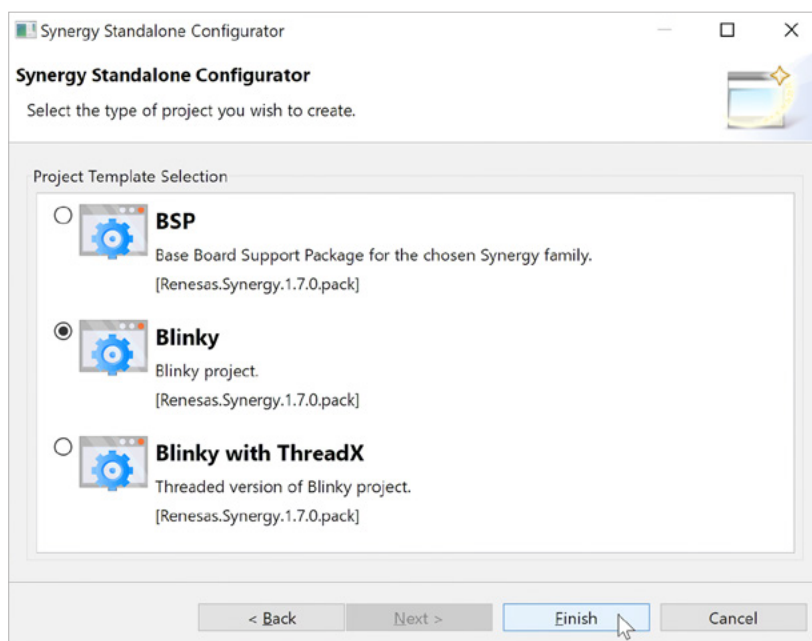


Figure 5-10: This screen lets you select the template for the project

## SYNERGY CONFIGURATOR

The Synergy Configurator first displays a summary of the current project and the main features of the MCU used. The next tab is the *BSP* tab, allowing to view and edit aspects of the board setup, like device or board selection. In the associated *Properties* view, additional settings for the Board Support Package can be made, for example the size of the main stack (the stack used outside of a thread context) or the time period of the watchdog timer.

The next tab, named *Clocks*, is intended for setting the initial clock configuration. A graphical representation of the on-chip clock system is shown and changes can be made to the clocking tree. Hovering with the mouse over the items will bring up a short description of them. If incompatible settings are made, the respective member will be highlighted in red and an explanation of the problem will be given. Also, the tab itself changes and displays a small exclamation mark, indicating the presence of a problem.

The *Pins* tab is covering the initial pin setup of the Synergy microcontroller. Pins can either be listed based on ports or peripherals. A package view to the right of the Configurator shows the package of the device highlighting configured pins and marking errors, if there are conflicts or incomplete settings. These will also be listed in the *Problems*, as well as in the *Pin Conflicts* view.

The tab following, *Threads*, allows you to add and configure the threads within a Synergy project. Different modules and objects can be added to the individual threads and the properties for all of them modified in the *Properties* view. The *Threads* view displays the stacks of the different threads, allowing for a graphical configuration of the modules. New threads can be added easily, with all modules necessary added automatically down to a level where an intervention of the developer is required. If this is the case, the module will be marked in red and a description of the necessary settings or problems given, once the mouse is hovering over the module.

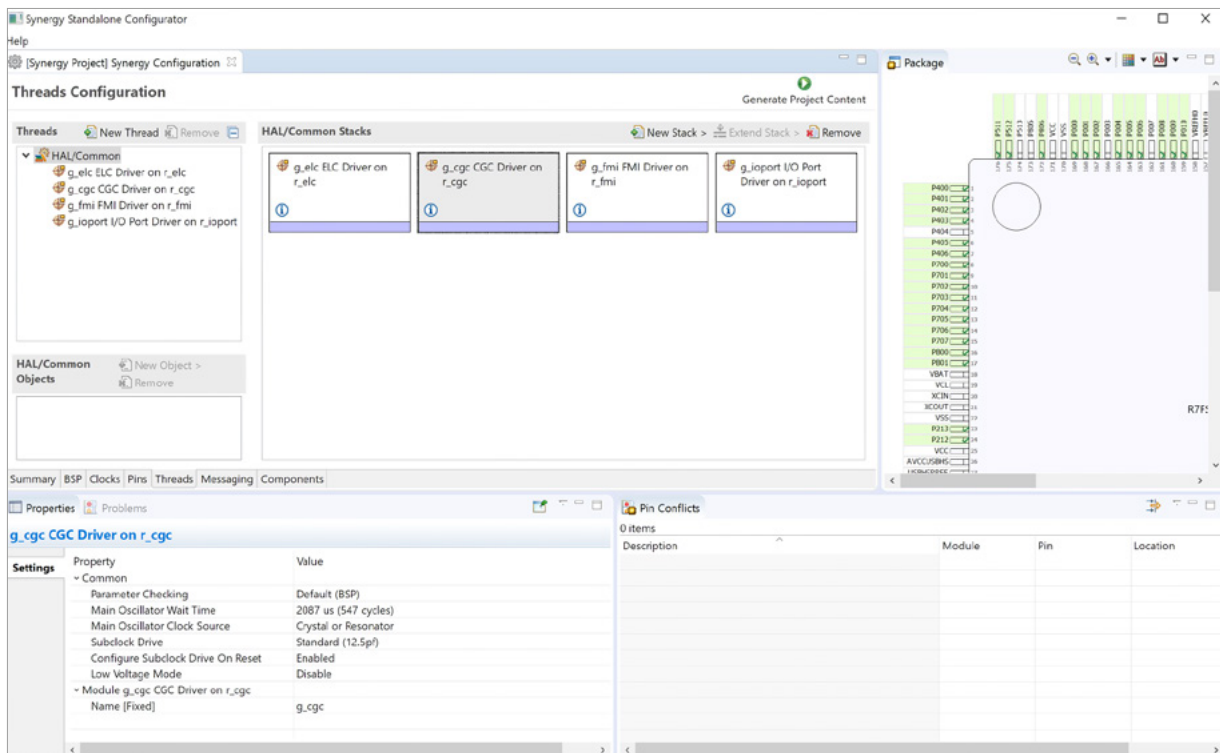


Figure 5-11: The Threads tab of the Synergy Configurator

If resolved, the module turns back to its standard colour.

The messaging framework is configured in the *Messaging* tab, allowing you to create classes of application data messages, and define a list of threads subscribing to each message of interest.

The final tab in the Synergy Configuration perspective is named *Components*, allowing you to display and select the different Synergy components. Modifications like adding or removing modules from the current project shouldn't be made in this view, but in the Threads tab, as the latter will allow configuration of the modules.

Once all settings are made in the Synergy Configuration, the related source code can be generated or extracted from the SSP and added to the project. This is done by clicking on the *Generate Project Content* symbol at the top of the perspective.

The Synergy Configurator is a great tool, as it guides you through all the steps needed to prepare a project and to make the initial settings for it. As with the Project Configurator, there is no need to browse through the thousands of pages of documentation and to study them deeply, as the configurator will provide the necessary information on an abstract level and will assure that all settings are correct and plausible. If you remember how much time you spent during your last project not using the Synergy Configurator on just getting, for example, all the pin routings and interrupt settings right, dealing with a lot of interdependent hardware registers, you will surely appreciate all these new tools from Renesas. It is much faster this way!

### 5.2.3 Importing and Exporting Synergy Projects

Importing a Synergy Project is simple. Copy the project you want to import to a directory of your choice. Inside the IAR Embedded Workbench®, go to *Project* → *Add Existing Project* and load the \*.ewp file to add the project to your existing workspace. If the project you want to import came with its own workspace (if there is a \*.eww file in your directory, it got one), you could also just open this workspace by clicking on *File* → *Open* → *Workspace* and you are all set.

Unfortunately, there is no export feature available right now in the IAR Embedded Workbench® for Renesas Synergy™. To export a project, you will need to manually export the SRC folder containing your customized files (but without the *synerg\_gen* subfolder), the file *configuration.xml*, which contains your Synergy configuration, the \*.pincfg file and the \*.eww, \*.ewp and \*.ewd project files from IAR.

Do not export the *synergy* and *synergy\_cfg* directories. They shouldn't be distributed this way due to the Synergy License agreement. And, there is no need for that, as these files can be regenerated by clicking on the *Generate Project Content* in the Synergy Standalone Configurator. Figure 5-12 also provides you with some information about that.

### 5.2.4 Team Collaboration and File Handling

Team collaboration inside IAR Embedded Workbench® for Renesas Synergy is possible and is easy, as it provides an interface for version control, so you can check-in and check-out files from inside the IDE. But which parts of the project need to be checked in? The rule of thumb is: Anything created by the Synergy Configurator does not need to be checked in.

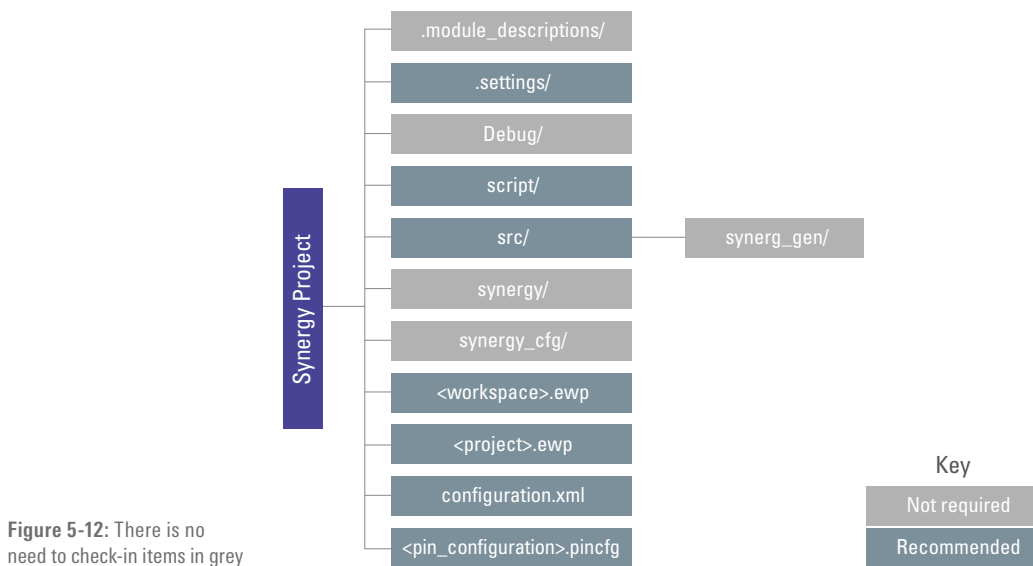


Figure 5-12: There is no need to check-in items in grey

#### Points to take away from this chapter:

- e<sup>2</sup> studio provides different perspectives and views to group functionality.
- IAR Embedded Workbench® for Renesas Synergy is a great tool and fully integrated into the Renesas Synergy™ Platform through the use of the Synergy Standalone Configurator.
- The Smart Manual feature in e<sup>2</sup> studio, the Project Configurator and the Synergy Configurator speed up the development and reduce errors.
- Files created by the Synergy Configurator do neither need to be exported nor checked into a versioning system.



Copyright: © 2020 Renesas Electronics Corporation

Disclaimer:

This volume is provided for informational purposes without any warranty for correctness and completeness. The contents are not intended to be referred to as a design reference guide and no liability shall be accepted for any consequences arising from the use of this book.