RX Smart Configurator

User's Guide: e² studio

R20AN0451ES0170
Rev.1.70
Jan 20, 2026

Introduction

This application note describes the basic usage of the RX Smart Configurator (hereafter called the Smart Configurator), which is an e² studio plug-in tool.

References to the e² studio integrated development environment in this application note apply to the following versions.

- e² studio 2024-01 and later

Target Devices and Compilers

Refer to the following URL for the range of supported devices and compilers:

https://www.renesas.com/rx-smart-configurator

Contents

1. Overview

1.1 Purpose

This application note describes the basic usage of the Smart Configurator and the e² studio integrated development environment, including the procedure for creating a project.

Refer to the User's Manual of the e² studio for how to use the e² studio.

1.2 Features

The Smart Configurator is a utility for combining software to meet your needs. It handles the following three functions to support the embedding of drivers from Renesas in your systems: importing middleware in the form of FIT (Firmware Integration Technology) modules, generating driver code, and making pin settings. Graphical presentation on Smart Configurator, for instance the timing waveform, makes your configuration of middleware and drivers easy.

1.3 Software Components

The Smart Configurator supports two types of software components: Code Generator (CG) and Firmware Integration Technology (FIT). Drivers and middleware supported by each software type are:

- Basic drivers:
  - CG drivers (CMT, A/D Converter, SCI, etc.)
  - FIT modules (CMT, DTC, DMAC, RSPI, SCIFA, etc.)
- Middleware:
  - FIT modules (USB, Ethernet, Flash Memory (programming the on-chip flash memory), etc.)

The basic driver is a control program for peripheral functions of microcomputer such as CMT, A/D converter, SCI, etc. It is convenient to embed a software component (CG driver or FIT module) using code generation function.
In addition, FIT modules can be embedded for using middleware such as USB, Ethernet, and Flash memory (programming the on-chip flash memory) as software components.

2. Creating a Project

The following describes the procedure for creating a C/C++ project using the Smart Configurator.

Refer to the related documents on the e² studio for the details of the e² studio project creation wizard.

2.1 Create a project not using RTOS

The following describes the procedure for creating a project not using RTOS.

(1) Start e² studio and launch a workspace. After starting, select [File] → [New] → [Renesas C/C++ Project] → [Renesas RX] to activate the project creation wizard.

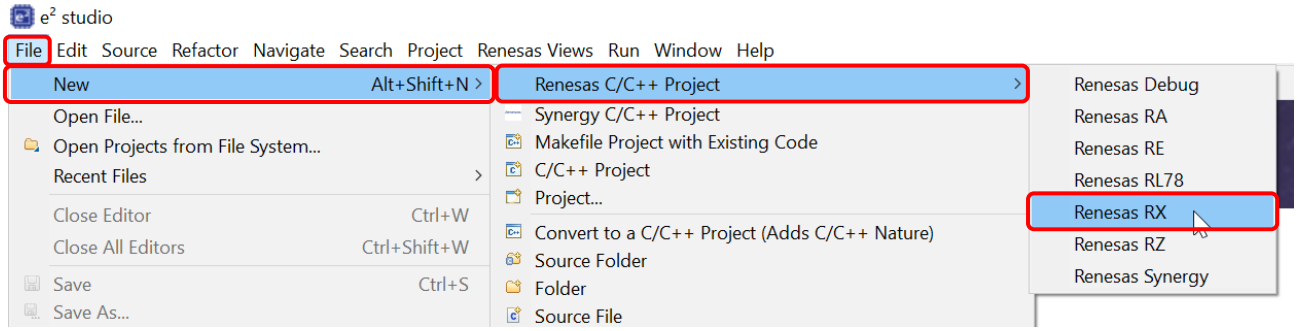


Figure 2-1 Creating a New Project

(2) In the project creation wizard, select [Renesas CC-RX C/C++ Executable Project] or [GCC for Renesas RX C/C++ Executable Project], and click on the [Next] button.

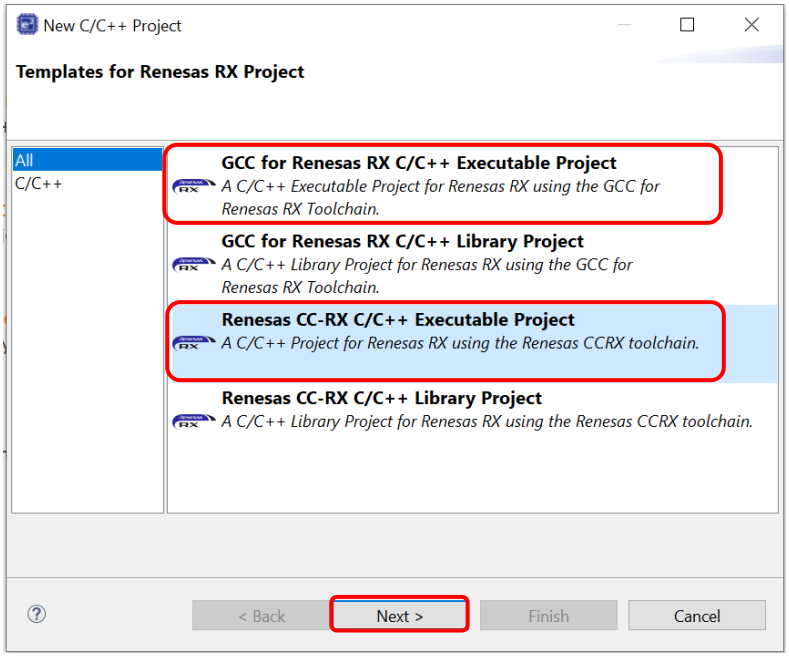


Figure 2-2 Templates for New C/C++ Project Dialog Box

(3) Enter project information. Click on the [Next] button to continue.
(for e.g. CC-RX executable project, Project name: "Smart_Configurator_Example")

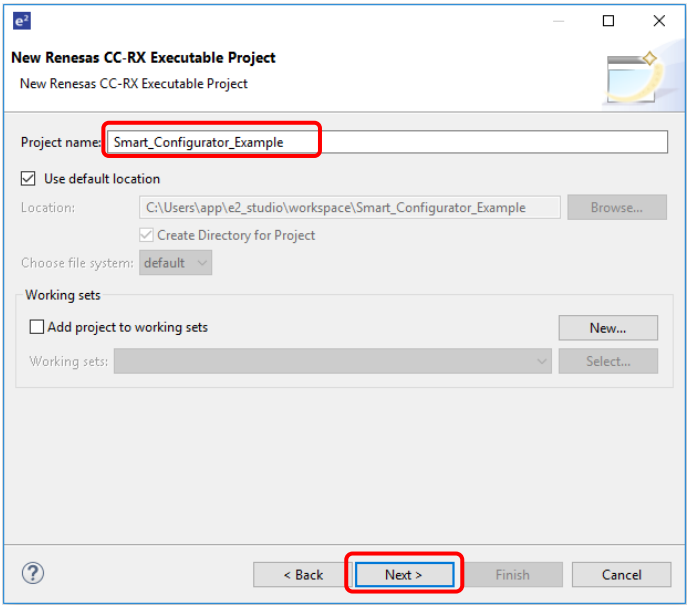


Figure 2-3 Creating a New Renesas CC-RX Executable Project

(4) Select C or C++, toolchain, board, device, and debug configuration. Keep RTOS as “None”. Click [Next].

(for e.g. Target Board: RSKRX64M)

Note: By selecting a board, the following settings is auto selected: Initial pin setting, Clock Frequency, Target device.

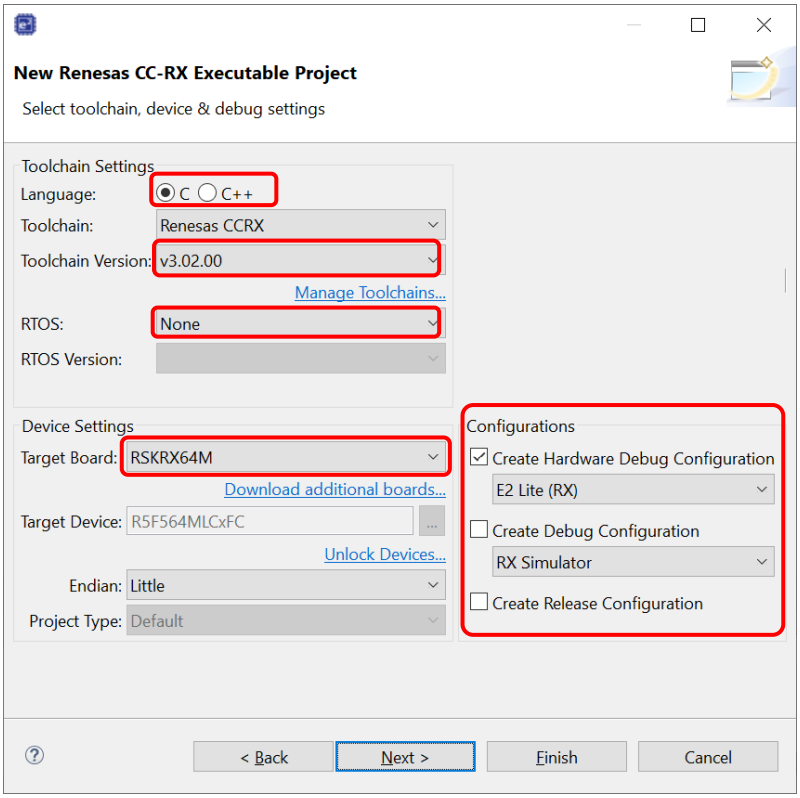


Figure 2-4 Selecting the Toolchain, Device, and Debug Configuration

(5) In the [Select Coding Assistant settings] dialog box, select the [Smart Configurator] checkbox and click on the [Finish] button.

Note: [Smart Configurator] checkbox is enabled only if device supported by Smart Configurator is selected at (4).

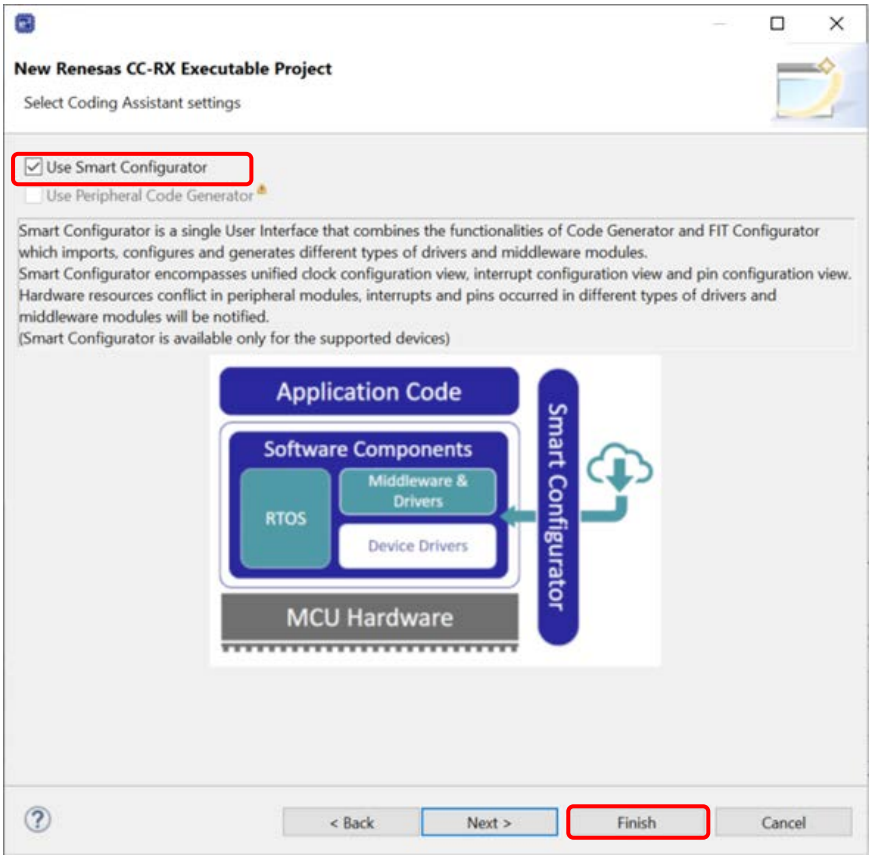


Figure 2-5 Selecting the Coding Assistant Tool

(6) Wait for completion of project creation.

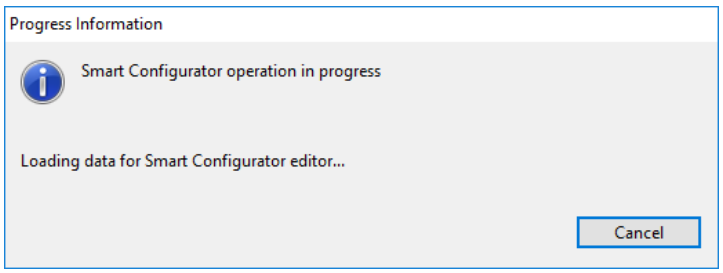


Figure 2-6 Progress of Project creation

(7) After a new C/C++ Project is successfully created, the project will be opened in the Smart Configurator perspective.

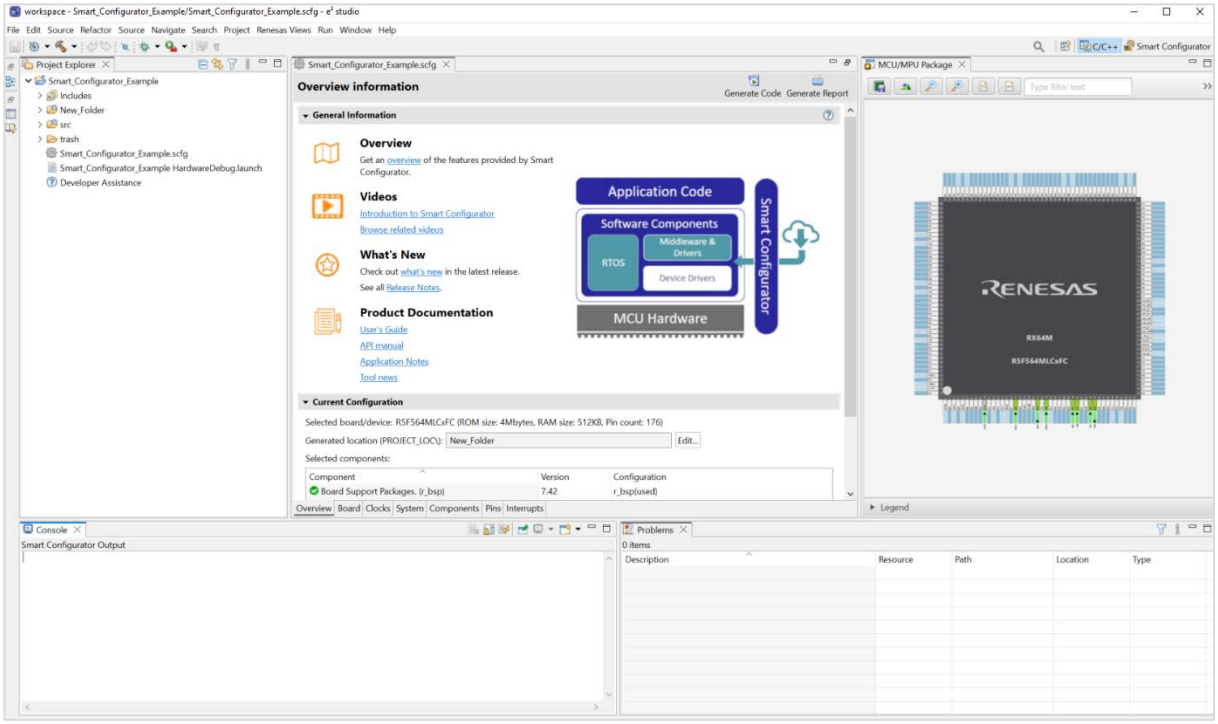


Figure 2-7 Smart Configurator Perspective

2.2 Create a RTOS project

The following describes the procedure for creating a RTOS project.

Note: For FreeRTOS project supported devices and Renesas FreeRTOS, refer to Renesas FreeRTOS related documentation ("chapter 12, Documents for Reference").

(1) Start e² studio and launch a workspace. After starting, select [File] → [New] → [Renesas C/C++ Project] → [Renesas RX] to activate the project creation wizard.

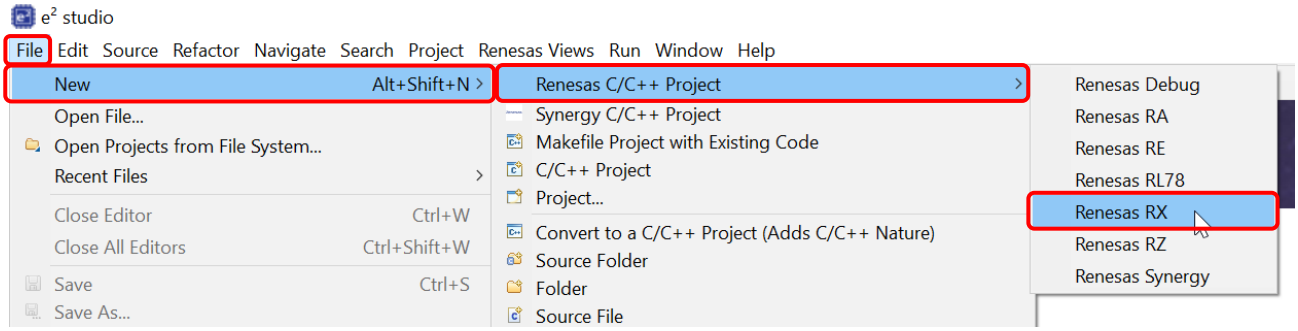


Figure 2-8 Creating a New Project

(2) In the project creation wizard, select [Renesas CC-RX C/C++ Executable Project] or [GCC for Renesas RX C/C++ Executable Project] and click on the [Next] button.

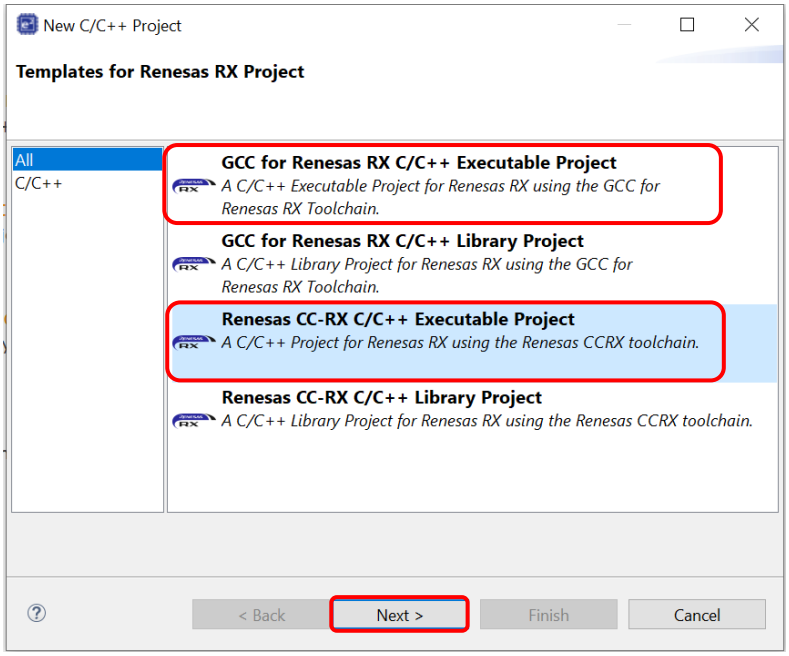


Figure 2-9 Templates for New C/C++ Project Dialog Box

(3) Enter project information. Click on the [Next] button to continue.

(for e.g. CC-RX executable project, Project name: "Smart_Configurator_Example")

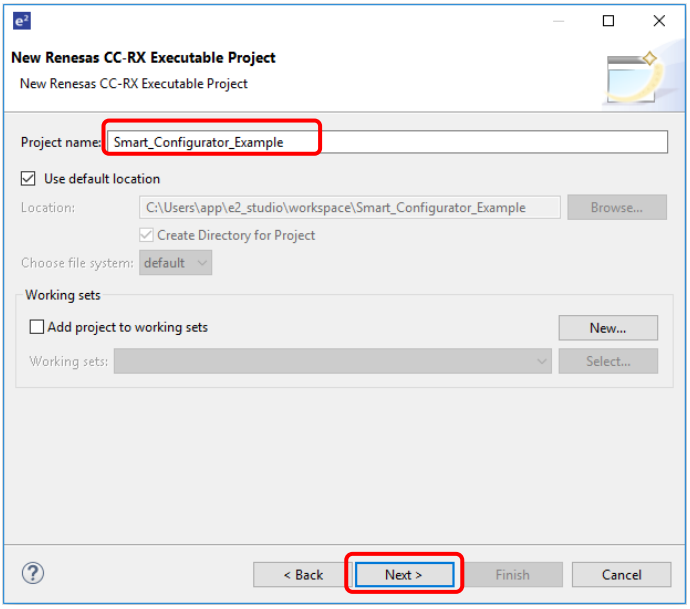


Figure 2-10 Creating a New Renesas CC-RX Executable Project

(4) Select the toolchain and RTOS configuration. (for e.g. RTOS: FreeRTOS (with IoT libraries))

- FreeRTOS (kernel only): create a project with FreeRTOS (kernel only).
- FreeRTOS (with IoT libraries): create a project with FreeRTOS (with IoT libraries).
- Azure RTOS : create a project with Azure RTOS (whole package)
- FreeRTOS (with IoT libraries)(deprecated structure): create a project with FreeRTOS IoT libraries (old structure of AWS).
- RI600V4: create a project with RI600V4 Real-time OS. You should install RI600V4 by yourself before this procedure: https://www.renesas.com/us/en/software-tool/ri600v4-real-time-os-rx-family

If FreeRTOS or Azure RTOS has not been downloaded, go to procedure (5).

If FreeRTOS or Azure RTOS has been downloaded, go to procedure (7).

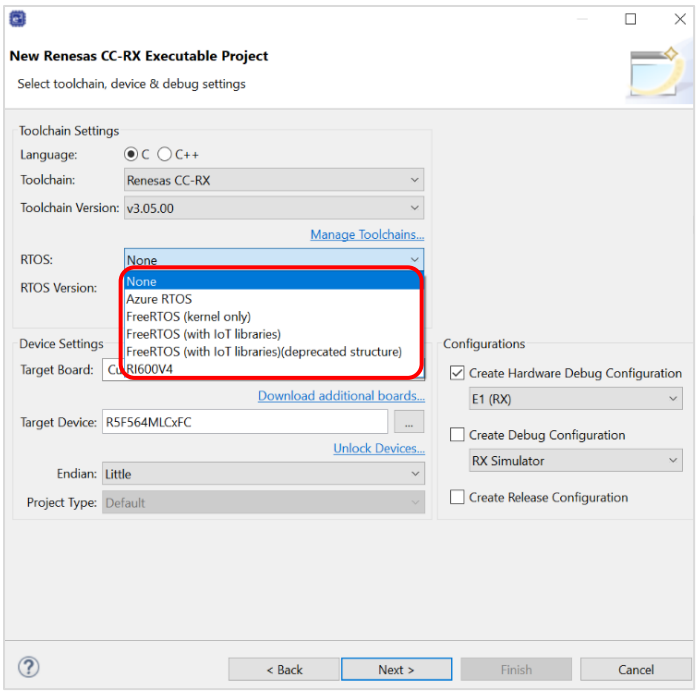


Figure 2-11 Selecting the Toolchain and RTOS Configuration

(5) In the [RTOS Module Download] dialog, select the RTOS package and click [Download].

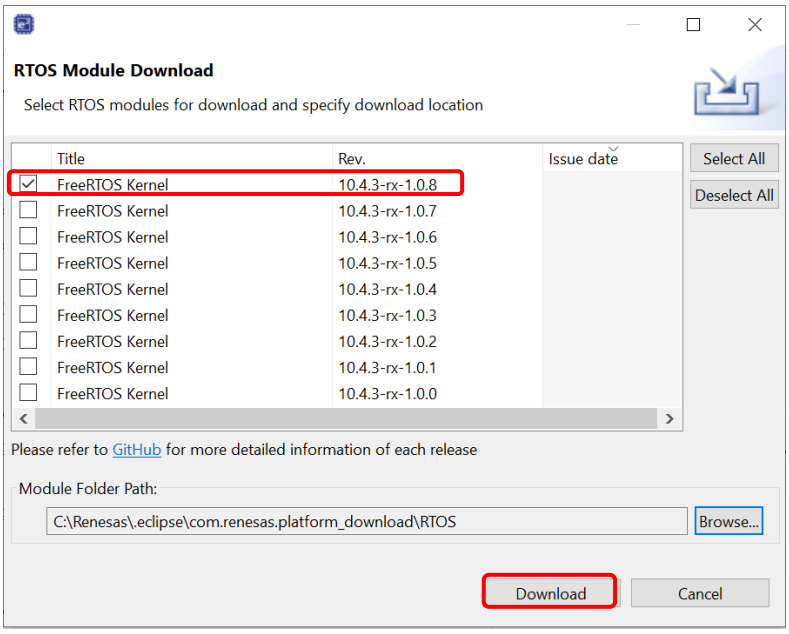


Figure 2-12 [RTOS Module Download] dialog

(6) When [End User License Agreement] dialog is displayed, click [Agree].

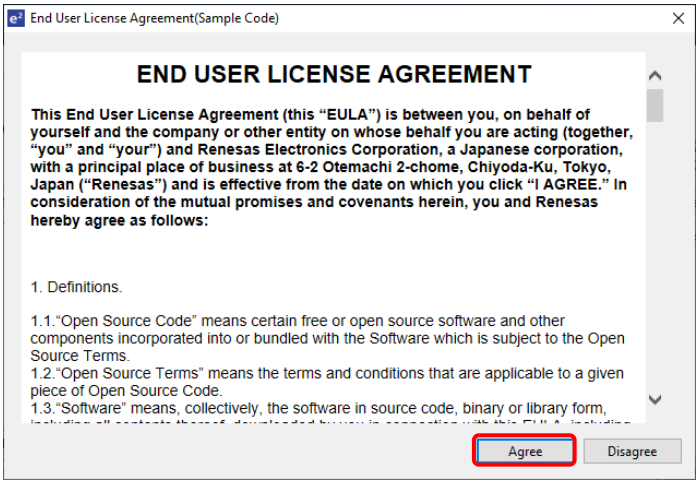


Figure 2-13 [End User License Agreement] dialog

(7) Select the board, device and debug configuration. Only device supported by the RTOS package can be selected. Click on the [Next] button

(for e.g. Target Board: RSKRX64M)

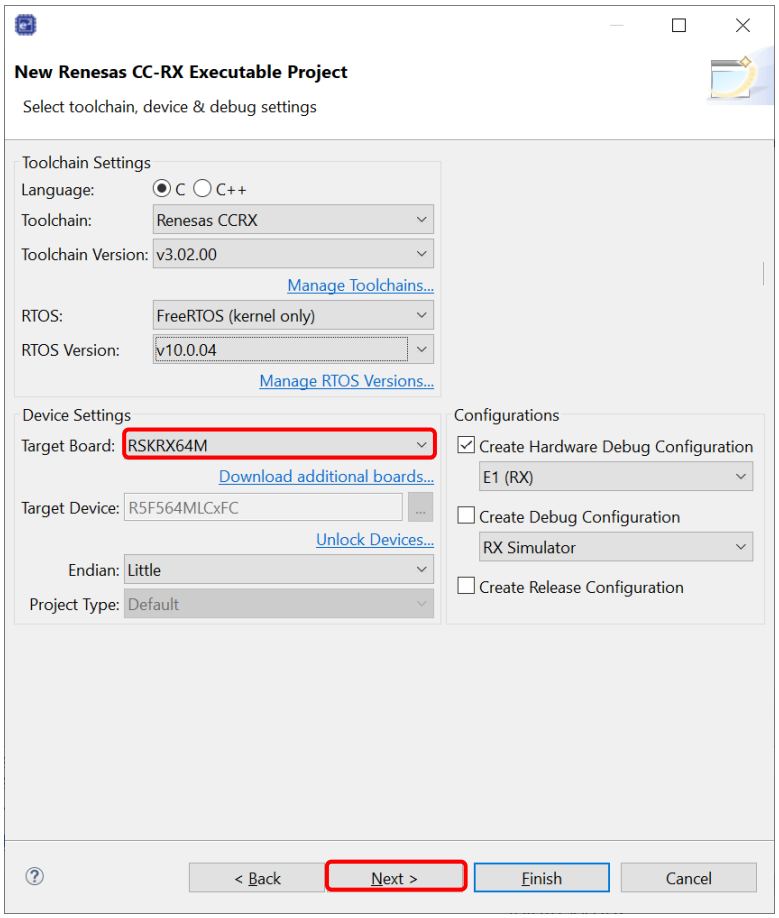


Figure 2-14 Selecting the Device and Debug Configuration

(8) In the [Select Coding Assistant settings] dialog box, select the [Smart Configurator] checkbox and click on the [Finish] button.

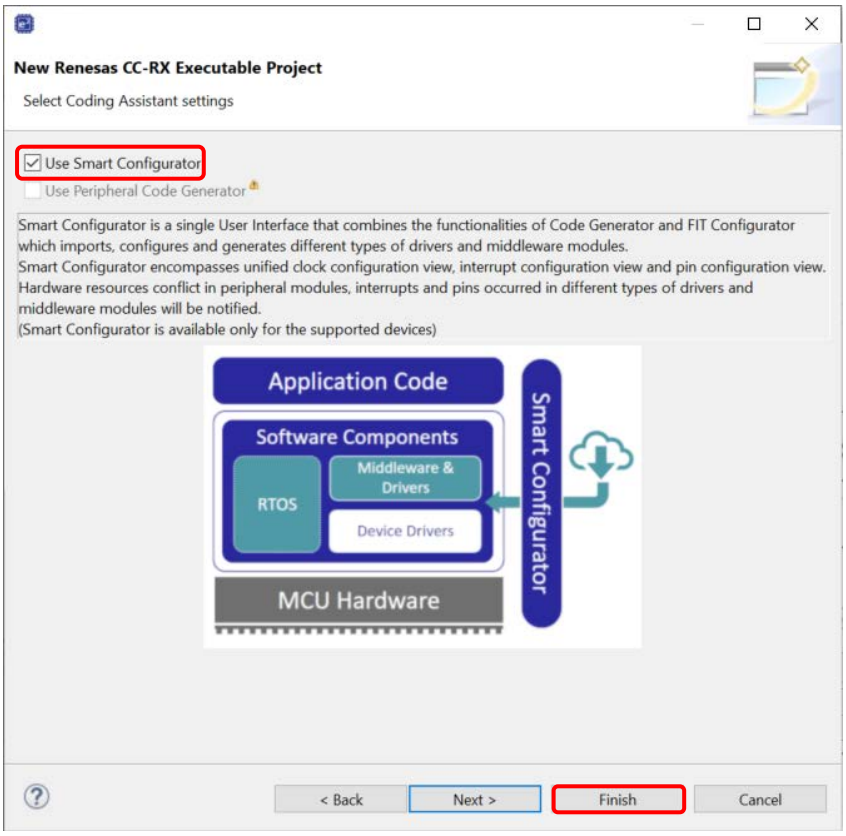


Figure 2-15 Select the Coding Assistant Tool

(9) Wait for completion of project creation.

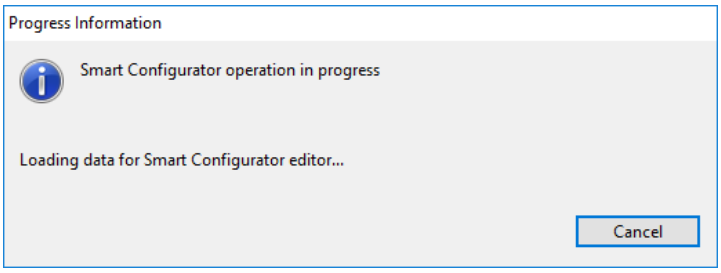


Figure 2-16 Progress of Project creation

(10) After a new C Project is successfully created, the project will be opened in the Smart Configurator perspective.

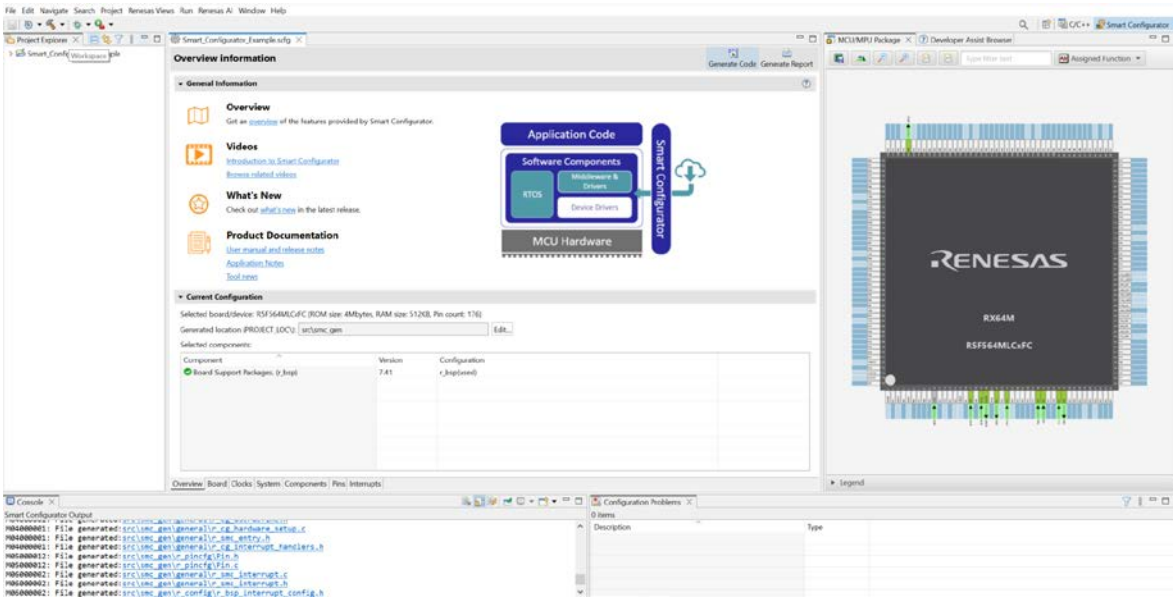


Figure 2-17 Smart Configurator Perspective

2.3 Create a Blinky project.

The following describes the procedure for creating a project using sample project.

(1) Start e² studio and launch a workspace. After starting, select [File] → [New] → [Renesas C/C++ Project] → [Renesas RX] to activate the project creation wizard.

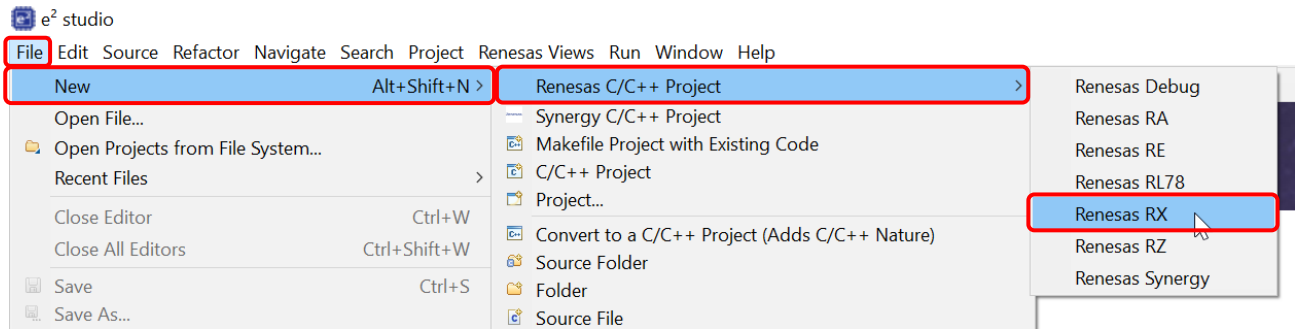


Figure 2-18 Creating a New Project

(2) In the project creation wizard, select [Renesas CC-RX C/C++ Executable Project] or [GCC for Renesas RX C/C++ Executable Project] and click on the [Next] button.

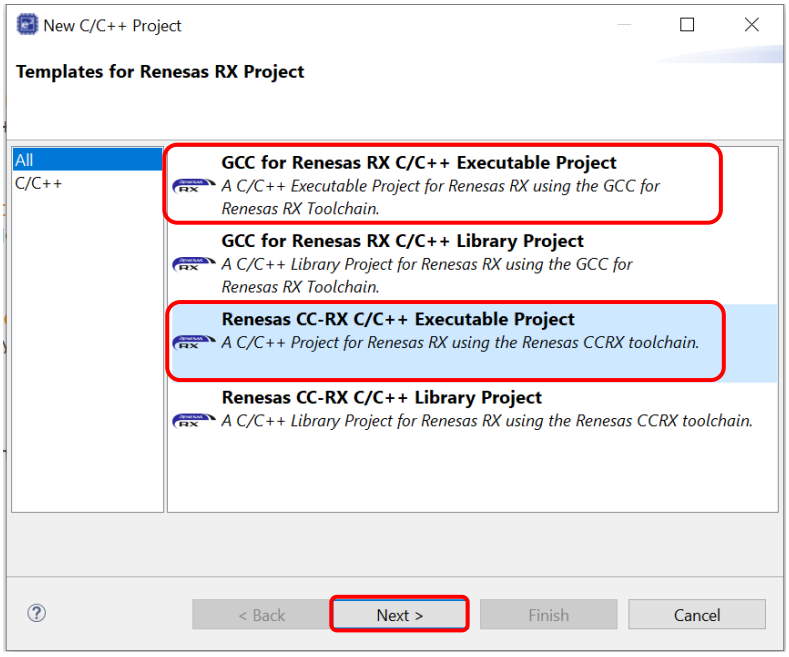


Figure 2-19 Templates for blinky Project Dialog Box

(3) Enter project information. Click on the [Next] button to continue.

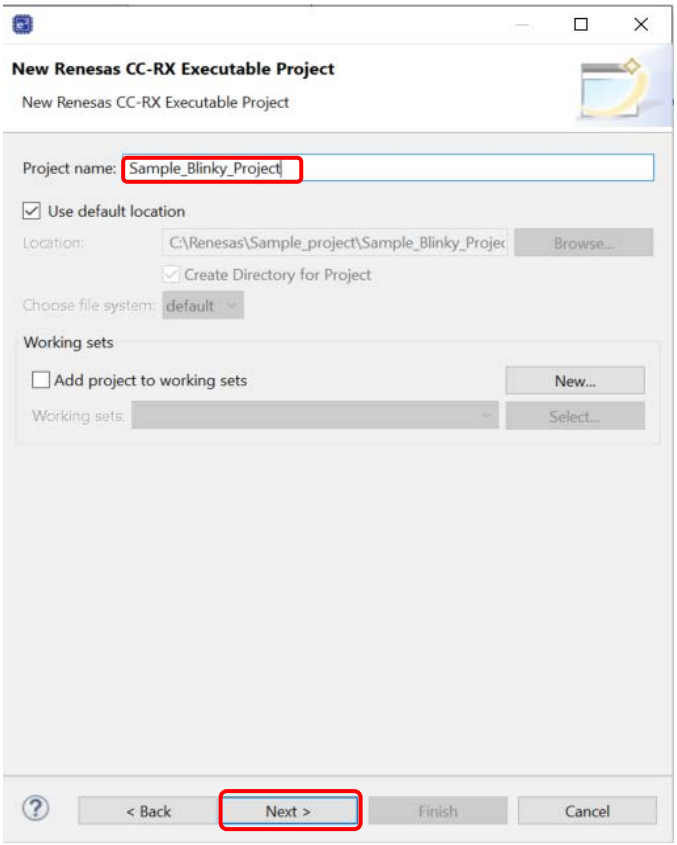


Figure 2-20 Name for blinky Project.

(4) Select C or C++, toolchain, board, device, and debug configuration. Keep RTOS as “None”. Click [Next]

Note: Blinky project is only available in Renesas boards. (for e.g. Target Board: CKRX65N)

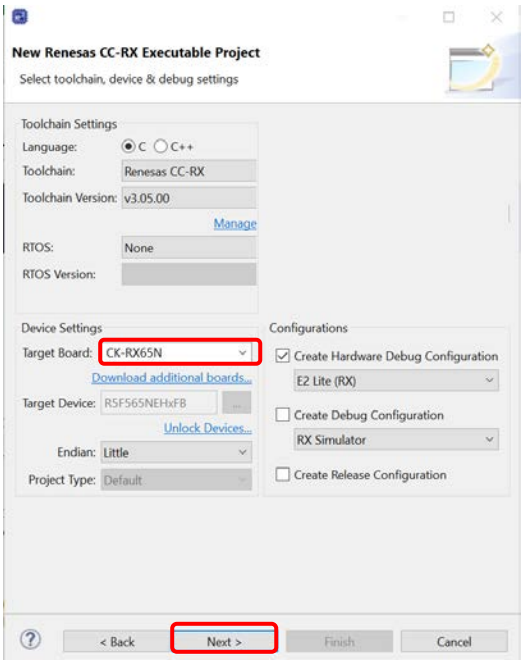


Figure 2-21 Selecting the Toolchain, Device, and Debug Configuration

(5) In the [Select Coding Assistant settings] dialog box, select the [Smart Configurator] checkbox and click on the [Next] button.

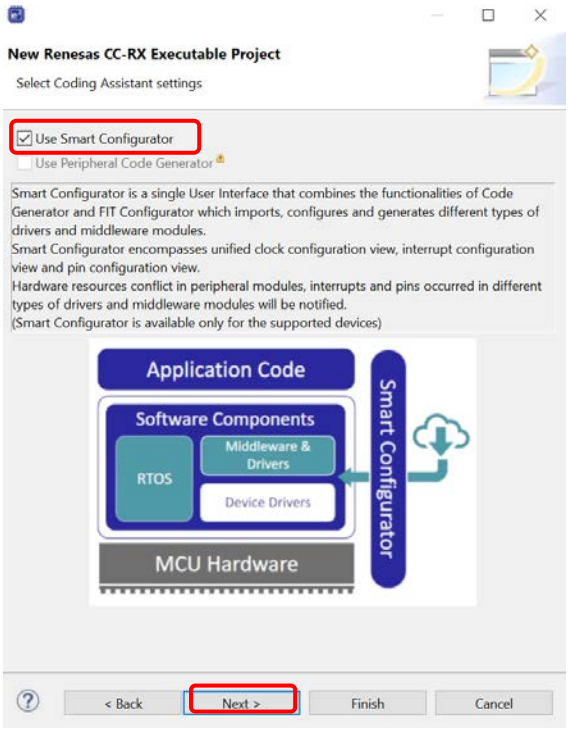


Figure 2-22 Selecting the Coding Assistant Tool

(6) In the [Project template selection] page, select the [Bare Metal - Blinky] checkbox and click on the [Finish] button.

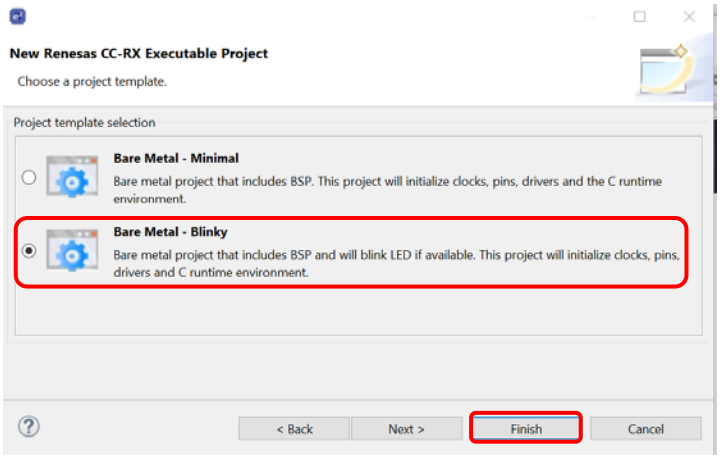


Figure 2-23 Selecting the project template

(7) After the Project is successfully created, the project will be opened in the Smart Configurator perspective.

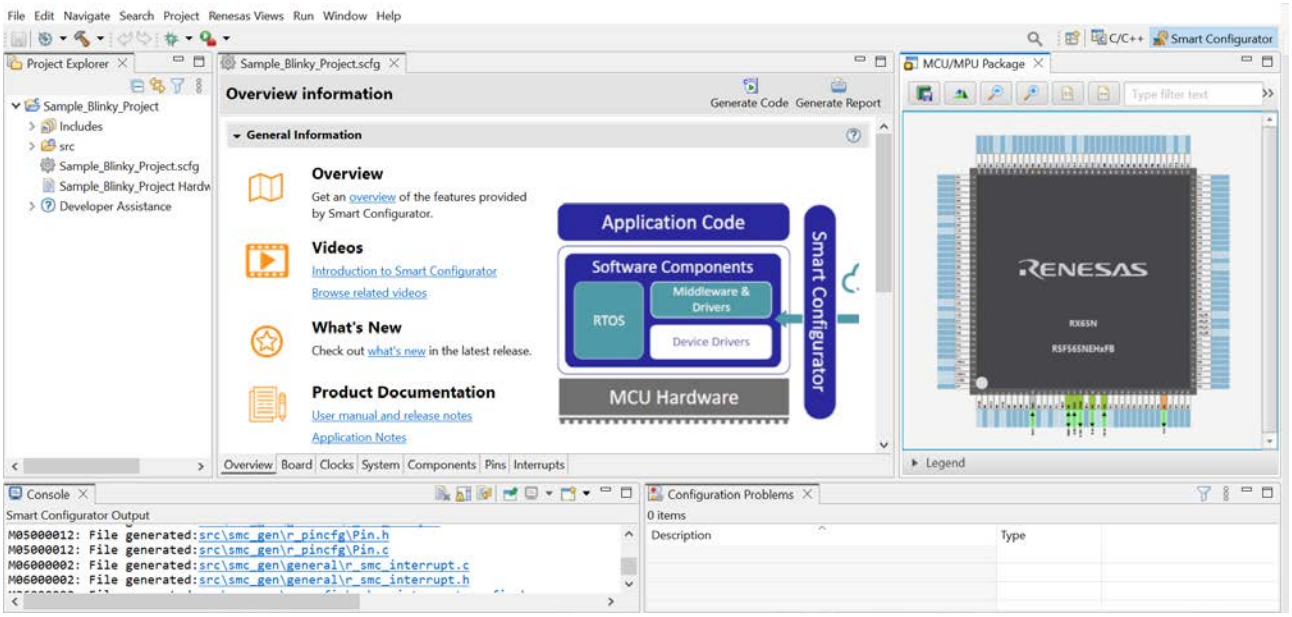


Figure 2-24 Smart Configurator Perspective

(8) Go to the component page, existing components and setting can be found.

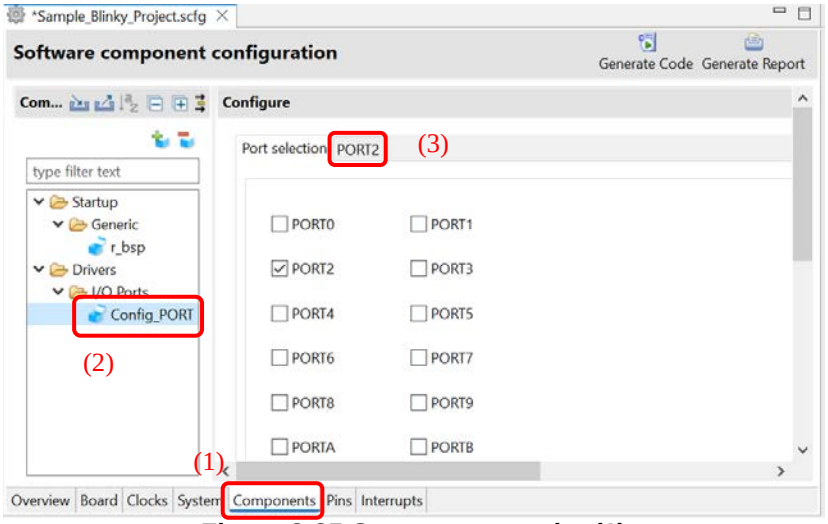


Figure 2-25 Component setting(1)

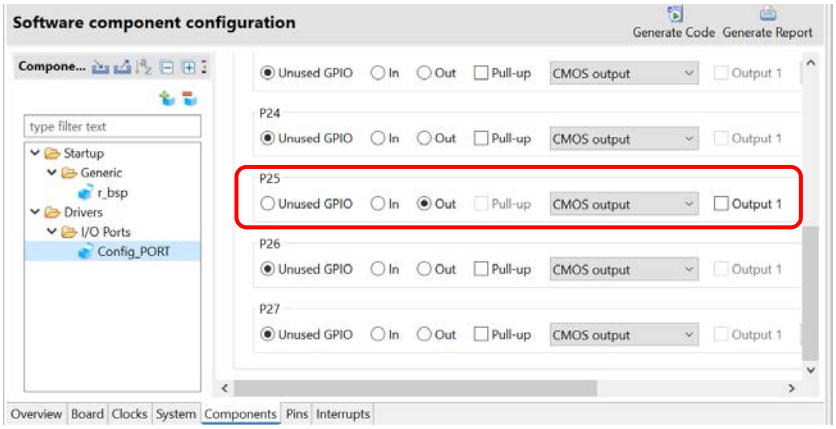


Figure 2-26 Component setting(2)

(9) Click [src] folder and double click [Sample_Blinky_Project.c] file in Project Explore to open the main function. Sample code will be displayed.

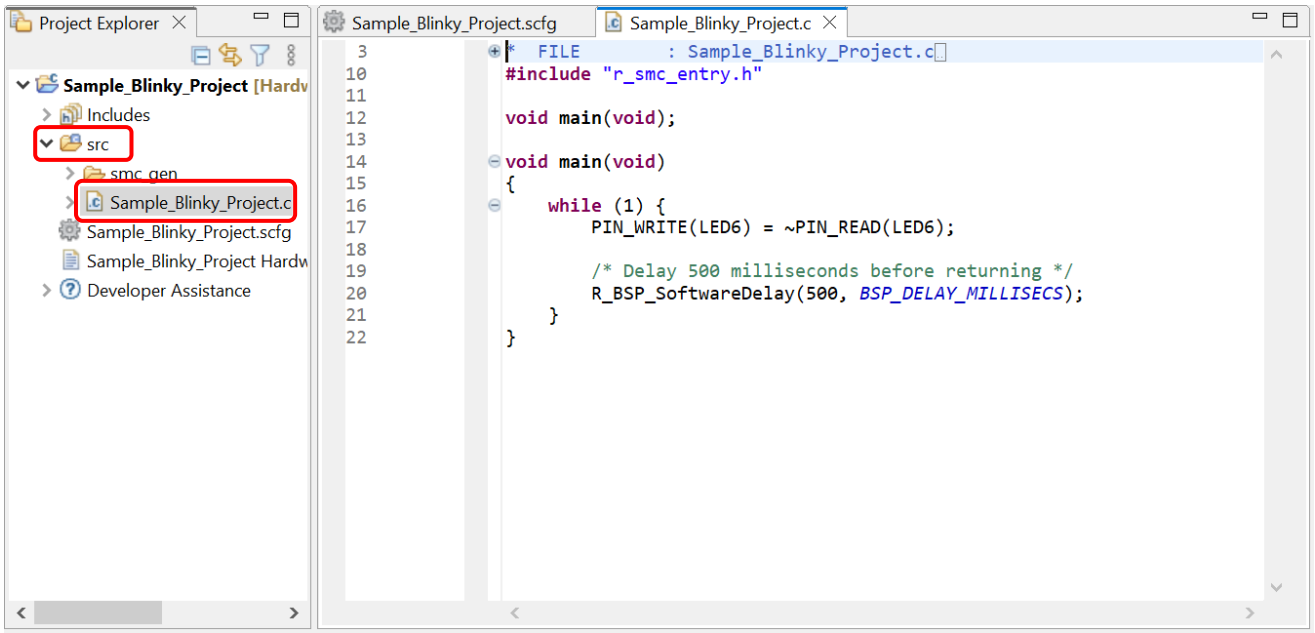


Figure 2-27 Main function

(10) By connecting computer with target board, user can build this project and debug it on target board to toggle LEDs.

3. Operating the Smart Configurator

3.1 Displaying the Smart Configurator Perspective

To fully utilize Smart Configurator features, ensure that the Smart Configurator perspective is opened. If it is not opened, select the perspective icon in the upper right corner of the e² studio window:

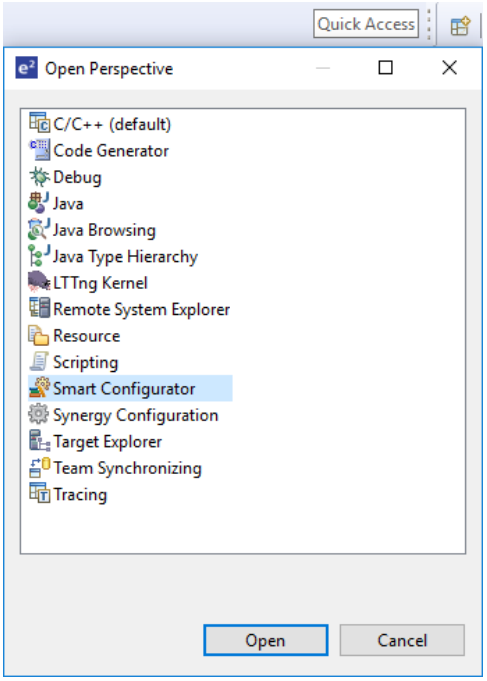


Figure 3-1 Opening the Smart Configurator Perspective

3.2 Procedure for Operations

Figure 3-2 shows the procedure for using the Smart Configurator to set up peripheral modules and build the project with the e² studio. Refer to the related documents on the e² studio for the operation of the e² studio.

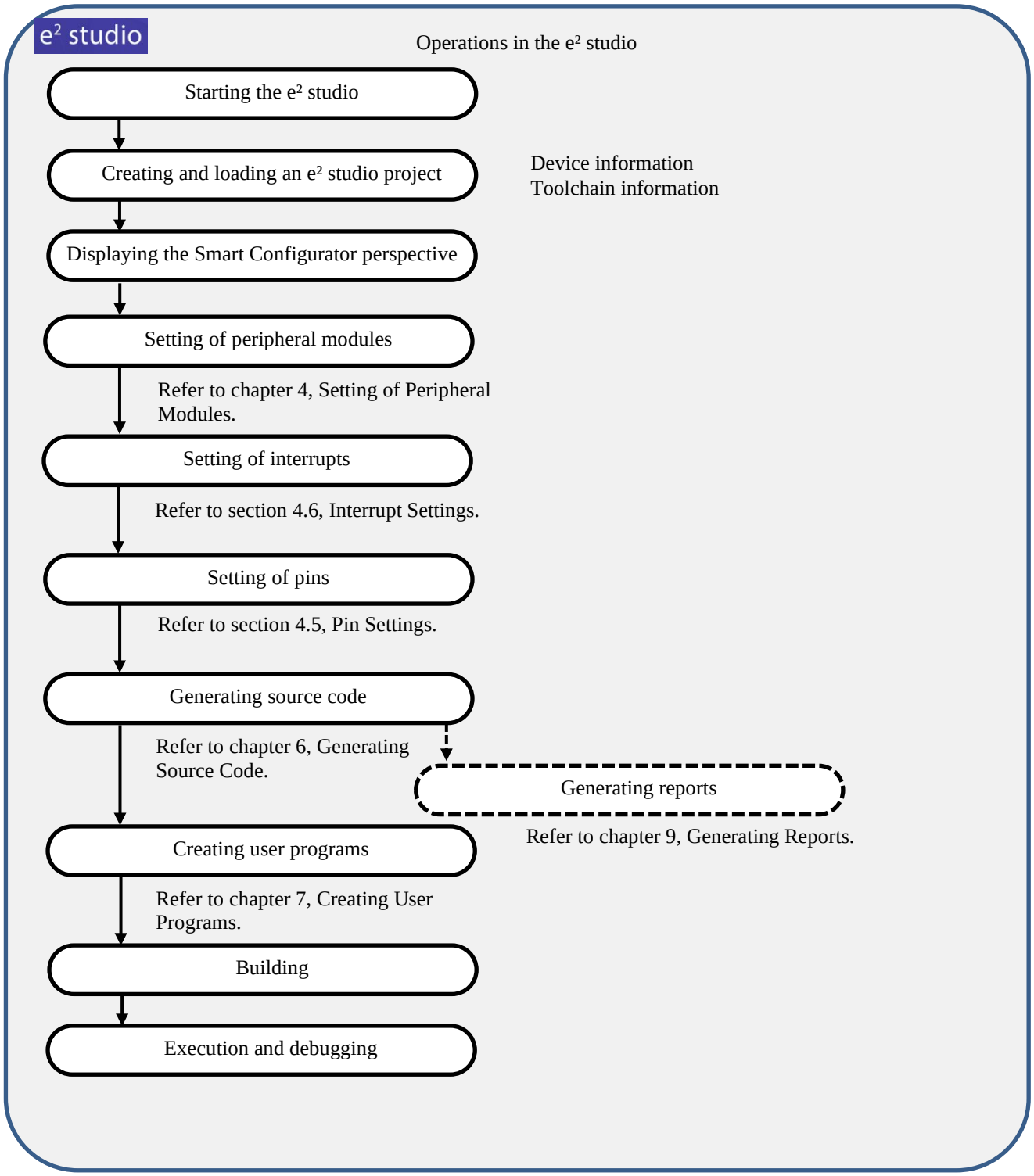


Figure 3-2 Procedure for Operations

3.3 File to be Saved as Project Information

The Smart Configurator saves the setting information such as the target MCU for the project, build tool, peripheral modules, and pin functions in a project file (*.scfg), and refers to this information.

The project file from the Smart Configurator is saved in “project name.scfg”, which is at the same level as the project file (.project) of the e² studio.

3.4 Window

The configuration of the Smart Configurator perspective is shown in Figure 3-3, Smart Configurator Perspective.

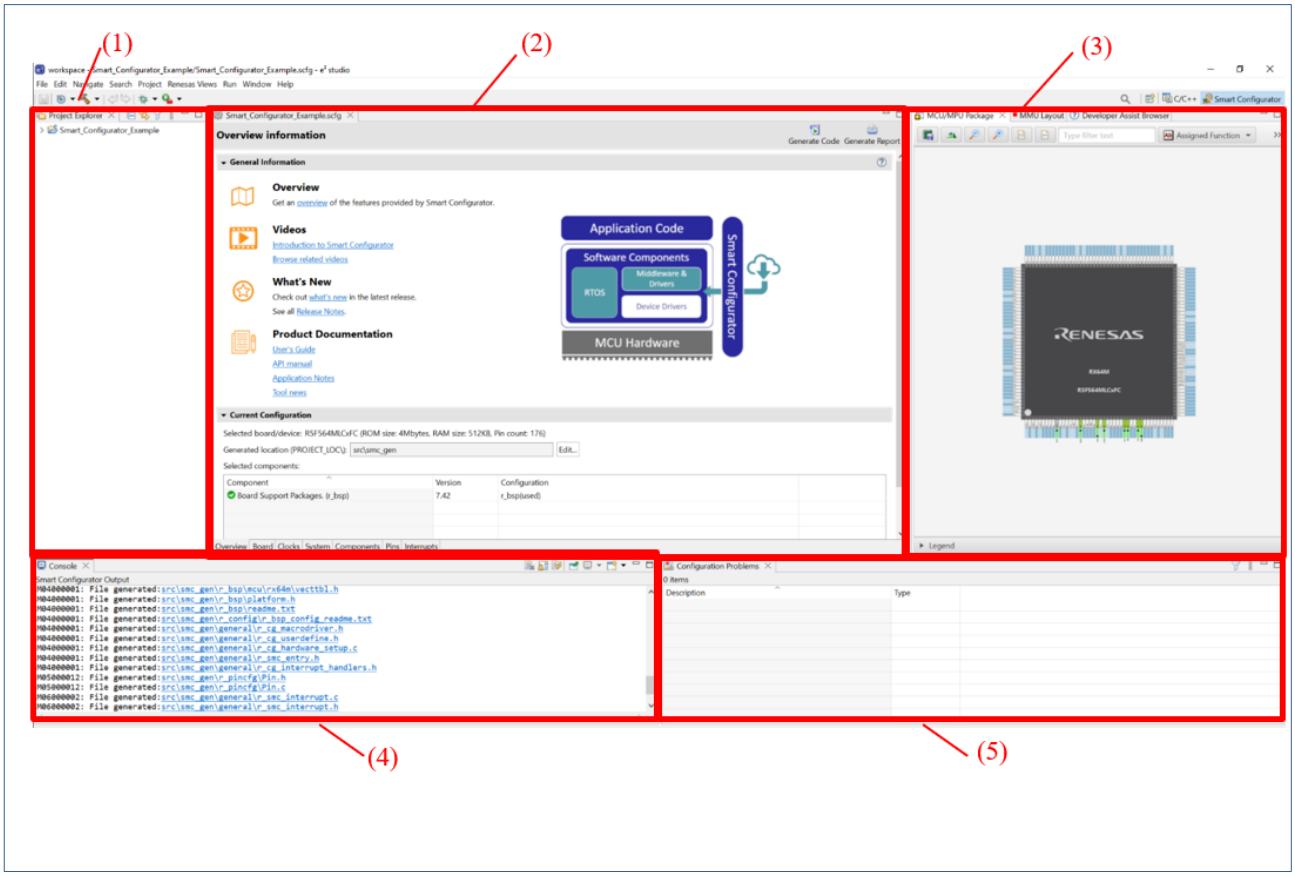


Figure 3-3 Smart Configurator Perspective

1) Project Explorer
2) Smart Configurator view
3) MCU/MPU Package view
4) Console view
5) Configuration Problems view

3.4.1 Project Explorer

The structure of the folders in the project is displayed in a tree form.

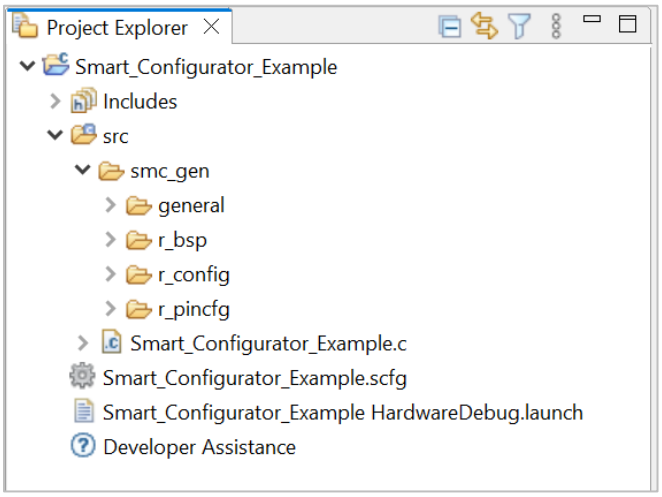


Figure 3-4 Project Explorer

When the Project Explorer is not opened, select [Window] → [Show View] → [Other] from the e² studio menu and select [General] → [Project Explorer] on the opened [Show View] dialog box.

3.4.2 Smart Configurator view

The Smart Configurator view consists of seven pages: [Overview], [Board], [Clocks], [System], [Components], [Pins], and [Interrupts]. Select a page by clicking on a tab; the displayed page will be changed.

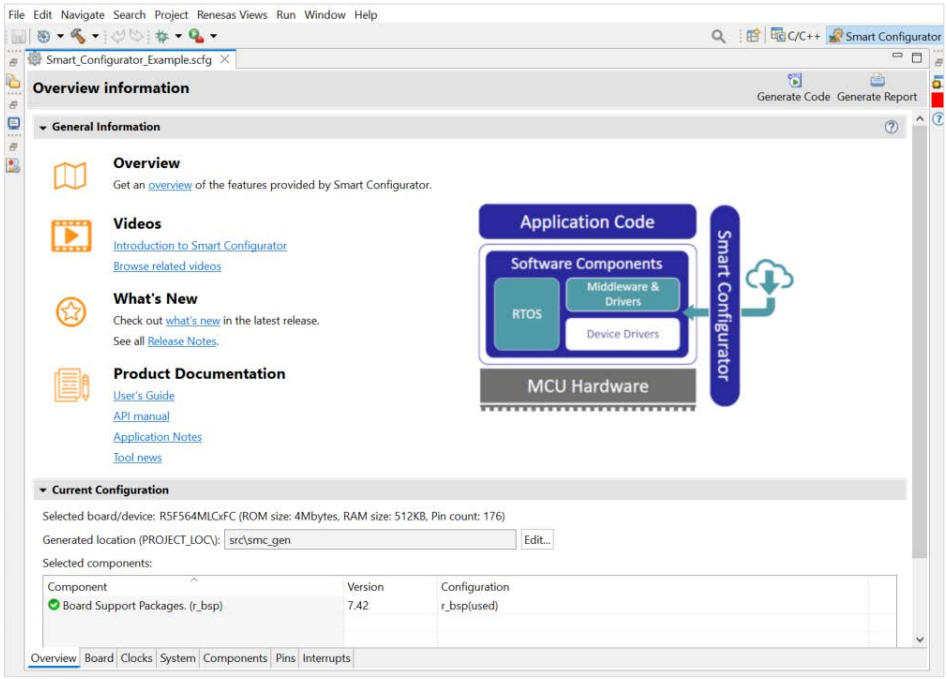


Figure 3-5 Smart Configurator View

When this view is not opened, right-click on the project file (*.scfg) in the Project Explorer and select [Open] from the context menu.

3.4.3 MCU/MPU Package view

The states of pins are displayed on the figure of the MCU/MPU package. The settings of pins can be modified from here.

Three types of package view can be switched between [Assigned Function], [Symbolic Name] and [Board Function]. [Assigned Function] displays the assignment status of the pin setting, and [Board Function] displays the initial pin setting information of the board. [Symbolic Name] displays the symbolic name information of the pins. The initial pin setting information of the board is the pin information of the board selected by [Board:] on the [Board] page (refer to "chapter 4.1.1 Selecting the board" and "chapter 4.5.6 Pin setting using board pin configuration information").

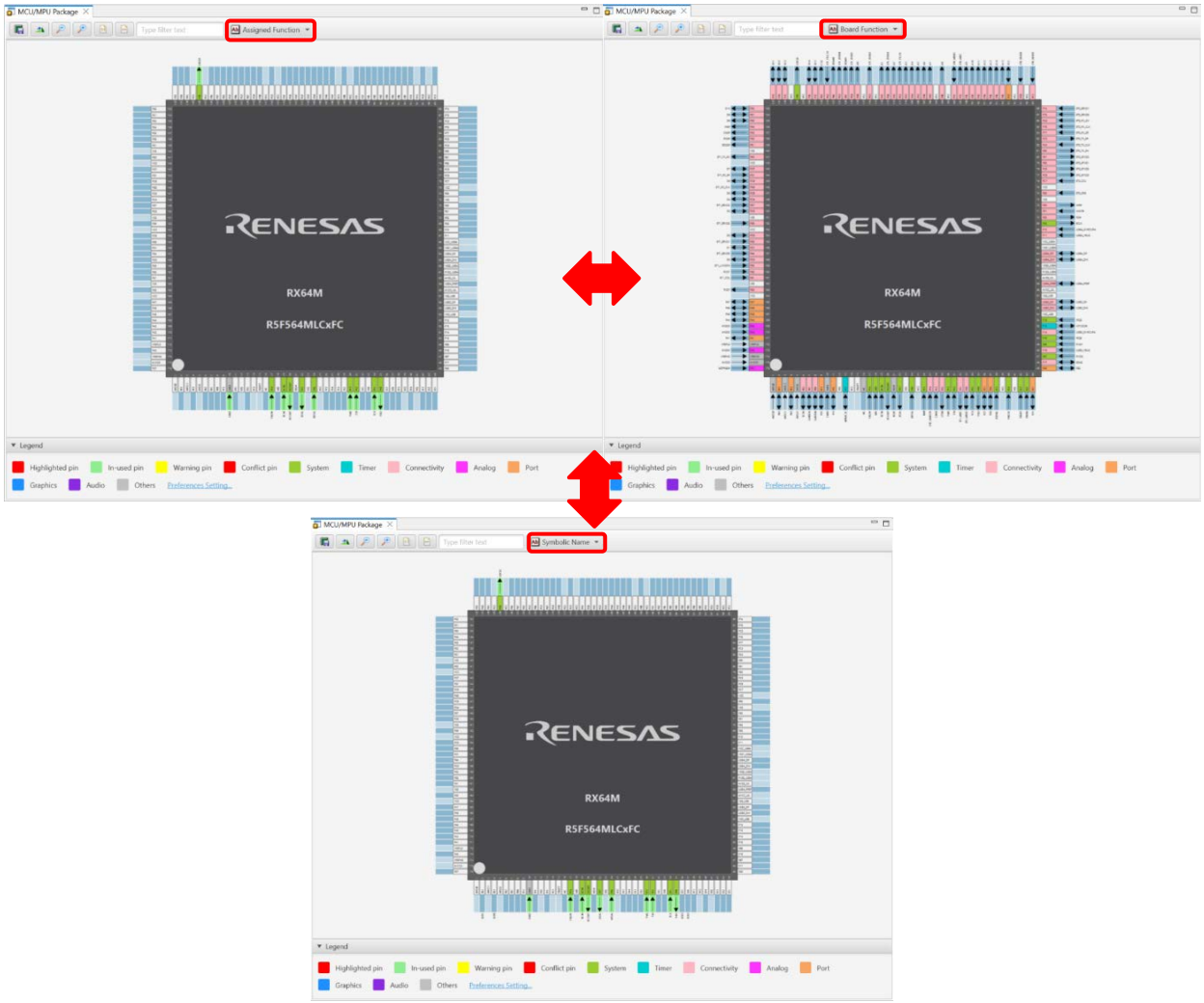


Figure 3-6 MCU/MPU Package View

When this view is not opened, select [Renesas Views] → [Smart Configurator] → [MCU/MPU Package] from the e² studio menu.

3.4.4 Console view

The Console view displays details of changes to the configuration made in the Smart Configurator or MCU/MPU Package view.

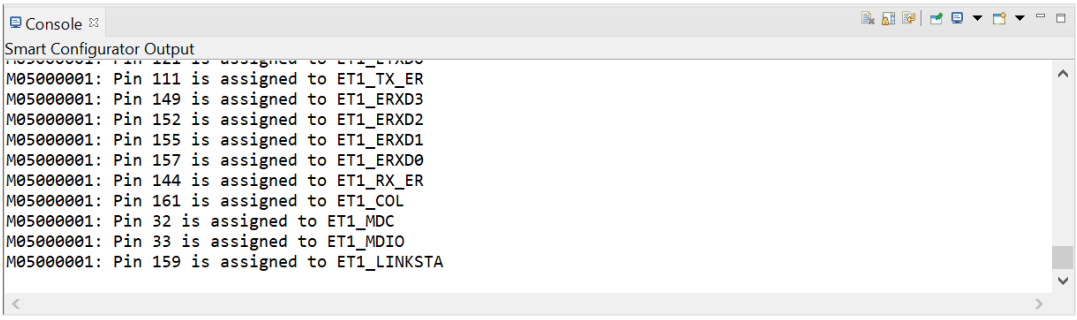


Figure 3-7 Console View

When this view is not opened, select [Window] → [Show View] → [Other] from the e² studio menu and select [General] → [Console] on the opened [Show View] dialog box.

3.4.5 Configuration Problems view

The Configuration Problems view displays the details of conflicts between pins.

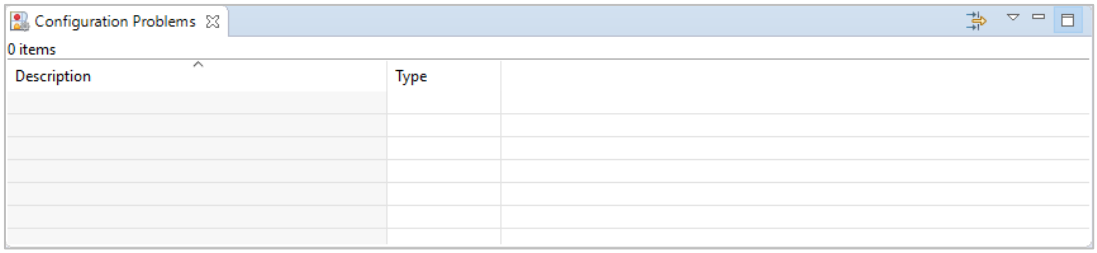


Figure 3-8 Configuration Problems View

When this view is not opened, select [Renesas Views] → [Smart Configurator] → [Configuration Problems] from the e² studio menu.

4. Setting of Peripheral Modules

You can select peripheral modules from the Smart Configurator view.

4.1 Board Settings

You can change the board and device on the [Board] tabbed page.

4.1.1 Selecting the board

Click on the [...] button to select a board.

By selecting a board, the following settings can be changed at one time.

- Pin assignment (Initial pin setting)
- Frequency of the main clock
- Frequency of the sub-clock
- Target device

The board setting information is defined in the Board Description File (.bdf).

The .bdf file of Renesas made board (for e.g. Renesas Starter Kit) can be downloaded from website and imported.

In addition, by downloading the .bdf file provided by the alliance partner from website and importing it, it is possible to select alliance partner boards.

Depending on the board selected, the device will change, device change is reflected to the target device of e² studio project. It is the same with the procedure of " chapter 4.7 MCU migration feature".

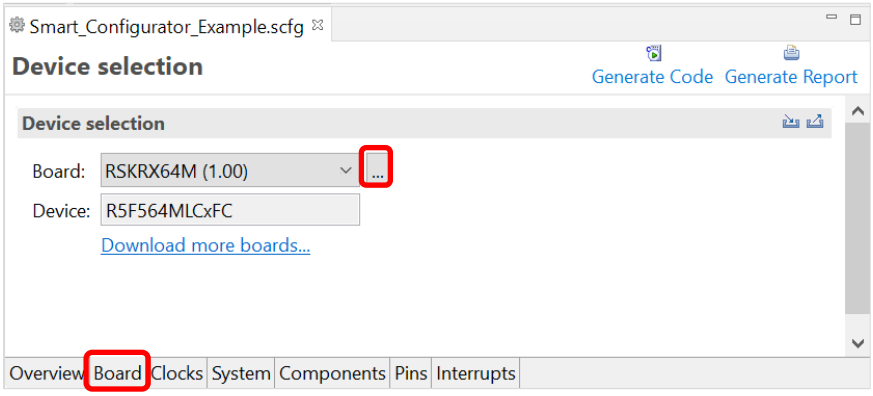


Figure 4-1 Selecting the Board

4.1.2 Exporting board settings

Follow the procedure below to export the board settings.

(1) Click on the [(Export board setting)] button on the [Board] tabbed page.

(2) Select the output location and specify a name (Display Name) for the file to be exported.

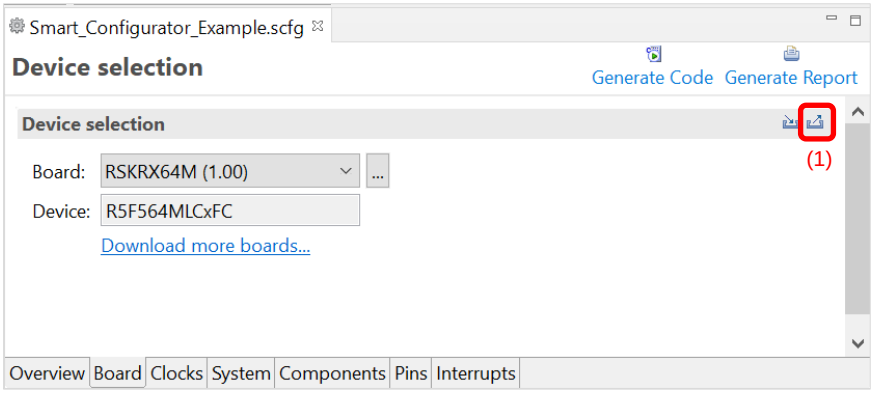


Figure 4-2 Exporting Board Settings (bdf Format)

4.1.3 Importing board settings

Follow the procedure below to import board settings.

(1) Click on the [(Import board setting)] button and select a desired bdf file.

(2) The board of the imported settings is added to the board selection menu.

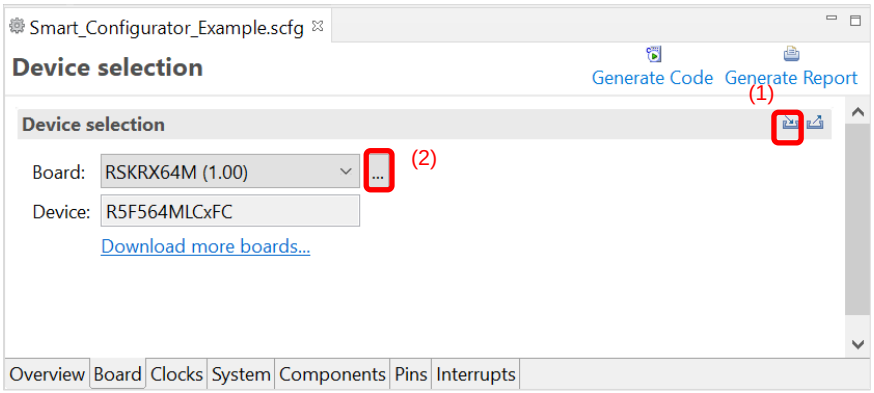


Figure 4-3 Importing Board Settings (bdf Format)

Once a board setting file is imported, the added board is also displayed in the board selection menu of other projects for the same device group.

4.2 Clock Settings

You can set the system clock on the [Clocks] tabbed page. The settings made on the [Clocks] page are used for all drivers and middleware.

Follow the procedure below to modify the clock settings.

(1) Specify the VCC voltage.

(2) Select the clocks required for device operations on the board (the main clock is selected by default).

(3) Specify the frequency of each clock in accordance with the board specifications (note that the frequency is fixed for some internal clocks).

(4) When using the PLL circuit, select the clock source for the PLL.

(5) For the multiplexer symbol, select the clock source for the output clocks.

(6) To obtain a desired output clock frequency, select a frequency division ratio from the drop-down list.

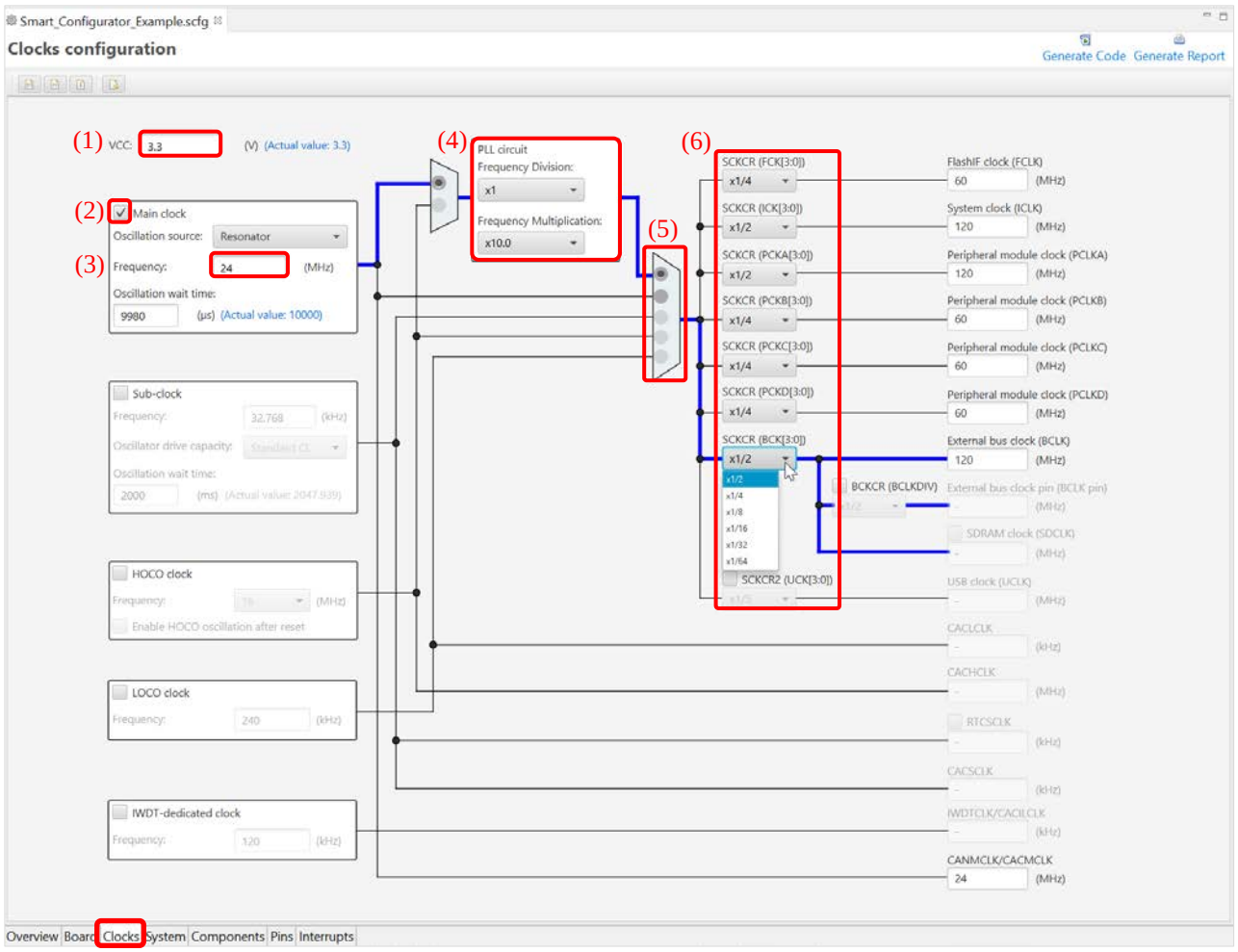


Figure 4-4 Clock Settings

4.3 System Settings

You can set the debug interface pins at [System] tabbed page.

There are 3 types of debug interface available: FINE, JTAG, JTAG (Trace)

You can check the pins configured from Console message or MCU/MPU Package view.

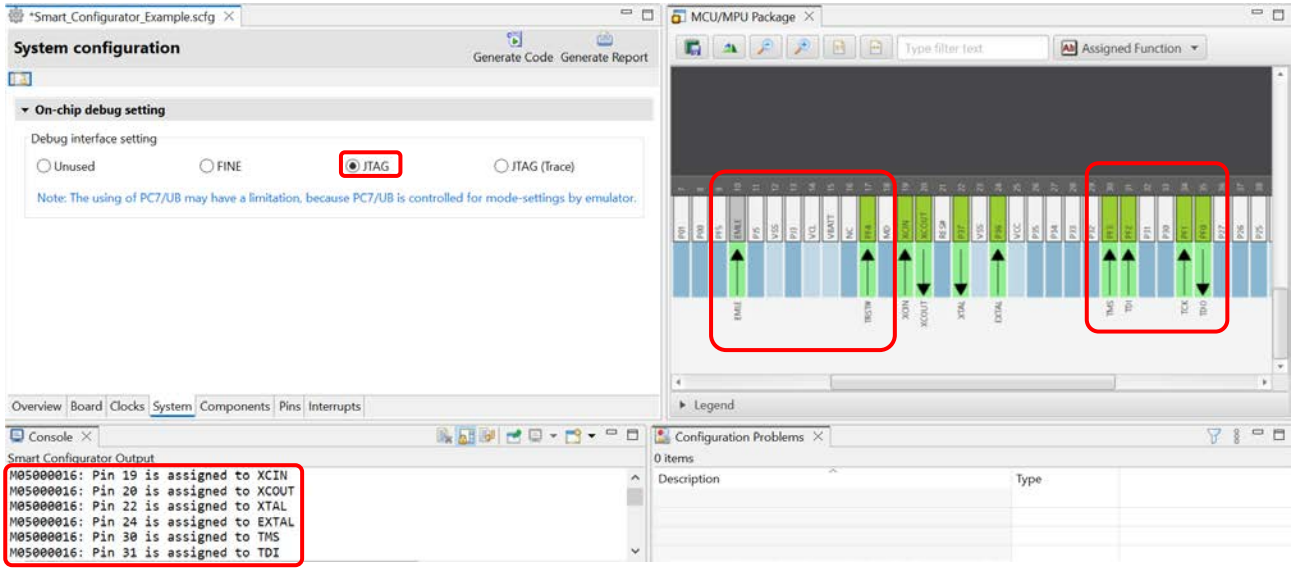


Figure 4-5 Debug Interface Setting at [System] Page

4.4 Component Settings

Drivers and middleware can be combined as software components on the [Components] page. Added components are displayed in the Components tree at the left of the page.

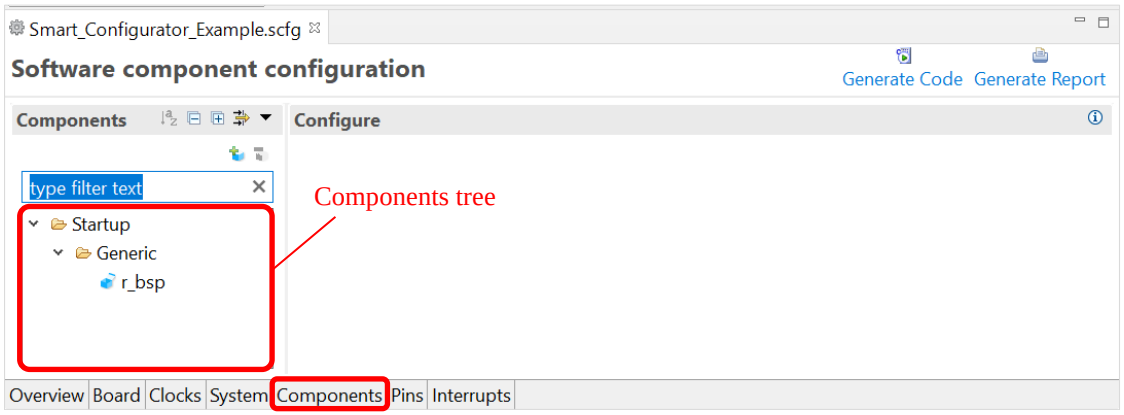


Figure 4-6 [Components] Page

The Smart Configurator supports two types of software components: Code Generator (CG) components and Firmware Integration Technology (FIT) modules.

4.4.1 Adding Code Generator components

The following describes the procedure for adding a component.

(1) Click on the [(Add component)] icon.

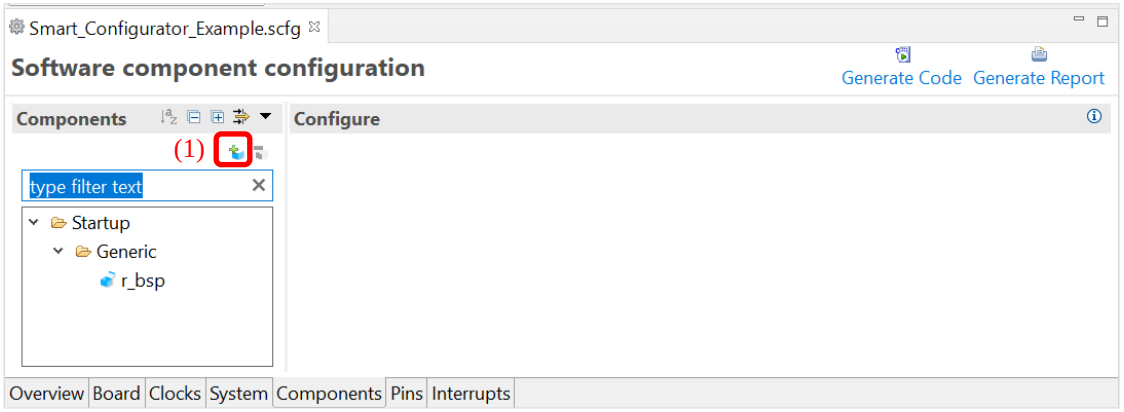


Figure 4-7 Adding a Component

(2) Select a component from the list in the [Software Component Selection] page of the [New Component] dialog box (for e.g. Single Scan Mode S12AD).

(3) Check that [Type] for the selected component is [Code Generator].

(4) Click on [Next].

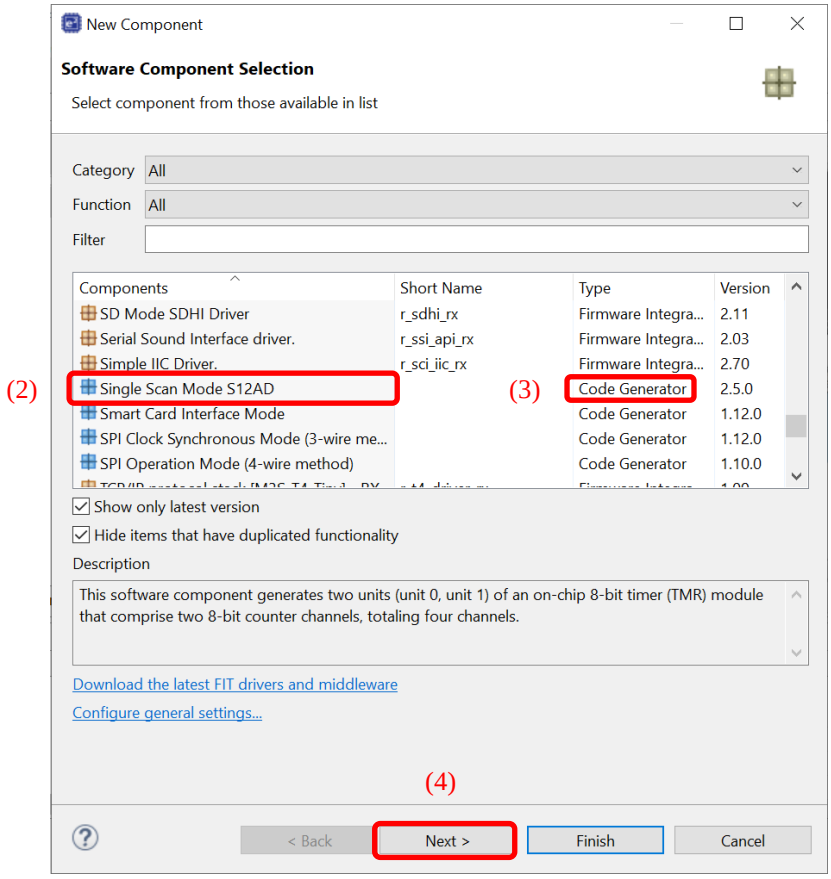


Figure 4-8 Adding a Code Generator Component

(5) Specify an appropriate configuration name in the [Add new configuration for selected component] page of the [New Component] dialog box or use the default name (for e.g. Config_S12AD0).

(6) Select a hardware resource or use the default resource (for e.g. S12AD0).

(7) Click on [Finish].

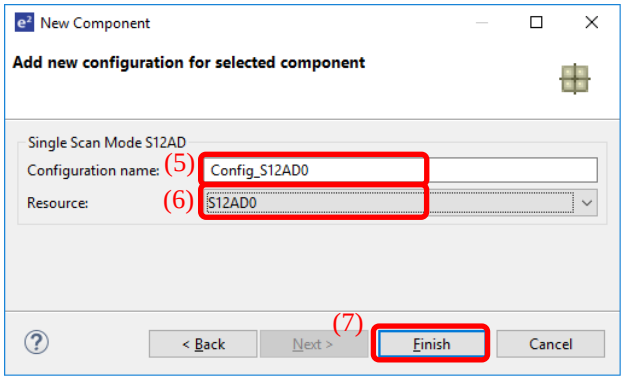


Figure 4-9 Adding a Component

4.4.2 Removing a software component

Follow the procedure below to remove a software component from a project.

(1) Select a software component from the Components tree.

(2) Click on the [(Remove component)] icon.

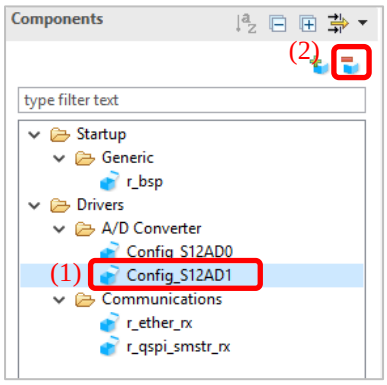


Figure 4-10 Removing a Software Component

The selected software component will be removed from the Components tree.

Multiple components can be selected by pressing [Ctrl] and clicking on components. Click on the [(Remove component)] icon. So multiple components can be removed at the same time.

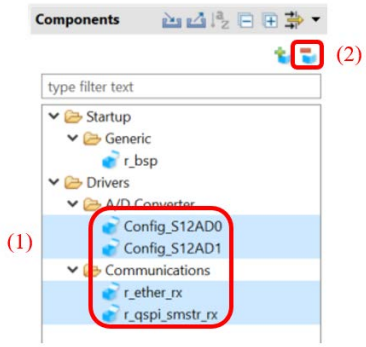


Figure 4-11 Removing Software Components

Source files generated for this component are not removed from the e² studio project tree. After generating source code by clicking [(Generate Code)] icon, the source files generated for removed component will be removed from the e² studio project tree.

4.4.3 Switching between the component view and hardware view

The Smart Configurator also provides a function for adding a new component by directly clicking a node in the Components tree. To use this function, you need to switch the view of the Components tree from the component view to the hardware view.

(1) Click on the [(View Menu)] icon and select [Show by Hardware View]. The Components tree will display the components in a hardware resource hierarchy.

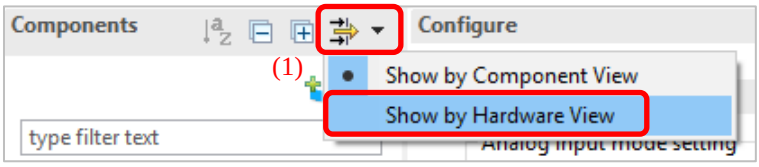


Figure 4-12 Switching to the Hardware View

(2) Double-click on a hardware resource node (for e.g. S12AD1 under 12-bit A/D converter) to open the [New Component] dialog box.

(3) Select a component from the list (for e.g. Single Scan Mode S12AD) to add a new configuration as described in chapter 4.4.1 Adding Code Generator components.

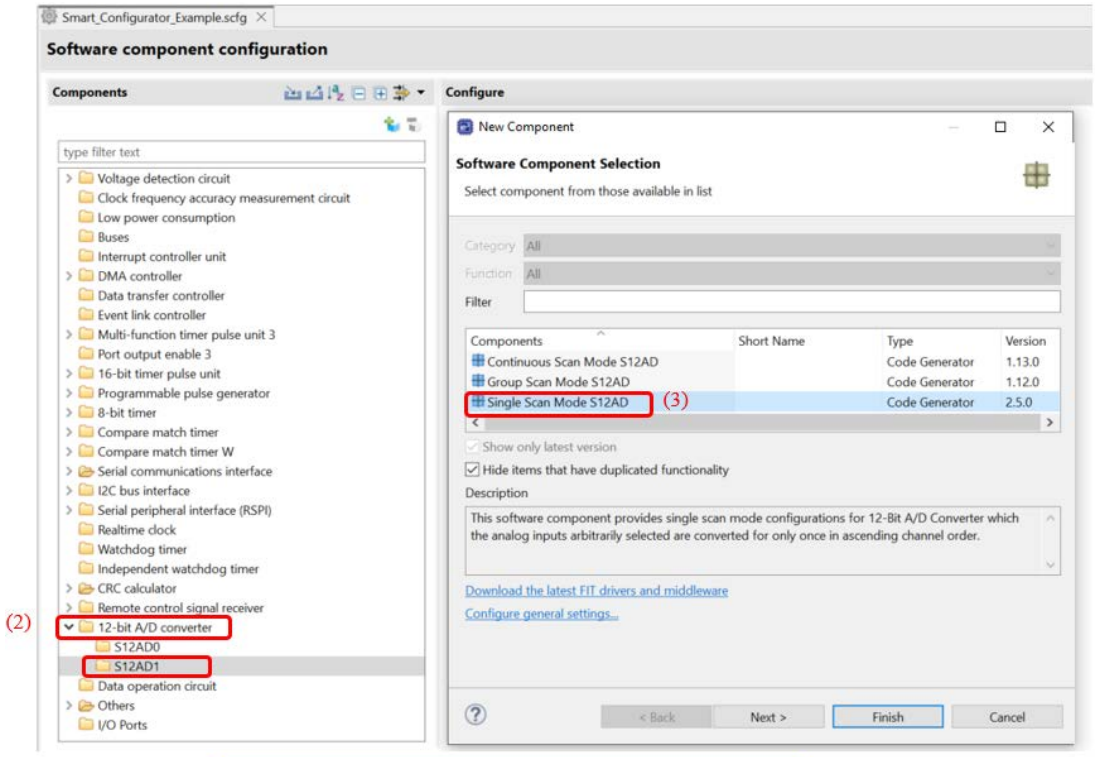


Figure 4-13 Adding a CG Component to the Hardware View

4.4.4 Setting a CG driver

Follow the procedure below to set up a CG configuration.

(1) Select a CG configuration from the Components tree (for e.g. Config_S12AD0).

(2) Configure the driver in the [Configure] panel to the right of the Components tree. The following steps and figure show an example.

a. Select AN000.

b. Select [A/D conversion start trigger pin] under [Conversion start trigger setting].

c. Click on [Advance setting] to expand the view.

d. Select [Discharge] for [Charge setting].

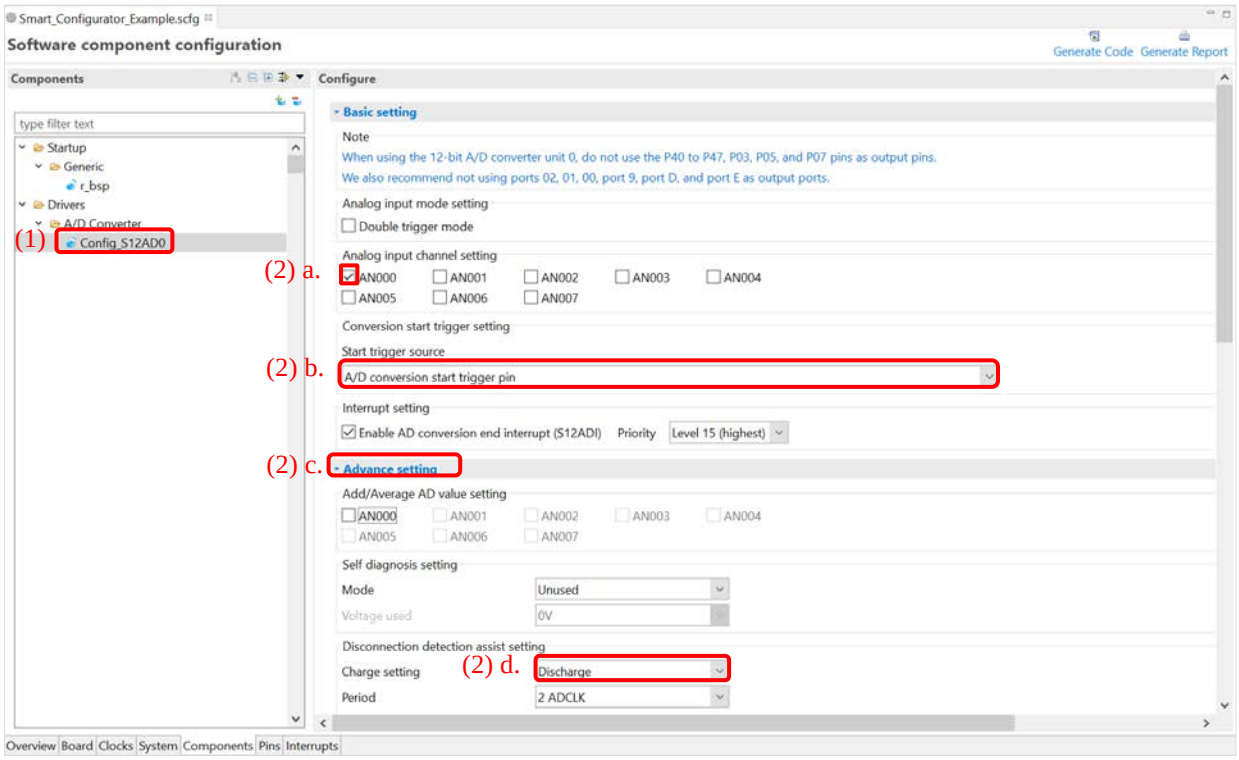


Figure 4-14 Setting of a CG Driver

Generation of a code in accordance with each CG configuration is enabled by default.

Right-clicking on a CG configuration and then selecting the [✓ Generate code] icon changes the icon to [Generate code] and disables code generation for the CG configuration.

To enable code generation again, click on the [Generate code] icon and change it to [✓ Generate code].

Note:

There is an information icon [ⓘ] on the top-right corner of added CG component, it provides component and resource information with tooltip. Whenever hover over the icon using a cursor, the related information message will be displayed as an example in the picture below.

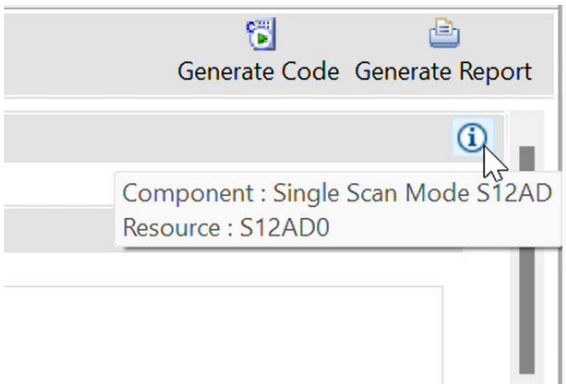


Figure 4-15 Component information icon

4.4.5 Changing the resource for a CG configuration

The Smart Configurator enables you to change the resource for a CG configuration (for e.g. from S12AD0 to S12AD1). Compatible settings can be ported from the current resource to the new resource selected.

Follow the procedure below to change the resource for an existing software component.

(1) Right-click on a CG configuration (for e.g. Config_S12AD0).

(2) Select [Change resource] from the context menu.

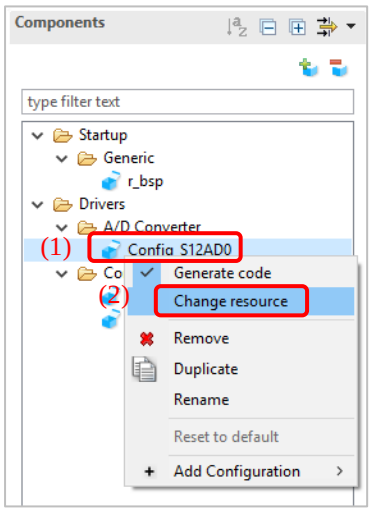


Figure 4-16 Changing the Resource

(3) Select a new resource (for e.g. S12AD1) in the [Resource Selection] dialog box.

(4) The [Next] button will be active; click on it.

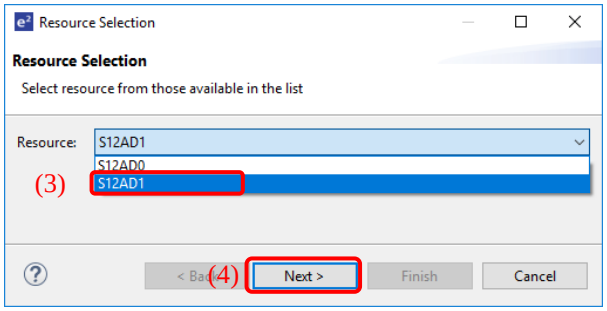


Figure 4-17 Components Page – Selecting a New Resource

(5) Configuration settings will be listed in the [Configuration setting selection] dialog box.

(6) Check the portability of the settings.

(7) Select whether to use the listed or default settings.

(8) Click on [Finish].

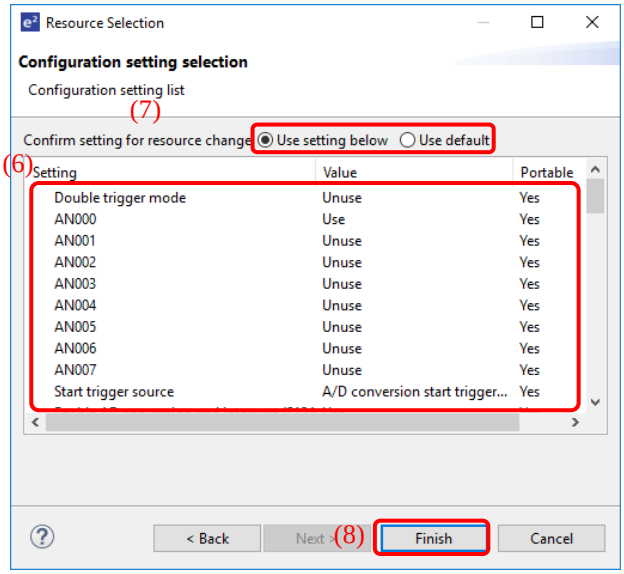


Figure 4-18 Checking the Settings of the New Resource

The resource is automatically changed (for e.g. changed from S12ADI0 to S12ADI1).

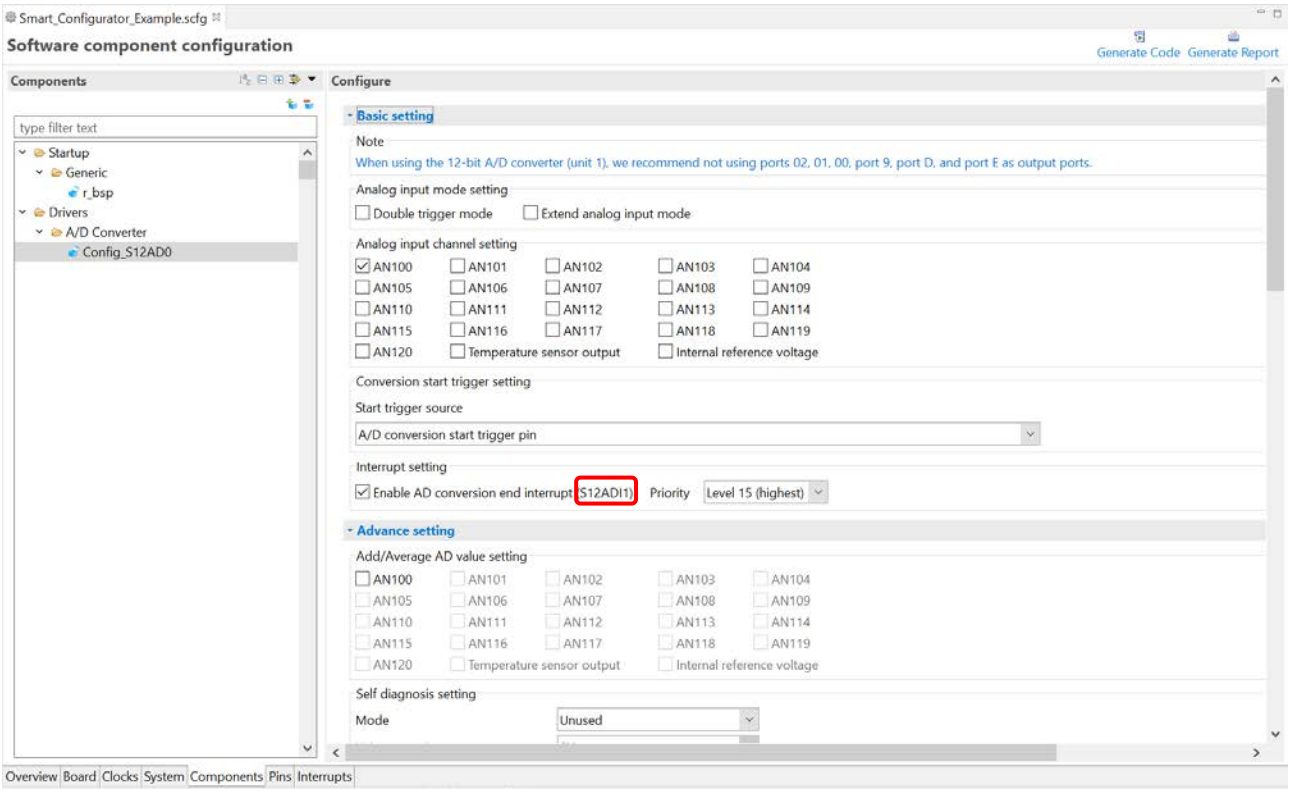


Figure 4-19 Resource Changed Automatically

To change the configuration name, follow the procedure below.

(9) Right-click on the CG configuration.

(10) Select [Rename] to rename the configuration (for e.g. change Config_S12AD0 to Config_S12AD1).

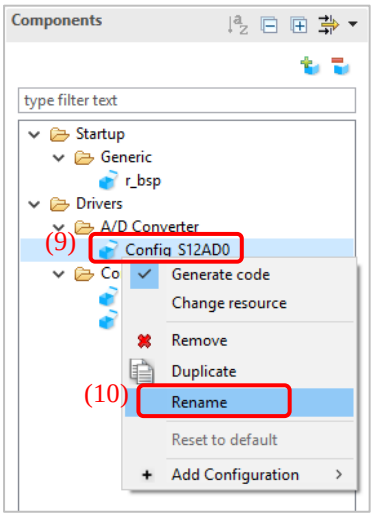


Figure 4-20 Renaming the Configuration

4.4.6 Downloading a FIT module

You need to download a desired FIT driver or middleware from the Renesas Electronics website.

(1) Click on the [(Add component)] icon.

(2) Click the [Download the latest FIT drivers and middleware] link in the [Software Component Selection] page of the [New Component] dialog box to download a FIT module.

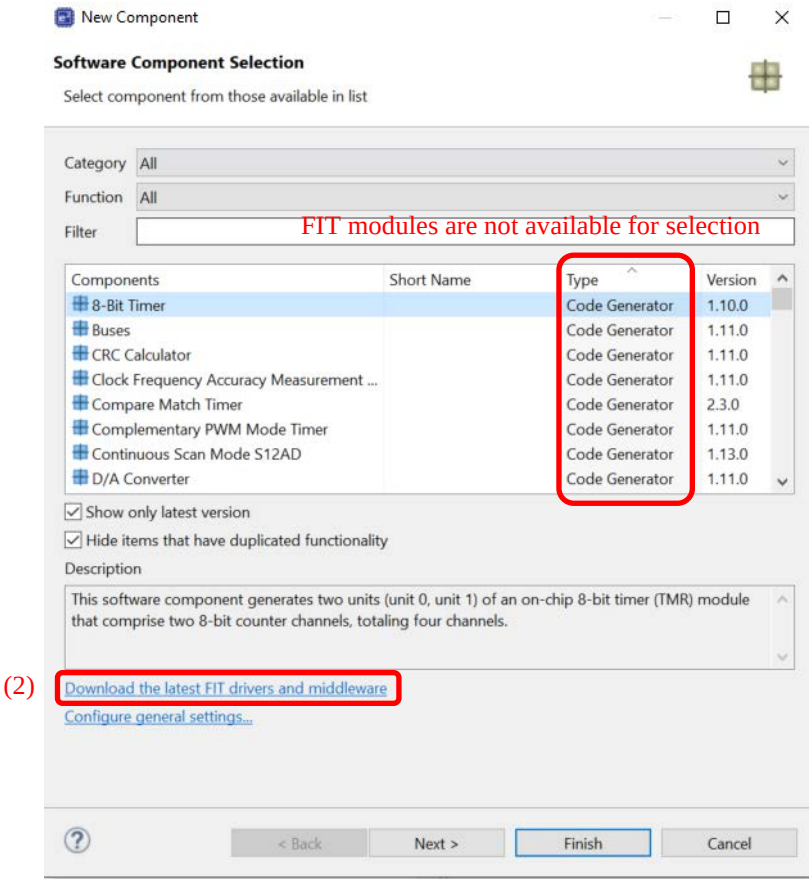


Figure 4-21 Downloading More Software Components

Note: Downloading requires login to "My Renesas". If you have not logged in, the following dialog box will prompt you to log in. To register as a new user, click on the [About My Renesas] button.

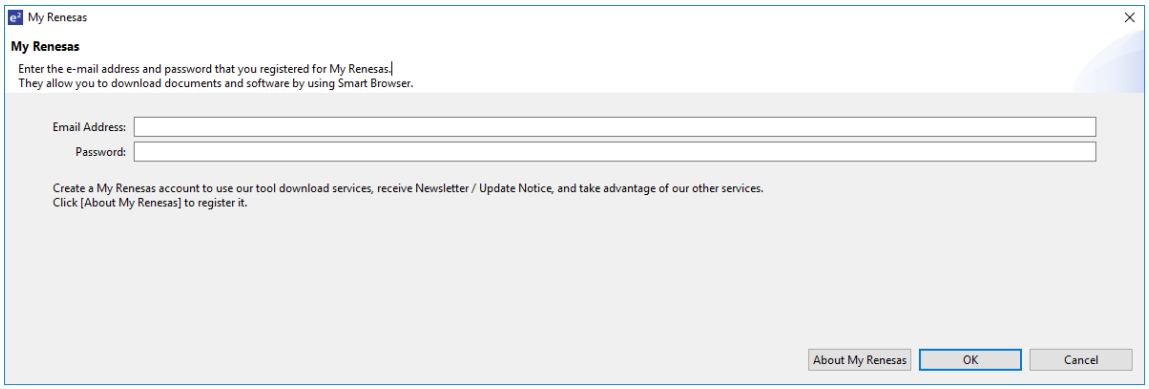


Figure 4-22 Login to My Renesas

(3) Select the checkbox of the required module in the [FIT Module Download] dialog box. If [Show RX Driver Package only] is unchecked, filtering of items is canceled.

(4) Click on [Browse...] to select the location where the downloaded module is to be stored.

(5) Click on [Download] to start downloading the selected FIT module.

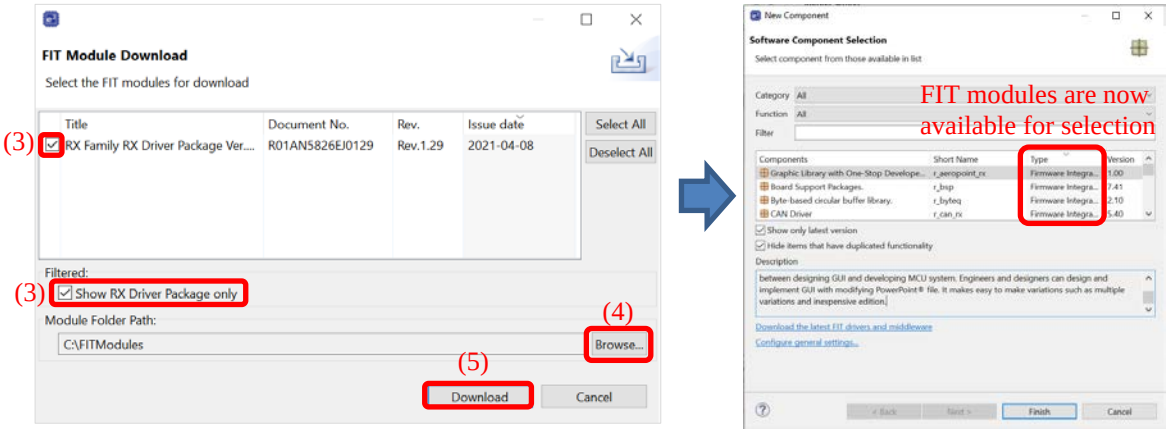


Figure 4-23 Downloading a FIT Module

4.4.7 Adding FIT drivers or middleware

The following describes the procedure for adding FIT drivers or middleware.

(1) Click on the [(Add component)] icon.

(2) Select components from the list in the [Software Component Selection] page of the [New Component] dialog box (for e.g. r_ether_rx and r_qspi_smstr_rx). Two or more components can be selected by clicking with the Ctrl key pressed.

(3) Check that [Type] for the selected components is [Firmware Integration Technology].

(4) Click on [Finish].

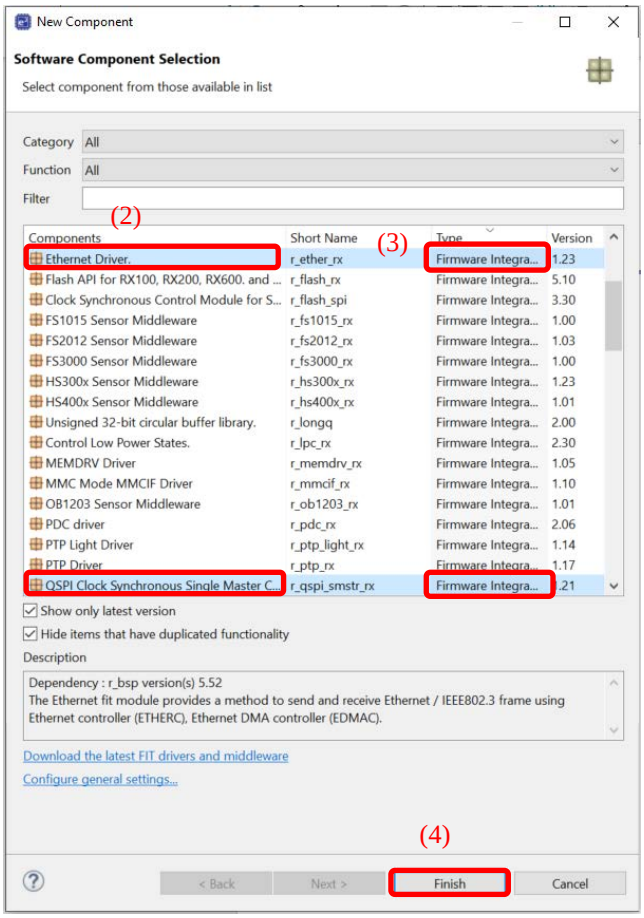


Figure 4-24 Adding FIT Modules

4.4.8 Download and import FIT sample project

When the FIT driver or middleware icon is [ ], you can download the sample project.

(1) Select the FIT driver or middleware of the [ ] icon and select [Download and import sample projects] from the right menu. (for e.g. CMT)

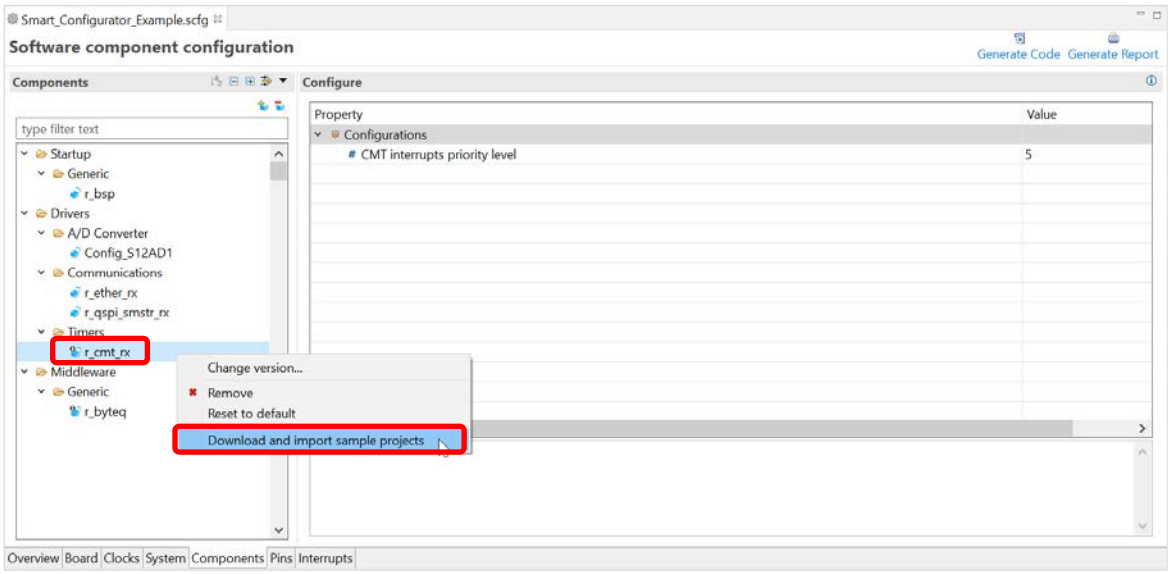


Figure 4-25 Download and import sample projects

(2) The sample code is displayed on the [Application Notes] tab of the Smart Browser, so select [Sample Code (import projects)] on the right menu.

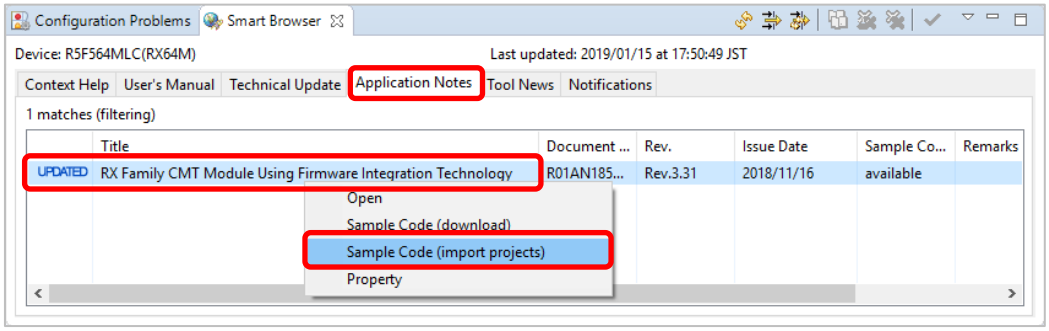


Figure 4-26 Sample Code (import projects)

(3) Specify the save destination of the sample code and click [Save].

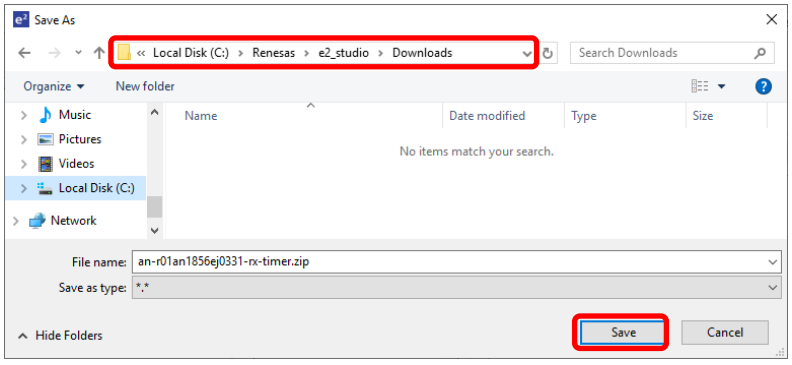


Figure 4-27 Sample code save destination

(4) When [End User License Agreement] dialog is displayed, click [Agree].

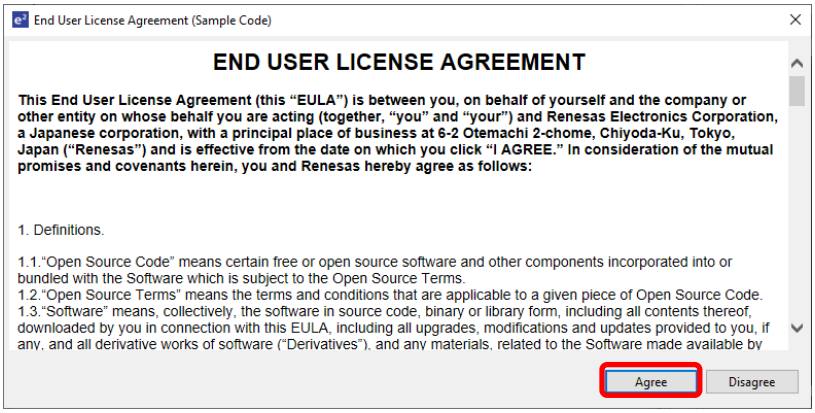


Figure 4-28 [End User License Agreement] dialog

(5) If the [Select import package] dialog is not displayed, go to procedure (6).

If the [Select import package] dialog box is displayed, select the package to import and click [OK].

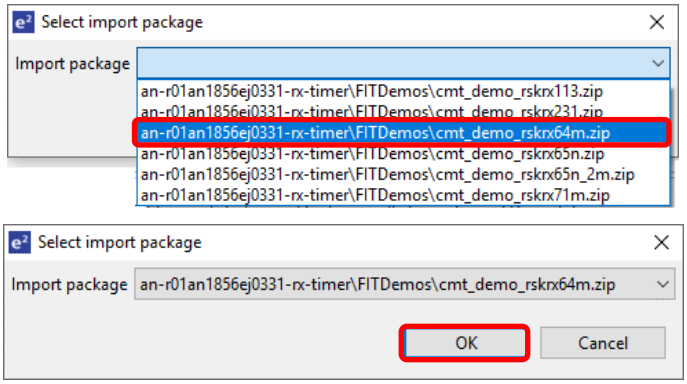


Figure 4-29 [Select import package] dialog

(6) When the [Import] dialog of the project appears, select the project and click [Finish].

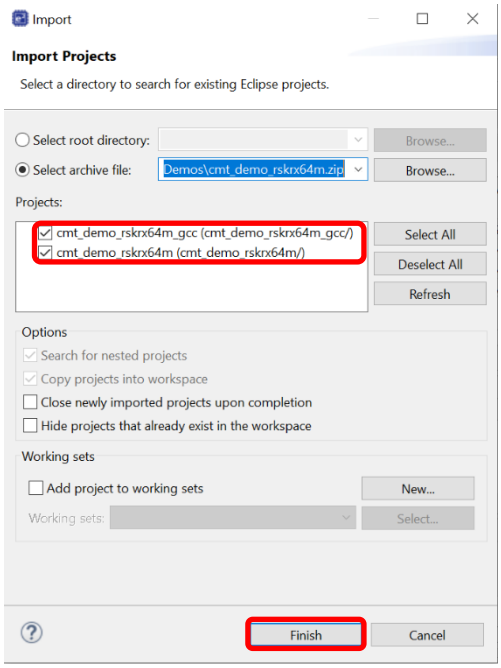


Figure 4-30 Import project

(7) It is added to [Project Explorer], and import of sample project is completed.

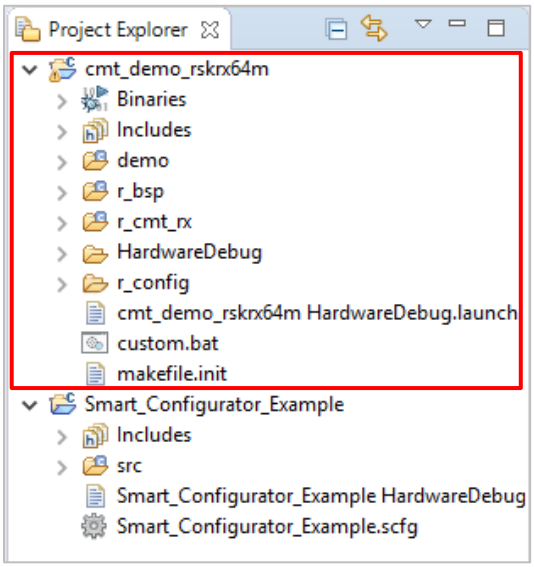


Figure 4-31 Addition to Project Explorer

4.4.9 Setting a FIT Software Component

To use FIT drivers or middleware, set configuration option. Setting methods depends on components,

- ✓ Set configuration options on Configure panel and settings will be generated to configuration file of FIT module automatically at each time of code generation action
- ✓ Set configuration options in configuration file of FIT module by manually

Configuration file of FIT module will be generated in the folder r_config. For the settings of the configuration options, refer to “chapter 7.1, Adding Custom Code in the Case of Firmware Integration Technology (FIT)”.

In addition, some components provide pin setting on the Configure panel. Followings are examples of pin setting on Configure panel.

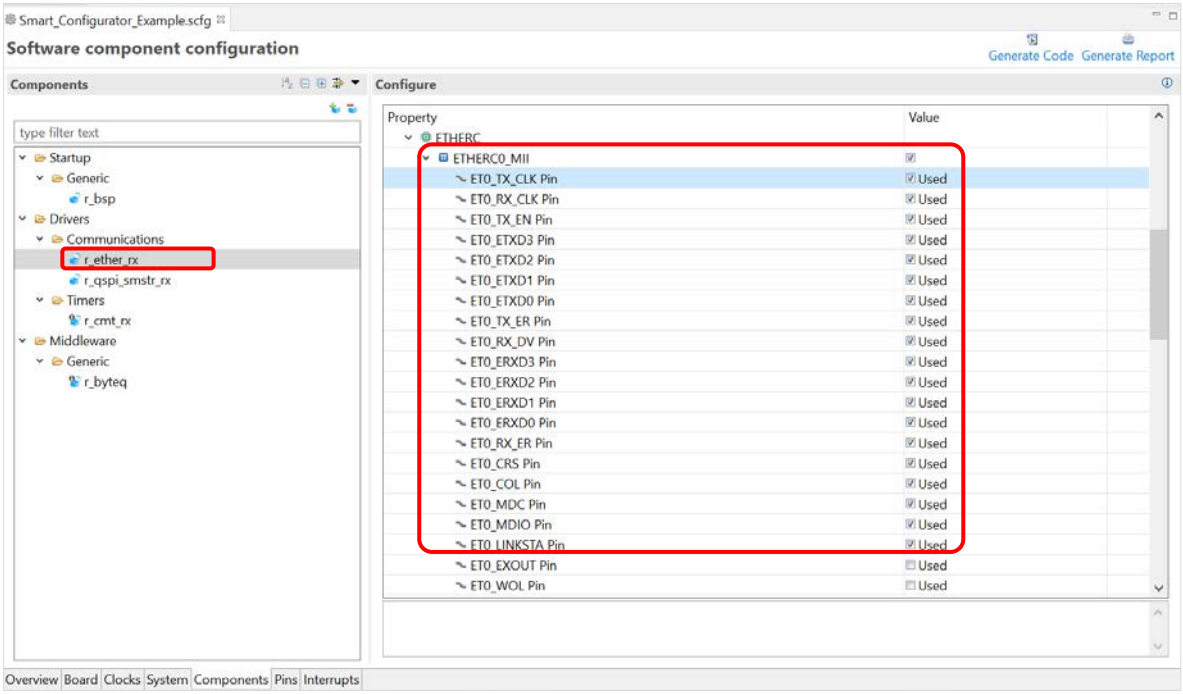


Figure 4-32 Pin Settings for r_ether_rx

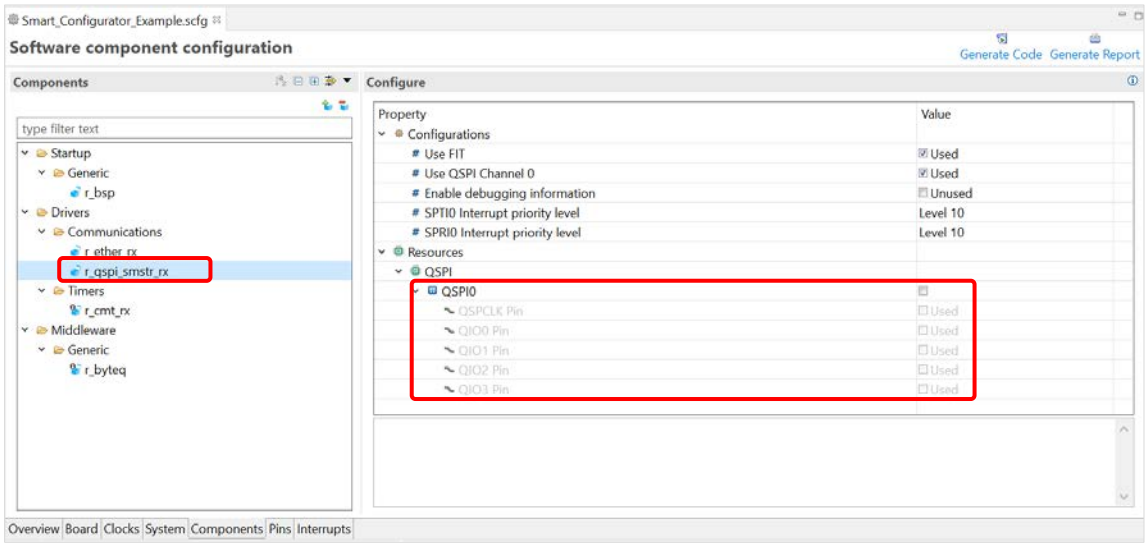


Figure 4-33 Pin Settings for r_qspi_smstr_rx

4.4.10 Version change of FIT software component

The following describes the procedure for version change of FIT software component.

(1) From the component tree, right-click the FIT software component that you want to change version.

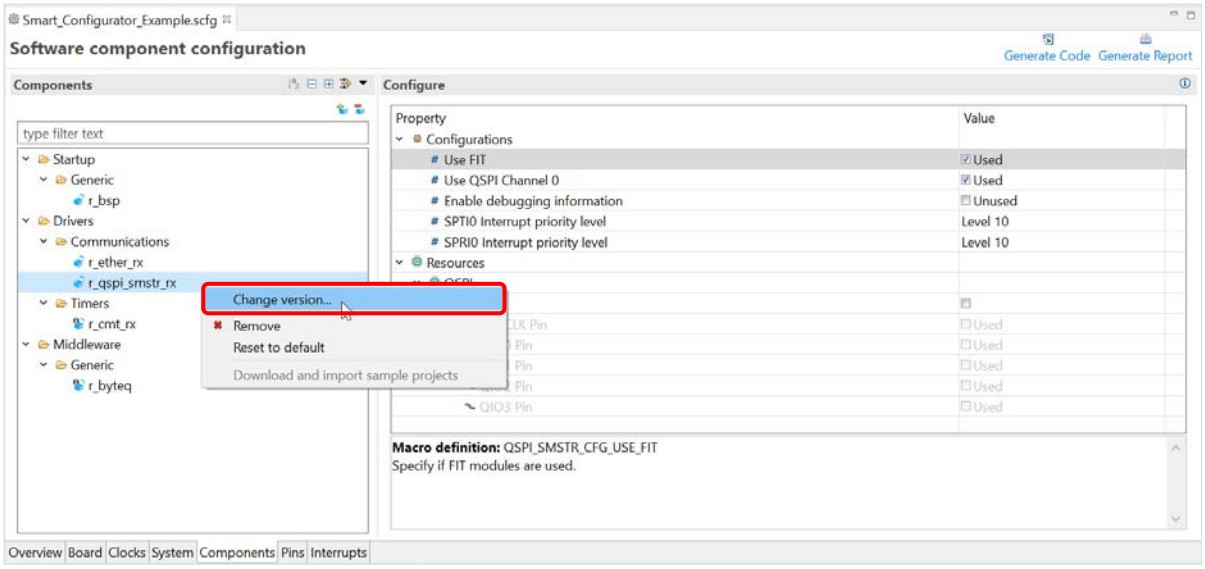


Figure 4-34 Version change of FIT software component

(2) Select [Change Version ...] from the context menu.

(3) In the [Change Version] dialog box, select the version you want to change. If you select a version that the device does not support, [Selected version doesn't support current device or toolchain] will be displayed, so select the corresponding version.

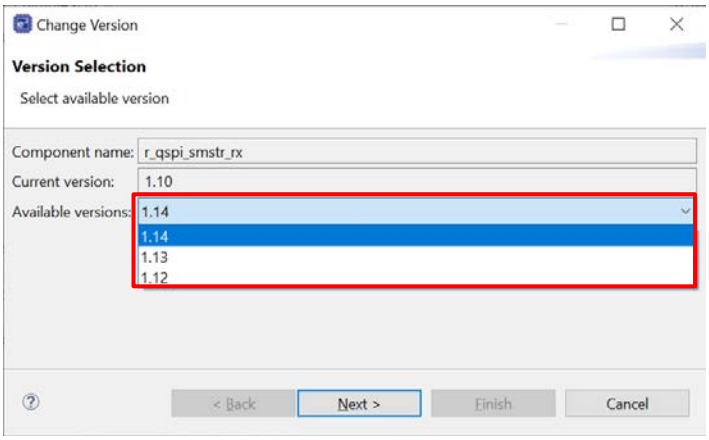


Figure 4-35 Select version of FIT software component

(4) Click [Next].

(5) By version change, a list of setting items to be changed is displayed. Confirm that there is no problem and click the [Finish].

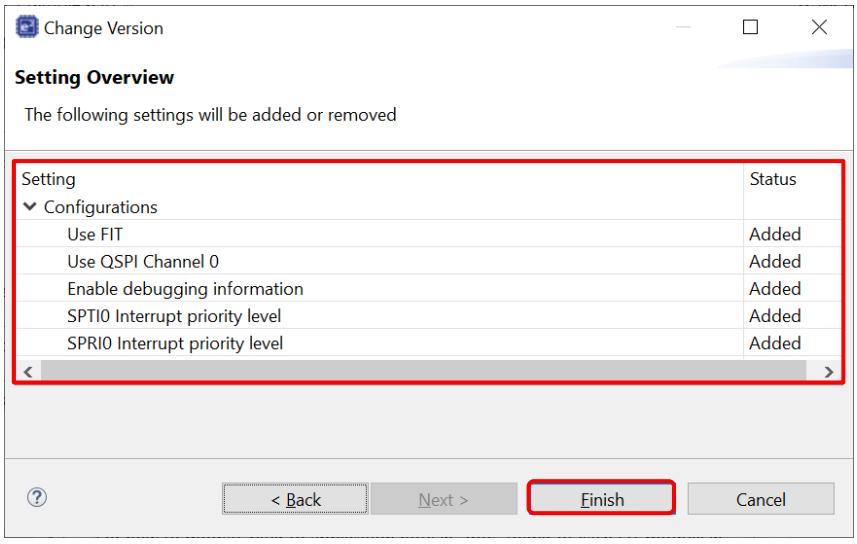


Figure 4-36 Confirm setting change item

(6) As [Confirm to change version and proceed to generate code] is displayed, if you are fine to proceed, click [Yes].

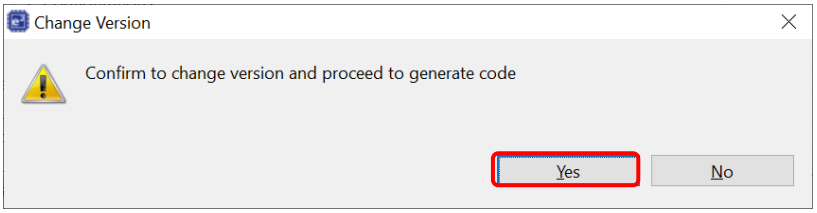


Figure 4-37 Confirm version change

(7) The FIT software component version is changed. Code generation is executed automatically.

4.4.11 Solving the grey-out component

When a component version is not available, it will be greyed out. Follow the procedure below to fix a greyed-out component.

(1) From the component tree, right-click the greyed-out component and select [Change version…]. Refer to chapter “chapter 4.4.10 Version change of FIT software component” to change to an available version.

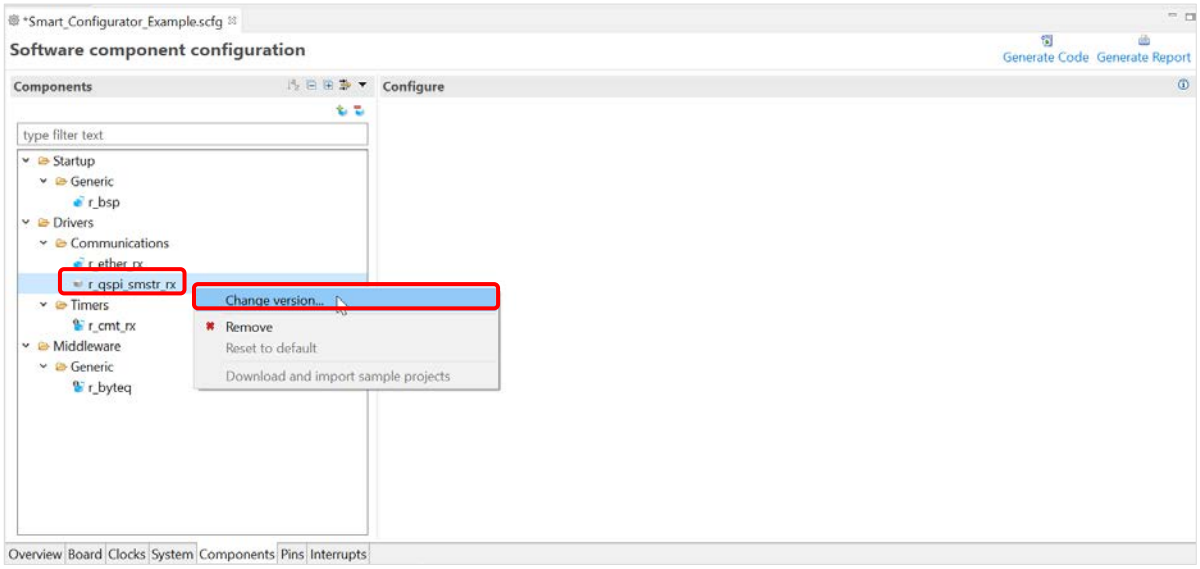


Figure 4-38 Change version of a greyed-out component

(2) If there is no available version for this component, refer to “chapter 4.4.6 Downloading a FIT module” to download this component from Renesas website.

4.4.12 Setting the RTOS Kernel

The following describes the procedure for setting the FreeRTOS Kernel.

For Renesas FreeRTOS, refer to Renesas FreeRTOS related documentation.

(1) Select [FreeRTOS_Kernel] in the component tree.

(2) Parameters corresponding to the RTOS kernel are displayed in the Configure panel, and configuration settings can be changed.

(3) Description of the parameter selected in the Configure panel and the corresponding macro definition are displayed in this area.

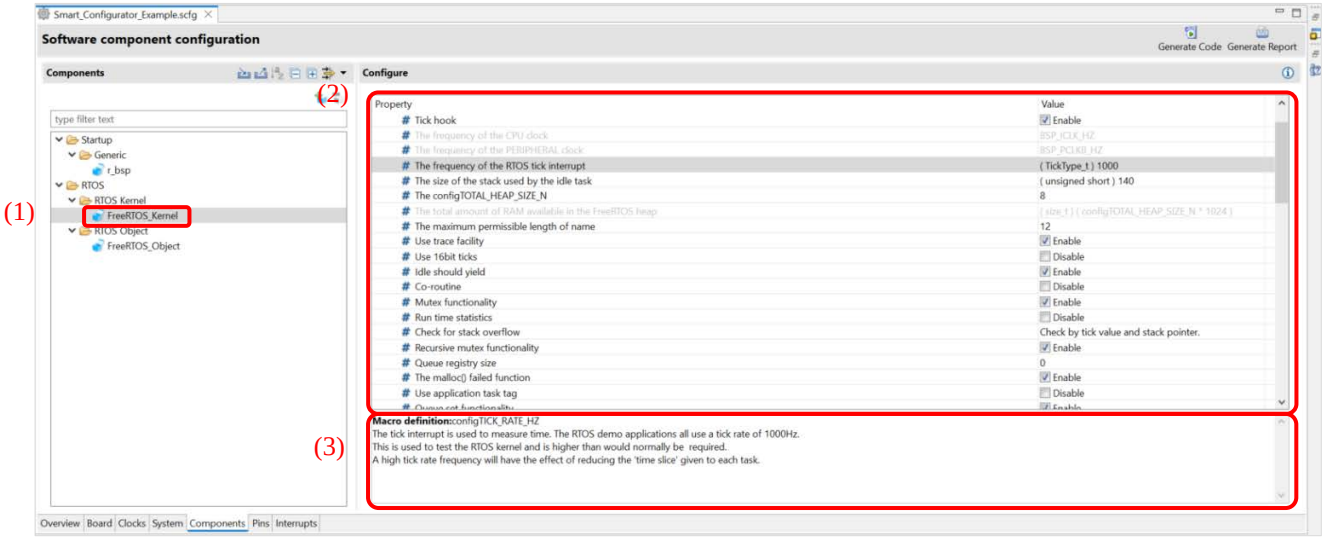


Figure 4-39 Settings for FreeRTOS_Kernel

4.4.13 Creating the RTOS Object

The following describes the procedure for setting the FreeRTOS Object.

(1) Select [FreeRTOS_Object] in the component tree.

(2) Configuration Tabs corresponding to the RTOS objects are displayed in the Configure panel. Select a tab corresponding to the object you want to create.

(3) Click on the [+] button to create a new object. Click on the [-] button to remove an object.

(4) Use text boxes and combo boxes to change the object setting.

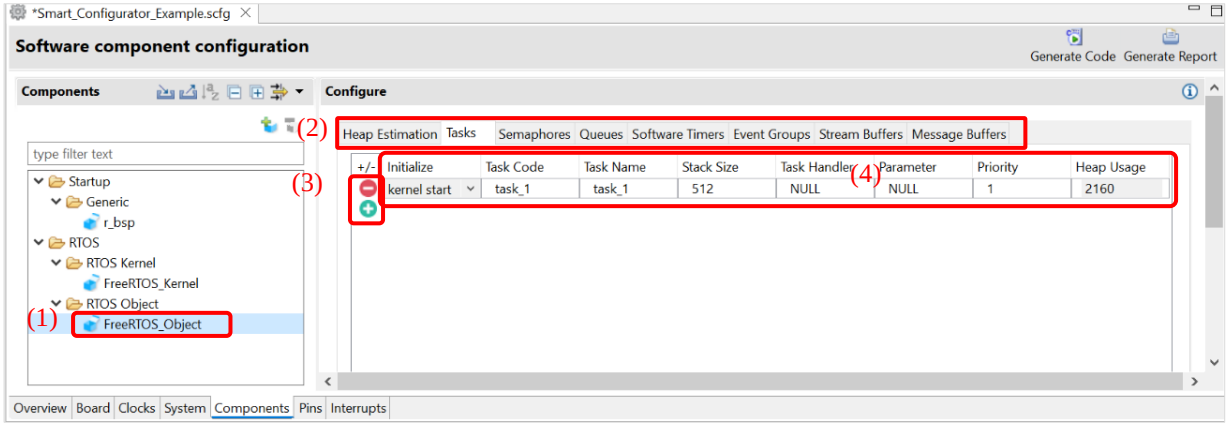


Figure 4-40 Create FreeRTOS_Object

4.4.14 Setting the RTOS Library

The following describes the procedure for setting the FreeRTOS Library.

(1) Select a library under [RTOS_Library] folder in the component tree.

Note: The RTOS Library is only available for “FreeRTOS (with IoT libraries)” project.

(2) Parameters corresponding to the RTOS Library are displayed in the Configure panel, and configuration settings can be changed.

(3) Description of the parameter selected in the Configure panel and the corresponding macro definition are displayed in this area.

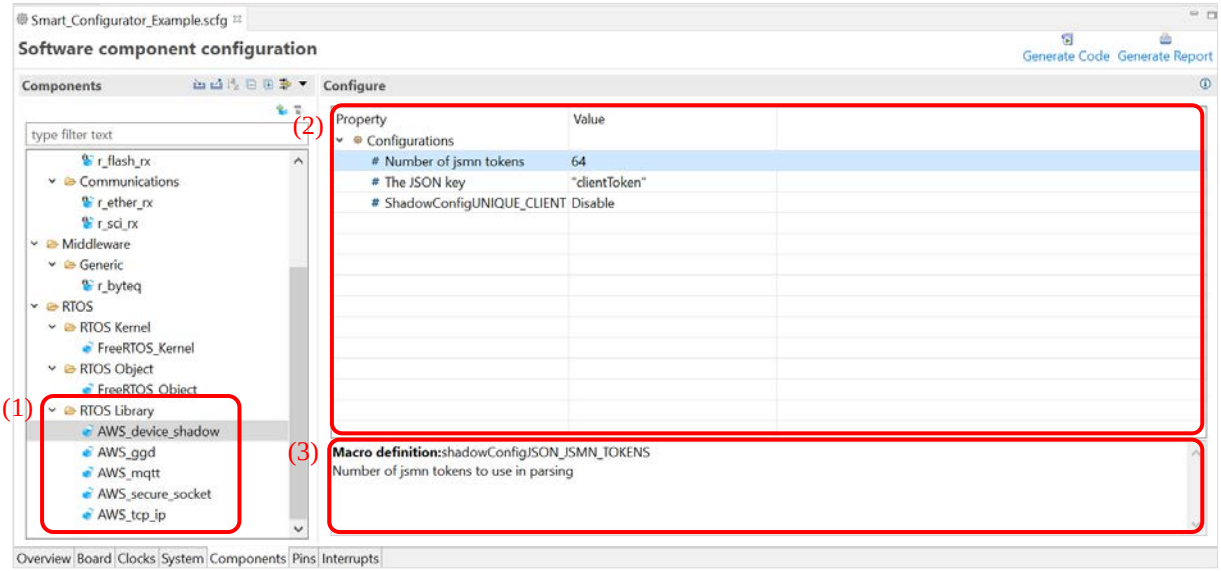


Figure 4-41 Settings for RTOS Library

4.4.15 Configure Analog Front End component

The RX23E-A group microcontrollers are equipped with an analog front end (AFE) that can measure temperature, pressure, flow, and weight with less than 0.1% precision without calibration, making it ideal for high-precision sensing, test and measurement equipment.

When creating project for RX23E-A, you can use the AFE configuration tool for:

- Easy setting AFE on GUI
- Easy checking pins confliction
- Easy checking analog multiplexer connection

This chapter will describe how to use analog multiplexer connection:

(1) In RX23E-A project, open smart configurator, select [Components] tab and add new component “Analog Front End” and “Continuous Scan Mode DSAD”

(2) Select [Config_DSAD0] from the Components Tree. Perform setting as following:

- Analog input channel setting: enable channel 0
- [Channel setting] > [Channel 0] > Positive input signal: AIN1
- [Channel setting] > [Channel 0] > Negative input signal: AIN3

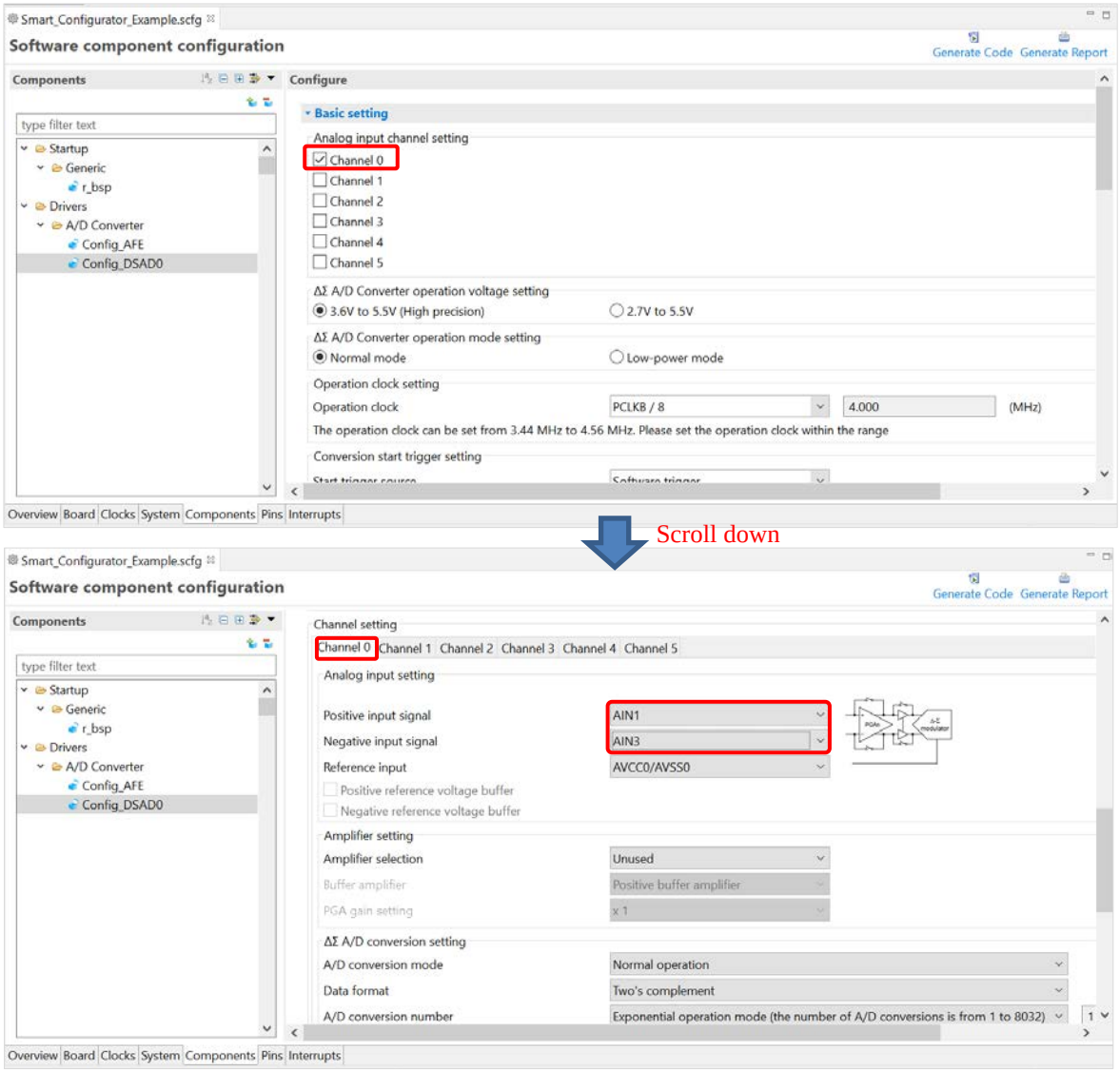


Figure 4-42 Config_DSAD0 Setting

(3) Select [Config_AFE] from the Components Tree. In the [AFE setting] tab, change the [Bias output setting] as follows:

- Enable bias voltage output: checked.
- AIN1 pin output: checked
- AIN3 pin output: checked

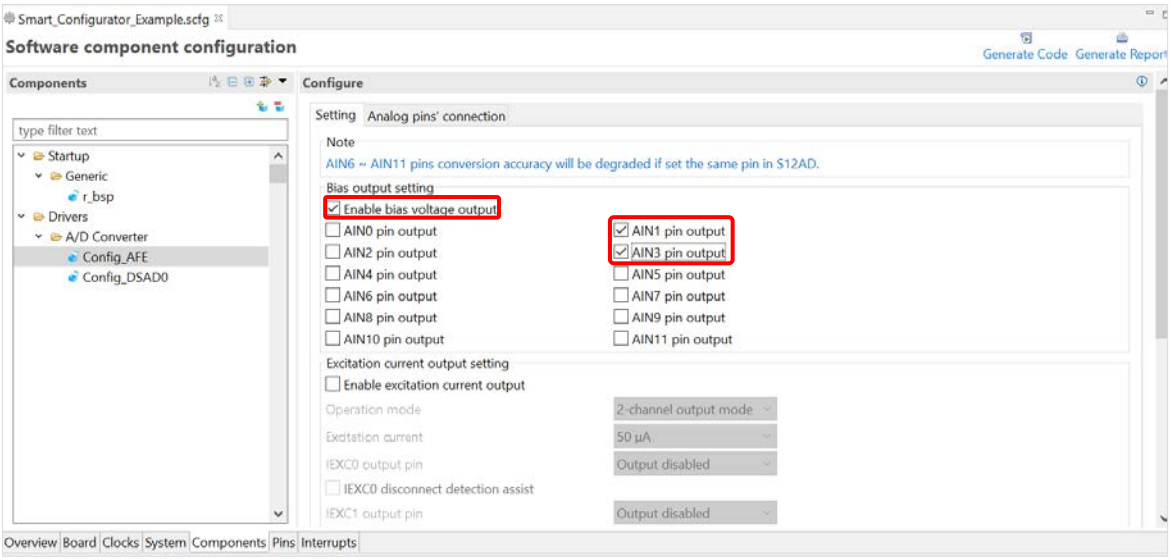


Figure 4-43 Config_AFE setting

(4) Select [Analog pin's connection] tab, you can see the block diagram of the AFE multiplexed pin connection. The active connection of analog multiplexer is highlighted. So, you can check the analog multiplexer connection easily and confirm the configuration.

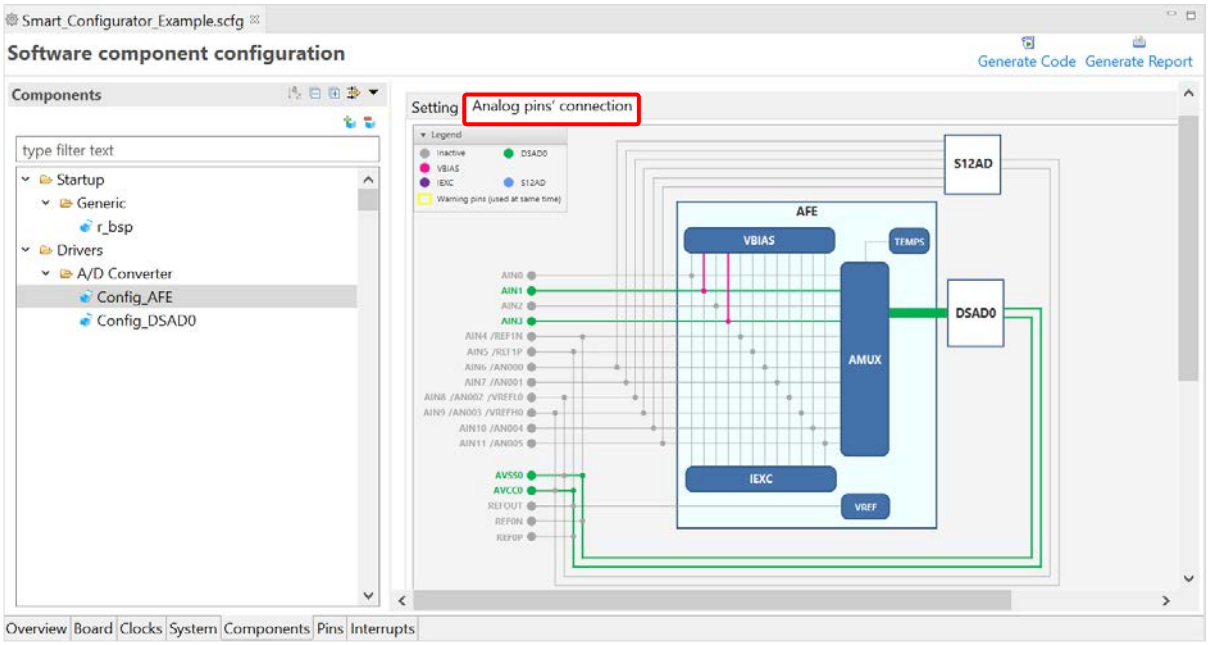


Figure 4-44 Block diagram of the AFE multiplexed pin connection

4.4.16 Configure Motor Component

Motor Driver Generator is a utility tool to generate drivers for all peripheral functions used for motor control from one GUI setting.

Note: The supported devices are RX13T, RX23T, RX24T, RX24U, RX26T, RX66T, RX72T, and RX72M.

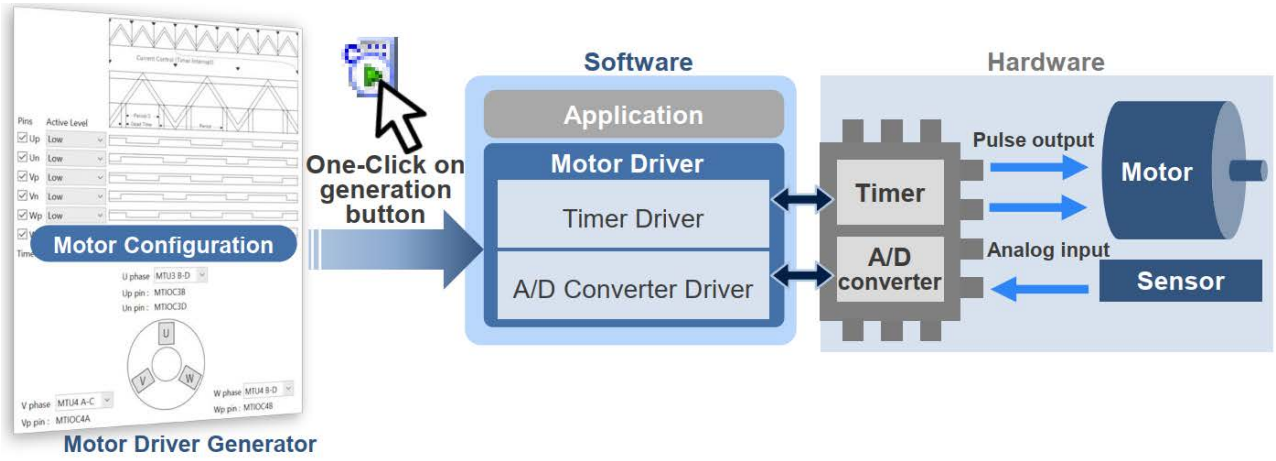


Figure 4-45 Motor Driver Generator

This chapter will describe how to use Motor Driver Generator:

(1) In the project of supported device (for e.g RX24T), open Smart Configurator, select [Components] tab and add new component “Motor”. In the [New Component] dialog, select the Motor type as you wish and click [Finish] button.

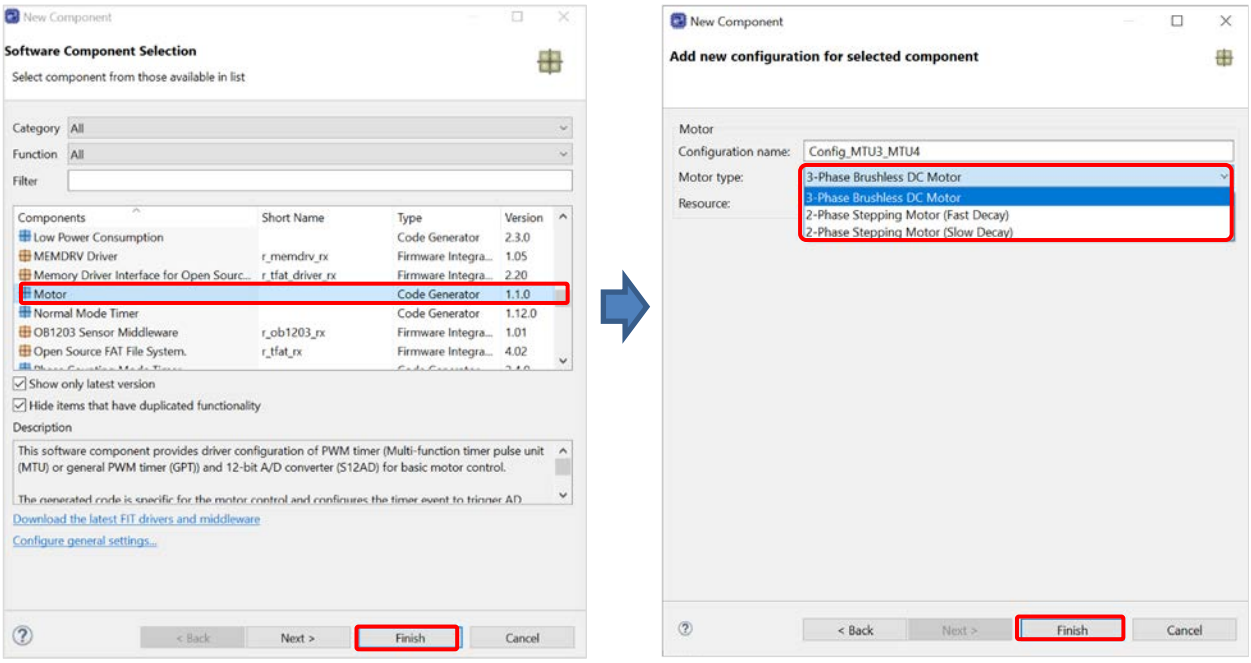


Figure 4-46 Add Motor component

(2) Select “Config_MTU3_MTU4” in the component tree, on the Configure panel, select the [Timer Setting] tab. In this tab, you will be able to use the GUI for Timer driver setting, including: Period Setting, Output Level Setting, Output Pin Select and Timer Interrupt Setting.

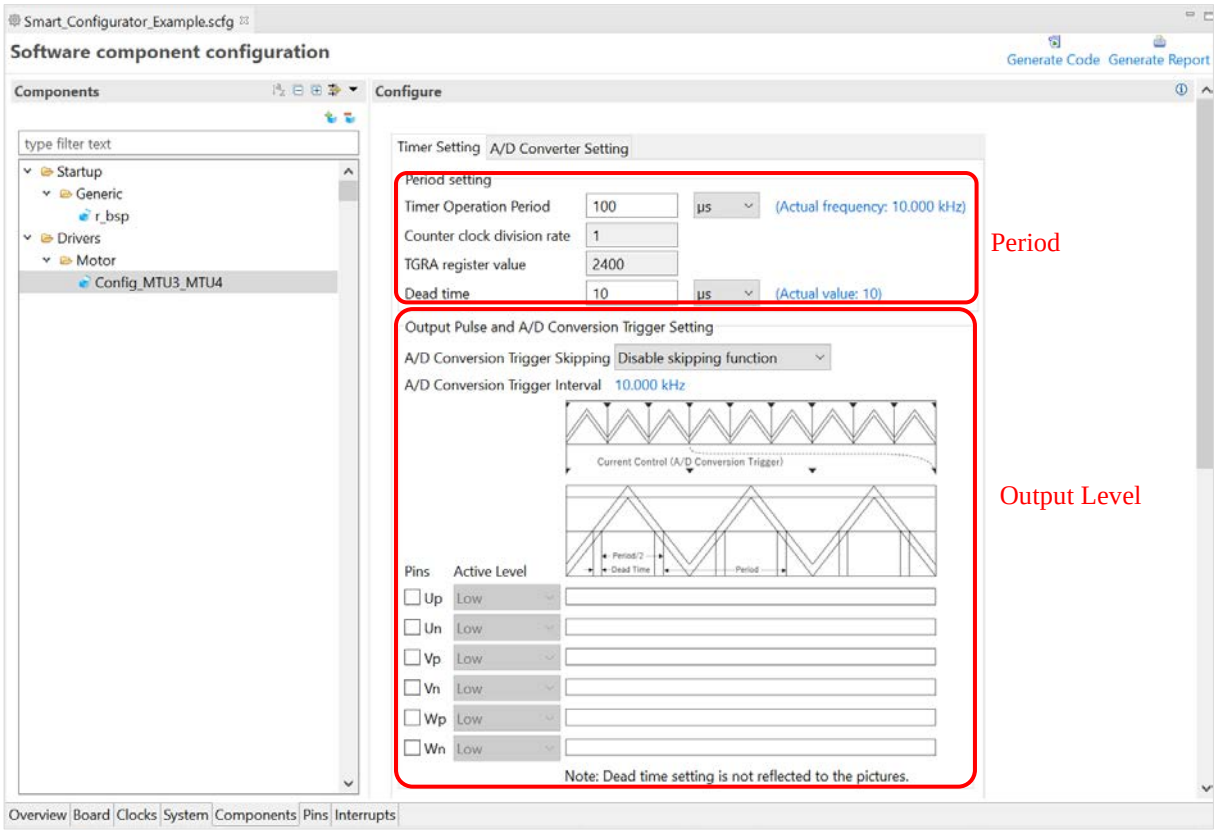


Figure 4-47 Timer Driver Setting (1)

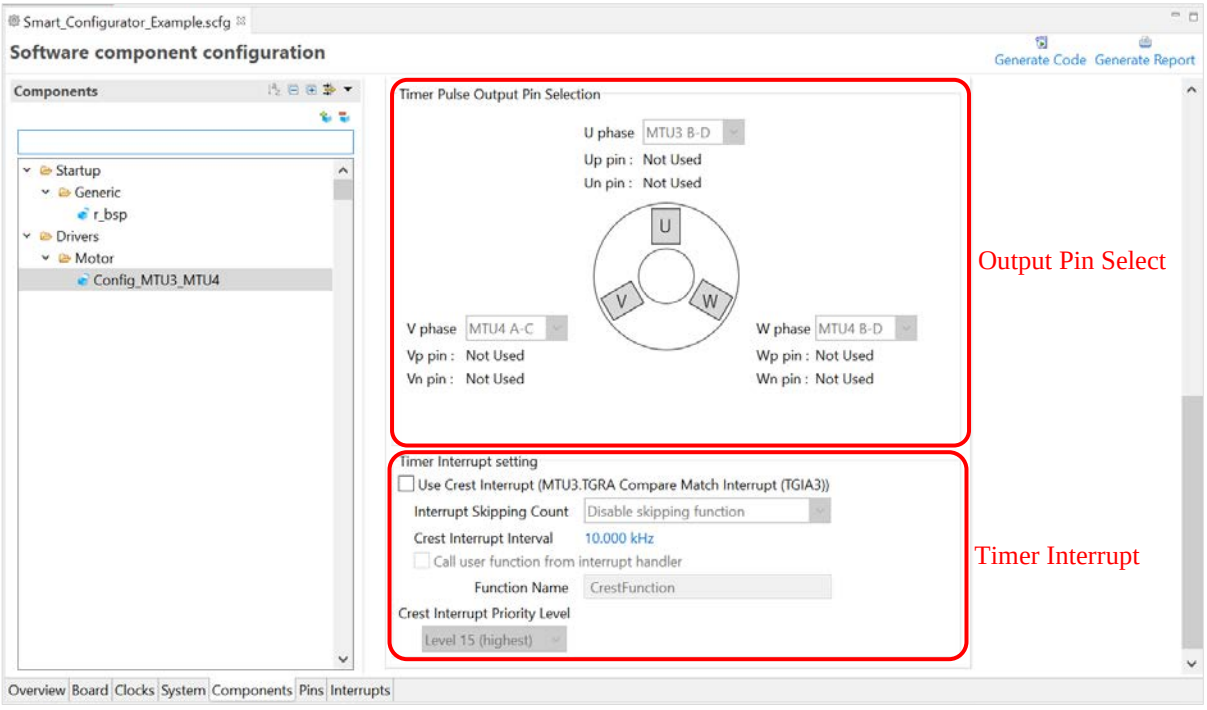


Figure 4-48 Timer Driver Setting (2)

(3) Select the [A/D Converter Setting] tab. In this tab, you will be able to use the GUI for A/D Converter driver setting, including: Analog Pin Select, A/D Interrupt Setting.

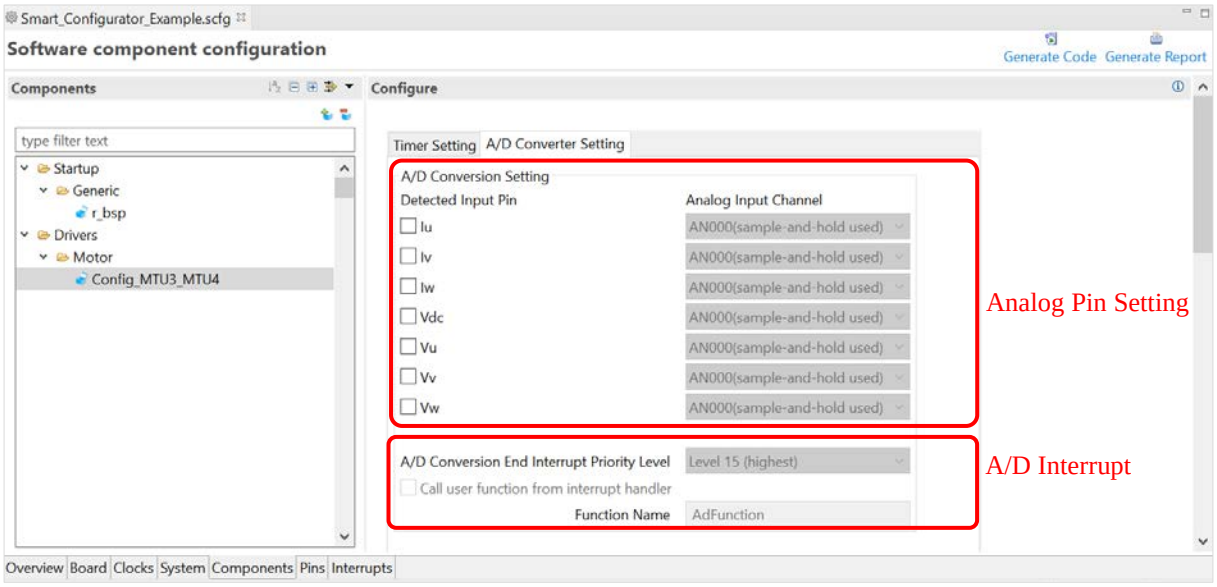


Figure 4-49 A/D Converter Driver Setting

(4) GPT peripheral is supported in some devices (for e.g., RX26T). Add new component “Motor”. In the [New Component] dialog, select the [Triangle_GPT] or [GPT0_GPT1_GPT2], [GPT4_GPT5_GPT6] resource and click [Finish] button.

Note: The [GPT0_GPT1_GPT2] and [GPT4_GPT5_GPT6] resources are exclusively designed for GPT Complementary Mode, which is accessible only in RX26T

The [Triangle_GPT] resource is intended for GPT Triangle PWM Mode. With this resource, users can modify GPT channel configurations for both Master and Slave channels, utilizing the currently available GPT channels. The [Triangle_GPT] resource is available on RX24T, RX24U, RX26T, RX66T, RX72M, RX72T.

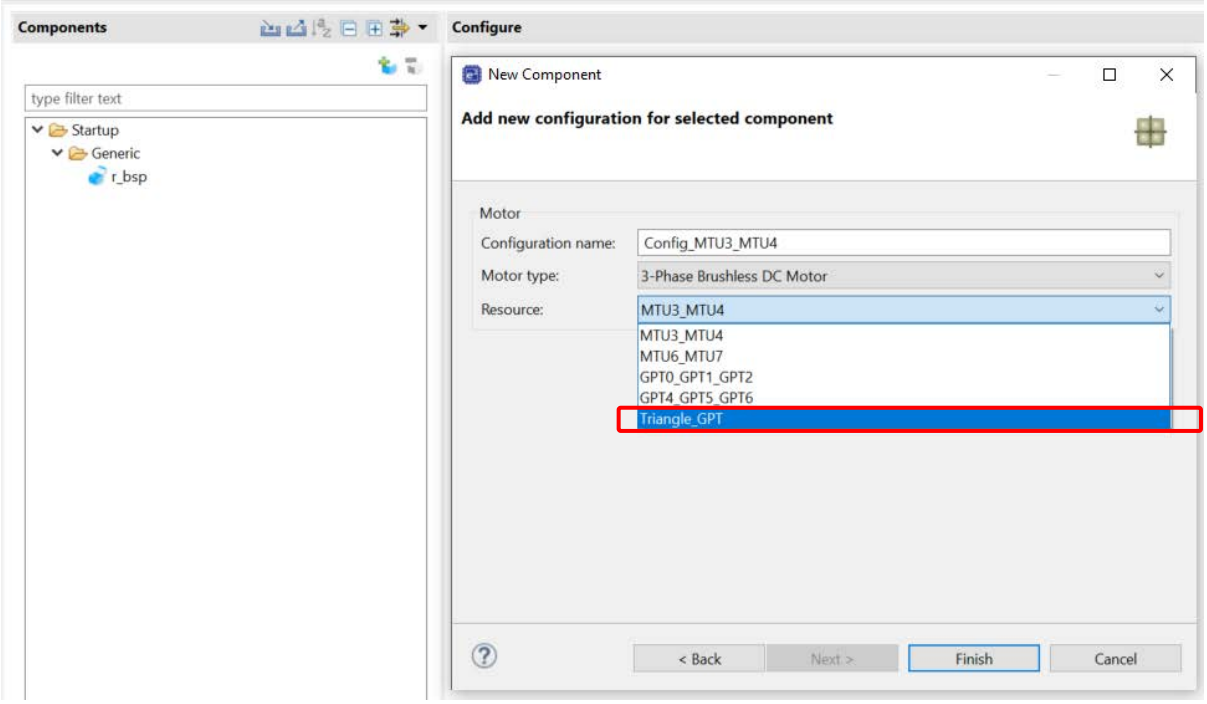


Figure 4-50 Select MOTOR resource

(5) GPT resource device has some differences between MTU resources

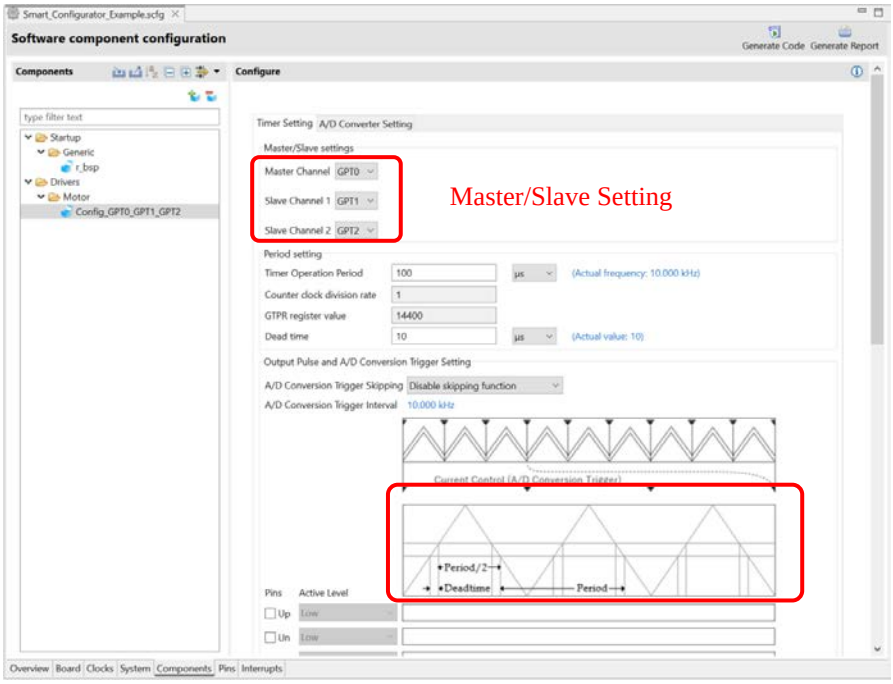


Figure 4-51 Timer Driver Setting(1)

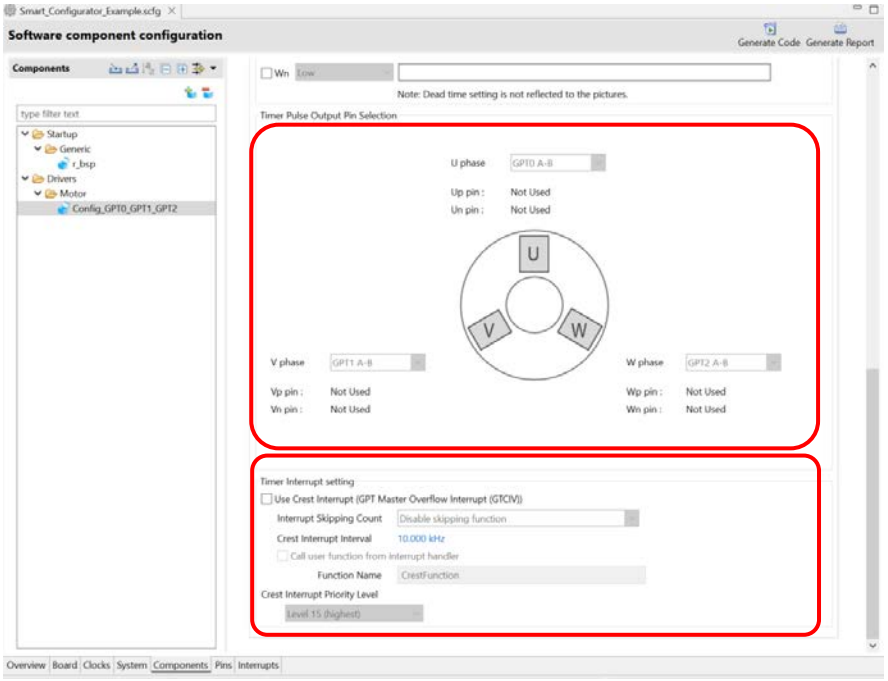


Figure 4-52 Timer Driver Setting (2)

4.4.17 Configure general setting of component

The general setting of the component, such as code generation component settings, FIT(RX) component settings, dependency settings and location settings, can be configured inside the [Preferences] dialog.

If you want to change the settings, click the [Configure general settings...] link on the [Software Component Selection] page displayed in the [New Component] dialog (Figure 4-8), and display the [Preferences] dialog.

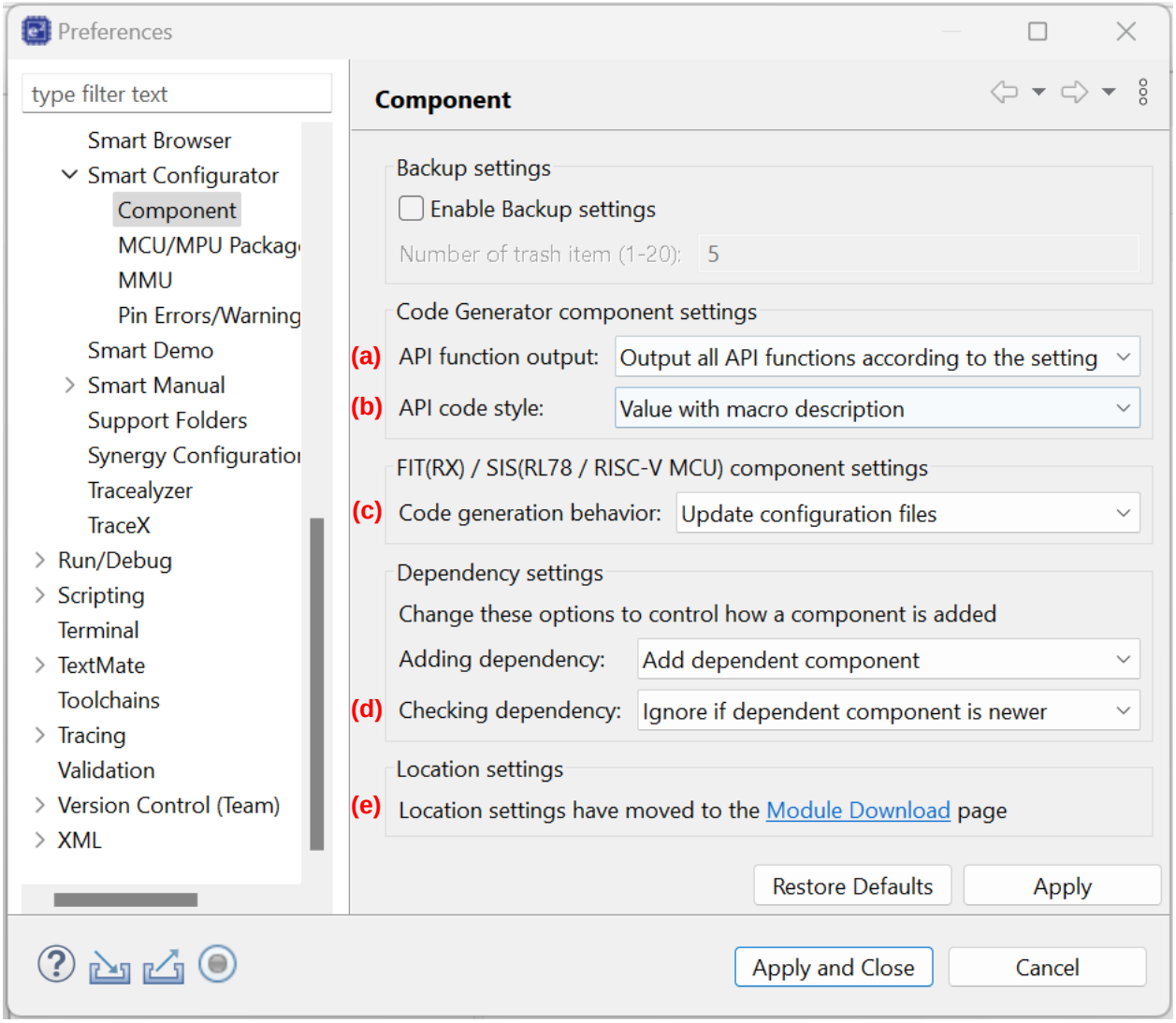


Figure 4-53 Configure general setting of component

Notes:

(a) The API function output has two options: “Output all API functions according to the setting” and “Output only initialization API function”. “Output all API functions according to the setting” is the default selection.

If “Output all API functions according to the setting” is selected, all API functions will be generated.

If “Output only initialization API function” is selected, only initialization API function will be generated. (Only void R_{ConfigurationName}_Create (void), void R_{ConfigurationName}_Create_UserInit (void) in *.h *, *c * are generated out.)

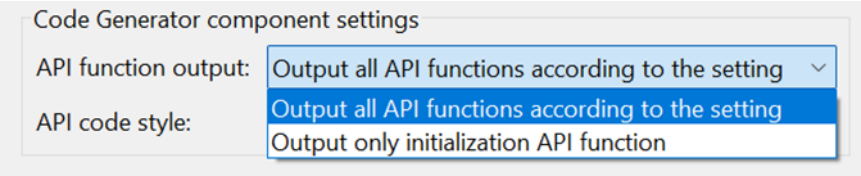


Figure 4-54 Updates of “API function output”

Output only initialization API feature is supported for individual configuration (Code Generator component). For using this feature, please right-click the selected component and select the "Output only initialization API" from the context menu.

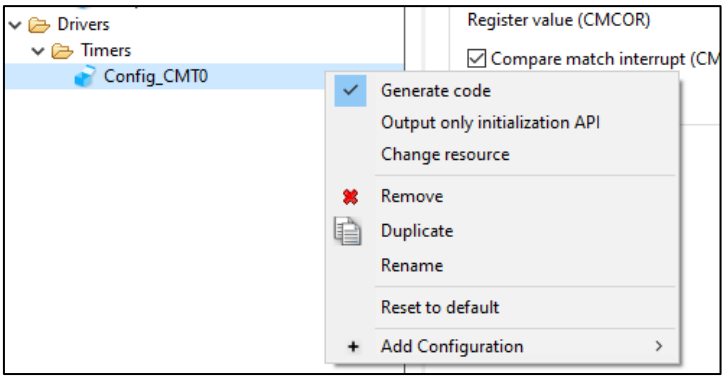


Figure 4-55 Output only initialization API

(b) The API code style has two options: “Value with macro description” and “Value without macro description (raw HEX)”. “Value with macro description” is the default selection.

If “Value with macro description” is selected, all API with macro description will be generated.

If “Value without macro description (raw HEX)” is selected, code with HEX value will be generated.

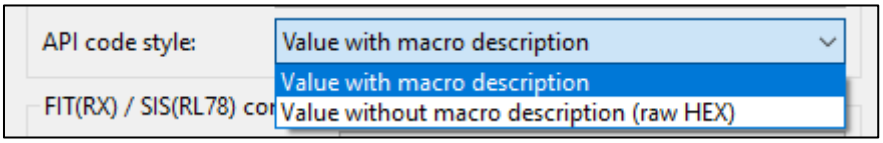


Figure 4-56 Updates of "API code style"

(c) The code generation behavior has two options: “Update configuration files” and “Re-generate all component files”. “Update configuration files” is the default selection.

If "Update configuration files" is selected and generate code, Smart Configurator will check whether the files are existing inside the user project. If the file exists, the file will not be overwritten. However, configuration files (e.g., xxx_config.h) will still be refreshed when code is generated.

If "Re-generate all component files" is selected and generate code, Smart Configurator does not check the existence of the file and the file will always be overwritten.

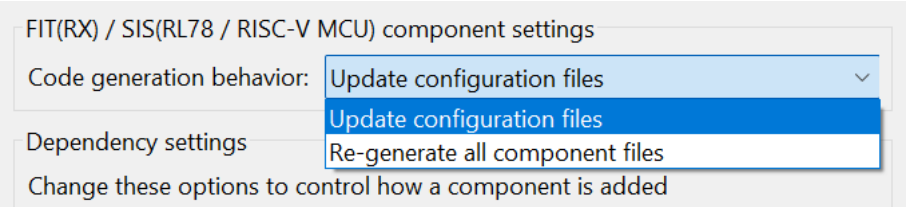


Figure 4-57 Updates of "Code generation behavior"

(d) Checking dependency has three options: “Do not check for dependent component”, “Ignore if dependent component is newer” and “Strict check for dependent component”. “Ignore if dependent component is newer” is the default selection.

If the version of the module and its dependency do not match, a warning message W04020011 is displayed. If you check the revision history of the module and its dependencies and you do not need to change the module you are using, you can ignore this warning.

To clear this warning, select "Do not check for dependent component" in the [Checking dependency] list box in component preferences, then click [Apply].

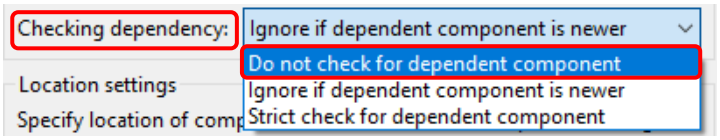


Figure 4-58 Updates of "Checking dependency"

(e) If you downloaded the FIT module directly from the website, unzip the downloaded zip file and copy the xml file and zip file in the FIT Modules folder to the [Module Download] - [Location (RX)] folder.

To change the location, click on the [Module Download] link, then find [Location (RX)], click [Browse…] and select another folder.

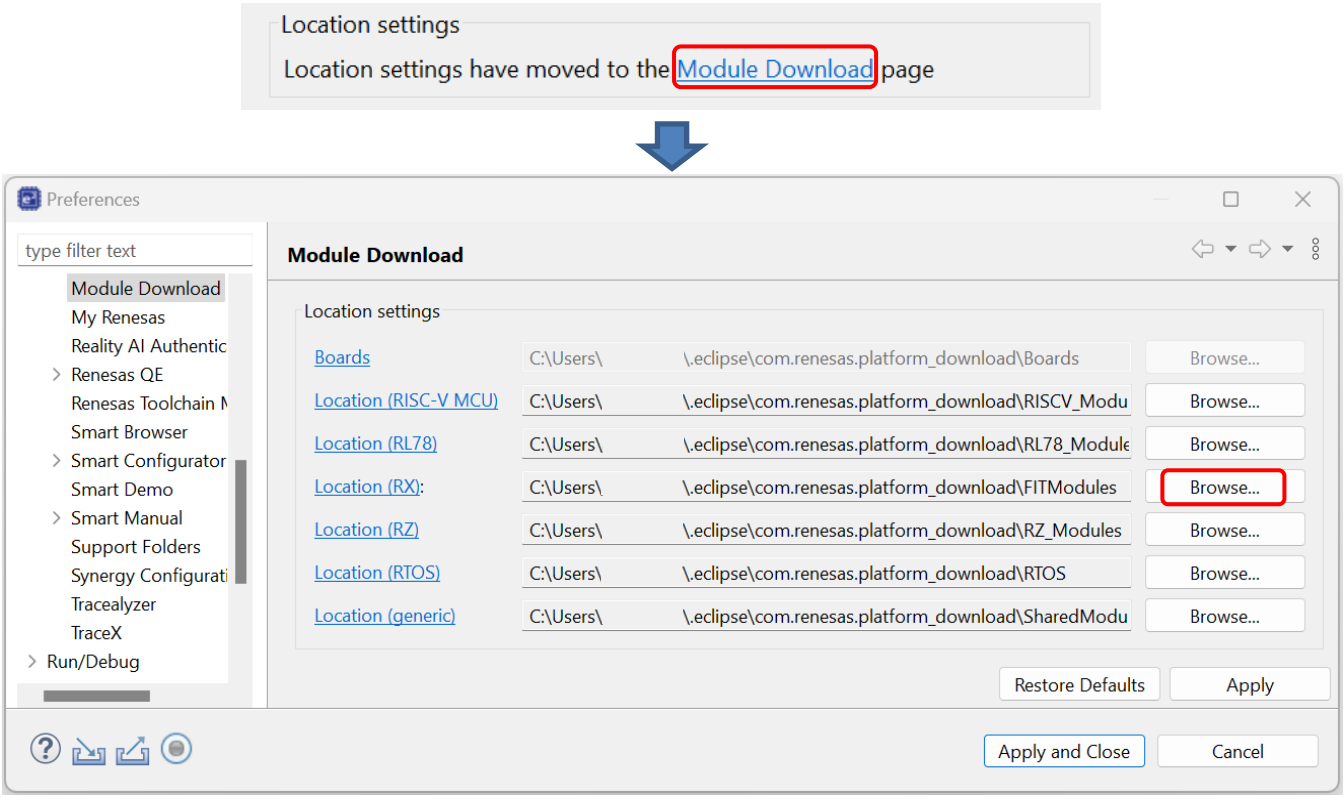


Figure 4-59 Updates of "Location settings"

4.5 Pin Settings

The [Pins] page is used for assigning pin functions. You can switch the view by clicking on the [Pin Function] and [Pin Number] tabs. The [Pin Function] list shows the pin functions for each of the peripheral functions, and the [Pin Number] list shows all pins in order of pin number.

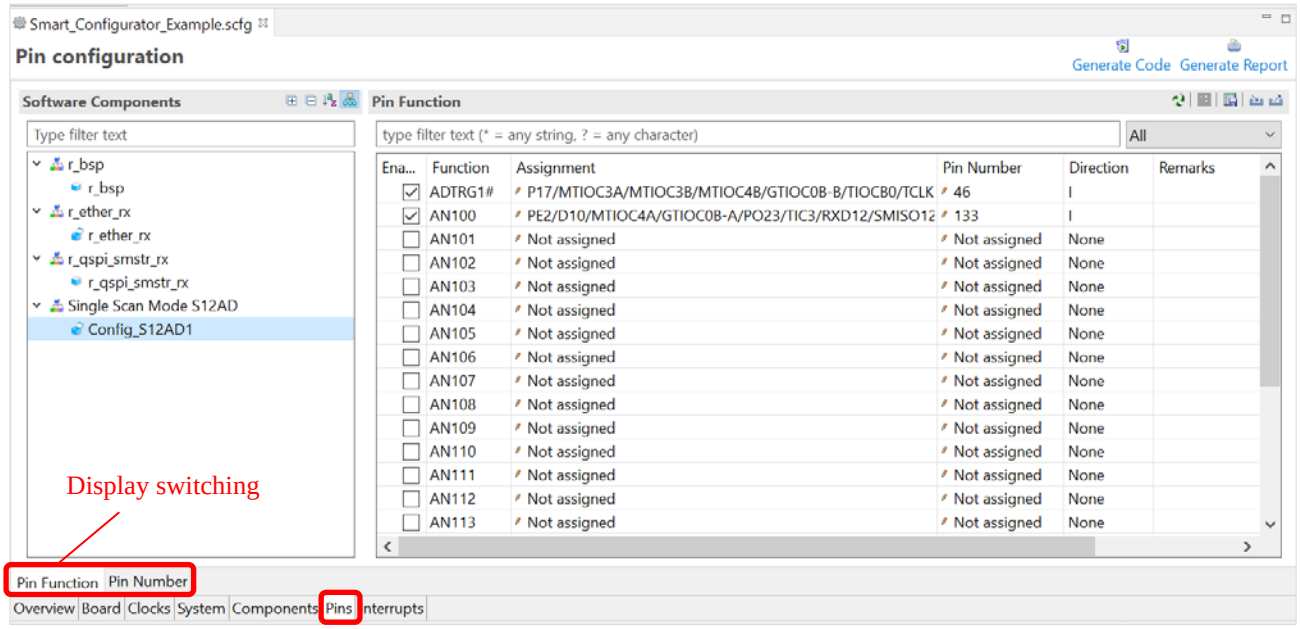


Figure 4-60 [Pins] Page ([Pin Function])

When you select a board on the [Board] page, the initial pin setting information of the board is displayed in [Default Function]. In addition, the [] icon displayed in the [Function] selection list indicates the initial pin function of the board.

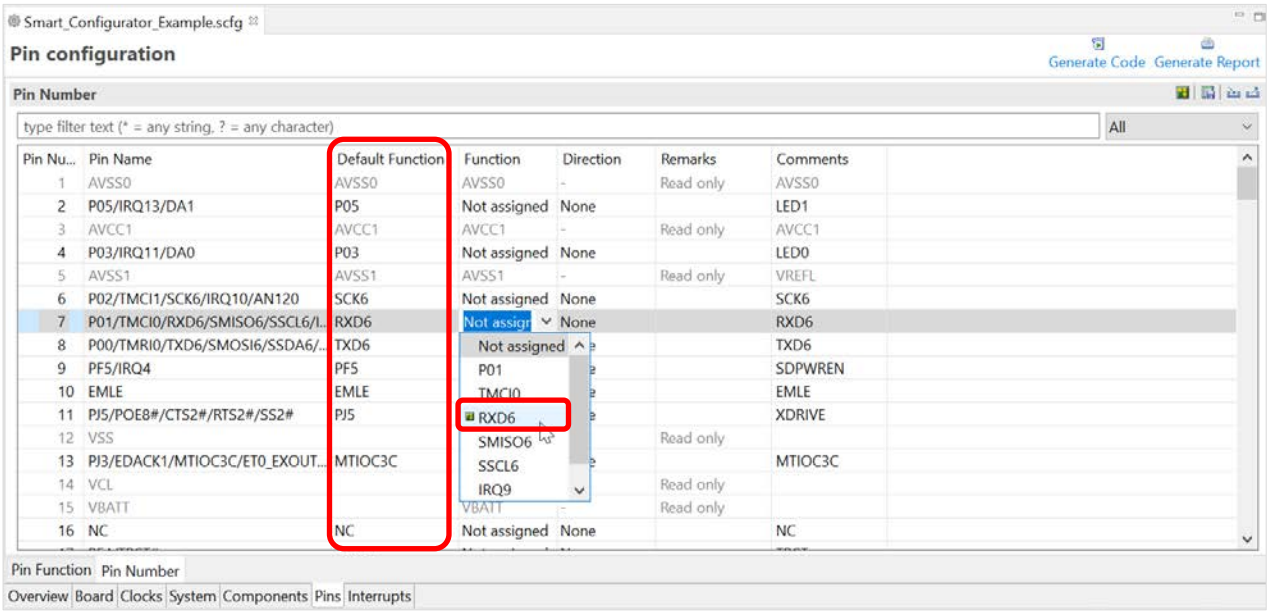


Figure 4-61 [Pins] Page ([Pin Number])

4.5.1 Changing the pin assignment of a software component

The Smart Configurator assigns pins to the software components that are added to the project. Assignment of the pins can be changed on the [Pins] page.

This page provides two lists: Pin Function and Pin Number.

Follow the procedure below to change the assignment of pins to a software component in the Pin Function list.

(1) Click on [(Show by Hardware Resource or Software Components)] to switch to the component view.

(2) Select the target software component (for e.g. Config_S12AD1).

(3) Click the [Enabled] header to sort by pins used.

(4) In the [Assignment] column or [Pin Number] column on the [Pin Function] list, change the pin assignment (for e.g. change from P17 to P13).

(5) In addition, assignment of a pin can be changed by clicking on the [(Next group of pins for the selected resource)] button. Pin that has peripheral function is displayed each time the button is clicked.

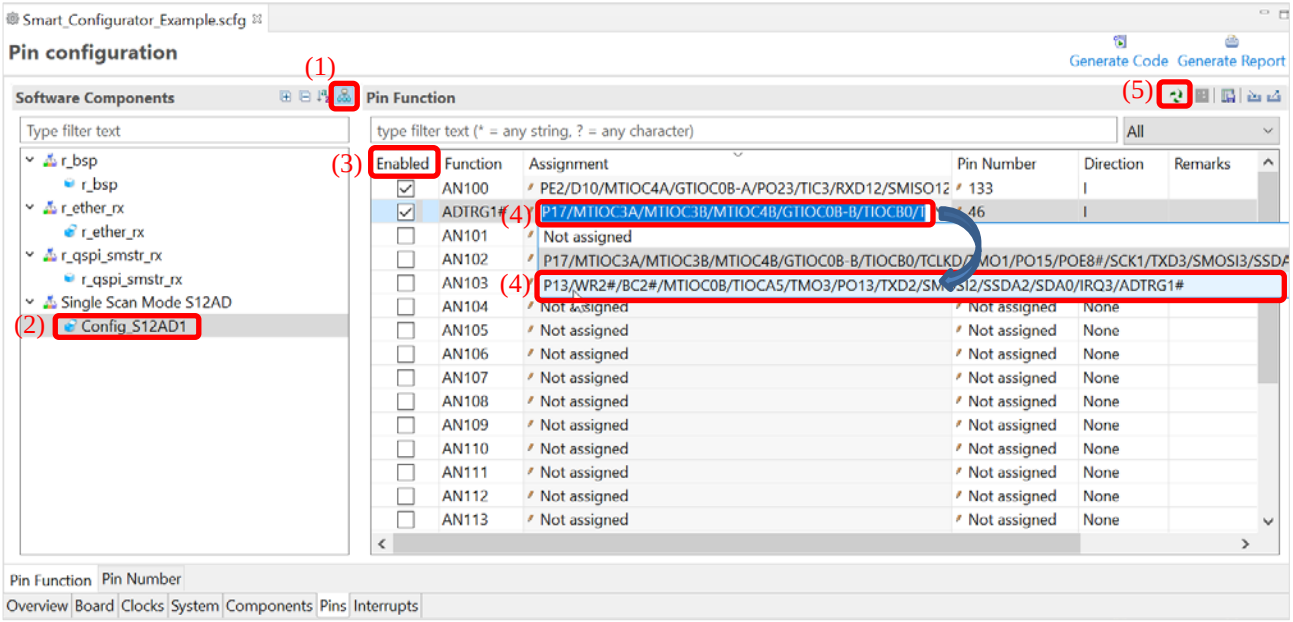


Figure 4-62 Pin Settings – Assigning Pins on the [Pin Function] List

The Smart Configurator allows you to enable pin functions on the [Pins] page without linking the current software component to another. To distinguish these pins from other pins that are used by another software component, there will be a remark "No component is using this pin" on the list.

Note:

The function for assigning pins is not available for some FIT modules.

For the method of assigning pins to such a FIT module, refer to the application note in the <ProjectDir>¥src¥smc_gen¥r_xxx¥doc folder for the FIT module.

4.5.2 Assigning pins using the MCU/MPU Package view

The Smart Configurator visualizes the pin assignment in the MCU/MPU Package view. You can save the MCU/MPU Package view as an image file, rotate it, and zoom in to and out from it.

Follow the procedure below to assign pins in the MCU/MPU Package view.

(1) Zoom in to the view by clicking the [(Zoom in)] button or scrolling the view with the mouse wheel.

(2) Right-click on the target pin.

(3) Select the signal to be assigned to the pin.

(4) The color of the pins can be customized through [Preference Setting...].

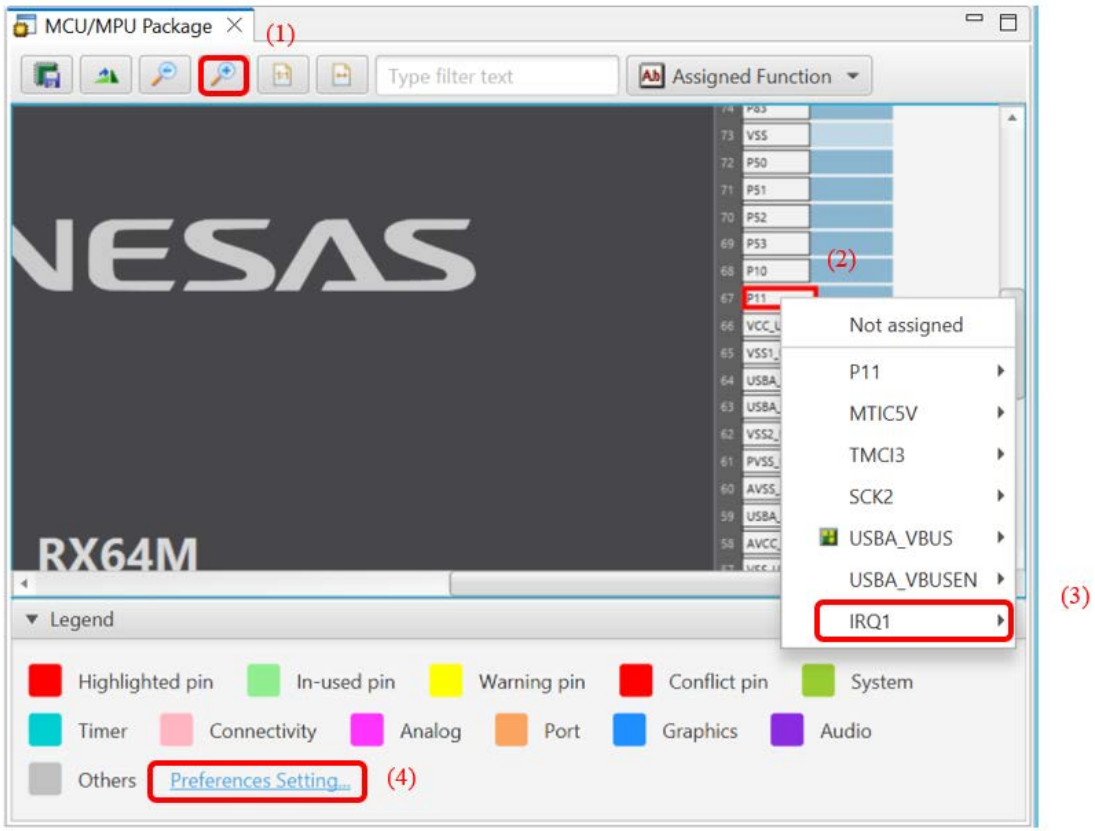


Figure 4-63 Assigning Pins Using the MCU/MPU Package View

4.5.3 Show pin number from pin functions

You can go to the pin number associated with a pin function.

Follow the procedure below to jump to pin number from a pin function.

(1) In the [Pin Function] tab, right click on a Pin Function to open the pop-up menu.

(2) Select “Jump to Pin Number”

(3) The [Pin Number] tab is opened with a Pin Number being selected. This is the pin number of the pin function.

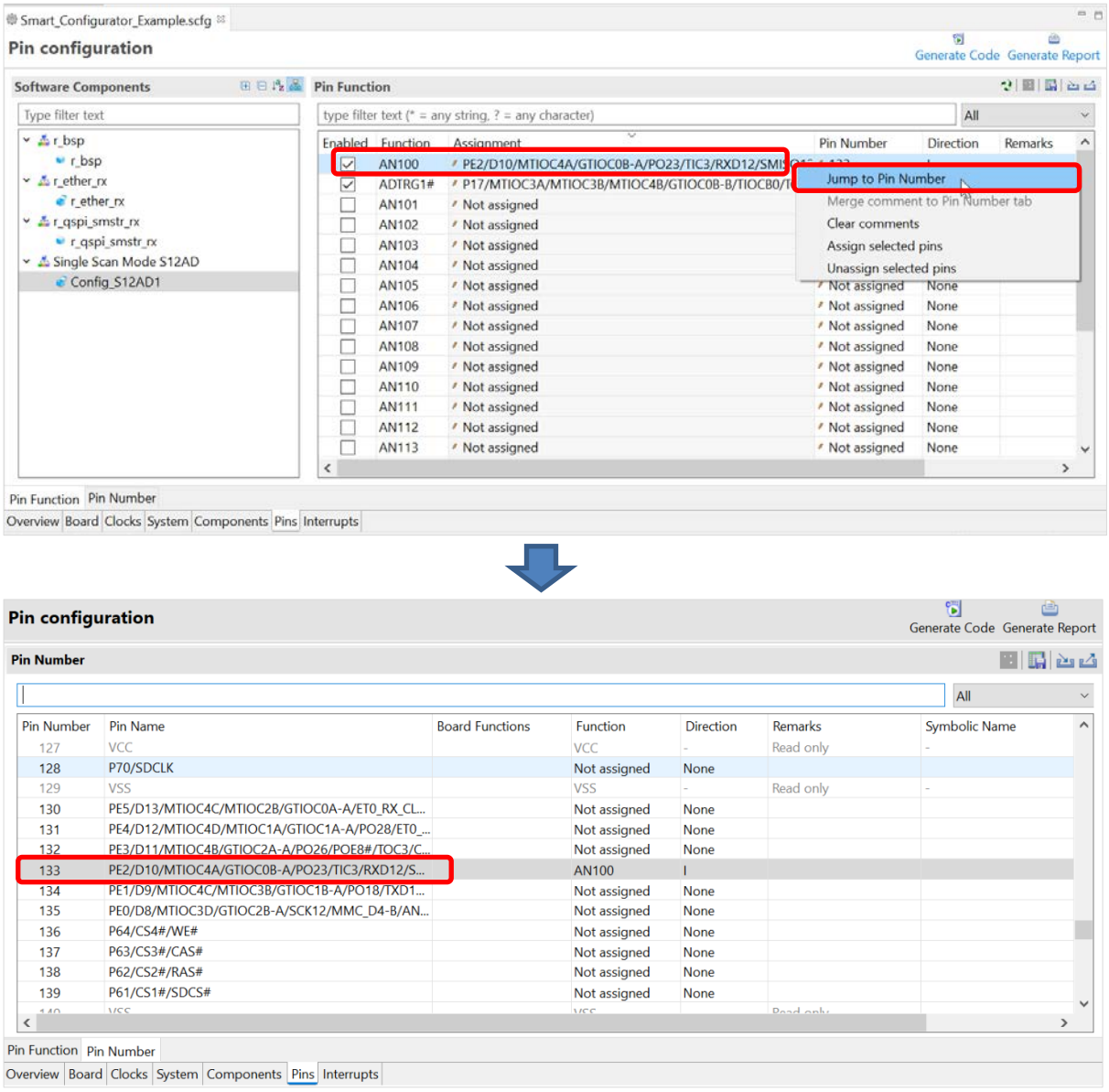


Figure 4-64 Jump to pin number

4.5.4 Exporting pin settings

The pin settings can be exported for later reference. Follow the procedure below to export the pin settings.

(1) Click on the [(Export board setting)] button on the [Pins] page.

(2) Select the output location and specify a name for the file to be exported.

The exported XML file can be imported to another project having the same device part number.

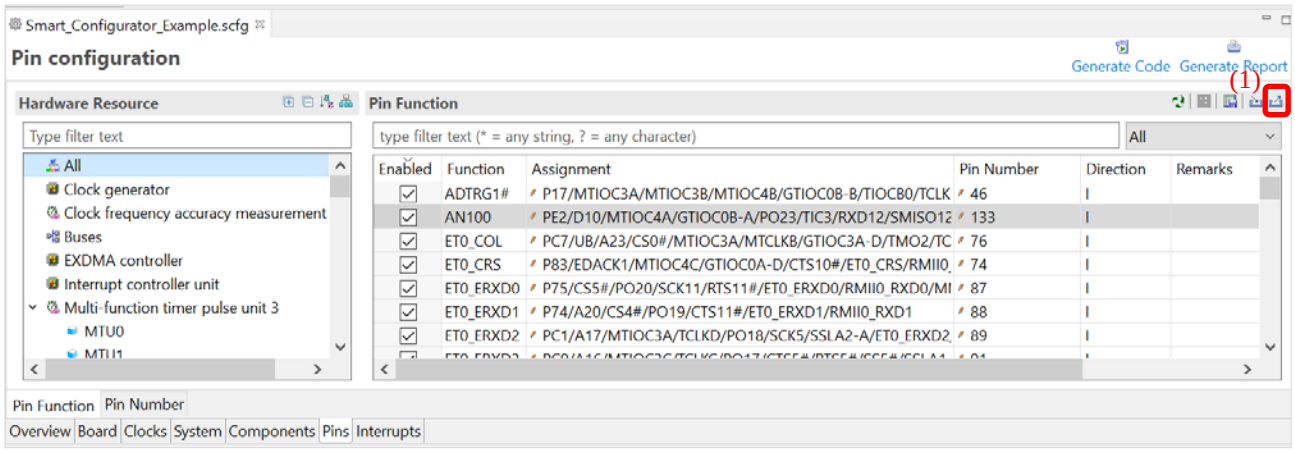


Figure 4-65 Exporting Pin Settings to an XML File

The Smart Configurator can also export the pin settings to a CSV file. Click on the [(Save the list to .csv file)] button on the [Pins] page.

4.5.5 Importing pin settings

To import pin settings into the current project, click on the [(Import board setting)] button and select the XML file that contains the desired pin settings. After the settings specified in this file are imported to the project, the settings will be reflected in the [Pin configuration] page.

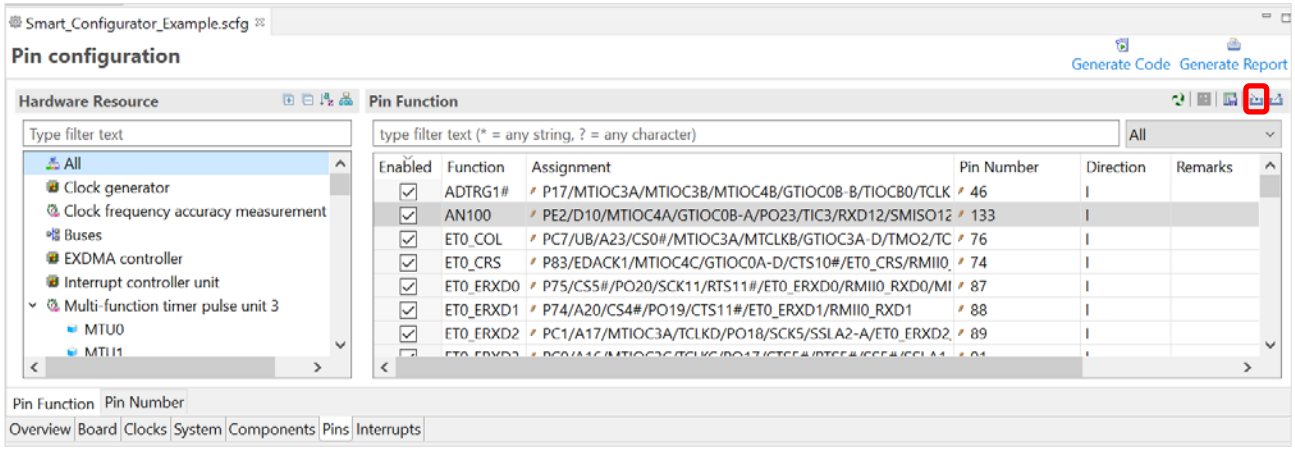


Figure 4-66 Importing Pin Settings from an XML File

Note: The pin setting is reflected, but it is not reflected in the component setting.

4.5.6 Pin setting using board pin configuration information

You can set the initial pin configuration according to the Renesas board that you selected to use. You can check the board that selected to use in [Board] tabbed page.

The following describes the procedure for collective setting of pins.

(1) Select [Board Function] in the MCU/MPU Package. (The initial pin configuration of the board can be referred.)

(2) Open the [Pin Configuration] page and click the [Assign default board pins] button.

(3) When [Assign default board pins] dialog opens, click [Select all].

(4) Click [OK].

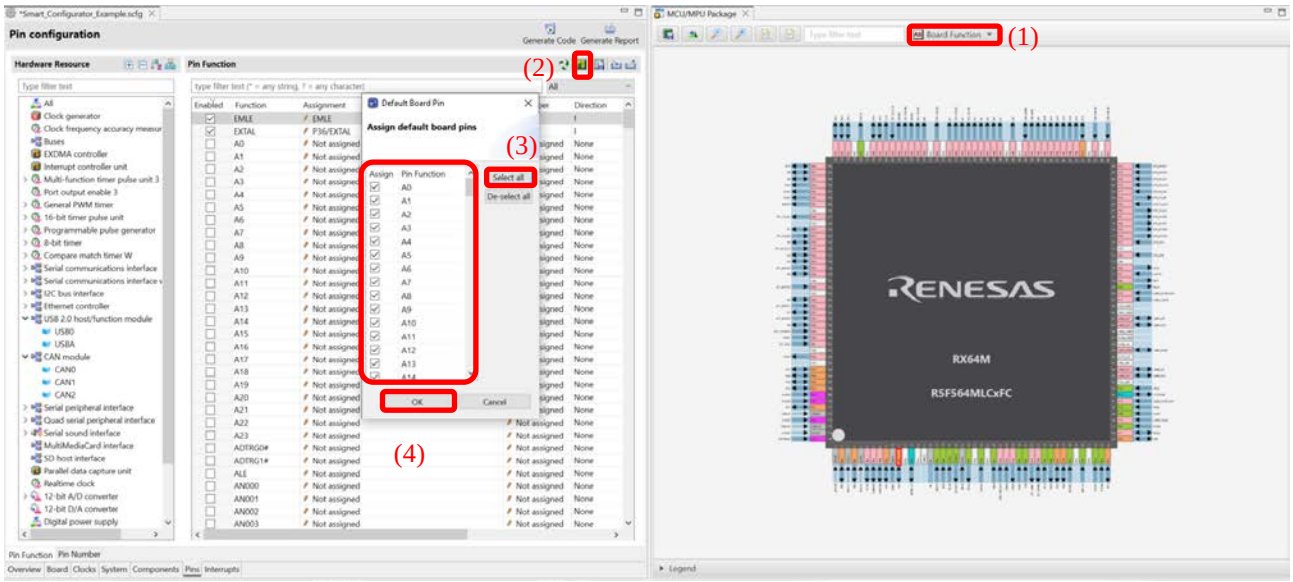


Figure 4-67 Setting for initial pin configuration

If you do not set pin settings all at once, specify them individually in procedure (3).

4.5.7 Pin filter feature

By specifying the filter range on the [Pin Function] tab and [Pin Number] tab on the [Pins] page, you can refer to it more easily.

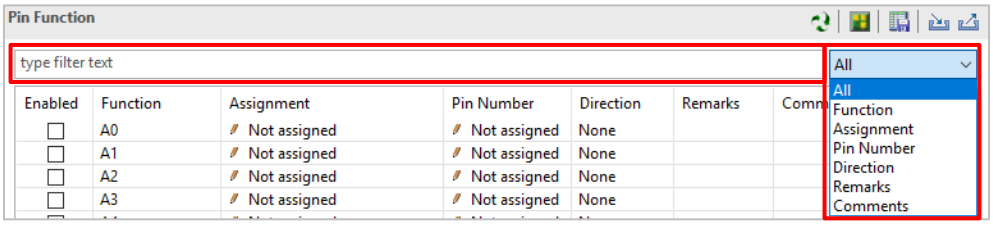


Figure 4-68 Filter for [Pin Function] tab

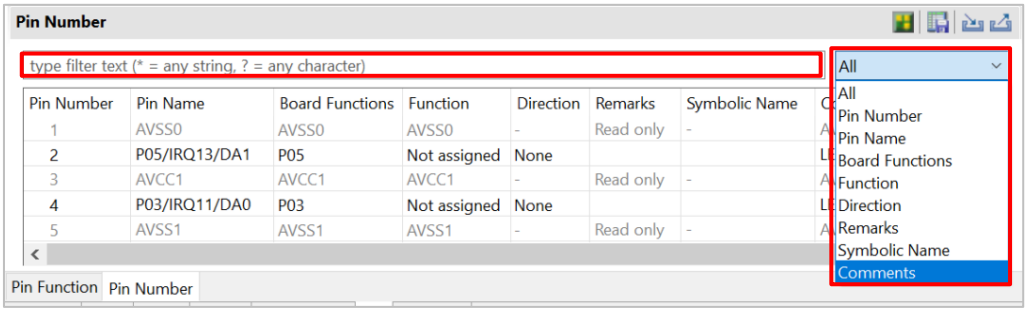


Figure 4-69 Filter for [Pin Number] tab

4.5.8 Pin Errors/Warnings setting

You can control how pin problem is displayed on Configuration Problems view by using the Pin Errors/Warnings setting. If you want to control it, on the [New Component] dialog, click the [Configure general settings...] link to display the [Preferences] dialog. Then select [Renesas] > [Smart Configurator] > [Pin Errors/Warnings] and use the combo boxes to change the errors/warning setting.

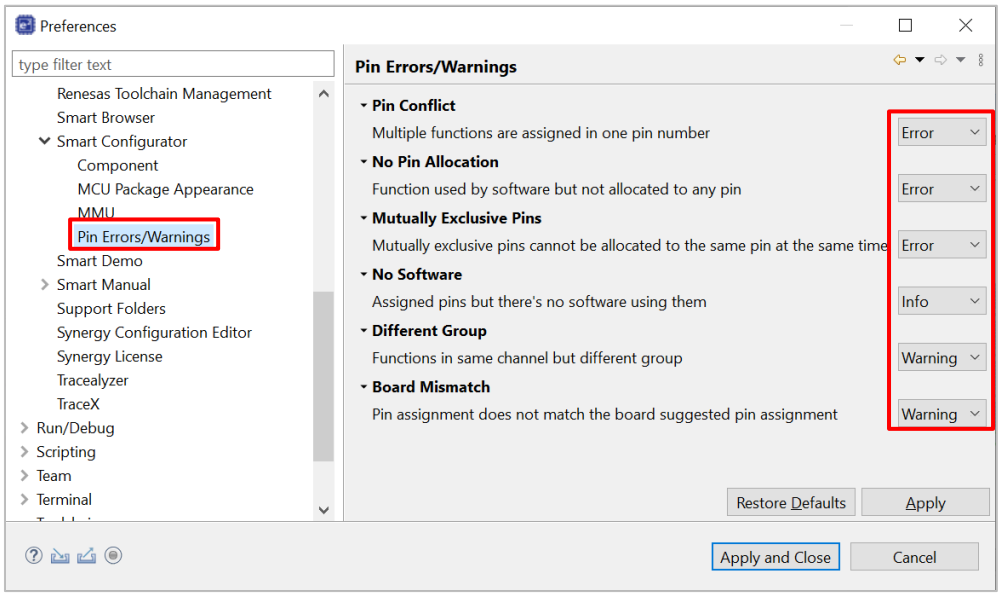


Figure 4-70 Pin Errors/Warnings settings at Preferences

Example: Change “No Software” setting from “Info” to “Error”

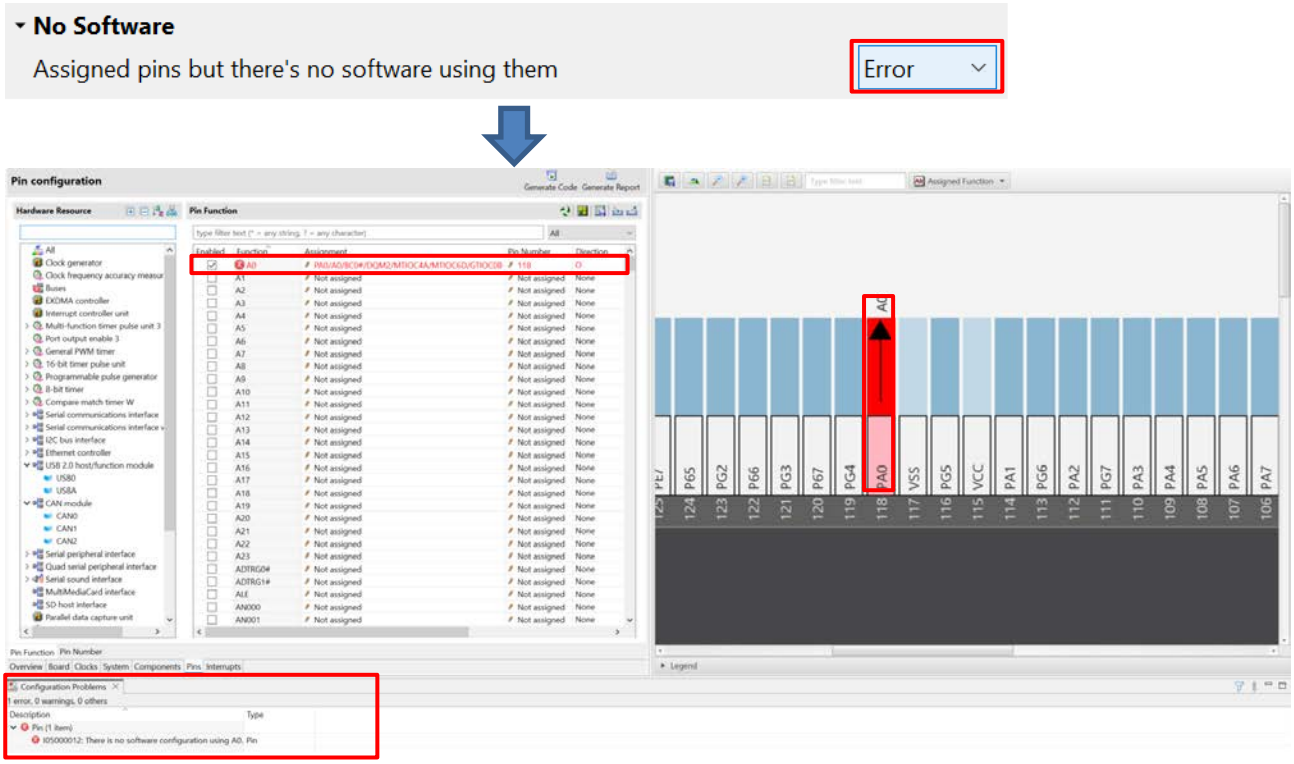


Figure 4-71 Change “No Software” setting from “Info” to “Error”

4.5.9 Symbolic name setting

[Symbolic Name] is an attribute of pins and can be found in [Pin Number] page and [MCU/MPU Package] page. It allows users to utilize their own symbols. The use of symbolic names in the user's application allows the source code to remain unchanged even when the MCU is changed, and pin assignments remapped. When a symbolic name is entered into the Pin page or the MCU/MPU Package view for any port pin, a macro definition will be generated in the Pin.h file

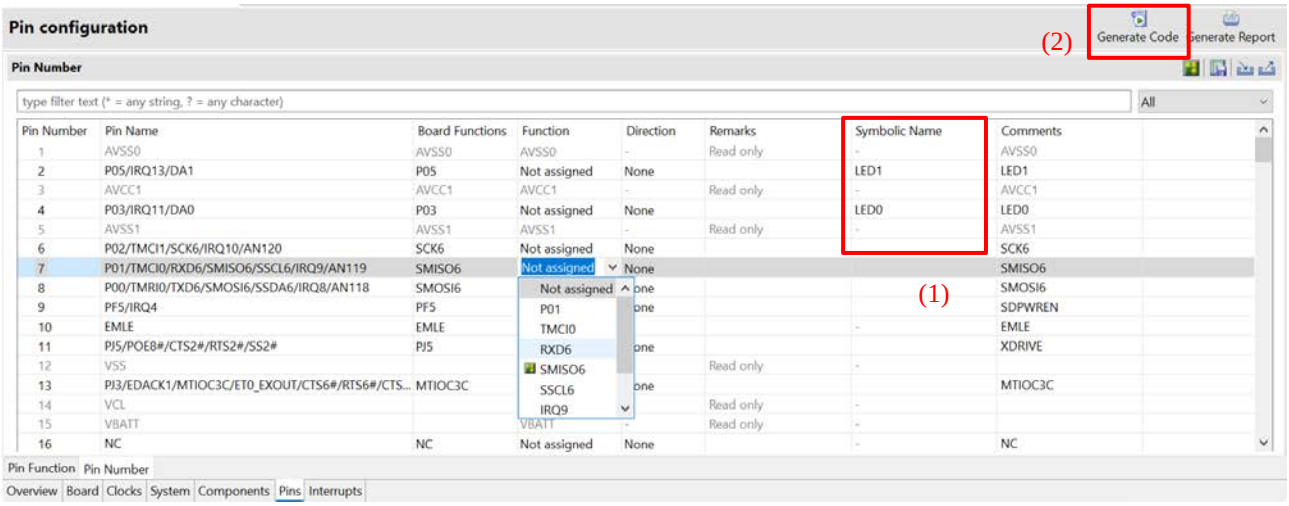


Figure 4-72 pin setting of symbolic name

After generating code, check at Pin.h file.

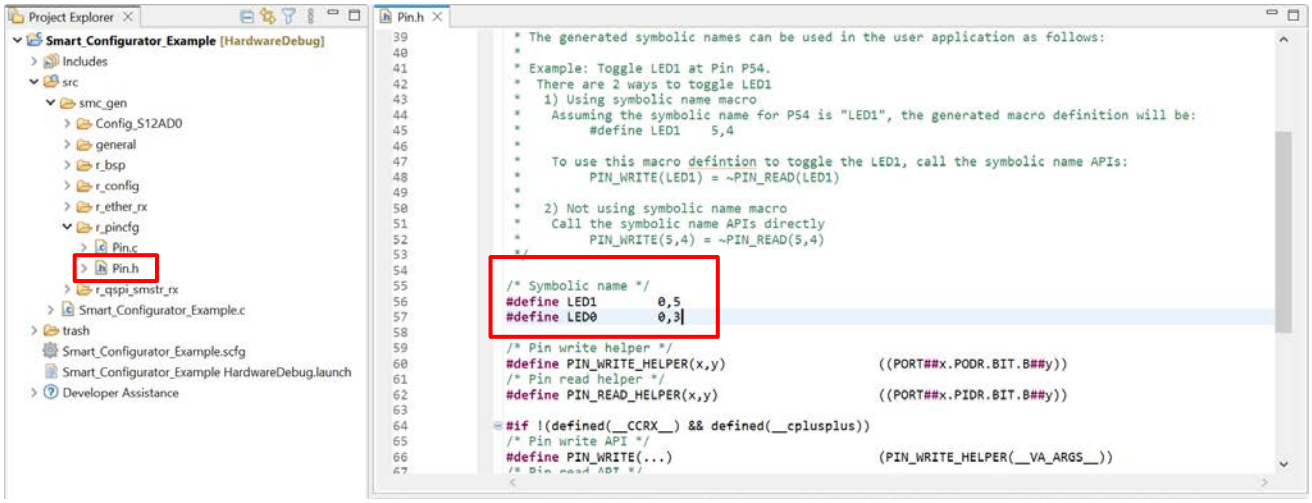


Figure 4-73 Symbolic Name in generated code

```
void main(void)
{
     // Set port direction
    PORTB.PDR.BYTE = 0x08U;
    PORTE.PDR.BYTE = 0x20U;

    //Init LED status
    PIN_WRITE(LED1) = 1U;
    PIN_WRITE(LED0) = 0U;

    //Toggle LEDs
    while(1){
        PIN_WRITE(LED0) = ~PIN_READ(LED1);
        PIN_WRITE(LED1) = ~PIN_READ(LED0);

        //Toggle LEDs
        for (int i = 0;i < 100000;i++){
            nop();
        }
    }
}
```

Figure 4-74 Using Symbolic Name in main function

4.6 Interrupt Settings

Check and set the interrupts of the peripheral modules that have been selected on the [Components] page. The interrupts are displayed for each of the vector numbers. Set the interrupt priority levels, the source of the fast interrupt, or a dynamic interrupt vector number.

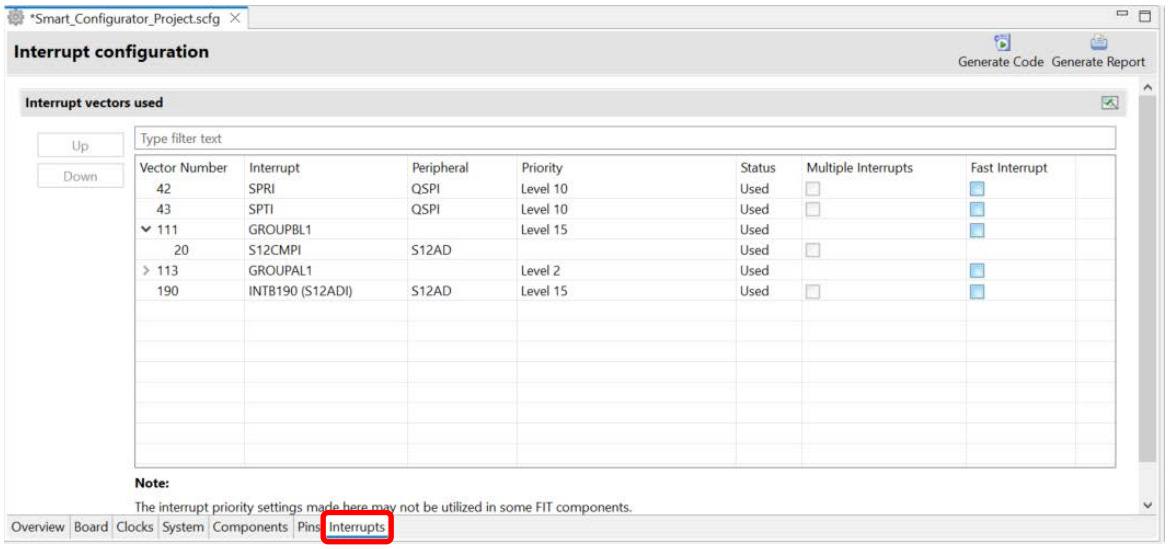


Figure 4-75 [Interrupts] Page

4.6.1 Changing the interrupt priority level and fast interrupt setting

When an interrupt is used in a CG configuration on the [Components] page, the status of the interrupt will be changed to "Used". To display the used interrupts only, click on the [(Show used interrupts)] button.

(1) You can change the interrupt priority level on the [Interrupts] page.

(2) To use an interrupt as a fast interrupt, tick the checkbox in the [Fast Interrupt] column. Only one interrupt can be specified as a fast interrupt among all interrupts and components used.

(3) Group interrupts are collapsed in the interrupt table. Click on the [(Open)] button to expand the view and see the interrupts in the group interrupt list.

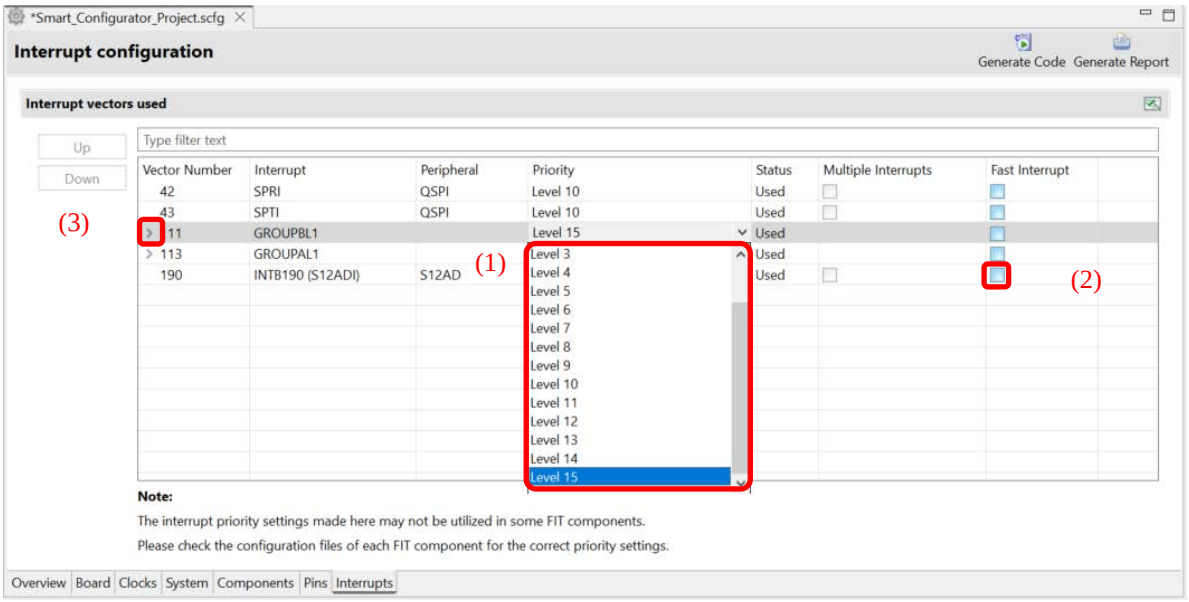


Figure 4-76 Interrupt Settings

Note:

The function for setting up interrupts is not available for the FIT modules.

For the method of setting up interrupts for each FIT module, refer to the application note in the <ProjectDir>¥src¥smc_gen¥r_xxx¥doc folder for the FIT module.

4.6.2 Changing the interrupt vector number

The [Interrupt configuration] page enables you to change the vector numbers of software configurable interrupts A and B.

(1) Select a desired software configurable interrupt.

(2) The [Up] and [Down] buttons will be enabled. Click on a button to change the vector number.

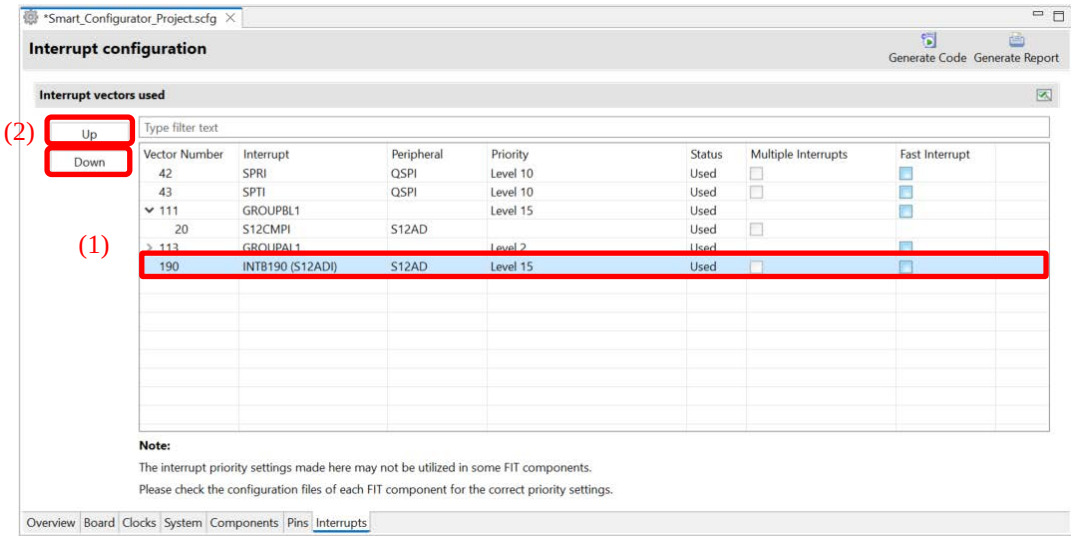


Figure 4-77 Changing the Vector Number of Software Configurable Interrupt A or B

4.6.3 Multiple interrupts setting

The multiple interrupt feature on the RX MCU allows the processing of another interrupt while the current interrupt is running. The setting of multiple interrupts can be configurated from both the Interrupt page and the Component configuration.

(1) Select a component(supported multiple interrupt) and enable its multiple interrupts settings.

(2) Multiple interrupts setting is bidirectional synchronization in [Interrupts] page.

(3) Open generated file in project explorer, generated code can be found.

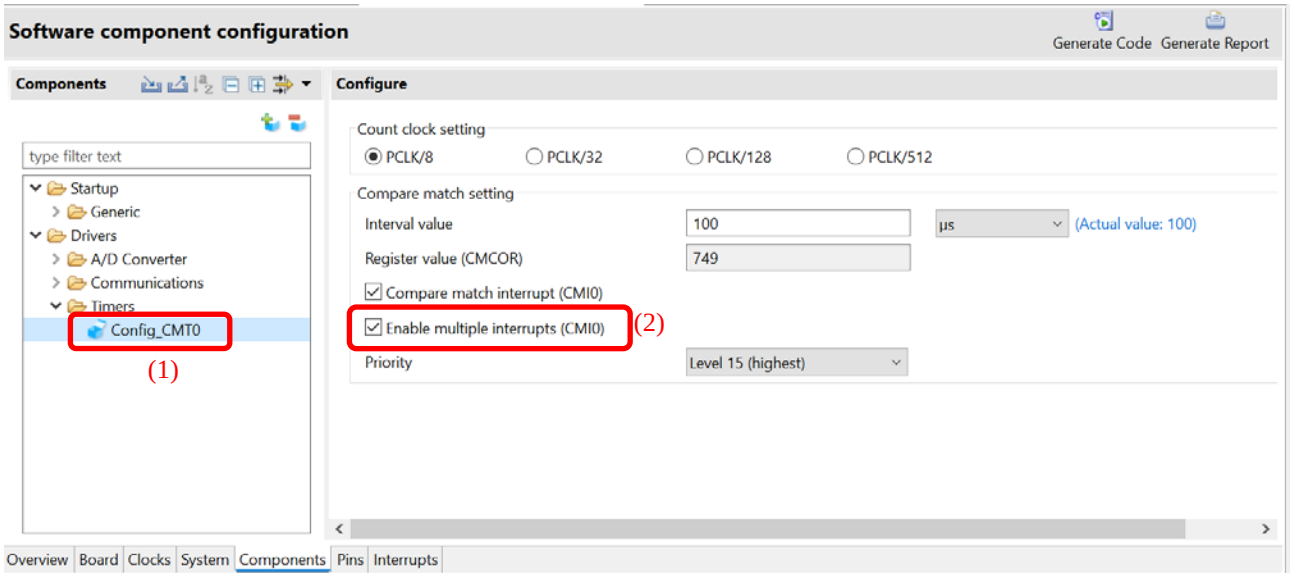


Figure 4-78 Multiple interrupts in component page

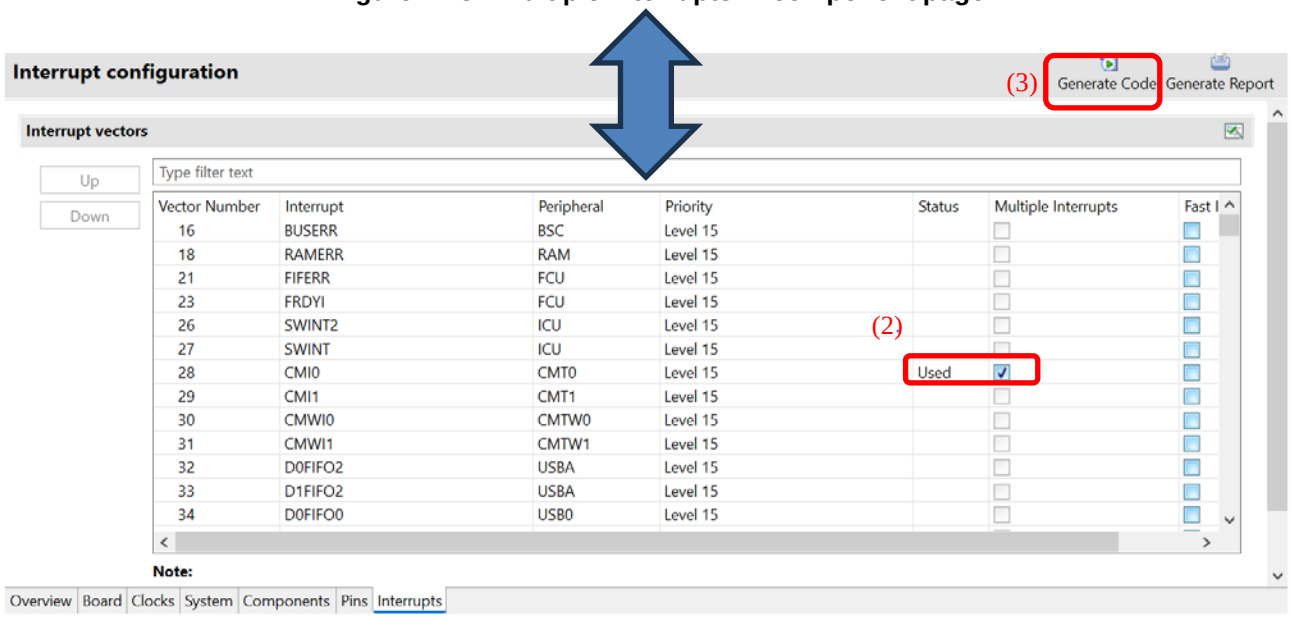


Figure 4-79 Multiple interrupts in component in Interrupts page

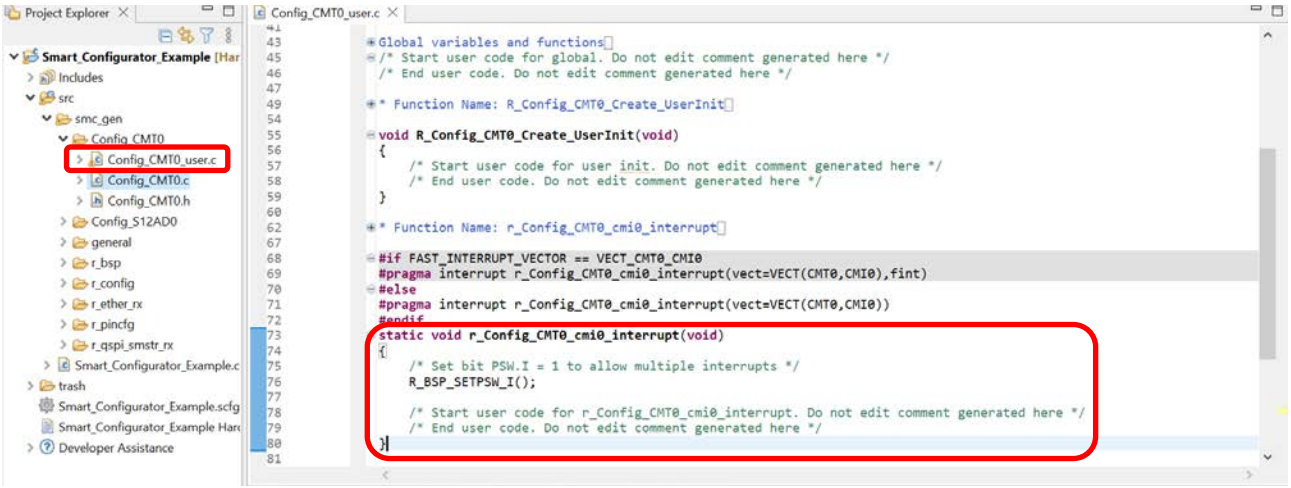


Figure 4-80 Multiple interrupts in generated code

4.7 MCU migration feature

The MCU migration feature helps to convert your project settings from device A to device B. Conversion of project settings can be done within the same family and can be done from e² studio project menu as follows.

Note: Project settings may change due to device change.
Back up the project before executing the device change.

(1) Select the project and choose [Change Device] from the [Project] menu.

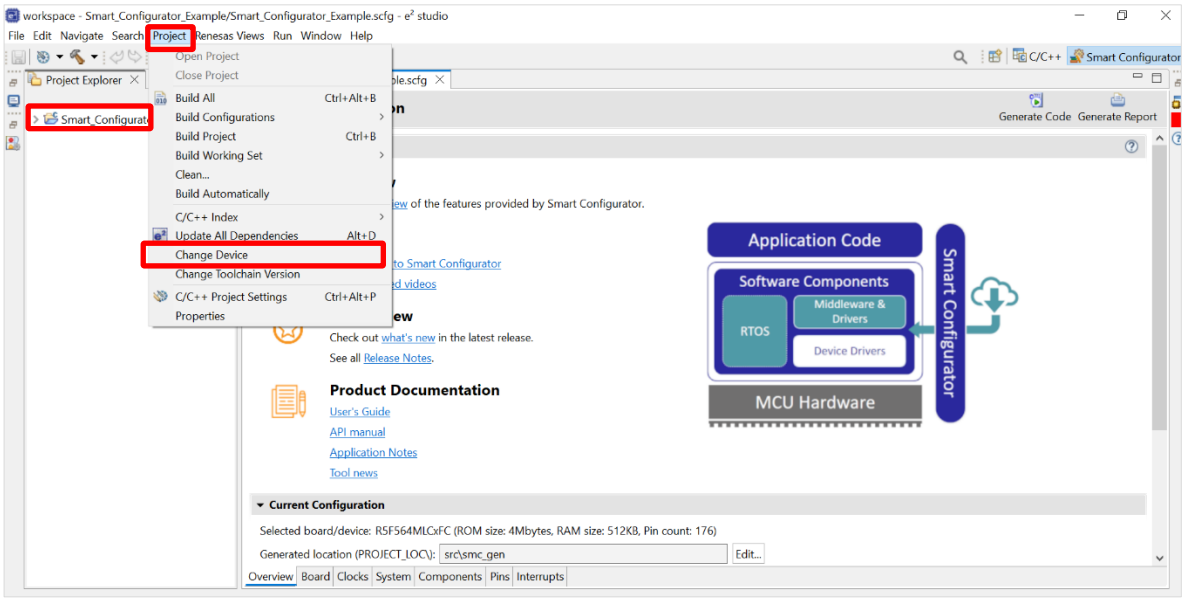


Figure 4-81 Select [Change Device]

(2) Select the target board from the board list, the device will be auto selected.

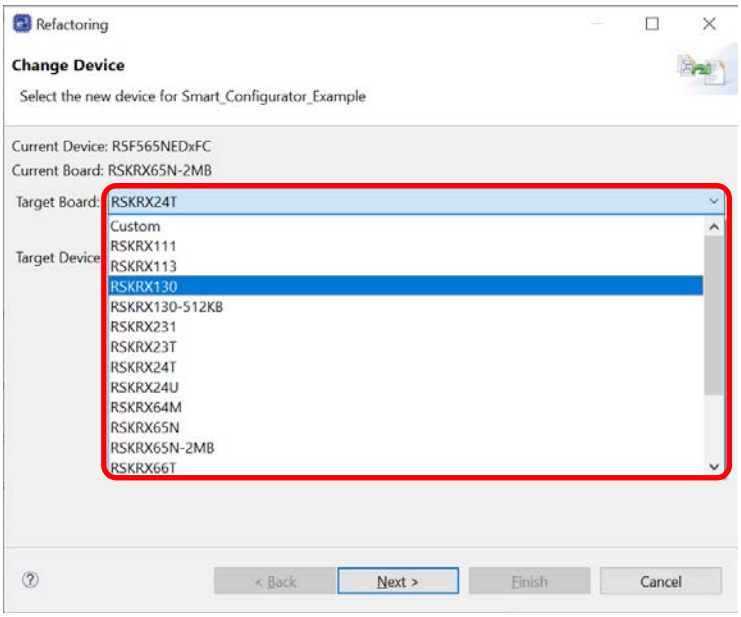


Figure 4-82 Select target board

If you want to remain target board as “Custom”, select the target device manually from the device selection list and click "OK". (Wild card search is supported)

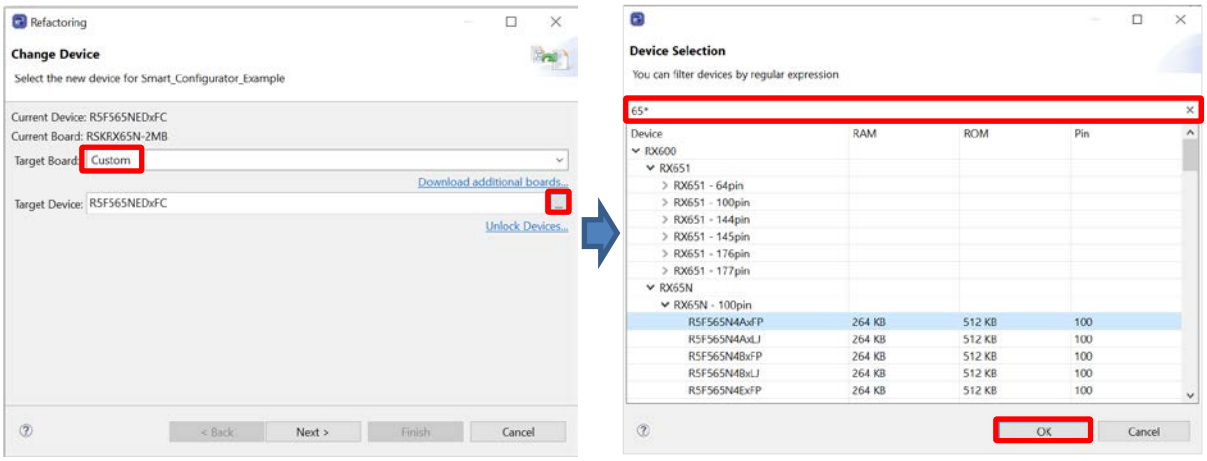


Figure 4-83 Select target device

(3) Confirm the message displayed in [Found problems] and click [Next].

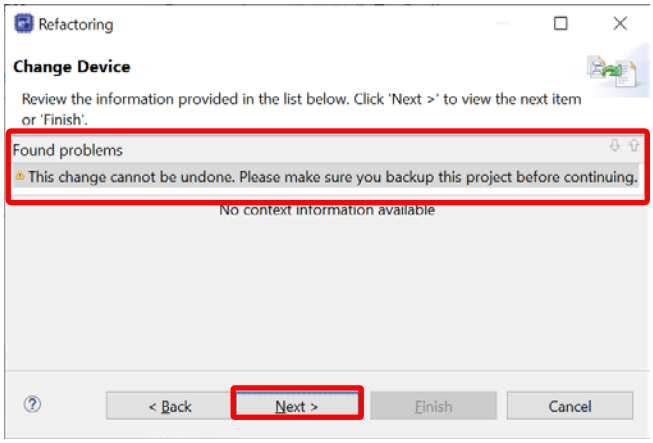


Figure 4-84 Found problems

Message	Explanation
Target device is not supported by Smart Configurator.	Displayed before changing to a device not supported by the Smart Configurator. You can’t convert Smart Configurator, but you can convert Project, Builder, Linker, Debugger.
This change cannot be undone. Please make sure you backup this project before continuing.	If you change the device, it can’t be restored before change, so please execute it after backing up the project.

(4) Confirm items to be changed and click “Finish”.

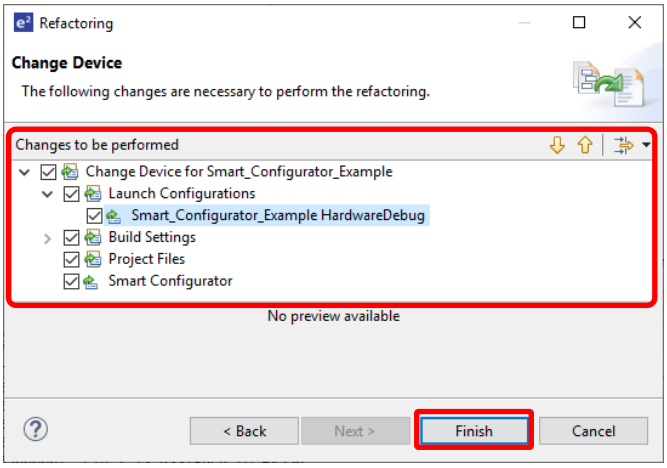


Figure 4-85 Changes to be performed

(5) The device name on the [Overview] page is updated.

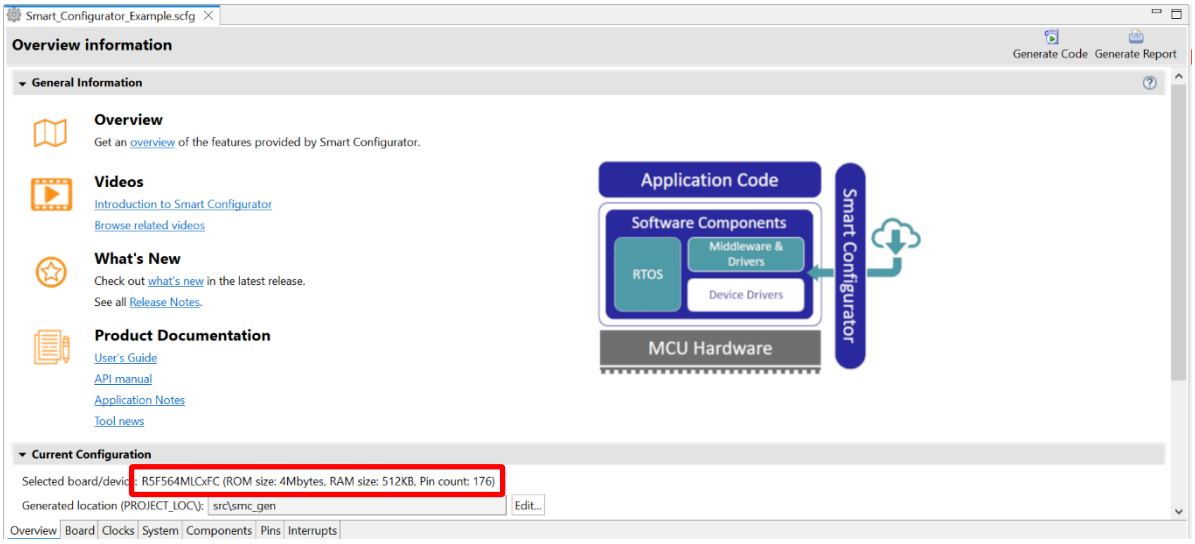


Figure 4-86 Device Update Confirmation

(6) A report of the configurations' conversion status is generated out in the console.

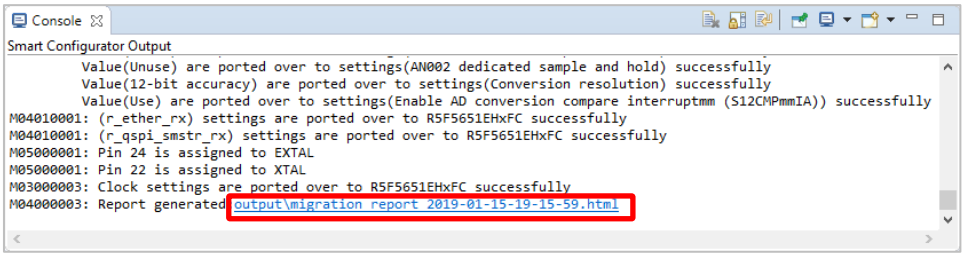


Figure 4-87 Configuration conversion status report

5. Managing Conflicts

When adding a component or configuring a pin or interrupt, problems in terms of resource conflict and missing dependency modules might occur. This information will be displayed in the Configuration Problems view. You can refer to the displayed information to fix the conflict issues.

5.1 Resource Conflicts

When two software components are configured to use the same resource (for e.g. S12AD0), an error mark ( ) will be displayed in the Components tree.

The Configuration Problems view will display messages on peripheral conflicts to inform in which software configurations peripheral conflicts have been detected.

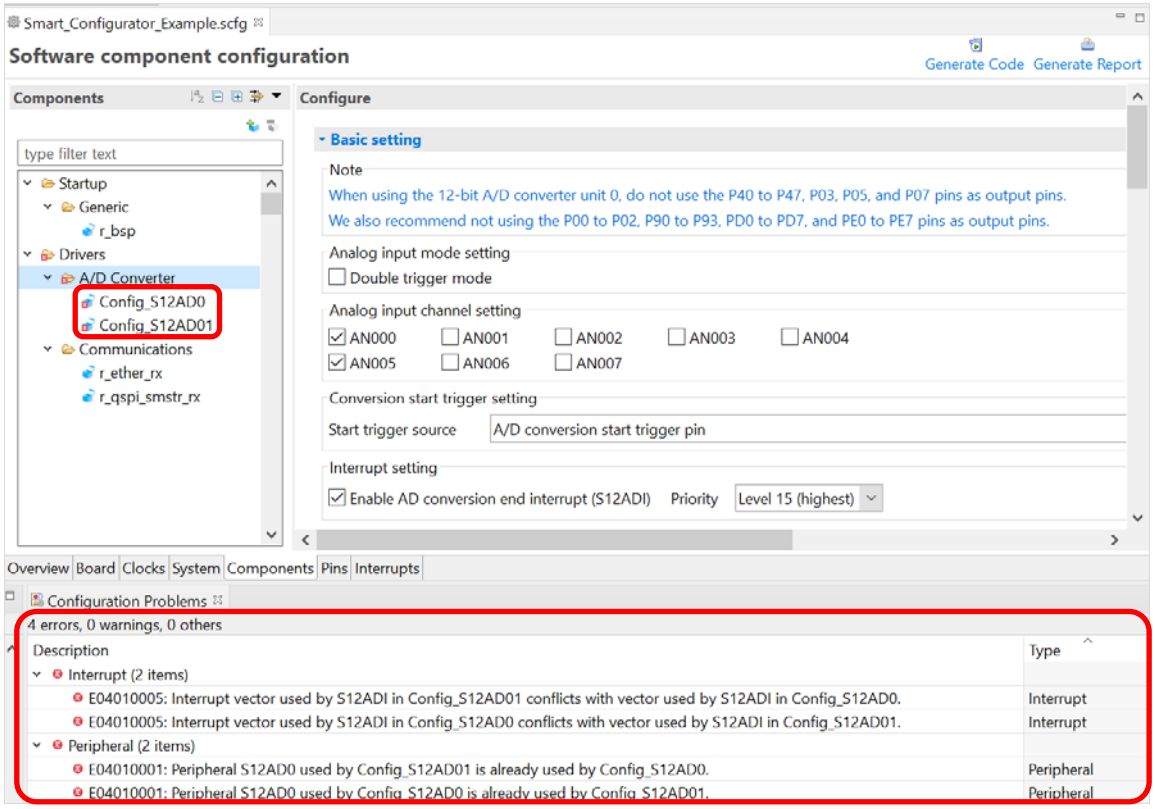


Figure 5-1 Resource Conflicts

5.2 Resolving pin conflicts

If there is a pin conflict, an error mark will appear on the tree and [Pin Function] list.

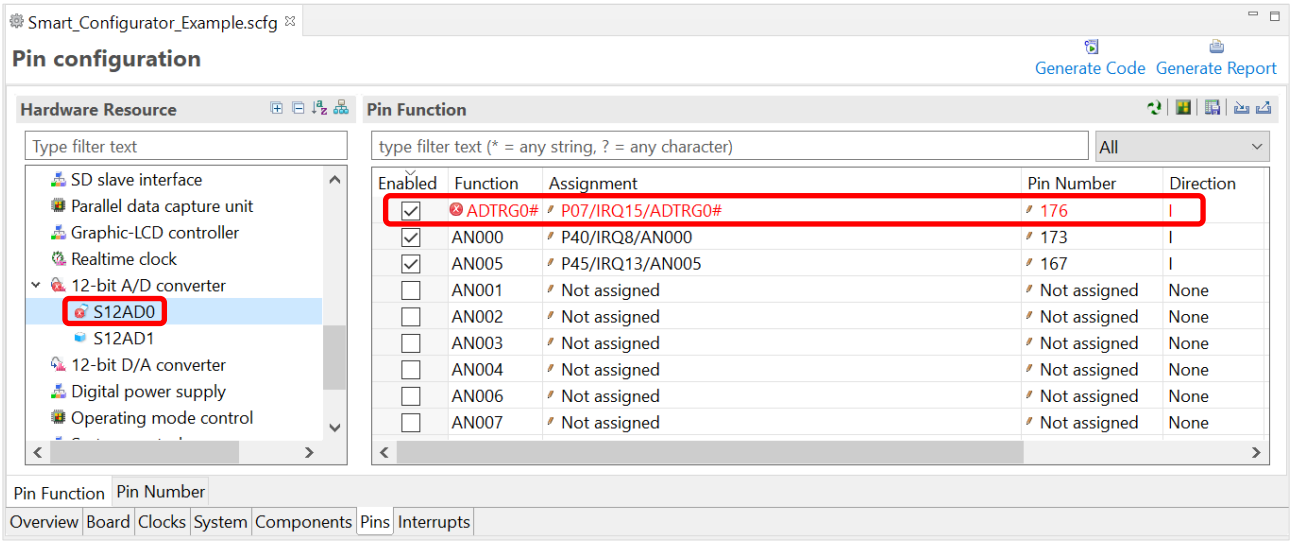


Figure 5-2 Pin Conflicts

The detailed information regarding conflicts is displayed in the Configuration Problems view.

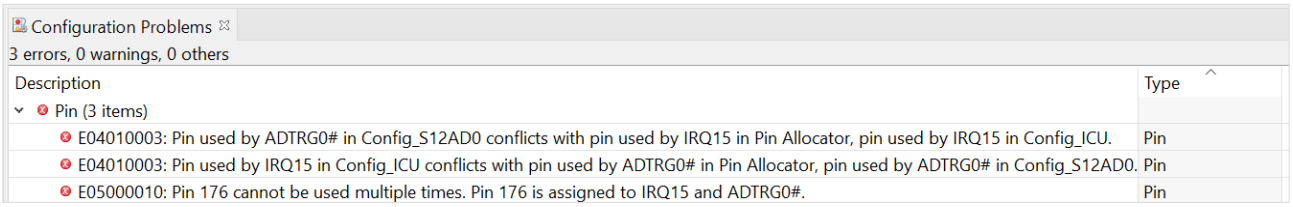


Figure 5-3 Pin Conflict Messages

To resolve a conflict, right-click on the node with an error mark on the tree and select [Resolve conflict].

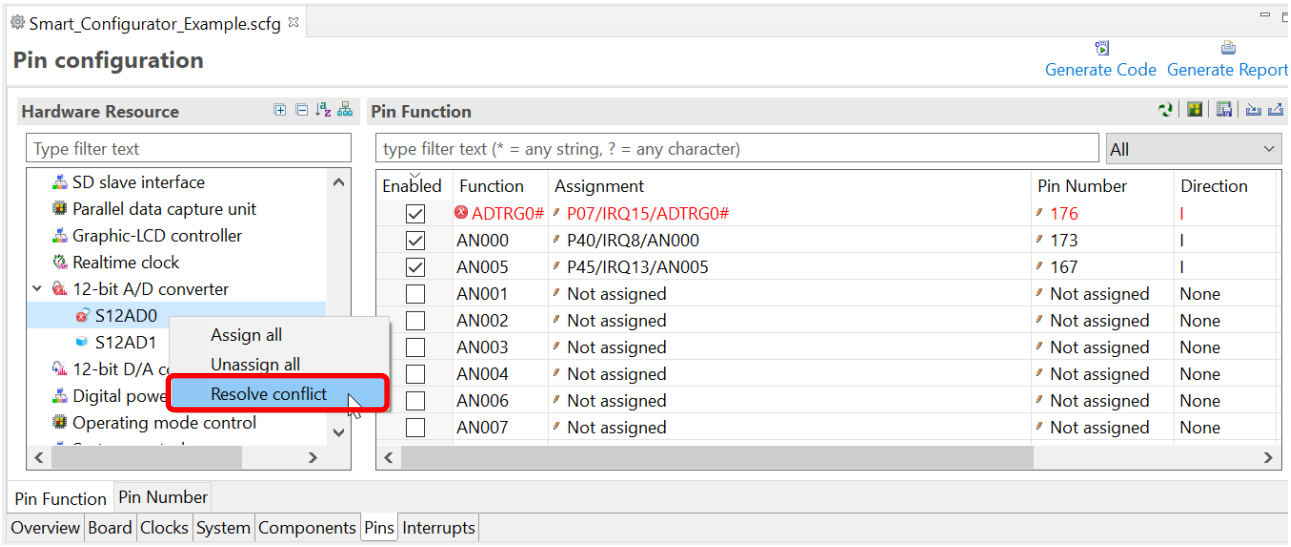


Figure 5-4 Resolving Pin Conflicts

The pins of the selected node will be re-assigned to other pins.

6. Generating Source Code

6.1 Outputting Generated Source Code

Output a source file for the configured details by clicking on the [(Generate Code)] button in the Smart Configurator view.

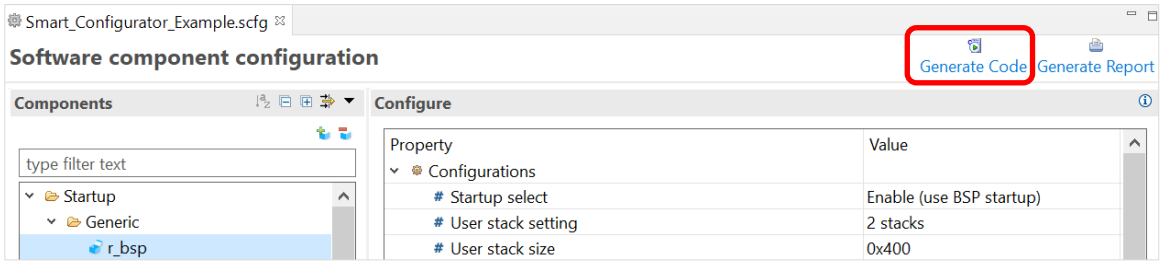


Figure 6-1 Generating a Source File

The Smart Configurator generates a source file in <ProjectDir>¥src¥smc_gen and updates the source file list in the Project Explorer. If the Smart Configurator has already generated a file, a backup copy of that file is also generated (refer to chapter 8, Backing up Generated Source Code).

Note: If you put a self-created source file in sms_gen folder, it will be erased at time of generating source code.

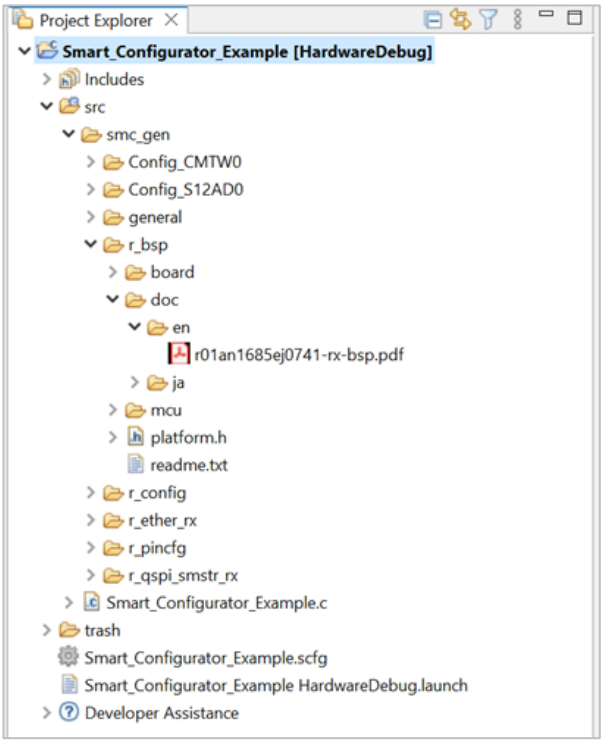


Figure 6-2 Source Files in the Project Explorer

6.2 Change Generated Code Location

(1) To change the generated code location, click on the [Edit] button under Current Configurations at [Overview] page.

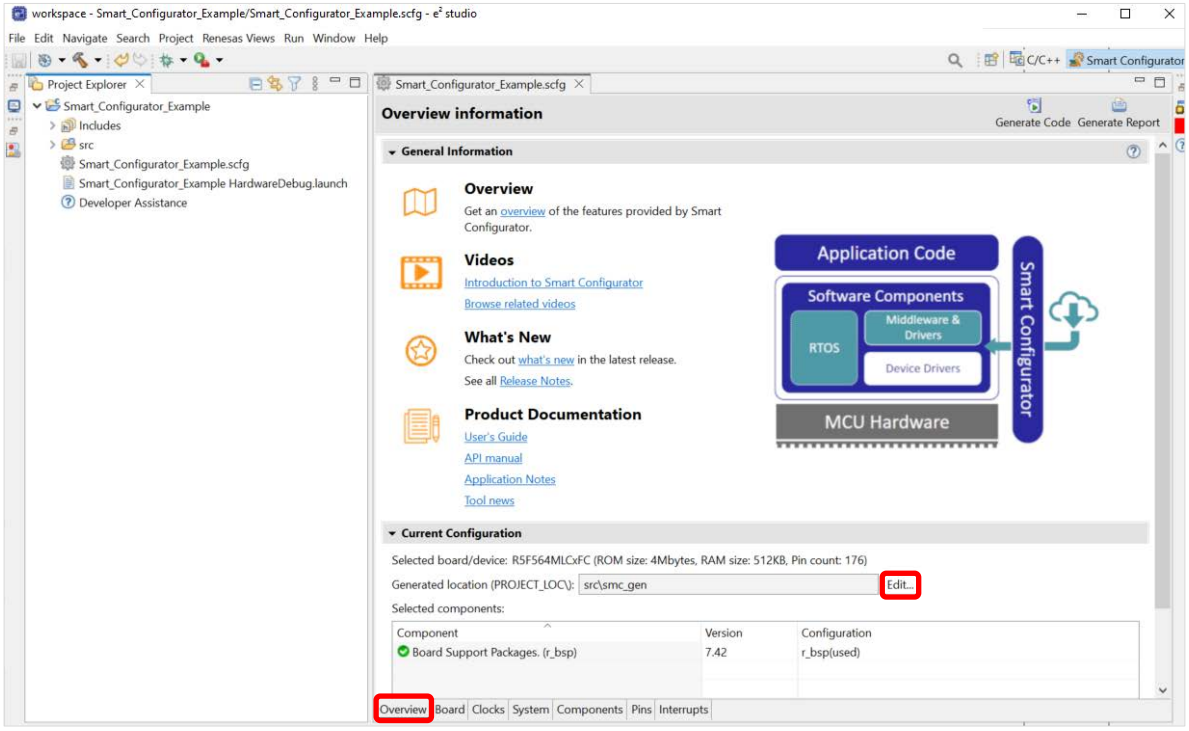


Figure 6-3 Edit the Generated Code Location

(2) In the Folder Selection dialog, select an empty folder for code generation or create a new folder.

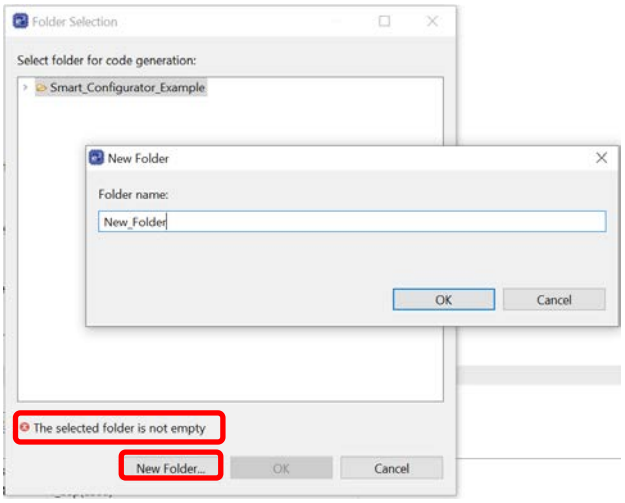


Figure 6-4 Folder Selection

(3) Click on [Generate Code] button. The source code will be generated in the new location. You can also check for the current generate code location in [Overview] page.

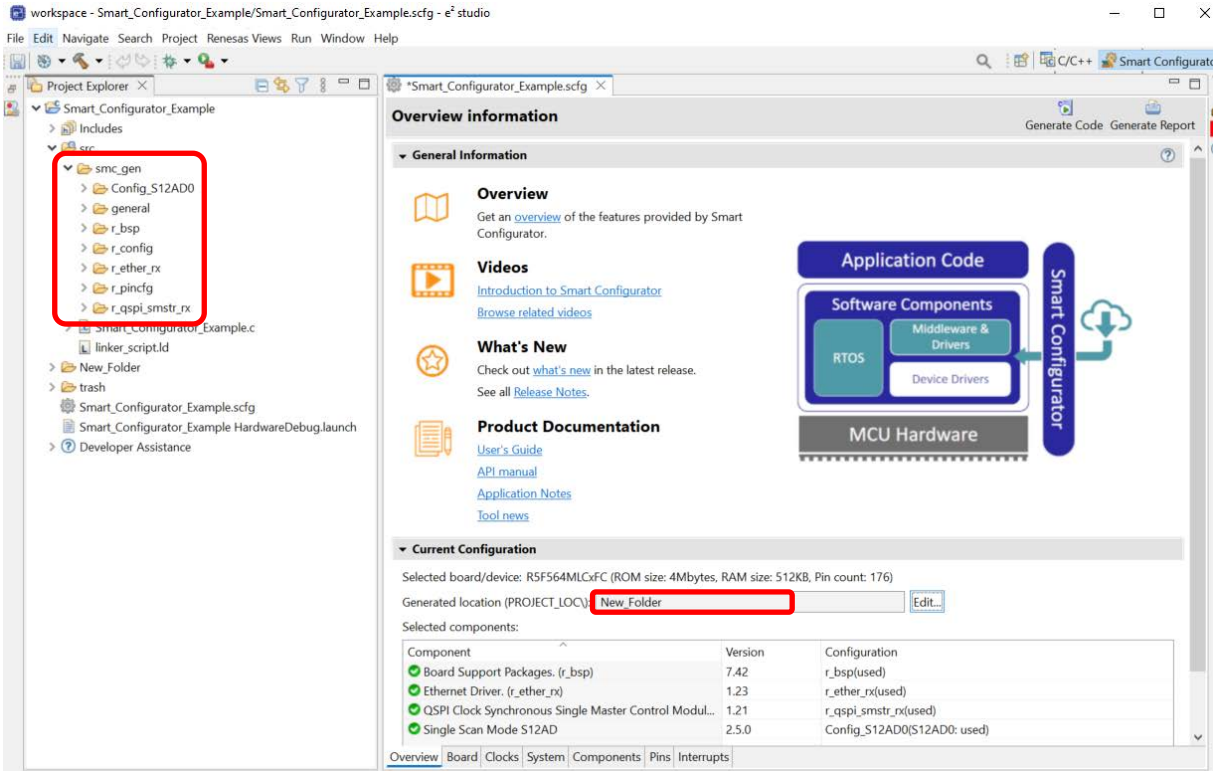


Figure 6-5 New generate code location

6.3 Configuration of Generated Files and File Names

The below figure “Configuration of Generated Files and File Names”, shows the folders and files output by the Smart Configurator. Function *main()* is included in *{Project name}.c,* which is generated when the project is created by the
e² studio.

r_xxx indicates the names of FIT modules, “ConfigName” indicates the name of the configuration formed by the component settings, and “Project name” indicates a project name set in the e² studio.

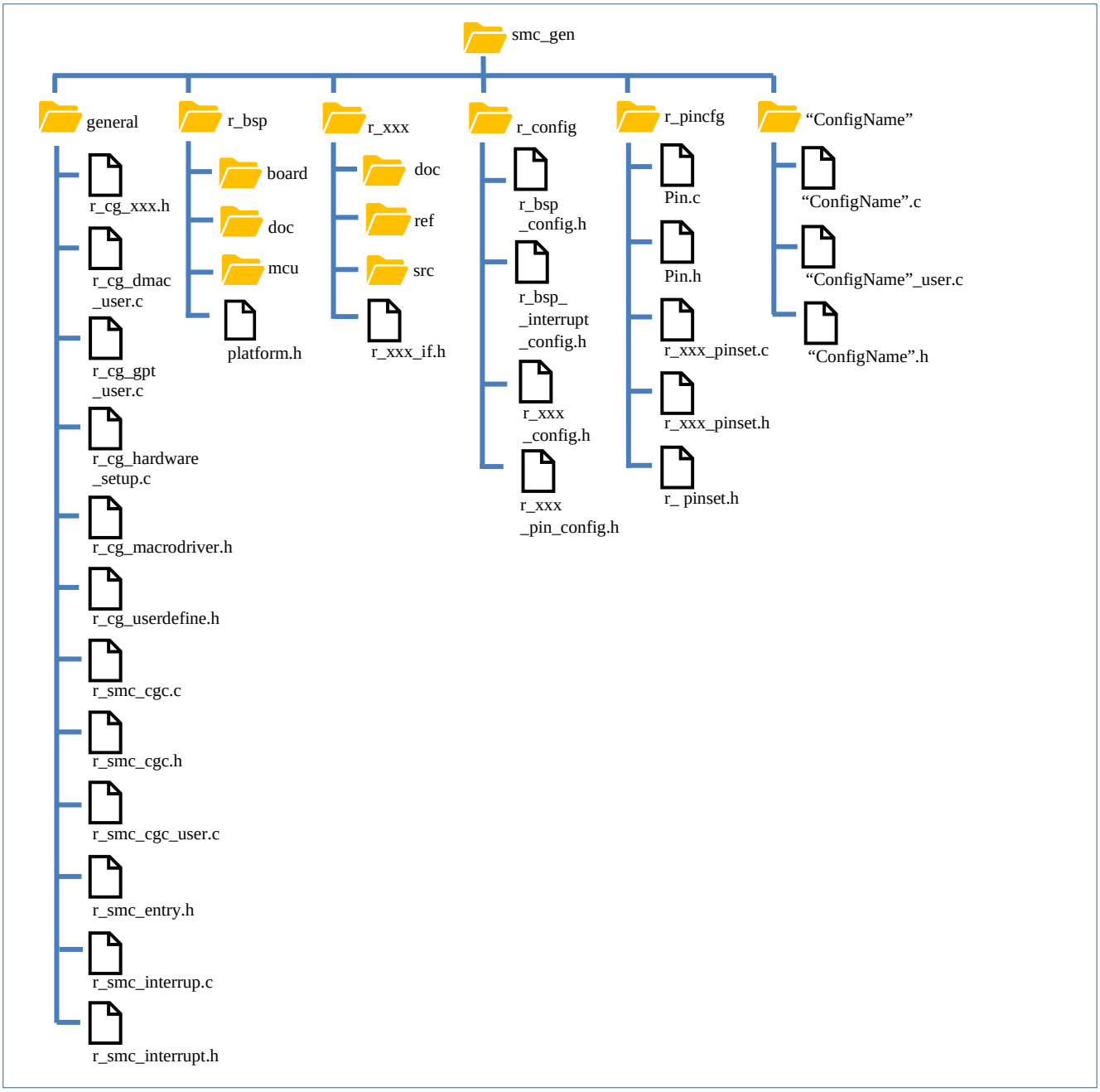


Figure 6-6 Configuration of Generated Files and File Names

Folder	File	Description
general		This folder is always generated. It contains header files and source files commonly used by CG drivers of the same peripheral function.
	r_cg_xxx.h(Note*1)	These files are always generated. The files contain macro definitions for setting SFR registers.
	r_cg_dmac_user.c	This file is always generated for a device with a DMAC function. It contains interrupt service routines and callback functions shared among some DMAC channels (depending on the hardware specifications).
	r_cg_gpt_user.c	This file is always generated for a device with a GPT function. It contains interrupt service routines and callback functions shared among some GPT channels (depending on the hardware specifications).
	r_cg_hardware_setup.c	This file is always generated. It contains *R_Systeminit* that calls all driver initialization functions with the name *R_ConfigName_Create.* *R_Systeminit* also calls the functions for initializing clocks other than the clock source, fast interrupt, and group interrupts.
	r_cg_macrodriver.h	This file is always generated. This header file contains common macro definitions used in drivers.
	r_cg_userdefine.h	This file is always generated. User can add macro definitions in the dedicated user code areas.
	r_smc_cgc.c	This file is always generated. It contains the initialization of clock sources other than the clock source selected in the [Clocks] page.
	r_smc_cgc.h	This file is always generated. This header file contains macro definitions to initialize clocks other than the selected clock source.
	r_smc_cgc_user.c	This file contains functions to be added to *R_CGC_Create*. User can add codes and functions in the dedicated user code areas.
	r_smc_entry.h	This file is always generated. This file includes the header files of CG drivers that are added to the project. When using functions of CG drivers in source files added by user, including this file is necessary.
	r_smc_interrupt.c	This file is always generated. It contains fast interrupt and group interrupt initialization (depending on hardware specification).
	r_smc_interrupt.h	This file is always generated. It contains macro definitions for fast interrupt and group interrupt initialization. It also contains the priority level of all interrupts that are configured in the [Interrupts] tabbed page. User can use these macro definitions in application codes.

r_bsp		This folder is always generated. It consists of multiple subfolders (*board, doc, mcu*) with: - Initialization codes to start up the MCU before entering *main()* (e.g. setup stack, initialize memory) - Definitions of all SFR registers in *iodefine.h* (*mcu* folder) - Application note of *r_bsp* It also contains *platform.h* that will include *r_bsp.h* of the device used in the project.

Folder	File	Description
r_xxx *(Note*1)*		This folder is generated for the FIT module that is added to the project. It consists of: - *doc* folder: Application note of this FIT module - *ref* folder: Reference of FIT module configuration file and pin configuration file - *src* folder: FIT module source files and header files - *r_xxx_if.h* *(Note*1)*: List of all API calls and interface definitions of this FIT module Note: Folders in *r_xxx* depends on the requirements of each FIT module.
r_config		This folder is always generated. It contains configuration header files for the MCU package, clocks, interrupts, and driver initialization functions with the name *R_xxx_Open* *(Note*1)*.
	r_bsp_config.h	This file is always generated. It contains configurations of *r_bsp* for clock initialization and other MCU related settings. Some MCU related settings are generated by Smart Configurator (e.g. package type) and other settings (e.g. stack size) are configured by user manually.
	r_bsp_interrupt_config.h	This file is always generated. It contains mapping of the software configurable interrupts A and B (depending on hardware specification).
	r_xxx_config.h(Note*1)	These are configuration header files for all FIT drivers that are added to the project. This file is configured by user manually.
	r_xxx_pin_config.h(Note*1)	These pin configuration header files are dedicated for FIT drivers with specific requirements in pin setting sequence.
r_pincfg	*Pin.c*	This file is always generated. It is a reference of pin function initialization for all peripherals configured in the [Pins] tabbed page (except I/O Ports).
	Pin.h	This file is always generated. It contains the function prototypes of pin settings in *Pin.c*
	r_xxx_pinset.c *(Note*1)*	This file contains pin function initialization for the FIT drivers that are added to the project. API function in this file is for user to call in the application codes.
	r_xxx_pinset.h *(Note*1)*	This file contains pin setting function prototypes in *r_xxx_pinset.c*
	r_pinset.h	This file includes all pin setting header files named with *r_xxx_pinset.h* *(Note*1)* in *r_pincfg* folder.
{ConfigName}		This folder is generated for the CG drivers that are added to the project. API functions in this folder are named after the *ConfigName* (configuration name).
	{ConfigName}.c	This file contains functions to initialize driver (*R_ConfigName_Create*) and perform operations that are driver-specific, e.g. start (*R_ConfigName_Start*) and stop (*R_ConfigName_Stop*).
	{ConfigName}_user.c	This file contains interrupt service routines and functions for user to add code after the driver initialization (*R_ConfigName_Create*). User can add codes and functions in the dedicated user code areas.

	{ConfigName}.h	This is header file for *{ConfigName}.c and {ConfigName}_user.c*.

Note *1: xxx is the name of a peripheral function.

6.4 Initializing Clocks

Configurations of the clock source selected in the [Clocks] page are generated to the macros in the r_bsp_config.h file located in ¥src¥smc_gen¥r_config folder. Clock initialization codes will be handled by r_bsp before entering main().

The r_bsp_config.h file also contains other MCU related settings (for e.g. package, stack size).

Configurations of other clocks are generated in ¥src¥smc_gen¥general folder.

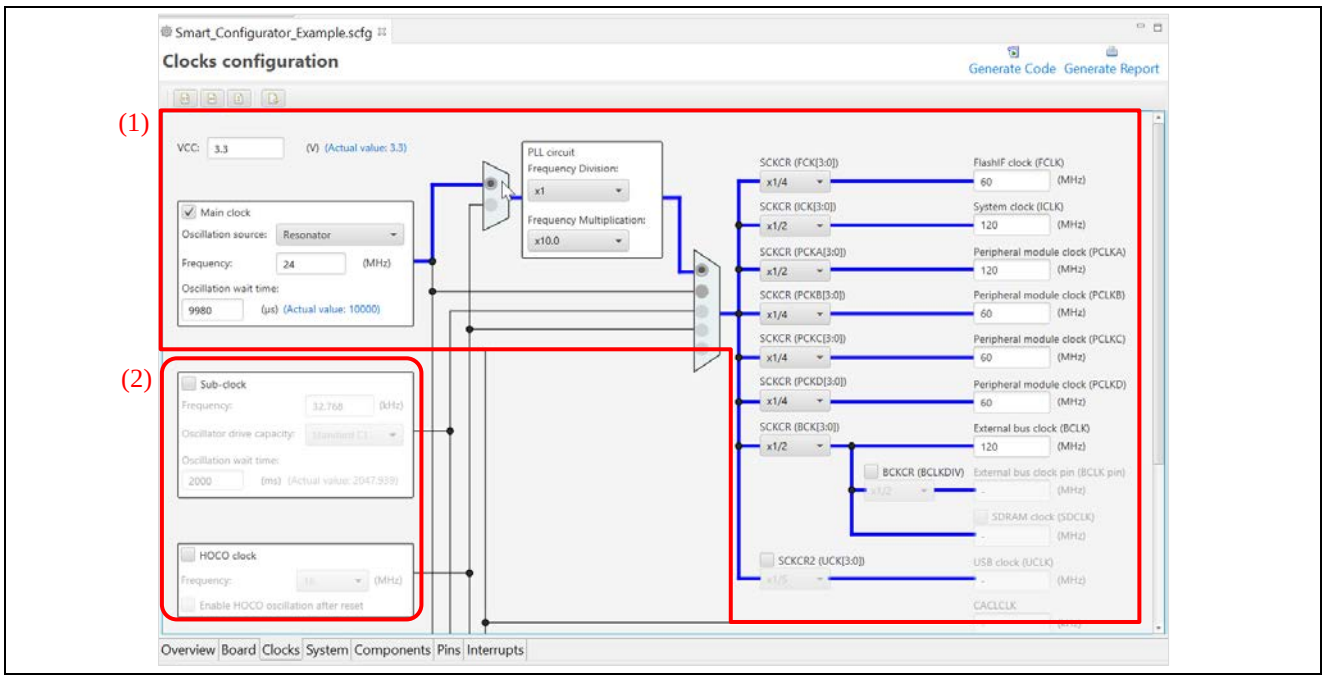


Figure 6-7 Clocks Configuration with Main Clock Selected as Clock Source

No	Folder	File	Macros/Functions	Description
(1)	**r_config**	*r_bsp_config.h*	Macros related to clocks	These settings are generated by Smart Configurator based on user's selection in the [Clocks] page for the clock source. Only one clock can be selected as the clock source at a time. *r_bsp* will handle the clock initialization before entering *main()*.
			Macros related to MCU settings	Some MCU related settings are generated by Smart Configurator (e.g. package type) and other settings (e.g. stack size) are configured by user manually. Refer to the application note in *r_bsp* folder before configuring these macros: *¥src¥smc_gen¥r_bsp¥doc*
(2)	**general**	*r_smc_cgc.c*	*R_CGC_Create*	This API function initializes clocks other than the selected clock source. *R_Systeminit* in *r_cg_hardware_setup.c* will call this function before entering *main()* function.
		r_smc_cgc.h	Macros related to clocks	These macros are for clock initialization in *R_CGC_Create*.
		r_smc_cgc_user.c	*R_CGC_Create_UserInit*	This API function is used to add code in *R_CGC_Create* after the CGC initialization.

r_bsp_config.h will be backed up to trash folder before each code generation (refer to chapter 8, Backing up Generated Source Code).

6.5 Initializing Pins

Configurations in the [Pins] page are generated in some source files depending on driver's requirements and hardware specifications.

(1) Pin initialization for drivers with *{ConfigName}*

Pin functions are initialized in *R_ConfigName_Create* of the file *¥src¥smc_gen¥{ConfigName}¥{ConfigName}.c*.

Pin initialization codes will be handled before entering *main()*.

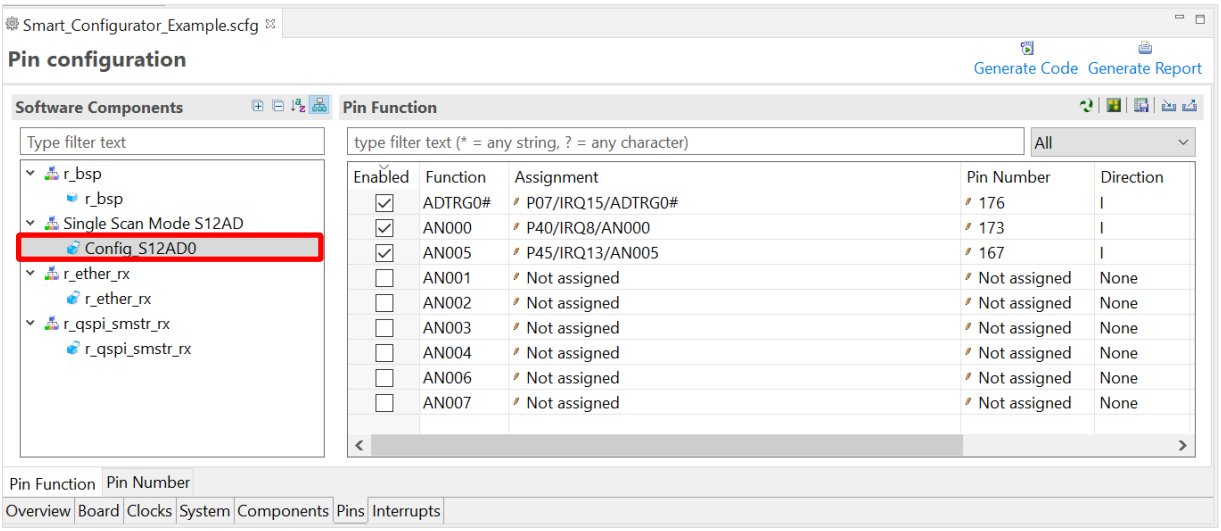


Figure 6-8 *Config_S12AD0* in Software Components View

Folder	File	Function	Driver	Description
{ConfigName}	*{ConfigName}.c*	*R_ConfigName_Create*	CG	This API function initializes the pins used by this driver. *R_Systeminit* in *r_cg_hardware_setup.c* will call this function before entering *main()* function.

(2) Pin initialization for drivers with *r_xxx* *(Note2)*

The pin setting source file will be generated in ¥src¥smc_gen¥r_pincfg folder with the name *r_xxx_pinset.c.*

The API functions in this file are called by the user from application codes.

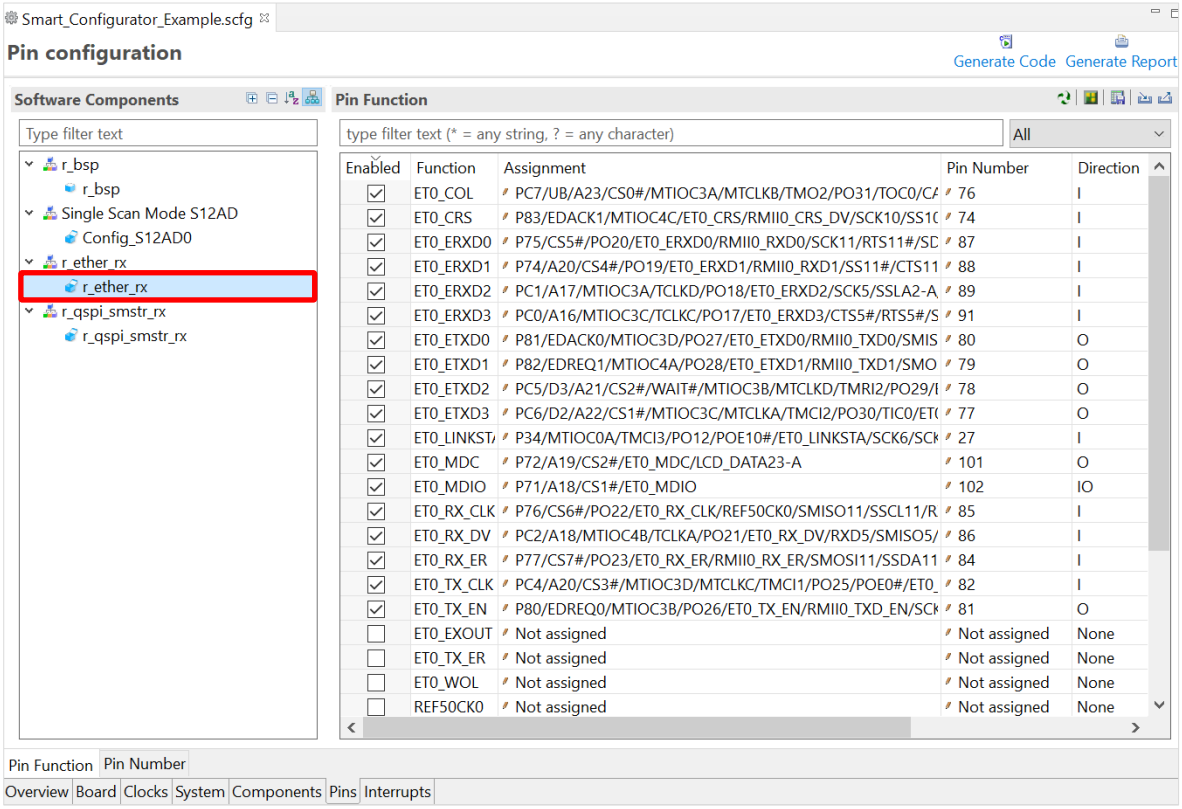


Figure 6-9 *r_ether_rx* in Software Components View

Folder	File	Function	Driver	Description
r_pincfg	*r_xxx_pinset.c* *(Note*2)*	*R_xxx_PinSet_xxxn* *(Note*2,3)*	FIT	This API function initializes the pins used by this driver. Refer to the application note in the corresponding *r_xxx* folder before calling this API function: *¥src¥smc_gen¥r_xxx¥doc* *(Note*2)*

Note *2: xxx is the name of a peripheral function.

*3: n is a peripheral channel number.

(3) Pin initialization for drivers with *r_xxx_smstr* *(Note4)*

The pin setting header file will be generated in ¥src¥smc_gen¥r_config folder with the name *r_xxx_smstr_rx_pin_config.h.*

The macro definitions in this file will be handled in the *r_xxx_smstr* source files.

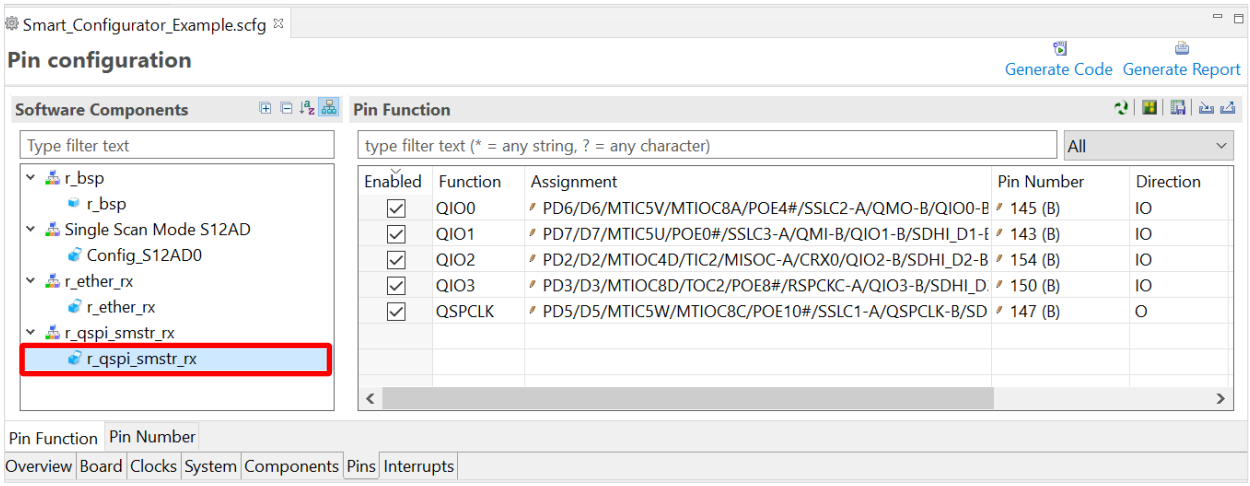


Figure 6-10 *r_qspi_smstr_rx* in Software Components View

Folder	File	Function	Driver	Description
r_config	*r_xxx_smstr_rx_pin_config.h* *(Note*4)*	-	FIT	Macro definitions in this header file initialize the pins used by this driver. These macros will be called in *r_xxx_smstr* source files.

Note *4: xxx is the name of a peripheral function.

(4) Reference to pin initialization codes

Refer to *Pin.c* in ¥src¥smc_gen¥r_pincfg folder for all peripheral pin functions used in the project (except I/O ports).

Folder	File	Function	Driver	Description
r_pincfg	*Pin.c*	*R_Pins_Create*	-	This file contains the initialization codes of all pin functions configured in the [Pins] page except I/O ports.

6.6 Initializing Interrupts

Configurations in the [Interrupts] page are generated in some source files.
Refer to the application note in the corresponding ¥src¥smc_gen¥r_xxx¥doc folder to initialize interrupts used in *r_xxx* modules (xxx is the name of peripheral function).

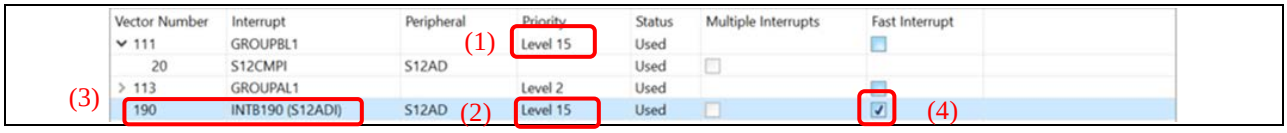


Figure 6-11 Interrupts Configuration in Interrupts View

No	Item	Folder	File	Driver	Description
(1)	Priority	**general**	*r_smc_interrupt.c*	CG	This interrupt priority level setting is for group interrupts *(Note5)*. It is initialized in *R_Interrupt_Create* of this file. *R_Systeminit* in *r_cg_hardware_setup.c* will call this function before entering *main()* function.
(2)	Priority	**{ConfigName}**	*{ConfigName}.c*	CG	This interrupt priority level setting is for normal interrupts and software configurable interrupts A and B *(Note5)*. It is initialized in *R_ConfigName_Create* of this file. *R_Systeminit* in *r_cg_hardware_setup.c* will call this function before entering *main()* function.
(3)	Vector Number	**r_config**	*r_bsp_interrupt_config.h*	CG FIT	Vector number of software configurable interrupts A and B *(Note5)* in the [Interrupts] tabbed page will be mapped in this file and handled by *r_bsp*.
(4)	Fast Interrupt	**general**	*r_smc_interrupt.c*	CG	Fast interrupt setting will be initialized in *R_Interrupt_Create* of this file. *R_Systeminit* in *r_cg_hardware_setup.c* will call this function before entering *main()* function.
			r_smc_interrupt.h	CG	Vector number of fast interrupt will be defined in this file. *{ConfigName}_user.c* will use this macro definition to prepare a fast interrupt service routine.
(1) (2)	Priority	**general**	*r_smc_interrupt.h*	-	Priority level of all interrupts configured in the [Interrupts] tabbed page is defined in this file. User can use these macro definitions in the application codes.

Note *5: The type of interrupt depends on hardware specifications.

6.7 Component Settings

6.7.1 FIT module configuration

1) Configuration for *r_bsp*

Configuration file of *r_bsp* is generated as *r_bsp_config.h* under the ¥src¥smc_gen¥r_config folder. It contains clock-initialization and other MCU-related settings (for e.g. the package).

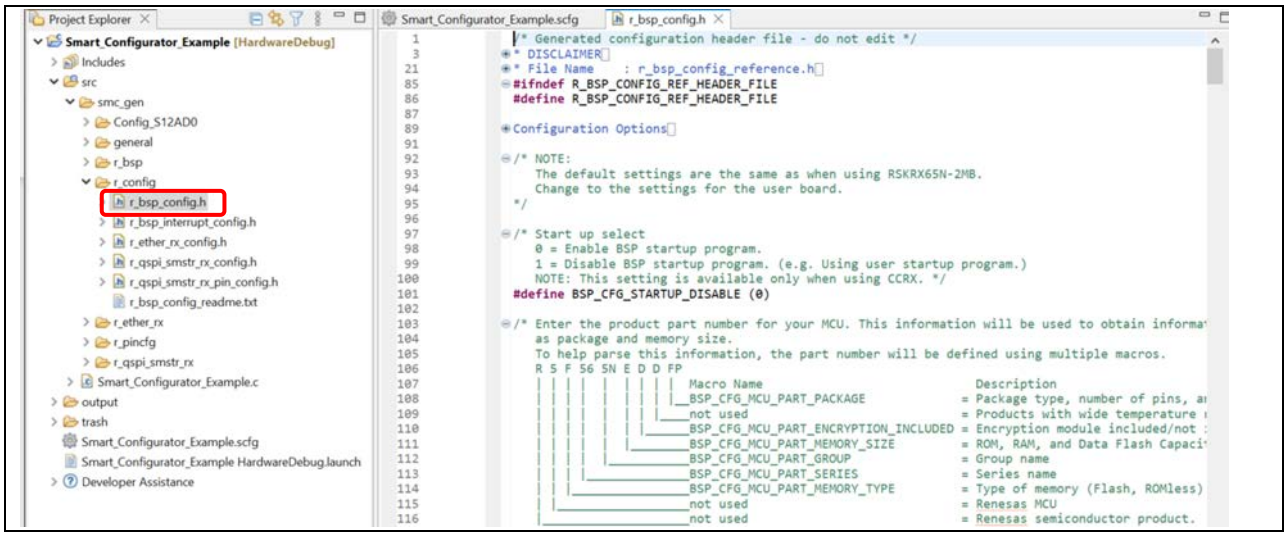


Figure 6-12 *r_bsp_config.h*

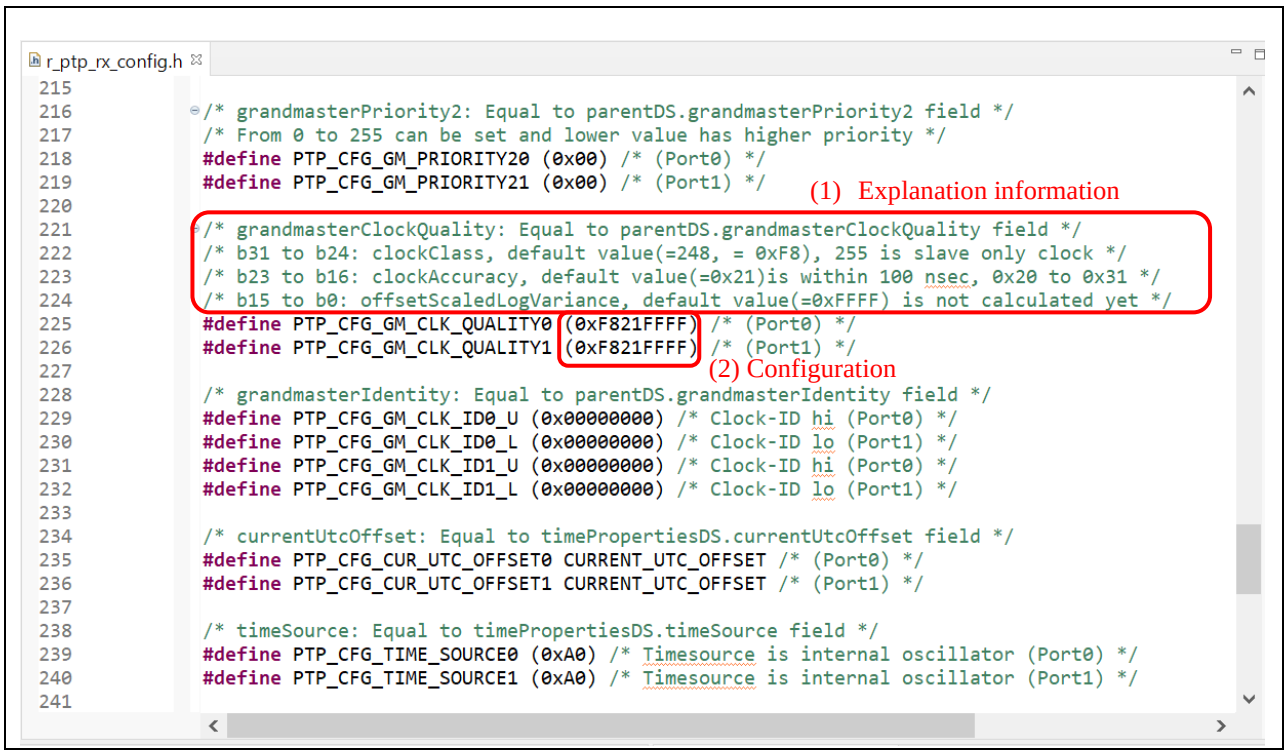
2) Configuration of FIT modules

Configuration files of FIT modules that are added to the project are generated as *r_xxx_config.h* under ¥src¥smc_gen¥r_config folder. (*r_xxx* is the name of FIT module)

For FIT modules that have configuration GUI at [Component] page, the configuration will be generated by Smart Configurator. Therefore, you do not need to change the *.h file manually.

For FIT modules that do not have configuration GUI at [Component] page, you need to modify these configurations at *.h file manually. As shown in the figure below, read *(1) Explanation information* before setting the macro definition value in *(2) Configuration*.

Refer to the application note in ¥src¥smc_gen¥r_xxx¥doc folder on how to modify *r_xxx_config.h*.




```
/* grandmasterPriority2: Equal to parentDS.grandmasterPriority2 field */
/* From 0 to 255 can be set and lower value has higher priority */
#define PTP_CFG_GM_PRIORITY20 (0x00) /* (Port0) */
#define PTP_CFG_GM_PRIORITY21 (0x00) /* (Port1) */

/* grandmasterClockQuality: Equal to parentDS.grandmasterClockQuality field */
/* b31 to b24: clockClass, default value(=248, = 0xF8), 255 is slave only clock */
/* b23 to b16: clockAccuracy, default value(=0x21)is within 100 nsec, 0x20 to 0x31 */
/* b15 to b0: offsetScaledLogVariance, default value(=0xFFFF) is not calculated yet */
#define PTP_CFG_GM_CLK_QUALITY0 (0xF821FFFF) /* (Port0) */
#define PTP_CFG_GM_CLK_QUALITY1 (0xF821FFFF) /* (Port1) */

/* grandmasterIdentity: Equal to parentDS.grandmasterIdentity field */
#define PTP_CFG_GM_CLK_ID0_U (0x00000000) /* Clock-ID hi (Port0) */
#define PTP_CFG_GM_CLK_ID0_L (0x00000000) /* Clock-ID lo (Port1) */
#define PTP_CFG_GM_CLK_ID1_U (0x00000000) /* Clock-ID hi (Port0) */
#define PTP_CFG_GM_CLK_ID1_L (0x00000000) /* Clock-ID lo (Port1) */

/* currentUtcOffset: Equal to timePropertiesDS.currentUtcOffset field */
#define PTP_CFG_CUR_UTC_OFFSET0 CURRENT_UTC_OFFSET /* (Port0) */
#define PTP_CFG_CUR_UTC_OFFSET1 CURRENT_UTC_OFFSET /* (Port1) */

/* timeSource: Equal to timePropertiesDS.timeSource field */
#define PTP_CFG_TIME_SOURCE0 (0xA0) /* Timesource is internal oscillator (Port0) */
#define PTP_CFG_TIME_SOURCE1 (0xA0) /* Timesource is internal oscillator (Port1) */
```

Figure 6-13 Example of *r_xxx_config.h (r_ptp_rx_config.h)*

6.7.2 FreeRTOS Kernel configuration

Configuration file of Renesas *FreeRTOS_Kernel* is generated as *FreeRTOS_Kernel.h* under the ¥src¥frtos_config.

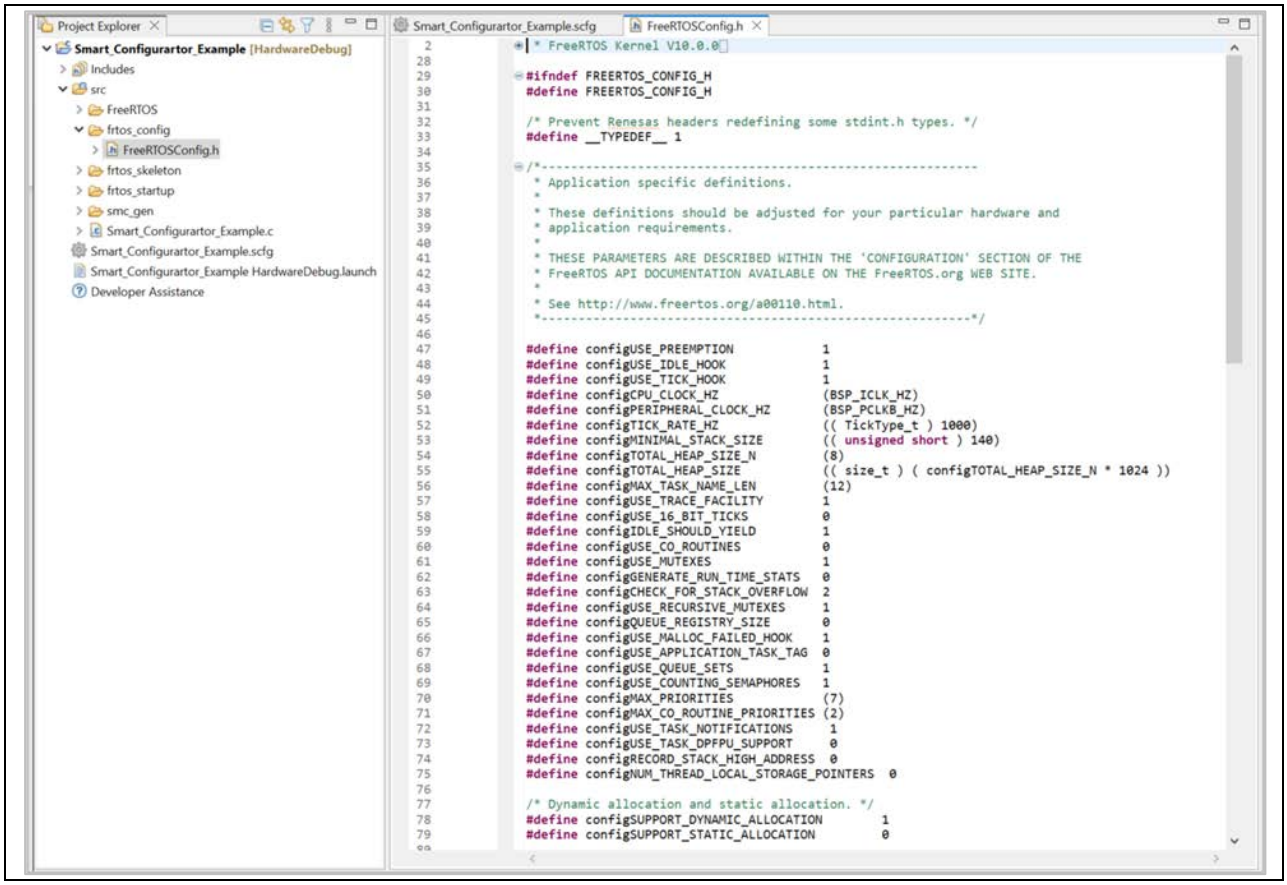


Figure 6-14 *FreeRTOSConfig.h*

7. Creating User Programs

The Smart Configurator handles two component types, [Firmware Integration Technology] and [Code Generator], with each requiring different methods to add custom code to the output source files. This section describes the methods to add custom code for both components.

7.1 Adding Custom Code in the Case of Firmware Integration Technology (FIT)

When [Firmware Integration Technology] is selected as the component type, the configuration options are set in r_xxx_config.h in the folder r_config. For the settings of the configuration options, refer to the application note (in the doc folder) on the FIT module (r_xxx) which you have added to the project tree.

If the target file already exists, the existing contents of the file are protected when source code is output.

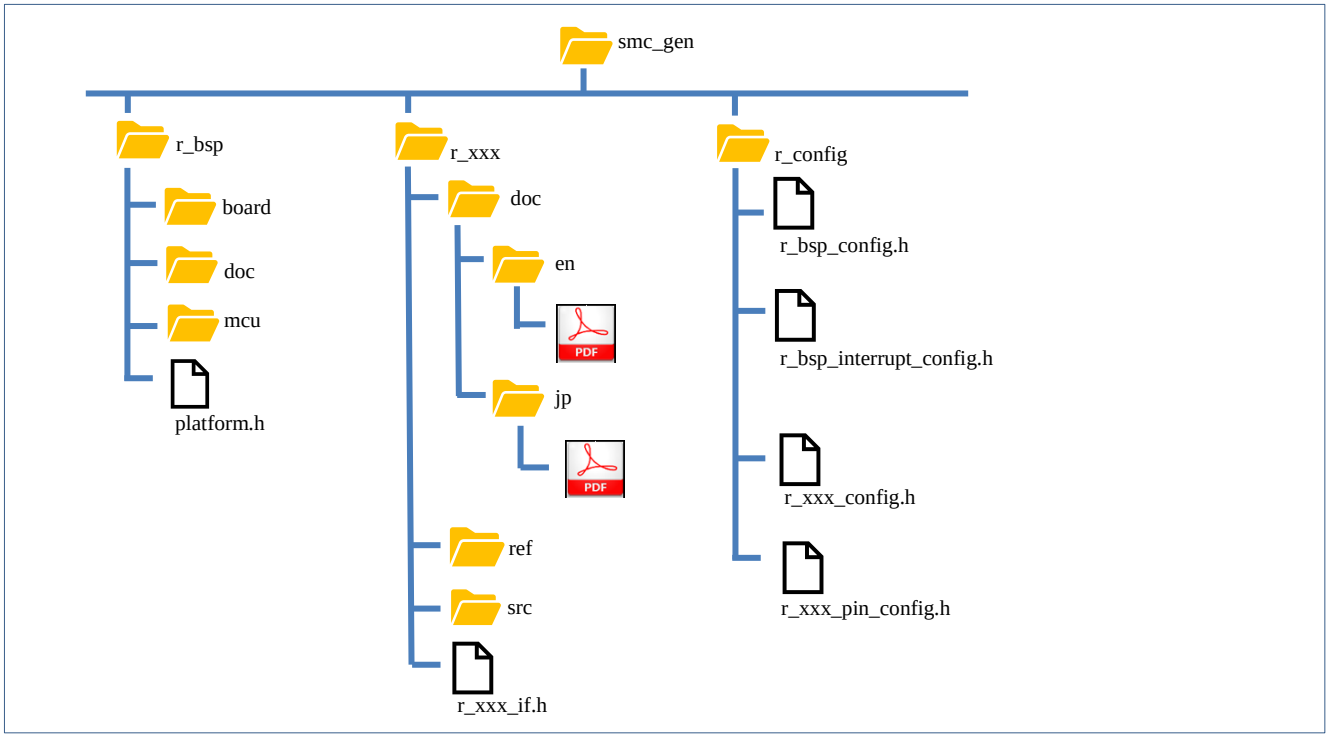


Figure 7-1 Tree Structure of Directories and Files for a FIT Module

7.2 Adding Custom Code in the Case of Code Generator

When [Code Generator] is selected as the component type, if files which have the same name already exist, new code will be merged only with the existing code that is between the comments below.

```
/* Start user code for xxxx. Do not edit comment generated here */

/* End user code. Do not edit comment generated here */
```

In the case of [Code Generator], three files are generated for each of the specified peripheral functions. The file names are “Config_xxx.h”, “Config_xxx.c”, and “Config_xxx_user.c” as the default, with “xxx” representing the name of the peripheral module. For example, “xxx” will be “CMT3” for the compare-match timer (resource CMT3). The comments to indicate where to add custom code are at the start and end of each of the three files. Comments to indicate where to add user code are also added to the interrupt function for the peripheral module corresponding to Config.xxx_user.c. The following examples are for CMT3 (Config_CMT3_user.c).

```
/***********************************************************************
Pragma directive
***********************************************************************/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_userdefine.h"
#include "Config_CMT3.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
Global variables and functions
***********************************************************************/
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
* Function Name: R_Config_CMT3_Create_UserInit
* Description  : This function adds user code after initializing the CMT3 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_CMT3_Create_UserInit(void)
{
    /* Start user code for user init. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}
```

```
/***********************************************************************
* Function Name: r_Config_CMT3_cmi3_interrupt
* Description  : This function is CMI3 interrupt service routine
* Arguments    : None
* Return Value : None
***********************************************************************/

#if FAST_INTERRUPT_VECTOR == VECT_PERIB_INTB129
#pragma interrupt r_Config_CMT3_cmi3_interrupt(vect=VECT(PERIB,INTB129),fint)
#else
#pragma interrupt r_Config_CMT3_cmi3_interrupt(vect=VECT(PERIB,INTB129))
#endif
static void r_Config_CMT3_cmi3_interrupt(void)
{
    /* Start user code for r_Config_CMT3_cmi3_interrupt. Do not edit comment
generated here */
    /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

7.3 Using Generated Code in user application

To use the generated code of FIT and Code Generator, follow the below steps:

1) Open the *{Project name}.c* file, add code to include the header files of the modules you want to use.

   In case of FIT, it is r_xxx_if.h.

   In case of Code Generator, it is added for you in “r_smc_entry.h” by automatically.

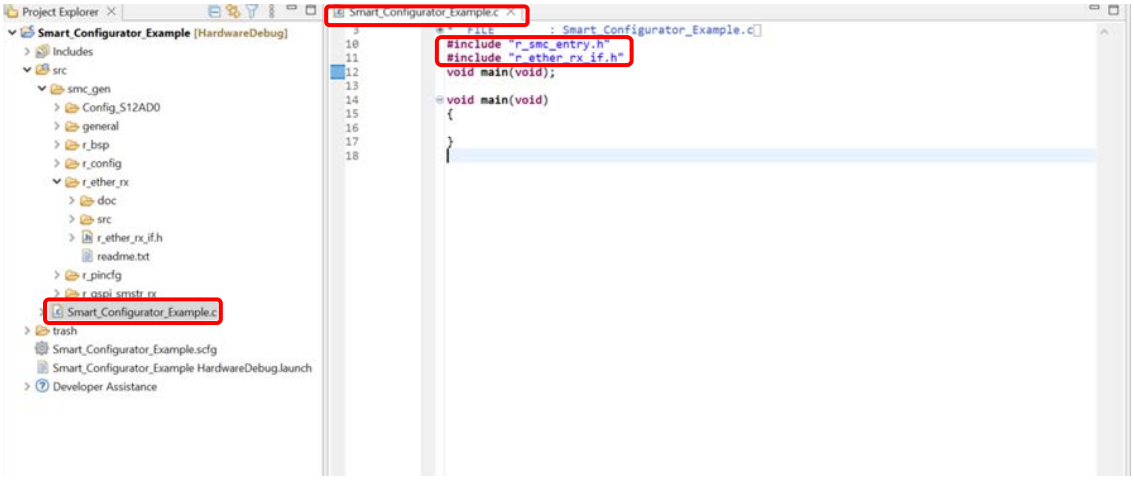


Figure 7-2 Add header files

2) In the main function, call the functions generated and add application codes.

   In case of Code Generator, driver initialization functions (R_ConfigName_Create) including initialization of pins have been called in *R_Systeminit* function of *r_cg_hardware_setup.c* by default. You just need to add application codes to perform operations that are driver-specific, for e.g. start (*R_ConfigName_Start*) and stop (*R_ConfigName_Stop*).

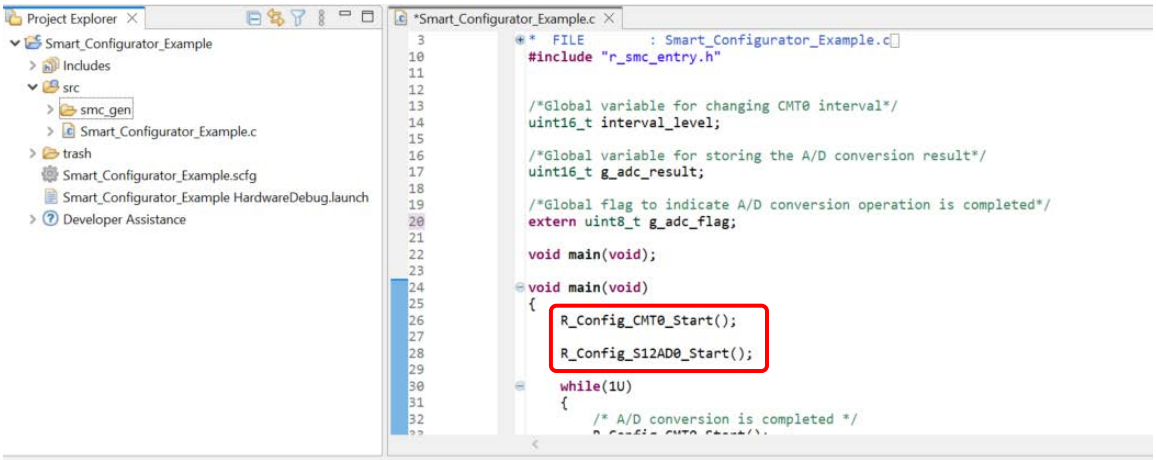


Figure 7-3 Call Code Generator functions

   In case of FIT module, refer to the examples provided in the “API Functions” chapter of corresponding Application Note. You can find the Application Note in [doc] folder under each FIT module.

For more reference, refer to “Smart Configurator Application Examples” in “chapter 12 Documents for Reference”.

8. Backing up Generated Source Code

The Smart Configurator has a function for backing up the source code.

The Smart Configurator generates a backup folder for the previously generated source code when new code is generated by clicking on the [ (Generate Code)] button. <Date-and-Time> indicates the date and time when the backup folder is created after code generation.

<ProjectDir>¥trash¥<Date-and-Time>

9. Generating Reports

The Smart Configurator generates a report on the configurations that you work on. Follow the procedure below to generate a report.

9.1 Report on All Configurations

A report is output in response to clicking on the [(Generate Report)] button in the Smart Configurator view. Two selections of output files are available (PDF, Text).

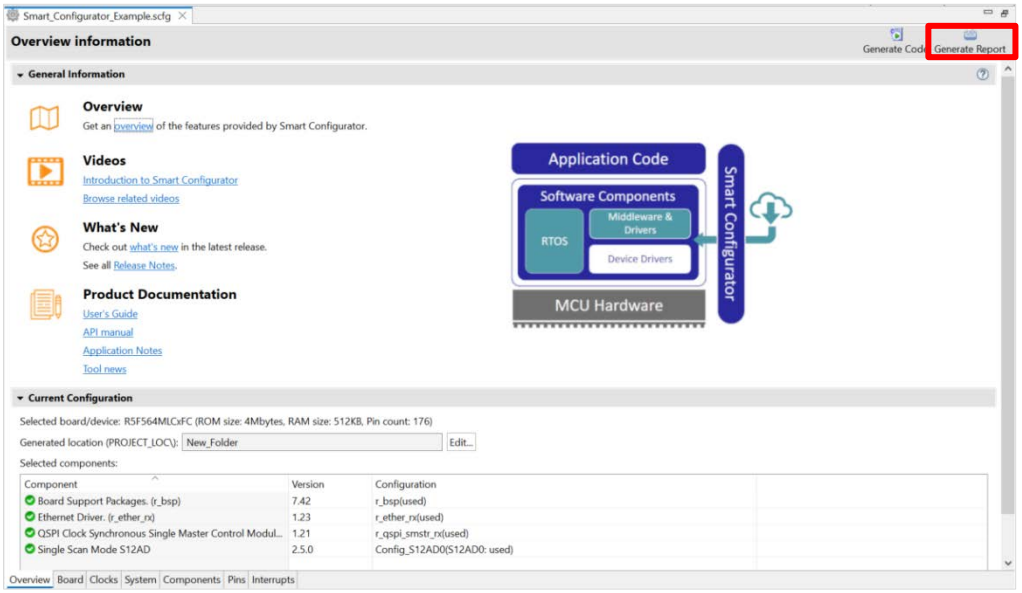


Figure 9-1 Output of a Report on the Configuration

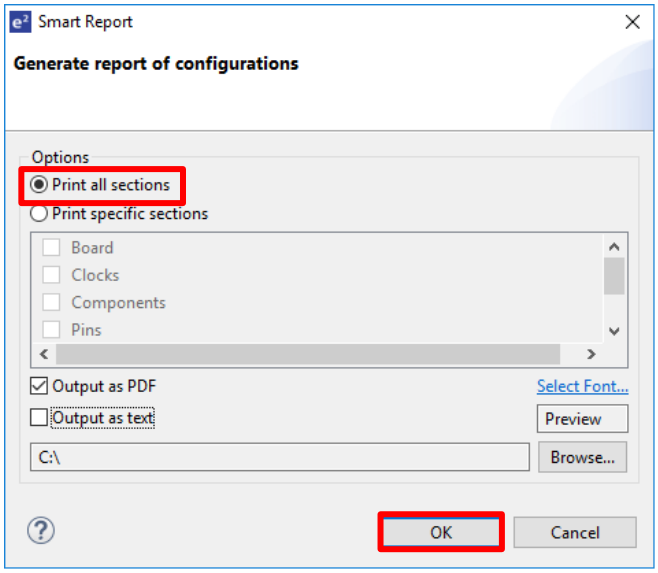


Figure 9-2 Dialog Box for Output of a Report (Example is selecting “Output as PDF”)

9.2 Configuration of Pin Function List and Pin Number List (in csv Format)

A list of the configuration of pin functions and pin numbers (whichever is selected at the time) is output in response to clicking on the [(Save the list to .csv file)] button on the [Pins] page of the Smart Configurator view.

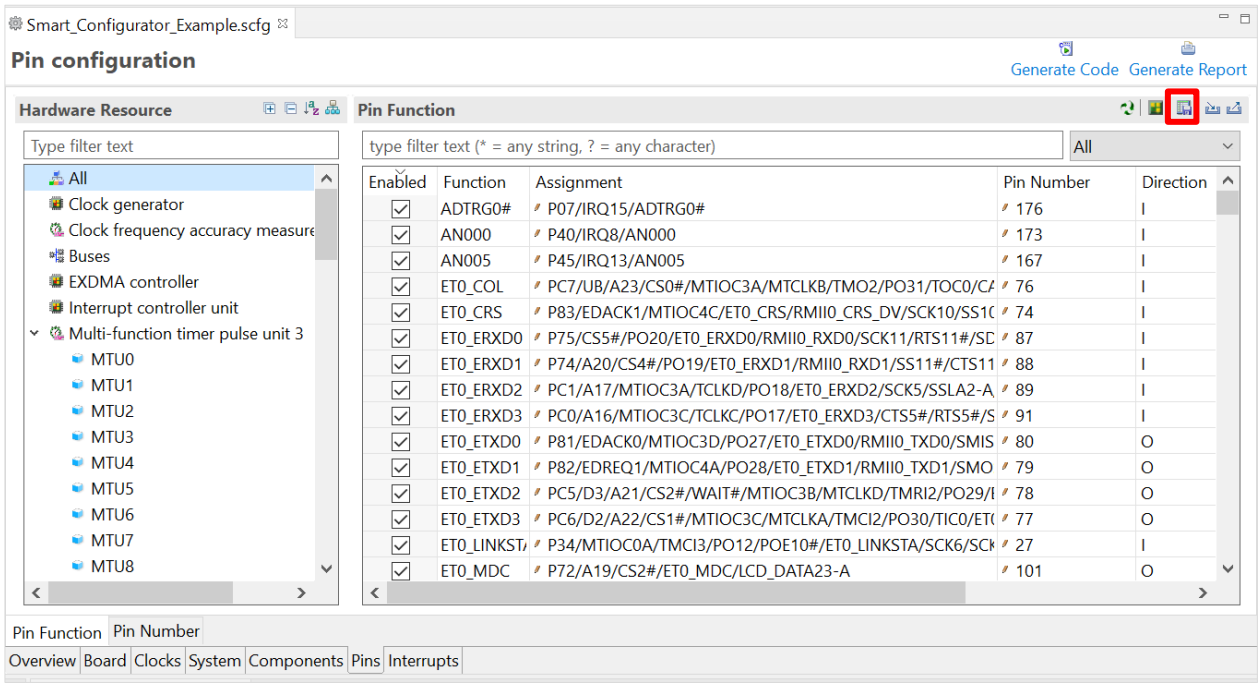


Figure 9-3 Output of a List of Pin Functions or Numbers (in csv Format)

9.3 Image of MCU/MPU Package (in png Format)

An image of the MCU/MPU package is output in response to clicking on the [(Save Package View to external image file)] button of the [MCU/MPU Package] view.

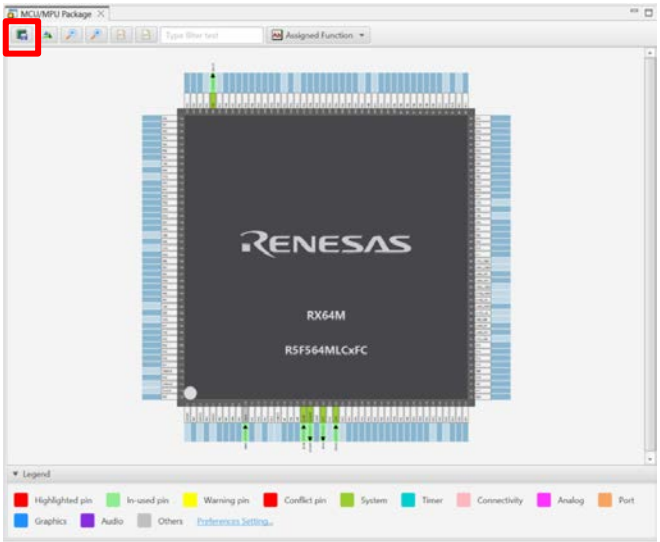


Figure 9-4 Outputting a Figure of MCU/MPU Package (in png Format)

10. User code protection feature for Smart Configurator Code Generation component

The Smart Configurator for RX Plug-in in e2 studio 2023-01 and later version now incorporates an enhanced user code protection feature. This feature empowers users to insert codes to any location in the generated codes by utilizing the specific tags, as shown in Figure 10-1. After the next code generation, the inserted user codes will be protected and automatically merged into the generated files.

The user code protection feature will only be supported on the files that are generated by the “Code Generation component”.

10.1 Specific tags for the user code protection feature

When using the user code protection feature, please insert /* Start user code */ and /* End user code */ as shown in Figure 10-1 and add the user codes between these tags. If the specific tags do not match exactly, the inserted user code will not be protected after the code generation.

```
/* Start user code */

User code can be added between the specific tags

/* End user code */
```

Figure 10-1 Specific tags for user code protection feature

10.2 Examples of using user code protection feature to add new user code

Figure 10-2 shows an example of adding new user code into the Create API of CMT module by using the specific tags shown in Figure 10-1. After updating the configuration in the CMT GUI and re-generating the codes, the inserted user codes will be automatically merged into the new generated file.

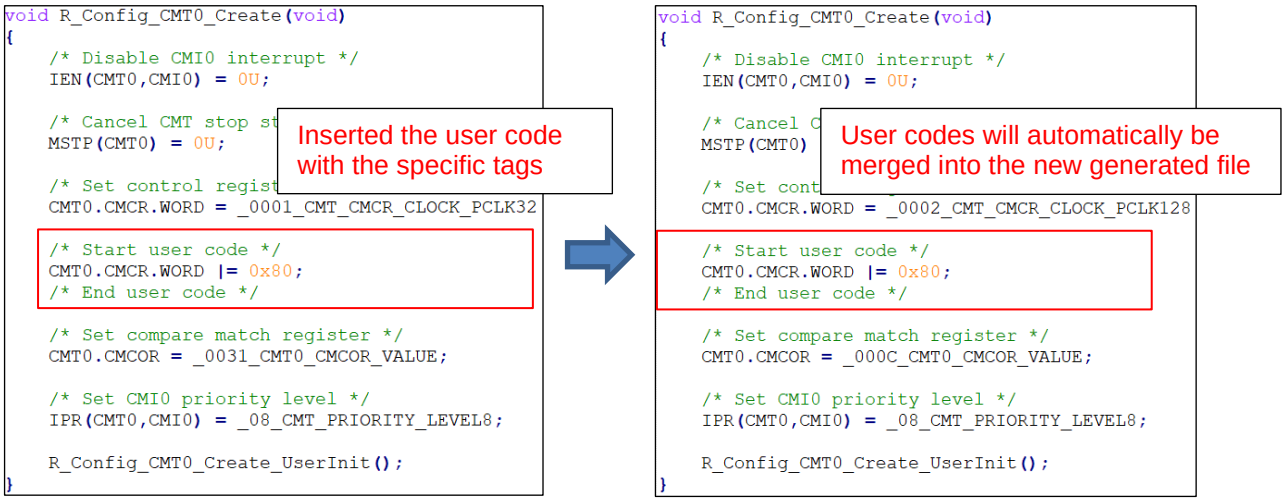


Figure 10-2 User code protection with auto merge

10.3 What to do when merge conflict occurs

10.3.1 What is Merge conflict

When the lines of generated codes before and after the inserted user codes are updated due to changes in GUI configuration or the version update of Smart Configurator, merge conflict codes will be generated out, as shown in Figure 10-3.

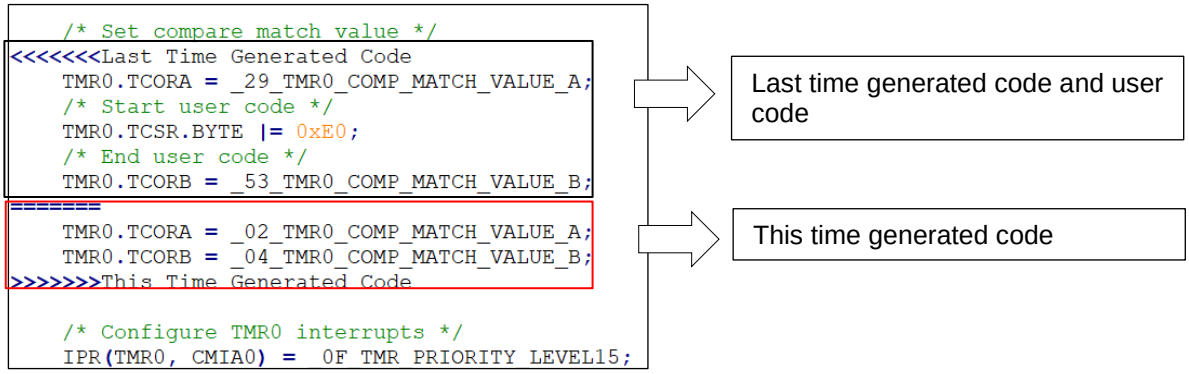


Figure 10-3 User code protection with merge conflict

If the merge conflict occurs, conflict message will be displayed in the Smart Configurator console, as shown in Figure 10-4.

Figure 10-4 The merge conflict message outputted in the Smart Configurator console

10.3.2 Steps for resolving the merge conflict

To resolve this merge conflict, open the highlighted conflict files and follow the steps below to solve the merge conflicts manually.

1) Copy the user code from “Last Time Generated Code” and paste it into the new position in “This Time Generated Code” as shown in Figure 10-5.

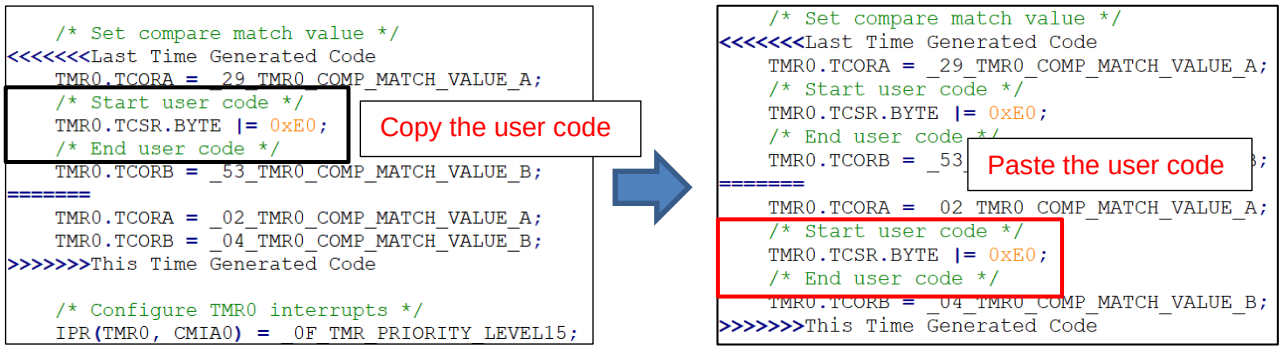


Figure 10-5 Generated conflict code

2) Remove last time generated code and the conflicts commend (<<<<<<<Last Time Generated Code, ======= and >>>>>>>This Time Generated Code) as shown in Figure 10-6**.**

```
/* Set compare match value */
TMR0.TCORA = _02_TMR0_COMP_MATCH_VALUE_A;
/* Start user code */
TMR0.TCSR.BYTE |= 0xE0;
/* End user code */
TMR0.TCORB = _04_TMR0_COMP_MATCH_VALUE_B;
```

Figure 10-6 The codes after resolving the merge conflict

Another way to solve merge conflict:
1). Click this console message to open the compare view

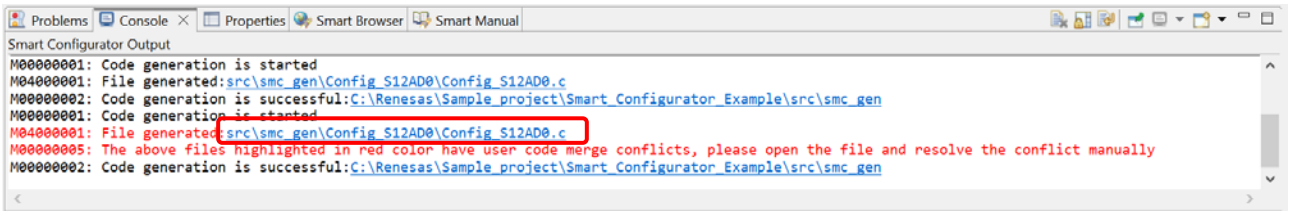


Figure 10-7 Error message in console

2). After compare view is opened, user can apply left change to the right. Or user can edit right side code manually.

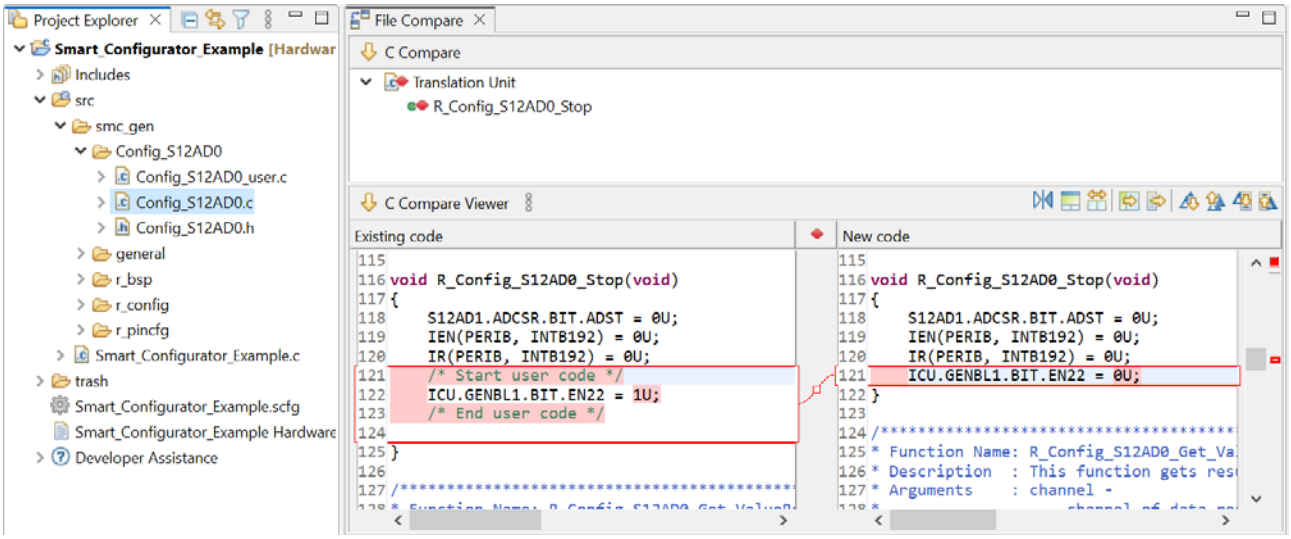


Figure 10-8 Compare Viewer for conflict code

11. Help

11.1 Help

Refer to the help system from the e² studio menu for detailed information on the Smart Configurator.

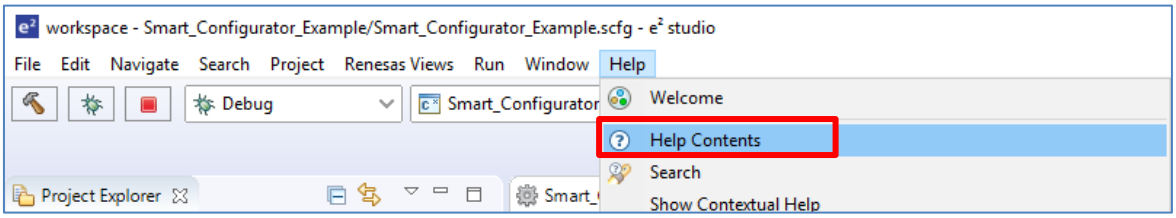


Figure 11-1 Help Menu

The help system can also be activated from the [Overview information] page by clicking  button.

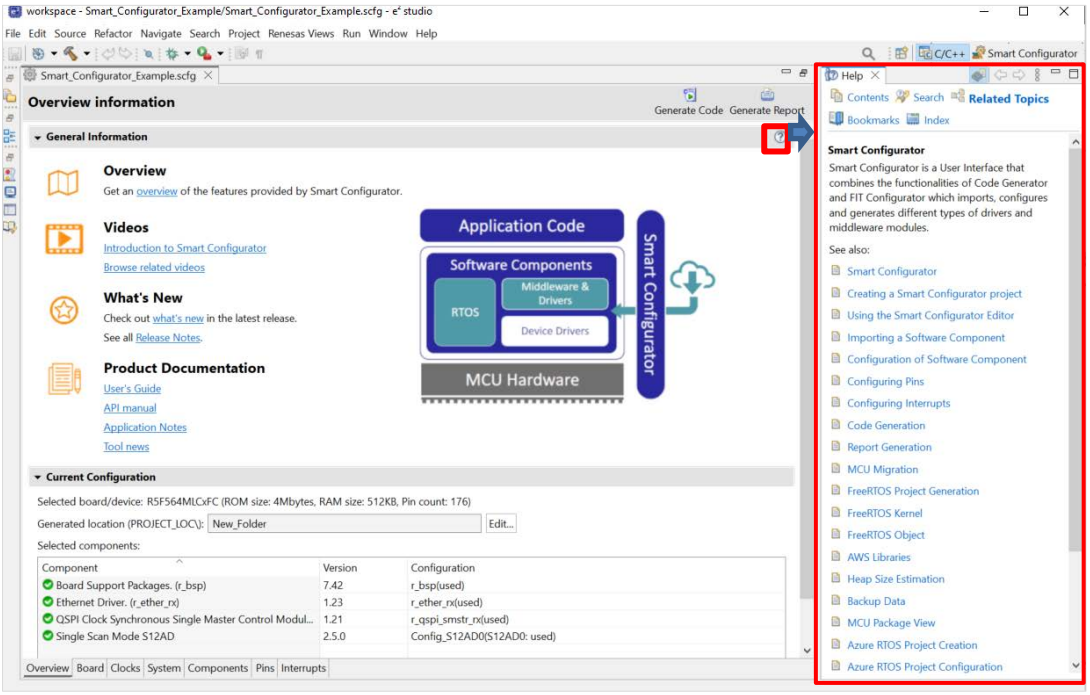


Figure 11-2 Smart Configurator Quick Start information

11.2 Developer assistance

Developer assistance provides user a tool to display API information and calling examples of components they added so user can easily develop without having to know the auto-generated code.

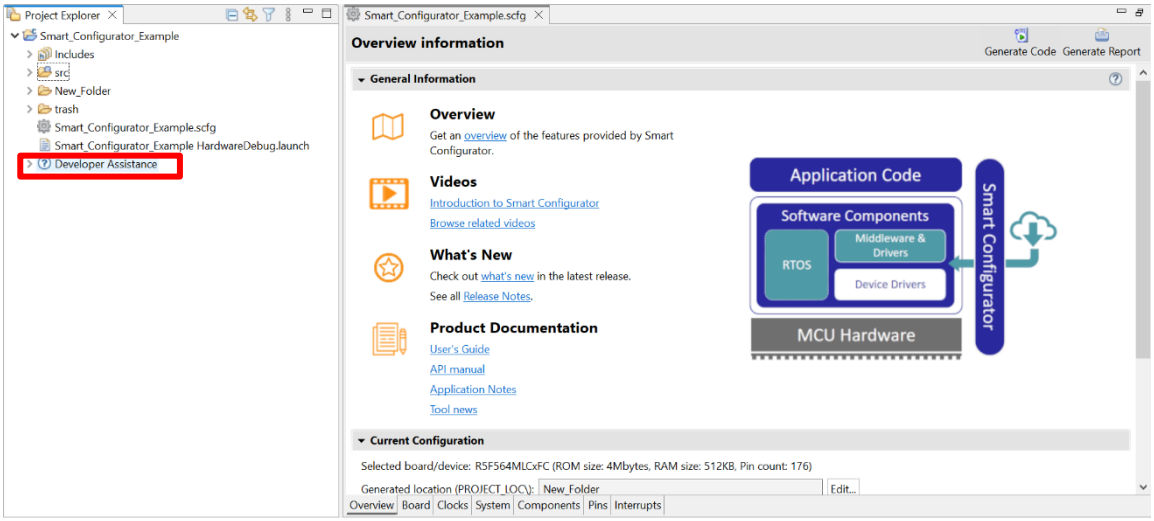


Figure 11-3 Developer assistance

(1) Add component and generate code.

(2) Check generated file exist.

(3) Expand [Developer assistance] in project explore and double click the component.

(4) Read API information and examples.

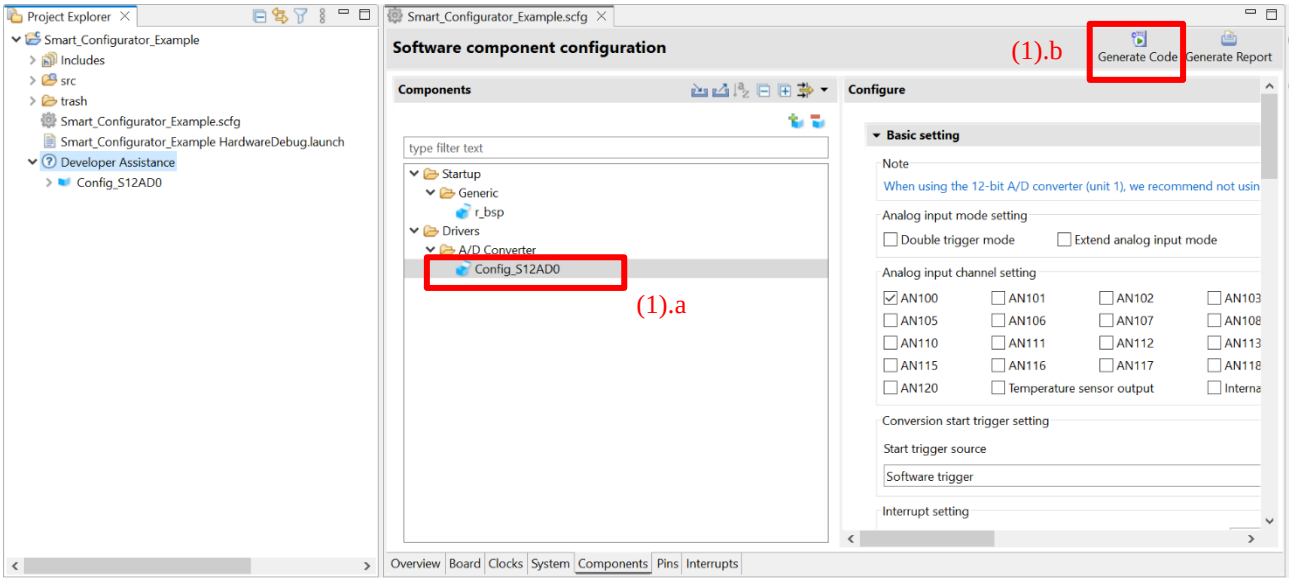


Figure 11-4 Adding component and generating code

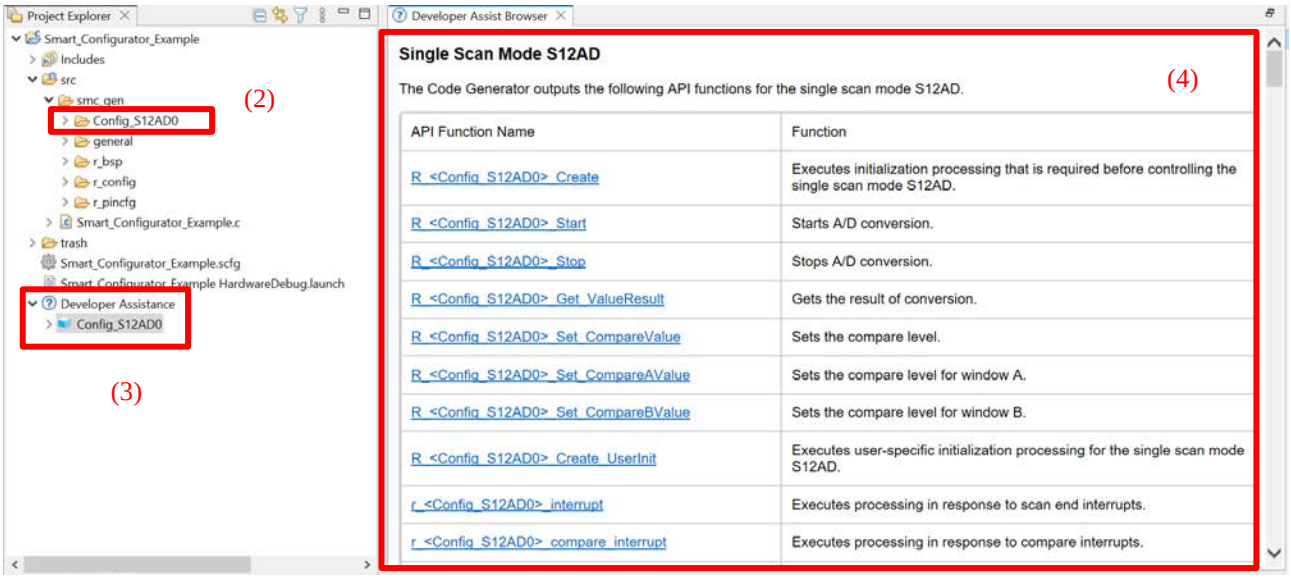


Figure 11-5 Developer assistance in Project Explorer

12. Documents for Reference

User’s Manual: Hardware

Obtain the latest version of the manual from the Renesas Electronics website.

Technical Update/Technical News

Obtain the latest information from the Renesas Electronics website.

User’s Manual: Development Environment

e2 studio Integrated Development Environment User’s Manual: Getting Started Guide (R20UT4374)

CC-RX Compiler User’s Manual (R20UT3248)

RX family Renesas FreeRTOS Application Note (R01AN4307)

e2 studio Partner RTOS Aware Debugging for RX (R20AN0586)

(Obtain the latest version from the Renesas Electronics website.)

Smart Configurator Application Examples

Smart Configurator Application Examples: Ethernet (R20AN0495)

Smart Configurator Application Examples: CMT, A/D, SCI, DMA, USB (R20AN0469)

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul 31, 2017	-	First edition issued
1.10	Mar 20, 2018	-	Update to e² studio v6.2
		4	1.2 Operating Environment updated
		8	Figure 2-4 Select toolchain, device and debug setting updated
		15	Figure 3-6 Add component updated
		16	Figure 3-7 Add Code Generator component updated
		19	Figure 3-12 Add FIT modules updated
		26	Figure 3-22 Configure pins for r_ether_rx updated
		30	Note added
		36	4.1 Generated files structure Description of the r_bsp folder in the table corrected
		50	8.1 Troubleshooting added
		53	9 Application Example updated
1.20	Nov 1, 2018	-	Update to e² studio v7.0
		3	1. Overview updated
		9	Figure 3-1 Procedure for Operations added
		10	3.3 File to be Saved as Project Information added
		11	3.4 Window added
		19	4.1.3 Exporting board setting, 4.1.4 Importing board setting added
		21	Figure 4-7 [Components] Page added
		33	Figure 4-26 [Pins] Page ([Pin Function]), Figure4-26 [Pins] Page ([Pin Number]) added
		39	Figure4-35 [Interrupts] Page
		42	Figure 5-1 Generating a Source File, Figure5-2 Source Files in the Project Explorer added
		55	7. Creating User Programs added
		60	9.2 Configuration of Pin Function List and Pin Number List (in csv Format), 9.3 Image of MCU Package (in png Format) added
		61	10.1 Help added
1.30	Jan 1, 2019	-	Update to e² studio v7.3
		9	2.2 Create a FreeRTOS project added
		21	3.4.3 MCU Package view updated
		23	4.1.1 Selecting the device updated
		31	4.3.4 Download and import FIT sample project added
		40	4.3.9 Setting a FIT Software Component update
		41	4.3.12 Version change of FIT software component added
		43	4.3.11 Setting the FreeRTOS Kernel added
		44	4.3.12 Configure general setting of component added
		45	Figure 4-41 [Pins] Page ([Pin Number]) updated
		49	4.4.5 Pin setting using board pin configuration information added 4.4.6 Pin filter feature added
		53	4.6 MCU migration feature added
		70	6.6.2 FreeRTOS Kernel configuration added
1.40	Jun 21, 2021	-	Update to e² studio v2021-04
		5	2. Creating a project updated
		19	3.4.2 Smart Configurator view updated
		22	4.1 Board Settings updated

		25	4.3 System Settings added
		34	4.4.4 Downloading a FIT module updated
		35	4.4.6 Adding FIT drivers or middleware updated
		39	4.4.9 Setting a FIT Software Component updated
		40	4.4.10 Version change of FIT software component updated
		42	4.4.11 Solving the grey-out component added
		43	4.4.13 Setting the RTOS Kernel added
		44	4.4.14 Creating the RTOS Object added
		45	4.4.15 Setting the RTOS Library added
		46	4.4.16 Configure Analog Front End component added
		48	4.4.17 Configure Motor Component added
		57	4.4.18 Configure general setting of component
		58	4.5.8 Pin Errors/Warnings setting added
		65	5 Managing Conflicts updated
		68	6.2 Change Generated Code Location added
		84	7.3 Using Generated Code in user application added
		89	11 Documents for Reference updated
1.50	Apr 16, 2023	1	Updated the URL for RX Smart Configurator
		95 - 97	Added new chapter 10 User code protection feature for Smart Configurator Code Generation Component
1.60	Apr 16, 2024	-	Figures are updated to e² studio 2024-01
		16	2.3 Create a Blinky project added
		26	3.4.3 MCU Package view updated
		34	4.4.2 Removing a software component updated
		55	4.4.16 Configure Motor Component updated
		59	4.4.17 Configure general setting of component updated
		68	4.5.9 Symbolic name setting added
		73	4.6.3 Multiple interrupts setting added
		106	10.3.2 Steps for resolving the merge conflict updated
		109	11.2 Developer assistance added
1.70	Jan 20, 2026	37	4.4.4 Added note for Information icon of CG driver
		60 - 63	4.4.17 Updated notes for general setting of component

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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