RH850 Smart Configurator

User's Guide: CS+

R20AN0516EJ0130
Rev.1.30
Apr 22, 2024

Introduction

This application note describes the basic usage of the RH850 Smart Configurator (hereafter called the Smart Configurator), and the procedure for adding its output files to CS+ projects.

References to the Smart Configurator and CS+ integrated development environment in this application note apply to the following versions.

- CS+ (CS+ for CC) V8.11.00 and later
- RH850 Smart Configurator V1.11.0 and later
- CS+ RH850 Smart Configurator Communication Plugins V1.11.00 and later

Target Devices and Compilers

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

https://www.renesas.com/rh850-smart-configurator

Contents

1. Overview 4
1.1 Purpose 4
1.2 Features 4

2. Before Using the Smart Configurator 5
2.1 Preparing the CS+ (CS+ for CC) Integrated Development Environment 5
2.2 Installing the Smart Configurator 5
2.3 Setting the CS+ Integrated Development Environment 5
2.3.1 Checking the plug-in settings 5
2.3.2 Checking the setting of the execution path 6
2.4 Uninstalling the Smart Configurator 6
2.5 Preparing Sample Projects 7

3. Operating the Smart Configurator 8
3.1 Procedure for Operations 8
3.2 Starting the Smart Configurator 9
3.3 File to be Saved as Project Information 9
3.4 Window 10
3.4.1 Main menu 11
3.4.2 Toolbar 11
3.4.3 Smart Configurator view 12
3.4.4 MCU/MPU Package view 13
3.4.5 Console view 14
3.4.6 Configuration Problems view 14

4. Setting of Peripheral Modules 15
4.1 Board Settings 15
4.1.1 Selecting the device 15
4.1.2 Selecting the board 15
4.1.3 Exporting board settings 17
4.1.4 Importing board settings 18
4.2 Clock Settings 19
4.3 System Settings (only for RH850/U2A) 20
4.4 Component Settings 21
4.4.1 Adding Code Generator components 21
4.4.2 Switching between the component view and hardware view 23
4.4.3 Removing software component 24
4.4.4 Setting a Code Generator Component 25
4.4.5 Changing the resource for a Code Generate Configuration 26
4.4.6 Configure general setting of the component 29
4.5 Pin Settings 30
4.5.1 Changing the pin assignment of a software component 31
4.5.2 Assigning pins using the MCU/MPU Package view 32
4.5.3 Show pin number from pin functions 33
4.5.4 Exporting pin settings 34

4.5.5 Importing pin settings 35
4.5.6 Pin setting using board pin configuration information 36
4.5.7 Pin filter feature 37
4.5.8 Pin Errors/Warnings setting 38
4.6 Interrupt Settings 39
4.6.1 Changing the interrupt priority level and OS management setting 40
4.6.2 Changing the PE*n* setting(RH850/U2A only) 41
4.6.3 Changing the interrupt handler name, Generate Entity and Generate Enable/Disable Function setting 43

5. Managing Conflicts 45
5.1 Resource Conflicts 45
5.2 Resolving pin conflicts 46

6. Generating Source Code 47
6.1 Registering Generated Source Code with CS+ 47
6.2 Loading files generated by All Toolchain (CC-RH, GHS, IAR) 48
6.3 Configuration of Generated Files and File Names 50
6.4 Initializing Clocks 53
6.5 Initializing Pins 54
6.6 Initializing Interrupts 56

7. Creating User Programs 57
7.1 Adding Custom Code in the Case of Code Generator 57

8. Backing up Generated Source Code 59

9. Generating Reports 60
9.1 Report on All Configurations 60
9.2 Configuration of Pin Function List and Pin Number List (in csv Format) 62

10. User code protection feature 62
10.1 Specific tags for the user code protection feature 62
10.2 Image of MCU/MPU Package (in png Format) 63
10.3 Examples of using user code protection feature to add new user code 64
10.4 What to do when merge conflict occurs 65
10.4.1 What is Merge conflict 65
10.4.2 Steps for resolving the merge conflict 66

11. Help 68
11.1 Help 68

12. Documents for Reference 69

Website and Support 70

1. Overview

1.1 Purpose

This application note describes the basic usage of the Smart Configurator and CS+ integrated development environment, including the procedure for creating a project and adding Smart Configurator output to CS+ projects.

Refer to the User’s Manual of CS+ for how to use CS+.

1.2 Features

The Smart Configurator is a utility for combining software to meet your needs. It handles the following two functions to support the embedding of drivers from Renesas in your systems: generating driver code and making pin settings.

2. Before Using the Smart Configurator

2.1 Preparing the CS+ (CS+ for CC) Integrated Development Environment

To create or build a program in the CS+ integrated development environment with the use of source code generated by the Smart Configurator, you will need to install CS+ to handle building for the target device.

2.2 Installing the Smart Configurator

Download the RH850 Smart Configurator and CS+ RH850 Smart Configurator communication plug-in from the URL below. The CS+ Smart Configurator communication plug-in is required for registering source code generated by the Smart Configurator with CS+.

https://www.renesas.com/rh850-smart-configurator

After activating the installer, install the Smart Configurator and the plug-in by following the procedure of the installer. You will require administrator privileges to do this.

Note: Source code generated with All Toolchain (CC-RH, GHS, IAR) by Smart Configurator can also be added and built in the CS+ Integrated Development Environment, but no need to install CS+ RH850 Smart Configurator communication plug-in and set the CS+ Integrated Development Environment. Please refer to 6.2 Loading files generated by All Toolchain (CC-RH, GHS, IAR) for detailed procedure.

2.3 Setting the CS+ Integrated Development Environment

Source files the Smart Configurator generates can be registered with CS+, and CS+ can be set to the configuration required to build the registered source files. This is set up automatically at the time the Smart Configurator is installed; however, you will need to check the settings against the following and modify them as required.

2.3.1 Checking the plug-in settings

Select [Plug-in Manager] from [Tool] of CS+ menu and confirm that there is a tick against “Smart Configurator for RH850 Communication Plug-in”. Tick it if it is not.

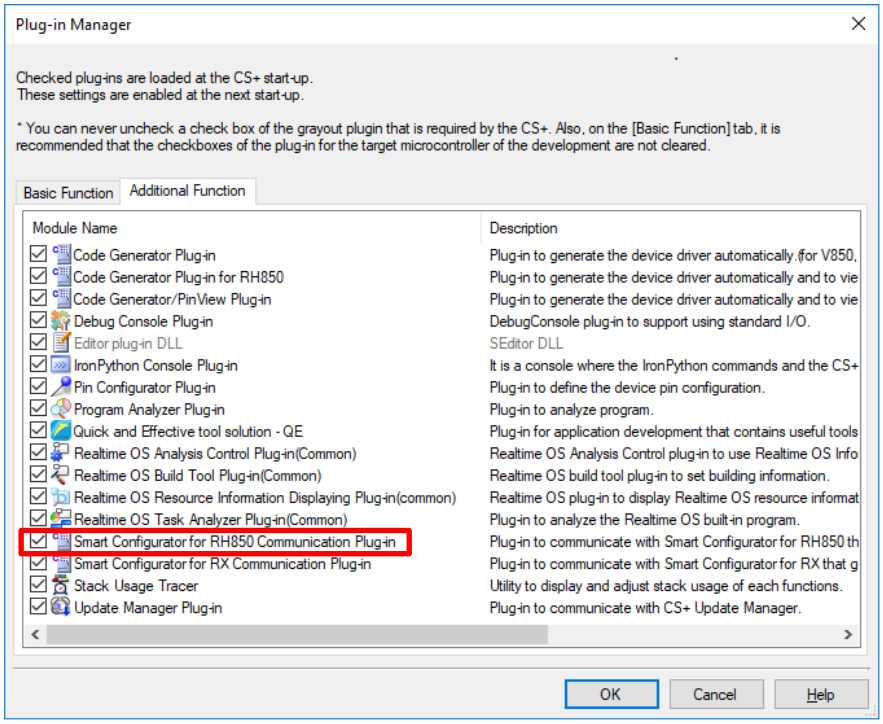

Figure 2-1 Plug-in Manager

2.3.2 Checking the setting of the execution path

[Smart Configurator (Design Tool)] is displayed under [Project name (Project)] in the Project Tree when you open the CS+ project for the target device of the Smart Configurator.

Click on [Smart Configurator (Design Tool)], and the Smart Configurator Property panel is displayed.

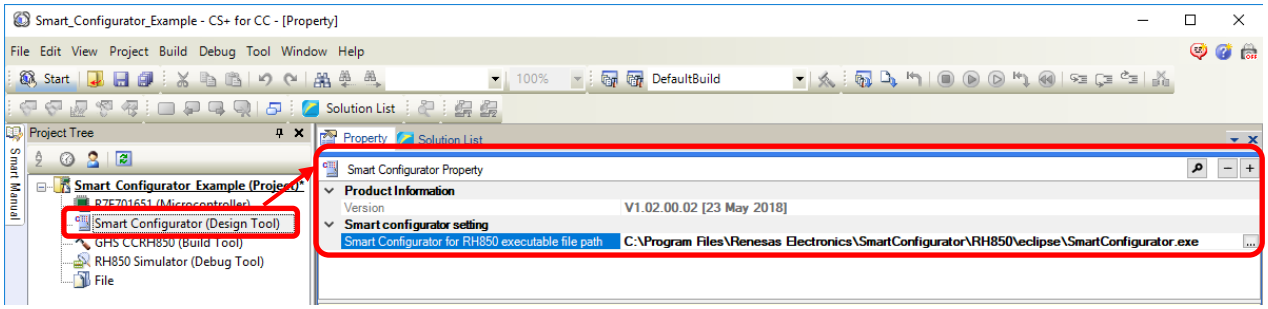

Figure 2-2 Displaying the Property

“Smart Configurator for RH850 executable file path” shows the executable file of the Smart Configurator. The following path is set when the Smart Configurator is installed with the default setting (where “CS+” and “Smart Configurator” are in the same level).

"C:\Program Files (x86)\Renesas Electronics\SmartConfigurator\RH850\eclipse\SmartConfigurator.exe"

When manually specifying the path of the executable file, “Smart Configurator for RH850 executable file path” can be set as either a relative or an absolute path.

2.4 Uninstalling the Smart Configurator

If you wish to uninstall the Smart Configurator, select “Smart Configurator for RH850” and “CS+ SC Communication Plugins for RH850” from [Apps and Features] in the control panel and uninstall them.

2.5 Preparing Sample Projects

The Smart Configurator outputs source files for the main function and for the initialization of peripheral modules that were set up by using Smart Configurator components. However, the Smart Configurator does not output source files for the initialization that is performed between a reset of the microcontroller and the start of the main function or for the startup routine, which initiates the main function and executes other necessary processing.

Therefore, we provide sample projects that include sample startup routines and other necessary processing so that user applications can be built immediately after peripheral modules are set up using the Smart Configurator.

Refer to either of the documents stored in the following locations and create a CS+ project from the sample project.

“C:\Program Files (x86)\Renesas Electronics\SmartConfigurator\RH850\RH850C1M_SampleProjects”

“C:\Program Files (x86)\Renesas Electronics\SmartConfigurator\RH850\RH850F1KH_SampleProjects”

“C:\Program Files (x86)\Renesas Electronics\SmartConfigurator\RH850\RH850F1KM_SampleProjects”

“C:\Program Files (x86)\Renesas Electronics\SmartConfigurator\RH850\RH850U2A_SampleProjects”

“C:\Program Files (x86)\Renesas Electronics\SmartConfigurator\RH850\RH850U2B_SampleProjects”

3. Operating the Smart Configurator

3.1 Procedure for Operations

Figure 3-1 shows the procedure for using the Smart Configurator to generate files for setting up peripheral modules, and to use them in building after registration with CS+. Refer to the related documents on CS+ for the operation of CS+.

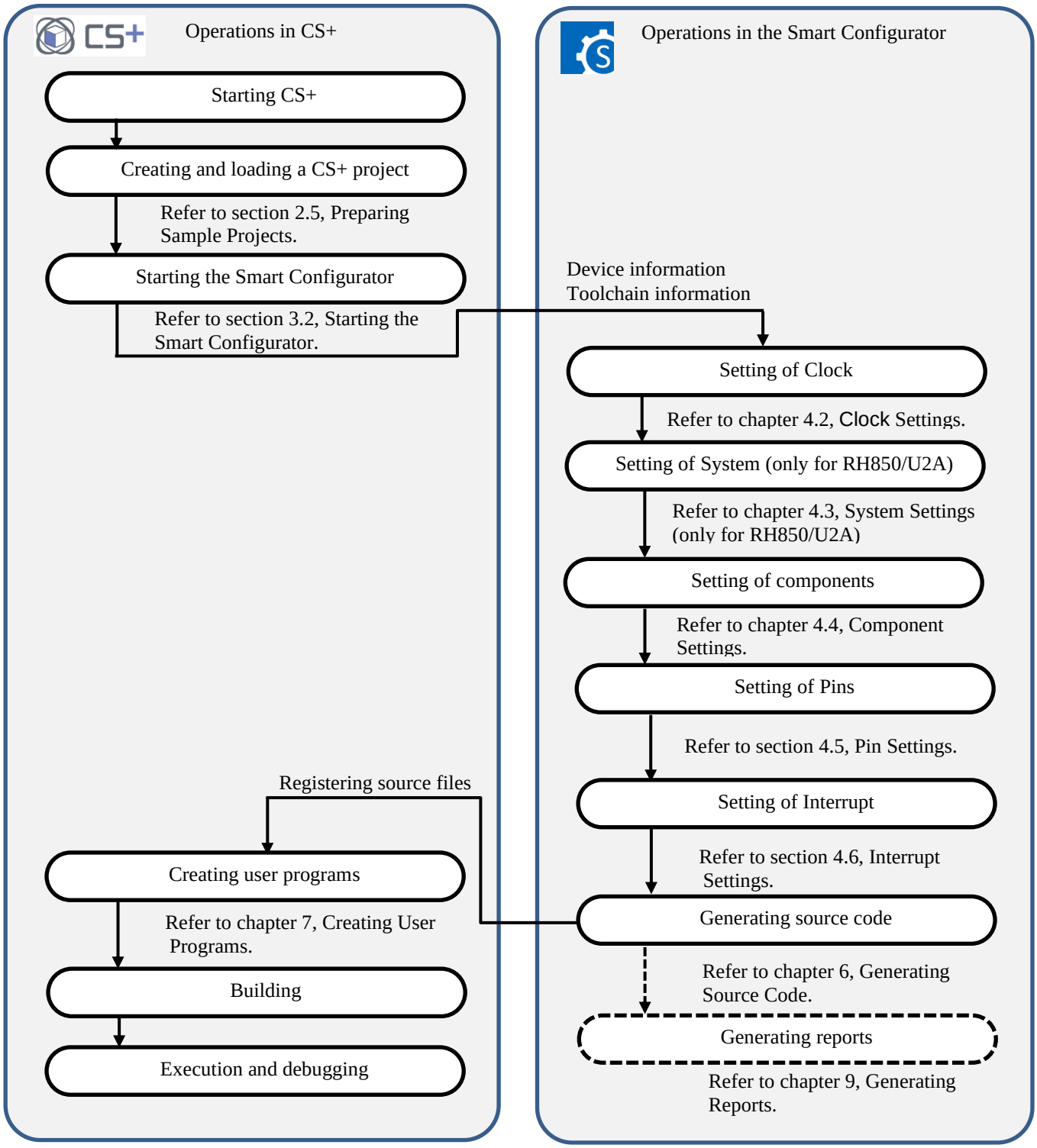

Figure 3-1 Procedure for Operations

Note: Sample project is provided by Smart Configurator for RH850 for easier usage, you can refer to chapter 2.5, Preparing Sample Projects for more information.

3.2 Starting the Smart Configurator

Double-click on [Smart Configurator (Design Tool)] under [Project name (Project)] in the Project Tree of CS+ to start the Smart Configurator. You do not need to select a device or toolchain for the Smart Configurator, since the settings of the project for CS+ are passed over to the Smart Configurator.

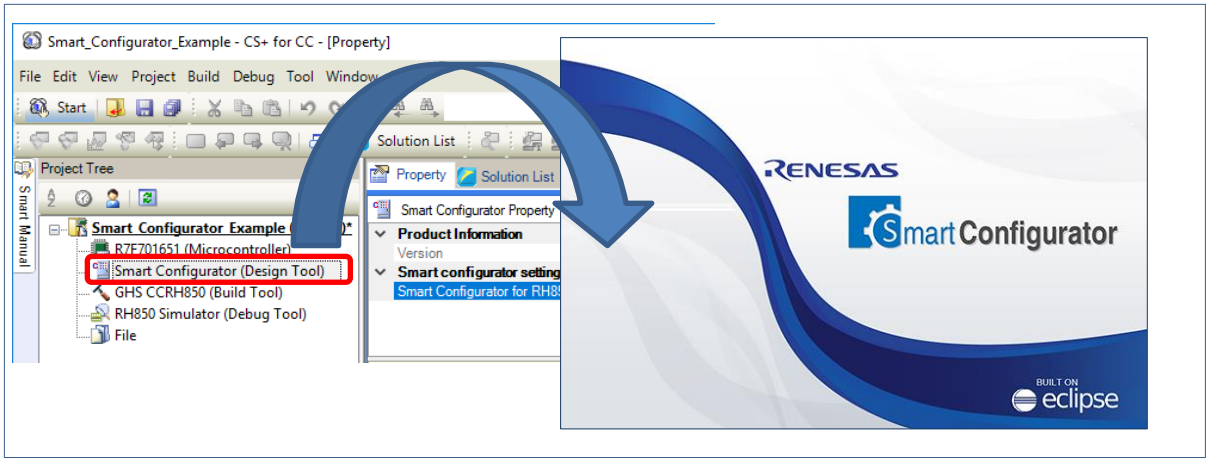

Figure 3-2 Activation of Smart Configurator

Note: The settings of CS+ are not passed over to the Smart Configurator in the following cases: when the Smart Configurator is activated from its executable file, when a new project is created from [File] menu of the Smart Configurator, or when an existing file from the Smart Configurator is opened.

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.

When the Smart Configurator is activated from CS+, the project file from the Smart Configurator is saved in “project name.scfg”, which is at the same level as the project file (*.mtpj) of CS+.

3.4 Window

The main window is displayed when the Smart Configurator is started. The configuration of the window is shown in Figure 3-3, Main Window.

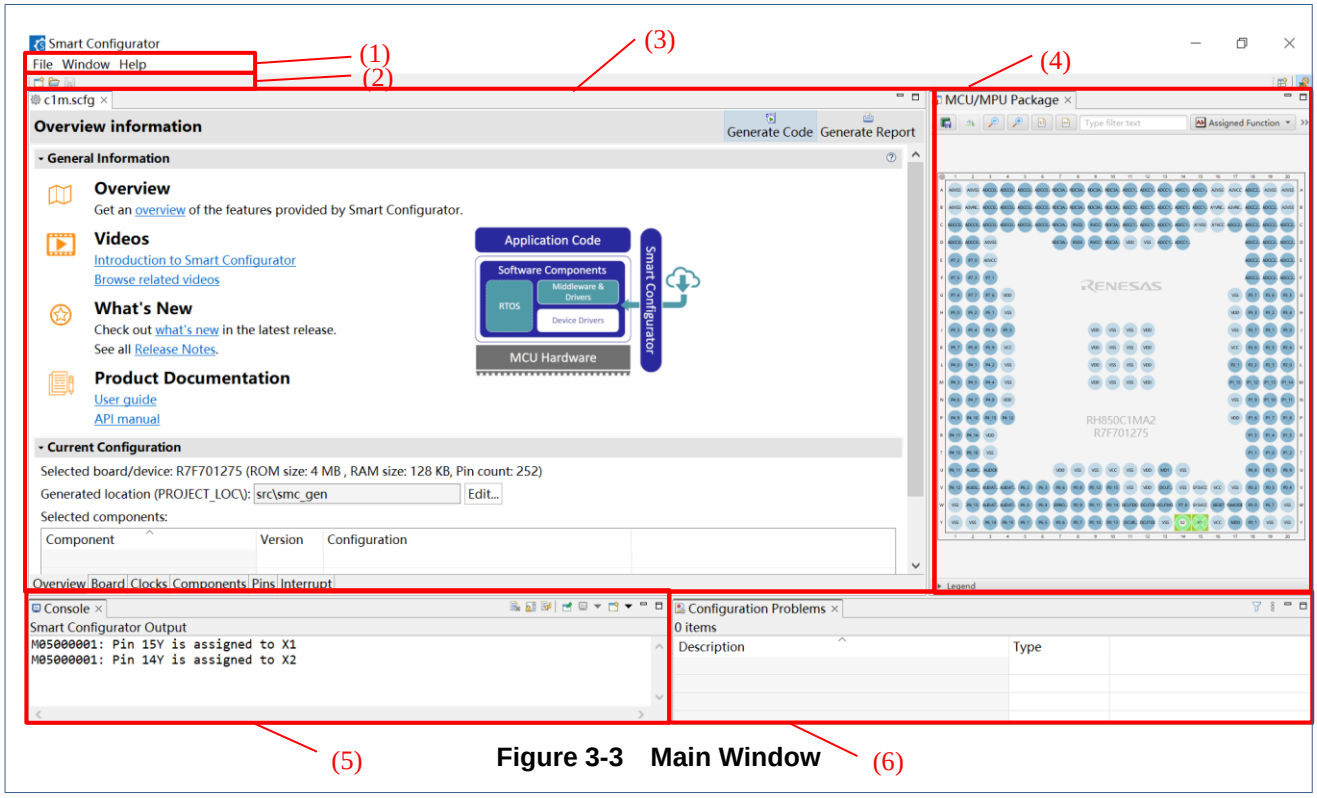

Figure 3-3 Main Window

1) Menu bar
2) Main toolbar
3) Smart Configurator view
4) MCU Package view
5) Console view
6) Configuration Problems view

3.4.1 Main menu

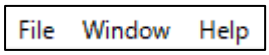

Table 3-1, Main Menu Items, lists the items of the main menu.

Table 3-1 Main Menu Items

Menu		Details
File	New	The dialog box [New Smart Configuration File], which is used to create a new project, is displayed.
	Open	The dialog box [Open], which opens an existing project, is displayed.
	Save	Saves a project with the same name.
	Restart	Smart Configurator is restarted. Do not use this menu item in general, as it leads to deletion of the project settings handed over from CS+.
	Exit	Execution of the Smart Configurator is terminated.
Window	Preference	The dialog box [Preference], which is used to specify the properties of the project, is displayed.
	Show view	The dialog box [Show view], which is used to set the view of the window, is displayed.
Help	Help Contents	The help menu is displayed.
	Home Page	Open the home page of Smart Configurator in Renesas website
	Release Notes	Search for release notes of Smart Configurator in Renesas website
	Tool News	Search for tool news of Smart Configurator in Renesas website
	API Manual	Search for the RH850 API Reference (R20UT4361) in Renesas website
	About	The version information is displayed.

3.4.2 Toolbar

Some functions of the main menu are allocated to the buttons on the toolbar. Table 3-2, Toolbar Buttons and Related Menu Items, shows the description of those tool buttons.

Table 3-2 Toolbar Buttons and Related Menu Items

Toolbar button	Related menu item
	[File] ® [New]
	[File] ® [Open]
	[File] ® [Save]

3.4.3 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.
Note: [System] page is supported only for RH850/U2A.

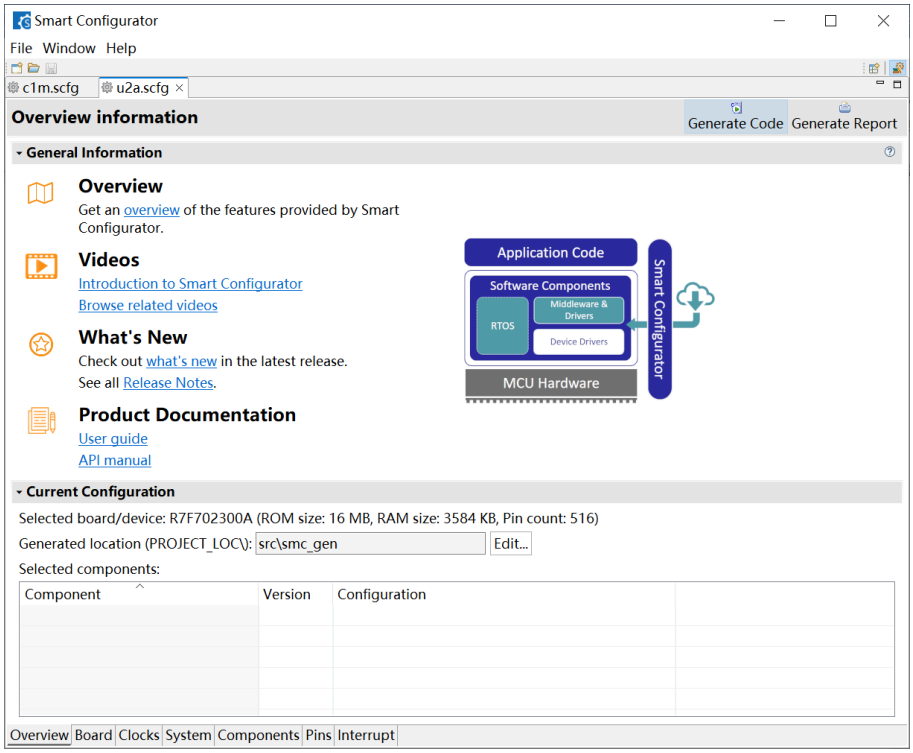

Figure 3-4 Smart Configurator View

3.4.4 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], [Board Function] and [Symbolic name].

- [Assigned Function] displays the assignment status of the pin setting.
- [Board Function] displays the initial pin setting information of the board.
- [Symbolic Name] displays the symbolic name defined by user for the pin. Macro definition for the symbolic name will be generated together with port read or write functions in Pin.h file.

The initial pin setting information of the board is the pin information of the board selected by [Board:] on the [Board] page (refer to "4.1.2 Selecting the board" and “Pin setting using board pin configuration information").

Note:
Symbolic Name feature is not applied to SC for RH850/F1KM and SC for RH850/F1KH.

Symbolic Name feature is not applied to APORT, JPORT and IPORT.

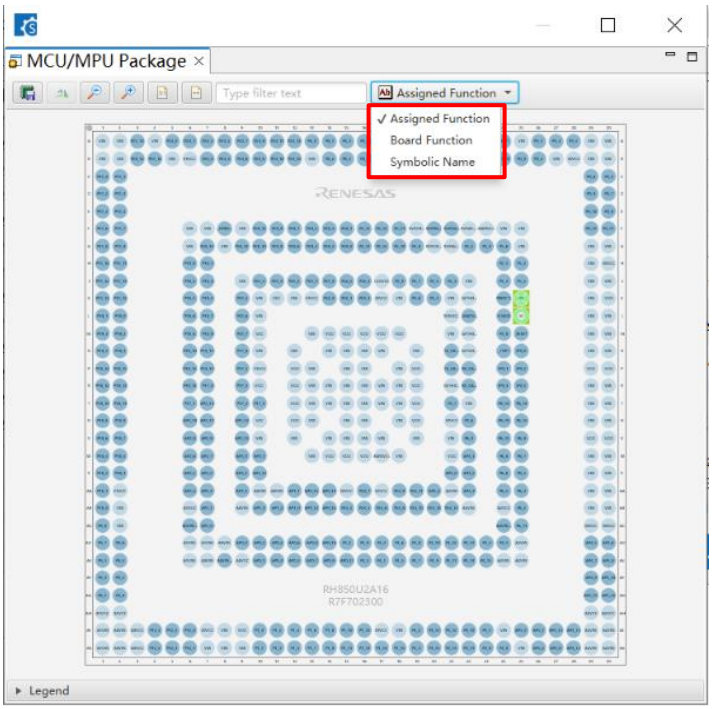

Figure 3-5 MCU/MPU Package View

3.4.5 Console view

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

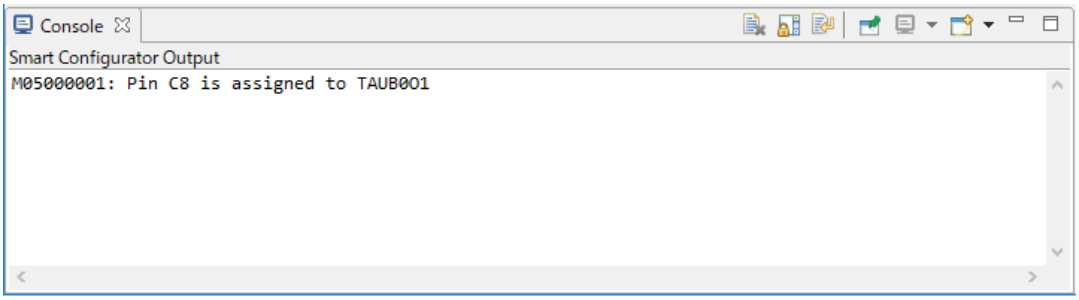

Figure 3-6 Console View

3.4.6 Configuration Problems view

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

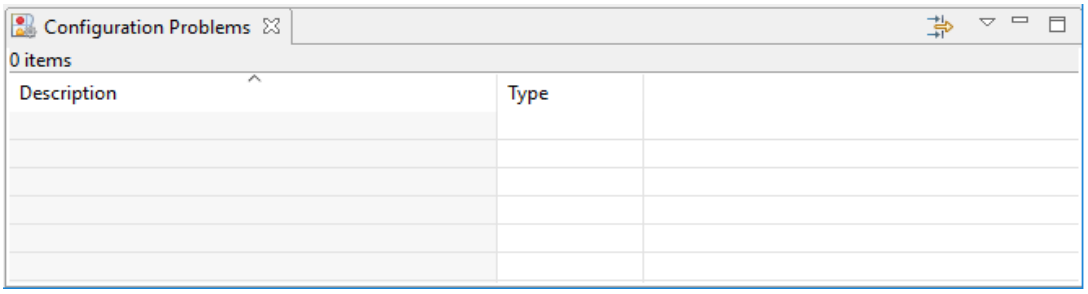

Figure 3-7 Configuration Problems View

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 device

Click on the [...] button to select a device. This procedure is not required if you start the Smart Configurator from CS+.

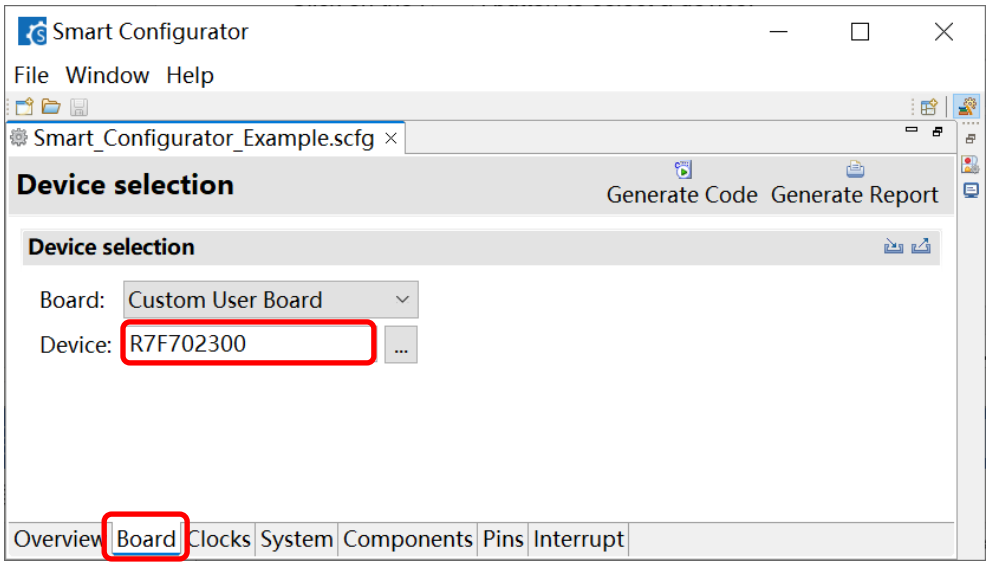

Figure 4-1 Selecting the Device

Note: Device change is not reflected to the device (microcontroller) of CS+ project.

4.1.2 Selecting the 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.

If the device shown in [Device] is different with device in file .bdf, dialog [Confirm device change] will popup:

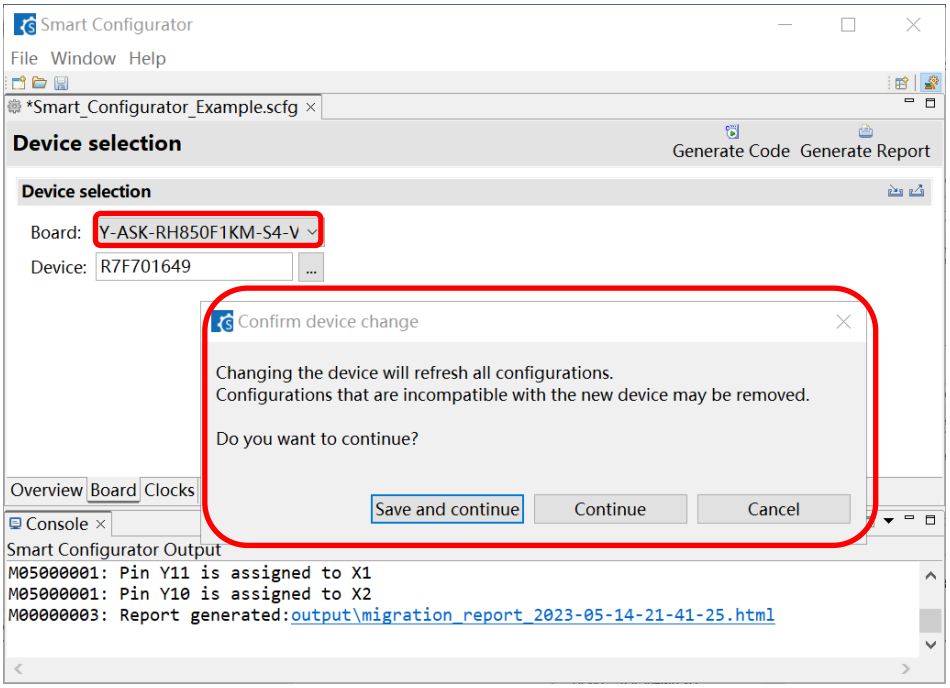

Figure 4-2 Selecting the Board with different device

If the device shown in [Device] is same as the device in file .bdf, dialog [Confirm board change] will popup:

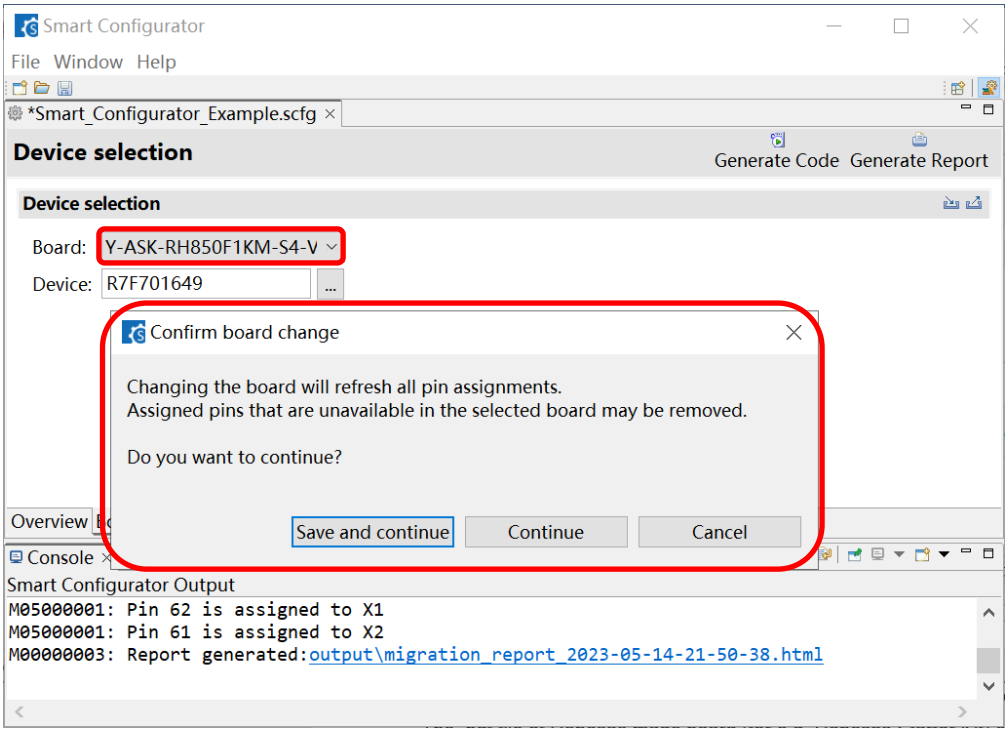

Figure 4-3 Selecting the Board with same device

Note: Depending on the board selected, the device will change, Device change is not reflected to the device (microcontroller) of CS+ project.

4.1.3 Exporting board settings

The board settings can be exported for later reference. 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.

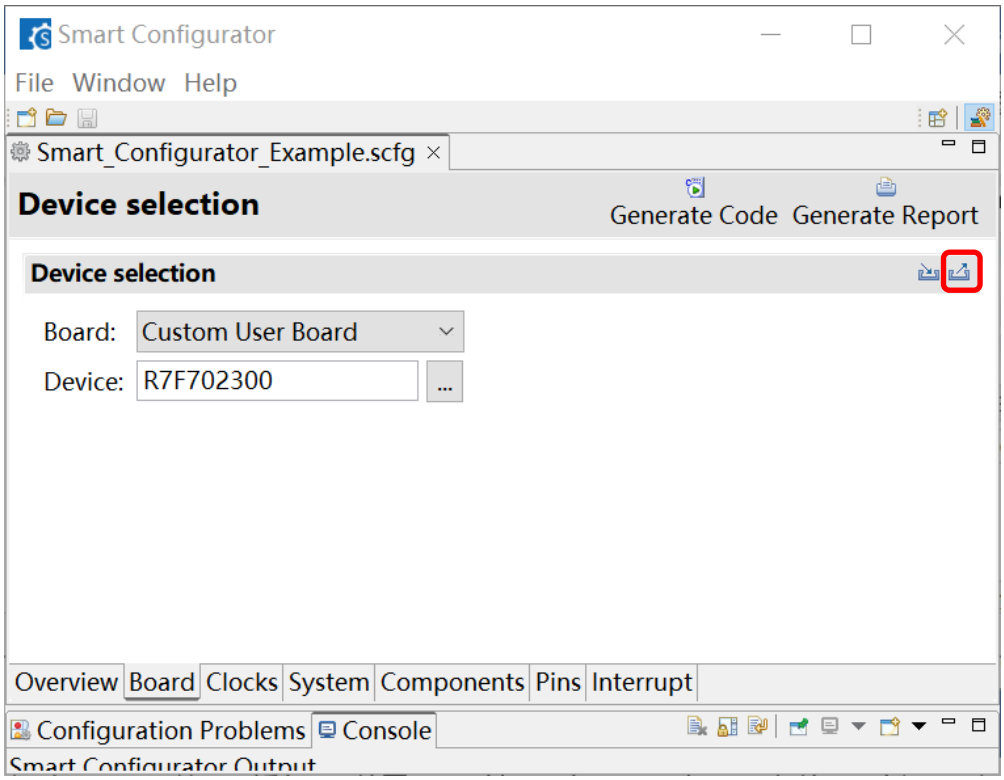

Figure 4-4 Exporting Board Settings (bdf Format)

4.1.4 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.

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

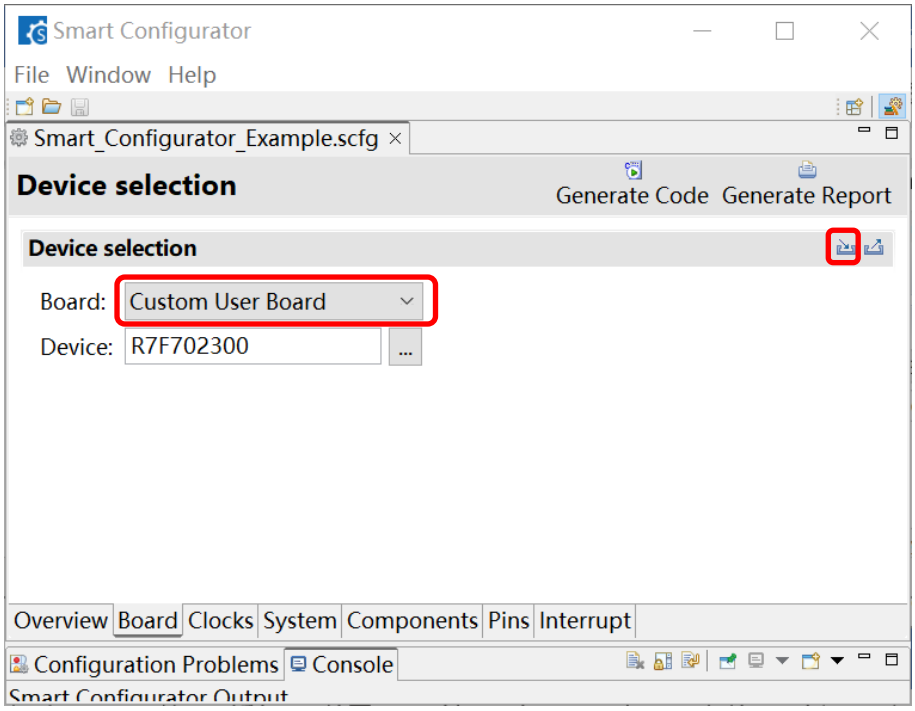

Figure 4-5 Importing Board Settings (bdf Format)

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 is used for all drivers.

Follow the procedure below to modify the clock settings.

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

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

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

(4) Enable the specific clock (only for RH850/F1KM and RH850/F1KH)

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

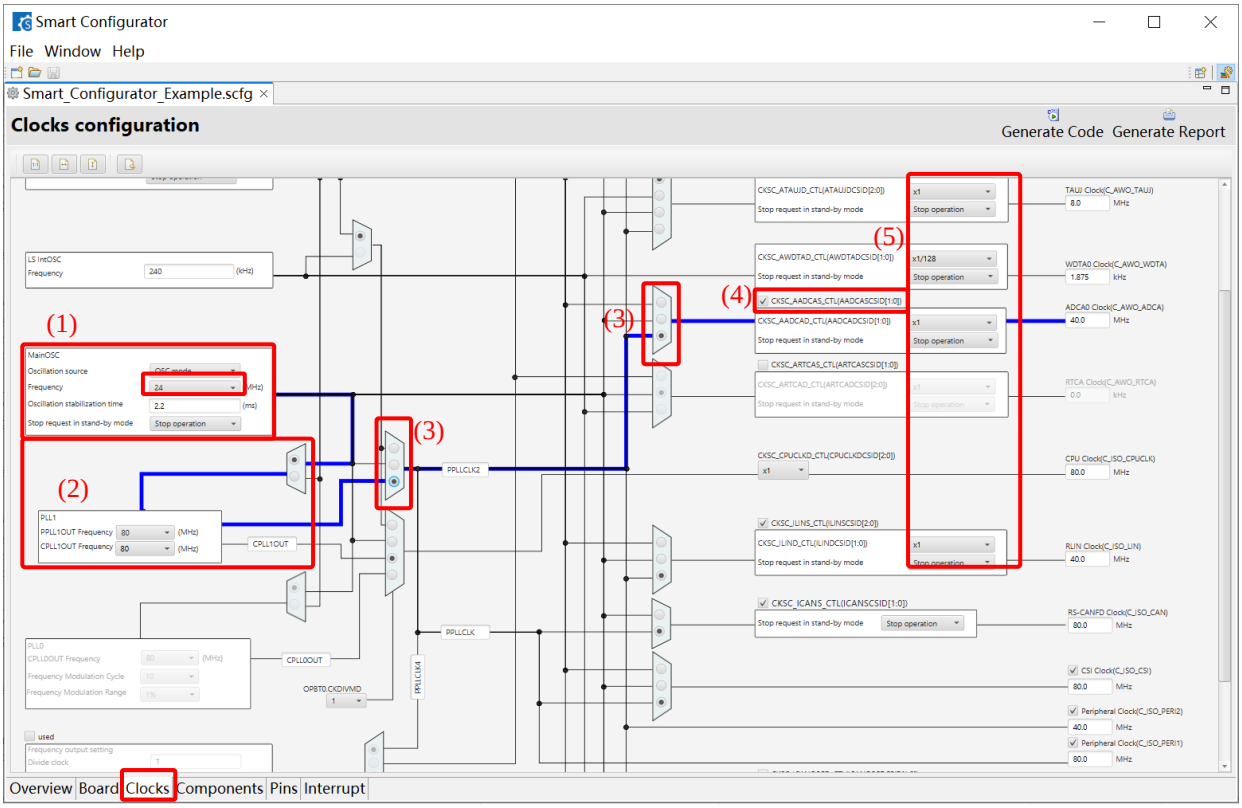

Figure 4-6 Clock Settings

4.3 System Settings (only for RH850/U2A)

You can select the CPUn (PEn) to be used at [System] tabbed page.

CPU0(PE0) is always selected to be used as the default setting.

Only RH850/U2A supports System settings.

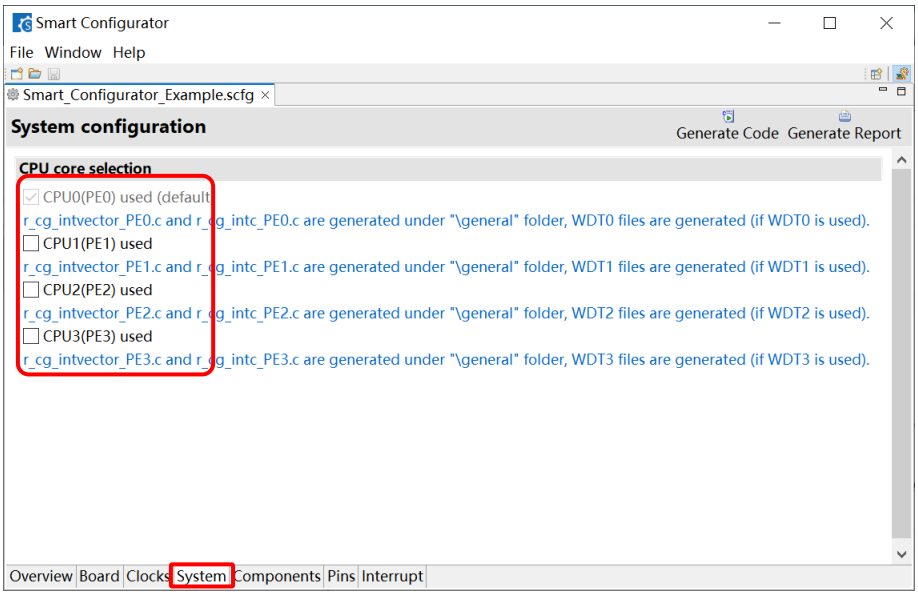

Figure 4-7 [System] Page

4.4 Component Settings

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

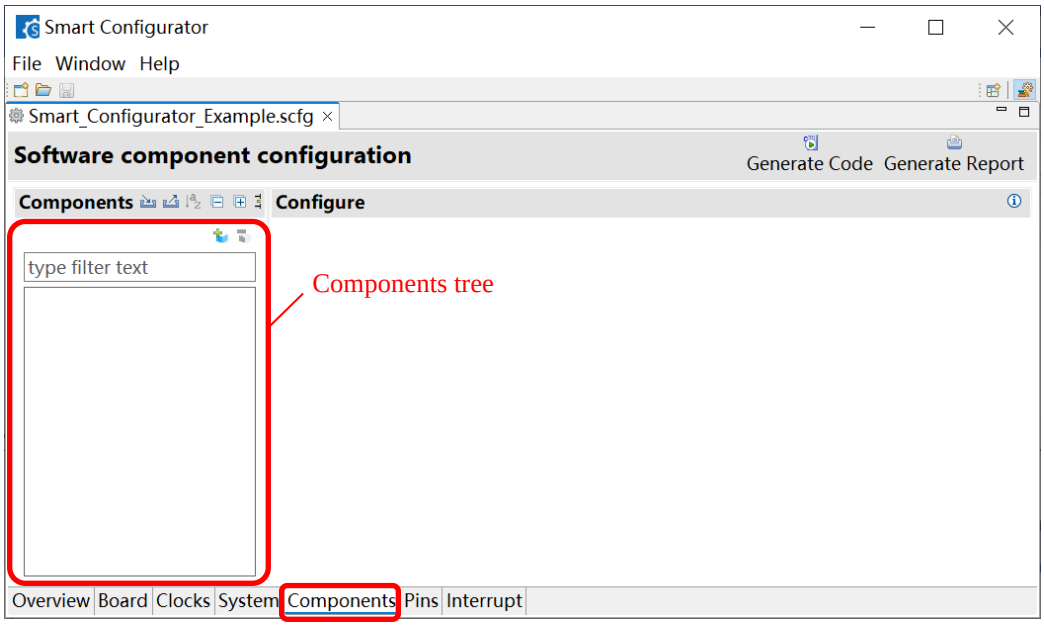

Figure 4-8 [Components] Page

4.4.1 Adding Code Generator components

The following describes the procedure for adding a component.

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

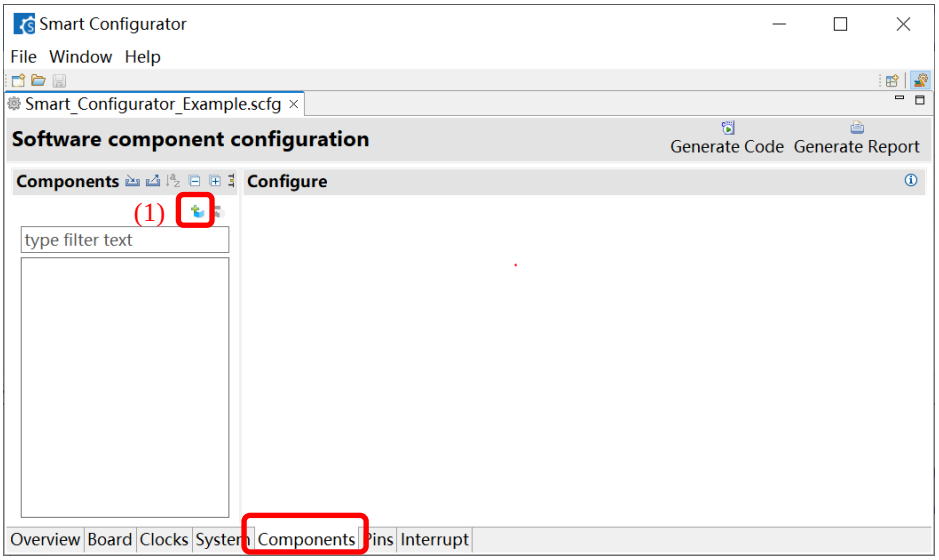

Figure 4-9 Adding a Component

(2) Select a component from the list in the [Software Component Selection] page of the [New Component] dialog box (for e.g., PWM Output).

(3) Click on [Next].

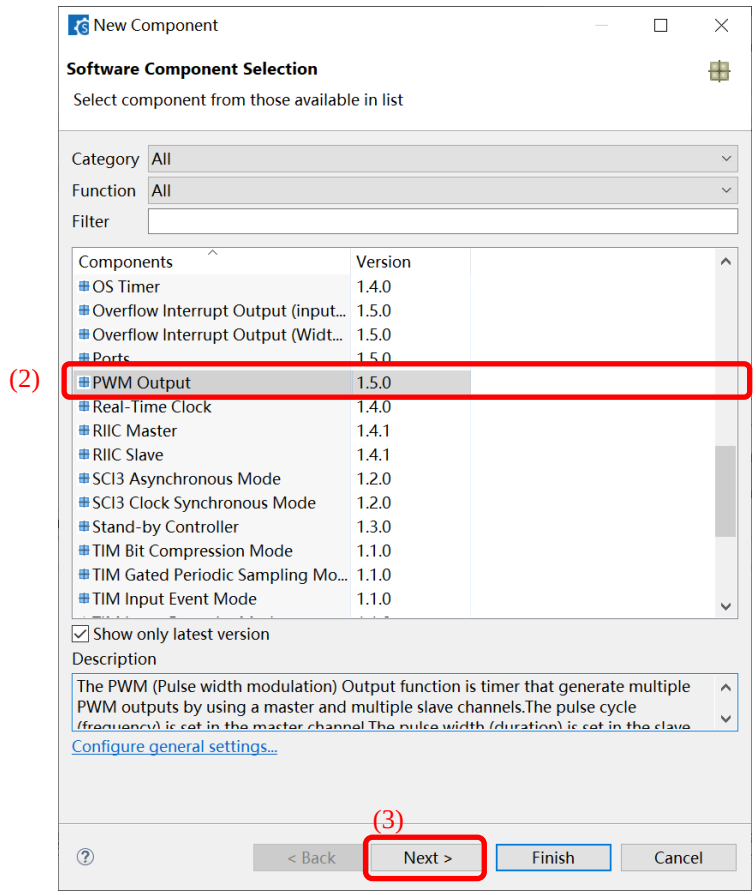

Figure 4-10 Adding a Code Generator Component

(4) Specify an appropriate configuration name in the [Add new configuration for selected component] dialog box or use the default name (for e.g.., Config_TAUB0).

(5) Select a hardware resource or use the default resource (for e.g., TAUB0).

(6) Click on [Finish].

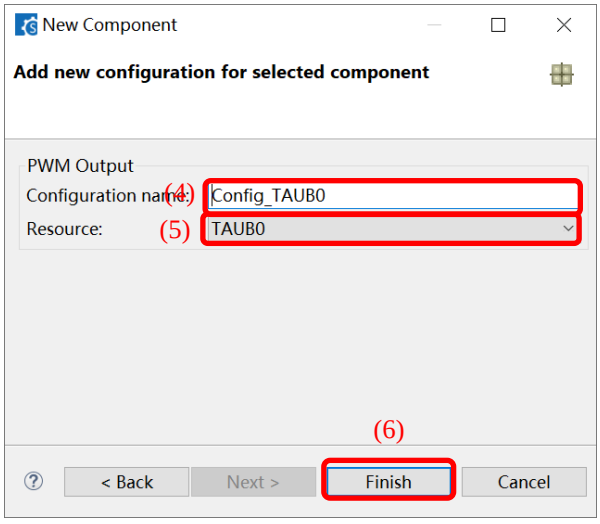

Figure 4-11 Adding a Component

4.4.2 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.

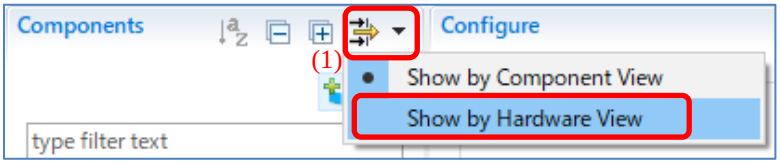

Figure 4-12 Switching to the Hardware View

(2) Double-click on a hardware resource node (for e.g., TAUB10 under Timer Array Unit B1) to open the [New Component] dialog box.

(3) Select a component from the list (for e.g.,PWM Output Function) to add a new configuration as described in “chapter 4.4.1 Adding Code Generator components”.

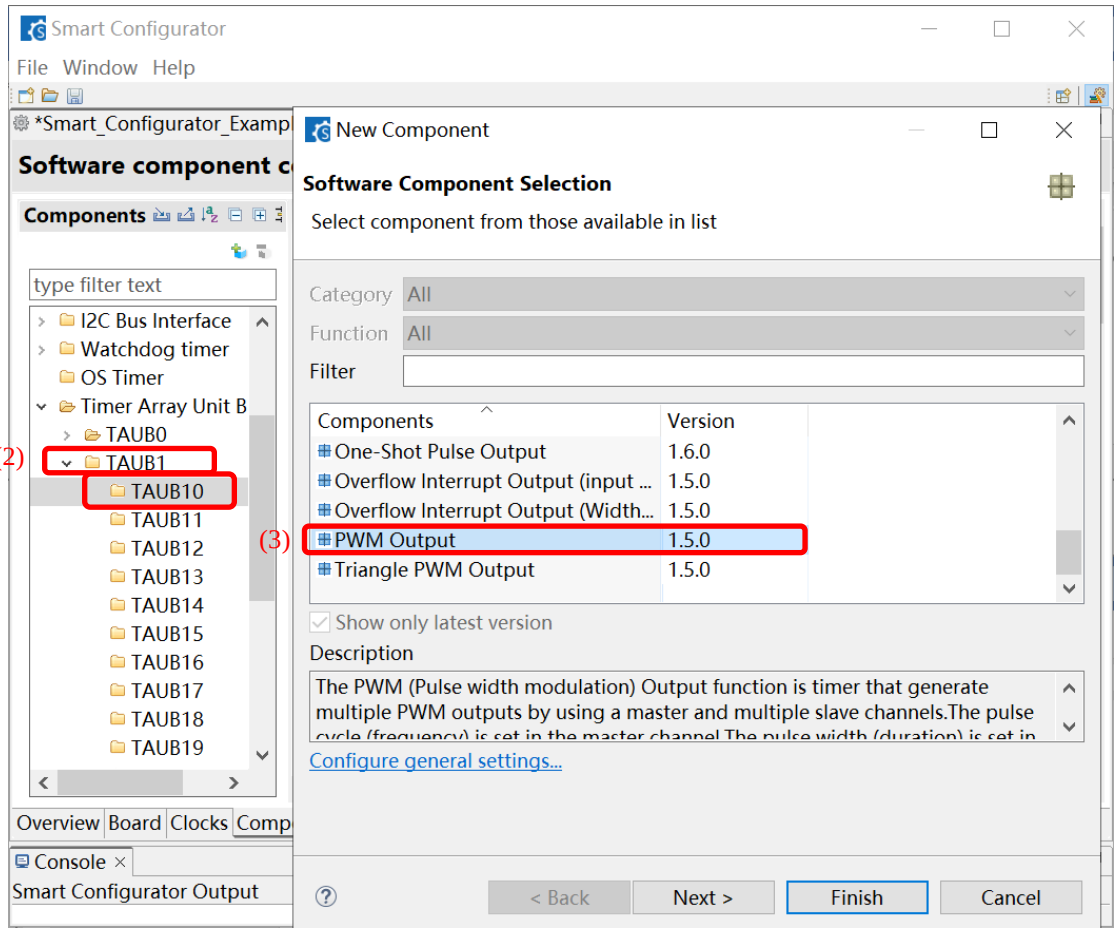

Figure 4-13 Adding a Component to the Hardware View

4.4.3 Removing software component

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

(1) Select a software component or multiple components (press and hold CTRL key while selecting the next component) on the Components tree.

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

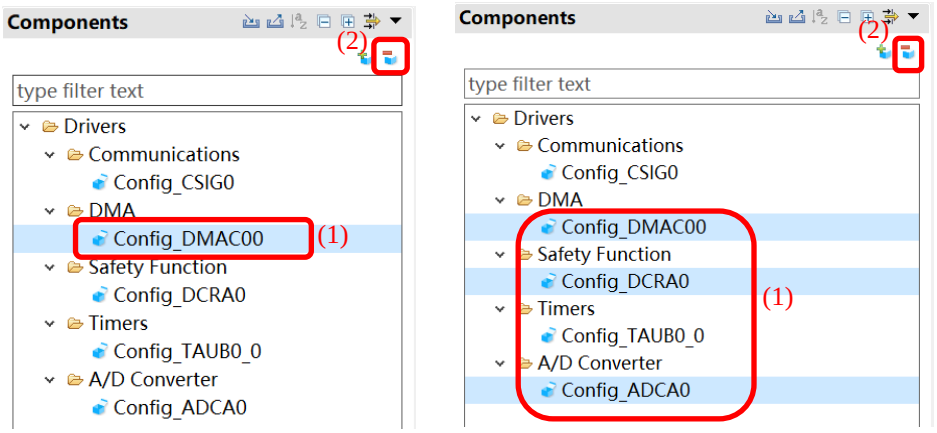

Figure 4-14 Removing a Component or Multiple Components

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

To delete the source files previously generated for the removed components from the CS+ project tree, click [(Generate Code)] icon.

4.4.4 Setting a Code Generator Component

Follow the procedure below to set up a Code Generator configuration.

(1) Select a code generator configuration from the Components tree (for e.g., Config_TAUB0).

(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 [PCLK/2] for [Clock source].

b. Select [Channel 1 slave], [Channel 2 slave], and [Channel 3 slave].

c. Specify [Pulse cycle] on the [Master0] tabbed page.

d. Specify [Duty] for each of the [Slave1], [Slave2], and [Slave3] tabbed pages.

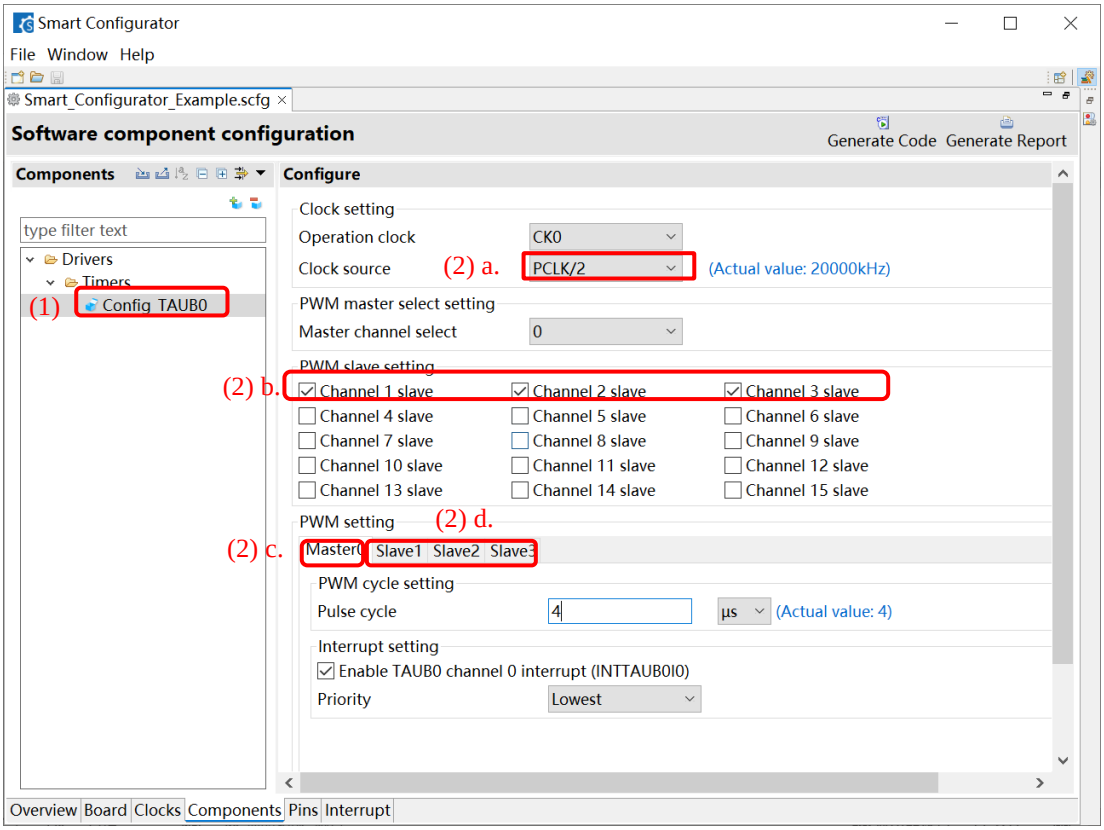

Figure 4-15 Setting a code generator configuration.

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

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

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

4.4.5 Changing the resource for a Code Generate Configuration

The Smart Configurator enables you to change the resource for a Code Generator configuration (for e.g., from TAUB0 to TAUB1). 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 configuration (for e.g., Config_TAUB0).

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

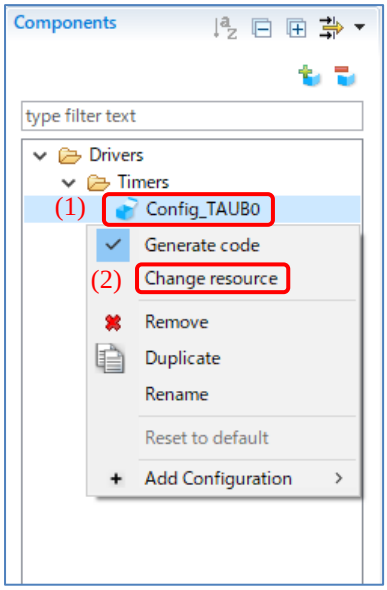

Figure 4-16 Changing the Resource

(3) Select a new resource (for e.g., TAUB1) in the [Resource Selection] dialog box.

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

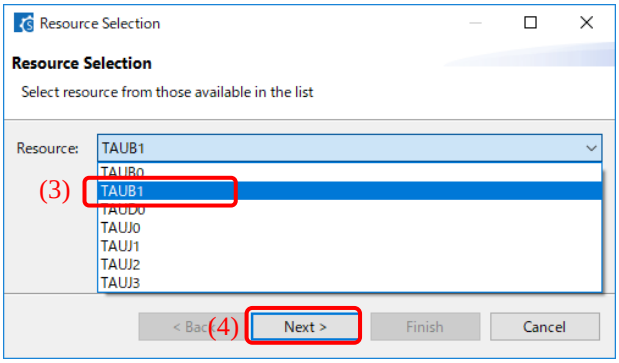

Figure 4-17 Components Page – Selecting a New Resource

(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].

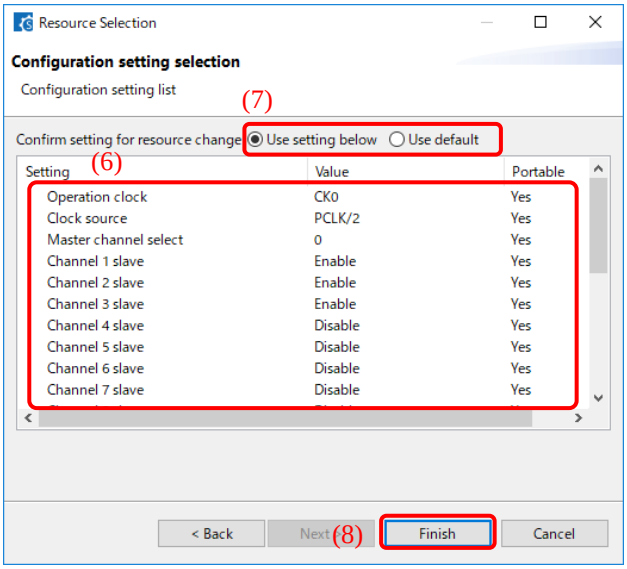

Figure 4-18 Checking the Settings of the New Resource

The resource is automatically changed (for e.g., changed from INTTAUB0I0 to INTTAUB1I0).

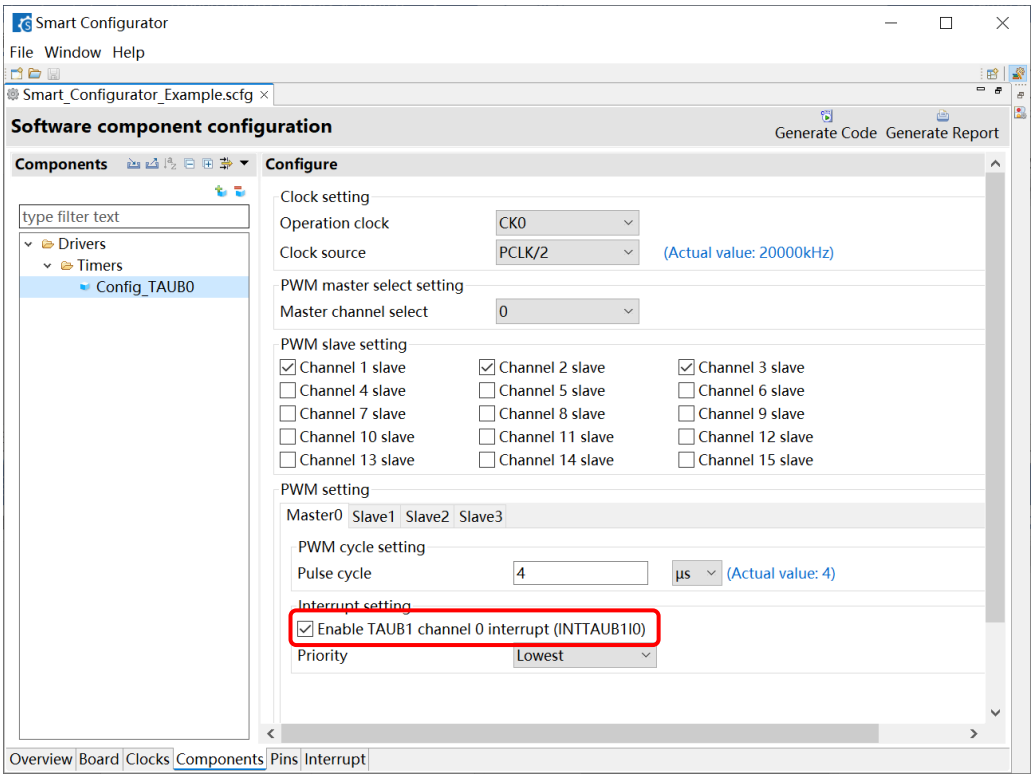

Figure 4-19 Resource Changed Automatically

To change the configuration name, follow the procedure below.

(9) Right-click on the configuration.

(10) Select [Rename] to rename the configuration (for e.g., change Config_TAUB0 to Config_TAUB1).

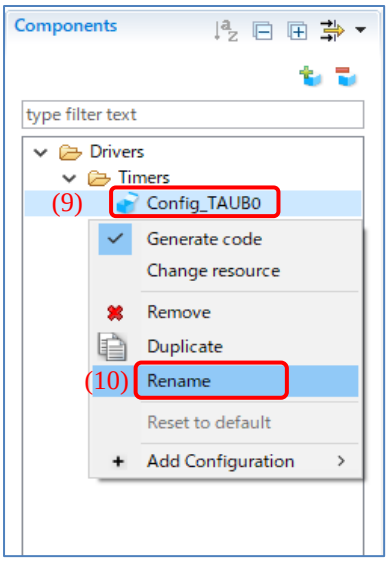

Figure 4-20 Renaming the Configuration

4.4.6 Configure general setting of the component

You can change the general setting of the component such as backup settings and API function output setting. If you want to change it, go to [Window] on the menu -> [Preferences], select [Smart Configurator] -> [Component].

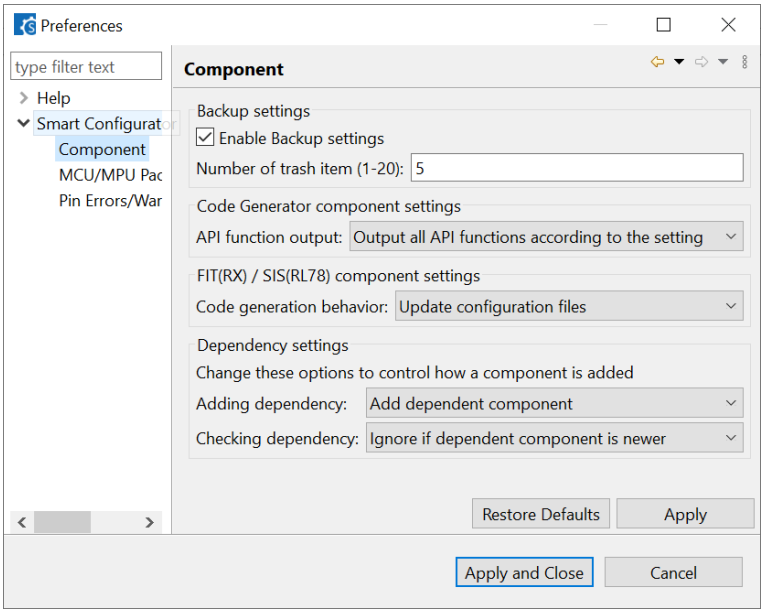

Figure 4-21 Configure general setting of component

Note:

1. You can select [Enable the Backup settings] and limit the number of folders created in the trash folder for backup purposes by setting the [Number of trash item (1-20)] option in the figure below. Once exceeding the limit, a folder with the newer timestamp will replace the oldest folder. Setting 0 will disable this backup feature.

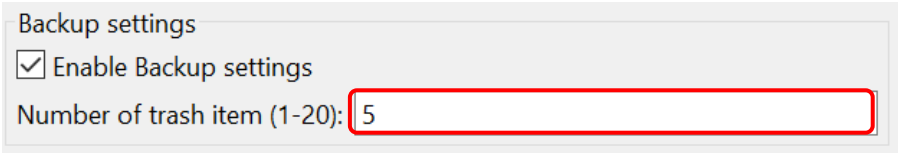

Figure 4-22 Trash number setting

2. If you want to only generate initialization API function, you can change to [Output only initialization API function] option in below figure. So that only void R_{ConfigurationName}_Create (void), void R_{ConfigurationName}_Create_UserInit (void) in *.h *, *c * are generated. If you change back to default option setting: [Output all API functions according to the setting], then all API functions will be generated again.

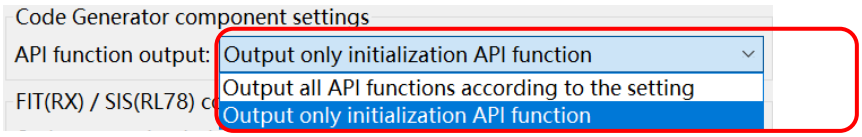

Figure 4-23 RH850 API function output setting

This feature is supported from Smart Configurator for RH850 V1.4.0.

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.

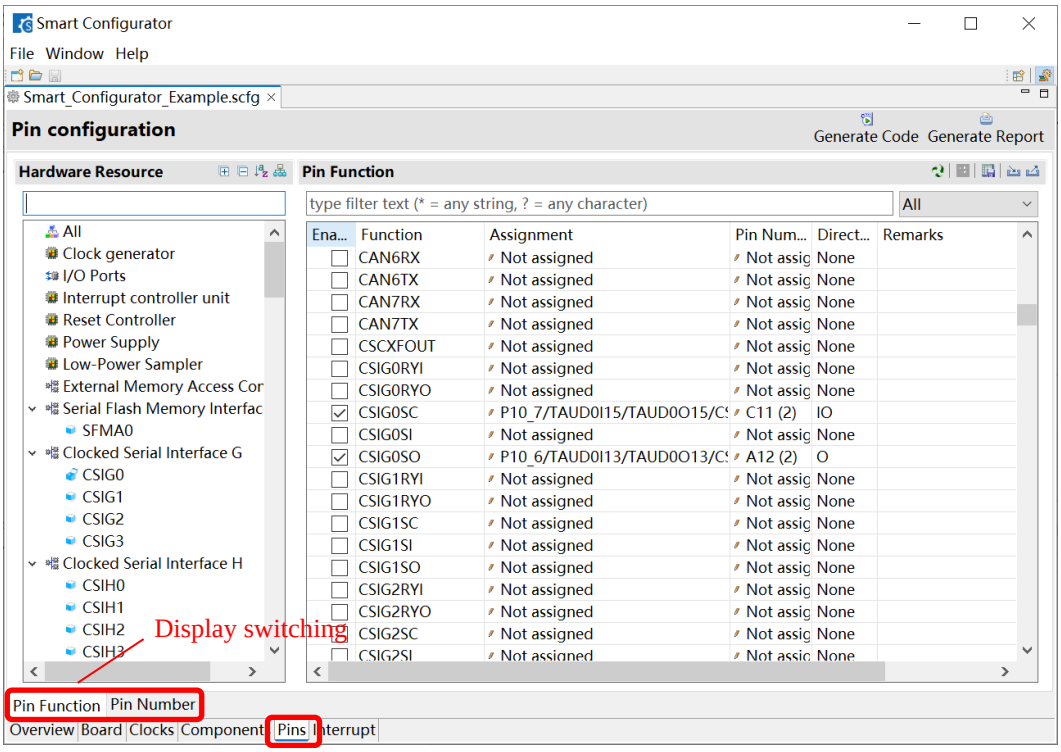

Figure 4-24 [Pins] Page ([Pin Function])

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

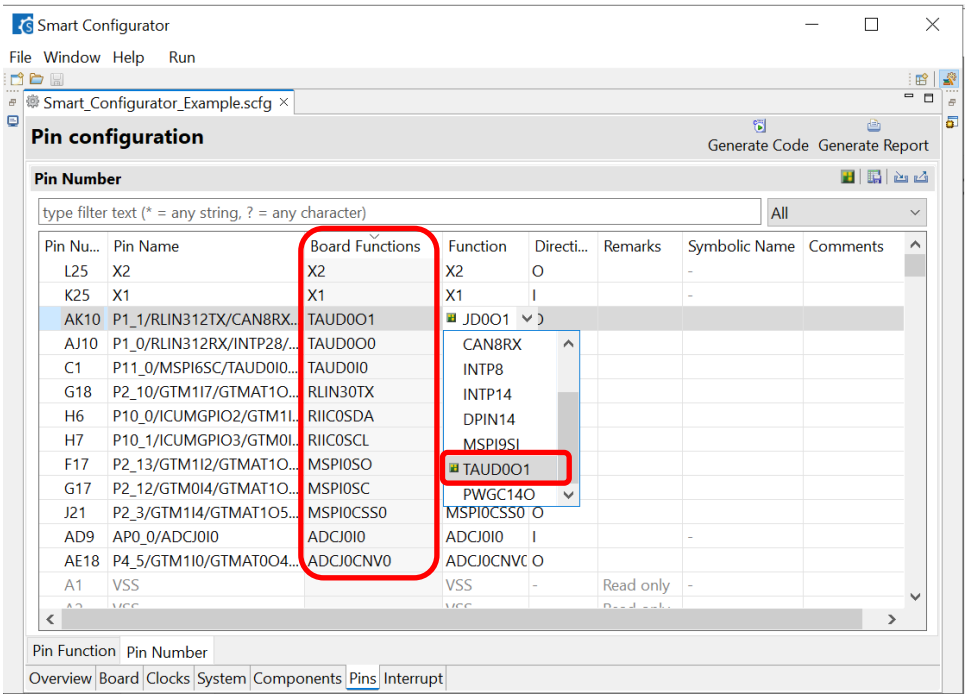

Figure 4-25 [Pins] Page ([Pin Number])

4.5.1 Changing the pin assignment of a software component

The Smart Configurator assigns pins to the software components 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_INTC).

(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 P10_13 to P11_2).

(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.

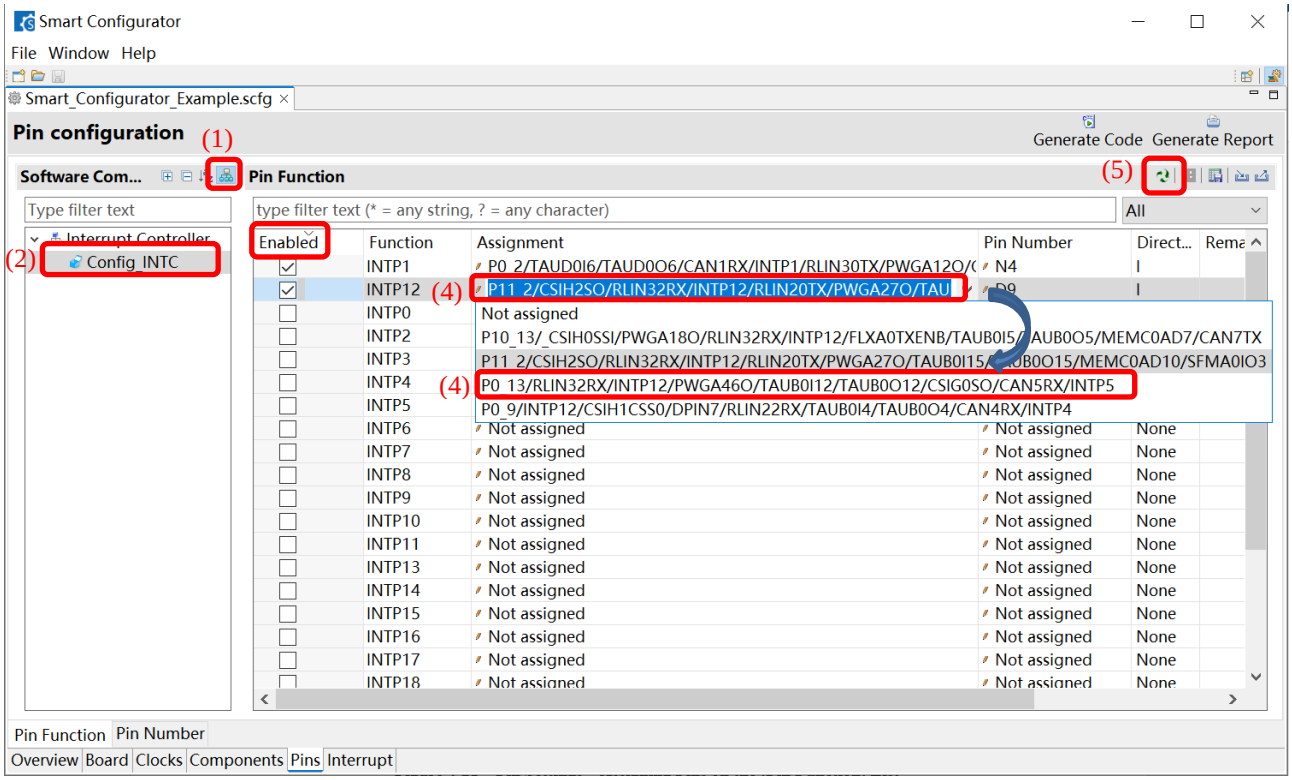

Figure 4-26 Pin Settings – Assigning Pins on the [Pin Function] List

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 "There is no software initializing this pin" on the list.

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…].

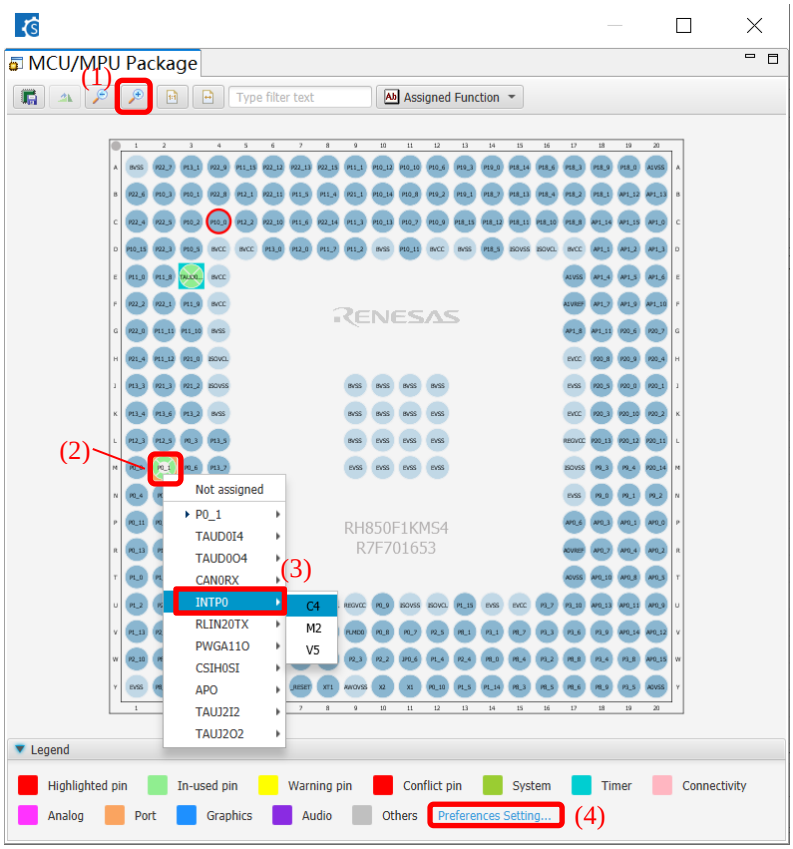

Figure 4-27 Assigning Pins Using the MCU/MPU Package View

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.

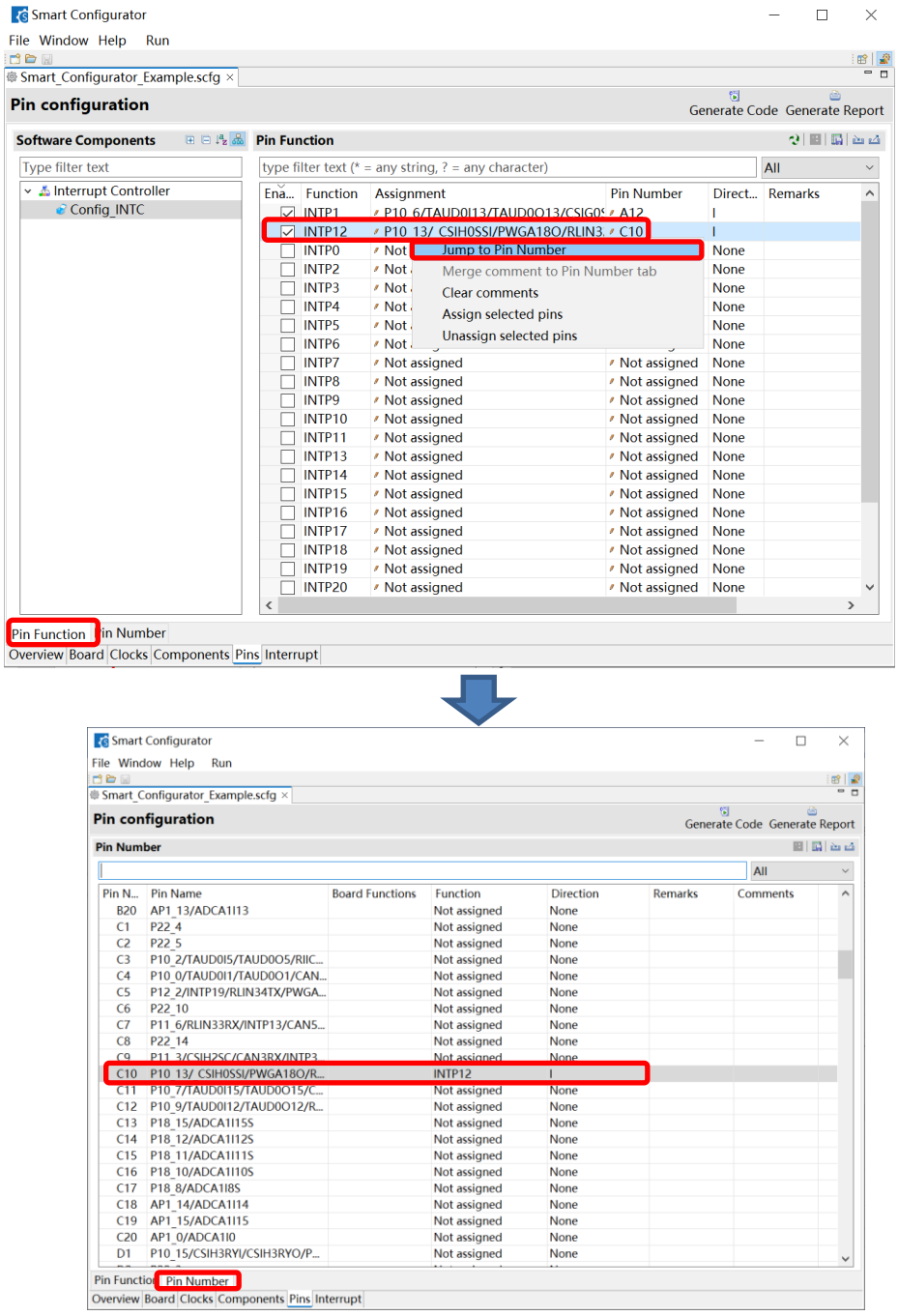

Figure 4-28 Jump to Pin Number

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.

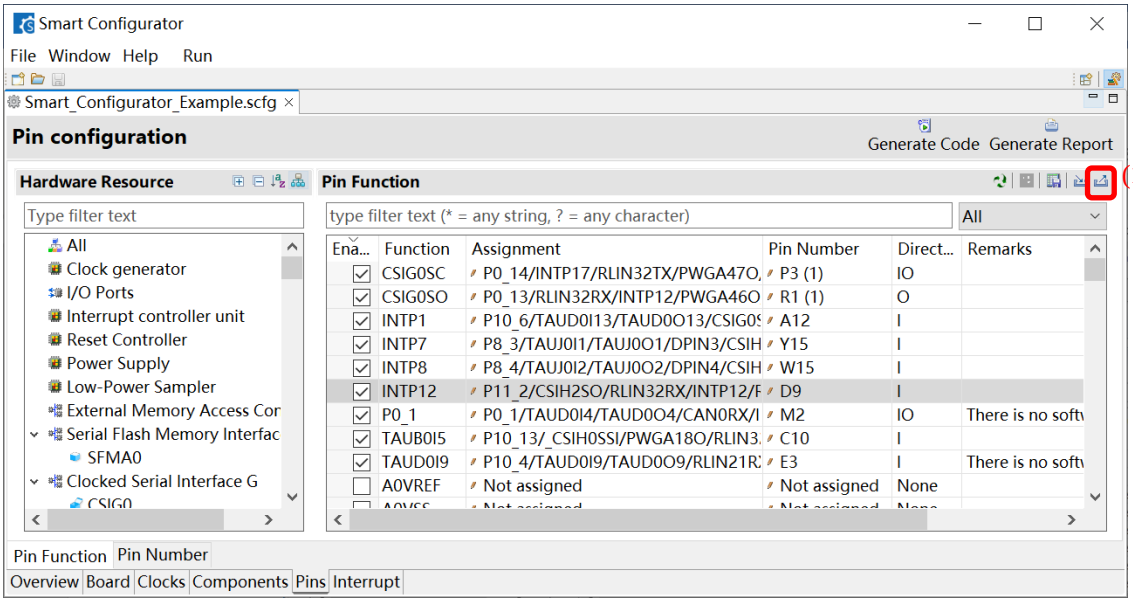

Figure 4-29 Exporting Pin Settings to an XML File

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.

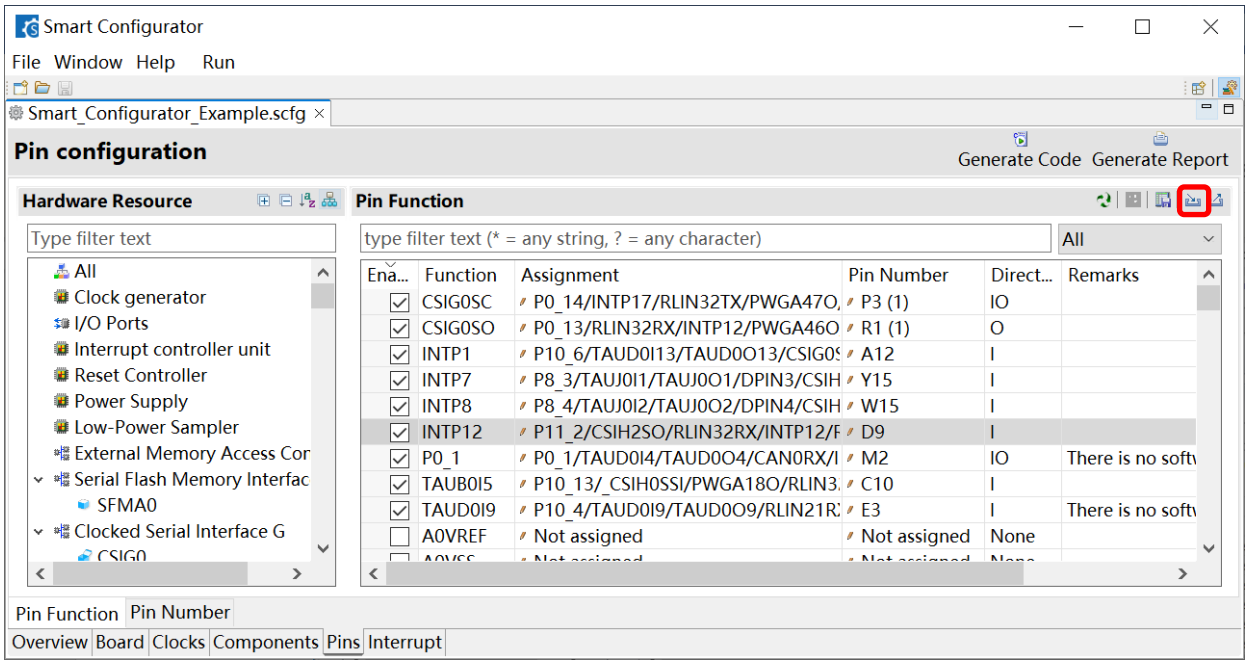

Figure 4-30 Importing Pin Settings from an XML File

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].

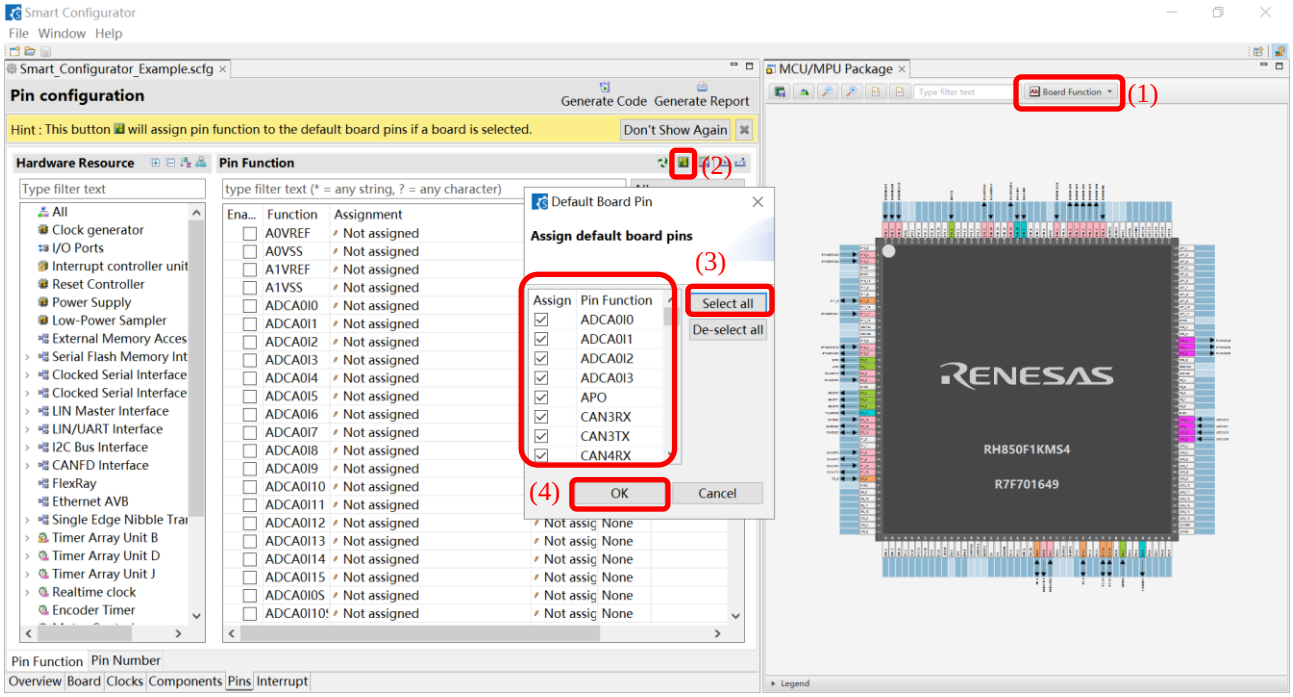

Figure 4-31 Setting for Initial Pin Configuration

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.

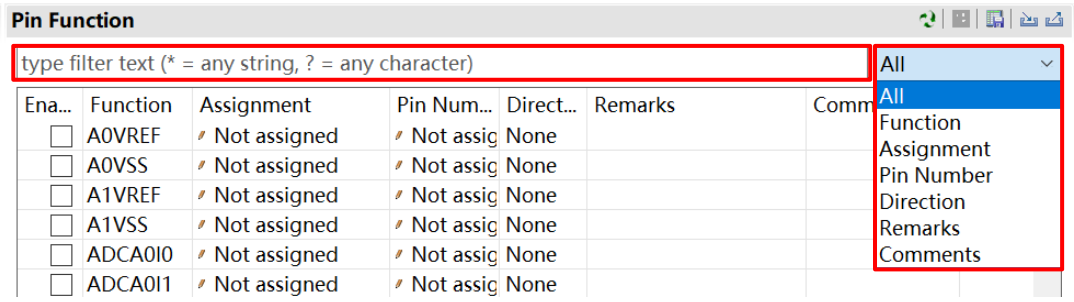

Figure 4-32 Filter for [Pin Function] tab

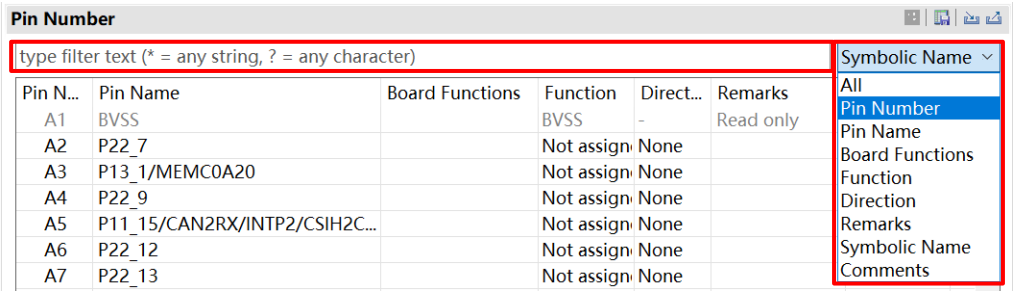

Figure 4-33 Filter for [Pin Number] tab

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, click Menu [Window]->[Preferences] to display the [Preferences] dialog. Then select [Smart Configurator] > [Pin Errors/Warnings] and use the combo boxes to change the errors/warning setting.

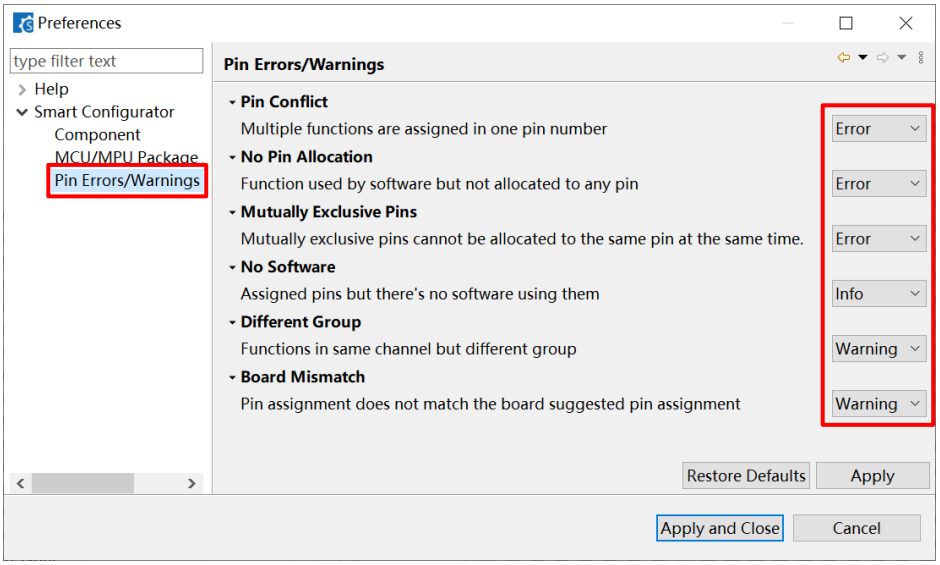

Figure 4-34 Pin Errors/Warnings settings at Preferences

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

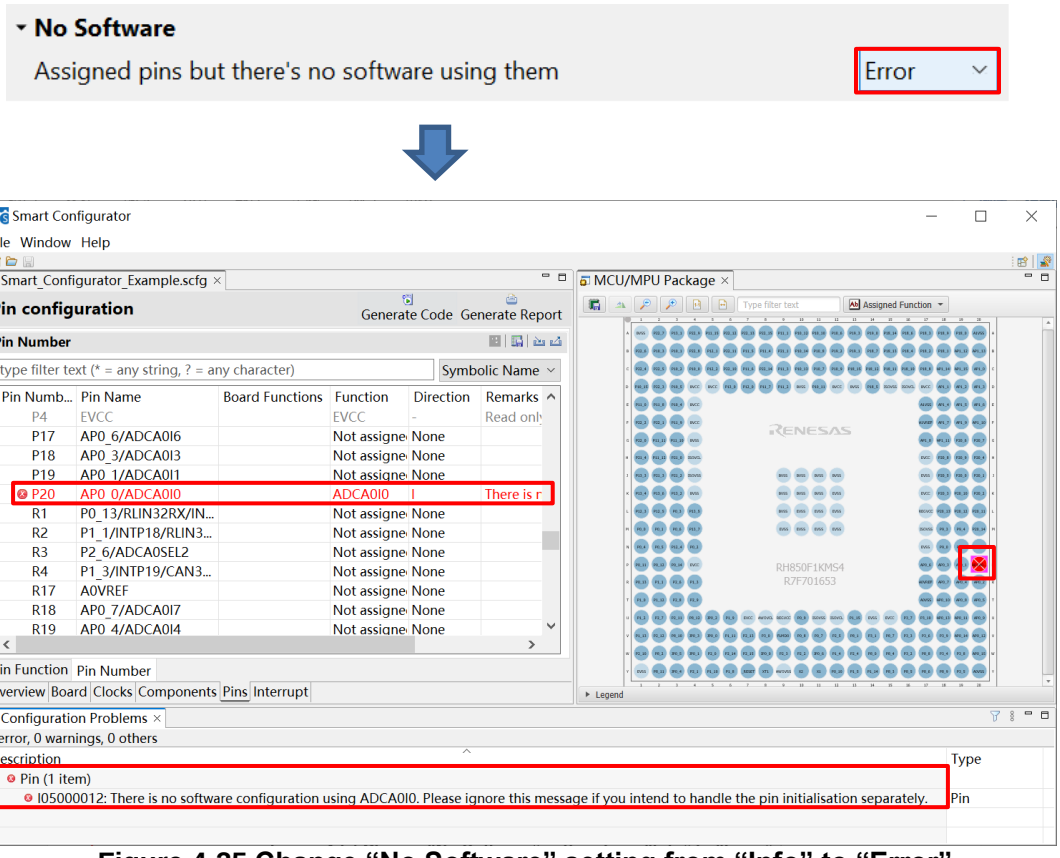

Figure 4-35 Change “No Software” setting from “Info” to “Error”

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. Generally, you can set the common settings such as interrupt priority levels, OS management, Interrupt Handler, Generate Entity and Generate Enable/Disable Function. For RH850/U2A, you can set PE*n* to decide if the interrupt is applied to the PE*n*.

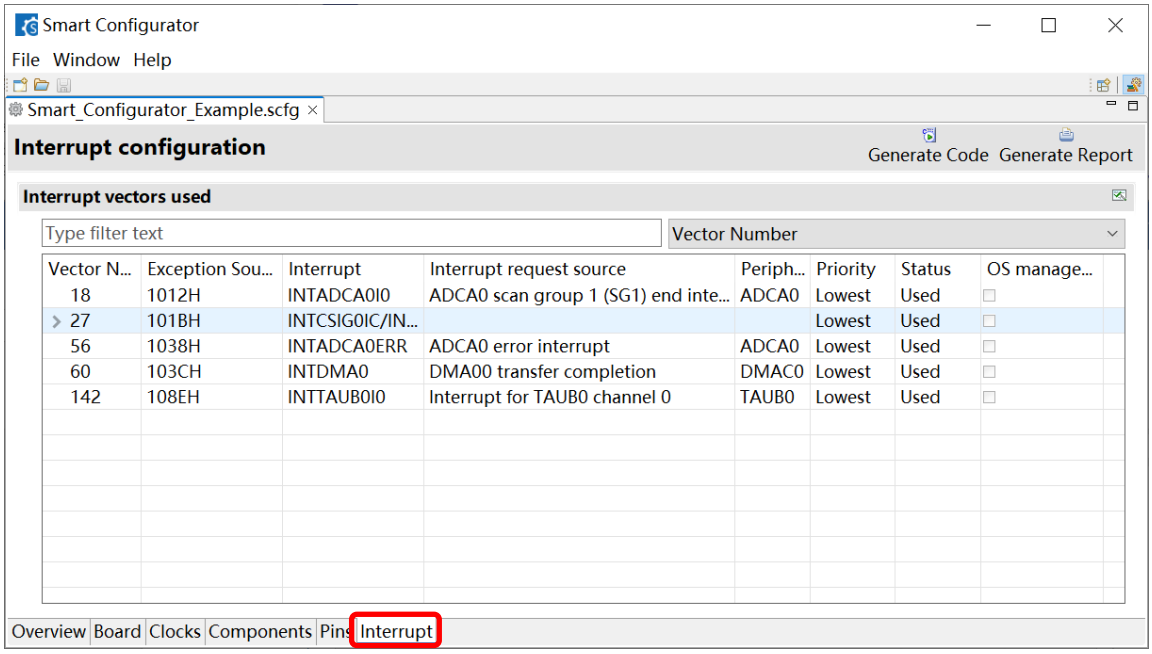

Figure 4-36 [Interrupt] Page

4.6.1 Changing the interrupt priority level and OS management setting

When an interrupt is used in a 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 [Interrupt] page.

(2) The [OS management] column becomes active for a project that uses RTOS (RI850V4). Selecting a checkbox in the column outputs the corresponding interrupt function in the interrupt format that can be managed by the OS.

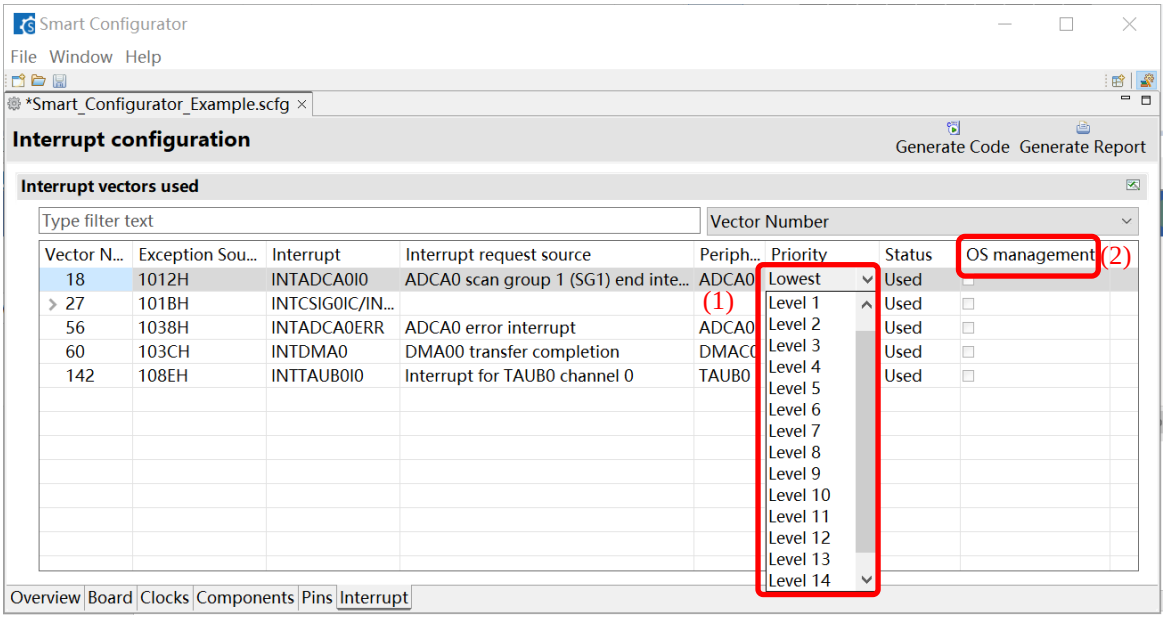

Figure 4-37 Interrupt Settings

4.6.2 Changing the PE*n* setting(RH850/U2A only)

In Smart Configurator for RH850, you can select which PE*n* to respond to the interrupt in use.
PE*n* can be set on the [Interrupt] page by below steps:
PE*n* is chosen to be used in [System] page (please refer to chapter 0,

(1) System Settings (only for RH850/U2A))
(2) Check or uncheck the checkbox in column PE*n* in [Interrupt] page to select which PE to respond to the interrupt. There are two types of interrupts:
a Connected to INTC1 of each PE, each PE selected in the PE*n* column can respond.
b Connected to INTC2 shared by multiple PEs, only one PE selected in PE*n* column can respond.

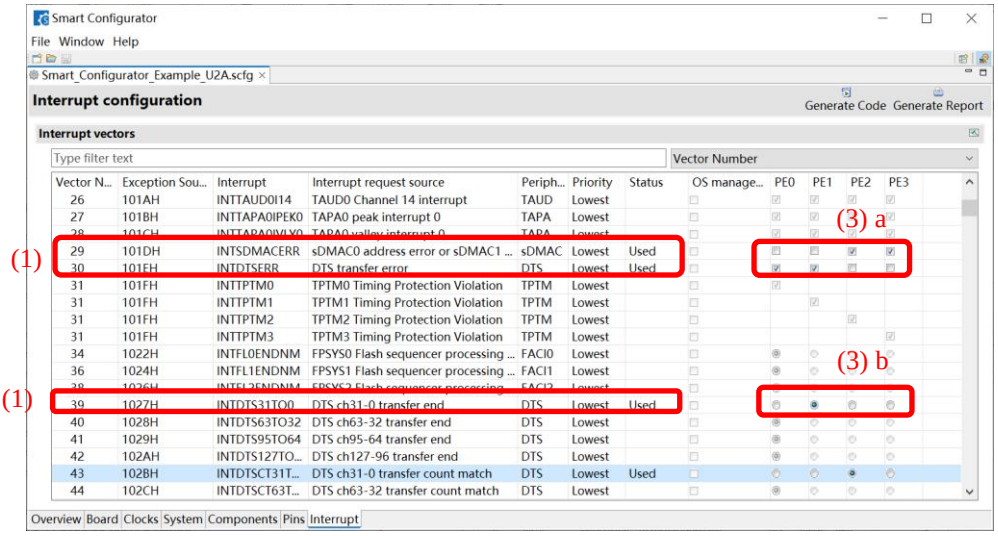

Figure 4-38 PE*n* setting

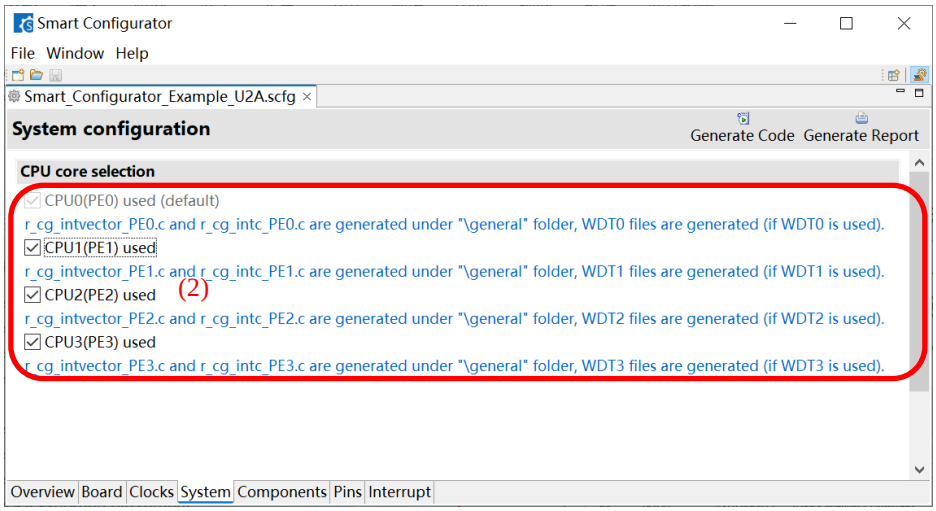

Figure 4-39 PE*n* is chosen to be used or unused in [System]

Note:
Only RH850/U2A supports PEn setting.
For other microcontrollers such as RH850/F1KH-D8, only PE0 is supported, so PEn cannot be selected by peripheral functions such as DMA or interrupts.

4.6.3 Changing the interrupt handler name, Generate Entity and Generate Enable/Disable Function setting

From Smart Configurator for RH850 V1.10.0, User-defined interrupt handler is supported for all RH850 devices supported by Smart Configurator. User can define the interrupt handler for each interrupt in [Interrupt] page and decide if to generate the interrupt handler entity and interrupt enable/disable function.

1) User can input his own interrupt handler name by editing column [interrupt Handler] manually.
   "eiintn" is displayed on column [Interrupt Handler] as default interrupt handler. Users can edit this column and enter a user-defined name (except the default name "eiintn") here according to below basic rule:
   - Only characters 'a'~'z', 'A'~'Z', '0'~'9' or '_' can be inputted.
   - The interrupt handler name starting with a number can't be inputted.
   - The interrupt handler can't be empty
   - The reserved interrupt handler name "eiintn" except for eiintn(n=current interrupt number) can't be inputted.
   - The any two same interrupt handler names can't be inputted.

   Note: interrupt handler which is used by components is non-editable.

2) User can specify whether user-defined interrupt handler entity is generated by checking/unchecking [Generated Entity].
   The default setting is always checked. When you change the setting to unchecked, the interrupt handler code won't be generated by Smart Configurator, then user can use his own handler code.

3) Users can specify whether the interrupt enable/disable function is generated by checking/unchecking [Generate Enable/Disable Function].
   The default setting is unchecked. When user changes the setting to checked, a pair of interrupt enable/disable functions will be provided in "*r_smc_interrupt.c*" file. User can easily use interrupt by calling these APIs directly.
   For code samples of the interrupt enable/disable functions, see

Figure 4-41 Interrupt Enable/Disable Functions Code Example.

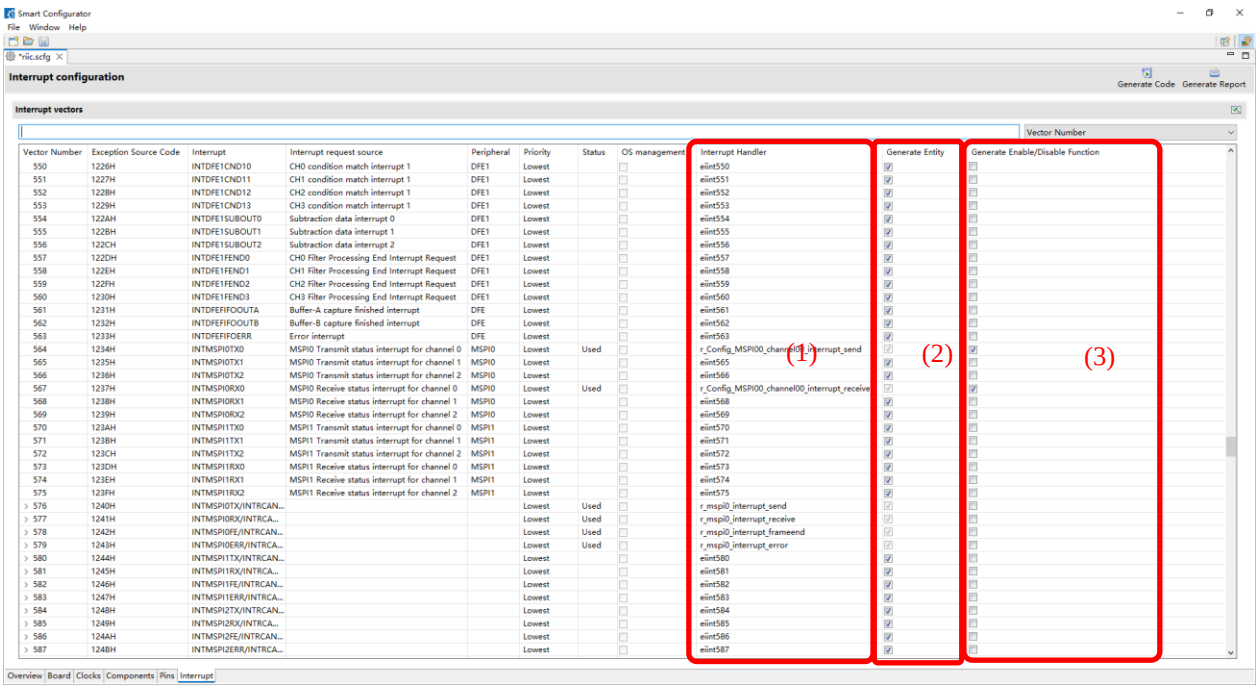

Figure 4-40 Interrupt Handler and Generate Entity settings

r_smc_interrupt.c

```
/***********************************************************************************************************************
* File Name        : r_smc_interrupt.c
* Version          : 1.3.0
* Device(s)        : R7F702301BEBBA
* Description      : None
***********************************************************************************************************************/
/***********************************************************************************************************************
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************************************************************
#include "r_cg_macrodriver.h"
#include "r_cg_userdefine.h"
#include "r_smc_interrupt.h"

/***********************************************************************************************************************
void R_Interrupt_Create(void)
{

}

void r_Config_MSPI00_channel00_interrupt_send_enable_interrupt(void)
{
    /* Clear INTMSPI0TX0 request and enable operation */
    INTC2.EIC244.BIT.EIRF244 = _INT_REQUEST_NOT_OCCUR;
    INTC2.EIC244.BIT.EIMK244 = _INT_PROCESSING_ENABLED;
}

void r_Config_MSPI00_channel00_interrupt_send_disable_interrupt(void)
{
    /* Disable INTMSPI0TX0 operation and clear request */
    INTC2.EIC244.BIT.EIMK244 = _INT_PROCESSING_DISABLED;
    INTC2.EIC244.BIT.EIRF244 = _INT_REQUEST_NOT_OCCUR;
}

void r_Config_MSPI00_channel00_interrupt_receive_enable_interrupt(void)
{
    /* Clear INTMSPI0RX0 request and enable operation */
    INTC2.EIC245.BIT.EIRF245 = _INT_REQUEST_NOT_OCCUR;
    INTC2.EIC245.BIT.EIMK245 = _INT_PROCESSING_ENABLED;
}

void r_Config_MSPI00_channel00_interrupt_receive_disable_interrupt(void)
{
    /* Disable INTMSPI0RX0 operation and clear request */
    INTC2.EIC245.BIT.EIMK245 = _INT_PROCESSING_DISABLED;
    INTC2.EIC245.BIT.EIRF245 = _INT_REQUEST_NOT_OCCUR;
}
```

Figure 4-41 Interrupt Enable/Disable Functions Code Example

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., DMAC00), 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.

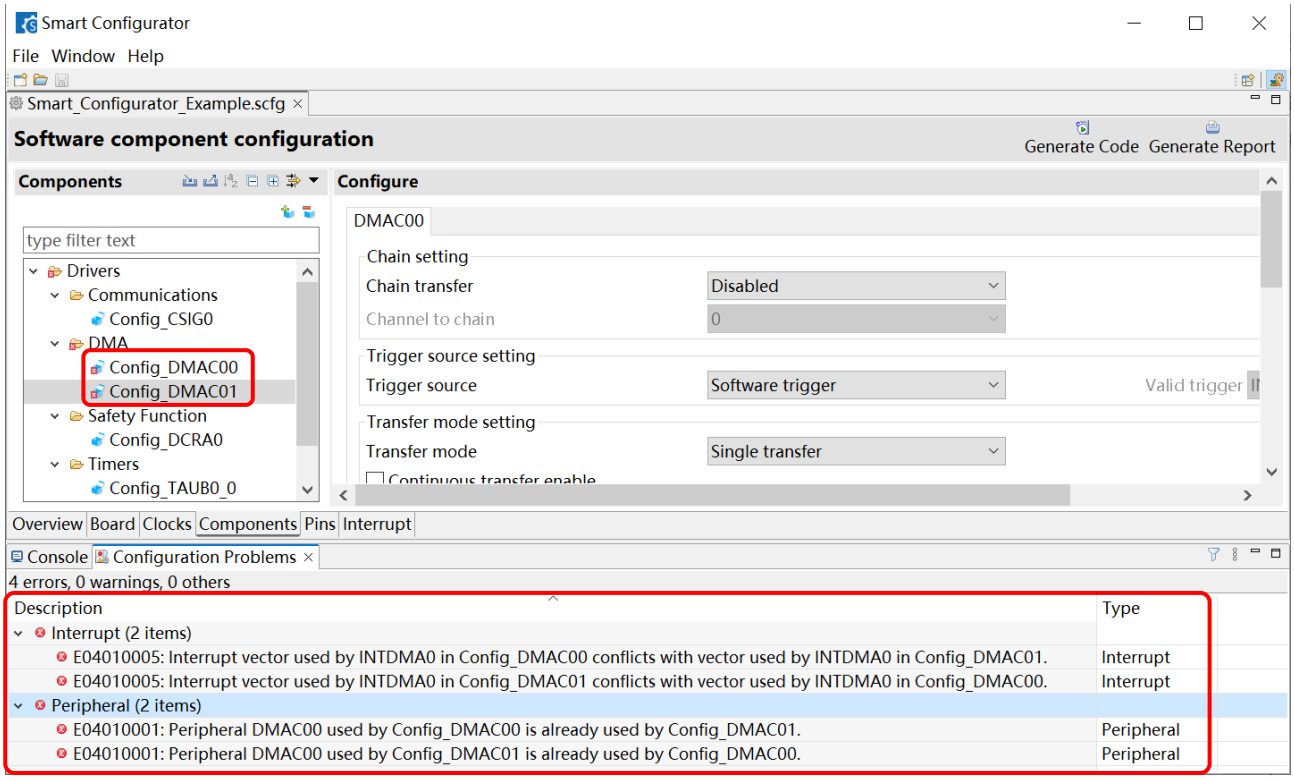

Figure 5-1 Resource Conflicts

5.2 Resolving pin conflicts

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

Detailed information regarding conflicts is displayed in the Configuration Problems view.

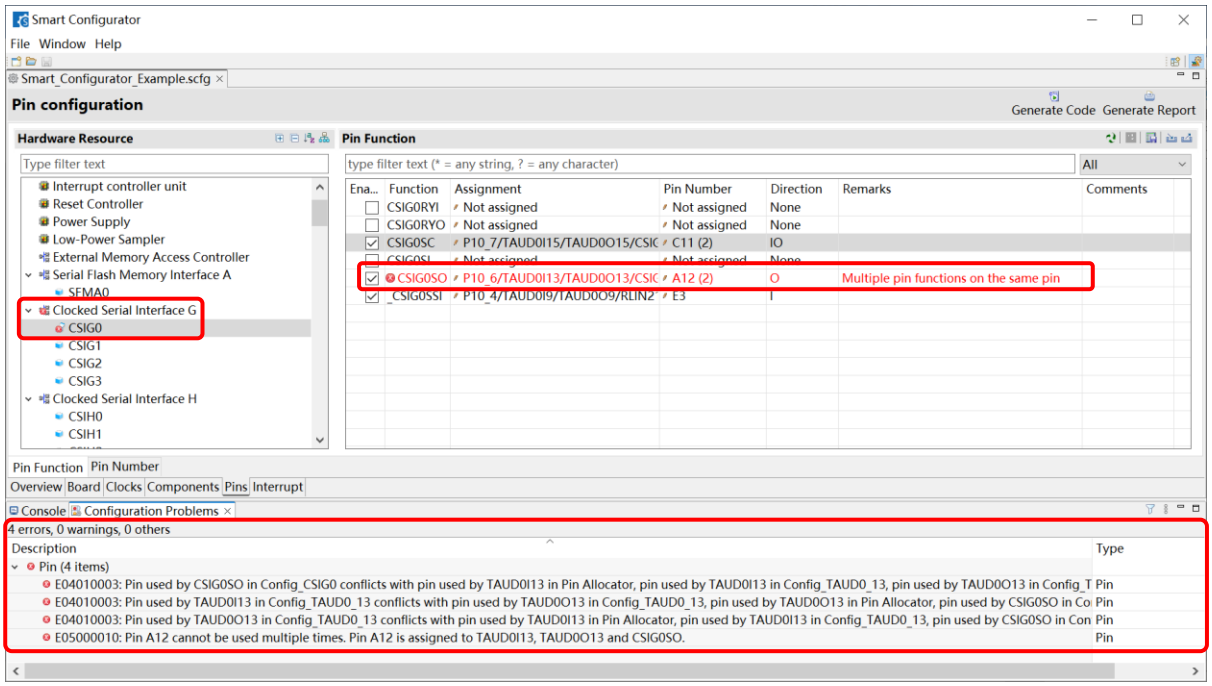

Figure 5-2 Pin Conflicts

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

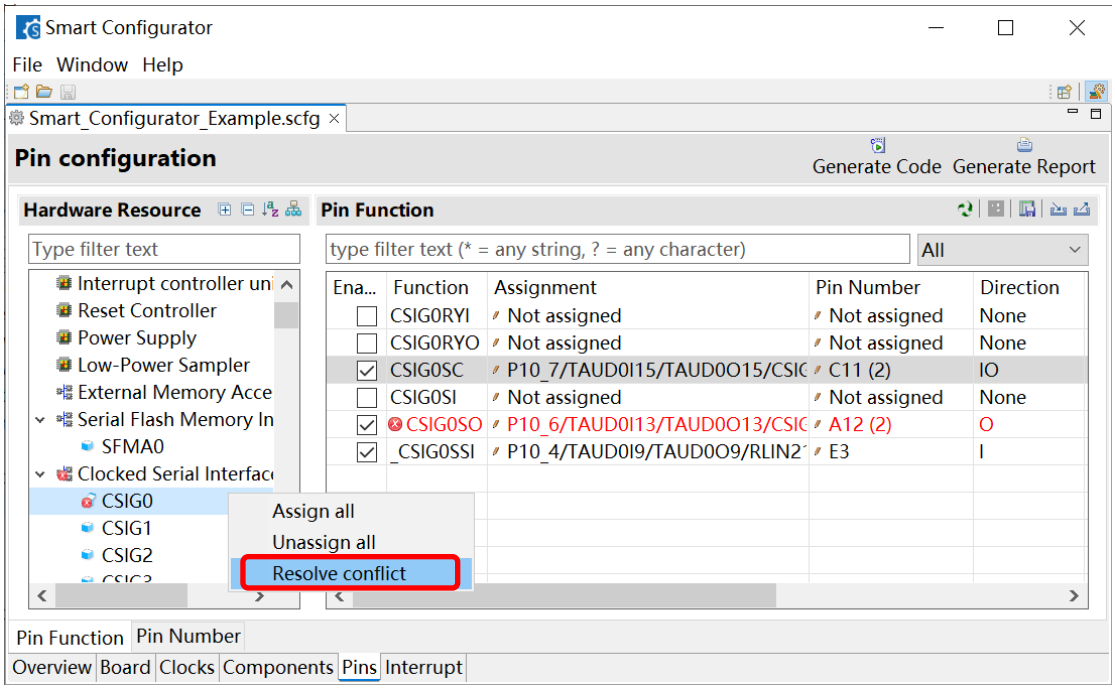

Figure 5-3 Resolving Pin Conflicts

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

6. Generating Source Code

6.1 Registering Generated Source Code with CS+

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

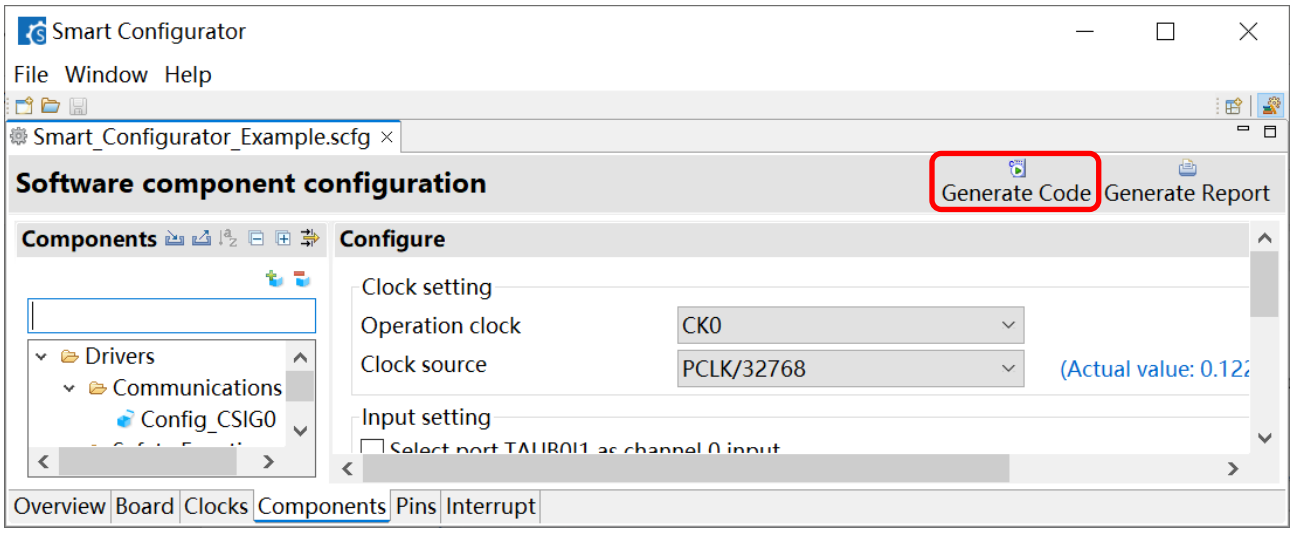

Figure 6-1 Generating a Source File

The Smart Configurator generates a source file in <ProjectDir>\src\smc_gen, and the file is registered with the given project of CS+. If your 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).

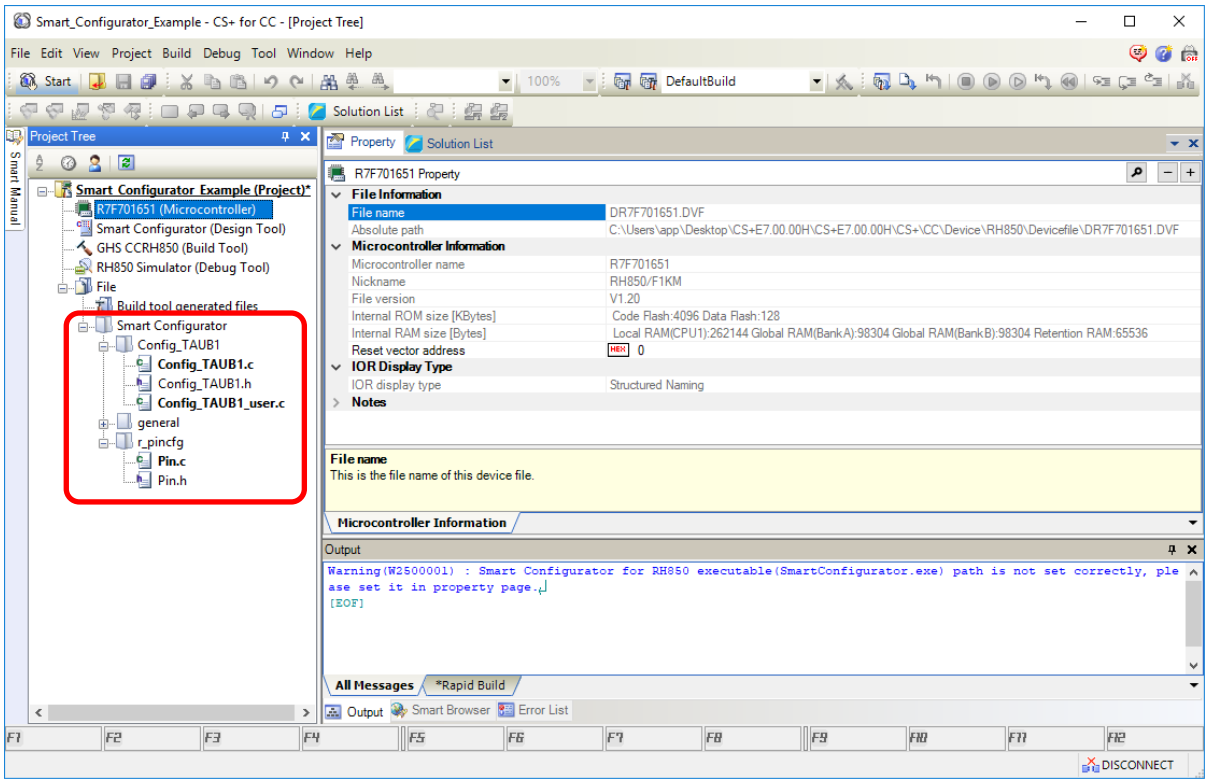

Figure 6-2 Registering a Source File with the CS+ Project

6.2 Loading files generated by All Toolchain (CC-RH, GHS, IAR)

When start the Smart Configurator standalone and select All Toolchain (CC-RH, GHS, IAR) as toolchain, Smart Configurator outputs the source files that can adapt to all three toolchains: CC-RH, IAR and GHS. When loading these generated files in CS+ project, please follow below procedures:

(1) Create a new project according to Project Wizard in CS+ project.

(2) Remove default file “main0.c” or “main.c” created by CS+ from CS+ project.

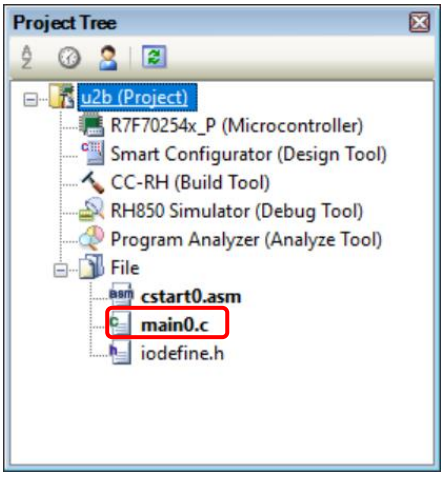

Figure 6-3 Remove main0.c

Note: “r_cg_main.c” generated by Smart Configurator generate is used as the entry function.

(3) Add include path of “iodefine.h” by [Compile Options].

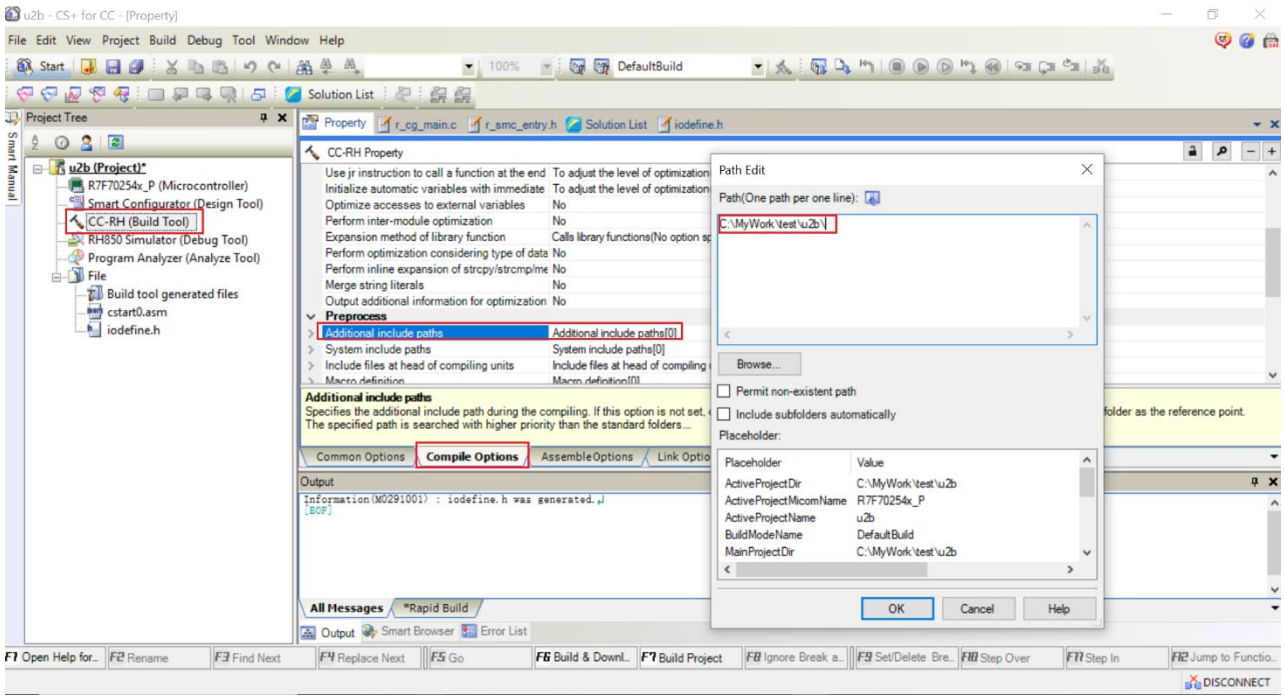

Figure 6-4 Add include path of “iodefine.h”

(4) Manually add the source files output by the Smart Configurator to the project, such as below figure shows:

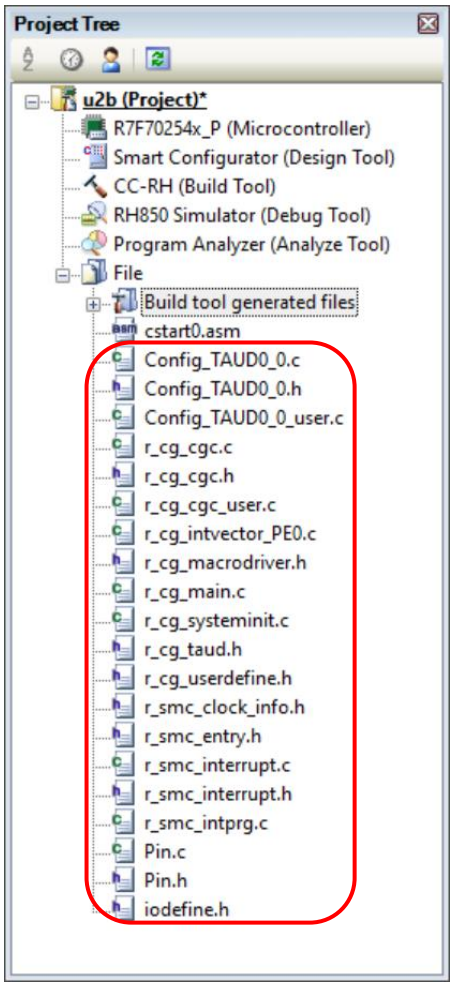

Figure 6-5. Add new files to CS+ project

Note: Because CS+ Smart Configurator communication plug-in does not support All Toolchain (CC-RH, GHS, IAR), it is necessary for the user to add or delete source files after configuration change in the Smart Configurator.

6.3 Configuration of Generated Files and File Names

Figure 6-6 Configuration of Generated Files and File Names shows the folders and files output by the Smart Configurator. Function *main()* is included in *main.c,* which is generated when the project is created by CS+.

“ConfigName” indicates the name of the configuration formed by the component settings.

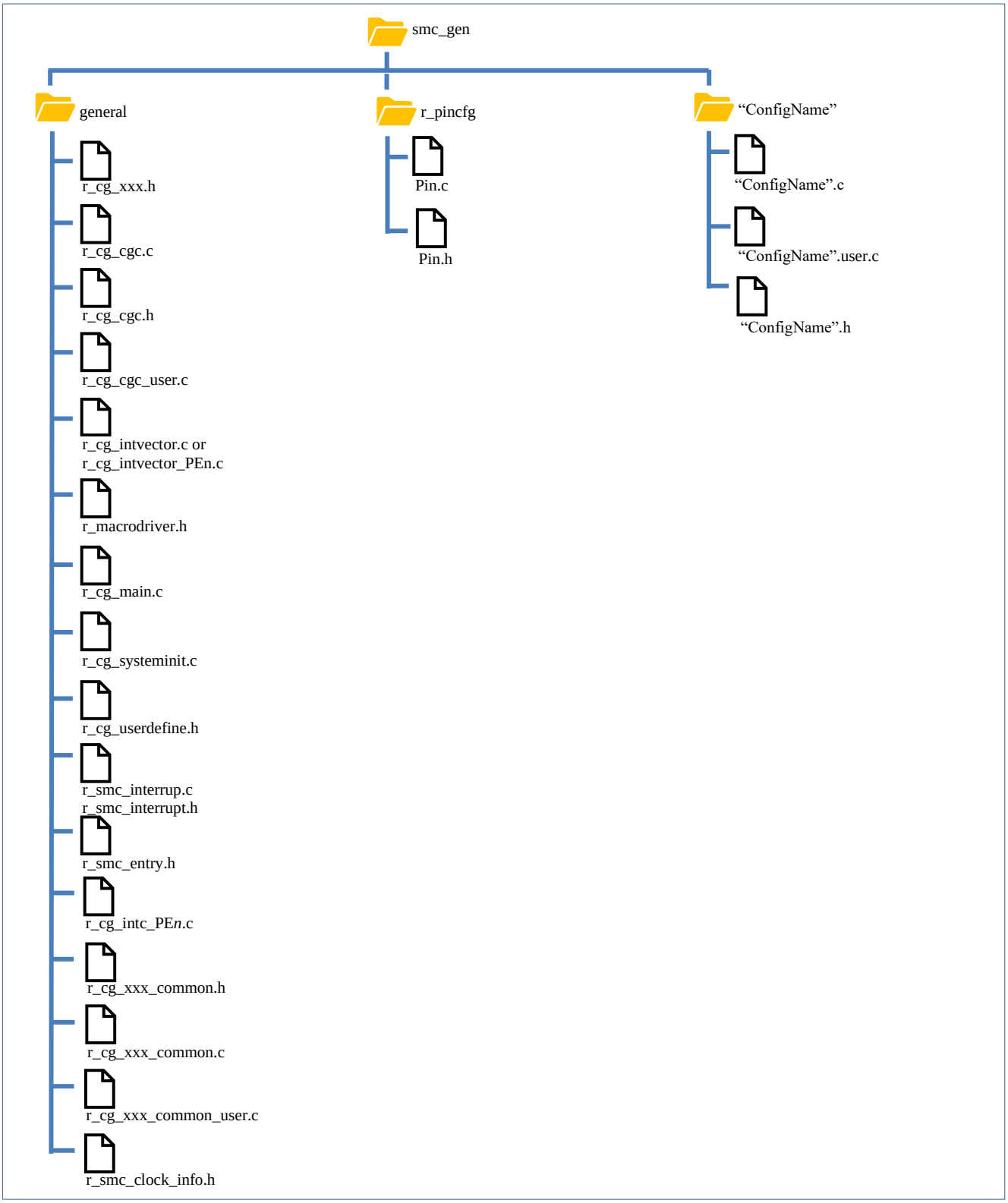

Figure 6-6 Configuration of Generated Files and File Names

Folder	File	Description
general	-	This folder is always generated. It contains header files and source files commonly used by Code Generator drivers of the same peripheral function.
	r_cg_xxx.h(Note*1)	These files are only generated for the used components. The files contain macro definitions for setting SFR registers.
	r_cg_cgc.c	This file is always generated. It contains the initialization of clock sources in accordance with the settings in the [Clocks] page.
	r_cg_cgc.h	This file is always generated. This header file contains macro definitions to initialize clocks.
	r_cg_cgc_user.c	This file contains functions to be added to *R_CGC_Create* after the CGC initialization. User can add codes and functions in the dedicated user code areas.
	r_cg_intvector.c *r_cg_intvector_PEn.c*	*r_cg_intvector.c* is generated only for: Smart Configurator for RH850/F1KM Smart Configurator for RH850/F1KH *r_cg_intvector_PEn.c* is generated only for: Smart Configurator for RH850/U2A(only PE*n*(n=0~3) which are chosen to be used, the r_cg_intvector_PE*n*.c(n=0~3) are generated.) Smart Configurator for RH850/C1M(r_cg_intvector_PE1.c is generated.) Smart Configurator for RH850/U2B(r_cg_intvector_PE0.c is generated. It contains interrupt vector table definitions.)
	r_cg_macrodriver.h	This file is always generated. This header file contains common macro definitions used in drivers.
	r_cg_main.c	This file is always generated. It defines the *main()* function.
	*r_cg_*systeminit.*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.
	r_cg_userdefine.h	This file is always generated. User can add macro definitions in the dedicated user code areas.
	r_smc_interrupt.c	This file is always generated.
	r_smc_interrupt.h	This file is always generated. It contains the priority level definition of all interrupts that are configured in the [Interrupts] tabbed page. User can use these macro definitions in application codes.

	r_smc_entry.h	This file is always generated. It contains the “include” clause which include: *“r_cg_xxx_common.h”* *"r_cg_macrodriver.h"* *"r_cg_userdefine.h"* *"r_cg_cgc.h"* *{ConfigName}.h* This file is included by file *“r_cg_main.c”*.
	r_cg_intc_PEn.c	This file is generated only for: RH850/U2A: only when PE PE*n*(n=0~3) is chosen to be used, the r_cg_intc_PE*n*.c(n=0~3) is generated. RH850/C1M: r_cg_intc_PE1.c is generated. RH850/U2B: r_cg_intc_PE0.c is generated. This file contains interrupt initialization API definitions.
	r_cg_xxx_common.c(Note*1)	This file is generated only for components which have some common settings shared by all resources of the component. Normally, it contains the shared API for multiple configurations and will be called by users.
	r_cg_xxx_common.h(Note*1)	This is header file for *r_cg_xxx_common.c and r_cg_xxx_common_user.c.* It is generated only for components which have some common settings shared by all resources of the component. Normally, it contains the shared API declaration for multiple configurations.
	r_cg_xxx_common_user.c(Note*1)	This file is generated only for components which have some common settings shared by all resources of the component. Normally, it contains the interrupt service routines for interrupts which are shared by multiple configurations. User can add codes and functions in the dedicated user code areas.
	r_smc_clock_info.h	This file is generated only for Smart Configurator for RH850/U2B. It contains macro definition for the clock source and module clock setting from [Clock] page. User can use the clock setting macro by including this file.
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,* Symbolic name definition, Symbolic name user guide and Symbolic name API.
{ConfigName}	-	This folder is generated for the Code Generator 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, for 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 sources in the [Clocks] page are generated to the macros in the r_cg_cgc.h file located in \src\smc_gen\general folder.

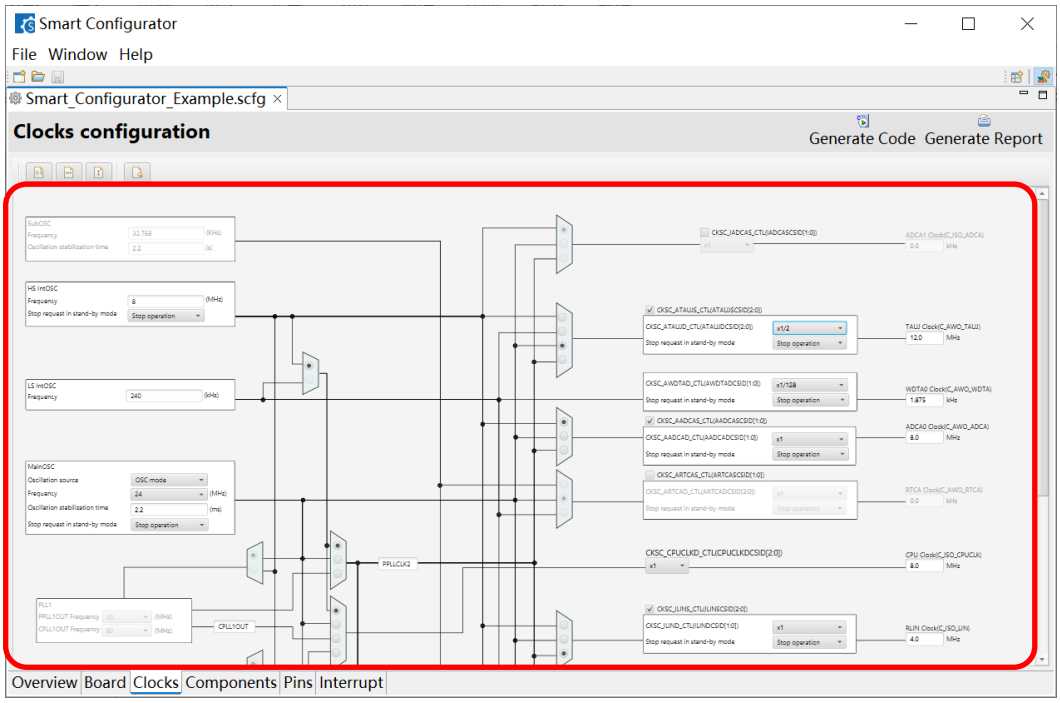

Figure 6-7 Clocks Configuration with Main Clock Selected as Clock Source

Folder	File	Macros/Functions	Description
general	*r_cg_cgc.c*	*R_CGC_Create*	This API function initializes clocks. *R_Systeminit* in *r_cg_systeminit.c* will call this function during execution of the *main()* function.
	r_cg_cgc.h	Macros related to clocks	These macros are for clock initialization in *R_CGC_Create*.
	r_cg_cgc_user.c	*R_CGC_Create_UserInit*	This API function is used to add code to *R_CGC_Create* after the CGC initialization.

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 this file *\src\smc_gen\{ConfigName}\{ConfigName}.c*.

Pin initialization codes will be handled in *main()*.

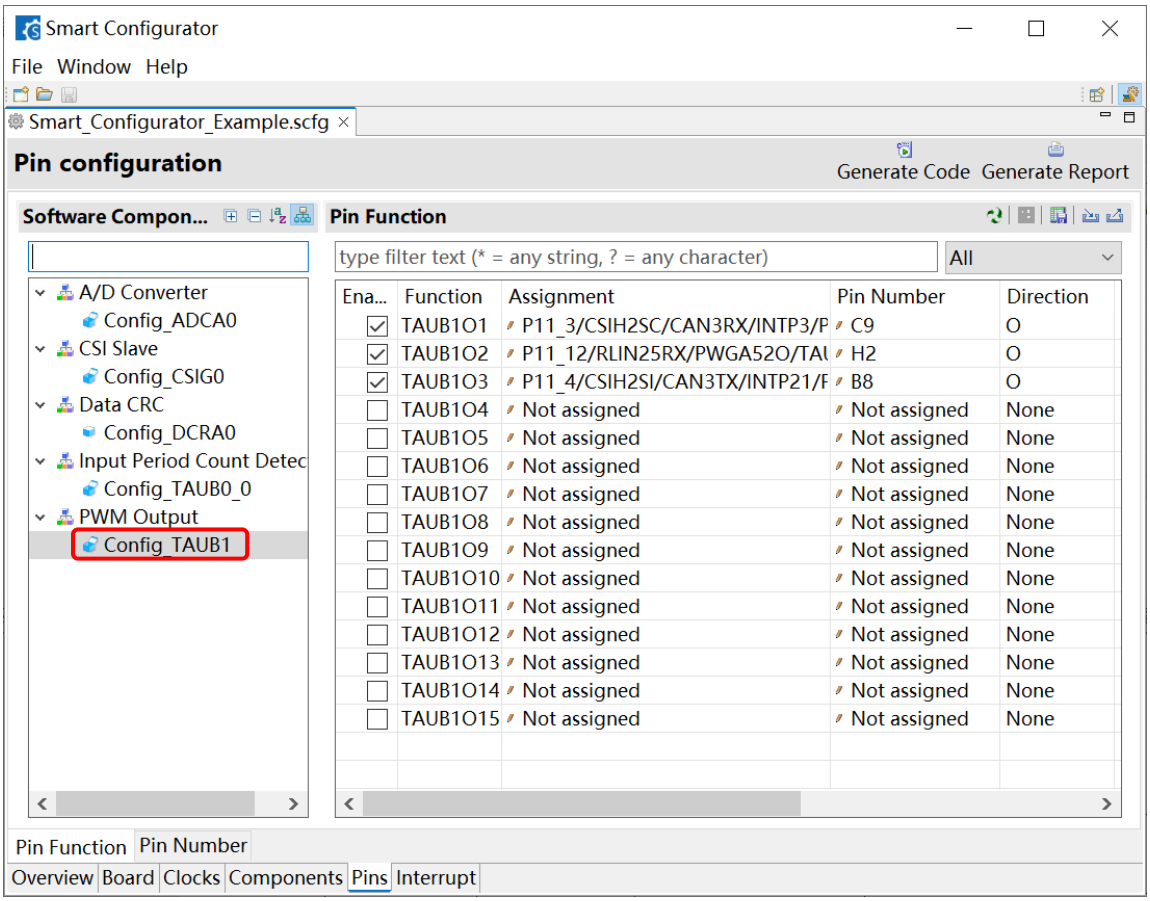

Figure 6-8 Pins Configuration for *Config_TAUB1*

Folder	File	Function	Description
{ConfigName}	*{ConfigName}.c*	*R_ConfigName_Create*	This API function initializes the pins used by this driver. *R_Systeminit* in *r_cg_systeminit.c* will call this function during execution of the *main()* function.

(2) 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	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.

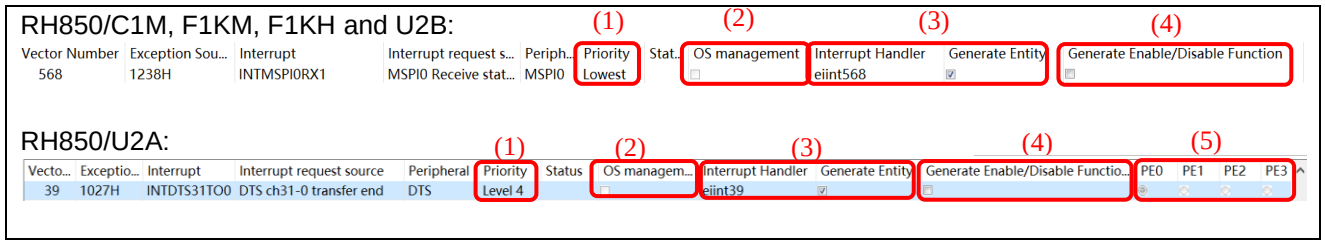

Figure 6-9 Interrupts Configuration in Interrupts View

RH850/C1M, F1KM, F1KH and U2B:

No	Item	Folder	File	Description
(1)	Priority	{ConfigName}	*{ConfigName}.c*	Interrupt priority level settings are initialized in *R_ConfigName_Create* in this file. *R_Systeminit* in *r_cg_systeminit.c* will call this function during execution of the *main()* function.
(2)	OS management	{ConfigName} or general	*{ConfigName}_user.c* *Or* *r_cg_xxx_common_user.c*	The interrupt functions defined in this file are output in the interrupt format that can be managed by the OS.
(3)	Interrupt Handler/Generate Entity	general	*r_smc_intprg.c* *r_cg_intvector_PE0.c*	The interrupt handler displayed on [Interrupt Handler] will be generated in file "*r_smc_intprg.c*" if [Generate Entity] is checked.
(4)	Generate Enable/Disable Function	general	*r_smc_interrupt.c* *r_smc_interrupt.h*	Interrupt enable/disable functions will be generated in *r_smc_interrupt.c* if [Generate Enable/Disable Function] is checked.

RH850/U2A:

No	Item	Folder	File	Description
(1)	Priority	general	*r_cg_intc_PEn.c*	Interrupt priority level settings are initialized in *R_Interrupt_Initialize_ForPE* in this file. *R_Systeminit* in *r_cg_systeminit.c* will call this function during execution of the *main()* function.
(2)	OS management	{ConfigName} or general	*{ConfigName}_user.c* *or* *r_cg_xxx_common_user.c*	The interrupt functions defined in this file are output in the interrupt format that can be managed by the OS.
(3)	Interrupt Handler/Generate Entity	general	*r_smc_intprg.c* *r_cg_intvector_PE0.c*	The interrupt handler displayed on [Interrupt Handler] will be generated in file "*r_smc_intprg.c*" if [Generate Entity] is checked.
(4)	Generate Enable/Disable Function	general	*r_smc_interrupt.c* *r_smc_interrupt.h*	Interrupt enable/disable functions will be generated in *r_smc_interrupt.c* if [Generate Enable/Disable] Function is checked.
(5)	PE*n* (the UI setting is only for RH850/U2A)	general	*r_cg_intc_PEn.c*	Interrupt binding is initialized in *R_Interrupt_Initialize_ForPE* in this file. *R_Systeminit* in *r_cg_systeminit.c* will call this function during execution of the *main()* function.

7. Creating User Programs

The Smart Configurator for RH850 only handles one component type: [Code Generator]. This chapter describes the method to add custom code for the Code Generator components.

7.1 Adding Custom Code in the Case of Code Generator

When creating configuration for Code Generator component, 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 “TAUB1” for the PWM output function (resource TAUB1). 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 TAUB1 (Config_TAUB1_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_TAUB1.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_TAUB1_Create_UserInit
* Description  : This function adds user code after initializing the TAUB1 channel
* Arguments    : None
* Return Value : None
***********************************************************************************/

void R_Config_TAUB1_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_TAUB1_channel0_interrupt
* Description  : This function is TAUB10 interrupt service routine
* Arguments    : None
* Return Value : None
***********************************************************************************/
#pragma interrupt r_Config_TAUB1_channel0_interrupt(enable=false, channel=256, fpu=true,
callt=false)
void r_Config_TAUB1_channel0_interrupt(void)
{
    /* Start user code for r_Config_TAUB1_channel0_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 */
```

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 the user works 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).

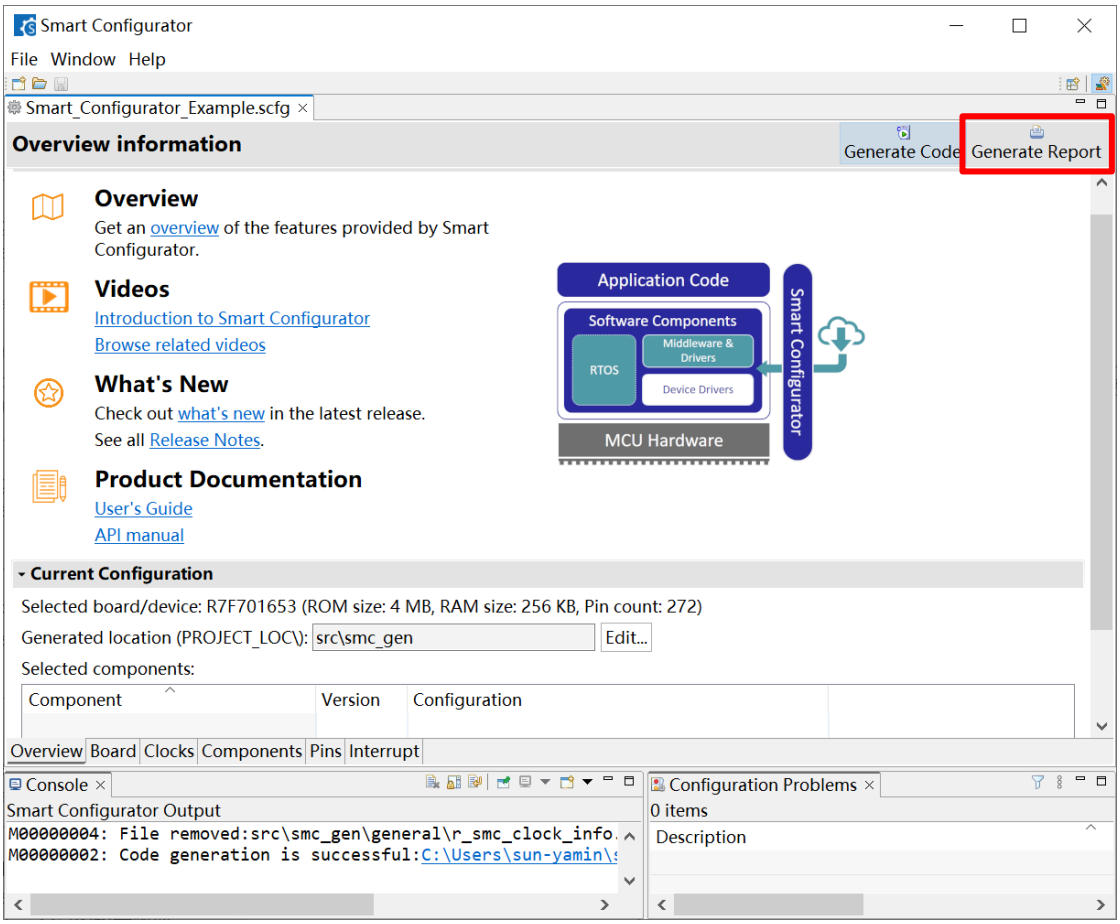

Figure 9-1 Output of a Report on the Configuration

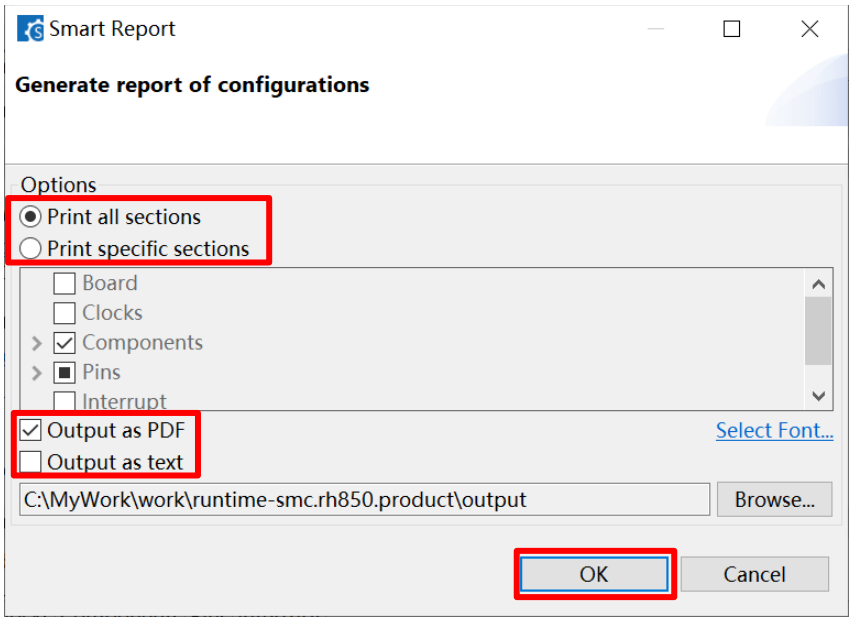

Figure 9-2 Dialog Box for Output of a Report (Example is selecting “Output as PDF”)

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.

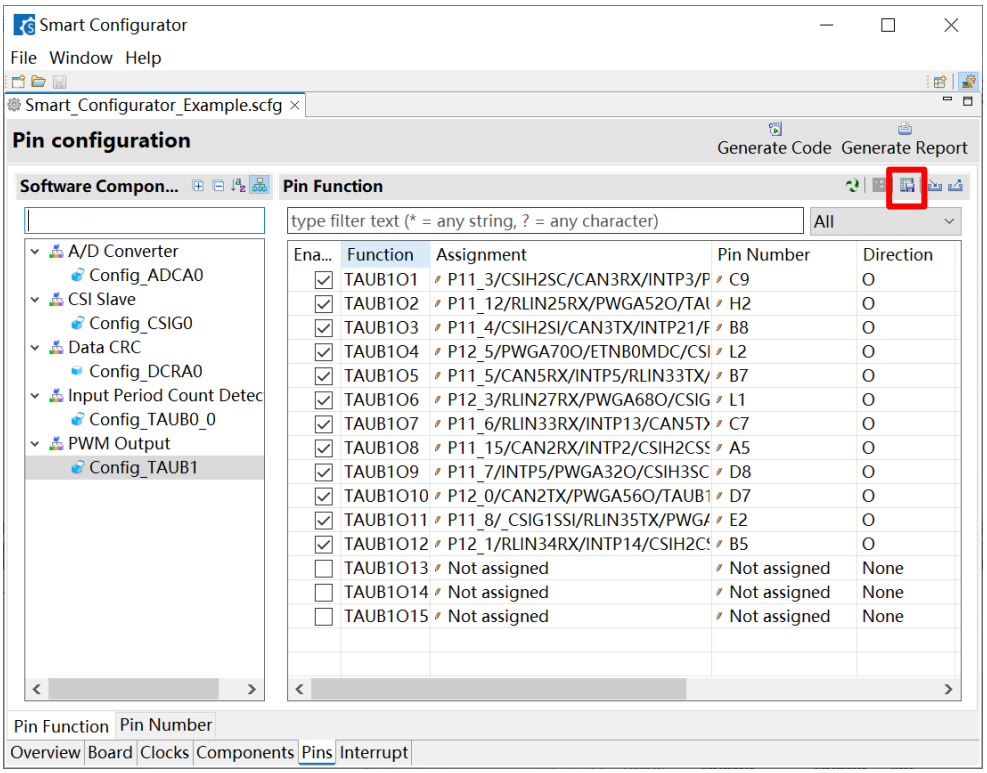

Figure 9-3 Output of a List of Pin Functions or Numbers (in csv Format)

10. User code protection feature

The Smart Configurator for RH850V1.9.0 and the later version incorporates the enhanced user code protection feature for Smart Configurator Code Generation component; from the Smart Configurator for RH850V1.10.0, the user code protection feature is extended to Clock generated file and Interrupt generated files. 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.

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 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.

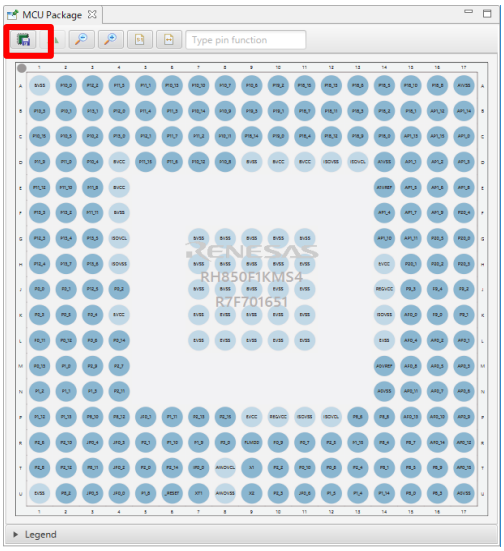

Figure 10-2 Outputting a Figure of MCU Package (in png Format)

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

Figure 10-3 shows an example of adding new user code into the Create API of PWM Output module by using the specific tags shown in Figure 10-1. After updating the configuration in the PWM output GUI and re-generating the codes, the inserted user codes will be automatically merged into the newly generated file.

```
void R_Config_TAUB1_Create(void)
{
    /* Disable channel counter operation */
    TAUB1.TT |= (_TAUB_CHANNEL1_COUNTER_STOP | _TAUB_CHANNEL0_COUNTER_STOP);
    /* Disable INTTAUB1I0 operation and clear request */
    INTC2.ICTAUB1I0.BIT.MKTAUB1I0 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB1I0.BIT.RFTAUB1I0 = _INT_REQUEST_NOT_OCCUR;
    /* Disable INTTAUB1I1 operation and clear request */
    INTC2.ICTAUB1I1.BIT.MKTAUB1I1 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB1I1.BIT.RFTAUB1I1 = _INT_REQUEST_NOT_OCCUR;
    /* Set INTTAUB1I0 setting */
    INTC2.ICTAUB1I0.BIT.TBTAUB1I0 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB1I0.UINT16 &= _INT_PRIORITY_LOWEST;
    /* Set INTTAUB1I1 setting */
    INTC2.ICTAUB1I1.BIT.TBTAUB1I1 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB1I1.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB1.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB1.TPS |= _TAUB_CK0_PRE_PCLK_15;
    /* Start user code */
    TAUB1.CMOR0 = 0x80;
    /* End user code */
    /* Set channel 0 setting */
    TAUB1.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_MASTER_CHANNEL | _TAUB_SOFTWARE_TRIGGER |
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE | _TAUB_START_INT_GENERATED;
    /* Set compare match register */
    TAUB1.CMUR0 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB1.CDR0 = _TAUB1_CHANNEL0_COMPARE_VALUE;
```

Inserted the user code with the specific tags

```
void R_Config_TAUB1_Create(void)
{
    /* Disable channel counter operation */
    TAUB1.TT |= (_TAUB_CHANNEL1_COUNTER_STOP | _TAUB_CHANNEL0_COUNTER_STOP);
    /* Disable INTTAUB1I0 operation and clear request */
    INTC2.ICTAUB1I0.BIT.MKTAUB1I0 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB1I0.BIT.RFTAUB1I0 = _INT_REQUEST_NOT_OCCUR;
    /* Disable INTTAUB1I1 operation and clear request */
    INTC2.ICTAUB1I1.BIT.MKTAUB1I1 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB1I1.BIT.RFTAUB1I1 = _INT_REQUEST_NOT_OCCUR;
    /* Set INTTAUB1I0 setting */
    INTC2.ICTAUB1I0.BIT.TBTAUB1I0 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB1I0.UINT16 &= _INT_PRIORITY_LOWEST;
    /* Set INTTAUB1I1 setting */
    INTC2.ICTAUB1I1.BIT.TBTAUB1I1 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB1I1.UINT16 &= _INT_PRIORITY_LEVEL7;
    TAUB1.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB1.TPS |= _TAUB_CK0_PRE_PCLK_1
    /* Start user code */
    TAUB1.CMOR0 = 0x80;
    /* End user code */
    /* Set channel 0 setting */
    TAUB1.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_MASTER_CHANNEL | _TAUB_SOFTWARE_TRIGGER |
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE | _TAUB_START_INT_GENERATED;
    /* Set compare match register */
    TAUB1.CMUR0 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB1.CDR0 = _TAUB1_CHANNEL0_COMPARE_VALUE;
```

User codes will automatically be merged into the newly generated

Figure 10-3 User code protection with auto merge

10.4 What to do when merge conflict occurs

10.4.1 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.

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

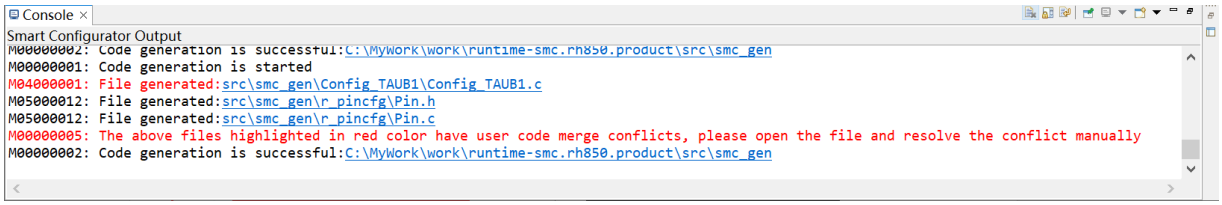

Figure 10-4 The merge conflict message outputted in the Smart Configurator console

User can click the conflicted file in the console message to open the File Compare view and then can resolve the conflict as next chapter 10.4.2, Steps for resolving the merge conflict described.

10.4.2 Steps for resolving the merge conflict

To resolve this merge conflict, User can follow the steps below to solve the merge conflicts.

1) Click on the conflicting file in the console to open the “File Compare” view (Figure 10-5 Code before resolving conflict).
2) Click on “Copy Current Change from Left to Right” (Figure 10-5 Code before resolving conflict).
3) Delete the codes that you do not want to use (Figure 10-6 Code after applying “Copy Current Change from Left to Right”).
4) Save the modified code (Figure 10-7 Code after deleting and saving).

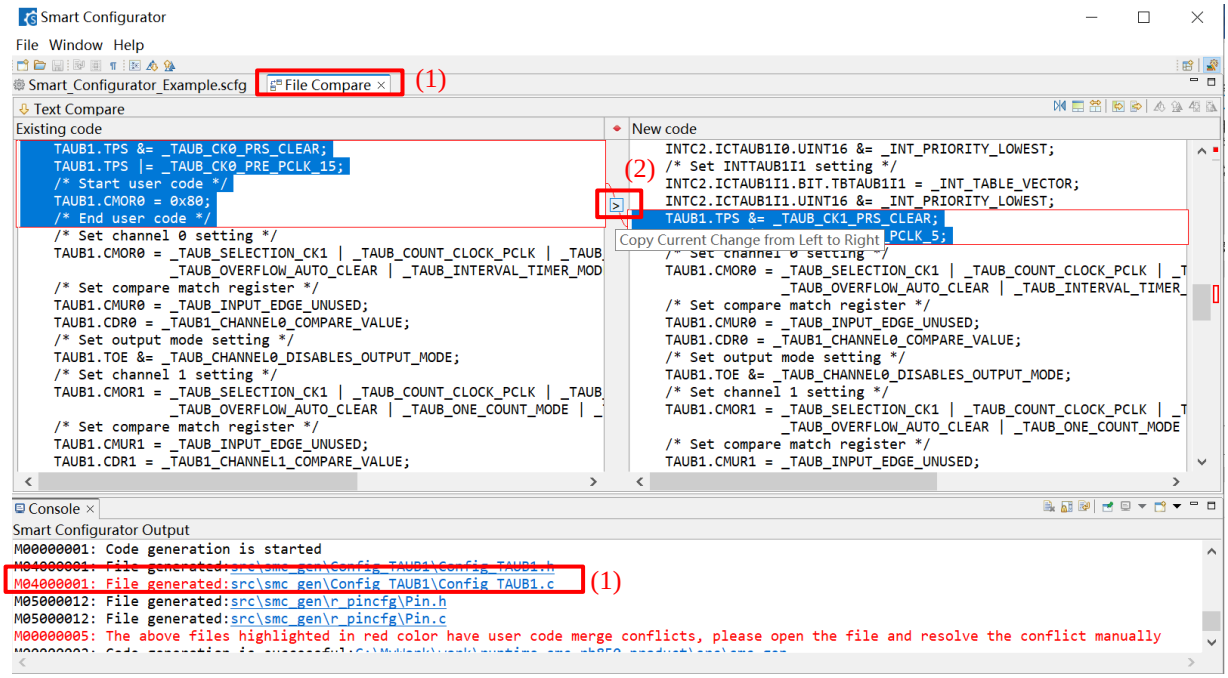

Figure 10-5 Code before resolving conflict

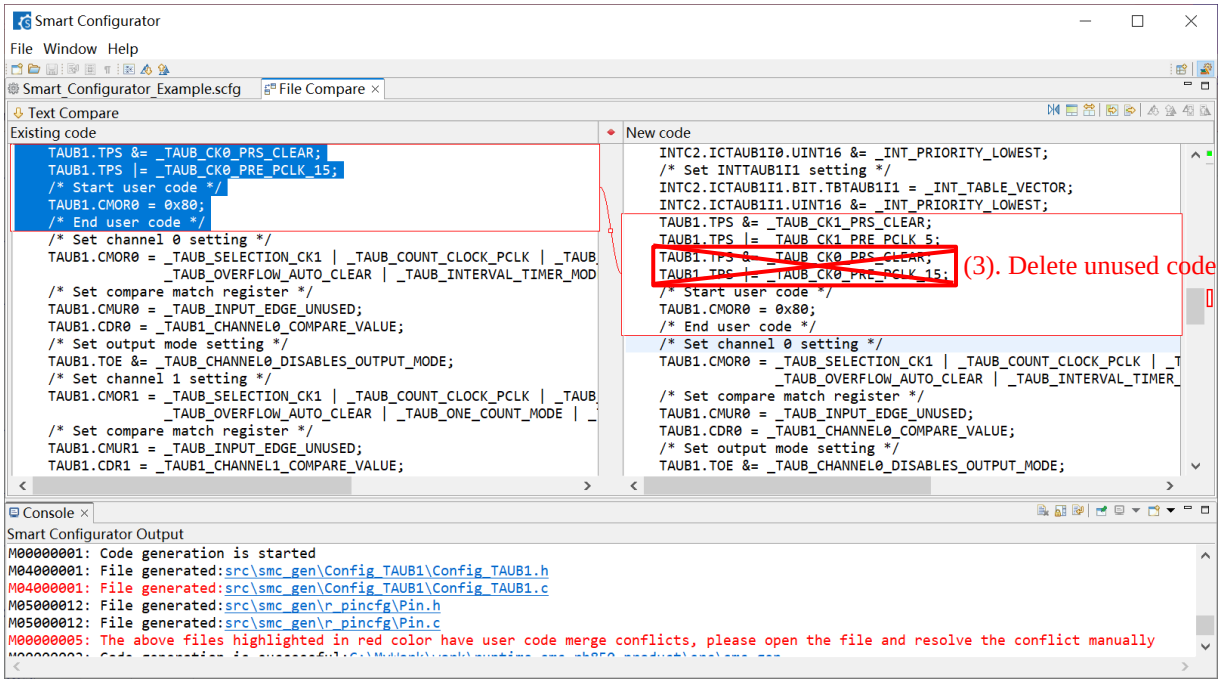

Figure 10-6 Code after applying “Copy Current Change from Left to Right”

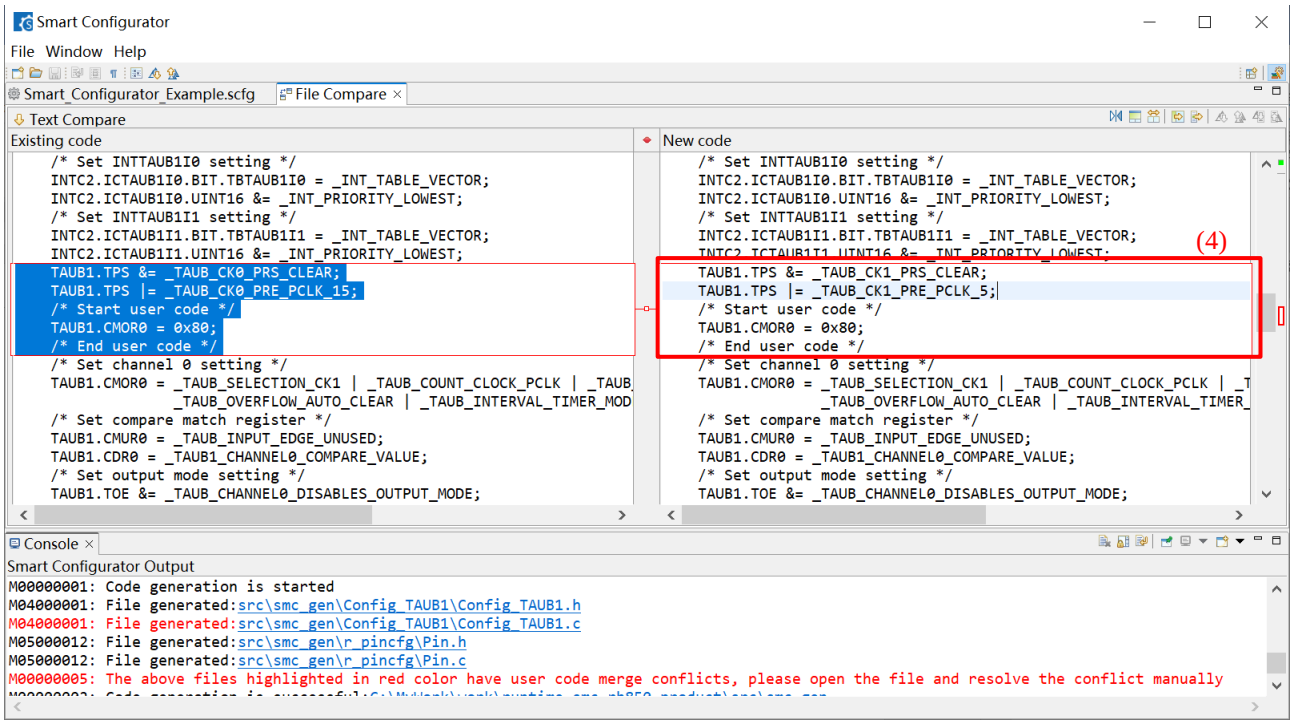

Figure 10-7 Code after deleting and saving

You can also resolve the confliction by editing the code in the right panel directly.

11. Help

11.1 Help

Refer to the help system for detailed information on the Smart Configurator.

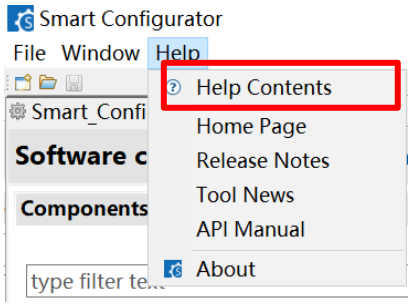

Figure 11-1 Help Menu

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

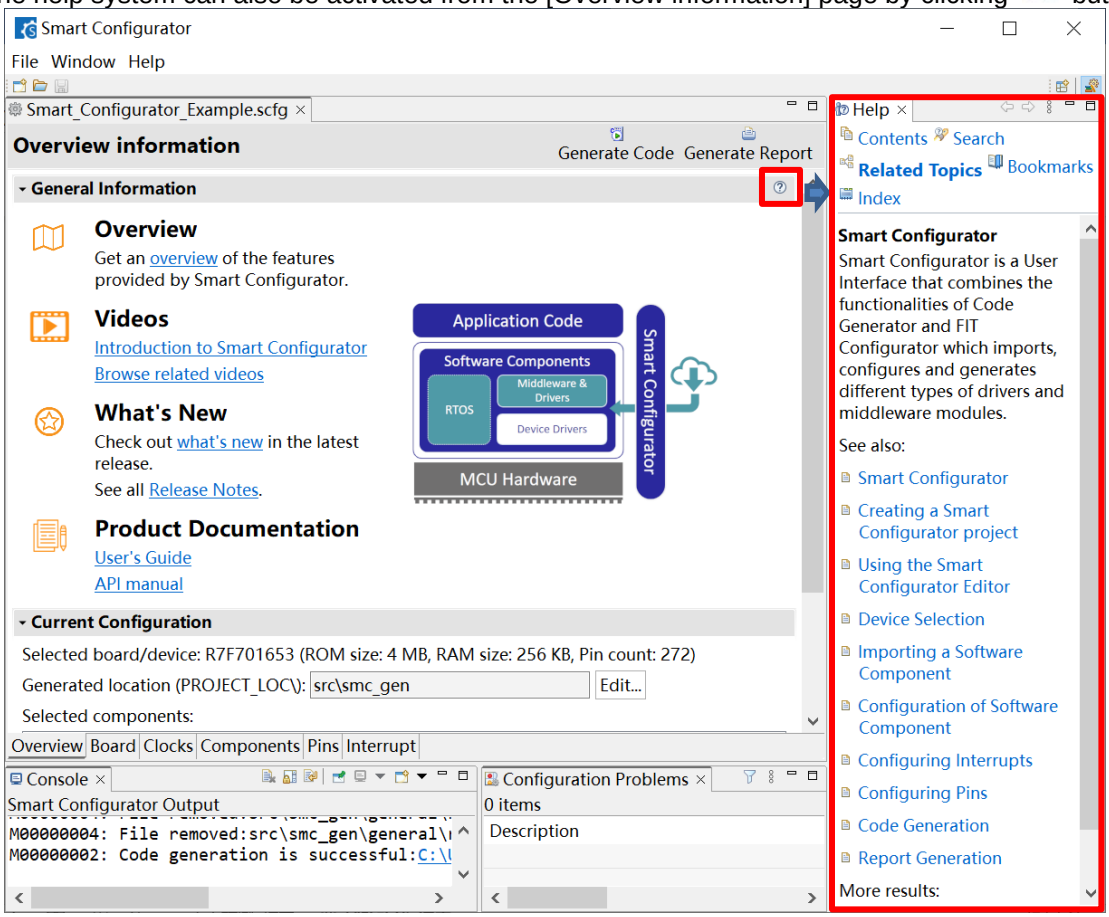

Figure 11-2 Smart Configurator Quick Start information

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

CS+ V8.11.00 Integrated Development Environment User’s Manual: Project Operation (R20UT5199)

CS+ V8.11.00 Integrated Development Environment User’s Manual: RH850 Debug Tool (R20UT5202)

CS+ V8.11.00 Integrated Development Environment User’s Manual: Message (R20UT5200)

CC-RH Compiler User’s Manual (R20UT3516)

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

Website and Support

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

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

Revision History

Rev.	Section	Description
1.00	All	First edition issued
1.10	Introduction	Update to CS+ (CS+ for CC) V8.10.00, RH850 Smart Configurator V1.9.0 and CS+ RH850 Smart Configurator Communication Plugins V1.10.00.
	All	All figures updated.
	2.5 Preparing Sample Projects	32 bit Environment, sample project path deleted. 64 bit environment, 4 new sample project path added.
	3.4.4 MCU/MPU Package view updated	Description and Figure updated.
	4.1.2 Selecting the board updated	Description and Figure updated.
	4.2 Clock Settings	Description and Figure updated.
	4.3 System Settings (only for RH850/U2A)	New added.
	4.4.3 Removing software component	Add description and figure about removing multiple components.
	4.4.6 Configure general setting of component	New added.
	4.5.3 Show pin number from pin functions	New added.
	4.5.6 Pin setting using board pin configuration information	New added.
	4.5.7 Pin filter feature	New added.
	4.5.8 Pin Errors/Warnings setting	New added.
	4.6.2 Changing the PEn setting	New added.
	4.6.3 Changing the interrupt handler name and Generate Entity setting new added	New added.
	5.2 Resolving pin conflicts	Moved to chapter 5. Managing Conflicts from chapter 4.5.2 Resolving pin conflicts.
	6.2 Configuration of Generated Files and File Names	Figure 6-3 Configuration of Generated Files and File Names and relative table content updated.
	6.5 Initializing Interrupts	RH850/U2A, RH850/U2B, RH850/C1M, RH850/F1KH interrupt description added.
	9.1 Report on All Configurations	PDF format file report added.
	10. User code protection feature for Smart Configurator Code Generation component	New added
	12. Documents for Reference updated	User manual is updated to the latest.
1.20	Introduction	Update to CS+ (CS+ for CC) V8.11.00, RH850 Smart Configurator V1.10.0 and CS+ RH850 Smart Configurator Communication Plugins V1.11.00.
	3.4 Window 9.1 Report on Configuration 11. Help	Update: Figure 3-3 Main Window Figure 3-4 Smart Configurator View Figure 9-1 Output of a Report on the Configuration Figure 11-2 Smart Configurator Quick Start information
	4.6.3 Changing the interrupt handler name, Generate Entity and Generate Enable/Disable Function setting	Interrupt handler edit function and Generate Entity function are extended to all RH850 devices. New function Generate Enable/Disable Function is added.
	6.5 Initializing Interrupts	Add interrupt handler and interrupt enable/disable function initialization.

	10. User code protection feature	User code protection feature is extended to Clock generated files and Interrupt generated files.
1.30	2.2 Installing the Smart Configurator	Add note for All Toolchain (CC-RH, GHS, IAR)
	4.6.3 Changing the Interrupt Handler Name, Generate Entity and Generate Enable/Disable Function Setting	Add detailed interrupt handler name rule
	6.2 Loading files generated by All Toolchain (CC-RH, GHS, IAR)	New added

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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.
- 2. Processing at Power-on
- The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.
- 3. Prohibition of Access to Reserved Addresses
- Access to reserved addresses is prohibited.
- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
- 4. Clock Signals
- After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.
- 5. Differences between Products
- Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
- The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the 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/.