RH850 Smart Configurator

User's Guide: IAREW, MULTI

R20AN0536EJ0104
Rev.1.04
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 importing its output files to IAR Embedded Workbench / GHS MULTI.

References to the Smart Configurator and Integrated Development Environment (IDE) in this application note apply to the following versions.

- IAR Embedded Workbench for RH850 V3.10
- GHS MULTI V8.14
- RH850 Smart Configurator V1.11.0

Target device and support compiler

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

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

Sample project

The sample project for RH850/F1KM is also included in the installation folder of Smart Configurator.

- Sample Project for IAR and IAR ICC
- Sample Project for MULTI and GHS CCRH850

Refer to the following instructions.

Smart Configurator: Guide on Sample Project for RH850/F1KM Device (R01AN4422)

Contents

1. Overview

1.1 Purpose

This application note describes the basic usage of the RH850 Smart Configurator (hereafter called the Smart Configurator), and the procedure for importing its output files to IAR Embedded Workbench / GHS MULTI.

Refer to the User's Manual of IAR Embedded Workbench / GHS MULTI for how to use them.

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. Installation and uninstallation

 This section describes the installation and uninstallation.

2.1   Installing the Smart Configurator

Download the Smart Configurator from the URL below.

  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.

2.2   Uninstalling the Smart Configurator

To uninstall the Smart Configurator, please select "Smart Configurator for RH850" from [Programs and Features] in the control panel.

2.3   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.

The sample project was installed in following location.

IAREW: "C:\Program Files (x86)\Renesas Electronics\SmartConfigurator\RH850\RH850F1KM_SampleProjects\SC_IAREW"

GHS: "C:\Program Files (x86)\Renesas Electronics\SmartConfigurator\RH850\RH850F1KM_SampleProjects\SC_MULTI"

Refer to the documents stored in the following location for how to use sample program.

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

3.   Operating the Smart Configurator

3.1   Procedure for Operations

Figure 3-1 Operating Procedure, shows the procedure for generating a source file using Smart Configurator and loading it into IAR Embedded Workbench. To use the sample project, refer to " Smart Configurator: Guide on Sample Project for RH850/F1KM Device (R01AN4422)". For the operation of IAR Embedded Workbench and GHS MULTI, refer to relevant document of IAR or GHS.

Operations in the Smart Configuration

```
Starting the Smart Configurator
        │ Refer to '3.2 Starting the Smart
        │ Configurator'
        ▼
Create and loading a configuration file
        │ Refer to '3.3 Create and Loading a
        │ Configuration '
        ▼
Setting of peripheral modules
        │ Refer to '4 Setting of Peripheral
        │ Modules'
        ▼
Setting pins
        │ Refer to '4.5 Pin settings'
        ▼
Setting of interrupts
        │ Refer to '4.6 Interrupt Settings'
        ▼
Generating source code
        │ Refer to '6 Generating Source Code'
        ┆
        ▼
Generating reports
Refer to '9 Generating Reports'
```

Operations in IAR Embedded Workbench, GHS MULTI

```
Loading generated files in IDE
        │ Refer to '7 Loading generated files in
        │ Integrated development environment'
        ▼
Creating the user program
        │ Refer to '8 Creating User Programs'
        ▼
Building
        ▼
Execution and debugging
```
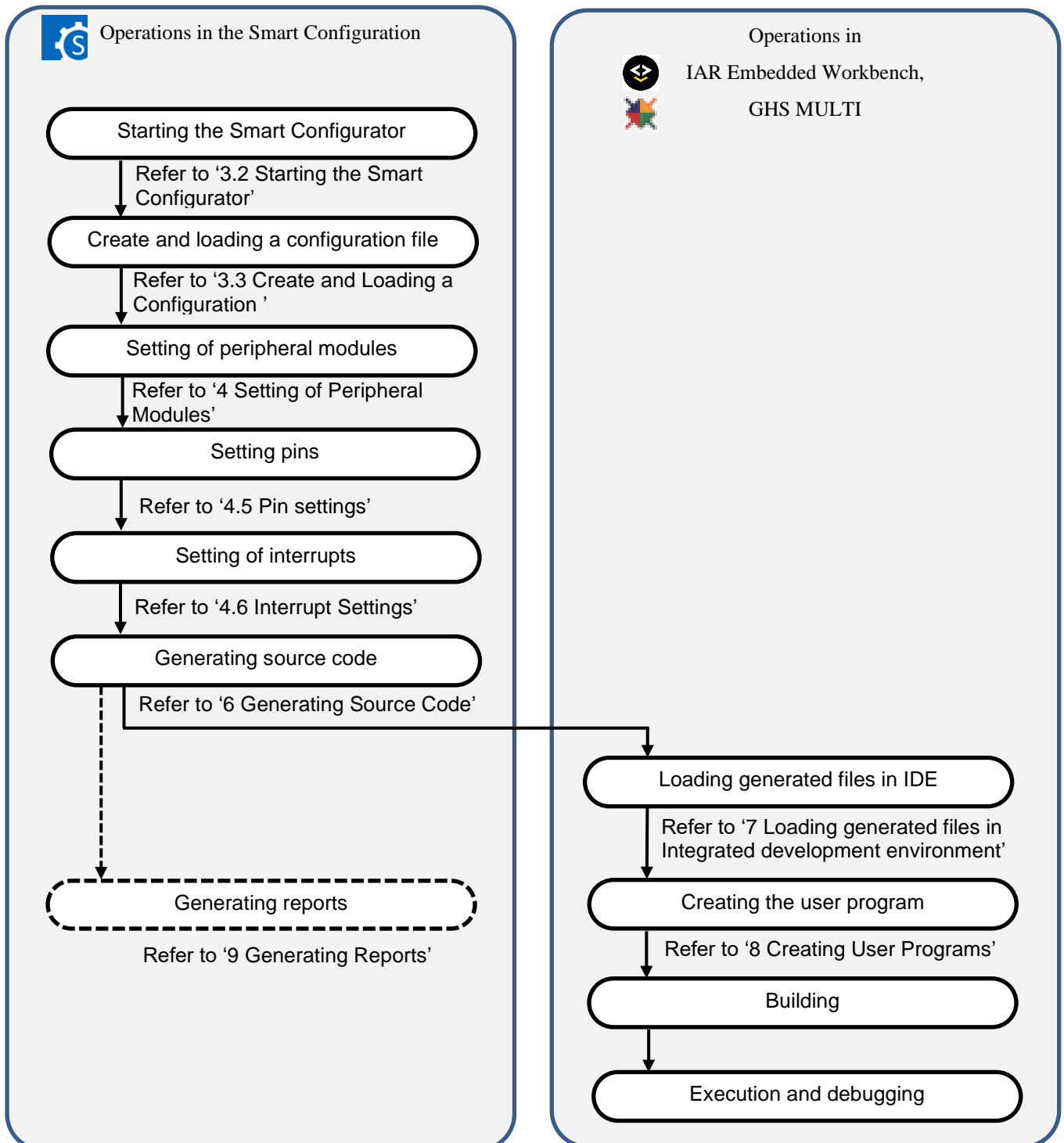
Figure 3-1 Operating Procedure

3.2   Starting the Smart Configurator

Select [Smart Configurator for RH850 Vx.x.x] of [Renesas Electronics Smart Configurator] from the Windows start menu. The main window of the Smart Configurator will be starting.

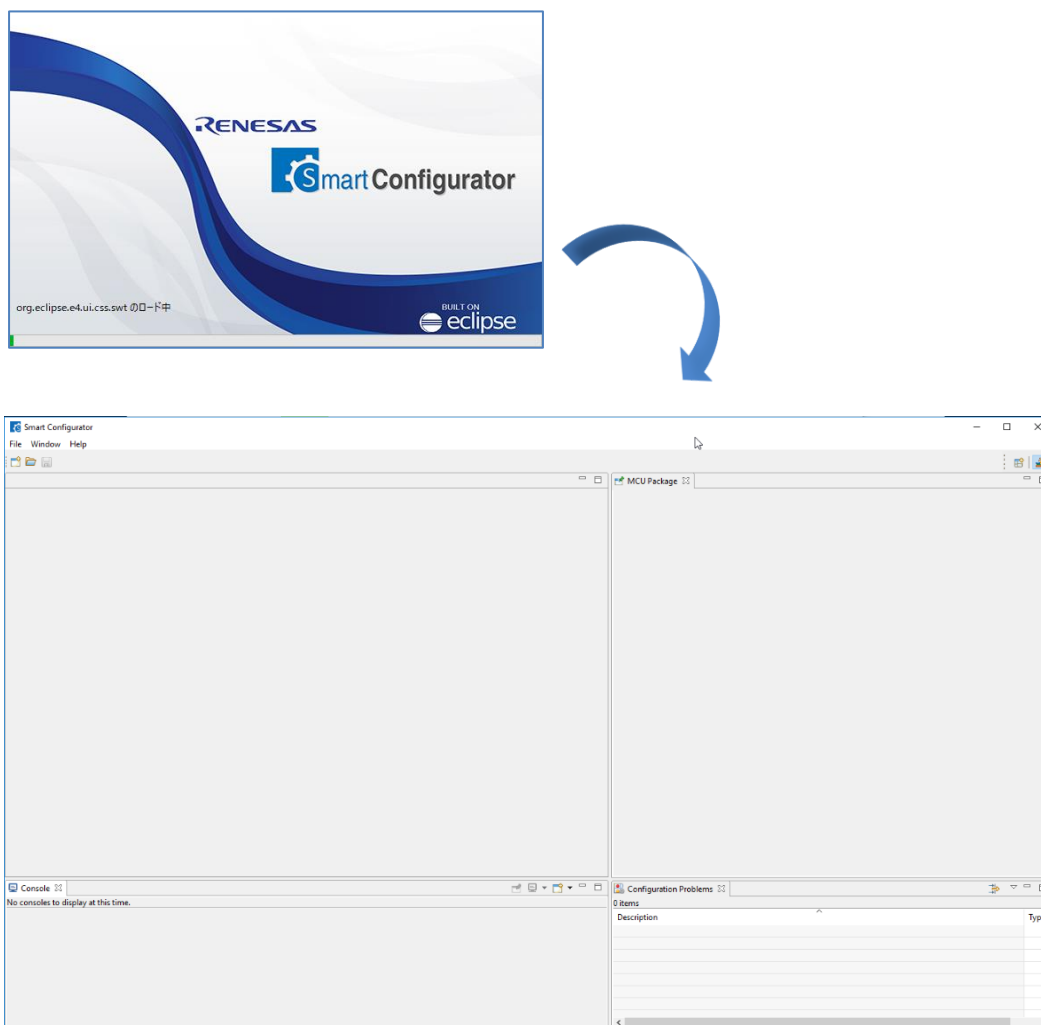
Note: Please replace Vx.x.x with your version.

Figure 3-2 Starting of Smart Configurator

3.3   Create and Loading a Configuration File

Smart Configurator saves and refers to the configuration file (*. scfg) the configuration information of the microcontroller, build tool, peripheral function, pin function etc. used in the project.

3.3.1   Creating a New Configuration File

On the main window, click the [New Configuration File] button to display the [New Smart Configuration File] dialog box.

(1)   In [Platform:], select the device.

(2)   In [Toolchain:], select the toolchain.

   To use the IAR compiler, select "IAR RH850 Toolchain".

   To use the GHS compiler, select "GHS RH850 Toolchain".

   To use the all complier, select "All Toolchain (CC-RH, GHS, IAR).
   For all compiler usage, please refer to 7.1.2 Loading files generated by All Toolchain (CC-RH, GHS, IAR) and 7.2.2 Loading files generated by All Toolchain (CC-RH, GHS, IAR)

(3)   In [File name:], enter the file name.

(4)   Confirm [Location:]. To change location, please click [Browse] and select the save destination.

   Note: The buildinfo.ipcf files for IAR Embedded Workbench, the default.gpj, project.gpj, sc_file.gpj files for MULTI will be generated to this location after clicking "Generate Code" button.

   The *.eww, *.ewp, *.ewd files for IAR Embedded Workbench will be generated to this location for the first time the configuration file (*. scfg) created.

(5)   To set the RTOS, click [Next]. When "IAR RH850 Toolchain" is selected, RTOS can't be set.

(6)   Click [Finish] to create the configuration file.
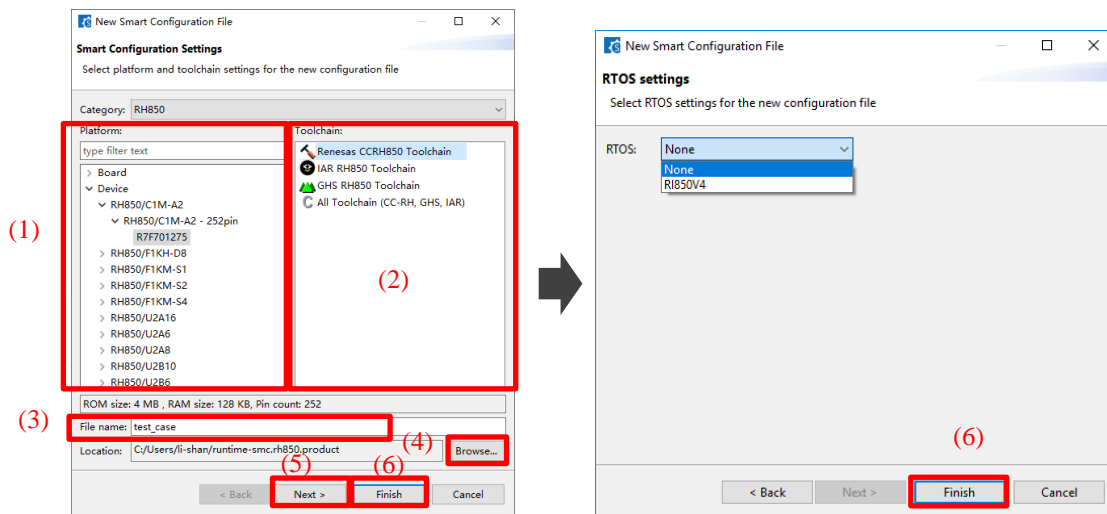
Figure 3-3 Create a Configuration File

(7)   Add driver component, configure the setting, generate code, and save the project.

   Note: The *.eww, *.ewp, *.ewd files will be generated only for the first-time the configuration file (*. scfg) created. While the buildinfo.ipcf file will be generated always for each time code generation.

When selecting [RI850V4] in the RTOS setting, user can select non-OS management interrupts / OS management interrupts for component interrupts. For details, refer to "4.6.1 Changing the Interrupt Priority Level and OS Management Setting".

3.3.2   Opening an Existing Configuration File

On the main window, click the 📂[Opening an Existing Configuration File] button to display the [Open] dialog box.
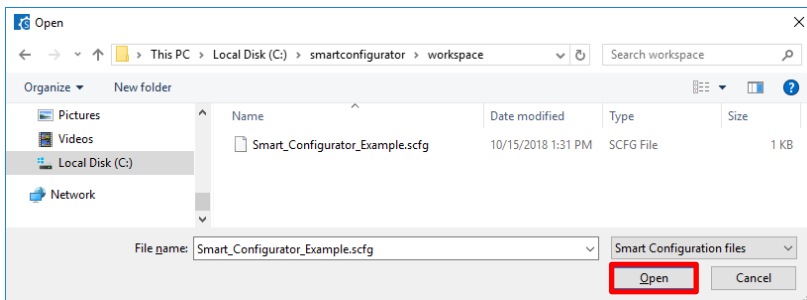
Figure 3-4 Opening an Existing Configuration File

3.4 Window

The main window is displayed when the Smart Configurator is started. The configuration of the window is shown in Figure 3-5, Main Window.

Figure 3-5 Main Window

(1) Menu bar

(2) Main tool bar

(3) Smart Configurator view

(4) MCU/MPU Package view

(5) Console view

(6) Configuration Problems view

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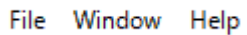
Table 3-1, Main Menu Items, lists the items of the main menu.

Table 3-1　Main Menu Items

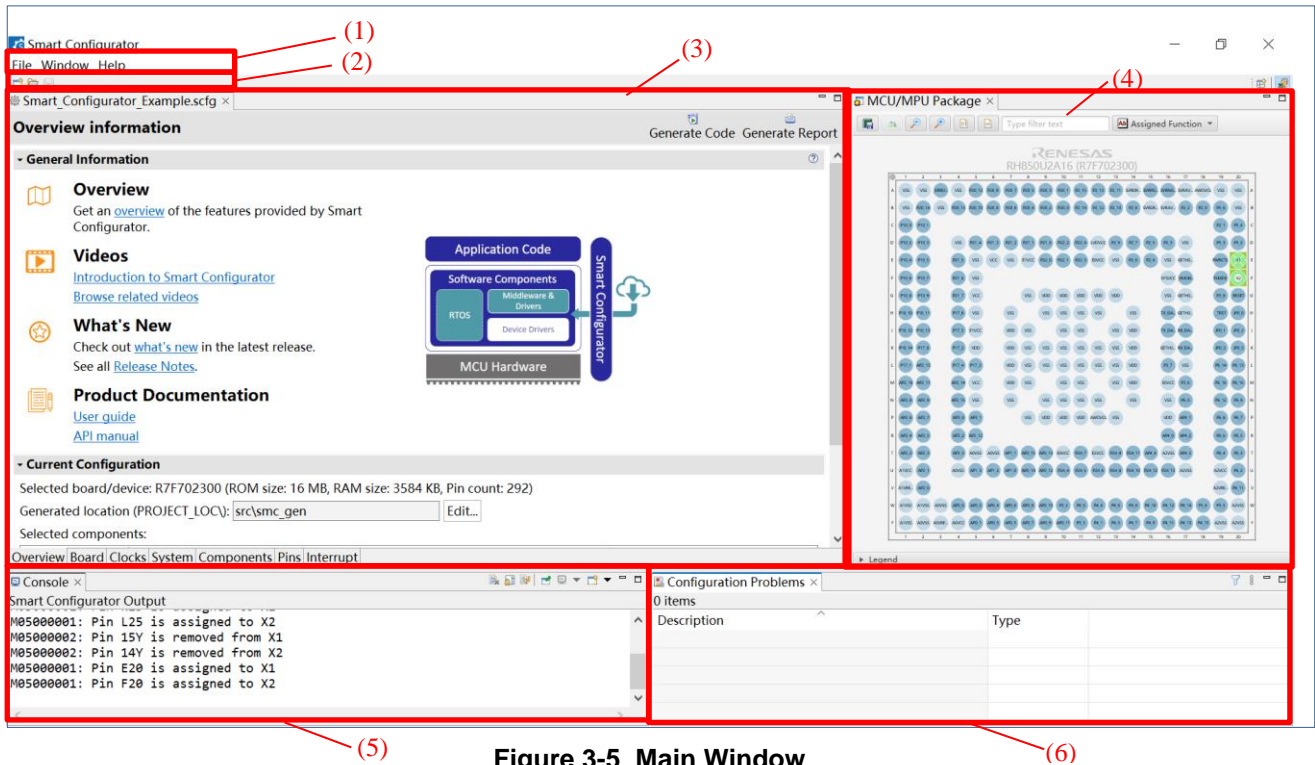| Menu		Details
File	New	The dialog box [New Smart Configuration File], which is used to create a new configuration file, is displayed.
	Open	The dialog box [Open], which opens an existing configuration file, is displayed.
	Save	Saves a configuration file with the same name.
	Restart	Smart Configurator is re-started.
	Exit	Execution of the Smart Configurator is terminated.
Window	Preference	The dialog box [Preference], which is used to specify the properties of the configuration file, 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 and navigate to Smart Configurator home page in web browser.
	Release Notes	Open and navigate to Smart Configurator release notes in web browser.
	Tool News	Open and navigate to Smart Configurator tool news in web browser.
	API Manual	Open and navigate to Smart Configurator API manual in web browser.
	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 Smart Configuration File]
	[File] → [Open]
	[File] → [Save]

3.4.3   Smart Configurator View

The Smart Configurator view consists of seven pages: [Overview information], [Board], [Clocks], [System], [Components], [Pins], and [Interrupts]. Select a page by clicking on a tab; the displayed page will be changed.
[System] page is only supported by RH850/U2A.
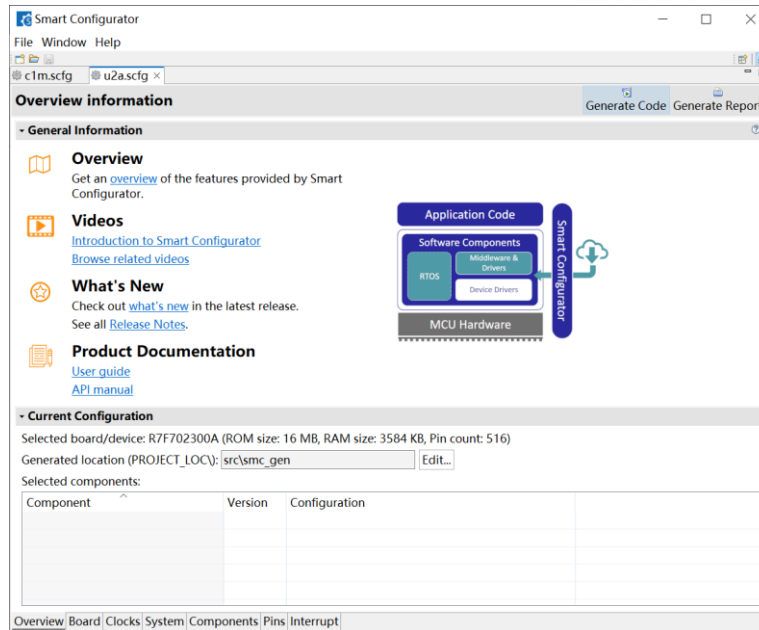
Figure 3-6 Smart Configurator View

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] and [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 "chapter 4.1.2 Selecting the Board" and "chapter 0

Pin setting using board pin configuration information").

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

Symbolic Name feature is not applied to APORT, JPORT and IPORT.
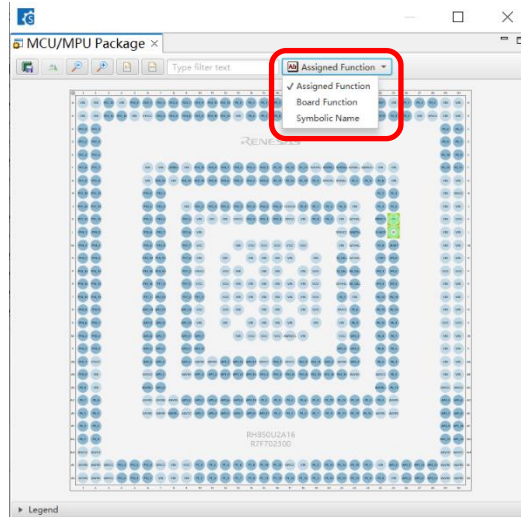
Figure 3-7 MCU/MPU Package View

3.4.5   Console View

The console displays details of changes to the configuration made in the Smart Configurator or MCU/MPU Package view.
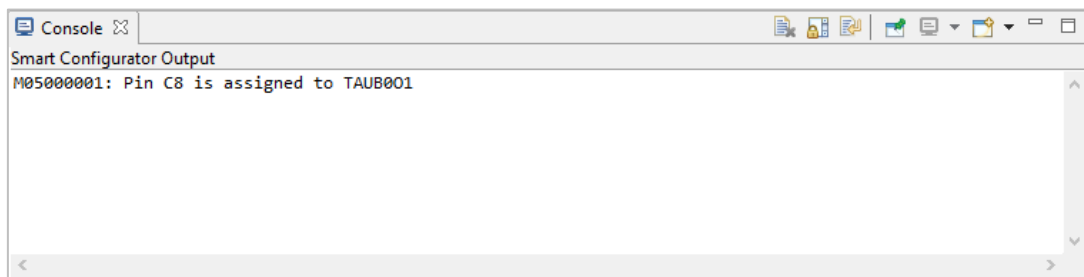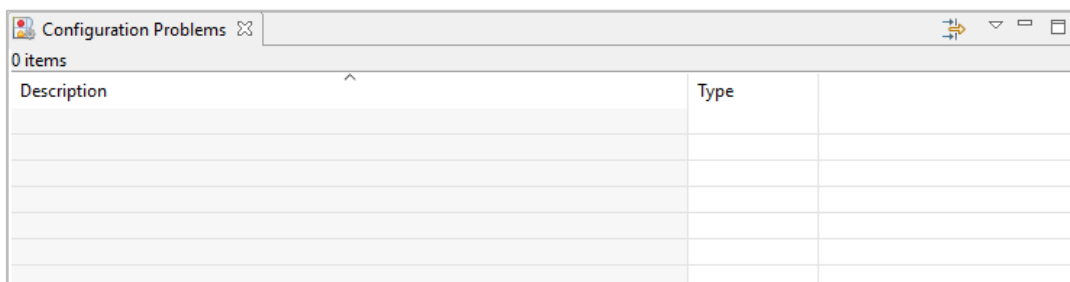
Figure 3-8 Console View

3.4.6 Configuration Problems View

The Configuration Problems view displays problems with peripheral functions, interrupts, and pin conflicts.

Figure 3-9 Configuration Problems View

4.   Setting of Peripheral Modules

User can select peripheral modules from the Smart Configurator view.

4.1   Board Setting

On the [Board] page, user can select boards and change devices.

4.1.1   Selecting the Device

Click on the [...] button to select a device.
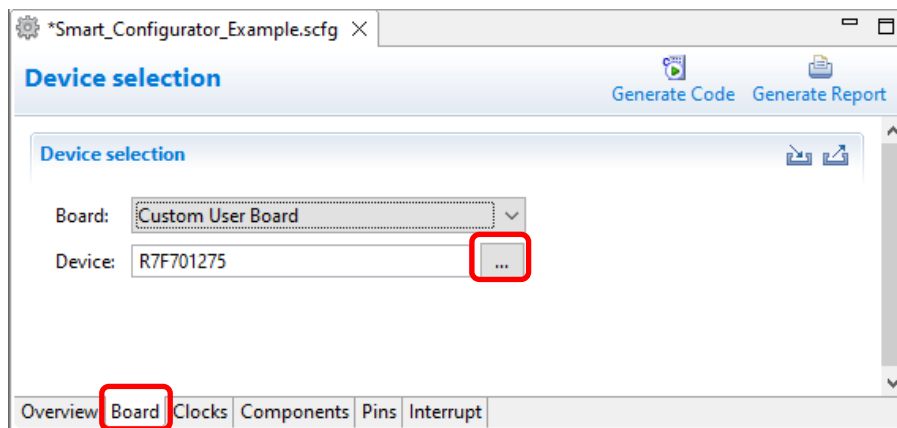
Figure 4-1 Selecting the Device

Note: Device change is not reflected in the device (micro controller) of IAR project and GHS project.

The following message is displayed when changing the device. For each button operation, refer to "Table 4-1, Device Change Confirmation Operation List".

Changing device after saving:
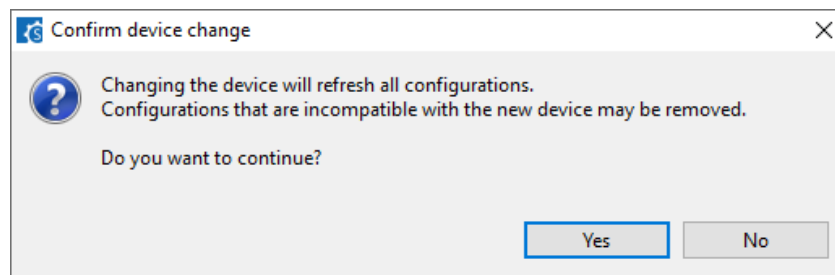
Figure 4-2 Confirm Device Change after saving

Changing device before saving
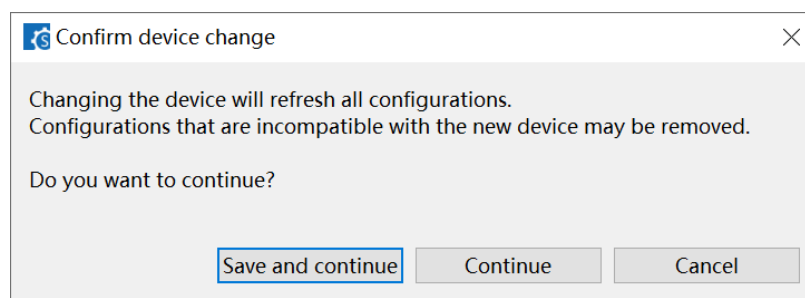
Figure 4-3 Confirm Device Change before saving

Table 4-1 Device Change Confirmation Operation List

Button	Operation explanation
Yes	Change to the selected device.
No	It does not change the device.
Save and continue	After saving the current configuration contents to the configuration file, change to the selected device.
Continue	Changes to the selected device without saving the current configuration contents to the configuration file.
Cancel	It does not change the device.

4.1.2   Selecting the Board

Click on the [^] to select a board from the list. When peripheral functions are configured by board selection, pins are automatically set according to board connection.
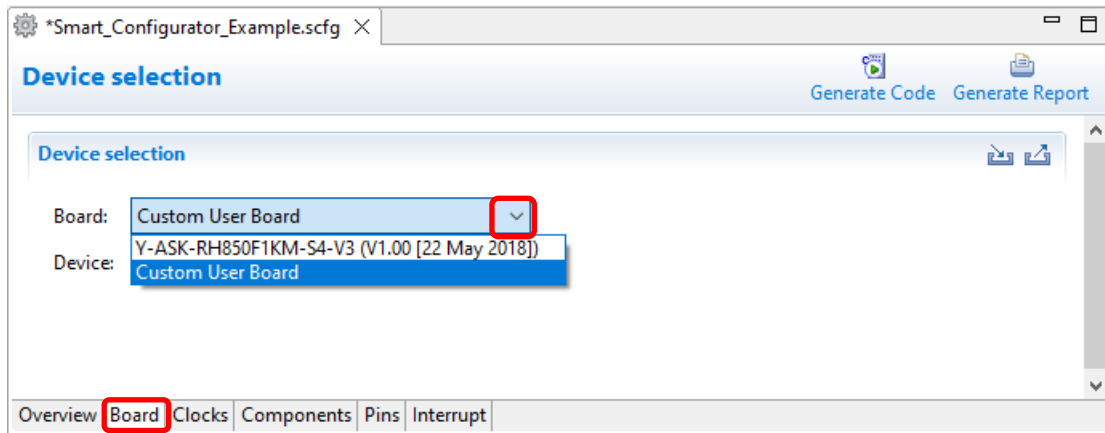
Figure 4-4 Selecting the Board

The following items are changed according to the configuration of the selected board.

- Pin assignment
- Frequency of the main clock
- Frequency of the sub-clock
- Target device

If user change the board, the message shown in "Figure 4-2" or the following message will be displayed. For each button operation, refer to "Table 4-2, Board Change Confirmation Operation List".
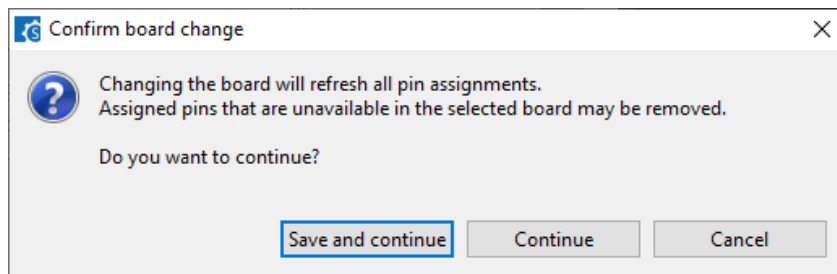
Figure 4-5 Confirm Board Change

Table 4-2 Board Change Confirmation Operation List

Button	Operation explanation
Save and continue	After saving the current configuration contents to the configuration file, change to the selected device.
Continue	Changes to the selected device without saving the current configuration contents to the configuration file.
Cancel	It does not change the device.

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

4.1.3 Export Board Configuration

The current main clock frequency, sub clock frequency and pin assignment settings can be exported as board configuration. Follow the procedure below to export the board configuration.

(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.
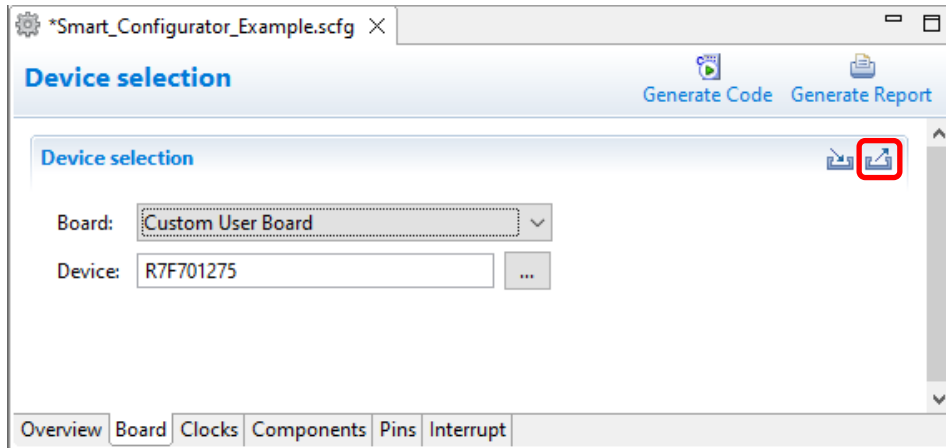
Figure 4-6 Export of Board Configuration (bdf format)

4.1.4 Import Board Configuration

The board setting is defined in bdf (Board Description File). Follow the procedure below to import board configuration.

(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.
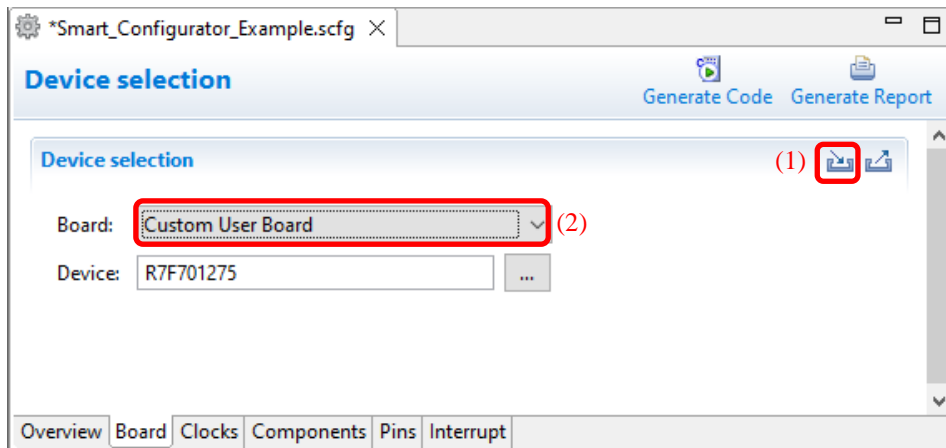
Figure 4-7 Import of Board Configuration (bdf format)

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

On the [Clocks] page, set the clock. The [Clocks] page setting is used as the clock source for each component. Set the clock before configuring the component.

The clocks setting is performed in the following procedure.

  (1)   Set the clock oscillator circuit.

  (2)   Sets the clock source to be supplied to the CPU and peripheral functions.

     (a) When you move the mouse on the screen, the clock signal is displayed in blue.

     (b) Click on the screen to select the clock selector.
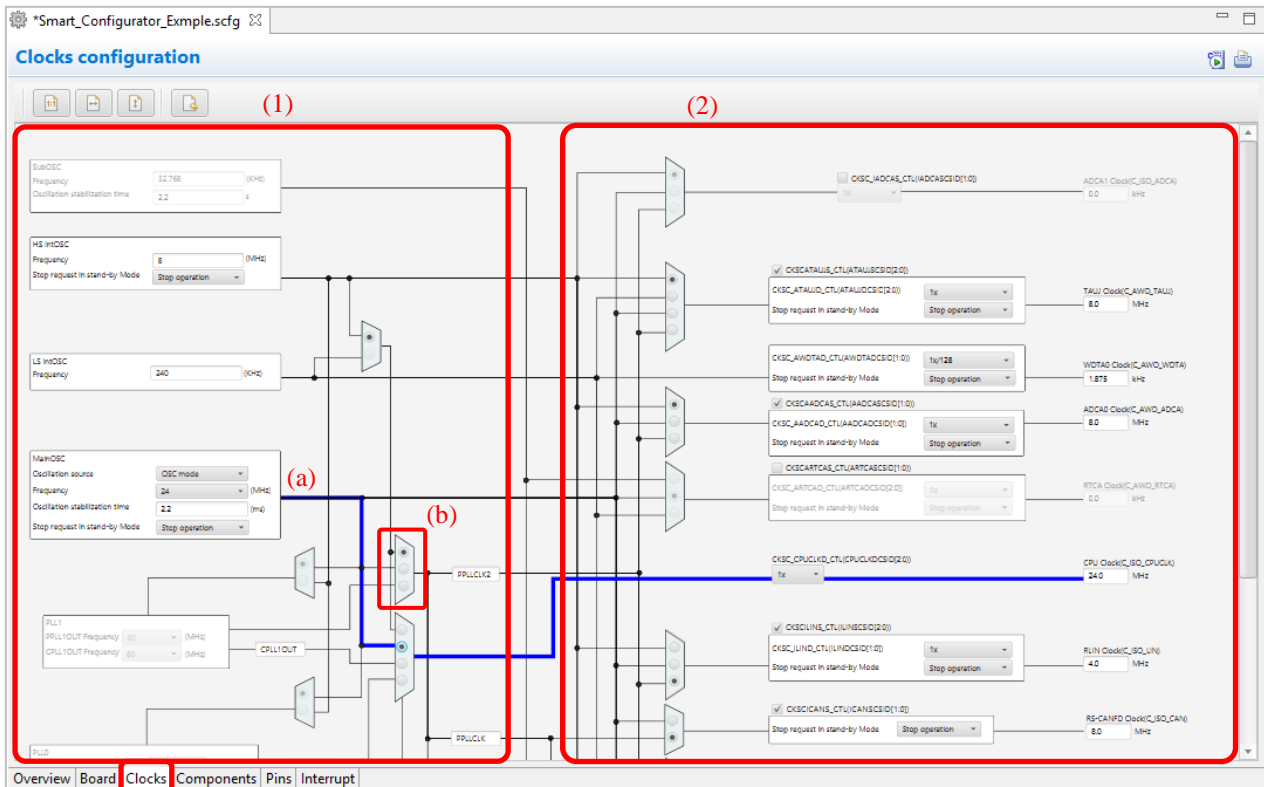
Figure 4-8   Clock Settings

4.3 System Settings (RH850/U2A only)

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.
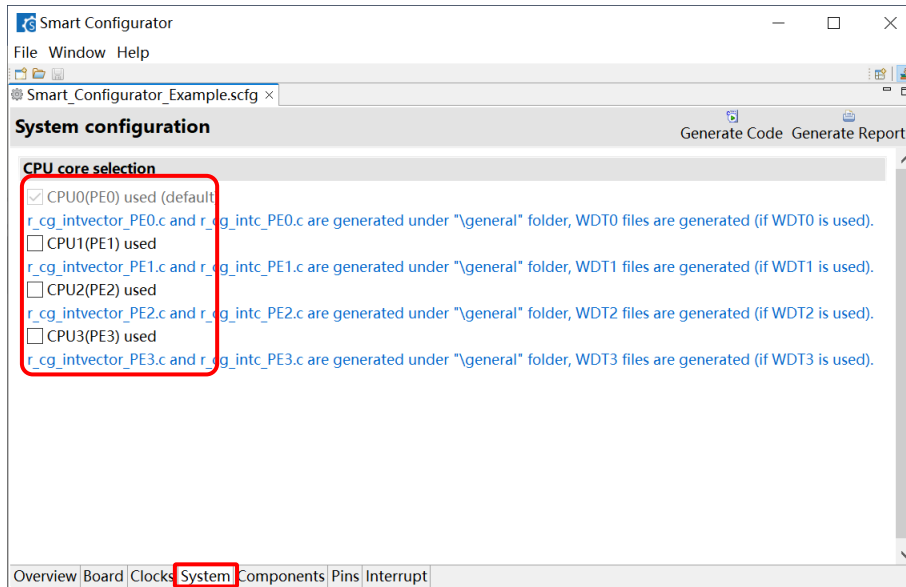
Figure 4-9 [System] Page

4.4   Component Settings

Drivers can be combined as software components on the [Components] page. Added components are displayed in the component tree at the left of the page.
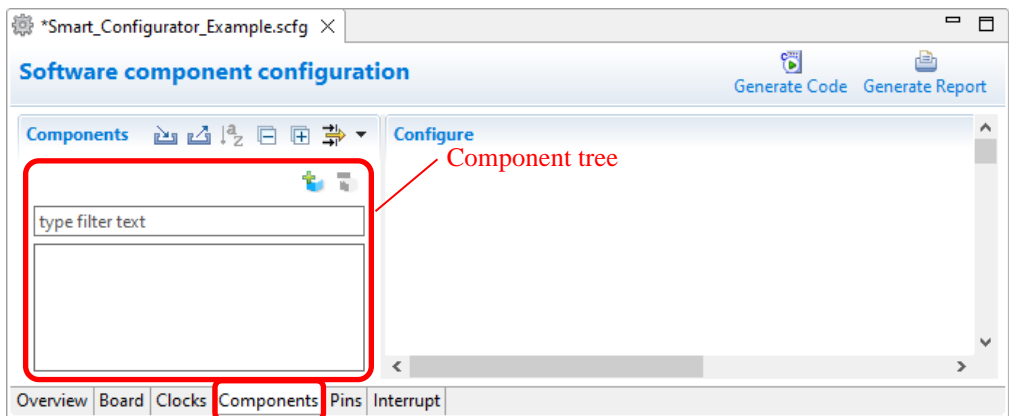
Figure 4-10 Component Page

4.4.1 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.
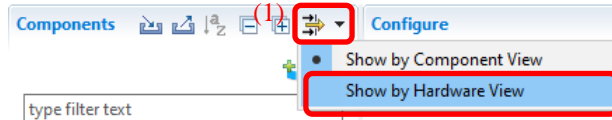
Figure 4-11 Switch to [Show by Hardware View]

(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) in the [Software Component Selection] page.

(4) Click the [Next].
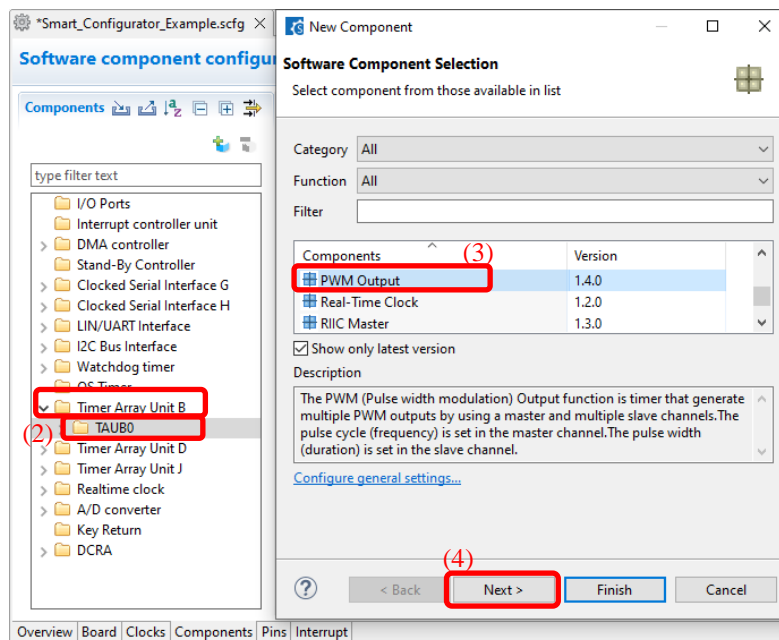
Figure 4-12 Adding CG Components form the Hardware View

(5) Specify an appropriate configuration name in the [Add new configuration for selected component] page or use the default name (for e.g., Config_TAUB1).

(6) Select a hardware resource or use the default resource (for e.g., TAUB1).

(7) Click on [Finish]. The component is added to the component tree.
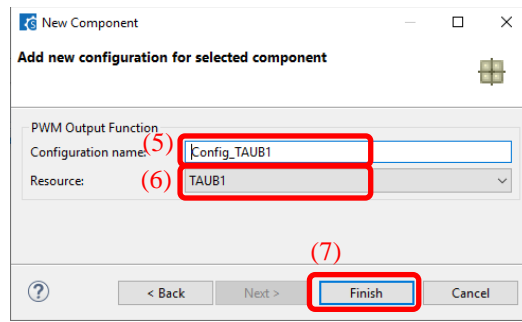
Figure 4-13 Add New Configuration for Selected Component (for e.g.,TAUB1)

4.4.2   Adding Component

The following describes the procedure for adding a component.

(1)   Click on the [(Add component)] icon.
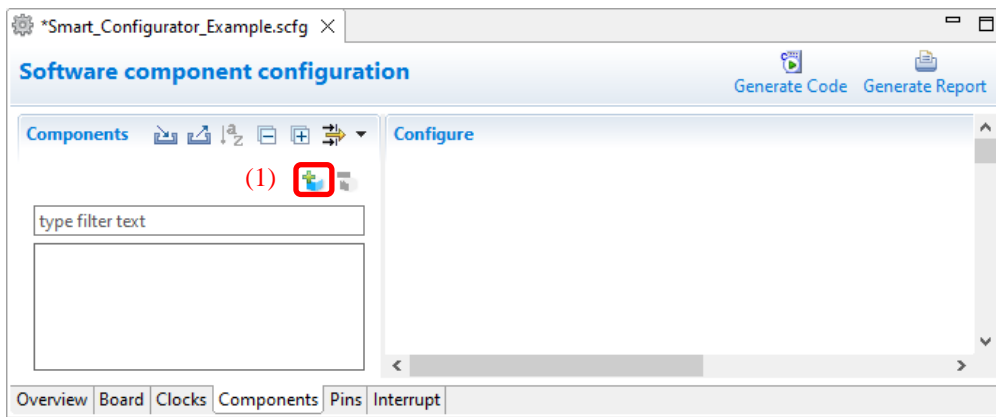
Figure 4-14 Adding Components

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

(3)   Click on [Next].
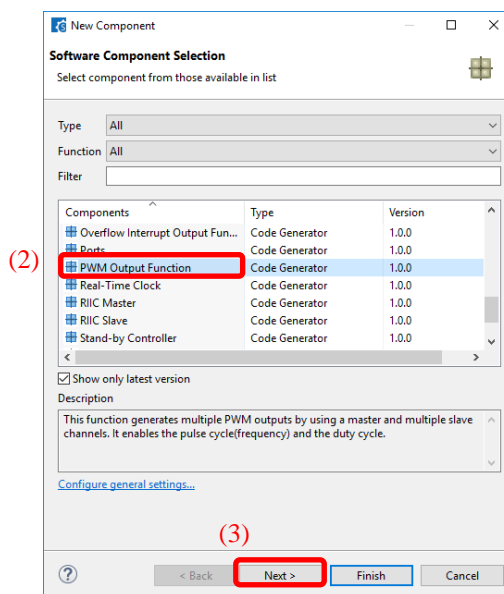
Figure 4-15 Selection of Software Components

(4) Specify an appropriate configuration name in the [Add new configuration for selected component] page 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]. The component is added to the component tree.
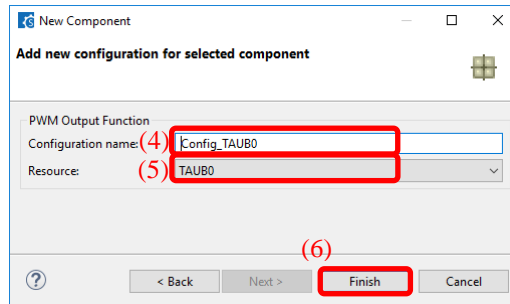
Figure 4-16 Add New Configuration for Selected Component (for e.g., TAUB0)

4.4.3   Removing 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.
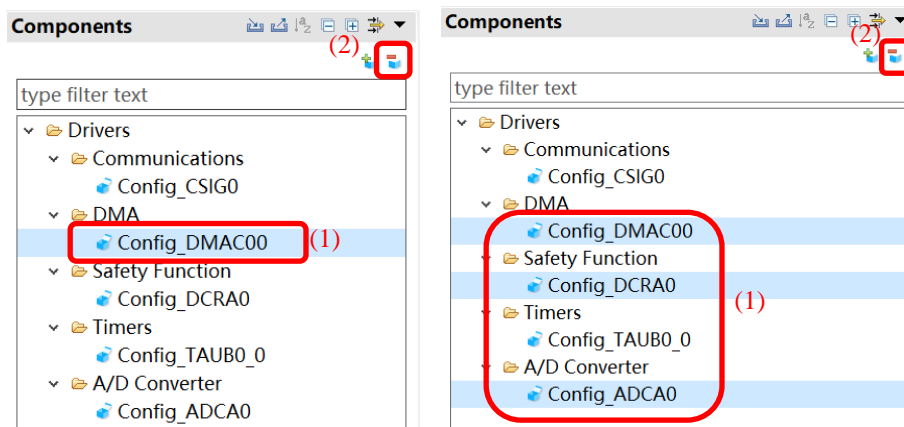
Figure 4-17 Removing a Component or Multiple Components

4.4.4   Setting a Code Generator Component

Follow the procedure below to set the component configuration.

(1)   Click the component in the component tree (for e.g., Config_TAUB0).

(2)   Configure the driver in the [Configure] panel to the right of the Components tree. The Figure 4-18 is an example.

      a.   Select [PCLK/2] in the item of [Clock source].

      b.   Select channel 1 slave, channel 2 slave, channel 3 slave.

      c.   Set the [Pulse cycle] on the [Master0] tab.

      d.   Set [Duty value] for each of the [Slave 1], [Slave 2], and [Slave 3] tabs.
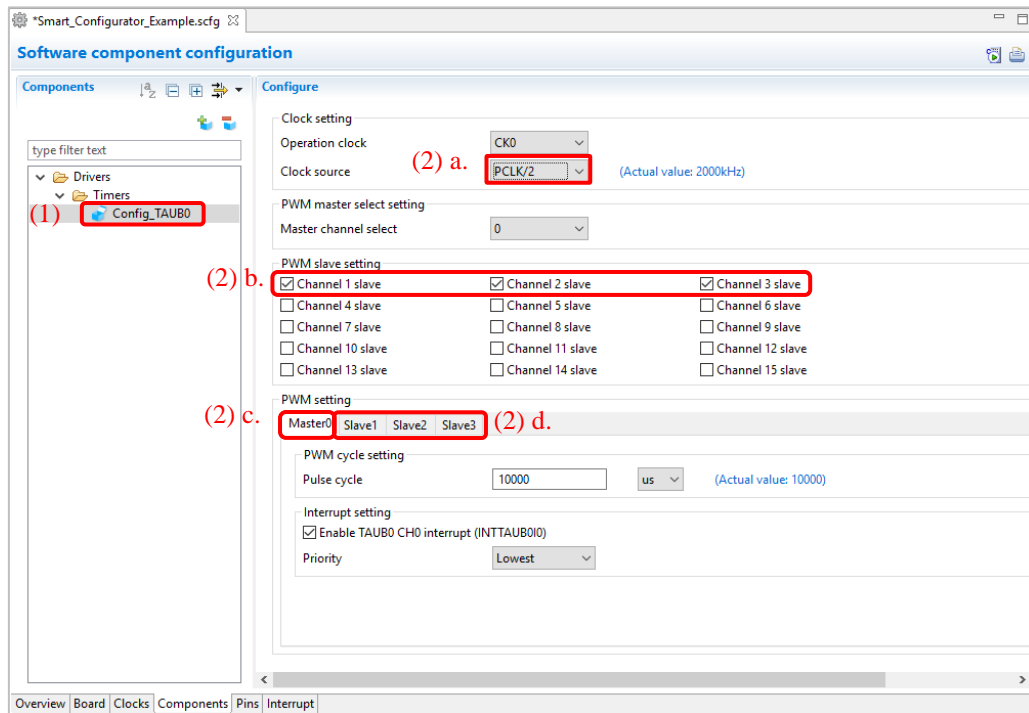
Figure 4-18 Component Configuration Settings

The code generation of the component is set to enabled by default.

Right click on the component and click [✓ Generate code], it changes to [Generate code] and no code is generated.

Clicking [Generate code] will change to [✓ Generate code] and generate code.

4.4.5   Changing the Resource for a Code Generator Component

You can change the resource of the component (for e.g., change from TAUB0 to TAUB1). Compatible configurations can be migrated from the current resource to the newly selected resource.

Follow the procedure below to change the resource.

(1)   Right-click on a component (for e.g., Config_TAUB0).

(2)   Select [Change resource] from the context menu.
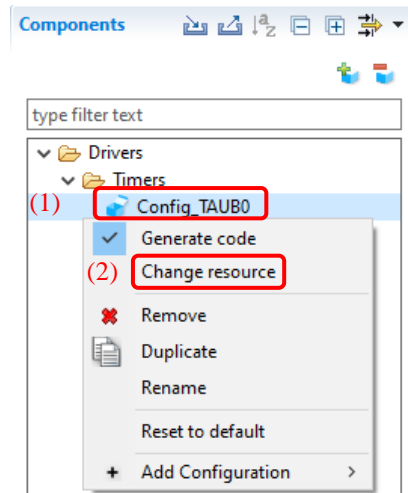
Figure 4-19 Resource Change

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

(4)   The [Next] button will be active; click on it.
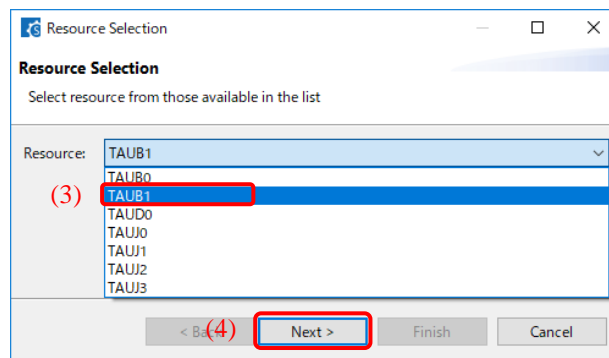
Figure 4-20 Select a New Resource

(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 below or default settings.

(8)   Click on [Finish].
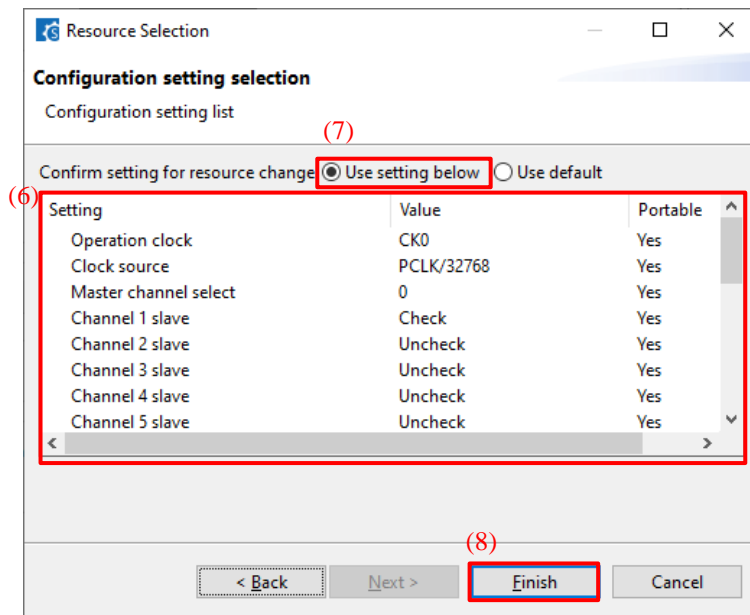
Figure 4-21 Confirm New Resource Settings

The resource is automatically changed (for e.g., changed INTTAUB0I0 to INTTAUB1I0).
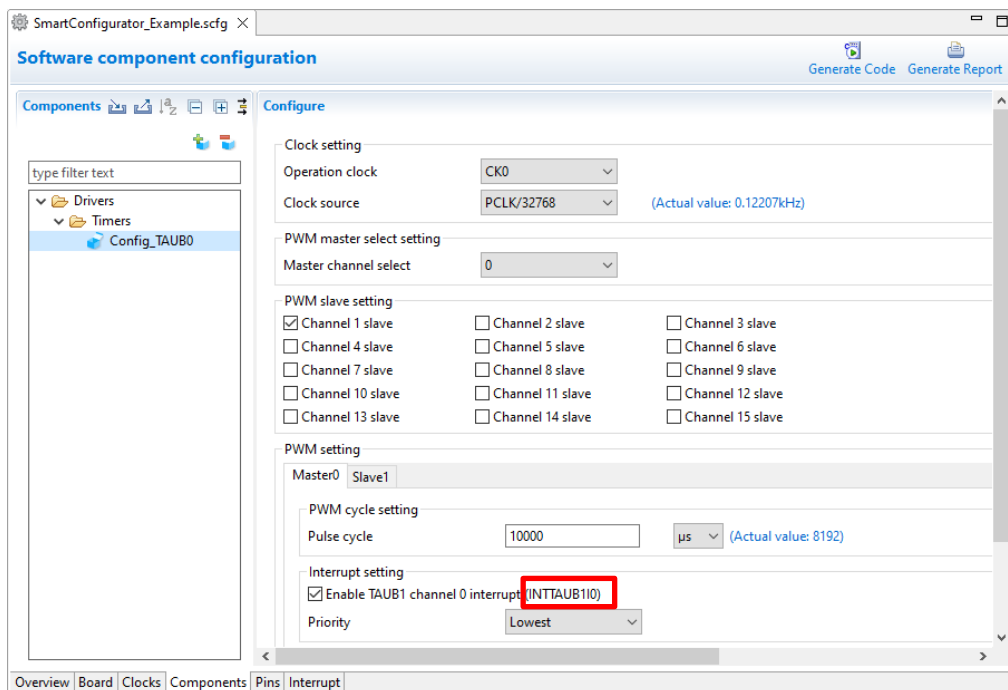
Figure 4-22 Resource Changed Automatically

To change the configuration name, follow the procedure below.

(9)　Right-click on the component.

(10)　Select [Rename] to rename the configuration (for e.g., change Config_TAUB0 to Config_TAUB1).
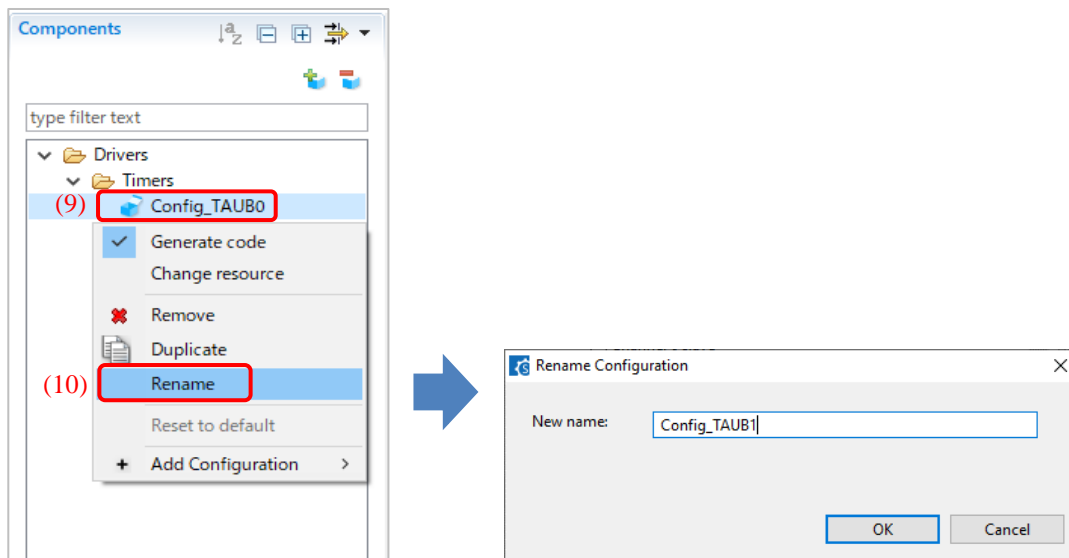
Figure 4-23 Renaming the Configuration

4.4.6   Export Component Configuration

To export the current configuration of a component, click the [Export Configuration] button on the [Components] tab page. The configuration will be saved as an *.xml file.
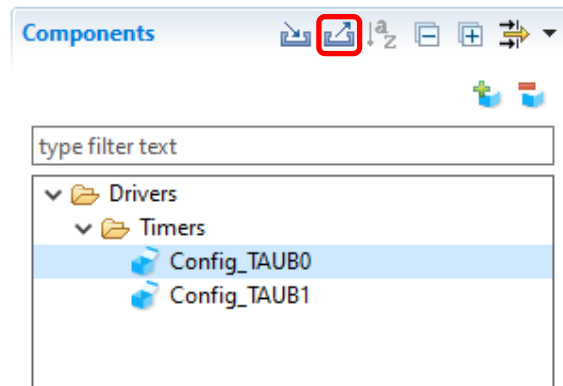
Figure 4-24 Export Configuration (xml format)

4.4.7   Import Component Configuration

To import the configuration of a component, click the [Import Configuration] button and select the exported XML file.
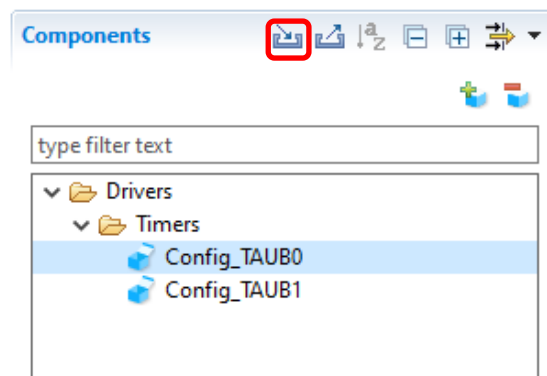
Figure 4-25 Import Configuration (xml format)

4.4.8   Configure General Setting of the component

User can change the general setting of the component such as location and dependency. To change it, please click the [Configure general settings...] link on the [Software Component Selection] page displayed in the [New Component] dialog (Figure 4-15 Selection of Software Components) and the [Preferences] dialog will be displayed.
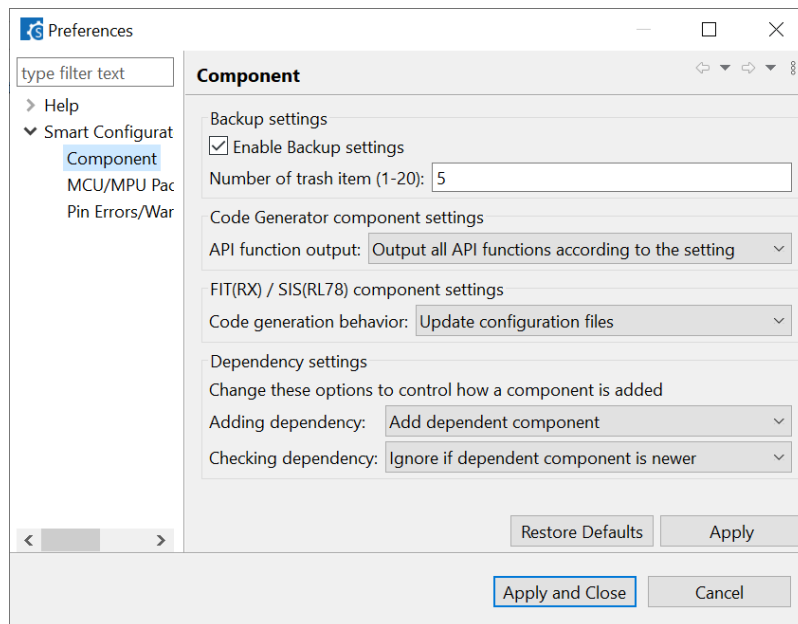
Figure 4-26. Configure General Setting of Component

Note:
1. You can select [Enable 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.
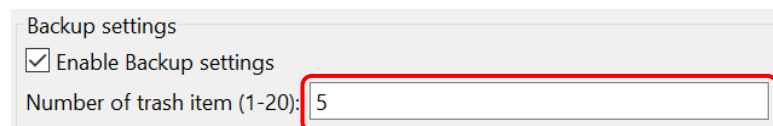
Figure 4-27 Trash number setting

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.
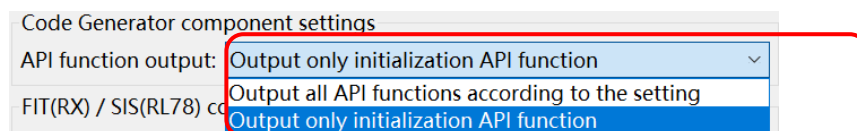
Figure 4-28 RH850 API function output setting

4.5   Pin settings

The [Pins] page is used for allocating pin functions. User can switch the display 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.
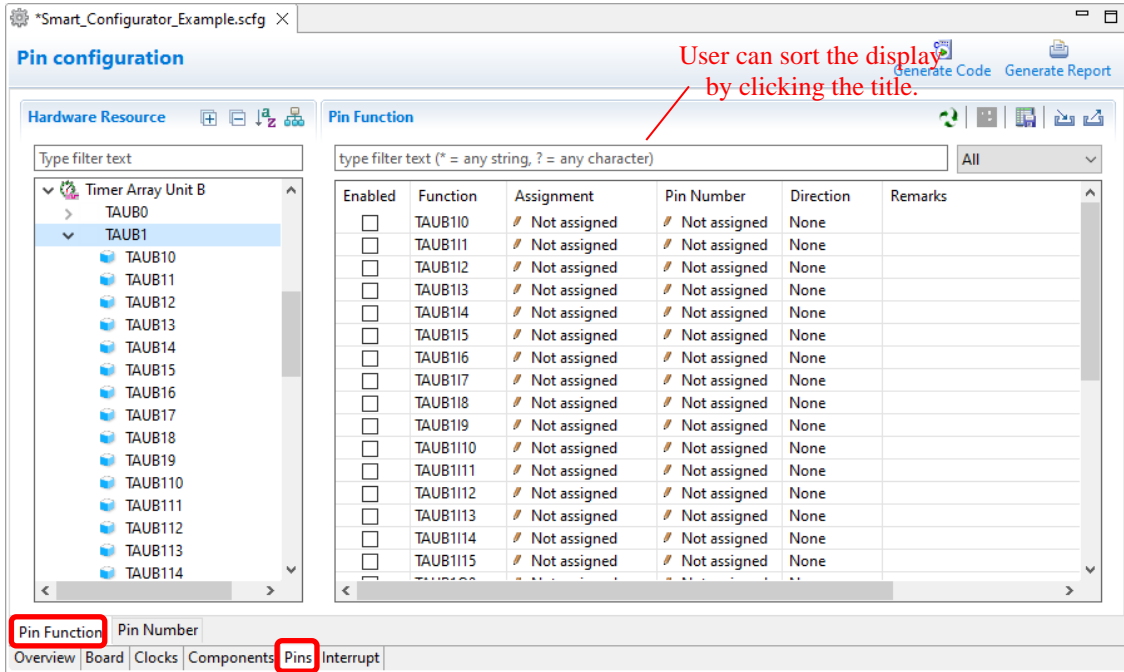
Figure 4-29 [Pins] Page ([Pin Function])

When you select a board on the [Board] page, the initial pin setting information of the board is displayed in [Board Functions]. In addition, the [■] icon displayed in the [Function] selection list indicates the initial pin function of the board.
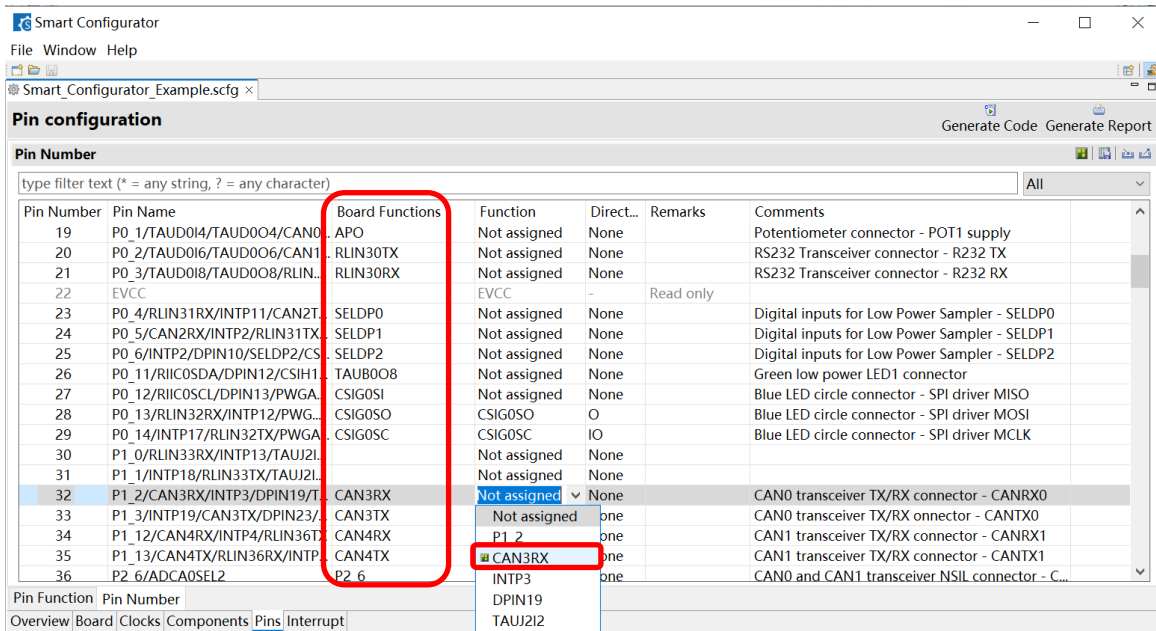
Figure 4-30 [Pins] Page ([Pin Number])

4.5.1   Assign Pins to Resources

In the Pins page, assign pin to the resource used by the component. Pin assignment can be done in either [Pin Function] list or [Pin Number] list.

The procedure for pin assignment in the [Pin Function] list is described below.

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

(2) Select the target software component (for e.g., Interrupt controller unit).

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

(4) Pin assignment is performed with the [Assignment], [Pin Number] column, or [🔄 (Next group of pins for the selected resource)] button.

   (a) Click [Assignment] or [Pin Number] and assign a terminal from the list (for e.g., change from P0_1 to P10_0 with [Assignment]).

   (b) Click the [🔄 (Next group of pins for the selected resource)] button and change the pin assignment. Each time click, the pin with the function switches.
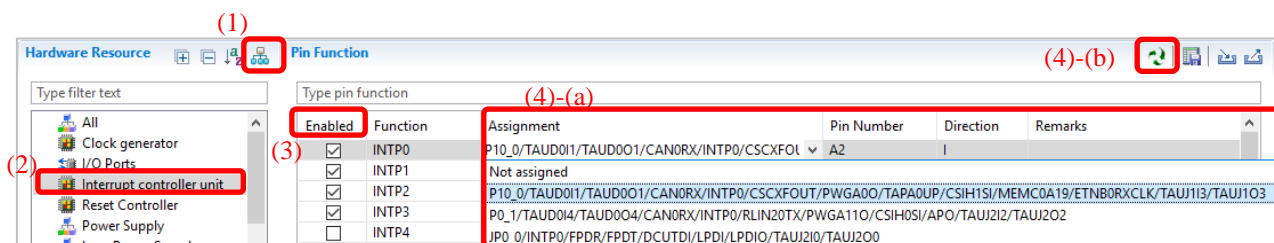
Figure 4-31 Pin Assignments in the [Pin function] List

When the component is set, the check box in the [Enabled] column is checked. Pin assignment is possible even when the component is not set. If pin assignment is done without component being set, we will display "There is no software initialising this pin" in the [Remarks] column.

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…].
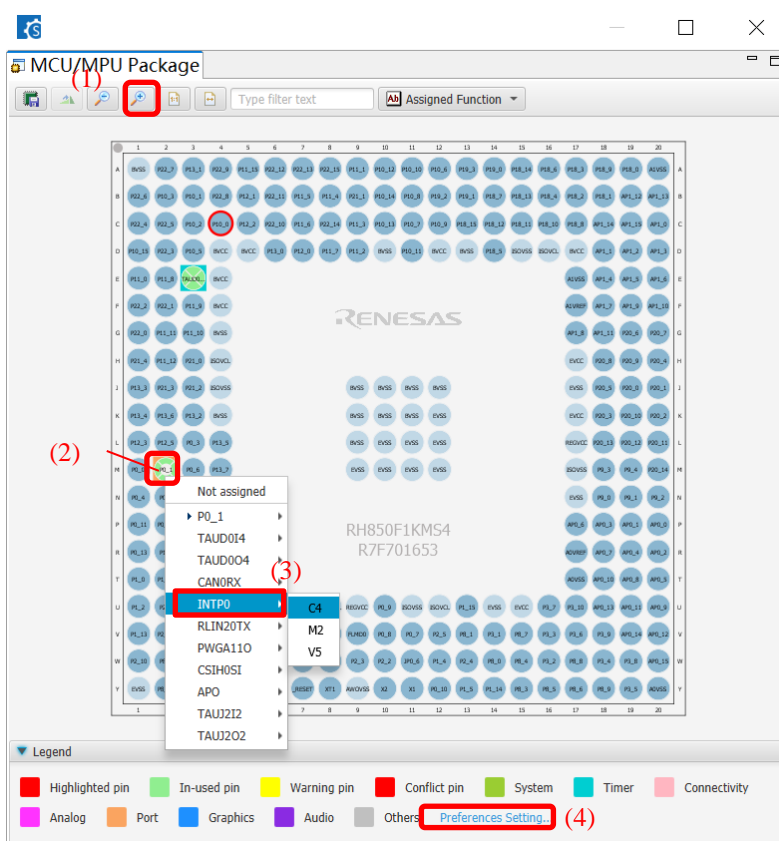
Figure 4-32 Assigning Pins Using the MCU/MPU Package View

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.
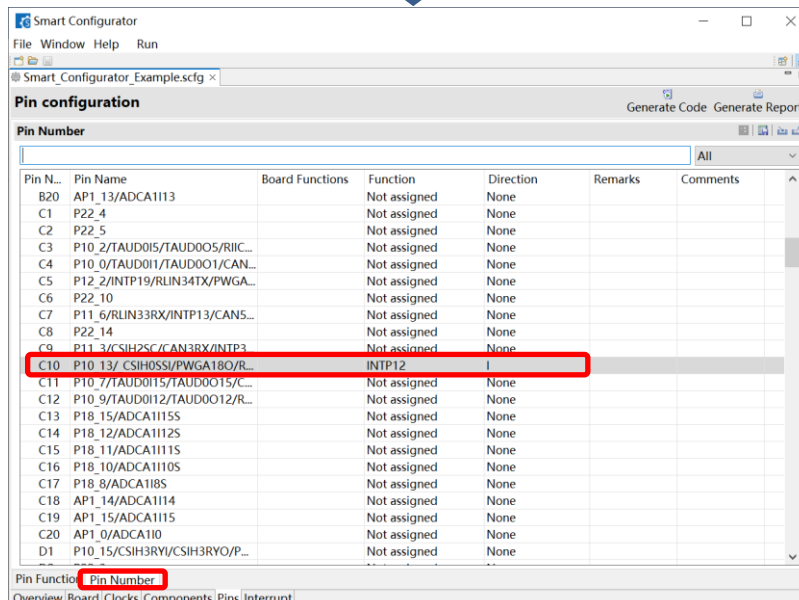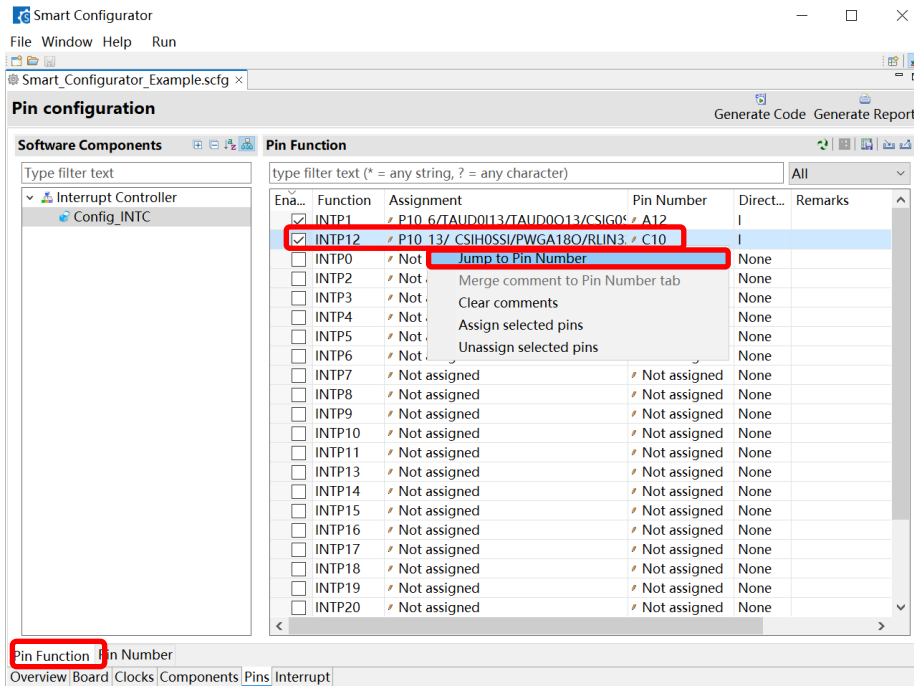
Figure 4-33 Jump to Pin Number

4.5.4 Export Pin Settings

The pin setting can be exported. Exported files can be imported into projects of the same device family. Follow the procedure below to export the pin settings.

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

(2) In the [Export] dialog, enter the file name to export.
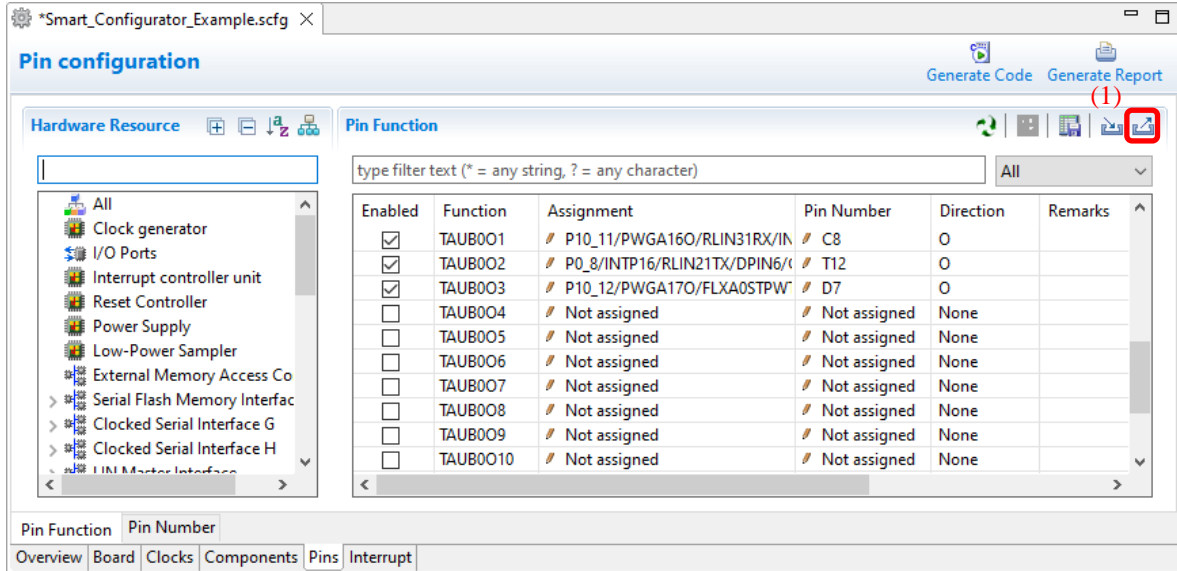
Figure 4-34 Export Pin Settings (XML format)

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 Import Pin Settings

Users can import XML format files including pin assignment settings. After importing a file, the terminal assignment setting will be reflected in the [Pin configuration] page. Please follow the procedure below to import the pin settings.

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

(2) In the [Import] dialog, enter the file name to import.
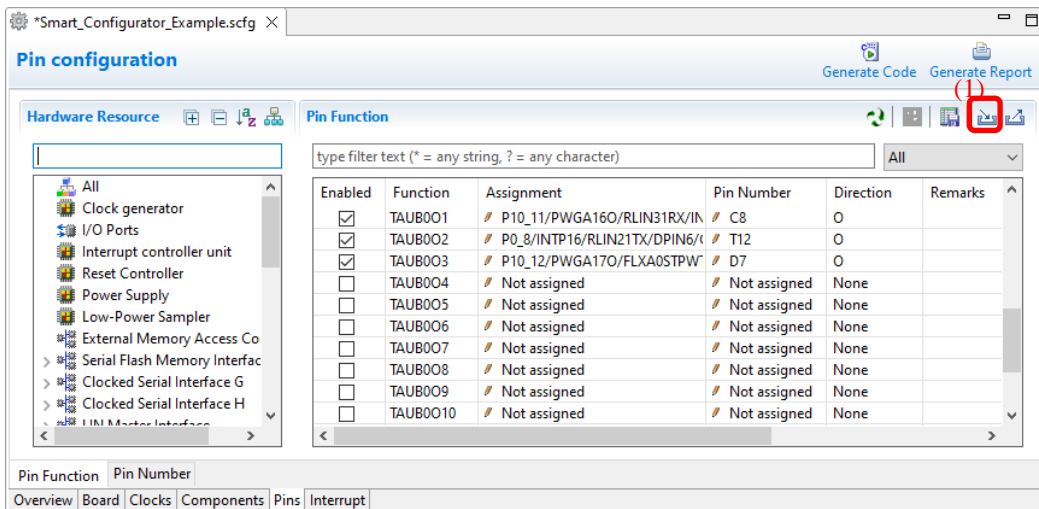
Figure 4-35 Import Pin Settings (XML format)

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 is selected to use in [Board] tabbed page.

The following describes the procedure for collective setting of pins.

(1)   Select a board setting information except [Custom User Board] in [Board] page. User can refer to 4.1.2 Selecting the Board.

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

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

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

(5)   Click [OK].
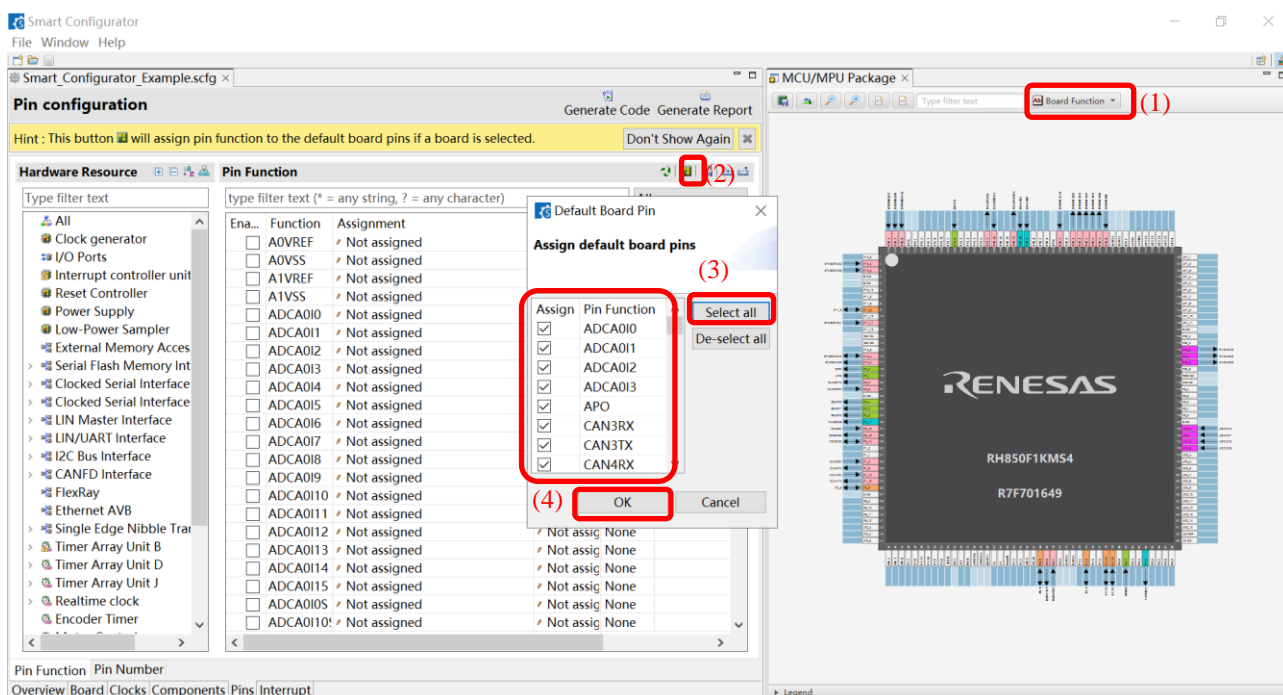
Figure 4-36   Setting for initial pin configuration

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.
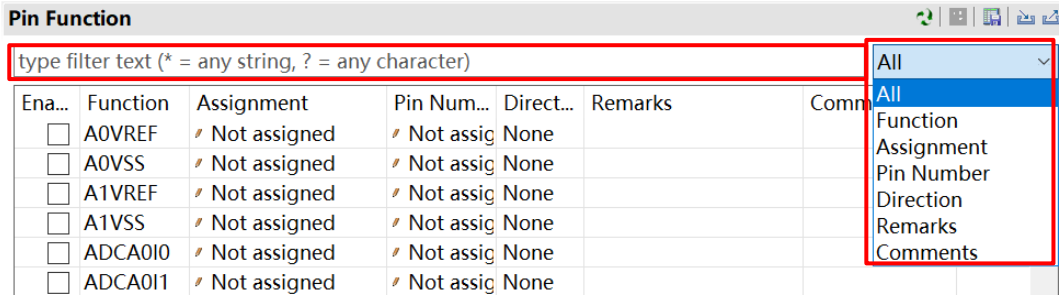
Figure 4-37 Filter for [Pin Function] tab
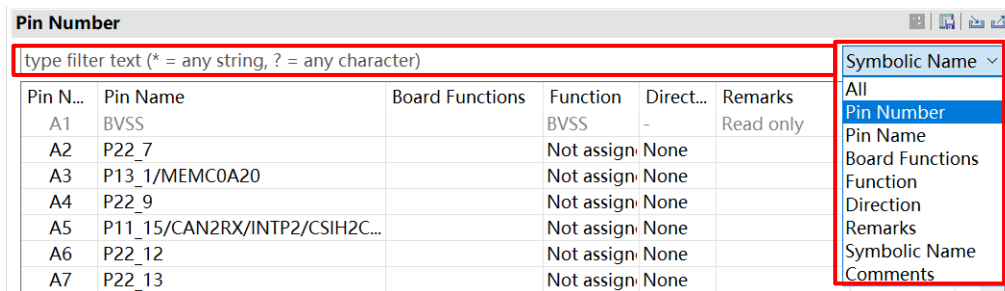
Figure 4-38 Filter for [Pin Number] tab

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 [Smart Configurator] > [Pin Errors/Warnings] and use the combo boxes to change the errors/warning setting.
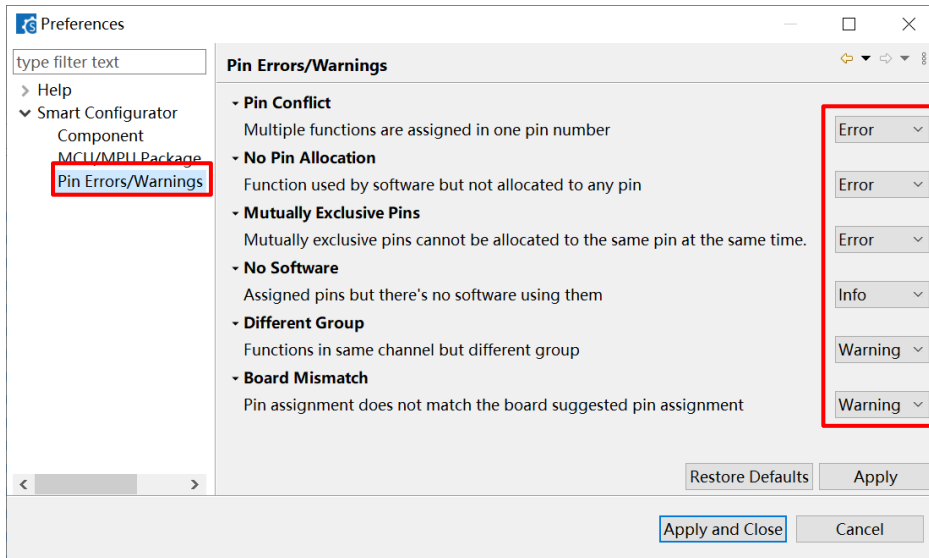
Figure 4-39 Pin Errors/Warnings settings at Preferences

Example: Change "No Software" setting from "Info" to "Error"
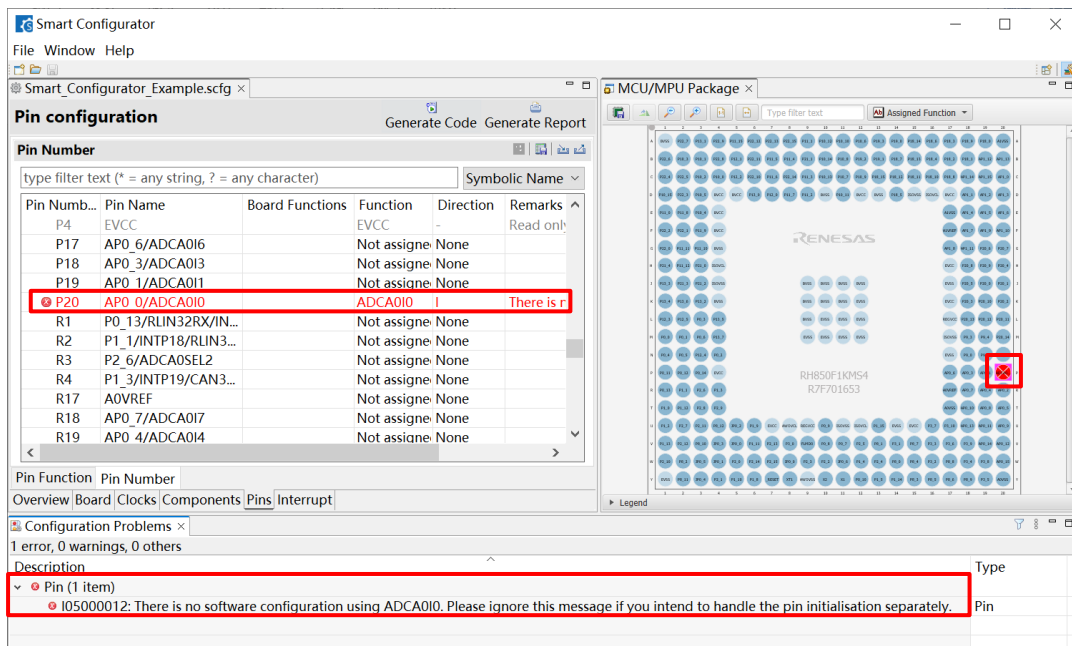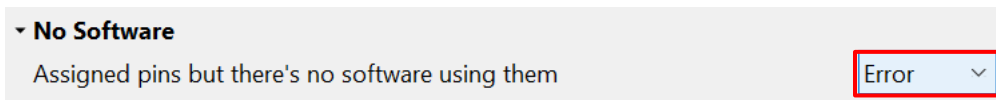
Figure 4-40 Change "No Software" setting from "Info" to "Error"

4.6   Interrupt Settings

Check and set the interrupts of the peripheral modules that have been selected on the [Components] page.
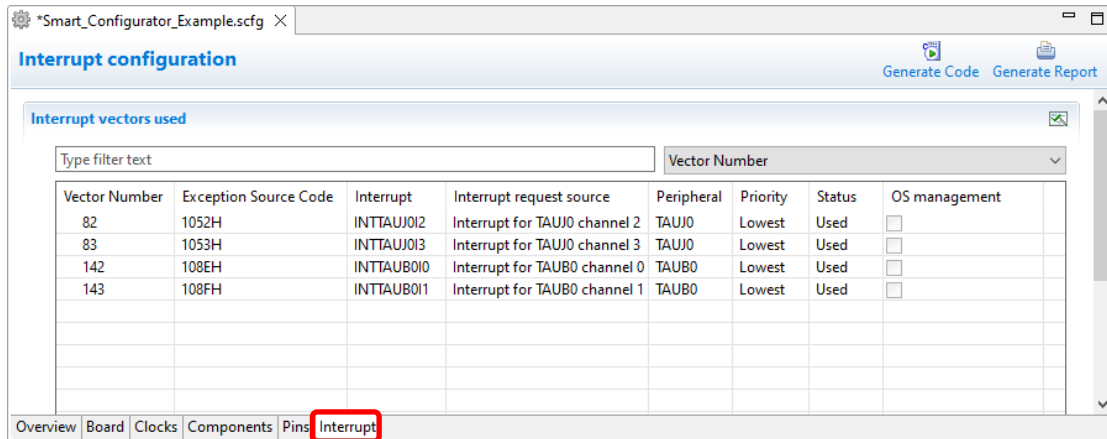
Figure 4-41 [Interrupts] Page

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)   The interrupt priority level can be changed in the [priority] column. Setting the priority level is reflected in the configuration settings of the component.

(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.
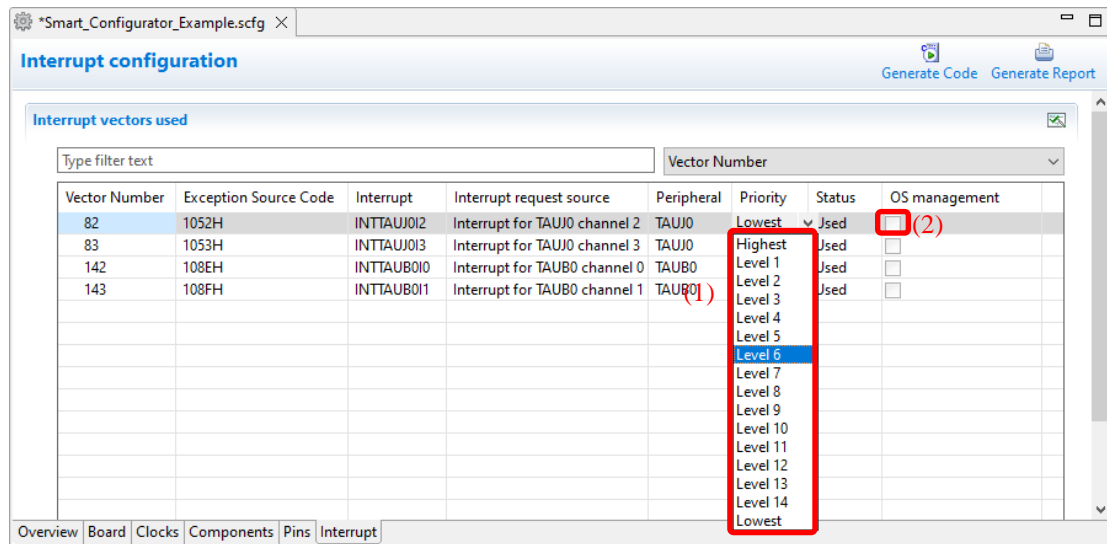
Figure 4-42 Interrupt Settings

4.6.2 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 4-44 Interrupt Enable/Disable Functions Code Example**
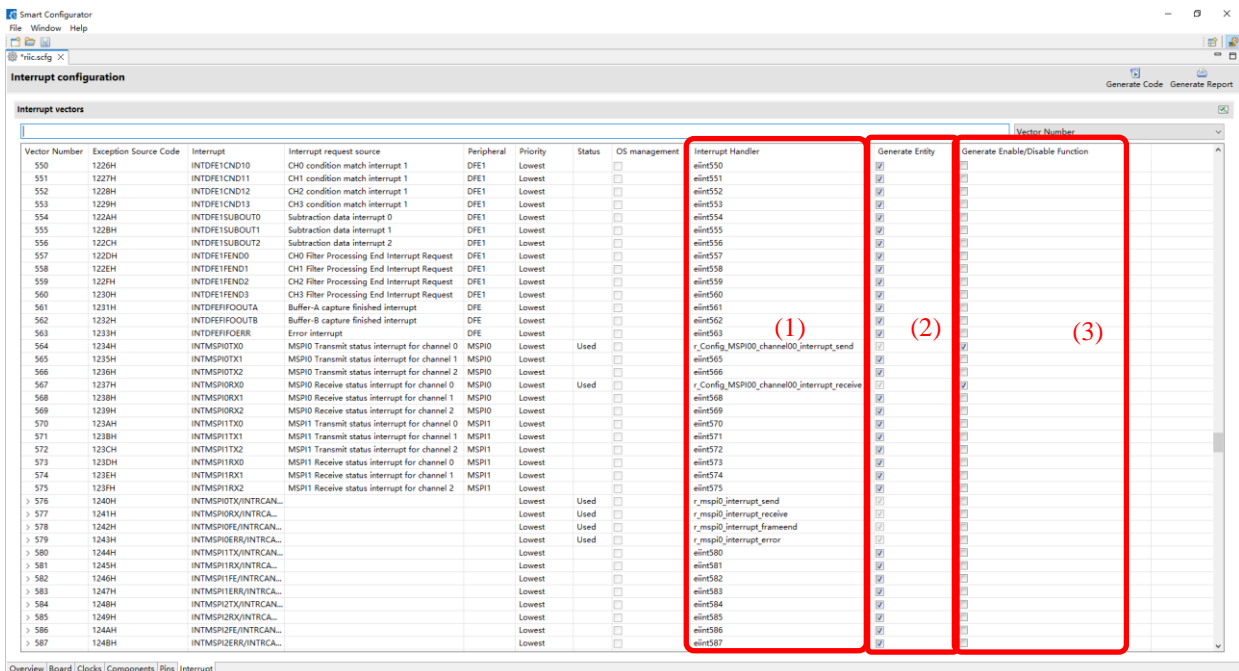
Figure 4-43 Interrupt Settings

```c
r_smc_interrupt.c ☒
19
20   /***********************************************************************
21   * File Name     : r_smc_interrupt.c
22   * Version       : 1.3.0
23   * Device(s)     : R7F702301BEBBA
24   * Description   : None
25   **********************************************************************
26   /***********************************************************************
29   /* Start user code for pragma. Do not edit comment generated here */
30   /* End user code. Do not edit comment generated here */
31
32   /***********************************************************************
35   #include "r_cg_macrodriver.h"
36   #include "r_cg_userdefine.h"
37   #include "r_smc_interrupt.h"
38
39   /***********************************************************************
42   void R_Interrupt_Create(void)
43   {
44
45   }
46
47   void r_Config_MSPI00_channel00_interrupt_send_enable_interrupt(void)
48   {
49       /* Clear INTMSPI0TX0 request and enable operation */
50       INTC2.EIC244.BIT.EIRF244 = _INT_REQUEST_NOT_OCCUR;
51       INTC2.EIC244.BIT.EIMK244 = _INT_PROCESSING_ENABLED;
52   }
53
54   void r_Config_MSPI00_channel00_interrupt_send_disable_interrupt(void)
55   {
56       /* Disable INTMSPI0TX0 operation and clear request */
57       INTC2.EIC244.BIT.EIMK244 = _INT_PROCESSING_DISABLED;
58       INTC2.EIC244.BIT.EIRF244 = _INT_REQUEST_NOT_OCCUR;
59   }
60
61   void r_Config_MSPI00_channel00_interrupt_receive_enable_interrupt(void)
62   {
63       /* Clear INTMSPI0RX0 request and enable operation */
64       INTC2.EIC245.BIT.EIRF245 = _INT_REQUEST_NOT_OCCUR;
65       INTC2.EIC245.BIT.EIMK245 = _INT_PROCESSING_ENABLED;
66   }
67
68   void r_Config_MSPI00_channel00_interrupt_receive_disable_interrupt(void)
69   {
70       /* Disable INTMSPI0RX0 operation and clear request */
71       INTC2.EIC245.BIT.EIMK245 = _INT_PROCESSING_DISABLED;
72       INTC2.EIC245.BIT.EIRF245 = _INT_REQUEST_NOT_OCCUR;
73   }
74
```

Figure 4-44 Interrupt Enable/Disable Functions Code Example

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

In Smart Configurator, 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:

(1) PE*n* is chosen to be used in [System] page (please refer to chapter 4.3, System Settings (RH850/U2A only))

(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 PEn column can respond.

     b   Connected to INTC2 shared by multiple PEs, only one PE selected in PEn column can respond.
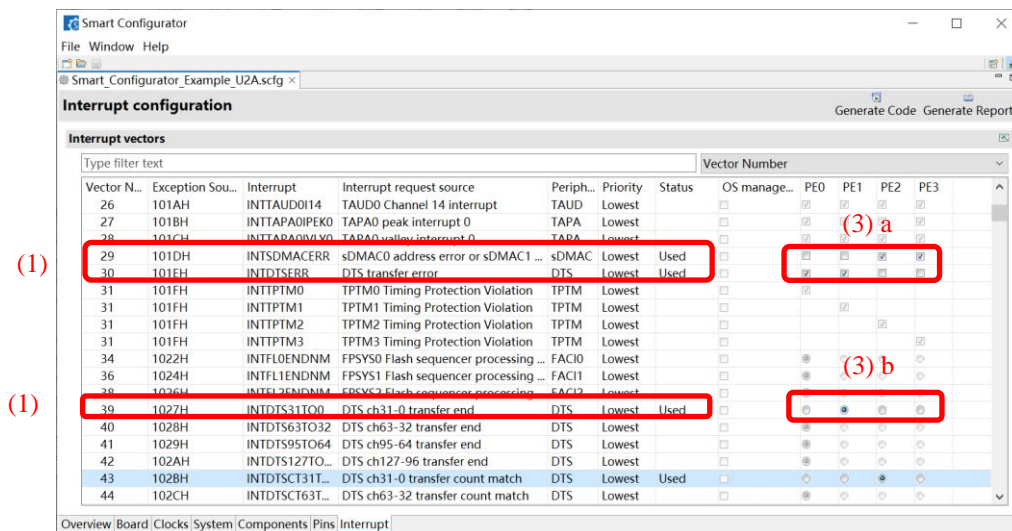
Figure 4-45 PE*n* setting
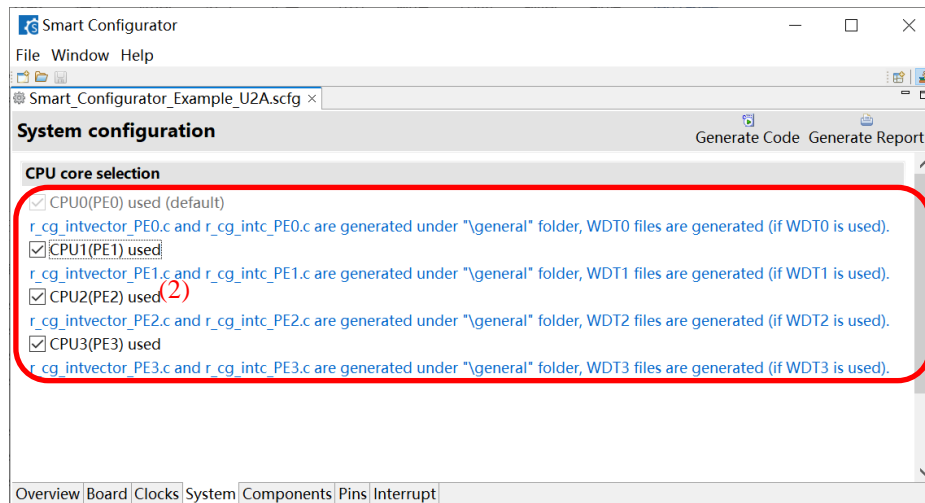
Figure 4-46   PE*n* is chosen to be used or unused in [System]

Note:

Only RH850/U2A supports PE*n* 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.

5. Managing Conflicts

Adding components, setting pins and interrupts may cause problems related to resource mismatches. This information will be displayed in the [Configuration Problems view]. User can refer to the information displayed to fix the conflict issues.

5.1   Resource Conflicts

When two software components are configured to use the same resource (for e.g., TAUB1), an error mark (🔳) will be displayed in the [Components tree].

The [Configuration Problems view] will display messages on peripheral conflicts to inform the user in which software configurations peripheral conflicts have been detected.
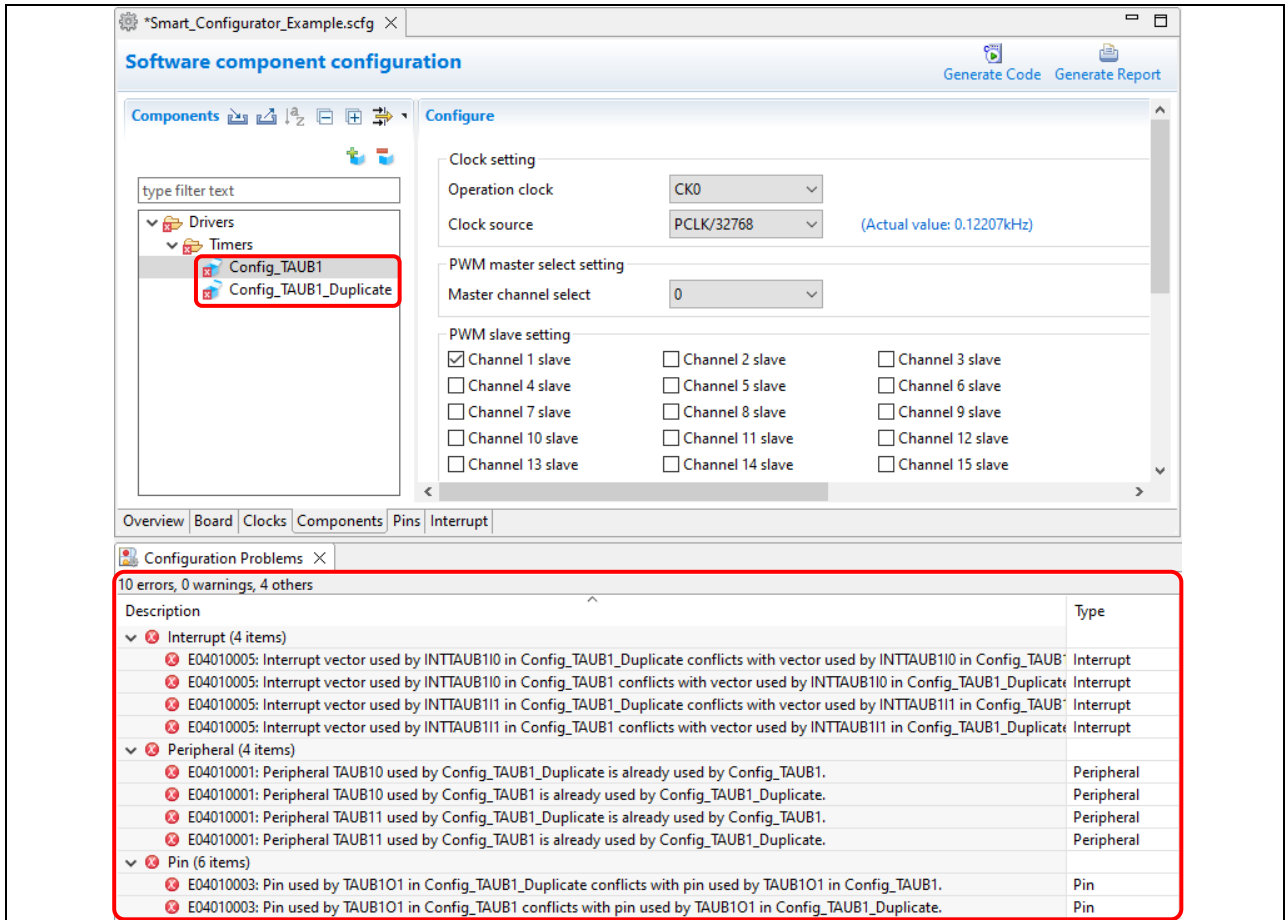
Figure 5-1 Resource Conflicts

5.2   Resolving Pin Conflicts

When multiple pin functions are assigned to the same pin, an error mark ❌ is displayed in the tree and [Pin Function] list on the [Pins] page.
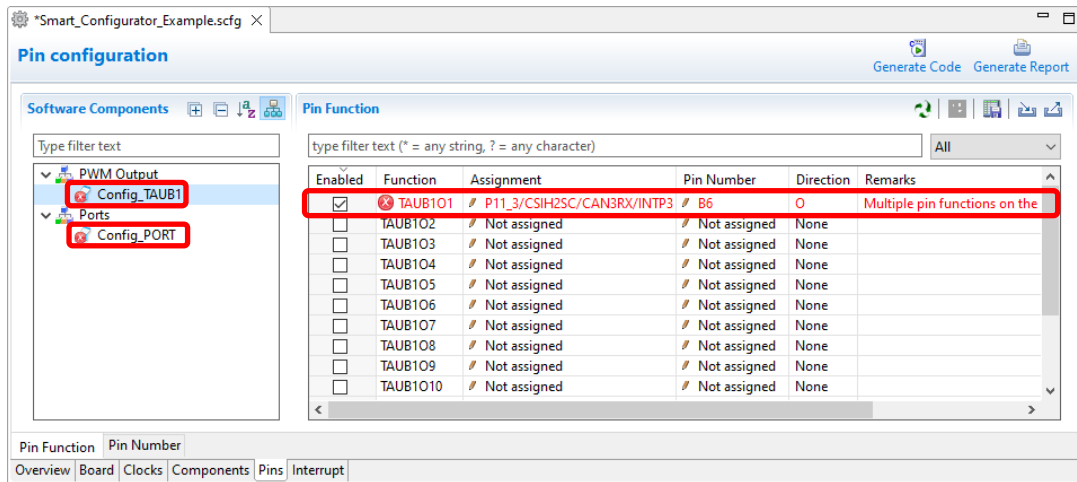
Figure 5-2 Pin Conflicts

Detailed information regarding conflicts is displayed in the [Configuration Problems view].
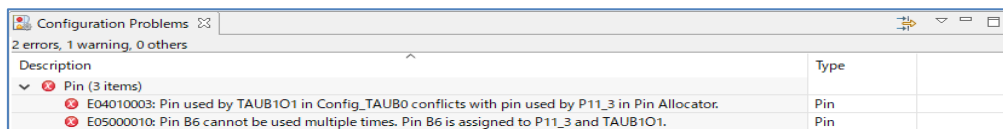
Figure 5-3 Pin Conflict Message

To resolve a conflict, right-click on the node with an error mark on the tree and select [Resolve conflict].
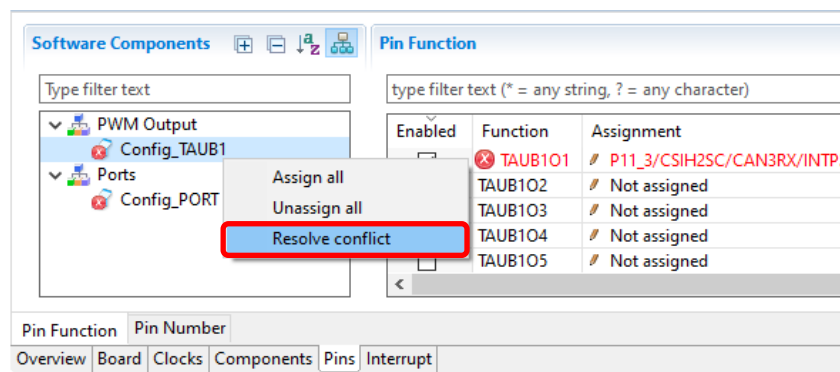
Figure 5-4 Resolving Pin Conflicts

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

6.   Generating Source Code

6.1   Generating Source Code File

Output a source file for the configured details by clicking on the [🖼 (Generate Code)] button in the Smart Configurator view.
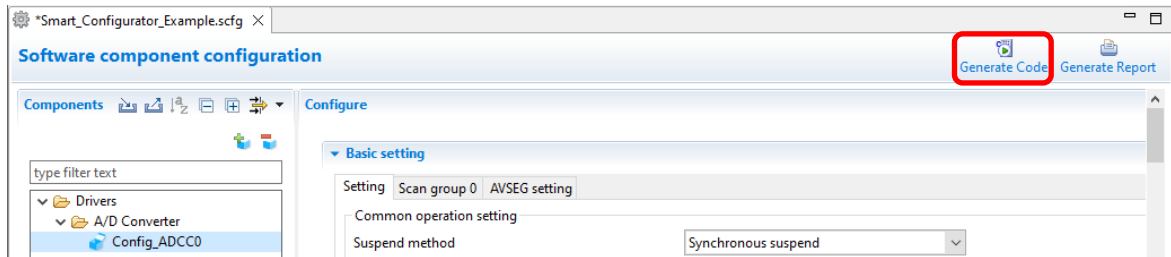
Figure 6-1 Generating a Source File

The Smart Configurator generates a source file in <ConfigurationFileDir>\src\smc_gen. If user's Smart Configurator has already generated a file, a backup copy of that file is also generated (refer to the section 6.6 Backing up Generated Source Code).

6.2 Configuration of Generated Files and File Names

Figure 6-2, Configuration of Generated Files and File Names, shows the folders and files output by the Smart Configurator. "*ConfigName*" indicates the configuration name set in the component. "*ProjectName*" indicates the project name.
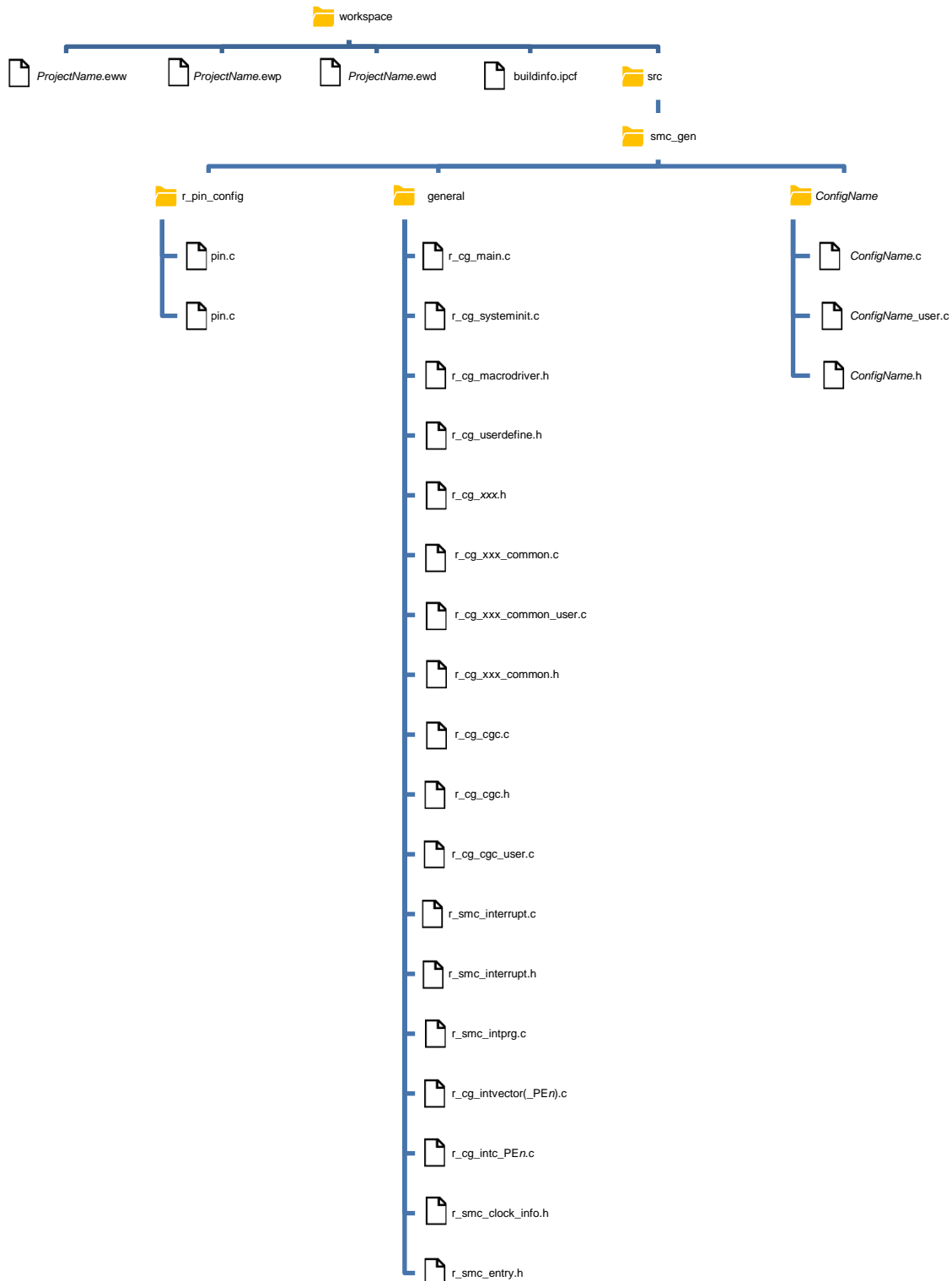
Figure 6-2 Configuration of Generated Files and File Names (IAREW)
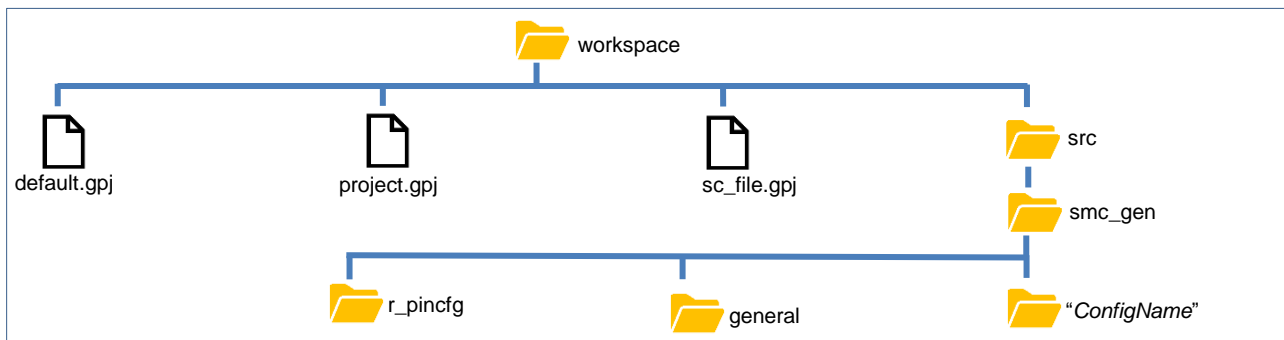
Figure 6-3 Configuration of Generated Files and File Names (MULTI)

Note: The generated files and file name under smc_gen folder is same for IAREW and MULTI.

Table 6-1 Description of Generated File

Folder	File	Description
Workspace	-	This folder is not generated by Smart Configurator,but is the location that user selects to create Smart Configurator project. All folders and files generated by Smart Configurator will be generated into this folder.
	ProjectName.eww	This file is generated when Smart Configurator project is created for the first time. It is the workspace file for IAREW.
	ProjectName.ewp	This file is generated when Smart Configurator project is created for the first time. It is the project file for IAREW.
	ProjectName.ewd	This file is generated when Smart Configurator project is created for the first time. It is the debug file for IAREW.
	buildinfo.ipcf	This file is generated when clicking button "Generate Code" in Smart Configurator each time. It is the connection file for IAREW. It includes the path for all files generated for each component by Smart Configurator.
	default.gpj	This file is generated when clicking button "Generate Code" in Smart Configurator each time. It is the Top project for MULTI. when user double-clicks this file, MULTI will start run. It includes the path for all files generated for each component by Smart Configurator.
	project.gpj	This file is generated when clicking button "Generate Code" in Smart Configurator each time. It is the Program type of MULTI project file which will list program source files.
	sc_file.gpj	This file is generated when clicking button "Generate Code" in Smart Configurator each time. It is the sub project for MULTI. It includes the path for all files generated for each component by Smart Configurator.
{ConfigName}	-	This folder is generated for the components 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*
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 Symbolic name definition/user guide/API and the function prototypes of pin setting in *Pin.c*.
general	-	This folder is always generated. It contains header files and source files commonly used by drivers of the same peripheral function.
	r_cg_xxx.h[*1]	These files are only generated for the used components. The files contain macro definitions for setting SFR registers.

Folder	File	Description
general	*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*. User can add codes and functions in the dedicated user code areas.
	r_cg_intvector(_PEn).c	*r_cg_intvector.c* is generated only for RH850/F1KM and RH850/F1KH *r_cg_intvector_PEn.c(n=0~3)* is generated only for RH850/U2A (only when PE*n*(n=0~3) is used, the r_cg_intvector_PE*n*.c(n=0~3) are generated.) r_cg_intvector_PE1.c is generated only for RH850/C1M. r_cg_intvector_PE0.c is generated only for RH850/U2B. This file 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 macro definition for interrupt priority.
	r_smc_intprg.c	This file is only generated with selecting RH850/U2B group devices. It contains all interrupt handler entity according to [Interrupts] page setting.
	r_smc_clock_info.h	This file is only generated when selecting RH850/U2B group devices. It contains clock setting macro definitions.
	r_cg_intc_PEn.c	This file is generated only for: RH850/U2A: Only when PE PE*n*(n=0~3) is 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. It 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.

Folder	File	Description
general	*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_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"*.

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

6.3 Initializing Clocks

Configurations of clock source in [Clocks] page are generated in \src\smc_gen\r_config folder.
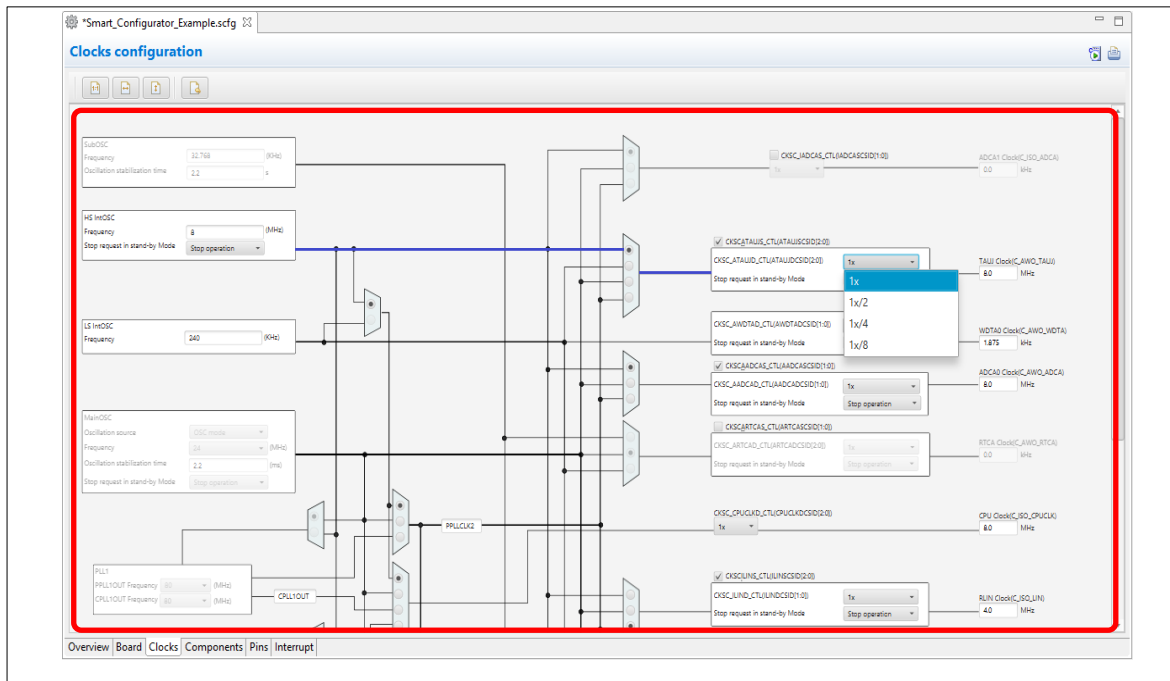
Figure 6-4 Clocks Source Configuration

Table 6-2 Clock Source File Description

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 clocks initialization in *R_CGC_Create*.
	r_cg_cgc_user.c	R_CGC_Create_UserInit	This API function is for user to add code in *R_CGC_Create* after the CGC initialization.
	r_smc_clock_info.h	Macros related to clocks	These macros are for clocks setting in [Clocks] page GUI.

6.4   Initializing Pins

Pin configuration settings are generated by the component into source files as shown in (1) and (2) below.

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

The pin function is initialized with *R_ConfigName_Create* of
\src\smc_gen\{ConfigName}\{ConfigName}.c.

Table 6-3 File to Initialize Pins

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

(2)   Reference pins initialization codes

Refer to *Pin.c* in the *\src\smc_gen\r_pincfg* folder for the initialization code of all pin functions set on the [Pins] page (except I/O ports).

Table 6-4 Reference File for Initialization of All Pins

Folder	File	Function	Description
r_pincfg	*Pin.c*	*R_Pins_Create*	This function contains the initialization codes of all pins function configured at [Pins] page except I/O ports.

6.5   Initializing Interrupts

Configurations in [Interrupt] page are generated in few source files.
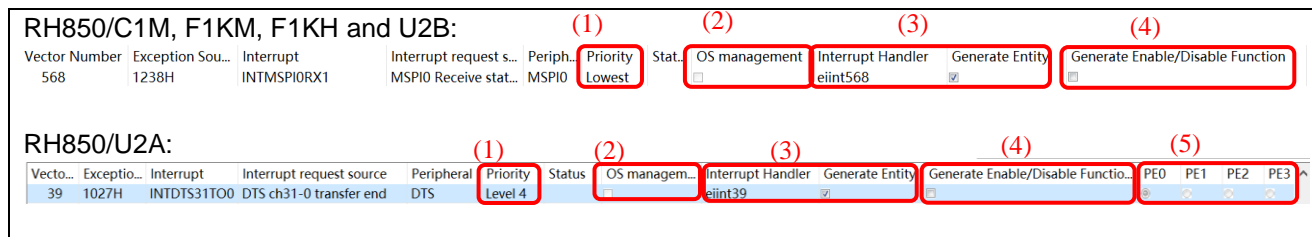
Figure 6-5 Interrupts Configuration in Interrupts View

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.

6.6　Backing up Generated Source Code

The smart configurator has a source code backup function.

<ConfigurationFileDir>\trash\<Date-and-Time>

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

7. Loading generated files in Integrated development environment

Load source code output by Smart Configurator on Integrated Development Environment Platform.

7.1 Loading generated files in IAR Embedded Workbench

7.1.1 Loading files generated by IAR RH850 Toolchain

When IAR environment is selected for the compiler to be used, Smart Configurator outputs the related project files (.eww/.ewp/.ewd/r_cg_main.c) together with the source files. It is not necessary for the user to create project files in IAR Embedded Workbench.

The usage procedure is as follows.

(1) Select [Open Workspace…] from the [File] menu of IAR Embedded Workbench.

(2) In the [Open Workspace] dialog box, browse to the folder where the project file is saved, select the project file (.eww), and click the [Open] button.
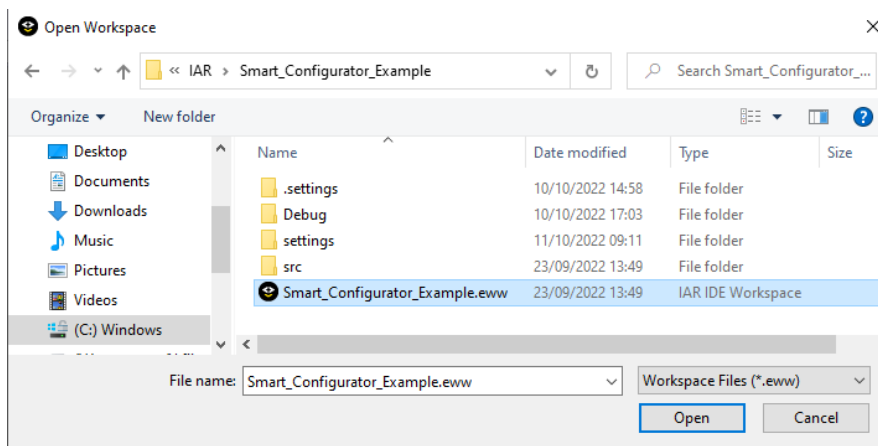
Figure 7-1. Load a *.eww File

(3) The source file output by the Smart Configurator is added to the IAR C project workspace.
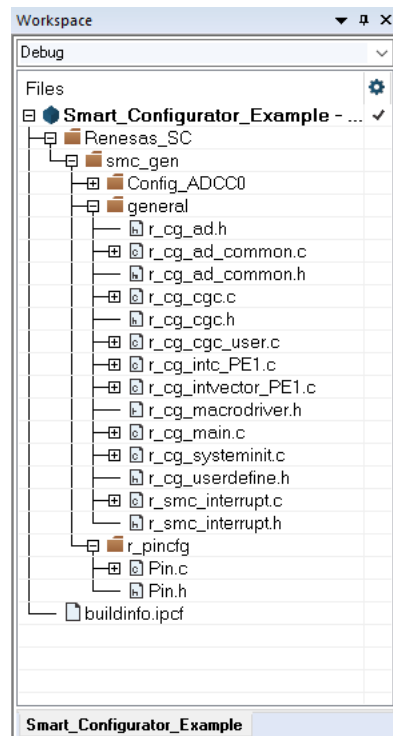
Figure 7-2. New Files Added to IAR Workspace

(4)   Select [Options…] from the [Project] menu of IAR Embedded Workbench.

(5)   In the [Options for node "*ProjectName*"] dialog box, change target device to match with the target device
       selected when creating Smart Configurator's configuration file.
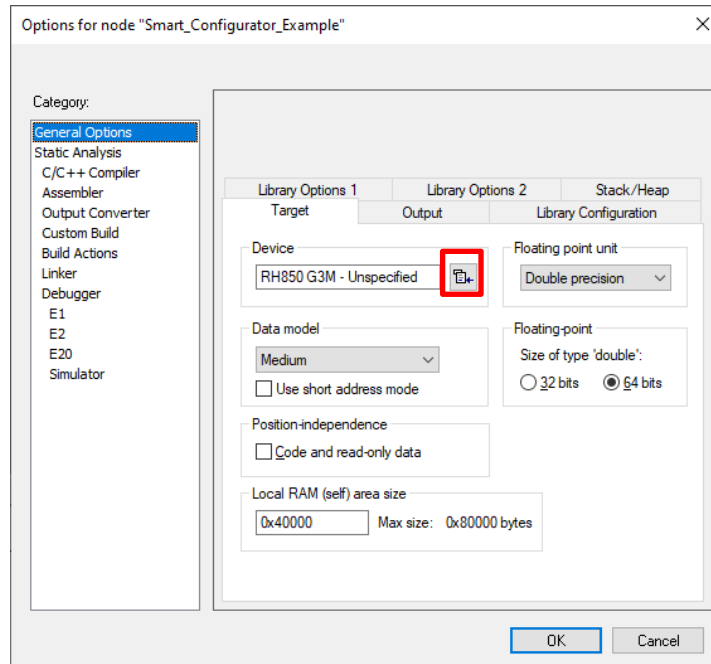
Figure 7-3. Change Target Device

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

When start Smart Configurator standalone and select All Toolchain (CC-RH, GHS, IAR) as toochain, Smart Configurator outputs the project connection file (.ipcf) together with the source files that can adapt to all three toolchains: CC-RH, GHS and IAR. When loading these generated files in IAR Embedded Workbench, the usage procedure is as follows.

(1)   Create a new project in IAR Embedded Workbench.

(2)   Select [Project] -> [Add Project Connection] as below show:
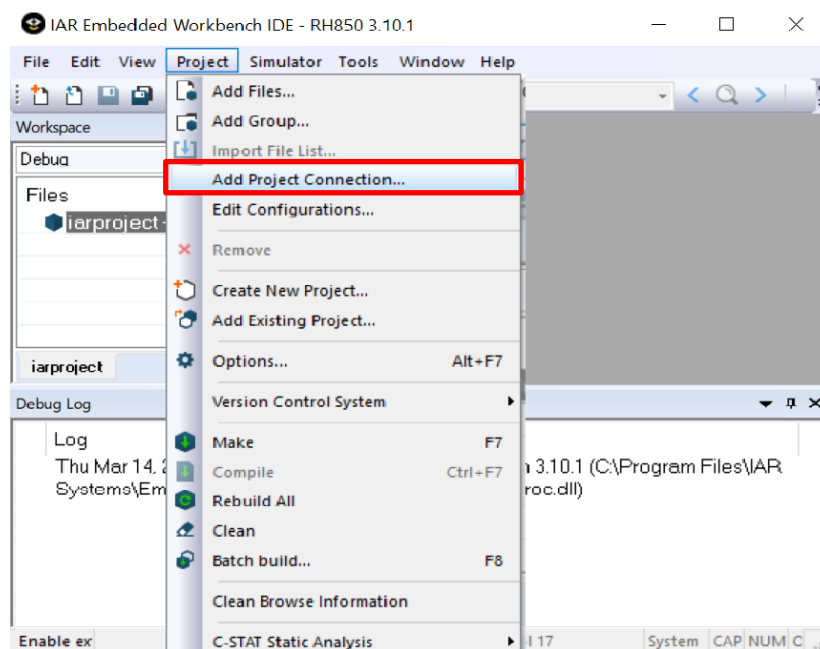
Figure 7-4. Add Project Connection

(3)   Selecting [IAR Project Connection] on view Add Project Connection
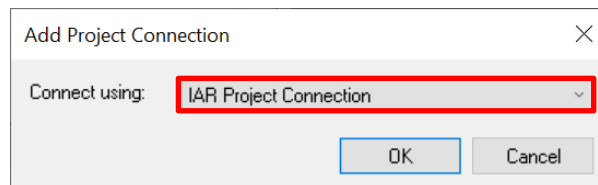
Figure 7-5. Select Connect using

(4) In [Select Project Connection File] view, browser to the Smart Configurator project folder, then select buildinfo.ipcf as below figure shows:
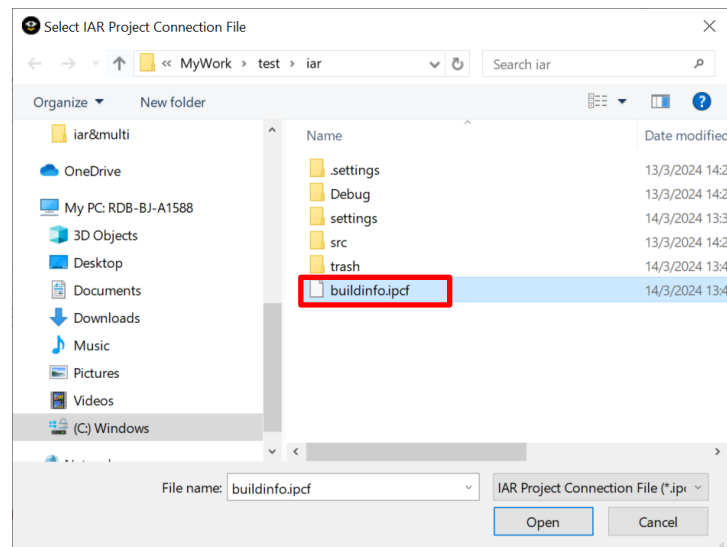
Figure 7-6. Select buildinfo.ipcf

(5) Select [Options…] from the [Project] menu of IAR Embedded Workbench.

(6) In the [Options for node "ProjectName"] view, select [C/C++ Compiler] to set additional include directories for "iodefine.h" manually by click "…":
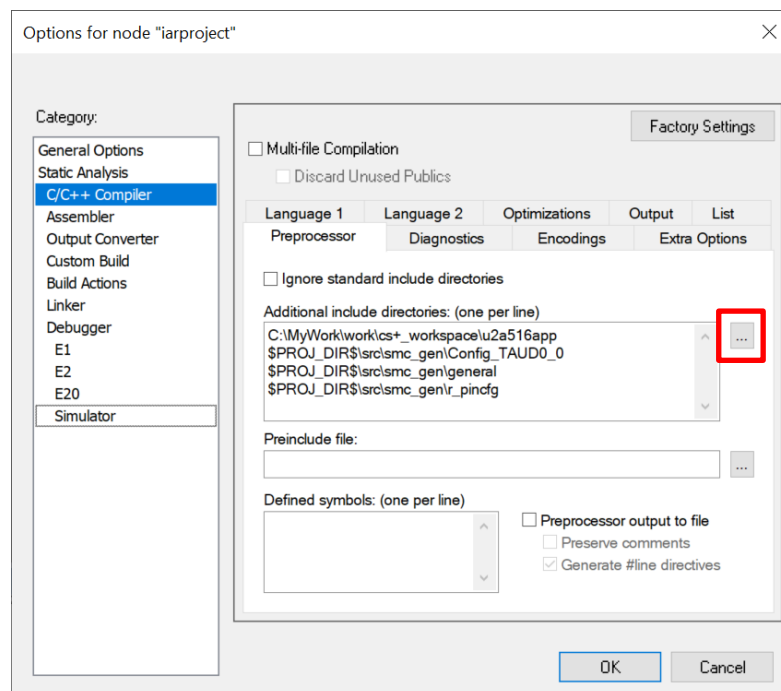
Figure 7-7. Set Additional include directories

(7)   In the [Options for node "ProjectName"] view, select [General Options] to change target device to match
       with the target device selected when creating Smart Configurator's configuration file.
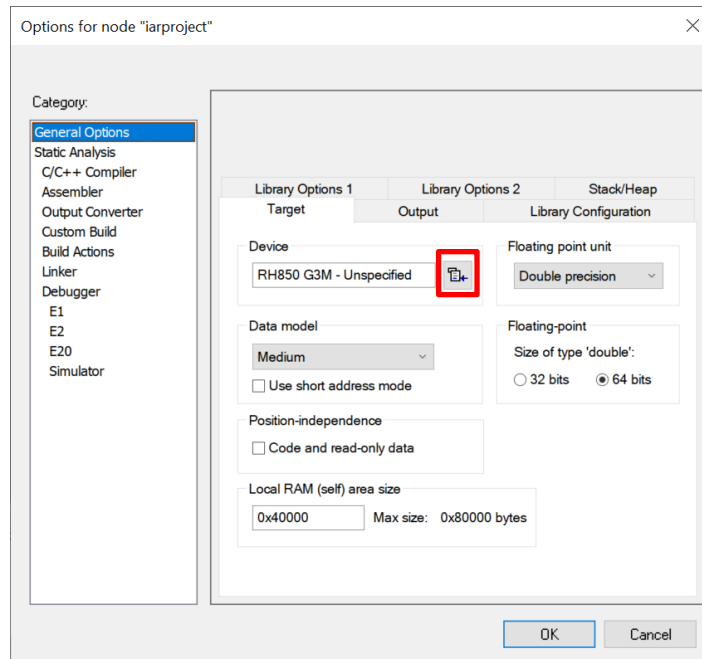
Figure 7-8. Specify Target Device

7.2   Loading generated files in GHS MULTI

7.2.1   Loading files generated by GHS RH850 Toolchain

When GHS environment is selected for the compiler to be used, Smart Configurator also outputs the project connection file (.gpj) together with the source files. The project connection file contains source file registration information. It is not necessary for the user to add or delete source files after configuration change in the Smart Configurator.

The usage procedure is as follows.

(1)   Select [Open Project Manager ...] from the MULTI [Components] menu.

(2)   As the [Select a project to open] dialog box appears, browse to the folder where the configuration file is saved, select default.gpj, and click the [Open] button.

(3)   The source file output by the Smart Configurator is added to the workspace.
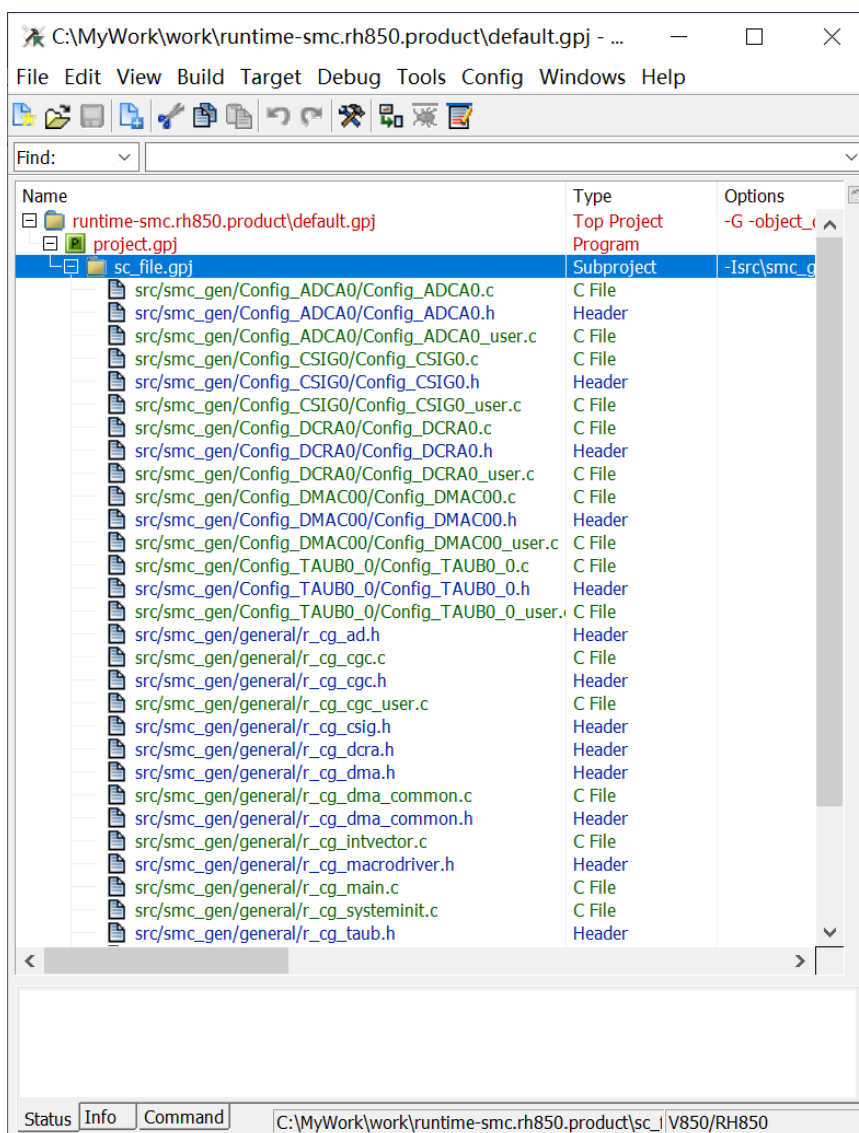
Figure 7-9. New Files Added to MULTI Workspace

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

When start Smart Configurator standalone and select All Toolchain (CC-RH, GHS, IAR) as toolchain, Smart Configurator outputs the project file (sc_file.gpj) together with the source files that can adapt to all three toolchains: CC-RH, IAR and GHS. When loading these generated files in MULTI workspace, the usage procedure is as follows.

(1)   Create a new project according to Project Wizard in MULTI workspace.

(2)   Right click on your MULTI project file(.gpj) and select [Add File into "projectname".gpj]:
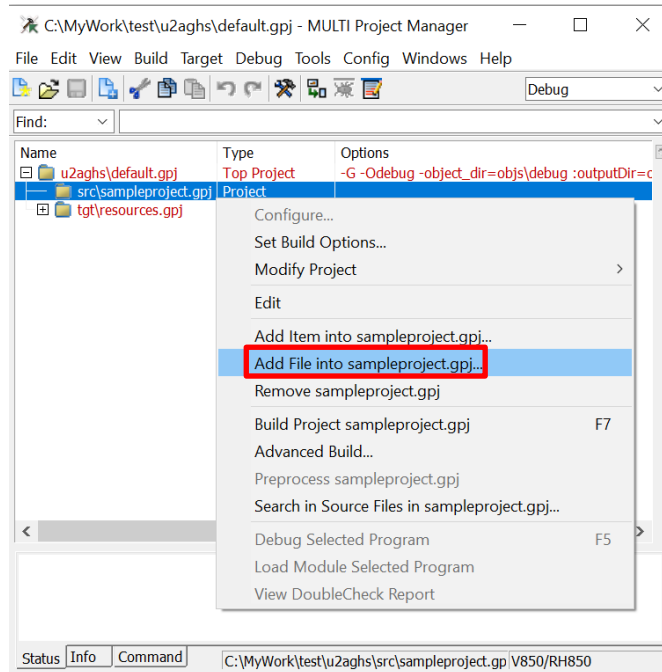
Figure 7-10. Add new files to MULTI Workspace

(3)   In the [Choose file(s) to add:] view, browser to the folder where Smart Configurator project file located, select file "sc_file.gpj", then click add, then all generated file by Smart Configurator will be added the MULTI workspace:
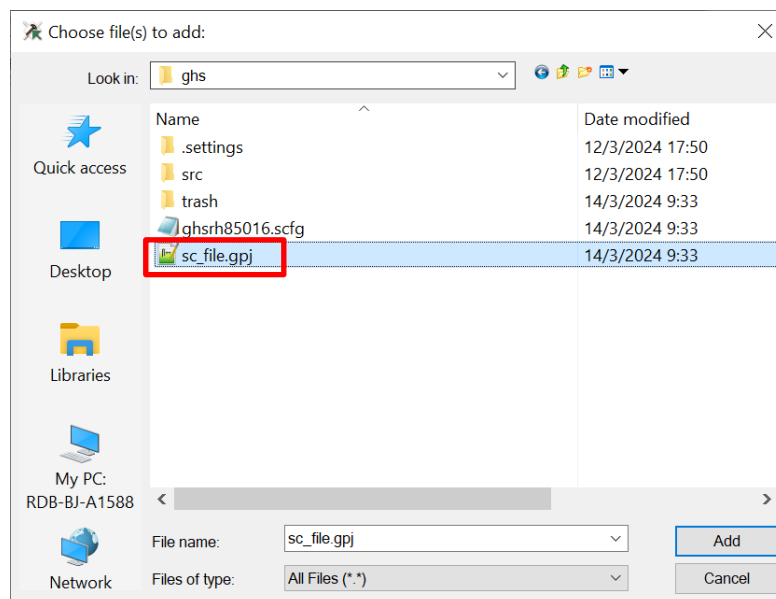
Figure 7-11. Select sc_file.gpj
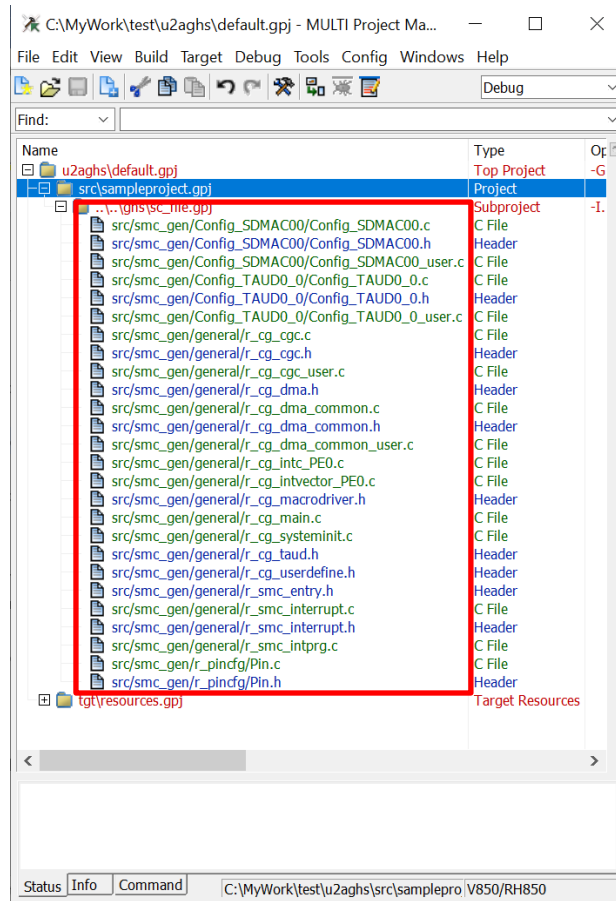
Figure 7-12. Select sc_file.gpj

(4)   Right click "xxx.gpj" file to select [Set Build Options…], then in popped [Build Options for "xxx.gpj"] to set [Include Directories] for file "iodefine.h" manually:
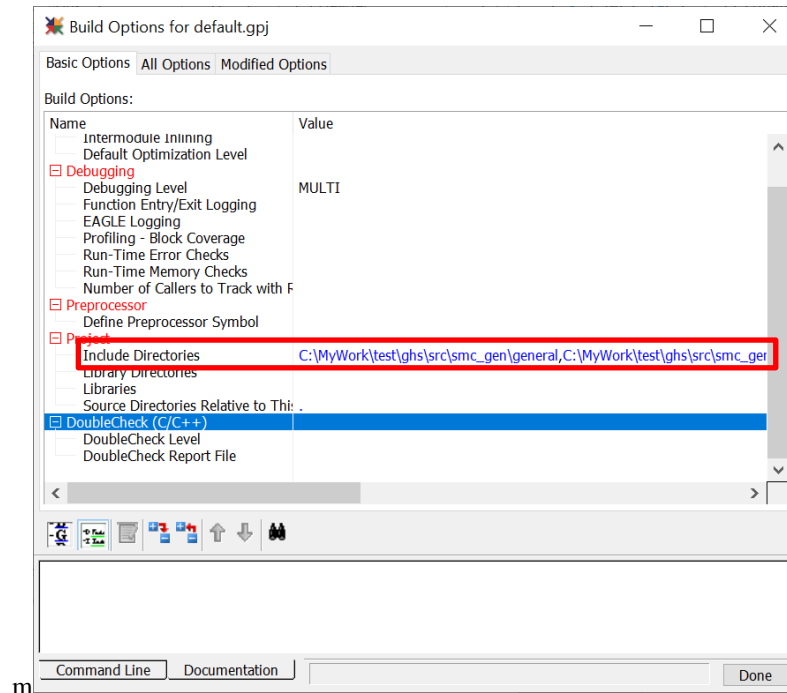
Figure 7-13. Set Include Directories

8.   Creating User Programs

Create a user program in the IDE. This chapter describes how to add custom code to the source file generated by the SC.

8.1   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 "TAUB1" for the compare-match timer (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 */
```

9.   Generating Reports

The Smart Configurator can output the configuration information of the project to the report. Follow the procedure below to generate a report.

9.1   Report on Configuration

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).
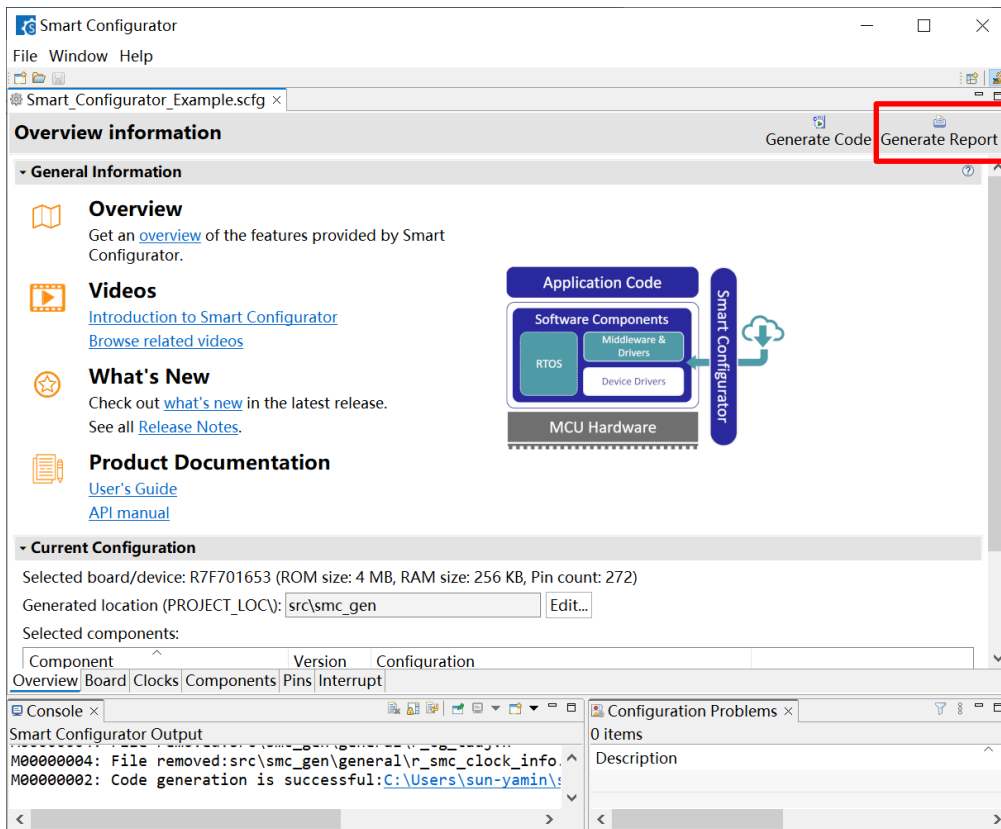
Figure 9-1 Output of a Report on the Configuration
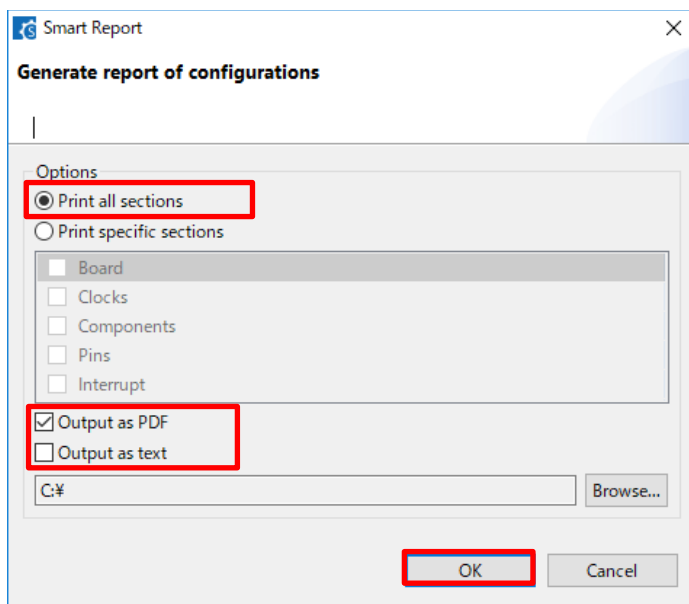
Figure 9-2 Dialog Box for Output of a Report (Example is selecting "Output as PDF")

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 [🖫 (Save the list to .csv file)] on the [Pins] page of the Smart Configurator view.
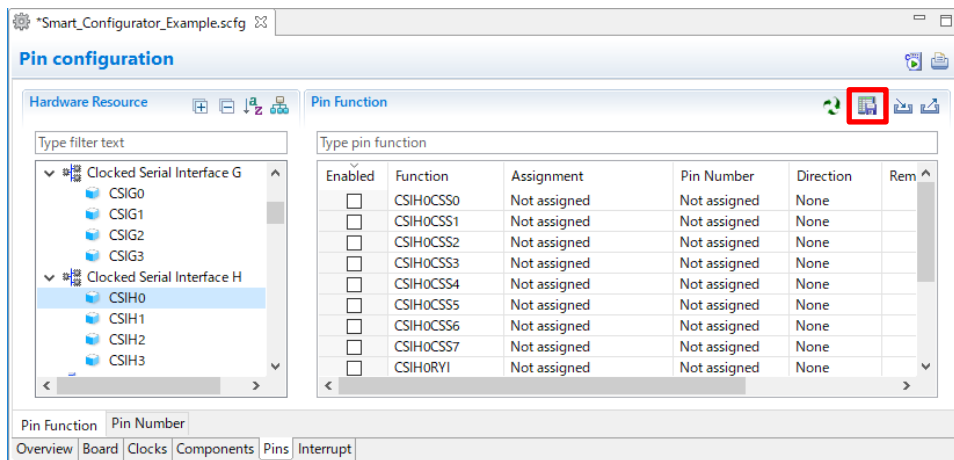
Figure 9-3 Output of a List of Pin Functions or Numbers (in csv Format)

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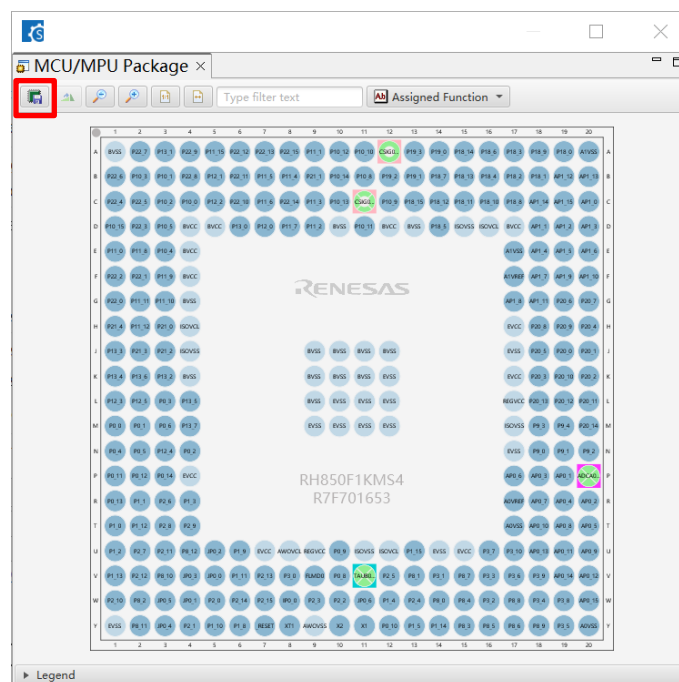
Figure 9-4 Outputting a Figure of MCU/MPU Package (in png Format)

10. User code protection feature

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
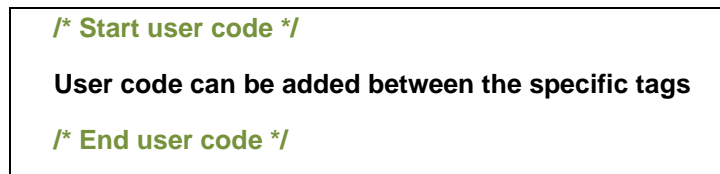
/* End user code */

Figure 10-1 Specific tags for user code protection feature

10.2   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 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 new generated file.

```c
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 */                      Inserted the user code
    TAUB1.CMOR0 = 0x80;                        with the specific tags
    /* 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;
```

Generate code again, the code protected by user code would not be changed.

```c
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_15;
    /* Start user code */                      User codes will automatically be
    TAUB1.CMOR0 = 0x80;                        merged into the new generated file
    /* 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;
```
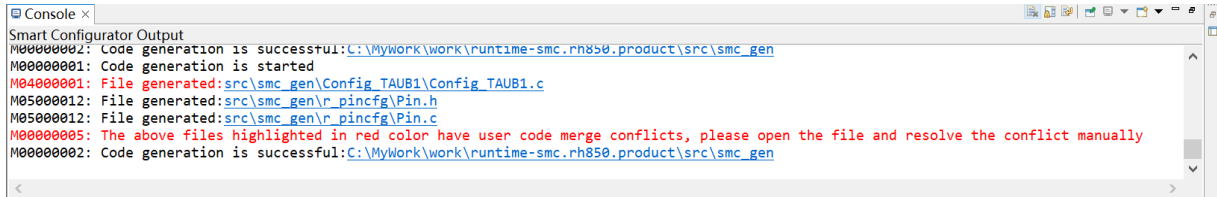
Figure 10-2 User code protection with auto merge

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.

If the merge conflict occurs, conflict message in red will be displayed in the Smart Configurator console, as shown in Figure 10-3 The merge conflict message outputted in the Smart Configurator console.

Figure 10-3 The merge conflict message outputted in the Smart Configurator console

10.3.2   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 10-4 Code before resolving conflict).
2) Click on "Copy Current Change from Left to Right" (Figure 10-4 Code before resolving conflict).
3) Delete the codes that you do not want to use (Figure 10-5 Code after applying "Copy Current Change from Left to Right").
4) Save the modified code (Figure 10-6 Code after deleting and saving).
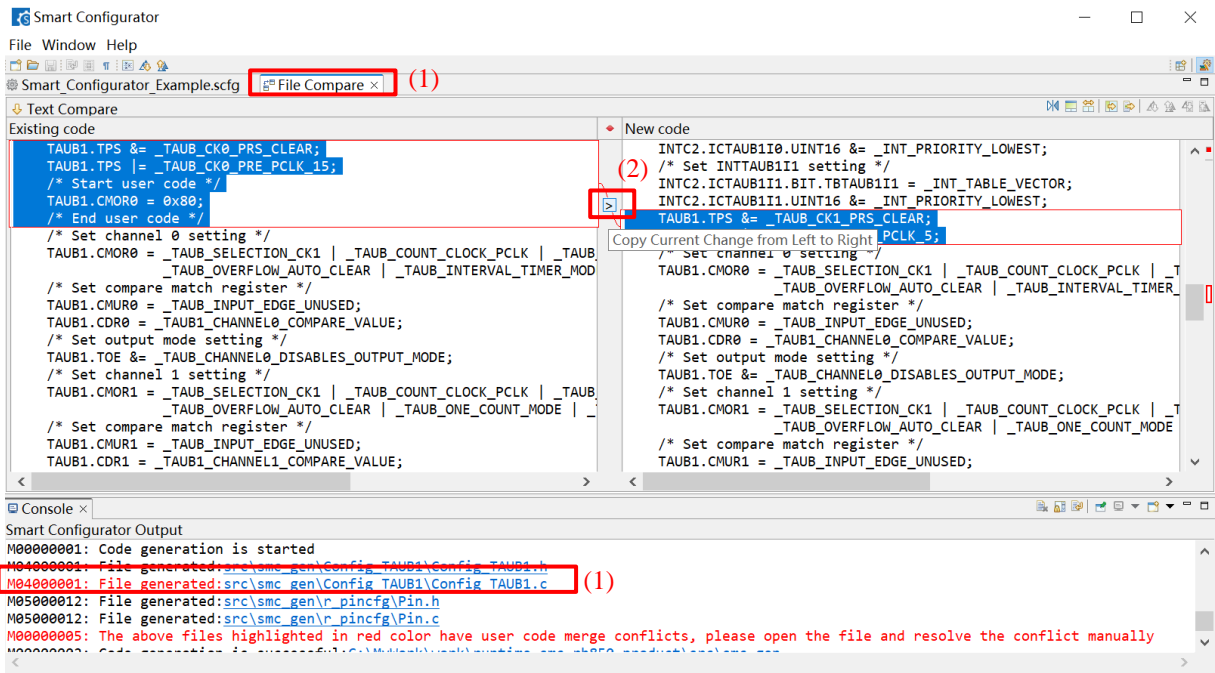
Figure 10-4 Code before resolving conflict
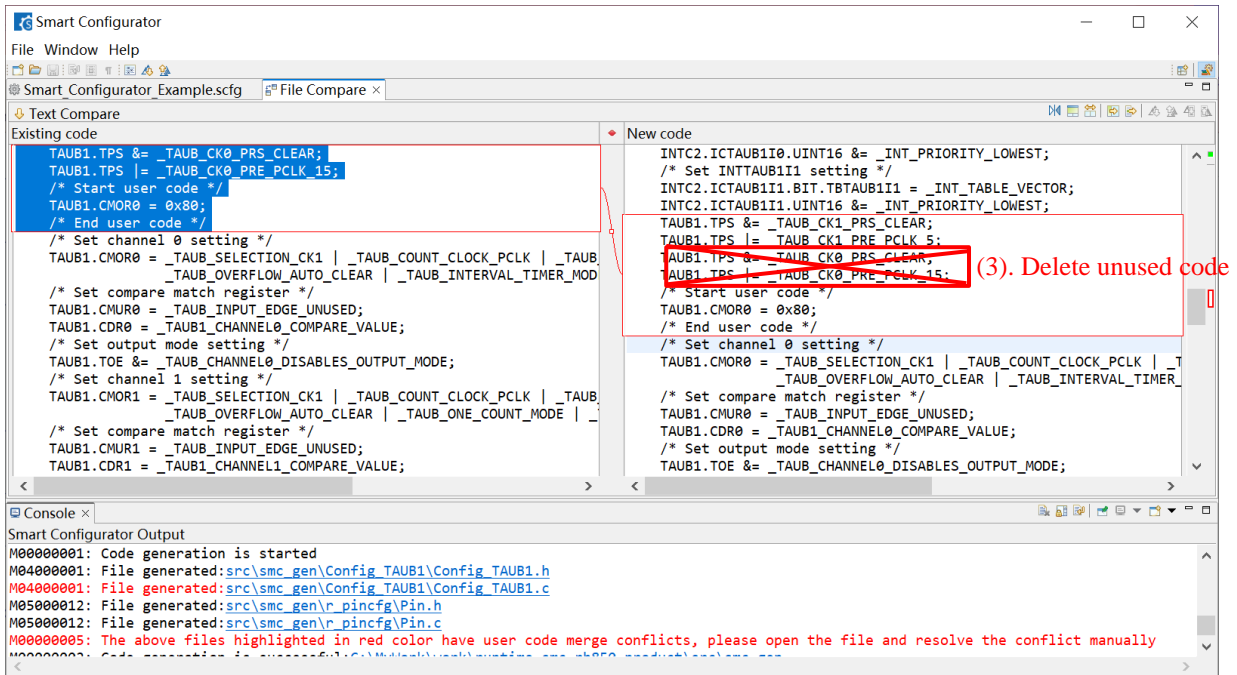
Figure 10-5 Code after applying "Copy Current Change from Left to Right"
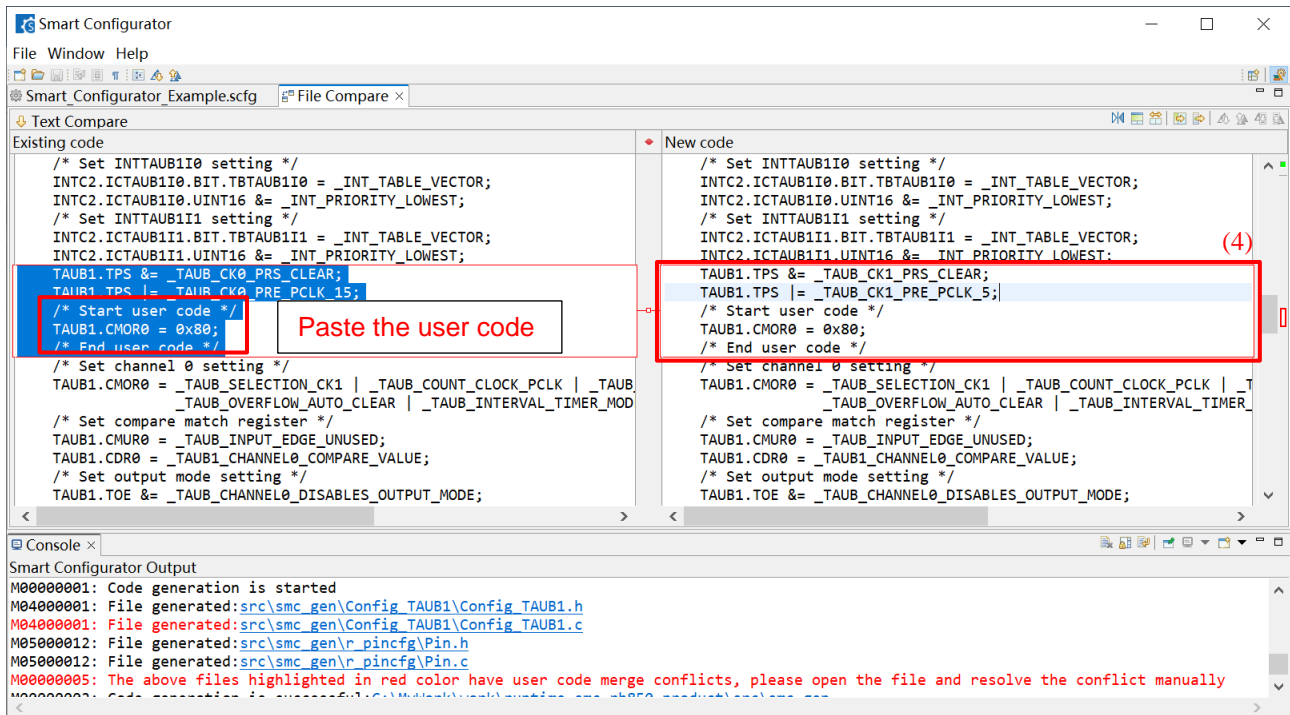
Figure 10-6 Code after deleting and saving

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

11. Help

Refer to the help system for detailed information on the Smart Configurator.
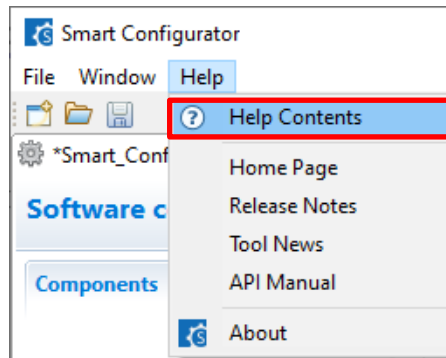
Figure 11-1 Help Menu

The help system can also be activated from the [Overview] page.
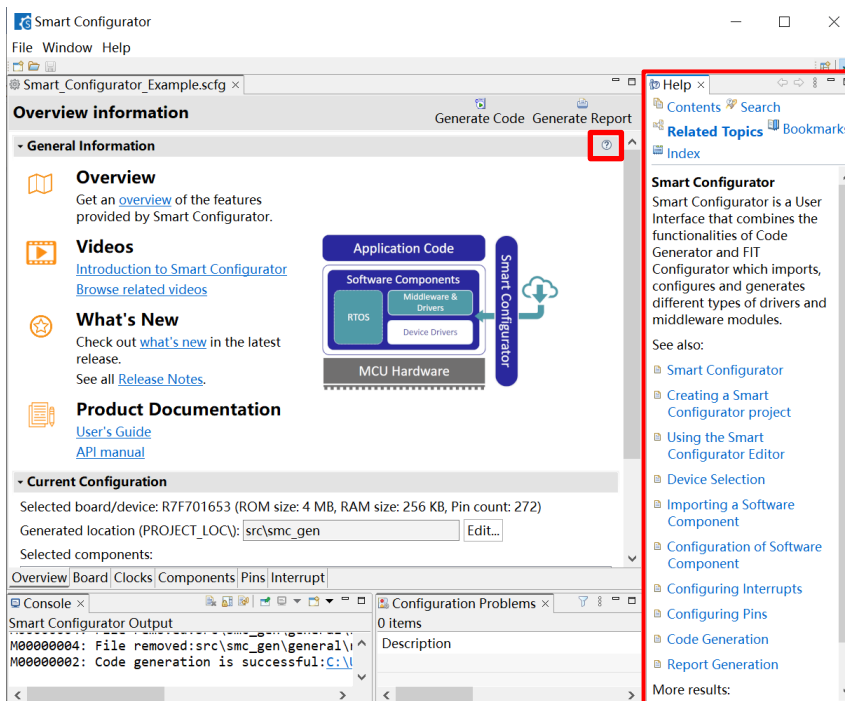
Figure 11-2 Quick Start

12. Documents for Reference

User's Manual: Hardware

Obtain the latest version of the manual from the web site of Renesas Electronics.

Technical Update/Technical News

Smart Configurator: Guide on Sample Project for RH850/F1KM Device (R01AN4422)

User's Manual: Development Environment

Obtain the latest version of the manual from each company web site.

Website and Support

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

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

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

Revision History

Rev	Section	Description
1.00	-	First edition issued
1.01	4.3.6 Export Component Configuration	New added.
	4.3.7 Import Component Configuration	New added.
	4.3.8 Configure General Setting of Component	New added.
	4.5.2 Changing the Interrupt Handler Name and Generate Entity Setting	New added
	6.2 Configuration of Generated Files and File Names	Updated.
	7.1 Loading in IAR Embedded Workbench	Updated.
1.02	3.1 Procedure for Operations	Figure 3-1 Operating Procedure updated.
	3.3.1 Creating a New Configuration File	Figure 3-3 Create a Configuration File updated.
	3.4.4 MCU/MPU Package View	Description and figure updated.
	4.1.1 Selecting the Device	Figure 4-3 Confirm Device Change before saving new added.
	4.1.2 Selecting the Board	The "Note" is new added.
	4.3 System Settings (only for RH850/U2A)	New added.
	4.4.3 Removing Component	Add content about removing multiple components.
	4.4.8 Configure General Setting of Component	Figure and description updated.
	4.5 Pin settings	Figure 4-30 [Pins] Page ([Pin Number]) updated
	4.5.2 Assigning pins using the MCU/MPU Package view	Figure and description updated.
	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
	4.6.3 Changing the PEn setting	New added
	6.2 Configuration of Generated Files and File Names	Figure and Table added below three new items: R_cg_xxx_common_user.c R_smc_entry.h r_cg_intc_PEn.c
	6.5 Initializing Interrupts	Figure 6-6 Interrupts Configuration in Interrupts View 2 new added. Table 6-6 Interrupt Generation File Description for RH850/U2A, RH850/C1M new added
	7.2 Loading in GHS MULTI	Figure 12-1. New Files Added to MULTI Workspace New added
	9.1 Report on Configuration	Description and Figure name changed.
	9.3 Image of MCU/MPU Package (in png Format)	Figure 9-4 Outputting a Figure of MCU/MPU Package (in png Format) updated.
	10 User code protection feature for Smart Configurator Code Generation component	New added
	11 Help	Figure 11-2 Quick Start updated.

Rev	Section	Description
1.03	Introduction	Update to RH850 Smart Configurator V1.10.0
	3.4 Window 9.1 Report on Configuration 11. Help	Update: Figure 3-5 Main Window Figure 3-6 Smart Configurator View Figure 9-1 Output of a Report on the Configuration Figure 11-2 Quick Start
	4.6.2 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.
1.04	3.3.1 Creating a New Configuration File	Add Common Toolchain (CC-RH, GHS, IAR) description
	4.6.2 Changing the Interrupt Handler Name, Generate Entity and Generate Enable/Disable Function Setting	Add detailed interrupt handler name rule
	7.1.2 Loading files generated by All Toolchain (CC-RH, GHS, IAR)	New added
	7.2.2 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/.