

RH850 Smart Configurator

R20AN0739EC0100

Rev.1.00

Jan.22.24

User's Guide: e² studio

Introduction

This application note describes the basic usage of the RH850 Smart Configurator (hereafter called the Smart Configurator), which is an e² studio plug-in tool.

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

- e² studio 2024-01 and later

Target Devices and Compilers

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

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

Contents

1. Overview	4
1.1 Purpose	4
1.2 Features	4
2. Creating a Project.....	4
3. Operating the Smart Configurator.....	8
3.1 Displaying the Smart Configurator Perspective	8
3.2 Procedure for Operations	9
3.3 File to be Saved as Project Information	10
3.4 Window.....	10
3.4.1 Project Explorer.....	11
3.4.2 Smart Configurator View	11
3.4.3 MCU/MPU Package View	12
3.4.4 Console View	13
3.4.5 Configuration Problems View.....	13
4. Setting of Peripheral Modules.....	14
4.1 Board Settings.....	14
4.1.1 Selecting the Device	14
4.1.2 Selecting the Board.....	15
4.1.3 Exporting Board Settings	17
4.1.4 Importing Board Settings	17
4.2 Clock Settings	18
4.3 System Settings (only for RH850/U2A).....	19
4.4 Component Settings.....	23
4.4.1 Switching Between the Component View and Hardware View	23
4.4.2 Adding a Software Component	24
4.4.3 Removing Software Component	26
4.4.4 Setting a Code Generator Component	27
4.4.5 Changing the Resource for a Code Generator Configuration	28
4.4.6 Export Component Configuration.....	31
4.4.7 Import Component Configuration.....	31
4.4.8 Configure General Setting of the Component.....	32
4.5 Pin Settings	33
4.5.1 Changing the Pin Assignment of a Software Component	34
4.5.2 Assigning Pins Using the MCU/MPU Package View	35
4.5.3 Show Pin Number from Pin Functions	36
4.5.4 Exporting pin settings.....	37
4.5.5 Importing pin settings	38
4.5.6 Pin Setting Using Board Pin Configuration Information	39
4.5.7 Pin Filter Feature.....	40
4.5.8 Pin Errors/Warnings setting	41
4.6 Interrupt Settings	42
4.6.1 Changing Interrupt Priority Level and OS Management Setting.....	43
4.6.2 Changing The PE _n Setting (RH850/U2A Only)	44

4.6.3	Changing Interrupt Handler Name, Generate Entity and Generate Enable/Disable Function Setting	45
4.7	MCU Migration Feature	47
5.	Managing Conflicts	51
5.1	Resource Conflicts	51
5.2	Resolving Pin Conflicts	52
6.	Generating Source Code	53
6.1	Outputting Generated Source Code	53
6.2	Change Generated Code Location	55
6.3	Configuration of Generated Files and File Names	57
6.4	Initializing Clocks	61
6.5	Initializing Pins	62
6.6	Initializing Interrupts	63
7.	Creating User Programs	64
7.1	Adding Custom Code	64
7.2	Using Generated Code in User Application	65
8.	Backing up Generated Source Code	66
9.	Generating Reports	67
9.1	Report on All Configurations (PDF or Text File)	67
9.2	Configuration of Pin Function List and Pin Number List (in csv Format)	69
9.3	Image of MCU/MPU Package (in png Format)	69
10.	User Code Protection Feature for Smart Configurator Code Generation Component	70
10.1	Specific Tags for the User Code Protection Feature	70
10.2	Examples of Using User Code Protection Feature to Add New User Code	70
10.3	What to Do When Merge Conflict Occurs	72
10.3.1	What is Merge Conflict	72
10.3.2	Steps for Resolving the Merge Conflict	73
11.	Help	75
12.	Documents for Reference	76
	Corporate Headquarters	1
	Contact information	1
	Trademarks	1

1. Overview

1.1 Purpose

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

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

1.2 Features

The Smart Configurator is a utility for combining software to meet your needs. It handles the following three functions to support the embedding of drivers from Renesas in your systems: importing middleware in the form of SW integration feature, generating driver code, and making pin settings.

2. Creating a Project

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

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

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

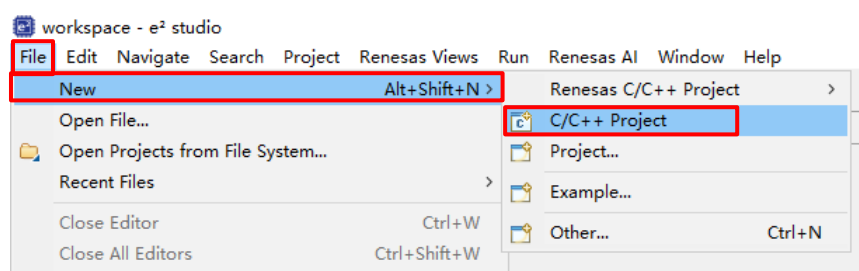


Figure 2-1 Creating a New Project

- (2) In the project creation wizard, select [Renesas RH850] → [Renesas CC-RH C Executable Project] and click on the [Next] button.

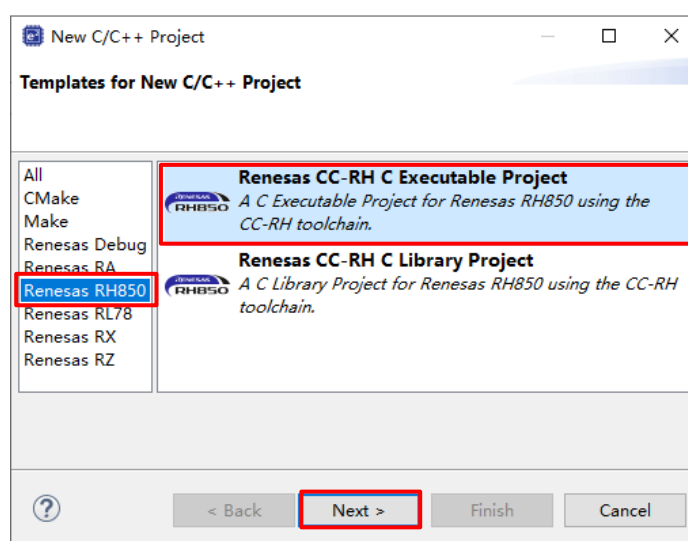


Figure 2-2 Templates for New C/C++ Project Dialog Box

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

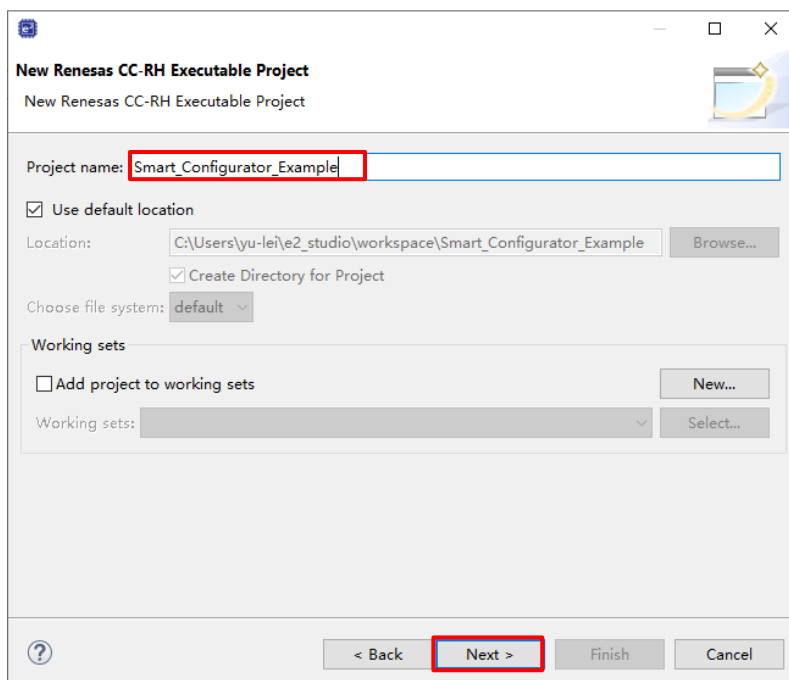


Figure 2-3 Creating a New Renesas CC-RL Executable Project

- (4) Select the toolchain, target board, device, and debug configuration. Click on the [Next] button.
(For e.g., Target Device: RH850F1KM – 272 pins (Part number: R7F701653))

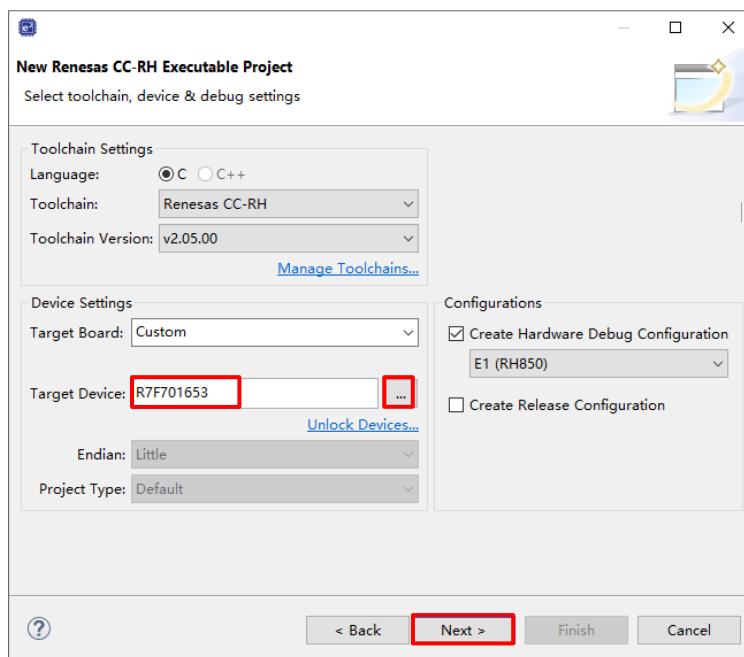


Figure 2-4 Selecting the Toolchain, Device, and Debug Configuration

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

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

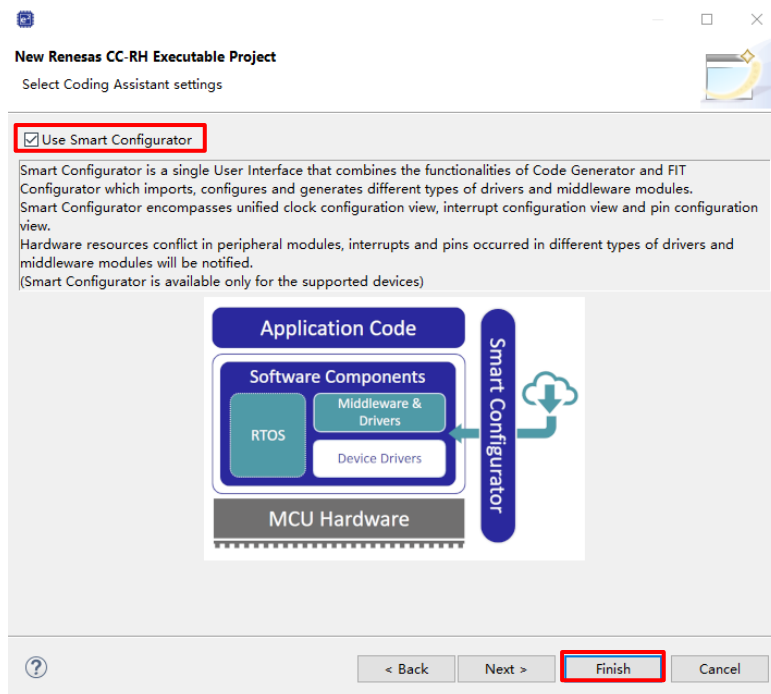


Figure 2-5 Selecting the Coding Assistant Tool

- (6) Wait for completion of project creation.

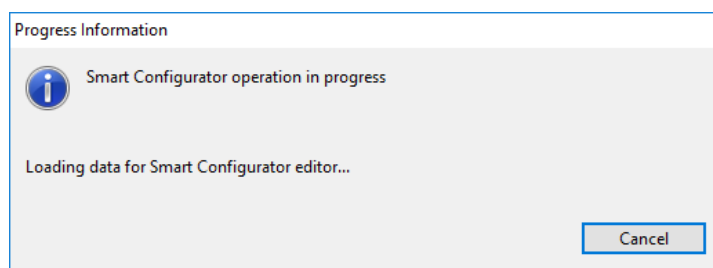


Figure 2-6 Progress of Project Creation

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

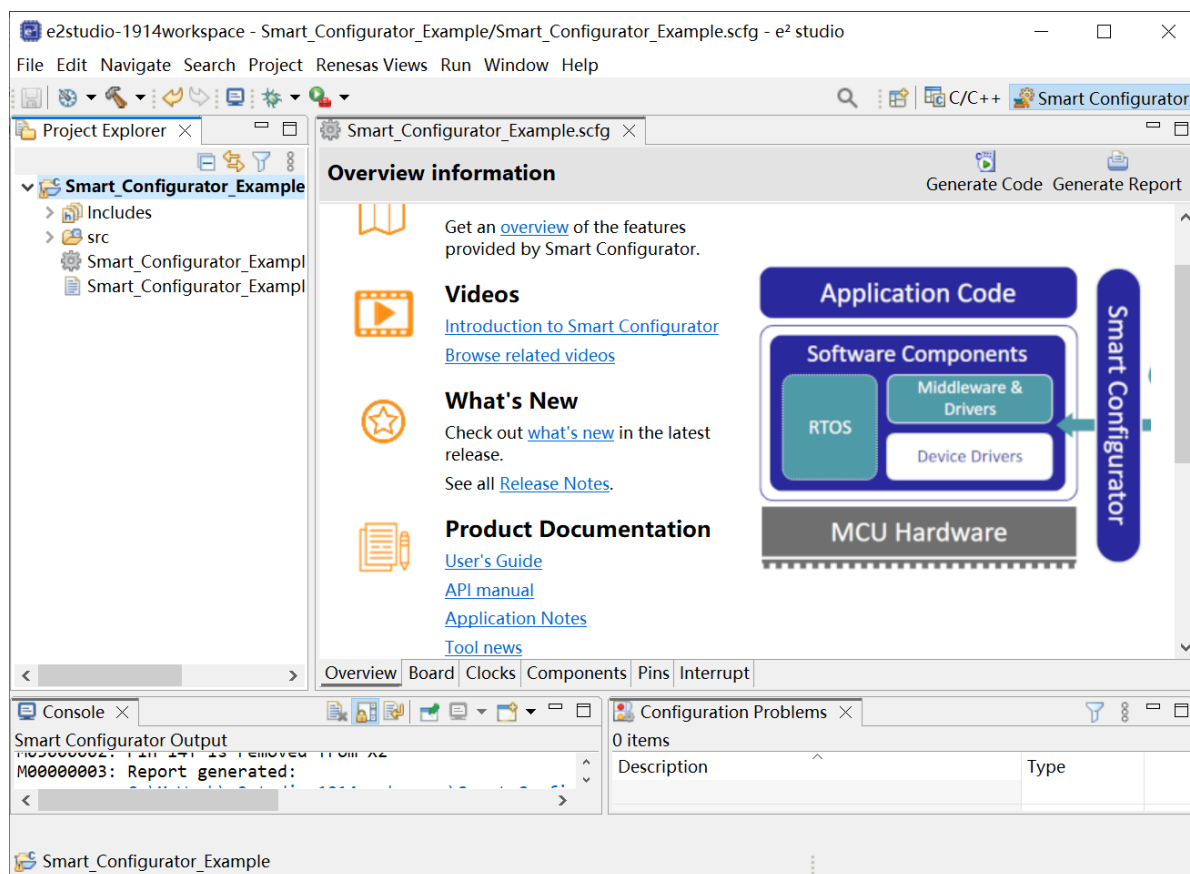


Figure 2-7 Smart Configurator Perspective

3. Operating the Smart Configurator

3.1 Displaying the Smart Configurator Perspective

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

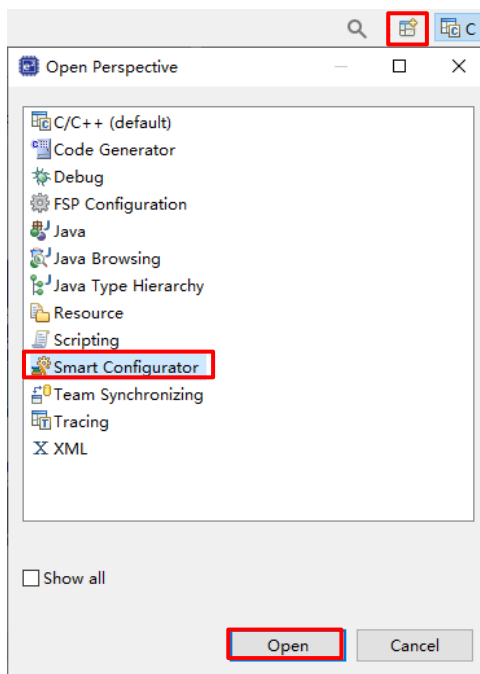


Figure 3-1 Opening the Smart Configurator Perspective

3.2 Procedure for Operations

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

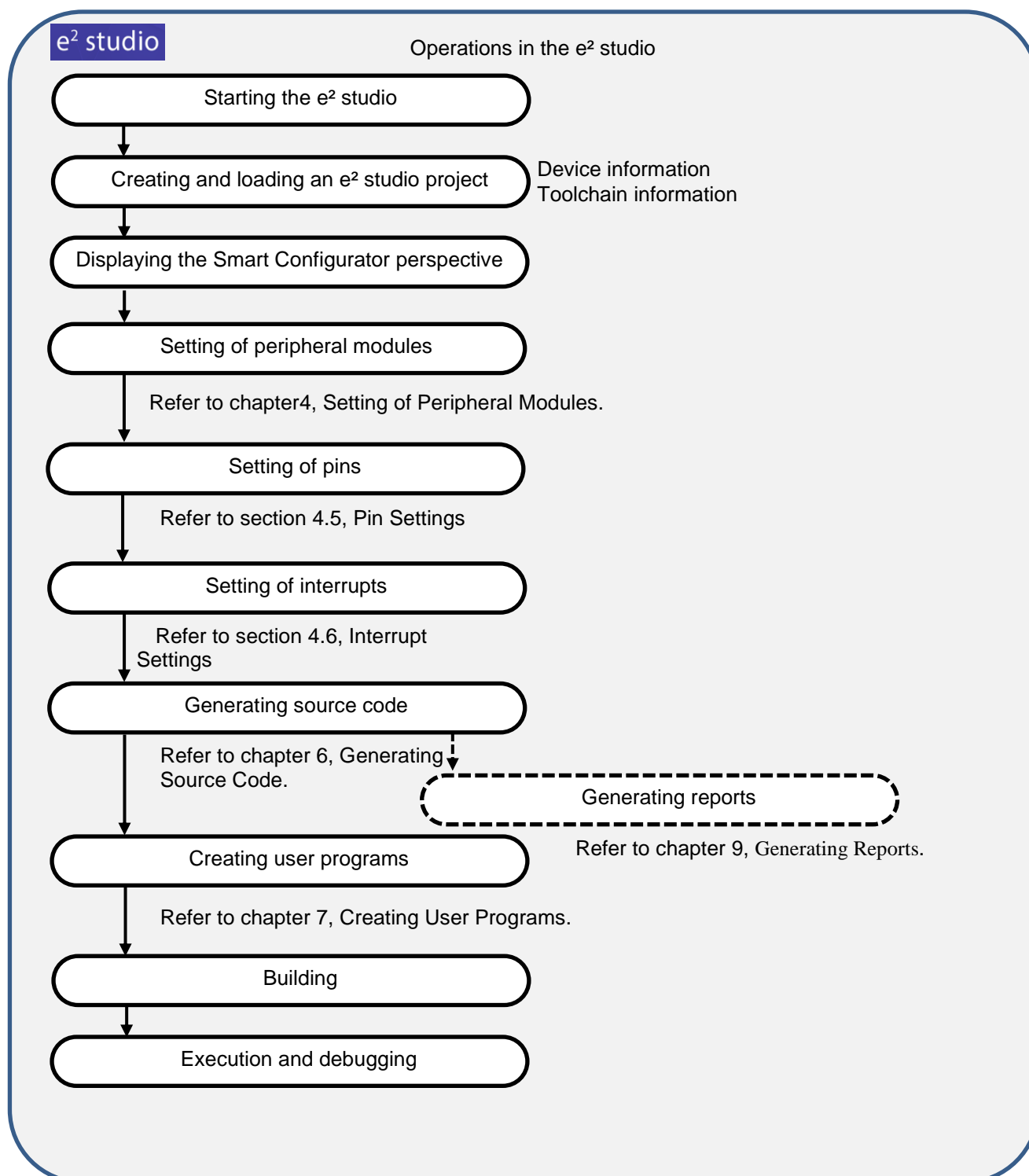


Figure 3-2 Procedure for Operations

3.3 File to be Saved as Project Information

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

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

3.4 Window

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

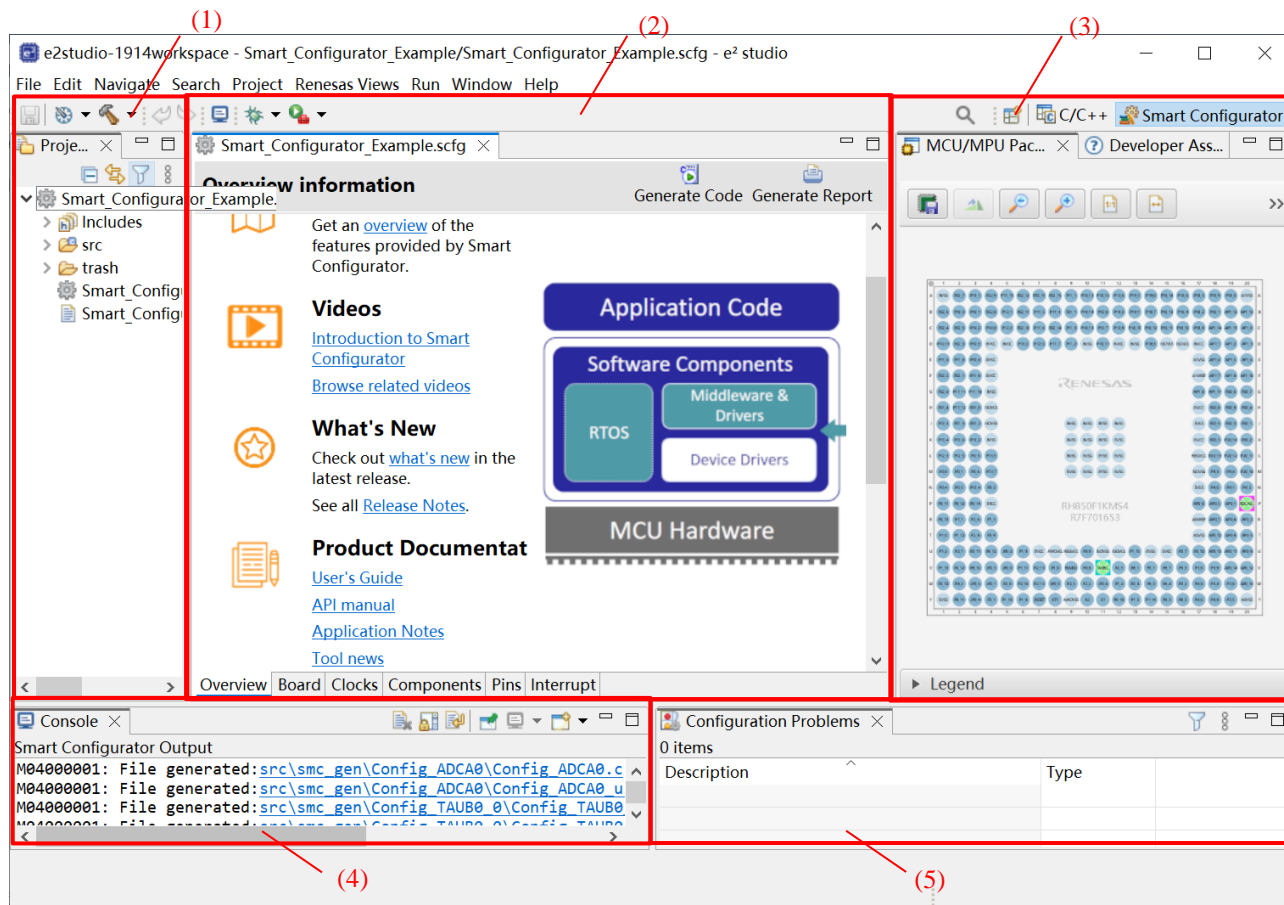


Figure 3-3 Smart Configurator Perspective

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

3.4.1 Project Explorer

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

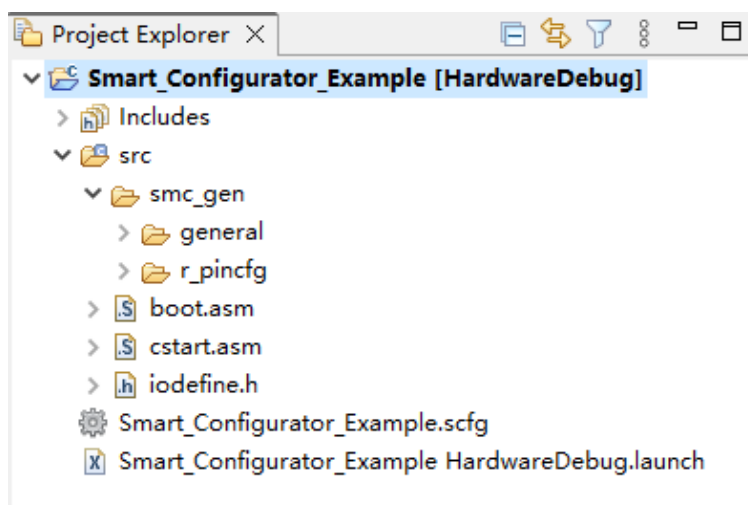


Figure 3-4 Project Explorer

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

3.4.2 Smart Configurator View

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

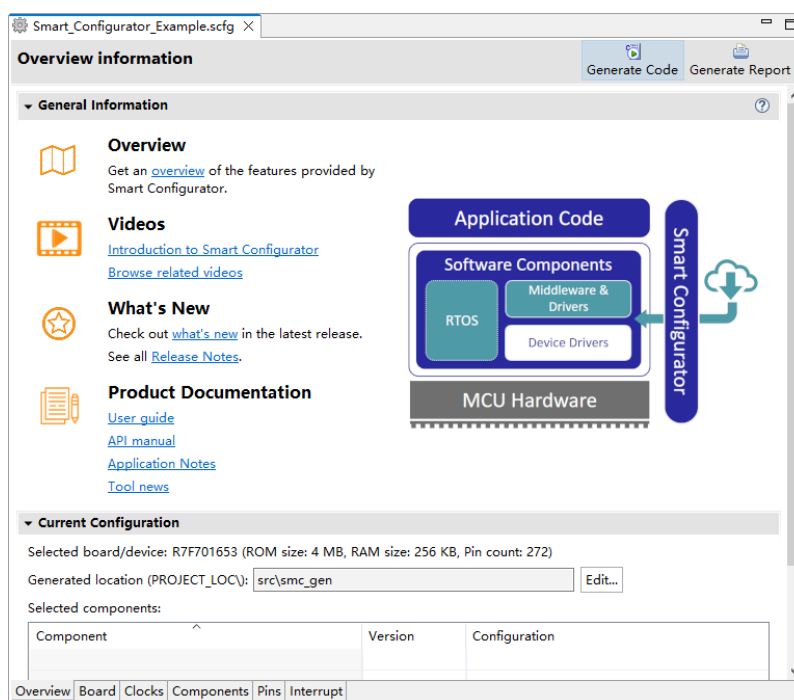


Figure 3-5 Smart Configurator View

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

3.4.3 MCU/MPU Package View

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

Three types of package view can be switched among [Assigned Function], [Board Function] and [Symbolic Name].

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

The initial pin setting information of the board is the pin information of the board selected by [Board:] on the [Board] page (refer to "chapter 4.1 Board Settings" and "chapter 4.5.6 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.

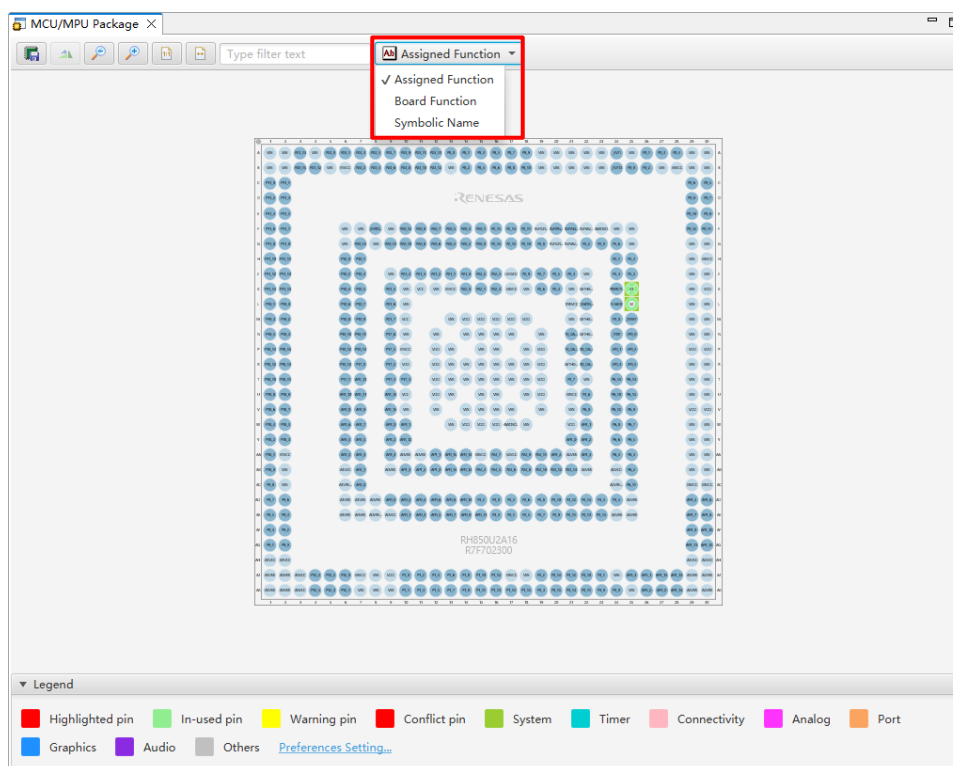


Figure 3-6 MCU/MPU Package View

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

3.4.4 Console View

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

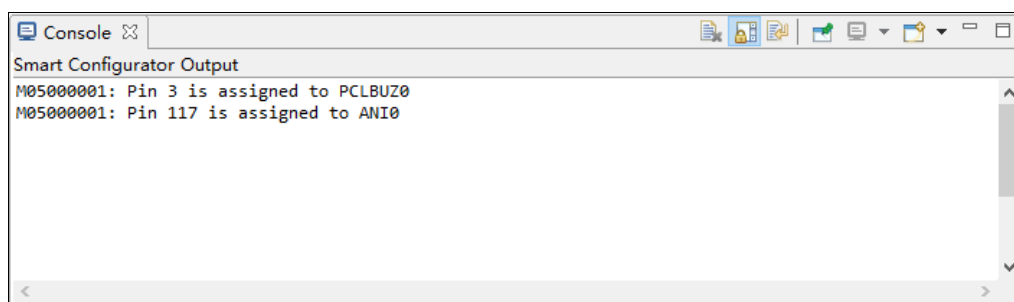


Figure 3-7 Console View

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

3.4.5 Configuration Problems View

The Configuration Problems view displays the details of conflicts between driver used interrupts, configured peripherals, used pins, used settings.

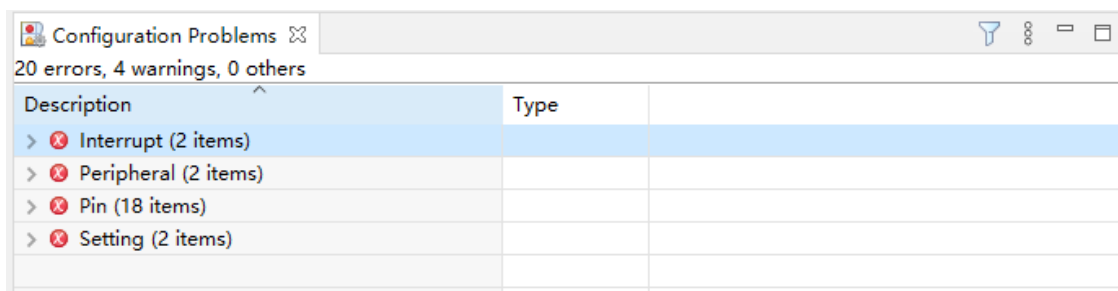


Figure 3-8 Configuration Problems View

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

4. Setting of Peripheral Modules

User can select peripheral modules from the Smart Configurator view.

4.1 Board Settings

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

4.1.1 Selecting the Device

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

Follow the procedure of "4.7 MCU Migration Feature" to change the device.

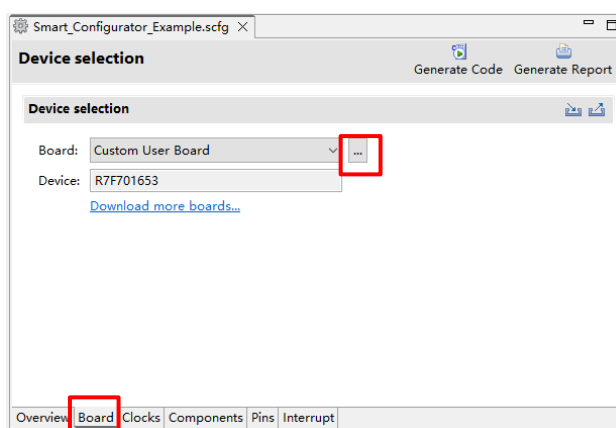



Figure 4-1 Selecting the Device

4.1.2 Selecting the Board

Click on the [] button to select a board.

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

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

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

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

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

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

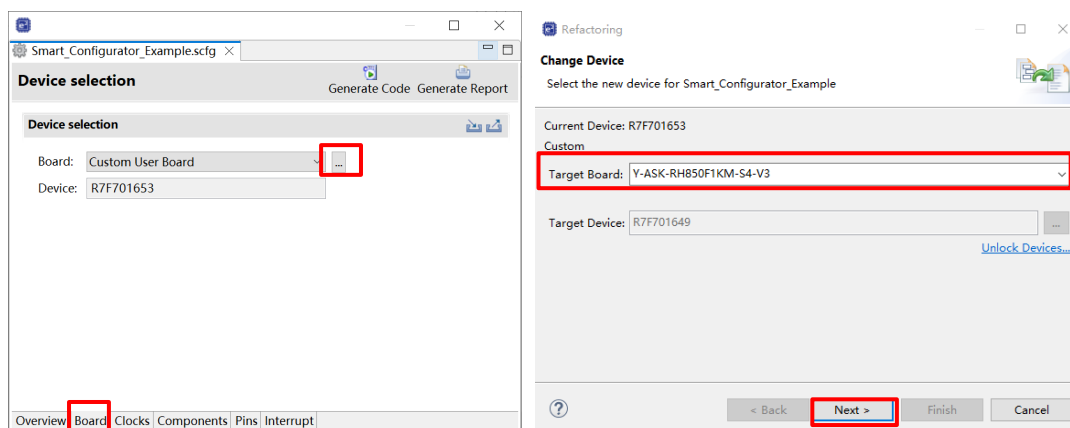


Figure 4-2 Selecting the Board

Confirm the message displayed in [Discovered Issues] and click [Next].

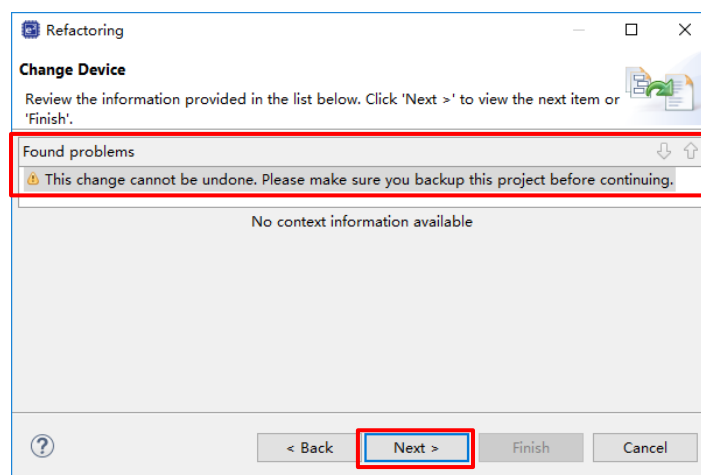
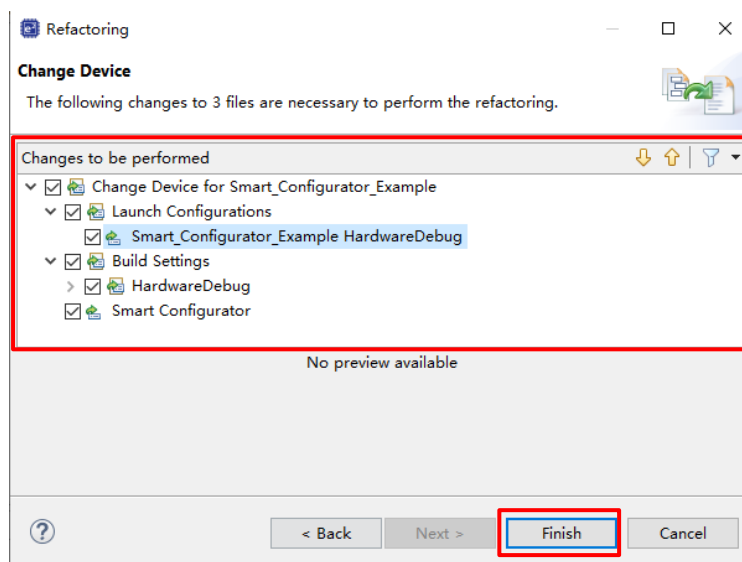


Figure 4-3 Found Problems

Table 4-1 Change Device Found Problems List


Message	Explanation
This change cannot be undone. Please make sure you backup this project before continuing.	If you change the device, it can't be restored before change, so please execute it after backing up the project.

Confirm items to be changed and click “Finish”.

**Figure 4-4 Changes to be Performed**

4.1.3 Exporting Board Settings

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

- (1) Click on the [ (Export board setting)] button on the [Board] page.
- (2) Select the output location and specify a name (Display Name) for the file to be exported.

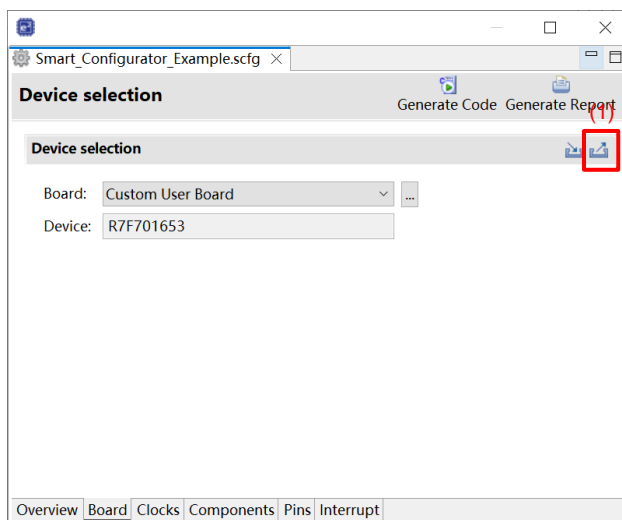



Figure 4-5 Exporting Board Settings (bdf Format)

4.1.4 Importing Board Settings

Follow the procedure below to import board settings.

- (1) Click on the [ (Import board setting)] button and select a desired bdf file.
- (2) The board of the imported settings is added to the board selection menu.

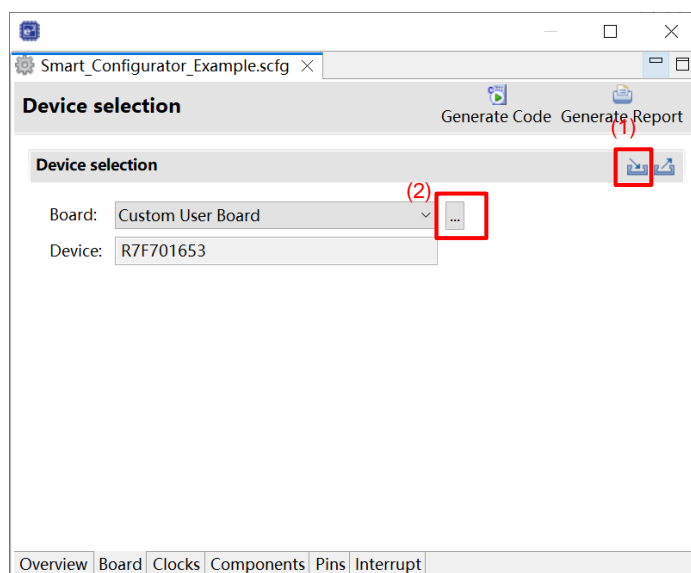


Figure 4-6 Importing Board Settings (bdf Format)

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

4.2 Clock Settings

User can set the system clock on the [Clocks] page. The settings made on the [Clocks] page is used for all drivers.

Follow the procedure below to modify the clock settings.

- (1) Specify the frequency of each clock in accordance with the board specifications (Note that the frequency is fixed for some internal clocks).
- (2) When using the PLL circuit, select the clock source for the PLL.
- (3) For the multiplexer symbol, select the clock source for the output clocks.
- (4) Enable the specific clock (only for RH850/F1KM and RH850/F1KH)
- (5) To obtain a desired output clock frequency, select a frequency division ratio from the drop-down list.

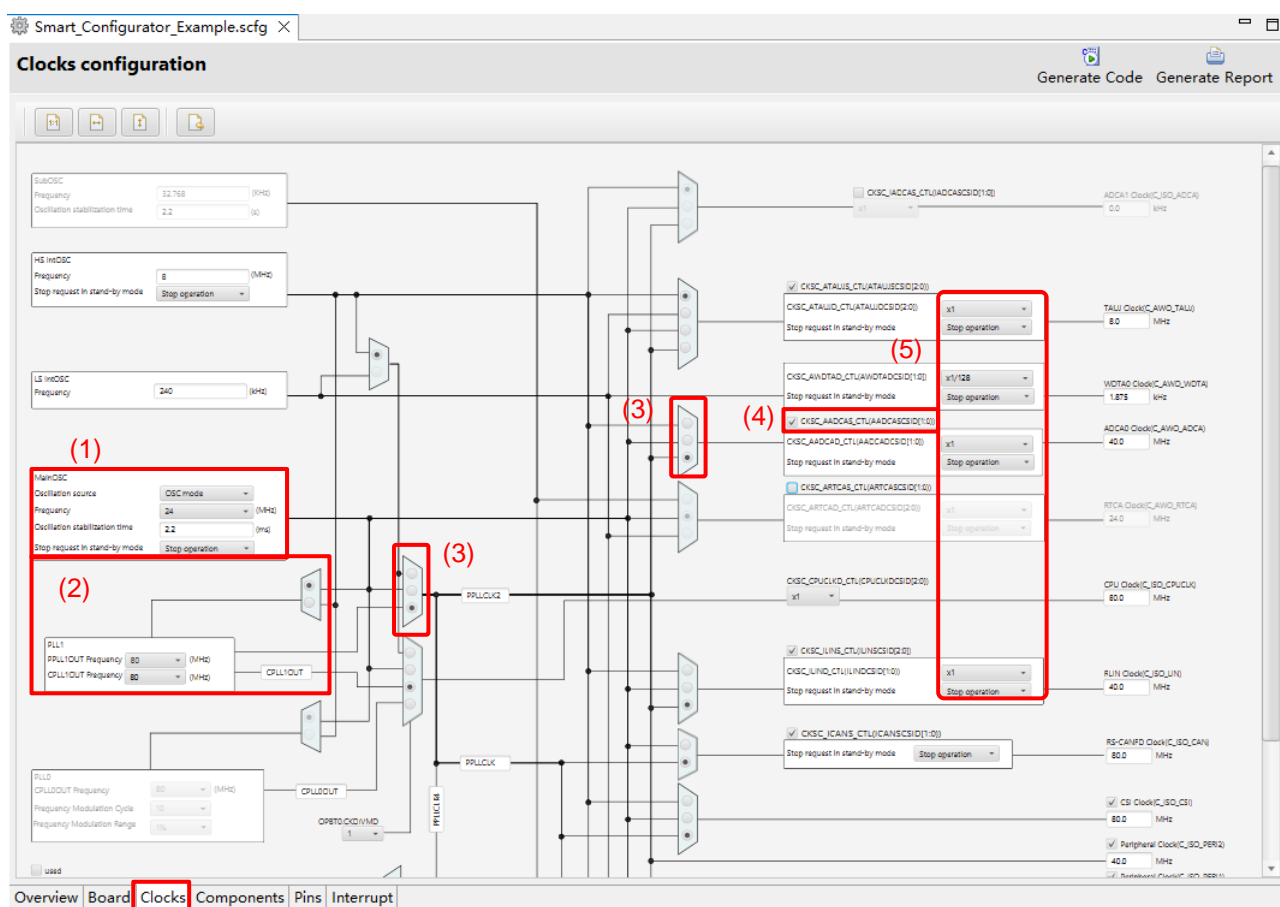


Figure 4-7 [Clocks] Page

4.3 System Settings (only for RH850/U2A)

User can select the CPU_n (PE_n) 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.

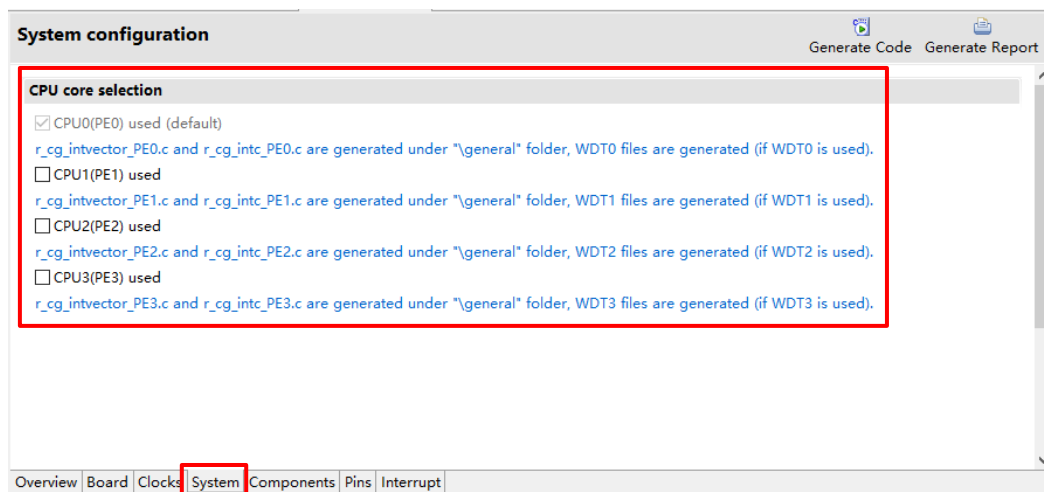


Figure 4-8 [System] Page

For example, if user plans to use CPU0(PE0) and CPU1(PE1), the following setting needs to be made.

- (1) Clicks on the [System] page of Smart Configurator, select CPU0(PE0) and CPU1(PE1). Generate code.

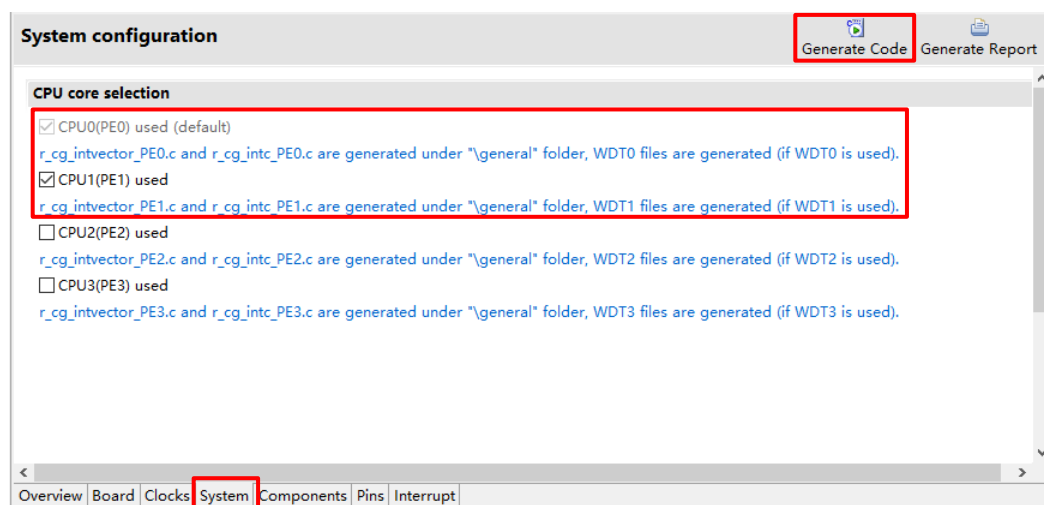


Figure 4-9 Select CPU core usage setting

- (2) Copy “cstart_pm0.asm” file as the new file “cstart_pm1.asm”.

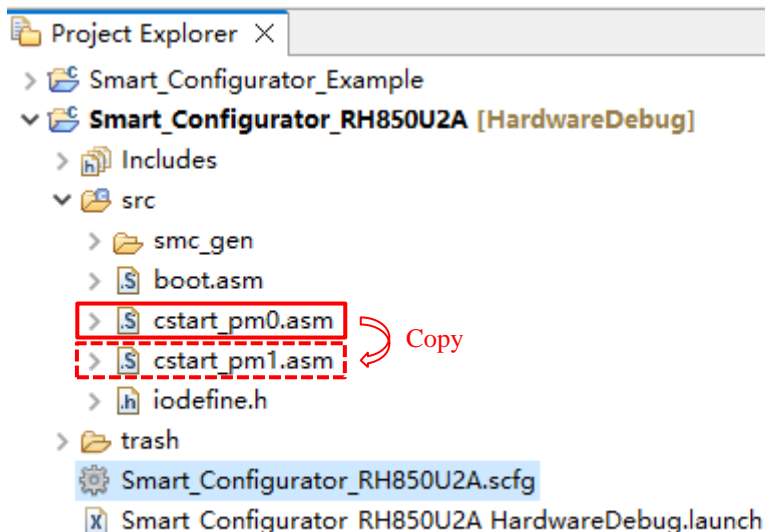


Figure 4-10 e2 studio copy cstart_pm0.asm file

- (3) Modify “cstart_pm1.asm” file contents as shown below.

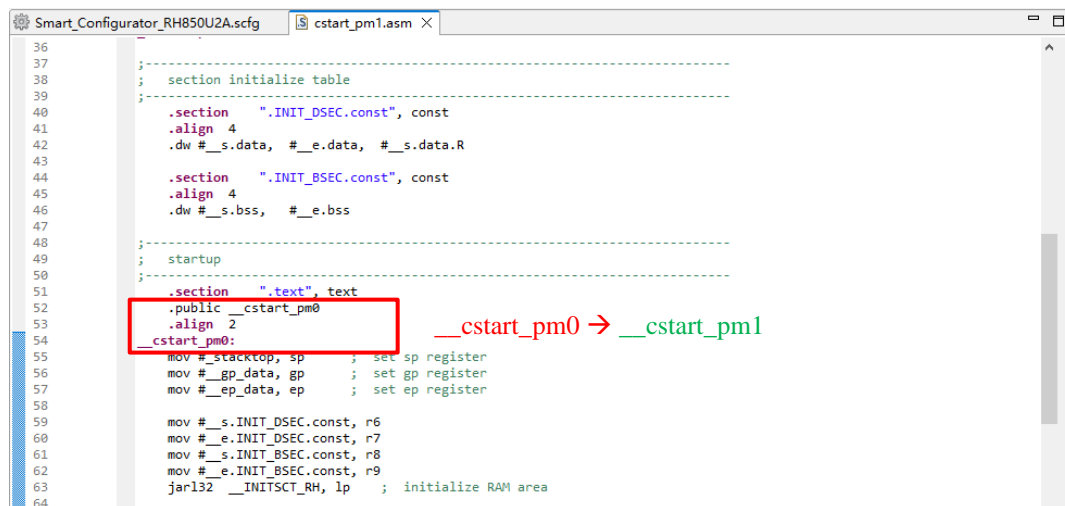


Figure 4-11 Modify cstart_pm1.asm file contents

(4) Modify “boot.asm” file contents as shown below.

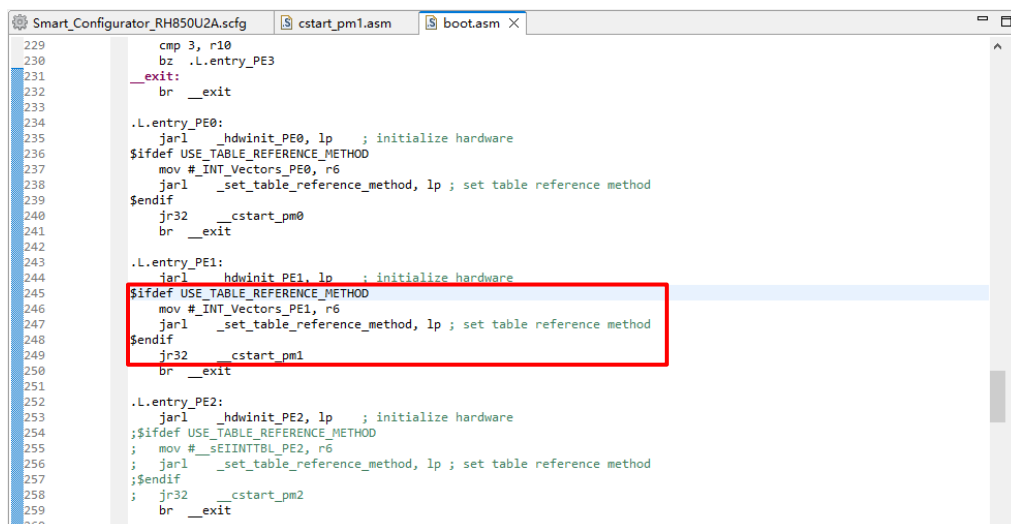


Figure 4-12 Modify boot.asm file contents

(5) Add section setting for PE1.

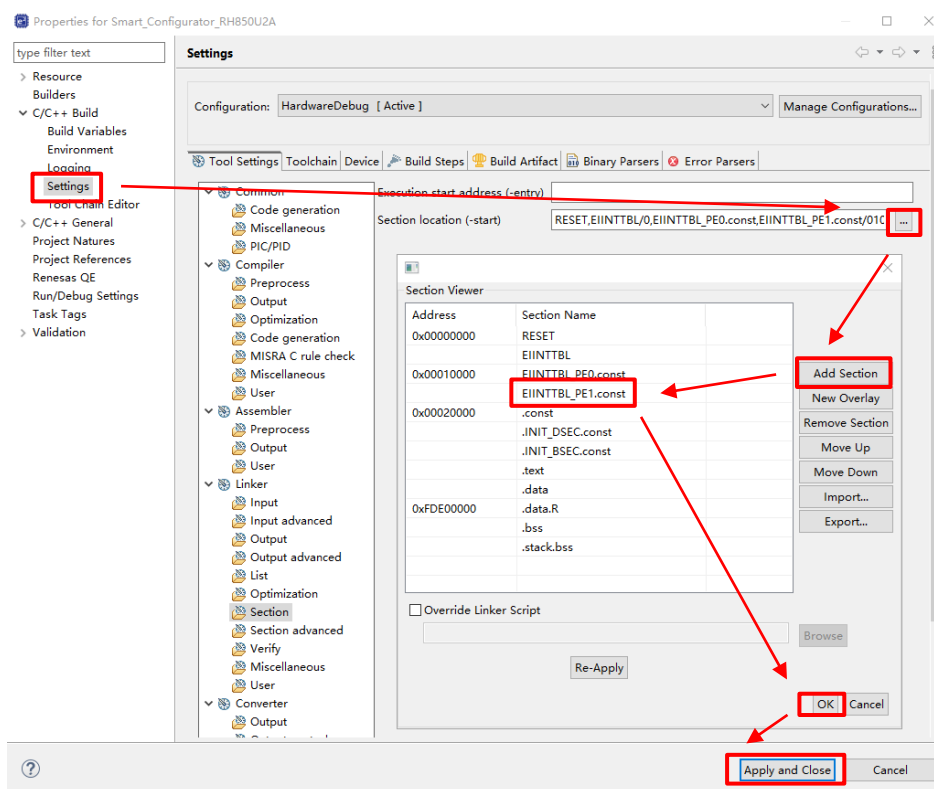


Figure 4-13 Add section setting for PE1

- (6) Rename API “R_Interrupt_Initialize_ForPE” as “R_Interrupt_Initialize_ForPE1” in r_cg_intc_PE1.c file.

```

2
19
21
27
29
30
31
33
35
36
37
38
39
41
43
44
45
46
+ * DISCLAIMER
+ * File Name      : r_cg_intc_PE1.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"
+ /* Start user code for include. Do not edit comment generated here */
+ /* End user code. Do not edit comment generated here */
+
+ Global variables and functions
+ void R_Interrupt_Initialize_ForPE1(void)
+ {
+ }

```

Figure 4-14 Rename R_Interrupt_Initialize_ForPE API

- (7) Add declaration code for “R_Interrupt_Initialize_ForPE1” API.

```

178
179
180
181
182
183
184
185
186
187
188
189
191
193
194
195
196
197
198
199
200
201
202
203
204
205
207
209
210
211
212
213
214
+ #define _PCR_ALT_OUT_SETTING (0xFFFFFFFUL) /* ALT Output setting */
+ #define _PCR_DIRECT_ALT_MODE_SETTING (0xFFFFFFFUL) /* Direct ALT mode setting */
+ /* Write protected macro definition */
+ #define _WRITE_PROTECT_ENABLE (0xASASAS01UL) /* Write protected */
+ #define _WRITE_PROTECT_DISABLE (0xASASAS00UL) /* Write protected */
+ #define _PORT_WRITE_PROTECT_DISABLE (0x00000000UL) /* Write protected */
+ /* Specify the interrupt bind (request) destination */
+ #define _INT_CPU_PE0 (0x00000000UL) /* Bound to PE0 */
+ #define _INT_CPU_PE1 (0x00000001UL) /* Bound to PE1 */
+ #define _INT_CPU_PE2 (0x00000002UL) /* Bound to PE2 */
+ #define _INT_CPU_PE3 (0x00000003UL) /* Bound to PE3 */
+
+ Typedef definitions
+ #ifndef __TYPEDEF__
+ typedef signed char int8_t;
+ typedef unsigned char uint8_t;
+ typedef signed short int16_t;
+ typedef unsigned short uint16_t;
+ typedef signed long int32_t;
+ typedef unsigned long uint32_t;
+ typedef signed long long int64_t;
+ typedef unsigned long long uint64_t;
+ typedef unsigned short MD_STATUS;
+ #define __TYPEDEF__
+ #endif
+
+ Global functions
+ void R_Systeminit(void);
+ void R_Interrupt_Initialize_ForPE1(void);
+ void R_Interrupt_Initialize_ForPE1(void);
+ /* Start user code for function. Do not edit comment generated here */
+ /* End user code. Do not edit comment generated here */
+ #endif

```

Figure 4-15 Add declaration code for “R_Interrupt_Initialize_ForPE1” API

- (8) Add calling code for “R_Interrupt_Initialize_ForPE1” API.

```

2
19
21
27
29
30
31
33
35
36
37
38
39
40
41
43
45
46
47
49
54
55
56
57
58
59
60
61
62
+ * DISCLAIMER
+ * File Name      : r_cg_systeminit.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_ADCJ0.h"
+ #include "r_cg_cgc.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_Systeminit
+ void R_Systeminit(void)
+ {
+     R_Interrupt_Initialize_ForPE1();
+     /* Set peripheral settings */
+     R_CGC_Create();
+     R_Config_ADCJ0_Create();
+ }

```

Figure 4-16 Add calling code for “R_Interrupt_Initialize_ForPE1” API

4.4 Component Settings

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

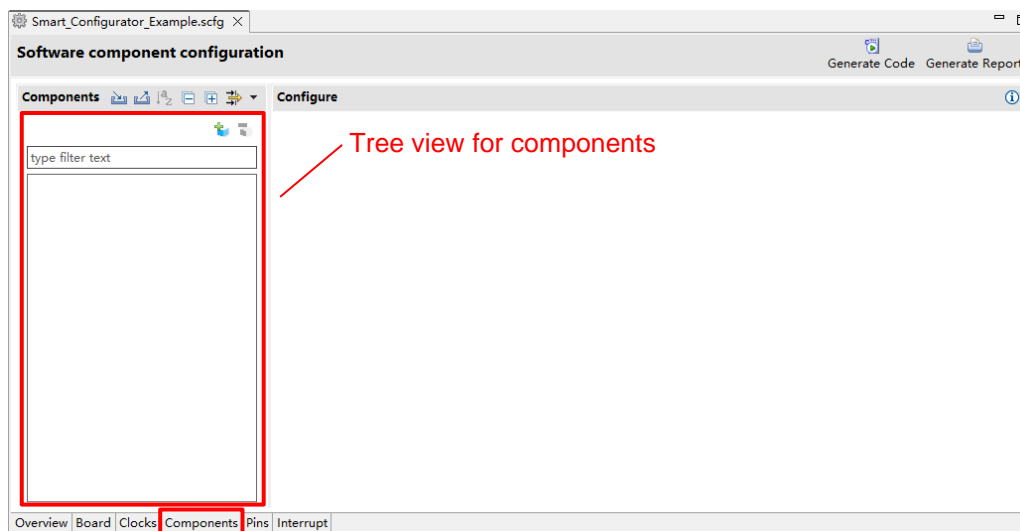


Figure 4-17 [Components] Page

4.4.1 Switching Between the Component View and Hardware View

The Smart Configurator provides two tree views: Component View and Hardware View. User can switch two views by clicking the following icons:

- (1) Click on the “Show by Component View” radio button. The tree view will display the components by component category.
- (2) Click on the “Show by Hardware View” radio button. The tree view will display the components in a hardware resource hierarchy.

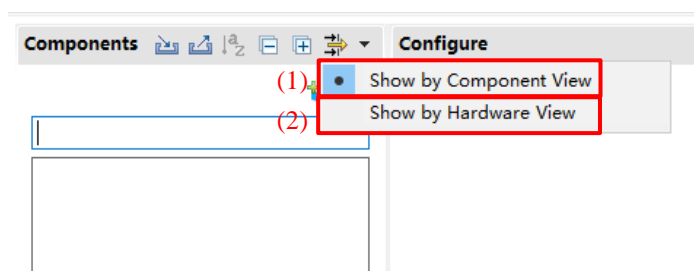





Figure 4-18 Switching to the Hardware View

4.4.2 Adding a Software Component

The Smart Configurator provides two methods for adding a new component:

- (a) Click on the  (Add component) icon.
- (b) On Hardware Tree, double-click on a hardware resource node

The following describes the procedure for adding a component by clicking on the  (Add component) icon.

- a-1. Click on the  (Add component) icon.

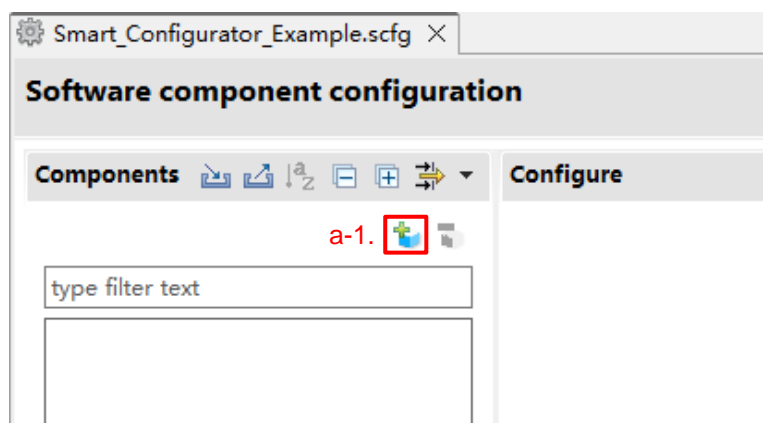


Figure 4-19 Adding a Component

- a-2. Select a component from the list in the [Software Component Selection] page of the [New Component] dialog box (e.g. A/D Converter).

- a-3. Click on [Next].

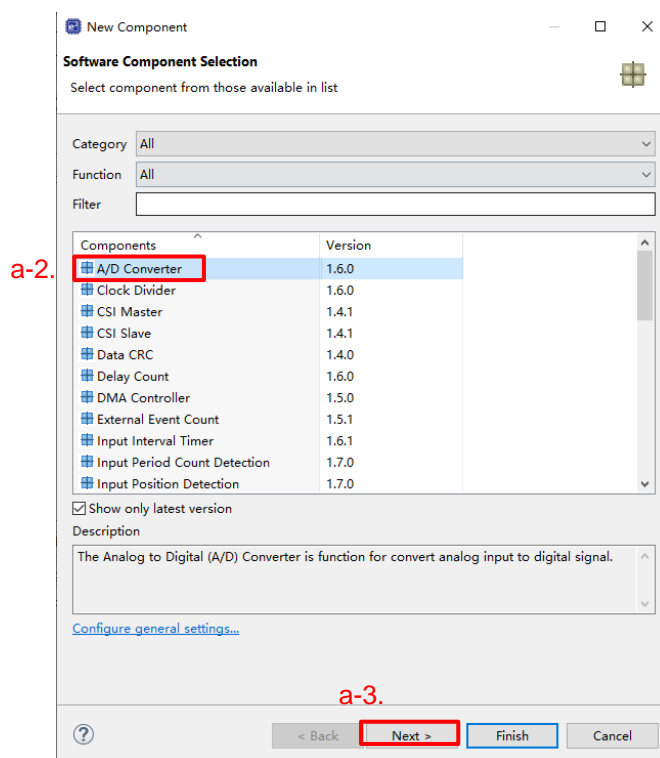


Figure 4-20 Adding a Code Generator Component

- a-4. Specify an appropriate configuration name in the [Add new configuration for selected component] page of the [New Component] dialog box or use the default name (for e.g., Config_ADCA0).
- a-5. Select a hardware resource or use the default resource (for e.g., ADCA0).
- a-6. Click on [Finish].

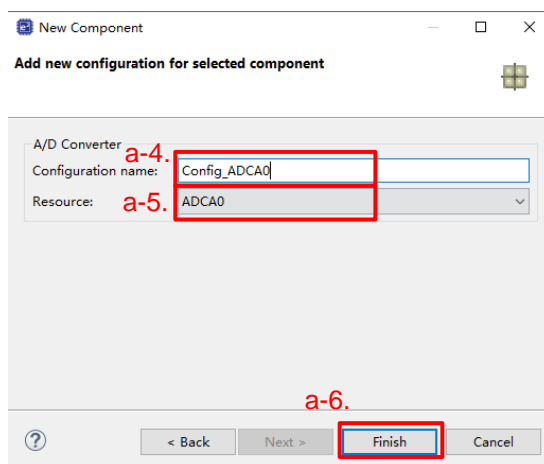


Figure 4-21 Adding a Component

To add a component on Hardware Tree directly, user can use the following procedure:

- b-1. Click on the “Show by Hardware View” radio button. The tree will display in a hardware resource hierarchy.
- b-2. Double-click on a hardware resource node (for e.g., A/D Converter) to open the [New Component] dialog box.
- b-3. (Select a component from the list (for e.g., A/D Converter) to add a new configuration.
- b-4. Follow the same procedure is same as “Click on [Add component] icon” step a-2 to a-6.

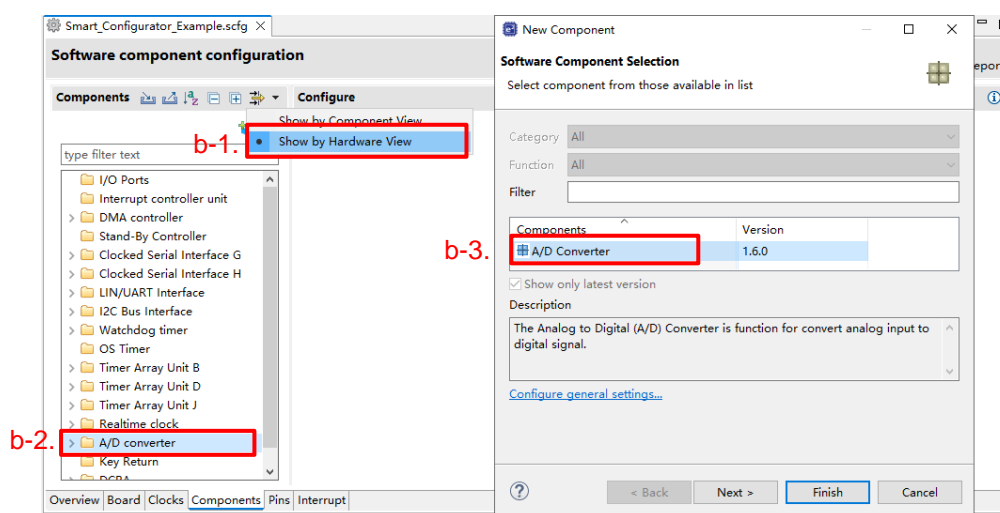



Figure 4-22 Adding a Code Generator Component to the Hardware View

4.4.3 Removing Software Component

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

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

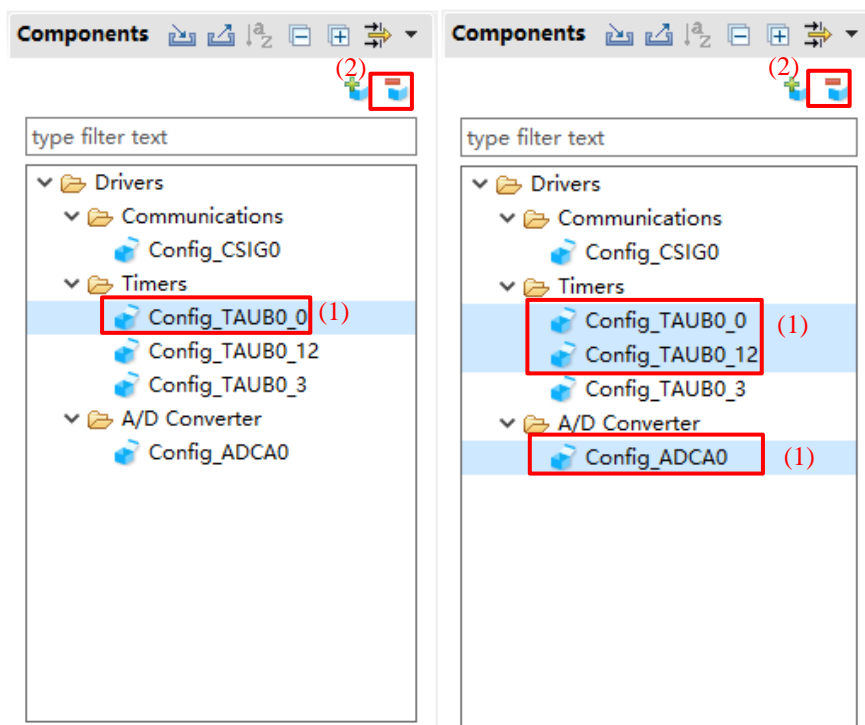



Figure 4-23 Removing a Software Component or Multiple Components

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

To delete the source files previously generated for the removed components from the e² studio project tree, click  (Generate Code)) icon.

4.4.4 Setting a Code Generator Component

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

- (1) Select a Code Generator configuration from the Components tree (for e.g., A/D Converter).
- (2) Configure the driver in the [Configure] panel to the right of the Components tree. The following steps and figure show an example.
 - a. Select [24 cycles] under [Sampling control setting].
 - b. Select [Use scan group 2] under [Scan group selecting].
 - c. Input [2] to [End pointer of virtual channel].

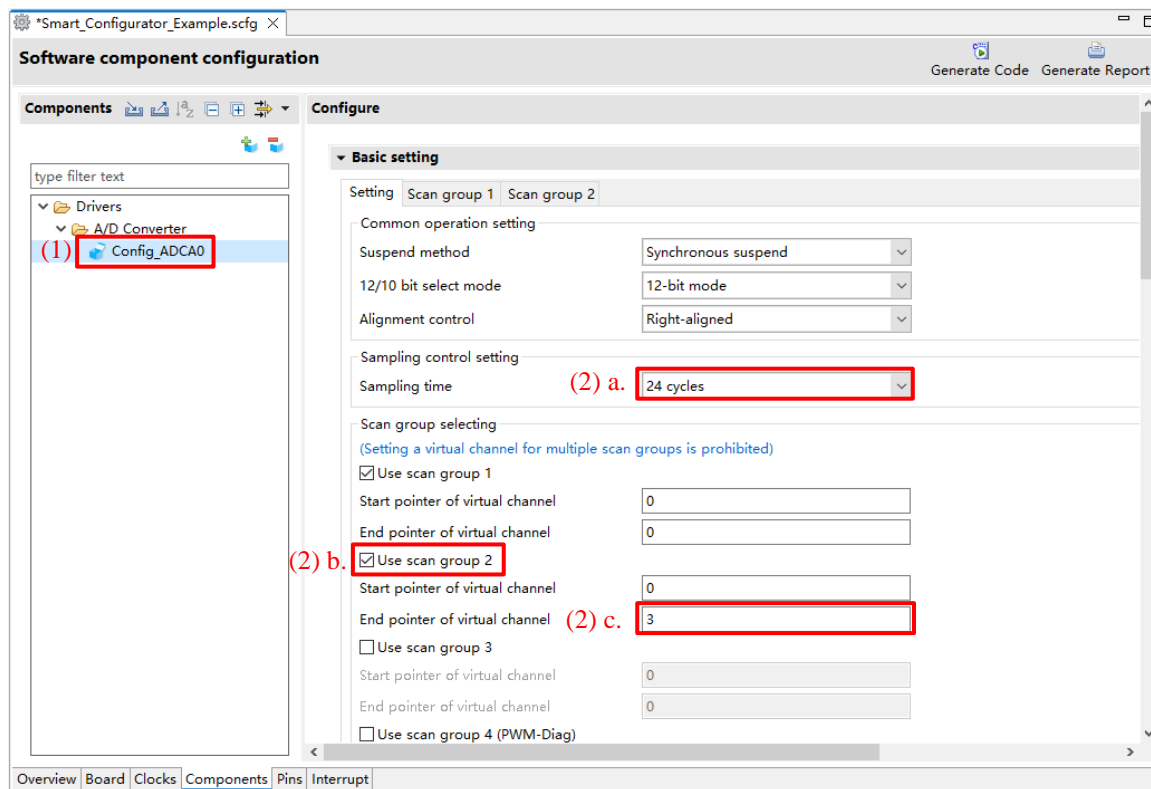


Figure 4-24 Setting of a Code Generator Driver

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

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

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

4.4.5 Changing the Resource for a Code Generator Configuration

The Smart Configurator enables user to change the resource for a Code Generator configuration (for e.g., from ADCA0 to ADCA1). Compatible settings can be ported from the current resource to the new resource selected.

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

- (1) Right-click on a Code Generator configuration (for e.g., Config_ADCA0).
- (2) Select [Change resource] from the context menu.

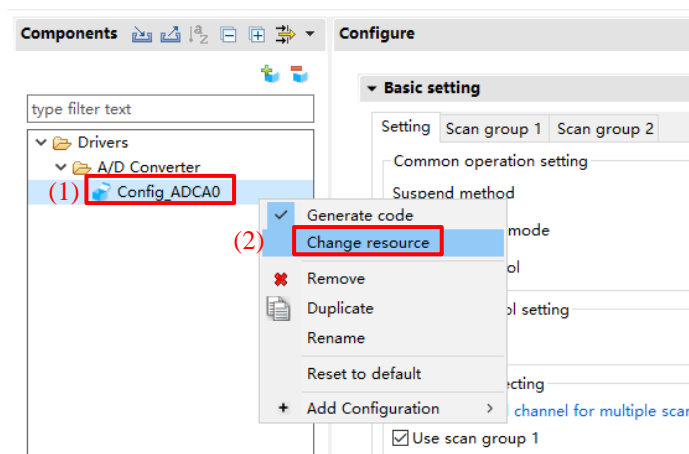


Figure 4-25 Changing the Resource

- (3) Select a new resource (for e.g., ADCA1) in the [Resource Selection] dialog box.
- (4) The [Next] button will be active, click on it.

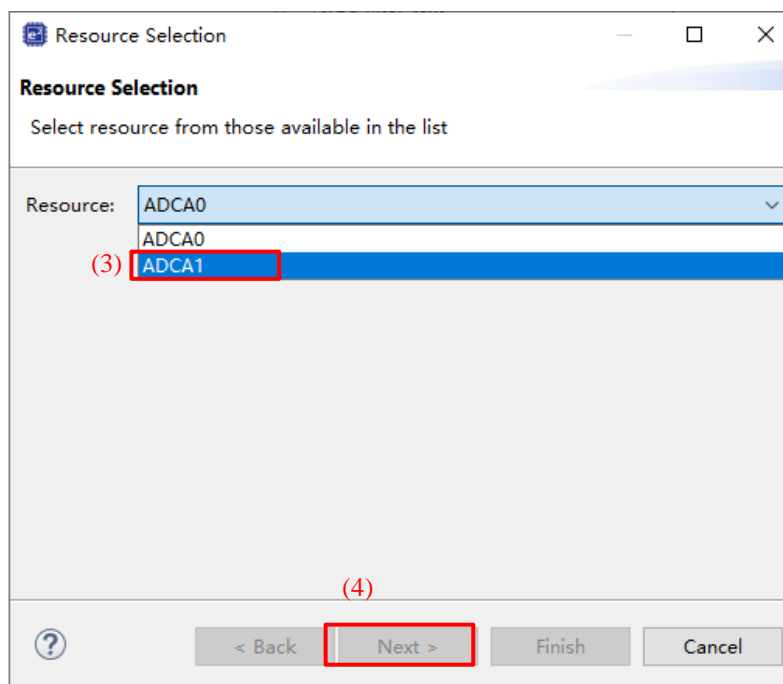


Figure 4-26 Components Page – Selecting a New Resource

- (5) Configuration settings will be listed in the [Configuration setting selection] dialog box.
- (6) Check the portability of the settings.
- (7) Select whether to use the listed below or default settings.
- (8) Click on [Finish].

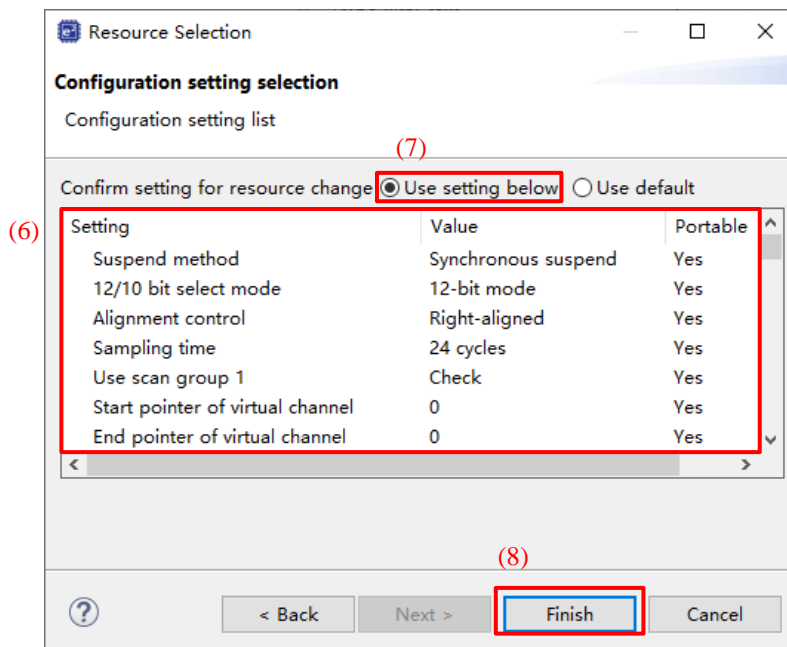


Figure 4-27 Checking the Settings of the New Resource

The resource is automatically changed (for e.g., changed from ADCA0 to ADCA1).

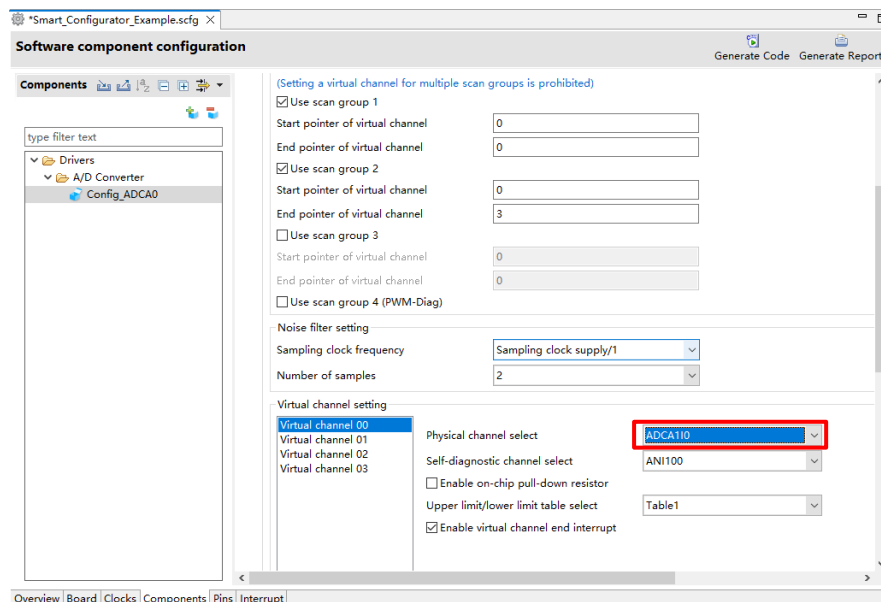


Figure 4-28 Resource Changed Automatically

To change the configuration name, follow the procedure below.

- (9) Right-click on the Code Generator configuration.
- (10) Select [Rename] to rename the configuration (for e.g., change Config_ADCA0 to Config_ADCA1).

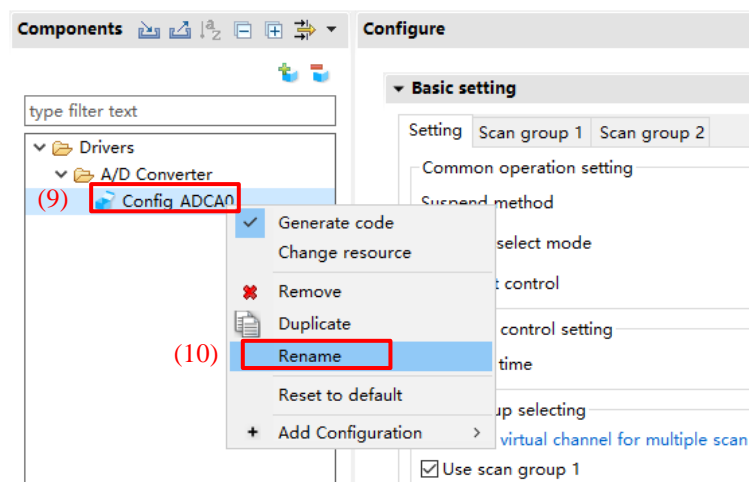


Figure 4-29 Renaming the Configuration

4.4.6 Export Component Configuration

The current configuration can be exported as *.xml file by clicking on the [📄] (Export Configuration) button on the [Components] tabbed page.

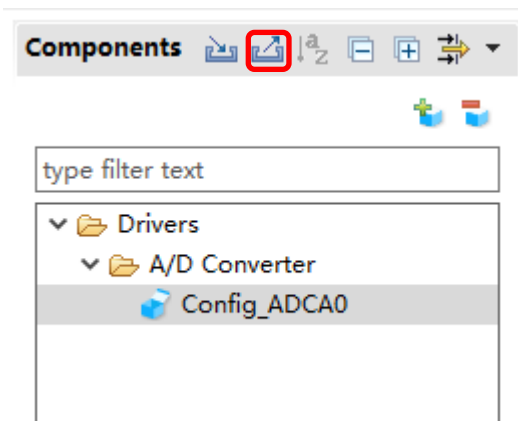


Figure 4-30 Export Configuration (xml format)

4.4.7 Import Component Configuration

Click on the [📄] (Import Configuration) button and select an exported xml file will import component configuration.

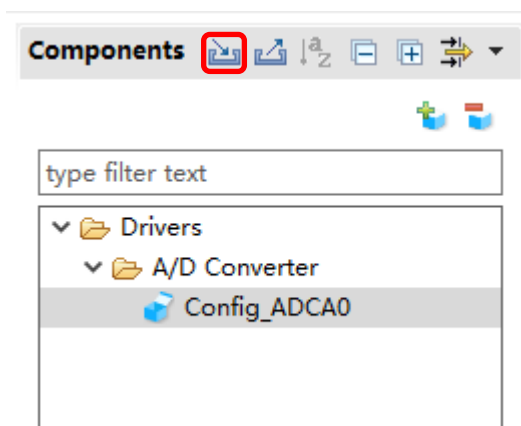


Figure 4-31 Import Configuration (xml format)

4.4.8 Configure General Setting of the Component

User can change the general setting of the component such as location and dependency. If user wants to change it, click the [Configure general settings...] link on the [Software Component Selection] page displayed in the [New Component] dialog (Figure 4-20), and display the [Preferences] dialog.

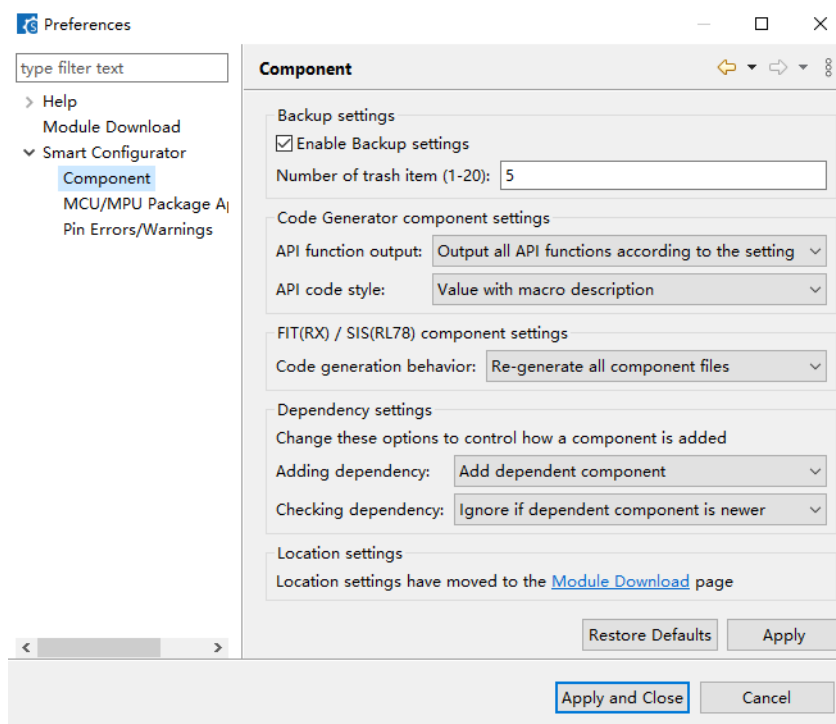


Figure 4-32 Configure General Setting of Component

Notes:

1. User can 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.

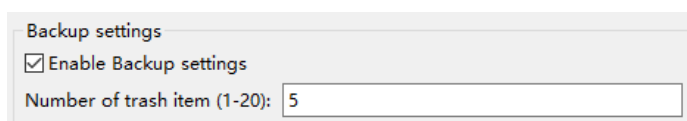


Figure 4-33 Trash number setting

2. If user wants to only generate initialization API function, user 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 user changes back to default option setting: [Output all API functions according to the setting], then all API functions will be generated again.

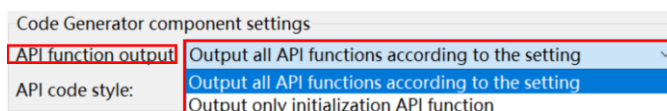


Figure 4-34 [API function output] setting

4.5 Pin Settings

The [Pins] page is used for assigning pin functions. User can switch the view by clicking on the [Pin Function] and [Pin Number] pages. 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.

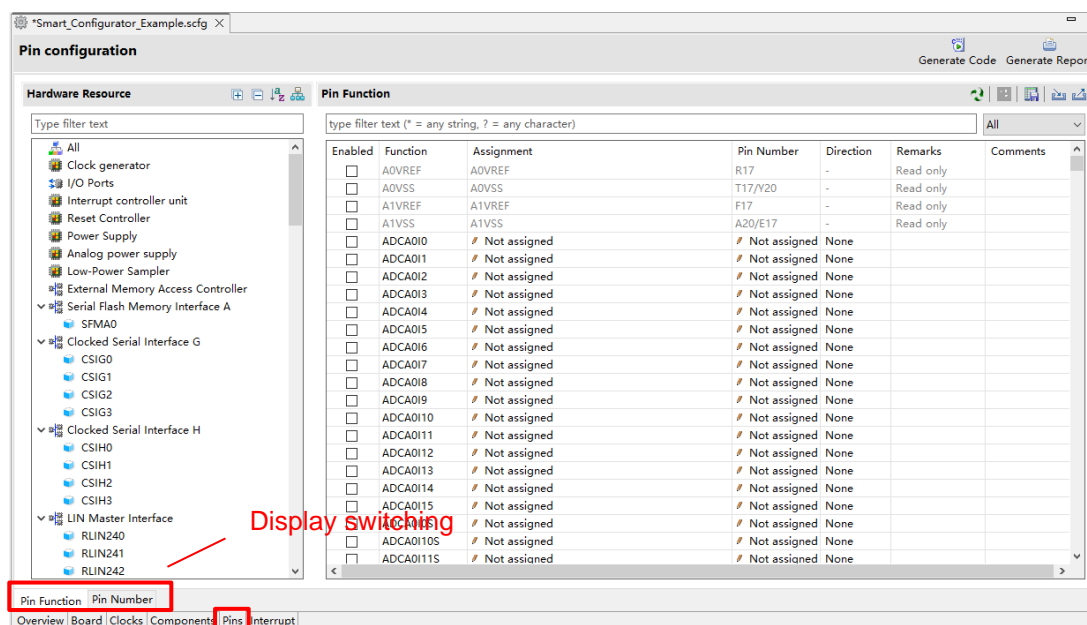


Figure 4-35 [Pins] Page ([Pin Function])

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

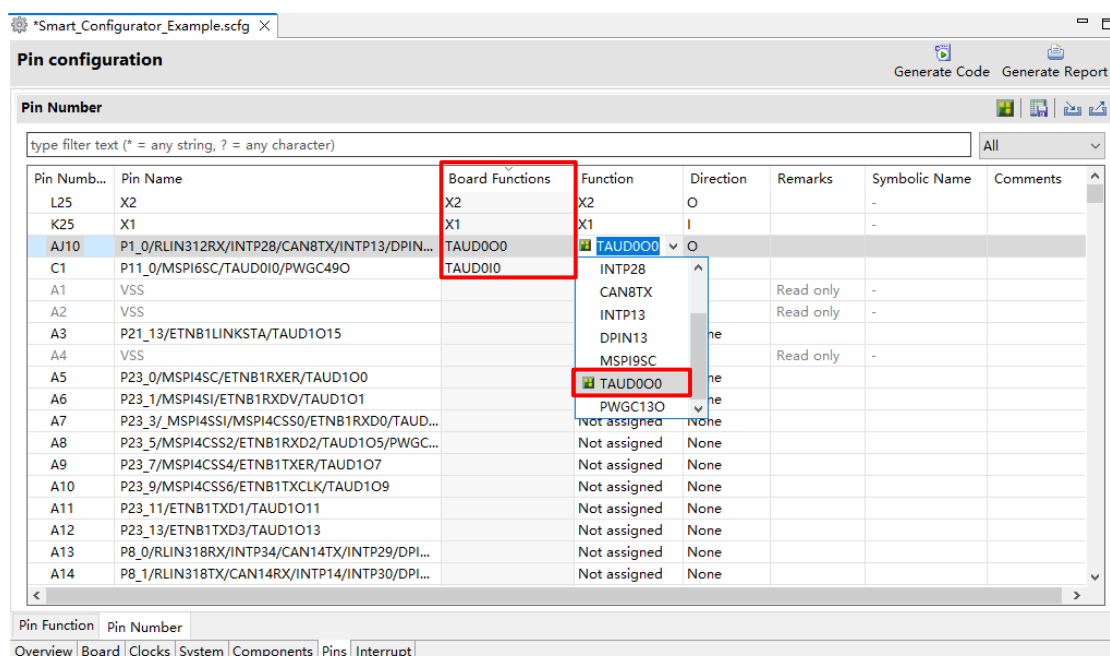




Figure 4-36 [Pins] Page ([Pin Number])

4.5.1 Changing the Pin Assignment of a Software Component

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

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

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

- (1) Click on  (Show by Hardware Resource or Software Components)] to switch to the component view.
- (2) Select the target software component (for e.g., Config_INTC).
- (3) Click the [Enabled] header to sort by pins used.
- (4) In the [Assignment] column or [Pin Number] column on the [Pin Function] list, change the pin assignment (for e.g., change from P10_0 to P0_1).
- (5) In addition, assignment of a pin can be changed by clicking on the  (Next group of pins for the selected resource)] button. Pin that has peripheral function is displayed each time the button is clicked.

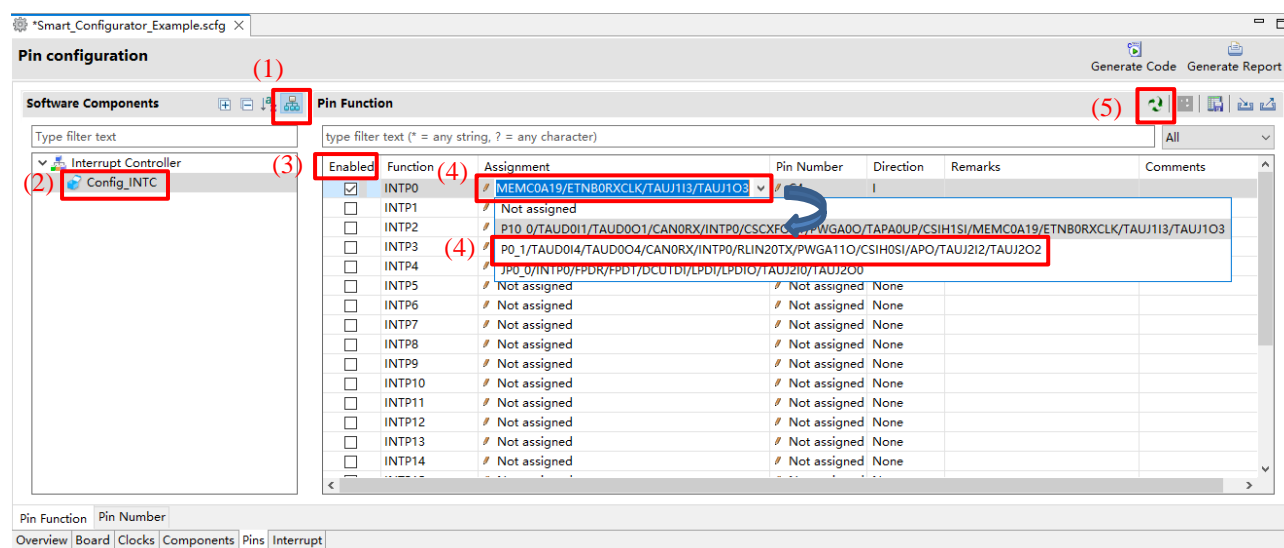



Figure 4-37 Pin Settings – Assigning Pins on the [Pin Function] list

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

4.5.2 Assigning Pins Using the MCU/MPU Package View

The Smart Configurator visualizes the pin assignment in the MCU/MPU Package view. User 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...].

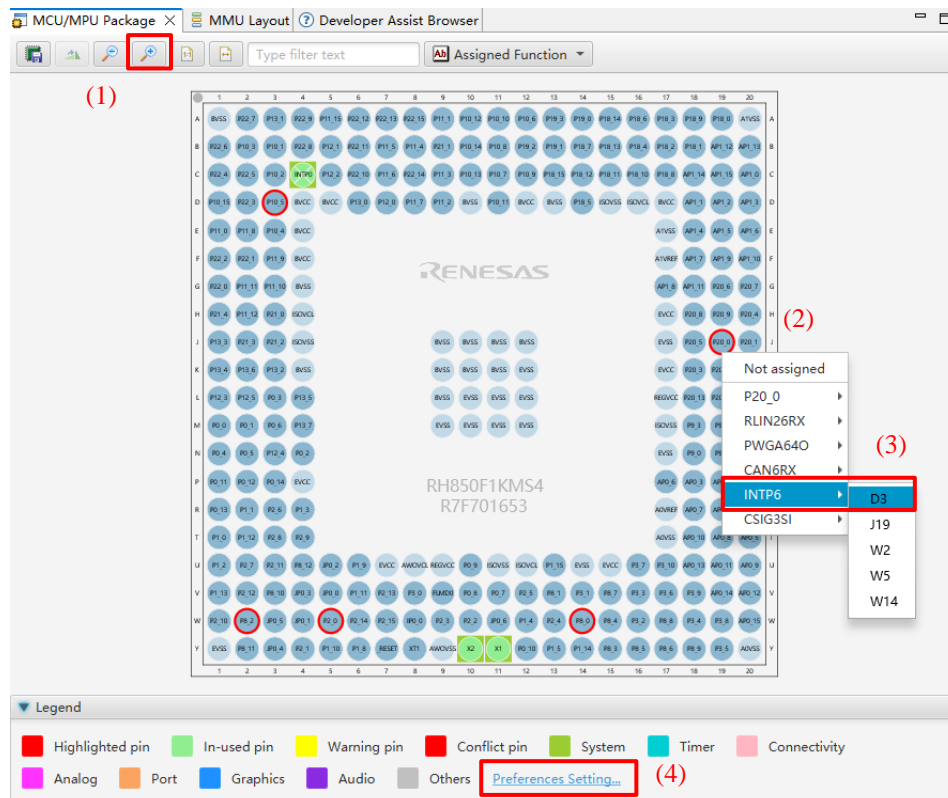


Figure 4-38 Assigning Pins Using the MCU/MPU Package View

4.5.3 Show Pin Number from Pin Functions

User 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.

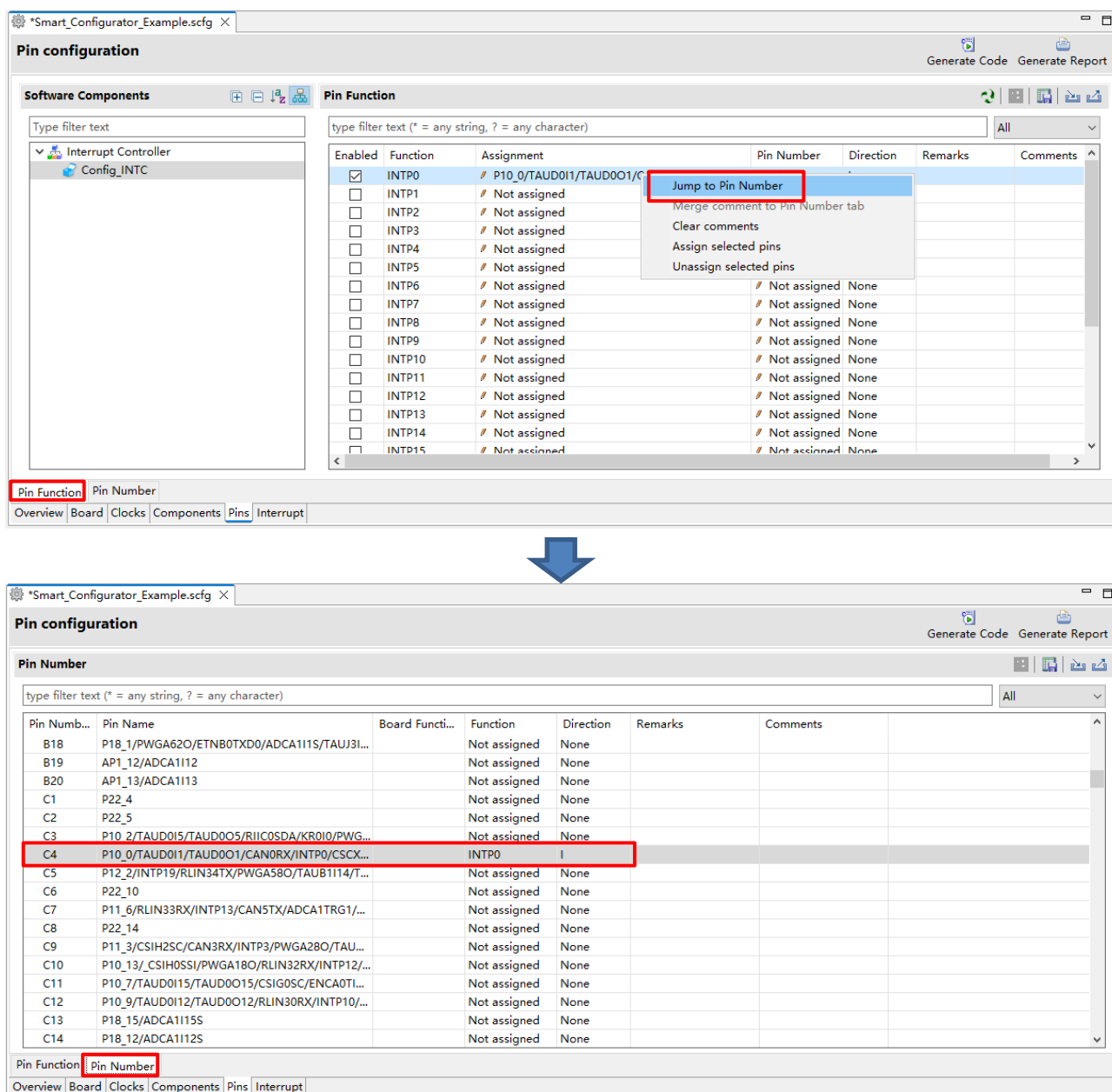



Figure 4-39 Jump to Pin Number

4.5.4 Exporting pin settings

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

- (1) Click on the  (Export board setting) button on the [Pins] page.
- (2) Select the output location and specify a name for the file to be exported.

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

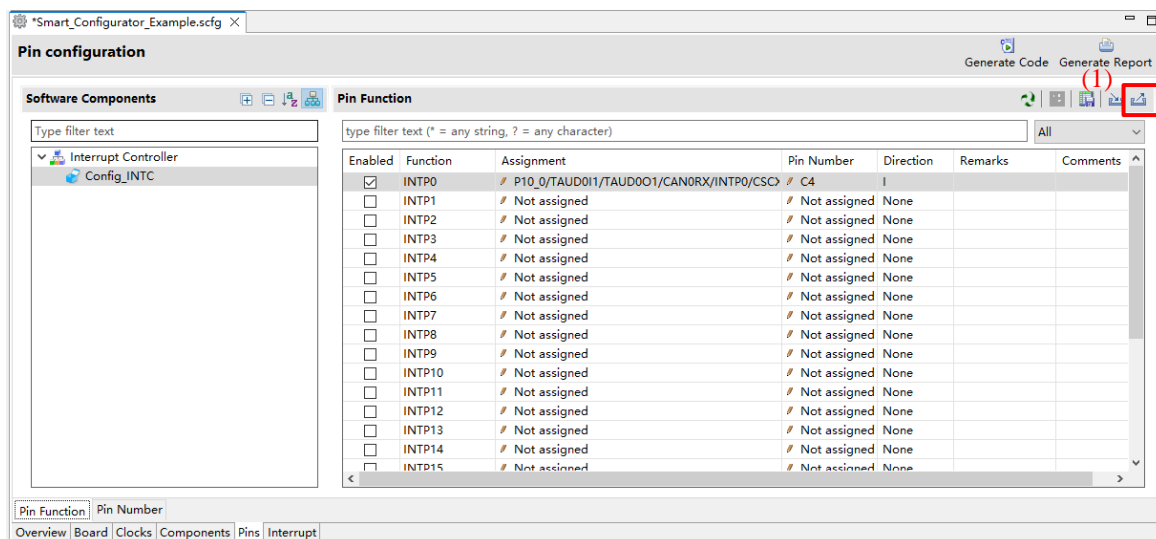




Figure 4-40 Exporting Pin Settings to an XML File

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

4.5.5 Importing pin settings

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

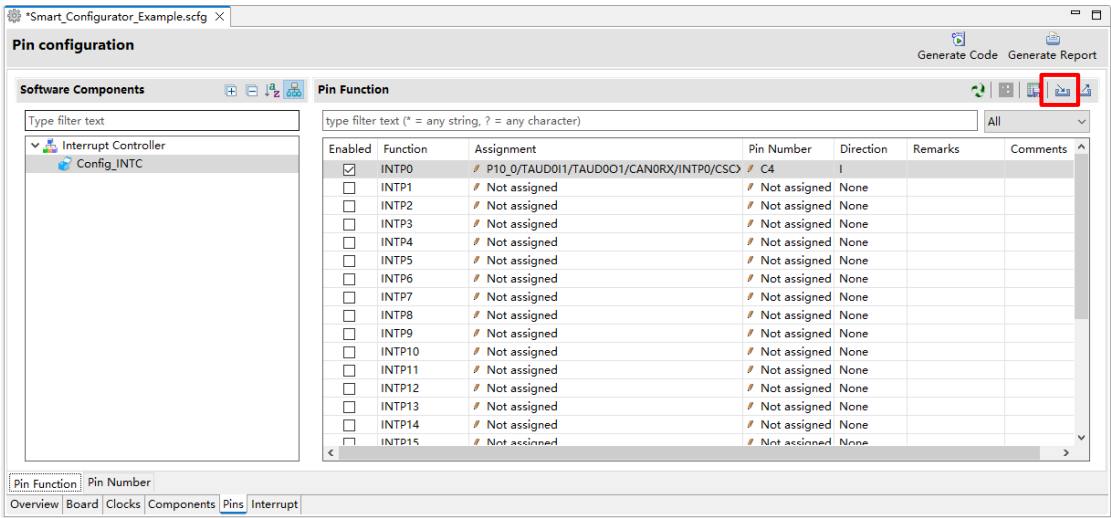


Figure 4-41 Importing Pin Settings from an XML File

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

4.5.6 Pin Setting Using Board Pin Configuration Information

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

The following describes the procedure for collective setting of pins.

- (1) Select [Board Function] in the MCU/MPU Package. (The initial pin configuration of the board can be referred.)
- (2) Open the [Pin Configuration] page and click the [Assign default board pins] button.
- (3) When [Assign default board pins] dialog opens, click [Select all].
- (4) Click [OK].

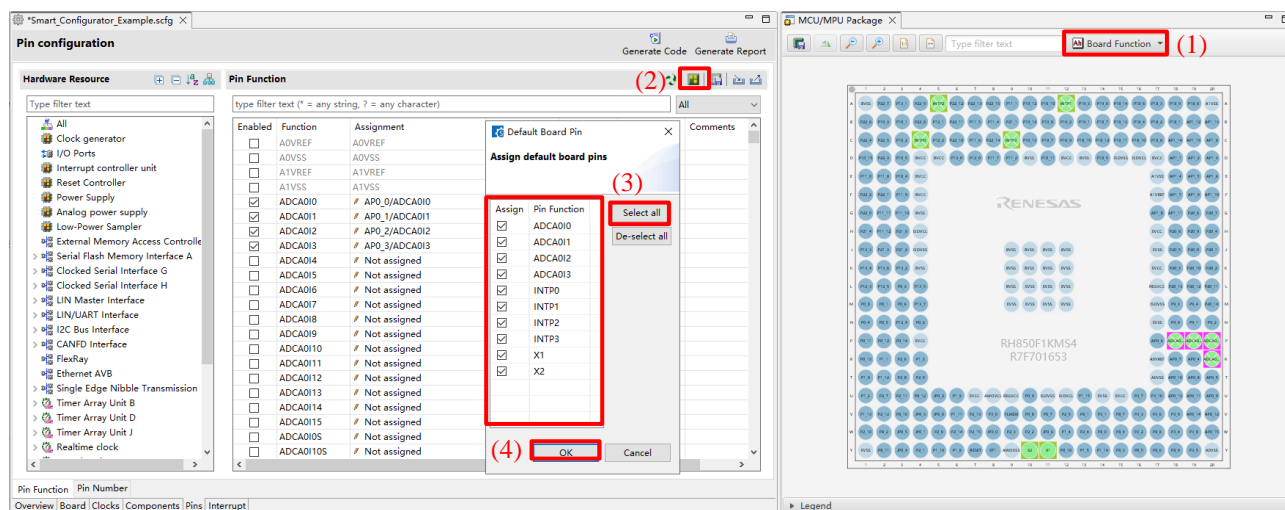


Figure 4-42 Importing Pin Settings from an XML File

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] page and [Pin Number] page on the [Pins] page, user can refer to it more easily.

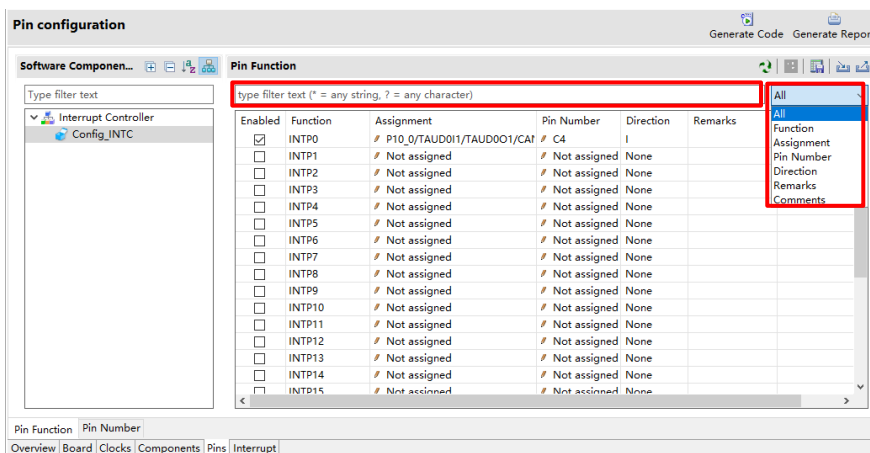


Figure 4-43 Filter for [Pin Function] Page

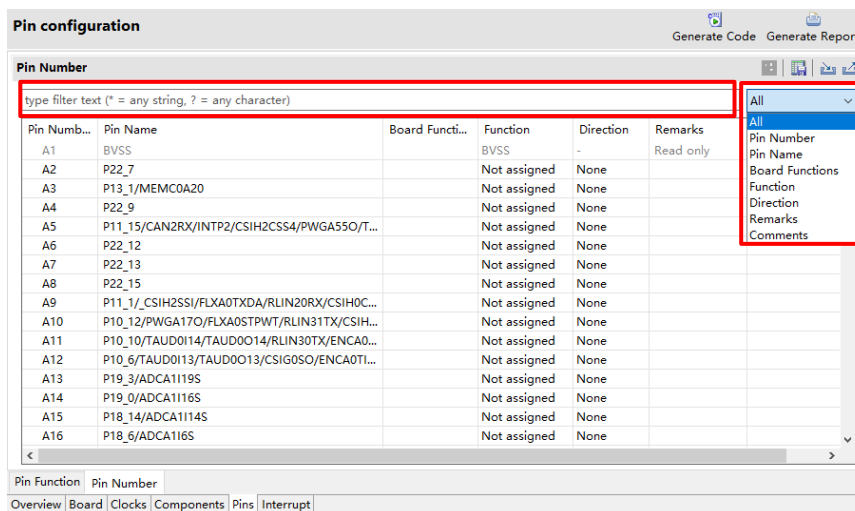


Figure 4-44 Filter for [Pin Number] Page

4.5.8 Pin Errors/Warnings setting

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

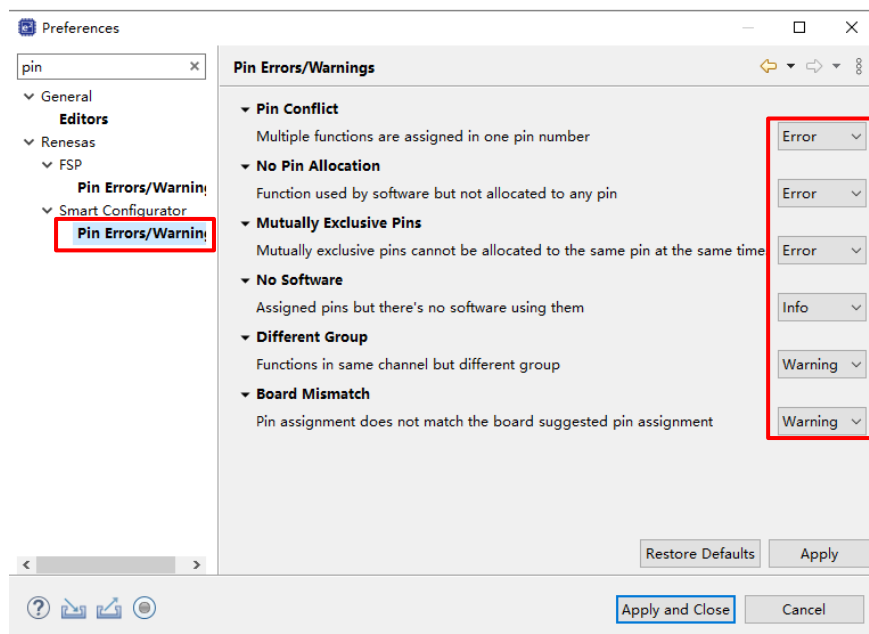


Figure 4-45 Pin Errors/Warnings settings at Preferences

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

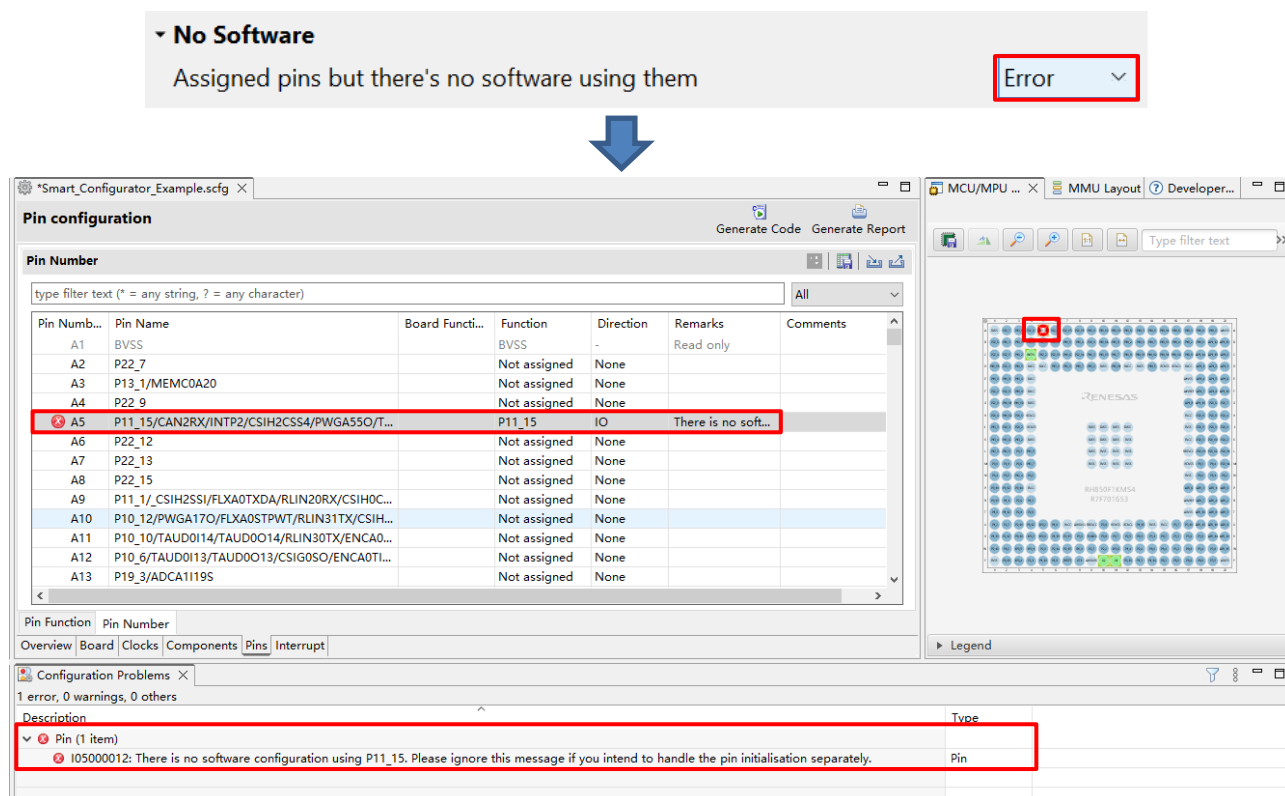


Figure 4-46 Change “No Software” Setting from “Info” to “Error”

4.6 Interrupt Settings

The [Interrupt] page displays all interrupt by each of the vector numbers. User can check and set the interrupts of the peripheral modules that have been selected on the [Components] page. The interrupts are displayed for each of the vector numbers. Generally, user can set the common settings such as interrupt priority levels, OS management, Interrupt Handler, Generate Entity and Generate Enable/Disable Function. For RH850/U2A, user can set PE_n to decide if the interrupt is applied to the PE_n .

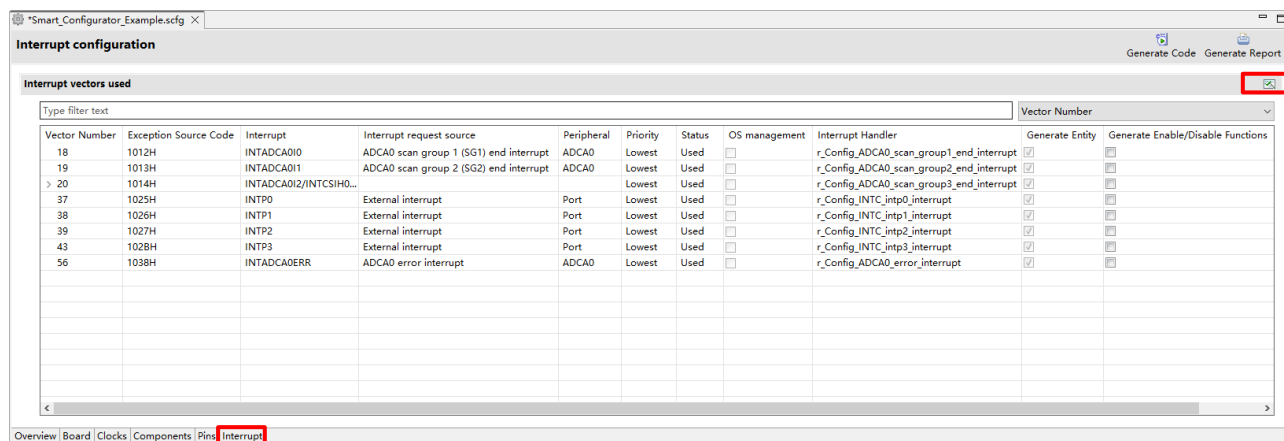



Figure 4-47 [Interrupts] Page

4.6.1 Changing 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) User can change the interrupt priority level on the [Interrupt] page.
- (2) The [OS management] column becomes active for a project that uses RTOS (RI850V4). Selecting a checkbox in the column outputs the corresponding interrupt function in the interrupt format that can be managed by the OS.

Note: Since the current e² studio doesn't support RTOS selection when creating an RH850 project, the [OS management] column is always inactive.

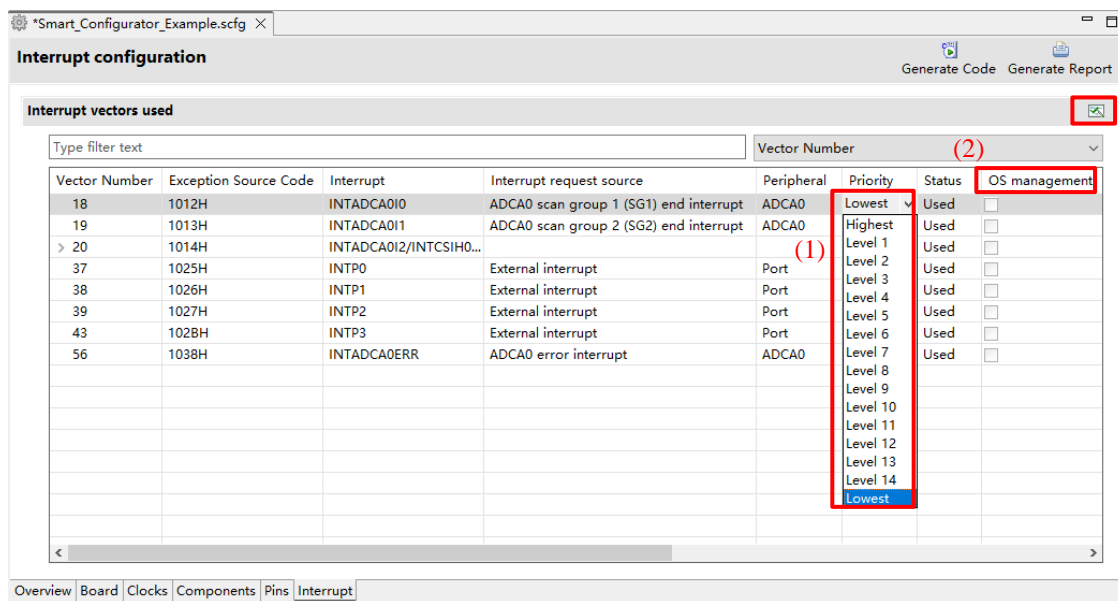


Figure 4-48 Interrupt Setting

4.6.2 Changing The PEn Setting (RH850/U2A Only)

User can select which PEn to respond to the interrupt in use. PEn can be set on the [Interrupt] page by below steps:

- (1) PEn is chosen to be used in [System] page (please refer to chapter 4.3, System Settings (only for RH850/U2A))
- (2) Check or uncheck the checkbox in column PEn 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.

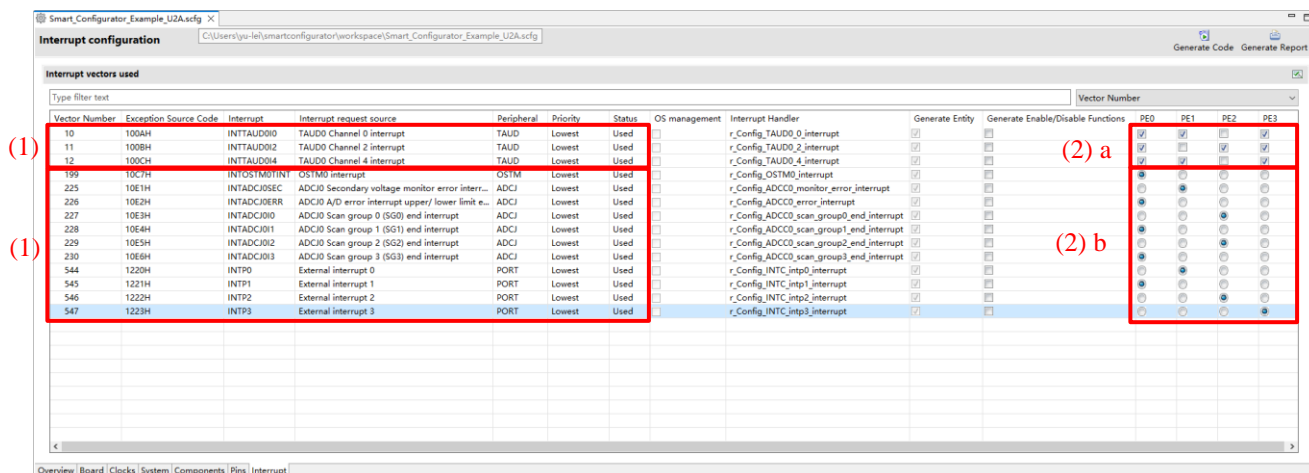


Figure 4-49 PEn Setting

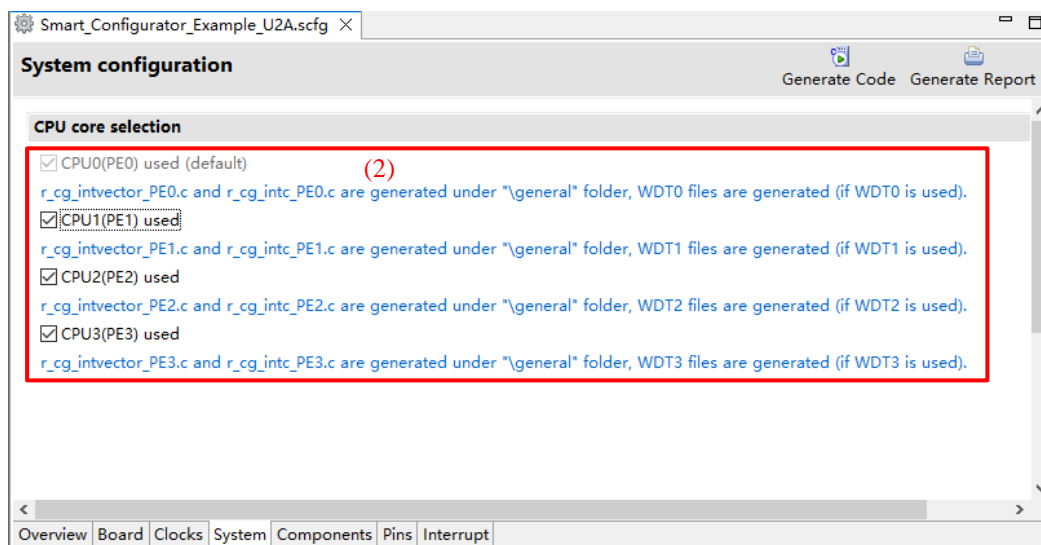


Figure 4-50 PEn is Chosen to be Used or Unused in [System]

Note:

Only RH850/U2A supports PEn setting.

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

4.6.3 Changing Interrupt Handler Name, Generate Entity and Generate Enable/Disable Function Setting

User can edit the interrupt handler for each interrupt in [Interrupt] page and decide if to generate the interrupt handler entity and interrupt enable/disable function by Smart Configurator for RH850.

- (1) User can rename the default name or input user-defined interrupt handler name manually by editing column [Interrupt Handler] which lists all interrupt handler.

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

- (2) User can specify whether the interrupt handler entity is generated by Smart Configurator 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) User can specify whether the interrupt enable/disable function is generated by Smart Configurator 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 4-52 Interrupt Enable/Disable Functions Code Example**.

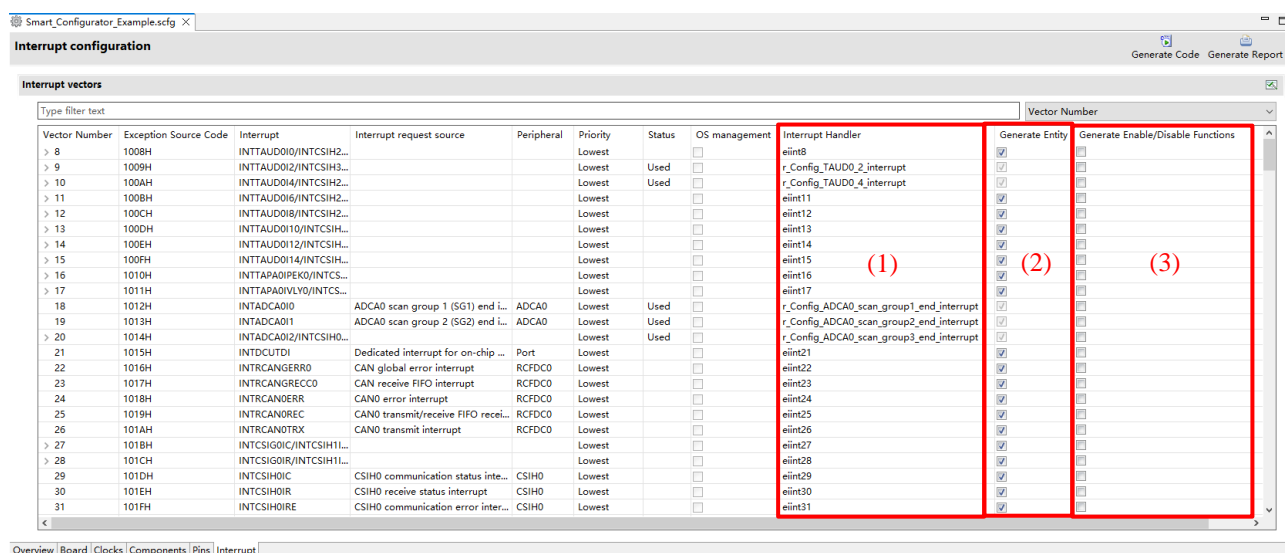
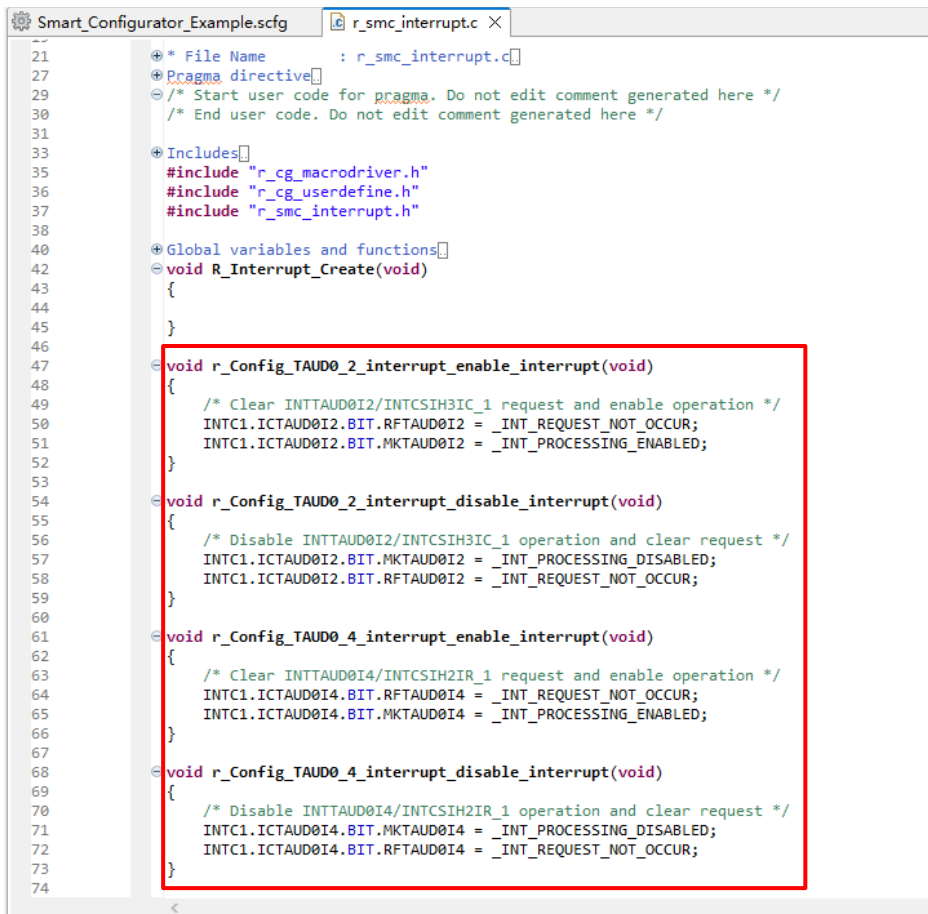


Figure 4-51 Interrupt Handler and Generate Entity Settings



```

21
27
29
30
31
33
35
36
37
38
40
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

/* File Name : r_smc_interrupt.c */
#pragma directive
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

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

Global variables and functions
void R_Interrupt_Create(void)
{
}

void r_Config_TAUD0_2_interrupt_enable_interrupt(void)
{
    /* Clear INTTAUD0I2/INTCSIH3IC_1 request and enable operation */
    INTC1.ICTAUD0I2.BIT.RFTAUD0I2 = _INT_REQUEST_NOT_OCCUR;
    INTC1.ICTAUD0I2.BIT.MKTAUD0I2 = _INT_PROCESSING_ENABLED;
}

void r_Config_TAUD0_2_interrupt_disable_interrupt(void)
{
    /* Disable INTTAUD0I2/INTCSIH3IC_1 operation and clear request */
    INTC1.ICTAUD0I2.BIT.MKTAUD0I2 = _INT_PROCESSING_DISABLED;
    INTC1.ICTAUD0I2.BIT.RFTAUD0I2 = _INT_REQUEST_NOT_OCCUR;
}

void r_Config_TAUD0_4_interrupt_enable_interrupt(void)
{
    /* Clear INTTAUD0I4/INTCSIH2IR_1 request and enable operation */
    INTC1.ICTAUD0I4.BIT.RFTAUD0I4 = _INT_REQUEST_NOT_OCCUR;
    INTC1.ICTAUD0I4.BIT.MKTAUD0I4 = _INT_PROCESSING_ENABLED;
}

void r_Config_TAUD0_4_interrupt_disable_interrupt(void)
{
    /* Disable INTTAUD0I4/INTCSIH2IR_1 operation and clear request */
    INTC1.ICTAUD0I4.BIT.MKTAUD0I4 = _INT_PROCESSING_DISABLED;
    INTC1.ICTAUD0I4.BIT.RFTAUD0I4 = _INT_REQUEST_NOT_OCCUR;
}

```

Figure 4-52 Interrupt Enable/Disable Functions Code Example

4.7 MCU Migration Feature

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

Note: Project settings may change due to device change.

Back up the project before executing the device change.

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

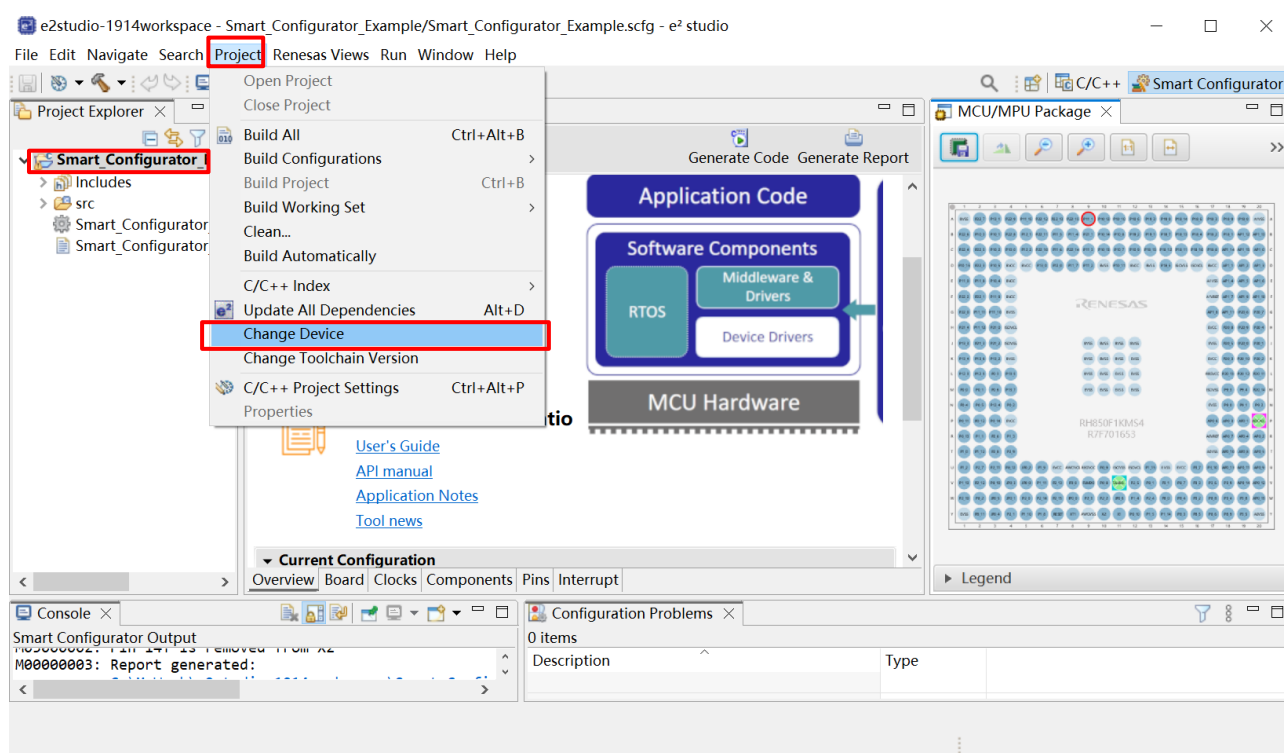


Figure 4-53 Select [Change Device]

- (2) Select the target device manually from the device selection list and click "OK". (Wild card search is supported) (e.g., change to RH850 – C1M 252pin Part number: R7F701275).

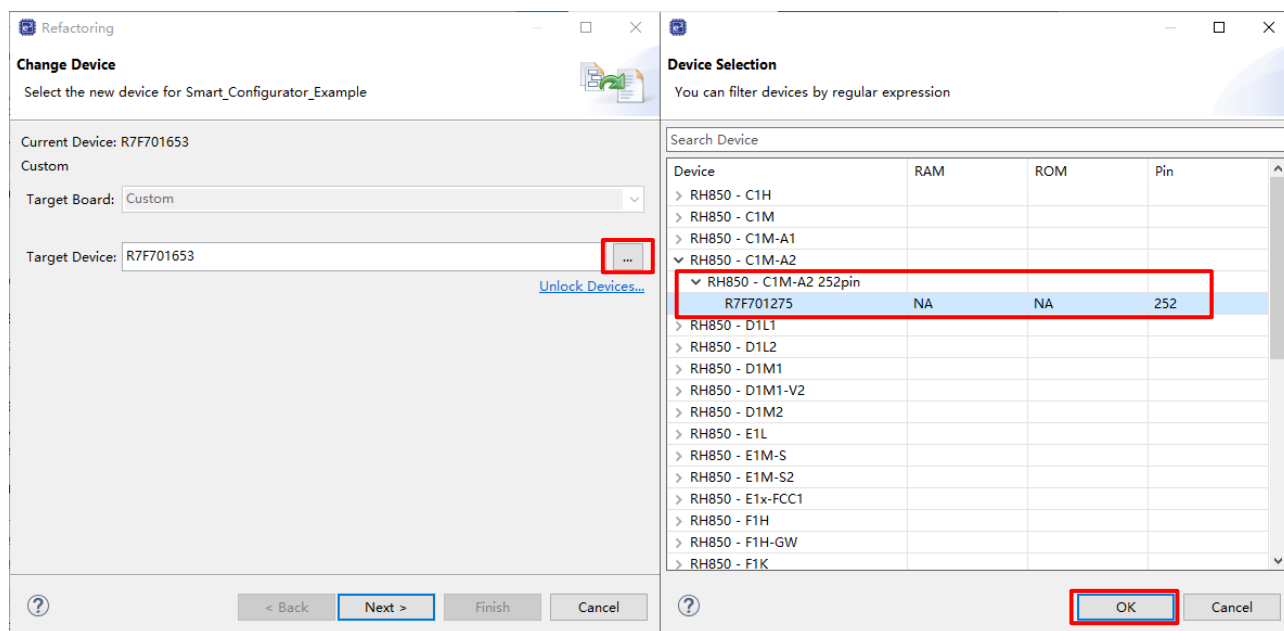


Figure 4-54 Select Target Device

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

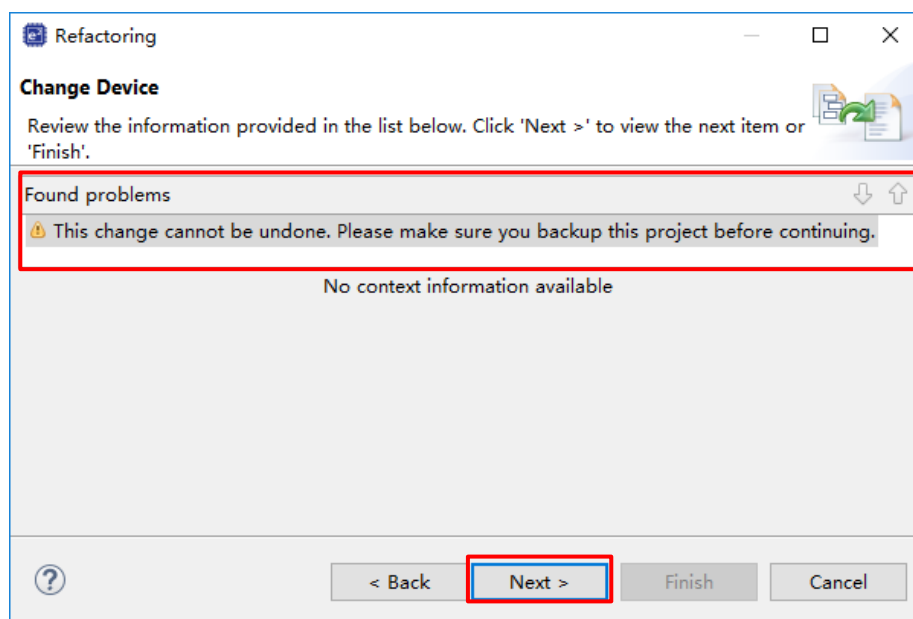
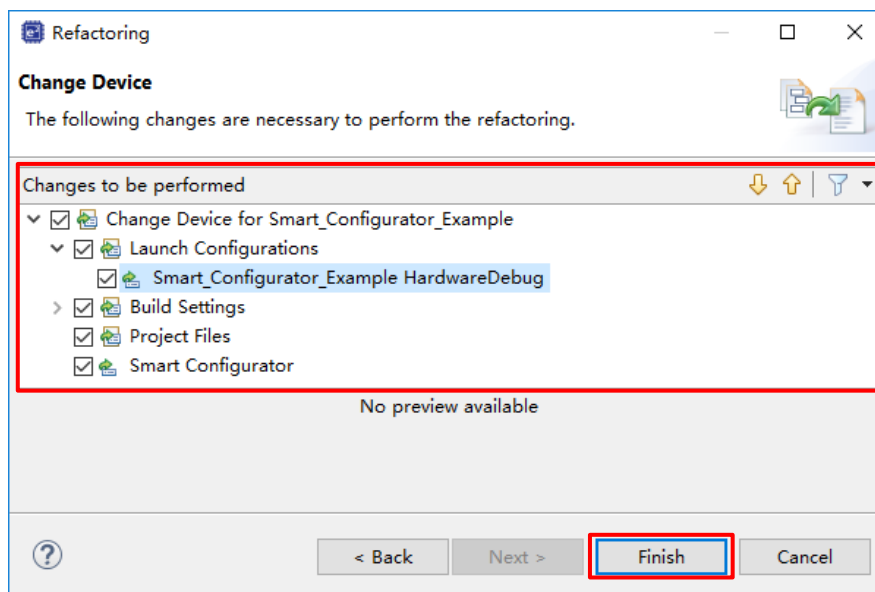


Figure 4-55 Found Problems

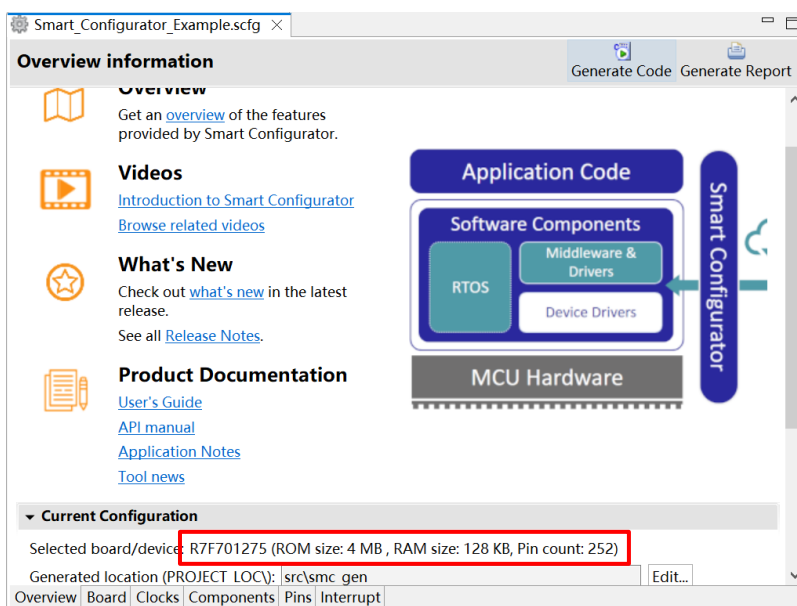
Table 4-2 Change Device Found Problems List

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

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

**Figure 4-56 Changes to be Performed**

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

**Figure 4-57 Device Update Confirmation**

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

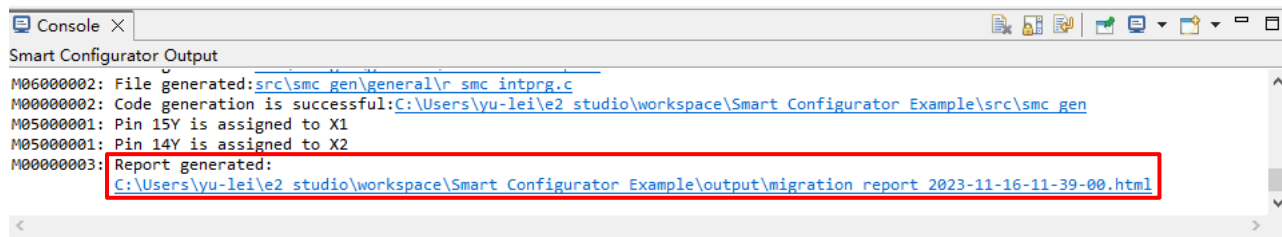


Figure 4-58 Configuration Conversion Status Report

5. Managing Conflicts

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

5.1 Resource Conflicts

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

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

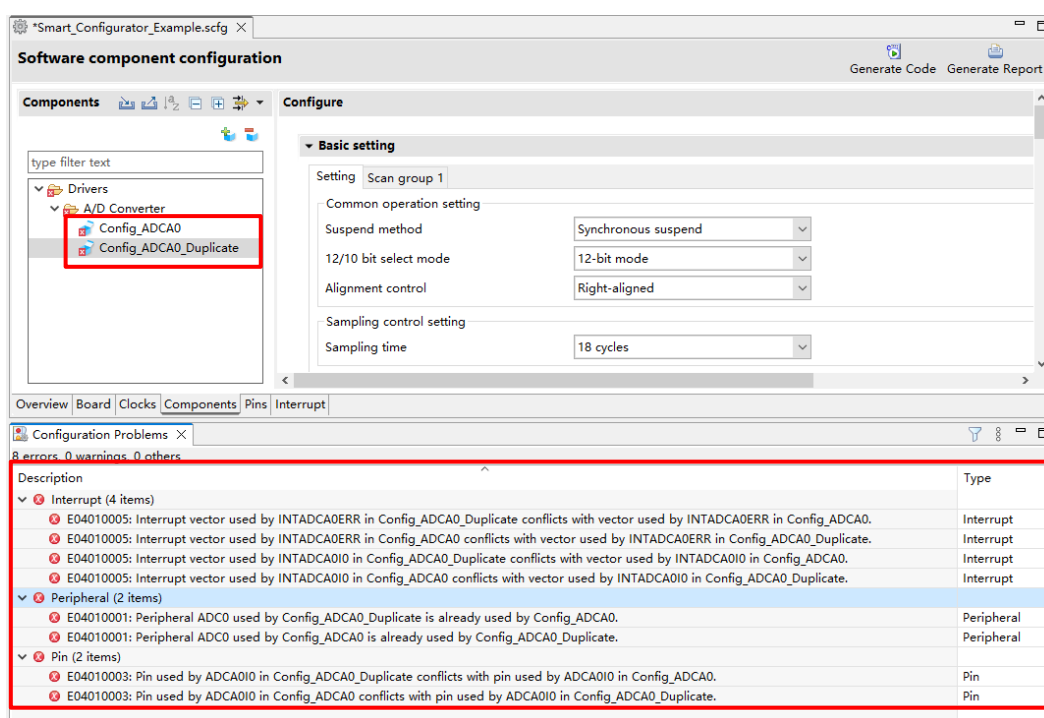


Figure 5-1 Resource Conflicts

5.2 Resolving Pin Conflicts

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

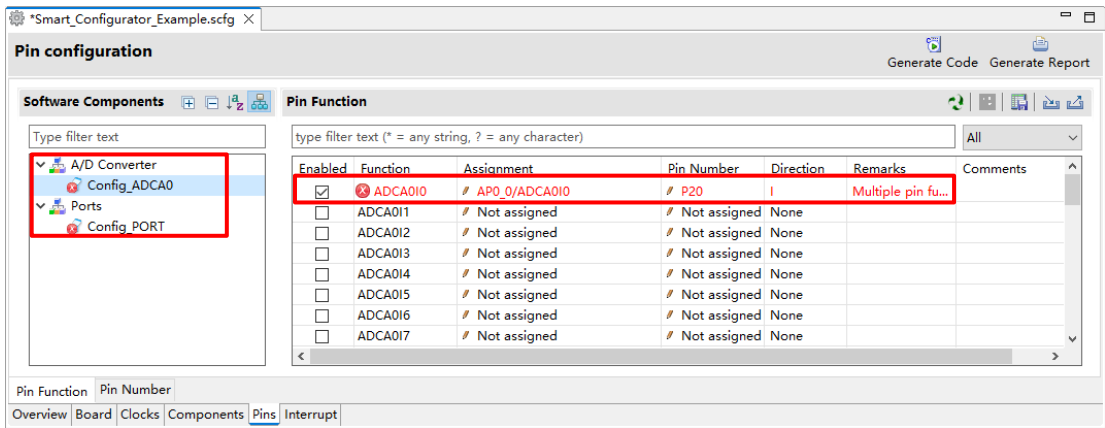


Figure 5-2 Pin Conflicts

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

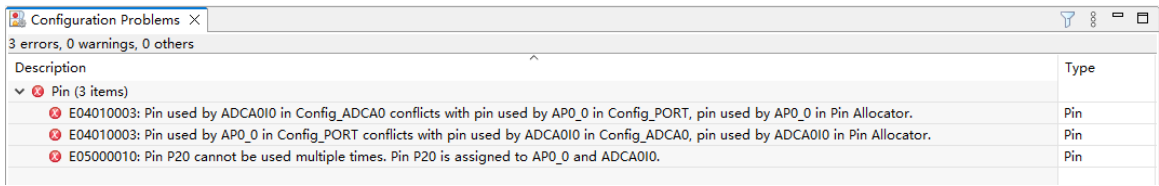


Figure 5-3 Pin Conflict Messages

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

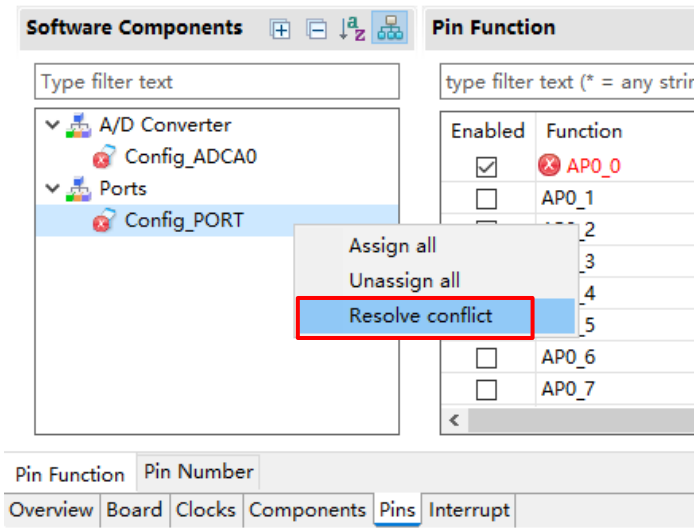


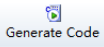
Figure 5-4 Resolving Pin Conflicts

The pin function of the selected nodes will be re-assigned to other pins.

6. Generating Source Code

Source generation can be generated even if there is a conflict in the Configuration Problems view.

6.1 Outputting Generated Source Code

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

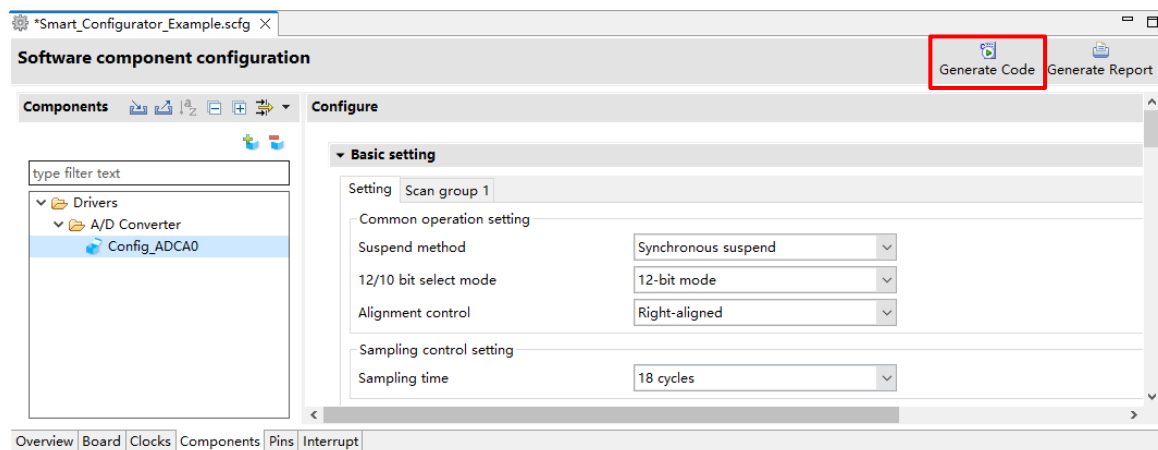


Figure 6-1 Generating a Source File

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

Note: If user puts a self-created source file in smc_gen folder, it will be erased at time of generating source code.

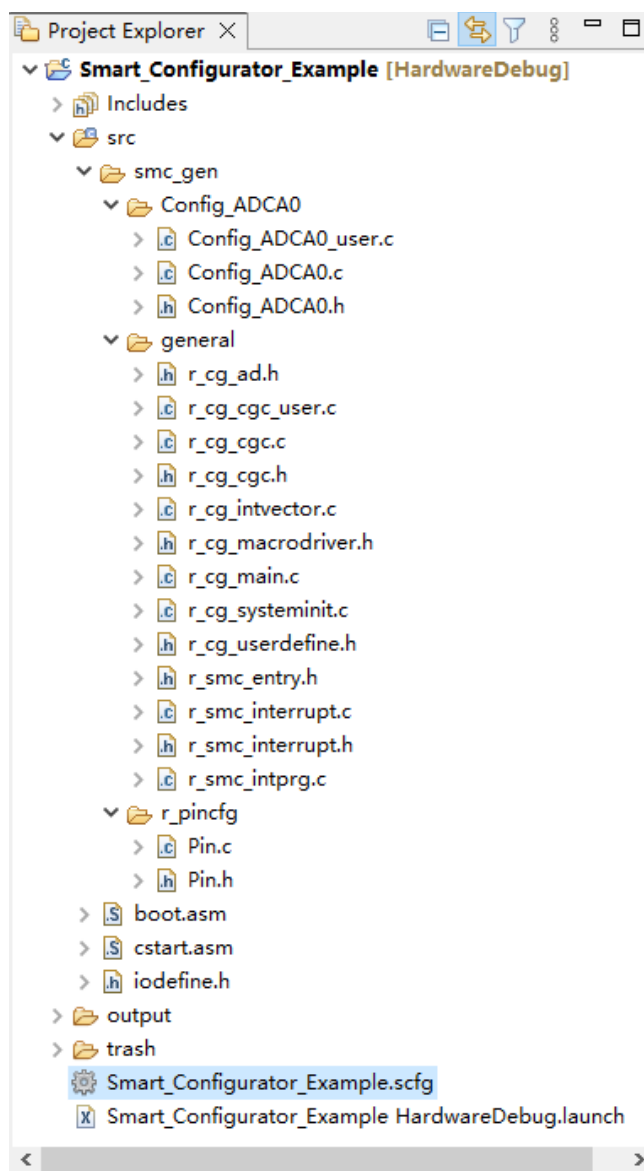


Figure 6-2 Source Files in the Project Explorer

6.2 Change Generated Code Location

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

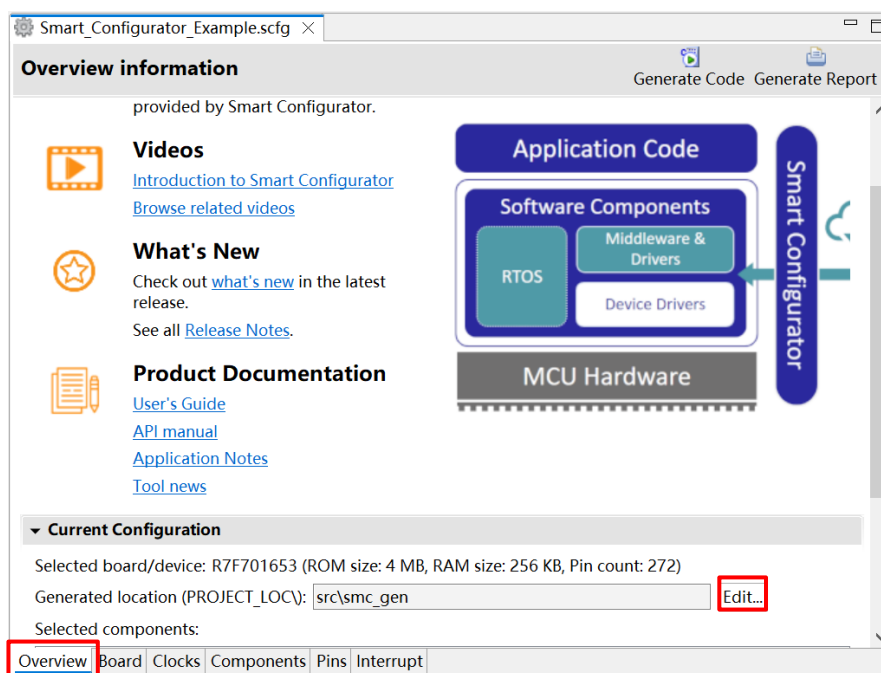


Figure 6-3 Edit the Generated Code Location

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

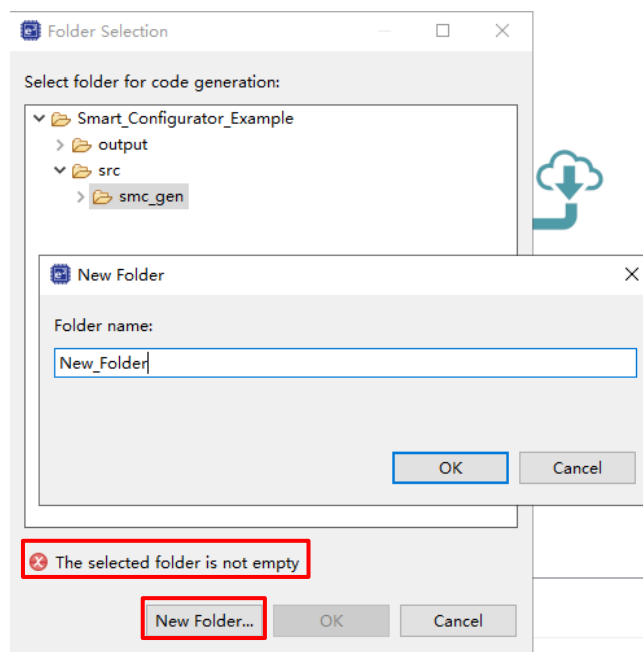



Figure 6-4 Folder Selection

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

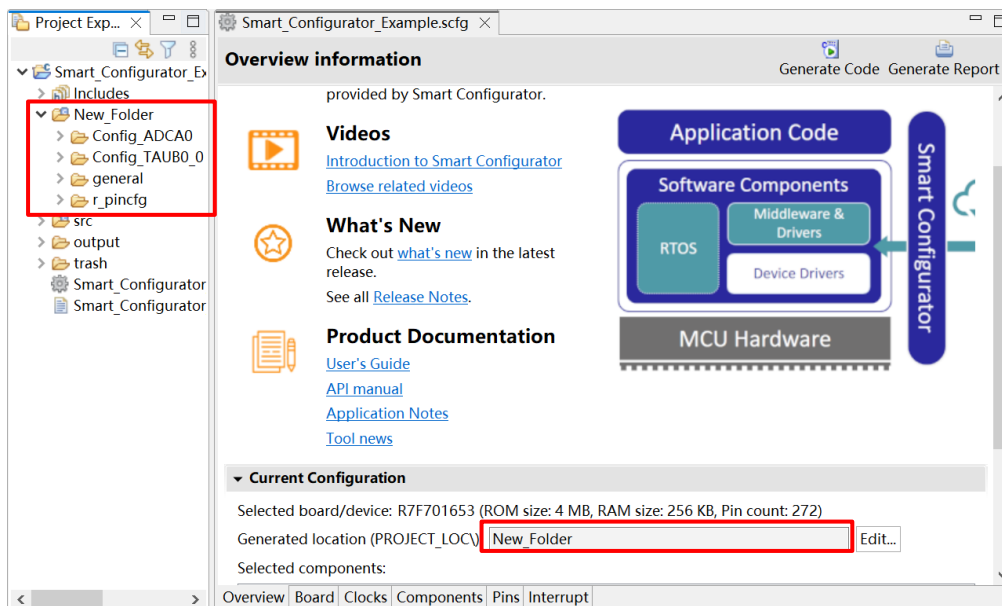


Figure 6-5 New Generate Code Location

6.3 Configuration of Generated Files and File Names

Figure 6-6 Configuration of Generated Files and File Names, shows the folders and files output by the Smart Configurator. Function `main ()` is included in `r_cg_main.c`, which is generated when the project is created by the e² studio.

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

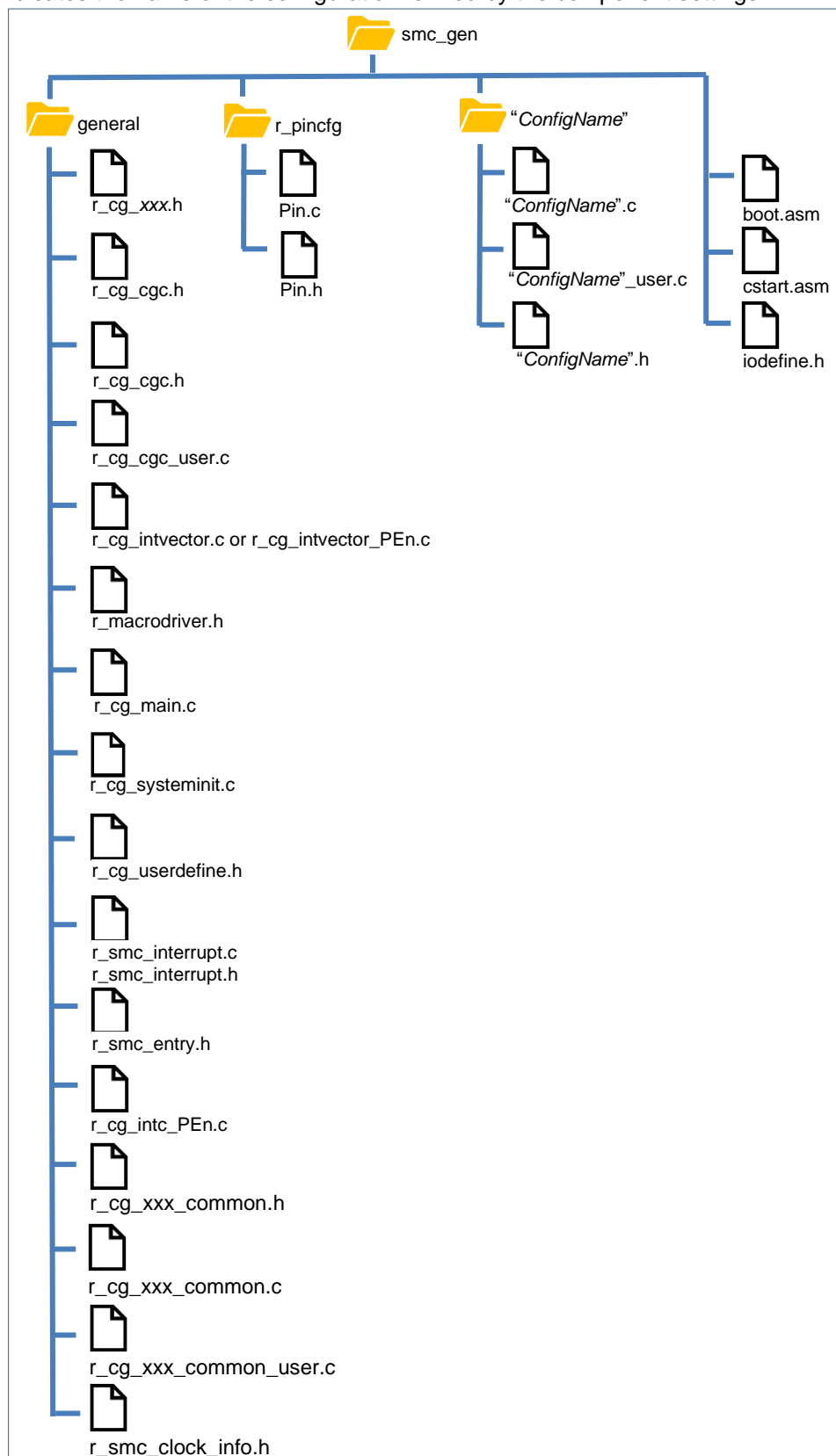


Figure 6-6 Configuration of Generated Files and File Names

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

Folder	File	Description
general	<i>r_cg_intc_PEn.c</i>	This file is generated only for: RH850/U2A: only when PE PEn(n=0~3) is chosen to be used, the r_cg_intc_PEn.c(n=0~3) is generated. RH850/C1M: r_cg_intc_PE1.c is generated. RH850/U2B: r_cg_intc_PE0.c is generated. This file contains interrupt initialization API definitions.
	<i>r_cg_xxx_common_user.c</i> ^(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.
	<i>r_cg_xxx_common.c</i> ^(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.
	<i>r_cg_xxx_common.h</i> ^(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.
	<i>r_smc_clock_info.h</i> ^(Note*2)	This file contains MCU information and clocks frequency define according to Clocks tabs setting.
r_pincfg	<i>Pin.h</i>	This file is always generated. It contains the function prototypes of pin settings in Pin.c, Symbolic name definition, Symbolic name user guide and Symbolic name API.
	<i>Pin.c</i>	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).
{ConfigName}	--	This folder is generated for the Code Generator drivers that are added to the project. API functions in this folder are named after the ConfigName (configuration name).
	{ConfigName}.c	This file contains functions to initialize driver (R_ConfigName_Create) and perform operations that are driver-specific, for e.g. start (R_ConfigName_Start) and stop (R_ConfigName_Stop).
	{ConfigName}_user.c	This file contains interrupt service routines and functions for user to add code after the driver initialization (R_ConfigName_Create). User can add codes and functions in the dedicated user code areas.
	{ConfigName}.h	This is header file for {ConfigName}.c and {ConfigName}_user.c

Folder	File	Description
smc_gen	-	This folder is always generated. It contains all the header files and source files generated by RH850 Smart Configurator.
	<i>boot.asm</i>	This file is automatically generated after creating the project. It contains some initialization assembly code required for device startup.
	<i>cstart.asm</i>	This file is automatically generated after creating t project. It contains some initialization assembly code required for device CPU Core startup.
	<i>iodefine.h</i>	This file is automatically generated after creating the project. It contains the description and definition of all registers information of the device.

Notes: 1. xxx is the name of a peripheral function.
2. This file is only generated in the RH850/U2B project.

6.4 Initializing Clocks

Configurations of the clock sources in the [Clocks] page are generated to the macros in the `r_cg_cgc.h` file located in `\src\smc_gen\general` folder.

For Smart Configurator RH850/U2B, the `r_smc_clock_info.h` file contains MCU information and clocks frequency define according to Clocks tabs setting.

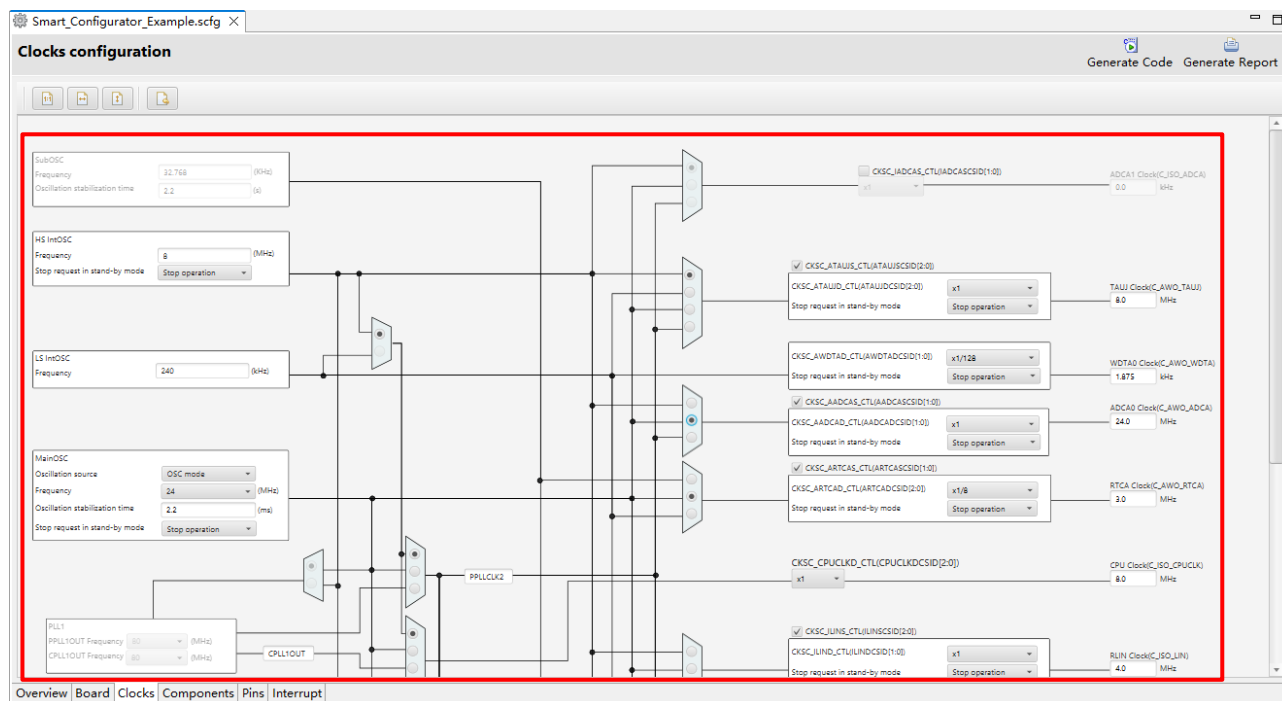


Figure 6-7 Clocks Configuration with Main Clock Selected as Clock Source

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

6.5 Initializing Pins

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

(1) Pin initialization for drivers with {ConfigName}

Pin functions are initialized in *R_{ConfigName}_Create* of the file *\src\smc_gen\{ConfigName}\{ConfigName}.c*.

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

Folder	File	Function	Description
{ConfigName}	{ConfigName}.c	<i>R_{ConfigName}_Create</i>	This API function initializes the pins used by this driver. <i>R_Systeminit</i> in <i>r_cg_systeminit.c</i> will call this function before entering main () function.

(2) Reference to pin initialization codes

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

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

6.6 Initializing Interrupts

Configurations in the [Interrupts] page are generated in some source files.

RH850/C1M, F1KM, F1KH and U2B:									
Vector Number	Exception Sou...	Interrupt	Interrupt request s...	Periph.	Priority	Status	OS management	Interrupt Handler	Generate Entity
568	1238H	INTMSPIORX1	MSPI0 Receive stat...	MSPI0	Lowest			uint568	
RH850/U2A:									
Vector...	Exceptio...	Interrupt	Interrupt request source	Peripheral	Priority	Status	OS managem...	Interrupt Handler	Generate Entity
39	1027H	INTDTS31TO0	DTS ch31-0 transfer end	DTS	Level 4			uint39	

Figure 6-8 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	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	The interrupt handler displayed on [Interrupt Handler] will be generated in file "r_smc_intprg.c" if [Generate Entity] is checked.
(4)	Generate Enable/Disable Function	general	r_smc_interrupt.c r_smc_interrupt.h	Interrupt enable/disable functions will be generated in r_smc_interrupt.c if [Generate Enable/Disable Function] is checked.
(5)	PE _n (the UI setting is only for RH850/U2A)	general	r_cg_intc_PEn.c	Interrupt binding is initialized in R_Interrupt_Initialize_ForPE in this file. R_Systeminit in r_cg_systeminit.c will call this function during execution of the main() function.

7. Creating User Programs

The Smart Configurator can add custom code to the output source files. This chapter describes how to add custom code to the source files generated by the Smart Configurator.

7.1 Adding Custom Code

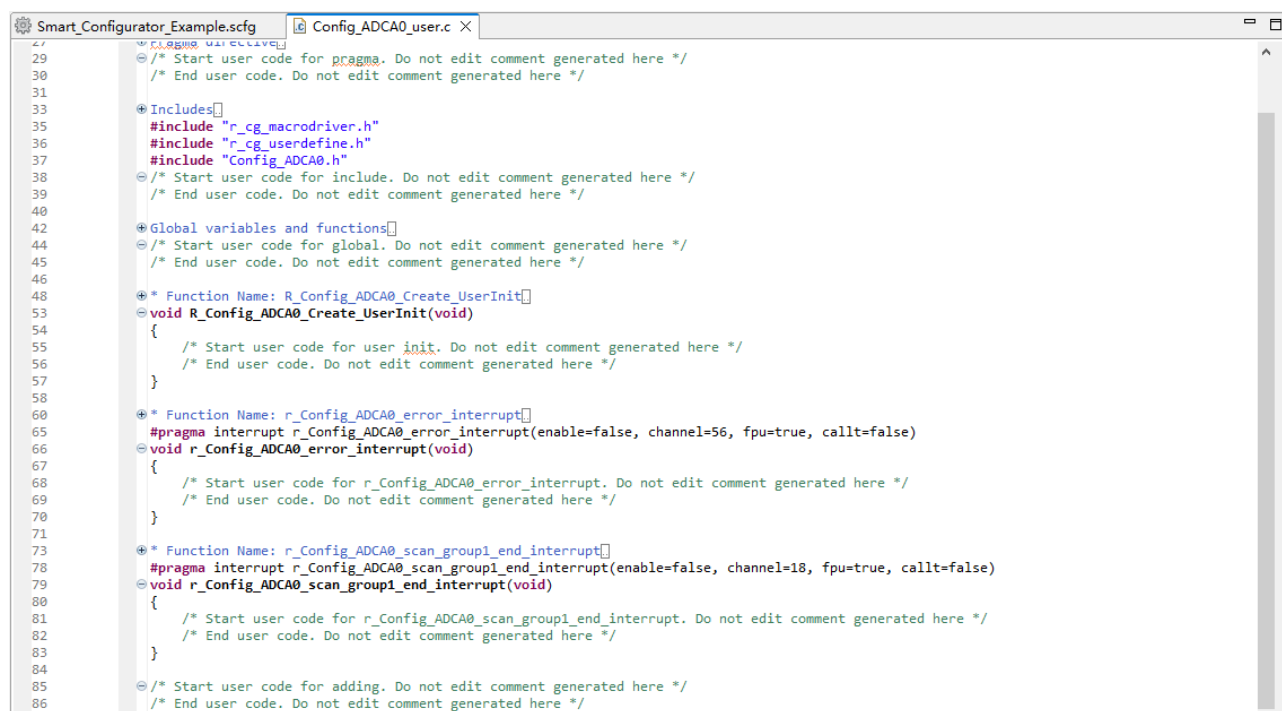
When creating component configuration, if files which have the same name already exist, new code will be merged only with the existing code that is between the comments below.

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

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

Figure 7-1 User Code Comments

In the RH850 Smart Configurator, 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 “ADCA0” for the A/D Converter (resource ADCA0). The comments to indicate where to add custom code are at the start and end of *.c files, and at the end of *.h file. 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 example is for ADCA0 (Config_ADCA0_user.c).



```
27
28
29 /* Start user code for pragma. Do not edit comment generated here */
30 /* End user code. Do not edit comment generated here */
31
32
33 #Includes[]
34 #include "r_cg_macrodriver.h"
35 #include "r_cg_userdefine.h"
36 #include "Config_ADCA0.h"
37
38 /* Start user code for include. Do not edit comment generated here */
39 /* End user code. Do not edit comment generated here */
40
41
42 #Global variables and functions[]
43
44 /* Start user code for global. Do not edit comment generated here */
45 /* End user code. Do not edit comment generated here */
46
47
48 /* Function Name: R_Config_ADCA0_Create_UserInit[]
49 void R_Config_ADCA0_Create_UserInit(void)
50 {
51     /* Start user code for user init. Do not edit comment generated here */
52     /* End user code. Do not edit comment generated here */
53 }
54
55
56 /* Function Name: r_Config_ADCA0_error_interrupt[]
57 #pragma interrupt r_Config_ADCA0_error_interrupt(enable=false, channel=56, fpu=true, callt=false)
58 void r_Config_ADCA0_error_interrupt(void)
59 {
60     /* Start user code for r_Config_ADCA0_error_interrupt. Do not edit comment generated here */
61     /* End user code. Do not edit comment generated here */
62 }
63
64
65 /* Function Name: r_Config_ADCA0_scan_group1_end_interrupt[]
66 #pragma interrupt r_Config_ADCA0_scan_group1_end_interrupt(enable=false, channel=18, fpu=true, callt=false)
67 void r_Config_ADCA0_scan_group1_end_interrupt(void)
68 {
69     /* Start user code for r_Config_ADCA0_scan_group1_end_interrupt. Do not edit comment generated here */
70     /* End user code. Do not edit comment generated here */
71 }
72
73
74 /* Start user code for adding. Do not edit comment generated here */
75 /* End user code. Do not edit comment generated here */
76
```

Figure 7-2 ADCA0 Code Generated Example

7.2 Using Generated Code in User Application

To use the generated code of Code Generator, follow the below description:

Open the `r_cg_main.c` file, call the functions generated and add application codes in the main function.

Driver initialization functions (`R_ConfigName_Create`) including initialization of pins have been called in `R_Systeminit` function of `r_cg_systeminit.c` by default.

User just needs to add application codes to perform operations that are driver-specific, for e.g., start (`R_ConfigName_Start`) and stop (`R_ConfigName_Stop`).

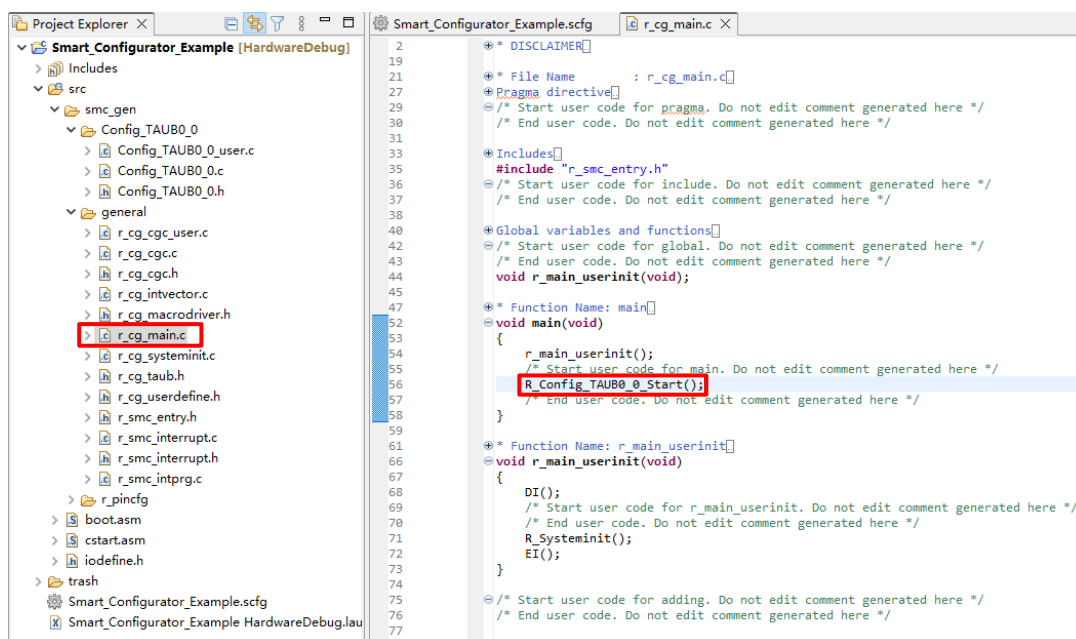



Figure 7-3 Call Code Generator Functions

8. Backing up Generated Source Code

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

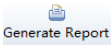
<ProjectDir>\trash\<Date-and-Time>

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

9. Generating Reports

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

9.1 Report on All Configurations (PDF or Text File)

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).

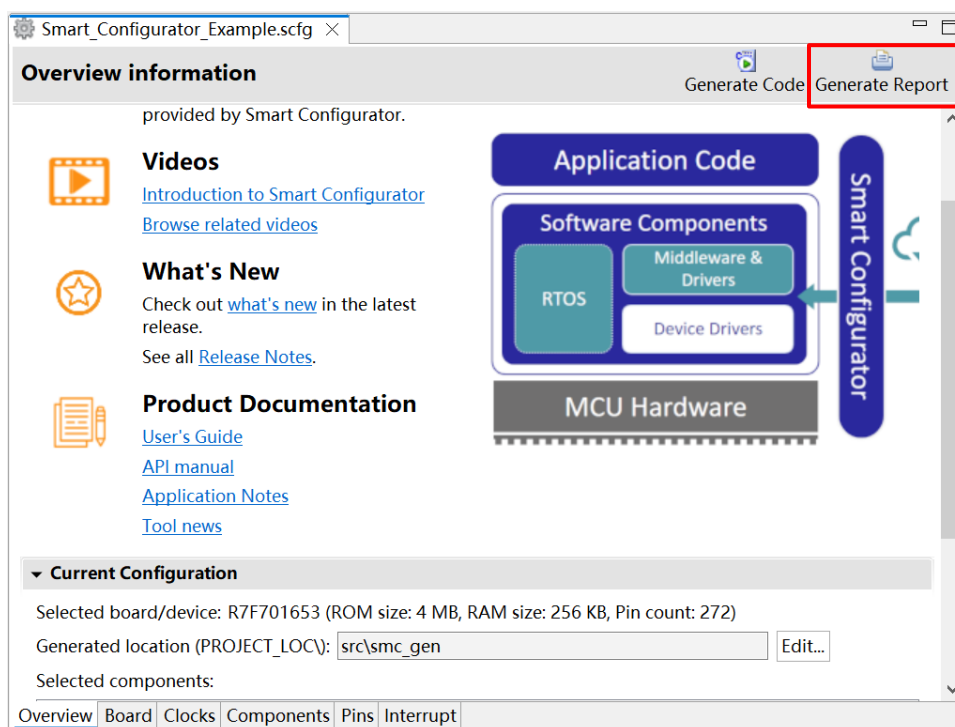


Figure 9-1 Output of a Report on the Configuration (as a PDF/Text File)

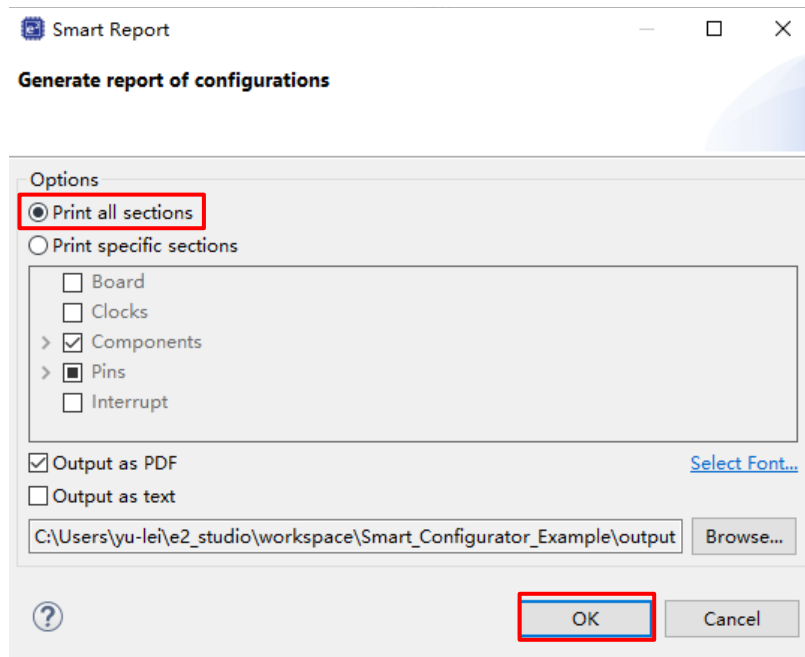



Figure 9-2 Dialog Box for Output of a Report (Example is selecting “Output as PDF”)

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

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

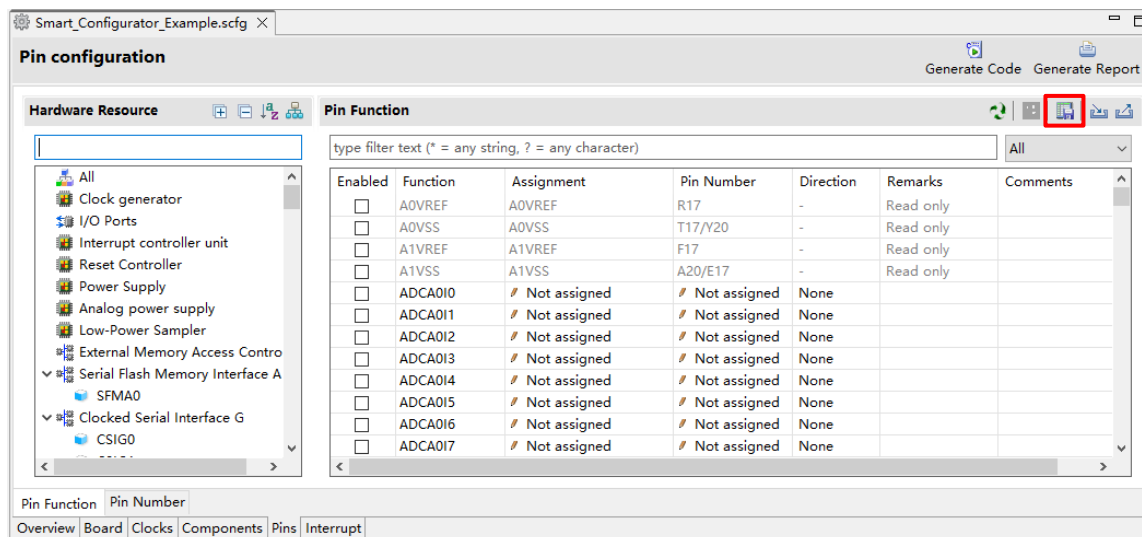



Figure 9-3 Output of a List of Pin Functions or Numbers (in csv Format)

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

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

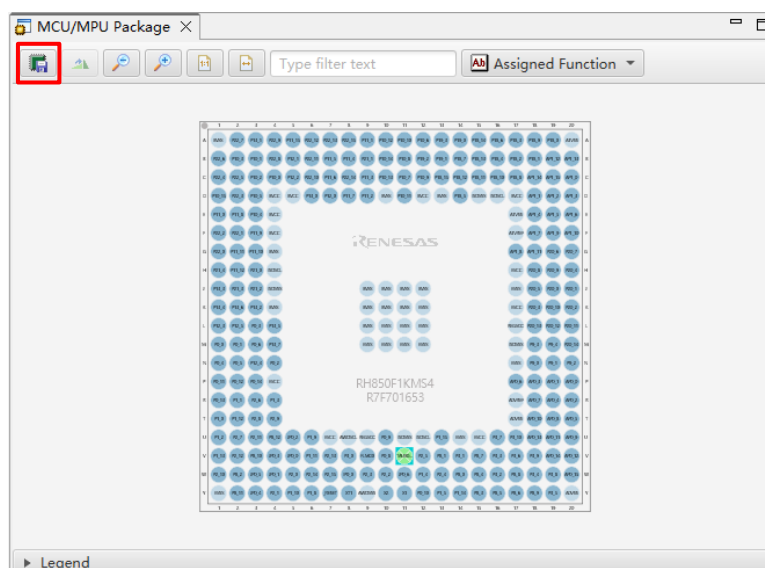


Figure 9-4 Outputting a Figure of MCU/MPU Package (in png Format)

10. User Code Protection Feature for Smart Configurator Code Generation Component

The Smart Configurator for RH850 Plug-in now incorporates an enhanced user code protection feature. This feature empowers users to insert codes to any location in the generated codes by utilizing the specific tags, as shown in Figure 10-1. After the next code generation, the inserted user codes will be protected and automatically merged into the generated files.

The user code protection feature will not be supported in the Pin.c and Pin.h files. For detailed information about the files, please refer to chapter 6.3 Configuration of Generated Files and File Names.

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.

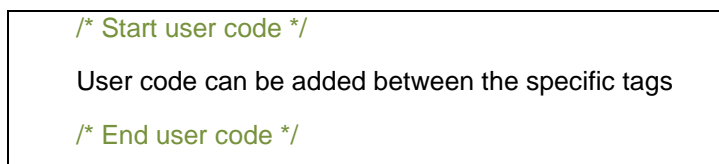


Figure 10-1 Specific Tags for User Code Protection Feature

10.2 Examples of Using User Code Protection Feature to Add New User Code

```

/* Function Name: R_Config_TAUB0_0_Create[]
@void R_Config_TAUB0_0_Create(void)
{
    /* Disable channel 0 counter operation */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* Disable INTTAUB010 operation and clear request */
    INT2.ICTAUB010.BIT.MKTAUB010 = _INT_PROCESSING_DISABLED;
    INT2.ICTAUB010.BIT.RFTAUB010 = _INT_REQUEST_NOT_OCCUR;
    /* Set INTTAUB010 setting */
    INT2.ICTAUB010.BIT.TBTAUB010 = _INT_TABLE_VECTOR;
    INT2.ICTAUB010.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK0_PRES_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRES_PCLK_15;
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_INDEPENDENT_CHANNEL |
    _TAUB_VALID_EDGE_SOFTWARE | _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE |
    _TAUB_START_INT_NOT_GENERATED;
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_FALLING;
    TAUB0.CDR0 = _TAUB00_COMPARE_VALUE;
    /* Start user code */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* End user code */
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Synchronization processing */
    g_cg_sync_read = TAUB0.TPS;
    _syncp();
    /* Set TAUB010 pin */
    PORT.PIBC0 &= _PORT_CLEAR_BIT7;
    PORT.PBDC0 &= _PORT_CLEAR_BIT7;
    PORT.PM0 |= _PORT_SET_BIT7;
    PORT.PMC0 &= _PORT_CLEAR_BIT7;
    PORT.PFC0 |= _PORT_SET_BIT7;
    PORT.PFCE0 |= _PORT_SET_BIT7;
    PORT.PFCAE0 &= _PORT_CLEAR_BIT7;
    PORT.PMC0 |= _PORT_SET_BIT7;
    DNF.ATAUB01ENL.BIT.ATAUB01ENL0 = _DIGITAL_FILTER_ENABLED;
    DNF.ATAUB01CTL = _DIGITAL_FILTER_SAMPLING_NUMBER_2 | _DIGITAL_FILTER_SAMPLING_CLOCK_1;
    R_Config_TAUB0_0_Create_UserInit();
}

```

```

/* Function Name: R_Config_TAUB0_0_Create[]
@void R_Config_TAUB0_0_Create(void)
{
    /* Disable channel 0 counter operation */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* Disable INTTAUB010 operation and clear request */
    INT2.ICTAUB010.BIT.MKTAUB010 = _INT_PROCESSING_DISABLED;
    INT2.ICTAUB010.BIT.RFTAUB010 = _INT_REQUEST_NOT_OCCUR;
    /* Set INTTAUB010 setting */
    INT2.ICTAUB010.BIT.TBTAUB010 = _INT_TABLE_VECTOR;
    INT2.ICTAUB010.UINT16 &= _INT_PRIORITY_LEVEL4;
    TAUB0.TPS &= _TAUB_CK0_PRES_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRES_PCLK_15;
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_INDEPENDENT_CHANNEL |
    _TAUB_VALID_EDGE_SOFTWARE | _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE |
    _TAUB_START_INT_NOT_GENERATED;
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_RISING;
    TAUB0.CDR0 = _TAUB00_COMPARE_VALUE;
    /* Start user code */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* End user code */
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Synchronization processing */
    g_cg_sync_read = TAUB0.TPS;
    _syncp();
    SL_TAUB0.SELB_TAUB01 |= _TAUB_INPUT_SIGNAL_CH0_TAUB11;
    /* Set TAUB011 pin */
    PORT.PIBC10 &= _PORT_CLEAR_BIT11;
    PORT.PBDC10 &= _PORT_CLEAR_BIT11;
    PORT.PM10 |= _PORT_SET_BIT11;
    PORT.PMC10 &= _PORT_CLEAR_BIT11;
    PORT.PPC10 &= _PORT_CLEAR_BIT11;
    PORT.PFC10 |= _PORT_SET_BIT11;
    PORT.PFCE10 &= _PORT_CLEAR_BIT11;
    PORT.PMC10 |= _PORT_SET_BIT11;
    DNF.ATAUB01ENL.BIT.ATAUB01ENL1 = _DIGITAL_FILTER_ENABLED;
    DNF.ATAUB01CTL = _DIGITAL_FILTER_SAMPLING_NUMBER_2 | _DIGITAL_FILTER_SAMPLING_CLOCK_32;
    R_Config_TAUB0_0_Create_UserInit();
}

```

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

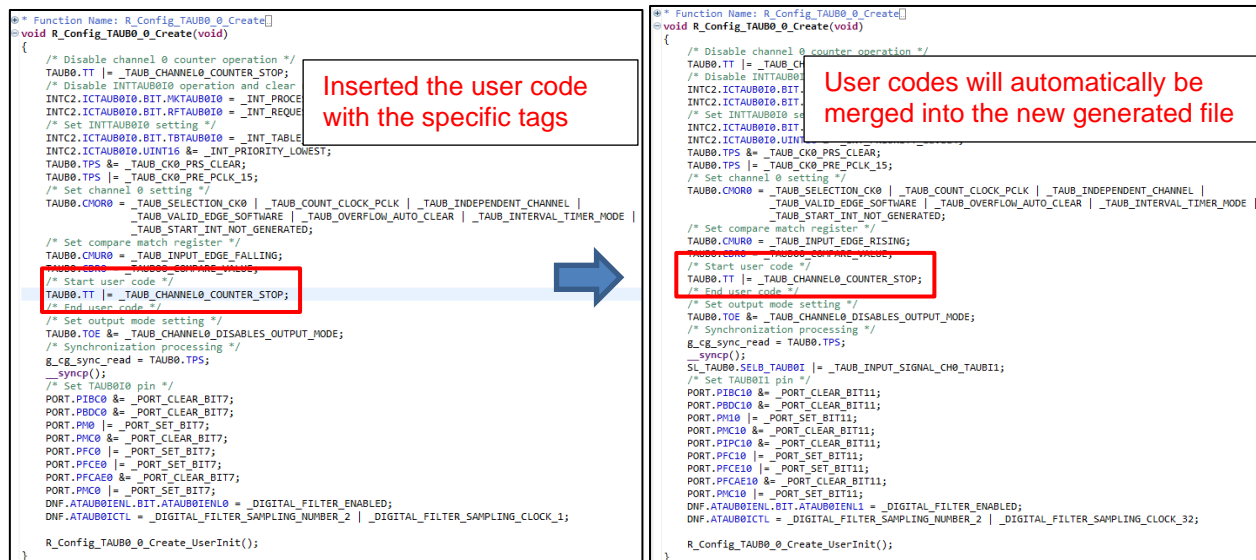


Figure 10-2 User Code Protection with Auto Merge

10.3 What to Do When Merge Conflict Occurs

10.3.1 What is Merge Conflict

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

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

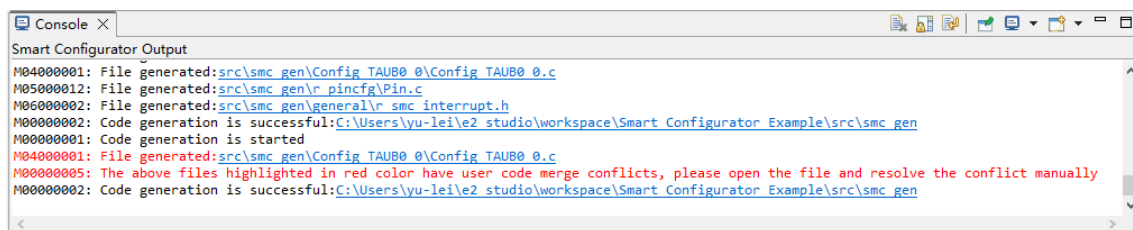


Figure 10-3 The Merge Conflict Message Outputted in the Smart Configurator Console

User can click the conflicted file in the console message to open the File Compare view and then can resolve the conflict as next chapter 10.3.2 Steps for Resolving the Merge Conflict described.

10.3.2 Steps for Resolving the 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).

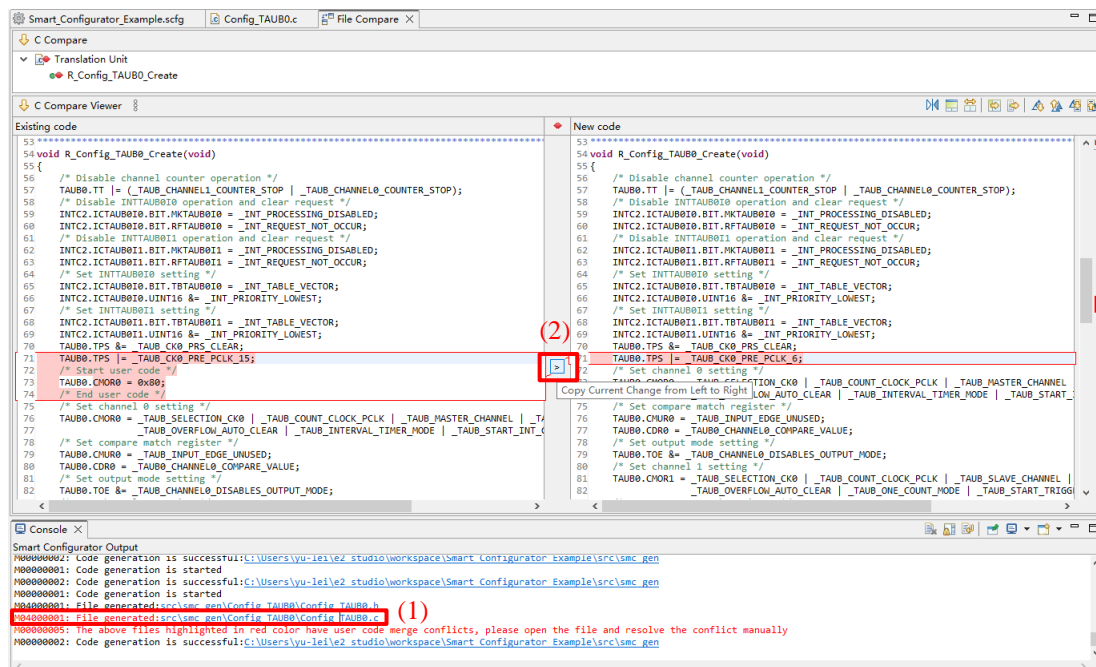


Figure 10-4 Code before Resolving Conflict

- (3) Delete the codes that user does not want to use (Figure 10-5).

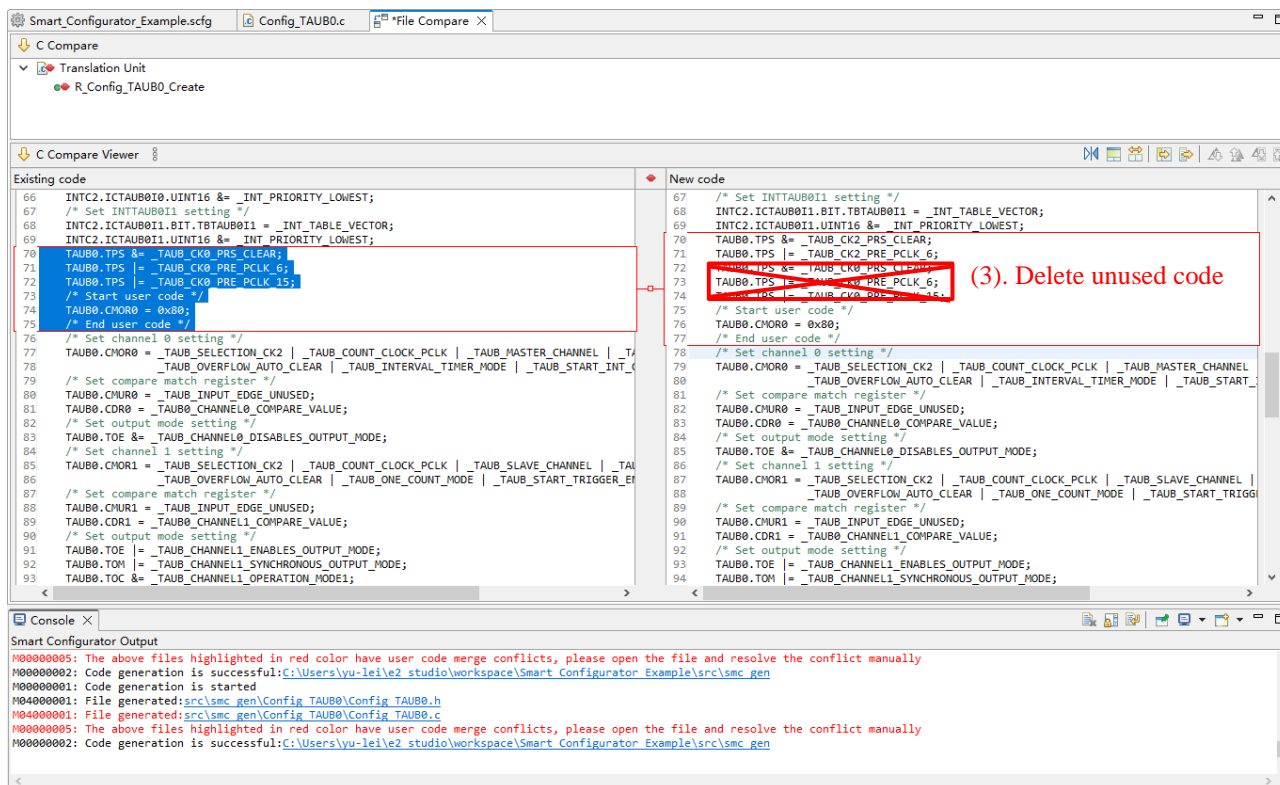


Figure 10-5 Code after Applying “Copy Current Change from Left to Right”

(4) Save the modified code (Figure 10-6).

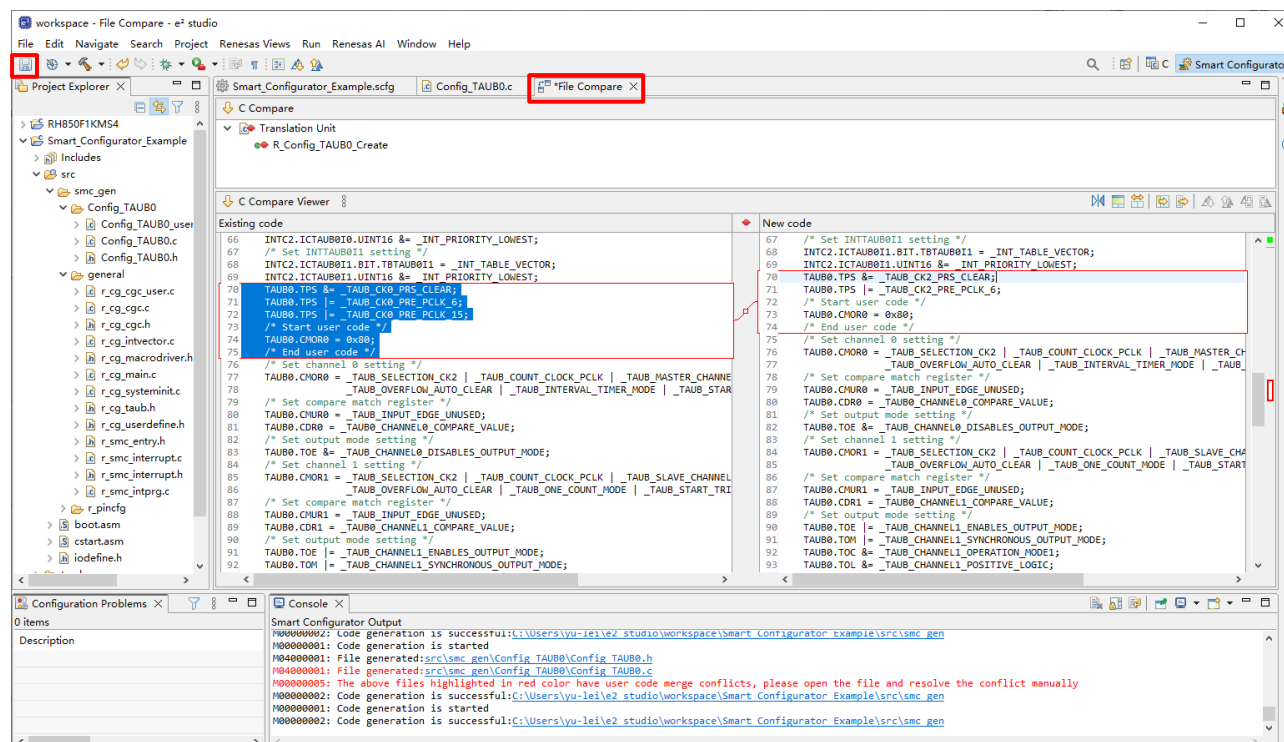


Figure 10-6 Code after Deleting and Saving

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

Note: After confliction resolved, if click the confliction message, it still can open [File Compare] view.

11. Help

Refer to the help system from the e² studio menu for detailed information on the Smart Configurator. If selected from Help menu, it will prompt out the Help dialog window, it shows all Help topics content.

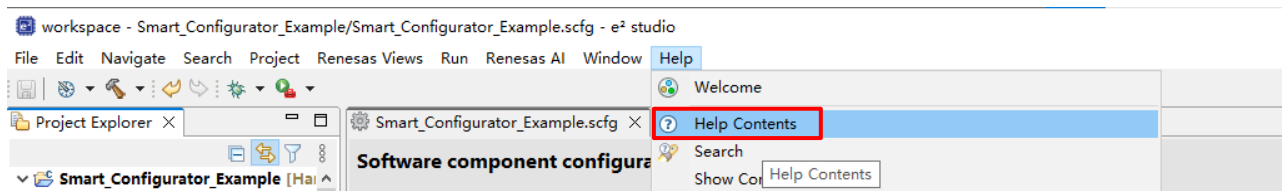



Figure 11-1 Help Menu

The help system can also be activated from the [Overview] page by clicking  button. If selected from this page, it will open a Help panel in the current GUI view, it is specially pointing to Smart Configurator portion in Help content.

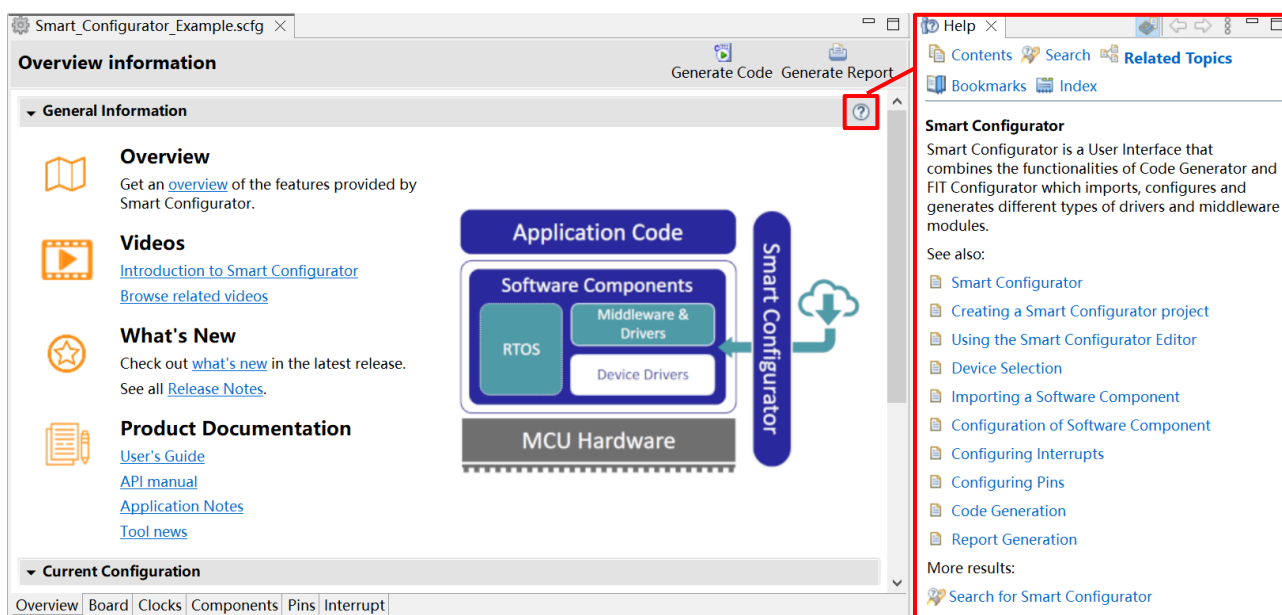


Figure 11-2 Smart Configurator Help Quick Start

In both ways to check Help information, the whole Help contents is the same.

12. Documents for Reference

User's Manual: Hardware

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

Technical Update/Technical News

Obtain the latest information from the Renesas Electronics website.

User's Manual: Development Environment

e2 studio Integrated Development Environment User's Manual: Getting Started Guide (R20UT2771)

CC-RH Compiler User's Manual (R20UT3516)

Smart Configurator User's Manual: RH850 API Reference (R20UT4361)

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

Revision History

Rev.	Section	Description
1.00	-	First edition issued

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

2. Processing at power-on

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

3. Input of signal during power-off state

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

4. Handling of unused pins

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

5. Clock signals

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

6. Voltage application waveform at input pin

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

7. Prohibition of access to reserved addresses

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

8. Differences between products

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

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

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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