
RL78 Smart Configurator

R20AN0579EC0103

Rev.1.03

User's Guide: e² studioJul.20.23

Introduction

This application note describes the basic usage of the RL78 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 2023-07 and later

Target Devices and Compilers

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

<https://www.renesas.com/rl78-smart-configurator>

Contents

1. Overview	4
1.1 Purpose	4
1.2 Features	4
1.3 Software Components.....	4
2. Creating a Project.....	5
3. Operating the Smart Configurator.....	10
3.1 Displaying the Smart Configurator Perspective	10
3.2 Procedure for Operations.....	11
3.3 File to be Saved as Project Information	12
3.4 Window.....	12
3.4.1 Project Explorer.....	13
3.4.2 Smart Configurator View.....	13
3.4.3 MCU/MPU Package View	14
3.4.4 Console View	15
3.4.5 Configuration Problems View.....	15
3.4.6 Developer Assist Browser View	16
4. Setting of Peripheral Modules.....	17
4.1 Board Settings.....	17
4.1.1 Selecting the Device	17
4.1.2 Selecting the Board.....	17
4.1.3 Exporting Board Settings	19
4.1.4 Importing Board Settings	20
4.2 Clock Settings	21
4.3 System Settings	22
4.4 Component Settings.....	24
4.4.1 Switching Between the Component View and Hardware View	24
4.4.2 Adding a Software Component	25
4.4.3 Removing Software Component	27
4.4.4 Setting a Code Generator Component	28
4.4.5 Changing the Resource for a Code Generator Configuration	29
4.4.6 Setting SNOOZE Mode Sequencer (SMS) Component	31
4.4.7 Update SMS Data Files	34
4.4.8 Logic Event Link Controller (ELCL) Modules Download.....	35
4.4.9 Setting an ELCL Component	36
4.4.10 Downloading RL78 Software Integration System Modules	37
4.4.11 Adding a RL78 Software Integration System Module.....	39
4.4.12 Setting a RL78 Software Integration System Module	40
4.4.13 Changing Version of BSP Configuration.....	41
4.4.14 Export Component Configuration.....	43
4.4.15 Import Component Configuration.....	43
4.4.16 Configure General Setting of the Component.....	44
4.5 Pin Settings	46
4.5.1 Changing the Pin Assignment by PIOR Function	47

4.5.2	Changing the Pin Assignment of a Software Component	48
4.5.3	Assigning Pins Using the MCU/MPU Package View	49
4.5.4	Show Pin Number from Pin Functions	50
4.5.5	Exporting pin settings.....	51
4.5.6	Importing pin settings.....	51
4.5.7	Pin setting using board pin configuration information	52
4.5.8	Pin Filter Feature.....	53
4.5.9	Pin Errors/Warnings setting	54
4.6	Interrupt Settings.....	55
4.6.1	Changing Interrupt Priority Setting.....	55
4.6.2	Changing Interrupt Bank Setting.....	56
4.7	MCU Migration Feature.....	57
5.	Managing Conflicts.....	61
5.1	Resource Conflicts	61
5.2	Resolving Pin Conflicts	62
6.	Generating Source Code.....	63
6.1	Outputting Generated Source Code	63
6.2	Change Generated Code Location	64
6.3	Configuration of Generated Files and File Names.....	66
6.4	Initializing Clocks.....	69
6.5	Initializing Pins.....	70
6.6	Initializing Interrupts	71
7.	Creating User Programs.....	72
7.1	Adding Custom Code	72
7.2	Using Generated Code in User Application	74
8.	Backing up Generated Source Code	75
9.	Generating Reports	76
9.1	Report on All Configurations (PDF or Text File)	76
9.2	Configuration of Pin Function List and Pin Number List (in csv Format)	77
9.3	Image of MCU/MPU Package (in png Format)	77
10.	Developer Assistance.....	78
11.	User Code Protection Feature for Smart Configurator Code Generation Component	79
11.1	Specific Tags for the User Code Protection Feature	79
11.2	Examples of Using User Code Protection Feature to Add New User Code	79
11.3	What to Do When Merge Conflict Occurs	80
11.3.1	What is Merge Conflict.....	80
11.3.2	Steps for Resolving the Merge Conflict	81
12.	Help.....	83
13.	Documents for Reference.....	84

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.

1.3 Software Components

The Smart Configurator supports three types of software components: Code Generator, Graphical Configurator, and RL78 Software Integration System:

- (1) Code Generator drivers (DTC, A/D Converter, Interrupt Controller, etc.)
The Code Generator drivers is a control program for peripheral functions of microcomputer such as DTC, A/D converter, Interrupt Controller, etc. It is convenient to embed a software component using code generation function.
- (2) Graphical Configurator (SMS, ELCL)
The Graphical Configurator module makes it easy to set up complex configurations by providing a graphical GUI compared to other drivers. It provides software components for SNOOZE mode sequencer (SMS) and logic and event link controller (ELCL).
- (3) RL78 Software Integration System (CAPACITIVE SENSING UNIT (CTSU2L), etc.)
The RL78 Software Integration System module is a software component of drivers, middleware SW that provides a simple GUI for generating code.

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.

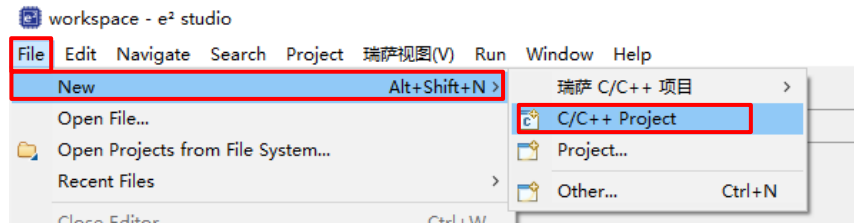


Figure 2-1 Creating a New Project

- (2) In the project creation wizard, select [Renesas RL78] → [Renesas CC-RL C Executable Project] or [LLVM for Renesas RL78 C/C++ Executable Project], and click on the [Next] button.

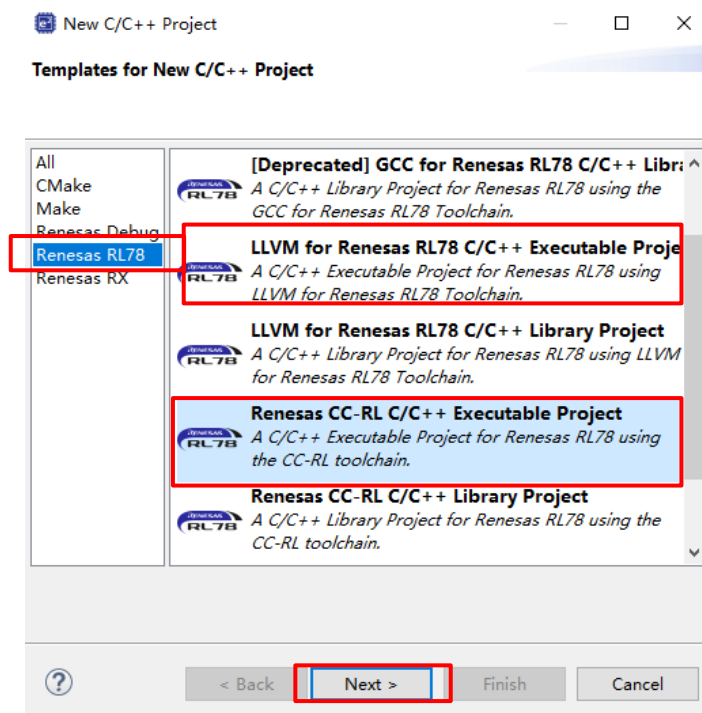


Figure 2-2 Templates for New C/C++ Project Dialog Box

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

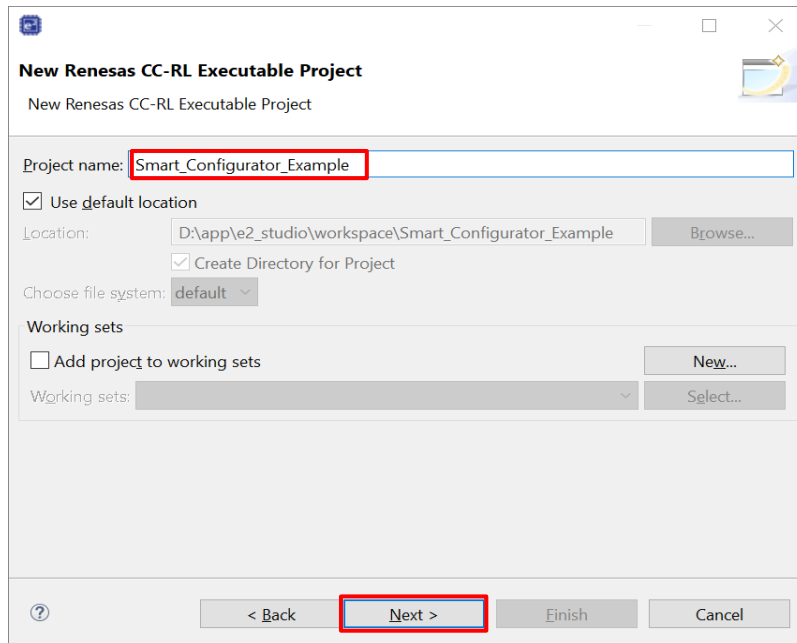


Figure 2-3 Creating a New Renesas CC-RL Executable Project

- (4) Select the toolchain, target board, device, and debug configuration. Click on the [Next] button.
 (For e.g., Target Device: RL78G23 – 128 pins (Part number: R7F100GSNxFB))

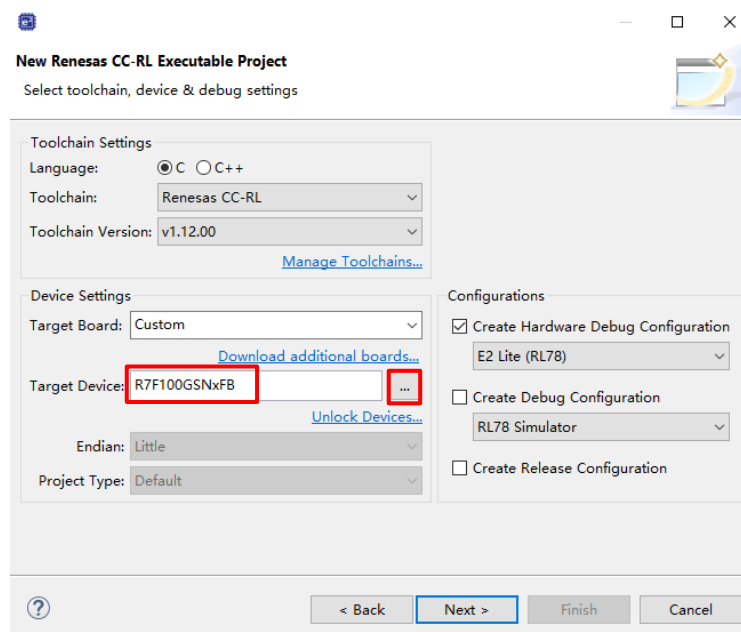


Figure 2-4 Selecting the Toolchain, Device, and Debug Configuration

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

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

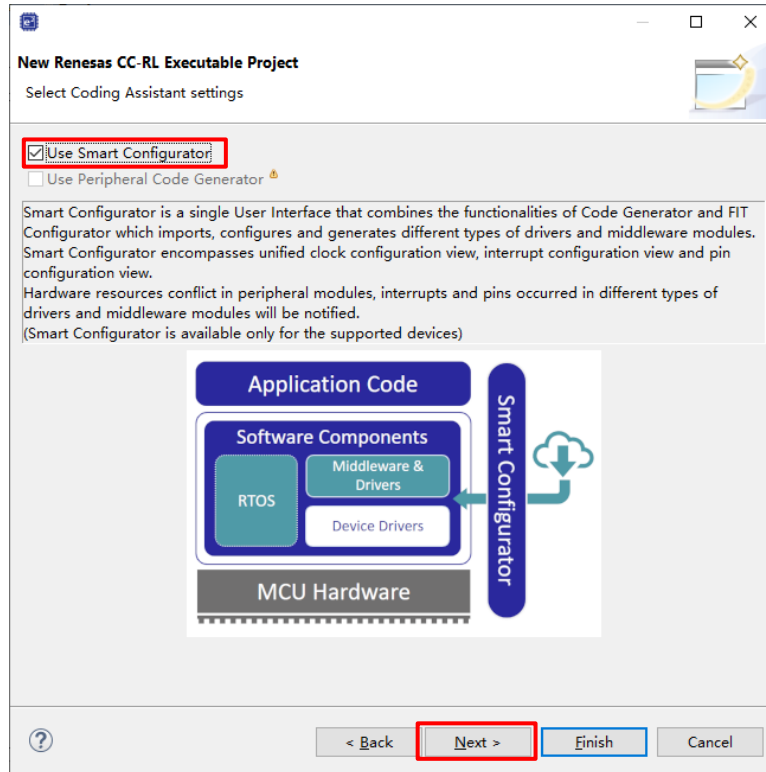


Figure 2-5 Selecting the Coding Assistant Tool

- (6) In the [Project Template Selection] dialog box, select the [Bare Metal - Blinky] or [Bare Metal - Minimal] project template and click on the [Finish] button.

Note: [Bare Metal - Blinky] selection is supported only if the target board selected at (4) is not [Custom] and the selected board have the LED resource(s). The custom board is only support [Bare Metal - Minimal] project template.

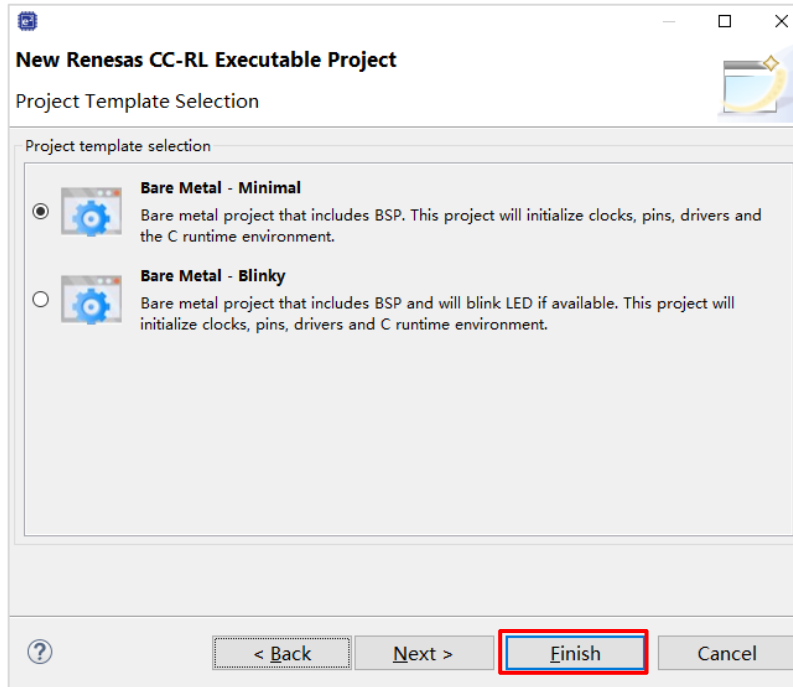


Figure 2-6 Selecting the Project Template

- (7) Wait for completion of project creation.

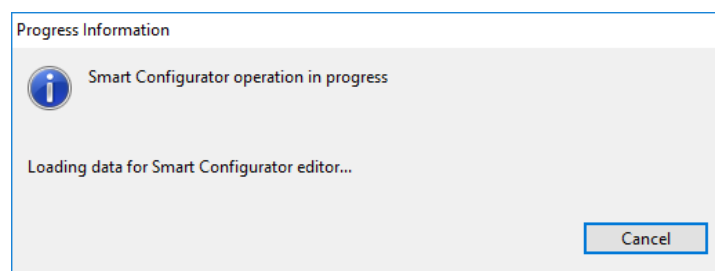


Figure 2-7 Progress of Project Creation

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

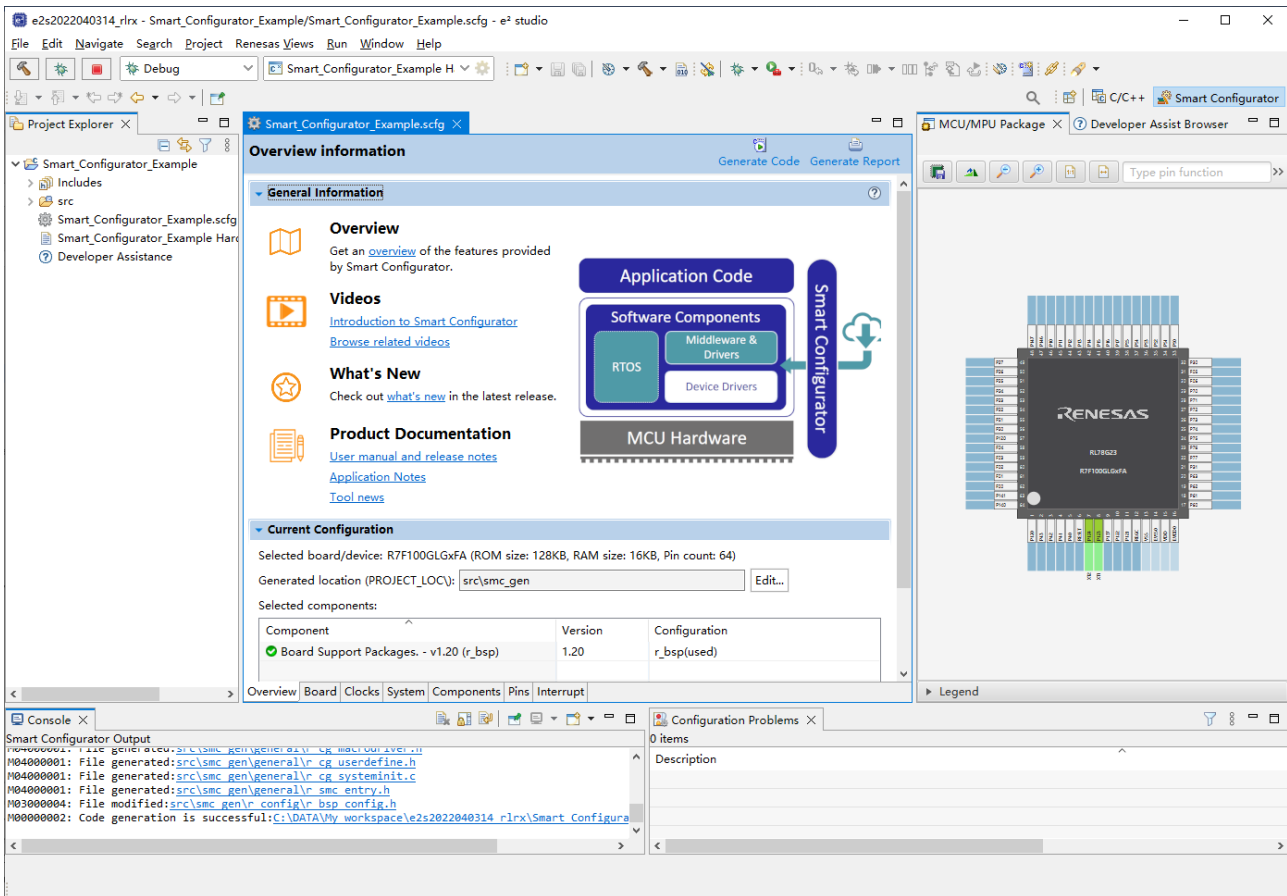


Figure 2-8 Smart Configurator Perspective

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:

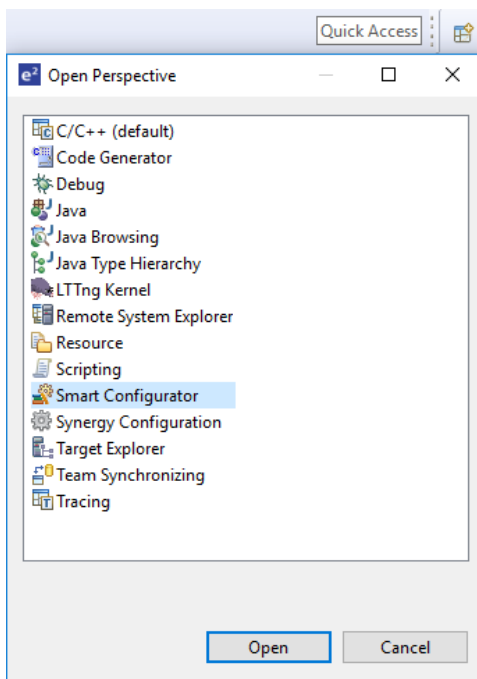


Figure 3-1 Opening the Smart Configurator Perspective

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.

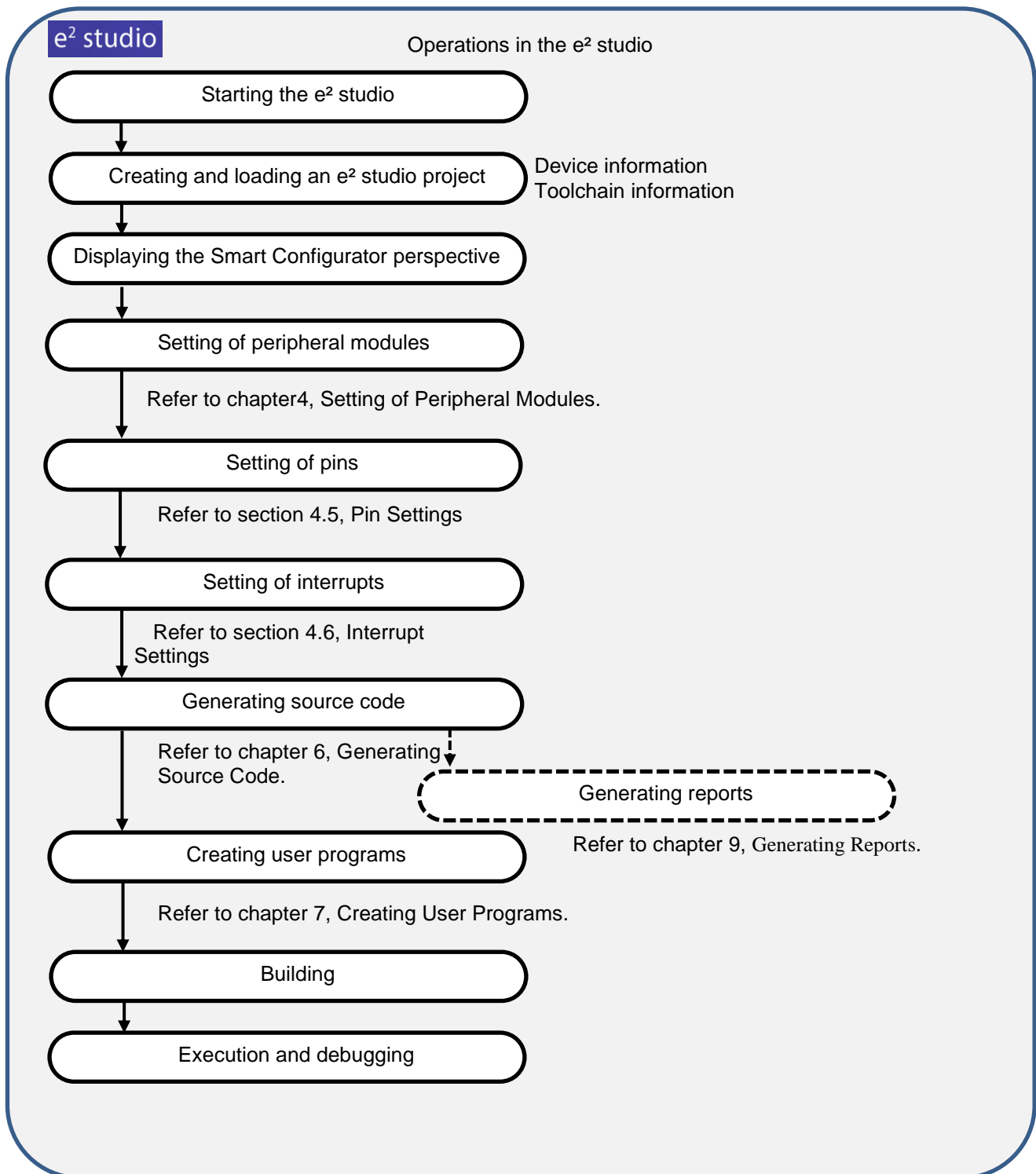


Figure 3-2 Procedure for Operations

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.

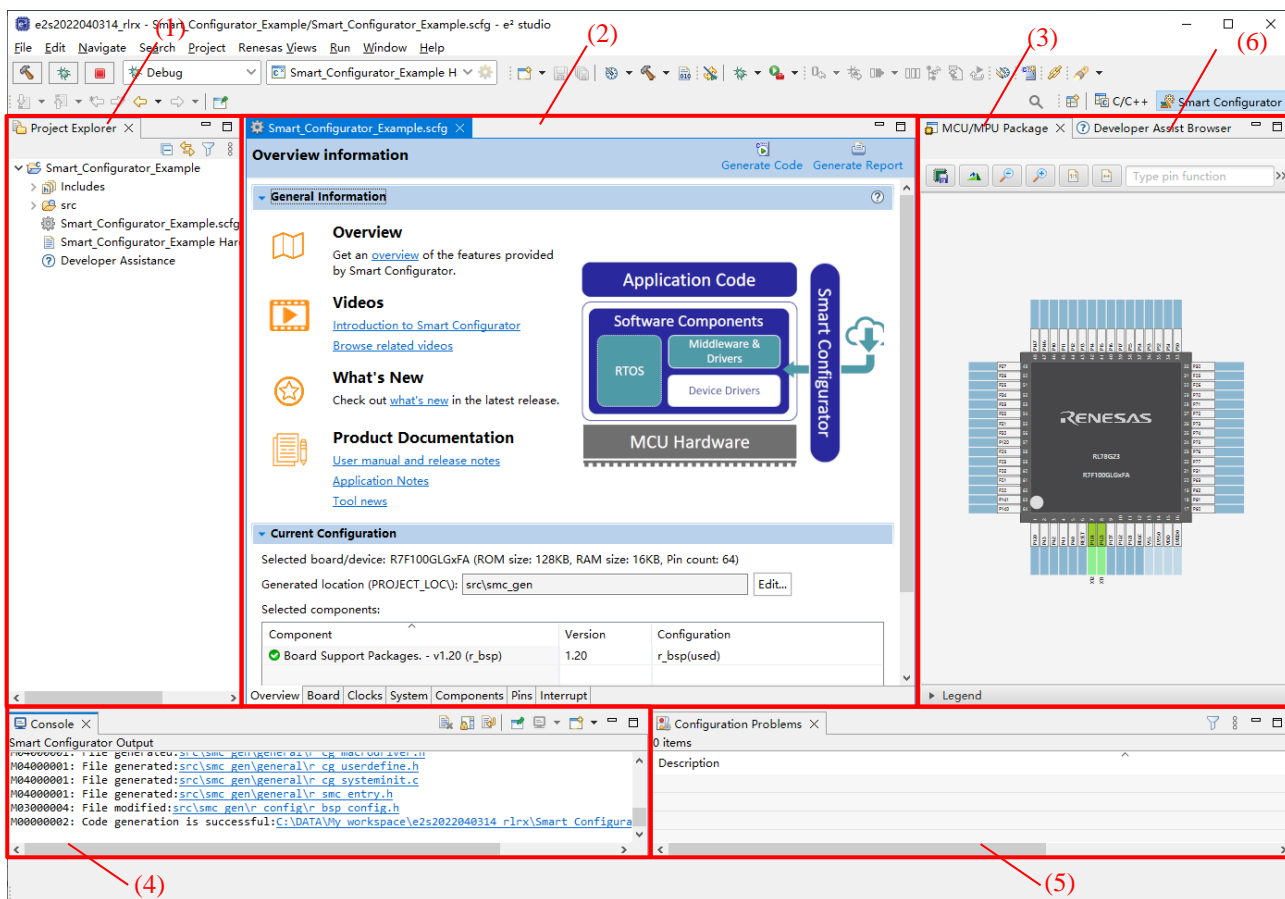


Figure 3-3 Smart Configurator Perspective

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

3.4.1 Project Explorer

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

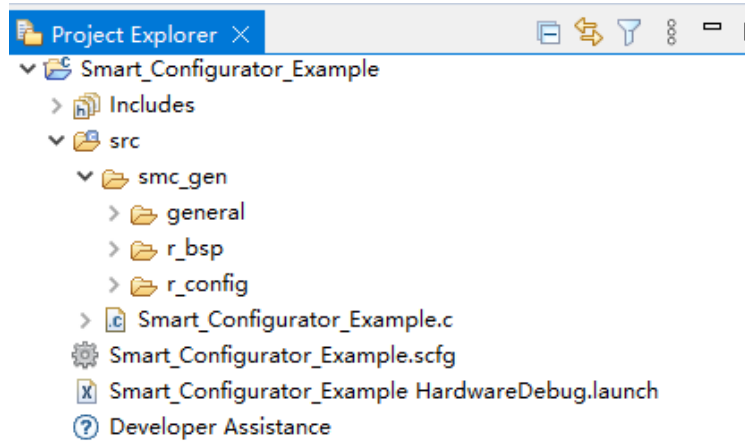


Figure 3-4 Project Explorer

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.

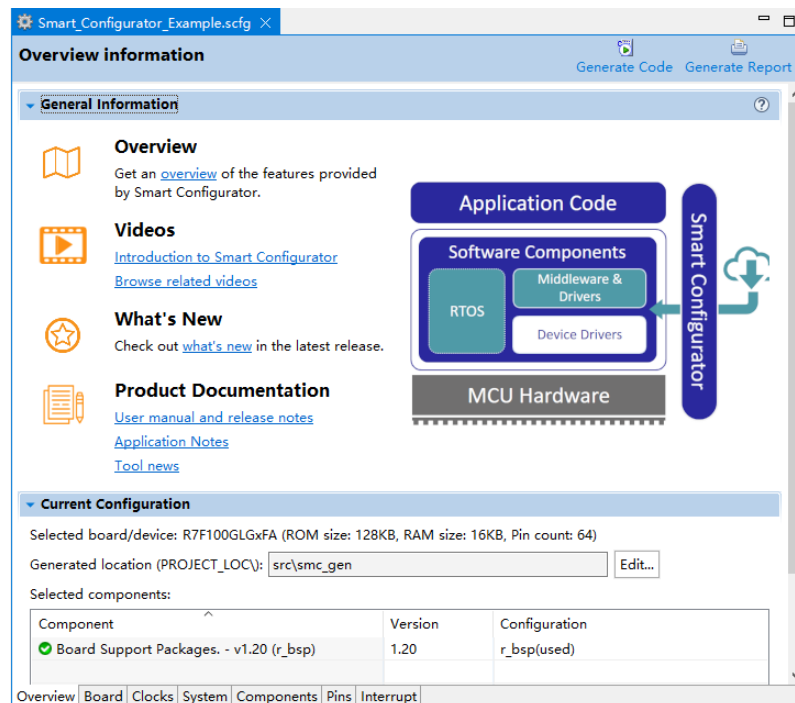


Figure 3-5 Smart Configurator View

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. 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 4.5.7 Pin setting using board pin configuration information").
- [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.

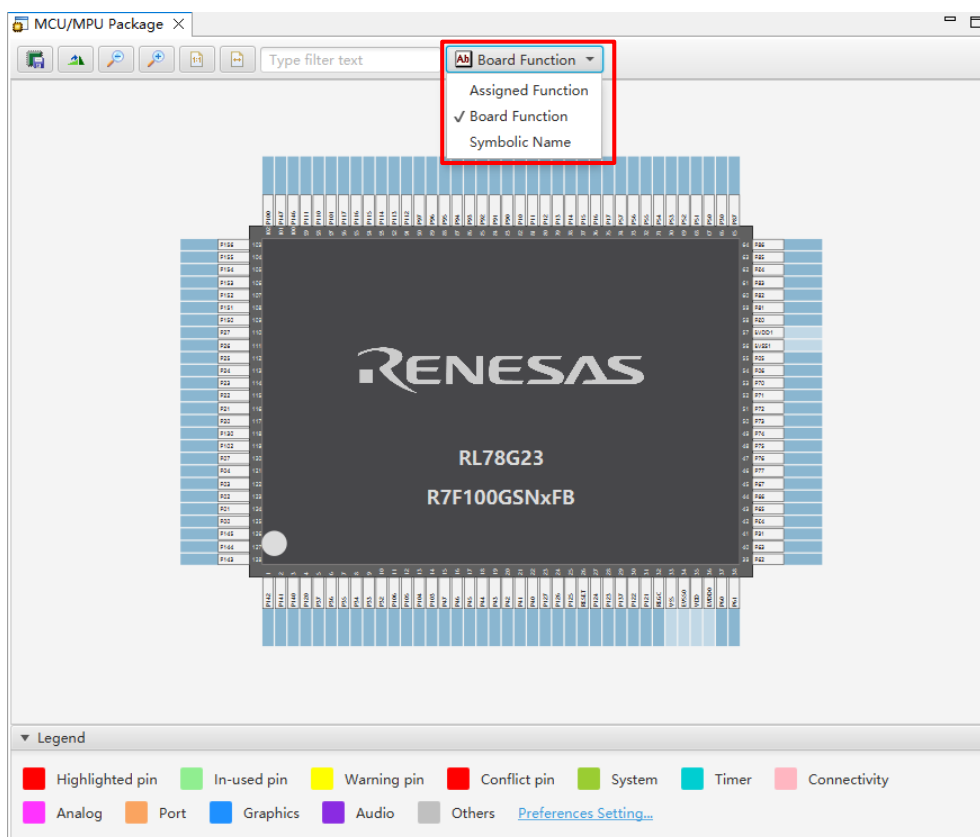


Figure 3-6 MCU/MPU Package View

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.

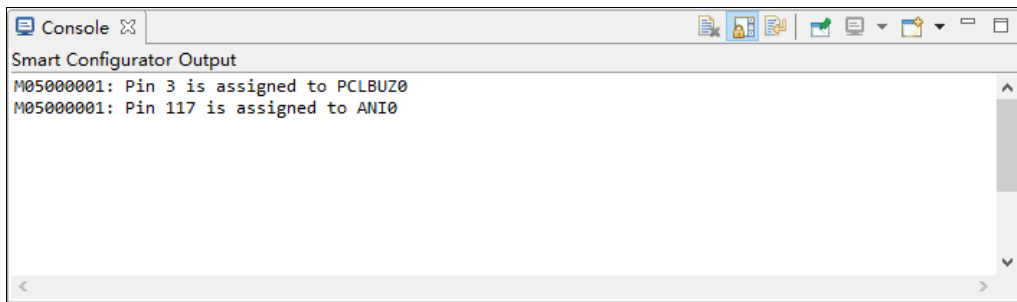


Figure 3-7 Console View

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.

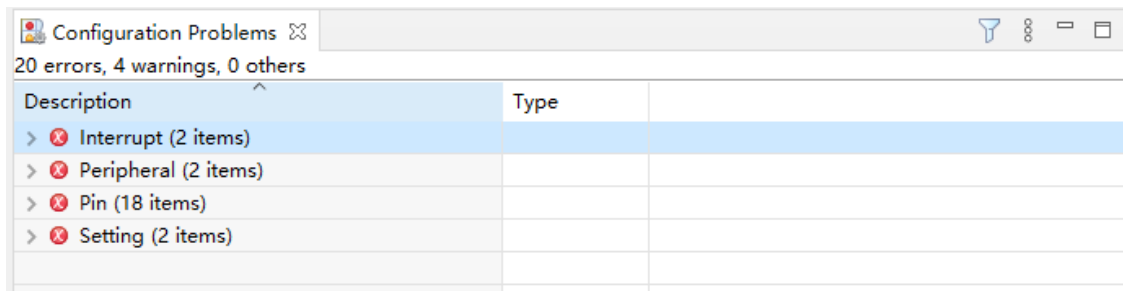


Figure 3-8 Configuration Problems View

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

3.4.6 Developer Assist Browser View

[Developer Assist Browser] view is a particular view service for Smart Configurator Developer Assistance feature, user can navigate and browse API info and through "Copy" context menu to paste the Code Generator component usage example code snippet to C/C++ editor.

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

The screenshot shows a window titled "MCU/MPU Package" and "Developer Assist Browser". The main content area is titled "General" and contains the text: "Below is a list of API functions output by the Smart Configurator for common use." Below this text is a table with three columns: "API Function Name", "Peripheral Name", and "Description".

API Function Name	Peripheral Name	Description
main	-	Main function.
R_Systeminit	-	Executes initialization processing that is required before controlling various peripheral modules.
R_DTC_Set_PowerOn	Data Transfer Controller	Starts the clock supply for DTC.
R_DTC_Set_PowerOff	Data Transfer Controller	Stops the clock supply for DTC.
R_TAUm_Create	Timer Array Unit	Executes initialization processing that is required before controlling TAUm (enables TAUm input clock supply and initializes TAUm module).
R_TAUm_Set_PowerOn		Starts the clock supply for TAUm.
R_TAUm_Set_PowerOff		Stops the clock supply for TAUm.
R_TAUm_Set_Reset		Sets TAUm module in reset state.
R_TAUm_Release_Reset		Releases TAUm module from reset state.
R_ITL_Create		Executes initialization processing that is required before controlling the 32-bits IT (enables input clock supply and initializes ITLm module).

Figure 3-9 Developer Assist Browser View

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.

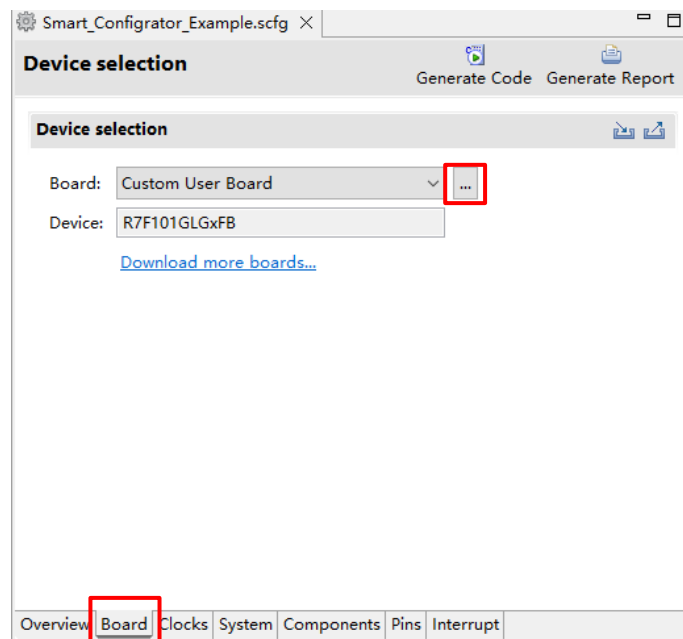


Figure 4-1 Selecting the Device

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 subsystem clock
- Target device
- On-chip debug operation setting and emulator setting

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

The .bdf file of Renesas made board (for e.g., Fast Prototyping Board) 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".

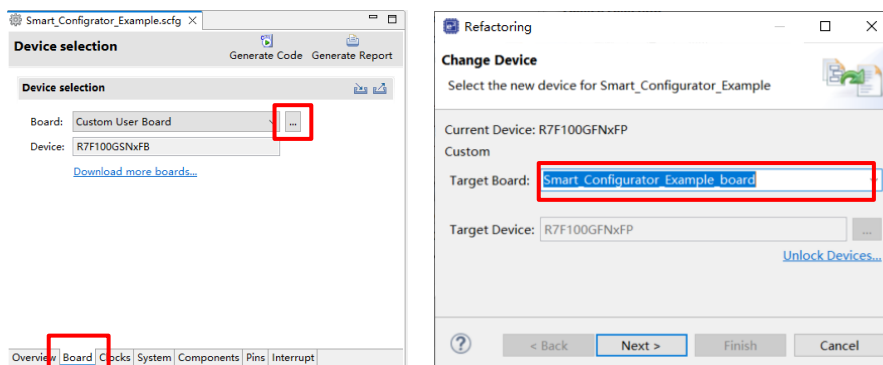


Figure 4-2 Selecting the Board

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

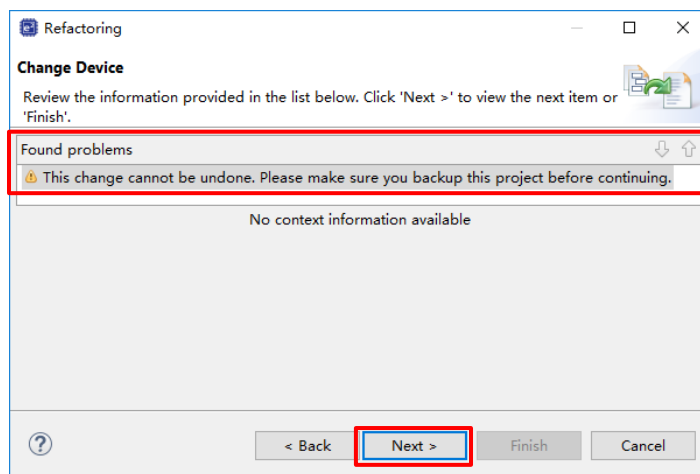


Figure 4-3 Found Problems

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”.

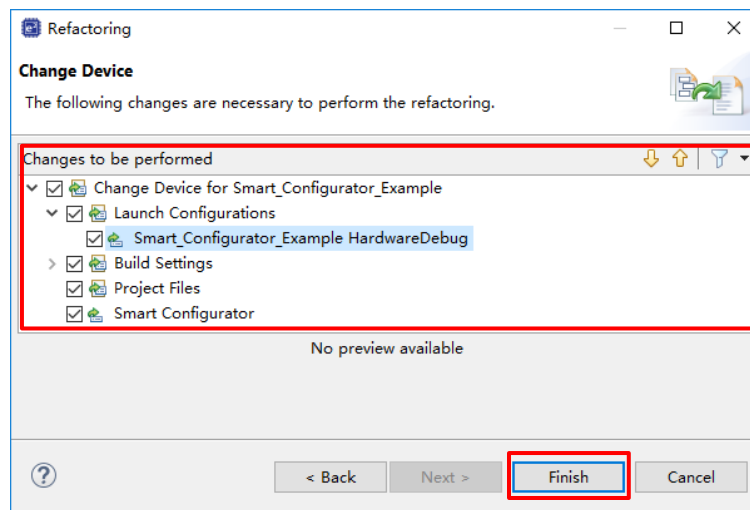



Figure 4-4 Changes to be Performed

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.

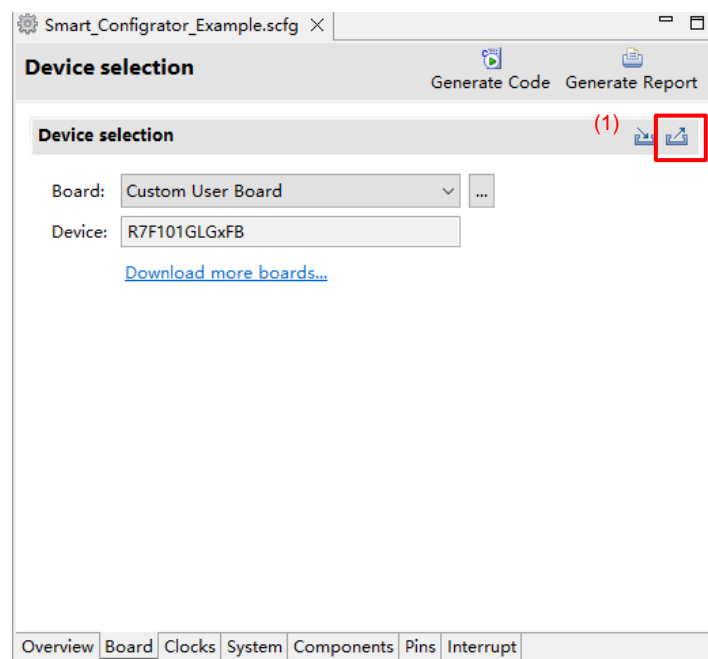



Figure 4-5 Exporting Board Settings (bdf Format)

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.

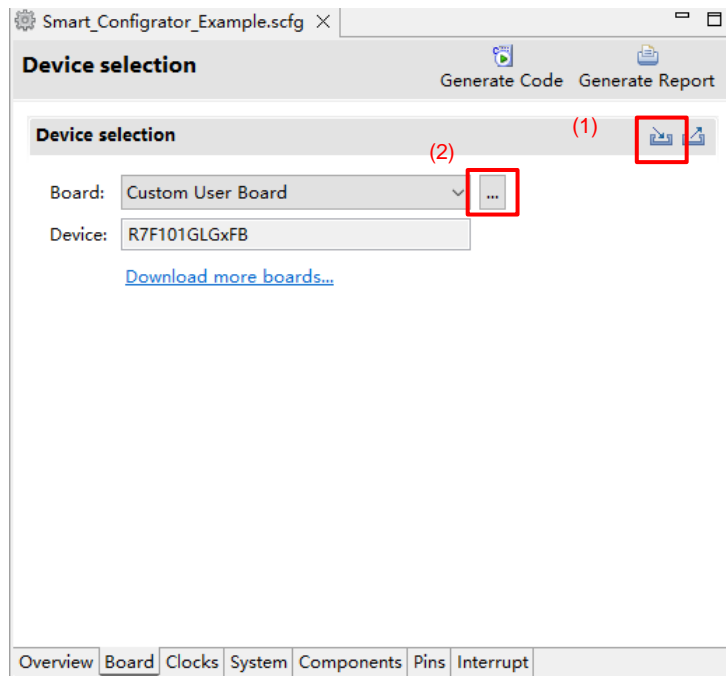


Figure 4-6 Importing Board Settings (bdf Format)

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 operation mode and EVDD setting.
- (2) Select the clocks required for device operations on the board (the high-speed on-chip oscillator is selected by default).
- (3) Specify the frequency of each clock in accordance with the board specifications (note that the frequency is fixed for some internal clocks).
- (4) For the multiplexer symbol, select the clock source for the output clocks.

The screenshot displays the 'Clocks configuration' window with the following settings and annotations:

- (1)** Operation mode: High-speed main mode 4.0(V)~5.5(V)
- (1)** EVDD setting: 4.0 V ≤ EVDD0 ≤ 5.5 V
- (2)** High-speed on-chip oscillator: (checked)
- (3)** Frequency: 32 (MHz)
- (4)** Multiplexer symbols for output clocks: fHHP, fMAIN, fCLK, fMXP, fIL, fSXP, fSXR.
- Other oscillators: Middle-speed on-chip oscillator (4 MHz), X1 oscillator (5 MHz), Low-speed on-chip oscillator (32.768 kHz), XT1 oscillator (32.768 kHz).

Figure 4-7 [Clocks] Page

4.3 System Settings

The Smart Configurator can change the link option setting according to [System] page when generate code. User can find and check the settings by right clicking project properties menu: Right click project in [Project Explorer] → [Properties] → [C/C++ Build] → [Settings] → [Linker] → [Device].

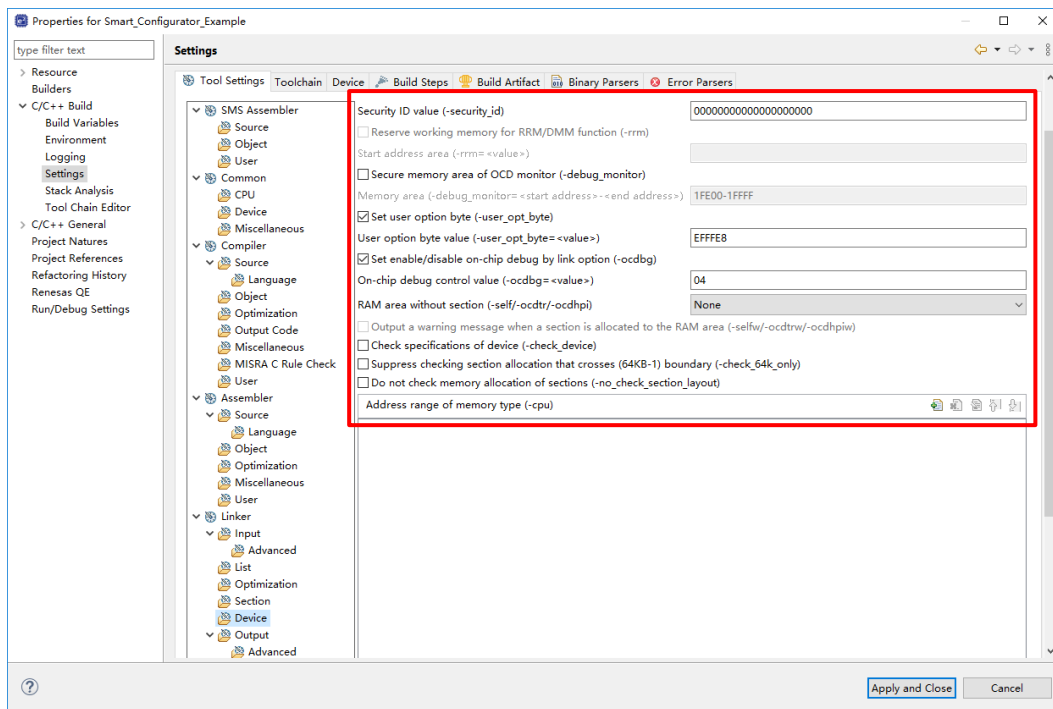


Figure 4-8 e2 studio Default Link Options View

After user clicks on [System] page of Smart Configurator, make desired setting as in below figure for illustration:

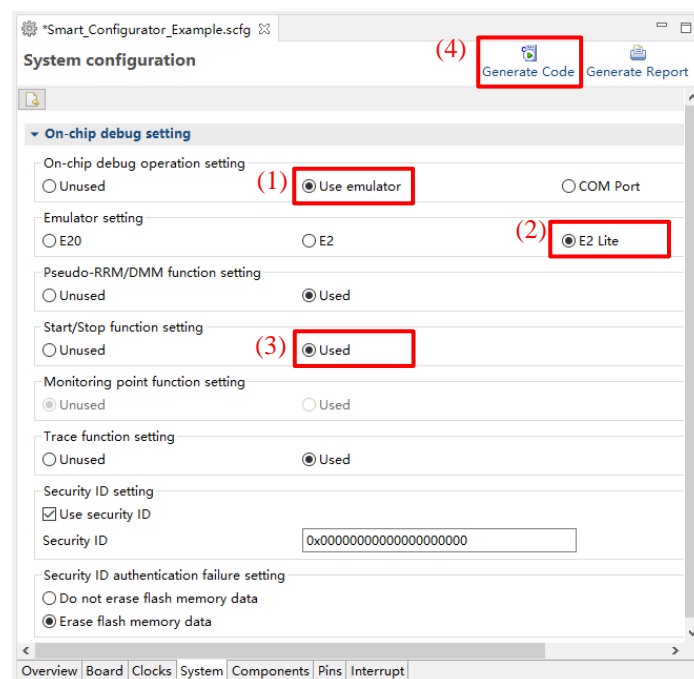


Figure 4-9 Smart Configurator [System] Page Setting

For example, please follow steps from (1) to (3) to make setting on [System] page, after that click on [Generate Code] button as in step (4), a dialog window will be prompted out as in below figure, to confirm with you for the linker option update in e² studio:

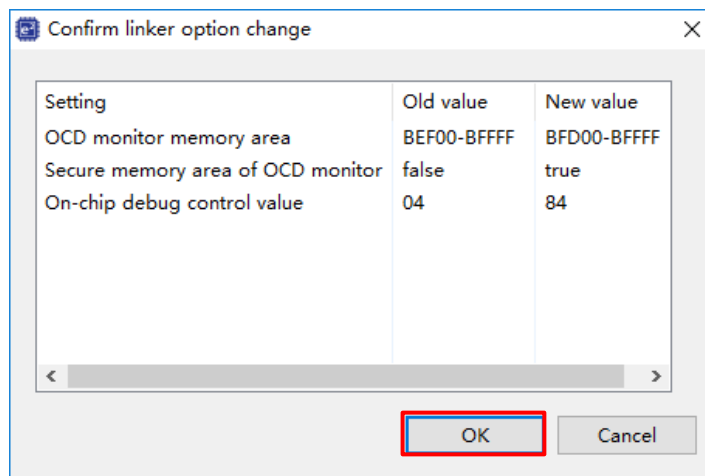


Figure 4-10 Confirm Linker Option Dialog

Please click [OK] button in the dialog, go back [Properties] window to check linker options updated as below:

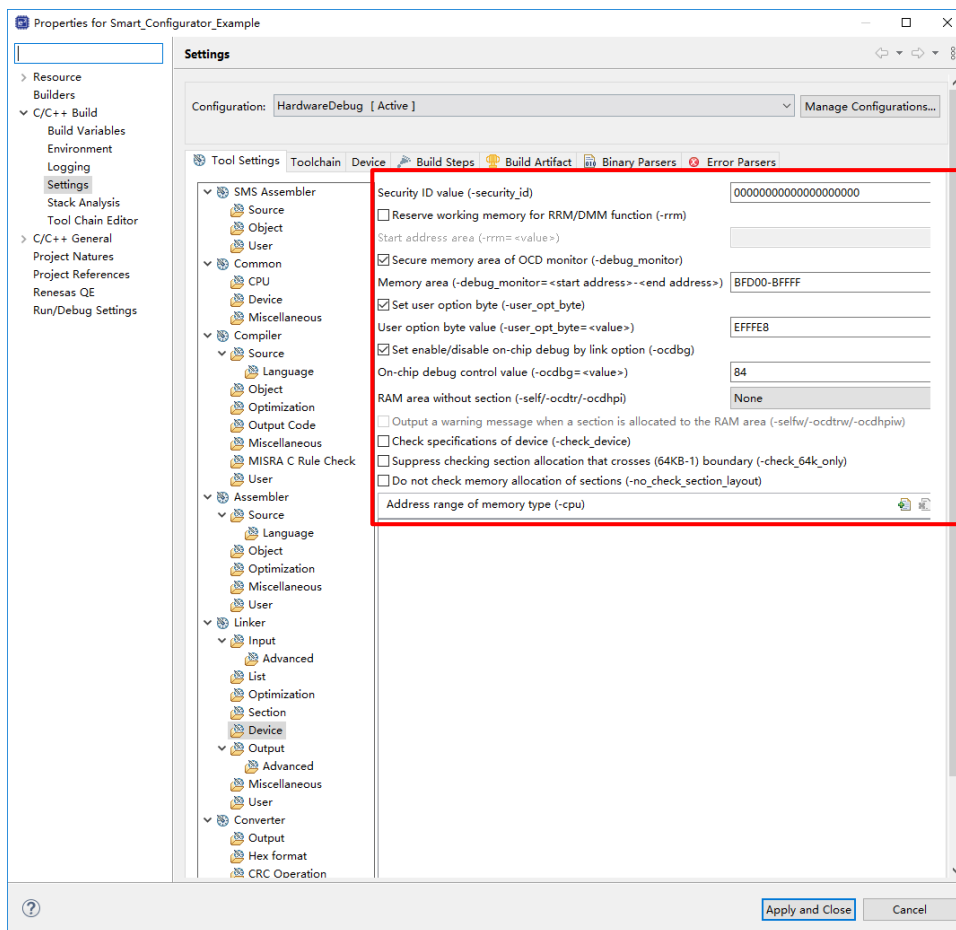


Figure 4-11 e² studio Updated Link Options View

Note: Depending on the MCU type selection or chip part numbers, these setting values varies. Please refer to the latest device User's Manual Hardware for the detail setting configuration.

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.

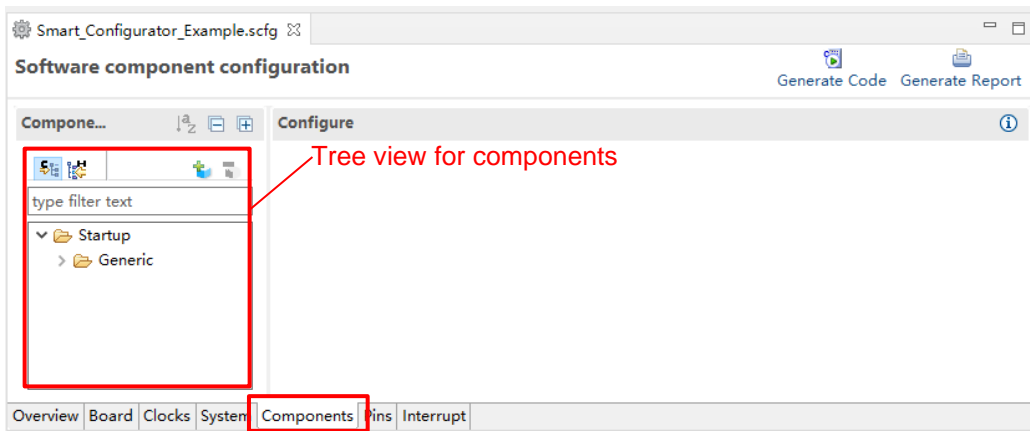




Figure 4-12 [Components] Page

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 [ (Component View)] icon. The tree view will display the components by component category.
- (2) Click on the [ (Hardware View)] icon. The tree view will display the components in a hardware resource hierarchy.

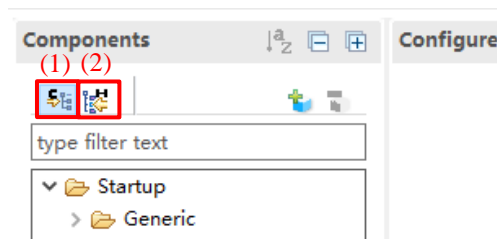





Figure 4-13 Switching to the Hardware View

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.

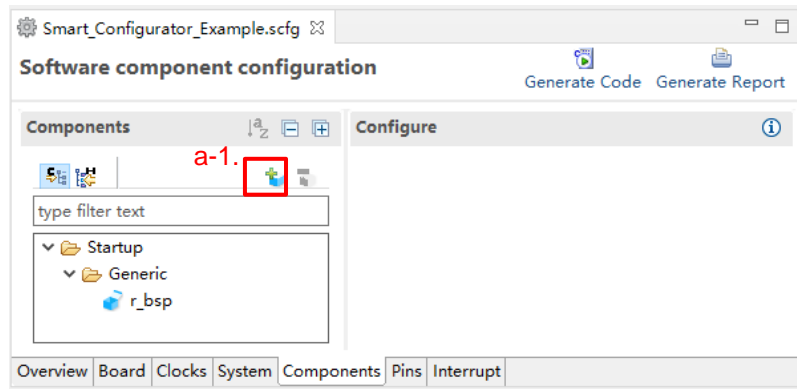


Figure 4-14 Adding a Component

- 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. Check that [Type] for the selected component is [Code Generator].
- a-4. Click on [Next].

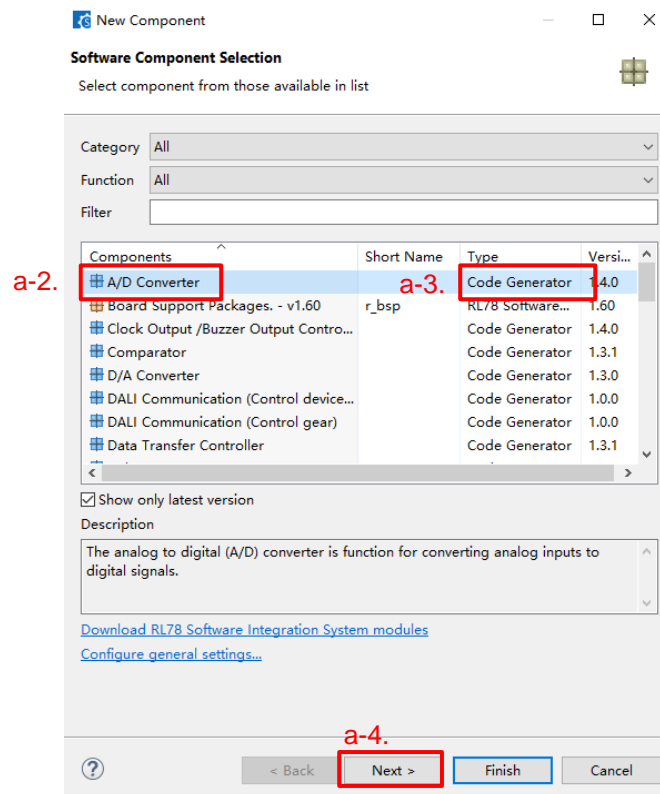


Figure 4-15 Adding a Code Generator Component

- a-5. Specify an appropriate configuration name in the [Add new configuration for selected component] page of the [New Component] dialog box or use the default name (for e.g., Config_ADC).
- a-6. Select a hardware resource or use the default resource (for e.g., ADC).
- a-7. Click on [Finish].

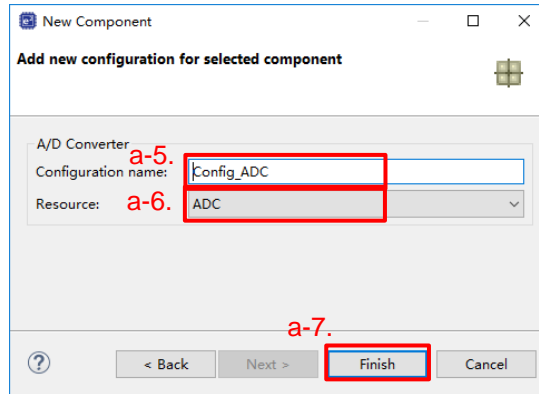




Figure 4-16 Adding a Component

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

- b-1. Click on the [ (Hardware View Menu)] icon. 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-3 to a-7.

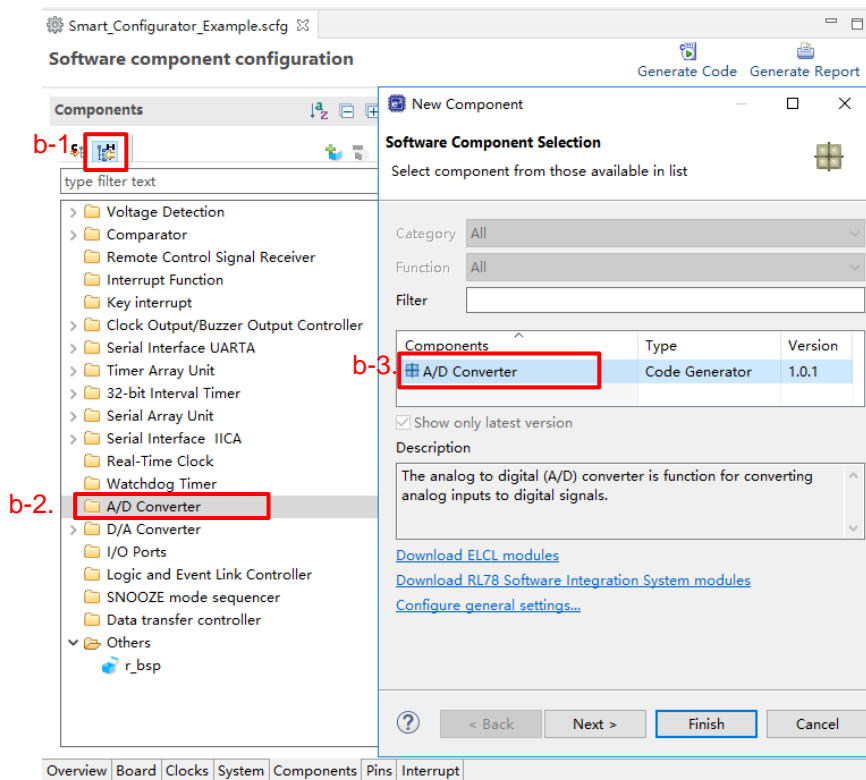



Figure 4-17 Adding a Code Generator Component to the Hardware View

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.

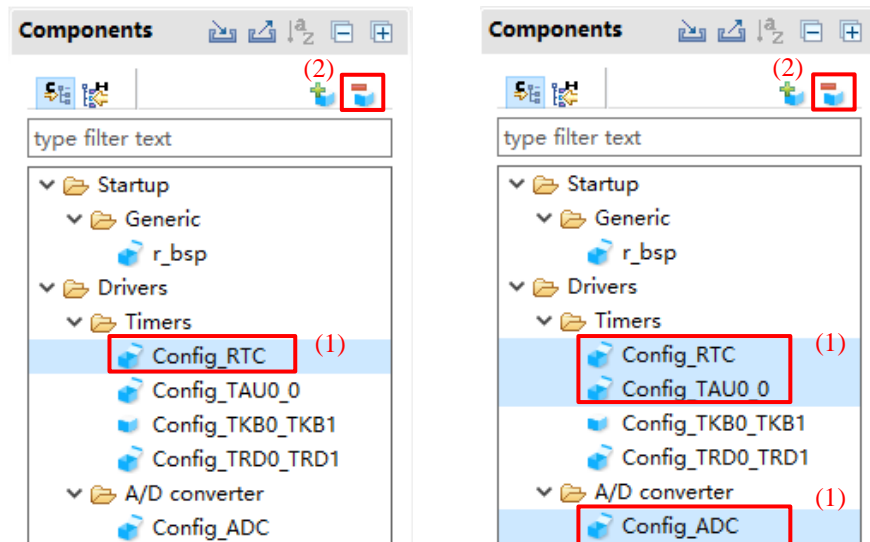



Figure 4-18 Removing a Software Component or Multiple Components

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 [10 bits] under [Resolution setting].
 - b. Select [Software trigger no wait mode] under [Trigger mode setting].
 - c. Select [ANI0] for [A/D channel selection].
 - d. Select [2112/fCLK] for [Conversion time].

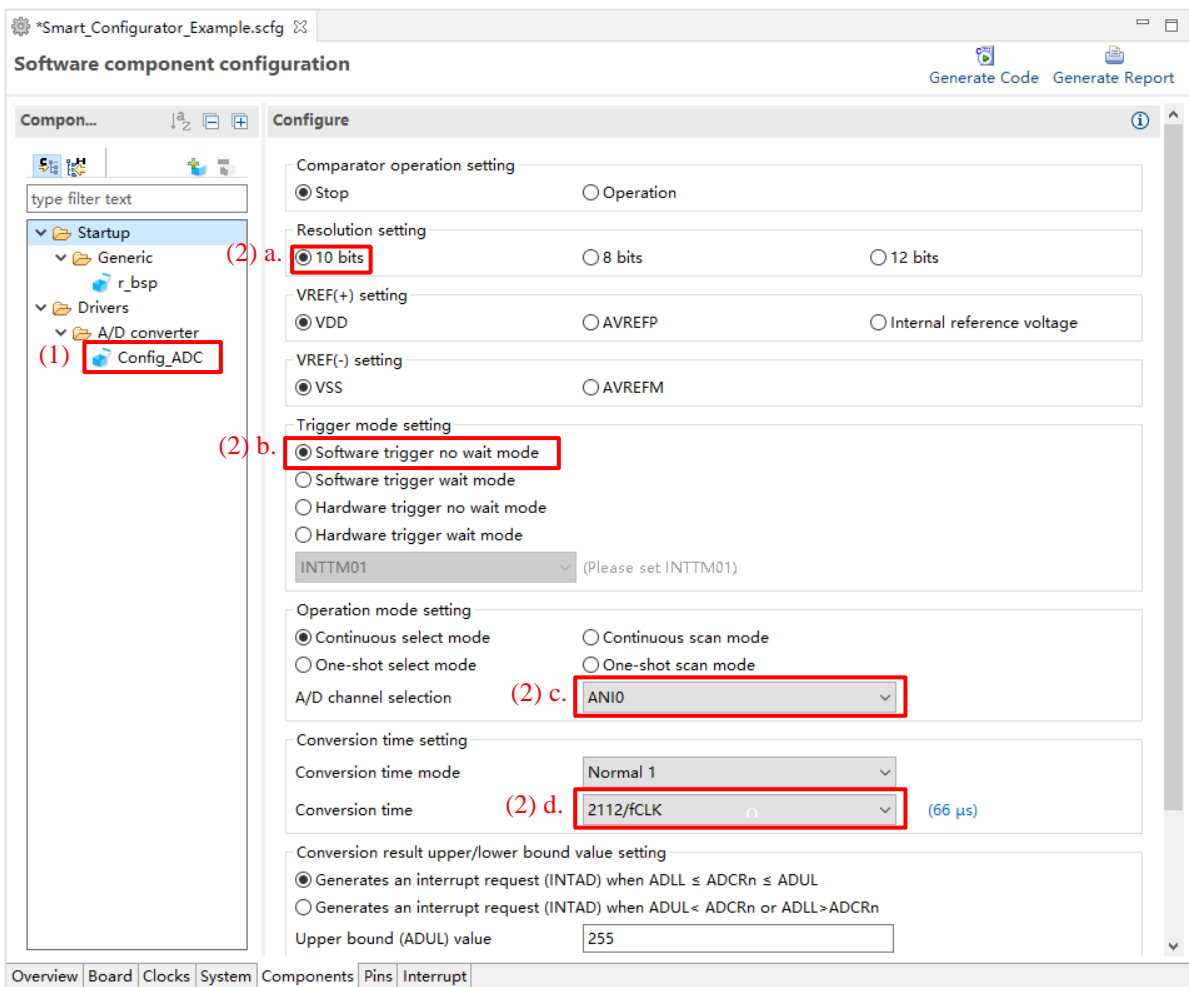


Figure 4-19 Setting of a Code Generator Driver

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 TAU0_1 to TAU0_3). 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_TAU0_1).
- (2) Select [Change resource] from the context menu.

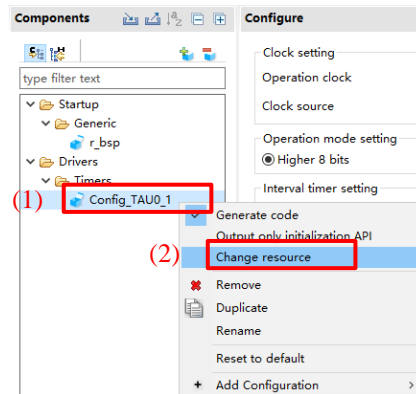


Figure 4-20 Changing the Resource

- (3) Select a new resource (for e.g., TAU0_3) in the [Resource Selection] dialog box.
- (4) The [Next] button will be active, click on it.

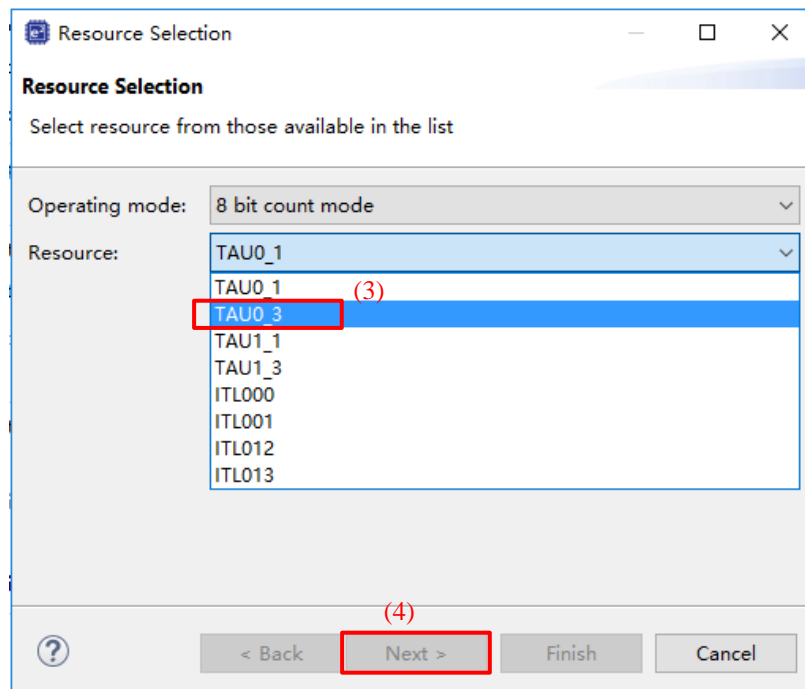


Figure 4-21 Components Page – Selecting a New Resource

- (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].

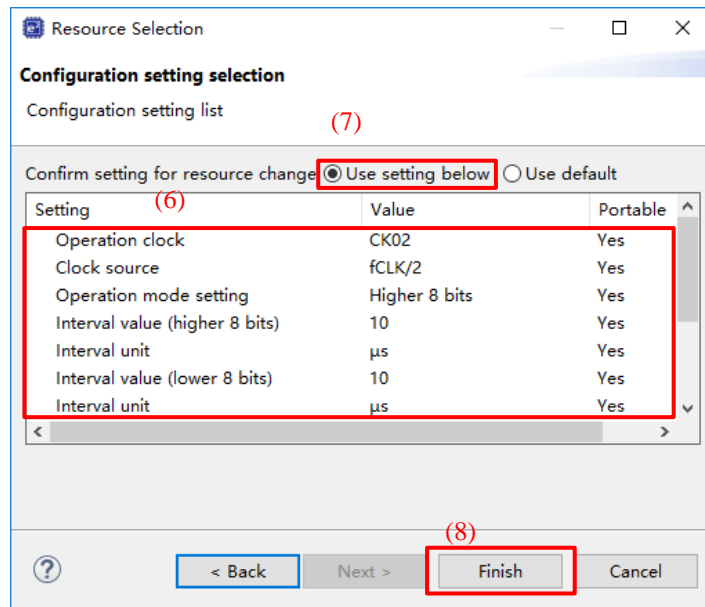


Figure 4-22 Checking the Settings of the New Resource

The resource is automatically changed (for e.g., changed from INTTM01 to INTTM03).

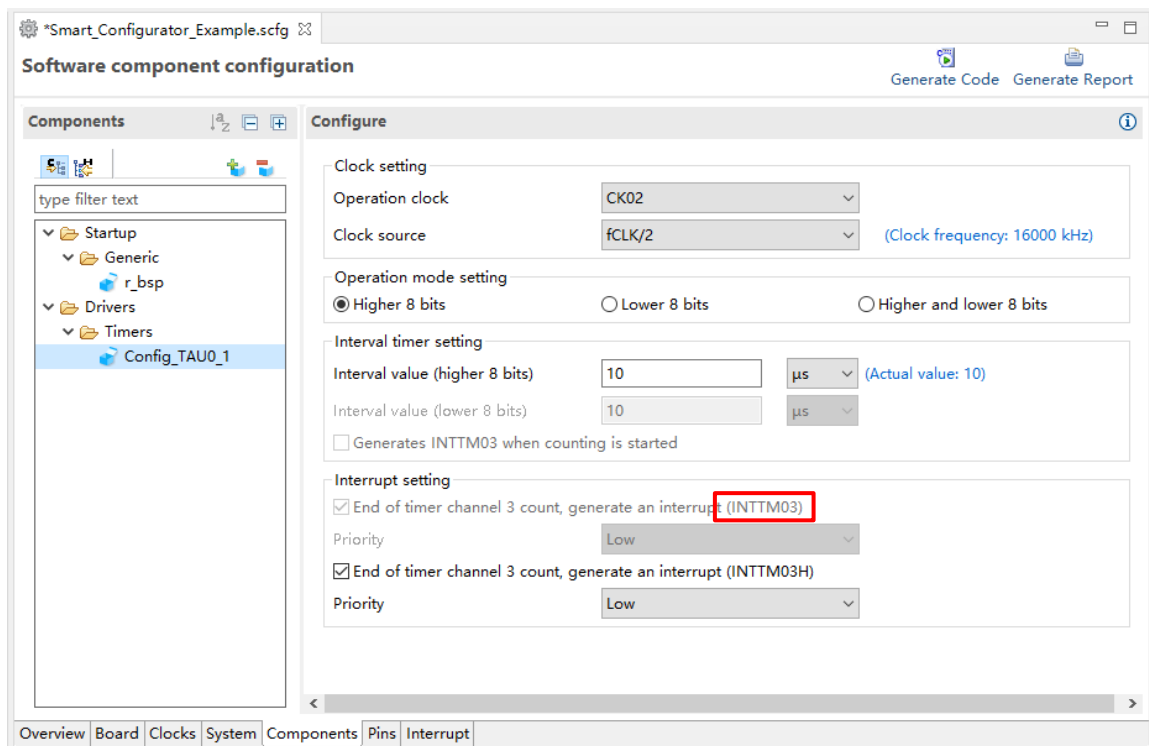


Figure 4-23 Resource Changed Automatically

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_TAU0_1 to Config_TAU0_3).

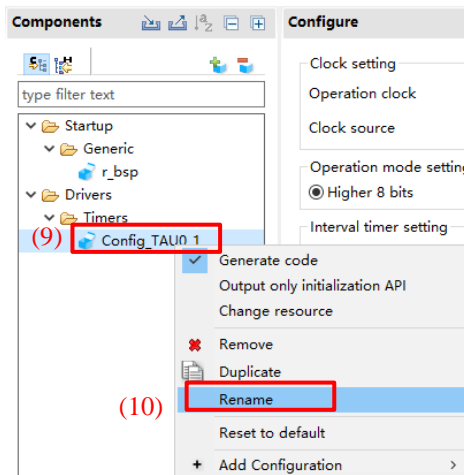


Figure 4-24 Renaming the Configuration

4.4.6 Setting SNOOZE Mode Sequencer (SMS) Component

SNOOZE Mode Sequencer (SMS) component is a new component type as “Graphical Configurator”, it is list and can be selected to use directly in default component list.

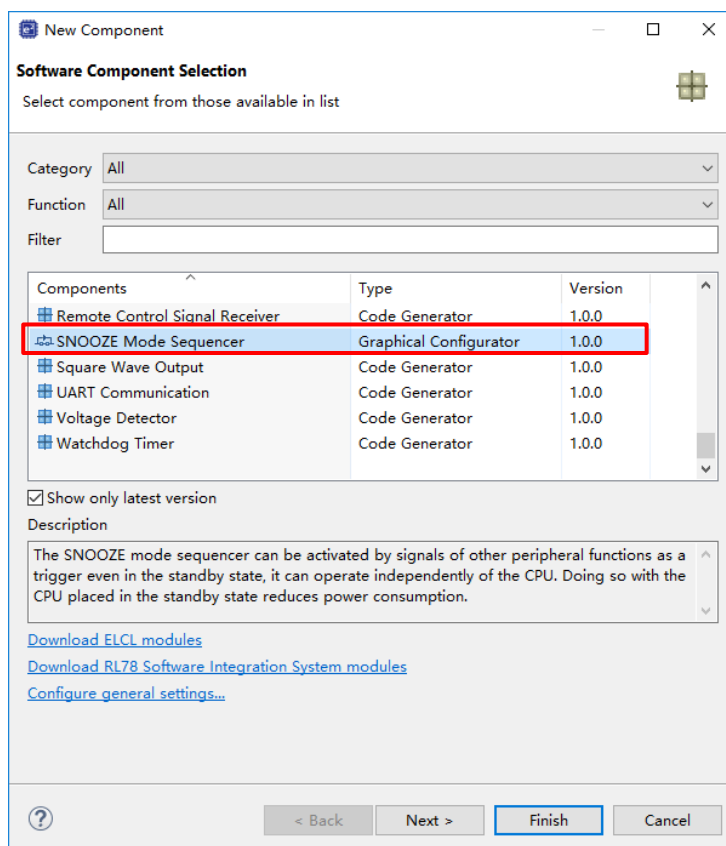


Figure 4-25 Add SNOOZE Mode Sequencer

A GUI look and feel of Graphical Configurator is displayed in below SMS figure; it is more graphically compared with Code Generator. User can drag, drop and configure the block which user wants to use.

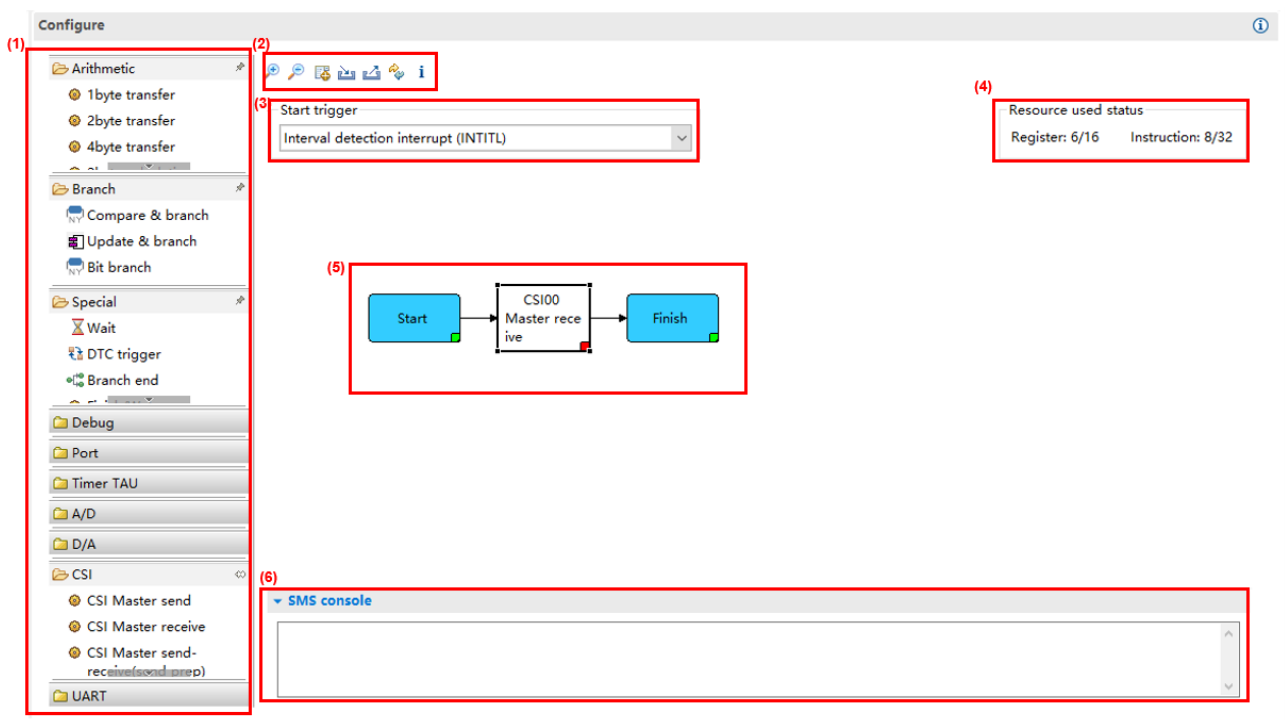



Figure 4-26 SMS Graphical Configurator GUI Overview

Table 4-2 SMS GUI area description

Area	Description
(1) Block elements	View the available blocks for SMS. A block is a part for forming a sequence (function), and includes A/D voltage acquisition, comparison & branching and 1-byte transfer.
(2) Toolbar	Zoom in.
	Zoom out.
	Display the SMS data management dialog and manage the variables to be used.
	Import the SMS sequence. User can use some sample sequences by clicking this icon.
	Export the SMS sequence.
	Update the SMS data file.
	Displays the information of the SMS data file.
(3) Start trigger selection	Select a startup trigger.
(4) Resource status	It shows registers and the number of instructions used.
(5) Canvas area	Place the SMS block and create the sequence.
(6) SMS console	Displays message for unavailable configurations.

Follow the procedure below to set up SMS block:

- (1) Select a block from Block elements list (for e.g., CSI Master receive).
- (2) Drag “CSI Master receive” block to SMS canvas between Start block and Finish block where the drop location doesn't show the indicator of .
- (3) User can configure the block by double click to pop the “CSI Master receive setting” property setting dialog.
- (4) User can specify the setting in the “CSI Master receive setting” property dialog.
- (5) Open “Data Management” setting, user can edit the receive data.
- (6) When user correctly configure the color of bottom right corner will change from red to green.
- (7) User can add some blocks, drag and drop to adjust the sequence.

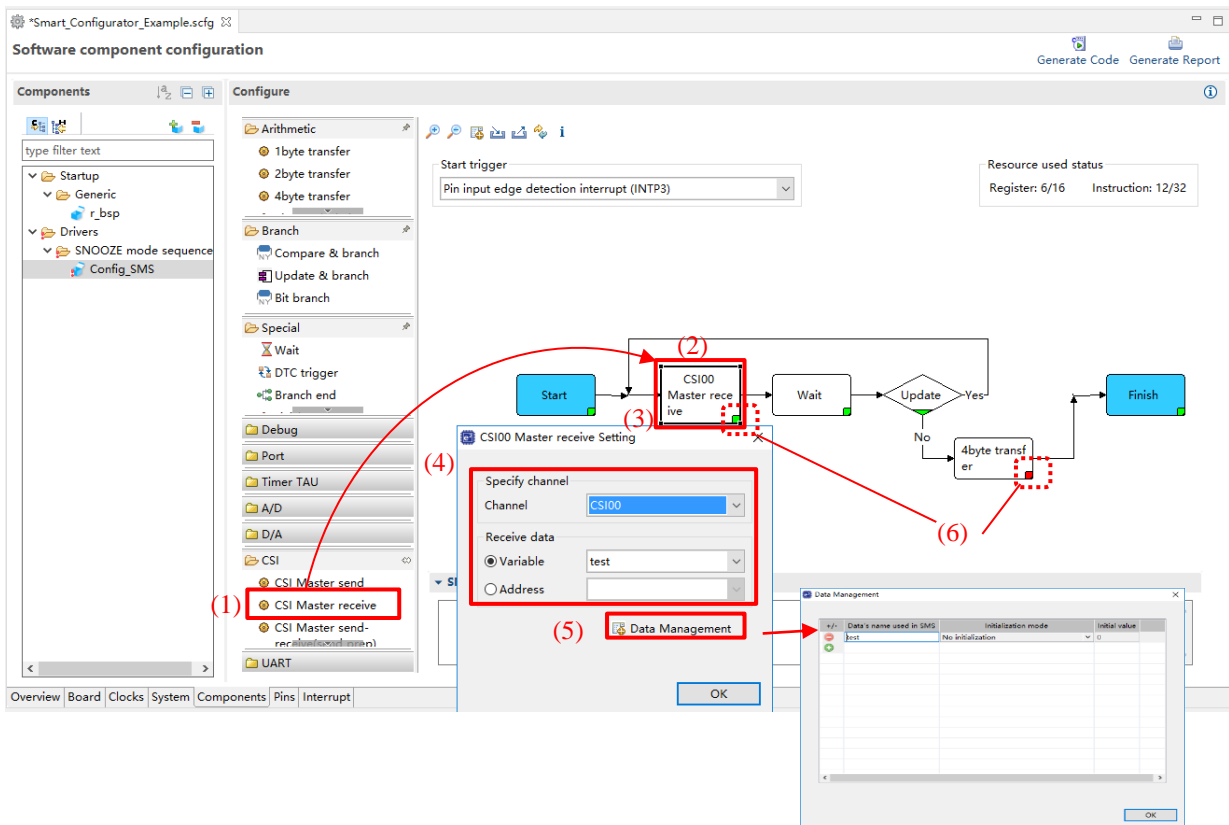



Figure 4-27 SMS Block Configure

Note:

To build SMS (Snooze Mode Sequencer) module successfully, user need to download and install the latest

SMSASM_Vxxx_setup.exe tool from Renesas website. Check the status of the integrated toolchains by clicking  button in toolbars.

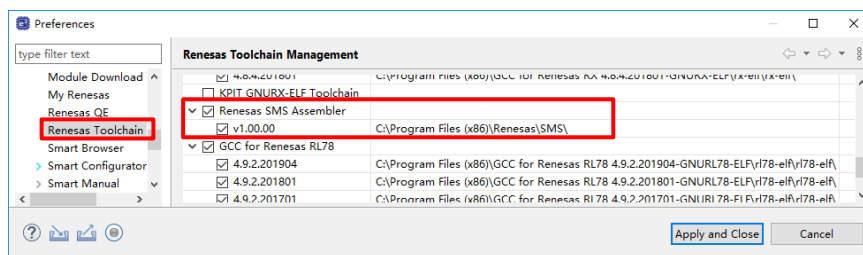


Figure 4-28 SMS Assembler

4.4.7 Update SMS Data Files

Follow the procedure below to update SMS data file (Block, Sequence) to the latest version. User can use new blocks and sequences by updating.

- (1) Click on SMS GUI button [Update SMS data files] to check if SMS data file have the newer version and download automatically from the web.
- (2) Waiting for the operation finished.
- (3) Finished the latest version update.

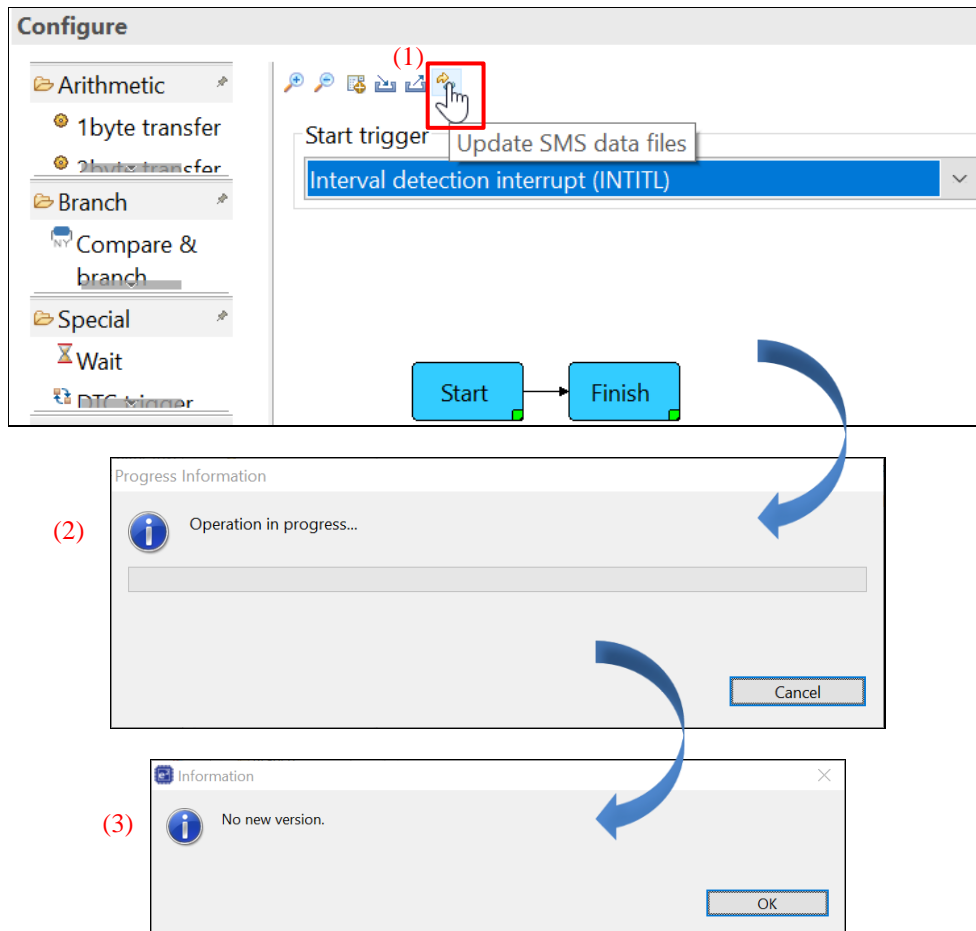


Figure 4-29 SMS Data File Download

4.4.8 Logic Event Link Controller (ELCL) Modules Download

The Software Component type for Logic Event Link Controller (ELCL) is Graphical Configurator. ELCL modules can be added from component list in New Component dialog. If you want to use other ELCL modules not included in Component list, you can click on [Download ELCL modules] link in New Component dialog to check and download more ELCL modules:

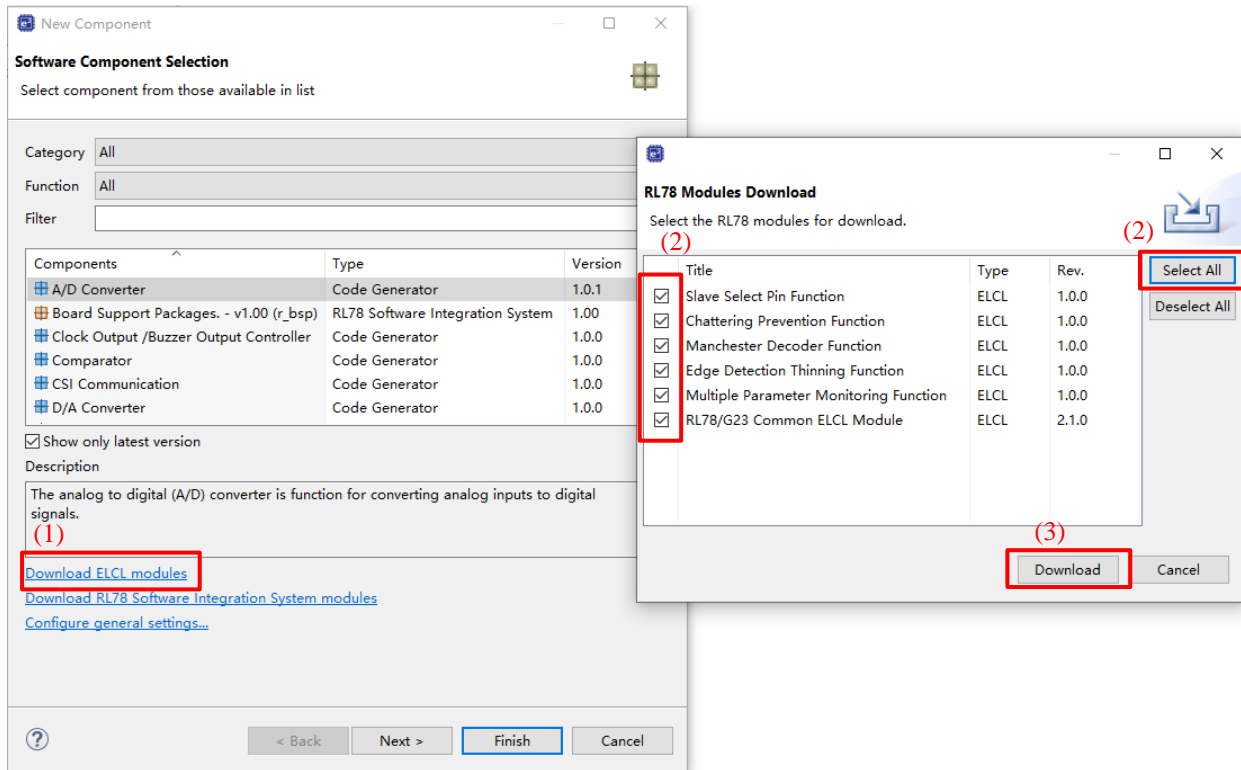


Figure 4-30 New ELCL Modules Download

After download, all ELCL modules are auto added to component list:

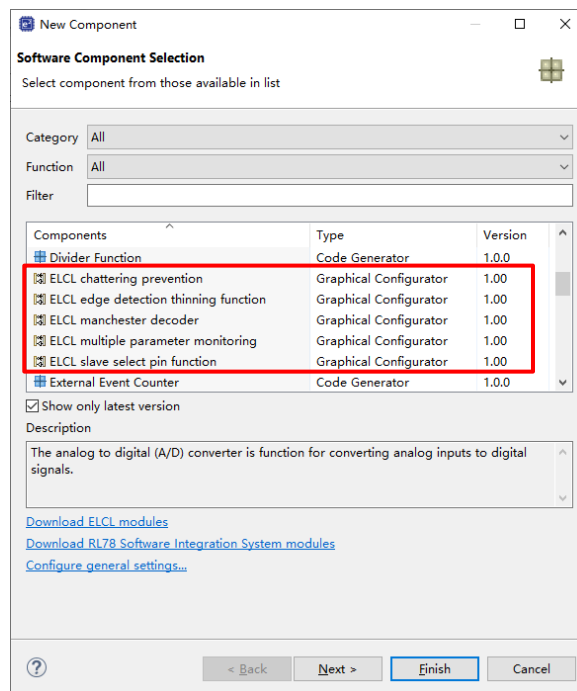


Figure 4-31 Add ELCL Module

4.4.9 Setting an ELCL Component

Follow the procedure below to set up an ELCL module.

- (1) Select an ELCL module from Software Component Selection list (for e.g., ELCL slave select pin function).
- (2) Configure the driver in the [Configure] panel. The following steps and figure show an example.
 - a. Select the input signal under [Input signal selector] UI part.
 - b. Select the logic block under [Event controller (link processor)] UI part.
 - c. Select the output signal under [Output signal selector] UI part.
- (3) If user wants for more details about current ELCL module usage, user can click the [ELCL_slave_select_pition.pdf] link to open the application notes for check.

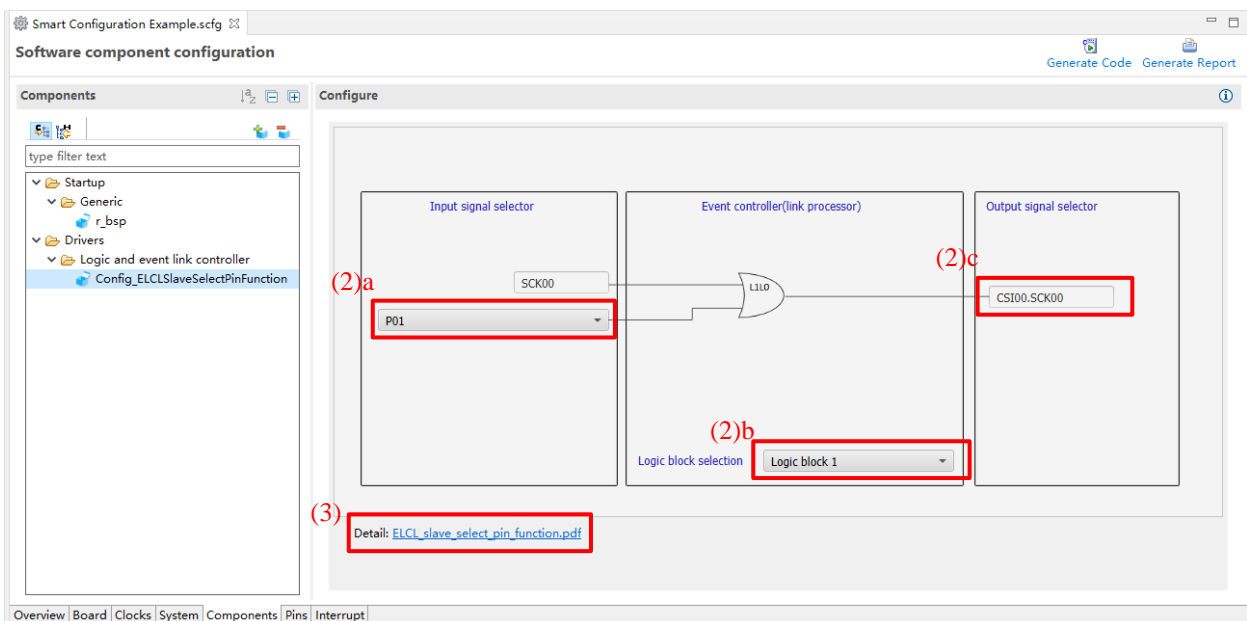



Figure 4-32 Configure an ELCL Module

4.4.10 Downloading RL78 Software Integration System Modules

RL78 Software Integration System modules is another software component type which can provide simple view for user to make driver/middle/application SW configuration and generate the code. The available RL78 Software Integration System modules can be downloaded from Renesas web.

- (1) Click on  (Add component)] as in Figure 4-14 to open New component dialog.
- (2) Click the [Download RL78 Software Integration System Modules] link in the [Software Component Selection] page of the [New Component] dialog box to download RL78 Software Integration System Modules.

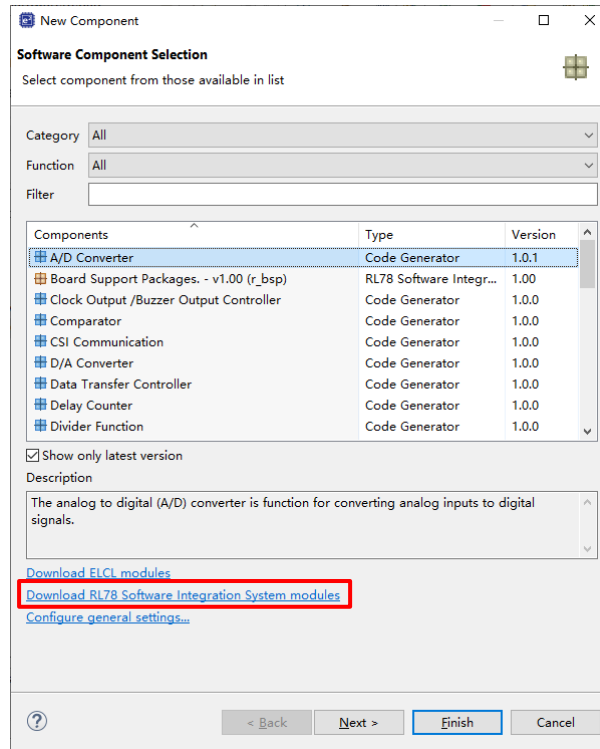


Figure 4-33 RL78 Software System Integration Download Entry

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

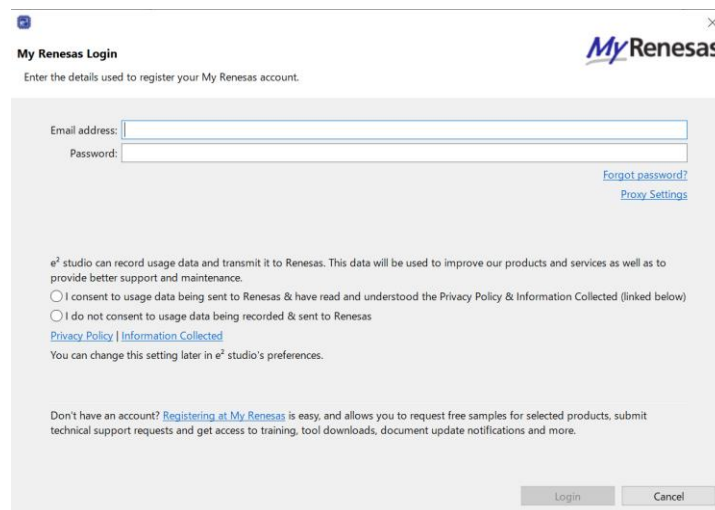


Figure 4-34 Login to My Renesas

- (3) Select the checkbox of the required module in the [RL78 Software Integration System Modules Download] dialog box.
- (4) Click on [Browse...] to select the location where the downloaded module is to be stored.
- (5) Click on [Download] to start downloading the selected RL78 Software Integration System Modules module.

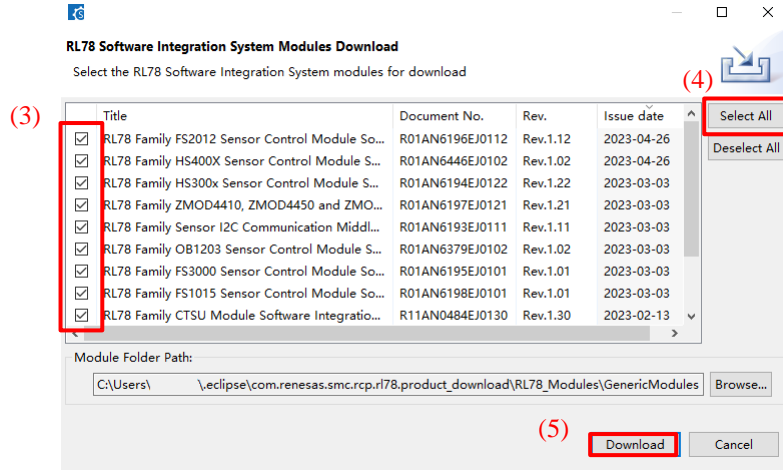



Figure 4-35 Downloading RL78 Software Integration System Modules

4.4.11 Adding a RL78 Software Integration System Module

The following describes the procedure for adding a RL78 Software Integration System Module.

- (1) Click on the  (Add component)] icon.
- (2) Select components which [Type] is [RL78 Software Integration System] from the list in the [Software Component Selection] page of the [New Component] dialog box. Two or more components can be selected by clicking with the Ctrl key pressed.
- (3) Click on [Finish].

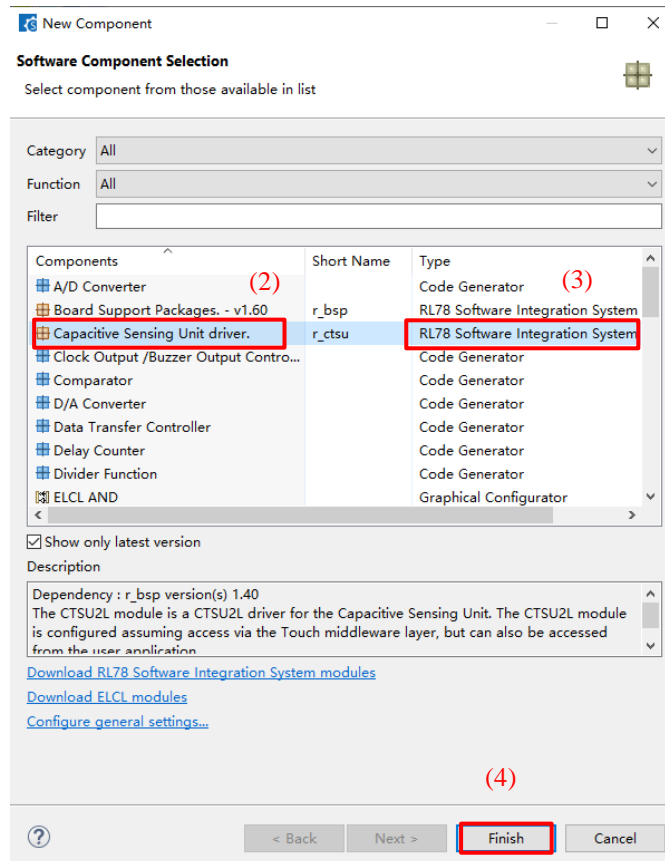


Figure 4-36 Adding RL78 Software Integration System Module

4.4.12 Setting a RL78 Software Integration System Module

To use RL78 Software Integration System module, set configuration option. Setting methods depends on components,

- ✓ Set configuration options on Configure panel and settings will be generated to configuration file of RL78 Software Integration System module automatically at each time of code generation action.

Note: The configuration file of RL78 Software Integration System module will be generated in the r_config folder.

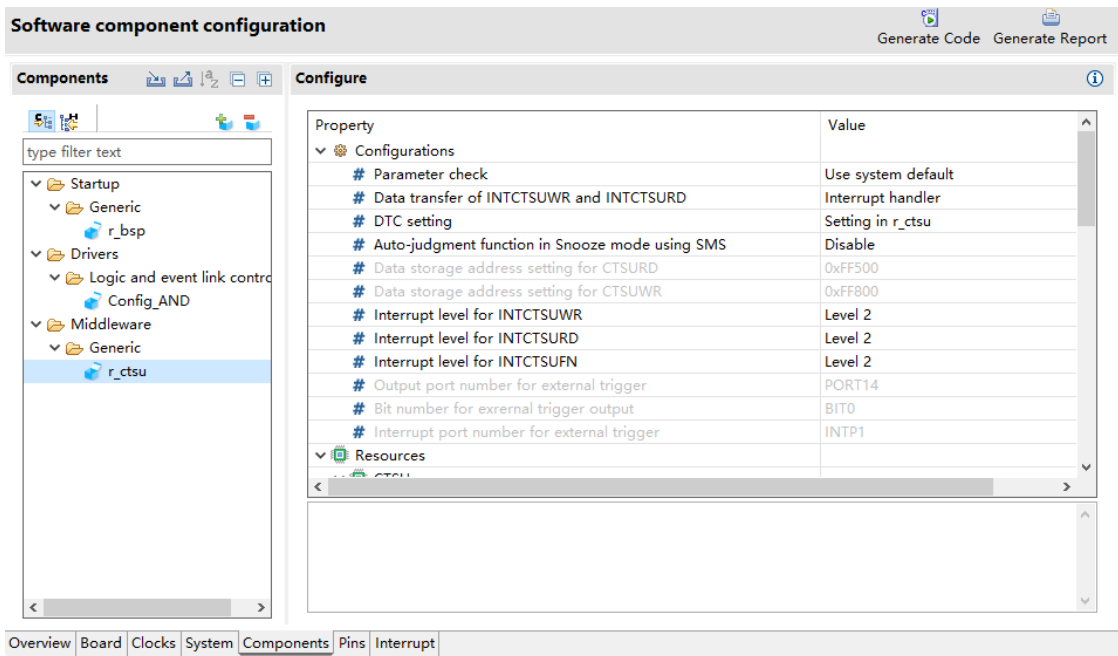


Figure 4-37 Setting RL78 Software Integration System Module

4.4.13 Changing Version of BSP Configuration

The following describes the procedure for version change of BSP configuration.

- (1) From the component tree, right-click the r_bsp component whose version user wants to change.

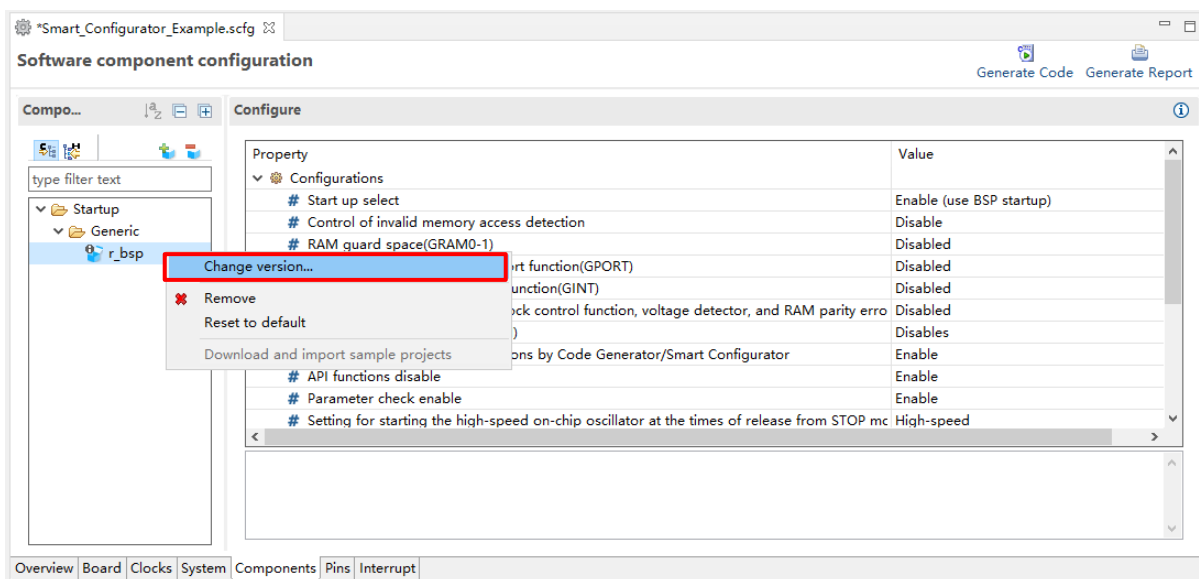


Figure 4-38 Version Change of BSP Configuration

- (2) Select [Change Version ...] from the context menu.
- (3) In the [Change Version] dialog box, select the version you want to change. If user selects a version that the device does not support, [Selected version doesn't support current device or toolchain] will be displayed, so select the corresponding version.
- (4) Click [Next].

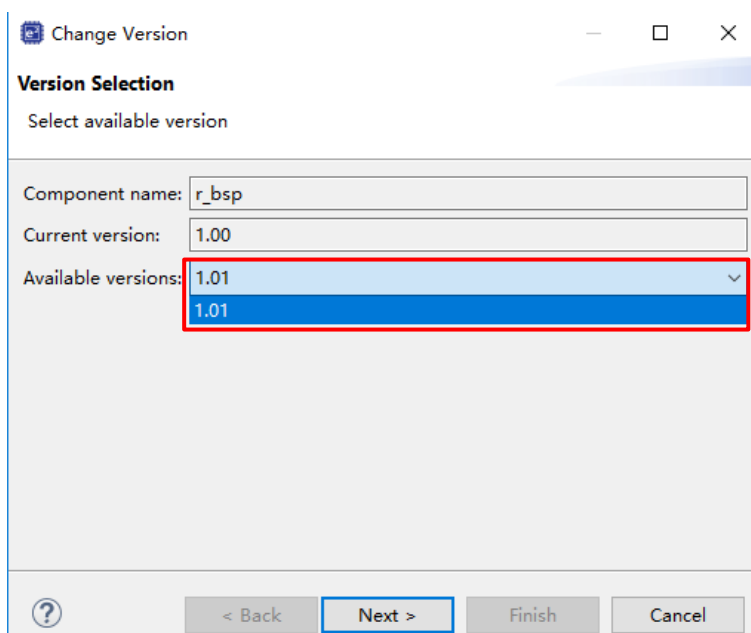


Figure 4-39 Select Version of BSP Component

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

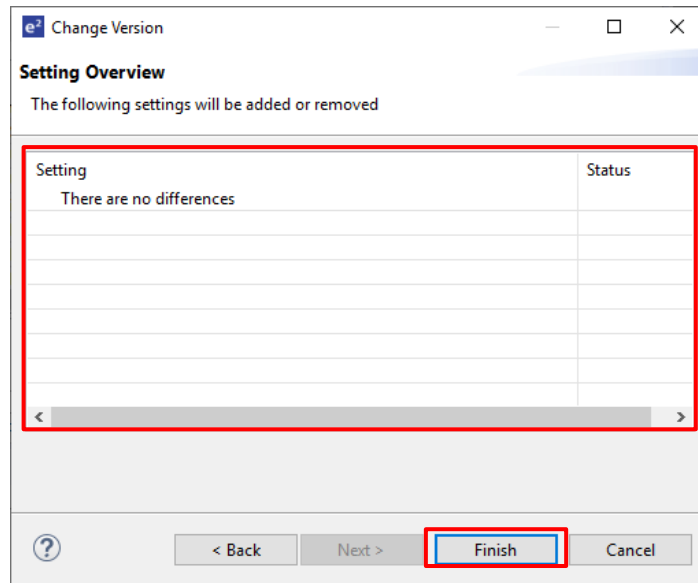


Figure 4-40 Confirm Setting Change Item

- (6) As [Confirm to change version and proceed to generate code] Is displayed, if user does not have any problem, click [Yes].

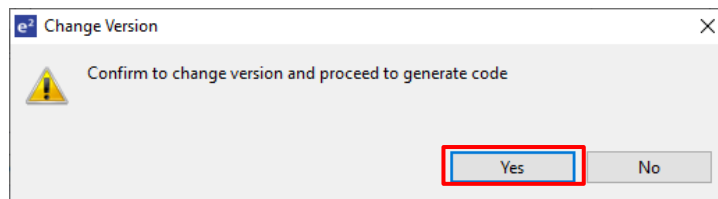


Figure 4-41 Confirm Version Change

- (7) The BSP component version is change and code generation is executed automatically.

4.4.14 **Export Component Configuration**

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

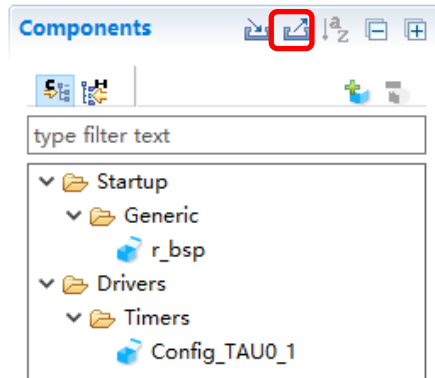


Figure 4-42 Export Configuration (xml format)

4.4.15 **Import Component Configuration**

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

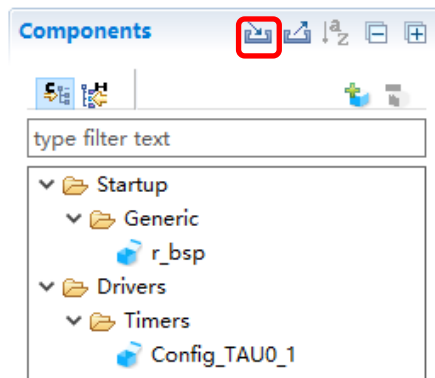


Figure 4-43 Import Configuration (xml format)

4.4.16 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 4-15), and display the [Preferences] dialog.

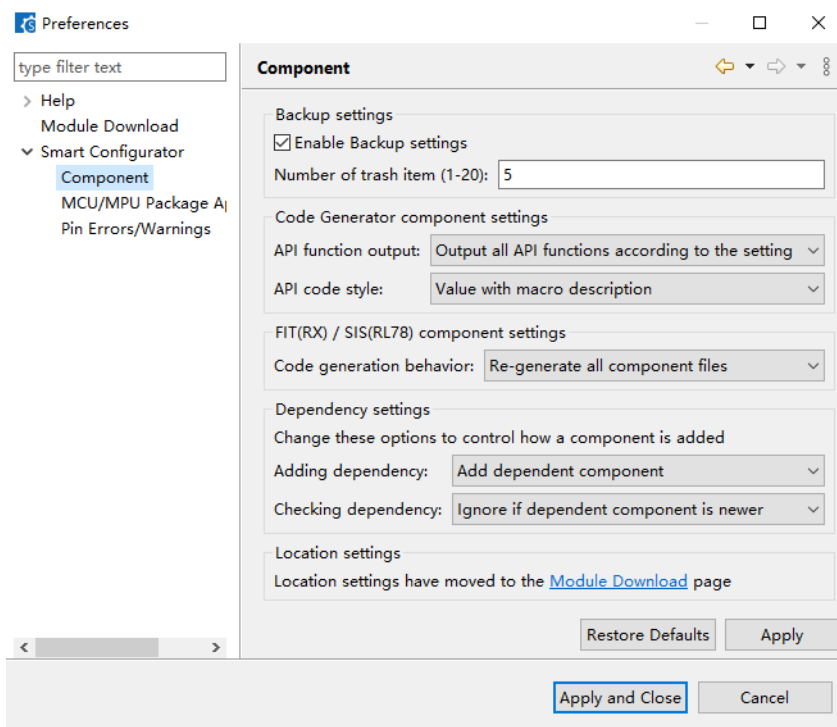


Figure 4-44 Configure General Setting of Component

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.

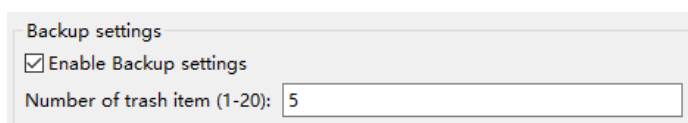


Figure 4-45 Trash number setting

2. The code generation behavior has two options: [Update configuration files] and [Re-generate all component files]. [Update configuration files] is the default selection. If "Update configuration files" is being selected and generate code, Smart Configurator will check whether the files are existing inside the user project. If the file exists, the file will not be overwritten. However, configuration files (e.g., xxx_config.h) will still be refreshed when code is generated. If "Re-generate all component files" being selected and generate code, Smart Configurator does not check the existence of the file and the file will always be overwritten.

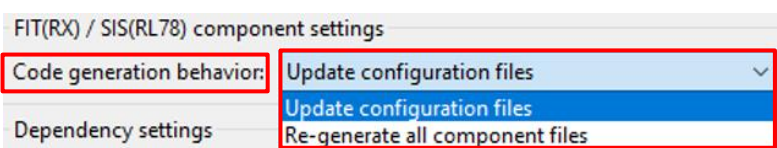


Figure 4-46 [Code generation behavior] Change

3. 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.

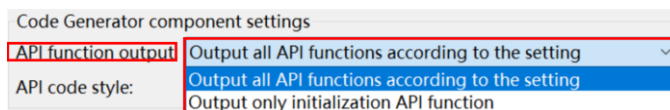


Figure 4-47 [RL78 API function output] Change

From e² studio 2022-10, output only initialization API feature can be applied for individual configuration (Code Generator component). Please right-click the selected component and select the "Output only initialization API" from the context menu.

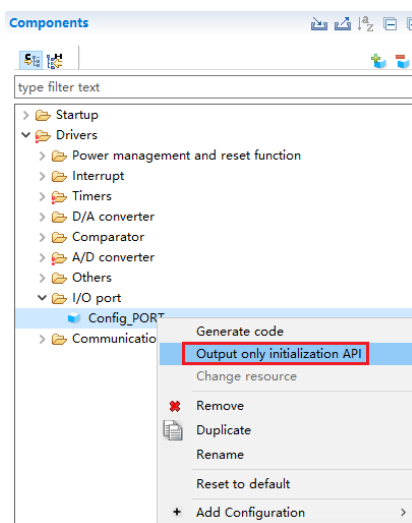


Figure 4-48 Context Menu “Output only initialization API” for Each Configuration

4. To generate code with HEX value, please change to [Value without macro description (raw HEX)] option in below figure. If user change back to default option setting: [Value with macro description], then all API with macro description will be generated again.

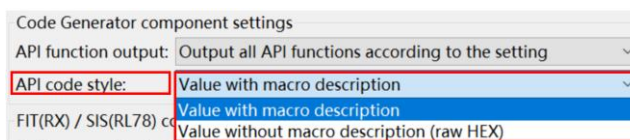


Figure 4-49 [API code style] Change

5. If the version of the module and its dependency do not match, a warning message W04020011 is displayed. If user checks the revision history of the module and its dependencies and you do not need to change the module you are using, you can ignore this warning. To clear this warning, select [Do not check for dependent component] in the [Checking dependency] list box in component preferences, then click [OK].

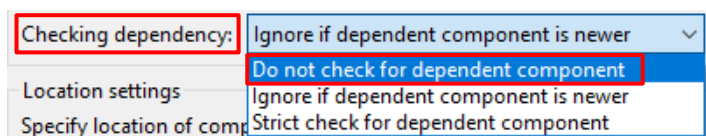


Figure 4-50 [Checking dependency] Change

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.

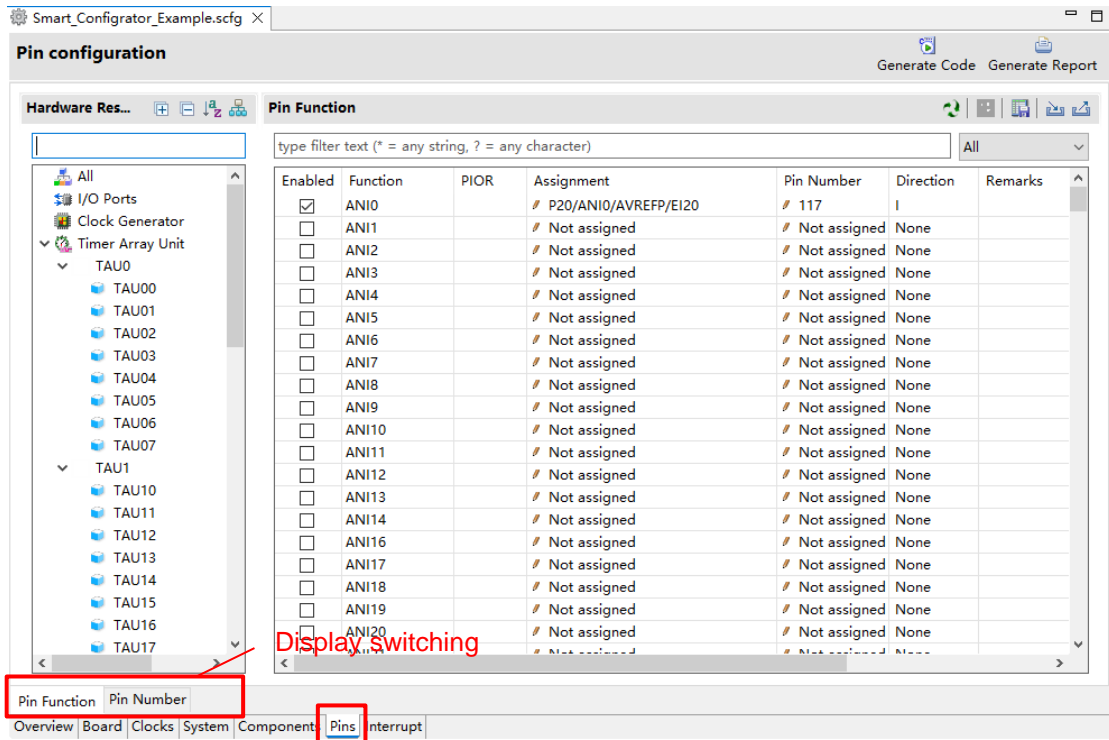


Figure 4-51 [Pins] Page ([Pin Function])

When user selects 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.

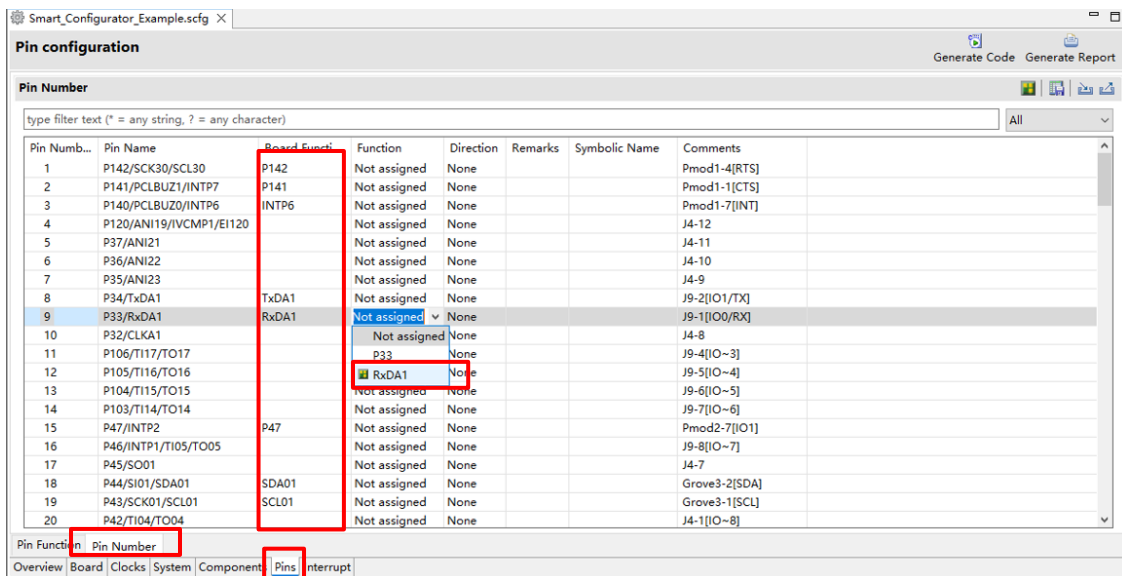


Figure 4-52 [Pins] Page ([Pin Number])

4.5.1 Changing the Pin Assignment by PIOR Function

PIOR “Filter Function” is a powerful feature to help user manage pin function settings, re-configure pin function settings or check pin function conflicts.

Follow the procedure below to change the assignment by PIOR function.

- (1) Type “pior1” in the tool text input box, all pin functions which related to PIOR1 will be listed out.
- (2) One of pin assignments is changed, all pin function assignments which related to PIOR1 will be re-assigned automatically.
- (3) The pin error messages may display in [Remark] column and [Configuration Problems view].
- (4) Please re-configure pin assignment according to the error message.

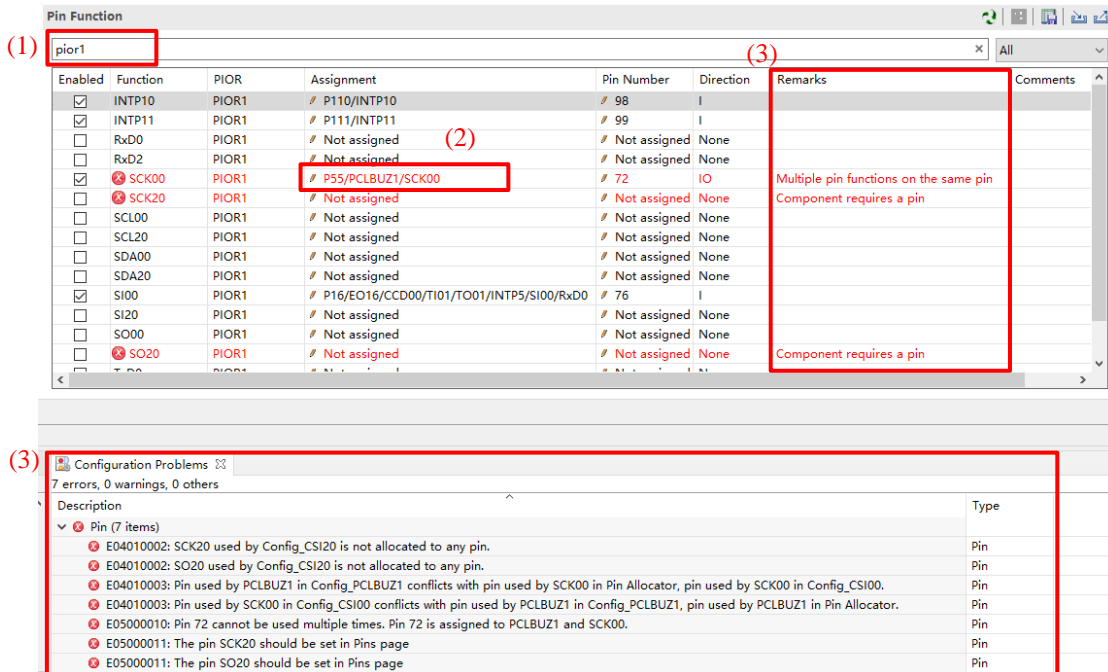


Figure 4-53 PIOR Filter Function

The PIOR setting can be reflected into generated code.

The PIOR code is generated in r_bsp file: <ProjectDir>\src\smc_gen\r_bsp\r_config\r_bsp_config.h file. If user wants to change the PIOR setting code value, change the assignment of related pin and generate code again.

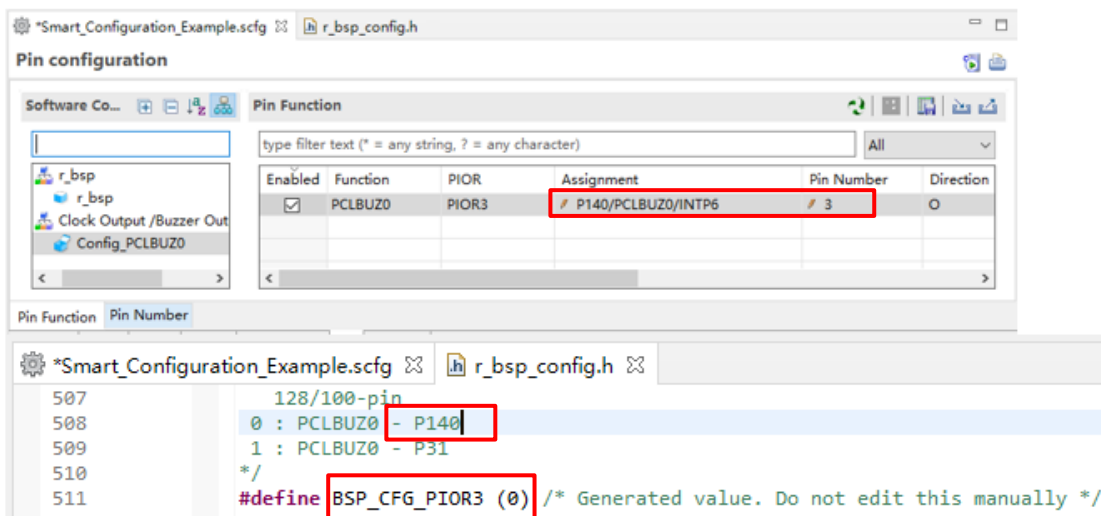




Figure 4-54 PIOR Code Generation

4.5.2 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 P12 to P16).
- (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.

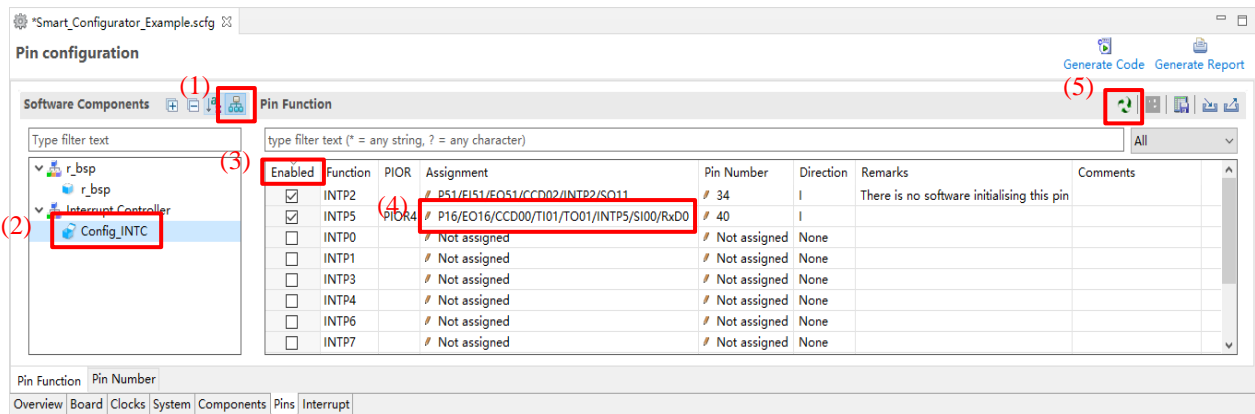


Figure 4-55 Pin Settings – Assigning Pins on the [Pin Function] List

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.

4.5.3 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...].

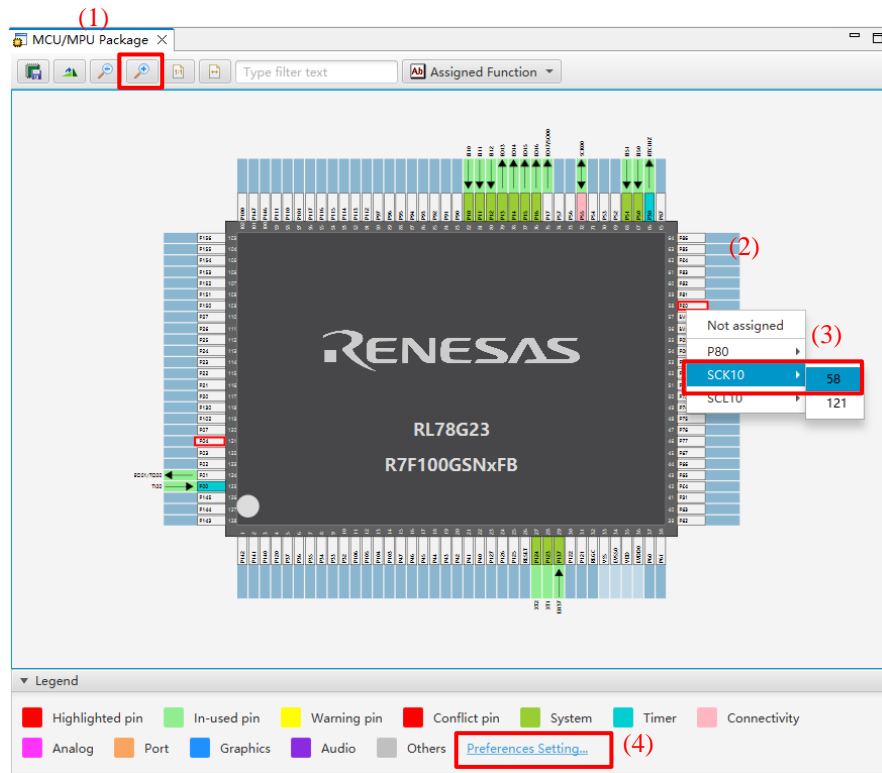


Figure 4-56 Assigning Pins Using the MCU/MPU Package View

4.5.4 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.

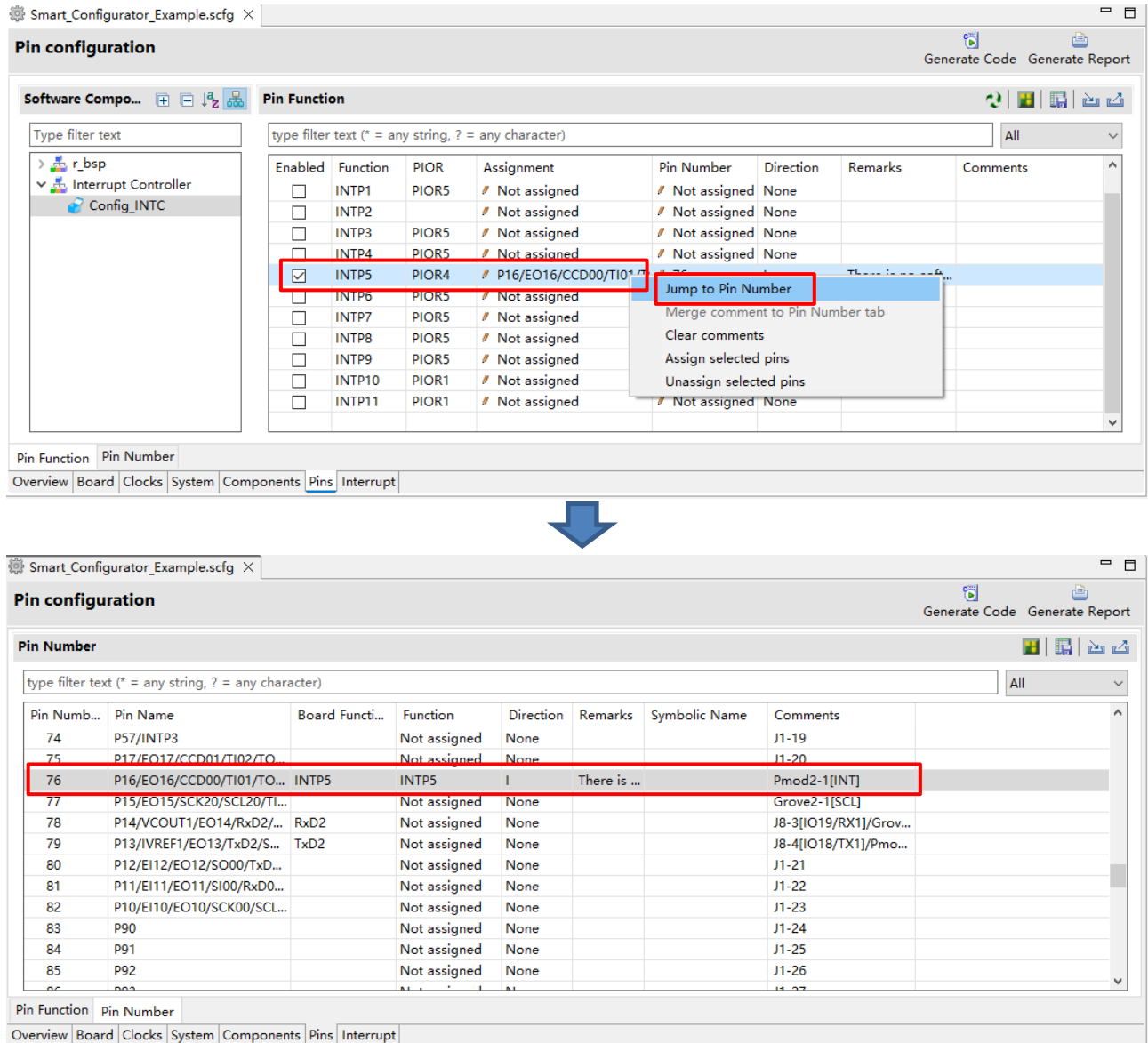



Figure 4-57 Jump to Pin Number

4.5.5 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.

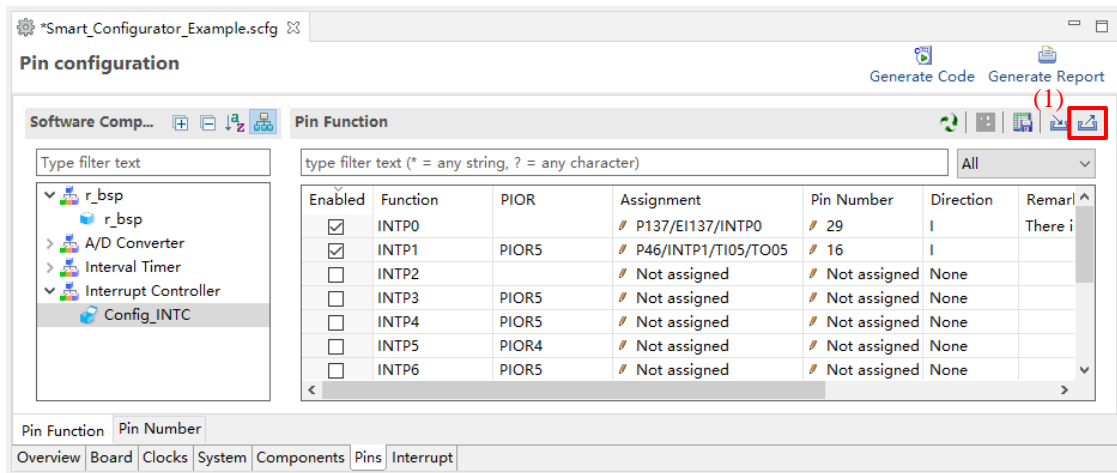




Figure 4-58 Exporting Pin Settings to an XML File

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.

4.5.6 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.

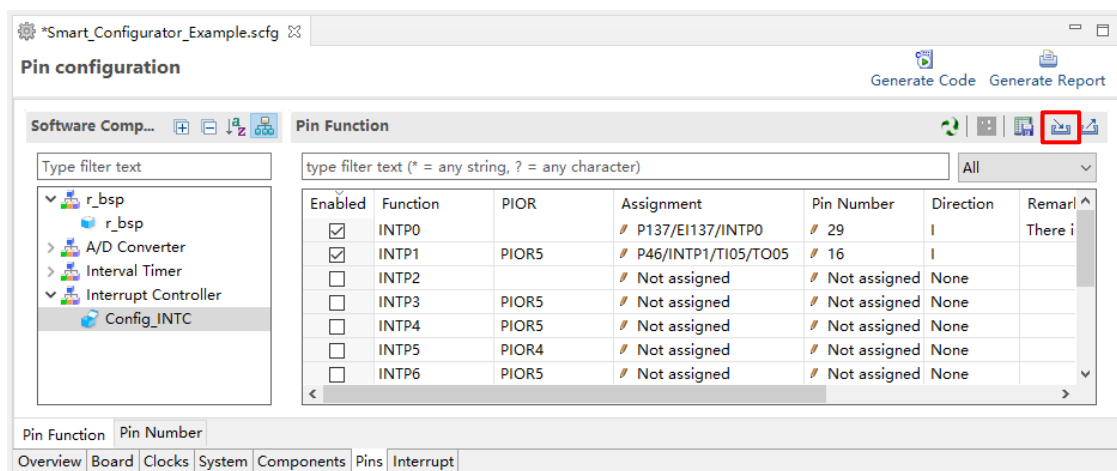


Figure 4-59 Importing Pin Settings from an XML File

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

4.5.7 Pin setting using board pin configuration information

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

The following describes the procedure for collective setting of pins.

- (1) Select a board setting information except [Custom User Board] in [Board] page. User can refer to 4.1.2 Selecting the Board.
- (2) Select [Board Function] in the MCU/MPU Package. (The initial pin configuration of the board can be referred.)
- (3) Open the [Pin Configuration] page and click the [Assign default board pins] button.
- (4) When [Assign default board pins] dialog opens, click [Select all].
- (5) Click [OK].

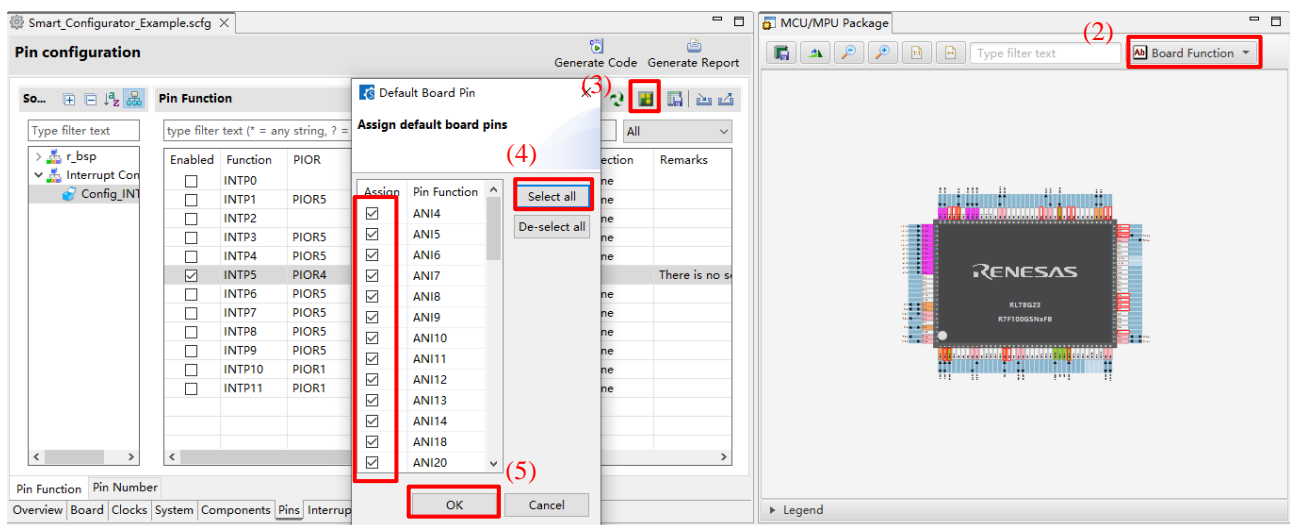


Figure 4-60 Setting for Initial Pin Configuration

If user does not set pin settings all at once, specify them individually in procedure (4).

4.5.8 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.

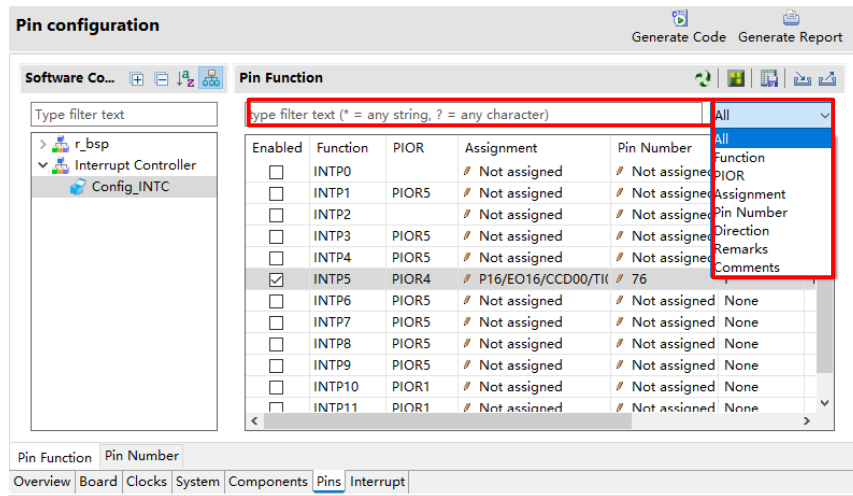


Figure 4-61 Filter for [Pin Function] Page

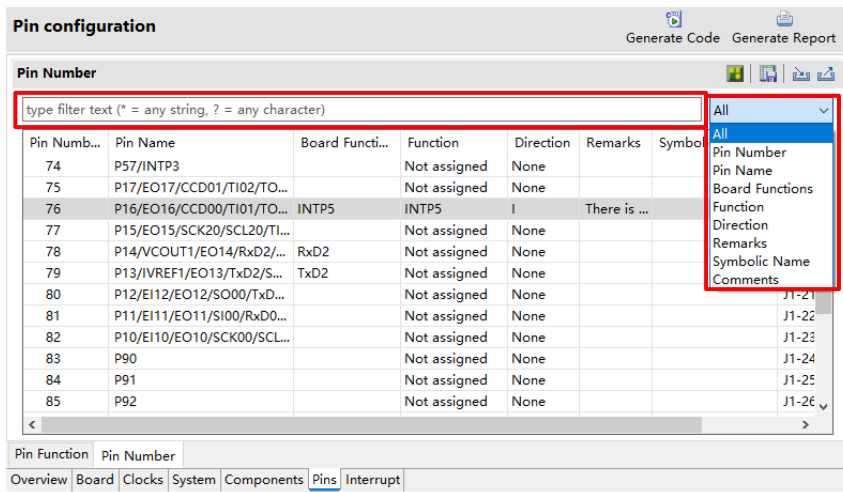


Figure 4-62 Filter for [Pin Number] Page

4.5.9 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.

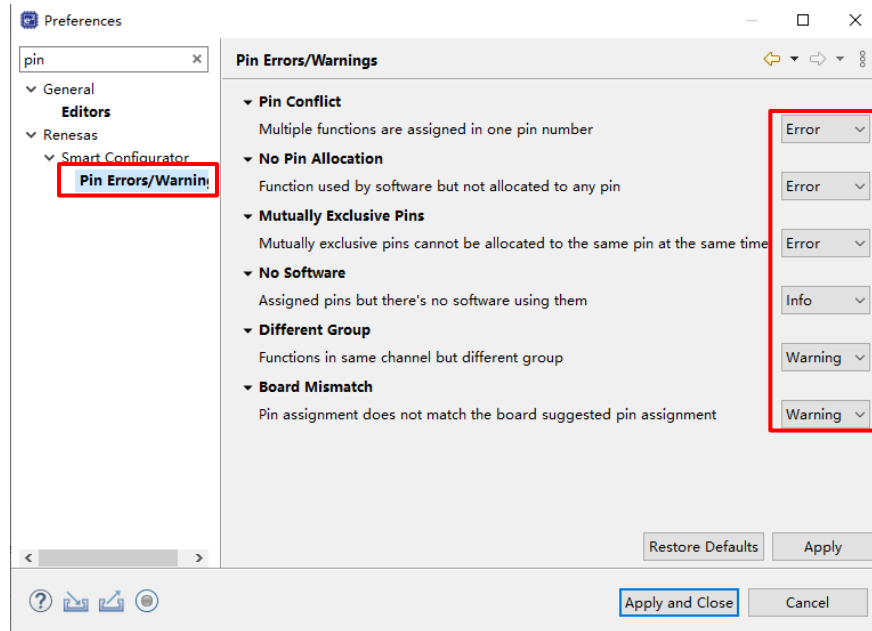


Figure 4-63 Pin Errors/Warnings settings at Preferences

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

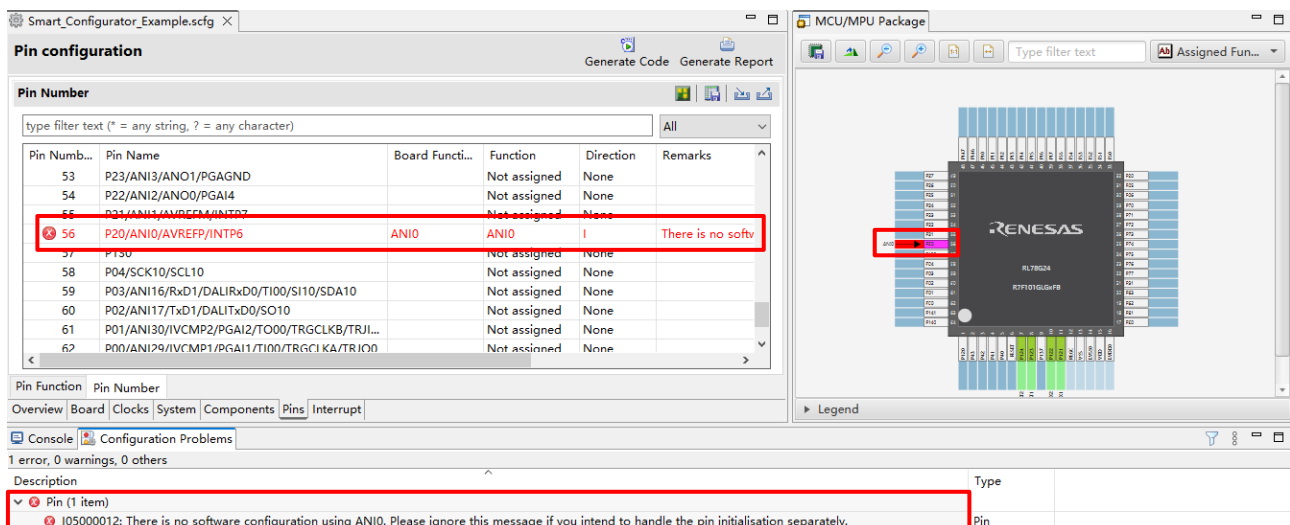
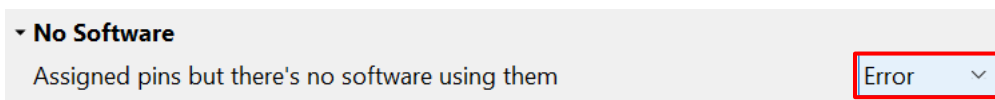




Figure 4-64 Change “No Software” Setting from “Info” to “Error”

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. When an interrupt is used in a Code Generator configuration on the [Components] page, the status of the interrupt will be changed to "Used".

- (1) To display the used interrupts only, click on the  (Show used interrupts) button.
- (2) Group interrupts are collapsed in the interrupt table. Click on the  (Open) button to expand the view and see the interrupts in the group interrupt list.

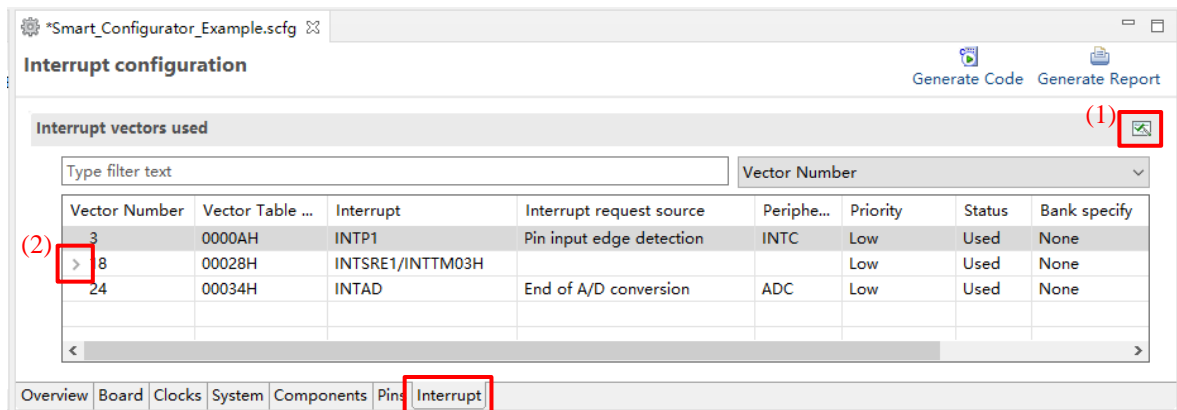


Figure 4-65 [Interrupts] Page

4.6.1 Changing Interrupt Priority Setting

User can change the interrupt priority level on the [Interrupts] page using the following procedure:

- (1) Find the interrupt which user wants to change priority setting on this page.
- (2) Click the priority cell and select an interrupt priority level from the drop-down list.

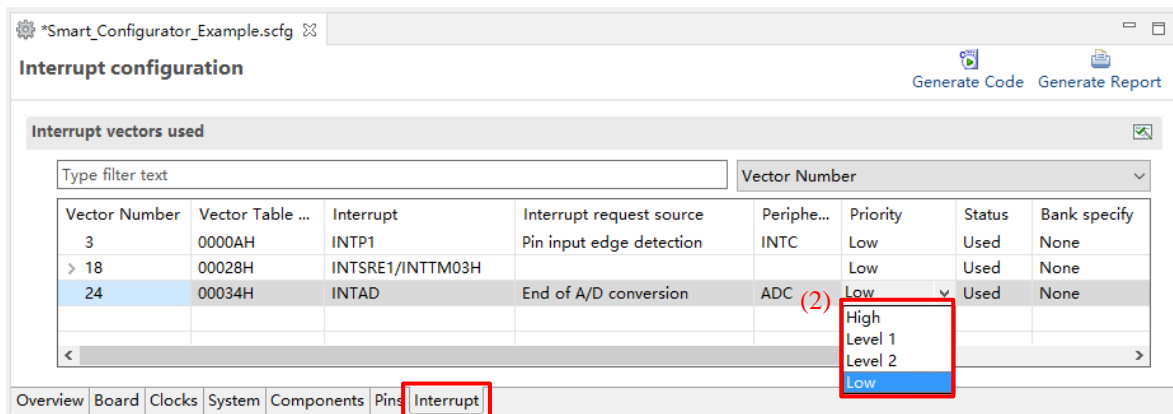


Figure 4-66 Interrupt Settings

4.6.2 Changing Interrupt Bank Setting

User can change the interrupt bank level on the [Interrupts] page using the following procedure:

- (1) Find the interrupt which you want to change bank setting on this page.
- (2) Click the [Bank specify] cell and select a bank setting from the drop-down list (There are four levels [None / 1 / 2 / 3])
- (3) If the same bank levels are selected for different interrupt priorities, a warning mark will be displayed, and warning message is displayed in [Remarks]. User should check and re-set the bank setting.

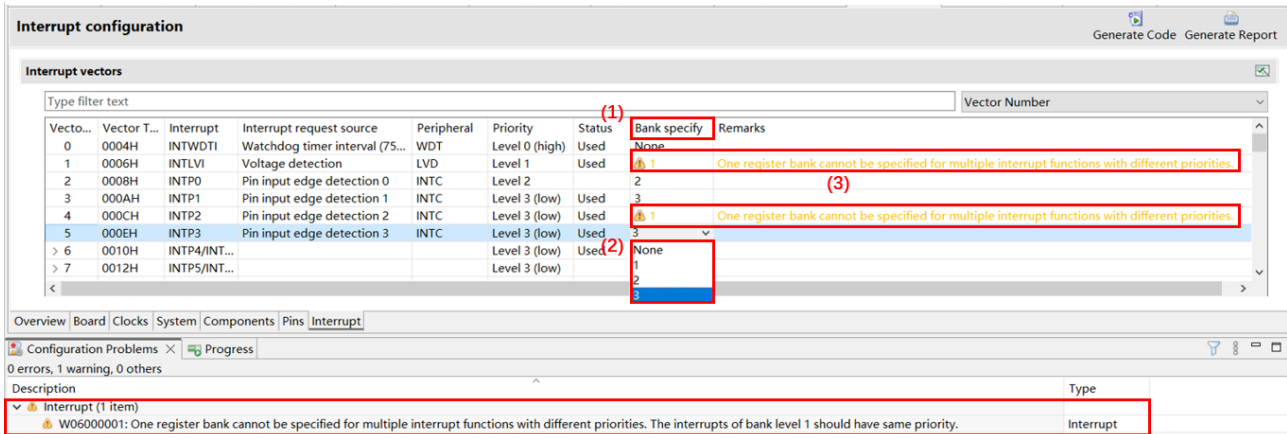


Figure 4-67 Change Interrupt Bank Setting Example

The interrupt bank setting can be reflected into generated code.

- For CCRL compile, interrupt bank code is generated in component's {ConfigurationName}_user.c file.

```

/*****
*****
Pragma directive
*****
*****/
#pragma interrupt r_Config_INTC_intp1_interrupt(vect=INTP1, bank=RB3)
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
    
```

Figure 4-68 Interrupt Bank Setting Example (CCRL Project)

- For LLVM compiler, interrupt bank code is generated in \<ProjectDir>src\smc_gen\general\r_cg_interrupt_handler.h file.

```

/*
 * INT_P1 (0xA)
 */
void r_Config_INTC_intp1_interrupt(void) __attribute__((interrupt(bank=RB3)));
    
```

Figure 4-69 Interrupt Bank Setting Example (LLVM Project)

The concrete generated code specification is different for different compilers; user can get more information in corresponding IDE user guide.

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.

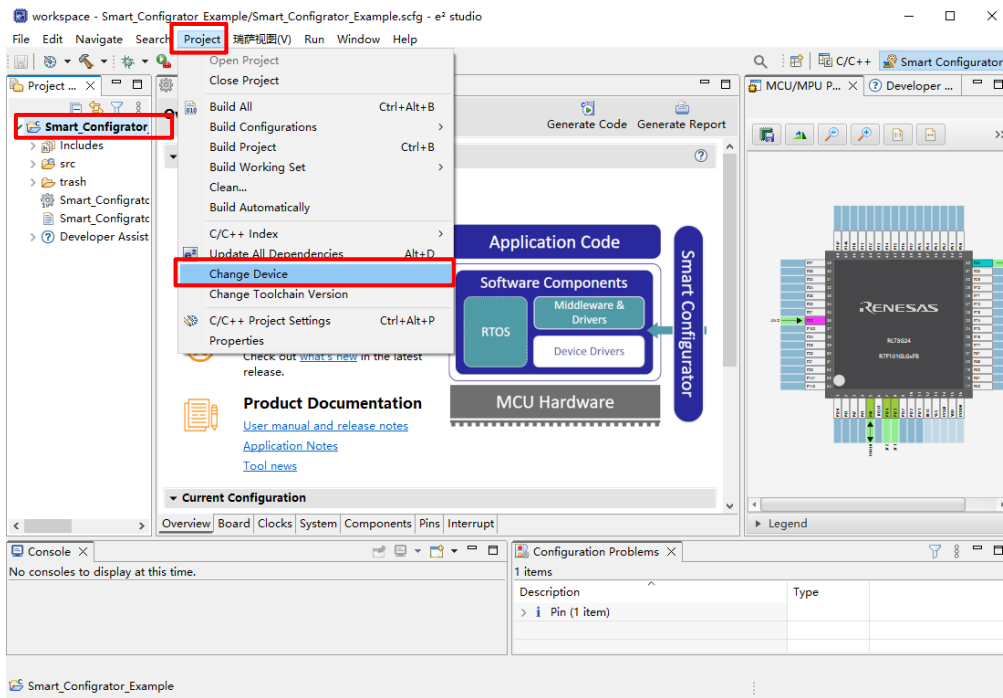


Figure 4-70 Select [Change Device]

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

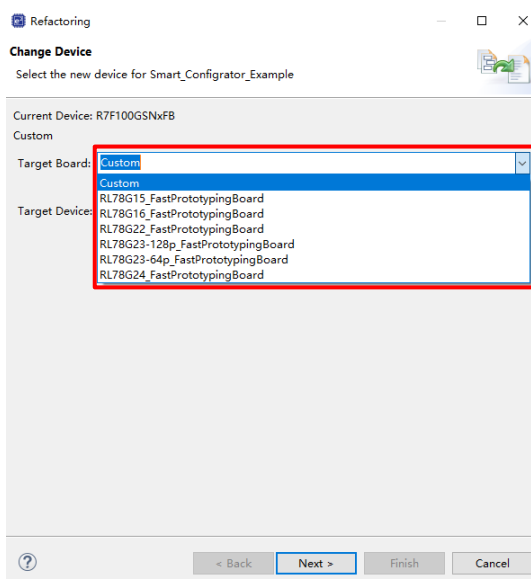


Figure 4-71 Select Target Board

If user want to remain target board as “Custom”, select the target device manually from the device selection list and click "OK". (Wild card search is supported) (for e.g., change to RL78 - G23 100pin Part number: R7F100PGxFB).

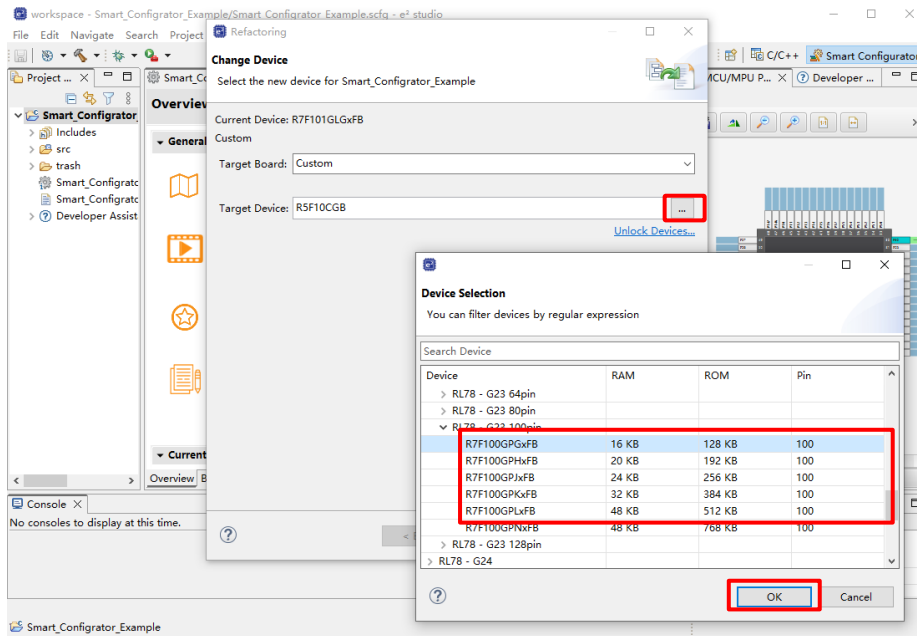


Figure 4-72 Select Target Device

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

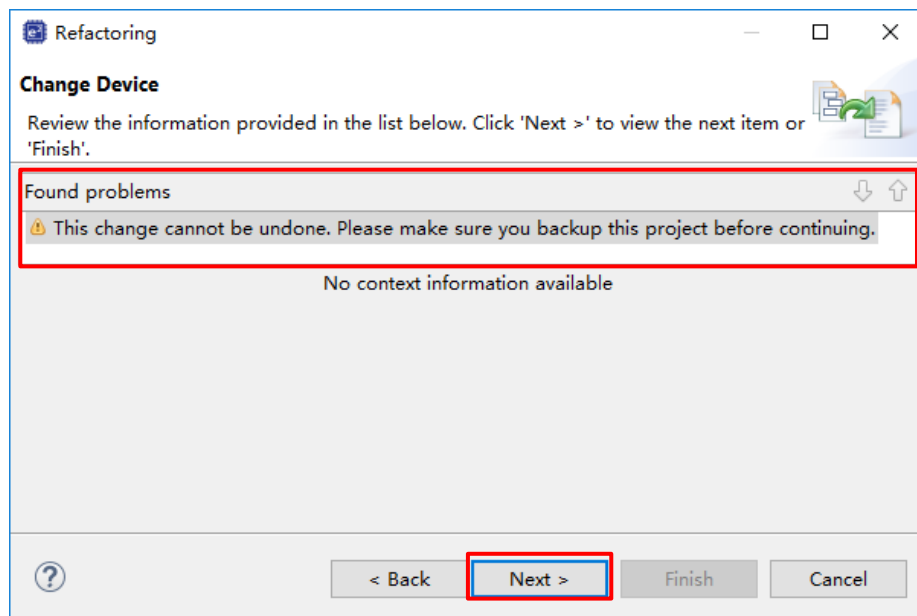


Figure 4-73 Found Problems

Table 4-3 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".

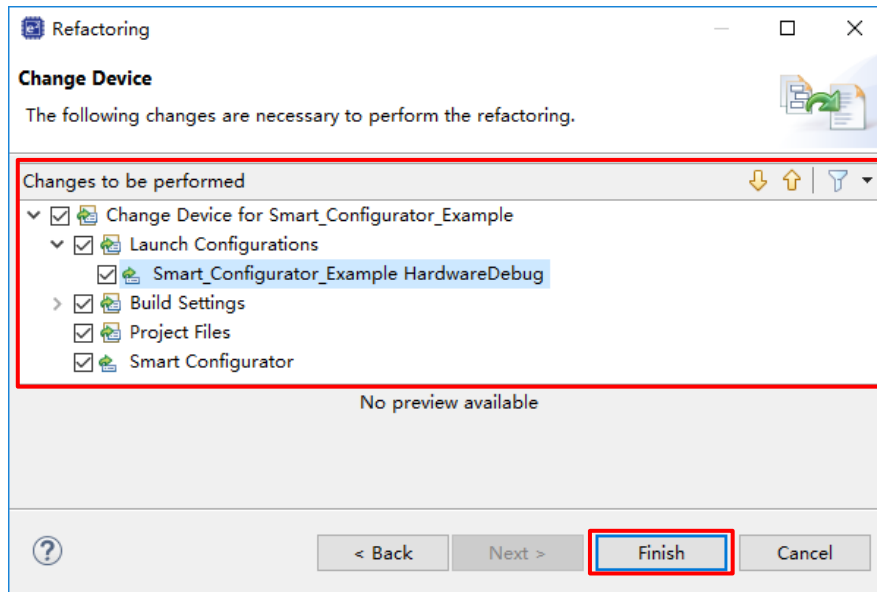


Figure 4-74 Changes to be Performed

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

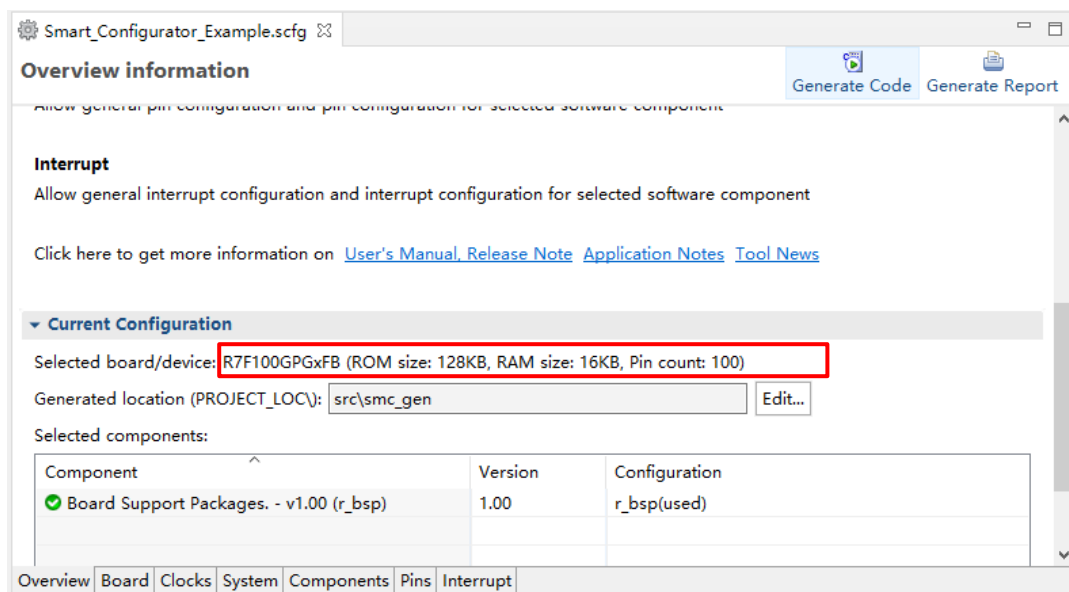


Figure 4-75 Device Update Confirmation

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



Figure 4-76 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.

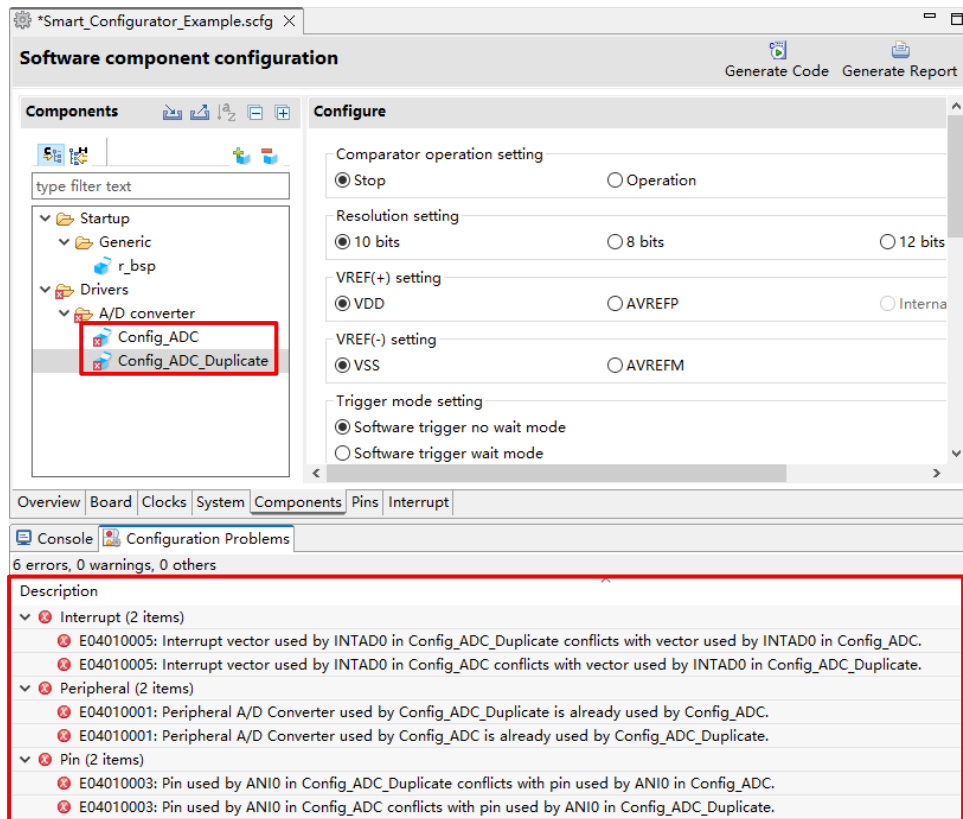


Figure 5-1 Resource Conflicts

5.2 Resolving Pin Conflicts

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

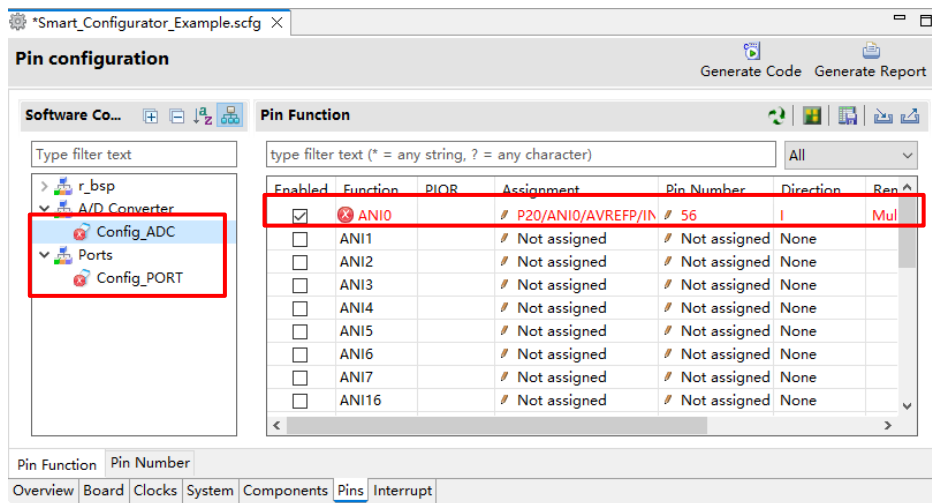


Figure 5-2 Pin Conflicts

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

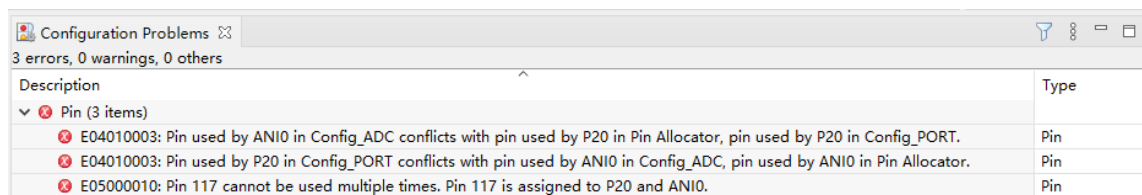


Figure 5-3 Pin Conflict Messages

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

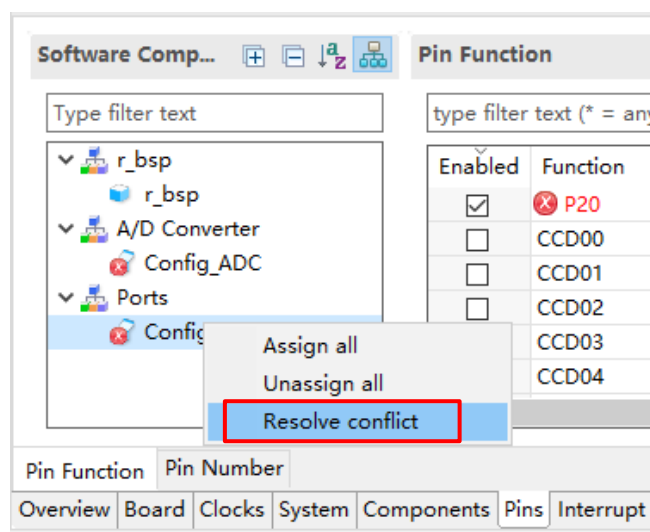


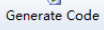
Figure 5-4 Resolving Pin Conflicts

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.

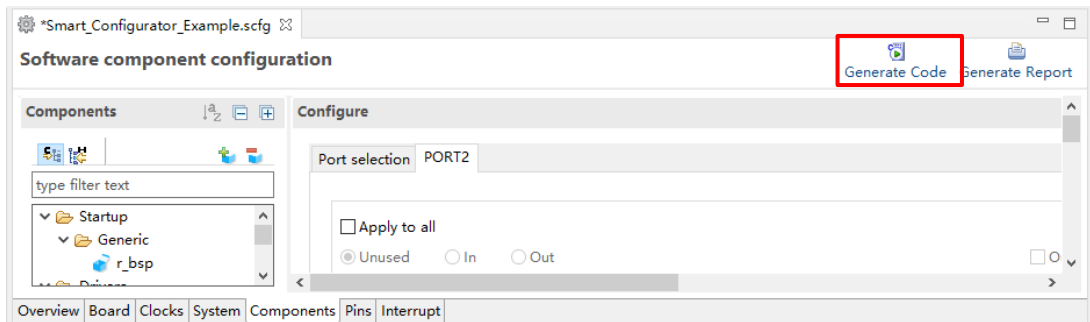


Figure 6-1 Generating a Source File

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

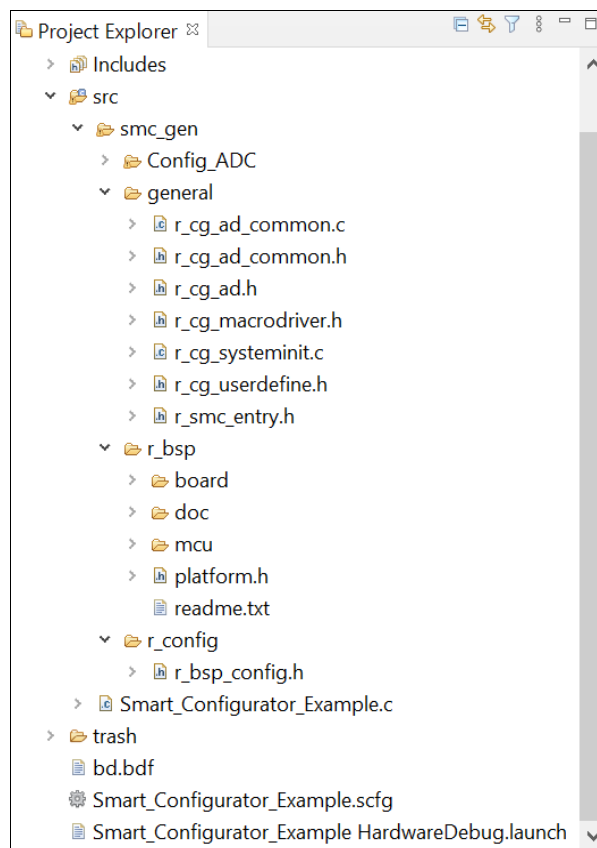


Figure 6-2 Source Files in the Project Explorer

6.2 Change Generated Code Location

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

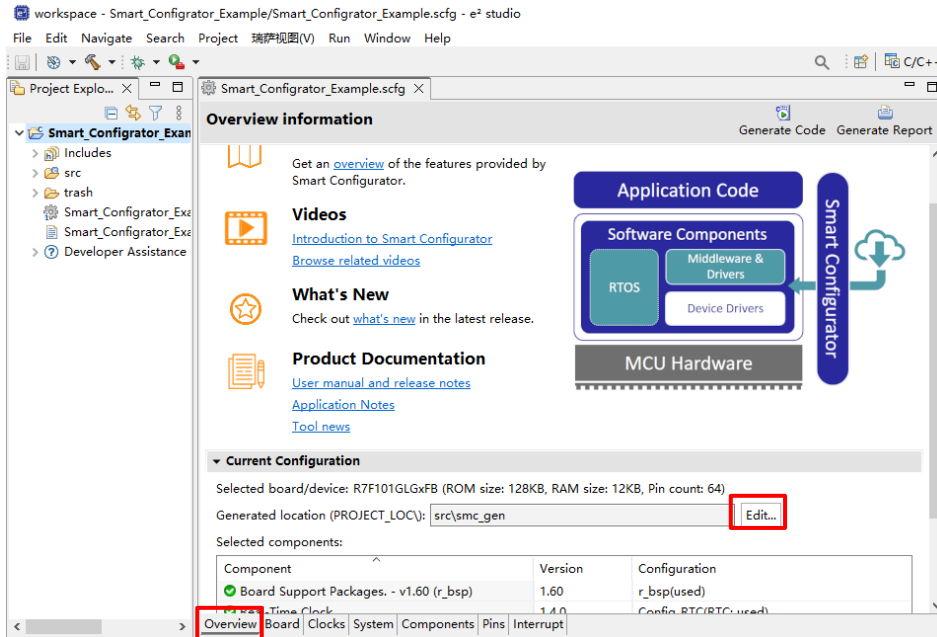


Figure 6-3 Edit the Generated Code Location

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

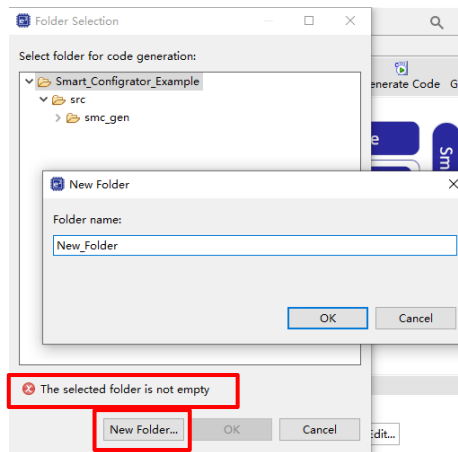



Figure 6-4 Folder Selection

- 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.

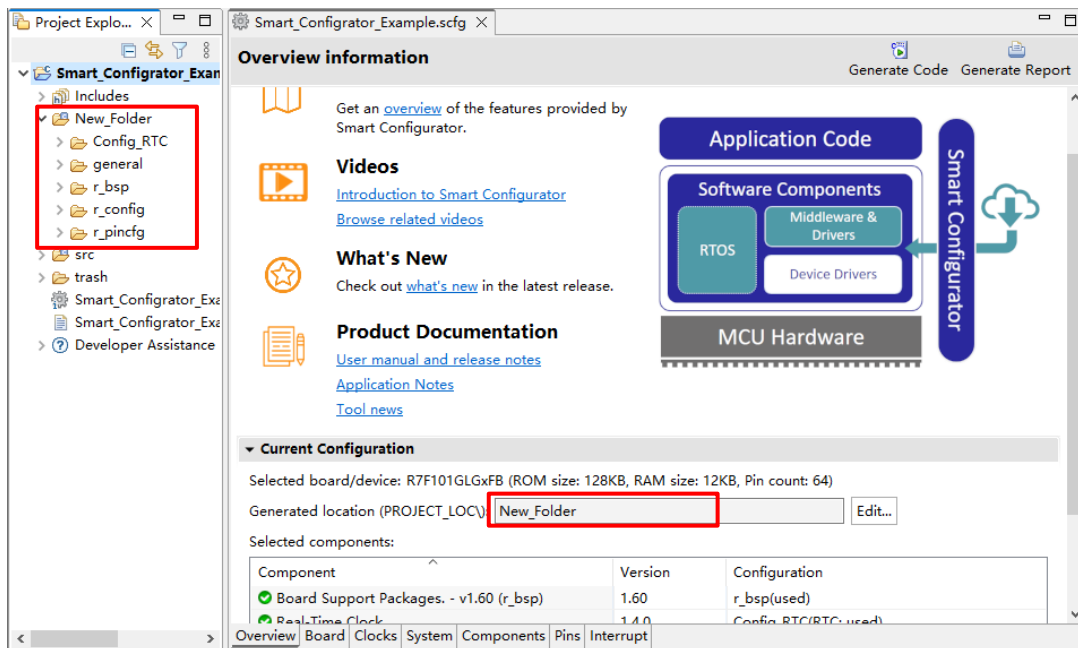


Figure 6-5 New Generate Code Location

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 {Project name}.c, which is generated when the project is created by the e² studio.

R_XXX indicates the names of Software Integration System Modules, "ConfigName" indicates the name of the configuration formed by the component settings, and "Project name" indicates a project name set in the e² studio.

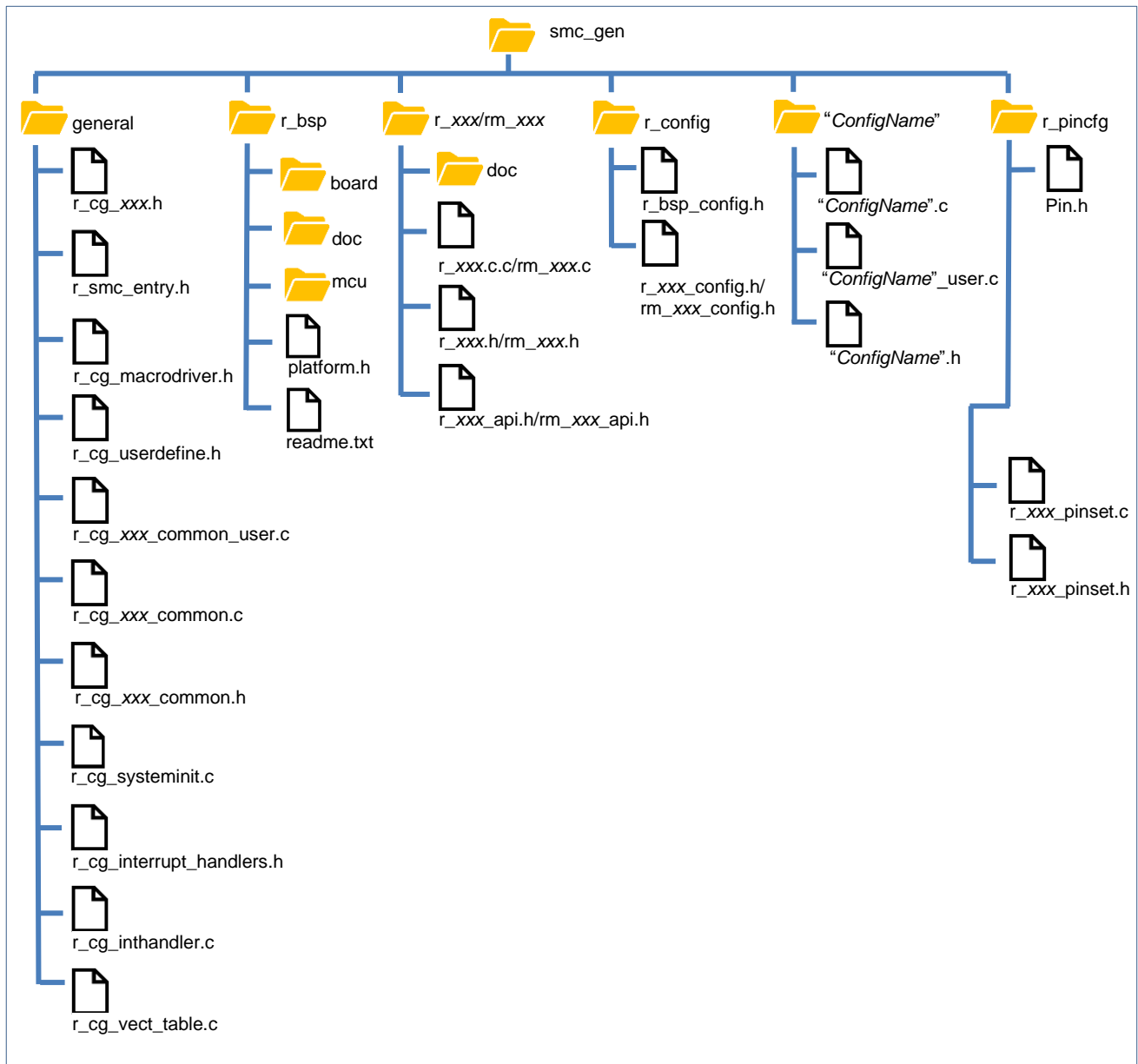


Figure 6-6 Configuration of Generated Files and File Names

Folder	File	Description
general		This folder is always generated. It contains header files and source files commonly used by Code Generator drivers of the same peripheral function.
	R_cg_xxx.h ^(Note*1)	The files contain macro definitions for setting SFR registers.
	R_smc_entry.h	This file is always generated. This file includes the header files of Code Generator drivers that are added to the project. When using functions of Code Generator drivers in source files added by user, including this file is necessary.
	R_cg_macrodriver.h	This file is always generated. This header file contains common macro definitions used in drivers.
	R_cg_userdefine.h	This file is always generated. User can add macro definitions in the dedicated user code areas.
	R_cg_systeminit.c	This file is always generated. This file contains all component's Create () function, it is used for peripheral modules initialization.
	R_cg_xxx_common_user.c ^(Note*1)	The files contain common interrupt API of used peripherals.
	R_cg_xxx_common.c ^(Note*1)	This file is generated when related peripherals are used.
	R_cg_xxx_common.h ^(Note*1)	This file is generated when related peripherals are used.
	R_cg_interrupt_handlers.h ^(Note*2)	This file includes all interrupt routine declaration. If no configuration created, all interrupt routine is default routine; if specific configuration is created, corresponding interrupt routine declaration of this configuration will replace the default routine declaration.
	R_cg_Inthandler.c ^(Note*2)	This file includes all default interrupt routine definition.
	r_cg_vect_table.c ^(Note*2)	This file includes an interrupt vector table which includes all interrupt routine entry address. if no configuration created, all interrupt routine entry address is default; if specific configuration is created, corresponding interrupt routine entry address of this configuration will replace the default routine entry address.
r_bsp		This folder is always generated. It consists of multiple subfolders (board, doc, mcu) with: <ul style="list-style-type: none"> - Initialization codes to start up the MCU before entering main () (e.g. setup stack, initialize memory) - Definitions of all SFR registers in iodefine.h (mcu folder) - Application note of r_bsp (doc folder) It also contains platform.h that will include r_bsp.h of the device used in the project.
r_xxx/ rm_xxx ^(Note*1)		This folder is generated for the RL78 Software Integration System module that is added to the project. It consists of: <ul style="list-style-type: none"> - doc folder: Application note of this RL78 Software Integration System module - r_xxx.c/rm_xxx.c^(Note*1): RL78 Software Integration System module source file - r_xxx.c/rm_xxx.h^(Note*1): RL78 Software Integration System header file - r_xxx_api.h/rm_xxx_api.h^(Note*1): List of all API calls and interface definitions of this RL78 Software Integration System module

Folder	File	Description
r_config		It contains configuration header files for the MCU package, clocks, interrupts, and RL78 Software Integration System drivers/middleware.
	r_bsp_config.h	This file is always generated. It contains configurations of r_bsp for clock initialization and other MCU related settings. Some MCU related settings are generated by Smart Configurator (e.g. package type) and other settings (e.g. stack size) are configured by user manually.
	r_xxx_config.h/rm_xxx_config.h ^(Note*1)	These are configuration header files for all RL78 Software Integration drivers/middleware that are added to the project.
r_pincfg	Pin.h	This file is always generated. It is generated for supporting pin symbol and included in smc_entry.h.
	r_xxx_pinset.c	This file is RL78 Software Integration System module pin setting source file.
	r_xxx_pinset.h	This file is RL78 Software Integration System module pin setting header file.
{ <i>ConfigName</i> }		This folder is generated for the Code Generator drivers that are added to the project. API functions in this folder are named after the <i>ConfigName</i> (configuration name).
	{ <i>ConfigName</i> }.c	This file contains functions to initialize driver (R_ <i>ConfigName</i> _Create) and perform operations that are driver-specific, e.g. start (R_ <i>ConfigName</i> _Start) and stop (R_ <i>ConfigName</i> _Stop).
	{ <i>ConfigName</i> }_user.c	This file contains interrupt service routines and functions for user to add code after the driver initialization (R_ <i>ConfigName</i> _Create). User can add codes and functions in the dedicated user code areas.
	{ <i>ConfigName</i> }.h	This is header file for { <i>ConfigName</i> }.c and { <i>ConfigName</i> }_user.c.

Notes: 1. xxx is the name of a peripheral function.
2. Only LLVM toolchain supports this file.

6.4 Initializing Clocks

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

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

The screenshot shows the 'Smart Configurator' interface with the 'Clocks configuration' page active. The 'Middle-speed on-chip oscillator' is selected and highlighted with a red box, with its frequency set to 4 MHz. Below the interface, the generated code in `r_bsp_config.h` is shown, with the line `0 : 4MHz` highlighted in red, indicating the selected configuration.

```

/* Selection of middle-speed on-chip oscillator clock frequency
Middle-speed on-chip oscillator frequency select register (MOCODIV)
MOCODIV1, MOCODIVO
0 : 4MHz
1 : 2MHz
2 : 1MHz
Other than above : Setting prohibited
*/
#define BSP_CFG_MOCO_DIVIDE (0) /* Generated value. Do not edit this manually */
    
```

Figure 6-7 Clocks Configuration and Generated Code in `r_bsp_config.h`

Folder	File	Macros/Functions	Description
r_config	<code>r_bsp_config.h</code>	Macros related to clocks	These settings are generated by Smart Configurator based on user's selection in the [Clocks] page for the clock source. <code>r_bsp</code> will handle the clock initialization before entering <code>main ()</code> .
		Macros related to MCU settings	Some MCU related settings are generated by Smart Configurator (e.g. package type) macros. For the detail macro information, user can refer to the application note in <code>r_bsp</code> folder: <code>\src\smc_gen\r_bsp\doc</code>

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

6.5 Initializing Pins

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

(1) Pin initialization for drivers with {ConfigName}

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

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

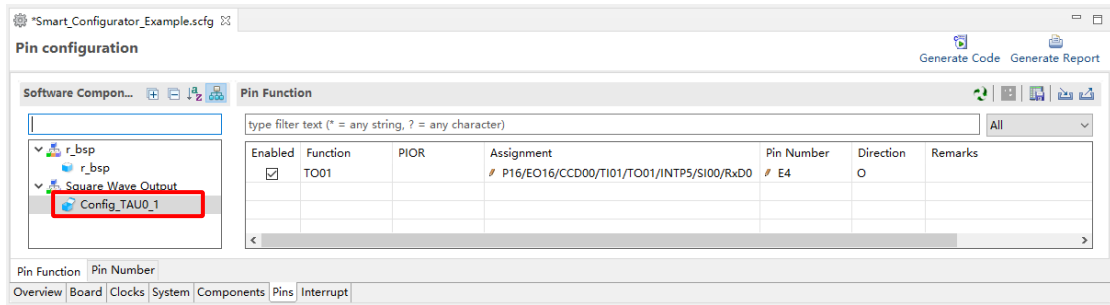


Figure 6-8 Config_TAU0_1 in Software Components View

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

(2) Pin initialization for RL78 Software Integration System component

Pin functions are initialized in R_*{PeripheralName}*_PinSetInit of the file \src\smc_gen\r_pincfg\{*ConfigName*}_pinset.c.

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

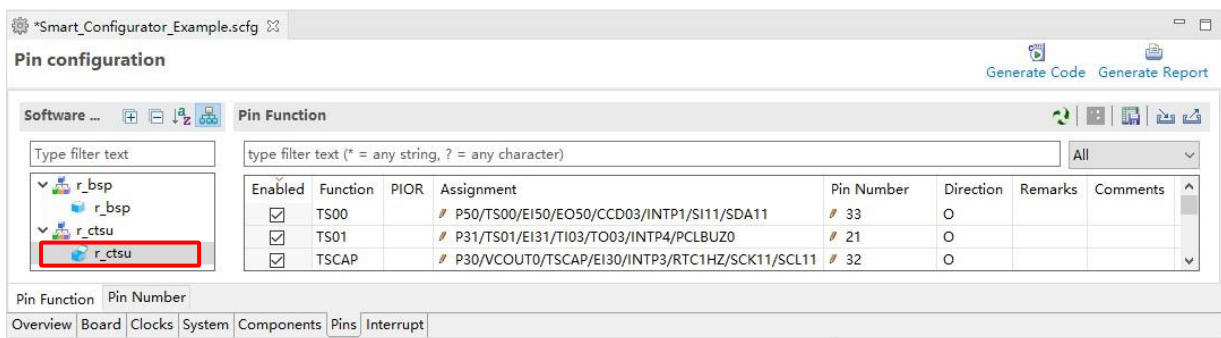


Figure 6-9 r_ctsu in Software Components View

Folder	File	Function	Component type	Description
r_pincfg	{ <i>ConfigName</i> }_pinset.c	R_ <i>{PeripheralName}</i> _PinSetInit	RL78 Software Integration System	This API function initializes the pins used by this driver. User need call this function in main () function.

6.6 Initializing Interrupts

Configurations in the [Interrupts] page are generated in some source files. Interrupt functions are initialized in `R_ConfigName_Create` of the file `\src\smc_gen\{ConfigName}\{ConfigName}.c`.

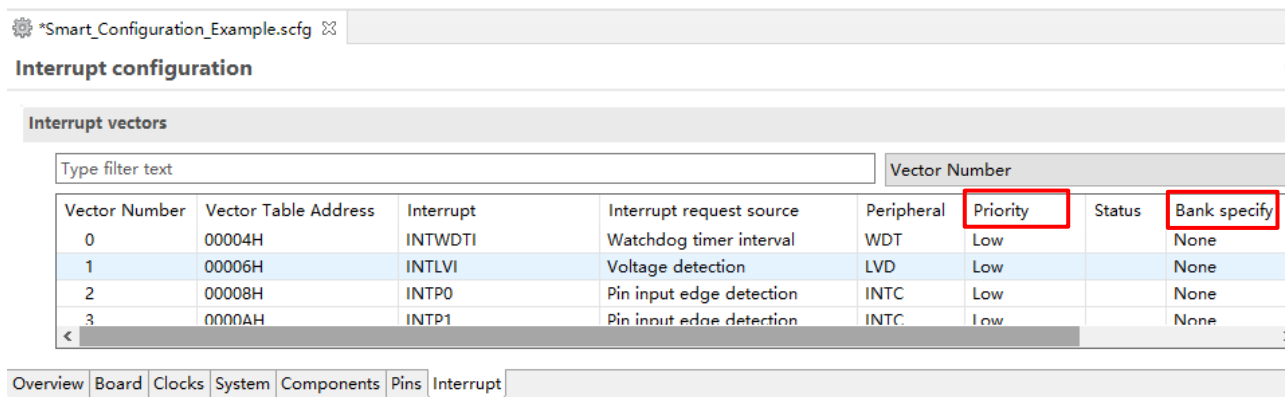


Figure 6-10 Interrupts Configuration in Interrupts View

Item	Folder	File	Component type	Description
Priority	{ConfigName}	{ConfigName}.c	Code Generator	It is initialized in <code>R_ConfigName_Create</code> of this file. <code>R_systeminit</code> in <code>r_cg_systeminit.c</code> will call this function before entering <code>main ()</code> function.
Bank (CCRL Project)	{ConfigName}	{ConfigName}_user.c	Code Generator	Declaration of interrupt as: #pragma interrupt "Interrupt API Name"(vect="Interrupt Name", bank=RBbankNumber). Please see example in Figure 4-68
Bank (LLVM Project)	{%projectDIR%src%smc_gen%general}	r_cg_interrupt_handle.h	-	Declaration of interrupt as: void "Interrupt API Name" (void) __attribute__((interrupt(bank=RBbankNumber))); Please see example in Figure 4-69

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

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

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

In the case of [Code Generator], three files are generated for each of the specified peripheral functions. The file names are "Config_xxx.h", "Config_xxx.c", and "Config_xxx_user.c" as the default, with "xxx" representing the name of the peripheral module. For example, "xxx" will be "ADC" for the A/D Converter (resource ADC). 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 ADC (Config_ADC_user.c).

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

/******
*****
Pragma directive
*****
*****/
#pragma interrupt r_Config_ADC_interrupt(vect=INTAD)
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

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



```
/******  
*****  
* Function Name: R_Config_ADC_Create_UserInit  
* Description : This function adds user code after initializing the AD  
converter.  
* Arguments : None  
* Return Value : None  
*****  
*****/  
void R_Config_ADC_Create_UserInit(void)  
{  
    /* Start user code for user init. Do not edit comment generated here */  
    /* End user code. Do not edit comment generated here */  
}  
  
/******  
*****  
* Function Name: r_Config_ADC_interrupt  
* Description : This function is INTAD interrupt service routine.  
* Arguments : None  
* Return Value : None  
*****  
*****/  
static void __near r_Config_ADC_interrupt(void)  
{  
    /* Start user code for r_Config_ADC_interrupt. Do not edit comment  
generated here */  
    /* End user code. Do not edit comment generated here */  
}  
  
/* Start user code for adding. Do not edit comment generated here */  
/* End user code. Do not edit comment generated here */
```

7.2 Using Generated Code in User Application

To use the generated code of RL78 Software Integration System Modules and Code Generator, follow the below steps:

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

In case of RL78 Software Integration System Modules, it is `r_XXX.h`.

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

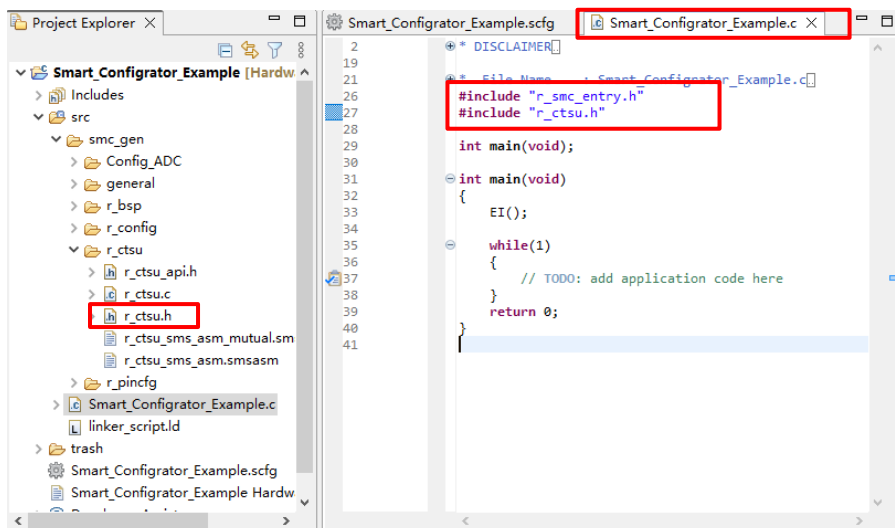


Figure 7-1 Add Header Files

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

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

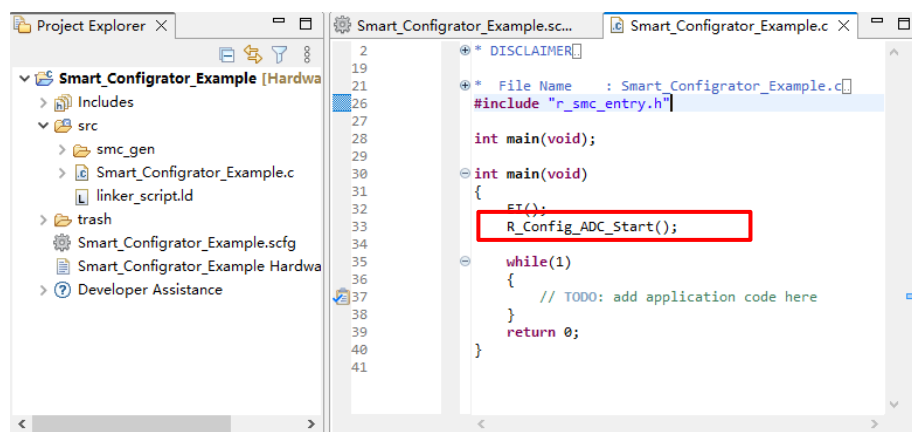



Figure 7-2 Call Code Generator Functions

In case of Software Integration System Modules, refer to the examples provided in the “API Functions” chapter of corresponding Application Note.

8. Backing up Generated Source Code

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

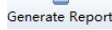
<ProjectDir>\trash\<>Date-and-Time>

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).

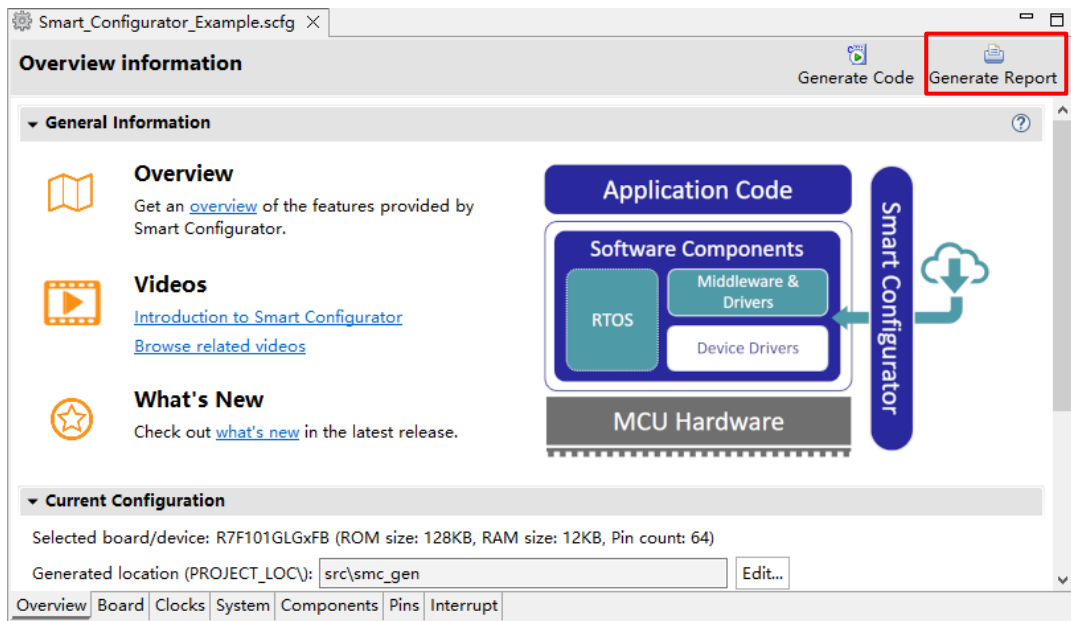


Figure 9-1 Output of a Report on the Configuration (as a PDF/Text File)

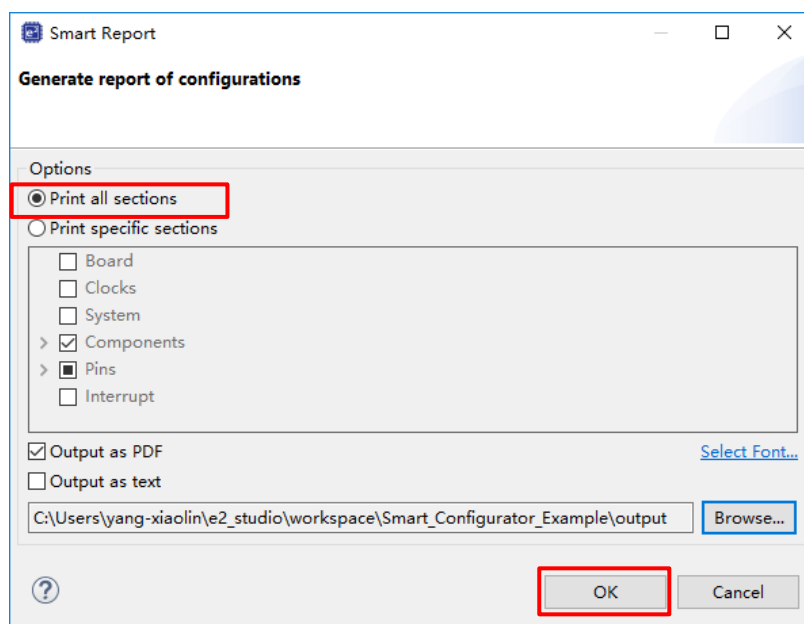



Figure 9-2 Dialog Box for Output of a Report (Example is selecting “Output as PDF”)

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.

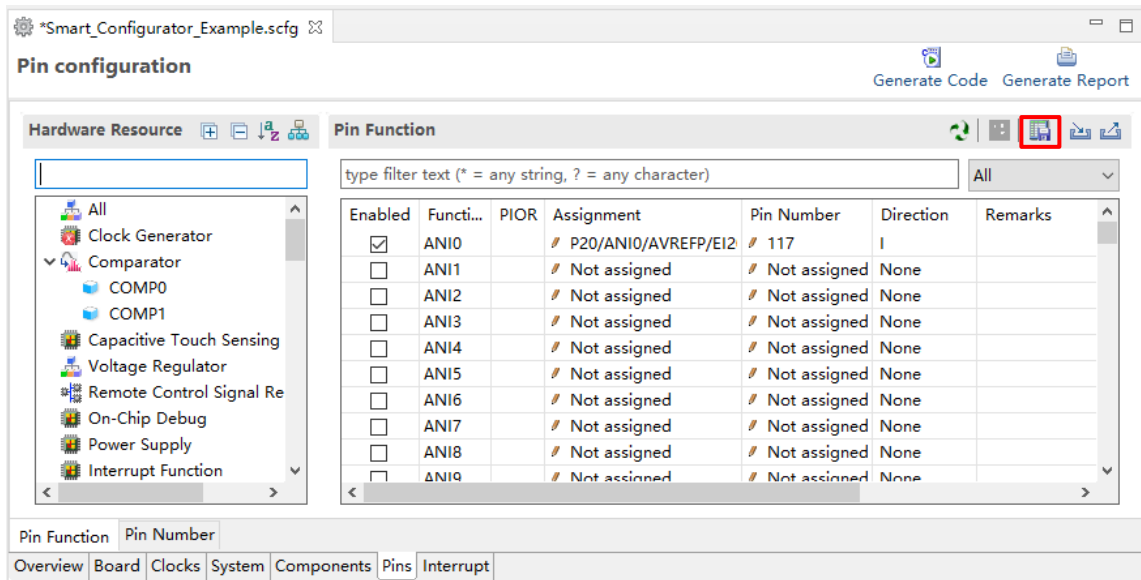



Figure 9-3 Output of a List of Pin Functions or Numbers (in csv Format)

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.

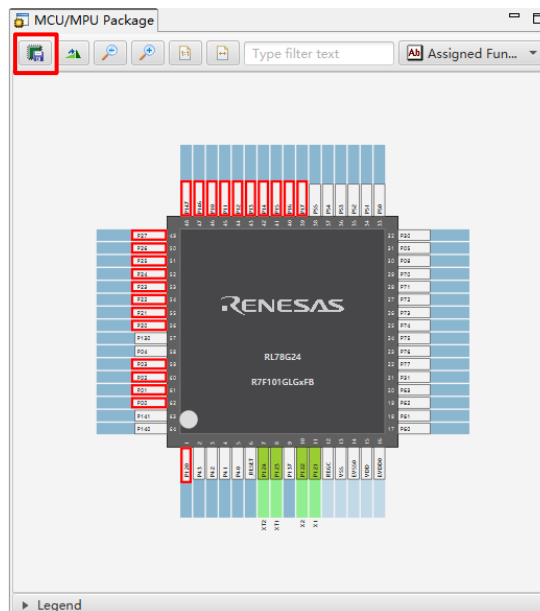


Figure 9-4 Outputting a Figure of MCU/MPU Package (in png Format)

10. Developer Assistance

Developer assistance function will provide a virtual root node named "Developer Assistance" in project tree when Smart Configurator selected in Project Generator.

Through "Developer Assistance" tree, user can navigate API information in new Smart Configurator [Developer Assist Browser] view for the added Code Generator components and user can drag and drop the API template node to C/C++ editor during coding.

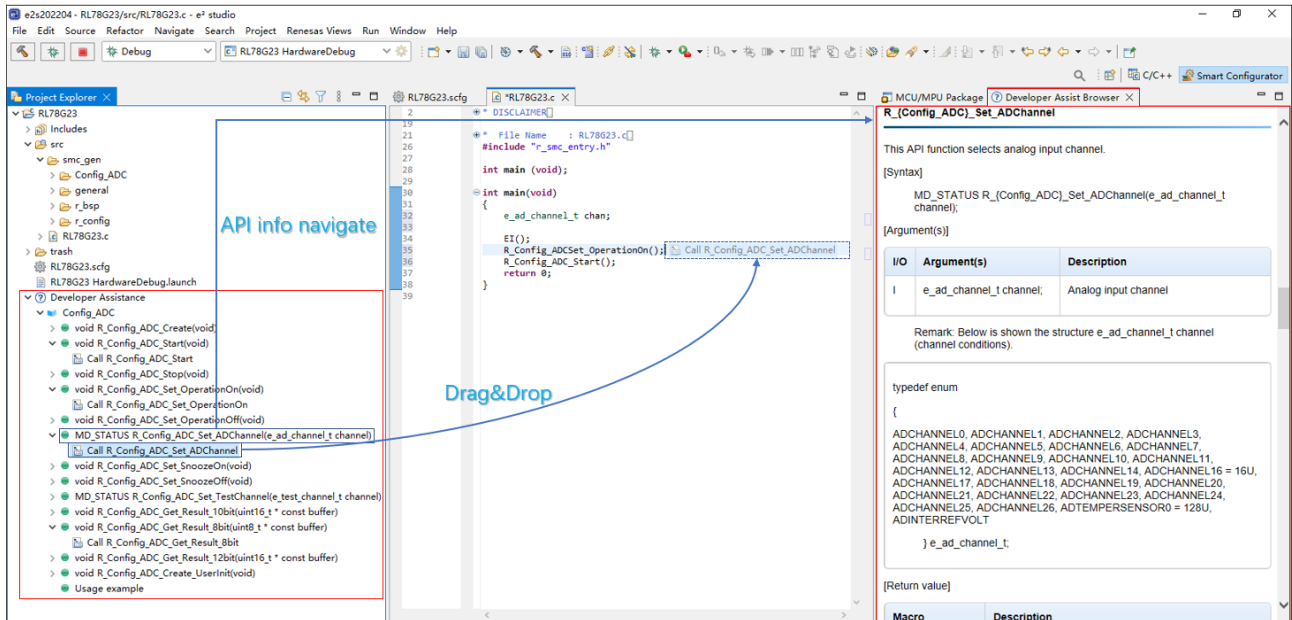


Figure 10-1 API info Navigate and Call API Drag&Drop

User can select text in [Developer Assist Browser] view and through "Copy" context menu to paste the Code Generator component usage example code snippet to C/C++ editor.

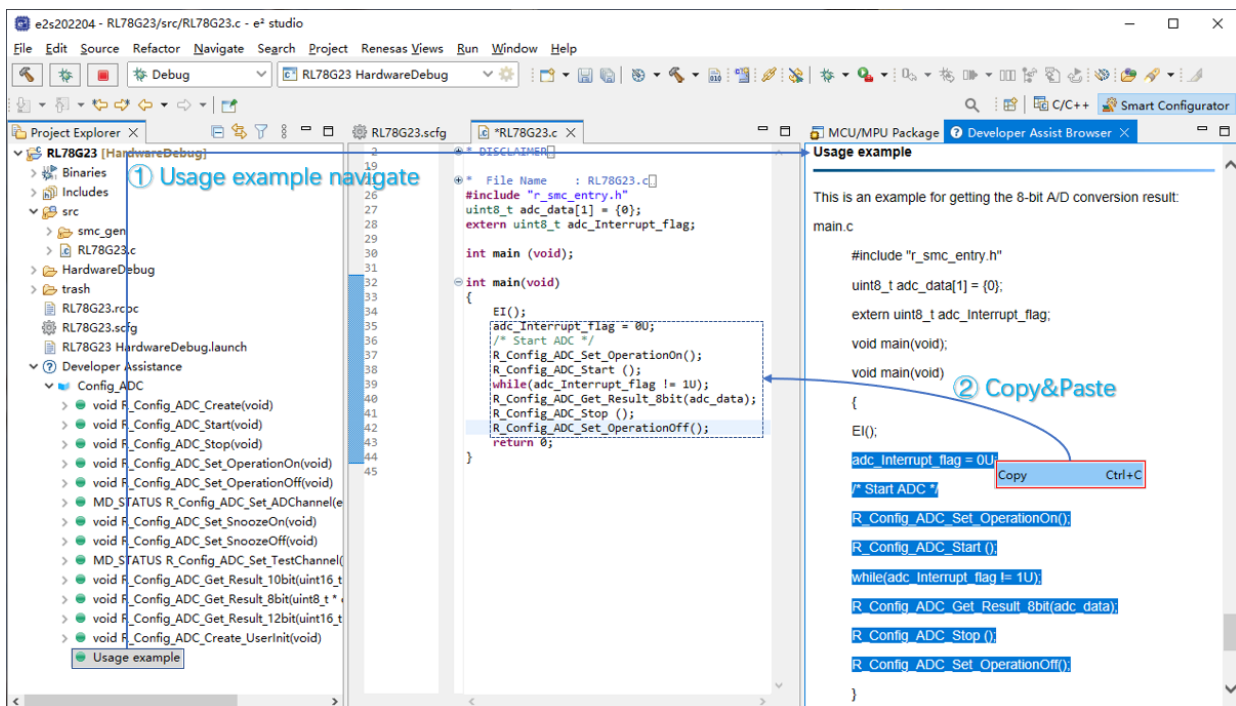


Figure 10-2 Usage Example Copy&Paste

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

The Smart Configurator for RL78 Plug-in in e² studio 2023-01 and later version now incorporates an enhanced user code protection feature. This feature empowers users to insert codes to any location in the generated codes by utilizing the specific tags, as shown in Figure 11-1. After the next code generation, the inserted user codes will be protected and automatically merged into the generated files.

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

11.1 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 11-1 and add the user codes between these tags. If the specific tags do not match exactly, the inserted user code will not be protected after the code generation.

```

/* Start user code */

User code can be added between the specific tags

/* End user code */

```

Figure 11-1 Specific Tags for User Code Protection Feature

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

```

void R_Config_ADC_Create(void)
{
    ADCEN = 1U;    /* supply AD clock */
    ADMK0 = 1U;    /* disable INTAD0 int
    ADIF0 = 0U;    /* clear INTAD0 interr
    /* Set INTAD0 priority */
    ADPR10 = 1U;
    ADPR00 = 1U;
    /* Set ANI0 pin */
    PMCA2 |= 0x01U;
    PM2 |= 0x01U;
    ADM0 = _00_AD_OPERMODE_SELECT | _00_A
    ADM1 = _C0_AD_TRIGGER_HARDWARE_WAIT
    ADM2 = _00_AD_NEGATIVE_VSS | _00_AD_A
    ADUL = _FF_AD_ADUL_VALUE;
    ADLL = _00_AD_ADLL_VALUE;
    /* Start user code */
    AWC = 0U;
    /* End user code */
    ADS = _00_AD_INPUT_CHANNEL_0;
    ADM2 &= _3F_AD_POSITIVE_CLEAR; /*
    ADM2 |= _00_AD_POSITIVE_VDD; /* s
}

R_Config_ADC_Create_UserInit();

```

```

void R_Config_ADC_Create(void)
{
    ADCEN = 1U;    /* supply AD clock */
    ADMK0 = 1U;    /* disable INTAD0 inte
    ADIF0 = 0U;    /* clear INTAD0 interr
    /* Set INTAD0 priority */
    ADPR10 = 1U;
    ADPR00 = 1U;
    /* Set ANI0 pin */
    PMCA2 |= 0x01U;
    PM2 |= 0x01U;
    /* Set AVREFF pin */
    PMCA2 |= 0x01U;
    PM2 |= 0x01U;
    ADM0 = _00_AD_OPERMODE_SELECT | _00_A
    ADM1 = _C0_AD_TRIGGER_HARDWARE_WAIT
    ADM2 = _00_AD_NEGATIVE_VSS | _00_AD_A
    ADUL = _FF_AD_ADUL_VALUE;
    ADLL = _00_AD_ADLL_VALUE;
    /* Start user code */
    AWC = 0U;
    /* End user code */
    ADS = _00_AD_INPUT_CHANNEL_0;
    ADM2 &= _3F_AD_POSITIVE_CLEAR; /*
    ADM2 |= _40_AD_POSITIVE_AVREFF; /*
}

R_Config_ADC_Create_UserInit();

```

Figure 11-2 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 11-1. 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.

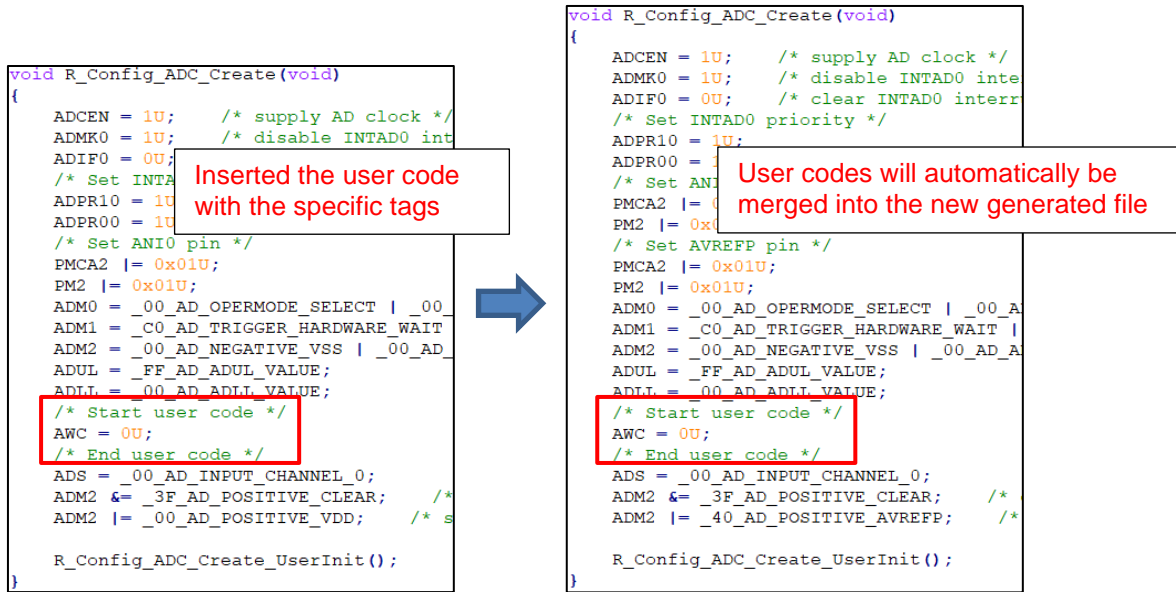


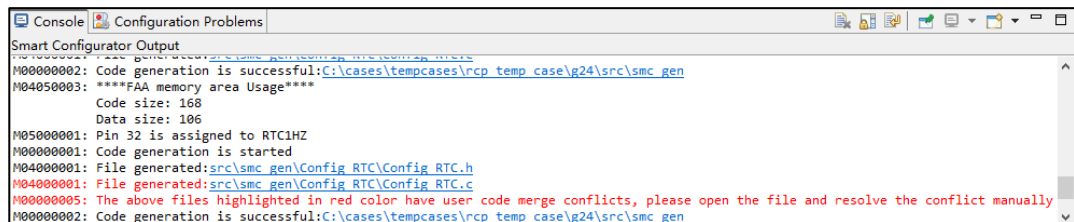
Figure 11-2 User Code Protection with Auto Merge

11.3 What to Do When Merge Conflict Occurs

11.3.1 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 11-3 The Merge Conflict Message Outputted in the Smart Configurator Console.



```
Smart Configurator Output
M0000002: Code generation is successful:C:\cases\tempcases\rpc temp case\g24\src\smc_gen
M04050003: ****FAA memory area Usage****
Code size: 168
Data size: 106
M05000001: Pin 32 is assigned to RTC1HZ
M00000001: Code generation is started
M04000001: File generated:src\smc_gen\Config_RTC\Config_RTC.h
M04000001: File generated:src\smc_gen\Config_RTC\Config_RTC.c
M00000005: The above files highlighted in red color have user code merge conflicts, please open the file and resolve the conflict manually
M00000002: Code generation is successful:C:\cases\tempcases\rpc temp case\g24\src\smc_gen
```

Figure 11-3 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 11.3.2 Steps for Resolving the Merge Conflict described.

11.3.2 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 11-4 Code before Resolving Conflict).
- (2) Click on “Copy Current Change from Left to Right” (Figure 11-4 Code before Resolving Conflict).

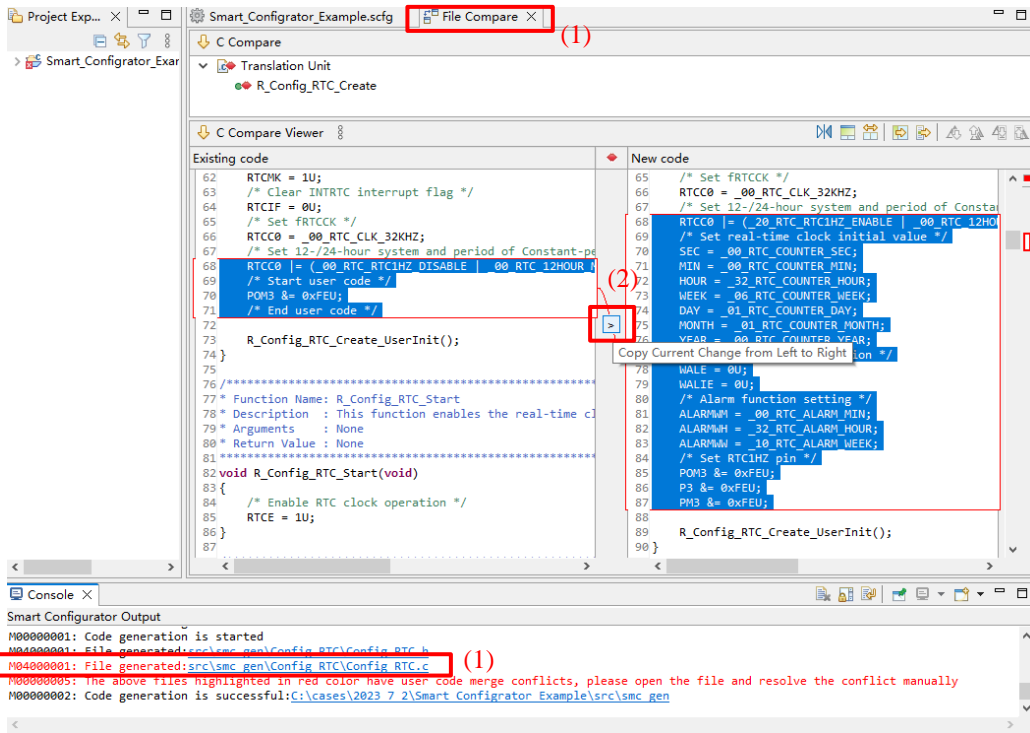


Figure 11-4 Code before Resolving Conflict

- (3) Delete the codes that user does not want to use (Error! Reference source not found.).

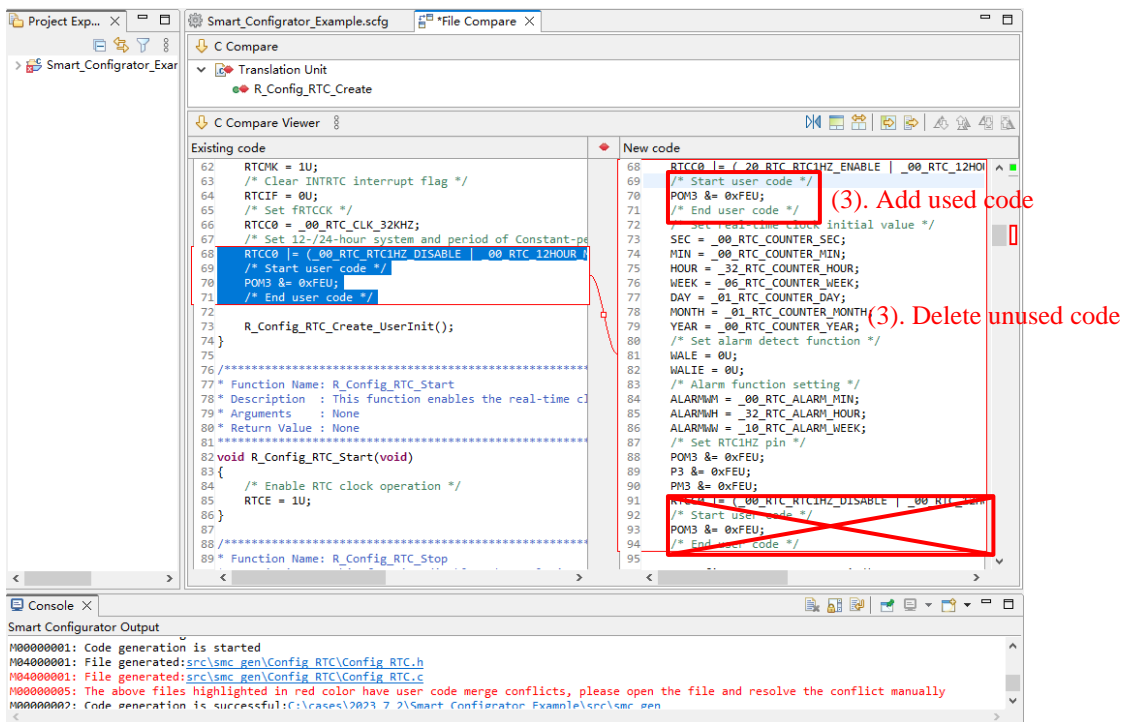


Figure 11-5 Code after Applying “Copy Current Change from Left to Right”

(4) Save the modified code (Figure 11-6 Code after Deleting and Saving).

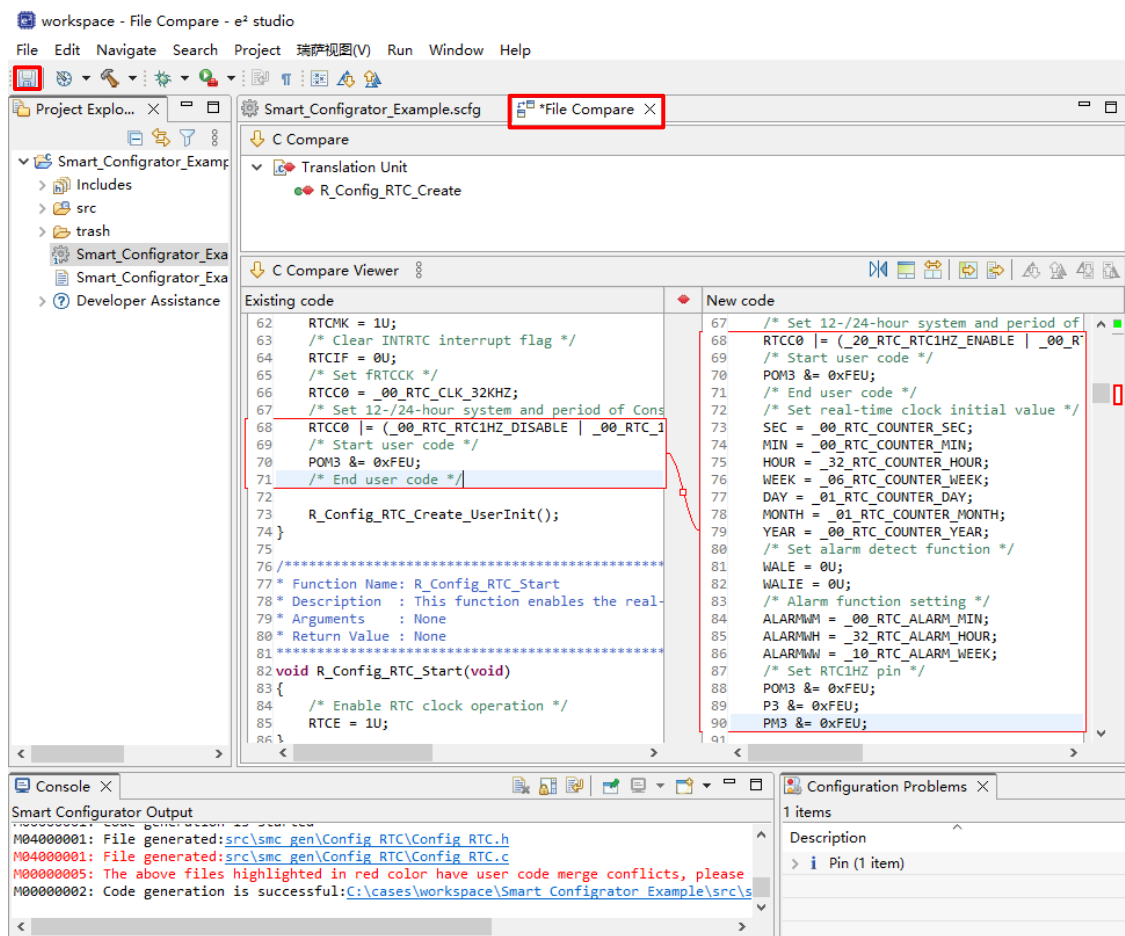


Figure 11-6 Code after Deleting and Saving

User can also resolve the conflict 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.

12. 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.

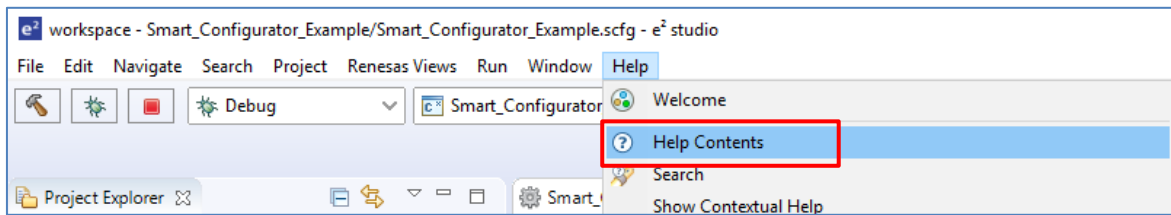



Figure 12-1 Help Menu

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.

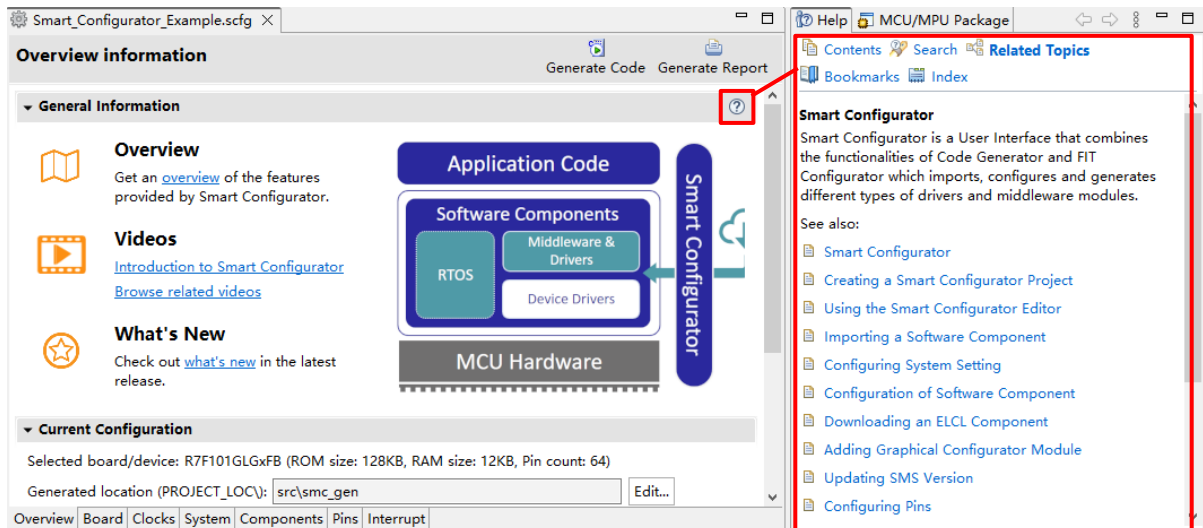


Figure 12-2 Smart Configurator Quick Start inform

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

13. Documents for Reference

User's Manual: Hardware

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

Technical Update/Technical News

Obtain the latest information from the Renesas Electronics website.

User's Manual: Development Environment

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

CC-RL Compiler User's Manual (R20UT3123)

Smart Configurator User's Manual: RL78 API Reference (R20UT4852)

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

SMS & ELCL Application Notes:

Obtain the latest information from the website of Renesas Electronics.

Revision History

Rev.	Section	Description
1.00	-	First edition issued
1.01	Section 2 Creating a Project	2.1 Create a C Project: Updated Figure 2-5, Figure 2-7
	Section 3 Operating the Smart Configurator	3.4 Window: Updated Figure 3-3, Figure 3-4, Figure 3-5
		3.4.6 Developer Assist Browser View: Added chapter 3.4.6 Developer Assist Browser View
Section 10 Developer Assistance	10. Developer assistance: Added chapter 10. Developer assistance	
1.02	Section Introduction	URL was updated.
	Section 4 Setting of Peripheral Modules	4.1.1 Selecting the Device: Device Change Confirmation Operation was deleted.
		4.4.12 Changing Version of BSP Configuration: Note was deleted.
		4.4.13 Configure General Setting of Component: Figure 4-38 Configure General Setting of Component was updated.
		4.4.13 Configure General Setting of Component: Note 1 was updated.
		4.4.13 Configure General Setting of Component: Note 2 was updated.
		4.4.13 Configure General Setting of Component: Note 3 was added.
		4.6.2 Changing Interrupt Bank Setting: The description of step (3) was updated.
4.6.2 Changing Interrupt Bank Setting: Figure 4-55. Change Interrupt Bank Setting Example was modified.		
Section 12 Documents for Reference	SMS & ELCL Application Notes: SMS and ELCL reference was deleted.	
1.03	Section 2 Creating a Project	2. Creating a Project: Add Project Template Selection step in e ² studio project creation wizard.
	Section 3 Operating the Smart Configurator	3.4.3 MCU/MPU Package View: Update description and Figure 3-6. MCU/MPU Package View.
	Section 4 Setting of Peripheral Modules	4.1.2 Selecting the Board: modify description
		4.4.3 Removing Software Component: Add description about removing multiple components from a project.
		4.4.10 Downloading RL78 Software Integration System Modules: Update description
		Add 4.4.11 Adding a RL78 Software Integration System Module
		4.4.12 Setting a RL78 Software Integration System Module: Update description
		4.5 Pin Settings: Update description and Figure 4-48 and 4-49.
		4.5.3 Assigning Pins Using the MCU/MPU Package View: Update description and Figure 4-53.
		Add 4.5.4 Show pin number from pin functions.
	Add 4.5.9 Pin Errors/Warnings setting.	
	4.7 MCU Migration Feature: Update description	
	Section 6 Generating Source Code	Add 6.2 Change Generated Code Location
	6.3 Configuration of Generated Files and File Names: Update the description and Figure 6-6 Configuration of Generated Files and File Names for supporting pin symbol.	
Section 7 Creating User Programs	Add 7.2 Using Generated Code in User Application	

Rev.	Section	Description
1.03	Section 11 User code protection feature for Smart Configurator Code Generation component	Add Section 11 User code protection feature for Smart Configurator Code Generation component.

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

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

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.

Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

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

3. Input of signal during power-off state

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

4. Handling of unused pins

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

5. Clock signals

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

6. Voltage application waveform at input pin

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

7. Prohibition of access to reserved addresses

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

8. Differences between products

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
 - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
 - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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