# RL78 Smart Configurator

User's Guide: IAREW

R20AN0581EC0104
Rev.1.04
Apr.21.25

## Introduction

This application note describes the basic usage of the RL78 Smart Configurator (hereafter called the Smart Configurator), and the procedure for importing its output files to IAR Embedded Workbench.

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

- IAR Embedded Workbench for Renesas RL78 V5.10.3 and later
- RL78 Smart Configurator V1.13.0 and later

### Target device and support compiler

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

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

# Contents

# 1. Overview

## 1.1 Purpose

This application note describes the basic usage of the RL78 Smart Configurator (hereafter called the Smart Configurator), and the procedure for importing its output files to IAR Embedded Workbench.

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

## 1.2 Features

The Smart Configurator is a utility for combining software to meet user's needs. It handles the following three functions to support the embedding of drivers from Renesas in user's 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. Installing and Uninstalling the Smart Configurator

## 2.1 Installing the Smart Configurator

Download the Smart Configurator from the URL below.

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

After activating the installer, install the Smart Configurator by following the procedure of the installer. User will require administrator privileges to do this.

## 2.2 Uninstalling the Smart Configurator

To uninstall the Smart Configurator, please select "Smart Configurator for RL78" from [Apps & features] in the control panel.

# 3. Operating the Smart Configurator

## 3.1 Procedure for Operations

Figure 3-1 Operating Procedure, shows the procedure for generating IAR related files using Smart Configurator and loading it into IAR Embedded Workbench. For the operation of IAR Embedded Workbench, refer to relevant document of IAR.

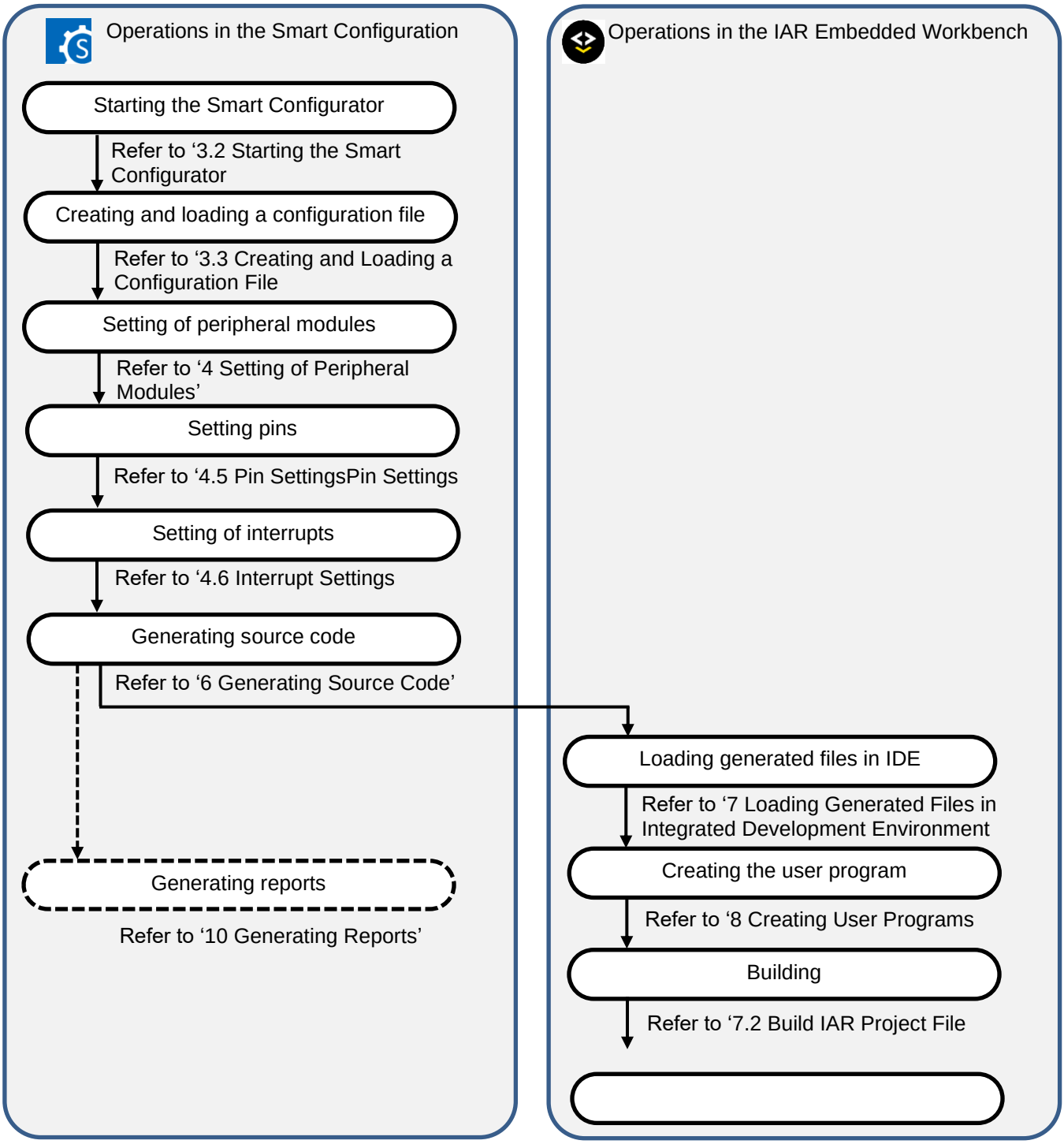



**Figure 3-1 Operating Procedure**

## 3.2 Starting the Smart Configurator

Select [Smart Configurator for RL78 Vx.x.x] of [Renesas Electronics Smart Configurator] from the Windows start menu. The main window of the Smart Configurator will be starting.

Note: Please replace Vx.x.x with user's version.

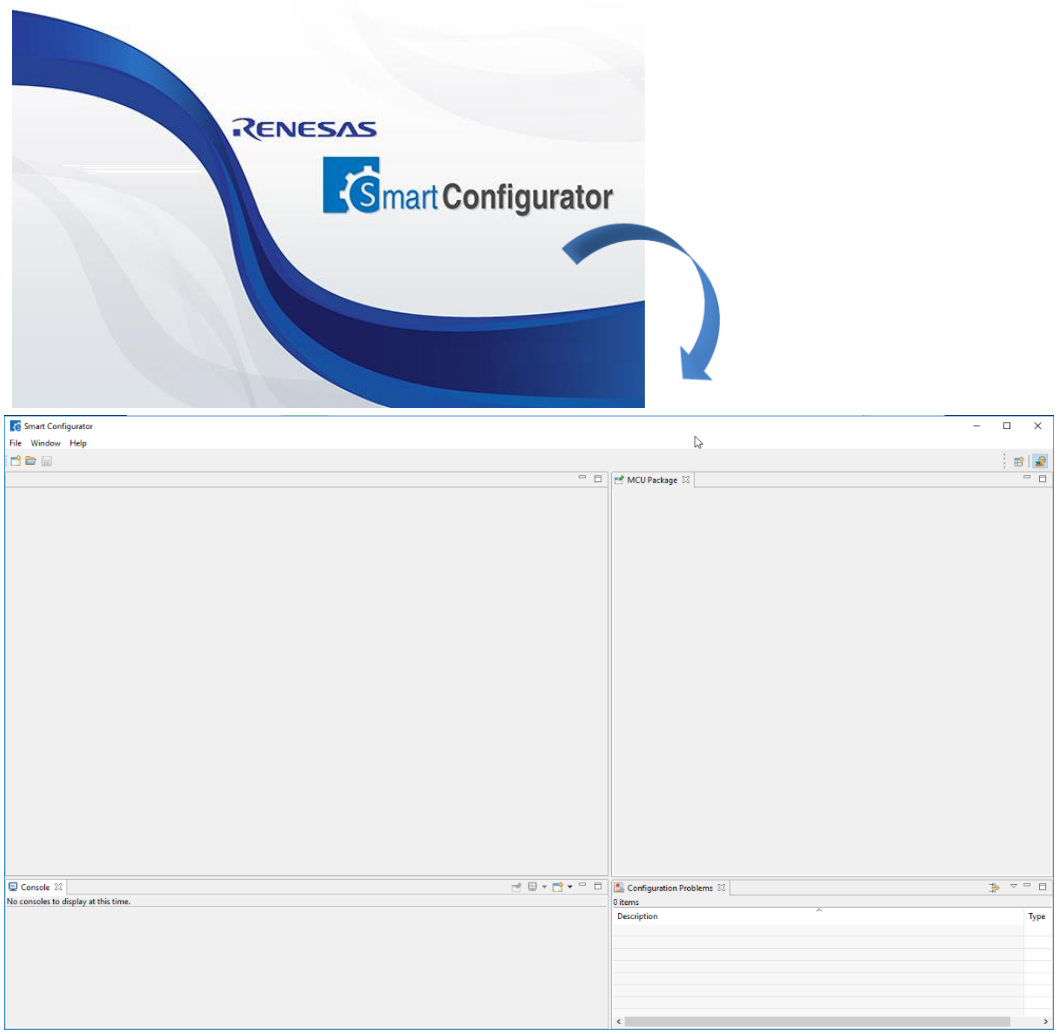



**Figure 3-2 Starting of Smart Configurator**

## 3.3 Creating and Loading a Configuration File

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

### 3.3.1 Creating a New Smart Configurator Configuration File

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

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

(2) In [Toolchain:], select [IAR RL78 Toolchain].

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

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

Note: The *.eww, *.ewp, *.ewd, main.c and buildinfo.ipcf files will be generated to this location after clicking "Generate Code" button.

(5) Click [Finish] to create the configuration file.

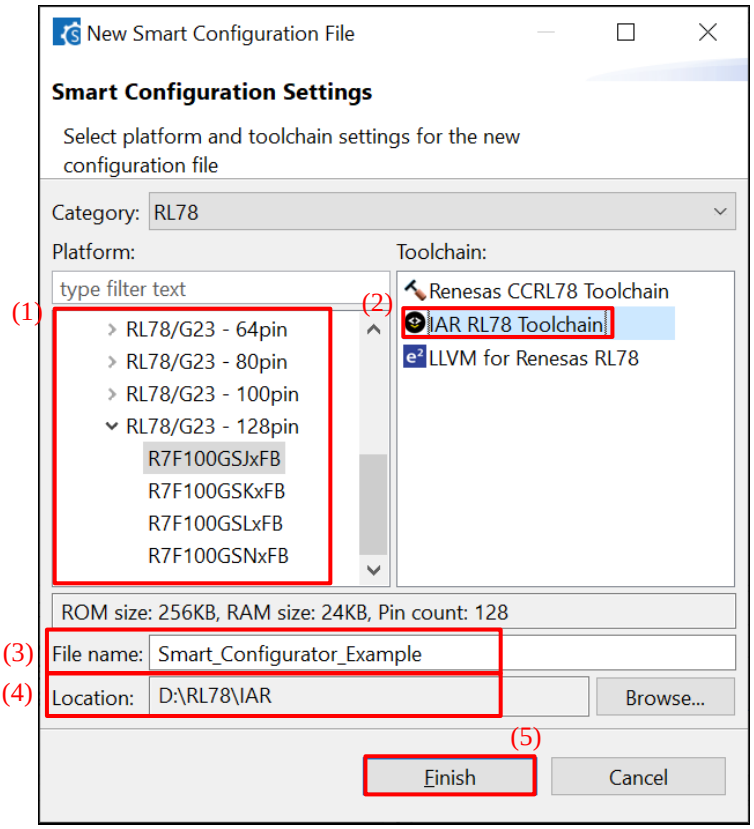



**Figure 3-3 Create a Configuration File**

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

Note: The *.eww, *.ewp, *.ewd and main.c files will be generated only for the first-time code generation, while the buildinfo.ipcf file will be generated always for each time code generation.

### 3.3.2 Opening an Existing Configuration File

On the main window, click the [Opening an Existing Configuration File] button to display the [Open] dialog box. Select the (*. scfg) file and click [Open].

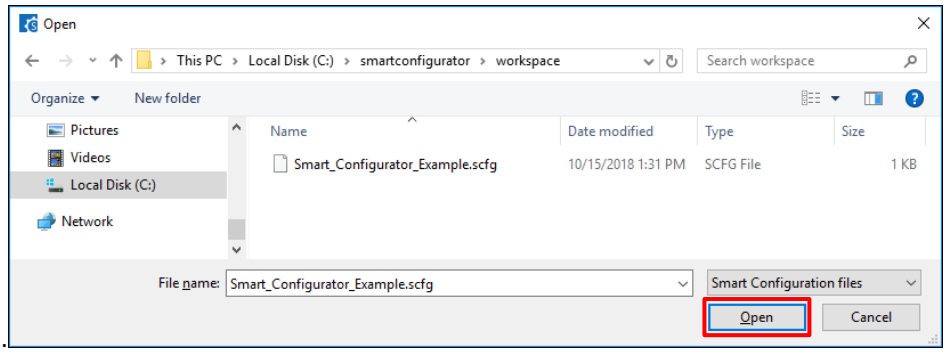



**Figure 3-4 Opening an Existing Configuration File**

## 3.4 Window

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

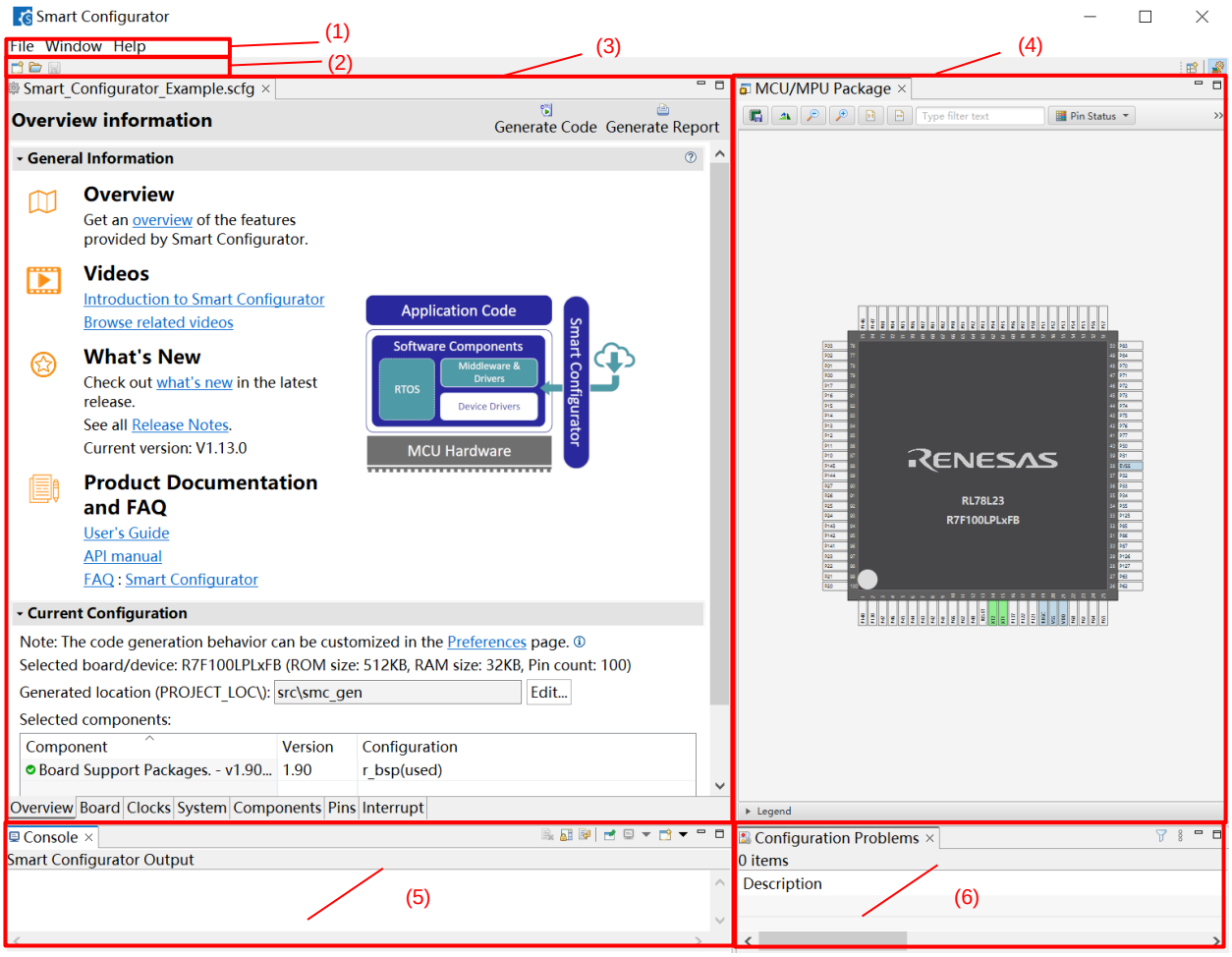



**Figure 3-5 Main Window**

(1) Menu bar

(2) Main toolbar

(3) Smart Configurator view

(4) MCU/MPU Package view

(5) Console view

(6) Configuration Problems view

### 3.4.1 Main Menu

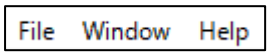
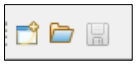



Table 3-1, Main Menu Items, lists the items of the main menu.

**Table 3-1. Main Menu Items**

| Menu | | Details |
|---|---|---|
| File | New | The dialog box [New Smart Configurator File], which is used to create a new project, is displayed. |
| | Open | The dialog box [Open], which opens an existing project, is displayed. |
| | Save | Saves a project with the same name. |
| | Restart | Smart Configurator is restarted. |
| | Exit | Execution of the Smart Configurator is terminated. |
| Window | Preference | The dialog box [Preference], which is used to specify the properties of the project, is displayed. |
| | Show View | The dialog box [Show view], which is used to set the view of the window, is displayed. |
| Help | Help Contents | The help menu is displayed. |
| | Home Page | Open the home page of the Smart Configurator on the Renesas Electronics website. |
| | Release Notes | Open the release note of the Smart Configurator on the Renesas Electronics website. |
| | Tool News | Open the tool news of the Smart Configurator on the Renesas Electronics website. |
| | API Manual | Open the API manual of the Smart Configurator on the Renesas Electronics website. |
| | About | The version information is displayed. |

### 3.4.2 Toolbar



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

**Table 3-2. Toolbar Buttons and Related Menu Items**

| Toolbar button | Related menu item |
|---|---|
|  | [File]→[New] |
|  | [File]→[Open] |
|  | [File]→[Save] |

### 3.4.3 Smart Configurator View

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

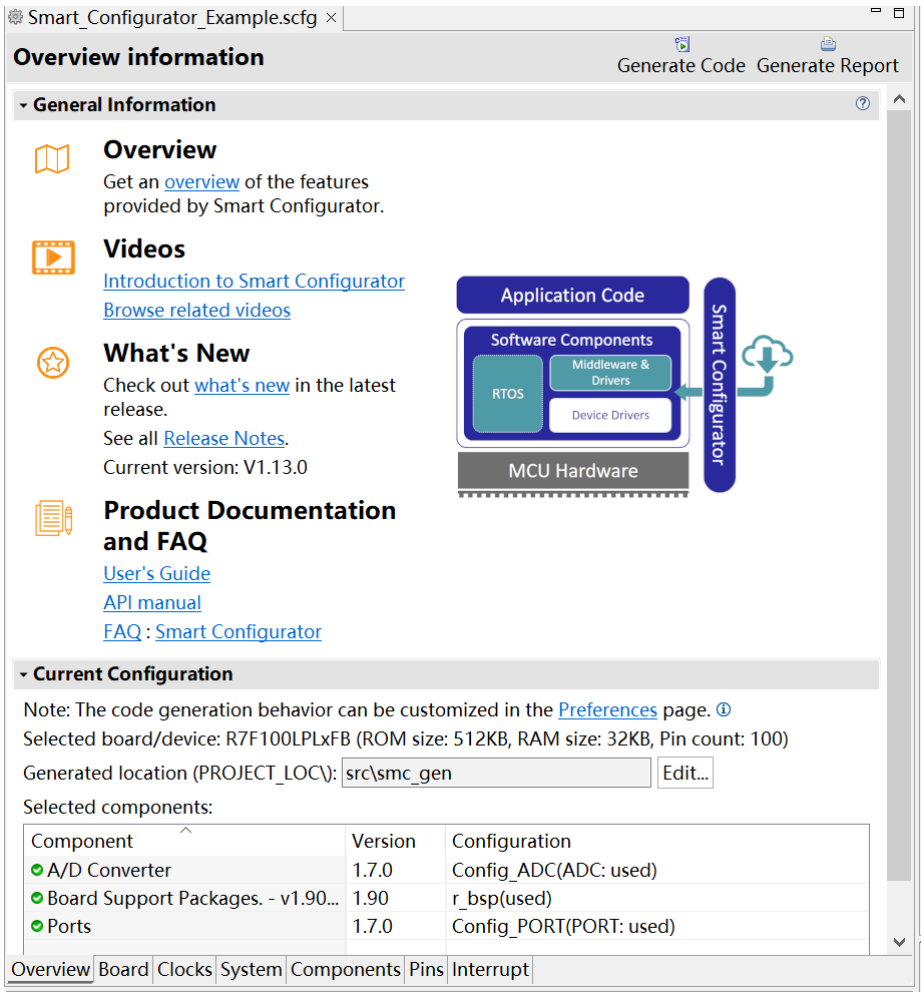



**Figure 3-6 Smart Configurator View**

### 3.4.4 MCU/MPU Package View

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

Three types of package view can be switched 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 Setting" 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.

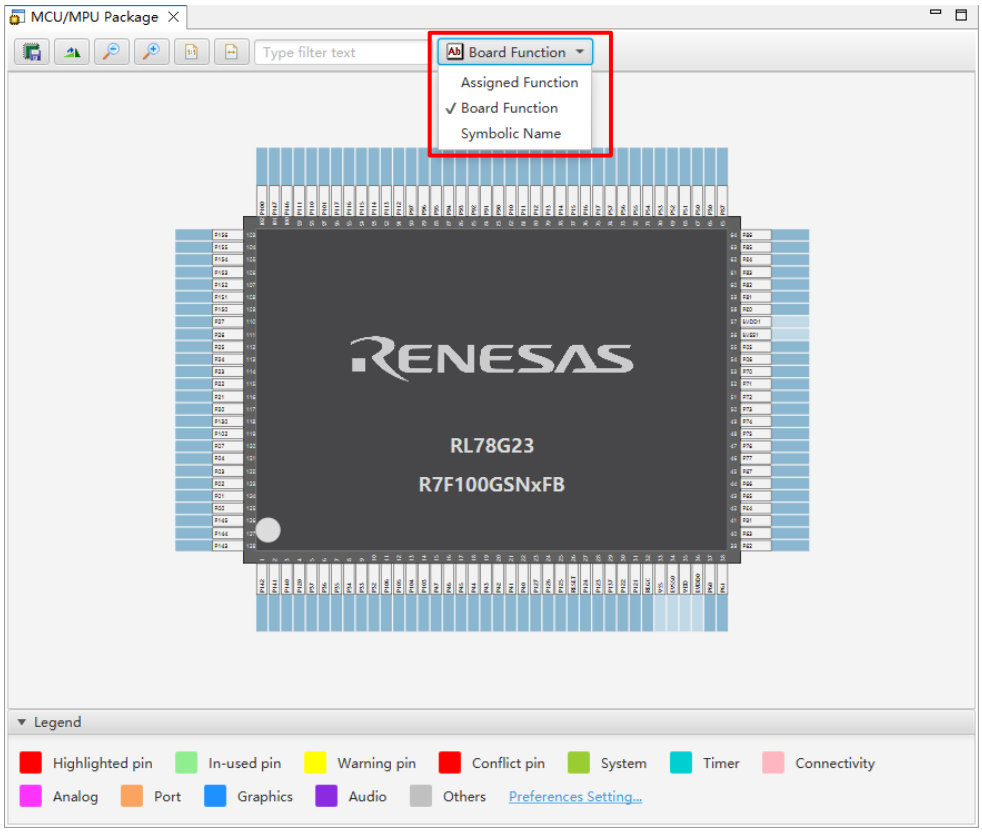



**Figure 3-7 MCU/MPU Package View**

### 3.4.5 Console View

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


**Figure 3-8 Console View**

### 3.4.6 Configuration Problems View

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

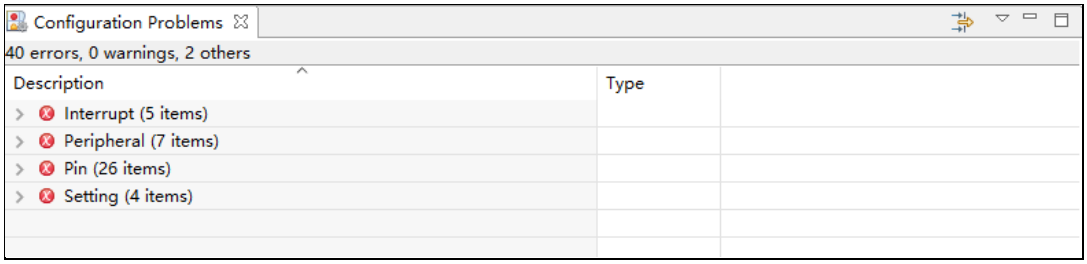



**Figure 3-9 Configuration Problems View**

# 4. Setting of Peripheral Modules

User can select peripheral modules from the Smart Configurator view.

## 4.1 Board Setting

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.

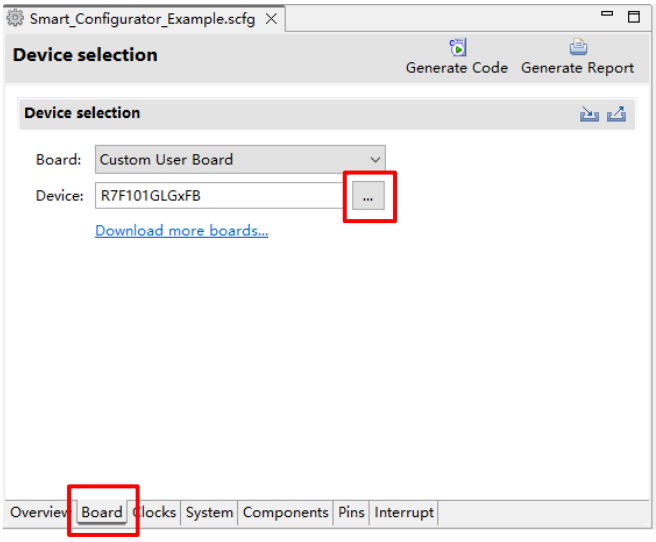



**Figure 4-1 Selecting the Device**

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

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

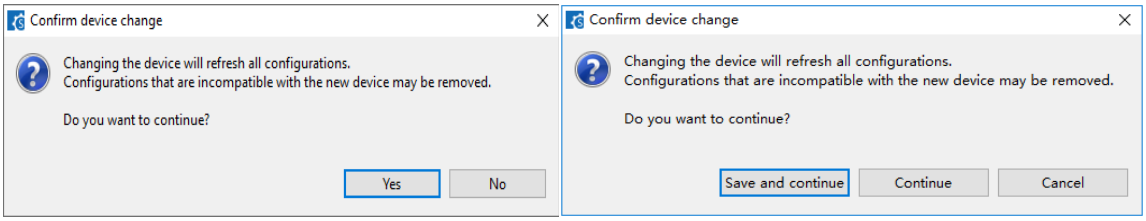



**Figure 4-2 Confirm Device Change**

**Table 4-1. Device Change Confirmation Operation List**

| Button | Operation explanation |
|---|---|
| Yes | Change to the selected device. |
| No | It does not change the device. |
| Save and continue(Note*1) | After saving the current configuration contents to the configuration file, change to the selected device. |
| Continue(Note*1) | Changes to the selected device without saving the current configuration contents to the configuration file. |
| Cancel(Note*1) | It does not change the device. |

Note *1: Smart Configurator view is marked with dirty *.

### 4.1.2 Selecting the Board

Click on the [ ] to select a board from the list. After board selection, the pins, clock and system setting will be automatically configured according to board connection.

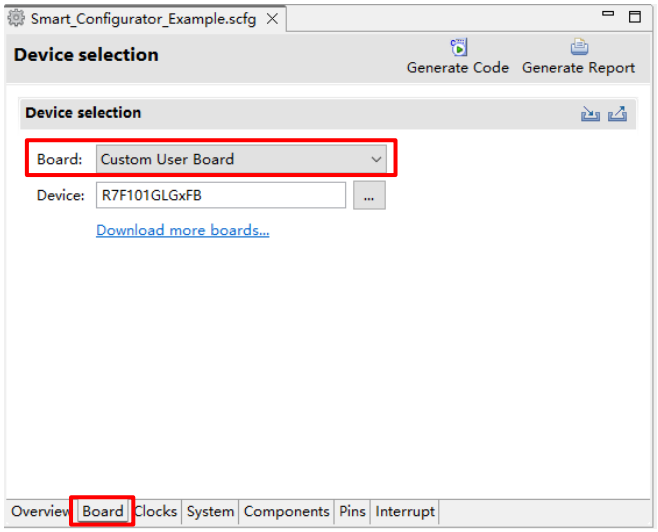



**Figure 4-3 Selecting the Board**

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

- 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

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

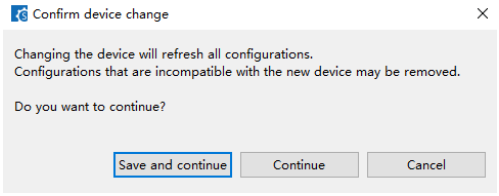



**Figure 4-4 Confirm Board Change**

**Table 4-2. Board Change Confirmation Operation List**

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

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

### 4.1.3 Exporting Board Settings

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

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

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

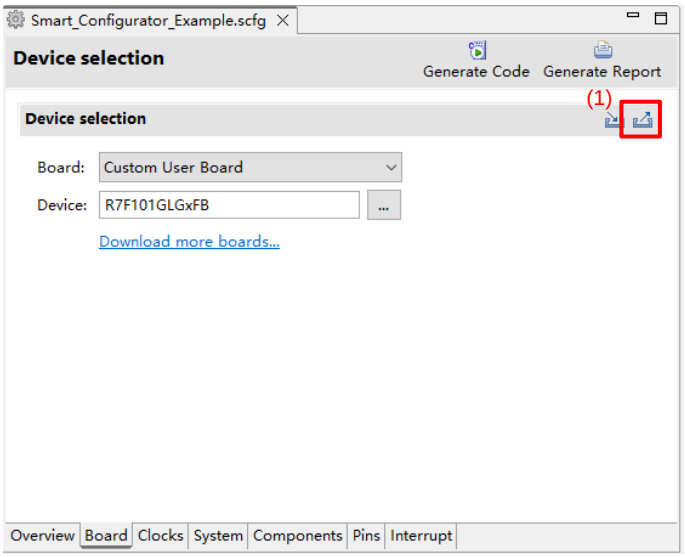



**Figure 4-5 Exporting Board Settings (bdf Format)**

### 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.

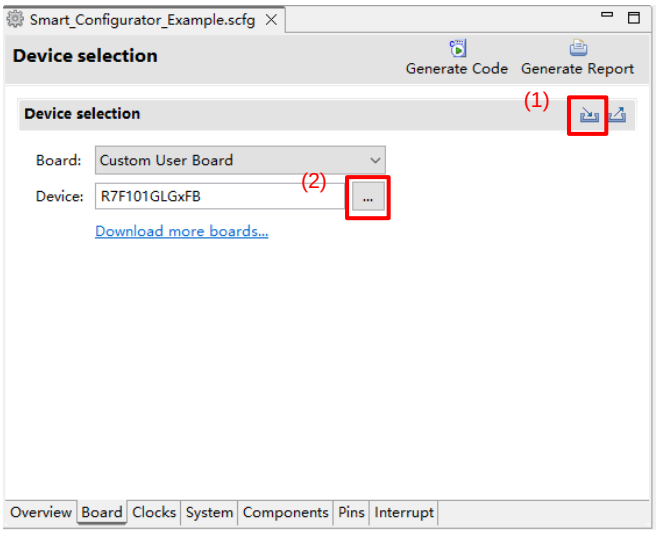



**Figure 4-6 Importing Board Settings (bdf Format)**

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.

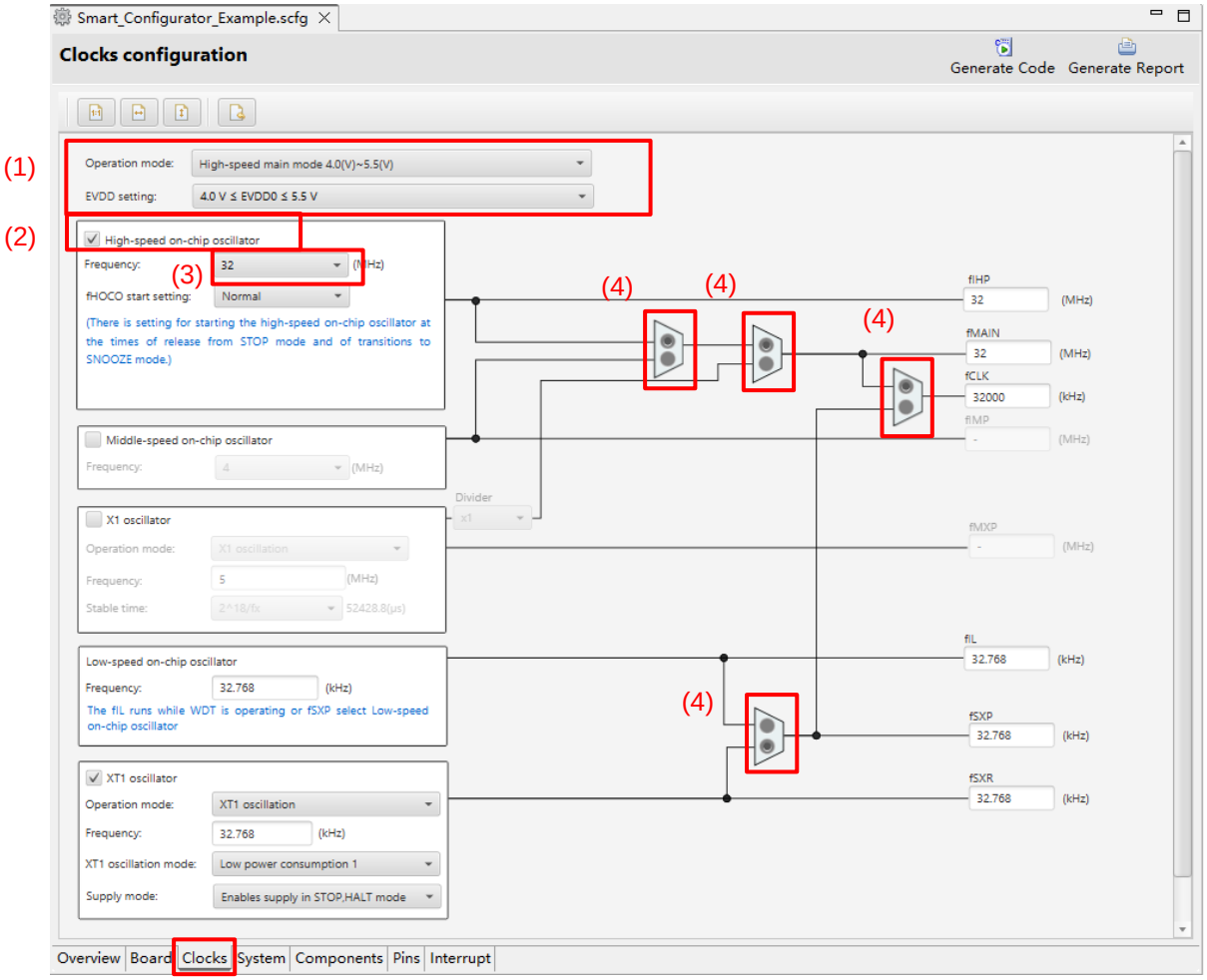



**Figure 4-7 [Clocks] Page**

## 4.3 System Settings

User can set the on-chip debug setting on the [System] page. This setting is reflected in into r_bsp file.

For example, make desired setting as in below figure for illustration, after that click on [ Generate Code Generate Code] button as in step (4).

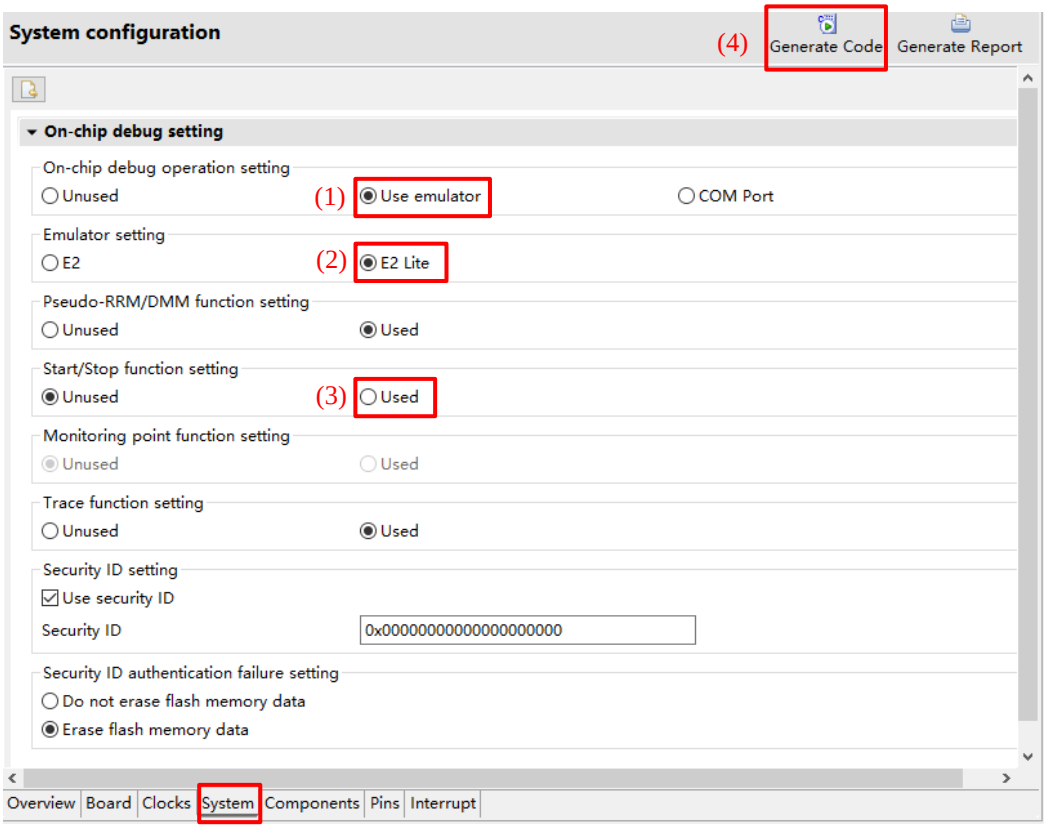



**Figure 4-8 Smart Configurator [System] Page Setting**

The on-chip debug setting is reflected into r_bsp file in: \<ProjectDir>\src\smc_gen\r_config\r_bsp_config.h file. To change the on-chip debug setting code value, please change desired setting and generate code again.

```
#if defined(__ICCRL78__)
/* Option byte setting(When using IAR) */
#define BSP_CFG_OPTBYTE0_VALUE (0xEFU) /* Generated value. Do not edit this manually */
#define BSP_CFG_OPTBYTE1_VALUE (0x3AU) /* Generated value. Do not edit this manually */
#define BSP_CFG_OPTBYTE2_VALUE (0x6CU) /* Generated value. Do not edit this manually */
#define BSP_CFG_OPTBYTE3_VALUE (0x84U) /* Generated value. Do not edit this manually */
```

**Figure 4-9 On-Chip Debug Setting Code Generation**

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

## 4.4 Software 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.

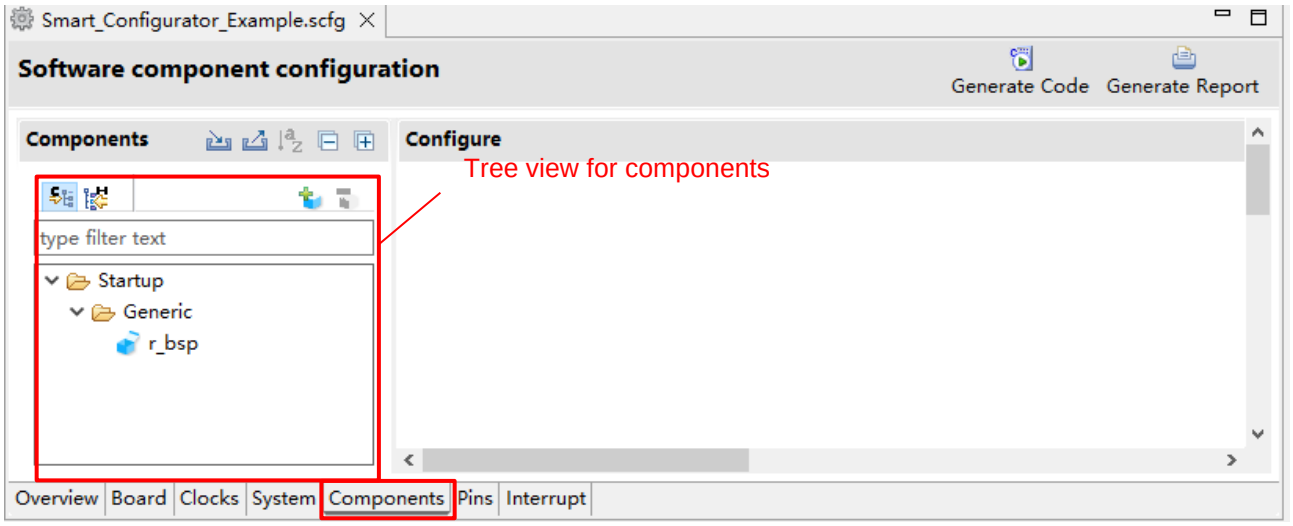



**Figure 4-10 [Components] Page**

### 4.4.1 Switching Between the Component View and Hardware View

The Smart Configurator provides two tree view: Component View and Hardware View. Please 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.

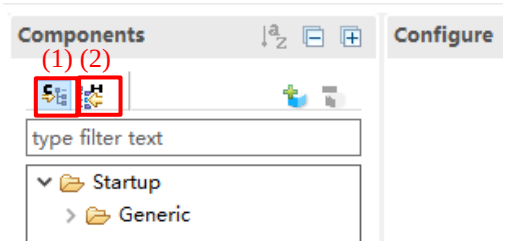



**Figure 4-11 Switching to the Hardware View**

### 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.

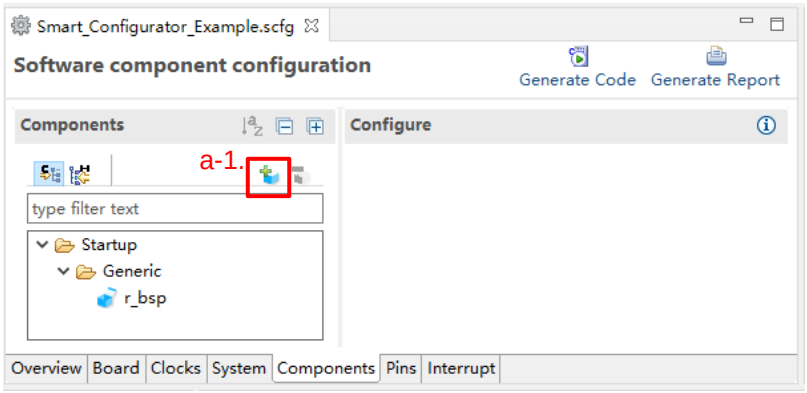



**Figure 4-12 Adding a Component**

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].

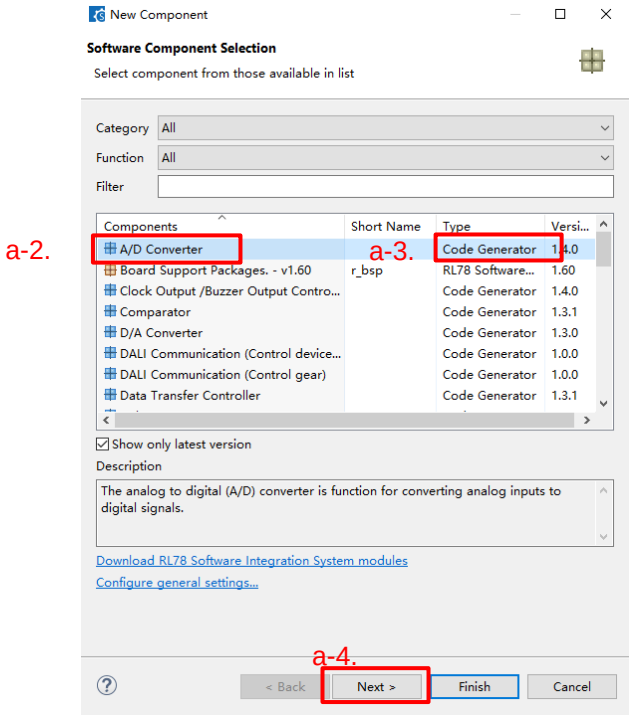



**Figure 4-13 Adding a Code Generator Component**

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].

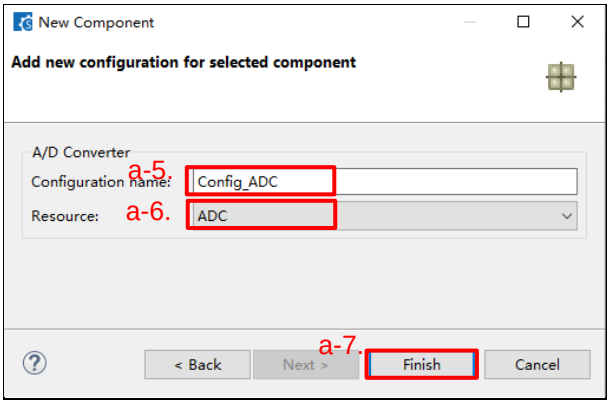



**Figure 4-14 Adding a Component**

To add a component on Hardware Tree directly, please 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 as above "adding a component by clicking adding icon" step a-3 to a-7.

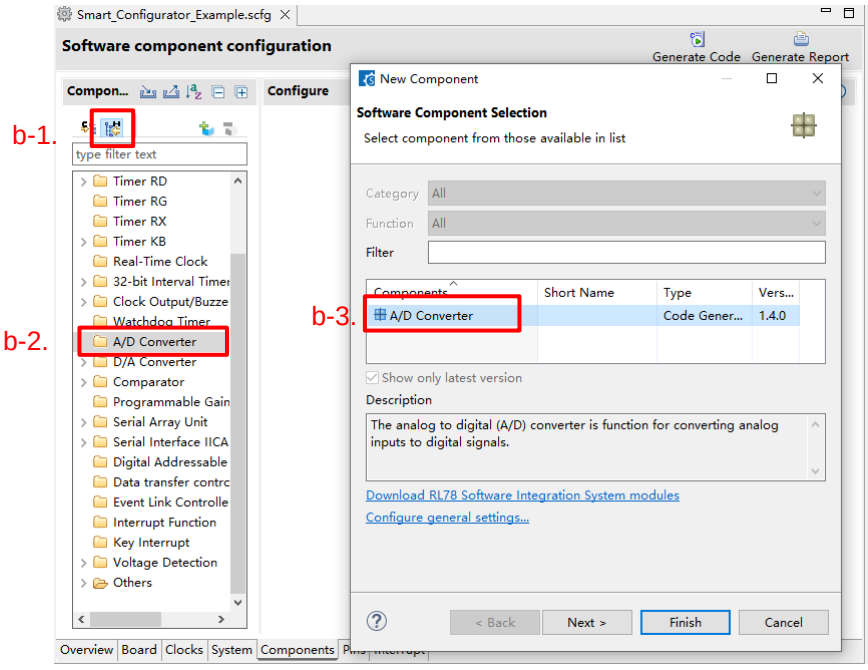



**Figure 4-15 Adding a Code Generator Component to the Hardware View**

### 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.

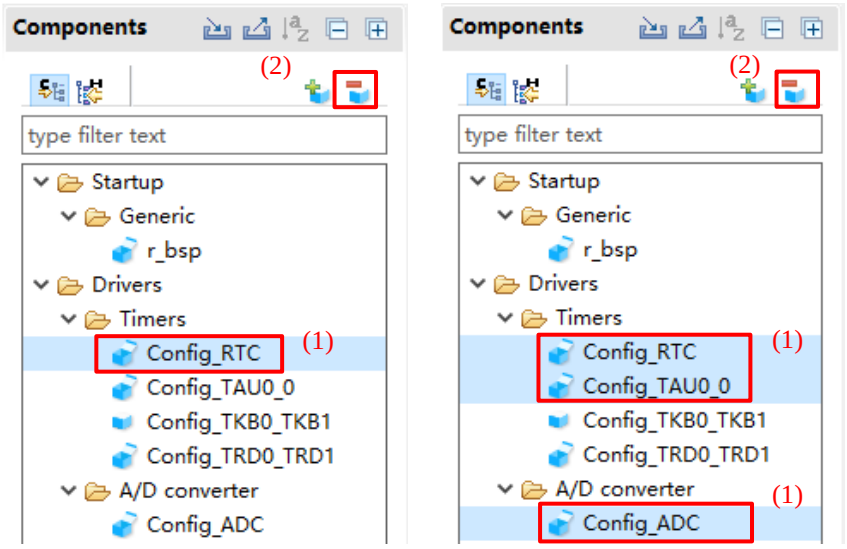



**Figure 4-16 Removing a Software Component or Multiple Components**

The selected software components will be removed from the Components tree.
To delete the source files previously generated for the removed components from the IAR Embedded Workbench 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 figures 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].

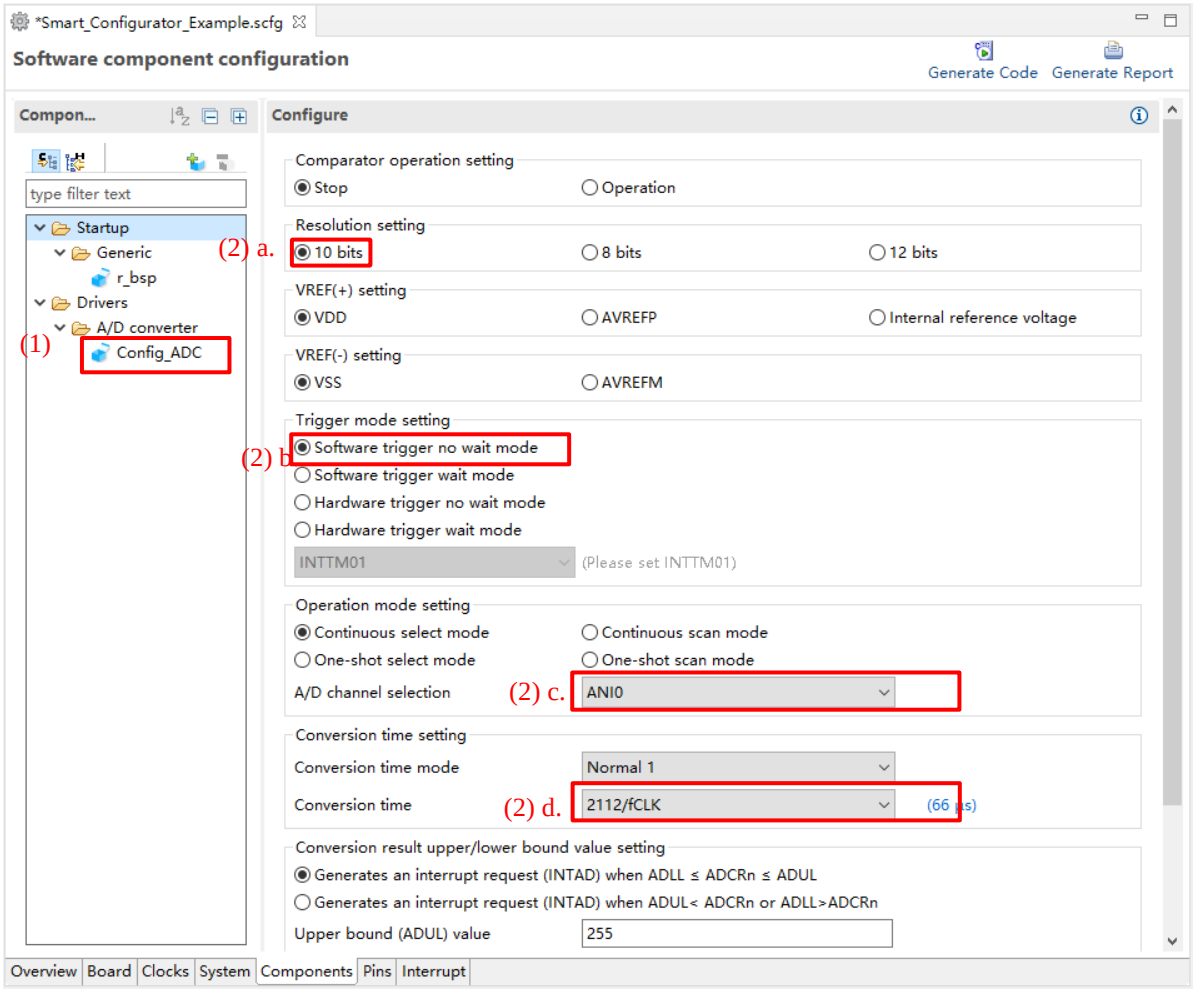



**Figure 4-17 Setting of a Code Generator Driver**

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.

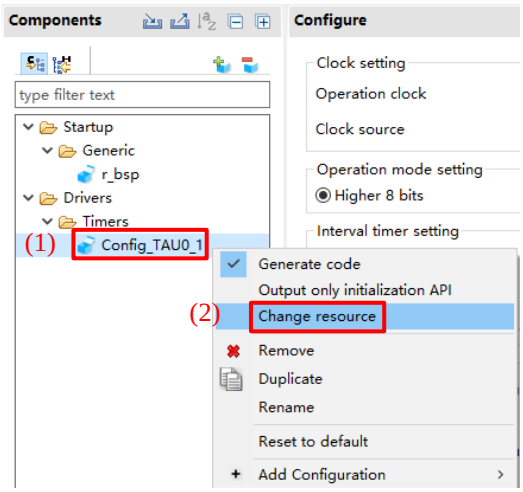



**Figure 4-18 Changing the Resource**

(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.

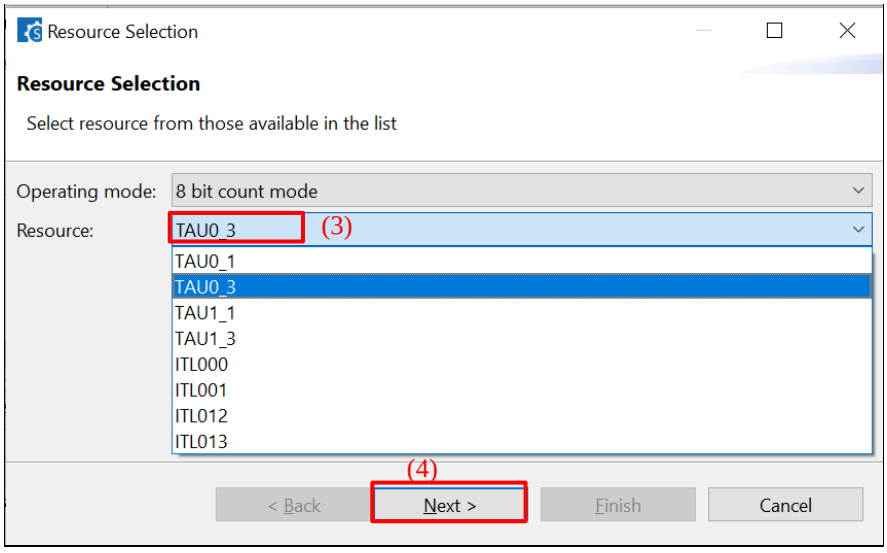



**Figure 4-19 Components Page – Selecting a New Resource**

(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].

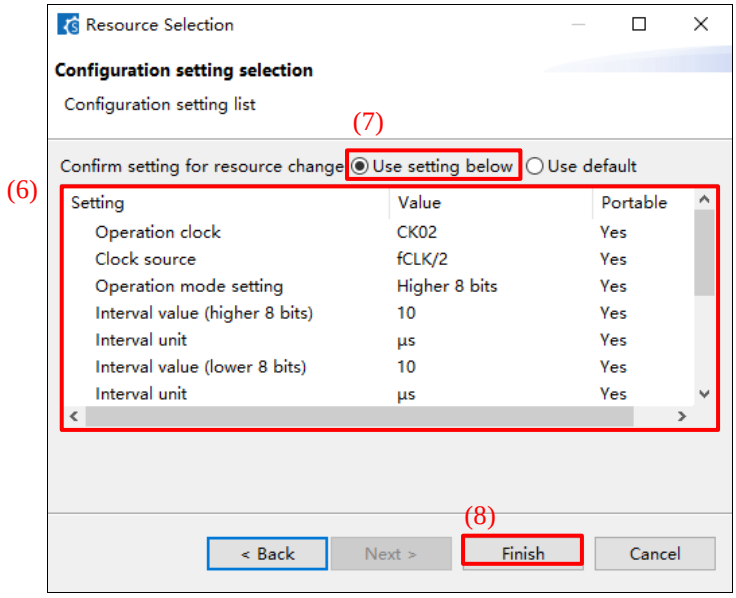



**Figure 4-20 Checking the Settings of the New Resource**

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

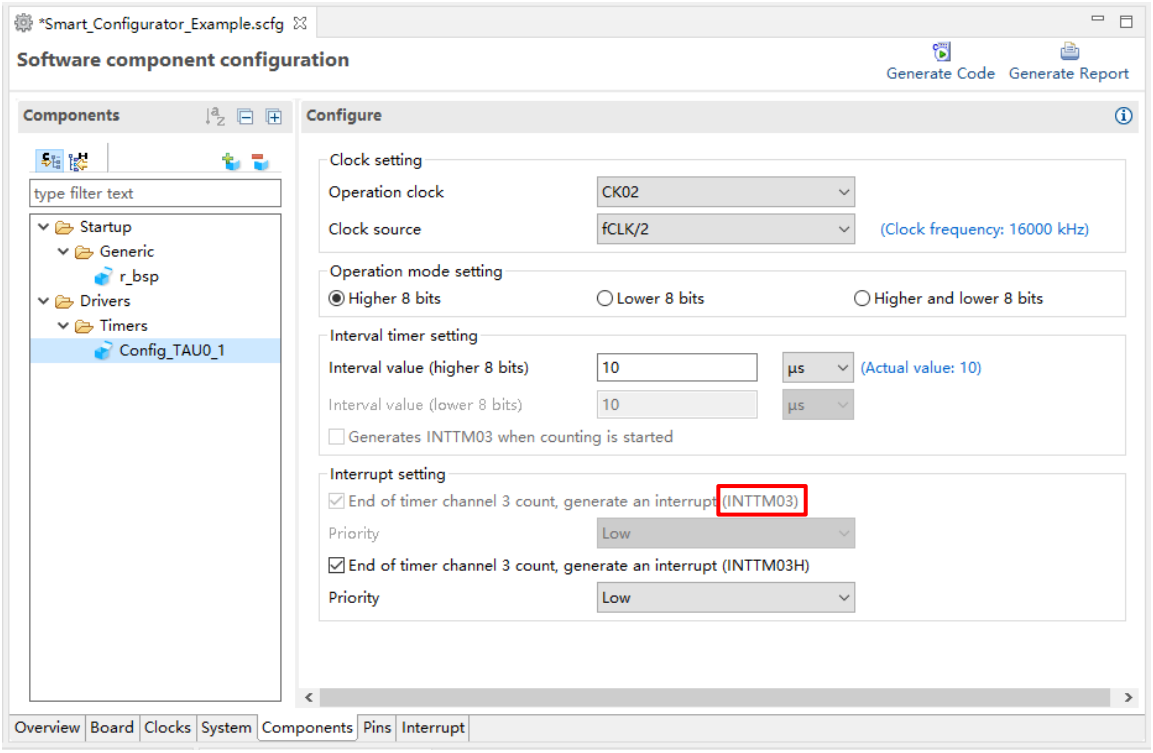



**Figure 4-21 Resource Changed Automatically**

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).

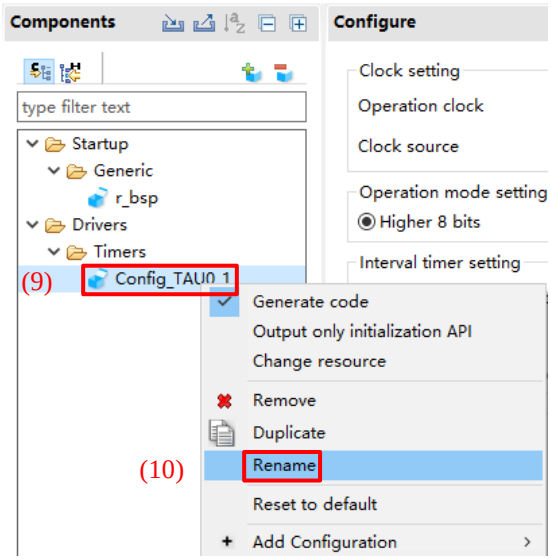



**Figure 4-22 Renaming the Configuration**

### 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.

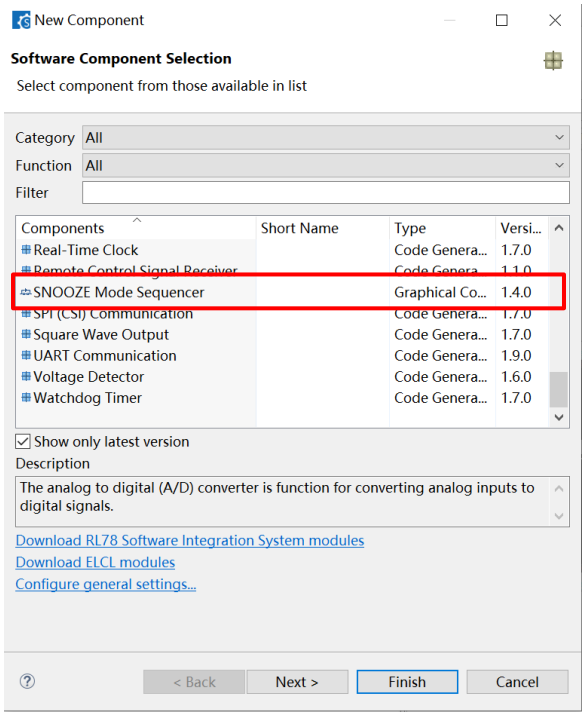



**Figure 4-23 Add SNOOZE Mode Sequencer**

A GUI of Graphical Configurator is displayed in below SMS figure, it is more graphically compared with Code Generator. User can Drag and Drop and configure the block which user want to use.

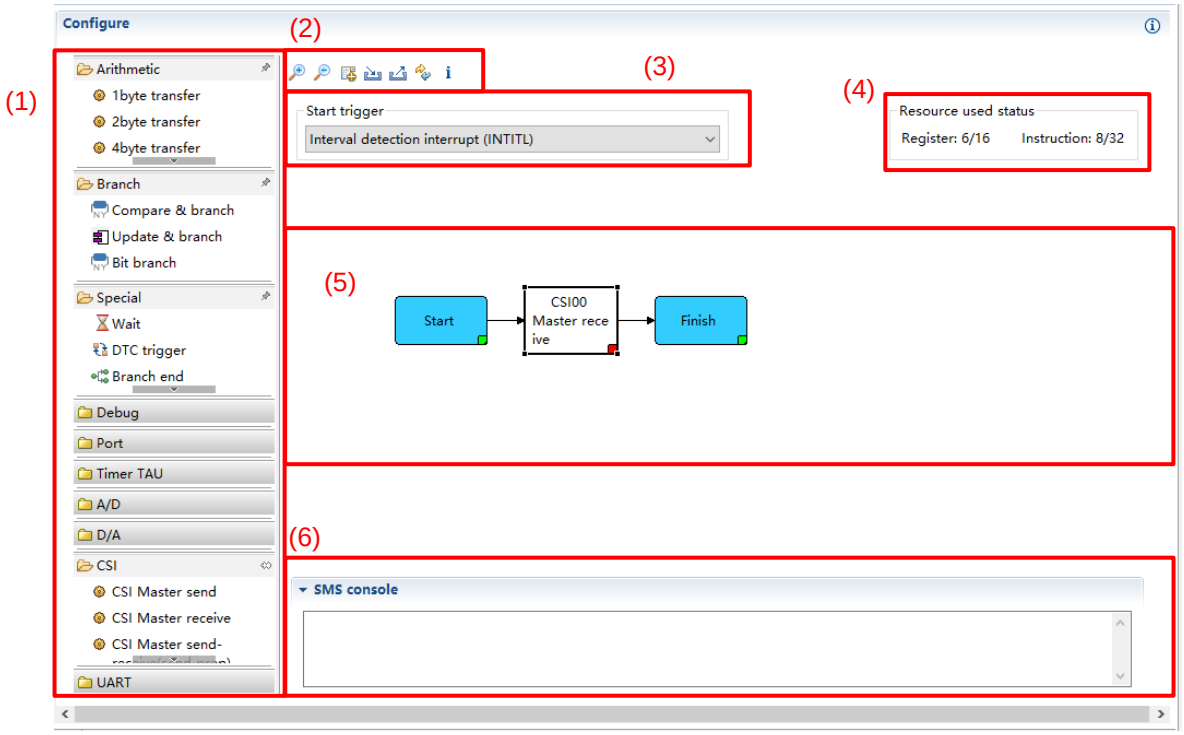



**Figure 4-24 SNOOZE Mode Sequencer (SMS) GUI**

**Table 4-3. SMS GUI area description**

| Area | | Description |
|---|---|---|
| (1) Block elements | | View the available blocks for SMS.<br>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) Configure the block by double click to pop the "CSI Master receive setting" property setting dialog.

(4) Specify the setting in the "CSI Master receive setting" property dialog.

(5) Open "Data Management" setting, edit the receive data.

(6) When configuring correctly, the color of bottom right corner will change from red to green.

(7) Add some blocks, drag and drop to adjust the sequence.

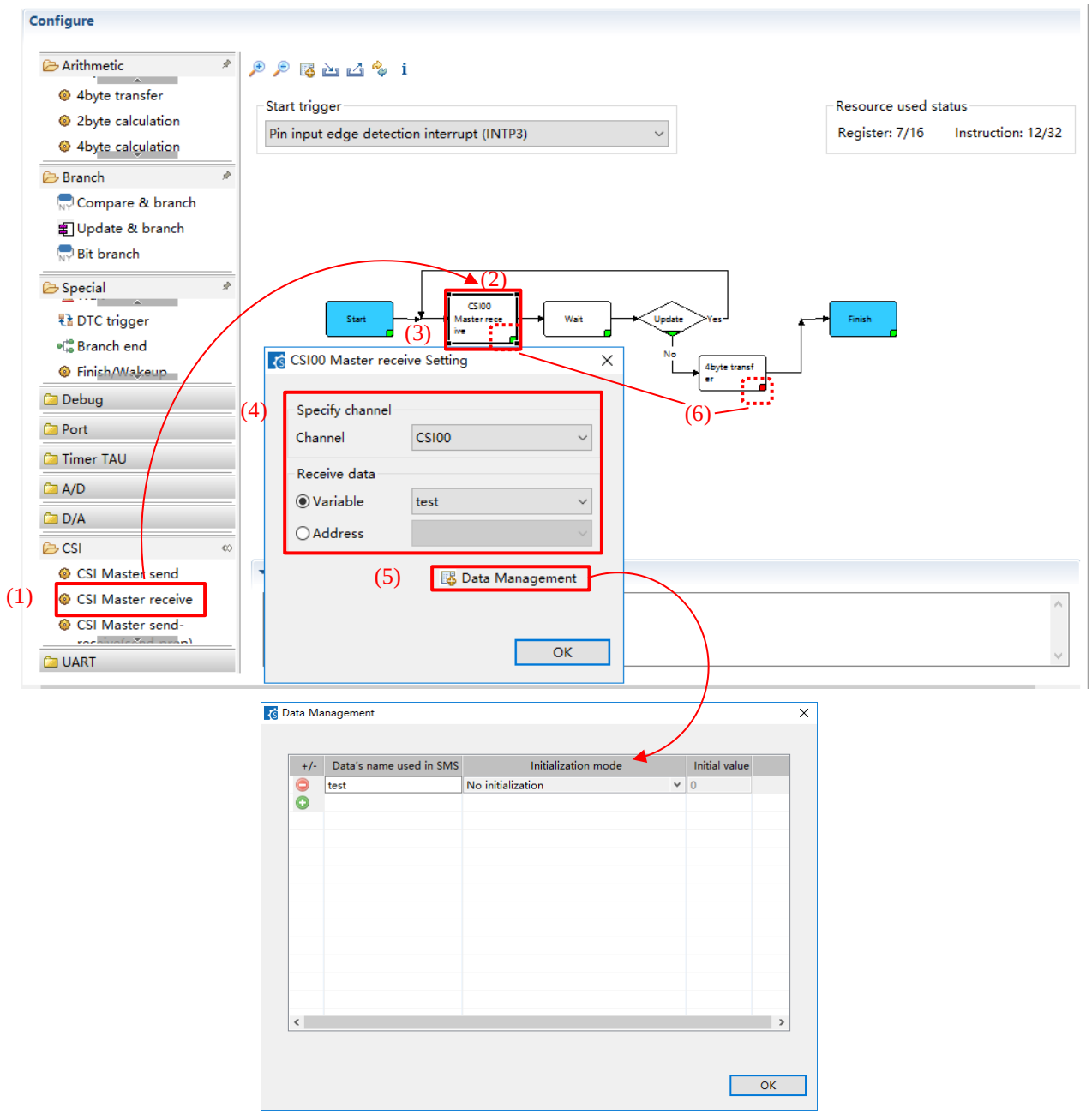



**Figure 4-25 SMS Block Configure**

### 4.4.7 Update SMS Data Files

Follow the procedure below to update SMS data file (Block, Sequence) to the latest version. Please 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.

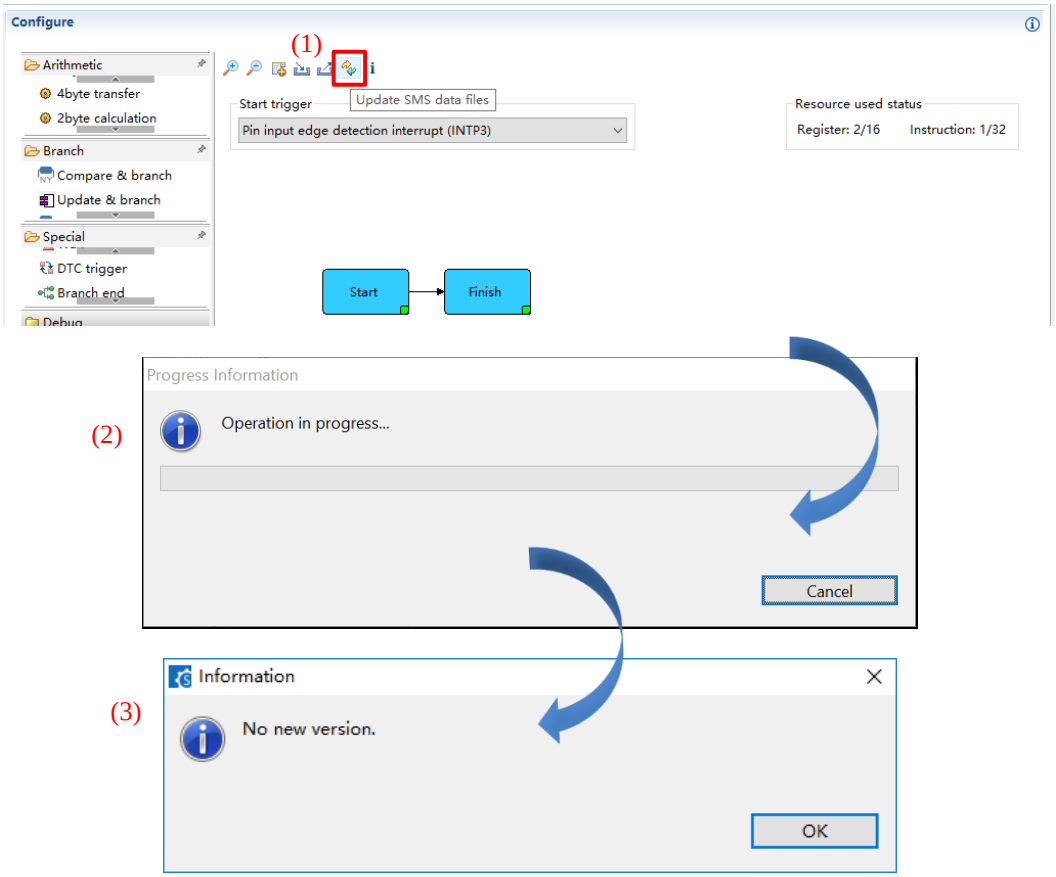



**Figure 4-26 SMS Data File Download**

### 4.4.8 ELCL Fixed Function Modules Download

The Software Component type for ELCL (Logic Event Link Controller) is Graphical Configurator. ELCL component have 2 types, 1 type is fixed function ELCL component such as "Slave Select Pin Function", "Chattering Prevention Function" and so on, the other one is ELCL flexible circuit, user can use it to create flexible ELCL circuit.

ELCL fixed function modules can be added from component list in New Component dialog. If user wants to use other ELCL fixed function modules which are not included in Component list, the user can click on [Download ELCL modules] link in New Component dialog to check and download more ELCL fixed function modules:

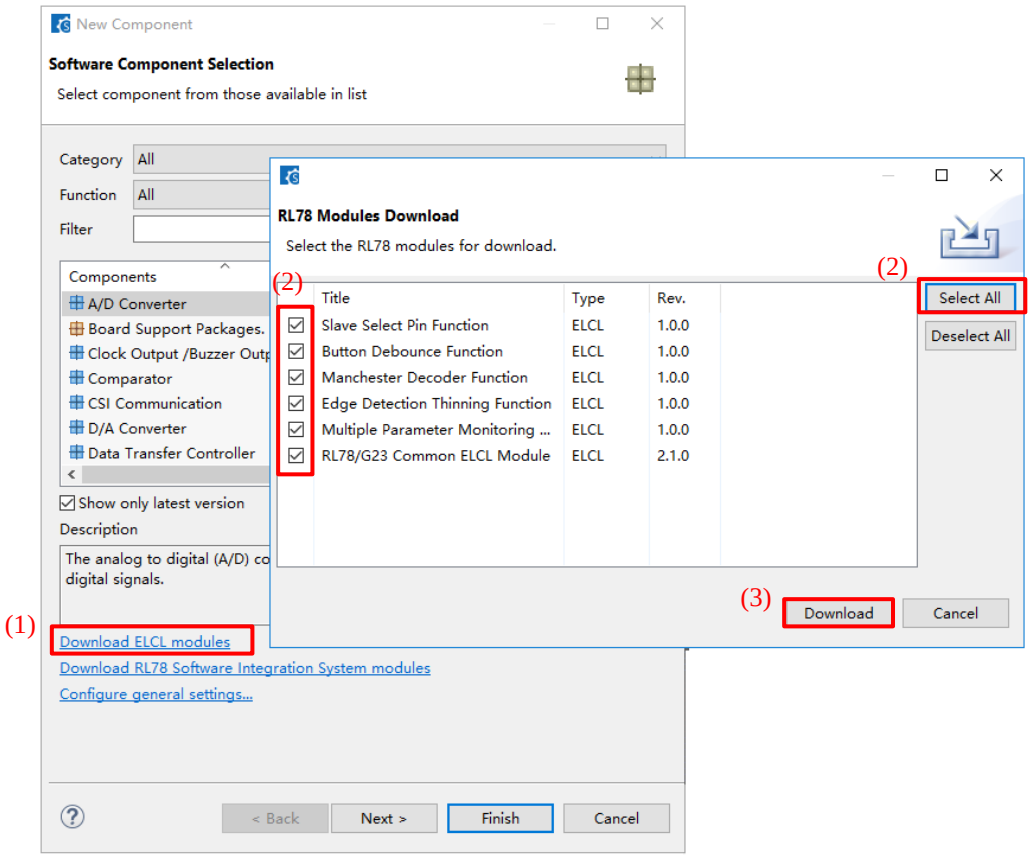



**Figure 4-27 ELCL Fixed Function Modules Download**

After downloading, all ELCL fixed function modules are auto added to component list:

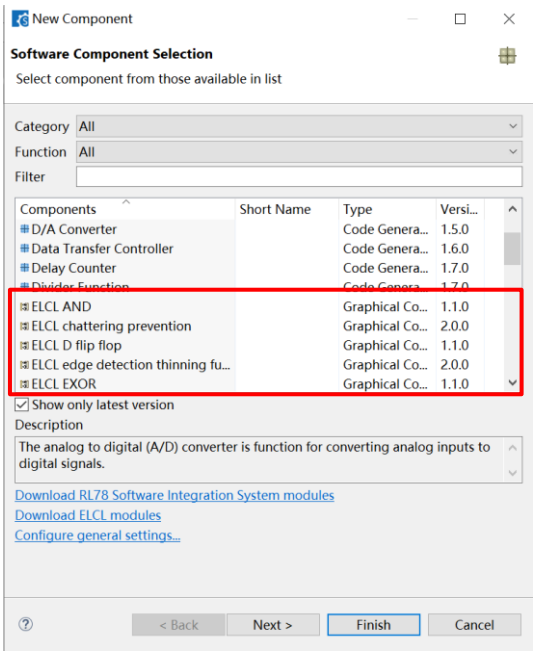



**Figure 4-28 Add ELCL Fixed Function Module**

### 4.4.9 Setting a Fixed function ELCL Component

Follow the procedure below to set up a fixed function ELCL component.

(1) Select a fixed function ELCL component 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 the user wants more details about current fixed function ELCL component usage, the user can click the [ELCL_slave_select_pition.pdf] link to open the application notes for check.

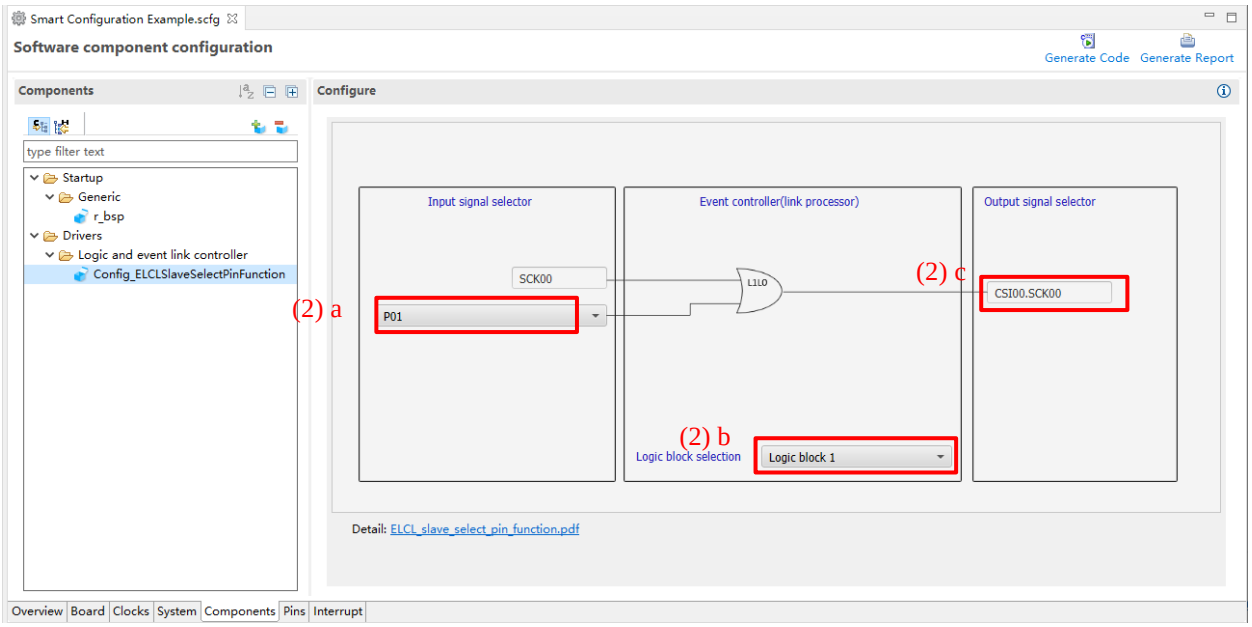



**Figure 4-29 Configure a Fixed Function ELCL Component**

### 4.4.10 Create and Edit ELCL Flexible Circuit

ELCL (Logic and Event Link Controller) flexible circuit component is a new component type as "Graphical Configurator", it is list and can be selected to use directly in default component list.

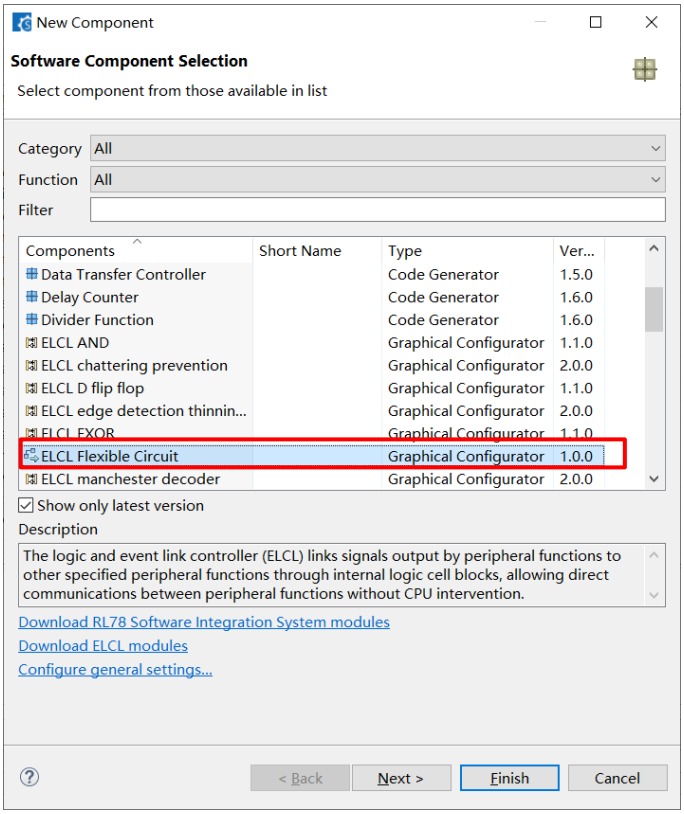



**Figure 4-30 Add "ELCL Flexible Circuit" Component**

ELCL flexible circuit component provide intuitive GUI supporting drag & drop operation for ELCL circuit creation and editing, after user finished circuit design, the ELCL registers setting can be generated automatically.

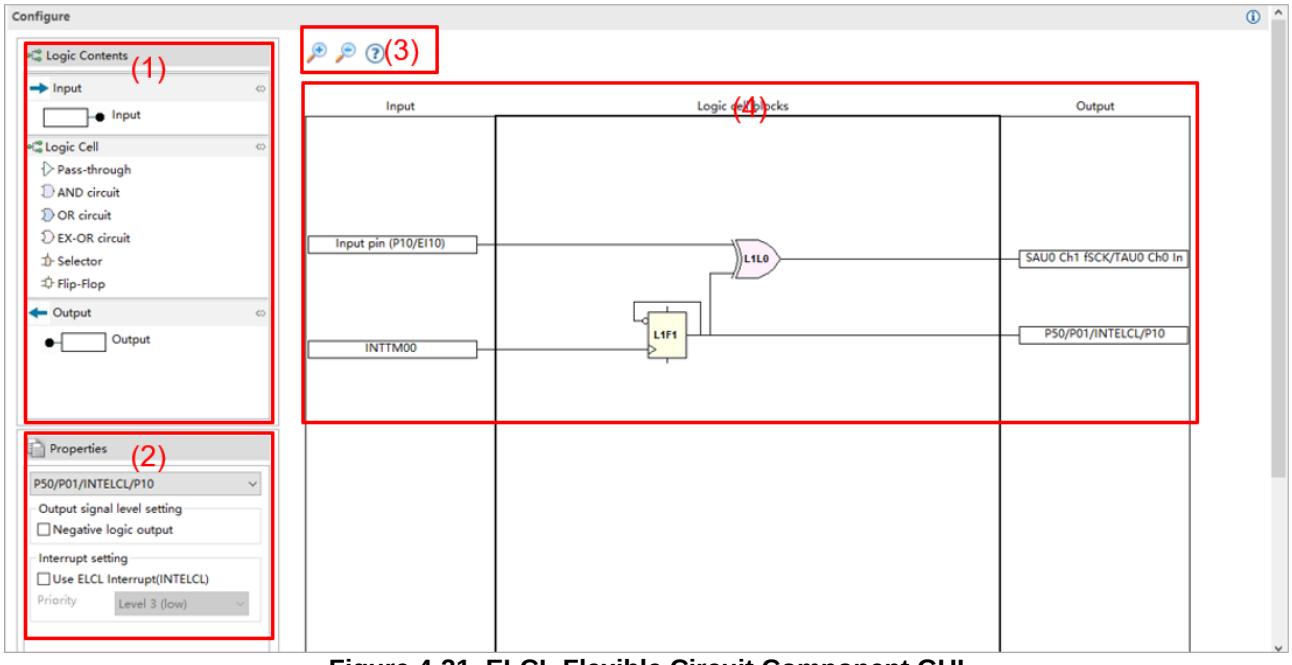



**Figure 4-31 ELCL Flexible Circuit Component GUI**

**Table 4-4 ELCL flexible circuit GUI area description**

| Area | | Description |
|---|---|---|
| (1) ELCL elements | | View the available elements for ELCL. |
| (2) Property | | Displays the setting of selected ELCL element. |
| (3) Toolbar |  | Zoom in. |
| |  | Zoom out. |
| |  | Show help. |
| (4) Canvas area | Input | The area is ELCL input elements are placed. |
| | Logic cell blocks | The area is ELCL logic cell elements are placed. |
| | Output | The area is ELCL output elements are placed. |

Follow the procedure below to create ELCL flexible circuit:

(1) Drag and drop the ELCL elements to canvas. The elements include input, logic cell and output.

(2) Select element and set the property for each dropped element.

(3) Make line connection by drag & drop from start point to end point.

(4) After ELCL circuit created, click "Generate Code" button will generate ELCL registers setting automatically.

Note: Procedure (1)(2)(3) is not fixed sequence steps, user can freely do each step to create or edit ELCL circuit.

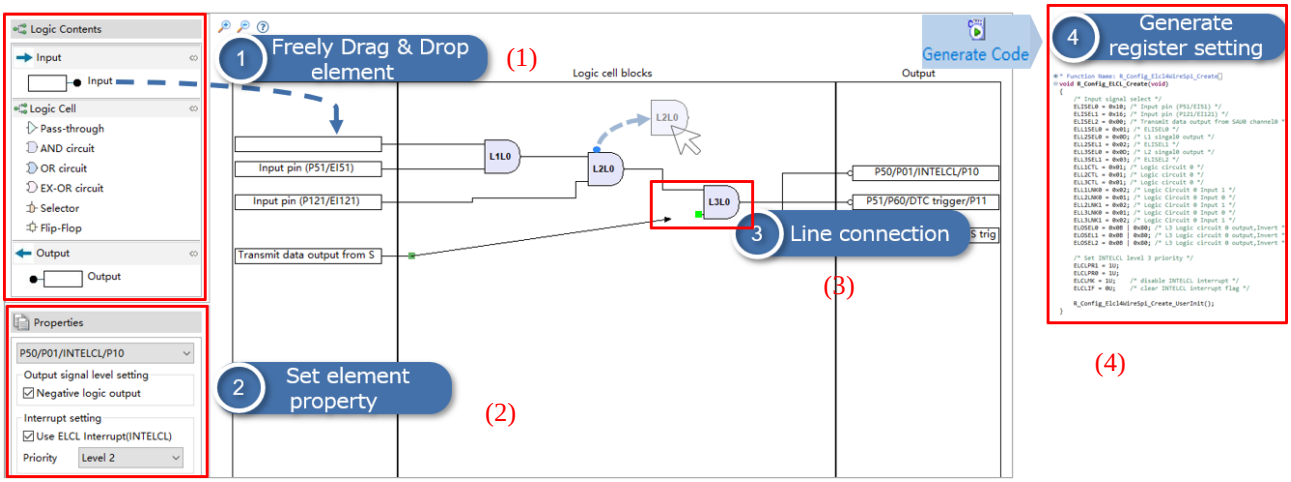



**Figure 4-32 ELCL Flexible Circuit Creation Procedure**

Below list some GUI operation details which help the user create ELCL circuit with ease and guide user towards correct designs.

(1) Connect the line through drag & drop the start point to end point.

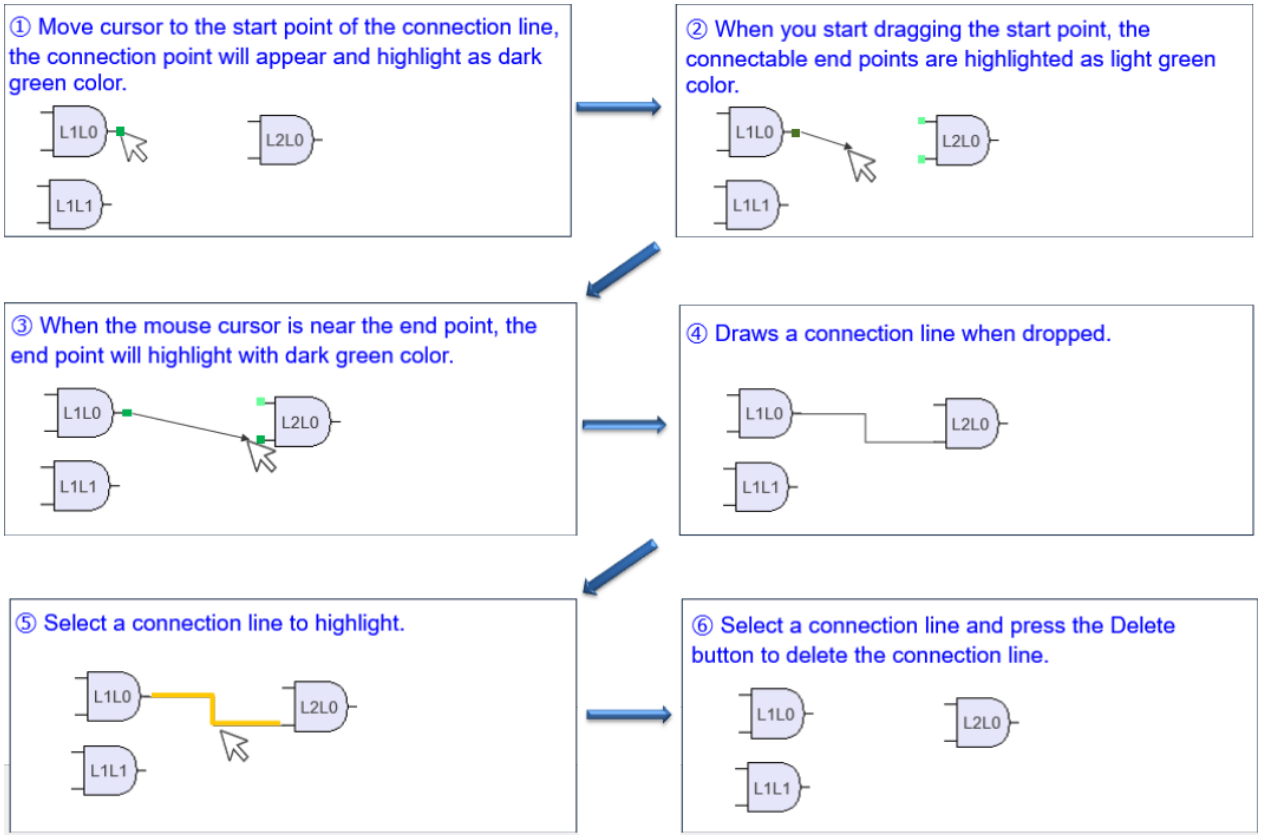



**Figure 4-33 Line Connection Procedure**

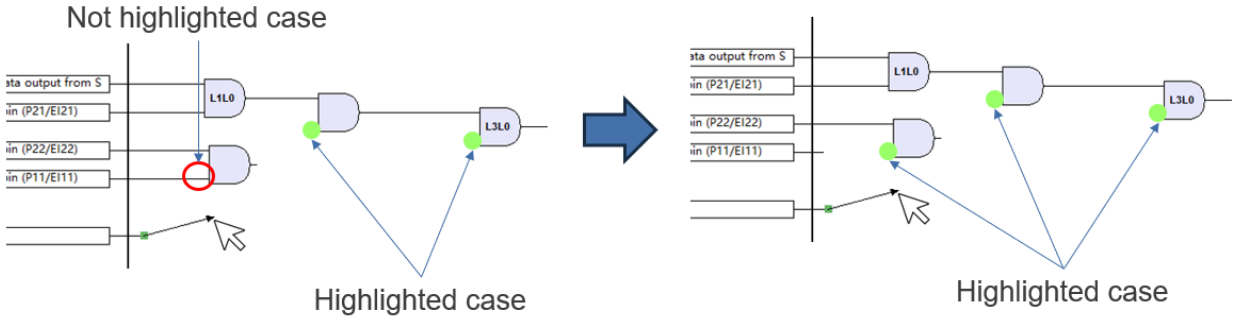



**Figure 4-34 Highlight/Not Highlight Case When Making the Line Connection**

Note: When you drag the start point, the connectable end points are highlighted as light green.

(2) When user make connection or settings out of hardware limitation, errors will be displayed in each parts. User can resolve the error base on the indicated message.

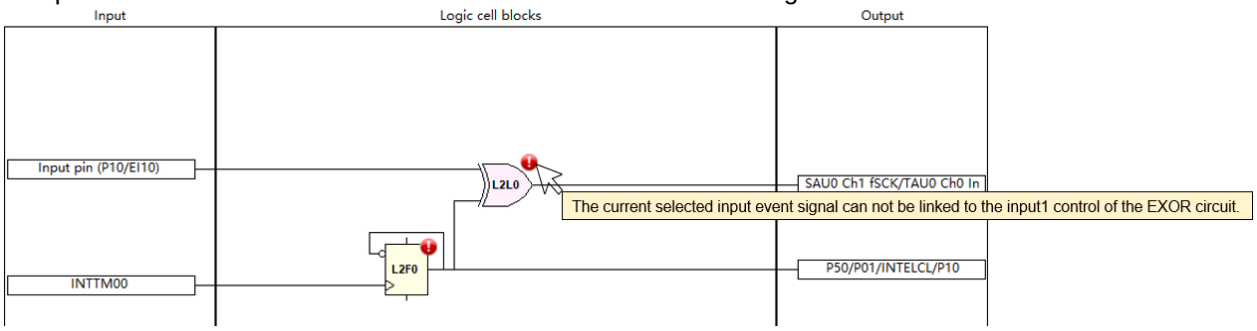



**Figure 4-35 ELCL Flexible Circuit GUI Error Indication**

**Table 4-1 ELCL flexible circuit GUI error message list**

| No. | Error message |
|---|---|
| 1 | The current selected input event signal can not be linked to the inputx control of the pass-through/AND/OR/EX-OR circuit. |
| 2 | Signal select register xxx are all occupied, no resources available for allocation. |
| 3 | Both ELL1SEL4 and ELL1SEL5 are occupied, if you want to configure this setting, please use the same signal source as another Flip-Flop in same logic cell block. |
| 4 | The selected event signal for clock is not consistent with another flip-flop, please use same event signal with another flip-flop. |
| 5 | Set and Reset of flip-flop must select different event signal to be linked. |
| 6 | The ELCL circuit exist unallocated resource or not finished line connection, so some correspondence code will not be generated or incorrect. |
| 7 | When connectING an interrupt request signal to an input signal, can connect only hardware triggers for event-receiving peripheral functions. Please connect a hardware trigger to output signal. |
| 8 | Logic cell block Lx celly (flip-flop n) is already occupied, please select other available resources. |

### 4.4.11 Downloading RL78 Software Integration System Modules

RL78 Software Integration System modules are 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 4-12 to open a dialog.

(2) Click the [Download RL78 Software Integration System module] link in the [New Component] dialog box to start the download.

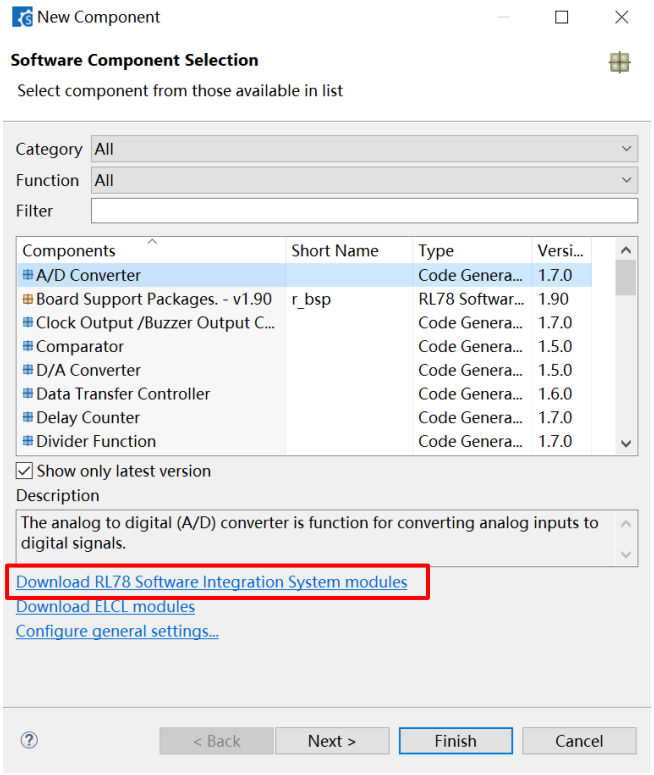



**Figure 4-36 RL78 Software Integration System Download Link**

Note: Downloading requires login to "My Renesas". If the user has not logged in, the following dialog box will prompt the user to log in.

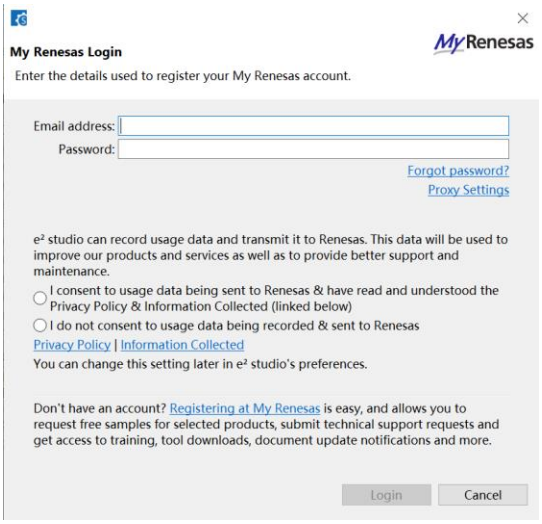



**Figure 4-37 Login to My Renesas**

(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.

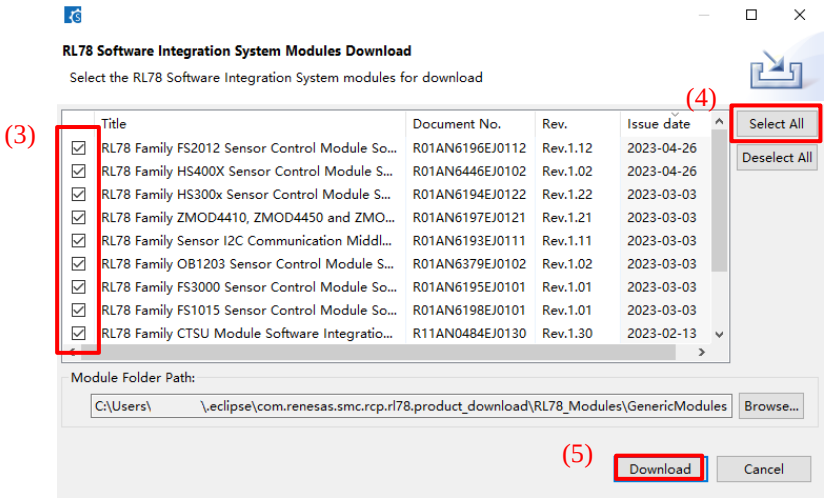



**Figure 4-38 Downloading RL78 Software Integration System Modules**

### 4.4.12 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].

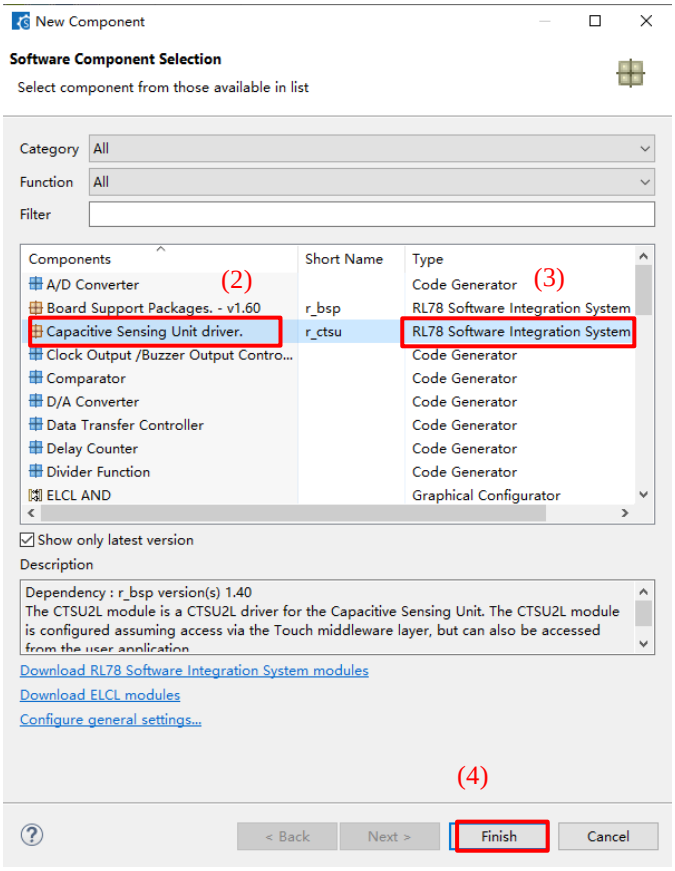



**Figure 4-39 Adding RL78 Software Integration System Module**

### 4.4.13 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.

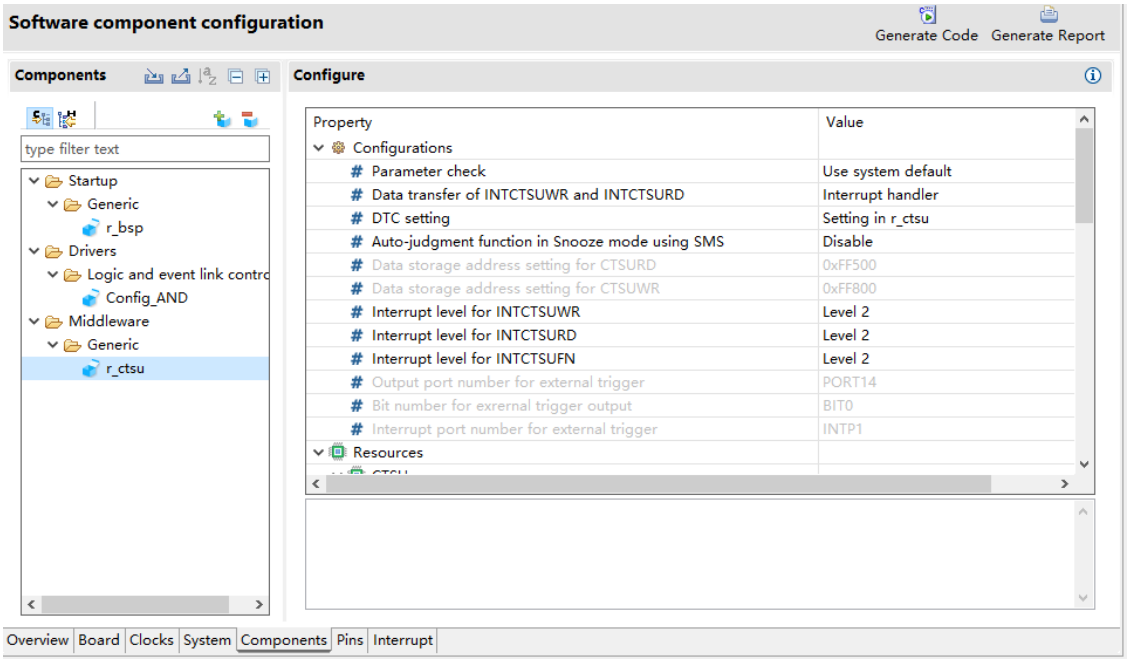



**Figure 4-40 Setting RL78 Software Integration System Module**

### 4.4.14 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.

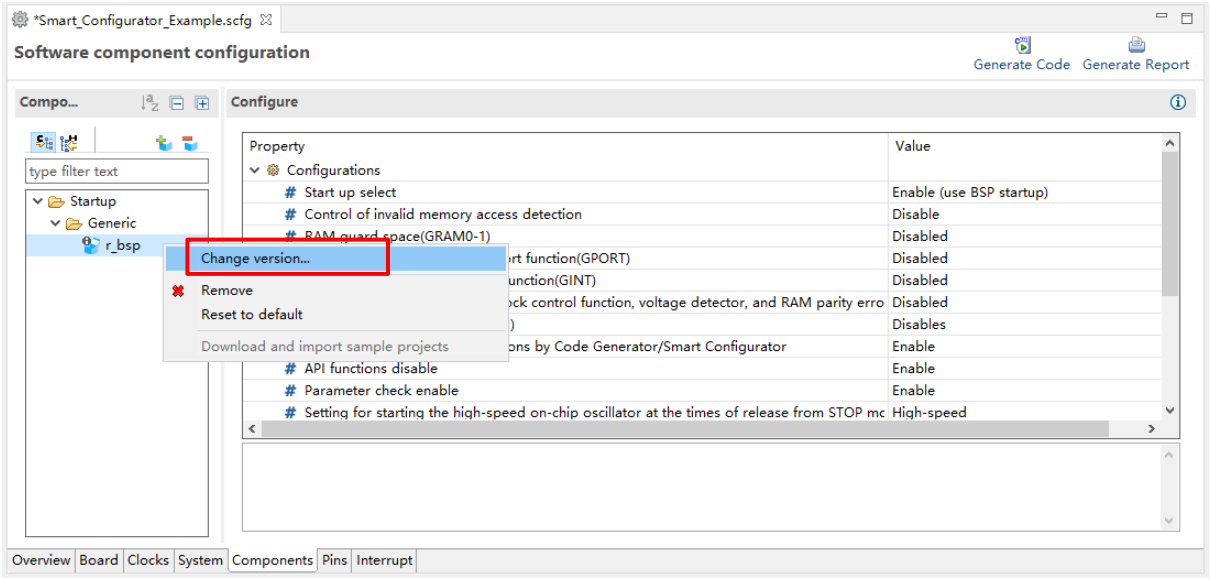



**Figure 4-41 Version Change of BSP Configuration**

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

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

(4) Click [Next].

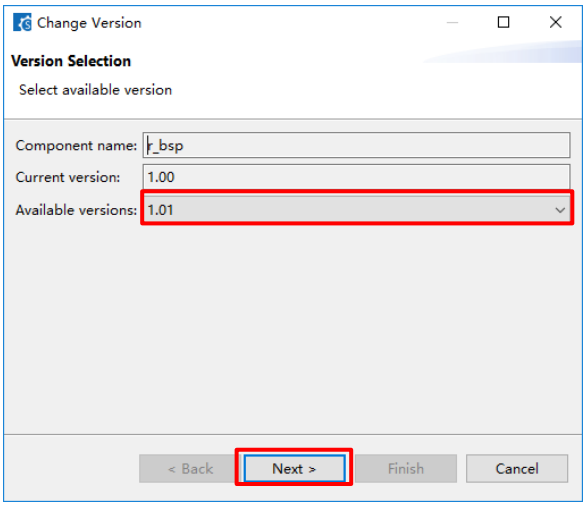



**Figure 4-42 Select Version of BSP Component**

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

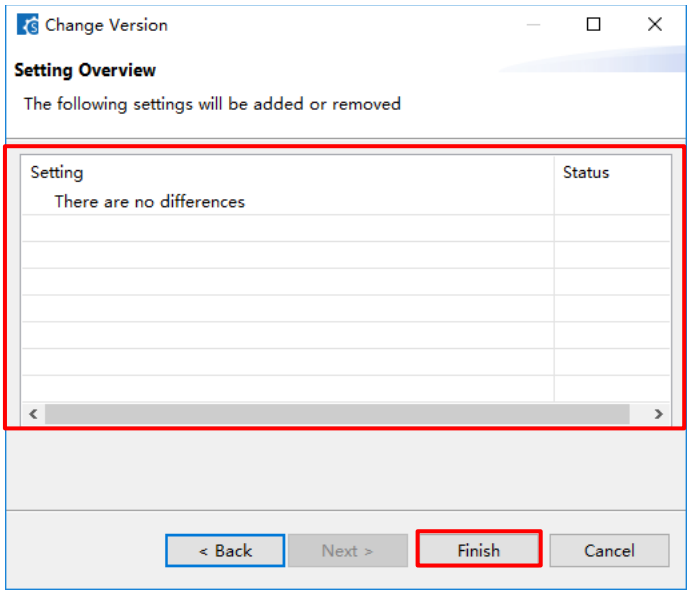



**Figure 4-43 Confirm Setting Change Item**

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

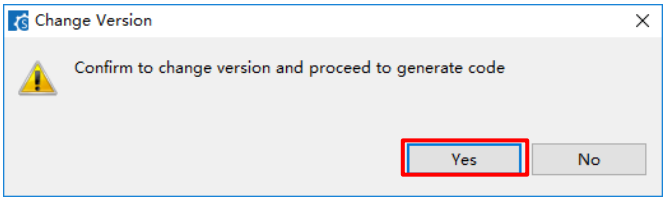



**Figure 4-44 Confirm Version Change**

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

### 4.4.15 Export Component Configuration

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

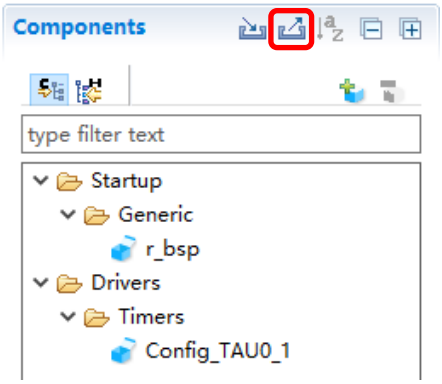



**Figure 4-45 Export Configuration (xml format)**

### 4.4.16 Import Component Configuration

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

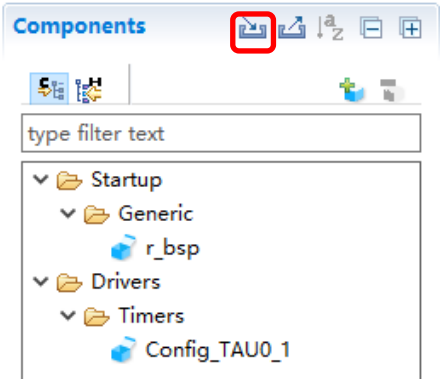



**Figure 4-46 Import Configuration (xml format)**

### 4.4.17 Configure General Setting of the Component

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

If you want to change the settings, please click the [Configure general settings...] link on the [Software Component Selection] page displayed in the [New Component] dialog (Figure 4-13) and display the [Preferences] dialog. Or click "Preferences" of "Window" in Main Menu.

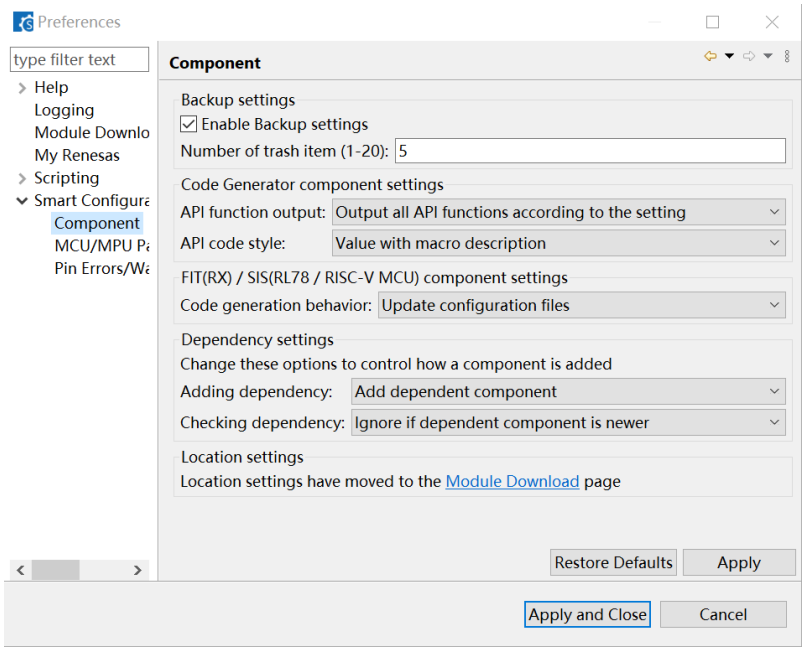



**Figure 4-47 Configure General Setting of Component**

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.

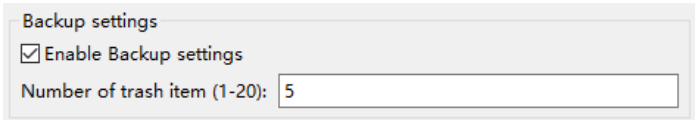



**Figure 4-48 Trash number setting**

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.

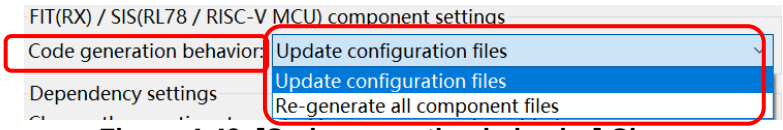



**Figure 4-49 [Code generation behavior] Change**

3. To only generate initialization API function, please 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 change back to default option setting: [Output all API functions according to the setting], then all API functions will be generated again.

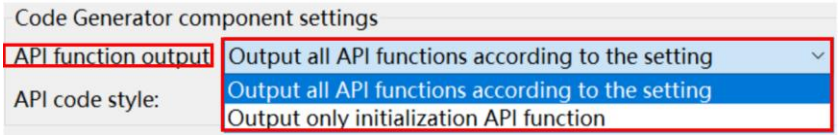



**Figure 4-50 [RL78 API function output] Change**

From Smart Configurator for RL78 V1.4.0, 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.

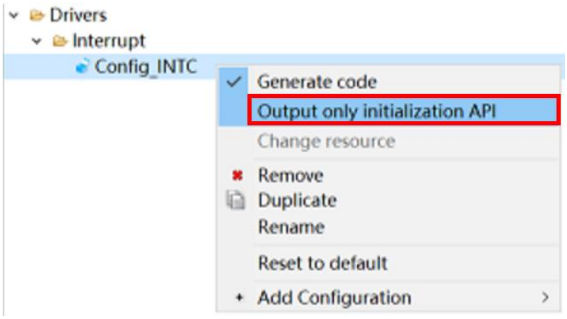



**Figure 4-51 Context Menu "Output only initialization API" for Each Configuration**

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.

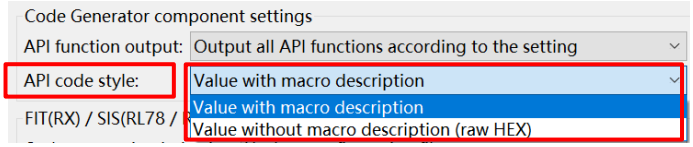



**Figure 4-52 [API code style] Change**

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

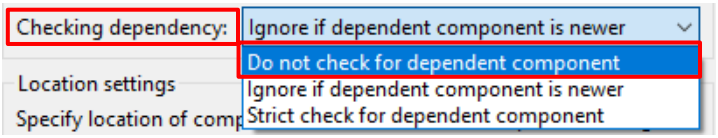



**Figure 4-53 [Checking dependency] Change**

## 4.5 Pin Settings

The [Pins] page is used for assigning pin functions. Users 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.

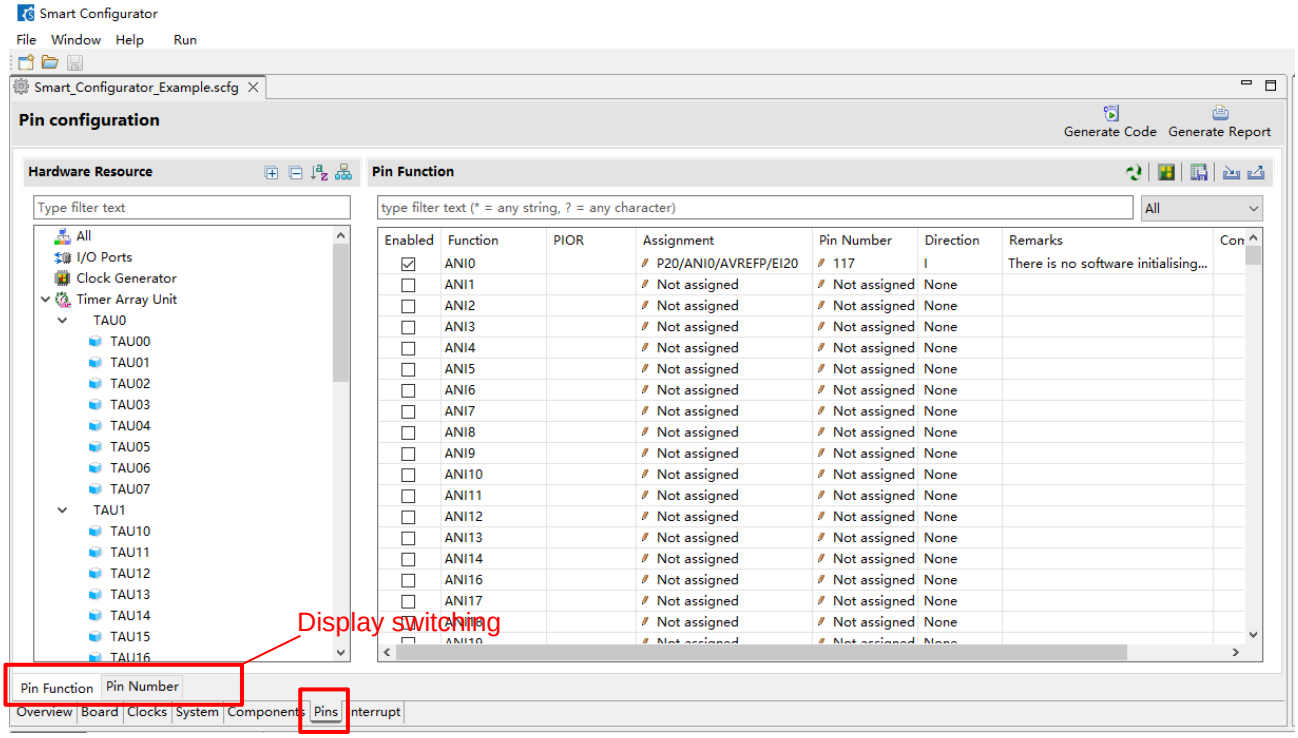



**Figure 4-54 [Pins] Page ([Pin Function])**

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

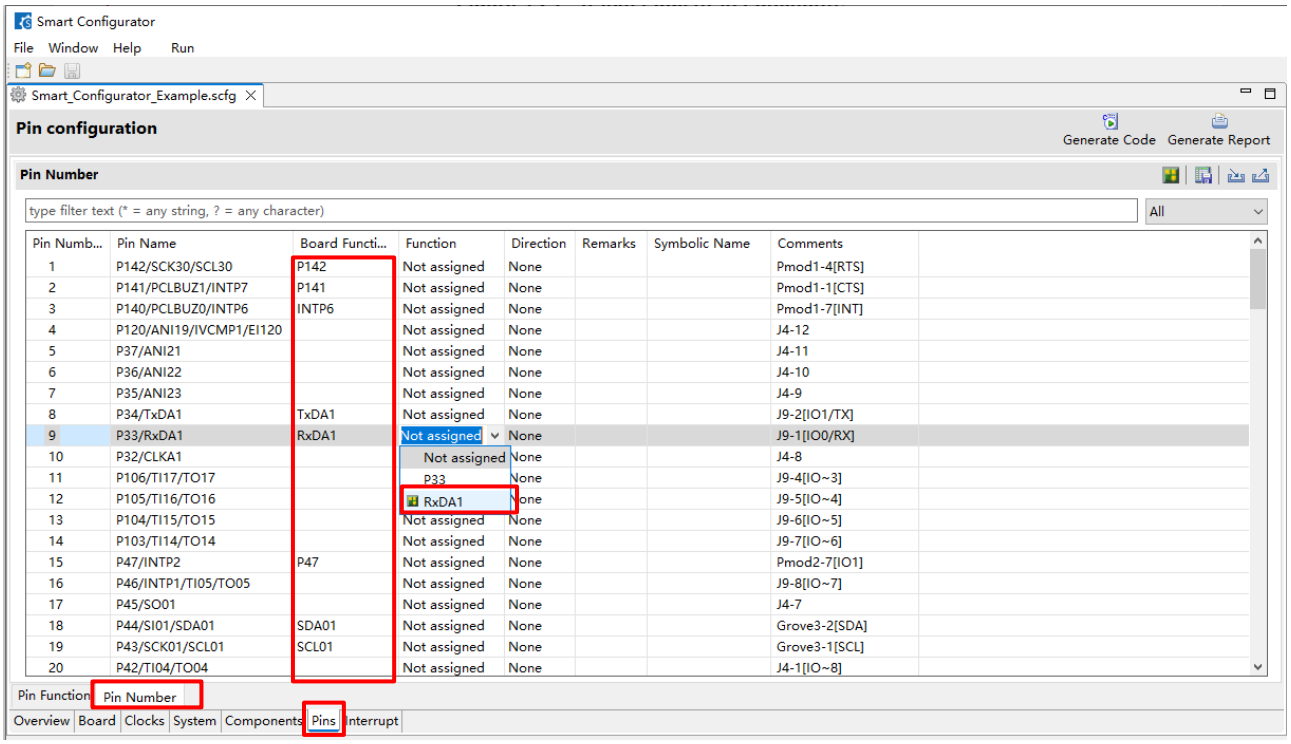



**Figure 4-55 [Pins] Page ([Pin Number])**

### 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) If user change one of pin assignment, 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.

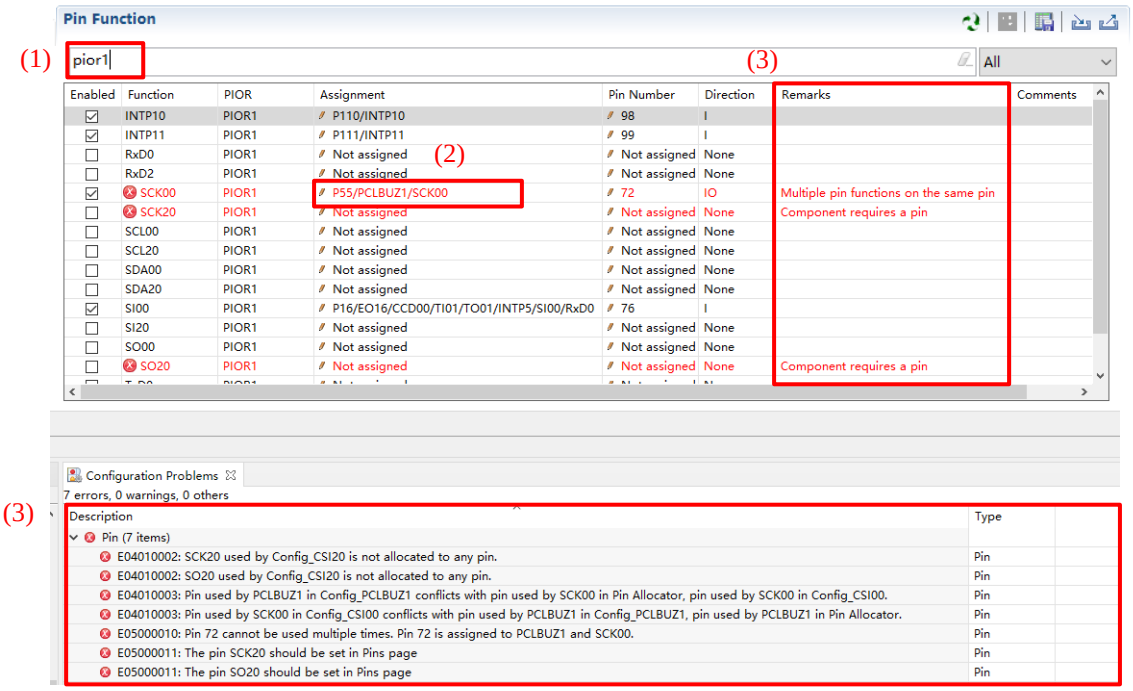



**Figure 4-56 PIOR Filter Function**

The PIOR setting can be reflected into r_bsp file in: \<ProjectDir>\src\smc_gen\r_config\r_bsp_config.h file. To change the PIOR setting code value, please change the assignment of related pin and generate code again.

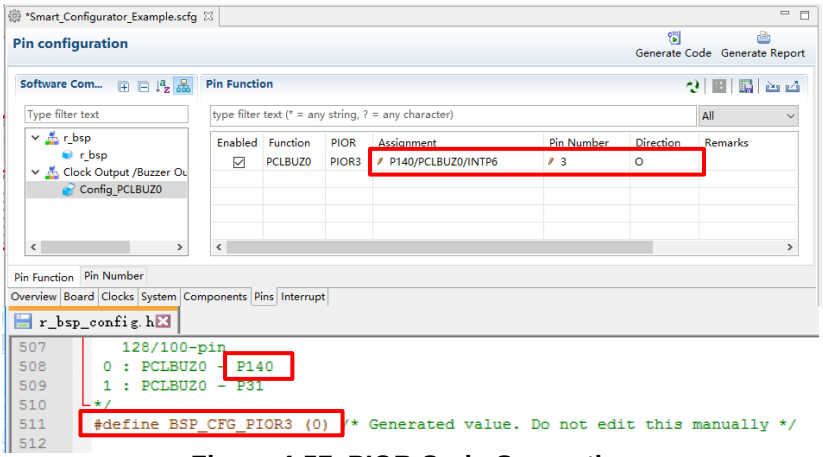



**Figure 4-57 PIOR Code Generation**

### 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.

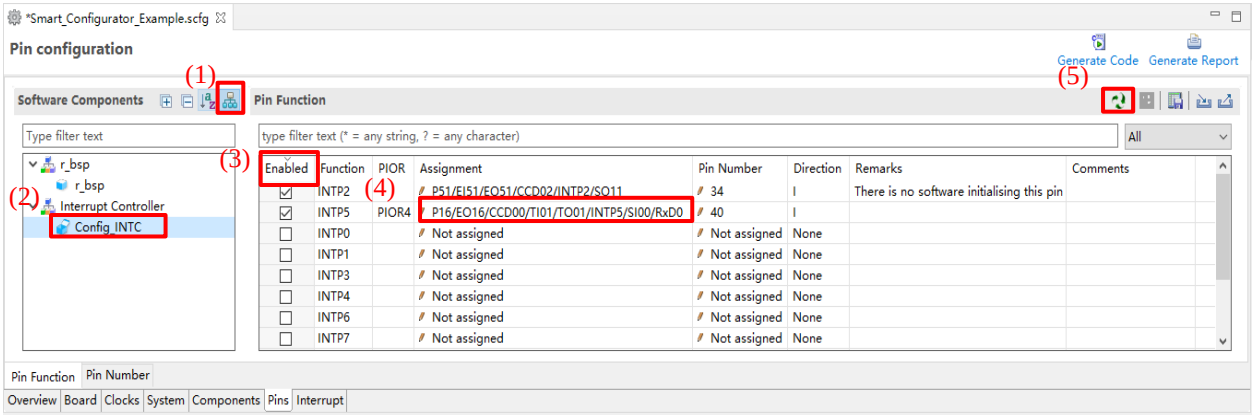



**Figure 4-58 Pin Settings – Assigning Pins on the [Pin Function] List**

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. In this case, no initialization code will be generated, so add the component.

### 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...].

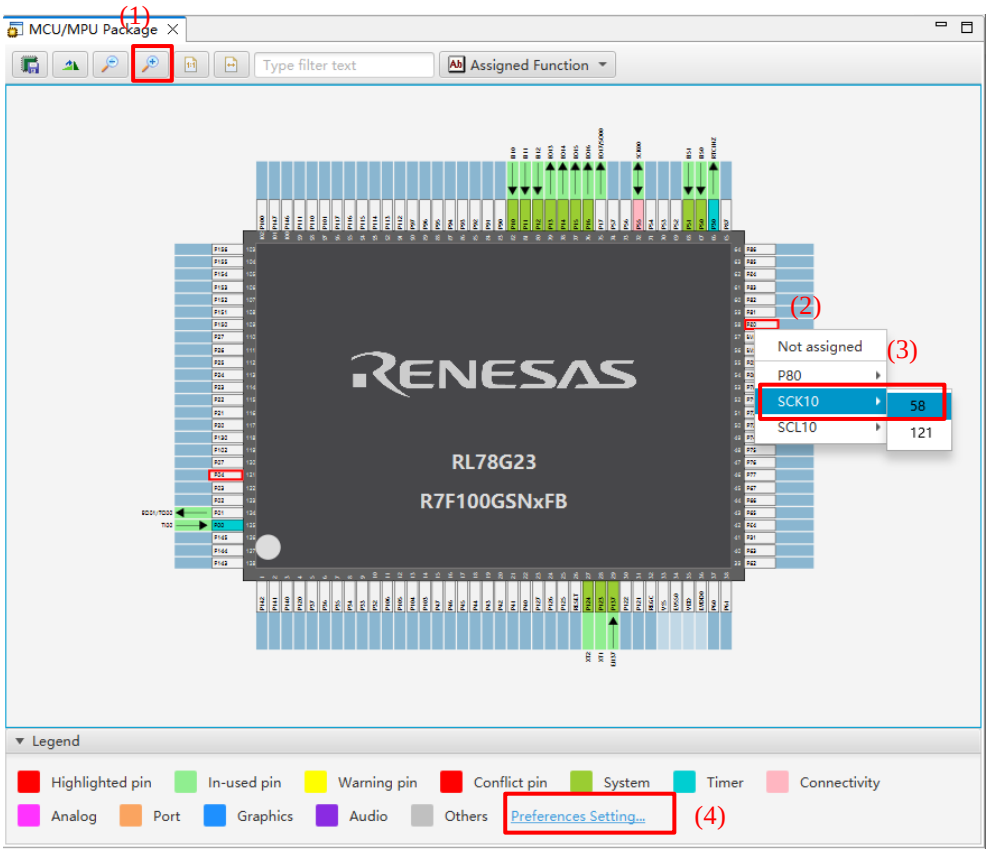



**Figure 4-59 Assigning Pins Using the MCU/MPU Package View**

### 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.

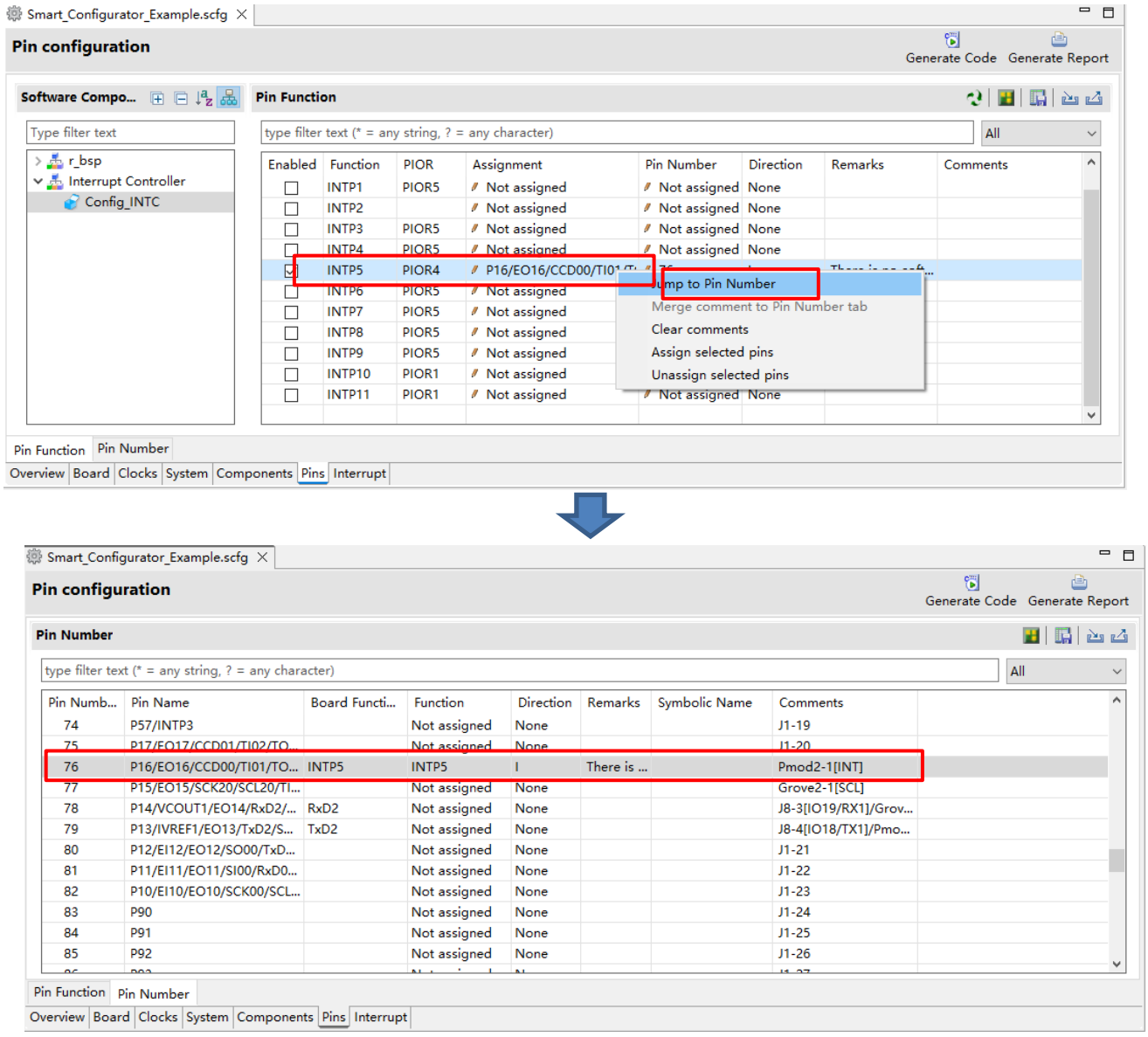



**Figure 4-60 Jump to Pin Number**

### 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.

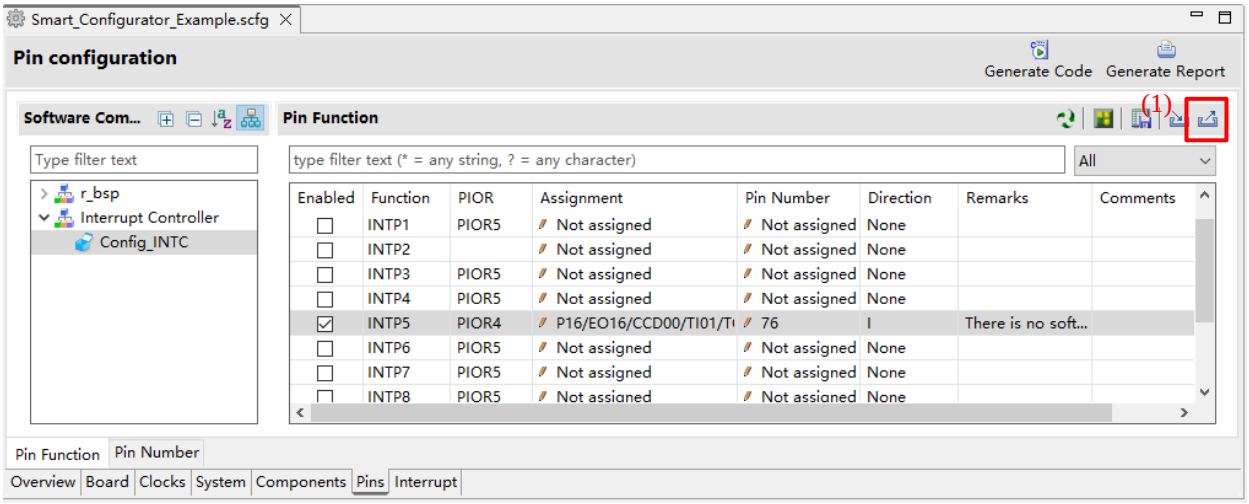



**Figure 4-61 Exporting Pin Settings to an XML File**

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.

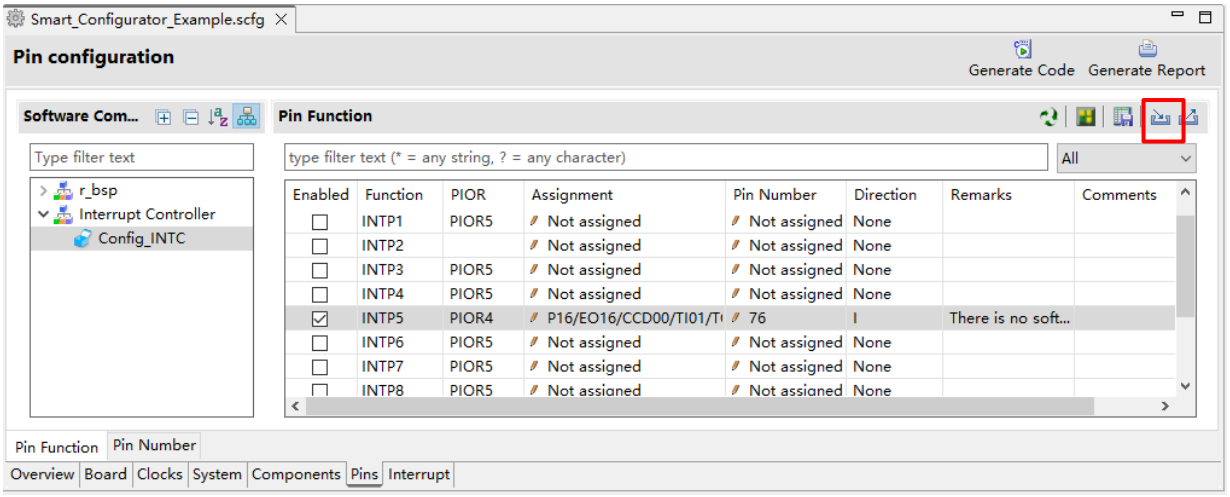



**Figure 4-62 Importing Pin Settings from an XML File**

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 of the board at once. The following describes the procedure for collective setting of pins.

(1) Select a board setting information except [Custom User Board] in [Board] page. Please 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].

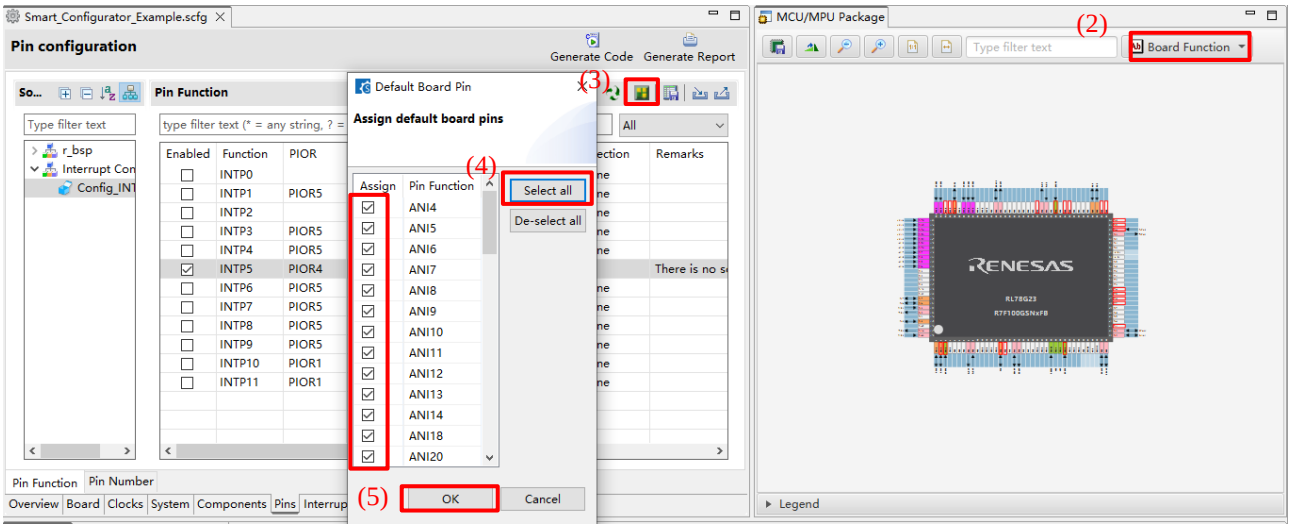



**Figure 4-63 Setting for Initial Pin Configuration**

If the 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.

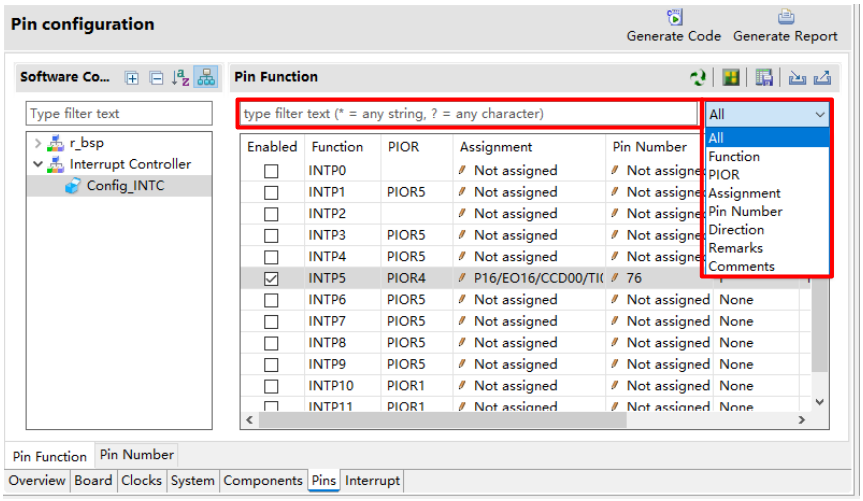



**Figure 4-64 Filter for [Pin Function] Page**

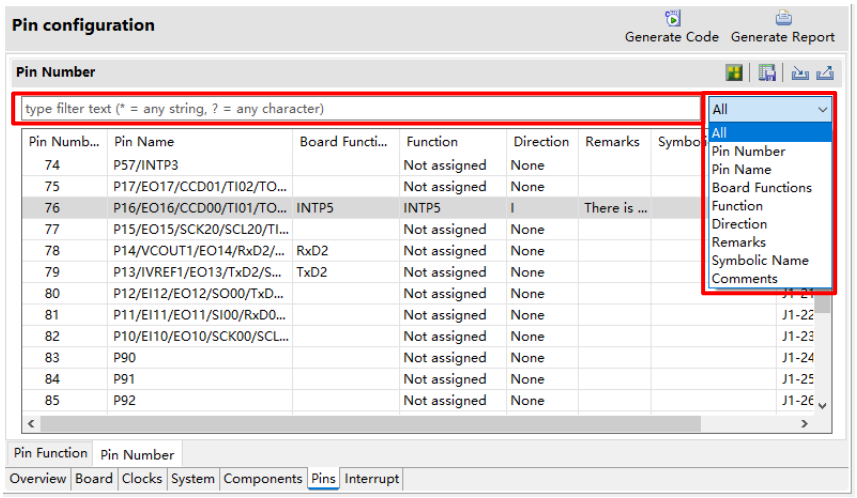



**Figure 4-65 Filter for [Pin Number] Page**

### 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 you want to control it, on the [New Component] dialog, click the [Configure general settings...] link to display the [Preferences] dialog. Then select [Smart Configurator] > [Pin Errors/Warnings] and use the combo boxes to change the errors/warning setting.

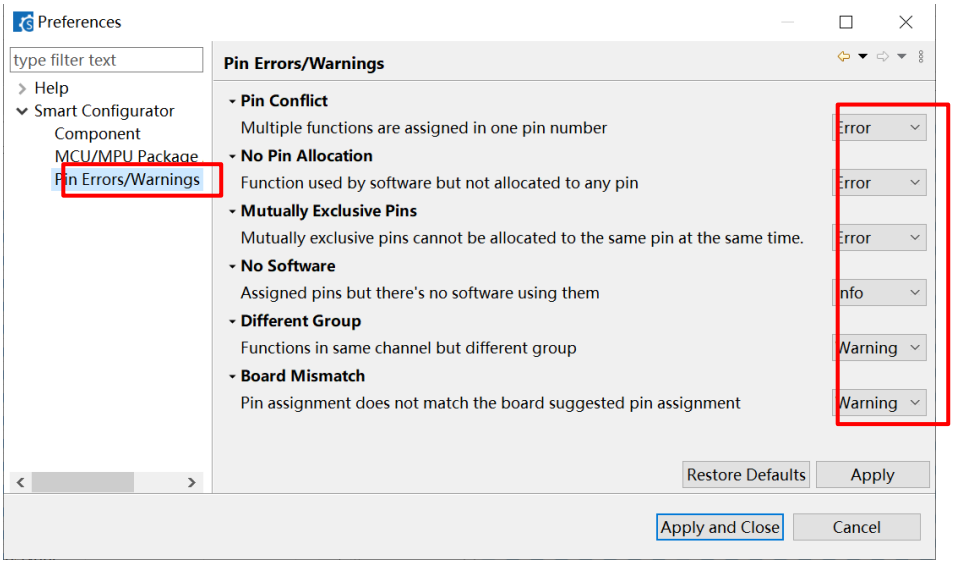
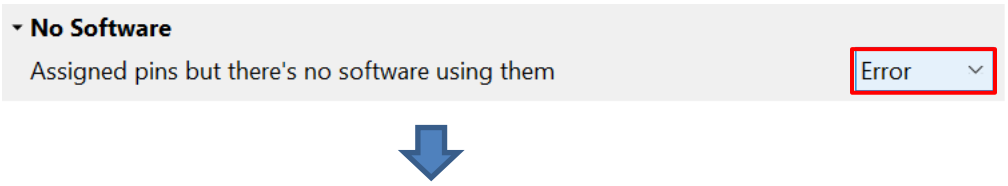



**Figure 4-66 Pin Errors/Warnings Settings at Preferences**

Example: Change "No Software" setting from "Info" to "Error"



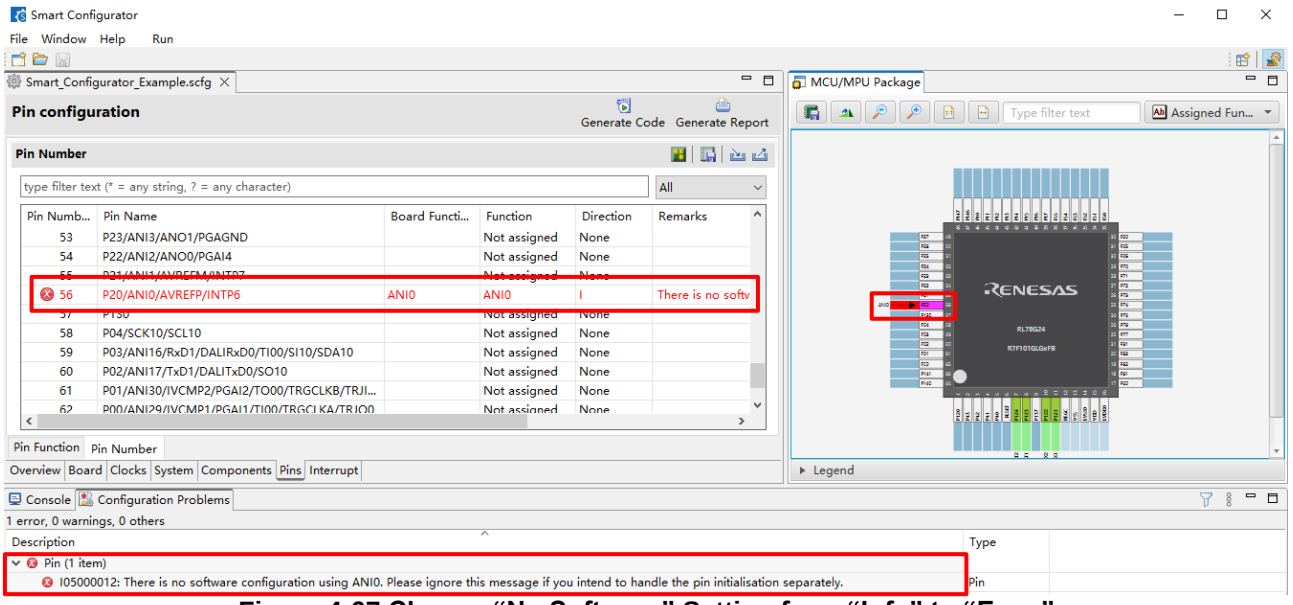




**Figure 4-67 Change "No Software" Setting from "Info" to "Error"**

## 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.

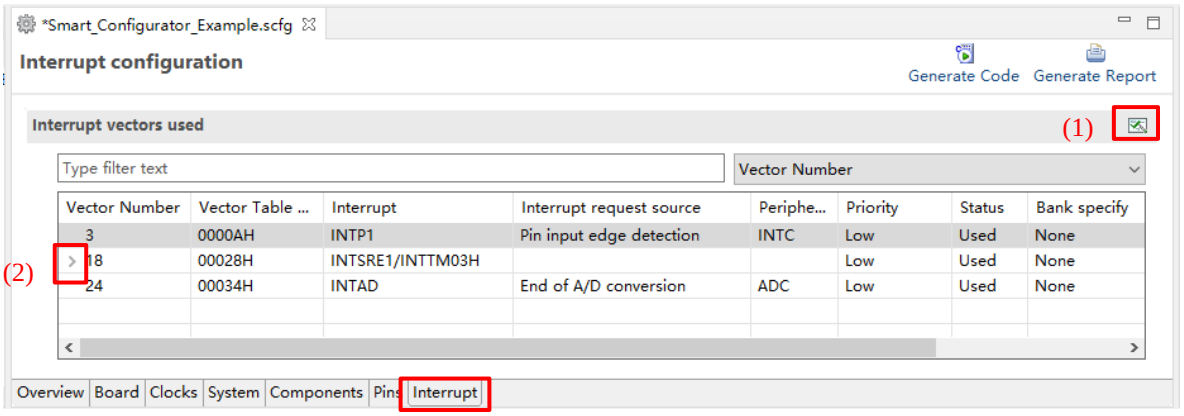



**Figure 4-68 [Interrupts] Page**

### 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 want to change priority setting on this page.

(2) Click the priority cell and select an interrupt priority level from the drop-down list.

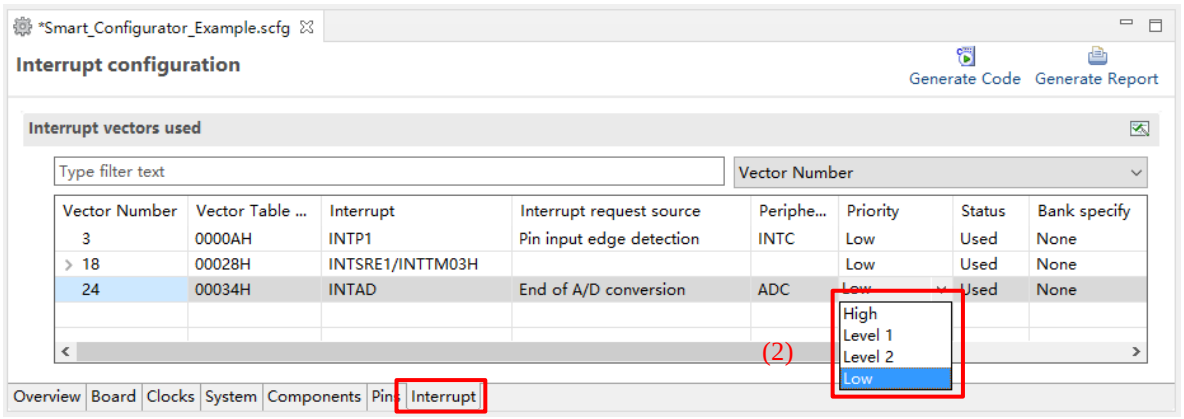



**Figure 4-69 Interrupt Settings**

### 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 user 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.

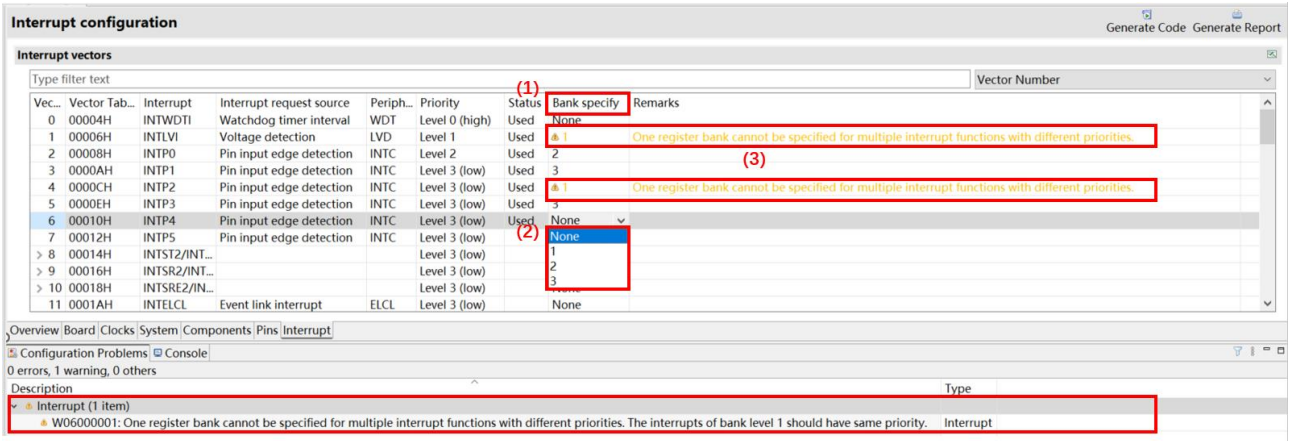



**Figure 4-70 Change Interrupt Bank Setting Example**

The interrupt bank setting can be reflected into generated code in component's {ConfigurationName}_user.c file.

```
#pragma vector = INTP1_vect
#pragma bank = 1
__interrupt static void r_Config_INTC_intp1_interrupt(void)
{
    /* Start user code for r_Config_INTC_intp1_interrupt. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}
```

**Figure 4-71 Interrupt Bank Setting Example (IAR Project)**

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

# 5. Managing Conflicts

When user add a component or configuring a pin or interrupt may cause problems in terms of resource conflict and missing dependency modules. This information will be displayed in the Configuration Problems view. User can refer to the displayed information to fix the conflict issues and 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.

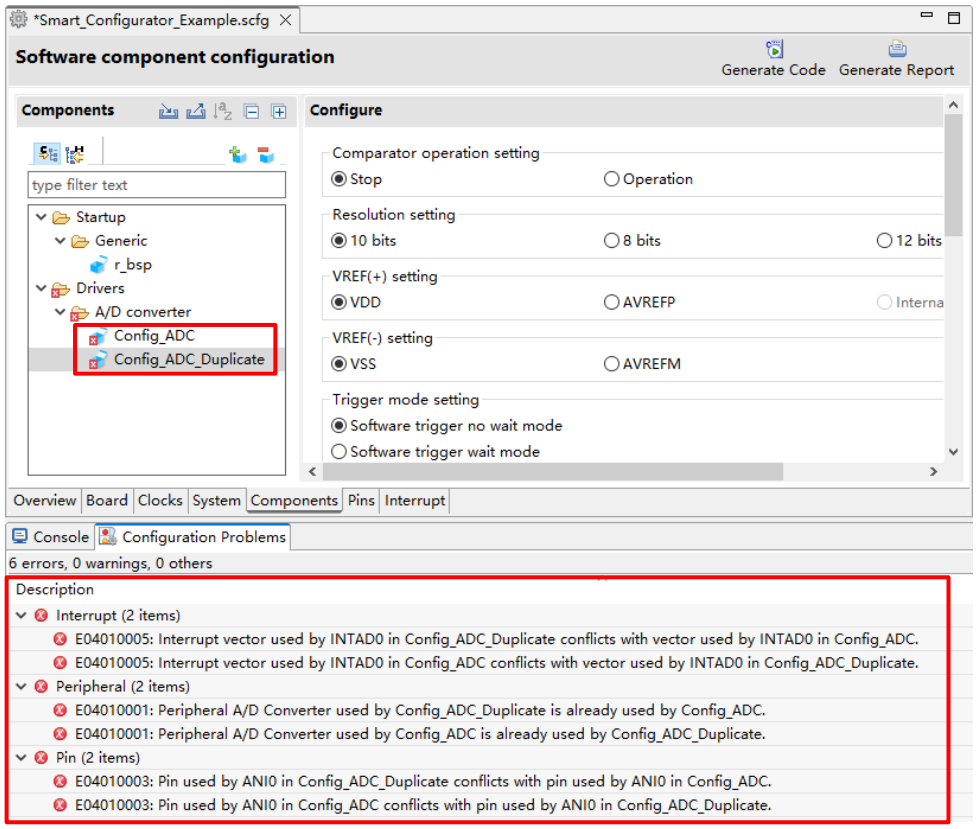



**Figure 5-1 Resource Conflicts**

## 5.2 Resolving Pin Conflicts

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

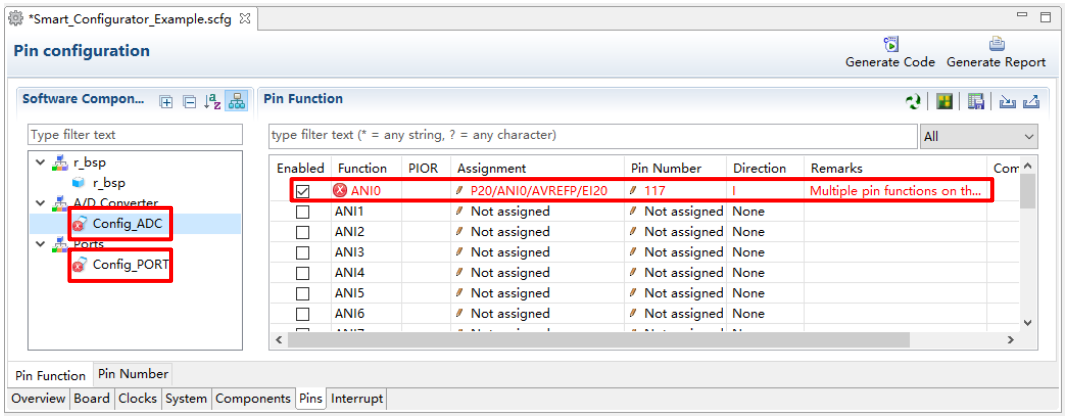



**Figure 5-2 Pin Conflicts**

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

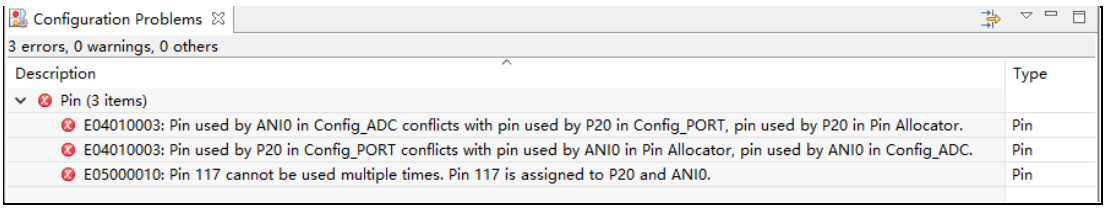



**Figure 5-3 Pin Conflict Messages**

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

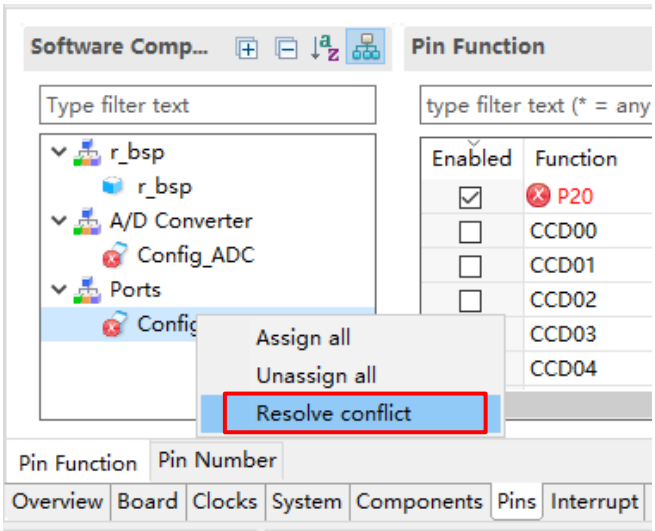



**Figure 5-4 Resolving Pin Conflicts**

The pins of the selected node 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 Generating Source Code File

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

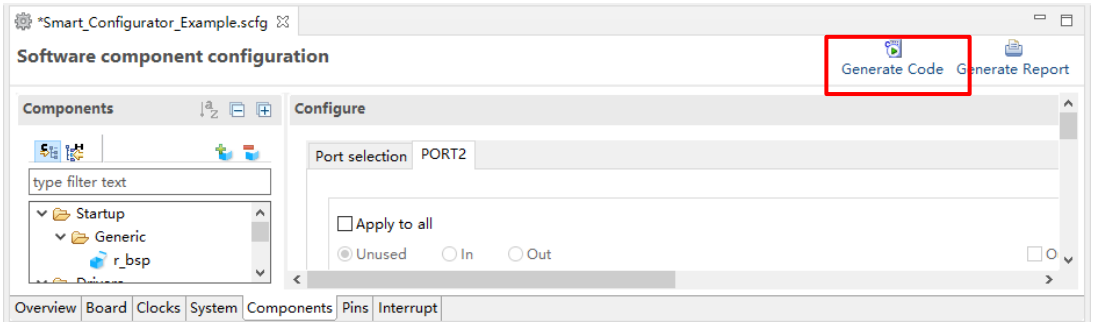



**Figure 6-1 Generating a Source File**

The Smart Configurator generates a source file in <ProjectDir>\src\smc_gen and IAR related files in save location (refer to 3.3.1 Creating a New Smart Configurator Configuration File). If user's Smart Configurator has already generated a file, a backup copy of that file is also generated (refer to the chapter 9 Backing up Generated Source Code).

## 6.2 Configuration of Generated Files and File Names

Figure 6-2 Configuration of Generated Files and File Names, shows the folders and files output by the Smart Configurator. Function main () is included in main.c which is generated when clicking "Generated Code" in the Smart Configurator.

*r_xxx* indicates the names of Software Integration System Modules, "*ConfigName*" indicates the name of the configuration formed by the component settings and "*ProjectName*" indicates a project name set in the Smart Configurator.

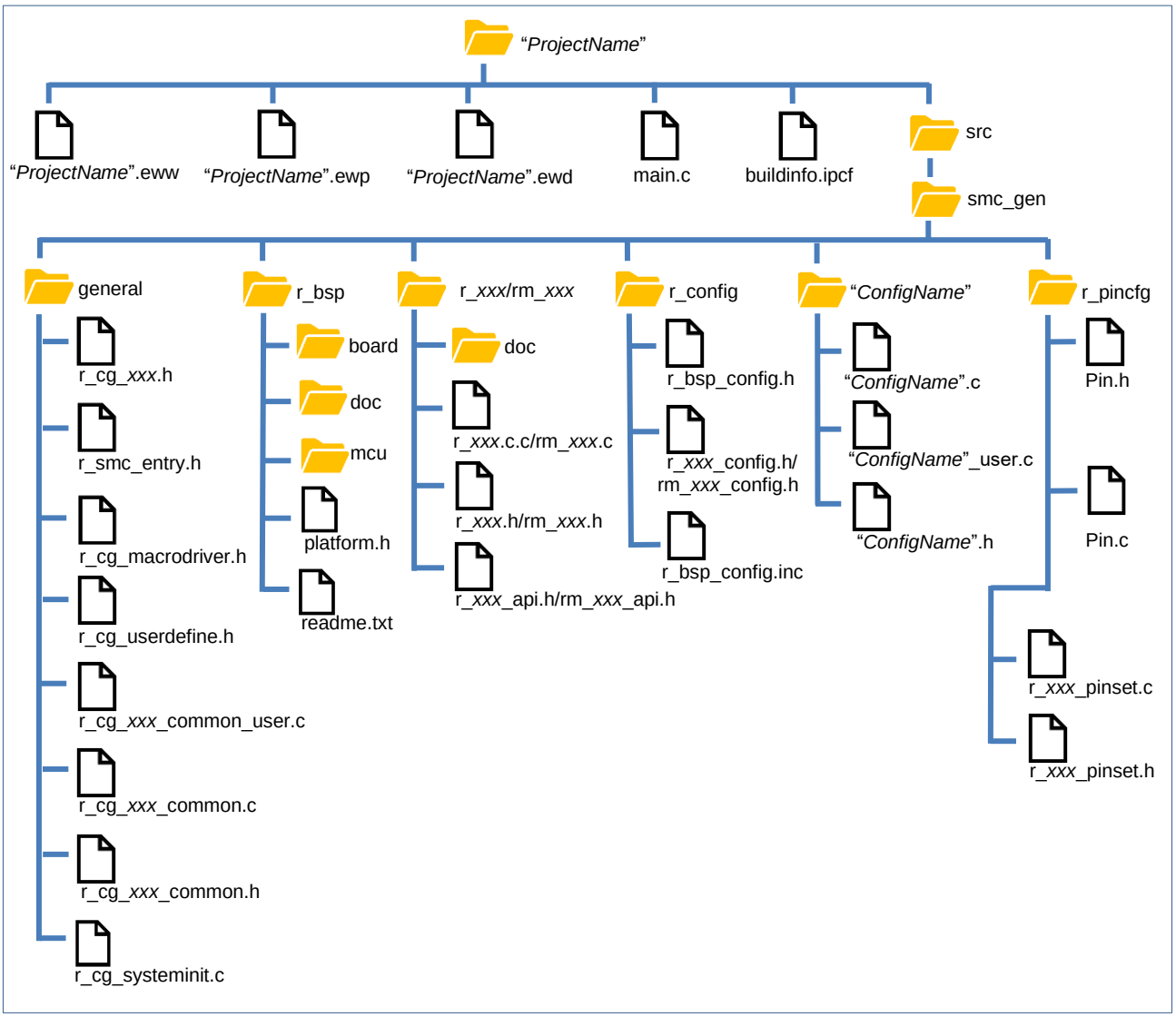



**Figure 6-2 Configuration of Generated Files and File Names**

| Folder | File | Description |
|---|---|---|
| {*ProjectName*} | {*ProjectName*}.eww | This file generates once only in the first code generation.<br>{*ProjectName*}.ewp file path is specified in this file. |
| | {*ProjectName*}.ewp | This file generates once only in the first code generation.<br>It appends the "buildinfo.ipcf" and "main.c" files at the end of this file. |
| | {*ProjectName*}.ewd | This file generates once only in the first code generation.<br>It is totally same as the default *.ewd file generated by IAR Embedded Workbench. |
| | main.c | This file generates once only in the first code generation.<br>It contains function main (). |
| | buildinfo.ipcf | This file is always generated.<br>It contains source file registration information. From Smart Configurator for RL78 V1.4.0, the name of .ipcf file is "buildinfo.ipcf". If user loads a project which created before Smart Configurator for RL78 V1.4.0, the .ipcf file will be re-generated as "buildinfo.ipcf" and the original .ipcf file ({*ProjectName*}.ipcf) still exist under the folder, won't be removed. |
| general | | This folder is always generated.<br>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.<br>This file includes the header files of Code Generator drivers that are added to the project.<br>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.<br>This header file contains common macro definitions used in drivers. |
| | r_cg_userdefine.h | This file is always generated.<br>User can add macro definitions in the dedicated user code areas. |
| | r_cg_systeminit.c | This file is always generated.<br>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_bsp | | This folder is always generated.<br>It consists of multiple subfolders (board, doc, mcu) with:<br>- Initialization codes to start up the MCU before entering main () (e.g. setup stack, initialize memory)<br>- Definitions of all SFR registers in iodefine.h (mcu folder)<br>- Application note of r_bsp (doc folder)<br>It also contains platform.h that will include r_bsp.h of the device used in the project. |

| Folder | File | Description |
|---|---|---|
| r_*xxx*/<br>rm_*xxx*(Note*1) | | This folder is generated for the RL78 Software Integration System module that is added to the project.<br>It consists of:<br>- doc folder: Application note of this RL78 Software Integration System module<br>- r_xxx.c/rm_xxx.c(Note*1): RL78 Software Integration System module source file<br>- r_xxx.c/rm_xxx.h(Note*1): RL78 Software Integration System header file<br>- 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 |
| r_config | | This folder is always generated.<br>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.<br>It contains configurations of r_bsp for clock initialization and other MCU related settings. Some MCU related settings are generated by Smart Configurator (e.g. package type) and other settings (e.g. stack size) are configured by user manually. |
| | r_bsp_config.inc | This file is always generated.<br>It generates configuration header file. |
| | 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.<br>It is generated for supporting pin symbol and included in smc_entry.h. |
| | Pin.c | This file is always generated.<br>It is generated pin setting enabled in [Pins] page. It only generate pin setting which no need to set PIOR. |
| | 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. |
| {*ConfigName*} | | This folder is generated for the Code Generator drivers that are added to the project.<br>API functions in this folder are named after the *ConfigName* (configuration name). |
| | {*ConfigName*}.c | This file contains functions to initialize driver (R_*ConfigName*_Create) and perform operations that are driver-specific, e.g. start (R_*ConfigName*_Start) and stop (R_*ConfigName*_Stop). |
| | {*ConfigName*}_user.c | This file contains interrupt service routines and functions for user to add code after the driver initialization (R_*ConfigName*_Create).<br>User can add codes and functions in the dedicated user code areas. |
| | {*ConfigName*}.h | This is header file for {*ConfigName*}.c and {*ConfigName*}_user.c. |

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

## 6.3 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).

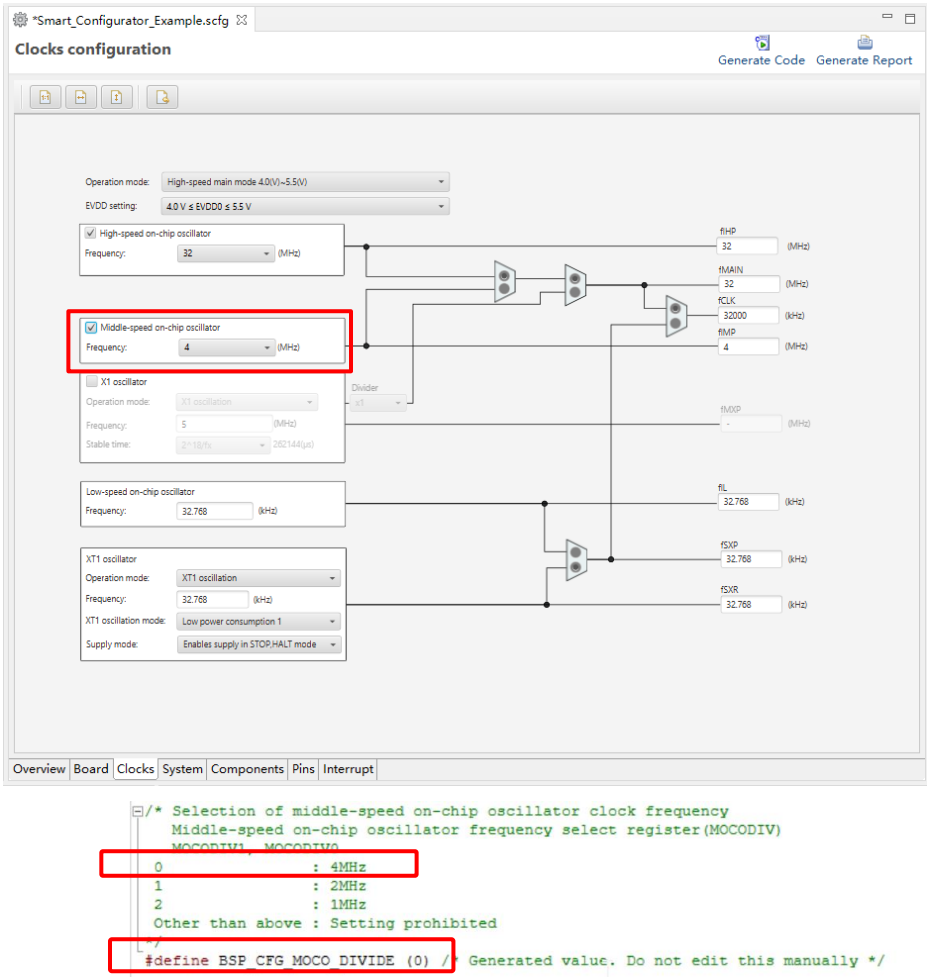



**Figure 6-3 Clocks Configuration and Generated Code in r_bsp_config.h**

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

Note: r_bsp_config.h will be backed up to trash folder before each code generation (refer to chapter 9 Backing up Generated Source Code).

## 6.4 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 ().

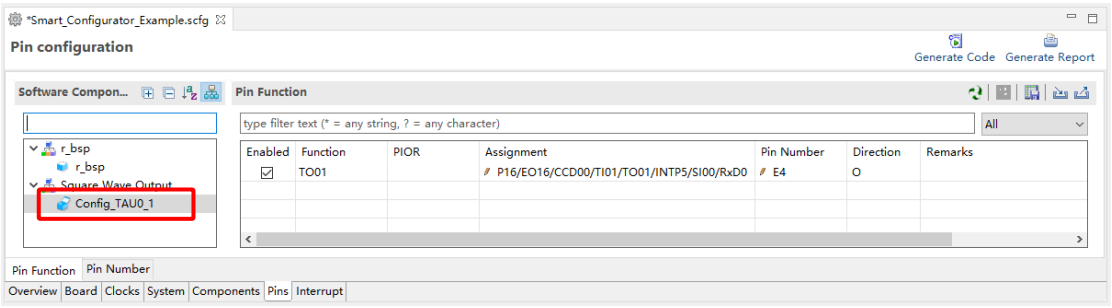



**Figure 6-4 *Config_TAU0_1* in Software Components View**

| Folder | File | Function | Component type | Description |
|---|---|---|---|---|
| {*ConfigName*} | {*ConfigName*}.c | R_*ConfigName*_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.

User will call the pin initialization codes in main ().

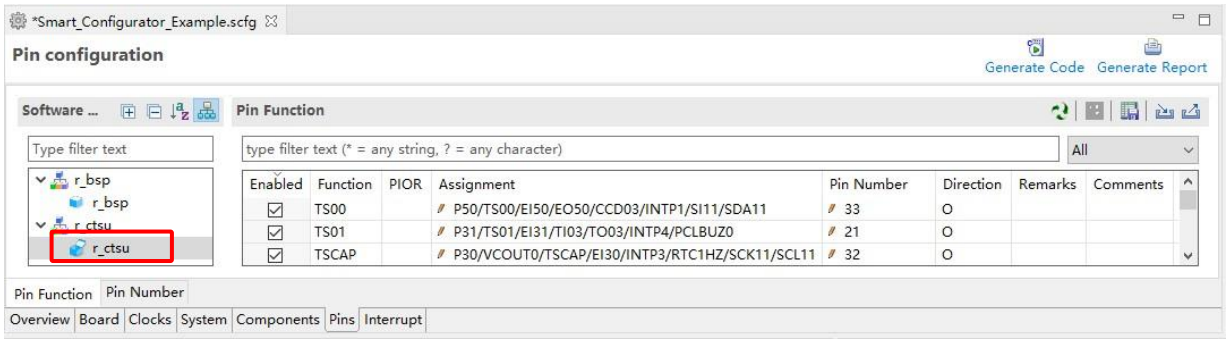



**Figure 6-5 *r_ctsu* in Software Components View**

| Folder | File | Function | Component type | Description |
|---|---|---|---|---|
| r_pincfg | {*ConfigName*} _pinset.c | R_{*PeripheralName*}_ PinSetInit | RL78 Software Integration System | This API function initializes the pins used by this driver. User need call this function in main () function. |

## 6.5 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*.

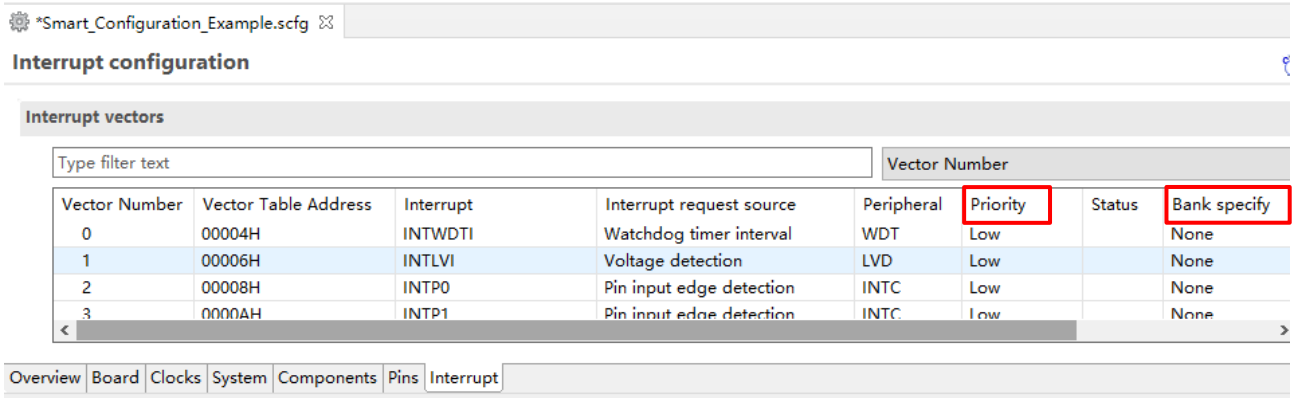



**Figure 6-6 Interrupts Configuration in Interrupts View**

| Item | Folder | File | Component type | Description |
|---|---|---|---|---|
| Priority | {*ConfigName*} | {*ConfigName*}.c | Code Generator | It is initialized in R_*ConfigName*_Create of this file. r_cg_systeminit will call this function before entering *main ()* function. |
| Bank | {*ConfigName*} | {*ConfigName*}_user.c | Code Generator | Declaration of interrupt as:<br>#pragma vector = "*InterruptVectorName*"_vect<br>#pragma bank = "*bankNumber*"<br>please see example in Figure 4-71 |

# 7. Loading Generated Files in Integrated Development Environment

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

## 7.1 Loading in IAR Embedded Workbench

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

The usage procedure is as follows.

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

(2) In the [Open Workspace] dialog box, browse to the folder where the project file is saved, select the project file (.eww), and click the [Open] button.

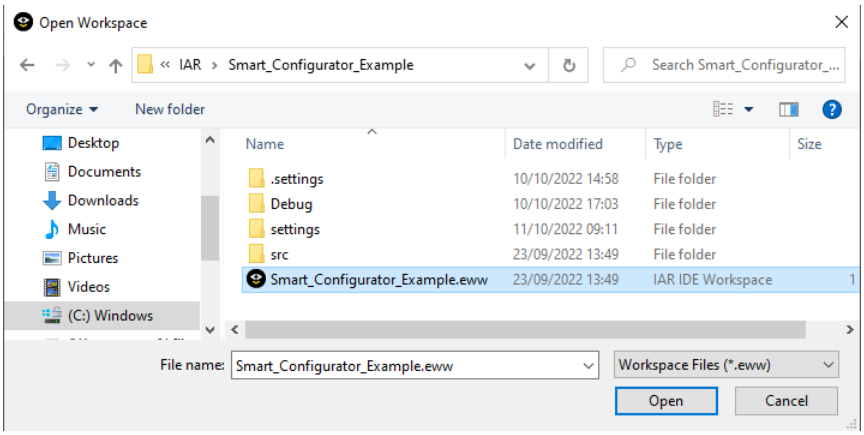



**Figure 7-1 Load a *.eww File**

(3) The source file output by the Smart Configurator is added to the IAR C project workspace.

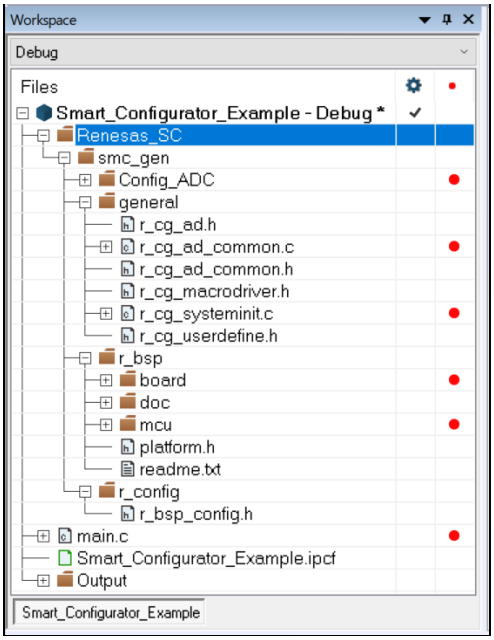



**Figure 7-2 New Files Added to IAR Workspace**

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

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

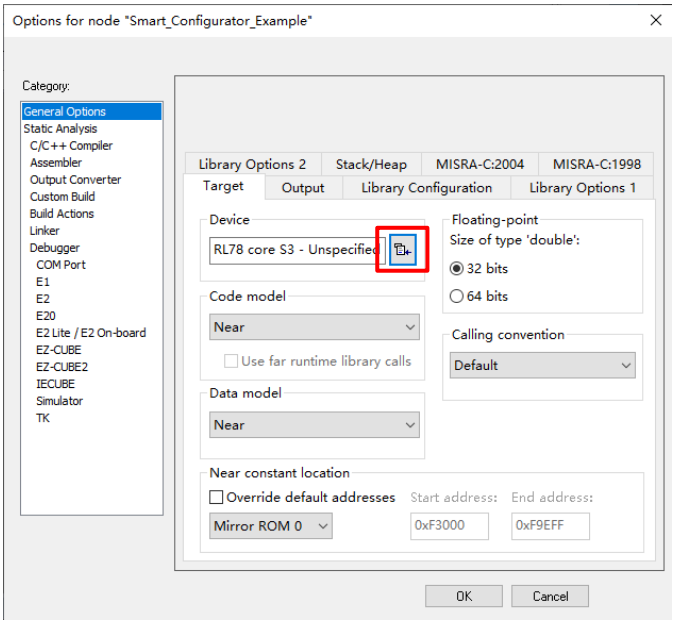



**Figure 7-3 Change Target Device**

## 7.2 Build IAR Project File

After loading Smart Configurator project file to IAR Embedded Workbench successfully, user can right-click on project name, select [Rebuild All] from context manual, then build operation will be executed successfully.

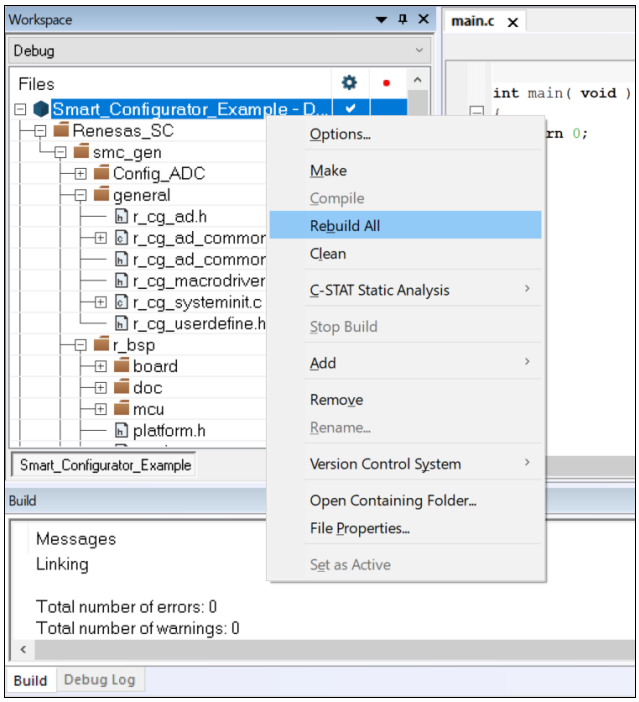



**Figure 7-4 Build C Project File in IAR**

# 8. 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 file generated by the Smart Configurator. Please follow the procedure in ， to create an IAR project file first and make configuration setup, build and run the project.

## 8.1 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 xxxx. Do not edit comment generated here */

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

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

/***********************************************************************************
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
***********************************************************************************/
#pragma vector = INTAD_vect
__interrupt static void 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 */
```

## 8.2 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.

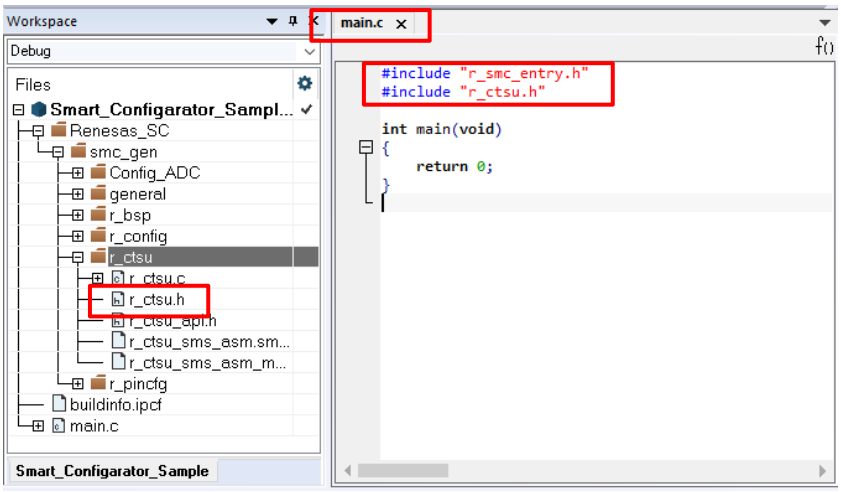



**Figure 8-1 Add Header Files**

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*).

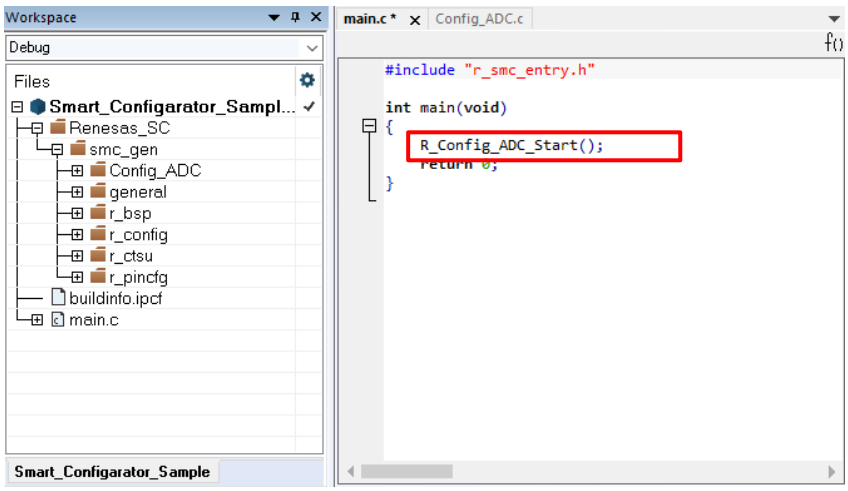



**Figure 8-2 Call Code Generator Functions**

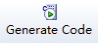
**In case of Software Integration System Modules,** refer to the examples provided in the "API Functions" chapter of corresponding Application Note.

For more reference, refer to "Smart Configurator Application Examples" in "chapter 13 Documents for Reference".

# 9. 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)] button. <Date-and-Time> indicates the date and time when the backup folder is created after code generation.

# 10. Generating Reports

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

## 10.1 Report on All Configurations (PDF or Text File)

A report is output in response to clicking on the [Generate Report (Generate Report)] button in the Smart Configurator view.

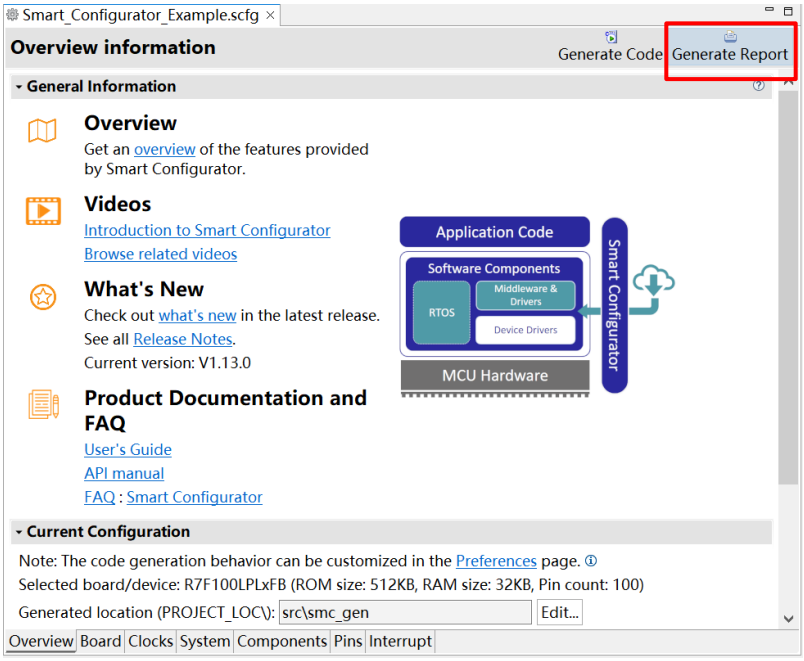



**Figure 10-1 Output of a Report on the Configuration (as a PDF/Text File)**

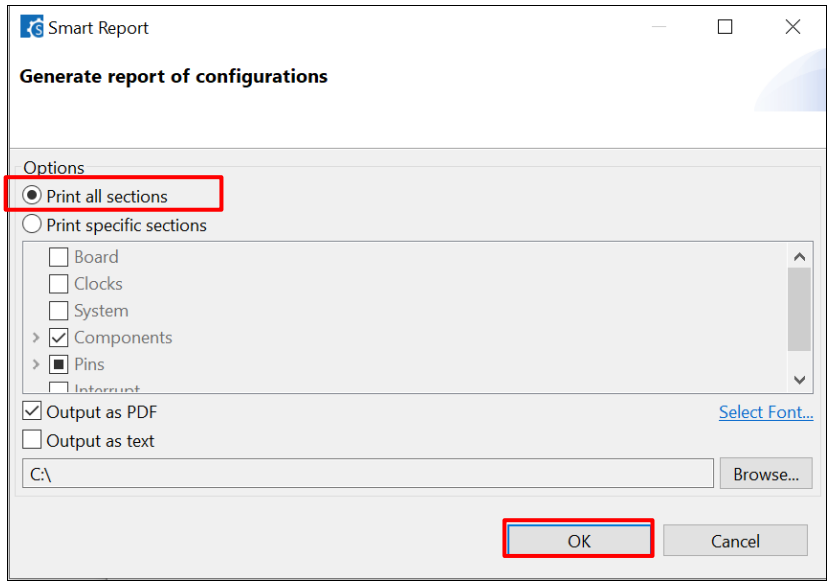



**Figure 10-2 Dialog Box for Output of a Report (Example is selecting "Output as PDF")**

## 10.2 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.

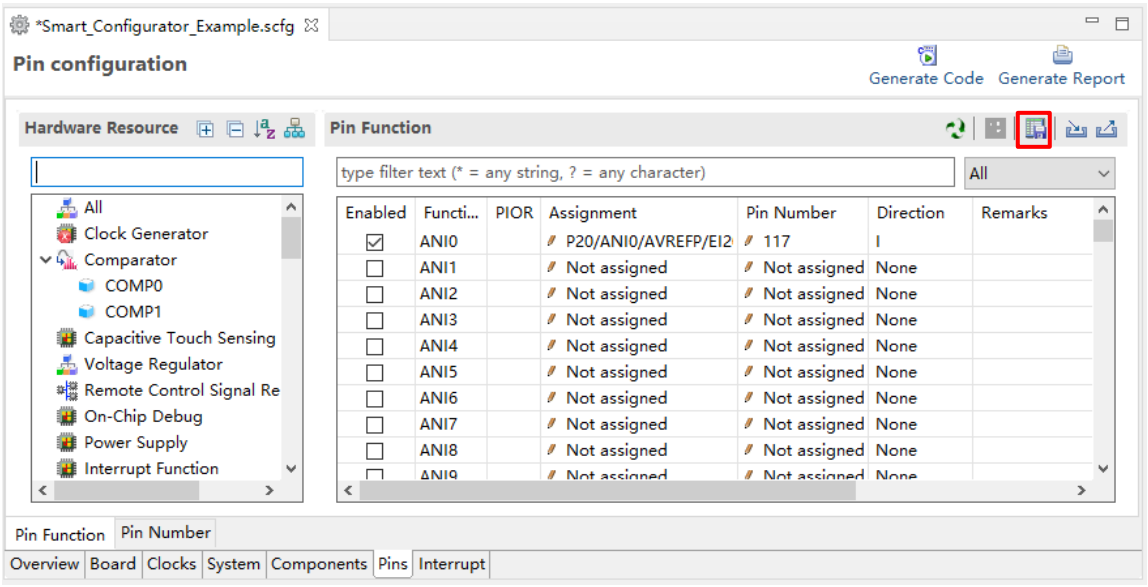



**Figure 10-3 Output of a List of Pin Functions or Numbers (in csv Format)**

## 10.3 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.

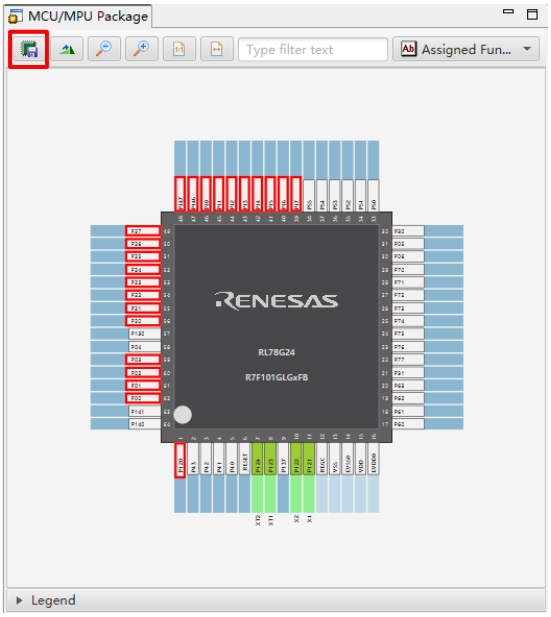



**Figure 10-4 Outputting a Figure of MCU/MPU Package (in png Format)**

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

The Smart Configurator for RL78 V1.6.0 and the 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

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.

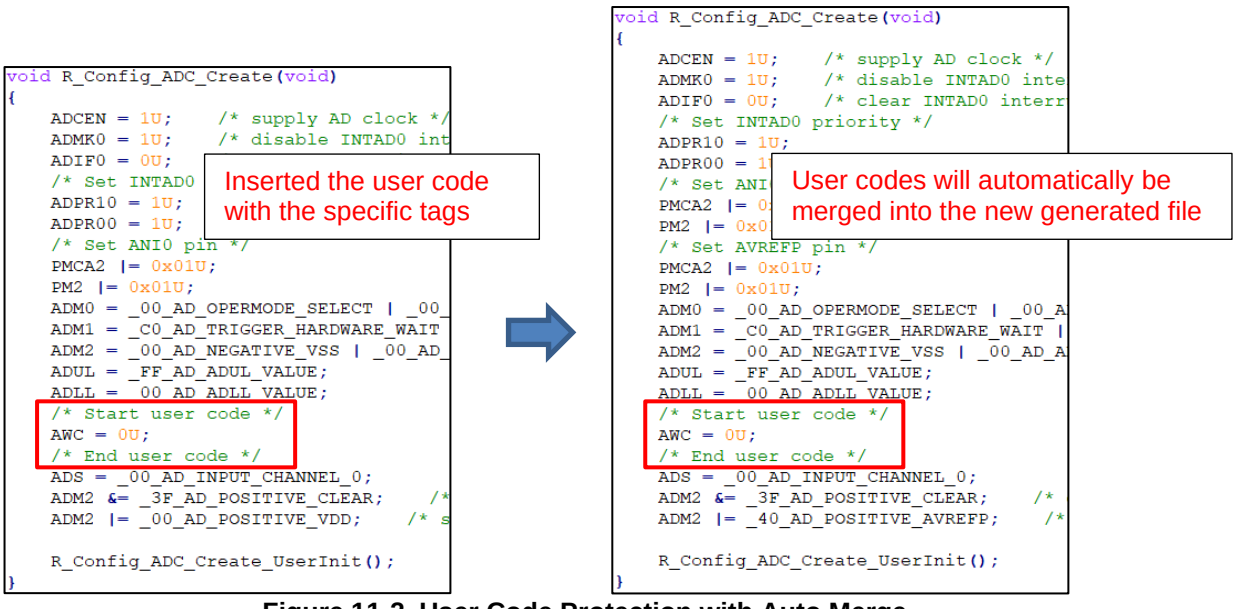



**Figure 11-2 User Code Protection with Auto Merge**

## 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 in red will be displayed in the Smart Configurator console, as shown in Figure 11-3 The Merge Conflict Message Outputted in the Smart Configurator Console.

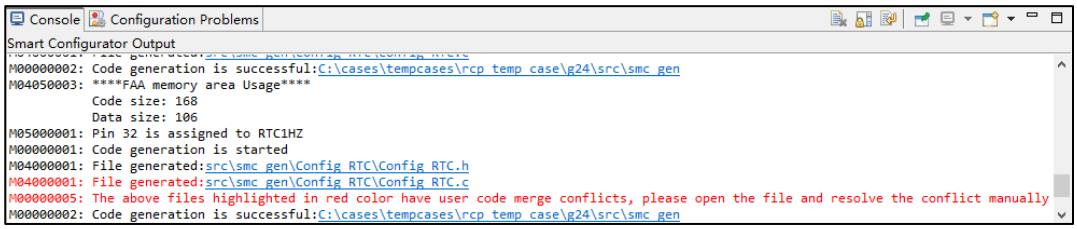



**Figure 11-3 The Merge Conflict Message Outputted in the Smart Configurator Console**

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.

### 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).

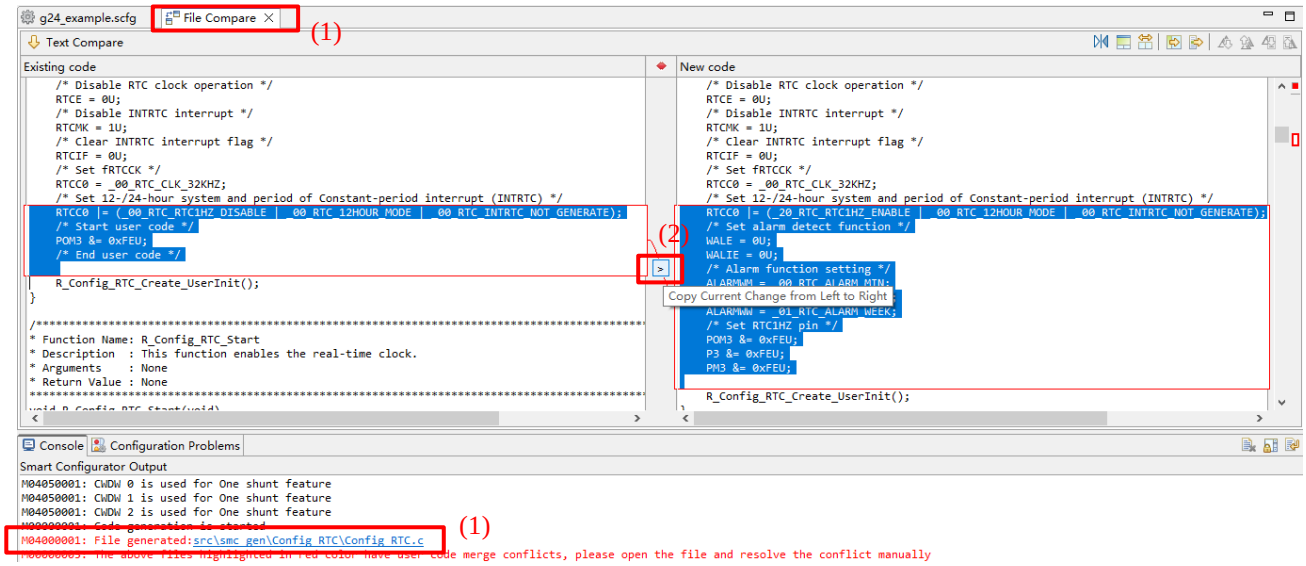



**Figure 11-4 Code before Resolving Conflict**

(3) Delete the codes that user does not want to use (Figure 11-5 Code after Applying "Copy Current Change from Left to Right").

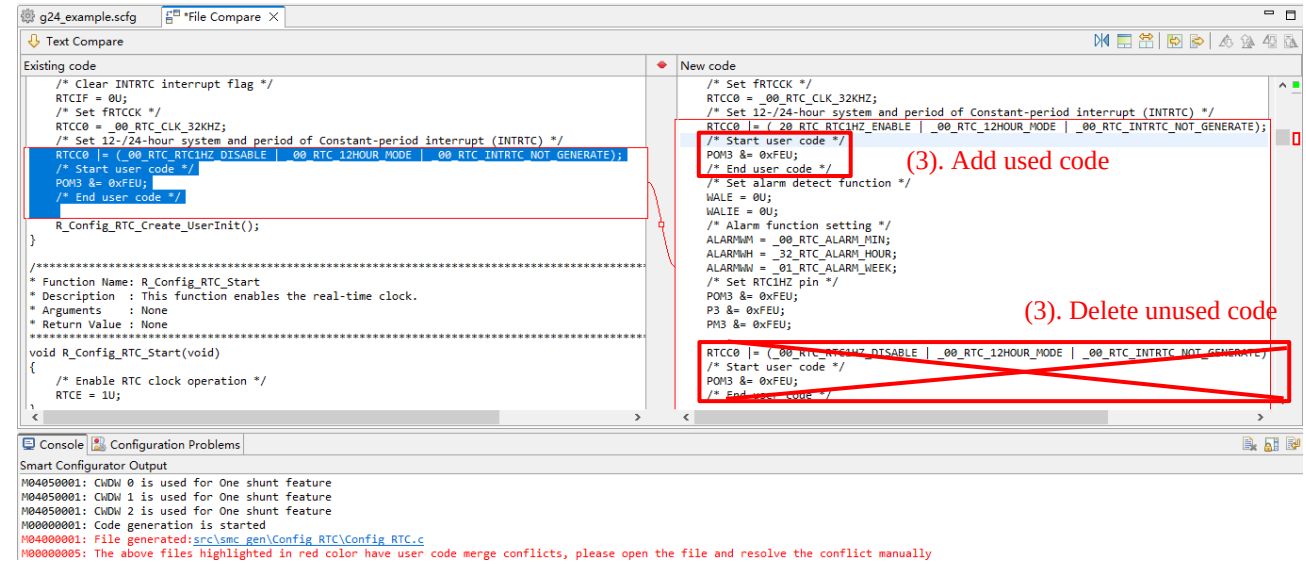



**Figure 11-5 Code after Applying "Copy Current Change from Left to Right"**

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

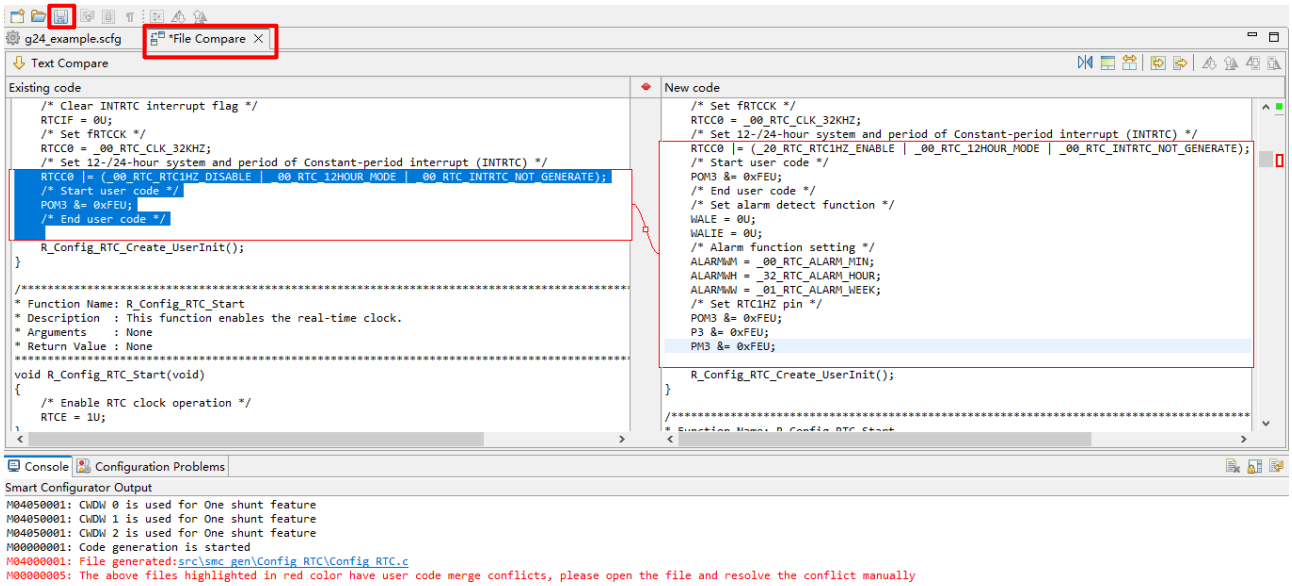



**Figure 11-6 Code after Deleting and Saving**

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

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

# 12. Help

Refer to the help system for detailed information on the Smart Configurator by clicking the [Help Contents] menu..

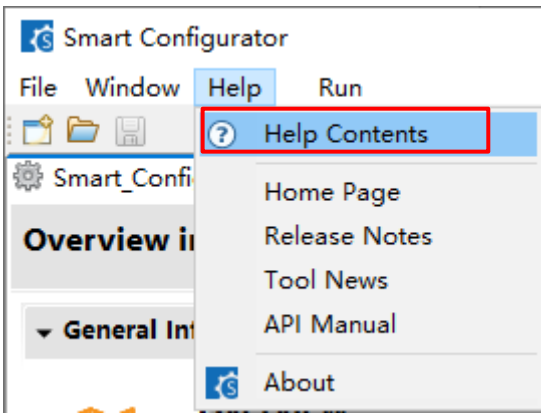



**Figure 12-1 Help Menu**

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

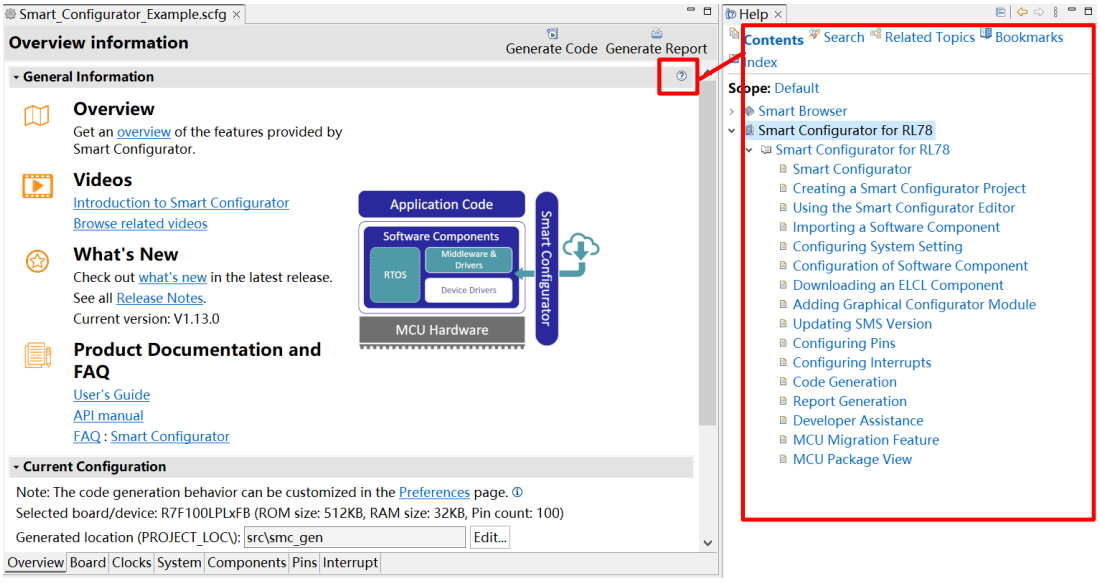



**Figure 12-2 Quick Start**

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

# 13. Documents for Reference

User's Manual: Hardware

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

Technical Update/Technical News

Obtain the latest information from the website of Renesas Electronics.

User's Manual: Development Environment

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

Obtain the latest version of IAR Embedded Workbench for Renesas RL78 manual from the website of IAR.

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 Installing and Uninstalling the Smart Configurator | Removed 2.1 Creating a New IAR C Project File |
| | Section 3 Operating the Smart Configurator | 3.1 Procedure for Operations: Updated Figure 3-1. Operating Procedure |
| | | 3.3 Creating and Loading a Configuration File: description was updated |
| | Section 4 Setting of Peripheral Modules | 4.4.13 Export Component Configuration: description was updated |
| | | 4.4.14 Import Component Configuration: description was updated |
| | | 4.4.15 Configure General Setting of Component: description was updated |
| | Section 6 Generating Source Code | 6.2 Configuration of Generated Files and File Names: description was updated |
| | Section 7 Loading Generated Files in Integrated Development Environment | 7. Loading Generated Files in Integrated Development Environment: description was updated |
| 1.02 | Section 4 Setting of Peripheral Modules | 4.4.12 Changing Version of BSP Configuration: Note was deleted. |
| | | 4.4.15 Configure General Setting of Component: Figure 4-38 Configure General Setting of Component was updated. |
| | | 4.4.15 Configure General Setting of Component: Note 1 was updated. |
| | | 4.4.15 Configure General Setting of Component: Note 2 was updated. |
| | | 4.4.15 Configure General Setting of Component: Note 3 was added. |
| | | 4.6.2 Changing Interrupt Bank Setting:<br>The description of step (3) was updated. |
| | | 4.6.2 Changing Interrupt Bank Setting:<br>Figure 4-55. Change Interrupt Bank Setting Example was modified. |
| 1.03 | Section 3 Operating the Smart Configurator | 3.4.4 MCU/MPU Package View: Update description and Figure 3-7. 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 4-50 and 4-51. |
| | | 4.5.3 Assigning Pins Using the MCU/MPU Package View: Update description and Figure 4-55. |
| | | Add 4.5.4 Show pin number from pin functions. |
| | | Add 4.5.9 Pin Errors/Warnings setting. |
| | Section 6 Generating Source Code | 6.2 Configuration of Generated Files and File Names: Update the description and Figure 6-3 Configuration of Generated Files and File Names for supporting pin symbol. |
| | Section 8 Creating User Programs | Add 8.2 Using Generated Code in User Application |
| | 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. |

| Rev. | Section | Description |
|---|---|---|
| 1.04 | Section 4 Setting of Peripheral Modules | Update 4.4.8 ELCL Fixed Function Modules Download and 4.4.9 Setting a Fixed Function ELCL Component |
| | | Add 4.4.10 Create and Edit ELCL Flexible Circuit |
| | Section 6 Generating Source Code | Update 6.2 Configuration of Generated Files and File Names |

# 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/.