RH850 Smart Configurator

User's Guide: e² studio

R20AN0739EC0100
Rev.1.00
Jan.22.24

Introduction

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

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

- e² studio 2024-01 and later

Target Devices and Compilers

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

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

Contents

1. Overview

1.1 Purpose

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

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

1.2 Features

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

2. Creating a Project

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

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

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

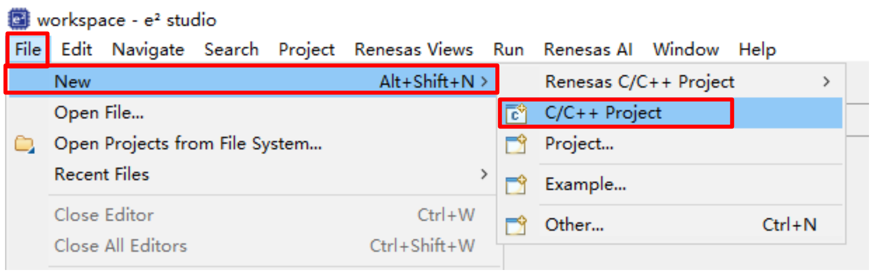


Figure 2-1 Creating a New Project

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

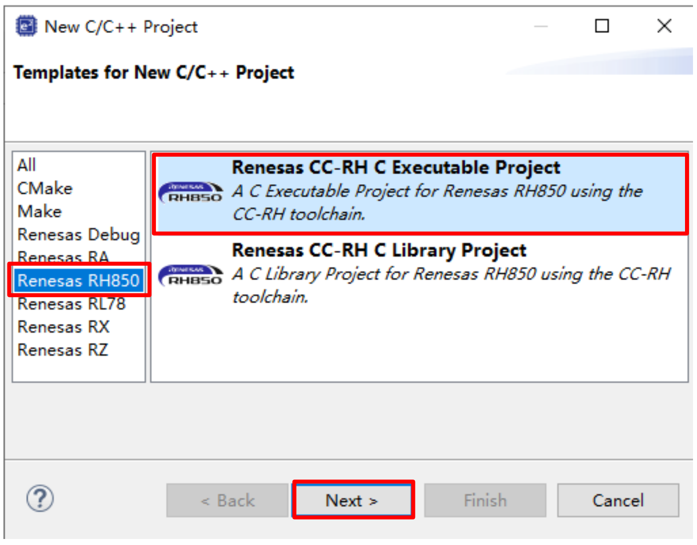
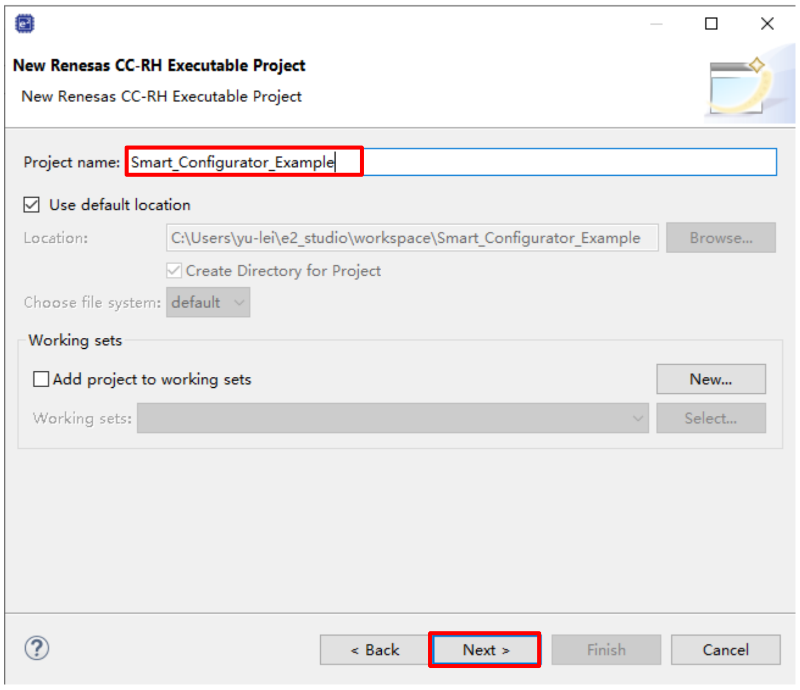

New C/C++ Project
Templates for New C/C++ Project
All
CMake
Make
Renesas Debug
Renesas RA
Renesas RH850
Renesas RL78
Renesas RX
Renesas RZ
Renesas CC-RH C Executable Project
A C Executable Project for Renesas RH850 using the CC-RH toolchain.
Renesas CC-RH C Library Project
A C Library Project for Renesas RH850 using the CC-RH toolchain.
< Back
Next >
Finish
Cancel

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

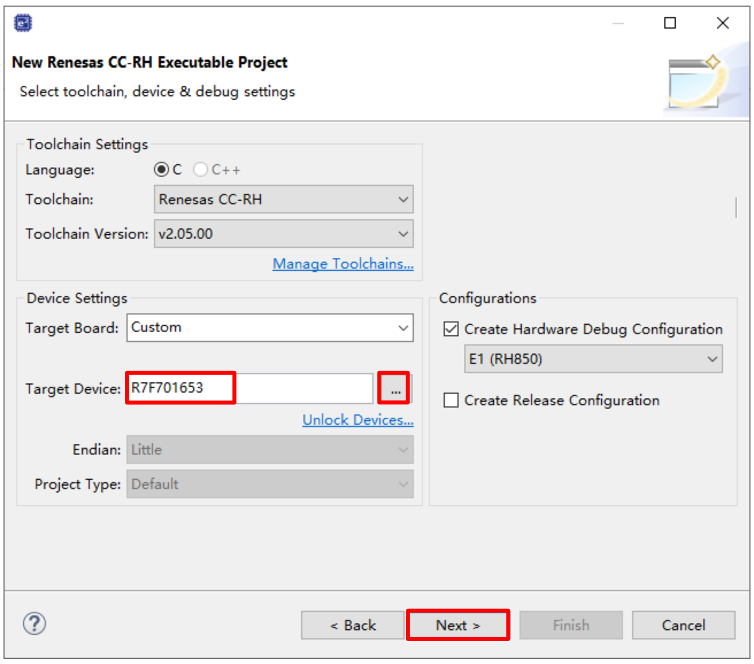
(3) Enter project information. Click on the [Next] button to continue.
(For e.g., CC-RH executable project, Project name: "Smart_Configurator_Example")


New Renesas CC-RH Executable Project
New Renesas CC-RH Executable Project

Project name: Smart_Configurator_Example
☑ Use default location
Location: C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example
☑ Create Directory for Project
Choose file system: default
Working sets
☐ Add project to working sets
Working sets:

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

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

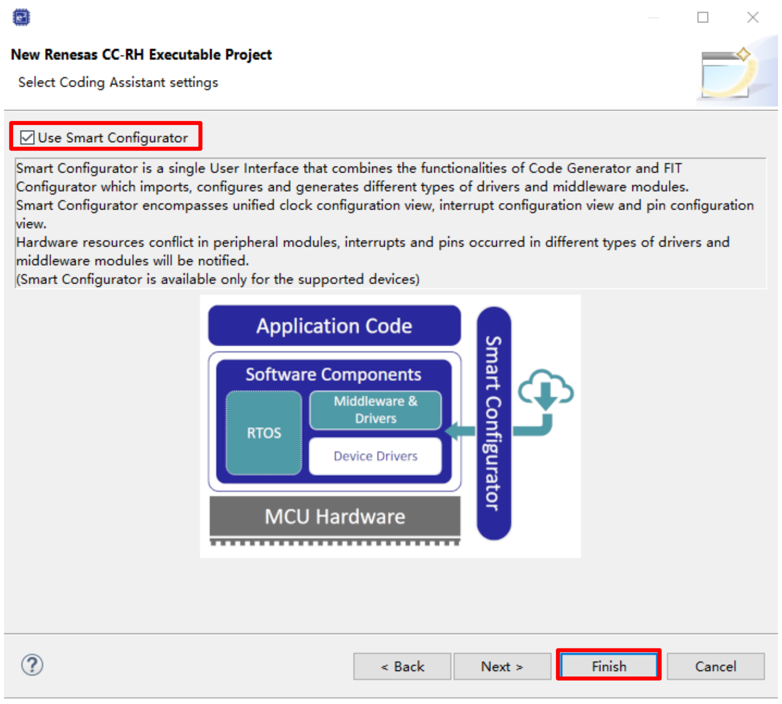

New Renesas CC-RH Executable Project
Select toolchain, device & debug settings

Toolchain Settings
Language: ● C ○ C++
Toolchain: Renesas CC-RH
Toolchain Version: v2.05.00
Device Settings
Target Board: Custom
Target Device: R7F701653
Endian: Little
Project Type: Default
Configurations
☑ Create Hardware Debug Configuration
E1 (RH850)
☐ Create Release Configuration

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

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

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



New Renesas CC-RH Executable Project

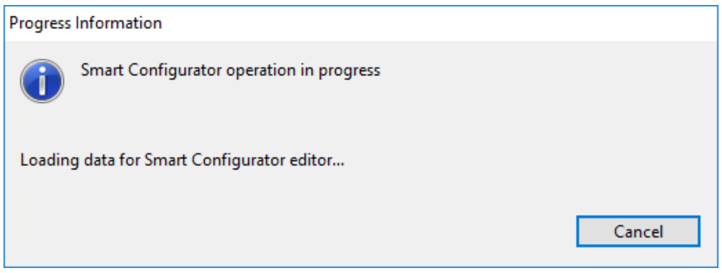
Select Coding Assistant settings

☑ Use Smart Configurator

Smart Configurator is a single User Interface that combines the functionalities of Code Generator and FIT Configurator which imports, configures and generates different types of drivers and middleware modules.
Smart Configurator encompasses unified clock configuration view, interrupt configuration view and pin configuration view.
Hardware resources conflict in peripheral modules, interrupts and pins occurred in different types of drivers and middleware modules will be notified.
(Smart Configurator is available only for the supported devices)

Figure 2-5 Selecting the Coding Assistant Tool

(6) Wait for completion of project creation.



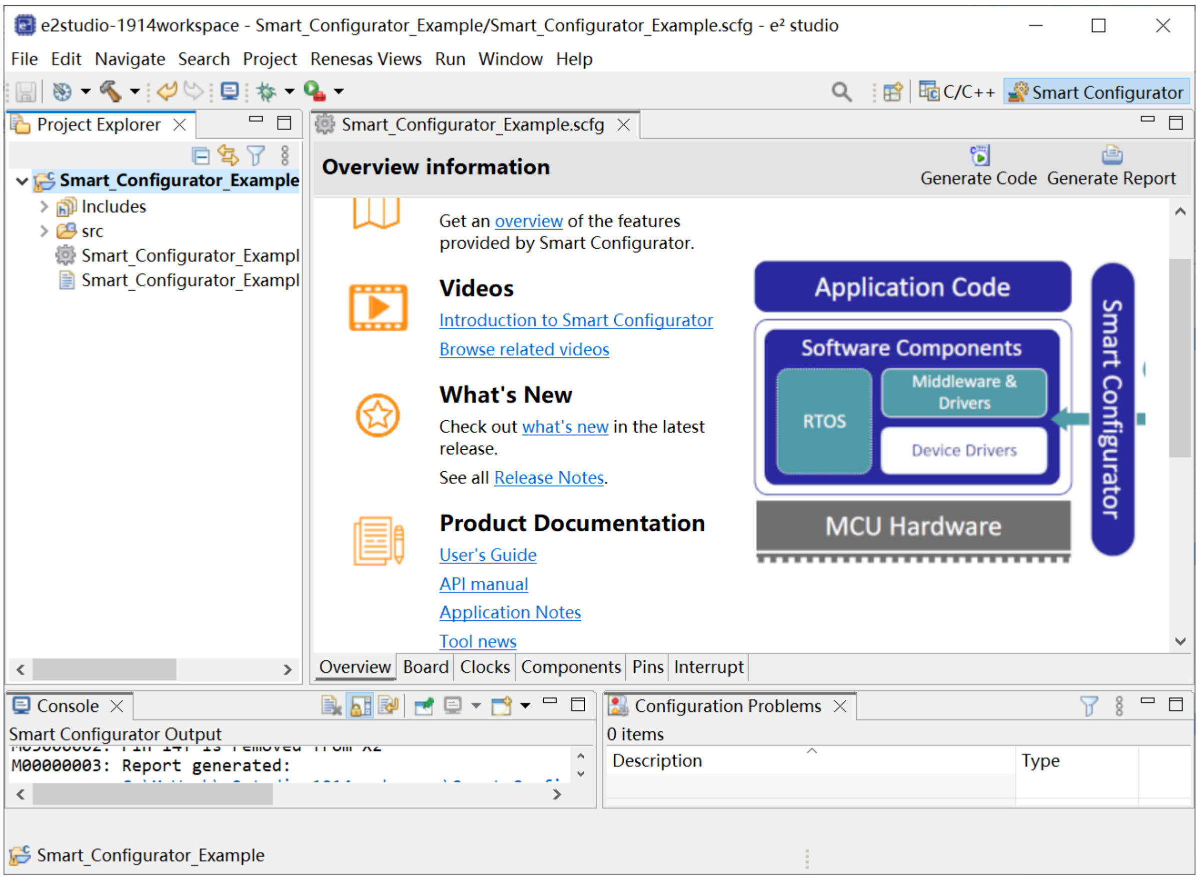
Progress Information

Smart Configurator operation in progress

Loading data for Smart Configurator editor...

Figure 2-6 Progress of Project Creation

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


e2studio-1914workspace - Smart_Configurator_Example/Smart_Configurator_Example.scfg - e² studio
File Edit Navigate Search Project Renesas Views Run Window Help
C/C++
Smart Configurator
Project Explorer
Smart_Configurator_Example
Includes
src
Smart_Configurator_Exampl
Smart_Configurator_Exampl
Smart_Configurator_Example.scfg
Overview information
Generate Code
Generate Report
Get an overview of the features provided by Smart Configurator.
Videos
Introduction to Smart Configurator
Browse related videos
What's New
Check out what's new in the latest release.
See all Release Notes.
Product Documentation
User's Guide
API manual
Application Notes
Tool news
Application Code
Software Components
Middleware & Drivers
RTOS
Device Drivers
MCU Hardware
Smart Configurator
Overview Board Clocks Components Pins Interrupt
Console
Smart Configurator Output
Configuration Problems
0 items
Description
Type
Smart_Configurator_Example

```
M00000003: Report generated:
```

Figure 2-7 Smart Configurator Perspective

3. Operating the Smart Configurator

3.1 Displaying the Smart Configurator Perspective

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

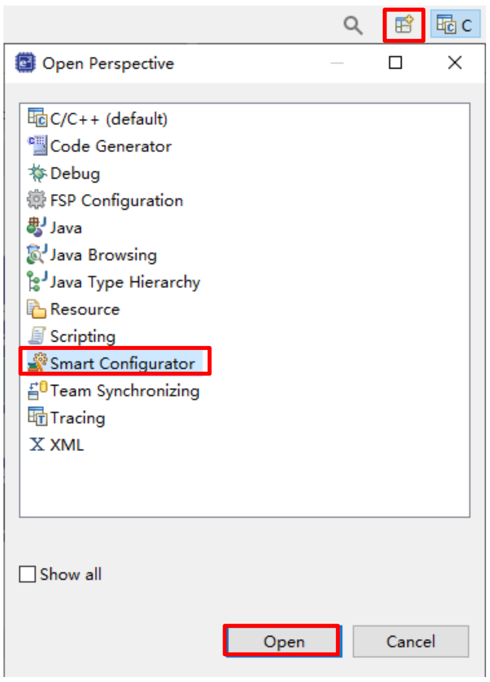


Figure 3-1 Opening the Smart Configurator Perspective

3.2 Procedure for Operations

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

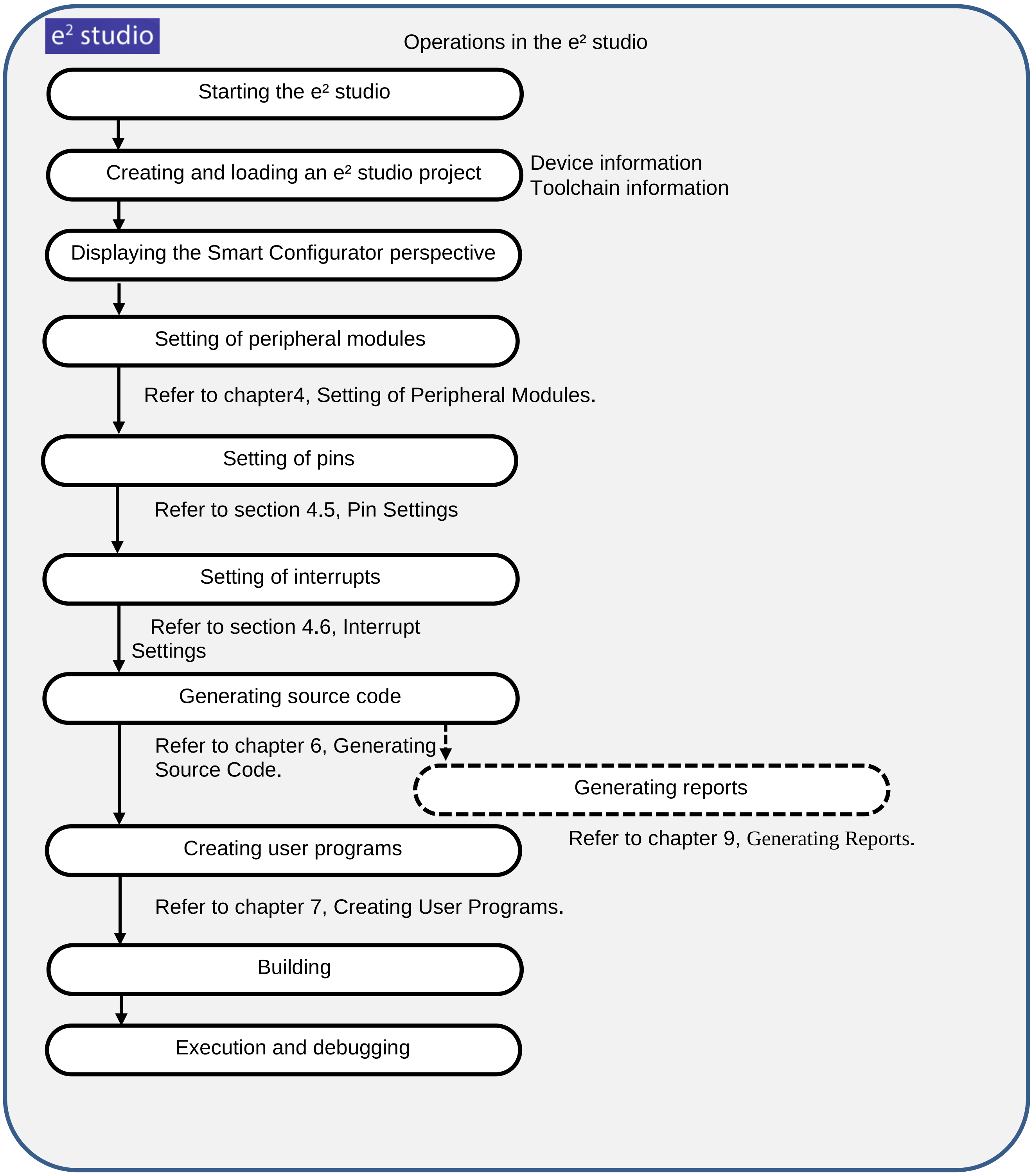

e² studio
Operations in the e² studio
Starting the e² studio
Creating and loading an e² studio project
Device information
Toolchain information
Displaying the Smart Configurator perspective
Setting of peripheral modules
Refer to chapter4, Setting of Peripheral Modules.
Setting of pins
Refer to section 4.5, Pin Settings
Setting of interrupts
Refer to section 4.6, Interrupt Settings
Generating source code
Refer to chapter 6, Generating Source Code.
Generating reports
Refer to chapter 9, Generating Reports.
Creating user programs
Refer to chapter 7, Creating User Programs.
Building
Execution and debugging

Figure 3-2 Procedure for Operations

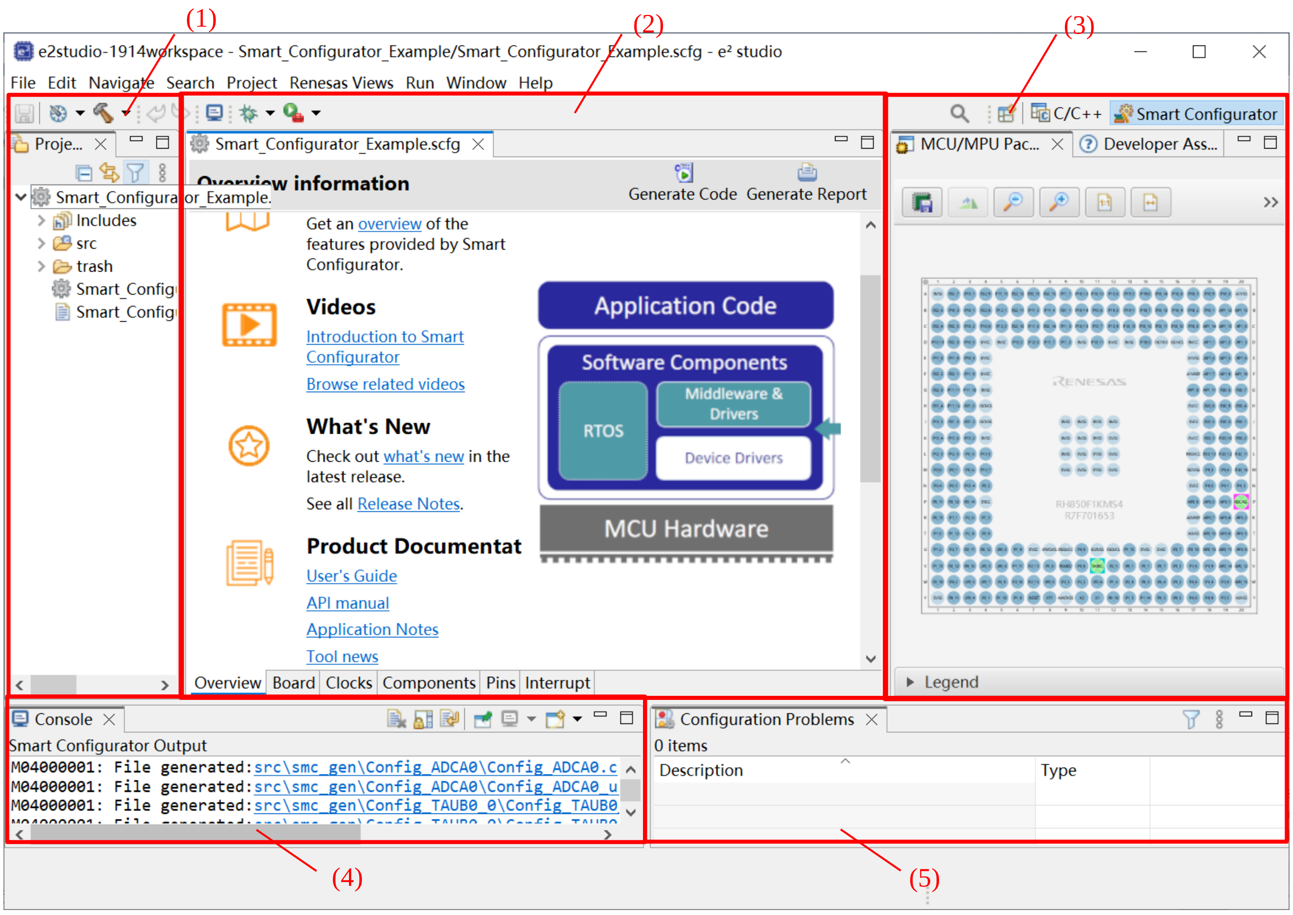
3.3 File to be Saved as Project Information

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

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

3.4 Window

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



Get an overview of the features provided by Smart Configurator.

Check out what's new in the latest release.

See all Release Notes.

```
M04000001: File generated:src\smc_gen\Config_ADCA0\Config_ADCA0.c
M04000001: File generated:src\smc_gen\Config_ADCA0\Config_ADCA0_u
M04000001: File generated:src\smc_gen\Config_TAUB0_0\Config_TAUB0
```

Figure 3-3 Smart Configurator Perspective

(1) Project Explorer

(2) Smart Configurator view

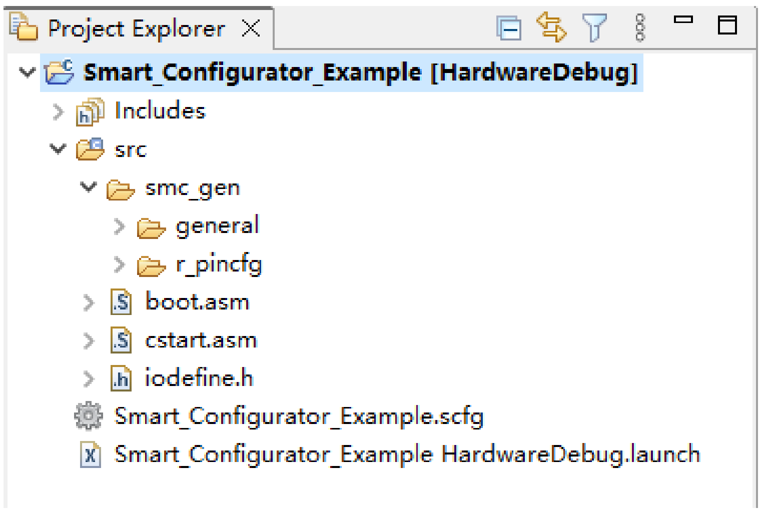
(3) MCU/MPU Package view

(4) Console view

(5) Configuration Problems view

3.4.1 Project Explorer

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



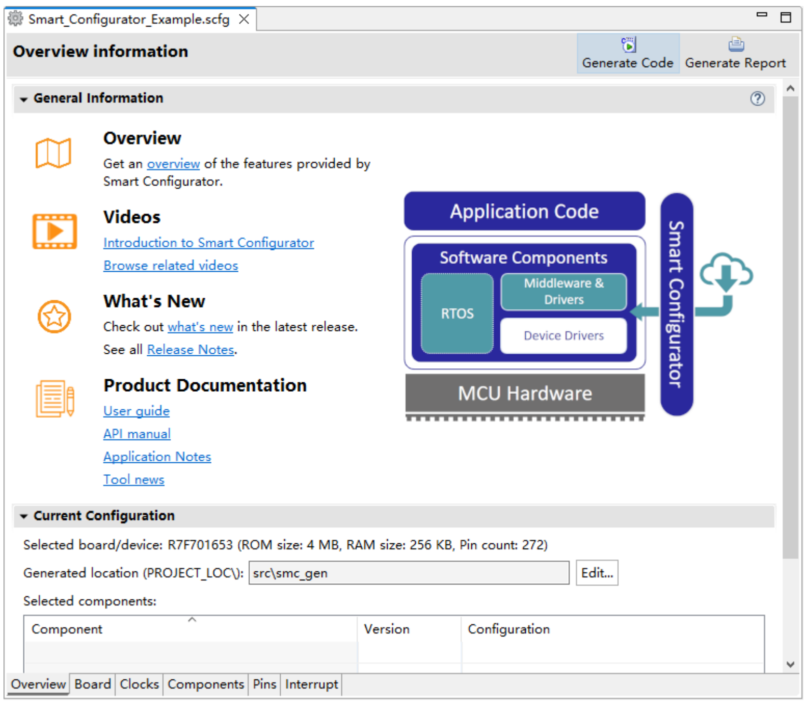
```
Smart_Configurator_Example [HardwareDebug]
  Includes
  src
    smc_gen
      general
      r_pincfg
    boot.asm
    cstart.asm
    iodefine.h
  Smart_Configurator_Example.scfg
  Smart_Configurator_Example HardwareDebug.launch
```

Figure 3-4 Project Explorer

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

3.4.2 Smart Configurator View

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



Overview information

General Information

Overview
Get an overview of the features provided by Smart Configurator.

Videos
Introduction to Smart Configurator
Browse related videos

What's New
Check out what's new in the latest release.
See all Release Notes.

Product Documentation
User guide
API manual
Application Notes
Tool news

Current Configuration
Selected board/device: R7F701653 (ROM size: 4 MB, RAM size: 256 KB, Pin count: 272)
Generated location (PROJECT_LOC\): src\smc_gen Edit...
Selected components:

Component	Version	Configuration

Figure 3-5 Smart Configurator View

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

3.4.3 MCU/MPU Package View

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

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

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

The initial pin setting information of the board is the pin information of the board selected by [Board:] on the [Board] page (refer to "chapter **4.1** Board Settings” and “chapter **4.5.6** Pin Setting Using Board Pin Configuration Information**).**

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

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

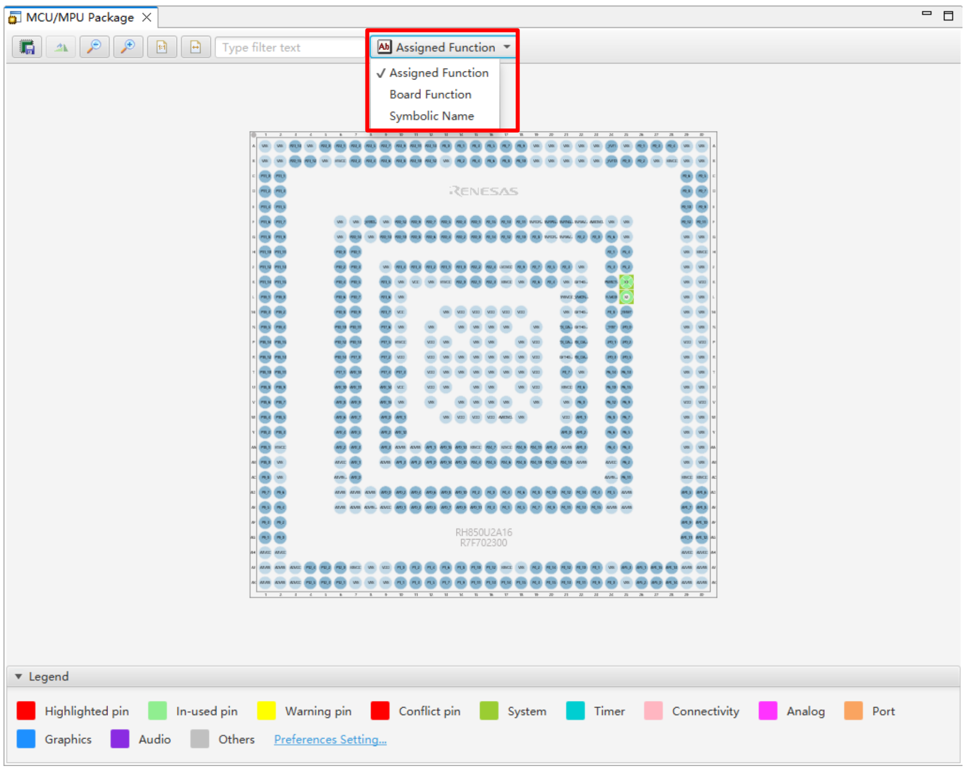
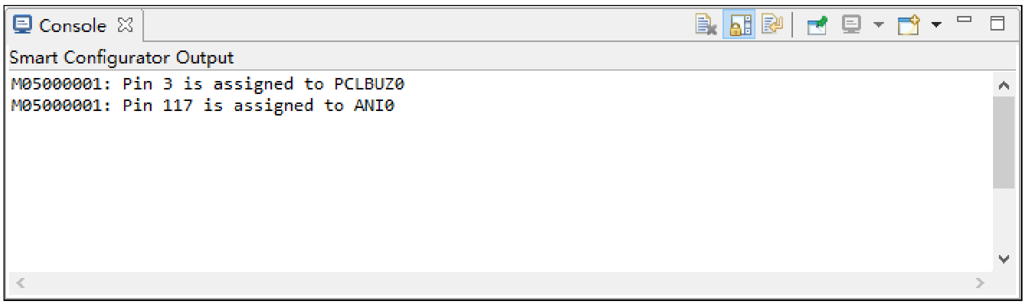


Figure 3-6 MCU/MPU Package View

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

3.4.4 Console View

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



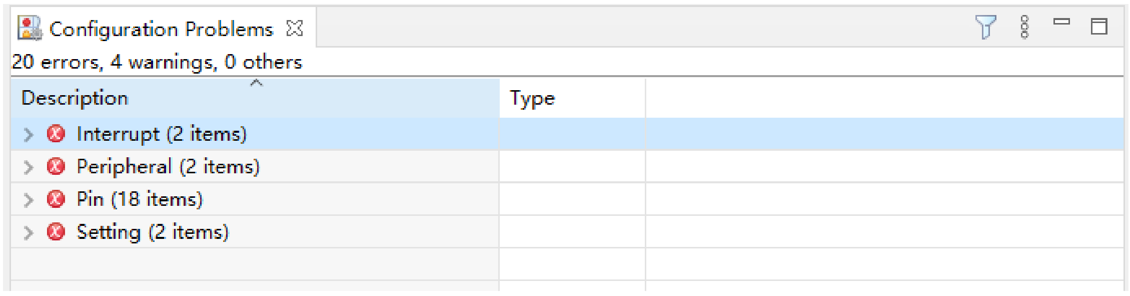
```
M05000001: Pin 3 is assigned to PCLBUZ0
M05000001: Pin 117 is assigned to ANI0
```

Figure 3-7 Console View

When this view is not opened, select [Window]→ [Show View] → [Other] from the e^2 studio menu and select [General] → [Console] on the opened [Show View] dialog box.

3.4.5 Configuration Problems View

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



20 errors, 4 warnings, 0 others

Description	Type	
Interrupt (2 items)		
Peripheral (2 items)		
Pin (18 items)		
Setting (2 items)		

Figure 3-8 Configuration Problems View

When this view is not opened, select [Renesas Views] → [Smart Configurator] → [Configuration Problems] from the e^2 studio menu.

4. Setting of Peripheral Modules

User can select peripheral modules from the Smart Configurator view.

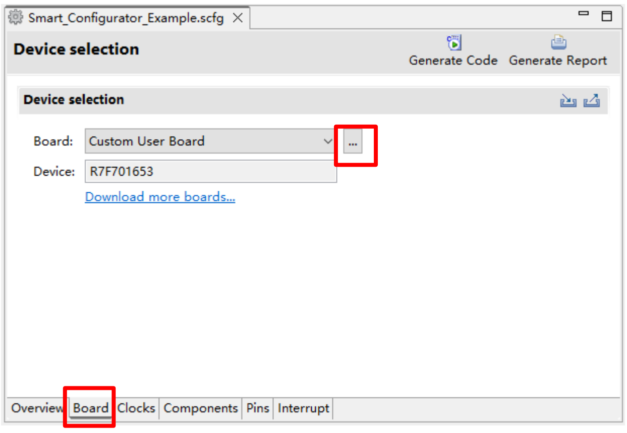
4.1 Board Settings

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

4.1.1 Selecting the Device

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

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



Board: Custom User Board

Device: R7F701653

Figure 4-1 Selecting the Device

4.1.2 Selecting the Board

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

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

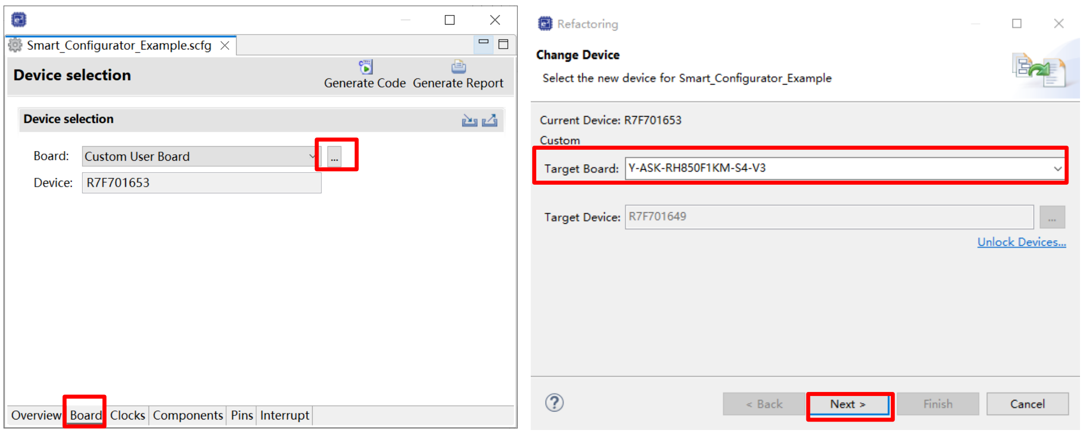
- Pin assignment (Initial pin setting)
- Frequency of the main clock
- Frequency of the sub-clock
- Target device

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

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

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

Depending on the board selected, the device will change, device change is reflected to the target device of e^2 studio project. It is the same with the procedure of " chapter 4.7 MCU Migration Feature".


Smart_Configurator_Example.scfg

Device selection

Board: Custom User Board

Device: R7F701653

Refactoring

Change Device

Select the new device for Smart_Configurator_Example

Current Device: R7F701653

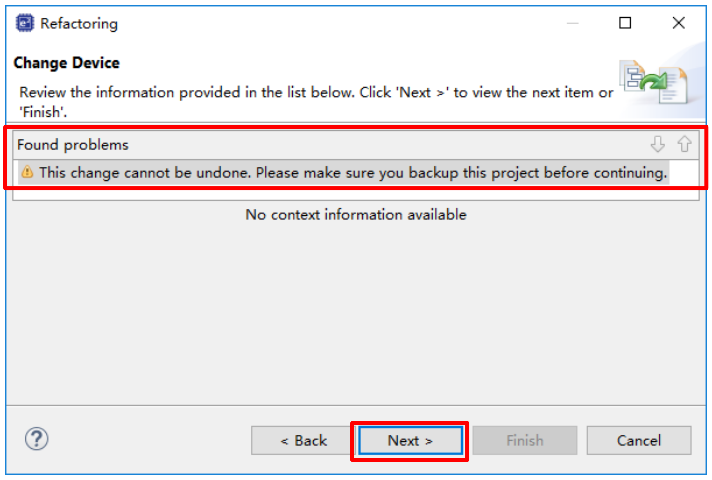
Custom

Target Board: Y-ASK-RH850F1KM-S4-V3

Target Device: R7F701649

Figure 4-2 Selecting the Board

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


Refactoring

Change Device

Review the information provided in the list below. Click 'Next >' to view the next item or 'Finish'.

Found problems

This change cannot be undone. Please make sure you backup this project before continuing.

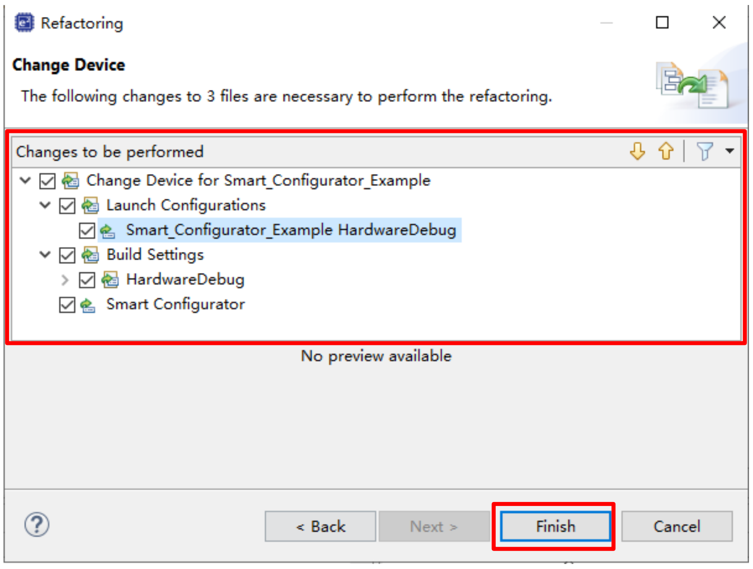
No context information available

Figure 4-3 Found Problems

Table 4-1 Change Device Found Problems List

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

Confirm items to be changed and click "Finish".



Refactoring

Change Device

The following changes to 3 files are necessary to perform the refactoring.

Changes to be performed

- ☑ Change Device for Smart_Configurator_Example
  - ☑ Launch Configurations
    - ☑ Smart_Configurator_Example HardwareDebug
  - ☑ Build Settings
    - ☑ HardwareDebug
  - ☑ Smart Configurator

No preview available

Figure 4-4 Changes to be Performed

4.1.3 Exporting Board Settings

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

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

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

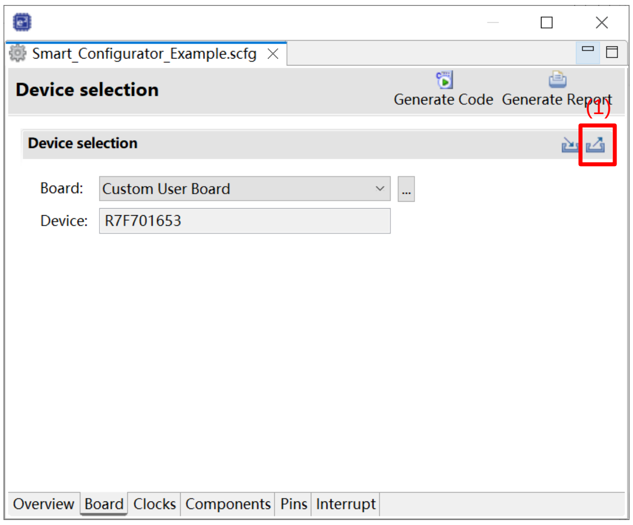


Figure 4-5 Exporting Board Settings (bdf Format)

4.1.4 Importing Board Settings

Follow the procedure below to import board settings.

(1) Click on the [(Import board setting)] button and select a desired bdf file.

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

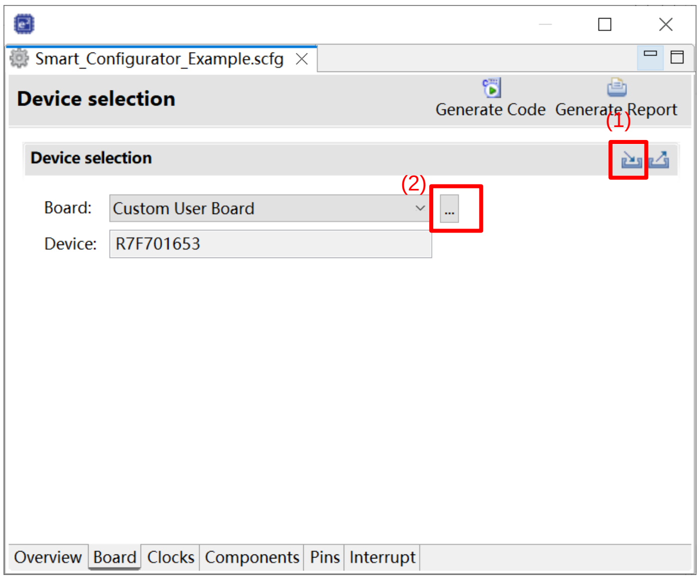


Figure 4-6 Importing Board Settings (bdf Format)

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

4.2 Clock Settings

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

Follow the procedure below to modify the clock settings.

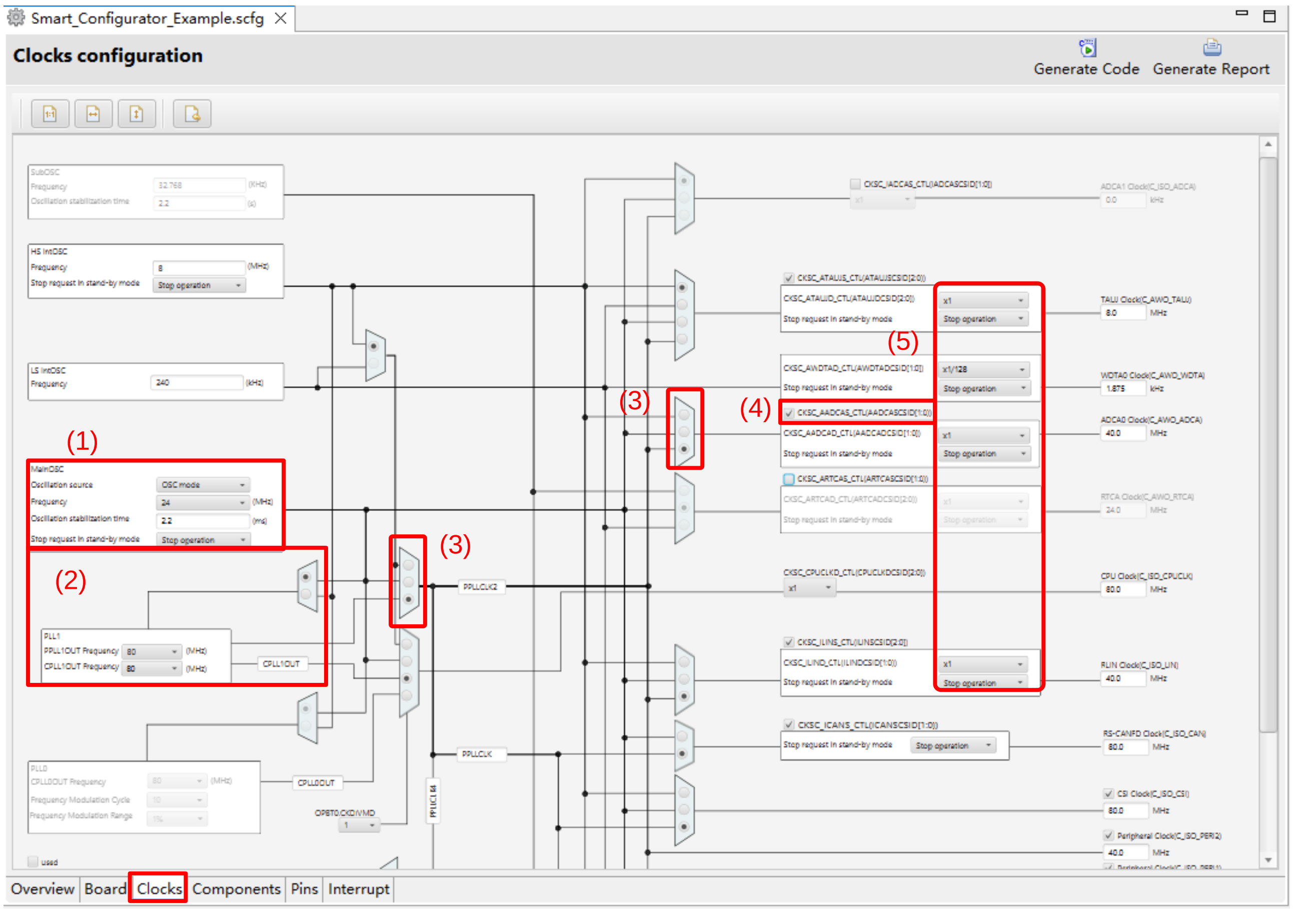
(1) Specify the frequency of each clock in accordance with the board specifications (Note that the frequency is fixed for some internal clocks).

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

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

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

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



Clocks configuration

SubOSC
Frequency 32.768 (KHz)
Oscillation stabilization time 2.2 (s)

HS IntOSC
Frequency 8 (MHz)
Stop request in stand-by mode Stop operation

LS IntOSC
Frequency 240 (kHz)

MainOSC
Oscillation source OSC mode
Frequency 24 (MHz)
Oscillation stabilization time 2.2 (ms)
Stop request in stand-by mode Stop operation

PLL1
PPLL1OUT Frequency 80 (MHz)
CPLL1OUT Frequency 80 (MHz)

PLL0
CPLL0OUT Frequency 80 (MHz)
Frequency Modulation Cycle 10
Frequency Modulation Range 1%

OPBT0.CKDIVMD 1

used

CKSC_IADCAS_CTL(IADCASCSID[1:0]) x1 — ADCA1 Clock(C_ISO_ADCA) 0.0 kHz

✓ CKSC_ATAUJS_CTL(ATAUJSCSID[2:0])
CKSC_ATAUJD_CTL(ATAUJDCSID[2:0]) x1
Stop request in stand-by mode Stop operation
TAUJ Clock(C_AWO_TAUJ) 8.0 MHz

CKSC_AWDTAD_CTL(AWDTADCSID[1:0]) x1/128
Stop request in stand-by mode Stop operation
WDTA0 Clock(C_AWO_WDTA) 1.875 kHz

✓ CKSC_AADCAS_CTL(AADCASCSID[1:0])
CKSC_AADCAD_CTL(AADCADCSID[1:0]) x1
Stop request in stand-by mode Stop operation
ADCA0 Clock(C_AWO_ADCA) 40.0 MHz

CKSC_ARTCAS_CTL(ARTCASCSID[1:0])
CKSC_ARTCAD_CTL(ARTCADCSID[2:0]) x1
Stop request in stand-by mode Stop operation
RTCA Clock(C_AWO_RTCA) 24.0 MHz

CKSC_CPUCLKD_CTL(CPUCLKDCSID[2:0]) x1
CPU Clock(C_ISO_CPUCLK) 80.0 MHz

✓ CKSC_ILINS_CTL(ILINSCSID[2:0])
CKSC_ILIND_CTL(ILINDCSID[1:0]) x1
Stop request in stand-by mode Stop operation
RLIN Clock(C_ISO_LIN) 40.0 MHz

✓ CKSC_ICANS_CTL(ICANSCSID[1:0])
Stop request in stand-by mode Stop operation
RS-CANFD Clock(C_ISO_CAN) 80.0 MHz

✓ CSI Clock(C_ISO_CSI) 80.0 MHz

✓ Peripheral Clock(C_ISO_PERI2) 40.0 MHz

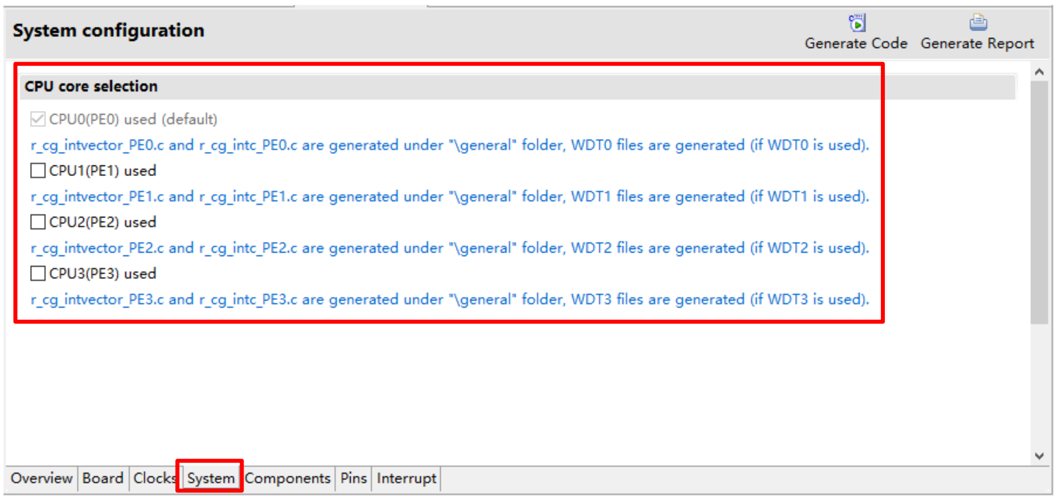
Figure 4-7 [Clocks] Page

4.3 System Settings (only for RH850/U2A)

User can select the CPUn (PE*n*) to be used at [System] tabbed page.

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

Only RH850/U2A supports System settings.

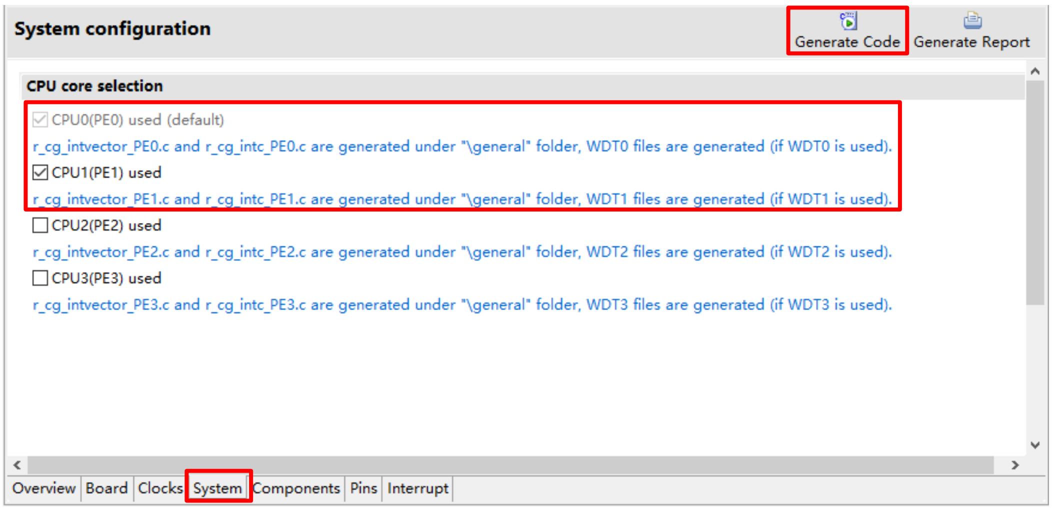


☑ CPU0(PE0) used (default)
r_cg_intvector_PE0.c and r_cg_intc_PE0.c are generated under "\general" folder, WDT0 files are generated (if WDT0 is used).
☐ CPU1(PE1) used
r_cg_intvector_PE1.c and r_cg_intc_PE1.c are generated under "\general" folder, WDT1 files are generated (if WDT1 is used).
☐ CPU2(PE2) used
r_cg_intvector_PE2.c and r_cg_intc_PE2.c are generated under "\general" folder, WDT2 files are generated (if WDT2 is used).
☐ CPU3(PE3) used
r_cg_intvector_PE3.c and r_cg_intc_PE3.c are generated under "\general" folder, WDT3 files are generated (if WDT3 is used).

Figure 4-8 [System] Page

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

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



☑ CPU0(PE0) used (default)
r_cg_intvector_PE0.c and r_cg_intc_PE0.c are generated under "\general" folder, WDT0 files are generated (if WDT0 is used).
☑ CPU1(PE1) used
r_cg_intvector_PE1.c and r_cg_intc_PE1.c are generated under "\general" folder, WDT1 files are generated (if WDT1 is used).
☐ CPU2(PE2) used
r_cg_intvector_PE2.c and r_cg_intc_PE2.c are generated under "\general" folder, WDT2 files are generated (if WDT2 is used).
☐ CPU3(PE3) used
r_cg_intvector_PE3.c and r_cg_intc_PE3.c are generated under "\general" folder, WDT3 files are generated (if WDT3 is used).

Figure 4-9 Select CPU core usage setting

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

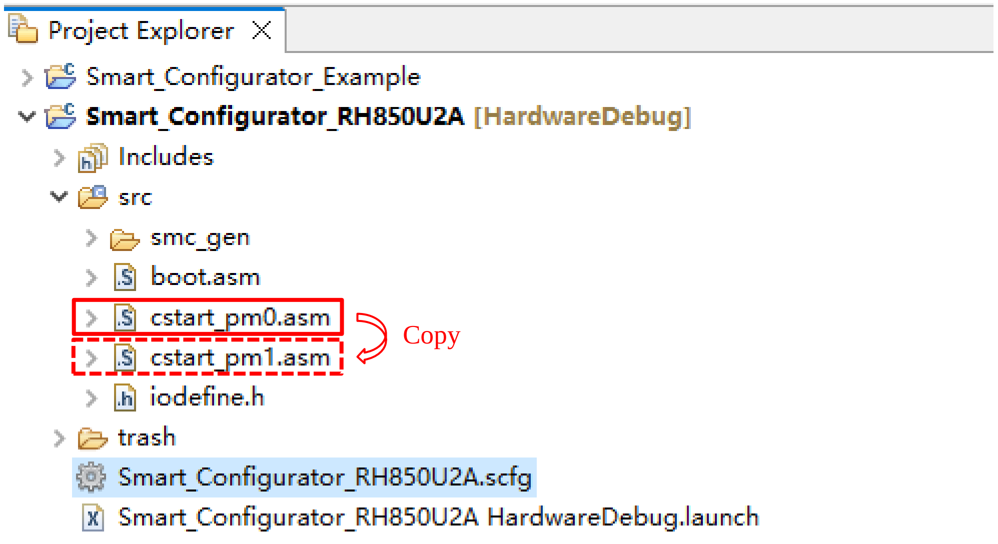
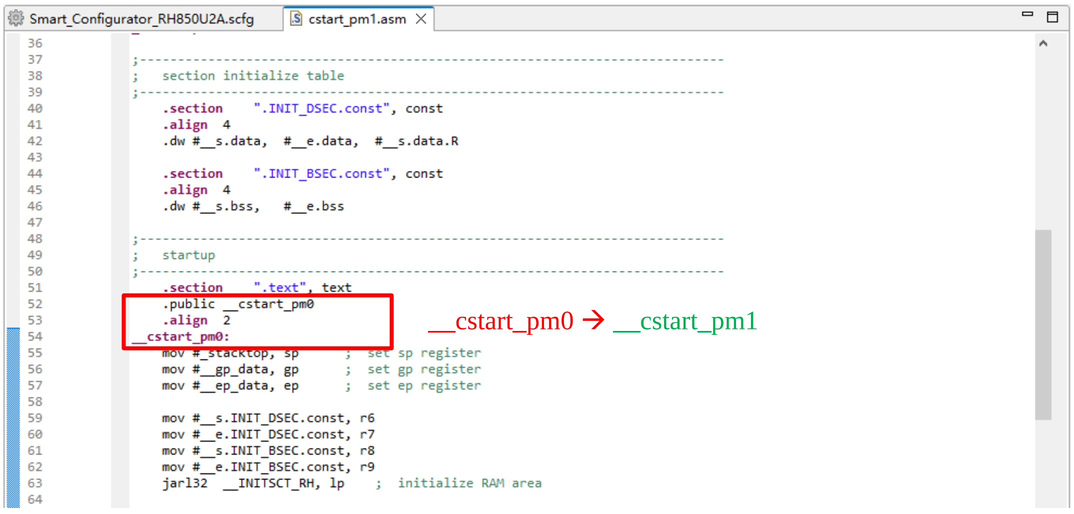

Project Explorer
Smart_Configurator_Example
Smart_Configurator_RH850U2A [HardwareDebug]
Includes
src
smc_gen
boot.asm
cstart_pm0.asm
cstart_pm1.asm
Copy
iodefine.h
trash
Smart_Configurator_RH850U2A.scfg
Smart_Configurator_RH850U2A HardwareDebug.launch

Figure 4-10 e2 studio copy cstart_pm0.asm file

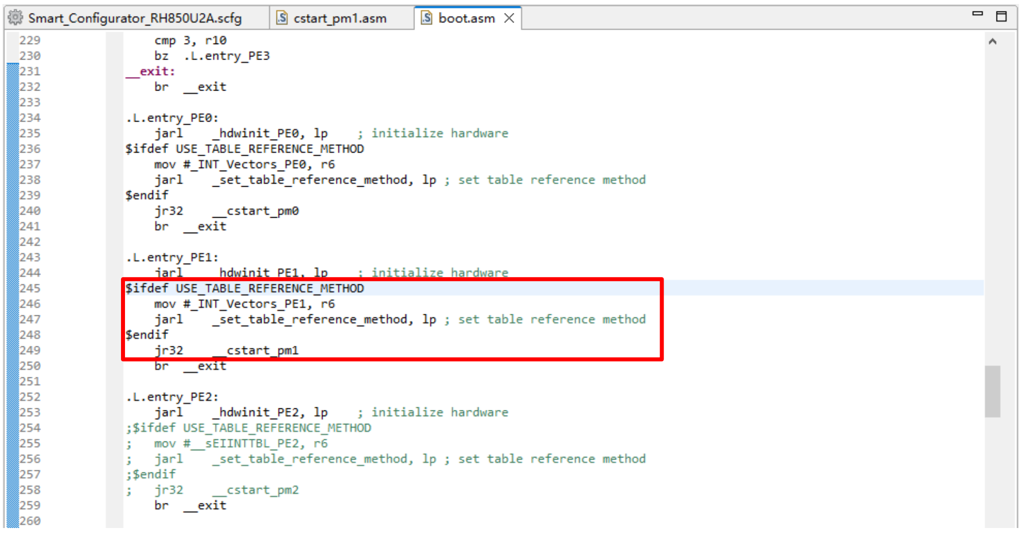
(3) Modify “cstart_pm1.asm” file contents as shown below.



```

;--------------------------------------------------------------------------------
;    section initialize table
;--------------------------------------------------------------------------------
    .section    ".INIT_DSEC.const", const
    .align  4
    .dw #__s.data,  #__e.data,  #__s.data.R

    .section    ".INIT_BSEC.const", const
    .align  4
    .dw #__s.bss,   #__e.bss

;--------------------------------------------------------------------------------
;    startup
;--------------------------------------------------------------------------------
    .section    ".text", text
    .public __cstart_pm0
    .align  2
__cstart_pm0:
    mov #_stacktop, sp       ;  set sp register
    mov #__gp_data, gp       ;  set gp register
    mov #__ep_data, ep       ;  set ep register

    mov #__s.INIT_DSEC.const, r6
    mov #__e.INIT_DSEC.const, r7
    mov #__s.INIT_BSEC.const, r8
    mov #__e.INIT_BSEC.const, r9
    jarl32  __INITSCT_RH, lp    ;  initialize RAM area

```

Figure 4-11 Modify cstart_pm1.asm file contents

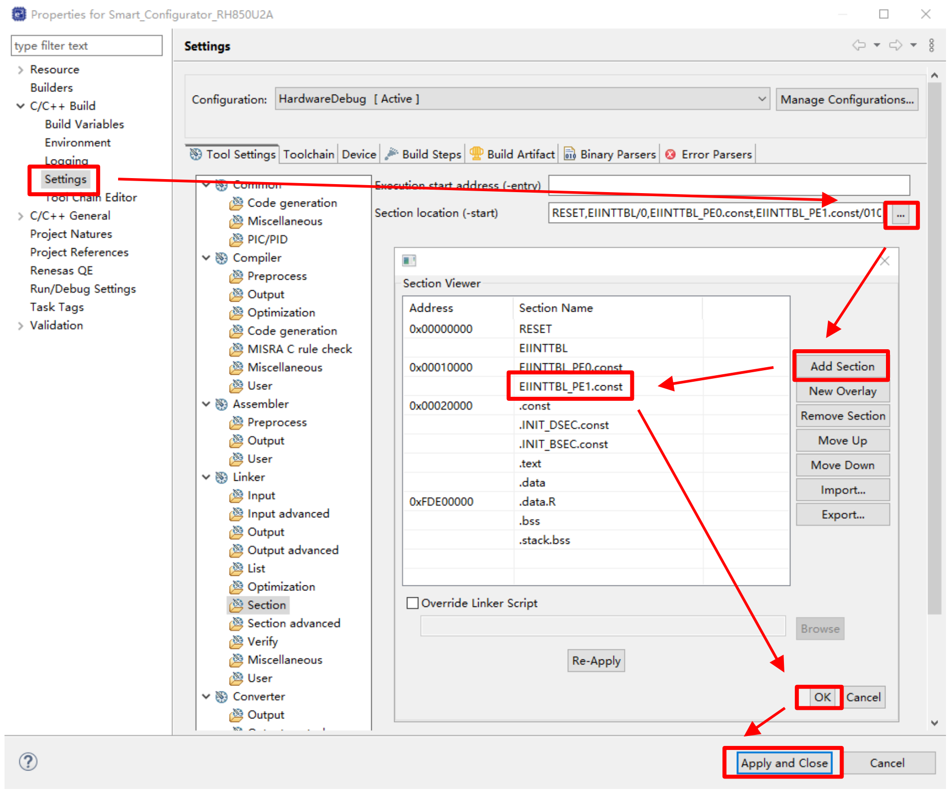
(4) Modify “boot.asm” file contents as shown below.



```
    cmp 3, r10
    bz  .L.entry_PE3
__exit:
    br  __exit

.L.entry_PE0:
    jarl    _hdwinit_PE0, lp    ; initialize hardware
$ifdef USE_TABLE_REFERENCE_METHOD
    mov #_INT_Vectors_PE0, r6
    jarl    _set_table_reference_method, lp ; set table reference method
$endif
    jr32    __cstart_pm0
    br  __exit

.L.entry_PE1:
    jarl    _hdwinit_PE1, lp    ; initialize hardware
$ifdef USE_TABLE_REFERENCE_METHOD
    mov #_INT_Vectors_PE1, r6
    jarl    _set_table_reference_method, lp ; set table reference method
$endif
    jr32    __cstart_pm1
    br  __exit

.L.entry_PE2:
    jarl    _hdwinit_PE2, lp    ; initialize hardware
;$ifdef USE_TABLE_REFERENCE_METHOD
;   mov #__sEIINTTBL_PE2, r6
;   jarl    _set_table_reference_method, lp ; set table reference method
;$endif
;   jr32    __cstart_pm2
    br  __exit
```

Figure 4-12 Modify boot.asm file contents

(5) Add section setting for PE1.



Configuration: HardwareDebug [Active]

Execution start address (-entry)

Section location (-start) RESET,EIINTTBL/0,EIINTTBL_PE0.const,EIINTTBL_PE1.const/010

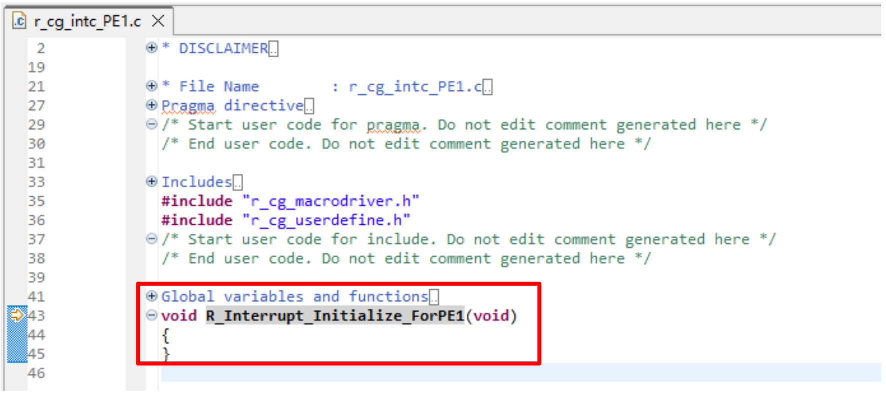
Section Viewer

Address	Section Name
0x00000000	RESET
	EIINTTBL
0x00010000	EIINTTBL_PE0.const
	EIINTTBL_PE1.const
0x00020000	.const
	.INIT_DSEC.const
	.INIT_BSEC.const
	.text
	.data
0xFDE00000	.data.R
	.bss
	.stack.bss

☐ Override Linker Script

Figure 4-13 Add section setting for PE1

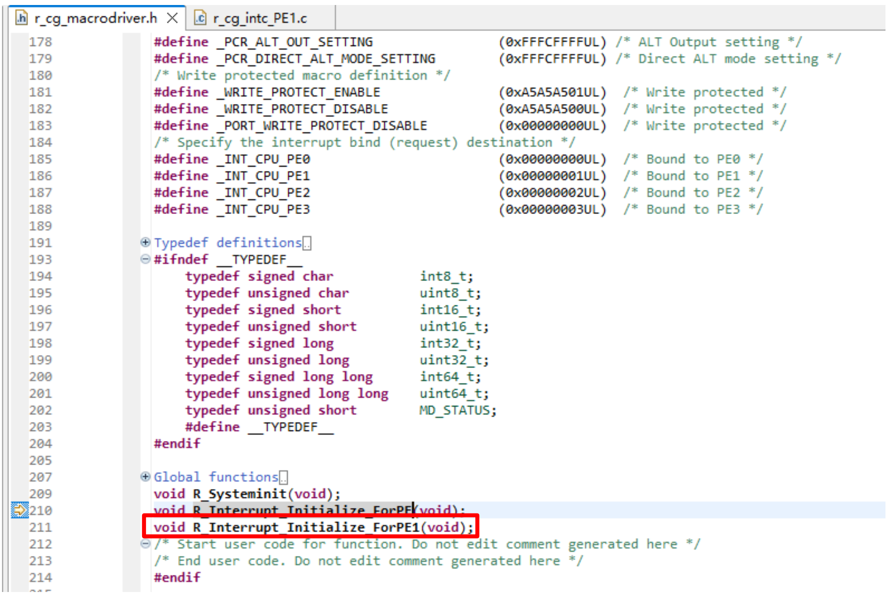
(6) Rename API “R_Interrupt_Initialize_ForPE” as “R_Interrupt_Initialize_ForPE1” in r_cg_intc_PE1.c file.


r_cg_intc_PE1.c

```
* DISCLAIMER

* File Name        : r_cg_intc_PE1.c
Pragma directive
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

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

Global variables and functions
void R_Interrupt_Initialize_ForPE1(void)
{
}
```

Figure 4-14 Rename R_Interrupt_Initialize_ForPE API

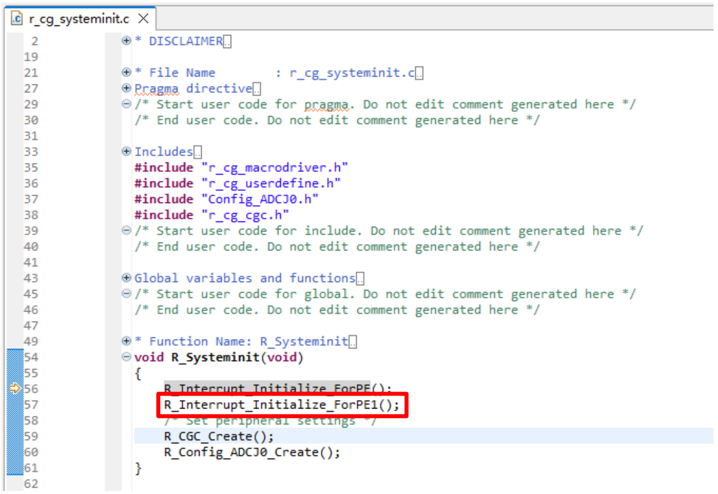
(7) Add declaration code for “R_Interrupt_Initialize_ForPE1” API.


r_cg_macrodriver.h r_cg_intc_PE1.c

```
#define _PCR_ALT_OUT_SETTING                (0xFFFCFFFFUL) /* ALT Output setting */
#define _PCR_DIRECT_ALT_MODE_SETTING        (0xFFFCFFFFUL) /* Direct ALT mode setting */
/* Write protected macro definition */
#define _WRITE_PROTECT_ENABLE               (0xA5A5A501UL)  /* Write protected */
#define _WRITE_PROTECT_DISABLE              (0xA5A5A500UL)  /* Write protected */
#define _PORT_WRITE_PROTECT_DISABLE         (0x00000000UL)  /* Write protected */
/* Specify the interrupt bind (request) destination */
#define _INT_CPU_PE0                        (0x00000000UL)  /* Bound to PE0 */
#define _INT_CPU_PE1                        (0x00000001UL)  /* Bound to PE1 */
#define _INT_CPU_PE2                        (0x00000002UL)  /* Bound to PE2 */
#define _INT_CPU_PE3                        (0x00000003UL)  /* Bound to PE3 */

Typedef definitions
#ifndef __TYPEDEF__
    typedef signed char           int8_t;
    typedef unsigned char         uint8_t;
    typedef signed short          int16_t;
    typedef unsigned short        uint16_t;
    typedef signed long           int32_t;
    typedef unsigned long         uint32_t;
    typedef signed long long      int64_t;
    typedef unsigned long long    uint64_t;
    typedef unsigned short        MD_STATUS;
    #define __TYPEDEF__
#endif

Global functions
void R_Systeminit(void);
void R_Interrupt_Initialize_ForPE(void);
void R_Interrupt_Initialize_ForPE1(void);
/* Start user code for function. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#endif
```

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

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


r_cg_systeminit.c

```
* DISCLAIMER

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

Includes
#include "r_cg_macrodriver.h"
#include "r_cg_userdefine.h"
#include "Config_ADCJ0.h"
#include "r_cg_cgc.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

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

* Function Name: R_Systeminit
void R_Systeminit(void)
{
    R_Interrupt_Initialize_ForPE();
    R_Interrupt_Initialize_ForPE1();
    /* Set peripheral settings */
    R_CGC_Create();
    R_Config_ADCJ0_Create();
}
```

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

4.4 Component Settings

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

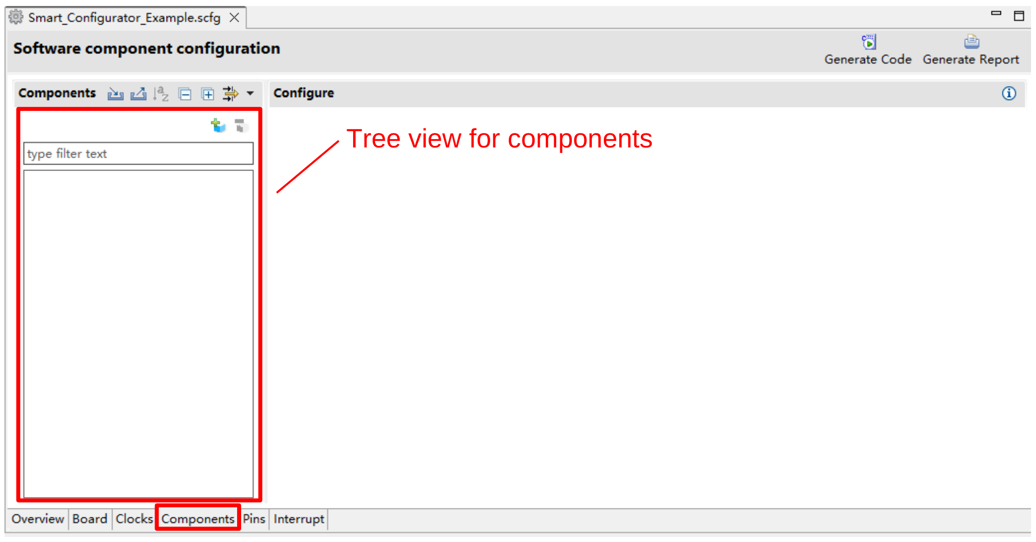


Figure 4-17 [Components] Page

4.4.1 Switching Between the Component View and Hardware View

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

(1) Click on the “Show by Component View” radio button. The tree view will display the components by component category.

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

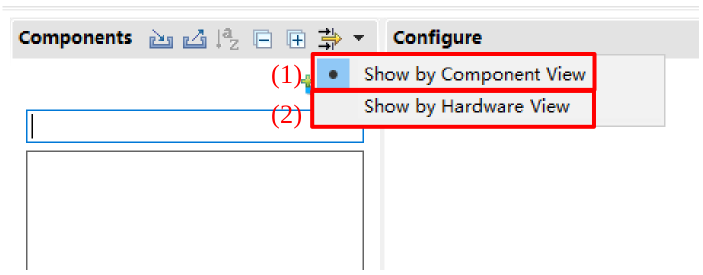


Figure 4-18 Switching to the Hardware View

4.4.2 Adding a Software Component

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

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

(b) On Hardware Tree, double-click on a hardware resource node

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

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

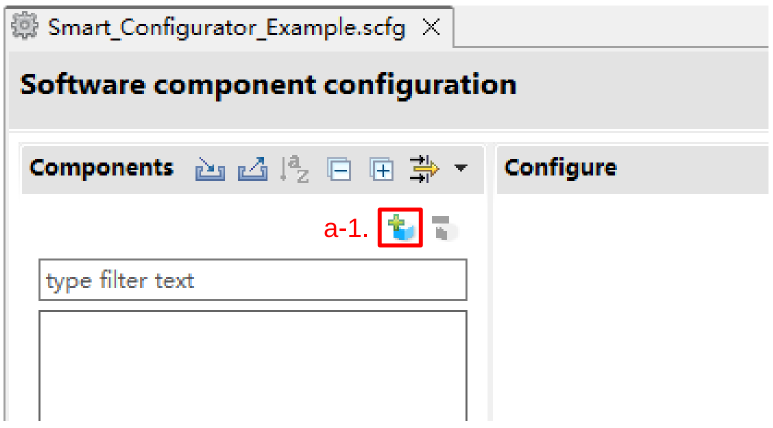
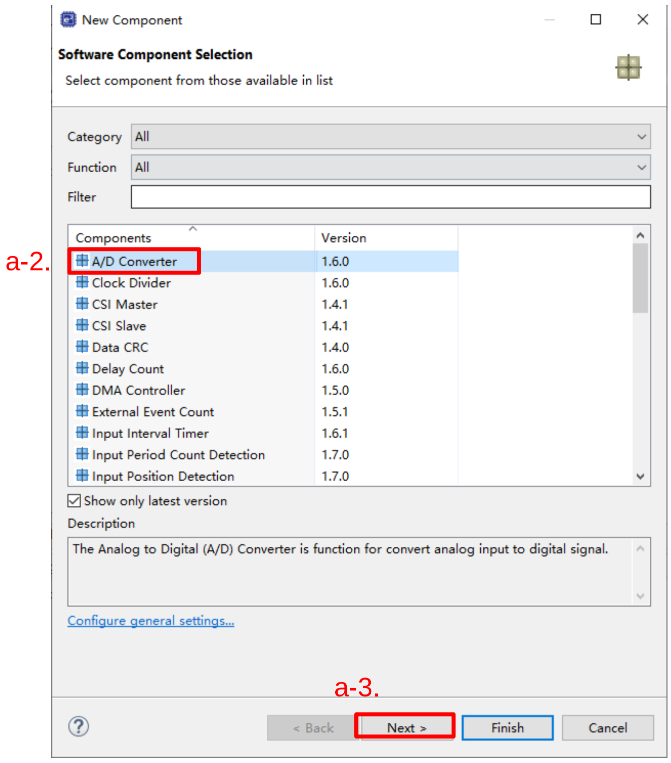

Smart_Configurator_Example.scfg
Software component configuration
Components
Configure
a-1.
type filter text

Figure 4-19 Adding a Component

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

a-3. Click on [Next].



Components	Version
A/D Converter	1.6.0
Clock Divider	1.6.0
CSI Master	1.4.1
CSI Slave	1.4.1
Data CRC	1.4.0
Delay Count	1.6.0
DMA Controller	1.5.0
External Event Count	1.5.1
Input Interval Timer	1.6.1
Input Period Count Detection	1.7.0
Input Position Detection	1.7.0

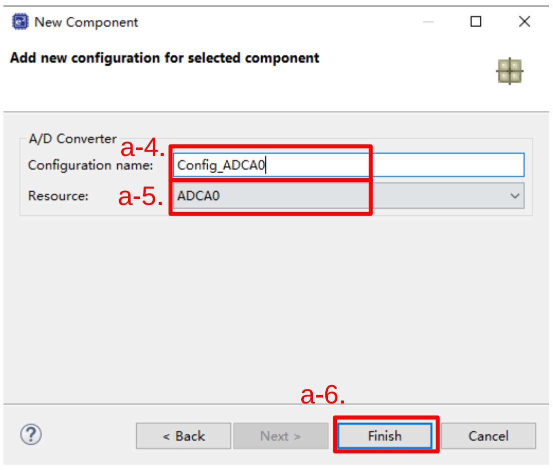
The Analog to Digital (A/D) Converter is function for convert analog input to digital signal.

Figure 4-20 Adding a Code Generator Component

a-4. Specify an appropriate configuration name in the [Add new configuration for selected component] page of the [New Component] dialog box or use the default name (for e.g., Config_ADCA0).

a-5. Select a hardware resource or use the default resource (for e.g., ADCA0).

a-6. Click on [Finish].


New Component

Add new configuration for selected component

A/D Converter

Configuration name: Config_ADCA0

Resource: ADCA0

Figure 4-21 Adding a Component

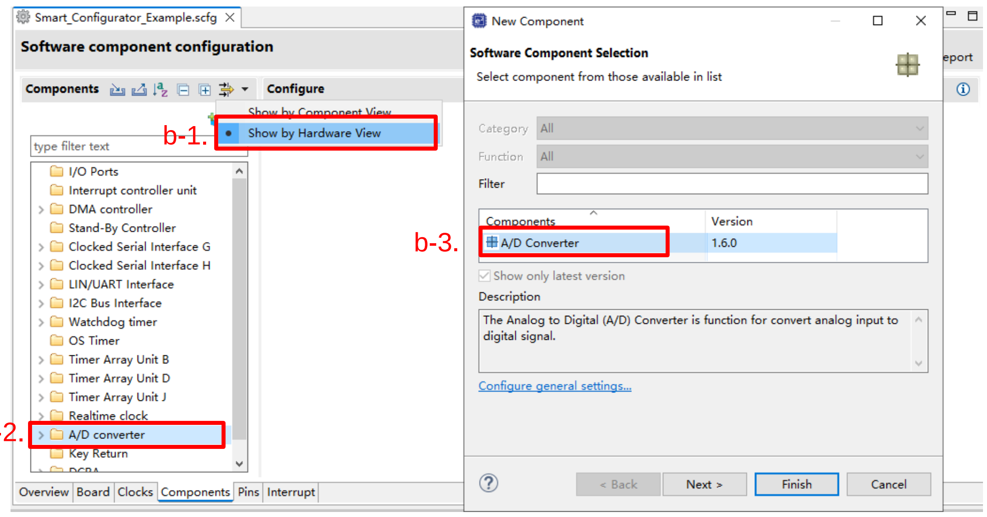
To add a component on Hardware Tree directly, user can use the following procedure:

b-1. Click on the “Show by Hardware View” radio button. The tree will display in a hardware resource hierarchy.

b-2. Double-click on a hardware resource node (for e.g., A/D Converter) to open the [New Component] dialog box.

b-3. (Select a component from the list (for e.g., A/D Converter) to add a new configuration。

b-4. Follow the same procedure is same as “Click on [(Add component)] icon” step a-2 to a-6.


Smart_Configurator_Example.scfg

Software component configuration

Components | Configure

Show by Component View
● Show by Hardware View

type filter text

I/O Ports
Interrupt controller unit
DMA controller
Stand-By Controller
Clocked Serial Interface G
Clocked Serial Interface H
LIN/UART Interface
I2C Bus Interface
Watchdog timer
OS Timer
Timer Array Unit B
Timer Array Unit D
Timer Array Unit J
Realtime clock
A/D converter
Key Return
DCRA

Overview | Board | Clocks | Components | Pins | Interrupt

New Component

Software Component Selection

Select component from those available in list

Category: All
Function: All
Filter:

Components	Version
A/D Converter	1.6.0

Show only latest version

Description

The Analog to Digital (A/D) Converter is function for convert analog input to digital signal.

Configure general settings...

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

4.4.3 Removing Software Component

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

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

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

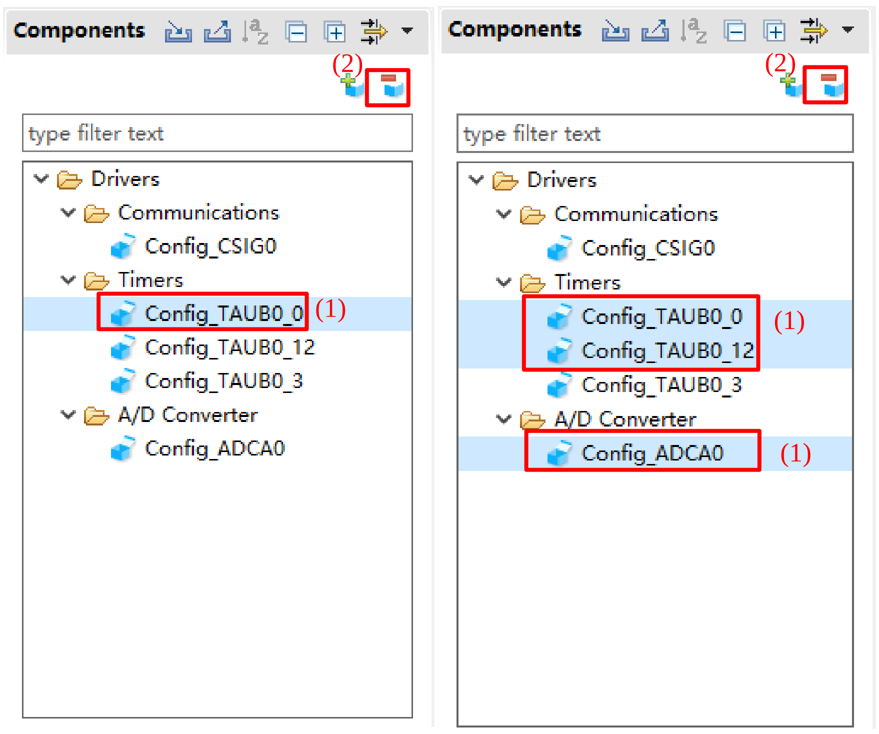


Figure 4-23 Removing a Software Component or Multiple Components

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

To delete the source files previously generated for the removed components from the e^2 studio project tree, click [(Generate Code)] icon.

4.4.4 Setting a Code Generator Component

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

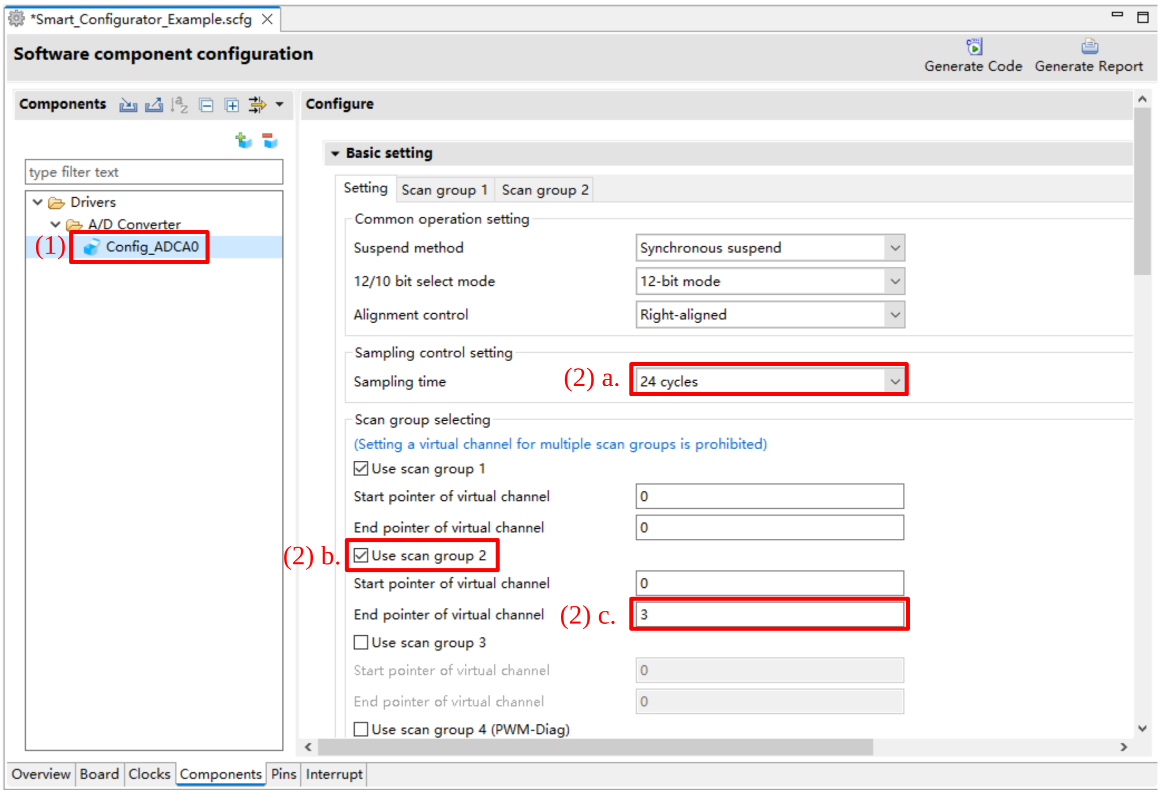
(1) Select a Code Generator configuration from the Components tree (for e.g., A/D Converter).

(2) Configure the driver in the [Configure] panel to the right of the Components tree. The following steps and figure show an example.

a. Select [24 cycles] under [Sampling control setting].

b. Select [Use scan group 2] under [Scan group selecting].

c. Input [2] to [End pointer of virtual channel].



*Smart_Configurator_Example.scfg

Software component configuration

Components

type filter text

Drivers
A/D Converter
(1) Config_ADCA0

Configure

Basic setting

Setting | Scan group 1 | Scan group 2

Common operation setting

Suspend method	Synchronous suspend
12/10 bit select mode	12-bit mode
Alignment control	Right-aligned

Sampling control setting

Sampling time (2) a.	24 cycles

Scan group selecting

(Setting a virtual channel for multiple scan groups is prohibited)

☑ Use scan group 1	
Start pointer of virtual channel	0
End pointer of virtual channel	0
(2) b. ☑ Use scan group 2	
Start pointer of virtual channel	0
End pointer of virtual channel (2) c.	3
☐ Use scan group 3	
Start pointer of virtual channel	0
End pointer of virtual channel	0
☐ Use scan group 4 (PWM-Diag)	

Figure 4-24 Setting of a Code Generator Driver

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

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

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

4.4.5 Changing the Resource for a Code Generator Configuration

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

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

(1) Right-click on a Code Generator configuration (for e.g., Config_ADCA0).

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

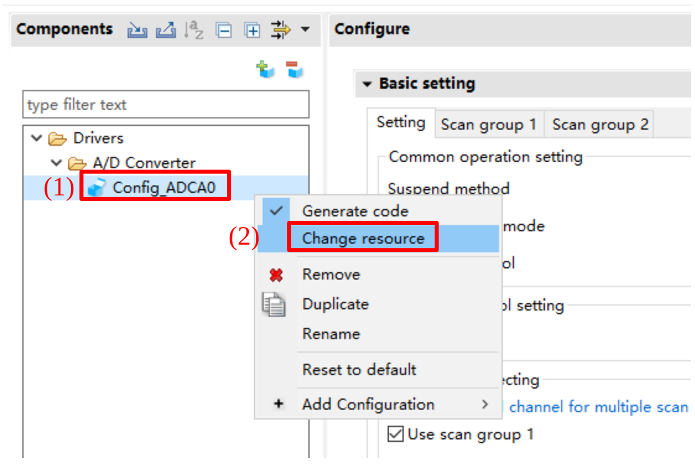


Figure 4-25 Changing the Resource

(3) Select a new resource (for e.g., ADCA1) in the [Resource Selection] dialog box.

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

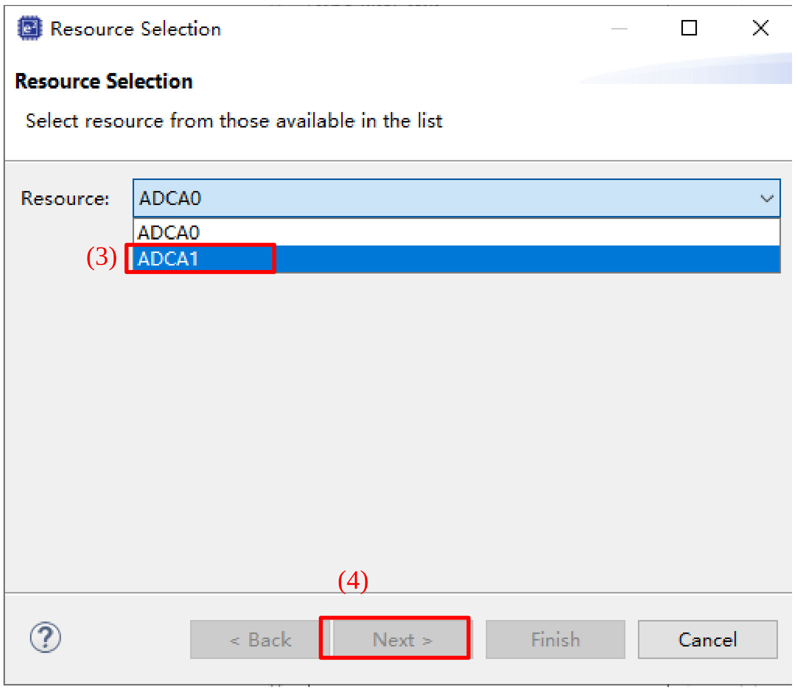


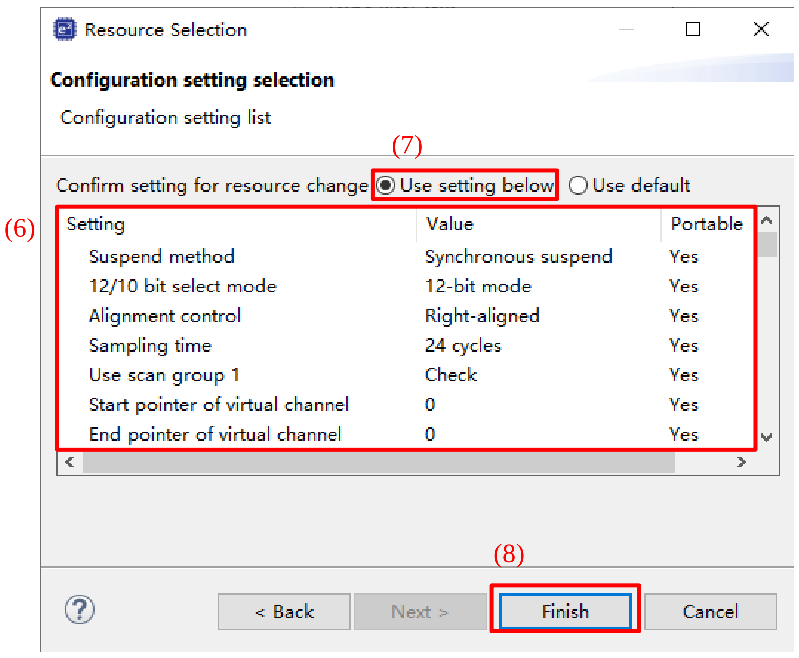
Figure 4-26 Components Page – Selecting a New Resource

(5) Configuration settings will be listed in the [Configuration setting selection] dialog box.

(6) Check the portability of the settings.

(7) Select whether to use the listed below or default settings.

(8) Click on [Finish].


Resource Selection

Configuration setting selection

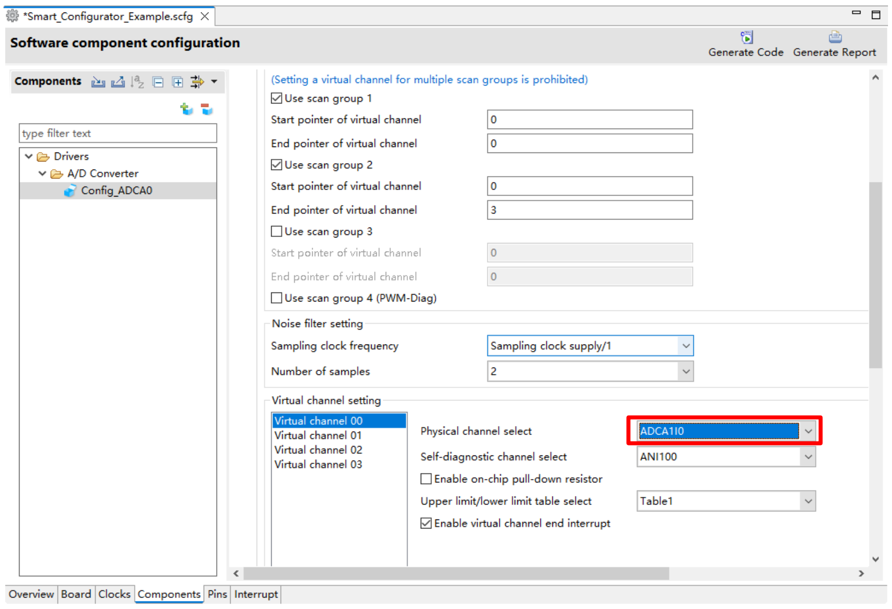
Configuration setting list

Confirm setting for resource change [✓ Use setting below] Use default

Setting	Value	Portable
Suspend method	Synchronous suspend	Yes
12/10 bit select mode	12-bit mode	Yes
Alignment control	Right-aligned	Yes
Sampling time	24 cycles	Yes
Use scan group 1	Check	Yes
Start pointer of virtual channel	0	Yes
End pointer of virtual channel	0	Yes

Figure 4-27 Checking the Settings of the New Resource

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


*Smart_Configurator_Example.scfg

Software component configuration

Components: Drivers > A/D Converter > Config_ADCA0

(Setting a virtual channel for multiple scan groups is prohibited)

[✓] Use scan group 1
Start pointer of virtual channel: 0
End pointer of virtual channel: 0
[✓] Use scan group 2
Start pointer of virtual channel: 0
End pointer of virtual channel: 3
[ ] Use scan group 3
Start pointer of virtual channel: 0
End pointer of virtual channel: 0
[ ] Use scan group 4 (PWM-Diag)

Noise filter setting
Sampling clock frequency: Sampling clock supply/1
Number of samples: 2

Virtual channel setting
Virtual channel 00
Virtual channel 01
Virtual channel 02
Virtual channel 03
Physical channel select: ADCA1I0
Self-diagnostic channel select: ANI100
[ ] Enable on-chip pull-down resistor
Upper limit/lower limit table select: Table1
[✓] Enable virtual channel end interrupt

Figure 4-28 Resource Changed Automatically

To change the configuration name, follow the procedure below.

(9) Right-click on the Code Generator configuration.

(10) Select [Rename] to rename the configuration (for e.g., change Config_ADCA0 to Config_ADCA1).

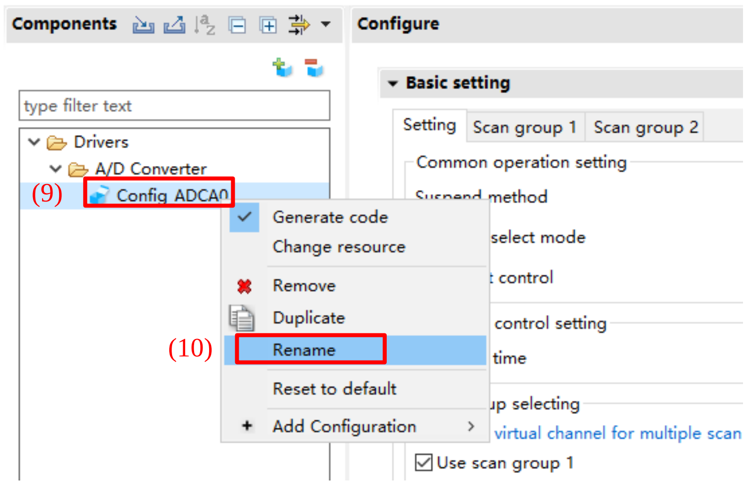


Figure 4-29 Renaming the Configuration

4.4.6 Export Component Configuration

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

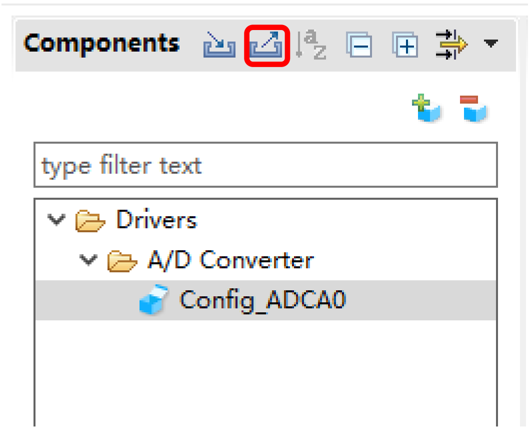


Figure 4-30 Export Configuration (xml format)

4.4.7 Import Component Configuration

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

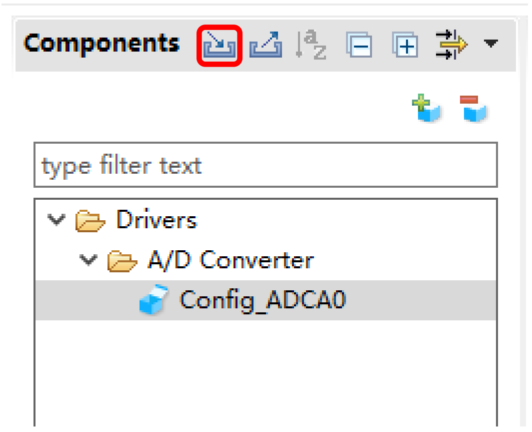
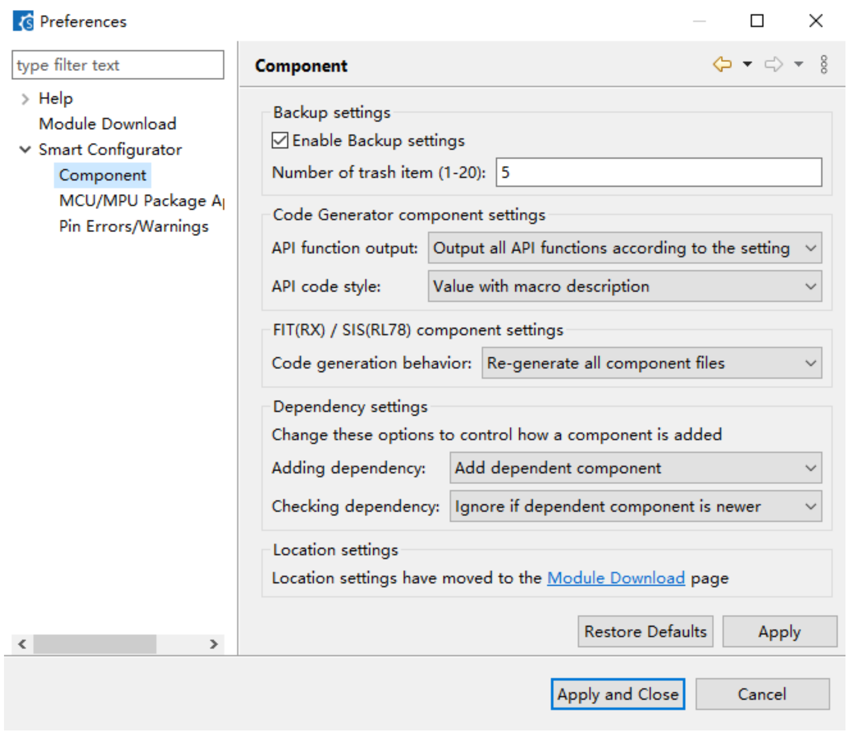


Figure 4-31 Import Configuration (xml format)

4.4.8 Configure General Setting of the Component

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



Component

Backup settings
☑ Enable Backup settings
Number of trash item (1-20): 5

Code Generator component settings
API function output: Output all API functions according to the setting
API code style: Value with macro description

FIT(RX) / SIS(RL78) component settings
Code generation behavior: Re-generate all component files

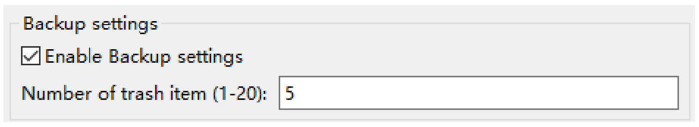
Dependency settings
Change these options to control how a component is added
Adding dependency: Add dependent component
Checking dependency: Ignore if dependent component is newer

Location settings
Location settings have moved to the Module Download page

Figure 4-32 Configure General Setting of Component

Notes:

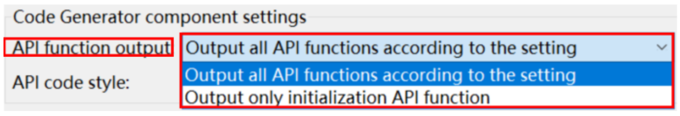
1. User can limit the number of folders created in the trash folder for backup purposes by setting the [Number of trash item (1-20)] option in the figure below. Once exceeding the limit, a folder with the newer timestamp will replace the oldest folder.



Backup settings
☑ Enable Backup settings
Number of trash item (1-20): 5

Figure 4-33 Trash number setting

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

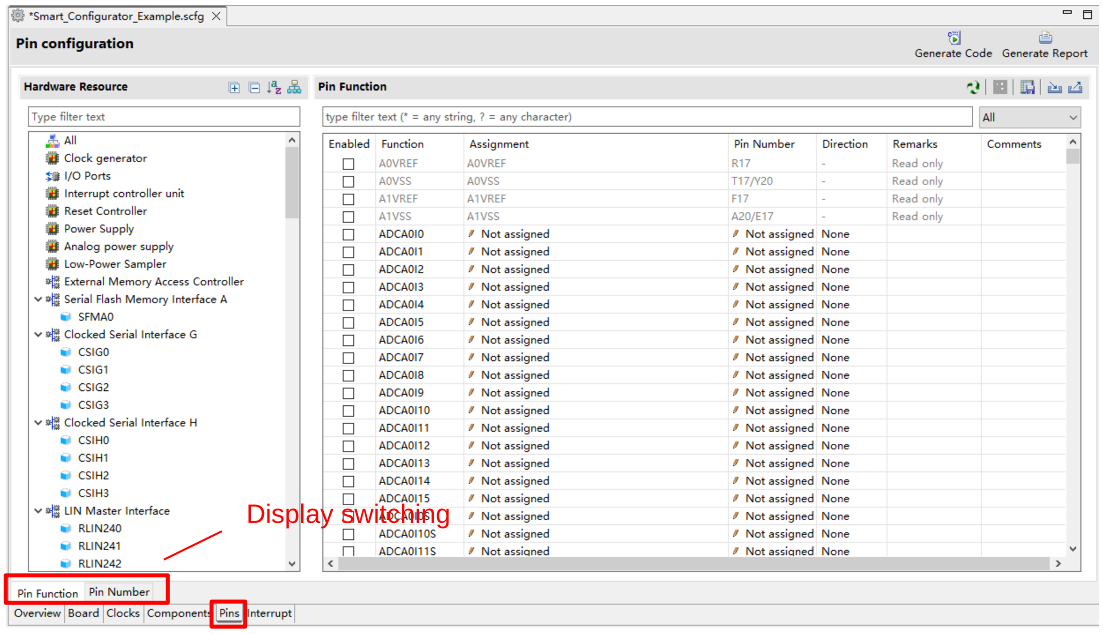


Code Generator component settings
API function output: Output all API functions according to the setting
API code style:
Output all API functions according to the setting
Output only initialization API function

Figure 4-34 [API function output] setting

4.5 Pin Settings

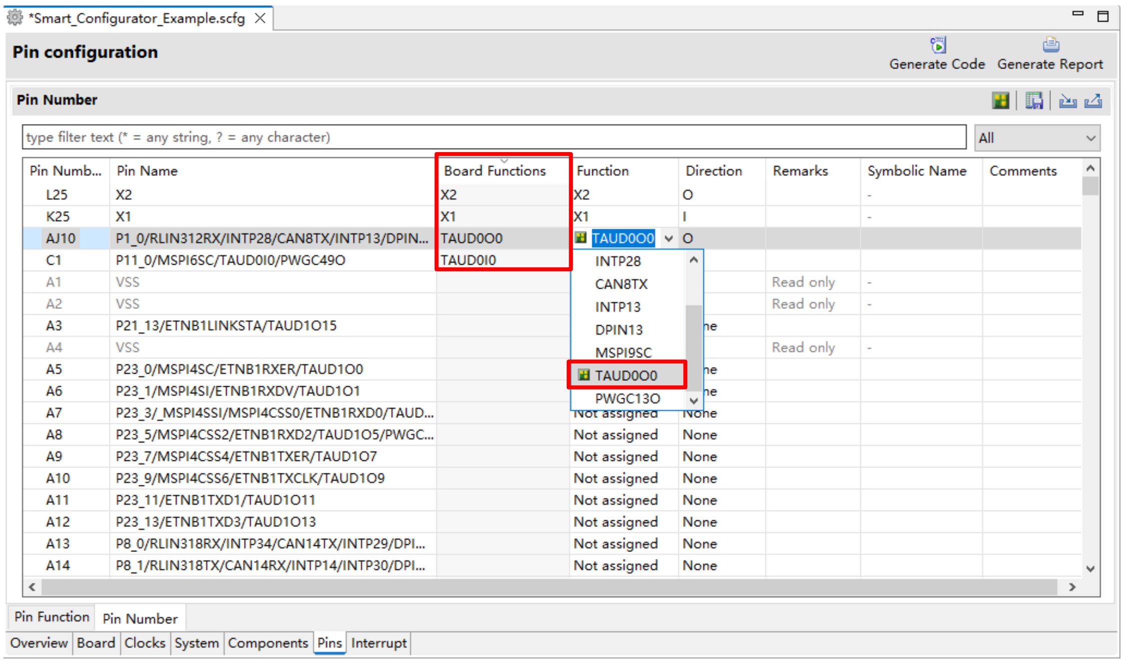
The [Pins] page is used for assigning pin functions. User can switch the view by clicking on the [Pin Function] and [Pin Number] pages. The [Pin Function] list shows the pin functions for each of the peripheral functions, and the [Pin Number] list shows all pins in order of pin number.



Enabled	Function	Assignment	Pin Number	Direction	Remarks	Comments
☐	A0VREF	A0VREF	R17	-	Read only	
☐	A0VSS	A0VSS	T17/Y20	-	Read only	
☐	A1VREF	A1VREF	F17	-	Read only	
☐	A1VSS	A1VSS	A20/E17	-	Read only	
☐	ADCA0I0	Not assigned	Not assigned	None		
☐	ADCA0I1	Not assigned	Not assigned	None		
☐	ADCA0I2	Not assigned	Not assigned	None		
☐	ADCA0I3	Not assigned	Not assigned	None		
☐	ADCA0I4	Not assigned	Not assigned	None		
☐	ADCA0I5	Not assigned	Not assigned	None		
☐	ADCA0I6	Not assigned	Not assigned	None		
☐	ADCA0I7	Not assigned	Not assigned	None		
☐	ADCA0I8	Not assigned	Not assigned	None		
☐	ADCA0I9	Not assigned	Not assigned	None		
☐	ADCA0I10	Not assigned	Not assigned	None		
☐	ADCA0I11	Not assigned	Not assigned	None		
☐	ADCA0I12	Not assigned	Not assigned	None		
☐	ADCA0I13	Not assigned	Not assigned	None		
☐	ADCA0I14	Not assigned	Not assigned	None		
☐	ADCA0I15	Not assigned	Not assigned	None		
☐	[illegible]	Not assigned	Not assigned	None		
☐	ADCA0I10S	Not assigned	Not assigned	None		
☐	ADCA0I11S	Not assigned	Not assigned	None		

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

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



Pin Numb...	Pin Name	Board Functions	Function	Direction	Remarks	Symbolic Name	Comments
L25	X2	X2	X2	O		-	
K25	X1	X1	X1	I		-	
AJ10	P1_0/RLIN312RX/INTP28/CAN8TX/INTP13/DPIN...	TAUD0O0	TAUD0O0	O			
C1	P11_0/MSPI6SC/TAUD0I0/PWGC49O	TAUD0I0					
A1	VSS				Read only	-	
A2	VSS				Read only	-	
A3	P21_13/ETNB1LINKSTA/TAUD1O15			ne			
A4	VSS				Read only	-	
A5	P23_0/MSPI4SC/ETNB1RXER/TAUD1O0			ne			
A6	P23_1/MSPI4SI/ETNB1RXDV/TAUD1O1			ne			
A7	P23_3/_MSPI4SSI/MSPI4CSS0/ETNB1RXD0/TAUD...		Not assigned	None			
A8	P23_5/MSPI4CSS2/ETNB1RXD2/TAUD1O5/PWGC...		Not assigned	None			
A9	P23_7/MSPI4CSS4/ETNB1TXER/TAUD1O7		Not assigned	None			
A10	P23_9/MSPI4CSS6/ETNB1TXCLK/TAUD1O9		Not assigned	None			
A11	P23_11/ETNB1TXD1/TAUD1O11		Not assigned	None			
A12	P23_13/ETNB1TXD3/TAUD1O13		Not assigned	None			
A13	P8_0/RLIN318RX/INTP34/CAN14TX/INTP29/DPI...		Not assigned	None			
A14	P8_1/RLIN318TX/CAN14RX/INTP14/INTP30/DPI...		Not assigned	None			

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

4.5.1 Changing the Pin Assignment of a Software Component

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

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

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

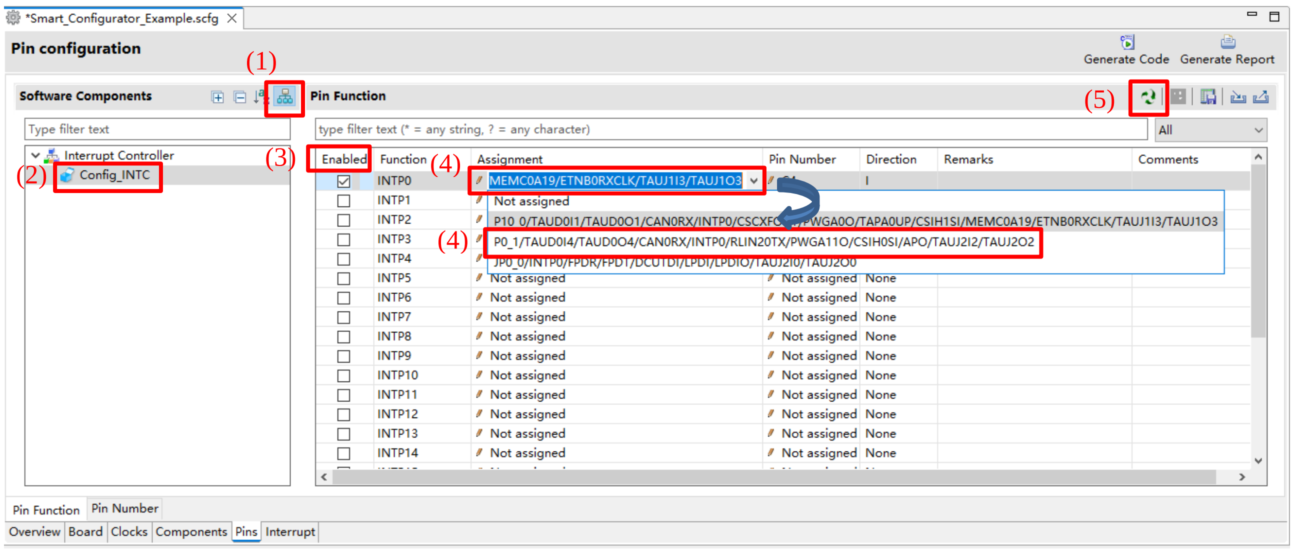
(1) Click on [(Show by Hardware Resource or Software Components)] to switch to the component view.

(2) Select the target software component (for e.g., Config_INTC).

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

(4) In the [Assignment] column or [Pin Number] column on the [Pin Function] list, change the pin assignment (for e.g., change from P10_0 to P0_1).

(5) In addition, assignment of a pin can be changed by clicking on the [(Next group of pins for the selected resource)] button. Pin that has peripheral function is displayed each time the button is clicked.


*Smart_Configurator_Example.scfg
Pin configuration
Generate Code
Generate Report
(1)
Software Components
Pin Function
(5)
Type filter text
type filter text (* = any string, ? = any character)
All
Interrupt Controller
(2)
Config_INTC
(3)
(4)
Not assigned
P10 0/TAUD0I1/TAUD0O1/CAN0RX/INTP0/CSCXFO…/PWGA0O/TAPA0UP/CSIH1SI/MEMC0A19/ETNB0RXCLK/TAUJ1I3/TAUJ1O3
(4)
P0_1/TAUD0I4/TAUD0O4/CAN0RX/INTP0/RLIN20TX/PWGA11O/CSIH0SI/APO/TAUJ2I2/TAUJ2O2
JP0_0/INTP0/FPDR/FPDT/DCUTDI/LPDI/LPDIO/TAUJ2I0/TAUJ2O0
Pin Function
Pin Number
Overview
Board
Clocks
Components
Pins
Interrupt

Enabled	Function	Assignment	Pin Number	Direction	Remarks	Comments
☑	INTP0	MEMC0A19/ETNB0RXCLK/TAUJ1I3/TAUJ1O3		I		
☐	INTP1					
☐	INTP2					
☐	INTP3					
☐	INTP4					
☐	INTP5	Not assigned	Not assigned	None		
☐	INTP6	Not assigned	Not assigned	None		
☐	INTP7	Not assigned	Not assigned	None		
☐	INTP8	Not assigned	Not assigned	None		
☐	INTP9	Not assigned	Not assigned	None		
☐	INTP10	Not assigned	Not assigned	None		
☐	INTP11	Not assigned	Not assigned	None		
☐	INTP12	Not assigned	Not assigned	None		
☐	INTP13	Not assigned	Not assigned	None		
☐	INTP14	Not assigned	Not assigned	None		

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

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

4.5.2 Assigning Pins Using the MCU/MPU Package View

The Smart Configurator visualizes the pin assignment in the MCU/MPU Package view. User can save the MCU/MPU Package view as an image file, rotate it, and zoom in to and out from it.

Follow the procedure below to assign pins in the MCU/MPU Package view.

(1) Zoom in to the view by clicking the [(Zoom in)] button or scrolling the view with the mouse wheel.

(2) Right-click on the target pin.

(3) Select the signal to be assigned to the pin.

(4) The color of the pins can be customized through [Preference Setting...].

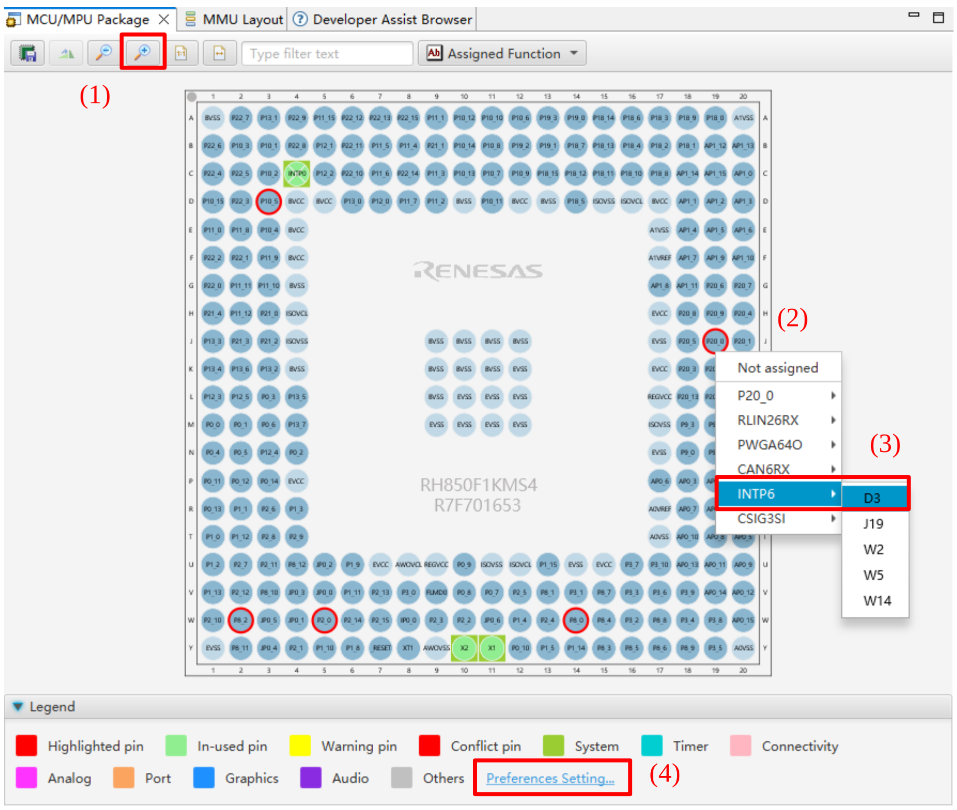


Figure 4-38 Assigning Pins Using the MCU/MPU Package View

4.5.3 Show Pin Number from Pin Functions

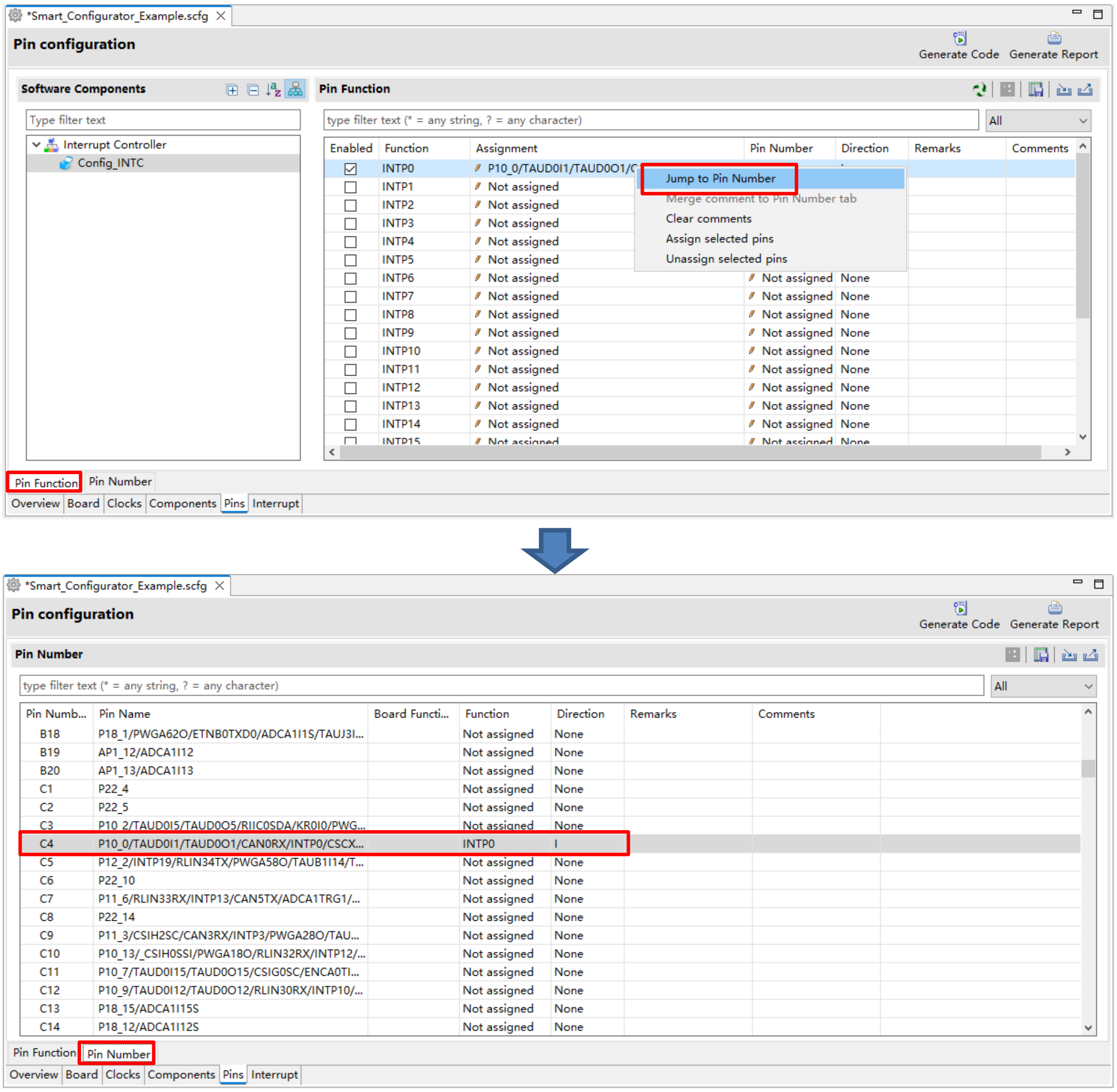
User can go to the pin number associated with a pin function.

Follow the procedure below to jump to pin number from a pin function.

(1) In the [Pin Function] tab, right click on a Pin Function to open the pop-up menu.

(2) Select “Jump to Pin Number”.

(3) The [Pin Number] tab is opened with a Pin Number being selected. This is the pin number of the pin function.



Enabled	Function	Assignment	Pin Number	Direction	Remarks	Comments
☑	INTP0	P10_0/TAUD0I1/TAUD0O1/C				
☐	INTP1	Not assigned				
☐	INTP2	Not assigned				
☐	INTP3	Not assigned				
☐	INTP4	Not assigned				
☐	INTP5	Not assigned				
☐	INTP6	Not assigned	Not assigned	None		
☐	INTP7	Not assigned	Not assigned	None		
☐	INTP8	Not assigned	Not assigned	None		
☐	INTP9	Not assigned	Not assigned	None		
☐	INTP10	Not assigned	Not assigned	None		
☐	INTP11	Not assigned	Not assigned	None		
☐	INTP12	Not assigned	Not assigned	None		
☐	INTP13	Not assigned	Not assigned	None		
☐	INTP14	Not assigned	Not assigned	None		
☐	INTP15	Not assigned	Not assigned	None		

Pin Numb...	Pin Name	Board Functi...	Function	Direction	Remarks	Comments
B18	P18_1/PWGA62O/ETNB0TXD0/ADCA1I1S/TAUJ3I...		Not assigned	None		
B19	AP1_12/ADCA1I12		Not assigned	None		
B20	AP1_13/ADCA1I13		Not assigned	None		
C1	P22_4		Not assigned	None		
C2	P22_5		Not assigned	None		
C3	P10_2/TAUD0I5/TAUD0O5/RIIC0SDA/KR0I0/PWG...		Not assigned	None		
C4	P10_0/TAUD0I1/TAUD0O1/CAN0RX/INTP0/CSCX...		INTP0	I		
C5	P12_2/INTP19/RLIN34TX/PWGA58O/TAUB1I14/T...		Not assigned	None		
C6	P22_10		Not assigned	None		
C7	P11_6/RLIN33RX/INTP13/CAN5TX/ADCA1TRG1/...		Not assigned	None		
C8	P22_14		Not assigned	None		
C9	P11_3/CSIH2SC/CAN3RX/INTP3/PWGA28O/TAU...		Not assigned	None		
C10	P10_13/_CSIH0SSI/PWGA18O/RLIN32RX/INTP12/...		Not assigned	None		
C11	P10_7/TAUD0I15/TAUD0O15/CSIG0SC/ENCA0TI...		Not assigned	None		
C12	P10_9/TAUD0I12/TAUD0O12/RLIN30RX/INTP10/...		Not assigned	None		
C13	P18_15/ADCA1I15S		Not assigned	None		
C14	P18_12/ADCA1I12S		Not assigned	None		

Figure 4-39 Jump to Pin Number

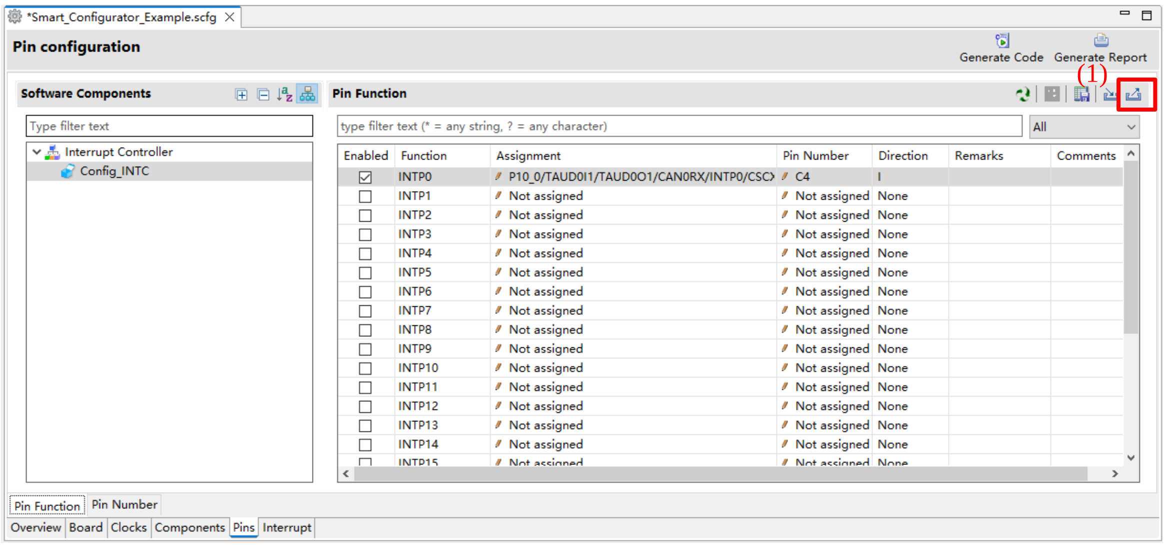
4.5.4 Exporting pin settings

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

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

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

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



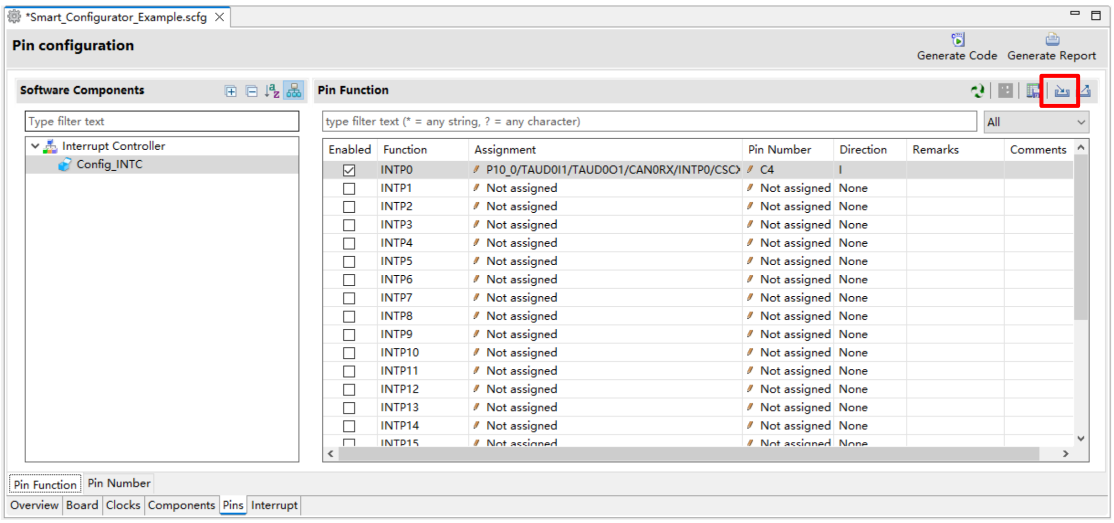
Enabled	Function	Assignment	Pin Number	Direction	Remarks	Comments
☑	INTP0	P10_0/TAUD0I1/TAUD0O1/CAN0RX/INTP0/CSC>	C4	I		
☐	INTP1	Not assigned	Not assigned	None		
☐	INTP2	Not assigned	Not assigned	None		
☐	INTP3	Not assigned	Not assigned	None		
☐	INTP4	Not assigned	Not assigned	None		
☐	INTP5	Not assigned	Not assigned	None		
☐	INTP6	Not assigned	Not assigned	None		
☐	INTP7	Not assigned	Not assigned	None		
☐	INTP8	Not assigned	Not assigned	None		
☐	INTP9	Not assigned	Not assigned	None		
☐	INTP10	Not assigned	Not assigned	None		
☐	INTP11	Not assigned	Not assigned	None		
☐	INTP12	Not assigned	Not assigned	None		
☐	INTP13	Not assigned	Not assigned	None		
☐	INTP14	Not assigned	Not assigned	None		
☐	INTP15	Not assigned	Not assigned	None		

Figure 4-40 Exporting Pin Settings to an XML File

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

4.5.5 Importing pin settings

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



Enabled	Function	Assignment	Pin Number	Direction	Remarks	Comments
☑	INTP0	P10_0/TAUD0I1/TAUD0O1/CAN0RX/INTP0/CSC	C4	I		
☐	INTP1	Not assigned	Not assigned	None		
☐	INTP2	Not assigned	Not assigned	None		
☐	INTP3	Not assigned	Not assigned	None		
☐	INTP4	Not assigned	Not assigned	None		
☐	INTP5	Not assigned	Not assigned	None		
☐	INTP6	Not assigned	Not assigned	None		
☐	INTP7	Not assigned	Not assigned	None		
☐	INTP8	Not assigned	Not assigned	None		
☐	INTP9	Not assigned	Not assigned	None		
☐	INTP10	Not assigned	Not assigned	None		
☐	INTP11	Not assigned	Not assigned	None		
☐	INTP12	Not assigned	Not assigned	None		
☐	INTP13	Not assigned	Not assigned	None		
☐	INTP14	Not assigned	Not assigned	None		
☐	INTP15	Not assigned	Not assigned	None		

Figure 4-41 Importing Pin Settings from an XML File

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

4.5.6 Pin Setting Using Board Pin Configuration Information

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

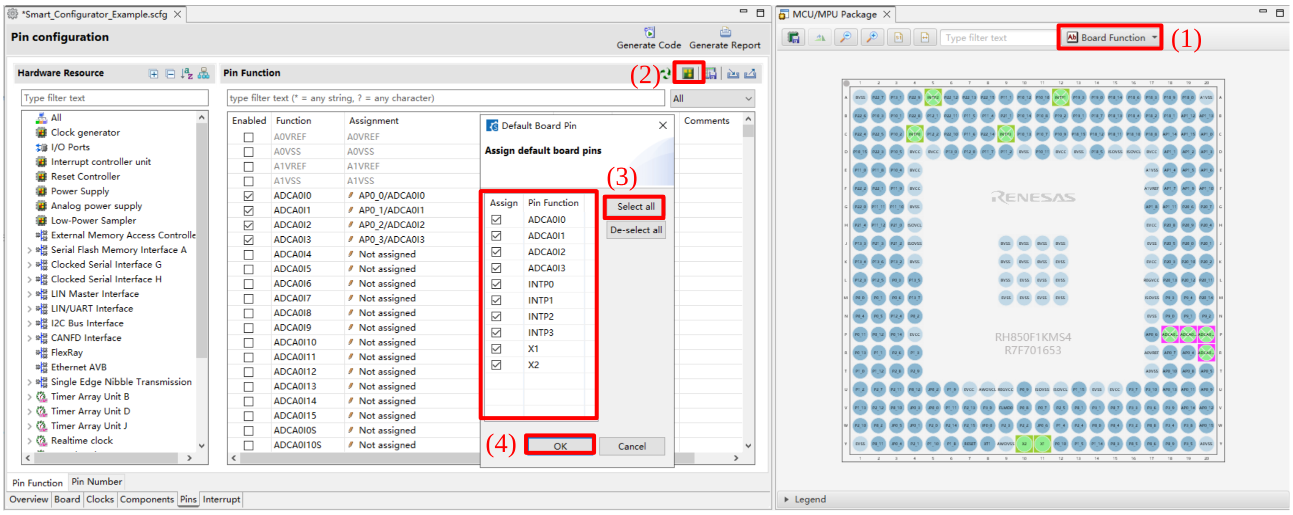
The following describes the procedure for collective setting of pins.

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

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

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

(4) Click [OK].



Enabled	Function	Assignment	Comments
☐	A0VREF	A0VREF	
☐	A0VSS	A0VSS	
☐	A1VREF	A1VREF	
☐	A1VSS	A1VSS	
☑	ADCA0I0	AP0_0/ADCA0I0	
☑	ADCA0I1	AP0_1/ADCA0I1	
☑	ADCA0I2	AP0_2/ADCA0I2	
☑	ADCA0I3	AP0_3/ADCA0I3	
☐	ADCA0I4	Not assigned	
☐	ADCA0I5	Not assigned	
☐	ADCA0I6	Not assigned	
☐	ADCA0I7	Not assigned	
☐	ADCA0I8	Not assigned	
☐	ADCA0I9	Not assigned	
☐	ADCA0I10	Not assigned	
☐	ADCA0I11	Not assigned	
☐	ADCA0I12	Not assigned	
☐	ADCA0I13	Not assigned	
☐	ADCA0I14	Not assigned	
☐	ADCA0I15	Not assigned	
☐	ADCA0I0S	Not assigned	
☐	ADCA0I10S	Not assigned	

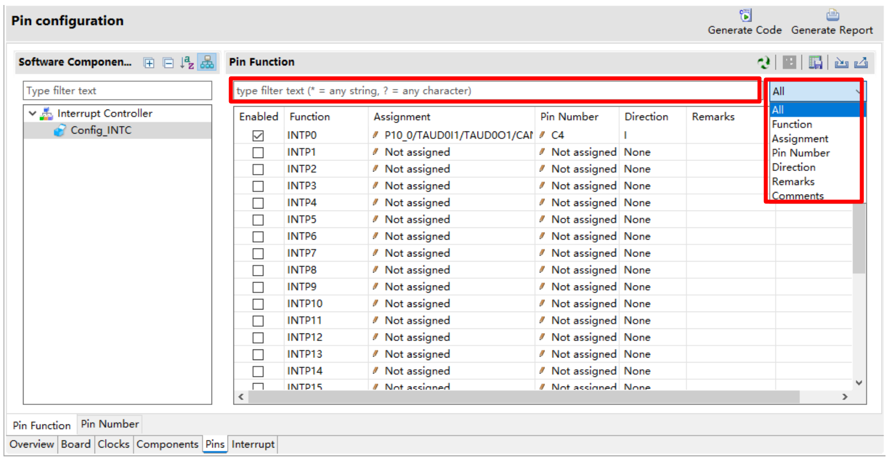
Assign	Pin Function
☑	ADCA0I0
☑	ADCA0I1
☑	ADCA0I2
☑	ADCA0I3
☑	INTP0
☑	INTP1
☑	INTP2
☑	INTP3
☑	X1
☑	X2

Figure 4-42 Importing Pin Settings from an XML File

If you do not set pin settings all at once, specify them individually in procedure (3).

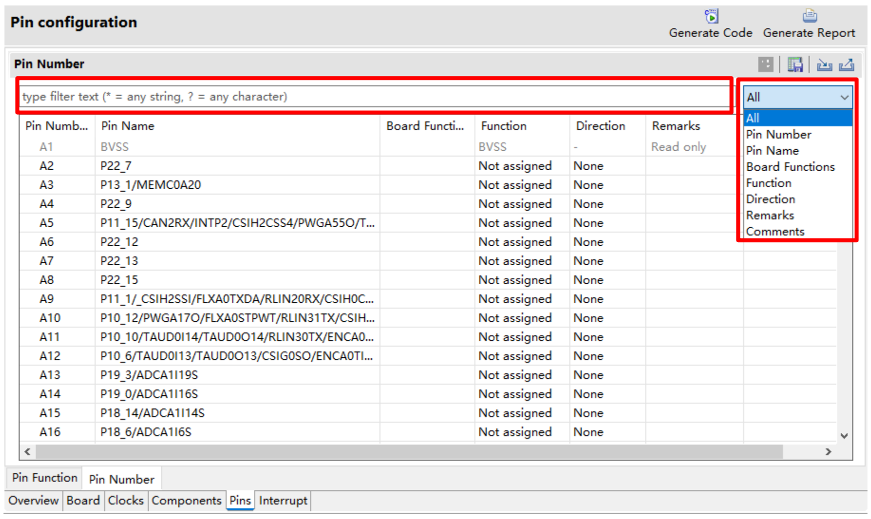
4.5.7 Pin Filter Feature

By specifying the filter range on the [Pin Function] page and [Pin Number] page on the [Pins] page, user can refer to it more easily.



Enabled	Function	Assignment	Pin Number	Direction	Remarks
✓	INTP0	P10_0/TAUD0I1/TAUD0O1/CAI	C4	I	
	INTP1	Not assigned	Not assigned	None	
	INTP2	Not assigned	Not assigned	None	
	INTP3	Not assigned	Not assigned	None	
	INTP4	Not assigned	Not assigned	None	
	INTP5	Not assigned	Not assigned	None	
	INTP6	Not assigned	Not assigned	None	
	INTP7	Not assigned	Not assigned	None	
	INTP8	Not assigned	Not assigned	None	
	INTP9	Not assigned	Not assigned	None	
	INTP10	Not assigned	Not assigned	None	
	INTP11	Not assigned	Not assigned	None	
	INTP12	Not assigned	Not assigned	None	
	INTP13	Not assigned	Not assigned	None	
	INTP14	Not assigned	Not assigned	None	
	INTP15	Not assigned	Not assigned	None	

Figure 4-43 Filter for [Pin Function] Page

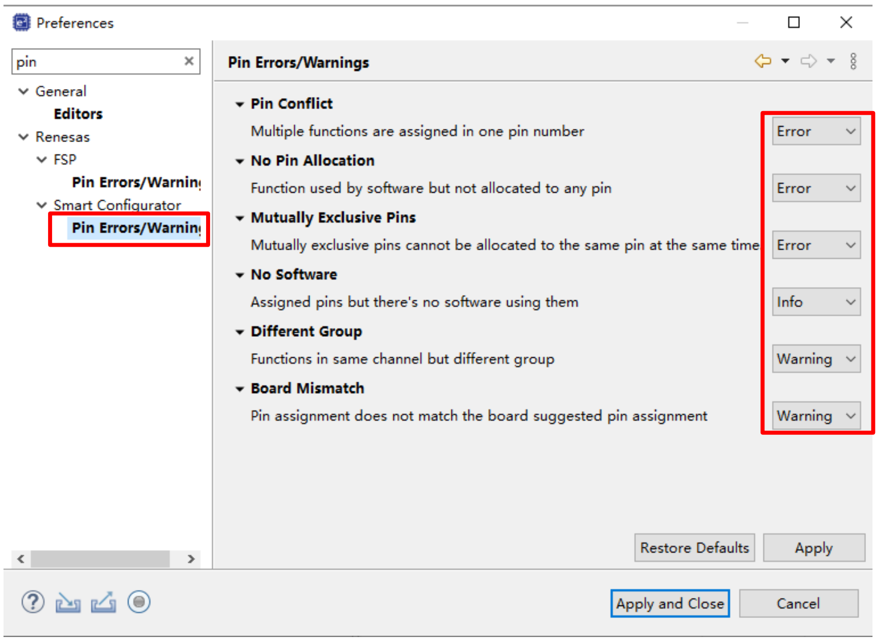


Pin Numb...	Pin Name	Board Functi...	Function	Direction	Remarks
A1	BVSS		BVSS	-	Read only
A2	P22_7		Not assigned	None	
A3	P13_1/MEMC0A20		Not assigned	None	
A4	P22_9		Not assigned	None	
A5	P11_15/CAN2RX/INTP2/CSIH2CSS4/PWGA55O/T...		Not assigned	None	
A6	P22_12		Not assigned	None	
A7	P22_13		Not assigned	None	
A8	P22_15		Not assigned	None	
A9	P11_1/_CSIH2SSI/FLXA0TXDA/RLIN20RX/CSIH0C...		Not assigned	None	
A10	P10_12/PWGA17O/FLXA0STPWT/RLIN31TX/CSIH...		Not assigned	None	
A11	P10_10/TAUD0I14/TAUD0O14/RLIN30TX/ENCA0...		Not assigned	None	
A12	P10_6/TAUD0I13/TAUD0O13/CSIG0SO/ENCA0TI...		Not assigned	None	
A13	P19_3/ADCA1I19S		Not assigned	None	
A14	P19_0/ADCA1I16S		Not assigned	None	
A15	P18_14/ADCA1I14S		Not assigned	None	
A16	P18_6/ADCA1I6S		Not assigned	None	

Figure 4-44 Filter for [Pin Number] Page

4.5.8 Pin Errors/Warnings setting

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

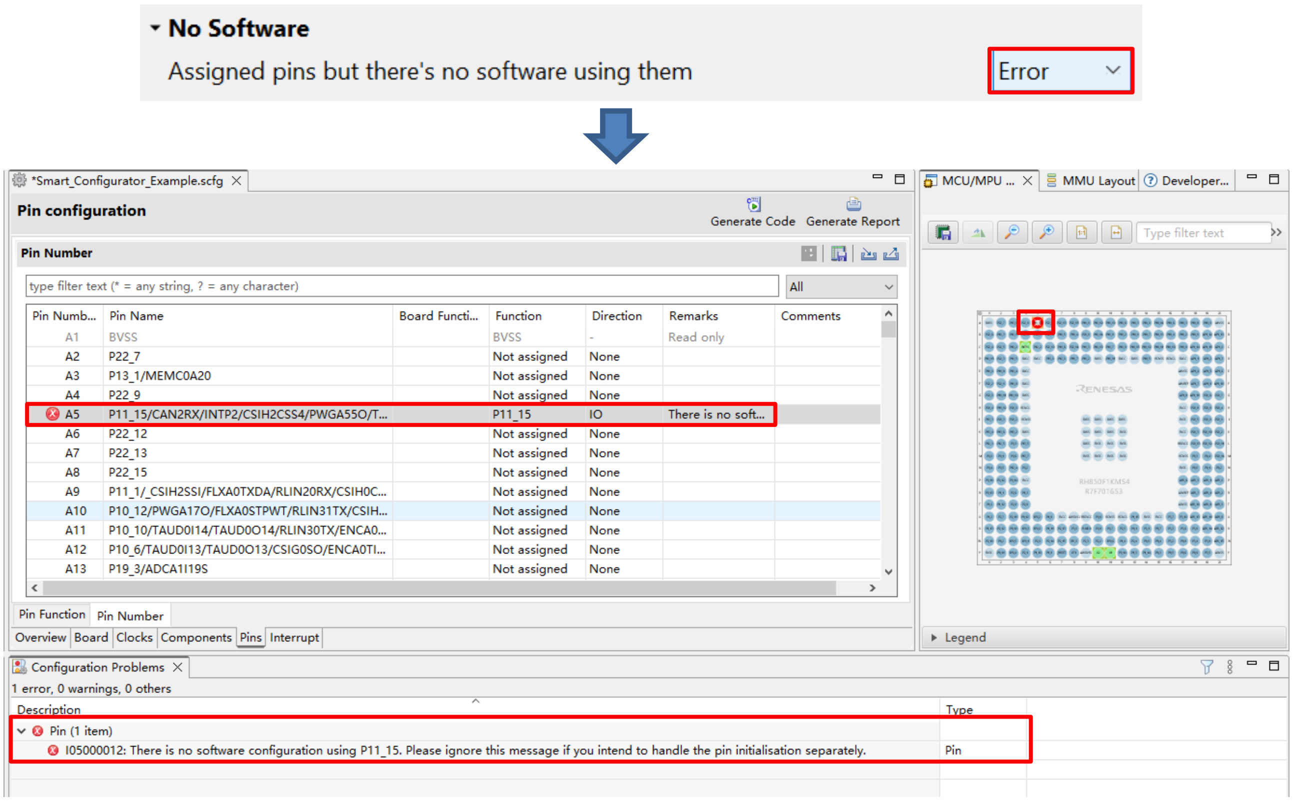


Pin Errors/Warnings

Pin Conflict	
Multiple functions are assigned in one pin number	Error
No Pin Allocation	
Function used by software but not allocated to any pin	Error
Mutually Exclusive Pins	
Mutually exclusive pins cannot be allocated to the same pin at the same time	Error
No Software	
Assigned pins but there's no software using them	Info
Different Group	
Functions in same channel but different group	Warning
Board Mismatch	
Pin assignment does not match the board suggested pin assignment	Warning

Figure 4-45 Pin Errors/Warnings settings at Preferences

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



No Software

Assigned pins but there's no software using them — Error

Pin configuration

Pin Number

type filter text (* = any string, ? = any character) — All

Pin Numb...	Pin Name	Board Functi...	Function	Direction	Remarks	Comments
A1	BVSS		BVSS	-	Read only	
A2	P22_7		Not assigned	None		
A3	P13_1/MEMC0A20		Not assigned	None		
A4	P22_9		Not assigned	None		
A5	P11_15/CAN2RX/INTP2/CSIH2CSS4/PWGA55O/T...		P11_15	IO	There is no soft...	
A6	P22_12		Not assigned	None		
A7	P22_13		Not assigned	None		
A8	P22_15		Not assigned	None		
A9	P11_1/_CSIH2SSI/FLXA0TXDA/RLIN20RX/CSIH0C...		Not assigned	None		
A10	P10_12/PWGA17O/FLXA0STPWT/RLIN31TX/CSIH...		Not assigned	None		
A11	P10_10/TAUD0I14/TAUD0O14/RLIN30TX/ENCA0...		Not assigned	None		
A12	P10_6/TAUD0I13/TAUD0O13/CSIG0SO/ENCA0TI...		Not assigned	None		
A13	P19_3/ADCA1I19S		Not assigned	None		

Configuration Problems

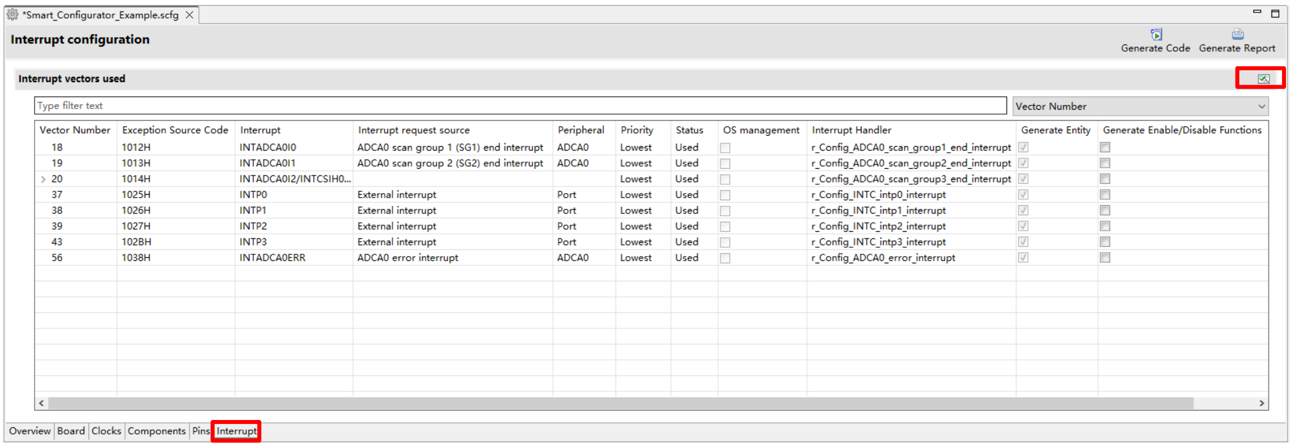
1 error, 0 warnings, 0 others

Description	Type
Pin (1 item)	
I05000012: There is no software configuration using P11_15. Please ignore this message if you intend to handle the pin initialisation separately.	Pin

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

4.6 Interrupt Settings

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



Vector Number	Exception Source Code	Interrupt	Interrupt request source	Peripheral	Priority	Status	OS management	Interrupt Handler	Generate Entity	Generate Enable/Disable Functions
18	1012H	INTADCA0I0	ADCA0 scan group 1 (SG1) end interrupt	ADCA0	Lowest	Used	☐	r_Config_ADCA0_scan_group1_end_interrupt	✓	☐
19	1013H	INTADCA0I1	ADCA0 scan group 2 (SG2) end interrupt	ADCA0	Lowest	Used	☐	r_Config_ADCA0_scan_group2_end_interrupt	✓	☐
> 20	1014H	INTADCA0I2/INTCSIH0...			Lowest	Used	☐	r_Config_ADCA0_scan_group3_end_interrupt	✓	☐
37	1025H	INTP0	External interrupt	Port	Lowest	Used	☐	r_Config_INTC_intp0_interrupt	✓	☐
38	1026H	INTP1	External interrupt	Port	Lowest	Used	☐	r_Config_INTC_intp1_interrupt	✓	☐
39	1027H	INTP2	External interrupt	Port	Lowest	Used	☐	r_Config_INTC_intp2_interrupt	✓	☐
43	102BH	INTP3	External interrupt	Port	Lowest	Used	☐	r_Config_INTC_intp3_interrupt	✓	☐
56	1038H	INTADCA0ERR	ADCA0 error interrupt	ADCA0	Lowest	Used	☐	r_Config_ADCA0_error_interrupt	✓	☐

Figure 4-47 [Interrupts] Page

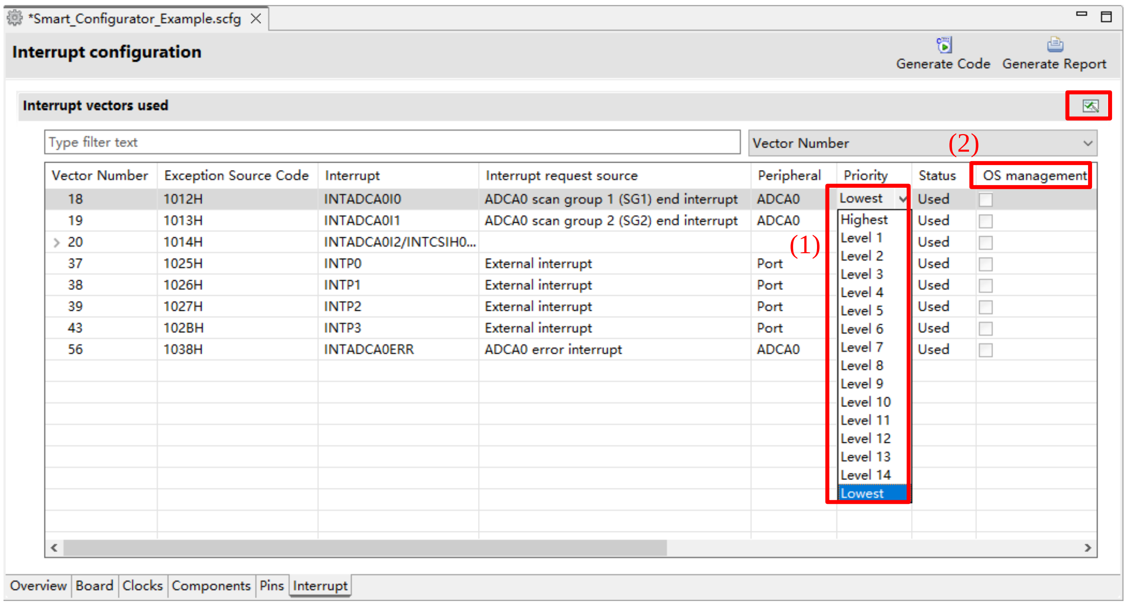
4.6.1 Changing Interrupt Priority Level and OS Management Setting

When an interrupt is used in a configuration on the [Components] page, the status of the interrupt will be changed to "Used". To display the used interrupts only, click on the [(Show used interrupts)] button.

(1) User can change the interrupt priority level on the [Interrupt] page.

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

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



Vector Number	Exception Source Code	Interrupt	Interrupt request source	Peripheral	Priority	Status	OS management
18	1012H	INTADCA0I0	ADCA0 scan group 1 (SG1) end interrupt	ADCA0	Lowest	Used	☐
19	1013H	INTADCA0I1	ADCA0 scan group 2 (SG2) end interrupt	ADCA0		Used	☐
> 20	1014H	INTADCA0I2/INTCSIH0...				Used	☐
37	1025H	INTP0	External interrupt	Port		Used	☐
38	1026H	INTP1	External interrupt	Port		Used	☐
39	1027H	INTP2	External interrupt	Port		Used	☐
43	102BH	INTP3	External interrupt	Port		Used	☐
56	1038H	INTADCA0ERR	ADCA0 error interrupt	ADCA0		Used	☐

Highest
Level 1
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7
Level 8
Level 9
Level 10
Level 11
Level 12
Level 13
Level 14
Lowest

Figure 4-48 Interrupt Setting

4.6.2 Changing The PE*n* Setting (RH850/U2A Only)

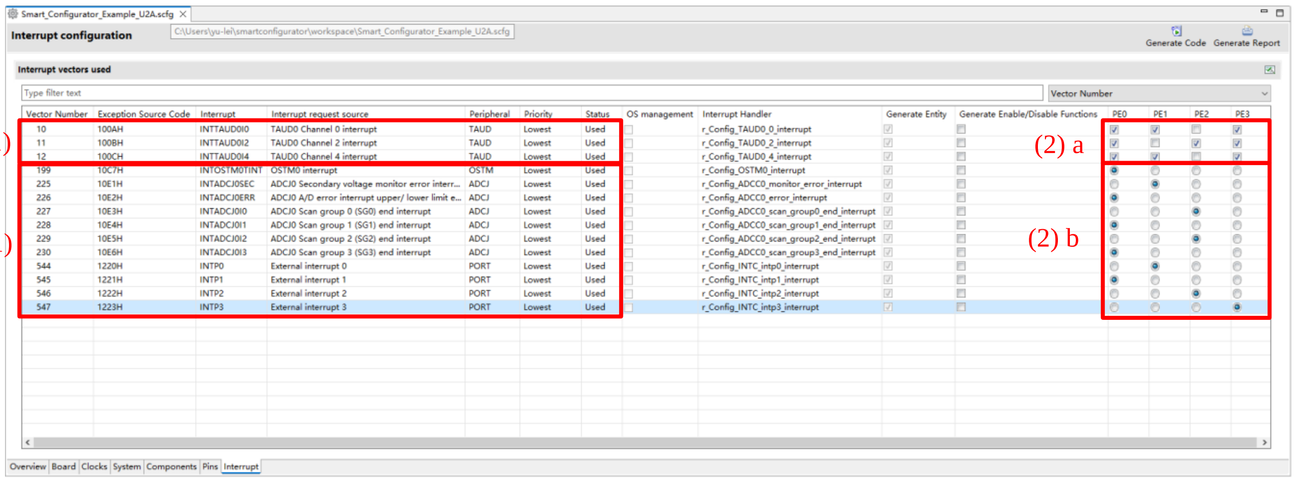
User can select which PE*n* to respond to the interrupt in use. PE*n* can be set on the [Interrupt] page by below steps:

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

(2) Check or uncheck the checkbox in column PE*n* in [Interrupt] page to select which PE to respond to the interrupt. There are two types of interrupts:

a Connected to INTC1 of each PE, each PE selected in the PE*n* column can respond.

b Connected to INTC2 shared by multiple PEs, only one PE selected in PE*n* column can respond.

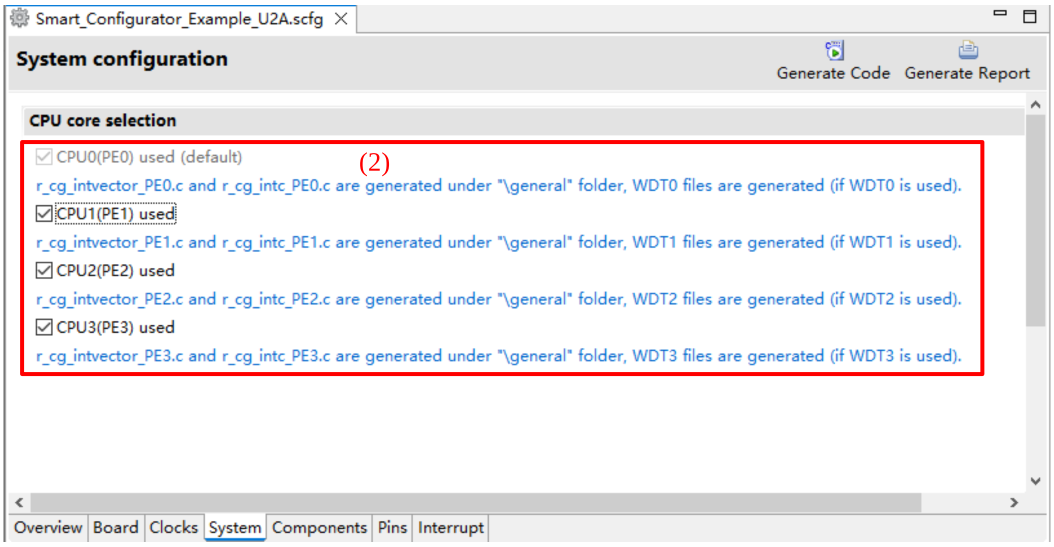


Interrupt configuration C:\Users\yu-lei\smartconfigurator\workspace\Smart_Configurator_Example_U2A.scfg

Interrupt vectors used

Vector Number	Exception Source Code	Interrupt	Interrupt request source	Peripheral	Priority	Status	OS management	Interrupt Handler	Generate Entity	Generate Enable/Disable Functions	PE0	PE1	PE2	PE3
10	100AH	INTTAUD0I0	TAUD0 Channel 0 interrupt	TAUD	Lowest	Used	☐	r_Config_TAUD0_0_interrupt	✓	☐	✓	✓	☐	✓
11	100BH	INTTAUD0I2	TAUD0 Channel 2 interrupt	TAUD	Lowest	Used	☐	r_Config_TAUD0_2_interrupt	✓	☐	✓	☐	✓	✓
12	100CH	INTTAUD0I4	TAUD0 Channel 4 interrupt	TAUD	Lowest	Used	☐	r_Config_TAUD0_4_interrupt	✓	☐	✓	✓	☐	✓
199	10C7H	INTOSTM0TINT	OSTM0 interrupt	OSTM	Lowest	Used	☐	r_Config_OSTM0_interrupt	✓	☐	●	○	○	○
225	10E1H	INTADCJ0SEC	ADCJ0 Secondary voltage monitor error interr...	ADCJ	Lowest	Used	☐	r_Config_ADCC0_monitor_error_interrupt	✓	☐	○	●	○	○
226	10E2H	INTADCJ0ERR	ADCJ0 A/D error interrupt upper/ lower limit e...	ADCJ	Lowest	Used	☐	r_Config_ADCC0_error_interrupt	✓	☐	●	○	○	○
227	10E3H	INTADCJ0I0	ADCJ0 Scan group 0 (SG0) end interrupt	ADCJ	Lowest	Used	☐	r_Config_ADCC0_scan_group0_end_interrupt	✓	☐	○	○	●	○
228	10E4H	INTADCJ0I1	ADCJ0 Scan group 1 (SG1) end interrupt	ADCJ	Lowest	Used	☐	r_Config_ADCC0_scan_group1_end_interrupt	✓	☐	●	○	○	○
229	10E5H	INTADCJ0I2	ADCJ0 Scan group 2 (SG2) end interrupt	ADCJ	Lowest	Used	☐	r_Config_ADCC0_scan_group2_end_interrupt	✓	☐	○	○	●	○
230	10E6H	INTADCJ0I3	ADCJ0 Scan group 3 (SG3) end interrupt	ADCJ	Lowest	Used	☐	r_Config_ADCC0_scan_group3_end_interrupt	✓	☐	●	○	○	○
544	1220H	INTP0	External interrupt 0	PORT	Lowest	Used	☐	r_Config_INTC_intp0_interrupt	✓	☐	○	●	○	○
545	1221H	INTP1	External interrupt 1	PORT	Lowest	Used	☐	r_Config_INTC_intp1_interrupt	✓	☐	●	○	○	○
546	1222H	INTP2	External interrupt 2	PORT	Lowest	Used	☐	r_Config_INTC_intp2_interrupt	✓	☐	○	○	●	○
547	1223H	INTP3	External interrupt 3	PORT	Lowest	Used	☐	r_Config_INTC_intp3_interrupt	✓	☐	○	○	○	●

Figure 4-49 PE*n* Setting



System configuration

CPU core selection

☑ CPU0(PE0) used (default)
r_cg_intvector_PE0.c and r_cg_intc_PE0.c are generated under "\general" folder, WDT0 files are generated (if WDT0 is used).
☑ CPU1(PE1) used
r_cg_intvector_PE1.c and r_cg_intc_PE1.c are generated under "\general" folder, WDT1 files are generated (if WDT1 is used).
☑ CPU2(PE2) used
r_cg_intvector_PE2.c and r_cg_intc_PE2.c are generated under "\general" folder, WDT2 files are generated (if WDT2 is used).
☑ CPU3(PE3) used
r_cg_intvector_PE3.c and r_cg_intc_PE3.c are generated under "\general" folder, WDT3 files are generated (if WDT3 is used).

Figure 4-50 PE*n* is Chosen to be Used or Unused in [System]

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

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

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

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

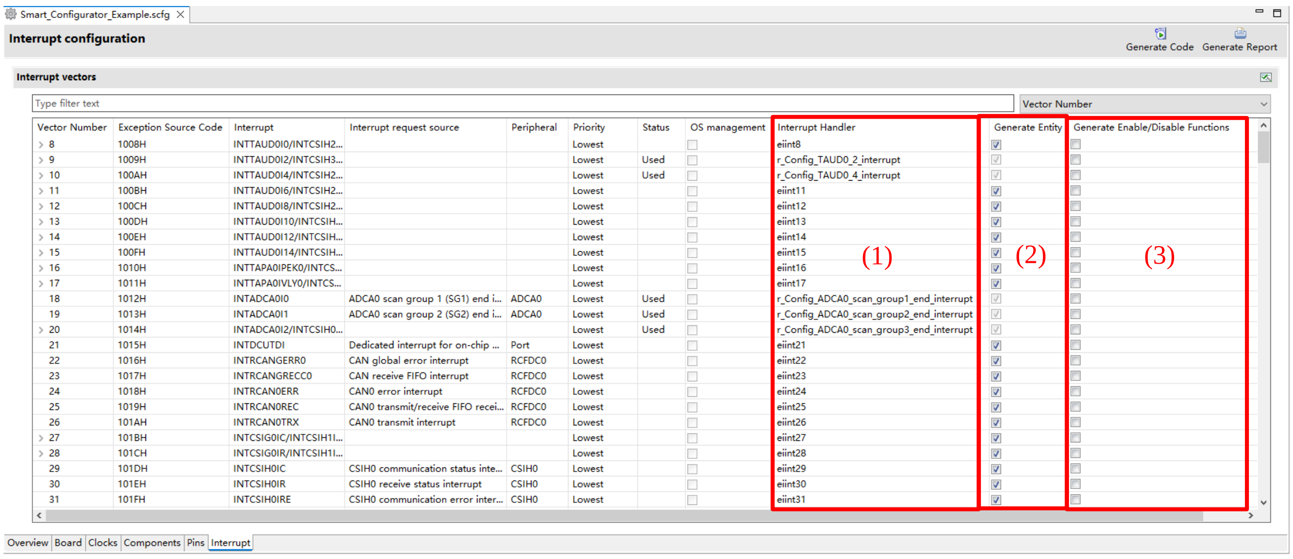
Note: interrupt handler which is used by components is non-editable.

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

(3) User can specify whether the interrupt enable/disable function is generated by Smart Configurator by checking/unchecking [Generate Enable/Disable Function].

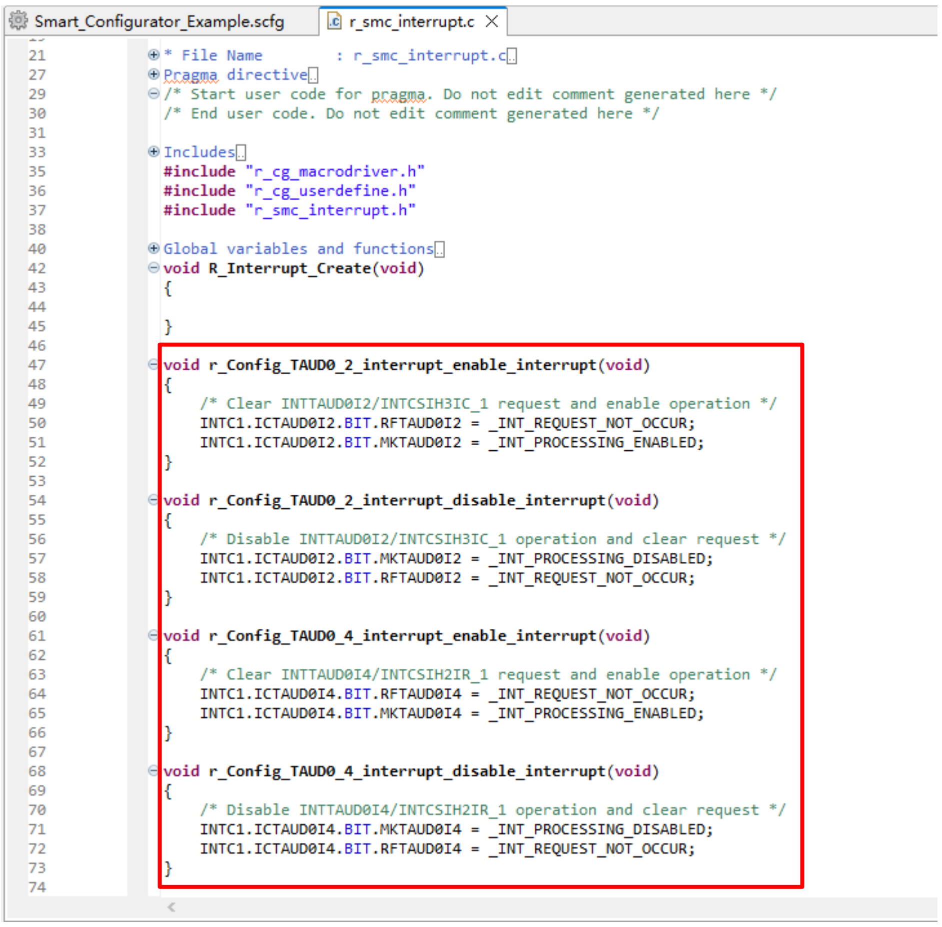
The default setting is unchecked. When user changes the setting to checked, a pair of interrupt enable/disable functions will be provided in "*r_smc_interrupt.c*" file. User can easily use interrupt by calling these APIs directly.

For code samples of the interrupt enable/disable functions, see **Figure 4-52 Interrupt Enable/Disable Functions Code Example**.



Vector Number	Exception Source Code	Interrupt	Interrupt request source	Peripheral	Priority	Status	OS management	Interrupt Handler	Generate Entity	Generate Enable/Disable Functions
8	1008H	INTTAUD0I0/INTCSIH2...			Lowest		☐	eiint8	✓	☐
9	1009H	INTTAUD0I2/INTCSIH3...			Lowest	Used	☐	r_Config_TAUD0_2_interrupt	✓	☐
10	100AH	INTTAUD0I4/INTCSIH2...			Lowest	Used	☐	r_Config_TAUD0_4_interrupt	✓	☐
11	100BH	INTTAUD0I6/INTCSIH2...			Lowest		☐	eiint11	✓	☐
12	100CH	INTTAUD0I8/INTCSIH2...			Lowest		☐	eiint12	✓	☐
13	100DH	INTTAUD0I10/INTCSIH...			Lowest		☐	eiint13	✓	☐
14	100EH	INTTAUD0I12/INTCSIH...			Lowest		☐	eiint14	✓	☐
15	100FH	INTTAUD0I14/INTCSIH...			Lowest		☐	eiint15	✓	☐
16	1010H	INTTAPA0IPEK0/INTCS...			Lowest		☐	eiint16	✓	☐
17	1011H	INTTAPA0IVLY0/INTCS...			Lowest		☐	eiint17	✓	☐
18	1012H	INTADCA0I0	ADCA0 scan group 1 (SG1) end i...	ADCA0	Lowest	Used	☐	r_Config_ADCA0_scan_group1_end_interrupt	✓	☐
19	1013H	INTADCA0I1	ADCA0 scan group 2 (SG2) end i...	ADCA0	Lowest	Used	☐	r_Config_ADCA0_scan_group2_end_interrupt	✓	☐
20	1014H	INTADCA0I2/INTCSIH0...			Lowest	Used	☐	r_Config_ADCA0_scan_group3_end_interrupt	✓	☐
21	1015H	INTDCUTDI	Dedicated interrupt for on-chip ...	Port	Lowest		☐	eiint21	✓	☐
22	1016H	INTRCANGERR0	CAN global error interrupt	RCFDC0	Lowest		☐	eiint22	✓	☐
23	1017H	INTRCANGRECC0	CAN receive FIFO interrupt	RCFDC0	Lowest		☐	eiint23	✓	☐
24	1018H	INTRCAN0ERR	CAN0 error interrupt	RCFDC0	Lowest		☐	eiint24	✓	☐
25	1019H	INTRCAN0REC	CAN0 transmit/receive FIFO recei...	RCFDC0	Lowest		☐	eiint25	✓	☐
26	101AH	INTRCAN0TRX	CAN0 transmit interrupt	RCFDC0	Lowest		☐	eiint26	✓	☐
27	101BH	INTCSIG0IC/INTCSIH1I...			Lowest		☐	eiint27	✓	☐
28	101CH	INTCSIG0IR/INTCSIH1I...			Lowest		☐	eiint28	✓	☐
29	101DH	INTCSIH0IC	CSIH0 communication status inte...	CSIH0	Lowest		☐	eiint29	✓	☐
30	101EH	INTCSIH0IR	CSIH0 receive status interrupt	CSIH0	Lowest		☐	eiint30	✓	☐
31	101FH	INTCSIH0IRE	CSIH0 communication error inter...	CSIH0	Lowest		☐	eiint31	✓	☐

Figure 4-51 Interrupt Handler and Generate Entity Settings



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

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

Global variables and functions
void R_Interrupt_Create(void)
{

}

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

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

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

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

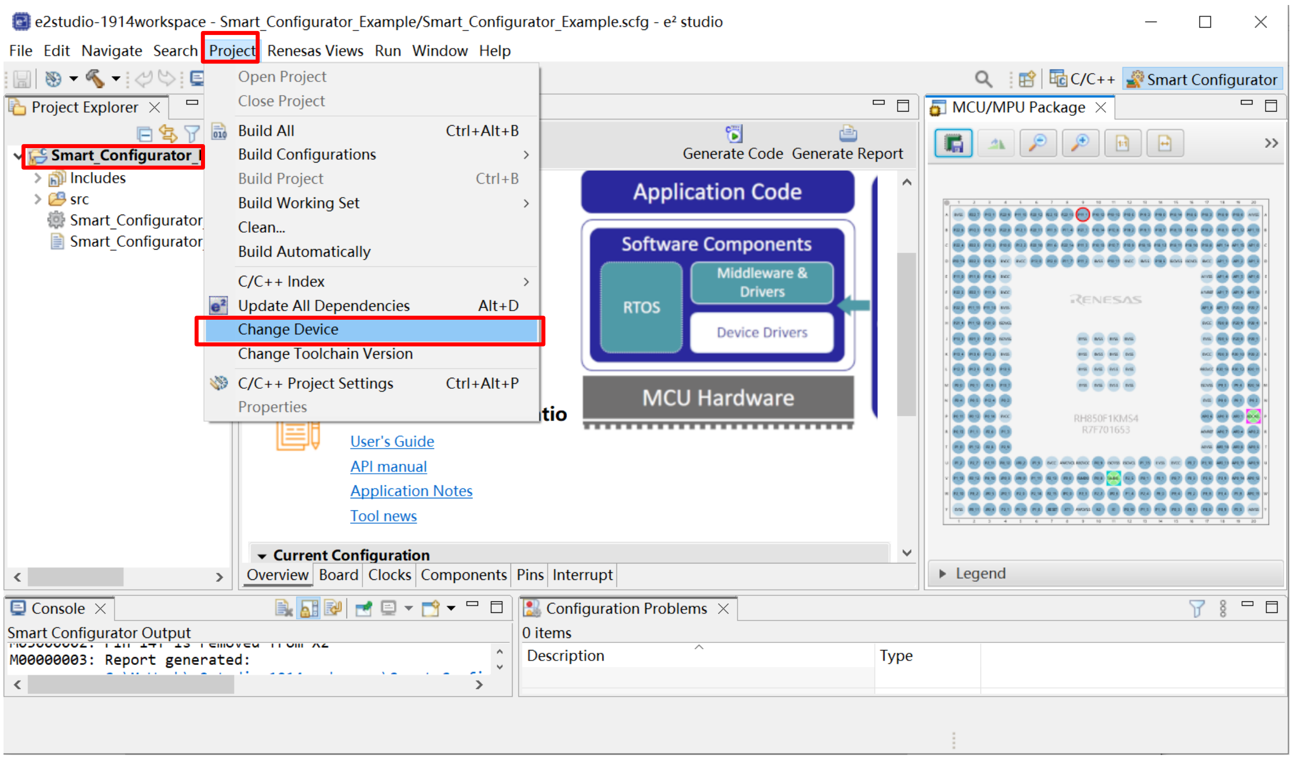
Figure 4-52 Interrupt Enable/Disable Functions Code Example

4.7 MCU Migration Feature

The MCU migration feature helps to convert your project settings from device A to device B. Conversion of project settings can be done within the same family and can be done from e^2 studio project menu as follows.

Note: Project settings may change due to device change.
Back up the project before executing the device change.

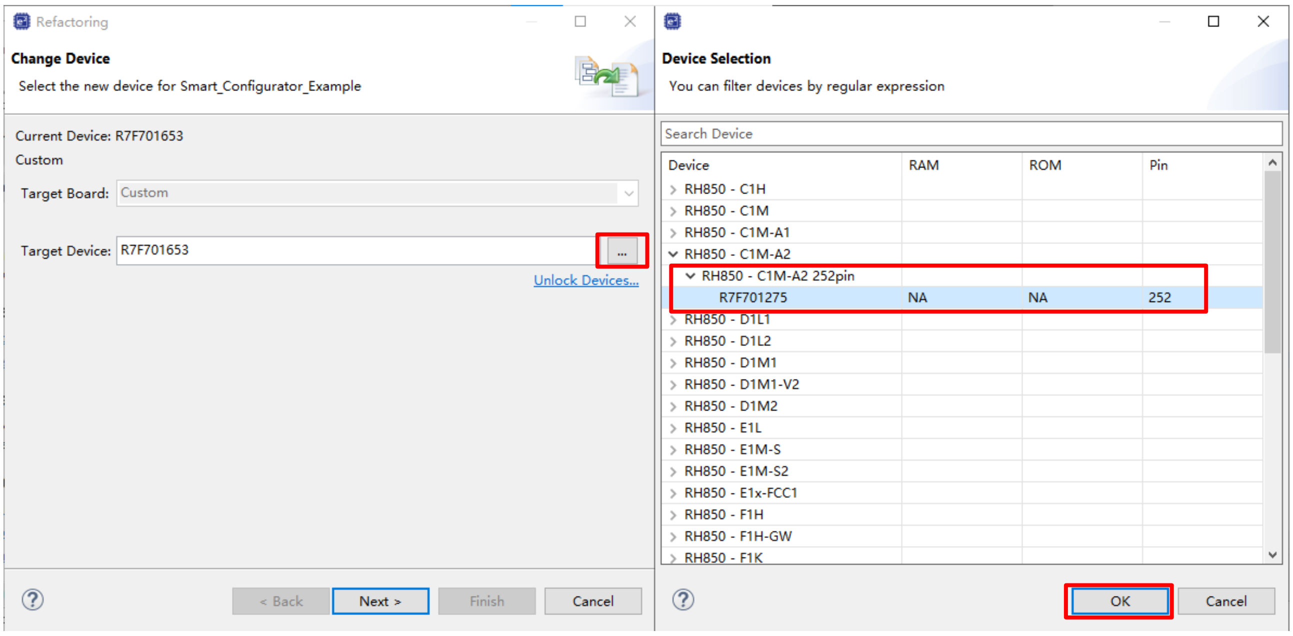
(1) Select the project and choose [Change Device] from the [Project] menu.



```
M00000003: Report generated:
```

Figure 4-53 Select [Change Device]

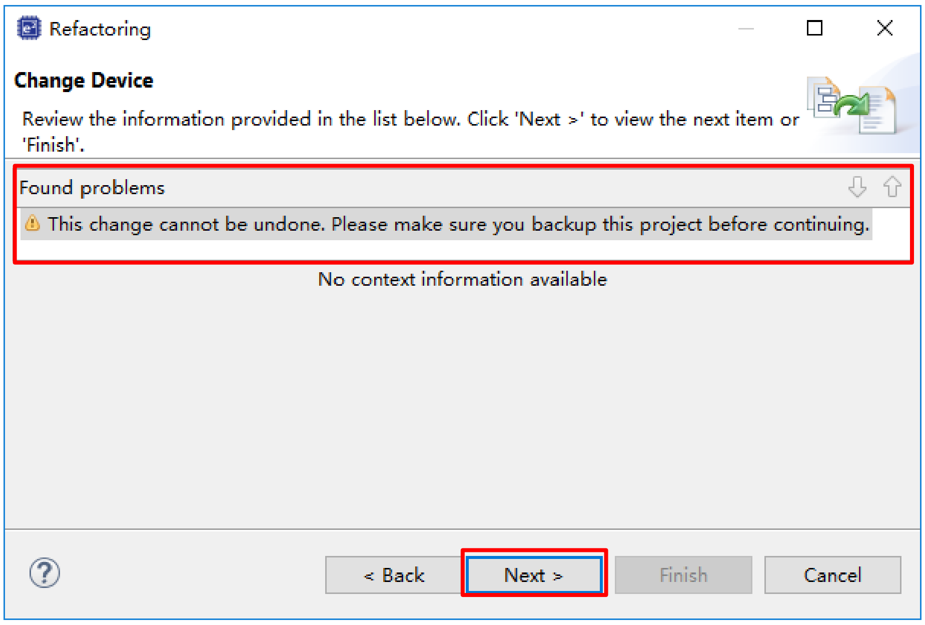
(2) Select the target device manually from the device selection list and click "OK". (Wild card search is supported) (e.g., change to RH850 – C1M 252pin Part number: R7F701275).



Device	RAM	ROM	Pin
RH850 - C1H			
RH850 - C1M			
RH850 - C1M-A1			
RH850 - C1M-A2			
RH850 - C1M-A2 252pin			
R7F701275	NA	NA	252
RH850 - D1L1			
RH850 - D1L2			
RH850 - D1M1			
RH850 - D1M1-V2			
RH850 - D1M2			
RH850 - E1L			
RH850 - E1M-S			
RH850 - E1M-S2			
RH850 - E1x-FCC1			
RH850 - F1H			
RH850 - F1H-GW			
RH850 - F1K			

Figure 4-54 Select Target Device

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



Review the information provided in the list below. Click 'Next >' to view the next item or 'Finish'.

Found problems

⚠ This change cannot be undone. Please make sure you backup this project before continuing.

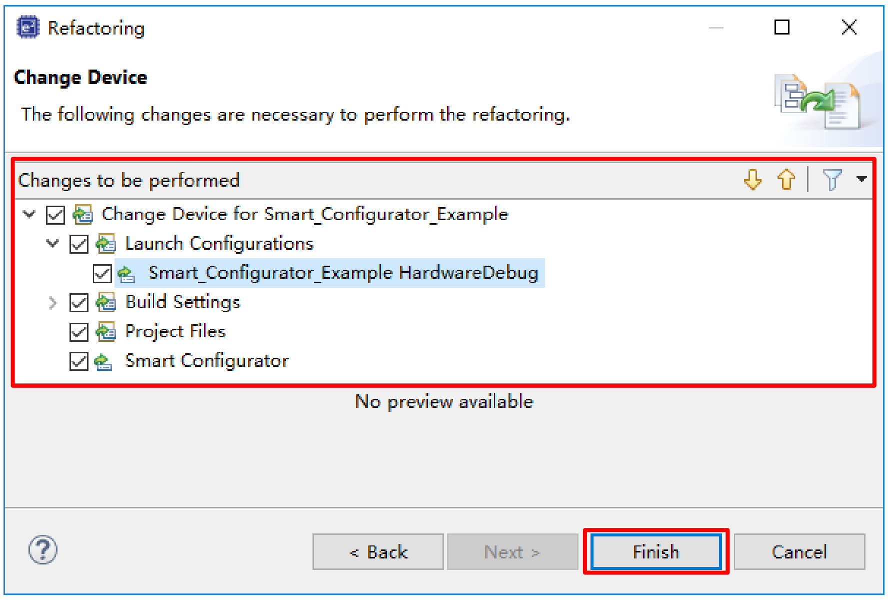
No context information available

Figure 4-55 Found Problems

Table 4-2 Change Device Found Problems List

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

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



Refactoring

Change Device

The following changes are necessary to perform the refactoring.

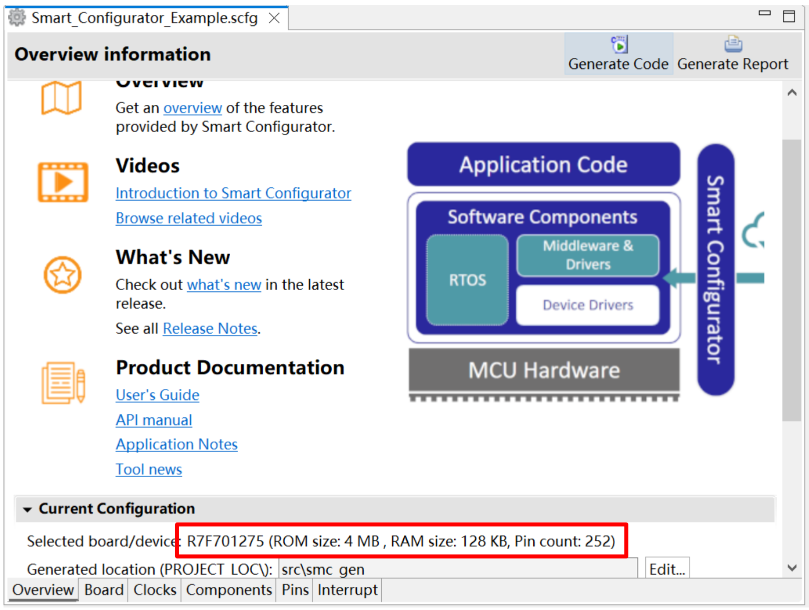
Changes to be performed

- Change Device for Smart_Configurator_Example
  - Launch Configurations
    - Smart_Configurator_Example HardwareDebug
  - Build Settings
  - Project Files
  - Smart Configurator

No preview available

Figure 4-56 Changes to be Performed

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



Smart_Configurator_Example.scfg

Overview information

Overview

Get an overview of the features provided by Smart Configurator.

Videos

Introduction to Smart Configurator

Browse related videos

What's New

Check out what's new in the latest release.

See all Release Notes.

Product Documentation

User's Guide

API manual

Application Notes

Tool news

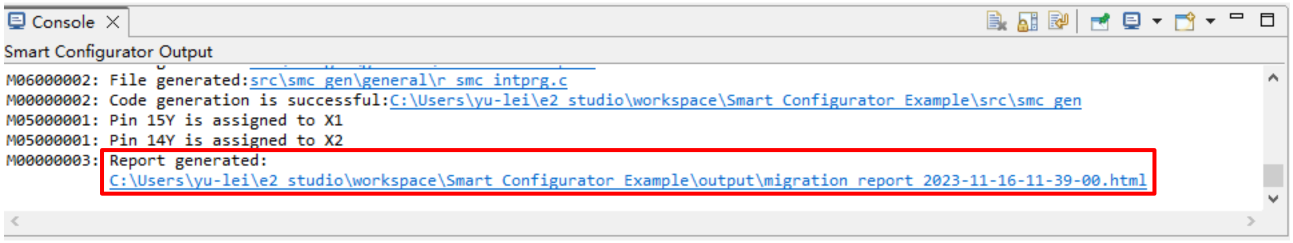
Current Configuration

Selected board/device: R7F701275 (ROM size: 4 MB , RAM size: 128 KB, Pin count: 252)

Generated location (PROJECT_LOC\): src\smc_gen

Figure 4-57 Device Update Confirmation

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



```
M06000002: File generated:src\smc_gen\general\r_smc_intprg.c
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
M05000001: Pin 15Y is assigned to X1
M05000001: Pin 14Y is assigned to X2
M00000003: Report generated:
           C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\output\migration_report_2023-11-16-11-39-00.html
```

Figure 4-58 Configuration Conversion Status Report

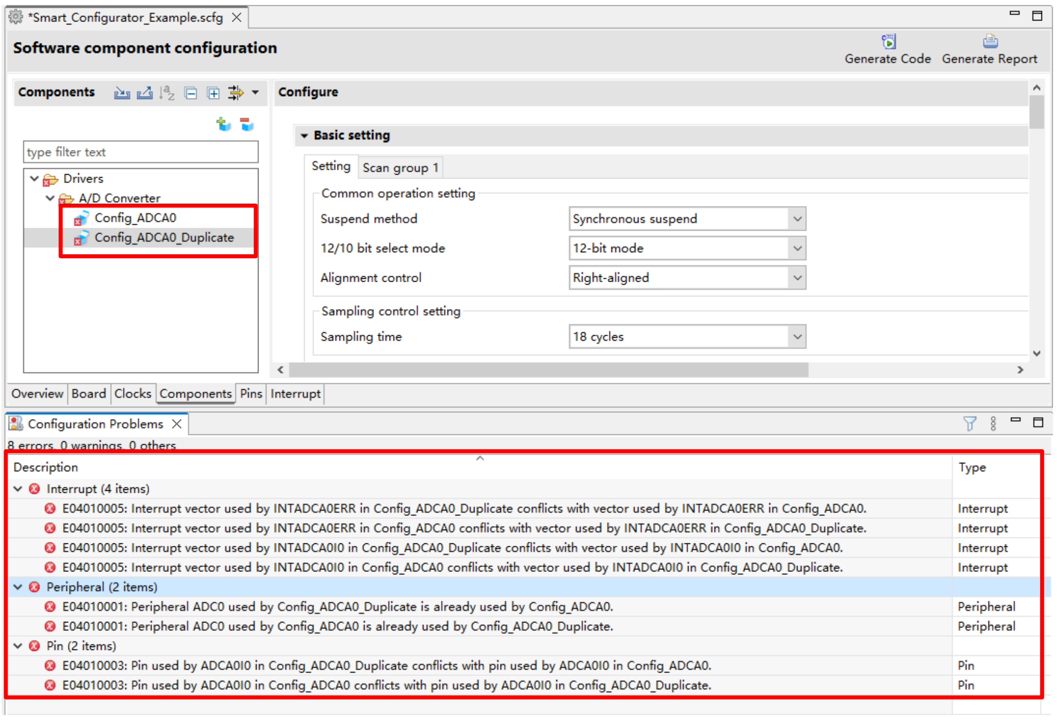
5. Managing Conflicts

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

5.1 Resource Conflicts

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

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



Common operation setting

Suspend method	Synchronous suspend
12/10 bit select mode	12-bit mode
Alignment control	Right-aligned

Sampling control setting

Sampling time	18 cycles

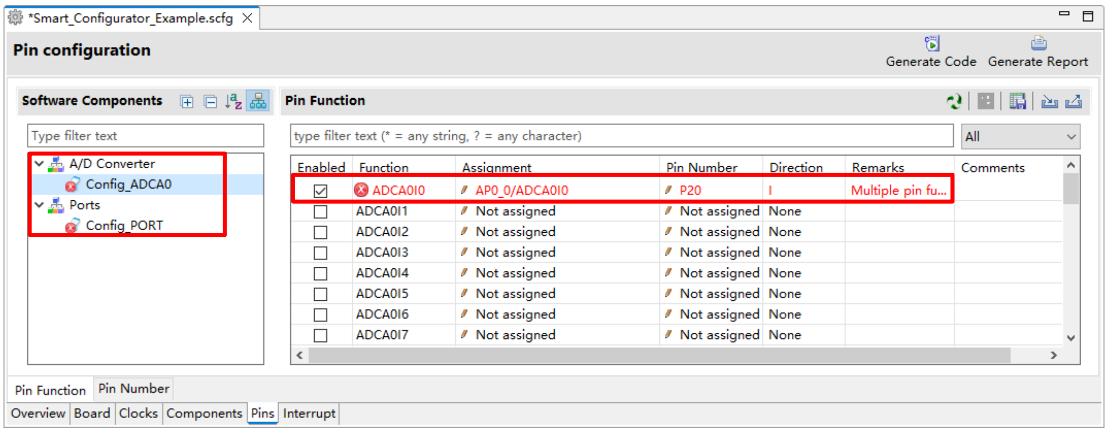
8 errors, 0 warnings, 0 others

Description	Type
Interrupt (4 items)	
E04010005: Interrupt vector used by INTADCA0ERR in Config_ADCA0_Duplicate conflicts with vector used by INTADCA0ERR in Config_ADCA0.	Interrupt
E04010005: Interrupt vector used by INTADCA0ERR in Config_ADCA0 conflicts with vector used by INTADCA0ERR in Config_ADCA0_Duplicate.	Interrupt
E04010005: Interrupt vector used by INTADCA0I0 in Config_ADCA0_Duplicate conflicts with vector used by INTADCA0I0 in Config_ADCA0.	Interrupt
E04010005: Interrupt vector used by INTADCA0I0 in Config_ADCA0 conflicts with vector used by INTADCA0I0 in Config_ADCA0_Duplicate.	Interrupt
Peripheral (2 items)	
E04010001: Peripheral ADC0 used by Config_ADCA0_Duplicate is already used by Config_ADCA0.	Peripheral
E04010001: Peripheral ADC0 used by Config_ADCA0 is already used by Config_ADCA0_Duplicate.	Peripheral
Pin (2 items)	
E04010003: Pin used by ADCA0I0 in Config_ADCA0_Duplicate conflicts with pin used by ADCA0I0 in Config_ADCA0.	Pin
E04010003: Pin used by ADCA0I0 in Config_ADCA0 conflicts with pin used by ADCA0I0 in Config_ADCA0_Duplicate.	Pin

Figure 5-1 Resource Conflicts

5.2 Resolving Pin Conflicts

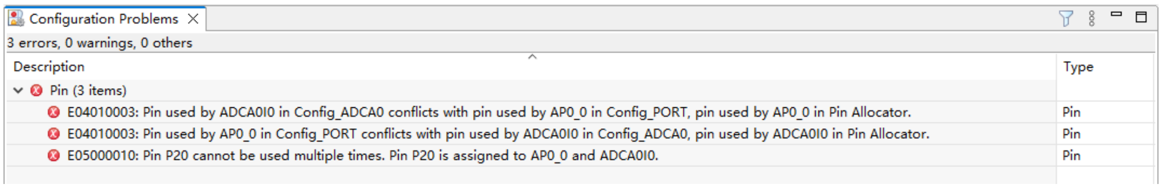
If there is a pin conflict, an error mark ⊗ will appear on the tree and [Pin Function] list.



Enabled	Function	Assignment	Pin Number	Direction	Remarks	Comments
☑	⊗ ADCA0I0	AP0_0/ADCA0I0	P20	I	Multiple pin fu...	
☐	ADCA0I1	Not assigned	Not assigned	None		
☐	ADCA0I2	Not assigned	Not assigned	None		
☐	ADCA0I3	Not assigned	Not assigned	None		
☐	ADCA0I4	Not assigned	Not assigned	None		
☐	ADCA0I5	Not assigned	Not assigned	None		
☐	ADCA0I6	Not assigned	Not assigned	None		
☐	ADCA0I7	Not assigned	Not assigned	None		

Figure 5-2 Pin Conflicts

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


Configuration Problems

3 errors, 0 warnings, 0 others

Description	Type
Pin (3 items)	
E04010003: Pin used by ADCA0I0 in Config_ADCA0 conflicts with pin used by AP0_0 in Config_PORT, pin used by AP0_0 in Pin Allocator.	Pin
E04010003: Pin used by AP0_0 in Config_PORT conflicts with pin used by ADCA0I0 in Config_ADCA0, pin used by ADCA0I0 in Pin Allocator.	Pin
E05000010: Pin P20 cannot be used multiple times. Pin P20 is assigned to AP0_0 and ADCA0I0.	Pin

Figure 5-3 Pin Conflict Messages

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

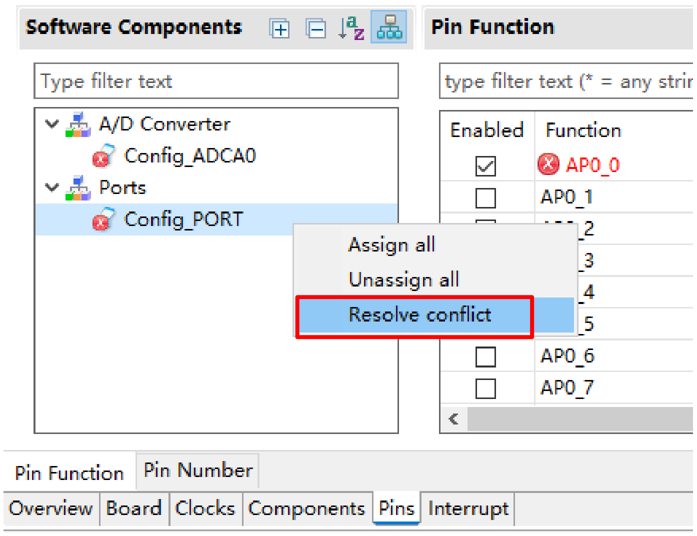


Figure 5-4 Resolving Pin Conflicts

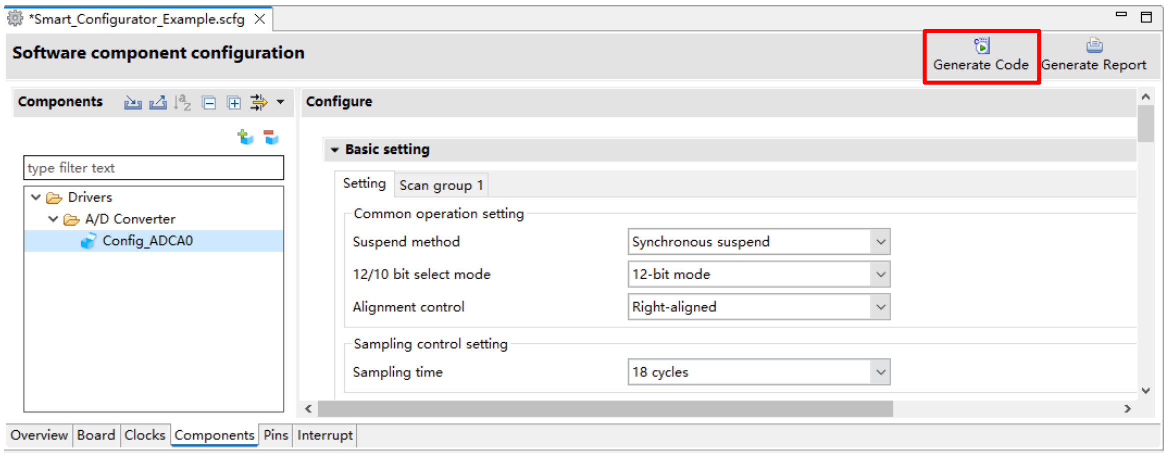
The pin function of the selected nodes will be re-assigned to other pins.

6. Generating Source Code

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

6.1 Outputting Generated Source Code

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



Common operation setting

Suspend method	Synchronous suspend
12/10 bit select mode	12-bit mode
Alignment control	Right-aligned

Sampling control setting

Sampling time	18 cycles

Figure 6-1 Generating a Source File

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

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

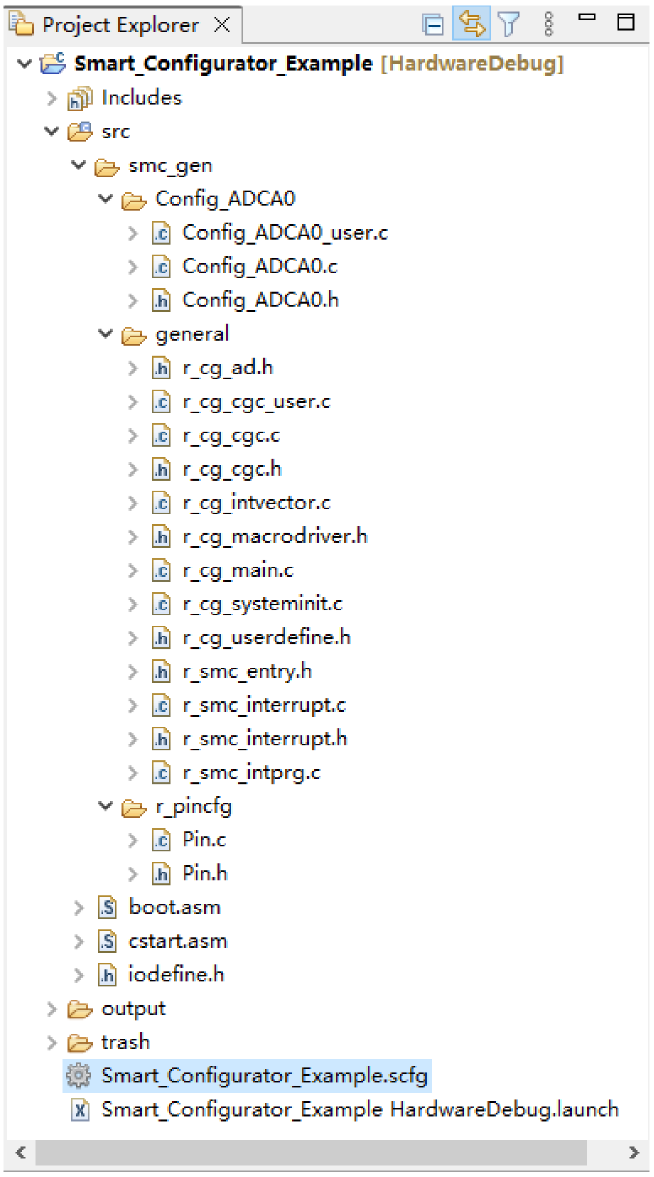
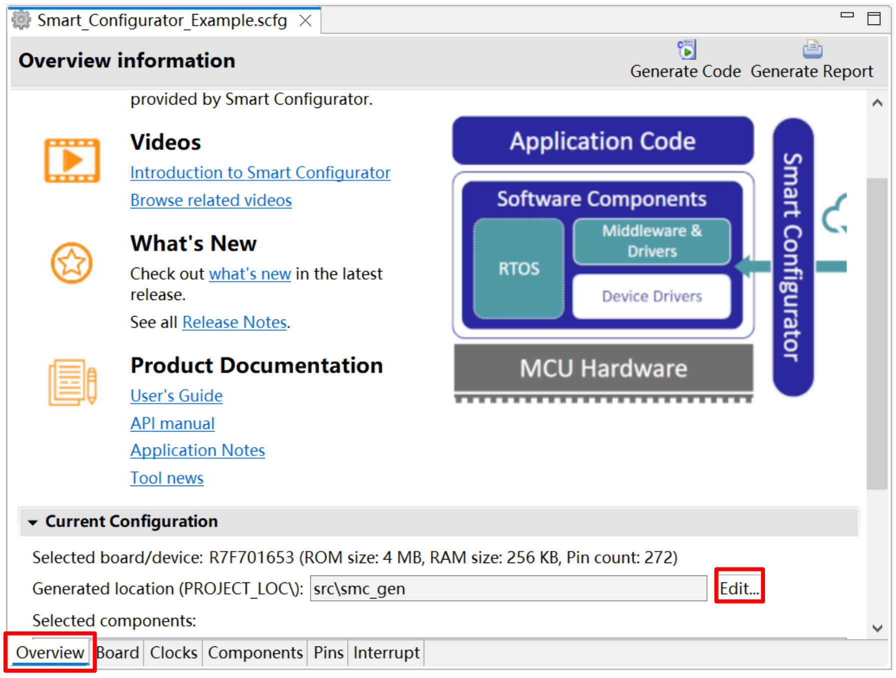

Project Explorer
Smart_Configurator_Example [HardwareDebug]
Includes
src
smc_gen
Config_ADCA0
Config_ADCA0_user.c
Config_ADCA0.c
Config_ADCA0.h
general
r_cg_ad.h
r_cg_cgc_user.c
r_cg_cgc.c
r_cg_cgc.h
r_cg_intvector.c
r_cg_macrodriver.h
r_cg_main.c
r_cg_systeminit.c
r_cg_userdefine.h
r_smc_entry.h
r_smc_interrupt.c
r_smc_interrupt.h
r_smc_intprg.c
r_pincfg
Pin.c
Pin.h
boot.asm
cstart.asm
iodefine.h
output
trash
Smart_Configurator_Example.scfg
Smart_Configurator_Example HardwareDebug.launch

Figure 6-2 Source Files in the Project Explorer

6.2 Change Generated Code Location

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



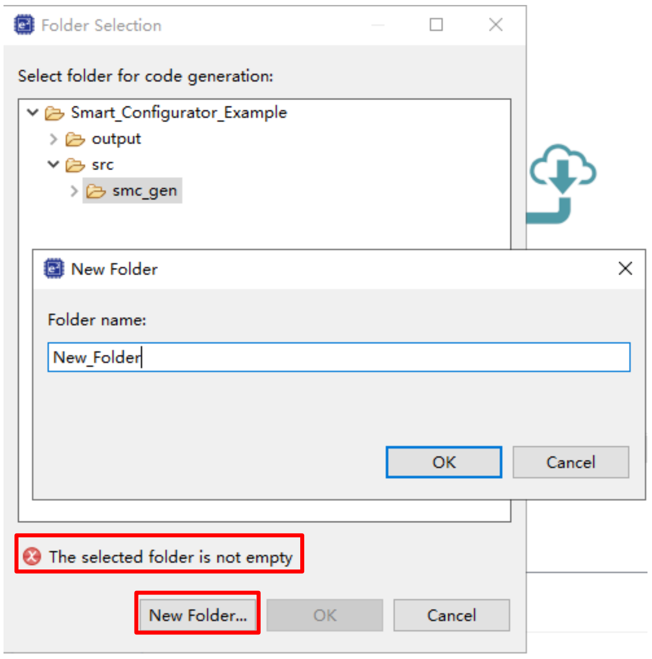
Selected board/device: R7F701653 (ROM size: 4 MB, RAM size: 256 KB, Pin count: 272)

Generated location (PROJECT_LOC\): src\smc_gen

Selected components:

Figure 6-3 Edit the Generated Code Location

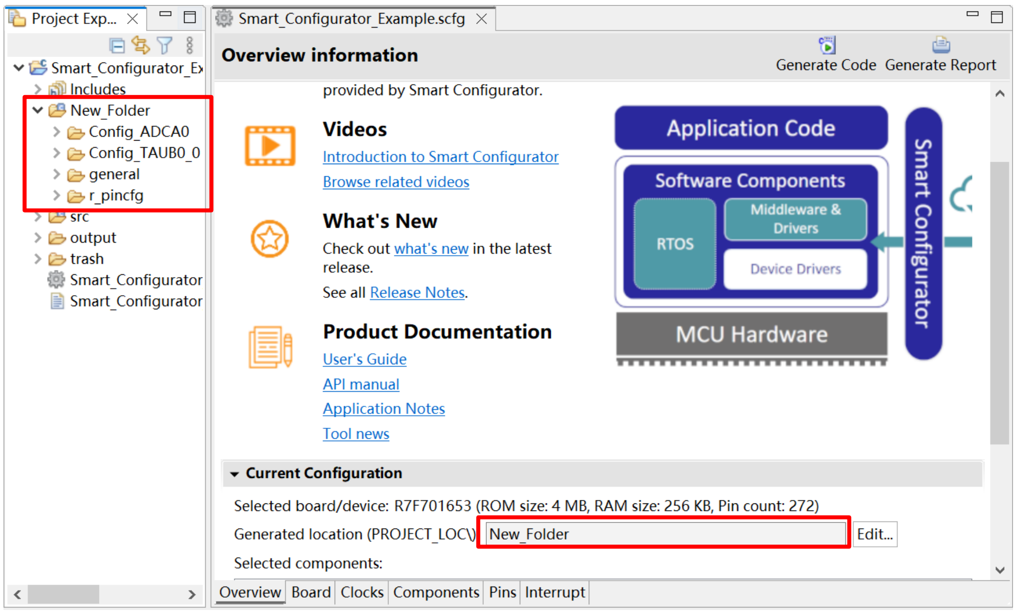
(2) In the Folder Selection dialog, select an empty folder for code generation or create a new folder.



The selected folder is not empty

Figure 6-4 Folder Selection

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



Selected board/device: R7F701653 (ROM size: 4 MB, RAM size: 256 KB, Pin count: 272)

Generated location (PROJECT_LOC\) New_Folder

Selected components:

Figure 6-5 New Generate Code Location

6.3 Configuration of Generated Files and File Names

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

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

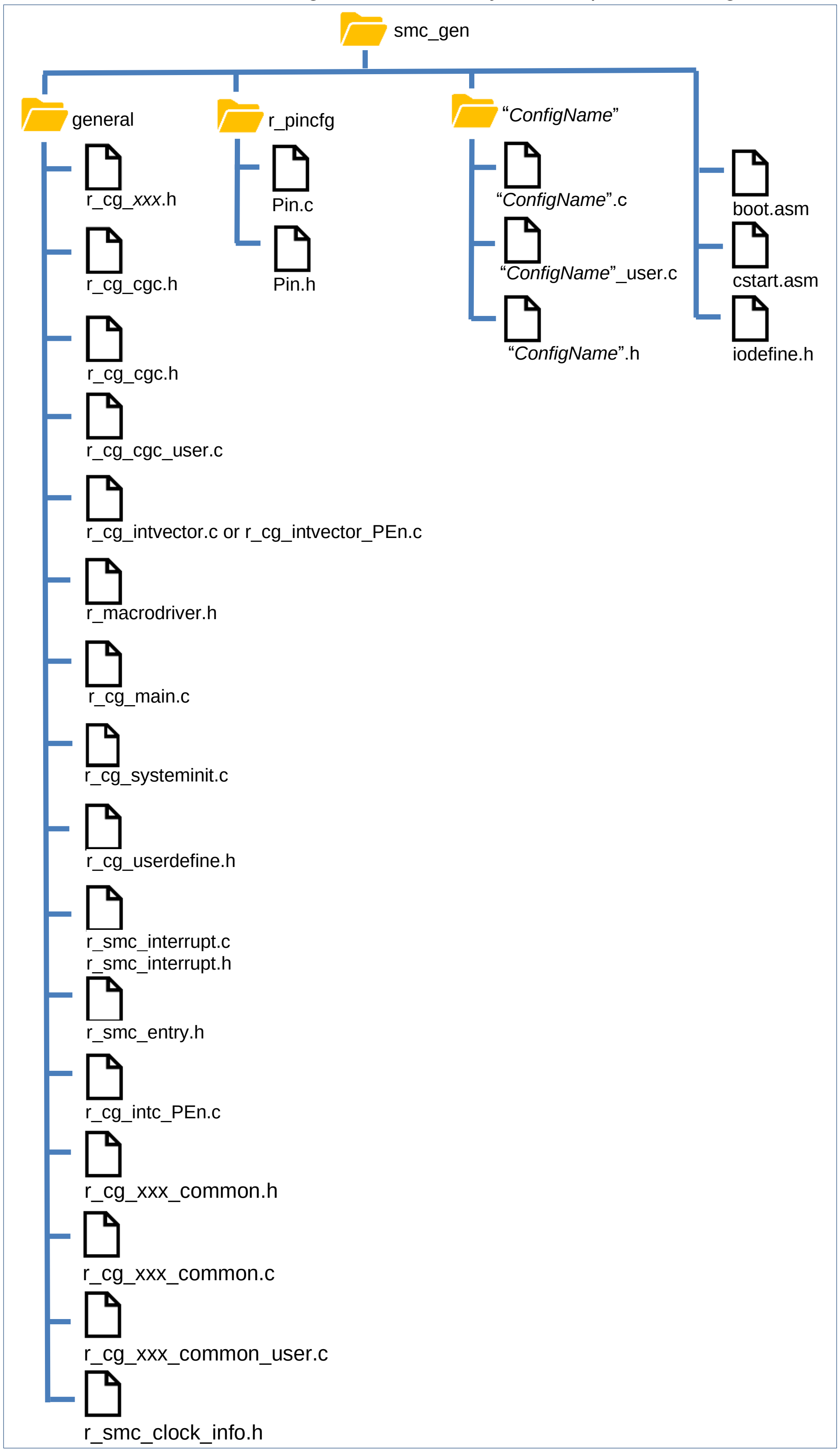


Figure 6-6 Configuration of Generated Files and File Names

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

Folder	File	Description
general	*r_cg_intc_PEn.c*	This file is generated only for: RH850/U2A: only when PE PEn(n=0~3) is chosen to be used, the r_cg_intc_PEn.c(n=0~3) is generated. RH850/C1M: r_cg_intc_PE1.c is generated. RH850/U2B: r_cg_intc_PE0.c is generated. This file contains interrupt initialization API definitions.
	r_cg_xxx_common_user.c(Note*1)	This file is generated only for components which have some common settings shared by all resources of the component. Normally, it contains the interrupt service routines for interrupts which are shared by multiple configurations. User can add codes and functions in the dedicated user code areas.
	r_cg_xxx_common.c(Note*1)	This file is generated only for components which have some common settings shared by all resources of the component. Normally, it contains the shared API for multiple configurations and will be called by users.
	r_cg_xxx_common.h(Note*1)	This is header file for r_cg_xxx_common.c and r_cg_xxx_common_user.c. It is generated only for components which have some common settings shared by all resources of the component. Normally, it contains the shared API declaration for multiple configurations.
	r_smc_clock_info.h(Note*2)	This file contains MCU information and clocks frequency define according to Clocks tabs setting.
r_pincfg	*Pin.h*	This file is always generated. It contains the function prototypes of pin settings in Pin.c, Symbolic name definition, Symbolic name user guide and Symbolic name API.
	Pin.c	This file is always generated. It is a reference of pin function initialization for all peripherals configured in the [Pins] tabbed page (except I/O Ports).
{ConfigName}	--	This folder is generated for the Code Generator drivers that are added to the project. API functions in this folder are named after the ConfigName (configuration name).
	{ConfigName}.c	This file contains functions to initialize driver (R_ConfigName_Create) and perform operations that are driver-specific, for e.g. start (R_ConfigName_Start) and stop (R_ConfigName_Stop).
	{ConfigName}_user.c	This file contains interrupt service routines and functions for user to add code after the driver initialization (R_ConfigName_Create). User can add codes and functions in the dedicated user code areas.
	{ConfigName}.h	This is header file for {ConfigName}.c and {ConfigName}_user.c

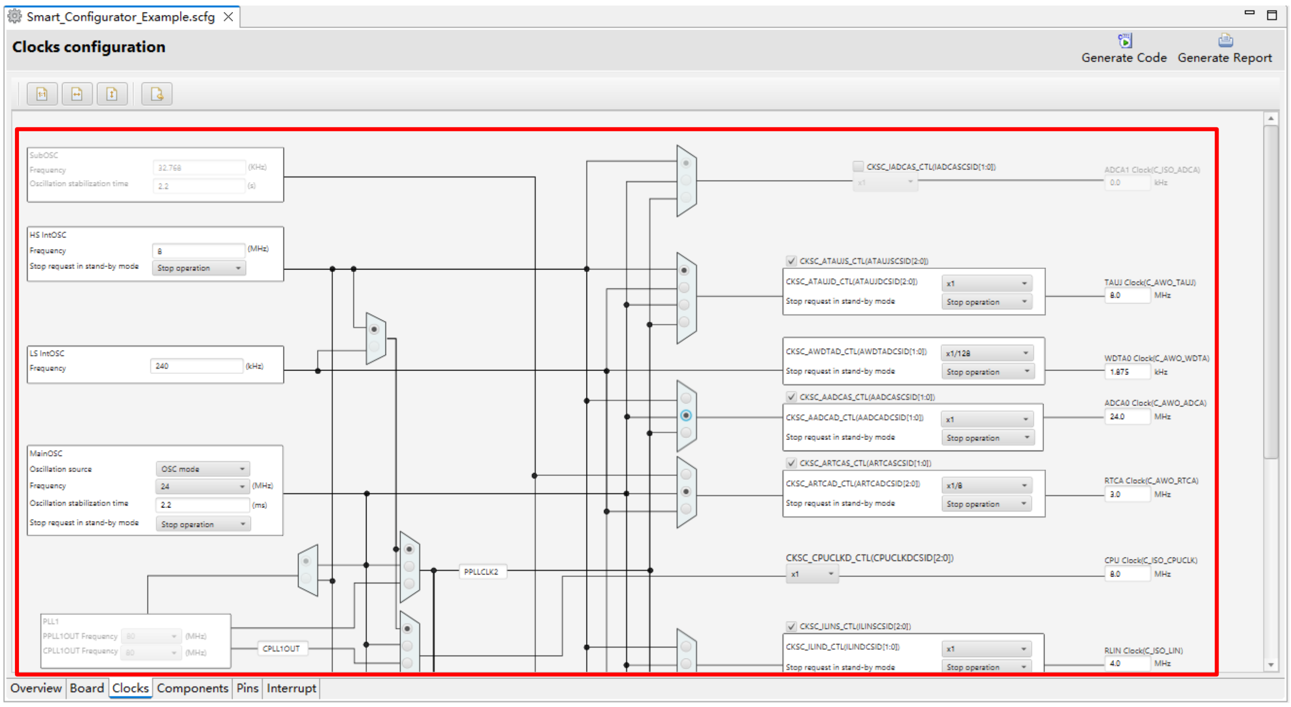
Folder	File	Description
smc_gen	-	This folder is always generated. It contains all the header files and source files generated by RH850 Smart Configurator.
	boot.asm	This file is automatically generated after creating the project. It contains some initialization assembly code required for device startup.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

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

6.4 Initializing Clocks

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

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



SubOSC
Frequency 32.768 (KHz)
Oscillation stabilization time 2.2 (s)

HS IntOSC
Frequency 8 (MHz)
Stop request in stand-by mode: Stop operation

LS IntOSC
Frequency 240 (kHz)

MainOSC
Oscillation source: OSC mode
Frequency: 24 (MHz)
Oscillation stabilization time 2.2 (ms)
Stop request in stand-by mode: Stop operation

PLL1
PPLL1OUT Frequency: 80 (MHz)
CPLL1OUT Frequency: 80 (MHz)

☐ CKSC_IADCAS_CTL(IADCASCSID[1:0]): x1
ADCA1 Clock(C_ISO_ADCA) 0.0 kHz

✓ CKSC_ATAUJS_CTL(ATAUJSCSID[2:0])
CKSC_ATAUJD_CTL(ATAUJDCSID[2:0]): x1
Stop request in stand-by mode: Stop operation
TAUJ Clock(C_AWO_TAUJ) 8.0 MHz

CKSC_AWDTAD_CTL(AWDTADCSID[1:0]): x1/128
Stop request in stand-by mode: Stop operation
WDTA0 Clock(C_AWO_WDTA) 1.875 kHz

✓ CKSC_AADCAS_CTL(AADCASCSID[1:0])
CKSC_AADCAD_CTL(AADCADCSID[1:0]): x1
Stop request in stand-by mode: Stop operation
ADCA0 Clock(C_AWO_ADCA) 24.0 MHz

✓ CKSC_ARTCAS_CTL(ARTCASCSID[1:0])
CKSC_ARTCAD_CTL(ARTCADCSID[2:0]): x1/8
Stop request in stand-by mode: Stop operation
RTCA Clock(C_AWO_RTCA) 3.0 MHz

CKSC_CPUCLKD_CTL(CPUCLKDCSID[2:0]): x1
CPU Clock(C_ISO_CPUCLK) 8.0 MHz

✓ CKSC_ILINS_CTL(ILINSCSID[2:0])
CKSC_ILIND_CTL(ILINDCSID[1:0]): x1
Stop request in stand-by mode: Stop operation
RLIN Clock(C_ISO_LIN) 4.0 MHz

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

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

6.5 Initializing Pins

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

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

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

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

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

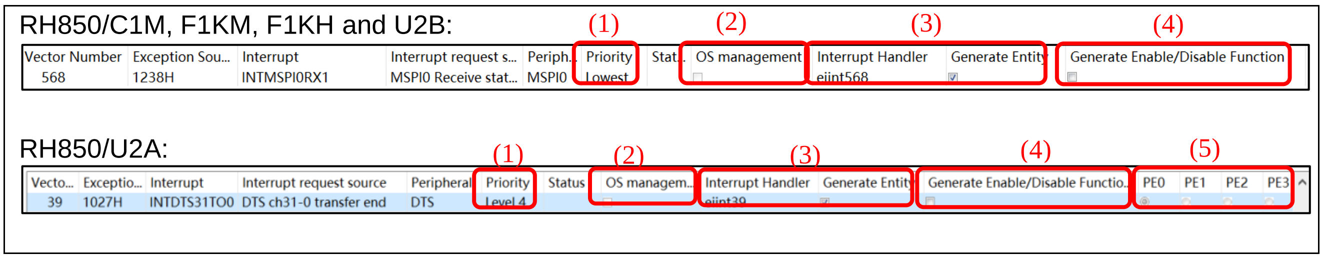
(2) Reference to pin initialization codes

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

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

6.6 Initializing Interrupts

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



RH850/C1M, F1KM, F1KH and U2B:

Vector Number	Exception Sou...	Interrupt	Interrupt request s...	Periph...	Priority	Stat...	OS management	Interrupt Handler	Generate Entity	Generate Enable/Disable Function
568	1238H	INTMSPI0RX1	MSPI0 Receive stat...	MSPI0	Lowest		☐	eiint568	✓	☐

RH850/U2A:

Vecto...	Exceptio...	Interrupt	Interrupt request source	Peripheral	Priority	Status	OS managem...	Interrupt Handler	Generate Entity	Generate Enable/Disable Functio...	PE0	PE1	PE2	PE3
39	1027H	INTDTS31TO0	DTS ch31-0 transfer end	DTS	Level 4		☐	eiint39	✓	☐	●	○	○	○

Figure 6-8 Interrupts Configuration in Interrupts View

RH850/C1M, F1KM, F1KH and U2B:

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

RH850/U2A:

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

7. Creating User Programs

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

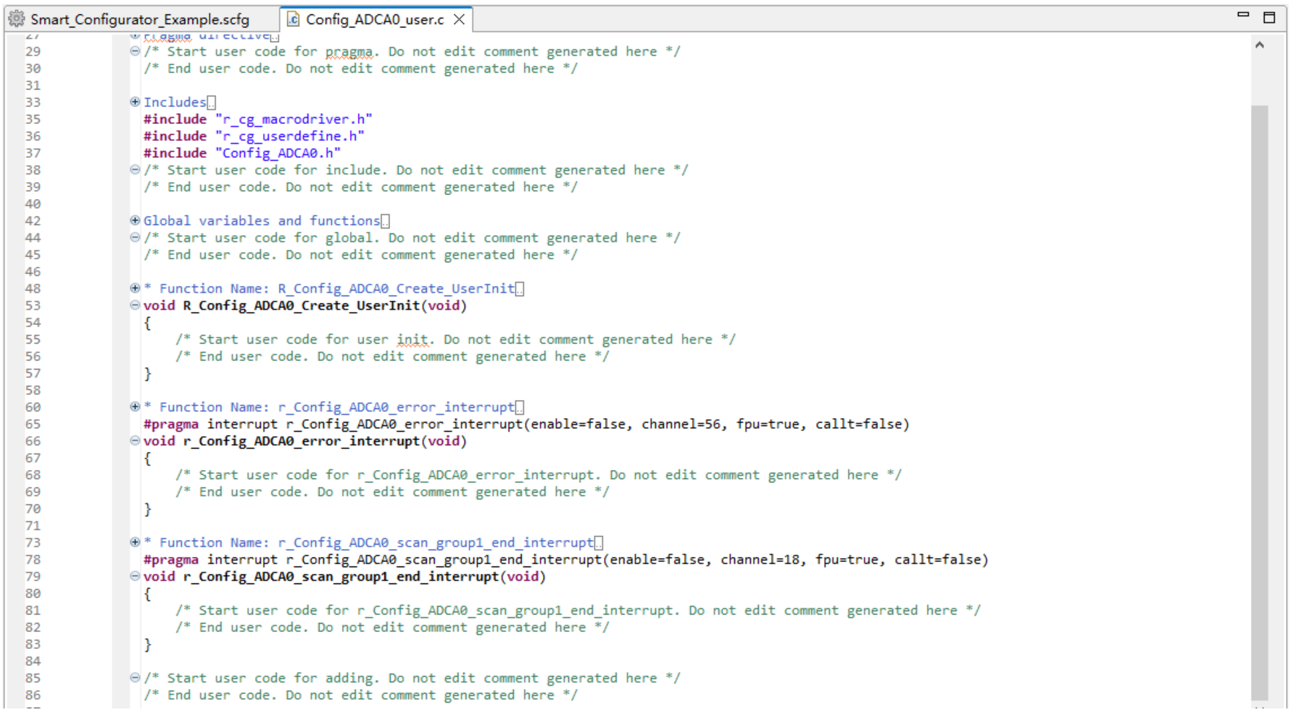
7.1 Adding Custom Code

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

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

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

Figure 7-1 User Code Comments

In the RH850 Smart Configurator, three files are generated for each of the specified peripheral functions. The file names are "Config_*xxx*.h", "Config_*xxx*.c", and "Config_*xxx*_user.c" as the default, with "*xxx*" representing the name of the peripheral module. For example, "xxx" will be "ADCA0" for the A/D Converter (resource ADCA0). The comments to indicate where to add custom code are at the start and end of *.c files, and at the end of *.h file. Comments to indicate where to add user code are also added to the interrupt function for the peripheral module corresponding to Config.*xxx*_user.c. The following example is for ADCA0 (Config_ADCA0_user.c).



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

Includes
#include "r_cg_macrodriver.h"
#include "r_cg_userdefine.h"
#include "Config_ADCA0.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_ADCA0_Create_UserInit
void R_Config_ADCA0_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_ADCA0_error_interrupt
#pragma interrupt r_Config_ADCA0_error_interrupt(enable=false, channel=56, fpu=true, callt=false)
void r_Config_ADCA0_error_interrupt(void)
{
    /* Start user code for r_Config_ADCA0_error_interrupt. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}

* Function Name: r_Config_ADCA0_scan_group1_end_interrupt
#pragma interrupt r_Config_ADCA0_scan_group1_end_interrupt(enable=false, channel=18, fpu=true, callt=false)
void r_Config_ADCA0_scan_group1_end_interrupt(void)
{
    /* Start user code for r_Config_ADCA0_scan_group1_end_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 */
```

Figure 7-2 ADCA0 Code Generated Example

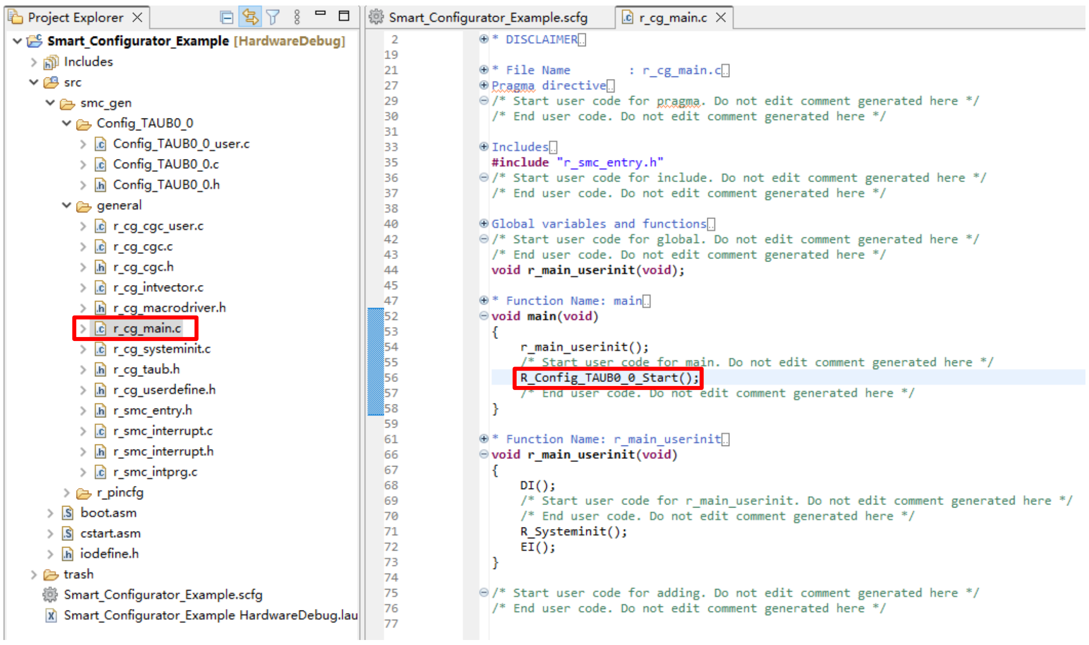
7.2 Using Generated Code in User Application

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

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

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

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



```
* DISCLAIMER

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

Includes
#include "r_smc_entry.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 */
void r_main_userinit(void);

* Function Name: main
void main(void)
{
    r_main_userinit();
    /* Start user code for main. Do not edit comment generated here */
    R_Config_TAUB0_0_Start();
    /* End user code. Do not edit comment generated here */
}

* Function Name: r_main_userinit
void r_main_userinit(void)
{
    DI();
    /* Start user code for r_main_userinit. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
    R_Systeminit();
    EI();
}

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

Figure 7-3 Call Code Generator Functions

8. Backing up Generated Source Code

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

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

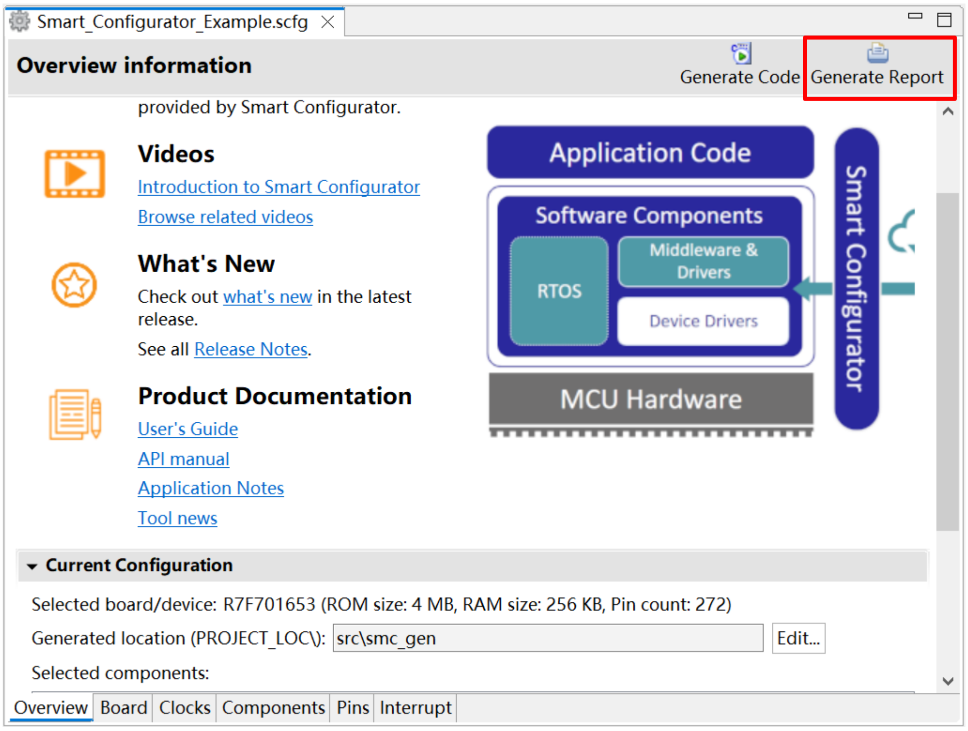
The Smart Configurator generates a backup folder for the previously generated source code when new code is generated by clicking on the [Generate Code (Generate Code)] button. <Date-and-Time> indicates the date and time when the backup folder is created after code generation.

9. Generating Reports

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

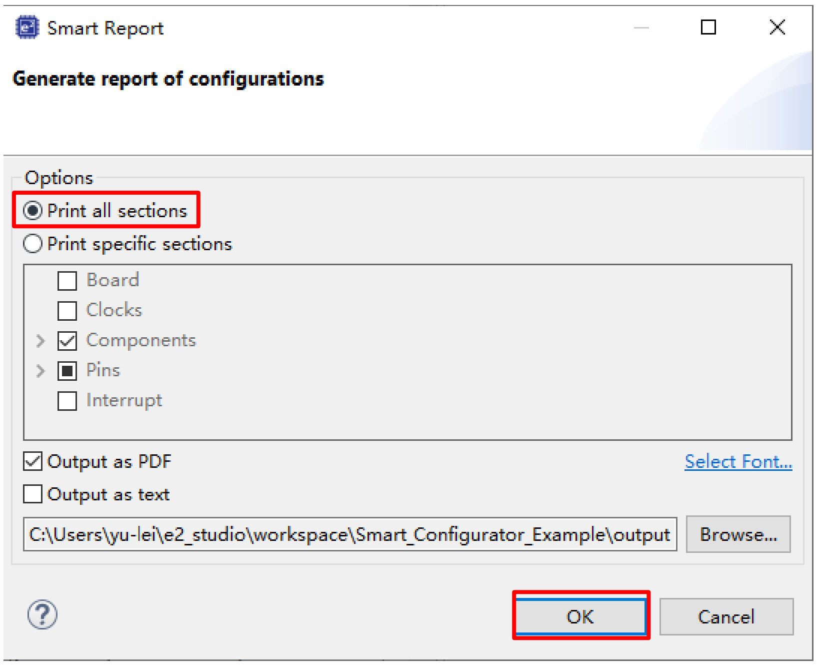
9.1 Report on All Configurations (PDF or Text File)

A report is output in response to clicking on the [Generate Report (Generate Report)] button in the Smart Configurator view. Two selections of output files are available (PDF, Text).



Selected board/device: R7F701653 (ROM size: 4 MB, RAM size: 256 KB, Pin count: 272)
Generated location (PROJECT_LOC\): src\smc_gen
Selected components:

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


Smart Report

Generate report of configurations

Options

[✓ Print all sections]
Print specific sections
- ☐ Board
- ☐ Clocks
- ☑ Components
- ■ Pins
- ☐ Interrupt

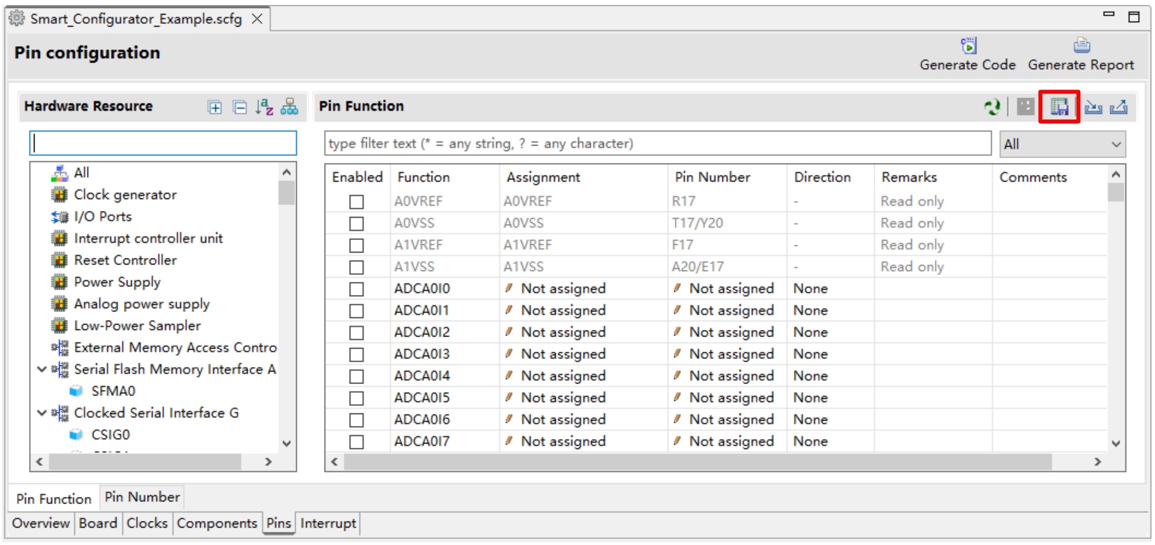
☑ Output as PDF
☐ Output as text

C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\output

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

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

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


Smart_Configurator_Example.scfg

Pin configuration

Hardware Resource

- All
- Clock generator
- I/O Ports
- Interrupt controller unit
- Reset Controller
- Power Supply
- Analog power supply
- Low-Power Sampler
- External Memory Access Contro
- Serial Flash Memory Interface A
  - SFMA0
- Clocked Serial Interface G
  - CSIG0

Pin Function

type filter text (* = any string, ? = any character) All

Enabled	Function	Assignment	Pin Number	Direction	Remarks	Comments
☐	A0VREF	A0VREF	R17	-	Read only	
☐	A0VSS	A0VSS	T17/Y20	-	Read only	
☐	A1VREF	A1VREF	F17	-	Read only	
☐	A1VSS	A1VSS	A20/E17	-	Read only	
☐	ADCA0I0	Not assigned	Not assigned	None		
☐	ADCA0I1	Not assigned	Not assigned	None		
☐	ADCA0I2	Not assigned	Not assigned	None		
☐	ADCA0I3	Not assigned	Not assigned	None		
☐	ADCA0I4	Not assigned	Not assigned	None		
☐	ADCA0I5	Not assigned	Not assigned	None		
☐	ADCA0I6	Not assigned	Not assigned	None		
☐	ADCA0I7	Not assigned	Not assigned	None		

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

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

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

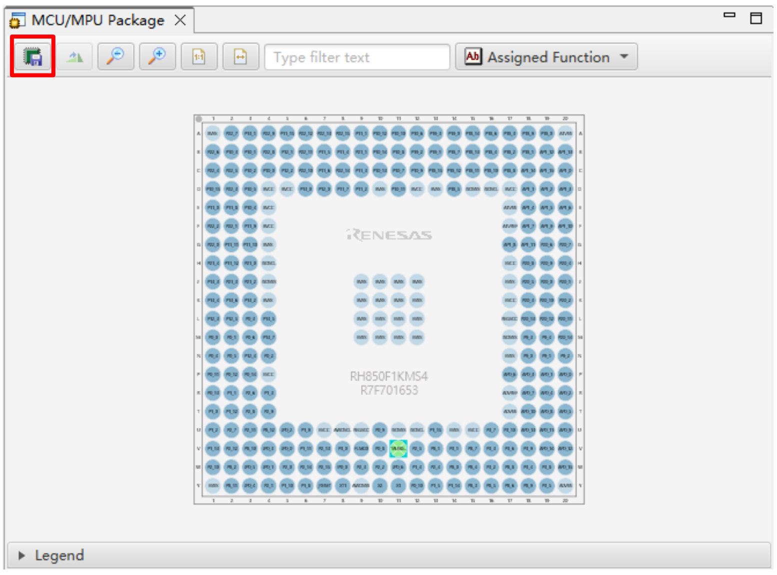


Figure 9-4 Outputting a Figure of MCU/MPU Package (in png Format)

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

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

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

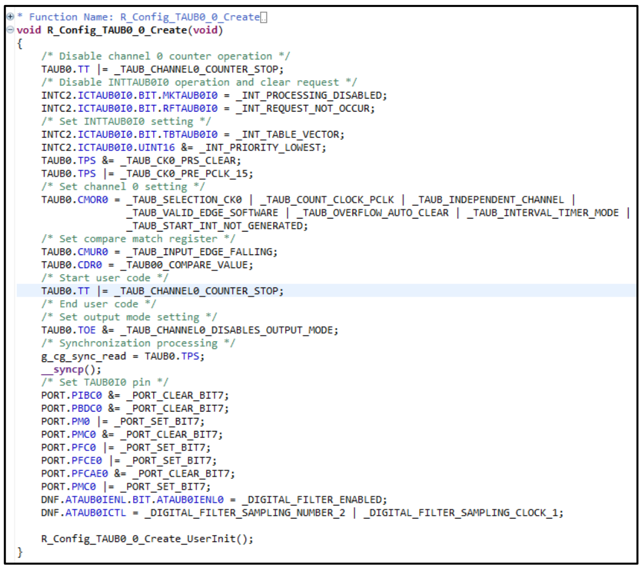
10.1 Specific Tags for the User Code Protection Feature

When using the user code protection feature, please insert */* Start user code */* and */* End user code */* as shown in Figure 10-1 and add the user codes between these tags. If the specific tags do not match exactly, the inserted user code will not be protected after the code generation.

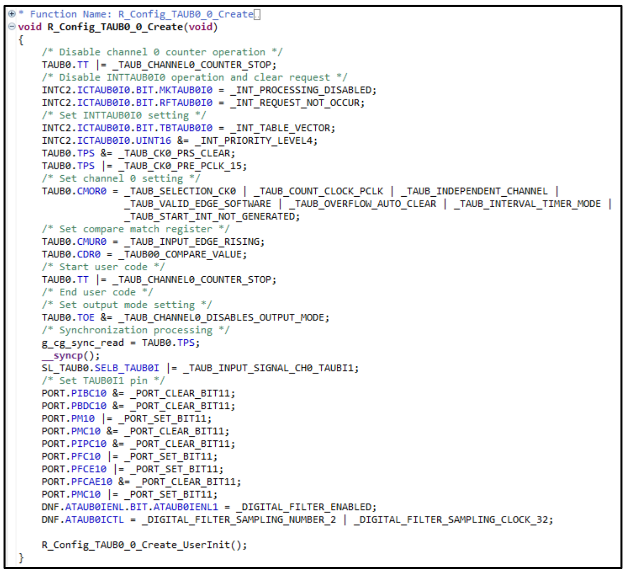
```
/* Start user code */

User code can be added between the specific tags

/* End user code */
```

Figure 10-1 Specific Tags for User Code Protection Feature

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

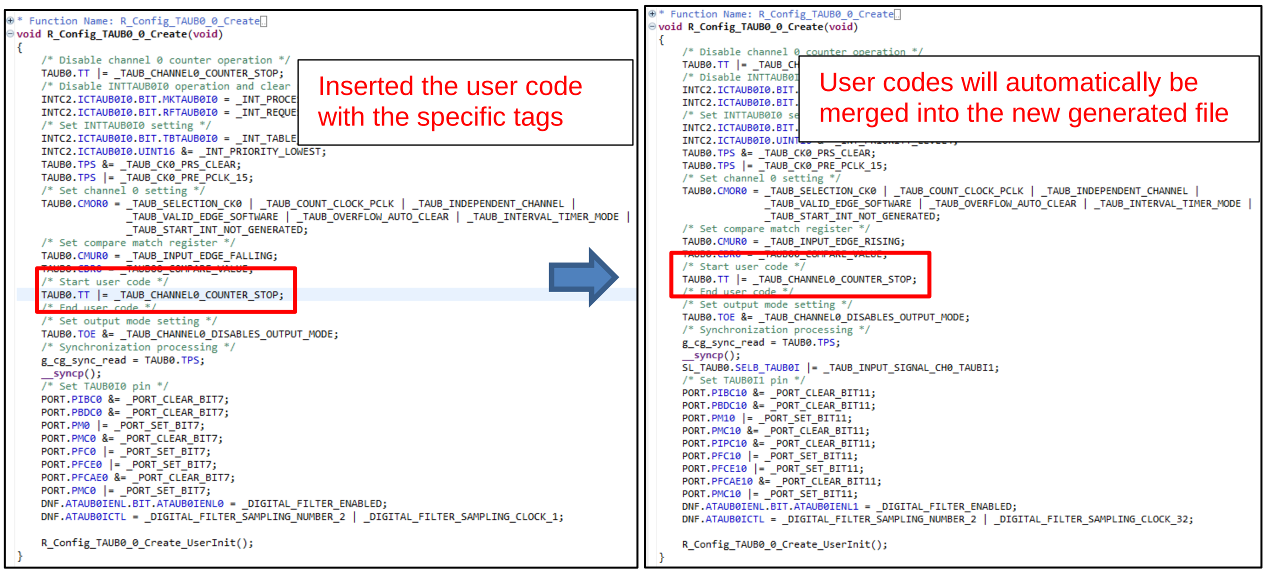


```
* Function Name: R_Config_TAUB0_0_Create
void R_Config_TAUB0_0_Create(void)
{
    /* Disable channel 0 counter operation */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* Disable INTTAUB0I0 operation and clear request */
    INTC2.ICTAUB0I0.BIT.MKTAUB0I0 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB0I0.BIT.RFTAUB0I0 = _INT_REQUEST_NOT_OCCUR;
    /* Set INTTAUB0I0 setting */
    INTC2.ICTAUB0I0.BIT.TBTAUB0I0 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I0.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_15;
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_INDEPENDENT_CHANNEL |
                  _TAUB_VALID_EDGE_SOFTWARE | _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE |
                  _TAUB_START_INT_NOT_GENERATED;
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_FALLING;
    TAUB0.CDR0 = _TAUB00_COMPARE_VALUE;
    /* Start user code */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* End user code */
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Synchronization processing */
    g_cg_sync_read = TAUB0.TPS;
    __syncp();
    /* Set TAUB0I0 pin */
    PORT.PIBC0 &= _PORT_CLEAR_BIT7;
    PORT.PBDC0 &= _PORT_CLEAR_BIT7;
    PORT.PM0 |= _PORT_SET_BIT7;
    PORT.PMC0 &= _PORT_CLEAR_BIT7;
    PORT.PFC0 |= _PORT_SET_BIT7;
    PORT.PFCE0 |= _PORT_SET_BIT7;
    PORT.PFCAE0 &= _PORT_CLEAR_BIT7;
    PORT.PMC0 |= _PORT_SET_BIT7;
    DNF.ATAUB0IENL.BIT.ATAUB0IENL0 = _DIGITAL_FILTER_ENABLED;
    DNF.ATAUB0ICTL = _DIGITAL_FILTER_SAMPLING_NUMBER_2 | _DIGITAL_FILTER_SAMPLING_CLOCK_1;

    R_Config_TAUB0_0_Create_UserInit();
}
```



```
* Function Name: R_Config_TAUB0_0_Create
void R_Config_TAUB0_0_Create(void)
{
    /* Disable channel 0 counter operation */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* Disable INTTAUB0I0 operation and clear request */
    INTC2.ICTAUB0I0.BIT.MKTAUB0I0 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB0I0.BIT.RFTAUB0I0 = _INT_REQUEST_NOT_OCCUR;
    /* Set INTTAUB0I0 setting */
    INTC2.ICTAUB0I0.BIT.TBTAUB0I0 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I0.UINT16 &= _INT_PRIORITY_LEVEL4;
    TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_15;
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_INDEPENDENT_CHANNEL |
                  _TAUB_VALID_EDGE_SOFTWARE | _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE |
                  _TAUB_START_INT_NOT_GENERATED;
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_RISING;
    TAUB0.CDR0 = _TAUB00_COMPARE_VALUE;
    /* Start user code */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* End user code */
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Synchronization processing */
    g_cg_sync_read = TAUB0.TPS;
    __syncp();
    SL_TAUB0.SELB_TAUB0I |= _TAUB_INPUT_SIGNAL_CH0_TAUBI1;
    /* Set TAUB0I1 pin */
    PORT.PIBC10 &= _PORT_CLEAR_BIT11;
    PORT.PBDC10 &= _PORT_CLEAR_BIT11;
    PORT.PM10 |= _PORT_SET_BIT11;
    PORT.PMC10 &= _PORT_CLEAR_BIT11;
    PORT.PIPC10 &= _PORT_CLEAR_BIT11;
    PORT.PFC10 |= _PORT_SET_BIT11;
    PORT.PFCE10 |= _PORT_SET_BIT11;
    PORT.PFCAE10 &= _PORT_CLEAR_BIT11;
    PORT.PMC10 |= _PORT_SET_BIT11;
    DNF.ATAUB0IENL.BIT.ATAUB0IENL1 = _DIGITAL_FILTER_ENABLED;
    DNF.ATAUB0ICTL = _DIGITAL_FILTER_SAMPLING_NUMBER_2 | _DIGITAL_FILTER_SAMPLING_CLOCK_32;

    R_Config_TAUB0_0_Create_UserInit();
}
```

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


Inserted the user code with the specific tags
User codes will automatically be merged into the new generated file

```
* Function Name: R_Config_TAUB0_0_Create
void R_Config_TAUB0_0_Create(void)
{
    /* Disable channel 0 counter operation */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* Disable INTTAUB0I0 operation and clear
    INTC2.ICTAUB0I0.BIT.MKTAUB0I0 = _INT_PROCE
    INTC2.ICTAUB0I0.BIT.RFTAUB0I0 = _INT_REQUE
    /* Set INTTAUB0I0 setting */
    INTC2.ICTAUB0I0.BIT.TBTAUB0I0 = _INT_TABLE
    INTC2.ICTAUB0I0.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_15;
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_INDEPENDENT_CHANNEL |
                  _TAUB_VALID_EDGE_SOFTWARE | _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE |
                  _TAUB_START_INT_NOT_GENERATED;
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_FALLING;
    TAUB0.CDR0 = _TAUB00_COMPARE_VALUE;
    /* Start user code */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* End user code */
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Synchronization processing */
    g_cg_sync_read = TAUB0.TPS;
    __syncp();
    /* Set TAUB0I0 pin */
    PORT.PIBC0 &= _PORT_CLEAR_BIT7;
    PORT.PBDC0 &= _PORT_CLEAR_BIT7;
    PORT.PM0 |= _PORT_SET_BIT7;
    PORT.PMC0 &= _PORT_CLEAR_BIT7;
    PORT.PFC0 |= _PORT_SET_BIT7;
    PORT.PFCE0 |= _PORT_SET_BIT7;
    PORT.PFCAE0 &= _PORT_CLEAR_BIT7;
    PORT.PMC0 |= _PORT_SET_BIT7;
    DNF.ATAUB0IENL.BIT.ATAUB0IENL0 = _DIGITAL_FILTER_ENABLED;
    DNF.ATAUB0ICTL = _DIGITAL_FILTER_SAMPLING_NUMBER_2 | _DIGITAL_FILTER_SAMPLING_CLOCK_1;

    R_Config_TAUB0_0_Create_UserInit();
}
```

```
* Function Name: R_Config_TAUB0_0_Create
void R_Config_TAUB0_0_Create(void)
{
    /* Disable channel 0 counter operation */
    TAUB0.TT |= _TAUB_CH
    /* Disable INTTAUB0I
    INTC2.ICTAUB0I0.BIT.
    INTC2.ICTAUB0I0.BIT.
    /* Set INTTAUB0I0 se
    INTC2.ICTAUB0I0.BIT.
    INTC2.ICTAUB0I0.UINT
    TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_15;
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_INDEPENDENT_CHANNEL |
                  _TAUB_VALID_EDGE_SOFTWARE | _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE |
                  _TAUB_START_INT_NOT_GENERATED;
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_RISING;
    TAUB0.CDR0 = _TAUB00_COMPARE_VALUE;
    /* Start user code */
    TAUB0.TT |= _TAUB_CHANNEL0_COUNTER_STOP;
    /* End user code */
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Synchronization processing */
    g_cg_sync_read = TAUB0.TPS;
    __syncp();
    SL_TAUB0.SELB_TAUB0I |= _TAUB_INPUT_SIGNAL_CH0_TAUBI1;
    /* Set TAUB0I1 pin */
    PORT.PIBC10 &= _PORT_CLEAR_BIT11;
    PORT.PBDC10 &= _PORT_CLEAR_BIT11;
    PORT.PM10 |= _PORT_SET_BIT11;
    PORT.PMC10 &= _PORT_CLEAR_BIT11;
    PORT.PIPC10 &= _PORT_CLEAR_BIT11;
    PORT.PFC10 |= _PORT_SET_BIT11;
    PORT.PFCE10 |= _PORT_SET_BIT11;
    PORT.PFCAE10 &= _PORT_CLEAR_BIT11;
    PORT.PMC10 |= _PORT_SET_BIT11;
    DNF.ATAUB0IENL.BIT.ATAUB0IENL1 = _DIGITAL_FILTER_ENABLED;
    DNF.ATAUB0ICTL = _DIGITAL_FILTER_SAMPLING_NUMBER_2 | _DIGITAL_FILTER_SAMPLING_CLOCK_32;

    R_Config_TAUB0_0_Create_UserInit();
}
```

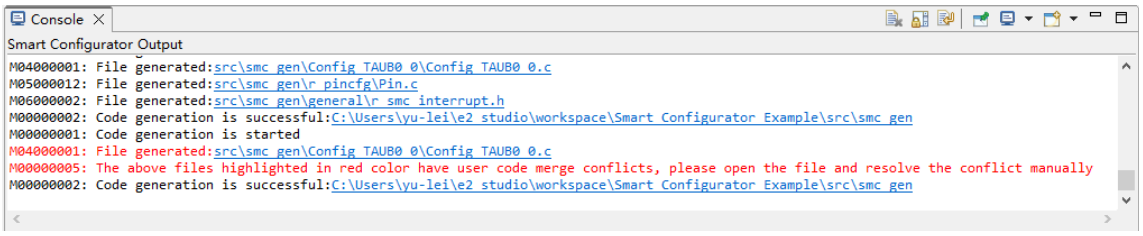
Figure 10-2 User Code Protection with Auto Merge

10.3 What to Do When Merge Conflict Occurs

10.3.1 What is Merge Conflict

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

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



```
M04000001: File generated:src\smc_gen\Config_TAUB0_0\Config_TAUB0_0.c
M05000012: File generated:src\smc_gen\r_pincfg\Pin.c
M06000002: File generated:src\smc_gen\general\r_smc_interrupt.h
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
M00000001: Code generation is started
M04000001: File generated:src\smc_gen\Config_TAUB0_0\Config_TAUB0_0.c
M00000005: The above files highlighted in red color have user code merge conflicts, please open the file and resolve the conflict manually
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
```

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

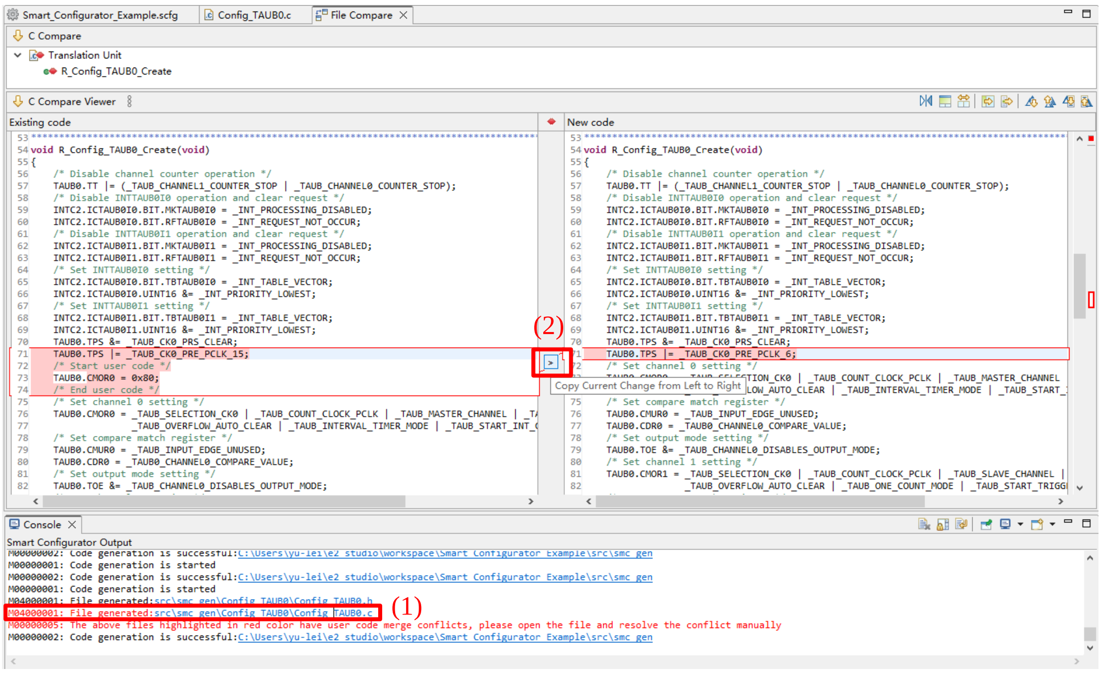
User can click the conflicted file in the console message to open the File Compare view and then can resolve the conflict as next chapter **10.3.2** Steps for Resolving the Merge Conflict described.

10.3.2 Steps for Resolving the Merge Conflict

User can follow the steps below to solve the merge conflicts.

(1) Click on the conflicting file in the console to open the “File Compare” view (Figure 10-4 Code before Resolving Conflict).

(2) Click on “Copy Current Change from Left to Right” (Figure 10-4).



Existing code:

```
*****************************************************************************
void R_Config_TAUB0_Create(void)
{
    /* Disable channel counter operation */
    TAUB0.TT |= (_TAUB_CHANNEL1_COUNTER_STOP | _TAUB_CHANNEL0_COUNTER_STOP);
    /* Disable INTTAUB0I0 operation and clear request */
    INTC2.ICTAUB0I0.BIT.MKTAUB0I0 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB0I0.BIT.RFTAUB0I0 = _INT_REQUEST_NOT_OCCUR;
    /* Disable INTTAUB0I1 operation and clear request */
    INTC2.ICTAUB0I1.BIT.MKTAUB0I1 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB0I1.BIT.RFTAUB0I1 = _INT_REQUEST_NOT_OCCUR;
    /* Set INTTAUB0I0 setting */
    INTC2.ICTAUB0I0.BIT.TBTAUB0I0 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I0.UINT16 &= _INT_PRIORITY_LOWEST;
    /* Set INTTAUB0I1 setting */
    INTC2.ICTAUB0I1.BIT.TBTAUB0I1 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I1.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_15;
    /* Start user code */
    TAUB0.CMOR0 = 0x80;
    /* End user code */
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_MASTER_CHANNEL | _TA
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE | _TAUB_START_INT_
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR0 = _TAUB0_CHANNEL0_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
```

New code:

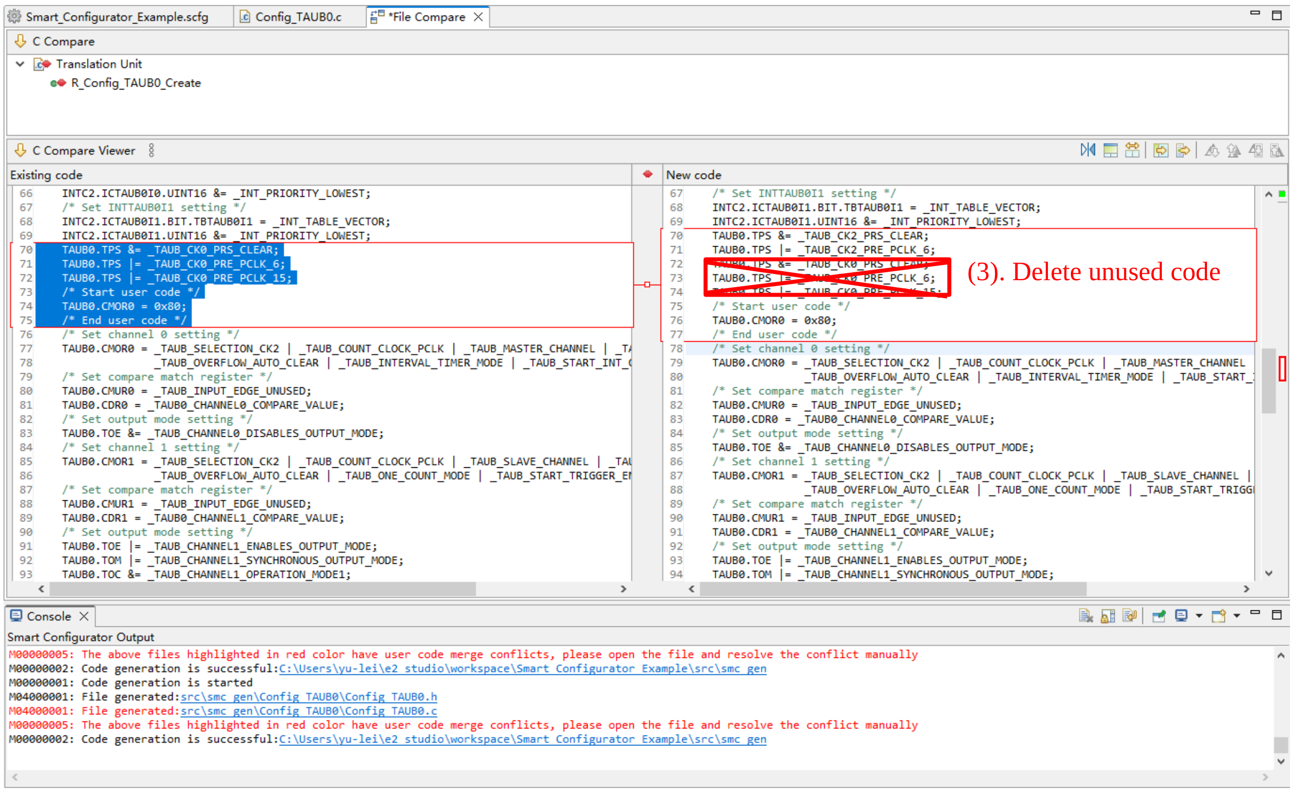
```
*****************************************************************************
void R_Config_TAUB0_Create(void)
{
    /* Disable channel counter operation */
    TAUB0.TT |= (_TAUB_CHANNEL1_COUNTER_STOP | _TAUB_CHANNEL0_COUNTER_STOP);
    /* Disable INTTAUB0I0 operation and clear request */
    INTC2.ICTAUB0I0.BIT.MKTAUB0I0 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB0I0.BIT.RFTAUB0I0 = _INT_REQUEST_NOT_OCCUR;
    /* Disable INTTAUB0I1 operation and clear request */
    INTC2.ICTAUB0I1.BIT.MKTAUB0I1 = _INT_PROCESSING_DISABLED;
    INTC2.ICTAUB0I1.BIT.RFTAUB0I1 = _INT_REQUEST_NOT_OCCUR;
    /* Set INTTAUB0I0 setting */
    INTC2.ICTAUB0I0.BIT.TBTAUB0I0 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I0.UINT16 &= _INT_PRIORITY_LOWEST;
    /* Set INTTAUB0I1 setting */
    INTC2.ICTAUB0I1.BIT.TBTAUB0I1 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I1.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_6;
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_MASTER_CHANNEL
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE | _TAUB_START_
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR0 = _TAUB0_CHANNEL0_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Set channel 1 setting */
    TAUB0.CMOR1 = _TAUB_SELECTION_CK0 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_SLAVE_CHANNEL |
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_ONE_COUNT_MODE | _TAUB_START_TRIGG
```

Console:

```
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
M00000001: Code generation is started
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
M00000001: Code generation is started
M04000001: File generated:src\smc_gen\Config_TAUB0\Config_TAUB0.h
M04000001: File generated:src\smc_gen\Config_TAUB0\Config_TAUB0.c
M00000005: The above files highlighted in red color have user code merge conflicts, please open the file and resolve the conflict manually
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
```

Figure 10-4 Code before Resolving Conflict

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



Existing code:

```
    INTC2.ICTAUB0I0.UINT16 &= _INT_PRIORITY_LOWEST;
    /* Set INTTAUB0I1 setting */
    INTC2.ICTAUB0I1.BIT.TBTAUB0I1 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I1.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_6;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_15;
    /* Start user code */
    TAUB0.CMOR0 = 0x80;
    /* End user code */
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK2 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_MASTER_CHANNEL | _TA
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE | _TAUB_START_INT_
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR0 = _TAUB0_CHANNEL0_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Set channel 1 setting */
    TAUB0.CMOR1 = _TAUB_SELECTION_CK2 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_SLAVE_CHANNEL | _TAU
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_ONE_COUNT_MODE | _TAUB_START_TRIGGER_EN
    /* Set compare match register */
    TAUB0.CMUR1 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR1 = _TAUB0_CHANNEL1_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE |= _TAUB_CHANNEL1_ENABLES_OUTPUT_MODE;
    TAUB0.TOM |= _TAUB_CHANNEL1_SYNCHRONOUS_OUTPUT_MODE;
    TAUB0.TOC &= _TAUB_CHANNEL1_OPERATION_MODE1;
```

New code:

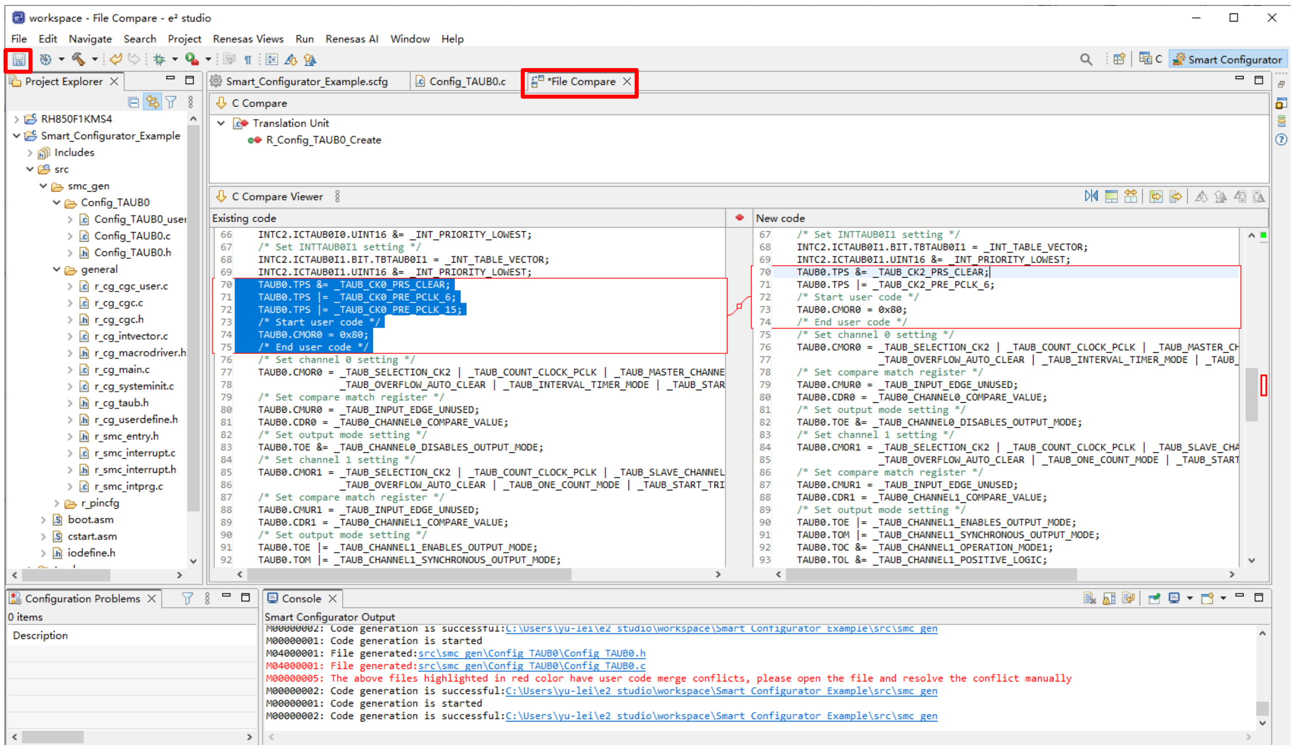
```
    /* Set INTTAUB0I1 setting */
    INTC2.ICTAUB0I1.BIT.TBTAUB0I1 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I1.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK2_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK2_PRE_PCLK_6;
    ~~TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;~~
    ~~TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_6;~~
    ~~TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_15;~~
    /* Start user code */
    TAUB0.CMOR0 = 0x80;
    /* End user code */
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK2 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_MASTER_CHANNEL
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE | _TAUB_START_
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR0 = _TAUB0_CHANNEL0_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Set channel 1 setting */
    TAUB0.CMOR1 = _TAUB_SELECTION_CK2 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_SLAVE_CHANNEL |
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_ONE_COUNT_MODE | _TAUB_START_TRIGG
    /* Set compare match register */
    TAUB0.CMUR1 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR1 = _TAUB0_CHANNEL1_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE |= _TAUB_CHANNEL1_ENABLES_OUTPUT_MODE;
    TAUB0.TOM |= _TAUB_CHANNEL1_SYNCHRONOUS_OUTPUT_MODE;
```

Console:

```
M00000005: The above files highlighted in red color have user code merge conflicts, please open the file and resolve the conflict manually
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
M00000001: Code generation is started
M04000001: File generated:src\smc_gen\Config_TAUB0\Config_TAUB0.h
M04000001: File generated:src\smc_gen\Config_TAUB0\Config_TAUB0.c
M00000005: The above files highlighted in red color have user code merge conflicts, please open the file and resolve the conflict manually
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
```

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

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



Existing code:

```
    INTC2.ICTAUB0I0.UINT16 &= _INT_PRIORITY_LOWEST;
    /* Set INTTAUB0I1 setting */
    INTC2.ICTAUB0I1.BIT.TBTAUB0I1 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I1.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK0_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_6;
    TAUB0.TPS |= _TAUB_CK0_PRE_PCLK_15;
    /* Start user code */
    TAUB0.CMOR0 = 0x80;
    /* End user code */
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK2 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_MASTER_CHANNE
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE | _TAUB_STAR
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR0 = _TAUB0_CHANNEL0_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Set channel 1 setting */
    TAUB0.CMOR1 = _TAUB_SELECTION_CK2 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_SLAVE_CHANNEL
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_ONE_COUNT_MODE | _TAUB_START_TRI
    /* Set compare match register */
    TAUB0.CMUR1 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR1 = _TAUB0_CHANNEL1_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE |= _TAUB_CHANNEL1_ENABLES_OUTPUT_MODE;
    TAUB0.TOM |= _TAUB_CHANNEL1_SYNCHRONOUS_OUTPUT_MODE;
```

New code:

```
    /* Set INTTAUB0I1 setting */
    INTC2.ICTAUB0I1.BIT.TBTAUB0I1 = _INT_TABLE_VECTOR;
    INTC2.ICTAUB0I1.UINT16 &= _INT_PRIORITY_LOWEST;
    TAUB0.TPS &= _TAUB_CK2_PRS_CLEAR;
    TAUB0.TPS |= _TAUB_CK2_PRE_PCLK_6;
    /* Start user code */
    TAUB0.CMOR0 = 0x80;
    /* End user code */
    /* Set channel 0 setting */
    TAUB0.CMOR0 = _TAUB_SELECTION_CK2 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_MASTER_CH
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_INTERVAL_TIMER_MODE | _TAUB_
    /* Set compare match register */
    TAUB0.CMUR0 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR0 = _TAUB0_CHANNEL0_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE &= _TAUB_CHANNEL0_DISABLES_OUTPUT_MODE;
    /* Set channel 1 setting */
    TAUB0.CMOR1 = _TAUB_SELECTION_CK2 | _TAUB_COUNT_CLOCK_PCLK | _TAUB_SLAVE_CHA
                  _TAUB_OVERFLOW_AUTO_CLEAR | _TAUB_ONE_COUNT_MODE | _TAUB_START
    /* Set compare match register */
    TAUB0.CMUR1 = _TAUB_INPUT_EDGE_UNUSED;
    TAUB0.CDR1 = _TAUB0_CHANNEL1_COMPARE_VALUE;
    /* Set output mode setting */
    TAUB0.TOE |= _TAUB_CHANNEL1_ENABLES_OUTPUT_MODE;
    TAUB0.TOM |= _TAUB_CHANNEL1_SYNCHRONOUS_OUTPUT_MODE;
    TAUB0.TOC &= _TAUB_CHANNEL1_OPERATION_MODE1;
    TAUB0.TOL &= _TAUB_CHANNEL1_POSITIVE_LOGIC;
```

```
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
M00000001: Code generation is started
M04000001: File generated:src\smc_gen\Config_TAUB0\Config_TAUB0.h
M04000001: File generated:src\smc_gen\Config_TAUB0\Config_TAUB0.c
M00000005: The above files highlighted in red color have user code merge conflicts, please open the file and resolve the conflict manually
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
M00000001: Code generation is started
M00000002: Code generation is successful:C:\Users\yu-lei\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen
```

Figure 10-6 Code after Deleting and Saving

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

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

11. Help

Refer to the help system from the e^2 studio menu for detailed information on the Smart Configurator. If selected from Help menu, it will prompt out the Help dialog window, it shows all Help topics content.

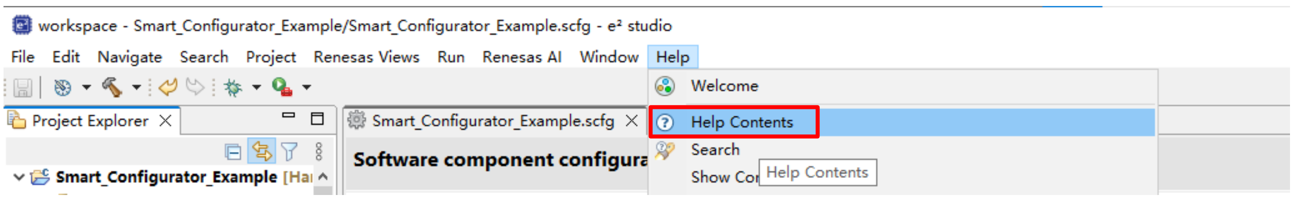
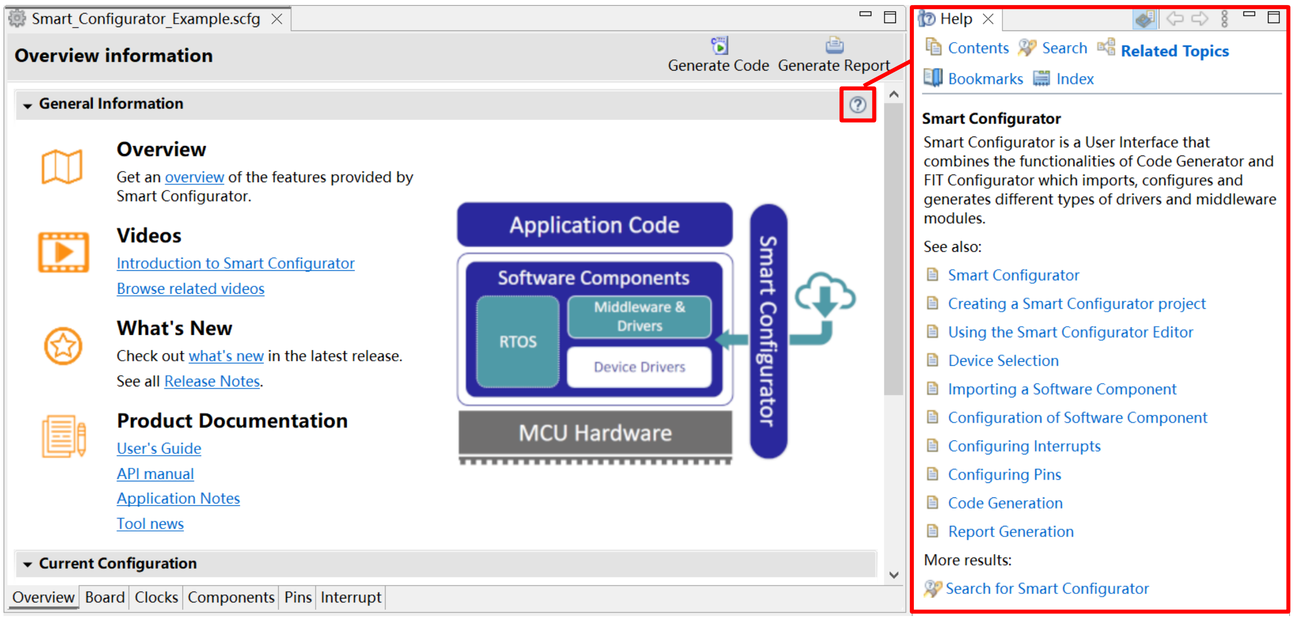

workspace - Smart_Configurator_Example/Smart_Configurator_Example.scfg - e² studio
File Edit Navigate Search Project Renesas Views Run Renesas AI Window Help
Project Explorer
Smart_Configurator_Example
Smart_Configurator_Example.scfg
Software component configura
Welcome
Help Contents
Search
Show Co
Help Contents

Figure 11-1 Help Menu

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



Smart Configurator

Smart Configurator is a User Interface that combines the functionalities of Code Generator and FIT Configurator which imports, configures and generates different types of drivers and middleware modules.

See also:

- Smart Configurator
- Creating a Smart Configurator project
- Using the Smart Configurator Editor
- Device Selection
- Importing a Software Component
- Configuration of Software Component
- Configuring Interrupts
- Configuring Pins
- Code Generation
- Report Generation

More results:

Search for Smart Configurator

Figure 11-2 Smart Configurator Help Quick Start

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

12. Documents for Reference

User’s Manual: Hardware

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

Technical Update/Technical News

Obtain the latest information from the Renesas Electronics website.

User’s Manual: Development Environment

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

CC-RH Compiler User’s Manual (R20UT3516)

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

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

Revision History

Rev.	Section	Description
1.00	-	First edition issued

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Corporate Headquarters

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

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/.

Trademarks