RL78 Software Porting Guide

RL78/G13 sample code porting (CC-RL) (CS+, e2 studio)

Introduction

This application note describes how to port the RL78/G13 peripheral sample code to another RL78.

Target Device

RL78 Family

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

1. Overview

1.1 Target for Porting

Most of the sample codes for RL78/G13 peripheral functions use the program (device driver) generated by Code Generator (CG). This application note describes the procedure for porting the device driver generated by CG to the device driver for other RL78 product.

However, if the RL78 product as the porting device does not have the same peripheral functions as the RL78/G13, the device driver cannot be ported.
When the device driver cannot be ported, create a new project for the target RL78 product and do programming newly.

The RL78 products as the porting device is shown in Table 1-1.

Table 1-1 Porting Target RL78 Products

RL78/G1x	RL78/G11, RL78/G12, RL78/G14, RL78/G13A, RL78/G1A, RL78/G1C, RL78/G1E, RL78/G1F, RL78/G1G, RL78/G1H
RL78/H1x	RL78/H1D
RL78/I1x	RL78/I1A, RL78/I1B, RL78/I1C, RL78/I1D, RL78/I1E
RL78/L1x	RL78/L12, RL78/L13, RL78L1A, RL78/L1C

Table 1-2 Criteria for Determining Whether Porting is Possible

Porting Target RL78 Product	Possible/ Impossible to Port
The target RL78 product does not have the same peripheral functions as the source product.	Impossible
The target RL78 product has the same peripheral functions and the same channels as the source product.	Possible
The target RL78 product has the same peripheral functions as the source product but not the same channels as the source product.	Possible if the channel can be changed
The operating voltage of the target RL78 product differs from that of the source product.	Possible if the operating voltage can be set within the range of operation [Note]
The operating frequency of the target RL78 product differs from that of the source product.	Possible if the operating frequency can be set within the range of operation [Note]

Note. When creating actual circuits, design them to meet the electrical characteristics of the porting RL78 product.

1.2 Sample Code Structure

The sample code of an RL78 product consists of files generated by Code Generator (CG) (CG generated files) and other files (non-CG generated files). CG generated files contain code that configures the specific peripheral functions for an RL78 product managed by a project.

Figure 1-1 shows an example project containing sample code. Table 1-3 provides an overview of CG generated files and functions. Figure 1-2 shows an overview of processing from occurrence of CPU reset to calling of the main function.

Figure 1-1 Example of Sample Code Project

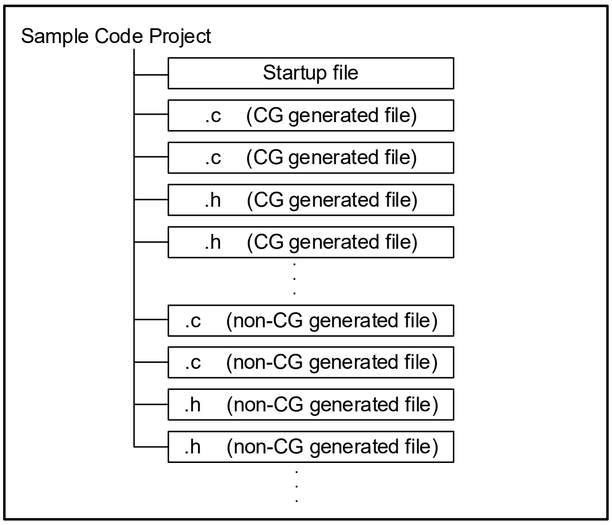


Table 1-3 Outline of CG generated files and functions

File Name [Note 1]	Function Name [Note 1]	Description
r_main.c	main R_MAIN_Userinit	main function Processing before main function
r_systeminit.c	hdwinit R_Systeminit	Calling R_Systeminit Calling peripheral functions initial setting processing
r_cg_macrodriver.h	-	Macro and Typedef definitions used in common by CG generated files
r_cg_userdefine.h	-	User-defined macros, Typedefs, etc
r_cg_xxx.c [Note 2]	R_XXX_Create R_XXX_Start [Note 3] R_XXX_Stop [Note 3]	Peripheral function initial setting processing Starting peripheral function operation Stopping peripheral function operation
r_cg_xxx_user.c	r_xxx_interrupt	Interrupt function of peripheral function
r_cg_xxx.h	-	Macro and Typedef definitions for peripheral function

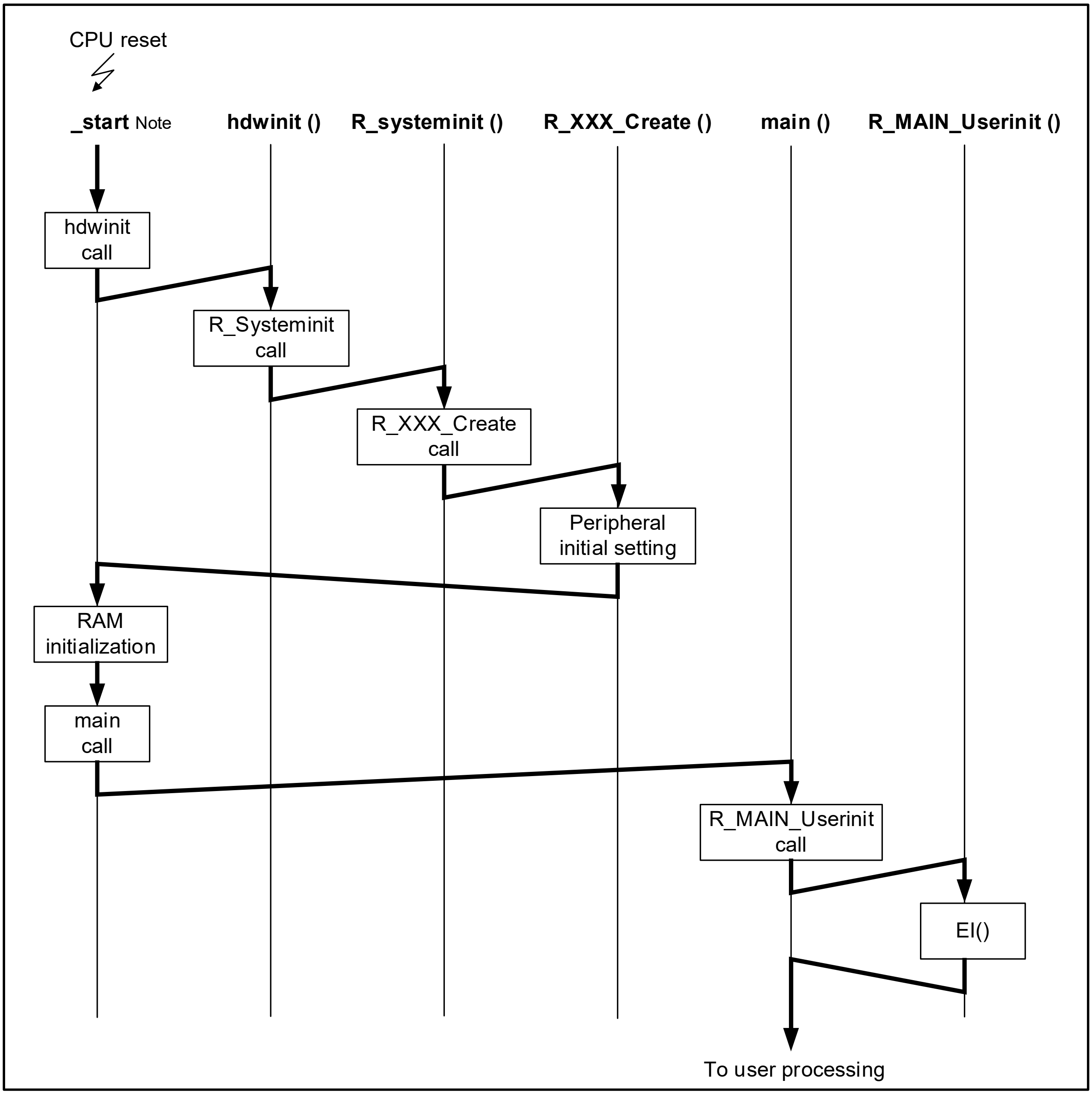
Note 1. xxx and XXX are replaced by abbreviations of peripheral feature names.

Note 2. For some peripheral features, CG might generate functions that are not listed in the “Function Name” column.

Note 3. For some peripheral features, the names of generated functions might not end with Start or Stop.

Caution: Some sample code might not use the CG generated files and functions listed in Table 1-3.

Figure 1-2 Processing image from CPU reset generation to main function



Note. Startup processing in cstart.asm

cstrat.asm is generated automatically when a new C project is created on CS+ for CC or e^2 studio.

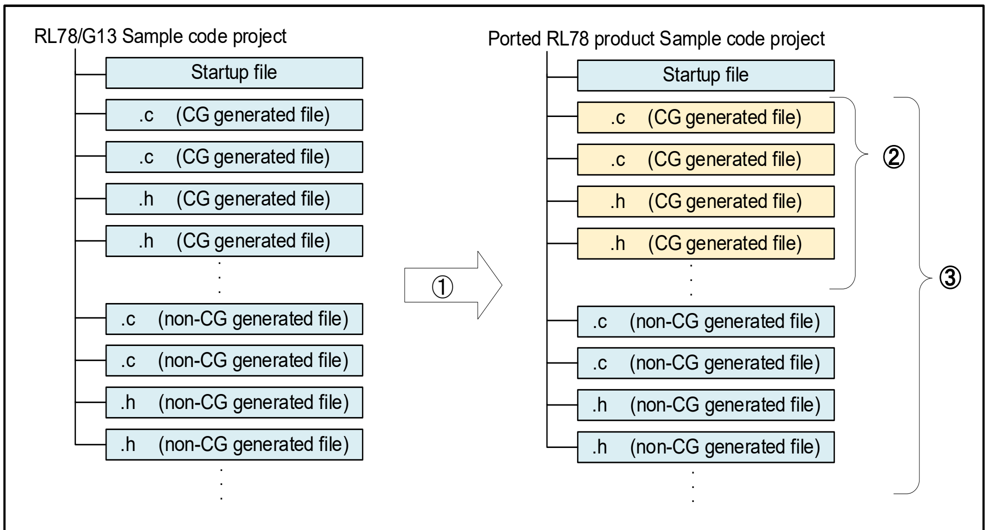
1.3 Porting Method

When replacing sample code, change the device in the project for the porting target RL78 product sample code and change resources to fit the porting target RL78 product.

In some sample code, part of CG generated code might be commented out or otherwise changed. If the porting source project contains such changes, you need to make the same changes in the porting target project before you rebuild it. To check whether changes are made in the source project, create a new project for the porting source RL78 product and check CG generated code for any changes. Note that the porting target RL78 product might not be able to use the same resources as the porting source product. If that is the case, you need to modify the code that is affected by resource changes.

Figure 1-3 shows the concept of porting sample code and Figure 1-4 provides an overview of the porting procedure.

Figure 1-3 Porting image

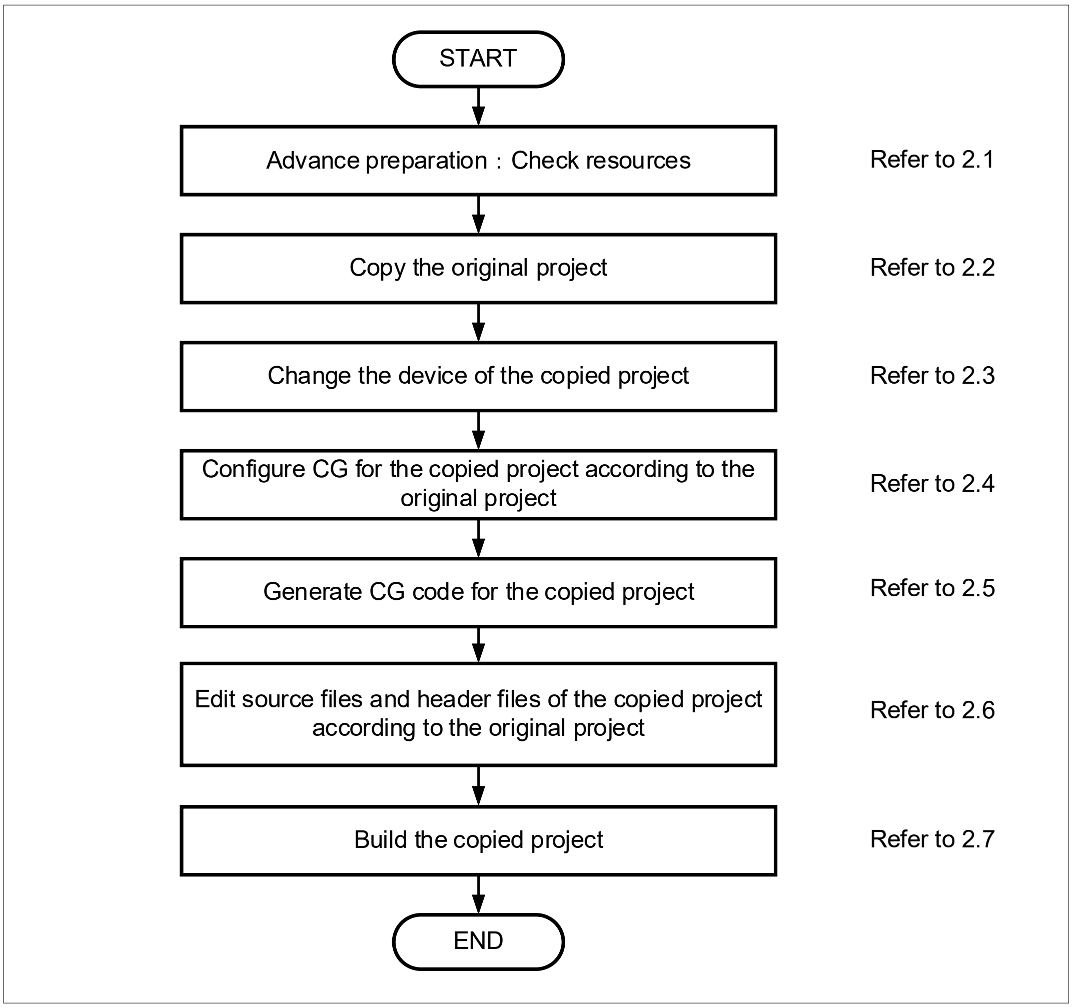

RL78/G13 Sample code project
Startup file
.c (CG generated file)
.c (CG generated file)
.h (CG generated file)
.h (CG generated file)
.c (non-CG generated file)
.c (non-CG generated file)
.h (non-CG generated file)
.h (non-CG generated file)
①
Ported RL78 product Sample code project
Startup file
.c (CG generated file)
.c (CG generated file)
.h (CG generated file)
.h (CG generated file)
.c (non-CG generated file)
.c (non-CG generated file)
.h (non-CG generated file)
.h (non-CG generated file)
②
③

① Change the target device to the porting RL78 product.

② Check whether original CG generated codes are changed.

③ If any code is affected by resource changes, change the code.

Figure 1-4 Overview of Porting Flow


START
Advance preparation : Check resources
Refer to 2.1
Copy the original project
Refer to 2.2
Change the device of the copied project
Refer to 2.3
Configure CG for the copied project according to the original project
Refer to 2.4
Generate CG code for the copied project
Refer to 2.5
Edit source files and header files of the copied project according to the original project
Refer to 2.6
Build the copied project
Refer to 2.7
END

Remark. The copied project = The porting project

2. Porting Procedure

This section describes each step in the porting flow shown in Figure 1-4.

2.1 Advance Preparation

Use the documents provided with sample code to check the port pins and peripheral functions used in the porting source sample code and porting target sample code. Then, read the user's manual for the porting target RL78 product to determine the porting methods for the pins and peripheral functions.

Table 2-1 lists the items to be checked for peripheral functions and Table 2-2 lists the items to be checked for used pins.

Table 2-1 Items to be Checked for Peripheral Functions

Status of Porting Target RL78 Product	**Porting Method**
If the target product has the same channels as the source product:	Use the same channels.
If the target product does not have the same channels as the source product:	Use different channels.

Table 2-2 Items to be Checked for Used Pins

Status of Porting Target RL78 Product	**Porting Method**
If the target product has the same port pins with the same multiplexed functions as the source product:	Use the same pins.
If the target product has the same port pins as the source product but the multiplexed functions of the pins are different:	When using the port function without using the multiplexed functions: Use the same port pins. When using the multiplexed functions: Use different pins with the same multiplexed functions.
If the target product does not have the same port pins as the source product:	When using the port function without using the multiplexed functions: Use different port pins. When using the multiplexed functions: Use different pins with the same multiplexed functions.

2.2 Copying Original Project

Copy the original sample code project to any folder.

In the following descriptions, the original sample code project is 'original project' and the copied project is 'porting project'.

2.3 Device Change for Porting Project

Change the target device of the porting project.

Refer to 2.3.1 when using the porting project for CS+ for CC, refer to 2.3.2 when using the porting project for e^{2} studio.

2.3.1 Porting Project for CS+ for CC

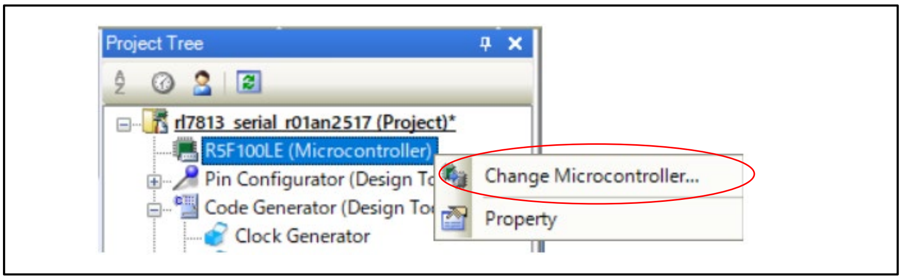
[Steps]

(1) Start CS+ for CC. Open the mtpj file for the porting project.

Remark. If the tool version when the sample code was created is different from the currently installed version, the warning message is displayed informing you that the tool version is different.

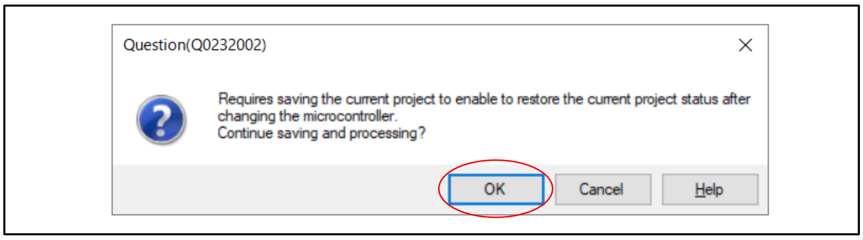
(2) Right click the microcontroller name in the project tree of CS+ for CC and click [Change Microcontroller].

Figure 2-1 [Change Microcontroller] Menu


Project Tree
rl7813_serial_r01an2517 (Project)*
R5F100LE (Microcontroller)
Pin Configurator (Design To
Code Generator (Design To
Clock Generator
Change Microcontroller...
Property

(3) Click [OK] on the [Question] dialog box.

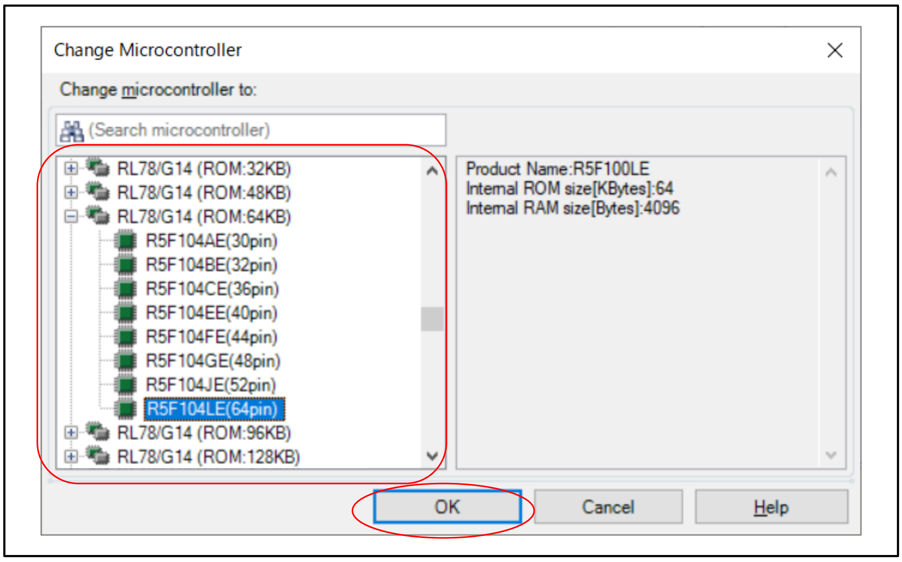
Figure 2-2 [Question] Dialog Box


Question(Q0232002)

Requires saving the current project to enable to restore the current project status after changing the microcontroller.
Continue saving and processing?

(4) Select the porting device in [Change Microcontroller] and click [OK].

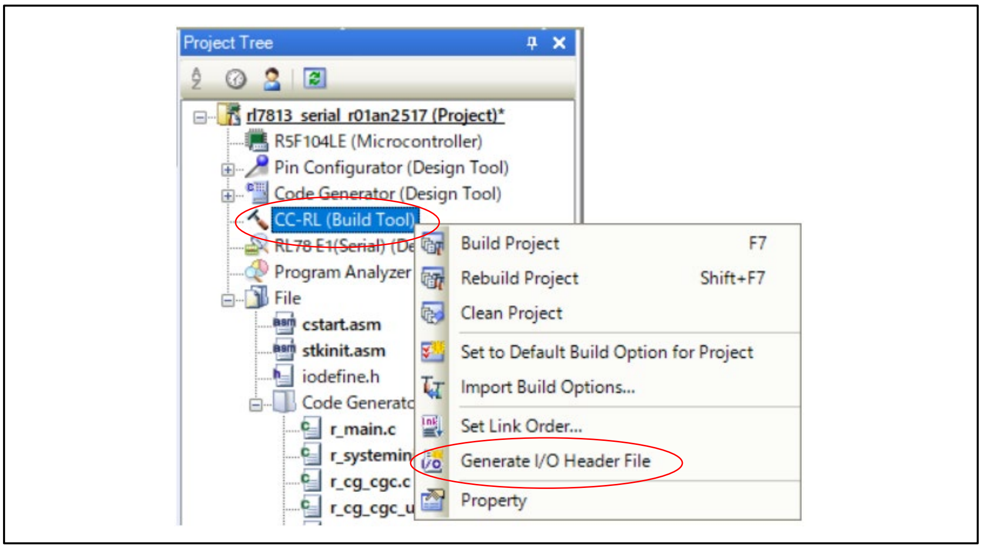
Figure 2-3 [Change Microcontroller]



Product Name:R5F100LE
Internal ROM size[KBytes]:64
Internal RAM size[Bytes]:4096

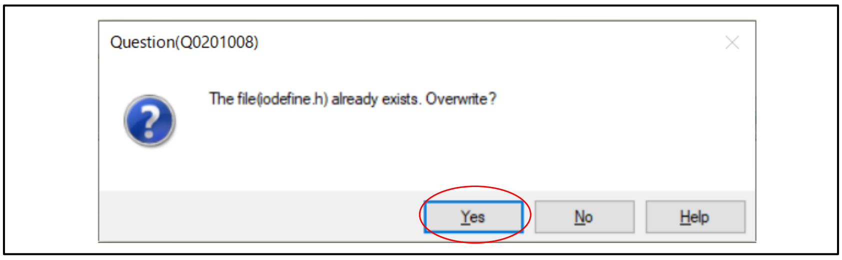
(5) Generate the I/O header file for the changed device. Right click [CC-RL (Build Tool)] and click [Generate I/O Header File].

Figure 2-4 [Generate I/O Header File] Menu



(6) Click [Yes] on the [Question] dialog box.

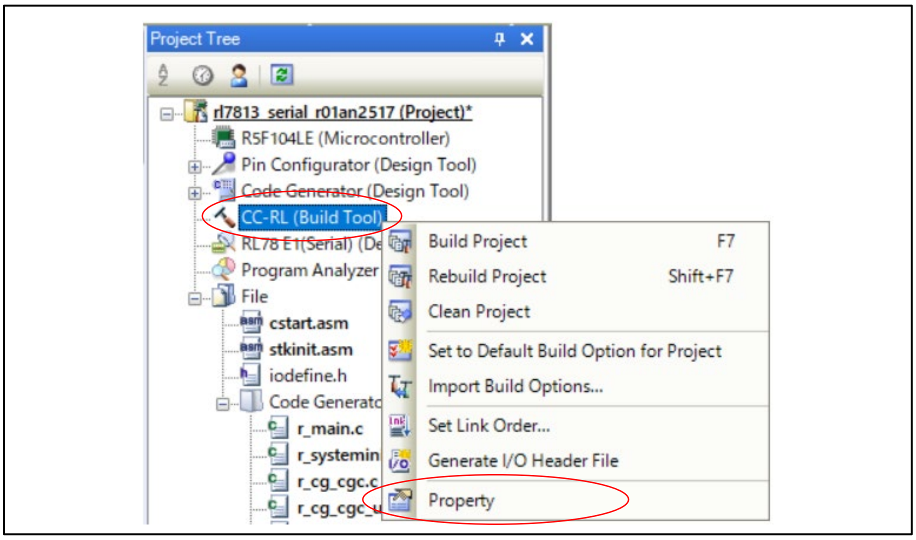
Figure 2-5 [Question] Dialog Box


Question(Q0201008)

The file(iodefine.h) already exists. Overwrite?

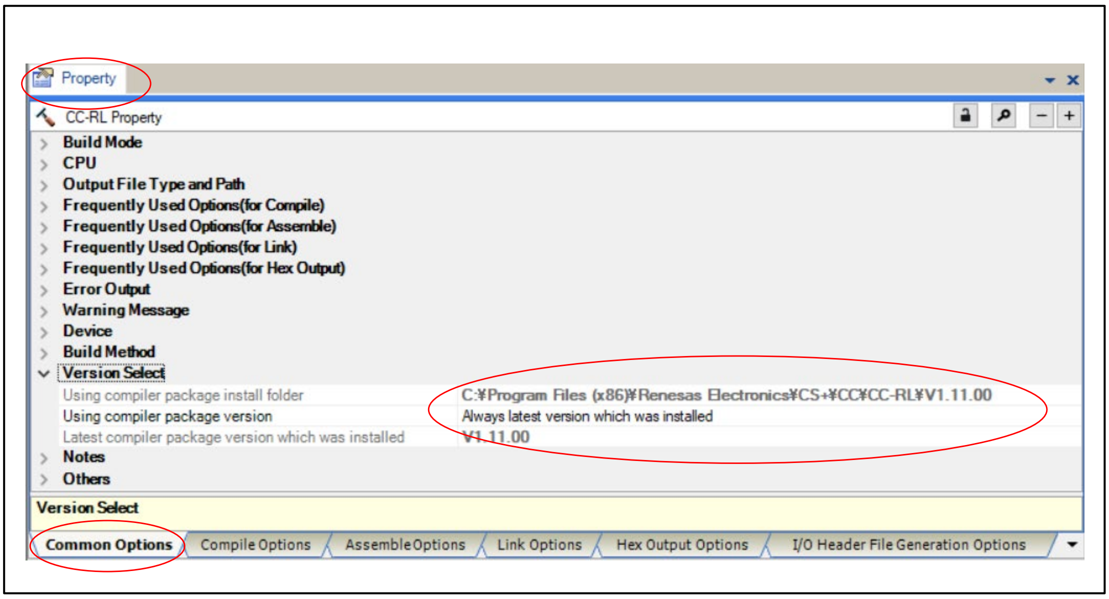
(7) Specify the version of the CC-RL Compiler. Right click [CC-RL (Build Tool)] and click [Property].

Figure 2-6 CC-RL [Property] Menu



(8) Specify the version of the installed CC-RL Compiler currently in use.

Figure 2-7 CC-RL [Property] – [Version Select]



- Build Mode
- CPU
- Output File Type and Path
- Frequently Used Options(for Compile)
- Frequently Used Options(for Assemble)
- Frequently Used Options(for Link)
- Frequently Used Options(for Hex Output)
- Error Output
- Warning Message
- Device
- Build Method
- Version Select

Using compiler package install folder	C:¥Program Files (x86)¥Renesas Electronics¥CS+¥CC¥CC-RL¥V1.11.00
Using compiler package version	Always latest version which was installed
Latest compiler package version which was installed	V1.11.00

- Notes
- Others

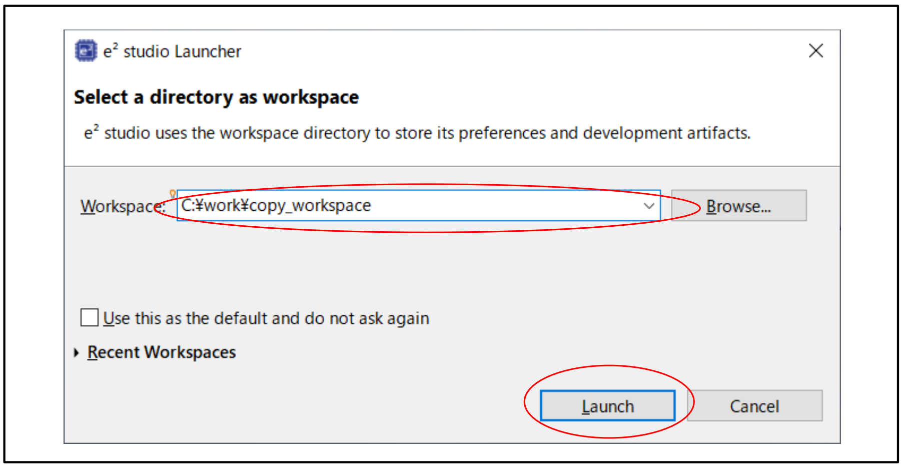
Version Select

2.3.2 Porting project for e^2 studio

[Steps]

(1) Start e^2 studio. Specify any folder at [Workspace:] on [e^2 studio Launcher] and click [Launch].

Figure 2-8 [e^2 studio Launcher]



e^2 studio Launcher

Select a directory as workspace

e^2 studio uses the workspace directory to store its preferences and development artifacts.

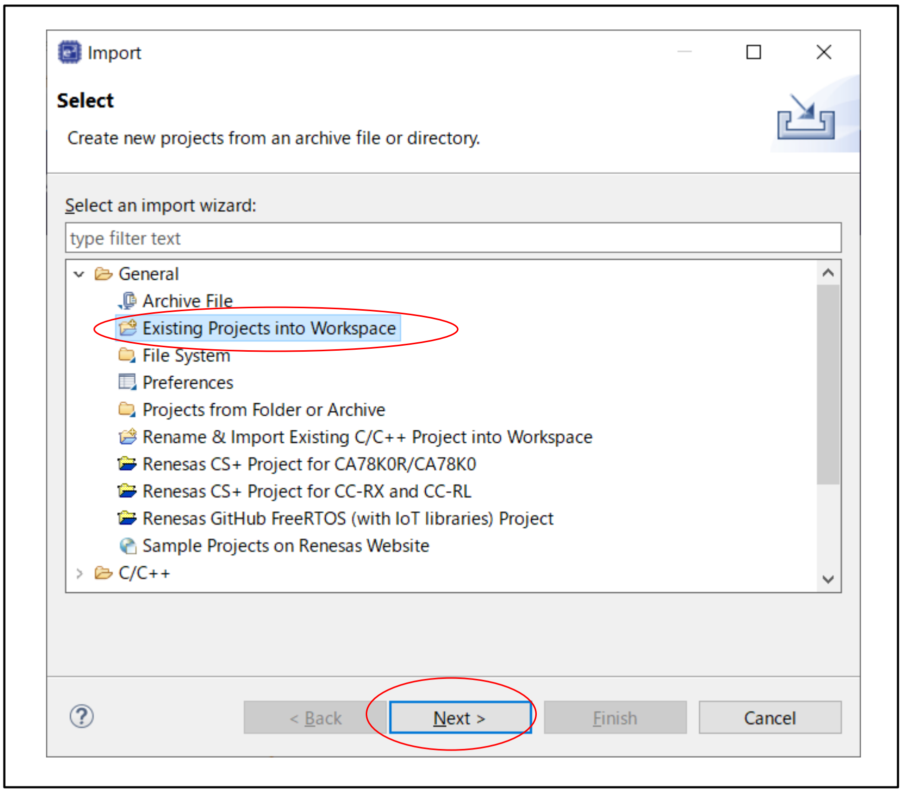
Workspace: C:¥work¥copy_workspace

□ Use this as the default and do not ask again

(2) Select the [File] – [Import] menu of e^2 studio.

(3) Select [General] - [Existing Projects into Workspace] and click [Next] on [Import].

Figure 2-9 [Import] (1/2)



Import

Select

Create new projects from an archive file or directory.

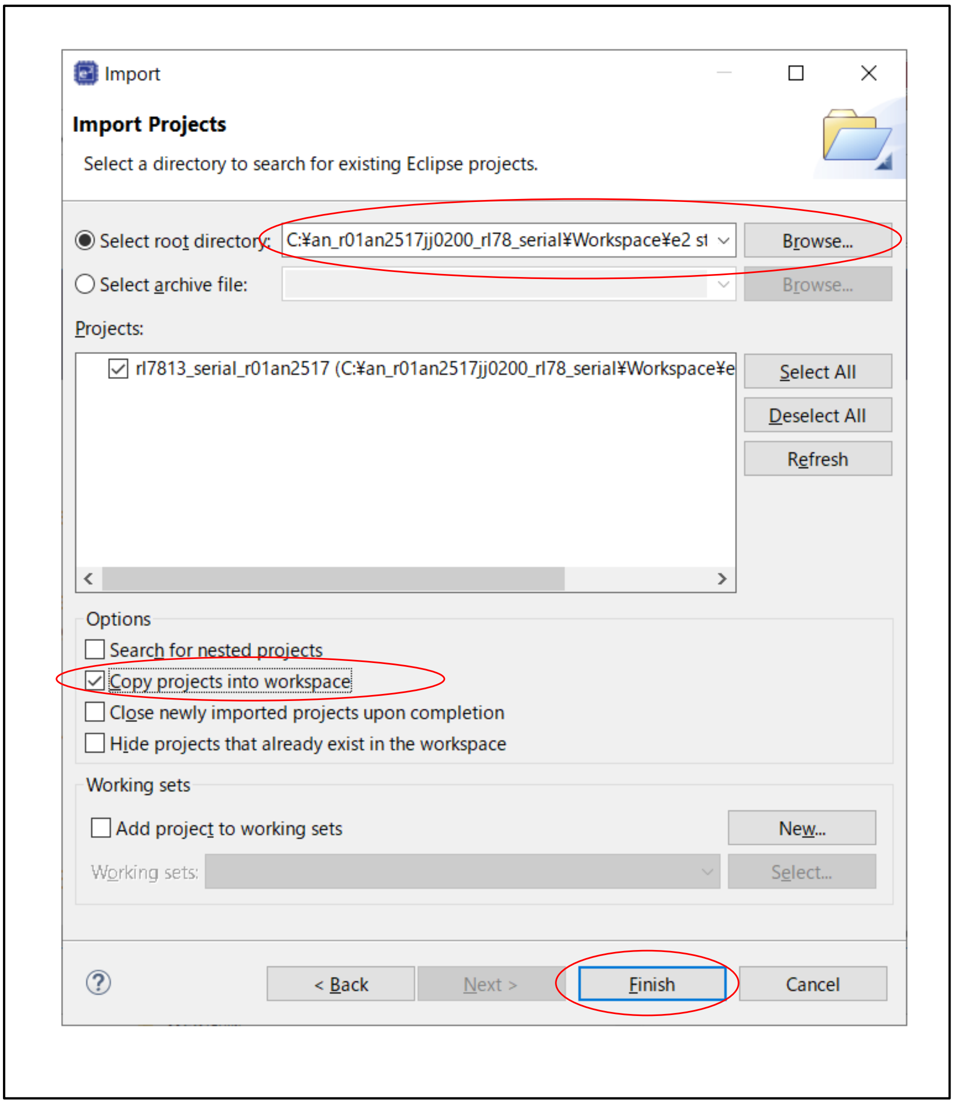
Select an import wizard:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
  - Projects from Folder or Archive
  - Rename & Import Existing C/C++ Project into Workspace
  - Renesas CS+ Project for CA78K0R/CA78K0
  - Renesas CS+ Project for CC-RX and CC-RL
  - Renesas GitHub FreeRTOS (with IoT libraries) Project
  - Sample Projects on Renesas Website
- C/C++

(4) Specify the folder where the porting project exists at [Select root directory:] on [Import]. Check [Copy projects into workspace] and click [Finish].

Figure 2-10 [Import] (2/2)



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:¥an_r01an2517jj0200_rl78_serial¥Workspace¥e2 st

○ Select archive file:

Projects:

☑ rl7813_serial_r01an2517 (C:¥an_r01an2517jj0200_rl78_serial¥Workspace¥e

Options

☐ Search for nested projects

☑ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

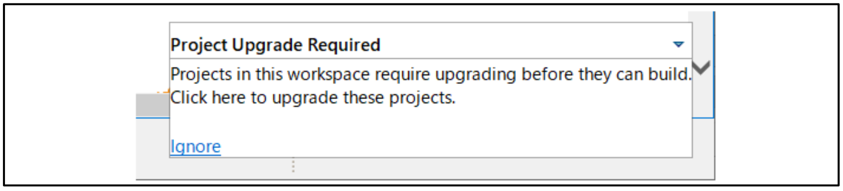
Working sets

☐ Add project to working sets

Working sets:

(5) After importing the porting project, if the message shown in Figure 2-11 is displayed at the bottom right of the e2 studio screen, or if the error shown in Figure 2-12 is displayed at the [Problems] view of e2 studio, go to step (6). If the message or the [Problems] view is not displayed, go to step (7).

Figure 2-11 Message requesting the project upgrade

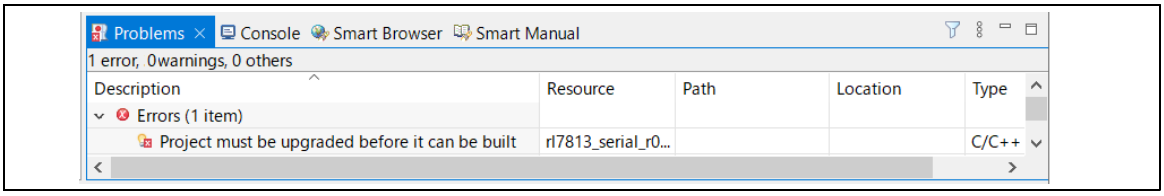


Project Upgrade Required

Projects in this workspace require upgrading before they can build. Click here to upgrade these projects.

Ignore

Figure 2-12 [Problems] View

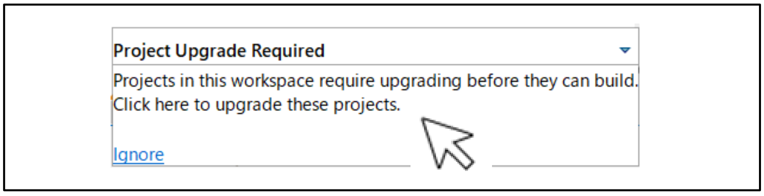


1 error, 0warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
Project must be upgraded before it can be built	rl7813_serial_r0...			C/C++

(6) If the e2 studio version when the sample code was created is different from the currently installed version, the porting project needs to be upgraded to match the currently installed version. Click the message requesting the project upgrade. (Figure 2-13)
[Upgrade Legacy e2 studio Projects] appears next. Check the box for the porting project and click [Finish]. (Figure 2-14)

Figure 2-13 Clicking the message requesting the project upgrade

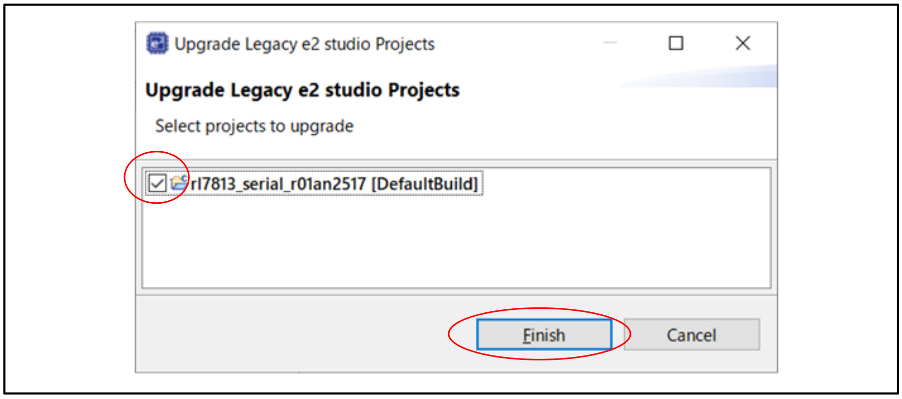


Project Upgrade Required

Projects in this workspace require upgrading before they can build. Click here to upgrade these projects.

Ignore

Figure 2-14 [Upgrade Legacy e2 studio Projects]



Upgrade Legacy e2 studio Projects

Upgrade Legacy e2 studio Projects

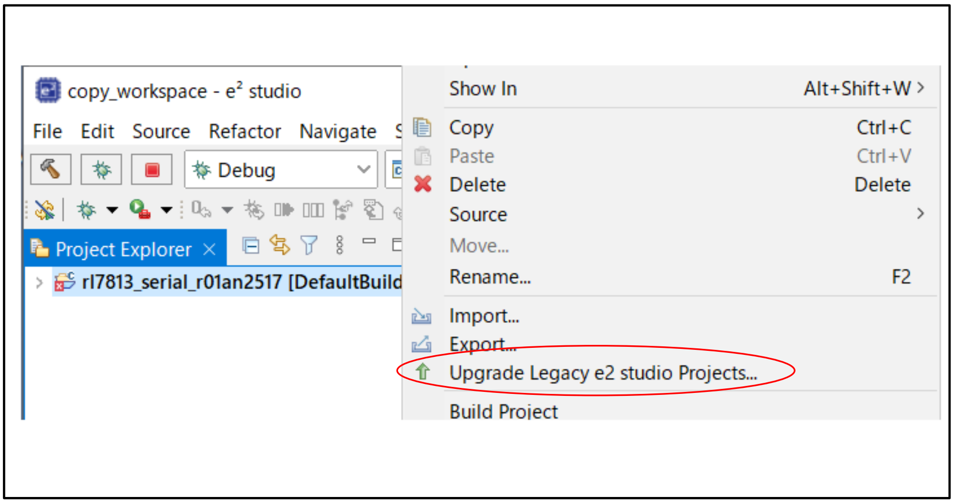
Select projects to upgrade

☑ rl7813_serial_r01an2517 [DefaultBuild]

Remark. If the message requesting the project upgrade is closed before upgrading the porting project, upgrade the porting project by the following procedure.

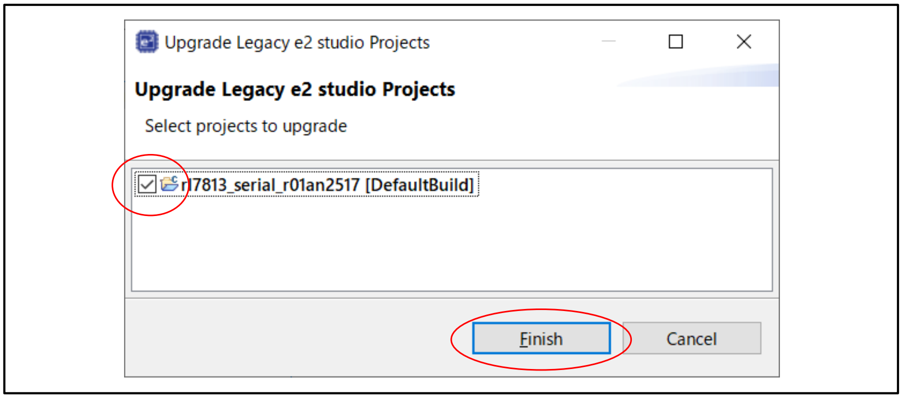
Right click the porting project in the Project Explorer of e2 studio and click [Upgrade Legacy e2 studio Projects]. (Figure 2-15)

Figure 2-15 [Upgrade Legacy e2 studio Projects] Menu



[Upgrade Legacy e2 studio Projects] appears next. Check the box for the porting project and click [Finish].

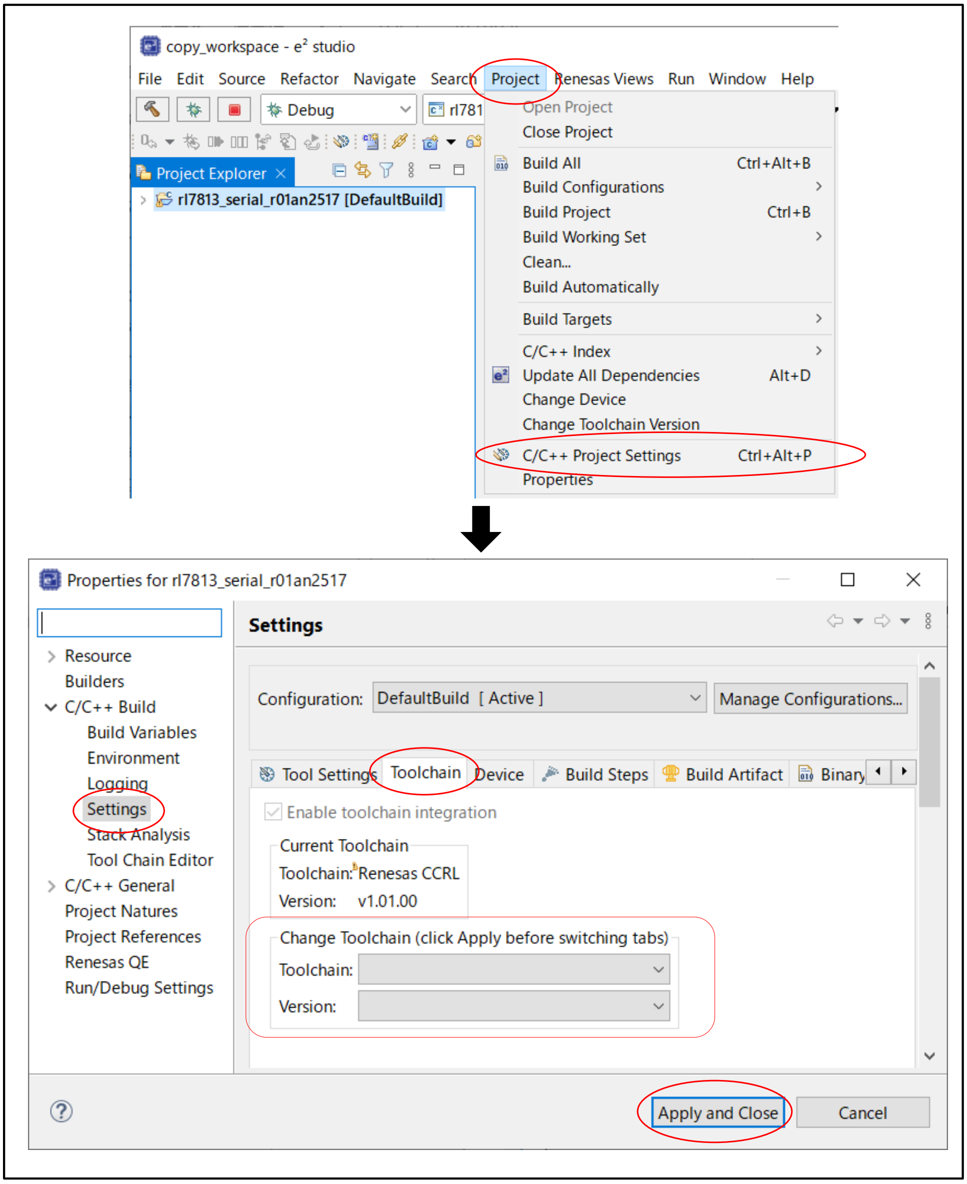
Figure 2-16 [Upgrade Legacy e2 studio Projects]



Upgrade Legacy e2 studio Projects
Upgrade Legacy e2 studio Projects
Select projects to upgrade
☑ rl7813_serial_r01an2517 [DefaultBuild]
Finish
Cancel

(7) After importing the porting project, if the CC-RL Compiler version when the sample code was created is different from the currently installed version, the toolchain setting needs to be set to match the currently installed version. Click the porting project in the Project Explorer of e2 studio and click the [Project] – [C/C++ Project Settings] menu. In the [Properties] dialog box, select [C/C++ Build] - [Settings]. Specify the version currently installed and click [Apply and Close].

Figure 2-17 Toolchain setting in [Properties]



copy_workspace - e² studio

rl7813_serial_r01an2517 [DefaultBuild]

Properties for rl7813_serial_r01an2517

Settings

Configuration: DefaultBuild [Active]

☑ Enable toolchain integration

Current Toolchain

Toolchain: Renesas CCRL

Version: v1.01.00

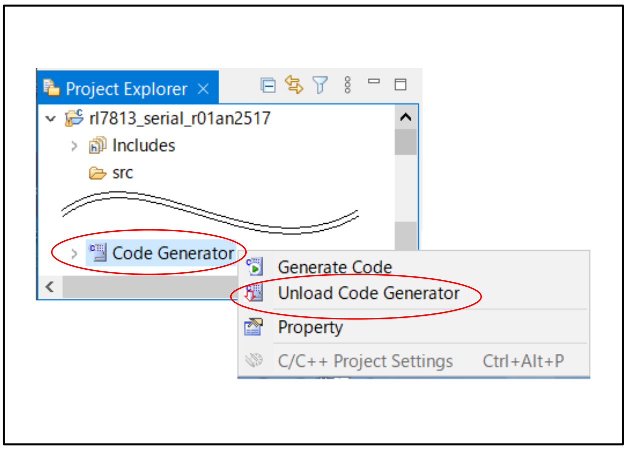
Change Toolchain (click Apply before switching tabs)

Toolchain:

Version:

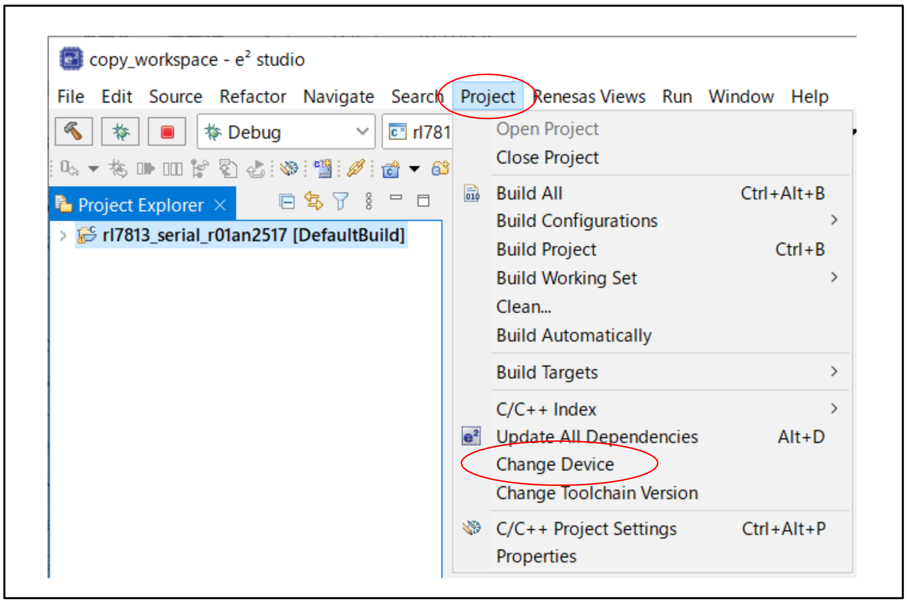
(8) Right click [Code Generator] in the Project Explorer of e^2 studio and click [Unload Code Generator].

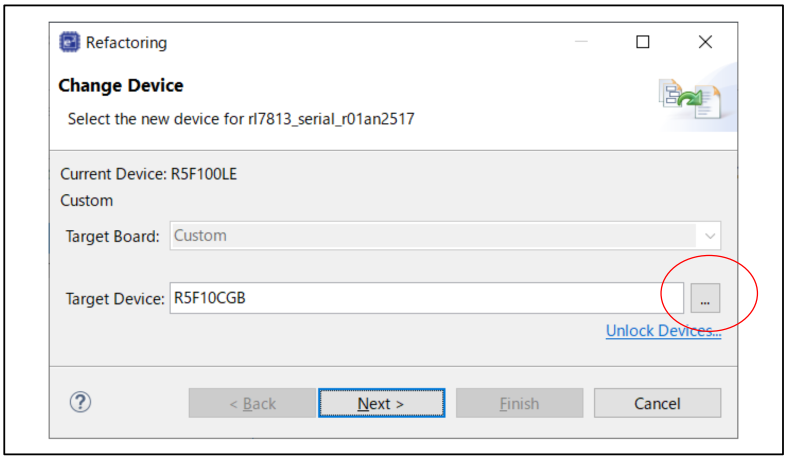
Figure 2-18 Deleting Code Generator



(9) Click the porting project in the Project Explorer of e2 studio and click the [Project] – [Change Device] menu. Select the device for the porting project in the [Device Selection] dialog box and click [OK]. Click [Next] / [Finish] / [OK] in the dialog boxes that are displayed one after another.

Figure 2-19 Change Device





Refactoring

Change Device

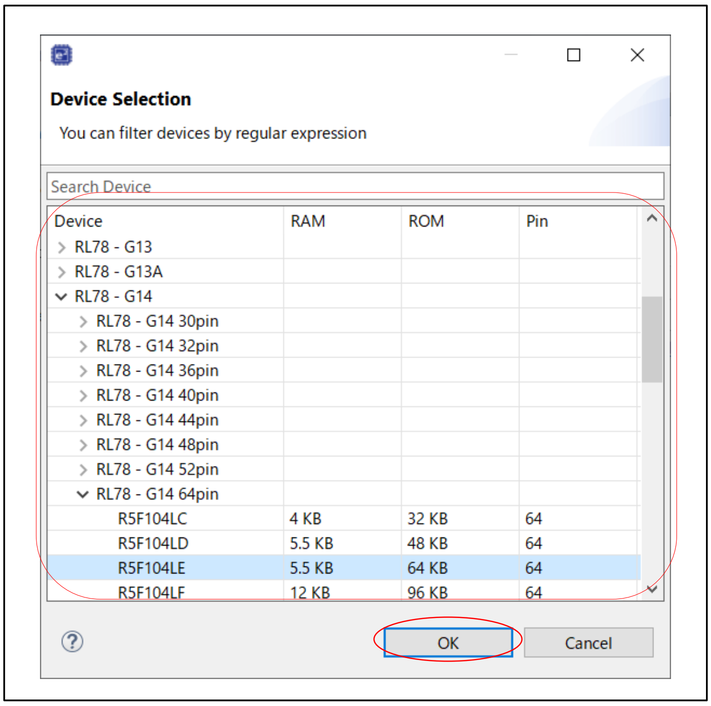
Select the new device for rl7813_serial_r01an2517

Current Device: R5F100LE

Custom

Target Board: Custom

Target Device: R5F10CGB

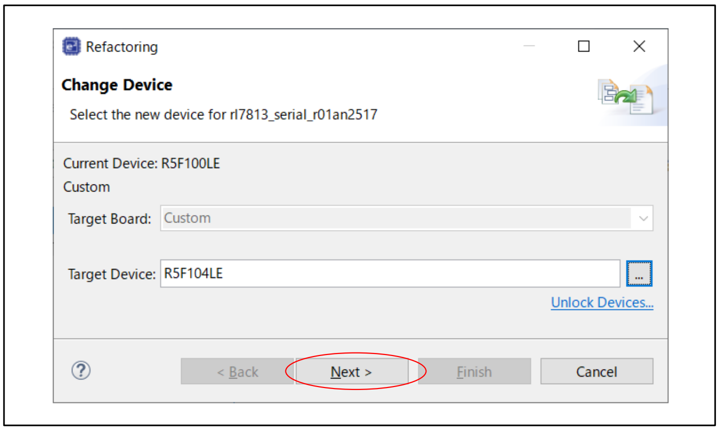


Device Selection

You can filter devices by regular expression

Search Device

Device	RAM	ROM	Pin
RL78 - G13			
RL78 - G13A			
RL78 - G14			
RL78 - G14 30pin			
RL78 - G14 32pin			
RL78 - G14 36pin			
RL78 - G14 40pin			
RL78 - G14 44pin			
RL78 - G14 48pin			
RL78 - G14 52pin			
RL78 - G14 64pin			
R5F104LC	4 KB	32 KB	64
R5F104LD	5.5 KB	48 KB	64
R5F104LE	5.5 KB	64 KB	64
R5F104LF	12 KB	96 KB	64



Refactoring

Change Device

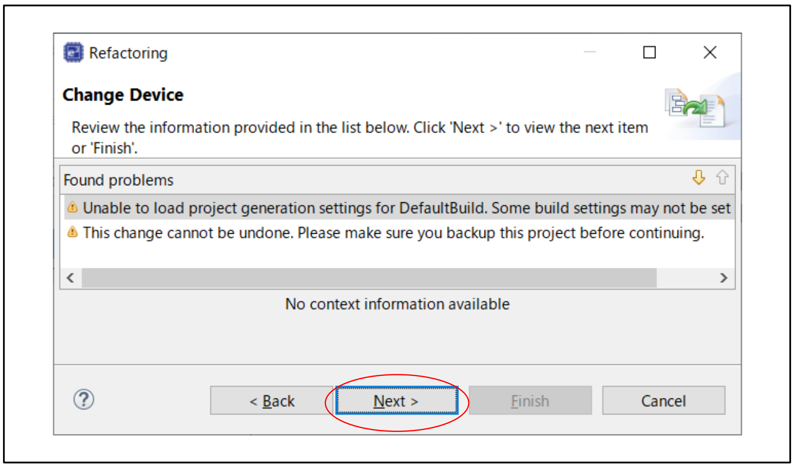
Select the new device for rl7813_serial_r01an2517

Current Device: R5F100LE

Custom

Target Board: Custom

Target Device: R5F104LE



Refactoring

Change Device

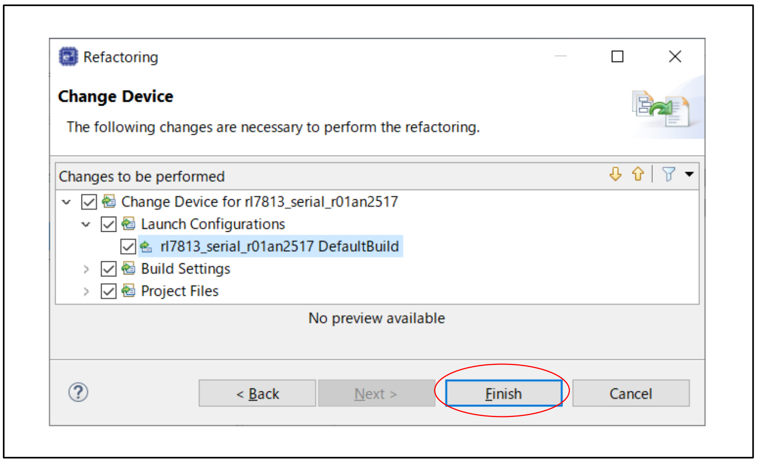
Review the information provided in the list below. Click 'Next >' to view the next item or 'Finish'.

Found problems

Unable to load project generation settings for DefaultBuild. Some build settings may not be set

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

No context information available



Refactoring

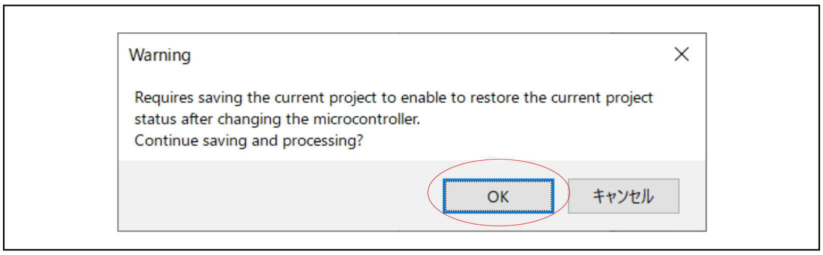
Change Device

The following changes are necessary to perform the refactoring.

Changes to be performed

- ☑ Change Device for rl7813_serial_r01an2517
  - ☑ Launch Configurations
    - ☑ rl7813_serial_r01an2517 DefaultBuild
  - ☑ Build Settings
  - ☑ Project Files

No preview available

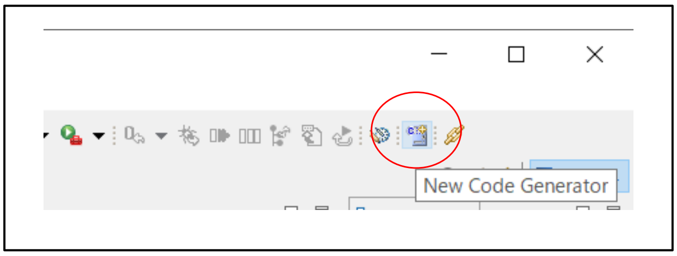


Warning

Requires saving the current project to enable to restore the current project status after changing the microcontroller.
Continue saving and processing?

(10) Add Code Generator for the changed device to the porting project. Click the [New Code Generator] button on the tool bar of e2 studio. The [Code Generator] node is added in the Project tree.

Figure 2-20 [New Code Generator] Button

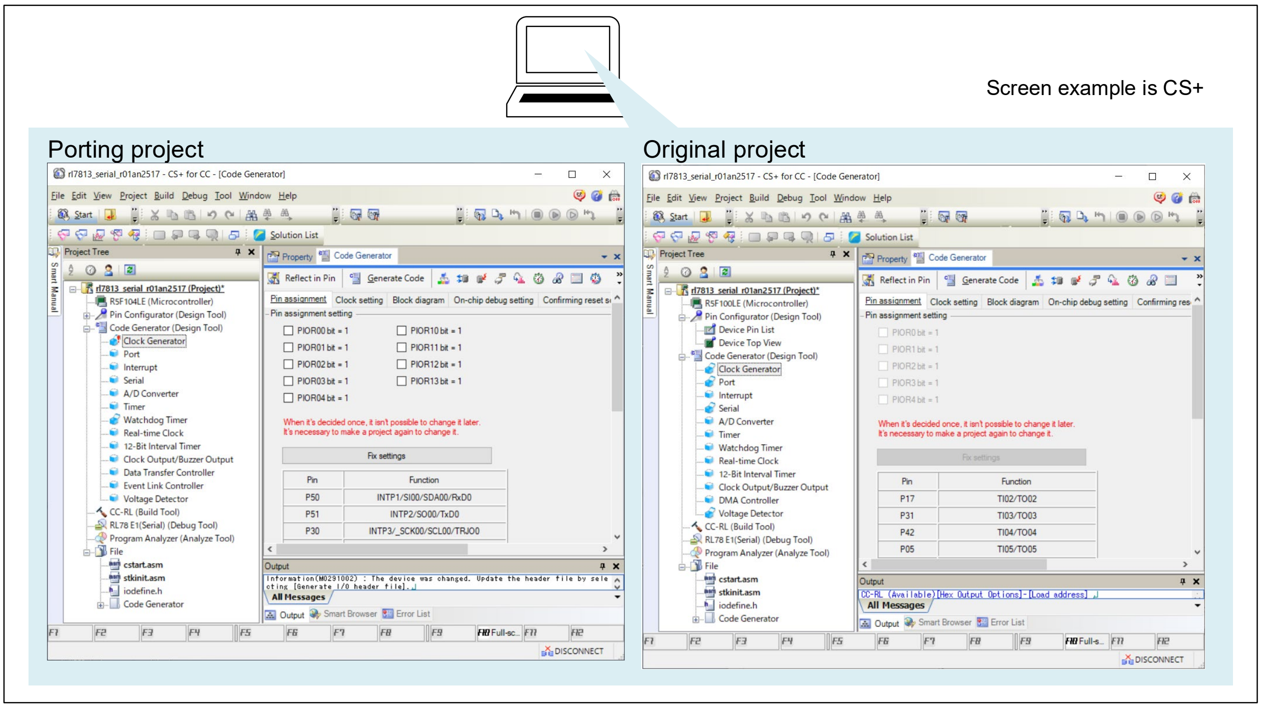


2.4 Code Generator Configuration for Porting Project

In order to replace the code related to the peripheral function settings with the code for the porting device, configure the peripheral functions in Code Generator of the porting project according to the Code Generator settings in the original project.

Start another CS+ for CC for the original project side by side and configure the peripheral functions for the porting project.

Figure 2-21 Operation Image of CG configuration



Screen example is CS+

Porting project

rl7813_serial_r01an2517 - CS+ for CC - [Code Generator]

Pin assignment setting

- □ PIOR00 bit = 1
- □ PIOR01 bit = 1
- □ PIOR02 bit = 1
- □ PIOR03 bit = 1
- □ PIOR04 bit = 1
- □ PIOR10 bit = 1
- □ PIOR11 bit = 1
- □ PIOR12 bit = 1
- □ PIOR13 bit = 1

When it's decided once, it isn't possible to change it later.
It's necessary to make a project again to change it.

Pin	Function
P50	INTP1/SI00/SDA00/RxD0
P51	INTP2/SO00/TxD0
P30	INTP3/_SCK00/SCL00/TRJO0

```
Information(M0291002) : The device was changed. Update the header file by sele
cting [Generate I/O header file].
```

Original project

rl7813_serial_r01an2517 - CS+ for CC - [Code Generator]

Pin assignment setting

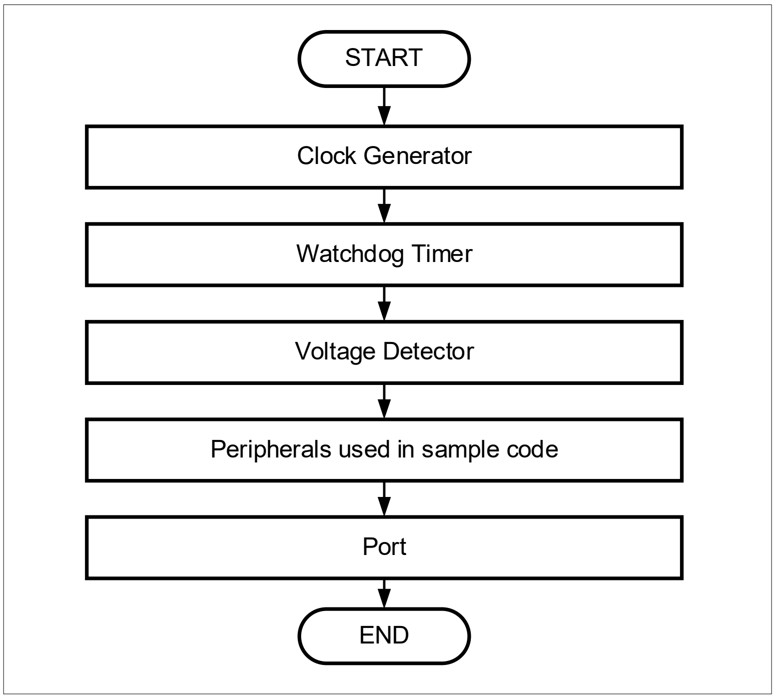
- □ PIOR0 bit = 1
- □ PIOR1 bit = 1
- □ PIOR2 bit = 1
- □ PIOR3 bit = 1
- □ PIOR4 bit = 1

When it's decided once, it isn't possible to change it later.
It's necessary to make a project again to change it.

Pin	Function
P17	TI02/TO02
P31	TI03/TO03
P42	TI04/TO04
P05	TI05/TO05

```
CC-RL (Available)[Hex Output Options]-[Load address]
```

Figure 2-22 Flow of CG Configuration



The following procedure is described using CS+ for CC. The procedure for e2 studio is the same as for CS+ for CC.
There may be differences in the presence or absence of settings and selections between the original project and the porting project. If there are differences, configure Code Generator for the porting project as follows.

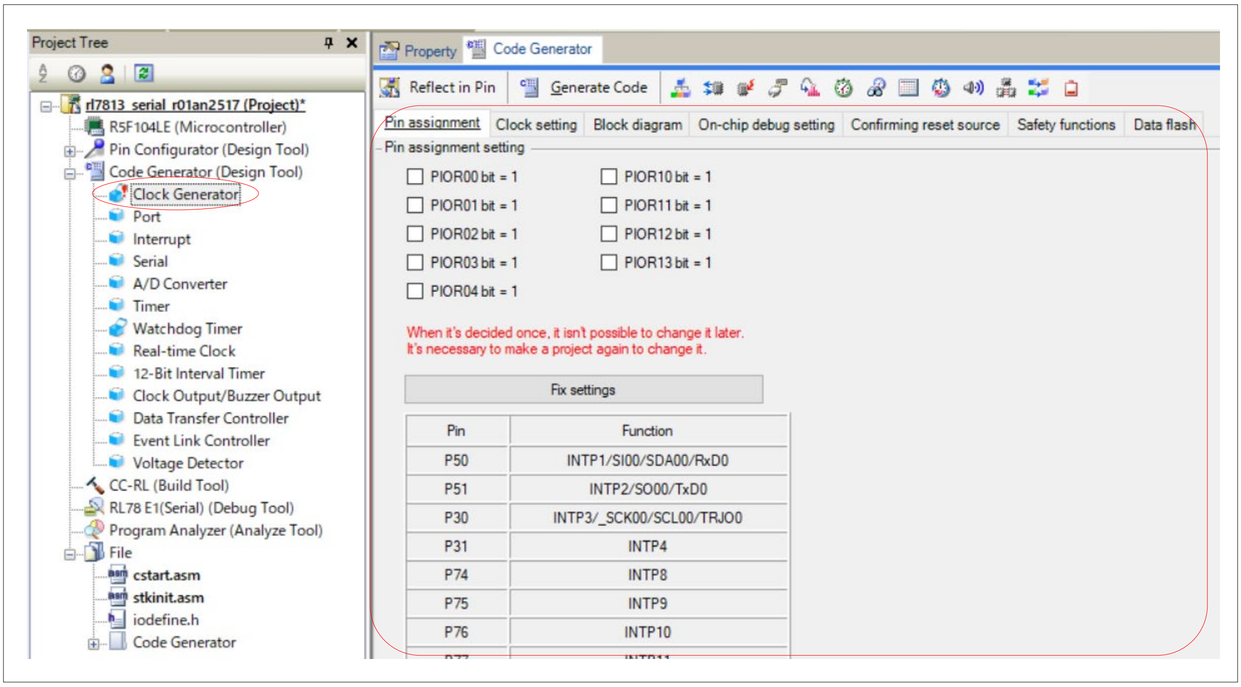
- If the setting item is in the original project but there is no setting item in the porting project.
  → Set nothing in the porting project.
- If there is no setting item in the original project but there is the setting item in the porting project.
  → In the porting project, use the default setting or set it according to your environment.
- If there is the same setting item in both the original project and the porting project but the selections of the item are different.
  → In the porting project, use the default setting or set it according to your environment.

[Steps]

(1) Start another CS+ for CC and open the original project. Two CS+ for CCs are running, one is for the porting project, and another is for the original project.

(2) Configure [Clock Generator]. Double click [Clock Generator] under the [Code Generator (Design Tool)] node of both projects. Set each item for the porting project that corresponds to the setting for the original project. Configure all tabs of [Pin assignment], [Clock setting], [On-chip debug setting], [Confirming reset source], [Safety functions] and [Data flash].

Figure 2-23 [Clock Generator]



Pin assignment setting

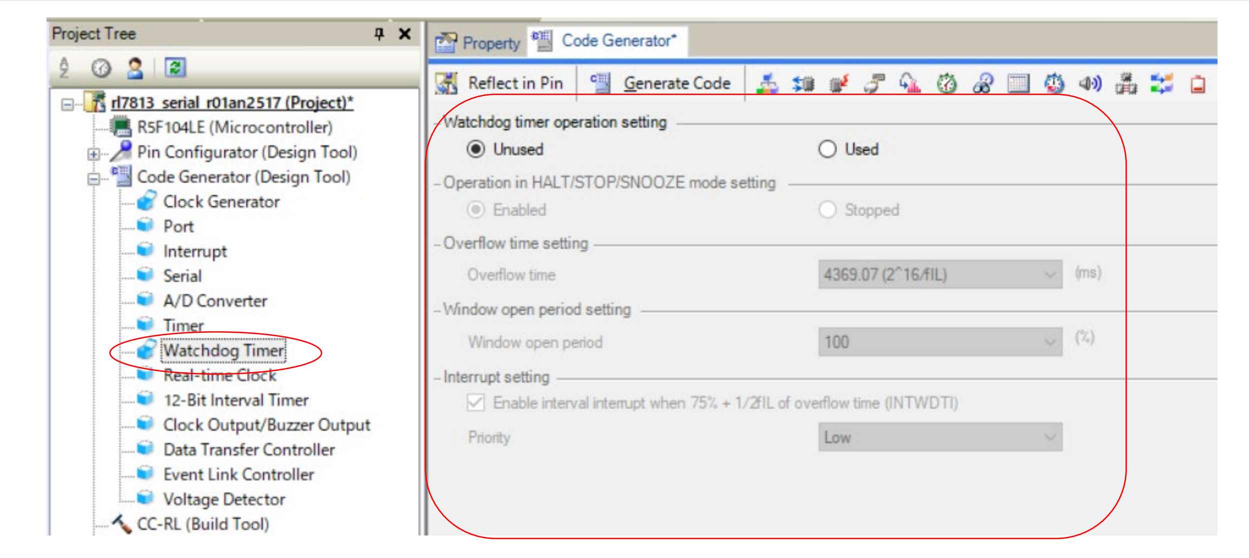
□ PIOR00 bit = 1 □ PIOR10 bit = 1
□ PIOR01 bit = 1 □ PIOR11 bit = 1
□ PIOR02 bit = 1 □ PIOR12 bit = 1
□ PIOR03 bit = 1 □ PIOR13 bit = 1
□ PIOR04 bit = 1

When it's decided once, it isn't possible to change it later.
It's necessary to make a project again to change it.

Pin	Function
P50	INTP1/SI00/SDA00/RxD0
P51	INTP2/SO00/TxD0
P30	INTP3/_SCK00/SCL00/TRJO0
P31	INTP4
P74	INTP8
P75	INTP9
P76	INTP10

(3) Configure [Watchdog Timer]. Double click [Watchdog Timer] under the [Code Generator (Design Tool)] node of both projects. Set each item for the porting project that corresponds to the setting for the original project.

Figure 2-24 [Watchdog Timer]



Watchdog timer operation setting
● Unused ○ Used

Operation in HALT/STOP/SNOOZE mode setting
● Enabled ○ Stopped

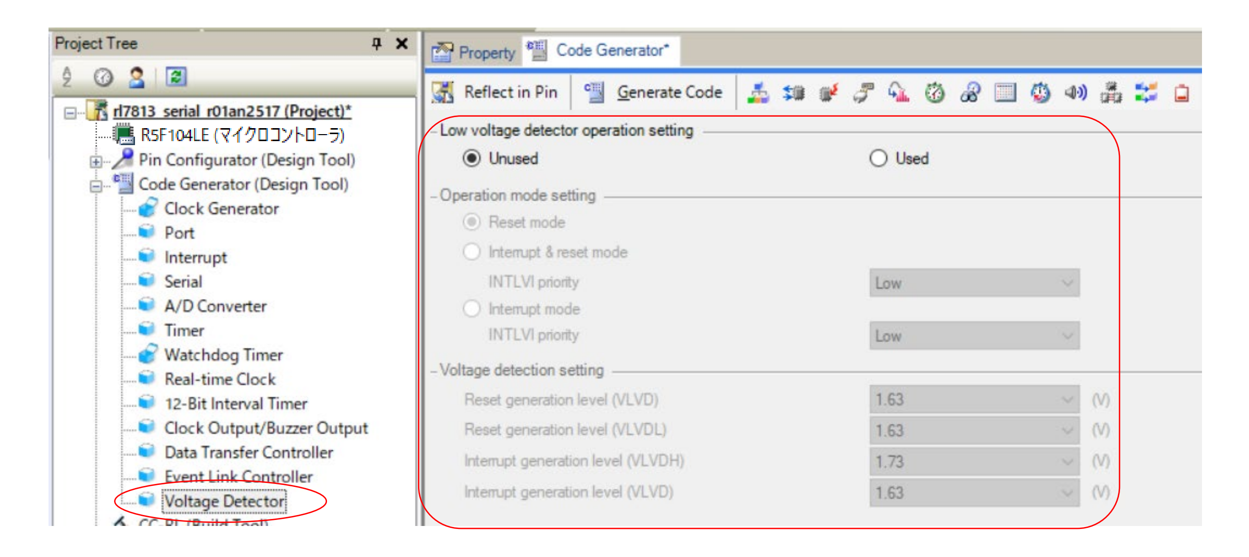
Overflow time setting
Overflow time 4369.07 (2^16/fIL) (ms)

Window open period setting
Window open period 100 (%)

Interrupt setting
☑ Enable interval interrupt when 75% + 1/2fIL of overflow time (INTWDTI)
Priority Low

(4) Configure [Voltage Detector]. Double click [Voltage Detector] under the [Code Generator (Design Tool)] node of both projects. Set each item for the porting project that corresponds to the setting for the original project.

Figure 2-25 [Voltage Detector]



Low voltage detector operation setting
● Unused ○ Used

Operation mode setting
● Reset mode
○ Interrupt & reset mode
INTLVI priority Low
○ Interrupt mode
INTLVI priority Low

Voltage detection setting
Reset generation level (VLVD) 1.63 (V)
Reset generation level (VLVDL) 1.63 (V)
Interrupt generation level (VLVDH) 1.73 (V)
Interrupt generation level (VLVD) 1.63 (V)

(5) Configure the same peripheral functions used in the original project for the porting project. The icon shows whether the peripheral function has any setting. Check icons under the [Code Generator (Design Tool)] node of the original project and configure the peripheral functions for the porting project to match the configuration for the original project.

.

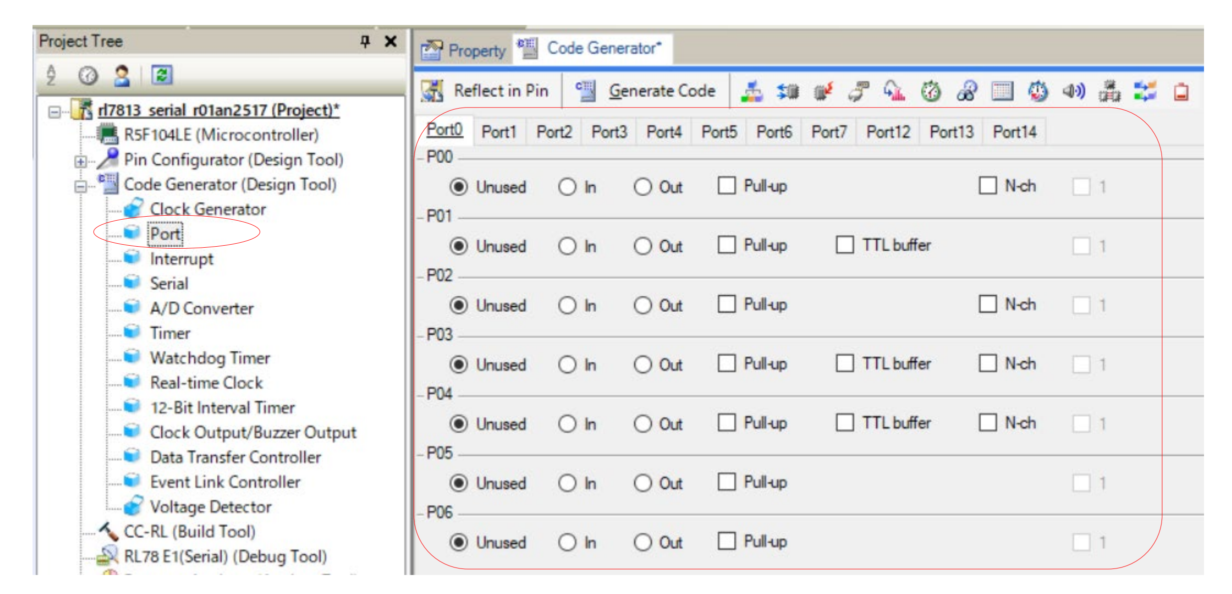
: The peripheral function has any setting. (Used)

: The peripheral function has no setting. (Unused)

(6) Configure [Port]. Double click [Port] under the [Code Generator (Design Tool)] node of both projects. Set each item for the porting project that corresponds to the setting for the original project. Refer to Table 2-2 about I/O port setting in [Port].

Caution. Provide proper treatment for unused pins so that their electrical specifications are observed. Connect each of any unused input-only ports to VDD or VSS via a separate resistor.

Figure 2-26 [Port]



P00: ● Unused ○ In ○ Out □ Pull-up □ N-ch □ 1

P01: ● Unused ○ In ○ Out □ Pull-up □ TTL buffer □ 1

P02: ● Unused ○ In ○ Out □ Pull-up □ N-ch □ 1

P03: ● Unused ○ In ○ Out □ Pull-up □ TTL buffer □ N-ch □ 1

P04: ● Unused ○ In ○ Out □ Pull-up □ TTL buffer □ N-ch □ 1

P05: ● Unused ○ In ○ Out □ Pull-up □ 1

P06: ● Unused ○ In ○ Out □ Pull-up □ 1

(7) Save the configuration setting for the porting project.

CS+ for CC: Select the [File] – [Save Project] menu

e^2 studio: Select the [File] – [Save All] menu

2.5 Generating code for Porting Project

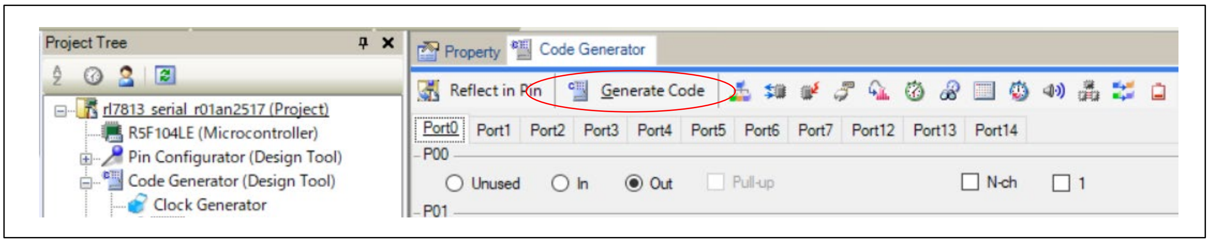
Generate the code for the porting project according to the setting of Code Generator (CG).

2.5.1 Porting project for CS+ for CC

[Steps]

(1) Click the [Generate Code] button in the porting project. The codes in CG generated files with the same name are overwritten with the code based on the changed device's peripheral functions.

Figure 2-27 [Generate Code] button



2.5.2 Porting Project for e^2 studio

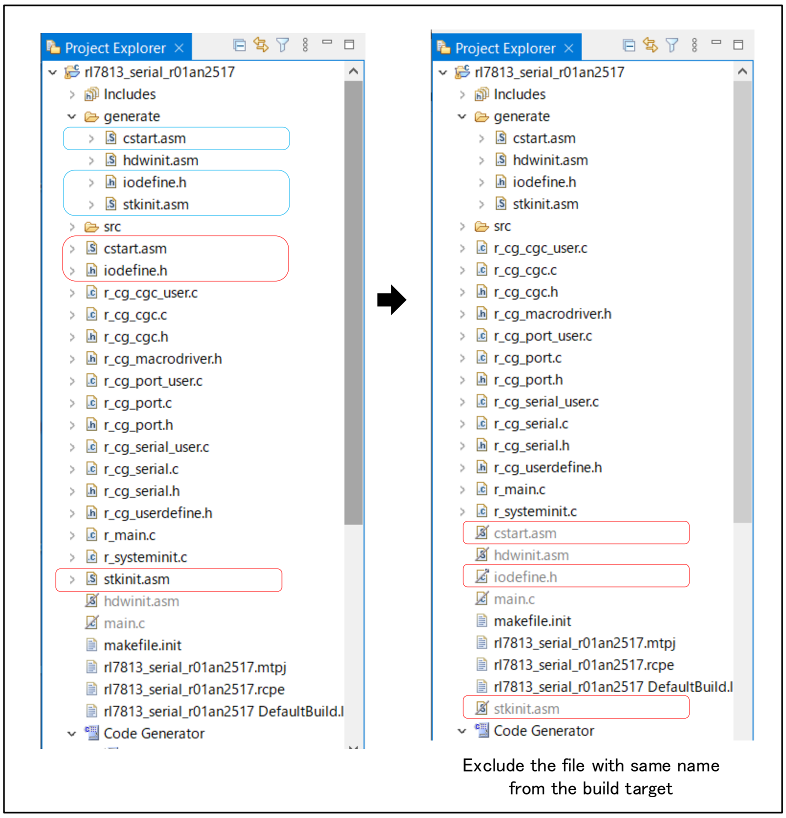
In e^2 studio, the startup file, etc. based on the changed device have been generated in the “generate” folder when the target device for the porting project was changed in 2.3.2. Furthermore, Code Generator (CG) generates files in the “src” folder.
Therefore, the following steps are required before generating code.

- Of the build target files inherited from the original project, the files with the same name as the files in the “generate” folder are excluded from the build target.
- Move the files generated by CG to the “src” folder.

[Steps]

(1) Of the build target files already registered in the porting project, exclude the files with the same name as the files in the “generate” folder generated in “2.3.2 Porting project for e2 studio”.
Right click the file with same name and click [Properties]. Select [C/C++ Build] - [Settings] in the [Properties] dialog box. Check [Exclude source from build] and click [Apply and Close]. Do so for all files with same name.

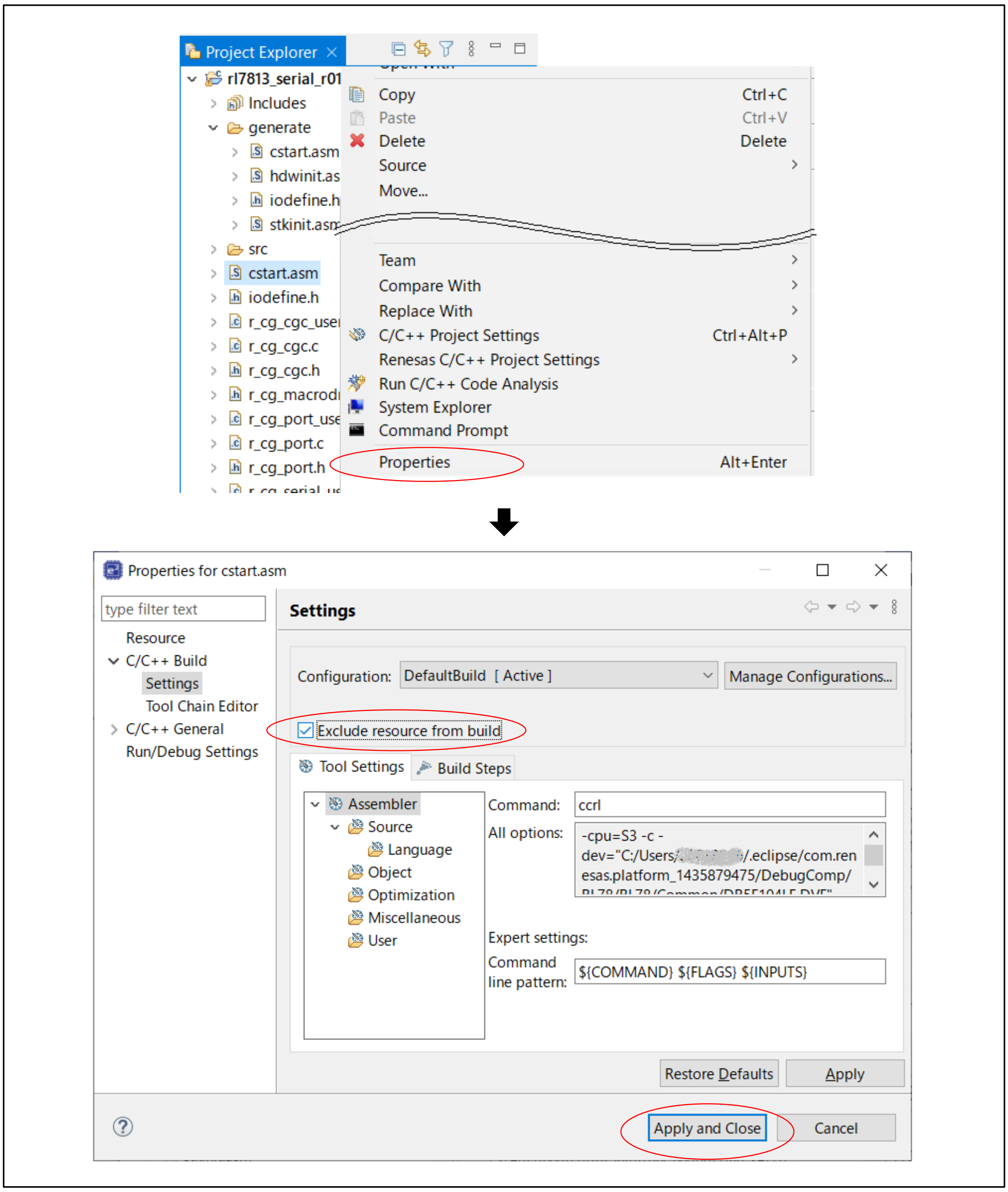
Figure 2-28 Excluding File with Same Name



Exclude the file with same name from the build target

Remark. “hdwinit.asm” in the “generate” folder will be automatically excluded by generating code in Step 3. If “hdwinit.asm” is not excluded from the build target after Step3, exclude it manually from the build target by the operation shown in Figure 2-29.

Figure 2-29 Excluding Resource from Build



Configuration: DefaultBuild [Active]

☑ Exclude resource from build

Command: ccrl

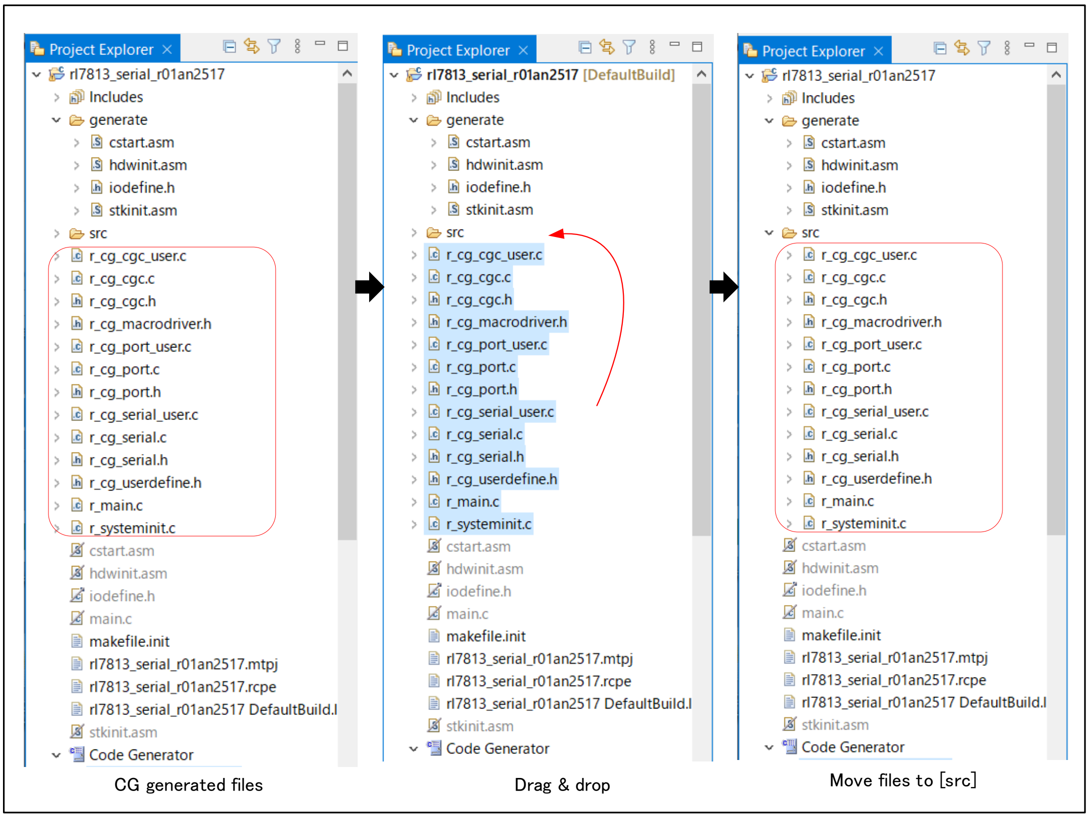
All options: -cpu=S3 -c -dev="C:/Users/[redacted]/.eclipse/com.renesas.platform_1435879475/DebugComp/RL78/RL78/Common/DR5F104LE.DVF"

Expert settings:

Command line pattern: ${COMMAND} ${FLAGS} ${INPUTS}

(2) Check whether files generated by Code Generator (CG) are registered in the “src” folder. If files shown in “Table 1-3 Outline of CG generated files and functions” don’t exist in the “src” folder, move CG generated files to the “src” folder by drag & drop.
If there is not the “src” folder, first create the “src” folder and move CG generated files.

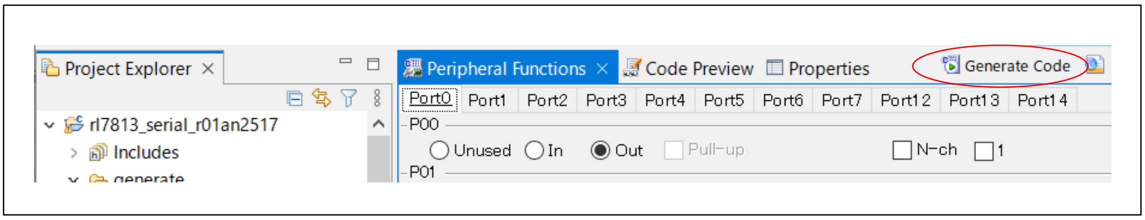
Figure 2-30 Moving CG Generated Files



CG generated files   Drag & drop   Move files to [src]

(3) Click the [Generate Code] button in the porting project. The codes in CG generated files with the same name under the “src” folder are overwritten with the code based on the changed device's peripheral functions.

Figure 2-31 [Generate Code] button



2.6 Editing Files in Porting Project

Some sample code doesn't use the code generated by Code Generator (CG) as it is and change the code, such as commenting out the code. However the code that have been changed in the original project is overwritten and deleted by generating code in the porting project in "2.5 Generating code for Porting Project". Therefore, check the code changes to CG generated files in the original project and edit CG generated files in the porting project to match the changes in the original project.
Furthermore, if any resources such as the channel of peripheral function or I/O port were changed when porting the sample code, other than the CG generated files are needed to edit the code.

＜Parts that needs to be edited＞

- The part that was changed in the original project, such as commenting out the code.

  → Make the same changes to the ported project as the original project
- The part that calls API function generated by CG to enable or stop the peripheral function operation if the used channel of peripheral function is changed in the porting project.

  → Change to the API function call for the changed channel in the porting project. However, the API function body is generated for the changed device in "2.5 Generating code for Porting Project", so that the API function body doesn't need to be edited.
- The part that sets value to input or output if the used I/O port is changed in the porting project.

  → Change to the I/O port name for the changed port in the porting project.

[Steps]

(1) Open the files with the same name of both the original project and the porting project in the editor. Check the code differences and reflect the deletion / changes of the original project in the porting project.

(2) Do Step (1) for all files.

(3) If there is a file in the porting project that is not in the original project, exclude it from the build target.

CS+ for CC: Right click the file name → Select [Remove from Project]

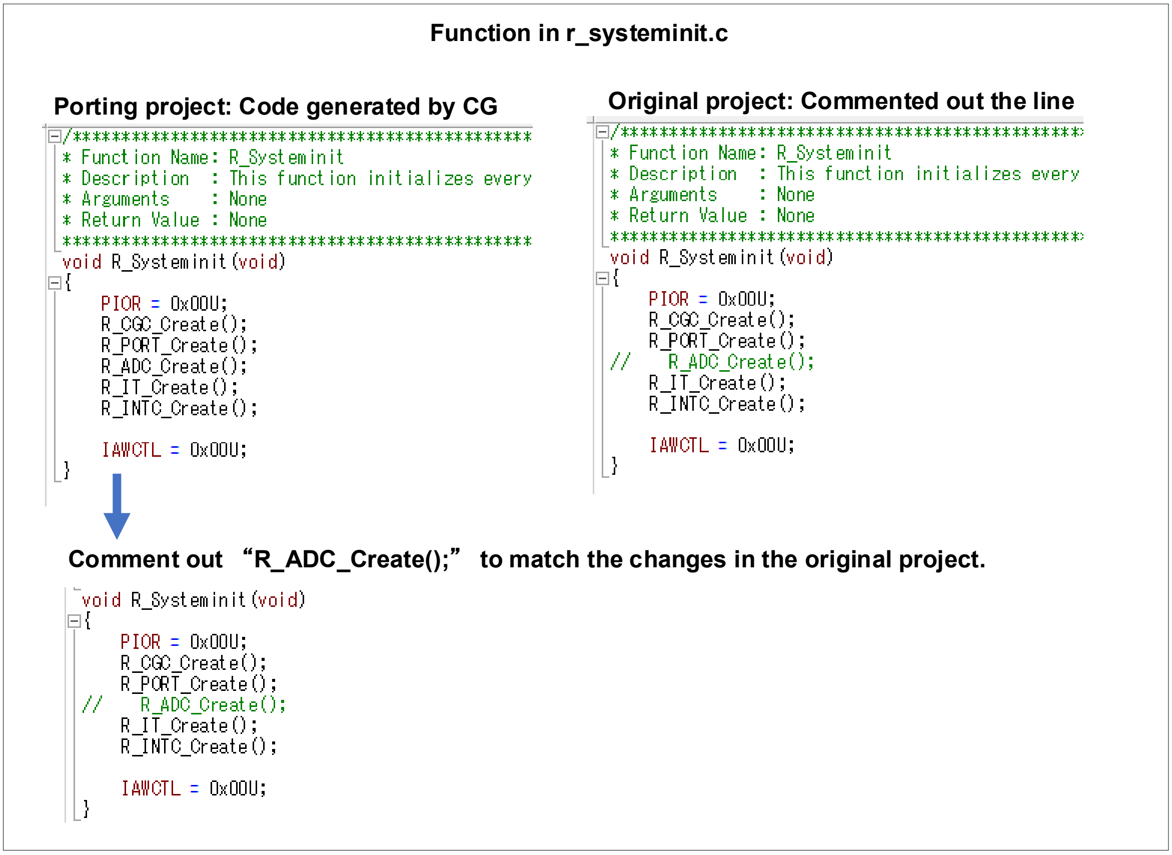
e^2 studio: Right click the file name → Select [Properties]

Select [C/C++ Build] in the [Properties] dialog box and check [Exclude resource from build]

Figure 2-32 shows the example of the changing the CG generated code, Figure 2-33 shows the example of the editing code due to the channel changes of peripheral function and

Figure 2-34 shows the example of the editing code due to the Port changes.

Figure 2-32 Code Editing Example of CG Generated Code



Function in r_systeminit.c

Porting project: Code generated by CG

```
/***********************************************
* Function Name: R_Systeminit
* Description  : This function initializes every
* Arguments    : None
* Return Value : None
************************************************
void R_Systeminit(void)
{
    PIOR = 0x00U;
    R_CGC_Create();
    R_PORT_Create();
    R_ADC_Create();
    R_IT_Create();
    R_INTC_Create();

    IAWCTL = 0x00U;
}
```

Original project: Commented out the line

```
/***********************************************
* Function Name: R_Systeminit
* Description  : This function initializes every
* Arguments    : None
* Return Value : None
************************************************
void R_Systeminit(void)
{
    PIOR = 0x00U;
    R_CGC_Create();
    R_PORT_Create();
//    R_ADC_Create();
    R_IT_Create();
    R_INTC_Create();

    IAWCTL = 0x00U;
}
```

Comment out “R_ADC_Create();” to match the changes in the original project.

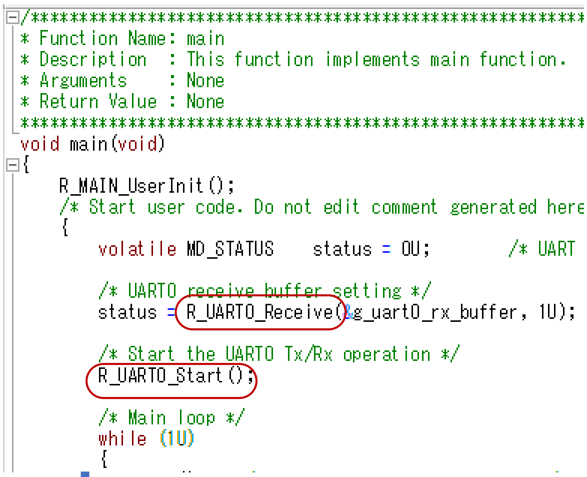
```
void R_Systeminit(void)
{
    PIOR = 0x00U;
    R_CGC_Create();
    R_PORT_Create();
//    R_ADC_Create();
    R_IT_Create();
    R_INTC_Create();

    IAWCTL = 0x00U;
}
```

Figure 2-33 Code Editing Example for Channel Changes of Peripheral Function

Function in r_main.c

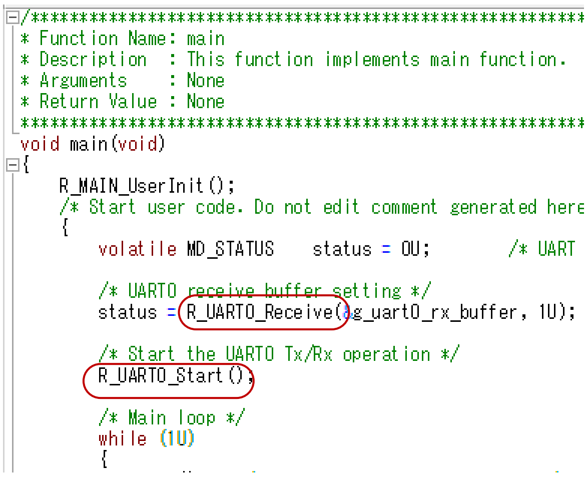
Porting project: Changed to UART1

API function body is generated for the UART1, but the API function call is not changed by the code generation.



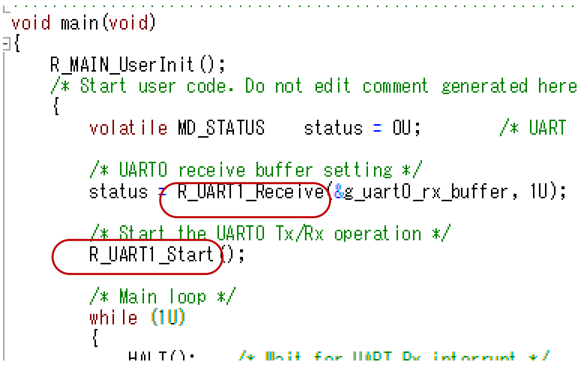
```
/***********************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here
    {
        volatile MD_STATUS    status = 0U;        /* UART

        /* UART0 receive buffer setting */
        status = R_UART0_Receive(&g_uart0_rx_buffer, 1U);

        /* Start the UART0 Tx/Rx operation */
        R_UART0_Start();

        /* Main loop */
        while (1U)
        {
```

Original project: Use UART0



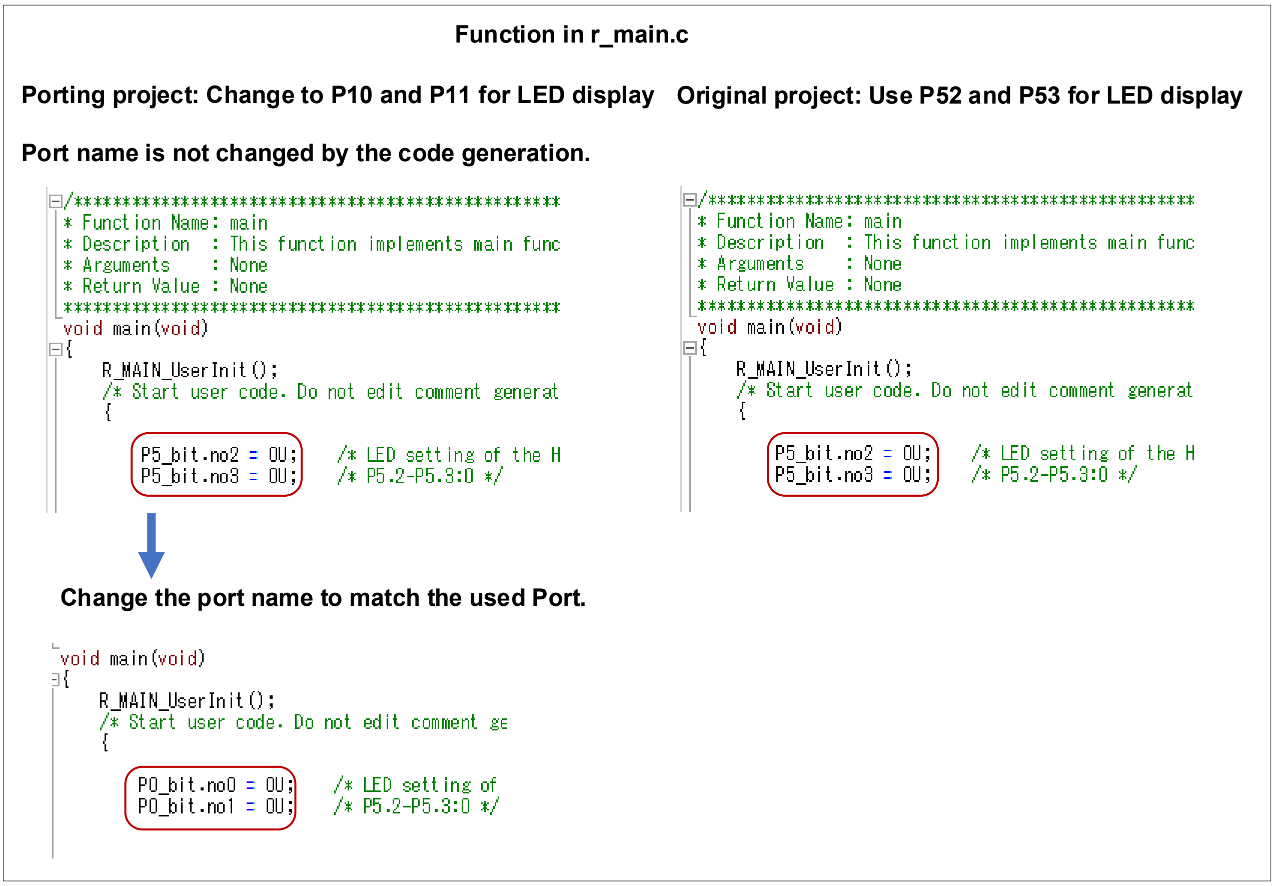
```
/***********************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here
    {
        volatile MD_STATUS    status = 0U;        /* UART

        /* UART0 receive buffer setting */
        status = R_UART0_Receive(&g_uart0_rx_buffer, 1U);

        /* Start the UART0 Tx/Rx operation */
        R_UART0_Start();

        /* Main loop */
        while (1U)
        {
```

Change the API function name to match the used channel.



```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here
    {
        volatile MD_STATUS    status = 0U;        /* UART

        /* UART0 receive buffer setting */
        status = R_UART1_Receive(&g_uart0_rx_buffer, 1U);

        /* Start the UART0 Tx/Rx operation */
        R_UART1_Start();

        /* Main loop */
        while (1U)
        {
            HALT();     /* Wait for UART Rx interrupt */
```

Figure 2-34 Code Editing Example for I/O Port Changes



Function in r_main.c

Porting project: Change to P10 and P11 for LED display

Port name is not changed by the code generation.

```
/***************************************************
* Function Name: main
* Description  : This function implements main func
* Arguments    : None
* Return Value : None
****************************************************
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generat
    {

        P5_bit.no2 = 0U;    /* LED setting of the H
        P5_bit.no3 = 0U;    /* P5.2-P5.3:0 */
```

Change the port name to match the used Port.

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment ge
    {

        P0_bit.no0 = 0U;    /* LED setting of
        P0_bit.no1 = 0U;    /* P5.2-P5.3:0 */
```

Original project: Use P52 and P53 for LED display

```
/***************************************************
* Function Name: main
* Description  : This function implements main func
* Arguments    : None
* Return Value : None
****************************************************
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generat
    {

        P5_bit.no2 = 0U;    /* LED setting of the H
        P5_bit.no3 = 0U;    /* P5.2-P5.3:0 */
```

2.7 Building Porting Project

Build the porting project after the target device was changed and the code was edited to match the original project.

[Steps]

(1) Build the porting project.

CS+ for CC: Select the [Build] – [Rebuild Project] menu

e^2 studio: Select the [Project] – [Build Project] menu

Caution. If any error occurs, resolve the error based on the error message.

3. Example of Porting

This chapter describes the example of porting sample code.

The application notes used for the porting example are shown below.

Porting Example 1

Item	Description
Application Note (Sample Code)	RL78/G13 Serial Array Unit for 3-Wire Serial I/O (Master Transmission/Reception) CC-RL Rev2.01 (R01AN2547EJ0201)
Porting Device	RL78/G12
Integrated development environment	CS+ for CC V8.07.00 from Renesas Electronics Corp.
C Compiler	CC-RL V1.11.00 from Renesas Electronics Corp.
Outline of Porting	Porting example with the I/O port changes

Porting Example 2

Item	Description
Application Note (Sample Code)	RL78/G13 Serial Interface IICA (for Master Transmission/Reception) Rev2.01 (R01AN2759EJ0201)
Porting Device	RL78/G14
Integrated development environment	e^2 studio V2022-01 (22.1.0) from Renesas Electronics Corp.
C Compiler	CC-RL V1.11.00 from Renesas Electronics Corp.
Outline of Porting	Porting example with editing the code generated by Code Generator

3.1 Porting Example 1

The procedure of porting the RL78/G13 sample code used in the application note “RL78/G13 Serial Array Unit for 3-Wire Serial I/O (Master Transmission/Reception) CC-RL R01AN2547EJ0201” (RL78/G13 AN) to the sample code for the RL78/G12 are described below.

3.1.1 Advance Preparation

Refer to “2. Operation Check Conditions”, “Table-1.1 Peripheral Functions to be Used and Their Uses”, “4.2 List of Pins to be Used” and “4.1 Hardware Configuration Example” in the RL78/G13 AN, and then check the used resources.
The configuration example of porting to the RL78/G12 is described below.

- Operation Check Conditions

The difference is shown in red letters. The RL78/G12 cannot operate with 32MHz, so that the operating frequency is changed to 24MHz.The various register setting value need to be changed because of the operating frequency changes. However, Code Generator (CG) changes the value according to the operation frequency, so the operation of the sample code is not affected by the operation frequency changes.

Table 3-1 Operation Check Conditions (Porting Example 1)

Item	RL78/G13	RL78/G12
Microcontroller used	RL78/G13（R5F100LEA）	RL78/G12（R5F1026A）
Operating Frequency	● High-speed on-chip oscillator (HOCO) clock: 32MHz ● CPU/peripheral hardware clock: 32MHz	● High-speed on-chip oscillator (HOCO) clock: 24MHz ● CPU/peripheral hardware clock: 24MHz
Operating Voltage	5.0V (can run on a voltage range of 2.9V to 5.5V) LVD operation (V_{LVD}): Reset mode 2.81V (2.76V to 2.87V)	5.0V (can run on a voltage range of 2.9V to 5.5V) LVD operation (V_{LVD}): Reset mode 2.81V (2.76V to 2.87V)

- Peripheral Functions to be Used and Their Uses

Use the same peripheral functions and channels for the RL78/G12 as for the RL78/G13 AN.

Table 3-2 Peripheral Functions to be Used (Porting Example 1)

RL78/G13

Peripheral Function	Use
Serial Array Unit 0 channel 0	CSI00 master transmission/reception
Timer Array Unit 0 channel 0	Interval timer operation

RL78/G12

Peripheral Function	Use
Serial Array Unit 0 channel 0	CSI00 master transmission/reception
Timer Array Unit 0 channel 0	Interval timer operation

- Pins to be Used

The difference is shown in red letters. There is not P00 pin in the RL78/G12 (R5F1026A). Therefore, P00 is changed to P41.

Table 3-3 Pins to be Used (Porting Example 1)

RL78/G13

Pin Name	I/O	Description
P10/SCK00/SCL00	Output	Serial clock output pin
P11/SI00/RxD0/TOOL RxD/SDA00	Input	Data reception pin
P12/SO00/TxD0/TOOLTxD	Output	Data transmission pin
P00/ANI17/TI00/TxD1	Input	BUSY signal detection pin

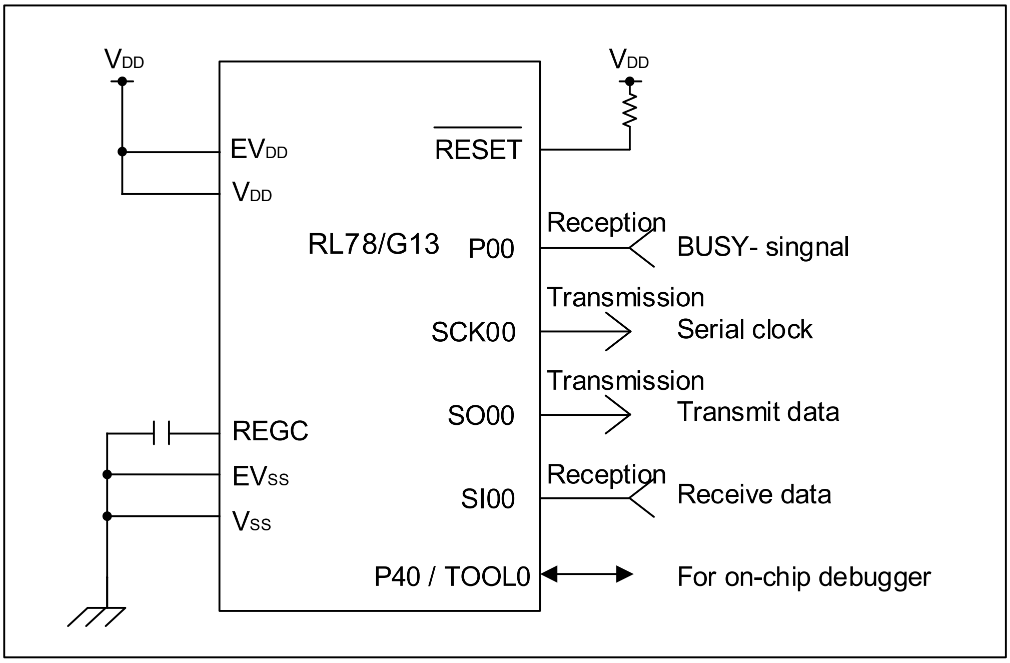
RL78/G12

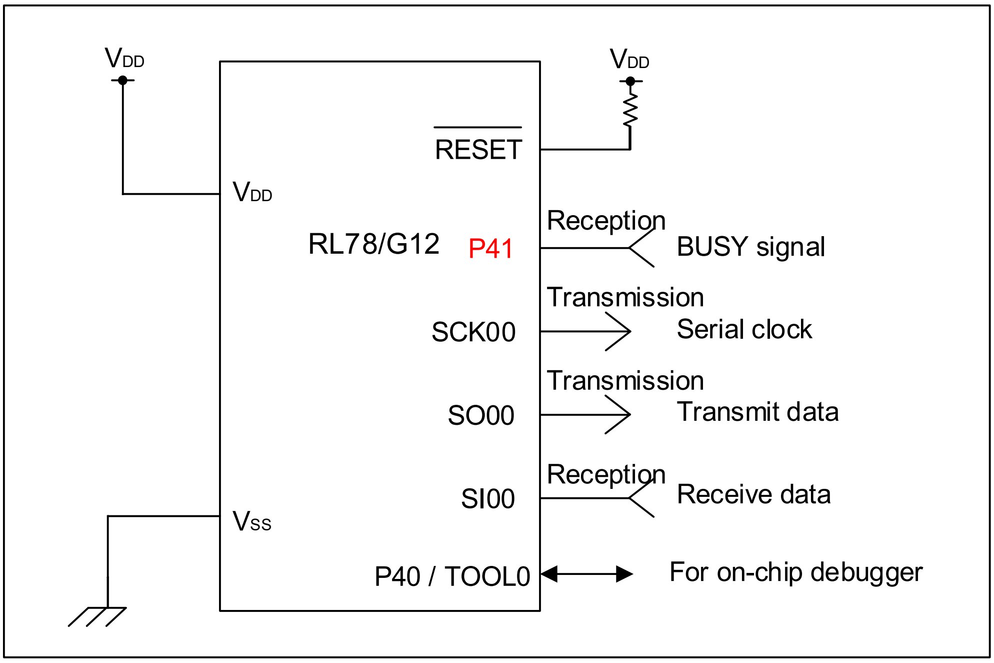
Pin Name	I/O	Description
P10/ANI16/PCLBUZ0/SCK00/SCL00	Output	Serial clock output pin
P11/ANI17/SI00/RxD0/SDA00 /TOOLRxD	Input	Data reception pin
P12/ANI18/SO00/TxD0/TOOLTxD	Output	Data transmission pin
P41/ANI22/SO01/SDA01/TI02/TO02/INTP1	Input	BUSY signal detection pin

- Hardware Configuration

The difference is shown in red letters. There is not P00 pin in the RL78/G12 (R5F1026A). Therefore, P00 is changed to P41.

Figure 3-1 Hardware Configuration Example (Porting Example 1)





Cautions: 1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to V_{DD} or V_{SS} via a resistor).

2. Connect any pins whose name begins with EV_{SS} to V_{SS} and any pins whose name begins with EV_{DD} to V_{DD}, respectively.

3. V_{DD} must be held at not lower than the reset release voltage (V_{LVD}) that is specified as LVD.

3.1.2 Copying Original Project

Refer to “2.2 Copying Original Project” and create a porting project.

3.1.3 Device Change for Porting project

Refer to “2.3.1 Porting Project for CS+ for CC” and change the target device for the porting project.

3.1.4 Code Generator Configuration for Porting Project

In order to replace the code related to the peripheral function settings with the code for the porting device, configure the peripheral functions in Code Generator of the porting project according to the Code Generator settings in the original project.

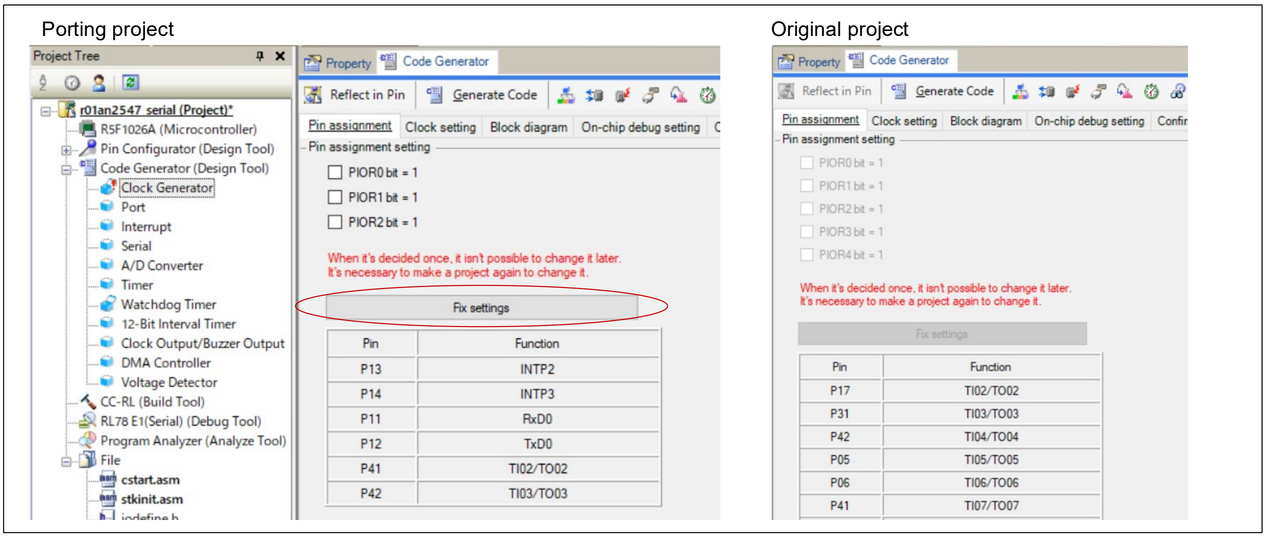
Start another CS+ for CC for the original project side by side and configure the peripheral functions for the porting project.

(1) Clock Generator

- Pin assignment

The peripheral I/O redirect function is not used in this porting example, click [Fix settings] without changing any settings.

Figure 3-2 Pin assignment (Porting Example 1)



Porting project

Pin assignment setting

☐ PIOR0 bit = 1
☐ PIOR1 bit = 1
☐ PIOR2 bit = 1

When it's decided once, it isn't possible to change it later.
It's necessary to make a project again to change it.

Pin	Function
P13	INTP2
P14	INTP3
P11	RxD0
P12	TxD0
P41	TI02/TO02
P42	TI03/TO03

Original project

Pin assignment setting

☐ PIOR0 bit = 1
☐ PIOR1 bit = 1
☐ PIOR2 bit = 1
☐ PIOR3 bit = 1
☐ PIOR4 bit = 1

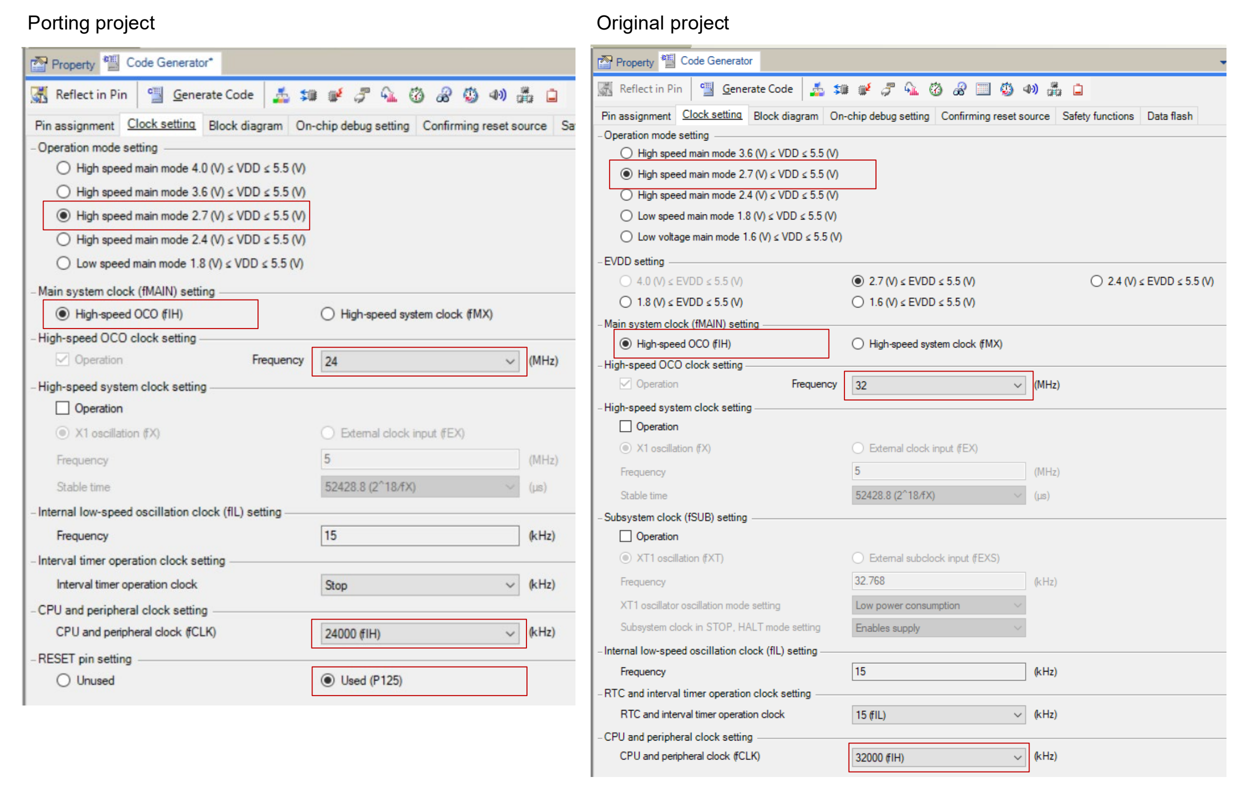
When it's decided once, it isn't possible to change it later.
It's necessary to make a project again to change it.

Pin	Function
P17	TI02/TO02
P31	TI03/TO03
P42	TI04/TO04
P05	TI05/TO05
P06	TI06/TO06
P41	TI07/TO07

- Clock setting

Some setting items are different between the original project and the porting project.
Configure the items like below in this porting example according to” Table 3-1 Operation Check Conditions (Porting Example 1)”.

Figure 3-3 Clock setting (Porting Example 1)



Porting project

Operation mode setting
○ High speed main mode 4.0 (V) ≤ VDD ≤ 5.5 (V)
○ High speed main mode 3.6 (V) ≤ VDD ≤ 5.5 (V)
● High speed main mode 2.7 (V) ≤ VDD ≤ 5.5 (V)
○ High speed main mode 2.4 (V) ≤ VDD ≤ 5.5 (V)
○ Low speed main mode 1.8 (V) ≤ VDD ≤ 5.5 (V)

Main system clock (fMAIN) setting
● High-speed OCO (fIH) ○ High-speed system clock (fMX)

High-speed OCO clock setting
☑ Operation Frequency 24 (MHz)

High-speed system clock setting
□ Operation
● X1 oscillation (fX) ○ External clock input (fEX)
Frequency 5 (MHz)
Stable time 52428.8 (2^18/fX) (µs)

Internal low-speed oscillation clock (fIL) setting
Frequency 15 (kHz)

Interval timer operation clock setting
Interval timer operation clock Stop (kHz)

CPU and peripheral clock setting
CPU and peripheral clock (fCLK) 24000 (fIH) (kHz)

RESET pin setting
○ Unused ● Used (P125)

Original project

Operation mode setting
○ High speed main mode 3.6 (V) ≤ VDD ≤ 5.5 (V)
● High speed main mode 2.7 (V) ≤ VDD ≤ 5.5 (V)
○ High speed main mode 2.4 (V) ≤ VDD ≤ 5.5 (V)
○ Low speed main mode 1.8 (V) ≤ VDD ≤ 5.5 (V)
○ Low voltage main mode 1.6 (V) ≤ VDD ≤ 5.5 (V)

EVDD setting
○ 4.0 (V) ≤ EVDD ≤ 5.5 (V) ● 2.7 (V) ≤ EVDD ≤ 5.5 (V) ○ 2.4 (V) ≤ EVDD ≤ 5.5 (V)
○ 1.8 (V) ≤ EVDD ≤ 5.5 (V) ○ 1.6 (V) ≤ EVDD ≤ 5.5 (V)

Main system clock (fMAIN) setting
● High-speed OCO (fIH) ○ High-speed system clock (fMX)

High-speed OCO clock setting
☑ Operation Frequency 32 (MHz)

High-speed system clock setting
□ Operation
● X1 oscillation (fX) ○ External clock input (fEX)
Frequency 5 (MHz)
Stable time 52428.8 (2^18/fX) (µs)

Subsystem clock (fSUB) setting
□ Operation
● XT1 oscillation (fXT) ○ External subclock input (fEXS)
Frequency 32.768 (kHz)
XT1 oscillator oscillation mode setting Low power consumption
Subsystem clock in STOP, HALT mode setting Enables supply

Internal low-speed oscillation clock (fIL) setting
Frequency 15 (kHz)

RTC and interval timer operation clock setting
RTC and interval timer operation clock 15 (fIL) (kHz)

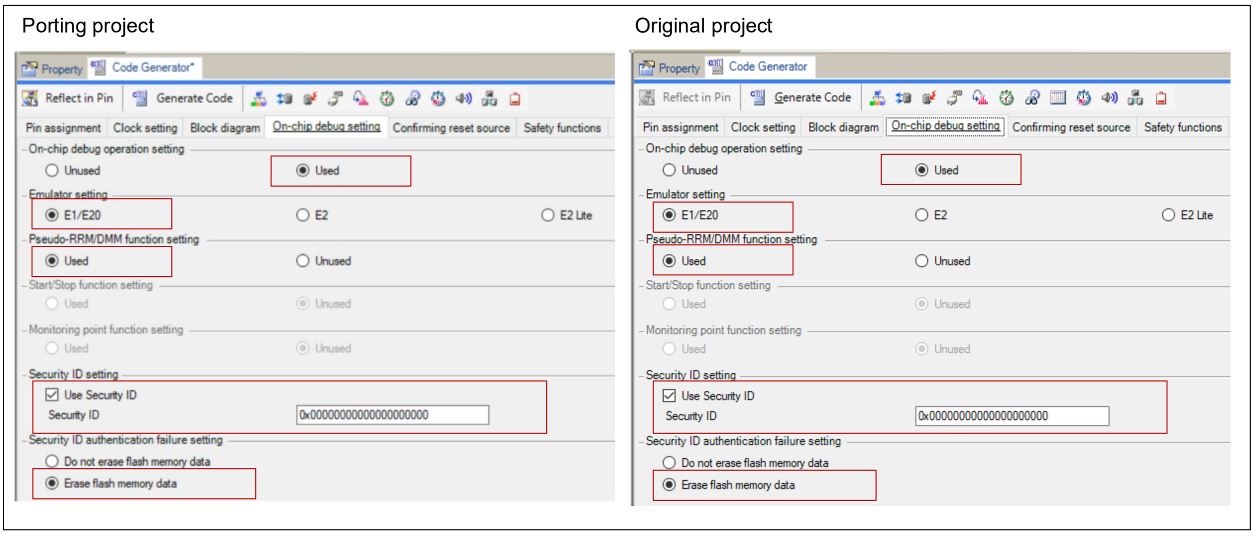
CPU and peripheral clock setting
CPU and peripheral clock (fCLK) 32000 (fIH) (kHz)

- On-chip debug setting

Configure the items with the same settings as the original project.

Remark. The emulator setting can be changed from the original project according to the emulator you are using.

Figure 3-4 On-chip debug setting (Porting Example 1)



Porting project

On-chip debug operation setting: ○ Unused ● Used

Emulator setting: ● E1/E20 ○ E2 ○ E2 Lite

Pseudo-RRM/DMM function setting: ● Used ○ Unused

Start/Stop function setting: ○ Used ● Unused

Monitoring point function setting: ○ Used ● Unused

Security ID setting: ☑ Use Security ID

Security ID: 0x00000000000000000000

Security ID authentication failure setting: ○ Do not erase flash memory data ● Erase flash memory data

Original project

On-chip debug operation setting: ○ Unused ● Used

Emulator setting: ● E1/E20 ○ E2 ○ E2 Lite

Pseudo-RRM/DMM function setting: ● Used ○ Unused

Start/Stop function setting: ○ Used ● Unused

Monitoring point function setting: ○ Used ● Unused

Security ID setting: ☑ Use Security ID

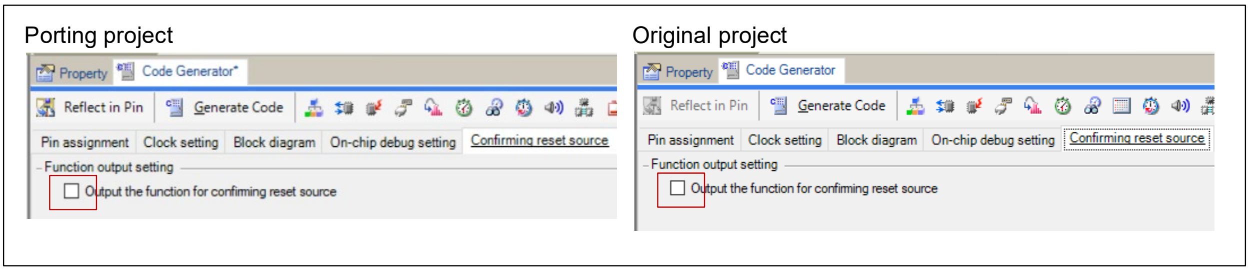
Security ID: 0x00000000000000000000

Security ID authentication failure setting: ○ Do not erase flash memory data ● Erase flash memory data

- Confirming reset source

Configure the item with the same setting as the original project.

Figure 3-5 Confirming reset source (Porting Example 1)



Porting project

Function output setting: □ Output the function for confirming reset source

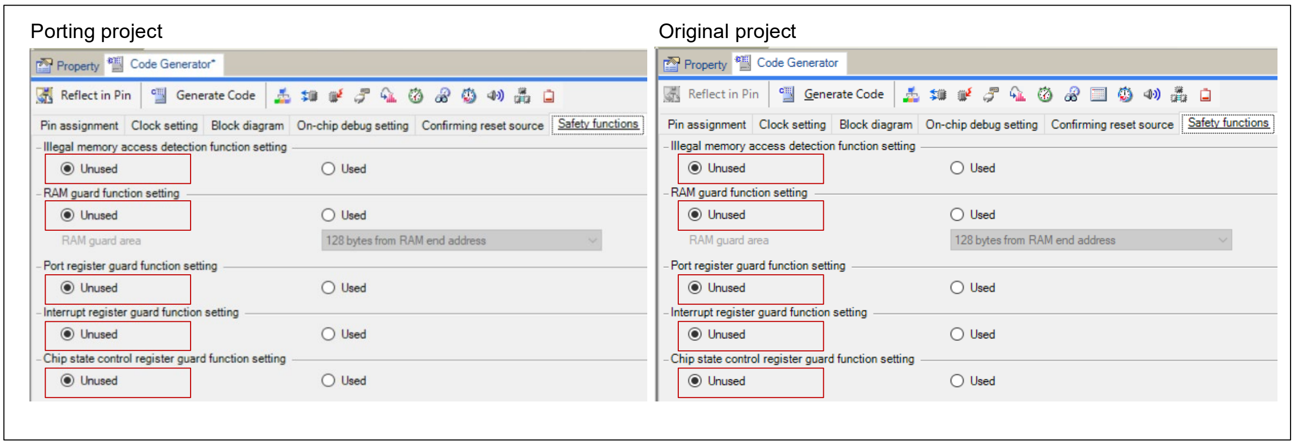
Original project

Function output setting: □ Output the function for confirming reset source

- Safety functions

Configure the items with the same settings as the original project.

Figure 3-6 Safety functions (Porting Example 1)



Porting project

Illegal memory access detection function setting: ● Unused ○ Used

RAM guard function setting: ● Unused ○ Used

RAM guard area: 128 bytes from RAM end address

Port register guard function setting: ● Unused ○ Used

Interrupt register guard function setting: ● Unused ○ Used

Chip state control register guard function setting: ● Unused ○ Used

Original project

Illegal memory access detection function setting: ● Unused ○ Used

RAM guard function setting: ● Unused ○ Used

RAM guard area: 128 bytes from RAM end address

Port register guard function setting: ● Unused ○ Used

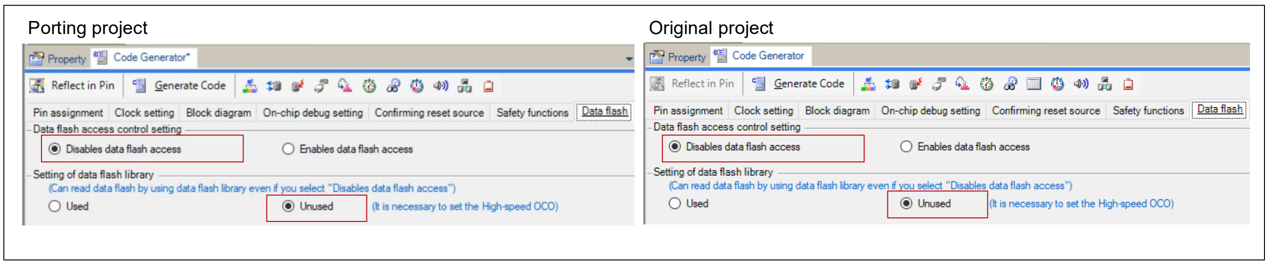
Interrupt register guard function setting: ● Unused ○ Used

Chip state control register guard function setting: ● Unused ○ Used

- Data flash

Configure the items with the same settings as the original project.

Figure 3-7 Data flash (Porting Example 1)



Porting project

Data flash access control setting: ● Disables data flash access ○ Enables data flash access

Setting of data flash library
(Can read data flash by using data flash library even if you select "Disables data flash access")
○ Used ● Unused (It is necessary to set the High-speed OCO)

Original project

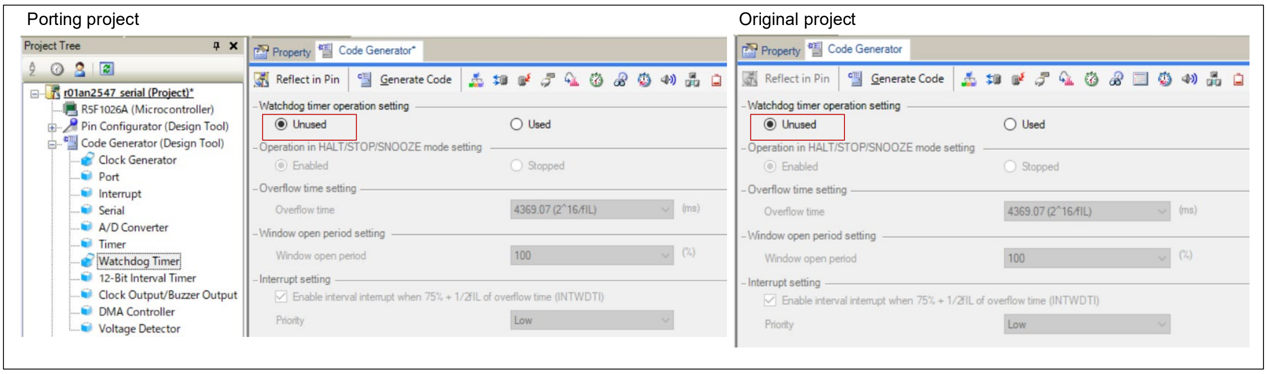
Data flash access control setting: ● Disables data flash access ○ Enables data flash access

Setting of data flash library
(Can read data flash by using data flash library even if you select "Disables data flash access")
○ Used ● Unused (It is necessary to set the High-speed OCO)

(2) Watchdog Timer

Configure the items with the same settings as the original project.

Figure 3-8 Watchdog Timer (Porting Example 1)



Porting project

Watchdog timer operation setting: ● Unused ○ Used

Operation in HALT/STOP/SNOOZE mode setting: ● Enabled ○ Stopped

Overflow time setting: Overflow time 4369.07 (2^16/fIL) (ms)

Window open period setting: Window open period 100 (%)

Interrupt setting: ☑ Enable interval interrupt when 75% + 1/2fIL of overflow time (INTWDTI)

Priority: Low

Original project

Watchdog timer operation setting: ● Unused ○ Used

Operation in HALT/STOP/SNOOZE mode setting: ● Enabled ○ Stopped

Overflow time setting: Overflow time 4369.07 (2^16/fIL) (ms)

Window open period setting: Window open period 100 (%)

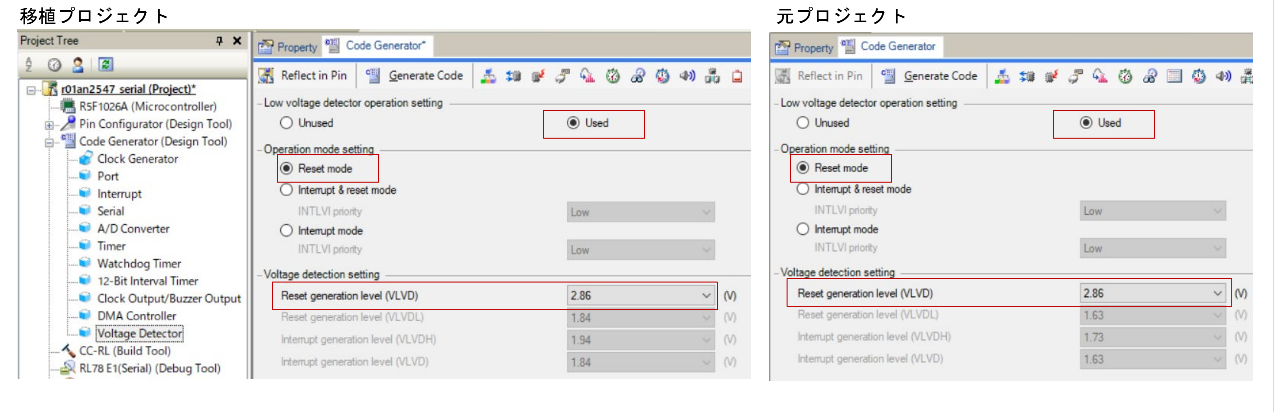
Interrupt setting: ☑ Enable interval interrupt when 75% + 1/2fIL of overflow time (INTWDTI)

Priority: Low

(3) Voltage Detector

Configure the items with the same settings as the original project.

Figure 3-9 Voltage Detector (Porting Example 1)



移植プロジェクト

Project Tree

- r01an2547 serial (Project)*
  - R5F1026A (Microcontroller)
  - Pin Configurator (Design Tool)
  - Code Generator (Design Tool)
    - Clock Generator
    - Port
    - Interrupt
    - Serial
    - A/D Converter
    - Timer
    - Watchdog Timer
    - 12-Bit Interval Timer
    - Clock Output/Buzzer Output
    - DMA Controller
    - Voltage Detector
  - CC-RL (Build Tool)
  - RL78 E1(Serial) (Debug Tool)

Property | Code Generator*

Reflect in Pin | Generate Code

Low voltage detector operation setting
○ Unused ● Used

Operation mode setting
● Reset mode
○ Interrupt & reset mode
INTLVI priority Low
○ Interrupt mode
INTLVI priority Low

Voltage detection setting
Reset generation level (VLVD) 2.86 (V)
Reset generation level (VLVDL) 1.84 (V)
Interrupt generation level (VLVDH) 1.94 (V)
Interrupt generation level (VLVD) 1.84 (V)

元プロジェクト

Property | Code Generator

Reflect in Pin | Generate Code

Low voltage detector operation setting
○ Unused ● Used

Operation mode setting
● Reset mode
○ Interrupt & reset mode
INTLVI priority Low
○ Interrupt mode
INTLVI priority Low

Voltage detection setting
Reset generation level (VLVD) 2.86 (V)
Reset generation level (VLVDL) 1.63 (V)
Interrupt generation level (VLVDH) 1.73 (V)
Interrupt generation level (VLVD) 1.63 (V)

(4) Peripheral function used in sample code

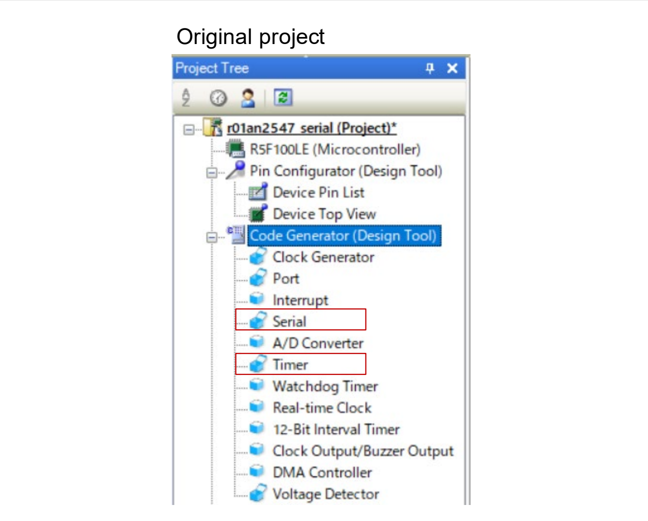
Check icons under the [Code Generator (Design Tool)] node of the original project and configure the peripheral functions for the porting project to match the configuration for the original project.

“Serial” and “Timer” are used in the original project (RL78/G13 AN). Note that Port is configured in Step (5).

: The peripheral function has any setting. (Used)

: The peripheral function has no setting. (Unused)

Figure 3-10 Peripheral Function Has Settings (Porting Example 1)



Original project

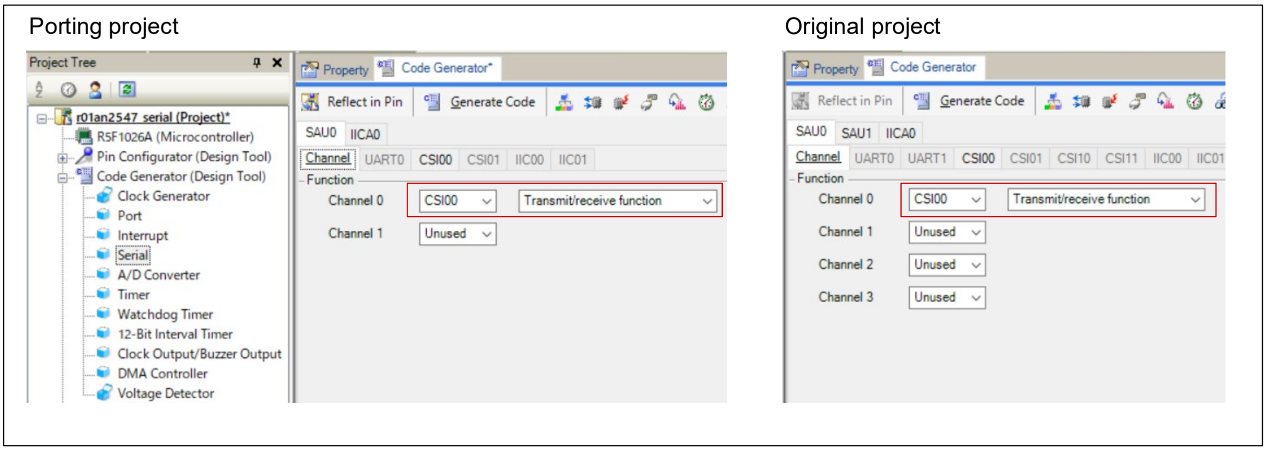
Project Tree

- r01an2547 serial (Project)*
  - R5F100LE (Microcontroller)
  - Pin Configurator (Design Tool)
    - Device Pin List
    - Device Top View
  - Code Generator (Design Tool)
    - Clock Generator
    - Port
    - Interrupt
    - Serial
    - A/D Converter
    - Timer
    - Watchdog Timer
    - Real-time Clock
    - 12-Bit Interval Timer
    - Clock Output/Buzzer Output
    - DMA Controller
    - Voltage Detector

- Serial

Configure the items with the same settings as the original project.

Figure 3-11 Serial – CSI00 (Porting Example 1)



Porting project

Project Tree

r01an2547_serial (Project)*
R5F1026A (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
Clock Generator
Port
Interrupt
Serial
A/D Converter
Timer
Watchdog Timer
12-Bit Interval Timer
Clock Output/Buzzer Output
DMA Controller
Voltage Detector

Property | Code Generator*

Reflect in Pin | Generate Code

SAU0 | IICA0

Channel | UART0 | CSI00 | CSI01 | IIC00 | IIC01

Function

Channel 0: CSI00 – Transmit/receive function

Channel 1: Unused

Original project

Property | Code Generator

Reflect in Pin | Generate Code

SAU0 | SAU1 | IICA0

Channel | UART0 | UART1 | CSI00 | CSI01 | CSI10 | CSI11 | IIC00 | IIC01

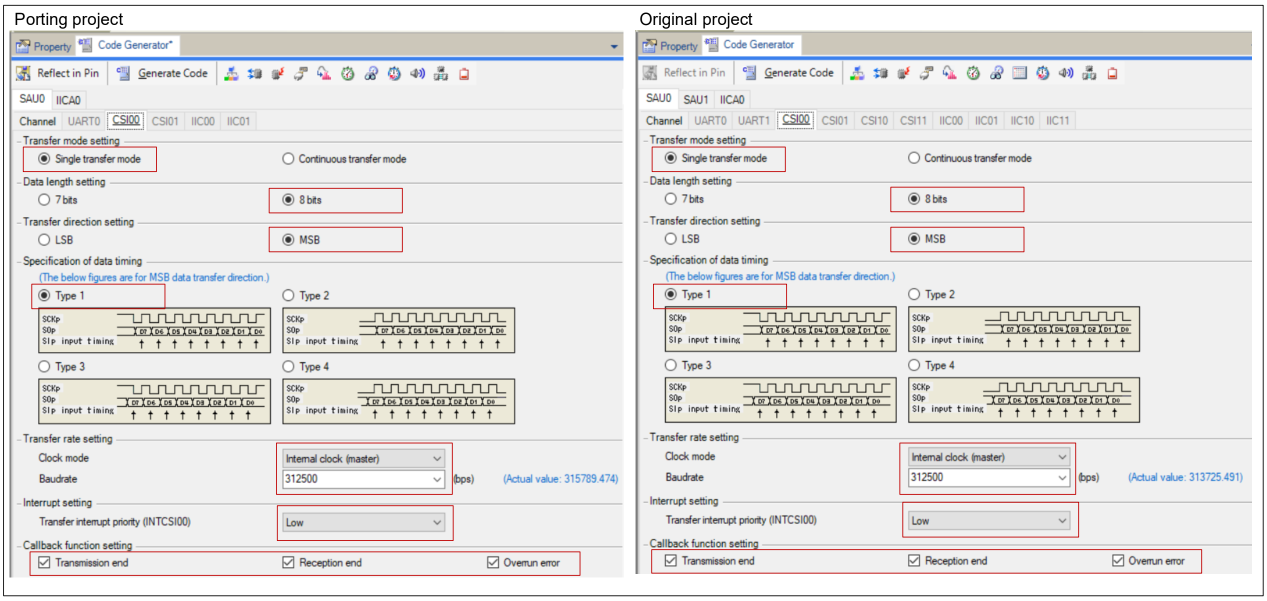
Function

Channel 0: CSI00 – Transmit/receive function

Channel 1: Unused

Channel 2: Unused

Channel 3: Unused



Porting project

Property | Code Generator*

Reflect in Pin | Generate Code

SAU0 | IICA0

Channel | UART0 | CSI00 | CSI01 | IIC00 | IIC01

Transfer mode setting
● Single transfer mode ○ Continuous transfer mode

Data length setting
○ 7 bits ● 8 bits

Transfer direction setting
○ LSB ● MSB

Specification of data timing
(The below figures are for MSB data transfer direction.)
● Type 1 ○ Type 2
○ Type 3 ○ Type 4

Transfer rate setting
Clock mode: Internal clock (master)
Baudrate: 312500 (bps) (Actual value: 315789.474)

Interrupt setting
Transfer interrupt priority (INTCSI00): Low

Callback function setting
☑ Transmission end ☑ Reception end ☑ Overrun error

Original project

Property | Code Generator

Reflect in Pin | Generate Code

SAU0 | SAU1 | IICA0

Channel | UART0 | UART1 | CSI00 | CSI01 | CSI10 | CSI11 | IIC00 | IIC01 | IIC10 | IIC11

Transfer mode setting
● Single transfer mode ○ Continuous transfer mode

Data length setting
○ 7 bits ● 8 bits

Transfer direction setting
○ LSB ● MSB

Specification of data timing
(The below figures are for MSB data transfer direction.)
● Type 1 ○ Type 2
○ Type 3 ○ Type 4

Transfer rate setting
Clock mode: Internal clock (master)
Baudrate: 312500 (bps) (Actual value: 313725.491)

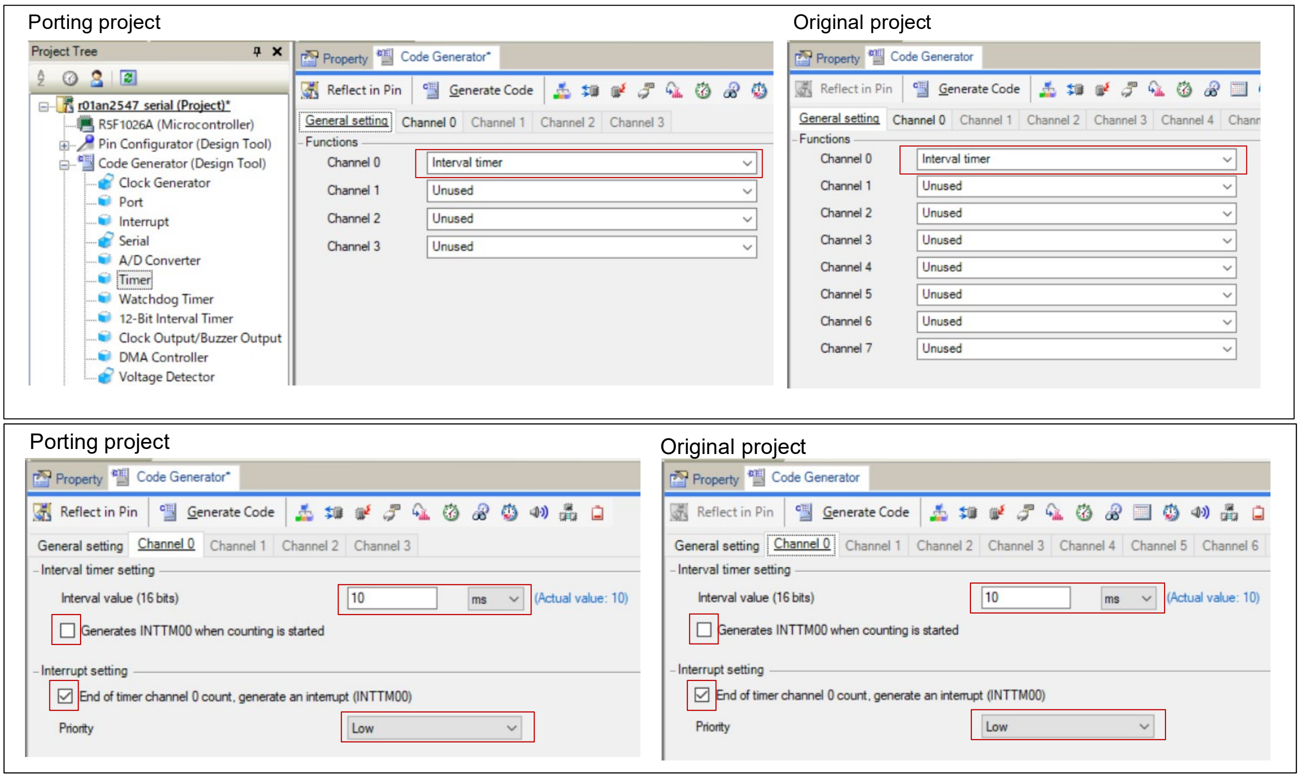
Interrupt setting
Transfer interrupt priority (INTCSI00): Low

Callback function setting
☑ Transmission end ☑ Reception end ☑ Overrun error

- Timer

Configure the items with the same settings as the original project.

Figure 3-12 Timer – channel 0 (Porting Example 1)



Porting project – Functions:

Channel 0: Interval timer
Channel 1: Unused
Channel 2: Unused
Channel 3: Unused

Original project – Functions:

Channel 0: Interval timer
Channel 1: Unused
Channel 2: Unused
Channel 3: Unused
Channel 4: Unused
Channel 5: Unused
Channel 6: Unused
Channel 7: Unused

Porting project – Channel 0:

Interval timer setting
Interval value (16 bits) 10 ms (Actual value: 10)
□ Generates INTTM00 when counting is started
Interrupt setting
☑ End of timer channel 0 count, generate an interrupt (INTTM00)
Priority Low

Original project – Channel 0:

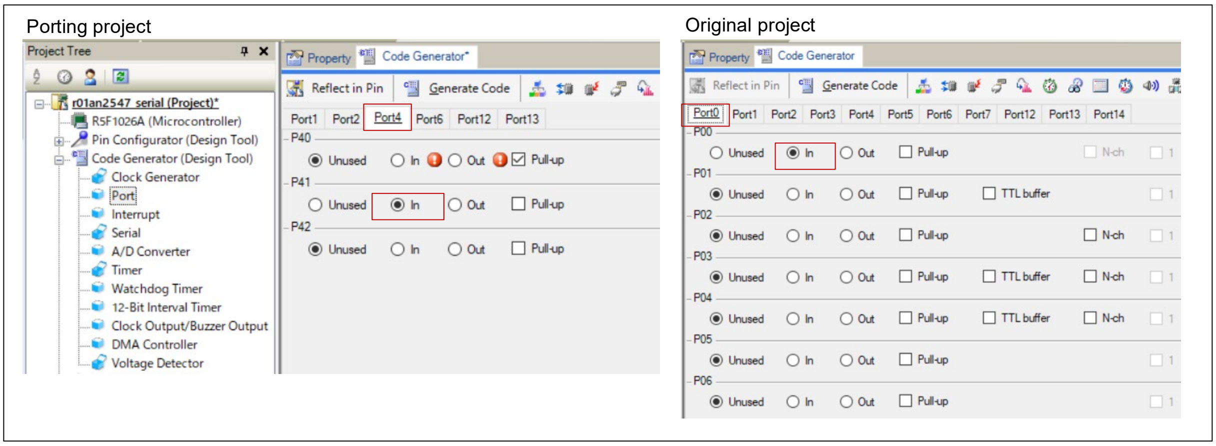
Interval timer setting
Interval value (16 bits) 10 ms (Actual value: 10)
□ Generates INTTM00 when counting is started
Interrupt setting
☑ End of timer channel 0 count, generate an interrupt (INTTM00)
Priority Low

(5) Port

P00 is used as the BUSY signal detection pin in the original project, but P41 is used in the porting project. Set P41 to the input port.

Caution. Provide proper treatment for unused pins so that their electrical specifications are observed.
Connect each of any unused input-only ports to VDD or VSS via a separate resistor.

Figure 3-13 Port (Porting Example 1)



Porting project – Port4:

P40 ● Unused ○ In ○ Out ☑ Pull-up
P41 ○ Unused ● In ○ Out □ Pull-up
P42 ● Unused ○ In ○ Out □ Pull-up

Original project – Port0:

P00 ○ Unused ● In ○ Out □ Pull-up □ N-ch
P01 ● Unused ○ In ○ Out □ Pull-up □ TTL buffer
P02 ● Unused ○ In ○ Out □ Pull-up □ N-ch
P03 ● Unused ○ In ○ Out □ Pull-up □ TTL buffer □ N-ch
P04 ● Unused ○ In ○ Out □ Pull-up □ TTL buffer □ N-ch
P05 ● Unused ○ In ○ Out □ Pull-up
P06 ● Unused ○ In ○ Out □ Pull-up

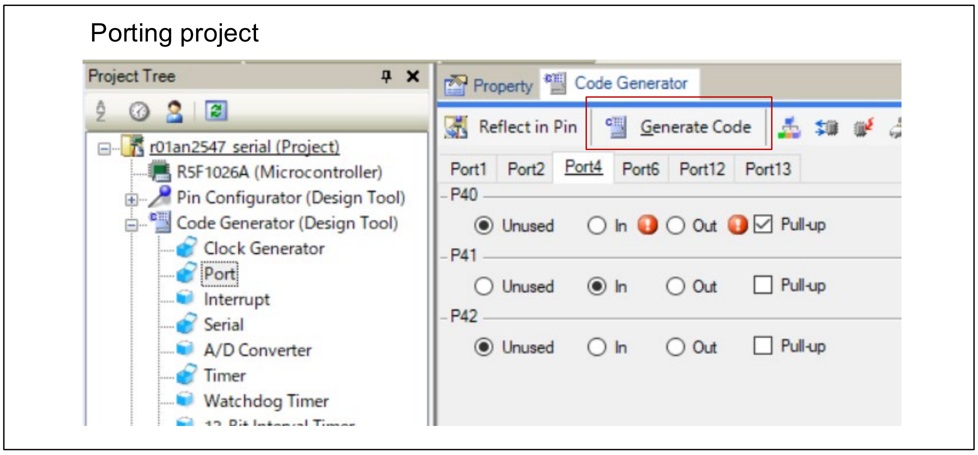
(6) Saving configuration

Save the configuration for the porting project. Select the [File] – [Save Project] menu of CS+.

3.1.5 Code Generation for Porting Project

Generate the code for the porting project according to the setting of Code Generator (CG). Click the [Generate Code] button.

Figure 3-14 Code Generation (Porting Example 1)

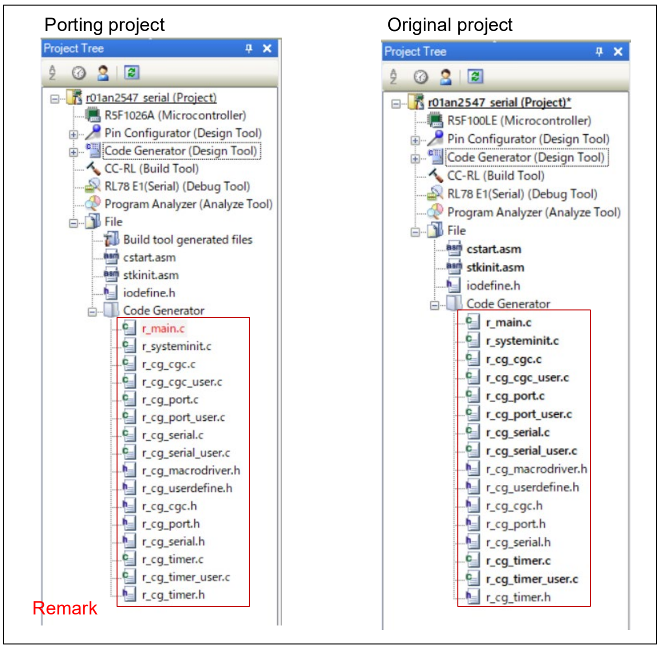

Porting project
Project Tree
r01an2547_serial (Project)
R5F1026A (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
Clock Generator
Port
Interrupt
Serial
A/D Converter
Timer
Watchdog Timer
Property
Code Generator
Reflect in Pin
Generate Code
Port1
Port2
Port4
Port6
Port12
Port13
P40
● Unused ○ In ○ Out ☑ Pull-up
P41
○ Unused ● In ○ Out □ Pull-up
P42
● Unused ○ In ○ Out □ Pull-up

3.1.6 Editing File in Porting Project

Check if there are any changes such as commenting the CG generated code in the original project and if there are any changes, make same changes to the porting project. Open the file with the same name of the original project and the ported project in the editor, check the difference in the code, and reflect the deletion / change of the original project in the ported project.

Also, change the code for the BUSY signal detection pin because P00 is used for it in the original project but P00 is changed to P41 in the porting project.

Figure 3-15 Files to be checked (Porting Example 1)



Remark. Another [Code Generator] node may be added after code generation in the porting project but ignore it because of no effect on the sample code.

Table 3-4 Files to be Checked and Whether Editing is Required (Porting Example 1)

Porting Project File Name	Whether Editing is Required: ✓: Required ―: Not Required	Action to be Taken When Difference is Found
r_main.c	✓	Edit code to operate BUSY signal detection pin
r_systeminit.c	―	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_cgc.c	―	Code difference in the function "R_CGC_Create" derives from the difference in device specifications between the source and target devices. No editing is necessary.
r_cg_cgc_user.c	―	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_port.c	―	Code difference in the function "R_PORT_Create" derives from the changing from P00 to P41 for BUSY signal detection pin. No editing is necessary. Caution on unused ports: When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to VDD or VSS via a resistor).
r_cg_port_user.c	―	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_serial.c	―	Code difference in the function "R_CSI00_Create" derives from the difference in device specifications between the source and target devices. No editing is necessary.
r_cg_serial_user.c	―	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_microdriver.h	―	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_userdefine.h	―	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_cgc.h	―	Difference in macro definitions and type definitions derives from the difference in device specifications between the source and target devices. No editing is necessary.
r_cg_port.h	―	Difference in macro definitions derives from the difference in device specifications between the source and target devices. No editing is necessary.
r_cg_serial.h	―	Difference in macro definitions derives from the difference in device specifications between the source and target devices. No editing is necessary.
r_cg_timer.c	―	Code difference in the function "R_TAU0_Create" derives from the difference in device specifications between the source and target devices. No editing is necessary.
r_cg_timer_user.c	―	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_timer.h	―	Difference in macro definitions derives from the difference in device specifications between the source and target devices. No editing is necessary.

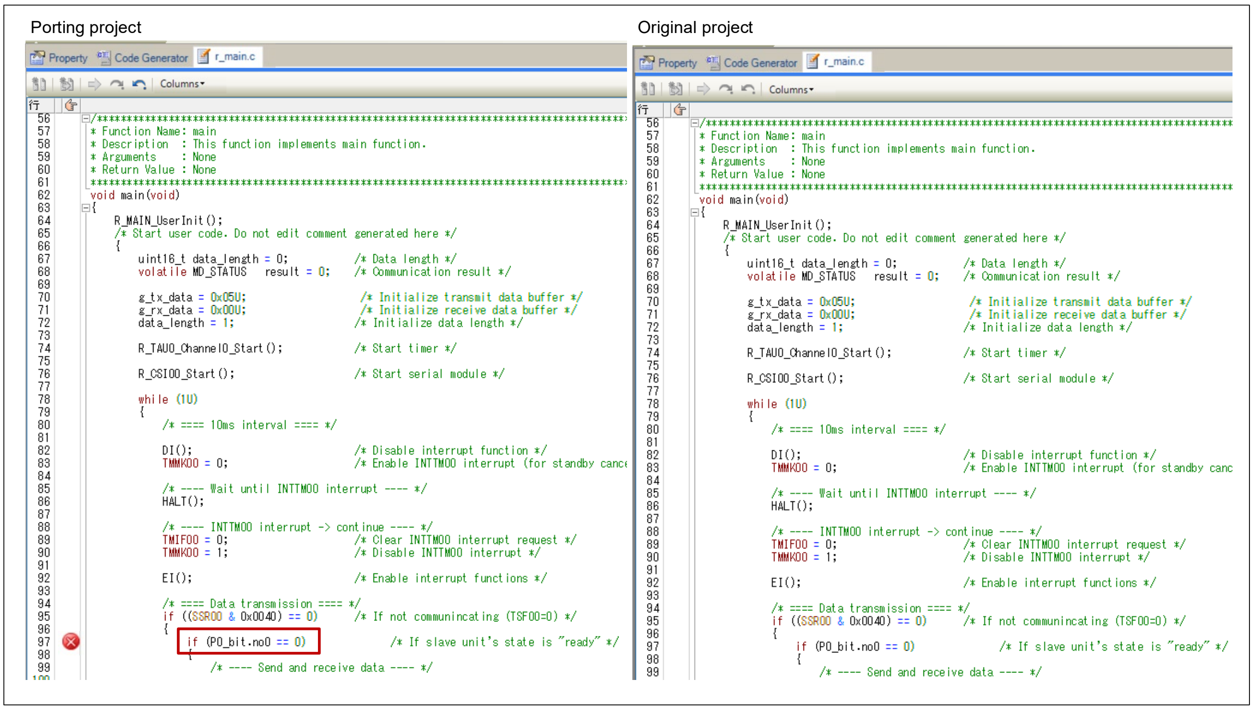
(a) r_main.c

The code for the BUSY signal detection pin is on line 97. Change the code for P00 to the code for P41.

Before the change: P0_bit.no0

After the change: P4_bit.no1

Figure 3-16 r_main.c (Porting Example 1)

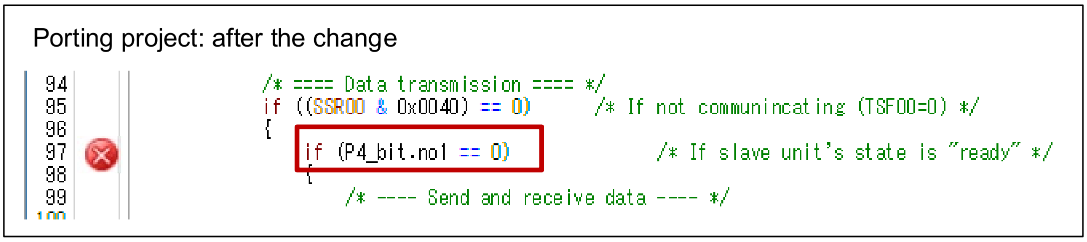


Porting project

```
56  /*****************************************************************************
57  * Function Name: main
58  * Description  : This function implements main function.
59  * Arguments    : None
60  * Return Value : None
61  *****************************************************************************
62  void main(void)
63  {
64      R_MAIN_UserInit();
65      /* Start user code. Do not edit comment generated here */
66      {
67          uint16_t data_length = 0;          /* Data length */
68          volatile MD_STATUS  result = 0;    /* Communication result */
69
70          g_tx_data = 0x05U;                  /* Initialize transmit data buffer */
71          g_rx_data = 0x00U;                  /* Initialize receive data buffer */
72          data_length = 1;                   /* Initialize data length */
73
74          R_TAU0_Channel0_Start();           /* Start timer */
75
76          R_CSI00_Start();                   /* Start serial module */
77
78          while (1U)
79          {
80              /* ==== 10ms interval ==== */
81
82              DI();                          /* Disable interrupt function */
83              TMMK00 = 0;                    /* Enable INTTM00 interrupt (for standby cance
84
85              /* ---- Wait until INTTM00 interrupt ---- */
86              HALT();
87
88              /* ---- INTTM00 interrupt -> continue ---- */
89              TMIF00 = 0;                    /* Clear INTTM00 interrupt request */
90              TMMK00 = 1;                    /* Disable INTTM00 interrupt */
91
92              EI();                          /* Enable interrupt functions */
93
94              /* ==== Data transmission ==== */
95              if ((SSR00 & 0x0040) == 0)     /* If not communincating (TSF00=0) */
96              {
97                  if (P0_bit.no0 == 0)            /* If slave unit's state is "ready" */
98                  {
99                      /* ---- Send and receive data ---- */
```

Original project

```
56  /*****************************************************************************
57  * Function Name: main
58  * Description  : This function implements main function.
59  * Arguments    : None
60  * Return Value : None
61  *****************************************************************************
62  void main(void)
63  {
64      R_MAIN_UserInit();
65      /* Start user code. Do not edit comment generated here */
66      {
67          uint16_t data_length = 0;          /* Data length */
68          volatile MD_STATUS  result = 0;    /* Communication result */
69
70          g_tx_data = 0x05U;                  /* Initialize transmit data buffer */
71          g_rx_data = 0x00U;                  /* Initialize receive data buffer */
72          data_length = 1;                   /* Initialize data length */
73
74          R_TAU0_Channel0_Start();           /* Start timer */
75
76          R_CSI00_Start();                   /* Start serial module */
77
78          while (1U)
79          {
80              /* ==== 10ms interval ==== */
81
82              DI();                          /* Disable interrupt function */
83              TMMK00 = 0;                    /* Enable INTTM00 interrupt (for standby canc
84
85              /* ---- Wait until INTTM00 interrupt ---- */
86              HALT();
87
88              /* ---- INTTM00 interrupt -> continue ---- */
89              TMIF00 = 0;                    /* Clear INTTM00 interrupt request */
90              TMMK00 = 1;                    /* Disable INTTM00 interrupt */
91
92              EI();                          /* Enable interrupt functions */
93
94              /* ==== Data transmission ==== */
95              if ((SSR00 & 0x0040) == 0)     /* If not communincating (TSF00=0) */
96              {
97                  if (P0_bit.no0 == 0)            /* If slave unit's state is "ready" */
98                  {
99                      /* ---- Send and receive data ---- */
```



Porting project: after the change

```
94              /* ==== Data transmission ==== */
95              if ((SSR00 & 0x0040) == 0)     /* If not communincating (TSF00=0) */
96              {
97                  if (P4_bit.no1 == 0)            /* If slave unit's state is "ready" */
98                  {
99                      /* ---- Send and receive data ---- */
```

3.1.7 Building Porting Project

Build the porting project after the target device was changed and the code was edited to match the original project.

Select the [Build] – [Rebuild Project] menu of CS+.

3.2 Porting Example 2

The procedure of porting the RL78/G13 sample code used in the application note “RL78/G13 Serial Interface IICA (for Master Transmission/Reception) R01AN2759EJ0201” (RL78/G13 AN) to the sample code for the RL78/G14 are described below

3.2.1 Advance Preparation

Refer to “2. Operation Check Conditions”, “Table-1.1 Peripheral Function to be Used and Its Use”, “4.2 List of Pins to be Used” and “4.1 Hardware Configuration Example” in the RL78/G13 AN, and then check the used resources.

The configuration example of porting to the RL78/G14 is described below.

- Operation Check Conditions

There is no difference. It is possible to port without changing the operating frequency and the operating voltage.

Table 3-5 Operation Check Conditions (Porting Example 2)

Item	RL78/G13	RL78/G14
Microcontroller used	RL78/G13（R5F100LEA）	RL78/G14（R5F104MLA）
Operating Frequency	● High-speed on-chip oscillator (HOCO) clock: 32MHz ● CPU/peripheral hardware clock: 32MHz	● High-speed on-chip oscillator (HOCO) clock: 32MHz ● CPU/peripheral hardware clock: 32MHz
Operating Voltage	5.0V (can run on a voltage range of 2.7V to 5.5V) LVD operation (V_{LVD}): Reset mode 2.81V (2.76V to 2.87V)	5.0V (can run on a voltage range of 2.7V to 5.5V) LVD operation (V_{LVD}): Reset mode 2.81V (2.76V to 2.87V)

- Peripheral Functions to be Used and Their Uses

Use the same peripheral functions and channels for the RL78/G14 as for the RL78/G13 AN.

Table 3-6 Peripheral Functions to be Used (Porting Example 2)

RL78/G13

Peripheral Function	Use
Serial Interface IICA0	IIC communication in a single master system (using the SCLA0 and SDAA0 pins)
12-bit Interval Timer	1ms interval measurement
Timer Array Unit 0 channel 2	maximum 2 ms interval measurement

RL78/G14

Peripheral Function	Use
Serial Interface IICA0	IIC communication in a single master system (using the SCLA0 and SDAA0 pins)
12-bit Interval Timer	1ms interval measurement
Timer Array Unit 0 channel 2	maximum 2 ms interval measurement

- Pins to be Used

Use the same pins for the RL78/G14 as for the RL78/G13 AN.

Table 3-7 Pins to be Used (Porting Example 2)

RL78/G13

Pin Name	I/O	Description
P60/SCLA0	Input/Output	Serial clock input/output pin
P61/SDAA0	Input/Output	Serial data transmission/reception pin
P62	Output	Signal to drive Status LED
P63	Output	Signal to drive Error LED
P137	Input	Switch input signal for designating operation start

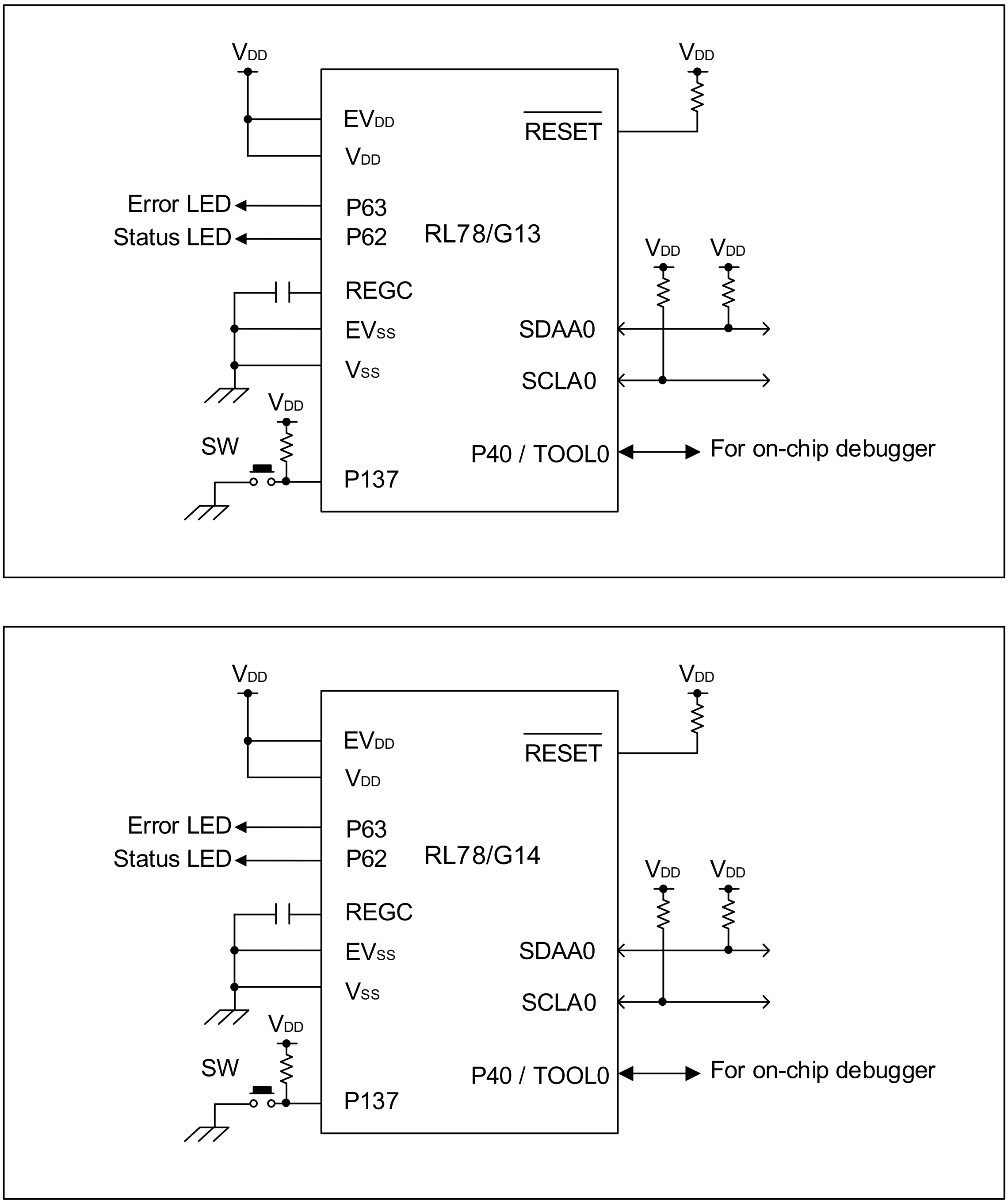
RL78/G14

Pin Name	I/O	Description
P60/SCLA0	Input/Output	Serial clock input/output pin
P61/SDAA0	Input/Output	Serial data transmission/reception pin
P62	Output	Signal to drive Status LED
P63	Output	Signal to drive Error LED
P137	Input	Switch input signal for designating operation start

- Hardware Configuration

Use the same hardware configuration for the RL78/G14 as for the RL78/G13 AN.

Figure 3-17 Hardware Configuration Example (Porting Example 2)



Cautions: 1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to V_{DD} or V_{SS} via a resistor).

2. Connect any pins whose name begins with EV_{SS} to V_{SS} and any pins whose name begins with EV_{DD} to V_{DD}, respectively.

3. V_{DD} must be held at not lower than the reset release voltage (V_{LVD}) that is specified as LVD.

3.2.2 Copying Original Project

Refer to “2.2 Copying Original Project” and create a porting project.

3.2.3 Device Change for Porting project

Refer to “2.3.2 Porting project for e2 studio” and change the target device for the porting project.

3.2.4 Code Generator Configuration for Porting Project

In order to replace the code related to the peripheral function settings with the code for the porting device, configure the peripheral functions in Code Generator of the porting project according to the Code Generator settings in the original project.

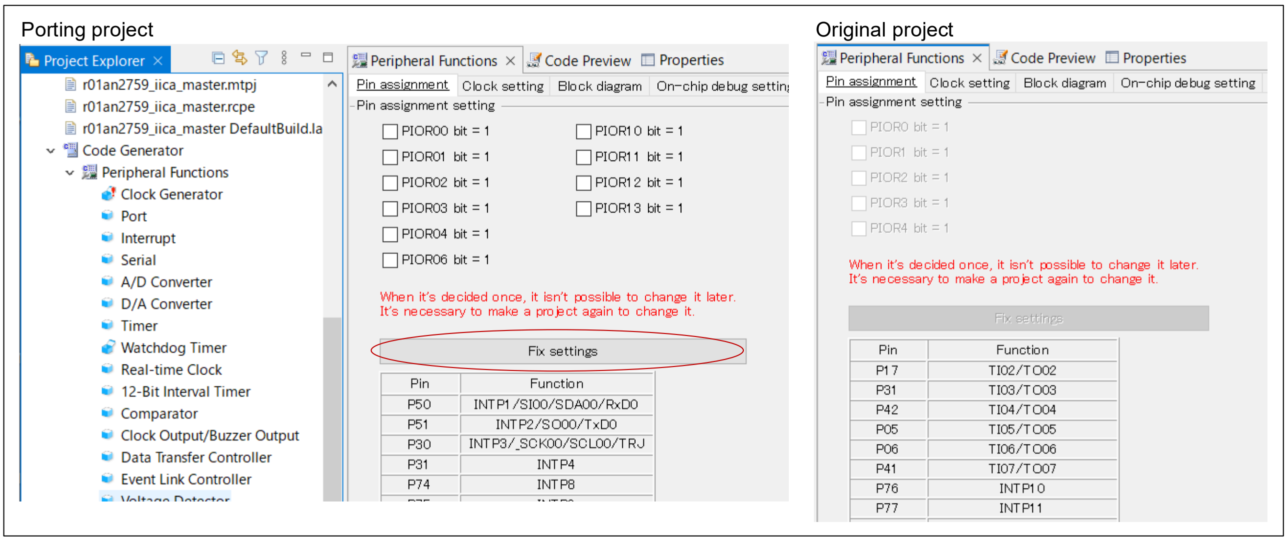
Start another e2 studio for the original project side by side and configure the peripheral functions for the porting project.

(1) Clock Generator

- Pin assignment

The peripheral I/O redirect function is not used in this porting example, click [Fix settings] without changing any settings.

Figure 3-18 Pin assignment (Porting Example 2)



Porting project

Pin assignment setting

☐ PIOR00 bit = 1
☐ PIOR01 bit = 1
☐ PIOR02 bit = 1
☐ PIOR03 bit = 1
☐ PIOR04 bit = 1
☐ PIOR06 bit = 1
☐ PIOR10 bit = 1
☐ PIOR11 bit = 1
☐ PIOR12 bit = 1
☐ PIOR13 bit = 1

When it’s decided once, it isn’t possible to change it later.
It’s necessary to make a project again to change it.

Pin	Function
P50	INTP1/SI00/SDA00/RxD0
P51	INTP2/SO00/TxD0
P30	INTP3/_SCK00/SCL00/TRJ
P31	INTP4
P74	INTP8

Original project

Pin assignment setting

☐ PIOR0 bit = 1
☐ PIOR1 bit = 1
☐ PIOR2 bit = 1
☐ PIOR3 bit = 1
☐ PIOR4 bit = 1

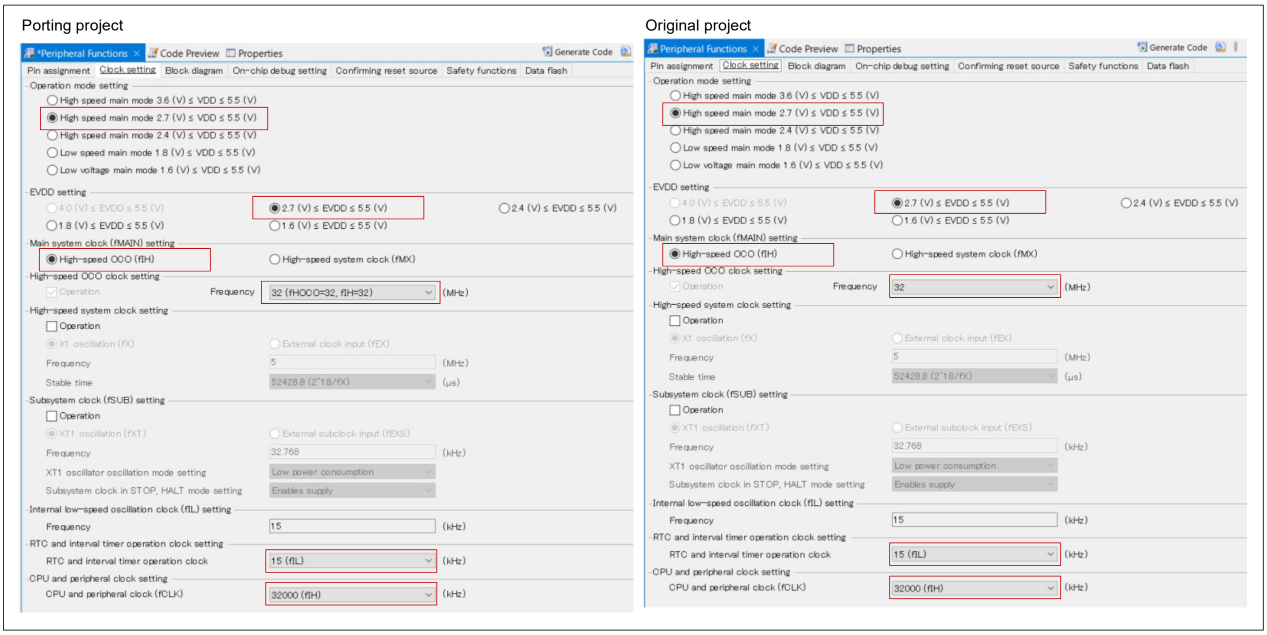
When it’s decided once, it isn’t possible to change it later.
It’s necessary to make a project again to change it.

Pin	Function
P17	TI02/TO02
P31	TI03/TO03
P42	TI04/TO04
P05	TI05/TO05
P06	TI06/TO06
P41	TI07/TO07
P76	INTP10
P77	INTP11

- Clock setting

Configure the items with the same settings as the original project.

Figure 3-19 Clock setting (Porting Example 1)



Porting project

Operation mode setting
○ High speed main mode 3.6 (V) ≤ VDD ≤ 5.5 (V)
● High speed main mode 2.7 (V) ≤ VDD ≤ 5.5 (V)
○ High speed main mode 2.4 (V) ≤ VDD ≤ 5.5 (V)
○ Low speed main mode 1.8 (V) ≤ VDD ≤ 5.5 (V)
○ Low voltage main mode 1.6 (V) ≤ VDD ≤ 5.5 (V)

EVDD setting
○ 4.0 (V) ≤ EVDD ≤ 5.5 (V) ● 2.7 (V) ≤ EVDD ≤ 5.5 (V) ○ 2.4 (V) ≤ EVDD ≤ 5.5 (V)
○ 1.8 (V) ≤ EVDD ≤ 5.5 (V) ○ 1.6 (V) ≤ EVDD ≤ 5.5 (V)

Main system clock (fMAIN) setting
● High-speed OCO (fIH) ○ High-speed system clock (fMX)

High-speed OCO clock setting
☑ Operation Frequency 32 (fHOCO=32, fIH=32) (MHz)

High-speed system clock setting
□ Operation
● X1 oscillation (fX) ○ External clock input (fEX)
Frequency 5 (MHz)
Stable time 52428.8 (2^18/fX) (μs)

Subsystem clock (fSUB) setting
□ Operation
● XT1 oscillation (fXT) ○ External subclock input (fEXS)
Frequency 32.768 (kHz)
XT1 oscillator oscillation mode setting Low power consumption
Subsystem clock in STOP, HALT mode setting Enables supply

Internal low-speed oscillation clock (fIL) setting
Frequency 15 (kHz)

RTC and interval timer operation clock setting
RTC and interval timer operation clock 15 (fIL) (kHz)

CPU and peripheral clock setting
CPU and peripheral clock (fCLK) 32000 (fIH) (kHz)

Original project

Operation mode setting
○ High speed main mode 3.6 (V) ≤ VDD ≤ 5.5 (V)
● High speed main mode 2.7 (V) ≤ VDD ≤ 5.5 (V)
○ High speed main mode 2.4 (V) ≤ VDD ≤ 5.5 (V)
○ Low speed main mode 1.8 (V) ≤ VDD ≤ 5.5 (V)
○ Low voltage main mode 1.6 (V) ≤ VDD ≤ 5.5 (V)

EVDD setting
○ 4.0 (V) ≤ EVDD ≤ 5.5 (V) ● 2.7 (V) ≤ EVDD ≤ 5.5 (V) ○ 2.4 (V) ≤ EVDD ≤ 5.5 (V)
○ 1.8 (V) ≤ EVDD ≤ 5.5 (V) ○ 1.6 (V) ≤ EVDD ≤ 5.5 (V)

Main system clock (fMAIN) setting
● High-speed OCO (fIH) ○ High-speed system clock (fMX)

High-speed OCO clock setting
☑ Operation Frequency 32 (MHz)

High-speed system clock setting
□ Operation
● X1 oscillation (fX) ○ External clock input (fEX)
Frequency 5 (MHz)
Stable time 52428.8 (2^18/fX) (μs)

Subsystem clock (fSUB) setting
□ Operation
● XT1 oscillation (fXT) ○ External subclock input (fEXS)
Frequency 32.768 (kHz)
XT1 oscillator oscillation mode setting Low power consumption
Subsystem clock in STOP, HALT mode setting Enables supply

Internal low-speed oscillation clock (fIL) setting
Frequency 15 (kHz)

RTC and interval timer operation clock setting
RTC and interval timer operation clock 15 (fIL) (kHz)

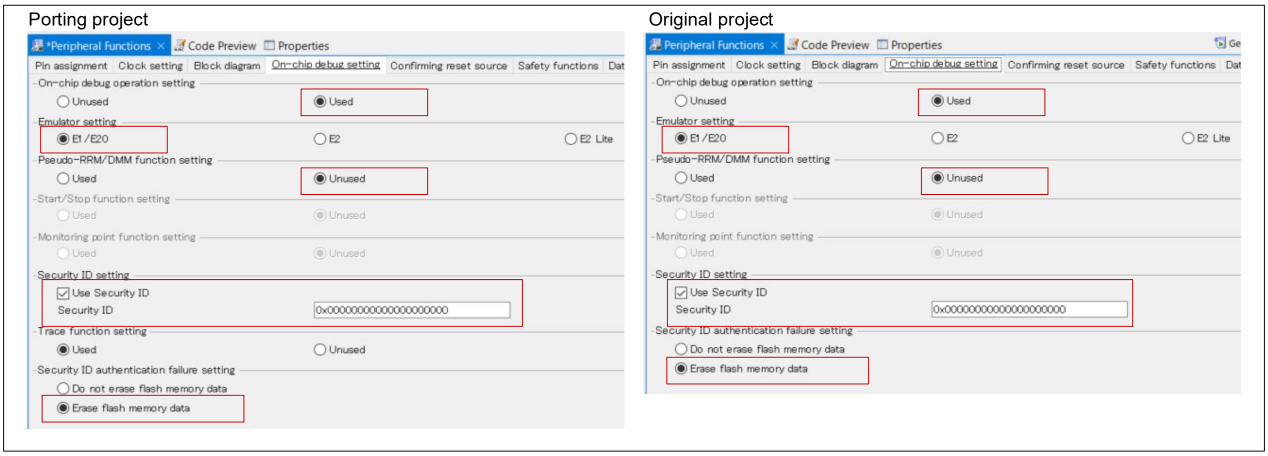
CPU and peripheral clock setting
CPU and peripheral clock (fCLK) 32000 (fIH) (kHz)

- On-chip debug setting

Configure the items with the same settings as the original project. Leave the default settings for items that only exist in the porting project.

Remark. The emulator setting can be changed from the original project according to the emulator you are using.

Figure 3-20 On-chip debug setting (Porting Example 2)



Porting project

On-chip debug operation setting
○ Unused ● Used

Emulator setting
● E1/E20 ○ E2 ○ E2 Lite

Pseudo-RRM/DMM function setting
○ Used ● Unused

Start/Stop function setting
○ Used ● Unused

Monitoring point function setting
○ Used ● Unused

Security ID setting
☑ Use Security ID
Security ID 0x00000000000000000000

Trace function setting
● Used ○ Unused

Security ID authentication failure setting
○ Do not erase flash memory data
● Erase flash memory data

Original project

On-chip debug operation setting
○ Unused ● Used

Emulator setting
● E1/E20 ○ E2 ○ E2 Lite

Pseudo-RRM/DMM function setting
○ Used ● Unused

Start/Stop function setting
○ Used ● Unused

Monitoring point function setting
○ Used ● Unused

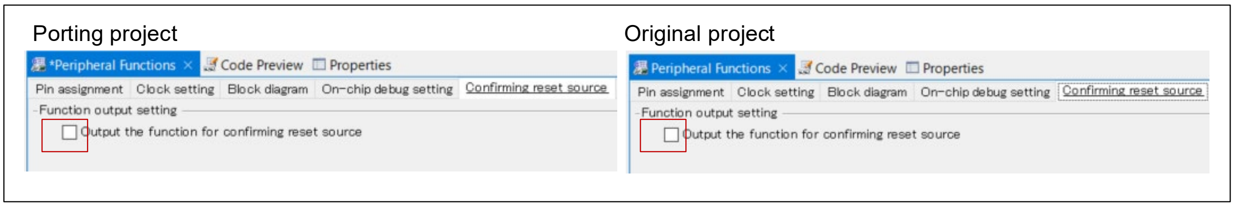
Security ID setting
☑ Use Security ID
Security ID 0x00000000000000000000

Security ID authentication failure setting
○ Do not erase flash memory data
● Erase flash memory data

- Confirming reset source

Configure the item with the same setting as the original project.

Figure 3-21 Confirming reset source (Porting Example 2)



Porting project

Function output setting

☐ Output the function for confirming reset source

Original project

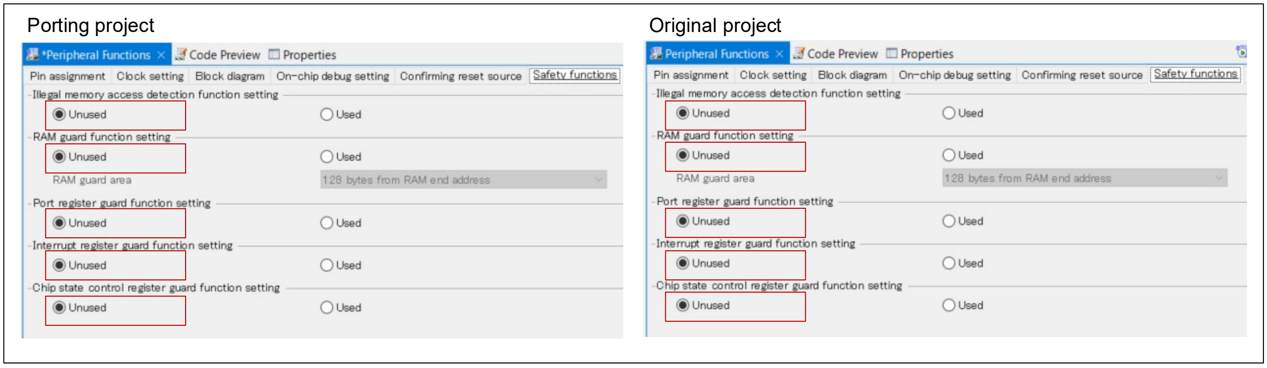
Function output setting

☐ Output the function for confirming reset source

- Safety functions

Configure the items with the same settings as the original project.

Figure 3-22 Safety functions (Porting Example 2)



Porting project

Illegal memory access detection function setting

● Unused ○ Used

RAM guard function setting

● Unused ○ Used

RAM guard area 128 bytes from RAM end address

Port register guard function setting

● Unused ○ Used

Interrupt register guard function setting

● Unused ○ Used

Chip state control register guard function setting

● Unused ○ Used

Original project

Illegal memory access detection function setting

● Unused ○ Used

RAM guard function setting

● Unused ○ Used

RAM guard area 128 bytes from RAM end address

Port register guard function setting

● Unused ○ Used

Interrupt register guard function setting

● Unused ○ Used

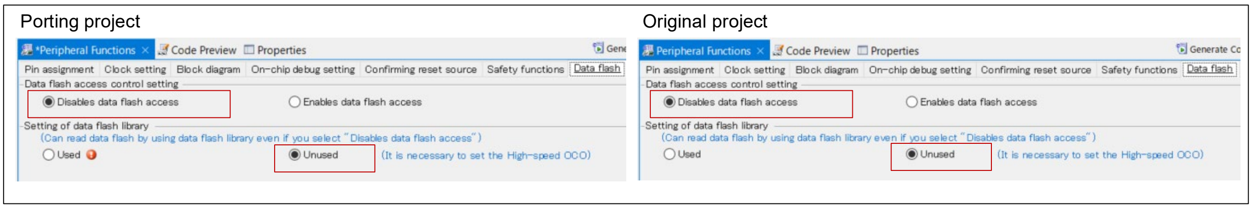
Chip state control register guard function setting

● Unused ○ Used

- Data flash

Configure the items with the same settings as the original project.

Figure 3-23 Data flash (Porting Example 2)



Porting project

Data flash access control setting

● Disables data flash access ○ Enables data flash access

Setting of data flash library

(Can read data flash by using data flash library even if you select "Disables data flash access")

○ Used ● Unused (It is necessary to set the High-speed OCO)

Original project

Data flash access control setting

● Disables data flash access ○ Enables data flash access

Setting of data flash library

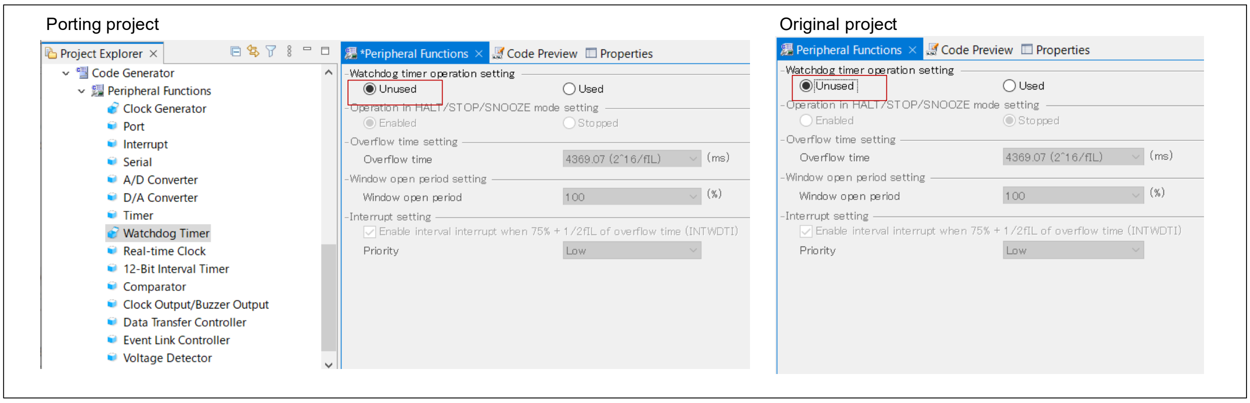
(Can read data flash by using data flash library even if you select "Disables data flash access")

○ Used ● Unused (It is necessary to set the High-speed OCO)

(2) Watchdog Timer

Configure the items with the same settings as the original project.

Figure 3-24 Watchdog Timer (Porting Example 2)



Porting project

- Code Generator
  - Peripheral Functions
    - Clock Generator
    - Port
    - Interrupt
    - Serial
    - A/D Converter
    - D/A Converter
    - Timer
    - Watchdog Timer
    - Real-time Clock
    - 12-Bit Interval Timer
    - Comparator
    - Clock Output/Buzzer Output
    - Data Transfer Controller
    - Event Link Controller
    - Voltage Detector

Watchdog timer operation setting
● Unused ○ Used

Operation in HALT/STOP/SNOOZE mode setting
● Enabled ○ Stopped

Overflow time setting
Overflow time 4369.07 (2^16/fIL) (ms)

Window open period setting
Window open period 100 (%)

Interrupt setting
☑ Enable interval interrupt when 75% + 1/2fIL of overflow time (INTWDTI)
Priority Low

Original project

Watchdog timer operation setting
● Unused ○ Used

Operation in HALT/STOP/SNOOZE mode setting
○ Enabled ● Stopped

Overflow time setting
Overflow time 4369.07 (2^16/fIL) (ms)

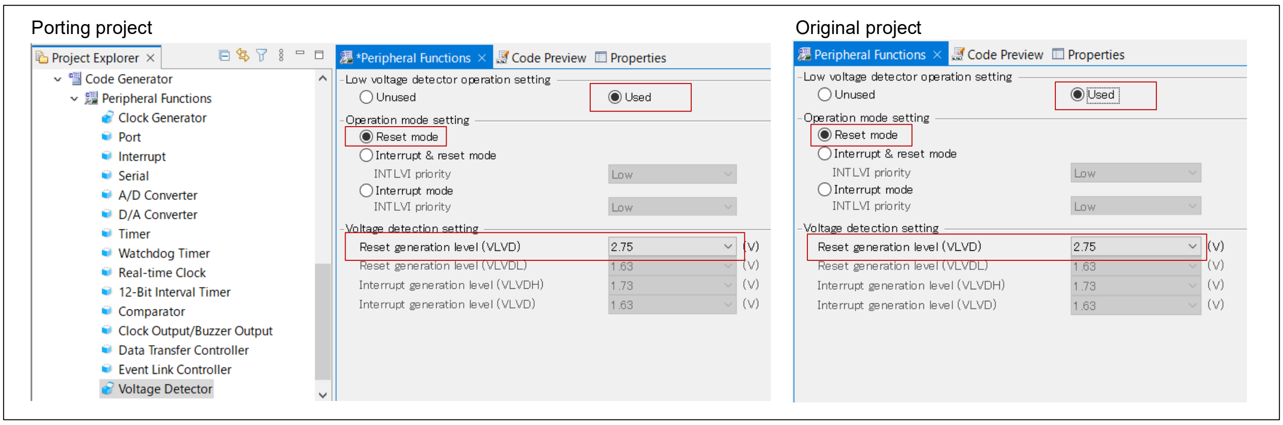
Window open period setting
Window open period 100 (%)

Interrupt setting
☑ Enable interval interrupt when 75% + 1/2fIL of overflow time (INTWDTI)
Priority Low

(3) Voltage Detector

Configure the items with the same settings as the original project.

Figure 3-25 Voltage Detector (Porting Example 2)



Porting project

- Code Generator
  - Peripheral Functions
    - Clock Generator
    - Port
    - Interrupt
    - Serial
    - A/D Converter
    - D/A Converter
    - Timer
    - Watchdog Timer
    - Real-time Clock
    - 12-Bit Interval Timer
    - Comparator
    - Clock Output/Buzzer Output
    - Data Transfer Controller
    - Event Link Controller
    - Voltage Detector

Low voltage detector operation setting
○ Unused ● Used

Operation mode setting
● Reset mode
○ Interrupt & reset mode
INTLVI priority Low
○ Interrupt mode
INTLVI priority Low

Voltage detection setting
Reset generation level (VLVD) 2.75 (V)
Reset generation level (VLVDL) 1.63 (V)
Interrupt generation level (VLVDH) 1.73 (V)
Interrupt generation level (VLVD) 1.63 (V)

Original project

Low voltage detector operation setting
○ Unused ● Used

Operation mode setting
● Reset mode
○ Interrupt & reset mode
INTLVI priority Low
○ Interrupt mode
INTLVI priority Low

Voltage detection setting
Reset generation level (VLVD) 2.75 (V)
Reset generation level (VLVDL) 1.63 (V)
Interrupt generation level (VLVDH) 1.73 (V)
Interrupt generation level (VLVD) 1.63 (V)

(4) Peripheral function used in sample code

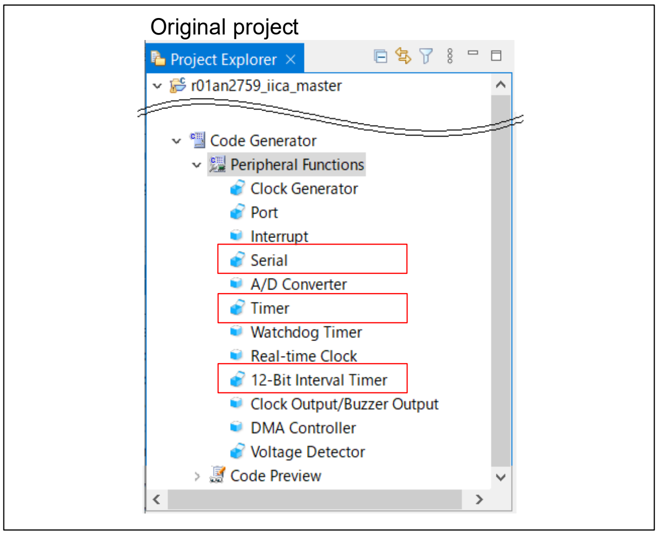
Check icons under the [Code Generator (Design Tool)] node of the original project and configure the peripheral functions for the porting project to match the configuration for the original project.

“Serial”, “Timer” and “12-Bit Interval Timer” are used in the original project (RL78/G13 AN). Note that Port is configured in Step (5).

: The peripheral function has any setting. (Used)

: The peripheral function has no setting. (Unused)

Figure 3-26 Peripheral Function Has Settings (Porting Example 2)



Original project

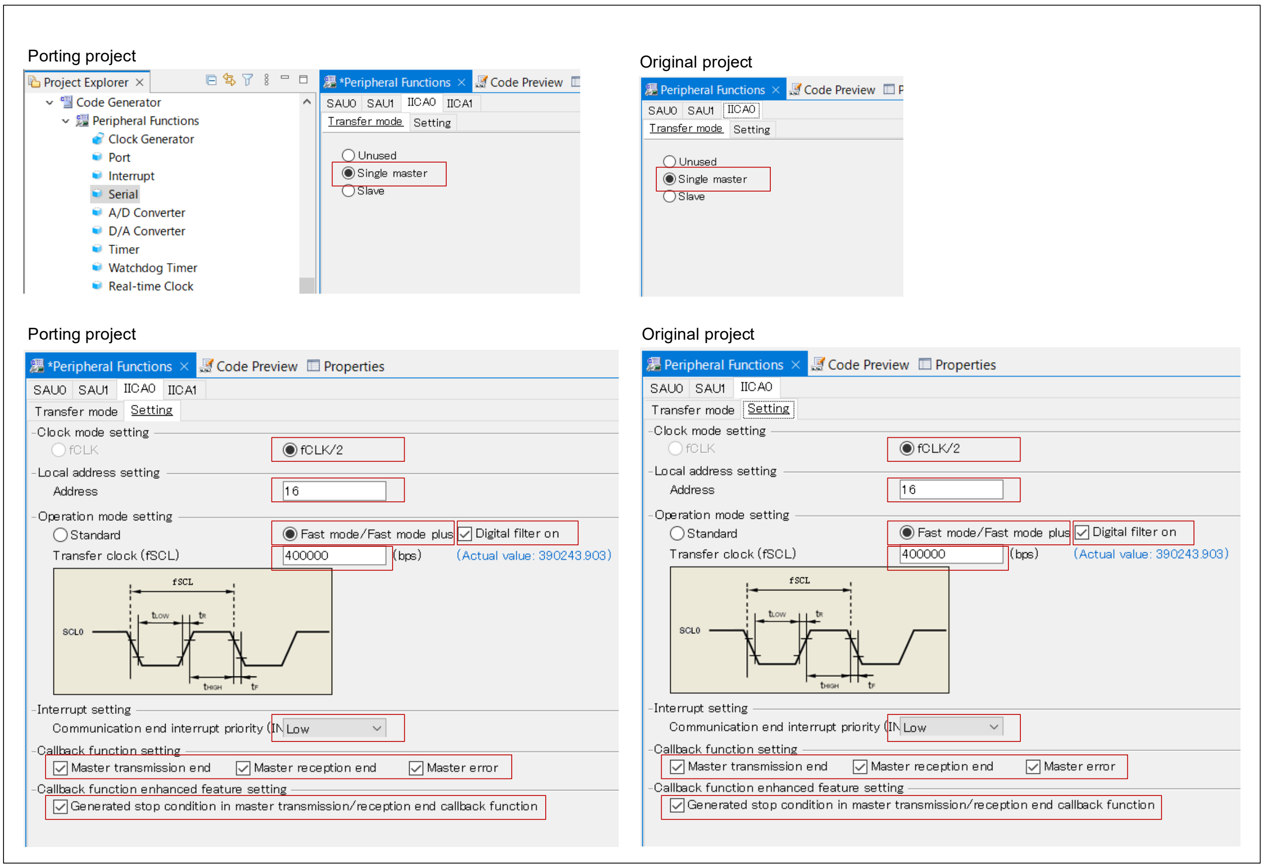
Project Explorer ×

- r01an2759_iica_master
  - Code Generator
    - Peripheral Functions
      - Clock Generator
      - Port
      - Interrupt
      - Serial
      - A/D Converter
      - Timer
      - Watchdog Timer
      - Real-time Clock
      - 12-Bit Interval Timer
      - Clock Output/Buzzer Output
      - DMA Controller
      - Voltage Detector
    - Code Preview

- Serial

Configure the items with the same settings as the original project.

Figure 3-27 Serial - IICA0 (Porting Example 2)



Porting project

Project Explorer ×

- Code Generator
  - Peripheral Functions
    - Clock Generator
    - Port
    - Interrupt
    - Serial
    - A/D Converter
    - D/A Converter
    - Timer
    - Watchdog Timer
    - Real-time Clock

*Peripheral Functions × | Code Preview

SAU0 | SAU1 | IICA0 | IICA1

Transfer mode | Setting

○ Unused
● Single master
○ Slave

Original project

Peripheral Functions × | Code Preview

SAU0 | SAU1 | IICA0

Transfer mode | Setting

○ Unused
● Single master
○ Slave

Porting project

*Peripheral Functions × | Code Preview | Properties

SAU0 | SAU1 | IICA0 | IICA1

Transfer mode | Setting

Clock mode setting
○ fCLK ● fCLK/2

Local address setting
Address 16

Operation mode setting
○ Standard ● Fast mode/Fast mode plus ☑ Digital filter on
Transfer clock (fSCL) 400000 (bps) (Actual value: 390243.903)

Interrupt setting
Communication end interrupt priority (IN Low

Callback function setting
☑ Master transmission end ☑ Master reception end ☑ Master error

Callback function enhanced feature setting
☑ Generated stop condition in master transmission/reception end callback function

Original project

Peripheral Functions × | Code Preview | Properties

SAU0 | SAU1 | IICA0

Transfer mode | Setting

Clock mode setting
○ fCLK ● fCLK/2

Local address setting
Address 16

Operation mode setting
○ Standard ● Fast mode/Fast mode plus ☑ Digital filter on
Transfer clock (fSCL) 400000 (bps) (Actual value: 390243.903)

Interrupt setting
Communication end interrupt priority (IN Low

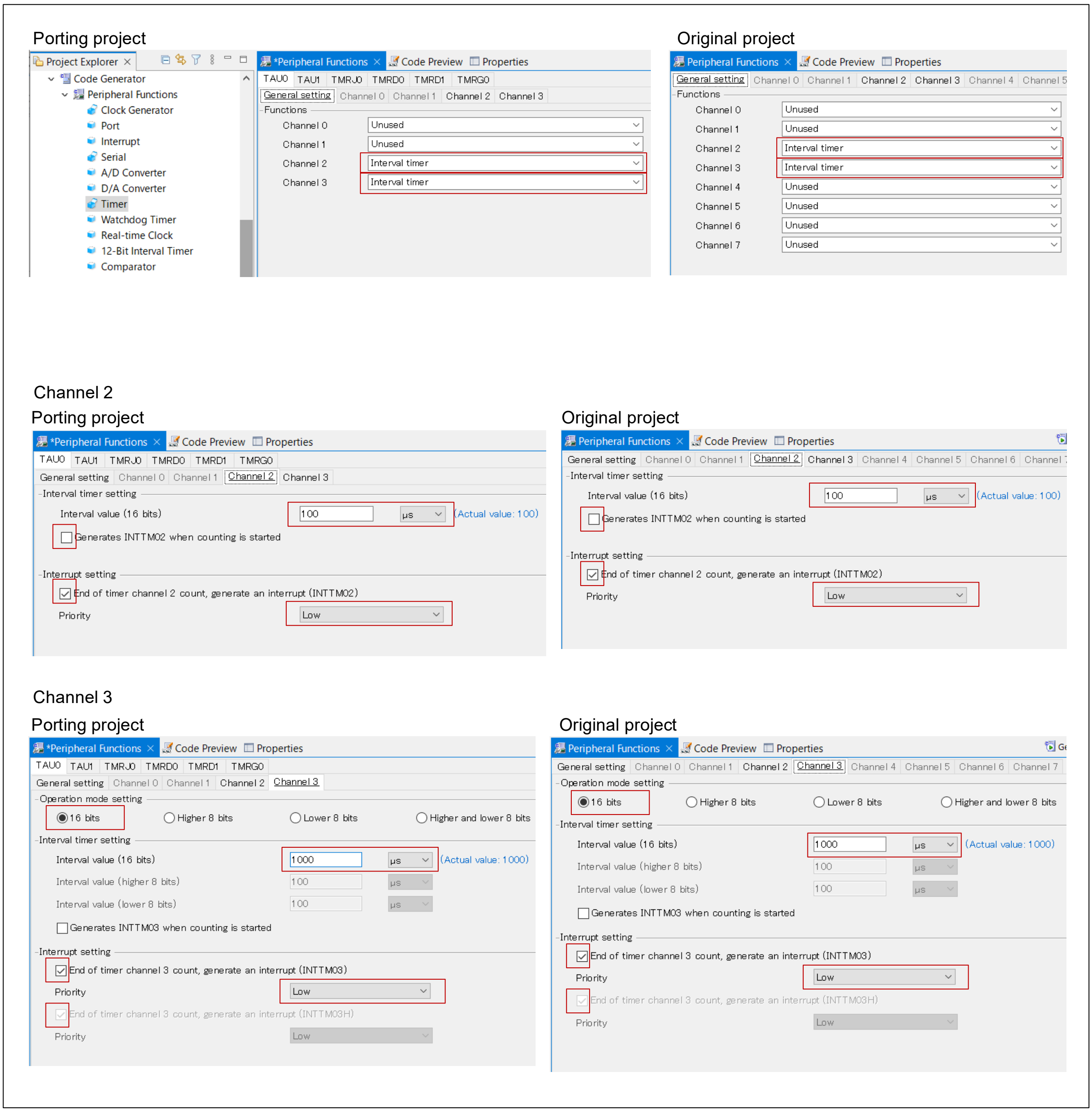
Callback function setting
☑ Master transmission end ☑ Master reception end ☑ Master error

Callback function enhanced feature setting
☑ Generated stop condition in master transmission/reception end callback function

- Timer

Configure the items with the same settings as the original project.

Figure 3-28 Timer – channel 2, channel3 (Porting Example 2)



Porting project

Functions
Channel 0: Unused
Channel 1: Unused
Channel 2: Interval timer
Channel 3: Interval timer

Original project

Functions
Channel 0: Unused
Channel 1: Unused
Channel 2: Interval timer
Channel 3: Interval timer
Channel 4: Unused
Channel 5: Unused
Channel 6: Unused
Channel 7: Unused

Channel 2

Porting project

Interval timer setting
Interval value (16 bits) 100 μs (Actual value: 100)
□ Generates INTTM02 when counting is started
Interrupt setting
☑ End of timer channel 2 count, generate an interrupt (INTTM02)
Priority: Low

Original project

Interval timer setting
Interval value (16 bits) 100 μs (Actual value: 100)
□ Generates INTTM02 when counting is started
Interrupt setting
☑ End of timer channel 2 count, generate an interrupt (INTTM02)
Priority: Low

Channel 3

Porting project

Operation mode setting
● 16 bits ○ Higher 8 bits ○ Lower 8 bits ○ Higher and lower 8 bits
Interval timer setting
Interval value (16 bits) 1000 μs (Actual value: 1000)
Interval value (higher 8 bits) 100 μs
Interval value (lower 8 bits) 100 μs
□ Generates INTTM03 when counting is started
Interrupt setting
☑ End of timer channel 3 count, generate an interrupt (INTTM03)
Priority: Low
☑ End of timer channel 3 count, generate an interrupt (INTTM03H)
Priority: Low

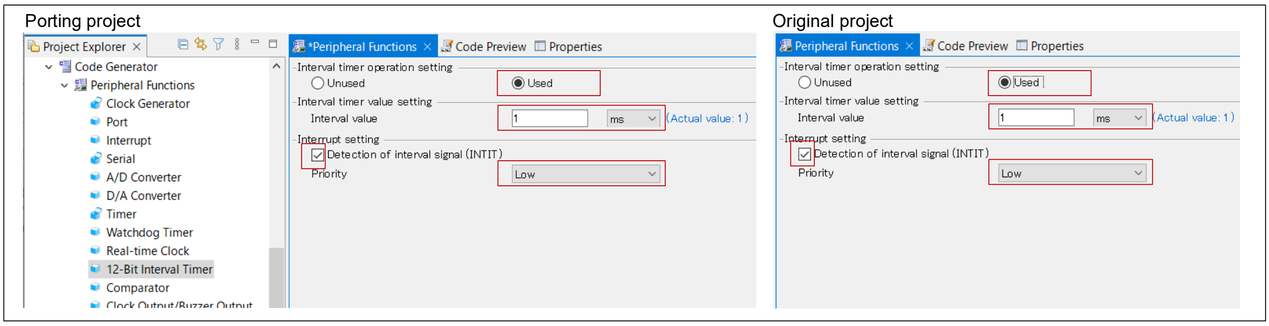
Original project

Operation mode setting
● 16 bits ○ Higher 8 bits ○ Lower 8 bits ○ Higher and lower 8 bits
Interval timer setting
Interval value (16 bits) 1000 μs (Actual value: 1000)
Interval value (higher 8 bits) 100 μs
Interval value (lower 8 bits) 100 μs
□ Generates INTTM03 when counting is started
Interrupt setting
☑ End of timer channel 3 count, generate an interrupt (INTTM03)
Priority: Low
☑ End of timer channel 3 count, generate an interrupt (INTTM03H)
Priority: Low

- 12-Bit Interval Timer

Configure the items with the same settings as the original project.

- Figure 3-29 12-Bit Interval Timer (Porting Example 2)



Porting project

Project Explorer: Code Generator > Peripheral Functions > Clock Generator, Port, Interrupt, Serial, A/D Converter, D/A Converter, Timer, Watchdog Timer, Real-time Clock, 12-Bit Interval Timer, Comparator, Clock Output/Buzzer Output

*Peripheral Functions | Code Preview | Properties

Interval timer operation setting: ○ Unused ● Used

Interval timer value setting: Interval value 1 ms (Actual value: 1)

Interrupt setting: ☑ Detection of interval signal (INTIT)

Priority: Low

Original project

Peripheral Functions | Code Preview | Properties

Interval timer operation setting: ○ Unused ● Used

Interval timer value setting: Interval value 1 ms (Actual value: 1)

Interrupt setting: ☑ Detection of interval signal (INTIT)

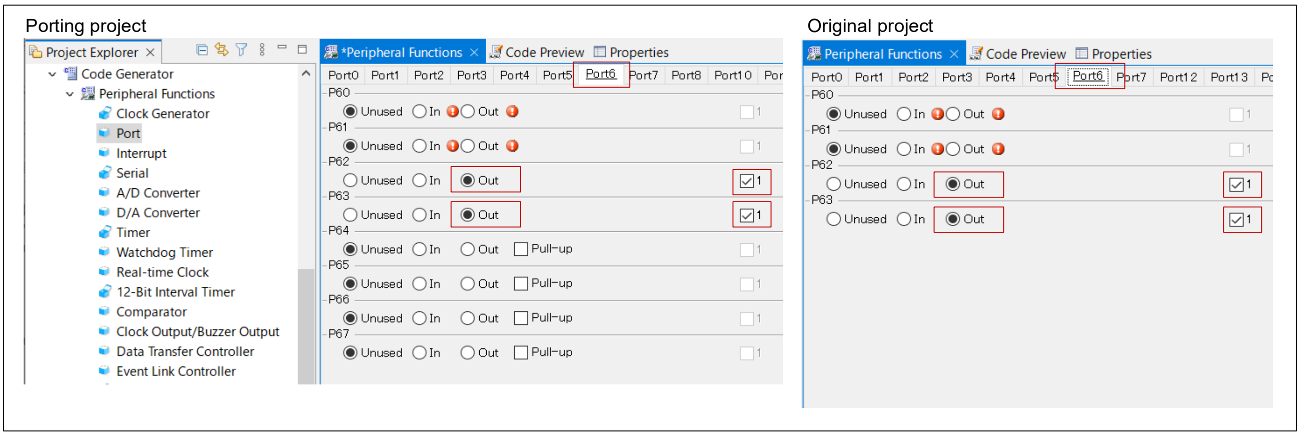
Priority: Low

(5) Port

Configure the items with the same settings as the original project.

Caution. Provide proper treatment for unused pins so that their electrical specifications are observed. Connect each of any unused input-only ports to VDD or VSS via a separate resistor.

Figure 3-30 Port (Porting Example 2)



Porting project

Project Explorer: Code Generator > Peripheral Functions > Clock Generator, Port, Interrupt, Serial, A/D Converter, D/A Converter, Timer, Watchdog Timer, Real-time Clock, 12-Bit Interval Timer, Comparator, Clock Output/Buzzer Output, Data Transfer Controller, Event Link Controller

*Peripheral Functions | Code Preview | Properties

Port0 | Port1 | Port2 | Port3 | Port4 | Port5 | Port6 | Port7 | Port8 | Port10 | Por

P60: ● Unused ○ In ○ Out □ 1

P61: ● Unused ○ In ○ Out □ 1

P62: ○ Unused ○ In ● Out ☑ 1

P63: ○ Unused ○ In ● Out ☑ 1

P64: ● Unused ○ In ○ Out □ Pull-up □ 1

P65: ● Unused ○ In ○ Out □ Pull-up □ 1

P66: ● Unused ○ In ○ Out □ Pull-up □ 1

P67: ● Unused ○ In ○ Out □ Pull-up □ 1

Original project

Peripheral Functions | Code Preview | Properties

Port0 | Port1 | Port2 | Port3 | Port4 | Port5 | Port6 | Port7 | Port12 | Port13 | Po

P60: ● Unused ○ In ○ Out □ 1

P61: ● Unused ○ In ○ Out □ 1

P62: ○ Unused ○ In ● Out ☑ 1

P63: ○ Unused ○ In ● Out ☑ 1

(6) Saving configuration

Save the configuration for the porting project. Select the [File] – [Save All] menu of e^2 studio.

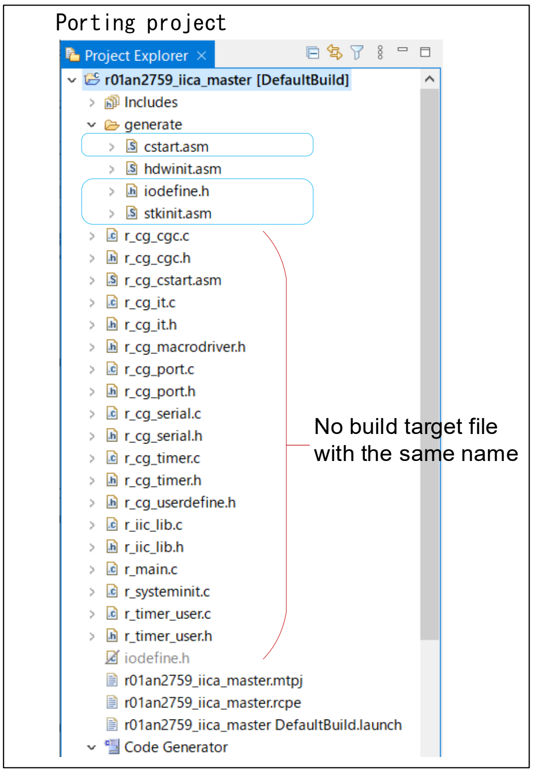
3.2.5 Code Generation for Porting Project

Before generating the code, exclude the file with the same name as the file in the “generate” folder from the build target. In addition, move the Code Generator (CG) generated file to the “src” folder.
After that, generate the code for the porting project according to the setting of Code Generator (CG)

(1) Checking the file with the same name as the file in the “generate” folder

Of the build target files inherited from the original project, there is no file with the same name as the file in the “generate” folder. Therefore, there is no file to be excluded from the build target.

Figure 3-31 Checking File with Same Name (Porting Example 2)



Porting project

Project Explorer

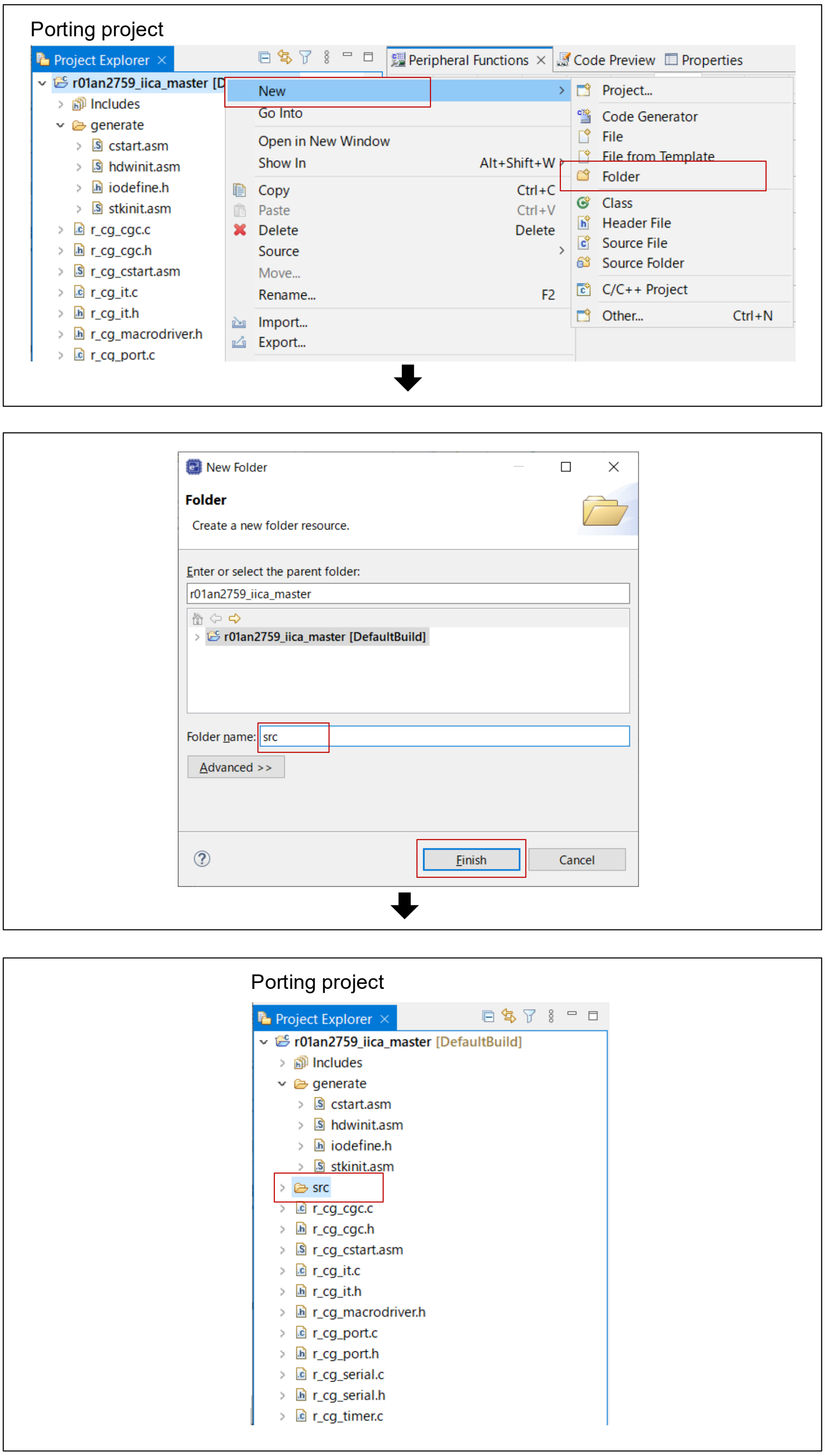
- r01an2759_iica_master [DefaultBuild]
  - Includes
  - generate
    - cstart.asm
    - hdwinit.asm
    - iodefine.h
    - stkinit.asm
  - r_cg_cgc.c
  - r_cg_cgc.h
  - r_cg_cstart.asm
  - r_cg_it.c
  - r_cg_it.h
  - r_cg_macrodriver.h
  - r_cg_port.c
  - r_cg_port.h
  - r_cg_serial.c
  - r_cg_serial.h
  - r_cg_timer.c
  - r_cg_timer.h
  - r_cg_userdefine.h
  - r_iic_lib.c
  - r_iic_lib.h
  - r_main.c
  - r_systeminit.c
  - r_timer_user.c
  - r_timer_user.h
  - iodefine.h
  - r01an2759_iica_master.mtpj
  - r01an2759_iica_master.rcpe
  - r01an2759_iica_master DefaultBuild.launch
  - Code Generator

No build target file with the same name

(2) Moving CG generated file to “src” folder

Check whether files generated by Code Generator (CG) shown in “Table 1-3 Outline of CG generated files and functions” are registered in the “src” folder.
Create the “src” folder newly because the “src” folder does not exist in the original project for this porting example. Right click the project name and click the [New] – [Folder] menu. Input “src” in [Folder name] and click [Finish] in the [New Folder] dialog box.
After creating the “src” folder, move the CG generated files to the “src” folder by drag & drop.

Figure 3-32 Creating “src” folder (Porting Example 2)



Porting project

New Folder

Folder

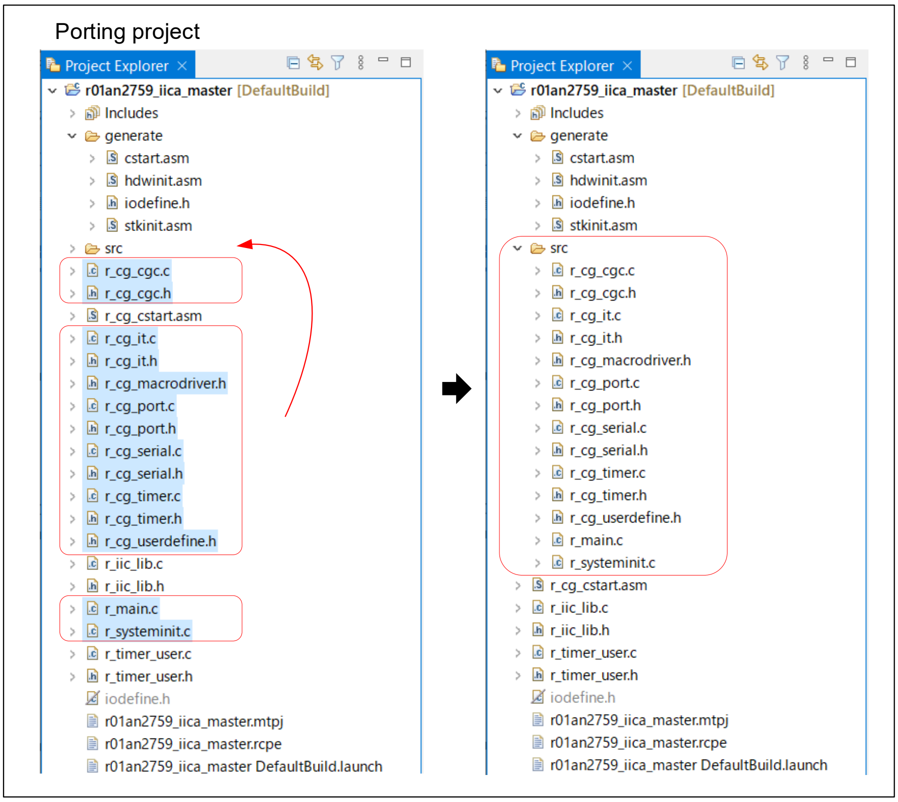
Create a new folder resource.

Enter or select the parent folder: r01an2759_iica_master

Folder name: src

Porting project

Figure 3-33 Moving CG generated Files to "src" folder (Porting Example 2)



Porting project

Before:

```
r01an2759_iica_master [DefaultBuild]
  Includes
  generate
    cstart.asm
    hdwinit.asm
    iodefine.h
    stkinit.asm
  src
  r_cg_cgc.c
  r_cg_cgc.h
  r_cg_cstart.asm
  r_cg_it.c
  r_cg_it.h
  r_cg_macrodriver.h
  r_cg_port.c
  r_cg_port.h
  r_cg_serial.c
  r_cg_serial.h
  r_cg_timer.c
  r_cg_timer.h
  r_cg_userdefine.h
  r_iic_lib.c
  r_iic_lib.h
  r_main.c
  r_systeminit.c
  r_timer_user.c
  r_timer_user.h
  iodefine.h
  r01an2759_iica_master.mtpj
  r01an2759_iica_master.rcpe
  r01an2759_iica_master DefaultBuild.launch
```

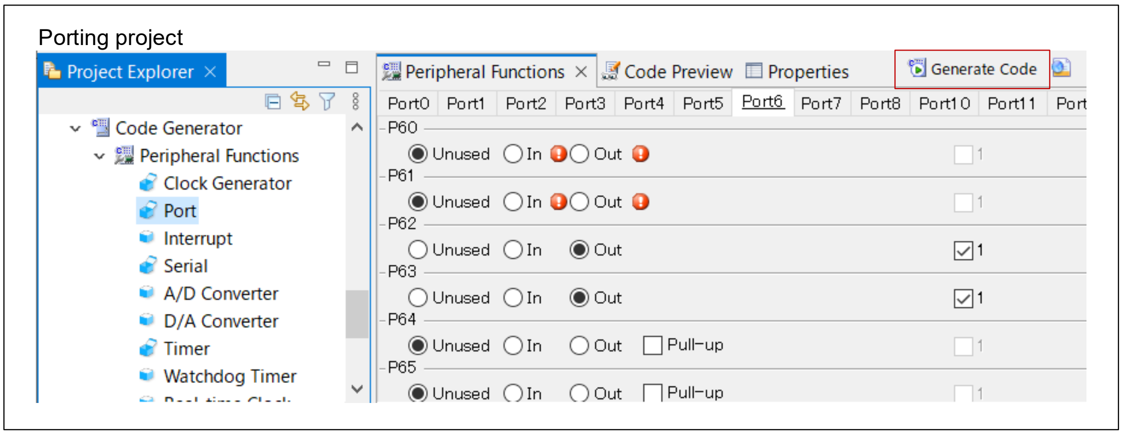
After:

```
r01an2759_iica_master [DefaultBuild]
  Includes
  generate
    cstart.asm
    hdwinit.asm
    iodefine.h
    stkinit.asm
  src
    r_cg_cgc.c
    r_cg_cgc.h
    r_cg_it.c
    r_cg_it.h
    r_cg_macrodriver.h
    r_cg_port.c
    r_cg_port.h
    r_cg_serial.c
    r_cg_serial.h
    r_cg_timer.c
    r_cg_timer.h
    r_cg_userdefine.h
    r_main.c
    r_systeminit.c
  r_cg_cstart.asm
  r_iic_lib.c
  r_iic_lib.h
  r_timer_user.c
  r_timer_user.h
  iodefine.h
  r01an2759_iica_master.mtpj
  r01an2759_iica_master.rcpe
  r01an2759_iica_master DefaultBuild.launch
```

(3) Generating code

Generate the code for the porting project according to the setting of Code Generator (CG). Click the [Generate Code] button.

Figure 3-34 Generating Code (Porting Example 2)



Porting project

P60: ● Unused ○ In ○ Out □ 1

P61: ● Unused ○ In ○ Out □ 1

P62: ○ Unused ○ In ● Out ☑ 1

P63: ○ Unused ○ In ● Out ☑ 1

P64: ● Unused ○ In ○ Out □ Pull-up □ 1

P65: ● Unused ○ In ○ Out □ Pull-up □ 1

3.2.6 Editing File in Porting Project

Check if there are any changes such as commenting the CG generated code in the original project and if there are any changes, make same changes to the porting project. Open the file with the same name of the original project and the ported project in the editor, check the difference in the code, and reflect the deletion / change of the original project in the ported project.

Figure 3-35 Files to be checked (Porting Example 2)

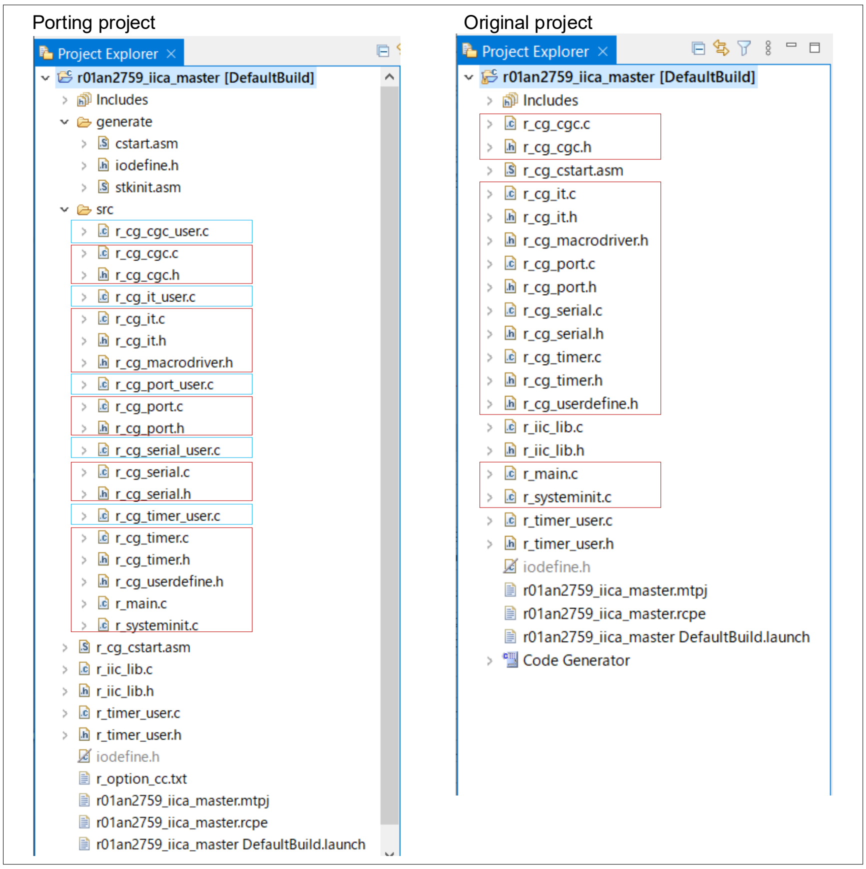


Table 3-8 Files to be Checked and Whether Editing is Required (Porting Example 2) (1/2)

Porting Project File Name	**Whether Editing is Required:** **✓: Required** **—: Not Required**	**Action to be Taken When Difference is Found**
r_cg_cgc_user.c	Exclude from build target	Although the file is generated by CG for the porting project, the file is excluded from build target because it is not used in the original project.
r_cg_cgc.c	—	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_cgc.h	—	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_it_user.c	Exclude from build target	Although the file is generated by CG for the porting project, the file is excluded from build target because it is not used in the original project.
r_cg_it.c	✓	Delete the functions that are deleted from the original project.

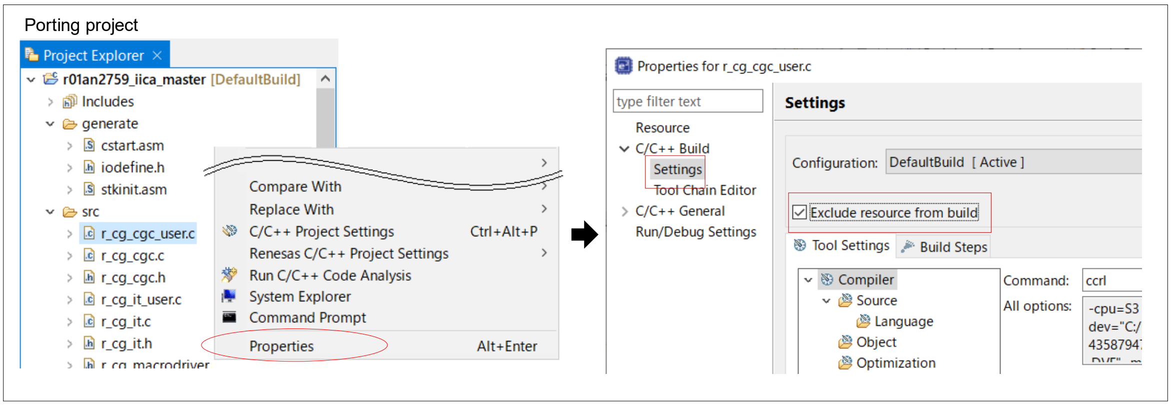
Table 3-9 Files to be Checked and Whether Editing is Required (Porting Example 2) (2/2)

Porting Project File Name	**Whether Editing is Required:** **✓: Required** **—: Not Required**	**Action to be Taken When Difference is Found**
r_cg_it.h	✓	Delete the function prototypes that are deleted from the original project.
r_cg_microdriver.h	—	No difference exists because no change (such as commenting out) was made in the original project.
r_cg_port_user.c	Exclude from build target	Although the file is generated by CG for the porting project, the file is excluded from build target because it is not used in the original project.
r_cg_port.c	—	Code difference in the function “R_PORT_Create” derives from the difference in device specifications between the source and target devices. No editing is necessary. Caution on unused ports: When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware’s electrical specifications are met (connect the input-only ports separately to VDD or VSS via a resistor).
r_cg_port.h	—	Difference in macro definitions derives from the difference in device specifications between the source and target devices. No editing is necessary.
r_cg_serial_user.c	Exclude from build target	Although the file is generated by CG for the porting project, the file is excluded from build target because it is not used in the original project.
r_cg_serial.c	✓	Delete the variables and functions that are deleted from the original project. In addition, edit generated code.
r_cg_serial.h	✓	Delete the function prototypes that are deleted from the original project. In addition, modify macro definitions.
r_cg_timer_user.c	Exclude from build target	Although the file is generated by CG for the porting project, the file is excluded from build target because it is not used in the original project.
r_cg_timer.c	✓	Code difference in the function “R_TAU0_Create” derives from the difference in device specifications between the source and target devices. No editing is necessary. Delete the functions that are deleted from the original project.
r_cg_timer.h	✓	Delete the function prototypes that are deleted from the original project. Difference in macro definitions derives from the difference in device specifications between the source and target devices. No editing is necessary.
r_cg_userdefine.h	—	No difference exists because no change (such as commenting out) was made in the original project.
r_main.c	—	No difference exists because no change (such as commenting out) was made in the original project.
r_systeminit.c	—	Code difference in the function “R_Systeminit” derives from the difference in device specifications between the source and target devices. No editing is necessary.

(a) r_cg_cgc_user.c、r_cg_it_user.c、r_cg_port_user.c、r_cg_serial_user.c、r_cg_timer_user.c

Right click the file and click [Properties]. Select [C/C++ Build] - [Settings] in the [Properties] dialog box. Check [Exclude source from build] and click [Apply and Close]. Do so for the above files.

Figure 3-36 r_cg_cgc_user.c (Porting Example 2)



Configuration: DefaultBuild [Active]

☑ Exclude resource from build

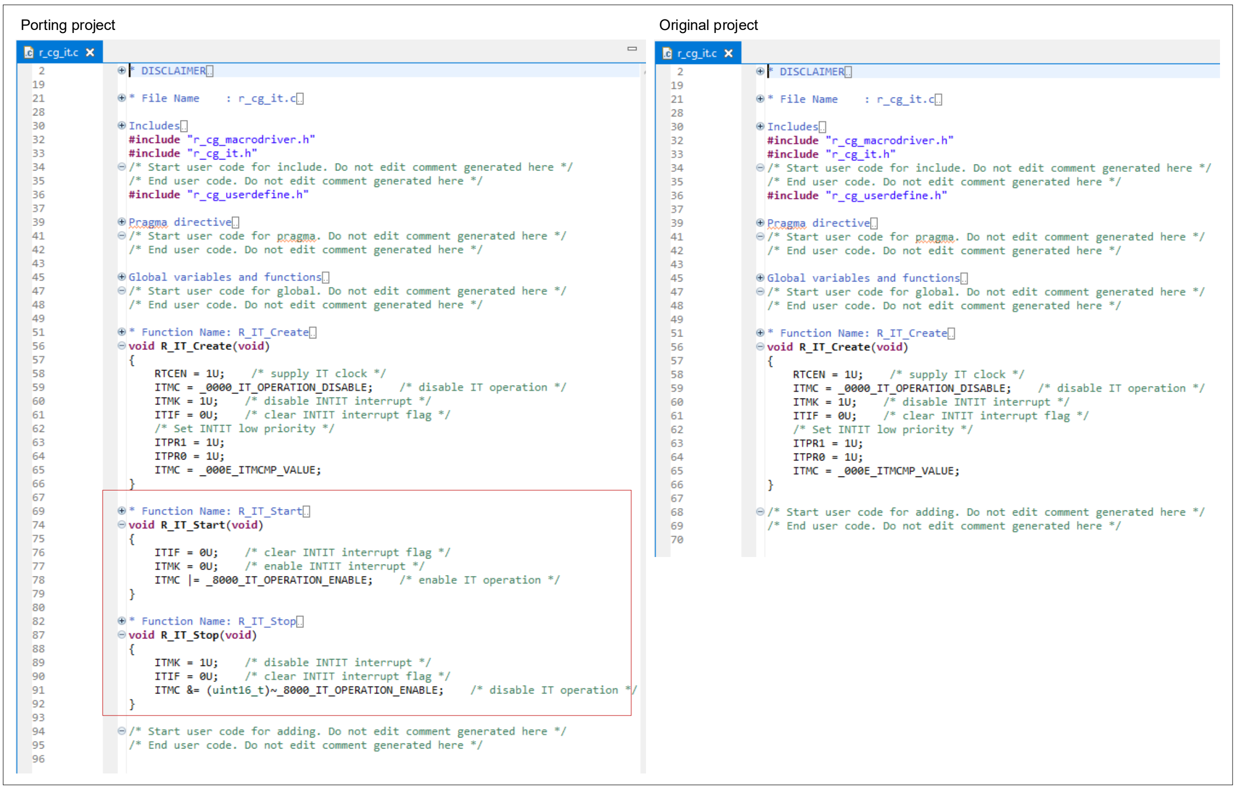
Command: ccrl

All options: -cpu=S3 dev="C:/ 4358794

(b) r_cg_it.c

Delete the function “R_IT_Start” and “R_IT_Stop”.

Figure 3-37 r_cg_it.c (Porting Example 2)



Porting project

r_cg_it.c

```
* DISCLAIMER

* File Name    : r_cg_it.c

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

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

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

* Function Name: R_IT_Create
void R_IT_Create(void)
{
    RTCEN = 1U;    /* supply IT clock */
    ITMC = _0000_IT_OPERATION_DISABLE;    /* disable IT operation */
    ITMK = 1U;    /* disable INTIT interrupt */
    ITIF = 0U;    /* clear INTIT interrupt flag */
    /* Set INTIT low priority */
    ITPR1 = 1U;
    ITPR0 = 1U;
    ITMC = _000E_ITMCMP_VALUE;
}

* Function Name: R_IT_Start
void R_IT_Start(void)
{
    ITIF = 0U;    /* clear INTIT interrupt flag */
    ITMK = 0U;    /* enable INTIT interrupt */
    ITMC |= _8000_IT_OPERATION_ENABLE;    /* enable IT operation */
}

* Function Name: R_IT_Stop
void R_IT_Stop(void)
{
    ITMK = 1U;    /* disable INTIT interrupt */
    ITIF = 0U;    /* clear INTIT interrupt flag */
    ITMC &= (uint16_t)~_8000_IT_OPERATION_ENABLE;    /* disable IT operation */
}

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

Original project

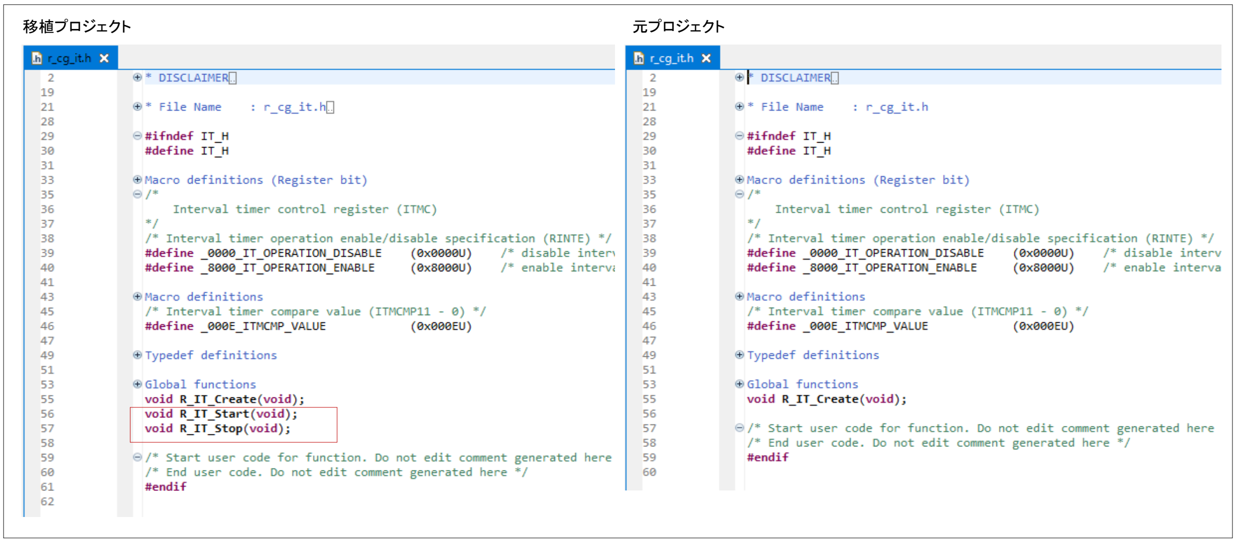
r_cg_it.c

```
* DISCLAIMER

* File Name    : r_cg_it.c

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

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

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

* Function Name: R_IT_Create
void R_IT_Create(void)
{
    RTCEN = 1U;    /* supply IT clock */
    ITMC = _0000_IT_OPERATION_DISABLE;    /* disable IT operation */
    ITMK = 1U;    /* disable INTIT interrupt */
    ITIF = 0U;    /* clear INTIT interrupt flag */
    /* Set INTIT low priority */
    ITPR1 = 1U;
    ITPR0 = 1U;
    ITMC = _000E_ITMCMP_VALUE;
}

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

(c) r_cg_it.h

Delete the prototype definition of the function “R_IT_Start” and “R_IT_Stop” have been deleted from r_cg_it.c.

Figure 3-38 r_cg_it.h (Porting Example 2)



移植プロジェクト

```
* DISCLAIMER

* File Name    : r_cg_it.h

#ifndef IT_H
#define IT_H

Macro definitions (Register bit)
/*
    Interval timer control register (ITMC)
*/
/* Interval timer operation enable/disable specification (RINTE) */
#define _0000_IT_OPERATION_DISABLE    (0x0000U)    /* disable interv
#define _8000_IT_OPERATION_ENABLE     (0x8000U)    /* enable interva

Macro definitions
/* Interval timer compare value (ITMCMP11 - 0) */
#define _000E_ITMCMP_VALUE            (0x000EU)

Typedef definitions

Global functions
void R_IT_Create(void);
void R_IT_Start(void);
void R_IT_Stop(void);

/* Start user code for function. Do not edit comment generated here
/* End user code. Do not edit comment generated here */
#endif
```

元プロジェクト

```
* DISCLAIMER

* File Name    : r_cg_it.h

#ifndef IT_H
#define IT_H

Macro definitions (Register bit)
/*
    Interval timer control register (ITMC)
*/
/* Interval timer operation enable/disable specification (RINTE) */
#define _0000_IT_OPERATION_DISABLE    (0x0000U)    /* disable interv
#define _8000_IT_OPERATION_ENABLE     (0x8000U)    /* enable interva

Macro definitions
/* Interval timer compare value (ITMCMP11 - 0) */
#define _000E_ITMCMP_VALUE            (0x000EU)

Typedef definitions

Global functions
void R_IT_Create(void);

/* Start user code for function. Do not edit comment generated here
/* End user code. Do not edit comment generated here */
#endif
```

(d) r_cg_serial.c

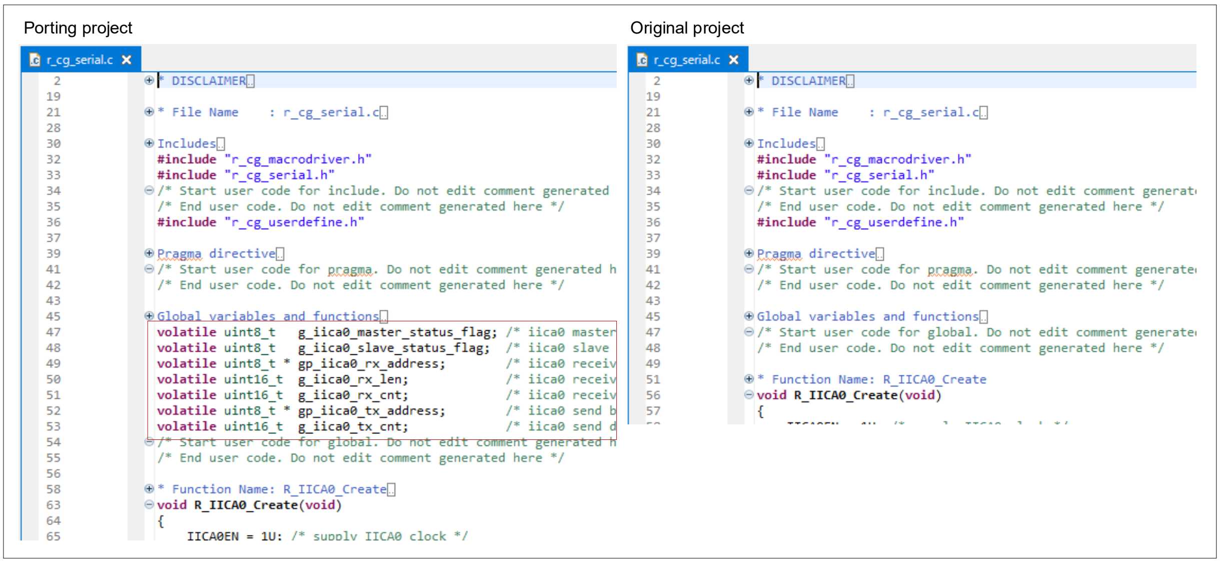
Delete all global variables. (Figure 3-39)

Delete the function “R_IICA0_Stop”, “R_IICA0_StopCondtition”, “R_IICA0_Master_Send” and “R_IICA0_Master_Receive”. (Figure 3-40)

Edit the code in the function “R_IICA0_Create” to match the original project. (Figure 3-41)

- The code for the SCLA0 and SDAA0 pin
- The setting value for the SVA0 register (The local address when in slave mode)

Figure 3-39 r_cg_serial.c – Deletion Valuables (Porting Example 2)



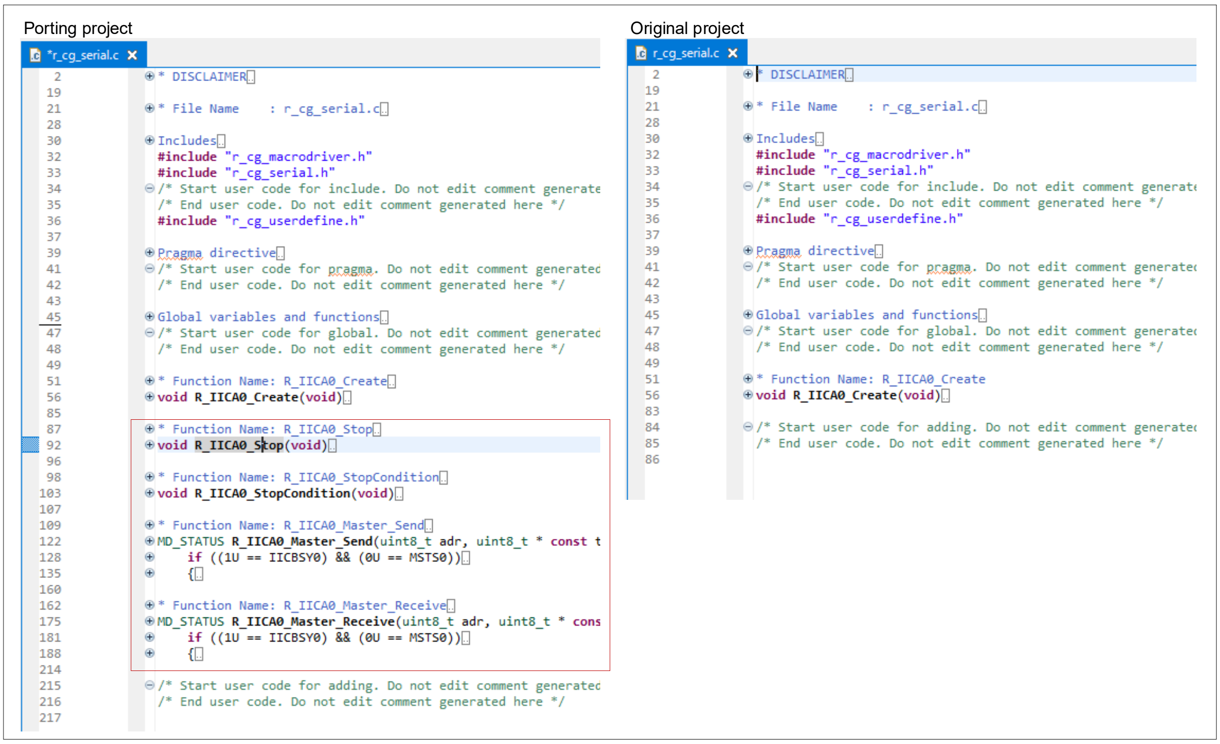
Porting project

```
* DISCLAIMER

* File Name    : r_cg_serial.c

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

Pragma directive
/* Start user code for pragma. Do not edit comment generated h
/* End user code. Do not edit comment generated here */

Global variables and functions
volatile uint8_t   g_iica0_master_status_flag; /* iica0 master
volatile uint8_t   g_iica0_slave_status_flag;  /* iica0 slave
volatile uint8_t * gp_iica0_rx_address;        /* iica0 receiv
volatile uint16_t  g_iica0_rx_len;             /* iica0 receiv
volatile uint16_t  g_iica0_rx_cnt;             /* iica0 receiv
volatile uint8_t * gp_iica0_tx_address;        /* iica0 send b
volatile uint16_t  g_iica0_tx_cnt;             /* iica0 send d
/* Start user code for global. Do not edit comment generated h
/* End user code. Do not edit comment generated here */

* Function Name: R_IICA0_Create
void R_IICA0_Create(void)
{
    IICA0EN = 1U; /* supply IICA0 clock */
```

Original project

```
* DISCLAIMER

* File Name    : r_cg_serial.c

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

Pragma directive
/* Start user code for pragma. Do not edit comment generate
/* End user code. Do not edit comment generated here */

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

* Function Name: R_IICA0_Create
void R_IICA0_Create(void)
{
```

Figure 3-40 r_cg_serial.c – Deletion Functions (Porting Example 2)



Porting project

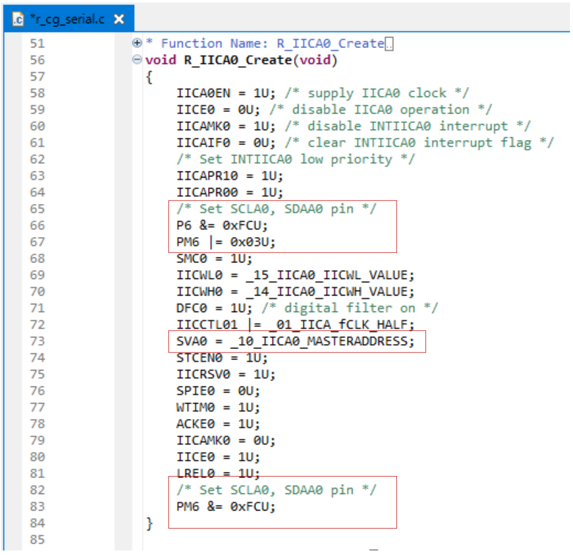
```
* DISCLAIMER

* File Name    : r_cg_serial.c

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

Pragma directive
/* Start user code for pragma. Do not edit comment generatec
/* End user code. Do not edit comment generated here */

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

* Function Name: R_IICA0_Create
void R_IICA0_Create(void)

* Function Name: R_IICA0_Stop
void R_IICA0_Stop(void)

* Function Name: R_IICA0_StopCondition
void R_IICA0_StopCondition(void)

* Function Name: R_IICA0_Master_Send
MD_STATUS R_IICA0_Master_Send(uint8_t adr, uint8_t * const t
    if ((1U == IICBSY0) && (0U == MSTS0))
    {

* Function Name: R_IICA0_Master_Receive
MD_STATUS R_IICA0_Master_Receive(uint8_t adr, uint8_t * cons
    if ((1U == IICBSY0) && (0U == MSTS0))
    {

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

Original project

```
* DISCLAIMER

* File Name    : r_cg_serial.c

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

Pragma directive
/* Start user code for pragma. Do not edit comment generatec
/* End user code. Do not edit comment generated here */

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

* Function Name: R_IICA0_Create
void R_IICA0_Create(void)

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

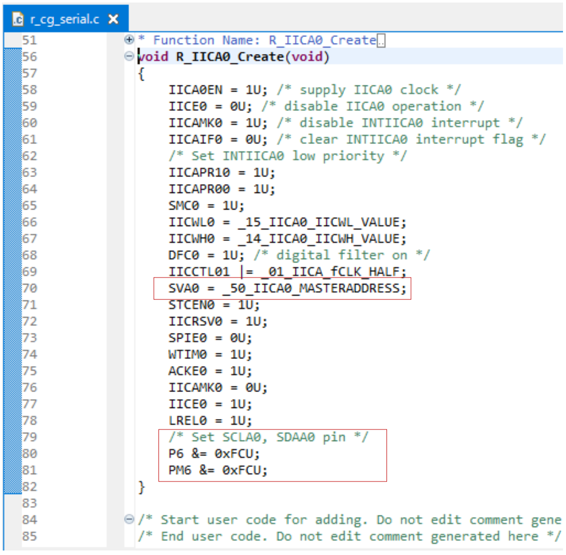
Figure 3-41 r_cg_serial.c – Editing Codes (Porting Example 2)

Porting project



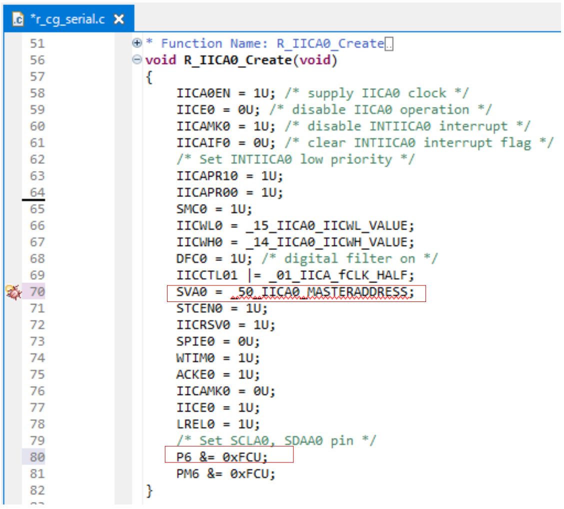
```
* Function Name: R_IICA0_Create
void R_IICA0_Create(void)
{
    IICA0EN = 1U; /* supply IICA0 clock */
    IICE0 = 0U; /* disable IICA0 operation */
    IICAMK0 = 1U; /* disable INTIICA0 interrupt */
    IICAIF0 = 0U; /* clear INTIICA0 interrupt flag */
    /* Set INTIICA0 low priority */
    IICAPR10 = 1U;
    IICAPR00 = 1U;
    /* Set SCLA0, SDAA0 pin */
    P6 &= 0xFCU;
    PM6 |= 0x03U;
    SMC0 = 1U;
    IICWL0 = _15_IICA0_IICWL_VALUE;
    IICWH0 = _14_IICA0_IICWH_VALUE;
    DFC0 = 1U; /* digital filter on */
    IICCTL01 |= _01_IICA_fCLK_HALF;
    SVA0 = _10_IICA0_MASTERADDRESS;
    STCEN0 = 1U;
    IICRSV0 = 1U;
    SPIE0 = 0U;
    WTIM0 = 1U;
    ACKE0 = 1U;
    IICAMK0 = 0U;
    IICE0 = 1U;
    LREL0 = 1U;
    /* Set SCLA0, SDAA0 pin */
    PM6 &= 0xFCU;
}
```

Original project



```
* Function Name: R_IICA0_Create
void R_IICA0_Create(void)
{
    IICA0EN = 1U; /* supply IICA0 clock */
    IICE0 = 0U; /* disable IICA0 operation */
    IICAMK0 = 1U; /* disable INTIICA0 interrupt */
    IICAIF0 = 0U; /* clear INTIICA0 interrupt flag */
    /* Set INTIICA0 low priority */
    IICAPR10 = 1U;
    IICAPR00 = 1U;
    SMC0 = 1U;
    IICWL0 = _15_IICA0_IICWL_VALUE;
    IICWH0 = _14_IICA0_IICWH_VALUE;
    DFC0 = 1U; /* digital filter on */
    IICCTL01 |= _01_IICA_fCLK_HALF;
    SVA0 = _50_IICA0_MASTERADDRESS;
    STCEN0 = 1U;
    IICRSV0 = 1U;
    SPIE0 = 0U;
    WTIM0 = 1U;
    ACKE0 = 1U;
    IICAMK0 = 0U;
    IICE0 = 1U;
    LREL0 = 1U;
    /* Set SCLA0, SDAA0 pin */
    P6 &= 0xFCU;
    PM6 &= 0xFCU;
}

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

Porting project: after the change

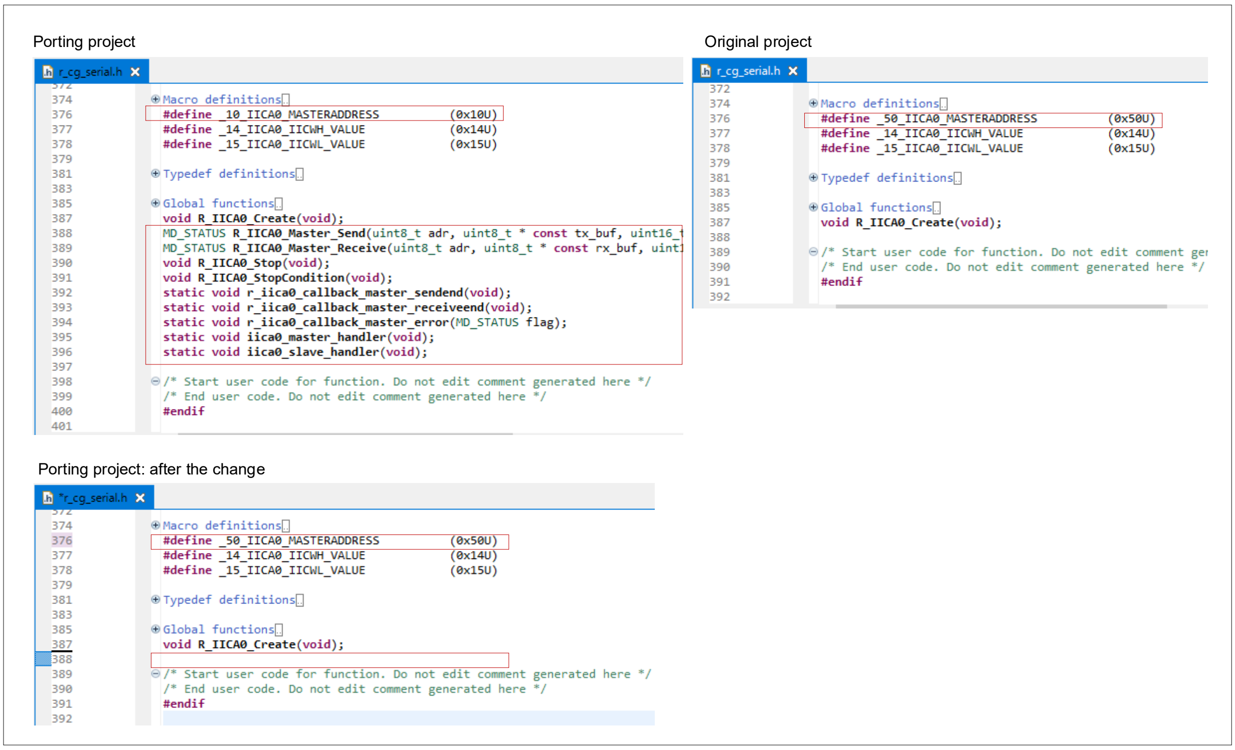


```
* Function Name: R_IICA0_Create
void R_IICA0_Create(void)
{
    IICA0EN = 1U; /* supply IICA0 clock */
    IICE0 = 0U; /* disable IICA0 operation */
    IICAMK0 = 1U; /* disable INTIICA0 interrupt */
    IICAIF0 = 0U; /* clear INTIICA0 interrupt flag */
    /* Set INTIICA0 low priority */
    IICAPR10 = 1U;
    IICAPR00 = 1U;
    SMC0 = 1U;
    IICWL0 = _15_IICA0_IICWL_VALUE;
    IICWH0 = _14_IICA0_IICWH_VALUE;
    DFC0 = 1U; /* digital filter on */
    IICCTL01 |= _01_IICA_fCLK_HALF;
    SVA0 = _50_IICA0_MASTERADDRESS;
    STCEN0 = 1U;
    IICRSV0 = 1U;
    SPIE0 = 0U;
    WTIM0 = 1U;
    ACKE0 = 1U;
    IICAMK0 = 0U;
    IICE0 = 1U;
    LREL0 = 1U;
    /* Set SCLA0, SDAA0 pin */
    P6 &= 0xFCU;
    PM6 &= 0xFCU;
}
```

(e) r_cg_serial.h

- Chane the definition of macro.
- Delete the prototype definition of the function “R_IICA0_Stop”, “R_IICA0_StopCondtition”, “R_IICA0_Master_Send” and “R_IICA0_Master_Receive” have been deleted from r_cg_serial.c.

Figure 3-42 r_cg_serial.h (Porting Example 2)



Porting project

```
Macro definitions
#define _10_IICA0_MASTERADDRESS              (0x10U)
#define _14_IICA0_IICWH_VALUE                (0x14U)
#define _15_IICA0_IICWL_VALUE                (0x15U)

Typedef definitions

Global functions
void R_IICA0_Create(void);
MD_STATUS R_IICA0_Master_Send(uint8_t adr, uint8_t * const tx_buf, uint16_t
MD_STATUS R_IICA0_Master_Receive(uint8_t adr, uint8_t * const rx_buf, uint1
void R_IICA0_Stop(void);
void R_IICA0_StopCondition(void);
static void r_iica0_callback_master_sendend(void);
static void r_iica0_callback_master_receiveend(void);
static void r_iica0_callback_master_error(MD_STATUS flag);
static void iica0_master_handler(void);
static void iica0_slave_handler(void);

/* Start user code for function. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#endif
```

Original project

```
Macro definitions
#define _50_IICA0_MASTERADDRESS              (0x50U)
#define _14_IICA0_IICWH_VALUE                (0x14U)
#define _15_IICA0_IICWL_VALUE                (0x15U)

Typedef definitions

Global functions
void R_IICA0_Create(void);

/* Start user code for function. Do not edit comment ger
/* End user code. Do not edit comment generated here */
#endif
```

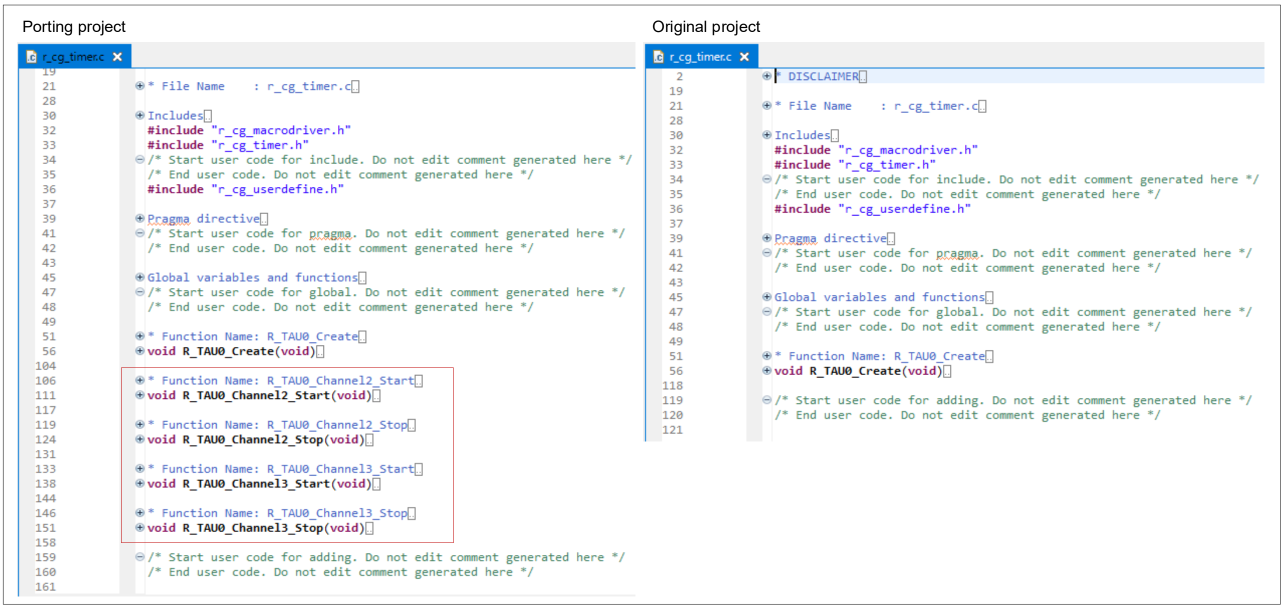
Porting project: after the change

```
Macro definitions
#define _50_IICA0_MASTERADDRESS              (0x50U)
#define _14_IICA0_IICWH_VALUE                (0x14U)
#define _15_IICA0_IICWL_VALUE                (0x15U)

Typedef definitions

Global functions
void R_IICA0_Create(void);

/* Start user code for function. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#endif
```

(f) r_cg_timer.c

Delete the function “R_TAU0_Channel2_Strat”, “R_TAU0_Channel2_Stop”, “R_TAU0_Channel3_Strat” and “R_TAU0_Channel3_Stop”.

Figure 3-43 r_cg_timer.c (Porting Example 2)



Porting project

```
* File Name    : r_cg_timer.c

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

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

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

* Function Name: R_TAU0_Create
void R_TAU0_Create(void)

* Function Name: R_TAU0_Channel2_Start
void R_TAU0_Channel2_Start(void)

* Function Name: R_TAU0_Channel2_Stop
void R_TAU0_Channel2_Stop(void)

* Function Name: R_TAU0_Channel3_Start
void R_TAU0_Channel3_Start(void)

* Function Name: R_TAU0_Channel3_Stop
void R_TAU0_Channel3_Stop(void)

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

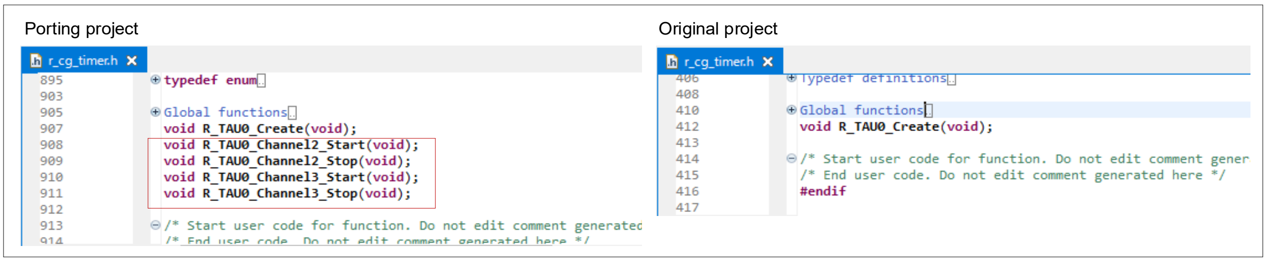
Original project

```
* DISCLAIMER

* File Name    : r_cg_timer.c

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

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

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

* Function Name: R_TAU0_Create
void R_TAU0_Create(void)

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

(g) r_cg_timer.h

Delete the prototype definition of the function “R_TAU0_Channel2_Strat”, “R_TAU0_Channel2_Stop”, “R_TAU0_Channel3_Strat” and “R_TAU0_Channel3_Stop” have been deleted form r_cg_timer.c.

Figure 3-44 r_cg_timer.h (Porting Example 2)



Porting project

```
typedef enum

Global functions
void R_TAU0_Create(void);
void R_TAU0_Channel2_Start(void);
void R_TAU0_Channel2_Stop(void);
void R_TAU0_Channel3_Start(void);
void R_TAU0_Channel3_Stop(void);

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

Original project

```
Typedef definitions

Global functions
void R_TAU0_Create(void);

/* Start user code for function. Do not edit comment gener
/* End user code. Do not edit comment generated here */
#endif
```

Save the edited files in the porting project. Select the [File] – [Save All] menu of e^2 studio.

3.2.7 Building Porting Project

Build the porting project after the target device was changed and the codes were edited to match the original project.

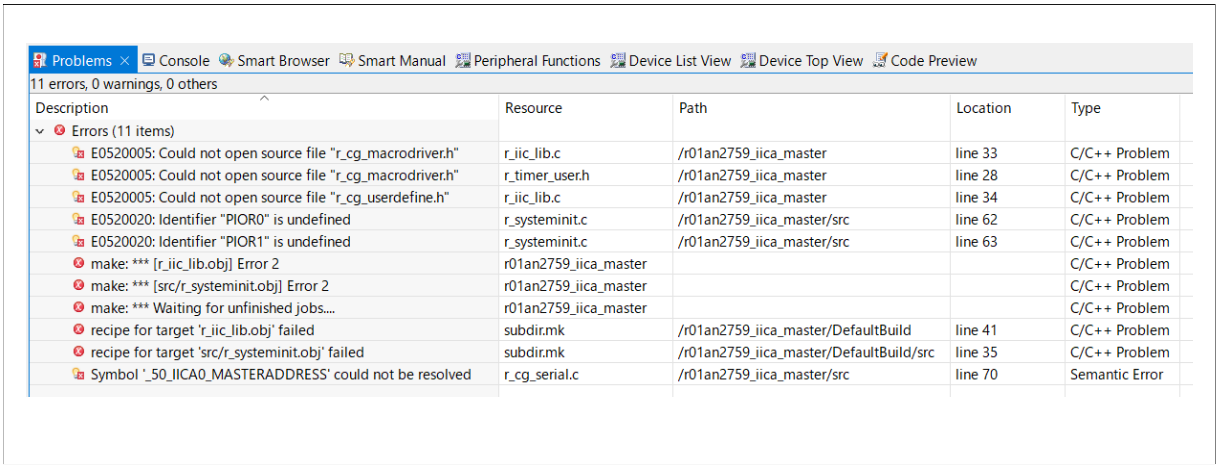
Select the [Project] – [Build Project] menu of e^2 studio.

The build error occurs in this porting example. Therefore, resolve the error based on the error message.

(1) Action for Error - 1

Because the include path to the “src” folder is not set, the error occurs.
Add the include path to the “src” folder in the Compiler option.

Figure 3-45 Build Error -1 (Porting Project 2)



11 errors, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (11 items)				
E0520005: Could not open source file "r_cg_macrodriver.h"	r_iic_lib.c	/r01an2759_iica_master	line 33	C/C++ Problem
E0520005: Could not open source file "r_cg_macrodriver.h"	r_timer_user.h	/r01an2759_iica_master	line 28	C/C++ Problem
E0520005: Could not open source file "r_cg_userdefine.h"	r_iic_lib.c	/r01an2759_iica_master	line 34	C/C++ Problem
E0520020: Identifier "PIOR0" is undefined	r_systeminit.c	/r01an2759_iica_master/src	line 62	C/C++ Problem
E0520020: Identifier "PIOR1" is undefined	r_systeminit.c	/r01an2759_iica_master/src	line 63	C/C++ Problem
make: *** [r_iic_lib.obj] Error 2	r01an2759_iica_master			C/C++ Problem
make: *** [src/r_systeminit.obj] Error 2	r01an2759_iica_master			C/C++ Problem
make: *** Waiting for unfinished jobs....	r01an2759_iica_master			C/C++ Problem
recipe for target 'r_iic_lib.obj' failed	subdir.mk	/r01an2759_iica_master/DefaultBuild	line 41	C/C++ Problem
recipe for target 'src/r_systeminit.obj' failed	subdir.mk	/r01an2759_iica_master/DefaultBuild/src	line 35	C/C++ Problem
Symbol '_50_IICA0_MASTERADDRESS' could not be resolved	r_cg_serial.c	/r01an2759_iica_master/src	line 70	Semantic Error

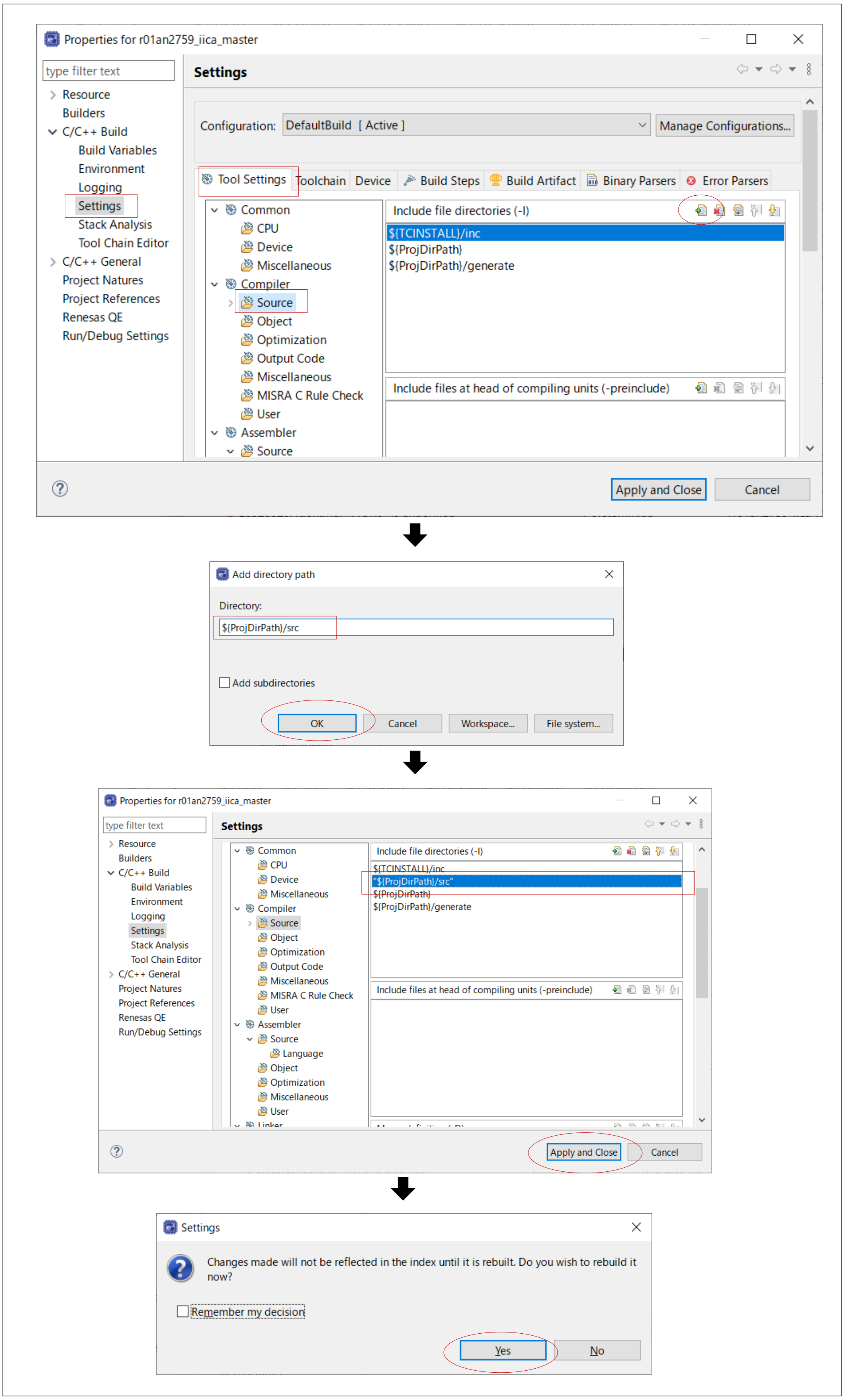
- Adding the include path

  Select the [Project] – [C/C++ Project Settings] menu of e^2 studio. Select [C/C++ Build] – [Settings] in the [Properties] dialog box and add the include path in [Source] of the [Tool Settings] tab.

  After adding the path, select the [Project] – [Build Project] menu to build again.

  Path to add: ${ProjDirPath}/src

Figure 3-46 Adding Include Path (Porting Example 2)


Properties for r01an2759_iica_master

C/C++ Build > Settings > Tool Settings > Compiler > Source

Configuration: DefaultBuild [Active]

Include file directories (-I)

```
${TCINSTALL}/inc
${ProjDirPath}
${ProjDirPath}/generate
```

Include files at head of compiling units (-preinclude)

Add directory path

Directory: ${ProjDirPath}/src

□ Add subdirectories

Properties for r01an2759_iica_master

Include file directories (-I)

```
${TCINSTALL}/inc
"${ProjDirPath}/src"
${ProjDirPath}
${ProjDirPath}/generate
```

Settings

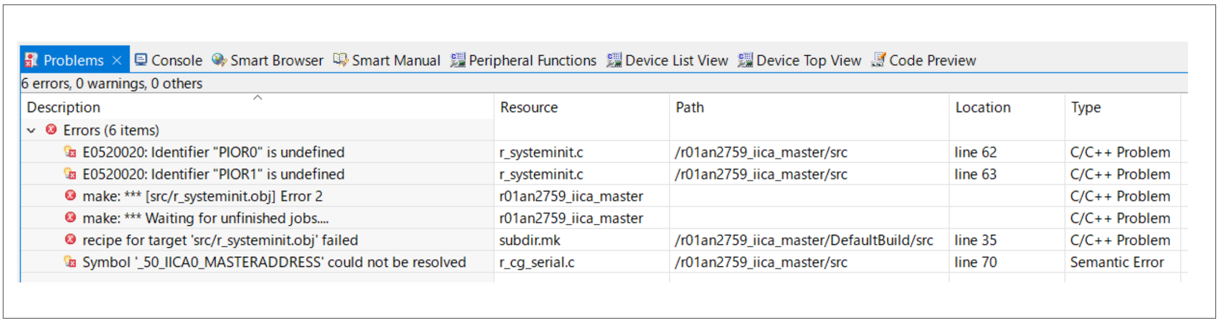
Changes made will not be reflected in the index until it is rebuilt. Do you wish to rebuild it now?

□ Remember my decision

(2) Action for Error - 2

Because “iodefine.h” (for the RL78/G13) that has existed from the original project is included, not “iodefine.h” (for the RL78/G14) in the “generate” folder generated when changing the device, so that the error occurs. To include “iodefine.h” in the “generate” folder, move the “generate” folder to the top in the Compiler option.

Figure 3-47 Build Error - 2 (Porting Example 2)



6 errors, 0 warnings, 0 others

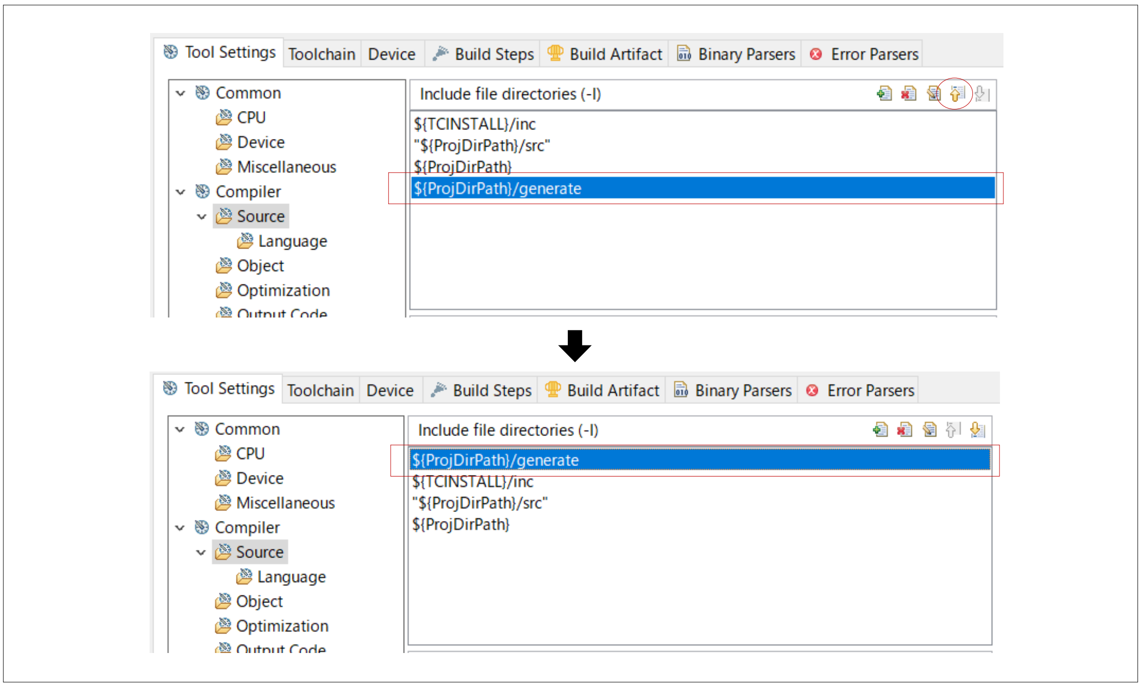
Description	Resource	Path	Location	Type
Errors (6 items)				
E0520020: Identifier "PIOR0" is undefined	r_systeminit.c	/r01an2759_iica_master/src	line 62	C/C++ Problem
E0520020: Identifier "PIOR1" is undefined	r_systeminit.c	/r01an2759_iica_master/src	line 63	C/C++ Problem
make: *** [src/r_systeminit.obj] Error 2	r01an2759_iica_master			C/C++ Problem
make: *** Waiting for unfinished jobs....	r01an2759_iica_master			C/C++ Problem
recipe for target 'src/r_systeminit.obj' failed	subdir.mk	/r01an2759_iica_master/DefaultBuild/src	line 35	C/C++ Problem
Symbol '_50_IICA0_MASTERADDRESS' could not be resolved	r_cg_serial.c	/r01an2759_iica_master/src	line 70	Semantic Error

- Changing the order of the include paths

  Select the [Project] – [C/C++ Project Settings] menu of e^2 studio. Select [C/C++ Build] – [Settings] in the [Properties] dialog box and change the order of the include path in [Source] of the [Tool Settings] tab.

  After changing of the path order, select the [Project] – [Build Project] menu to build again.

Figure 3-48 Changing Order of Include Path (Porting Example 2)



Include file directories (-I)

```
${TCINSTALL}/inc
"${ProjDirPath}/src"
${ProjDirPath}
${ProjDirPath}/generate
```

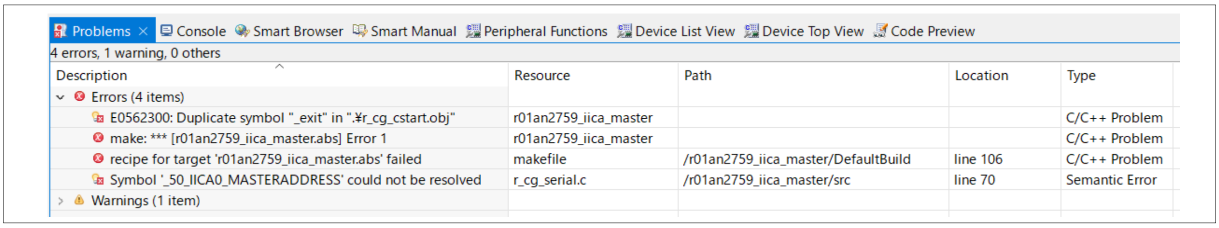
Include file directories (-I)

```
${ProjDirPath}/generate
${TCINSTALL}/inc
"${ProjDirPath}/src"
${ProjDirPath}
```

(3) Action for Error - 3

Because the "_exit" symbol is defined in duplicate, the error occurs.
Check the file in which the "_exit" is defined and act to resolve the error.

Figure 3-49 Build Error - 3 (Porting Example 2)



4 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Errors (4 items)				
E0562300: Duplicate symbol "_exit" in ".¥r_cg_cstart.obj"	r01an2759_iica_master			C/C++ Problem
make: *** [r01an2759_iica_master.abs] Error 1	r01an2759_iica_master			C/C++ Problem
recipe for target 'r01an2759_iica_master.abs' failed	makefile	/r01an2759_iica_master/DefaultBuild	line 106	C/C++ Problem
Symbol '_50_IICA0_MASTERADDRESS' could not be resolved	r_cg_serial.c	/r01an2759_iica_master/src	line 70	Semantic Error
Warnings (1 item)				

- Searching the file in which "_exit" is defined

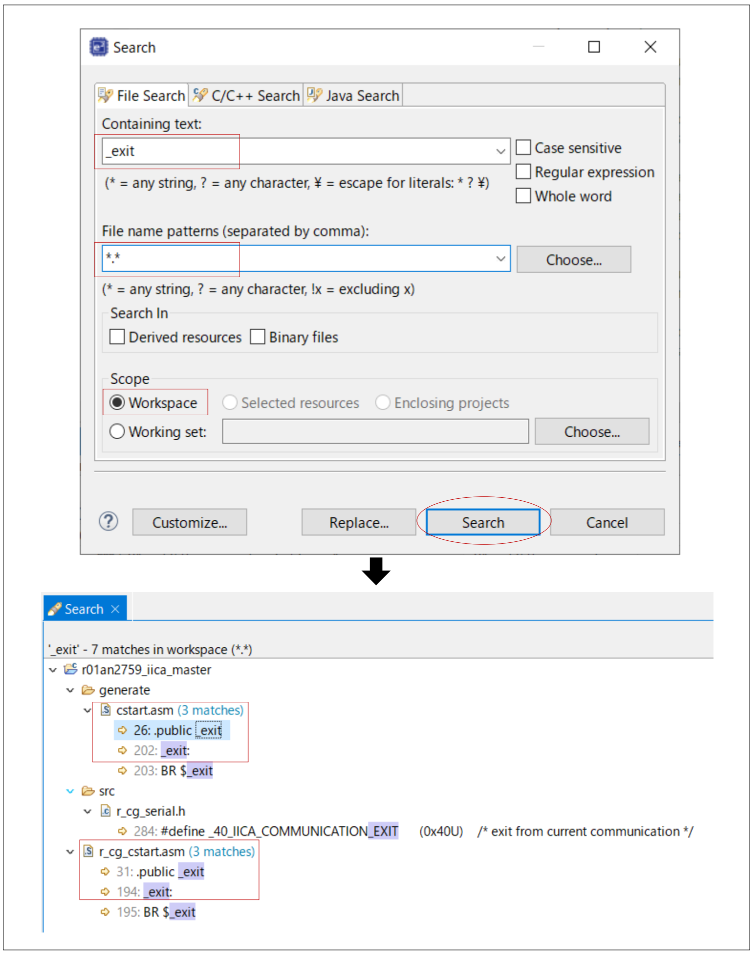
Select the [Search] – [Search] menu of e^2 studio. Input keywords in the [Search] dialog box to search the code for the "_exit" symbol

Containing text: _exit

File name patterns: *.*

Scope: Workspace

Figure 3-50 [Search] dialog box (Porting Example 2)



Containing text:
_exit
□ Case sensitive
□ Regular expression
□ Whole word
(* = any string, ? = any character, ¥ = escape for literals: * ? ¥)

File name patterns (separated by comma):
.
(* = any string, ? = any character, !x = excluding x)

Search In
□ Derived resources □ Binary files

Scope
● Workspace ○ Selected resources ○ Enclosing projects
○ Working set:

'_exit' - 7 matches in workspace (*.*)

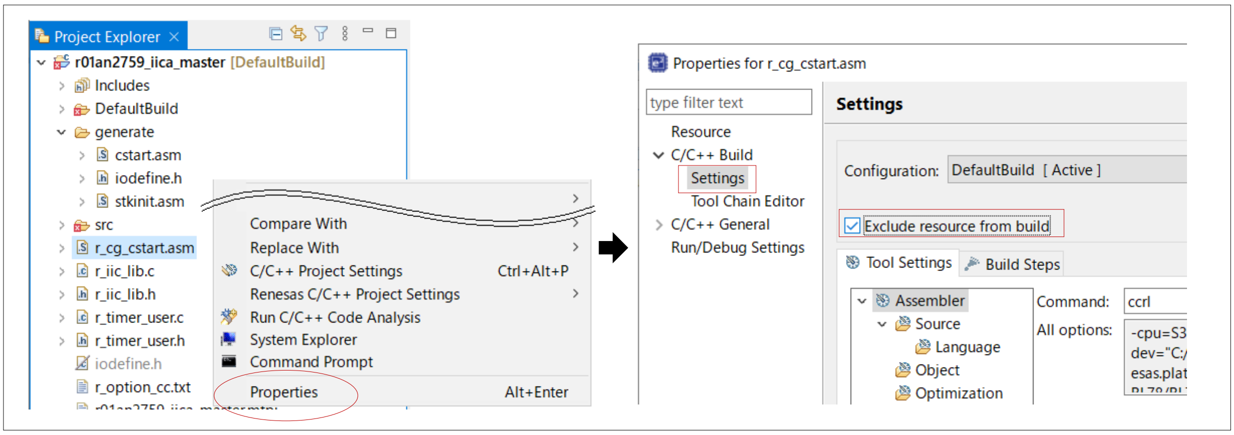
```
r01an2759_iica_master
  generate
    cstart.asm (3 matches)
      26: .public _exit
      202: _exit:
      203: BR $_exit
  src
    r_cg_serial.h
      284: #define _40_IICA_COMMUNICATION_EXIT    (0x40U)    /* exit from current communication */
  r_cg_cstart.asm (3 matches)
    31: .public _exit
    194: _exit:
    195: BR $_exit
```

- Comparing “r_cg_cstart.asm” and “cstart.asm”

Open “r_cg_cstart.asm” and “cstart.asm” in the editor and check the code in them.
After checking, both files are the startup file, therefore, keep “cstart.asm” generated after the device changes and exclude “r_cg_cstart.asm” from the build target.

After excluding “r_cg_cstart.asm” from the build target, select the [Project] – [Build Project] menu to build again. Some warnings remain, but all errors are resolved.

Figure 3-51 Excluding “r_cg_cstart.asm” from Build Target (Porting Example 2)



4. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

5. Reference Documents

RL78/G13 User’s Manual: Hardware (R01UH0146J)

RL78 family user's manual software (R01US0015J)

The latest versions can be downloaded from the Renesas Electronics website.

Technical update

The latest versions can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.0	Apr.04.2022	-	First Edition

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

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

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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