RL78 Family

Renesas Flash Driver RL78 Type 11 SC version (Bank Operation)

Introduction

This document explains Renesas Flash Driver RL78 Type 11 for the RL78/L2x group in the case of using Smart Configurator(SC). It is a process which builds into a user program the functions for “the bank programming / bank swapping control driver” included in RFD RL78 Type 11, and it is a method for executing bank programming control and bank swapping control using the attached sample program.

In this document, “Renesas Flash Driver RL78 Type 11” is described to be “RFD RL78 Type 11” or “RFD”.

This document includes the contents in comparison with conventional RFD RL78 Type 11 not using SC. This document distinguishes and expresses it.

Simple version : Conventional RFD RL78 Type 11 not using SC.
SC version : RFD RL78 Type 11 using “SC” currently explained by this document.

Target Device

The target device group by which the operation for RFD RL78 Type 11 was confirmed.

RL78/L23 group

If this application note is applied to other microcomputers, it is necessary to modify in accordance with the specification of the microcomputer. And, be sure to evaluate enough.

Contents

1. Specification 3
1.1 Operating Environment 3
1.2 Structure of Sample Program Folders 4
1.3 File Structure of RFD Driver 5
1.3.1 File Structure of RFD Common Driver (r_rfd_rl78_common) 5
1.3.2 File Structure for RFD Bank Programming / Bank Swapping Control Driver (r_rfd_rl78_bankoperation) 6
1.4 Bank Programming and Bank Swapping Control Processing Using a Sample Program 7

2. Creating a Sample Project for Bank Programming / Bank Swapping Control 8
2.1 Example of Creating a Sample Project 8
2.1.1 In Case of CS+ 8
2.1.2 In Case of e² studio(CC-RL) 8
2.1.3 In Case of IAR EW for Renesas RL78 8
2.1.4 In Case of e² studio(LLVM) 9
2.2 Example of Source Code Registration 10
2.2.1 In Case of CS+ 10
2.2.2 In Case of e² studio(CC-RL) 13
2.2.3 In Case of IAR EW for Renesas RL78 16
2.2.4 In Case of e² studio(LLVM) 21
2.3 Project registration of sample program 24
2.4 The Check of Operation for Sample Program 39
2.4.1 In Case of CS+ 39
2.4.2 In Case of e² studio(CC-RL) 41
2.4.3 In Case of IAR EW for Renesas RL78 44
2.4.4 In Case of e² studio(LLVM) 45

3. Precautions 48

4. Reference document 50

5. Revision History 51

1. Specification

This sample program executes bank programming control and bank swap control. Bank programming control executes the user program which reprograms rewrite bank located to startup bank. Bank swap control replaces startup bank and rewrite bank.

1.1 Operating Environment

- C Compiler Packages

Table 1-1 The target C Compiler Packages

Package	IDE (Integrated Development Environment)	Manufacturer	Version
CC-RL	CS+ or e^2 studio	Renesas Electronics	V1.15 or later
IAR	IAR Embedded Workbench® for Renesas RL78	IAR Systems®	V5.10.3 or later
LLVM	e^2 studio	(Open source software)	V17.0.1.202412 or later

Note. Integrated development environment and compiler must support the target device.
IAR Systems, IAR Embedded Workbench, IAR, and the logotype of IAR Systems are trademarks or registered trademarks owned by IAR Systems AB.

- Emulator
  Table 1-2 shows the emulator on which the operation of RFD RL78 Type 11 was confirmed.

Table 1-2 Emulator on which RFD RL78 Type 11 Operation was Confirmed

Emulator	Manufacturer
E2 emulator Lite	Renesas Electronics

- Target MCU

  RL78/L23

- Renesas Flash Driver (RFD) RL78 Type 11

  Table 1-3 shows the Renesas Flash Driver (RFD) RL78 Type 11 supported by this manual.

Table 1-3 The RFD RL78 Type 11 Supported by This Manual

Package	Manufacturer	Package Version
RFD RL78 Type 11	Renesas Electronics	V1.00

1.2 Structure of Sample Program Folders

Figure 1.1 shows the structure of sample program folders.

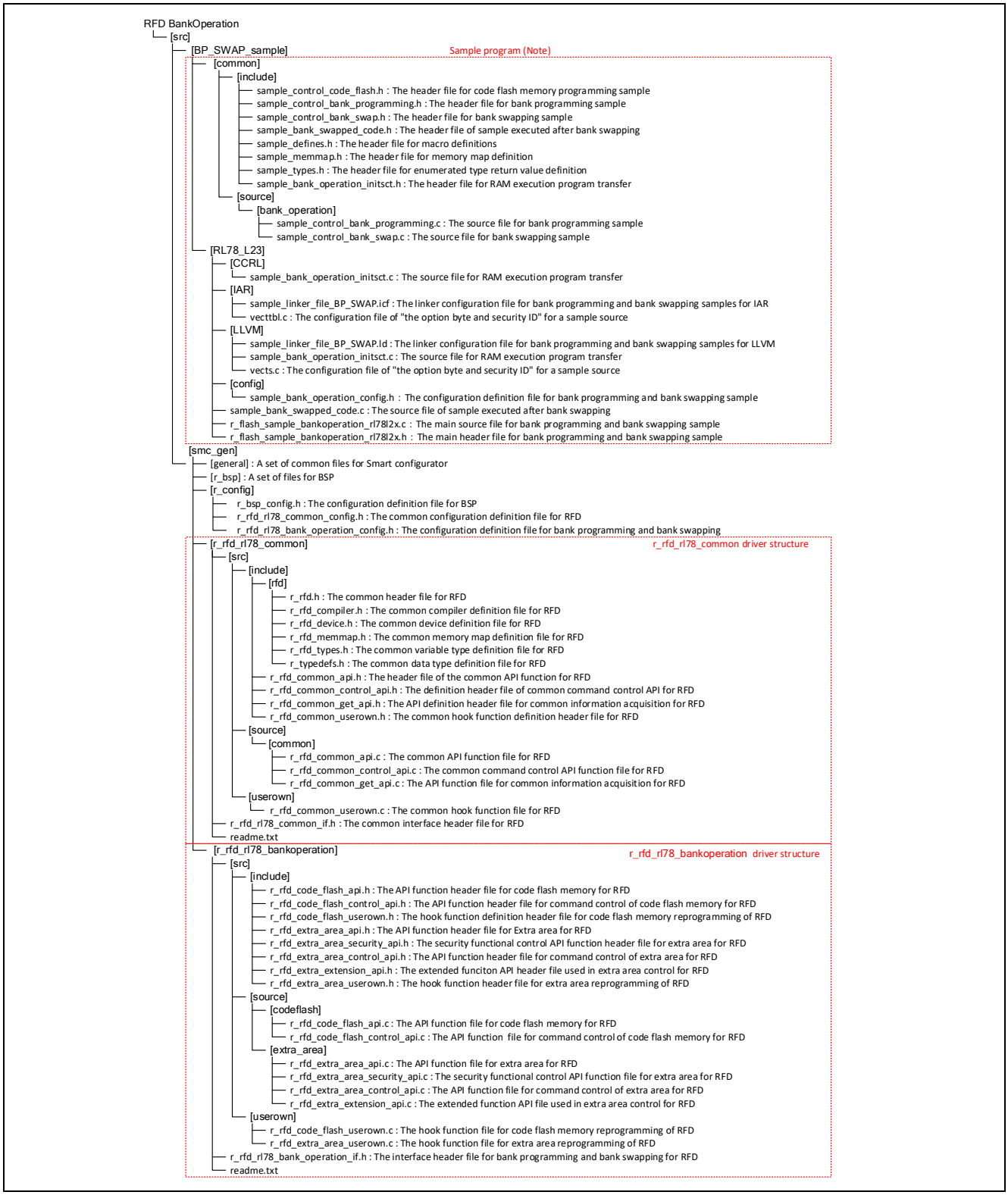


Figure 1.1 Structure of Sample Program Folders

Note: The sample program of a zip file format downloads by Smart configurator. Extract the compressed file (BP_SWAP_sample.zip) outputted to the “demo” folder, and move the [BP_SWAP_sample] folder under the [src] folder. Refer to “2.3 Project registration of sample program” for the details of project registration.

1.3 File Structure of RFD Driver

1.3.1 File Structure of RFD Common Driver (r_rfd_rl78_common)

The difference of SC version RFD common driver and Simple version common driver is shown.Refer to the “RL78 Family Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539) ” for the detail specification for the RFD common driver.

Table 1-4 Difference of the files of the SC version RFD and the simple version RFD (Common API : r_rfd_rl78_common\src\source\common)

File name	Simple version	SC version
r_rfd_common_api.c	No change	
r_rfd_common_control_api.c	No change	
r_rfd_common_get_api.c	No change	

Table 1-5 Difference of the file of the SC version RFD and the simple version RFD (Common API : r_rfd_rl78_common\src\userown)

File name	Simple version	SC version
r_rfd_common_userown.c	No change	

Table 1-6 Difference of the files of the SC version RFD and the simple version RFD (Common header : r_rfd_rl78_common\src\include)

File name	Simple version	SC version
r_rfd_common_api.h	No change	
r_rfd_common_control_api.h	No change	
r_rfd_common_get_api.h	No change	
r_rfd_common_userown.h	No change	

Table 1-7 Difference of the files of the SC version RFD and the simple version RFD (Common header : r_rfd_rl78_common\src\include\rfd)

File name	Simple version	SC version
r_rfd.h	No change	
r_rfd_compiler.h	No change	
r_rfd_device.h	No change	
r_rfd_memmap.h	No change	
r_rfd_types.h	No change	
r_typedefs.h	No change	

Table 1-8 Difference of the file of the SC version RFD and the simple version RFD (Common interface header : r_rfd_rl78_common)

File name	Simple version	SC version
r_rfd_rl78_common_if.h	-	Newly created. Include the header file for common API.

1.3.2 File Structure for RFD Bank Programming / Bank Swapping Control Driver (r_rfd_rl78_bankoperation)

The difference of SC version RFD bank programming / bank swapping control driver and Simple version bank programming / bank swapping control driver is shown. Refer to the “RL78 Family Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539) ” for the detail specification for the RFD bank programming / bank swapping control driver.

Table 1-9 Difference of the files of the SC version RFD and the simple version RFD (Code flash API:r_rfd_rl78_bankoperation\src\source\codeflash)

File name	Simple version	SC version
r_rfd_code_flash_api.c	No change	
r_rfd_code_flash_control_api.c	No change	

Table 1-10 Difference of the files of the SC version RFD and the simple version RFD (Extra area API:r_rfd_rl78_bankoperation\src\source\extra_area)

File name	Simple version	SC version
r_rfd_extra_area_api.c	No change	
r_rfd_extra_area_security_api.c	No change	
r_rfd_extra_area_control_api.c	No change	
r_rfd_extra_extension_api.c	No change	

Table 1-11 Difference of the files of the SC version RFD and the simple version RFD (Extra area API : r_rfd_rl78_bankoperation\src\source\userown)

File name	Simple version	SC version
r_rfd_code_flash_userown.c	No change	
r_rfd_extra_area_userown.c	No change	

Table 1-12 Difference of the files of the SC version RFD and the simple version RFD (Code flash API / Extra area API header : r_rfd_rl78_bankoperation\src\include)

File name	Simple version	SC version
r_rfd_code_flash_api.h	No change	
r_rfd_code_flash_control_api.h	No change	
r_rfd_extra_area_api.h	No change	
r_rfd_extra_area_security_api.h	No change	
r_rfd_extra_area_control_api.h	No change	
r_rfd_extra_extension_api.h	No change	
r_rfd_code_flash_userown.h	No change	
r_rfd_extra_area_userown.h	No change	

Table 1-13 Difference of the file of the SC version RFD and the simple version RFD (Bank Operation interface header : r_rfd_rl78_bankoperation)

File name	Simple version	SC version
r_rfd_rl78_bank_operation_if.h	-	Newly created. Include the header file for code flash API and extra area API.

1.4 Bank Programming and Bank Swapping Control Processing Using a Sample Program

Figure 1.2 shows the flow chart of the sample program. The sample_bankoperation_main function copies the “program executed on RAM” to RAM area from ROM area. And bank swapping control is executed on RAM.

Sample_BankProgrammingControl function and Sample_BankSwapControl function processing do not have change from Simple version. Refer to the item of “Sample_BankProgrammingControl function” and “Sample_BankSwapControl function” on Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539).

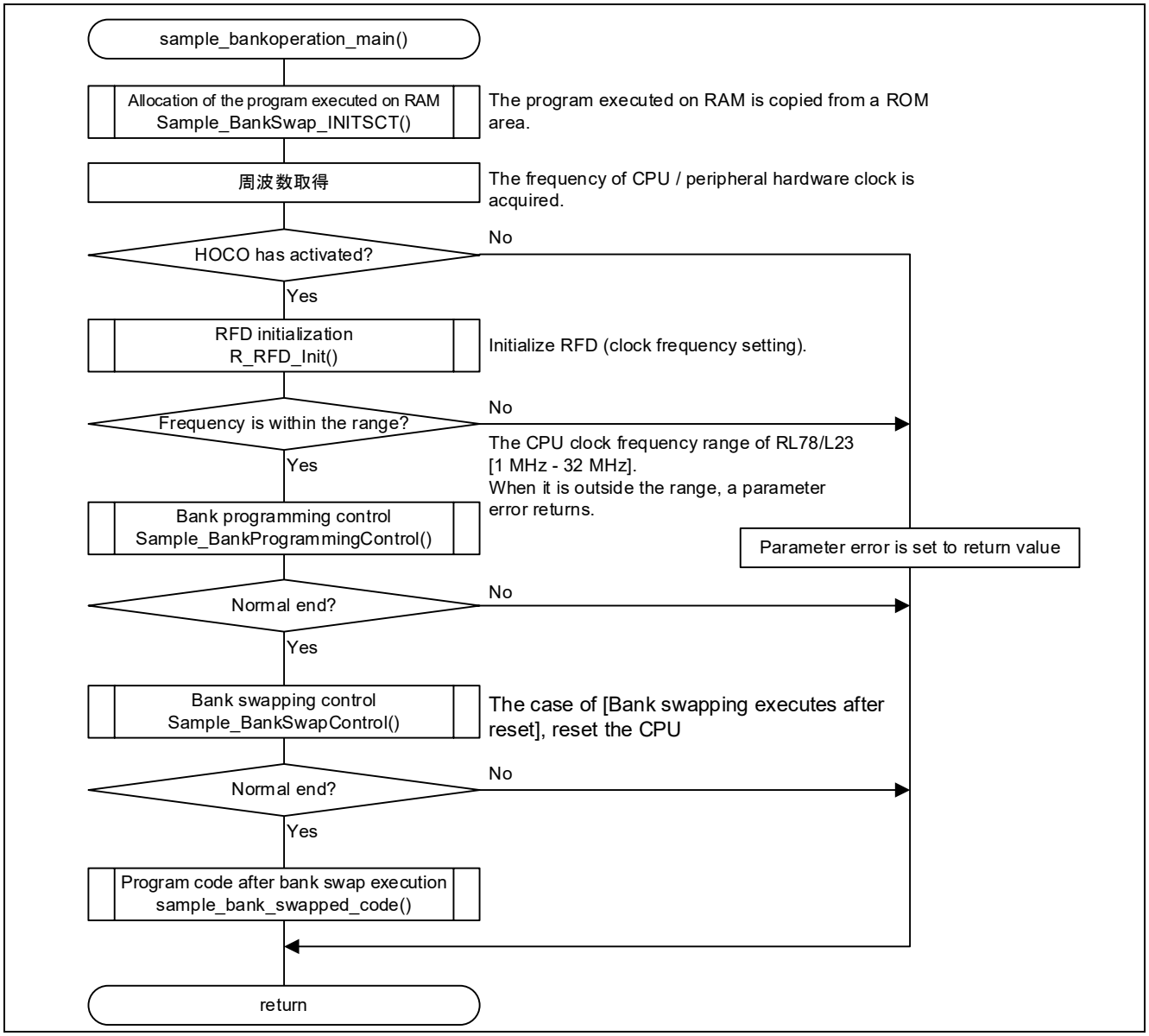


Figure 1.2 the flow chart of the sample program

Notes: 1. In the code flash memory programming mode, the programs in the code flash memory cannot be executed. Copy the Sample_BankSwapControl function and the processing to be executed and data to be referenced to inside the function to RAM in advance, and execute and reference them in RAM.

2. “Frequency acquisition” of CPU and peripheral hardware clock settings is using the function included in the “RL78 Family Board Support Package”.

2. Creating a Sample Project for Bank Programming / Bank Swapping Control

2.1 Example of Creating a Sample Project

2.1.1 In Case of CS+

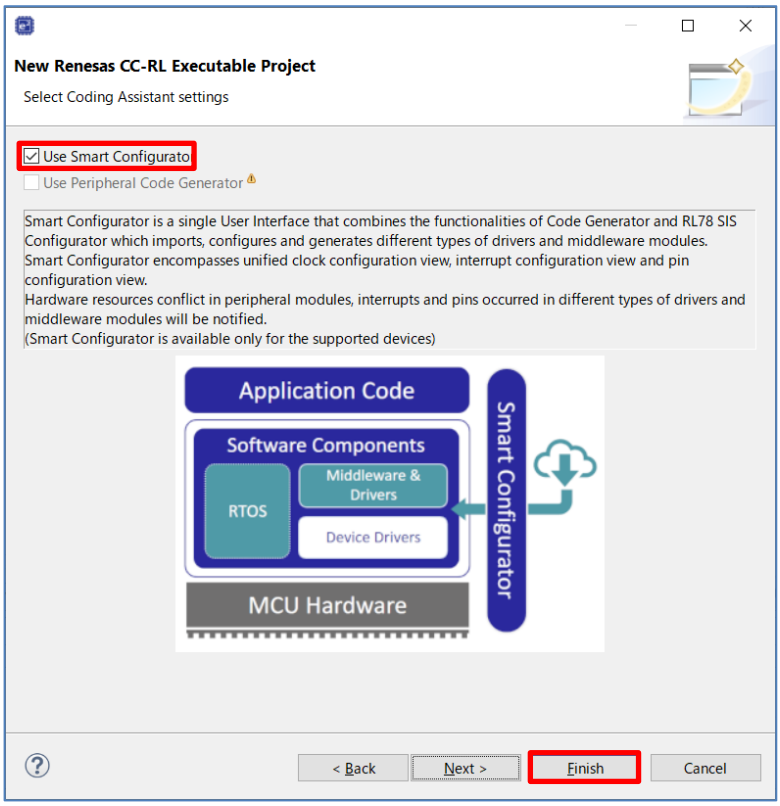
Refer to Renesas Flash Driver RL78 Type 11 user's manual (R20UT5539) "Example of Creating a Sample Project" to create a project.

2.1.2 In Case of e^2 studio(CC-RL)

Refer to Renesas Flash Driver RL78 Type 11 user's manual (R20UT5539) "Example of Creating a Sample Project" to create a project.

In this application note, because smart Configurator is used, press a "Next" button after selecting a target device and a debugging tool. And perform the following processes.

Select "Use Smart Configurator" and press a "Finish" button.



2.1.3 In Case of IAR EW for Renesas RL78

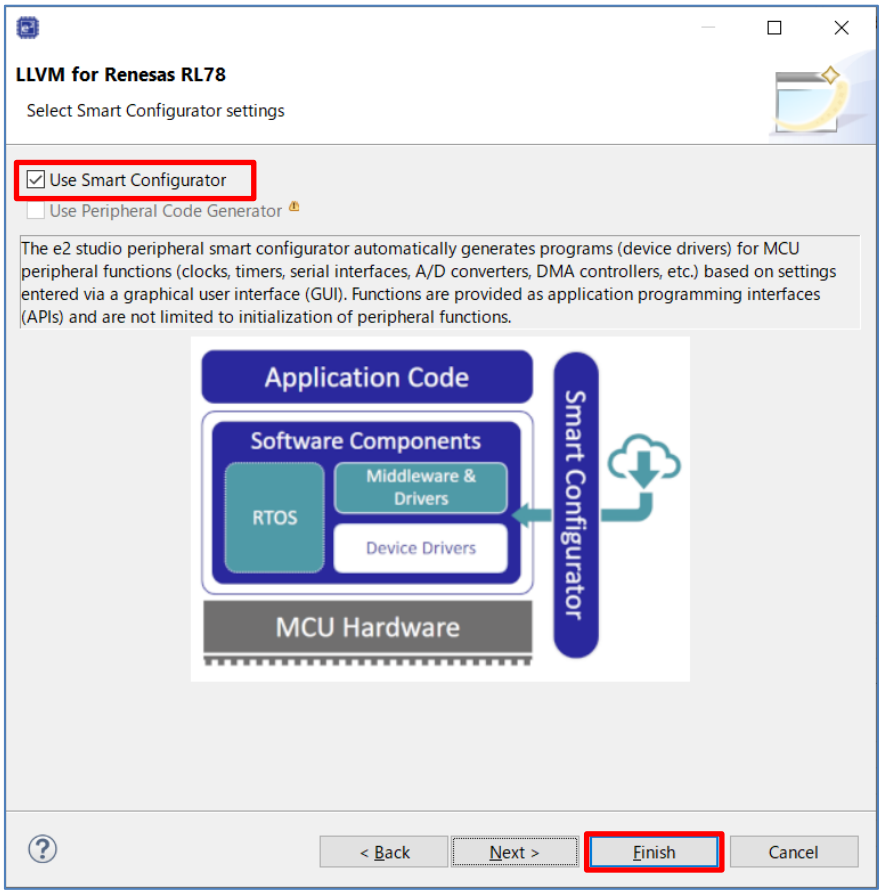
Refer to Renesas Flash Driver RL78 Type 11 user's manual (R20UT5539) "Example of Creating a Sample Project" to create a project.

2.1.4 In Case of e² studio(LLVM)

Refer to Renesas Flash Driver RL78 Type 11 user's manual (R20UT5539) "Example of Creating a Sample Project" to create a project.

In this application note, because smart Configurator is used, press a "Next" button after selecting a target device and a debugging tool. And perform the following processes.

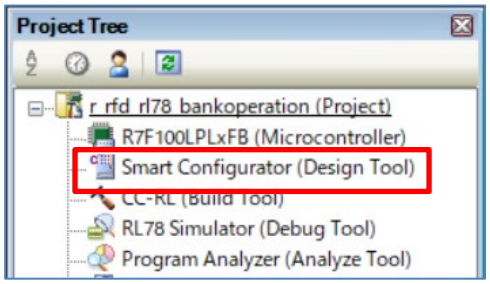
Select "Use Smart Configurator" and press a "Finish" button.



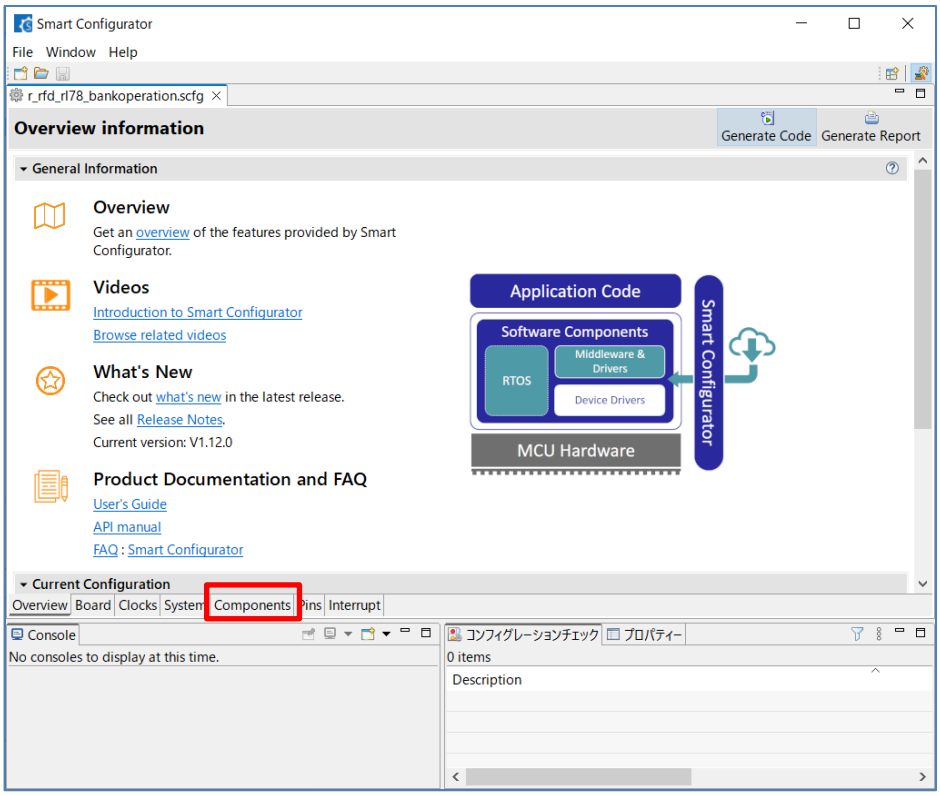
2.2 Example of Source Code Registration

2.2.1 In Case of CS+

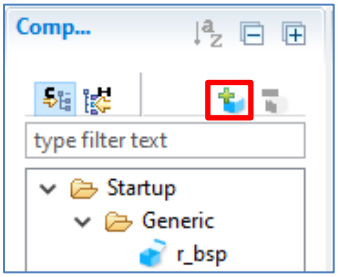
(1) Double-click “Smart Configurator” (design Tool) of “Project Tree”, and start Smart Configurator.



(2) Select a “Components” tab.



(3) Press the “Add component” button and open the “New Component” dialog.

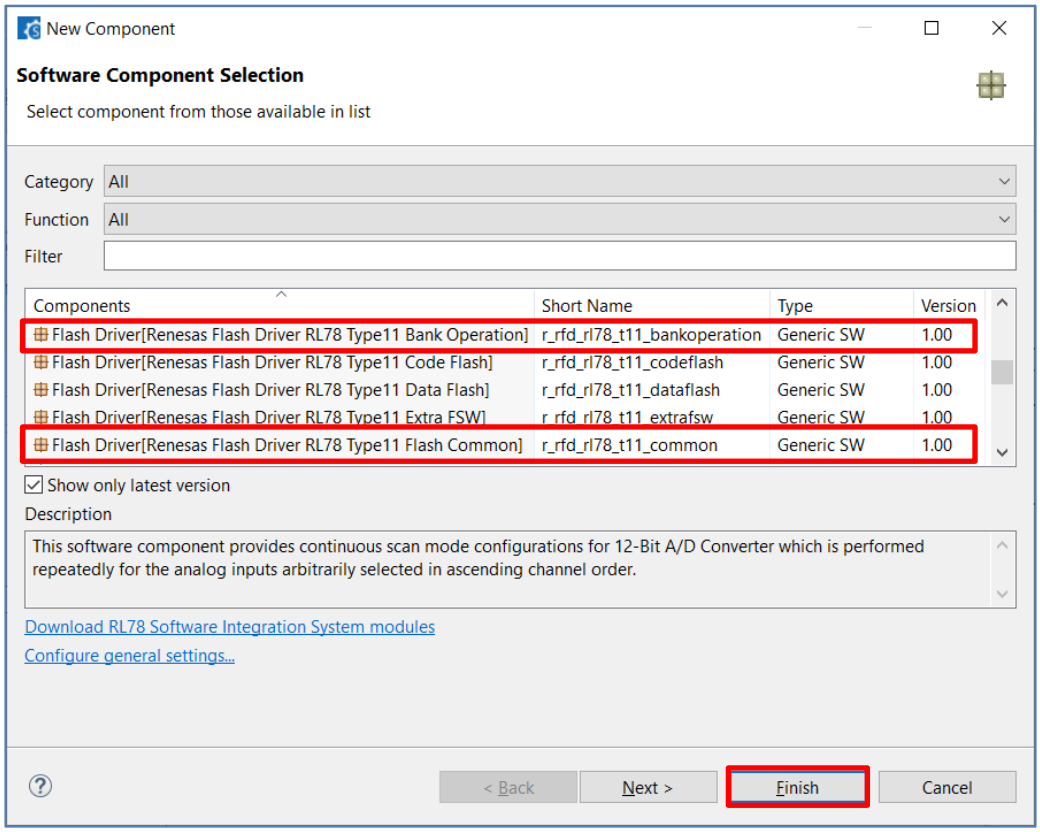


(4) Select the following components and press a “Finish” button.

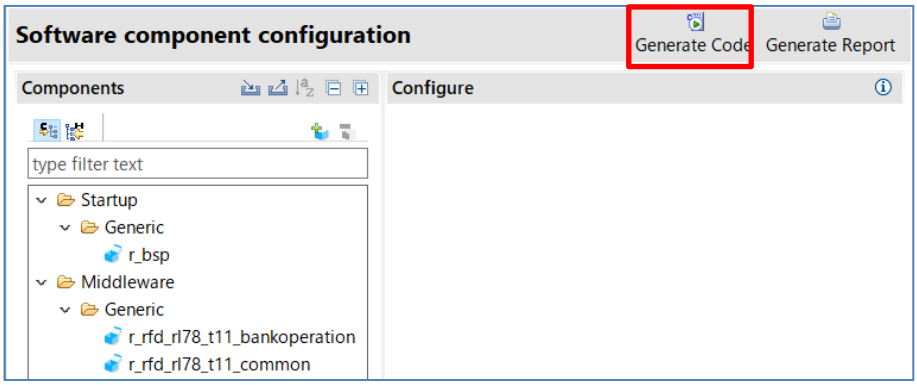
- Flash Driver[Renesas Flash Driver RL78 Type11 Bank Operation]

(r_rfd_rl78_t11_bankoperation)

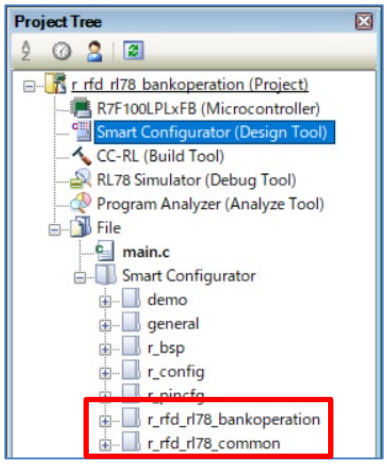
- Flash Driver[Renesas Flash Driver RL78 Type 11 Flash Common](r_rfd_rl78_t11_common)



(5) Press a “Generate Code” button and close “Smart Configurator” after the completion of generation for the code.

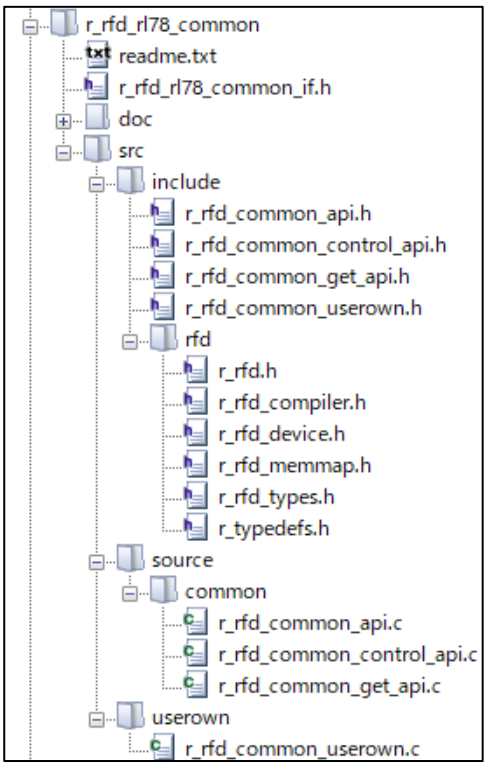


(6) The “r_rfd_rl78_common” folder and the “r_rfd_rl78_bankoperation” folder are added to the project tree.

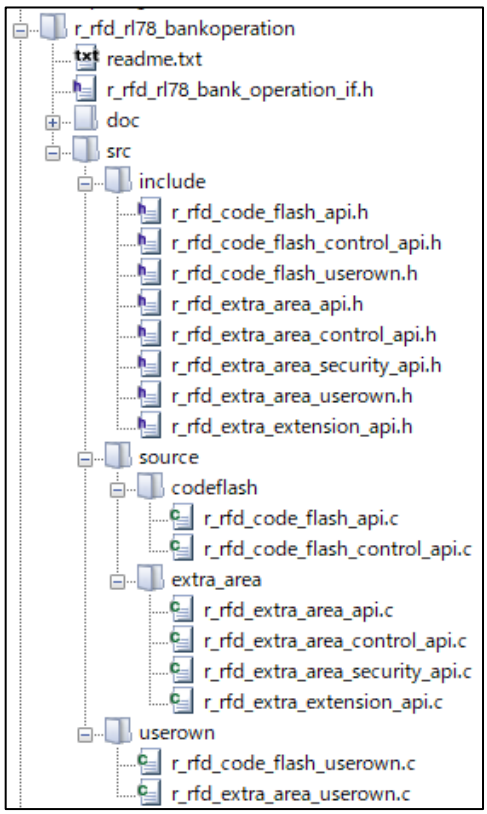


Each folder is developed as follows

The developed r_config folder



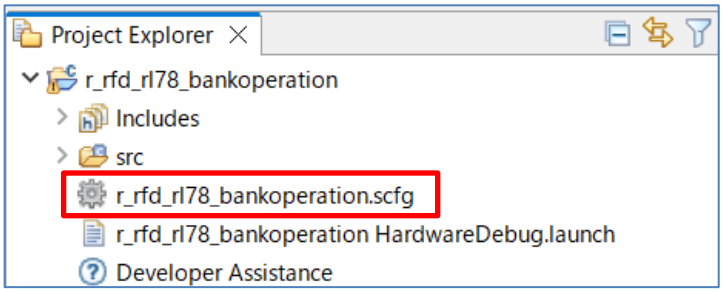
The developed r_rfd_rl78_common folder



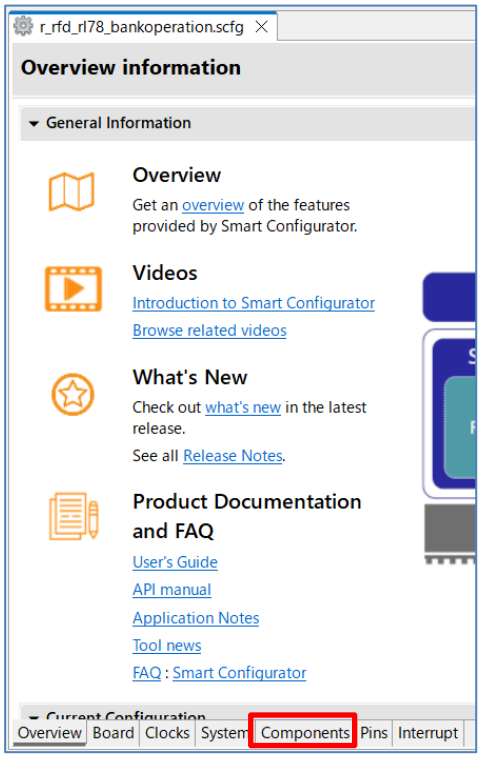
The developed r_rfd_rl78_bankoperation folder

2.2.2 In Case of e² studio(CC-RL)

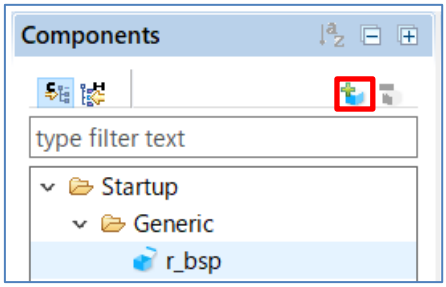
(1) Open the project file of “Smart Configurator” after starting e² studio.



(2) Select a “Components” tab.

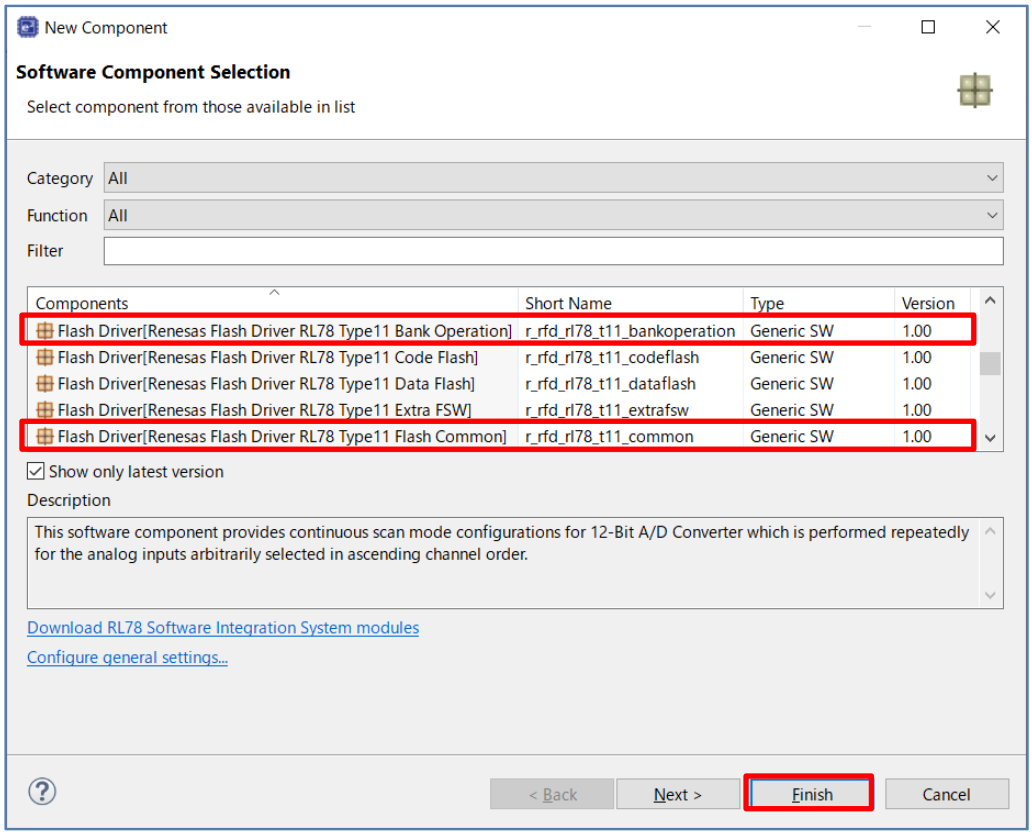


(3) Press the “Add component” button and open the “New Component” dialog.

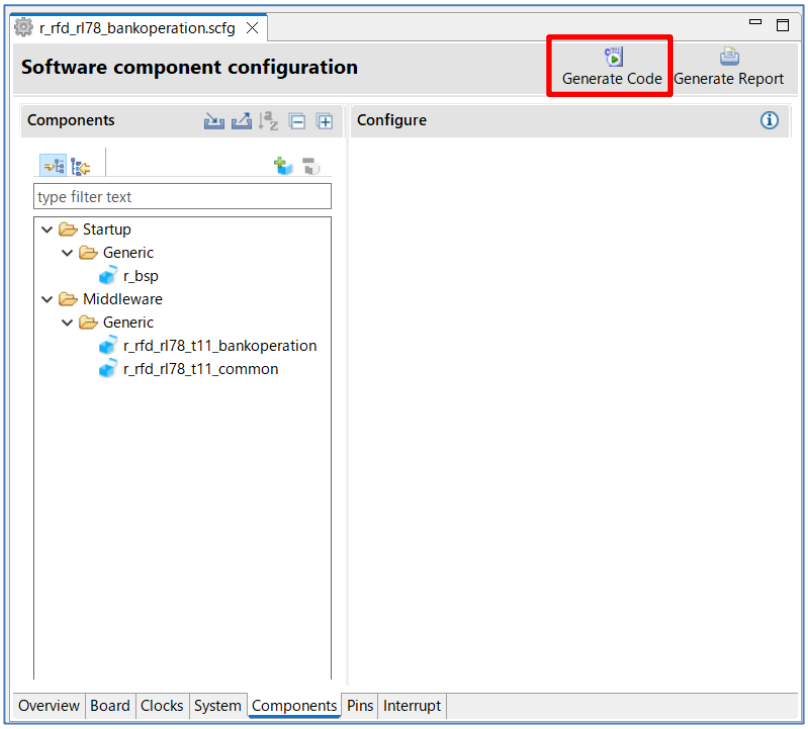


(4) Select the following components and press a “Finish” button.

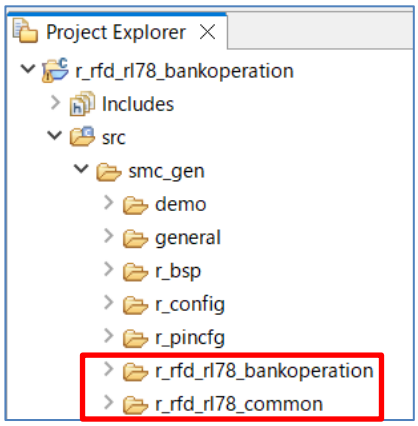
- Flash Driver[Renesas Flash Driver RL78 Type11 Bank Operation]
(r_rfd_rl78_t11_bankoperation)
- Flash Driver[Renesas Flash Driver RL78 Type 11 Flash Common](r_rfd_rl78_t11_common)



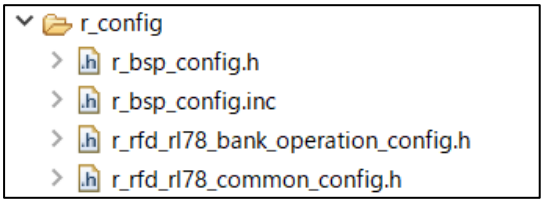
(5) Press a “Generate Code” button and generate the code.



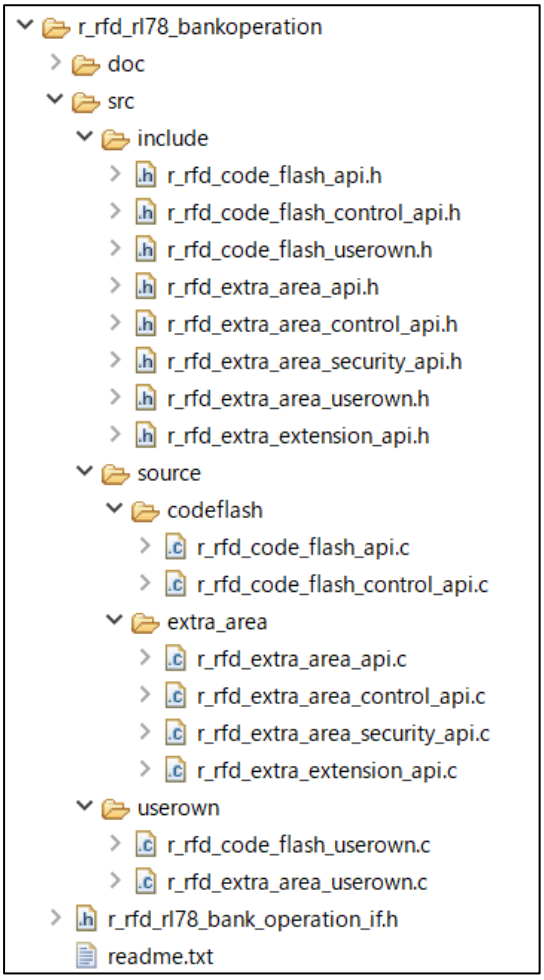
(6) The “r_rfd_rl78_common” folder and the “r_rfd_rl78_bankoperation” folder are added to the project tree.



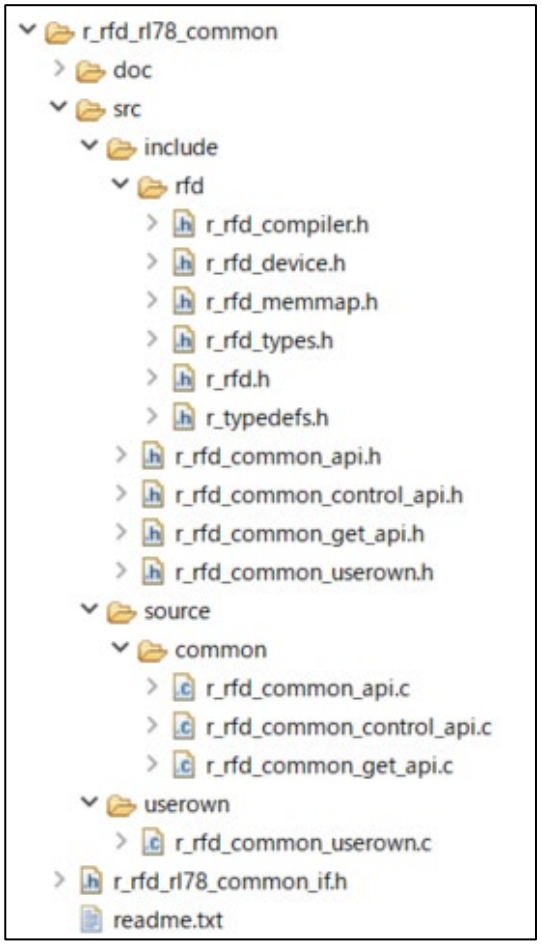
Each folder is developed as follows.



The developed r_config folder



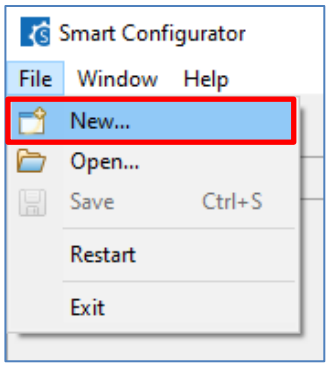
The developed r_rfd_rl78_bankoperation folder



The developed r_rfd_rl78_common folder

2.2.3 In Case of IAR EW for Renesas RL78

(1) Select “File” [New...] after starting Smart Configurator for RL78.



(2) Select the item of “Platform” and “Toolchain”.

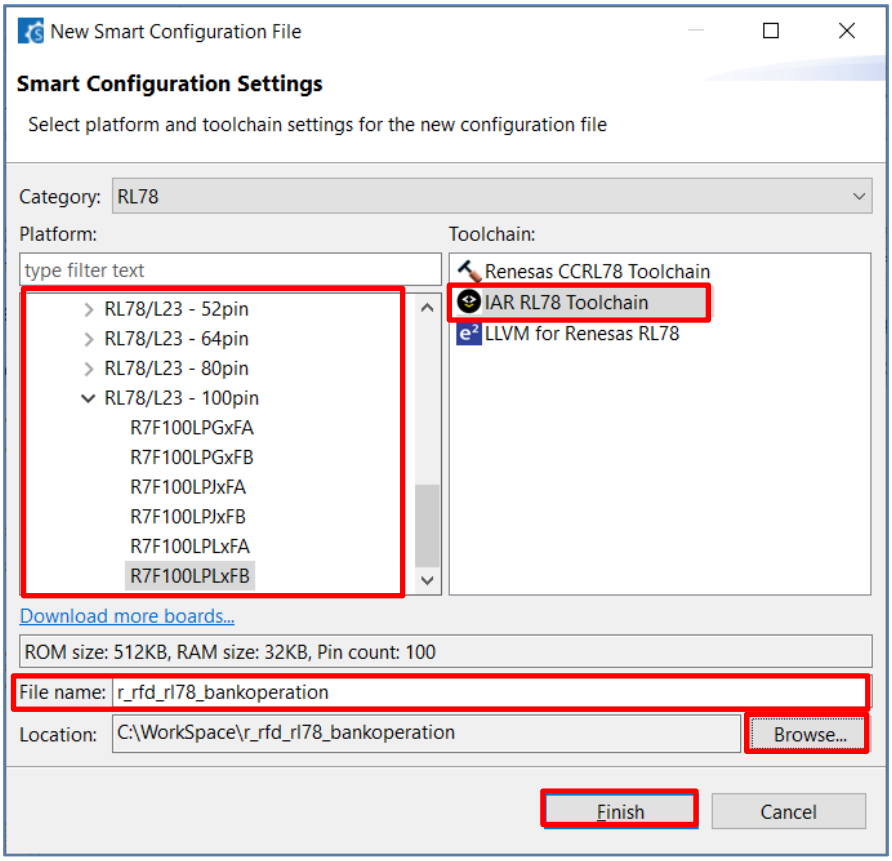
Select the same device as the device selected in “the project of IAR EW for Renesas RL78” by “Platform”.

Select “IAR RL78 Toolchain” as “Toolchain”.

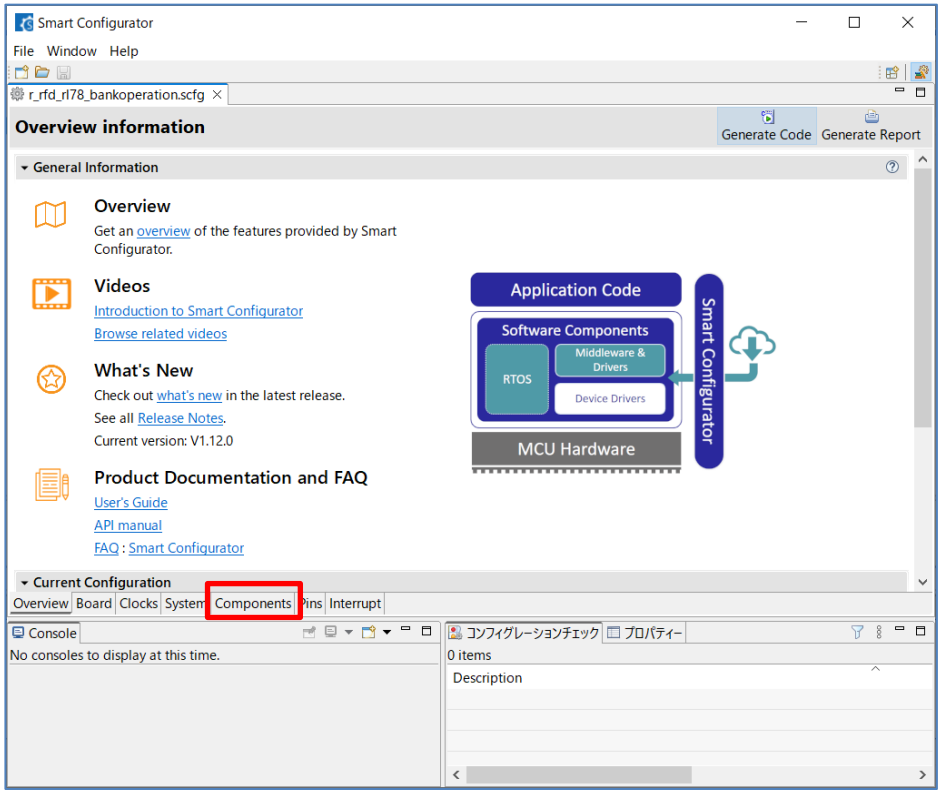
Input arbitrary names into “File name”.

Press the “Browse...” button and set the location of the project folder for IAR EW for Renesas RL78. And press a “Finish” button.

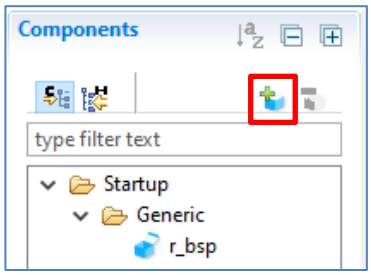
The “.setting” folder and the “<file name>.scfg” file are created to the set location.



(3) Select a “Components” tab.

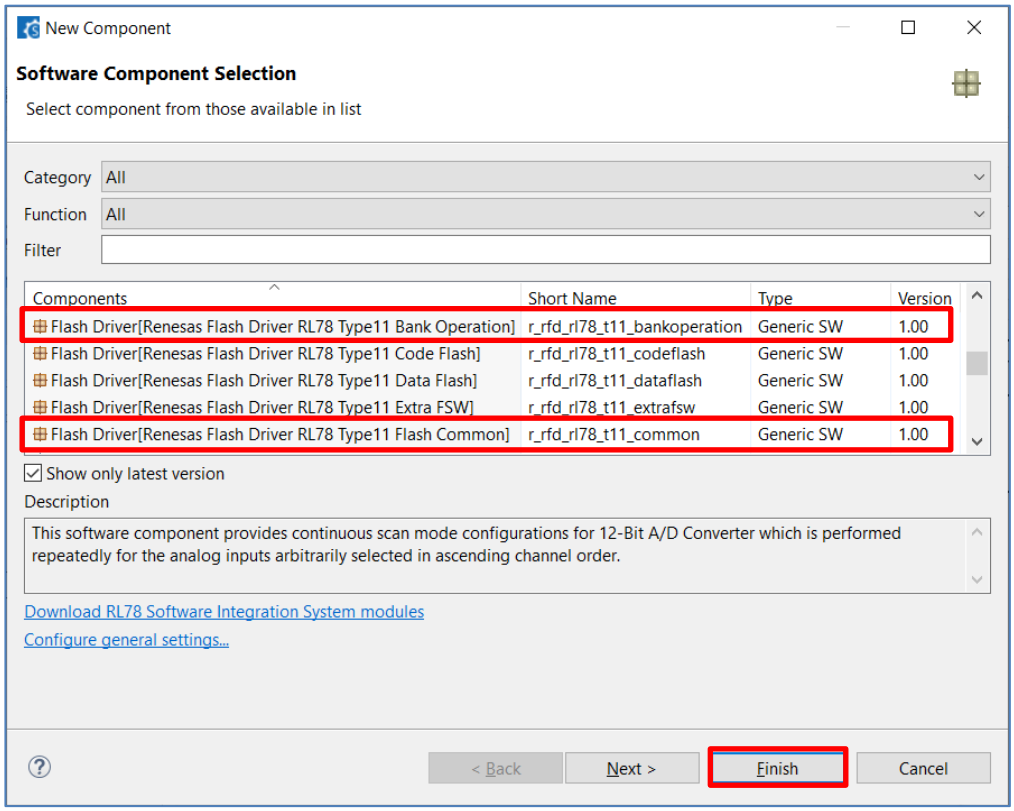


(4) Press the “Add component” button and open the “New Component” dialog.

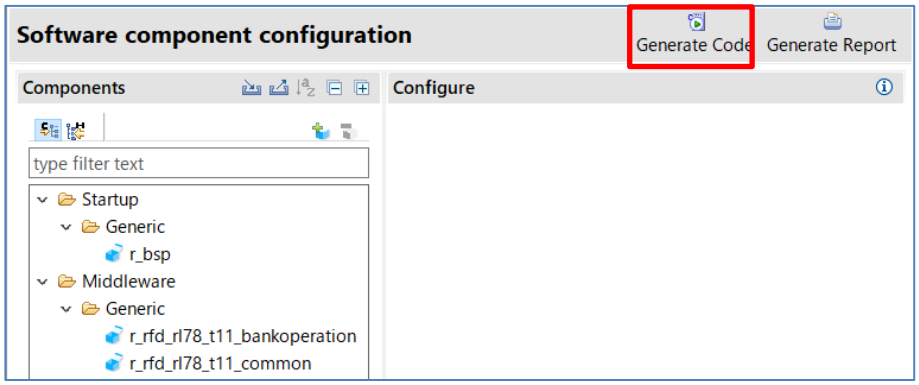


(5) Select the following components and press a “Finish” button.

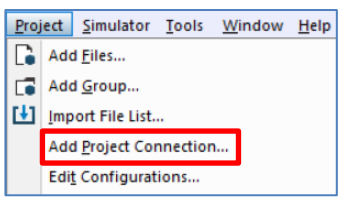
- Flash Driver[Renesas Flash Driver RL78 Type11 Bank Operation]
(r_rfd_rl78_t11_bankoperation)
- Flash Driver[Renesas Flash Driver RL78 Type 11 Flash Common](r_rfd_rl78_t11_common)



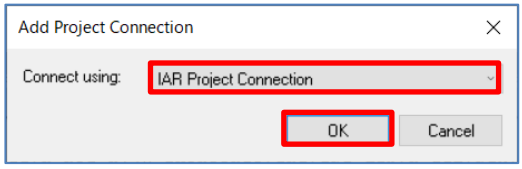
(6) Press a “Generate Code” button and close “Smart Configurator” after the completion of generation for the code.



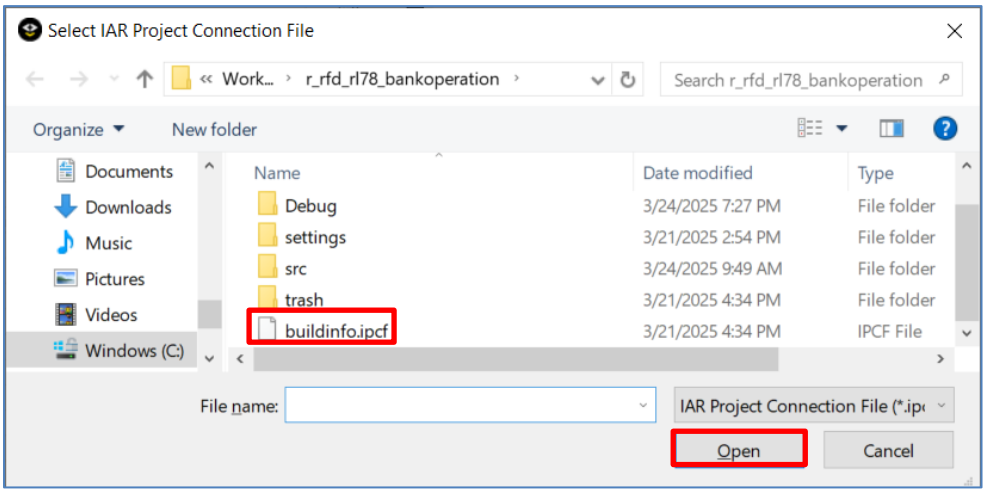
(7) Start IAR EW for Renesas RL78. And select “Project” menu [Add Project Connection], and open the “Add Project Connection” dialog.



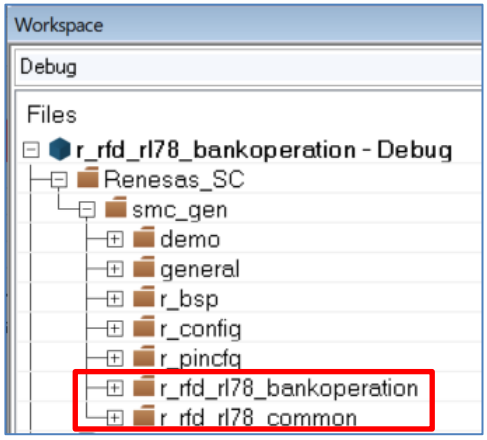
(8)Select “IAR Project Connection”, and press an “OK” button.



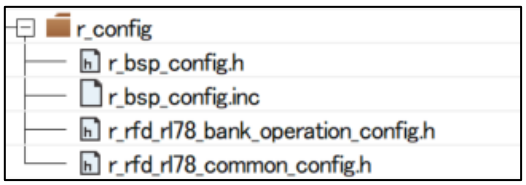
(9) Select the ipcf file created by Smart Configurator and press an “Open” button.



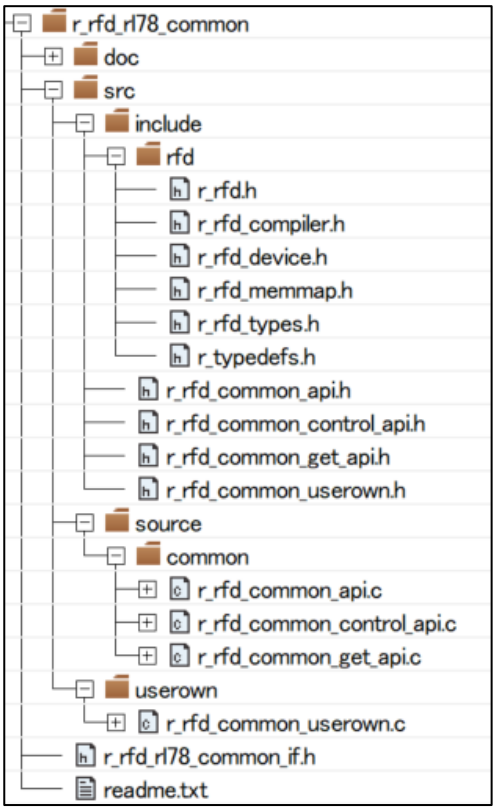
(10) “r_rfd_rl78_common” and “r_rfd_rl78_bankoperation” are added to Workspace.



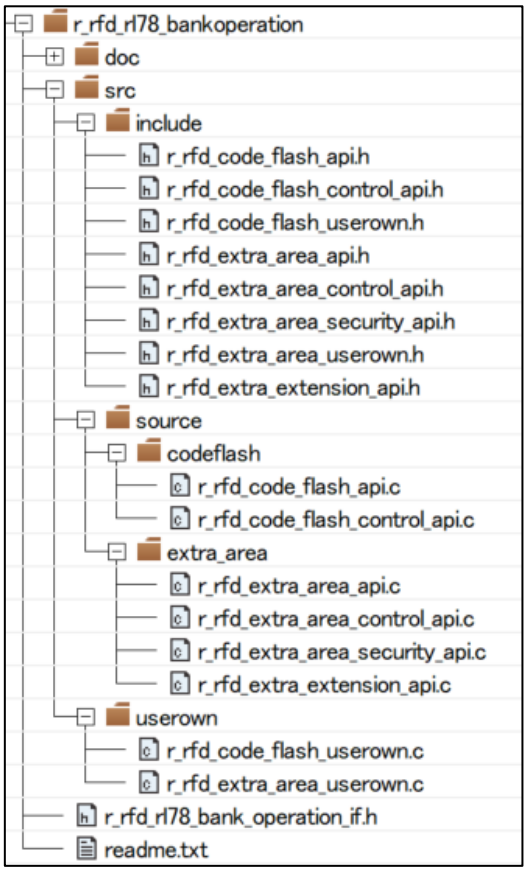
Each folder is developed as follows.



The developed r_config folder



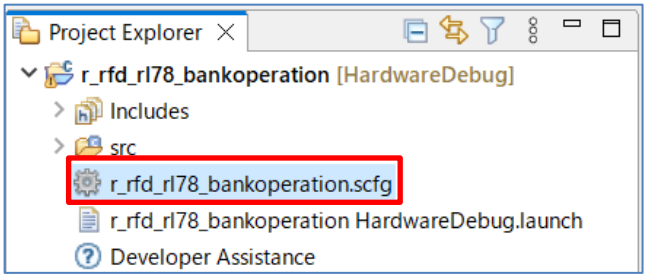
The developed r_rfd_rl78_common folder



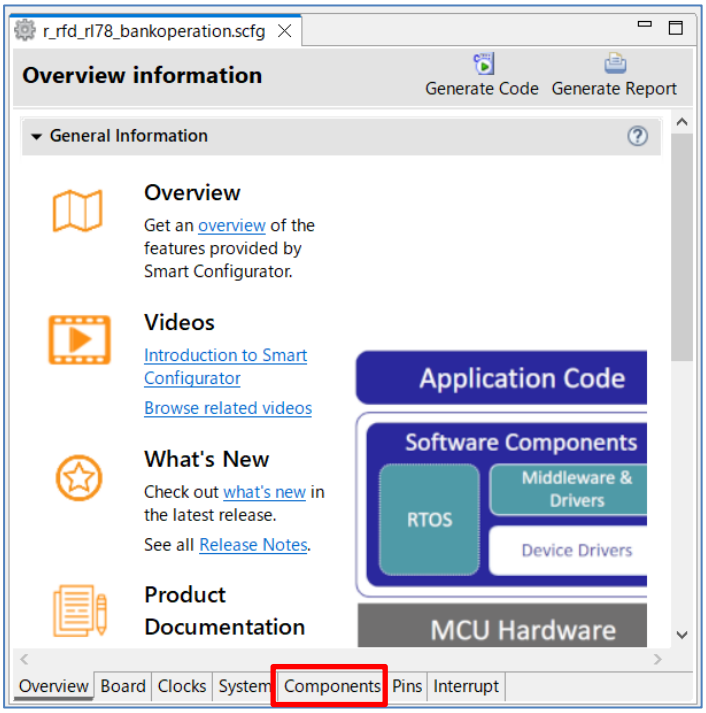
The developed r_rfd_rl78_bankoperation folder

2.2.4 In Case of e² studio(LLVM)

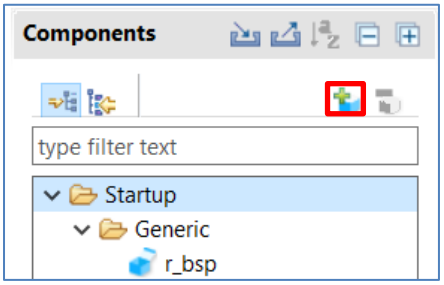
(1) Open the project file of “Smart Configurator” after starting e² studio.



(2) Select a “Components” tab.

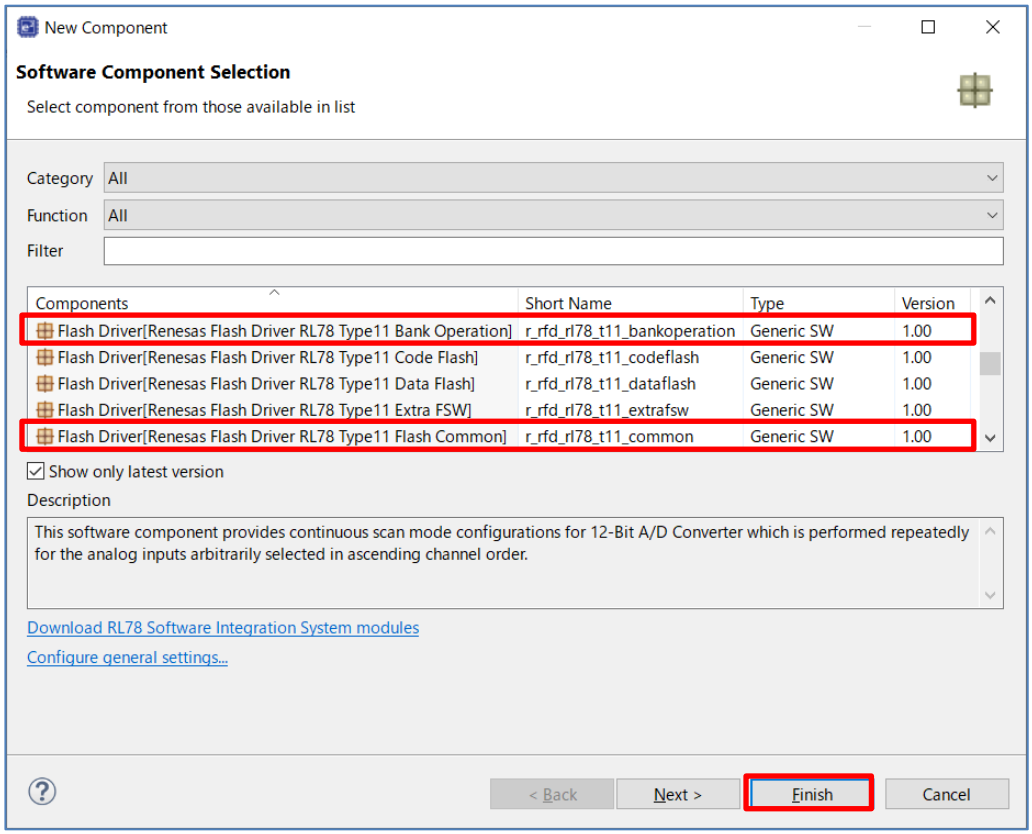


(3) Press the “Add component” button and open the “New Component” dialog.

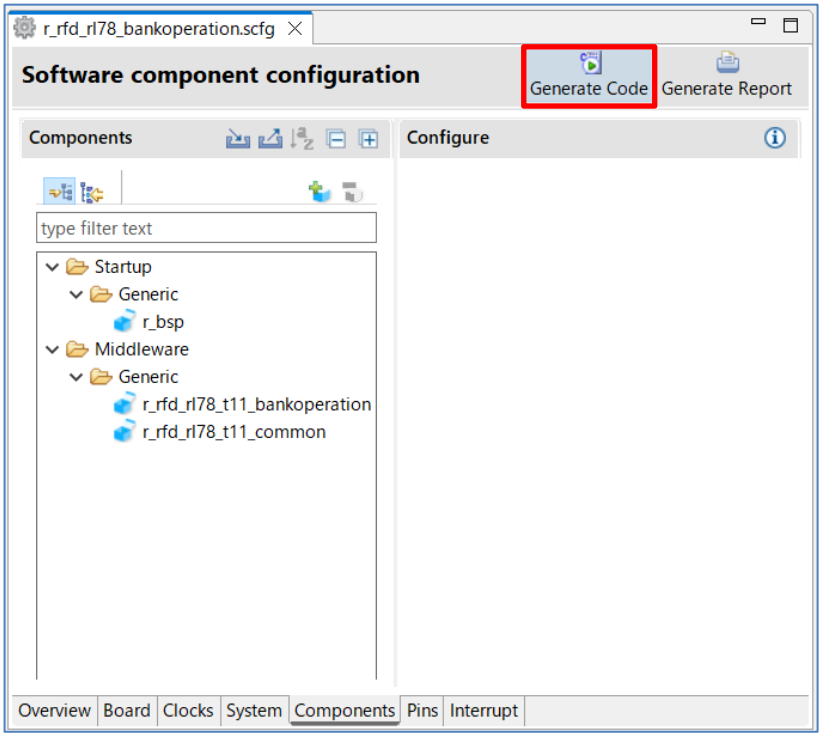


(4) Select the following components and press a “Finish” button.

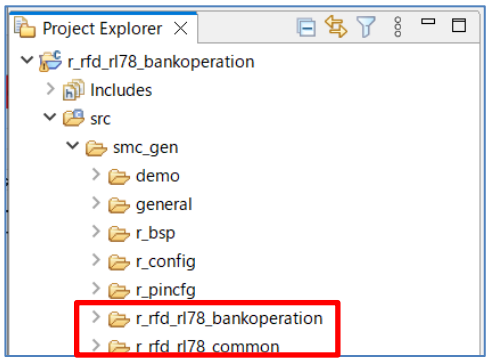
- Flash Driver[Renesas Flash Driver RL78 Type11 Bank Operation]
(r_rfd_rl78_t11_bankoperation)
- Flash Driver[Renesas Flash Driver RL78 Type 11 Flash Common](r_rfd_rl78_t11_common)



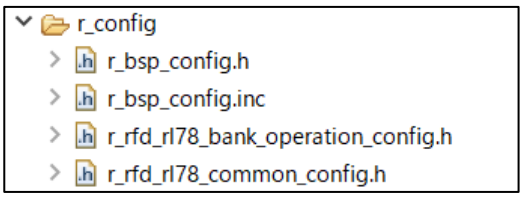
(5) Press a “Generate Code” button and generate the code.



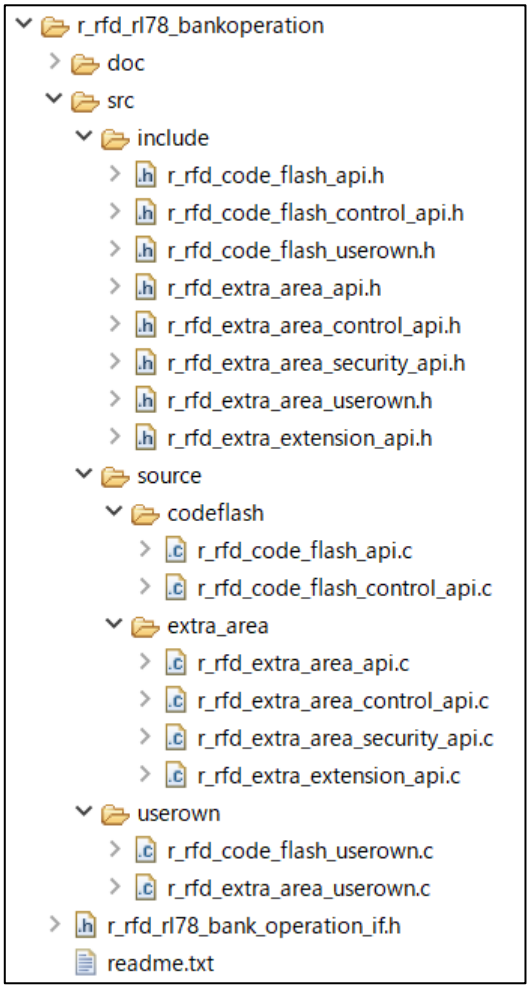
(6) The “r_rfd_rl78_common” folder and the “r_rfd_rl78_bankoperation” folder are added to the project tree.



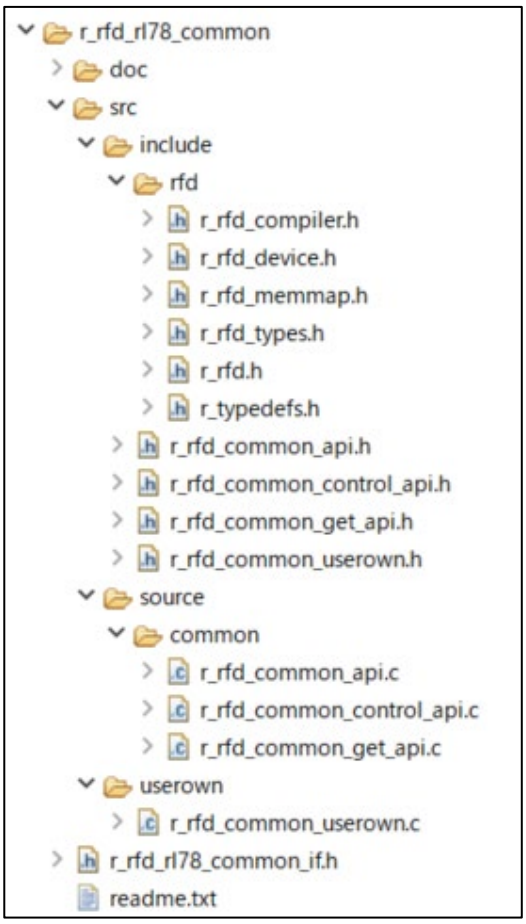
Each folder is developed as follows.



The developed r_config folder



The developed r_rfd_rl78_bankoperation folder



The developed r_rfd_rl78_common folder

2.3 Project registration of sample program

(1) Extract “BP_SWAP_sample.zip.”

The common file duplicates in the case which uses it at the same time with the sample program of a code flash or a data flash area. Extract to overwrite both as the same folder name.

(2) Register the folder of the sample program into the project of CS+, e^2 studio, or IAR.

* Files included in the folder other than the compiler package used, do not need to be registered.

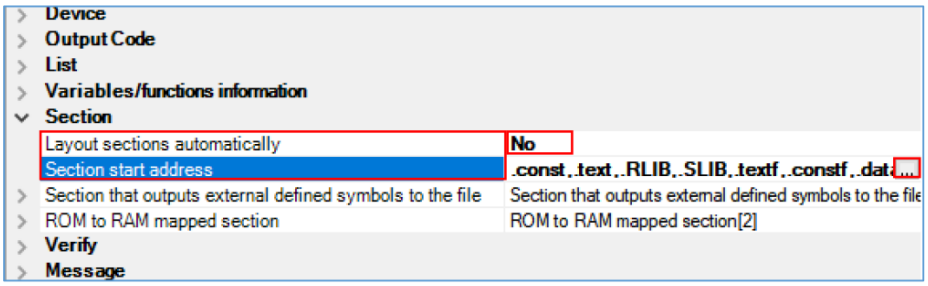
(3)Setting the section items

(3-1) Setting of the section items on CS+

Setting of the section Items on CS+ inputs in the “Link Options” tab. (Common in each area)

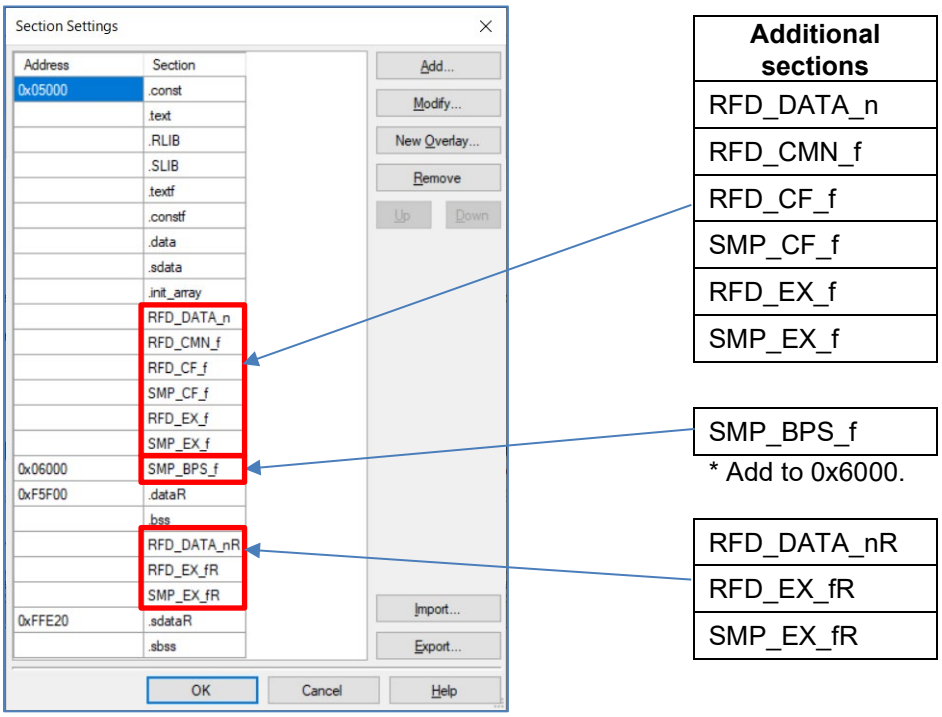
- Setting the [Section] items

Set “No” to [Layout sections automatically]. And sections come to be displayed on [Section start address]. Press the “ ... ” button of the right-hand side which sections are displaying, and a “Section Settings” screen is displayed.



Add sections necessary for extra area(FSW) reprogramming on a “Section Settings” screen.

Add to the program area : RFD_DATA_n, RFD_CMN_f, RFD_CF_f, SMP_CF_f, RFD_EX_f, SMP_EX_f
Program code after bank swap execution : SMP_BPS_f
Add to the RAM area : RFD_DATA_nR, RFD_EX_fR, SMP_EX_fR



Additional sections
RFD_DATA_n
RFD_CMN_f
RFD_CF_f
SMP_CF_f
RFD_EX_f
SMP_EX_f

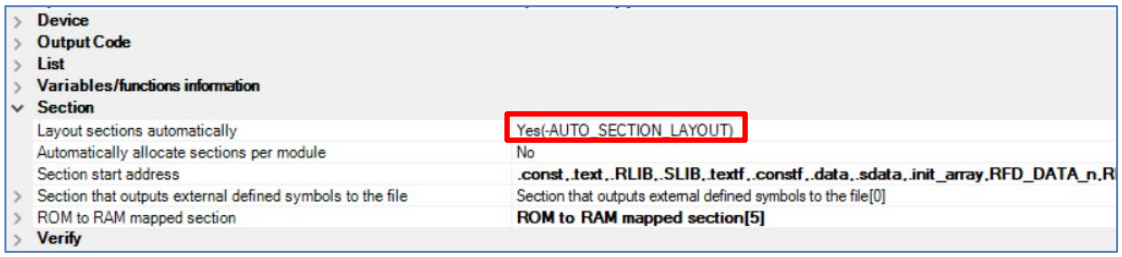
SMP_BPS_f

* Add to 0x6000.

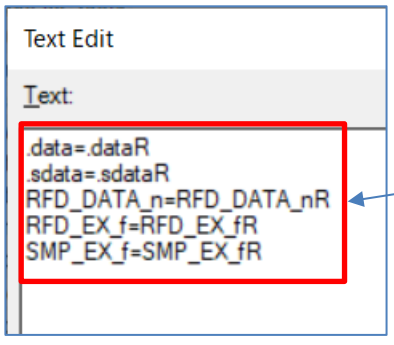
RFD_DATA_nR
RFD_EX_fR
SMP_EX_fR

Note: About setting of the sample project in the case of using each device, refer to the “Setting related to changing devices” section of the “Creating a Sample Project for RFD RL78 Type 11” chapter after “RL78 Family Renesas Flash Driver RL78 Type 11 User’s Manual(R20UT5539)” Rev.1.00 or later.

Be sure to return [Layout sections automatically] to “Yes”, after pressing the “OK” button.



Press the right-hand side “ ... ” button by [ROM to RAM mapped section], display the “Text Edit” screen, and add the section for copying to RAM from ROM.

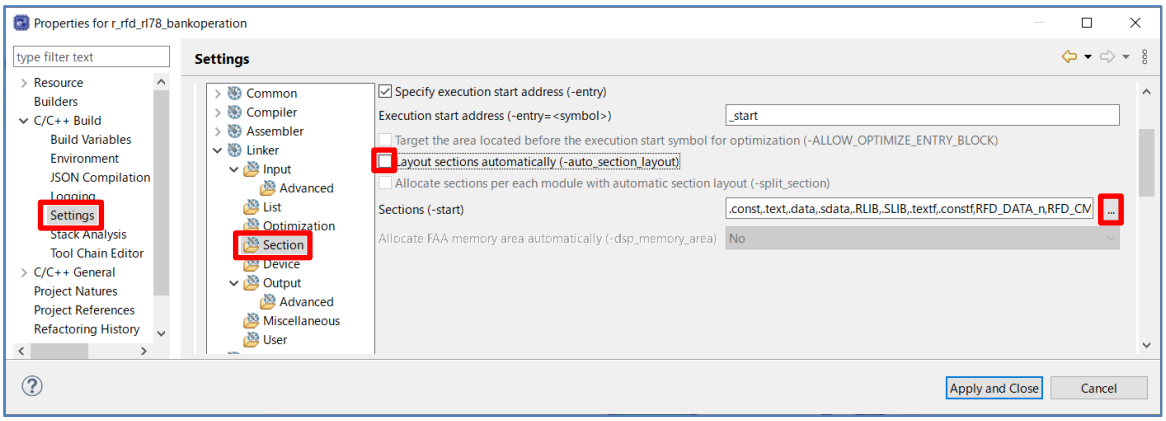


ROM to RAM mapped section
.data=.dataR
.sdata=.sdataR
RFD_DATA_n=RFD_DATA_nR
RFD_EX_f=RFD_EX_fR
SMP_EX_f=SMP_EX_fR

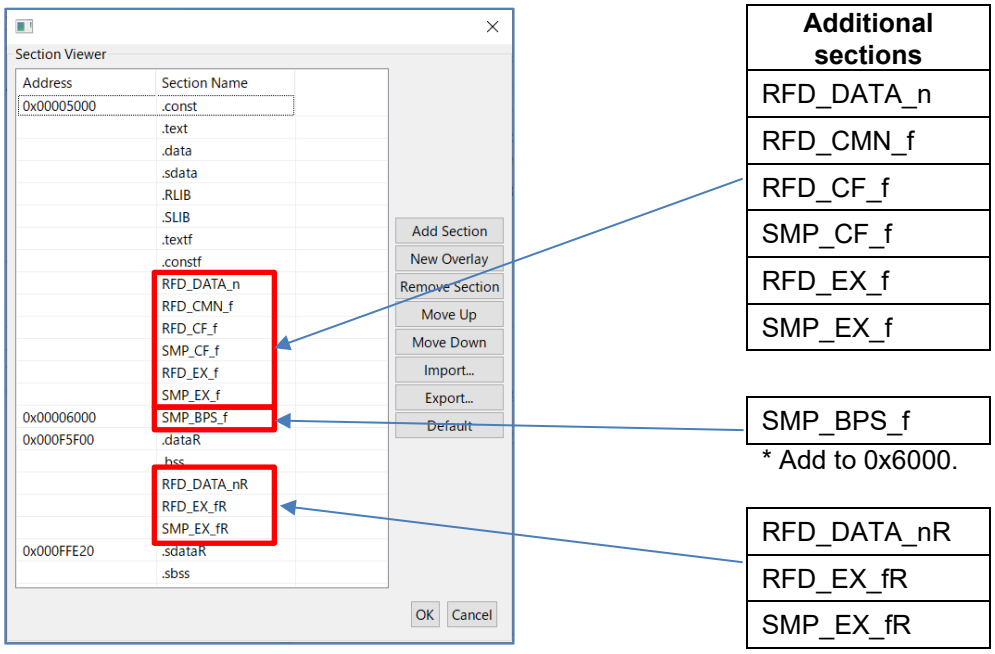
(3-2) Setting of the section items on e² studio(CC-RL)

On e² studio(CC-RL), sections are set on a “Section Viewer” screen.

The process which displays a “Section Viewer” screen is shown. Select the [Properties] from the [Project] menu and open the window of the properties for a target project. And select “Linker” [Section] from [Settings] of “C/C++ Build”. Remove a check mark from [Layout sections automatically] (-auto_section_layout), and press the “ ” button positioned in the rightmost of [Sections (-start)], and display Section Viewer.

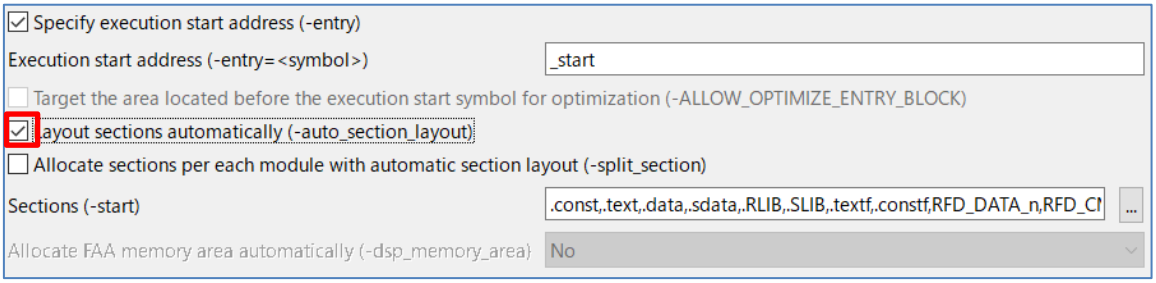


Add to the program area : RFD_DATA_n, RFD_CMN_f, RFD_CF_f, SMP_CF_f, RFD_EX_f, SMP_EX_f
Program code after bank swap execution : SMP_BPS_f
Add to the RAM area : RFD_DATA_nR, RFD_EX_fR, SMP_EX_fR

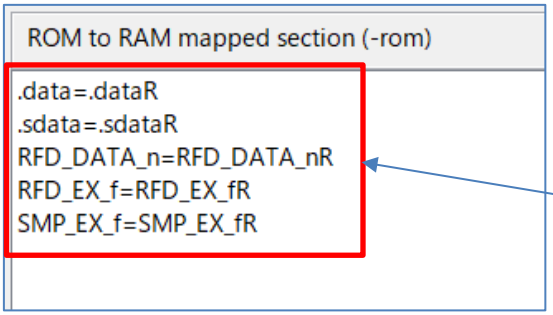


Note: About setting of the sample project in the case of using each device, refer to the “Setting related to changing devices” section of the “Creating a Sample Project for RFD RL78 Type 11” chapter after “RL78 Family Renesas Flash Driver RL78 Type 11 User’s Manual(R20UT5539)” Rev.1.00 or later.

Be sure to put a check mark to [Layout sections automatically (-auto_section_layout)], after pressing the “OK” button.



Select “C/C++ Build” [Setting] - “Linker” [Section], display the “ROM to RAM mapped section (-rom)” screen, and add the section for copying to RAM from ROM.



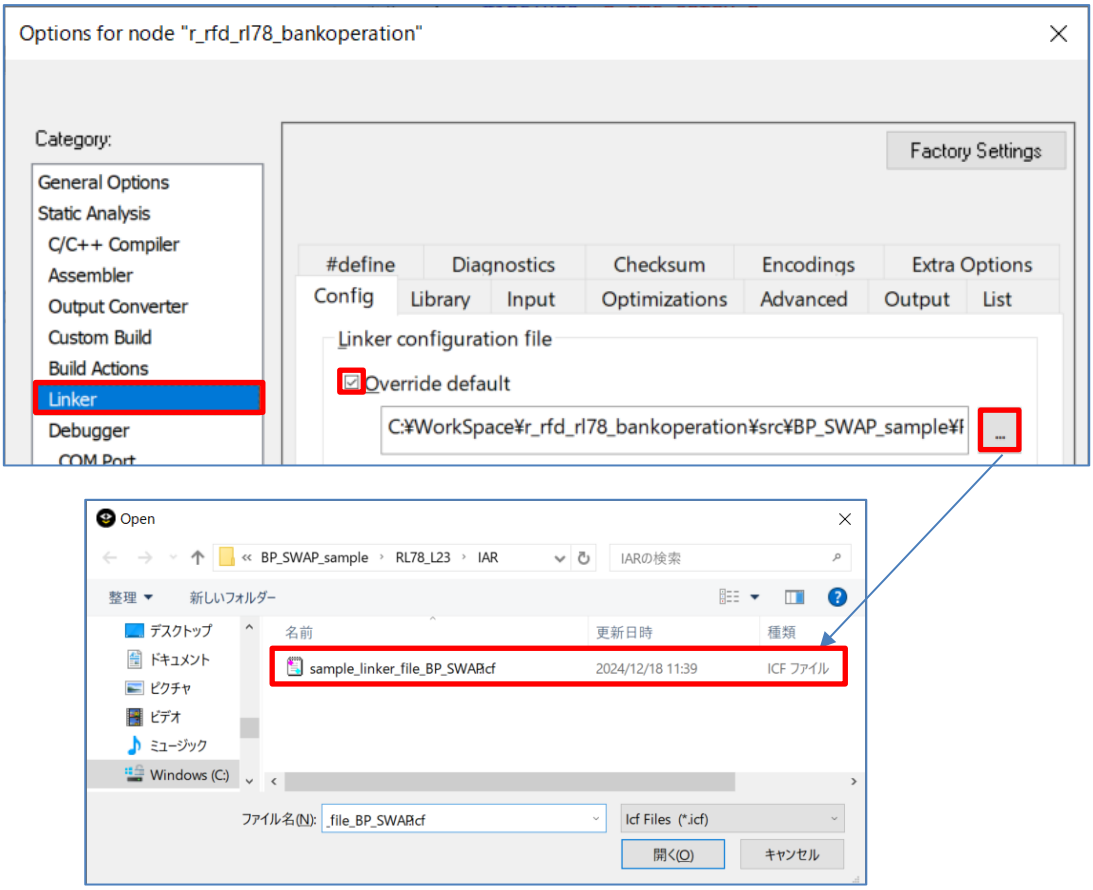
ROM to RAM mapped section
.data=.dataR
.sdata=.sdataR
RFD_DATA_n=RFD_DATA_nR
RFD_EX_f=RFD_EX_fR
SMP_EX_f=SMP_EX_fR

(3-3) Setting of the section items on IAR EW for Renesas RL78

On IAR Embedded Workbench, Linker configuration file (*. icf) describes link setting executed by building. Select “Options” by the click right mouse button of project with tree. Select [Linker] by “Category” in the displayed window, And put a check mark to “Override default” of the [Config] tab. Select Linker configuration file (*. icf) in the “Open” window of “ ... ” button. Select the “sample_linker_file_(area name).icf” file prepared for RFD RL78 Type 11. Linker configuration file (*. icf) for every reprogramming area is as follows.

- For bank programming / bank swapping control : sample_linker_file_BP_SWAP.icf

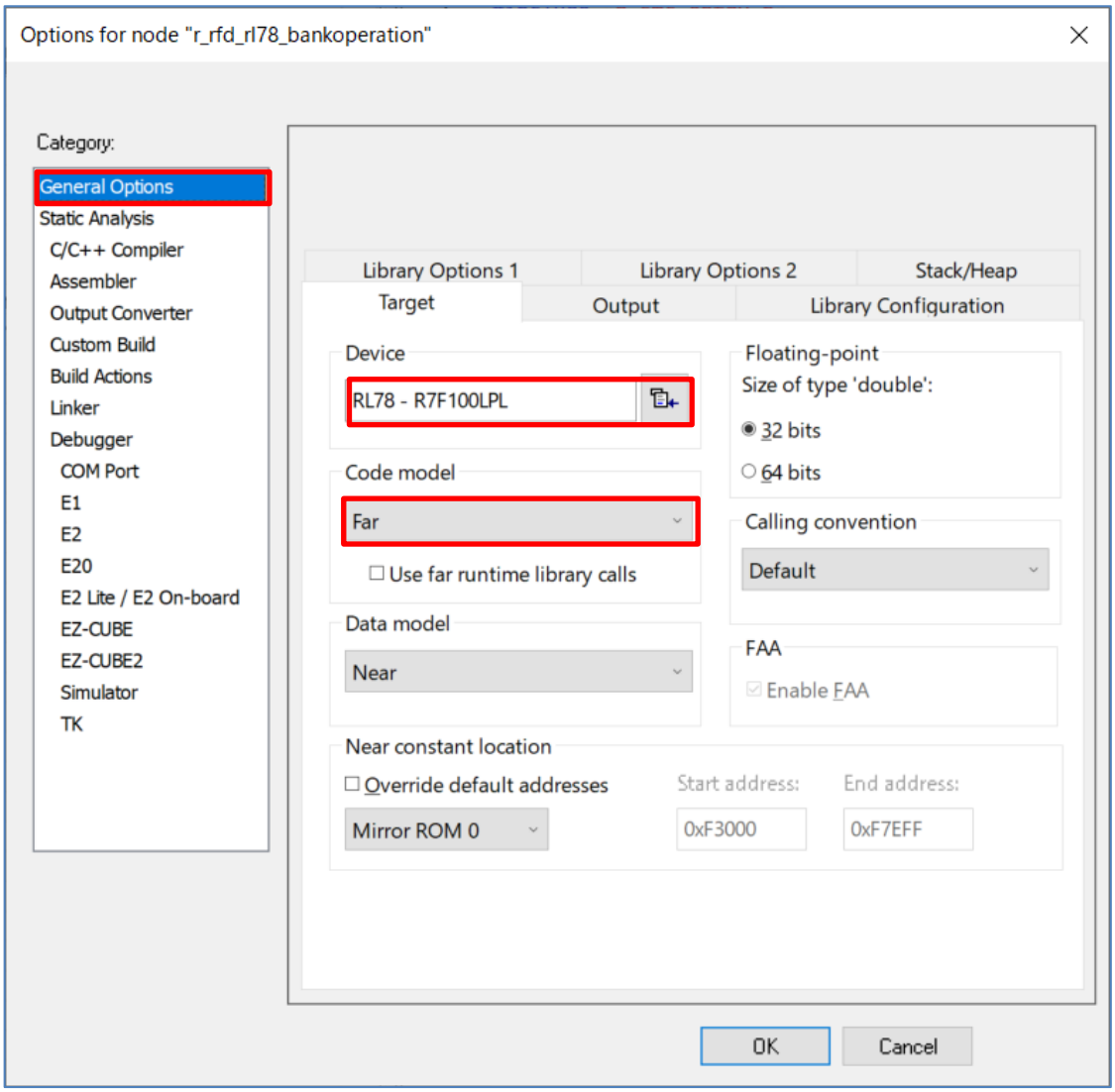
RL78/L23:(\BP_SWAP_sample\RL78_L23\IAR\)



* On the case used at the same time with code flash area or data flash area, it is necessary to modify the icf file suitable for the sample program for the area used.

Note: About setting of the sample project in the case of using each device, refer to the “Setting related to changing devices” section of the “Creating a Sample Project for RFD RL78 Type 11” chapter after “RL78 Family Renesas Flash Driver RL78 Type 11 User’s Manual(R20UT5539)” Rev.1.00 or later.

Set the items of [General Options] - [Target] tab in the “Options” screen. Select the target device for [Device] and “Far” for [Code model].

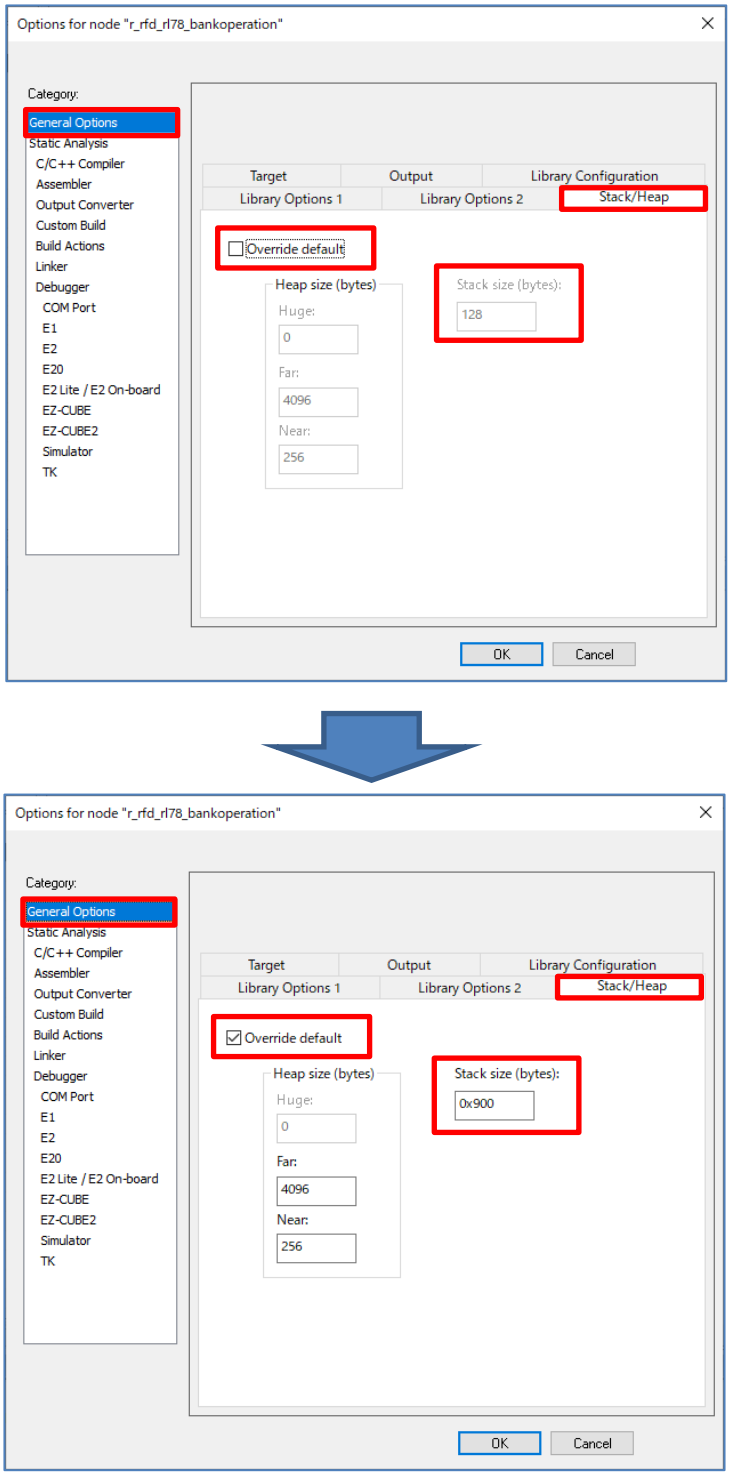


(3-3-1) Stack setting

The initial value of the stack size of IAR Embedded Workbench for RL78 is 128 bytes. When the stack used by a user program and RFD RL78 Type11 exceeds this size, it is necessary to modify use stack size.

The code flash reprogramming for bank programming / bank swapping control obtains the buffer for 1-block(2 KB) data to a stack. Therefore, the stack size about 0x900(2048+256)bytes is necessary.

<<The example of a stack setting>>

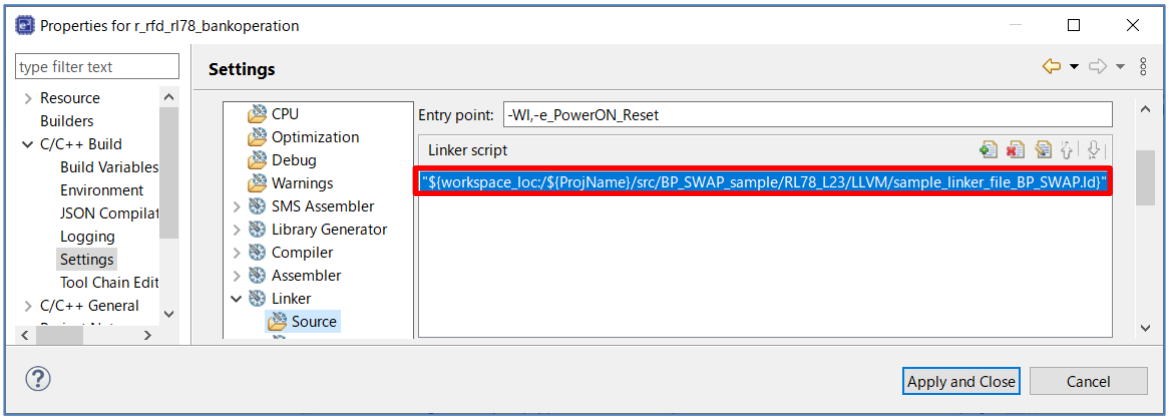


(3-4) Setting of the section items on e^2 studio(LLVM)

On e^2 studio (LLVM), Linker script (*. ld) describes link setting executed by building. Select the [Property] in a project, and open a Properties window. Input the linker script directory path in the window displayed by selection of “C/C++” build [Setting] - “Linker” [Source].

That linker script path to add is shown below.

${workspace_loc:/${ProjName}/src/BP_SWAP_sample/RL78_L23/LLVM/sample_linker_file_BP_SWAP.ld}



* It is an example of the include path in the case which extracted BP_SWAP_sample.zip directly under the src folder.

Note: About setting of the sample project in the case of using each device, refer to the “Setting related to changing devices” section of the “Creating a Sample Project for RFD RL78 Type 11” chapter after “RL78 Family Renesas Flash Driver RL78 Type 11 User’s Manual(R20UT5539)” Rev.1.00 or later.

(4) Include Path Settings

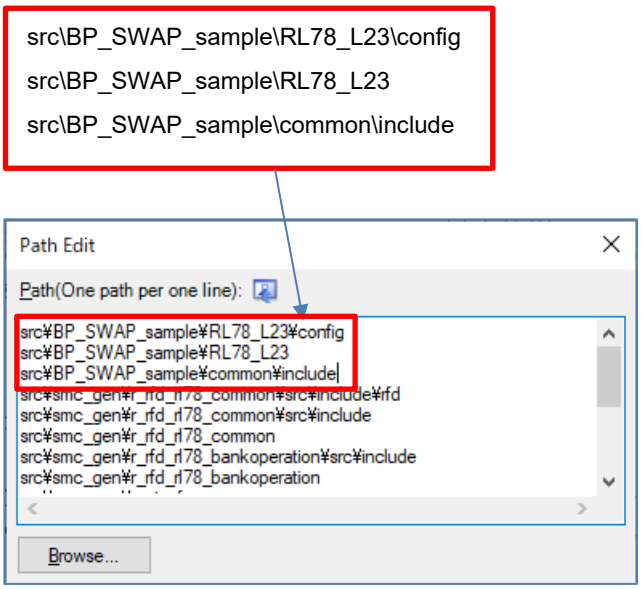
(4-1) Setting of the include path on CS+

Setting of the include path on CS+ inputs path in “Common Options” tab.

- Add the Include directory path in the “Path Edit” window displayed by selection of [Frequently Used Options(for Compile)] - [Additional include paths].

In the phase where code generation was performed by (5) of “2.1.1 In Case of CS+”, the include path of files other than the sample program is registered. For a reason, it needs to register the include path of the sample program.

Those include path to add is shown below.



* It is an example of the include path in the case which extracted BP_SWAP_sample.zip directly under the src folder.

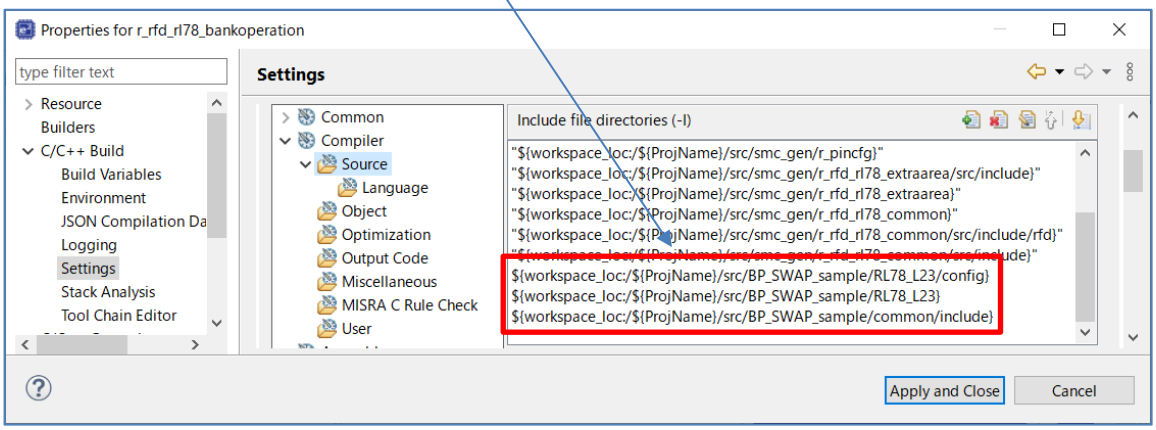
(4-2) Setting of the include path on e^2 studio(CC-RL)

Setting of the include path on e^2 studio(CC-RL) inputs path in “Properties” window.

- Input the Include directory path in the window displayed by selection of “C/C++” build [Setting] - “Compiler” [Source].

Those include path to add is shown below.

```
${workspace_loc:/${ProjName}/src/BP_SWAP_sample/RL78_L23/config}
${workspace_loc:/${ProjName}/src/BP_SWAP_sample/RL78_L23}
${workspace_loc:/${ProjName}/src/BP_SWAP_sample/common/include}
```



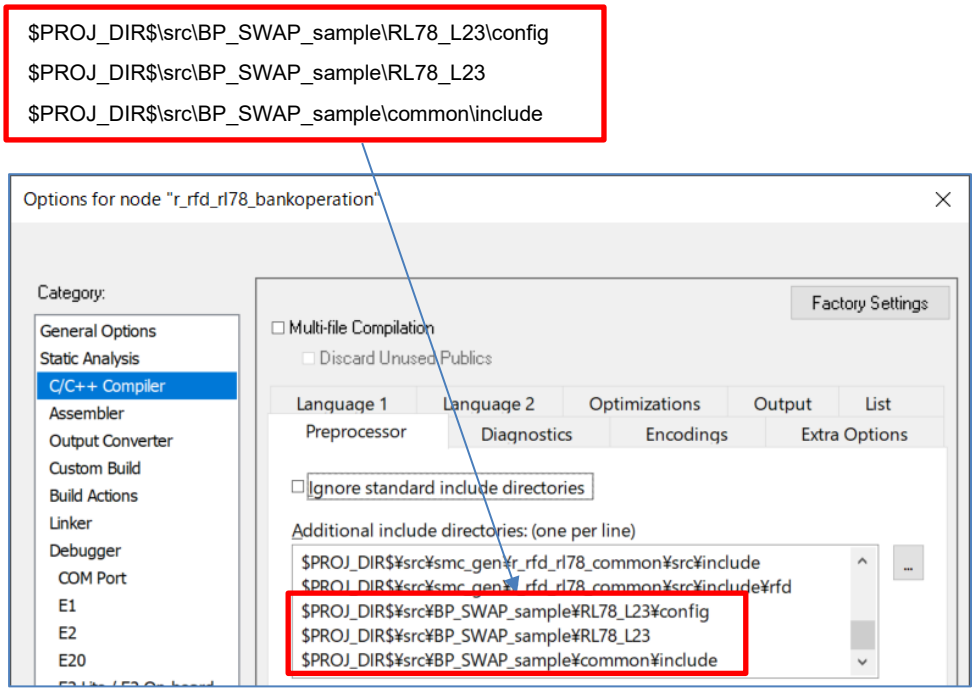
* It is an example of the include path in the case which extracted BP_SWAP_sample.zip directly under the src folder.

(4-3) Setting of the include path on IAR EW for Renesas RL78

Setting of the include path on IAR Embedded Workbench selects “C/C++ Compiler” of “Category”, and inputs path in “Preprocessor” tab.

- Input the Include directory path in the “Edit include Directories” window displayed by selection of [Additional include directories: (one per line)].

Those include path to add is shown below.



* It is an example of the include path in the case which extracted BP_SWAP_sample.zip directly under the src folder.

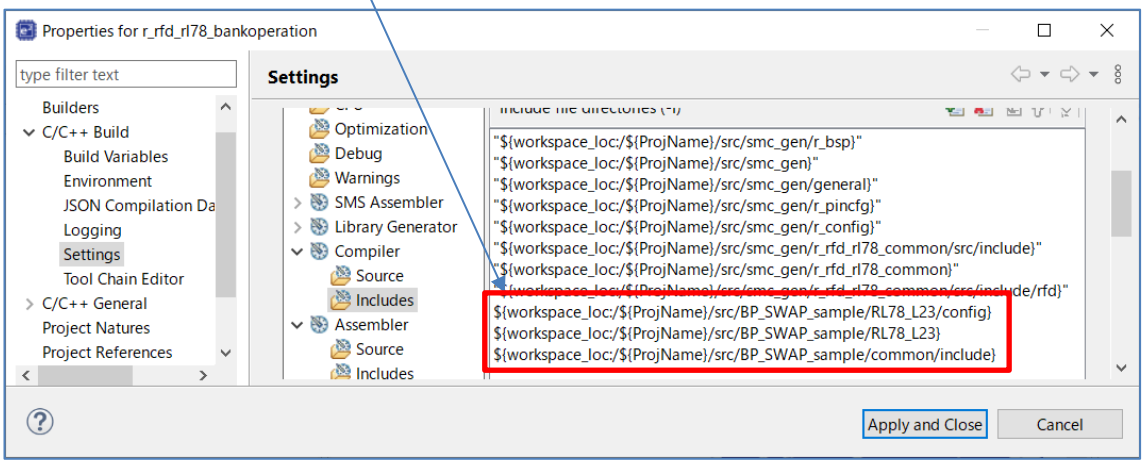
(4-4) Setting of the include path on e^2 studio(LLVM)

Setting of the include path on e^2 studio(LLVM) inputs path in “Properties” window.

- Input the Include directory path in the window displayed by selection of “C/C++” build [Setting] - “Compiler” [Includes].

Those include path to add is shown below.

```
${workspace_loc:/${ProjName}/src/BP_SWAP_sample/RL78_L23/config}
${workspace_loc:/${ProjName}/src/BP_SWAP_sample/RL78_L23}
${workspace_loc:/${ProjName}/src/BP_SWAP_sample/common/include}
```



* It is an example of the include path in the case which extracted BP_SWAP_sample.zip directly under the src folder.

(5) The setting of user definition macro

(5-1) Setting of the user definition macro on CS+

Refer to the chapter of “The Setting of User Definition Macro (CC-RL Compiler)” of Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539).

(5-2) Setting of the user definition macro on e^2 studio(CC-RL)

Refer to the chapter of “The Setting of User Definition Macro (CC-RL Compiler)” of Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539).

(5-3) Setting of the user definition macro on IAR EW for Renesas RL78

Refer to the chapter of “The Setting of User Definition Macro (IAR Compiler)” of Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539).

(5-4) Setting of the user definition macro on e^2 studio(LLVM)

Refer to the chapter of “The Setting of User Definition Macro (LLVM Compiler)” of Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539).

(6) Device Item Settings

(6-1) Setting of the device Items on CS+

Refer to the chapter of “Device Item Settings” of Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539).

(6-2) Setting of the device Items on e^2 studio(CC-RL)

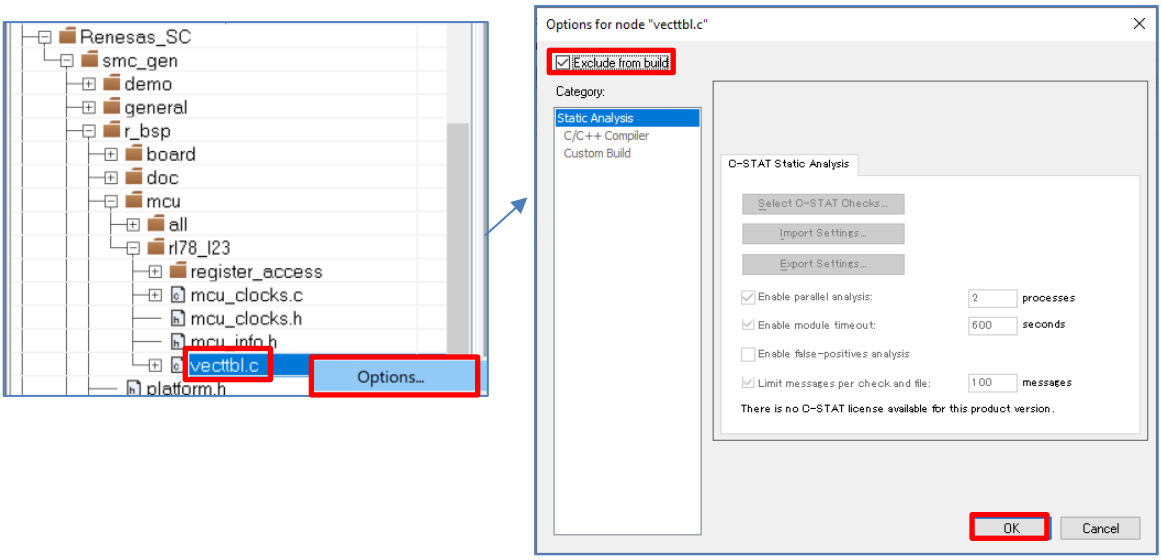
Refer to the chapter of “Device Item Settings” of Renesas Flash Driver RL78 Type 11 user’s manual (R20UT5539).

(6-3) Setting of the device Items on IAR EW for Renesas RL78

Build including vecttbl.c prepared as a sample program. A user option byte’s value is set to 0x6EFFE8, and an on-chip debugging option byte’s value is set to 0x85.

When the code is generated by Smart Configurator, vecttbl.c is generated to a “smc_gen\r_bsp\mcu\rl78_l23\” folder. And because vecttbl.c duplicates, it is necessary to repeal this file.

Right-click a mouse by “Renesas_SC\smc_gen\r_bsp\mcu\rl78_l23\vecttbl.c” in the [project] on a tree. And select an “option” and set a “check” to [Exclude from build] in the displayed screen. And vecttbl.c is excluded from build target.



(6-4) Setting of the device Items on e^2 studio(LLVM)

Build including vects.c prepared as a sample program. A user option byte’s value is set to 0x6EFFE8, and an on-chip debugging option byte’s value is set to 0x85.

When the code is generated by Smart Configurator, r_cg_vect_table.c is generated to a “src\general\” folder. And because r_cg_vect_table.c and vects.c duplicate the settings, it is necessary to repeal this file.

Right-click a mouse by “src\general\r_cg_vect_table.c” in the [project] on a Project Explorer. And select [Resource Configuration] - [Exclude from build...] and use it to exclude r_cg_vect_table.c from build target.

(7) Execute the sample program from a main function.

Describe the sample_bankoperation_main function included in r_flash_sample_bankoperation_rl78l2x.c like the “main function” for the project. And build, download and execute it.

* The header file which described the prototype declaration for sample_bankoperation_main function. Include prepared “r_flash_sample_bankoperation_rl78l2x.h.”

2.4 The Check of Operation for Sample Program

Bank programming control writes the reset vector, the Option Bytes, and program code after bank swap execution in the rewrite bank. Then, bank swap control replaces the write bank and the startup bank. Confirm this operation by the following methods.

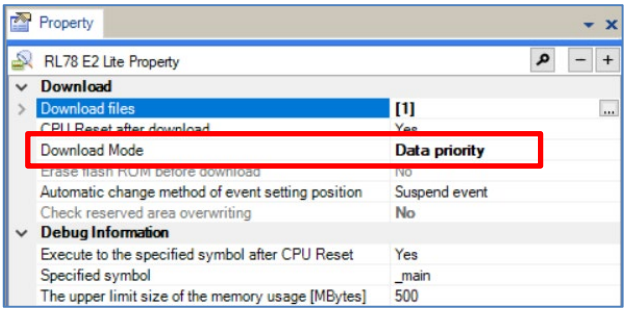
Note: To use the boot swapping or bank swapping function, make settings for bank swapping in the BTBLS bits. The setup using a dedicated flash memory programmer beforehand. Refer to the user's manual of RL78 target micro controller for the details of the setting method.

[ex BTBLS = 07H Bank size = 256KB (R7F100LxL) / 128KB (R7F100LxJ)]

2.4.1 In Case of CS+

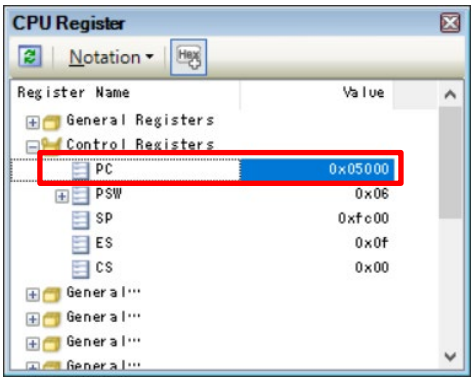
(1) Write an object file to the device, run it stand-alone, and then connect it to the emulator.

(2) On the project, select "Download File Settings" tab from "RL78 E2 [Lite] (Debug Tool)", and select "Datapriority" to "Download" - "Download Mode".



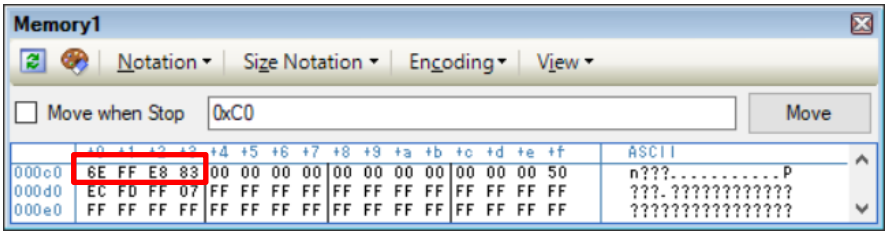
(3) Select [Debug]menu - [Connect to Debug Tool].

(4) Select [View]menu - [CPU Register] and display CPU Register window. Confirm that the program counter is 0x5000 at reset.



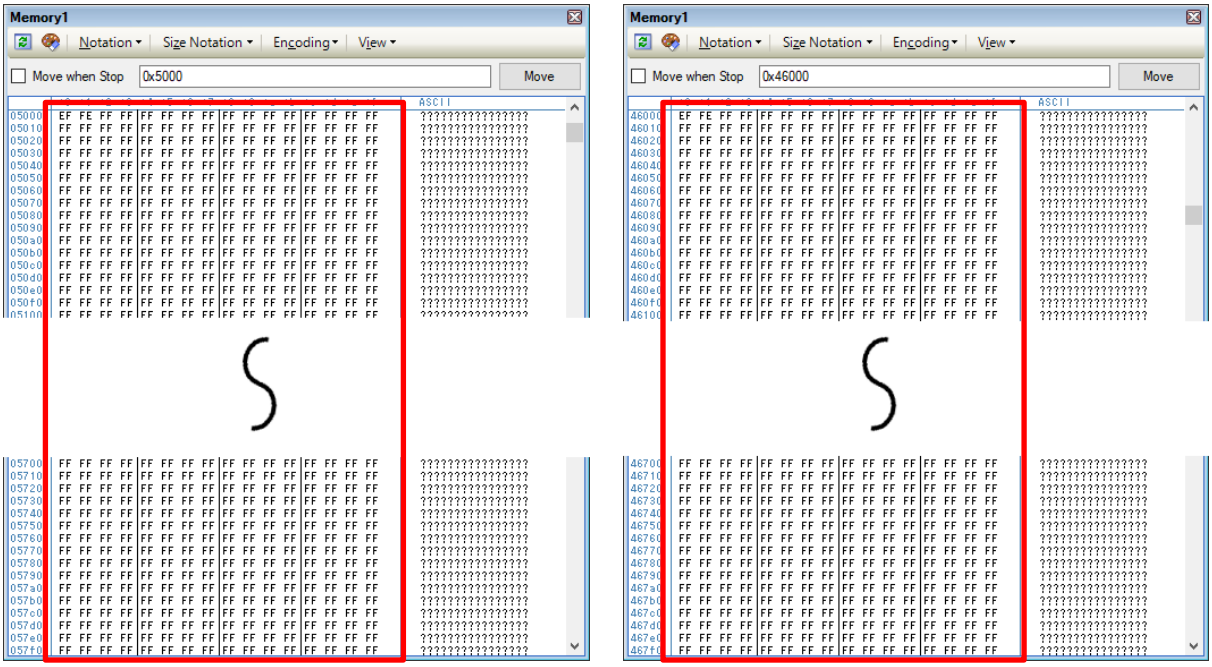
(5) After selecting [View]menu - [Memory], select "Memory1", "Memory2", "Memory3", or "Memory4", and display a memory window.

(6) Confirm that the Option Bytes (0xC0 to 0xC3) are 0x6E, 0xFF, 0xE8, and 0x85.



Note: Since bit 3-1 of the On-Chip Debug Option Byte is undefined, make sure that the values of all bits except bit 3-1 match.

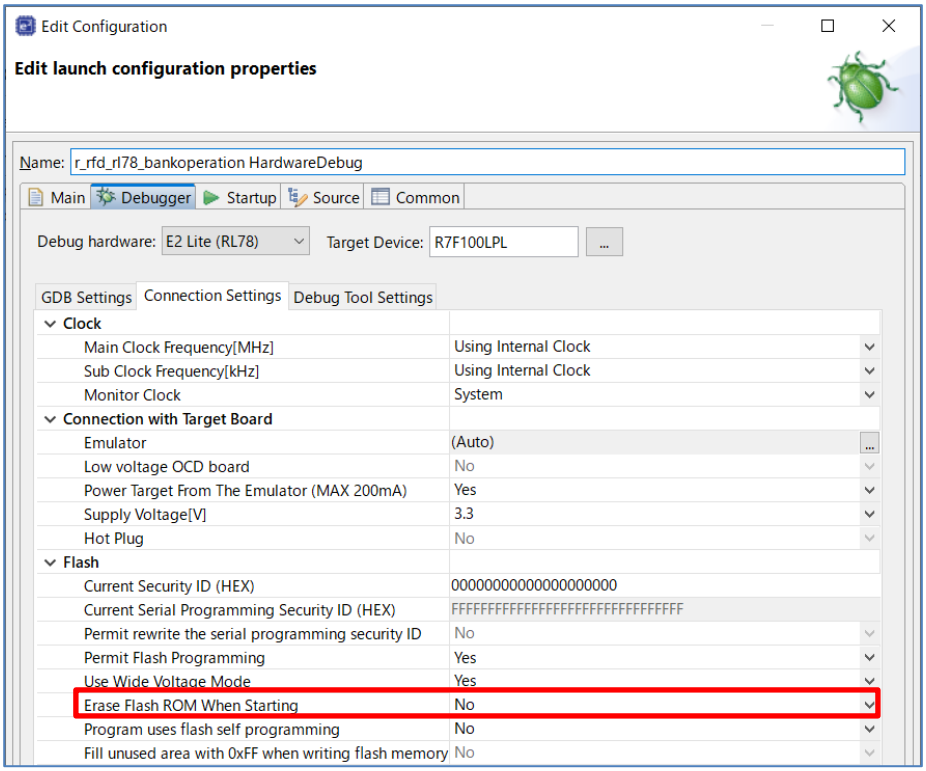
(7) Confirm that addresses 0x5000 to 0x57FF and addresses 0x46000 to 0x467FF match..



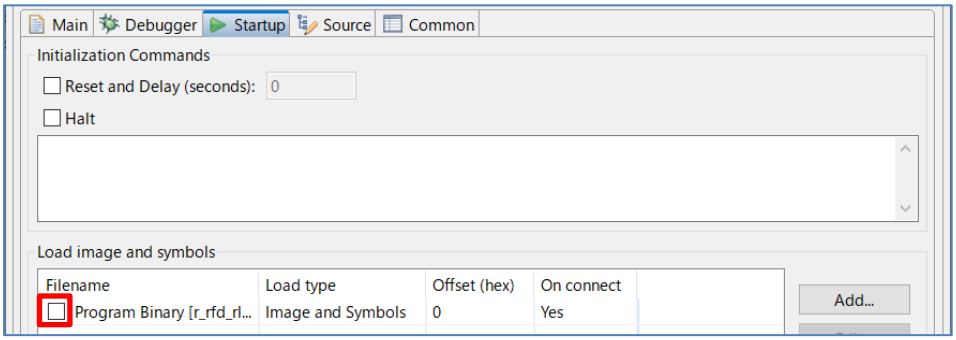
2.4.2 In Case of e² studio(CC-RL)

(1) Write an object file to the device, run it stand-alone, and then connect it to the emulator.

(2) On the project, select “Property” - “Run/Debug Settings”, and edit the target “HardwareDebug” setting. On the displayed screen, select “Debugger” tab - “Connection Settings” tab, and set “No” to “Flash” - “Erase Flash ROM When Starting”.

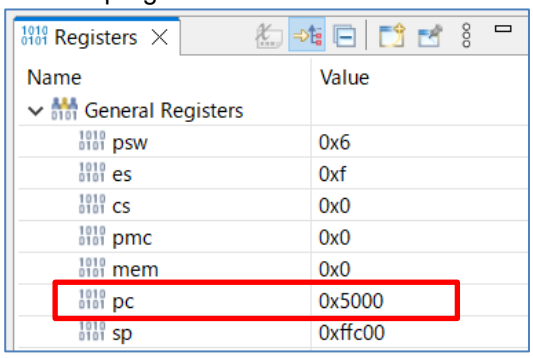


(3) Remove a check mark from “Program Binary” under “Load Image and symbols” in the “Startup” tab.



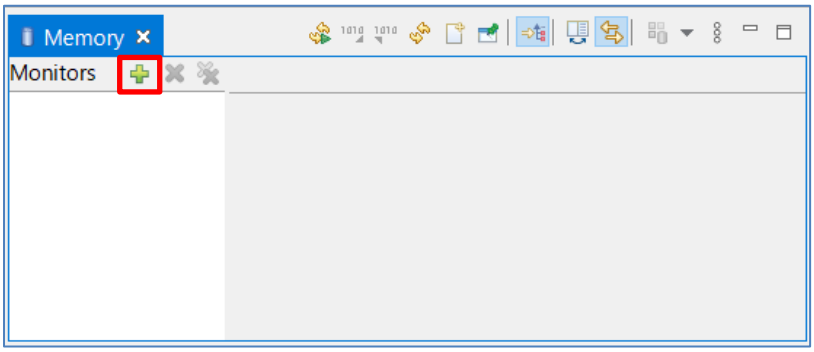
(4) Select [Run]menu - [Debug] and start debugging.

(5) Select [Window] - [Show View] - [Other…] - [IO Registers], and display IO Registers view. Confirm that the program counter is 0x5000 at reset.

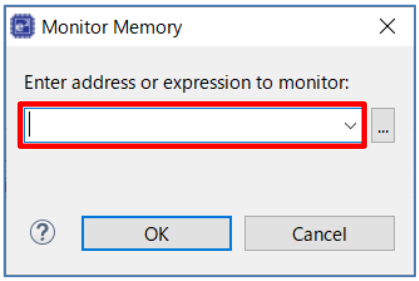


(6) Select [Window]menu - [Show View] - [Memory] and display Memory view.

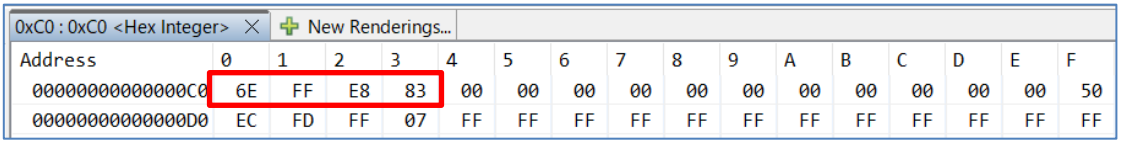
(7) Press " " button and display the Monitor Memory window.



(8) Input "0xC0" into the address to monitor, and press the OK button. The Option Byte is displayed on the memory view.

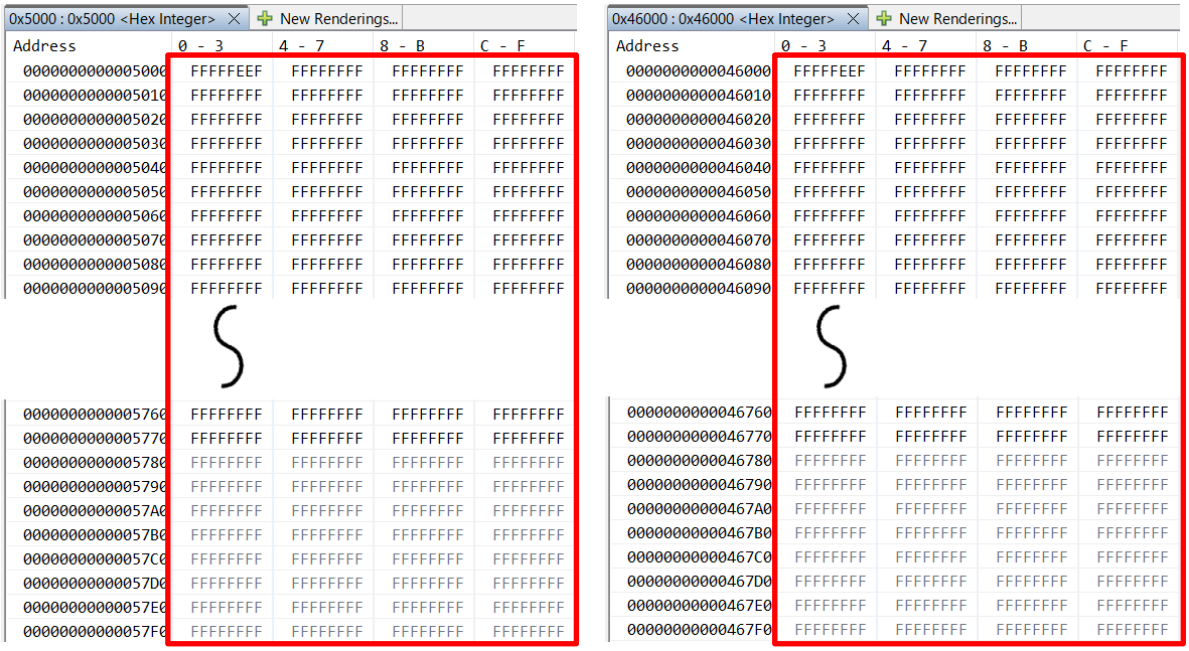


(9) Confirm that the Option Bytes (0xC0 to 0xC3) are 0x6E, 0xFF, 0xE8, and 0x85.



Note: Since bit 3-1 of the On-Chip Debug Option Byte is undefined, make sure that the values of all bits except bit 3-1 match.

(10) Follow the same procedure to display addresses 0x5000 and 0x46000, and confirm that the values of 0x5000 to 0x57FF and 0x46000 to 0x467FF match.



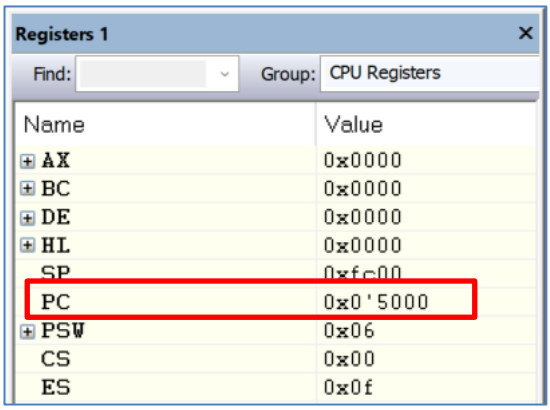
2.4.3 In Case of IAR EW for Renesas RL78

(1) Write an object file to the device, run it stand-alone, and then connect it to the emulator.

(2) Check a [Hardware Setup…] of [Emulator] menu.

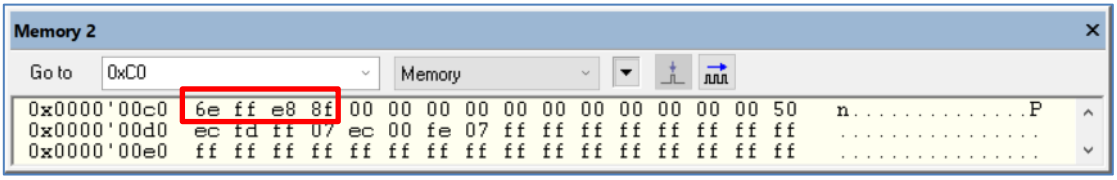
(3) Select [Project] menu - [Debug without Downloading] and start debugging. And remove the check mark to “Erase flash before next ID check” in an “E2 Lite Hardware Setup” window. Then, select [Debug] menu - [Break] and stop the program.

(4) After select [View]menu - [Register], select either from “Registers 1” to “Registers 4”, and the register window is displayed. Confirm that the program counter is 0x5000 at reset.



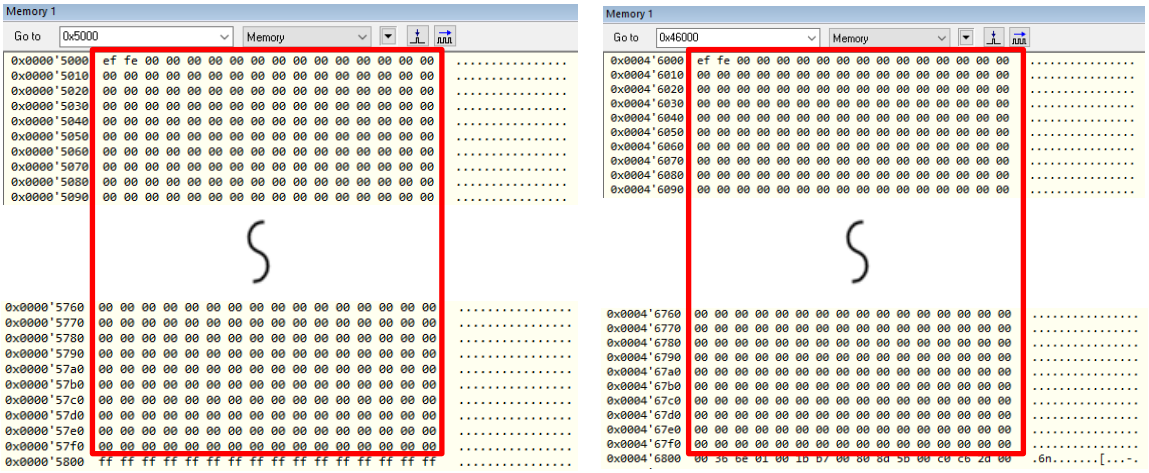
(5) After select [View]menu - [Memory], select either from “Memory 1” to “Memory 4”, and the memory window is displayed.

(6) Set “0xC0” to the address of the memory window. And the Option Bytes (0xC0 to 0xC3) are displayed. Confirm that the Option Bytes (0xC0 to 0xC3) are 0x6E, 0xFF, 0xE8, and 0x85.



Note: Since bit 3-1 of the On-Chip Debug Option Byte is undefined, make sure that the values of all bits except bit 3-1 match.

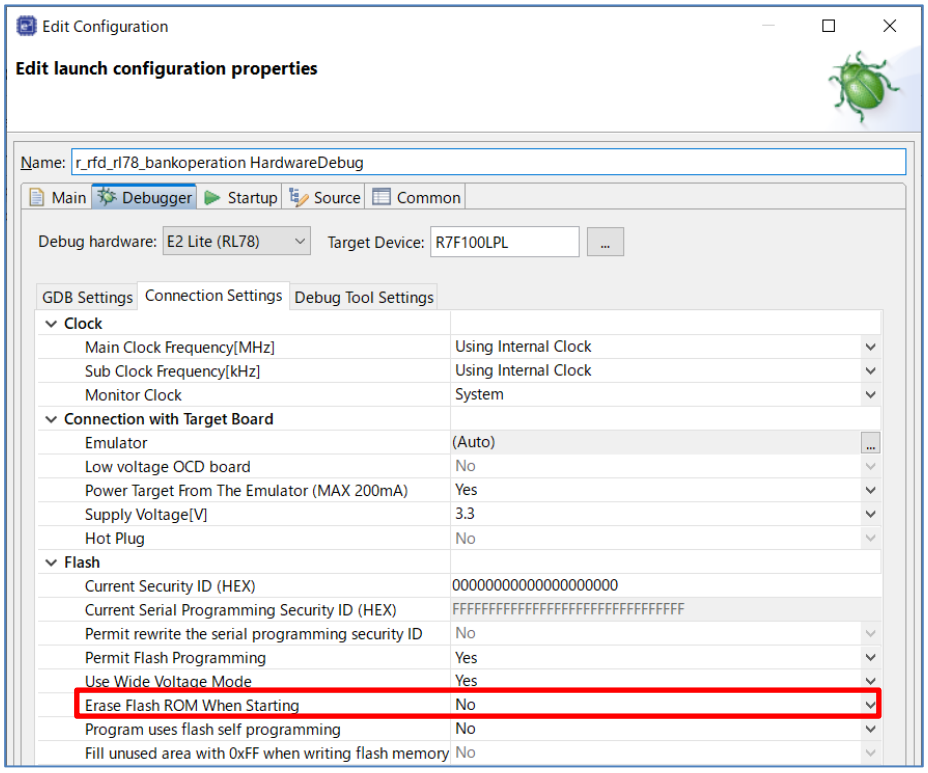
(7) Set “0x5000” and “0x46000” to the address of the memory window. Confirm that the values of 0x5000 to 0x57FF and 0x46000 to 0x467FF match.



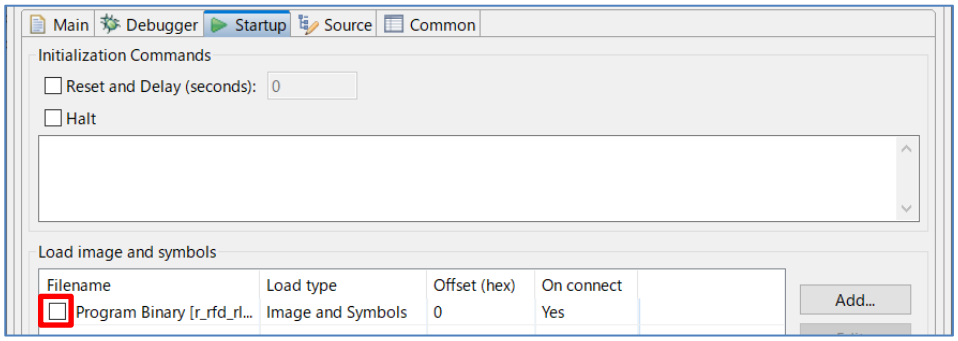
2.4.4 In Case of e² studio(LLVM)

(1) Write an object file to the device, run it stand-alone, and then connect it to the emulator.

(2) On the project, select “Property” - “Run/Debug Settings”, and edit the target “HardwareDebug” setting. On the displayed screen, select “Debugger” tab - “Connection Settings” tab, and set “No” to “Flash” - “Erase Flash ROM When Starting”.

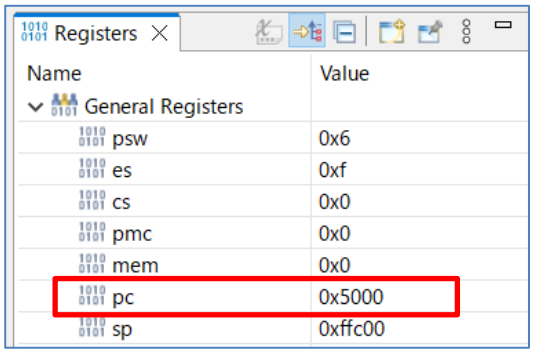


(3) Remove a check mark from “Program Binary” under “Load Image and symbols” in the “Startup” tab.



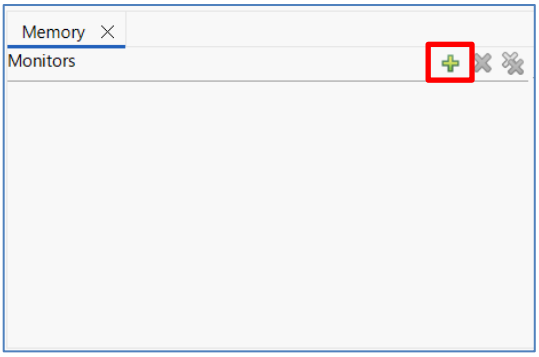
(4) Select [Run]menu - [Debug] and start debugging.

(5) Select [Window] - [Show View] - [Registers], and display Registers view. Confirm that the program counter is 0x5000 at reset.

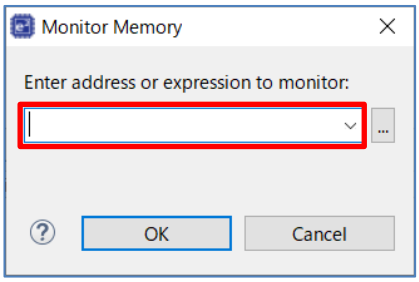


(6) Select [Window]menu - [Show View] - [Memory] and display Memory view.

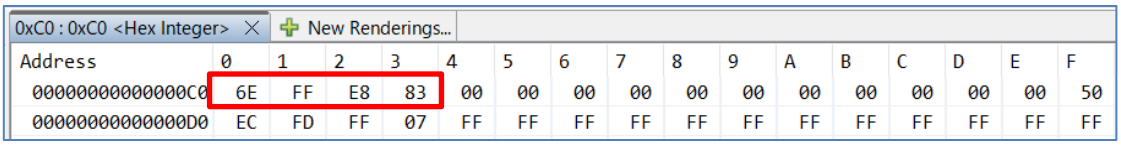
(7) Press " " button and display the Monitor Memory window.



(8) Input "0xC0" into the address to monitor, and press the OK button. The Option Byte is displayed on the memory view.

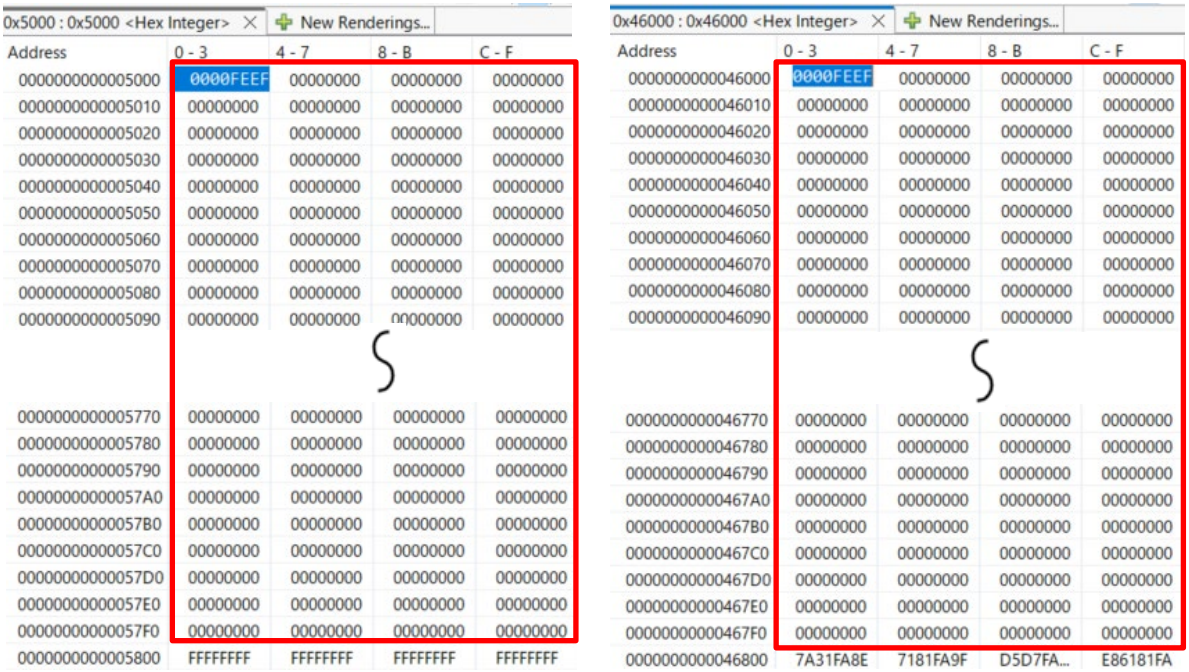


(9) Confirm that the Option Bytes (0xC0 to 0xC3) are 0x6E, 0xFF, 0xE8, and 0x85.



Note: Since bit 3-1 of the On-Chip Debug Option Byte is undefined, make sure that the values of all bits except bit 3-1 match.

(10) Follow the same procedure to display addresses 0x5000 and 0x46000, and confirm that the values of 0x5000 to 0x57FF and 0x46000 to 0x467FF match.



0x5000 : 0x5000 <Hex Integer>

Address	0 - 3	4 - 7	8 - B	C - F
0000000000005000	0000FEEF	00000000	00000000	00000000
0000000000005010	00000000	00000000	00000000	00000000
0000000000005020	00000000	00000000	00000000	00000000
0000000000005030	00000000	00000000	00000000	00000000
0000000000005040	00000000	00000000	00000000	00000000
0000000000005050	00000000	00000000	00000000	00000000
0000000000005060	00000000	00000000	00000000	00000000
0000000000005070	00000000	00000000	00000000	00000000
0000000000005080	00000000	00000000	00000000	00000000
0000000000005090	00000000	00000000	00000000	00000000
0000000000005770	00000000	00000000	00000000	00000000
0000000000005780	00000000	00000000	00000000	00000000
0000000000005790	00000000	00000000	00000000	00000000
00000000000057A0	00000000	00000000	00000000	00000000
00000000000057B0	00000000	00000000	00000000	00000000
00000000000057C0	00000000	00000000	00000000	00000000
00000000000057D0	00000000	00000000	00000000	00000000
00000000000057E0	00000000	00000000	00000000	00000000
00000000000057F0	00000000	00000000	00000000	00000000
0000000000005800	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF

0x46000 : 0x46000 <Hex Integer>

Address	0 - 3	4 - 7	8 - B	C - F
0000000000046000	0000FEEF	00000000	00000000	00000000
0000000000046010	00000000	00000000	00000000	00000000
0000000000046020	00000000	00000000	00000000	00000000
0000000000046030	00000000	00000000	00000000	00000000
0000000000046040	00000000	00000000	00000000	00000000
0000000000046050	00000000	00000000	00000000	00000000
0000000000046060	00000000	00000000	00000000	00000000
0000000000046070	00000000	00000000	00000000	00000000
0000000000046080	00000000	00000000	00000000	00000000
0000000000046090	00000000	00000000	00000000	00000000
0000000000046770	00000000	00000000	00000000	00000000
0000000000046780	00000000	00000000	00000000	00000000
0000000000046790	00000000	00000000	00000000	00000000
00000000000467A0	00000000	00000000	00000000	00000000
00000000000467B0	00000000	00000000	00000000	00000000
00000000000467C0	00000000	00000000	00000000	00000000
00000000000467D0	00000000	00000000	00000000	00000000
00000000000467E0	00000000	00000000	00000000	00000000
00000000000467F0	00000000	00000000	00000000	00000000
0000000000046800	7A31FA8E	7181FA9F	D5D7FA...	E86181FA

3. Precautions

(1) Reprogramming of the code flash memory or extra area
Place the reprogramming code in RAM when reprogramming the code flash memory or extra area. However, place the reprogramming code in ROM when reprogramming the code flash memory on bank programming mode.

(2) Precondition for control of the data flash area
Be sure to set the DFLEN bit (bit 0) of the data flash control register (DFLCTL) to 1 (enable access to the data flash area) before controlling the data flash area.

(3) Program execution during reprogramming of the flash memory
Self-programming in the RL78/L23 uses the flash memory sequencer to control the reprogramming of the flash memory. In the following flash memory control modes in which the flash memory can be reprogrammed, the CPU cannot read data from the target flash memory.

- In the code flash memory programming mode, the CPU cannot read data from the code flash memory. The API functions of RFD RL78 Type 11 and the user program to be executed in the code flash memory programming mode should be copied from ROM to RAM in advance and executed and referenced in RAM.
- In the data flash memory programming mode, the CPU cannot read data from the data flash memory. The data to be read in the data flash memory programming mode should be copied from the data flash memory to RAM in advance and referenced in RAM.

(4) Clock setting under self-programming execution
The high-speed on-chip oscillator should be kept operating during self-programming. When self-programming is executed, it is necessary to use the main system clock and to select “X1 oscillator or the high-speed on-chip oscillator.” Cannot execute self-programming, if the middle-speed on-chip oscillator or the subsystem clock is selected. Stop the middle-speed on-chip oscillator (MIOEN = 0) and select the high-speed on-chip oscillator (MCM1 = 0) as the main on-chip oscillator clock (fOCO).

* For the clock setting under self-programming execution, refer to the user’s manual of the target RL78 microcontroller.

(5) The precautions in the case of debugging self-programming with an on-chip debugger
In the case which debugs self-programming with an on-chip debugger, because 128 bytes of area is used from the top address of RAM when a debugger is executed, it is necessary to vacate this area. Additionally, in case CS+ or e^2 studio is used as the development environment, the debugger settings need to be configured to use flash self-programming

- Example settings for CS+:
  On the project, select “Connect Settings” tab from “RL78 E2 [Lite] (Debug Tool)”, and set “Yes” to “Flash” - “Using the flash self programming”.
- Example settings for e^2 studio:
  On the project, select “Property” - “Run/Debug Settings”, and edit the target “HardwareDebug” setting.
  On the displayed screen, select “Debugger” tab - “Connection Settings” tab, and set “Yes” to “Flash” - “Program uses flash self programming”.

(6) The precautions in the case of executing the data copy from ROM to RAM, when using CC-RL compiler.
When using CC-RL compiler, the Sample_BankSwap_INITSCT function is called from the main function of main.c file. This function copies the data and the program for RFD RL78 Type 11 to RAM from ROM.
However, the following setting will be necessary if this processing is executed by the start-up routine in the cstart.asm file which is a CC-RL compiler function.
(CC-RL compiler function : “Initialization of RAM area sections by using an initialization table [V1.12 or later]”)

- Set “-ram_init_table_section” by linker.
- Set “__USE_RAM_INIT_TABLE” to the column which defines the macro of assemble options.

* For details, please refer to the user’s manual of CC-RL compiler

Because “copy processing from ROM to RAM” of a Sample_BankSwap_INITSCT function duplicates in this case,It is necessary to set same [Macro definition] as “Compiler Option”, and to cancel processing of a Sample_BankSwap_INITSCT function.

- Set “__USE_RAM_INIT_TABLE” to the column which defines the macro of compiler options.

(7) The precautions in case where an on-chip debugger is connected after bank swap execution

The bank swap is executed and the swapping state of the bank 0 and the bank 1 has restriction in on-chip debugger connection. If “the option byte and on-chip debugging security ID” of the rewrite bank side in a chip are not programmed, it will become impossible to connect an on-chip debugger. If bank programming / bank swapping execution, and a debugger are cut, the setup is necessary to subsequent reconnection. It is setting up an Integrated Development Environment so that “the option byte and on-chip debugging security ID” of rewrite bank side may not be erased by program download of debugger reconnection.

- Example settings for CS+:

  On the project, select “Download File Settings” tab from “RL78 E2 [Lite] (Debug Tool)”, and select “Data priority” to “Download” - “Download Mode”.

  Example settings for e^2 studio (CC-RL) : On the project, select “Property” - “Run/Debug Settings”, and edit the target “HardwareDebug” setting. On the displayed screen, select “Debugger” tab - “Connection Settings” tab, and set “No” to “Flash” - “Erase Flash ROM When Starting”.

- Example settings for IAR (Embedded Workbench) : Remove the check mark to “Erase flash before next ID check” in an “E2 Lite Hardware Setup” window.

- Example settings for e^2 studio (LLVM) :

  Description which fills up a debugging monitor area with 0xff is added. (*. ld file, vects.c file)

  - *.ld file:

```
.debug_monitor 0xCE : AT(0xCE)
{
        KEEP(*(.debug_monitor))
} > OCDSTAD
```

  - vects.c file:

```
const unsigned char Debug_Monitor[]  __attribute__ ((section (".debug_monitor"))) = {
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
};
```

  Caution: The countermeasure method when the option byte in rewrite bank side and on-chip debugging security ID are erased accidentally.

  If problems occur in debugger starting, it is necessary to execute erase (select “Erase Chip” as Erase Option) by the dedicated flash memory programmer. And be sure to restore a chip to an initial state and to redo from the beginning.

4. Reference document

Please get the latest version of each document from the Renesas Electronics Corp. website (https://www.renesas.com).

No	Document Title	Document Number
1	RL78/L23 User's Manual Hardware	R01UH1082
2	RL78 Family Board Support Package Module	R01AN5522
3	RL78 Family Renesas Flash Driver RL78 Type 11 user's manual	R20UT5539
4	E1/E20/E2 Emulator, E2 Emulator Lite Additional Document for User's Manual (Notes on Connection of RL78)	R20UT1994

5. Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul.31.25	—	Newly created.

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

Contact information

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

Trademarks

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