RL78 Family
EEPROM Emulation Library Pack02 Package Ver.3.00
Release Note

R20UT2645EJ0110
Rev.1.10
Mar 15, 2024

Thank you for using the RL78 Family EEPROM Emulation Library Pack02 Package Ver.3.00.
This document contains precautionary and other notes regarding use of the EEPROM Emulation Library Pack02 Package Ver.3.00. Please read this document before using the library.

Contents

Chapter 1 Target Product

EEPROM Emulation Library Pack02 package Ver.3.00 contains EEPROM Emulation Library Pack02 for CA78K0R compiler, CC-RL compiler and LLVM compiler (LLVM for Renesas RL78).

The following shows the target product for this release note.

Product Name	Ver.	Installer Name	Ver.
RL78 Family EEPROM Emulation Library Pack02 for the CA78K0R Compiler	V1.01	RENESAS_RL78_EEL-FDL_T02_PACK02_3V00.exe	V3.00
RL78 Family EEPROM Emulation Library Pack02 for the CC-RL Compiler	V1.01		
RL78 Family EEPROM Emulation Library Pack02 for the LLVM Compiler	V1.01		

Chapter 2 User's Manual

The following user's manual covers this version of the library.

Target Compiler	Title of User's Manual	Document Number
CA78K0R, CC-RL and LLVM Compilers	RL78 Family EEPROM Emulation Library Pack02 Japanese Release User's Manual Note	R01US0068EJ0110

Note: Download this document from the Renesas Electronics website.

Chapter 3 Revisions

The following shows the items upgraded in the new version.

No.	Package Ver.	Target	Contents
1	V3.00	RL78 Family EEPROM Emulation Library Pack02 for CA78K0R	There is no change in the library body from Package Ver.2.00.
		RL78 Family EEPROM Emulation Library Pack02 for CC-RL	There is no change in the library body from Package Ver.2.00.
		RL78 Family EEPROM Emulation Library Pack02 for LLVM	Newly added.
		User's Manual	Rev.1.01 to Rev.1.10 Revision For revision contents, please refer to the revision history of the user's manual (R01US0068EJ0110).

Chapter 4 Supported Tools

Use the following tool version when using tools in combination with this library.

Target Library	Tool Used	Version
Library for CA78K0R Compiler	Integrated development environment CubeSuite+	V1.00.00 or later
	Integrated development environment CS+	V3.00.00 or later
Library for CC-RL Compiler	Integrated development environment CS+	V3.01.00 or later
	Integrated development environment e^2 studio	Listed from Version: 2024-01 Note
Library for LLVM Compiler	Integrated development environment e^2 studio	Version: 2024-01 or later

Note: The CC-RL Compiler V1.00 or later can be used with the installed version.

Chapter 5 Installation

This chapter describes how to install and uninstall the EEPROM Emulation Library Pack02 package Ver.3.00.

5.1 Installation

Install the EEPROM Emulation Library Pack02 by using the following procedure:

(1) Start Windows.

(2) Decompress the file that contains the EEPROM Emulation Library Pack02 package and run the installer.

(3) Select "Asia/Oceania - English" from the drop-down list.

(4) Click on the "OK" button to proceed installation according to the instructions of the installer.

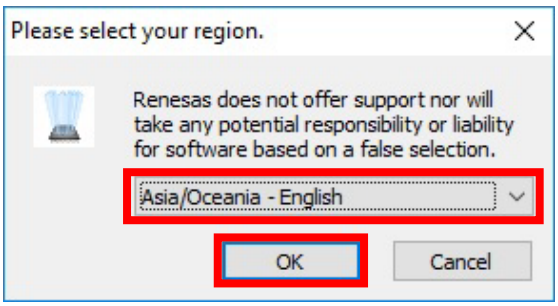

Figure 5-1. Select "Asia/Oceania - English"

5.2 Uninstallation

Uninstall the EEPROM Emulation Library Pack02 by using the following procedure:

(1) Start Windows.

(2) Delete the folder that contains the EEPROM Emulation Library Pack02 files.

5.3 File Organization

The file organization after this library is installed is shown below.

```
Installation folder
├─ r20ut2645ejxxxx_rl78.pdf Note1          : Release Note (this document)
├─ support.txt                             : Support information file for EEL
└─ CA78K0R_110, CCRL_100 or LLVM_202312    : The target CA78K0R compiler version is V1.10 or later,
                                             the target CC-RL compiler version is V1.00 or later
                                             or the target LLVM for Renesas RL78 10.0.0.202312 or later
   ├─ lib
   │  ├─ eel.lib or libeel.a               : EEPROM emulation library (EEL)
   │  ├─ fdl.lib  or libfdl.a              : Data flash library (FDL)
   │  ├─ eel.h                             : EEL header file for C program
   │  ├─ eel.inc                           : EEL header file for assembler Note5
   │  ├─ eel_types.h                       : EEL header file that specifies definitions for C program
   │  ├─ eel_types.inc                     : EEL header file that specifies definitions for assembler Note5
   │  ├─ fdl.h                             : FDL header file for C program
   │  ├─ fdl.inc                           : FDL header file for assembler Note5
   │  └─ fdl_types.h                       : FDL header file that specifies definitions for C program
   └─ Sample
      ├─ asm Note5
      │  ├─ eel_descriptor.inc             : EEL descriptor header file
      │  ├─ eel_descriptor.asm             : EEL descriptor source file
      │  ├─ eel_sample_linker_file.dr      : EEL sample link directive file (CA78K0R version only)
      │  ├─ fdl_descriptor.inc             : FDL descriptor header file
      │  └─ fdl_descriptor.asm             : FDL descriptor source file
      └─ C
         ├─ eel_descriptor.h               : EEL descriptor header file
         ├─ eel_descriptor.c               : EEL descriptor source file
         ├─ eel_sample_linker_file.dr      : EEL sample link directive file (CA78K0R version only)
         ├─ eel_user_types.h               : EEL user-defined header file
         ├─ fdl_descriptor.h               : FDL descriptor header file
         ├─ fdl_descriptor.c               : FDL descriptor source file
         ├─ r_eel_sample_c.c               : EEL sample program file Note2,3,4
         ├─ r_eel_sample_c.dr              : Link directive file for EEL sample program (CA78K0R version only) Note2
         └─ or r_eel_sample_c.ld           : Linker script file for EEL sample program (LLVM version only) Note4
```

Notes: 1. x indicates the omitted numerals in version or revision numbers.

2. If you wish to use the sample program for CA78K0R, include both the program file (*.c) and the link directive file (*.dr) [setting file for link information].
3. To use the sample program for CC-RL, the program file (*.c) should be embedded. The link information for the sample program for CC-RL should be specified through the link setting window on the CS+ or the e^2 studio.
4. To use the sample program for LLVM, the program file (*.c) and linker script file (*.ld) should be embedded together.
5. The assembler files are only included in the CA78K0R and CC-RL folders.

Chapter 6 How to Build a Program

This chapter describes how to build a program using the EEPROM Emulation Library Pack02.

6.1 Software to be Used

Below are the system requirements for building programs using the EEPROM Emulation Library Pack02.

- For CA78K0R compiler : Integrated development environment CS+ V3.00.00 or later or integrated development environment CubeSuite+ V1.00.00 or later.
- For CC-RL compiler : Integrated development environment CS+ V3.01.00 or later or integrated development environment e^2 studio listed from Version 2024-01 or later [Note].

Note Available for e^2 studio with embedded CC-RL compiler V1.00 or later.

- For LLVM compiler : Integrated development environment e^2 studio Version 2024-01 or later.

6.2 Building Using CS+ (Formerly CubeSuite+)

This section describes how to include the EEPROM Emulation Library Pack02 in a user-created program and build the user program by using CS+. The target compilers for CS+ are CC-RL compiler and CA78K0R compiler.

6.2.1 Building a C Program

(1) Creating a project and specifying the source files

Create a project by using CS+. In the Project Tree window displayed on the left, right-click the File node, click Add, and then click Add File. The Add Existing File dialog box is displayed (as shown in Figure 6-1).
Click the Files of type drop-down list, select C source file (*.c), and then register the user-created program file (r_eel_sample_c.c for the sample file of source code) and the descriptor files for the EEPROM emulation library and data flash library (eel_descriptor.c and fdl_descriptor.c) as the source files.

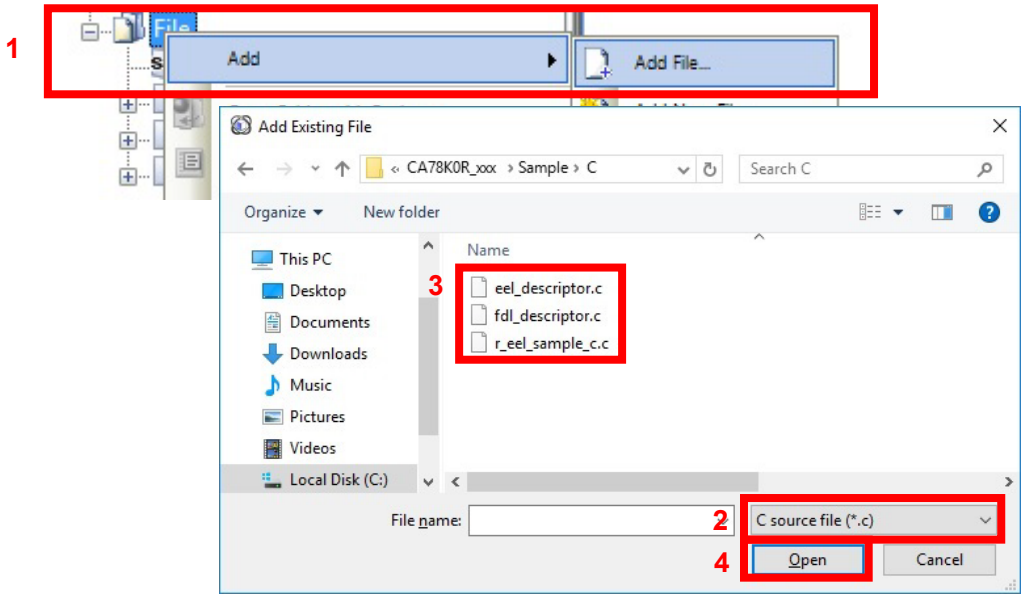

Figure 6-1. Specifying the Source Files

(2) Specifying the include file

In the CS+ Project Tree window, right-click the File node, click Add, and then click Add File.
The Add Existing File dialog box is displayed (as shown in Figure 6-2).
Click the Files of type drop-down list, select Header file (*.h; *.inc), and then register the header files and descriptor header files for the EEPROM emulation library and data flash library (eel.h, eel_types.h, fdl.h, fdl_types.h, eel_descriptor.h, fdl_descriptor.h, and eel_user_types.h).

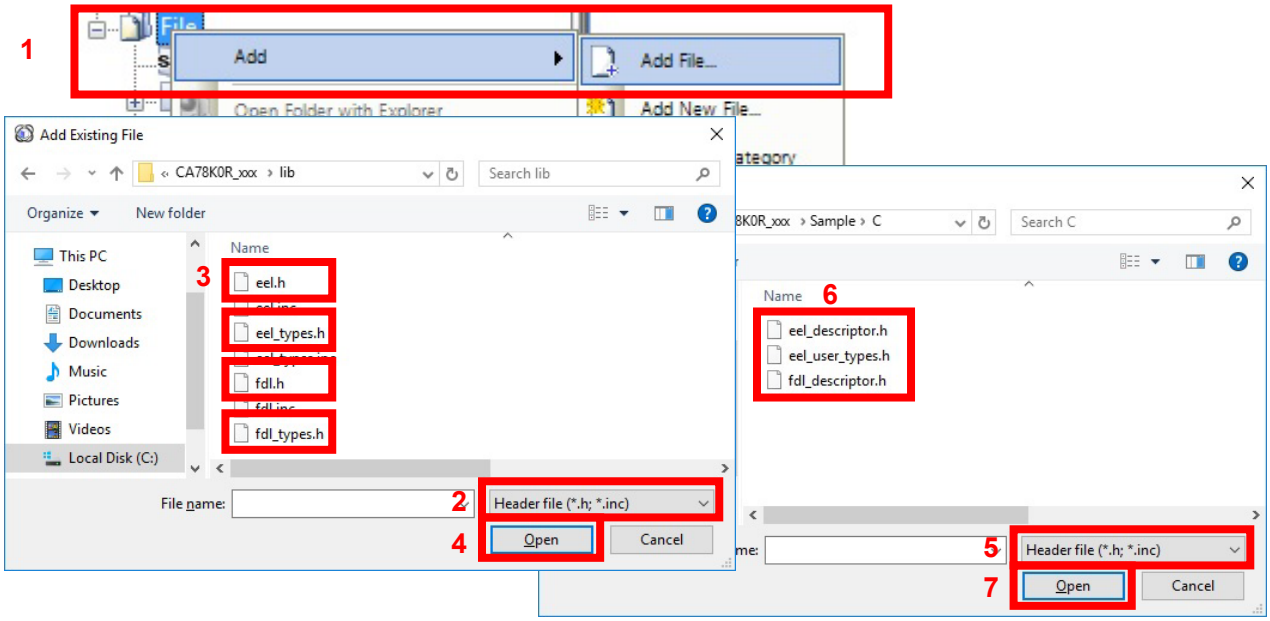

Figure 6-2. Specifying the Include Files

(3) Specifying the library file

In the CS+ Project Tree window, right-click the File node, click Add, and then click Add File. The Add Existing File dialog box is displayed (as shown in Figure 6-3).
Click the Files of type drop-down list, select Library file (*.lib), and then register the EEPROM emulation library and data flash library files (eel.lib and fdl.lib).

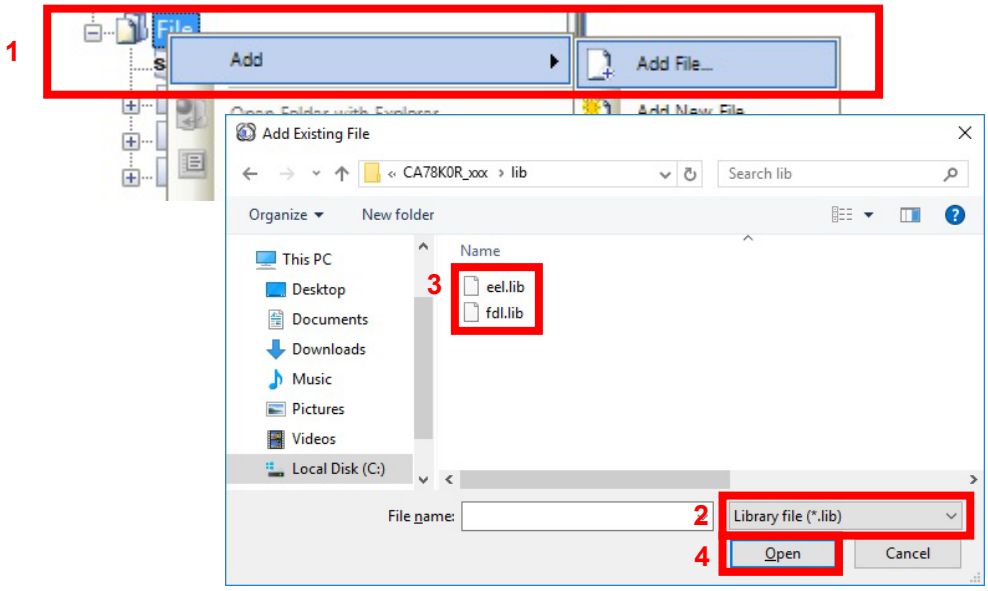

Figure 6-3. Specifying the Library Files

(4) Specifying the link directive file (only when the CA78K0R compiler is used)

In the CS+ Project Tree window, right-click the File node, click Add, and then click Add File. The Add Existing File dialog box is displayed (as shown in Figure 6-4).

Click the Files of type drop-down list, select Link Directive File (*.dr; *.dir), and then register the link directive file that has the same name as the user-created program (r_eel_sample_c.dr for the sample file of source code [Note]).

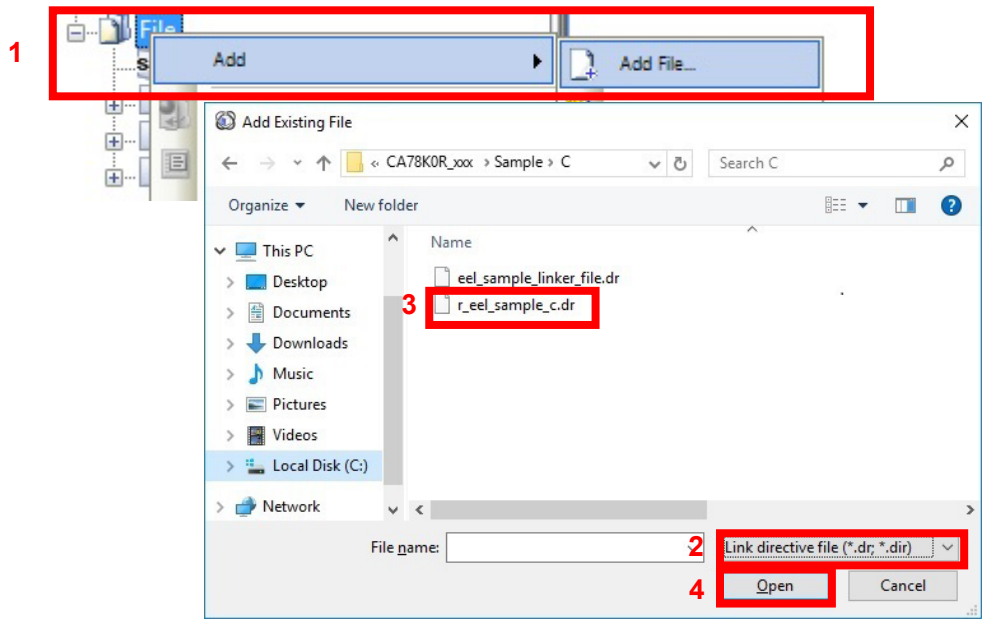

Figure 6-4. Specifying the Link Directive File

Note: The sample directive file that comes with the library may require editing or modification before use.

(5) Building

On the CS+ Build menu, click Build Project to build the project.

6.2.2 Building an Assembly-Language Program

(1) Creating a project and specifying the source files

Create a project by using CS+. In the Project Tree window displayed on the left, right-click the File node, click Add, and then click Add File. The Add Existing File dialog box is displayed (as shown in Figure 6-5).
Click the Files of type drop-down list, select Assemble file (*.asm), and then register the user-created program file and the descriptor files for the EEPROM emulation library and data flash library (eel_descriptor.asm and fdl_descriptor.asm) as the source files.

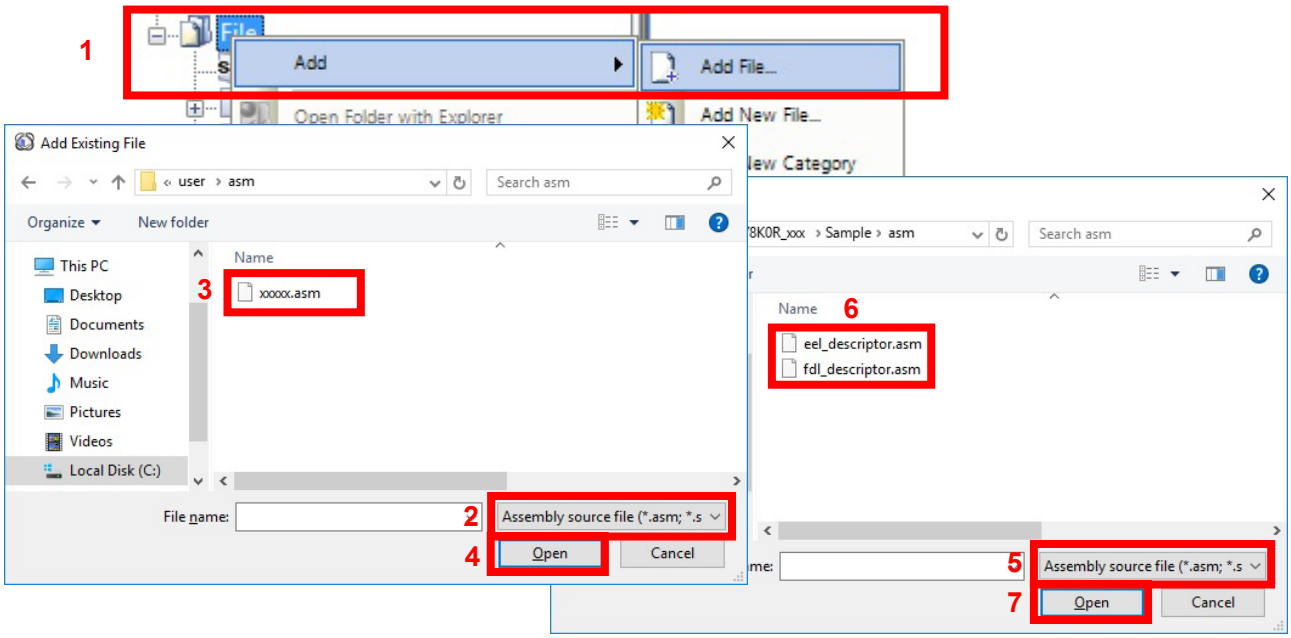

Figure 6-5. Specifying the Assemble Files

(2) Specifying the include file

In the CS+ Project Tree window, right-click the File node, click Add, and then click Add File.
The Add Existing File dialog box is displayed (as shown in Figure 6-6).
Click the Files of type drop-down list, select Header file (*.h; *.inc), and then register header files and the descriptor header files for the EEPROM emulation library and data flash library (eel.inc, eel_types.inc, fdl.inc, eel_descriptor.inc and fdl_descriptor.inc).

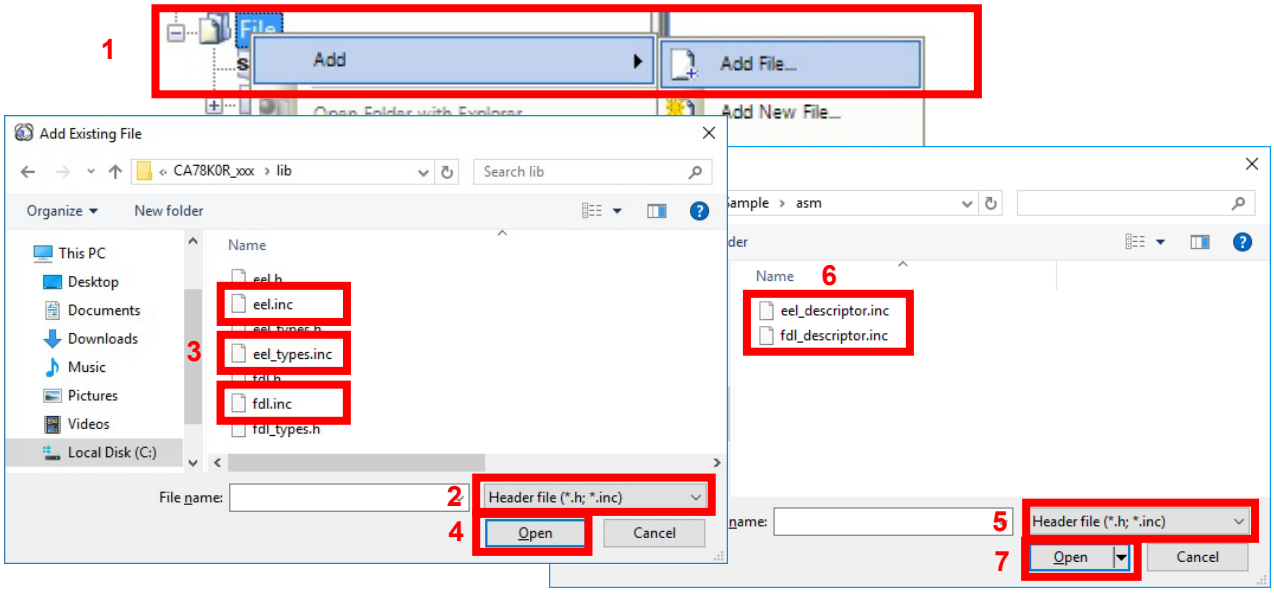

Figure 6-6. Specifying the Include Files

(3) Specifying the library file

In the CS+ Project Tree window, right-click the File node, click Add, and then click Add File. The Add Existing File dialog box is displayed (as shown in Figure 6-7).

Click the Files of type drop-down list, select Library file (*.lib), and then register the EEPROM emulation library and data flash library files (eel.lib and fdl.lib).

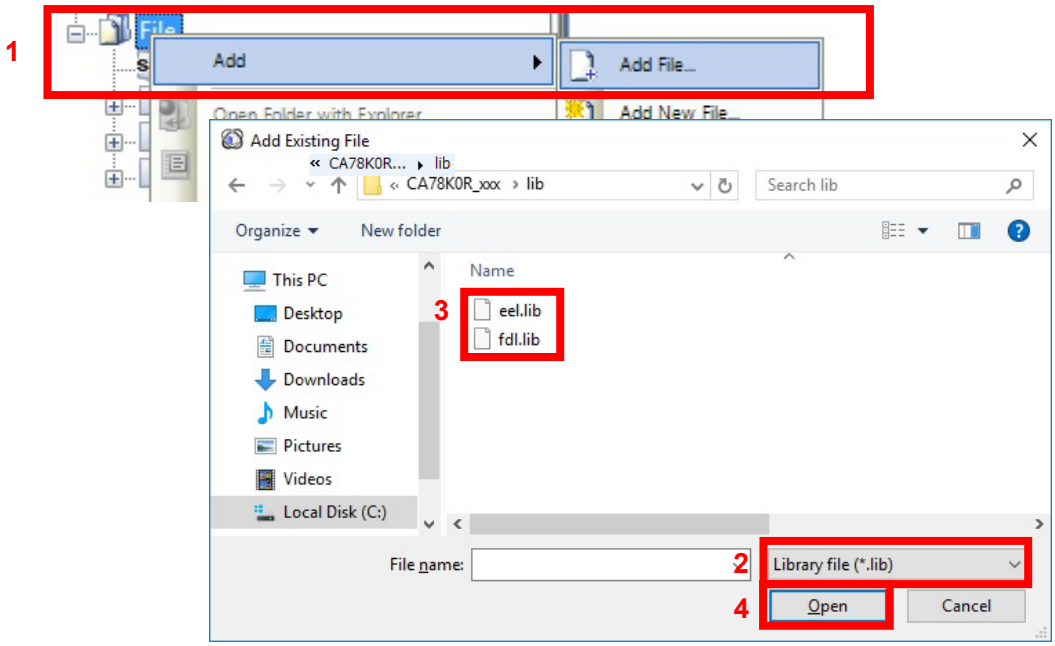

Figure 6-7. Specifying the Library Files

(4) Specifying the link directive file (only when the CA78K0R compiler is used)

In the CS+ Project Tree window, right-click the File node, click Add, and then click Add File. The Add Existing File dialog box is displayed (as shown in Figure 6-8).

Click the Files of type drop-down list, select Link Directive File (*.dr; *.dir), and then register the link directive file that has the same name as the user-created program.

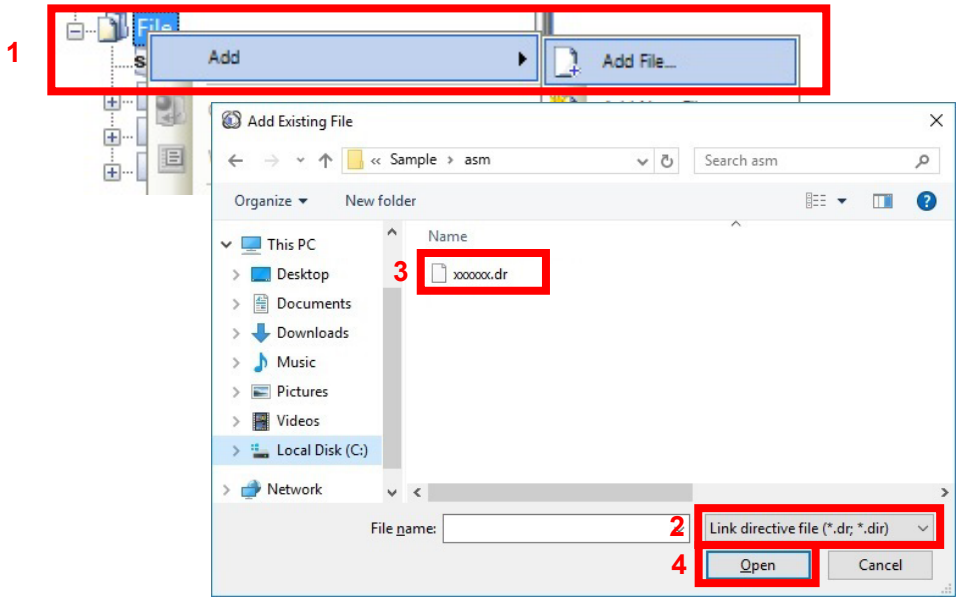

Figure 6-8. Specifying the Link Directive File

(5) Removing the automatically generated files (only when the CC-RL compiler is used)

CS+ for the CC-RL compiler automatically generates some files under the File node in the Project Tree window. Among these, the processing of the "main.c" file is included in the EEPROM emulation library. Therefore, remove this file from the target of the build process (as shown in Figure 6-9).

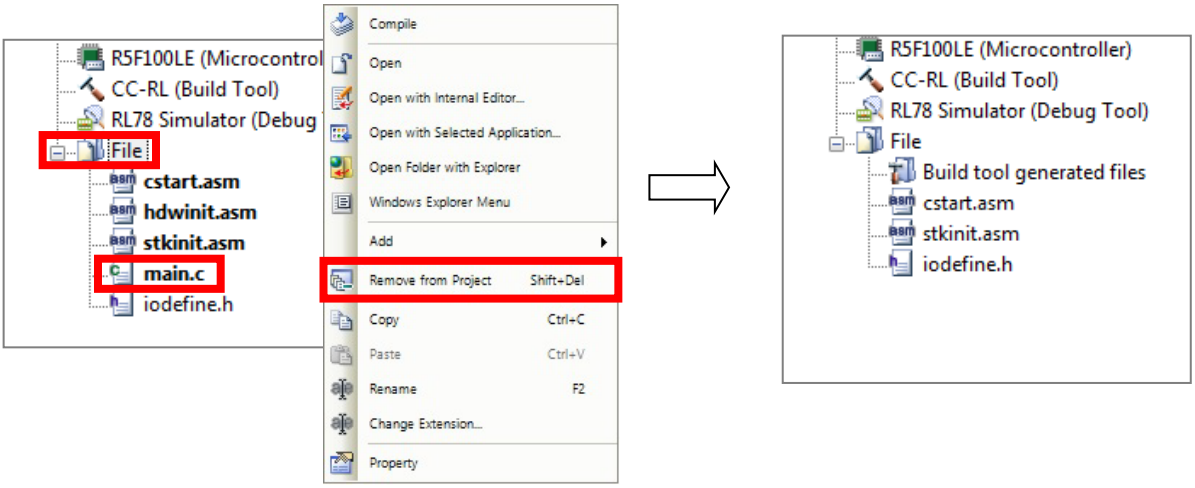

Figure 6-9. Removing the Automatically Generated Files

(6) Building

On the CS+ Build menu, click Build Project to build the project.

6.3 Building Using e2 studio

This section describes how to include the EEPROM Emulation Library Pack02 in a user-created program and build the user program by using e2 studio. The target compilers for e2 studio are CC-RL compiler and LLVM compiler.

6.3.1 Creating a Project

The e2 studio starts and from the [File] menu, select [New] – [C/C++ Project], the "Templates for New C/C++ Project" window will open (as shown in Figure 6-10).

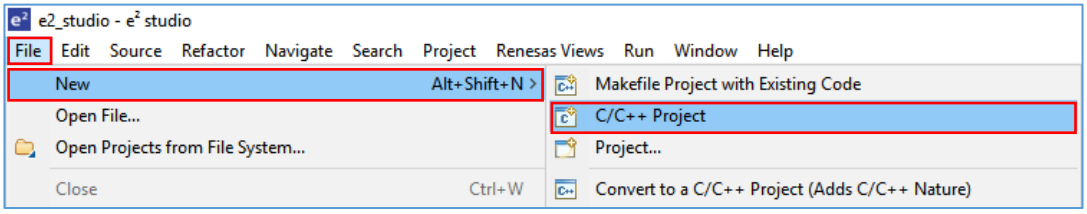

Figure 6-10. Create a New Project

- When using the CC-RL compiler, select [Renesas CC-RL C/C++ Executable Project] displayed after selection in [Renesas RL78], and press "Next" button (as shown in Figure 6-11).

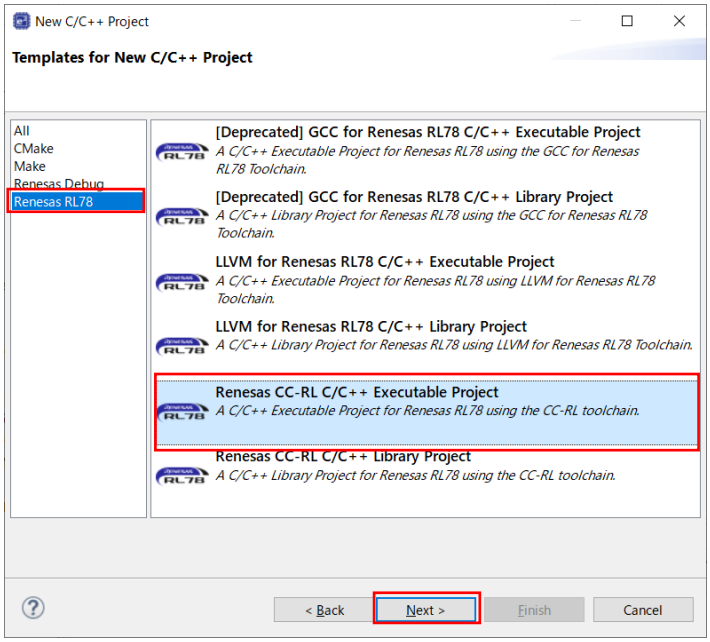

Figure 6-11. Select the CC-RL Compiler for the Tool Chain

Input "project name" on "New Renesas CC-RL Executable Project" window, and press "Next" button.

- When using the LLVM compiler, Select [LLVM for Renesas RL78 C/C++ Executable Project] displayed after selection in [Renesas RL78], and press "Next" button (as shown in Figure 6-12).

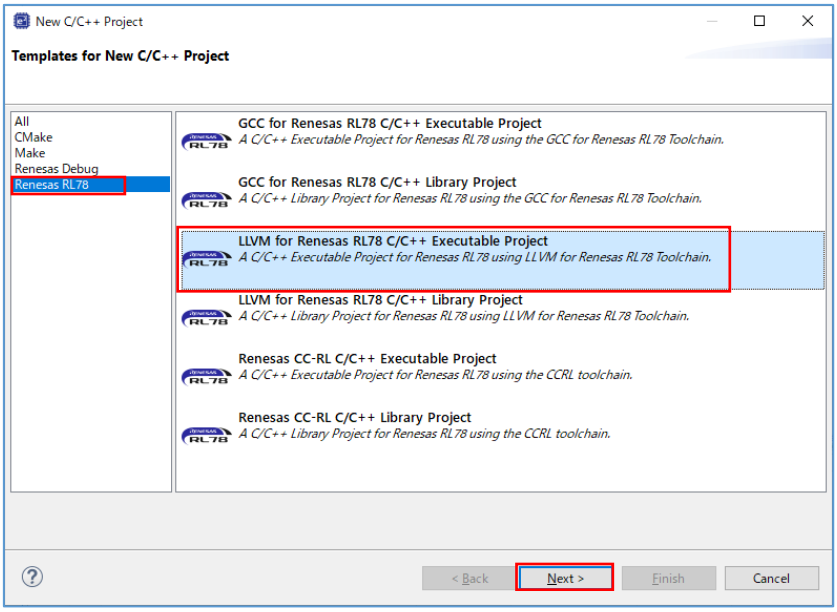

Figure 6-12. Select the LLVM Compiler for the Tool Chain

Input "Project name" on "New LLVM for Renesas RL78 Executable Project" window, and press "Next" button.

Select the [Target Device] of [Device Settings] and select "RL78 – G13" - "R5F100LE". (When the target device is RL78/G13 [Part Number: R5F100LE].)
It is a premise that E2 Lite is selected as a debugging tool and on-chip debugging is executed. Put a check mark to "Create Hardware Debug Configuration" by [Configurations]. And select "E2 Lite(RL78)". Press "Finish" button (as shown in Figure 6-13).

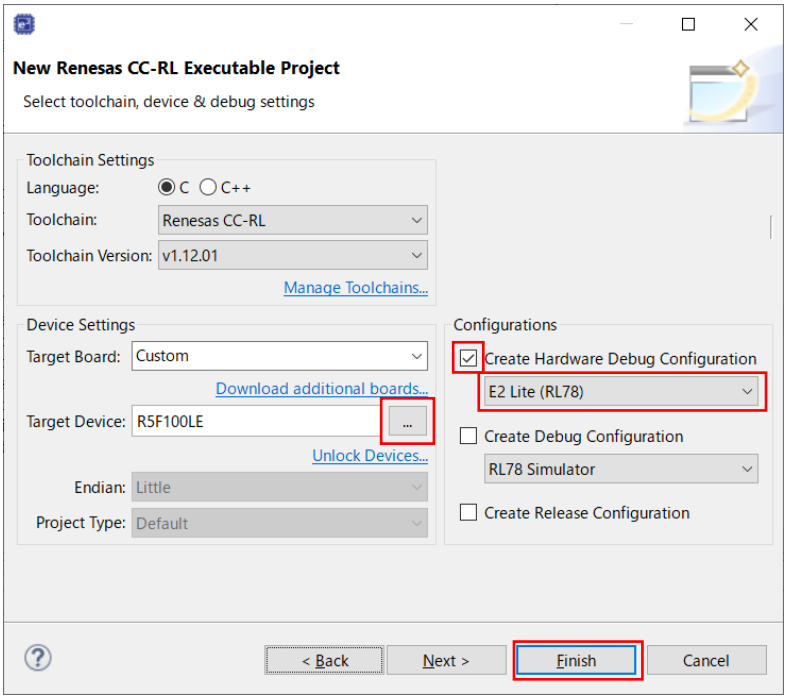

Figure 6-13. Device Selection

6.3.2 Building a C Program

(1) Specifying the source and include files

Specifying the EEPROM emulation library and the data flash library files in the created project.

- CC-RL: Register "eel.h", "eel_types.h", "eel.lib", "fdl.h", "fdl_types.h" and "fdl.lib" in the "src" folder output by e^2 studio. Also, register descriptor files "eel_descriptor.c", "eel_descriptor.h", "eel_user_types.h", "fdl_descriptor.c". "fdl_descriptor.h" and sample program file "r_eel_sample_c.c" (as shown in Figure 6-14).

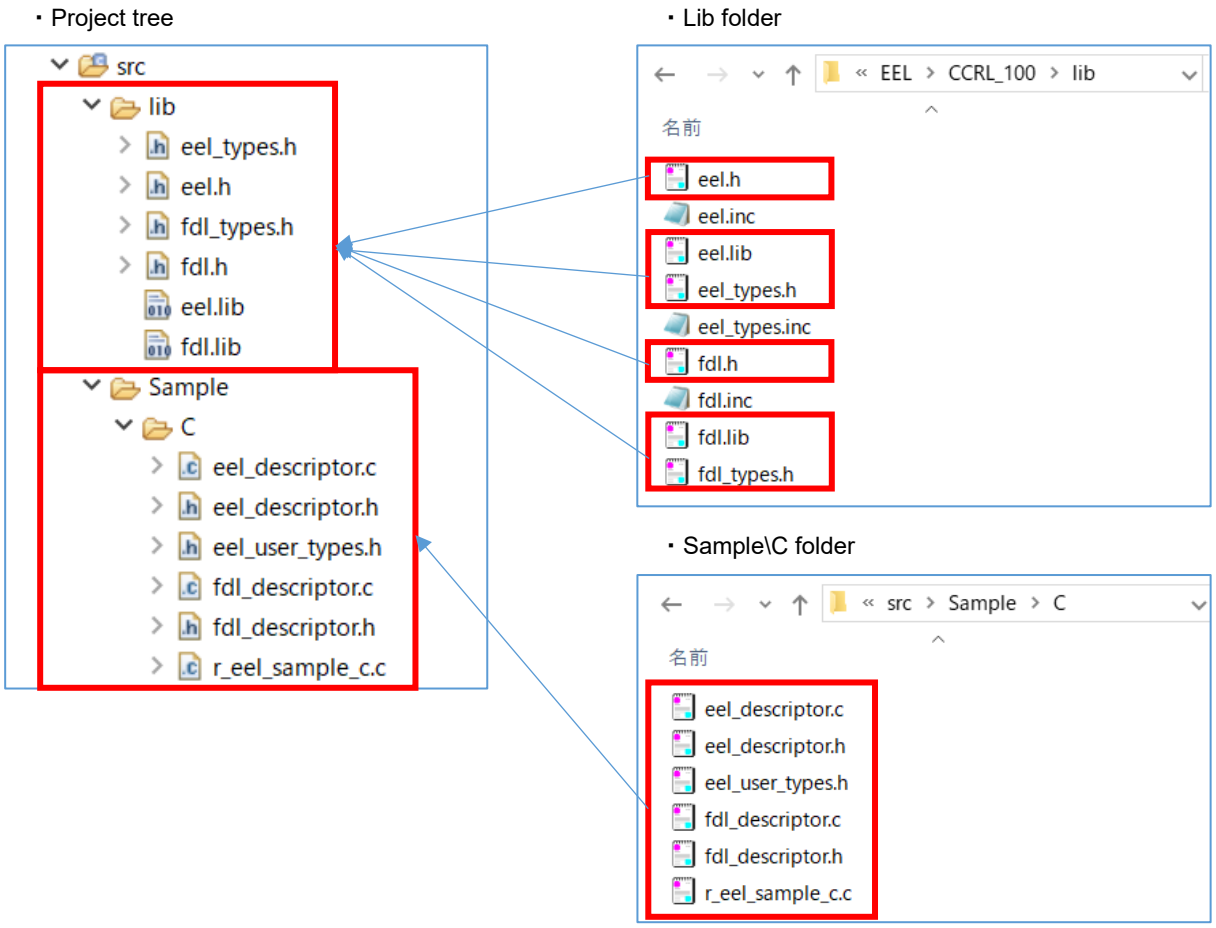

Figure 6-14. Specifying the Source and Include Files (CC-RL)

- LLVM: Register "eel.h", "eel_types.h", "libeel.a", "fdl.h", "fdl_types.h" and "libfdl.a" in the "src" folder output by e^2 studio. Also, register descriptor files "eel_descriptor.c", "eel_descriptor.h", "eel_user_types.h", "fdl_descriptor.c". "fdl_descriptor.h" and sample program files "r_eel_sample_c.c" and "r_eel_sample_c.ld" (as shown in Figure 6-15).

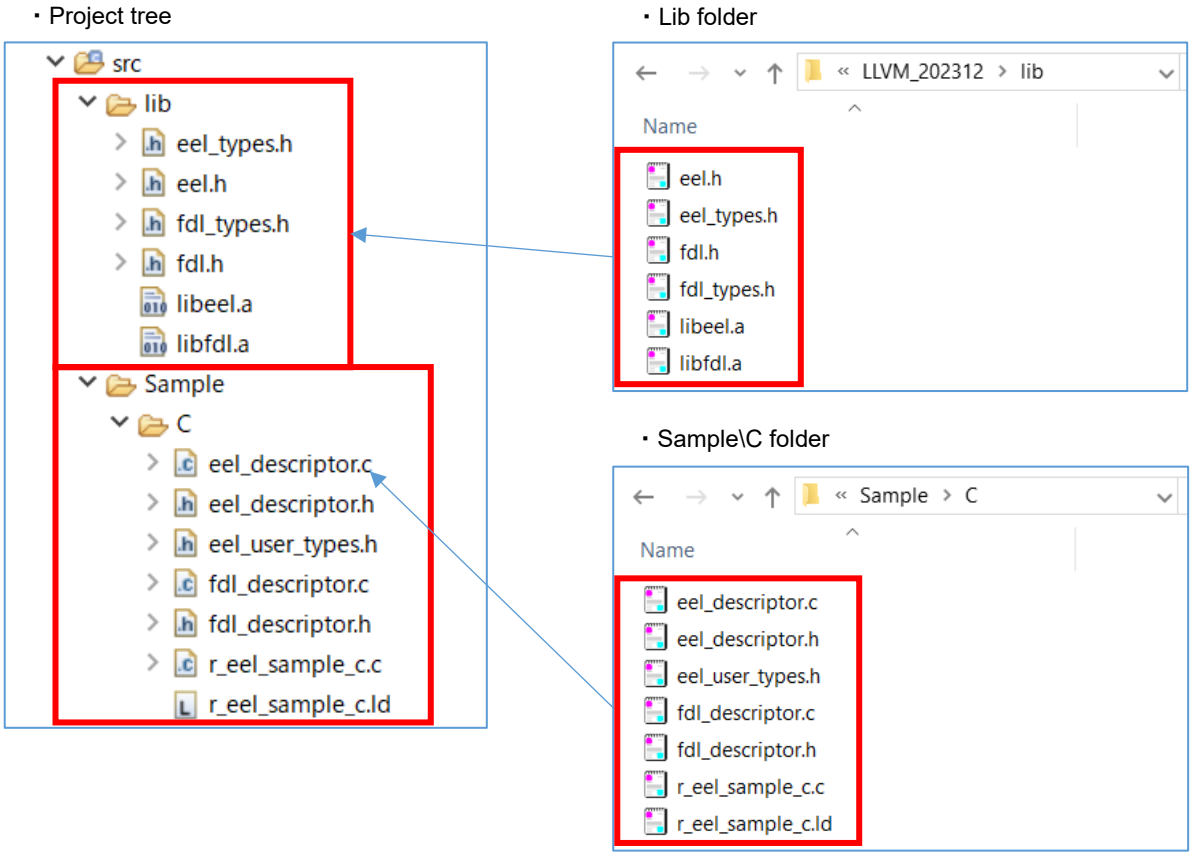

Figure 6-15. Specifying the Source and Include Files (LLVM)

Exclusion of the file automatically added by the function of e^2 studio.

There are files added automatically in the created project. The same file as these exists also in the "sample" folder of EEPROM Emulation Library Pack02. Therefore, using the function of IDE, Select those files from tree, and excludes from a project.
Clicks the right mouse button for the file of tree. And On the [Settings] screen displayed by the "Properties", put a check mark to [Exclude resource from build] and exclude a target file (target folder).

- CC-RL: Target file is [project name] .c (ex: "EELPack02_PJ01.c") in a [project name]/src folder.
- LLVM: Target files are "linker_script.ld" in a [project name]/generate folder, and [project name] .c ("EELPack02_PJ01.c") in a [project name]/src folder.

(2) Specifying the library files

- CC-RL: Click the right mouse button for the project in a tree, and select "Properties". In the "Add file" window that appears by clicking the "+" button to the right of "Relocatable files, object files, and library files" on the "C/C++ Build" [Settings] – "Linker" [Input] screen, change the [Format] to "library", and register the path to the library files "eel.lib" and "fdl.lib" (as shown in Figure 6-16).

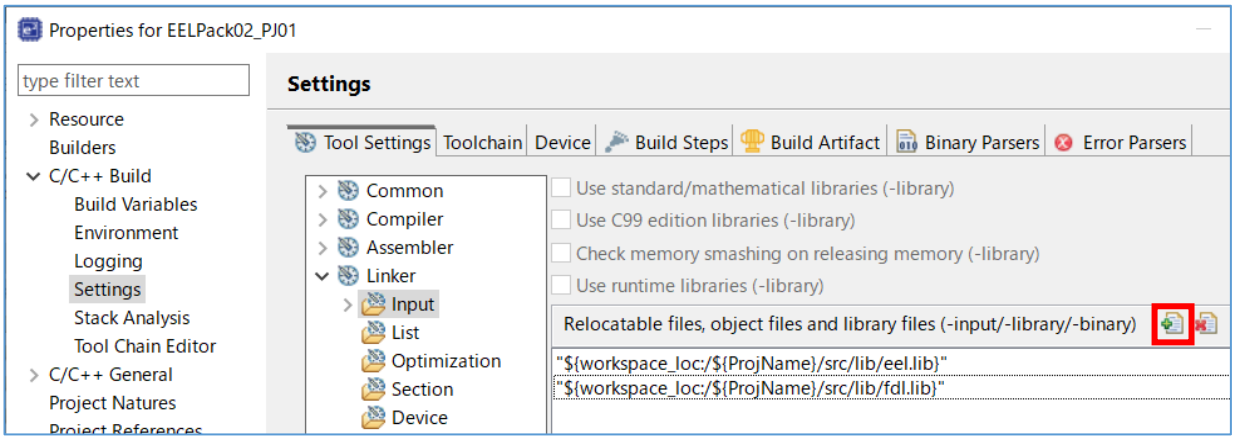

Figure 6-16 (a). Specifying the Library Files (CC-RL)

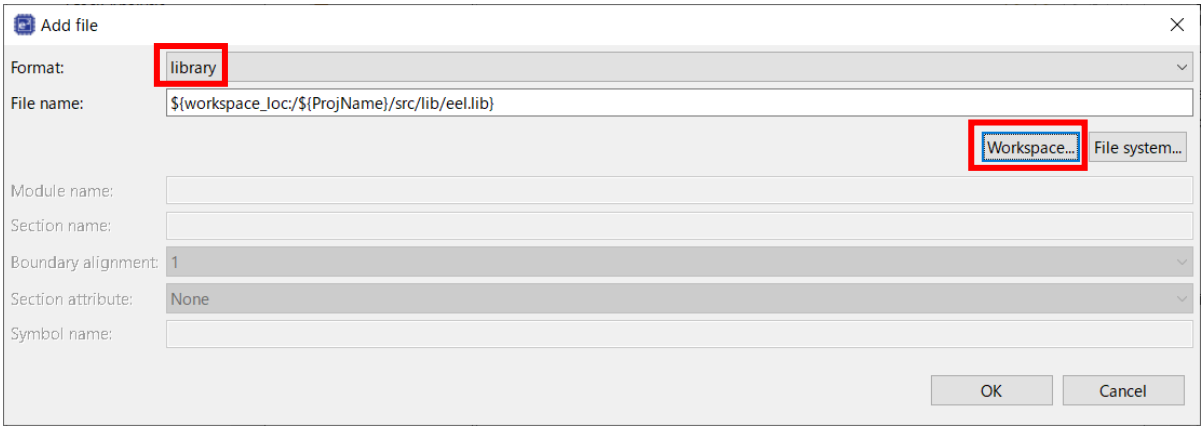

Figure 6-16 (b). Specifying the Library Files (CC-RL)

- LLVM: Click the right mouse button for the project in a tree, and select "Properties". Register the file path of the library files "libeel.a" and "libfdl.a" in the "Additional input files" field on the screen displayed in "C/C++ Build" [Settings] – "Linker" [Source] (as shown in Figure 6-17).

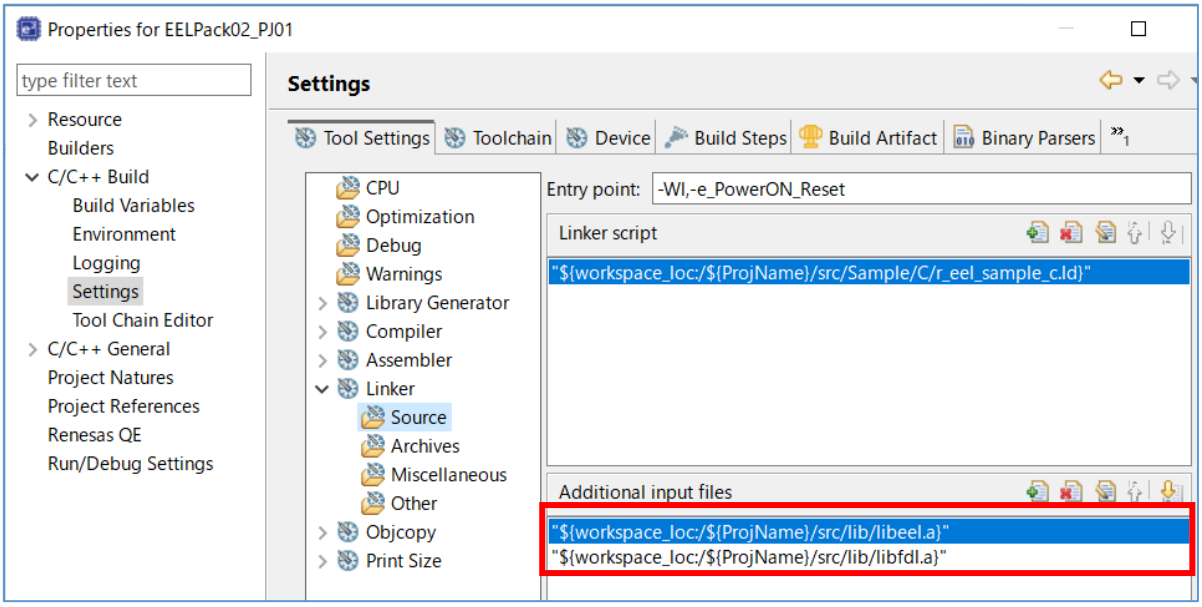

Figure 6-17. Specifying the Library File (LLVM)

(3) Specifying the linker script file (only when the LLVM compiler is used)

Click the right mouse button for the project in a tree, and select "Properties". Register the file path of the linker script file ".ld" in the "Linker script" field on the screen displayed in "C/C++ Build" [Settings] – "Linker" [Source] (as shown in Figure 6-18).

Here, select the file path of "r_eel_sample_c.ld" prepared for the EEPROM Emulation Library Pack02.

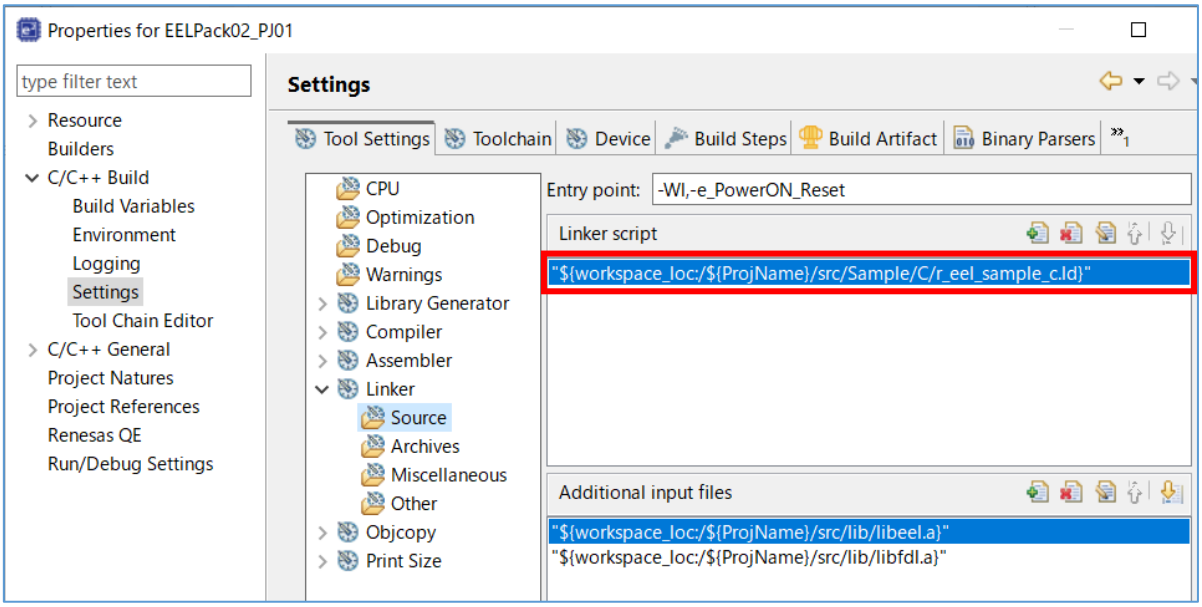

Figure 6-18. Specifying the Linker Script File (LLVM only)

Note: Refer to each reference manual of LLVM about the descriptive content of linker script file, and the details of the description method.

(4) Building

Right-click on the [Project] in the e^2 studio project tree and select "Build Project" to build the project.

6.3.3 Building an Assembly-Language Program (only when the CC-RL Compiler is used)

(1) Specifying the source and include files

Specifying the EEPROM emulation library and the data flash library files in the created project.

Register user program file ("xxxxxx.asm"), the EEPROM emulation library and the data flash library files "eel.inc", "fdl.inc", "eel.lib" and "fdl.lib" in the "src" folder output by e^2 studio (as shown in Figure 6-19).

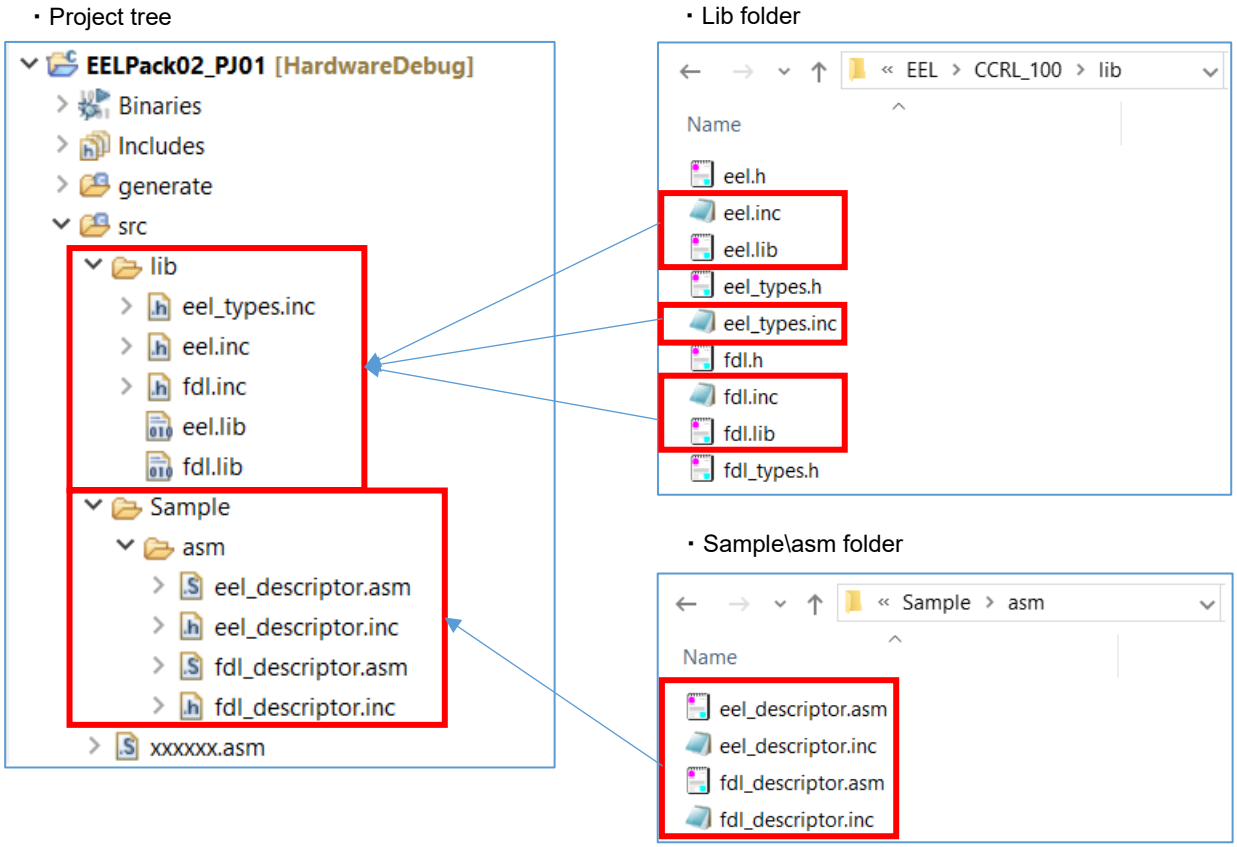

Figure 6-19. Specifying the Source and Include Files

- Exclusion of the file automatically added by the function of IDE.

There are files added automatically in the created project. The same file as these exists also in the "sample" folder of EEPROM Emulation Library Pack02. Therefore, using the function of IDE, Select those files from tree, and excludes from a project.

Clicks the right mouse button for the file of tree. And On the [Settings] screen displayed by the "Properties", put a check mark to [Exclude resource from build] and exclude a target file (target folder). (Exclusion of a folder is also possible)

Target file is [project name] .c (ex: "EELPack02_PJ01.c") in a [project name]/src folder.

(2) Specifying the library files

Click the right mouse button for the project in a tree, and select "Properties". In the "Add file" window that appears by clicking the "+" button to the right of "Relocatable files, object files, and library files" on the "C/C++ Build" [Settings] – "Linker" [Input] screen, change the [Format] to "library", and register the path to the library files "eel.lib" and "fdl.lib" (as shown in Figure 6-20).

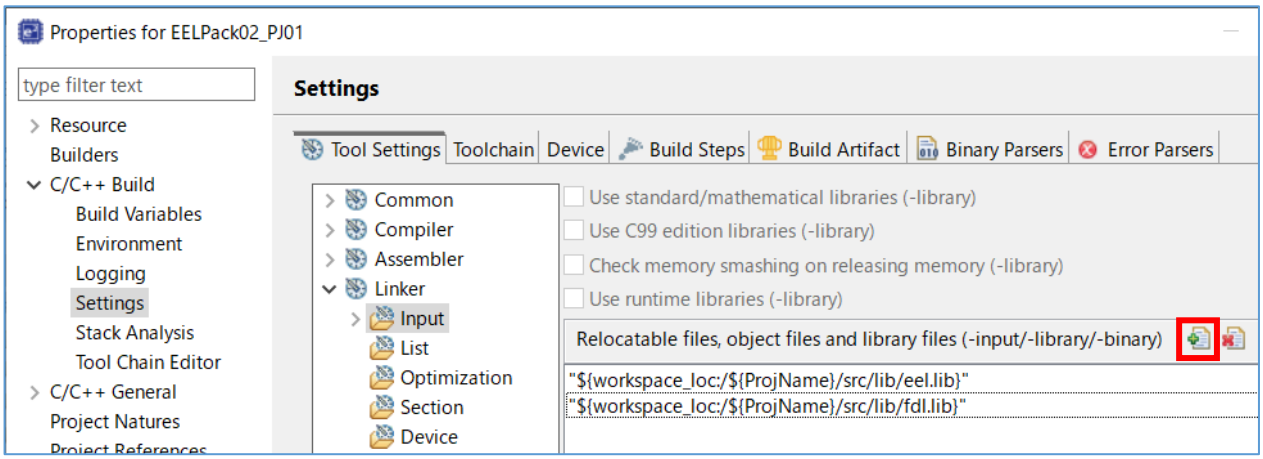

Figure 6-20 (a). Specifying the Library Files (CC-RL)

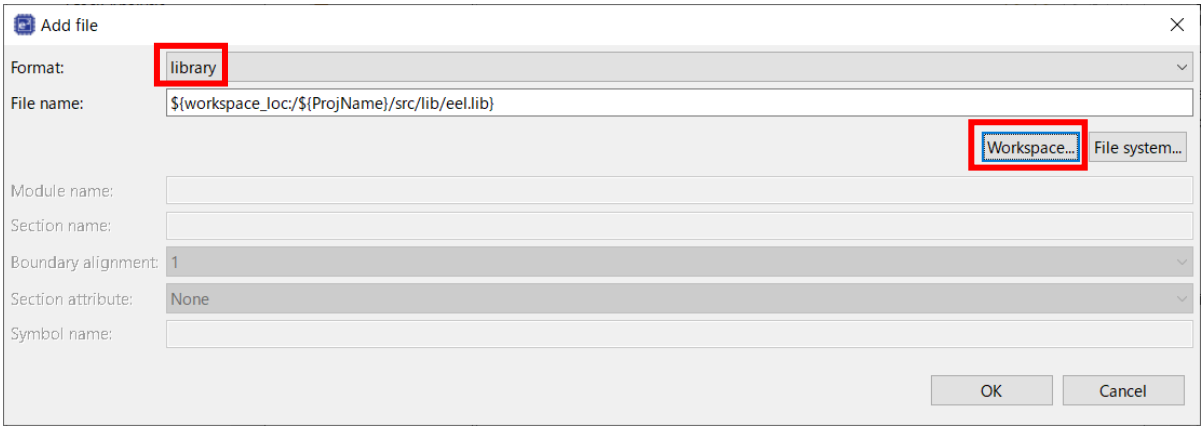

Figure 6-20 (b). Specifying the Library Files (CC-RL)

(3) Building

Right-click on the [Project] in the e^2 studio project tree and select "Build Project" to build the project.

6.4 Notes at Build

6.4.1 When the CA78K0R Compiler is Used

(1) When the on-chip debugging function is in use

After the on-chip debugging function is enabled in the CS+, building a program generates the following type of error.

```
RA78K0R error E3212: Default segment can't allocate to memory - ignored
                     Segment '??OCDROM' at xxxxxH-200H
```

This error occurs when the segment for the monitor area (OCDROM) used by the on-chip debugging function cannot be allocated. Therefore, to avoid this error, add the following code to the link directive file (*.dr) embedded in the project and prepare a separate area for allocating the segment.

```
MEMORY OCD_ROM : ( 0xxxxxH, 00200H )
```

Notes: 1. xxxxx: Start address of the location where the error occurred.
2. The area name "OCD_ROM" is an example of the notation.

6.4.2 When the CC-RL Compiler is Used

(1) When the on-chip debugging function is in use

After the on-chip debugging function is enabled in the CS+, building a program may generate the following type of error.

E0562321:Section ".monitor2" overlaps section "xxxxx"

Remark 1. xxxxx: Indicates the section name.

This error occurs when the section for the monitor area (OCDROM) used by the on-chip debugging function cannot be allocated. Therefore, to avoid this error, right click the CC-RL (Build Tool) node (1) in the CS+ Project Tree window, select Property to open the CC-RL Property panel (2), and select the Link Options tab (3). In the Section category (4), modify the setting for Section start address (5) so that no other areas overlap the area where the section for the on-chip debugger monitor is allocated (monitor2: the initial address range is 0xFE00 to 0xFFFF in R5F100LE). (as shown in Figure 6-21)

For details of the section settings, refer to the CC-RL Compiler User's Manual.

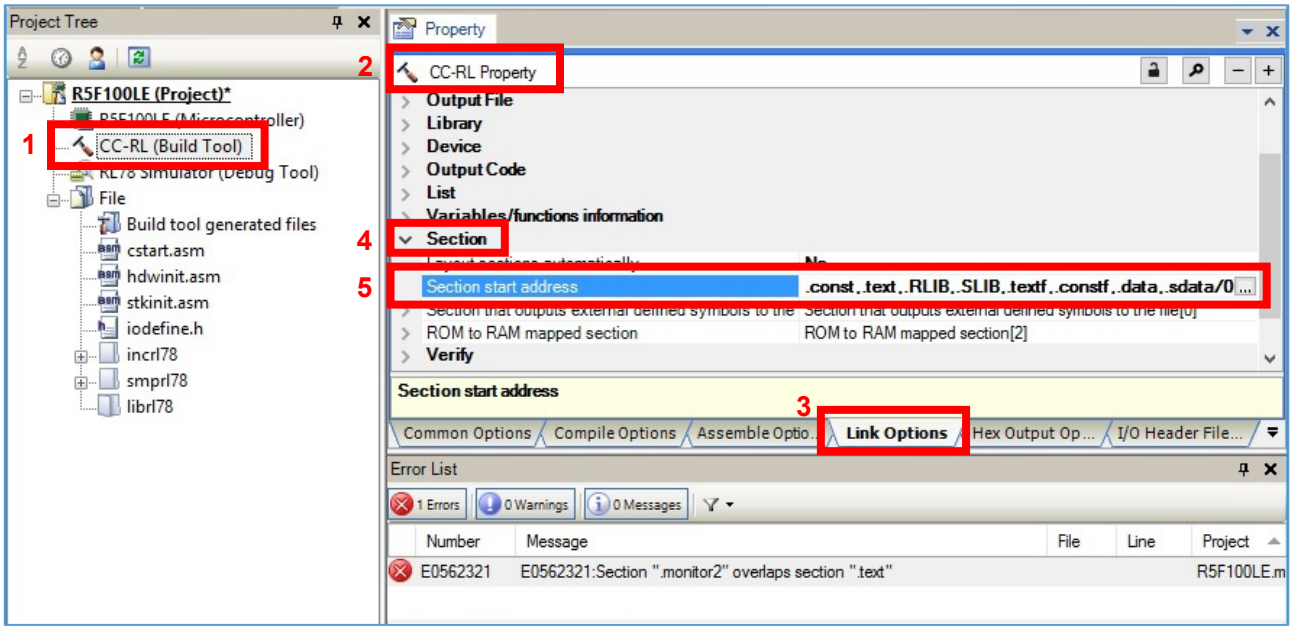

Figure 6-21. Modifying the Section Allocation

Chapter 7 How to Debug a Program

For details about how to perform debugging by using IECUBE or the on-chip debug emulator E1, E2, E2 emulator Lite or E20, see the following document.

Title
CubeSuite+ Integrated Development Environment User's Manual: RL78 Debug [CS+ for CA,CX]
CS+ Integrated Development Environment User's Manual: RL78 Debug Tool [CS+ for CC]
e² studio Integrated Development Environment User's Manual: Getting Started Guide

Note: You can download this document from the "CS+ Integrated Development Environment" or "e² studio Integrated Development Environment" page of the Renesas Electronics website.

7.1 Notes at Debug

The following describes notes apply when using the EEPROM Emulation Library Pack02 with the E1, E2, E2 emulator Lite or E20 on-chip debugging emulator.

(1) When a command of the EEPROM Emulation Library Pack02 is executed in a version older than CubeSuite+ Ver. 1.01 and the E1 or E20 on-chip debugging emulator is in use, do not execute a break until you have confirmed completion of the command by the sequencer. The sequencer will malfunction if a break occurs before the sequencer has completed the command.

(2) The flash library cannot be debugged by a simulator. To perform debugging, either use the on-chip debugging function of the RL78 microcontroller or prepare the IECUBE.

Chapter 8 Sample Program

The attached sample program (r_eel_sample_c.c) is provided to enable the usage method of the EEPROM Emulation Library Pack02 to be easily confirmed on the QB-R5F100LE-TB boards with R5F100LEA (RL78/G13) as the target microcontrollers. The sample program is just a reference example and the user program does not have to be created to match the sample program. The sample program should be used as a simple program to confirm operation.

- The link directive file (r_eel_sample_c.dr) for the sample program for the CA78K0R compiler has a purpose to specify that a stack or data buffer used by the sample program is not allocated to an area where allocation is prohibited Note1. When using the sample program, this file should also be embedded with the sample program.Note2
- The sample program for the CC-RL compiler, should be allocated appropriately in the section category on the "Link Options" tabbed page in the CS+ window, or on the "Linker" [Section] page in the e² studio, so that a stack or data buffer used by the sample program is not allocated to an area where allocation is prohibited Note1,2.
- The linker script file (r_eel_sample_c.ld) for the sample program for the LLVM compiler has a purpose to specify that a stack or data buffer used by the sample program is not allocated to an area where allocation is prohibited Note1. When using the sample program, this file should also be embedded with the sample program.Note2

Notes: 1. For details, refer to chapter "6.2 Software Resource" in the EEPROM Emulation Library Pack02 user's manual.
2. The data in usage may be placed at an unintended area depending on how the environment in use or the program is changed. After an execution module is generated, the map file (*.map) and allocation state of programs or data must be confirmed. For the definition method and allocation conditions of each code or data, refer to the user's manual of the compiler used.

8.1 Initial Settings of the Sample Program

The sample program operates with the following initial settings. When these settings need to be changed, modify the sample program.

- CPU operating frequency : High-speed on-chip oscillator 32 MHz
- Flash memory programming mode : Full-speed mode

8.2 Settings of Option Byte and On-Chip Debugging

(1) When using CA78K0R or CC-RL compiler with the CS+

When performing on-chip debugging, set "Set enable/disable on-chip debug by link option" to "Yes" and specify "84" for "Option byte values for OCD". For the property of the CC-RL compiler, set "Set debug monitor area" to "Yes".

The sample program normally operates by setting the high-speed on-chip oscillator at 32 MHz. After setting "Set user option byte" to "Yes" on select "Link Options" tab on CS+, specify "xxxxE8" for "User option byte value" (as shown in Figure 8-1).

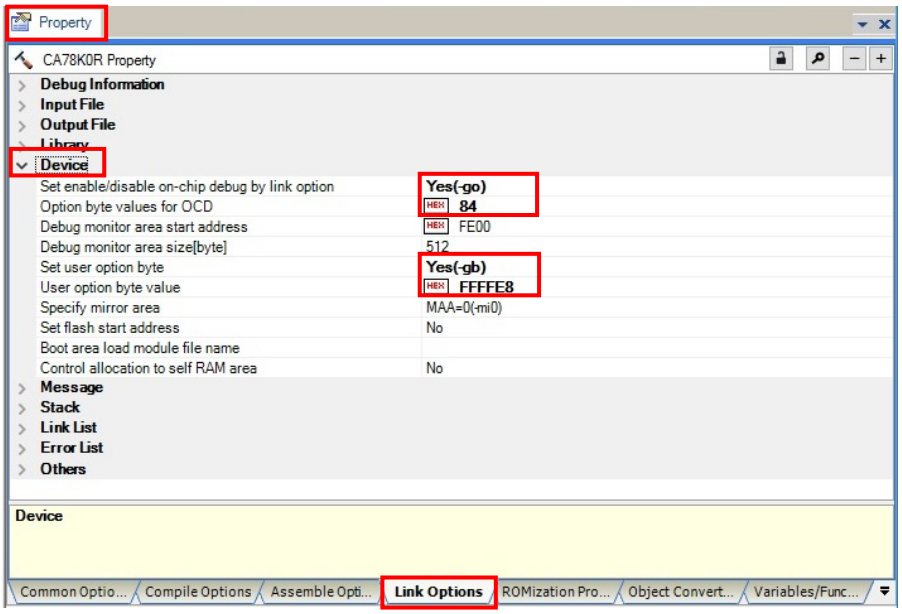

Figure 8-1 (a) Setting of Option Byte when Using the CS+ (CA78K0R Compiler)

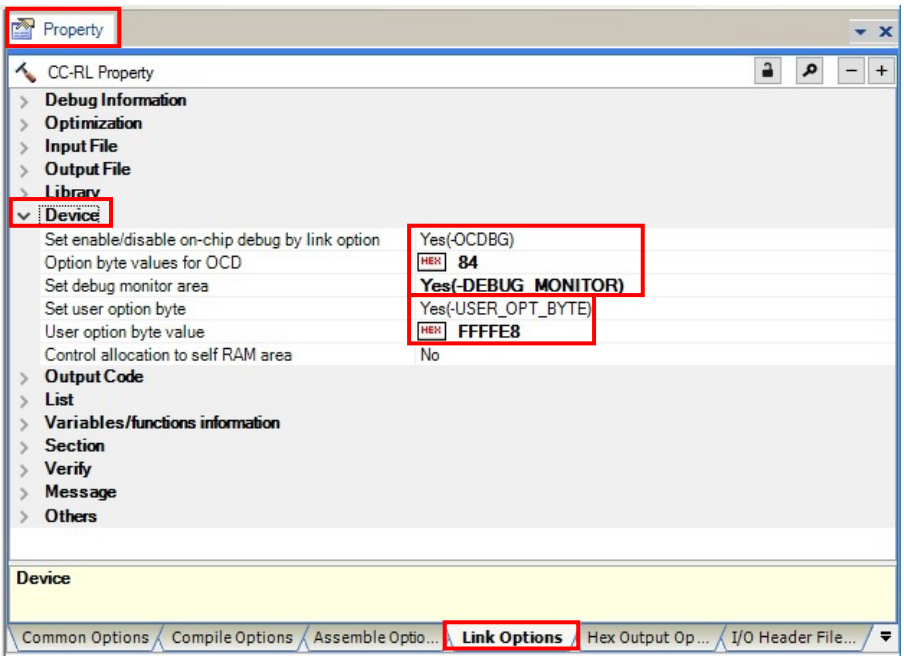

Figure 8-1 (b) Setting of Option Byte when Using the CS+ (CC-RL Compiler)

(2) When using CC-RL compiler with the e^2 studio

Select "C/C++ Build" [Settings] - "Linker" [Device]. And set device items on the displayed screen.

When performing on-chip debug, put a check mark to "Set enable/disable on-chip debug by link option" and specify "84" for "On-chip debug control value". Put a check mark to "Secure memory area of OCD monitor".

The sample program normally operates by setting the high-speed on-chip oscillator at 32 MHz. Put a check mark to "Set user option byte" on the "Tool Settings" tabbed page, specify "xxxxE8" for "User option byte value" and set the high-speed on-chip oscillator at 32 MHz (as shown in Figure 8-2).

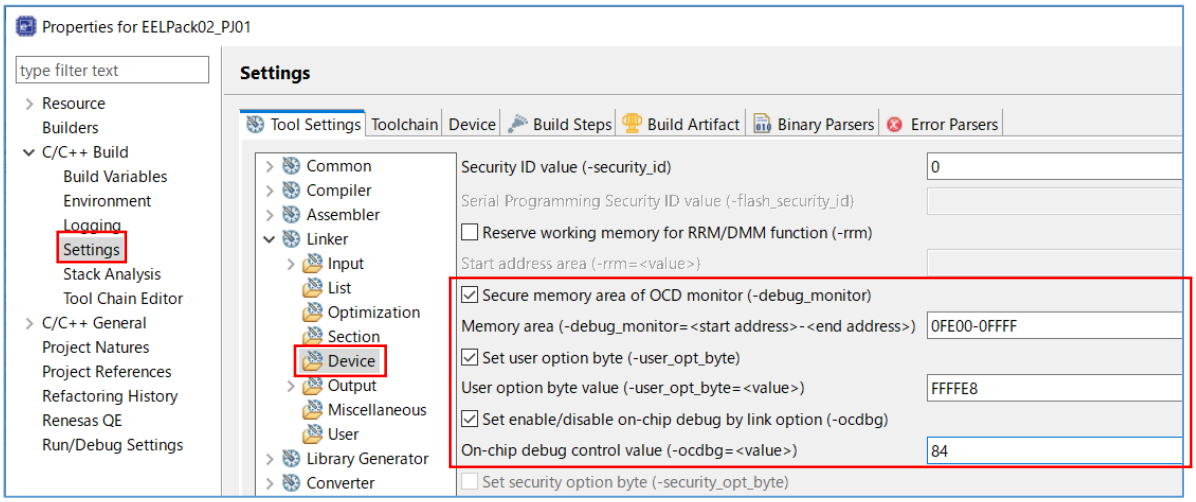

Figure 8-2 Setting of Option Byte when Using the e^2 studio (CC-RL Compiler)

(3) When using LLVM compiler with the e^2 studio

Device item settings are configured in the "vects.c" file output from e^2 studio.

Target file path: "Project Folder"¥generate¥vects.c

The sample program normally operates by setting the high-speed on-chip oscillator at 32 MHz. Therefore, set the user option byte value "xxxxe8" and the on-chip debug option byte value in the "Option_Bytes" of the "vects.c" file as follows:

[The example for RL78/G13]

"0xff, 0xff, 0xe8, 0x84" (WDT Enable, LVD reset mode, HS mode /32MHz, Enable on-chip debug operation)

```
const unsigned char Option_Bytes[]  __attribute__ ((section (".option_bytes"))) = {
        0xff, 0xff, 0xe8, 0x84
};
```

Note: Be sure to confirm the contents of "User option byte" of the chapter of "Option Bytes" and "On-chip debug option byte" by the user's manual of a target device. And describe the set value used with user application.

8.3 Defining the On-Chip RAM Area

8.3.1 When the CA78K0R Compiler is Used

The following describes how to define the on-chip RAM area in the link directive file.

Normally, the entire on-chip RAM area is automatically defined as an area with the name "RAM" unless otherwise stated in the link directive file. The stack and data buffers are to be allocated to this area except when specifically stated otherwise [Note]. However, in this case, the stack and data buffers would be allocated by default to an area (self-RAM and FFE20H to FFEFFH) for which use by the EEPROM Emulation Library Pack02 is prohibited, so the program may not run correctly.

In the attached link directive file for the sample program, as a solution, re-define the area with the name "RAM" so that it does not include the above area, ensuring that stack and so on are not allocated to the area for which usage is prohibited.

```
MEMORY RAM      :(0FF080H, 000DA0H)
```

The above statement redefines the area with the name "RAM" to be the DA0H bytes area starting from the address FF080H (FF080H to FFE1FH) [Note]. This prevents attempted use of the area which the EEPROM Emulation Library Pack02 is prohibited to use by excluding the prohibited portion from the area with the name "RAM".

However, if this is the only change setting that is explicitly made, the area from FFE20H to FFEFFH is also unusable for any other purpose. Accordingly, separately add the following definition. No particular restrictions apply to the name of this area.

```
MEMORY SADDR_RAM:(0FFE20H, 0000E0H)
```

If there is a self-RAM area, automatic allocation of variables to this area can be restricted by defining its range as an area with the name "SELFRAM".

```
MEMORY SELFRAM  :(0FEF00H, 000180H)
```

An example of the settings for an RL78/G13 (the product with 4 KB of RAM and 64 KB of ROM) is given below.

```
; ---------------------------------------------------------
; Define new memory entry for Self-RAM
; ---------------------------------------------------------
MEMORY SELFRAM   : ( 0FEF00H, 000180H )   <-- Definition of the self-RAM area

; ---------------------------------------------------------
; Redefined default data segment RAM
; ---------------------------------------------------------
MEMORY RAM       : ( 0FF080H, 000DA0H )   <-- Definition of the RAM area to be used normally

; ---------------------------------------------------------
; Define new memory entry for saddr area
; ---------------------------------------------------------
MEMORY RAM_SADDR : ( 0FFE20H, 0000E0H )   <-- Definition of the area from FFE20H to FFEFFH
```

Note: The CA78K0R linker allocates data with a non-specified destination for allocation (segment types DSEG and BSEG) to the on-chip RAM area according to the re-allocation attribute of the data. Accordingly, specific data may not be allocated to the area with the name "RAM" in some situations.
For details on the methods of defining and allocating the individual categories of data, refer to the user's manual for CS+.
Reference to the map file (*.map) generated at the time of building is required to confirm the state of allocation.

8.3.2 When the CC-RL Compiler is Used

(1) Adding the include path

In CS+ and e^2 studio, no include path is specified in the initial state: The include paths for the header files used by the EEPROM emulation library need to be added. The EEPROM emulation library uses header files "eel.h", "eel_types.h", "eel_user_types.h", "eel_descriptor.h", "fdl.h", "fdl_types.h", "fdl_descriptor.h" and "iodefine.h" (this file is automatically generated by CS+ and e^2 studio).

- In CS+, add the include path where each file resides in [Compile Options] – [Preprocessing] – [Additional Include Path].
- In e^2 studio, in the "Properties" window, add the include path where each file exists in the "Include file directories (-I)" field on the screen displayed by "C/C++ Build" [Settings] – "Compiler" [Source].

(2) Defining sections

The sections used for the ROM and RAM areas need to be defined.

- Sections can be defined in the Section category on the Link Options tab in the CS+ window. When the Layout sections automatically property is set to No, select the Section start address property to open the Section Settings dialog box and add the sections necessary for the EEPROM emulation library to the ROM area (as shown in Figure 8-3). (In this example, the FDL_CODE, FDL_CNST_f, EEL_CODE and EEL_CNST_f sections that are necessary for operation of the sample are added.) Also add a section of the EEPROM emulation library to the RAM area (as shown in Figure 8-4). (In this example, the FDL_SDAT and EEL_SDAT sections that are necessary for operation of the sample are added.)
  After adding sections, return Layout sections automatically property to Yes.

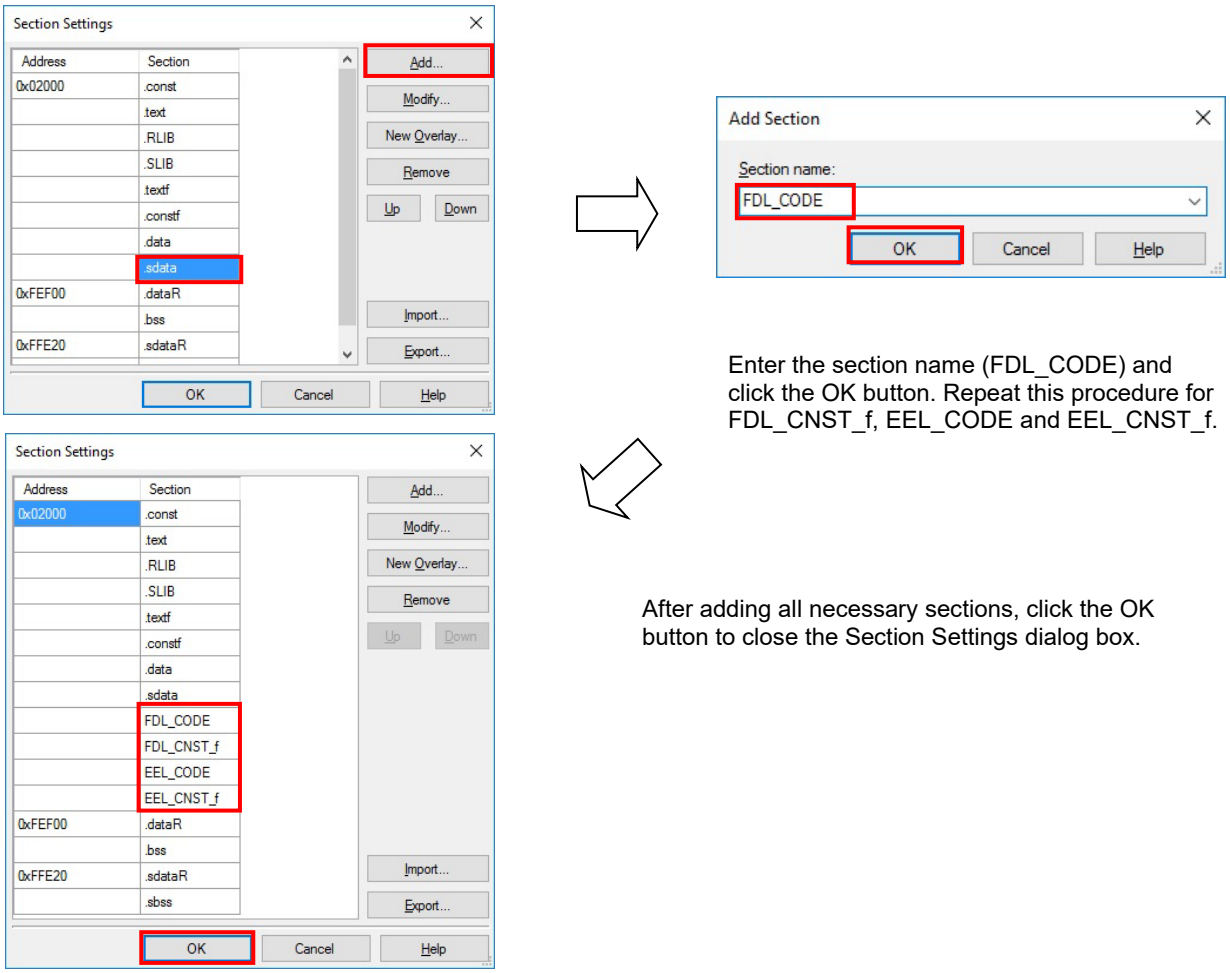

Figure 8-3. Example of Section Settings for EEPROM Emulation Library when Using CS+ (ROM Area)

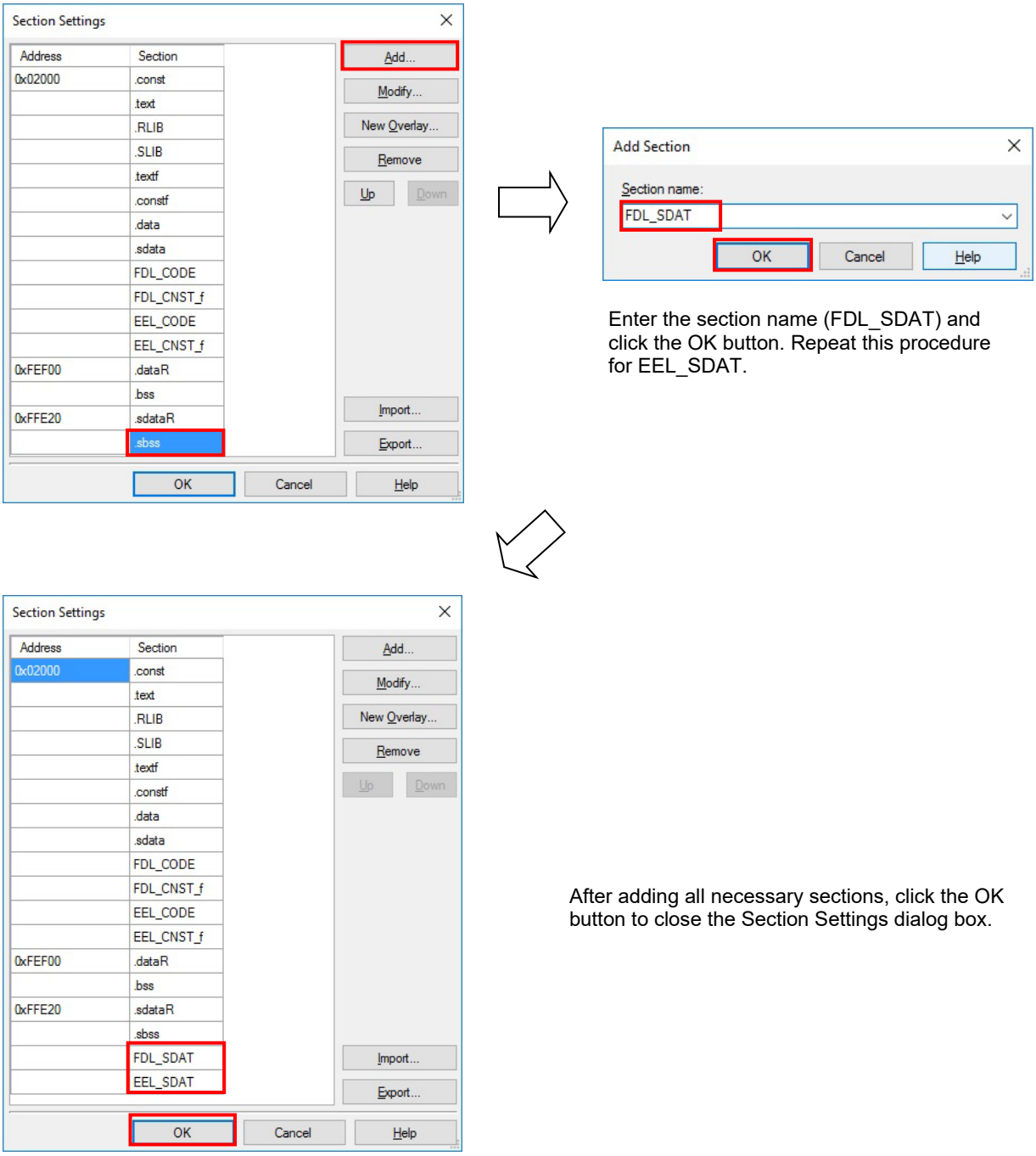

Figure 8-4. Example of Section Settings for EEPROM Emulation Library when Using CS+ (RAM Area)

- Setting of the section items on e^2 studio inputs in the "Properties" window. Select "C/C++ Build" [Setting] - "Linker" [Section]. And set section items on the displayed screen. Remove a check mark to [Layout sections automatically(-auto_section_layout)]. Press the "..." button of the right-hand side which sections are displaying, and a "Section Viewer" screen is displayed and add the sections necessary for the EEPROM emulation library to the ROM area (as shown in Figure 8-5). (In this example, the FDL_CODE, FDL_CNST_f, EEL_CODE and EEL_CNST_f sections that are necessary for operation of the sample are added.) Also add a section of the EEPROM emulation library to the RAM area (as shown in Figure 8-6). (In this example, the FDL_SDAT and EEL_SDAT sections that are necessary for operation of the sample are added.)

  After adding a section, check the [Layout sections automatically(-auto_section_layout)] checkbox.

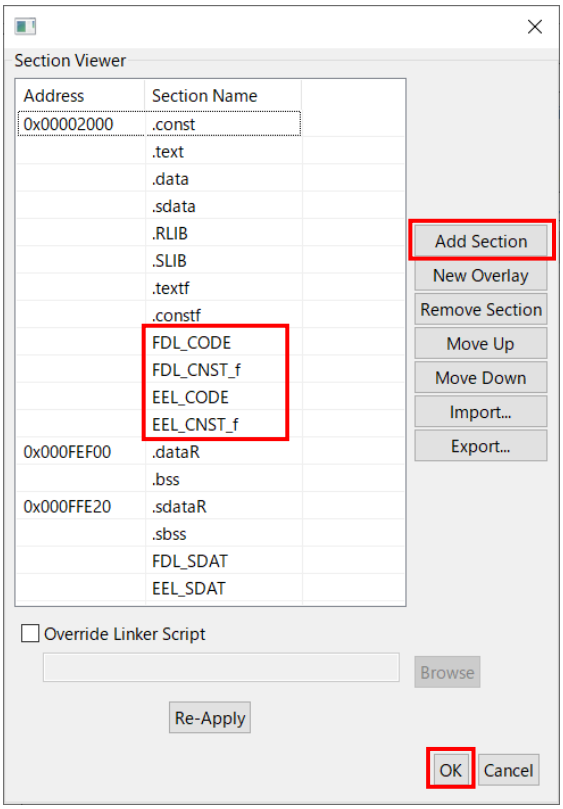

Figure 8-5. Example of Section Settings for EEPROM Emulation Library when Using e^2 studio (ROM Area)

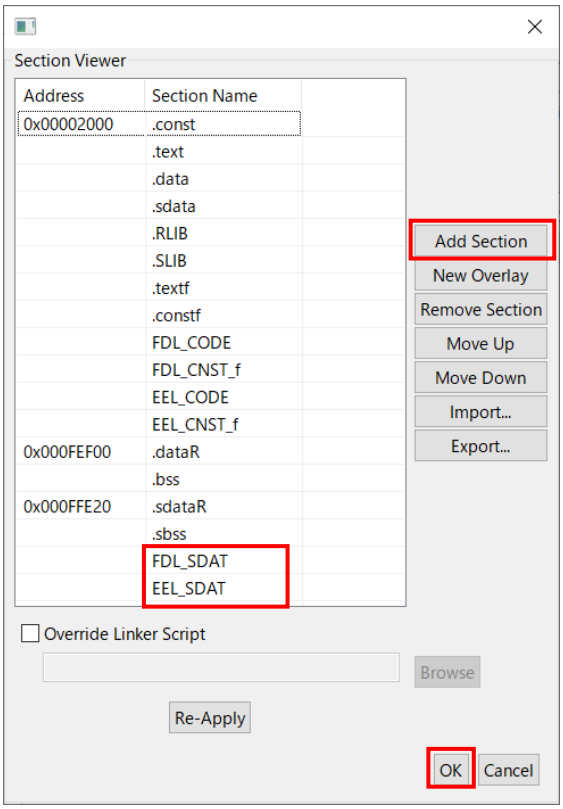

Figure 8-6. Example of Section Settings for EEPROM Emulation Library when Using e^2 studio (RAM Area)

(3) Allocating the Self-RAM Area

In the initial state of the section settings in CS+ for the CC-RL compiler, the user RAM area is allocated at the beginning of the internal RAM area (from address FEF00H for R5F100LEA, which is the target microcontroller of the sample program). However, in R5F100LEA, the EEPROM emulation library uses the address range from 0xFEF00 to 0xFF07F as the self-RAM area. Therefore, the user RAM area must be allocated outside this area. In this example, the user data start address 0xFEF00 is changed to 0xFF080 (as shown in Figure 8-7 or Figure 8-8).

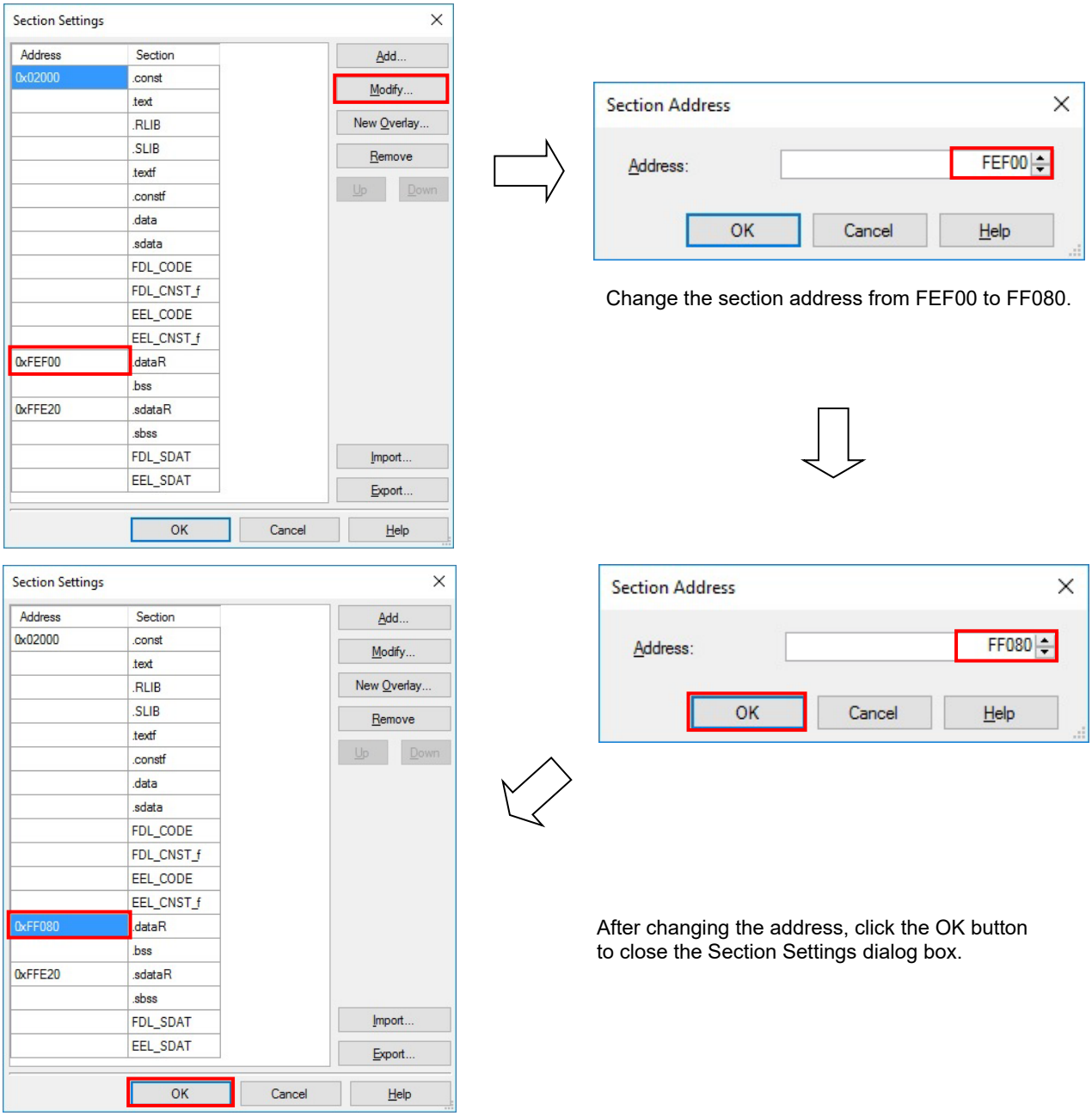

Figure 8-7. Example of Changing the User RAM Area Allocation when Using CS+ (RAM Area)

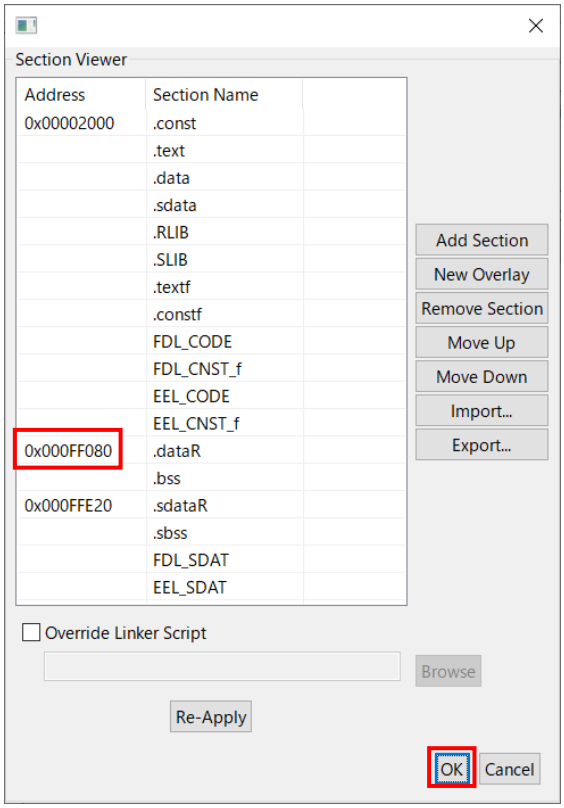

Figure 8-8. Example of Changing the User RAM Area Allocation when Using e² studio (RAM Area)

Note: When sections are automatically allocated again after the automatic section allocation setting has been removed and the user RAM area has been changed in the section settings, the sections will be automatically allocated again, including user-defined sections. In this case, sections may be allocated to areas that are not specified by the user; that is, data may be placed in unintended areas. Be sure to refer to the map file (*.map) to check if the software resources (especially RAM data) used by the EEPROM emulation library are placed in relocatable areas.

8.3.3 When the LLVM Compiler is Used

(1) Adding the include path

In e² studio, no include path is specified in the initial state: The include path for the header files used by the EEPROM emulation library need to be added. The EEPROM emulation library uses header files "eel.h", "eel_types.h", "eel_user_types.h", "eel_descriptor.h", "fdl.h", "fdl_types.h", "fdl_descriptor.h", and "iodefine.h" and "iodefine_ext.h" (these files are automatically generated by e² studio).

In e² studio, in the "Properties" window, add the include path where each file exists in the "Include file directories(-I)" field on the screen displayed by "C/C++ Build" [Settings] – "Compiler" [Includes].

(2) Allocating the Self-RAM Area

The LLVM compiler describes the link settings to be performed in the build in a linker script file (*.ld).
In the linker script file (linker_script.ld) output from e^2 studio, the built-in RAM area is defined as "RAM" section. In addition, the software resources used by the EEPROM emulation library are defined as an area called "SELFRAM" section.
(Only for devices that require "Self-RAM" area)
In the linker script file "r_eel_sample_c.ld" included with the sample program, the "RAM" section and "SELFRAM" section are defined so that they do not overlap.

Note: The "r_eel_sample_c.ld" provided in the sample program is prepared on the assumption that R5F100LE will be used. When using other devices, please check the Self-RAM list and modify it according to the device. Refer to each reference manual of LLVM about the descriptive content of linker script file (*.ld), and the details of the description method.

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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. 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.4.0-1 November 2017)

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/.