# RL78/I1C(512KB) Continuous Operation FOTA

## Continuous Operation FOTA Example Project

## Table of Contents

# Table of Figures

# Tables

# 1. Introduction

This document describes a sample application using Continuous Operation FOTA on RL78/I1C (512KB) and its software design.

**・What is Continuous Operation FOTA?**

RL78/I1C (512KB) is mainly used as a microcontroller for power metering (Metrology), and it has a function to update firmware without stopping the power metering function, which is called Continuous Metrology FOTA.

This sample software shows the application of Continuous Metrology FOTA to applications other than metering. For this reason, Continuous Metrology FOTA is replaced by Continuous Operation FOTA in this document.

**Continuous Metrology FOTA: firmware update without stopping the Metrology function**

**Continuous Operation FOTA: Firmware update without stopping a function of the microcontroller**

## 1.1. Assumptions and Advisory Notes

(1) Tool experience: It is assumed that the user has prior experience working with IDEs such as CS+ or e$^2$studio, and terminal emulation programs such as Tera Term.

(2) It is assumed that the user has basic knowledge about microcontrollers, embedded systems, and Code Generator in CS+ to create and modify the example project as described in this document.

(3) The images and screenshots provided throughout this document are for reference. The actual screen content may differ depending on the version of software or development tool.

## 1.2. Required Environments

**Hardware Requirements:** (Figure1-1)

- RL78/I1C (512KB) Fast Prototyping Board [RTK5RL10N0CPL000BJ] (“1” in the figure.)
- PMOD OLEDrgb [Digilent Pmod OLEDrgb (Revision B)] (96x64 RGB Display,“2” in the figure.)
- PMOD KYPD [Digilent PmodKYPD (Revision B)] (4x4 Keypad, “3” in the figure)
- Micro USB Device Cable (“4” in the figure.)
- PC with at least 1 USB port (“5” in the figure.)

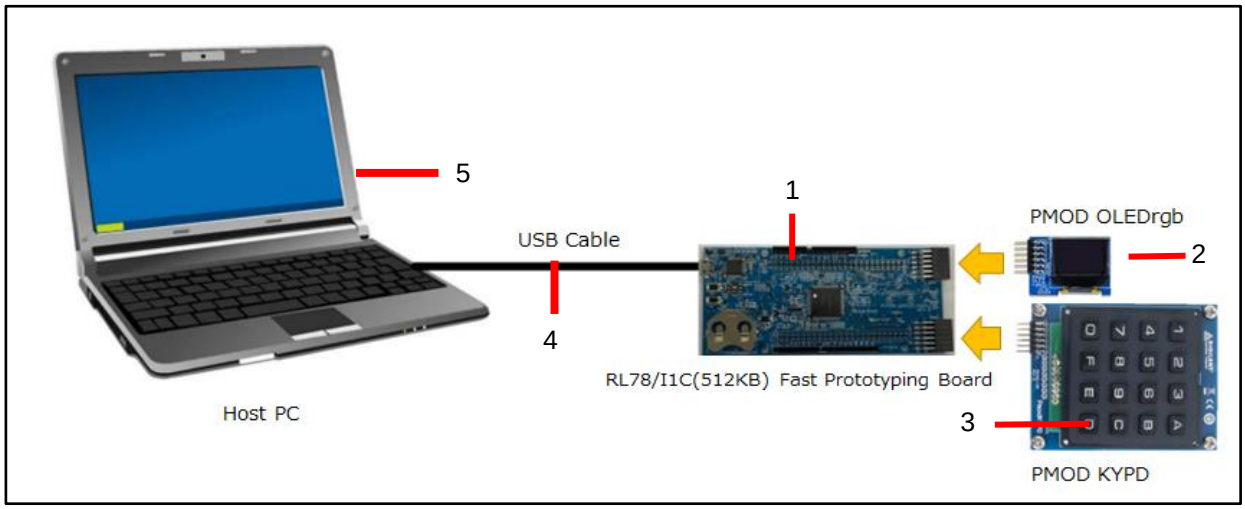


**Figure 1-1 : Hardware Requirements**

**Software Requirements:**

- Windows® 10 operating system
- USB Serial Drivers (included in Windows 10)
- Tera Term (or similar) terminal console application
- CS+ Ver. 9.06.00 or e2studio 2021-10
- CC-RL compiler V1.10.00
- Renesas Flash Programmer V3.08.03

## 1.3. Definition of Terms

**Table 1-1 : Definition of Terms.**

| | |
|---|---|
| FOTA | Firmware update Over-The-Air |
| Bank swap | There are two banks, bank 0 (256KB) for current execution area and bank 1 (256KB) for new firmware, swap the bank 1 and bank 0 by using the Bank Swap Library |
| Self-programming | The operation that takes place when the firmware update target is the firmware itself |
| FSL | Flash Self-programming Library |

# 2. Features of Continuous Operation FOTA Example Project

In this sample project, the function of judging the correctness of the passcode entered by the user and the LCD display function work. If you want to change the passcode, you need to update the authentication firmware. In this case, RL78/I1C(512KB) Continuous Operation FOTA allows you to update the firmware (change the passcode) while keeping the LCD display operation.

## 2.1. Passcode Correctness Judgment Function

It accepts passcode input from PMOD KYPD (4x4 keypad) and judges the passcode as correct or incorrect. The passcode consists of four digits (0-9) and letters (A-F) except for E. There is only one set of correct passcodes.

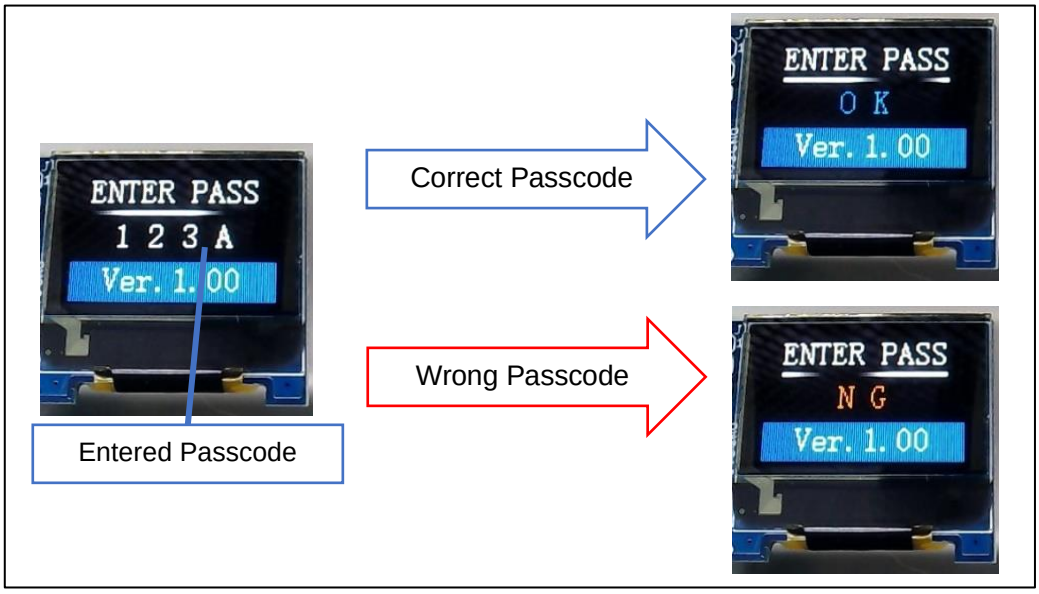


**Figure 2-1 : Passcode Authentication**

## 2.2. Screen Display Function

The indicator bar on the LCD (96x64 RGB Display) is always in operation even during the firmware update period. The LCD also displays the "ENTER PASS" message, the entered passcode, the result of correct/incorrect judgment and the current firmware version.

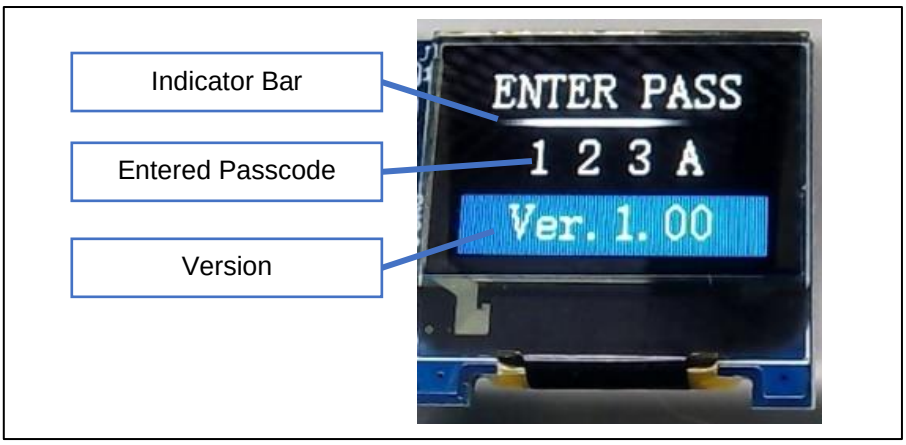


**Figure 2-2 : Screen Display Function**

## 2.3. FOTA Function

This sample project downloads and programs the new passcode judgment application C and switches the application from "B" to "C" without stopping the LCD display function (OLED control F/W). This Continuous Operation FOTA is realized by using two flash memory banks of RL78/I1C (512KB), bank swap function, and executing the program on RAM.

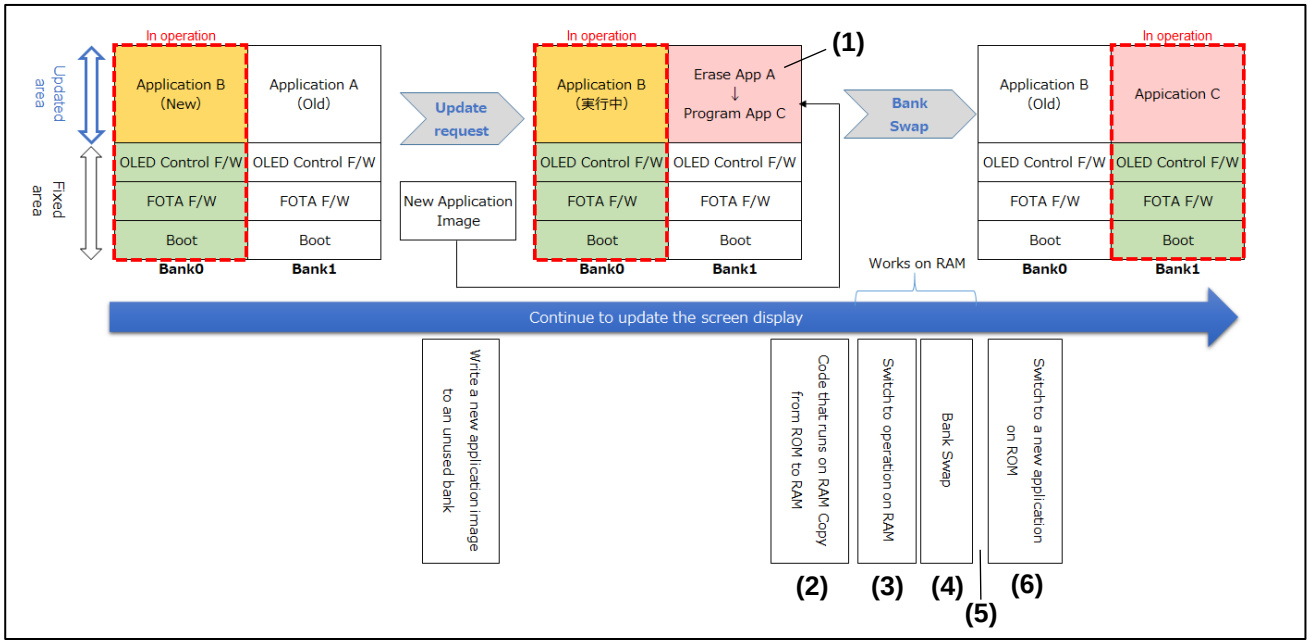


**Figure 2-3 : Internal Operation Flow Including Bank Swap**

- Application A : Previous applications that are not currently in use
- Application B : Currently running application
- Application C : New application
- OLED Control F/W : Firmware to update the display
- Boot : Bootloader

**<Operation Flow>**

(1) While application B is running, write new application C to an unused bank (Bank1) where the previous application A is located.

(2) After programming, the code to be executed on RAM (e.g., updating the LCD indicator bar) is extracted from ROM to RAM upon receiving the bank swap command.

(3) Switch the interrupt vector table to the one on RAM and switch to the operation on RAM.

(4) Bank swap is executed. Meanwhile, the program in RAM continues to operate.

(5) After the bank swap is completed, return the interrupt vector table to the one on ROM.

(6) Start the operation of the updated application.

# 3. Running the Continuous Operation FOTA Example Project

This chapter shows the operating procedure for the Continuous Operation FOTA sample.

## 3.1. Extracting the Packages

The sample project contains the following three subfolders, and the files in (1) and (2) are used to run the sample.

(1) RFP RI78I1C Production folder: contains the Renesas Flash Programmer project file [i1c_512k_production.rpj] and the "Ver.1.00" MOT file ([rl78i1c_production.mot]). For details on how to create the MOT file, refer to Chapter 4.8.

(2) New Application File folder: contains [rl78i1c_v100.mot] and [rl78i1c_v200.mot], which are "Ver.1.00" and "Ver.2.00" generated in Chapter 4.8, respectively.

(3) Source folder: Contains a set of software including configuration files and source code.

## 3.2. Programing the MCU

The following steps show how to program the MCU Flash.

(1) Set the on-board dip switch (SW3) to “Debug” and connect the Micro USB cable to the Micro USB connector on the RL78/I1C (512KB) Fast Prototyping Board.

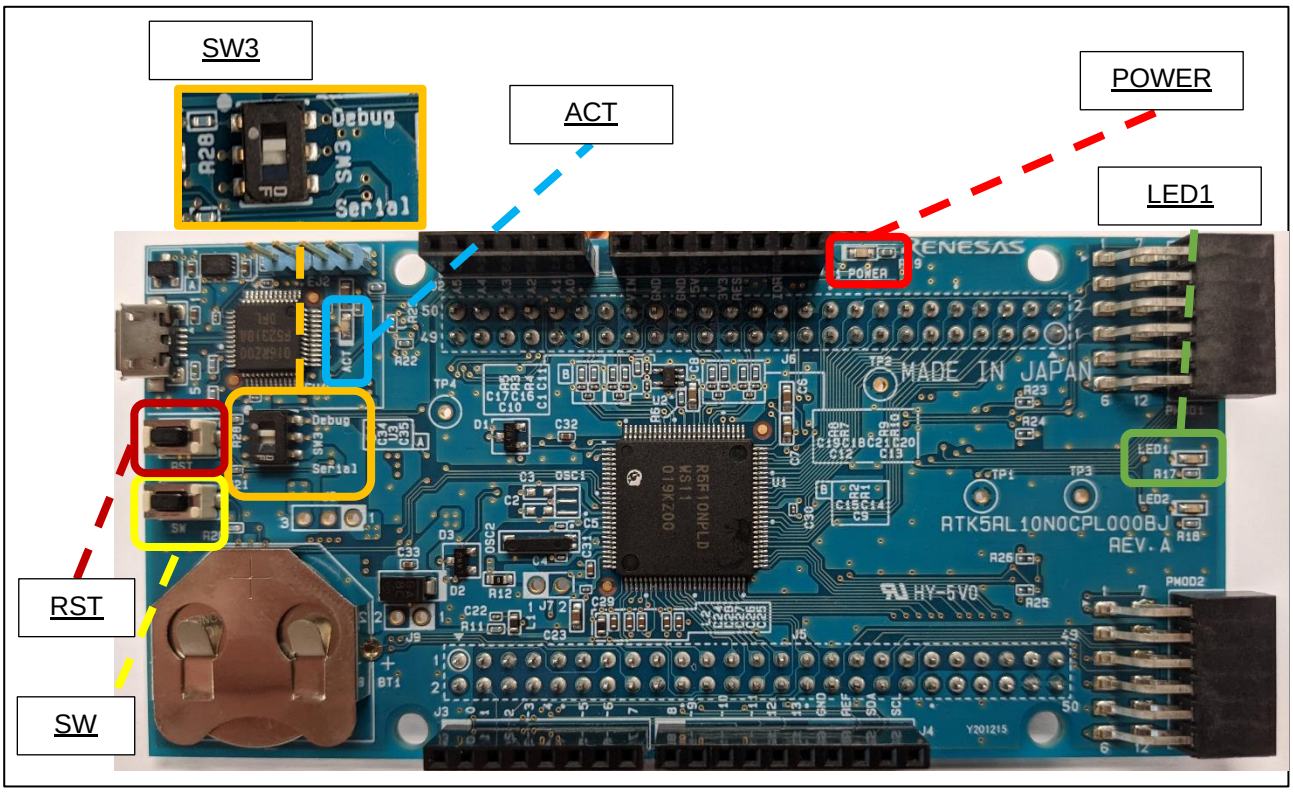


**Figure 3-1 : Outline of Fast Prototyping Board**

(2) Connect the other end of the Micro USB cable to the host PC. LED3 (POWER) ON.

(3) Open [i1c_512k_production.rpj] in Chapter 3.1 (1) in Renesas Flash Programmer.

(4) Select [rl78i1c_production.mot] and click the “Start” button to initiate the download.

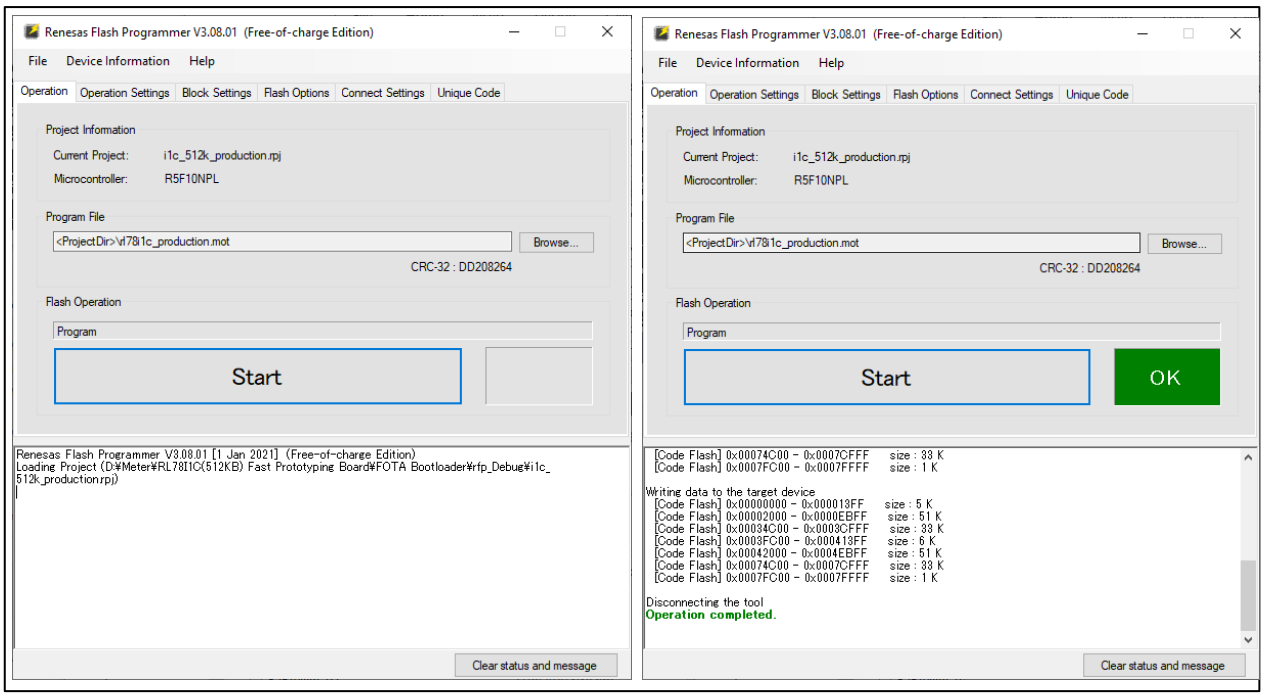


**Figure 3-2 : Renesas Flash Programmer**

## 3.3. Execution Procedure

To run the Example Operation Package, use the following instructions:

1. Set the on-board dip switch (SW3) to "**Serial**" and connect the Micro USB cable to the Micro USB connector on the RL78/I1C (512KB) Fast Prototyping Board.

2. Connect the Pmod OLEDrgb and Pmod KYPD to the RL78/I1C (512KB) Fast Prototyping Board.

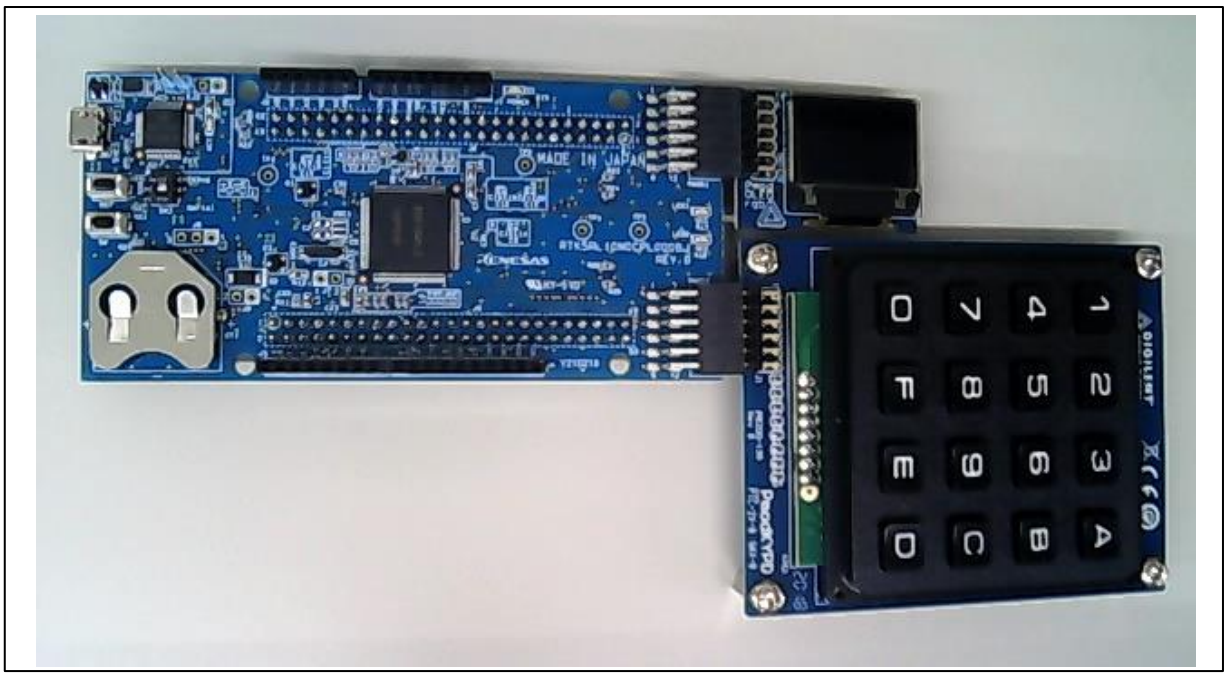

**Figure 3-3 : Peripherals connection to RL78/I1C (512KB) Fast Prototyping Board**

3. Connect the other end of the Micro USB cable to the host PC. LED3 (POWER) ON.

4. On the host PC, open Windows Device Manager. Expand **Ports (Com & LPT)**, located **USB Serial Device (COMxx)** and note down the COM port number for reference in the next step.

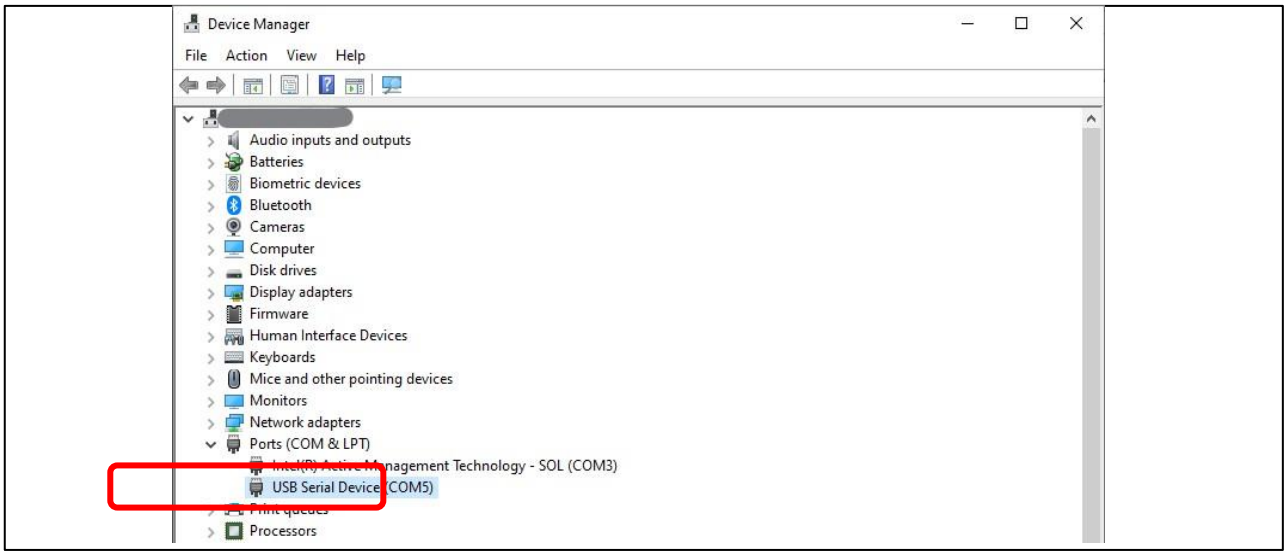


**Figure 3-4 : USB Serial Device in Windows Device Manager**

Note: USB Serial Device drivers are required to communicate between the RL78/I1C (512KB) Fast Prototyping Board and the terminal application on the host PC.

5. Open Tera Term, select **Serial** and **COMxx**: **Serial Device (COMxx)** and click **OK**.

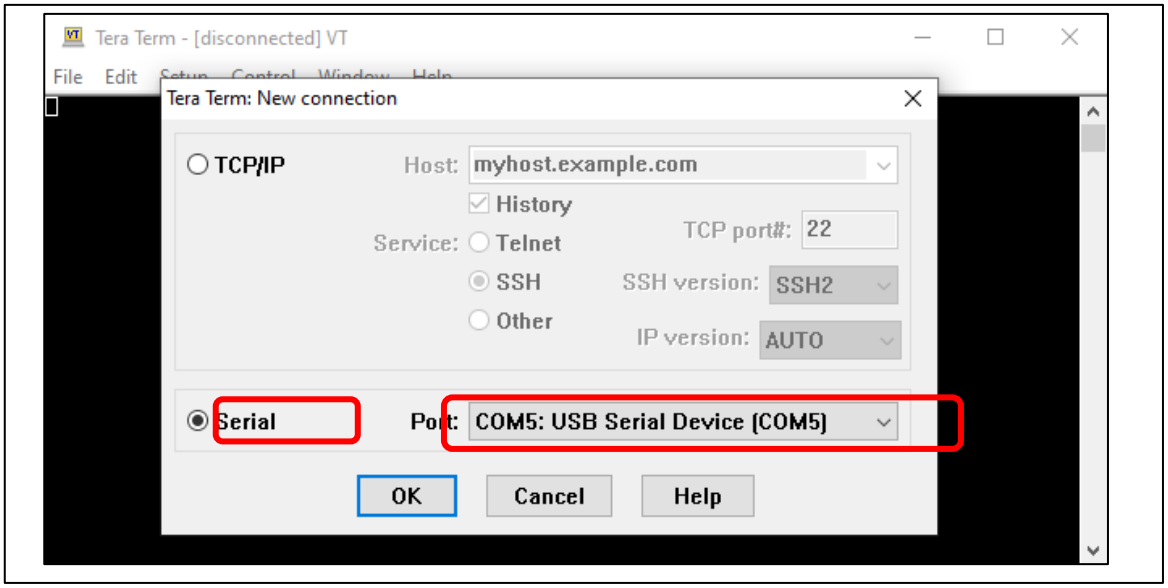


**Figure 3-5 : Selecting the Serial Port on TeraTerm**

6. In Tera Term, select **Setup** and **Serial port…** for the **Tera Term: Serial port setup and connection** window. Configure the setup as follows (**38400 baud, 8N1**) and click **New setting**.

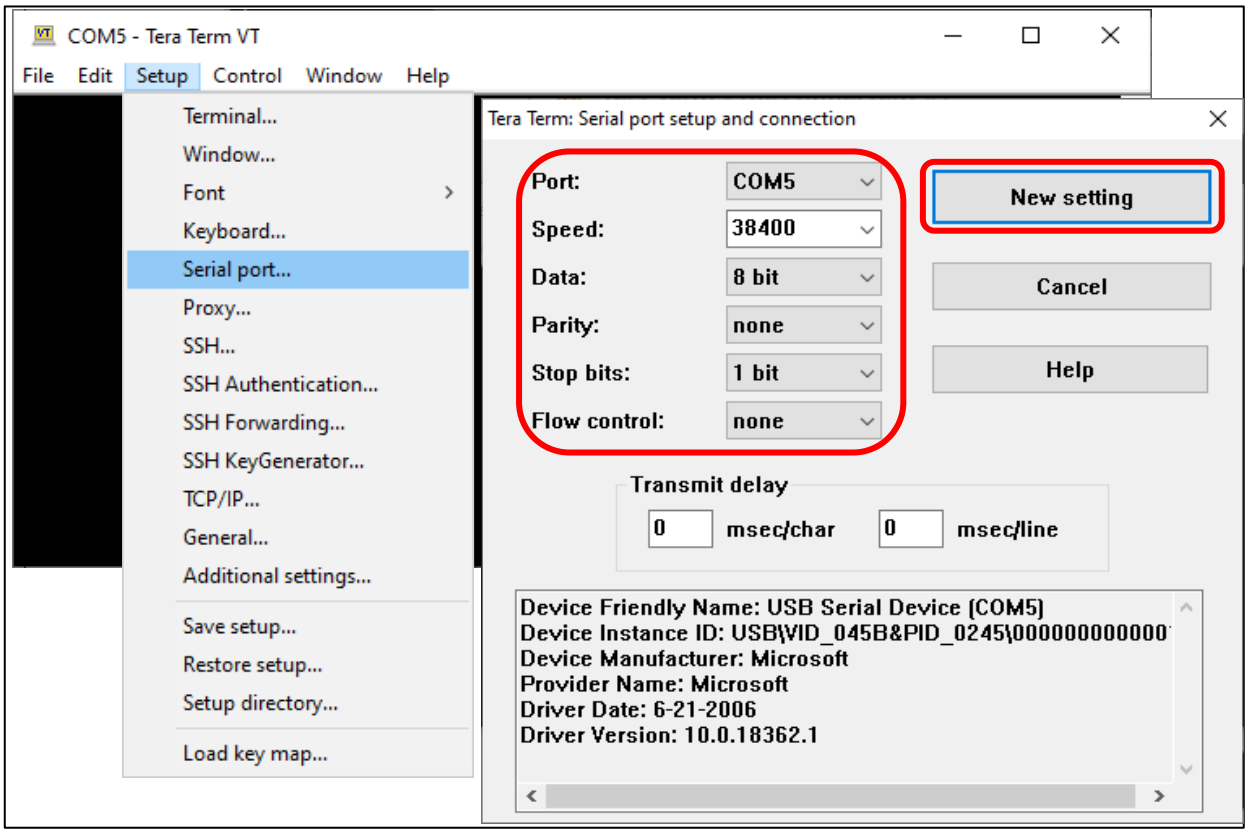


**Figure 3-6 : Setting up the Serial Port on Tera Term**

7. Press the on-board **RST** button on the RL78/I1C (512KB) Fast Prototyping Board. The start-up message is displayed on Tera Term.

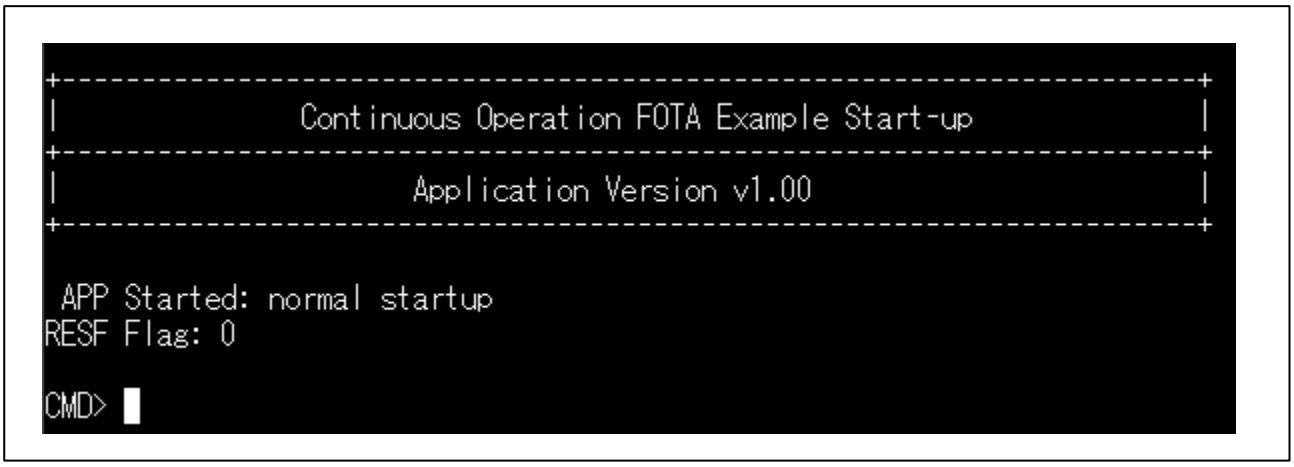


**Figure 3-7 : Start-up Message displayed on Tera Term**

8. Type “?” and press Enter key to observe the possible commands.

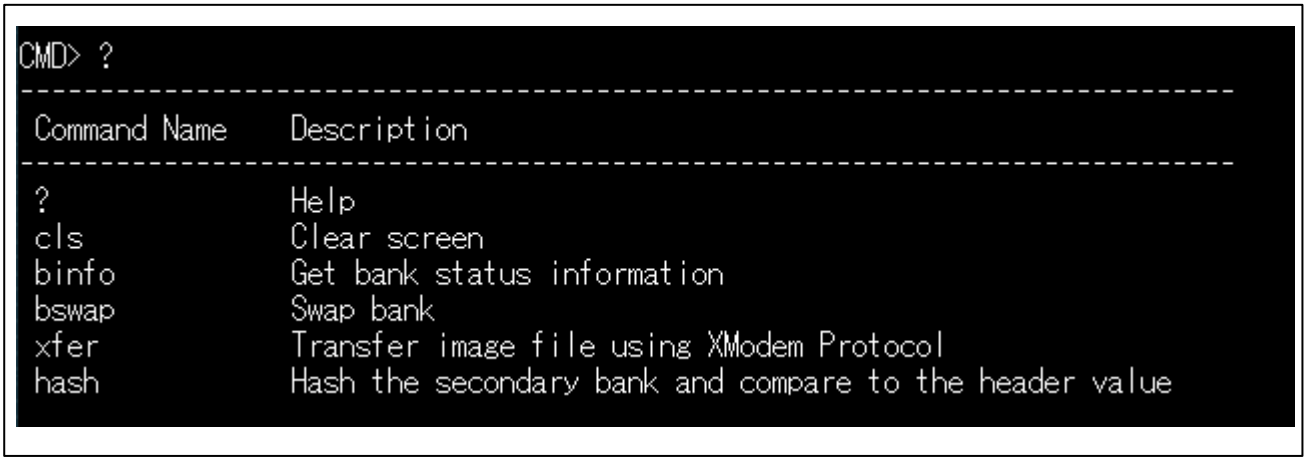


**Figure 3-8 : Available Commands in Continuous Operation FOTA**

9. Type the command and press Enter key to execute the function.

The usage and operation of each command will be explained in the next section.

### 3.3.1. Image Transfer

This section describes the image file transfer of the Continuous Operation FOTA sample project. The file transfer function is tested by using Tera Term v4.106.

- The Image Transfer is performed using the XMODEM checksum protocol.
- Variations such as XMODEM CRC are not supported.

1. Send the “xfer” command from Tera Term and select the XMODEM transfer file(rl78i1c_v200.mot). In the XMODEM protocol, data transfer begins with the receiver sending a NAK to the sender. After executing the “xfer” command, the receiving user application is put into a state of sending NAK every 10 seconds so that the file transfer starts.


**Figure 3-9 : Starting the Image Transfer**

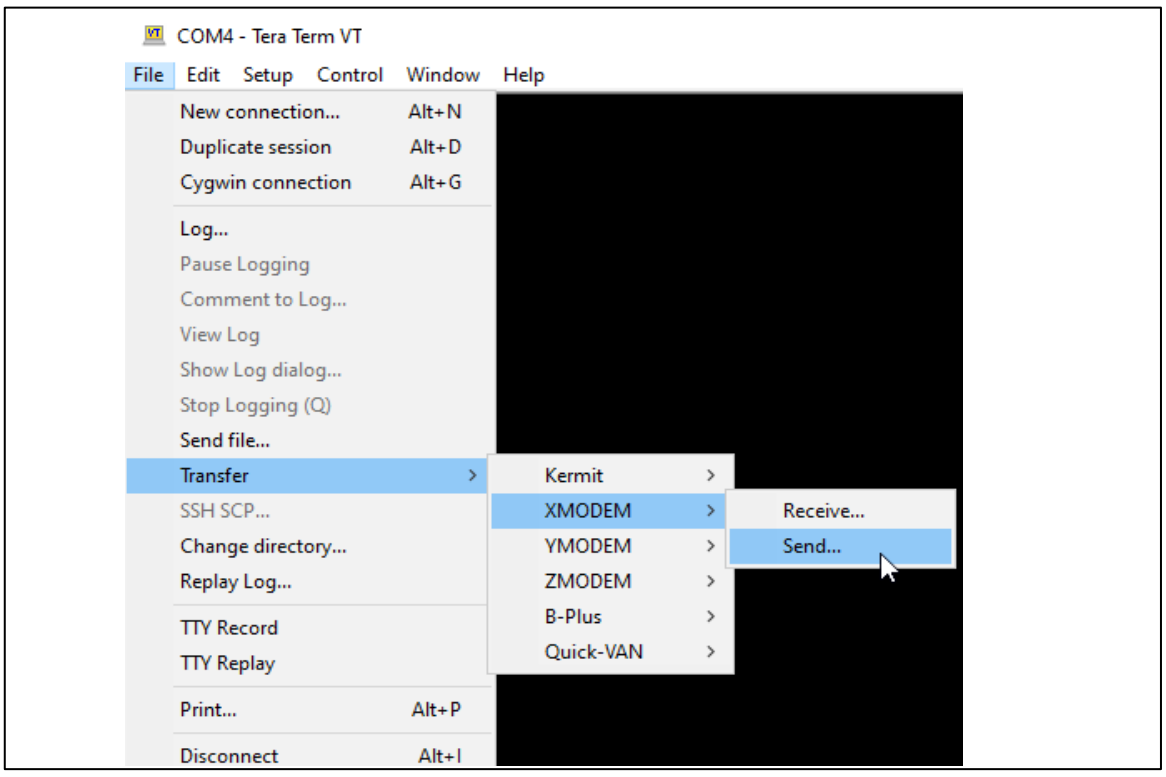


**Figure 3-10 : Select XMODEM**

2. On receiving the NAK, Tera Term will initiate the data transfer.

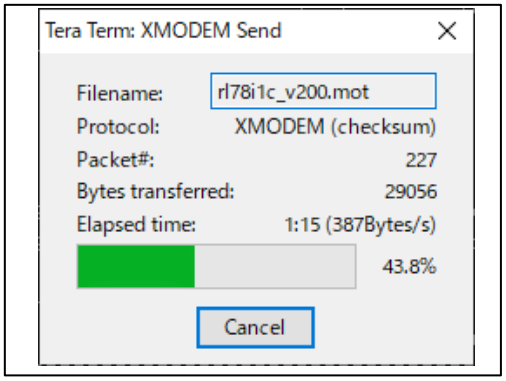


**Figure 3-11 : XMODEM Dialog**

3. When the user application receives a block data, the data is verified with the checksum. If the verification is successful, the data is written in the Flash, the LED1 on the RL78/I1C(512KB) Fast Prototyping Board blinks, and the user application sends back ACK to Tera Term. This process is repeated until all the data has been transferred.

4. After all the data transfer completes successfully, the message is displayed as shown in Figure 3-12.

```
CMD> xfer
Please start file transfer using XModem protocol.
Transfer will initiate within 10 seconds.

End of File.
Updating Image Header...
Transfer Complete.
```

**Figure 3-12 : Data Transfer Complete Message**

5. The “hash” command can be used to verify the hash value of the transferred application image.

```
CMD> hash
Hash value OK
Calculated program hash: 0x6c30
Embedded program hash:   0x6c30
```

**Figure 3-13 : “hash” Command**

6. The “binfo” command can be used to see the software version, etc.

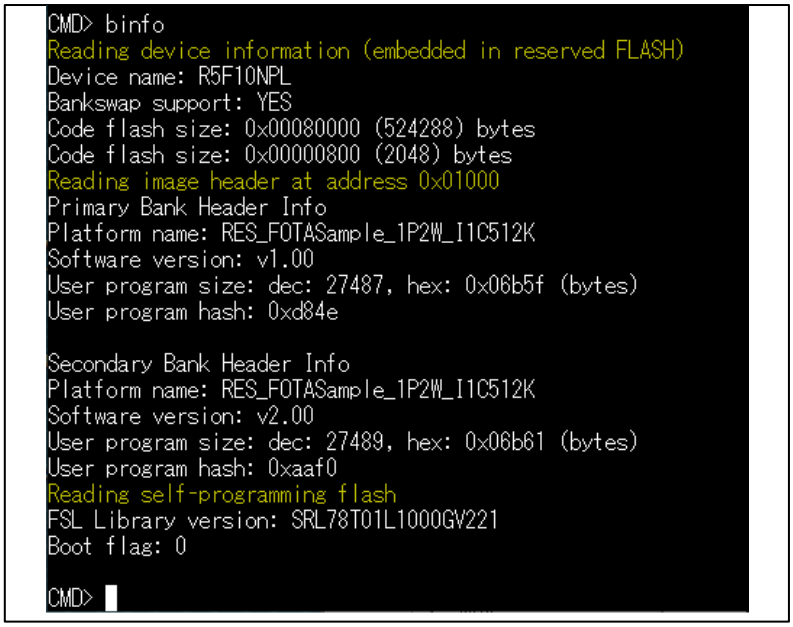


**Figure 3-14 : “binfo” Command**

7. The transferred User Application image can be activated using the “bswap” command described in the next section.

### 3.3.2. Bank-Swap Function

The “bswap” command initiates Flash bank switching. The very fact that the firmware has been updated is confirmed by the following.

■ Before “bswap” command

In Pmod KYPD, enter the correct passcode for Ver.1.00 ( 1 2 3 A in this sample), and press "E", "OK " will be displayed.

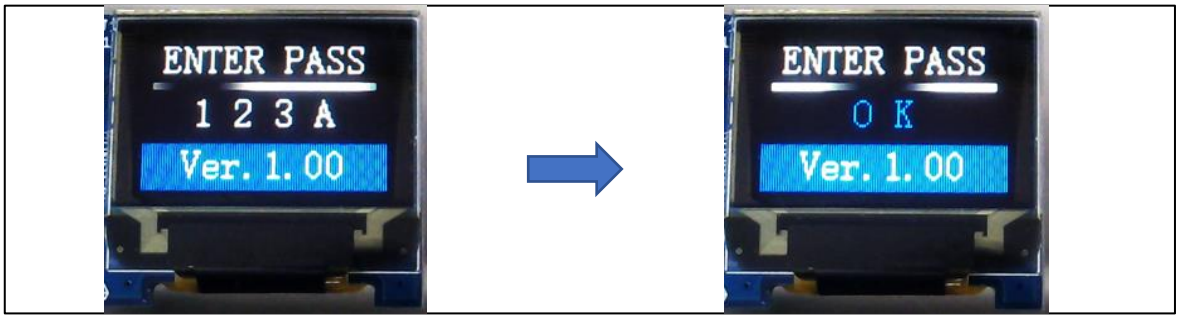


**Figure 3-15 : Authentication OK on Ver. 1.00**

■ Execute “bswap” command

When you execute the "bswap" command, the flash bank in the RL78/I1C (512KB) is switched, and the operation with the new firmware (Ver.2.00) starts immediately.

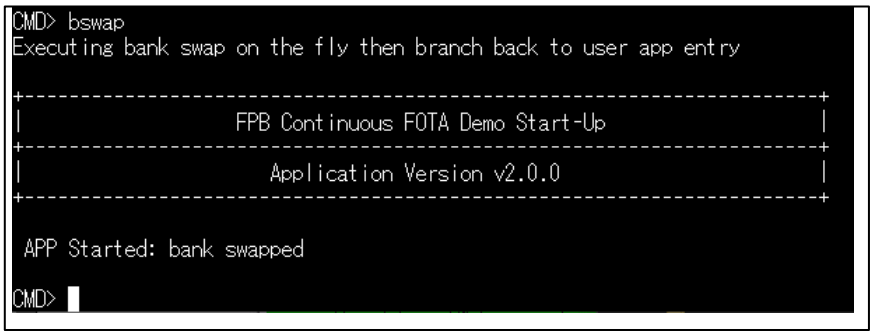


**Figure 3-16 : “bswap” Command**

On the other hand, you can also see the change to Ver.2.00 on the LED display.

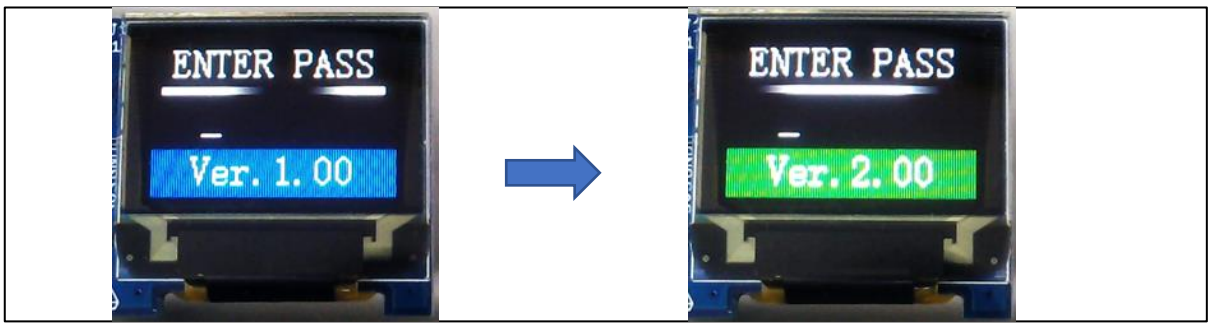


**Figure 3-17 : Version display change by "bswap" command**

In this sample, the passcode of Ver.1.00 is "123A" and the passcode of Ver.2.00 is "456B", so the result of passcode authentication in Ver.2.00 firmware is shown in Figure 3-18 and Figure 3-19 respectively.

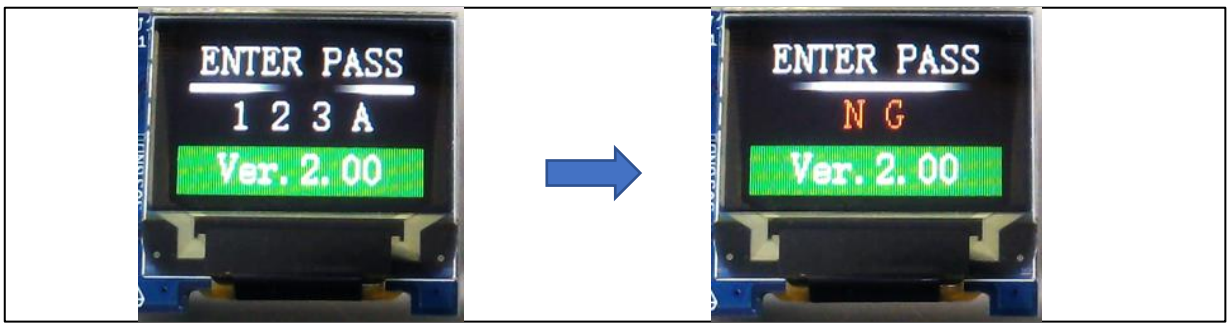


**Figure 3-18 : Authentication NG on Ver.2.00**

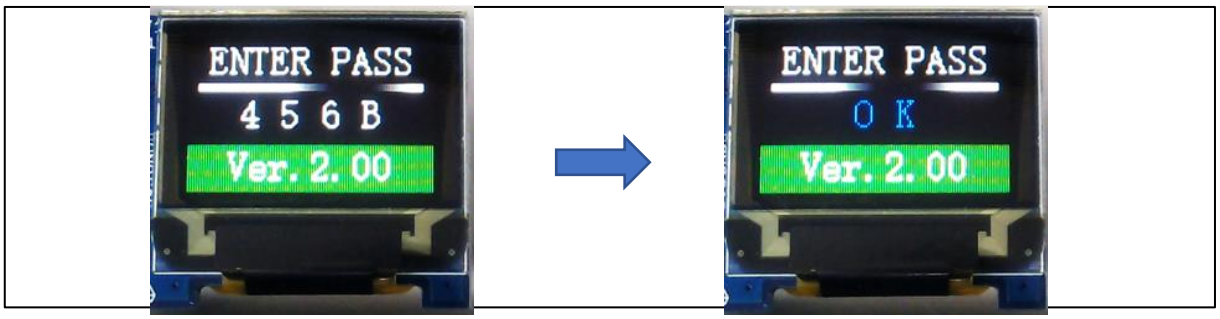


**Figure 3-19 : Authentication OK on Ver.2.00**

# 4. Project Settings

This chapter describes the project settings and operations in this example project.

## 4.1. Project Configuration

This sample project consists of three projects. The main project is the user application project to be updated by the FOTA function. The middleware subproject contains the programs related to the screen display functions. The bootloader subproject contains the bootloader functions and the bootloader library.

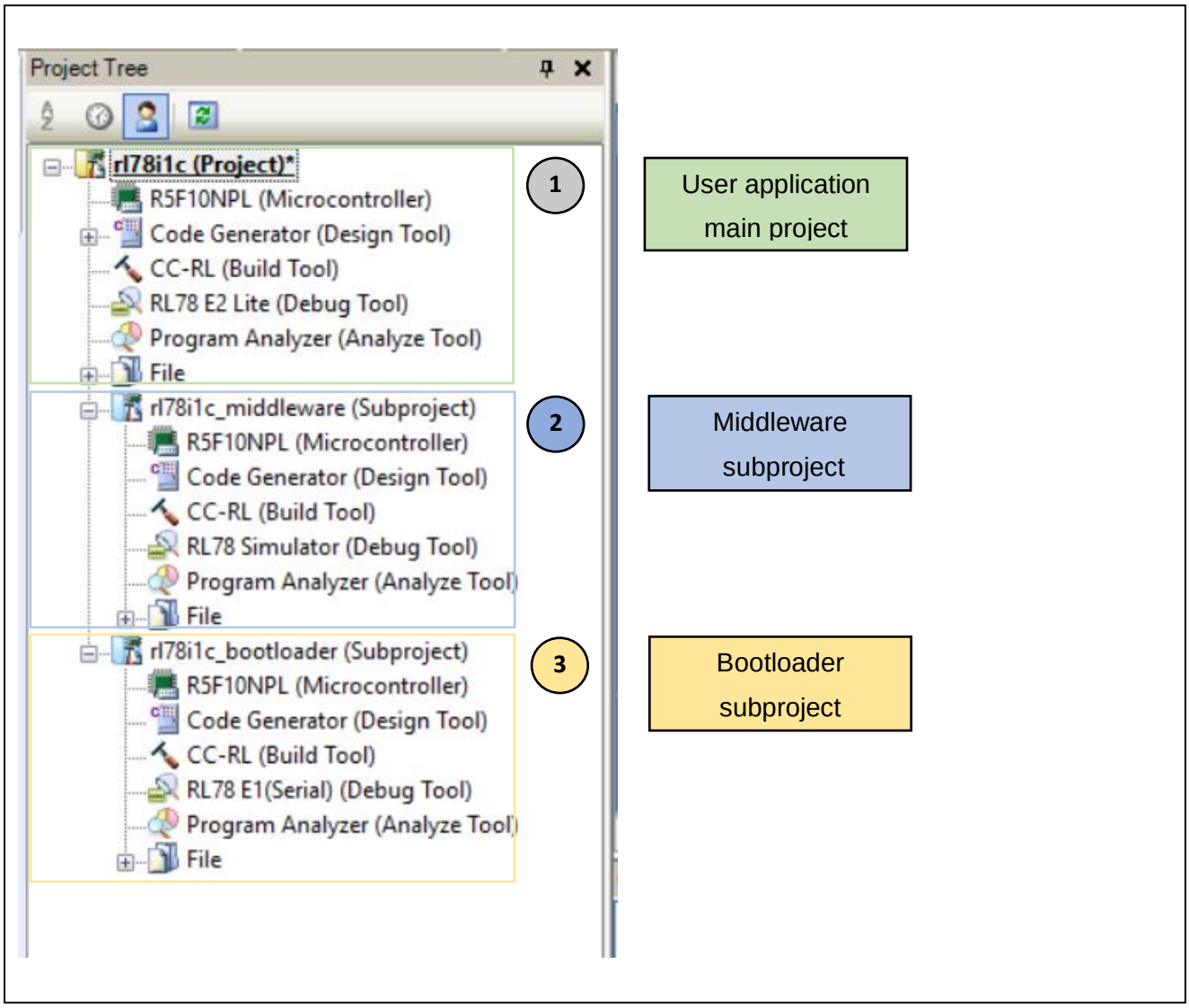


**Figure 4-1 : Project Configuration**

## 4.2. Memory Allocation

The ROM and RAM mappings are described below.

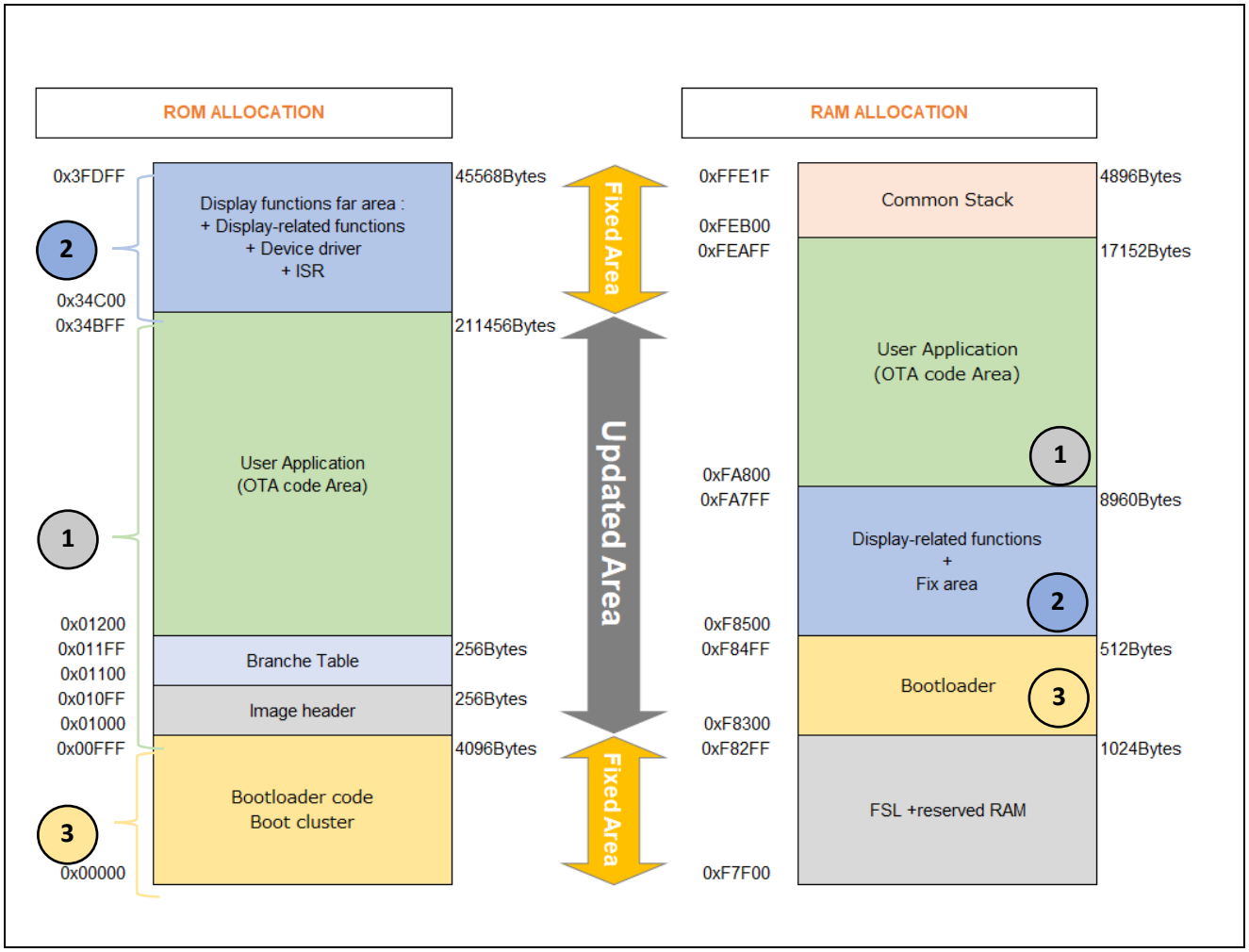


**Figure 4-2 : Memory Mapping**

The numbers (1), (2), and (3) above indicate the correspondence with each project in Figure 4-1.

The rewriting target is the entire “Updated Area” in the figure. The bootloader and display functions are located in the “Fixed Area” and cannot be rewritten by FOTA.

The startup bank will be placed at 0x00000 to 0x3FDFF, and similar ones will be placed in the other bank after 0x40000.

### 4.2.1. Memory Allocation for User Application Project

This section describes how to allocate ROM and RAM for the rl78i1c project (Figure 4-1 (1)).

The allocation method is shown below for CS+ and e2studio respectively.

- 00000-0007f : ROM area for on-chip debugger functions.
- 000c0-000d7 : ROM area for the Option Byte and Security ID required for MCU operation.
- 01000-34bff : ROM area for user applications.
- 3fe00-3ffff : ROM area for debugger monitor2.
- fa800-feaff : RAM area for user applications.

■ Address range of memory type.

**For CS+:**

CC-RL (Build Tool) → Link Options → Verify → Address range of memory type

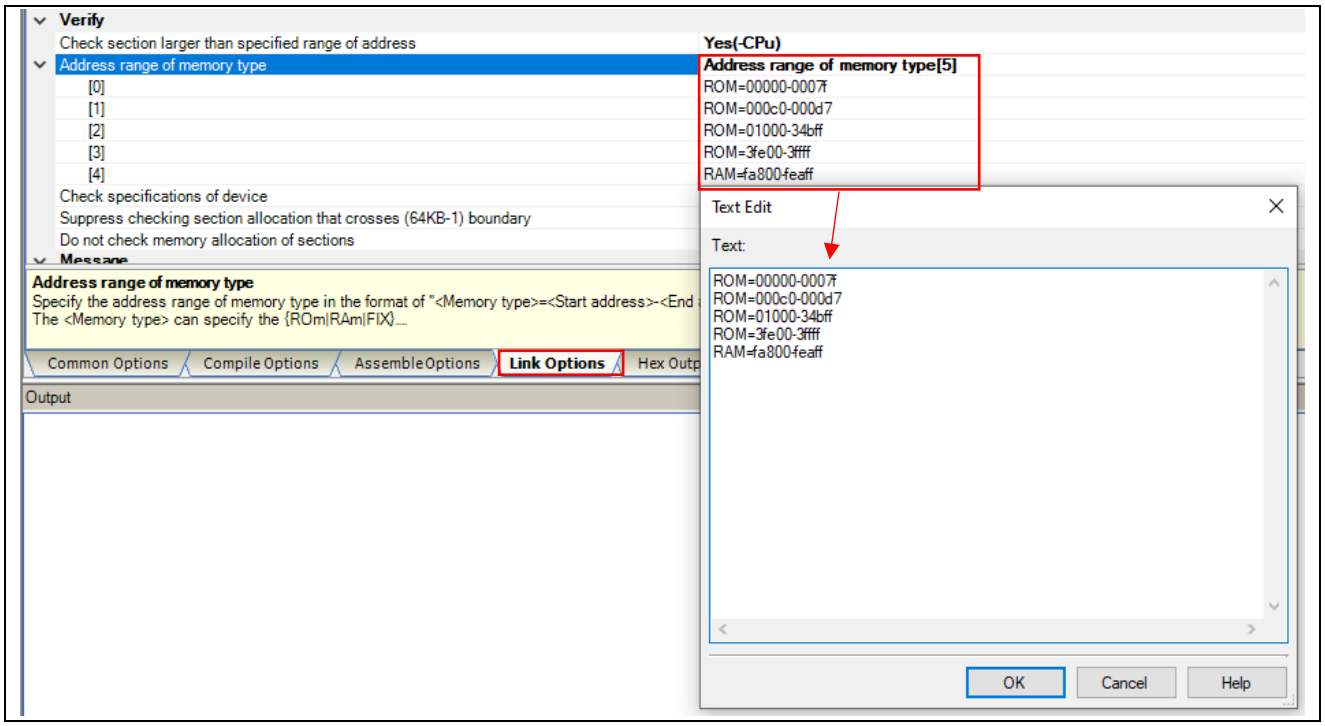


**Figure 4-3 : User Application - Address Range of Memory Type (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Device-→ Address range of memory type

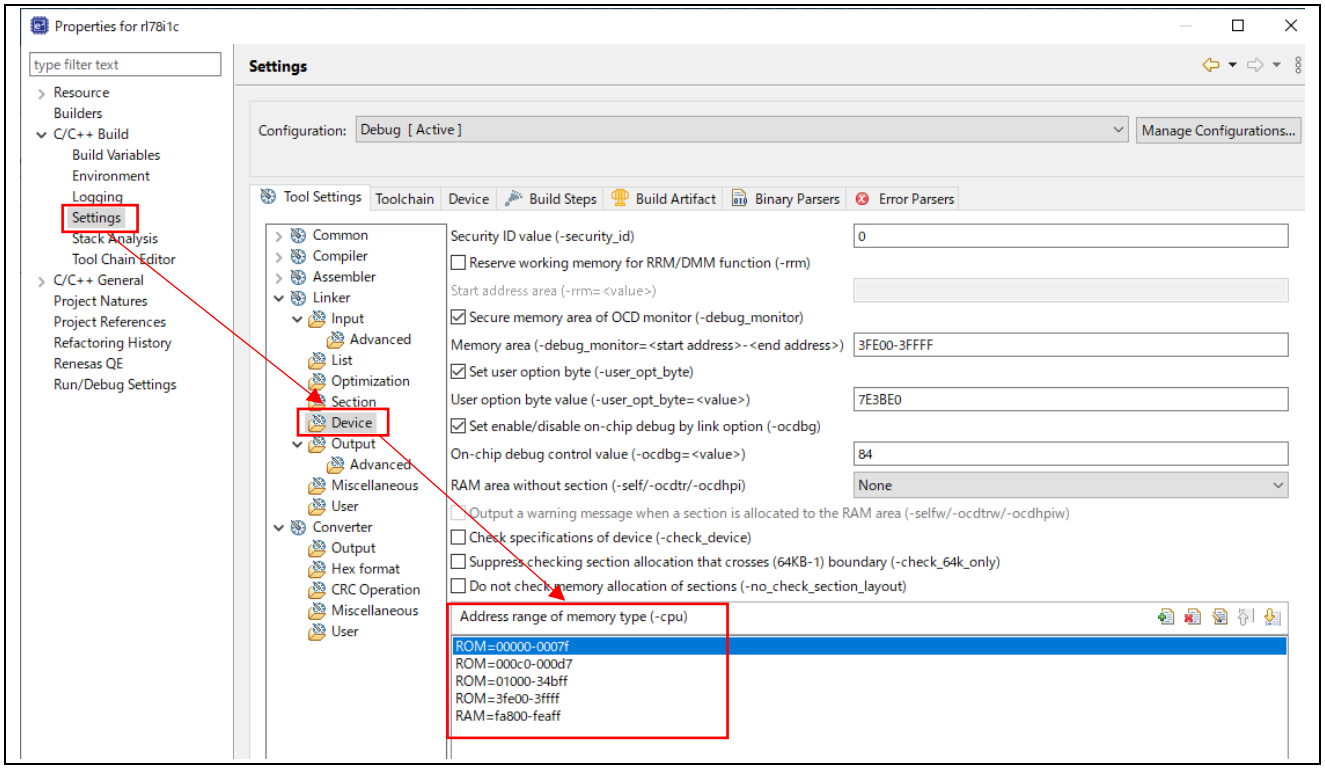


**Figure 4-4 : User Application - Address Range of Memory Type (e²studio)**

■ Debug monitor area.

**For CS+:**

CC-RL (Build Tool) → Link Options → Device → Set debug monitor area

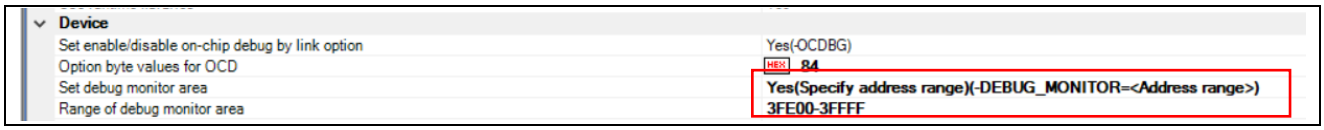


**Figure 4-5 : User Application - Debug Monitor Area (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Device → Memory Area

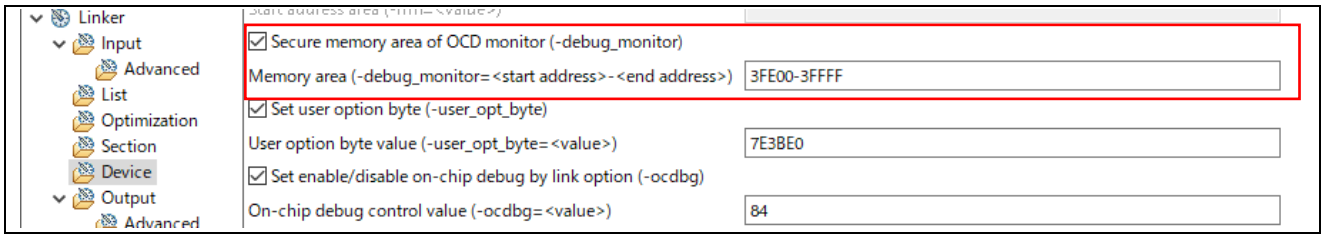


**Figure 4-6 : User Application - Debug Monitor Area (e²studio)**

■ Section layout.

**For CS+:**

CC-RL (Build Tool) → Link Options → Section → Section start address

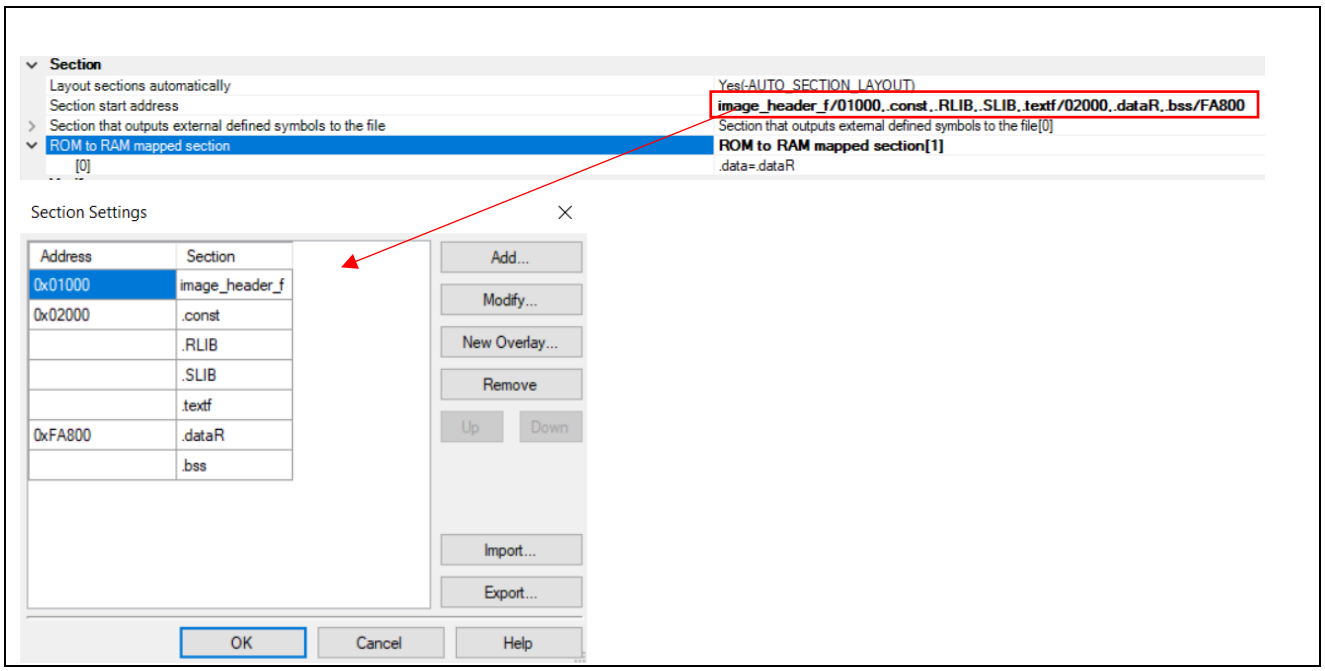


**Figure 4-7 : User Application - Section Layout (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker →Section → Sections

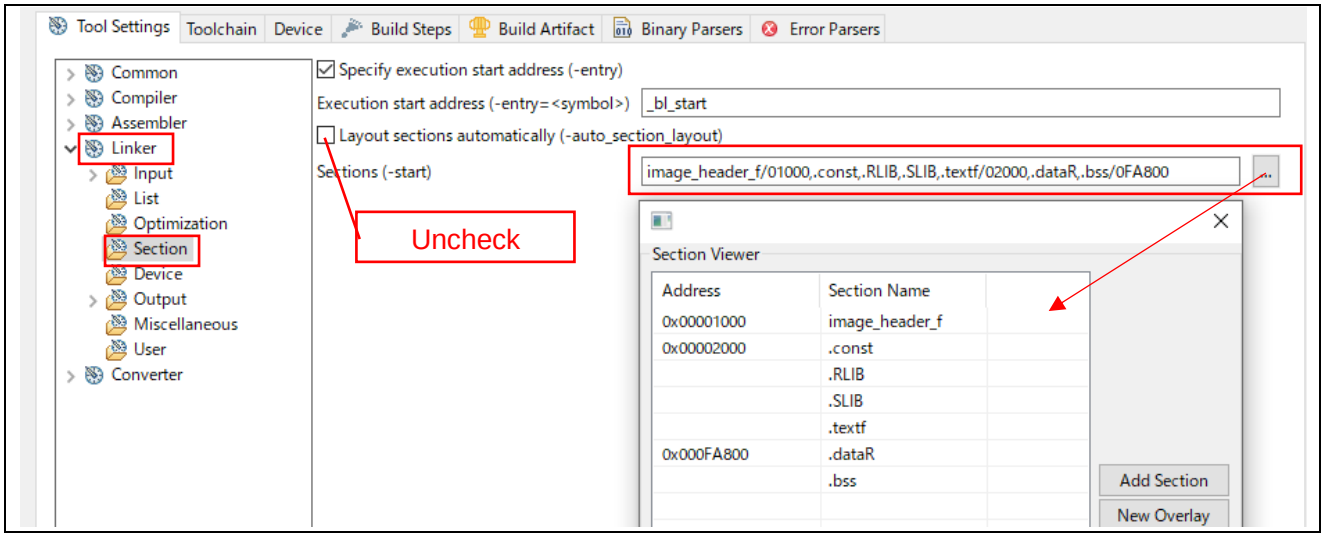


**Figure 4-8 : User Application - Section Layout (e²studio)**

### 4.2.2. Memory Allocation for Middleware Subproject

This section describes how to allocate ROM and RAM for the rl78i1c_middleware subproject (Figure 4-1 (2)). The allocation method is shown below for CS+ and e2studio respectively.

■ Address range of memory type.

**For CS+:**

CC-RL (Build Tool) → Link Options → Section → Device → Address range of memory type

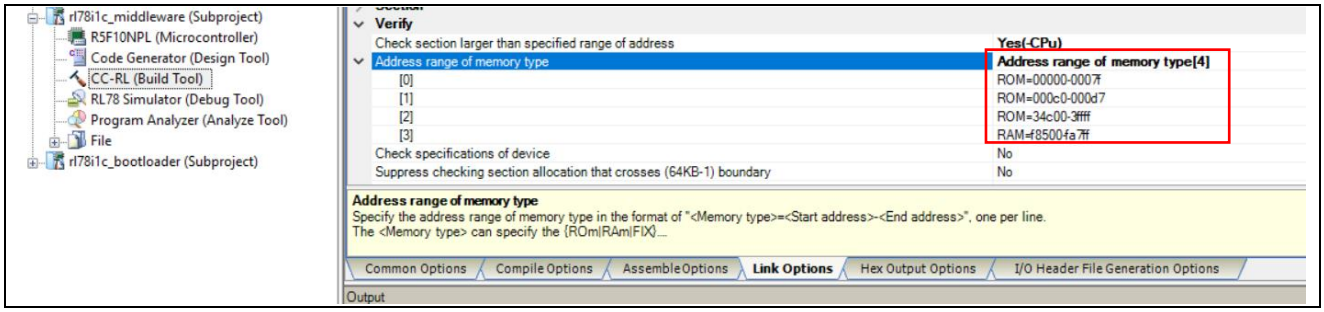


**Figure 4-9 : Middleware - Address Range of Memory Type (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Device → Address range of memory type

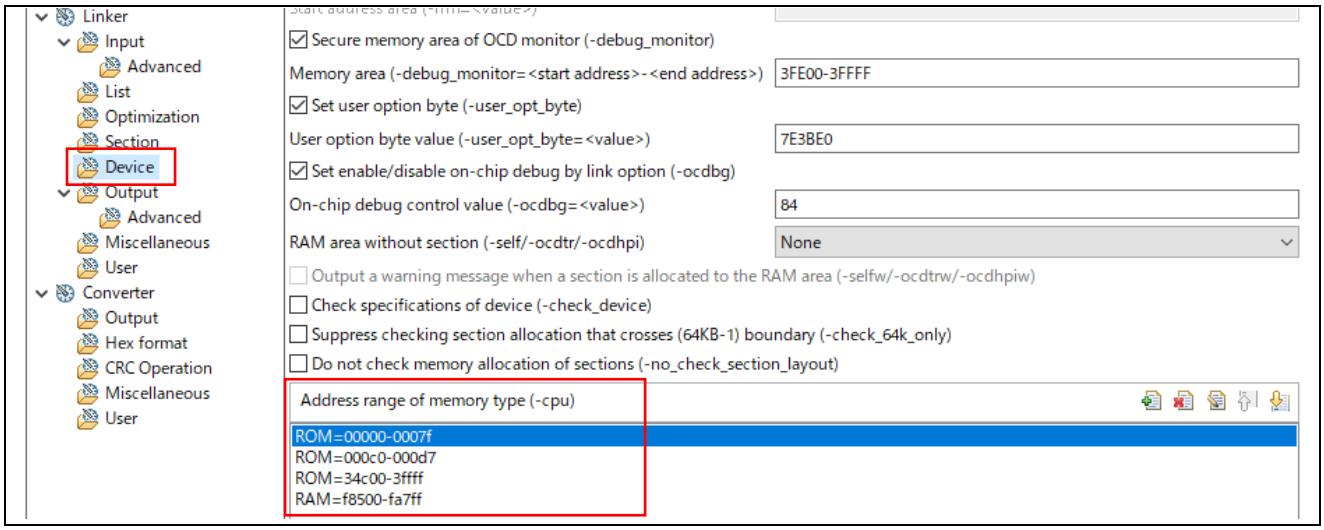


**Figure 4-10 : Middleware - Address Range of Memory Type (e²studio)**

■ Section layout and External defined symbols.

**For CS+:**

CC-RL (Build Tool) → Link Options → Section → Section start address

CC-RL (Build Tool) → Link Options → Section → Section that outputs external defined symbols to the file

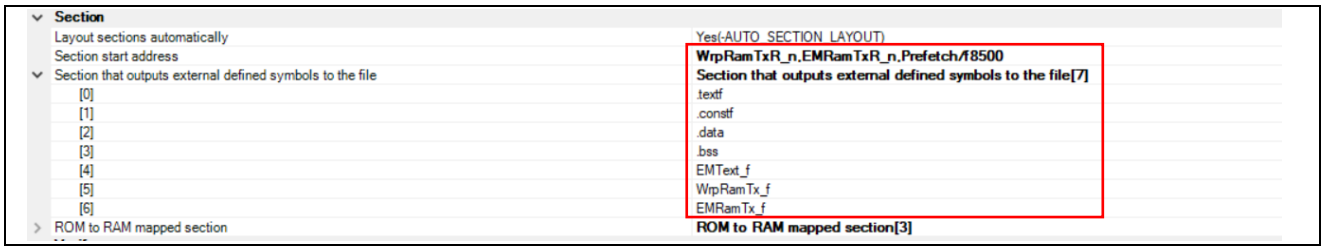


**Figure 4-11 : Middleware - Section Layout and External Defined Symbols (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Section → Sections

Properties → C/C++ Build → Settings → Linker → Section → Section that outputs external symbols to the file

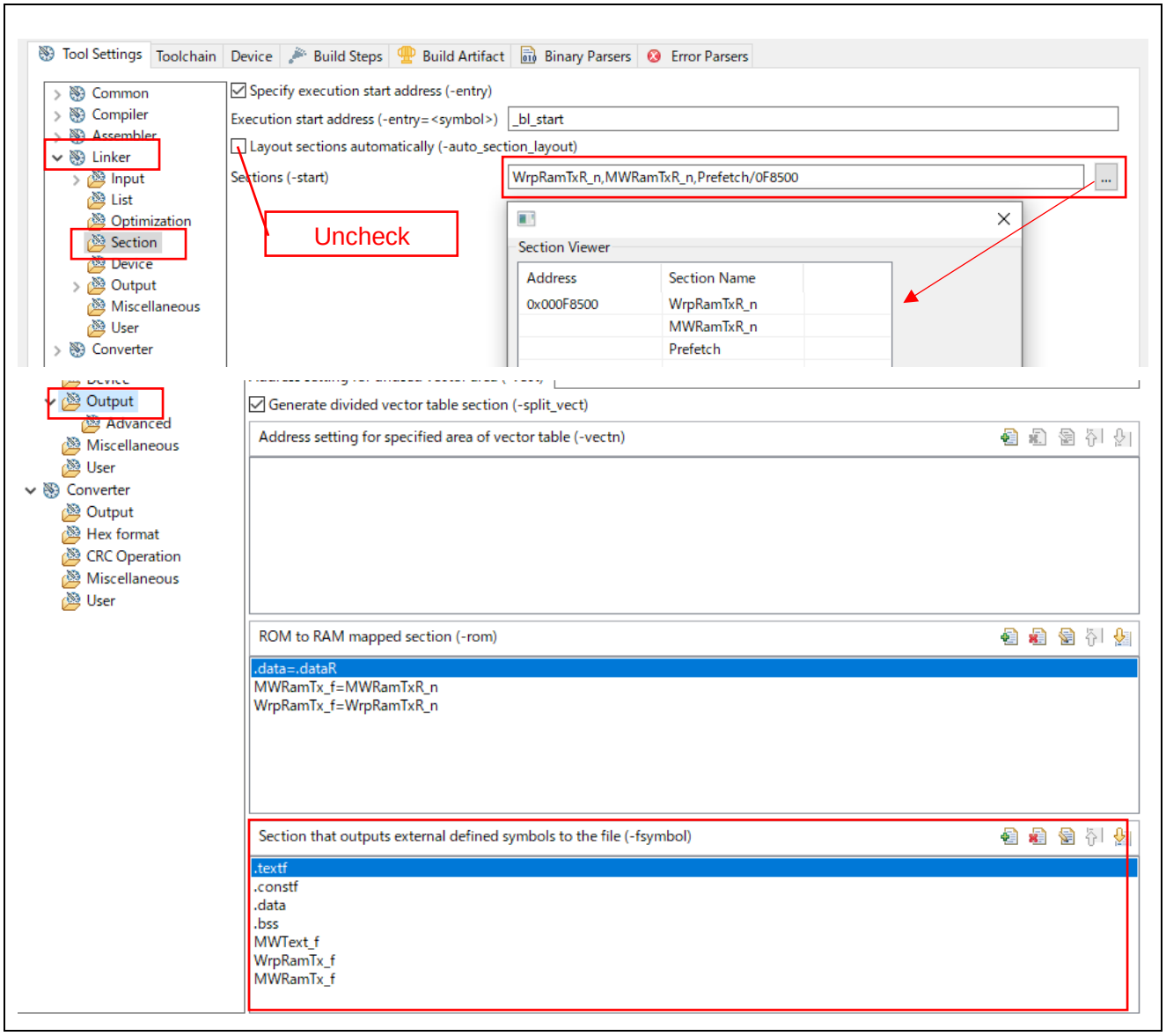


**Figure 4-12 : Middleware - Section Layout and External Defined Symbols (e²studio)**

### 4.2.3. Memory Allocation for Bootloader Subproject

This section describes how to allocate ROM and RAM for the rl78i1c_bootloader subproject (Figure 4-1 (3)). The allocation method is shown below for CS+ and e2studio respectively.

■ Address range of memory type.

**For CS+:**

CC-RL (Build Tool) → Link Options → Device → Verify → Address range of memory type

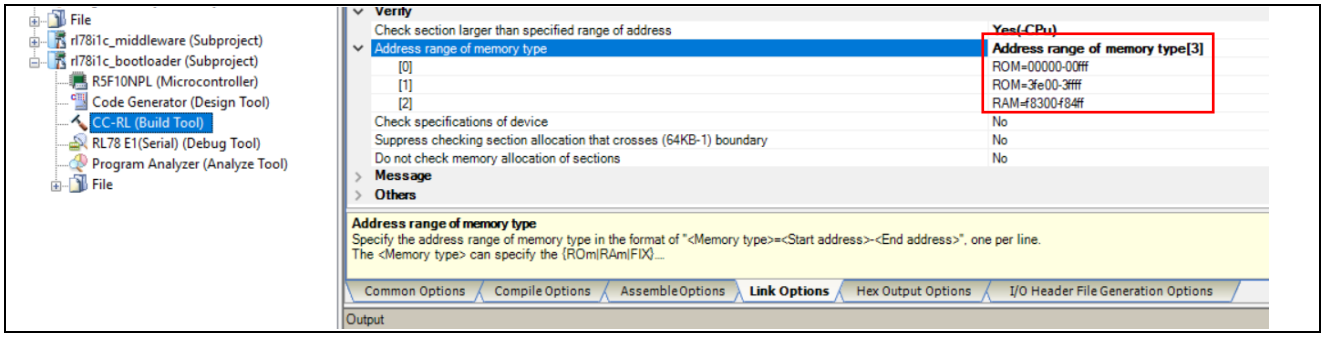


**Figure 4-13 : Bootloader - Address Range of Memory Type (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Device → Address range of memory type

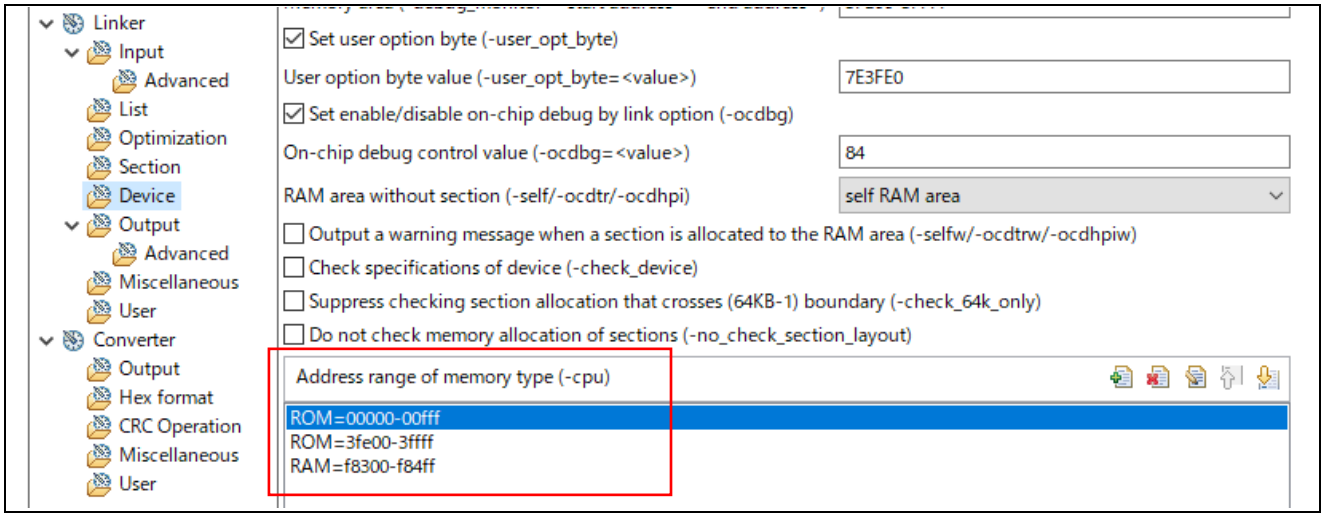


**Figure 4-14 : Bootloader - Address Range of Memory Type (e²studio)**

■ Section layout and External defined symbols.

**For CS+:**

CC-RL (Build Tool) → Link Options → Section → Section start address

CC-RL (Build Tool) → Link Options → Section → Section that outputs external defined symbols to the file

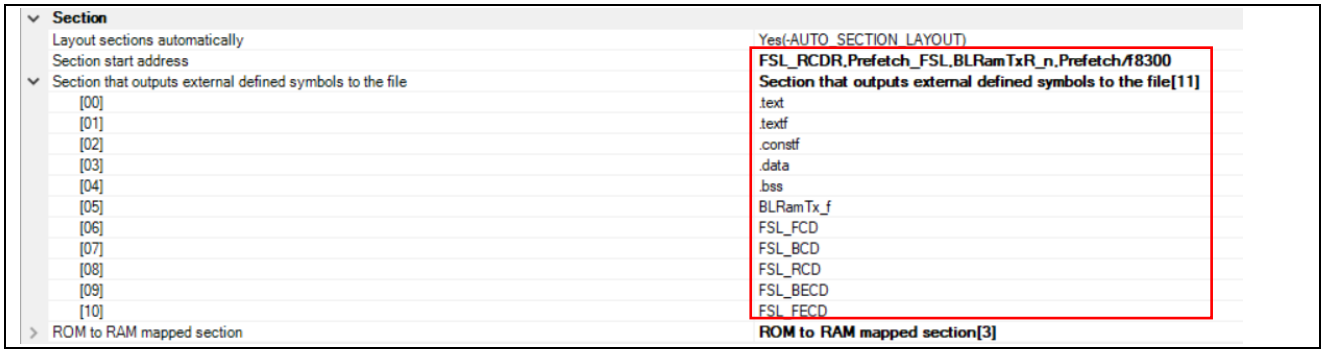


**Figure 4-15 : Bootloader - Section Layout and External Defined Symbols (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Section → Sections

Properties → C/C++ Build → Settings → Linker → Section → Section that outputs external symbols to the file

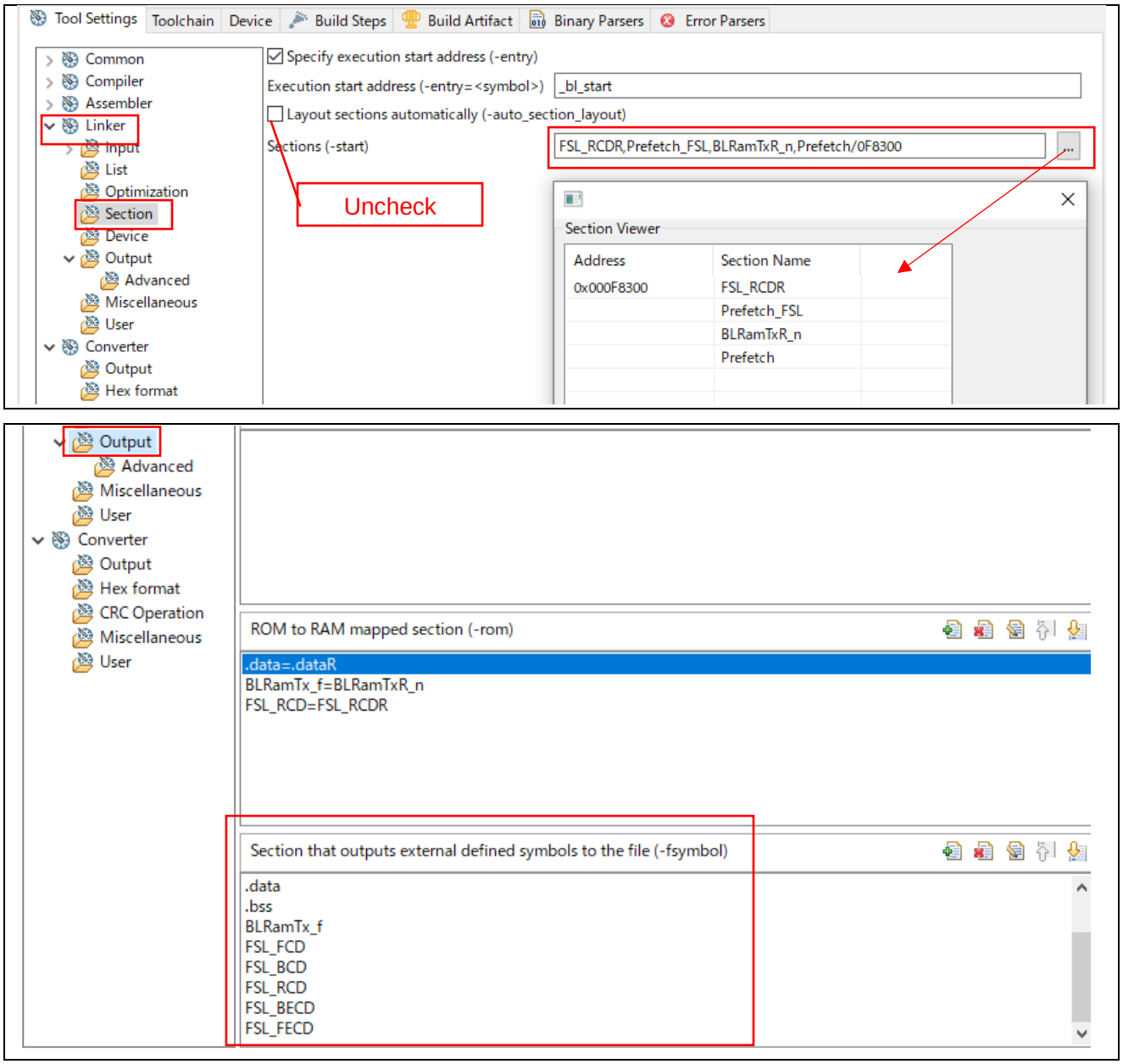


**Figure 4-16 : Bootloader - Section Layout and External Defined Symbols (e²studio)**

## 4.3. Supplemental Information on Link Options (e²studio)

In e2studio, if the checkbox [Layout sections automatically (-auto_section_layout)] in [C/C ++ Build→ Settings→ Linker→ Sections] is checked, the linker option "-start" does not appear. Therefore specify "-auto_section_layout" in the Use-defined options (Figure 4-17).

This is required for all rl78i1c project (Figure 4-1 (1)), rl78i1c_middleware subproject (Figure 4-1 (2)), and rl78i1c_bootloader subproject (Figure 4-1 (3)).

**For e²studio only:**

Properties → C/C++ Build → Settings → Linker → User → User-defined options

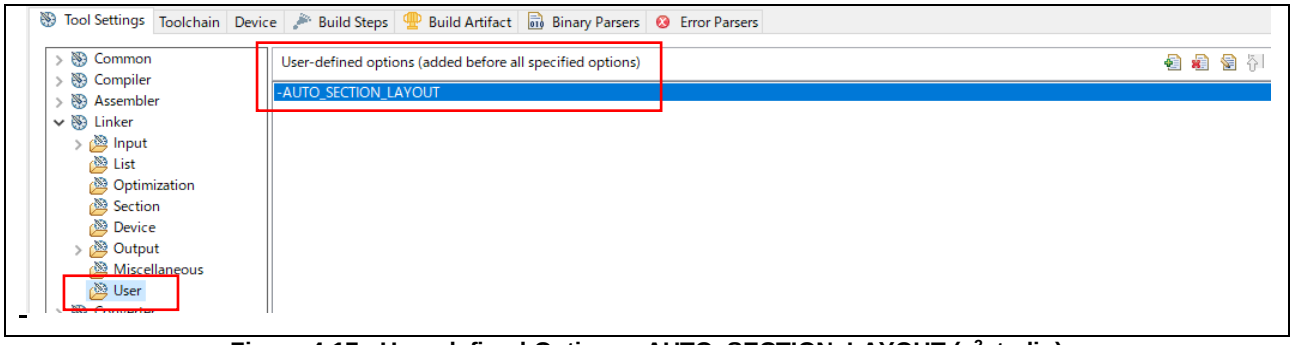


**Figure 4-17 : User-defined Options - AUTO_SECTION_LAYOUT (e²studio)**

## 4.4. Using External Defined Symbol Files

The external defined symbol files are used for invocations and information sharing between main and subprojects. In order to use them, it is necessary to include the external defined symbol file in the project after outputting the symbol file in Chapter 4.2.2 and Chapter 4.2.3.

In this example project, the external symbol files are referenced as follows.

- Refer to the middleware subproject and bootloader subproject in the main project.
- Refer to the bootloader project in the middleware subproject

**For CS+:**

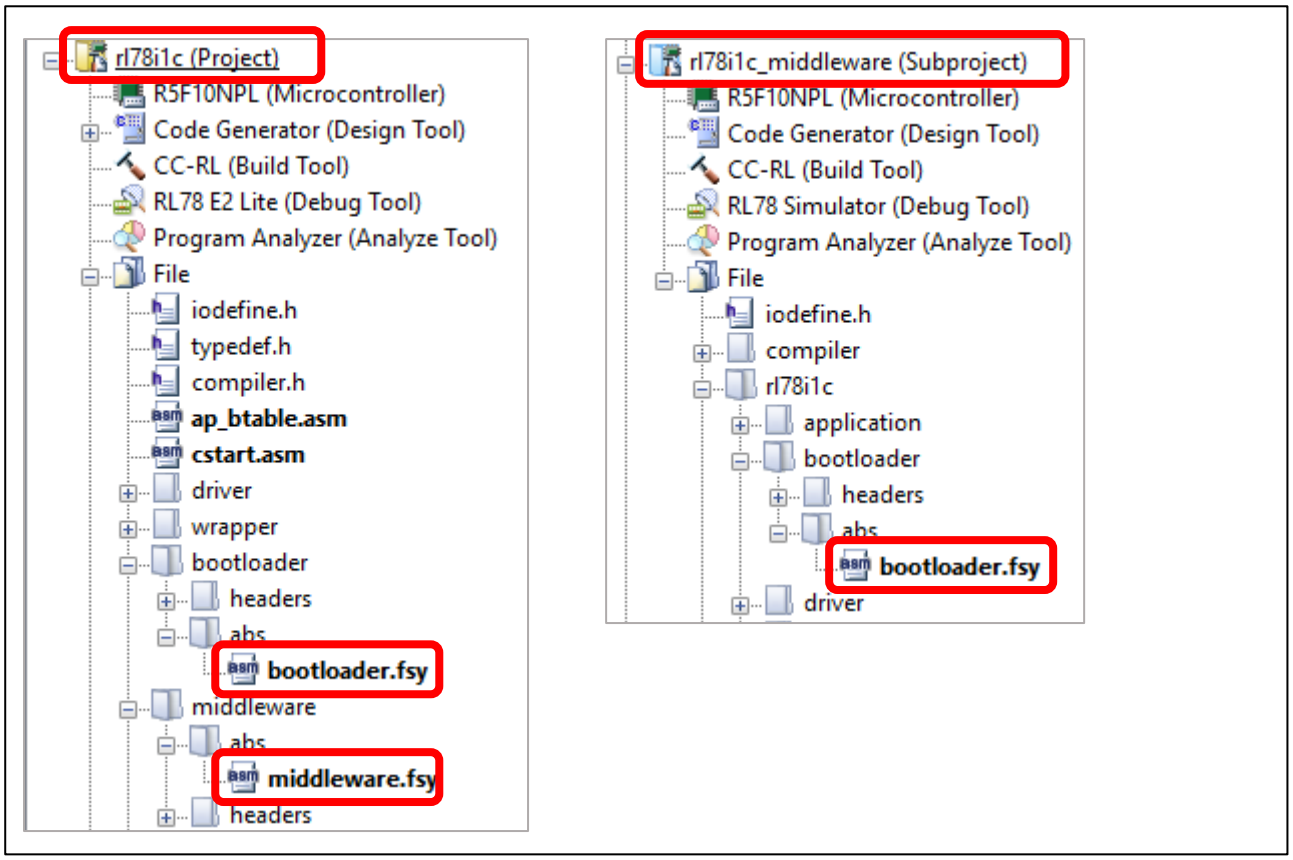


**Figure 4-18 : Include External Definition Symbol File**

**For e²studio:**

The case of e2studio is omitted because it is the same as the case of CS+.

## 4.5. ROM to ROM Mapping Settings

### 4.5.1. LCD Update Process Routine

The screen display functions are invoked by a timer interrupt.

The interrupt process is used to update the indicator bar on the screen and display the inputted characters, etc.

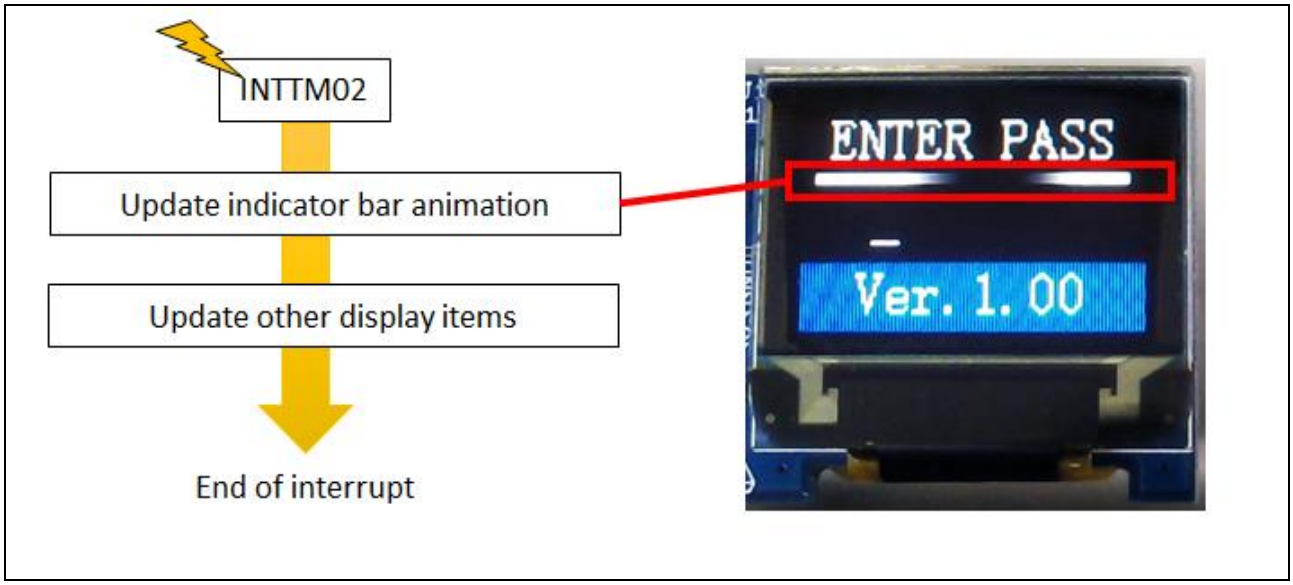


**Figure 4-19 : LCD Update Process**

It runs on the ROM in the normal operation. On the other hand, it runs on the RAM during the bank swap period to continue to execute the LCD display function.


**Figure 4-20 : Display Update**

Figure 4-21 shows an image of a code copy from ROM to RAM.

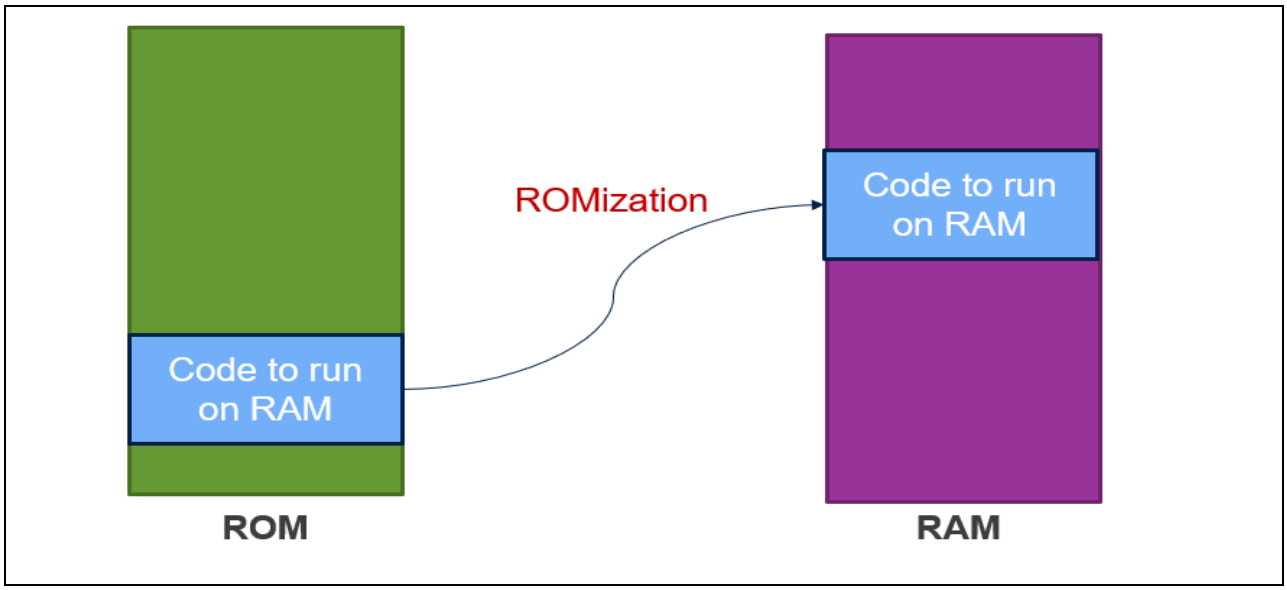


**Figure 4-21 : Code Copy from ROM to RAM**

The interrupt callback function while running on ROM is replaced by the MW_RunOnRam_RamIsr function, which controls all interrupts while running on RAM. This function checks each interrupt flag (specifically, TMIF02 and CSIIF30) and the corresponding interrupt process is executed. After processing, each interrupt flag must be cleared manually.

function performs the equivalent of each interrupt. We must also manually clear each interrupt flag after processing.

Copying the code from ROM to RAM is done on a section-by-section basis. Therefore, divide the sections in advance and isolate the code to be executed on RAM.

The MW_RunOnRam_PrepareFunctions function is used to copy the code from ROM to RAM.

### 4.5.2. Memory Mapping of User Application Project

For the rl78i1c project (Figure 4-1 (1)), set the area to be mapped from ROM to RAM as follows.

**For CS+:**

CC-RL (Build Tool) → Link Options → Section → ROM to RAM mapped section

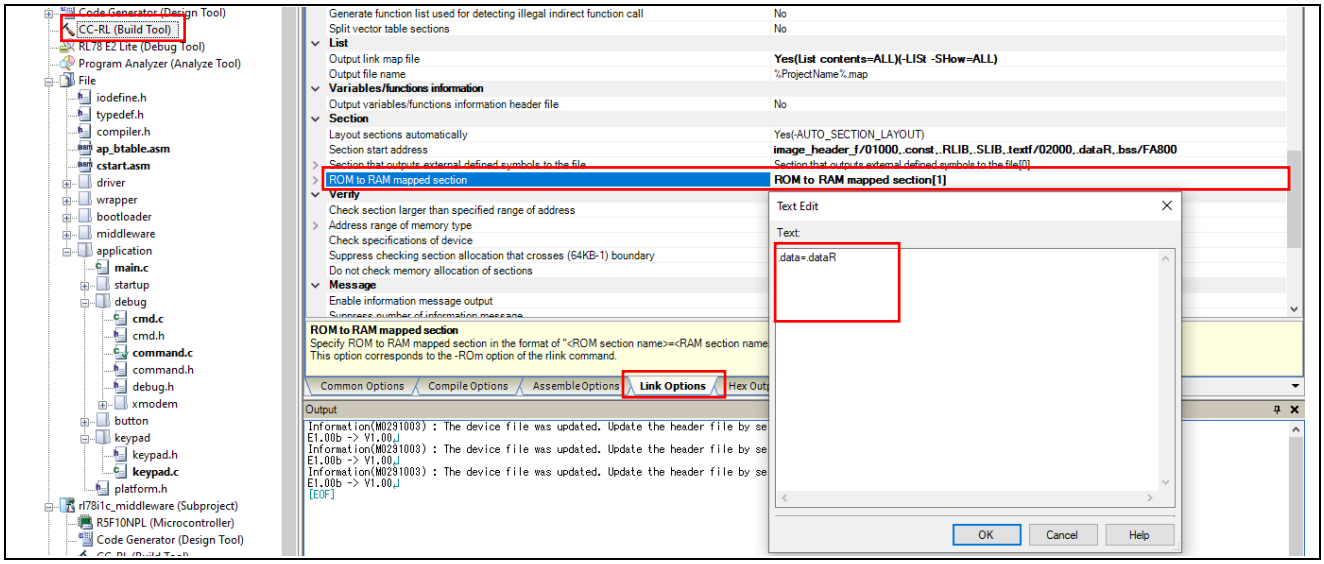


**Figure 4-22 : User Application - ROM to RAM Mapped Section (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Output → ROM to RAM mapped section

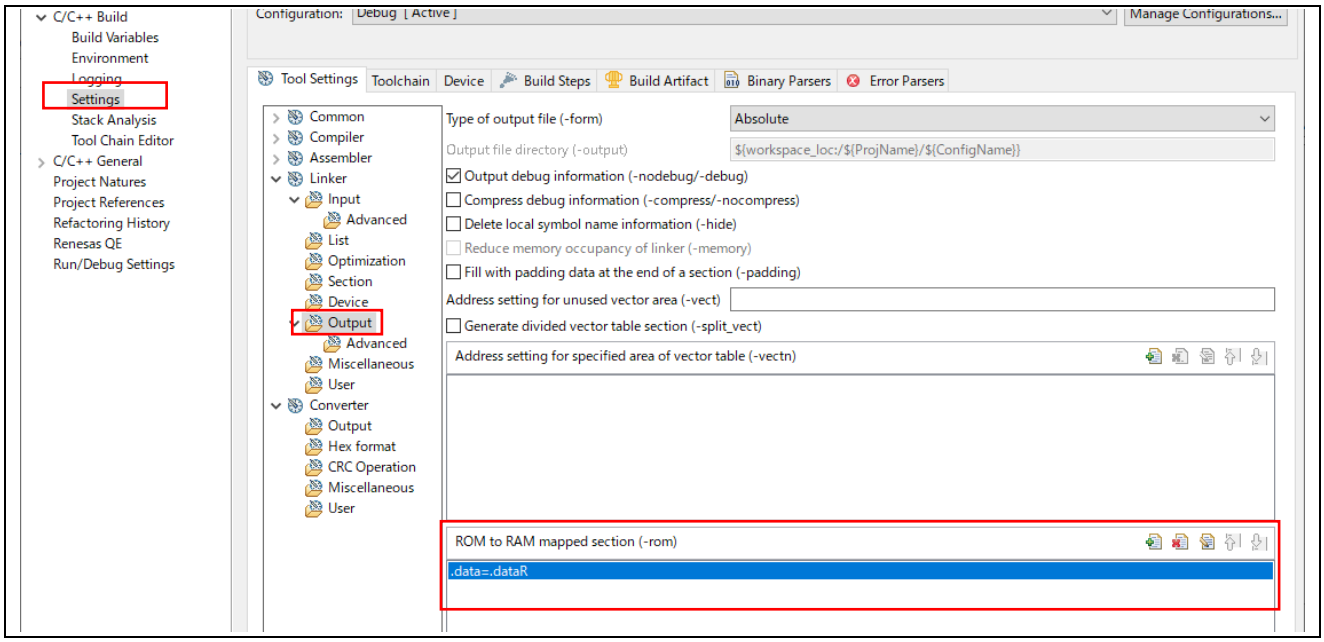


**Figure 4-23 : User Application - ROM to RAM Mapped Section (e²studio)**

### 4.5.3. Memory Mapping of Middleware Subproject

For the rl78i1c_middleware subproject (Figure 4-1 (2)), set the area to be mapped from ROM to RAM as follows.

**For CS+:**

CC-RL (Build Tool) → Link Options → Section → ROM to RAM mapped section

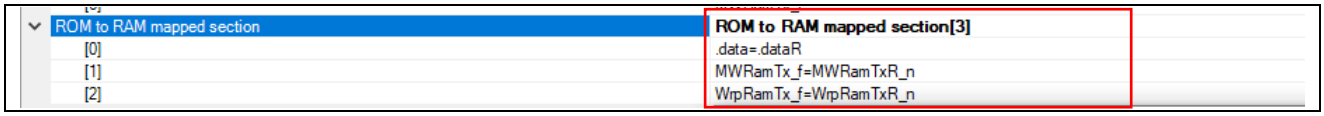


**Figure 4-24 : Middleware - ROM to RAM Mapped Section (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Output → ROM to RAM mapped section

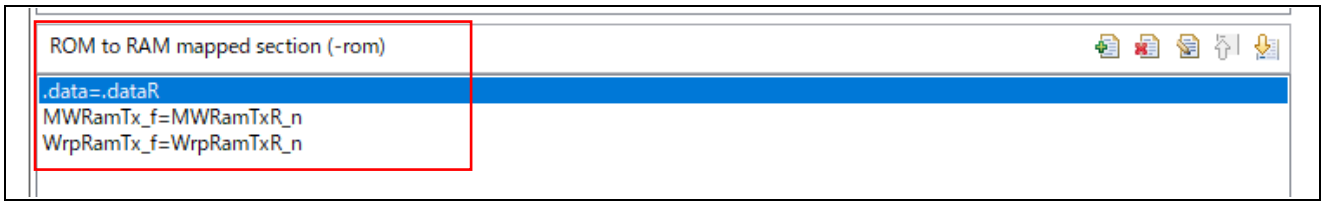


**Figure 4-25 : Middleware - ROM to RAM Mapped Section (e2studio)**

MWRamTx : This section contains the functions for updating the LCD display on RAM.

WrpRamTx : his is a utility function section used by functions in MWRamTx.

### 4.5.4. Memory Mapping of Bootloader Subproject

For the rl78i1c_bootloader subproject (Figure 4-1 (3)), set the area to be mapped from ROM to RAM as follows.

**For CS+:**

CC-RL (Build Tool) → Link Options → Section → ROM to RAM mapped section

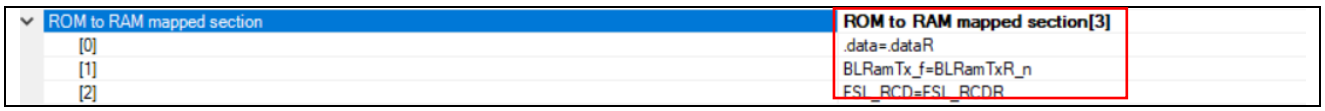


**Figure 4-26 : Bootloader - ROM to RAM Mapped Section (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Output → ROM to RAM mapped section

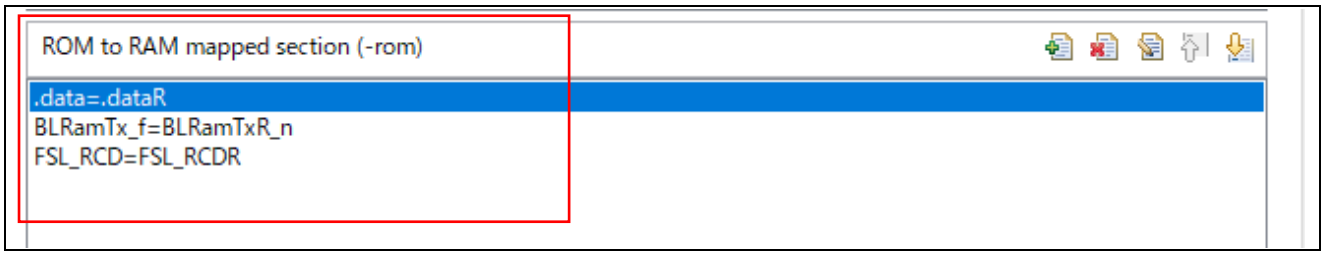


**Figure 4-27 : Bootloader - ROM to RAM Mapped Section (e²studio)**

BLRamTx : This is the section of functions that the bootloader uses on RAM.

FSL_RCD : FSL library section,

## 4.6. Branch Table Flow

The user application has a branch table for interrupt processing that is separate from the bootloader vector table. The branch table method would be faster than the function pointer call method.

The user application and the boot loader can share the vector table. On the other hand, the vector table is fixed and cannot be changed according to the update of the user application.

In addition, the code related to the middleware is placed in the far area. Therefore, the branch table is divided into two parts, **ap_btable.asm** for user application projects and **mw_btable.asm** for middleware subprojects. Branches to each starting from the vector table of the boot loader subproject. Therefore, in the bootloader subproject, the vector table section needs to be split.

**For CS+:**

CC-RL (Build Tool) → Link Options → Output Code → Split vector table sections

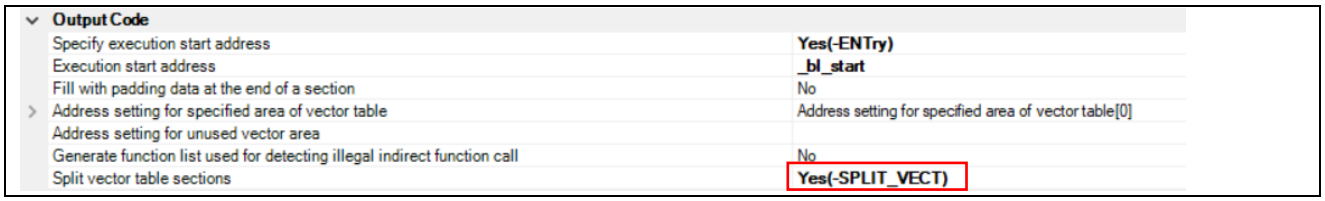


**Figure 4-28 : Split Vector Table Sections (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Linker → Output → Generate divided vector table section

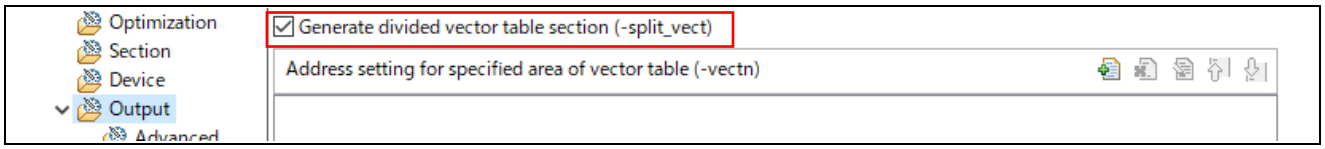


**Figure 4-29 : Split Vector Table Sections (e2studio)**

The following figure shows the branch table flow for the display functions.

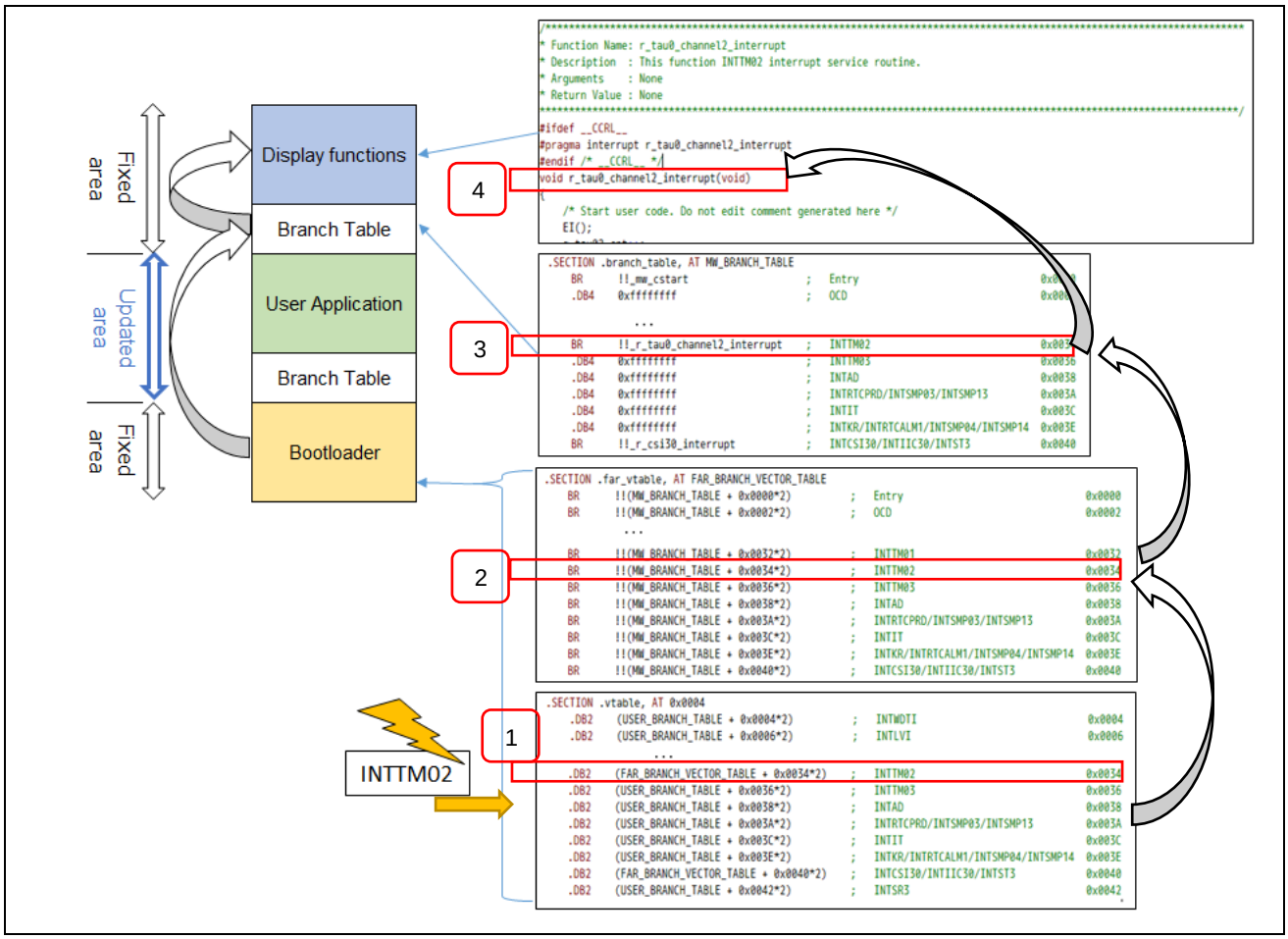


**Figure 4-30 : Branch Table Flow**

The display process starts from the timer interrupt using the INTTM02. When the interrupt INTTM02 occurs, it jumps to the specified location. (Step 1)

```
.DB2    (FAR_BRANCH_VECTOR_TABLE + 0x0034*2)    ;   INTTM02
```

The display functions are located in the far area, so it jumps to an intermediate branch table. (Step 2)

Then it jumps to the branch table of the display functions (Step 3) and reaches the interrupt function itself. (Step 4)

Note: The related source files are as follows.

1. **bl_vtable.asm** in rl78i1c_bootloader subproject
2. **bl_far_vtable.asm** in rl78i1c_bootloader subproject
3. **mw_btable.asm** in rl78i1c_middleware subproject
4. **r_tau_user_mw.c** in rl78i1c_middleware subproject

## 4.7. Continuous Operation FOTA Example Project API Functions

### 4.7.1. API functions

**Table 4-1 : API Functions**

| Function | Explanation |
|---|---|
| COMMAND_PollingProcessing | Processes received UART commands |
| COMMAND_InvokeBankSwap | Command to invoke Continuous FOTA Bank Swap |
| MW_RunOnRam_NonStopBankSwap | Continuous update sequence |
| MW_RunOnRam_PrepareFunctions | Prepare to run on RAM (Copy code from ROM to RAM) |
| MW_RunOnRam_DisableInterruptsExceptDisplayRelated | Mask off all other interrupt except display related |
| BL_RunOnRam_PrepareFunctions | Prepare to run on RAM (Copy code from ROM to RAM) |
| BL_FLASH_RAM_SwapBankWithRamIsr | Swap active boot cluster with running interrupt service routine on RAM |
| FSL_ChangeInterruptTable | Change vector table to RAM ISR |
| FSL_SwapActiveBootCluster | Swap the bank |
| FSL_RestoreInterruptTable | Restore vector table to ROM ISR |
| BL_FLASH_RAM_JumpBankSwapEntry | Call bankswap entry function |

### 4.7.2. Continuous Operation FOTA Sequence

Figure 4-31 shows an example of the API usage for Continuous Operation FOTA.

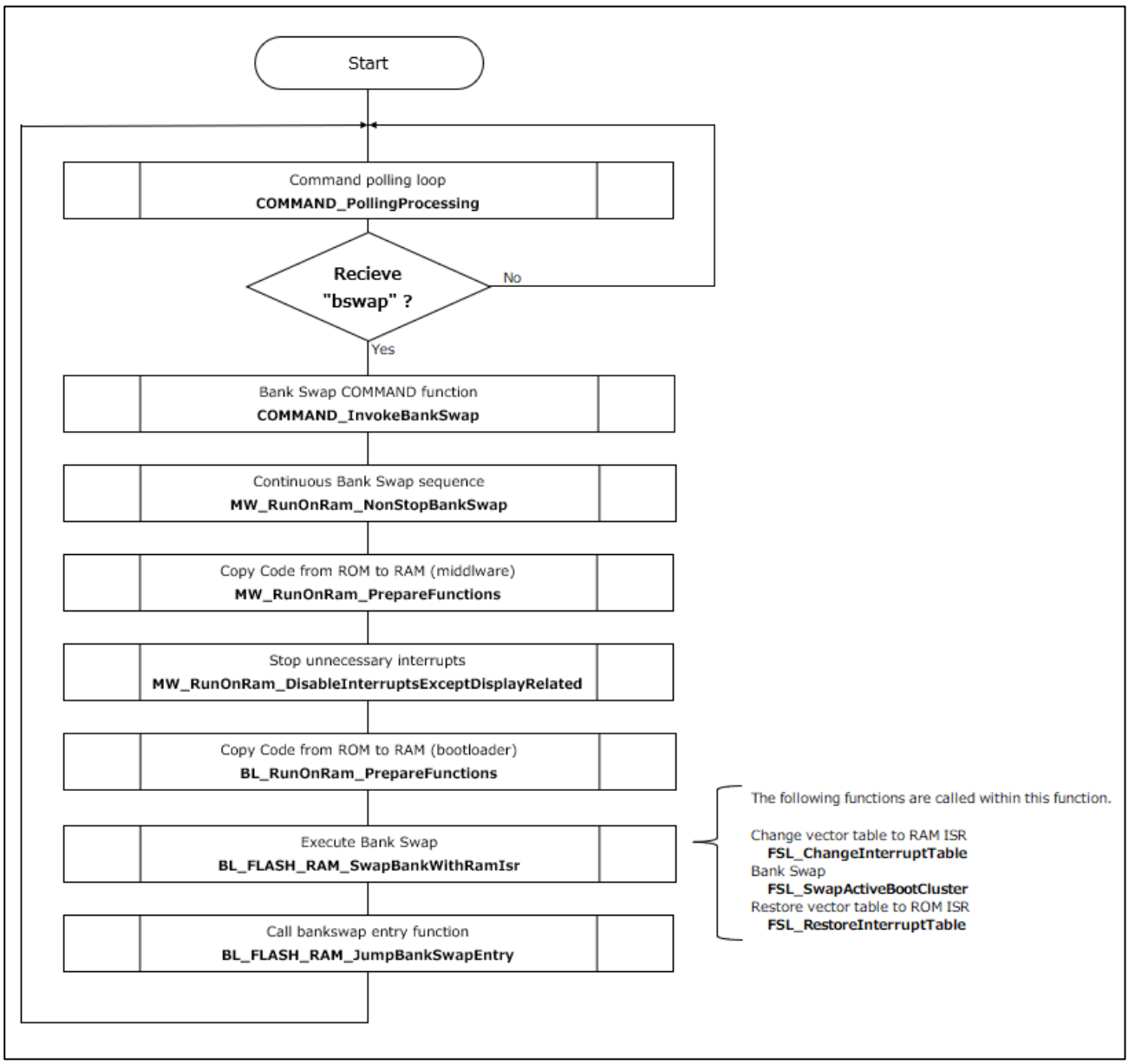


**Figure 4-31 : Example of API Function Usage**

## 4.8. Build

This section describes how to create [rl78i1c_production.mot] in Chapter 3.1 (1) and [rl78i1c_v100.mot] and [rl78i1c_v200.mot] in Chapter 3.1 (2). Set the passcode as described in Chapter 2.1 and the version information as described in Chapter 2.2 as follows.

■ Passcode setting.

The 4-digit passcode is defined in [platform.h]. APP_PASSCODE_1~4 corresponds to the 1st~4th digits of the passcode, and any character from "0123456789ABCDF" can be set. ("E" cannot be set because it is used as a decision key.)

```
23  /*********************************************************************************
24  Macro definitions
25  *********************************************************************************/
26  /* Software version to be printed on start-up of FOTA Demo */
27  #define APP_SOFTWARE_VERSION                (1)
28  /* Passcode */
29  #define APP_PASSCODE_1                      ('1')
30  #define APP_PASSCODE_2                      ('2')
31  #define APP_PASSCODE_3                      ('3')
32  #define APP_PASSCODE_4                      ('A')
```

**Figure 4-32 : Passcode and Version Settings [platform.h]**

■ Version setting.

The version information is specified in [platform.h] and the build configuration file. (Values from 0 to 9 can be set.)

The specified value is reflected in the first digit of the version.

・APP_SOFTWARE_VERSION in [platform.h].

Define a value between 0 and as shown in Figure 4-32.

・Build configuration.

Set the values 0x000000~0x000009 (the same values as APP_SOFTWARE_VERSION above) in the locations shown in Figure 4-33 and Figure 4-34.

**For CS+:**

CC-RL (Build Tool) → Link Options → Others → Command executed after ling processing

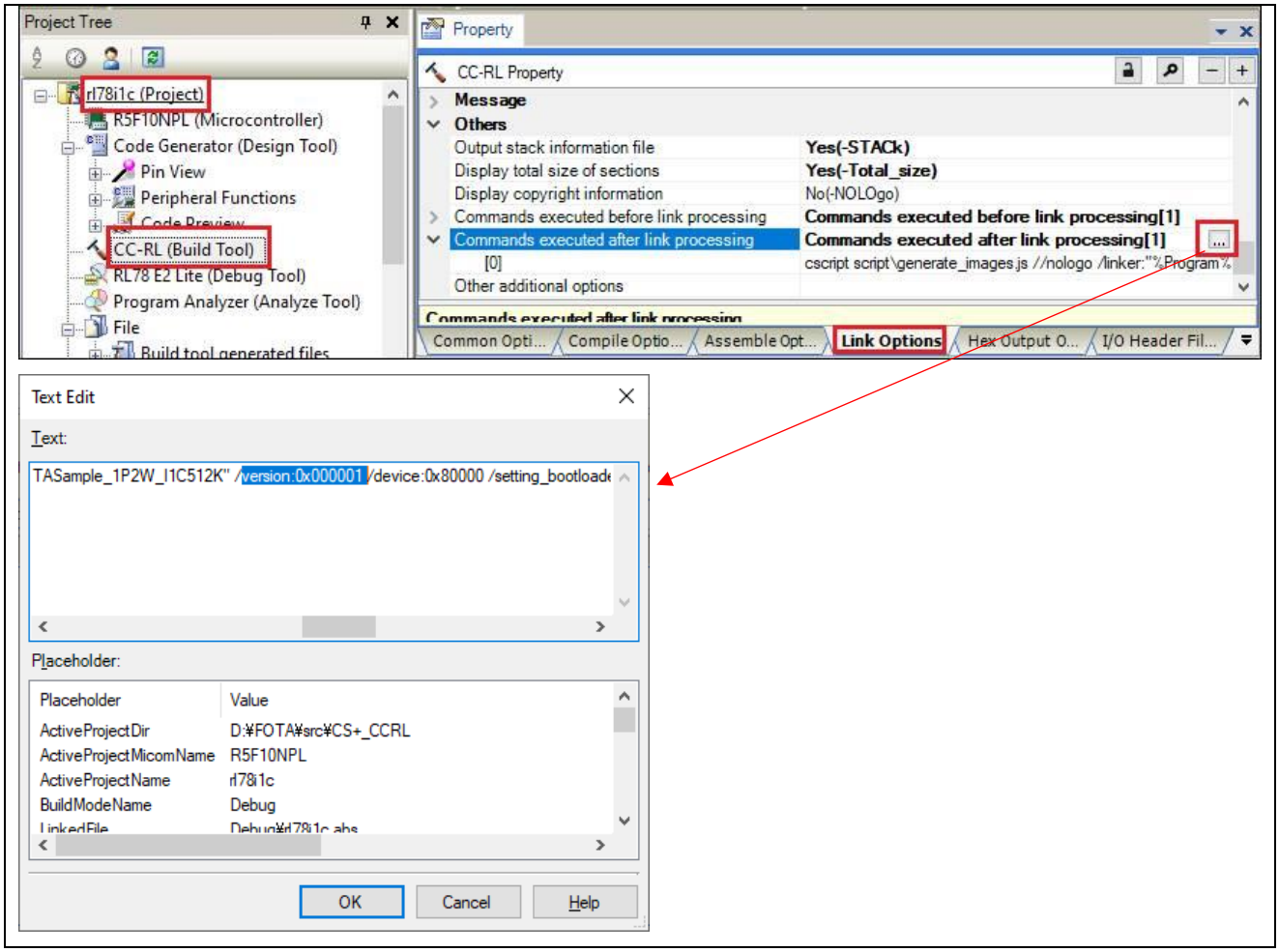


**Figure 4-33 : Version Setting (CS+)**

**For e²studio:**

Properties → C/C++ Build → Settings → Build Steps → Post-build steps

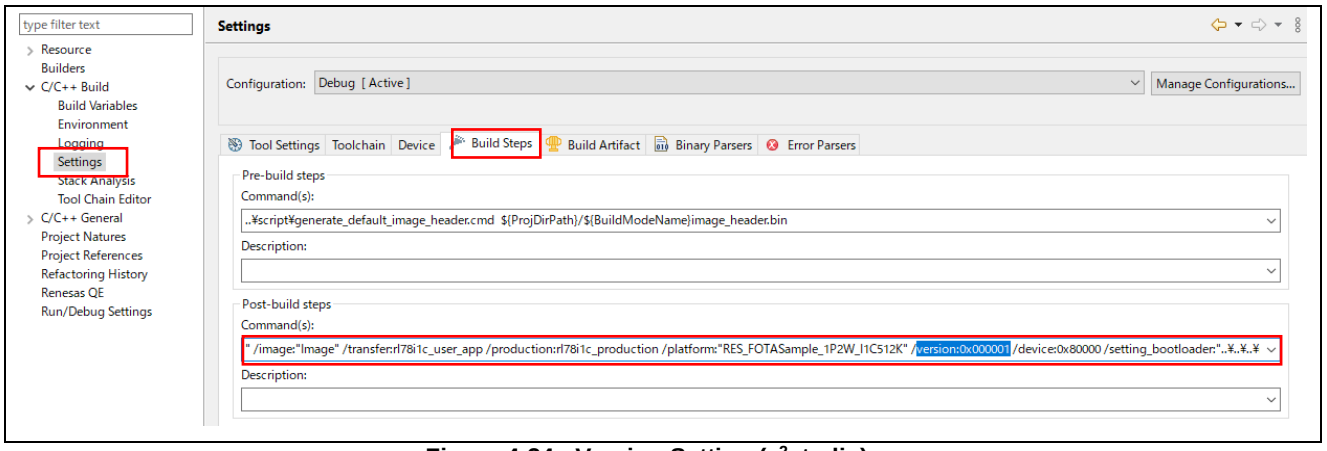


**Figure 4-34 : Version Setting (e²studio)**

■ Motorola file output.

・Firmware of Ver.1.00

Set APP_SOFTWARE_VERSION as "1" in [platform.h] and the parameter as “0x0000001” in the build configuration. (Change the passcode if necessary.)

When you build it, [**rl78i1c_production.mot**] is generated in the following folder.

CS+

\Source\CS+_CCRL\Debug\Image

e²studio

\Source\e2studio\rl78i1c\Debug\Image

[**rl78i1c_production.mot**] is the one used in Chapter 3.1 (1).

[**rl78i1c.mot**] is also generated in the same folder.

Rename [**rl78i1c0.mot**] to [**rl78i1c_v100.mot**] and use it in Chapter 3.1 (2).

・Firmware of Ver.2.00

Set APP_SOFTWARE_VERSION as "2" in [platform.h] and the parameter as “0x0000002” in the build configuration. (Change the passcode if necessary.)

When you build it, [**rl78i1c.mot**] is generated in the following folder.

CS+

\Source\CS+_CCRL\Debug\Image

e²studio

\Source\e2studio\rl78i1c\Debug\Image

After that, rename [**rl78i1c0.mot**] to [**rl78i1c0_v200.mot**] and use it in Chapter 3.1 (2).

In the same way, you can generate [**rl78i1c0_v300.mot**], [**rl78i1c0_v400.mot**], … , [**rl78i1c0_v900.mot**].

## 5. Diving Deeper

1. To learn more about the RL78/I1C (512KB) Fast Prototyping Board, refer to the RL78/I1C (512KB) User's Manual available in the User Guides & Manuals of the RL78/I1C webpage at renesas.com/br/en/products/microcontrollers-microprocessors/rl78-low-power-8-16-bit-mcus/rl78i1c-ultra-low-power-microcontrollers-high-end-smart-electricity-meter-market
2. Renesas provides several example projects that demonstrate different capabilities of the RL78/I1C (512KB) Fast Prototyping Board. These example projects can serve as a good starting point for users to develop custom applications. Example projects (source code and project files) are available in the RL78/I1C (512KB) Fast Prototyping Board Example Project Bundle.

## 6. Website and Support

Visit the following URLs to learn about the kit and the RA family of microcontrollers, download tools and documentation, and get support.

- RL78/I1C Resource renesas.com/br/en/products/microcontrollers-microprocessors/rl78-low-power-8-16-bit-mcus/rl78i1c-ultra-low-power-microcontrollers-high-end-smart-electricity-meter-market
- RL78 Product Information renesas.com/br/en/products/microcontrollers-microprocessors/rl78-low-power-8-16-bit-mcus
- RL78 Knowledge Base en-support.renesas.com/knowledgeBase#31025
- Renesas Support en-support.renesas.com/dashboard

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | February 25, 2022 | − | Initial release |
| | | | |

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

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

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field、 when exposed to a CMOS device、 can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible、 and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry、 a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container、 static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin、 the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way、 the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state、 extra electromagnetic noise is induced in the vicinity of the LSI、 an associated shoot-through current flows internally、 and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset、 only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution、 wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset、 ensure that the reset line is only released after full stabilization of the clock signal. Additionally、 when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress、 wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise、 for example、 the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed、 and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another、 for example to a product with a different part number、 confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity、 layout pattern、 and other factors、 which can affect the ranges of electrical characteristics、 such as characteristic values、 operating margins、 immunity to noise、 and amount of radiated noise. When changing to a product with a different part number、 implement a system-evaluation test for the given product.

# *Notice*

1. Descriptions of circuits、 software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits、 software、 and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits、 software、 or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents、 copyrights、 or other intellectual property rights of third parties、 by or arising from the use of Renesas Electronics products or technical information described in this document、 including but not limited to、 the product data、 drawings、 charts、 programs、 algorithms、 and application examples.
3. No license、 express、 implied or otherwise、 is granted hereby under any patents、 copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties、 and obtaining such licenses for the lawful import、 export、 manufacture、 sales、 utilization、 distribution or other disposal of any products incorporating Renesas Electronics products、 if required.
5. You shall not alter、 modify、 copy、 or reverse engineer any Renesas Electronics product、 whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration、 modification、 copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade、 as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles、 trains、 ships、 etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document、 Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.)、 or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet、 user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products、 Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach、 including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS、 OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION、 ATTACK、 VIRUSES、 INTERFERENCE、 HACKING、 DATA LOSS OR THEFT、 OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE、 TO THE EXTENT PERMITTED BY APPLICABLE LAW、 RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES、 EXPRESS OR IMPLIED、 WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE、 INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY、 OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products、 refer to the latest product information (data sheets、 user’s manuals、 application notes、 “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook、 etc.)、 and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings、 operating power supply voltage range、 heat dissipation characteristics、 installation、 etc. Renesas Electronics disclaims any and all liability for any malfunctions、 failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products、 semiconductor products have specific characteristics、 such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document、 Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury、 injury or damage caused by fire、 and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products、 such as safety design for hardware and software、 including but not limited to redundancy、 fire control and malfunction prevention、 appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical、 you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances、 including without limitation、 the EU RoHS Directive、 and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture、 use、 or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products、 or any other party who distributes、 disposes of、 or otherwise sells or transfers the product to a third party、 to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted、 reproduced or duplicated in any form、 in whole or in part、 without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.5.0-1 October 2020)

## Corporate Headquarters

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

## Trademarks



## 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/.