EPSON
EXCEED YOUR VISION

# S1C17 M01/W22/W23/W15
# Self-Modifying Software (FLS) Manual

SEIKO EPSON CORPORATION

Rev.1.1

## Evaluation board/kit and Development tool important notice

1. This evaluation board/kit or development tool is designed for use for engineering evaluation, demonstration, or development purposes only. Do not use it for other purpose. It is not intended to meet the requirement of design for finished product.
2. This evaluation board/kit or development tool is intended for use by an electronics engineer, and it is not the product for consumer. The user should use this goods properly and safely. Seiko Epson dose not assume any responsibility and liability of any kind of damage and/or fire coursed by usage of it. User should cease to use it when any abnormal issue occurs even during proper and safe use.
3. The part used for this evaluation board/kit or development tool is changed without any notice.

## NOTICE

## Summary

This material is intended to provide reference materials for application programs to self-modify internal flash memory by using the FLS17 M01/W22/W23/W15 library, the erasing/writing software of flash memory embedded in S1C17M01/W22/W23/W15.

## Operating Environment

- PC
  GNU17 (S5U1C17001C) development tool installed*
  ICDmini USB driver installed
- ICDmini (S5U1C17001H2, S5U1C17001H3)
  The USB cable is necessary for connecting to PC.
- Target system (user target board or our company's evaluation board)
- S1C17xxx self-modifying software package (this package)

## Precautions for Usage

The library which comes with this package is only a sample. Our company does not take any responsibility for any problems caused by this library. You should fully verify the behavior of the product.

This material is common to S1C17M01/W22/W23/W15.
In this material, "xxx" of S1C17 (xxx) represents the concerned model name of M01/W22/W23/W15.

To supply power from outside, fully take time until the power supply becomes stable.
When the operation is finished, stop the power supply from outside

# Table of Contents

# 1. Overview

The self-modifying software package provides a library to rewrite program codes and data in the flash memory embedded in target models. The function calls from an application program can implement the operations to erase and write flash memory by linking this library with the application program.

When the function calls from the application program erase and write the flash memory, ICDmini (S5U1C17001H2, S5U1C17001H3) becomes unnecessary.
Note: ICDmini is used for debugging and writing an initial program.

## 1.1 Function

This section describes the functions to be provided by the library.

Flash memory erase function:
flash_erase(char * flsTopAdd, unsigned short startSct, unsigned short endSct)

> The sectors of the flash memory embedded in S1C17 (xxx) are erased by specifying the start address of the flash memory, and start and end sectors to be erased.
> The sectors to be erased can be 1 to 16 for M01, 1 to 32 for W22/W15, and 1 to 48 for W23.

Flash memory write function:
flash_load(char * loadAdd, unsigned short loadSize, unsigned char* pData)

> The data in the memory is written into the flash memory by specifying the start destination address, data size, and pointer to data to be written.
> Note: The scope of the data to be written is not checked in the library.

**<Points to note on this package>**

1. This package is created for developing programs in GNU17 (S5U1C17001C).
2. The verify function is incorporated in the flash memory erase/write functions.

## 1.2 Folder Configuration

This section describes the folder configuration of this package:

```
+ s1c17(xxx)self_r1x
    + lib                                  : Self-modifying library
    + c17(xxx)_sample_gnu17v2              : Sample program for GNU17 Ver.2.x
    + c17(xxx)_sample_gnu17v3              : Sample program for GNU17 Ver.3.x
    - s1c17m01_w22_w23_w15flsan_i_revXXX.pdf : Manual (Japanese)
    - s1c17m01_w22_w23_w15flsan_e_revXXX.pdf : Manual (English)
    - s1c17(xxx)self_notes_j.txt           : Notes for each MCU model (Japanese)
    - s1c17(xxx)self_notes_e.txt           : Notes for each MCU model (English)
    - License_e.txt                        : Software license agreement (English)
```

## 1.3 File Configuration

The table below lists the library file configuration.

Table 1 s1c17(xxx)self_r1x /lib

| Filename | Function |
|---|---|
| fls17(xxx)RAM.o | Erases/writes the flash memory of S1C17 (xxx) |
| fls17(xxx)ROM.o | Erases/writes the flash memory of S1C17 (xxx) |
| fl_self.h | Header file for declaring functions |

The table blow lists the file configuration of sample program.

Table 2 c17(xxx)_sample

| File/folder name | Function |
|---|---|
| Lib | Self-modifying library (folder) |
| boot.c | boot program |
| main.c | main program |
| data.c | Data (before update) |
| update.c | Data for update |

# 2. How to Use Library

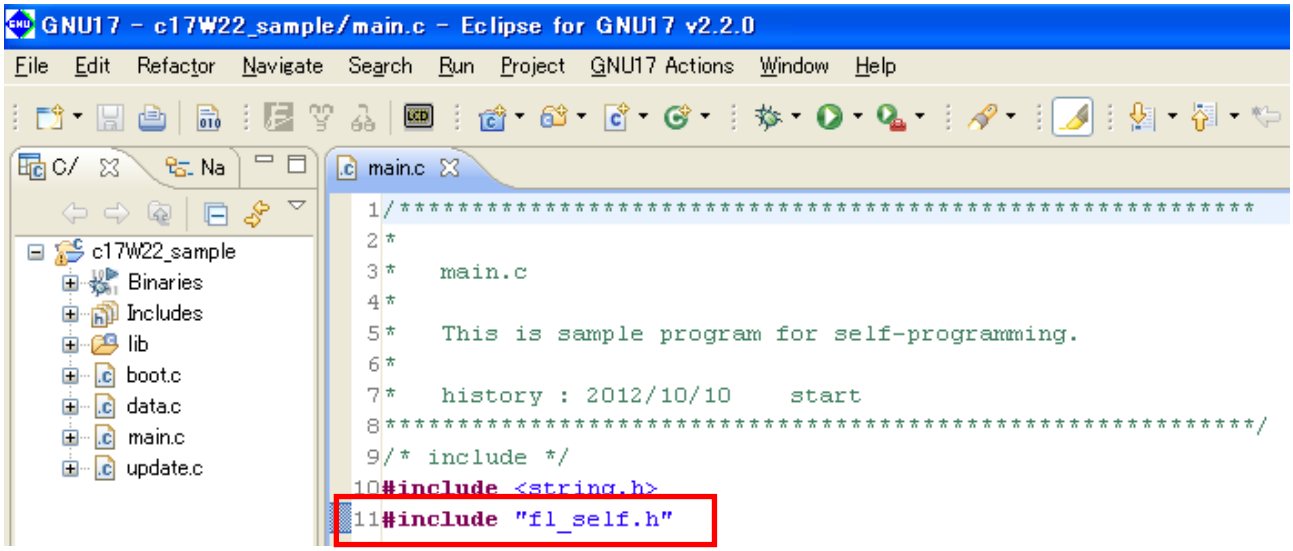
This chapter describes necessary information and precautions for using this library. This chapter also describes a sample program using this library.

## 2.1 How to Use Library in Application Program

This section describes necessary information for using this library in the source files of application programs.
In addition, for information on how to incorporate the library into a project of application program, refer to Appendix B. How to Incorporate Library into Project.

### 1. Declaration of Header File

Declare to include "fl_self.h" in a source file using this library.
Note: If an include path is not set, specify the path before the inclusion.




### 2. Adding Flash Memory Erase/Write functions

Add the functions of this library to erase/write flash memory to the source code of the application program.
For the function specification, refer to 3. Library Specification.

```
int update(void) {
        int result;

        // erase 0xB000-0xBFFF
        result = flash_erase((char *)0x8000, 7, 7);

        if(result == FLS_E_SUCCESS) {
                // write data to 0xb000
                result = flash_load((char *)0xB000, 16, updateLineBit);
        }

        return result;
}
```

## 2.2 Flash Programming Voltage VPP

When erasing/writing flash memory, supply the Flash programming power (VPP power).
The procedures to supply VPP power are as follows. In addition, stop the VPP power supply other than when erasing/writing flash memory.

① Check to see that VDD is not less than 1.8V.
② Put the desired data in RAM.
③ Supply the VPP power.
④ Wait until the VPP voltage becomes stable.
⑤ Execute the flash_erase function to erase specified area.
⑥ Execute the flash_load function to write data into the specified address in erased area.
⑦ To write data into another address in erased area, repeat step ⑥.
⑧ Stop the VPP power.

Refer to "S1C17 (xxx) Technical Manual" for Flash programming voltage and basic external connection diagram.

## 2.3 Internal RAM Usage

This library uses internal RAM area.
Please refer Notes for each MCU model “s1c17(xxx)self_notes_e.txt” to know internal RAM usage.

## 2.4 Precautions for Using Library

When using this library, be careful about the followings:

- Disable interrupts before the flash_erase and flash_load functions.
- Do not destruct the area where the library is laid out while executing library.
- When using this library, be aware of rewritable count of flash memory. For information about flash memory specification, refer to "S1C17 (xxx) Technical Manual."
- This library works as follows:
  1. The library uses 16bit timer (T16), ch. 0. Therefore, the register of 16bit timer, ch.0 is changed. Be aware when a program uses both the 16bit timer and this library.
  2. In W22/W23/W15, VD1 is changed to 1.8V while rewriting flash memory. After rewriting flash memory, VD1 is reverted back to 1.4V. When using this library, make sure to stop all peripheral circuits set by the customer.
  3. The system clock is changed to: IOSC for M01, OSC3 for W22/W23/W15.
  4. In W22/W23/W15, this library reverts CGL OSC3 Control Register of clock generator (CLG) back to initial value. Therefore, CLG OSC3 Control Register is changed. Be aware when a program uses both CLG OSC3 Control Register and this library.

## 2.5 Sample Program

### 1. Sample Program Specification

The sample program performs the following operation using this library.

- Erase a sector in the address 0xB000 area and then write 16byte of data.

### 2. Preparation

To run the sample program in IDE, refer to the procedures below. Also, when using the library, keep in mind the items described in the sections 2.1 to 2.4 above.

① Import a project
Launch IDE and import the sample program.
② Build
Use IDE to build the sample program.
③ Connect
Connect ICDmini and target system to PC.
④ Unlock Flash security
To debug the sample program with IC supporting the Flash security, unlock the Flash security.
⑤ Load program
Use IDE to load the sample program.

⑥ Supply VPP power
Supply VPP power.
⑦ Run
Reset the target system to run the program.

For more information, refer to "S1C17 (xxx) Technical Manual," "S5U1C17001C Manual," and "S5U1C17001H User Manual (ICDmini)."

### 3. Operations Overview

① Erase the sector 7 (0xB000 to 0xB7FF) in internal flash memory (main.c/flash_erase).
- Start address of flash memory: 0x8000
- Erase start sector of flash memory: 7
- Erase end sector of flash memory: 7

② Write the update data updateLineBit[] into the erased sector 0xB000 (main.c/flash_load).
- Write destination address: 0xB000
- Write data size: 16byte
- Pointer to writing data: updateLineBit

The data starting 0xB000 is as follows before and after the rewriting.

Before the rewriting (0xB000)

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

After the rewriting (0xB000)

```
0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00
```

Confirm the rewritten result by the memory dump.
For information on the sector numbers and corresponding addresses, refer to Appendix A. Flash Memory Specification.
For information about flash_erase, flash_load functions, refer to 3.1 Flash Memory Erase/Write Function Details.

# 3. Library Specification

## 3.1 Flash Memory Erase/Write Function Details

This section describes the functions described in fl_17(xxx).o (fl_17(xxx).c).

**Erase flash memory**

| **Function Name** | | |
|---|---|---|
| flash_erase(char * flsTopAdd, unsigned short startSct, unsigned short endSct) | | |
| **Argument** | | |
| flsTopAdd | unsigned long | Represents the start address of flash memory. |
| startSct | unsigned long | Represents the erase start address of flash memory. |
| endSct | unsigned long | Represents the erase end address of flash memory. |
| **Return Value** | | |
| int | Represents the erase result (error code) of flash memory. | |
| **Function** | | |
| Erase the flash memory according to the parameter specified by arguments.<br>① Check whether the arguments are correct.<br>② Erase the memory.<br>③ Return a return value. | | |
| **Remarks** | | |
| The scope of sector numbers specified by second and third arguments: 1 to 16 for M01, 1 to 32 for W22/W15, and 1 to 48 for W23. | | |

**Write flash memory**

| **Function Name** | | |
|---|---|---|
| flash_load(char * loadAdd, unsigned short loadSize, unsigned char* pData) | | |
| **Argument** | | |
| loadAdd | unsigned long | Represents the write destination address. |
| loadSize | unsigned long | Represents the write data size. |
| pData | unsigned char* | Represents a pointer to the write data. The pointer should point the RAM space. |
| **Return Value** | | |
| int | Represents the write result (error code) of flash memory. | |
| **Function** | | |
| Write data into the flash memory according to the parameter specified by arguments.<br>① Convert virtual address into physical address.<br>② Write data.<br>- Initialize the setting.<br>- Check whether the desired data and the data in the destination area match each other.<br>- If they do not match each other, write data.<br>③ Return a return value. | | |
| **Remarks** | | |
| The write data unit is "byte (8bit)." | | |

## 3.2 Error Code Definition

Table 3 Error Code

| Definition Name | Value | Description |
|---|---|---|
| FLS_E_SUCCESS | 0 | The flash operation ended successfully |
| FLS_E_PROGRAM_COUNT_OVER_ERR | 1 | Program count error<br>Returns if the verify count exceeds the maximum program count by verifying data during the writing/erasing (compare the data with desired data during writing and 0xFFFF during erasing). |

| | | |
|---|---|---|
| FLS_E_STRAT_SCT_ERR | 3 | The erase start sector is out of scope<br>Returns if the erase start sector is less than zero or exceeds the maximum sector number. |
| FLS_E_END_SCT_ERR | 4 | The erase end sector is out of scope<br>Returns if the erase end sector is less than zero or exceeds the maximum sector number. |
| FLS_E_TOP_ADDRESS_ERR | 5 | Start address error<br>Returns if the first argument (start address of flash memory) of the FLASH_ERASE function is not equal to the start address value of flash memory. |

# Appendix

## A. Flash Memory Specification

This section describes internal flash memory specification of each model. For information about internal flash memory, refer to "S1C17 (xxx) Technical Manual."

### 1. S1C17M01 Internal Flash Memory Configuration

- Sector number : 16sectors
- Memory size : 32Kbyte
- Erase unit : Sector (2Kbyte)
- Write unit : Byte (8bit)
- Write time : Initial sector programming time: 109msec (typical)
  Initial sector erase time: 121msec (typical)
- Flash programming voltage ($V_{PP}$) : erase: 7.5V, write: 7.5V
  To supply power from outside, fully take time until the power supply becomes stable.

  When the operation is finished, stop the power supply from outside.

Table 4 Correspondence between S1C17M01 Memory Address and Flash Sector

| S1C17M01 Address | Flash Sector Number |
|---|---|
| 0x0f800 to 0x0ffff | 16 |
| 0x0f000 to 0x0f7ff | 15 |
| 0x0e800 to 0x0efff | 14 |
| 0x0e000 to 0x0e7ff | 13 |
| 0x0d800 to 0x0dfff | 12 |
| 0x0d000 to 0x0d7ff | 11 |
| 0x0c800 to 0x0cfff | 10 |
| 0x0c000 to 0x0c7ff | 9 |
| 0x0b800 to 0x0bfff | 8 |
| 0x0b000 to 0x0b7ff | 7 |
| 0x0a800 to 0x0afff | 6 |
| 0x0a000 to 0x0a7ff | 5 |
| 0x09800 to 0x09fff | 4 |
| 0x09000 to 0x097ff | 3 |
| 0x08800 to 0x08fff | 2 |
| 0x08000 to 0x087ff | 1 |

## 2. S1C17W22 Internal Flash Memory Configuration

- Sector number : 32sectors
- Memory size : 64Kbyte
- Erase unit : Sector (2Kbyte)
- Write unit : Byte (8bit)
- Write time : Initial sector programming time: 220msec (typical)
  Initial sector erase time: 208msec (typical)
- Flash programming voltage ($V_{PP}$) : erase: 7.5V, write: 7.5V
  To supply power from outside, fully take time until the power supply becomes stable.

  When the operation is finished, stop the power supply from outside.

Table 5 Correspondence between S1C17W22 Memory Address and Flash Sector

| S1C17W22 Address | Flash Sector Number |
|---|---|
| 0x17800 to 0x17fff | 32 |
| 0x17000 to 0x177ff | 31 |
| 0x16800 to 0x16fff | 30 |
| 0x16000 to 0x167ff | 29 |
| 0x15800 to 0x15fff | 28 |
| 0x15000 to 0x157ff | 27 |
| 0x14800 to 0x14fff | 26 |
| 0x14000 to 0x147ff | 25 |
| 0x13800 to 0x13fff | 24 |
| 0x13000 to 0x137ff | 23 |
| 0x12800 to 0x12fff | 22 |
| 0x12000 to 0x127ff | 21 |
| 0x11800 to 0x11fff | 20 |
| 0x11000 to 0x117ff | 19 |
| 0x10800 to 0x10fff | 18 |
| 0x10000 to 0x107ff | 17 |
| 0x0f800 to 0x0ffff | 16 |
| 0x0f000 to 0x0f7ff | 15 |
| 0x0e800 to 0x0efff | 14 |
| 0x0e000 to 0x0e7ff | 13 |
| 0x0d800 to 0x0dfff | 12 |
| 0x0d000 to 0x0d7ff | 11 |
| 0x0c800 to 0x0cfff | 10 |
| 0x0c000 to 0x0c7ff | 9 |
| 0x0b800 to 0x0bfff | 8 |
| 0x0b000 to 0x0b7ff | 7 |
| 0x0a800 to 0x0afff | 6 |
| 0x0a000 to 0x0a7ff | 5 |
| 0x09800 to 0x09fff | 4 |
| 0x09000 to 0x097ff | 3 |
| 0x08800 to 0x08fff | 2 |
| 0x08000 to 0x087ff | 1 |

### 3. S1C17W23 Internal Flash Memory Configuration

- Sector number : 48sectors
- Memory size : 96Kbyte
- Erase unit : Sector (2Kbyte)
- Write unit : Byte (8bit)

- Write time : Initial sector programming time: 220msec (typical)
  Initial sector erase time: 208msec (typical)
- Flash programming voltage ($V_{PP}$) : erase: 7.5V, write: 7.5V
  To supply power from outside, fully take time until the power supply becomes stable.

  When the operation is finished, stop the power supply from outside.

Table 6 Correspondence between S1C17W23 Memory Address and Flash Sector

| S1C17W23 Address | Flash Sector Number |
|---|---|
| 0x17800 to 0x17fff | 32 |
| 0x17000 to 0x177ff | 31 |
| 0x16800 to 0x16fff | 30 |
| 0x16000 to 0x167ff | 29 |
| 0x15800 to 0x15fff | 28 |
| 0x15000 to 0x157ff | 27 |
| 0x14800 to 0x14fff | 26 |
| 0x14000 to 0x147ff | 25 |
| 0x13800 to 0x13fff | 24 |
| 0x13000 to 0x137ff | 23 |
| 0x12800 to 0x12fff | 22 |
| 0x12000 to 0x127ff | 21 |
| 0x11800 to 0x11fff | 20 |
| 0x11000 to 0x117ff | 19 |
| 0x10800 to 0x10fff | 18 |
| 0x10000 to 0x107ff | 17 |
| 0x0f800 to 0x0ffff | 16 |
| 0x0f000 to 0x0f7ff | 15 |
| 0x0e800 to 0x0efff | 14 |
| 0x0e000 to 0x0e7ff | 13 |
| 0x0d800 to 0x0dfff | 12 |
| 0x0d000 to 0x0d7ff | 11 |
| 0x0c800 to 0x0cfff | 10 |
| 0x0c000 to 0x0c7ff | 9 |
| 0x0b800 to 0x0bfff | 8 |
| 0x0b000 to 0x0b7ff | 7 |
| 0x0a800 to 0x0afff | 6 |
| 0x0a000 to 0x0a7ff | 5 |
| 0x09800 to 0x09fff | 4 |
| 0x09000 to 0x097ff | 3 |
| 0x08800 to 0x08fff | 2 |
| 0x08000 to 0x087ff | 1 |

| S1C17W23 Address | Flash Sector Number |
|---|---|
| 0x1f800 to 0x1ffff | 48 |
| 0x1f000 to 0x1f7ff | 47 |
| 0x1e800 to 0x1efff | 46 |
| 0x1e000 to 0x1e7ff | 45 |
| 0x1d800 to 0x1dfff | 44 |
| 0x1d000 to 0x1d7ff | 43 |
| 0x1c800 to 0x1cfff | 42 |
| 0x1c000 to 0x1c7ff | 41 |
| 0x1b800 to 0x1bfff | 40 |
| 0x1b000 to 0x1b7ff | 39 |
| 0x1a800 to 0x1afff | 38 |
| 0x1a000 to 0x1a7ff | 37 |
| 0x19800 to 0x19fff | 36 |
| 0x19000 to 0x197ff | 35 |
| 0x18800 to 0x18fff | 34 |
| 0x18000 to 0x187ff | 33 |

## 4. S1C17W15 Internal Flash Memory Configuration

- Sector number : 32sectors
- Memory size : 64Kbyte
- Erase unit : Sector (2Kbyte)
- Write unit : Byte (8bit)

- Write time : Initial sector programming time: 220msec (typical)
  Initial sector erase time: 208msec (typical)
- Flash programming voltage ($V_{PP}$) : erase: 7.5V, write: 7.5V
  To supply power from outside, fully take time until the power supply becomes stable.

  When the operation is finished, stop the power supply from outside.

Table 7 Correspondence between S1C17W15 Memory Address and Flash Sector

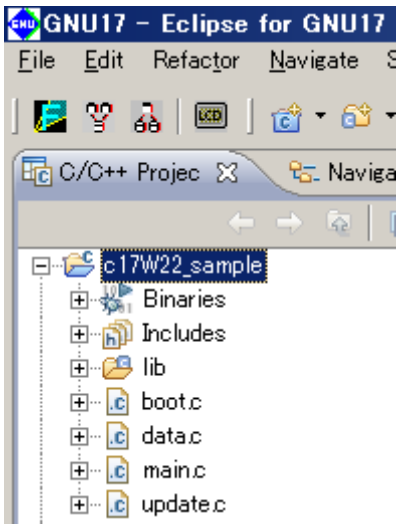
| S1C17W15 Address | Flash Sector Number |
|---|---|
| 0x17800 to 0x17fff | 32 |
| 0x17000 to 0x177ff | 31 |
| 0x16800 to 0x16fff | 30 |
| 0x16000 to 0x167ff | 29 |
| 0x15800 to 0x15fff | 28 |
| 0x15000 to 0x157ff | 27 |
| 0x14800 to 0x14fff | 26 |
| 0x14000 to 0x147ff | 25 |
| 0x13800 to 0x13fff | 24 |
| 0x13000 to 0x137ff | 23 |
| 0x12800 to 0x12fff | 22 |
| 0x12000 to 0x127ff | 21 |
| 0x11800 to 0x11fff | 20 |
| 0x11000 to 0x117ff | 19 |
| 0x10800 to 0x10fff | 18 |
| 0x10000 to 0x107ff | 17 |
| 0x0f800 to 0x0ffff | 16 |
| 0x0f000 to 0x0f7ff | 15 |
| 0x0e800 to 0x0efff | 14 |
| 0x0e000 to 0x0e7ff | 13 |
| 0x0d800 to 0x0dfff | 12 |
| 0x0d000 to 0x0d7ff | 11 |
| 0x0c800 to 0x0cfff | 10 |
| 0x0c000 to 0x0c7ff | 9 |
| 0x0b800 to 0x0bfff | 8 |
| 0x0b000 to 0x0b7ff | 7 |
| 0x0a800 to 0x0afff | 6 |
| 0x0a000 to 0x0a7ff | 5 |
| 0x09800 to 0x09fff | 4 |
| 0x09000 to 0x097ff | 3 |
| 0x08800 to 0x08fff | 2 |
| 0x08000 to 0x087ff | 1 |

## B. How to Incorporate Library into Project (GNU17 Ver.2.x)

This section describes how to use the library in this package in GNU17 Ver.2.x.
(Now, W22 is taken as an example)
For more information about how to use GNU17 Ver.2.x, refer to the compiler manual.

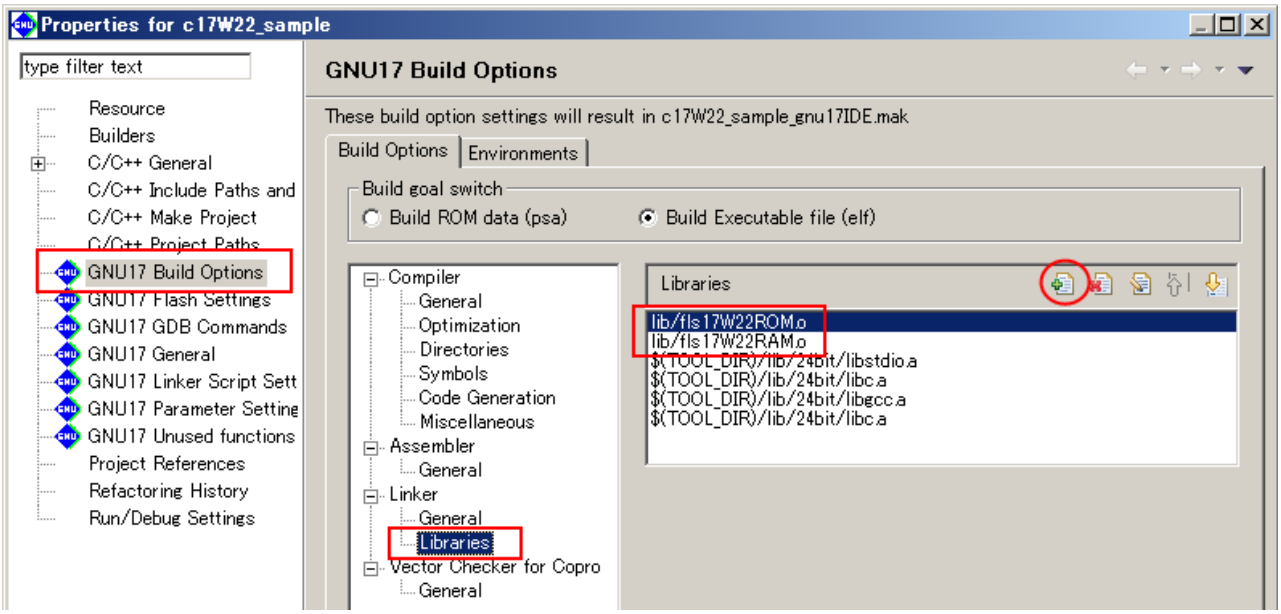
### 1. Adding Library and Header File

Import the lib folder in the package into the project folder.



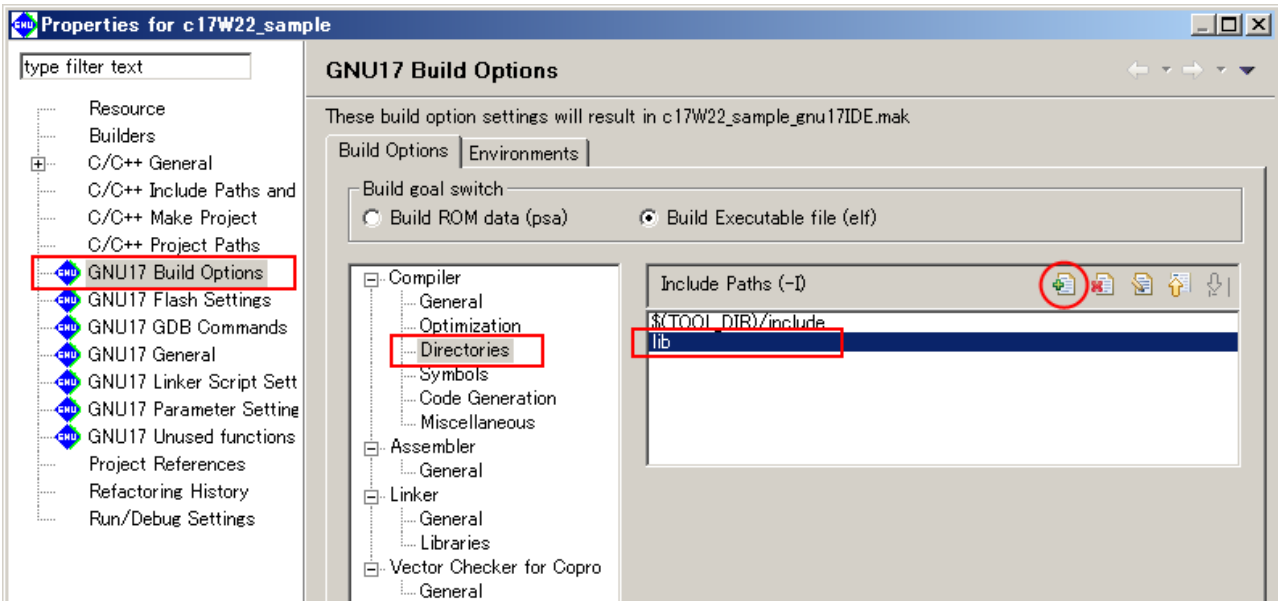

### 2. Library Setting

In order to use the library imported, add the library setting.
Click [Properties]-[GNU17 Build Options]-[Linker]-[Libraries] of the project, click the button circled in red in the figure below, and select and add "fls17(xxx)ROM.o ", "fls17(xxx)RAM.o " in the lib folder.



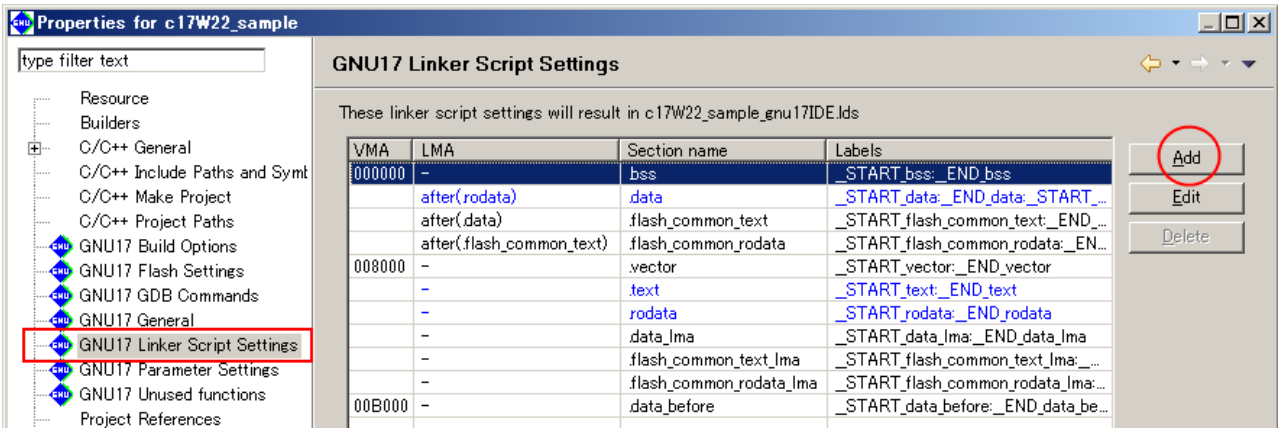
### 3. Include Path Setting

In order to use "fl_self.h" in the lib folder, set the include path.
Click [Properties]-[GNU17 Build Options]-[Directories] of the project, click the button circled in red in the figure below, and set the include path for the lib folder.
Note: This setting is not necessary if the include path is specified directly in the source file.

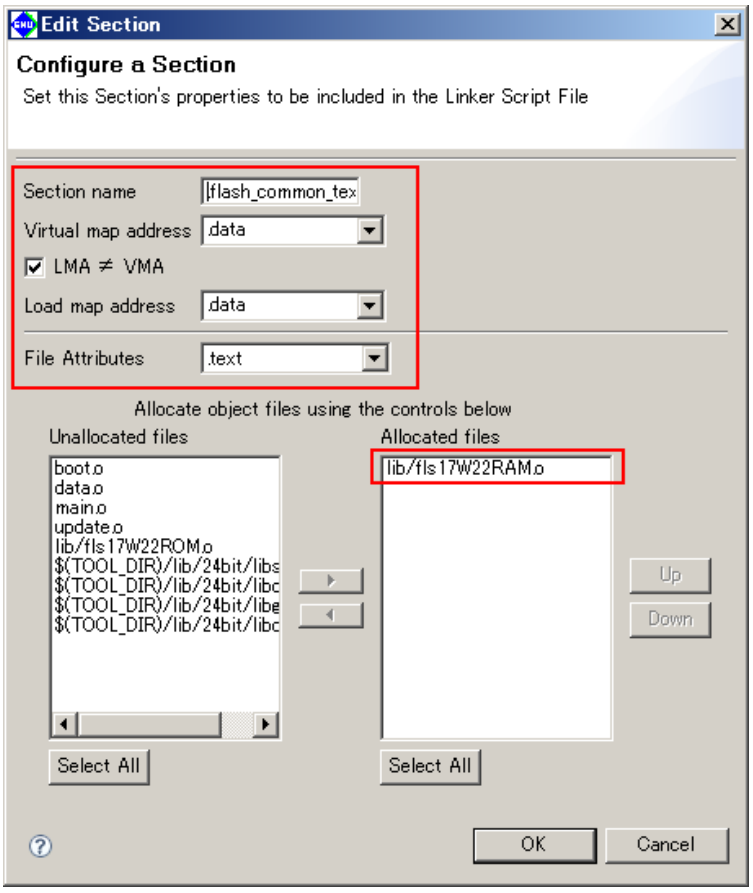


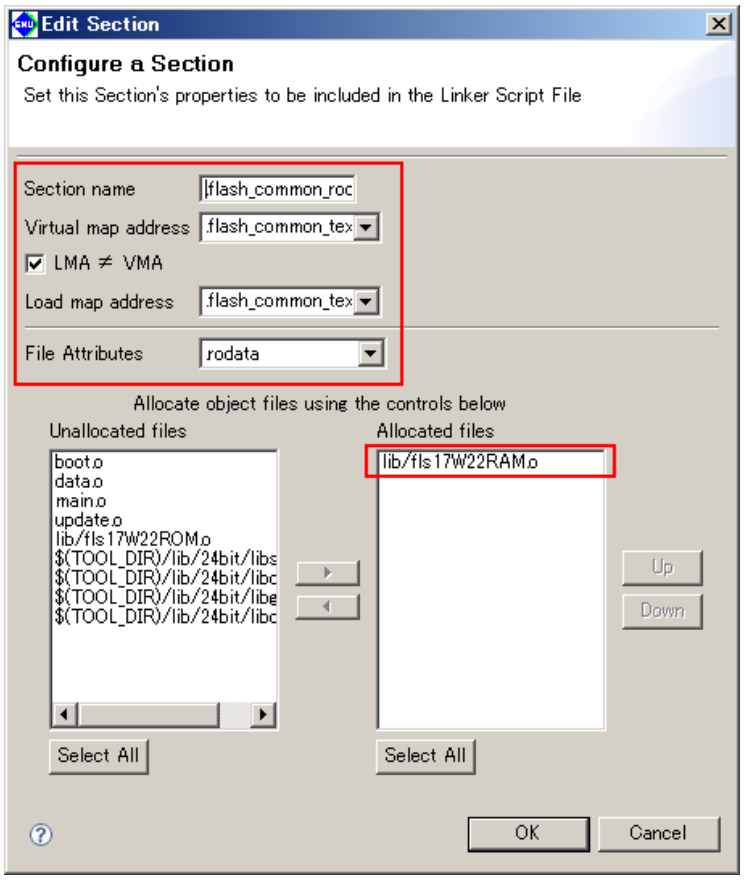
## 4. Linker Script Setting

Set the linker script of the library imported.
Click [Properties]-[GNU17 Linker Script Settings] of the project, click the button circled in red in the figure below, and add sections for the library layout.




Add the sections: .flash_common_text and .flach_common_rodata. Add the sections whose name begins with a dot (".") as written above, referring to figures below.
As shown in the figure below, allocate "fls1(xxx)RAM.o" to each section above. Since "fls17(xxx)RAM.o" is a library for issuing control commands to flash memory and need to be executed in internal RAM, allocate execution addresses (VMA) in internal RAM. Note that it is not necessary to allocate "fls17(xxx)ROM.o" in RAM.




Edit Section
Configure a Section
Set this Section's properties to be included in the Linker Script File
Section name
flash_common_roc
Virtual map address
.flash_common_tex
LMA ≠ VMA
Load map address
.flash_common_tex
File Attributes
rodata
Allocate object files using the controls below
Unallocated files
boot.o
data.o
main.o
update.o
lib/fls17W22ROM.o
$(TOOL_DIR)/lib/24bit/libs
$(TOOL_DIR)/lib/24bit/libc
$(TOOL_DIR)/lib/24bit/libg
$(TOOL_DIR)/lib/24bit/libc
Allocated files
lib/fls17W22RAM.o
Up
Down
Select All
Select All
OK
Cancel

## C. How to Incorporate Library into Project (GNU17 Ver.3.x)

This section describes how to use the library in this package in GNU17 Ver.3.x.
For more information about how to use GNU17 Ver.3.x, refer to the compiler manual.

### 1. Adding Library and Header File

Import the "lib" folder in the package into the project folder "src".

### 2. Library Setting

In order to use the library imported, add the library setting.
Select [Properties]-[C/C++ Build]-[Enviroment] of the project, and add "fls17(xxx) ROM.o", "fls17(xxx) RAM.o" in the "src¥lib" folder to Value of Variable "GCC17_USER_LIBS".

..¥src¥lib¥fls17(xxx)RAM.o;..¥src¥lib¥fls17(xxx)ROM.o

### 3. Include Path Setting

In order to use "fl_self.h" in the lib folder, set the include path.
Select [Properties]-[C/C++ Build]-[Settings]-[Tool Settings]-[Cross GCC Compiler]-[Includes] of the project, and set the include path for the "src¥lib" folder.

"${workspace_loc:/${ProjName}/src/lib}"

### 4. Linker Script Setting

Set the linker script of the library.
The example of linker script described for self modify internal flash exists in the following folder. Copy this script to the project folder.

¥c17(xxx)_sample_gnu17v3¥selfmodifying.x

Select [Properties]-[C/C++ Build]-[Settings]-[Tool Settings]-[Cross GCC Linker]-[Miscellaneous] of the project, and set the copied linker script file at [Other options].

-T ..¥selfmodifying.x

In this linker script, the following symbols necessary for the operation of the library are defined, and the execution address of the library "fls17(xxx)RAM.o" is arranged in internal RAM.

```
__START_flash_common_text_lma
__START_flash_common_text
__END_flash_common_text
__START_flash_common_rodata_lma
__START_flash_common_rodata
__END_flash_common_rodata
```

Moreover, this script sets that "fls17(xxx) RAM.o" is not arranged in ROM by the following descriptions.

```
*(EXCLUDE_FILE (*fls17*RAM.o ) .text)
*(EXCLUDE_FILE (*crt0.o *fls17*RAM.o) .rodata)
```

In this script, arrange the data to be rewritten to the ".updatable" section. When such a section is unnecessary, delete the ".udatable" section.

# Revision History

Attachment-1

| Rev. No. | Date | Page | Category | Contents |
|---|---|---|---|---|
| Rev 1.0 | 2013/12/26 | All | new | |
| Rev 1.1 | 2015/08/07 | 1, 3, 5, 13 | modify | Addition for GNU17V3 |
| | | 17 | additional | Addition for GNU17V3 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

EPSON

# International Sales Operations

## AMERICA

**EPSON ELECTRONICS AMERICA, INC.**
214 Devcon Drive,
San Jose, CA 95112, USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

## EUROPE

**EPSON EUROPE ELECTRONICS GmbH**
Riesstrasse 15, 80992 Munich,
GERMANY
Phone: +49-89-14005-0 FAX: +49-89-14005-110

## ASIA

**EPSON (CHINA) CO., LTD.**
7F, Jinbao Bldg., No.89 Jinbao St.,
Dongcheng District,
Beijing 100005, CHINA
Phone: +86-10-8522-1199 FAX: +86-10-8522-1125

**SHANGHAI BRANCH**
7F, Block B, Hi-Tech Bldg., 900 Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5577 FAX: +86-21-5423-4677

**SHENZHEN BRANCH**
**12F, Dawning Mansion, Keji South 12th Road,**
**Hi-Tech Park, Shenzhen 518057, CHINA**
Phone: +86-755-2699-3828 FAX: +86-755-2699-3838

**EPSON TAIWAN TECHNOLOGY & TRADING LTD.**
14F, No. 7, Song Ren Road,
Taipei 110, TAIWAN
Phone: +886-2-8786-6688 FAX: +886-2-8786-6660

**EPSON SINGAPORE PTE., LTD.**
1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

**SEIKO EPSON CORP.**
**KOREA OFFICE**
5F, KLI 63 Bldg., 60 Yoido-dong,
Youngdeungpo-Ku, Seoul 150-763, KOREA
Phone: +82-2-784-6027 FAX: +82-2-767-3677

**SEIKO EPSON CORP.**
**MICRODEVICES OPERATIONS DIVISION**

**Device Sales & Marketing Department**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117

Document Code: 412668701
First Issue December 2013 Ⓥ
Revised August 2015 in JAPAN