EPSON
EXCEED YOUR VISION

S1C17 Series
Self-Modifying Software
Manual

SEIKO EPSON CORPORATION

Rev.1.3

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 purposes. It is not intended to meet the requirements of design for finished products.
2. This evaluation board/kit or development tool is intended for use by an electronics engineer and is not a consumer product. The user should use it properly and in a safe manner. Seiko Epson dose not assume any responsibility or liability of any kind of damage and/or fire coursed by the use of it. The 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 may be changed without any notice.

NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as, medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. When exporting the products or technology described in this material, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You are requested not to use, to resell, to export and/or to otherwise dispose of the products (and any technical information furnished, if any) for the development and/or manufacture of weapon of mass destruction or for other military purposes.

All brands or product names mentioned herein are trademarks and/or registered trademarks of their respective companies.

©SEIKO EPSON CORPORATION 2020, All rights reserved.

Summary

This material is intended to provide reference materials for application programs to self-modify internal flash memory by using the erasing/writing library of flash memory embedded in S1C17 series.

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

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 S1C17 series self-modifying software package. Please refer to the notes file for different specifications at each package.
Each package is different in the file composition according to the corresponding model. In this material, locations different according to each package are represented by “xxx” = corresponding model name.

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 S1C17 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 are erased by specifying the start address of the flash memory, and start and end sectors to be erased.
> The sectors to be erased is different depending on the S1C17 model.

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 the S1C17 series self-modifying software 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
    - 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)
```

Specifications different to each MCU model are described in the s1c17(xxx)self_notes_e.txt.

1.3 File Configuration

The table below lists the configuration of the flash memory erasing / writing library.

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

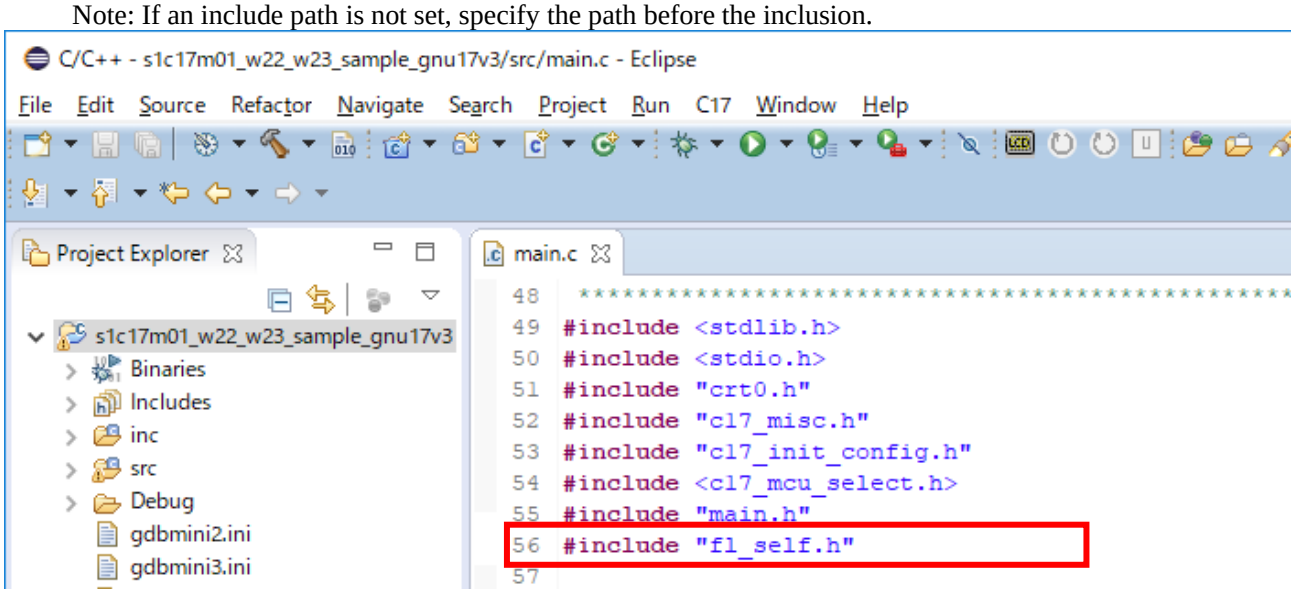
This chapter describes necessary information and precautions for using the flash memory erasing / writing library. This chapter also describes a sample program using this library.

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 A. 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 functions of the flash memory erasing / writing library 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 the corresponding MCU models technical manual of each self-modifying software package for Flash programming voltage and basic external connection diagram.

2.3 Internal RAM Usage

The flash memory erasing / writing library uses internal RAM area.
Please refer the Notes for each MCU model file to know internal RAM usage of each self-modifying software package.

2.4 Precautions for Using Library

When using the flash memory erasing / writing 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 corresponding MCU models technical manual of each self-modifying software package.
- When using this library, stop all on-chip peripheral circuits. The flash memory erasing / writing 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. The system clock is changed to High-Speed clock(OSC3 or IOSC) in using the library. Be aware when a program uses CLG Control Register in using the library.
- Refer the Notes for each MCU model file to know notes different according to each self-modifying software package.
- When using this library, connect a capacitor to the Vpp pin as shown in the basic external connection diagram in the corresponding MCU models technical manual, and disconnect the connection between the FLASH_VCC_OUT pin of ICDmini and the Vpp pin of the MCU.

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 the corresponding MCU models technical manual of each self-modifying software package, "S5U1C17001C Manual," and "S5U1C17001H User Manual (ICDmini)."

3. Operations Overview

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

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 Notes for each MCU model file.
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	char*	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. ① Check whether the arguments are correct. ② Erase the memory. ③ Return a return value.		
Remarks		
Refer the Notes(s1c17(xxx)self_notes_x.txt) for each MCU model file to know the scope of sector numbers specified by second and third arguments. Effective sector numbers are shown in “Flash memory specification” of the file.		

Write flash memory

Function Name		
flash_load(char * loadAdd, unsigned short loadSize, unsigned char* pData)		
Argument		
loadAdd	char*	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. ① Convert virtual address into physical address. ② Write data. - Initialize the setting. - Check whether the desired data and the data in the destination area match each other. - If they do not match each other, write data. 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 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 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 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 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.

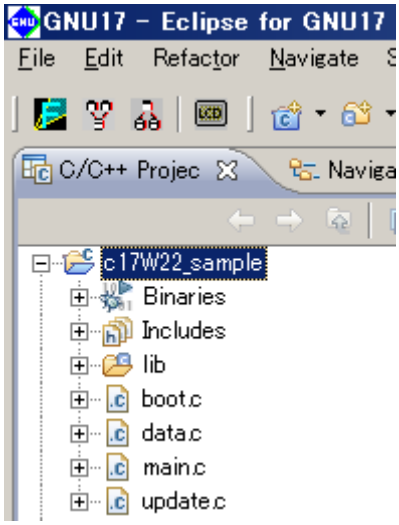
Appendix

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

This section describes how to use the flash memory erasing / writing library in this package in GNU17(S5U1C17001C).
(Now, the self-modifying sample program and the flash memory erasing / writing library for S1C17W22 is taken as an example)
For more information about how to use GNU17(S5U1C17001C), refer to the compiler manual.

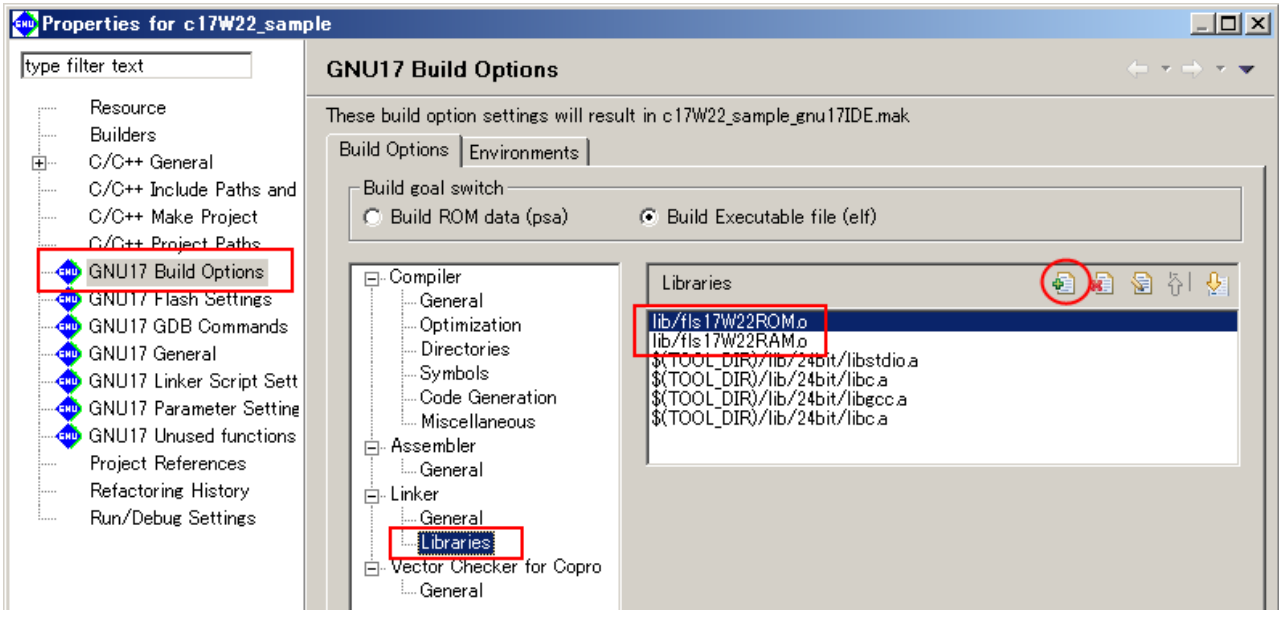
1. Adding Library and Header File

Import the lib folder in the S1C17 series self-modifying software package into the project folder.



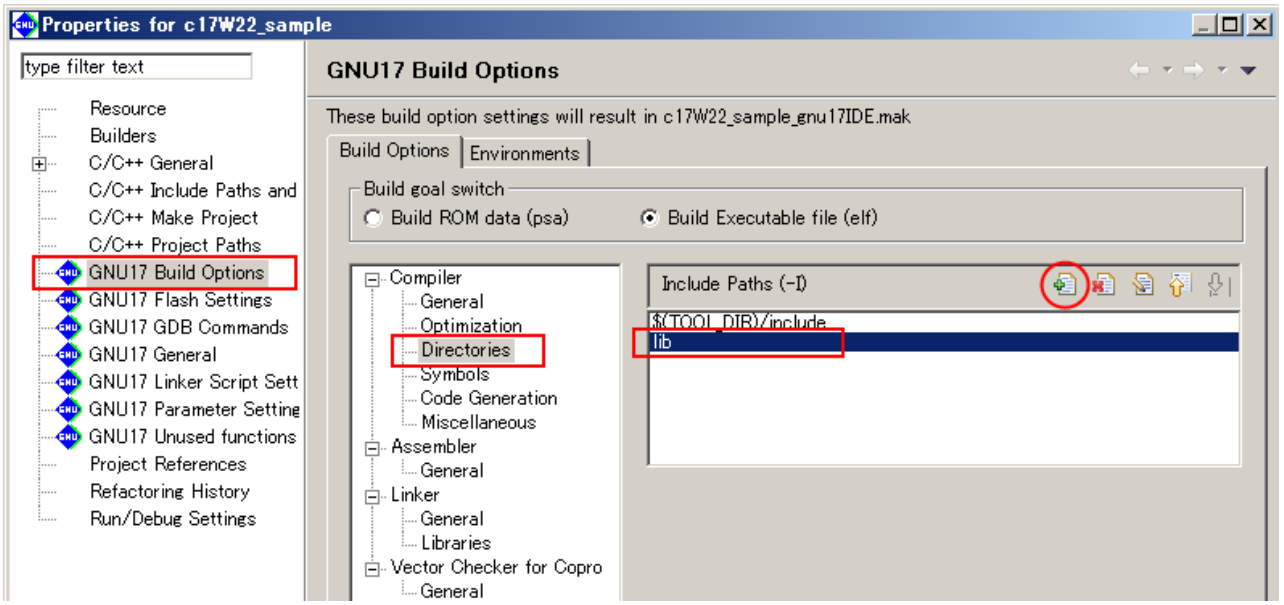
2. Library Setting

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.



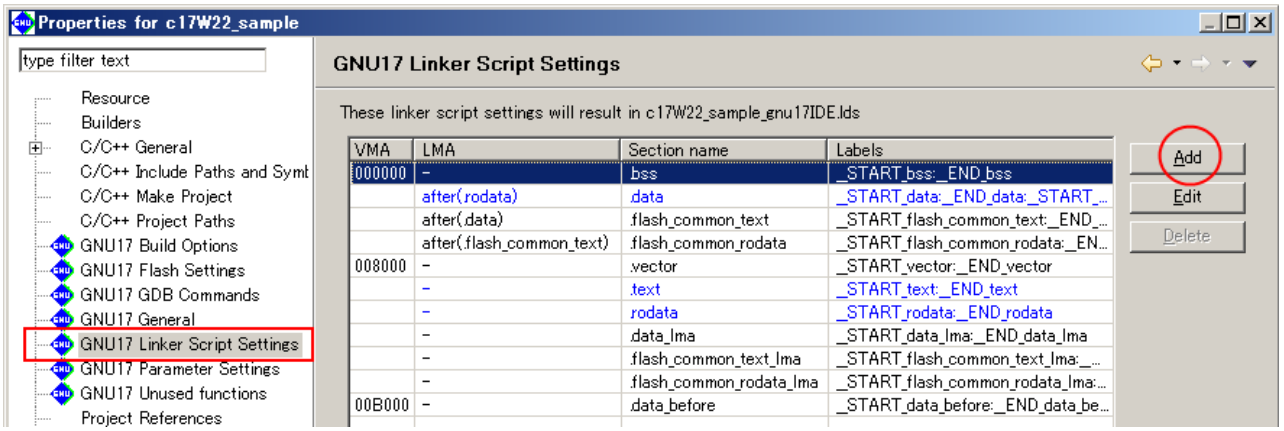
3. Include Path Setting

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.

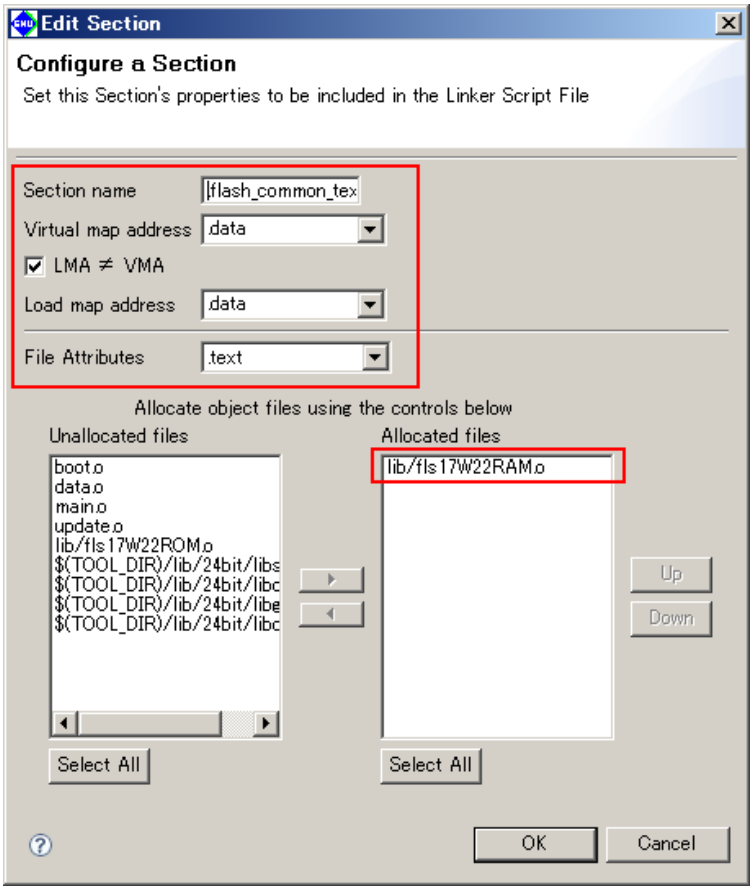


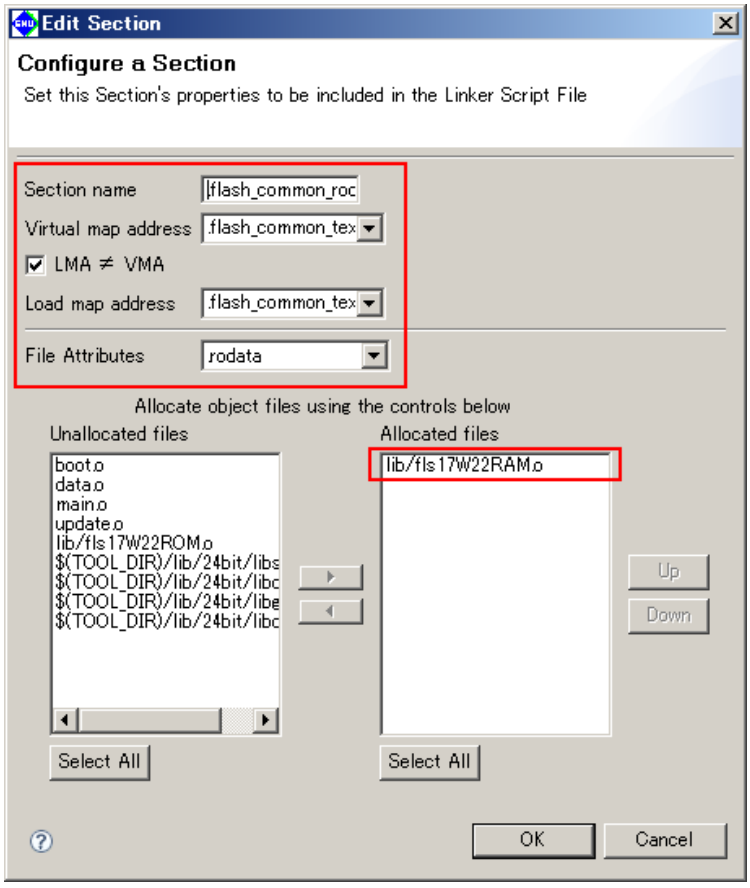
4. Linker Script Setting

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_tex
Virtual map address
.data
LMA ≠ VMA
Load map address
.data
File Attributes
.text
Allocate object files using the controls below
Unallocated files
Allocated 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/libe
$(TOOL_DIR)/lib/24bit/libc
lib/fls17W22RAM.o
Up
Down
Select All
Select All
OK
Cancel


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
Allocated 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/libe
$(TOOL_DIR)/lib/24bit/libc
lib/fls17W22RAM.o
Up
Down
Select All
Select All
OK
Cancel

B. 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. (Now, the self-modifying sample program and the flash memory erasing / writing library for S1C17W22 is taken as an example)

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¥fls17W22RAM.o;..¥src¥lib¥fls17W22ROM.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.

```
¥c17W22_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 "fls17W22RAM.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 "fls17W22RAM.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
Rev 1.2	2018/06/15	All	Modify	Standardization for all S1C17 series.
Rev 1.3	2020/05/27	4	Modify	Modified descriptions in Section 2.4, “Precautions on Use of Library.”

EPSON

International Sales Operations

America

Epson America, Inc.

Headquarter:
3840 Kilroy Airport Way
Long Beach, California 90806-2452 USA
Phone: +1-562-290-4677

San Jose Office:
214 Devcon Drive
San Jose, CA 95112 USA
Phone: +1-800-228-3964 or +1-408-922-0200

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.
4th Floor, Tower 1 of China Central Place, 81 Jianguo Road, Chaoyang District, Beijing 100025 China
Phone: +86-10-8522-1199 FAX: +86-10-8522-1120

Shanghai Branch
Room 1701 & 1704, 17 Floor, Greenland Center II,
562 Dong An Road, Xu Hui District, Shanghai, China
Phone: +86-21-5330-4888 FAX: +86-21-5423-4677

Shenzhen Branch
Room 804-805, 8 Floor, Tower 2, Ali Center, No.3331
Keyuan South RD(Shenzhen bay), Nanshan District, Shenzhen 518054, China
Phone: +86-10-3299-0588 FAX: +86-10-3299-0560

Epson Taiwan Technology & Trading Ltd.
15F, No.100, Songren Rd, Sinyi Dist, Taipei City 110. Taiwan
Phone: +886-2-8786-6688

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
10F Posco Tower Yeoksam, Teheranro 134 Gangnam-gu,
Seoul, 06235, Korea
Phone: +82-2-3420-6695

Seiko Epson Corp.
Sales & Marketing Division

Device Sales & Marketing Department
29th Floor, JR Shinjuku Miraina Tower, 4-1-6 Shinjuku,
Shinjuku-ku, Tokyo 160-8801, Japan

Document Code: 412668703
First Issue December 2013
Revised May 2020 in JAPAN