EPSON
EXCEED YOUR VISION

S1C17M30/M31/M32/M34 EEPROM Emulation Library 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

Summary

This reference material describes the S1C17M30/M31/M32/M34 EEPROM emulation library that provides an EEPROM emulation function.

Operating Environment

- PC
  The GNU17 (S5U1C17001C) development tool and the ICDmini USB driver must be installed.
- ICDmini (S5U1C17001H2 or S5U1C17001H3)
  A USB cable is required for connecting with the PC.
- Target system (user target board or our company's evaluation board)
- S1C17xxx EEPROM emulation library package (this package)

Precautions for Usage

The library included in this package is provided as a sample. Our company will not take any responsibility for any problems caused by this library. Please thoroughly verify the operation when using this library for your product.

This material is common to S1C17M30/M31/M32/M34 microcontrollers.
In this material, "xxx" represents an S1C17 model name.
The EEPROM emulation library is provided for each model. For the models that support the EEPROM emulation library, please visit our website.

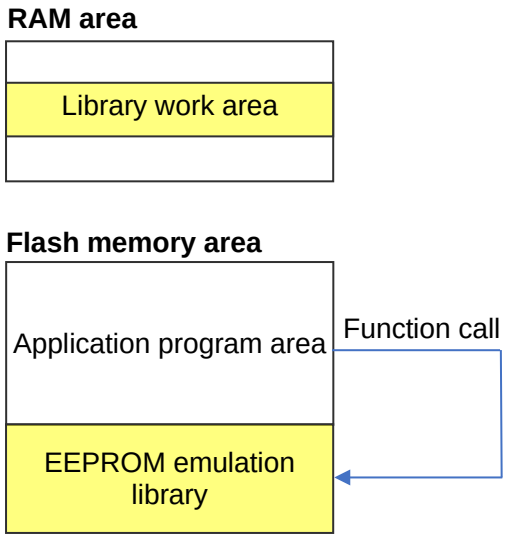
Table of Contents

1. Overview

The EEPROM emulation library package of S1C17M30/M31/M32/M34 provides a library to emulate an EEPROM. Application programs with this library linked can emulate an EEPROM by calling the library functions.

1.1 Features

This library implements the emulation function for a 256-byte EEPROM. The guaranteed number of rewriting the flash memory embedded in the target models supported with this library is about 1,000 times. The emulation function allocates one flash memory sector for one EEPROM address, this makes it possible to increase the number of rewriting times of each emulated EEPROM address to 100,000 times or more theoretically.



1.2 Folder Configuration

The folders of this package are configured as shown below.

+ s1c17(xxx)eeprom
  + eeprom : EEPROM emulation library
  + s1c17(xxx)eeprom_gnu17v2 : Sample program for GNU17 Ver. 2.x
  + s1c17(xxx)eeprom_gnu17v3 : Sample program for GNU17 Ver. 3.x
  - s1c17(xxx)eeprom_notes_e.txt : Supplementary document (English)
  - s1c17(xxx)eeprom_notes_j.txt : Supplementary document (Japanese)
  - License_e.txt : Software license agreement (English)

1.3 File Configuration

The table below lists the library file configuration.

Table 1 s1c17(xxx)eeprom/eeprom

Filename	Function
dataFlash17(xxx).a	S1C17(xxx) EEPROM emulation library (for running on flash memory)
dataFlash17(xxx)ram.a	S1C17(xxx) EEPROM emulation library (for running on RAM)
FlashControlErase.o	Flash memory erasing function
FlashControlWrite.o	Flash memory writing function
DataFlashConfig.h	EEPROM configuration header file
DataFlashConfig.c	EEPROM configuration source file
DataFlashCommand.h	Function declaration header file
OscControl.h	Clock source control header file
OscControl.c	Clock source control source file

The table below lists the file configuration of sample program.

Table 2 s1c17(xxx)eeprom/s1c17(xxx)eeprom_gnu17vx

File/folder name	Function
eeprom	EEPROM emulation library (folder)
boot.c	Boot program
main.c	Main program

2. How to Use Library

This chapter describes the necessary information and precautions on use of this library, and the sample program with this library used.

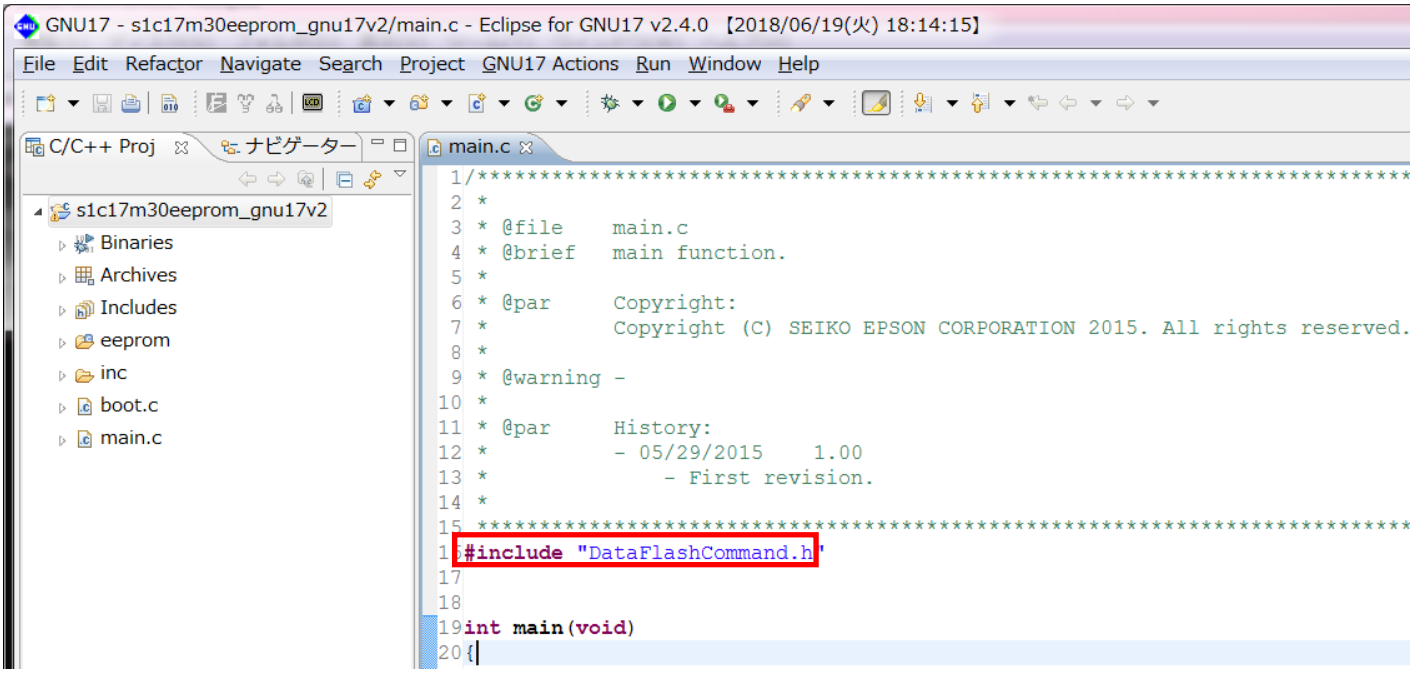
2.1 Settings for Using Library in Application Program

This section describes the items that must be added to and configured in the source file of the application program. For how to incorporate the library into the project of an application program, refer to Appendix A, “How to Incorporate Library into Project.”

1. Header file declaration

Include the header file “DataFlashCommand.h” in the source file that uses this library.

Note: Add the path to the header file if no include path has been defined.



2. Setting EEPROM size and number of write retries

Edit “DataFlashConfig.h” to redefine the constants shown below.
Rewrite the “CONFIG_EEPROM_SIZE_MAX” value with the size of EEPROM to be emulated. The size that can be set in this library is fixed at 256-byte.
Rewrite the “CONFIG_RETRY_COUNT” value with the number of write retries when a writing has failed. Increasing the number of write retries causes the processing time of the writing routine to increase and performance to decrease. So it should only be set to several times.

```
#define CONFIG_EEPROM_SIZE_MAX        (256)
#define CONFIG_RETRY_COUNT            (4)
```

3. Setting clock source

The EEPROM write function in this library changes the CPU clock and T16 Ch.0 configurations so as to optimize the flash memory programming timing. These configurations can be changed by rewriting the "OscClockSourceInitialize()" and "OscClockSourceFinalize()" functions defined in "OscControl.c."
The "OscClockSourceInitialize()" function configures the CPU clock and the T16 Ch.0 for writing data to the EEPROM. When rewriting this function, be sure to note the following points.

- By default, the EEPROM write function uses a 4 MHz clock that is optimum for EEPROM operations.
- When changing the clock configuration, the clock division ratio should be set so that the clock frequency does not exceed 4 MHz.
- Use the same clock source for both the CPU and T16 Ch.0.

The "OscClockSourceFinalize()" function is called at the end of writings. By editing this function, the CPU clock and T16 Ch.0 configurations can be restored to the state before using this library.

```
void OscClockSourceInitialize(void)
{
        /// It doesn't do at all when having already started.
        if(CLGSCLK_CLKSRC != 2)
        {
                /// Disable write-protect.
                MSCPROT = 0x96;
                CLGOSC_OSC3EN = 0;    /// Stop OSC3.
                /// Clear interrupt flag(CLGINTF.OSC3STAIF).
                CLGINTF = 0x0004;
                // OSC3 = Internal
                CLGOSC3_OSC3MD = 0;
                // OSC3 = 4MHz
                CLGOSC3_OSC3FQ = 3;
                // OSC3 enable
                CLGOSC_OSC3EN = 1;
                while(CLGINTF_OSC3STAIF == 0) {
                        asm("nop");    // wait ...
                }
                // Clock = OSC3
                CLGSCLK_CLKSRC = 2;
        }
        // T16 setting
        T16_0CLK = 0x0112;  // T16 Debug mode run, Div = 1/2, Clock = OSC3 4MHz
        T16_0CTL = 1;       // T16 enable
        T16_0MOD = 1;       // One shot mode
}

void OscClockSourceFinalize(void)
{
        return;
        MSCPROT = 0x96;
        /// Clear interrupt flag(CLGINTF.IOSCSTAIF).
        CLGINTF = 0x0001;
        CLGOSC_IOSCEN = 1;          /// Start oscillation.
        // Clock = IOSC
        CLGSCLK_CLKSRC = 0;
        // OSC3 disable
        CLGOSC_OSC3EN = 0;
        // T16 setting
        T16_0CTL = 0;               // T16 disable
        MSCPROT = 0x0;
}
```

4. Adding EERROM read/write functions

Add the EEPROM read/write functions included in this library to the application program source. For detailed information on the functions, refer to Chapter 3, “Library Specifications.”

```
for(i = 0 ; i < CONFIG_EEPROM_SIZE_MAX ; i++)
{
        if(DataFlashWrite(i, i) != DATAFLASH_SUCCESS)
        {
              asm("nop");
        }
        testdata[i] = DataFlashRead(i);
}
//compare
for(i = 0 ; i < CONFIG_EEPROM_SIZE_MAX ; i++)
{
        if(testdata[i] != (i & 0xff))
        {
              asm("nop");
        }
}
```

2.2 Internal RAM and Flash Memory Usage

This library uses an internal RAM area and a flash memory area. For the memory usage in each model, refer to the supplementary document “s1c17(xxx)eeprom_notes_x.txt.”

2.3 Write Time

The write time using this library depends on the EEPROM size configured, number of flash memory rewriting times, the clock source to be used, and other conditions.
For reference, the following shows a write time example when data is written to the same EEPROM address 100,000 times using the S1C17W18 with the internal oscillator OSC3 (4 MHz) as the clock source for the CPU:

Typ. value	7 ms
Max.value	43 ms

The actual write time should be determined using the target system with this library implemented.

2.4 Precautions on Use of Library

When using this library, be sure to note the following points:

- Ch.0 of the 16-bit timer (T16) is exclusively used for this library to control the flash write timing.
- When using this library, please set the CPU operating clock to 4MHz or less.
- The same operating clock should be used for the CPU and T16 Ch.0.
- When using this library, connect a capacitor to the Vpp pin as shown in the basic external connection diagram in the “S1C17(xxx) Technical Manual”, and disconnect the connection between the FLASH_VCC_OUT pin of ICDmini and the Vpp pin of the MCU.
- Be aware of the number of flash memory rewriting guarantee times when using this library. For the flash memory specifications, refer to the “S1C17(xxx) Technical Manual”.
- While executing the write function, supply the V_{DD} operating voltage for Flash programing (When V_{PP} is generated internally) specified in the data sheet of each model. If the voltage falls outside the range, the written value is not guaranteed.

2.5 Sample Program

1. Sample program specification

The sample program performs the operation shown below using this library.

- Writes data, which starts from 0 and is incremented by 1 in each address, to the address range from 0 to <CONFIG_EEPROM_SIZE_MAX – 1> and then verifies the data written.

2. Preparation

Follow the procedure shown below to run the sample program on IDE. Also keep the descriptions under Sections 2.1 to 2.4 in mind when using the library.

(1) Importing project
Launch IDE and import the sample program.

(2) Building
Build the sample program using IDE.

(3) Connecting
Connect ICDmini and the target system to the PC.

(4) Unprotecting flash
When debugging the sample program in an IC with protected flash, it must changed to unprotected.

(5) Loading program
Load the program into IDE.

(6) Executing
Run the program by resetting the target system or other method.

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

3. Operation overview

(1) Initializes the EEPROM address to 0 and the write data to 0.

(2) Writes data by calling the EEPROM write function (DataFlashWrite() in main.c).

(3) Reads data from the address to which data is written in Step (2) (Data Flash Read() in main.c).

(4) Increments the address and write data by 1 and returns to Step (2) if the current address is smaller than CONFIG_EEPROM_SIZE_MAX.

(5) Compares the read data and the write data.

For the DataFlashRead() and DataFlashWrite() functions, refer to Section 3.1, "EEPROM Read/Write Function Details."

3. Library Specifications

3.1 EEPROM Read/Write Function Details

This section describes the functions defined in this library.

EEPROM write function

Format	DataFlashWrite(unsigned short *address*, unsigned char *data*);	
Arguments	unsigned short *address*	EEPROM address
	unsigned char *data*	Write data
Return value	int	Writing result (error code)
Description	This function writes data according to the conditions specified via the arguments. (1) Checks whether the arguments are correct or not. (2) Writes data to the specified address. (3) Returns the error code as the return value.	
Remarks	The effective range of the first argument is 0 to <CONFIG_EEPROM_SIZE_MAX - 1>.	

EEPROM read function

Format	DataFlashRead(unsigned short *address*);	
Argument	unsigned short *address*	EEPROM address
Return value	unsigned char	Read data
Description	This function reads data from the address specified via the argument. (1) Checks whether the argument is correct or not. (2) Reads data from the specified address. (3) Returns the read data as the return value.	
Remarks	The effective range of the argument is 0 to <CONFIG_EEPROM_SIZE_MAX - 1>. 0xff is read from the address in which no data has been written.	

Sequential EEPROM read function

Format	DataFlashReadCurrent(void);	
Argument	–	None
Return value	unsigned char	Read data
Description	This function reads data from the current address. (1) Reads data from the current address. • The address is incremented by 1 after being read. • The address is reset to 0 after data is read from the end address. • If this function is called after the DataFlashWrite() function is executed, data is read from the address specified in the DataFlashWrite() function. • If this function is called after the DataFlashRead() function is executed, data is read from the next address of the address specified in the DataFlashRead() function. (2) Returns the read data as the return value.	
Remarks	The initial current address is 0. 0xff is read from the address in which no data has been written.	

3.2 Error Code Definitions

Table 3 Error Codes

Definition Name	Value	Description
DATAFLASH_SUCCESS	0	The writing has successfully completed.
DATAFLASH_ERROR_ERASE	1	An erase error has occurred.
DATAFLASH_ERROR_WRITE	2	A write error has occurred.
DATAFLASH_ERROR_PARAMETER	3	A parameter error has occurred.

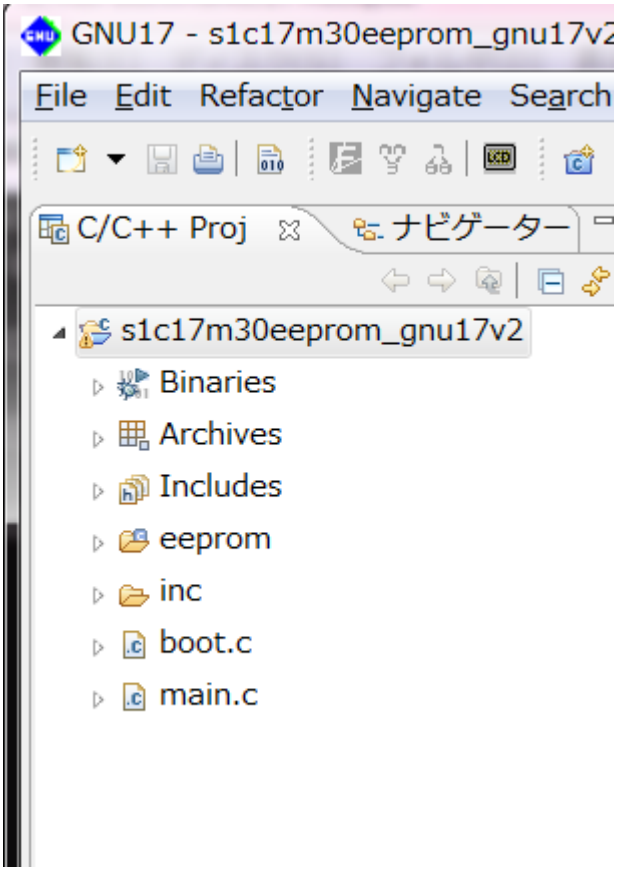
Appendix

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

The following describes how to handle this library with GNU17 Ver. 2.x using S1C17M30 as an example.
For detailed information on usage of GNU17 Ver. 2.x, refer to the “S5U1C17001C Manual (Ver. 2.x.x).”

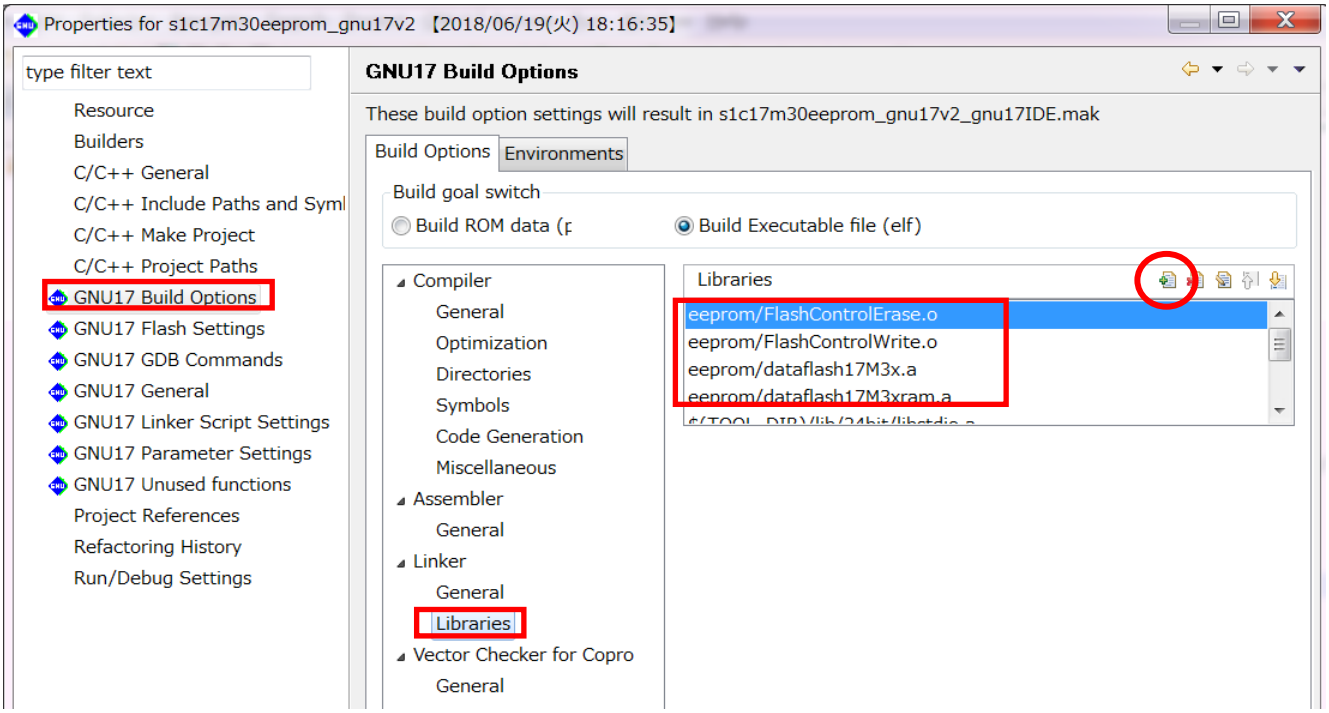
1. Importing library and header files

Import the eeprom folder included in this package into the project folder.



2. Adding libraries

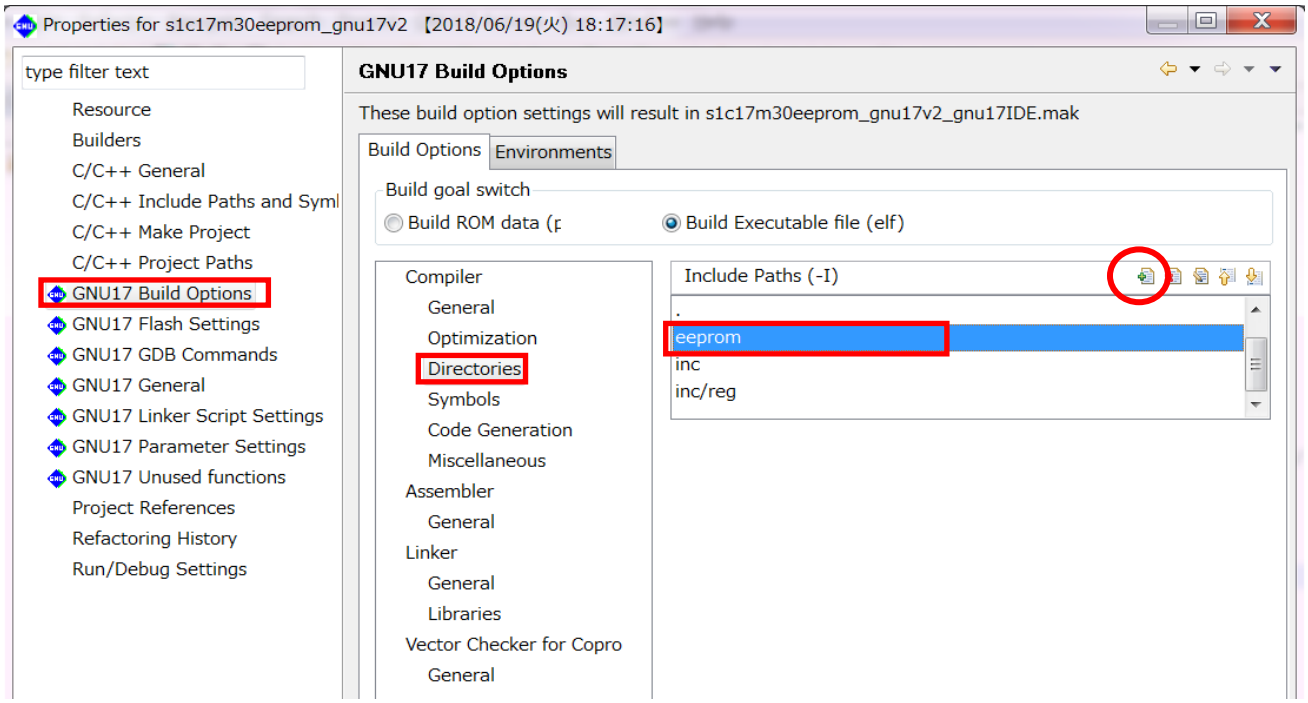
The libraries imported must be added to the library list of the build option before they can be used.
Open the [Properties] dialog box of the project and select [GNU17 Build Options] - [Linker] - [Libraries].
Click the [Add] button (indicated with a red circle in the figure below) and select “dataflash17M3x.a,” “dataflash17M3xram.a,” “FlashControlErase.o,” and “FlashControlWrite.o,” which are included in the eeprom folder, to add them to the library list.



3. Setting include path

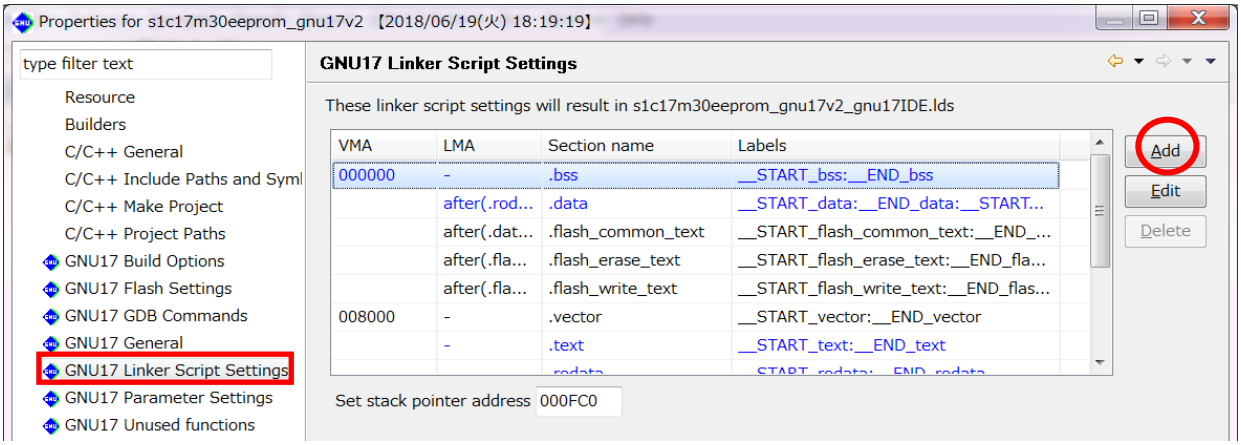
Set the include path to use “DataFlashCommand.h” included in the eeprom folder.
Open the [Properties] dialog box of the project and select [GNU17 Build Options] - [Directories]. Click the [Add] button (indicated with a red circle in the figure below) and select the eeprom folder as an include path.

Note: This setting is not necessary if the include path is directly specified in the source file.



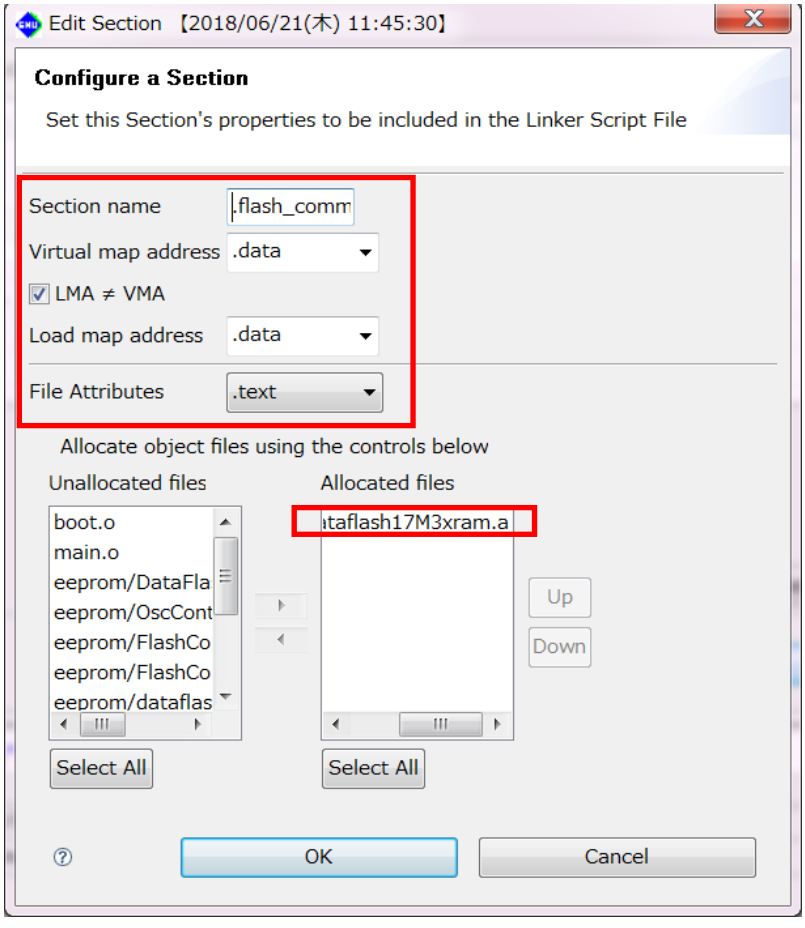
4. Editing linker script

Edit the linker script for the library imported.
Open the [Properties] dialog box of the project and select [GNU17 Linker Script Settings]. Click the [Add] button (indicated with a red circle in the figure below) and add the sections to which the libraries will be placed.

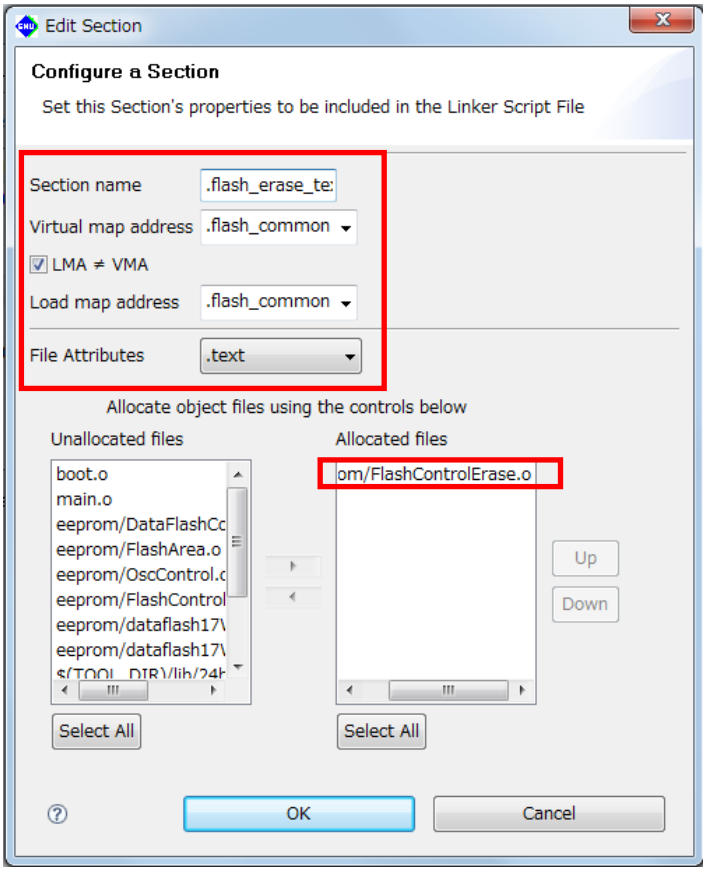


Add the ".flash_common_text," ".flash_erase_text," and ".flash_write_text" sections as below. The section name must begin with a dot (.).

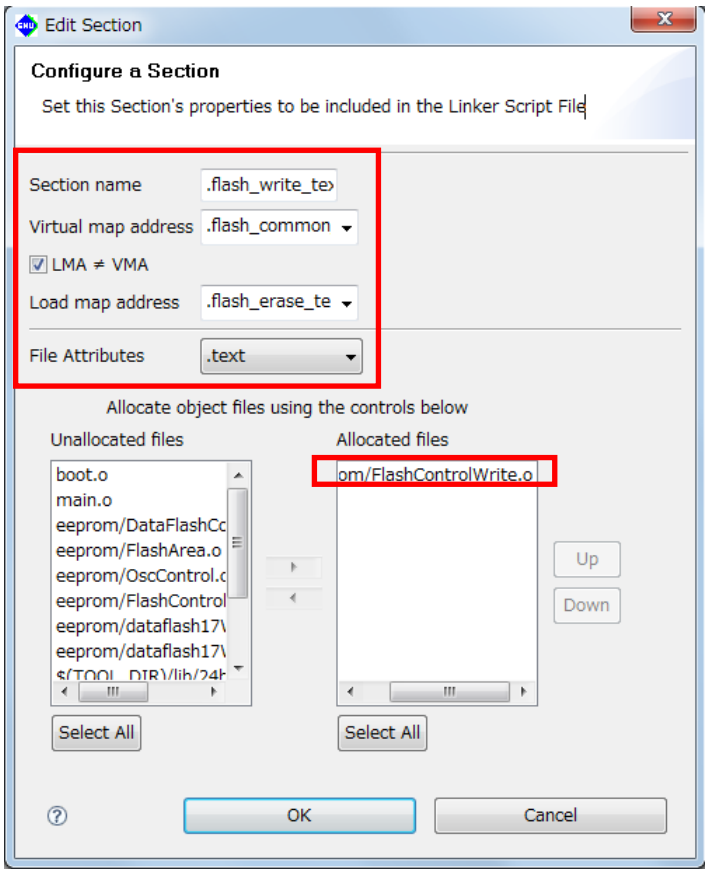
Place "dataflashM3xram.a" in the ".flash_common_text" section.



Place “FlashControlErase.o” in the “.flash_erase_text” section.
Set “.flash_common_text” to VMA and LMA.



Place “FlashControlWrite.o” in the “.flash_write_text” section.
Set “.flash_common_text” and “.flash_erase_text” to VMA and LMA, respectively.



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

The following describes how to handle this library with GNU17 Ver. 3.x.
For detailed information on usage of GNU17 Ver. 3.x, refer to the "S5U1C17001C Manual (Ver. 3.x.x)."

1. Importing library and header files

Import the eeprom folder included in this package into the src folder in the project.

2. Adding libraries

The libraries imported must be added to the library list before they can be used.
Open the [Properties] dialog box of the project and select [C/C++ Build] - [Environment]. Add "dataflash17M3x.a," "dataflash17M3xram.a," "FlashControlErase.o," and "FlashControlWrite.o," which are included in the src/eeprom folder, to the Value of Variable GCC17_USER_LIBS.

```
../src/eeprom/FlashControlErase.o;../src/eeprom/FlashControlWrite.o;
../src/eeprom/dataflash17M3x.a;../src/eeprom/dataflash17M3xram.a
```

3. Setting include path

Set the include path to use "DataFlashCommand.h" included in the eeprom folder.
Open the [Properties] dialog box of the project and select [C/C++ Build] - [Settings] - [Tool Settings] - [Cross GCC Compiler] - [Includes]. Set the include path to the src/eeprom folder.

```
"../src/eeprom"
```

4. Setting linker script

Specify the linker script for the library.
A sample linker script file for the EEPROM emulation library exists in the folder shown below. Copy it to the project folder.

```
/c17(xxx)_sample_gnu17v3/eeprom.x
```

Open the [Properties] dialog box and select [C/C++ Build] - [Settings] - [Tool Settings] - [Cross GCC Linker] - [Miscellaneous]. Enter the option shown below into [Other options] to specify the linker script file copied.

```
-T ../eeprom.x
```

This linker script defines the symbols shown below that are required for the processing of the library and arranges the library execution address in the internal RAM.

```
__START_flash_common_text_lma
__START_flash_erase_text_lma
__START_flash_write_text_lma
```

The script shown below specifies that "FlashControlCommon.o," "FlashControlWrite.o," and "FlashControlErase.o" will not be placed in the RAM.

```
*(EXCLUDE_FILE(*FlashTimeTable*.o*FlashControlCommon.o
               *FlashControlWrite.o*FlashControlErase.o) .text)
```

Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
Rev 1.0	2018/07/01	All	New	New establishment
Rev. 1.1	2018/10/18	6	addition	Added descriptions in Section 2.4, “Precautions on Use of Library.”
Rev. 1.2	2019/11/01	16	Revision	Corrected descriptions in Appendix B, “2 Adding libraries.”
Rev. 1.3	2020/05/26	6	Revision	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

Epson Korea Co.,Ltd
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: 413686203
First Issue July 2018
Revised May 2020 in JAPAN Ⓛ