EPSON S1C31 Family Self-Modifying Library User Manual

EPSON
EXCEED YOUR VISION

S1C31 Family Application Note
S1C31 Family
Self-Modifying Library
Manual

arm

SEIKO EPSON CORPORATION

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 does not assume any responsibility or liability for any kind of damage and/or fire coursed by its 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

Contents

Overview

The S1C31 self-modifying library is a program for rewriting the program code and data in the built-in flash memory of the target model from the application program. By linking this library to an application program and calling a function, the flash memory can be erased and written.
This library and sample software are included in the S1C31xxx peripheral circuit sample software package. The S1C31xxx peripheral circuit sample software package is available on Seiko Epson’s website. In addition to this manual, please also refer to the “S1C31xxx Technical Manual”.

1.1 Working Environment

The following is required when writing and debugging the sample software.

- **Evaluation Board**
  – S5U1C31xxxTx evaluation board with S1C31 series.
  • Debug Probes *1*2
  – IAR Systems I-jet or SEGGER J-Link
- Integrated Development Environment
  – IAR Embedded Workbench for ARM®
  (IAR EWARM) or MDK-ARM®
  (uVision)
- S1C31SetupTool package
  – Includes Flash loader and Configuration files (.svd etc).
- S1C31xxx Peripheral circuit sample software package

*1: Debug probes are not required for library function calls from the sample software.

*2: I-jet is available only with IAR EWARM. J-Link is available for both IAR EWARM and MDK-ARM.

For details on the above, refer to the attached manual.

1.2 Precautions for Usage

The S1C31 self-modifying library and sample software are for reference only. 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 manual is common to the self-modifying library provided for each model of the S1C31 series. About the specifications (Sector information and RAM usage, etc.) that differ depending on the model, refer to the readme included in the S1C31xxx peripheral circuit sample software package.

Library configuration

2.1 Folder Configuration

The configuration of the S1C31 self-modifying library, sample software, and related programs included in the S1C31xxx peripheral circuit sample software package is as follows.

```
S1C31xxxSamplePKG_very_yy.zip
[S1C31xxxSamplePKG_very_yy]
    |- [Licenses]
    |- [Drivers] : Drivers
    |     |- [board] : Evaluation board related driver
    |     |- [CMSIS] : CMSIS driver
    |     |     |- [Device]
    |     |     |     |- [S1C31xxx]
    |     |     |           |- [Include]
    |     |     |           |     |-S1C31xxx.h : CMSIS peripheral circuit access layer header file
    |     |     |           |     |- …
    |     |     |           |- [Source]
    |     |     |                 |- [ARM]
    |     |     |                 |- [IAR]
    |     |     |                 |     |- startup_S1C31xxx.s : CMSIS sartup program
    |     |     |                 |- system_S1C31xxx.c : CMSIS peripheral circuit access layer header program
    |     |     |- [Driver]
    |     |     |     |- [Include]
    |     |     |     |     |- Driver_Common.h : Common driver definition
    |     |     |     |     |- Driver_Flash.h : CMSIS self-modifying library driver definition
    |     |     |     |     |- …
    |     |     |     |- [Source]
    |     |     |- [SVD]
    |     |- [sePeripheralLibrary] : Peripheral circuit library
    |
    |- [Middlewares] : Middlewares
    |     |- [seEepromLibrary] : Self-modifying library
    |     |     |- [Device]
    |     |     |     |- [S1C31xxx]
    |     |     |           |- seFlashLibraryS1C31xxx.a : Library for IAR EAWRM
    |     |     |           |- seFlashLibraryS1C31xxx.lib : Library for MDK-ARM
    |     |     |- flashLibraryForS1c31xxx_readme_e.txt : readme
    |     |     |- flashLibraryForS1c31xxx_readme_j.txt
    |     |- …
    |- [Projects] : Sample softwares
    |     |- [Applications] : Various application software
    |     |     |- [FLASH] : Sample software for Self-modifying library
    |     |     |     |- [ARM] : Project for MDK-ARM
    |     |     |     |- [IAR] : Project for IAR EWARM
    |     |     |     |- main.c
    |     |     |- …
    |     |- …
    |
    README_e.txt
    README_j.txt
```

Figure 2.1.1 S1C31xxx Sample software package configuration

2.2 Library function

The functions provided by this library are defined in Drivers¥CMSIS¥Driver¥Include¥ Driver_Flash.h. The functions provided by this library are as follows.

Table 2.2.1 Functions provided by this library

Function name	Functional overview
int32_t Initialize (ARM_Flash_SignalEvent_t cb_event)	Initialization of this library
int32_t Uninitialize (void)	Restore the settings before initialization of th is library
int32_t EraseSector (uint32_t addr)	Erase built-in flash memory
int32_t ProgramData (uint32_t addr, const void *data, uint32_t c nt)	Write built-in flash memory
int32_t ReadData (uint32_t addr, unsigned char *data, int32_t cn t)	Read built-in flash memory
ARM_DRIVER_VERSION GetVersion (void)	Get this library version
ARM_FLASH_INFO * GetInfo (void)	Get information on built-in flash memory

The verify function is also built into the ProgramData and EraseSector functions.

How to Use Library

1. Explains how to use the S1C31 self-modifying library and sample software.

   3.1 How to Use Library in Application Program

   This section describes how to use this library on the application program. For how to incorporate the library into the project of an application program, refer to “Appendix x. How to Incorporate Library into Project.”

2. **Declaration of Header File**

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

   /* include */

   #include <stdio.h>

   #include <string.h>

   #include “Driver_Flash.h”

3. **Add function**

Add the functions provided by the library to the source file that uses this library. About the function specifications, refer to “Chapter 4 Library Specifications”.

```
extern   ARM_DRIVER_FLASH Driver_Flash;
...
int main(void)
{
  unsigned char compbuf[16];

  //disable Interrupt
  __asm("CPSID i");

  ARM_FLASH_INFO *Info = Driver_Flash.GetInfo();

  Driver_Flash.GetVersion();

  //Initialize
  Driver_Flash.Initialize(NULL);

  //Erase at 0x1D000
  if (Driver_Flash.EraseSector(0x1D000)==ARM_DRIVER_OK)
  {
    printf("Erase: OK¥n");
    //Write at 0x1D000
    if (Driver_Flash.ProgramData(0x1D000, updateLineBit,16)==ARM_DRIVER_OK)
    {
      printf("Program: OK¥n");
      //Read at 0x1D000
      Driver_Flash.ReadData(0x1D000, compbuf, 16);
      if (memcmp(updateLineBit, compbuf, 16) == 0) {
        printf("Verify: OK¥n");
      } else {
        printf("Verify: NG¥n");
      }
    } else {
      printf("Program: NG¥n");
    }


  } else {
    printf("Erase NG¥n");
  }

  printf("Exit¥n");

  //Uninitialize
  Driver_Flash.Uninitialize();

  //enable Interrupt
  __asm("CPSIE i");

    return 0;
}
```

Disable interrupts in peripheral

Library initialization (initialization of peripheral circuits to be used)

Erase built-in flash memory

Write built-in flash memory

Read built-in flash memory

Restore the settings before initialization of this library

Enable interrupts in peripheral circuits

3.2 Internal RAM Usage

This library uses an internal RAM area. About the RAM usage of the self-modifying library of each model, refer to the readme included in the S1C31xxx peripheral circuit sample software package.

3.3 Precautions for Using Library

When using this library, be careful about the followings:

- Disable interrupts before using the functions provided by this library.
- Do not destruct the area where the library is laid out while executing this library.
- When using this library, be aware of the rewritable count of flash memory. For information about flash memory

specification, refer to the corresponding “S1C31xxx Technical Manual”.

- When using this library, stop all peripheral circuits. This library works as follows:
  1. The S1C31D01/S1C31D5x/S1C31W74 uses 16bit timer (T16) ch.0. Therefore, the register of the 16bit timer, ch.0 is changed. Be aware when the application program uses the 16bit timer.
  2. The system clock is changed to a High-Speed clock (OSC3 or IOSC) in using the library. Be aware when a program uses CLG Control Register in using the library.
- About the specifications (Sector information and RAM usage, etc.) that differ depending on the model, refer to the readme included in the S1C31xxx peripheral circuit sample software package.
- When using this library, connect a capacitor to the Vpp pin as shown in the basic external connection diagram in “S1C31xxx Technical Manual”, and disconnect the connection between the Vpp pin and other pins.

3.4 Sample Software

1. **Sample Software Specification**
   In this sample software, this library is used to erase the sector at address 0x1D000 and then write 16 bytes.
2. **Preparation**
   For details on how to execute this sample software project, refer to the “S1C31xxx Peripheral Circuit Sample Software Manual”.
3. Operations Overview
   (1) Disables interrupts in peripheral circuits.
   (2) Get the information on the internal flash memory. (Optional)
   (3) Get the version of this library. (Optional)
   (4) Initialize this library. (Initialization of peripheral circuits used)
   (5) Erase in internal flash memory (0x1D000).
   (6) Write the update data updateLineBit[] (16byte) to internal flash memory (0x1D000).
   The data of 0x1D000 after rewriting is as follows.
   0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00
   (7) Read the internal flash memory (0x1D000).
   (8) Compare the read data cmpbuf [] with the update data updateLineBit [] and display the result.
   (9) Restore the settings before the initialization of this library.
   (10) Enables interrupts in peripheral circuits. (Optional)

Library Specification

4.1 Library Function Details

The details of the functions provided by this libra are described below.

Function Name

int32_t Initialize (ARM_Flash_SignalEvent_t cb_event)

Argument

cb_event	ARM_Flash_SignalEvent_t	Normally set to NULL

Return Value

int32_t	ARM_DRIVER_OK (0)

Function

Initialize the peripheral circuits used in this library.

(1) Change the system clock
(2) Initialize of T16 Ch.0 (S1C31D01/S1C31D5x/S1C31W74 only)

Remarks

Disable interrupts in peripheral circuits before using this function.

Function Name	
int32_t Uninitialize (void)	
Return Value	
int32_t	ARM_DRIVER_OK (0)
Function	
Restore the settings before initialization with the Initialize function. (1) Set T16 Ch.0 (S1C31D01/S1C31D5x/S1C31W74 only) (2) Change the system clock	
Remarks	
If necessary, allow interrupts in peripheral circuits after using this function.	

Function Name		
int32_t EraseSector (uint32_t addr)		
Argument		
addr	uint32_t	Start address of erase sector
Return Value		
int32_t	Erase result (error code)	
Function		
Erase the internal flash memory. (1) Check that the argument is less than or equal to the final address of the internal flash memory. (2) Check if the erased sector has been erased (0xffff). (3) When (2) is not erased, the sector is erased. (4) When erasing is executed in (3), check whether the erase destination sector has been erased (0xffff). (Verify) (5) Returns the erasure result.		
Remarks		

Library Specification

1. Erasing of this function is “one sector” units. To erase multiple sectors, call this function multiple times.
2. Disable interrupts in peripheral circuits before using this function.
3. Specify the start address of the sector in the argument. If you specify an address other than the start address of the sector, a verification error may occur.

4. For sector information, refer to the readme included in the S1C31xxx peripheral circuit sample software package.

Function Name		
int32_t ProgramData (uint32_t addr, const void *data, uint32_t cnt)		
Argument		
addr	uint32_t	Write address.
data	const void *	Write data. Represents a pointer to the written data. The pointer should point to the RAM space.
cnt	uint32_t	Write data size.
Return Value		
uint32_t	Write result (error code)	
Function		
Write the internal flash memory. (1) Check that the argument is less than or equal to the final address of the internal flash memory. (2) Write data to the specified write address. (3) Check if the written address is written data. (Verify) (4) Returns the writing result.		
Remarks		
1. Writing of this function is in “byte (8bit)” units. 2. It is assumed that the writing destination has been erased (0xffff). 3. Disable interrupts in peripheral circuits before using this function.		

Function Name		
int32_t ReadData (uint32_t addr, unsigned char *data, int32_t cnt)		
Argument		
addr	uint32_t	Read address.
data	const void *	Read data. Represents a pointer to the read data. The pointer should poi nt the RAM space.
cnt	uint32_t	Read data size.
Return Value		
uint32_t	ARM_DRIVER_OK (0)	
Function		
Read the internal flash memory. (1) Check that the argument is less than or equal to the final address of the internal flash memory. (2) Read to the specified read address. (3) Returns the reading result.		
Remarks		
1. Reading of this function is in “byte (8bit)” units. 2. Disable interrupts in peripheral circuits before using this function.		

Function Name	
ARM_DRIVER_VERSION GetVersion (void)	
Return Value	
ARM_DRIVER_VERSION	Version of this library
Function	
Get a version of this library	
Remarks	
None.	

Function Name	
ARM_FLASH_INFO * GetInfo (void)	
Return Value	
ARM_FLASH_INFO *	Information of the internal flash
Function	
Get information on the internal flash. • Sector number • Sector size	
Remarks	
None.	

4.2 Error Code Definition
The error code used in the return value of each function is as follows.
Table 4.2.1 Error Code

Definition Name	Value	Description
ARM_DRIVER_OK	0	Successful completion
ARM_DRIVER_ERROR_TIMEOUT	-3	Time out / Verify error
ARM_DRIVER_ERROR_UNSUPPORTED	-4	Unsupported operation
ARM_DRIVER_ERROR_PARAMETER	-5	Argument error

These are defined in “Drivers¥CMSIS¥Driver¥Include¥Driver_Common.h”.

Appendix

A. How to Incorporate Library into Project (IAR EWARM)

The method of incorporating this library into the project of the application program created by IAR EWARM is described below. For more information on IAR EWARM, please refer to the attached manual.

1. **Add Library**

   (1) Select [Project]> [Options] from the IAR EWARM menu.

   (2) Select [Linker] from the [Category] list in the displayed dialog.

   (3) From the [Library] tab, add this library included in the S1C31xxx peripheral circuit sample software package to “Additional libraries”.

   Middlewares¥seFlashLibrary¥Device¥S1C31xxx¥seFlashLibraryS1C31xxx.a

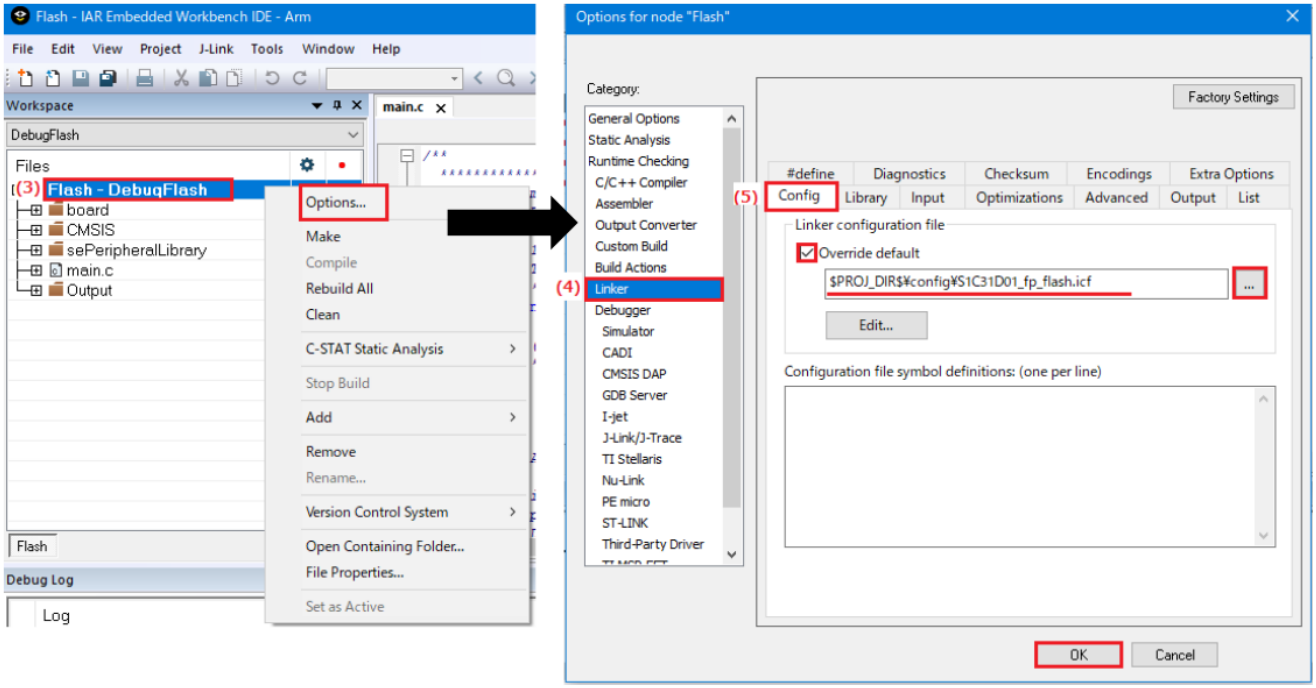


Figure A.3 Set linker script

2. **Add include path**

   (1) Select [Project]> (1) IAR EWARM menu [Project]> [Options] from the IAR EWARM menu.

   (2) Select [C / C++ Compiler] from the [Category] list in the displayed dialog.

   (3) From the [Preprocessor] tab, add the following include path of the driver definition included in the S1C31xxx peripheral circuit sample software package to the “Additional include directory”.

   Drivers¥CMSIS¥Driver¥Include

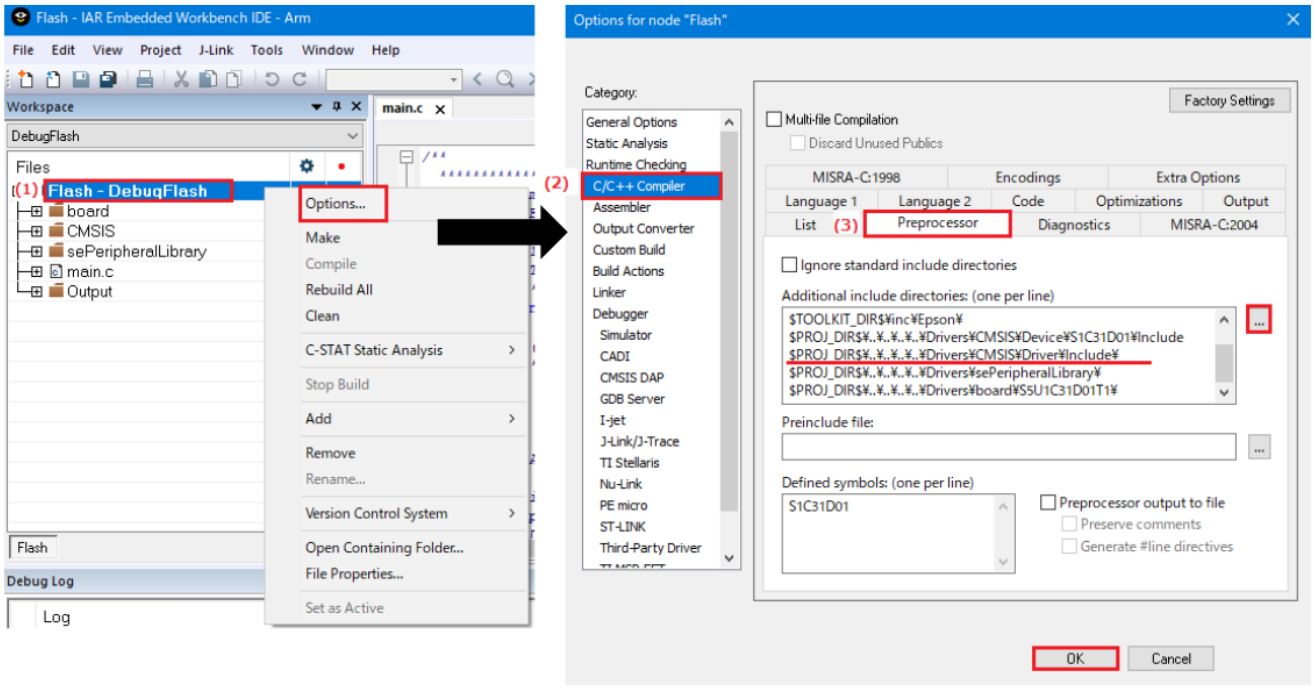


Figure A.2 Add include path

3. **Set linker script**

(1) Edit the linker script file (.icf) included in the project.

(2) S1C31xxx Peripheral circuit sample software package Add the following section by referring to the sample software linker script file (S1C31xxx_fp_flash.icf) included in the package.

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
…
initialize by copy { readwrite };

initialize manually with packing = none { section .flash_common_text};

//initialize by copy with packing = none { section __DLIB_PERTHREAD }; // Required in a multi-threaded application
do not initialize   { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };

place in ROM_region     { readonly };
place in RAM_region     { readwrite,
                          block CSTACK, block HEAP };

place in ROM_region     { section .flash_common_text_init};
place in RAM_region     { section .flash_common_text };
```

Generate flash_common_tex section

Specifying the copy source section of the ROM area

Specifying the copy destination section of the RAM area

Add the above and place the code of this library in the RAM area.

(3) Select [Project]> [Options] from the IAR EWARM menu.

(4) Select [Linker] from the [Category] list in the displayed dialog.

(5) Check “Override default ” from the [Config] tab and specify the edited linker script file.

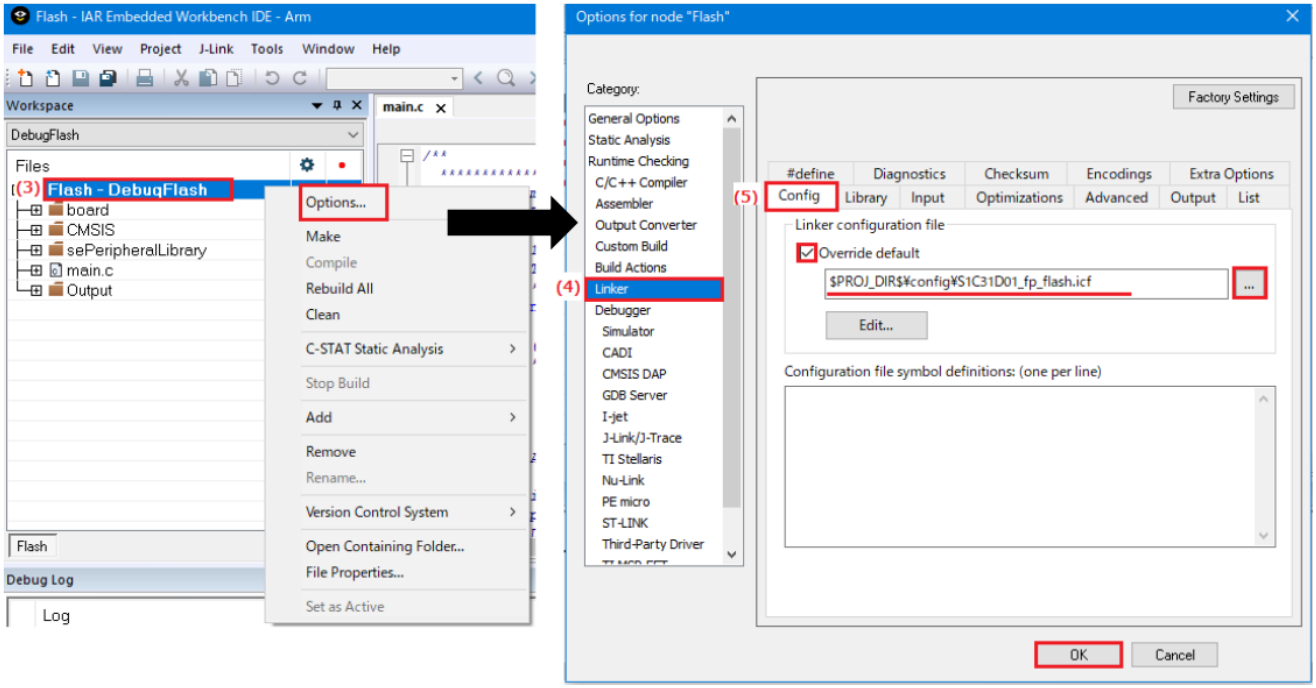


Figure A.3 Set linker script

B. How to Incorporate Library into Project (MDK-ARM)

The method of incorporating this library into the project of the application program created MDK-ARM (uVsion) is described below. For more information on MDK-ARM, please refer to the attached manual.

1. **Add Library**

   (1) Right-click the target source folder from the [Project] window of uVision and select [Add Existing Files to Group 'xxx'...].

   (2) From the displayed dialog, add this library included in the S1C31xxx peripheral circuit sample software package below.

   Middlewares¥seFlashLibrary¥Device¥S1C31xxx¥seFlashLibraryS1C31xxx.lib

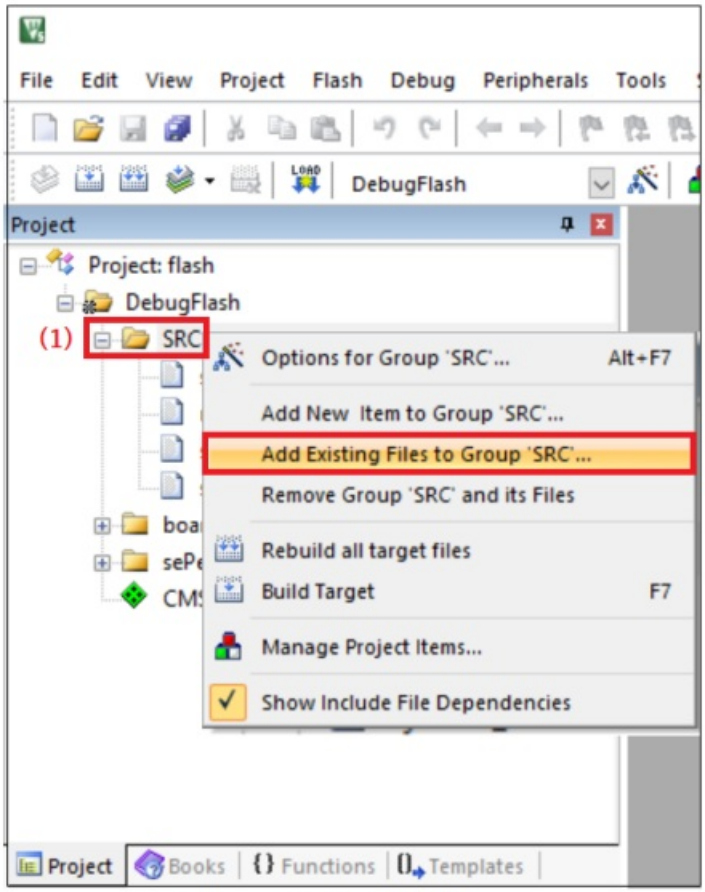


Figure B.1 Add library

2. **Add include path**

   (1) Select [Project]> [Options for Target ‘xxx’…] from the uVision menu.

   (2) Browse to the folder from [C / C ++]> ‘Include Paths’ in the displayed dialog.

   (3) From [New (Insert)], add the following include path of the driver definition included in the S1C31xxx peripheral circuit sample software package.

   Drivers¥CMSIS¥Driver¥Include

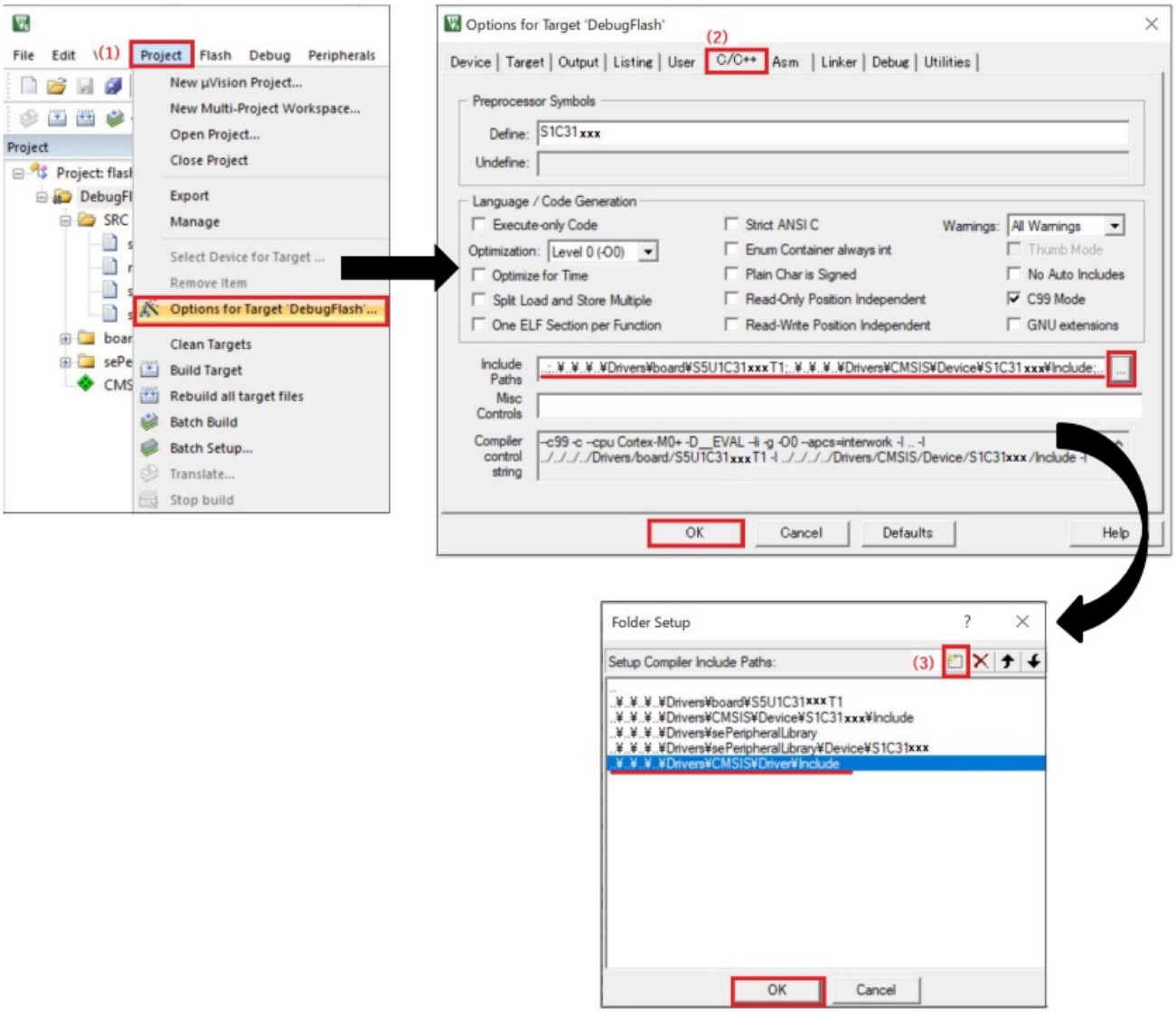


Figure B.2 Add include path

3. **Set linker script**

(1) Edit the linker script file (.sct) included in the project.

(2) Add the following section by referring to the linker script file (flash_flash.sct) of the sample software included in S1C31xxx Peripheral circuit sample software package.

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************
…
  RW_IRAM1 0x00150000 {    ; RW data
    .ANY (+RW +ZI)
  }
  RW_IRAM2 +0 {    ; RW data
    *(.flash_common_text)
  }
…
```

Generate flash_common_text section in RAM area

(3) Select [Project]> [Options for Target 'xxx'…] from the uVision menu.

(4) Specify the linker script file edited from [Linker]> 'Scatter File' in the displayed dialog.

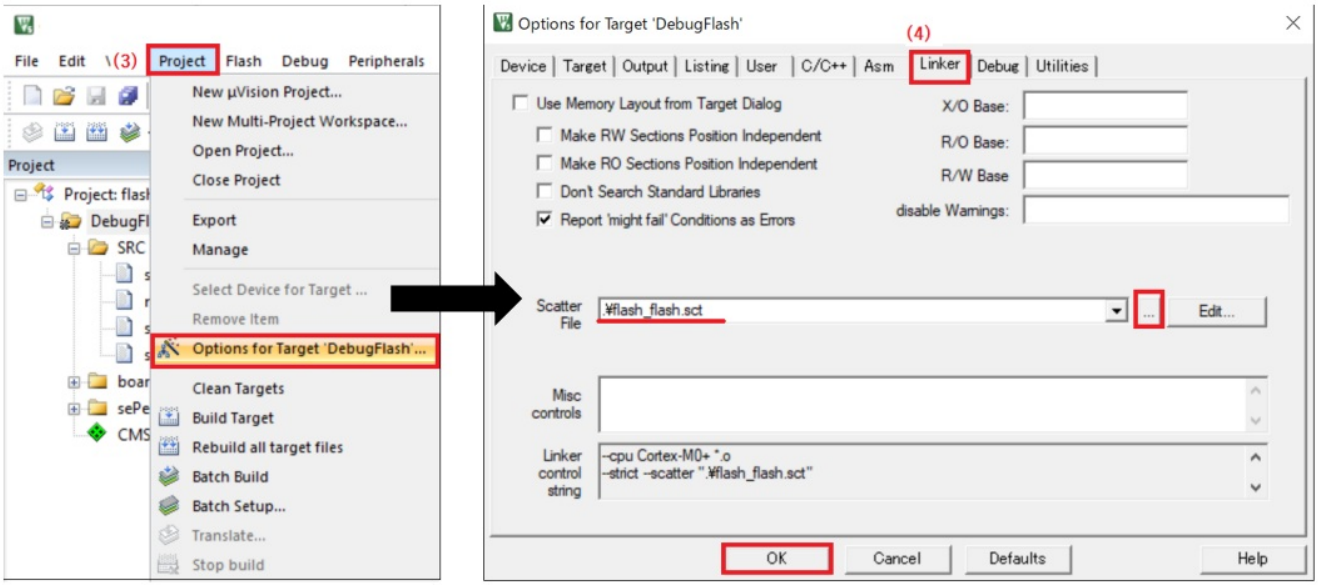


Figure B.3 Set linker script

Revision History

Rev. No.	Date	Page	Category	Contents
Rev 1.0	Apl.30,2021	All	new	New establishment

International Sales Operations

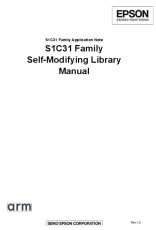
America
Epson America, Inc.
Headquarter:
3131 Katella Ave., Los Alamitos, CA 90720, USA
Phone: +1-562-290-4677San Jose Office:
214 Devcon Drive
San Jose, CA 95112 USA
Phone: +1-800-228-3964 or +1-408-922-0200

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: 414177600
First Issue May 2021 in JAPAN

S1C31 Series Self-Modifying
Library Manual
(Rev.1.0)

Documents / Resources

	**EPSON S1C31 Family Self-Modifying Library** [pdf] User Manual S1C31 Family Self-Modifying Library