External Flash Definition Editor

Creating a Custom Program

REJ06J0098-0100
Rev.1.00
Apr. 16, 2010

Summary

This application note provides a summary of the procedure for creating a custom program that is specifiable in the External Flash Definition Editor (EFE).

Table of Contents

[Abbreviations]

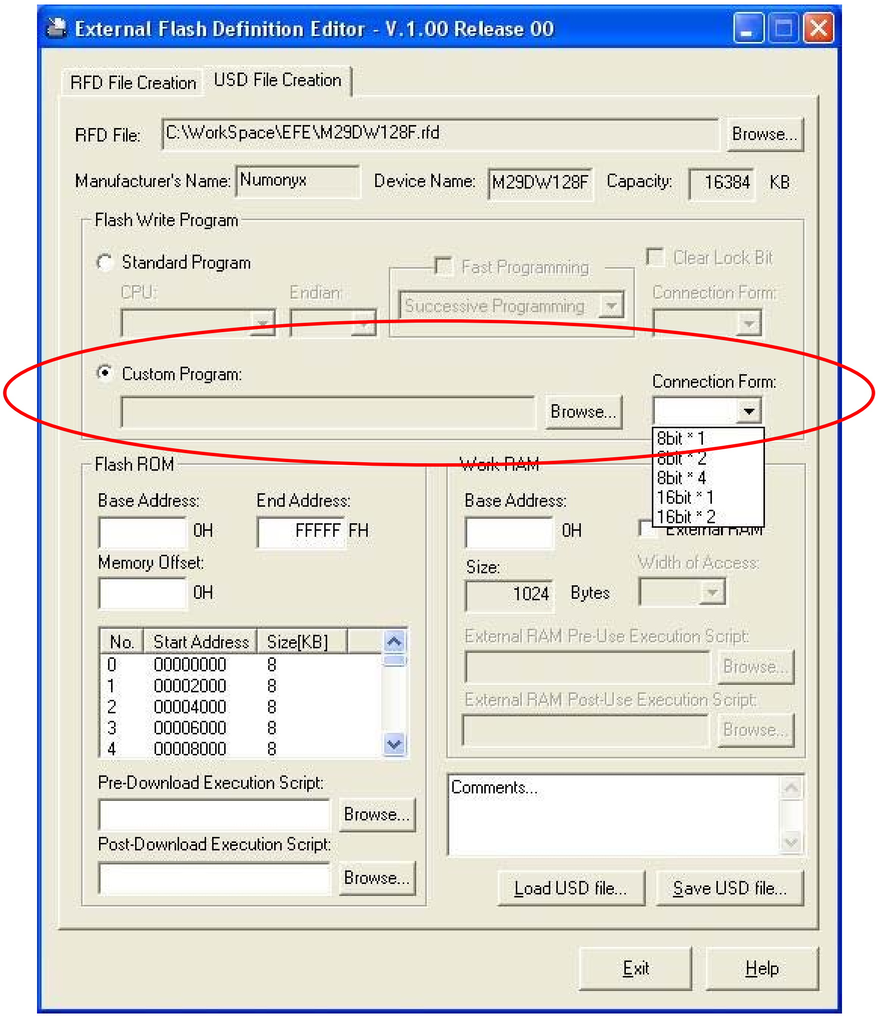
This application note uses the following abbreviations.

Abbreviation	Description
EFE	External Flash Definition Editor
RFD tab	RFD file generation tab of the EFE
USD tab	USD file generation tab of the EFE
RFD file	Renesas Flash Definition file (flash memory device definition file)
USD file	User System Definition file (user system definition file)
External flash memory	Flash memory device connecting to the external bus of the MCU
External flash download facility	Facility for downloading data into the external flash memory
Write program	Program for processing a write to the external flash memory
Standard program	Standard write program preinstalled in the EFE
Custom program	Nonstandard write program specifiable in the EFE
JEDEC method	Flash write method based on JEDEC standard commands
CUI method	Flash write method using the Intel/Sharp CUI commands
Emulator	Emulator system made by Renesas
HEW	High-performance Embedded Workshop, an integrated development environment from Renesas

1. Overview

The External Flash Definition Editor (EFE), assuming a case where your flash memory device has a command set unwritable by the EFE’s standard write program, allows a write program that you’ve programmed yourself to be specified in it as a “custom program.” (See Figure 1-1.)

This application note describes how to create a custom program.



External Flash Definition Editor - V.1.00 Release 00

RFD File: C:\WorkSpace\EFE\M29DW128F.rfd

Manufacturer's Name: Numonyx

Device Name: M29DW128F

Capacity: 16384 KB

Flash Write Program: ○ Standard Program ● Custom Program

Flash ROM

Base Address: 0H

End Address: FFFFF FH

Memory Offset: 0H

No.	Start Address	Size[KB]
0	00000000	8
1	00002000	8
2	00004000	8
3	00006000	8
4	00008000	8

Work RAM

Base Address: 0H

Size: 1024 Bytes

Figure 1-1. Custom Program Select Menu

2. Development Environment

The following shows the environment requirements that need to be met in order to create a custom program.

2.1 Creation Environment

■ Write program sample

Download the file given below from the EFE product website.

http://www.renesas.com/products/tools/emulation_debugging/onchip_debuggers/efe/efe.jsp

1. Sample C source program

   JEDEC method ················EFE_JEDEC_Sample.c

   CUI method ·····················EFE_CUI_Sample.c

2. Header file ·····································EFE_fwif.h

3. Library file ·····································EFE_***.lib (*** denotes the MCU family name)

* The supplied library file differs for each MCU family.

The sample C source program and header file are common to all MCU families.

■ C/C++ compiler and simulator

Please prepare the compiler package suitable for the MCU you're using.

2.2 Debug and Evaluation Environment

■ External Flash Definition Editor (EFE)

Download from the website given below and install in your system.

http://www.renesas.com/products/tools/emulation_debugging/onchip_debuggers/efe/efe.jsp

■ Emulator software (debugger)

Download and install the one that is appropriate for the emulator you're using.

■ Emulator

■ Target system (with external flash memory installed)

3. Flow of Custom Program Creation and the Precautions to Take

3.1 Flow of Custom Program

Follow the procedure described below to create a custom program.

1. Using the sample C source program as a base, customize the functions that require correction.
2. Link the header and library files to build.
3. Create an RFD and a USD file using the EFE.
4. Register the USD file in the emulator software and check with the actual system that downloading to the external flash memory operates normally.

3.2 Precautions to Take

The following shows the precautions to take when creating a custom program.

- Of the supplied samples (sample source program file, header file, and library file), processing of the functions defined in the sample C source program are customizable.
- When terminating execution of functions, always be sure to use the Return_Result function to return the predefined execution status.
- Make sure the memory size occupied by the custom program (not including the stack) is 8,192 bytes (2000h bytes) or less.
- Make sure the stack size used by the custom program does not exceed 256 bytes.

4. Behavior of the Emulator Software

The operation for downloading to the external flash memory is processed by the emulator software and the flash write program in cooperation. The behavior of the emulator software that you need to know before creating a custom program is explained below.

4.1 Work RAM Area

When a request for download to the external flash memory is generated, the emulator software allocates memory in the RAM as a work area in which to place the external flash memory write program.

Note, however, that the original data in the RAM is backed up by the emulator software in advance and then restored after the download process is completed.

The structure of the work area is shown in Figure 4-1.

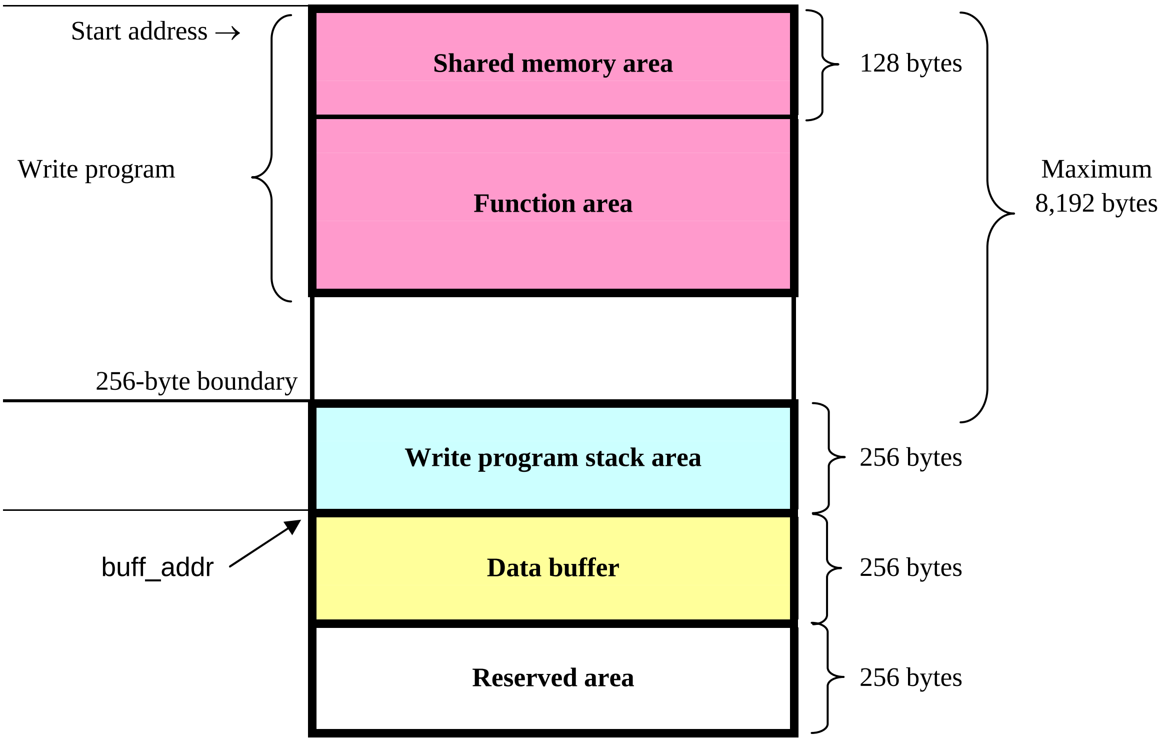


Figure 4-1. Work RAM Area

4.2 Transfer of Parameters

4.2.1 From the Emulator Software to the Write Program

Before executing the write program functions, the emulator software stores the control parameters to be used in a shared memory area of the work RAM.

Then, when the emulator software starts running the write program, the library function named Set_Parameter() is executed, by which the said parameters are transferred from the shared memory area to the global variables of the write program.

4.2.2 From the Write Program to the Emulator Software

Each function of the write program invokes the Return_Result function before they return, to pass their execution status as an argument to it.

The Return_Result function stores the received argument in the shared area.

The emulator software reads data from the shared area to check the execution status.

4.3 Execution Control by the Emulator Software

The control flow in flash memory write processing by the emulator software is shown in Figure 4-2.

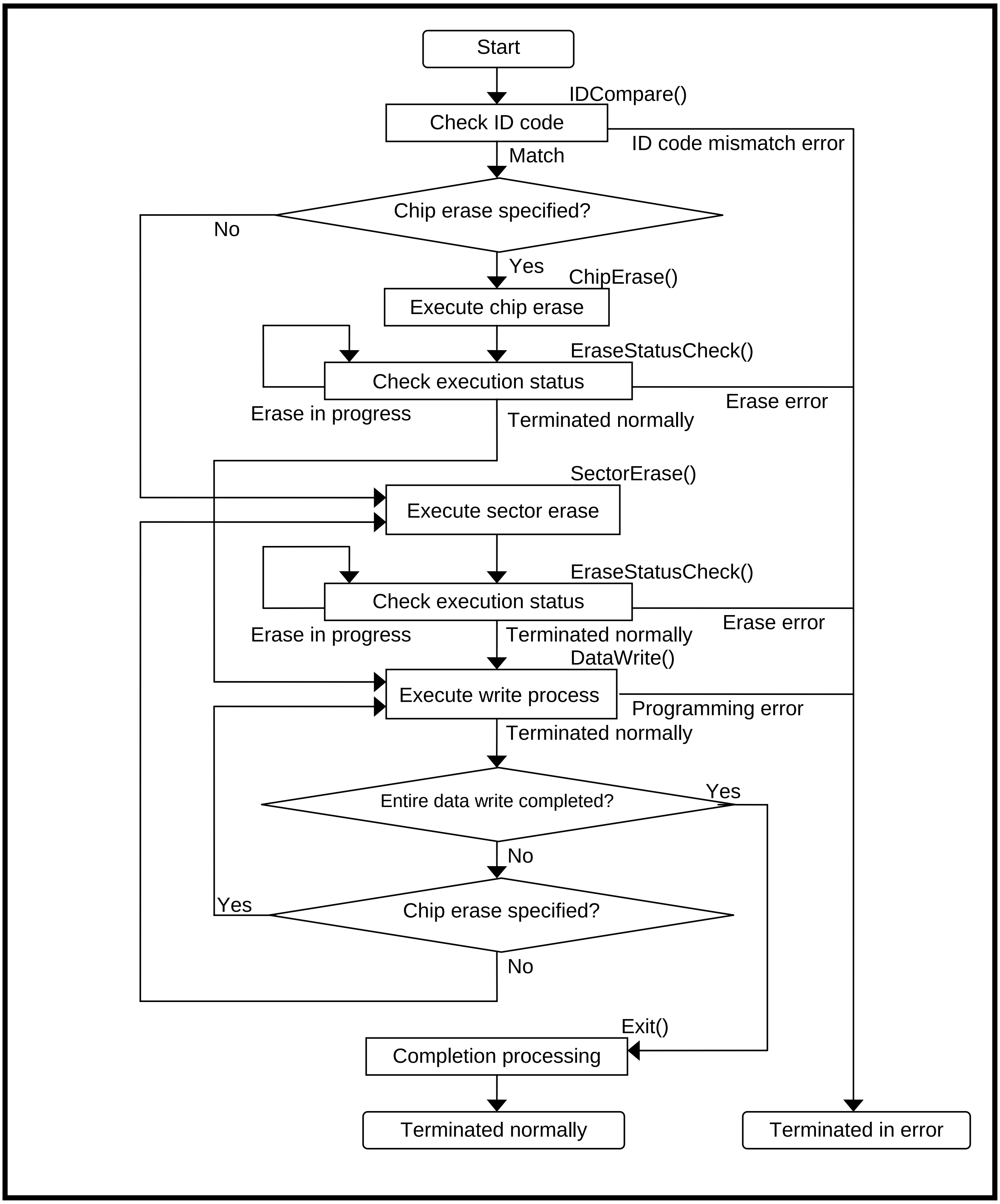


Figure 4-2. Flash Memory Write Control Flow

5. Functions

5.1 Function List

The functions used in a custom program are listed in Table 5-1.

Table 5-1 Function List

No.	Function name		Description
1	void	Return_Result(int)	Returns the execution results of No. 2 thru No. 7.
2	void	IDCompare(void)	Checks the manufacture ID and device ID.
3	void	ChipErase(void)	Issues a command to erase the entire chip.
4	void	SectorErase(void)	Issues a command to erase a sector.
5	void	EraseStatusCheck(void)	Inspects the status register after the erase command is executed.
6	void	Exit(void)	Issues a command to complete the erase operation.
7	void	DataWrite(void)	Issues a programming command.

5.2 Control Parameters

The control parameters used in a custom program are described below.

The control parameters are defined in the header file “EFE_fwif.h” as define statements or global variables.

```
#define TRUE  1
#define FALSE 0

#define BUF_SIZE 0x100    // Data Buffer Size

#define MODE_32x1  0
#define MODE_16x2  1
#define MODE_8x4   2
#define MODE_16x1  3
#define MODE_8x2   4
#define MODE_8x1   5

#define CODE_OK           0x00    // Normal Complete Code
#define CODE_ERASE_EXE    0x00    // Erase Execute Code
#define CODE_ERASE_OK     0x01    // Erase Complete Code
#define CODE_ERASE_ERR    0x02    // Erase Error Code
#define CODE_PROGRAM_ERR  0x02    // Program Error Code
#define CODE_ID_ERR       0x03    // ID Error Code

extern volatile unsigned long*  write_addr;      // Write Address
extern unsigned long  base_addr;  // Sector Address
extern unsigned long  buff_addr;  // Buffer Address
extern unsigned long  wbuf_size;  // Write-Buffer size of Flash Memory
extern unsigned long  connect;    // Connection between MCU and External Flash
Memory
extern unsigned long  maker_id;   // Manufacturer ID
extern unsigned long  device_id;  // Device ID
extern unsigned long  com8bit_0;  // JEDEC 1'st Command at  8bits-bus mode of
Flash Memory
extern unsigned long  com8bit_1;  // JEDEC 2'nd Command at  8bits-bus mode of
Flash Memory
extern unsigned long  com16bit_0; // JEDEC 1'st Command at 16bits-bus mode of
Flash Memory
extern unsigned long  com16bit_1; // JEDEC 2'nd Command at 16bits-bus mode of
Flash Memory
extern unsigned long  com32bit_0; // JEDEC 1'st Command at 32bits-bus mode of
Flash Memory
extern unsigned long  com32bit_1; // JEDEC 2'nd Command at 32bits-bus mode of
Flash Memory

extern void  Return_Result(int);
```

Figure 5-1. EFE_fwif.h

5.2.1 List of Control Parameters (Fixed Values)

Table 5-2 Control Parameters Defined by define (Fixed Values)

No.	Parameter	Value	Classification	Meaning
1	TRUE	1	General purpose	True
2	FALSE	0		False
3	BUF_SIZE	0x100	Data buffer size	Size of the area in which download data is temporarily stored
4	MODE_32x1	0	Form of data bus connection between the MCU and external flash memory	32-bit MCU <-> 32-bit flash memory
5	MODE_16x2	1		32-bit MCU <-> 16-bit flash memory × 2
6	MODE_8x4	2		32-bit MCU <-> 8-bit flash memory × 4
7	MODE_16x1	3		16-bit MCU <-> 16-bit flash memory
8	MODE_8x2	4		16-bit MCU <-> 8-bit flash memory × 2
9	MODE_8x1	5		8-bit MCU <-> 8-bit flash memory
10	CODE_OK	0x00	Execution status of function	Terminated normally
11	CODE_ERASE_EXE	0x00		Erase under execution
12	CODE_ERASE_OK	0x01		Erase terminated normally
13	CODE_ERASE_ERR	0x02		Erase error
14	CODE_PROGRAM_ERR	0x02		Programming error
15	CODE_ID_ERR	0x03		ID code mismatch error

5.2.2 List of Control Parameters (Global Variables)

Table 5-3 Control Parameters Defined as Global Variables

No.	Parameter	Type	Description
1	write_addr	Volatile unsigned long *	Write address
2	base_addr	unsigned long	start address of the target sector
3	buff_addr	unsigned long	start address of the data buffer
4	wbuf_size	unsigned long	Flash memory write buffer size
5	connect	unsigned long	Form of data bus connection between the MCU and external flash memory
6	maker_id	unsigned long	Manufacturer ID of flash memory
7	device_id	unsigned long	Device ID of flash memory
8	com8bit_0	unsigned long	JEDEC command pattern
9	com8bit_1	unsigned long	
10	com16bit_0	unsigned long	
11	com16bit_1	unsigned long	
12	com32bit_0	unsigned long	
13	com32bit_1	unsigned long	

5.2.3 Functional Description of Global Variables

■ write_addr

The write start address of the 256-byte data transferred to the data buffer.

This address is dynamically controlled by the emulator software according to the control flow in Figure 4-2.

■ base_addr

The start address of the target sector to be erased or programmed is passed to the write program.

This address is dynamically controlled by the emulator software according to the control flow in Figure 4-2.

■ buff_addr

The start address of the area in which download data is buffered is passed to the write program.

■ wbuf_size

The control parameter for a buffer-writable type of flash memory.

The size of the flash memory’s internal buffer is passed to the write program.

(This is the “Buffer size” itself that you specify on the RFD tab of the EFE.)

This parameter is used when you’ve specified a buffer write mode.

■ connect

The form of connection between the MCU and flash memory is passed to the write program.

(This is the “Connection form” for the custom program itself that you specify on the RFD tab of the EFE.)

Table 5-4 Connection Forms

Value	Parameter	MCU external bus setting	External flash memory
0	MODE_32x1	32-bit	32-bit × 1
1	MODE_16x2	32-bit	16-bit × 2
2	MODE_8x4	32-bit	8-bit × 4
3	MODE_16x1	16-bit	16-bit × 1
4	MODE_8x2	16-bit	8-bit × 2
5	MOED_8x1	8-bit	8-bit × 1

■ maker_id

The manufacturer ID of the flash memory is passed to the write program.

(This is the “Manufacturer ID” itself that you specify on the RFD tab of the EFE.)

This is used to check the connection with external flash memory.

■ device_id

The device ID of the flash memory is passed to the write program.

(This is the “Device ID” itself that you specify on the RFD tab of the EFE.)

This is used to check the connection with external flash memory.

■ com8bit_0, com8bit_1, com16bit_0, com16bit_1, com32bit_0, and com32bit_1

These parameters are used for JEDEC method-based writes to flash memory.

(This is the “1st Address” and “2nd Address” for each bus width itself that you specify on the RFD tab of the EFE.)

A JEDEC method-based write begins by executing two Unlock cycles first.

In this process, 1st Address is used in the first Unlock cycle and 2nd Address is used in the second Unlock cycle.

The command address actually issued in this process is calculated according to the connection form of the external flash memory as follows:

32-bit × 1:	base_addr + com32bit0/1 << 2
16-bit × 2:	base_addr + com16bit0/1 << 2
16-bit × 1:	base_addr + com16bit0/1 << 1
8-bit × 4:	base_addr + com8bit0/1 << 2
8-bit × 2:	base_addr + com8bit0/1 << 1
8-bit × 1:	base_addr + com8bit_0/1

5.3 Function Description

The functionality of each function used in a custom program is described below.

5.3.1 Execution Status Return Function

[Description]

Informs the execution status of functions 5.3.2 through 5.3.7 to the emulator system via a specific address area.

The function body is included in the library file.

[Function name] **`void Return_Result(int)`**

[Parameters] The argument to this function defined in each function

* If any value other than the argument to this function defined in each function 5.3.2 through 5.3.7 is returned, this function does not behave normally.

[Return values] None

[Behavior]

This function is called by functions 5.3.2 through 5.3.7 at the end of processing giving their execution status as an argument to it.

In this function, the received execution status is stored in a specific address area.

The emulator software reads out the execution status to determine the subsequent control to be exercised.

5.3.2 ID Check Function

[Description]

This function is used to check whether the external flash memory is in a normally accessible state. This function is executed at the beginning of a download process.

[Function name] **`void IDCompare(void)`**

[Parameters] None

[Return values] None

[Argument to the Return_Result function]

`CODE_OK` ······· Terminated normally

`CODE_ID_ERR` ··· ID code mismatch error

* Make sure that no codes other than the above are returned.

[Behavior]

Accesses the external flash memory’s internal registers to read out the manufacturer ID and device ID. The read-out ID value and the expected value (the one supplied on the EFE’s RFD tab) are compared to see if they match.

If the compared values match (i.e., terminated normally), the emulator software determines that there is no problem with connection settings [*1] in accessing the external flash memory and goes to the subsequent processing.

If the compared values do not match (i.e., ID code mismatch error), the emulator software stops executing the subsequent process. Please check the MCU-to-external flash memory connection in hardware, external bus controller settings, MCU clock settings, external bus clock settings, etc. for errors and omissions.

5.3.3 Sector Erase Function

[Description]

This function erases one sector. This function is executed before writing download data to the sector.

[Function name] `void ChipErase(void)`

[Parameters] None

[Return values] None

[Argument to the Return_Result function]

`CODE_OK` ······· Terminated normally

* Make sure that no codes other than the above are returned.

[Behavior]

Issues a sector erase (block erase) command.

No check is made to see if the command is completed. (This check is made by the erase status check function.)

If the lock bit needs to be cleared (to remove protection) prior to an erase, be sure to add a lock bit clearing process before the erase command is issued.

5.3.4 Chip Erase Function

[Description]

This function erases all sectors of the chip.

For the E1/E20 emulator debugger, this function is called when “Write After Erasing All Sectors” is selected on the External Flash Write tab of the External Flash Memory tab of the Configuration Properties dialog (see Figure 5-2).

[Function name] `void ChipErase(void)`

[Parameters] None

[Return values] None

[Argument to the Return_Result function]

`CODE_OK` ······· Terminated normally

* Make sure that no codes other than the above are returned.

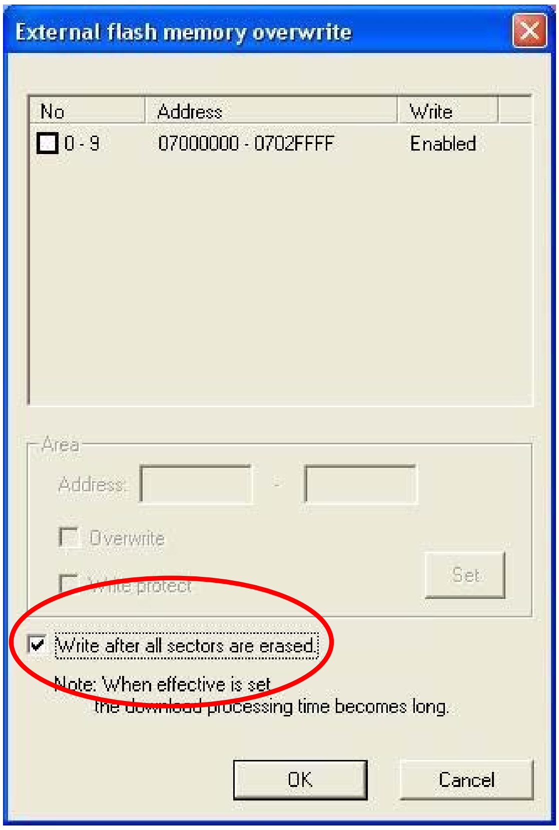
[Behavior]

For the JEDEC method, a chip erase command is issued.

No check is made to see if the command is completed. (This check is made by the erase status check function.)

For the CUI method, because there are no chip erase definitions, a sector erase command is executed on all sector areas repeatedly until the entire chip is erased.

If the lock bit needs to be cleared (to remove protection) prior to an erase, be sure to add a lock bit clearing process before the erase command is issued.



External flash memory overwrite

No	Address	Write
□ 0 - 9	07000000 - 0702FFFF	Enabled

Area

Address: ____ - ____

□ Overwrite

□ Write protect

☑ Write after all sectors are erased.

Note: When effective is set,
the download processing time becomes long.

Figure 5-2. Write After Erasing All Sectors Option

5.3.5 Erase Status Check Function

[Description]

This function checks the processing status of the chip erase and sector erase functions.

This function is executed after the chip erase and sector erase functions.

[Function name] **`void EraseStatusCheck(void)`**

[Parameters] None

[Return values] None

[Argument to the Return_Result function]

`CODE_ERASE_OK` · · · Terminated normally

`CODE_ERASE_EXE` · · Erase under execution

`CODE_ERASE_ERR` · · Erase error

* Make sure that no codes other than the above are returned.

[Behavior]

Inspects the external flash memory’s status register.

From the register’s bit state, one of the following is assumed:

[Terminated normally] [Erase under execution] [Erase error]

If an erase is terminated normally, the emulator software goes to the subsequent processing.

If an erase is under execution, the emulator software reexecutes this function.

If an erase error occurs, the emulator software stops executing the subsequent process.

5.3.6 Sequence Clear Function

[Description]

This function clears a command sequence.

[Function name] **`void Exit(void)`**

[Parameters] None

[Return values] None

[Argument to the Return_Result function]

`CODE_OK` · · · · · · · · · Terminated normally

* Make sure that no codes other than the above are returned.

[Behavior]

This function is used after termination of an erase command in order to clear the command sequence.

5.3.7 Write Control Function

[Description]

This function controls a write to flash memory.

[Function name] **`void DataWrite(void)`**

[Parameters] None

[Return values] None

[Argument to the Return_Result function]

`CODE_OK` ··········· Terminated normally

`CODE_PROGRAM_ERR` ·· Programming error

* Make sure that no codes other than the above are returned.

[Behavior]

This function writes download data from the data buffer to the external flash memory successively.

After a unit of data for write to flash memory, or a write unit [*1], is transferred, the function polls the status register.

If the operation is found to have terminated normally, the function continues with a write of the remaining data.

When a write of a finite amount of data equal to the data buffer size (256 bytes) is completed, the function returns terminated-normally code.

If an error occurs during a write, the function returns error code. In this case, the emulator software stops executing the subsequent process.

*1: A “write unit” refers to the amount of data handled in one write process.

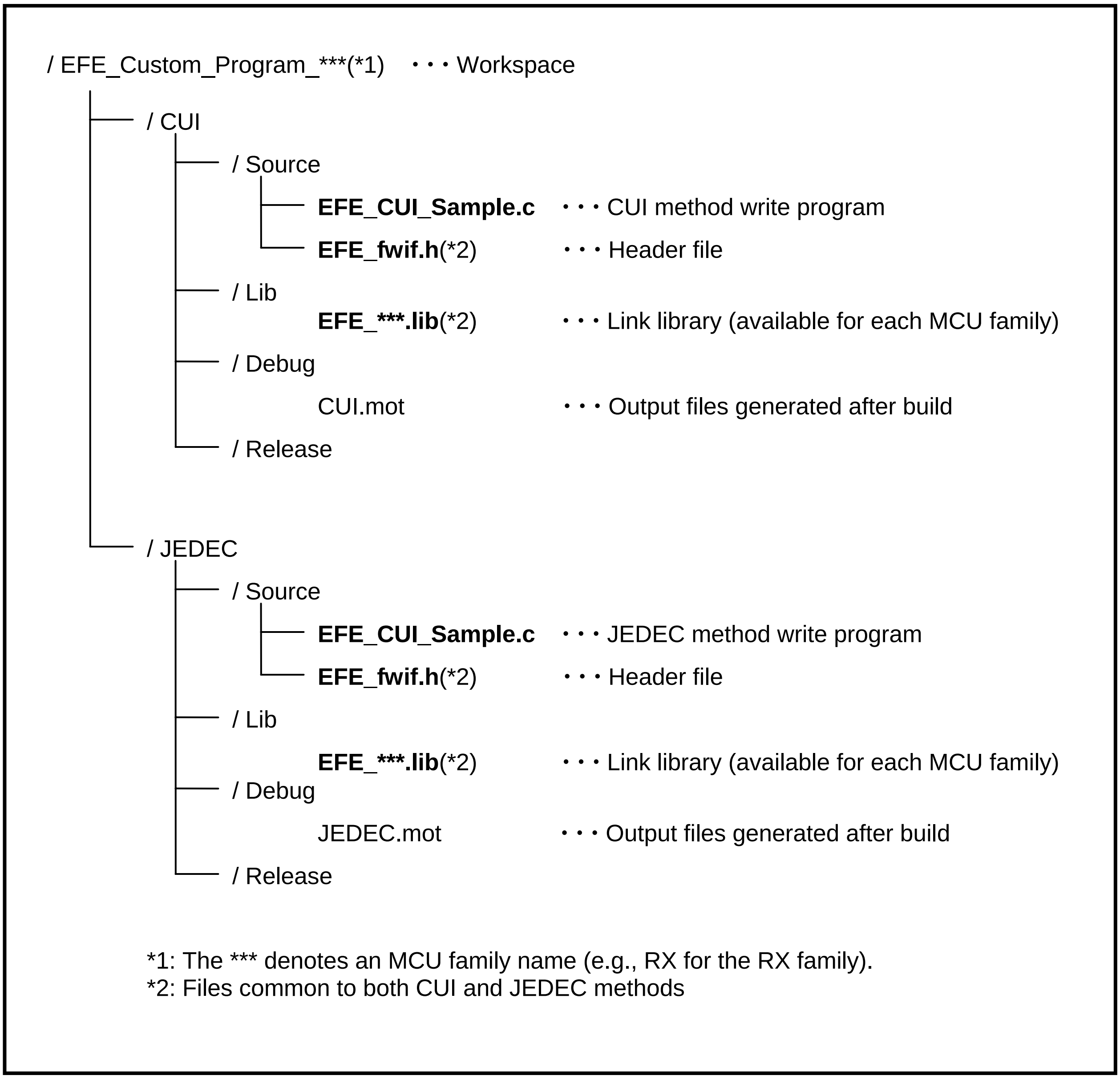
For an ordinary write mode with an 8-bit bus width, for example, the write unit is 1 byte.

Also, for a buffer write mode with a 16-bit bus width and one transfer consisting of 32 words, the write unit is 64 bytes.

6. Composition of the Sample Program

6.1 Folder Structure of the Sample Program

The folder structure of the sample program workspace directory and those under it is shown in Figure 6-1.



```
/ EFE_Custom_Program_***(*1)   ・・・Workspace
    ├── / CUI
    │     ├── / Source
    │     │     ├── EFE_CUI_Sample.c    ・・・CUI method write program
    │     │     └── EFE_fwif.h(*2)      ・・・Header file
    │     ├── / Lib
    │     │           EFE_***.lib(*2)   ・・・Link library (available for each MCU family)
    │     ├── / Debug
    │     │           CUI.mot           ・・・Output files generated after build
    │     └── / Release
    │
    └── / JEDEC
          ├── / Source
          │     ├── EFE_CUI_Sample.c    ・・・JEDEC method write program
          │     └── EFE_fwif.h(*2)      ・・・Header file
          ├── / Lib
          │           EFE_***.lib(*2)   ・・・Link library (available for each MCU family)
          ├── / Debug
          │           JEDEC.mot         ・・・Output files generated after build
          └── / Release
```

*1: The *** denotes an MCU family name (e.g., RX for the RX family).
*2: Files common to both CUI and JEDEC methods

Figure 6-1. Folder Structure of the Sample Program Workspace

6.2 Workspace Window

The HEW workspace window structure of the sample program is shown in Figure 6-2.

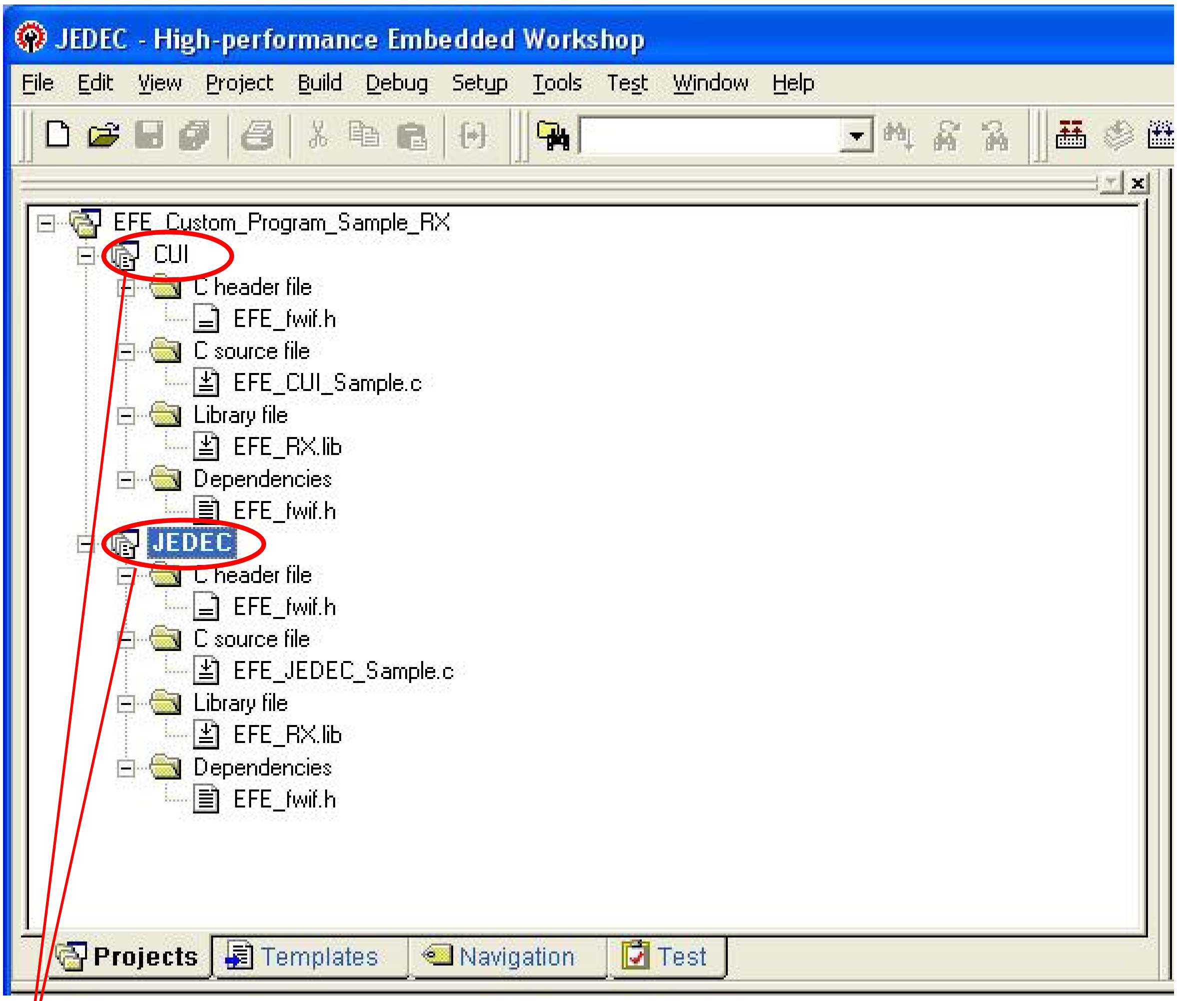


Figure 6-2. Workspace Window

Project

Activate the write method for the custom program to create and execute a build.

(Right-click on target project name and then select “Set To Active Project” before executing a build.)

6.3 Linker Options

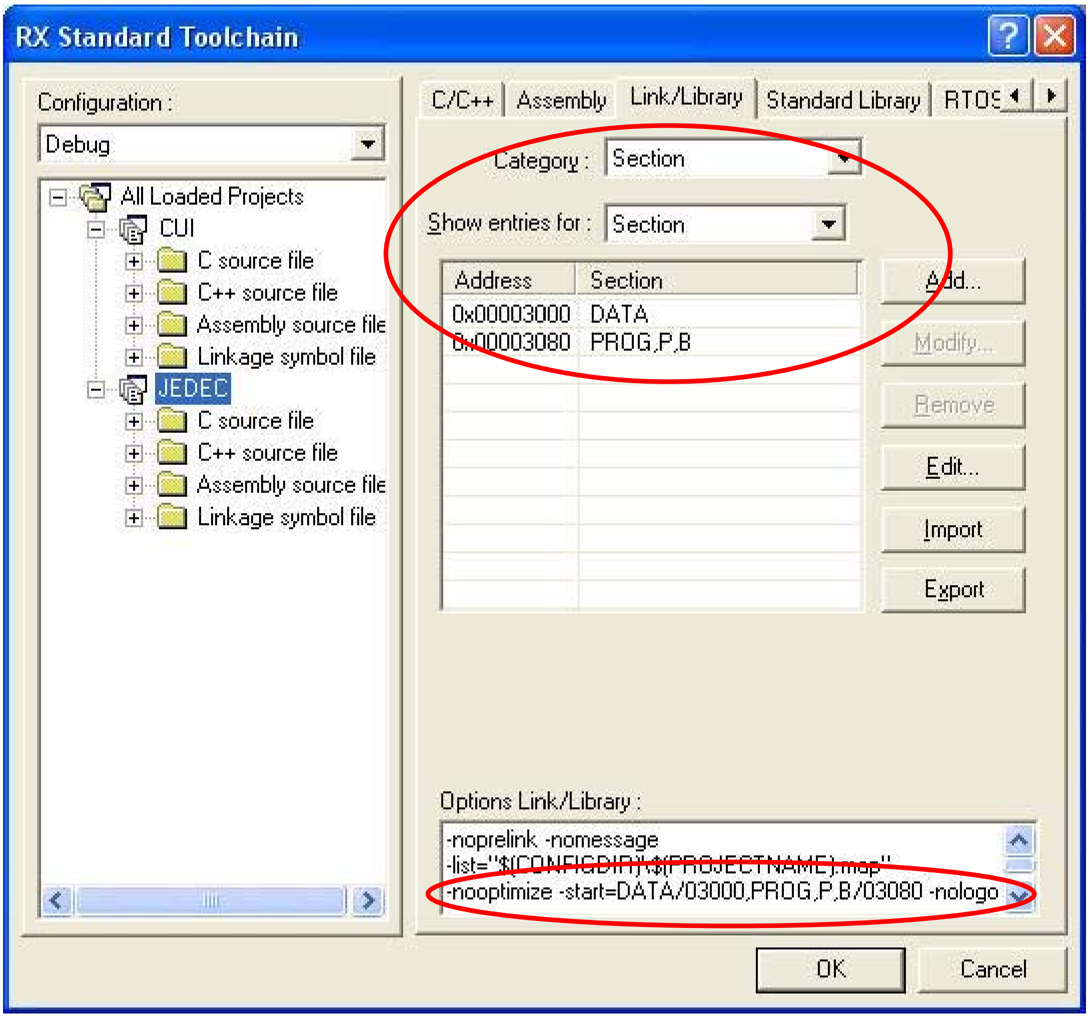
The linker options that are needed when executing a build are explained below.

Note, however, that the sample program workspace has had its options already applied.

■ Section

Specify the option given below.

-start=DATA/03000, PROG,P,B/0x3080



Configuration : Debug

Category : Section

Show entries for : Section

Address	Section
0x00003000	DATA
0x00003080	PROG,P,B

Options Link/Library :

```
-noprelink -nomessage
-list="$(CONFIGDIR)\$(PROJECTNAME).map"
-nooptimize -start=DATA/03000,PROG,P,B/03080 -nologo
```

Figure 6-3. Specifying a Section

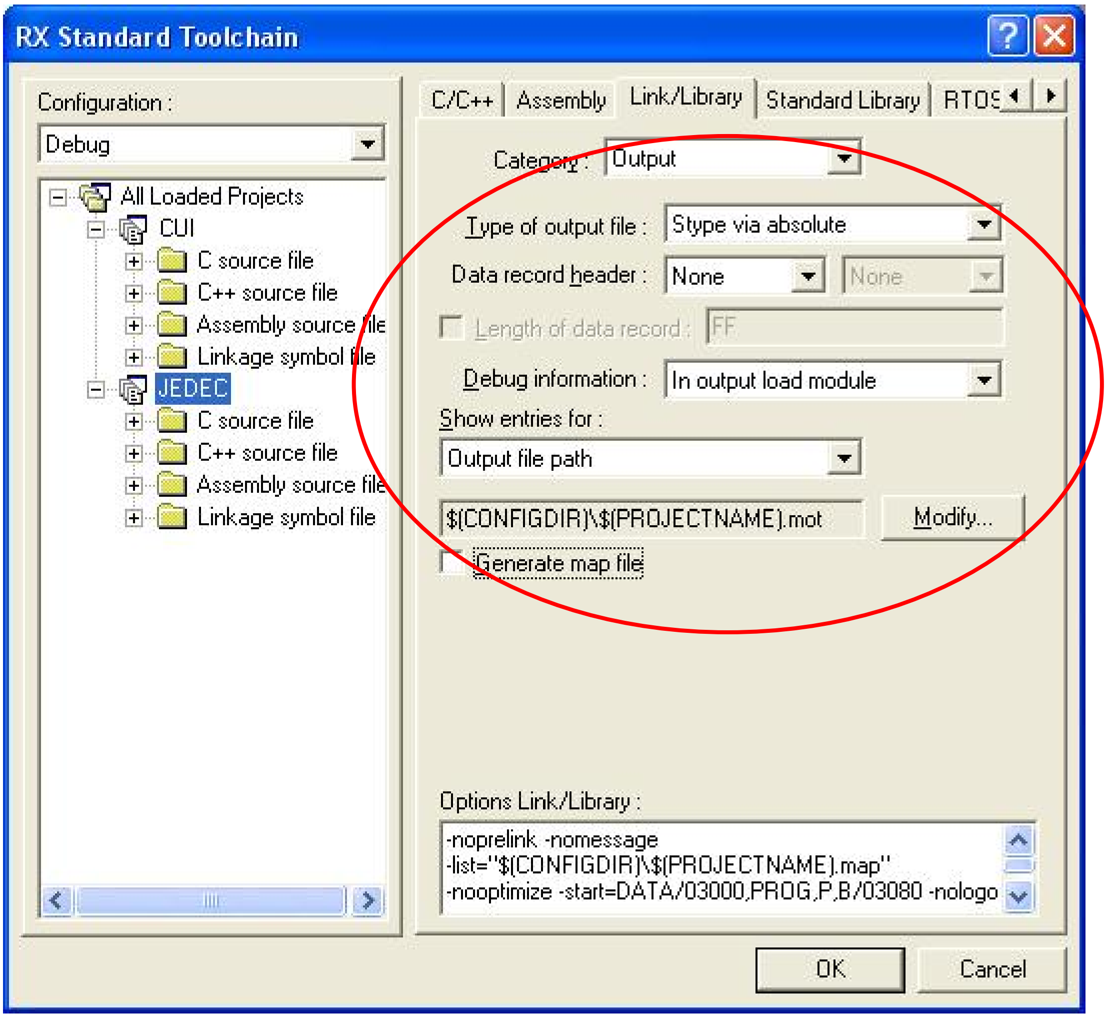
■ Burn-into-ROM support option

Deselect the burn-into-ROM support option.

-rom=・・・・ ← Remove it

■ Output file specification

Specify Motorola S Format as the output file format.



Configuration : Debug

Category : Output

Type of output file : Stype via absolute

Data record header : None None

☐ Length of data record : FF

Debug information : In output load module

Show entries for : Output file path

$(CONFIGDIR)\$(PROJECTNAME).mot

☐ Generate map file

Options Link/Library :

```
-noprelink -nomessage
-list="$(CONFIGDIR)\$(PROJECTNAME).map"
-nooptimize -start=DATA/03000,PROG,P,B/03080 -nologo
```

Figure 6-4. Output File Format

Website and Support

Renesas Electronics Tools Website

http://www.renesas.com/tools

Inquiries

http://www.renesas.com/inquiry

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Apr.16.10	—	First edition issued

All trademarks and registered trademarks are the property of their respective owners.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0