

F²MC-8L/8FX FAMILY

SOFTUNE™ LINKAGE KIT MANUAL

for V3

Support Soft Manual



F²MC-8L/8FX FAMILY
SOFTUNE™ LINKAGE KIT MANUAL
for V3

Support Soft Manual



PREFACE

■ Objectives and Intended Readership

This manual describes the functions and operations of the Spansion SOFTUNE Linkage Kit.

This manual is intended for engineers who are developing application programs using the F²MC-8L/8FX family microprocessor. Please read through this manual.

The linkage kit consists of three kinds of program: linker, librarian and converter.

■ Trademarks

SOFTUNE is a trademark of Spansion LLC.

Microsoft, MS-DOS, and Windows are registered trademarks of Microsoft Corporation in the U.S.A. and other countries.

The company names and brand names herein are the trademarks or registered trademarks of their respective owners.

■ Configuration of this manual

This manual consists of the following four parts and appendix:

PART I LINKAGE KIT

Explains an outline of the tools included in linkage kit and the common items that apply to all tools.

PART II LINKER

Explains the specifications, options, and output lists of a linker.

PART III LIBRARIAN

Explains the specifications, options, and output lists of a librarian.

PART IV OBJECT FORMAT CONVERTERS

Explains the types, options, and functional description of a object format converters and conversion of object format.

APPENDIX

Explains the error messages of the linkage kit, HEX format, S record format, etc.

CONTENTS

PART I	LINKAGE KIT	1
CHAPTER 1	SPECIFICATIONS OF LINKAGE KIT	3
1.1	Outline of Linkage Kit	4
1.2	Startup Procedure	5
1.3	Forced Termination	7
1.4	End Code	8
1.5	Startup Message	9
1.6	End Message	10
1.7	Help Message	11
1.8	Identifiers	12
1.9	Filename Rules	13
1.10	Environment Variables	14
1.10.1	TMP (Work Directory)	15
1.10.2	FELANG	16
1.10.3	FETOOL	17
1.10.4	LIB896	18
1.10.5	OPT896	19
1.10.6	OPT	20
CHAPTER 2	OPTIONS	21
2.1	Option	22
2.2	Numeric Expression of Option Parameters	23
2.3	Notes and Evaluation when Option is Specified	24
2.4	Specifying Options that have Inclusive or Contradictory Relation Each Other	25
2.5	Example of Specifying Command Lines	26
CHAPTER 3	COMMON OPTIONS	27
3.1	List of Common Options	28
3.2	Details of Common Options	29
3.2.1	Specifying Suppression of Default Option File (-Xdof)	30
3.2.2	Specifying Reading from Option Files (-f)	31
3.2.3	Specifying Help Message (-help)	33
3.2.4	Specifying Version Number/Message Output (-V)	34
3.2.5	Suppression of Version Number/Message Output (-XV)	35
3.2.6	Specifying Display of End Message (-cmsg)	36
3.2.7	Specifying Suppression to Display End Message (-Xcmsg)	37
3.2.8	Specifying End Code to 1 when Warning is Issued (-cwno)	38
3.2.9	Specifying End Code to 0 when Warning is Issued (-Xcwno)	39

CHAPTER 4	OPTION FILES	41
4.1	Outline of Option File	42
4.2	Specification to Continue in the Option File	43
4.3	Specifying Comment in the Option File	44
4.4	Example of Describing Option File	45
4.5	Default Option File	46
PARTII	LINKER	49
CHAPTER 5	SPECIFICATIONS OF A LINKER	51
5.1	Overview of a Linker	52
5.2	Functions of a Linker	53
5.2.1	Control on Input-Output Files and Messages	55
5.2.2	Control on Combining and Locating Sections	56
5.2.3	Control on Searching Libraries	57
5.2.4	Setting Entry Addresses and Symbol Values	58
5.3	Types of Sections	59
5.4	Combining Sections	61
5.5	Locating Sections	62
5.5.1	Example of Location when the Order of Combining Sections is not Specified	63
5.5.2	Example of Location when the Order of Combining Sections is Specified	64
5.5.3	Example of Location when the Section Group is Specified	65
5.6	Automatically Locating Sections	66
5.6.1	Automatically Locating Sections when -AL 1 is Specified	67
5.6.2	Automatically Locating Sections when -AL 2 is Specified	70
5.7	Searching Libraries	73
5.7.1	Example of a Search when there is One Library File (1)	74
5.7.2	Example of a Search when there is One Library File (2)	75
5.7.3	Example of a Search when there is One Library File (3)	76
5.7.4	Example of a Search when there are Multiple Library Files (1)	77
5.7.5	Example of a Search when there are Multiple Library Files (2)	78
5.7.6	Processing when Library Files are Individually Specified	79
5.8	ROM and RAM Areas	80
5.9	Sections to be Transferred from ROM to RAM	81
5.10	CPU Information File	84
5.11	Mixing of Objects for a Linker	85
CHAPTER 6	LINKER OPTIONS	87
6.1	List of Linker Options	88
6.2	Details of Linker Options	91
6.2.1	Output Load Module File Name Specification (-o)	92
6.2.2	Debug Information Output Specification (-g)	93
6.2.3	Debug Information Delete Specification (-Xg)	94
6.2.4	Absolute Format Load Module Output Specification (-a)	95
6.2.5	Relative Format Load Module Output Specification (-r)	96
6.2.6	Specification to fill ROM area (-fill)	97
6.2.7	Map List File Name Specification (-m)	100

6.2.8	Map List Output Inhibit Specification (-Xm)	101
6.2.9	Canceling the Omission of Names Displayed in the List (-dt)	102
6.2.10	Output Specification of Memory Used Information List (-mmi)	103
6.2.11	Specification of the Number of Digits in the List Line (-pw)	104
6.2.12	Specification of the Number of Lines on One List Page (-pl)	105
6.2.13	Checksum specification of ROM area (-cs)	106
6.2.14	Warning Message Output Level Specification (-w)	113
6.2.15	ROM Area Specification (-ro)	114
6.2.16	RAM Area Specification (-ra)	115
6.2.17	Section Allocation Order/Address Specification (-sc)	116
6.2.18	Section Group Specification (-gr)	119
6.2.19	Register Bank Area Specification (-rg)	120
6.2.20	Automatic Allocation Specification (-AL)	122
6.2.21	Retrieval Library File Specification (-l)	124
6.2.22	Library Retrieval Path Specification (-L)	125
6.2.23	Library Specification for Each Symbol (-el)	126
6.2.24	Library Retrieval Inhibit Specification (-nl)	127
6.2.25	Default Library Retrieval Inhibit Specification (-nd)	128
6.2.26	Entry Address Specification (-e)	129
6.2.27	Dummy Setting of External Symbol Values (-df)	130
6.2.28	Target CPU Specification (-cpu)	131
6.2.29	Specifying CPU Information File (-cif)	132
6.2.30	Specification for inhibiting Check for Presence of Debug Data (-NCI0302LIB)	133
6.2.31	Function that Automatically Sets Internal ROM/RAM Areas (-set_rora)	134
6.2.32	Specifies to Prevent the Internal ROM/RAM Area from being Automatically Set (-Xset_rora) ..	135
6.2.33	User-specified-area Check Specification (-check_rora)	136
6.2.34	User-specified-area Check Suppression Specification (-Xcheck_rora)	138
6.2.35	Section-placed-area Check Specification (-check_locate)	139
6.2.36	Section-placed-area Check Suppression Specification (-Xcheck_locate)	143
6.2.37	Specifying Check User-unspecified Section (-check_section)	144
6.2.38	Specifying Inhibit Check for Presence of User-unspecified Section (-Xcheck_section)	146
6.2.39	Object Mixing Check Specification (-objmixchk)	147
6.2.40	Object Mixing Check Inhibit Specification (-Xobjmixchk)	148
6.2.41	Relative Assemble List Input Directory Specification (-alin)	149
6.2.42	Absolute Assemble List Output Directory Specification (-alout)	150
6.2.43	Absolute Assemble List Output Specification (-als)	151
6.2.44	Absolute Assemble List Output Module Specification (-alsf)	152
6.2.45	Absolute Assemble List Output Inhibit Specification (-Xals)	153
6.2.46	ROM/RAM and ARRAY Lists Output Specification (-alr)	154
6.2.47	ROM/RAM and ARRAY Lists Output Module Specification (-alrf)	155
6.2.48	ROM/RAM and ARRAY Lists Output Inhibit Specification (-Xalr)	156
6.2.49	ROM/RAM and ARRAY Lists Symbol and Address Display Position Specification (-na, -an) ...	157
6.2.50	External Symbol Cross-reference Information List Output Specification (-xl)	158
6.2.51	External Symbol Cross-reference Information List File Name Specification (-xlf)	159
6.2.52	External Symbol Cross-reference Information List Output Inhibit Specification (-Xxl)	160
6.2.53	Local Symbol Information List Output Specification (-sl)	161
6.2.54	Local Symbol Information List File Name Specification (-slf)	162

6.2.55	Local Symbol Information List Output Inhibit Specification (-Xsl)	163
6.2.56	Section Detail Map List Output Specification (-ml)	164
6.2.57	Section Detail Map List File Name Specification (-mlf)	165
6.2.58	Section Detail Map List Output Inhibit Specification (-Xml)	166
CHAPTER 7 OUTPUT LIST FILE OF THE LINKER		167
7.1	Types of List Files Output by the Linker	168
7.2	Link List File	169
7.2.1	Control List	170
7.2.2	Map List	172
7.2.3	Memory Used Information List	174
7.2.4	Symbol List	177
7.3	Absolute Assemble List File	178
7.3.1	Header and Information List	180
7.3.2	ROM/RAM and ARRAY Lists	181
7.3.3	Assemble Source List	183
7.3.4	Section Information List	185
7.3.5	Cross-reference List	186
7.4	External Symbol Cross-reference Information List File	187
7.5	Local Symbol Information List File	188
7.6	Section Allocation Detailed Information List File	190
CHAPTER 8 LINKER RESTRICTIONS AND Q&A		193
8.1	Linker Restrictions	194
8.2	Q&A for Using the Linker	195
PARTIII LIBRARIAN		197
CHAPTER 9 SPECIFICATIONS OF LIBRARIAN		199
9.1	Functions of Librarian	200
9.2	Function Types of Librarian	201
9.3	Creating and Editing Library File	202
9.4	Extracting a Module from Library File	204
9.5	Deleting Debugging Information of Library	205
9.6	Checking and Displaying the Contents of Library File	206
9.7	Mixing of Objects for Librarian	207
CHAPTER 10 OPTIONS OF LIBRARIAN		209
10.1	List of Options of Librarian	210
10.2	Details of the Options of Librarian	211
10.2.1	Adding (Registering) a Module (-a)	212
10.2.2	Replacing (Registering) a Module (-r)	213
10.2.3	Deleting a Module (-d)	214
10.2.4	Extracting a Module (-x)	215
10.2.5	Specifying to Output a List File (-m)	216
10.2.6	Specifying not to Output a List File (-Xm)	217
10.2.7	Specifying to Output Detailed Information of a List File (-dt)	218

10.2.8	Specifying the Number of Lines Per Page of a List (-pl)	219
10.2.9	Specifying the Number of Columns Per Line of a List (-pw)	220
10.2.10	Creating a Backup File (-b)	221
10.2.11	Inhibiting the Creation of a Backup File (-Xb)	222
10.2.12	Checking the Contents of a Library File (-c)	223
10.2.13	Optimizing the Contents of a File (-O)	224
10.2.14	Specifying to Output Debugging Information (-g)	225
10.2.15	Specifying not to Output Debugging Information (-Xg)	226
10.2.16	Specifying a Target CPU (-cpu)	227
10.2.17	Specifying CPU Information File (-cif)	228
10.2.18	Object Mixing Check Specification (-objmixchk)	229
10.2.19	Object Mixing Check Inhibit Specification (-Xobjmixchk)	230
CHAPTER 11	LIST FORMATS OF LIBRARIAN	231
11.1	Contents of Information in a List File	232
11.2	List of Module Names	233
11.3	Detailed Information of a Module	234
11.4	External Defined and Reference Symbol Information in a Library	235
CHAPTER 12	RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN	237
12.1	Restrictions on a Librarian	238
12.2	Questions and Answers on Using a Librarian	239
PARTIV	OBJECT FORMAT CONVERTERS	241
CHAPTER 13	SPECIFICATIONS OF AN OBJECT FORMAT CONVERTER	243
13.1	Outline of Object Format Converter	244
13.2	Types of Object Format Converters	246
13.3	Executing an Object Format Converter	248
CHAPTER 14	COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER	249
14.1	List of Option of an Object Format Converter	250
14.2	Changing an Output File Name (-o)	251
14.3	Padding (-p)	253
CHAPTER 15	LOAD MODULE CONVERTER (f2ms, f2hs, f2is, f2es)	255
15.1	Outline of Load Module Converter	256
15.2	List of Options of the Load Module Converter	257
15.3	Details of Load Module Converter Options	258
15.3.1	Output S Format Option (-S1/-S2/-S3)	259
15.3.2	Output HEX Format Option (-I16/-I20/-I32)	260
15.3.3	Specifying to Output Start Address Record (-entry)	261
15.3.4	Specifying not to Output Start Address Record (-Xentry)	262
15.3.5	Specifying to Adjust (-adjust)	263
15.4	f2ms (Converting an Absolute Format Load Module into the S Format)	264
15.5	f2hs (Converting an Absolute Format Load Module into the HEX Format)	265

15.6	f2is (Converting an Absolute Format Load Module into the HEX8 Format), f2es (Converting an Absolute Format Load Module into the HEX16 Format)	266
CHAPTER 16 FORMAT ADJUSTER (m2ms, h2hs)		269
16.1	Overview of the Format Adjuster	270
16.2	List of Options of the Format Adjuster	273
16.3	Details of Options of the Format Adjuster	274
16.3.1	Specifying the Data Length in an Output Record (-len)	275
16.3.2	Specifying the Output Range (-ran)	276
16.3.3	Specifying an Output S Format (-S1/-S2/-S3)	277
16.3.4	Specifying an Output HEX Format (-I16/-I20/-I32)	279
16.3.5	Specifying Changes to the Starting Address (-ST)	280
CHAPTER 17 THE BINARY CONVERTER (m2bs, h2bs)		281
17.1	Outline of Binary Converter	282
17.2	List of Options of the Binary Converter	284
17.3	Details on Options of the Binary Converter	285
17.3.1	Specifying the Output Range (-ran)	286
17.3.2	Specifying the Split Mode (-sp)	287
17.3.3	Specifying the Inhibition of the Split Mode (-Xsp)	288
17.3.4	Specifying to Create a Map List File (-m)	289
17.3.5	Specifying not to Create a Map List File (-Xm)	291
CHAPTER 18 OTHER CONVERTERS		293
18.1	m2is (Converting a S Format File into the HEX8 Format)	294
18.2	m2es (Converting a S Format File into the HEX16 Format)	295
18.3	i2ms (Converting a HEX8 Format File into the S Format)	296
18.4	e2ms (Converting a HEX16 Format File into the S Format)	297
CHAPTER 19 RESTRICTIONS AND QUESTIONS AND ANSWERS ON AN OBJECT FORMAT CONVERTER		299
19.1	Restrictions on an Object Format Converter	300
19.2	Questions and Answers on Using an Object Format Converter	301
APPENDIX		303
APPENDIX A ERROR MESSAGES OF THE LINKAGE KIT		304
APPENDIX B HEX FORMAT		329
B.1	Common Format	330
B.2	Data Record (HEX8/HEX16/HEX32) Type: 00	332
B.3	End Record (HEX8/HEX16/HEX32) Type: 01	333
B.4	Extended Segment Address Record (HEX16/HEX32) Type: 02	334
B.5	Start Segment Address Record (HEX16/HEX32) Type: 03	335
B.6	Extended Linear Address Record (HEX32) Type: 04	336
B.7	Start Linear Address Record (HEX32) Type: 05	337
APPENDIX C S RECORD FORMAT		338
C.1	S0 Type (Header Record)	339
C.2	S1 Type (Data Record: 2-Byte Address)	340

C.3	S2 Type (Data Record: 3-Byte Address)	341
C.4	S3 Type (Data Record: 4-Byte Address)	342
C.5	S5 Type (Record to Manage the Number of Records)	343
C.6	S7 Type (Terminator Record)	344
C.7	S8 Type (Terminator Record)	345
C.8	S9 Type (Terminator Record)	346
APPENDIX D	LIST OF LINKER OPTIONS	347
APPENDIX E	LIST OF LIBRARIAN OPTIONS	350
APPENDIX F	LIST OF COMMANDS AND OPTIONS OF THE OBJECT FORMAT CONVERTER	351
APPENDIX G	SPECIFICATION DIFFERENCES DEPENDING ON THE OS	353
APPENDIX H	IDENTIFICATION METHOD FOR SAME OBJECT WHEN SOFTUNE LANGUAGE TOOL IS TRANSFERRED	354
H.1	Outline of Comparison Procedure	355
H.2	Execution Example	356
H.3	When Data Mismatched	357
APPENDIX I	DIFFERENCE IN SPECIFICATION OF SOFTUNE LINKER (FLNK896S) AND OLD LINKER (LINK96)	359
APPENDIX J	DIFFERENCE IN SPECIFICATION OF SOFTUNE LIBRARIAN (FLIB896S) AND OLD LIBRARIAN (LIB96)	364
APPENDIX K	Major Changes	367
INDEX.....		369

PART I LINKAGE KIT

Provides an outline of the tools included in linkage kit and the common items that apply to all tools.

CHAPTER 1	SPECIFICATIONS OF LINKAGE KIT
CHAPTER 2	OPTIONS
CHAPTER 3	COMMON OPTIONS
CHAPTER 4	OPTION FILES



CHAPTER 1

SPECIFICATIONS OF LINKAGE KIT

This chapter outlines the tools included in the linkage kit, how to start up and terminate, and identifiers.

- 1.1 Outline of Linkage Kit
- 1.2 Startup Procedure
- 1.3 Forced Termination
- 1.4 End Code
- 1.5 Startup Message
- 1.6 End Message
- 1.7 Help Message
- 1.8 Identifiers
- 1.9 Filename Rules
- 1.10 Environment Variables

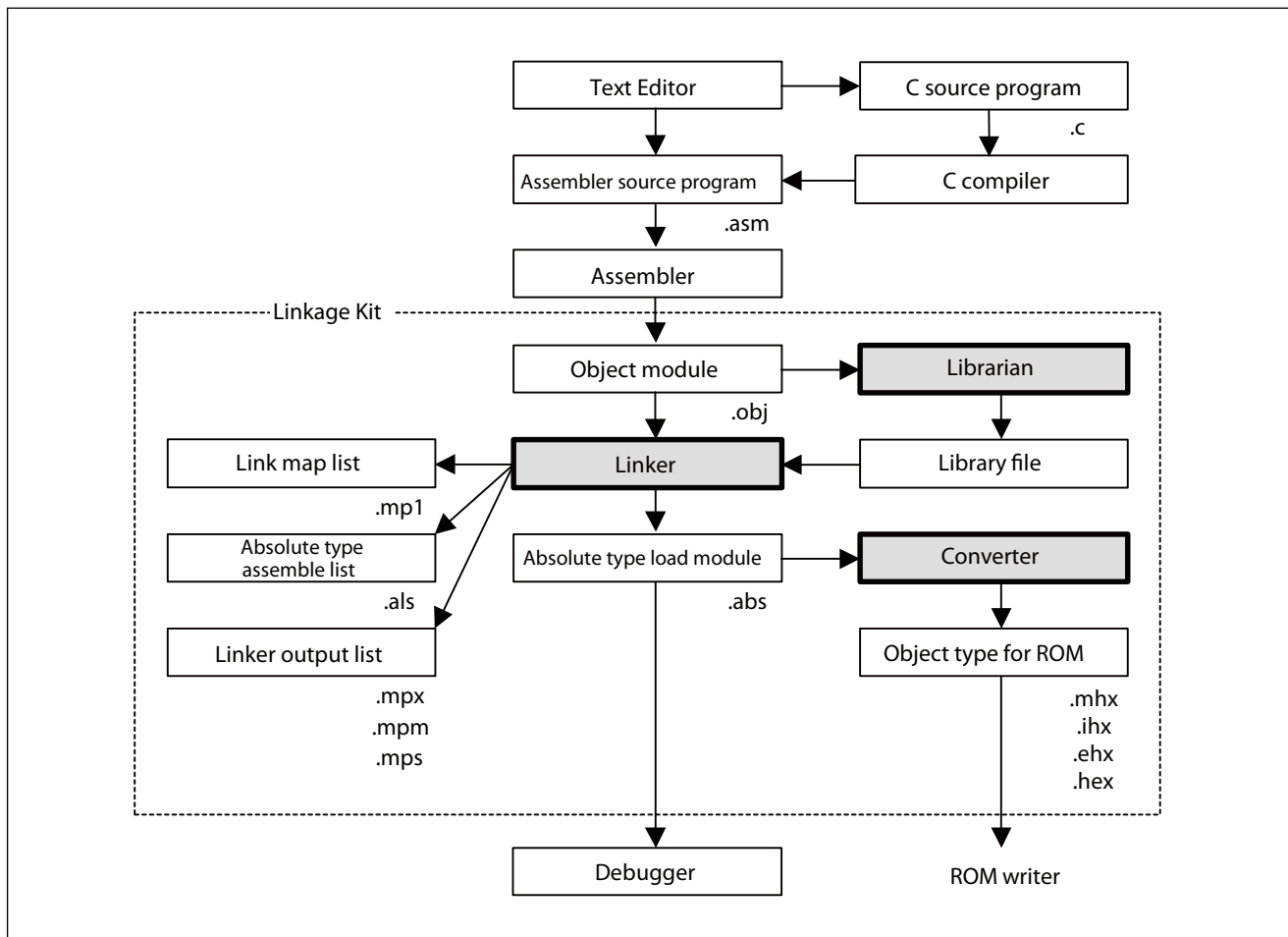
1.1 Outline of Linkage Kit

The linkage kit consists of a linker that is used to connect object modules, a librarian that is used to control object modules and a converter that converts to object type in order to write information on a ROM.

■ Support Range of Linkage Kit

Figure 1.1-1 shows the support range of linkage kit.

Figure 1.1-1 Support Range of Linkage Kit



1.2 Startup Procedure

Command line format and procedure for specification to execute the linkage kit (linker, librarian and object format converter) are described.

■ Command Line Format

To specify the command line (startup command syntax) of the SOFTUNE linkage kit.

- Specify the filename and options as many times as required following the command name.

Below, option is specified after the command name, but the position where option is described can be either before or after the filename. Refer to "CHAPTER 2 OPTIONS" for more details.

● Linker

```
flnk896s [ Option ] ... < Filename >
```

The command name flnk896s differs depending on the target CPU of the tool to be used.

The command name of each target CPU is shown below.

Specify the object module filename to be input to <filename>.

Insert a space to specify two or more filenames.

A wild card such as *.obj can also be used. Expanding the wild card of filename depends on the OS. Refer to "APPENDIX G SPECIFICATION DIFFERENCES DEPENDING ON THE OS".

In linker, the target CPU must be specified using the -cpu option. Be sure to specify the -cpu option when executing the link processing.

● Librarian

```
flib896s [ Option ] ... < Filename >
```

The command name flib896s differs depending on the target CPU of the tool to be used.

Specify the library file that is the target of editing, to <filename>.

In librarian, the target CPU must be specified using the -cpu option. Be sure to specify -cpu option when executing the library processing.

● Object format converter

```
Command name [ Option ] ... < Filename >
```

Determine the object type filename to <filename> based on the functions of each tool. Files of the



following three types are the target format.

- Absolute type load module of linker output
- S format
- HEX format

1.3 Forced Termination

When you want to suspend executing a program in the middle, press the CTRL key and the C key at the same time. (Hereafter referred to as "Press CTRL-C".) Pressing CTRL-C will suspend a program.

■ Forced Termination

When a program processing is suspended by CTRL-C, the output result file cannot be created correctly.
The work file that linkage kit uses during execution is cleared.

1.4 End Code

Each tool of linkage kit returns the end status of its processing to OS as the end code.

■ End Code Value and End Status

Each linkage kit tool returns the end status of the processing (whether the processing has ended normally or an error has occurred) to the OS as the end code. Table 1.4-1 shows the relation between the end codes and end status of processing.

Table 1.4-1 End Codes and End Status of Processing

End code	End status of processing
0	When ended normally or when an error of warning level occurs.
1	When an error of warning level occurs with the -cwno option specified.
2	When an error occurs making it impossible to achieve the correct output result.
3	When a fatal error occurs making it impossible to continue processing.

1.5 Startup Message

Linkage kit shows the startup message with the -V option. In the default processing, the startup message is not displayed.

■ Startup Message and the -V Option

Linkage kit shows a message when errors are detected during processing but does not display a message when starting up in the default processing. If you want a message to be displayed during startup, use the -V option.

When you want to disable the -V option, specify the -XV option after the -V option. Refer to Sections "3.2.4 Specifying Version Number/Message Output (-V)" and "3.2.5 Suppression of Version Number/Message Output (-XV)" for more details.

■ Startup Message

The startup message consists of program name, version number and copyright message.



1.6 End Message

Linkage kit shows end message using the -cmsg option. The end message is not shown in the default processing.

■ End Message and -cmsg Option

Linkage kit shows a message when errors are detected during processing, but no message appears to indicate the end in the default processing. If you want a message to appear at the end of processing, use the -cmsg option.

When you want to disable the -cmsg option, specify the -Xcmsg option after the -cmsg option. Refer to Sections "3.2.6 Specifying Display of End Message (-cmsg)" and "3.2.7 Specifying Suppression to Display End Message (-Xcmsg)".

■ End Message

The end message shows tool names and errors.

Examples of the end message are shown below.

When errors do not occur:

```
Program name COMPLITED FOUND NO ERROR
```

When errors occur:

```
Program name COMPLITED FOUND ERROR
```

1.7 Help Message

The following two kinds of messages are shown as help messages.

- Command line description format
- List of options at startup

■ Help Message

When nothing is specified other than the command name at startup, or when the -help option is specified at startup, program ends while showing the description format of command line and the list of startup options. Refer to Section "3.2.3 Specifying Help Message (-help)" for more details.

● Example of help message

The below figure shows an example of help message in the case of linker (English).

Figure 1.7-1 Example of Help Message in the Case of Linker (English)

```
usage : flnk896s [-option...] object [object...]      *1
-----
options :
----- linker mode option -----
-r      :relocatable linking mode
-a      :absolute linking mode(default)
----- library option -----
-l filename [, ...]      :set library file name      *2
-nl     :not search library file
-nd     :not search default library file
:
:
```

[Description of example]

*1: Command line syntax (startup procedure) is displayed.

*2: List of options and simple description.

This message can be shown in Japanese depending on the setting of the Environment variable FELANG (Refer to Section "1.10.2 FELANG".)

1.8 Identifiers

Linkage kit can handle the following seven kinds of identifiers such as creating program.

- Filename
 - Module name
 - Option name
 - Section name
 - Group name
 - ROM/RAM area name
 - Symbol name
-

■ Types of Characters Consisting of Identifiers

The following characters can be used as identifiers.

- Alphabetical letters
- Numbers
- Underscore (_)

Numbers cannot be used at the top of letters.

As the same, types of characters that can be used for the filename depends on the OS being used. The module name that is created from the filename also depends on the OS being used.

■ Indicating Identifiers

English uppercase and lowercase are indicated.

■ Limiting the Number of Letters for Identifiers

The number of letters for an identifier cannot exceed 255 letters (255 bytes). A character string exceeding this length is rolled off at 255 bytes.

■ Displaying Identifier Name when Outputting List

All identifier names up to 255 characters are not always displayed in the various list files that linkage kit creates.

Some of the longer identifier names only have the top 20 characters or so output and the remaining characters are not displayed.

Number of characters that can be displayed in one line increases or decreases depending on the setting of page width of a list. The format of easy viewing can be selected.

An option is also available to display the identifier name using multiple lines.

1.9 Filename Rules

Filename of the input/output files complies with the limited use of characters that are set for the OS.

There are cases in which the number of characters and code system must be taken into account because the filename is also set in the object modules.

■ Number of Characters for the Filename

The filename of the input/output files complies with the limited use of characters that are set for the OS.

■ Character Code of the Filename

The source filenames of the C language and assembler are not only set as the source filename information in object module, but also set as module names.

The module name can be in English letters, numbers and the underscore symbol (_) only as described in Section "1.8 Identifiers". Therefore, the filenames that use Chinese characters or spaces must be modified specifying module name at the time of assembling.

● Characters that can be used for the filename

(for Windows)

Alphabetical letters, numbers, Japanese kana characters, shift-JIS kanji code and symbols except for the following:

`\ / : ; , * ? " < > |`

When specifying a filename that includes spaces, enclose the filename with double quotations ("").

When specifying a director's name including spaces as the environment variable, do not enclose the filename with double quotations ("").

1.10 Environment Variables

Linkage kit supports the following six kinds of environment variable.

- TMP
- FELANG
- FETOOL
- LIB896
- OPT896
- OPT

■ TMP (Work Directory)

TMP specifies work directory. Refer to Section "1.10.1 TMP (Work Directory)" for more details.

■ FELANG

FELANG selects and specifies the message language. Refer to Section "1.10.2 FELANG" for more details.

■ FETOOL

FETOOL specifies the directory in which the development tool is installed. Refer to Section "1.10.3 FETOOL" for more details.

■ LIB896 (Library File Search Directory)

LIB896 specifies the directory in which library is stored. Refer to Section "1.10.4 LIB896" for more details.

■ OPT896 (Default Option File Storage Directory)

OPT896 specifies directory in which default option files of linker and librarian are stored. Refer to Section "1.10.5 OPT896" for more details.

■ OPT (Default Option File Storage Directory)

OPT specifies a directory in which the default option file of the object tool is stored. Refer to Section "1.10.6 OPT" for more details.

1.10.1 TMP (Work Directory)

TMP (work directory) specifies the work directory that the linkage kit uses during execution.

The section below gives the description format, an explanation and an example of specification.

■ TMP (Work Directory)

[Description format]

SET TMP = < Path name >

[Explanation]

It specifies the work directory that the linkage kit uses during execution.

This environmental variable TMP can also be used in other development tools. (Such as C compiler and assembler)

When the environmental variable TMP is not specified, the current directory is used.

[Example]

```
SET TMP=G:\WORK
```

1.10.2 FELANG

FELANG selects and specifies the message language of help message and error message.

The following section gives the description format, an explanation and an example of FELANG.

■ FELANG

[Description format]

`SET FELANG={ ASCII | EUC | SJIS }`

ASCII: English	ASCII code (default)
EUC : Japanese	EUC code
SJIS : Japanese	SJIS code

[Explanation]

Selects and specifies either English or Japanese (message language) of the help message and error message.

If it is not specified, the English message (specified by ASCII) is selected. When your system does not have Japanese language environment and EUC or SJIS codes, do not specify the FELANG environment variable or specify ASCII.

This environment variable FELANG can also be used in other development tools. (Such as C compiler and assembler)

[Example]

```
SET FELANG=ASCII
```

1.10.3 FETOOL

FETOOL specifies the root directory in which linkage kit is installed.

The following section gives the description format, an explanation and an example of FETOOL.

■ FETOOL

[Description format]

```
SET FETOOL = < Path name >
```

Specify the <path name> including drive name.

[Explanation]

Specify the directory in which linkage kit is installed.

The linkage kit can determine the directory in which message file and library file are installed using the specified directory as the start point. It accesses the files that are necessary for execution.

When it is not specified, the directory in which the executed load module is located becomes the root directory.

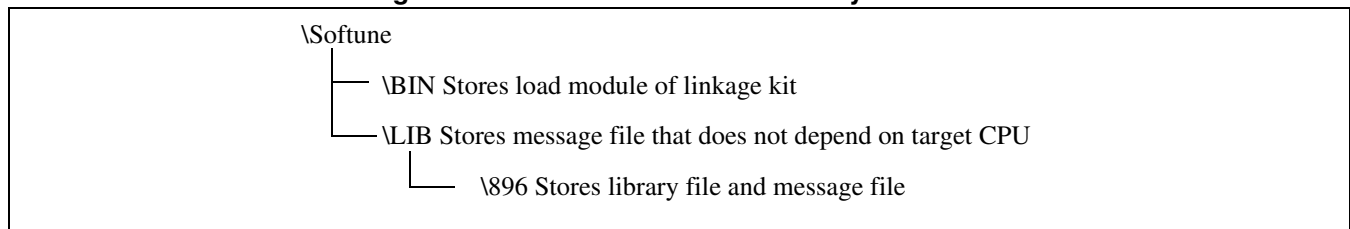
This environment variable FETOOL can also be used in other development tools. (Such as C compiler and assembler)

[Example]

```
SET FETOOL=D:\Softune
```

[Recommended directory structure]

Figure 1.10-1 Recommended Directory Structure



[Supplement]

Linkage kit is created on the premises that the respective files are stored in the directory structure as shown above.

The environment variable FETOOL allows linkage kit to notify the directory path of "Softune".

1.10.4 LIB896

LIB896 (library file search directory) specifies the directory in which the library files that linker searches are located.

The section below gives the description format, an explanation and an example of LIB896.

■ LIB896

[Description format]

```
SET LIB896 = < Path name > [ ; < Path name > ... ]
```

Specify the <path name> including the drive name.

[Explanation]

It specifies the directory in which the library files that linker searches are located.

Specify the directory in which the C-library is stored normally.

When specifying two or more searching paths, separate the <path name> using the following symbol.

- Semicolon (;)

The order in which two or more paths are searched is the same order in which they are specified.

The environment variable LIB896 differs depending on the linker to be used.

[Example]

```
SET LIB896=D:\Softune\LIB\896
```

[Supplement]

When the environment variable FETOOL is specified, the library storage directory of the directory structure, as described in the previous item, is also searched. So the C-library is searched even though the environment variable LIB896 is not set.

The library searching path can be specified by the option-L while executing linker.

When the composite is being specified, the order of the library searching path's priority is:

- 1) The directory that is specified by linker with option-L.
- 2) The directory that is specified by the environment variable LIB896.
- 3) The directory (%FETOOL%\LIB\896) that is directed by the environment variable FETOOL.

If the user created the library himself, specify the paths while taking note of the order of searching with the C-library.

1.10.5 OPT896

OPT896 (default option file storage directory) specifies the directory in which the default option files of linker and librarian are stored.

The description format, an explanation and an example of OPT896 are given below.

■ OPT896

[Description format]

SET OPT896 = < Path name >

Specify the <path name> including the drive name.

[Explanation]

It specifies the directory in which the default option files that linker and librarian uses are stored.

The environment variable OPT896 differs depending on the target CPU of tool to be used.

This environment variable can be omitted.

When it is omitted, the default option files in the development environment directory are referred to.

The default option files in the development environment directory are shown below.

● Linker

%FETOOL%\LIB\896\FLNK896.OPT

● Librarian

%FETOOL%\LIB\896\FLIB896.OPT

[Example]

SET OPT896=D:\Softune\LIB\896

1.10.6 OPT

OPT (default option file storage directory) specifies the directory in which the default option files of object tools are stored.

Description format, explanation and example of OPT are shown below.

■ OPT

[Description format]

```
SET OPT = < Path name >
```

Specify the <path name> including the drive name.

[Explanation]

It specifies the directory in which the default option files that are used by the object tools are stored.

This environment variable can be omitted.

When it is omitted, the default option files in the development environment directory are referred to.

The default option files in the development environment directory are shown below.

● Object tools

- %FETOOL%\LIB\F2M.OPT
- %FETOOL%\LIB\F2H.OPT
- %FETOOL%\LIB\F2I.OPT
- %FETOOL%\LIB\F2E.OPT
- %FETOOL%\LIB\M2I.OPT
- %FETOOL%\LIB\M2E.OPT
- %FETOOL%\LIB\I2M.OPT
- %FETOOL%\LIB\E2M.OPT
- %FETOOL%\LIB\M2B.OPT
- %FETOOL%\LIB\M2M.OPT
- %FETOOL%\LIB\H2H.OPT
- %FETOOL%\LIB\H2B.OPT

[Example]

```
SET OPT=D:\Softtune\LIB
```

CHAPTER 2

OPTIONS

This chapter explains options of the linkage kit.

- 2.1 Option
- 2.2 Numeric Expression of Option Parameters
- 2.3 Notes and Evaluation when Option is Specified
- 2.4 Specifying Options that have Inclusive or Contradictory Relation Each Other
- 2.5 Example of Specifying Command Lines

2.1 Option

An option consists of an option name and parameter. This section gives a synopsis of an option and how to specify an option.

■ Synopsis of Option

The following section is a synopsis of an option.

- Option name [Parameter] ...

Add a hyphen (-) to the top of the option name.

Insert a space to separate the option name from the parameter.

Whether the parameter is used or not used and the format of the parameter is defined in each option. Refer to the description of the respective options.

Pay attention to the following points when specifying an option.

- Capital letters and small letters of alphabetical letters must be distinguished when specifying option name.
- When a parameter needs an option, the parameters cannot be omitted entirely.
- When specifying two or more options, they cannot be specified as a group. For example, -a and -v as -av is not acceptable.
- Spaces cannot be used in between hyphens and the option name.

■ Parameter

Parameters are used to specify a filename or module name, which become the target of operation of an option. Two or more parameters are usually separated using a comma (.). However, symbols other than the comma (,) are also used when specifying sophisticated parameters. Refer to the description of each option for more details.

[Example]

```
-a  
gets.obj,puts.obj,getc.obj,putc.obj  
-sc CODE=0xC1000,DATA=0x0100
```

2.2 Numeric Expression of Option Parameters

Decimal numbers and hexadecimal numbers can be used for the numeric expression of option parameters.

■ Numeric Expression of Option Parameters

When the numeric value of an option parameter starts with (0x), the numeral is recognized as a hexadecimal number. The other numerals are recognized as decimal numbers. Both capital and small letters can be used for a to f of the hexadecimal notation.

[Example]

0x100 ... Hexadecimal notation (= 256)
100 ... Decimal notation (= 0x64)
0xff and 0xFF are the same.

2.3 Notes and Evaluation when Option is Specified

When specifying options, some options need duplicated specification and some need sequence to specify them.

In the linkage kit, the options are evaluated according to rules.

■ Notes and Evaluation when Specified Option

The precautions and rules of evaluation when specifying options are described below.

- Options that require no parameters
Specifying only once is enough. Duplicated specifications have no meaning.

[Example]

-V : Specifying the message output

Duplicated specification like -V -V has no meaning and is error-free.

- Options that require parameters

When duplicated specification is required, there are different methods of evaluation as shown below.

- Only the last specification is effective.
- The order in which the specifications appear has specific intent, and all specifications are effective.
- The order in which the specifications appear is irrelevant, and all specifications are effective.

[ex. Only the last option which is specified is valid]

-o file.abs : Specifying the output file

When options are specified two or more times like -o file.abs -o file.rel, the specification that is entered last becomes effective. (In this case, file.rel becomes effective.)

[ex. Order of specifying options has meaning and all specifications are effective]

-l lib1.lib -l lib2.lib : Specifying the library retrieval (linker)

When options are specified in order, such as -l lib2.lib -l lib1.lib, order of retrieving library is inverted.

[ex. Order of specifying options has no meaning, yet all specifications are effective]

-sc code=0x8000 -sc data=0x200 : Specifying location of sections (linker)

When options are specified in order, such as -sc data=0x200 -sc code=0x8000, all options are effective because the location of sections are individually independent.

2.4 Specifying Options that have Inclusive or Contradictory Relation Each Other

When an option has an inclusive relation with other options, specifying an option of higher order becomes effective. When an option has a contradictory relation with other options, the option that is specified later becomes effective.

■ Example of Specifying an Option that has an Inclusive Relation with Other Options

[Example]

`Xm -pw 80` : Specifying suppression of outputting list and specifying page width

Since the option `-pw` is effective only in specifying output of list, this option itself has no meaning when the option `-Xm` (suppression of outputting list) is specified. These options have no meaning even though the order is inverted, for example `-pw 80 -Xm`.

■ Example of Specifying an Option that has a Contradictory Relation with Other Options

When an option that has a contradictory relation with other options is specified, the option that is specified later becomes effective.

[Example 1]

`a -r`

Specifying absolute format output and specifying relative format output (linker) `-r` becomes effective.

[Example 2]

`-m mapfile -Xm`

Specifying a name of list file and suppression of list output `-m` are canceled so that list output is not executed.

2.5 Example of Specifying Command Lines

The three types of examples when specifying command lines are listed and described as follows.

■ Example of Specifying Command Lines

[Example 1]

```
flnk896s
flnk896s file1.obj file2.obj -g -a -help
```

When only the command name is specified or details of options are unclear, the simple help message is displayed by specifying the -help option HELP.

[Example 2]

```
flib896s sys.lib -m sys.mp2 ... *1
flib896s -m sys.mp2 sys.lib ... *2
```

Since the position of options is not fixed, options can be freely written on command line. Options in both examples *1 and *2 are valid and have the same meaning.

[Example 3]

```
flnk896s *.obj -g -o sample.abs
```

Wild card is used to specify two or more input files in this example.

CHAPTER 3

COMMON OPTIONS

Linkage kit has common options that can be used in any tools. These options are also prepared in C compiler and assembler.

This chapter explains the common options of the linkage kit.

The options that are unique in this tool are also described in the respective paragraphs.

- 3.1 List of Common Options
- 3.2 Details of Common Options

3.1 List of Common Options

The following table lists common options that can be used in the linkage kit.

■ List of Common Options

Table 3.1-1 lists common options that can be specified in the linkage kit.

Table 3.1-1 List of Common Options

Function	Option	Remarks
Specifying suppression to read default option file	- Xdof	
Specifying option file name	- f	
Specifying display of help message	- help	
Specifying version number and startup message of program	- V	
Suppression to output version number and startup message of program	- XV	Default
Specifying display of end message	- cmsg	
Suppression to output end message	- Xcmsg	Default
Specifying to set the end code to 1 when warning is issued	- cwno	
Specifying to set the end code to 0 when warning is issued	- Xcwno	Default

3.2 Details of Common Options

The following section describes the common options that can be used in the linkage kit.

■ -Xdof Option

The -Xdof option cancels reading of the default option file. Refer to Section "3.2.1 Specifying Suppression of Default Option File (-Xdof)" for more details.

■ -f Option

The -f option starts reading option from the file in which option is described. Refer to Section "3.2.2 Specifying Reading from Option Files (-f)" for more details.

■ -help Option

The -help option displays the help message. Refer to Section "3.2.3 Specifying Help Message (-help)" for more details.

■ -V Option

The -V option outputs a message at program startup. This message is not displayed when the default processing is executed. Refer to Section "3.2.4 Specifying Version Number/Message Output (-V)" for more details.

■ -XV Option

The -XV option suppresses output of message during startup. Refer to Section "3.2.5 Suppression of Version Number/Message Output (-XV)" for more details.

■ -cmsg Option

The -cmsg option displays the end message of the program. Refer to Section "3.2.6 Specifying Display of End Message (-cmsg)" for more details.

■ -Xcmsg Option

The -Xcmsg option suppresses display of the end message of the program. Refer to Section "3.2.7 Specifying Suppression to Display End Message (-Xcmsg)" for more details.

■ -cwno Option

When a warning is issued in this program, 1 is returned to OS as the end code. Refer to Section "3.2.8 Specifying End Code to 1 when Warning is Issued (-cwno)" for more details.

■ -Xcwno Option

When a warning is issued in this program, 0 is returned to OS as the end code. Refer to Section "3.2.9 Specifying End Code to 0 when Warning is Issued (-Xcwno)" for more details.

3.2.1 Specifying Suppression of Default Option File (-Xdof)

It cancels reading of default option file.

When this option is not specified, default option file is always read.

■ Specifying Suppression of Default Option File (-Xdof)

[Synopsis]

-Xdof

[Parameter]

None

[Description]

It cancels reading of default option file.

When this option is not specified, default option file is always read.

Refer to Section "4.5 Default Option File" for the default option file.

[Note]

This option is valid only when specified in the command line.

[Example]

```
flnk896s test.obj -Xdof
```

3.2.2 Specifying Reading from Option Files (-f)

-f option issues directions to read option from the file that describes option. Contents of the file in the command line and this file are regarded equally.

■ Specifying Reading from Option Files (-f)

[Synopsis]

```
-f < Option file name >
```

[Parameter]

<Option file name>
option or file name that describes an input file.

[Description]

Describe options and input files name into the file that is specified by < Option file name >.
This option issues direction to read contents of the option from the file in which options is described.
Contents of the file specified in the command line and this file are evaluated and processed equally.
Extension of the file name is not determined in default.

[Note]

The -f option itself cannot be specified in the option file.
A maximum of 1,023 letters can be described in the option file.

[Example 1]

```
f2ms -V -f optfile.f2m
```

Contents of optfile.f2m

```
#  
# from FJ-OMF to Motorola-S  
#  
ccp903.abs          # IN  ABS-LM  
-o ccp903.mhx       # OUT Motorola-S
```

This is equivalent to what is written in the command line as follows.

```
f2ms -V ccp903.abs -o ccp903.mhx
```

[Example 2]

```
flib896s syslib.lib -f objfile.opt
```

Describe the module that is registered in syslib.lib to objfile.opt. The librarian creates a library file by referring to the contents of this file.

For example, contents of objfile.opt are as follows:

```
-a
putc.obj, getc.obj, puts.obj, gets.obj,
memchr.obj, strcat.obj, strerr.obj, strpbrk.obj,
strchr.obj, strcmp.obj, strcpy.obj, strlen.obj
```

It can also be specified as shown below including specifying the library name.

```
flib896s -f libfile.opt
```

In this case, contents of objfile.opt are as follows.

```
syslib.lib
-a
putc.obj, getc.obj, puts.obj, gets.obj,
memchr.obj, strcat.obj, strerr.obj, strpbrk.obj,
strchr.obj, strcmp.obj, strcpy.obj, strlen.obj
```

Option file name can be specified twice.

```
flib896s syslib.lib -f objgr1.opt -f objgr2.opt
```

For example, contents of objgr1.opt and objgr2.opt are as follows:

Contents of objgr1.opt

```
-a putc.obj, getc.obj, puts.obj, gets.obj
```

Contents of objgr2.opt

```
-a memchr.obj, strcat.obj, strerr.obj, strpbrk.obj,
strchr.obj, strcmp.obj, strcpy.obj, strlen.obj
```

3.2.3 Specifying Help Message (-help)

-help option issues directions to display the help message without executing the program. Format to specify the command line and option outline are displayed as help message.

■ Specifying Help Message (-help)

[Synopsis]

-help

[Parameter]

None

[Description]

-help option briefly displays the format to specify the command line and list of options.

Help message is output to the standard output (stdout).

When only the command name is specified, the same help message is output.

When input file name and other options are specified, the help message only is displayed without executing programs when this option is specified.



3.2.4 Specifying Version Number/Message Output (-V)

-V option outputs the message during program startup.

■ Specifying Version Number/Message Output (-V)

[Synopsis]

```
-V
```

[Parameter]

None

[Description]

-V option specifies to output the startup message. Note that the respective tools of the linkage kit do not output the startup message in default setting. Be sure to use this -V option to output the startup message.

The startup message includes the program version number, copyright message, etc.

Message is output to the standard output (stdout).

[Example 1]

```
flnk896s ccp903
```

If this option is not specified, the startup message is not displayed when starting execution of program.

When a program is terminated, the OS prompt appears while waiting for the next command input.

[Example 2]

```
flib896s -V
```

When only the -V option is specified, a message including program name, version number, and copyright message, is displayed and the program is terminated immediately.

3.2.5 Suppression of Version Number/Message Output (-XV)

-XV option disables the -V option. This prevents the startup message of a program from being output.

■ Suppression of Version Number/Message Output (-XV)

[Synopsis]

-XV

[Parameter]

None

[Description]

Since the respective tools of the linkage kit do not output the startup message in default setting, specify the -V option to show the startup message.

-XV option is set to disable the -V setting.

[Example 1]

```
flnk896s ccp903
flnk896s ccp903 -XV
```

The startup message is not output when starting execution of program in default setting.

The two options specified as above have the same meaning.

[Example 2]

```
f2ms -f lkit.opt ccp903 -XV
```

When a program is being executed using option files, sometimes setting of an option file might need to be changed temporarily.

When lkit.opt has -V option in it, contents of lkit.opt remain unchanged. However, -XV on the command line can be specified in order to cancel the -V option.



3.2.6 Specifying Display of End Message (-cmsg)

It displays the end message of the program.

■ Specifying Display of End Message (-cmsg)

[Synopsis]

`-cmsg`

[Parameter]

None

[Description]

It displays the end message of a program.

The linkage kit does not display the end message of a program in default setting.

[Example 1]

```
flnk896s ccp903
```

If this option is not specified, no message is output at the end of the program.

At the end of the program, the OS prompt appears waiting for input of next command.

[Example 2]

```
f2ms ccp903 -cmsg  
F2MS COMPLETED FOUND NO ERROR
```

At the end of the program, the end message (program name and presence or absence of errors) is displayed.

3.2.7 Specifying Suppression to Display End Message (-Xcmsg)

It suppresses display of end message.

■ Specifying Suppression to Display End Message (-Xcmsg)

[Synopsis]

`-Xcmsg`

[Parameter]

None

[Description]

It suppresses display of end message.

The linkage kit does not display the end message of the program in the default setting.

Use this option to cancel the display option (-cmsg) to display the end message of the program.

[Example 1]

```
flnk896s ccp903
flnk896s ccp903 -Xcmsg
```

The end message is not output at the end of the program in the default setting.

The two options that are specified as shown above are the same.

[Example 2]

```
f2ms -f lkit.opt ccp903 -Xcmsg
```

When a program is executed using option files, setting of an option file may occasionally need temporary changes.

When the -cmsg option is used in lkit.opt, the -cmsg option can be canceled by specifying -Xcmsg on the command line without changing contents of the lkit.opt.

3.2.8 Specifying End Code to 1 when Warning is Issued (-cwno)

It sets the end code to 1 when a warning is issued while the program is being executed.

■ Specifying End Code to 1 when Warning is Issued (-cwno)

[Synopsis]

```
-cwno
```

[Parameter]

None

[Description]

The end code is set to 1 when a warning is issued while the program is being executed.

SOFTUNE linkage kit sets the end code of 0 when a warning is issued normally.

[Example 1]

```
flnk896s ccp903 -cwno
```

When a warning is issued during execution of program, the end code to OS is 1.

[Example 2]

```
flnk896s ccp903
```

When a warning is issued during execution of program, the end code to OS remains 0 that is the default value.

3.2.9 Specifying End Code to 0 when Warning is Issued (-Xcwno)

It sets the end code to 0 when warning is issued while the program is being executed.

■ Specifying End Code to 0 when Warning is Issued (-Xcwno)

[Synopsis]

`-Xcwno`

[Parameter]

None

[Description]

It returns the end code to 0, that is the default value, when warning only is issued during execution of program.

SOFTUNE linkage kit sets the end code to 0 when warning only is issued.

Use this option to cancel the option (-cwno) that sets the end code to 1 when warning is issued.

[Example 1]

```
flnk896s ccp903
flnk896s ccp903 -Xcwno
```

When warning is issued, the end code is 0 in the default setting.

The two settings as specified above are the same.

[Example 2]

```
f2ms -f lkit.opt ccp903 -Xcwno
```

When a program is executed using option files, setting of an option file may occasionally need temporary changes.

When the -cwno option is used in lkit.opt, the -cwno option can be canceled by specifying -Xcwno on the command line without changing contents of the lkit.opt.



CHAPTER 4

OPTION FILES

This chapter explains option files of linkage kit.

- 4.1 Outline of Option File
- 4.2 Specification to Continue in the Option File
- 4.3 Specifying Comment in the Option File
- 4.4 Example of Describing Option File
- 4.5 Default Option File

4.1 Outline of Option File

In option files, file names and options required for processing could have been input earlier in order to simplify input into command line every time.

■ Option File

Option file is the file in which input file name and options that are input from the command line are described.

Syntax for description remains the same as that on the command line.

However, the following two items are added in option file.

- Comment statement can be described.
- Line feed is possible at any desired separating point.

Starts a comment statement with the comment symbol (#) and ends with line feed.

Comment statement and line feed symbol are handled equally as a space on command line.

■ Execution by Specifying Option File

Since the number of characters to be input into command line is limited when specification alone is used, specification becomes impossible if there are too many file names and options to be specified. Also, it can decrease efficiency and affect operation due to input errors.

When process becomes formalized or when there are too many options and file names to be specified, the contents that are described in the file can be treated equally as the specification on the command line in order to save inputting work. Input the necessary file names and options into option file using text editor and execute it using the -f option.

[Example]

```
flib896s -f optfile
```

Content of option file "optfile"

```
prg.lib
-a main.obj
-a send.obj, receive.obj, exchange.obj
-a account.obj
-m prg.mp2
```

The format of the statement in option file is the same as the one in command line. In the above example, options are written separately for each line. However, they can be written in one line.

The number of characters should be limited to 1023 bytes for one line.

```
prg.lib -a main.obj ..... -a account.obj -m prg.mp2
```

This example describes not only options but also the library file (prg.lib) that is a target of editing.

As described above, all specifications that can be described in command line (excluding -f option and -Xdof option) can be described in the option file using the same format.

4.2 Specification to Continue in the Option File

In option file, specification to continue option file is possible by using line feed at a separating point in option and parameter.

■ Specification to Continue in the Option File

When describing options and file names into option file, there are cases that option cannot be described in one line or more than two lines are desired.

Line feed is made possible at a separating point in option and parameter. The following two types of examples describe this occurrence.

[Example 1 Example when content of option file is described in one line]

```
      :  
-a mod01, mod02, obj03, obj04  
      :
```

[Example 2 Example when content of option file is described in two lines]

```
      :  
-a mod01, mod02,  
  obj03, obj04    # the continued line  
      :
```

4.3 Specifying Comment in the Option File

Comment can be input into option file.

■ Specifying Comment in the Option File

When inputting comment into option file, use (#) as the comment start symbol.

[Example when comment is input into content of option file]

The underlined portion is comment.

```
# Example of Library Options  
syslib.lib          # INDICATES LIBRARY FILE  
-a mod01, mod02, obj03, obj04 # Add Modules
```


4.4 Example of Describing Option File

The examples below show how to specify option file that can be handled equally as -a mod01, mod02, obj03, and obj04 in command line.

■ Example of Describing Option File

<pre>-a mod01,mod02,obj03,obj04</pre>	Same
<pre>-a mod01, mod02 ,obj03 , obj04</pre>	Inserting a space before and after comma
<pre>-a mod01,mod02,obj03,obj04 # comment</pre>	Adding comment to end of statement
<pre># comment line -a mod01,mod02,obj03,obj04</pre>	Inserting a comment line
<pre>-a mod01,mod02,obj03, obj04</pre>	line feed after comma continues the parameter
<pre>-a mod01,mod02,obj03 ,obj04</pre>	line feed before comma continues the parameter
<pre>-a mod01,mod02,obj03, # comment obj04</pre>	inserting a comment statement continues the parameter
<pre>-a mod01,mod02,obj03,obj04</pre>	line feed after -a continues all parameters

4.5 Default Option File

This is one of the functions of option file. The previously specified option files can be read and executed without specifying -f at system startup.

This function is called default option file.

■ Default Option File

The default option file is one of the functions of option file. The previously specified option files can be read and executed without specifying -f at system startup. This function is called default option file.

The default option file is read each time the system starts up. The user can choose to specify the startup option earlier.

Specify option -Xdof at system startup in order to suppress function of the default option file. When this option is specified, default option file is not read.

Table 4.5-1 shows names of the default option files as they are set.

Table 4.5-1 Names of Default Option Files of the Linkage Kit

Name of tool	Name of program	Name of option file
Linker	flnk896s	flnk896.opt
Librarian	flib896s	flib896.opt
Object type converter	f2ms	f2m.opt
	f2hs	f2h.opt
	f2is	f2i.opt
	f2es	f2e.opt
	m2is	m2i.opt
	m2es	m2e.opt
	i2ms	i2m.opt
	e2ms	e2m.opt
	m2bs	m2b.opt
	m2ms	m2m.opt
	h2bs	h2b.opt
	h2hs	h2h.opt

Procedure to refer to the default option file is shown as follows.

- When the environment variable OPT896 or OPT has already been set.
The file in the directory that is set by the environment variable is referred to.
 - Linker, librarian
%OPT896%\default option file
 - Object tool
%OPT%\default option file
- When the environment variable OPT896 or OPT has not been set
The default option file in the development environment directory is referred to.
 - Linker, librarian
%FETOOL%\LIB\896\default option file
 - Object tool
%FETOOL%\LIB\default option file

Note:

When default option file cannot be found, the linkage kit does not issue error message.

PARTII LINKER

PARTII describes the specifications, options, and output lists of a linker.

CHAPTER 5 SPECIFICATIONS OF A LINKER

CHAPTER 6 LINKER OPTIONS

CHAPTER 7 OUTPUT LIST FILE OF THE LINKER

CHAPTER 8 LINKER RESTRICTIONS AND Q&A

CHAPTER 5

SPECIFICATIONS OF A LINKER

This chapter explains the overview and functions of a linker.

- 5.1 Overview of a Linker
- 5.2 Functions of a Linker
- 5.3 Types of Sections
- 5.4 Combining Sections
- 5.5 Locating Sections
- 5.6 Automatically Locating Sections
- 5.7 Searching Libraries
- 5.8 ROM and RAM Areas
- 5.9 Sections to be Transferred from ROM to RAM
- 5.10 CPU Information File
- 5.11 Mixing of Objects for a Linker

5.1 Overview of a Linker

A linker is a tool that combines multiple object modules that are output by an assembler, then allocates memory location addresses. The purpose is to create a load module in the executable form.

■ Overview of a Linker

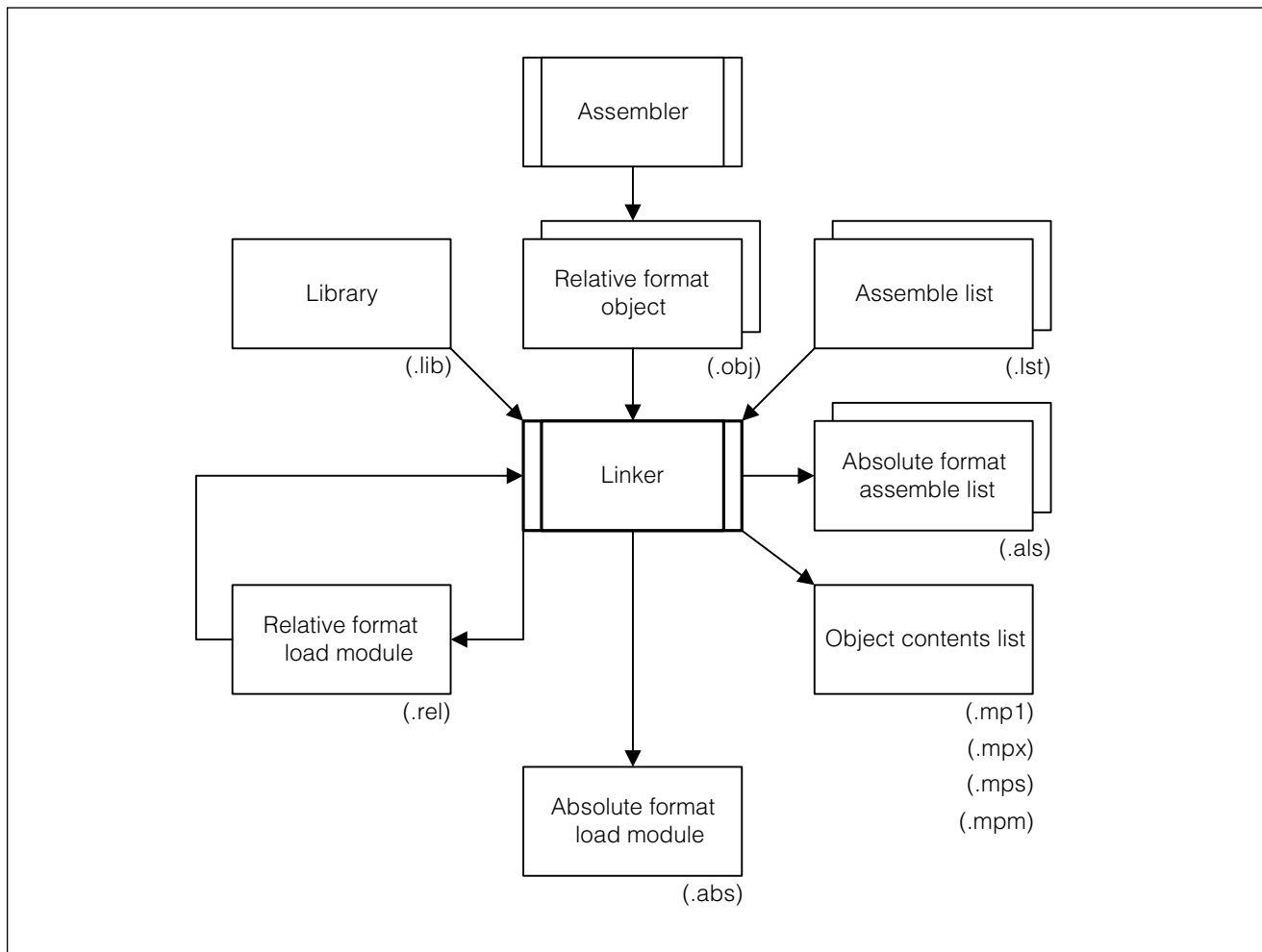
The larger a program to be developed, the more it is difficult to describe everything in one source program.

Also, if you develop a program using a C compiler, you usually need to import library files in C.

A linker is used to combine multiple object modules that are related with each other, then allocate memory location addresses to create a load module in the executable form.

Figure 5.1-1 shows the relationship between a linker and input-output files.

Figure 5.1-1 Relationship between a Linker and Input-output Files



5.2 Functions of a Linker

A linker has many functions which can be roughly classified into the following four groups.

- Control on input-output files and messages
 - Control on combining and locating sections
 - Control on searching libraries
 - Setting entry addresses and symbol values
-

■ Control on Input-output Files and Messages

The following describes the overview of control and input-output files. For details, see Section "5.2.1 Control on Input-Output Files and Messages".

- There are the following four types of input files:
 - Object module file that an assembler outputs
 - List file
 - Relative format load module file that a linker outputs
 - Library file
- Object module files and load module files are processed in the order that they are written on a command line or in an option file.
- There are the following three types of output files:
 - Absolute format load module file that is the final objective
 - Relative format load module that can be input again
 - Link map list file
- The output load module format (absolute or relative) may be specified and the output file name may be changed.
- For a map list, the number of lines per page and the width of a page may be changed.
- Specify an option to output one of the following four types of files.
 - To output these files, the link load module must be in the absolute format.
 - Absolute format assemble list file is created by adding to the absolute format a list file that an assembler has output.
 - External symbol cross-reference information list that shows the cross-reference between external defined symbols and reference symbols used in modules
 - Local symbol information list that shows the information on local symbols used in each module
 - Section detailed map list that shows section location addresses in each module
- Messages consist of startup messages including the program version number, help messages briefly describing how to use the program, and error messages. You can specify whether or not to output a startup message and the level of detecting a warning status.

■ Control on Combining and Locating Sections

The following is an overview of the control on combining and locating sections. For details, see Section "5.2.2 Control on Combining and Locating Sections".

- You can make sure that a section is not located outside the specified area by specifying the address ranges of ROM and RAM.
- When specifying sections, you may create a group of multiple sections to process them in a batch or select sections according to attributes.
- You may use a wildcard to specify sections. This will allow you to easily specify sections to be combined or located when many sections are involved.
- The function to support creating a ROM is provided.
- The sections may be automatically located to the specified ROM and RAM areas.

■ Control on Searching Libraries

The following describes in detail the control on searching a library. For details, see Section "5.2.3 Control on Searching Libraries".

- When a program is developed in C, the runtime library in C required for linking can be automatically identified and combined (searching the default library file).
- Multiple libraries created by the user may be searched.
- The library file to be searched may be specified for each symbol.
- Library searching may be inhibited.

■ Setting Entry Addresses and Symbol Values

A value may be temporarily allocated to an undefined external symbol or an entry address may be temporarily set. For details, see Section "5.2.4 Setting Entry Addresses and Symbol Values".

5.2.1 Control on Input-Output Files and Messages

Section 5.2.1 describes those linker functions that control the input-output files and messages.

■ Specifying Input Object Files

The input files for a linker include object module files that an assembler outputs and relative format load module files that a linker outputs.

All the input files must be specified, which can be facilitated by using wildcard.

■ Specifying an Output Load Module File Name

An output load module file name created after linking is based on the file name of the module that a linker first inputs at default.

This function is provided to change the default output file name because it is often inappropriate as a name to represent the entire linking result.

In particular, the output file name is difficult to understand if a file name is specified using a wildcard. In such a case, you are recommended to specify an output file name.

■ Inheriting Debugging Information

Information on symbols and source files is required for debugging.

If debugging information is specified to be created (-g option) in C or for an assembler, an object module will contain debugging information.

A linker inherits this debugging information. You can decide either to output it into a load module or to delete it.

■ Specifying the Output Format

Use this function to specify creating an absolute or relative format load module as a result of linking.

■ Specifying a List File Name

A list file name is created based on the output object file name. Use this function to change this default file name.

■ Changing the Format of a List File

Page control is performed when a list file is created. You may change the number of lines per page and number of characters per line.

A long symbol name is truncated to fit into one line. You can specify to display the name exactly as you defined it.

■ Selecting the Warning Check Level

A warning indicates a minor error. A warning message is issued if a problem occurs but the linking processing may be continued. Some warnings must be resolved and others may be ignored. Use this function to select the check level.

■ Selecting whether or not to Display a Startup Message

You can select whether or not to display the tool name and the copyright at startup.

■ Selecting whether or not to Display a Termination Message

You can select whether or not to display a termination message.



5.2.2 Control on Combining and Locating Sections

Section 5.2.2 describes those linker functions used to control combining and locating sections.

■ Specifying ROM and RAM Areas

Defining an area name by specifying the address range of the ROM and RAM areas allows you to use this area name instead of an address when specifying where to locate a section. It also allows you to make sure that the section is not located outside the range.

To have sections automatically located, locate them in this area range.

■ Specifying the Order of Locating Sections and the Location Addresses

All the sections may be located in any area in any order. To specify a section name, use a wildcard.

Additionally, you can specify sections using a name plus a section contents type. Use these together with a wildcard to collect only the sections with the same contents type (code, data, etc.).

■ Creating a Group of Sections

A linker combines and locates sections. If many sections are used to create a program, specifying where to locate the sections will be troublesome.

Multiple sections may be handled as if they are one section by giving them a group name and collecting them in continuous areas.

■ Support for Creating a ROM

When developing a program in C, a variable with an initial value is created. The variable must often be rewritten and other processing must be performed.

In an application to be imported, the initial value data must be placed in ROM and transferred to RAM before the application is executed.

This function enables these operations. For details, see Section "5.9 Sections to be Transferred from ROM to RAM".

5.2.3 Control on Searching Libraries

Section 5.2.3 describes those linker functions that control searching for a library.

■ Specifying a Path to Search a Library

To specify a path to search a library, specify the directory containing the C library in an environment variable normally. However, specify the path if a library created by the user is stored in another directory.

■ Specifying a Library File to be Searched

As the library to be searched, specify the name of a library file created by the user in addition to the runtime libraries provided by a C compiler.

■ Specifying a Library File to be Searched for Each Symbol

If, in the linking processing, multiple library files are searched and you know that the same external symbol is contained in the libraries, use this function to explicitly specify which library module should be linked.

■ Inhibiting the Search for a Library

You can disable the search of a default library or all the libraries.

5.2.4 Setting Entry Addresses and Symbol Values

Section 5.2.4 describes those linker functions that set entry addresses and symbol values.

■ Specifying an Entry Address

Use this function to set in an output load module an address at which to start executing the program.

■ Setting an External Symbol Value

An error occurs if an external symbol is not defined after linking because of an incomplete program or an incorrect external symbol name.

Use this function to set a temporary value to temporarily remove this error and create a load module executable for the time being.

5.3 Types of Sections

The minimum unit that a linker can combine is a section.

Depending on the purpose of using sections in a program, they are located and combined differently.

Section 5.3 describes section names, contents types, location attributes, and combination attributes.

■ Section Name

A section name is used to identify a section.

■ Types of Section Contents

Depending on the purpose of usage, there are the following seven types of section contents:

An assembler determines the attributes of execution, read, or write. Table 5.3-1 shows the section types.

Table 5.3-1 Section Types

Type	Description	Attribute
CODE	Program code area	Executable, Read
DATA	Variable area	Read, Write
CONST	Area of variable with initial value	Read
STACK	Stack area	Read, Write
IO	I/O area	Read, Write
DIR	direct area	Read, Write
DIRCONST	direct area with initial values	Read

■ Section Location Attribute

There are two section location attributes representing whether or not the section is relocatable. Table 5.3-2 shows the section location attributes.

Table 5.3-2 Section Location Attributes

Attribute	Description
ABS	Section in which absolute addresses are specified
REL	Relocatable section

■ Section Combination Attribute

There are two section combination attributes representing whether the section is shared or combined. Table 5.3-3 shows the section combination attributes.

Table 5.3-3 Section Combination Attributes

Attribute	Description
PUBLIC	Sections are combined in succession.
COMMON	Sections are combined and overlapped at the same address.

■ Section Identification

A linker handles the sections with the same section name, contents type, and combination attribute and the REL attribute as the same section.

A linker does not locate a section with the ABS attribute.

Since a linker identifies a section by the section name, do not define sections with the same section name and different types of contents and attributes.

5.4 Combining Sections

For a linker, we often write "combining multiple objects" but more accurately we should write "combining sections in objects".

A section may be combined through a simple connection combination (PUBLIC) and shared combination (COMMON).

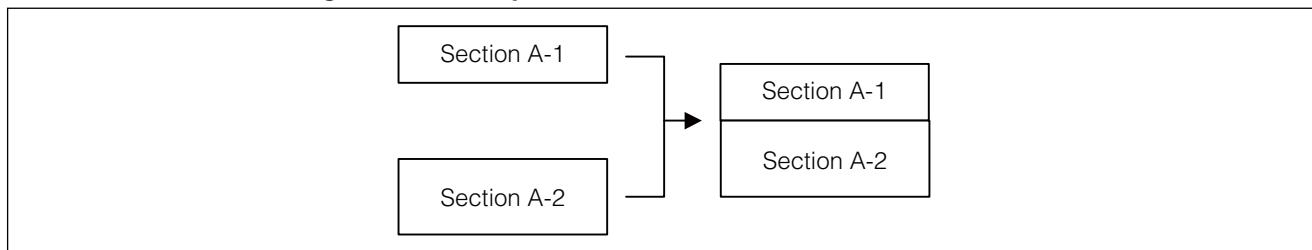
■ Simple Connection Combination of Sections

The REL sections with the same section name and contents type and the connection attribute of PUBLIC are connected through simple connection combination.

Figure 5.4-1 is an overview of the simple connection combination of the same sections in two object files.

The entire size after combination is the total of the A-1 and A-2 sizes plus the gap size between A-1 and A-2 generated due to boundary adjustment.

Figure 5.4-1 Simple Connection Combination of Sections



■ Shared Combination of Sections

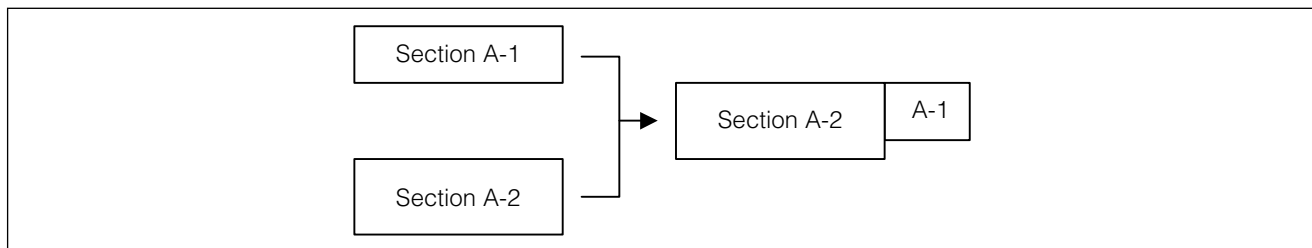
The REL sections with the same section name and contents type and the connection attribute of COMMON are connected through shared combination.

Use this function, for example, for a data section without initial values.

Figure 5.4-2 is an overview of shared combination of the same sections in two object files.

The entire size after combination is the larger one of the A-1 and A-2 sizes.

Figure 5.4-2 Shared Combination of Sections



5.5 Locating Sections

A linker combines the same sections and then determines the location addresses of sections. Section 5.5 describes how a linker locates sections including a case where the user specifies the addresses.

■ Links of Sections

Only relative sections may be combined or located.

Absolute sections may not be combined or located.

Sections are combined or located in the following way:

- 1) The same sections are collected from object modules.
- 2) These sections are combined according to their combination attributes.
- 3) Then the sections are located.

Sections are located according to an option concerning a section location order, if any specified, or otherwise, according to the order in which they appear in an object file.

For details, see Sections "5.5.1 Example of Location when the Order of Combining Sections is not Specified", "5.5.2 Example of Location when the Order of Combining Sections is Specified", and "5.5.3 Example of Location when the Section Group is Specified".

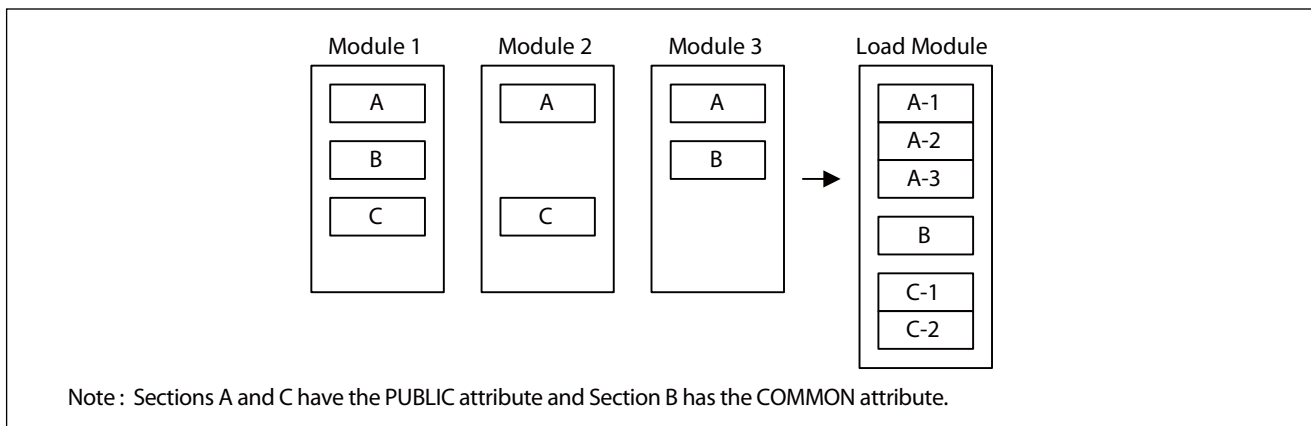
5.5.1 Example of Location when the Order of Combining Sections is not Specified

Section 5.5.1 describes an example of location when the order of combining sections is not specified according to Figure 5.5-1.

■ Example of Location when the Order of Combining Sections is not Specified

If Modules 1, 2, and 3 are input in this order, Sections A, B, and C appear in this order. Therefore, the location addresses are A, B, and C in ascending order.

Figure 5.5-1 Example of Location when the Order of Combining Sections is not Specified



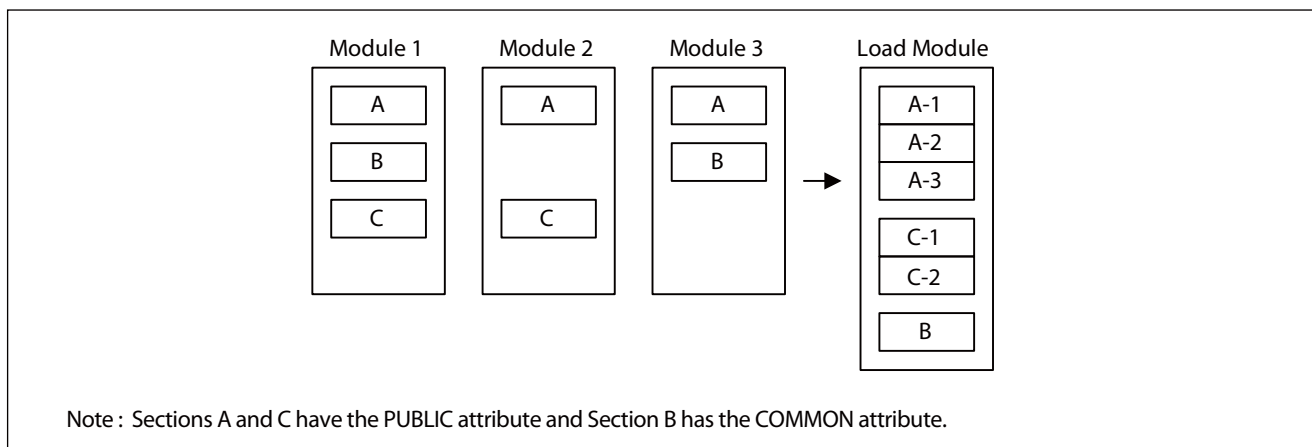
5.5.2 Example of Location when the Order of Combining Sections is Specified

Section 5.5.2 describes an example of location when the order of combining sections is specified according to Figure 5.5-2.

■ Example of Location when the Order of Combining Sections is Specified

If Modules 1, 2, and 3 are input in this order, Sections A, B, and C appear in this order. However, the order of location is specified as A, C, and B.

Figure 5.5-2 Example of Location when the Order of Combining Sections is Specified



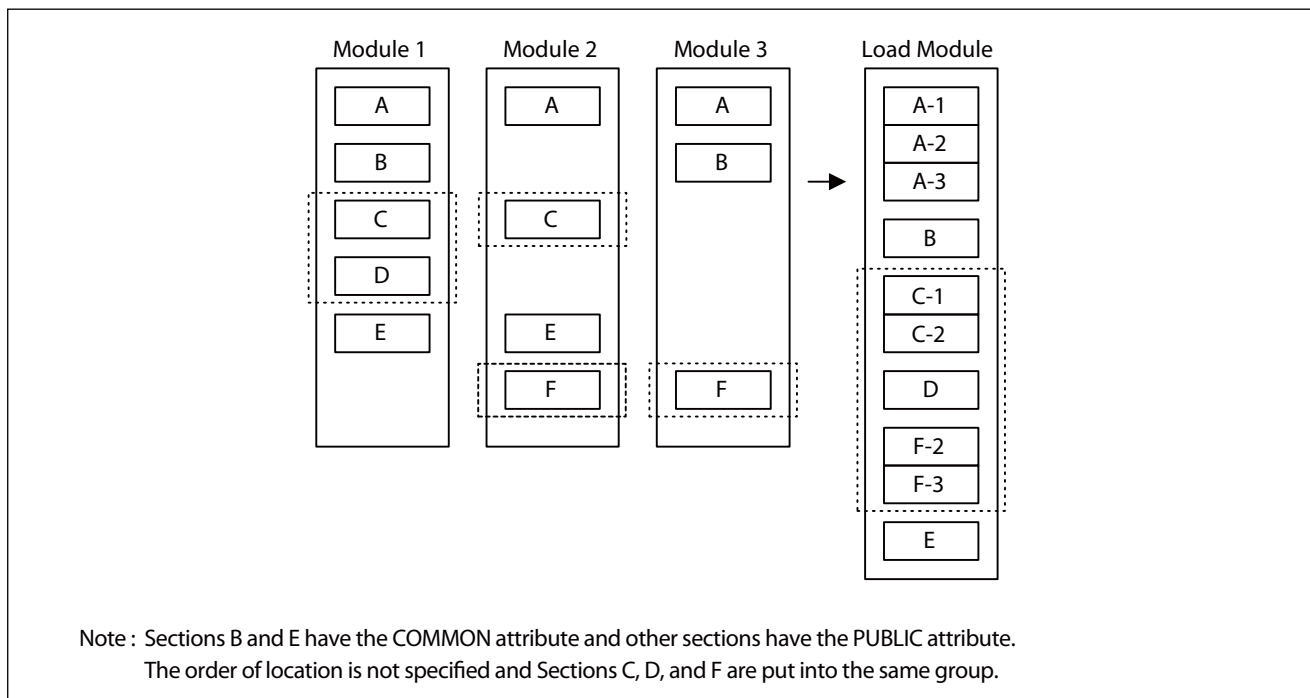
5.5.3 Example of Location when the Section Group is Specified

Section 5.5.3 describes an example of location when the section group is specified according to Figure 5.5-3.

■ Example of Location when the Section Group is Specified

If the group is specified, the sections to belong to each group are located in continuous areas. Sections A, B, C, D, E, and F appear in this order. F is located before E because it belongs to the group to which C belongs (C, D, and F).

Figure 5.5-3 Example of Location when the Section Group is Specified





5.6 Automatically Locating Sections

Normally, a linker determines the section location addresses according to the location specification provided by the user. However, you can have the linker automatically determine section location addresses by specifying the **-AL** option.

If any absolute section exists when sections are located in the area specified in the **-ra** or **-ro** option, the relocatable sections are located so that the location addresses do not overlap. At this time, the sections with a larger alignment value or size are located before others so that the optimal location is achieved and the available area is the smallest.

■ Automatically Locating Sections

This linker supports automatically locating the following two types of sections:

- Automatically locating sections when **-AL 1** is specified
- Automatically locating sections when **-AL 2** is specified

For details of automatically allocating sections, see Sections "5.6.1 Automatically Locating Sections when **-AL 1** is Specified", and "5.6.2 Automatically Locating Sections when **-AL 2** is Specified".

5.6.1 Automatically Locating Sections when -AL 1 is Specified

If -AL 1 is specified, a linker locates the relocatable sections so that their location addresses do not overlap with the absolute sections existing in the area.

■ Determining Location Addresses

The sections with the area name specified in the -sc option may be automatically located.

Sections with larger alignment values are located before others. For sections with the same alignment values, ones with a larger size are located before others.

Table 5.6-1 shows the alignment values and sizes of sections.

Table 5.6-1 Alignment Values and Sizes of Sections

Section name	Alignment value	Size
code1	2	0x0180
code2	2	0x0100
code3	2	0x0200
code4	4	0x0100
code5	4	0x0200
code6	2	0x0020

For example, the order of locating sections shown in Table 5.6-1 is determined as follows:

1. Sections with an alignment value of 4 (code4 and code5) are located before those with an alignment value of 2 (code1,code2,and code3).
2. The code5 sections, being larger in size, are located before the code4 sections.

Therefore, the order of locating sections shown in Table 5.6-1 is as shown in Table 5.6-2.

Table 5.6-2 Alignment Values and Sizes of Sections

Order of location processing	Section name	Alignment value	Size
1	code5	4	0x0200
2	code4	4	0x0100
3	code3	2	0x0200
4	code1	2	0x0180
5	code2	2	0x0100
6	code6	2	0x0020

Sections are located in the smallest available area where they can be located.

■ Example of Location when -AL 1 is Specified

The following is an example of location when the linker options are specified and the sections contents are as shown below (Figure 5.6-1 and Table 5.6-3).

Figure 5.6-1 Option Specification for a Linker

```
-ro ROM=0x8000/0x88FF
-sc code1+code2+code3+code4+code5+code6=ROM -AL 1
:
```

Table 5.6-3 Contents of Sections

Section name	Location attribute	Address range	Alignment value	Size
code1	REL	-	2	0x0180
code2	REL	-	2	0x0100
code3	REL	-	2	0x0200
code4	REL	-	4	0x0100
code5	REL	-	4	0x0200
code6	REL	-	2	0x0020
AbsSec	ABS	0x8120 to 0x821F	0	0x0100

Figure 5.6-2 shows an example of location when -AL 1 is specified.

As shown in this example, specify -AL 1 to have a linker optimally locate sections in the specified area so that they do not overlap with absolute sections and the available area is the smallest.

Figure 5.6-2 Example of Location when -AL 1 is Specified

0x8000	code4
0x8100	code6
0x8120	AbsSec
0x8220	code5
0x8420	code3
0x8620	code1
0x87A0	code2
0x88A0	Unused available
0x88FF	

Area specified in -sc

5.6.2 Automatically Locating Sections when -AL 2 is Specified

If -AL 2 is specified, a linker determines whether sections without any location address specified in the -sc option should be located in the ROM or RAM area, and locates them in an available space of the selected area.

Section 5.6.2 describes the location destinations for each section type and the order of determining location addresses.

Figure 5.6-4 shows an example of location when -AL 2 is specified.

■ Section Types and Location Destinations

If -AL 2 is specified, a linker automatically locates sections without location specifications. At this time, the linker determines their location destinations (areas) according to their section types as shown in Figure 5.6-4.

Table 5.6-4 Section Types and Location Destinations

Location destination	Section type
ROM area (specified in -ro)	CODE CONST DIRCONST
RAM area (specified in -ra)	IO DIR DATA STACK

■ Determining Location Addresses

If -AL 2 is specified, a linker determines the section location addresses in the order shown in Table 5.6-5.

As shown in Table 5.6-5, the order specified by the user is prioritized to the automatic location.

A linker always searches a place to locate a section, starting from low-order addresses.

Table 5.6-5 Section Location Destinations

Order	Section to be processed	Location destination and method
1	Section with the ABS attribute	Located at the address provided with the section
2	Register bank specified in the -rg option	Located at the concerned register bank area
3	Section with a specified location address in the -sc option, e.g., as "-sc Section=0x0100"	Located at the specified address
4	Section with a specified location area in the -sc option as "-sc Section=ROM"	In the specified area, the linker searches a place where it can locate the section without overlapping with another section, then locates it.
5	Section without a specified location	The linker determines the location area according to Table 5.6-4. Then, in the determined area, the linker searches a place where it can locate the section without overlapping with another section. The linker then locates it.

■ Example of Location when -AL 2 is Specified

The following is an example of location when the linker options are specified and the sections contained in modules are as shown below (Figure 5.6-3 and Table 5.6-6).

The area of register bank 0 are the sixteen bytes from 0x0180 through 0x018F.

Figure 5.6-3 Linker Option Specification

<pre> file1.obj, file2.obj, file3.obj -ro ROM=0x8000/0xFFFF -ra RAM=0x000000/0x0007FF -sc irect=0xFF00 -AL 2 </pre>

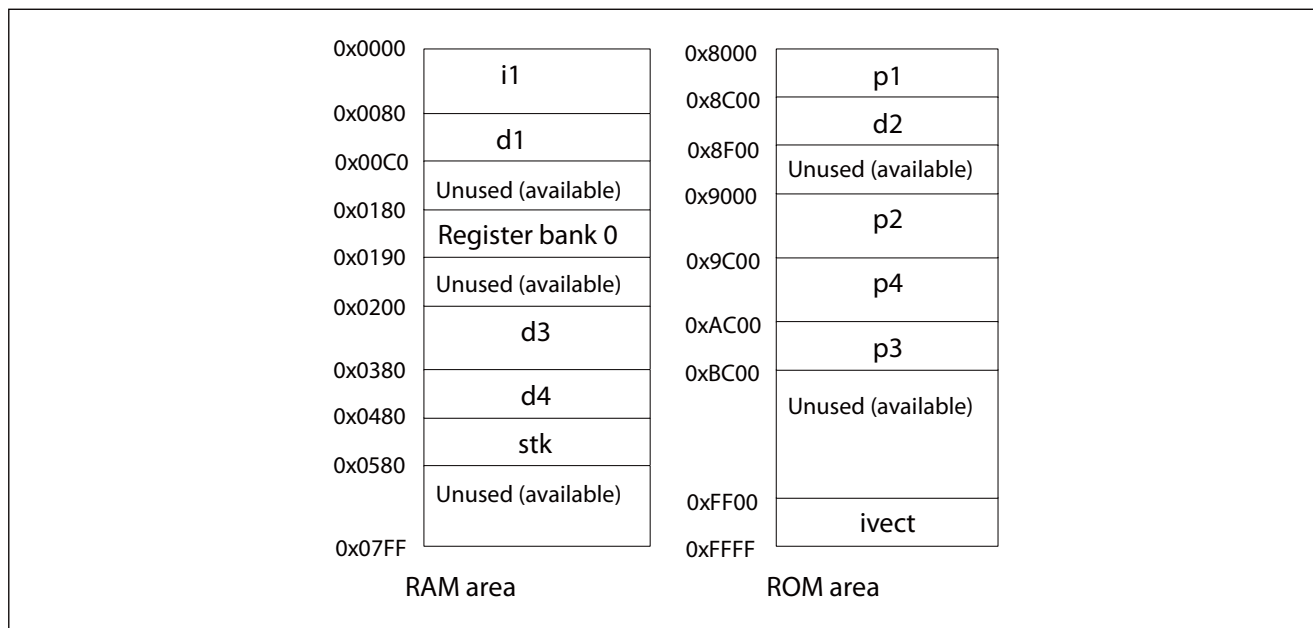
Table 5.6-6 Sections Contained in Modules

Module	Section Name	Type	Location attribute	Address	Size
file1.obj	p1	CODE	REL	-	0x0C00
	p4	CODE	REL	-	0x1000
	stk	STACK	REL	-	0x0100
	ivect	CONST	REL	-	0x0100
file2.obj	i1	IO	ABS	0x0000	0x0080
	d1	DATA	REL	-	0x0040
	d3	DATA	ABS	0x0200	0x0180
file3.obj	p3	CODE	REL	-	0x1000
	d2	CONST	REL	-	0x0300
	p2	CODE	ABS	0x9000	0x0C00
	d4	DATA	REL	-	0x0100

Figure 5.6-4 is an example of location when -AL 2 is specified.

Thus, specifying -AL 2 causes a linker to automatically locate sections in the specified area. This allows the user to specify only the area and section location minimally required for the program operations which frees the user from having to specify the section location.

Figure 5.6-4 Example of Location when -AL 2 is Specified



5.7 Searching Libraries

Specify a library file to be searched by a linker in one of the following three ways:

- **Setting the default library**
 - **Specification in the -l option**
 - **Specification in the -el option**
-

■ Specifying a Library to be Searched

A linker, if any undefined symbol remains when the specified input files have been combined, searches a library file to solve this status.

The linker searches library files recursively so as not to omit anything in the search.

Specify a library file to be searched in one of the following three ways:

● Setting the default library

If a program is written in C, a C library is required at the time of linking.

It is troublesome for the user to specify the library files to be searched at the time of linking. Furthermore, if the user specifies wrong files, unintended modules will be combined.

To prevent this, a C compiler uses an assembler's pseudo instructions to provide the information on library file names to be selected. Then, the assembler sets the information in an object module.

A library file name thus set in the object module to be linked is called the default library.

● Specification in the -l option

To search a library file not defined as the default library, you must specify it at the time of linking.

If you want to create and link a library file, either write a pseudo instruction to specify a library using an assembler, or specify it in the -l option when you start up a linker.

For details on the -l option, see Section "6.2.21 Retrieval Library File Specification (-l)".

● Specification in the -el option

One or more library files may be specified.

Different library files may contain an external defined symbol with the same name. (Avoid such a status because it may cause a malfunction.)

The -el option is used to identify a library file name in which to search a symbol for each symbol. For details, see Section "6.2.23 Library Specification for Each Symbol (-el)".

■ Order of Searching a Library File

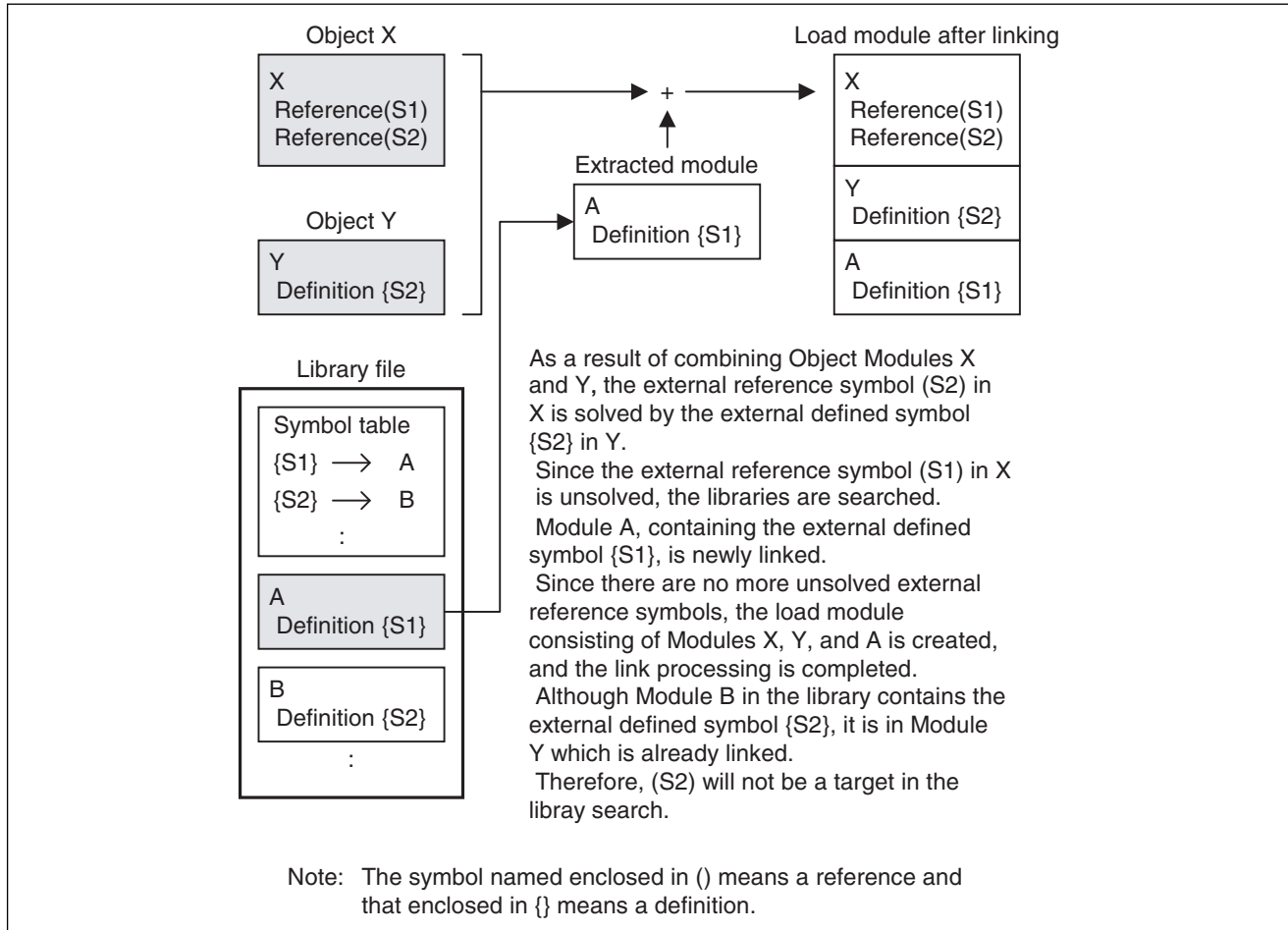
A linker solves a symbol with the -el option specified, then searches libraries in the order specified in the -l option. Lastly, it searches the default library. This series of searches will be repeated until no more modules are imported from library files.

5.7.1 Example of a Search when there is One Library File (1)

There are three examples of a search when there is one library file. Section 5.7.1 provides one of these examples as shown in Figure 5.7-1.

■ Example of a Search when there is One Library File (1)

Figure 5.7-1 Example of a Search when there is One Library File (1)

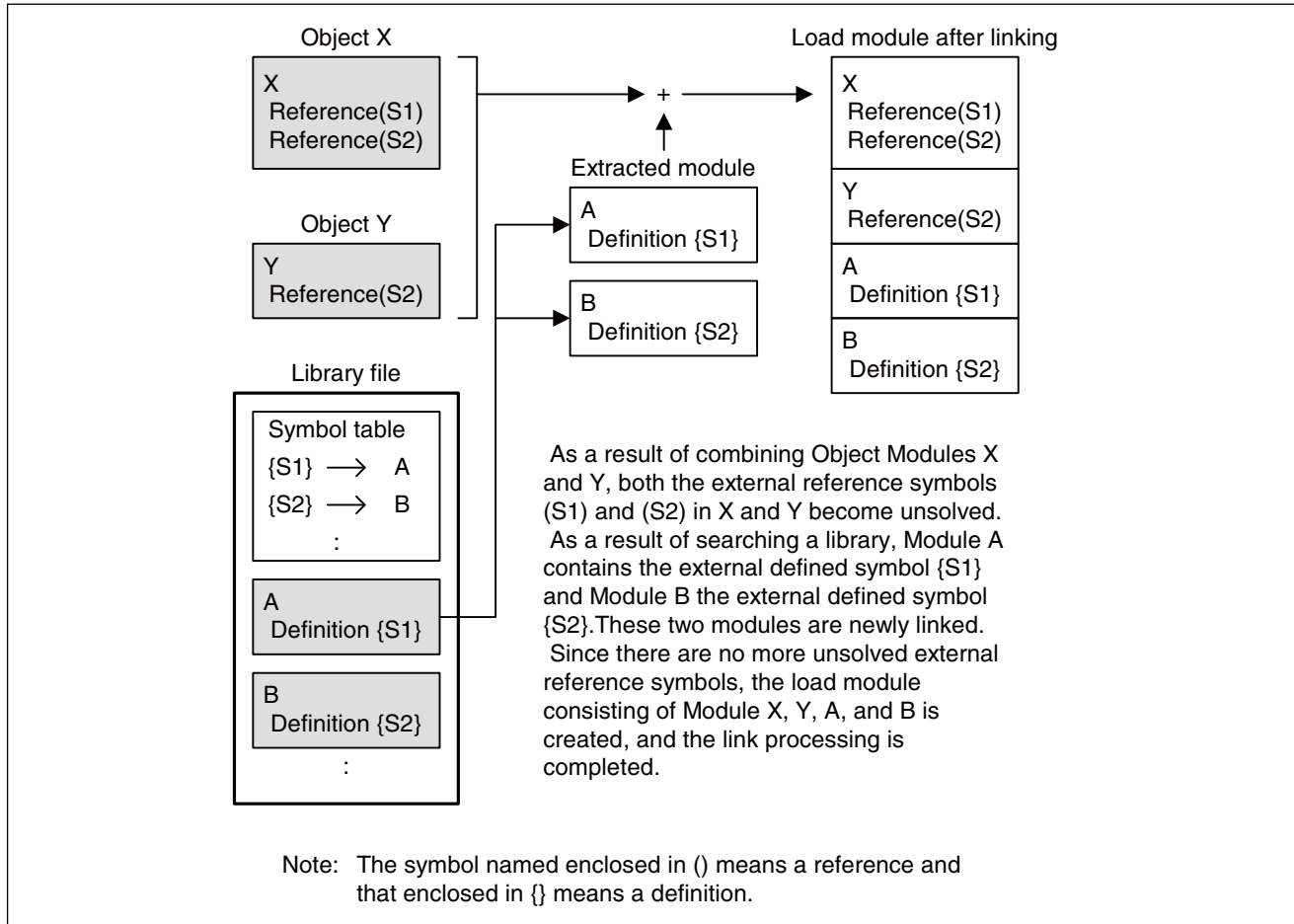


5.7.2 Example of a Search when there is One Library File (2)

There are three examples of a search when there is one library file. Section 5.7.2 describes one of these examples as shown in Figure 5.7-2.

■ Example of a Search when there is One Library File (2)

Figure 5.7-2 Example of a Search when there is One Library File (2)

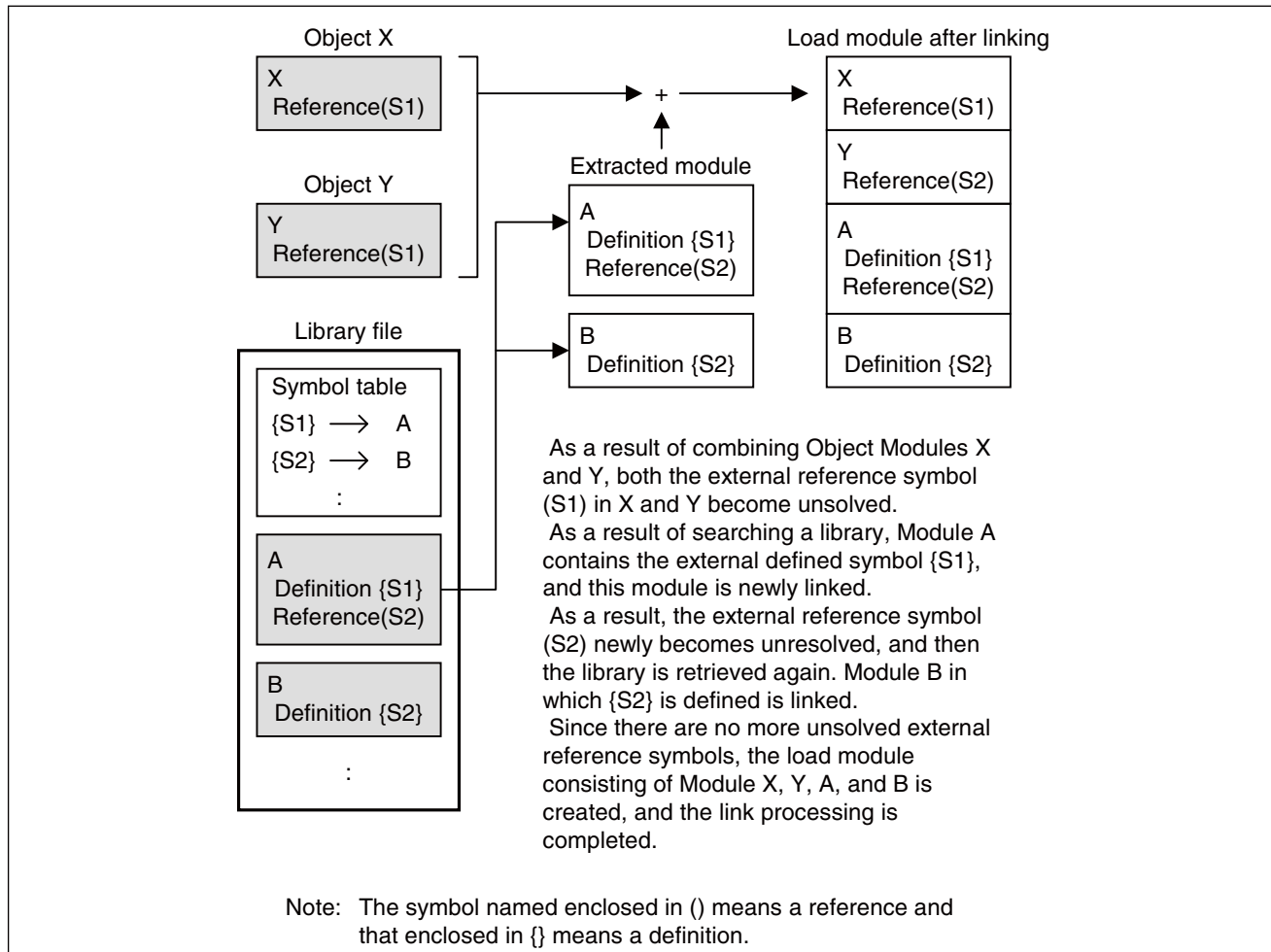


5.7.3 Example of a Search when there is One Library File (3)

There are three examples of a search when there is one library file. Section 5.7.3 describes one of these examples as shown in Figure 5.7-3.

■ Example of a Search when there is One Library File (3)

Figure 5.7-3 Example of a Search when there is One Library File (3)

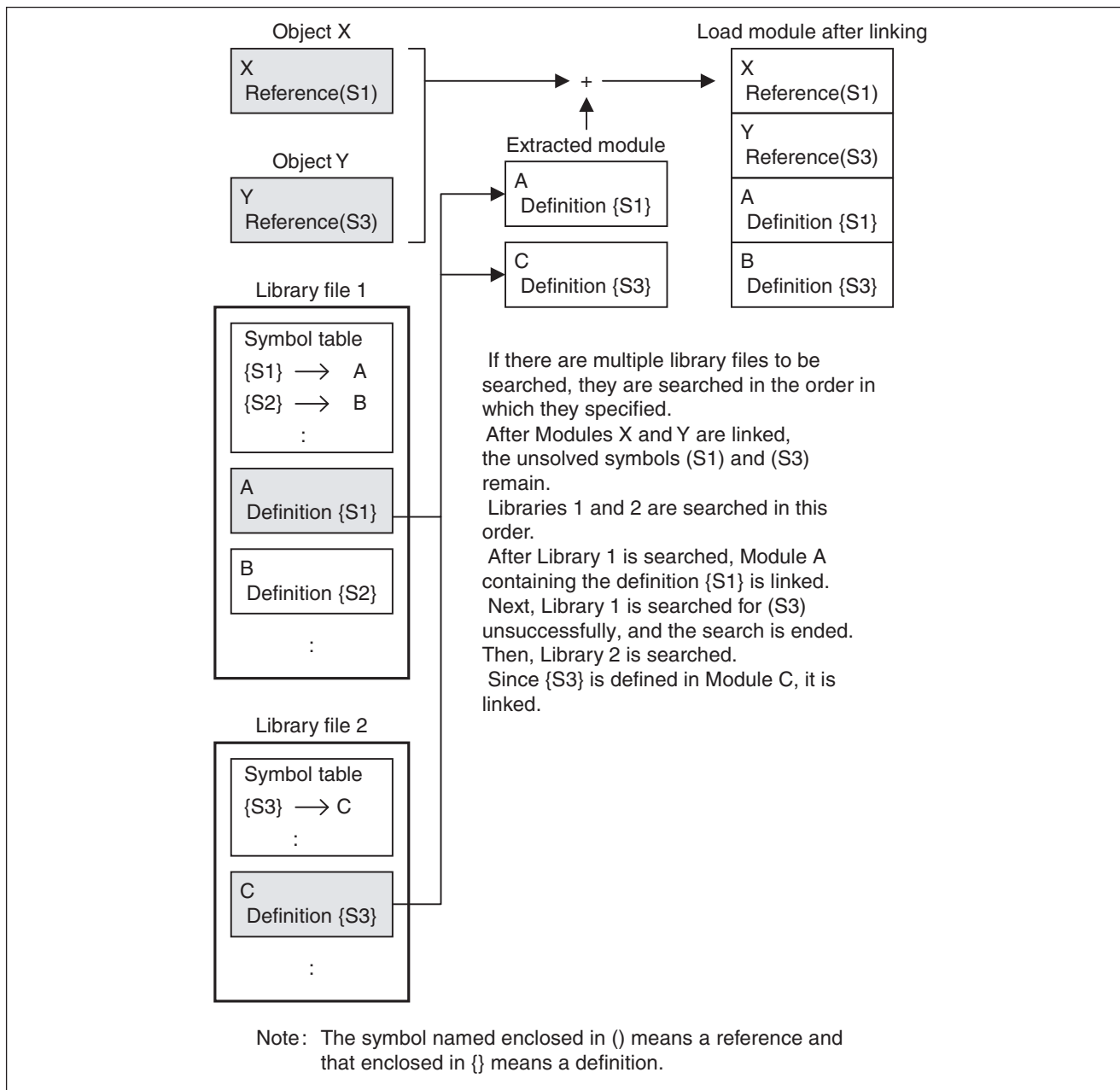


5.7.4 Example of a Search when there are Multiple Library Files (1)

There are two examples of a search when there are multiple library files. Section 5.7.4 describes one of these examples as shown in Figure 5.7-4.

■ Example of a Search when there are Multiple Library Files (1)

Figure 5.7-4 Example of a Search when there are Multiple Library Files (1)

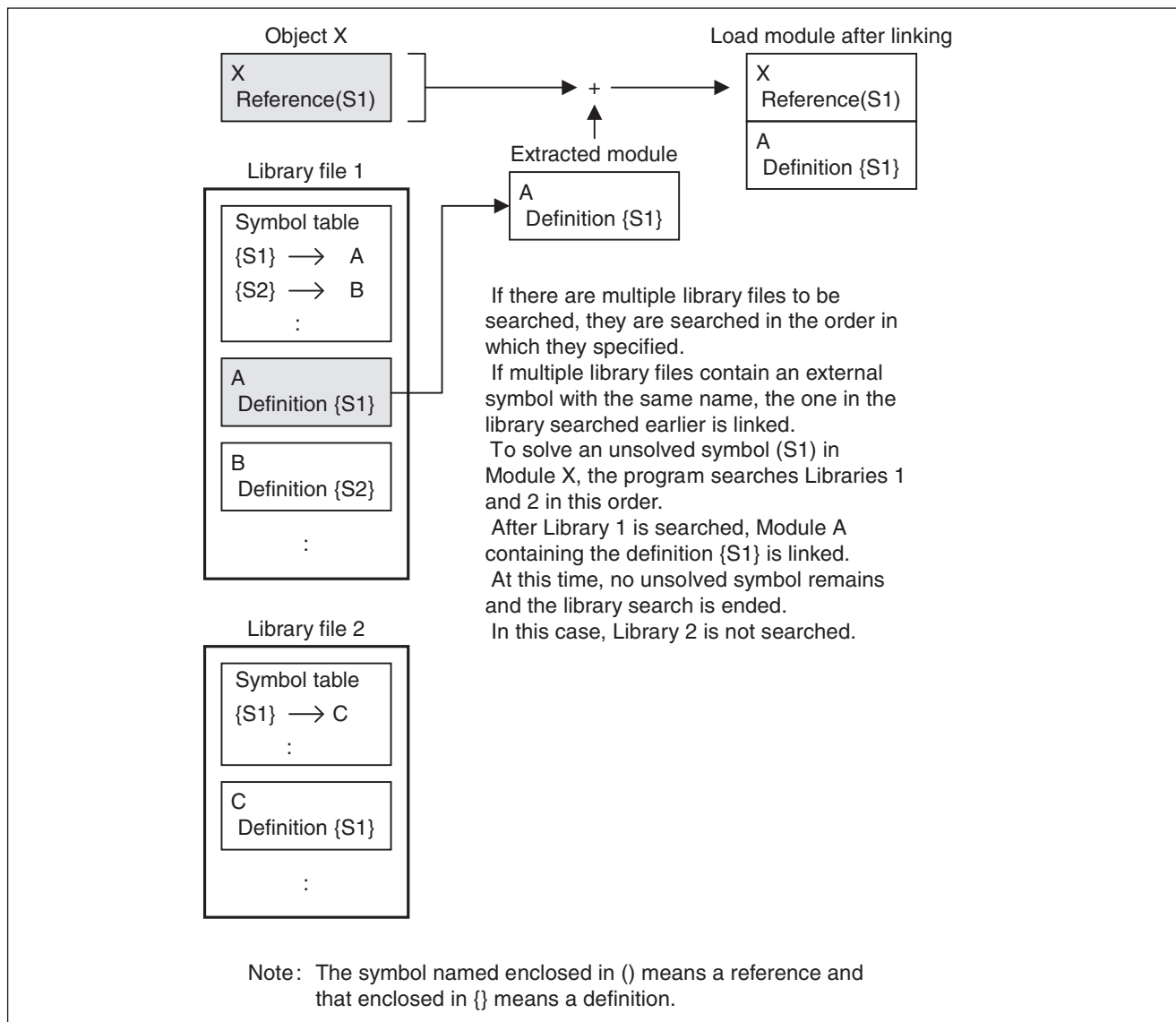


5.7.5 Example of a Search when there are Multiple Library Files (2)

There are two examples of a search when there are multiple library files. Section 5.7.5 describes one of these examples as shown in Figure 5.7-5.

■ Example of a Search when there are Multiple Library Files (2)

Figure 5.7-5 Example of a Search when there are Multiple Library Files (2)

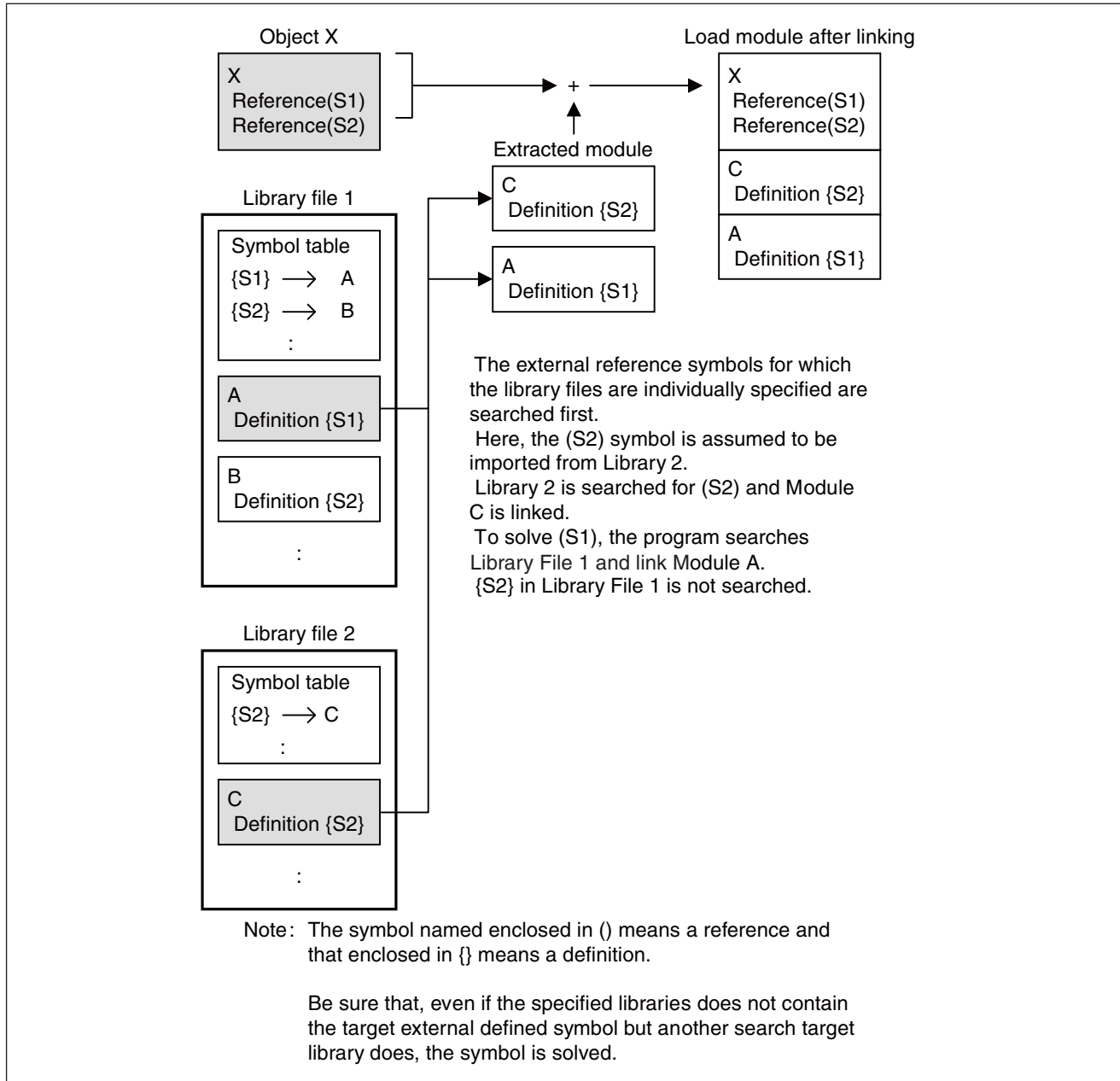


5.7.6 Processing when Library Files are Individually Specified

Section 5.7.6 shows the processing when library files are individually specified as shown in Figure 5.7-6.

■ Processing when Library Files are Individually Specified

Figure 5.7-6 Processing when Library Files are Individually Specified



5.8 ROM and RAM Areas

When you develop an application to be imported, there are often restrictions on the ROM and RAM sizes and the address range that can be used.

By notifying these areas to a linker at the time of linking, you may check for a size exceeding the limit and a section located to an unusable address.

To allocate sections automatically, sections are allocated in the area specification range.

■ Setting of ROM and RAM Areas and Section Location

To have a linker locate sections, specify the section name (CODE) and the starting address at which to locate the section (0x8000) as shown in [Example 1].

[Example 1]

```
-sc CODE=0x8000, DATA=0x0180
```

Check the ending address of a section in the map list that is output as a result of linking.

Specify the starting and ending addresses at which to locate a section to have this linker check whether it is located within the specified range.

First, as shown in [Example 2], use the -ro and -ra options to determine the location address range and associate it with the area name.

[Example 2]

```
-ro CodeA=0x8000/0xFFFF  
-ra DataA=0x0100/0x57F
```

The area name CodeA represents the address range of 0x8000 through 0xFFFF.

The area name DataA represents the address range of 0x0100 through 0x057F.

The section location specification using an area name is as shown in [Example 3].

[Example 3]

```
-sc CODE=CodeA, DATA=DataA
```

In the section location using the area name, this program can check whether the section is located in the specified address range.

5.9 Sections to be Transferred from ROM to RAM

When developing a program using a C compiler, a variable with an initial value is created and you must often rewrite the variable and perform other processing. Such a variable, being rewritten at the time of execution, must be in RAM when the application is executed. Therefore, in a program to be imported, the initial value data must be placed in ROM and transferred to RAM before the application is executed. The section to be transferred from ROM to RAM is a function that enables such a usage.

■ Sections to be Transferred from ROM to RAM

When developing a program using a C compiler, a variable with an initial value is created and you must often rewrite the variable and perform other processing.

In a program to be imported, the variable data with initial values are placed in ROM. However, since it is being rewritten at the time of execution, it must be in RAM when the application is executed.

Therefore, the initial value data is transferred to RAM before the application is executed.

To facilitate such a usage, this linker, as long as the sections to be transferred from ROM to RAM are specified, supports the system of solving the reference addresses of a program on RAM and placing the data with initial values on ROM.

■ Using the Sections to be Transferred from ROM to RAM

Specify the sections to be transferred from ROM to RAM in the -sc option as follows:

The sections gathering variables with initial values shall be INIT.

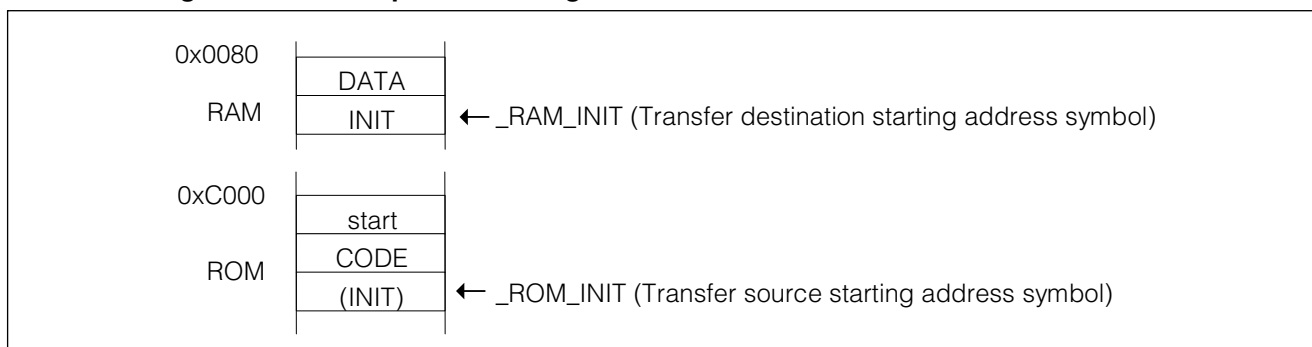
DATA shall be the sections containing variables without initial values, start shall be a program used to transfer the data of variables with initial values on ROM to RAM, and CODE shall be the application program to be executed.

-sc DATA+INIT=0x0080,start+CODE+@INIT=0xC000

As shown in the above, specify to locate INIT on RAM (0x0080), and also to locate INIT on ROM (0xC000) using a section name starting with an at sign, @.

If the section is thus located, INIT is processed as a section to be transferred from ROM to RAM, and the location is as shown in Figure 5.9-1. Then, the addresses are solved on RAM and the initial value data is located on ROM.

Figure 5.9-1 Example of Locating a Section to be Transferred from ROM to RAM



At this time, the symbols for showing the beginning of a section to be transferred from ROM to RAM are automatically generated as "_ROM_section-name" and "_RAM_section-name". For example, as _ROM_INIT or _RAM_INIT.

You can use these symbols in the program to transfer the data of variables with initial values on ROM to RAM.

"_ROM_section-name" and "_RAM_section-name" are symbols reserved for a linker. Do not define these names in a program. For details, see Section "8.1 Linker Restrictions". For an example of a program to transfer data of variables with initial values, see Section "8.2 Q&A for Using the Linker".

■ Precautions on the Sections to be Transferred from ROM to RAM

Specifying the sections to be transferred from ROM to RAM changes the Write attribute of the sections unconditionally.

For the section located on ROM, the Write attribute is disabled. For the sections located on RAM, the Write attribute is enabled.

The following section is an example of sections to be transferred from ROM to RAM in a program:

```
.program    sample
.section    init, data
val1:      .word    0x1234
val2:      .word    0x5678
.section    Data, data
val3:      .res.w    0x1
.section    Prog1, code
ld         @val1, r2
ld         @val2, r3
.section    Prog2, code
ldi        @val3, r13
cmp        #0, r13
.end
```

The above program has the init section containing variables with initial values, the Data section containing variable areas, and the Prog1 and Prog2 sections containing program code.

Table 5.9-1 shows for each section whether execution, read or write is enabled or disabled after assembling.

Table 5.9-1 Attributes of Sections after Assembly

Section name	Execution	Read	Write
init	×	○	○
Data	×	○	○
Prog1	○	○	×
Prog2	○	○	×

○ : Enabled ×: Disabled

If init and Prog1 are specified as the sections to be transferred from ROM to RAM at the time of linking, the execution, read, and write for these sections after linking is enabled or disabled as shown in Table 5.9-2.

Table 5.9-2 Attributes of Sections after Linking

Section name	Execution	Read	Write	Remarks
init (RAM)	×	○	○	
init (ROM)	×	○	×	ROM section provided by the linker. Write is disabled.
Data	×	○	○	
Prog1(RAM)	○	○	○	Write is enabled.
Prog1(ROM)	○	○	×	ROM section provided by the linker
Prog2	○	○	○	

○ : Enabled ×: Disabled



5.10 CPU Information File

The linker specifies the CPU from `-cpu` option and automatically specifies the ROM/RAM areas from the CPU information file.

■ CPU Information File

The linker specifies the CPU from the `-cpu` option and selects the information of the appropriate chip to automatically specify the ROM/RAM areas from the CPU information file.

■ ROM/RAM Areas Names

The linker sets the following names for the ROM/RAM areas.

- ROM Areas: `_ROM_*_`
 - Numbers are entered at the asterisk (*) in order from the lower address region starting from 1. If there is only 1 area, the number will be `"_ROM_1_"`.
- RAM Areas: `_RAM_*_`
 - Numbers are entered at the asterisk (*) in order from the lower address region starting from 1. If there is only 1 area, the number will be `"_RAM_1_"`.

These names are used by the `-sc` options.

■ CPU Information File

The following shows the CPU information file name and the search directory.

- CPU Information file name

`896.csv`

- Search Directory

`%FETOOL%\LIB\896`

■ Specifies to Prevent the Internal ROM/RAM Area from being Set Automatically

The linker specifies the CPU from the `-cpu` option by default and selects the appropriate chip information from the CPU information file to automatically set the ROM/RAM areas. Specify the `-Xset_rora` option when you want to deter this function.

Note:

If the CPU information file cannot be found, or the appropriate MB number in the CPU information file does not exist, the linker will issue an error.

5.11 Mixing of Objects for a Linker

The linker (flnk896s) causes an error if objects and libraries incompatible with the target CPU specified by the -cpu option are mixed.

■ Mixing of Objects for a Linker

The F²MC-8FX is different in the operation of the DIVU instruction from the F²MC-8L(see Table 5.11-1).

Table 5.11-1 Operational Comparison of DIVU Instruction

F ² MC-8L	F ² MC-8FX
$T(16\text{bit})/AL(8\text{bit}) = AL(8\text{bit}) \text{ remainder } TL(8\text{bit})$	$T(16\text{bit})/AL(16\text{bit}) = AL(16\text{bit}) \text{ remainder } TL(16\text{bit})$

For this reason, if the F²MC-8L objects and the F²MC-8FX objects are mixed to link, the linker (flnk896s) causes an error.

If the DIVU instruction is unprovided, the F²MC-8L objects and the F²MC-8FX objects can be mixed to link.

To have the F²MC-8L objects and the F²MC-8FX objects mixed to link, specify the Object Mixing Check Inhibit Specification option(-Xobjmixchk).

Notes:

- A mixed link between the F²MC-8L objects and the F²MC-8FX objects with DIVU instruction may cause an unexpected error.
- For the mixed link, pay full attention to the presence or absence of the DIVU instruction.



CHAPTER 6

LINKER OPTIONS

This chapter explains each linker option in more detail.

- 6.1 List of Linker Options
- 6.2 Details of Linker Options

6.1 List of Linker Options

Options are available to specify linker operations in more detail.

■ List of Linker Options

Table 6.1-1 shows the list of linker option.

Table 6.1-1 List of Linker Options (1 / 3)

		Function	Option	Remarks
Input/output control options	Specification regarding output load modules	Output load module file name specification	-o	Default
		Debug information output specification	-g	
		Debug information delete specification	-Xg	Default
		Specification of outputting absolute format load module	-a	Default
		Specification of outputting relative format load module	-r	
		Specification to fill ROM area	-fill	
	Specification regarding output lists	Map list file name specification	-m	Default
		Specification for inhibiting map list output	-Xm	
		Cancellation of omitting names displayed in the list	-dt	
		Output Specification of the Memory Used Information List	-mmi	
		Specification of the number of digits in the list line	-pw	Default 132
		Specification of the number of lines on one list page	-pl	Default 60
		Checksum specification of ROM area	-cs	
	Specification regarding output messages	Warning message output level specification	-w	
Allocation/link options		ROM area specification	-ro	
		RAM area specification	-ra	
		Section allocation	-sc	
		Section group specification	-gr	
		Register bank area specification	-rg	
		Automatic allocation specification	-AL	Default 0

Table 6.1-1 List of Linker Options (2 / 3)

	Function	Option	Remarks
Library control options	Retrieval library file specification	-l	
	Library retrieval path specification	-L	
	Library specification for each symbol	-el	
	Library retrieval inhibit specification	-nl	
	Specification for inhibiting default library retrieval	-nd	
Other link control options	Entry address specification	-e	
	Dummy setting of external symbol values	-df	
	Target CPU specification	-cpu	need
	Specifying CPU Information File	-cif	
	Specification for inhibiting check for presence of debug data	-NCI0302LIB	
	Function that automatically sets internal ROM/RAM area	-set_rora	Default
	Specifies to prevent the internal ROM/RAM area from being automatically set	-Xset_rora	
	User-specified-area check specification	-check_rora	
	User-specified-area check suppression specification	-Xcheck_rora	Default
	Section-placed-area check specification	-check_locate	
	Section-placed-area check suppression specification	-Xcheck_locate	Default
	Specifying check user-unspecified section	-check_section	
	Specifying inhibit check for presence of user-unspecified section	-Xcheck_section	Default
	Object mixing check specification	-objmixchk	Default
	Object mixing check inhibit specification	-Xobjmixchk	
Options regarding absolute format assemble list output	Specification for relative assemble list input directory	-alin	
	Specification for absolute assemble list output directory	-alout	
	Specification for absolute assemble list output	-als	
	Specification for absolute assemble list output module	-alsf	
	Specification for inhibiting absolute assemble list output	-Xals	
	Specification for ROM/RAM and ARRAY list output	-alr	

Table 6.1-1 List of Linker Options (3 / 3)

	Function	Option	Remarks
Options regarding absolute format assemble list output	Specification for ROM/RAM and ARRAY list output module	-alrf	
	Specification for inhibiting ROM/RAM and ARRAY list output	-Xalr	
	Specification for ROM/RAM and ARRAY list symbol and address display position	-na/-an	
Options regarding object content list output	Specification for external symbol cross-reference information list output	-xl	
	Specification for external symbol cross-reference information list file name	-xlf	
	Specification for inhibiting the external symbol cross-reference information list output	-Xxl	
	Specification for local symbol list output	-sl	
	Specification for local symbol list file name	-slf	
	Specification for inhibiting local symbol list output	-Xsl	
	Specification for section detail map list output	-ml	
	Specification for section detail map list file name	-mlf	
	Specification for inhibiting section detail map list output	-Xml	
Common options	Specification for inhibiting default option file read	-Xdof	
	Option file read specification	-f	
	Help message display specification	-help	
	Specification for version number/message output	-V	
	Inhibiting version number/message output	-XV	Default
	End message display specification	-cmsg	
	Specification for inhibiting end message display	-Xcmsg	Default
	Specification to set end code to 1 when warning occurs	-cwno	
	Specification to set end code to 0 when warning occurs	-Xcwno	Default

6.2 Details of Linker Options

This section explains each option of the linker.

For common options in the linkage kit, see "CHAPTER 3 COMMON OPTIONS".

■ Options Related to the Output Module

Details of options related to the output module in Sections "6.2.1 Output Load Module File Name Specification (-o)" to "6.2.6 Specification to fill ROM area (-fill)".

■ Options Related to the Output List

Details of options related to the output list in Sections "6.2.7 Map List File Name Specification (-m)" to "6.2.13 Checksum specification of ROM area (-cs)".

■ Specification Related to Output Messages

Details of options related to output messages in Section "6.2.14 Warning Message Output Level Specification (-w)".

■ Allocation/Link Options

Details of options related to the allocation/link in Sections "6.2.15 ROM Area Specification (-ro)" to "6.2.20 Automatic Allocation Specification (-AL)".

■ Library Control Option

Details of options related to the library control in Sections "6.2.21 Retrieval Library File Specification (-l)" to "6.2.25 Default Library Retrieval Inhibit Specification (-nd)".

■ Other Link Control Options

Details of options related to other link controls in Sections "6.2.26 Entry Address Specification (-e)" to "6.2.40 Object Mixing Check Inhibit Specification (-Xobjmixchk)".

■ Options Related to the Absolute Format Assemble List Output

Details of options related to the absolute format assemble list output in Sections "6.2.41 Relative Assemble List Input Directory Specification (-alin)" to "6.2.49 ROM/RAM and ARRAY Lists Symbol and Address Display Position Specification (-na, -an)".

■ Options Related to the Object Content List Output

Details of options related to the object content list output in Sections "6.2.50 External Symbol Cross-reference Information List Output Specification (-xl)" to "6.2.58 Section Detail Map List Output Inhibit Specification (-Xml)".

6.2.1 Output Load Module File Name Specification (-o)

Specify the file name for the linked load module. If this option is not specified, the output file name is created from the first input file name.

■ Output Load Module File Name Specification (-o)

[FORMAT]

-o < Load module file name > (Default)
--

[Parameters]

<Load module file name>

Output load module file name

[Explanation]

Specify the file name for the linked output load module.

If this option is not specified, the linker creates an output file of the name with the added extension corresponding to the link mode based on the first input file name.

Link mode	Default extension
Absolute format output (-a option)	.abs
Relative format output (-r option)	.rel

If the extension is omitted when specifying a <load module file name>, a similar extension is also added depending on the link mode.

[Example 1]

```
flnk896s putc.obj,getc.obj
```

A load module file is created with the file name putc.abs.

[Example 2]

```
flnk896s *.obj -o outfile
```

A load module file is created with the file name outfile.abs.

When using the wild card to specify an input object file as shown in this example, it is recommended to specify the output file name using this option.

[Example 3]

```
flnk896s *.obj -o outfile.
```

A load module file is created with the file name outfile.

If a period is placed at the end of a file name, an extension is assumed.

[Example 4]

```
flnk896s *.obj -r -o outfile.rel
```

A load module file in the relative format is created with the file name outfile.rel.

6.2.2 Debug Information Output Specification (-g)

Object modules created by specifying the output of debug information in the C compiler or assembler contain debug information to be used by the debugger.

To use debug information after linking, specify the -g option.

■ Debug Information Output Specification (-g)

[FORMAT]

-g

[Parameters]

None

[Explanation]

If debug information is contained in the input object module file or relative format load module file, the linker deletes the debug information in the default output.

To leave debug information in the output load module file, specify this option.

Since the linker does not create new debug information, it is meaningless to specify this option when the input file contains no debug information.

To perform symbolic debug while debugging, specify the -g option for all tools from the C compiler and assembler to the linker.

[Example]

```
flnk896s -f rlnk.opt b1 b2
```

```
— rlnk.opt —  
-r          # relocatable LM output  
-g          # debug info.  
-pw 100     # page width  
-o rel1.rel  # output filename
```

6.2.3 Debug Information Delete Specification (-Xg)

Object modules that are created by specifying the output of debug information in the C compiler or assembler contain debug information to be used by the debugger.

To remove debug information after linking, specify the -Xg option. Otherwise, do not specify the -g option.

■ Debug Information Delete Specification (-Xg)

[FORMAT]

-Xg	(Default)
-----	-------------

[Parameters]

None

[Explanation]

If debug information is contained in the input object module file or relative format load module file, the linker deletes the debug information in the default output. So there is no need to specify this option.

This option is used to cancel the -g option if, for example, the -g option is contained in the option file when linking using the option file.

[Example]

```
flnk896s -f rlnk.opt b1 b2 -Xg
```

rlnk.opt	
-r	# relocatable LM output
-g	# debug info.
-pw 100	# page width
-o rel1.rel	# output filename

6.2.4 Absolute Format Load Module Output Specification (-a)

The **-a** option is an option to specify the creation of a load module of the absolute format which is the final object file of the linker.

■ Absolute Format Load Module Output Specification (-a)

[FORMAT]

-a	(Default)
----	-------------

[Parameters]

None

[Explanation]

This option specifies the load module file output in the absolute format.

Since the default output of the linker is in the absolute format, this option is normally not used. This option is used to cancel the **-r** option specification and to enable the **-a** specification.

Output files of the absolute format are created with the following names.

- If the **-o** option is not specified
Name of the input file specified first with the extension changed to ".abs".
- If the **-o** option is specified
Specified name. If no extension is specified, ".abs" is added to the name.

[Example]

```
flnk896s a1 a2 a3 -r -o a123.abs -a
```

The **-r** option in the middle of the command line is canceled.

An output load module file is created in the absolute format.

6.2.5 Relative Format Load Module Output Specification (-r)

The **-r** option is an option to specify creating a load module of the relative format that can be reentered. A load module of the relative format has a format that gathers multiple modules in one file without performing address resolution.

■ Relative Format Load Module Output Specification (-r)

[FORMAT]

-r

[Parameters]

None

[Explanation]

This option specifies the load module file output in the relative format.

Specify this option when changing the default output (absolute format) of the linker.

If the **-r** option is specified after **-a** option, the **-a** option can be canceled.

A load module of the relative format has a format that gathers multiple object modules in one file without performing address resolution. A file of this format can be reentered in the linker, reducing the number of input files to be specified for the following link processing. However, if any change occurs in a module contained in the load module, it cannot be replaced with a library format file.

If this option is specified, all options related to the absolute format assemble list and object content list are ignored and their files are not output.

Output files are created with the following names.

- If the **-o** option is not specified
Name of the input file specified first with the extension changed to ".rel".
- If the **-o** option is specified
Specified name. If no extension is specified, ".rel" is added to the name.

[Example]

```
flnk896s a1 a2 a3 -r -o a123.rel
```

The output object of the linker is changed to the relative format.

Note:

If the extension of the input file specified first is ".rel", the output file name will be the same. Since the contents of the input file are not saved in this case, specify the output file name with **-o** option to avoid any inconvenience.

6.2.6 Specification to fill ROM area (-fill)

The -fill option is used to fill the specified area with the specified value.

■ Specification to fill ROM area (-fill)

[Format]

`-fill <start address>/<end address>,<filling value>[/<width of bit>][/<endian>]`

[Parameters]

<start address>

Specify the start address in the area where the value is filled. This parameter is not omissible.

<end address>

Specify the ending address in the area where the value is filled. This parameter is not omissible.

<filling value>

Specify the value in which the part where the object data in the area specified in the start address and the end address doesn't exist is filled. This parameter is not omissible.

< width of bit >

Specify the width of the bit of the filling value by either of 8, 16 or 32. This parameter is omissible. The width of the bit at default is 8.

8 : 8 bits

16 : 16 bits

32 : 32 bits

<endian>

Specify the endian of the filling value by B or L. This parameter is omissible. The endian at default is B.

B : Big endian

L : Little endian

[Description]

The part where the object data of the area specified in start address and end address does not exist is filled by the filling value.

Please two or more specify -fill options when there are two or more areas where the value is filled.

When the area where the value is filled overlaps, the overlapping area is filled by the filling value of -fill option that will be specified later.

When the area specified in -cs option overlaps with the area specified in -fill option, the overlapping area is always filled by the filling value specified in -fill option.

When the absolute format load module is made, -fill option becomes effective.

When the relative format load module is made, the following warning message is output, and the specification of -fill option is disregarded.

[warning message]

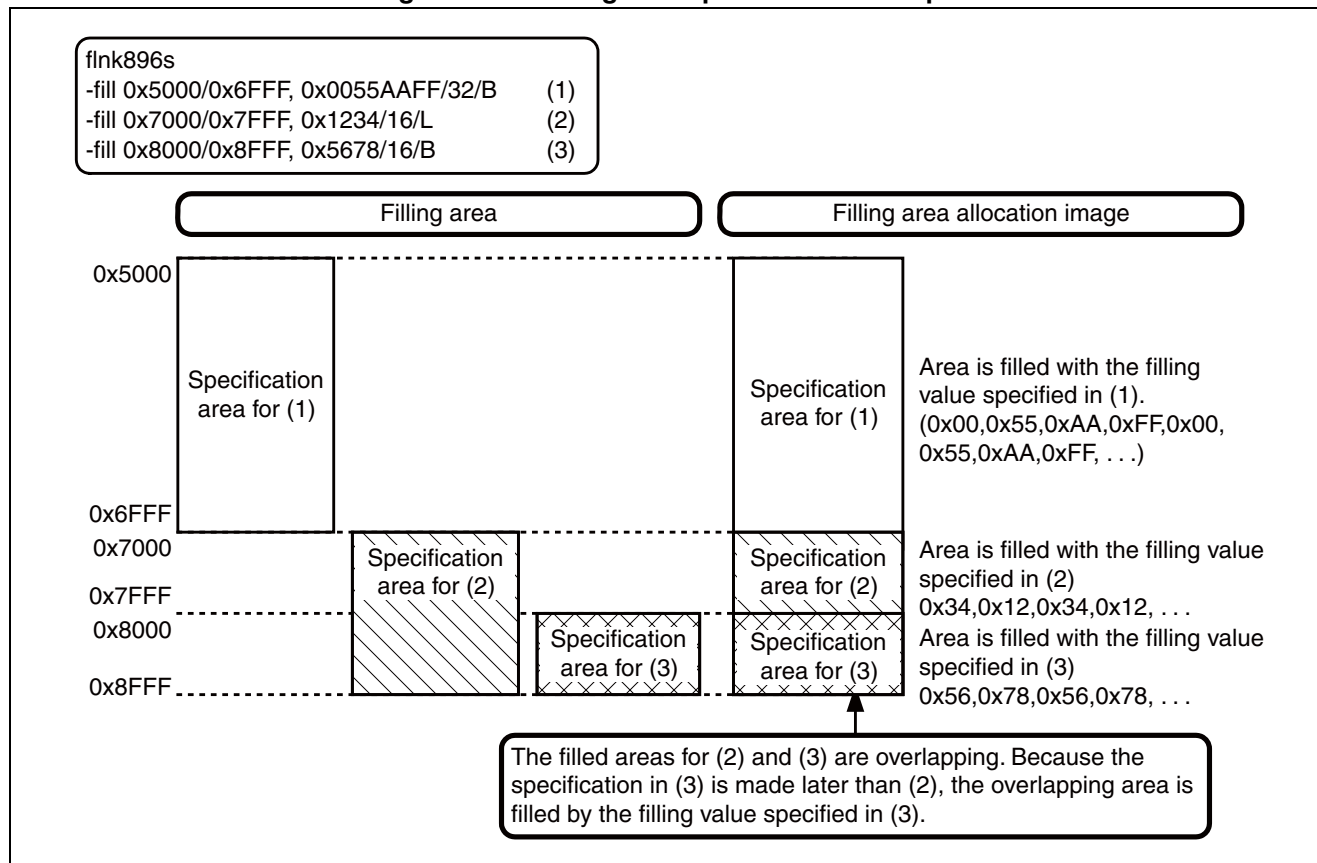
*** W1321L: Ignore (-fill) option at REL mode

[Example]

When the specification is made as follows, the filling area is allocated as shown in Figure 6.2-1.

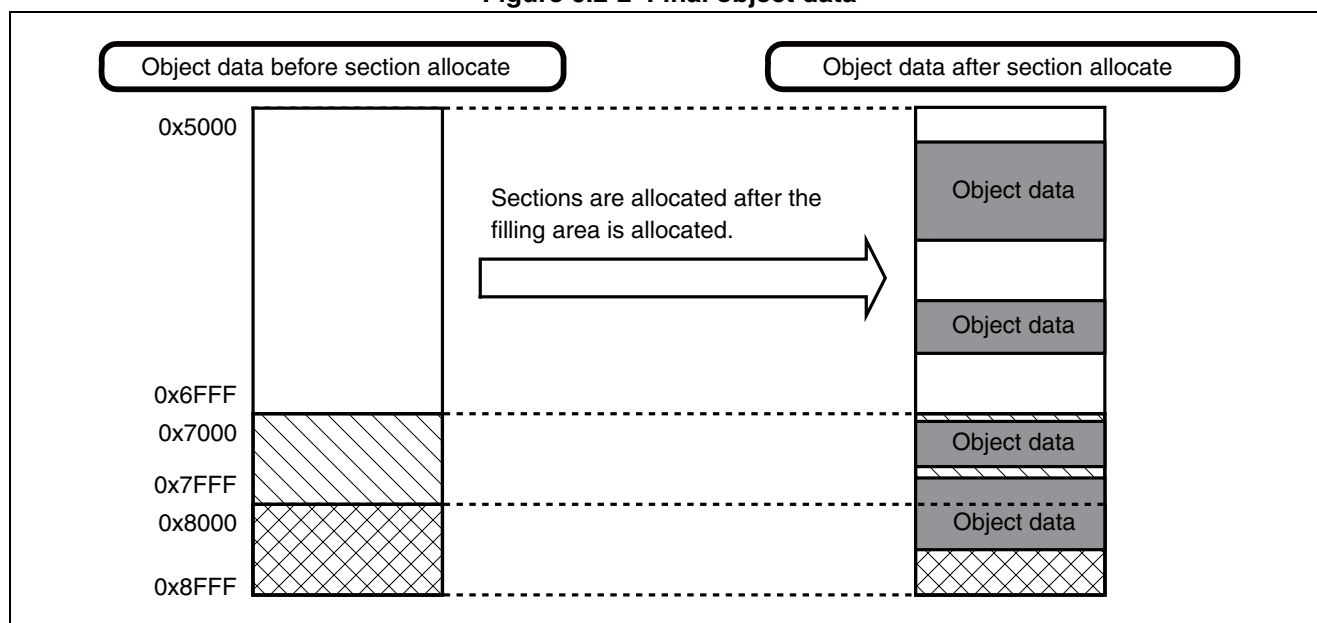
```
flnk896s -fill 0x5000/0x6FFF,0x0055AAFF/32/B -fill
0x7000/0x8FFF,0x1234/16/L -fill 0x8000/0x8FFF,0x5678/16/B
```

Figure 6.2-1 Filling area specification example



When the object data overlaps with the area filled by the filling value, after the area is filled by the filling value as shown in Figure 6.2-2, the object data is arranged.

Figure 6.2-2 Final object data



6.2.7 Map List File Name Specification (-m)

This option specifies the name of the map list file to be output by the linker.
If this option is not specified, a file name is created from the output load module file name.

■ Map List File Name Specification (-m)

[FORMAT]

-m < Map list file name > (Default)

[Parameters]

<Map list file name>

Output map list file name

The parameters cannot be omitted.

[Explanation]

By default, the linker outputs a map list file. At this point, a file is created with the name of the output load module file whose extension is changed to ".mpl".

The -m option is used to change the default map list file name.

If the -m option is specified after the -Xm option, the -Xm option can be canceled.

[Example]

```
flnk896s a1 a2 a3 -r -o a123.rel -m a123.map
```

The map list file name output by the linker is changed to a123.map.

6.2.8 Map List Output Inhibit Specification (-Xm)

This option instructs the linker not to output map list files. If this option is not specified, a map list file is always created.

■ Map List Output Inhibit Specification (-Xm)

[FORMAT]

-Xm

[Parameters]

None

[Explanation]

This option inhibits output of map list files.

If the -Xm option is specified after the -m option, the -m option can be canceled.

By specifying the -Xm option, the -dt, -pw, and -pl options can be canceled.

[Example]

```
flnk896s a1 a2 a3 -r -o a123.rel -Xm
```

Creating a map list file is inhibited.

6.2.9 Canceling the Omission of Names Displayed in the List (-dt)

Names such as the section names and symbol names are displayed in the map list and object content list of the linker. By default list output, considering legibility of the lists, long names are only partially displayed.

This option instructs to output the names without omission.

For the display format of the map lists, see Section "7.2 Link List File".

■ Canceling the Omission of Names Displayed in the List (-dt)

[FORMAT]

-dt

[Parameters]

None

[Explanation]

Symbol names or section names displayed in the map list or object content list are displayed without omission. In such cases, one symbol name or section name is displayed in several lines.

By default output, about 80 characters can be displayed. If a small number is set as the number of digits to be displayed in one list line, the number of characters that can be displayed decreases accordingly. In such cases, symbol names or section names may be displayed with some parts omitted.

[Example]

```
flnk896s a1 a2 a3 -o a123.abs -m a123.map -dt
```

The symbol names and section names in use are displayed in the list without omission.

6.2.10 Output Specification of Memory Used Information List (-mmi)

This outputs the memory used information of the Map List Files that are output by the Linker.

■ Output Specification of the Memory Used Information List (-mmi)

[FORMAT]

-mmi

[Parameters]

None

[Explanation]

This outputs the memory used information list that indicates the usage conditions of the ROM or RAM areas specified by the options of the map list files that are output by the Linker as the default.

The information of usable area, used area and the gap position in the area, and size, etc. is displayed.

[Example]

```
flnk896s 10mp00 im_lnk -mmi
-ra RAM1=0x0080/0x02FF, RAM2=0x0300/0x047F
-ro ROM1=0xC000/0xCFFF, ROM2=0xD000/0xFFFF -AL 2
```

Note:

The following will not be output despite -mmi being specified.

- The memory area is not specified. (-ra or -ro is not specified).
- The map list file output is not valid.

6.2.11 Specification of the Number of Digits in the List Line (-pw)

By default, up to 132 digits can be displayed in one line of the map list and object content list output by the linker. This option is specified to change the number of digits to be displayed in one line.

■ Specification of the Number of Digits in the List Line (-pw)

[FORMAT]

-pw <Number of digits> (Default : 132)
--

[Parameters]

<Number of digits>

Number of digits to be displayed in one line. Specify the number in the range of 70 to 1023.

[Explanation]

The length of one line of the link list file and object content list is specified.

If this value is not specified, 132 digits is set.

[Example]

```
flnk896s a1 a2 a3 -o a123.abs -m a123.map -dt -pw 80
```

The number of digits to be displayed in list line is set to 80.

Note:

The following lists can specify the number of digits of one line by -pw option.

- Link Map List(.mp1)
- Section Detail Map List(.mpm)

The following lists can not specify the number of digits of one line by -pw option.

- Absolute Format Assemble List(.als)
 - External Symbol Cross-reference Information List(.mpx)
 - Local Symbol Information List(.mps)
-

6.2.12 Specification of the Number of Lines on One List Page (-pl)

By default, up to 60 lines are displayed on one page of the map list and object content list output by the linker.

This option is specified to change the number of lines to be displayed on one page.

■ Specification of the Number of Lines on One List Page (-pl)

[FORMAT]

-pl	< Number of lines >	(Default : 60)
-----	---------------------	------------------

[Parameters]

<Number of lines>

Number of lines to be displayed on one page. Specify 0 or in the range of 20 to 255.

[Explanation]

The number of lines on one page of the link list file and object content list is specified.

If this value is not specified, 60 lines is set.

If 0 is specified, page control is canceled.

[Example]

```
flnk896s a1 a2 a3 -o a123.abs -m a123.map -dt -pl 64 -pw 100
```

The number of lines to be displayed on one page of the list is set to 64.

Note:

The following lists can specify the number of lines on one list page by -pl option.

- Link Map List(.mp1)
- Section Detail Map List(.mpm)

The following lists can not specify the number of lines on one list page by -pl option.

- Absolute Format Assemble List(.als)
 - External Symbol Cross-reference Information List(.mpx)
 - Local Symbol Information List(.mps)
-

6.2.13 Checksum specification of ROM area (-cs)

The **-cs** option operates checksum in the specified area, and outputs the result to the map list.

The method of operating checksum has simple addition (SUM) and cyclic redundancy check (CRC).

■ Checksum specification of ROM area (-cs)

[Format]

```
-cs <start address>/<end address>[,<start address><end address>,...],  
<checksum algorithm>,<filling value>
```

[Parameters]

<start address>

Specify the start address in the area where the checksum is operated. This parameter is not omissible.

<end address>

Specify the end address in the area where the checksum is operated. This parameter is not omissible.

<checksum algorithm>

Specify the checksum operation methods by either SUM16, SUM32, CRC16 or CRC32. This parameter is not omissible.

SUM16: 16 bit simple addition

SUM32: 32 bit simple addition

CRC16: 16 bit cyclic redundancy check

CRC32: 32 bit cyclic redundancy check

When SUM16 or SUM32 is specified, either 0 and 1 or 2 can be specified for a complement form following equal (=). The complement form when the specification of the complement form is omitted is 0.

0: no complement

1: complement of 1

2: complement of 2

When CRC16 or CRC32 is specified, the generating polynomial can be specified following equal(=). Generating polynomial when specification is omitted is as follows.

CRC16: 0x8005(CRC-ANSI)

CRC32: 0x04C11DB7(CRC-32 ITU-T)

<filling value>

Specify the filling value in which the part where the object data in the area where the checksum is operated doesn't exist is filled. This parameter is omissible. The filling value at default is 0xff.

[Description]

The checksum is operated by the operation method of specification for the area specified by the start address and the end address.

The result is output to the map file.

At this time, the part where the object data in the area where the checksum is operated doesn't exist is filled by the filling value before the operation of the checksum.

Figure 6.2-3 shows the output example to the map file.

S_Addr., E_Addr., Algorithm and Value are the results of the method of operating the start address, the end address, and the checksum algorithm and the checksum operation.

Figure 6.2-3 output example to the map file

Check Sum(s)			
S_Addr.	-E_Addr.	Algorithm	Value
0x00007000	-0x00007FFF	SUM32=1	0x92DA8F5B

Specify two or more areas by one -cs option when the checksum is operated bringing two or more areas together.

Delimit by the comma and specify the pair in the start address and the end address when two or more areas are specified.

When the area where the checksum is operated overlaps, the overlapping area is filled by the filling value of the -cs option that will be specified later.

When the area specified in -cs option overlaps with the area specified in -fill option, the overlapping area is always filled by the filling value specified in -fill option.

When the absolute format load module is made, -cs option becomes effective.

When the relative format load module is made, the following warning message is output, and the specification of -cs option is disregarded.

[warning message]

*** W1321L: Ignore (-cs) option at REL mode

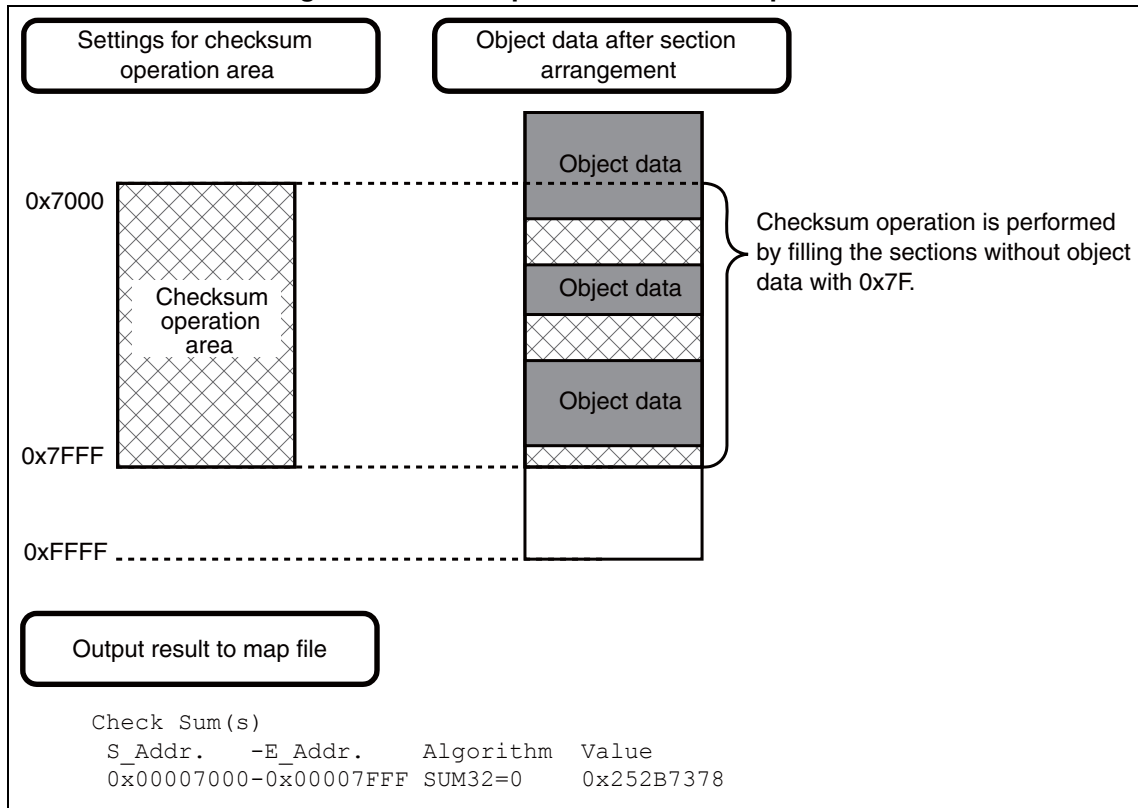
[Example 1]

```
flnk896s -cs 0x7000/0x7FFF,SUM32,0x7F
```

Operate the area of 0x7000/0x7FFF in the checksum by 32-bit simple addition.

The part in which the object data does not exist in the area where the checksum is operated is filled with 0x7F. (Please refer to Figure 6.2-4.)

Figure 6.2-4 Example 1 of Checksum operation



[Example 2]

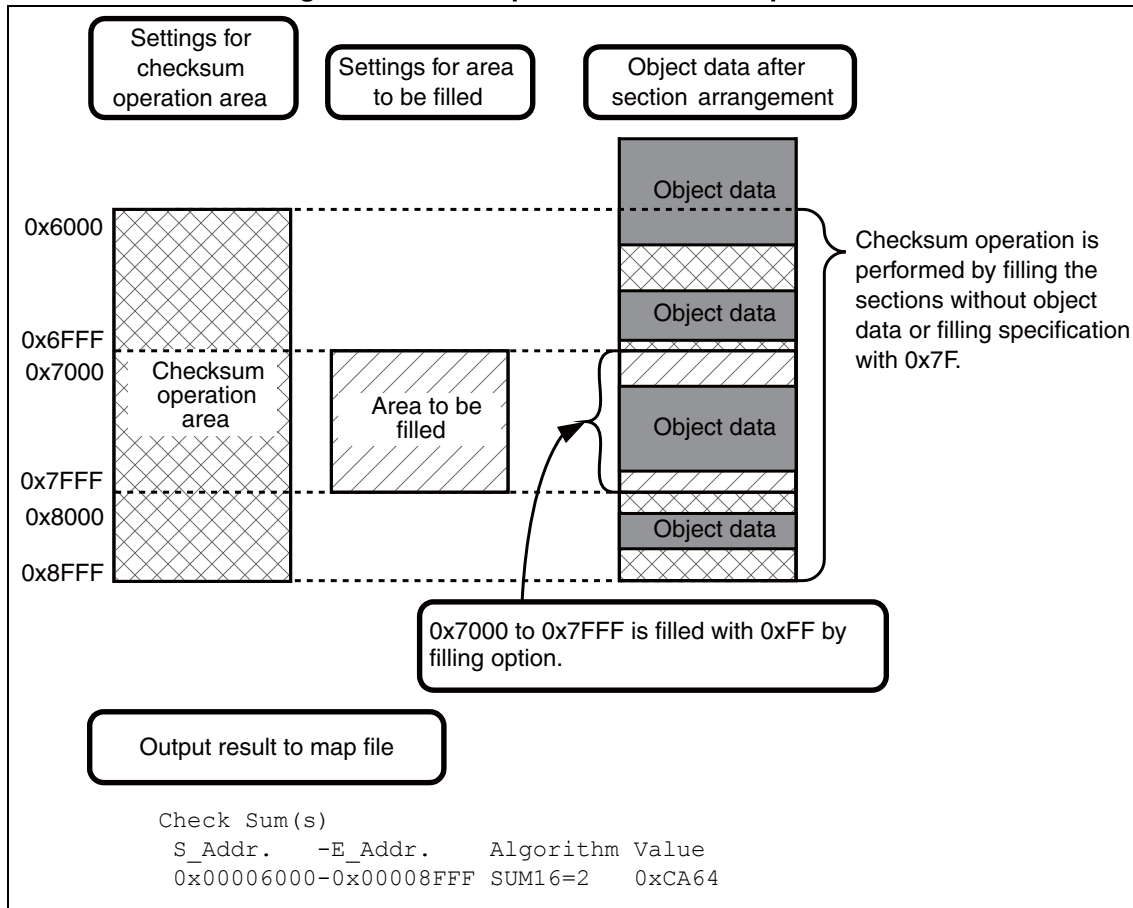
```
flnk896s -fill 0x7000/0x7FFF,0xFF/8 -cs 0x6000/0x8FFF,SUM16=2,0x7F
```

Operate the area of 0x6000/0x8FFF in the checksum by 16 bit simple addition (complement 2).

The part for which there is no object data when -cs and -fill are specified at the same time and -fill is not specified is filled with 0x7F.

Afterwards, the checksum operation is done. (Refer to Figure 6.2-5.)

Figure 6.2-5 Example 2 of Checksum operation

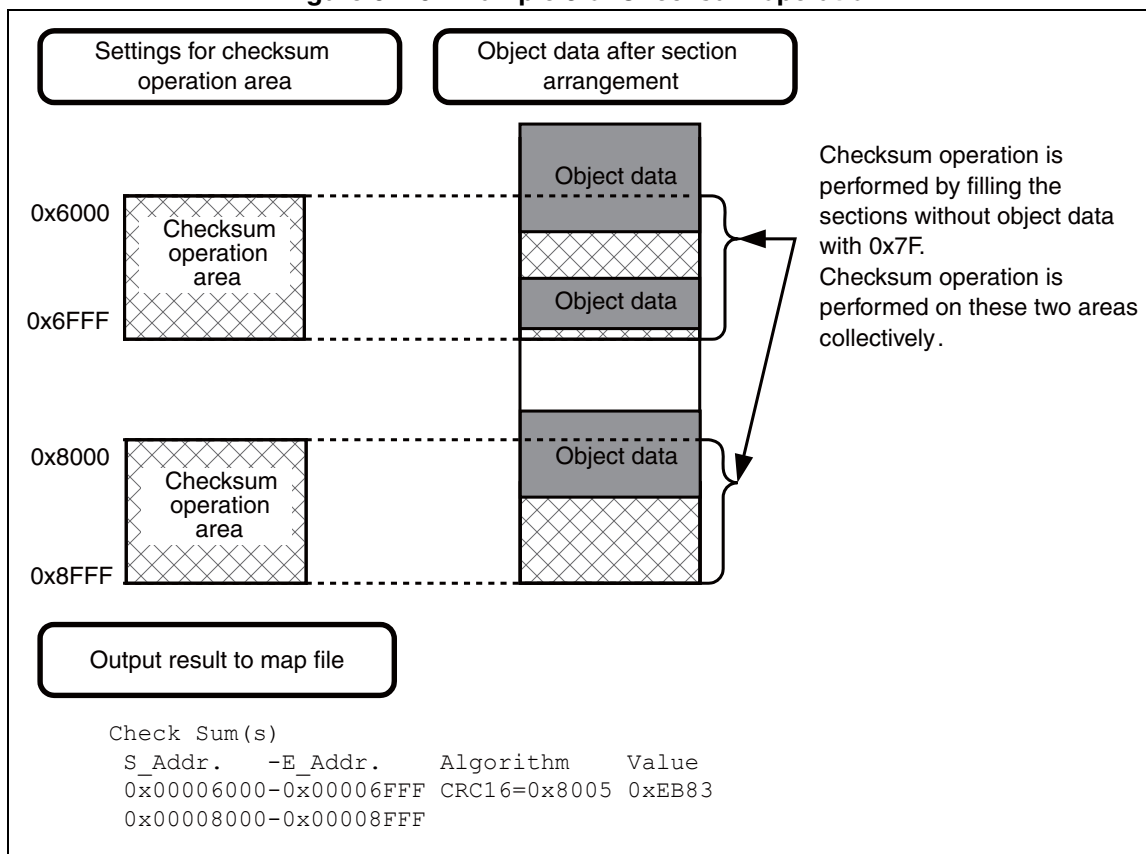


[Example 3]

```
flnk896s -cs 0x6000/0x6FFF,0x8000/0x8FFF,CRC16,0x7F
```

The areas of 0x6000/0x6FFF and 0x8000/0x8FFF are operated in the checksum by 16 bit cyclic redundancy check (CRC16). (Refer to Figure 6.2-6.)

Figure 6.2-6 Example 3 of Checksum operation



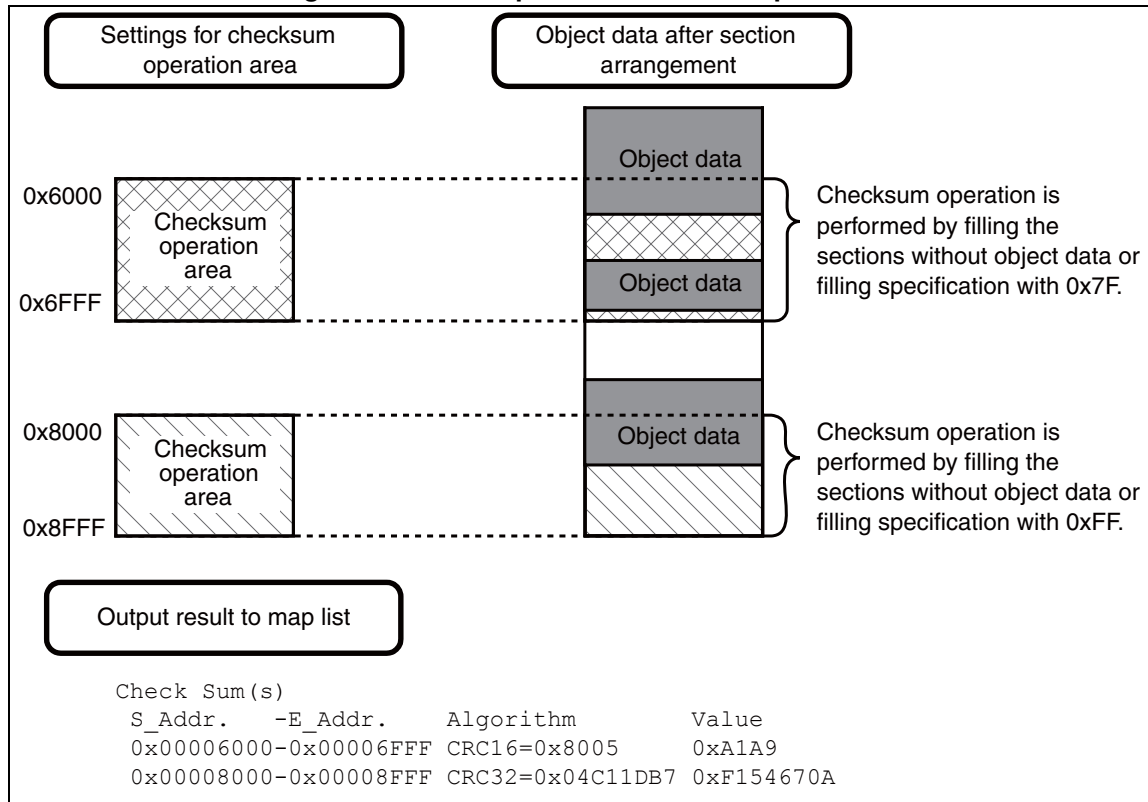
[Example 4]

```
flnk896s -cs 0x6000/0x6FFF,CRC16,0x7F -cs 0x8000/0x8FFF,CRC32,0xFF
```

The area of 0x6000/0x6FFF is operated in the checksum by 16 bit cyclic redundancy check (CRC16).

Moreover, the area of 0x8000/0x8FFF is operated in the checksum by 32 bit cyclic redundancy check (CRC32). (Refer to Figure 6.2-7.)

Figure 6.2-7 Example 4 of Checksum operation



[Example 5]

```
f1nk896s -cs 0x7000/0x8FFF,SUM16=1,0x7F -cs 0x8000/0x8FFF,SUM32=2,0xFF
```

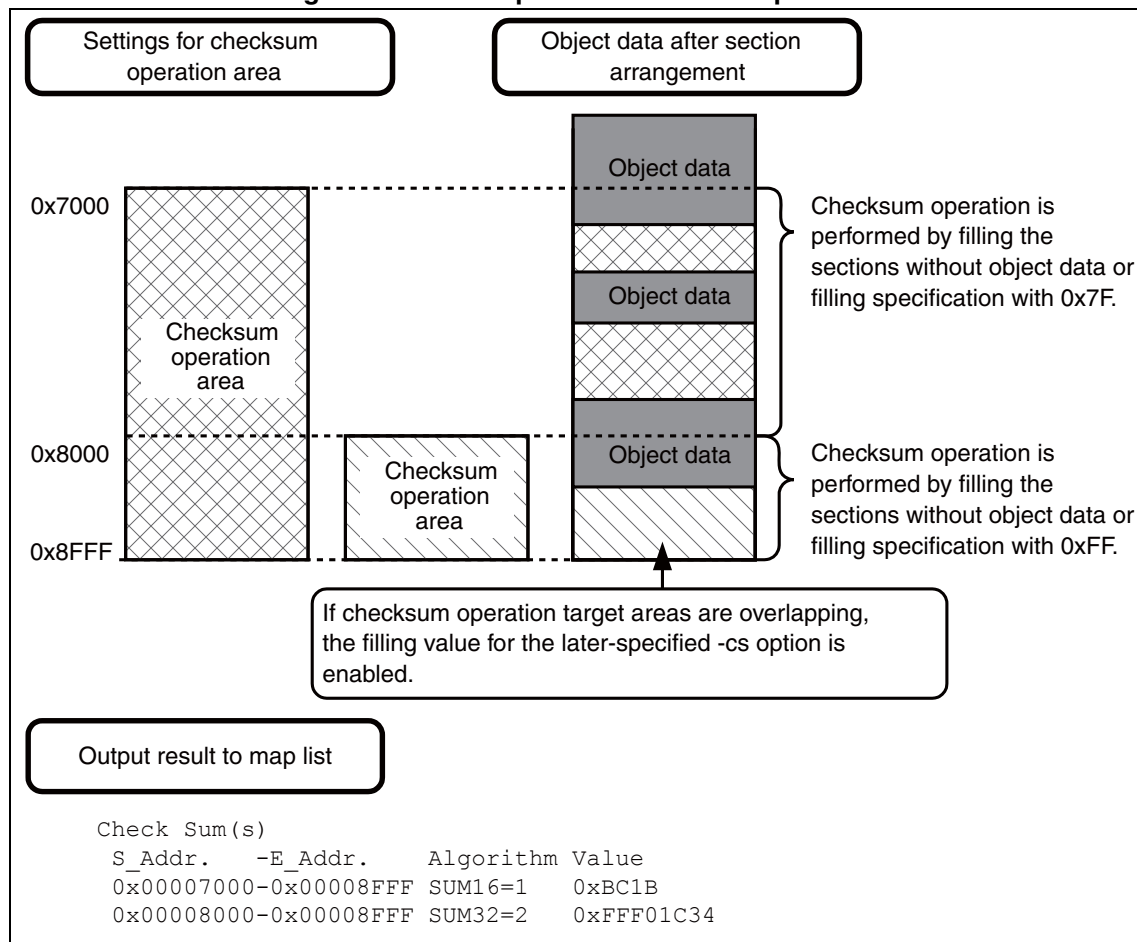
The area of 0x7000/0x8FFF is operated in the checksum by 16 bit simple addition.

The part in which the object data does not exist in the area of 0x7000/0x7FFF is filled with 0x7F.

Moreover, the area of 0x8000/0x8FFF is operated in the checksum by 32 bit simple addition.

The part in which the object data does not exist in the area of 0x8000/0x8FFF is filled with 0xFF.(Refer to Figure 6.2-8.)

Figure 6.2-8 Example 5 of Checksum operation



6.2.14 Warning Message Output Level Specification (-w)

The output level of warning messages is set. This option is used to inhibit warning messages of the linker completely or to check the operating state of the linker.

■ Warning Message Output Level Specification (-w)

[FORMAT]

-w < Numeric value >

[Parameters]

<Numeric value>

Specify 0, 1, or 2 as a warning level.

[Explanation]

Information to be obtained is controlled such as the inhibition of outputting warning level messages or the output of more detailed messages.

- 0..... Warning level messages are not output.
- 1..... Normal checking. (default)
- 2..... Messages of the level that can be normally ignored and those to report simply linker operations are also output.

For details, see "APPENDIX A ERROR MESSAGES OF THE LINKAGE KIT".

[Example]

```
flnk896s a1 a2 a3 -o a123.abs -w 2 -Xm
```

All messages are output.

6.2.15 ROM Area Specification (-ro)

Addressing for section allocation can be simplified by defining the ROM area to be used by a program.

Program size checking is also enabled.

■ ROM Area Specification (-ro)

[Format]

```
-ro < Area name > = < Start address > / < End address >  
[, < Area name > = < Start address > / < End address > ] ...
```

[Parameters]

<Area name>

Name to indicate the address area to be set

<Start address>

Start address of the address area to be set

<End address>

End address of the address area to be set

[Explanation]

The ROM areas are defined. As many areas as required can be defined.

Specify the start address and end address, then name the area.

The area name defined by this option is used in the section allocation option.

The definition of the -ro option alone does not affect linker operation. Be sure to use the area name defined here for the addressing parameter in the section allocation option.

[Example]

```
flnk896s *.obj -o ap.abs -ro RomA=0x8000/0x9FFF -sc code=RomA ...
```

Section allocation of the section name code to the address 0x8000 to 0x9FFF is specified.

The section is allocated starting with the address 0x8000. A warning message is output if the address 0x9FFF is exceeded.

6.2.16 RAM Area Specification (-ra)

By defining the RAM area to be used by a program, addressing for section allocation can be simplified.

The program size can be also checked.

■ RAM Area Specification (-ra)

[FORMAT]

```
-ra <Area name> = <Start address> / <End address>  
[ , <Area name> = <Start address> / <End address> ] ...
```

[Parameters]

<Area name>

Name to indicate the address area to be set

<Start address>

Start address of the address area to be set

<End address>

End address of the address area to be set

[Explanation]

The RAM areas are defined. As many areas as required can be defined.

Specify the start address and end address, then name the area.

The area name defined by this option is used in the section allocation option.

The definition of the -ra option alone does not affect linker operation. Be sure to use the area name defined here for the addressing parameter in the section allocation option.

[Example]

```
flnk896s *.obj -o ap.abs -ra RamD=0x0100/0x01FF -sc data=RamD ...
```

The section allocation of the section name data to the address 0x0100 to 0x01FF is specified.

The section is allocated starting with the address 0x0100, and a warning message is output if the address 0x01FF is exceeded.

6.2.17 Section Allocation Order/Address Specification (-sc)

This option specifies the start address and allocation order of the section allocation for the linker.

■ Section Allocation Order/Address Specification (-sc)

[Format]

```
-sc < Section name list > [ / < Content type > ]  
    [ = { < Address > | < Area name > } ] [ , ... ]
```

[Parameters]

<Section name list>

The wild card can be used to specify the section name, section group name, or list section name.

When specifying multiple names, link them with the + symbol.

<Content type>

code, data, stack, const, dir, dirconst, IO, dir, dirconst

<Address>

Start address of the allocation

<Area name>

Area name specified in the ROM/RAM specification option

[Explanation]

The order of section allocation and the allocation address are specified.

The order of section allocation follows the order described in the parameters.

Allocation starts with the address 0 if the address or area name is not specified.

If the @ mark is attached to the head of a section name, the address is specified on the ROM side of the ROM -> RAM transfer section. This section operates transferring data from ROM to RAM during execution.

Use double quotation marks to indicate a wild card. Do not use double quotation marks (") in an option file.

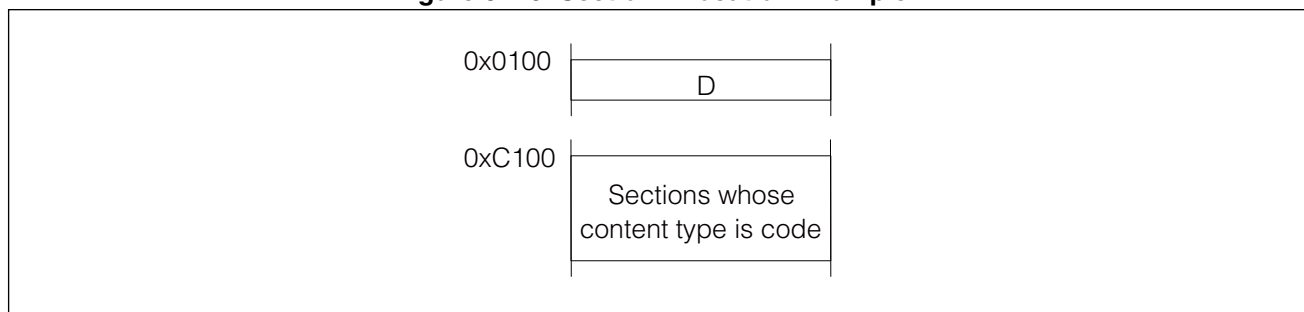
[Example 1]

```
f1nk896s *.obj -o ap.abs -sc "**/code"=0xC100,D=0x100 ...
```

Figure 6.2-9 shows the allocation example in this case.

The section whose content type is code is allocated starting with the address 0xC100 and section D is allocated starting with the address 0x0100.

Figure 6.2-9 Section Allocation Example 1



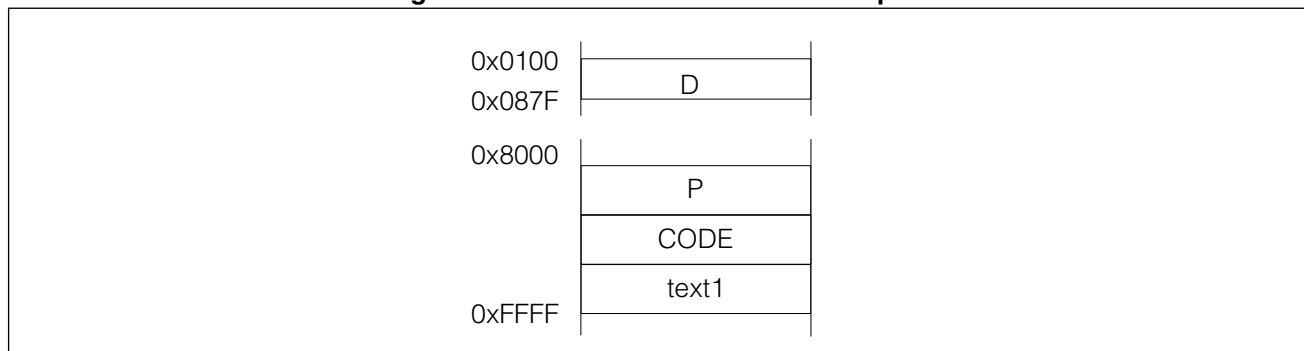
[Example 2]

```
f1nk896s *.obj -o ap.abs -ro ROM=0x8000/0xFFFF -ra RAM=0x0100/0x087F -sc P+code+text1=ROM,D=RAM
```

Figure 6.2-10 shows the allocation example in this case.

If section allocation is specified using the ROM/RAM option, the allocation end address can be checked.

Figure 6.2-10 Section Allocation Example 2



The wild card character that can be used in the <section name list> is only '*' and the following four description patterns are available.

Pattern		Example	Explanation
*		-sc */code	Match with all sections whose content type is code
	Match	ab_1, code_1, XXsect, etc	
	Mismatch	None	
*		-sc ab*/code	Match with all sections whose content type is code and whose first three characters are ab_
	Match	ab_1, ab_XX, ab_, etc	
	Mismatch	aab_XX, ab, etc	
*_		-sc *_1/code	Match with all sections whose content type is code and whose last two characters are 1
	Match	ab_1, XX_1, _1, etc	
	Mismatch	ab_11, _, etc	
*		-sc ab_*_1/code	Match with all sections whose content type is code, whose first three characters are ab_, and whose last two characters are _1
	Match	ab_XX_1, ab_1, etc	
	Mismatch	aab_XX_11, ab_1, etc	

6.2.18 Section Group Specification (-gr)

Multiple sections are linked to create a group according to the purpose of the user and a group name is given to the group.

By using this group name when specifying the section allocation, multiple sections can be handled as one group.

■ Section Group Specification (-gr)

[Format]

```
-gr < Group name > = < Section name list > [ / < Content type > ] [ , ... ]
```

[Parameters]

<Group name>

Generic name of multiple sections to be grouped

<Section name list>

Describes section names to be grouped.

The wild card can be used.

When specifying multiple names, link them with the + symbol.

<Content type>

code, data, stack, const, dir, dirconst, IO

[Explanation]

Sections to be grouped and the order of allocation of sections in the group are specified.

The order of section allocation follows the order described in the parameters.

Each group name must be a unique name which does not overlap with the section names and other group names.

A section that belongs to one group must not belong to another group.

When using the wild card, indicate it using double quotation marks. Do not use double quotation marks (") in an option file.

[Example 1]

```
flnk896s *.obj -o ap.abs -ro ROM=0x8000/0xFFFF -ra RAM=0x0100/0x57F ...  
-gr romG=P+CODE+text1 -sc romG=ROM,D=RAM ...
```

When using the grouping option, the whole group can be represented by its group name instead of specifying any number of section names.

[Example 2]

```
flnk896s *.obj -o ap.abs -gr cdgrp="*/code" -sc cdgrp=0x8000
```

All sections whose content type is code are linked to a group and the group name cdgrp is given to the group. Then the group is allocated to the address 0x8000 using the -sc option.



6.2.19 Register Bank Area Specification (-rg)

The register bank area used by programs is specified.

■ Register Bank Area Specification (-rg)

[Format]

```
-rg < Register bank number >  
    [ { , < Register bank number > | - < Register bank number > } ] ...
```

[Parameters]

<Register bank number>
Specify the register bank number to be set.

[Explanation]

The register bank area used by programs is specified.

When using general registers in a program, areas must be reserved using this option.

When specifying the consecutive register bank numbers, it is possible to specify them using a hyphen, for example, "3-8" (In this case, specify the register banks 3, 4, 5, 6, 7, and 8).

If this option is not specified, the linker does not reserve any register bank area.

Table 6.2-1 shows the areas reserved by the -rg option.

Table 6.2-1 Register Bank Numbers and Areas

No.	Reserved area	No.	Reserved area
0	0x0100 to 0x0107	16	0x0180 to 0x0187
1	0x0108 to 0x010F	17	0x0188 to 0x018F
2	0x0110 to 0x0117	18	0x0190 to 0x0197
3	0x0118 to 0x011F	19	0x0198 to 0x019F
4	0x0120 to 0x0127	20	0x01A0 to 0x01A7
5	0x0128 to 0x012F	21	0x01A8 to 0x01AF
6	0x0130 to 0x0137	22	0x01B0 to 0x01B7
7	0x0138 to 0x013F	23	0x01B8 to 0x01BF
8	0x0140 to 0x0147	24	0x01C0 to 0x01C7
9	0x0148 to 0x014F	25	0x01C8 to 0x01CF
10	0x0150 to 0x0157	26	0x01D0 to 0x01D7
11	0x0158 to 0x015F	27	0x01D8 to 0x01DF
12	0x0160 to 0x0167	28	0x01E0 to 0x01E7
13	0x0168 to 0x016F	29	0x01E8 to 0x01EF
14	0x0170 to 0x0177	30	0x01F0 to 0x01F7
15	0x0178 to 0x017F	31	0x01F8 to 0x01FF

[Example]

```
flnk896s *.obj -o ap.abs -rg 0,1,5-7...
```

The register bank areas 0, 1, 5, 6, and 7 are reserved.

6.2.20 Automatic Allocation Specification (-AL)

Automatic allocation of sections is specified.

■ Automatic Allocation Specification (-AL)

[Format]

```
-AL { 0 | 1 | 2 }
```

[Parameters]

<0>

No automatic allocation (default)

<1>

If an absolute section exists in the area, the order of allocation is changed appropriately to avoid overlapping with the section.

<2>

Whether to allocate to the ROM area or RAM area is determined based on the section attributes. Then allocation is carried out to free space of each area.

(link907a automatic allocation compatible)

[Explanation]

Automatic allocation of sections is specified.

- If the parameter is 1
If an absolute section exists when allocating sections to the areas specified by the -ra or -ro option, relocatable sections are allocated in such a way that allocation addresses do not overlap. In such cases, sections are allocated in descending order of the alignment value and size starting with the section of the largest alignment value and size. In this way, optimal allocation is implemented where free space is minimum.
- If the parameter is 2
Whether to allocate sections whose allocation is not specified by the -sc option to the ROM area or RAM area is determined based on the section attributes. Then such sections are allocated to free space of each area.

See also Section "5.6 Automatically Locating Sections".

[Example]

```
flnk896s -AL 1 -ro ROM=0x8000/0x8FFF -sc code1+code2+code3=ROM ...
```

Each section is given as follows.

- code1 : relocatable, size=0x18
- code2 : relocatable, size =0x10
- code3 : relocatable, size =0x30
- AbsSec : absolute, address range=0x8010 to 0x8017

The following figure shows the link map for this case.

S_Addr. -E_Addr	Size	Section Type	AL	Sec
00008000-0000800F	00000010	CODE P	R-XI	02 REL code2
00008010-00008017	00000008	CODE N	R-XI	00 ABS AbsSec
00008018-00008047	00000030	CODE P	R-XI	02 REL code3
00008048-0000805F	00000018	CODE P	R-XI	02 REL code1

Note:

Even if -AL 1 is specified, automatic allocation is not carried out in the following cases.

- No area is set. (-ra or -ro is not set)
- No area name is used in addressing of -sc.
- No absolute section exists to be allocated in the specified area.

Even if -AL 2 is specified, errors occur and no link processing is performed in the following cases.

- No area is set. (-ra or -ro is not set)

If -w 2 is specified and the automatic allocation function works, a message is output.

6.2.21 Retrieval Library File Specification (-l)

Libraries to be retrieved other than the default library are specified. If multiple library files are available, they are specified in order of retrieval.

■ Retrieval Library File Specification (-l)

[Format]

`-l < Library file name > [ , ... ]`

[Parameters]

<Library file name>

Describe the names of the library files to be retrieved. Library file names with path names are allowed. The wild card can be also used.

[Explanation]

Library files are retrieved in the specified order.

Libraries specified here are retrieved before the default library.

If the library files are specified without path names, directories are retrieved in the following order: the directory specified by the -L option, the directory specified in the environmental variable LIB896, and the system library path derived from the environmental variable FETOOL.

The current directory is not retrieved. If you want to retrieve the current directory, specify either the -L option or a period (.) in the environmental variable LIB896.

By specifying the -l option after the -nl option, the -nl option can be canceled.

Indicate a wild card with double quotation marks. Do not use double quotation marks (") in an option file.

[Example]

```
flnk896s *.obj -o ap.abs -l ..\lib\com.lib,libu
flnk896s *.obj -o ap.abs -l "p*.lib"
```

All library files whose file name's first character is "p" are retrieved.

6.2.22 Library Retrieval Path Specification (-L)

The path name for retrieving the library file is specified.

■ Library Retrieval Path Specification (-L)

[Format]

-L <Library path name> [, ...]

[Parameters]

<Library path name>

Name of the path storing the library file

[Explanation]

In which directory the library file specified by the -l option exists is instructed for the linker.

Normally, specify the environmental variable LIB896 so that this option need not to be specified.

The C libraries attached to the C compiler are stored in the path specified by LIB896. To manage libraries specially created by the user, use the -L option to manage them in another directory.

If multiple paths are specified, they are retrieved in order of specification.

The library file is first retrieved in the path specified here, then in the environmental variable LIB896 and system library path derived from the environmental variable FETOOL.

If a library file is specified with its path name, only the specified path is retrieved.

If the -nl option exists, library retrieval is not performed, thus the -L option is canceled.

[Example]

```
flnk896s *.obj -o ap.abs -L C:\usr\usr\lib -l com.lib,libu
```

6.2.23 Library Specification for Each Symbol (-el)

The library file to be used for the resolution of external reference symbols can be specified.

■ Library Specification for Each Symbol (-el)

[Format]

```
-el <Symbol name list> = <Library file name> [ , ... ]
```

[Parameters]

<Symbol name list>

Describe the external reference symbol names.

When specifying multiple symbols, separate them with /.

<Library file name>

Library file name to be retrieved.

Library file names with path names can be also specified. The wild card cannot be used.

[Explanation]

The library file to be used for the resolution of the external reference symbol value specified in this option is specified.

This option is used when a module containing the same external definition symbol name exists in multiple libraries and the linker links undesirable modules in the standard library retrieval order.

Libraries are not often created appropriately so it is inevitable that this function will be applied when using multiple libraries. However, since using this function may cause unexpected problems.

Examine whether library files can be recreated.

The retrieval directories when a library file name is specified without its path name are the same as those for the -l option.

[Example]

```
flnk896s *.obj -o sp.abs -L C:\usr\usrlib -l libu,sublib -el sym1=sublib
```

6.2.24 Library Retrieval Inhibit Specification (-nl)

The inhibition of the library file retrieval is instructed.

■ Library Retrieval Inhibit Specification (-nl)

[Format]

-nl

[Parameters]

None

[Explanation]

The inhibition of the library file retrieval is specified.

[Example 1]

```
flnk896s -L C:\usr\usr\lib -l libu,sublib *.obj -o ap.abs -nl
```

The -L, -l, and -el options previously specified are canceled and library retrieval including the default library is inhibited.

[Example 2]

```
flnk896s -l lib1 *.obj -o ap.abs -nl -l lib2
```

lib1.lib specified once is canceled and retrieval of lib2.lib is instructed.

As shown in this example, if the -nl option is specified between multiple -l options, specified before -nl are all canceled. However, the -L option specification, -el option specification, -nd option specification, and default library retrieval are restored to the settings before the -nl specification.

Thus, in this example, lib2.lib and the default library are retrieved.

6.2.25 Default Library Retrieval Inhibit Specification (-nd)

The default library is a library file presumed to be used by the C compiler and its library file names are set in the object file.

This option instructs not to retrieve the default library file.

■ Default Library Retrieval Inhibit Specification (-nd)

[Format]

-nd

[Parameters]

None

[Explanation]

The specification of the default library is canceled, therefore, it is not retrieved.

[Example]

```
flnk896s -L C:\usr\usrlib -l libu,sublib *.obj -o ap.abs -nd
```

Only the libraries specified by the -l option are retrieved and the default library is not retrieved.

6.2.26 Entry Address Specification (-e)

The start address of a user program is specified using an external definition symbol.

■ Entry Address Specification (-e)

[Format]

-e <Symbol name >

[Parameters]

<Symbol name>

Symbol name of the entry point

Only external definition symbols can be used.

[Explanation]

The start address of a user program is changed to that specified by the external definition symbol.

The start address can be specified using the .end pseudo-instruction of the assembler.

The entry point is set as the initial value of the PC (program counter) when starting execution of the simulator debugger.

[Example]

```
flnk896s *.obj -o ap.abs -e ProgStart
```

6.2.27 Dummy Setting of External Symbol Values (-df)

Undefined symbol values of a user program are forced to be defined.

■ Dummy Setting of External Symbol Values (-df)

[Format]

`-df < Symbol name > = { < Numeric value > | < External definition symbol name > }`

[Parameters]

<Symbol name>

Symbol name of an external reference symbol

<Numeric value>

Value to be defined

<External definition symbol name>

External symbol name whose value is defined

[Explanation]

Values of undefined external reference symbols are forced to be defined.

The linker creates object data using this value for the resolution of relocation. Symbol information in the absolute format load module file to be output is not affected.

If loaded using a debugger, the symbol name specified here remains undefined.

[Example]

```
flnk896s -L \usr\usr\lib -l libu,sublib *.obj -o ap.abs -df Sym1=100
```

If Sym1 is not defined, 100 is set as its value.

6.2.28 Target CPU Specification (-cpu)

The target CPU is specified.

The target CPU of programs to be linked is specified using the MB number.

■ Target CPU Specification (-cpu)

[Format]

`-cpu <MB number>`

[Parameters]

<MB number>

MB number of the target CPU

[Explanation]

The target CPU of programs to be linked is specified using the MB number.

[Example]

```
flnk896s *.obj -o ap.abs -cpu MB89051
flnk896s *.obj -o ap.abs -cpu MB89P133A
```

Note:

When executing link processing, the target CPU must be specified using this option. This option cannot be omitted.

6.2.29 Specifying CPU Information File (-cif)

This specifies to determine the selecting of the target CPU information from the CPU information file.

■ CPU Information File Specification (-cif)

[Format]

External definition symbol name

-cif < CPU information file name >

[Parameters]

< CPU information file name >

CPU information file name of the target

[Explanation]

This determines the getting of the target CPU ROM/RAM region information from the CPU information file.

[Example]

```
flnk896s * .obj -o ap.abs -cpu MB89501 -cif C:\Softune\lib\896\cpu_info\MB89501.csv
```

Note:

SOFTUNE Tools get CPU information by referencing the CPU information file. Reference to a CPU information file different between the related tools may cause an error to the program to be created. The CPU information file that comes standard with SOFTUNE Tools is located at:

Installation place directory\lib\896\896.csv

When installing the compiler assembler packs in a different directory and using the compiler, assembler and linkage kit instead of SOFTUNE workbench, specify -cif so that each tool can reference the same CPU information file.

6.2.30 Specification for inhibiting Check for Presence of Debug Data (-NCI0302LIB)

This option inhibits check for presence of debug data in the module extracted from the library file.

■ Specification for Inhibiting Check for Presence of Debug Data (-NCI0302LIB)

[Format]

- NCI0302LIB

[Parameters]

None

[Explanation]

When debug data output (-g) and warning level 2 (-w2) are specified to operate the linker, the linker outputs the following information to the module that has no debug data.

I0302L: Debug information not exist (file name)

When this option is specified, the linker does not output the above information message to the module extracted from the library file.

[Example]

```
flnk896s -cpu MB89051 -g -w 2 test .obj -l lib896c .lib  
*** I0302L:Debug information not exist(C:\Softune\lib\896\lib896c.lib)  
*** I0302L:Debug information not exist(C:\Softune\lib\896\lib896c.lib)
```

•
•
•

The linker outputs the information (I0302L).

```
flnk896s -cpu MB89051 -g -w 2 test .obj -l lib896c .lib -NCI0302LIB
```

The linker does not output the information (I0302L).



6.2.31 Function that Automatically Sets Internal ROM/RAM Areas (-set_rora)

Refers to CPU information file to set the information regarding the internal ROM/RAM areas of the targeted CPU.

■ Function that Automatically Sets Internal ROM/RAM Area (-set_rora)

[Format]

-set_rora

[Parameters]

None

[Explanation]

Refers to CPU information file to set the information regarding the internal ROM/RAM area of the targeted CPU.

The linker, when this option is specified, refers to CPU information file and automatically sets the internal ROM/RAM area of the appropriate chip.

The linker sets the following names for the ROM/RAM areas.

- ROM Areas: _ROM_*_
 - Numbers are entered at the asterisk (*) in order from the lower address region starting from 1. If there is only 1 area, the number will be "_ROM_1_".
- RAM Areas: _RAM_*_
 - Numbers are entered at the asterisk (*) in order from the lower address region starting from 1. If there is only 1 area, the number will be "_RAM_1_".

These names are used by the -sc options.

[Example]

```
flnk896s *.obj -o ap.abs -cpu MB89121 -set_rora
```

6.2.32 Specifies to Prevent the Internal ROM/RAM Area from being Automatically Set (-Xset_rora)

Prevents the information from being set regarding the internal ROM/RAM area of the targeted CPU that is referenced to the CPU information file.

■ Specifies to Prevent the Internal ROM/RAM Area from being Automatically Set (-Xset_rora)

[Format]

`-Xset_rora`

[Parameters]

None

[Explanation]

Prevents the information from being set regarding the internal ROM/RAM area of the targeted CPU that is referenced to the CPU information file.

[Example]

```
flnk896s *.obj -o ap.abs -cpu MB89121 -Xset_rora
```

6.2.33 User-specified-area Check Specification (-check_rora)

You can check whether the memory map has been changed, by changing only the specified MB number.

The **-check_rora** option checks whether the specified ROM and RAM areas (**-ro** and **-ra** options) correspond to the actual addresses of the internal ROM and internal RAM. Use this option when single-chip mode is used.

■ User-specified-area Check Specification (-check_rora)

[Format]

```
-check_rora
```

[Parameters]

None

[Explanation]

A check is made on whether the specified ROM and RAM areas (**-ro** and **-ra** options) exceed the internal ROM and internal RAM.

The following warnings are output when the areas specified for the **-ro** and **-ra** options are not within the internal ROM and internal RAM:

W1368L: The area specified for the **-ro** option is outside the internal-ROM area (area name)

W1369L: The area specified for the **-ra** option is outside the internal-RAM area (area name)

When the **-check_rora** option is specified in using single-chip mode, a check is made on whether this option and the MB number correspond to the actual addresses of the internal ROM and internal RAM of the product concerned. So, for instance, when a program is ported to a different product, you can check whether the memory map has been changed, by changing only the specified MB number.

Also, when the **-check_locate** option is specified together with the **-check_rora** option, you can check whether the program is within the internal ROM and internal RAM.

[Example]

```
flnk896s -cpu MB89051 -check_rora -ro ROM = 0x8000/0xFFFF
-ra RAM=0x0080/0x087F ...
```

-> The areas specified for the **-ro** and **-ra** options are within the ranges of the internal ROM and internal RAM; so no warning is output.

```
flnk896s -cpu MB89135L -check_rora -ro ROM = 0x8000/0xFFFF
-ra RAM=0x0080/0x087F ...
```

W1368L: The area specified for the **-ro** option is outside the internal-ROM area (ROM)

W1369L: The area specified for the **-ra** option is outside the internal-RAM area (RAM)

-> The underlined areas specified are not within the ranges of the internal ROM and internal RAM; so the warnings are output.

```
flnk896s -cpu MB89F051 -check_rora -ro ROM = 0x8000/0xFFFF
-ra RAM1=0x0080/0x087F ...
```

-> The areas specified for the **-ro** and **-ra** options are within the ranges of the internal ROM and internal RAM; so no warning is output.

Notes:

- A warning is output when the `-check_rora` option is specified even if warning output suppression (`-w 0`) is specified.
 - The `-check_rora` option is only valid when creating absolute format load modules. The `-check_rora` option is ignored when creating relocatable load modules.
 - The `-check_rora` option uses internal-ROM data and internal-RAM data contained in the CPU data file. So, no warning is output when the CPU data file contains no data concerned. Specify the correct MB number for the `-cpu` option.
-

6.2.34 User-specified-area Check Suppression Specification (-Xcheck_rora)

The -Xcheck_rora option suppresses the check on the specified ROM area and the specified RAM area (-ro and -ra options) and on the internal-ROM area and the internal-RAM area.

Use the -Xcheck_rora option when you want to cancel the -check_rora option.

■ User-specified-area Check Suppression Specification (-Xcheck_rora)

[Format]

`-Xcheck_rora`

[Parameters]

None

[Explanation]

The -Xcheck_rora option suppresses the check on the specified ROM area and the specified RAM area (-ro and -ra options) and on the addresses of the internal ROM and the internal RAM.

Use the -Xcheck_rora option when you want to cancel the -check_rora option.

[Example]

```
flnk896s -cpu MB89135L -check_rora -ro ROM = 0x8000/0xFFFF  
-ra RAM = 0x0080/0x87F ...
```

W1368L: The area specified for the -ro option is outside the internal-ROM area (ROM)

W1369L: The area specified for the -ra option is outside the internal-RAM area (RAM)

-> The underlined areas specified are not within the ranges of the internal ROM and internal RAM; so the warnings are output.

```
flnk896s -cpu MB89135L -check_rora -ro ROM = 0x8000/0xFFFF  
-ra RAM1 = 0x0080/0x087F -Xcheck_rora
```

-> Although the underlined areas specified are not within the ranges of the internal ROM and internal RAM, no warning is output because the check is suppressed using the -Xcheck_rora option.

6.2.35 Section-placed-area Check Specification (-check_locate)

A check is made to see that no section is specified outside the memory area. The -check_locate option checks the section-placed address, based on the specified ROM area and the specified RAM area (-ro and -ra options) or based on internal ROM data and internal RAM data in the CPU data file; and outputs a warning if a section is specified outside the area.

■ Section-placed-area Check Specification (-check_locate)

[Format]

`-check_locate`

[Parameters]

None

[Explanation]

The -check_locate option checks the section-placed address, based on the specified ROM area and the specified RAM area (-ro and -ra options) or based on internal ROM data and internal RAM data in the CPU data file; and outputs a warning if a section is specified outside the area.

W1370L: The section is placed outside the ROM area (section name)

W1371L: The section is placed outside the RAM area (section name)

W1372L: The section is placed outside the RAM area or the I/O area (section name)

W1373L: The section is placed outside the I/O area (section name)

The section types and the check areas are shown below.

Section type	Area to be checked	
	The -check_rora option is specified.	The -check_rora option is not specified.
Section that should be placed within the ROM area		
CODE	A warning is output when a section is placed outside the area specified for -ro and outside the internal-ROM area.	A warning is output when a section is placed outside the area specified for -ro.
CONST		
DIRCONST		
ROM/RAM Send-from section		
Section that should be placed within the RAM area		
STACK	A warning is output when a section is placed outside the area specified for -ra and outside the internal-RAM area.	A warning is output when a section is placed outside the area specified for -ra.
DIR		
ROM/RAM Send-to section		
Section that should be placed within the RAM area or the I/O area		
DATA	A warning is output when a section is placed outside the area specified for -ra and outside the internal-RAM area or the internal-I/O area.	A warning is output when a section is placed outside the area specified for -ra.
Section that should be placed within the I/O area		
IO	A warning is output when a section is placed outside the internal-I/O area.	A warning is output when a section is placed outside the internal-I/O area.

In single-chip mode, when the -check_locate option is specified together with the -check_rora option, you can check whether the program is placed outside the internal-memory area.

Also, even not in single-chip mode, when the -check_locate option is specified together with the ROM area specification and RAM area specification (-ro and -ra options), you can check whether the program is placed outside the memory area.

[Example]

When DATA_A, DATA_B, DATA_C (these are of section type DATA), CODE_D, CODE_E, CODE_F (these are of section type CODE), and STACK_G (this is of section type STACK) are placed in the following memory map, the following check will be made:

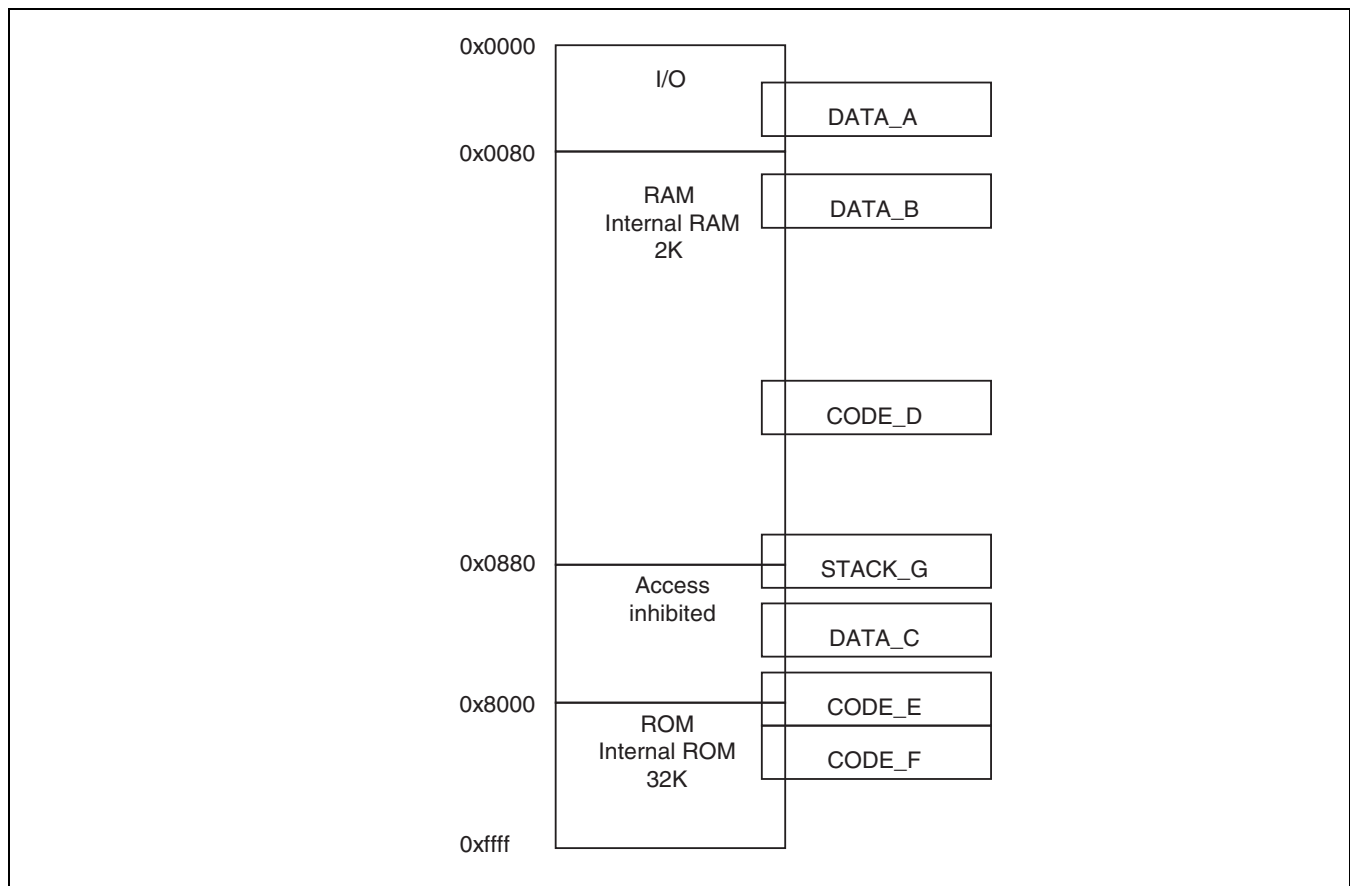
```
flnk896s -cpu MB89051 -check_locate -ro ROM = 0x8000/0xFFFF
-ra RAM=0x0080/0x087F ...
```

W1371L: The section is placed outside the RAM area (STACK_G).

W1372L: The section is placed outside the RAM area or the I/O area (DATA_C).

W1370L: The section is placed outside the ROM area (CODE_D).

W1370L: The section is placed outside the ROM area (CODE_E).



DATA_A is placed within the internal-I/O area; so no warning is output.

DATA_B is placed within the area specified for -ra; so no warning is output.

DATA_C is placed outside the area specified for -ra; so the warning is output.

CODE_D is placed outside the area specified for -ro; so the warning is output.

CODE_E is placed within the area specified for -ro; so no warning is output.

CODE_F is placed outside the area specified for -ro; so the warning is output.

STACK_G is placed outside the area specified for -ra; so the warning is output.



Notes:

- A warning is output when the `-check_locate` option is specified even if warning output suppression (`-w 0`) is specified.
 - The `-check_locate` option is only valid when creating absolute format load modules.
The `-check_locate` option is ignored when creating relocatable load modules.
 - When using the `-check_locate` option, you must specify the areas for the `-ro` and `-ra` options in advance.
 - The `-check_locate` option also checks ABS-attribute sections not to be processed by the linker.
-

6.2.36 Section-placed-area Check Suppression Specification (-Xcheck_locate)

The -Xcheck_locate option suppresses the section-placed-area check.
Use the -Xcheck_locate option when you want to cancel the -check_locate option.

■ Section-placed-area Check Suppression Specification (-Xcheck_locate)

[Format]

-Xcheck_locate

[Parameters]

None

[Explanation]

The -Xcheck_locate option suppresses the section-placed-area check.

Use the -Xcheck_locate option when you want to cancel the -check_locate option.

6.2.37 Specifying Check User-unspecified Section (-check_section)

Check for the presence of a section for which the user does not specify layout.

■ Specifying Check User-unspecified Section (-check_section)

[Format]

`-check_section`

[Parameters]

None

[Explanation]

Check for the presence of a section for which the user does not specify layout.

If this option is specified, the linker will output the following warning against the section not specified in the specify section layout (-sc) option.

W1375L: This is the section (section name) for which layout is not specified in the -sc option.

If this option is not specified, the linker makes no check.

[Example]

The following operation is performed when the objects having the CODE, INIT, CONST, DATA and STACK sections are linked.

```
flnk896s -cpu MB89051 -check_section
-ro ROM=0x8000/0xFFFF
-ra RAM=0x0080/0x087F
-sc CODE+CONST=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2...
```

W1375L: This is the section (INIT) for which layout is not specified in the -sc option.

-> Layout is not specified for the INIT section.

Since the -check_section option is specified, the warning is output to the INIT section.

```
flnk896s -cpu MB89051 -check_section
-ro ROM=0x8000/0xFFFF
-ra RAM=0x0080/0x087F
-gr ROMGROUP=CODE+CONST -sc DATA=RAM1
-sc STACK=RAM2 object1 object2...
```

W1375L: This is the section (CODE) for which layout is not specified in the -sc option.

W1375L: This is the section (INIT) for which layout is not specified in the -sc option.

W1375L: This is the section (CONST) for which layout is not specified in the -sc option.

-> The CODE and CONST sections are grouped in the -gr option.

Since layout is not specified for these sections in the -sc option, however, the warning is output to these sections.

```
flnk896s -cpu MB89051-check_section  
-ro ROM=0x8000/0xFFFF  
-ra RAM=0x0080/0x087F  
-sc *=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2...
```

-> Since the section name is specified by the wild cards, layout is specified for all the sections except the DATA and STACK sections.

Since layout is specified for all the sections, no warning is output.

Notes:

1. No check is made on the sections for which the layout address is specified in the source program.
 2. If layout is not specified for the sections in the -sc option even when section group is specified in the -gr option, the warning is output.
 3. If the wild cards are used to specify the section name in the -sc option, layout is specified for all the sections that match the wild cards.
Therefore, no warning is output to the sections that match the wild cards.
Take care when using the wild cards to specify the section name.
-

6.2.38 Specifying Inhibit Check for Presence of User-unspecified Section (-Xcheck_section)

Inhibit the check for the presence of a section for which the user does not specify layout. This option is used to cancel the -check_section option.

■ Specifying Inhibit Check for Presence of User-unspecified Section (-Xcheck_section)

[Format]

`-Xcheck_section`

[Parameters]

None

[Explanation]

Inhibit the check for the presence of a section for which the user does not specify layout. This option is used to cancel the -check_section option.

This option is default.

[Example]

The following operation is performed when the objects having the CODE, INIT, CONST, DATA and STACK sections are linked.

```
flnk896s -cpu MB89051 -check_section
      -ro ROM=0x8000/0xFFFF
      -ra RAM=0x0080/0x087F
      -sc CODE+CONST=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2 ...
```

->Layout is not specified for the INIT section.

Since the check is inhibited in the -Xcheck_section option, no warning is output.

```
flnk896s -cpu MB89051 -Xcheck_section
      -ro ROM=0x8000/0xFFFF
      -ra RAM=0x0080/0x087F
      -sc CODE+CONST=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2 ...
```

->Layout is not specified for the INIT section.

Since no check is specified in the -check_section option, however, no warning is output.

This is the default operation of the linker.

6.2.39 Object Mixing Check Specification (-objmixchk)

This option checks whether F²MC-8L objects and F²MC-8FX objects are mixed or not. If these are mixed, an error occurs.

■ Object Mixing Check Specification (-objmixchk)

[Format]

`-objmixchk`

[Parameters]

None

[Explanation]

This option checks whether F²MC-8L objects and F²MC-8FX objects are mixed or not. The mix check of objects is performed at default.

If the objects are mixed, the error (E4312L: Uncompatible cpu type module (file name)) is issued.

[Example]

```
flnk896s -cpu MB89051 -o loadmodule.abs obj_1.obj obj_2.obj obj_3.obj ...  
flnk896s -cpu MB89051 -objmixchk -o loadmodule.abs obj_1.obj obj_2.obj obj_3.obj ...
```

If obj_1.obj and obj_3.obj are the F²MC-8FX objects, the following errors are issued.

*** E4312L: Uncompatible cpu type module(obj_1.obj)

*** E4312L: Uncompatible cpu type module(obj_3.obj)

6.2.40 Object Mixing Check Inhibit Specification (-Xobjmixchk)

This option inhibits the mix check of the F²MC-8L objects and F²MC-8FX objects. Specify this option when mixing the objects.

■ Object Mixing Check Inhibit Specification (-Xobjmixchk)

[Format]

`-Xobjmixchk`

[Parameters]

None

[Explanation]

This option inhibits the mix check of the F²MC-8L objects and F²MC-8FX objects. Specify this option when mixing the objects.

[Example]

```
flnk896s -cpu MB89051 -objmixchk -o loadmodule.abs obj_1.obj obj_2.obj obj_3.obj
-Xobjmixchk...
flnk896s -Xobjmixchk -cpu MB89051 -objmixchk -o loadmodule.abs obj_1.obj obj_2.obj obj_3.obj ...
```

If the F²MC-8L objects and F²MC-8FX objects are mixed, processing is performed without issuing any error.

6.2.41 Relative Assemble List Input Directory Specification (-alin)

The directory in which a relative assemble list file is stored is specified.
If this option is not specified, the directory is the same layout as the object module.

■ Relative Assemble List Input Directory Specification (-alin)

[Format]

-alin < Path name >

[Parameters]

<Path name>

Directory in which a relative assemble list file is stored

[Explanation]

This is an option to be used when an absolute assemble list file should be output.

The directory in which a relative assemble list file is stored is specified.

If this option is not specified, the directory is the same layout as the object module.

If a relative assemble list file with its path name is specified using the -alf option, the path specified by the -alf option is prioritized.

[Example]

```
flnk896s *.obj -o ap.abs -alin d:\F2MC8L -alf swctrl.lst,mstdef.lst
```

```
flnk896s *.obj -o ap.abs -alalf d:\F2MC8L\swctrl.lst,d:\F2MC8L\mstdef.lst
```

The above two examples share the same meaning.

6.2.42 Absolute Assemble List Output Directory Specification (-alout)

The directory which outputs an absolute assemble list file is specified.

■ Absolute Assemble List Output Directory Specification (-alout)

[Format]

-alout < Path name >

[Parameters]

<Path name>

Directory which outputs an absolute assemble list file

[Explanation]

The directory which outputs an absolute assemble list file is specified.

If this option is not specified, the current directory is specified.

[Example]

```
flnk896s *.obj -o ap.abs -alin d:\F2MC8L -alf swctrl.lst,mstdef.lst -alout d:\F2MC8L\als
```

6.2.43 Absolute Assemble List Output Specification (-als)

The output of absolute assemble list files is specified.
This is an output instruction for all object modules.

■ Absolute Assemble List Output Specification (-als)

[Format]

-als

[Parameters]

None

[Explanation]

All modules are instructed to create absolute assemble lists.

If this option is not specified, absolute assemble lists are not created.

-alsf and -Xals specified before are canceled.

[Example]

```
flnk896s *.obj -o ap.abs -als
```

6.2.44 Absolute Assemble List Output Module Specification (-alsf)

Modules to output absolute assemble list files are specified.
Selective output for object modules is instructed.

■ Absolute Assemble List Output Module Specification (-alsf)

[Format]

```
-alsf < Relative assemble list file name > [ , ... ]
```

[Parameters]

<Relative assemble list file name>

Name of the relative assemble list file which is the source of creating the absolute assemble list

The wild card can be used when the file name is specified.

[Explanation]

Modules to create the absolute assemble lists are selected.

The relative assemble list file names are used to specify the modules. If the extension is omitted, ".lst" is assumed.

Any module which is not specified does not create any absolute assemble list.

This option can be specified in divided multiple parts.

The previously specified -als and -Xals are canceled.

Since the store path of a relative assemble list can be specified in the -alin option, the path specification can be omitted when this option is specified.

[Example]

```
flnk896s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst
flnk896s *.obj -o ap.abs -alsf swctrl.lst -alsf mstdef.lst
flnk896s *.obj -o ap.abs -alsf swctrl -alsf mstdef
```

The above three examples share the same meaning.

6.2.45 Absolute Assemble List Output Inhibit Specification (-Xals)

The inhibition of creating the absolute assemble lists is instructed for all modules.

■ Absolute Assemble List Output Inhibit Specification (-Xals)

[Format]

-Xals	(Default)
-------	-------------

[Parameters]

None

[Explanation]

The inhibition of creation of absolute assemble lists is instructed for all modules.

This option is a default option and is used to cancel the previously specified -als and -alsf.

[Example]

```
flnk896s *.obj -o ap.abs -alf sectrl.lst,mstdef.lst -Xals
```



6.2.46 ROM/RAM and ARRAY Lists Output Specification (-alr)

The ROM/RAM list and ARRAY lists output is specified.

The output of debug information when compiling or assembling is required.

■ ROM/RAM and ARRAY Lists Output Specification (-alr)

[Format]

-alr

[Parameters]

None

[Explanation]

The ROM/RAM and ARRAY lists are added all absolute assemble lists.

When using this option, the -als option can be omitted.

The previously specified -alrf and -Xalr are canceled.

To output the ROM/RAM and ARRAY lists, specify the debug information output option (-g) when compiling, assembling, and linking.

[Example]

```
flnk896s *.obj -o ap.abs -als -alr -g
```

```
flnk896s *.obj -o ap.abs -alr -g
```

The above two examples share the same meaning.

6.2.47 ROM/RAM and ARRAY Lists Output Module Specification (-alrf)

Modules to output the ROM/RAM and ARRAY lists are specified.
The output of debug information when compiling or assembling is required.

■ ROM/RAM and ARRAY Lists Output Module Specification (-alrf)

[Format]

-alrf < Relative assemble list file name > [, ...]

[Parameters]

<Relative assemble list file name>

Specify the modules to output the ROM/RAM and ARRAY lists using the relative assemble list file names.

The wild card can be used to specify the file name.

[Explanation]

Modules to output the absolute assemble lists with the added ROM/RAM and ARRAY lists are selected.

Modules are specified using the relative assemble list file names. If the extension is omitted, ".lst" is assumed.

Any module which is not specified does not create any ROM/RAM and ARRAY lists.

If this option is used, the -alsf option can be omitted.

This option can be specified in divided multiple parts.

Previously specified -alr and -Xalr are canceled.

[Example]

```
flnk896s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst -alrf swctrl.lst,mstdef.lst
flnk896s *.obj -o ap.abs -alsf swctrl.lst -alrf mstdef.lst,mstdef.lst
flnk896s *.obj -o ap.abs -alrf swctrl -alrf mstdef
```

The above three examples share the same meaning.

6.2.48 ROM/RAM and ARRAY Lists Output Inhibit Specification (-Xalr)

The inhibition of adding the ROM/RAM and ARRAY lists is specified for all absolute assemble lists.

■ ROM/RAM and ARRAY Lists Output Inhibit Specification (-Xalr)

[Format]

-Xalr	(Default)
-------	-------------

[Parameters]

None

[Explanation]

The inhibition of adding the ROM/RAM and ARRAY lists is specified for all absolute assemble lists.

This option is a default option, therefore, it is not necessary to specify it.

The previously specified -alr and -alrf are canceled.

[Example]

```
flnk896s *.obj -o ap.abs -als -alr -Xalr
```

```
flnk896s *.obj -o ap.abs -als -Xalr
```

The above two examples share the same meaning.

6.2.49 ROM/RAM and ARRAY Lists Symbol and Address Display Position Specification (-na, -an)

The display positions of the symbols and addresses of the ROM/RAM and ARRAY lists are specified.

If -na is specified, symbols (NAME) are output first, then addresses (ADDRESS).

If -an is specified, addresses (ADDRESS) are output first, then symbols (NAME).

■ ROM/RAM and ARRAY Lists Symbol and Address Display Position Specification (-na, -an)

[Format]

-na	(Default)
-----	-------------

[Parameters]

None

[Explanation]

Symbols and addresses of the ROM/RAM and ARRAY lists are output in order of NAME and ADDRESS.

Symbols are output in alphabetical order.

This option is valid only for modules for which the output of the ROM/RAM and ARRAY lists is specified.

This option is a default option and is specified to cancel the -an option.

[Example]

```
flnk896s *.obj -o ap.abs -alr -na
```

```
flnk896s *.obj -o ap.abs -alr
```

The above two examples share the same meaning.

[Format]

-an

[Parameters]

None

[Explanation]

Symbols and addresses of the ROM/RAM and ARRAY lists are output in order of ADDRESS and NAME.

Symbols are output in order of address.

This option is used to change the -na option (default).

This option is valid only for modules for which the output of the ROM/RAM and ARRAY lists is specified.

[Example]

```
flnk896s *.obj -o ap.abs -alr -na -an
```

```
flnk896s *.obj -o ap.abs -alr -an
```

The above two examples share the same meaning.

6.2.50 External Symbol Cross-reference Information List Output Specification (-xl)

The output of the external symbol cross-reference information list file is specified.

■ External Symbol Cross-reference Information List Output Specification (-xl)

[Format]

-xl

[Parameters]

None

[Explanation]

The creation of a external symbol cross-reference information list file is instructed.

If this option is not specified, no external symbol cross-reference information list file is created.

[Example]

```
flnk896s *.obj -o ap.abs -xl
```

6.2.51 External Symbol Cross-reference Information List File Name Specification (-xlf)

This option is used to change the output destination directory or file name of the external symbol cross-reference information list file.

■ External Symbol Cross-reference Information List File Name Specification (-xlf)

[Format]

`-xlf < Output file name >`

[Parameters]

<Output file name>

Specify the output file name. To change the directory of the output destination, add the path name prior to the output file name.

[Explanation]

The external symbol cross-reference information list file is created with the specified name.

When using this option, the -xl option can be omitted.

If the extension is omitted in the <output file name> specification, the default extension ".mpx" is added.

If this option is not specified, the absolute format load module file name is used whose extension is changed to ".mpx" as the output file name.

[Example]

```
flnk896s *.obj -o ap.abs -xl -xlf ccp903.mpx
flnk896s *.obj -o ap.abs -xlf ccp903
```

The above two examples share the same meaning.



6.2.52 External Symbol Cross-reference Information List Output Inhibit Specification (-Xxl)

Inhibiting the output of the external symbol cross-reference information list file is specified.

■ External Symbol Cross-reference Information List Output Inhibit Specification (-Xxl)

[Format]

-Xxl	(Default)
------	-------------

[Parameters]

None

[Explanation]

Inhibiting the output of the external symbol cross-reference information list file is specified.

This option is a default option, therefore, it is not necessary to specify it.

The previously specified -xl and -xlf are canceled.

[Example]

```
flnk896s *.obj -o ap.abs -xl -Xxl
flnk896s *.obj -o ap.abs -Xxl
flnk896s *.obj -o ap.abs
```

The above three examples share the same meaning.

6.2.53 Local Symbol Information List Output Specification (-sl)

The output of the local symbol information list file is specified.

The output of debug information when compiling, assembling, or linking is required.

■ Local Symbol Information List Output Specification (-sl)

[Format]

-sl

[Parameters]

None

[Explanation]

The creation of a local symbol information list file is instructed.

If this option is not specified, no local symbol information list file is created.

To output the local symbol information list file, specify the debug information output option (-g) when compiling, assembling, and linking.

[Example]

```
flnk896s *.obj -o ap.abs -sl -g
```

6.2.54 Local Symbol Information List File Name Specification (-slf)

This option is used to change the output destination directory or file name of the local symbol information list file.

■ Local Symbol Information List File Name Specification (-slf)

[Format]

-slf < Output file name >

[Parameters]

<Output file name>

Specify the output file name. To change the directory of the output destination, add the path name prior to the output file name.

[Explanation]

The local symbol information list file is created with the specified name.

When using this option, the -sl option can be omitted.

If the extension is omitted in the <output file name> specification, the default extension ".mps" is added.

If this option is not specified, the absolute format load module file name is used whose extension is changed to ".mps" as the output file name.

[Example]

```
flnk896s *.obj -o ap.abs -sl -slf ccp903.mps -g
flnk896s *.obj -o ap.abs -slf ccp903 -g
```

The above two examples share the same meaning.

6.2.55 Local Symbol Information List Output Inhibit Specification (-Xsl)

Inhibiting the output of the local symbol information list file is specified.

■ Local Symbol Information List Output Inhibit Specification (-Xsl)

[Format]

-Xsl	(Default)
------	-------------

[Parameters]

None

[Explanation]

Inhibiting the output of the local symbol information list file is specified.

This option is a default option, therefore, it is not necessary to specify it.

The -sl and -slf specified before are canceled.

[Example]

```
flnk896s *.obj -o ap.abs -sl -Xsl
flnk896s *.obj -o ap.abs -Xsl
flnk896s *.obj -o ap.abs
```

The above three examples share the same meaning.

6.2.56 Section Detail Map List Output Specification (-ml)

The output of the section detail map list file is specified.

■ Section Detail Map List Output Specification (-ml)

[Format]

-ml

[Parameters]

None

[Explanation]

The creation of the section detail map list file is specified.

If this option is not specified, no section detail map list file is created.

[Example]

```
flnk896s *.obj -o ap.abs -ml
```


6.2.57 Section Detail Map List File Name Specification (-mlf)

This option is used to change the output destination directory or file name of the section detail map list file.

■ Section Detail Map List File Name Specification (-mlf)

[Format]

`-mlf < Output file name >`

[Parameters]

<Output file name>

Specify the output file name. To change the directory of the output destination, add the path name prior to the output file name.

[Explanation]

The section detail map list file is created with the specified name.

When using this option, the -ml option can be omitted.

If the extension is omitted in the <output file name> specification, the default extension ".mpm" is added.

If this option is not specified, the absolute format load module file name is used whose extension is changed to ".mpm" as the output file name.

[Example]

```
flnk896s *.obj -o ap.abs -ml -mlf ccp903.mpm
```

```
flnk896s *.obj -o ap.abs -mlf ccp903
```

The above two examples share the same meaning.

6.2.58 Section Detail Map List Output Inhibit Specification (-Xml)

Inhibiting the output of the section detail map list file is specified.

■ Section Detail Map List Output Inhibit Specification (-Xml)

[Format]

-Xml	(Default)
------	-------------

[Parameters]

None

[Explanation]

Inhibiting the output of the section detail map list file is specified.

This option is a default option, therefore, it is not necessary to specify it.

The -ml and -mlf specified before are canceled.

[Example]

```
flnk896s *.obj -o ap.abs -ml -Xml
flnk896s *.obj -o ap.abs -Xml
flnk896s *.obj -o ap.abs
```

The above three examples share the same meaning.

CHAPTER 7

OUTPUT LIST FILE OF THE LINKER

This chapter explains the formats of each list file output by the linker and how to view the information.

- 7.1 Types of List Files Output by the Linker
- 7.2 Link List File
- 7.3 Absolute Assemble List File
- 7.4 External Symbol Cross-reference Information List File
- 7.5 Local Symbol Information List File
- 7.6 Section Allocation Detailed Information List File

7.1 Types of List Files Output by the Linker

The following five types of list files are output by the linker.

- Link list file
- Absolute format assemble list
- External symbol cross-reference information list
- Local symbol information list
- Section detail map list

Whether to output these files can be selected as options when activating the linker.

■ Link List File

The link list file outputs the options for linker activation, input module names, and information about sections and external symbols after module linking.

■ Absolute Format Assemble List

The absolute format assemble list is a list which displays the assemble list in the relative format output by the assembler in the absolute format based on information after module linking.

This list can be referenced when debugging on the assembler language level, and the addresses in each step of the machine language that are unidentifiable in the link list can be known.

■ External Symbol Cross-reference Information List

The external symbol cross-reference information list outputs information about external definition symbols of each module after linking and inter-module cross-reference of external reference symbols.

■ Local Symbol Information List

The local symbol information list outputs information about variables and functions including local symbols of each module after linking.

■ Section Detail Map List

The section detail map list creates information about the section allocation of each module after linking.

7.2 Link List File

The link list file can be divided into the following four parts depending on the information contents.

- Control list
- Map list
- Memory used information list
- External symbol list

This section explains the items output in each list.

■ Configuration of Link List File

The linker list file can be divided into the following four parts.

- Control list
 - Specified option
 - Input option
 - Error message
- Map list
 - Section name
 - Section attributes
 - Section allocation address after linking
- Memory used information list
 - ROM/RAM used information
 - Area internal information
 - General evaluation value information
- External symbol list
 - External symbol name
 - Types of the definition and reference
 - Symbol values

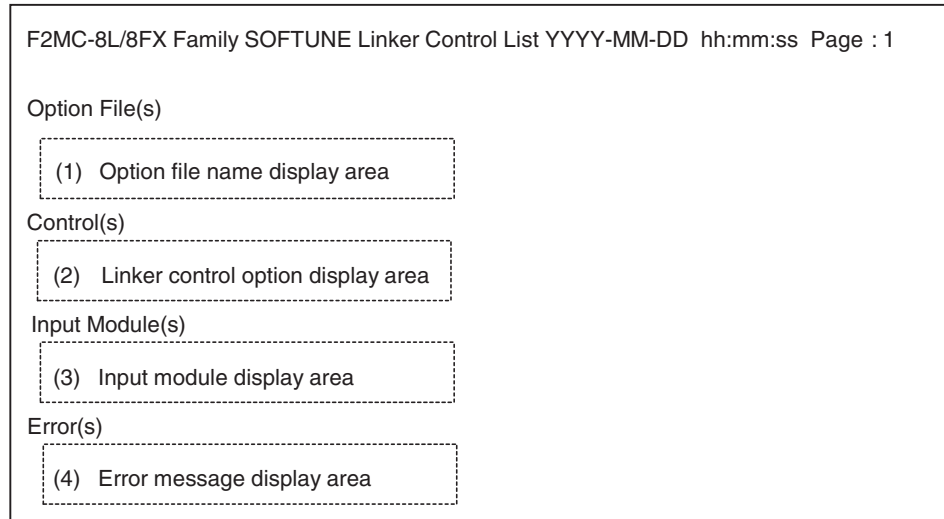
A control character of the page break is output at the boundary of each list.

7.2.1 Control List

In the control list, the options specified when the linker was executed and input module names are displayed. Errors detected during linking are also displayed.

■ List Output Format of the Control List Part

The following shows the list output format of the control list part.



Page header

The linker name, list name, date and time, and page number are displayed in the first line.

(1) Option file name display area

If any option file is used, the file name is displayed.

If no option file is used, [** no use **] is displayed.

(2) Linker control option display area

Specified options and valid options by default are displayed.

If an option is specified in any option file, @ is put prior to the option.

(3) Input module display area

File names and module names with the serial number starting with 1 are displayed.

(4) Error message display area

Error messages detected during processing are displayed.

If no error has been detected, [** Nothing **] is displayed.

■ List Display Example of the Control List Part

The following shows a list display example of the control list part.

```
F2MC-8L/8FX SOFTUNE Linker   Control List   2003-08-08 11:04:51   Page: 1
Option File(s)
  ** no use **
Control(s)
  -g
  -a
  -l usrlb.lib
  -ro prog=0x8000/0xffff
  -ra data=0x0080/0x047f
  -sc P+code=prog,D+data+S=data
  -cpu MB89PV630
Input Module(s)
  1 pca02.obj(pca01)
  2 pcasb.obj(pcasb)
  3 xccdef.obj(xccdef)
Error(s)
  ** Nothing **
```



7.2.2 Map List

In the map list, the section names, content types, attributes, and section allocation addresses after linking are displayed.

■ List Output Format of the Map List Part

The following shows the list output format of the map list part.

```
F2MC-8L/8FX Family SOFTUNE Linker Mapping List YYYY-MM-DD hh:mm:ss Page:2
(1)      Map information display area
```

Page header

The linker name, list name, date and time, and page number are displayed in the first line.

(1) Map information display area

Map information is displayed in order of start address, or if the start address is the same, in order of section occurrence.

S-Addr : Section start address(hexadecimal)

E-Addr : Section end address(hexadecimal)

Size : Section size (hexadecimal)

Section: Section content type

The section content type is displayed.

CODE	Program section
DATA	Data section
CONST	Data section with initial values
STACK	Stack section
DIR	direct section
DIRCONST	direct section with initial values
IO	IO section

After the section type, the link attribute is displayed.

P	Simple concatenation link
C	Shared link
N	No link

Type: Section attributes

The following attributes are displayed from left.

R/-	Read enabled/disabled
W/-	Write enabled/disabled
X/-	Executable/non-executable
I/-	Initial value Yes/No

AI : Boundary adjustment value for section allocation(hexadecimal)

If the boundary adjustment value is 0x100 or greater, "****" is displayed.

Sec.(Top **): Section name

The section generated by ROM → RAM transfer section is displayed "#" in the beginning.

"***" indicates how many digits of the section name can be displayed with the specified page width.

Note: The end address of a section with the section size 0 is displayed by [.....].

■ List Display Example of the Map List Part

The following shows a list display example of the map list part.

```
F2MC-8L/8FX SOFTUNE Linker Mapping List 2003-08-08 11:04:51 Page: 2
```

S-Addr.	- E-Addr.	Size	Section	type	AI	Sec.	(Top 81)	C
00000000	- 0000007F	00000080	IO	N	RW --	00 ABS	IO	
00000080	- 00000099	0000001A	DATA	P	RW --	01 REL	DATA	
00000080	-	00000000	DATA	P	RW --	02 REL	INIT	
00000400	- 0000047F	00000080	STACK	N	RW --	00 ABS	STACK	
00008007	- 000089E1	000009DB	CODE	P	R-XI	01 REL	CODE	
00008008	-	00000000	DATA	P	R---	02 REL	#INIT	
000089E2	- 00008A44	00000063	CONST	P	R--I	01 REL	CONST	

7.2.3 Memory Used Information List

The memory used information list displays the name of the area of the RAM specified area or the ROM specified area and the size of the free space or the over area and the header and end address of the specified area.

■ Memory Used Information List Part List Output Format

The following shows the memory used information list part list output format.

F2MC-8L/8FX Family SOFTUNE Linker ROM/RAM Used Info List YYYY-MM-DD hh:mm:ss Page: 3				
S_Addr.	-E_Addr.	Size	Remain	Name/State
(1) ROM/RAM Used Information				C
....				
(2) Area Internal Information				
(3) General Evaluation Value Information				
....				

Page Header

Linker name, list name, data and time, and page number are shown in the first line.

(1) ROM/RAM Used Information Display Area

- Lines containing a # character at their head among those output to the ROM/RAM used information display area indicate the information specified by the -ro or -ra options.
- Lines that do not contain a # character at the head of the lines that were output, display the header address and end address of the sections in the valid area and the size of the area that is used including the gap area and the remainder or insufficient values for the sizes of the used areas.
- Areas specified by the -ro or -ra option, are displayed the names of their sections that could not be located and the size of the section.

S_Addr:	Start Address (hexadecimal)
E_Addr:	End Address (Hex)
Size:	Area Size (Hex)
Remain:	Size of the area in the memory (Hex)

The following shows the header symbols.

+:	Free Area Size
-:	Over area size
Space:	When free/over area is 0.
Name/State:	Memory Area Name and Section Name

(2) Area Internal Information Area

The area internal information area displays the free area of the valid area, the used area and the gap area in a map image. The information displayed in this area is information only of the located section with regard to the specified area.

S_Addr:	Area Start Address (hexadecimal)
E_Addr:	Area End Address (Hex)
Size:	Area Size (Hex)
Name/State:	Area Type
	FREE:Free Area
	USED:Used Area
	GAP:Gap Area

(3) General Evaluation Value Information

General evaluation value information shows the following types of information for the overall ROM and RAM areas.

1. Total value of the specified area (Total)
2. Total value of the used area (Used) (Note: Includes the Gap area.)
3. Total value of the free or over area (Remainder)

■ Example Display of the Memory Used Information List Part List

The following is an example of the memory used information list part list display.

F2MC-8L/8FX Family SOFTUNE Linker ROM/RAM Used Info List 2003-08-04 12:59:23 Page: 3					
S_Addr.	E_Addr.	Size	Remain	Name/State	C
# 00000080-00000200 00000181 ----- RAM_1 00000080-00000105 00000086 +000000FB 00000080-00000087 00000008 ----- USED 00000088-000000FF 00000078 ----- GAP 00000100-00000105 00000006 ----- USED 00000106-00000200 000000FB ----- FREE <-Memory Area Specified in the Option <-Memory Status of the Section Located RAM_1 Area Information					
# 00000300-0000087F 00000580 ----- RAM_2 00000300-0000097F 00000680 -00000100 ** Not Locate ** 00000680 ----- DATA_3 00000300-0000087F 00000580 ----- FREE # 00000080-0000087F 00000800 ----- _RAM_1_ 00000080-..... 00000000 +00000800 00000080-0000087F 00000800 ----- FREE <-(*) <-Section Information Not Located (*) RAM_2 Area Information					
RAM -- Total(00000F01) Used(00000706) Remainder(+000007FB) <-All RAM Area Information					
# 00008007-0000A000 00001FFA ----- ROM_1 00008007-0000AF8A 00002F84 -00000F8A 00008007-0000A000 00001FFA ----- USED <-(*) ROM_1 Area Information					
# 00008000-0000FFFF 00008000 ----- ROM_2 00008000-..... 00000000 +00008000 00008000-0000FFFF 00008000 ----- FREE ROM_2 Area Information					
ROM -- Total(00009FFA) Used(00002F84) Remainder(+00007076) <-All ROM Area Information					

Notes:

- *1. Memory area information that has sections not located that were to be located in the memory area by the automatic locating option indicates the status with added section size. (When the numerical value expression exceeds 0xFFFFFFFF, the lower part of the 32-bit is displayed that value.)
- *2. When Mode 2 of the automatic location option is specified, this displays the memory area specified last for the section not located (for either the ROM area or the RAM area).
- *3. Sections specified by the user location are included in the memory area which includes the section header address.

7.2.4 Symbol List

In the symbol list, the external symbol names, definitions, reference types, and symbol values are displayed.

■ List Output Format of the Symbol List Part

F2MC-8L/8FX Family SOFTUNE Linker Symbol List YYYY- MM-DD hh:mm:ss Page: 4					
Symbol Value	Type	Def.	Symbol Name (Top **)	C	
(1) Symbol list display area					

Page header

The linker name, list name, date and time, and page number are displayed in the first line.

(1) Symbol list display area

Symbol Value : Symbol address or symbol value (hexadecimal)

Type : Symbol type

One of the following is displayed.

Addr. Address level

EQU EQU defined symbol

bit Bit attributes

Def. : Symbol definition

One of the following is displayed.

OM/LM Defined in the input object module or relative format load module.

LIB Defined in the linked library

user Symbol whose value is temporarily set using the -df option.

Symbol Name : Symbol name

"**" indicates how many digits of the symbol name can be displayed with the specified page width.

Note: If a symbol is referenced, the symbol name is displayed as it is. If it is not referenced, @ is displayed prior to the symbol name.

■ List Display Example of the Symbol List Part

The following shows a list display example of the symbol list part.

F2MC-8L/8FX Family SOFTUNE Linker Symbol List 2003-08-04 12:59:23 Page: 4					
Symbol Value	Type	Def.	Symbol Name (Top **)	C	
0000800F(ABS)	Addr.	OM/LM	_c01		
00008041(ABS)	Addr.	OM/LM	_main		
00008032(ABS)	Addr.	OM/LM	_func_01		
00008064(ABS)	Addr.	OM/LM	_func_02		

7.3 Absolute Assemble List File

The absolute format assemble list output by the linker consists of the following parts.

- Header
- Information list
- ROM/RAM and ARRAY lists
- Assemble source list
- Section information list
- Cross-reference list

This section explains the items output in each list.

■ Absolute Assemble List Format

● Header

Output in the first line of each page.

● Information list

The information list output by the assembler is output as it is.

● ROM/RAM and ARRAY lists

• ROM/RAM lists

Global symbol names allocated in the ROM/RAM areas and absolute address information are output.

• ARRAY list

Array element names, structure member names, and absolute address information are output.

If the option -alr is specified, the ROM/RAM and ARRAY lists are displayed. If -Xalr is specified, the ROM/RAM and ARRAY lists are not displayed.

● Assemble source list

The assemble source list displays a variety of information about assembling of the source program in units of lines. Error information, locations, object code are displayed.

● Section information list

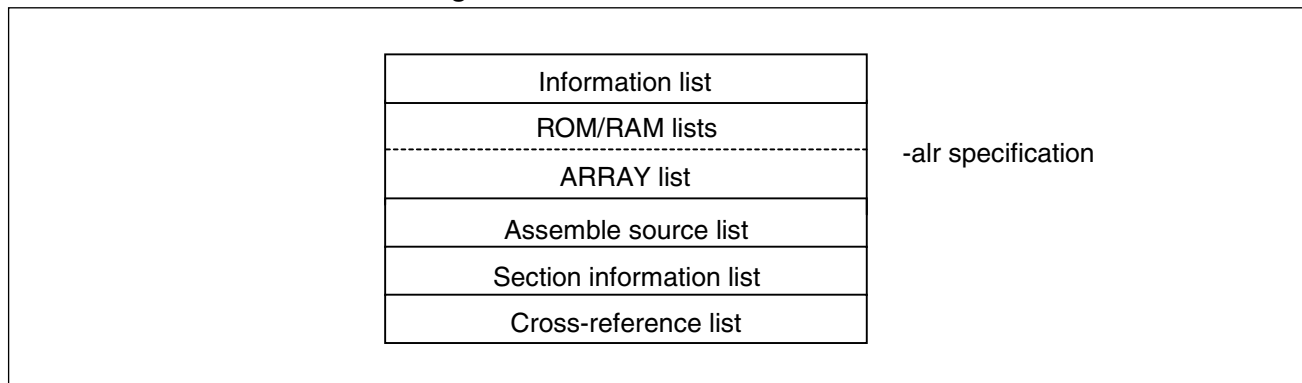
Names and attributes of the section defined in the source program are displayed.

● Cross-reference list

Definitions of the symbol names used in the source program and references are displayed using the line numbers.

Figure 7.3-1 shows the structure of the list.

Figure 7.3-1 Assemble List Structure



■ Error Messages in the Assemble List

If errors which occurred when assembling are in the assemble list, the error messages are displayed as they are.

7.3.1 Header and Information List

The header consists of four lines.

On the first page of the list, the information list (settings when activating the assembler, number of errors, number of warnings, source file names, include file names, option file names) is displayed after the header.

■ Header Format

The header consists of four lines. It is displayed at the head of each page. The first and second lines have the same format for all assemble lists, and the third line depends on each source program. The following shows the header format.

[1st and 2nd lines]

F2MC-8L/8FX Family SOFTUNE Linker VxxLxx yyyy-mm-dd hh:mm:ss Page: xxxx		
Tool name	Absolute assemble list creation date	Page number

[3rd line]

- Each list name - (Module name)

The contents of the original assemble list are displayed as they are.

[4th line]

A blank line is output.

7.3.2 ROM/RAM and ARRAY Lists

The ROM/RAM and ARRAY lists show information about the symbols described in the C source programs.

To output the ROM/RAM and ARRAY list, the output specification of debug information when compiling or assembling is required.

■ ROM/RAM and ARRAY Lists

The following shows the format of ROM/RAM and ARRAY lists.

- ROM/RAM LISTING - (Module name)				
NAME	ADDRESS	VALUE	KIND	MEMORY
--- Name ---	xxxxxxx	xxxxxxx x		xxx
*1	*2	*3	*4	*5

The following shows the format of ARRAY list.

- ARRAY LISTING - (Module name)		
NAME	ADDRESS	VALUE
--- Name ---	xxxxxxx	xxxxxxx
*1	*2	*3

*1:Symbol name

The names of symbols described in programs are displayed.

In the case of an ARRAY list, all elements of the arrays and structures are displayed.

*2:Address

The absolute address is displayed using a hexadecimal value.

The display positions of the symbol name and address can be changed as options. If -na is specified, the symbol name is first displayed and then the address is displayed. If -an is specified, the address is first displayed and then the symbol name is displayed.

For details, see 6.2.49 ROM/RAM and ARRAY Lists Symbol and Address Display Position Specification (-na, -an).

*3:Value

If an initial value is assigned to the symbol, the value is displayed as a decimal number.

*4:Symbol type

The following symbol types are available.

L :Variable value

F :Function name

T :Tag name reference

*5:Memory allocation

The memory area to which a symbol is allocated is displayed.

ROM :ROM area

RAM :RAM area



[Example 1]

- ROM/RAM LISTING - (sample1)				
NAME	ADDRESS	VALUE	KIND	MEMORY
_func_01	0000800F		F	ROM
_func_02	00008032		F	ROM
_symbol_01	00000086	000016	L	RAM
_symbol_02	00000082	000100	L	RAM
_struct_03_0003	0000062E		T	RAM

[Example 2]

- ARRAY LISTING - (src_0001)		
NAME	ADDRESS	VALUE
_array_1[0]	00000088	0
_array_1[1]	0000008C	1500
_array_2[0].val_1	00000092	100
_array_2[0].val_2	00000094	317
_array_2[0].val_3	00000096	65
_array_3[0][0]	0000009E	200
_array_3[0][1]	000000A0	30

7.3.3 Assemble Source List

The assemble source list is displayed with the location part using the absolute address and the object code part using the determined code after linking.

■ Assemble Source List Format

The following shows the assemble source list format.

- SOURCE LISTING - (Module name)						
SN	LOC	OBJ		LINE	SOURCE	
xx	xxxxxxxx	xxxxxxxxxxxxxxxx	x	xxxxx	x	-----
*1	*2	*3	*4	*5	*6	*7

The first line of the above format is called the source list header. The source list header is displayed on each page.

*1:Section acronym

The first two characters of a section name are displayed.

*2:Location

The 32-bit location value is displayed as a hexadecimal value.

*3:Object code

The determined object code after linking is displayed as hexadecimal values. If the object code cannot be displayed in one line, it is displayed in multiple lines.

*4:Object code type

The attributes of values contained in object code are displayed in the following order of priority.

I :External reference value

S :Section value

Blank :Absolute value

Since "R" displayed in the relative assemble list is converted to an absolute value in the absolute assemble list, it is not displayed.

*5:Line number

The line number is displayed as a decimal 5-digit number.

*6:Processing display of the preprocessor and optimization code check

Preprocessor

X :Line for assembling

& :Macro expansion line

Optimization code check

X :Instruction deletion for optimization

C :Replaced with another instruction for optimization

O :New instruction generated for optimization

V :Replaced with a low-level instruction for optimization (pair with A)

A :Replaced with an informational instruction for optimization (pair with V)

*7:Source line

One line of the source programs is displayed. If the line does not fit in one line of the list, it is displayed in multiple lines of the list.

SN	LOC	OBJ	LLINE	SOURCE
IN	0080	-----<INIT>-----	32	.SECTION INIT, DATA,ALIGN=1
IN	0080	0001	33	.DATA.H 1
CO	8007	-----<CODE>-----	62	.SECTION CODE, CODE, ALIGN=1
CO	8007	41	63	PUSHW TX
CO	8008	F1	64	MOVW A, SP
CO	8009	318033	65	CALL _abs_01_0002A
				----- Source program
			---	Line number
		-----		Object code : Determined value after linking
		-----		Location counter : Displayed with the 16-bit absolute address
		-----		Section acronym : First two characters of the section names

7.3.4 Section Information List

The section information list shows information about the sections defined in a program. The section information list is displayed on a new page after page break.

■ Section Information List Format

The following shows the section information list format.

```
F2MC-8L/8FX Family SOFTUNE Linker VxxLxx yyyy-mm-dd hh:mm:ss Page: xxxx
- SECTION LISTING - (Module name)
```

NO	SECTION-NAME	SIZE	ATTRIBUTES
xx	-----Section Name-----	xxxxxxxx	xxx xxx xxxx=xxxx
*1	*2	*3	*4 *5

*1:Section occurrence number

Starts with 0. This number corresponds to the section number in an object file.

*2:Section name

The defined section names are displayed in order of occurrence.

*3:Section size

The section size is displayed as a 32-bit hexadecimal value.

*4:Section type

The type of the sections is displayed. The following types are available.

CODE	: Code section
DAT A	: Data section
CONST	: Data section with initial values
COMMON	: Shared section
STACK	: Stack section
DIR	: direct section
DIRCONST	: direct section with initial values
IO	: IO section

*5:Section allocation format

The display of a relative assemble list is output as it is.

The ALIGN value is displayed for a relative section when assembling. The LOCATE value is displayed for an absolute section.

[Example]

```
F2MC-8L/8FX Family SOFTUNE LinkerV30L12 2003-08-05 13:23:19 Page: 5
- SECTION LISTING - ( src_0001 )
```

NO	SECTION-NAME	SIZE	ATTRIBUTES
0	DATA	04F0	DATA REL ALIGN=1
1	CODE	0033	CODE REL ALIGN=1
2	INIT	0008	DATA REL ALIGN=4

7.3.5 Cross-reference List

The cross-reference list shows information about the names described in a program and the relations between the their definition and reference.

The cross-reference list is displayed on a new page after page break.

■ Cross-reference List Format

The following shows the cross-reference list format.

```
F2MC-8L/8FX Family SOFTUNE Linker VxxLxx yyyy-mm-dd hh:mm:ss Page: xxxx
- CROSS REFERENCE LISTING - (Module name)
```

NAME	ATTRIB.	VALUE	DEFINITION/REFERENCE
-----Name-----	xxxx/xxxx	xxxxxxxxx xxx xxx xxx	
*1	*2	*3	*4

*1:Name

The symbol names and section names are displayed in order of upper-case letter, lower-case letter and alphabet.

*2:Symbol type

Symbol types are displayed in the following formats.

ABS	:Absolute symbol
REL	:Relative symbol
ABS/EXP	:Absolute symbol (external definition specification)
REL/EXP	:Relative symbol(external definition specification)
IMP	:External reference symbol
SECT/ABS	:Absolute section
SECT/REL	:Relative section
UNDEFINED	:Undefined symbol
REGLIST	:Register symbol

*3:Value

If the symbol has a value, the value is displayed with a 32-bit absolute address.

*4:Definition and reference line number

The line in which the symbol is defined and the lines which are referenced are displayed.

The sharp symbol "#" is added to the end of the line number in which the symbol is defined.

[Example]

```
F2MC-8L/8FX Family SOFTUNE Linker V30L12 2003-08-05 15:09:52 Page: 5
- CROSSREFERENCE LISTING - ( src_0001 )
```

NAME	ATTRIB.	VALUE	DEFINITION/REFERENCES
CODE	.SECT/REL	8007	18 #
L_22	REL	00008028	51 # 44
L_23	RELv	0000802C	54 # 50
L_func_0001	REL	0000802F	60 #
_abs_01_0002	IMP		19 37
_func_0001	.REL/EXP	00008007	27 # 26

7.4 External Symbol Cross-reference Information List File

The external symbol cross-reference information list file displays the external definition symbols of each object module after linking and cross-reference information between modules of the external reference symbols.

■ External Symbol Cross-reference Information List File

The following shows the output format of the external symbol cross-reference information list.

```
External Symbol Cross Reference List   yyyy-mm-dd hh:mm:ss   Page:  1
Module(s)
  1. module01
  2. module02
    :
 15. module15
External symbol Cross Reference List   yyyy-mm-dd hh:mm:ss   Page:  2
--- symbol ---          --- type/value ---   --- module (No.) ---
_extsym1      Addr.  0x0000008E      1# 2 4 8 14
_extsym2      Addr.  0x00000090      1 5# 6 7 11 15
_extsym3      Addr.  0x00000092      1# 3 12 13
_extsym7      ????.  0x00000000      2 5 8
_func_01      Addr.  0x00008007      1# 4 8
_func_02      Addr.  0x0000803F      1# 9 10 14
_func_03      Addr.  0x00008097      2# 7 15
n1m           EQU   0xFFFFFFFF      3# 5 8 12 15
n1p           EQU   0x00000001      3 7# 9 10 13
```

● Module(s)

The serial number starting with 1 is added to indicate the module name.

● symbol

The symbol names are displayed. (by default, up to 20 characters)

● type/value

The following types are available.

- Addr. :Address
- EQU :EQU symbol
- Bit :Bit symbol
- ????. :Undefined

"value" indicates a value. In the case of a bit symbol, its bit position is displayed in parentheses.

● module (No.)

Modules which are defined/referenced are displayed using their numbers. The # symbol indicates a defined module.

7.5 Local Symbol Information List File

The local symbol information list file displays information about the variables and functions which include local symbols for each module constituting an absolute format load module.

Since this list is created based on debug information when compiling and assembling, it is necessary to specify the **-g** option.

■ Local Symbol Information List File

The following shows the output format of the local symbol information list.

```
Local Symbol List          yyyy-mm-dd hh:mm:ss   Page:  1
Module(s)
  1. module01
  2. module02
   :
 15. module15

Local Symbol List          yyyy-mm-dd hh:mm:ss   Page:  2

=== Module No. 1 (module01) ===
--- symbol ---      -- kind --  --- val ---
C
func1                Func.  g 0x8007
localstatic1         Var.   s 0x0080

=== Module No. 2 (src_0002) ===
--- symbol ---      -- kind--  --- val ---
C
func2                Func.  g 0x8056
static_symbol_01     loc.   s 0x0100
global_symbol>       Var.   g 0x0102
```

- Module(s)

The serial number starting with 1 is added to indicate the module name.

- symbol

The symbol names are displayed.

Symbols used in a function are displayed from the 3rd columns.

Up to 20 characters of the symbol name are displayed in one line of the list.

- Kind

The following symbol types are displayed.

Var.	:Variable (C)
Func.	:Function (C)
loc.	:Local (C)
Addr.	:Address (ASM)
EQU	:EQU symbol (ASM)
bit	:bit symbol (ASM)
???	:Undefined
s	:static (C)
g	:global (C)

"value" indicates a value.

- val

The value of a symbol is indicated.

Note:

No detailed information about structures (member names) nor the typedef definitions are displayed.

7.6 Section Allocation Detailed Information List File

The section allocation detailed information list file creates information about section allocation for each module constituting an absolute format load module.

A mapping list of a whole section is displayed in one map list file and more detailed information about section allocation can be found.

■ Section Allocation Detailed Information List File

The following shows the output format of the section allocation detailed information list.

```

Section Mapping List          yyyy-mm-dd hh:mm:ss   Page:   1

Module(s)
  1. module01
  2. module02
  :
  4. module15
Section Mapping List          yyyy-mm-dd hh:mm:ss   Page:   2

S_Addr.  - E_Addr.  Size      Section  Type  AI  M.No.  Sec.(Top 80)   C
00000080 - 0000008D 0000000E DATA P  RW-- 01  4    INIT
0000008E - 00000093 00000006 DATA P  RW-- 01  2    DATA
00000094 - 000000AD 0000001A DATA P  RW-- 01  3    DATA
000000AE - 000000D5 00000028 DATA P  RW-- 01  4    DATA
00000400 - 0000047F 00000080 STACK N  RW-- 01  6    STACK
00008007 - 0000806A 00000064 CODE P   R-XI 01  1    CODE
0000806B - 000080A6 0000003C CODE P   R-XI 01  2    CODE
000080A7 - 000080D5 0000002F CODE P   R-XI 01  3    CODE
000080D6 - 000081E9 00000114 CODE P   R-XI 01  4    CODE
000081EA - 000081F7 0000000E DATA P   R--- 01  4    #INIT
  
```

- Module(s)

The serial number starting with 1 is added to indicate the module name.

- S_Addr. -E_Addr.

Section's start and end addresses.

A section with overlapping addresses has (*) prior to the start address.

If the size of a section is 0, the end address of it is displayed as ".....".

- Size

A section in which allowed address space is overflowed is displayed with the maximum size + 1.

- Section

The following section content types are available.

CODE	:Program section
DATA	:Data section
CONST	:Data section with initial values
STACK	:Stack section
DIR	:direct section
DIRCONST	:Direct section
IO	:IO section

The link attribute is added to the end of the section type.

P	:Simple concatenation link
C	:Shared link
N	:no link

- type

The following attributes are displayed from left.

R/-	:Read enabled/disabled
W/-	:Write enabled/disabled
X/-	:Executable/non-executable
I/-	:Initial value Yes/No

- AI

The boundary adjustment value for section allocation is displayed as hexadecimal values.

- M.No.

The module numbers are displayed. Module numbers displayed in Module(s) are displayed.

- Sec.(Top xx)

The section names are displayed. xx of (Top xx) has the number of characters that allows the section name to be displayed in one line.

prior to a section name indicates that data with initial values to be transferred to RAM before execution is allocated in the section.

CHAPTER 8

LINKER RESTRICTIONS AND Q&A

This chapter explains about linker restrictions and Q&A for use.

- 8.1 Linker Restrictions
- 8.2 Q&A for Using the Linker

8.1 Linker Restrictions

There is no particular restriction on the number of input files and sections that can be processed when using the linker. However, there are some restrictions for the number input files and library files on the object format.

■ Linker Restrictions

The linker has no restriction on the number of input files, sections, or symbols that can be processed. If it becomes necessary to register the section names or symbols names for internal processing of the linker, memory is dynamically allocated. And processing is possible until all memory that can be used by the linker is used up.

If it becomes impossible to continue processing, a message to report insufficient memory is issued and then the linker processing ends.

■ Linker Reservation Symbol

In the linker, symbols with the "_ROM_section name" and "_RAM_section name" are automatically created for each section using the ROM → RAM transfer function.

Therefore, if a symbol of the same name is defined in a user program, "W1327L: Duplicate symbol definition (symbol name)" is generated. The user must not define symbols with the "_ROM_section name" and "_RAM_section name".

■ Restrictions on the Object File Format

Restrictions of the linker are listed above. There are following restrictions on the numbers of symbols and others that can be managed in object files.

Number of input files	65535
Number of sections (per file)	65535
Number of external definition symbols (per file)	65535
Number of external reference symbols (per file)	65535

Note:

The files here mean object files. When using relative format load module files, count them as the total number of input object files when creating them.

8.2 Q&A for Using the Linker

Section 8.2 shows the questions and answers on using a linker.

■ Q&A for Using the Linker

- Using the wild card

Q.	There are a large number of input object module files. Can the wild card be still used?
A.	If you use the wild card for specifying the input file on the command line, the linker expands and executes it. You can specify the input file name in an option file, and the wild card can be also used here. Refer to the following example for using the wild card.
Example.	<code>flnk896s *.obj -o outfile.abs</code> <code>flnk896s mactrl.obj xz???.obj</code>

Q.	The wild card can be used when specifying section allocation, but how can I use the wild card?
A.	The wild card may be useful when many sections with the same content type should be unified or when programs are created using many section names. It may become necessary to decide characters to be the keywords when naming the section names, considering the use of the wild card by the linker.
Example.	Section names are defined with the names like DTdata1, DTdata2, DTdata3, DTdata4for the sections whose content type is data, and the names like CDprog1, CDprog2, CDprog3, CDprog4.... for the sections whose content type is code. In this case, the following specification method can be selected. (Only the -sc option part is shown) <code>-sc DTdata1+DTdata2+DTdata3+DTdata4=0x0100,</code> <code>CDprog1+CDprog2+CDprog3+CDprog4=0x8000</code> <code>-sc DT*=0x0100,CD*=0x8000</code> <code>-sc */data=0x0100,*/code=0x8000</code>

■ Handling Variables with Initial Values

Q.	When developing embedded programs using C compilers, variables with initial values are created. Since these variables are rewritten during execution of programs, they must be on RAM during execution. Tell us the procedure for creating programs and precautions.
A.	In embedded programs, variables with initial values are on ROM first and they must be on RAM when they are referenced. Therefore, programs become inoperable if the reference address in programs is not set to RAM and a mechanism to transfer initial data from ROM to RAM before application execution is not implemented. This mechanism is implemented by using the ROM → RAM transfer section function supported by the linker. Variables with initial values generated C compilers are gathered in the INIT section. There is no need of particular care when creating programs except the total number of bytes of the variables with initial values and RAM size. For the ROM → RAM transfer section function, see Section "5.9 Sections to be Transferred from ROM to RAM". The user must write a program to transfer initial value data using assembler. Example 1 shows "Program example for transferring initial value data".
Example.	[Program example for transferring initial value data] <pre> .import _ROM_INIT, _RAM_INIT (1) .section INIT, data (2) .section start, code MOVW IX, #_ROM_INIT MOVW EP, #_RAM_INIT MOVW A, #sizeof (INIT) JMP END_CHK LOOP: MOV A, @IX + 0 MOV @EP, A INCW IX INCW EP XCH A, T DECW A END_CHK: BNZ LOOP </pre> (1) ROM_INIT is a symbol to indicate the start address of the INIT section (transfer source) on ROM. _RAM_INIT is a symbol to indicate the start address of the INIT section (transfer destination) on RAM. These symbols are automatically generated by the Sections to be transferred from ROM to RAM. Declare them using the .import instruction. (2) Define an INIT section without content to extract the section size (transfer size).

PARTIII LIBRARIAN

PART3 describes the specifications, options, and output lists of a librarian.

CHAPTER 9 SPECIFICATIONS OF LIBRARIAN

CHAPTER 10 OPTIONS OF LIBRARIAN

CHAPTER 11 LIST FORMATS OF LIBRARIAN

CHAPTER 12 RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN

CHAPTER 9

SPECIFICATIONS OF LIBRARIAN

This chapter explains the functions and an outline of the function of a librarian.

A librarian is a tool used to create a library file.

- 9.1 Functions of Librarian
- 9.2 Function Types of Librarian
- 9.3 Creating and Editing Library File
- 9.4 Extracting a Module from Library File
- 9.5 Deleting Debugging Information of Library
- 9.6 Checking and Displaying the Contents of Library File
- 9.7 Mixing of Objects for Librarian

9.1 Functions of Librarian

A librarian is a tool used to create a library file by combining multiple object modules that an assembler has output

■ Roles of Librarian

To develop a program, divide a source program for each function into modules, each of which you then compile and assemble.

A linker then combines the compiled and assembled modules into one to create the target program.

A librarian is used to create a library file by combining multiple object modules that an assembler has output.

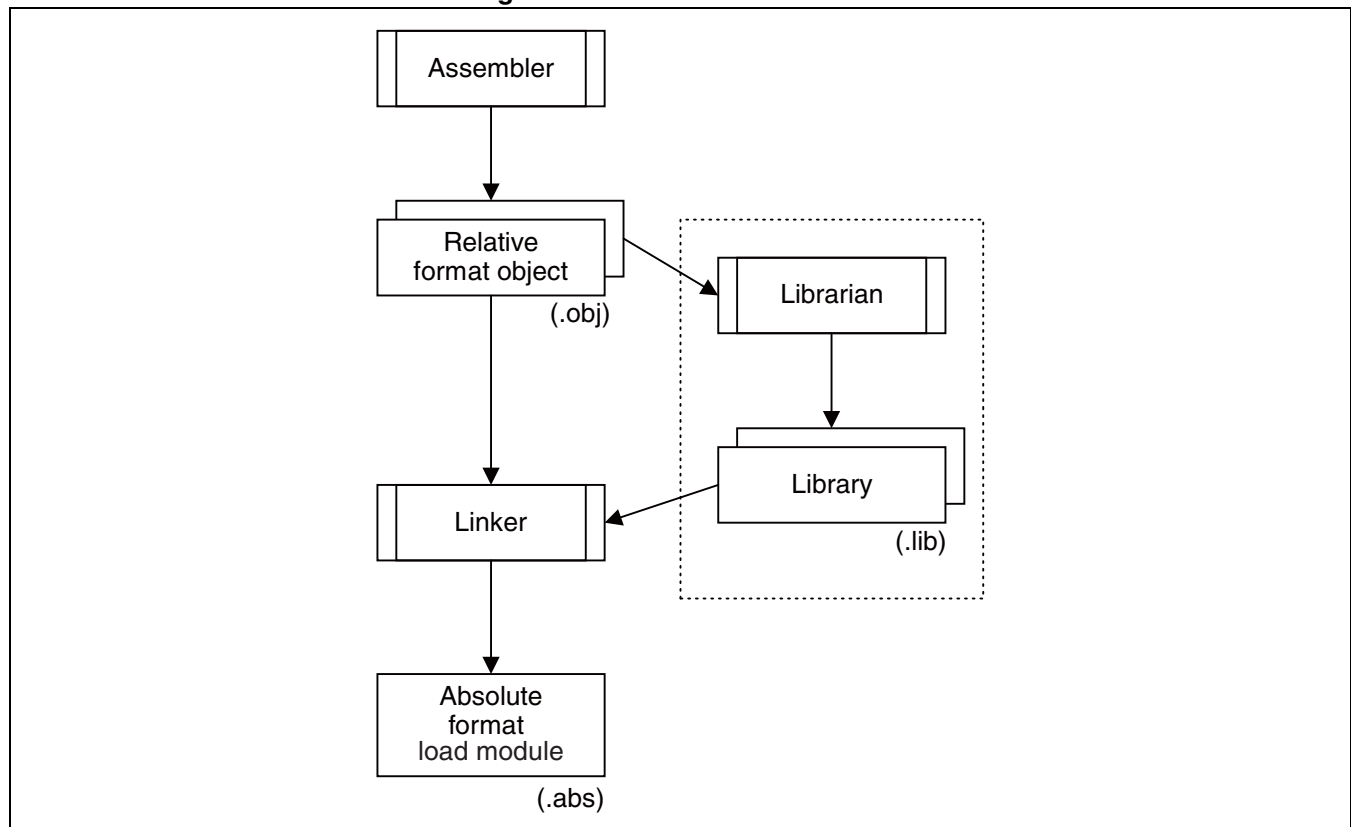
You may register multiple modules that make up a program in the library file dedicated to the program to manage them in a batch.

Registering the frequently used modules in a batch to create a general-purpose library file enables them to be easily used in other programs. A correct example of this is a library in C.

Since a librarian allows you to add, delete, or replace modules of a library file, you can keep them up-to-date.

Figure 9.1-1 shows the roles of a librarian.

Figure 9.1-1 Roles of a Librarian



9.2 Function Types of Librarian

A librarian has the following six functions:

- **Creates a new library file**
 - **Edits a library file**
 - **Extracts a module from a library file**
 - **Deletes debugging information**
 - **Checks the contents of a library file**
 - **Displays the contents of a library file**
-

■ Creating a New Library File

This function is used to create a new library file using object module files as input files.

■ Editing a Library File

This function is used to add an object module to, or delete an unnecessary object module from an existing library file.

If a module registered in a library file is found to be defective or you want to change its functions, you need to replace it with a modified one. This may be done by deleting and adding, but a replacement function is also provided.

■ Extracting a Module from a Library File

This function is used to extract an object module registered in a library file and put it back in the format of an object module file.

■ Deleting Debugging Information

This function is used, when an object module with debugging information is registered, to remove only debugging information from it and register it again.

■ Checking the Contents of a Library File

This function is used to check that correspondence between external defined and reference symbols is properly solved in the group of object modules that make up a library file.

This function is also used to check whether object modules with debugging information are registered.

■ Displaying the Contents of a Library File

This function is used to output information such as module names and external symbols registered in a library file into a list file or the standard output.

9.3 Creating and Editing Library File

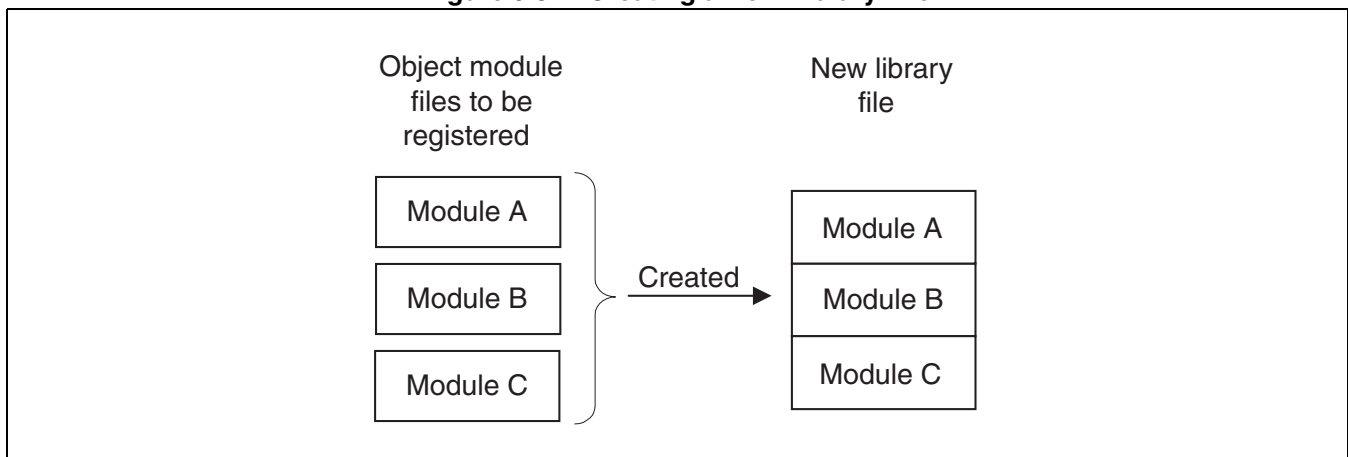
Object modules (more than one) that an assembler has output may be united and registered as a library file.

A module may be added to, deleted from, or replaced with the one in an existing library file.

■ Creating a New Library File

Object modules (more than one) that an assembler has output may be united and registered as a library file (See Figure 9.3-1).

Figure 9.3-1 Creating a New Library File



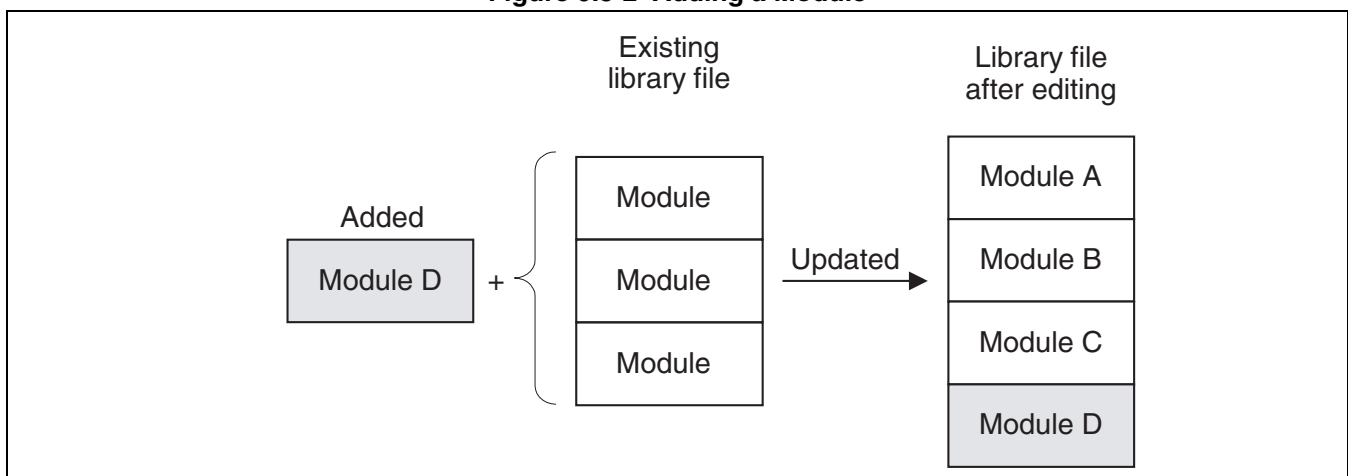
■ Editing a Library File

A module may be added to, deleted from, or replaced with the one in an existing library file.

● Adding a module

A module may be added to an existing library file (See Figure 9.3-2).

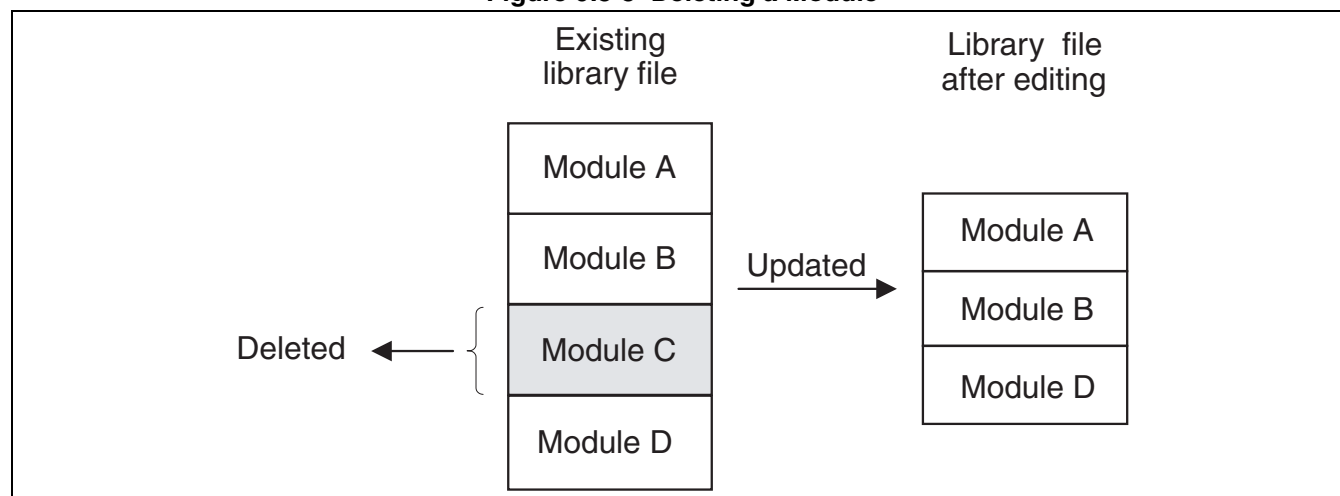
Figure 9.3-2 Adding a Module



- Deleting a module

An unnecessary module may be deleted from an existing library file (See Figure 9.3-3).

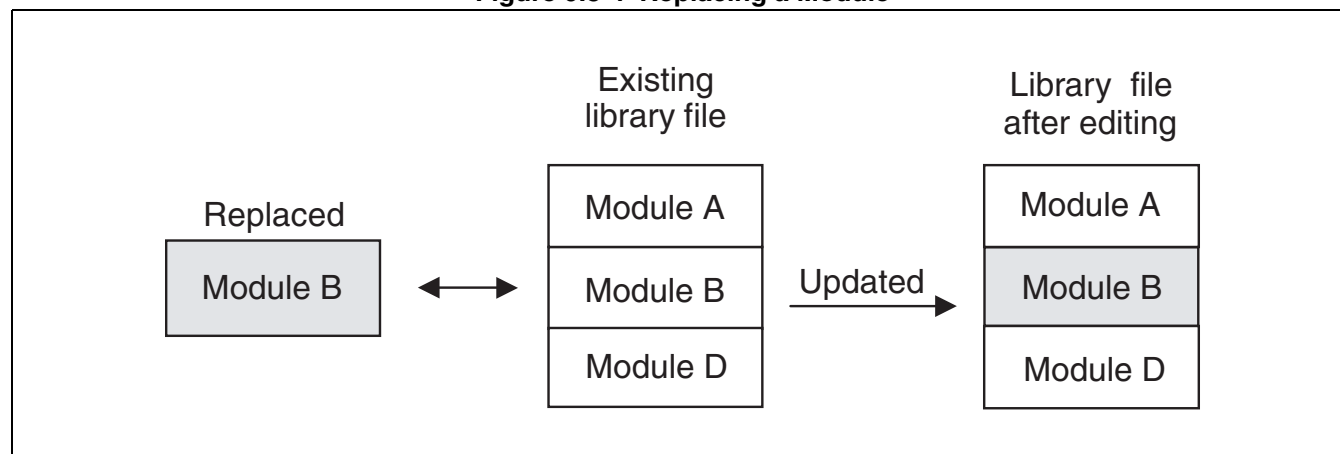
Figure 9.3-3 Deleting a Module



- Replacing a module

A module in an existing library file may be replaced with a new one (See Figure 9.3-4).

Figure 9.3-4 Replacing a Module



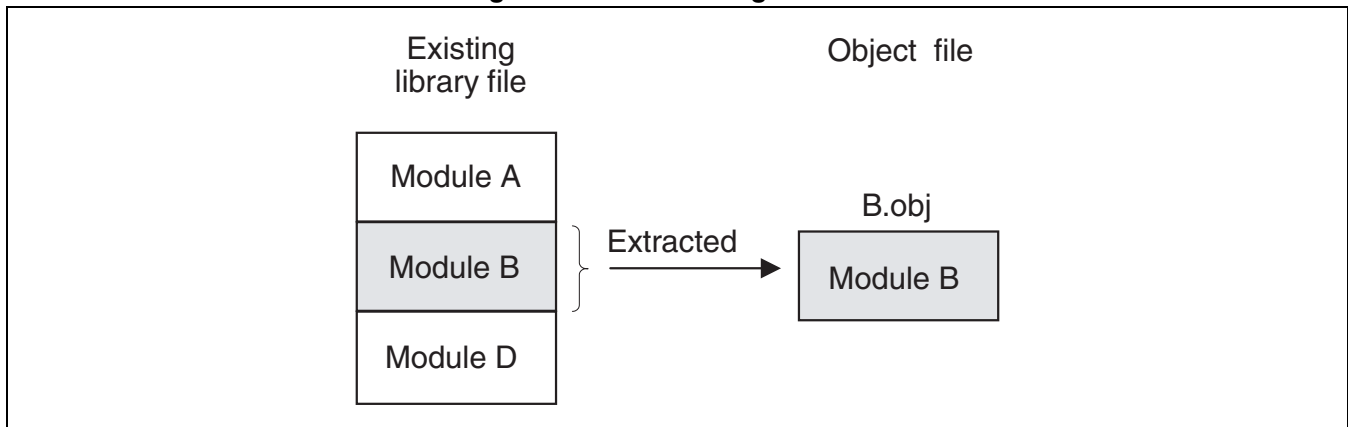
9.4 Extracting a Module from Library File

A module may be extracted from a library file and put back into an object module format file.

■ Extracting a Module from a Library File

A module may be extracted from a library file and put back into an object module file (See Figure 9.4-1).

Figure 9.4-1 Extracting a Module



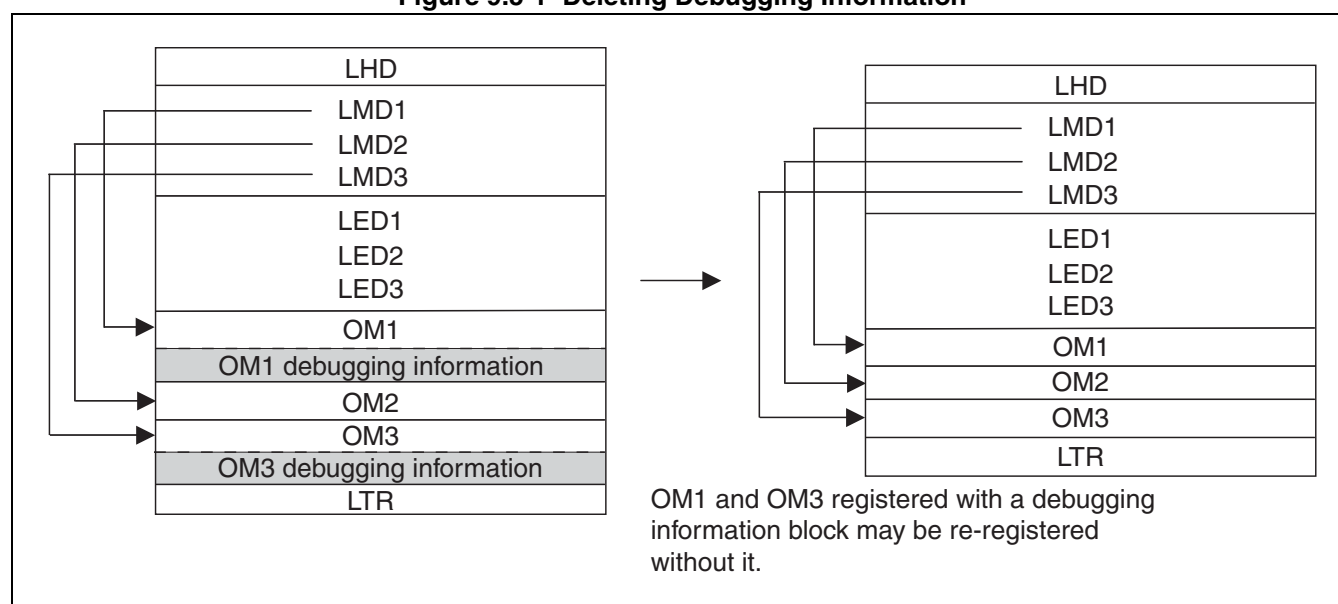
9.5 Deleting Debugging Information of Library

An object module registered with a debugging information block in a library may be re-registered without it.

■ Deleting Debugging Information

An object module registered with a debugging information block in a library may be re-registered without it (See Figure 9.5-1).

Figure 9.5-1 Deleting Debugging Information



9.6 Checking and Displaying the Contents of Library File

The following two items are checked.

- Whether an unsolved external reference symbol exists in a library
- Whether a module with debugging information exists

Additionally, this function provides you with information such as the date and time of creating and updating a library file and registering a module and the names of external definition symbols defined in each module.

■ Checking the Contents of a Library File

The following two items may be checked.

- Whether an unsolved external reference symbol exists in a library

A linker, when a module taken in from a library contains an external reference symbol, first searches for a defined symbol in the same library file.

Therefore, it is recommended that whenever an external reference symbol exists in a module in a library file, the module containing the concerned external definition symbol should be registered in the same library file.

The program checks the correspondence between external reference and defined symbols in a library file. If any undefined reference symbols remains, It outputs a diagnostic message.

- Whether a module with debugging information exists

If an object module with debugging information is contained, a diagnostic message is output.

If a module to be registered with the -g option specification in a library file contains debugging information in the object module, it is registered in the library with the debugging information unremoved.

The above function is provided so that a module registered in a library may be debugged. However, after the operation check, the debugging information will no longer be required.

■ Displaying the Contents of a Library File

The module and external definition symbol information of a library file is edited and output to a list file or the standard output.

The list provides information such as: the date and time a library file was created and updated, when a module was registered, and the name of an external definition symbol defined in each module.

For the contents of display, see "CHAPTER 11 LIST FORMATS OF LIBRARIAN".

9.7 Mixing of Objects for Librarian

The librarian (flib896s) causes an error if objects incompatible with the target CPU specified by the -cpu option and libraries are mixed.

■ Mixing of Objects for Librarian

The F²MC-8FX is different in the operation of the DIVU instruction from the F²MC-8L(see Table 9.7-1).

Table 9.7-1 Operational Comparison of DIVU Instruction

F ² MC-8L	F ² MC-8FX
T(16bit)/AL(8bit) = AL(8bit) remainder TL(8bit)	T(16bit)/A(16bit) = A(16bit) remainder T(16bit)

For this reason, if the F²MC-8L objects and the F²MC-8FX objects are mixed, the librarian (flib896s) causes an error.

If the DIVU instruction is unprovided, the F²MC-8L objects and the F²MC-8FX objects can be mixed.

To have the F²MC-8L objects and the F²MC-8FX objects mixed, specify the object mixing check inhibit specification option (-Xobjmixchk).

Note:

A mixed between the F²MC-8L objects and the F²MC-8FX objects with DIVU instruction may cause an unexpected error.

For the mixed, pay full attention to the presence or absence of the DIVU instruction.



CHAPTER 10

OPTIONS OF LIBRARIAN

This chapter explains the syntax, parameters, and precautions for options of a librarian.

10.1 List of Options of Librarian

10.2 Details of the Options of Librarian

10.1 List of Options of Librarian

Options are provided to specify the operations of a librarian in detail.

■ List of Options

The table below gives the option names and an overview of their functions.

For details on the parameters and functions required for an option, see the description on each option.

Table 10.1-1 List of Options for a Librarian

	Function	Option	Remarks
Options for creating and editing a library	Adding (registering) a module	-a	
	Replacing (registering) a module	-r	
	Deleting a module	-d	
	Extracting a module	-x	
Options for outputting a list	Specifying to output a list file	-m	
	Specifying not to output a list file	-Xm	Default
	Specifying to output detailed information of a list file	-dt	s,d,r,a
	Specifying the number of lines per page of a list	-pl	Default: 60
	Specifying the number of columns per line of a list	-pw	Default: 70
Options for searching and protecting a file	Creating a backup file	-b	
	Inhibiting the creation of a backup file	-Xb	Default
	Checking the contents of a library file	-c	
	Optimizing the contents of a file	-O	
Other options	Specifying to output debugging information	-g	
	Specifying not to output debugging information	-Xg	
	Specifying a target CPU	-cpu	need
	Specifying CPU Information File	-cif	
	Object mixing check specification	-objmixchk	Default
	Object mixing check inhibit specification	-Xobjmixchk	
Common options	Specifying not to read a default option file	-Xdof	
	Specifying to read an option file	-f	
	Specifying to display help messages	-help	
	Specifying to output the version number and messages	-V	
	Specifying not to output the version number and messages	-XV	Default
	Specifying to output a termination message	-cmsg	
	Specifying not to output a termination message	-Xcmsg	Default
	Specifying to set the termination code to 1 when a warning occurs	-cwno	
	Specifying to set the termination code to 0 when a warning occurs	-Xcwno	Default

10.2 Details of the Options of Librarian

Section 10.2 describes the options of a librarian.

The common options throughout linkage kit are described in "CHAPTER 3 COMMON OPTIONS".

■ Options for Creating and Editing a Library

Details of the options for creating and editing a library are described in Sections "10.2.1 Adding (Registering) a Module (-a)" through "10.2.4 Extracting a Module (-x)".

■ Options for Outputting a List

Details of the options for outputting a list are described in Sections "10.2.5 Specifying to Output a List File (-m)" through "10.2.9 Specifying the Number of Columns Per Line of a List (-pw)".

■ Options for Searching and Protecting a File

Details of the options for searching and protecting a file are described in Sections "10.2.10 Creating a Backup File (-b)" through "10.2.13 Optimizing the Contents of a File (-O)".

■ Other Options

Details of other options are described in Sections "10.2.14 Specifying to Output Debugging Information (-g)" through "10.2.19 Object Mixing Check Inhibit Specification (-Xobjmixchk)".

10.2.1 Adding (Registering) a Module (-a)

Use the -a option to create a new library file or to add a module to an existing library file.

■ Adding (Registering) a Module (-a)

[Syntax]

```
-a < Object module file name > [ , ... ]
```

[Parameter]

<Object module file name>

Object module file name that the assembler has output

[Description]

Specify a file name for the module to be registered in a library file.

If the file name has no extension specified, the ".obj" extension is assigned.

If an already registered module has the same name as the one to be registered, an error message is output and the latter module is not registered.

If an external definition symbol with the same name exists, a module is not registered either.

To specify <Object module file name>, you may use a wild card.

[Example 1]

```
flib896s syslib.lib -a mod1.obj,mod2.obj,modx.obj
```

Register the object module files, mod1.obj, mod2.obj, and modx.obj in the library file, syslib.lib:

- If syslib.lib does not exist: Creating one
- If syslib.lib exists: Adding modules to it and re-registering it

[Example 2]

```
flib896s syslib -a "mod*.obj" -a chksw
```

The object module files with the extension .obj and the first three characters as mod in the current directory and chksw.obj are registered.

Note:

Up to 65535 modules may be registered.

When using a wild card, you cannot specify <Object module file name> separated with commas. Specify them in multiple -a options as shown in the above example.

If a wildcard is specified, you are not supplied with an extension. Be sure to specify an extension.

For information on the expansion of a wildcard in a file name which depends on the OS, see "APPENDIX G SPECIFICATION DIFFERENCES DEPENDING ON THE OS".

10.2.2 Replacing (Registering) a Module (-r)

A module in an existing library file is replaced a new module with the same name.

■ Replacing (Registering) a Module (-r)

[Syntax]

```
-r < Object module file name > [ , ... ]
```

[Parameter]

<Object module file name>

An object module file name that the assembler has output

[Description]

If a module in the library file being edited has the same name as the one in the specified file, the former module is replaced. Otherwise, the specified module is registered.

If the file name has no extension specified, the ".obj" extension is assigned.

To specify <Object module file name>, you may use a wild card.

[Example 1]

```
flib896s syslib.obj -r loadx.obj,loady.obj
```

The two modules in loadx.obj and loady.obj replaces those with the same names in the library file being edited.

If no module with the same name exists in the library file, the modules are added to the library file which is then re-registered.

[Example 2]

```
flib896s syslib.lib -r "load?.obj"
```

The object module files with the extension .obj and the first four characters as load followed by one arbitrary character in the current directory are replaced.

Note:

When using a wild card, you cannot specify <Object module file name> separated with commas. Specify them in multiple -r options.

If a wildcard is specified, you are not supplied with an extension. Be sure to specify an extension.

For information on the expansion of a wildcard in a file name which depends on the OS, see "APPENDIX G SPECIFICATION DIFFERENCES DEPENDING ON THE OS".

10.2.3 Deleting a Module (-d)

An unnecessary module is deleted from a library file.

■ Deleting a Module (-d)

[Syntax]

`-d <Module name> [ , ... ]`

[Parameter]

<Module name>

Name of a module to be deleted

[Description]

The specified module is deleted from a library file.

Be sure to specify a module name, not a file name.

[Example]

```
flib896s syslib.lib -d inchar,outchar
```

Two modules, inchar and outchar are deleted from syslib.lib.

Note:

Although you may specify a module name using a pseudo instruction of assembler, use the same name for an object module file and a module unless absolutely cause. Using different names for them will cause an error when editing a library.

To specify the same name for a file and a module, use a name consisting only of alphanumeric characters and underscores.

To check the module name, use the list output option (-m) to refer to the module name that is output in the beginning of a list file.

10.2.4 Extracting a Module (-x)

A module is extracted from a library file and put it back into an object module file before registration.

■ Extracting a Module (-x)

[Syntax]

```
-x <Module name> [, <Object module file name> ]
```

[Parameters]

<Module name>

Name of module to be extracted

<Object module file name>

Name of output file of extracted module

[Description]

The specified module is extracted from a library file.

The extracted module becomes the same object module file before registration.

If the <Object module file name> is not specified, a file is created with the <Module name> followed by the ".obj" extension.

[Example]

```
flib896s syslib -x add
```

```
flib896s syslib.lib -x add,add.obj
```

The module, add is extracted from an existing library file and the add.obj file is created.

```
flib896s syslib -x add,add.o
```

The module, add is extracted from an existing library file and the add.o file is created.

Note:

You may specify as many -x options as the modules to be extracted.

If two module names are specified, the second one specified is valid. In the following example, add.obj is not created and only addfunc.obj is created.

```
flib896s syslib -x add -x add,addfunc.obj
```

10.2.5 Specifying to Output a List File (-m)

Module names and external definition symbol names registered in a library file are output as the information list.

■ Specifying to Output a List File (-m)

[Syntax]

```
-m { < List file name > | - }
```

[Parameter]

<List file name>

Specify the file name of the librarian list to be output.

Specify a hyphen (-) to output the list in the standard output.

[Description]

Module names and external definition symbol names registered in a library file are output as the information list.

If the <List file name> has no extension specified, ".mp2" is added to it.

This option allows you to output only the information of module names registered. To display more detailed information use the -dt option described later.

The list contents show the status when the librarian is terminated.

If no other option related to editing is provided, the contents of the specified library file is listed.

If you want to check the contents of a library file on the screen without storing it in a list, specify a hyphen in the parameter.

[Example 1]

```
flib896s syslib.lib -m libx.mp2
```

The module name list registered in syslib.lib is output to libx.mp2.

[Example 2]

```
flib896s syslib -a obj1,obj2 -m libx.lis
```

The contents of syslib.lib created after the obj1.obj and obj2.obj modules are added are output to libx.lis.

[Example 3]

```
flib896s syslib -m -
```

The module name list registered in syslib.lib is output to the standard output.

10.2.6 Specifying not to Output a List File (-Xm)

This specification inhibits a librarian from outputting a list file.

■ Specifying not to Output a List File (-Xm)

[Syntax]

-Xm

[Parameter]

None

[Description]

This specification inhibits the output of a list file.

Specifying the -Xm option after the -m option disables the -m option.

[Example]

```
flib896s syslib.lib -m libx.mp2 -Xm
```

The list file is not created.

10.2.7 Specifying to Output Detailed Information of a List File (-dt)

The -m option specifies outputting the list but only displays the list of registered module names.

Use the -dt option to obtain information on sections and external symbols for each module registered in a library, or to obtain information on external definitions and external reference symbols for the entire library.

■ Specifying to Output Detailed Information of a List File (-dt)

[Syntax]

```
-dt < Information type > [ , < Information type > ] ...
```

[Parameter]

<Information type>

s : Outputs a section name and its size for each module.

d : Outputs external definition symbols for each module.

r : Outputs external reference symbols for each module.

a : Outputs for the entire library external definition symbols and external reference symbols yet unsolved in the library.

[Description]

If this option is not specified, only the registered module names are output in a list file. This option is used to obtain more detailed information.

The <Information type> must be always specified.

The <Information type> may be specified by listing multiple keywords separated with commas.

If the -m option is not specified, this option is invalid.

[Example 1]

```
flib896s syslib.lib -m libx.mp2 -dt r,s
```

The list containing external reference symbols and section names is output to libx.mp2.

[Example 2]

```
flib896s syslib -m libx.lis -dt s,d,r,a
```

All the information that a librarian can output is output to libx.lis.

10.2.8 Specifying the Number of Lines Per Page of a List (-pl)

Use this option to change the number of lines output per page of a list from its default (60 lines).

■ Specifying the Number of Lines Per Page of a List (-pl)

[Syntax]

<code>-pl < Number of lines ></code> (Default: 60)
--

[Parameter]

<Number of lines>

Specify 0 or between 20 and 255 inclusive.

[Description]

Specify the number of lines to be printed per page of a list file.

Specifying 0 disables the page control when the list file is output.

If the -m option is not specified, this option is invalid.

[Example 1]

```
flib896s syslib.lib -m libx.mp2 -pl 40
```

The number of lines per page of a list is 40.

[Example 2]

```
flib869s syslib.lib -m - -dt s -pl 0
```

A list with section information added is output to the standard output without page ejection.



10.2.9 Specifying the Number of Columns Per Line of a List (-pw)

Use this option to change the number of columns per line of a list from its default (70 characters).

■ Specifying the Number of Columns Per Line of a List (-pw)

[Syntax]

-pw < Number of columns >	(Default: 70)
---------------------------	-----------------

[Parameter]

<Number of columns>

Specify between 70 and 1023 inclusive.

[Description]

Specify the number of columns to be printed per line of a list file.

Use this option when the default number of columns causes a long symbol name, section name, or module name to extend over two lines, making it difficult to comprehend.

At the default number of columns (70), the number of characters displayable per line are as follows:

- Module name : 22 characters
- Section name : 20 characters
- Symbol name : 29 characters

If the -m option is not specified, this option is invalid.

[Example]

```
flib896s syslib.lib -m libx.mp2 -pw 80
```

The number of columns per line of a list is changed to 80.

In this case, the number of characters displayable per line for each name are as follows:

- Module name : 32 characters
- Section name : 30 characters
- Symbol name : 34 characters

Note:

In a list of a librarian, specifying the number of columns per line in the -pw option changes the number of characters displayed per line for the module name, section name, and symbol name. Since one line of a symbol name is separated on the screen into two fields on the left and right, specifying twice the number of characters in the longest symbol name plus 12 will display an easy-to-understand list.

10.2.10 Creating a Backup File (-b)

Editing a library file causes its contents to be lost.

Use the -b option to store a backup of the library file before editing it.

■ Creating a Backup File (-b)

[Syntax]

-b

[Parameter]

None

[Description]

When a librarian edits a library file by adding or deleting a module, the original contents of the file are changed and lost.

Use this option to create a backup file of the original.

The backup file has the ".bak" extension.

A backup is created for only one generation of a library file. If you edit an important library file, you must create a backup for yourself before using a librarian.

[Example]

```
f1ib896s syslib -a putc.obj -b
```

syslib.lib before editing is stored as syslib.bak after editing.
To syslib.lib after editing, putc.obj and getc.obj are added.

10.2.11 Inhibiting the Creation of a Backup File (-Xb)

Use the -Xb option to cancel the -b option used to obtain a backup.

■ Inhibiting the Creation of a Backup File (-Xb)

[Syntax]

-Xb	(Default)
-----	-------------

[Parameter]

None

[Description]

By default, a librarian does not create a backup for the library file to be edited. This is the same as specifying the -Xb option.

Specify this option to nullify the -b option when it is specified.

[Example]

All of the following three specifications result in the same processing.

```
flib896s syslib -a putc.obj,getc.obj
flib896s syslib -a putc.obj,getc.obj -Xb
flib896s syslib -b -a putc.obj,getc.obj -Xb
```

syslib.lib before editing is deleted after editing.

To syslib.lib after editing, putc.obj and getc.obj are added.

10.2.12 Checking the Contents of a Library File (-c)

Use this option to briefly check the contents of a library file.

■ Checking the Contents of a Library File (-c)

[Syntax]

-c

[Parameter]

None

[Description]

The following two items are checked.

- Whether an unsolved external reference symbol exists in the library

A linker, when a module taken in from a library contains an unsolved external reference symbol, first tries to solve the symbol in the same library file, assuming that a module containing a defined symbol exists in the same library file.

The program checks the correspondence between external reference and defined symbols in a library file. If any external reference symbols without the corresponding external definition symbol is contained, it outputs a message.

- Whether a module with debugging information exists

If a module to be registered with the -g option specification in a library file contains debugging information, it is registered in the library with the debugging information unremoved.

The above function is provided so that a module registered in a library may be debugged. However, after the operation check, the debugging information will no longer be required.

The program checks whether a module with debugging information is registered in a library and, if so, outputs a message.

[Example]

```
flib896s syslib.lib -c
```

The contents of syslib.lib are checked.

Note:

This option cannot be specified with other options.
Specify this option alone as shown in the above example.

10.2.13 Optimizing the Contents of a File (-O)

Any debugging information contained in an object module registered in a library is removed.

■ Optimizing the Contents of a File (-O)

[Syntax]

-O

[Parameter]

None

[Description]

An object module registered with a debugging information block in a library is re-registered without it. Since debugging information occupies a very large part of an object module file, you can make a library file significantly smaller by deleting the debugging information.

[Example]

```
flib896s syslib -O
```

Debugging information is deleted from the syslib.lib file.

Note:

This optimization option cannot be specified with other options. Specify this option alone as shown in the above example.

10.2.14 Specifying to Output Debugging Information (-g)

Use this option so as not to delete debugging information when registering an object module in a library file.

■ Specifying to Output Debugging Information (-g)

[Syntax]

-g

[Parameter]

None

[Description]

A librarian usually removes any debugging information that may be contained in an object module before registering it in a library file. Use this option to register the specified object without any changes regardless of whether or not it has debugging information.

To delete debugging information after creating a library, use the optimization option -O to recreate a library file.

[Example]

```
flib896s syslib.lib -a inchar,outchar -g
```

inchar.obj,outchar.obj is registered with debugging information unremoved, if any are contained, in a library file.

10.2.15 Specifying not to Output Debugging Information (-Xg)

Use the -Xg option to nullify the -g option used to specify not to delete debugging information.

■ Specifying not to Output Debugging Information (-Xg)

[Syntax]

-Xg	(Default)
-----	-------------

[Parameter]

None

[Description]

A librarian usually removes any debugging information that may be contained in an object module before registering it in a library file. This is the same as specifying this -Xg option.

Specify this option to nullify the specification of the -g option.

You may use the optimization option -O to delete debugging information in a batch after creating a library.

[Example]

The following three specifications result in the same processing.

```
flib896s syslib.lib -a inchar,outchar
flib896s syslib.lib -a inchar,outchar -Xg
flib896s syslib.lib -g -a inchar,outchar -Xg
```

Debugging information contained in inchar.obj,outchar.obj is not registered in a library file.

10.2.16 Specifying a Target CPU (-cpu)

Use this option to specify a target CPU.

Use an MB number to specify a target CPU of the programs to be combined into a library file.

■ Specifying a Target CPU (-cpu)

[Syntax]

`-cpu < MB number >`

[Parameter]

<MB number>

MB number of a target CPU

[Description]

A target CPU of the programs to be combined into a library file is specified using an MB number.

[Example]

```
flib896s  syslib.lib -a inchar,outchar -cpu MB89123A
flib896s  syslib.lib -a inchar,outchar -cpu MB89051
```

Note:

To create a library, you must specify a target CPU using this option.
This option may not be omitted.

10.2.17 Specifying CPU Information File (-cif)

This specifies to determine the selecting of the target CPU information from the CPU information file.

■ CPU Information File Specification (-cif)

[Syntax]

```
-cif < CPU information file name >
```

[Parameters]

< CPU information file name >

CPU information file name of the target

[Description]

This determines the getting of the target CPU ROM/RAM region information from the CPU information file.

[Example]

```
flib896s syslib.lib -a inchar,outchar -cpu MB89123A  
-cif C:\Softune\lib\896\cpu_info\MB89501.csv
```

Note:

SOFTUNE Tools get CPU information by referencing the CPU information file. Reference to a CPU information file different between the related tools may cause an error to the program to be created. The CPU information file that comes standard with SOFTUNE Tools is located at:

Installation place directory\lib\896\896.csv

When installing the compiler assembler packs in a different directory and using the compiler, assembler and linkage kit instead of SOFTUNE workbench, specify -cif so that each tool can reference the same CPU information file.

10.2.18 Object Mixing Check Specification (-objmixchk)

This option checks whether F²MC-8L objects and F²MC-8FX objects are mixed or not. If these are mixed, an error occurs.

■ Object Mixing Check Specification (-objmixchk)

[Syntax]

`-objmixchk`

[Parameter]

None

[Description]

This option checks whether F²MC-8L objects and F²MC-8FX objects are mixed or not. The mix check of objects is performed at default.

If the objects are mixed, the error (E4407U: Invalid module : conflict CPU type (file name)) is issued.

[Example]

```
flib896s -cpu MB89051 -o library.lib -a obj_1.obj -r obj_2.obj -d obj_3.obj ...  
flib896s -cpu MB89051 -objmixchk -o library.lib -a obj_1.obj -r obj_2.obj -d obj_3.obj ...
```

If obj_1.obj and obj_3.obj are the F²MC-8FX objects, the following errors are issued.

*** E4407U: Invalid module : conflict CPU type(obj_1.obj)

*** E4407U: Invalid module : conflict CPU type(obj_3.obj)

10.2.19 Object Mixing Check Inhibit Specification (-Xobjmixchk)

This option inhibits the mix check of the F²MC-8L objects and F²MC-8FX objects. Specify this option when mixing the objects.

■ Object Mixing Check Inhibit Specification (-Xobjmixchk)

[Syntax]

-Xobjmixchk

[Parameter]

None

[Description]

This option inhibits the mix check of the F²MC-8L objects and F²MC-8FX objects. Specify this option when mixing the objects.

[Example]

```
f1ib896s -Xobjmixchk -cpu MB89051 -o library.lib -a obj_1.obj -r obj_2.obj -d obj_3.obj ...
f1ib896s -cpu MB89051 -objmixchk -o library.lib -a obj_1.obj -r obj_2.obj -d obj_3.obj
-Xobjmixchk ...
```

If the F²MC-8L objects and F²MC-8FX objects are mixed, processing is performed without issuing any error.

CHAPTER 11

LIST FORMATS OF LIBRARIAN

This chapter explains the configuration of a list file of a librarian.

- 11.1 Contents of Information in a List File
- 11.2 List of Module Names
- 11.3 Detailed Information of a Module
- 11.4 External Defined and Reference Symbol Information in a Library

11.1 Contents of Information in a List File

In a list file of a librarian, the contents of a library file are output in the following five groups.

- Module name
- Section information of each module
- External definition symbol information for each module
- External reference symbol information for each module
- External defined and reference symbol information for all the modules

To output a list, you must specify the **-m** option and the **-dt** option.

■ Configuration of a List File

Figure 11.1-1 shows the configuration of a list file.

Figure 11.1-1 Configuration of a List File

< List header >	
Library file name	Only -m
Numbers of registered modules and external definition symbols	
<Number of module >	
▪ Registered module name	Only -m
< Section information of each module >	-dt s specified
< External reference symbol name information for each module >	-dt r specified
< External definition symbol name information for each module >	-dt d specified
:	
:	
< External defined and reference symbol name information for all modules >	-dt a specified

11.2 List of Module Names

In the default list output of a librarian (when the -dt option is not specified), only the registered module names in a library file are displayed.

■ List Output Overview

If the -m option is specified, the contents of a library file are displayed.

Figure 11.2-1 shows the format of a librarian list.

Figure 11.2-1 Format of a Librarian List (Default)

```
*1 Library File Name : sample.lib
*2 Number of Modules : 3
*3 Number of Symbols : 9
*4 Librarian Name : F2MC-8L/8FX Family SOFTUNE Librarian
*5 Library Creation Date 1997-03-01 14:23:50
*6 Library Revision Date 1997-04-17 09:41:15

*7 [ Module Name ]      [ Entry Date ]      [ Creation Date ]
   ModuleA             1997-03-01 14:23:50    1996-03-19 10:03:21
   ModuleB             1997-04-17 09:41:15    1996-10-07 20:18:58
   ModuleC             1997-03-01 14:23:50    1997-02-23 15:15:00
```

*1 :Library file name
*2 :Number of modules registered in a library file (expressed in decimals)
*3 :Number of external definition symbol registered in a library file
(expressed in decimals)
*4 :Librarian name
*5 :Date and time when the library file is first created
*6 :Date and time when the library file is last updated Same as *5 for a
new file
*7 :[Module Name] Registered module name (in alphabetical order)
For a module name, each line displays as many
characters as (Page width-48).
The default (-pw 70) is 22 characters.
[Entry Date] Date and time when the module is registered in a library file
[Creation Date] Date and time when the module is created

11.3 Detailed Information of a Module

Detailed information of a module comes in the following three groups. Use the **-dt** option to specify outputting the detailed information.

- Section information (-dt s)
- External definition symbol information (-dt d)
- External reference symbol information(-dt r)

■ List Output Overview

Figure 11.3-1 shows the format of a librarian list (when detailed information is specified).

Figure 11.3-1 Format of a Librarian List (When Detailed Information is Specified)

Library File Name : sample.lib		
Number of Modules : 3		
Number of Symbols : 3		
Librarian Name : F2MC-8L/8FX Family SOFTUNE Librarian		
Library		
Creation Date 1997-03-01 14:23:50		
Library Revision Date 1997-04-17 09:41:15		
	[Module Name]	[Entry Date]
	ModuleA	1997-03-01 14:23:50
		[Creation Date]
		1996-03-19 10:03:21
*1	-- Section -	-- Type -
	code	code
	data	data
		-- Size -
		0x000002E8
		0x0000006A
*2	-- Ext_Ref Symbol(s) -	
	p_text	tx_len
*3	-- Ext_Def Symbol(s) -	
	prtext	
	[Module Name]	[Entry Date]
		[Creation Date]
		:
		:

*1 : Output by the s parameter in the -dt option.
This is the information of sections in a module.
The section name, section attribute, and size are displayed.
For a section name, each line displays as many characters as (Page width -50).

*2 : Output by the r parameter in the -dt option.
Each line displays two of the external reference symbol names in a module.

*3 : Output by the d parameter in the -dt option.
Each line displays two of the external definition symbol names in a module.
For an external symbol name, each line displays as many characters as ((Page -12) /2).

11.4 External Defined and Reference Symbol Information in a Library

The external defined and reference symbol information of all the modules registered in a library file may be displayed. Use the **-dt** option to specify to output this information. (**-dt a**)

■ List Output Overview

Figure 11.4-1 shows the format of a librarian list (when detailed information is specified).

Figure 11.4-1 Format of a Librarian List (When Detailed Information is Specified)

```

Library File Name : sample.lib
Number of Modules : 3
Number of Symbols : 3
Librarian Name : F2MC-8L/8FX Family SOFTUNE Librarian
Library Creation Date 1997-03-01 14:23:50
Library Revision Date 1997-04-17 09:41:15

[ Module Name ]      [ Entry Date ]      [ Creation Date ]
ModuleA              1997-03-01 14:23:50  1996-03-19 10:03:21
                    :
                    :

*1 [ ALL Ext_Def Symbol(s) ]
   chr1get           p_text
   prtext

*2 [ ALL Ext_Ref Symbol(s) ]
   chr_get           tx_len

```

*1:Output by the **a** parameter in the **-dt** option.

Each line displays two of the external definition symbol names in the entire library file.

*2:Each line displays two of the external reference symbols without corresponding external definition symbols in the library file.

For an external symbol name, each line displays as many characters as $((\text{Page width} - 12) / 2)$.



CHAPTER 12

RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN

This chapter covers the restrictions and questions and answers on using a librarian.

12.1 Restrictions on a Librarian

12.2 Questions and Answers on Using a Librarian



12.1 Restrictions on a Librarian

Section 12.1 describes restrictions on using a librarian, such as those on the number of modules and external symbols that can be registered in a library file and other precautions.

■ Restrictions on a Librarian

There are two restrictions on using a librarian:

- Number of modules that can be registered in a library file: 65535 maximum
- Number of external definition symbols that can be registered in a library file: 65535 maximum

■ Precautions on the Required Disk Space

Before editing an existing library file and creating a backup file, check that sufficient disk space is available to store the new library file and the existing library.

■ Precautions on Specifying Options

Specify the options for checking the contents of a library file (-c) and optimizing the contents of a file (-O) alone, respectively.

These options may not be specified together with other options.

12.2 Questions and Answers on Using a Librarian

Section 12.2 shows the questions and answers on using a librarian.

- Questions and answers on creating a library file

Q.	What is the format of a file that can be registered in a library file?
A.	It is an object module that an assembler outputs. This is a file created with the .obj extension by default.
Example.	<pre>fasm896s file1 → file1.obj is output. fasm896s file2 → file2.obj is output. flib896s libfile -a file1.obj, file2.obj</pre>

Q.	I want to debug a module taken in from a library file that I have created because it seems to have problems. But I cannot access the symbol information.
A.	You can access symbol information for debugging only in an object module with debugging information. You must replace the object module with the one with debugging information. When you create a library out of the object modules that may need to be debugged, it is recommended to register them with debugging information (using the -g option). When debugging is complete, you can delete debugging information (using the -O option).
Example.	<pre>fasm896s file1 -g → file1.obj with debugging information is output. flib896s libfile -r file1 -g → libfile.lib with debugging information is output. flib896s libfile -O → libfile.lib without debugging information is output.</pre>

Q.	I want to create a library out of the subroutines that I created for general purposes, but there are so many object modules and I do not want to specify all the file names.
A.	Use a wild card to specify object modules to be added to a library (in the -a option) or replaced with the one in a library (in the -r option).
Example.	<pre>flib896s libfile -a "*.obj" → All the files with the ".obj" extension are registered.</pre>



Q.	I forgot what was contained in a library file that I created a while ago. How can I find what modules are registered in it?
A.	Use the -m option to see what is contained in a library file. Specify "-m File-name" to create a list file with the .mp2 extension by default. If the contents output by the -m option is not sufficient for you, use the -dt option together to obtain more detailed information.
Example.	<pre>f1ib896s libfile -m libdoc → A list file, libdoc.mp2 is output. f1ib896s libfile -m libdoc -dt a,s → A list file with detailed information, libdoc.mp2 is output.</pre>

Q.	I want to check the contents of a library file, but they need not be stored in a file. How can I simply display them on the screen?
A.	Specify a hyphen (-) instead of a file name in the -m option to display the contents in the standard output.
Example.	<pre>f1ib896s libfile -m - f1ib896s libfile -m - -dt a,s f1ib896s libfile -a file3.obj -m -</pre>

Q.	I output the contents of a library file into a list, but some long symbol names are displayed over two lines and difficult to comprehend. How can I evade this?
A.	Use the -pw option to increase the number of columns to be displayed per line. It is 70 columns by default. If four characters are in the second line, add twice that number, i.e., eight to specify the number of columns as 78. Then the names will fit in one line.
Example.	<pre>f1ib896s libfile -m libdoc -pw 80</pre>

Q.	I replaced modules in a library file with the new ones only to realize later that I had registered some of them by mistake. Since I did not keep any backup for the library file before the replacement, I had a hard time restoring it.
A.	A librarian allows you to create a backup file for one generation (with the ".bak" extension) using the -b option. Naturally, it is recommended to create a backup of a library before editing it. However, specify the -b option as required.
Example.	<pre>f1ib896s libfile -r file1,file2 -d mod4 -b</pre>

PARTIV OBJECT FORMAT CONVERTERS

Part 4 describes the types of object format converters, list of options, functions, and conversions of object formats.

- CHAPTER 13 SPECIFICATIONS OF AN OBJECT FORMAT CONVERTER
- CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER
- CHAPTER 15 LOAD MODULE CONVERTER (f2ms, f2hs, f2is, f2es)
- CHAPTER 16 FORMAT ADJUSTER (m2ms, h2hs)
- CHAPTER 17 THE BINARY CONVERTER (m2bs, h2bs)
- CHAPTER 18 OTHER CONVERTERS
- CHAPTER 19 RESTRICTIONS AND QUESTIONS AND ANSWERS ON AN OBJECT FORMAT CONVERTER

CHAPTER 13

SPECIFICATIONS OF AN OBJECT FORMAT CONVERTER

Chapter 13 gives an overview and describes the types of object format converters.

An object format converter is a tool used to convert an object format.

- 13.1 Outline of Object Format Converter
- 13.2 Types of Object Format Converters
- 13.3 Executing an Object Format Converter

13.1 Outline of Object Format Converter

The object format converter processes four types of the following file formats.

- Absolute format load module of linker output
- S format
- HEX format
- Binary data file

■ Outline of Object Format Converter

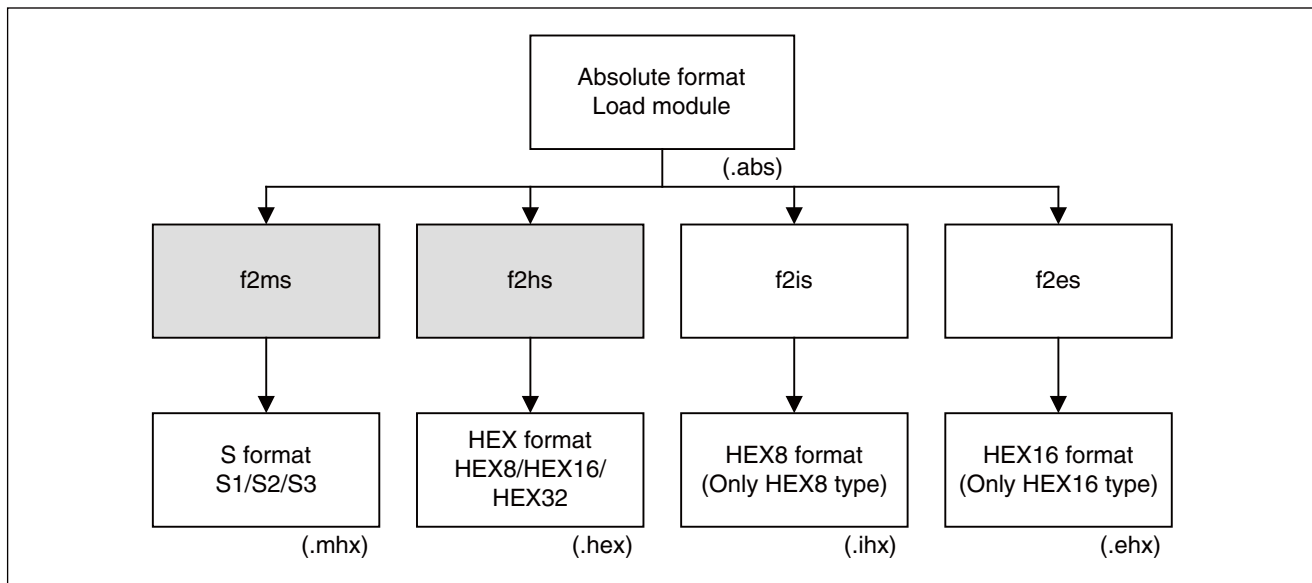
Object format converters include four types of converters; load module converter, adjuster (adjusting tool), binary converter and a converter.

● Load module converter

This converter converts the absolute format load module of the linker output to a general-purpose format.

Figure 13.1-1 shows input and output of the load module converter.

Figure 13.1-1 Input and Output of the Load Module Converter

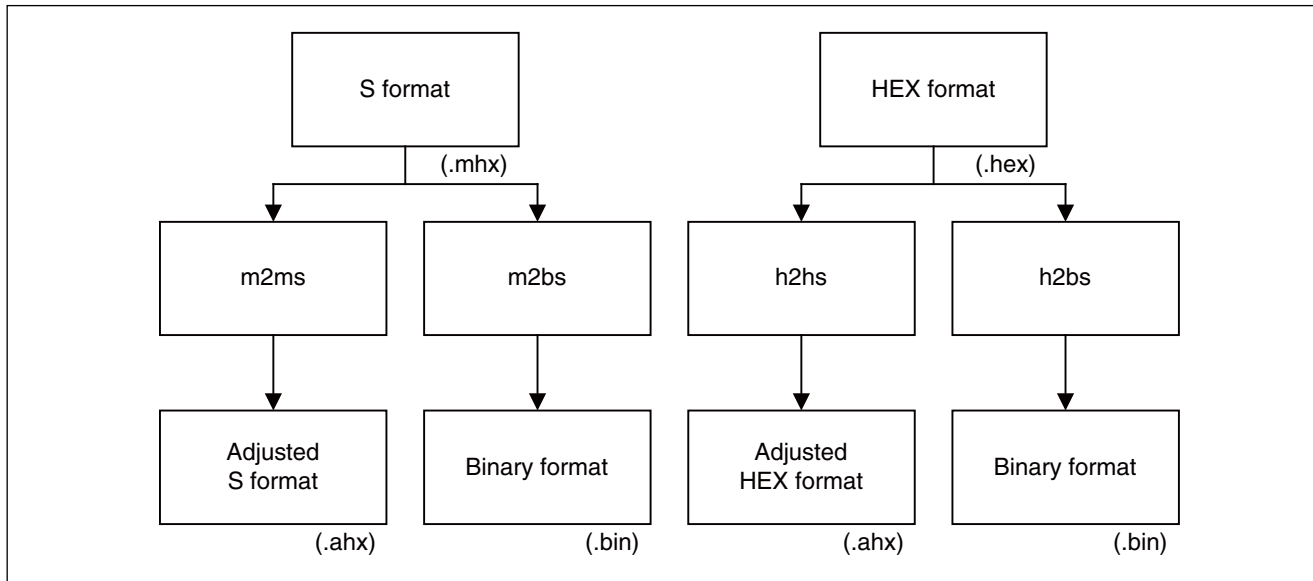


● Adjuster, Binary converter

The adjuster adjusts the S format or Hex format. The binary converter converts the S format or Hex format into a binary format.

Figure 13.1-2 shows input and output of the adjuster and binary converter.

Figure 13.1-2 Input and Output of the Adjuster and Binary Converter

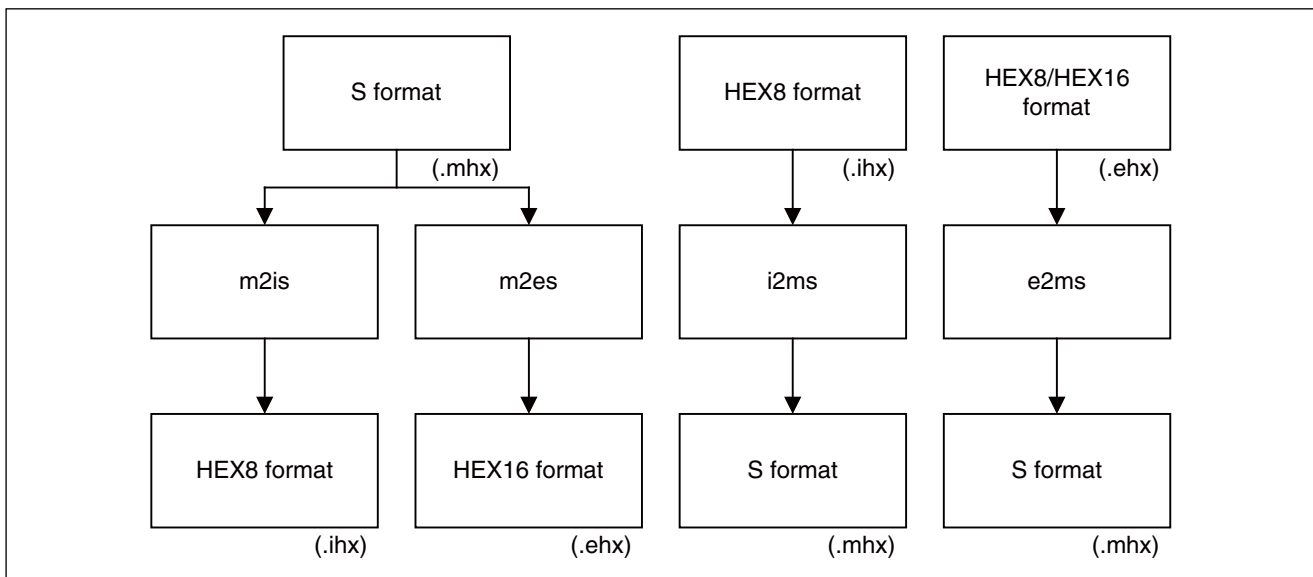


● Converter

The converter converts the S format and HEX8/HEX16 format with each other.

Figure 13.1-3 shows input and output of the converter.

Figure 13.1-3 Input and Output of the Converter



13.2 Types of Object Format Converters

The command for an object format converter is **x2ys**, where the **x** represents the object format of an input file and the **y** represents the object format of an output file.

An alphabetic character assigned in the **x** and **y** format means one of the following formats:

- **f**: Absolute format load module that a linker outputs
- **m**: S format
- **h**: HEX format (HEX8/HEX16/HEX32)
- **b**: Binary data format
- **i**: HEX8 format (Only HEX8)
- **e**: HEX16 format (Only HEX16)

■ Types of Load Module Converters

Use the commands in Table 13.2-1 to convert an object format.

f2ms is used for conversion to the S format.

f2hs is used for conversion to the HEX format.

It is enabled to convert to the HEX8 format using **f2is** and to the HEX16 format using **f2es**, but it is recommended to use **f2hs** corresponding HEX8/HEX16/HEX32.

Table 13.2-1 Conversions Made by Load Module Converters

Command name	Conversion
f2ms	Absolute format load module to S format
f2hs	Absolute format load module to HEX8/HEX16/HEX32 format
f2is	Absolute format load module to HEX8 format
f2es	Absolute format load module to HEX16 format

■ Format Adjuster

The Format Adjuster formats an object file in the S format and HEX format. For details, see "CHAPTER 16 FORMAT ADJUSTER (m2ms, h2hs)".

■ Binary Converter

An object file in the S format and HEX format is converted to binary data (a memory image) and output into a file. For details, see "CHAPTER 17 THE BINARY CONVERTER (m2bs, h2bs)".

■ Types of Other Converters

Use the commands in Table 13.2-2 to convert an object format.

Table 13.2-2 Conversions Made by Other Converters

Command name	Conversion
m2is	S format to HEX8 format
m2es	S format to HEX16 format
i2ms	HEX8 format to S format
e2ms	HEX16 format to S format



13.3 Executing an Object Format Converter

Simply specify a command name followed by an input file name to execute an object format converter.

■ Executing a Command of an Object Format Converter

Simply specify a command name followed by an input file name to execute the command.

```
x2ys < Input file name > [ Option ]
```

The specified <Input file name> is processed as the x format and a file in the y format is created.

A converter uses the following extensions by default to identify an object format from a file name:

- Absolute format load module : .abs
- S format : .mhx, ahx
- HEX8/HEX16/HEX32 : .hex, .aix
- Binary data format : .bin
- HEX8 : .ihx
- HEX16 : .ehx

The -ran option is always required to execute the Binary Converter and adjuster. For details, see Section "16.3.2 Specifying the Output Range (-ran)".

[Example]

```
f2ms sample
```

The absolute format load module that a linker outputs, sample.abs, is input and the sample.mhx file in the S format is output.

CHAPTER 14

COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER

This chapter explains the common options of an object format converter in detail.

- 14.1 List of Option of an Object Format Converter
- 14.2 Changing an Output File Name (-o)
- 14.3 Padding (-p)



14.1 List of Option of an Object Format Converter

Simply specify a command name followed by an input file name to execute an object format converter. Also, some options may be used.

■ Common Options of an Object Format Converter

For each command of an object format converter, the following common options may be used.

Table 14.1-1 lists types of common options of an object format converter.

Table 14.1-1 List of Common Options of an Object Format Converter

Function	Option	Remarks
Changing an output file name	-o	
Specifying padding data	-p	
Specifying not to read a default option file	-Xdof	* Common option
Specifying to read an option file	-f	* Common option
Specifying to display help messages	-help	* Common option
Specifying to output the version number and messages	-V	* Common option
Specifying not to output the version number and messages	-XV	* Common option
Specifying to display a termination message	-cmsg	* Common option
Specifying not to display a termination message	-Xcmsg	* Common option
Specifying to set the termination code to 1 when a warning occurs	-cwno	* Common option
Specifying to set the termination code to 0 when a warning occurs	-Xcwno	* Common option

To display a brief explanation of an option, enter the command name alone or use the -help option.

```
x2ys
x2ys -help
```

14.2 Changing an Output File Name (-o)

The directory in which to create an output file after conversion and the file name are changed from the default.

■ Changing an Output File Name (-o)

[Syntax]

```
-o < Object file name >
```

[Parameter]

<Object file name> Output file name

[Description]

Specify this option to change the output file name after conversion.

Specify this option with a path name to change also the output destination directory.

If this option is omitted, the output file name will be the same as the input file name. However, its extension will be changed to the default of one of the formats used after conversion.

If the extension is omitted in the <Object file name> specification, the default extension is added.

One of the following six default extensions is used for each format:

- Absolute format load module .abs
- S format .mhx .ahx
- HEX8/HEX16/HEX32 .hex, .aix
- Binary data file .bin
- HEX8 .ihx
- HEX16 .ehx

[Example 1]

```
f2ms ccp903 (Example of not using the -o option)
```

The absolute format load module ccp903.abs is input and ccp903.mhx in the S format is output. The following four examples are equivalent to the above.

```
f2ms ccp903.abs -o ccp903.mhx
f2ms ccp903.abs -o ccp903
f2ms ccp903 -o ccp903.mhx
f2ms ccp903 -o ccp903
```

[Example 2]

```
f2ms ccp903 -o ccp903.hex
```

The output file name is changed to ccp903.hex.

[Example 3]

```
f2ms ccp903 -o ..\hex\ccp903m.hex
```

The current output destination directory is changed to ..\hex and the output file name to ccp903m.hex.



Note:

If the -sp option is specified for Binary Converter, the <Object file name> is evaluated differently. The <Object file name> is assumed to have no extension and an extension is unconditionally added to the file name specified in <Object file name>.

For example, if "binary.bin" is specified as the object file name, the output file names will be "binary.bin.b01", "binary.bin.b02", ..., "binary.bin.bxx".

14.3 Padding (-p)

**The specified range of addresses is padded with data of a specified value.
With the Binary Converter and adjuster, the portion of an file containing no data is padded with data of a specified value.**

■ Padding (-p)

[Syntax]

```
-p < Value > , < Starting address > , < Ending address >
```

[Parameters]

<Value> One-byte data

<Starting address> Starting address at which to set <Value>

* The parameter cannot be used in binary converter and adjuster.

<Ending address> Ending address at which to set <Value>

* The parameter cannot be used in binary converter and adjuster.

[Description]

Embed the specified address range with the specified value data.

Set only for <value> with the binary converter and adjuster.

Embeds with value data specified by locations that do not exist for data in the file with the binary converter and adjuster.

[Example 1]

```
f2ms ccp903 -p 0xEF, 0x1FE4, 0x1FFF
```

An absolute format load module is converted into the S format.

At this time, the data at the addresses 0x1FE4 through 0x1FFF is created as the 0xEF data and added to the end of a S format file.

[Example 2]

```
f2ms ccp903 -p 0xEF, 0x1FE4, 0x1FFF -adjust
```

The error is generated at specifying adjust(-adjust) because the starting/ending addresses are specified by the padding option(-p).

```
f2ms ccp903 -p 0xEF -adjust
```

An absolute format load module is converted into the S format. At this time, pad the portion which the data does not exist using data of 0xEF.

[Example 3]

```
m2bs ccp903 -ran 0x0, 0x1FFF -p 0xEF
```

The data at the addresses 0x0 through 0x1FFF in the S format file is converted to a binary image.

At this time, the portion of an S format file containing no data is padded with data of 0xEF.



CHAPTER 15

LOAD MODULE CONVERTER

(f2ms, f2hs, f2is, f2es)

This chapter explains the Load Module Converter.

- 15.1 Outline of Load Module Converter
- 15.2 List of Options of the Load Module Converter
- 15.3 Details of Load Module Converter Options
- 15.4 f2ms (Converting an Absolute Format Load Module into the S Format)
- 15.5 f2hs (Converting an Absolute Format Load Module into the HEX Format)
- 15.6 f2is (Converting an Absolute Format Load Module into the HEX8 Format), f2es (Converting an Absolute Format Load Module into the HEX16 Format)

15.1 Outline of Load Module Converter

The load module converter converts an absolute format load module to the S format or Hex format which are general-purpose formats.

■ Outline of Load Module Converter

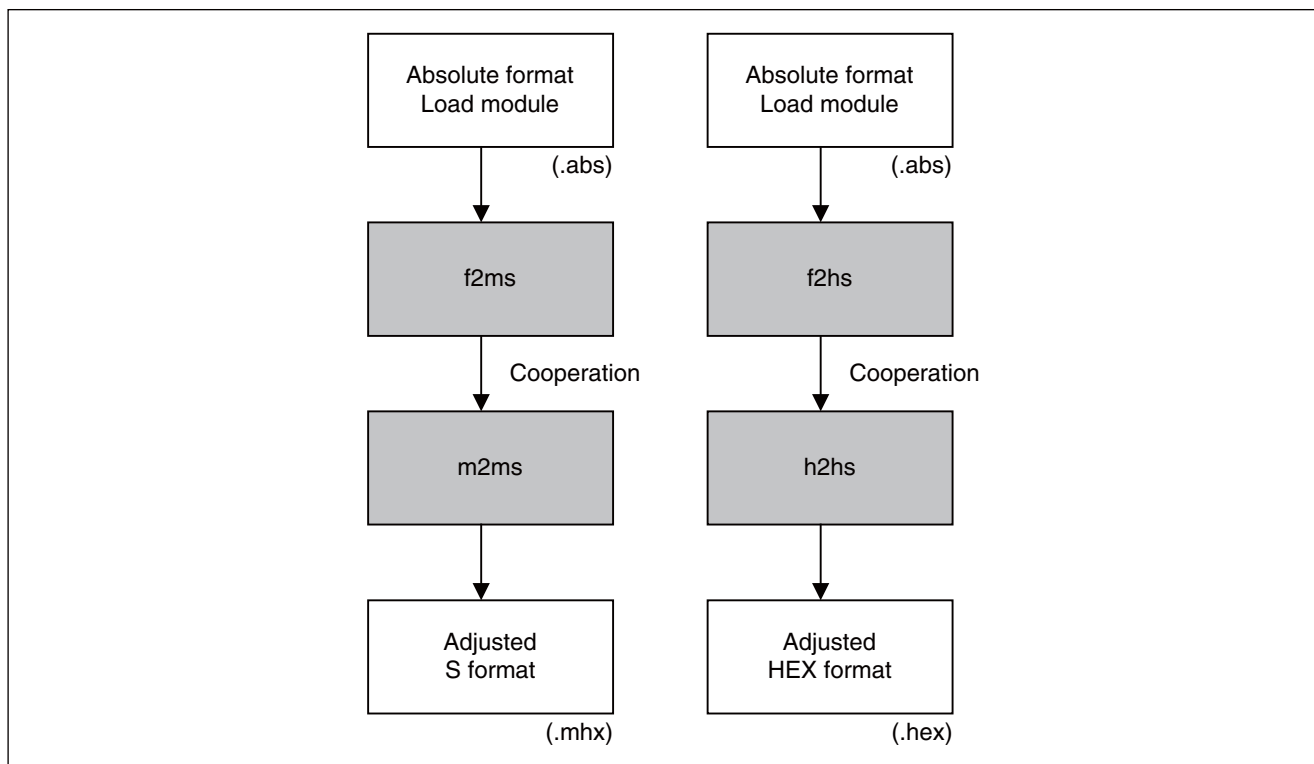
The load module converter converts the absolute format load module to an S format or Hex format which are general-purpose formats.

f2ms can be used to convert to the S format. f2hs can be used to convert to the Hex format.

f2is converts to Hex 8 format, and f2es to Hex 16 format, but if f2hs is used, it is possible to convert to all Hex 8, Hex 16 and Hex 32 Hex formats.

As shown in Figure 15.1-1, by specifying the adjust option (-adjust), f2ms and f2hs can adjust the output file together with adjuster.

Figure 15.1-1 Outline of Load Module Converter



15.2 List of Options of the Load Module Converter

Options are available to specify Load Module Converter operations in more detail. The following lists the option names and function outlines.

■ List of Options of the Load Module Converter

The table below gives the option names and an overview of their functions.

For details on the parameters and functions required for an option, see the description on each option.

Table 15.2-1 List of Options of the Load Module Converter

Function	Option	Remarks
Changing an output file name	-o	* Common option of a converter
Specifying padding data	-p	* Common option of a converter
Specifying to output S1 format	-S1	Only f2ms
Specifying to output S2 format	-S2	Only f2ms
Specifying to output S3 format	-S3	Only f2ms
Specifying to output HEX8 format	-I16	Only f2hs
Specifying to output HEX16 format	-I20	Only f2hs
Specifying to output HEX32 format	-I32	Only f2hs
Specifying to output start address record	-entry	Only f2hs
Specifying to not output start address record	-Xentry	Only f2hs
Specifying to Adjust	-adjust	Only f2ms and f2hs
Specifying suppression not to read default option file	-Xdof	* Common option
Specifying option file name	-f	* Common option
Specifying display of help message	-help	* Common option
Specifying version number and startup message of program	-V	* Common option
Suppression not to output version number and startup message of program	-XV	* Common option
Suppression to output end message	-cmsg	* Common option
Suppression not to output end message	-Xcmsg	* Common option
Specifying to set the termination code to 1 when a warning occurs	-cwno	* Common option
Specifying to set the termination code to 0 when a warning occurs	-Xcwno	* Common option



15.3 Details of Load Module Converter Options

This section describes each of the options for the load module converter.

Note that options common to the linkage kit are described in "CHAPTER 3 COMMON OPTIONS", and options common to the converter are described in "CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER".

■ Output S Format Option (-S1/-S2/-S3)

Specifies the record used to output with S format. For details, see Section "15.3.1 Output S Format Option (-S1/-S2/-S3)".

■ Output HEX Format Option (-I16/-I20/-I32)

Specifies the record used to output with Hex format. For details, see Section "15.3.2 Output HEX Format Option (-I16/-I20/-I32)".

■ Start Address Output Option (-entry)

This outputs the start segment address record or start linear address record when outputting with Hex format. For details, see Section "15.3.3 Specifying to Output Start Address Record (-entry)".

■ Start Address Output Inhibit Option (-Xentry)

This specifies when inhibiting the start address record output. For details, see Section "15.3.4 Specifying not to Output Start Address Record (-Xentry)".

■ Adjust Option (-adjust)

This specifies to start the adjuster after outputting the S format or Hex format. For details, see Section "15.3.5 Specifying to Adjust (-adjust)".

15.3.1 Output S Format Option (-S1/-S2/-S3)

This option specifies the format used to output data.

■ Output S Format Option (-S1/-S2/-S3)

[Syntax]

-S1

-S2

-S3

[Parameter]

None

[Description]

Specifies the format used to output data.

f2ms outputs the data using either the S1 record, S2 record or the S3 record.

It will not output if both the S1 record and the S2 record are used.

The -S1, -S2 and -S3 options take effect when specified last. If these options are not specified, the f2ms command outputs data in mixed formats of S1/S2/S3 according to the data address.

[Precautions]

If the specification with this option conflicts with the output range, this option outputs an error and performs no processing.

The terminator record (S9 record, S8 record, S7 record) for output varies with the specification of this option. (See Table 15.3-1.)

Table 15.3-1 Output S Format Specification List

Specification	Range of data that can be output	Terminator record	Remarks
-S1	0x00000000-0x0000FFFF	S9 record	
-S2	0x00000000-0x00FFFFFF	S8 record	
-S3	0x00000000-0xFFFFFFFF	S7 record	



15.3.2 Output HEX Format Option (-I16/-I20/-I32)

This option specifies the Hex format used to output data.

■ Output HEX Format Option (-I16/-I20/-I32)

[Syntax]

-I16

-I20

-I32

[Parameter]

None

[Description]

This option specifies the HEX format used to output data. The f2hs command outputs data using either HEX8, HEX16 or HEX32 format. The -I16, -I20, and -I32 options take effect when specified last. If these options are not specified, the f2hs command outputs data in mixed formats of HEX8, HEX16, and HEX32 according to the data address.

[Precautions]

If the specification with this option conflicts with the output range, the f2hs command outputs an error and performs no processing.

Table 15.3-2 List of Output Hex Format Options

Specification	Range of data that can be output	Remarks
-I16	0x00000000-0x0000FFFF	HEX8 format
-I20	0x00000000-0x000FFFFF	HEX16 format
-I32	0x00000000-0xFFFFFFFF	HEX32 format

15.3.3 Specifying to Output Start Address Record (-entry)

Use this option to specify the start segment address record or the start linear address record are output.

This option can specify using f2hs only.

■ Specifying to Output Start Address Record (-entry)

[Syntax]

-entry

[Parameter]

None

[Description]

Use this option to specify the start segment address record or the start linear address record are output.

If there is no start address information in the input file, a warning will be output (W1504U: Start address information is not in an input file).

To the Table 15.3-3 described below, the start address record output according to specify Output HEX Format Option (-I16/-I20/-I32) and the data address.

Table 15.3-3 The Start Address Records of HEX Format Output

Specifying an Output HEX format	input data range	output start address record
-I16	-	The warning 'W1503U: -entry option was specified at the time of -I16 specification' is displayed. The start address record does not output.
-I20	0x00000000-0x000FFFFFFF	Start segment address record
-I32	0x00000000-0x000FFFFFFF	Start segment address record
	0x00100000-0xFFFFFFFF	Start liner address record

When the output HEX format option is omitted, the same processing as the HEX32 format output specification option (-I32) is specified.

[Example]

```
f2hs ccp903.abs -entry -I16
```

The warning '-entry option was specified at the time of -I16 specification' is displayed because s -entry option specified when the HEX8 format output specification option (-I16) was specified. The start address record does not output.

```
f2hs ccp903.abs -entry -I20
```

The start segment address record output.

```
f2hs ccp903.abs -entry -I32
```

The start segment address record is output when the data address size is from 0x0 to 0xFFFFFFFF.

The start Linear address record is output when the data address size is from 0x100000 to 0xFFFFFFFF.



15.3.4 Specifying not to Output Start Address Record (-Xentry)

It specifies to suppress the start segment address record and the start Linear address record output.

This option can specify only for the f2hs.

■ Specifying not to Output Start Address Record (-Xentry)

[Syntax]

-Xentry

[Parameter]

None

[Description]

It specifies to suppress the start segment address record and the start Linear address record output.

Use this option when you want to cancel the -entry option.

[Example]

```
f2hs -entry cpp903.abs -I20 -Xentry
```

The start address record output specification(-entry) is canceled and the start address record does not output.

15.3.5 Specifying to Adjust (-adjust)

This option automatically calls the HEX format adjuster for adjustment after converting the HEX format.

■ Specifying to Adjust (-adjust)

[Syntax]

```
-adjust
```

[Parameter]

None

[Description]

This option automatically calls the HEX format adjuster for adjustment after converting the load module to the HEX format.

The starting/ending addresses to be adjusted are automatically set.

If the starting/ending address parameters are specified with the padding (-p) option when this option is given, an error occurs.

When this option is specified, the option for the HEX format adjuster can also be specified.

[Example]

```
f2hs ccp903 -p 0xEF,0x1FE4,0x1FFF -adjust
```

Because the starting/ending addresses are specified with the padding option, an error occurs.

```
f2hs ccp903 -p 0xEF -adjust
```

This option converts the absolute format load module to an adjusted HEX format. In this case, it pads the location where no data exists with data named 0xEF.

15.4 f2ms (Converting an Absolute Format Load Module into the S Format)

An absolute format load module that is output by a linker is converted into the S format. Data at the addresses 0 through 0xFFFFFFFF is to be converted.

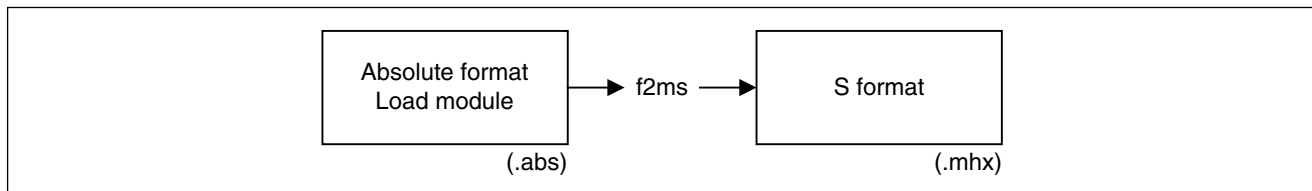
The f2ms command can process the absolute format load module for SOFTUNE V5/V6.

■ f2ms (Converting an Absolute Format Load Module into the S Format)

[Function]

The f2ms command reads the object data from an absolute format load module that is output by a linker and converts it into a S format file.

Figure 15.4-1 f2ms (Converting an Absolute Format Load Module into the S Format)



[Address]

The maximum values of addresses that can be represented in the S format are:

S1 type 0x0000FFFF

S2 type 0x00FFFFFF

S3 type 0xFFFFFFFF

The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive.

Since the S format supports the same range of addresses, conversions can be made without losing any data.

According to the address allocation, f2ms outputs the following records:

0x0000000 to 0x0000FFFF : S1 type

0x0010000 to 0x00FFFFFF : S2 type

0x1000000 to 0xFFFFFFFF : S3 type

15.5 f2hs (Converting an Absolute Format Load Module into the HEX Format)

An absolute format load module that is output by a linker is converted into the HEX format.

Data at the addresses 0 through 0xFFFFFFFF is to be converted.

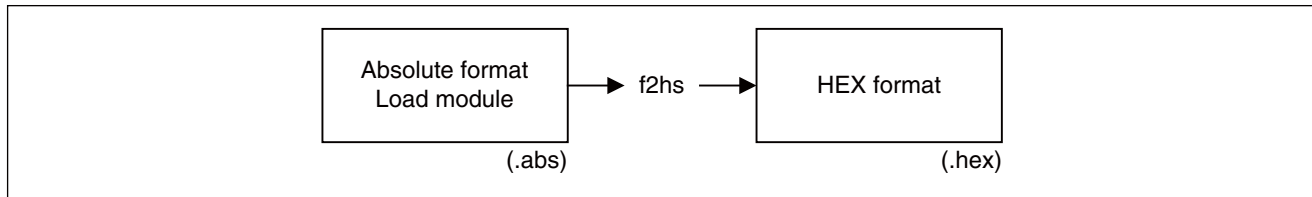
The f2hs command can process the absolute format load module for SOFTUNE V5/V6.

■ f2hs (Converting an Absolute Format Load Module into the HEX Format)

[Function]

The f2hs command reads the object data from an absolute format load module that is output by a linker and converts it into a HEX format file.

Figure 15.5-1 f2hs (Converting an Absolute Format Load Module into the HEX Format)



[Address]

The maximum values of addresses that can be represented in the HEX format are:.

HEX8 0x0000FFFF

HEX16 0x000FFFFF

HEX32 0xFFFFFFFF

The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive.

Since the HEX format supports the same range of addresses, conversions can be made without losing any data.

According to the address allocation, f2hs outputs the following records:

0x0000000 to 0x0000FFFF : HEX8

0x0010000 to 0x000FFFFF : HEX16

0x1000000 to 0xFFFFFFFF : HEX32



15.6 f2is (Converting an Absolute Format Load Module into the HEX8 Format), f2es (Converting an Absolute Format Load Module into the HEX16 Format)

The f2is command converts the absolute format load module of the linker output to a HEX8 format, and the f2es converts it to a HEX16 format.

The f2is command converts data at 0x0000 to 0xFFFF and the f2es command converts data at 0x00000 to 0xFFFFF.

These f2is and f2es commands can also process the absolute format load modules for SOFTUNE V5/V6.

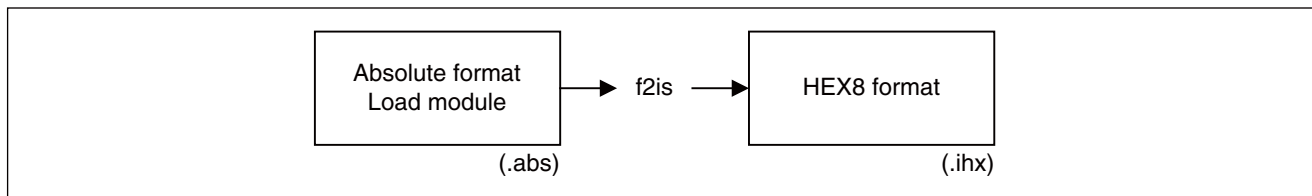
To keep compatibility with the previous versions, these commands are included in the linkage kit. Use of the f2hs command for conversion to a HEX format is recommended.

■ f2is (Converting an Absolute Format Load Module into the HEX8 Format)

[Function]

The f2is command reads only the object data part from the absolute format load module of the linker output, and converts it to a HEX8 format file.

Figure 15.6-1 f2is (Converting an Absolute Format Load Module into the HEX8 Format)



[Description]

The common options shown in "CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER" can be used for f2is.

[Address]

The maximum value of addresses that can be represented in the HEX8 format is 0xFFFF.

Note:

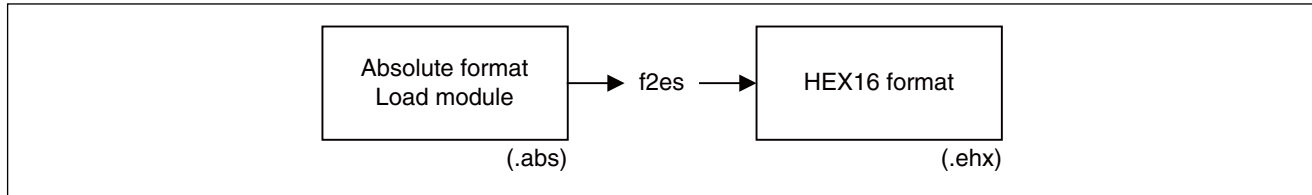
The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive. However, when the absolute format load module is converted into the HEX8 format, the data allocated to the addresses 0x10000 and higher are truncated. When using this command, be careful of the range of addresses in the conversion source. An HEX8 format file consists of data records and a trailer record.

■ f2es (Converting an Absolute Format Load Module into the HEX16 Format)

[Function]

The f2es command reads only the object data part from the absolute format load module of the linker output, and converts it to a HEX16 format file.

Figure 15.6-2 f2es (Converting an Absolute Format Load Module into the HEX16 Format)



[Description]

The common options shown in "CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER" can be used for f2es.

[Address]

The maximum value of addresses that can be represented in the HEX16 format is 0xFFFFF.

Note:

The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive. However, when the absolute format load module is converted into the HEX16 format, the data allocated to the addresses 0x100000 and higher are truncated.

When using this command, be careful of the range of addresses in the conversion source.

In the HEX16 format, an extended segment address record is used to represent the addresses 0x10000 and higher.

An extended segment address record in a file is valid until the next extended segment address record appears. If a data record appears without an extended segment address record, then the program calculates addresses assuming that the extended segment address is specified to be 0.

A starting address record is created at the beginning of an HEX16 format file.

CHAPTER 16

FORMAT ADJUSTER

(m2ms, h2hs)

This chapter explains Format Adjuster.

- 16.1 Overview of the Format Adjuster
- 16.2 List of Options of the Format Adjuster
- 16.3 Details of Options of the Format Adjuster

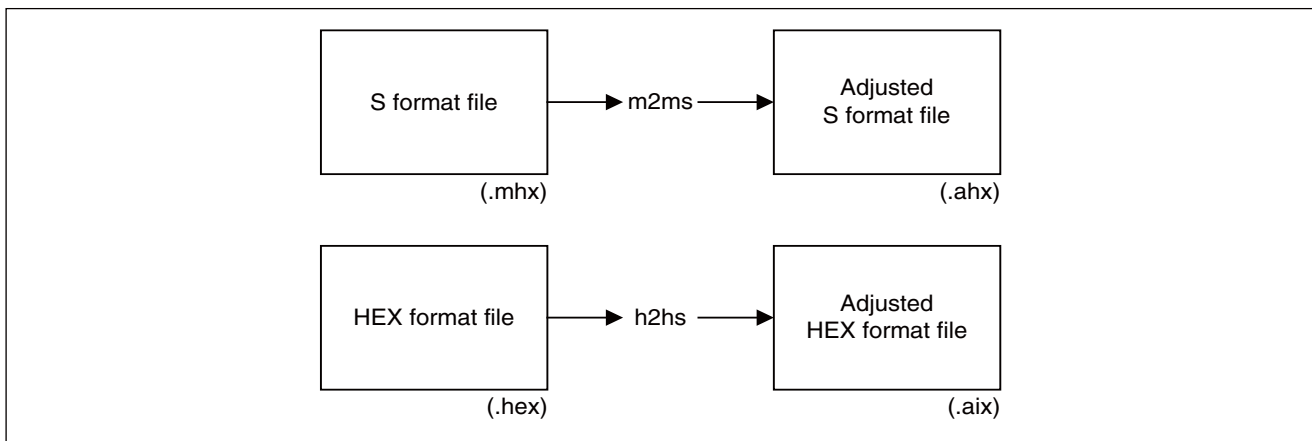
16.1 Overview of the Format Adjuster

The Format Adjuster sorts data created in the S format or HEX format in the ascending order of addresses, and causes each of the records to contain the specified number of data.

■ Overview of the Format Adjuster

The Format Adjuster causes each of the records in one file of the format to contain the specified number of data to unify the format. Figure 16.1-1 shows the concept of the Format Adjuster.

Figure 16.1-1 Overview of the Format Adjuster



The portion of an input format file containing no data is padded with 0xFF (default value).

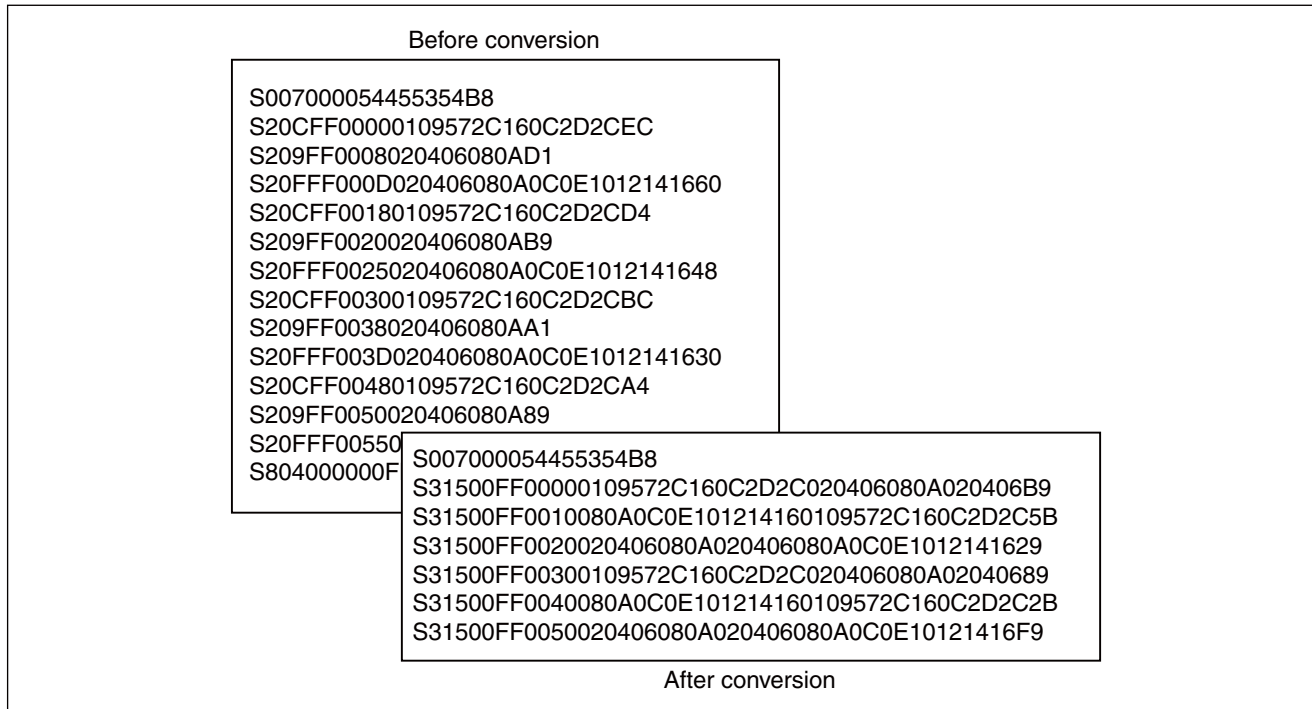
The output file is the contents of memory converted into the S format or HEX format without changes.

Use the padding option (-p option) to pad the portion of a format file containing no data with a specified value. For information on using this option, see Section "14.3 Padding (-p)".

■ Example of Operation

Use this tool to unify the lengths of data contained in one record if an existing S format file has records of varying lengths.

Figure 16.1-2 Example of Operation of the Format Adjuster



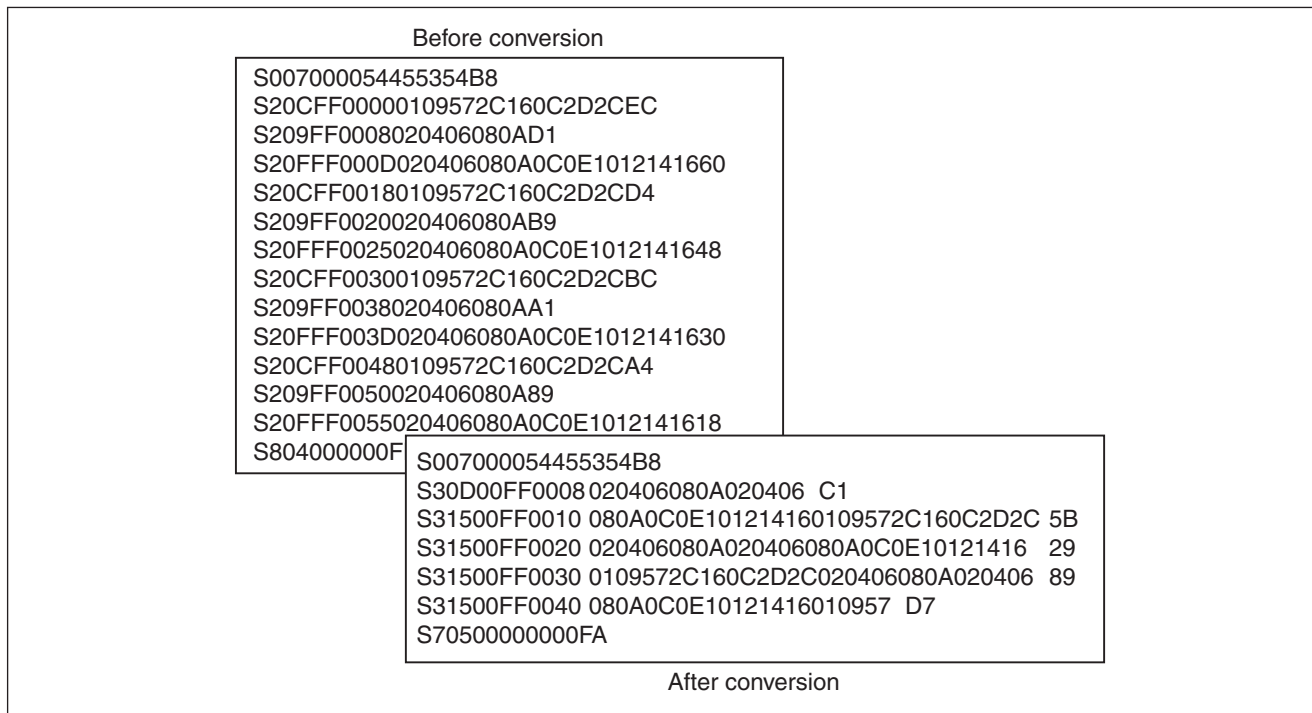
■ Functions of the Format Adjuster

The Format Adjuster has the following functions:

- The data is sorted in the ascending order of addresses.
- The portion of the specified range of addresses containing no data is padded with the specified data at the time of startup (with 0xff by default).
- As the starting address of a record, specify a value coordinated with the output data length specified at the time of startup.
 - If a remainder exists when the starting address specified for output is coordinated with the data length specified at the time of startup (if the data length in a record is 16 bytes and the starting address is not a multiple of 16), the first record appearing in the output information stores the data from the specified starting address to the address coordinated with the specified data length.
 - The second and later record starting addresses become the one coordinated with the specified length.
- If the input format information contains multiple terminator records, the entry address of the terminator record that appears at last is converted and output. Any other terminator records are deleted.
 - If the value of the entry address defined in the conversion source data is not within the range of data address after conversion, 0 is set in the terminator record after conversion.

The following shows an example of converting the address range 0xff0008 through 0xff004a at the record length of 16.

Figure 16.1-3 Example of Converting of the S Format Adjuster



16.2 List of Options of the Format Adjuster

Section 16.2 lists the names and functions of options of the Format Adjuster.

■ List of Options of the Format Adjuster

Table 16.2-1 lists the options of the Format Adjuster.

Table 16.2-1 List of Options of the Format Adjuster

Function	Option	Remarks
Changing an output file name	-o	* Common option of a converter
Specifying padding data	-p	* Common option of a converter
Specifying the output data length	-len	Default 16
Specifying the output range	-ran	need
Specifying the S1 format output	-S1	Only m2ms
Specifying the S2 format output	-S2	Only m2ms
Specifying the S3 format output	-S3	Only m2ms
Specifying the HEX8 format output	-I16	Only h2hs
Specifying the HEX16 format output	-I20	Only h2hs
Specifying the HEX32 format output	-I32	Only h2hs
Specifying to change the starting address	-ST	
Specifying not to read default option file	-Xdof	* Common option
Specifying to read option file	-f	* Common option
Specifying display of help message	-help	* Common option
Specifying to input version number and messages	-V	* Common option
Suppression not to output version number and messages	-XV	* Common option
Specifying to output end message	-cmsg	* Common option
Suppression not to output end message	-Xcmsg	* Common option
Specifying to set the termination code to 1 when a warning occurs	-cwno	* Common option
Specifying to set the termination code to 0 when a warning occurs	-Xcwno	* Common option

16.3 Details of Options of the Format Adjuster

This section explains the options of the Format Adjuster.

For information on the common options of Linkage kit, see "CHAPTER 3 COMMON OPTIONS". For information on the common options of a converter, see "CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER".

■ Specifying the Output Record Data Length (-len)

This option specifies the output record data length. For details, see Section "16.3.1 Specifying the Data Length in an Output Record (-len)".

■ Specifying the Output Range (-ran)

This option specifies the range of formatting. For details, see Section "16.3.2 Specifying the Output Range (-ran)".

■ Specifying an Output S Format (-S1/-S2/-S3)

This option specifies the S record to be used when data is output in the S format using m2ms. For details, see Section "16.3.3 Specifying an Output S Format (-S1/-S2/-S3)".

■ Specifying an Output HEX Format (-I16/-I20/-I32)

Specifies the record used to output with Hex format using h2hs. For details, see Section "16.3.4 Specifying an Output HEX Format (-I16/-I20/-I32)".

■ Specifying to Change the Starting Address of the Record (-ST)

This changes the starting address of the record used when outputting in the format. For details, see Section "16.3.5 Specifying Changes to the Starting Address (-ST)".

16.3.1 Specifying the Data Length in an Output Record (-len)

Use this option to specify the number of data to be output into a record of a format.

■ Specifying the Data Length in an Output Record (-len)

[Syntax]

```
-len <Data length>
```

[Parameter]

<Data length>

Select 16, 32, 64, or 128.

[Description]

Use this option to specify the number of bytes of data to be output into one record when a format file is formatted.

Specify 16, 32, 64, or 128 as the data length.

If this option is omitted, 16 is assumed to be specified in the processing.

[Precautions]

This option specifies the number of bytes of data contained in one record, not the record length itself.

[Example]

```
m2ms sfmtfile.mhx -len 32
```

sfmtfile.mhx is formatted and 32-byte data is output per record.

```
m2ms sfmtfile.mhx (An example of omitting the -len specification)
```

sfmtfile.mhx is formatted and 16-byte data is output per record.

```
m2ms sfmtfile.mhx -len 96
```

An error occurs because the specified data length is out of the specifiable range.

```
m2ms sfmtfile.mhx -len (An example of omitting all the parameters)
```

An error occurs because the data length specification is omitted.

16.3.2 Specifying the Output Range (-ran)

Use this option to specify the range of formatting using addresses.

■ Specifying the Output Range (-ran)

[Syntax]

```
-ran <Starting address> [ , <Ending address>]
```

[Parameters]

<Starting address>

Starting address

<Ending address>

Ending address

[Description]

Use this option to specify the range of formatting using addresses.

You must specify this option in order to convert.

Specify the starting and ending addresses between 0x0 and 0xffffffff inclusive.

The ending address may be omitted. If omitted, data as much as 64Kbytes from the starting address is formatted.

You cannot specify values that will make the conversion size more than 2Gbytes.

[Example]

```
m2ms sfmtfile.mhx (Example of not using the -ran option)
```

An error occurs because the output range is not specified.

```
m2ms sfmtfile.mhx -ran 0xD000,0xFFFF
```

The data in sfmtfile.mhx at the addresses 0xD000 through 0xFFFF is formatted.

```
m2ms sfmtfile.mhx -ran 0xD000 (An example of omitting the ending address)
```

The data in sfmfile.mhx as much as 64Kbytes from the addresses 0xD000 (0x0D000 through 0x1CFFF) is formatted.

```
m2ms sfmtfile.mhx -ran 0xFFFF,0xD000
```

An error occurs because the specified ending address is smaller than the starting address.

```
m2ms sfmtfile.mhx -ran (An example of omitting all the parameters)
```

An error occurs because the starting address is omitted.

16.3.3 Specifying an Output S Format (-S1/-S2/-S3)

Use this option to specify an S format to be used when data is output.
This option is for the S format adjuster (m2ms).

■ Specifying an Output S Format (-S1/-S2/-S3)

[Syntax]

-S1

-S2

-S3

[Parameter]

None

[Description]

Use this option to specify an record to be used when data contents are output.

The S Format Adjuster outputs the data contents using one of the S1, S2, and S3 records.

It never outputs data using both the S1 and S2 records.

If more than one of the -S1, -S2, and -S3 options are specified, the one most recently specified is valid. If none of the -S1, -S2, and -S3 options are specified, the S Format Adjuster outputs the data contents into the S3 record.

[Precautions]

If the specification in this option and the output range are not consistent with each other, the S Format Adjuster reports an error and performs no processing.

Specifying this option changes the terminator record to be used for output (S9, S8, and S7 records). (See Table 16.3-1)

Table 16.3-1 List of Output Record Specifications

Specification	Range of data that can be output	Terminator record	Remarks
-S1	0x00000000-0x0000FFFF	S9 record	
-S2	0x00000000-0x00FFFFFF	S8 record	
-S3	0x00000000-0xFFFFFFFF	S7 record	(Default)

[Example]

```
m2ms sfmtfile.mhx -ran 0xD000,0x10000 -S1
```

An error occurs because the output range is up to 0x10000, which cannot be represented in the S1 record.

```
m2ms sfmtfile.mhx -ran 0xE000,0xFFFF -S1
```

The data from 0xE000 through 0xFFFF is formatted and output in the S1 record.

```
m2ms sfmtfile.mhx -ran 0xE000,0xFFFF -S2
```

The data from 0xE000 through 0xFFFF is formatted and output in the S2 record.

```
m2ms sfmtfile.mhx -ran 0xE000,0xFFFF -S3
```

The data from 0xE000 through 0xFFFF is formatted and output in the S3 record.

16.3.4 Specifying an Output HEX Format (-I16/-I20/-I32)

Use this option to specify a HEX format to be used when data is output.
This option is for the HEX format adjuster(h2hs).

■ Specifying an Output HEX Format (-I16/-I20/-I32)

[Syntax]

-I16

-I20

-I32

[Parameter]

None

[Description]

Use this option to specify an record to be used when data contents are output.

The HEX Format Adjuster outputs the data contents using one of the HEX8, HEX16, and HEX32 format.

If more than one of the -I16, -I20, and -I32 options are specified, the one most recently specified is valid.

If none of the -I16, -I20, and -I32 options are specified, the HEX format adjuster outputs the data contents into the HEX32 format.

[Precautions]

If the specification in this option and the output range are not consistent with each other, the HEX Format Adjuster reports an error and performs no processing.

[Example]

```
h2hs hfmtfile.hex -ran 0xD000,0x10000 -I16
```

An error occurs because the output range is up to 0x10000, which cannot be represented in the HEX8 format.

```
h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I16
```

The data from 0xE000 through 0xFFFF is formatted and output in the HEX8 format.

```
h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I20
```

The data from 0xE000 through 0xFFFF is formatted and output in the HEX16 format.

```
h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I32
```

The data from 0xE000 through 0xFFFF is formatted and output in the HEX32 format.

16.3.5 Specifying Changes to the Starting Address (-ST)

This specifies the starting address used when outputting data. This is used to change the address of the data.

■ Specifying Changes to the Starting Address (-ST)

[Syntax]

```
-ST <Starting address>
```

[Parameters]

<Starting address>

Starting address

[Description]

This specifies the starting address used when outputting data.

The Format Adjuster determines the starting address of data normally using the starting address specified by the output range specification -ran.

Specifying this option changes the starting address when outputting.

[Example]

```
m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST 0x0000
```

This forms the data in the sfmtfile.mhx from 0xD000 to 0xFFFF address. It outputs this as data from 0x000 address.

```
m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST 0x10000
```

This forms the data in the sfmtfile.mhx from 0xD000 to 0xFFFF address. It outputs this as data from 0x10000 address.

```
m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST
```

(Example where parameters are omitted.)

There is an error where the starting address is omitted.

CHAPTER 17

THE BINARY CONVERTER

(m2bs, h2bs)

This chapter explains the conversion formats of the Binary Converter.

- 17.1 Outline of Binary Converter
- 17.2 List of Options of the Binary Converter
- 17.3 Details on Options of the Binary Converter

17.1 Outline of Binary Converter

The binary converter converts files output by the S format or Hex format into binary data files.

It not only simply converts to binary data, but also supports a split mode that separates files into several files for output.

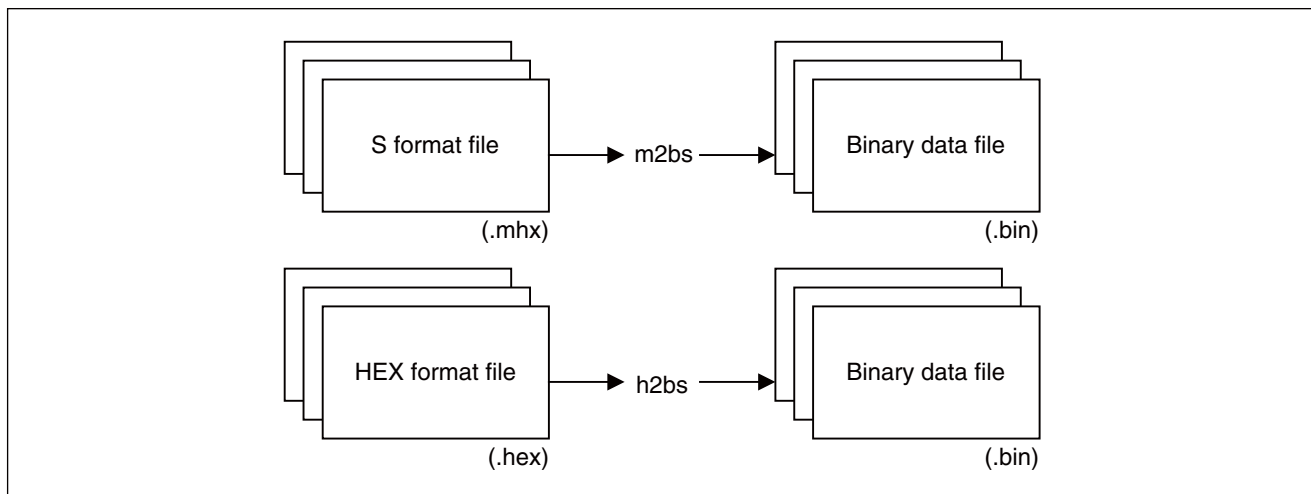
■ Outline of the Binary Converter

The binary converter converts object files of the S format or Hex format made by the linkage kit into binary data (memory images) and outputs them to files.

Use m2bs to convert the S format into binary data; use h2bs to convert the Hex format into binary data.

It is possible to specify a multiple of input files (S format or Hex format). Also, it is possible to separate converted binary data into a multiple of files of specified byte sizes (hereinafter referred to as the split mode).

Figure 17.1-1 Overview of the Binary Converter



Note:

The portion of an input file containing no data is padded with 0xFF (default value). The binary data file to be output is the contents of memory that is output into a file without changes.

Use the padding option (-p) to pad with a specified value the portion of an input file containing no data. For details on how to use this option, see Section "14.3 Padding (-p)".

The default extension of an output binary file is .bin. In the split mode, the extension .bxx (xx is a two digit number (01 through 16)) is unconditionally added.

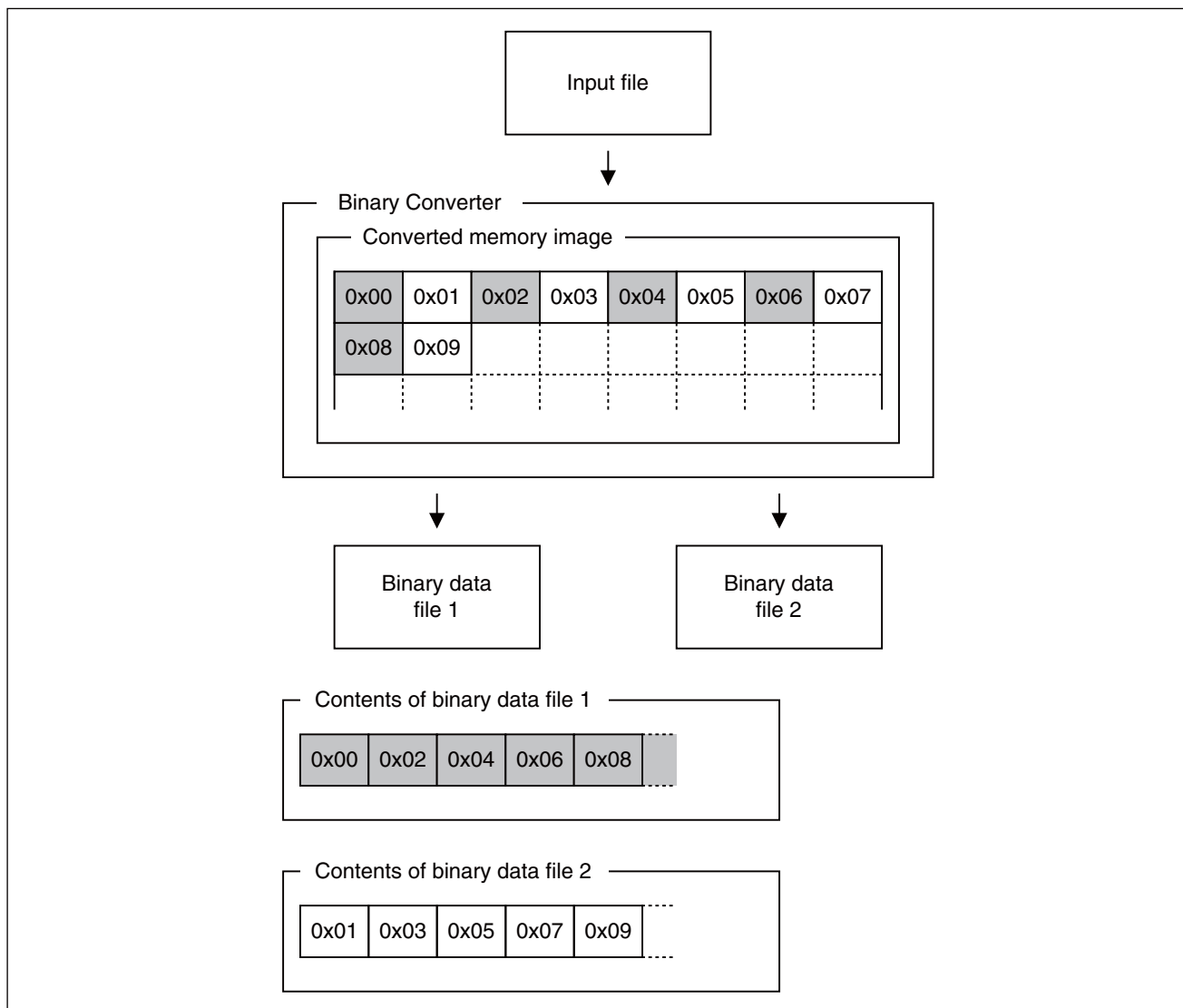
■ Overview of the Split Mode

A split mode means that the memory image converted by Binary Converter is split for specified bytes and output into multiple binary data file.

Figure 17.1-2 shows an overview of the split mode. In Figure 17.1-2, every byte of data is output in turns into two files. In the split mode, every specified byte of data may be output in turns into sixteen files maximum.

Use the -sp option to specify the split mode. For details on how to use the -sp option, see Section "17.3.2 Specifying the Split Mode (-sp)".

Figure 17.1-2 Overview of the Split Mode



17.2 List of Options of the Binary Converter

Section 17.2 lists the option names and functions of the Binary Converter.

■ List of Options of the Binary Converter

Table 17.2-1 lists options of the Binary Converter.

Table 17.2-1 List of Options of the Binary Converter

Function	Option	Remarks
Changing an output file name	-o	* Common option for a converter
Specifying padding data	-p	* Common option for a converter
Specifying the output range	-ran	need
Specifying the split mode	-sp	
Specifying the inhibition of the split mode	-Xsp	
Specifying to create a map list file	-m	
Specifying not to create a map list file	-Xm	
Specifying not to read a default option file	-Xdof	* Common option
Specifying to read an option file	-f	* Common option
Specifying to display help messages	-help	* Common option
Specifying to output the version number and messages	-V	* Common option
Specifying not to output the version number and messages	-XV	* Common option
Specifying to display a termination message	-cmsg	* Common option
Specifying not to display a termination message	-Xcmsg	* Common option
Specifying to set the termination code to 1 when a warning occurs	-cwno	* Common option
Specifying to set the termination code to 0 when a warning occurs	-Xcwno	* Common option

17.3 Details on Options of the Binary Converter

Section 17.3 describes the options of the Binary Converter.

For information on the common options for Linkage kit, see "CHAPTER 3 COMMON OPTIONS". For information on the common options for a converter, see "CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER".

■ Specifying the Output Range (-ran)

This option specifies the range of an S format file or HEX format file to be converted to a binary image. For details, see Section "17.3.1 Specifying the Output Range (-ran)".

■ Specifying the Split Mode (-sp)

This option specifies that a binary image is output in the split mode. For details, see Section "17.3.2 Specifying the Split Mode (-sp)".

■ Specifying the Inhibition of the Split Mode (-Xsp)

This option nullifies the split mode specification (-sp). For details, see Section "17.3.3 Specifying the Inhibition of the Split Mode (-Xsp)".

■ Specifying to Create a Map List File (-m)

This option specifies that conversion information is output in a map list file. For details, see Section "17.3.4 Specifying to Create a Map List File (-m)".

■ Specifying not to Create a Map List File (-Xm)

This option nullifies the specification to create a map list file (-m). For details, see Section "17.3.5 Specifying not to Create a Map List File (-Xm)".

17.3.1 Specifying the Output Range (-ran)

Use addresses to specify the range to be converted to a binary image.
This option must be specified for Binary Converter.

■ Specifying the Output Range (-ran)

[Syntax]

```
-ran < Starting address > [ , < Ending address > ]
```

[Parameters]

<Starting address>

Starting address

<Ending address>

Ending address

[Description]

Use addresses to specify the range to be converted to a binary image.

This option must be specified before converting.

Specify the starting and ending addresses between 0x0 and 0xFFFFFFFF inclusive.

The ending address may be omitted. If so, data as much as 64Kbytes from the starting address is converted to binary.

You cannot specify values that will make the conversion size more than 2Gbytes.

[Example 1]

```
m2bs sfmtfile.mhx (An example of not using the -ran option)
```

An error occurs because the output range is not specified.

[Example 2]

```
m2bs sfmtfile.mhx -ran 0xD000,0xFFFF
```

The data in sfmtfile.mhx at addresses 0xD000 through 0xFFFF is extracted and output into a binary image file.

[Example 3]

```
m2bs sfmtfile.mhx -ran 0xD000 (An example of omitting the ending address)
```

The data in sfmfile.mhx as much as 64Kbytes from the address 0xD000 (0x0D000 through 0x1CFFF) is extracted and output into a binary image file.

[Example 4]

```
m2bs sfmtfile.mhx -ran 0xFFFF,0xD000
```

An error occurs because the specified ending address is smaller than the starting address.

[Example 5]

```
m2bs sfmtfile.mhx -ran (Example of omitting all the parameters)
```

An error occurs because the starting address is omitted.

17.3.2 Specifying the Split Mode (-sp)

Specify this option to output a binary image in the split mode.

■ Specifying the Split Mode (-sp)

[Syntax]

```
-sp < Number of output files > [ , < Number of bytes > ]
```

[Parameters]

<Number of output files>

Specifies how many output destination files the data should be split into. A value between 2 and 16 inclusive may be specified.

<Number of bytes>

Specifies the unit of splitting data in bytes. A value between 0x01 and 0xFFFFFFFF inclusive may be specified.

[Description]

This option is used to output data in turns into multiple files. Use this option, for example, to output every two bytes of data in turns into two files when data in 32bits units is configured using two ROMs with a 16bits data width.

In <Number of bytes>, you cannot specify a value that will cause one or more output files to have zero-byte output.

The <Number of bytes> may be omitted. If so, the data is split in 1byte units by default.

This option allows you to output the 64Kbytes data into two 32Kbytes binary image files. However, this option is used only to output data in turns into multiple files. If you specify the parameters to split the 64Kbytes data into two 32Kbytes files, the last 1Kbytes data is output into the former file.

If this option is specified, the extension ".bxx" (xx is a two-digit number between 01 and 16 inclusive) is unconditionally added to the output file.

[Example 1]

```
m2bs sfmtfile.mhx (Example of not using the -sp option)
```

A binary image is output into sfmtfile.bin.

[Example 2]

```
m2bs sfmtfile.mhx -sp 2
```

Every one byte of a binary image is output in turns into sfmtfile.b01 and sfmtfile.b02.

[Example 3]

```
m2bs sfmtfile.mhx -sp 2,2
```

Every two bytes of a binary image is output in turns into sfmtfile.b01 and sfmtfile.b02.

17.3.3 Specifying the Inhibition of the Split Mode (-Xsp)

The -Xsp option nullifies the split mode specification (-sp).

■ Specifying the Inhibition of the Split Mode (-Xsp)

[Syntax]

-Xsp

[Parameter]

None

[Description]

Specify this option to nullify the -sp specification.

This option needs not be specified in particular because it is the default.

[Example]

```
m2bs  ccp903.mhx -ran 0xE000,0xFFFF
m2bs  cpp903.mhx -Xsp -ran 0xE000,0xFFFF
```

The default processing does not run in the split mode.

The above specifications are equivalent.

```
m2bs  -f option.file ccp903 -Xsp
```

You may sometimes want to temporarily change the specification in an option file when the option file is used to execute the program.

If the -sp option exists in option.file, there is no need to change the contents of option.file. Simply specify the -Xsp option on a command line to nullify the -sp option.

17.3.4 Specifying to Create a Map List File (-m)

Use this option to output the information at the time of conversion into a map list file.

■ Specifying to Create a Map List File (-m)

[Syntax]

```
-m <Map list file name >
```

[Parameter]

<Map list file name>
Output map list file name

[Description]

Use this option to output information at the time of conversion into a map list file.

The information at the time of conversion is output into a map list file. The following information is output. Item (4) is output only if the -sp option is specified.

- 1)Input file name information
- 2)Output file name information
- 3)Output range information
- 4)Split unit information
- 5)Padding data value information

If the <Map list file> has no extension, the default extension ".mp3" is added.

[Example 1]

```
m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF(Example of not specifying the -m option)
```

A map list file is not created because the -m option is not specified.

[Example 2]

```
m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF -m logfile
```

The information at the time of conversion is output into logfile.mp3.

Figure 17.3-1 Example 1 of Contents of logfile.mp3

Input file	:sfmtfile.mhx
Output file	:sfmtfile.bin
Convert range	:0x00010000 - 0x0001FFFF
Padding data	:0xFF

[Example 3]

```
m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF -m logfile -sp 2,2
```

The information at the time of conversion is output into logfile.mp3.

Figure 17.3-2 Example 2 of Contents of logfile.mp3

Input file	:sfmtfile.mhx
Output file	:sfmtfile.b01
	:sfmtfile.b02
Convert range	:0x00010000 - 0x0001FFFF
Split byte	:2
Padding data	:0xFF

17.3.5 Specifying not to Create a Map List File (-Xm)

Use the -Xm option not to create a map list file.

■ Specifying not to Create a Map List File (-Xm)

[Syntax]

-Xm

[Parameter]

None

[Description]

Specify this option to nullify the -m specification.

There is no need to specify this option because it is the default.

[Example 1]

```
m2bs  ccp903.mhx -ran 0xE000,0xFFFF
m2bs  ccp903.mhx -Xm -ran 0xE000,0xFFFF
```

A map list file is not output by default.

The above specifications are equivalent.

[Example 2]

```
m2bs  -f option.file ccp903 -Xm
```

You may sometimes want to temporarily change the specification in an option file when the option file is used to execute the program.

If the -m option exists in option.file, you need not change the contents of option.file but simply specify the -Xm option on a command line to nullify the -m option.



CHAPTER 18

OTHER CONVERTERS

This chapter explains the commands of other converters in detail.

- 18.1 m2is (Converting a S Format File into the HEX8 Format)
- 18.2 m2es (Converting a S Format File into the HEX16 Format)
- 18.3 i2ms (Converting a HEX8 Format File into the S Format)
- 18.4 e2ms (Converting a HEX16 Format File into the S Format)

18.1 m2is (Converting a S Format File into the HEX8 Format)

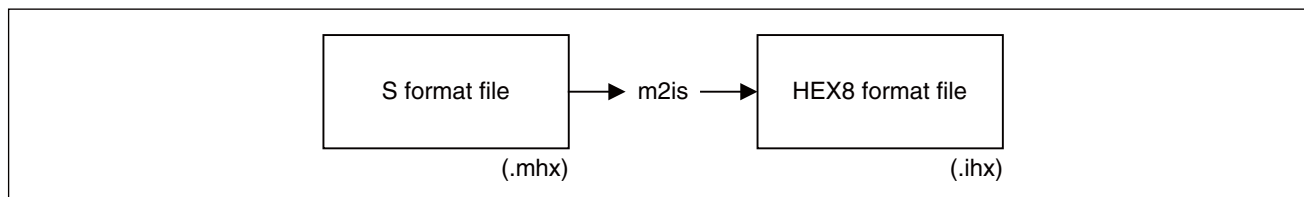
A S Format file is converted into the HEX8 format.
Data at the addresses 0 through 0xFFFF is to be converted.

■ m2is (Converting a S Format File into the HEX8 Format)

[Function]

The m2is command converts a S format file into the HEX8 format.

Figure 18.1-1 m2is (Converting S Format File into HEX8 Format)



Note:

The S format can represent the addresses 0 through 0xFFFFFFFF. However, when converted into the HEX8 format, the data allocated at the addresses 0x10000 and higher is truncated.
When using this command, be careful of the range of addresses in the conversion source.
Since a HEX8 format file consists of data records and a trailer record, the starting address information in the S format will be lost.

18.2 m2es (Converting a S Format File into the HEX16 Format)

A S Format file is converted into the HEX16 format.

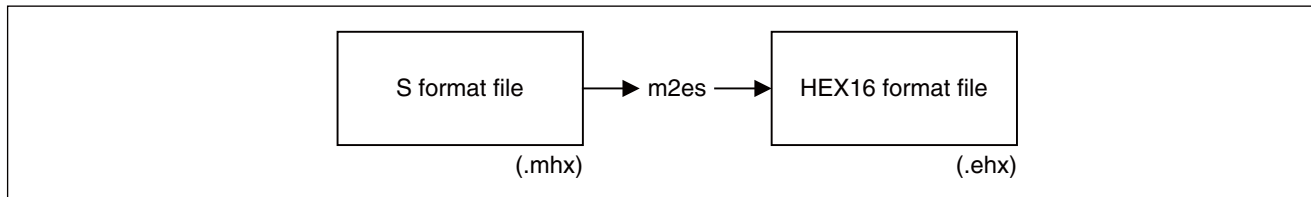
The data at the addresses 0 through 0xFFFFF is to be converted.

■ m2es (Converting a S Format File into the HEX16 Format)

[Function]

The m2es command converts a S format file into the HEX16 format.

Figure 18.2-1 m2is (Converting S Format File into HEX16 Format)



Note:

The S format can represent the addresses 0 through 0xFFFFFFFF. However, when converted into the HEX16 format, the data allocated at the addresses 0x100000 and higher is truncated. When using this command, be careful of the range of addresses in the conversion source.

18.3 i2ms (Converting a HEX8 Format File into the S Format)

A HEX8 format is converted into the S format.

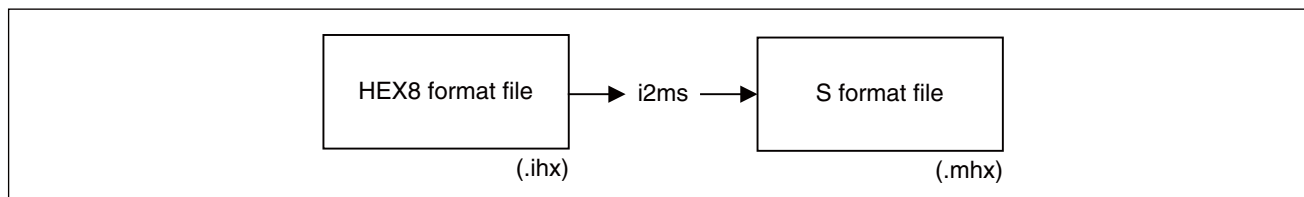
The data at the addresses 0 through 0xFFFF is to be converted.

■ i2ms (Converting a HEX8 Format File into the S Format)

[Function]

The i2ms command converts a HEX8 format file into the S format.

Figure 18.3-1 i2ms (Converting HEX8 Format File into S Format)



Note:

Although the S format can represent the addresses 0 through 0xFFFFFFFF, the HEX8 format cannot represent the addresses 0x10000 and higher. A S format file after conversion is created without the S2, S3, S7, and S8 types.

Since the HEX8 format does not have starting address information, the starting address after conversion is to be 0.

18.4 e2ms (Converting a HEX16 Format File into the S Format)

A HEX16 format is converted into the S format.

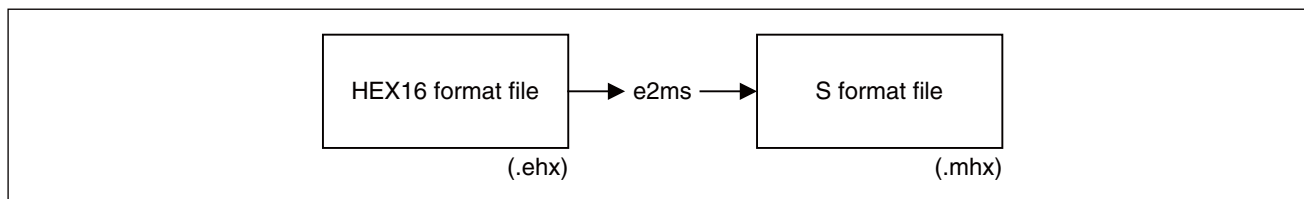
The data at the addresses 0 through 0xFFFFF is to be converted.

■ e2ms (Converting a HEX16 Format File into the S Format)

[Function]

The e2ms command converts a HEX16 format file into the S format.

Figure 18.4-1 e2ms (Converting HEX16 Format File into S Format)



Note:

Although the S format can represent the addresses 0 through 0xFFFFFFFF, the HEX16 format cannot represent the addresses 0x100000 and higher. A S format file after conversion is created without the S7 and S3 types.

The starting address information in the HEX16 format is set in the S9 or S8 type.

CHAPTER 19

RESTRICTIONS AND QUESTIONS AND ANSWERS ON AN OBJECT FORMAT CONVERTER

This chapter explains restrictions and questions and answers on using an object format converter.

19.1 Restrictions on an Object Format Converter

19.2 Questions and Answers on Using an Object Format Converter



19.1 Restrictions on an Object Format Converter

There are several restrictions in the binary converter and format adjuster. No other restrictions have been created for processing when using the other object format converters. It is possible to process using the entire memory that the object format converter can use in execution.

■ Restrictions on an Object Format Converter

Table 19.1-1 shows the restrictions when using the Object Format Converter.

However, this is not the maximum limit value available for processing. If it is necessary to register the section name or the symbol name with the internal processing of the object format converter, the object format dynamically retrieves memory. When the object format converter reaches the limit of memory for execution, a message is output to indicate that there is not enough memory and the object format converter process ends.

Table 19.1-1 Object Format Converter Restrictions List

Item	Restriction Value	Remarks
Option file count	Limitless	Memory dependent
Option file internal line count	Limitless	Memory dependent
Option file internal character count per 1 line	Limitless	Memory dependent
Option file nest	Not possible	
Input file count (m2bs,m2ms)	64	
Input file count (except m2bs,m2ms)	1	
I/O file size	Limitless	OS dependent
I/O file Line Count	Limitless	OS dependent
File name character count	Limitless	OS dependent
Maximum Memory Address	0xFFFFFFFF	
Maximum convert size (m2bs,m2ms)	2Gbytes to 1 byte	Memory dependent

■ Cautionary Information Concerning Binary Converter and Format Adjuster

You can specify up to a total of 64 items for the Input File.

If you have set a number of Input Files, they are processed in order their being set.

If there is data for the same address in the Input File, the subsequent data overwrites the antecedent data.

You can convert a maximum of 2G bytes -1 byte at one time.

19.2 Questions and Answers on Using an Object Format Converter

Section 19.1 covers the questions and answers on using an object format converter.

■ Questions and Answers on Using an Object Format Converter

Q.	There are many converters available. Which one should I use?
A.	A converter is used to convert an absolute format load module file that is output by a linker to an object format that can be read by a ROM writer. It is recommended to use f2ms converting to the S format and f2hs to the HEX format because the f2ms and f2hs fully support the 32-bit addressing space. Use other conversion tools as required.
Example.	f2ms absfile.abs -> Outputs absfile.mhx in the S format

Q.	When I use Binary Converter and format adjuster, an error, "F9001U: Insufficient memory" is output and I cannot convert data to a binary image. What should I do?
A.	Binary Converter and format adjuster secure as much memory as the area to be converted. If you try to convert a large area to the memory image at once, an error, "F9001U: Insufficient memory" may be output and the processing interrupted. In such a case, split the target area into multiple continuous areas and create a memory image for each of them. Then, merge the files into one binary image.
Example.	Converting the binary image area from 0xC00000 through 0xFFFFFFFF - If enough memory can be secured (Normal): 1 m2bs absfile.mhx -ran 0xC00000,0xFFFFFFFF - If an error, "F9001U: Insufficient memory" is output: 1 m2bs absfile.mhx -ran 0xC00000,0xDFFFFFF -o absfile1.bin 2 m2bs absfile.mhx -ran 0xE00000,0xFFFFFFFF -o absfile2.bin 3 copy /b absfile1.bin + absfile2.bin absfile.bin



APPENDIX

These appendixes describe the error messages of the linkage kit, HEX format, and S format.

APPENDIX A	ERROR MESSAGES OF THE LINKAGE KIT
APPENDIX B	HEX FORMAT
APPENDIX C	S RECORD FORMAT
APPENDIX D	LIST OF LINKER OPTIONS
APPENDIX E	LIST OF LIBRARIAN OPTIONS
APPENDIX F	LIST OF COMMANDS AND OPTIONS OF THE OBJECT FORMAT CONVERTER
APPENDIX G	SPECIFICATION DIFFERENCES DEPENDING ON THE OS
APPENDIX H	IDENTIFICATION METHOD FOR SAME OBJECT WHEN SOFTUNE LANGUAGE TOOL IS TRANSFERRED
APPENDIX I	DIFFERENCE IN SPECIFICATION OF SOFTUNE LINKER(FLNK896S) AND OLD LINKER(LINK96)
APPENDIX J	DIFFERENCE IN SPECIFICATION OF SOFTUNE LIBRARIAN (FLIB896S) AND OLD LIBRARIAN (LIB96)
APPENDIX K	Major Changes

APPENDIX A ERROR MESSAGES OF THE LINKAGE KIT

This section explains the classification of error messages output by each tool of the linkage kit and the display format.

■ Linkage Kit Error Message Classes

Error messages can be classified into the following four levels based on their importance.

- Information

Information notifies the user of contents to confirm the processing. Since this is no errors, and the user can obtain correct processing results.

- Warning

Warnings are slighter than errors, and output results can be used almost without causing any trouble. It is possible that other processing than the user system is executed. Check the contents of messages before determining whether output results can be used.

- Error

Processing continues to be executed, but troubles have occurred which make it impossible to obtain correct results. The causes of errors must be removed before rerun.

- Fatal error

Errors which make it impossible to continue processing. This type of errors results from wrong specifications of the user or problems of the execution environment.

■ Linkage Kit Error Message Display Format

Error messages are output by each tool in the following format.

```
*** File name (line number)  XnnnnT: Message text (supplementary message)
```

Part	Explanation
File name (line number)	The name of the source file in which an error occurred and the source line number. This information is only output a part of message for the linkers
X	The level of each error is indicated by the following one alphabetical character. I ... Information E... Error W ... Warning message F... Fatal error
nnnn	Error number The following shows the correspondence between the error number and error level. 0000 to 0999 ... I 1000 to 1999 ... W 4000 to 4999 ... E 9000 to 9999 ... F
T	The tool identification is indicated by the following one alphabetical character. L ... Linker U ... Librarian, object format converter
Message text	Error message text (Japanese/English can be selected)
Supplementary message	Detailed information about error. The causes of error are displayed with the symbol names. This may also be output in error message text.

[Examples]

```
*** sample.c(234) E4329L: Value out of range (0xFFFE37D4)
```

This is an example in which the source file name and line number are also displayed.

```
*** E4402U: Duplicated module name (date.obj setdate)
```

This is an example in which neither the source file name nor line number is displayed.

```
*** F9001U: Insufficient memory
```

This is an example in which neither the source file name nor line number nor supplementary message is displayed.

■ Error Messages of the Linker

I0301L	Unused library (file name)
--------	----------------------------

There are one or more libraries that were not used in linking process.

This is a message which is notified when 2 is specified in the -w option.

I0302L	Debug information not exist (file name)
--------	---

Debug information is not contained in the input file.

This is a message which is notified when 2 is specified in the -w option.

The error is slighter than W1351.

The message only report that debug information does not exist.

I0303L	Removed debug information
--------	---------------------------

Output file was created after removing debug information.

This is a message which is notified when 2 is specified in the -w option.

I0304L	File include WARNING level error (file name)
--------	--

The file indicated here was warned in a warning message when linking.

This is a message which is notified when 2 is specified in the -w option.

I0305L	Ignore address alignment
--------	--------------------------

Because the -pk option was specified, allocation was carried out ignoring boundary alignment when linking.

This is a message which is notified when 2 is specified in the -w option.

I0306L	Section allocated automatically in "area name" area
--------	---

Optimal section allocation was carried out in the area indicated here.

This is a message which is notified when 2 is specified in the -w option.

I0307L	Lower compatible cpu type object (file name)
--------	--

The file indicated here is a module for a different CPU type with downward compatibility.

This is a message which is notified when 2 is specified in the -w option.

W1301L	Writable section located in ROM area (section name)
--------	---

A writable section was allocated in the address range specified as a ROM area.

Examine the section allocation specification.

W1303L	Section with initial data located in RAM area (section name)
--------	--

A section with initial data was allocated in the address range specified as a RAM area.

Examine the section allocation specification.

W1305L	Section (section name) located on out of ROM/RAM area (area name)
--------	---

A section exists which is allocated outside the address range specified by using the -ro or -ra option. Check the section using the section map.

If you cannot get full information from the linker's map list, you can get further information from the section detail map list.

W1306L	Exceeded maximum address (section name)
--------	---

A section which was allocated exceeding the maximum address exists.

W1307L	Duplicate section name exist (section name)
--------	---

Sections with the same name and with different attributes or types exist in multiple modules.

W1308L	Overlap located section (section 1, section 2)
--------	--

Section allocation is overlapped. Since program operations may be affected depending on the situation, care must be taken.

It is recommended to avoid overlapping by using the options.

W1314L	Specified address to absolute section (section name)
--------	--

Since the absolute address has been determined for the section indicated here, the specified section relocation is illegal.

W1320L	Not search library at mode-identification mode
--------	--

Library retrieval processing is used only when creating absolute format load modules.

The library retrieval specification when the -r option is specified is illegal.

The following is displayed in the mode identification of the supplementary message.

- If the relative format load module output is specified(-r):REL

W1321L	Ignore (option) Option at mode-identification mode
--------	--

This message is notified when an illegal specification in the specified link mode is set.

For example, since no section allocation is carried out when relative format load module output is specified, it is illegal to specify the -sc option.

W1325L	Entry point already set
--------	-------------------------

Entry points are set in multiple input modules.

The entry point set first is valid.

W1326L	Entry point was changed
--------	-------------------------

The entry point already set has been changed by specifying the -e option.

W1327L	Duplicate symbol definition (symbol name)
--------	---

The same external definition symbol exists in multiple input modules.

The symbol defined first is valid.

W1328L	Mismatch symbol type (symbol name)
--------	------------------------------------

This message is notified if the external definition symbol specified when an entry point was set using the -e option is neither function name nor variable name nor address label name.

This message is notified if the level is lower than E4326.

W1330L	Too long ROM section name : exceeded 250 character (section name)
--------	---

Up to 250 characters are allowed for the section name to be used for ROM- RAM transfer.

W1332L	No match (file name) argument
--------	-------------------------------

A file was specified using the wild card, but no corresponding file was found. This specification is ignored.

W1351L	Debug information not exist
--------	-----------------------------

Part of the list cannot be created due to insufficient debug information.

Specify the debug information add option -g when compiling, assembling, or linking.

W1368L	The area specified for the -ro option is outside the internal-ROM area (area name)
--------	--

The area specified for the -ro option is outside the internal-ROM area.

Check whether the specified area is correct.

This warning is output even if warning output suppression (-w 0) is specified.

W1369L	The area specified for the -ra option is outside the internal-RAM area (area name)
--------	--

The area specified for the -ra option is outside the internal-RAM area.

Check whether the specified area is correct.

This warning is output even if warning output suppression (-w 0) is specified.

W1370L	The section is placed outside the ROM area (section name)
--------	---

The section that should be placed within the ROM area is placed outside the ROM area.

Check the map file to see if the placement is correct.

This warning is output even if warning output suppression (-w 0) is specified.

W1371L	The section is placed outside the RAM area (section name)
--------	---

The section that should be placed within the RAM area is placed outside the RAM area.
Check the map file to see if the placement is correct.
This warning is output even if warning output suppression (-w 0) is specified.

W1372L	The section is placed outside the RAM area or the I/O area (section name)
--------	---

The section that should be placed within the RAM area or the I/O area is placed outside the RAM area or the I/O area.
Check the map file to see if the placement is correct.
This warning is output even if warning output suppression (-w 0) is specified.

W1373L	The section is placed outside the I/O area (section name)
--------	---

The section that should be placed within the I/O area is placed outside the RAM area.
Check the map file to see if the placement is correct.
This warning is output even if warning output suppression (-w 0) is specified.

W1375L	This is the section (section name) for which layout is not specified in the -sc option.
--------	---

This is the section for which the user does not specify layout. Use the specified section layout (-sc) option to specify layout for the corresponding section.

E4302L	Not found section or section group name (section name or group name)
--------	--

The section name or group name specified by the -sc option cannot be found.

E4303L	Undefined ROM/RAM area name (area name)
--------	---

The ROM/RAM area name specified by the -sc option is not defined.

E4304L	Symbol name is not found (symbol name)
--------	--

This message is notified if the external reference symbol specified by the -df option or the external definition symbol specified by the -e option cannot be found.

E4305L	Unresolved external symbol (symbol name)
--------	--

Because the external definition symbol definition could not be found, relocation could not be carried out.
This message is also notified if the external definition symbol name specified by the -df option could not found.

A module which contains external definition symbols must be linked.

E4312L	Incompatible cpu type module (file name)
--------	--

The file displayed here is an incompatible module.

The target CPU for input modules must be the same or a compatible CPU.

E4319L	Section not exists (section name)
--------	-----------------------------------

The section specified by the grouping option(-gr) cannot be found.

This message is also notified if the section indicated here is an absolute section, since it is not intended for section relocation.

E4326L	Different symbol type (symbol name)
--------	-------------------------------------

This message is notified if the external definition symbol specified when an entry point was set using the -e option is neither function name nor variable name nor address label name.

E4327L	Illegal RL information
--------	------------------------

This message is notified if any error in relocation information is found.

E4329L	Value out of range (value)
--------	----------------------------

Overflow occurred in relocation operations.

This message contains the source program name including the description of data for relocation and the line numbers. Use this information to check the program.

If you can use the tag jump of your editor, you can jump to the corresponding source lines immediately.

E4330L	Divide by 0
--------	-------------

The divisor in the division for relocation operations is 0.

E4331L	Indispensable to locate address (section name)
--------	--

This message is notified if there is no addressing by the -sc option or a section without any specification is found.

E4332L	Not handling group name (group name)
--------	--------------------------------------

A group name cannot be specified when a ROM- RAM transfer section is specified using the -sc option.

E4333L	Not specified ROM address (section name)
--------	--

No ROM address is specified for relocation information to be allocated in the ROM area.
Specify a ROM address using the -sc option.

E4351L	Relocatable assemble list not correspond to object file (file name)
--------	---

No object corresponding to the relative assemble list file indicated by the supplementary message is found.
The linker stops processing of the list file indicated by the supplementary message, and proceeds with processing of the next list file.

E4352L	Relocatable assemble list file not found (file name)
--------	--

The relative assemble list file indicated by the supplementary message is not found.
The linker does not create an absolute assemble list file corresponding to the file name indicated by the supplementary message, and continues processing.

E4354L	Illegal relocatable assemble list file format (file name)
--------	---

The linker cannot process the format of the relative assemble list file indicated by the supplementary message.
The linker stops processing of the list file indicated by the supplementary message, and proceeds with processing of the next list file.

E4355L	Object data not correspond (file name)
--------	--

Object data of the absolute format load module file and source data of the relative format assemble list indicated by the supplementary message do not match.

Rerun the program after reassembling.

E4357L	Too many array dimension or structure nested : exceeded 8
--------	---

The number of dimensions of an array output to the ARRAY list or that of nesting levels of a structure exceeds 8.

Processing continues, but those parts which exceed this limit are not output.

E4362L	DUMMY section specified (file name)
--------	-------------------------------------

A dummy section is specified in a file indicated by the supplementary message.

Do not write any dummy section when creating an absolute assemble list.

E4363L	Exceeded maximum section size (section name)
--------	--

One or more sections which exceed the maximum size of a section exist. Some sections have the maximum size depending on their section identification. This message is output if such limits are exceeded.

Examine such sections.

E4364L	Exceeded maximum bank address (section name)
--------	--

This message is notified if the maximum address of the bank is exceeded when the section is allocated from the specified location.

Examine the section configuration in the bank.

E4365L	Not found locatable address in area name (section name)
--------	---

No place in the specified allocation area can be found where the applicable section can be allocated.

Examine the section configuration in the area.

E4366L	Not found locatable address (section name)
--------	--

No place in all address space can be found where the section can be allocated.

Examine the program structure.

E4367L	Duplicated module name (file name module name)
--------	--

The same module name already exists as that intended for linking.

Duplicated module names are not allowed. Change the module names.

E4368L	Referenced other bank symbol (symbol name)
--------	--

A symbol defined in another bank was referenced in an intra-bank branch instruction.

Use a branch instruction to outside banks if the branch to the corresponding location (symbol) is required.

E4370L	CPU information file not found (file name)
--------	--

Can't find the target CPU information file specified by the -cpu option.

This is detected when the file below is not found.

- flnk 896s: %FETOOL%\LIB\896\896.CSV

E4371L	CPU information not found (file name)
--------	---------------------------------------

Can't find the target CPU information, in the CPU information file, specified by the -cpu option.

This is detected when the target CPU information specified in the file below is not found.

- flnk 896s: %FETOOL%\LIB\896\896.CSV

F9001L	Insufficient memory
--------	---------------------

Enough memory is not available to execute the program. If the linker is activated from a batch file, start the program directly from the command line.

F9011L	Input file not found (file name)
--------	----------------------------------

The specified input file cannot be found.

F9012L	Library file not found (file name)
--------	------------------------------------

If the name of the file indicated here is a default library file, the library file is not stored in the directory indicated by the environmental variable LIB896 or derived from the environmental variable FETOOL.

If the file is a library file specified by the -l or -el option, check whether the file name is specified correctly or the retrieval path specified in the -L option is correct.

F9015L	File open error (file name)
--------	-----------------------------

If the file indicated here is an output file, it is possible that the number of files that can be managed by one directory is exceeded. Remove or move unnecessary files.

F9016L	File read error (file name)
--------	-----------------------------

It is probable that the file is read-protected.

F9017L	File write error (file name)
--------	------------------------------

This message is also notified if a write-protected file with the same name exists.

The disk does not have enough free space, so the file indicated here can be written. Prepare sufficient free space in the disk before rerunning the linker.

F9021L	Too many options
--------	------------------

Too many input file names and options (including those in the option file) are specified on the command line.

Divide the files and options, and then activate the linker more than once

F9022L	Illegal option name (option)
--------	------------------------------

The option cannot be used by the linker. See the -help option or this manual.

F9023L	Illegal option parameter (option)
--------	-----------------------------------

An error is found in parameters to be specified for this option.

It is probable that the parameters fall short or a syntactical error such as a delimiter error has occurred.

F9024L	Illegal character (option)
--------	----------------------------

An error is found in parameters to be specified for this option.

This message is notified if an error due to the use of illegal characters such as that in the method of specifying numeric values is assumed.

F9026L	Specified value out of range (value)
--------	--------------------------------------

A value is specified which is outside the range of allowed values by the -pl, -pw, or -w option.

F9027L	Option file nested
--------	--------------------

Nesting of option files is not allowed. Delete the -f option described in the option file.

F9030L	Missing input file name
--------	-------------------------

Specify an input file.

F9032L	Output file name same as input one (file name)
--------	--

Since the output file name indicated here is the same as the input file name, processing cannot continue.

F9033L	Illegal file format (file name)
--------	---------------------------------

This message is notified in one of the following cases.

- The library file format is not correct.
- The object module format in a library file is not correct.
- The input file is an absolute format load module.
- Contents of an input module are not correct.
- The format of a relative format assemble list file is not correct.
- CPU Information file format is not correct.

F9040L	Duplicated file or path name (file name)
--------	--

This message is notified in one of the following three cases.

- Input module files of the same name are specified.
- Library files of the same name are specified in the -l, or -el option.
- The same library path name is specified in the -L option.

F9042L	Duplicated section name (section name)
--------	--

The same section name was specified in the -sc or -gr option more than once.

F9043L	Duplicated symbol name (symbol name)
--------	--------------------------------------

The same external reference symbol was specified in the -el option more than once.

F9044L	Duplicated section group name (group name)
--------	--

The same group name was used more than once when setting groups using the -gr option.

F9047L	No match file name argument
--------	-----------------------------

A file was specified using the wild card, but no corresponding file was found.

F9052L	Missing '-cpu' option
--------	-----------------------

No target CPU is specified using the -cpu option.

F9053L	Missing '-ro' or '-ra' option
--------	-------------------------------

The -ro option or -ra option required for automatic allocation is not specified.

F9056L	Mismatch CPU information file version
--------	---------------------------------------

The CPU information file is not suited old.
Please obtain the CPU information file of the latest.

F9998L	File open failed (file name)
--------	------------------------------

The message files used by the linker could not be opened.

Store the error message files(lkt896_a.msg,lkt896_e.msg)in a predetermined directory.

F9999L	Internal error (identification information)
--------	---

If this error occurs, report it to Spansion immediately.

■ Error Messages for the Librarian

I0401U	Reference to undefined symbol
--------	-------------------------------

This is a message reported if the -c option is specified. This message indicates that, after checking external symbols in the library file, external reference symbols that cannot be resolved in the library file are contained. When a library file for which this message is output is used in the linker, care must be taken to know to which module the external definition symbols to be used belong.

I0402U	Debug information exists
--------	--------------------------

This is a message reported if the -c option is specified. This library file contains a module which contains debug information. Debug information in the library file can be removed using the -O option.

I0407U	Lower compatible cpu type object (file name)
--------	--

The file indicated here is an object file or a library file which contains a module of a different CPU type downwardly compatible with the library file.

W1401U	Ignore '-pl' option
--------	---------------------

Though the number of lines of the list is specified(-pl options), the output of the target map list is not specified. This specification is ignored.

W1402U	Ignore '-pw' option
--------	---------------------

Though the number of digits of the list is specified(-pw options), the output of the target map list is not specified. This specification is ignored.

W1403U	Ignore '-g' option
--------	--------------------

The specification of creation of a library with debug information (-g option) has any meaning only if addition (-a option) or replacement (-r option) of a module is specified. This specification is ignored.

W1404U	Nothing to operate
--------	--------------------

There was no library file change nor module extract operation.

W1405U	Module not exists to delete (module name)
--------	---

The module specified by the -d option is not included in the library file.

Check the registered module names using the -m option.

W1406U	Module not exists to extract (module name)
--------	--

The module specified by the -x option is not included in the library file.

Check the registered module names using the -m option.

E4402U	Duplicated module name (file name module name)
--------	--

An attempt was made to register a module with the same name as that which has been already registered.

Duplicated names are not allowed in one library and so the module indicated here is not registered. Use the -r option for replacement.

E4403U	Duplicated external definition symbol name (file name symbol name)
--------	--

The registered modules contain external definition symbols and the symbol indicated here has been already registered in the library.

Duplicated external definition symbols are not allowed in one library, the module which contains the symbol indicated here is not registered.

E4404U	Invalid module : type (file name)
--------	-----------------------------------

Only the object module format output by the assembler can be registered in the library file. The absolute format and relative format load modules output by the linker cannot be registered.

E4405U	Invalid module : conflict tool name (file name)
--------	---

This is not an object file output by the family assembler that can be handled by this librarian. Or a library file created for another family librarian is specified.

E4406U	Invalid module : conflict compile model (file name)
--------	---

A module of a different compile model (such as the memory model) cannot be registered in the same library.

E4407U	Invalid module : conflict CPU type (file name)
--------	--

A module of a different target CPU cannot be registered in the same library.

E4408U	Too many entry modules : exceed 65535
--------	---------------------------------------

Create another library so that one library contains at most 65535 modules.

E4409U	Too many entry external symbols : exceed 65535
--------	--

Create another library so that one library contains at most 65535 symbols.

E4470U	CPU information file not found (file name)
--------	--

Can't find the target CPU information file specified by the -cpu option.
This is detected when the file below is not found.

- %FETOOL%\LIB\896\896.CSV

E4471U	CPU information not found (file name)
--------	---------------------------------------

Can't find the target CPU information, in the CPU information file, specified by the -cpu option.
This is detected when the target CPU information specified in the file below is not found.

- %FETOOL%\LIB\896\896.CSV

E9001U	Insufficient memory
--------	---------------------

Enough memory is not available to execute the program.

F9015U	File open error (file name)
--------	-----------------------------

If the file indicated here is an output file, it is possible that the number of files that can be managed by one directory is exceeded. Remove or move unnecessary files.

F9016U	File read error (file name)
--------	-----------------------------

It is possible that the file is read-protected.

F9017U	File write error (file name)
--------	------------------------------

It is possible that a write-protected file with the same name exists.

Or it is also possible that not enough free space of the disk is available, so the file can be written. Prepare enough free space in the disk before rerunning the linker.

F9021U	Too many options
--------	------------------

Too many input file names and options (including those in the option file) are specified on the command line.

Divide the files and options, and then activate the librarian more than once.

F9022U	Illegal option name (option)
--------	------------------------------

An error is found in the option name specification. Correct the command line and then reactivate.

F9023U	Illegal option parameter (option)
--------	-----------------------------------

An error is found in the parameters to be specified in this option.

F9026U	Specified value out of range (value)
--------	--------------------------------------

The value cannot be specified in the parameters of the -pl or -pw option.

See the -help option or this manual.

F9027U	Option file nested
--------	--------------------

Nesting of option files is not allowed. Delete the -f option described in the option file.

F9033U	Illegal file format (file name)
--------	---------------------------------

The CPU information file format is not correct or the library file format is not correct (no library file) or the input file specified for registration or replacement is not in the object format output from the assembler.

A file of a different format is entered or the file is damaged.

F9035U	Missing library file name
--------	---------------------------

The library file name is not correctly specified.

Specify the library file name correctly.

F9036U	Multiple library file name specified (file name)
--------	--

Only one library file can be specified. Select either the file name indicated here or a library file name specified before.

F9045U	'-O' option conflict with another option
--------	--

If the -O option is specified, other options cannot be specified.

F9046U	'-c' option conflict with another option
--------	--

Do not combine the -c option with any other options.

F9047U	No match file name argument
--------	-----------------------------

A file was specified using the wild card, but no corresponding file was found.

F9052U	Missing '-cpu' option
--------	-----------------------

No target CPU is specified using the -cpu option.

F9056U	Mismatch CPU information file version
--------	---------------------------------------

The CPU information file is not suited old.
Please obtain the CPU information file of the latest.

F9998U	File open failed (file name)
--------	------------------------------

The message files used by the library file could not be opened.
Store the error message files(lkt896_a.msg,lkt896_e.msg)in a predetermined directory.

F9999U	Internal error (identification information)
--------	---

If this error occurs, report it to our sales representatives immediately.

■ Error Messages of the Converter

I0501U	Skip start address record
--------	---------------------------

The start address record was contained in the HEX format, but it was skipped since it was not required. Converter processing is not affected and conversion is carried out correctly.

W1501U	File include WARNING level error (file name)
--------	--

The file specified for input contained an error of the warning level when linking. Check the file before using it.

W1502U	Unable to convert address (address)
--------	-------------------------------------

The file to be converted contains address data that cannot be represented in the converted format. All data after the address indicated here is discarded.

Change the converted format.

W1503U	-entry option was specified at the time of -I16 specification
--------	---

A start address output specification option was specified at the time of -I16 was specified.

W1504U	Start address information is not in an input file
--------	---

There is no start address information in an input load module file. The f2hs outputs HEX format without outputting the start address record.

F9001U	Insufficient memory
--------	---------------------

Enough memory is not available to execute the program.

F9011U	Input file not found (file name)
--------	----------------------------------

The file specified in the input file cannot be found.

F9015U	File open error (file name)
--------	-----------------------------

If the file specified here is an output file, it is possible that the number of files that can be managed by one directory is exceeded. Remove or move unnecessary files.

F9016U	File read error (file name)
--------	-----------------------------

It is probable that the file is read-protected or hardware fault has occurred.

F9017U	File write error (file name)
--------	------------------------------

The disk does not have enough free space, so the file indicated here can be written. Prepare sufficient free space in the disk before rerunning the converter.

F9021U	Too many options
--------	------------------

Too many input file names and options (including those in the option file) are specified on the command line.

F9022U	Illegal option name (option)
--------	------------------------------

The option cannot be used for the converter. See the -help option or this manual.

F9023U	Illegal option parameter (option)
--------	-----------------------------------

An error is found in parameters to be specified for this option.

F9026U	Specified value out of range (value)
--------	--------------------------------------

A value outside the range of values allowed by the option is specified.

F9027U	Option file nested
--------	--------------------

Nesting of option files is not allowed. Delete the -f option described in the option file.

F9028U	Specified address too large (option: s=address1 e=address2)
--------	---

The address specified in parameters of the option cannot be represented in the converted file format. Specify another address.

F9029U	Start address opposite to end one (option: s=address1 e=address2)
--------	---

The end address in parameters of the option is smaller than the start address.

F9030U	Missing input file name
--------	-------------------------

Specify the name of an input file to be converted.

F9031U	Multiple input file name (file name)
--------	--------------------------------------

Only one input file can be specified. Specify either the file name indicated here or a file name specified before as the input file.

F9032U	Output file name same as input one (file name)
--------	--

Since the output file name indicated here is the same as the input file name, processing cannot continue.

F9033U	Illegal file format (file name)
--------	---------------------------------

The input file is not in the object format to be processed.

A file of a different format is entered or the file is damaged.

F9034U	Not absolute load module file (file name)
--------	---

A file is input which is not an absolute format load module output from the linker. Change the format of the file to the absolute format and then use it in the converter.

F9048U	Missing output range
--------	----------------------

The output range is not specified.

Specify the range of output(-ran).

F9049U	Output range exceeded
--------	-----------------------

The output range exceeds the limit value.

F9050U	Output file name same as other output one (file name)
--------	---

The output file has the same name as that of another file.

Change the output file name.

F9051U	File name too long (file name)
--------	--------------------------------

The output file name indicated here is too long to be processed.

Make the specified file name shorter.

F9998U	File open failed (file name)
--------	------------------------------

The message files used by the converter could not be opened.

Store the error message file (lkt_a.msg and lkt_e.msg) in the predetermined directory.

F9999U	Internal error (identification information)
--------	---

If this error occurs, report it to SpanSion immediately.

APPENDIX B HEX FORMAT

This appendix explains the following formats of the HEX formats.

- **HEX8 format: Format set for 8 bits**
 - **HEX16 format: Format set for 16 bits**
 - **HEX32 format: Format set for 32 bits**
-

■ HEX Format

- Common Format (See Appendix "B.1")
- Data Record (HEX8/HEX16/HEX32) Type: 00 (See Appendix "B.2")
- End Record (HEX8/HEX16/HEX32) Type: 01 (See Appendix "B.3")
- Extended Segment Address Record (HEX16/HEX32) Type: 02 (See Appendix "B.4")
- Start Segment Address Record (HEX16/HEX32) Type: 03 (See Appendix "B.5")
- Extended Linear Address Record (HEX32) Type: 04 (See Appendix "B.6")
- Start Linear Address Record (HEX32) Type: 05 (See Appendix "B.7")

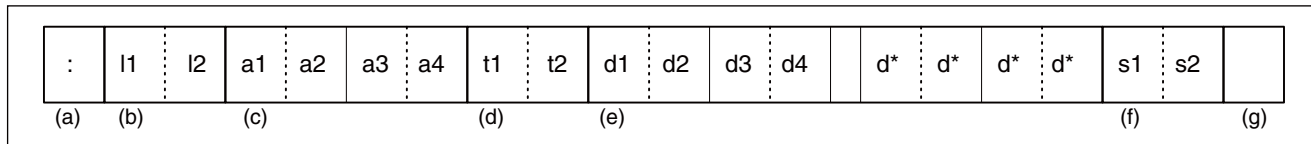
B.1 Common Format

The HEX format consists of six fields, (a) to (f).

Each field is set using the ASCII code. Field (g) is explained later.

■ Common Format

Figure B.1-1 Common Format



(a):

Indicates the start of a record. The character ":" (0x3A) is used.

(b):

Indicates the number of bytes in the data part of (e).

Since the actual 1-byte data is represented by 2-byte ASCII code in this format, d1 and d2 in the above figure are counted as one.

l1 indicates the high-order digits and l2 indicates the low-order digits. Values in the range of 0 to 255 can be set.

It is "00" to "FF" in ASCII notation and "0x3030" to "0x4646" in hexadecimal notation.

(c):

Indicates the address allocated to the first data if the contents of (e) are object data.

a1 indicates the high-order digits and a4 indicates the low-order digits. Values in the range of 0 to 65535 can be set.

It is "0000" to "FFFF" in ASCII notation and "0x30303030" to "0x46464646" in hexadecimal notation.

(d):

Indicates the record type.

00 : Data record (Intel HEX8/HEX16/HEX32 format)

01 : End record (Intel HEX8/HEX16/HEX32 format)

02 : Extended segment address record (Intel HEX16/HEX32 format)

03 : Start segment address record (Intel HEX16/HEX32 format)

04 : Extended linear address record (Intel HEX32 format)

05 : Start linear address record (Intel HEX32 format)

(e):

This field depends on the record type of (d). See the explanations of each record in B.2 to B.7.

(f):

Checksum. Each 2-byte data of (b), (c), (d), and (e) represented in ASCII is converted into 1-byte hexadecimal data. Each byte without a sign is added regardless of overflow.

The two's complement of the result is calculated, then set as 2-byte ASCII data.

s1 indicates the high-order digits.

The two's complement: Value obtained by adding 1 to the value obtained by reversing each bit.

(g):

Generally, a control code (such as CR and LF) is added.

Data in this field is skipped until the start character ":" of (a) appears.

Since the (a), (b), (c), (d), and (f) fields always exist, the minimum length of a record is 11 bytes long and the maximum length is 521 bytes long.

[Example]

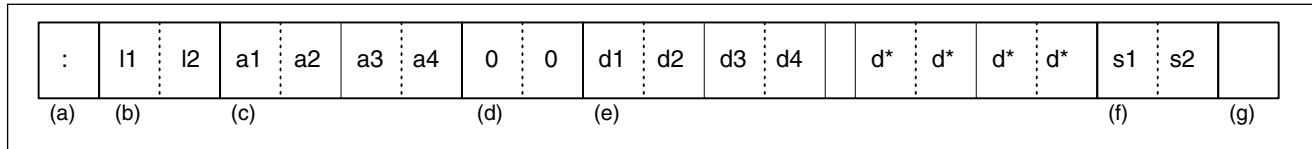
:020000020036C6	Extended address record
:0600100090D9226BB4FD43	Data record
:040000035162000541	Start address record
:00000001FF	End record

B.2 Data Record (HEX8/HEX16/HEX32) Type: 00

d1 and d2 are byte data at the address indicated by (C) and d3 and d4 are byte data of the next address.

■ Data Record (HEX8/HEX16/HEX32)

Figure B.2-1 Data Record (HEX8/HEX16/HEX32)



For (a), (b), (c), (d), (f), and (g), see the explanation in section "B.1 Common Format".

(e) is object data and the actual 1-byte data is represented by 2-byte ASCII code.

In the above figure, byte data at the address indicated by (c) is d1 and d2.

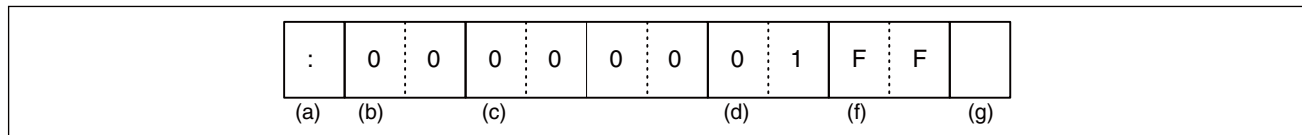
Likewise, byte data of the next address is d3 and d4.

B.3 End Record (HEX8/HEX16/HEX32) Type: 01

The end record is always 00000001FF.
Only one end record exists as the last record.

■ End Record (HEX8/HEX16/HEX32)

Figure B.3-1 End Record (HEX8/HEX16/HEX32)



The (e) field does not exist. Therefore 0 is set in (b).

(c) is not used and normally 0 is set.

B.4 Extended Segment Address Record (HEX16/HEX32) Type: 02

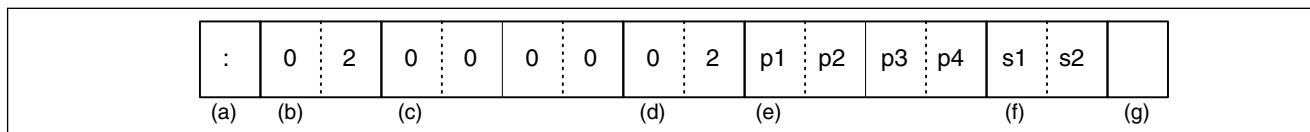
If extended segment address record appears, the address of each byte data of the following data records is calculated according to the following formula until next record appears.

$$((PA \times 0x10) + ((DA + DP) \bmod 0x10000)) \bmod 0x100000$$

- PA: (e) field value of the record
- DA: (c) field value of the data record. Here, this value is a relative address.
- DP: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

Extended Segment Address Record (HEX16/HEX32)

Figure B.4-1 Extended Segment Address Record (HEX16/HEX32)



(e):

This is a paragraph address and the actual 2-byte data is represented as 4-byte ASCII code.

In the above figure, p1 indicates the high-order digits.

(c):

This field is not used and normally 0 is set.

If extended segment address record appears, the address of each byte data of the following data records is calculated according to the following formula until next record appears.

$$[(PA \times 0x10) + [(DA + DP) \bmod 0x10000]] \bmod 0x100000$$

- PA: (e) field value of the record
- DA: (c) field value of the data record. Here, this value is a relative address.
- DP: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

This is the same method used for calculating the physical address in i8086. With the addition of extended segment address record, up to 20-bit addresses can be represented.

For the data records which appear before the extended segment address record, the address is calculated with the above PA setting as 0x0000.

B.5 Start Segment Address Record (HEX16/HEX32) Type: 03

This is a record to specify the start address of a program being executed.

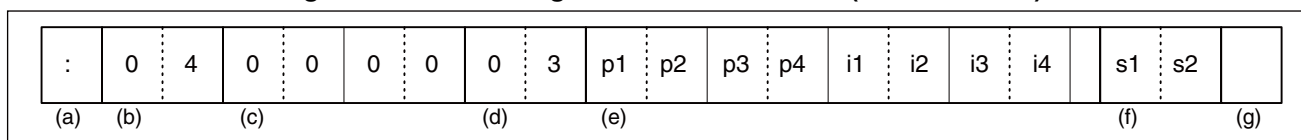
The start address is calculated using the following formula.

$$((PA \times 0x10) + IP) \text{ MOD } 0x100000$$

- PA: (e1) field value of the record
- IP: (e2) field value of the record

■ Start Segment Address Record (HEX16/HEX32)

Figure B.5-1 Start Segment Address Record (HEX16/HEX32)



As shown in the above figure, (e) is divided into two fields. The paragraph address is set in (e1), and the offset value is set in (e2).

Both p1 and i1 are high-order digits.

(c) is not used and normally 0 is set.

The start address is calculated using the following formula.

$$[(PA \times 0x10) + IP] \text{ MOD } 0x100000$$

- PA: (e1) field value of the record
- IP: (e2) field value of the record

This record can appear anywhere before the end record.

The appearance count is 0 or 1.

B.6 Extended Linear Address Record (HEX32) Type: 04

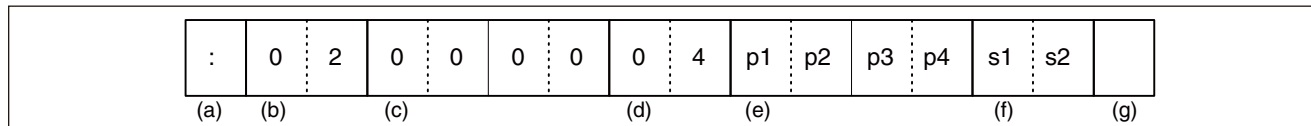
If extended linear address record appears, the address of each byte data of the following data records is calculated according to the following formula until next record appears.

$$((PA \times 0x10000) + ((DA + DP) \bmod 0x10000)) \bmod 0x100000000$$

- PA: (e) field value of the record
- DA: (c) field value of the record. Here, this value is a relative address.
- DP: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

Extended Linear Address Record (HEX32)

Figure B.6-1 Extended Linear Address Record (HEX32)



(e):

This is a paragraph address and the actual 2-byte data is represented as 4-byte ASCII code.

In the above figure, p1 indicates the high-order digits.

(c):

This field is not used and normally 0 is set.

If extended linear address record appears, the address of each byte data of the following data records is calculated according to the following formula until next record appears.

$$[(PA \times 0x10000) + [(DA + DP) \bmod 0x10000]] \bmod 0x100000000$$

- PA: (e) field value of the record
- DA: (c) field value of the record. Here, this value is a relative address.
- DP: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

This is the same method used for calculating the physical address in i80386. With the addition of extended linear address record, up to 32-bit addresses can be represented.

For the data records which appear before the extended linear address record, the address is calculated with the above PA setting as 0x0000.

B.7 Start Linear Address Record (HEX32) Type: 05

This is a record to specify the start address of a program being executed.

■ Start Linear Address Record (HEX32)

Figure B.7-1 Start Linear Address Record (HEX32)

:	0	4	0	0	0	0	0	5	e1	e2	e3	e4	e5	e6	e7	e8	s1	s2	
(a)	(b)		(c)				(d)		(e)								(f)		(g)

(e) is where the 32-bit execution start address is set.

The e1 is high-order digits.

(c) is not used and normally 0 is set.

This record can appear anywhere before the end record.

The appearance count is 0 or 1.

APPENDIX C S RECORD FORMAT

The S record format always starts with the character "S" (0x53). Eight types from S0 to S9 are available (S4 and S6 are not used).

■ S Record Format

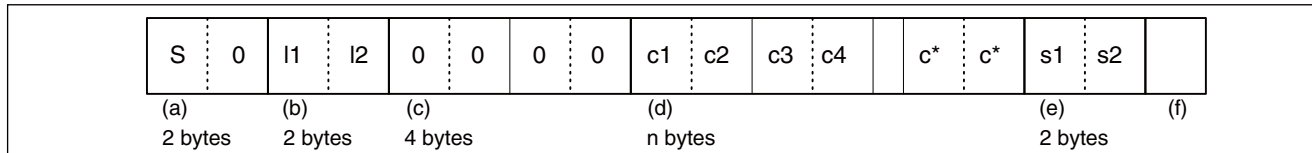
- S0 Type (Header Record) (See "Appendix C.1 S0 Type (Header Record)")
- S1 Type (Data Record: 2-Byte Address) (See "Appendix C.2 S1 Type (Data Record: 2-Byte Address)")
- S2 Type (Data Record: 3-Byte Address) (See "Appendix C.3 S2 Type (Data Record: 3-Byte Address)")
- S3 Type (Data Record: 4-Byte Address) (See "Appendix C.4 S3 Type (Data Record: 4-Byte Address)")
- S5 Type (Record to Manage the Number of Records) (See "Appendix C.5 S5 Type (Record to Manage the Number of Records)")
- S7 Type (Terminator Record) (See "Appendix C.6 S7 Type (Terminator Record)")
- S8 Type (Terminator Record) (See "Appendix C.7 S8 Type (Terminator Record)")
- S9 Type (Terminator Record) (See "Appendix C.8 S9 Type (Terminator Record)")

C.1 S0 Type (Header Record)

This record is used for comments.

■ S0 Type (Header Record)

Figure C.1-1 S0 Type (Header Record)



This record consists of the above five fields (a) to (e).

The S0 type is called the header record and is placed at the start of a file ahead of each record of S1 to S9. Each field is set in ASCII code.

(a):

Type field. "S0" (0x5330) in ASCII code.

(b):

Indicates the number of bytes in (c), (d), and (e).

Actual 1-byte data is represented as 2-byte ASCII code in this format and the number of characters in these fields divided by 2 is set.

I1 indicates the high-order digits and I2 indicates the low-order digits. Values in the range of 0 to 255 can be set.

It is "00" to "FF" in ASCII notation and "0x3030" to "0x4646" in hexadecimal notation.

(c):

This field is not used and normally "0000" in ASCII notation is set.

(d):

Messages such as version management information are set.

For the setting method, see the example below.

(e):

Checksum.

Each 2-byte data of (b), (c), and (d) represented in ASCII notation is converted into 1-byte data in hexadecimal notation. Each byte without a sign is added regardless of overflow.

One's complement of the result is calculated and then set as 2-byte ASCII code.

s1 indicates the high-order digits.

- One's complement: value obtained by reversing each bit

(f):

Generally, control code (such as CR and LF) is added.

Data in this field is skipped until the start character "S" of (a) appears.

[Example]

S00600004844521B



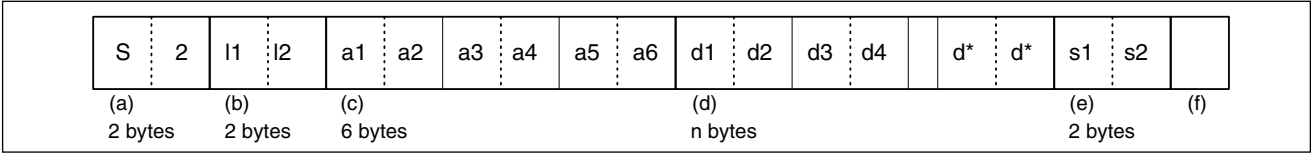
Indicates ASCII character string "HDR"

C.2 S1 Type (Data Record: 2-Byte Address)

This is a record to store object data that can be represented in two bytes (0x0000 to 0xFFFF).

■ S1 Type (Data Record: 2-byte Address)

Figure C.2-1 S1 Type (Data Record: 2-byte Address)



The S1 type consists of the above five fields (a) to (e).

(a):

Type field. "S1"(0x5331) in ASCII code.

(b):

Indicates the number of bytes in (c), (d), and (e).

(See the description in "Appendix C.1 S0 Type (Header Record)".)

(c):

Indicates the address allocated to the first data of (d).

a1 indicates the high-order digits and a4 indicates the low-order digits. Values in the range of 0 to 65535 can be set.

It is "0000" to "FFFF" in ASCII notation and "0x30303030" to "0x46464646" in hexadecimal notation.

(d):

Object data. Actual 1-byte data is represented by 2-byte ASCII code. In the above figure, d1 and d2 are byte data at the address indicated by (c).

Likewise, d3 and d4 are byte data of the next address.

(e):

Checksum.

(See the description in "Appendix C.1 S0 Type (Header Record)".)

(f):

Generally, control code (such as CR and LF) is added.

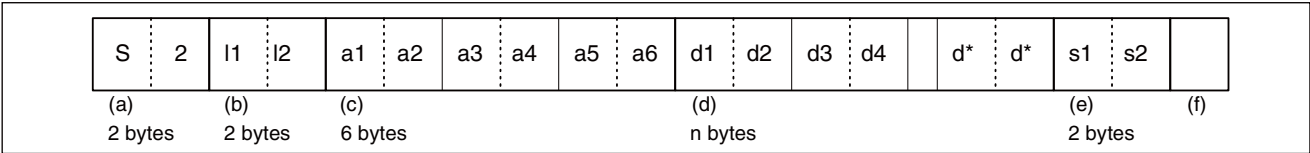
(See the description in "Appendix C.1 S0 Type (Header Record)".)

C.3 S2 Type (Data Record: 3-Byte Address)

The S2 type differs from the S1 type in (c) field size, and is a record to store object data which requires the 3-byte address.

■ S2 Type (Data Record: 3-byte Address)

Figure C.3-1 S2 Type (Data Record: 3-byte Address)



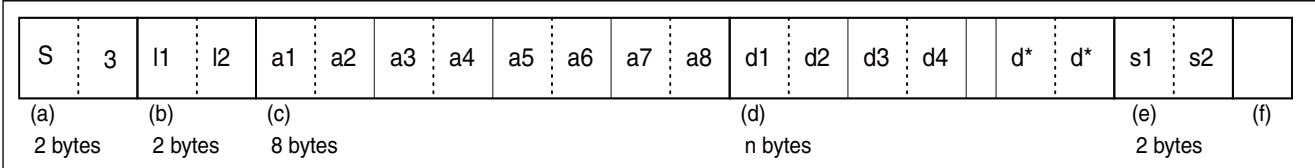
The S2 type differs from the S1 type in the above (c) field size, and is a record to store object data which requires the 3-byte address.

C.4 S3 Type (Data Record: 4-Byte Address)

The S3 type differs from the S1 type in (c) field size, and is a record to store object data which requires the 4-byte address.

■ S3 Type (Data Record: 4-byte Address)

Figure C.4-1 S3 Type (Data Record: 4-byte Address)



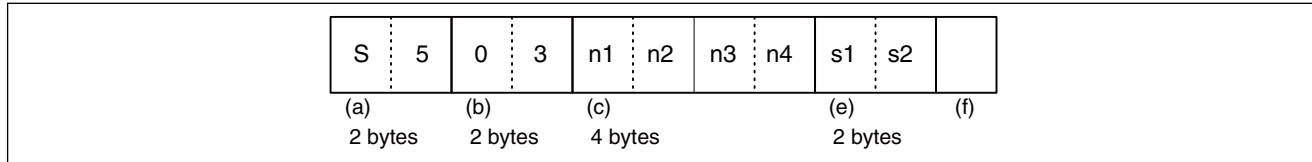
The S3 type differs from the S1 type in the above (c) field size, and is a record to store object data which requires the 4-byte address.

C.5 S5 Type (Record to Manage the Number of Records)

This record sets the number of records contained in a file and may not be specified. It can appear anywhere between S0 and S9.

■ S5 Type (Record to Manage the Number of Records)

Figure C.5-1 S5 Type (Record to Manage the Number of Records)



The S5 type consists of the above four fields (a) to (c), (e).

(a):

Type field. "S5"(0x5335)in ASCII code.

(b):

Indicates the number of bytes in (c) and (e).

(See the description in "Appendix C.1 S0 Type (Header Record)".)

(c):

Indicates the number of data records (S1, S2, S3) in a file.

n1 indicates the high-order digits and n4 indicates the low-order digits. Values in the range of 0 to 65535 can be set.

It is "0000" to "FFFF" in ASCII notation and "0x30303030" to "0x46464646" in hexadecimal notation.

(d):

This field does not exist.

(e):

Checksum.

(See the description in "Appendix C.1 S0 Type (Header Record)".)

(f):

Generally, control code (such as CR and LF) is added.

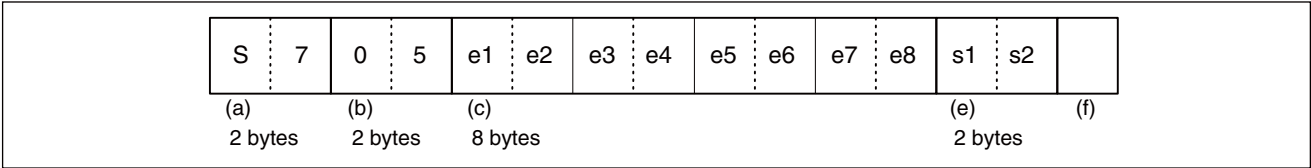
(See the description in "Appendix C.1 S0 Type (Header Record)".)

C.6 S7 Type (Terminator Record)

This record indicates the end of a file and contains the start address of execution.
This record is placed at the end of a file.
This terminator record is used when 4 bytes are required to represent the start address of execution.

■ S7 Type (Terminator Record)

Figure C.6-1 S7 Type (Terminator Record)



The S7 type consists of the above four fields (a) to (c), (e).

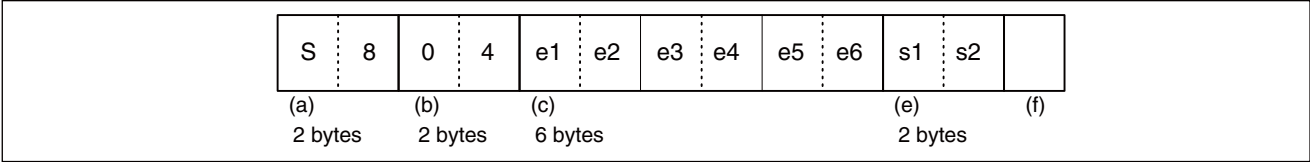
- (a):**
Type field. "S7"(0x5337)in ASCII code.
- (b):**
Indicates the number of bytes in (c) and (e). Always "05".
- (c):**
Indicates the start address of execution.
e1 indicates the high-order digits and e8 indicates the low-order digits.
- (d):**
This field does not exist.
- (e):**
Checksum.
- (f):**
Generally, control code (such as CR and LF) is added.

C.7 S8 Type (Terminator Record)

The S8 type differs from the S7 type in (c) field size and is a terminator record when 3 bytes are required to represent the start address of execution.

■ S8 Type (Terminator Record)

Figure C.7-1 S8 Type (Terminator Record)



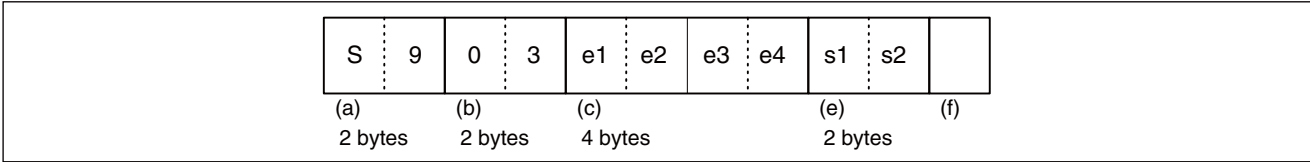
The S8 type differs from the S7 type in the above (c) field size and is a terminator record when 3 bytes are required to represent the start address of execution.

C.8 S9 Type (Terminator Record)

The S9 type differs from the S7 type in (c) field size and is a terminator record when 2 bytes are required to represent the start address of execution.

■ S9 Type (Terminator Record)

Figure C.8-1 S9 Type (Terminator Record)



The S9 type differs from the S7 type in the above (c) field size and is a terminator record when 2 bytes are required to represent the start address of execution.

APPENDIX D LIST OF LINKER OPTIONS

Table D-1 lists the linker options.

■ List of Linker Options

Table D-1 List of Linker Options (1 / 3)

Function	Option	Remarks
Specification for outputting load module file name	-o	Default
Specification for debugging information output	-g	
Specification for deleting debugging information	-Xg	Default
Specification for outputting absolute format load module	-a	Default
Specification for outputting relative format load module	-r	
Map list file name specification	-m	Default
Specification for inhibiting map list output	-Xm	
Canceling the omission of names displayed in the list	-dt	
Specification for outputting memory used information list	-mmi	
Specification of the number of digits in the list line	-pw	Default 132
Specification of the number of lines on one list page	-pl	Default 60
Specification for warning message output level	-w	
ROM area specification	-ro	
RAM area specification	-ra	
Section allocation	-sc	
Section group specification	-gr	
Register bank area specification	-rg	
Automatic allocation specification	-AL	Default 0
Retrieval library file specification	-l	
Library retrieval path specification	-L	
Library specification for each symbol	-el	
Library retrieval inhibit specification	-nl	
Default library retrieval inhibit specification	-nd	
Entry address specification	-e	
Dummy setting of external symbol values	-df	

Table D-1 List of Linker Options (2 / 3)

Function	Option	Remarks
Target CPU Specification	-cpu	must be specified
Specifying CPU Information File	-cif	
Specification for Inhibiting Check for Presence of Debug Data	-NCI0302LIB	
Function that sets automatically internal ROM/RAM areas	-set_rora	Default
Specifies to prevent the internal ROM/RAM areas from being set automatically	-Xset_rora	
User-specified-area check specification	-check_rora	
User-specified-area check suppression specification	-Xcheck_rora	Default
Section-placed-area check specification	-check_locate	
Section-placed-area check suppression specification	-Xcheck_locate	Default
Specification for check user-unspecified section	-check_section	
Specification for inhibit check for presence of user-unspecified section	-Xcheck_section	Default
Object mixing check specification	-objmixchk	Default
Object mixing check inhibit specification	-Xobjmixchk	
Specification for relative assemble list input directory	-alin	
Specification for absolute assemble list output directory	-alout	
Specification for absolute assemble list output	-als	
Specification for absolute assemble list output module	-alsf	
Specification for inhibiting absolute assemble list output	-Xals	
Specification for outputting ROM/RAM and ARRAY lists	-alr	
Specification for ROM/RAM and ARRAY lists output module	-alrf	
Specification for inhibiting ROM/RAM and ARRAY lists output	-Xalr	
Specification for ROM/RAM and ARRAY lists symbol and address display position	-na/-an	
Specification for outputting external symbol cross-reference information list	-xl	
Specification for external symbol cross-reference information list file name	-xlf	
Specification for inhibiting external symbol cross-reference information list output	-Xxl	
Specification for outputting local symbol list	-sl	
Specification for local symbol list file name	-slf	
Specification for inhibiting local symbol list output	-Xsl	

Table D-1 List of Linker Options (3 / 3)

Function	Option	Remarks
Specification for outputting section detail map list	-ml	
Specification for section detail map list file name	-mlf	
Specification for inhibiting section detail map list output	-Xml	
Specification for inhibiting default option file read	-Xdof	* Common option
Option file read specification	-f	* Common option
Specification for help message display	-help	* Common option
Specification for outputting version number/message	-V	* Common option
Specification for inhibiting version number/message output	-XV	* Common option
End message display specification	-cmsg	* Common option
End message display inhibit specification	-Xcmsg	* Common option
Specification to set end code to 1 when warning occurs	-cwno	* Common option
Specification to set end code to 0 when warning occurs	-Xcwno	* Common option

APPENDIX E LIST OF LIBRARIAN OPTIONS

Table E-1 lists the librarian options.

■ List of Librarian Options

Table E-1 List of Librarian Options

Function	Option	Remarks
Module addition (registration)	-a	
Module replacement (registration)	-r	
Module deletion	-d	
Module extraction	-x	
List file output specification	-m	
Specification for inhibiting list file output	-Xm	Default
Specification for outputting list file detail information	-dt	s, d, r, a
Specification of the number of lines on one list page	-pl	Default 60
Specification of the number of digits in one list line	-pw	Default 70
Creating backup file	-b	
Inhibiting backup file creation	-Xb	Default
Library file content inspection	-c	
File content optimization	-O	
Specification for outputting debugging information	-g	
Specification for inhibiting the output of debugging information	-Xg	
Target CPU specification	-cpu	must be specified
Specifying CPU Information File	-cif	
Object mixing check specification	-objmixchk	Default
Object mixing check inhibit specification	-Xobjmixchk	
Specification for inhibiting default option file read	-Xdof	* Common option
Option file read specification	-f	* Common option
Help message display specification	-help	* Common option
Specification for outputting version number/message	-V	* Common option
Specification for inhibiting version number/message output	-XV	* Common option
End message output specification	-cmsg	* Common option
Specification for inhibiting end message output	-Xcmsg	* Common option
Specification to set end code to 1 when warning occurs	-cwno	* Common option
Specification to set end code to 0 when warning occurs	-Xcwno	* Common option

APPENDIX F LIST OF COMMANDS AND OPTIONS OF THE OBJECT FORMAT CONVERTER

Table F-1 lists the commands of the object format converter and Table F-2 lists the options of the object format converter.

■ List of Commands of the Object Format Converter

Table F-1 List of Commands of the Object Format Converter

Command name	Function
f2ms	Absolute format load module → S format
f2hs	Absolute format load module → HEX format (HEX8/HEX16/HEX32)
f2is	Absolute format load module → HEX8 format
f2es	Absolute format load module → HEX16 format
m2ms	S format → S format (Adjusted)
h2hs	HEX format → HEX format (Adjusted)
m2bs	S format → binary data (memory image)
h2bs	HEX format → binary data (memory image)
m2is	S format → HEX8 format
m2es	S format → HEX16 format
i2ms	HEX8 format → S format
e2ms	HEX16 format → S format

■ List of Options of the Object Format Converter

Table F-2 List of Options of the Object Format Converter

Function	Option	Remarks
Specification for outputting file name	-o	
Padding data specification	-p	
Output range specification	-ran	m2ms,h2hs,m2bs,h2bs only need
Split mode specification	-sp	m2b only
Specification for inhibiting split mode	-Xsp	m2b only
Map list file creation specification	-m	m2b only
Specification for inhibiting map list file creation	-Xm	m2b only
Specifying an output file S Format	-S1,-S2, -S3	f2ms,m2ms only
Specifying an output file HEX format	-I16, -I20-I32	f2hs,h2hs only
Specifying to adjust	-adjust	f2ms,f2hs only
Specifying to change the starting address	-ST	m2ms,h2hs only
Specifying to output start address record	-entry	f2hs only
Specifying not to output start address record	-Xentry	f2hs only
Specification for inhibiting default option file read	-Xdof	* Common option
Option file read specification	-f	* Common option
Help message display specification	-help	* Common option
Specification for version number/message output	-V	* Common option
Inhibiting version number/message output	-XV	* Common option
End message display specification	-cmsg	* Common option
Specification for inhibiting end message display	-Xcmsg	* Common option
Specification to set end code to 1 when warning occurs	-cwno	* Common option
Specification to set end code to 0 when warning occurs	Xcwno	* Common option

APPENDIX G SPECIFICATION DIFFERENCES DEPENDING ON THE OS

Table G-1 to Table G-3 show specification differences depending on the OS.

■ Specification Differences Depending on the OS

Table G-1 Specification Differences Depending on the OS

OS-dependent item	OS type	
	for UNIX system OS	for Windows
Distinction of uppercase/lowercase characters in the file name	Yes	No
Default extension of the file name	Lowercase character	Uppercase and lowercase characters are not distinguished
Working directory when no environmental variable TMP is not specified	/tmp	Current directory
File specification using the wild card on the command line	Expanded by the shell and delivered to tools	Expanded inside tools

Table G-2 Differences in Wild Card Expansion

Wild card pattern	OS type	
	for UNIX system OS	for Windows
?	Matches any one character	Matches the null character or any one character
*	Matches any character string	Matches any character string

Table G-3 Concrete Examples of Wild Card Expansion

Wild card pattern	OS type	
	for UNIX system OS	for Windows
a?.obj	a1.obj matches. a.obj does not match.	a1.obj matches. a.obj also matches.
a*	a1.obj, a.obj, and a.abs match.	a1.obj, a.obj, and a.abs match.
*	abcz and abc.z match.	abcz and abc.z match.

APPENDIX H IDENTIFICATION METHOD FOR SAME OBJECT WHEN SOFTUNE LANGUAGE TOOL IS TRANSFERRED

This document describes the procedure to examine whether the new and old absolute format files are exactly the same at the binary level when the source program for the old 8L assembler (asm96) is transferred to the new assembler (fasm896s).

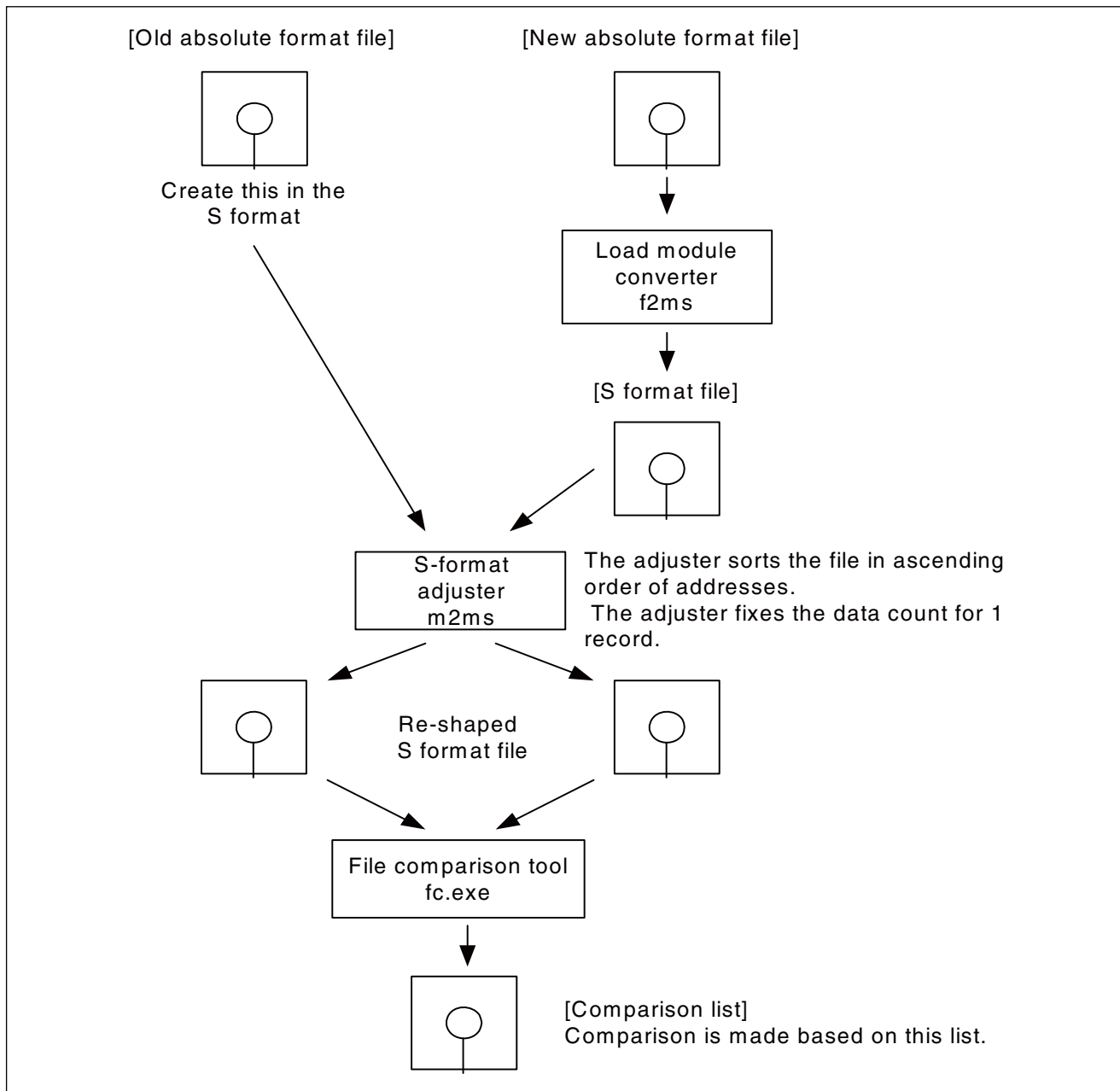
- H.1 Outline of Comparison Procedure
- H.2 Execution Example
- H.3 When Data Mismatched

H.1 Outline of Comparison Procedure

The absolute format files created by the new and old assemblers are converted to S format files, and the results are compared.

■ Outline of Comparison Procedure

The absolute format files created by the new and old assemblers are converted to S format files, and the results are compared.



Ignore the differences, if any, between the S0 record at the first line and the S7 record at the last line.

H.2 Execution Example

The absolute format file created by the old assembler is "old.mhx". The absolute format file created by the new assembler is "new.abs".

■ Execution Example

The absolute format file created by the old assembler is "old.mhx". The absolute format file created by the new assembler is "new.abs".

Note:

For the absolute format file created using the tool for old 8L, specify "FM" at linkage to create it in the S format.

```
>m2ms old.mhx -o old.ahx -ran 0xff00,0xffff -S3
                                ; Sorts old S format
>f2ms new.abs -o new.mhx      ; Converts new absolute format file to S format
>m2ms new.mhx -o new.ahx -ran 0xff00,0xffff -S3
                                ; Sorts new S format
>fc old.ahx new.ahx          ; Compares files
```

The comparison results may indicate the following difference, but data with a record beginning with "S0" or "S7" should be ignored.

This record indicates a comment, so a match is not required.

```
[Output example]
>fc old.ahx new.ahx
***** old.ahx
S0080000746573743106
***** new.ahx
S0080000544553543285
*****
```


H.3 When Data Mismatched

The following shows an FC output example. This information is used to locate mismatched locations.

■ When Data Mismatched

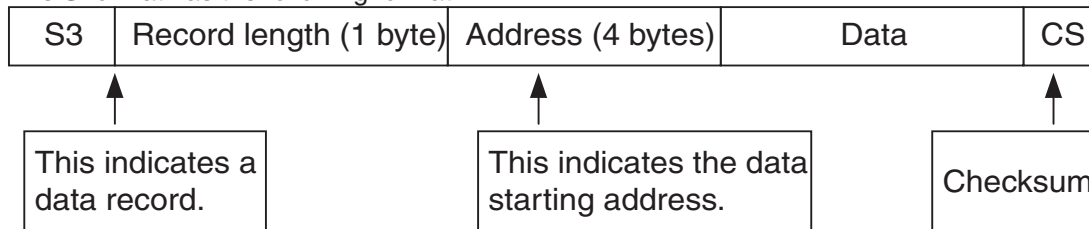
The following shows an FC output example. This information is used to locate mismatched locations..

[Output example]

```
fc old.ahx new.ahx
***** old.ahx
S3150000FF00000102032C37060708090A0B0C0D0E0F19
***** new.ahx
S3150000FF00000102030405060708090A0B0C0D0E0F73
*****
```

■ Obtaining Addresses from S Format

The S format has the following format:



Note: Ignore records other than "S3" records.

The address of data with a mismatch can be obtained from the above output.

S3150000FF00000102030405060708090A0B0C0D0E0F73

This record data begins at address 0x00ff00. From this, it is determined that there is mismatched data at "0405" at addresses 0xff05, and 0xff06.

■ Locating Module Using Link Map

The module where the address obtained S3 records is written is located using the link map.

At linkage, the -mlf option and the -als option must be specified.

The -mlf option creates detailed map information and the -als option creates the absolute assemble list.

[Output example]

```
>flnk896s main sub1 - AL 2 - mlf main.mpm - als - ro rom=0xff00/0xffff - ra rom=0/0xff
```

Detailed map information (main.mpm)

Section Mapping List							
Module(s)							
1. main							
2. sub1							
S_Addr	-E_Addr	Size	Section	Type	Al	M.No	Sec.(Top 80)
0000FF00	- 0000FF0F	00000010	CONST	P R--I	02	1	data
0000FF10	- 0000FF1F	00000010	CONST	P R--I	02	2	data

From the detailed map information, it is determined that the appropriate addresses (0xff05, 0xff06) are defined by the "main" module (module No. 1).

■ Locating Mismatched Locations

Mismatched locations are located from the absolute assemble list created in "Locating Module Using Link Map".

Absolute assemble list (main.als)

SN	LOC	OBJ	LLINE	SOURCE
				<main.asm> =====
			1	
			2	.TITLE main
			3	.PROGRAM main
			4	
			5	
			6	.SECTION data
				da FF00 -----<data>-----
				, CONST, ALIGN=2
			7	.DATA.B 0,1,2,3,44,55
				da FF00 00 01 02 03 2C 37 06
				da FF07 07 08 09 0A 0B 0C 0D
				da FF0E 0E 0F
			8	;data ends
			9	.END
				==

APPENDIX I DIFFERENCE IN SPECIFICATION OF SOFTUNE LINKER(FLNK896S) AND OLD LINKER(LINK96)

The difference in the specification of the SOFTUNE linker (FLNK896S) and the old linker (LINK96) are shown below.

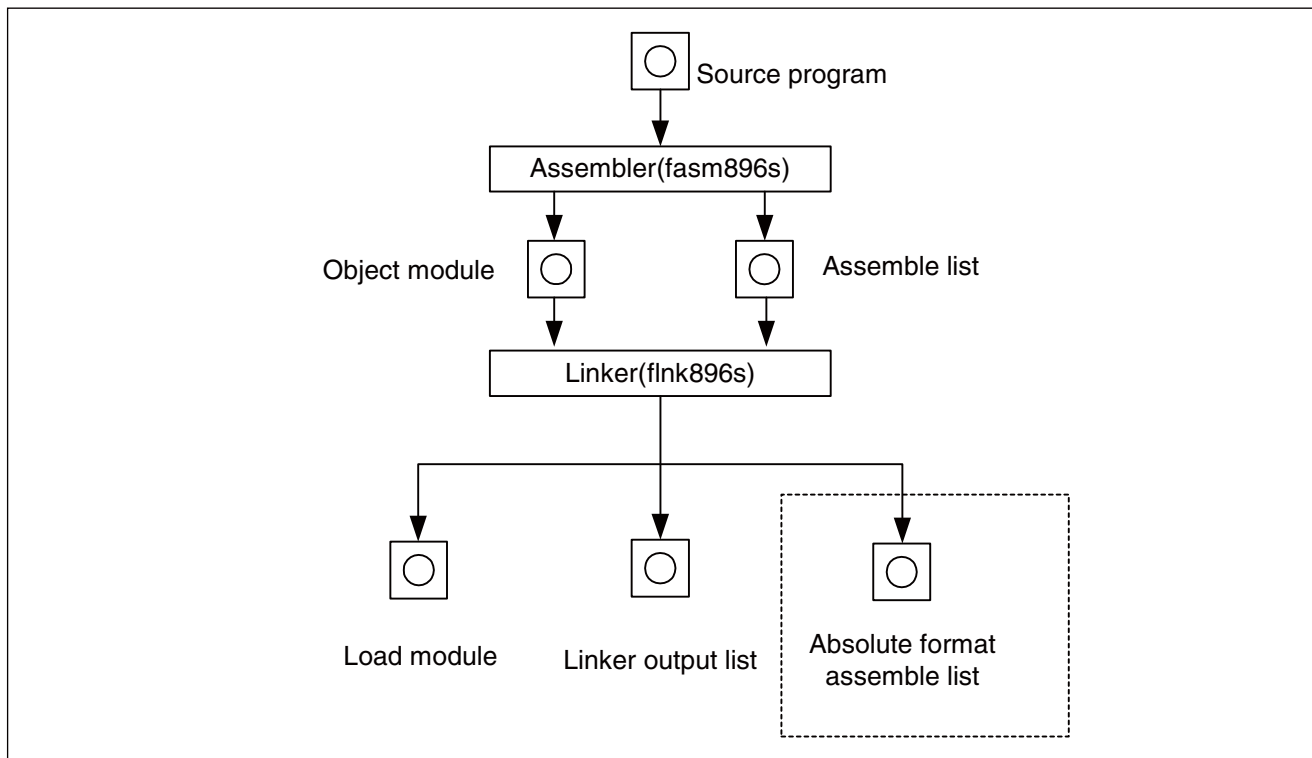
■ Preface

FLNK896S is updated from LINK96 in the following points:

- Command line input
- Absolute assemble list creation function added to linker
- Alleviation of restrictions (input file count, section count, symbol count, etc.)
- Enhancement of environmental variables
- Support for external symbol cross reference information list creation function
- Support for symbol detailed information list creation function
- Support for section map creation function for each module

■ Absolute Assemble List Creation Function

The function of the absolute assemble list creation tool is added to linker.



■ Difference in Command Lines

The difference in the command line specification at activation is given in Table I-1 and Table I-2.

Table I-1 Activation Format for Linker

FLNK896S	LINK96
flnk896S [option].... [file name]	link96 file name [option]..
	link96 @option file

Table I-2 Option Specification for Linker

	FLNK896S	LINK96
Where to specify options	Any position allowed	After specification of file name
Specification of option file	-f option file	@ option file
Specification of two or more option files	Allowed Not	Allowed

■ Environmental Variables

The difference in the environmental variables is given in Table I-3.

Table I-3 Environmental Variables

	FLNK896S	LINK96
Specification of working directory	TMP	TMP
Specification of installation path	FETOOL	-
Specification of display character code	FELANG	-
Specification of default option file store path	OPT896	-
Specification of library retrieval path	LIB896	96LIB(MS-DOS) LIB96(UNIX)

Table I-4 Linker Options-1

Function	FLNKB896S	LINK96
Load module output specification	Not required	-OJ
Load module output suppression	-	-NJ
Error information file output allowed/suppressed	-	-EP/-NOEP
Specification for file of OS return value	-	-RC
Specification for description termination	Not required	-END
Output load module file name specification	-o	-OJ
Debug information output specification	-g	-DBG
Debug information deletion specification	-Xg	-
Absolute format load module output specification	-a	-
Relative format load module output specification	-r	-
Map list file name specification	-m	-L
Map list output suppression specification	-Xm	-NL
List display part name omission cancel	-dt	-NP
Output specification of memory used Information List	-mmi	-
List 1 line digit count specification	-pw	PW
List 1 page line count specification	-pl	-PL
Warning message output level setting	-w	-NW
ROM area specification	-ro	-ROMSIZE
RAM area specification	-ra	-RAMSIZE
Section arrangement order	-sc	-SEG
Section group Specification	-gr	-
Register bank area setting	-rg	-RG
Auto arrangement specification	-AL	-
Retrieval library file specification	-l	-LIB
Library retrieval path specification	-L	-
Library specification for individual symbols	-el	-
Library retrieval suppression specification	-nl	-
Default library retrieval suppression specification	-nd	-
Entry address specification	-e	-
External symbol value dummy setting	-df	-
Target CPU specification	-cpu	-
Specifying inhibit check for presence of debug data	-NCI0302LIB	-

Table I-5 Linker Options-2

Function	FLNKB896S	LINK96
Function that sets automatically internal ROM/ RAM areas	-set_rora	-
Specifies to prevent the internal ROM/RAM areas from being set automatically	-Xset_rora	-
User-specified-area check specification	-check_rora	-
User-specified-area check suppression specification	-Xcheck_rora	-
Section-placed-area check specification	-check_locate	-
Section-placed-area check suppression specification	-Xcheck_locate	-
Specifying check user-unspecified section	-check_section	-
Specifying inhibit check for presence of user-unspecified section	-Xcheck_section	-
Object mixing check specification	-objmixchk	-
Object mixing check inhibit specification	-Xobjmixchk	-
Specification for relative assemble list input directory	-alin	-
Specification for absolute assemble list output directory	-alout	-
Specification for absolute assemble list output	-als	-
Specification for absolute assemble list output module	-alsf	-
Specification for inhibiting absolute assemble list output	-Xals	-
Specification for ROM/RAM and ARRAY lists output	-alr	-
Specification for ROM/RAM and ARRAY lists output module	-alrf	-
Specification for inhibiting ROM/RAM and ARRAY lists output	-Xalr	-
Specification for ROM/RAM and ARRAY lists symbol and address display position	-na/-an	-

Table I-6 Linker Options-3

Function	FLNKB896S	LINK96
External symbol cross reference information list output	-xl	-
External symbol cross reference information list file name	-xlf	Not required
External symbol cross reference information list output suppression	-Xxl	-
Local symbol list output specification	-sl	-SL/-HSL
Local symbol list file name specification	-slf	-
Local symbol list output suppression specification	-Xsl	-
Section detail map list output specification	-ml	-
Section detail map list file name specification	-mlf	-
Section detail map list output suppression	-Xml	-
Default option file read suppression specification	-Xdof	-
Specifying option file read	- f	-
Specifying display of help message	- help	-
Specifying version number and message output	- V	-
Suppression version number and message output	- XV	-
Specifying display of end message	- cmsg	-
Specifying for suppression to display end message	- Xcmsg	-
Specifying to set the end code to 1 when warning is issued	- cwno	-
Specifying to set the end code to 0 when warning is issued	- Xcwno	-

■ Alleviation of Restrictions

Table I-7 Processing Limits

	FLNK896S	LINK96
Input file count	65535	255
Linked module count	65535	255
Linked section count	65535	255
External definition symbol count	65535	65534
External reference symbol count	65535	65535

APPENDIX J DIFFERENCE IN SPECIFICATION OF SOFTUNE LIBRARIAN (FLIB896S) AND OLD LIBRARIAN (LIB96)

The difference in the specification of the SOFTUNE librarian (FLIB896S) and the old librarian (LIB96) are shown below.

■ Difference in Command Lines

The difference in the command line specification at activation is given in Table J-1 and Table J-2.

Table J-1 Activation Format for Librarian

FLIB896S	LIB96
flib896s [option]... [file name]	lib96 file name [option]...
	lib96 @option file

Table J-2 Option Specification for Librarian

	FLIB896S	LIB96
Where to specify options	Any position	After file name
Specification of option file	-f option file	@ option file
Specification of two or more option files	Allowed	Not Allowed

■ Environmental Variables

The difference in environmental variables is given in Table J-3.

Table J-3 Environmental Variables

	FLIB896S	LIB96
Specification of working directory	TMP	TMP
Specification of installation path	FETOOL	-
Specification of display character code	FELANG	-
Specification of default option file store path	OPT896	-
Specification of library retrieval path	LIB896	-

Table J-4 Option of Librarian

Function	Option	Remarks
Module selection	-	-S
Error information file output enabled/inhibited	-	-EP/-NEP
Status code file output	-	-RETCODE(-RC)
Termination of option file	Not required	-E
Module addition (registration)	-a	-A
Module replacement (registration)	-r	-R
Module deletion	-d	-D
Module extraction	-x	-X
List file output specification	-m	-L
Module name list output	Always output	-LM
Specifying list file output inhibited	-Xm	-
Output specification of list file detail information Segment information list output External definition symbol information output External reference symbol information output	-dt -dt s -dt d -dt a -dt r -dt a	- -LP -dt r -dt a
Creation of cross reference	-	-LX
List 1 page line count specification	-pl	-
List 1 line digit count specification	-pw	-
Backup file creation allowed/suppressed	-b / -Xb	-
Library file contents check	-c	-
File contents optimization	-O	-
Debug information output allowed/suppressed	-g / -Xg	-
Specifying a target CPU	-cpu	-
Object mixing check specification	-objmixchk	-
Object mixing check inhibit specification	-Xobjmixchk	-
Specifying suppression to read default option file	-Xdof	-
Specifying option file read	-f	-
Specifying display of help message	-help	-
Specifying version number and message output	-V	-
Suppression to output version number and message	-XV	-
Specifying output of end message	-cmsg	-
Specifying suppression to output end message	-Xcmsg	-
Specifying to set the end code to 1 when warning is issued	-cwno	-
Specifying to set the end code to 0 when warning is issued	-Xcwno	-

Table J-5 Processing Limits

	FLIB896S	LIB96
Input file count	65535	255
Linked module count	65535	255
Linked section count	65535	255
External definition symbol count	65535	4000
External reference symbol count	65535	1500

APPENDIX K Major Changes

Page	Section	Change Results
Revision 4.1		
-	-	Company name and layout design change
Revision 5.0		
9	1.5 Startup Message ■ Startup Message	Deleted the display example of Startup message.
34	3.2.4 Specifying Version Number/Message Output (-V) ■ Specifying Version Number/Message Output (-V)	Deleted the example.
318	APPENDIX A ERROR MESSAGES OF THE LINKAGE KIT Error code: F9999L	Revised the contact details.
328	APPENDIX A ERROR MESSAGES OF THE LINKAGE KIT Error code: F9999U	Revised the contact details.



INDEX

The index follows on the next page.
This is listed in alphabetic order.

Index

A

-a

Absolute Format Load Module Output Specification
(-a) 95

Adding (Registering) a Module (-a) 212

Absolute Assemble List

Absolute Assemble List Creation Function 359

Absolute Assemble List Format 178

Absolute Assemble List Output Directory
Specification (-alout) 150

Absolute Assemble List Output Inhibit Specification
(-Xals) 153

Absolute Assemble List Output Module Specification
(-alsf) 152

Absolute Assemble List Output Specification
(-als) 151

Absolute Format

Absolute Format Assemble List 168

Absolute Format Load Module Output Specification
(-a) 95

Adding

Adding (Registering) a Module (-a) 212

-adjust

Adjust Option (-adjust) 258

Specifying to Adjust (-adjust) 263

-AL

Automatic Allocation Specification (-AL) 122

Example of Location when -AL 1 is Specified 68

Example of Location when -AL 2 is Specified 71

-alin

Relative Assemble List Input Directory Specification
(-alin) 149

Alleviation

Alleviation of Restrictions 363

Allocation

Allocation/Link Options 91

Automatic Allocation Specification (-AL) 122

Section Allocation Detailed Information
List File 190

Section Allocation Order/Address Specification
(-sc) 116

-alout

Absolute Assemble List Output Directory
Specification (-alout) 150

-alr

ROM/RAM and ARRAY Lists Output Specification
(-alr) 154

-alrf

ROM/RAM and ARRAY Lists Output Module
Specification (-alrf) 155

-als

Absolute Assemble List Output Specification
(-als) 151

-alsf

Absolute Assemble List Output Module Specification
(-alsf) 152

-an

ROM/RAM and ARRAY Lists Symbol and Address
Display Position Specification
(-na, -an) 157

Assemble Source List Format

Assemble Source List Format 183

Attribute

Section Combination Attribute 60

Section Location Attribute 59

Automatic Allocation Specification

Automatic Allocation Specification (-AL) 122

Automatically Locating Sections

Automatically Locating Sections 66

B

-b

Creating a Backup File (-b) 221

Backup File

Creating a Backup File (-b) 221

Inhibiting the Creation of a Backup File
(-Xb) 222

Binary Converter

Binary Converter 246

Cautionary Information Concerning Binary Converter
and Format Adjuster 300

List of Options of the Binary Converter 284

Outline of the Binary Converter 282

C

-c

Checking the Contents of a Library File
(-c) 223

Canceling

Canceling the Omission of Names Displayed in the
List (-dt) 102

Cautionary Information

Cautionary Information Concerning Binary Converter
and Format Adjuster 300

Change	
Changing an Output File Name (-o)	251
Changing the Format of a List File.....	55
Specifying to Change the Starting Address of the Record (-ST).....	274
Character	
Character Code of the Filename.....	13
Number of Characters for the Filename	13
Types of Characters Consisting of Identifiers.....	12
Check	
Checking the Contents of a Library File	201, 206
Checking the Contents of a Library File (-c)	223
Object Mixing Check Inhibit Specification (-Xobjmixchk)	148, 230
Object Mixing Check Specification (-objmixchk).....	147, 229
Section-placed-area Check Specification (-check_locate)	139
Section-placed-area Check Suppression Specification (-Xcheck_locate).....	143
Selecting the Warning Check Level	55
Specification for Inhibiting Check for Presence of Debug Data (-NCI0302LIB).....	133
Specifying Check User-unspecified Section (-check_section).....	144
Specifying Inhibit Check for Presence of User- unspecified Section (-Xcheck_section) ..	146
User-specified-area Check Specification (-check_rora)	136
User-specified-area Check Suppression Specification (-Xcheck_rora)	138
-check_locate	
Section-placed-area Check Specification (-check_locate)	139
-check_rora	
User-specified-area Check Specification (-check_rora)	136
-check_section	
Specifying Check User-unspecified Section (-check_section).....	144
-cif	
CPU Information File Specification (-cif).....	228
-cmsg	
-cmsg Option	29
End Message and -cmsg Option.....	10
Specifying Display of End Message (-cmsg).....	36
Code	
Character Code of the Filename.....	13
End Code Value and End Status	8
Specifying End Code to 0 when Warning is Issued (-Xcwno).....	39
Specifying End Code to 1 when Warning is Issued (-cwno)	38
Combination	
Section Combination Attribute.....	60
Shared Combination of Sections.....	61
Simple Connection Combination of Sections	61
Command	
Command Line Format	5
Difference in Command Lines	360, 364
Example of Specifying Command Lines	26
Executing a Command of an Object Format Converter.....	248
List of Commands of the Object Format Converter.....	351
Comment	
Specifying Comment in the Option File	44
Common	
Common Format.....	330
Common Options of an Object Format Converter.....	250
List of Common Options	28
Comparison	
Outline of Comparison Procedure.....	355
Configuration	
Configuration of a List File.....	232
Configuration of Link List File	169
Connection	
Simple Connection Combination of Sections	61
Control	
Control on Combining and Locating Sections.....	54
Control on Input-output Files and Messages	53
Control on Searching Libraries.....	54
Converter	
Binary Converter	246
Cautionary Information Concerning Binary Converter and Format Adjuster.....	300
Common Options of an Object Format Converter.....	250
Error Messages of the Converter	325
Executing a Command of an Object Format Converter.....	248
List of Commands of the Object Format Converter.....	351
List of Options of the Binary Converter	284
List of Options of the Load Module Converter.....	257
List of Options of the Object Format Converter.....	352
Outline of Load Module Converter.....	256
Outline of Object Format Converter	244
Outline of the Binary Converter	282
Restrictions on an Object Format Converter	300
Types of Load Module Converters	246
Types of Other Converters.....	247

CPU

CPU Information File.....	84
CPU Information File Specification	
(-cif).....	132, 228
Specifying a Target CPU (-cpu)	227
Target CPU Specification (-cpu)	131

-cpu

Specifying a Target CPU (-cpu)	227
Target CPU Specification (-cpu)	131

Creating

Creating a Backup File (-b).....	221
Creating a Group of Sections	56
Creating a New Library File	201, 202
Options for Creating and Editing a Library	211

Cross-reference

Cross-reference List Format.....	186
External Symbol Cross-reference Information List	
File.....	187
External Symbol Cross-reference Information List	
File Name Specification (-xlf).....	159
External Symbol Cross-reference Information List	
Output Inhibit Specification (-Xxl)	160
External Symbol Cross-reference Information List	
Output Specification (-xl)	158

-cs

Checksum specification of ROM area (-cs)	106
--	-----

-cwno

-cwno Option	29
Specifying End Code to 1 when Warning is Issued	
(-cwno).....	38

D

-d

Deleting a Module (-d).....	214
-----------------------------	-----

Data Length

Specifying the Data Length in an Output Record	
(-len)	275
Specifying the Output Record Data Length	
(-len)	274

Data Record

Data Record (HEX8/HEX16/HEX32).....	332
S1 Type (Data Record: 2-byte Address)	340
S2 Type (Data Record: 3-byte Address)	341
S3 Type (Data Record: 4-byte Address)	342

Debug

Debug Information Delete Specification	
(-Xg)	94
Debug Information Output Specification	
(-g).....	93
Deleting Debugging Information.....	201, 205
Inheriting Debugging Information	55
Specifying not to Output Debugging Information	
(-Xg)	226
Specifying to Output Debugging Information	
(-g).....	225

Default

Default Library Retrieval Inhibit Specification	
(-nd).....	128
Default Option File.....	46

Delete

Debug Information Delete Specification	
(-Xg)	94
Deleting a Module (-d).....	214
Deleting Debugging Information	201, 205

Detailed Information

Section Allocation Detailed Information	
List File	190
Specifying to Output Detailed Information of a List	
File (-dt)	218

Determining

Determining Location Addresses	67, 71
--------------------------------------	--------

-df

Dummy Setting of External Symbol Values	
(-df)	130

Difference

Difference in Command Lines	360, 364
Specification Differences Depending	
on the OS.....	353

Directory

Absolute Assemble List Output Directory	
Specification (-alout)	150
LIB896 (Library File Search Directory).....	14
OPT (Default Option File Storage Directory)	14
OPT896 (Default Option File	
Storage Directory)	14
Relative Assemble List Input Directory Specification	
(-alin).....	149
TMP (Work Directory)	14, 15

Display

Displaying Identifier Name	
when Outputting List	12
Displaying the Contents of a	
Library File.....	201, 206
Linkage Kit Error Message Display Format	305

-dt

Canceling the Omission of Names Displayed in the	
List (-dt)	102
Specifying to Output Detailed Information of a List	
File (-dt)	218

Dummy

Dummy Setting of External Symbol Values	
(-df)	130

E

-e

Entry Address Specification (-e)	129
--	-----

e2ms

e2ms (Converting a HEX16 Format File into the S	
Format)	297

Editing	Specified.....	65
Editing a Library File	Example of Operation	201, 202
Options for Creating and Editing a Library	Example of Specifying an Option that has a Contradictory Relation with Other Options.....	211
-el	Example of Specifying an Option that has an Inclusive Relation with Other Options	25
Library Specification for Each Symbol (-el)	Example of Specifying Command Lines	126
End	Execution Example	356
End Code Value and End Status	List Display Example of the Control List Part	8
End Message	List Display Example of the Map List Part	10
End Message and -cmsg Option.....	List Display Example of the Symbol List Part	10
End Record (HEX8/HEX16/HEX32)		333
Specifying Display of End Message (-cmsg).....		36
Specifying End Code to 0 when Warning is Issued (-Xcwno).....		39
Specifying End Code to 1 when Warning is Issued (-cwno)		38
Specifying Suppression to Display End Message (-Xcmsg).....		37
Entry	Executing	
Entry Address Specification (-e)	Executing a Command of an Object Format Converter.....	248
Setting Entry Addresses and Symbol Values.....	Execution	
Specifying an Entry Address	Execution by Specifying Option File	42
-entry	Execution Example	356
Specifying to Output Start Address Record (-entry).....	Extended Linear Address Record	
Start Address Output Option (-entry)	Extended Linear Address Record (HEX32)	336
Environmental Variables	Extended Segment Address Record	
Environmental Variables.....	Extended Segment Address Record (HEX16/HEX32)	334
Error	External Symbol	
Error Messages for the Librarian	Dummy Setting of External Symbol Values (-df).....	130
Error Messages in the Assemble List.....	External Symbol Cross-reference Information List.....	168
Error Messages of the Converter.....	External Symbol Cross-reference Information List File	187
Error Messages of the Linker	External Symbol Cross-reference Information List File Name Specification (-xlf).....	159
Linkage Kit Error Message Classes.....	External Symbol Cross-reference Information List Output Inhibit Specification (-Xxl)	160
Linkage Kit Error Message Display Format	External Symbol Cross-reference Information List Output Specification (-xl)	158
Example	Setting an External Symbol Value	58
Example Display of the Memory Used Information List Part List.....	Extracting	
Example of a Search when there are Multiple Library Files (1).....	Extracting a Module (-x)	215
Example of a Search when there are Multiple Library Files (2).....	Extracting a Module from a Library File	201, 204
Example of a Search when there is One Library File (1)		
Example of a Search when there is One Library File (2)	F	
Example of a Search when there is One Library File (3)	-f	
Example of Describing Option File	-f Option	29
Example of Location when -AL 1 is Specified	Specifying Reading from Option Files (-f)	31
Example of Location when -AL 2 is Specified	f2es	
Example of Location when the Order of Combining Sections is not Specified	f2es (Converting an Absolute Format Load Module into the HEX16 Format).....	267
Example of Location when the Order of Combining Sections is Specified.....		
Example of Location when the Section Group is		

f2hs	m2es (Converting a S Format File into the HEX16 Format)	295
f2hs (Converting an Absolute Format Load Module into the HEX Format).....	m2is (Converting a S Format File into the HEX8 Format)	294
265	Map List File Name Specification (-m).....	100
f2is	Number of Characters for the Filename	13
f2is (Converting an Absolute Format Load Module into the HEX8 Format).....	OPT (Default Option File Storage Directory)	14
266	OPT896 (Default Option File Storage Directory)	14
f2ms	Optimizing the Contents of a File (-O)	224
f2ms (Converting an Absolute Format Load Module into the S Format)	Option File	42
264	Options for Searching and Protecting a File	211
FELANG	Order of Searching a Library File.....	73
FELANG	Output Load Module File Name Specification (-o)	92
14, 16	Restrictions on the Object File Format.....	194
FETOOL	Retrieval Library File Specification (-l)	124
FETOOL	Section Allocation Detailed Information List File	190
14, 17	Section Detail Map List File Name Specification (-mlf)	165
File	Specification to Continue in the Option File	43
Changing an Output File Name (-o).....	Specifying a Library File to be Searched.....	57
Changing the Format of a List File	Specifying a Library File to be Searched for Each Symbol.....	57
Character Code of the Filename	Specifying a List File Name	55
Checking the Contents of a Library File	Specifying an Output Load Module File Name.....	55
201, 206	Specifying Comment in the Option File.....	44
Checking the Contents of a Library File (-c)	Specifying not to Create a Map List File (-Xm).....	285, 291
223	Specifying not to Output a List File (-Xm).....	217
Configuration of a List File.....	Specifying Suppression of Default Option File (-Xdof)	30
Configuration of Link List File	Specifying to Create a Map List File (-m)	285, 289
CPU Information File.....	Specifying to Output a List File (-m).....	216
CPU Information File Specification (-cif)	Specifying to Output Detailed Information of a List File (-dt)	218
132, 228		
Creating a Backup File (-b).....	-fill	
Creating a New Library File	Specification to fill ROM area (-fill)	97
Default Option File	Forced Termination	
Displaying the Contents of a Library File	Forced Termination	7
201, 206	Format	
e2ms (Converting a HEX16 Format File into the S Format).....	Absolute Assemble List Format	178
297	Absolute Format Assemble List	168
Editing a Library File	Absolute Format Load Module Output Specification (-a).....	95
201, 202	Cautionary Information Concerning Binary Converter and Format Adjuster	300
Example of a Search when there is One Library File (1)	Changing the Format of a List File	55
74	Command Line Format	5
Example of a Search when there is One Library File (2)	Common Format	330
75		
Example of a Search when there is One Library File (3)		
76		
Example of Describing Option File		
45		
Execution by Specifying Option File		
42		
External Symbol Cross-reference Information List File.....		
187		
External Symbol Cross-reference Information List File Name Specification (-xlf).....		
159		
Extracting a Module from a Library File		
201, 204		
i2ms (Converting a HEX8 Format File into the S Format).....		
296		
Inhibiting the Creation of a Backup File (-Xb)		
222		
Link List File		
168		
Local Symbol Information List File.....		
188		
Local Symbol Information List File Name Specification (-slf)		
162		

Common Options of an Object	
Format Converter	250
Cross-reference List Format	186
e2ms (Converting a HEX16 Format File into the S Format)	297
Executing a Command of an Object	
Format Converter	248
f2es (Converting an Absolute Format Load Module into the HEX16 Format)	267
f2hs (Converting an Absolute Format Load Module into the HEX Format)	265
f2is (Converting an Absolute Format Load Module into the HEX8 Format)	266
f2ms (Converting an Absolute Format Load Module into the S Format)	264
Format Adjuster	246
Functions of the Format Adjuster	271
Header Format	180
HEX Format	329
i2ms (Converting a HEX8 Format File into the S Format)	296
Linkage Kit Error Message	
Display Format	305
List of Commands of the Object	
Format Converter	351
List of Options of the Format Adjuster	273
List of Options of the Object	
Format Converter	352
List Output Format of the Control List Part	170
List Output Format of the Map List Part	172
List Output Format of the Symbol List Part	177
m2es (Converting a S Format File into the HEX16 Format)	295
m2is (Converting a S Format File into the HEX8 Format)	294
Memory Used Information List Part List Output Format	174
Obtaining Addresses from S Format	357
Options Related to the Absolute Format Assemble List Output	91
Outline of Object Format Converter	244
Output HEX Format Option (-I16/-I20/-I32)	258, 260
Output S Format Option (-S1/-S2/-S3)	258, 259
Overview of the Format Adjuster	270
Relative Format Load Module Output Specification (-r)	96
Restrictions on the Object File Format	194
S Record Format	338
Specifying an Output HEX Format (-I16/-I20/-I32)	274, 279
Specifying an Output S Format (-S1/-S2/-S3)	274, 277
Specifying the Output Format	55
Function	
Absolute Assemble List Creation Function	359
Function that Automatically Sets Internal ROM/RAM Area (-set_rora)	134
Functions of the Format Adjuster	271
G	
-g	
Debug Information Output Specification (-g)	93
Specifying to Output Debugging Information (-g)	225
-gr	
Section Group Specification (-gr)	119
H	
Handling Variables	
Handling Variables with Initial Values	196
Header	
Header Format	180
Help	
Help Message	11
-help	
-help Option	29
Specifying Help Message (-help)	33
HEX	
HEX Format	329
HEX Format	
f2hs (Converting an Absolute Format Load Module into the HEX Format)	265
Output HEX Format Option (-I16/-I20/-I32)	258, 260
Specifying an Output HEX Format (-I16/-I20/-I32)	274, 279
HEX16	
Data Record (HEX8/HEX16/HEX32)	332
End Record (HEX8/HEX16/HEX32)	333
Extended Segment Address Record (HEX16/HEX32)	334
Start Segment Address Record (HEX16/HEX32)	335
HEX16 Format	
e2ms (Converting a HEX16 Format File into the S Format)	297
f2es (Converting an Absolute Format Load Module into the HEX16 Format)	267
m2es (Converting a S Format File into the HEX16 Format)	295
HEX32	
Data Record (HEX8/HEX16/HEX32)	332
End Record (HEX8/HEX16/HEX32)	333
Extended Linear Address Record (HEX32)	336
Extended Segment Address Record (HEX16/HEX32)	334
Start Linear Address Record (HEX32)	337

Start Segment Address Record (HEX16/HEX32).....	335	-len	Specifying the Data Length in an Output Record (-len).....	275
HEX8		Specifying the Output Record Data Length (-len).....	274	
Data Record (HEX8/HEX16/HEX32).....	332	LIB896		
End Record (HEX8/HEX16/HEX32).....	333	LIB896.....	18	
f2is (Converting an Absolute Format Load Module into the HEX8 Format).....	266	LIB896 (Library File Search Directory).....	14	
m2is (Converting a S Format File into the HEX8 Format).....	294	Librarian		
HEX8 Format		Error Messages for the Librarian.....	319	
i2ms (Converting a HEX8 Format File into the S Format).....	296	List of Librarian Options.....	350	
I		Mixing of Objects for Librarian.....	207	
-I16/-I20/-I32		Restrictions on a Librarian.....	238	
Output HEX Format Option (-I16/-I20/-I32).....	258, 260	Roles of Librarian.....	200	
Specifying an Output HEX Format (-I16/-I20/-I32).....	274, 279	Library		
i2ms		Checking the Contents of a Library File.....	201, 206	
i2ms (Converting a HEX8 Format File into the S Format).....	296	Checking the Contents of a Library File (-c).....	223	
Identifier		Creating a New Library File.....	201, 202	
Displaying Identifier Name when Outputting List.....	12	Default Library Retrieval Inhibit Specification (-nd).....	128	
Indicating Identifiers.....	12	Displaying the Contents of a Library File.....	201, 206	
Limiting the Number of Letters for Identifiers.....	12	Editing a Library File.....	201, 202	
Types of Characters Consisting of Identifiers.....	12	Example of a Search when there are Multiple Library Files (1).....	77	
Indicating		Example of a Search when there are Multiple Library Files (2).....	78	
Indicating Identifiers.....	12	Example of a Search when there is One Library File (1).....	74	
Inheriting		Example of a Search when there is One Library File (2).....	75	
Inheriting Debugging Information.....	55	Example of a Search when there is One Library File (3).....	76	
Inhibiting		Extracting a Module from a Library File.....	201, 204	
Inhibiting the Creation of a Backup File (-Xb).....	222	Inhibiting the Search for a Library.....	57	
Inhibiting the Search for a Library.....	57	LIB896 (Library File Search Directory).....	14	
Initial Values		Library Control Option.....	91	
Handling Variables with Initial Values.....	196	Library Retrieval Inhibit Specification (-nl).....	127	
Input		Library Retrieval Path Specification (-L).....	125	
Relative Assemble List Input Directory Specification (-alin).....	149	Library Specification for Each Symbol (-el).....	126	
Specifying Input Object Files.....	55	Options for Creating and Editing a Library.....	211	
Internal ROM/RAM Area		Order of Searching a Library File.....	73	
Specifies to Prevent the Internal ROM/RAM Area from being Automatically Set (-Xset_rora).....	135	Processing when Library Files are Individually Specified.....	79	
L		Retrieval Library File Specification (-l).....	124	
-L		Specifying a Library File to be Searched.....	57	
Library Retrieval Path Specification (-L).....	125	Specifying a Library File to be Searched for Each Symbol.....	57	
-l		Specifying a Library to be Searched.....	73	
Retrieval Library File Specification (-l).....	124	Specifying a Path to Search a Library.....	57	

Limiting		External Symbol Cross-reference Information List	
Limiting the Number of Letters		Output Specification (-xl)	158
for Identifiers	12	Link List File	168
Link		List Display Example of the Control	
Allocation/Link Options	91	List Part	171
Configuration of Link List File	169	List Display Example of the Map List Part	173
Link List File	168	List Display Example of the Symbol	
Links of Sections	62	List Part	177
Locating Module Using Link Map	358	List of Commands of the Object	
Other Link Control Options	91	Format Converter	351
Linkage Kit		List of Common Options	28
Linkage Kit Error Message Classes	304	List of Librarian Options	350
Linkage Kit Error Message		List of Linker Options	88, 347
Display Format	305	List of Options	210
Support Range of Linkage Kit	4	List of Options of the Binary Converter	284
Linker		List of Options of the Format Adjuster	273
Error Messages of the Linker	306	List of Options of the Load	
Linker Reservation Symbol	194	Module Converter	257
Linker Restrictions	194	List of Options of the Object	
List of Linker Options	88, 347	Format Converter	352
Mixing of Objects for a Linker	85	List Output Format of the Control List Part	170
Overview of a Linker	52	List Output Format of the Map List Part	172
Q&A for Using the Linker	195	List Output Format of the Symbol List Part	177
List		List Output Overview	233, 234, 235
Absolute Assemble List Creation Function	359	Local Symbol Information List	168
Absolute Assemble List Format	178	Local Symbol Information List File	188
Absolute Assemble List Output Directory		Local Symbol Information List File Name	
Specification (-alout)	150	Specification (-slf)	162
Absolute Assemble List Output Inhibit Specification		Local Symbol Information List Output Inhibit	
(-Xals)	153	Specification (-Xsl)	163
Absolute Assemble List Output Module Specification		Local Symbol Information List Output Specification	
(-alsf)	152	(-sl)	161
Absolute Assemble List Output Specification		Map List File Name Specification (-m)	100
(-als)	151	Map List Output Inhibit Specification	
Absolute Format Assemble List	168	(-Xm)	101
Assemble Source List Format	183	Memory Used Information List Part List Output	
Canceling the Omission of Names Displayed in the		Format	174
List (-dt)	102	Options for Outputting a List	211
Changing the Format of a List File	55	Options Related to the Absolute Format Assemble List	
Configuration of a List File	232	Output	91
Configuration of Link List File	169	Options Related to the Object Content	
Cross-reference List Format	186	List Output	91
Displaying Identifier Name		Options Related to the Output List	91
when Outputting List	12	Output Specification of the Memory Used Information	
Error Messages in the Assemble List	179	List (-mmi)	103
Example Display of the Memory Used Information		Relative Assemble List Input Directory Specification	
List Part List	176	(-alin)	149
External Symbol Cross-reference		Section Allocation Detailed Information	
Information List	168	List File	190
External Symbol Cross-reference Information List		Section Detail Map List	168
File	187	Section Detail Map List File Name Specification	
External Symbol Cross-reference Information List		(-mlf)	165
File Name Specification (-xlf)	159	Section Detail Map List Output Inhibit Specification	
External Symbol Cross-reference Information List		(-Xml)	166
Output Inhibit Specification		Section Detail Map List Output Specification	
(-Xxl)	160	(-ml)	164
		Section Information List Format	185

Specification of the Number of Digits in the List Line (-pw)	104	Example of Location when -AL 1 is Specified.....	68
Specification of the Number of Lines on One List Page (-pl)	97, 105, 106	Example of Location when -AL 2 is Specified.....	71
Specifying a List File Name.....	55	Example of Location when the Order of Combining Sections is not Specified	63
Specifying not to Create a Map List File (-Xm)	285, 291	Example of Location when the Order of Combining Sections is Specified	64
Specifying not to Output a List File (-Xm)	217	Example of Location when the Section Group is Specified	65
Specifying the Number of Columns Per Line of a List (-pw)	220	Section Location Attribute	59
Specifying the Number of Lines Per Page of a List (-pl)	219	Section Types and Location Destinations.....	70
Specifying to Create a Map List File (-m).....	285, 289	Setting of ROM and RAM Areas and Section Location	80
Specifying to Output a List File (-m)	216	Specifying the Order of Locating Sections and the Location Addresses	56
Specifying to Output Detailed Information of a List File (-dt)	218		
Load Module		M	
Absolute Format Load Module Output Specification (-a)	95	-m	
f2es (Converting an Absolute Format Load Module into the HEX16 Format)	267	Map List File Name Specification (-m).....	100
f2hs (Converting an Absolute Format Load Module into the HEX Format).....	265	Specifying to Create a Map List File (-m)	285, 289
f2is (Converting an Absolute Format Load Module into the HEX8 Format).....	266	Specifying to Output a List File (-m).....	216
f2ms (Converting an Absolute Format Load Module into the S Format)	264	m2es	
List of Options of the Load Module Converter	257	m2es (Converting a S Format File into the HEX16 Format)	295
Outline of Load Module Converter	256	m2is	
Output Load Module File Name Specification (-o)	92	m2is (Converting a S Format File into the HEX8 Format)	294
Relative Format Load Module Output Specification (-r)	96	Map	
Specifying an Output Load Module File Name	55	Map List File Name Specification (-m).....	100
Types of Load Module Converters	246	Map List Output Inhibit Specification (-Xm).....	101
Local Symbol Information List		Memory	
Local Symbol Information List	168	Example Display of the Memory Used Information List Part List	176
Local Symbol Information List File.....	188	Memory Used Information List Part List Output Format	174
Local Symbol Information List File Name Specification (-slf)	162	Output Specification of the Memory Used Information List (-mmi)	103
Local Symbol Information List Output Inhibit Specification (-Xsl).....	163	Message	
Local Symbol Information List Output Specification (-sl)	161	Control on Input-output Files and Messages.....	53
Locating		End Message.....	10
Automatically Locating Sections.....	66	End Message and -cmsg Option.....	10
Control on Combining and Locating Sections	54	Error Messages for the Librarian.....	319
Locating Mismatched Locations.....	358	Error Messages in the Assemble List.....	179
Locating Module Using Link Map.....	358	Error Messages of the Converter	325
Specifying the Order of Locating Sections and the Location Addresses.....	56	Error Messages of the Linker	306
Location		Help Message	11
Determining Location Addresses.....	67, 71	Linkage Kit Error Message Classes	304
		Linkage Kit Error Message Display Format	305
		Selecting whether or not to Display a Startup Message.....	55
		Selecting whether or not to Display a Termination Message	55

Specification Related to Output Messages	91	Outline of Load Module Converter	256
Specifying Display of End Message (-cmsg)	36	Output Load Module File Name Specification	
Specifying Help Message (-help)	33	(-o)	92
Specifying Suppression to Display End Message		Relative Format Load Module Output Specification	
(-Xcmsg)	37	(-r)	96
Specifying Version Number/Message Output		Replacing (Registering) a Module (-r)	213
(-V)	34	ROM/RAM and ARRAY Lists Output Module	
Startup Message	9	Specification (-alrf)	155
Startup Message and the -V Option	9	Specifying an Output Load Module	
Suppression of Version Number/Message Output		File Name	55
(-XV)	35	Types of Load Module Converters	246
Warning Message Output Level Specification		Multiple	
(-w)	113	Example of a Search when there are Multiple Library	
Mixing		Files (1)	77
Mixing of Objects for a Linker	85	Example of a Search when there are Multiple Library	
Mixing of Objects for Librarian	207	Files (2)	78
Object Mixing Check Inhibit Specification		N	
(-Xobjmixchk)	148, 230	-na	
Object Mixing Check Specification		ROM/RAM and ARRAY Lists Symbol and Address	
(-objmixchk)	147, 229	Display Position Specification	
-ml		(-na, -an)	157
Section Detail Map List Output Specification		-NCI0302LIB	
(-ml)	164	Specification for Inhibiting Check for Presence of	
-mlf		Debug Data (-NCI0302LIB)	133
Section Detail Map List File Name Specification		-nd	
(-mlf)	165	Default Library Retrieval Inhibit Specification	
-mmi		(-nd)	128
Output Specification of the Memory Used Information		-nl	
List (-mmi)	103	Library Retrieval Inhibit Specification (-nl)	127
Mode		Notes	
Overview of the Split Mode	283	Notes and Evaluation when Specified Option	24
Specifying the Inhibition of the Split Mode		Number	
(-Xsp)	285, 288	Limiting the Number of Letters	
Specifying the Split Mode (-sp)	285, 287	for Identifiers	12
Module		Number of Characters for the Filename	13
Absolute Assemble List Output Module Specification		S5 Type (Record to Manage the Number	
(-alsf)	152	of Records)	343
Absolute Format Load Module Output Specification		Specification of the Number of Digits in the List Line	
(-a)	95	(-pw)	104
Adding (Registering) a Module (-a)	212	Specification of the Number of Lines on One List Page	
Deleting a Module (-d)	214	(-pl)	97, 105, 106
Extracting a Module (-x)	215	Specifying the Number of Columns Per Line of a List	
Extracting a Module from a		(-pw)	220
Library File	201, 204	Specifying the Number of Lines Per Page of a List	
f2es (Converting an Absolute Format Load Module		(-pl)	219
into the HEX16 Format)	267	Specifying Version Number/Message Output	
f2hs (Converting an Absolute Format Load Module		(-V)	34
into the HEX Format)	265	Suppression of Version Number/Message Output	
f2is (Converting an Absolute Format Load Module		(-XV)	35
into the HEX8 Format)	266	Numeric Expression	
f2ms (Converting an Absolute Format Load Module		Numeric Expression of Option Parameters	23
into the S Format)	264		
List of Options of the Load			
Module Converter	257		
Locating Module Using Link Map	358		
Options Related to the Output Module	91		

O		Example of Specifying an Option that has a Contradictory Relation with Other Options	25
-O	Optimizing the Contents of a File (-O)		224
-o	Changing an Output File Name (-o)		251
	Output Load Module File Name Specification (-o)		92
Object			
	Common Options of an Object		
	Format Converter		250
	Executing a Command of an Object		
	Format Converter		248
	List of Commands of the Object		
	Format Converter		351
	List of Options of the Object		
	Format Converter		352
	Mixing of Objects for a Linker		85
	Mixing of Objects for Librarian		207
	Object Mixing Check Inhibit Specification (-Xobjmixchk)		148, 230
	Object Mixing Check Specification (-objmixchk)		147, 229
	Options Related to the Object Content		
	List Output		91
	Outline of Object Format Converter		244
	Questions and Answers on Using an Object Format Converter		301
	Restrictions on the Object File Format		194
	Specifying Input Object Files		55
-objmixchk	Object Mixing Check Specification (-objmixchk)		147, 229
Obtaining Addresses			
	Obtaining Addresses from S Format		357
Operation			
	Example of Operation		271
OPT			
	OPT		20
	OPT (Default Option File Storage Directory)		14
OPT896			
	OPT896		19
	OPT896 (Default Option File Storage Directory)		14
Optimizing			
	Optimizing the Contents of a File (-O)		224
Option			
	Adjust Option (-adjust)		258
	Allocation/Link Options		91
	-cmsg Option		29
	Common Options of an Object		
	Format Converter		250
	-cwno Option		29
	Default Option File		46
	End Message and -cmsg Option		10
	Example of Describing Option File		45
	Example of Specifying an Option that has a Contradictory Relation with Other Options		25
	Example of Specifying an Option that has an Inclusive Relation with Other Options		25
	Execution by Specifying Option File		42
	-f Option		29
	-help Option		29
	Library Control Option		91
	List of Common Options		28
	List of Librarian Options		350
	List of Linker Options		88, 347
	List of Options		210
	List of Options of the Binary Converter		284
	List of Options of the Format Adjuster		273
	List of Options of the Load		
	Module Converter		257
	List of Options of the Object		
	Format Converter		352
	Notes and Evaluation when Specified		
	Option		24
	Numeric Expression of Option Parameters		23
	OPT (Default Option File Storage Directory)		14
	OPT896 (Default Option File Storage Directory)		14
	Option File		42
	Options for Creating and Editing a Library		211
	Options for Outputting a List		211
	Options for Searching and Protecting a File		211
	Options Related to the Absolute Format Assemble List Output		91
	Options Related to the Object Content		
	List Output		91
	Options Related to the Output List		91
	Options Related to the Output Module		91
	Other Link Control Options		91
	Other Options		211
	Output HEX Format Option (-I16/-I20/-I32)		258, 260
	Output S Format Option (-S1/-S2/-S3)		258, 259
	Precautions on Specifying Options		238
	Specification to Continue in the Option File		43
	Specifying Comment in the Option File		44
	Specifying Reading from Option Files (-f)		31
	Specifying Suppression of Default Option File (-Xdof)		30
	Start Address Output Inhibit Option (-Xentry)		258
	Start Address Output Option (-entry)		258
	Startup Message and the -V Option		9
	Synopsis of Option		22
	-V Option		29
	-Xcmsg Option		29
	-Xcwno Option		29
	-Xdof Option		29
	-XV Option		29

Order	
Order of Searching a Library File	73
OS	
Specification Differences Depending on the OS	353
Other	
Example of Specifying an Option that has a Contradictory Relation with Other Options	25
Example of Specifying an Option that has an Inclusive Relation with Other Options	25
Other Link Control Options	91
Other Options	211
Types of Other Converters	247
Outline	
Outline of Comparison Procedure	355
Outline of Load Module Converter	256
Outline of Object Format Converter	244
Outline of the Binary Converter	282
Output	
Absolute Assemble List Output Directory Specification (-alout)	150
Absolute Assemble List Output Inhibit Specification (-Xals)	153
Absolute Assemble List Output Module Specification (-alsf)	152
Absolute Assemble List Output Specification (-als)	151
Absolute Format Load Module Output Specification (-a)	95
Changing an Output File Name (-o)	251
Debug Information Output Specification (-g)	93
Displaying Identifier Name when Outputting List	12
External Symbol Cross-reference Information List Output Inhibit Specification (-Xxl)	160
External Symbol Cross-reference Information List Output Specification (-xl)	158
List Output Format of the Control List Part	170
List Output Format of the Map List Part	172
List Output Format of the Symbol List Part	177
List Output Overview	233, 234, 235
Local Symbol Information List Output Inhibit Specification (-Xsl)	163
Local Symbol Information List Output Specification (-sl)	161
Map List Output Inhibit Specification (-Xm)	101
Memory Used Information List Part List Output Format	174
Options for Outputting a List	211
Options Related to the Absolute Format Assemble List Output	91
Options Related to the Object Content List Output	91
Options Related to the Output List	91
Options Related to the Output Module	91
Output HEX Format Option (-I16/-I20/-I32)	258, 260
Output Load Module File Name Specification (-o)	92
Output S Format Option (-S1/-S2/-S3)	258, 259
Output Specification of the Memory Used Information List (-mmi)	103
Relative Format Load Module Output Specification (-r)	96
ROM/RAM and ARRAY Lists Output Inhibit Specification (-Xalr)	156
ROM/RAM and ARRAY Lists Output Module Specification (-alrf)	155
ROM/RAM and ARRAY Lists Output Specification (-alr)	154
Section Detail Map List Output Inhibit Specification (-Xml)	166
Section Detail Map List Output Specification (-ml)	164
Specification Related to Output Messages	91
Specifying an Output HEX Format (-I16/-I20/-I32)	274, 279
Specifying an Output Load Module File Name	55
Specifying an Output S Format (-S1/-S2/-S3)	274, 277
Specifying not to Output a List File (-Xm)	217
Specifying not to Output Debugging Information (-Xg)	226
Specifying not to Output Start Address Record (-Xentry)	262
Specifying the Data Length in an Output Record (-len)	275
Specifying the Output Format	55
Specifying the Output Range (-ran)	274, 276, 285, 286
Specifying the Output Record Data Length (-len)	274
Specifying to Output a List File (-m)	216
Specifying to Output Debugging Information (-g)	225
Specifying to Output Detailed Information of a List File (-dt)	218
Specifying to Output Start Address Record (-entry)	261
Specifying Version Number/Message Output (-V)	34
Start Address Output Inhibit Option (-Xentry)	258
Start Address Output Option (-entry)	258
Suppression of Version Number/Message Output (-XV)	35
Warning Message Output Level Specification (-w)	113

Overview

List Output Overview.....	233, 234, 235
Overview of a Linker	52
Overview of the Format Adjuster	270
Overview of the Split Mode.....	283

P

-p

Padding (-p)	253
--------------------	-----

Padding

Padding (-p)	253
--------------------	-----

Parameter

Parameter.....	22
----------------	----

-pl

Specification of the Number of Lines on One List Page (-pl).....	97, 105, 106
Specifying the Number of Lines Per Page of a List (-pl).....	219

Precautions

Precautions on Specifying Options	238
Precautions on the Required Disk Space	238
Precautions on the Sections to be Transferred from ROM to RAM	82

Preface

Preface	359
---------------	-----

Procedure

Outline of Comparison Procedure.....	355
--------------------------------------	-----

Processing

Processing when Library Files are Individually Specified.....	79
---	----

Protecting

Options for Searching and Protecting a File.....	211
--	-----

-pw

Specification of the Number of Digits in the List Line (-pw)	104
Specifying the Number of Columns Per Line of a List (-pw)	220

Q

Q&A

Q&A for Using the Linker.....	195
-------------------------------	-----

Questions and Answers

Questions and Answers on Using an Object Format Converter.....	301
--	-----

R

-r

Relative Format Load Module Output Specification (-r)	96
Replacing (Registering) a Module (-r)	213

-ra

RAM Area Specification (-ra).....	115
-----------------------------------	-----

RAM

Function that Automatically Sets Internal ROM/RAM Area (-set_roma).....	134
Precautions on the Sections to be Transferred from ROM to RAM.....	82
RAM Area Specification (-ra).....	115
ROM/RAM and ARRAY Lists	181
ROM/RAM and ARRAY Lists Output Inhibit Specification (-Xalr)	156
ROM/RAM and ARRAY Lists Output Module Specification (-alrf)	155
ROM/RAM and ARRAY Lists Output Specification (-alr).....	154
ROM/RAM and ARRAY Lists Symbol and Address Display Position Specification (-na, -an).....	157
ROM/RAM Areas Names	84
Sections to be Transferred from ROM to RAM	81
Setting of ROM and RAM Areas and Section Location	80
Specifies to Prevent the Internal ROM/RAM Area from being Automatically Set (-Xset_roma)	135
Specifies to Prevent the Internal ROM/RAM Area from being Set Automatically	84
Specifying ROM and RAM Areas.....	56
Using the Sections to be Transferred from ROM to RAM.....	81

-ran

Specifying the Output Range (-ran).....	274, 276, 285, 286
---	--------------------

Range

Specifying the Output Range (-ran).....	274, 276, 285, 286
Support Range of Linkage Kit	4

Reading

Specifying Reading from Option Files (-f)	31
---	----

Record

Data Record (HEX8/HEX16/HEX32)	332
End Record (HEX8/HEX16/HEX32)	333
Extended Linear Address Record (HEX32).....	336
Extended Segment Address Record (HEX16/HEX32).....	334
S Record Format	338
S0 Type (Header Record).....	339
S1 Type (Data Record: 2-byte Address).....	340
S2 Type (Data Record: 3-byte Address).....	341
S3 Type (Data Record: 4-byte Address).....	342
S5 Type (Record to Manage the Number of Records)	343
S7 Type (Terminator Record)	344
S8 Type (Terminator Record)	345
S9 Type (Terminator Record)	346
Specifying not to Output Start Address Record (-Xentry)	262

Specifying the Data Length in an Output Record (-len).....	275
Specifying the Output Record Data Length (-len).....	274
Specifying to Output Start Address Record (-entry).....	261
Start Linear Address Record (HEX32)	337
Start Segment Address Record (HEX16/HEX32)	335
Register Bank	
Register Bank Area Specification (-rg).....	120
Registering	
Adding (Registering) a Module (-a)	212
Replacing (Registering) a Module (-r).....	213
Relation	
Example of Specifying an Option that has a Contradictory Relation with Other Options	25
Example of Specifying an Option that has an Inclusive Relation with Other Options	25
Relative	
Relative Assemble List Input Directory Specification (-alin).....	149
Relative Format Load Module Output Specification (-r).....	96
Replacing	
Replacing (Registering) a Module (-r).....	213
Restrictions	
Alleviation of Restrictions	363
Linker Restrictions	194
Restrictions on a Librarian	238
Restrictions on an Object Format Converter	300
Restrictions on the Object File Format	194
Retrieval Library File	
Retrieval Library File Specification (-l).....	124
-rg	
Register Bank Area Specification (-rg).....	120
-ro	
ROM Area Specification (-ro)	114
Roles	
Roles of Librarian	200
ROM	
Function that Automatically Sets Internal ROM/RAM Area (-set_rora).....	134
Precautions on the Sections to be Transferred from ROM to RAM.....	82
ROM Area Specification (-ro)	114
ROM/RAM and ARRAY Lists.....	181
ROM/RAM and ARRAY Lists Output Inhibit Specification (-Xalr).....	156
ROM/RAM and ARRAY Lists Output Module Specification (-alrf)	155
ROM/RAM and ARRAY Lists Output Specification (-alr)	154

ROM/RAM and ARRAY Lists Symbol and Address Display Position Specification (-na, -an)	157
ROM/RAM Areas Names.....	84
Sections to be Transferred from ROM to RAM	81
Setting of ROM and RAM Areas and Section Location.....	80
Specifies to Prevent the Internal ROM/RAM Area from being Automatically Set (-Xset_rora).....	135
Specifies to Prevent the Internal ROM/RAM Area from being Set Automatically	84
Specifying ROM and RAM Areas	56
Support for Creating a ROM.....	56
Using the Sections to be Transferred from ROM to RAM	81

S

S Format

e2ms (Converting a HEX16 Format File into the S Format).....	297
f2ms (Converting an Absolute Format Load Module into the S Format)	264
i2ms (Converting a HEX8 Format File into the S Format).....	296
m2es (Converting a S Format File into the HEX16 Format).....	295
m2is (Converting a S Format File into the HEX8 Format).....	294
Obtaining Addresses from S Format	357
Output S Format Option (-S1/-S2/-S3)	258, 259
Specifying an Output S Format (-S1/-S2/-S3)	274, 277

S Record

S Record Format.....	338
----------------------	-----

S0 Type

S0 Type (Header Record)	339
-------------------------------	-----

S1 Type

S1 Type (Data Record: 2-byte Address)	340
---	-----

-S1/-S2/-S3

Output S Format Option (-S1/-S2/-S3)	258, 259
Specifying an Output S Format (-S1/-S2/-S3)	274, 277

S2 Type

S2 Type (Data Record: 3-byte Address)	341
---	-----

S3 Type

S3 Type (Data Record: 4-byte Address)	342
---	-----

S5 Type

S5 Type (Record to Manage the Number of Records)	343
---	-----

S7 Type

S7 Type (Terminator Record)	344
-----------------------------------	-----

S8 Type	Section Identification.....	60
S8 Type (Terminator Record)	Section Information List Format	185
S9 Type	Section Location Attribute	59
S9 Type (Terminator Record)	Section Name.....	59
-sc	Section Types and Location Destinations.....	70
Section Allocation Order/Address Specification	Sections to be Transferred	
(-sc).....	from ROM to RAM	81
Search	Setting of ROM and RAM Areas	
Control on Searching Libraries.....	and Section Location	80
Example of a Search when there are Multiple Library	Shared Combination of Sections	61
Files (1)	Simple Connection Combination of	
Example of a Search when there are Multiple Library	Sections.....	61
Files (2)	Specifying Check User-unspecified Section	
Example of a Search when there is One Library File	(-check_section)	144
(1)	Specifying Inhibit Check for Presence of User-	
Example of a Search when there is One Library File	unspecified Section	
(2)	(-Xcheck_section)	146
Example of a Search when there is One Library File	Specifying the Order of Locating Sections and the	
(3)	Location Addresses	56
Inhibiting the Search for a Library.....	Types of Section Contents	59
LIB896 (Library File Search Directory)	Using the Sections to be Transferred from ROM to	
Options for Searching and Protecting	RAM.....	81
a File	Section-placed-area Check	
Order of Searching a Library File	Section-placed-area Check Specification	
Specifying a Library File to be Searched.....	(-check_locate).....	139
Specifying a Library File to be Searched for Each	Section-placed-area Check Suppression Specification	
Symbol	(-Xcheck_locate)	143
Specifying a Library to be Searched	Selecting	
Specifying a Path to Search a Library	Selecting the Warning Check Level	55
Section	Selecting whether or not to Display a Startup	
Automatically Locating Sections.....	Message	55
Control on Combining and Locating	Selecting whether or not to Display a Termination	
Sections	Message	55
Creating a Group of Sections	-set_rora	
Example of Location when the Order of Combining	Function that Automatically Sets Internal ROM/RAM	
Sections is not Specified.....	Area (-set_rora).....	134
Example of Location when the Order of Combining	Setting	
Sections is Specified	Dummy Setting of External Symbol Values	
Example of Location when the Section Group is	(-df)	130
Specified.....	Setting an External Symbol Value.....	58
Links of Sections	Setting Entry Addresses and Symbol Values	54
Precautions on the Sections to be Transferred from	Setting of ROM and RAM Areas	
ROM to RAM	and Section Location	80
Section Allocation Detailed Information	Shared Combination	
List File	Shared Combination of Sections	61
Section Allocation Order/Address Specification	Simple Connection Combination	
(-sc).....	Simple Connection Combination of Sections	61
Section Combination Attribute.....	-sl	
Section Detail Map List.....	Local Symbol Information List Output Specification	
Section Detail Map List File Name Specification	(-sl)	161
(-mlf).....	-slf	
Section Detail Map List Output Inhibit Specification	Local Symbol Information List File Name	
(-Xml)	Specification (-slf).....	162
Section Detail Map List Output Specification	Source	
(-ml)	Assemble Source List Format	183
Section Group Specification (-gr)		

-sp		
Specifying the Split Mode (-sp)	285, 287	
Specification		
Absolute Assemble List Output Directory		
Specification (-alout)	150	
Absolute Assemble List Output Inhibit Specification		
(-Xals)	153	
Absolute Assemble List Output Module Specification		
(-alsf)	152	
Absolute Assemble List Output Specification		
(-als)	151	
Absolute Format Load Module Output Specification		
(-a)	95	
Automatic Allocation Specification		
(-AL)	122	
CPU Information File Specification		
(-cif)	132, 228	
Debug Information Delete Specification		
(-Xg)	94	
Debug Information Output Specification		
(-g)	93	
Default Library Retrieval Inhibit Specification		
(-nd)	128	
Entry Address Specification (-e)	129	
External Symbol Cross-reference Information List		
File Name Specification (-xlf)	159	
External Symbol Cross-reference Information List		
Output Inhibit Specification		
(-Xxl)	160	
External Symbol Cross-reference Information List		
Output Specification (-xl)	158	
Library Retrieval Inhibit Specification		
(-nl)	127	
Library Retrieval Path Specification (-L)	125	
Library Specification for Each Symbol		
(-el)	126	
Local Symbol Information List File Name		
Specification (-slf)	162	
Local Symbol Information List Output Inhibit		
Specification (-Xsl)	163	
Local Symbol Information List Output Specification		
(-sl)	161	
Map List File Name Specification (-m)	100	
Map List Output Inhibit Specification		
(-Xm)	101	
Object Mixing Check Inhibit Specification		
(-Xobjmixchk)	148, 230	
Object Mixing Check Specification		
(-objmixchk)	147, 229	
Output Load Module File Name Specification		
(-o)	92	
Output Specification of the Memory Used Information		
List (-mmi)	103	
RAM Area Specification (-ra)	115	
Register Bank Area Specification (-rg)	120	
Relative Assemble List Input Directory Specification		
(-alin)	149	
Relative Format Load Module Output Specification		
(-r)	96	
Retrieval Library File Specification (-l)	124	
ROM Area Specification (-ro)	114	
ROM/RAM and ARRAY Lists Output Inhibit		
Specification (-Xalr)	156	
ROM/RAM and ARRAY Lists Output Module		
Specification (-alrf)	155	
ROM/RAM and ARRAY Lists Output Specification		
(-alr)	154	
ROM/RAM and ARRAY Lists Symbol and Address		
Display Position Specification		
(-na, -an)	157	
Section Allocation Order/Address Specification		
(-sc)	116	
Section Detail Map List File Name Specification		
(-mlf)	165	
Section Detail Map List Output Inhibit Specification		
(-Xml)	166	
Section Detail Map List Output Specification		
(-ml)	164	
Section Group Specification (-gr)	119	
Section-placed-area Check Specification		
(-check_locate)	139	
Section-placed-area Check Suppression Specification		
(-Xcheck_locate)	143	
Specification Differences Depending		
on the OS	353	
Specification for Inhibiting Check for Presence of		
Debug Data (-NCI0302LIB)	133	
Specification of the Number of Digits in the List Line		
(-pw)	104	
Specification of the Number of Lines on One List Page		
(-pl)	105	
Specification Related to Output Messages	91	
Specification to Continue in the Option File	43	
Target CPU Specification (-cpu)	131	
User-specified-area Check Specification		
(-check_rora)	136	
User-specified-area Check Suppression Specification		
(-Xcheck_rora)	138	
Warning Message Output Level Specification		
(-w)	113	
Specify		
Example of Specifying an Option that has a		
Contradictory Relation		
with Other Options	25	
Example of Specifying an Option that has an Inclusive		
Relation with Other Options	25	
Example of Specifying Command Lines	26	
Execution by Specifying Option File	42	
Precautions on Specifying Options	238	
Specifies to Prevent the Internal ROM/RAM Area		
from being Automatically Set		
(-Xset_rora)	135	
Specifies to Prevent the Internal ROM/RAM Area		
from being Set Automatically	84	

Specifying a Library File to be Searched.....	57	Specifying the Output Record Data Length	
Specifying a Library File to be Searched for Each		(-len).....	274
Symbol	57	Specifying the Split Mode (-sp)	285, 287
Specifying a Library to be Searched	73	Specifying to Adjust (-adjust)	263
Specifying a List File Name.....	55	Specifying to Change the Starting Address of the	
Specifying a Path to Search a Library	57	Record (-ST).....	274
Specifying a Target CPU (-cpu)	227	Specifying to Create a Map List File	
Specifying an Entry Address.....	58	(-m)	285, 289
Specifying an Output HEX Format		Specifying to Output a List File (-m).....	216
(-I16/-I20/-I32).....	274, 279	Specifying to Output Debugging Information	
Specifying an Output Load Module		(-g)	225
File Name	55	Specifying to Output Detailed Information of a List	
Specifying an Output S Format		File (-dt)	218
(-S1/-S2/-S3).....	274, 277	Specifying to Output Start Address Record	
Specifying Changes to the Starting Address		(-entry)	261
(-ST)	280	Specifying Version Number/Message Output	
Specifying Check User-unspecified Section		(-V).....	34
(-check_section)	144	Split	
Specifying Comment in the Option File	44	Overview of the Split Mode	283
Specifying Display of End Message		Specifying the Inhibition of the Split Mode	
(-cmsg)	36	(-Xsp).....	285, 288
Specifying End Code to 0 when Warning is Issued		Specifying the Split Mode (-sp)	285, 287
(-Xcwno)	39	-ST	
Specifying End Code to 1 when Warning is Issued		Specifying Changes to the Starting Address	
(-cwno)	38	(-ST)	280
Specifying Help Message (-help).....	33	Specifying to Change the Starting Address of the	
Specifying Inhibit Check for Presence of User-		Record (-ST).....	274
unspecified Section		Start Address	
(-Xcheck_section).....	146	Specifying Changes to the Starting Address	
Specifying Input Object Files.....	55	(-ST)	280
Specifying not to Create a Map List File		Specifying not to Output Start Address Record	
(-Xm)	285, 291	(-Xentry)	262
Specifying not to Output a List File		Specifying to Change the Starting Address of the	
(-Xm)	217	Record (-ST).....	274
Specifying not to Output Debugging Information		Specifying to Output Start Address Record	
(-Xg)	226	(-entry)	261
Specifying not to Output Start Address Record		Start Address Output Inhibit Option	
(-Xentry).....	262	(-Xentry)	258
Specifying Reading from Option Files (-f)	31	Start Address Output Option (-entry).....	258
Specifying ROM and RAM Areas	56	Start Linear Address Record	
Specifying Suppression of Default Option File		Start Linear Address Record (HEX32).....	337
(-Xdof)	30	Start Segment Address Record	
Specifying Suppression to Display End Message		Start Segment Address Record	
(-Xcmsg)	37	(HEX16/HEX32).....	335
Specifying the Data Length in an Output Record		Startup Message	
(-len)	275	Selecting whether or not to Display	
Specifying the Inhibition of the Split Mode		a Startup Message.....	55
(-Xsp).....	285, 288	Startup Message	9
Specifying the Number of Columns Per Line of a List		Startup Message and the -V Option	9
(-pw)	220	Status	
Specifying the Number of Lines Per Page of a List		End Code Value and End Status.....	8
(-pl).....	219	Support	
Specifying the Order of Locating Sections and the		Support for Creating a ROM	56
Location Addresses.....	56	Support Range of Linkage Kit	4
Specifying the Output Format	55		
Specifying the Output Range			
(-ran)	274, 276, 285, 286		

Suppression	S8 Type (Terminator Record)	345
Section-placed-area Check Suppression Specification	S9 Type (Terminator Record)	346
(-Xcheck_locate).....	TMP	
143	TMP (Work Directory).....	14, 15
Specifying Suppression of Default Option File	Types	
(-Xdof).....	Section Types and Location Destinations	70
30	Types of Characters Consisting of	
Specifying Suppression to Display End Message	Identifiers.....	12
(-Xcmmsg).....	Types of Load Module Converters	246
37	Types of Other Converters	247
Suppression of Version Number/Message Output	Types of Section Contents	59
(-XV).....		
35		
User-specified-area Check Suppression Specification		
(-Xcheck_rora)		
138		
Symbol		
Dummy Setting of External Symbol Values	U	
(-df).....	User-specified-area Check	
130	User-specified-area Check Specification	
External Symbol Cross-reference	(-check_rora).....	136
Information List	User-specified-area Check Suppression Specification	
168	(-Xcheck_rora)	138
External Symbol Cross-reference Information List		
File	Using	
187	Locating Module Using Link Map.....	358
External Symbol Cross-reference Information List	Q&A for Using the Linker	195
File Name Specification (-xlf)	Questions and Answers on Using an Object Format	
159	Converter	301
External Symbol Cross-reference Information List	Using the Sections to be Transferred from ROM to	
Output Inhibit Specification	RAM	81
(-Xxl).....		
160		
External Symbol Cross-reference Information List	V	
Output Specification (-xl)	-V	
158	Specifying Version Number/Message Output	
Library Specification for Each Symbol	(-V)	34
(-el)	Startup Message and the -V Option	9
126	-V Option.....	29
Linker Reservation Symbol	Variables	
194	Environmental Variables	360, 364
List Display Example of the Symbol	Handling Variables with Initial Values	196
List Part		
177	W	
List Output Format of the Symbol	-w	
List Part	Warning Message Output Level Specification	
177	(-w)	113
Local Symbol Information List.....	Warning	
168	Selecting the Warning Check Level	55
Local Symbol Information List File	Specifying End Code to 0 when Warning is Issued	
188	(-Xcwno)	39
Local Symbol Information List File Name	Specifying End Code to 1 when Warning is Issued	
Specification (-slf).....	(-cwno).....	38
162	Warning Message Output Level Specification	
Local Symbol Information List Output Inhibit	(-w)	113
Specification (-Xsl)		
163	When Data Mismatched	
Local Symbol Information List Output Specification	When Data Mismatched	357
(-sl).....		
161		
ROM/RAM and ARRAY Lists Symbol and Address		
Display Position Specification		
(-na, -an)		
157		
Setting an External Symbol Value		
58		
Setting Entry Addresses		
and Symbol Values		
54		
Specifying a Library File to be Searched for Each		
Symbol		
57		
Synopsis		
Synopsis of Option.....		
22		
T		
Target		
Target CPU Specification (-cpu).....		
131		
Terminator		
S7 Type (Terminator Record).....		
344		

X

-x	Extracting a Module (-x)	215
-Xalr	ROM/RAM and ARRAY Lists Output Inhibit Specification (-Xalr)	156
-Xals	Absolute Assemble List Output Inhibit Specification (-Xals)	153
-Xb	Inhibiting the Creation of a Backup File (-Xb)	222
-Xcheck_locate	Section-placed-area Check Suppression Specification (-Xcheck_locate)	143
-Xcheck_rora	User-specified-area Check Suppression Specification (-Xcheck_rora)	138
-Xcheck_section	Specifying Inhibit Check for Presence of User-unspecified Section (-Xcheck_section)	146
-Xcmsg	Specifying Suppression to Display End Message (-Xcmsg)	37
	-Xcmsg Option	29
-Xcwno	Specifying End Code to 0 when Warning is Issued (-Xcwno)	39
	-Xcwno Option	29
-Xdof	Specifying Suppression of Default Option File (-Xdof)	30
	-Xdof Option	29
-Xentry	Specifying not to Output Start Address Record (-Xentry)	262
	Start Address Output Inhibit Option (-Xentry)	258
-Xg	Debug Information Delete Specification (-Xg)	94
	Specifying not to Output Debugging Information (-Xg)	226
-xl	External Symbol Cross-reference Information List Output Specification (-xl)	158

-xlf	External Symbol Cross-reference Information List File Name Specification (-xlf)	159
-Xm	Map List Output Inhibit Specification (-Xm)	101
	Specifying not to Create a Map List File (-Xm)	285, 291
	Specifying not to Output a List File (-Xm)	217
-Xml	Section Detail Map List Output Inhibit Specification (-Xml)	166
-Xobjmixchk	Object Mixing Check Inhibit Specification (-Xobjmixchk)	148, 230
-Xset_rora	Specifies to Prevent the Internal ROM/RAM Area from being Automatically Set (-Xset_rora)	135
-Xsl	Local Symbol Information List Output Inhibit Specification (-Xsl)	163
-Xsp	Specifying the Inhibition of the Split Mode (-Xsp)	285, 288
-XV	Suppression of Version Number/Message Output (-XV)	35
	-XV Option	29
-Xxl	External Symbol Cross-reference Information List Output Inhibit Specification (-Xxl)	160

CM25-00321-5E

SpanSion • SOFTWARE SUPPORT MANUAL

F²MC-8L/8FX FAMILY

SOFTUNE™ LINKAGE KIT MANUAL

for V3

September 2014 Rev. 5.0

Published **SpanSion Inc.**
Edited **Communications**

Colophon

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for any use that includes fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for any use where chance of failure is intolerable (i.e., submersible repeater and artificial satellite). Please note that Spansion will not be liable to you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the US Export Administration Regulations or the applicable laws of any other country, the prior authorization by the respective government entity will be required for export of those products.

Trademarks and Notice

The contents of this document are subject to change without notice. This document may contain information on a Spansion product under development by Spansion. Spansion reserves the right to change or discontinue work on any product without notice. The information in this document is provided as is without warranty or guarantee of any kind as to its accuracy, completeness, operability, fitness for particular purpose, merchantability, non-infringement of third-party rights, or any other warranty, express, implied, or statutory. Spansion assumes no liability for any damages of any kind arising out of the use of the information in this document.

Copyright © 2004 - 2014 Spansion All rights reserved. Spansion®, the Spansion logo, MirrorBit®, MirrorBit® Eclipse™, ORNAND™ and combinations thereof, are trademarks and registered trademarks of Spansion LLC in the United States and other countries. Other names used are for informational purposes only and may be trademarks of their respective owners.