PREFACE

■ Objectives and Intended Readership

This manual describes the functions and operations of the Fujitsu Softune Linkage Kit operating on Windows 95/98/Me, Windows NT and Windows 2000.

This manual is intended for engineers who are developing application programs using the FR family/F²MC-16 family/F²MC-8L family microcontroller.

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

■ Trademarks

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

This manual consists of the following two parts:

PART 1  LINKAGE KIT

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

PART 2  LINKER

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

PART 3  LIBRARIAN

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

PART 4  OBJECT FORMAT CONVERTERS

Explains the specifications, options, and output lists of an object format converters.

Appendices

Explains the error messages of the linkage kit, Intel HEX, Intel extended HEX format, and Motorola S record format.
The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.

The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended to be incorporated in devices for actual use. Also, FUJITSU is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.

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 use accompanying 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 use requiring extremely high reliability (i.e., submersible repeater and artificial satellite). Please note that Fujitsu will not be liable against 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 prior authorization by Japanese government will be required for export of those products from Japan.
READING THIS MANUAL

■ Product name abbreviation

In this manual, product names are abbreviated as follows:

Microsoft® Windows® 95 operating system, Microsoft® Windows® 98 operating system and Microsoft® Windows® Millennium Edition operating system are abbreviated Windows 95/98/Me.

Microsoft® Windows NT® Server network operating system Version 4.0 and Microsoft® Windows NT® Workstation operating system Version 4.0 are abbreviated Windows NT.

Microsoft® Windows ® 2000 Professional operating system is abbreviated Windows 2000.
## CONTENTS

### PART 1  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 LIB911, LIB907, LIB896 .........................................................................................................18
    1.10.5 OPT911, OPT907, OPT896 .....................................................................................................19
    1.10.6 OPT .......................................................................................................................................20

### CHAPTER 2  OPTIONS..........................................................................................................................21
  2.11 Option ............................................................................................................................................22
  2.12 Numeric Expression of Option Parameters ....................................................................................23
  2.13 Notes and Evaluation When Option Is Specified ........................................................................24
  2.14 Specifying Options that Have Inclusive or Contradictory Relation Each Other ..........................25
  2.15 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) ...............................................................................................32
    3.2.4 Specifying Version Number/Message Output (-V) ....................................................................35
    3.2.5 Suppression of Version Number/Message Output (-XV) ............................................................36
    3.2.6 Specifying Display of End Message (-cmsg) .............................................................................37
    3.2.7 Specifying Suppression to Display End Message (-Xcmsg) .......................................................38
    3.2.8 Specifying End Code to 1 When Warning is Issued (-cwno) .....................................................39
    3.2.9 Specifying End Code to 0 When Warning is Issued (-Xcwno) ...................................................40

### CHAPTER 4  OPTION FILES................................................................................................................41
  4.1 Option File .......................................................................................................................................42
  4.2 Specification to Continue in the Option File ..................................................................................43
  4.3 Specifying Comment in the Option File .........................................................................................44
PART 2  LINKER .................................................................................................................. 49

CHAPTER 5  SPECIFICATIONS OF A LINKER ..................................................................... 51
5.1 Overview of a Linker ...................................................................................................... 52
5.2 Functions of a Linker ................................................................................................... 54
5.2.1 Control on Input-Output Files and Messages ........................................................... 56
5.2.2 Control on Combining and Locating Sections ......................................................... 58
5.2.3 Control on Searching Libraries ............................................................................... 59
5.2.4 Setting Entry Addresses and Symbol Values ......................................................... 60
5.3 Types of Sections ....................................................................................................... 62
5.4 Combining Sections ................................................................................................... 64
5.5 Locating Sections ........................................................................................................ 65
5.5.1 Example of Location when the Order of Combining Sections is not Specified ........... 66
5.5.2 Example of Location when the Order of Combining Sections is Specified ............... 67
5.5.3 Example of Location when the Section Group is Specified ....................................... 68
5.6 Automatically Locating Sections .................................................................................. 69
5.6.1 Automatically Locating Sections when -AL 1 is Specified ........................................... 70
5.6.2 Automatically Locating Sections when -AL 2 is Specified ........................................... 72
5.7 Searching Libraries .................................................................................................... 74
5.7.1 Example of a Search when there is one Library File (1) ............................................. 75
5.7.2 Example of a Search when there is one Library File (2) ............................................. 76
5.7.3 Example of a Search when there is one Library File (3) ............................................. 77
5.7.4 Example of a Search when there are Multiple Library Files (1) ................................. 78
5.7.5 Example of a Search when there are Multiple Library Files (2) ................................. 79
5.7.6 Processing when Library Files are Individually Specified ......................................... 80
5.8 ROM and RAM Areas ................................................................................................. 81
5.9 Sections to be Transferred from ROM to RAM .......................................................... 82
5.10 CPU Information File ................................................................................................ 85
5.11 Mixing of Objects for a F2MC-16 Family Linker ........................................................ 86

CHAPTER 6  LINKER OPTIONS ......................................................................................... 87
6.1 List of Linker Options ................................................................................................ 88
6.2 Details of Linker Options .......................................................................................... 90
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 Map List File Name Specification (-m) ....................................................................... 97
6.2.7 Map List Output Inhibit Specification (-Xm) .............................................................. 98
6.2.8 Canceling the Omission of Names Displayed in the List (-dt) ...................................... 99
6.2.9 Output Specification of the Map List File Memory Used Information List (-mmi) ......... 100
6.2.10 Specification of the Number of Digits in the List Line (-pw) ..................................... 101
6.2.11 Specification of the Number of Lines on one List Page (-pl) ................................... 102
6.2.12 Warning Message Output Level Specification (-w) ..............................................................103
6.2.13 ROM Area Specification (-ro) .............................................................................................104
6.2.14 RAM Area Specification (-ra) .............................................................................................105
6.2.15 Section Allocation Order/Address Specification (-sc) ........................................................106
6.2.16 Section Group Specification (-gr) ........................................................................................109
6.2.17 Pack Link Specification (-pk) .............................................................................................110
6.2.18 Register Bank Area Specification (-rg) ..............................................................................112
6.2.19 Automatic Allocation Specification (-AL) ............................................................................114
6.2.20 Retrieval Library File Specification (-l) ..............................................................................116
6.2.21 Library Retrieval Path Specification (-L) ............................................................................117
6.2.22 Library Specification for Each Symbol (-el) ......................................................................118
6.2.23 Library Retrieval Inhibit Specification (-nl) ......................................................................119
6.2.24 Default Library Retrieval Inhibit Specification (-nd) ..........................................................120
6.2.25 Entry Address Specification (-e) .......................................................................................121
6.2.26 Dummy Setting of External Symbol Values (-df) ...............................................................122
6.2.27 Target CPU Specification (-cpu) .......................................................................................123
6.2.28 Inhibiting Check for Presence of Debug Data (-NCI0302LIB) ..........................................124
6.2.29 Function that sets automatically internal ROM/RAM area (-set_rora) ..............................125
6.2.30 Specifies to prevent the internal ROM/RAM area from being set automatically (-Xset_rora)126
6.2.31 User-specified-area check specification (-check_rora) .......................................................127
6.2.32 User-specified-area check suppression specification (-Xcheck_rora) ...............................129
6.2.33 Section-placed-area check specification (-check_locate) ...................................................130
6.2.34 Section-placed-area check suppression specification (-Xcheck_locate) .................................134
6.2.35 Check User-unspecified Section (-check_section) .............................................................135
6.2.36 Inhibit check for presence of user-unspecified section (-Xcheck_section) ......................137
6.2.37 Relative Assemble List Input Directory Specification (-alin) ............................................138
6.2.38 Absolute Assemble List Output Directory Specification (-alout) .........................................139
6.2.39 Absolute Assemble List Output Specification (-als) ............................................................140
6.2.40 Absolute Assemble List Output Module Specification (-alsf) ............................................141
6.2.41 Absolute Assemble List Output Inhibit Specification (-Xals) .............................................142
6.2.42 ROM/RAM and ARRAY List Output Specification (-alr) .....................................................143
6.2.43 ROM/RAM and ARRAY List Output Module Specification (-alrf) ........................................144
6.2.44 ROM/RAM and ARRAY List Output Inhibit Specification (-Xalr) .........................................145
6.2.45 ROM/RAM and ARRAY List Symbol and Address Display Position Specification (-na,-an)146
6.2.46 External Symbol Cross-reference Information List Output Specification (-xl) ....................147
6.2.47 External Symbol Cross-reference Information List File Name Specification (-xlf) ...............148
6.2.48 External Symbol Cross-reference Information List Output Inhibit Specification (-Xxl) .........149
6.2.49 Local Symbol Information List Output Specification (-sl) ..................................................150
6.2.50 Local Symbol Information List File Name Specification (-slf) ..............................................151
6.2.51 Local Symbol Information List Output Inhibit Specification (-Xsl) .........................................152
6.2.52 Section Detail Map List Output Specification (-ml) ..............................................................153
6.2.53 Section Detail Map List Name Specification (-mlf) ...............................................................154
6.2.54 Section Detail Map List Output Inhibit Specification (-Xml) ................................................155

CHAPTER 7  OUTPUT LIST FILE OF THE LINKER .............................................................................157
7.1 Types of List Files Output by the Linker .................................................................................158
7.2 Link List File .........................................................................................................................159
7.2.1 Control List .......................................................................................................................160
CHAPTER 9 SPECIFICATIONS OF A LIBRARIAN ........................................................ 197

9.1 Functions of a Librarian .......................................................................................... 199
9.2 Function Types of a Librarian ................................................................................. 200
9.3 Creating and Editing a Library File ......................................................................... 201
9.4 Extracting a Module from a Library File ............................................................... 203
9.5 Deleting Debugging Information of a Library ....................................................... 204
9.6 Checking and Displaying the Contents of a Library File ....................................... 205
9.7 Mixing of Objects for an F2MC-16 Family Librarian ........................................... 206

CHAPTER 10 OPTIONS A LIBRARIAN ........................................................................ 207
10.1 List of Options of a Librarian ................................................................................ 208
10.2 Details of the Options of a Librarian ..................................................................... 209
10.2.1 Adding (Registering) a Module (-a) ................................................................. 210
10.2.2 Replacing (Registering) a Module (-r) ............................................................. 211
10.2.3 Deleting a Module (-d) .................................................................................... 212
10.2.4 Extracting a Module (-x) ................................................................................ 213
10.2.5 Specifying to Output a List File (-m) ................................................................. 214
10.2.6 Specifying not to Output a List File (-Xm) ....................................................... 215
10.2.7 Specifying the Number of Lines Per Page of a List (-pl) ................................. 217
10.2.8 Specifying the Number of Columns Per Line of a List (-pw) ......................... 218
10.2.9 Creating a Backup File (-b) ............................................................................. 219
10.2.10 Creating a Backup File (-Xb) .......................................................................... 220
10.2.11 Inhibiting the Creation of a Backup File (-Xb) .............................................. 221
10.2.12 Optimizing the Contents of a File (-O) ........................................................... 222
10.2.13 Specifying to Output Detailed Information of a List File (-dt) ....................... 216
10.2.14 Specifying to Output Debugging Information (-g) ......................................... 223
10.2.15 Specifying not to Output Debugging Information (-Xg) ............................... 224
10.2.16 Specifying a Target CPU (-cpu) ................................................................. 225
CHAPTER 11 LIST FORMATS OF A LIBRARIAN ............................................................ 227
11.1 Contents of Information in a List File ................................................................. 228
11.2 List of Module Names .......................................................................................... 229
11.3 Detailed Information of a Module......................................................................... 230
11.4 External Defined and Reference Symbol Information in a Library......................... 231

CHAPTER 12 RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN 233
12.1 Restrictions on a Librarian ................................................................................... 234
12.2 Questions and Answers on Using a Librarian ....................................................... 235

PART 4 OBJECT FORMAT CONVERTERS ................................................................. 237

CHAPTER 13 SPECIFICATIONS OF AN OBJECT FORMAT CONVERTER .............. 239
13.1 Overview of an Object Format Converter ............................................................ 240
13.2 Types of Converters ............................................................................................ 241
13.3 Executing an Object Format Converter ............................................................... 242

CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER .......... 243
14.1 List of Common Options of an Object Format Converter .................................... 244
14.2 Changing an Output File name (-o) .................................................................... 245
14.3 Padding (-p) ...................................................................................................... 247

CHAPTER 15 THE S FORMAT BINARY CONVERTER ............................................. 249
15.1 Specification of the S Format Binary Converter .................................................. 250

CHAPTER 16 OPTIONS OF THE S FORMAT BINARY CONVERTER .................... 253
16.1 Options of the S Format Binary Converter ......................................................... 254
16.2 Details on Options of the S Format Binary Converter ......................................... 255
16.2.1 Specifying the Output Range (-ran) .............................................................. 256
16.2.2 Specifying the Split Mode (-sp) .................................................................... 257
16.2.3 Specifying the Inhibition of the Split Mode (-Xsp) ........................................ 258
16.2.4 Specifying to Create a Map List File (-m) ................................................... 259
16.2.5 Specifying not to Create a Map List File (-Xm) ........................................... 261

CHAPTER 17 SPECIFICATIONS OF THE S FORMAT ADJUSTER ....................... 263
17.1 Specifications of the S Format Adjuster ............................................................... 264

CHAPTER 18 OPTIONS OF THE S FORMAT ADJUSTER ..................................... 267
18.1 List of Options of the S Format Adjuster ............................................................ 268
18.2 Details of Options of the S Format Adjuster ....................................................... 269
18.2.1 Specifying the Data Length in an Output Record (-len) .................................. 270
18.2.2 Specifying the Output Range (-ran) ............................................................ 271
18.2.3 Specifying an Output Record (-S1/-S2/-S3) ................................................. 272

CHAPTER 19 LOAD MODULE CONVERTERS ....................................................... 275
19.1 Converting an Absolute Format Load Module into the Motorola S format (f2ms) .... 276
PART 1  LINKAGE KIT

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

CHAPTER 1  SPECTIFICATIONS 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 shows the support range of linkage kit.

![Diagram showing the support range of linkage kit]

Figure 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 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

```
flnk***s  [ Option ] ... < Filename >
```

The command name flnk***s differs depending upon the target CPU of the tool to be used. The command names of the respective target CPUs are shown below.

- In the case of FR family: flnk911s
- In the case of F2MC-16 family: flnk907s
- In the case of F2MC-8L family: flnk896s

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 Differences Based 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

```
flib***s  [ Option ] ... < Filename >
```

The command name flib***s differs based on the target CPU of the tool to be used. The command names of the respective target CPUs are shown below.

- In the case of FR family: flib911s
- In the case of F2MC-16 family: flib907s
- In the case of F2MC-8L family: flib896s

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.
CHAPTER 1 SPECIFICATIONS OF LINKAGE KIT

Object format converter

Commandname [ Option ] ... < Filename >

Determine the object type filename to <filename> based on the functions of each tool. Files of the following four types are the target format.

• Absolute type load module of linker output
• Motorola S format
• Intel HEX format
• Intel expansion 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 CTR-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 shows the relation between the end codes and end status of processing.

<table>
<thead>
<tr>
<th>End code</th>
<th>End status of processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>When ended normally or when an error of warning level occurs.</td>
</tr>
<tr>
<td>1</td>
<td>When an error of warning level occurs with the -cwno option specified</td>
</tr>
<tr>
<td>2</td>
<td>When an error occurs making it impossible to achieve the correct output result</td>
</tr>
<tr>
<td>3</td>
<td>When a fatal error occurs making it impossible to continue processing</td>
</tr>
</tbody>
</table>
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 send 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 “3.2.4 Version number/Specifying message output (-V)” and “3.2.5 Version number/Suppression message output (-XV)” for more details.

■ Startup message

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

The startup message is shown below.

<table>
<thead>
<tr>
<th>Program name</th>
<th>VxxLxx</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 1992-1997</td>
<td></td>
</tr>
<tr>
<td>LICENSED MATERIAL - PROGRAM PROPERTY OF FUJITSU LIMITED</td>
<td></td>
</tr>
</tbody>
</table>
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 “3.2.6 Specifying display of end message (-cmsg)” and “3.2.7 Specifying suppression of end message display (-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 “3.2.3 Help message display (-help)” for more details.

- Example of help message

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

```
usage : flnk911s [-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
    -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 “1.10.2 FELANG”.)
1.8 Identifiers

Linkage kit can handle the following seven kinds of identifiers.

- 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.

At 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 “1.8 Identifier”. Therefore, the filenames that use Chinese characters or spaces must be modified specifying module name at the time of assembling.

  - Filename and characters that can be used for the filename
    
    (Windows 95/98/Me, Windows NT and Windows 2000 Version)

    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 support the following six kinds of environment variable.

- TMP
- FELANG
- FETOOL
- LIB911, LIB907, LIB896
- OPT911, OPT907, OPT896
- OPT

■ TMP (Work directory)

TMP specifies work directory. Refer to “1.10.1 TMP (work directory)” for more details.

■ FELANG

FELANG selects and specifies the message language. Refer to “1.10.2 FELANG” for more details.

■ FETOOL

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

■ LIB911, LIB907, LIB896 (Library search directory)

LIB911, LIB907, and LIB896 specify the directory in which library is stored. Refer to “1.10.4 LIB911, LIB907, LIB896” for more details.

■ OPT911, OPT907, OPT896 (Default option file storage directory)

OPT911, OPT907, OPT896 specify directory in which default option files of linker and librarian are stored. Refer to “1.10.5 OPT911, OPT907, 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 “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]

```
\softune
  \BIN  Stores load module of linkage kit
  \LIB  Stores message file that does not depend on CPU
  \911  Stores library file for FR family and message file
  \907  Stores library file for F2MC-16 family and message file
  \896  Stores library file for F2MC-8L family and message file
```

[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”.

17
1.10.4 LIB911, LIB907, LIB896

LIB911, LIB907 and LIB896 (library file search directory) specify 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 LIB911, LIB907 and LIB896.

---

**LIB911, LIB907, LIB896**

**[Description format]**

```
SET LIB*** = <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 name LIB*** differs depending upon the linker to be used.

The environment variable names for each linker are shown below.

- In the case of flnk911s: LIB911
- In the case of flnk907s: LIB907
- In the case of flnk896s: LIB896

**[Example]**

```
SET LIB911=D:\softune\LIB\911
```

**[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 LIB*** 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 LIB***.
3) The directory (%FETOOL%\LIB\***') 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 OPT911, OPT907, OPT896

OPT911, OPT907 and 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 OPT911, OPT907 and
OPT896 are given below.

### OPT911, OPT907, OPT896

[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 linker and librarian uses are stored.

The environment variable name OPT*** differs depending upon the target CPU. The environment variable names for target CPU are shown below.

- In the case of FR family : OPT911
- In the case of F²MC-16 family : OPT907
- In the case of F²MC-8L family : OPT896

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\911\FLNK911.OPT (In the case of FR family)
  - %FETOOL%LIB\907\FLNK907.OPT (In the case of F²MC-16 family)
  - %FETOOL%LIB\896\FLNK896.OPT (In the case of F²MC-8L family)

- **Librarian**
  - %FETOOL%LIB\911\FLIB911.OPT (In the case of FR family)
  - %FETOOL%LIB\907\FLIB907.OPT (In the case of F²MC-16 family)
  - %FETOOL%LIB\896\FLIB896.OPT (In the case of F²MC-8L family)

[Example]

```
SET OPT911=D:\softune\LIB\911
```
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\F2I.OPT
  - %FETOOL%\LIB\F2E.OPT
  - %FETOOL%\LIB\M2I.OPT
  - %FETOOL%\LIB\M2E.OPT
  - %FETOOL%\LIB\M2B.OPT
  - %FETOOL%\LIB\M2M.OPT

[Example]

```
SET OPT=D:\softune\LIB
```
CHAPTER 2 OPTIONS

This section describes 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 the Inclusive or Contradictory Relation Each Other
2.5 Example of Specifying Command Lines
2.11 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.

```
- Optionname [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.

- Large letters and small letters of alphanumeric 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 options 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=0x1000
```
2.12 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 large 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.
2.13 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=0x1000 -sc data=0x200 : Specifying location of sections (linker)
  When options are specified in order, such as -sc data=0x200 -sc code=0x1000 , all options are effective because the location of sections are individually independent.
2.14 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.
2.15 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]

```
flnk911s
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.

[Example 2]

```
flib911s sys.lib -m sys.mp2 ... *1
flib911s -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]

```
flnk911s *.obj -g -o sample.abs
```

Wild card is used to specify two or more input files in this example.
Linkage kit has common options that can be used in any tools. These options are also prepared in C compiler and assembler. 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 options that can be used in the linkage kit.

### List of common options

Table 3.1 lists common options that can be specified in any tool.

**Table 3.1  List of common options**

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifying suppression not to read default option file</td>
<td>- Xdof</td>
<td></td>
</tr>
<tr>
<td>Specifying option file name</td>
<td>- f</td>
<td></td>
</tr>
<tr>
<td>Specifying display of help message</td>
<td>- help</td>
<td></td>
</tr>
<tr>
<td>Specifying version number and startup message of program</td>
<td>- V</td>
<td></td>
</tr>
<tr>
<td>Suppression not to output version number and startup message of program</td>
<td>- XV</td>
<td>Default</td>
</tr>
<tr>
<td>Specifying display of end message</td>
<td>- cmsg</td>
<td></td>
</tr>
<tr>
<td>Suppression not to output end message</td>
<td>- Xcmsg</td>
<td>Default</td>
</tr>
<tr>
<td>Specifying to set the end code to 1 when warning is issued</td>
<td>- cwno</td>
<td></td>
</tr>
<tr>
<td>Specifying to set the end code to 0 when warning is issued</td>
<td>- Xcwno</td>
<td>Default</td>
</tr>
</tbody>
</table>
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 “3.2.1 Specifying Suppression of Default Option Files (-Xdof)” for more details.

- **-f option**
  The -f option starts reading option from the file in which option is described. Refer to “3.2.2 Specifying Reading from Option File (-f)” for more details.

- **-help option**
  The -help option displays the help message. Refer to “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 “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 “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 “3.2.6 Specifying Display of End Message (-cmsg)” for more details.

- **-Xcmsg option**
  The -Xcmsg option suppresses display of the end message. Refer to “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 “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 “3.2.9 Specifying End Code to 1 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.

[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 “4.5 Default Option File” for the option file.

[Note]

This option is valid when specified in the command line.

[Example]

flnk911s 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 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]

flib911s 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.
3.2 Details of Common Options

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.

```
flib911s -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.

```
flib911s 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 the command name only 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.

- V option specifies to output the startup message. Note that the respective tools of the linkage kit do not output the startup message. 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]
flnk911s 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 command input.

[Example 2]
f2ms ccp903 -V
Softune FJ-OMF to S-FORMAT Converter VxxLxx
ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 1992-1997
LICENSED MATERIAL - PROGRAM PROPERTY OF FUJITSU LIMITED
When starting execution of a program, a startup message (program name, version number, and copyright) is displayed.

[Example 3]
flib911s -V
When only the -V option is specified, a message including program name, version number, and copyright, 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]**

```
fink911s ccp903
fink911s 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 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]

  Note

  [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]
  
  flnk911s 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 COMPLITED 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 at the end 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**
  - `flnk911s ccp903`
  - `flnk911s 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.

  [Example 1]
  
  flnk911s ccp903 -cwno
  
  When a warning is issued during execution of program, the end code to OS is 1.

  [Example 2]
  
  flnk911s ccp903
  
  When a warning is issued during execution of program, the end code to OS remains 0.
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.

**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**

```
flnk911s ccp903
flnk911s ccp903 -Xcwno
```

When warning only is issued during execution of program, 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 section describes option files.

4.1 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 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]

```
flib911s -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 line 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 using the same format.
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 Example when comment is input into content of option file]

  The underlined portion is comment.

<table>
<thead>
<tr>
<th>syslib.lib</th>
<th># INDICATES LIBRARY FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a mod01, mod02, obj03, obj04</td>
<td># Add Modules</td>
</tr>
</tbody>
</table>
## 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, obj04` in command line.

<table>
<thead>
<tr>
<th>Example of Describing Option File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-a mod01,mod02,obj03,obj04</code></td>
<td>Same</td>
</tr>
<tr>
<td><code>-a mod01, mod02 ,obj03 , obj04</code></td>
<td>Inserting a space before and after comma</td>
</tr>
<tr>
<td><code>-a mod01,mod02,obj03,obj04 # comment</code></td>
<td>Adding comment to end of statement</td>
</tr>
<tr>
<td><code># comment line</code> <code>-a mod01,mod02,obj03,obj04</code></td>
<td>Inserting a comment line</td>
</tr>
<tr>
<td><code>-a mod01,mod02,obj03, obj04</code></td>
<td>line feed after comma, continues the parameter</td>
</tr>
<tr>
<td><code>-a mod01,mod02,obj03, obj04</code></td>
<td>line feed before comma, continues the parameter</td>
</tr>
<tr>
<td><code>-a mod01,mod02,obj03, # comment obj04</code></td>
<td>inserting a comment statement continues the parameter</td>
</tr>
<tr>
<td><code>-a mod01,mod02,obj03,obj04</code></td>
<td>line feed after <code>-a</code> continues all parameters</td>
</tr>
</tbody>
</table>
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.

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 -X dof in order to suppress function of the default option. When this option is specified, default option file is not read.

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

Table 4.5 Names of default option files of the linkage kit

<table>
<thead>
<tr>
<th>Name of tool</th>
<th>Name of program</th>
<th>Name of option file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linker</td>
<td>flnk911s</td>
<td>flnk911.opt</td>
</tr>
<tr>
<td></td>
<td>flnk907s</td>
<td>flnk907.opt</td>
</tr>
<tr>
<td></td>
<td>flnk896s</td>
<td>flnk896.opt</td>
</tr>
<tr>
<td>Librarian</td>
<td>lib911s</td>
<td>lib911.opt</td>
</tr>
<tr>
<td></td>
<td>lib907s</td>
<td>lib907.opt</td>
</tr>
<tr>
<td></td>
<td>lib896s</td>
<td>lib896.opt</td>
</tr>
<tr>
<td>Object type converter</td>
<td>f2ms</td>
<td>f2m.opt</td>
</tr>
<tr>
<td></td>
<td>f2is</td>
<td>f2i.opt</td>
</tr>
<tr>
<td></td>
<td>f2es</td>
<td>f2e.opt</td>
</tr>
<tr>
<td></td>
<td>m2is</td>
<td>m2i.opt</td>
</tr>
<tr>
<td></td>
<td>m2e</td>
<td>m2e.opt</td>
</tr>
<tr>
<td></td>
<td>i2ms</td>
<td>i2m.opt</td>
</tr>
<tr>
<td></td>
<td>e2ms</td>
<td>e2m.opt</td>
</tr>
<tr>
<td></td>
<td>m2bs</td>
<td>m2b.opt</td>
</tr>
<tr>
<td></td>
<td>m2ms</td>
<td>m2m.opt</td>
</tr>
</tbody>
</table>

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

- When the environment variable OPT*** or OPT has already been set. The file in the directory that is set by the environment variable is referred to.
  - Linker, librarian
    %OPT***\default option file
  - Object tool
    %OPT%\default option file
4.5 Default Option File

- When the environment variable OPT*** or OPT has not been set
  The default option file in the development environment directory is referred to.
  
  - Linker, librarian
    
    %FETOOL%\LIB\***\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.
Part 2 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

Chapter 5 describes 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 F₂MC-16 Family 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 difficult it becomes 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 shows the 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

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 external 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.
  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 a 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”.

---

58
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. 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 five types of section contents:
  
  An assembler determines the attributes of execution, read, or write. Table 5.3a shows the section types.

  **Table 5.3a  Section types**
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE</td>
<td>Program code area</td>
<td>Executable, Read</td>
</tr>
<tr>
<td>DATA</td>
<td>Variable area</td>
<td>Read, Write</td>
</tr>
<tr>
<td>CONST</td>
<td>Area of variable with initial value</td>
<td>Read</td>
</tr>
<tr>
<td>STACK</td>
<td>Stack area</td>
<td>Read, Write</td>
</tr>
<tr>
<td>IO</td>
<td>I/O area</td>
<td>Read, Write</td>
</tr>
<tr>
<td>DIR</td>
<td>direct area (F²MC family)</td>
<td>Read, Write</td>
</tr>
<tr>
<td>DIRCONST</td>
<td>direct area with initial values (F²MC family)</td>
<td>Read</td>
</tr>
</tbody>
</table>

- **Section location attribute**
  There are two section location attributes representing whether or not the section is relocatable. Table 5.3b shows the section location attributes.

  **Table 5.3b  Section location attributes**
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>Section in which absolute addresses are specified</td>
</tr>
<tr>
<td>REL</td>
<td>Relocatable section</td>
</tr>
</tbody>
</table>

- **Section combination attribute**
  There are two section combination attributes representing whether the section is shared or combined. Table 5.3c shows the section combination attributes.

  **Table 5.3c  Section combination attributes**
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC</td>
<td>Sections are combined in succession.</td>
</tr>
<tr>
<td>COMMON</td>
<td>Sections are combined and overlapped at the same address.</td>
</tr>
</tbody>
</table>
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 5.4a 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.

- **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 5.4b 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.
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 Section 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”.
Section 5.5.1 describes an example of location when the order of combining sections is not specified according to Figure 5.5a.

**Note:** Sections A and C have the PUBLIC attribute and Section B the COMMON attribute.

**Figure 5.5a** 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 5.5b.

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.

Note: Sections A and C have the PUBLIC attribute and Section B the COMMON attribute.

Figure 5.5b 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 5.5c.

Example of location when the section group is specified

If the group is specified, the sections belonging to each section 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).

Note: Sections B and E have the COMMON attribute and other sections the PUBLIC attribute.

The order of location is not specified and Sections C, D, and F are put into the same group.

Figure 5.5c 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.

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

Table 5.6a shows the alignment values and sizes of sections.

Table 5.6a   Alignment values and sizes of sections

<table>
<thead>
<tr>
<th>Section name</th>
<th>Alignment value</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>code1</td>
<td>2</td>
<td>0x0180</td>
</tr>
<tr>
<td>code2</td>
<td>2</td>
<td>0x0100</td>
</tr>
<tr>
<td>code3</td>
<td>2</td>
<td>0x0200</td>
</tr>
<tr>
<td>code4</td>
<td>4</td>
<td>0x0100</td>
</tr>
<tr>
<td>code5</td>
<td>4</td>
<td>0x0200</td>
</tr>
<tr>
<td>code6</td>
<td>2</td>
<td>0x0020</td>
</tr>
</tbody>
</table>

For example, the order of locating sections shown in Table 5.6a 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 5.6a is as shown in Table 5.6b.

Table 5.6b    Alignment values and sizes of sections

<table>
<thead>
<tr>
<th>Order of location processing</th>
<th>Section name</th>
<th>Alignment value</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>code5</td>
<td>4</td>
<td>0x0200</td>
</tr>
<tr>
<td>2</td>
<td>code4</td>
<td>4</td>
<td>0x0100</td>
</tr>
<tr>
<td>3</td>
<td>code3</td>
<td>2</td>
<td>0x0200</td>
</tr>
<tr>
<td>4</td>
<td>code1</td>
<td>2</td>
<td>0x0180</td>
</tr>
<tr>
<td>5</td>
<td>code2</td>
<td>2</td>
<td>0x0100</td>
</tr>
<tr>
<td>6</td>
<td>code6</td>
<td>2</td>
<td>0x0020</td>
</tr>
</tbody>
</table>

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 5.6a and Table 5.6c).

- ro ROM=0xC1000/0xC18FF
  - sc code1+code2+code3+code4+code5+code6=ROM
  - AL 1

<table>
<thead>
<tr>
<th>Section name</th>
<th>Location attribute</th>
<th>Address range</th>
<th>Alignment value</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>code1</td>
<td>REL</td>
<td>-</td>
<td>2</td>
<td>0x0180</td>
</tr>
<tr>
<td>code2</td>
<td>REL</td>
<td>-</td>
<td>2</td>
<td>0x0100</td>
</tr>
<tr>
<td>code3</td>
<td>REL</td>
<td>-</td>
<td>2</td>
<td>0x0200</td>
</tr>
<tr>
<td>code4</td>
<td>REL</td>
<td>-</td>
<td>4</td>
<td>0x0100</td>
</tr>
<tr>
<td>code5</td>
<td>REL</td>
<td>-</td>
<td>4</td>
<td>0x0200</td>
</tr>
<tr>
<td>code6</td>
<td>REL</td>
<td>-</td>
<td>2</td>
<td>0x0020</td>
</tr>
<tr>
<td>AbsSec</td>
<td>ABS</td>
<td>0x1120-0x121F</td>
<td>0</td>
<td>0x0100</td>
</tr>
</tbody>
</table>

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

Table 5.6d Section types and location destinations

<table>
<thead>
<tr>
<th>Location destination</th>
<th>F²MC-8L family</th>
<th>F²MC-16 Family</th>
<th>FR family</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM area (specified in -ro)</td>
<td>CODE</td>
<td>CODE</td>
<td>CODE</td>
</tr>
<tr>
<td></td>
<td>CONST</td>
<td>CONST</td>
<td>CONST</td>
</tr>
<tr>
<td></td>
<td>DIRCONST</td>
<td>DIRCONST</td>
<td></td>
</tr>
<tr>
<td>RAM area (specified in -ra)</td>
<td>IO</td>
<td>IO</td>
<td>IO</td>
</tr>
<tr>
<td></td>
<td>DIR</td>
<td>DIR</td>
<td>DATA</td>
</tr>
<tr>
<td></td>
<td>DATA</td>
<td>DATA</td>
<td>STACK</td>
</tr>
<tr>
<td></td>
<td>STACK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Determining location addresses

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

As shown in Table 5.6e, 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 5.6e Section location destinations

<table>
<thead>
<tr>
<th>Order</th>
<th>Section to be processed</th>
<th>Location destination and method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Section with the ABS attribute</td>
<td>Located at the address provided with the section</td>
</tr>
<tr>
<td>2</td>
<td>Register bank specified in the -rg option</td>
<td>Located at the concerned register bank area</td>
</tr>
<tr>
<td>3</td>
<td>Section with a specified location address in the -sc option, e.g., as &quot;-sc Section=0x0100&quot;</td>
<td>Located at the specified address</td>
</tr>
<tr>
<td>4</td>
<td>Section with a specified location address in the -sc option as &quot;-sc Section=ROM&quot;</td>
<td>In the specified area, the linker searches a place where it can locate the section without overlapping with another section, then locates it.</td>
</tr>
<tr>
<td>5</td>
<td>Section without a specified location address</td>
<td>The linker determines the location area according to Table 5.6d. 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.</td>
</tr>
</tbody>
</table>
5.6 Automatically Locating Sections

**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 5.6c and Table 5.6f).

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

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

Figure 5.6d 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.
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 6.2.19, “Specifying a library file to be searched (-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 for each symbol a library file name in which to search a symbol. For details, see Section 6.2.21, “Specifying a library 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 Searching Libraries

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 5.7a.

---

**Example of a search when there is one library file (1)**

![Diagram](image)

**Note:** The symbol name enclosed in ( ) means a reference and that enclosed in { } means a definition.

**Figure 5.7a** 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 5.7.2 describes one of these examples as shown in Figure 5.7b.

Example of a search when there is one library file (2)

As a result of combining Objects X and Y, both the external reference symbols (S1) and (S2) in X and Y become unsolved.

As a result of searching a library, Module A contains the external defined symbol (S1) and Module B the external defined symbol (S2). These two modules are newly linked.

Since there are no more unsolved external reference symbols, the load module consisting of Modules X, Y, A, and B is created, and the link processing is completed.

Note: The symbol name enclosed in ( ) means a reference and that enclosed in { } means a definition.

Figure 5.7b 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 5.7.3 describes one of these examples as shown in Figure 5.7c.

As a result of combining Object Modules X and Y, both the external reference symbol (S1) in X and Y become unsolved.

As a result of searching a library, Module A contains the external defined symbol (S1), and this module is newly linked.

As a result, the external reference symbol (S2) newly becomes unresolved, and then the library is retrieved again. Module B in which (S2) is defined is linked.

Since there are no more unsolved external reference symbols, the load module consisting of Modules X, Y, A, and B is created and the link processing is completed.

**Note:** The symbol name enclosed in ( ) means a reference and that enclosed in { } means a definition.
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 5.7d.

- Example of a search when there are multiple library files (1)

![Diagram showing search process](image)

If there are multiple library files to be searched, they are searched in the order in which they are specified.

After Modules X and Y modules are linked, the unsolved symbols (S1) and (S3) remain.

Libraries 1 and 2 are searched in this order.

After Library 1 is searched, Module A containing the definition (S1) is linked.

Next, Library 1 is searched for (S3) unsuccessfully, and the search is ended. Then, Library 2 is searched.

Since (S3) is defined in Module C, it is linked.

(Note) The symbol name enclosed in () means a reference and that enclosed in { } means a definition.

Figure 5.7d 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 5.7e.

If there are multiple library files to be searched, they are searched in the order in which they are specified.

If multiple library files contain an external symbol with the same name, the one in the library searched earlier is linked.

To solve an unsolved symbol (S1) in Module X, the program searches Libraries 1 and 2 in this order.

After Library 1 is searched, Module A containing the definition (S1) is linked.

At this time, no unsolved symbol remains and the library search is ended.

In this case, Library 2 is not searched.

(Note) The symbol name enclosed in () means a reference and that enclosed in { } means a definition.

Figure 5.7e Example of a search when there are multiple library files (2)
Section 5.7.6 shows the processing when library files are individually specified as shown in Figure 5.7f.

The external reference symbols for which the library files are individually specified are searched first.

Here, the (S2) symbol is assumed to be imported from Library 2.

Library 2 is searched for (S2) and Module C is linked.

To solve (S1), the program searches Library File 1 and links Module A.

(S2) in Library File 1 is not searched.

(Note)

- The symbol name enclosed in () means a reference and that enclosed in { } means a definition.
- Be sure that, even if the specified libraries does not contain the target external defined symbol but another search target library does, the symbol is solved.

Figure 5.7f  Processing when library files are individually specified
5.8 ROM and RAM Areas

When you develop an application to be imported, there often are 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.

To have a linker locate sections, specify the section name (CODE) and the starting address at which to locate the section (0x1000) as shown in [Example 1].

[Example 1]
-sc CODE=0x1000,DATA=0x0180

Check the ending address of a section in the list map 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=0x1000/0x3FFF
-ra DataA=0x0180/0x57F

The area name CodeA represents the address range of 0x1000 through 0x3FFF. The area name DataA represents the address range of 0x0180 through 0x057F. The section location specification using an area name is as shown in [Example 3].

[Example 3]
-sc CODE=CodeA, DATA=DataA

If you specify an address name to locate a section, 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 containing 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=0x1000,start+CODE+@INIT=0xC000
```

As shown in the figure, specify to locate INIT on RAM (0x1000), and also to locate INIT on ROM (x000) 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 5.9a. Then, the addresses are solved on RAM and the initial value data is located on ROM.
5.9 Sections to be Transferred from ROM to RAM

Figure 5.9a  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, “Restrictions on a Linker.” For an example of a program to transfer data of variables with initial values, see Section 8.2, “Questions and Answers on Using a 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:
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 5.9a shows for each section whether execution, read or write is enabled or disabled after assembling.

Table 5.9a  Attributes of sections after assembly

<table>
<thead>
<tr>
<th>Section name</th>
<th>Execution</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Data</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Prog1</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>Prog2</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

⊙...Enabled  ×...Disabled

If init and Prog1 is 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 5.9b.

Table 5.9b  Attributes of sections after linking

<table>
<thead>
<tr>
<th>Section name</th>
<th>Execution</th>
<th>Read</th>
<th>Write</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>init (RAM)</td>
<td>×</td>
<td>○</td>
<td>○</td>
<td>ROM section provided by the linkerWrite is disabled.</td>
</tr>
<tr>
<td>init (ROM)</td>
<td>×</td>
<td>○</td>
<td>×</td>
<td>ROM section provided by the linkerWrite is disabled.</td>
</tr>
<tr>
<td>Data</td>
<td>×</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Prog1(RAM)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>Write is enabled.</td>
</tr>
<tr>
<td>Prog1(ROM)</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td>ROM section provided by the linker</td>
</tr>
<tr>
<td>Prog2</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

⊙...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
  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
    - F²MC-8L Family : 896.csv
    - F²MC-16 Family : 907.csv
    - FR Family : 911.csv
  - Search Directory
    - F²MC-8L Family : %FETOOL%\LIB\896
    - F²MC-16 Family : %FETOOL%\LIB\907
    - FR Family : %FETOOL%\LIB\911

- 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 a error.
5.11 Mixing of Objects for a F²MC-16 Family Linker

For a linker for the F²MC-16 family (flnk907s), if the target CPU specified in the -cpu option is an upwardly compatible CPU, objects and libraries created for lower CPUs may be mixed.

- Mixing of objects for a F²MC-16 family linker
  
  For a linker for the F²MC-16 family (flnk907s), if the target CPU specified in the -cpu option is an upwardly compatible CPU, objects and libraries created for lower CPUs may be mixed.

  A linker outputs information only if 2 is specified in the -w option and objects are mixed.

- List of CPUs for which objects may be mixed with the -cpu option

  Table 5.10 lists the objects and libraries for lower CPUs that may be mixed with the -cpu option.

**Table 5.11 List of CPUs for which objects may be mixed with the -cpu option**

<table>
<thead>
<tr>
<th>-cpu option specified for linking</th>
<th>-cpu option for creating objects and libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F²MC-16L/16/16H</td>
</tr>
<tr>
<td>CPU of F²MC -16L/16H</td>
<td>☄</td>
</tr>
<tr>
<td>CPU of F²MC -16LX</td>
<td>☐</td>
</tr>
<tr>
<td>CPU of F²MC -16F</td>
<td>☐</td>
</tr>
</tbody>
</table>

- ☄: Same target specified. This is not mixing.
- ☐: Mixing allowed. However, the linker outputs information if 2 is specified in the -w option.
- ☒: Mixing not allowed. The linker outputs an error.
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. The following lists the option names and function outlines.

---

### List of linker options

For details of the parameters and functions required for the options, see the explanations of each option. For explanations of common options, see “Part 1, Chapter 3, Common Options”.

**Table 6.1 List of linker options**

<table>
<thead>
<tr>
<th>Specification regarding output load modules</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output load module file name specification</td>
<td>-o</td>
<td></td>
<td>Default</td>
</tr>
<tr>
<td>Output load module file name specification</td>
<td>-g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug information delete specification</td>
<td>-Xg</td>
<td></td>
<td>Default</td>
</tr>
<tr>
<td>Specification of outputting absolute format load module</td>
<td>-a</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>Specification of outputting relative format load module</td>
<td>-r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specification regarding output lists</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map list file name specification</td>
<td>-m</td>
<td></td>
<td>Default</td>
</tr>
<tr>
<td>Specification for inhibiting map list output</td>
<td>-Xm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cancellation of omitting names displayed in the list</td>
<td>-dt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output specification of the map list file memory used information list</td>
<td>-mmi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification of the number of digits in the list line</td>
<td>-pw</td>
<td>Default 132</td>
<td></td>
</tr>
<tr>
<td>Specification of the number of lines on one list page</td>
<td>-pl</td>
<td>Default 60</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specification regarding output messages</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning message output level specification</td>
<td>-w</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Allocation/link options</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM area specification</td>
<td>-ro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM area specification</td>
<td>-ra</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section allocation</td>
<td>-sc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section group specification</td>
<td>-gr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pack link specification</td>
<td>-pk</td>
<td></td>
<td>FR family only</td>
</tr>
<tr>
<td>Register bank area specification</td>
<td>-rg</td>
<td></td>
<td>F²MC family only</td>
</tr>
<tr>
<td>Automatic allocation specification</td>
<td>-AL</td>
<td></td>
<td>Default 0</td>
</tr>
</tbody>
</table>
### Table 6.1 List of linker options (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library control options</td>
<td>Retrieval library file specification</td>
<td>-l</td>
</tr>
<tr>
<td>Library retrieval path specification</td>
<td>-L</td>
<td></td>
</tr>
<tr>
<td>Library specification for each symbol</td>
<td>-el</td>
<td></td>
</tr>
<tr>
<td>Library retrieval inhibit specification</td>
<td>-nl</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting default library retrieval</td>
<td>-nd</td>
<td></td>
</tr>
<tr>
<td>Entry address specification</td>
<td>-e</td>
<td></td>
</tr>
<tr>
<td>Dummy setting external symbol values</td>
<td>-df</td>
<td></td>
</tr>
<tr>
<td>Target CPU specification</td>
<td>-cpu need</td>
<td></td>
</tr>
<tr>
<td>Inhibiting Check for Presence of Debug Data</td>
<td>-NCI0302LIB</td>
<td></td>
</tr>
<tr>
<td>Function that sets automatically internal ROM/RAM area</td>
<td>-set_rora Default</td>
<td></td>
</tr>
<tr>
<td>Specifies to prevent the internal ROM/RAM area from being set automatically</td>
<td>-Xset_rora</td>
<td></td>
</tr>
<tr>
<td>User-specified-area check specification</td>
<td>-check_rora</td>
<td></td>
</tr>
<tr>
<td>User-specified-area check suppression specification</td>
<td>-Xcheck_rora Default</td>
<td></td>
</tr>
<tr>
<td>Section-placed-area check specification</td>
<td>-check_locate</td>
<td></td>
</tr>
<tr>
<td>Section-placed-area check suppression specification</td>
<td>-Xcheck_locate Default</td>
<td></td>
</tr>
<tr>
<td>Check User-unspecified Section</td>
<td>-check_section</td>
<td></td>
</tr>
<tr>
<td>Inhibit check for presence of user-unspecified section</td>
<td>-Xcheck_section Default</td>
<td></td>
</tr>
<tr>
<td>Options regarding the absolute format assemble list output</td>
<td>Specification for relative assemble list input directory</td>
<td>-alin</td>
</tr>
<tr>
<td>Specification for absolute assemble list output directory</td>
<td>-alout</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output</td>
<td>-als</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output module</td>
<td>-alsf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting absolute assemble list output</td>
<td>-Xals</td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list output</td>
<td>-alr</td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list output module</td>
<td>-alrf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting ROM/RAM and ARRAY list output</td>
<td>-Xalr</td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list symbol and address display position</td>
<td>-na/-an</td>
<td></td>
</tr>
</tbody>
</table>
## Table 6.1 List of linker options (Continued)

<table>
<thead>
<tr>
<th>Function Description</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options regarding the object content list output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for external symbol cross-reference information list output</td>
<td>-xl</td>
<td></td>
</tr>
<tr>
<td>Specification for external symbol cross-reference information list file name</td>
<td>-xlf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting the external symbol crossreference information list output</td>
<td>-Xxl</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting the external symbol crossreference information list output</td>
<td>-sl</td>
<td></td>
</tr>
<tr>
<td>Specification for local symbol list output</td>
<td>-slf</td>
<td></td>
</tr>
<tr>
<td>Specification for local symbol list file name</td>
<td>-Xsl</td>
<td></td>
</tr>
<tr>
<td>Specification for section detail map list output</td>
<td>-ml</td>
<td></td>
</tr>
<tr>
<td>Specification for section detail map list file name</td>
<td>-mlf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting section detail map list output</td>
<td>-Xml</td>
<td></td>
</tr>
<tr>
<td>Common options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting default option file read</td>
<td>-Xdof</td>
<td></td>
</tr>
<tr>
<td>Option file read specification</td>
<td>-f</td>
<td></td>
</tr>
<tr>
<td>Help message display specification</td>
<td>-help</td>
<td></td>
</tr>
<tr>
<td>Specification for version number/message output</td>
<td>-V</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting version number/message output</td>
<td>-XV</td>
<td>Default</td>
</tr>
<tr>
<td>End message display specification</td>
<td>-cmsg</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting end message display</td>
<td>-Xcmsg</td>
<td>Default</td>
</tr>
<tr>
<td>Specification to set the end code to 1 when warning occurs</td>
<td>-cwno</td>
<td></td>
</tr>
<tr>
<td>Specification to set the end code to 0 when warning occurs</td>
<td>-Xcwno</td>
<td>Default</td>
</tr>
</tbody>
</table>
6.2 Details of Linker Options

This section explains each option of the linker. For common options in the linkage kit, see “Part 1, Chapter 3, Common Options”.

- **Options related to the output module**
  Details of options related to the output module in “6.2.1 Output load module file name specification (-o)”- “6.2.5 Relative format load module output specification (-r)”.

- **Options related to the output list**
  Details of options related to the output list in “6.2.6 Map list file name specification (-m)”- “6.2.11 Specification of the number of lines on one page of the list (-pl)”.

- **Specification related to output messages**
  Details of options related to output messages in “6.2.12 Specification of the warning message output level (-w)”.

- **Allocation/link options**
  Details of options related to the allocation/link in “6.2.13 ROM area specification (-ro)”- “6.2.19 Automatic allocation specification (-AL)”.

- **Library control option**
  Details of options related to the library control in “6.2.20 retrieval library file specification (-l)”- “6.2.24 Default library retrieval inhibit specification (-nd)”.

- **Other link control options**
  Details of options related to other link controls in “6.2.25 Entry address specification (-e)”- “6.2.36 Inhibit check for presence of user-unspecified section (-Xcheck_section)”.

- **Options related to the absolute format assemble list output**
  Details of options related to the absolute format assemble list output in “6.2.37 Relative assemble list input directory specification (-alin)”- “6.2.45 ROM/RAM, ARRAY list 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 “6.2.46 External symbol cross-reference information list output specification (-xl)”- “6.2.54 Section detail map list output inhibit specification (-Xsl)”. 
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.

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

<table>
<thead>
<tr>
<th>Link mode</th>
<th>Default extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute format output (-a option)</td>
<td>.abs</td>
</tr>
<tr>
<td>Relative format output (-r option)</td>
<td>.rel</td>
</tr>
</tbody>
</table>

Example 1

flnk911s putc.obj,getc.obj

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

Example 2

flnk911s *.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

flnk911s *.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

flnk911s *.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]

flnk911s -f rlink.opt b1 b2
rlink.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]

```
flnk911s -f rllnk.opt a1 a2 a3 -Xg -o a123.rel
```

rllnk.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]

<table>
<thead>
<tr>
<th>- a</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Default )</td>
</tr>
</tbody>
</table>

[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 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]

flnk911s 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.

<table>
<thead>
<tr>
<th>Relative format load module output specification (-r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[FORMAT]</td>
</tr>
<tr>
<td>- r</td>
</tr>
</tbody>
</table>

[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, 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]
flnk911s a1 a2 a3 -r -o a123.rel
The output object format 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 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 “.mp1”.

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]

flnk911s a1 a2 a3 -r -o a123.rel -m a123.map

The map list file name output by the linker is changed to a123.map.
CHAPTER 6 LINKER OPTIONS

6.2.7 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]
  flnk911s a1 a2 a3 -r -o a123.rel -Xm
  Creating a map list file is inhibited.
6.2.8 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 “7.2 Link List Files”.

- 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, about 80 characters can be displayed. If a small number is set as the number of characters to be displayed in one 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]

flnk911s 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.
6.2.9 Output Specification of the Map List File Memory Used Information List (-mmi)

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

- 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.

Example
flnk911s l0mp00 im_lnk -mmi -ra RAM1=0x1000/0x1FFF,RAM2=0x2000/0x23FF -ro ROM1=0xBC000/0xBCFFF,ROM2=0xBD000/0xBFFFF -AL 2

Note: The following will not be output despite -mmi being specified.
(1) The memory area is not specified. (-ra or -ro is not specified).
(2) The Map List File output is not valid.
6.2.10 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.

**[FORMAT]**

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>- pw &lt; Number of digits &gt;</td>
</tr>
</tbody>
</table>

**[Parameters]**

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

**[Explanation]**

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

If this value is not specified, 132 is set.

**[Example]**

flnk911s a1 a2 a3 -o a123.abs -m a123.map -dt -pw 80

The number of digits to be displayed in one line is set to 80.
6.2.11 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 20-255.

  [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 is set.
  If 0 is specified, page control is canceled.

  [Example]

  flnk911s 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.
6.2.12 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 normally be ignored and those to report simply linker operations are also output.

For details, see “Appendix A Error Messages of the Linker.”

[Explanation]

flnk911s a1 a2 a3 -o a123.abs -w 2 -Xm

All messages are output.
6.2.13 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.

For the F2MC-16 system, use the addresses including the bank number (8-bit) to specify the start address and end address. The start address and end address must not extend over a bank.

[Example]

```
flnk911s *.obj -o ap.abs -ro RomA=0x1000/0x2fff -sc code=RomA ...
```

Section allocation of the section name code to the address 0x1000-0x2fff is specified.

The section is allocated starting with the address 0x1000. A warning message is output if the address 0x2fff is exceeded.
6.2.14 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 also be 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 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 -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.

  For the F2MC-16 system, use the addresses including the bank number (8-bit) to specify the start address and end address. The start address and end address must not extend over a bank.

  [Example]
  
  flnk911s *.obj -o ap.abs -ra RamD=0x0100/0x01ff -sc data=RamD ...

  The section allocation of the section name data to the address 0x0100-0x01ff is specified.

  The section is allocated starting with the address 0x0100, and a warning message is output if the address 0x01ff is exceeded.
6.2.15 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 > ] [ / < Alignment > ]
  [ = { < 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 : only for F²MC-8L

dirconst : only for the F²MC-16 system

<Alignment>
Only F²MC-16 can be specified. The boundary of the next area that can be allocated can be specified using the following keywords.

- BYTE : Allocates at the byte boundary that can be allocated. If any allocated segment exists, allocation is carried out without a gap.
- WORD : Allocates at the word boundary of the address that can be allocated.
- PAGE : Allocates at the page boundary of the address that can be allocated.

<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]

flnk911s *.obj -o ap.abs -sc "/code"=0xC1000,D=0x1000 ...

Figure 6.2.14b shows the allocation example in this case. The section whose content type is code is allocated starting with the address 0xC1000 and section D is allocated starting with the address 0x1000.

![Section allocation example 1]

[Example 2]

flnk911s *.obj -o ap.abs -ro ROM=0xC1000/0xCFFFF -ra RAM=0x1000/0x13FF -sc
P+code+text1=ROM,D=RAM

Figure 6.2b shows the allocation example in this case. If section allocation is specified using the ROM/RAM option, the allocation end address can be checked.
The wild card character that can be used in the `<section name list>` is only `"*"` and the following four description patterns are available.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Example</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>-sc &quot;*/code&quot;</td>
<td>Match with all sections whose content type is code</td>
</tr>
<tr>
<td></td>
<td>ab_1, code_1, XXsect, etc</td>
<td></td>
</tr>
<tr>
<td>Match</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__*</td>
<td>-sc &quot;ab_*/code&quot;</td>
<td>Match with all sections whose content type is code and whose first three characters are &quot;ab_*&quot;</td>
</tr>
<tr>
<td>Match</td>
<td>ab_1, ab_XX, ab_, etc</td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td>aab_XX, ab, etc</td>
<td>characters are &quot;ab_&quot;</td>
</tr>
<tr>
<td>__*</td>
<td>-sc &quot;*/_1/code&quot;</td>
<td>Match with all sections whose content type is code and whose first three characters are &quot;ab_*&quot;, and whose last two characters are &quot;_1&quot;</td>
</tr>
<tr>
<td>Match</td>
<td>ab_1, XX_1, _1, etc</td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td>ab_11, _, etc</td>
<td>two characters are &quot;_1&quot;</td>
</tr>
<tr>
<td>__*</td>
<td>-sc &quot;ab_*_/1/code&quot;</td>
<td>Match with all sections whose content type is code, whose first three characters are &quot;ab_*&quot;, and whose last two characters are &quot;_1&quot;</td>
</tr>
<tr>
<td>Match</td>
<td>ab_XX, ab_1, etc</td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td>aab_XX_11, ab_1, etc</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Sections agreeing with the wild card specifying do not include sections specified by other -sc options or absolute addresses (sections arrangement attributes that are ABS). -sc *=RAM -sc CODE=RAM

All wild card specifying, in specifying like the one above, agree with sections other than the 'CODE' that can be rearranged (sections arrangement attributes that are REL).
6.2.16 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
  dir : only for F^MC-8L
  dirconst : only for the F^MC-16 system

[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]

```
flnk911s *.obj -o ap.abs -ro ROM=0xC1000/0xCffff -ra RAM=0x1000/0x13ff ...
-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]

```
flnk911s *.obj -o ap.abs -gr cdgrp="*/code" -sc cdgrp=0xC1000
```

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 0xC1000 using the -sc option.
6.2.17 Pack Link Specification (-pk)

The alignment specification (boundary adjustment) of allocation information of sections is canceled, then sections are linked and allocated with 1-byte boundary. Application programs may not be executed correctly. Sufficient care must be taken when using this option.
This option is supported only for the FR family.

Pack link specification (-pk)

[Format]

- pk < Section name >

[Parameters]

<Section name>
Specify the section name to be pack linked.

[Explanation]

The alignment specification (boundary adjustment) of allocation information of sections is canceled and then sections are linked (with 1-byte boundary).
This option is supported only for the FR family.
Be careful when specifying -pk. By specifying this option, program operations may be slower or programs may become inoperable.
The -pk option is not intended for alignment of structured members.

[Example]

flnk911s *.obj -o ap.abs -pk code ...
6.2.18 Register Bank Area Specification (-rg)

The register bank area used by programs is specified. This option is supported only for the F\textsuperscript{2}MC family.

- **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.
  
  This option is valid only for the F\textsuperscript{2}MC-8L family/F\textsuperscript{2}MC-16 family.
  
  Table 6.2 shows the areas reserved by the -rg option.
### Table 6.2 Register bank numbers and areas

<table>
<thead>
<tr>
<th>No.</th>
<th>Reserved area F²MC family</th>
<th>Reserved area F²MC-8L family</th>
<th>No.</th>
<th>Reserved area F²MC family</th>
<th>Reserved area F²MC-8L family</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000180 - 0x00018F</td>
<td>0x0100 - 0x0107</td>
<td>16</td>
<td>0x000280 - 0x00028F</td>
<td>0x0180 - 0x0187</td>
</tr>
<tr>
<td>1</td>
<td>0x000190 - 0x00019F</td>
<td>0x0108 - 0x010F</td>
<td>17</td>
<td>0x000290 - 0x00029F</td>
<td>0x0188 - 0x018F</td>
</tr>
<tr>
<td>2</td>
<td>0x0001A0 - 0x0001AF</td>
<td>0x0110 - 0x0117</td>
<td>18</td>
<td>0x0002A0 - 0x0002AF</td>
<td>0x0190 - 0x0197</td>
</tr>
<tr>
<td>3</td>
<td>0x0001B0 - 0x0001BF</td>
<td>0x0118 - 0x011F</td>
<td>19</td>
<td>0x0002B0 - 0x0002BF</td>
<td>0x0198 - 0x019F</td>
</tr>
<tr>
<td>4</td>
<td>0x0001C0 - 0x0001CF</td>
<td>0x0120 - 0x0127</td>
<td>20</td>
<td>0x0002C0 - 0x0002CF</td>
<td>0x01A0 - 0x01A7</td>
</tr>
<tr>
<td>5</td>
<td>0x0001D0 - 0x0001DF</td>
<td>0x0128 - 0x012F</td>
<td>21</td>
<td>0x0002D0 - 0x0002DF</td>
<td>0x01A8 - 0x01AF</td>
</tr>
<tr>
<td>6</td>
<td>0x0001E0 - 0x0001FF</td>
<td>0x0130 - 0x0137</td>
<td>22</td>
<td>0x0002E0 - 0x0002FF</td>
<td>0x01B0 - 0x01B7</td>
</tr>
<tr>
<td>7</td>
<td>0x0001F0 - 0x0001FF</td>
<td>0x0138 - 0x013F</td>
<td>23</td>
<td>0x0002F0 - 0x0002FF</td>
<td>0x01B8 - 0x01BF</td>
</tr>
<tr>
<td>8</td>
<td>0x000200 - 0x00020F</td>
<td>0x0140 - 0x0147</td>
<td>24</td>
<td>0x000300 - 0x00030F</td>
<td>0x01C0 - 0x01C7</td>
</tr>
<tr>
<td>9</td>
<td>0x000210 - 0x00021F</td>
<td>0x0148 - 0x014F</td>
<td>25</td>
<td>0x000310 - 0x00031F</td>
<td>0x01C8 - 0x01CF</td>
</tr>
<tr>
<td>10</td>
<td>0x000220 - 0x00022F</td>
<td>0x0150 - 0x0157</td>
<td>26</td>
<td>0x000320 - 0x00032F</td>
<td>0x01D0 - 0x01D7</td>
</tr>
<tr>
<td>11</td>
<td>0x000230 - 0x00023F</td>
<td>0x0158 - 0x015F</td>
<td>27</td>
<td>0x000330 - 0x00033F</td>
<td>0x01D8 - 0x01DF</td>
</tr>
<tr>
<td>12</td>
<td>0x000240 - 0x00024F</td>
<td>0x0160 - 0x0167</td>
<td>28</td>
<td>0x000340 - 0x00034F</td>
<td>0x01E0 - 0x01E7</td>
</tr>
<tr>
<td>13</td>
<td>0x000250 - 0x00025F</td>
<td>0x0168 - 0x016F</td>
<td>29</td>
<td>0x000350 - 0x00035F</td>
<td>0x01F0 - 0x01FF</td>
</tr>
<tr>
<td>14</td>
<td>0x000260 - 0x00026F</td>
<td>0x0170 - 0x0177</td>
<td>30</td>
<td>0x000360 - 0x00036F</td>
<td>0x01F0 - 0x01F7</td>
</tr>
<tr>
<td>15</td>
<td>0x000270 - 0x00027F</td>
<td>0x0178 - 0x017F</td>
<td>31</td>
<td>0x000370 - 0x00037F</td>
<td>0x01F8 - 0x01FF</td>
</tr>
</tbody>
</table>

[Example]

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

The register bank areas 0, 1, 5, 6, and 7 are reserved.
6.2.19 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 “5.6 Automatic Allocation of Sections”.

[Example]

flnk911s -AL 1 -ro ROM=0x1000/0x1FFF -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=0x1010-0x1017

The following figure shows the link map for this case.
6.2 Details of Linker Options

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.

<table>
<thead>
<tr>
<th>S_Addr. -E_Addr</th>
<th>Size</th>
<th>Section</th>
<th>Type</th>
<th>AL</th>
<th>Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001000-000010F</td>
<td>00000010</td>
<td>CODE P</td>
<td>R-XI</td>
<td>02 REL</td>
<td>code2</td>
</tr>
<tr>
<td>00001010-00001017</td>
<td>00000008</td>
<td>CODE N</td>
<td>R-XI</td>
<td>00 ABS</td>
<td>AbsSec</td>
</tr>
<tr>
<td>00001018-00001047</td>
<td>00000030</td>
<td>CODE P</td>
<td>R-XI</td>
<td>02 REL</td>
<td>code3</td>
</tr>
<tr>
<td>00001048-0000105F</td>
<td>00000018</td>
<td>CODE P</td>
<td>R-XI</td>
<td>02 REL</td>
<td>code1</td>
</tr>
</tbody>
</table>
6.2.20 Retrieval Library File Specification (-l)

Libraries to be retrieved other than the default library are specified. If multiple libraries 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 also be 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 LIB***, 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 LIB***.

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]

flnk911s *.obj -o ap.abs -l ..\lib\com.lib,libu
flnk911s *.obj -o ap.abs -l "p*.lib"

All library files whose file name's first character is "p" are retrieved.
6.2.21 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 LIB*** so that this option need not be specified. The C libraries attached to the C compiler are stored in the path specified by LIB***. To manage libraries created specially 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 LIB*** and system library path derived from the environmental variable FETOOL.

If a library path 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]

flnk911s *.obj -o ap.abs -L C:\usr\usr\lib -l com.lib,libu
6.2.22 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 also be 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]

flnk911s *.obj -o sp.abs -L C:\usr\usr\lib -l libu,sublib -el sym1=sublib
6.2.23 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]
  flnk911s -L C:\usr\usrlib -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]
  flnk911s -l libl *.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.
6.2.24 Default Library Retrieval Inhibit Specification (-nd)

The default library is a library 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.

- 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]
flnk911s -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.
6.2.25 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]

flnk911s *.obj -o ap.abs -e ProgStart
6.2.26 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]**

```
flnk911s -L \usr\usrlib -l libu,sublib *.obj -o ap.abs -df Sym1=100
```

If Sym1 is not defined, 100 is set as its value.
6.2.27 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]
  
  External definition symbol name

  - cpu < MB number>name >

  [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]

  flnk907s *.obj -o ap.abs -cpu MB90600A
  flnk907s *.obj -o ap.abs -cpu MB90700

  **Note:** When executing link processing, the target CPU must be specified using this option.
6.2.28 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.

- 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 data message 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]
  flnk907s *.obj -o ap.abs -cpu MB90704 -NCI0302LIB
  flnk911s *.obj -o ap.abs -cpu MB91110 -NCI0302LIB
6.2.29 Function that sets automatically internal ROM/RAM area (-set_rora)

Refers to CPU information file to set the information regarding the internal ROM/RAM area of the targeted CPU.

- **Function that sets automatically 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 sets automatically the internal ROM/RAM area of the appropriate CPU.

  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]**

  flnk911s *.obj -o ap.abs -cpu MB91110 -set_rora
  flnk907s *.obj -o ap.abs -cpu MB90704 -set_rora
  flnk896s *.obj -o ap.abs -cpu MB89121 -set_rora
6.2.30 Specifies to prevent the internal ROM/RAM area from being set automatically (-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 set automatically (-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]

  flnk911s *.obj -o ap.abs -cpu MB91110 -Xset_rora
  flnk907s *.obj -o ap.abs -cpu MB90704 -Xset_rora
  flnk896s *.obj -o ap.abs -cpu MB89121 -Xset_rora
6.2.31 User-specified-area check specification (-check_rora)

This When the -check_rora option is previously specified, 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 the specified ROM and RAM areas 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]

```
flnk911s -cpu MB91F155 -check_rora -ro ROM = 0x00080800/0x000fffff
   -ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...
-> 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.
```

```
flnk911s -cpu MB91154 -check_rora -ro ROM = 0x00080800/0x000fffff
   -ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...
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.
```

```
flnk911s -cpu MB91154 -check_rora -ro ROM = 0x000A0000/0x000fffff
   -ra RAM1=0x00001000/0x00005fff, RAM2=0x00080000/0x000807ff ...
-> 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.
```

**Note:**
1. Specify the correct MB number for the -cpu option.
2. A warning is output when the -check_rora option is specified even if warning output
   suppression (-w 0) is specified.
3. The -check_rora option is only valid when creating absolute format load modules.
   The -check_rora option is ignored when creating relocatable load modules.
4. 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.
6.2.32 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]
flnk911s -cpu MB91154 -check_rora -ro ROM = 0x00080800/0x000fffff
-ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...
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.
flnk911s -cpu MB91154 -check_rora -Xcheck_rora -ro ROM = 0x00080800/0x000fffff
-ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...
- 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.
6.2.33 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 the CPU data file fs internal ROM data and internal RAM data; and outputs a warning if a section is specified outside the memory area.

- check_locate

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.
### 6.2 Details of Linker Options

<table>
<thead>
<tr>
<th>Section type</th>
<th>Area to be checked</th>
<th>Section that should be placed within the ROM area</th>
<th>Section that should be placed within the RAM area</th>
<th>Section that should be placed within the RAM area or the I/O area</th>
<th>Section that should be placed within the I/O area</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE</td>
<td>The -check_rora option is specified.</td>
<td>A warning is output when a section is placed outside the area specified for -ro and outside the internal-ROM area.</td>
<td>A warning is output when a section is placed outside the area specified for -ro.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONST</td>
<td>The -check_rora option is not specified.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIRCONST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM/RAM Send-from section</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STACK</td>
<td>A warning is output when a section is placed outside the area specified for -ra and outside the internal-RAM area.</td>
<td>A warning is output when a section is placed outside the area specified for -ra.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM/RAM Send-to section</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>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.</td>
<td>A warning is output when a section is placed outside the area specified for -ra.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO</td>
<td>A warning is output when a section is placed outside the internal-I/O area.</td>
<td>A warning is output when a section is placed outside the internal-I/O area.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
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:

```
flnk911s -cpu MB91154 -check_locate -ro ROM = 0x000A0000/0x000fffff
-ra RAM1=0x00001000/0x00005fff, RAM2=0x00080000/0x000807ff ...
```

W1372L: The section is placed outside the ROM area or the I/O area (DATA_A).
W1371L: The section is placed outside the RAM area (STACK_G).
W1372L: The section is placed outside the ROM 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_F).

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.

**Note:**
1. A warning is output when the -check_locate option is specified even if warning output suppression (-w 0) is specified.
2. The -check_locate option is only valid when creating absolute format load modules. The -check_locate option is ignored when creating relocatable load modules.
3. When using the -check_locate option, you must specify the areas for the -ro and -ra options in advance.
4. The -check_locate option also checks ABS-attribute sections not to be processed by the linker.
6.2.34 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.
6.2.35 Check User-unspecified Section (-check_section)

Check for the presence of a section for which the user does not specify layout.

- 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.

flnk911s -cpu MB91154 -check_section
   -ro ROM=0x000a0000/0x000fffff
   -ra RAM1=0x00001000/0x00008fff,RAM2=0x00080000/0x000807ff
   -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.

flnk911s -cpu MB91154 -check_section
   -ro ROM=0x000a0000/0x000fffff
   -ra RAM1=0x00001000/0x00008fff,RAM2=0x00080000/0x000807ff
   -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.

flnk911s -cpu MB91154 -check_section
    -ro ROM=0x000a0000/0x000fffff
    -ra RAM1=0x00001000/0x00008fff,RAM2=0x00080000/0x000807ff
    -sc *=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2...

-> Since the section name is specified by the wildcards, 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 wildcards are used to specify the section name in the -sc option, layout is specified for all the sections that match the wildcards.
   Therefore, no warning is output to the sections that match the wildcards.
Take care when using the wildcards to specify the section name.
6.2.36 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.

- 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.

flnk911s -cpu MB91154 -check_section -Xcheck_section
   -ro ROM=0x000a0000/0x000fffff
   -ra RAM1=0x00001000/0x00008000, RAM2=0x00000000/0x0000807ff
   -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.

flnk911s -cpu MB91154
   -ro ROM=0x000a0000/0x000fffff
   -ra RAM1=0x00001000/0x00008000, RAM2=0x00000000/0x0000807ff
   -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.
6.2.37 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 in which the object module is located is specified.

- 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 in which the object module is located is specified.
  
  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]
  
  flnk911s *.obj -o ap.abs -alin d:\fr20 -alf swctr1.lst,mstdef.lst
  
  flnk911s *.obj -o ap.abs -alf d:\fr20\swctrl.lst,d:\fr20\mstdef.lst

  The above two examples share the same meaning.
6.2.38 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]

  flnk911s *.obj -o ap.abs -alin d:\fr20 -alf swctrl.lst,mstdef.lst -alout d:\fr20\als
6.2.39 Absolute Assemble List Output Specification (-als)

The output of absolute assemble list files is specified. This is an 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]
  
  flnk911s *.obj -o ap.abs -als
6.2.40 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

  **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**
  
  flnk911s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst
  flnk911s *.obj -o ap.abs -alsf swctrl.lst -alsf mstdef.lst
  flnk911s *.obj -o ap.abs -alsf swctrl -alsf mstdef
  
  The above three examples share the same meaning.
6.2.41 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]
flnk911s *.obj -o ap.abs -alf sectrl.lst,mstdef.lst -nl
6.2.42 ROM/RAM and ARRAY List Output Specification (-alr)

The ROM/RAM list and ARRAY list output is specified. The output of debug information when compiling or assembling is required.

ROM/RAM and ARRAY list 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]

flnk911s *.obj -o ap.abs -als -alr -g
flnk911s *.obj -o ap.abs -alr -g

The above two examples share the same meaning.
6.2.43 ROM/RAM and ARRAY List 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 list 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.

[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]

flnk911s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst -alrf swctrl lst,mstdef.lst
flnk911s *.obj -o ap.abs -alsf swctrl.lst -alrf swctrl.lst,mstdef.lst
flnk911s *.obj -o ap.abs -alrf swctrl -alrf mstdef

The above three examples share the same meaning.
6.2.44 ROM/RAM and ARRAY List Output Inhibit Specification (-Xalr)

The inhibition of adding the ROM/RAM and ARRAY lists is specified.

ROM/RAM and ARRAY list output inhibit specification (-Xalr)

[Format]

- Xalr  ( Default )

[Parameters]

None

[Explanation]

The inhibition of adding the ROM/RAM and ARRAY lists is specified.
This option is a default option, therefore, it is not necessary to specify it.
The previously specified -alr and -alrf are canceled.

[Example]

flnk911s *.obj -o ap.abs -als -alr -Xalr
flnk911s *.obj -o ap.abs -als -Xalr

The above two examples share the same meaning.
6.2.45 ROM/RAM and ARRAY List 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 list symbol and address display position specification (-na,-an)

**[Format]**

```
- na
```

**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**

```
flnk911s *.obj -o ap.abs -alr -na
flnk911s *.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**

```
flnk911s *.obj -o ap.abs -alr -na -an
flnk911s *.obj -o ap.abs -alr -an
```

The above two examples share the same meaning.
6.2.46 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]

  flnk911s *.obj -o ap.abs -xl
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]**

  `flnk911s *.obj -o ap.abs -xl -xlf ccp903.mpx`

  `flnk911s *.obj -o ap.abs -xlf ccp903`

  The above two examples share the same meaning.
6.2.48 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]
  
  flnk911s *.obj -o ap.abs -xl -Xxl
  flnk911s *.obj -o ap.abs -Xxl
  flnk911s *.obj -o ap.abs

  The above three examples share the same meaning.
6.2.49 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 or assembling is required.

[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]

flnk911s *.obj -o ap.abs -sl -g
6.2.50 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]**
  
  flnk911s *.obj -o ap.abs -sl -slf ccp903.mps -g
  flnk911s *.obj -o ap.abs -slf ccp903 -g

  The above two examples share the same meaning.
6.2.51 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.
  -sl and -slf specified before are canceled.

  [Example]
  flnk911s *.obj -o ap.abs -sl -Xsl
  flnk911s *.obj -o ap.abs -Xsl
  flnk911s *.obj -o ap.abs
  
  The above three examples share the same meaning.
6.2.52 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 a section detail map list file is specified.
  If this option is not specified, no section detail map list file is created.

  [Example]
  flnk911s *.obj -o ap.abs -ml
6.2.53 Section Detail Map List 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 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]

flnk911s *.obj -o ap.abs -ml -mlf ccp903.mpm
flnk911s *.obj -o ap.abs -mlf ccp903

The above two examples share the same meaning.
Section Detail Map List Output Inhibit Specification (-Xml)

Inhibiting the output of the section detail map list file is specified.

[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.

-ml and -mlf specified before are canceled.

[Example]

flnk911s *.obj -o ap.abs -ml -Xml
flnk911s *.obj -o ap.abs -Xml
flnk911s *.obj -o ap.abs

The above two 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 in the absolute format the assemble list output by the assembler in the relative 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 three parts depending on the information contents.

- Control list
- Map list
- Memory used information list
- External symbol list

We explain here the items output in each list.

- Link List File

The link list file can be divided into the following three parts.

- Control list
  - Specified option
  - Input module
  - 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.

<table>
<thead>
<tr>
<th>FR Family Softune Linker</th>
<th>Control List</th>
<th>YYYY-MM-DD hh:mm:ss</th>
<th>Page: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option File(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) Option file name display area</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Linker control option display area</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Module(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) Input module display area</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) Error message display area</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>FR Family Softune Linker</th>
<th>Control List</th>
<th>1997-11-08 11:04:51</th>
<th>Page: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option File(s)</td>
<td>** no use **</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control(s)</td>
<td>-g</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-a</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-l usrlb.lib</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-ro prog=0x8000/0xffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-ra data=0x0000/0x03ff</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-sc P+code=prog,D+data+S=data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Module(s)</td>
<td>1 pca02.obj(pca01)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 pcasb.obj(pcasb)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 xccdef.obj(xccdef)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error(s)</td>
<td>** Nothing **</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>FR Family Softune Linker</th>
<th>Mapping List</th>
<th>YYYY-MM-DD hh:mm:ss</th>
<th>Page: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-Addr. - E-Addr.</td>
<td>Size</td>
<td>Section type</td>
<td>Al</td>
</tr>
</tbody>
</table>

(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 (F^2MC family only)
- **DIRCONST** : direct section with initial values (F^2MC family only)
- **IO** : IO section

After the section type, the link attribute is displayed.

- **P** : Simple concatenation link
- **C** : Shared link
- **A** : 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

**Al** : Boundary adjustment value for section allocation (hexadecimal)

If the boundary adjustment value is 0x100 or greater, "**" is displayed.
Sec.(Top **) : Section name

"****" indicates how many digits of the section name can be displayed with the specified page width.

(*1) 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.

<table>
<thead>
<tr>
<th>S-Addr.</th>
<th>E-Addr.</th>
<th>Size</th>
<th>Section</th>
<th>type</th>
<th>Al</th>
<th>Sec.(Top 81)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0000001F</td>
<td>00000020</td>
<td>DATA</td>
<td>P RW-</td>
<td>08</td>
<td>REL D</td>
<td></td>
</tr>
<tr>
<td>00000020</td>
<td>00000039</td>
<td>0000001A</td>
<td>DATA</td>
<td>P RW-</td>
<td>02</td>
<td>REL data</td>
<td></td>
</tr>
<tr>
<td>0000003A</td>
<td>00000053</td>
<td>0000001A</td>
<td>STAK</td>
<td>P RW-</td>
<td>02</td>
<td>REL S</td>
<td></td>
</tr>
<tr>
<td>00000054</td>
<td>0000006D</td>
<td>0000001A</td>
<td>DATA</td>
<td>P RW-I</td>
<td>02</td>
<td>REL init</td>
<td></td>
</tr>
<tr>
<td>00008000</td>
<td>00008039</td>
<td>0000003A</td>
<td>CODE</td>
<td>P R-XI</td>
<td>02</td>
<td>REL P</td>
<td></td>
</tr>
<tr>
<td>0000803A</td>
<td>00008053</td>
<td>0000001A</td>
<td>CODE</td>
<td>P R-XI</td>
<td>02</td>
<td>REL code</td>
<td></td>
</tr>
<tr>
<td>00008054</td>
<td>0000806D</td>
<td>0000001A</td>
<td>DATA</td>
<td>P R--I</td>
<td>02</td>
<td>REL # init</td>
<td></td>
</tr>
</tbody>
</table>
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 remainder range and the header and end address of the specified area.

Memory Used Information List Field List Output Format

The following shows the memory used information list field list output format.

<table>
<thead>
<tr>
<th>Fr Family Softune Linker</th>
<th>ROM/RAM Used Info List</th>
<th>YYYY-MM-DD</th>
<th>hh:mm:ss</th>
<th>Page: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-Addr. - E-Addr.</td>
<td>Size</td>
<td>Remain</td>
<td>Name/Status</td>
<td></td>
</tr>
<tr>
<td>(1) ROM/RAM Used Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Area Internal Informations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) General Evaluation Value Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Page Header

Linker name, list name, time and page number are shown in the page header.

(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, have the names of their sections that could not be located and the size of the section.

  - **S_Addr:** Start Address (Hex)
  - **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
  - **-:** Remainder Area Size
  - **Space:** When free/remainder 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 (Hex)
- **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 remainder area (Remainder)
### Example Display of the Memory Used Information List Field List

The following is an example of the Memory Used Information List Field list display.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S_Addr: 00001000-00001FFF</td>
<td>00010000</td>
<td>RAM1</td>
</tr>
<tr>
<td>E_Addr: 00001000-00001774</td>
<td>00000775</td>
<td>0000088B</td>
</tr>
<tr>
<td>Size</td>
<td>Remain</td>
<td>Name/State</td>
</tr>
<tr>
<td>00001137-00001137</td>
<td>00000137</td>
<td>USED</td>
</tr>
<tr>
<td>00001137-00001137</td>
<td>00000001</td>
<td>GAP</td>
</tr>
<tr>
<td>00001138-00001152B</td>
<td>00003F4</td>
<td>USED</td>
</tr>
<tr>
<td>0000152C-0000152F</td>
<td>00000004</td>
<td>GAP</td>
</tr>
<tr>
<td>00001530-00001774</td>
<td>00000246</td>
<td>USED</td>
</tr>
<tr>
<td>0001775-0001FFF</td>
<td>0000088B</td>
<td>FREE</td>
</tr>
<tr>
<td># 0000200-000023FF</td>
<td>00000400</td>
<td>RAM2</td>
</tr>
<tr>
<td>0000200-00003A1E</td>
<td>0001A1F</td>
<td>0000161F</td>
</tr>
<tr>
<td><strong>Not Locate</strong></td>
<td><strong>00000B28</strong></td>
<td>data01</td>
</tr>
<tr>
<td><strong>Not Locate</strong></td>
<td><strong>00000EF6</strong></td>
<td>data02</td>
</tr>
<tr>
<td>0000200-000023FF</td>
<td>00000400</td>
<td>FREE</td>
</tr>
</tbody>
</table>

RAM: Total (00001400) Used (00002194) Remainder (-0000094) <= All RAM Area Information

| # 000BC000-000BCFFFF | 00001000 | ROM1 |
| 000BFF30-000BDFFFF | 00002100 | 00001100 | <= (3) |
| ROM1 Area Information |
| 000BC000-000BCFFFF | 00001000 | USED |

| # 000BD00-000BFFFF | 00003000 | ROM2 |
| 000BD00-000BFFFF | 00000000 | FREE |
| ROM2 Area Information |

| ROM: Total (00004000) Used (00002100) Remainder (+00001900) <= All ROM Area Information |

**Note:**
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 of 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 include 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

<table>
<thead>
<tr>
<th>Symbol Value</th>
<th>Type</th>
<th>Def</th>
<th>Symbol Name(Top **)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol address or symbol value (hexadecimal)</td>
<td>Symbol type</td>
<td>Symbol definition</td>
<td>Symbol name</td>
</tr>
</tbody>
</table>

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.

(*) 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.

<table>
<thead>
<tr>
<th>Symbol Value</th>
<th>Type Def</th>
<th>Symbol Name(Top **)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000004 (ABS)</td>
<td>Addr.OM/LM</td>
<td>_Filename</td>
<td></td>
</tr>
<tr>
<td>00000008 (ABS)</td>
<td>Addr.OM/LM</td>
<td>_c1</td>
<td></td>
</tr>
<tr>
<td>000000C4 (ABS)</td>
<td>Addr. OM/LM</td>
<td>_main</td>
<td></td>
</tr>
<tr>
<td>00000018 (ABS)</td>
<td>Addr. OM/LM</td>
<td>_ulval</td>
<td></td>
</tr>
</tbody>
</table>
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 list
- Assemble source list
- Section information list
- Cross-reference list

We explain here the items output in each list.

Absolute Assemble List File

- 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 list
  - ROM/RAM list
    Global symbol names allocated in the ROM/RAM area 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 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 shows the structure of the list.

| Information list |  
|------------------|---
| (ROM/RAM list)   |  
| (ARRAY list)     | -arl specification
| Assemble source  |  
| Section information |  
| Cross-reference list |  

**Figure 7.3  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]

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Absolute assemble list creation date</th>
<th>Page number</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR Family Softune Linker VxxLxx  yyyy-mm-dd hh:mm:ss  Page: xxxx</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[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.
CHAPTER 7  OUTPUT LIST FILE OF THE LINKER

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 ROM/RAM and ARRAY list format.

<table>
<thead>
<tr>
<th>- ROM/RAM LISTING - (Module name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Name</td>
</tr>
</tbody>
</table>

*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 “6.2.35 ROM/RAM and ARRAY list 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

The following shows the ARRAY list format.

<table>
<thead>
<tr>
<th>- ARRAY LISTING - (Module name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Name</td>
</tr>
</tbody>
</table>

*1: Symbol name

*2: Address

*3: Value

The names of symbols described in programs are displayed.
*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)

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>VALUE</th>
<th>KIND</th>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Line</td>
<td>00002B8</td>
<td>000100</td>
<td>L</td>
<td>ROM</td>
</tr>
<tr>
<td>_Namelen</td>
<td>000004C</td>
<td>000016</td>
<td>L</td>
<td>RAM</td>
</tr>
<tr>
<td>_symref</td>
<td>0005592</td>
<td></td>
<td>T</td>
<td>ROM</td>
</tr>
<tr>
<td>_Tfig</td>
<td>0000763</td>
<td>000002</td>
<td>L</td>
<td>RAM</td>
</tr>
</tbody>
</table>

[Example 2]

- ARRAY LISTING - (sample1)

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>_symref[0].val</td>
<td>0005592</td>
<td>000100</td>
</tr>
<tr>
<td>_symref[0].nam</td>
<td>0005596</td>
<td>000002</td>
</tr>
<tr>
<td>_symref[0].atr</td>
<td>0005598</td>
<td>000016</td>
</tr>
<tr>
<td>_symref[1].val</td>
<td>000559A</td>
<td>012000</td>
</tr>
<tr>
<td>_symref[1].nam</td>
<td>000559E</td>
<td>000002</td>
</tr>
<tr>
<td>_symref[1].atr</td>
<td>00055A0</td>
<td>000016</td>
</tr>
<tr>
<td>_Xpcr[0][0]</td>
<td>0001066</td>
<td>000400</td>
</tr>
<tr>
<td>_Xpcr[0][1]</td>
<td>00010A0</td>
<td>000120</td>
</tr>
</tbody>
</table>
### 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

The following shows the assemble source list format.

<table>
<thead>
<tr>
<th>SN</th>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
<th>*1</th>
<th>*2</th>
<th>*3</th>
<th>*4</th>
<th>*5</th>
<th>*6</th>
<th>*7</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>xxxxxxxx</td>
<td>xxxxxxxxxxxxxxx</td>
<td>x</td>
<td>xxxx</td>
<td>x</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
7.3 Absolute Assemble List File

*6: Progress 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.

[Example]

<table>
<thead>
<tr>
<th>SN</th>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>002CE724</td>
<td></td>
<td></td>
<td>&lt;INIT&gt;</td>
</tr>
<tr>
<td>IN</td>
<td>002CE724</td>
<td>[2]</td>
<td>02</td>
<td>.DATAB.B 2, 2</td>
</tr>
<tr>
<td>IN</td>
<td>002CE726</td>
<td>0020</td>
<td>0010</td>
<td>.DATA.H 32, 16</td>
</tr>
<tr>
<td>CO</td>
<td>0000A280</td>
<td></td>
<td></td>
<td>&lt;CODE&gt;</td>
</tr>
<tr>
<td>CO</td>
<td>0000A280</td>
<td>9F840000043C</td>
<td>1029</td>
<td>LDI:32 #LS1, R4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source program</td>
</tr>
<tr>
<td>Location counter : Displayed with the 32-bit absolute address</td>
</tr>
<tr>
<td>Section acronym : First two characters of the section names</td>
</tr>
</tbody>
</table>
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

The following shows the section information list format.

```
FR Family Softune Linker VxxLxx yyyy-mm-dd hh:mm:ss Page: xxxx
- SECTION LISTING - (Module name)
NO SECTION-NAME SIZE ATTRIBUTES
xx --------Section name-------- xxxxxxx 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
- DATA : Data section
- CONST : Data section with initial values
- COMMON : Shared section
- STACK : Stack section
- DIR : direct section (F 2MC family only)
- DIRCONST : direct section with initial values (F 2MC family only)
- 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]

```
FR Family Softune Linker V30L01 1997-03-03 10:04:51 Page: 78
- SECTION LISTING - (sample1)
NO SECTION-NAME SIZE ATTRIBUTES
0 DATA ..................... 00000004 DATA REL ALIGN=4
1 INIT ..................... 000001C DATA REL ALIGN=4
2 CONST ................... 0000020 CONST REL ALIGN=4
3 CODE................... 0000038 CODE REL ALIGN=2
```
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

The following shows the cross-reference list format.

<table>
<thead>
<tr>
<th>Name</th>
<th>Attrib.</th>
<th>Value</th>
<th>Definition/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*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)
- REL/EXP : Relative symbol (external definition)
- 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]
7.4 External Symbol Cross-reference Information List File

The 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.

<table>
<thead>
<tr>
<th>Module(s)</th>
<th>symbol</th>
<th>Address</th>
<th>Type/Value</th>
<th>Module (No.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. module01</td>
<td>extsym1</td>
<td>0x000012E8</td>
<td>1# 2 3 4 5 6 8 11 12 14</td>
<td></td>
</tr>
<tr>
<td>2. module02</td>
<td>extsym2</td>
<td>0x000C3F34</td>
<td>2 3 4# 5 6 8 11 12 14</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>extsym3</td>
<td>0x000012E6</td>
<td>1# 2 3 4 5 6 8 11 12 13 14 15</td>
<td></td>
</tr>
<tr>
<td>15. module15</td>
<td>extsym4</td>
<td>????</td>
<td>0x00000000</td>
<td>2 5 8</td>
</tr>
<tr>
<td></td>
<td>extsymxxlongxx+</td>
<td>????</td>
<td>0x000CA8ee</td>
<td>4 5 6 11 12 14#</td>
</tr>
<tr>
<td></td>
<td>longxx</td>
<td>????</td>
<td>0x00000000</td>
<td>10 15</td>
</tr>
<tr>
<td></td>
<td>extsymunresolved</td>
<td>????</td>
<td>0x00000000</td>
<td>10 15#</td>
</tr>
<tr>
<td></td>
<td>n1m</td>
<td>EQU</td>
<td>0xFFFFFF00</td>
<td>1 3 4 7 11 12 13 14 15#</td>
</tr>
<tr>
<td></td>
<td>n1p</td>
<td>EQU</td>
<td>0x00000001</td>
<td>4 5 6 11 12 14#</td>
</tr>
</tbody>
</table>

- 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)
7.4 External Symbol Cross-reference Information List File

- **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 Nos. 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.

<table>
<thead>
<tr>
<th>Local Symbol List</th>
<th>yyyy-mm-dd hh:mm:ss</th>
<th>Page: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. module01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. module02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. module15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local Symbol List | yyyy-mm-dd hh:mm:ss | Page: 2

--- Module No. 1 (module01) ---

<table>
<thead>
<tr>
<th>--- symbol ---</th>
<th>--- kind ---</th>
<th>--- val ---</th>
</tr>
</thead>
<tbody>
<tr>
<td>func1(int)</td>
<td>Func. g</td>
<td>0x000C3F34</td>
</tr>
<tr>
<td>localstatic1</td>
<td>Var. s</td>
<td>0x000012EE</td>
</tr>
<tr>
<td>localstatic2</td>
<td>Var. s</td>
<td>0x000012F0</td>
</tr>
</tbody>
</table>

--- Module No. 15 (module15) ---

<table>
<thead>
<tr>
<th>--- symbol ---</th>
<th>--- kind ---</th>
<th>--- val ---</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +(USDollar &amp;, USDollar &amp;)</td>
<td>Func. g</td>
<td>0x00000000</td>
</tr>
<tr>
<td>main</td>
<td>Func. g</td>
<td>0x0000004A</td>
</tr>
<tr>
<td>USDollar::USDollar(unsigned int)</td>
<td>Func. g</td>
<td>0x000000CA</td>
</tr>
<tr>
<td>Atable</td>
<td>loc. s</td>
<td>0xFFFFFFFFC</td>
</tr>
<tr>
<td>Extsym</td>
<td>Var. g</td>
<td>0x00001342</td>
</tr>
</tbody>
</table>
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 in the 3rd and following 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.
For loc., an offset value is displayed.

(*) 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.

The following shows the output format of the section allocation detailed information list.

<table>
<thead>
<tr>
<th>S.Addr. -E.Addr.</th>
<th>Size</th>
<th>Section Type</th>
<th>M.No.</th>
<th>Sec.(Top 18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000-0000048</td>
<td>00000049</td>
<td>DATA P RW--</td>
<td>04 1</td>
<td>data</td>
</tr>
<tr>
<td>0000004A-000003E1</td>
<td>00000348</td>
<td>DATA P RW--</td>
<td>04 2</td>
<td>data</td>
</tr>
<tr>
<td>000003E2-00000403</td>
<td>00000022</td>
<td>DATA P RW--</td>
<td>04 3</td>
<td>data</td>
</tr>
<tr>
<td>00000404-.........</td>
<td>00000000</td>
<td>DATA P RW-I</td>
<td>04 4</td>
<td>data</td>
</tr>
<tr>
<td>00000404-00000663</td>
<td>00000260</td>
<td>DATA P RW-I</td>
<td>04 4</td>
<td>init</td>
</tr>
<tr>
<td>00000664-00000763</td>
<td>00000100</td>
<td>STACK P RW--</td>
<td>04 6</td>
<td>sectionnametoolong+ toolong</td>
</tr>
<tr>
<td>000C0000-000C03DF</td>
<td>000003E0</td>
<td>CODE P R-XI</td>
<td>02 1</td>
<td>code</td>
</tr>
<tr>
<td>000C03E0-000C0441</td>
<td>00000062</td>
<td>CODE P R-XI</td>
<td>02 2</td>
<td>code</td>
</tr>
<tr>
<td>000C0442-000C148B</td>
<td>000104A</td>
<td>CODE P R-XI</td>
<td>02 3</td>
<td>code</td>
</tr>
<tr>
<td>*000C148C-000C201D</td>
<td>00000B92</td>
<td>CODE P R-XI</td>
<td>02 4</td>
<td>code</td>
</tr>
<tr>
<td>*000C2000-000C2203</td>
<td>00000204</td>
<td>CODE P R-XI</td>
<td>02 5</td>
<td>subprog1</td>
</tr>
<tr>
<td>000C3000-000C325F</td>
<td>00000260</td>
<td>DATA P R--I</td>
<td>04 4</td>
<td>#init</td>
</tr>
</tbody>
</table>
7.6 Section Allocation Detailed Information List File

- **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 its section name.
  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.
  
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE</td>
<td>Program section</td>
</tr>
<tr>
<td>DATA</td>
<td>Data section</td>
</tr>
<tr>
<td>CONST</td>
<td>Data section with initial values</td>
</tr>
<tr>
<td>STACK</td>
<td>Stack section</td>
</tr>
<tr>
<td>DIR</td>
<td>Direct section (F2MC Family only)</td>
</tr>
<tr>
<td>DIRCONST</td>
<td>Direct section with initial values (F2MC Family only)</td>
</tr>
<tr>
<td>IO</td>
<td>IO section</td>
</tr>
</tbody>
</table>

  The link attribute is added to the end of the section type.
  
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Simple concatenation link</td>
</tr>
<tr>
<td>C</td>
<td>Shared link</td>
</tr>
<tr>
<td>A</td>
<td>no link</td>
</tr>
</tbody>
</table>

- **type**
  The following attributes are displayed from left.
  
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/-</td>
<td>Read enabled/disabled</td>
</tr>
<tr>
<td>W/-</td>
<td>Write enabled/disabled</td>
</tr>
<tr>
<td>X/-</td>
<td>Executable/non-executable</td>
</tr>
<tr>
<td>I/-</td>
<td>Initial value Yes/No</td>
</tr>
</tbody>
</table>

- **Al**
  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 when using the wild card in the input files and library files.

- **Linker restriction**

  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", "_RAM_section name" are created automatically for each section using the ROM > RAM transfer function.

  Therefore, if a symbol of the same type is defined in a user program, "W1327L:This symbol is already defined (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 below. There are following restrictions on the numbers of symbols and others that can be managed in object files.

<table>
<thead>
<tr>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of input files</td>
<td>65535</td>
</tr>
<tr>
<td>Number of sections (per file)</td>
<td>65535</td>
</tr>
<tr>
<td>Number of external definition symbols (per file)</td>
<td>65535</td>
</tr>
<tr>
<td>Number of external reference symbols (per file)</td>
<td>65535</td>
</tr>
</tbody>
</table>

  **Note**: The files here mean object files. When using relative format load module files and incremental link 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.

## Using the wild card

<table>
<thead>
<tr>
<th>Q.</th>
<th>There are a large number of input object module files. Can the wild card still be used?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>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 also be used here. Refer to the following example for using the wild card.</td>
</tr>
</tbody>
</table>
| Example. | flnk911s *.obj -o outfile.abs  
flnk911s mactrl.obj xz????.obj |

<table>
<thead>
<tr>
<th>Q.</th>
<th>The wild card can be used when specifying section allocation, but how can I use the wild card?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>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 become the keywords when naming the section names, considering the use of the wild card.</td>
</tr>
</tbody>
</table>
| Example. | Section names are defined with the names like DTdata1, DTdata2, DTdata3, DTdata4....for 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)  
- sc DTdata1+DTdata2+DTdata3+DTdata4=0x1000,  
  CDprog1+CDprog2+CDprog3+CDprog4=0X3000  
- sc DT*=0x1000,CD*=0x3000  
- sc */data=0x1000,*/code=0x3000 |
### Handling variables with initial values

<table>
<thead>
<tr>
<th>Q.</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1. [FR family and F^2MC-8L family]</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>In the FR family and F^2MC-8L family, this mechanism is implemented by using the ROM &gt; RAM transfer section function supported by the linker. Variables with initial values generated by the FR family and F^2MC-8L family compilers are gathered in the INIT section.</td>
</tr>
<tr>
<td></td>
<td>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 &gt; RAM transfer section function, see “5.9 Sections to be Transferred from ROM to RAM”.</td>
</tr>
<tr>
<td></td>
<td>The user must write a program to transfer initial value data using assembler languages. Example 1 shows “Program example for transferring initial value data”.</td>
</tr>
</tbody>
</table>
### Example 1

[Program example for transferring initial value data: FR family (data transfer in two bytes)]

```assembly
.import _ROM_INIT, _RAM_INIT ...... (1)
.section INIT, data, align=4 ...... (2)
.section start, code, align=2
LDI # (sizeof INIT + 1) &~0x1, R13
CMP #0, R13
BEQ NOT_INIT
LDI #_ROM_INIT, R2
LDI #_RAM_INIT, R3
LOOP:
   ADD #-2, R13
   LDUH @(R13, R2), R0
   BGT:D LOOP
   STH R0, @(R13, R3)
NOT_INIT:
```

[Program example for transferring initial value data: F2MC-8L family]

```assembly
.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
```

1. ROM_INIT is a symbol to indicate the start address of the INIT section (source) on ROM. _RAM_INIT is a symbol to indicate the start address of the INIT section (destination) on RAM. These symbols are generated automatically by the Sections to be transferred from ROM to RAM. Declare them using the .import instruction.

2. Define an INIT section to extract the section size (transfer size).

(Continued)
### Example 2a

Program example for transferring initial value data: The memory model is either small or medium

```assembly
.section DCONST, const ...... (1)
DCONST_S

.section INIT, data ...... (2)
INIT_S

.section start, code
MOV A, #bnksym (DCONST_S)
MOV ADB, A
MOV A, #bnksym (INIT_S)
MOV DTB, A
MOVW RW0, #sizeof (INIT)
MOVW A, #INIT_S
MOVW A, #DCONST_S
MOVSI DTB, ADB
```

1. **DCONST_S** is a symbol to indicate the start address of the DCONST section (source) on ROM.
2. **INIT_S** is a symbol to indicate the start address of the INIT section (destination) on RAM.

These symbols must be defined at the start of the sections, and so the program to transfer initial value data must be linked first.
## Example 2b

<table>
<thead>
<tr>
<th>Program example for transferring initial value data: The memory model is either compact or large</th>
</tr>
</thead>
</table>

```
.section DTRANS, const
DTRANS_S
.section start, code
MOV A, #bnksym (DTRANS_S)
MOV DTB, A
MOVW RW1, #DTRANS_S
BRA LABEL2
LABEL1:
  MOVW A, @RW1+6
  MOV USB, A
  MOVW A, @RW1+2
  MOV ADB, A
  MOVW A, @RW1+4
  MOVW A, @RW1
  MOVW RW0, @RW1+8
  MOVSI SPB, ADB
  MOVN A, #10
  ADDW RW1, A
LABEL2:
  MOVW A, RW1
  SUBW A, #DTRANS_S
  CMPW A, #sizeof (DTRANS)
  BNE LABEL1
```

(1) \texttt{DTRANS\_S} is a symbol to indicate the start address of the \texttt{DTRANS} section (transfer information storing table) on ROM. These symbols must be defined at the start of the sections, and so the program to transfer initial value data must be linked first.

*(Continued)*
PART 3   LIBRARIAN

Part 3 describes the specifications, options, and output lists of a librarian.

CHAPTER 9   SPECIFICATIONS OF A LIBRARIAN
CHAPTER 10  OPTIONS A LIBRARIAN
CHAPTER 11  LIST FORMATS OF A LIBRARIAN
CHAPTER 12  RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN
Chapter 9 describes the functions and the function types of a librarian. A librarian is a tool used to create a library file.

9.1 Functions of a Librarian
9.2 Function Types of a Librarian
9.3 Creating and Editing a Library File
9.4 Extracting a Module from a Library File
9.5 Deleting Debugging Information of a Library
9.6 Checking and Displaying the Contents of a Library File
9.7 Mixing of Objects for a F²MC-16 Family Librarian
9.1 Functions of a Librarian

A librarian is a tool used to create a library file by combining multiple object modules that an assembler has output.

- **Roles of a 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 files 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 good 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 Roles of a librarian](image-url)
9.2 Function Types of a 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 a 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 9.3a).

![Figure 9.3a Creating a new library file](image)

■ 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 9.3b).

![Figure 9.3b Adding a module](image)
 Deleting a module

An unnecessary module may be deleted from an existing library file (See Figure 9.3c).

Replacing a module

A module in an existing library file may be replaced with a new one (See Figure 9.3d).
9.4 Extracting a Module from a Library File

A module may be extracted from a library file and put back into an object module 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).

![Figure 9.4 Extracting a module](image)

Figure 9.4 Extracting a module
9.5 Deleting Debugging Information of a 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).

Figure 9.5 Deleting debugging information
9.6 Checking and Displaying the Contents of a 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 remain, 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 optionspecification 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.

■ 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 and 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 a Librarian.”
9.7 Mixing of Objects for an F²MC-16 Family Librarian

For a librarian for the F²MC-16 family (flib907s), if the target CPU specified with the -cpu option is an upwardly compatible CPU, objects and libraries created for lower CPUs may be mixed.

- Mixing of objects for an F²MC-16 family
  For a librarian for the F²MC-16 family, if the target CPU specified with the -cpu option is an upwardly compatible CPU, objects created for lower CPUs may be mixed.

- List of CPUs for which objects may be mixed with the -cpu option
  Table 9.7 lists the objects and libraries for lower CPUs that may be mixed with the -cpu option.

<table>
<thead>
<tr>
<th>-cpu option specified for linking</th>
<th>-cpu option for creating objects and libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F²MC-16L/16/16H</td>
</tr>
<tr>
<td>CPU of F²MC -16L/16/16H</td>
<td>☀</td>
</tr>
<tr>
<td>CPU of F²MC -16LX</td>
<td>☀</td>
</tr>
<tr>
<td>CPU of F²MC -16F</td>
<td>✗</td>
</tr>
</tbody>
</table>

- ☀ Same target specified. This is not mixing.
- ☀ Mixing allowed. However, the linker outputs information if 2 is specified in the -w option.
- ✗ Mixing not allowed. The linker outputs an error.
Chapter 10 describes the syntax, parameters, and precautions for options of a librarian.

10.1 List of Options of a Librarian
10.2 Details of Options of a Librarian
## 10.1 List of Options of a Librarian

Options are provided to specify the operations of a librarian in detail.

### List of Options of a Librarian

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 List of options for a librarian**

<table>
<thead>
<tr>
<th>Options for creating and editing a library</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding (registering) a module</td>
<td></td>
<td>-a</td>
<td></td>
</tr>
<tr>
<td>Replacing (registering) a module</td>
<td></td>
<td>-r</td>
<td></td>
</tr>
<tr>
<td>Deleting a module</td>
<td></td>
<td>-d</td>
<td></td>
</tr>
<tr>
<td>Extracting a module</td>
<td></td>
<td>-x</td>
<td></td>
</tr>
</tbody>
</table>

| 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 file | -pl Default: 60 |         |
|                                          | Specifying the number of columns per line of a list file | -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

<table>
<thead>
<tr>
<th>Other options</th>
<th>Specifying to output debugging information</th>
<th>-g</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifying not to output debugging information</td>
<td>-Xg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying a target CPU</td>
<td>-cpu need</td>
<td></td>
</tr>
</tbody>
</table>

### Common options

<table>
<thead>
<tr>
<th>Common options</th>
<th>Specifying not to read a default option file</th>
<th>-Xdof</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifying to read an option file</td>
<td>-f</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying to display help messages</td>
<td>-help</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying to output the version and messages</td>
<td>-V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying not to output the version and messages</td>
<td>-XV Default</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying to display a termination message</td>
<td>-cmsg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying not to display a termination message</td>
<td>-Xcmsg Default</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying to set the termination code to 1 when a warning occurs</td>
<td>-cwno</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying to set the termination code to 0 when a warning occurs</td>
<td>-Xcwno Default</td>
<td></td>
</tr>
</tbody>
</table>
10.2 Details of the Options of a Librarian

Section 10.2 describes the options of a librarian. The options common throughout Linkagekit 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 10.2.16, “Specifying a target CPU (-cpu)”.

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
  
  To 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 wildcard, you cannot specify <Object module file name> separated with commas. Specify them in multiple -a options as shown in the following 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, “Differences of Specifications Depending on the OS”.


10.2 Details of the Options of a Librarian

10.2.2 Replacing (Registering) a Module (-r)

A module in an existing library file is replaced with 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]

```
flib911s 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]

```
flib911s 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 wildcard, 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, “Differences of Specifications 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]

  flib911s 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, use the same name for an object module file and a module unless absolutely necessary. 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 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]

  flib911s syslib -x add
  flib911s syslib.lib -x add,add.obj
  The module, add is extracted from an existing library file and the add.obj file is created.
  flib911s 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.
  flib911s 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]**

```bash
-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]**

```bash
fllib911s syslib.lib -m libx.mp2
```

The module name list registered in syslib.lib is output to libx.mp2.

**[Example 2]**

```bash
fllib911s syslib -a obj1.obj obj2.obj -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]**

```bash
fllib911s 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]

  flib911s 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.


[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 always be specified.

The <Information type> may be specified by listing multiple key words separated with commas.

If the -m option is not specified, this option is invalid.

[Example 1]

```
flib911s syslib.lib -m libx.mp2 -dt r,s
```

The list containing external reference symbols and section names is output to libx.mp2.

[Example 2]

```
flib911s 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]

  ```
  - pl < Number of lines >  ( 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]

  ```
  flib911s  syslib.lib -m libx.mp2 -pl 40
  ```

  The number of lines per page of a list is 40.

  [Example 2]

  ```
  flib911s  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**

  flib911s 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]

flib911s 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.

- Xb

```
flib911s  syslib    -a putc.obj,getc.obj
flib911s  syslib    -a putc.obj,getc.obj -Xb
flib911s  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. The following two items are checked.

- 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 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]

  flib911s 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 following 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]
```
flib911s 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 following 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]
  
  flib911s 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.

f1ib91ls syslib.lib  -a inchar, outchar
f1ib91ls syslib.lib  -a inchar, outchar -Xg
f1ib91ls 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.

[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]

flib907s syslib.lib -a inchar,outchar -cpu MB90600A
flib907s syslib.lib -a inchar,outchar -cpu MB90700

Note: To create a library, you must specify a target CPU using this option. This option may not be omitted.
Chapter 11 describes 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 is output in the following five groups:
- Module name
- Section information of each module
- External definition 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 shows the configuration of a list file.

<table>
<thead>
<tr>
<th>&lt; List header &gt;</th>
<th>Only -m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library file name</td>
<td></td>
</tr>
<tr>
<td>Numbers of registered modules and external definition symbols</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt; Number of modules &gt;</th>
<th>Only -m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of registered modules</td>
<td></td>
</tr>
<tr>
<td>&lt; Section information for each module &gt;</td>
<td>-dt s specified</td>
</tr>
<tr>
<td>&lt; External reference symbol name information for each module &gt;</td>
<td>-dt r specified</td>
</tr>
<tr>
<td>&lt; External definition symbol name information for each module &gt;</td>
<td>-dt d specified</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

| < External defined and reference symbol name information > | -dt a specified |

**Figure 11.1 Configuration of a list file**
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.
  The format of a librarian list is as follows:

  *1: Library file name
  *2: Number of modules registered in a library file (expressed in decimals)
  *3: Number of external definition symbols 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: 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.

<table>
<thead>
<tr>
<th>[ Module Name ]</th>
<th>[ Entry Date ]</th>
<th>[ Creation Date ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModuleA</td>
<td>1997-03-01 14:23:50</td>
<td>1996-03-19 10:03:21</td>
</tr>
<tr>
<td>ModuleC</td>
<td>1997-03-01 14:23:50</td>
<td>1997-02-23 15:15:00</td>
</tr>
</tbody>
</table>

*1: Library file name
*2: Number of modules registered in a library file (expressed in decimals)
*3: Number of external definition symbols 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

Figure 11.2 Format of a librarian list (default)
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 shows the format of a librarian list (when detailed information is specified).

<table>
<thead>
<tr>
<th>Library File Name</th>
<th>Number of Modules:</th>
<th>Number of Symbols:</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample.lib</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Librarian Name: FR Family Softune Librarian

Library Creation Date: 1997-03-01 14:23:50

Library Revision Date: 1997-04-17 09:41:15

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Entry Date</th>
<th>Creation Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModuleA</td>
<td>1997-03-01</td>
<td>1996-03-19 10:03:21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>code</td>
<td>0x00002E8</td>
</tr>
<tr>
<td>data</td>
<td>data</td>
<td>0x000006A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ext_Ref Symbol(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_text tx_len</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ext_Def Symbol(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>prtex</td>
</tr>
</tbody>
</table>

**Figure 11.3 Format of a librarian list (when detailed information is specified)**
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 shows the format of a librarian list (when detailed information is specified).

<table>
<thead>
<tr>
<th>Library File Name</th>
<th>sample.lib</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Modules</td>
<td>3</td>
</tr>
<tr>
<td>Number of Symbols</td>
<td>3</td>
</tr>
<tr>
<td>Librarian Name</td>
<td>FR Family Softune Librarian</td>
</tr>
<tr>
<td>Library Creation Date</td>
<td>1997-03-01 14:23:50</td>
</tr>
<tr>
<td>Library Revision Date</td>
<td>1997-04-17 09:41:15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Entry Date</th>
<th>Creation Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModuleA</td>
<td>1997-03-01</td>
<td>1996-03-19</td>
</tr>
</tbody>
</table>

*1: Output by the a parameter in the -dt option.

Each line displays two of all the external definition symbol names in the entire library.

*2: Each line displays two of all 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 \((Page \ \text{width} - 12) / 2\).

Figure 11.4 Format of a librarian list (when detailed information is specified)
Chapter 12 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
CHAPTER 12  RESTRICTIONS AND QUESTIONS AND ANSWERS ON 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.

Found Questions and answers on creating a library file

| Q. | \begin{itemize} \item What is the format of a file that can be registered in a library file? \end{itemize} |
| A. | \begin{itemize} \item It is an object module that an assembler outputs. This is a file created with the \texttt{.obj} extension by default. \end{itemize} |

Example.

\begin{itemize}
  \item fasm911s file1 \rightarrow file1.obj is output.
  \item fasm911s file2 \rightarrow file2.obj is output.
  \item flib911s libfile -a file1.obj,file2.obj
\end{itemize}

| Q. | \begin{itemize} \item 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. \end{itemize} |
| A. | \begin{itemize} \item 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, you are recommended to register them with debugging information (using the \texttt{-g} option). When debugging is complete, you can delete debugging information (using the \texttt{-O} option). \end{itemize} |

Example.

\begin{itemize}
  \item fasm911s file1 -g \rightarrow file1.obj with debugging information is output.
  \item flib911s libfile -r file1 -g \rightarrow libfile.lib with debugging information is output.
  \item flib911s libfile -O \rightarrow libfile.lib without debugging information is output.
\end{itemize}

| Q. | \begin{itemize} \item 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. \end{itemize} |
| A. | \begin{itemize} \item Use a wildcard to specify object modules to be added to a library (in the \texttt{-a} option) or replaced with the one in a library (in the \texttt{-r} option). \end{itemize} |

Example.

\begin{itemize}
  \item flib911s libfile -a "*.obj" \rightarrow All the files with the "\texttt{.obj}" extension are registered.
\end{itemize}
Q. I forgot what is contained in a library file that I created a while ago. How can I find what are the modules 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. flib911s libfile -m libdoc → A list file, libdoc.mp2 is output.
flib911s libfile -m libdoc -dt a,s
→ A list file with detailed information, libdoc.mp2 is output.

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 flib911s libfile -m -
flib911s libfile -m - -dt a,s
flib911s libfile -a file3.obj -m -

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 characters 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. flib911s libfile -m libdoc -pw 80

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, you are recommended to create a backup of a library before editing it. However, specify the -b option as required.

Example. flib911s libfile -r file1,file2 -d mod4 -b
PART 4  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  THE S FORMAT BINARY CONVERTER
CHAPTER 16  OPTIONS OF THE S FORMAT BINARY CONVERTER
CHAPTER 17  SPECIFICATIONS OF THE S FORMAT ADJUSTER
CHAPTER 18  OPTIONS OF THE S FORMAT ADJUSTER
CHAPTER 19  LOAD MODULE CONVERTERS
CHAPTER 20  OTHER CONVERTERS
CHAPTER 21  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 Overview of an Object Format Converter
13.2 Types of Converters
13.3 Executing an Object Format Converter
13.1 Overview of an Object Format Converter

An object format converter supports the following file formats:
- Absolute format load module that a linker outputs
- Motorola S format
- Intel HEX format
- Intel extended HEX format
- Binary data file

## Relationship between object file converters

Figure 13.1 shows the relationship between converters.
13.2 Types of 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**: Motorola S format
- **i**: Intel HEX format
- **e**: Intel extended HEX format
- **b**: Binary data format

---

**S Format Binary Converter**

An object file in the S format is converted to binary data (a memory image) and output into a file. For details, see Chapter 16, “Option of the S Format Binary Converter”.

**S Format Adjuster**

The S Format Adjuster formats an object file in the S format. For details, see Chapter 18, “Options of the S Format Adjuster”.

**Types of load module converters**

Use the commands in Table 13.2a to convert an object format.

<table>
<thead>
<tr>
<th>Command name</th>
<th>Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>f2ms</td>
<td>Converts an absolute format load module into the Motorola S format</td>
</tr>
<tr>
<td>f2is</td>
<td>Converts an absolute format load module into the Intel HEX format</td>
</tr>
<tr>
<td>f2es</td>
<td>Converts an absolute format load module into the Intel extended HEX format</td>
</tr>
</tbody>
</table>

**Types of other converters**

Use the commands in Table 13.2b to convert an object format.

<table>
<thead>
<tr>
<th>Command name</th>
<th>Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2is</td>
<td>Converts from the Motorola S format into the Intel HEX format</td>
</tr>
<tr>
<td>m2es</td>
<td>Converts from the Motorola S format into the Intel extended HEX format</td>
</tr>
<tr>
<td>i2ms</td>
<td>Converts an absolute format load module into the Intel extended HEX format</td>
</tr>
<tr>
<td>e2ms</td>
<td>Converts from the Intel extended HEX format into the Motorola S format</td>
</tr>
</tbody>
</table>
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
- Motorola S format: .mhx, .ahx
- Intel HEX: .ihx
- Intel extended HEX: .ehx
- Binary data format: .bin

The -ran option is always required to execute the S Format Binary Converter. For details, see Section 16.2.1, “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 Motorola S format is output.
Chapter 14 describes the common options of an object format converter in detail.

14.1 List of Common Options of an Object Format Converter

14.2 Changing an Output File Name (-o)

14.3 Padding (-p)
CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER

14.1 List of Common Options 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 a converter, the following common options may be used.

Table 14.1 lists types of common options of a converter.

#### Table 14.1 List of common options of a converter

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing an output file name</td>
<td>-o</td>
<td></td>
</tr>
<tr>
<td>Specifying padding data</td>
<td>-p</td>
<td></td>
</tr>
<tr>
<td>Specifying not to read a default option file</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to read an option file</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to display help messages</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to output the version and messages</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output the version and messages</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to display a termination message</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to display a termination message</td>
<td>-Xcmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to set the termination code to 1 when a warning occurs</td>
<td>-cwno</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to set the termination code to 0 when a warning occurs</td>
<td>-Xcwno</td>
<td>* Common option</td>
</tr>
</tbody>
</table>

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
- Motorola S format ...... .mhx
- Motorola S format (adjusted) ...... .ahx
- Intel HEX....... .ihx
- Intel extended HEX....... .ehx
- Binary data file ...... .bin

[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 Motorola 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 S Format 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”.
The specified range of addresses is padded with data of a specified value. With the S Format Binary Converter, the portion of an S format 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>`
    * Not to be specified for the S Format Binary Converter or the S Format Adjuster
  - `<Ending address>` Ending address at which to set `<Value>`
    * Not to be specified for the S Format Binary Converter or the S Format Adjuster

  **[Description]**

  The specified range of addresses is padded with data of a specified value.

  Set only the `<Value>` for the S Format Binary Converter.

  For the S Format Binary Converter, the portion of an S format file containing no data is padded with data of a specified value.

  **[Example 1]**

  f2ms ccp903 -p 0xEF,0x1FE4,0x1FFF

  An absolute format load module is converted into the Motorola 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 Motorola S format file.

  **[Example 2]**

  m2bs ccp903 -ran 0x0,0x1FFF -p 0xEF

  The data at the addresses 0x0 through 0xFFF in the Motorola 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 describes the conversion formats of the S Format Binary Converter.

15.1 Specification of the S Format Binary Converter
15.1 Specification of the S Format Binary Converter

The S Format Binary Converter converts a file output in the Motorola S format to a binary data file.
The S Format Binary Converter not only converts data to binary data but also supports the split mode in which to output data into multiple files.

- Overview of the S Format Binary Converter

An object file in the S format created by Linkagekit is converted by the S Format Binary Converter into binary data (memory image) and output into a file.

You may specify multiple input files (S format files). Additionally, binary data after conversion may be split for specified bytes and output into multiple files (hereafter called the split mode).

Note: The portion of an S format 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 S format 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 S Format Binary Converter is split for specified bytes and output into multiple binary data file.

Figure 15.1 shows an overview of the split mode. In Figure 15.1, 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 16.2.2, “Specifying the split mode (-sp)”.

Motorola S format file (.mhx) \(\rightarrow\) m2bs \(\rightarrow\) Binary data file (.bin)
15.1 Specification of the S Format Binary Converter

![Diagram of the S Format Binary Converter](image)

**Figure 15.1** Overview of the split mode
Chapter 16 describes options of the S Format Binary Converter.

16.1 Options of the S Format Binary Converter
16.2 Details on Options of the S Format Binary Converter
## 16.1 Options of the S Format Binary Converter

Section 16.1 lists the option names and functions of the S Format Binary Converter.

### Options of the S Format Binary Converter

Table 16.1 lists options of the S Format Binary Converter.

**Table 16.1 Options of the S Format Binary Converter**

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing an output file name</td>
<td>-o</td>
<td>* Common option for a converter</td>
</tr>
<tr>
<td>Specifying padding data</td>
<td>-p</td>
<td>* Common option for a converter</td>
</tr>
<tr>
<td>Specifying the output range</td>
<td>-ran</td>
<td></td>
</tr>
<tr>
<td>Specifying the split mode</td>
<td>-sp</td>
<td></td>
</tr>
<tr>
<td>Specifying the inhibition of the split mode</td>
<td>-Xsp</td>
<td></td>
</tr>
<tr>
<td>Specifying to create a map list file</td>
<td>-m</td>
<td></td>
</tr>
<tr>
<td>Specifying not to create a map list file</td>
<td>-Xm</td>
<td></td>
</tr>
<tr>
<td>Specifying not to read a default option file</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to read an option file</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to display help messages</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to output the version and messages</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output the version and messages</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to display a termination message</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to display a termination message</td>
<td>-Xcmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to set the termination code to 1 when a warning occurs</td>
<td>-cwno</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to set the termination code to 0 when a warning occurs</td>
<td>-Xcwno</td>
<td>* Common option</td>
</tr>
</tbody>
</table>
16.2 Details on Options of the S Format Binary Converter

Section 16.2 describes the options of the S Format Binary Converter. For information on the common options for Linkagekit, see Chapter 3, “Common Options”. For information on the common options for a converter, see Chapter 14, “Common Options for an Object Format Converter”.

- **Specifying the output range (-ran)**
  
  This option specifies the range of an S format file to be converted to a binary image. For details, see Section 16.2.1, “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 16.2.2, “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 16.2.3, “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 list file. For details, see Section 16.2.4, “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 16.2.5, “Specifying not to create a map list file (-Xm)”. 
16.2.1 Specifying the Output Range (-ran)

Use addresses to specify the range of an S format file to be converted to a binary image. This option must be specified for S Format 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 of an S format file 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 sfmfile.mhx (An example of not using the -ran option)
  An error occurs because the output range is not specified.

  [Example 2]
  m2bs sfmfile.mhx -ran 0xD000,0xFFFF
  The data in sfmfile.mhx at addresses 0xD000 through 0xFFFF is extracted and output into a binary image file.

  [Example 3]
  m2bs sfmfile.mhx -ran 0xD000 (An example of not using 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 sfmfile.mhx -ran 0xFFFF,0xD000
  An error occurs because the specified ending address is smaller than the starting address.

  [Example 5]
  m2bs sfmfile.mhx -ran (Example of omitting all the parameters)
  An error occurs because the starting address is omitted.
16.2.2 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.
16.2.3 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  ccp903.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.
16.2.4 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]

<table>
<thead>
<tr>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Map list file name&gt;</td>
</tr>
<tr>
<td>Output map list file name</td>
</tr>
</tbody>
</table>

[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 sfmfile.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 sfmfile.mhx -ran 0x10000,0x1FFFF -m logfile
```

The information at the time of conversion is output into logfile.mp3.

Contents of logfile.mp3

<table>
<thead>
<tr>
<th>Input file</th>
<th>sfmfile.mhx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output file</td>
<td>sfmfile.bin</td>
</tr>
<tr>
<td>Convert range</td>
<td>0x00010000 - 0x0001FFFF</td>
</tr>
<tr>
<td>Padding data</td>
<td>0xFF</td>
</tr>
</tbody>
</table>
[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.

Contents of logfile.mp3

<table>
<thead>
<tr>
<th>Input file</th>
<th>sfmtfile.mhx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output file</td>
<td>sfmtfile.b01</td>
</tr>
<tr>
<td></td>
<td>sfmtfile.b02</td>
</tr>
<tr>
<td>Convert range</td>
<td>0x00010000 - 0x0001FFFF</td>
</tr>
<tr>
<td>Split byte</td>
<td>2</td>
</tr>
<tr>
<td>Padding data</td>
<td>0xFF</td>
</tr>
</tbody>
</table>
16.2.5 Specifying not to Create a Map List File (-Xm)

Use the -Xm option to specify 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 17 describes the conversion formats of the S Format Adjuster.

17.1 Specifications of the S Format Adjuster
17.1 Specifications of the S Format Adjuster

The S Format Adjuster sorts data created in the Motorola S format in the ascending order of addresses, and causes each of the records to contain the specified number of data.

**Overview of the S Format Adjuster**

The S Format Adjuster causes each of the records in one file of the Motorola S format to contain the specified number of data to unify the format. Figure 17.1 shows the concept of the S Format Adjuster.

![Motorola S format file](.mhx) ➔ m2ms ➔ Motorola S format file

**(Figure 17.1 Concept of the S Format Adjuster)**

The portion of an input S format file containing no data is padded with 0xFF (default value). The output file is the contents of memory converted into the S format without changes.

Use the padding option (-p option) to pad the portion of an S 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 option to unify the lengths of data contained in one record if an existing S format file has records of varying lengths.

**Before conversion**

```
S007000054455354B8
S20FF00000109572C160C2D2CEC
S007000054455354B8
S20FF00000109572C160C2D2CEC
```

**After conversion**

```
S007000054455354B8
S31500FF00000109572C160C2D2CEC
S31500FF00000109572C160C2D2CEC
```
### Functions of the S Format Adjuster

The S 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.

The following section shows an example of converting the address range 0xff0008 through 0xff004a at the record length of 16.

**Before conversion**

```
S007000054455354B8
S20CFF0000109572C160C2D2CEC
S209FF000820406080AD1
S20FFF000D020406080A0C0E1012141660
S20CFF00180109572C160C2D2CD4
S209FF0020020406080AB9
S20FFF0025020406080A0C0E1012141648
S20CFF00300109572C160C2D2C8C
S209FF0038020406080AA1
S20FFF003D020406080A0C0E1012141630
S20CFF00480109572C160C2D2CA4
S209FF0050020406080A89
S20FFF0055020406080A89
S8040000000F
```

**After conversion**

```
S007000054455354B8
S30D00FF000820406080A20406C1
S31500FF0010080A0C0E101214160109572C160C2D2C5B
S31500FF0020020406080A020406080A0C0E1012141629
S31500FF00300109572C160C2D2C020406080A02040689
S31000FF0040080A0C0E10121416010957D7
S70500000000F
```

- If the input S 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 after conversion, 0 is set in the terminator record after conversion.
Chapter 18 describes the options of S Format Adjuster.

18.1 List of Options of the S Format Adjuster
18.2 Details of Options of the S Format Adjuster
# 18.1 List of Options of the S Format Adjuster

Section 18.1 lists the names and functions of options of the S Format Adjuster.

## Options of the S Format Adjuster

Table 18.1 lists the options of the S Format Adjuster.

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing an output file name</td>
<td>-o</td>
<td>* Common option of a converter</td>
</tr>
<tr>
<td>Specifying padding data</td>
<td>-p</td>
<td>* Common option of a converter</td>
</tr>
<tr>
<td>Specifying the output data length</td>
<td>-len</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying the output range</td>
<td>-ran</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying the S1 format output</td>
<td>-S1</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying the S2 format output</td>
<td>-S2</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying the S3 format output</td>
<td>-S3</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to read the default option file</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to read an option file</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to display help messages</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to output the version and messages</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output the version and messages</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to display a termination message</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to display a termination message</td>
<td>-Xcmsg</td>
<td>* Common option</td>
</tr>
</tbody>
</table>
## 18.2 Details of Options of the S Format Adjuster

Section 18.2 describes the options of the S Format Adjuster.

For information on the common options of Linkagekit, see Chapter 3, “Common Options.” For information on the common options of a converter, see Chapter 14, “Object Format Converter.”

- **Specifying the output record data length (-len)**
  
  This option specifies the output record data length. For details, see Section 18.2.1, “Specifying the data length in an output record (-len)”.

- **Specifying the output range (-ran)**
  
  This option specifies the range of formatting an S format file. For details, see Section 18.2.2, “Specifying the output range (-ran)”.

- **Specifying an output record (-S1/-S2/-S3)**
  
  This option specifies an S record to be used when data is output in the S format. For details, see Section 18.2.3, “Specifying an output record (-S1/-S2/-S3)”.
18.2.1 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 an S format file.

■ 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 an S 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.
This option is valid for the S1, S2, and S3 records.

[Example]

m2ms sfmfile.mhx -len 32
sfmfile.mhx is formatted and 32-byte data is output per record.

m2ms sfmfile.mhx (An example of omitting the -len specification)
sfmfile.mhx is formatted and 16-byte data is output per record.

m2ms sfmfile.mhx -len 96
An error occurs because the specified data length is out of the specifiable range.

m2ms sfmfile.mhx (An example of omitting all the parameters)
An error occurs because the data length specification is omitted.
18.2.2 Specifying the Output Range (-ran)

Use this option to specify the range of formatting an S format file 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 an S format file 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.
18.2.3 Specifying an Output Record (-S1/-S2/-S3)

Use this option to specify an S record to be used when data is output.

Specifying an output record (-S1/-S2/-S3)

[Syntax]

- S1
- S2
- S3

[Parameter]

None

[Description]

Use this option to specify an S 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 18.2.)

Table 18.2 List of output record specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Range of data that can be output</th>
<th>Terminator record</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>-S1</td>
<td>0x0000-0xFFFF</td>
<td>S9 record</td>
<td></td>
</tr>
<tr>
<td>-S2</td>
<td>0x0000000-0xFFFFFFFF</td>
<td>S8 record</td>
<td></td>
</tr>
<tr>
<td>-S3</td>
<td>0x0000000000-0xFFFFFFFF</td>
<td>S7 record</td>
<td>(Default)</td>
</tr>
</tbody>
</table>

[Example]

m2ms sfmfile.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 sfmfile.mhx -ran 0xE000,0xFFFF -S1
The data from 0xE000 through 0xFFFF is formatted and output in the S1 record.

m2ms sfmfile.mhx -ran 0xE000,0xFFFF -S2
The data from 0xE000 through 0xFFFF is formatted and output in the S2 record.

m2ms sfmfile.mhx -ran 0xE000,0xFFFF -S3
The data from 0xE000 through 0xFFFF is formatted and output in the S3 record.
Chapter 19 describes the commands of load module converters in detail.

19.1 Converting an Absolute Format Load Module into the Motorola S format
19.2 Converting an Absolute Format Load Module into the Intel HEX format
19.3 Converting an Absolute Format Load Module into the Intel extended HEX format
19.1 Converting an Absolute Format Load Module into the Motorola S format (f2ms)

An absolute format load module that is output by a linker is converted into the Motorola S format. Data at the addresses 0 through 0xFFFFFFFF is to be converted. The f2ms command is used.

- Converting an Absolute Format Load Module into the Motorola S format (f2ms)

  [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 Motorola S format file.

  ![Diagram]
  Absolute format load module \(\rightarrow\) f2ms \(\rightarrow\) Motorola S format

  (.abs) \(\rightarrow\) (.mhx)

  [Address]
  The maximum values of addresses that can be represented in the Motorola S format are:
  
  - S1 type 0xFFFF
  - S2 type 0xFFFFFFFF
  - S3 type 0xFFFFFFFF

  [Precautions]
  The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive.
  Since the Motorola 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:
  
  - 0x00000000 to 0x0000FFFF : S1 type
  - 0x00100000 to 0x0000FFFFF : S2 type
  - 0x10000000 to 0xFFFFFFFF : S3 type
19.2 Converting an Absolute Format Load Module into the Intel HEX format (f2is)

An absolute format load module that is output by a linker is converted into the Intel HEX format.
Data at the addresses 0 through 0xFFFF is to be converted.
The f2is command is used.

- Converting an Absolute Format Load Module into the Intel HEX format (f2is)
  
  [Function]
  The f2is command reads the object data from an absolute format load module that is output by a linker and converts it into an Intel HEX format file.

  ![Diagram]

  Absolute format load module (.abs) → f2is → Intel HEX format (.ihx)

  [Address]
  The maximum value of addresses that can be represented in the Intel HEX 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 Intel HEX 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 Intel HEX format file consists of data records and a trailer record.
19.3 Converting an Absolute Format Load Module into the Intel extended HEX format (f2es)

An absolute format load module that is output by a linker is converted into the Intel extended HEX format.
The data at the addresses 0 through 0xFFFFF is to be converted.
The f2es command is used.

- Converting an Absolute Format Load Module into the Intel extended HEX format (f2es)
  
  [Function]
  
  The f2es command reads the object data from an absolute format load module that is output by a linker and converts it into an Intel extended HEX format file.

  ![Diagram](image)

  Absolute format load module (abs) → f2es → Intel extended HEX format (.ehx)

  [Address]
  
  The maximum value of addresses that can be represented in the Intel extended HEX 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 Intel extended HEX 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 Intel extended HEX format, an extended address record is used to represent the addresses 0x100000 and higher. An extended address record in a file is valid until the next extended address record appears. If a data record appears without an extended address record, then the program calculates addresses assuming that the extended address is specified to be 0.

  A starting address record is created at the beginning of an Intel extended HEX format file.
Chapter 20 describes the commands of load module converters in detail.

20.1 Converting a Motorola S Format File into the Intel HEX Format
20.2 Converting a Motorola S Format File into the Intel Extended HEX Format
20.3 Converting an Intel HEX Format File into the Motorola S Format
20.4 Converting an Intel Extended HEX Format File into the Motorola S Format
20.1 Converting a Motorola S Format File into the Intel HEX Format (m2is)

A Motorola S Format file is converted into the Intel HEX format. Data at the addresses 0 through 0xFFFF is to be converted. The m2is command is used.

[Function]

The m2is command converts a Motorola S format file into the Intel HEX format.

Motorola S format file (.mhx) \(\rightarrow\) m2is \(\rightarrow\) Intel HEX format file (.ihx)

Note: The Motorola S format can represent the addresses 0 through 0xFFFFFFFF. However, when converted into the Intel HEX 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 an Intel HEX format file consists of data records and a trailer record, the starting address information in the Motorola S format will be lost.
20.2 Converting a Motorola S Format File into the Intel Extended HEX Format (m2es)

A Motorola S Format file is converted into the Intel extended HEX format. The data at the addresses 0 through 0xFFFFF is to be converted. The m2es command is used.

**Function**

The m2es command converts a Motorola S format file into the Intel extended HEX format.

![Diagram](attachment:image.png)

**Note:** The Motorola S format can represent the addresses 0 through 0xFFFFFFFF. However, when converted into the Intel extended HEX 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.
20.3 Converting an Intel HEX Format File into the Motorola S Format (i2ms)

An Intel HEX format is converted into the Motorola S format. The data at the addresses 0 through 0xFFFF is to be converted. The i2ms command is used.

Converting an Intel HEX Format File into the Motorola S Format (i2ms)

[Function]
The i2ms command converts an Intel HEX format file into the Motorola S format.

Note: Although the Motorola S format can represent the addresses 0 through 0xFFFFFFFF, the Intel HEX format cannot represent the addresses 0x10000 and higher. A Motorola S format file after conversion is created without the S2, S3, S7, and S8 types. Since the Intel HEX format does not have starting address information, the starting address after conversion is to be 0.
20.4 Converting an Intel Extended HEX Format File into the Motorola S Format (e2ms)

An Intel extended HEX format is converted into the Motorola S format. The data at the addresses 0 through 0xFFFFFFF is to be converted. The e2ms command is used.

- Converting an Intel Extended HEX Format File into the Motorola S Format (e2ms)
  
  [Function]
  
  The e2ms command converts an Intel extended HEX format file into the Motorola S format.

  ![Diagram]

  Intel extended HEX format file
  (.ehx) → e2ms → Motorola S format file
  (.mhx)

  **Note:** Although the Motorola S format can represent the addresses 0 through 0xFFFFFFFF, the Intel extended HEX format cannot represent the addresses 0x100000 and higher. A Motorola S format file after conversion is created without the S7 and S3 types. The starting address information in the Intel extended HEX format is set in the S9 or S8 type.
Chapter 21 describes restrictions and questions and answers on using an object format converter.

21.1 Restrictions on an Object Format Converter

21.2 Questions and Answers on Using an Object Format Converter
21.1 Restrictions on an Object Format Converter

There are some restrictions on the S Format Binary Converter. There are no restrictions on using other converters. Processing can be performed until a converter uses up all the available memory.

■ Restrictions on a converter

For a converter, there are no restrictions on the processing such as the numbers of sections and symbols.

If the numbers of sections and symbols need to be registered in the internal processing of a converter, the memory is dynamically acquired.

When the processing limit is reached, a message for a shortage of memory is reported and the converter is terminated.

■ Restrictions on S Format Binary Converter

Up to 64 S format files can be specified in total.

If multiple S format files are specified, they are processed in the order in which they are specified.

Any data contained in an S format file at the same address is overwritten with data created later.

Up to 2Gbytes minus 1 byte can be converted at one time.
21.2 Questions and Answers on Using an Object Format Converter

Section 21.2 covers the questions and answers on using an object format converter.

### Questions and answers on using an object tool

<table>
<thead>
<tr>
<th>Q.</th>
<th>A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>There are many converters available. Which one should I use?</td>
<td>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. You are recommended to use f2ms because the Motorola S format fully supports the 32bits addressing space. Use other conversion tools as required. Example. f2ms absfile.abs -&gt; Outputs absfile.mhx in the Motorola S format</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q.</th>
<th>A.</th>
</tr>
</thead>
</table>
| When I use S Format Binary Converter, an error, “F9001U: Insufficient memory” is output and I cannot convert data to a binary image. What should I do? | S Format Binary Converter secures as much memory as the area to be converted. If you try to convert a large area 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 0xC00000,0xFFFFFFFF -o absfile1.bin`  
    2. `m2bs absfile.mhx -ran 0xE00000,0xFFFFFFFF -o absfile2.bin`  
    3. `copy /b absfile1.bin + absfile2.bin absfile.bin` |
APPENDIXES

These appendixes describe the error messages of the linkage kit, Intel HEX, Intel extended HEX format, and Motorola S record format.

APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT
APPENDIX B  INTEL HEX AND INTEL EXTENDED HEX FORMAT
APPENDIX C  MOTOROLA 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 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 of processing 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.

```plaintext
*** File name (line number)  XnnnnT:  Message text (supplementary message)
```

<table>
<thead>
<tr>
<th>Part</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>File name</td>
<td>The name of the source file in which an error occurred and the source line number.</td>
</tr>
<tr>
<td>(line number)</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>The level of each error is indicated by the following one alphabetical character.</td>
</tr>
<tr>
<td></td>
<td>I … Information  E… Error</td>
</tr>
<tr>
<td></td>
<td>W … Warning message  F… Fatal error</td>
</tr>
<tr>
<td>nnnn</td>
<td>Error number</td>
</tr>
<tr>
<td></td>
<td>The following shows the correspondence between the error number and error level.</td>
</tr>
<tr>
<td></td>
<td>0000 - 0999 … I</td>
</tr>
<tr>
<td></td>
<td>1000 - 1999 … W</td>
</tr>
<tr>
<td></td>
<td>4000 - 4999 … E</td>
</tr>
<tr>
<td></td>
<td>9000 - 9999 … F</td>
</tr>
<tr>
<td>T</td>
<td>The tool identification is indicated by the following one alphabetical character.</td>
</tr>
<tr>
<td></td>
<td>L … Linker</td>
</tr>
<tr>
<td></td>
<td>U … Librarian, object format converter</td>
</tr>
<tr>
<td>Message text</td>
<td>Error message text (Japanese/English can be selected)</td>
</tr>
<tr>
<td>Supplementary message</td>
<td>Detailed information about error.</td>
</tr>
<tr>
<td></td>
<td>The causes of error are displayed with the symbol names.</td>
</tr>
<tr>
<td></td>
<td>This may also be output in error message text.</td>
</tr>
</tbody>
</table>

[Examples]

*** sample.c(234) E4329L: Value out of range (0xFFF8E37D4)

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

<table>
<thead>
<tr>
<th>Message Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0301L</td>
<td>Unused library (file name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>I0302L</td>
<td>Debug information not exist (file name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>I0303L</td>
<td>Removed debug information</td>
</tr>
<tr>
<td></td>
<td>Output file was created after removing debug information. This is a message which is notified when 2 is specified in the -w option.</td>
</tr>
<tr>
<td>I0304L</td>
<td>File include WARNING level error (file name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>I0305L</td>
<td>Ignore address alignment</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>I0306L</td>
<td>Section allocated automatically in “section name” area</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0307L</td>
<td>Lower compatible cpu type object (file name)</td>
</tr>
<tr>
<td>W1301L</td>
<td>Writable section located in ROM area (section name)</td>
</tr>
<tr>
<td>W1303L</td>
<td>Section with initial data located in RAM area (section name)</td>
</tr>
<tr>
<td>W1305L</td>
<td>Section (section name) located on out of ROM/RAM area (area name)</td>
</tr>
<tr>
<td>W1306L</td>
<td>Exceeded maximum address (section name)</td>
</tr>
<tr>
<td>W1307L</td>
<td>Duplicate section name exist (section name)</td>
</tr>
</tbody>
</table>

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.

A writable section was allocated in the address range specified as a ROM area. Examine the section allocation specification.

A section with initial data was allocated in the address range specified as a RAM area. Examine the section allocation specification.

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.

A section exists which was allocated exceeding the maximum address(0xFFFFFFFF).

Sections with the same name and with different attributes or types exist in multiple modules.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1308L</td>
<td>Overlap located section (section 1, section 2)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1314L</td>
<td>Specified address to absolute section (section name)</td>
</tr>
</tbody>
</table>

Since the absolute address has been determined for the section indicated here, the specified section relocation is illegal.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1320L</td>
<td>Not search library at mode-identification mode</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1321L</td>
<td>Ignore (option) Option at mode-identification mode</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1325L</td>
<td>Entry point already set</td>
</tr>
</tbody>
</table>

Entry points are set in multiple input modules. The entry point set first is valid.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1326L</td>
<td>Entry point was changed</td>
</tr>
</tbody>
</table>

The entry point already set has been changed by specifying the -e option.
APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

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.

W1362L  Missing ' -cpu' option

No target CPU is specified using the -cpu option.
It is assumed that the following is specified to continue the processing.
  •  flnk911s : MB91100
  •  flnk896s : MB89600
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1368L</td>
<td>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.</td>
</tr>
<tr>
<td>W1369L</td>
<td>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.</td>
</tr>
<tr>
<td>W1370L</td>
<td>The section 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.</td>
</tr>
<tr>
<td>W1371L</td>
<td>The section 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.</td>
</tr>
<tr>
<td>W1372L</td>
<td>The section 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.</td>
</tr>
<tr>
<td>W1373L</td>
<td>The section is placed outside 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.</td>
</tr>
</tbody>
</table>
This is the section for which the user does not specify layout.
Use the specify 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** Uncompatible 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.
## APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4326L</td>
<td>Different symbol type (symbol name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>E4327L</td>
<td>Illegal RL information</td>
</tr>
<tr>
<td></td>
<td>This message is notified if any error in relocation information is found.</td>
</tr>
<tr>
<td>E4329L</td>
<td>Value out of range (value)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>E4330L</td>
<td>Devided by 0</td>
</tr>
<tr>
<td></td>
<td>The divisor in the division for relocation operations is 0.</td>
</tr>
<tr>
<td>E4331L</td>
<td>Indispensable to locate address (section name)</td>
</tr>
<tr>
<td></td>
<td>This message is notified if there is no addressing by the -sc option or a section without any specification is found.</td>
</tr>
<tr>
<td>E4332L</td>
<td>Not handling group name (group name)</td>
</tr>
<tr>
<td></td>
<td>A group name cannot be specified when a ROM-RAM transfer section is specified using the -sc option.</td>
</tr>
</tbody>
</table>
E4333L  Not specified ROM address  (section name)

No ROM address is specified for location 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.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4357L</td>
<td>Too many array dimension or structure nested : exceeded 8</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4362L</td>
<td>DUMMY section specified (file name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4363L</td>
<td>Exceeded maximum section size (section name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4364L</td>
<td>Exceeded maximum bank address (section name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4365L</td>
<td>Not found locatable address in area name (section name)</td>
</tr>
</tbody>
</table>

No place in the specified allocation area can be found where the applicable section can be allocated.

Examine the section configuration in the area.
## APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4366L</td>
<td>Not found locatable address (section name)</td>
</tr>
<tr>
<td></td>
<td>No place in all address space can be found where the section can be allocated. Examine the program structure.</td>
</tr>
<tr>
<td>E4367L</td>
<td>Duplicated module name (file name module name)</td>
</tr>
<tr>
<td></td>
<td>The same module name already exists as that intended for linking. Duplicated module names are not allowed. Change the module names.</td>
</tr>
<tr>
<td>E4368L</td>
<td>Referenced other bank symbol (symbol name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>E4370L</td>
<td>CPU information file not found (file name)</td>
</tr>
<tr>
<td></td>
<td>Can't find the target CPU information file specified by the -cpu option. This is detected when the file below is not found.</td>
</tr>
<tr>
<td></td>
<td>- flnk896s : %FETOOL\LIB\896\896.CSV</td>
</tr>
<tr>
<td></td>
<td>- flnk907s : %FETOOL\LIB\907\907.CSV</td>
</tr>
<tr>
<td></td>
<td>- flnk911s : %FETOOL\LIB\911\911.CSV</td>
</tr>
<tr>
<td>E4371L</td>
<td>CPU information not found (file name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>- flnk896s : %FETOOL\LIB\896\896.CSV</td>
</tr>
<tr>
<td></td>
<td>- flnk907s : %FETOOL\LIB\907\907.CSV</td>
</tr>
<tr>
<td></td>
<td>- flnk911s : %FETOOL\LIB\911\911.CSV</td>
</tr>
</tbody>
</table>
APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

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 LIB*** or derived from the environmental variable FETOOL(...\fetool\lib).

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 or hardware fault has occurred.

F9017L  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 linker.

This message is also notified if a write-protected file with the same name exists.
APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

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.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9032L</td>
<td>Output file name same as input one (file name)</td>
</tr>
<tr>
<td></td>
<td>Since the output file name indicated here is the same as the input file name, processing cannot continue.</td>
</tr>
<tr>
<td>F9033L</td>
<td>Illegal file format (file name)</td>
</tr>
<tr>
<td></td>
<td>This message is notified in one of the following cases.</td>
</tr>
<tr>
<td></td>
<td>• The library file format is not correct.</td>
</tr>
<tr>
<td></td>
<td>• The object module format in a library file is not correct.</td>
</tr>
<tr>
<td></td>
<td>• The input file is an absolute format load module.</td>
</tr>
<tr>
<td></td>
<td>• Contents of an input module are not correct.</td>
</tr>
<tr>
<td></td>
<td>• The format of a relative format assemble list file is not correct.</td>
</tr>
<tr>
<td></td>
<td>• CPU Information file format is not correct.</td>
</tr>
<tr>
<td>F9040L</td>
<td>Duplicated file or path name (file name)</td>
</tr>
<tr>
<td></td>
<td>This message is notified in one of the following three cases.</td>
</tr>
<tr>
<td></td>
<td>• Input module files of the same name are specified.</td>
</tr>
<tr>
<td></td>
<td>• Library files of the same name are specified in the -l, or -el option.</td>
</tr>
<tr>
<td></td>
<td>• The same library path name is specified in the -L option more than once.</td>
</tr>
<tr>
<td>F9042L</td>
<td>Duplicated section name (section name)</td>
</tr>
<tr>
<td></td>
<td>The same section name was specified in the -sc or -gr option more than once.</td>
</tr>
<tr>
<td>F9043L</td>
<td>Duplicated symbol name (symbol name)</td>
</tr>
<tr>
<td></td>
<td>The same external reference symbol was specified in the -el option more than once.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Message Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>F9044L</td>
<td>Duplicated section group name (group name)</td>
</tr>
<tr>
<td></td>
<td>The same group name was used more than once when setting groups using the -gr option.</td>
</tr>
<tr>
<td>F9047L</td>
<td>No match file name argument</td>
</tr>
<tr>
<td></td>
<td>A file was specified using the wild card, but no corresponding file was found.</td>
</tr>
<tr>
<td>F9052L</td>
<td>Missing <code>-cpu</code> option</td>
</tr>
<tr>
<td></td>
<td>No target CPU is specified using the -cpu option.</td>
</tr>
<tr>
<td></td>
<td>In flnk907s, the target CPU must be specified using the -cpu option.</td>
</tr>
<tr>
<td>F9053L</td>
<td>Missing <code>-ro</code> or <code>-ra</code> option</td>
</tr>
<tr>
<td></td>
<td>The -ro option or -ra option required for automatic allocation is not specified.</td>
</tr>
<tr>
<td>F9056L</td>
<td>Mismatch CPU information file version</td>
</tr>
<tr>
<td></td>
<td>The CPU information file is not suited old.</td>
</tr>
<tr>
<td></td>
<td>Please obtain the CPU information file of the latest.</td>
</tr>
<tr>
<td>F9998L</td>
<td>File open failed (file name)</td>
</tr>
<tr>
<td></td>
<td>The message files used by the linker could not be opened.</td>
</tr>
<tr>
<td></td>
<td>Store the error message files(lkt***.a.msg,lkt***.e.msg) in a predetermined directory.</td>
</tr>
<tr>
<td>F9999L</td>
<td>Internal error (identification information)</td>
</tr>
<tr>
<td></td>
<td>If this error occurs, report it to Fujitsu immediately.</td>
</tr>
</tbody>
</table>
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 lines 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.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1404U</td>
<td>Nothing to operate</td>
</tr>
<tr>
<td>W1405U</td>
<td>Module not exists to delete (module name)</td>
</tr>
<tr>
<td>W1406U</td>
<td>Module not exists to extract (module name)</td>
</tr>
<tr>
<td>W1462U</td>
<td>Missing '-cpu' option</td>
</tr>
<tr>
<td>E4402U</td>
<td>Duplicated module name (file name module name)</td>
</tr>
</tbody>
</table>

There was no library file change nor module extract operation.

The module specified by the -d option is not included in the library file.
Check the registered module names using the -m option.

The module specified by the -x option is not included in the library file.
Check the registered module names using the -m option.

No target CPU is specified using the -cpu option.
It is assumed that the following is specified to continue the processing.

- flib911s: MB91100
- flib896s: MB89600

An attempt was made to register a module with the same name as that which has already been registered.

Duplicated names are not allowed in one library and so the module indicated here is not registered. Use the -r option for replacement.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4403U</td>
<td>Duplicated external definition symbol name (file name symbol name)</td>
</tr>
</tbody>
</table>

The registered modules contain external definition symbols and the symbol indicated here has already been 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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4404U</td>
<td>Invalid module : type (file name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4405U</td>
<td>Invalid module : conflict tool name (file name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4406U</td>
<td>Invalid module : conflict compile model (file name)</td>
</tr>
</tbody>
</table>

A module of a different compile model (such as the memory model) cannot be registered in the same library.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4407U</td>
<td>Invalid module : conflict CPU type (file name)</td>
</tr>
</tbody>
</table>

A module of a different target CPU cannot be registered in the same library.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4408U</td>
<td>Too many entry modules : exceed 65535</td>
</tr>
</tbody>
</table>

Create another library so that one library contains at most 65535 modules.
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4409U</td>
<td>Too many entry external symbols: exceed 65535</td>
</tr>
</tbody>
</table>

Create another library so that one library contains at most 65535 symbols.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4470U</td>
<td>CPU information file not found (file name)</td>
</tr>
</tbody>
</table>

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
- %FETOOL\LIB\907\907.CSV
- %FETOOL\LIB\911\911.CSV

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4471U</td>
<td>CPU information not found (file name)</td>
</tr>
</tbody>
</table>

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
- %FETOOL\LIB\907\907.CSV
- %FETOOL\LIB\907\907.CSV

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E9001U</td>
<td>Insufficient memory</td>
</tr>
</tbody>
</table>

Enough memory is not available to execute the program.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9015U</td>
<td>File open error (file name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9016U</td>
<td>File read error (file name)</td>
</tr>
</tbody>
</table>

It is possible that the file is read-protected.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9017U</td>
<td>File write error (file name)</td>
</tr>
<tr>
<td></td>
<td>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 cannot be written. Prepare enough free space in the disk before rerunning the linker.</td>
</tr>
<tr>
<td>F9021U</td>
<td>Too many options</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>F9022U</td>
<td>Illegal option name (option)</td>
</tr>
<tr>
<td></td>
<td>An error is found in the option name specification. Correct the command line and then reactivate.</td>
</tr>
<tr>
<td>F9023U</td>
<td>Illegal option parameter (option)</td>
</tr>
<tr>
<td></td>
<td>An error is found in the parameters to be specified in this option.</td>
</tr>
<tr>
<td>F9026U</td>
<td>Specified value out of range (value)</td>
</tr>
<tr>
<td></td>
<td>The value cannot be specified in the parameters of the -pl or -pw option. See the -help option or this manual.</td>
</tr>
<tr>
<td>F9027U</td>
<td>Option file nested</td>
</tr>
<tr>
<td></td>
<td>Nesting of option files is not allowed. Delete the -f option described in the option file.</td>
</tr>
<tr>
<td>F9033U</td>
<td>Illegal file format (file name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9035U</td>
<td>Missing library file name</td>
</tr>
<tr>
<td></td>
<td>The library name is not correctly specified. Specify the library file name correctly.</td>
</tr>
<tr>
<td>F9036U</td>
<td>Multiple library file name specified (file name)</td>
</tr>
<tr>
<td></td>
<td>Only one library file can be specified. Select either the file name indicated here or a library file name specified before.</td>
</tr>
<tr>
<td>F9045U</td>
<td>'-O' option conflict with another option</td>
</tr>
<tr>
<td></td>
<td>If the -O option is specified, other options cannot be specified.</td>
</tr>
<tr>
<td>F9046U</td>
<td>'-c' option conflict with another option</td>
</tr>
<tr>
<td></td>
<td>Do not combine the -c option with any other options.</td>
</tr>
<tr>
<td>F9047U</td>
<td>No match file name argument</td>
</tr>
<tr>
<td></td>
<td>A file was specified using the wild card, but no corresponding file was found.</td>
</tr>
<tr>
<td>F9052U</td>
<td>Missing '-cpu' option</td>
</tr>
<tr>
<td></td>
<td>No target CPU is specified using the -cpu option.</td>
</tr>
<tr>
<td></td>
<td>In flib907s, the target CPU must be specified using the -cpu option.</td>
</tr>
<tr>
<td>F9056U</td>
<td>Mismatch CPU information file version</td>
</tr>
<tr>
<td></td>
<td>The CPU information file is not suited old. Please obtain the CPU information file of the latest.</td>
</tr>
</tbody>
</table>
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9998U</td>
<td><strong>File open failed (file name)</strong></td>
</tr>
<tr>
<td></td>
<td>The message files used by the library file could not be opened.</td>
</tr>
<tr>
<td></td>
<td>Store the error message files(lkt<em><strong>_a.msg,lkt</strong></em>_e.msg) in a</td>
</tr>
<tr>
<td></td>
<td>predetermined directory.</td>
</tr>
<tr>
<td>F9999U</td>
<td><strong>Internal error (identification information)</strong></td>
</tr>
<tr>
<td></td>
<td>If this error occurs, report it to Fujitsu immediately.</td>
</tr>
</tbody>
</table>
### Error Messages of the Object Format Converter

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0501U</td>
<td>Skip start address record</td>
</tr>
</tbody>
</table>

The start address record was contained in the Intel HEX format, but it was skipped since it was not required. Converter processing is not affected and conversion is carried out correctly.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1501U</td>
<td>File include WARNING level error (file name)</td>
</tr>
</tbody>
</table>

The file specified for input contained an error of the warning level when linking. Check the file before using it.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1502U</td>
<td>Unable to convert address (address)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9001U</td>
<td>Insufficient memory</td>
</tr>
</tbody>
</table>

Enough memory is not available to execute the program.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9011U</td>
<td>Input file not found (file name)</td>
</tr>
</tbody>
</table>

The file specified in the input file cannot be found.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9015U</td>
<td>File open error (file name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9016U</td>
<td>File read error (file name)</td>
</tr>
</tbody>
</table>

It is probable that the file is read-protected or hardware fault has occurred.
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.

Too many input file names and options (including those in the option file) are specified on the command line.

The option cannot be used for the converter. See the -help option or this manual.

An error is found in parameters to be specified for this option.

A value outside the range of values allowed by the option is specified.

Nesting of option files is not allowed. Delete the -f option described in the option file.

The address specified in parameters of the option cannot be represented in the converted file format. Specify another address.
APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9029U</td>
<td>Start address opposite to end one (option: s=address1 e=address2)</td>
</tr>
</tbody>
</table>

The end address in parameters of the option is smaller than the start address.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9030U</td>
<td>Missing input file name</td>
</tr>
</tbody>
</table>

Specify the name of an input file to be converted.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9031U</td>
<td>Multiple input file name (file name)</td>
</tr>
</tbody>
</table>

Only one input file can be specified. Specify either the file name indicated here or a file name specified before as the input file.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9032U</td>
<td>Output file name same as input one (file name)</td>
</tr>
</tbody>
</table>

Since the output file name indicated here is the same as the input file name, processing cannot continue.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9033U</td>
<td>Illegal file format (file name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9034U</td>
<td>Not absolute load module file (file name)</td>
</tr>
</tbody>
</table>

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.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9048U</td>
<td>Missing output range</td>
</tr>
<tr>
<td></td>
<td>The output range is not specified. Specify range of</td>
</tr>
<tr>
<td></td>
<td>output(-ran).</td>
</tr>
<tr>
<td>F9049U</td>
<td>Output range exceeded</td>
</tr>
<tr>
<td></td>
<td>The output range exceeds the limit value.</td>
</tr>
<tr>
<td>F9050U</td>
<td>Output filename same as other output one (file name)</td>
</tr>
<tr>
<td></td>
<td>The output file has the same name as that of another</td>
</tr>
<tr>
<td></td>
<td>file.</td>
</tr>
<tr>
<td>F9051U</td>
<td>File name too long (file name)</td>
</tr>
<tr>
<td></td>
<td>The output file name indicated here is too long to be</td>
</tr>
<tr>
<td></td>
<td>processed. Make the specified file name shorter.</td>
</tr>
<tr>
<td>F9998U</td>
<td>File open failed (file name)</td>
</tr>
<tr>
<td></td>
<td>The message files used by the converter could not be</td>
</tr>
<tr>
<td></td>
<td>opened. Store both lkt_e.msg and lkt_a.msg in the</td>
</tr>
<tr>
<td></td>
<td>predetermined directory.</td>
</tr>
<tr>
<td>F9999U</td>
<td>Internal error (identification information)</td>
</tr>
<tr>
<td></td>
<td>If this error occurs, report it to Fujitsu immediately.</td>
</tr>
</tbody>
</table>

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 filename same as other output one (file name)

The output file has the same name as that of another file.

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 both lkt_e.msg and lkt_a.msg in the predetermined directory.

F9999U  Internal error (identification information)

If this error occurs, report it to Fujitsu immediately.
APPENDIX B  INTEL HEX AND INTEL EXTENDED HEX FORMAT

This appendix explains the following two formats of the Intel HEX formats.
• Intel HEX format: Format set for the 8-bit MPU (i8080)
• Intel extended HEX format: Format extended for the 16-bit MPU (i8086)

B.1 Common Format
B.2 Data Record (Intel HEX, Intel Extended HEX) Type:00
B.3 End Record (Intel HEX, Intel Extended HEX) Type:01
B.4 Extended Address Record (Intel Extended HEX) Type:02
B.5 Start Address Record (Intel Extended HEX) Type:03
APPENDIX B  INTEL HEX AND INTEL EXTENDED HEX FORMAT

B.1 Common Format

The Intel (extended) HEX format consists of six fields, (a) to (f). Each field is set using the ASCII code. Field (g) is explained later.

Common format

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>11</td>
<td>12</td>
<td>a1</td>
<td>a2</td>
<td>a3</td>
<td>a4</td>
</tr>
<tr>
<td>:</td>
<td>t1</td>
<td>t2</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d4</td>
</tr>
<tr>
<td>:</td>
<td>d*</td>
<td>d*</td>
<td>d*</td>
<td>d*</td>
<td>s1</td>
<td>s2</td>
</tr>
</tbody>
</table>

(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.
11 indicates the high-order digits and 12 indicates the low-order digits. Values in the range of 0 - 255 can be set.
It is “00” - “FF” in ASCII notation and “0x3030” - “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 - 65535 can be set.
It is “0000” - “FFFF” in ASCII notation and “0x30303030” - “0x46464646” in hexadecimal notation.

(d):
Indicates the record type.
00 : Data record (Intel HEX format, Intel extended HEX format)
01 : End record (Intel HEX format, Intel extended HEX format)
02 : Extended address record (Intel extended HEX format)
03 : Start address record (Intel extended HEX format)

(e):
This field depends on the record type of (d). See the explanations of each record in B2 to B5.

(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.

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 (Intel HEX, Intel Extended HEX) Type: 00

$d_1$ and $d_2$ are byte data at the address indicated by (C) and $d_3$ and $d_4$ are byte data of the next address.

| : | 11 | 12 | a1 | a2 | a3 | a4 | 0 | 0 | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d^*$ | $d^*$ | $d^*$ | $d^*$ | $s_1$ | $s_2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (a) | (b) | (c) | (d) | (e) | (f) | (g) |

For (a), (b), (c), (d), (f), and (g), see the explanation of the 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 $d_1$ and $d_2$.

Likewise, byte data of the next address is $d_3$ and $d_4$. 
B.3 End Record (Intel HEX, Intel Extended HEX) Type: 01

The end record is always 00000001FF. Only one end record exists as the last record.

- End record (Intel HEX, Intel extended HEX)

```
<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(f)</th>
<th>(g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
```

The (e) field does not exist. Therefore 0 is set in (b). (c) is not used and normally 0 is set.
B.4 Extended Address Record (Intel Extended HEX) Type: 02

If extended address code appears, the address of each byte data of the following data records is calculated according to the following formula.

\[( PA \times 0x10 ) + ( ( DA + DP ) \text{ MOD } 0x10000 ) \text{ MOD } 0x100000 \]

- **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 i8086. With the addition of extended address code, up to 20-bit addresses can be represented.

For the data records which appear before the extended address record, the address is calculated with the above PA setting as 0x0000.

### Extended address record (Intel extended HEX)

```
    : 0 i 2 0 i 0 0 i 0 0 i 2 p1 p2 p3 p4 s1 s2
```

- **(a)**
- **(b)**
- **(c)**
- **(d)**
- **(e)**
- **(f)**
- **(g)**

**(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 address code appears, the address of each byte data of the following data records is calculated according to the following formula.

\[( PA \times 0x10 ) + [ ( DA + DP ) \text{ MOD } 0x10000 ] \text{ MOD } 0x100000 \]

- **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.
B.5 Start Address Record (Intel Extended HEX) 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.

\[(\text{PA} \times 0x10) + \text{IP}\mod 0x100000\]

- PA: (e1) field value of the record
- IP: (e2) field value of the record

---

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>p1</td>
<td>p2</td>
<td>p3</td>
<td>p4</td>
<td>i1</td>
<td>i2</td>
<td>i3</td>
</tr>
</tbody>
</table>

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.

\[[(\text{PA} \times 0x10) + \text{IP}] \mod 0x100000\]

- 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.
The S record format of Motorola always starts with the character “S” (0x53). Eight types from S0 to S9 are available (S4 and S6 are not used).

C.1 S0 Type (Header Record)
C.2 S1 Type (Data Record: 2-Byte Address)
C.3 S2 Type (Data Record: 3-Byte Address)
C.4 S3 Type (Data Record: 4-Byte Address)
C.5 S5 Type (Record to Manage the Number of Records)
C.6 S7 Type (Terminator Record)
C.7 S8 Type (Terminator Record)
C.8 S9 Type (Terminator Record)
C.1 S0 Type (Header Record)

This record is used for comments.

- **S0 type (header record)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>Type field. “S0” (0x5330) in ASCII code.</td>
</tr>
<tr>
<td>(b)</td>
<td>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. 11 indicates the high-order digits and 12 indicates the low-order digits. Values in the range of 0-255 can be set. It is “00”-“FF” in ASCII notation and “0x3030”-“0x4646” in hexadecimal notation.</td>
</tr>
<tr>
<td>(c)</td>
<td>This field is not used and normally “0000” in ASCII notation is set.</td>
</tr>
<tr>
<td>(d)</td>
<td>Messages such as version number management information are set. For the setting method, see the example below.</td>
</tr>
<tr>
<td>(e)</td>
<td>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</td>
</tr>
<tr>
<td>(f)</td>
<td>Generally, control code (such as CR and LF) is added. Data in this field is skipped until the start character “S” of (a) appears.</td>
</tr>
</tbody>
</table>

**[Example]**

```plaintext
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 (0x0000-0xFFFF).

- **S1 type (data record: 2-byte address)**

<table>
<thead>
<tr>
<th>S</th>
<th>11,12</th>
<th>a1,a2</th>
<th>a3,a4</th>
<th>d1, d2</th>
<th>d3, d4</th>
<th>d*, d*</th>
<th>s1, s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td>(e)</td>
<td>(f)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>n bytes</td>
<td>2 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The S1 type consists of the above five fields (a)-(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 0-65535 can be set.
It is “0000”-“FFFF” in ASCII notation and “0x30303030”-“0x46464646” in hexadecimal notation.

(b):
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).”)

- \( S \) 1 11,12 a1,a2 a3,a4 d1, d2 d3, d4 d*, d* s1, s2

- (a) (b) (c) (d) (e) (f)

- 2 bytes 2 bytes 4 bytes n bytes 2 bytes
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.

| S   2 11 12  a1 a2  a3 a4  a5 a6  d1 d2  d3 d4  d* d*  s1  s2 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| (a) | (b) | (c) | (d) | (e) | (f) |

2 bytes 2 bytes 4 bytes \( n \) bytes 2 bytes

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.

<table>
<thead>
<tr>
<th>S3 type (data record: 4-byte address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 3</td>
</tr>
</tbody>
</table>

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)

```
<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>0</th>
<th>3</th>
<th>n1</th>
<th>n2</th>
<th>n3</th>
<th>n4</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(e)</td>
<td>(f)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>2 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

The S5 type consists of the above four fields (a)-(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 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 0-65535 can be set.

It is "0000"-"FFFF" in ASCII notation and "0x03003030"-"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)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>7</td>
<td>0</td>
<td>5</td>
<td>e1 e2 e3 e4 e5 e6 e7 e8 s1 s2</td>
</tr>
</tbody>
</table>

(a) (b) (c) (e) (f)
2 bytes 2 bytes 8 bytes 2 bytes

The S7 type consists of the above four fields (a)-(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.

<table>
<thead>
<tr>
<th>S</th>
<th>8</th>
<th>0</th>
<th>4</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
<th>e6</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(e)</td>
<td>(f)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 bytes 2 bytes 6 bytes 2 bytes

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)

<table>
<thead>
<tr>
<th>S</th>
<th>9</th>
<th>0</th>
<th>3</th>
<th>e1, e2</th>
<th>e3, e4</th>
<th>s1, s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(e)</td>
<td>(f)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 bytes 2 bytes 4 bytes 2 bytes

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

Appendix D lists the linker options.

- List of linker options

Attached Table D  List of linker options

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification for outputting load module file name</td>
<td>-o</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for debugging information output</td>
<td>-g</td>
<td></td>
</tr>
<tr>
<td>Specification for deleting debugging information</td>
<td>-Xg</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for outputting absolute format load module</td>
<td>-a</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for outputting relative format load module</td>
<td>-r</td>
<td></td>
</tr>
<tr>
<td>Map list file specification</td>
<td>-m</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for inhibiting map list output</td>
<td>-Xm</td>
<td></td>
</tr>
<tr>
<td>Canceling the omission of names displayed in the list</td>
<td>-dt</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting memory used information list</td>
<td>-mmi</td>
<td></td>
</tr>
<tr>
<td>Specification of the number of digits in the list</td>
<td>-pw</td>
<td>Default 132</td>
</tr>
<tr>
<td>Specification of the number of lines on one list page</td>
<td>-pl</td>
<td>Default 60</td>
</tr>
<tr>
<td>Specification for warning message output level</td>
<td>-w</td>
<td></td>
</tr>
<tr>
<td>ROM area specification</td>
<td>-ro</td>
<td></td>
</tr>
<tr>
<td>RAM area specification</td>
<td>-ra</td>
<td></td>
</tr>
<tr>
<td>Section allocation</td>
<td>-sc</td>
<td></td>
</tr>
<tr>
<td>Section group specification</td>
<td>-gr</td>
<td></td>
</tr>
<tr>
<td>Pack link specification</td>
<td>-pk</td>
<td>FR family only</td>
</tr>
<tr>
<td>Register bank area specification</td>
<td>-rg</td>
<td>F2MC-16 family/F2MC-8L family only</td>
</tr>
<tr>
<td>Automatic allocation specification</td>
<td>-AL</td>
<td>Default 0</td>
</tr>
<tr>
<td>Retrieval library file specification</td>
<td>-l</td>
<td></td>
</tr>
<tr>
<td>Library retrieval path specification</td>
<td>-L</td>
<td></td>
</tr>
<tr>
<td>Library specification for each symbol</td>
<td>-el</td>
<td></td>
</tr>
<tr>
<td>Library retrieval inhibit specification</td>
<td>-nl</td>
<td></td>
</tr>
<tr>
<td>Default library retrieval inhibit specification</td>
<td>-nd</td>
<td></td>
</tr>
<tr>
<td>Entry address specification</td>
<td>-e</td>
<td></td>
</tr>
<tr>
<td>Dummy setting of external symbol values</td>
<td>-df</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Option</td>
<td>Remarks</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>Target CPU Specification</td>
<td>-cpu</td>
<td>must be specified</td>
</tr>
<tr>
<td>Inhibiting Check for Presence of Debug Data</td>
<td>-NCI0302LIB</td>
<td></td>
</tr>
<tr>
<td>Function that sets automatically internal ROM/RAM area</td>
<td>-set_rora</td>
<td>Default</td>
</tr>
<tr>
<td>Specifies to prevent the internal ROM/RAM area from being set automatically</td>
<td>-Xset_rora</td>
<td></td>
</tr>
<tr>
<td>User-specified-area check specification</td>
<td>-check_rora</td>
<td></td>
</tr>
<tr>
<td>User-specified-area check suppression specification</td>
<td>-Xcheck_rora</td>
<td>Default</td>
</tr>
<tr>
<td>Section-placed-area check specification</td>
<td>-check_locate</td>
<td></td>
</tr>
<tr>
<td>Section-placed-area check suppression specification</td>
<td>-Xcheck_locate</td>
<td>Default</td>
</tr>
<tr>
<td>Check User-unspecified Section</td>
<td>-check_section</td>
<td></td>
</tr>
<tr>
<td>Inhibit check for presence of user-unspecified section</td>
<td>-Xcheck_section</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for relative assemble list input directory</td>
<td>-alin</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output directory</td>
<td>-alout</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output</td>
<td>-als</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output module</td>
<td>-alsf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting absolute assemble list output</td>
<td>-Xals</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting ROM/RAM and ARRAY list</td>
<td>-alr</td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list output module</td>
<td>-alrf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting ROM/RAM and ARRAY list output</td>
<td>-Xalr</td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list symbol and address display position</td>
<td>-na/-an</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting external symbol cross-reference information list</td>
<td>-xl</td>
<td></td>
</tr>
<tr>
<td>Specification for external symbol cross-reference information list file name</td>
<td>-xlf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting external symbol cross-reference information list output</td>
<td>-Xxl</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting local symbol list</td>
<td>-sl</td>
<td></td>
</tr>
<tr>
<td>Specification for local symbol list file name</td>
<td>-slf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting local symbol list output</td>
<td>-Xsl</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting section detail map list</td>
<td>-ml</td>
<td></td>
</tr>
<tr>
<td>Specification for section detail map list file name</td>
<td>-mlf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting section detail map list output</td>
<td>-Xml</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting default option file read</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Option file read specification</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Function</td>
<td>Option</td>
<td>Remarks</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>---------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Specification for help message display</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for outputting version number/message</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for inhibiting version number/message</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>End message display specification</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for End message display inhibit specifcation</td>
<td>-Xcmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification to set end code to 1 when warning occurs</td>
<td>-cwno</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification to set end code to 0 when warning occurs</td>
<td>-Xcwno</td>
<td>* Common option</td>
</tr>
</tbody>
</table>
Attached Table E lists the librarian options.

### List of librarian options

**Table E  Attached Table E List of librarian options**

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module addition (registration)</td>
<td>-a</td>
<td></td>
</tr>
<tr>
<td>Module replacement (registration)</td>
<td>-r</td>
<td></td>
</tr>
<tr>
<td>Module deletion</td>
<td>-d</td>
<td></td>
</tr>
<tr>
<td>Module extraction</td>
<td>-x</td>
<td></td>
</tr>
<tr>
<td>List file output specification</td>
<td>-m</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for inhibiting list file output</td>
<td>-Xm</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting list file detail information</td>
<td>-dt</td>
<td>s, d, r, a</td>
</tr>
<tr>
<td>Specification of the number of lines on one list page</td>
<td>-pl</td>
<td>Default 60</td>
</tr>
<tr>
<td>Specification of the number of digits in one list line</td>
<td>-pw</td>
<td>Default 70</td>
</tr>
<tr>
<td>Creating backup file</td>
<td>-b</td>
<td></td>
</tr>
<tr>
<td>Inhibiting backup file creation</td>
<td>-Xb</td>
<td>Default</td>
</tr>
<tr>
<td>Library file content inspection</td>
<td>-c</td>
<td></td>
</tr>
<tr>
<td>File content optimization</td>
<td>-O</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting debugging information</td>
<td>-g</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting the output of debugging information</td>
<td>-Xg</td>
<td></td>
</tr>
<tr>
<td>Target CPU specification</td>
<td>-cpu</td>
<td>must be specified</td>
</tr>
<tr>
<td>Specification for inhibiting default option file read</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Option file read specification</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Help message display specification</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for outputting version number/message</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for inhibiting version number/message output</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>End message output specification</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for inhibiting end message output</td>
<td>-Xcmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification to set end code to 1 when warning occurs</td>
<td>-cwno</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification to set end code to 0 when warning occurs</td>
<td>-Xcwno</td>
<td>* Common option</td>
</tr>
</tbody>
</table>
APPENDIX F  LIST OF COMMANDS AND OPTIONS OF THE OBJECT FORMAT CONVERTER

Attached Table F.a lists the commands of the object format converter and Attached Table F.b lists the options of the object format converter.

- List of commands of the object format converter

<table>
<thead>
<tr>
<th>Command name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>f2ms</td>
<td>Absolute format load module → Motorola S format</td>
</tr>
<tr>
<td>f2is</td>
<td>Absolute format load module → Intel HEX format</td>
</tr>
<tr>
<td>f2es</td>
<td>Absolute format load module → Intel extended HEX format</td>
</tr>
<tr>
<td>m2is</td>
<td>Motorola S format → Intel HEX format</td>
</tr>
<tr>
<td>m2es</td>
<td>Motorola S format → Intel extended HEX format</td>
</tr>
<tr>
<td>i2ms</td>
<td>Intel HEX format → Motorola S format</td>
</tr>
<tr>
<td>e2ms</td>
<td>Intel extended HEX format → Motorola S format</td>
</tr>
<tr>
<td>m2ms</td>
<td>Motorola S format → Motorola S format</td>
</tr>
<tr>
<td>m2bs</td>
<td>Motorola S format → binary data (memory image)</td>
</tr>
</tbody>
</table>
### List of options of the object format converter

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification for outputting load module name</td>
<td>-o</td>
<td></td>
</tr>
<tr>
<td>Padding data specification</td>
<td>-p</td>
<td></td>
</tr>
<tr>
<td>Output range specification</td>
<td>-ran</td>
<td>S format binary converter only</td>
</tr>
<tr>
<td>Split mode specification</td>
<td>-sp</td>
<td>S format binary converter only</td>
</tr>
<tr>
<td>Specification for inhibiting split mode</td>
<td>-Xsp</td>
<td>S format binary converter only</td>
</tr>
<tr>
<td>Map list file creation specification</td>
<td>-m</td>
<td>S format binary converter only</td>
</tr>
<tr>
<td>Specification for inhibiting map list file creation</td>
<td>-Xm</td>
<td>S format binary converter only</td>
</tr>
<tr>
<td>Specification for inhibiting default option file read</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Option file read specification</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Help message display specification</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for version number/message display</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Inhibiting version number/message display</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>End message display specification</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for inhibiting end message display</td>
<td>-Xcmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification to set end code to 1 when warning occurs</td>
<td>-cwno</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification to set end code to 0 when warning occurs</td>
<td>Xcwno</td>
<td>* Common option</td>
</tr>
</tbody>
</table>
### Specification differences depending on the OS

#### Table G.a  Specification differences depending on the OS

<table>
<thead>
<tr>
<th>OS-dependent item</th>
<th>OS type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UNIX system OS</td>
</tr>
<tr>
<td></td>
<td>Windows 95</td>
</tr>
<tr>
<td></td>
<td>Windows 98</td>
</tr>
<tr>
<td></td>
<td>Windows Me</td>
</tr>
<tr>
<td></td>
<td>Windows NT</td>
</tr>
<tr>
<td></td>
<td>Windows 2000</td>
</tr>
<tr>
<td>Distinction of uppercase/lowercase characters in the file name</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Default extension of the file name</td>
<td>Lowercase character</td>
</tr>
<tr>
<td></td>
<td>Uppercase and lowercase characters are not distinguished</td>
</tr>
<tr>
<td>Working directory when no environmental variable TMP is not specified</td>
<td>/tmp</td>
</tr>
<tr>
<td></td>
<td>Current directory</td>
</tr>
<tr>
<td>File specification using the wildcard on the command line</td>
<td>Expanded by the shell and delivered to tools</td>
</tr>
<tr>
<td></td>
<td>Expanded inside tools</td>
</tr>
</tbody>
</table>

#### Table G.b  Differences in wildcard expansion

<table>
<thead>
<tr>
<th>Wildcard pattern</th>
<th>OS type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UNIX system OS</td>
</tr>
<tr>
<td></td>
<td>Windows 95</td>
</tr>
<tr>
<td></td>
<td>Windows 98</td>
</tr>
<tr>
<td></td>
<td>Windows Me</td>
</tr>
<tr>
<td></td>
<td>Windows NT</td>
</tr>
<tr>
<td></td>
<td>Windows 2000</td>
</tr>
<tr>
<td>?</td>
<td>Matches any one character</td>
</tr>
<tr>
<td></td>
<td>Matches the null character or any one character</td>
</tr>
<tr>
<td>*</td>
<td>Matches any character string</td>
</tr>
<tr>
<td></td>
<td>Matches any character string</td>
</tr>
</tbody>
</table>
### Table G.c Concrete examples of wildcard expansion

<table>
<thead>
<tr>
<th>Wildcard pattern</th>
<th>UNIX system OS</th>
<th>Windows 95</th>
<th>Windows 98</th>
<th>Windows Me</th>
<th>Windows NT</th>
<th>Windows 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a?.obj</code></td>
<td>al.obj matches.</td>
<td>al.obj matches.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a.obj does not match.</td>
<td>a.obj also matches.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>a*</code></td>
<td>al.obj, a.obj, and a.abs match.</td>
<td>al.obj, a.obj, and al.abs match.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>*</code></td>
<td>abcz and abc.z match.</td>
<td>abcz and abc.z match.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
INDEX

A
-a.........................................................95, 210
a backup file..................................219
absolute assemble list file...............170
absolute assemble list output............140
absolute assemble list output directory 139
absolute assemble list output inhibit 142
Absolute assemble list output module ....141
absolute format assemble list output....91
absolute format load module...........276, 277
absolute format load module output.....95
Adding (registering) a module...........210
address..............................................55, 106
-AL .....................................................114
-alin ...............................................138
allocation...........................................91
-alout ..............................................139
-alr ...................................................143
-alrf ..................................................144
-als ...................................................140
-alsf .................................................141
assemble list...............................171
assemble source list.....................176
automatic allocation.....................114
automatically locating section ..........69
B
-b.......................................................219
backup file.....................................220
C
-c.......................................................221
canceling the omission of name displayed
  in the list.......................................99
-check_locate....................................130
-check_rora........................................127
-check_section....................................135
-cmsg ..........................................10, 29, 37
command line.................................5, 26
common option..................................28
control list part.............................160, 161
converter........................................241, 286
converting..............................276, 277, 278, 280, 281, 282, 283
-cpu.................................86, 123, 206, 225
CPU information.........................85
CPU information file......................85
CPUs for which object may be mixed.....86, 206
creating and editing a library............209
cross-reference list.......................179
cwno ..............................................29, 39
D
d.......................................................212
data length in an output record.........270
debug data.......................................124
debug information delete..................94
debug information output..................93
debugging information...............56, 200, 204, 223, 224
default library retrieval inhibit........120
default option file.........................30, 46
deleting a module.........................212
df......................................................122
disk space.......................................234
dt......................................................99, 216
E
-e.......................................................121
e2ms ................................................283
-el......................................................118
derend code......................................39, 40
derend code value............................8
derend message.................................10, 37, 38
derend status .....................................8
diagnostics......................................171
disk space.......................................234
disk space.......................................234
disk space.......................................234
example display of memory used information list
  field list.....................................166
example of a search.......................75, 76, 77, 78, 79
example of location......................66, 67, 68, 71, 73
extended HEX format....................278, 283
external symbol cross-reference information
  list.........................................158, 180
external symbol cross-reference information list
  file name.................................148
external symbol cross-reference information list
  output......................................147
external symbol cross-reference information list
  output inhibit............................149
external symbol value....................60, 122
extracting a module.....................213
### INDEX

**O**
- **-O** .............................................................. 92, 245
- **-o** .............................................................. 92, 245
-object content list output ................................ 91
-object file ....................................................... 56
-object file converter ......................................... 240
-object file format ............................................. 105
-object format converter ..................................... 242, 244
-object tool ....................................................... 287
-OPT .................................................................... 14, 20
OPT896 .............................................................. 19
OPT907 .............................................................. 19
OPT911 .............................................................. 19
-optimizing the content of a file ......................... 222
-options ............................................................ 24
-option .............................................................. 24, 22, 25, 28, 208, 234, 244, 254, 268
-option file ........................................................ 32, 42, 43, 44, 45
-option parameters ............................................. 23
-other option ...................................................... 209
-output a list file ............................................... 214, 215
-output detailed information of a list file .............. 216
-output file name ............................................... 245
-output format .................................................... 56
-output list ........................................................ 91
-output load module file name ............................. 92
-output message ............................................... 91
-output module .................................................. 91
-output range .................................................... 255, 256, 269, 271
-output record ................................................... 269, 272
-output record data length .................................. 269
-outputting a list ................................................. 209

**P**
- **-p** .................................................................. 247
-pack link .......................................................... 110
-padding .................................................................. 247
-parameter .......................................................... 22
-**-pk** .................................................................. 110
- **-pl** .................................................................. 102, 217
-precaution .......................................................... 234
- **-pw** .................................................................. 101, 218

**R**
- **-r** .................................................................. 96, 211
-RA ................................................................. 58
- **-ra** .................................................................. 105
-RAM area .......................................................... 105
- **-ran** .................................................................. 255, 256, 269, 271
register bank area ................................................ 112
Relative assemble list input directory .................... 138
relative format load module output ...................... 96
Replacing (registering) a module ......................... 211
replacing (registering) a module ......................... 211
restriction ............................................................ 234, 286
retrieval library file ............................................. 116
- **-rg** .................................................................. 112
- **-ro** .................................................................. 104
- **ROM** ............................................................. 58
-ROM and RAM area ............................................. 81
-ROM area ............................................................ 104
-ROM/RAM and ARRAY list ............................... 174
-ROM/RAM and ARRAY list output ...................... 143
-ROM/RAM and ARRAY list output inhibit .......... 145
-ROM/RAM and ARRAY list output module ......... 144
-ROM/RAM and ARRAY list symbol and address display position .................................. 146
-ROM/RAM area name ......................................... 85

**S**
- **S format** ....................................................... 276, 280, 281, 282, 283
S format adjuster ................................................. 241, 264, 265, 268
S format binary converter ..................................... 254
S format binary converter ..................................... 241, 250, 286
- **-S1/-S2/-S3** .................................................. 269, 272
- **-sc** .................................................................. 106
searching a library file ......................................... 74
searching and protecting a file .............................. 209
-section ............................................................... 55, 58
-section allocation detailed information list .......... 184
-section allocation order ....................................... 106
-section combination order ................................... 106
-section combination order ................................... 106
-section combination attribute ............................. 62
-section content .................................................. 62
-section detail map list ......................................... 158
-section detail map list name ................................ 154
-section detail map list output .............................. 153
-section detail map list output inhibit .................. 155
-section group ..................................................... 109
-section identification .......................................... 63
-section information list ........................................ 178
-section location attribute .................................... 62
-section name ...................................................... 62
-section to be transferred from ROM to RAM ........ 82, 83
-section type ........................................................ 72
-section-placed-area ............................................ 130, 134
- **-set_rora** ....................................................... 125
shared combination of section ............................. 64
simple connection combination of section .............. 64
INDEX

-slf ................................................................. 151  
-sp ................................................................. 255, 257
split mode ................................................. 250, 255, 257, 258
startup message ........................................... 9, 56
symbol list part .............................................. 168, 169
symbol value ................................................ 55, 60

T

target CPU ................................................. 123, 225
terminatio ..................................................... 7
termination message ...................................... 57
TMP ............................................................. 14, 15

U

ter-user-specified-area .................................. 127, 129
user-unspecified section ............................... 135, 137
using the wild card ...................................... 189

V

-V ................................................................. 9, 29, 35
variable with initial value .............................. 190
version number .......................................... 35, 36

W

-w ................................................................. 103
warning ....................................................... 39, 40
warning check level ...................................... 56
warning message output level ....................... 103

X

-x ................................................................. 213
-Xalr ............................................................ 145
-Xals ............................................................ 142
-Xb ............................................................... 220
-Xcheck_locate ............................................. 134
-Xcheck_rora ............................................... 129
-Xcheck_section .......................................... 137
-Xcmd .......................................................... 29, 38
-Xcwno ........................................................ 29, 40
-Xdof ........................................................... 29, 30
-Xg ............................................................... 94, 224
-xl ............................................................... 147
-xlf .............................................................. 148
-Xm ............................................................. 98, 215, 255, 261
-Xml ............................................................ 155
-Xset_rora .................................................... 126
-Xsl ............................................................. 152
-Xsp ............................................................ 255, 258

-XV ............................................................... 29, 36
-Xxl .............................................................. 149