PREFACE

■ Objectives and Intended Readership

This manual describes the functions and operations of the Fujitsu SOFTUNE Linkage Kit operating on Windows 98, Windows Me, Windows NT 4.0, Windows 2000, Windows XP.

This manual is intended for engineers who are developing application programs using FR-V family microprocessor.

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

■ Trademarks

Microsoft 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 four parts:

PART I LINKAGE KIT

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

PART II LINKER

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

PART III LIBRARIAN

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

PART IV OBJECT FORMAT CONVERTERS

Explains the specifications, options, and function explanation of an object format converters.

APPENDIX

Explains the error messages of the linkage kit, HEX8, HEX16, HEX32 record format, and 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, such as descriptions of function and application circuit examples, in this document are presented solely for the purpose of reference to show examples of operations and uses of Fujitsu semiconductor device; Fujitsu does not warrant proper operation of the device with respect to use based on such information. When you develop equipment incorporating the device based on such information, you must assume any responsibility arising out of such use of the information. Fujitsu assumes no liability for any damages whatsoever arising out of the use of the information.

Any information in this document, including descriptions of function and schematic diagrams, shall not be construed as license of the use or exercise of any intellectual property right, such as patent right or copyright, or any other right of Fujitsu or any third party or does Fujitsu warrant non-infringement of any third-party’s intellectual property right or other right by using such information. Fujitsu assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein.

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

■ Page layout

In this manual, one section or subsection is presented on a single page or spread whenever possible. The reader can thus view an entire section without having to flip pages.

The contents of each section/subsection are summarized immediately below the title. You can obtain a rough overview of the product by reading through these summaries.

In each subsection, the section title appears at the top of the page. This helps you to navigate the manual, without having to go back to the Table of Contents or the chapter title page.

■ Product name abbreviation

In this manual, product names are abbreviated as follows:

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

Microsoft® Windows NT® Workstation operating system, Version 4.0 is abbreviated Windows NT.

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

Microsoft® Windows® XP Professional operating system is abbreviated Windows XP.
CONTENTS

PART I LINKAGE KIT ........................................................................................................  1

CHAPTER 1 SPECIFICATIONS OF LINKAGE KIT ..........................................................  3
  1.1 Outline of Linkage Kit ...............................................................................................  4
  1.2 Startup Procedure ...................................................................................................  5
  1.3 Forced Termination .................................................................................................  7
  1.4 End Code ................................................................................................................  8
  1.5 Startup Message ......................................................................................................  9
  1.6 End Message ........................................................................................................... 10
  1.7 Help Message ......................................................................................................... 11
  1.8 Identifiers .............................................................................................................. 12
  1.9 File name Rules ..................................................................................................... 13
  1.10 Environment Variables .......................................................................................... 14
      1.10.1 TMP (Work directory) ..................................................................................... 15
      1.10.2 FELANG (Message language) ......................................................................... 16
      1.10.3 FETOOL ....................................................................................................... 17
      1.10.4 LIB935 (Library file search directory) ............................................................ 18
      1.10.5 OPT935 ....................................................................................................... 19
      1.10.6 OPT ........................................................................................................... 20

CHAPTER 2 OPTIONS .................................................................................................... 21
  2.1 Option ................................................................................................................... 22
  2.2 Numeric Expression of Option Parameters .............................................................. 23
  2.3 Notes and Evaluation When Option Is Specified ...................................................... 24
  2.4 Specifying Options that Have Inclusive or Contradictory Relation Each Other .......... 25
  2.5 Example of Specifying Command Lines .................................................................. 26

CHAPTER 3 COMMON OPTIONS .................................................................................. 27
  3.1 List of Common Options .......................................................................................... 28
  3.2 Details of Common Options ...................................................................................... 29
      3.2.1 Specifying Suppression of Default Option File (-Xdof) ........................................ 30
      3.2.2 Specifying Reading from Option Files (-f) ......................................................... 31
      3.2.3 Specifying Help Message (-help) ................................................................. 32
      3.2.4 Specifying Version Number/Message Output (-V) ........................................... 33
      3.2.5 Suppression of Version Number/Message Output (-XV) .................................... 34
      3.2.6 Specifying Display of End Message (-cmsg) .................................................... 35
      3.2.7 Specifying Suppression to Display End Message (-Xcmsg) .............................. 36
      3.2.8 Specifying End Code to 1 When Warning is Issued (-cwno) ......................... 37
      3.2.9 Specifying End Code to 0 When Warning is Issued (-Xcwno) ....................... 38

CHAPTER 4 OPTION FILES ........................................................................................... 41
  4.1 Outline of Option File ............................................................................................. 42
CHAPTER 7  OUTPUT LIST FILE OF THE LINKER ..................................................... 143

7.1 Types of List Files Output by the Linker ......................................................... 144

7.2 Link List File ........................................................................................................ 145

7.2.1 Control List ....................................................................................................... 146

7.2.2 Map List ............................................................................................................ 148

7.2.3 Memory Used Information List .......................................................................... 150

6.2.10 Map List Output Inhibit Specification (-Xm) ..................................................... 100

6.2.11 Canceling the Omission of Names Displayed in the List (-dt) .......................... 101

6.2.12 Output Specification of the Map List File Memory Used Information List (-mmi) 102

6.2.13 Disable Output of Demangled Symbol Name Displayed in List (-Xdemangle) 103

6.2.14 Enable Output of Demangled Symbol Name Displayed in List (-demangle) 104

6.2.15 Specification of the Number of Digits in the List Line (-pw) ......................... 105

6.2.16 Specification of the Number of Lines on one List Page (-pl) ....................... 106

6.2.17 Warning Message Output Level Specification (-w) ....................................... 107

6.2.18 ROM Area Specification (-ro) ................................................................. 108

6.2.19 RAM Area Specification (-ra) ................................................................. 109

6.2.20 Section Allocation Order/Address Specification (-sc) ................................. 110

6.2.21 Section Group Specification (-gr) .............................................................. 113

6.2.22 Automatic Allocation Specification (-AL) .................................................... 114

6.2.23 Retrieval Library File Specification (-l) ...................................................... 116

6.2.24 Library Retrieval Path Specification (-L) .................................................. 117

6.2.25 Library Specification for Each Symbol (-el) ........................................... 118

6.2.26 Library Retrieval Inhibit Specification (-nl) ............................................ 119

6.2.27 Default Library Retrieval Inhibit Specification (-nd) ............................... 120

6.2.28 Entry Address Specification (-e) ............................................................. 121

6.2.29 Dummy Setting of External Symbol Values (-df) ..................................... 122

6.2.30 Target CPU Specification (-cpu) ............................................................. 123

6.2.31 Specifying CPU Information File (-cif) .................................................. 124

6.2.32 Inhibiting Check for Presence of Debug Data (-NCI0302LIB) .................. 125

6.2.33 Function that sets automatically internal ROM/RAM area (-set_rora) ........ 126

6.2.34 Specifies to prevent the internal ROM/RAM area from being set automatically (-Xset_rora) 127

6.2.35 Disable Pre-linking (-XPLNK) .............................................................. 128

6.2.36 Relative Assemble List Input Directory Specification (-alin) .................... 129

6.2.37 Absolute Format Assemble List Output Directory Specification (-alout) ..... 130

6.2.38 Absolute Format Assemble List Output Specification (-als) ...................... 131

6.2.39 Absolute Format Assemble List Output Module Specification (-alsf) ......... 132

6.2.40 Absolute Format Assemble List Output Inhibit Specification (-Xals) ........... 133

6.2.41 External Symbol Cross-reference Information List Output Specification (-xl) 134

6.2.42 External Symbol Cross-reference Information List File Name Specification (-xlf) 135

6.2.43 External Symbol Cross-reference Information List Output Inhibit Specification (-Xxl) 136

6.2.44 Local Symbol Information List Output Specification (-sl) ....................... 137

6.2.45 Local Symbol Information List File Name Specification (-slf) .................... 138

6.2.46 Local Symbol Information List Output Inhibit Specification (-Xsl) ............ 139

6.2.47 Section Detail Map List Output Specification (-ml) .................................. 140

6.2.48 Section Detail Map List File Name Specification (-mlf) ............................ 141

6.2.49 Section Detail Map List Output Inhibit Specification (-Xml) ..................... 142
CHAPTER 10 OPTIONS OF A LIBRARIAN ................................................................. 181
  10.1 List of Options of a Librarian ................................................................. 182
  10.2 Details of the Options of a Librarian ...................................................... 183
     10.2.1 Adding (Registering) a Module (-a) ................................................... 184
     10.2.2 Replacing (Registering) a Module (-r) ............................................... 185
     10.2.3 Deleting a Module (-d) ................................................................. 186
     10.2.4 Extracting a Module (-x) .............................................................. 187
     10.2.5 Specifying to Output a List File (-m) ................................................. 188
     10.2.6 Specifying not to Output a List File (-Xm) ........................................ 189
     10.2.7 Specifying to Output Detailed Information of a List File (-dt) .......... 190
     10.2.8 Specifying the Number of Lines Per Page of a List (-pl) ................ 191
     10.2.9 Specifying the Number of Columns Per Line of a List (-pw) .......... 192
     10.2.10 Creating a Backup File (-b) .......................................................... 193
     10.2.11 Inhibiting the Creation of a Backup File (-Xb) ................................. 194
     10.2.12 Checking the Contents of a Library File (-c) ..................................... 195
     10.2.13 Optimizing the Contents of a File (-O) ............................................. 196
     10.2.14 Specifying to Output Debugging Information (-g) ......................... 197
     10.2.15 Specifying not to Output Debugging Information (-Xg) ................. 198
     10.2.16 Specifying CPU Information File (-cif) ........................................... 199
     10.2.17 Specifying a Target CPU (-cpu) .................................................... 200

CHAPTER 11 LIST FORMATS OF A LIBRARIAN ..................................................... 201
  11.1 Contents of Information in a List File .................................................... 202
11.2 List of Module Names ................................................................. 203
11.3 Detailed Information of a Module ................................................. 204
11.4 External Defined and Reference Symbol Information in a Library .................. 205

CHAPTER 12 RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN 207
12.1 Restrictions on a Librarian ............................................................. 208
12.2 Questions and Answers on Using a Librarian ................................. 209

PART IV OBJECT FORMAT CONVERTERS ........................................... 211

CHAPTER 13 SPECIFICATIONS OF AN OBJECT FORMAT CONVERTER ........... 213
13.1 Outline of Object Format Converter ............................................... 214
13.2 Types of Object Format Converters ............................................... 216
13.3 Executing an Object Format Converter ........................................... 217

CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER .......... 219
14.1 List of Common Options of an Object Format Converter ...................... 220
14.2 Changing an Output File name (-o) .................................................. 221
14.3 Padding (-p) .................................................................................. 223

CHAPTER 15 LOAD MODULE CONVERTER(f2ms, f2hs, f2is, f2es) .................. 225
15.1 Outline of Load Module Converter ................................................... 226
15.2 List of Options of the Load Module Converter ..................................... 227
15.3 Details of Load Module Converter Options ........................................ 228
15.3.1 Output S format Option (-S1/-S2/-S3) ........................................... 229
15.3.2 Output HEX Format Option (-I16/-I20/-I32) ................................... 230
15.3.3 Specifying to Output Start Address Record (-entry) ......................... 231
15.3.4 Specifying not to Output Start Address Record (-Xentry) .................. 232
15.3.5 Specifying to Adjust (-adjust) ...................................................... 233
15.4 f2ms (Converting an Absolute Format Load Module into the S format) ............ 234
15.5 f2hs (Converting an Absolute Format Load Module into the HEX format) ........ 235
15.6 f2is (Converting an Absolute Format Load Module into the HEX8 format),
f2es (Converting an Absolute Format Load Module into the HEX16 format) ............ 236

CHAPTER 16 FORMAT ADJUSTER (m2ms, h2hs) ........................................ 239
16.1 Outline of the Format Adjuster ....................................................... 240
16.2 List of Options of the Format Adjuster .............................................. 243
16.3 Details of Options of the Format Adjuster .......................................... 244
16.3.1 Specifying the Data Length in an Output Record (-len) ..................... 245
16.3.2 Specifying the Output Range (-ran) ............................................. 246
16.3.3 Specifying an Output S format (-S1/-S2/-S3) ................................... 247
16.3.4 Specifying an Output HEX format (-I16/-I20/-I32) ......................... 249
16.3.5 Specifying Changes to the Starting Address (-ST) ......................... 250
PART I  LINKAGE KIT

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

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

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

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

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

■ Support Range of Linkage Kit

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

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

■ Command line format

To specify the command line (startup command syntax) of the SOFTUNE linkage kit,
- Specify the file name 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 file name. Refer to "Chapter 2 Options" for more details.

● Linker

```
flnk935s [ Option ] ... < File name >
```

The command name flnk935s differs depending upon the target CPU of the tool to be used.
Specify the object module file name to be input to <File name>.
Insert a space to specify two or more file names.
A wild card such as *.obj can also be used. Expanding the wild card of file name depends on the OS. Refer to "APPENDIX G Specification Differences Depending on the OS".
In linker, the target CPU must be specified using the -cpu option. Be sure to specify the -cpu option when executing the link processing.

● Librarian

```
flibs [ Option ] ... < File name >
```

The librarian is common in the SOFTUNE V6.
Specify the library file that is the target of editing to <File name>.
In librarian, the target CPU must be specified using the -cpu option. Be sure to specify the -cpu option when executing the library processing.
Object format converter

Command name [ Option ] ... < File name >

Determine the object type file name to <File name> based on the functions of each tool. Files of the following three types are the target format.

- Absolute type load module of linker output
- S format
- HEX format
### 1.3 Forced Termination

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

**Forced termination**

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

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

- End code value and end status

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

Table 1.4-1 End codes and end status of processing

<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 display a message when starting up in the default processing. If you want a message to be displayed during startup, use the -V option.

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

  ![Startup Message Example]

  FR-V Family SOFTUNE Linker V60Lxx
  ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 1998
  LICENSED MATERIAL - PROGRAM PROPERTY OF FUJITSU LIMITED
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 to Display End Message (-Xcmsg)".

■ End message

The end message shows tool names and errors.

Examples of the end message are shown below.

When errors do not occur

| Program name | COMPLITED FOUND NO ERROR |

When errors occur

| Program name | COMPLITED FOUND ERROR |
1.7 Help Message

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

- Command line description format
- List of options at startup

### Help message

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

#### Example of help message

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

![Example of help message](image)

**[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 such as creating program.
- File name
- Module name
- Option name
- Section name
- Group name
- ROM/RAM area name
- Symbol name (Mangle 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 file name depends on the OS being used. The module name that is created from the file name 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 is limitless.

### Mangle name
As for the identifier of the function name etc. which the C++ compiler generates, information to show type information etc. on the function is added. This thing is called mangle and the identifier is called mangle name.

The identifier such as labels treated in the linker is specified by using this mangle name.

### Displaying identifier name when outputting list
When the list is output, the mangle name does and displays the decipherment like being comprehensible.

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

Some of the longer identifier names only have the top 33 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 File name Rules

File name 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 file name is also set in the object modules.

■ Number of characters for the file name
The file name of the input/output files complies with the limited use of characters that are set for the OS.

■ Character code of the file name
The source file names of the C/C++ language and assembler are not only set as the source file name 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 Identifiers". Therefore, the file names that use Japanese characters or spaces must be modified specifying module name at the time of assembling.

● File name and characters that can be used for the file name (Windows Version)
Alphabetical letters, numbers and symbols except for the following:
\ / : ; * ? " < > |
When specifying a file name that includes spaces, enclose the file name with double quotations ("”).
When specifying a directory name including spaces as the environment variable, do not enclose the file name with double quotations ("’).
1.10 Environment Variables

Linkage kit supports the following six kinds of environment variables:

- **TMP**
  - TMP specifies the 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.

- **LIB935**
  - LIB935 specifies the directory in which the library is stored. Refer to "1.10.4 LIB935 (Library file search directory)" for more details.

- **OPT935**
  - OPT935 specifies the directory in which the default option files of the linker are stored. Refer to "1.10.5 OPT935 (Default option file storage directory)" for more details.

- **OPT**
  - OPT specifies a directory in which the default option file of the librarian and the object tool are stored. Refer to "1.10.6 OPT (Default option file storage directory)" 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/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 (Message language)

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 uses the code other than 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/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/C++ compiler and assembler)

[Example]

```
SET FETOOL=C:\Softune6
```

[Recommended directory structure]

```
\Softune6
  BIN Stores load module of linkage kit
  LIB Stores message file that does not depend on target CPU
  \935 Stores library file for FR- V 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".
1.10.4 LIB935 (Library file search directory)

Specify the directory that stores the library file or CPU information file for which the Linker searched the LIB935 (library file search directory).
The section below gives the description format, an explanation and an example of LIB935.

**LIB935 (Library file search directory)**

[Description format]

```
SET LIB935 = <Path name> [ ; <Path name> … ]
```

Specify the <path name> including the drive name.

[Explanation]

It specifies the directory in which the library files or CPU information file that linker searches are located.
Specify the directory in which the C/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 LIB935 differs depending upon the linker to be used.

[Example]

```
SET LIB935=C:\Softune6\LIB\935
```

[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/C++ library is searched even though the environment variable LIB935 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 LIB935.
3) The directory (%FETOOL%\LIB\935) that is directed by the environment variable FETOOL.
If the user creates the library, specify the paths while taking note of the order of searching with the C/ C++ library.
1.10.5 OPT935

OPT935 (default option file storage directory) specifies the directory in which the default option files of linker is stored. The description format, an explanation and an example of OPT935 are given below.

**OPT935**

[Description format]

```plaintext
SET OPT935 = < Path name >
```

Specify the <path name> including the drive name.

[Explanation]

It specifies the directory in which the default option files that linker uses is 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 developing environment directory are shown below.

- **Linker**

  `%FETOOL%\LIB\935\FLNK935. OPT`

[Example]

```plaintext
SET OPT935=C:\Softune6\LIB\935
```
1.10.6 OPT

Specify the directory that stores the librarian and object tools default option files to the OPT (default option file storage directory).

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 librarian and 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.

- **Librarian**
  - `%FETOOL%\LIB\FLIB.OPT`

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

[Example]

```
SET OPT=C:\softune6\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 Inclusive or Contradictory Relation Each Other
2.5 Example of Specifying Command Lines
2.1 Option

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

■ Synopsis of option

The following section is a synopsis of an option.

```
-Option name [Parameter] ...
```

Add a hyphen (-) to the top of the option name.
Insert a space to separate the option name from the parameter.
Whether the parameter is used or not used and the format of the parameter is defined in each option. Refer to the description of the respective options.
Pay attention to the following points when specifying an option.
• Capital letters and small letters of alphabetical letters must be distinguished when specifying option name.
• When a parameter needs an option, the parameters cannot be omitted entirely.
• When specifying two or more options, they cannot be specified as a group. For example, -a and -V as -aV is not acceptable.
• Spaces cannot be used in between hyphens and the option name.

■ Parameter

Parameters are used to specify a file name 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 \text{CODE}=0xC1000,\text{DATA}=0x1000
2.2 Numeric Expression of Option Parameters

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

■ Numeric Expression of Option Parameters

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

[Example]

0x100 ...Hexadecimal notation (= 256)
100 ...Decimal notation (= 0x64)
0xff and 0xFF are the same.
CHAPTER 2 OPTIONS

2.3 Notes and Evaluation When Option Is Specified

When specifying options, some options need duplicated specification and some need sequence to specify them.
In the linkage kit, the options are evaluated according to rules.

■ Notes and Evaluation When Specified Option

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

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

[Example]
- V : Specifying the message output
Duplicated specification like -V -V has no meaning and is error-free.

● Options that require parameters
When duplicated specification is required, there are different methods of evaluation as shown below.

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

[ex. Only the last option which is specified is valid]
-o file.abs : Specifying the output file
When options are specified two or more times like -o file.abs -o file.rel, the specification that is entered last becomes effective. (In this case, file.rel becomes effective.)

[ex. Order of specifying options has meaning and all specifications are effective]
-l lib1.lib -l lib2.lib : Specifying the library retrieval (linker)
When options are specified in order, such as -l lib2.lib -l lib1.lib, order of retrieving library is inverted.

[ex. Order of specifying options has no meaning, yet all specifications are effective]
-sc CODE=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.
2.4 Specifying Options that Have Inclusive or Contradictory Relation Each Other

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

■ Example of specifying an option that has an inclusive relation with other options

[Example]
Xm -pw 80 : Specifying suppression of outputting list and specifying page width

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

■ Example of specifying an option that has a contradictory relation with other options

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

[Example 1]
-a -r

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

[Example 2]
-m mapfile -Xm

Specifying a name of list file and suppression of list output -m are canceled so that list output is not executed.
2.5 Example of Specifying Command Lines

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

- Example of Specifying Command Lines

  [Example 1]
  
  flnk935s
  flnk935s 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]
  
  flibs sys.lib -m sys.mp2 ... *1
  flibs -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]
  
  flnk935s *.obj -g -o sample.abs
  flnk935s '*.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/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 3.1-1 lists common options that can be specified in any tool.

<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 File (-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 Files (-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 0 When Warning is Issued (-Xcwno)" for more details.
3.2.1 Specifying Suppression of Default Option File (-Xdof)

It cancels reading of default option file. When this option is not specified, default option file is always read.

### Specifying Suppression of Default Option File (-Xdof)

[Synopsis]

-Xdof

[Parameter]

None

[Description]

It cancels reading of default option file. When this option is not specified, default option file is always read. Refer to "4.5 Default Option File" for the option file.

[Note]

This option is valid when specified in the command line.

[Example]

flnk935s test.obj -Xdof -cpu MB93501
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.

**[Example 1]**

f2ms -V -f optfile.f2m

Contents of optfile.f2m

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

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

f2ms -V ccp903.abs -o ccp903.mhx

**[Example 2]**

flibs syslib.lib -f objfile.opt

Describe the module that is registered in syslib.lib to objfile.opt. The librarian creates a library file by referring to the contents of this file.
For example, contents of objfile.opt are as follows:

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

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

```
flibs -f libfile.opt
```

In this case, contents of libfile.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.

```
flibs 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.

■ Display of 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.

#### Specifying Version Number Message Output (-V)

**[Synopsis]**

```
-V
```

**[Parameter]**
None

**[Description]**

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

```
flnk935s 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  V60L02

ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 1992

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]**

```
flibs -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]

flnk935s ccp903
flnk935s 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 a program.

### Specifying Display of End Message (-cmsg)

**[Synopsis]**

- cmsg

**[Parameter]**

None

**[Description]**

It displays the end message of a program.

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

**[Example 1]**

```bash
flnk935s 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]**

```bash
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]**

flnk935s ccp903
flnk935s 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]

flnk935s ccp903 -cwno

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

[Example 2]

flnk935s ccp903

When a warning is issued during execution of program, the end code to OS remains 0.
### Specifying End Code to 0 When Warning is Issued (-Xcwno)

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

- **Summary**

  - **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**
    - `flnk935s ccp903`
    - `flnk935s 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 of the linkage kit.

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

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

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

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

  However, the following two items are added in option file.
  
  - Comment statement can be described.
  - Line feed is possible at any desired separating point.

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

  Comment statement and line feed symbol are handled equally as a space on the 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]**

  ```
  flibs -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 not limited.

  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.
4.2 Specification to Continue in the Option File

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

■ Specification to Continue in the Option File

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

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

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

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

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

```
-a mod01, mod02, obj03, obj04  ← the continued line
```
4.3 Specifying Comment in the Option File

Comment can be input into option file.

■ Specifying comment in the Option File

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

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

The underlined portion is comment.

```
# Example of Library Options
syslib.lib # INDICATES LIBRARY FILE
-a mod01, mod02, obj03, obj04 # Add Modules
```
### 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 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 option at system startup. This function is called default option file.

Default Option File

The default option file is one of the functions of option file. The previously specified option files can be read and executed without specifying -f option 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 -Xdof in order to suppress function of the default option. When this option is specified, default option file is not read.

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

Table 4.5-1 Names of default option files of the linkage kit

<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>flnk935s</td>
<td>flnk935.opt</td>
</tr>
<tr>
<td>Librarian</td>
<td>flibs</td>
<td>flib.opt</td>
</tr>
<tr>
<td>f2ms</td>
<td>f2m.opt</td>
<td></td>
</tr>
<tr>
<td>f2hs</td>
<td>f2h.opt</td>
<td></td>
</tr>
<tr>
<td>m2bs</td>
<td>m2b.opt</td>
<td></td>
</tr>
<tr>
<td>m2ms</td>
<td>m2m.opt</td>
<td></td>
</tr>
<tr>
<td>h2bs</td>
<td>h2b.opt</td>
<td></td>
</tr>
<tr>
<td>h2hs</td>
<td>h2h.opt</td>
<td></td>
</tr>
<tr>
<td>f2is</td>
<td>f2i.opt</td>
<td></td>
</tr>
<tr>
<td>f2es</td>
<td>f2e.opt</td>
<td></td>
</tr>
<tr>
<td>m2is</td>
<td>m2i.opt</td>
<td></td>
</tr>
<tr>
<td>m2es</td>
<td>m2e.opt</td>
<td></td>
</tr>
<tr>
<td>i2ms</td>
<td>i2m.opt</td>
<td></td>
</tr>
<tr>
<td>e2ms</td>
<td>e2m.opt</td>
<td></td>
</tr>
</tbody>
</table>

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

- When the environment variable OPT935 or OPT has already been set. The file in the directory that is set by the environment variable is referred to.
  - Linker
    \%
  - Librarian, Object tool
    \%
● When the environment variable OPT935 or OPT has not been set
  The default option file in the development environment directory is referred to.

  • Linker
    %FETOOL%\LIB\935\default option file
  • Librarian, Object tool
    %FETOOL%\LIB\default option file

  Note: When default option file cannot be found, the linkage kit does not issue error message.
PART II LINKER

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 describes the overview and functions of a linker.

5.1 Outline 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 Interleaved Objects on the Linker
5.1 Outline 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.

■ Outline 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/C++ compiler, you usually need to import library files in C and C++.

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

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

Figure 5.1-1 Relationship between a linker and input-output files
5.2 Functions of a Linker

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

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

Control on input-output files and messages

The following describes the overview of control and input-output files. For details, see “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 file 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

To output these files, the linker load module must be in the absolute format.

- 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.
CHAPTER 5 SPECIFICATIONS OF A LINKER

■ Control on combining and locating sections
The following is an overview of the control on combining and locating sections. For details, see "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 "5.2.3 Control on Searching Libraries".

- When a program is developed in C/C++, the runtime library in C/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 "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, it is 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/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/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 "5.9 Sections to be Transferred from ROM to RAM".
5.2.3 Control on Searching Libraries

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

- **Specifying a path to search a library**
  To specify a path to search a library, specify the directory containing the C/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/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 in an output load module.

- **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 5.3-1 shows the 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>
</tbody>
</table>

■ Section location attribute

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

<table>
<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 5.3-3 shows the 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 combination attribute of PUBLIC are connected through simple connection combination.

Figure 5.4-1 is an overview of the simple connection combination of the same sections in two object files. The entire size after combination is the total of the A-1 and A-2 sizes plus the gap size between A-1 and A-2 generated due to boundary adjustment.

![Figure 5.4-1 Simple connection combination of sections](image)

■ Shared combination of sections

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

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

Figure 5.4-2 is an overview of shared combination of the same sections in two object files. The entire size after combination is the larger one of the A-1 and A-2 sizes.

![Figure 5.4-2 Shared combination of sections](image)
5.5 Locating Sections

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

- Links of sections
  
  Only relative sections may be combined or located.
  Absolute sections may not be combined or located.
  Sections are combined or located in the following way:
  1) The same sections are collected from object modules.
  2) These sections are combined according to their combination attributes.
  3) Then the sections are located.

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

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

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

Example of location when the order of combining sections is not specified

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

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

Figure 5.5-1 Example of location when the order of combining sections is not specified
5.5.2 Example of Location when the Order of Combining Sections is Specified

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

■ Example of location when the order of combining sections is specified

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

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

Figure 5.5-2 Example of location when the order of combining sections is specified
5.5.3 Example of Location when the Section Group is Specified

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

Example of location when the section group is specified

If the group is specified, the sections to belong to each 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 5.5-3 Example of location when the section group is specified
5.6  Automatically Locating Sections

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

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

■ Automatically Locating Sections

This linker supports automatically locating of 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 "5.6.1 Automatically Locating Sections when -AL 1 is Specified", and "5.6.2 Automatically Locating Sections when -AL 2 is Specified".
5.6.1 Automatically Locating Sections when -AL 1 is Specified

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

Determining location addresses

The sections with the area name specified in the -sc option may be automatically located. Sections with larger alignment values are located before others. For sections with the same alignment values, ones with a larger size are located before others.

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

Table 5.6-1 Alignment values and sizes of sections

<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 5.6-1 is determined as follows:

1) Sections with an alignment value of 4 (code4 and code5) are located before those with an alignment value of 2 (code1, code2, and code3).

2) The code5 sections, being larger in size, are located before the code4 sections.

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

Table 5.6-2 Alignment values and sizes of sections

<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 (Table 5.6-1 and Table 5.6-3).

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

Figure 5.6-1 Option specification for a linker

Table 5.6-3 Contents of sections

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

As shown in this example, specify -AL 1 to have a linker optimally locate sections in the specified area so that they do not overlap with absolute sections and the available area is the smallest.
5.6.2 Automatically Locating Sections when -AL 2 is Specified

If -AL 2 is specified, a linker automatically locates sections without location specification into an available space of area.
Linken determines into which the location place of a section shell be made between ROM area or RAM area based on the type of a section in that case.
Section 5.6.2 describes the location destinations for each section type and the order of determining location addresses.
Figure 5.6-4 shows an example of location when -AL 2 is specified.

Section types and location destinations

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

Table 5.6-4 Section types and location destinations

<table>
<thead>
<tr>
<th>Location destination</th>
<th>FR-V family</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM area (specified in -ro)</td>
<td>CODE&lt;br&gt;CONST</td>
</tr>
<tr>
<td>RAM area (specified in -ra)</td>
<td>IO&lt;br&gt;DATA&lt;br&gt;STACK</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 5.6-5. As shown in Table 5.6-5, the order specified by the user is prioritized to the automatic location.
A linker always searches a place to locate a section, starting from low-order addresses.

Table 5.6-5 Section location destinations

<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>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>3</td>
<td>Section with a specified location area 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>4</td>
<td>Section without a specified location</td>
<td>The linker determines the location area according to Table 5.6-4. Then, in the determined area, the linker searches a place where it can locate the section without overlapping with another section. The linker then locates it.</td>
</tr>
</tbody>
</table>
Example of location when -AL 2 is specified

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

![Figure 5.6-3 Linker option specification](image)

<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>

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

![Figure 5.6-4 Example of location when -AL 2 is specified](image)
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/C++, a C/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/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 "6.2.20 Retrieval Library File Specification (-l)".

- **Specification in the -el option**
  - One or more library files may be specified.
  - Different library files may contain an external defined symbol with the same name. (Avoid such a status because it may cause a malfunction.)
  - The -el option is used to identify for each symbol a library file name in which to search a symbol. For details, see "6.2.22 Library Specification for Each Symbol (-el)".

### Order of searching a library file

A linker solves a symbol with the -el option specified, then searches libraries in the order specified in the -l option. Lastly, it searches the default library. This series of searches will be repeated until no more modules are imported from library files.
5.7.1 Example of a Search when there is one Library File (1)

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

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

As a result of combining Object Modules X and Y, the external reference symbol (S2) in X is solved by the external defined symbol {S2} in Y.

Since the external reference symbol (S1) is unsolved, the libraries are searched.

Module A, containing the external defined symbol (S1), is newly linked.

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

Although Module B in the library contains the external defined symbol (S2), it is in Module Y which is already linked. Therefore, (S2) will not be a target in the library search.

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

Figure 5.7-1 Example of a search when there is one library file (1)
5.7.2 Example of a Search when there is one Library File (2)

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

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

As a result of combining Object Modules 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 Module 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 5.7-2 Example of a search when there is one library file (2)
There are three examples of a search when there is one library file. Section 5.7.3 describes one of these examples as shown in Figure 5.7-3.

Example of a search when there is one library file (3)

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 Module 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 5.7-3 Example of a search when there is one library file (3)
There are two examples of a search when there are multiple library files. Section 5.7.4 describes one of these examples as shown in Figure 5.7-4.

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

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

After Module X and Y 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 5.7-4  Example of a search when there are multiple library files (1)**
5.7.5 Example of a Search when there are Multiple Library Files (2)

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

■ Example of a search when there are multiple library files (2)

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 5.7-5 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 5.7-6.

### Processing when library files are individually specified

- **Object X**
  - Library file 1
    - Symbol table
      - \{S1\} → A
      - \{S2\} → B
      - A
        - Definition \{S1\}
      - B
        - Definition \{S2\}
  - Library file 2
    - Symbol table
      - \{S2\} → C
      - C
        - Definition \{S2\}

  - Load module after linking
    - Extracted module
      - C
        - Definition \{S2\}
      - A
        - Definition \{S1\}

- 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 link 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 5.7-6 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.

ROM and RAM Areas Setting and Section Allocation

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 is located 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 area name to locate a section, this program can check whether the section is located in the specified address range.

Note: The ROM/RAM areas are automatically set using the following name from the CPU information file based on the MB number specified by the -cpu option.

  ROM areas _ROM_*_
  RAM areas _RAM_*_

These are used to set -sc options. See "5.10 CPU Information File".
5.9 Sections to be Transferred from ROM to RAM

When developing a program using a C/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/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 (0xC000) using a section name starting with an at sign, @.
  - If the section is thus located, INIT is processed as a section to be transferred from ROM to RAM, and the location is as shown in Figure 5.9-1. Then, the addresses are solved on RAM and the initial value data is located on ROM.
Figure 5.9-1 Example of locating a section to be transferred from ROM to RAM

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

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

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

■ Precautions on the sections to be transferred from ROM to RAM

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

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

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

```
.program sample
.section init, data
val1: .word 0x1234
val2: .word 0x5678
.section Data, data
val3: .res.w 0x1
.section Prog1, code
sethi.p #hi(val1), gr4
setlo #lo(val1), gr4
ldd @(gr4, gr0), gr6
.section Prog2, code
```
sethi.p  #hi(val3), gr4
setlo  #lo(val3), gr4
ldd  @(gr4, gr0), gr6

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

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

**Table 5.9-1 Attributes of sections after assembly**

<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>O</td>
<td>O</td>
</tr>
<tr>
<td>Data</td>
<td>×</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Prog1</td>
<td>O</td>
<td>O</td>
<td>×</td>
</tr>
<tr>
<td>Prog2</td>
<td>O</td>
<td>O</td>
<td>×</td>
</tr>
</tbody>
</table>

O...Enabled  ×...Disabled

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

**Table 5.9-2 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>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>init (ROM)</td>
<td>×</td>
<td>O</td>
<td>×</td>
<td>ROM section provided by the linker Write is disabled.</td>
</tr>
<tr>
<td>Data</td>
<td>×</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prog1 (RAM)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>Write is enabled.</td>
</tr>
<tr>
<td>Prog1 (ROM)</td>
<td>O</td>
<td>O</td>
<td>×</td>
<td>ROM section provided by the linker</td>
</tr>
<tr>
<td>Prog2</td>
<td>O</td>
<td>O</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

O...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
  - 935.csv

- Search Directory
  - %FETOOL%\LIB\935

■ Specifies to prevent the internal ROM/RAM area from being set automatically

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

Note: If the CPU information file cannot be found, or the appropriate MB number in the CPU information file does not exist, the linker will issue an error.
5.11 Interleaved Objects on the Linker

You can interleave objects generated using the SOFTUNE V5 language tool on the Linker (flnk935s).

**Interleaved Objects on the Linker**

You can interleave objects and libraries generated using the SOFTUNE V5 language tool (fasm935s or flibs) on the Linker (flnk935s).

The linker outputs information when there are objects interleaved only when "2" is specified by the -w option.

**Note:** The load module file that is output by flnk935s, outputs in a new file format that corresponds high-speed down loading. Therefore, this file cannot be input to the SOFTUNE V5 Debugger or SOFTUNE V5 load module converter.

New load module converters (f2ms, f2is, f2es) can process SOFTUNE V5 format load modules.
CHAPTER 6

LINKER OPTIONS

This chapter explains each linker option in more detail.

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

Options are available to specify linker operations in more detail.

- List of linker options

Table 6.1-1 shows the list of linker options.

Table 6.1-1 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>Input/output control options</td>
<td>Output load module file name specification</td>
<td>-o</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Output debug information specification</td>
<td>-g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Debug information delete specification</td>
<td>-Xg</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Specification of outputting absolute format load module</td>
<td>-a</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Specification of outputting relative format load module</td>
<td>-r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifying padding data</td>
<td>-p</td>
<td>Default 0</td>
</tr>
<tr>
<td></td>
<td>Specification for external symbol information output</td>
<td>-symtab</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specification for inhibiting the external symbol information output</td>
<td>-Xsymtab</td>
<td>Default</td>
</tr>
</tbody>
</table>

| Specification regarding output lists         | Map list file name specification              | -m     | Default   |
|                                              | Specification for inhibiting map list output  | -Xm    |           |
|                                              | Cancellation of omitting names displayed in the list | -dt |           |
|                                              | Output Specification of the Memory Used Information List | -mmi |           |
|                                              | Disable Output of Demangled Symbol Name        | -Xdemangle |           |
|                                              | Enable Output of Demangled Symbol Name         | -demangle | Default   |
|                                              | Specification of the number of digits in the list line | -pw | Default 80 |
|                                              | Specification of the number of lines on one list page | -pl | Default 0 |

| Specification regarding output messages      | Warning message output level specification    | -w     |           |

| Allocation/link options                      | ROM area specification                          | -ro    |           |
|                                              | RAM area specification                           | -ra    |           |
|                                              | Section allocation                                | -sc    |           |
|                                              | Section group specification                      | -gr    |           |
|                                              | Automatic allocation specification               | -AL    | Default 0 |
### Table 6.1-1 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></td>
<td></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>Specification for inhibiting default library retrieval</td>
<td>-nd</td>
<td></td>
</tr>
<tr>
<td>Other link control options</td>
<td></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>Target CPU specification</td>
<td>-cpu</td>
<td>need</td>
</tr>
<tr>
<td>Specifying CPU information file</td>
<td>-cif</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 areas</td>
<td>-set_rora</td>
<td>Default</td>
</tr>
<tr>
<td>Specifies to prevent the internal ROM/RAM areas from being set automatically</td>
<td>-Xset_rora</td>
<td></td>
</tr>
<tr>
<td>Disable Pre-linking</td>
<td>-XPLNK</td>
<td></td>
</tr>
<tr>
<td>Options regarding the absolute format assemble list output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for relative format assemble list input directory</td>
<td>-alin</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute format assemble list output directory</td>
<td>-alout</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute format assemble list output</td>
<td>-als</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute format 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>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 cross-reference information list output</td>
<td>-Xxl</td>
<td></td>
</tr>
<tr>
<td>Specification for local symbol list output</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 the local symbol list output</td>
<td>-Xsl</td>
<td></td>
</tr>
</tbody>
</table>
### Table 6.1-1 List of linker options (Continued)

<table>
<thead>
<tr>
<th>Options regarding the object content list output</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification for section detail map list output</td>
<td>-ml</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for section detail map list file name</td>
<td>-mlf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting section detail map list output</td>
<td>-Xml</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Common options</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification for inhibiting default option file read</td>
<td>-Xdof</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Option file read specification</td>
<td>-f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Help message display specification</td>
<td>-help</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for version number/message output</td>
<td>-V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting version number/message output</td>
<td>-XV</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>End message display specification</td>
<td>-cmsg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting end message display</td>
<td>-Xcmsg</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>Specification to set the end code to 1 when warning occurs</td>
<td>-cwno</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification to set the end code to 0 when warning occurs</td>
<td>-Xcwno</td>
<td>Default</td>
<td></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 "6.2.9 Map List File Name Specification (-m)" - "6.2.16 Specification of the Number of Lines on one List Page (-pl)".

- **Specification related to output messages**
  Details of options related to output messages in "6.2.17 Warning Message Output Level Specification (-w)".

- **Allocation/link options**
  Details of options related to the allocation/link in "6.2.18 ROM Area Specification (-ro)" - "6.2.22 Automatic Allocation Specification (-AL)".

- **Library control option**
  Details of options related to the library control in "6.2.23 Retrieval Library File Specification (-l)" - "6.2.27 Default Library Retrieval Inhibit Specification (-nd)".

- **Other link control options**
  Details of options related to other link controls in "6.2.28 Entry Address Specification (-e)" - "6.2.35 Disable Pre-linking (-XPLNK)".

- **Options related to the absolute format assemble list output**
  Details of options related to the absolute format assemble list output in "6.2.36 Relative Assemble List Input Directory Specification (-alin)" - "6.2.40 Absolute Format Assemble List Output Inhibit Specification (-Xals)".

- **Options related to the object content list output**
  Details of options related to the object content list output in "6.2.41 External Symbol Cross-reference Information List Output Specification (-xl)" - "6.2.49 Section Detail Map List Output Inhibit Specification (-Xxml)".
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.

<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>

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

[Example 1]

flnk935s putc.obj,getc.obj
A load module file is created with the file name putc.abs.

[Example 2]

flnk935s *.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]

flnk935s *.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]

`flnk935s *.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/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/C++ compiler and assembler to the linker.

[Example]
flnk935s -f rllnk.opt b1 b2

```
rlnk.opt
-r # relocatable LM output
-g # debug info.
-pw 100 # page width
-o rel1.rel # output file name
```
6.2.3 Debug Information Delete Specification (-Xg)

Object modules that are created by specifying the output of debug information in the C/ 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]

  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]

  flnk935s -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 file name
6.2.4 Absolute Format Load Module Output Specification (-a)

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

- Absolute format load module output specification (-a)

  [FORMAT]

  - a  (Default)

  [Parameters]
  None

  [Explanation]
  This option specifies the load module file output in the absolute format.
  Since the default output of the linker is in the absolute format, this option is normally not used. This option is used to cancel the -r 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]
  flnk935s a1 a2 a3 -r -o a123.abs -a
  The -r option in the middle of the command line is canceled.
  An output load module file is created in the absolute format.
6.2.5 Relative Format Load Module Output Specification (-r)

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

Relative format load module output specification (-r)

[FORMAT]

-r

[Parameters]
None

[Explanation]
This option specifies the load module file output in the relative format.
Specify this option when changing the default output (absolute format) of the linker.
If the -r option is specified after -a option, the -a option can be canceled.
A load module of the relative format has a format that gathers multiple object modules in one file without performing address resolution. A file of this format can be reentered in the linker, reducing the number of input files to be specified for the following link processing. However, if any change occurs in a module contained in the load module, it cannot be replaced with a library format file.
If this option is specified, all options related to the absolute format assemble list and object content list are ignored and their files are not output.
Output files are created with the following names.
- If the -o option is not specified
  Name of the input file specified first with the extension changed to ".rel".
- If the -o option is specified
  Specified name. If no extension is specified, ".rel" is added to the name.

[Example]
flnk935s 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 Padding Data Specification (-p)

This option is an option filled up with the value which specified the crevice between the objects generated in boundary adjustment etc.
This option is effective only when creating a absolute format load module.
It is invalid when creating a relative format load module.

### Padding Data Specification (-p)

**[FORMAT]**

```
- p <Value>  (Default: 0)
```

**[Parameters]**

- `<Value>`

**[Explanation]**

The value of the byte that fills the crevice between the objects which generated the absolute format load module file by section arrangement at the time of creation is directed.
The value of 0 to 255 can be specified.
In case section arrangement is performed by linker, several bytes of crevice where object data does not exist as shown in Figure 6.2-1 may occur according to conditions, such as boundary adjustment of a section.

![Figure 6.2-1 Example which the crevice generated by boundary adjustment of section locate](image)

This option is used when the crevice generated by section arrangement is filled up with a specific data value.
Linker fills the crevice where object data does not exist by 0, when this option is not specified.

**[Example]**

- `flnk935s a1 a2 a3 -p 255... A crevice is filled up with 255.`
- `flnk935s a1 a2 a3 -p 0xff... A crevice is filled up with 255.`
- `flnk935s a1 a2 a3 -p 0xaa... A crevice is filled up with 170.`
6.2.7 Specification for external symbol information output (-symtab)

This option is an option which directs to output external symbol information to an absolute load module. This option is effective only when creating a absolute format load module. It is invalid when creating a relative format load module.

- Specification for external symbol information output (-symtab)

  [FORMAT]
  
  - symtab

  [Parameters]
  None

  [Explanation]
  It directs to include external symbol information in a absolute format load module file.
  External symbol information is information which linker uses for solution of an external symbol value, and is information other than the debugging information which is needed in case it debugs by SOFTUNE Workbench.
  By the default of linker, external symbol information is not outputted to an absolute format load module file.

  [Example]
  flnk935s a1 a2 a3 -symtab
6.2.8 Specification for inhibiting the external symbol information output (-Xsymtab)

This option is an option which deters outputting external symbol information to an absolute format load module. This option is effective only when creating an absolute format load module. It is invalid when creating a relative format load module.

- Specification for inhibiting the external symbol information output (-Xsymtab)
  
  [FORMAT]
  
  - Xsymtab

[Parameters]
None

[Explanation]
It directs not to output external symbol information in an absolute format load module file. External symbol information is information which linker uses for solution of an external symbol value, and is information other than the debugging information which is needed in case it debugs by SOFTUNE Workbench. It is the default of linker. This option is used to cancel the -symtab option.

[Example]
flnk935s a1 a2 a3 -symtab -Xsymtab
6.2.9 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]

flnk935s a1 a2 a3 -r -o a123.rel -m a123.map
The map list file name output by the linker is changed to a123.map.
6.2.10 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]
flnk935s a1 a2 a3 -r -o a123.rel -Xm
Creating a map list file is inhibited.
6.2.11 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 File".

- Canceling the omission of names displayed in the list (-dt)

  [FORMAT]

  - dt

  [Parameters]
  None

  [Explanation]
  Symbol names or section names displayed in the map list or object content list are displayed without omission. In such cases, one symbol name or section name is displayed in several lines.
  By default, about 80 characters can be displayed. If a small number is set as the number of characters to be displayed in one list line, the number of characters that can be displayed decreases accordingly. In such cases, symbol names or section names may be displayed with some parts omitted.

  [Example]
  flink935s 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.
6.2.12 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.

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

[FORMAT]

- mmi

[Parameters]
None

[Explanation]
This outputs the memory used information list that indicates the usage conditions of the ROM or RAM areas specified by the options of the map list files that are output by the linker as the default. The information, such as the usable area, used area, position of gap in area, size, is displayed.

[Example]
flnk935s 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.
6.2.13 Disable Output of Demangled Symbol Name Displayed in List (-Xdemangle)

This option disables the display of a demangled symbol name of the external symbol name displayed in the symbol list part in the linker’s map list.

- Demangled symbol name
  A symbol name mangled in the C++ compiler is used as a template function name so as not to coincide with other symbol names. The demangled symbol name is a user-identifiable symbol name into which the mangled symbol name is converted.

[Example]
  flnk935s a1 a2 a3 -o a123.abs -m a123.abs -Xdemangle
6.2.14 Enable Output of Demangled Symbol Name Displayed in List (-demangle)

This option enables the display of a demangled symbol name of the external symbol name displayed in the symbol list part in the linker’s map list.

■ Enable Output of Demangled Symbol Name Displayed in List (-demangle)

[FORMAT]

| - demangle | (Default) |

[Parameters]

None

[Explanation]

This option enables the display of a demangled symbol name of the external symbol name displayed in the symbol list part in the linker’s map list.

- Demangled symbol name

A symbol name mangled in the C++ compiler is used as a template function name so as not to coincide with other symbol names. The demangled symbol name is a user-identifiable symbol name into which the mangled symbol name is converted.

[Example]

flnk935s a1 a2 a3 -o a123.abs -m a123.abs -demangle
6.2.15 **Specification of the Number of Digits in the List Line** (-pw)

By default, up to 80 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.

<table>
<thead>
<tr>
<th>Specification of the number of digits in the list line (-pw)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[FORMAT]</strong></td>
</tr>
<tr>
<td>- pw &lt; Number of digits &gt;   (Default : 80)</td>
</tr>
</tbody>
</table>

**[Parameters]**

<Number of digits>

Number of digits to be displayed in one line. Specify the number in the range of 80 to 1023.

**[Explanation]**

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

If this value is not specified, 80 is set.

If the values 70 to 79 are specified, the linker sets the number of digits to 80 after outputting I0311L.

**[Example]**

```
flnk935s a1 a2 a3 -o a123.abs -m a123.map -dt -pw 100
```

The number of digits to be displayed in one list line is set to 100.
6.2.16 Specification of the Number of Lines on one List Page (-pl)

By default, no limit 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 : 0 )

**[Parameters]**

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

**[Explanation]**

The number of lines on one page of the link list file and object content list is specified. If this value is not specified, 0 is set. If 0 is specified, page control is canceled.

**[Example]**

flnk935s  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.
6.2.17 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 Linkage Kit."

[Explanation]

flnk935s a1 a2 a3 -o a123.abs -w 2 -Xm

All messages are output.
Addressing for section allocation can be simplified by defining the ROM area to be used by a program. Program size checking is also enabled.

### ROM area specification (-ro)

**[Format]**

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

**[Parameters]**

- **<Area name>**
  Name to indicate the address area to be set
- **<Start address>**
  Start address of the address area to be set
- **<End address>**
  End address of the address area to be set

**[Explanation]**

The ROM areas are defined. As many areas as required can be defined.
Specify the start address and end address, then name the area.
The area name defined by this option is used in the section allocation option.
The definition of the -ro option alone does not affect linker operation. Be sure to use the area name defined here for the addressing parameter in the section allocation option.

**[Example]**

```
flnk935s *.obj -o ap.abs -ro RomA=0x1000/0x2FFF -sc code=RomA ...
```

Section allocation of the section name code to the address 0x1000 to 0x2FFF is specified.
The section is allocated starting with the address 0x1000. A warning message is output if the address 0x2FFF is exceeded.
6.2.19 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 RAM areas are defined. As many areas as required can be defined. Specify the start address and end address, then name the area. The area name defined by this option is used in the section allocation option. The definition of the -ra option alone does not affect linker operation. Be sure to use the area name defined here for the addressing parameter in the section allocation option.

  **[Example]**
  
  flnk935s *.obj -o ap.abs -ra RamD=0x0100/0x01FF -sc data=RamD ...

  The section allocation of the section name data to the address 0x0100 to 0x01FF is specified. The section is allocated starting with the address 0x0100, and a warning message is output if the address 0x01FF is exceeded.
6.2.20 Section Allocation Order/Address Specification (-sc)

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

■ Section allocation order/address specification (-sc)

[Format]

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

[Parameters]

<Section name list>
The wild card can be used to specify the section name, section group name, or list section name.
When specifying multiple names, link them with the + symbol.

<Content type>
code, data, stack, const, IO

<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]

flnk935s *.obj -o ap.abs -sc 
"*/code"=0xC1000,D=0x1000 ...

Figure 6.2-2 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.

![Figure 6.2-2 Section allocation example 1](image)

[Example 2]

flnk935s *.obj -o ap.abs -ro ROM=0xC1000/0xCFFFF
-ra RAM=0x1000/0x13FF -sc P+code+text1=ROM,D=RAM

Figure 6.2-3 shows the allocation example in this case.
If section allocation is specified using the ROM/RAM option, the allocation end address can be checked.

![Figure 6.2-3 Section allocation example 1](image)
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><em>-sc &quot;</em>/code&quot;</td>
<td>ab_1, code_1, XXsect, etc</td>
<td>Match with all sections whose content type is code.</td>
</tr>
<tr>
<td>Mismatch</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><em>*-sc &quot;ab</em>*/code&quot;</td>
<td>ab_1, ab_XX, ab_, etc</td>
<td>Match with all sections whose content type is code and whose first three characters are &quot;ab_&quot;.</td>
</tr>
<tr>
<td>Mismatch</td>
<td>aab_XX, ab, etc</td>
<td></td>
</tr>
<tr>
<td>*-_ <em>-sc &quot;</em>_1/code&quot;</td>
<td>ab_1, XX_1, _1, etc</td>
<td>Match with all sections whose content type is code and whose last two characters are &quot;_1&quot;.</td>
</tr>
<tr>
<td>Mismatch</td>
<td>ab_11, _, etc</td>
<td></td>
</tr>
<tr>
<td><em>*-sc &quot;ab</em>*_1/code&quot;</td>
<td>ab_XX_1, ab_1</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>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).
6.2.21 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, IO

[Explanation]
Sections to be grouped and the order of section 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]

flnk935s *.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]

flnk935s *.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.
6.2.22 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.

[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 Automatically Locating Sections".
[Example]

flnk935s -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.

**Figure 6.2-4 Example of link map**

<table>
<thead>
<tr>
<th>S_Addr. -E_Addr.</th>
<th>Size</th>
<th>Section</th>
<th>Type</th>
<th>Al</th>
<th>Sec.(Top 29)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001000-0000100F</td>
<td>00000010</td>
<td>CODE</td>
<td>P</td>
<td>R-XI</td>
<td>REL</td>
<td>code2</td>
</tr>
<tr>
<td>00001010-00001017</td>
<td>00000008</td>
<td>CODE</td>
<td>N</td>
<td>R-XI</td>
<td>ABS</td>
<td>AbsSec</td>
</tr>
<tr>
<td>00001018-00001047</td>
<td>00000030</td>
<td>CODE</td>
<td>P</td>
<td>R-XI</td>
<td>REL</td>
<td>code3</td>
</tr>
<tr>
<td>00001048-0000105F</td>
<td>00000018</td>
<td>CODE</td>
<td>P</td>
<td>R-XI</td>
<td>REL</td>
<td>code1</td>
</tr>
</tbody>
</table>

**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)
- Normally, the linker will automatically set -ra and -ro from the CPU Information File.
- If -w 2 is specified and the automatic allocation function works, a message is output.
6.2.23 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.
1. The directory specified by the -L option.
2. The directory specified in the environmental variable LIB935.
3. 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 LIB935.
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]

flnk935s *.obj -o ap.abs -l ..\lib\com.lib,libu
flnk935s *.obj -o ap.abs -l "p*.lib"
All library files whose file name’s first character is “p” are retrieved.
6.2.24 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 LIB935 so that this option need not be specified.
The C libraries attached to the C/C++ compiler are stored in the path specified by the environmental variable LIB935. 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 LIB935 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]

flnk935s *.obj -o ap.abs -L C: \usr\usrlib -l com.lib,libu
6.2.25 Library Specification for Each Symbol (-el)

The library file to be used for the resolution of external reference symbol value 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]

flnk935s *.obj -o sp.abs -L C:\usr\usrlib -l libu,sublib -el sym1=sublib
6.2.26 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]
flnk935s -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]
flnk935s -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.
6.2.27 Default Library Retrieval Inhibit Specification (-nd)

The default library is a library presumed to be used by the C/C++ compiler and its library file names are set in the object file. This option instructs not to retrieve the default library file.

■ Default library retrieval inhibit specification (-nd)

[Format]

- nd

[Parameters]

None

[Explanation]

The specification of the default library file is canceled, therefore, it is not retrieved.

[Example]

flnk935s -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.
### 6.2.28 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]**

```
flnk935s *.obj -o ap.abs -e ProgStart
```
6.2.29 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**

flnk935s -L/usr/lib -l libu,sublib *.obj -o ap.abs -df Sym1=100

If Sym1 is not defined, 100 is set as its value.
6.2.30 Target CPU Specification (-cpu)

The target CPU is specified. The target CPU of programs to be linked is specified using the MB number.

**Target CPU specification (-cpu)**

**[Format]**

```
   - cpu < MB number>
```

**[Parameters]**

<MB number>
MB number of the target CPU

**[Explanation]**

The target CPU of programs to be linked is specified using the MB number.

**[Example]**

```
flnk935s *.obj -o ap.abs -cpu MB93501
```

**Note:** When executing link processing, the target CPU must be specified using this option.
6.2.31 Specifying CPU Information File (-cif)

This specifies the CPU information file used by the linker.

**CPU Information File Specification (-cif)**

**[Format]**

- cif <CPU information file name>

**[Parameters]**

<CPU information file name>
CPU information file used by linker

**[Explanation]**

This specifies the CPU information file used by linker.

**[Example]**

flnk935s *.obj -o ap.abs -cpu MB93501 -cif C:\Softune6\lib\935\MB93501.csv

**Note:** SOFTUNE Tools get CPU information by referencing the CPU information file. Reference to a CPU information file different between the related tools may cause an error to the program to be created. The CPU information file that comes standard with SOFTUNE Tools is located at:

Installation place directory\lib\935\935.csv

When installing the compiler and assembler pack in different directory, specify -cif so that each tool can refer the same CPU information file.
6.2.32 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.

**Inhibiting Check for Presence of Debug Data (-NCI0302LIB)**

[Format]

```
- NCI0302LIB
```

[Parameters]

None

[Explanation]

When debug data output (-g) and warning level 2 (-w 2) 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]

```
flnk935s -cpu MB93501 -g -w 2 test .obj -I lib935.lib
*** I0302L: Debug information not exist (C:\Softune6\lib935\lib935.lib)
*** I0302L: Debug information not exist (C:\Softune6\lib935\lib935.lib)
.
.
.
flnk935s -cpu MB93501 -g -w 2 test .obj -I lib935.lib -NCI0302LIB
Information 'I0302L' is not outputted.
```
Function that sets automatically internal ROM/RAM area (-set_rora)

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

Function that sets automatically internal ROM/RAM areas (-set_rora)

[Format]

- set_rora

[Parameters]
None

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

The linker, when this option is specified, refers to CPU information file and automatically sets the internal ROM/RAM areas of the appropriate chip.

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

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

These names are used by the -sc options.

[Example]
flnk935s *.obj -o ap.abs -cpu MB93501 -set_rora
6.2.34 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 referred to the CPU information file.

- Specifies to prevent the internal ROM/RAM area from being set automatically (-Xset_rora)
  
  **Format**
  
  ```plaintext
  - Xset_rora
  ```
  
  **Parameters**
  
  None
  
  **Explanation**
  
  Prevents the information from being set regarding the internal ROM/RAM area of the targeted CPU that is referred to the CPU information file.
  
  **Example**
  
  `flnk935s *.obj -o ap.abs -cpu MB93504 -Xset_rora`
6.2.35 Disable Pre-linking (-XPLNK)

This option disables pre-linking used for template function processing.

■ Disable Pre-linking (-XPLNK)

[Format]

- XPLNK

[Parameters]
None

[Explanation]
To disable an increase in code size by the template function for the C++ program, the linker starts the pre-linker to perform pre-linking before linking.
This option disables pre-linking.

[Example]
flnk935s a1 a2 a3 -o a123.abs -m a123.abs -XPLNK
6.2.36 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 format 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**
  - flnk935s *.obj -o ap.abs -alin d:\fr500 -alf swctr1.lst,mstdef.lst
  - flnk935s *.obj -o ap.abs -alsf d:\fr500\swctrl.lst,d:\fr500\mstdef.lst
  - The above two examples share the same meaning.
6.2.37 Absolute Format Assemble List Output Directory Specification (-alout)

The directory which outputs an absolute assemble list file is specified.

- **Absolute format 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]**

  flnk935s *.obj -o ap.abs -alin d:\fr500 -alf swctrl.lst,mstdef.lst -alout d:\fr500\als
6.2.38 Absolute Format Assemble List Output Specification (-als)

The output of absolute assemble list files is specified.
This is an instruction for all object modules.

- Absolute format 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]
  flnk935s *.obj -o ap.abs -als
6.2.39 Absolute Format Assemble List Output Module Specification (-alsf)

Modules to output absolute format assemble list files are specified. Selective output for object modules is instructed.

Absolute format 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 format assemble list.
The wild card can be used to specify the file name.

[Explanation]

Modules to create the absolute format 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 format 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]

flnk935s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst
flnk935s *.obj -o ap.abs -alsf swctrl.lst -alsf mstdef.lst
flnk935s *.obj -o ap.abs -alsf swctrl -alsf mstdef

The above three examples share the same meaning.
6.2.40 Absolute Format Assemble List Output Inhibit Specification (-Xals)

The inhibition of creating the absolute format assemble lists is instructed for all modules.

Absolute format assemble list output inhibit specification (-Xals)

[Format]

- Xals (Default)

[Parameters]

None

[Explanation]

The inhibition of creation for absolute format 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]

flnk935s *.obj -o ap.abs -alf sectrl.lst,mstdef.lst -nl
6.2.41 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]

flnk935s *.obj -o ap.abs -xl
6.2.42 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" is used as the output file name.

  **Example**
  
  `flnk935s *.obj -o ap.abs -xl -xlf ccp903.mp`x
  `flnk935s *.obj -o ap.abs -xlf ccp903`
  
  The above two examples share the same meaning.
6.2.43 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]
flnk935s *.obj -o ap.abs -xl -Xxl
flnk935s *.obj -o ap.abs -Xxl
flnk935s *.obj -o ap.abs

The above three examples share the same meaning.
6.2.44 Local Symbol Information List Output Specification (-sl)

The output of the local symbol information list file is specified. The output of debug information when compiling, assembling, and linking 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]

flnk935s *.obj -o ap.abs -sl -g
6.2.45  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" is used as the output file name.

[Example]
flnk935s *.obj -o ap.abs -sl -slf ccp903.mps -g
flnk935s *.obj -o ap.abs -slf ccp903 -g
The above two examples share the same meaning.
6.2.46 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]
  flnk935s *.obj -o ap.abs -sl -Xsl
  flnk935s *.obj -o ap.abs -Xsl
  flnk935s *.obj -o ap.abs
  The above three examples share the same meaning.
6.2.47 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]
  flnk935s *.obj -o ap.abs -ml
6.2.48 Section Detail Map List File Name Specification (-mlf)

This option is used to change the output destination directory or file name of the section detail map list file.

■ Section detail map list file name specification (-mlf)

[Format]
-mlf <Output file name>

[Parameters]

<Output file name>
Specify the output file name. To change the directory of the output destination, add the path name prior to the output file name.

[Explanation]
The section detail map list file is created with the specified name.
When using this option, the -ml option can be omitted.
If the extension is omitted in the <output file name> specification, the default extension ".mpm" is added.
If this option is not specified, the absolute format load module file name is used whose extension is changed to ".mpm" is used as the output file name.

[Example]

flnk935s *.obj -o ap.abs -ml -mlf ccp903.mpm
flnk935s *.obj -o ap.abs -mlf ccp903

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

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

Section detail map list output inhibit specification (-Xml)

[Format]

- Xml ( Default )

[Parameters]
None

[Explanation]
Inhibiting the output of the section detail map list file is specified. This option is a default option, therefore, it is not necessary to specify it. -ml and -mlf specified before are canceled.

[Example]
flnk935s *.obj -o ap.abs -ml -Xml
flnk935s *.obj -o ap.abs -Xml
flnk935s *.obj -o ap.abs

The above three examples share the same meaning.
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 Format 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 and input module name when starting up the Linker, the section and memory use status information after the module link and the external symbol information.

- **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 in the absolute format.
  This list can be referenced when debugging on the assembler language level, and the addresses in each step of the machine language that are unidentifiable in the link list can be known.

- **External symbol cross-reference information list**
  The external symbol cross-reference information list outputs information about external definition symbols of each module after linking and inter-module cross-reference of external reference symbols.

- **Local symbol information list**
  The local symbol information list outputs information about variables and functions including local symbols of each module after linking.

- **Section detail map list**
  The section detail map list creates information about the section allocation of each module after linking.
7.2 Link List File

The link list file can be divided into the following four parts depending on the information contents.
- Control list
- Map list
- Memory used information list
- External symbol list

We explain here the items output in each list.

Configuration of Link List File

The link list file can be divided into the following four 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>Figure 7.2-1 List output format of control list part</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR-V Family SOFTUNE Linker  Control List  YYYY-MM-DD hh:mm:ss  Page:  1</td>
</tr>
<tr>
<td>Option File(s)</td>
</tr>
<tr>
<td>(1) Option file name display area</td>
</tr>
<tr>
<td>Control(s)</td>
</tr>
<tr>
<td>(2) Linker control option display area</td>
</tr>
<tr>
<td>Input Module(s)</td>
</tr>
<tr>
<td>(3) Input module display area</td>
</tr>
<tr>
<td>Error(s)</td>
</tr>
<tr>
<td>(4) Error message display area</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.

**Figure 7.2-2 List display example of control list part**

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

![Figure 7.2-3 List output format of the map list file](image)

Page header

The linker name, list name, date and time, and page number are displayed in the first line.

(1) Map information display area

Map information is displayed in order of start address, or if the start address is the same, in order of section occurrence.

S.Addr : Section start address (hexadecimal)
E.Addr : Section end address (hexadecimal)
Size : Section size (hexadecimal)
Section : Section content type

The section content type is displayed.

- CODE Program section
- DATA Data section
- CONST Data section with initial values
- STACK Stack section
- 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

AI: 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.

**Note:** The end address of a section with the section size 0 is displayed by [........].

---

**List display example of the map list part**

The following shows a list display example of the map list part.

![List display example of map list part](image)

<table>
<thead>
<tr>
<th>S_Addr. -E_Addr.</th>
<th>Size</th>
<th>Section</th>
<th>Type</th>
<th>Al</th>
<th>Sec.(Top 29)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000-0000001F</td>
<td>00000020</td>
<td>DATA</td>
<td>P</td>
<td>RW--</td>
<td>08</td>
<td>REL</td>
</tr>
<tr>
<td>00000020-00000039</td>
<td>0000001A</td>
<td>DATA</td>
<td>P</td>
<td>RW--</td>
<td>02</td>
<td>REL</td>
</tr>
<tr>
<td>0000003A-00000053</td>
<td>0000001A</td>
<td>STAK</td>
<td>P</td>
<td>RW--</td>
<td>02</td>
<td>REL</td>
</tr>
<tr>
<td>00000054-0000006D</td>
<td>0000001A</td>
<td>DATA</td>
<td>P</td>
<td>RW-I</td>
<td>02</td>
<td>REL</td>
</tr>
<tr>
<td>00008000-00008039</td>
<td>0000003A</td>
<td>CODE</td>
<td>P</td>
<td>R-XI</td>
<td>02</td>
<td>REL</td>
</tr>
<tr>
<td>0000803A-00008053</td>
<td>0000001A</td>
<td>CODE</td>
<td>P</td>
<td>R-XI</td>
<td>02</td>
<td>REL</td>
</tr>
<tr>
<td>00008054-0000806D</td>
<td>0000001A</td>
<td>DATA</td>
<td>P</td>
<td>R--I</td>
<td>02</td>
<td>REL</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 over 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.

![Figure 7.2-5 Memory used Information list field output format](image)

Page Header
Linker name, list name, date and 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 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
- **-**: Over Area Size
- **Space**: When free/over area is 0.
- **Name/State**: Memory Area Name and Section Name
(2) Area Internal Information Area

The area internal information area displays the free area of the valid area, the used area and the gap area in a map image. The information displayed in this area is information only of the located section with regard to the specified area.

- **S_Addr**: Area Start Address (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 over 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.

**Figure 7.2-6 Example display of the memory used Information list field list**

<table>
<thead>
<tr>
<th>S_Addr.</th>
<th>E_Addr.</th>
<th>Size</th>
<th>Remain</th>
<th>Name/State</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001000-00001FFF</td>
<td>00001000</td>
<td>---------</td>
<td>RAM1</td>
<td>00001000-00001774 00000775 +0000088B</td>
<td></td>
</tr>
<tr>
<td>00001000-00001774</td>
<td>00000775</td>
<td>+0000088B</td>
<td>00001136-00001137 00001137</td>
<td>GAP</td>
<td></td>
</tr>
<tr>
<td>00001138-0000152B</td>
<td>000003F4</td>
<td>USED</td>
<td>0000152C-0000152F 00000004</td>
<td>GAP</td>
<td></td>
</tr>
<tr>
<td>00001530-00001774</td>
<td>00000245</td>
<td>USED</td>
<td>00001775-00001FFFF 0000088B</td>
<td>FREE</td>
<td></td>
</tr>
<tr>
<td>00002000-000023FF</td>
<td>00000400</td>
<td>FREE</td>
<td>** Not Locate ** 00000B28</td>
<td>data01</td>
<td></td>
</tr>
<tr>
<td>00002000-000023FF</td>
<td>00000400</td>
<td>FREE</td>
<td>** Not Locate ** 00000EF6</td>
<td>data02</td>
<td></td>
</tr>
</tbody>
</table>

**Memory Area Specified in Option**

**Memory Status of the Section Located**

**Section Information Not Located (**2)**

<table>
<thead>
<tr>
<th>S_Addr.</th>
<th>E_Addr.</th>
<th>Size</th>
<th>Remain</th>
<th>Name/State</th>
</tr>
</thead>
<tbody>
<tr>
<td>000BC000-000BFFFF</td>
<td>00001000</td>
<td>---------</td>
<td>ROM1</td>
<td></td>
</tr>
<tr>
<td>000BC000-000BFFFF</td>
<td>00001000</td>
<td>00000000</td>
<td>USED</td>
<td></td>
</tr>
<tr>
<td>000BD000-000BFFFF</td>
<td>00003000</td>
<td>---------</td>
<td>ROM2</td>
<td></td>
</tr>
<tr>
<td>000BD000-000BFFFF</td>
<td>00003000</td>
<td>+00003000</td>
<td>FREE</td>
<td></td>
</tr>
</tbody>
</table>

**RAM Area Information**

**ROM Area Information**

*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 includes the section header address.
7.2.4 Symbol List

In the symbol list, the external symbol names, definitions, reference types, and symbol values are displayed.

■ List output format of the symbol list part

The following shows the list output format of the symbol list part.

Figure 7.2-7 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 45)</th>
</tr>
</thead>
<tbody>
<tr>
<td></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) Symbol list display area

Symbol Value : Symbol address or symbol value (hexadecimal)

Type : Symbol type

One of the following is displayed.

Addr. : Address level

EQU : EQU defined symbol

bit : Bit attributes

Def. : Symbol definition

One of the following is displayed.

OM/LM: Defined in the input object module or relative format load module.

LIB: Defined in the linked library

user: Symbol whose value is temporarily set using the -df option.

Symbol Name: Symbol name

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

Note: If a symbol is referred, the symbol name is displayed as it is. If it is not referred, @ is displayed prior to the symbol name.

Moreover, the mangle name and the symbol name are displayed for the symbol generated with the C/C++ compiler.
**List display example of the symbol list part**

The following shows a list display example of the symbol list part.

**Figure 7.2-8  List display example of the symbol list port**

<table>
<thead>
<tr>
<th>Symbol Value</th>
<th>Type</th>
<th>Def.</th>
<th>Symbol Name (Top 45)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000CA(ABS)</td>
<td>Addr.</td>
<td>OM/LM</td>
<td>_ct__8USDollarFUiT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>USDollar::USDollar(unsigned int, unsigned int)</td>
</tr>
<tr>
<td>00001234(ABS)</td>
<td>Addr.</td>
<td>user</td>
<td>_nw__FUi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>operator new(unsigned int)</td>
</tr>
<tr>
<td>00000000(ABS)</td>
<td>Addr.</td>
<td>OM/LM</td>
<td>_pl__FR8USDollarT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>operator +(USDollar &amp;, USDollar &amp;)</td>
</tr>
<tr>
<td>0000004A(ABS)</td>
<td>Addr.</td>
<td>OM/LM</td>
<td>_main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>main</td>
</tr>
</tbody>
</table>
7.3 Absolute Format Assemble List File

The absolute format assemble list output by the linker consists of the following parts.
- Header
- Information list
- Assemble source list

We explain here the items output in each list.

■ Absolute Format 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.

- 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.

  Figure 7.3-1 shows the structure of the list.

**Figure 7.3-1 Assemble list structure**

<table>
<thead>
<tr>
<th>Information list</th>
</tr>
</thead>
<tbody>
<tr>
<td>assemble source list</td>
</tr>
</tbody>
</table>

■ 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 of is displayed after the header. The following information is displayed on an information list.

- Option setting when activating the assembler.
- Number of errors and number of warnings.
- Source file name.
- Include file name.
- Option file name.
- etc.

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.

Figure 7.3-2 Header format

[1st and 2nd lines]

<table>
<thead>
<tr>
<th>FR-V Family SOFTUNE Linker</th>
<th>V60Lxx</th>
<th>YYYY-MM-DD hh:mm:ss</th>
<th>Page:XXXX</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Absolute assemble list creation date</th>
<th>Page number</th>
</tr>
</thead>
</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.
7.3.2 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.

| - SOURCE LISTING - ( Module name ) |
| SN | LOC | OBJ | LLINE | SOURCE |
| XX | XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX | XXXXXXXXX X------- |

| *1 | *2 | *3 | *4 | *5 | *6 | *7 |

The first line of the above format is called the source list header. The source list header is displayed on each page.

*1: Section acronym

The first two characters of a section name are displayed.

*2: Location

The 32-bit location value is displayed as a hexadecimal value.

*3: Object code

The determined object code after linking is displayed as hexadecimal values. If the object code cannot be displayed in one line, it is displayed in multiple lines.

*4: Object code type

The attributes of values contained in object code are displayed in the following order of priority.

I : External reference value
S : Section value
Blank : Absolute value

Since "R" displayed in the relative assemble list is converted to an absolute value in the absolute format assemble list, it is not displayed.

*5: Line number

The line number is displayed as a decimal 10-digit number.
*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]

Figure 7.3-4 Example of assemble source list

<table>
<thead>
<tr>
<th>SN</th>
<th>LOC</th>
<th>OBJ</th>
<th>LLINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;INIT&gt;------------</td>
<td>1025</td>
<td>.SECTION INIT, data</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>002CE724</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>002CE724</td>
<td></td>
<td>1026</td>
<td>LS1</td>
</tr>
<tr>
<td>IN</td>
<td>002CE726</td>
<td>[2] 02</td>
<td>1027</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>002CE726</td>
<td>0020 0010</td>
<td>1028</td>
<td>.DATA.H 32, 16</td>
</tr>
<tr>
<td>CO</td>
<td>0000A280</td>
<td></td>
<td>1029</td>
<td>.SECTION CODE, CODE</td>
</tr>
<tr>
<td>CO</td>
<td>0000A280</td>
<td>88F8002C</td>
<td>1030</td>
<td>SETHI #HI(LS1), GR4</td>
</tr>
</tbody>
</table>

- Source program
- Line number
- Location counter: Displayed with the 32-bit absolute address
- Section acronym: Display first two characters of the section names
- Object Code: Determined value after linking
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.

Figure 7.4-1 Output format of external symbol cross-reference information list

<table>
<thead>
<tr>
<th>Module(s)</th>
<th>YYY-MM-DD hh:mm:ss Page: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. module01</td>
<td></td>
</tr>
<tr>
<td>2. module02</td>
<td></td>
</tr>
<tr>
<td>3. module03</td>
<td></td>
</tr>
<tr>
<td>15. module15</td>
<td></td>
</tr>
</tbody>
</table>

--- Symbol ---

| Extsym1 | Addr. 0x000012E8 | 1 2 3 4 5 6 8 11 12 14 |
| Extsym2 | Addr. 0x000C3F34 | 2 3 4# 5 6 8 11 12 14 |
| Extsym3 | Addr. 0x000012E6 | 1# 2 3 4 5 6 8 11 12 13 |
| Operator new(unsigned int) | Addr. 0x00000AAA | 1 2 3 4 7# 8 |
| Operator new[](unsigned int) | Addr. 0x000000B1E | 4 # |
| Extsymunresolved | Addr. 0x00000000 | 10 15 |
| N1p | EQU 0x00000001 | 4 5 6 11 12 14# |
| Main | Addr. 0x00000314 | 3# 5 |

- **Module(s)**
  
The serial number starting with 1 is added to indicate the module name.

- **Symbol**
  
The symbol names or function names are displayed. (by default, up to 50 characters)
CHAPTER 7 OUTPUT LIST FILE OF THE LINKER

● 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/referred are displayed using their numbers. The # symbol indicates a defined module.
### 7.5 Local Symbol Information List File

The local symbol information list file displays information about the variables and functions which include local symbols for each module constituting an absolute format load module.

Since this list is created based on debug information when compiling and assembling, it is necessary to specify the `-g` option.

#### Local Symbol Information List File

The following shows the output format of the local symbol information list.

#### Figure 7.5-1 Output format of 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>3. module03</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>

<table>
<thead>
<tr>
<th>Local Symbol List</th>
<th>YYYY-MM-DD hh:mm:ss</th>
<th>Page: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>=== Module No. 1 (module01) ===</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--- Symbol ---</td>
<td>--- kind ---</td>
<td>--- val ---</td>
</tr>
<tr>
<td>func1(int)</td>
<td>Func.</td>
<td>g</td>
</tr>
<tr>
<td>localstatic1</td>
<td>Var.</td>
<td>s</td>
</tr>
<tr>
<td>localstatic2</td>
<td>Var.</td>
<td>s</td>
</tr>
</tbody>
</table>

| === Module No. 2 (module02) === | | |
| --- Symbol --- | --- kind --- | --- val --- | C |
| operator +(USDollar &, USDollar &) | Func.  | g  | 0x00000000 |
| main                  | Func.  | g  | 0x00000004A|
| USDollar::USDollar(unsigned int) | Func.  | g  | 0x0000000CA |
| Atable                | loc.   | s  | 0xFFFFFFFC |
| Extsym                | Var.   | g  | 0x00001342 |

- **Module(s)**
  The serial number starting with 1 is added to indicate the module name.

- **symbol**
  The symbol names or function names are displayed.

  Symbols used in a function are displayed in the 3rd and following columns.

  Up to 50 characters of the symbol name are displayed in one line of the list.
CHAPTER 7  OUTPUT LIST FILE OF THE LINKER

- **Kind**

  The following symbol types are displayed.
  
  - Var.: Variable (C/C++)
  - Func.: Function (C/C++)
  - loc.: Local (C/C++)
  - Addr.: Address (ASM)
  - EQU: EQU symbol (ASM)
  - bit: bit symbol (ASM)
  - ????: Undefined
  - s: static (C/C++)
  - g: global (C/C++)

- **val**

  The value of a symbol is indicated.
  
  For loc., an offset value is displayed.
  
  **Note:** No detailed information about structures (member names) nor the typedef definitions are displayed.
7.6 Section Allocation Detailed Information List File

The section allocation detailed information list file creates information about section allocation for each module constituting an absolute format load module. A mapping list of a whole section is displayed in one map list file and more detailed information about section allocation can be found.

Section Allocation Detailed Information List File

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

Figure 7.6-1 Output format of section allocation detailed information list

<table>
<thead>
<tr>
<th>Module(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. module01</td>
</tr>
<tr>
<td>2. module02</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>15. module15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section Mapping List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module(s)</td>
</tr>
<tr>
<td>module01</td>
</tr>
<tr>
<td>module02</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>module15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section Mapping List</th>
</tr>
</thead>
<tbody>
<tr>
<td>S.Addr. - E.Addr.</td>
</tr>
<tr>
<td>Size</td>
</tr>
<tr>
<td>Section</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Al</td>
</tr>
<tr>
<td>M.No.</td>
</tr>
<tr>
<td>Sec.(Top 28)</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>00000000 - 00000048</td>
</tr>
<tr>
<td>00000049</td>
</tr>
<tr>
<td>DATA P</td>
</tr>
<tr>
<td>RW--</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>data</td>
</tr>
<tr>
<td>0000004A - 000003E1</td>
</tr>
<tr>
<td>00000348</td>
</tr>
<tr>
<td>DATA P</td>
</tr>
<tr>
<td>RW--</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>data</td>
</tr>
<tr>
<td>000003E2 - 00000403</td>
</tr>
<tr>
<td>00000222</td>
</tr>
<tr>
<td>DATA P</td>
</tr>
<tr>
<td>RW--</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>data</td>
</tr>
<tr>
<td>00000404 - ..........</td>
</tr>
<tr>
<td>00000000</td>
</tr>
<tr>
<td>DATA P</td>
</tr>
<tr>
<td>RW-I</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>data</td>
</tr>
<tr>
<td>00000404 - 00000663</td>
</tr>
<tr>
<td>00000260</td>
</tr>
<tr>
<td>DATA P</td>
</tr>
<tr>
<td>RW-I</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>init</td>
</tr>
<tr>
<td>00000664 - 00000763</td>
</tr>
<tr>
<td>00000100</td>
</tr>
<tr>
<td>STACK P</td>
</tr>
<tr>
<td>RW--</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>sectionnametoolong+ tooolong</td>
</tr>
<tr>
<td>000C0000 - 000C03DF</td>
</tr>
<tr>
<td>000003E0</td>
</tr>
<tr>
<td>CODE P</td>
</tr>
<tr>
<td>R-XI</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>code</td>
</tr>
<tr>
<td>000C03E0 - 000C0441</td>
</tr>
<tr>
<td>00000062</td>
</tr>
<tr>
<td>CODE P</td>
</tr>
<tr>
<td>R-XI</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>code</td>
</tr>
<tr>
<td>000C0442 - 000C148B</td>
</tr>
<tr>
<td>000104A</td>
</tr>
<tr>
<td>CODE P</td>
</tr>
<tr>
<td>R-XI</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>code</td>
</tr>
<tr>
<td>*000C148C - 000C201D</td>
</tr>
<tr>
<td>00000B92</td>
</tr>
<tr>
<td>CODE P</td>
</tr>
<tr>
<td>R-XI</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>code</td>
</tr>
<tr>
<td>*000C2000 - 000C2203</td>
</tr>
<tr>
<td>00000204</td>
</tr>
<tr>
<td>CODE P</td>
</tr>
<tr>
<td>R-XI</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>subprog1</td>
</tr>
<tr>
<td>000C3000 - 000C325F</td>
</tr>
<tr>
<td>00000260</td>
</tr>
<tr>
<td>DATA P</td>
</tr>
<tr>
<td>R--I</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>#init</td>
</tr>
</tbody>
</table>

- **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.
CHAPTER 7  OUTPUT LIST FILE OF THE LINKER

- **Section**
  
  The following section content types are available.
  
  - CODE : Program section
  - DATA : Data section
  - CONST : Data section with initial values
  - STACK : Stack section
  - IO : IO section
  
  The link attribute is added to the end of the section type.
  
  - P : Simple concatenation link
  - C : Shared link
  - A : no link

- **type**
  
  The following attributes are displayed from left.
  
  - R/- : Read enabled/disabled
  - W/- : Write enabled/disabled
  - X/- : Executable/non-executable
  - I/- : Initial value Yes/No

- **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 are restrictions shown in Table 8.1-1 for processing of the input file count, section count and symbol count when you are using the Linker.

■ Linker restriction

There are restrictions shown in Table 8.1-1 for processing of the input file count, section count and symbol count when you are using the Linker.

However, this is not the maximum limit value available for processing.

Linker performs processing, gaining a memory dynamically.

Linker outputs an error message with an insufficient memory, when gaining a memory required for processing becomes impossible, and processing is interrupted.

<table>
<thead>
<tr>
<th>Item</th>
<th>Restriction Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option file count</td>
<td>Limitless</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Option file internal line count</td>
<td>Limitless</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Option file internal character count per 1 line</td>
<td>Limitless</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Option file nest</td>
<td>Not possible</td>
<td></td>
</tr>
<tr>
<td>Input file count</td>
<td>4,294,967,295</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Input module count</td>
<td>4,294,967,295</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>I/O file size</td>
<td>Limitless</td>
<td>OS dependent</td>
</tr>
<tr>
<td>Module name/section name/symbol name character count</td>
<td>Limitless</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>File name character count</td>
<td>Limitless</td>
<td>OS dependent</td>
</tr>
<tr>
<td>Section count</td>
<td>4,294,967,295</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Maximum section size</td>
<td>4GB</td>
<td></td>
</tr>
<tr>
<td>Externally defined symbol count</td>
<td>4,294,967,295</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Externally referenced symbol count</td>
<td>4,294,967,295</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Externally defined symbol reference count</td>
<td>Limitless</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Maximum source count</td>
<td>4,294,967,295</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Maximum source line count</td>
<td>4,294,967,295</td>
<td>Memory dependent</td>
</tr>
</tbody>
</table>

■ Linker reservation symbol

The Linker automatically generates symbols with the "_ROM_ section name" or "_RAM_ section name" for each section using the from ROM to RAM transfer function. Therefore, if there are symbols in the user program that have the same name, a "W1327L: Duplicate symbol definition (symbol name)" will occur. The user should not define symbols using "_ROM_ section name" or "_RAM_ section name."

Also, the "_ROM_ value" or "_RAM_ value" for the ROM/RAM region names are automatically set from the CPU information file. The user should not define region names with the "_ROM_ value" or "_RAM_ value."
### 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>
<tr>
<td>Example.</td>
<td>flnk935s *.obj -o outfile.abs  &lt;br&gt;flnk935s mactrl.obj xz????.obj</td>
</tr>
</tbody>
</table>

<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)  <br>-sc DTdata1+DTdata2+DTdata3+DTdata4=0x1000, CDprog1+CDprog2+CDprog3+CDprog4=0X3000  <br>-sc DT*=0x1000,CD*=0x3000  <br>-sc */data=0x1000,*/code=0x3000 |
### Handling variables with initial values

**Q.** When developing embedded programs using C/C++ compilers, variables with initial values are created. Since these variables are rewritten during execution of programs, they must be on RAM during execution. Tell us the procedure for creating programs and precautions.

**A.** In embedded programs, variables with initial values are on ROM first and they must be on RAM when they are referred. Therefore, programs become inoperable if the reference address in programs is not set to RAM and a mechanism to transfer initial value data from ROM to RAM before application execution is not implemented.

In the FR-V family, this mechanism is implemented by using the ROM -> RAM transfer section function supported by the linker.

Variables with initial values generated by the FR-V family compilers are gathered in the INIT section.

There is no need of particular care when creating programs except the total number of bytes of the variables with initial values and RAM size.

For the ROM > RAM transfer section function, see "5.9 Sections to be Transferred from ROM to RAM".

The user must write a program to transfer initial value data using assembler languages. Example 1 shows "Program example for transferring initial value data".

<table>
<thead>
<tr>
<th>Example1</th>
<th>[Program example for transferring initial value data :FR500 family]</th>
</tr>
</thead>
<tbody>
<tr>
<td>.IMPORT</td>
<td>_ROM_INIT /* Start address of the INIT section in ROM */</td>
</tr>
<tr>
<td>.IMPORT</td>
<td>_RAM_INIT /* Start address of the INIT section in RAM */</td>
</tr>
<tr>
<td>.SECTION</td>
<td>INIT,DATA,ALIGN=4 /* Section for initial-value data that is allowed to be changed */</td>
</tr>
<tr>
<td>ptbl:</td>
<td>/* ptbl indicates the address in RAM */</td>
</tr>
<tr>
<td>.DATA</td>
<td>10,11,12 /* Initial-value data is stored in ROM */</td>
</tr>
<tr>
<td>.SECTION</td>
<td>CODE,CODE,ALIGN=4 /* ROM-to-RAM transfer program */</td>
</tr>
<tr>
<td>init_copy:</td>
<td></td>
</tr>
<tr>
<td>SETHI</td>
<td>#HI(_ROM_INIT),GR1 /* Source (ROM) address */</td>
</tr>
<tr>
<td>SETLO</td>
<td>#LO(_ROM_INIT),GR1</td>
</tr>
<tr>
<td>SETHI</td>
<td>#HI(_RAM_INIT),GR2 /* Destination (RAM) address */</td>
</tr>
<tr>
<td>SETLO</td>
<td>#LO(_RAM_INIT),GR2</td>
</tr>
<tr>
<td>SETHI</td>
<td>#HI(SIZEOF(INIT)),GR3 /* Obtain the INIT section size */</td>
</tr>
<tr>
<td>SETLO</td>
<td>#LO(SIZEOF(INIT)),GR3</td>
</tr>
<tr>
<td>CMP</td>
<td>GR3,#0,ICC0 /* Test the transfer size for 0 */</td>
</tr>
<tr>
<td>BEQ</td>
<td>ICC0,#0,init_copy_end</td>
</tr>
<tr>
<td>init_copy_loop:</td>
<td></td>
</tr>
<tr>
<td>SUBcc</td>
<td>GR3,#-1,GR3,ICC1 /* Reduce the size (GR3) by 1 */</td>
</tr>
<tr>
<td>LDUB</td>
<td>@(GR1,GR3),GR4 /* Obtain the ROM data */</td>
</tr>
<tr>
<td>STB</td>
<td>GR4,@(GR2,GR3) /* Transfer to RAM (delayed instruction) */</td>
</tr>
<tr>
<td>BNE</td>
<td>ICC1,#0,init_copy_loop /* Repeat until the size (GR3) becomes 0 */</td>
</tr>
<tr>
<td>init_copy_end:</td>
<td></td>
</tr>
</tbody>
</table>
PART III LIBRARIAN

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

CHAPTER 9 SPECIFICATIONS OF A LIBRARIAN
CHAPTER 10 OPTIONS OF 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 Objects Generated Using the SOFTUNE V5 Language Tool
9.8 Library Made by the SOFTUNE V5 Tool Language Tool
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 modules that an assembler has output.

You may register multiple modules that make up a program in the library file dedicated to the program to manage them in a batch.

Registering the frequently used modules in a batch to create a general-purpose library file enables them to be easily used in other programs. A 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 9.1-1 shows the roles of a librarian.

Figure 9.1-1 Roles of a librarian
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 9.3-1).

![Figure 9.3-1 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 9.3-2).

![Figure 9.3-2 Adding a module](image)
● Deleting a module

An unnecessary module may be deleted from an existing library file (See Figure 9.3-3).

Figure 9.3-3 Deleting a module

Existing library file

Deleted

Updated

Library file after editing

● Replacing a module

A module in an existing library file may be replaced with a new one (See Figure 9.3-4).

Figure 9.3-4 Replacing a module
9.4 Extracting a Module from a Library File

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

Diagram: Extracting a module from a library file

Figure 9.4-1 Extracting a module

Existing library file

Module A
Module B
Module D

Extracted

B.obj
Module B

Object file
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 9.5-1).

---

**Figure 9.5-1 Deleting debugging information**

OM1 and OM3 registered with a debugging information block may be re-registered without it.
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.

1. 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.

2. Whether a module with debugging information exists

   If an object module with debugging information is contained, a diagnostic message is output.

   If a module to be registered with the -g option specification in a library file contains debugging information in the object module, it is registered in the library with the debugging information unremoved.

   The above function is provided so that a module registered in a library may be debugged. However, after the operation check, the debugging information will no longer be required.

### Displaying the contents of a library file

The module and external definition symbol information of a library file is edited and output to a list file or the standard output.

The list provides information such as: the date and time a library file was created and updated, when a module was registered, and the name of an external definition symbol defined in each module.

For the contents of display, see “Chapter 11 LIST FORMATS OF A LIBRARIAN.”
9.7 Objects Generated Using the SOFTUNE V5 Language Tool

The librarian can inputs objects generated using the SOFTUNE V5 language tool as a library.

- Objects Generated Using the SOFTUNE V5 Tool
  Librarian (flibs) can interleaving objects generated using the SOFTUNE V5 language tool in the library.
  The librarian will output information when you process objects generated using the SOFTUNE V5 tool.
9.8 Library Made by the SOFTUNE V5 Tool Language Tool

Librarian can edit the library made by the SOFTUNE V5 language tool.

- **Library File Made by the SOFTUNE V5 Tool**
  
  Librarian can edit the library made by the SOFTUNE V5 language tool.
  
  When the library made by the SOFTUNE V5 language tool is edited, the librarian outputs information.
  
  And also, the librarian automatically makes a backup file with an extension changed .bak for the library file that has not been edited yet.
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 the 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>
<thead>
<tr>
<th>Table 10.1-1 List of options for a librarian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Options for creating and editing a library</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Options for outputting a list</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Other options</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<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 Linkage kit are described in "CHAPTER 3 COMMON OPTIONS".

■ Options for creating and editing a library
  Details of the options for creating and editing a library are described in "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 "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 "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 "10.2.14 Specifying to Output Debugging Information (-g)" through "10.2.17 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)

[Format]

- 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]

flibs 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]

flibs 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: 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 Specification Differences Depending on the OS".
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)

[Format]

- 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]

flibs syslib.lib -r loadx.obj,loady.obj

The two modules in loadx.obj and loady.obj replace 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]

flibs 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 Specification Differences Depending on the OS". 
10.2.3 Deleting a Module (-d)

An unnecessary module is deleted from a library file.

- Deleting a module (-d)

  **[Format]**

  ```
  - 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**

  `flibs syslib.lib -d inchar,outchar`

  Two modules, inchar and outchar are deleted from syslib.lib.

  **Note:** Although you may specify a module name using a pseudo instruction of assembler, use the same name for an object module file and a module unless absolutely 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 it back into an object module file before registration.

Extracting a module (-x)

[Format]

- 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]

flibs syslib -x add
flibs syslib.lib -x add,add.obj
The module, add is extracted from an existing library file and the add.obj file is created.
flibs 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.
flibs syslib -x add -x add,addfunc.obj
Modules generated using the SOFTUNE V5 tools output using the current object formats.
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)

[Format]

- 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]

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

[Example 2]

flibs 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]

flibs 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)

  [Format]

  - 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]

  flibs 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 definition and external reference symbols for the entire library.

### Specifying to output Detailed Information of a list file (-dt)

**[Format]**

```
- 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 keywords separated with commas. If the -m option is not specified, this option is invalid.

**[Example 1]**

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

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

**[Example 2]**

```
flibs 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)

[Format]

- 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]

flibs syslib.lib -m libx.mp2 -pl 40
The number of lines per page of a list is 40.

[Example 2]

flibs 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 (80 characters).

- Specifying the number of columns per line of a list (-pw)

  [Format]

  - pw < Number of columns > (Default: 80)

  [Parameter]

  <Number of columns>
  Specify between 80 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 (80), the number of characters displayable per line are as follows:
  • Module name : 21 characters
  • Section name : 19 characters
  • Symbol name : 34 characters
  If the -m option is not specified, this option is invalid.

  [Example]

  flibs syslib.lib -m libx.mp2 -pw 90
  The number of columns per line of a list is changed to 90.
  In this case, the number of characters displayable per line for each name are as follows:
  • Module name : 31 characters
  • Section name : 29 characters
  • Symbol name : 39 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)

[Format]

- 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]

flibs 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.

Note: If a library file which the input library is created using the SOFTUNE V5 tool is provided, the backup file will be stored as "bak" file according to the file using SOFTUNE V5.
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)

**[Format]**

```
-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.

```
flibs syslib  -a putc.obj,getc.obj
flibs syslib  -a putc.obj,getc.obj -Xb
flibs syslib -b -a putc.obj,getc.obj -Xb
```

Before editing, syslib.lib is deleted after editing. To syslib.lib after editing, putc.obj and getc.obj are added.
10.2.12 Checking the Contents of a Library File (-c)

Use this option to briefly check the contents of a library file.

**Checking the contents of a library file (-c)**

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>- c</td>
</tr>
</tbody>
</table>

**Parameter**

None

**Description**

The following two items are checked.

- **Whether an unsolved external reference symbol exists in the library**

  A linker, when a module taken in from a library contains an unsolved external reference symbol, first tries to solve the symbol in the same library file, assuming that a module containing a defined symbol exists in the same library file.

  The program checks the correspondence between external reference and defined symbols in a library file. If any external reference symbols without the corresponding external definition symbol is contained, it outputs a message.

- **Whether a module with debugging information exists**

  If a module to be registered with the -g option specification in a library file contains debugging information in the object module, it is registered in the library with the debugging information unremoved.

  The above function is provided so that a module registered in a library may be debugged. However, after the operation check, the debugging information will no longer be required.

  The program checks whether a module with debugging information is registered in a library and, if so, outputs a message.

**Example**

```
flibs syslib.lib -c
```

The contents of syslib.lib are checked.

**Note:** This option cannot be specified with other options. Specify this option alone as shown in the above example.
10.2.13 Optimizing the Contents of a File (-O)

Any debugging information contained in an object module registered in a library is removed.

- Optimizing the contents of a file (-O)

[Format]

```
-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]

```
flibs syslib -O
```

Debugging information is deleted from the syslib.lib file.

Note: This optimization option cannot be specified with other options. Specify this option alone as shown in the above example.
10.2.14 Specifying to Output Debugging Information (-g)

Use this option so as not to delete debugging information when registering an object module in a library file.

■ Specifying to output debugging information (-g)

[Format]

- 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]
flibs 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)

[Format]

- 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.
flibs syslib.lib -a inchar,outchar
flibs syslib.lib -a inchar,outchar -Xg
flibs 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 CPU Information File (-cif)

This specifies CPU information file used by the library.

CPU Information File Specification (-cif)

[Format]

- cif <CPU information file name>

[Parameter]

<CPU information file name>
CPU information file name used by library

[Description]

The specifies the CPU information file used by the library.

[Example]

flibs syslib.lib -a inchar.outchar -cpu MB93501A
- cif C:\Softune6\lib\935\cpu_info\MB93501A.csv
flibs syslib.lib -a inchar.outchar -cpu MB93530
- cif C:\Softune6\lib\935\935.csv

Note: SOFTUNE Tools get CPU information by referring the CPU information file. Refer to a CPU information file different between the related tools may cause an error to the program to be created.

The CPU information file that comes standard with SOFTUNE Tools is located at:

Installation Directory\lib\935\935.csv

When installing the compiler and assembler pack in a different directory, specify -cif so that each tool can refer the same CPU information file.
10.2.17 Specifying a Target CPU (-cpu)

Use this option to specify a target CPU. Use an MB number to specify a target CPU of the programs to be combined into a library file.

- Specifying a target CPU (-cpu)

[Format]

- 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]

flibs syslib.lib -a inchar,outchar -cpu MB93501A
flibs syslib.lib -a inchar,outchar -cpu MB93530

[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 reference symbol information for 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 11.1-1 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>CPU information etc.</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>Name of registered modules</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt; Section information for each module &gt;</th>
<th>-dt s specified</th>
</tr>
</thead>
<tbody>
<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>

<table>
<thead>
<tr>
<th>&lt; External defined and reference symbol name information for whole modules &gt;</th>
<th>-dt a specified</th>
</tr>
</thead>
</table>

Figure 11.1-1 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 : sample.lib
*2 : Number of Modules : 3
*3 : Number of Symbols : 9
*4 : CPU information : MB93501
*5 : Library Creation Date 2003-03-01 14:23:50
*6 : Library Revision Date 2003-04-17 09:41:15

[ Module Name ] [ Entry Date ] [ Creation Date ] [OMF]
*7 ModuleA 2003-03-01 14:23:50 2003-03-19 10:03:21
ModuleB 2002-04-17 09:41:15 2002-10-07 20:18:58 *
ModuleC 2003-03-01 14:23:50 2003-02-23 15:15:00
```

*1 : Library file name
*2 : Number of modules registered in a library file (expressed in decimals)
*3 : Number of external definition symbols registered in a library file (expressed in decimals)
*4 : CPU information (MB number)
*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 -59).
    The default (-pw 80) is 21 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
    [ OMF ] OMF Type
11.3 Detailed Information of a Module

Detailed information of a module comes in the following three groups. Use the -dt option to specify outputting the detailed information.

- Section information (-dt s)
- External definition symbol information (-dt d)
- External reference symbol information (-dt r)

### List output overview

Figure 11.3-1 shows the format of a librarian list (when detailed information is specified).

**Figure 11.3-1 Format of a librarian list (when detailed information is specified)**

<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>CPU information</td>
<td>MB93501</td>
</tr>
<tr>
<td>Library Creation Date</td>
<td>2003-03-01 14:23:50</td>
</tr>
<tr>
<td>Library Revision Date</td>
<td>2003-04-17 09:41:15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Entry Date</th>
<th>Creation Date</th>
<th>OMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModuleA</td>
<td>2003-03-01 14:23:50</td>
<td>2003-03-19 10:03:21</td>
<td>*</td>
</tr>
</tbody>
</table>

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

This is the information of sections in a module.

For a section name, each line displays as many characters as (Page width - 61).

*2 : Output by the r parameter in the -dt option.

Each line displays two of the external reference symbol names in a module.

*3 : Output by the d parameter in the -dt option.

Each line displays two of the external definition symbol names in a module.

For an external symbol name, each line displays as many characters as ((Page width -12) /2).
11.4 External Defined and Reference Symbol Information in a Library

The external defined and reference symbol information of all the modules registered in a library file may be displayed. Use the -dt option to specify to output this information. (-dt a)

**List output overview**

Figure 11.4-1 shows the format of a librarian list (when detailed information is specified).

**Figure 11.4-1 Format of a librarian list (when detailed information is specified)**

<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>CPU information</td>
<td>: MB93501</td>
</tr>
<tr>
<td>Library Creation Date</td>
<td>2003-03-01 14:23:50</td>
</tr>
<tr>
<td>Library Revision Date</td>
<td>2003-04-17 09:41:15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[ Module Name ]</th>
<th>[ Entry Date ]</th>
<th>[ Creation Date ]</th>
<th>[OMF]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModuleA</td>
<td>2003-03-01 14:23:50</td>
<td>2003-03-19 10:03:21</td>
<td>*</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 library file.

*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 width - 12) / 2).
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
12.1 Restrictions on a Librarian

This section describes the restrictions concerning the number of modules and external symbols that can be registered to one library file when using the Librarian and it describes circumstances of which you should be aware.

Restrictions on a librarian

Table 12.1-1 shows the restrictions when using the Librarian. However, this is not the maximum limit value available for processing.

Librarian performs processing, gaining a memory dynamically.

Librarian outputs an error message with an insufficient memory, when gaining a memory required for processing becomes impossible, and processing is interrupted.

Table 12.1-1 List of Restrictions on a Librarian

<table>
<thead>
<tr>
<th>Item</th>
<th>Restriction Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option File Count</td>
<td>Limitless</td>
<td>Memory Dependent</td>
</tr>
<tr>
<td>Option File Internal Line Count</td>
<td>Limitless</td>
<td>Memory Dependent</td>
</tr>
<tr>
<td>Option File Internal Character Count Per 1 Line</td>
<td>4,095</td>
<td></td>
</tr>
<tr>
<td>Option File Nest</td>
<td>Not Possible</td>
<td></td>
</tr>
<tr>
<td>Input File Count</td>
<td>4,294,967,295</td>
<td>Memory Dependent</td>
</tr>
<tr>
<td>Input Module Count</td>
<td>4,294,967,295</td>
<td>Memory Dependent</td>
</tr>
<tr>
<td>I/O File Size</td>
<td>Limitless</td>
<td>OS Dependent</td>
</tr>
<tr>
<td>Module Name/Section Name/Symbol Name Character Count</td>
<td>Limitless</td>
<td>Memory Dependent</td>
</tr>
<tr>
<td>File Name Character Count</td>
<td>Limitless</td>
<td>OS Dependent</td>
</tr>
<tr>
<td>Section Count</td>
<td>4,294,967,295</td>
<td>Memory Dependent</td>
</tr>
<tr>
<td>Maximum Section Size</td>
<td>4GB</td>
<td></td>
</tr>
<tr>
<td>Externally Defined Symbol Count</td>
<td>4,294,967,295</td>
<td>Memory Dependent</td>
</tr>
<tr>
<td>Externally Reference Symbol Count</td>
<td>4,294,967,295</td>
<td>Memory Dependent</td>
</tr>
<tr>
<td>Externally Defined Symbol Reference Count</td>
<td>Limitless</td>
<td>Memory Dependent</td>
</tr>
</tbody>
</table>

Cautionary Information Concerning the Necessary Disk Space

If you are editing your pre-existing library file and generating a back-up file, do so after checking if you have enough disk space to store both the newly generated library file and the pre-existing librarian.

Cautionary Information Concerning the Specification of Options

Individually specify both options of the librarian file contents search (-c) and file data optimizer (-O).

This cannot be used in conjunction with other options.
12.2 Questions and Answers on Using a Librarian

Section 12.2 shows the questions and answers on using a librarian.

Questions and answers on creating a library file

<table>
<thead>
<tr>
<th>Q.</th>
<th>What is the format of a file that can be registered in a library file?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>It is an object module that an assembler outputs. This is a file created with the .obj extension by default.</td>
</tr>
</tbody>
</table>

Example.

<table>
<thead>
<tr>
<th>Q.</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>You can access symbol information for debugging only in an object module with debugging information. You must replace the object module with the one with debugging information. When you create a library out of the object modules that may need to be debugged, it is recommended to register them with debugging information (using the -g option). When debugging is complete, you can delete debugging information (using the -0 option)</td>
</tr>
</tbody>
</table>

Example.

<table>
<thead>
<tr>
<th>Q.</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Use a wildcard to specify object modules to be added to a library (in the -a option) or replaced with the one in a library (in the -r option).</td>
</tr>
</tbody>
</table>

Example.

| fasm935s file1 -g -> file1.obj is output. |
| fasm935s file2 -g -> file2.obj is output. |
| flibs libfile -a file1.obj,file2.obj |

| fasm935s file1 -g -> file1.obj with debugging information is output. |
| flibs libfile -r file1 -g -> libfile.lib with debugging information is output. |
| flibs libfile -O -> libfile.lib without debugging information is output. |

| flibs libfile -a "*.obj" -> All the files with the ".obj" extension are registered. |
| Q. | I forgot what is contained in a library file that I created a while ago. How can I find what kind of modules are registered in it? |
| A. | Use the `-m` option to see what is contained in a library file. Specify `-m File-name` to create a list file with the `.mp2` extension by default. If the contents output by the `-m` option is not sufficient for you, use the `-dt` option together to obtain more detailed information. |
| Example. | `flibs libfile -m libdoc` -> A list file, `libdoc.mp2` is output.  
`flibs 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. | `flibs libfile -m -`  
`flibs libfile -m - -dt a,s`  
`flibs 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 80 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 88. Then the names will fit in one line. |
| Example. | `flibs libfile -m libdoc -pw 90` |

| Q. | I replaced modules in a library file with the new ones only to realize later that I had registered some of them by mistake. Since I did not keep any backup for the library file before the replacement, I had a hard time restoring it. |
| A. | A librarian allows you to create a backup file for one generation (with the "*.bak" extension) using the `-b` option. Naturally, it is recommended to create a backup of a library before editing it. However, specify the `-b` option as required. |
| Example. | `flibs libfile -r file1,file2 -d mod4 -b` |
Part IV OBJECT FORMAT CONVERTERS

Part IV describes the types of object format converters, list of options, functions, and conversions of object formats.

CHAPTER 13  SPECIFICATIONS OF AN OBJECT FORMAT CONVERTER
CHAPTER 14  COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER
CHAPTER 15  LOAD MODULE CONVERTER (f2ms, f2hs, f2is, f2es)
CHAPTER 16  FORMAT ADJUSTER (m2ms, h2hs)
CHAPTER 17  BINARY CONVERTER (m2bs, h2bs)
CHAPTER 18  OTHER CONVERTERS
CHAPTER 19  RESTRICTIONS AND QUESTIONS AND ANSWERS ON AN OBJECT FORMAT CONVERTER
Chapter 13 gives an overview and describes the types of object format converters. An object format converter is a tool used to convert an object format.

13.1 Outline of Object Format Converter
13.2 Types of Object Format Converters
13.3 Executing an Object Format Converter
13.1 Outline of Object Format Converter

The object format converter processes four types of the following file formats.
- Absolute format load module of linker output
- S format
- HEX format
- Binary data file

### Outline of Object Format Converter

Object format converters include four types of converters: load module converter, adjuster (adjusting tool), binary converter, and converter.

### Load module converter

This converter converts the absolute format load module of the linker output to a general-purpose format. Figure 13.1-1 shows input and output of the load module converter.
● Adjuster, Binary converter

The adjuster adjusts the S format or Hex format. The binary converter converts the S or Hex format into a binary format.

Figure 13.1-2 shows input and output of the adjuster and binary converter.

![Figure 13.1-2 Input and output of the adjuster and binary converter](image)

● Converter

The converter converts the S format and Hex 8/Hex 16 format with each other.

Figure 13.1-3 shows input and output of the converter.

![Figure 13.1-3 Input and output of the converter](image)
13.2 Types of Object Format Converters

The command for an object format converter is x2ys, where the x represents the object format of an input file and the y represents the object format of an output file. An alphabetical character assigned in the x and y format means one of the following formats:

- **f**: Absolute format load module that a linker outputs
- **m**: S format
- **h**: HEX format (HEX8/HEX16/HEX32)
- **b**: Binary data format
- **i**: HEX8 format (Only HEX8)
- **e**: HEX16 format (Only HEX16)

### Types of load module converters

Use the commands in Table 13.2-1 to convert an object format:

- `f2ms` is used for conversion to the S format.
- `f2hs` is used for conversion to the HEX format.

It is enabled to convert to the HEX8 format using `f2is` and to the HEX16 format using `f2es`, but it is recommended to use `f2hs` corresponding HEX8/HEX16/HEX32.

**Table 13.2-1 Conversions made by load module converters**

<table>
<thead>
<tr>
<th>Command name</th>
<th>Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>f2ms</td>
<td>Absolute format load module to S format</td>
</tr>
<tr>
<td>f2hs</td>
<td>Absolute format load module to HEX8/HEX16/HEX32 format</td>
</tr>
<tr>
<td>f2is</td>
<td>Absolute format load module to HEX8 format</td>
</tr>
<tr>
<td>f2es</td>
<td>Absolute format load module to HEX16 format</td>
</tr>
</tbody>
</table>

### Format Adjuster

The S Format Adjuster formats an object file in the S format and the HEX format. For details, see Chapter 16, "Options of the S Format Adjuster".

### Binary Converter

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

### Types of other converters

Use the commands in Table 13.2-2 to convert an object format.

**Table 13.2-2 Conversions made by other converters**

<table>
<thead>
<tr>
<th>Command name</th>
<th>Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2is</td>
<td>S format to HEX8 format</td>
</tr>
<tr>
<td>m2es</td>
<td>S format to HEX16 format</td>
</tr>
<tr>
<td>i2ms</td>
<td>HEX8 format to S format</td>
</tr>
<tr>
<td>e2ms</td>
<td>HEX16 format to 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.

```plaintext
x2ys < Input file name > [ Option ]
```

The specified <Input file name> is processed as the x format and a file in the y format is created.

A converter uses the following extensions by default to identify an object format from a file name:

- Absolute format load module: .abs
- S format: .mhx, .ahx
- HEX8/HEX16/HEX32: .hex, .aix
- Binary data format: .bin
- HEX8: .ihx
- HEX16: .ehx

The -ran option is always required to execute the binary converter and adjuster. For details, see Section 17.3.1, "Specifying the Output Range (-ran)."

[Example]

```plaintext
f2ms sample
```

The absolute format load module that a linker outputs, sample.abs, is input and the sample.mhx file in the S format is output.
Chapter 14 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)
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 an object format converter, the following common options may be used.

Table 14.1-1 lists types of common options of an object format converter.

Table 14.1-1 List of common options of an object format converter

<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 number and messages</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output the version number 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 creates an output file after conversion and the file name are changed from the default.

- Changing an Output File name (-o)

[Format]

- o < Object file name >

[Parameter]

<Object file name> Output file name

[Description]

Specify this option to change the output file name after conversion.
Specify this option with a path name to change also the output destination directory.
If this option is omitted, the output file name will be the same as the input file name. However, its extension will be changed to the default of one of the formats used after conversion.
If the extension is omitted in the <Object file name> specification, the default extension is added.

One of the following six default extensions is used for each format:

- Absolute format load module : .abs
- S format : .mhx .ahx
- HEX8/HEX16/HEX32 : .hex .aix
- Binary data file : .bin
- HEX8 : .ihx
- HEX16 : .ehx

[Example 1]

f2ms ccp903 (Example of not using the -o option)
The absolute format load module ccp903.abs is input and ccp903.mhx in the S format is output. The following four examples are equivalent to the above.

f2ms ccp903.abs -o ccp903.mhx
f2ms ccp903.abs -o ccp903
f2ms ccp903 -o ccp903.mhx
f2ms ccp903 -o ccp903

[Example 2]

f2ms ccp903 -o ccp903.hex
The output file name is changed to ccp903.hex.

[Example 3]

f2ms ccp903 -o ..\hex\ccp903m.hex
The current output destination directory is changed to ..\hex and the output file name to ccp903m.hex.
Note: If the -sp option is specified for binary converter, the <Object file name> is evaluated differently. The <Object file name> is assumed to have no extension and an extension is unconditionally added to the file name specified in <Object file name>.

For example, if "binary.bin" is specified as the object file name, the output file names will be "binary.bin.b01", "binary.bin.b02", .... "binary.bin.bxx".
14.3 Padding (-p)

The specified range of addresses is padded with data of a specified value. With the binary converter and adjuster, the portion of an input file containing no data is padded with data of a specified value when the adjust option of the load module converter (f2ms or f2hs) is specified.

- **Padding (-p)**

  [Format]

  ```
  - p < Value > [, < Starting address > , < Ending address > ]
  ```

  [Parameters]

  <Value> One-byte data
  <Starting address> Starting address at which sets <Value>
  * When the adjust option (-adjust) of f2ms or f2hs is specified, binary converter and adjuster cannot be used.
  <Ending address> Ending address at which sets <Value>
  * When the adjust option (-adjust) of f2ms or f2hs is specified, binary converter and adjuster cannot be used.

  [Description]

  Embed the specified address range with the specified value data.
  Set only for <value> with the adjust option (-adjust) for the load module converter (f2ms or f2hs) and with the binary converter and adjuster.
  Embeds with value data specified by locations that do not exist for data in the file with the adjust option (-adjust) for the load module converter (f2ms or f2hs) and with the binary converter and adjuster.

  [Example 1]

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

  An absolute format load module is converted into the S format.
  At this time, the data at the addresses 0x1FE4 through 0x1FFF is created as the 0xEF data and added to the end of a S format file.

  [Example 2]

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

  The error is generated at specifying adjust(-adjust) because the starting/ending addresses are specified by the padding option(-p).

  ```
  f2ms ccp903 -p 0xEF -adjust
  ```

  An absolute format load module is converted into the S format load module is converted into the S format. At that time, pad the portion which the data dose not exist using data of 0xEF.

  [Example 3]

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

  The data at the addresses 0x0 through 0x1FFF in the S format file is converted to a binary image.
  At this time, the portion of an S format file containing no data is padded with data of 0xEF.
Chapter 15 describes the conversion formats of the load module converter.

15.1 Outline of Load Module Converter
15.2 List of Options of the Load Module Converter
15.3 Details of Load Module Converter Options
15.4 f2ms (Converting an Absolute Format Load Module into the S format)
15.5 f2hs (Converting an Absolute Format Load Module into the HEX format)
15.6 f2is (Converting an Absolute Format Load Module into the HEX8 format), f2es (Converting an Absolute Format Load Module into the HEX16 format)
15.1 Outline of Load Module Converter

The load module converter converts an absolute format load module to the S format or HEX format which are general-purpose formats.

The load module converter converts the absolute format load module to an S format or HEX format which are general-purpose formats.

f2ms can be used to convert to the S format. f2hs can be used to convert to the HEX format.

f2is converts to HEX 8 format, and f2es to HEX 16 format, but if f2hs is used, it is possible to convert to all HEX 8, HEX 16 and HEX 32 HEX formats.

As shown in Figure 15.1-1, by specifying the adjust option (-adjust), f2ms and f2hs can adjust the output file together with adjuster.
15.2 **List of Options of the Load Module Converter**

The following lists the option names and function outlines of the load module converter.

### List of Options of the Load Module Converter

The table below gives the option of the load module converter.

**Table 15.2-1 f Options of the Load Module 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 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 to output S1 format</td>
<td>-S1</td>
<td>Only f2ms</td>
</tr>
<tr>
<td>Specifying to output S2 format</td>
<td>-S2</td>
<td>Only f2ms</td>
</tr>
<tr>
<td>Specifying to output S3 format</td>
<td>-S3</td>
<td>Only f2ms</td>
</tr>
<tr>
<td>Specifying to output HEX8 format</td>
<td>-I16</td>
<td>Only f2hs</td>
</tr>
<tr>
<td>Specifying to output HEX16 format</td>
<td>-I20</td>
<td>Only f2hs</td>
</tr>
<tr>
<td>Specifying to output HEX32 format</td>
<td>-I32</td>
<td>Only f2hs</td>
</tr>
<tr>
<td>Specifying to output start address record</td>
<td>-entry</td>
<td>Only f2hs</td>
</tr>
<tr>
<td>Specifying to not output start address record</td>
<td>-Xentry</td>
<td>Only f2hs</td>
</tr>
<tr>
<td>Specifying to Adjust</td>
<td>-adjust</td>
<td>Only f2ms and f2hs</td>
</tr>
<tr>
<td>Specifying not to read default option file</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to read option file</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying display of help message</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying version number and message</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output version number and message</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to output end message</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output end 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>
CHAPTER 15  LOAD MODULE CONVERTER (f2ms,f2hs,f2is,f2es)

15.3   Details of Load Module Converter Options

This section describes each of the options for the load module converter. Note that common options to the linkage kit are described in CHAPTER 3. COMMON OPTIONS, and common options to the converter are described in CHAPTER 14. COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER.

■ Output S format option (-S1/-S2/-S3)
  Specifies the record used to output with S format. For details, see section 15.3.1 Output S format Option (-S1/-S2/-S3).

■ Output HEX format option (-I16/-I20/-I32)
  Specifies the record used to output with HEX format. For details, see section 15.3.2 Output HEX Format Option (-I16/-I20/-I32).

■ Start address output option (-entry)
  This outputs the start segment address record or start linear address record when outputting with HEX format. For details, see section 15.3.3 Specifying to Output Start Address Record (-entry).

■ Start address output inhibit option (-Xentry)
  This specifies when inhibiting the start address record output. For details, see section 15.3.4 Specifying not to Output Start Address Record (-Xentry).

■ Adjust option (-adjust)
  This specifies to start the adjuster after outputting the S format or HEX format. For details, see section 15.3.5 Specifying to Adjust (-adjust).
15.3.1 Output S format Option (-S1/-S2/-S3)

This option specifies the format used to output data.

Output S format option (-S1/-S2/-S3)

[Format]

- -S1

- -S2

- -S3

[Parameter]
None

[Description]
Specifies the format used to output data.

f2ms outputs the data using either the S1 record, S2 record or the S3 record.
It will not output if both the S1 record and the S2 record are used.
The -S1, -S2 and -S3 options take effect when specified last. If these options are not specified, the f2ms command outputs data in mixed formats of S1/S2/S3 according to the data address.

[Precautions]
If the specification with this option conflicts with the output range, this option outputs an error and performs no processing.
The terminator record (S9 record, S8 record, S7 record) for output varies with the specification of this option. (See Table 15.3-2)

Table 15.3-1 Output S Format Specification List

<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>0x00000000-0x0000FFFF</td>
<td>S9 record</td>
<td>16-bit address</td>
</tr>
<tr>
<td>-S2</td>
<td>0x00000000-0x00FFFFFF</td>
<td>S8 record</td>
<td>24-bit address</td>
</tr>
<tr>
<td>-S3</td>
<td>0x00000000-0x0FFFFFF</td>
<td>S7 record</td>
<td>32-bit address</td>
</tr>
</tbody>
</table>
15.3.2 Output HEX Format Option (-I16/-I20/-I32)

This option specifies the HEX format used to output data.

**Output HEX Format Option (-I16/-I20/-I32)**

**[Format]**

- -I16
- -I20
- -I32

**[Parameter]**

None

**[Description]**

This option specifies the HEX format used to output data. The f2hs command outputs data using either HEX8, HEX16 or HEX32 format. The -I16, -I20, and -I32 options take effect when specified last. If these options are not specified, the f2hs command outputs data in mixed formats of HEX8, HEX16, and HEX32 according to the data address.

**[Precautions]**

If the specification with this option conflicts with the output range, the f2hs command outputs an error and performs no processing.

**Table 15.3-2 List of Output HEX Format Options**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Range of data that can be output</th>
<th>Format</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>-I16</td>
<td>0x00000000-0x0000FFFF</td>
<td>HEX 8 format</td>
<td>16-bit address</td>
</tr>
<tr>
<td>-I20</td>
<td>0x00000000-0x0000FFFF</td>
<td>HEX 16 format</td>
<td>20-bit address</td>
</tr>
<tr>
<td>-I32</td>
<td>0x00000000-0x0000FFFF</td>
<td>HEX 32 format</td>
<td>32-bit address</td>
</tr>
</tbody>
</table>
15.3.3 Specifying to Output Start Address Record (-entry)

When outputting the data the start segment address record or the start linear address record are output. This option can specify using f2hs only.

Specifying to Output Start Address Record (-entry)

[Format]

-entry

[Parameter]

None

[Description]

Use this option to specify the start segment address record or the start linear address record are output. If there is no start address information in the input file, a warning will be output (W1504U: Start address information is not in an input file).

To the Table 15.3-3 described below, the start address record is output according to specify output HEX format option (-I16/-I20/-I32) and the input data range.

Table 15.3-3 The start address records of HEX format output

<table>
<thead>
<tr>
<th>Specifying an Output HEX format</th>
<th>input data range</th>
<th>output start address record</th>
</tr>
</thead>
<tbody>
<tr>
<td>-I16</td>
<td></td>
<td>The warning 'W1503U: -entry option was specified at the time of -I16 specification' is displayed. The start address record does not output.</td>
</tr>
<tr>
<td>-I20</td>
<td>0x00000000-0x000FFFFF</td>
<td>Start segment address record</td>
</tr>
<tr>
<td>-I32</td>
<td>0x00000000-0x000FFFFF</td>
<td>Start segment address record</td>
</tr>
<tr>
<td></td>
<td>0x00100000-0xFFFFF</td>
<td>Start linear address record</td>
</tr>
</tbody>
</table>

[Example]

f2hs ccp903.abs -entry -I16

The warning 'W1503U: -entry option was specified at the time of -I16 specification' is displayed because -entry option specified when the HEX8 format output specification option (-I16) was specified. The start address record does not output.

f2hs ccp903.abs -entry -I20

The start segment address record output.

f2hs ccp903.abs -entry -I32

The start segment address record is output when the data address size is from 0x0 to 0xFFFFF. The start linear address record is output when the data address size is from 0x100000 to 0xFFFFFFFF.
15.3.4 Specifying not to Output Start Address Record (-Xentry)

It specifies to suppress the start segment address record and the start linear address record output. This option can specify only for the f2hs.

- Specifying not to Output Start Address Record (-Xentry)

[Format]

- Xentry

[Parameter]
None

[Description]
It specifies to suppress the start segment address record and the start linear address record output. Use this option when you want to cancel the -entry option.

[Example]

f2hs -entry cpp903.abs -I20 -Xentry

The start address record output specification(-entry) is canceled and the start address record does not output.
15.3.5 Specifying to Adjust (-adjust)

This option automatically calls the adjuster for adjustment after converting the load module.

- Specifying to Adjust (-adjust)

<table>
<thead>
<tr>
<th>Format</th>
<th>-adjust</th>
</tr>
</thead>
</table>

- Parameter
  None

- Description
  This option automatically calls the adjuster for adjustment after converting the load module.
  The starting/ending addresses to be adjusted are automatically set.
  If the starting/ending address parameters are specified with the padding (-p) option when this option is given, an error occurs.
  When this option is specified, the option for the adjuster can also be specified.

- Example
  ```
  f2ms ccp903 -p 0xEF,0x1FE4,0x1FFF -adjust
  
  Because the starting/ending addresses are specified with the padding option, an error occurs.
  
  f2ms ccp903 -p 0xEF -adjust
  
  This option converts the absolute format load module to an adjusted S format. In this case, it pads the location where no data exists with data named 0xEF.
  ```
15.4 f2ms (Converting an Absolute Format Load Module into the S format)

An absolute format load module that is output by a linker is converted into the S format. Data at the addresses 0 through 0xFFFFFFFF is to be converted. The f2ms command can process the absolute format load module of the SOFTUNE V3/V5.

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

[Function]
The f2ms command reads the object data from an absolute format load module that is output by a linker and converts it into a S format file.

Figure 15.4-1 f2ms (Converting an Absolute Format Load Module into the S format)

[Address]
The maximum values of addresses that can be represented in the S format are:

- S1 type 0xFFFF
- S2 type 0xFFFFFF
- S3 type 0xFFFFFFFF

The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive.

Since the S format supports the same range of addresses, conversions can be made without losing any data.

According to the address allocation, f2ms outputs the following records:

- 0x00000000 to 0x0000FFFF : S1 type
- 0x00010000 to 0x00FFFFFF : S2 type
- 0x01000000 to 0xFFFFFFFF : S3 type
15.5  f2hs (Converting an Absolute Format Load Module into the HEX format)

An absolute format load module that is output by a linker is converted into the HEX format.
Data at the addresses 0 through 0xFFFFFFFF is to be converted.
The f2hs command can process the absolute format load module of the SOFTUNE V3/V5.

■ f2hs (Converting an Absolute Format Load Module into the HEX format)

[Function]
The f2hs command reads the object data from an absolute format load module that is output by a linker and converts it into a HEX format file.

Figure 15.5-1  f2hs (Converting an Absolute Format Load Module into the HEX format)

[Address]
The maximum values of addresses that can be represented in the HEX format are:

- HEX8  0xFFFF
- HEX16 0xFFFFF
- HEX32 0xFFFFFFFF

The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive.
Since the HEX format supports the same range of addresses, conversions can be made without losing any data.
According to the address allocation, f2hs outputs the following records:

- 0x00000000 to 0x0000FFFF : HEX8
- 0x00010000 to 0x000FFFFF : HEX16
- 0x00100000 to 0xFFFFFFFF : HEX32
15.6  f2is (Converting an Absolute Format Load Module into the HEX8 format), f2es (Converting an Absolute Format Load Module into the HEX16 format)

The f2is command converts the absolute format load module of the linker output to a HEX8 format, and the f2es converts it to a HEX16 format.

These f2is and f2es commands can also process the absolute format load modules for SOFTUNE V5/V6.

To keep compatibility with the previous versions, these commands are included in the linkage kit. Use of the f2hs command for conversion to a HEX format is recommended.

■ f2is (Converting an Absolute Format Load Module into the HEX8 format)

[Function]

The f2is command only reads the object data part from the absolute format load module of the linker output, and converts it to a HEX8 format file.

[Description]

The common options shown in CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER can be used for f2is.

[Address]

The maximum value of addresses that can be represented in the HEX8 format is 0xFFFF.

Note: The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive. However, when the absolute format load module is converted into the HEX8 format, the data allocated to the addresses 0x10000 and higher are truncated. When using this command, be careful of the range of addresses in the conversion source. An HEX8 format file consists of data records and a trailer record.
f2es (Converting an Absolute Format Load Module into the HEX16 format)

[Function]
The f2es command reads only the object data part from the absolute format load module of the linker output, and converts it to a HEX16 format file.

[Description]
The common options shown in CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER can be used for f2es.

[Address]
The maximum value of addresses that can be represented in the HEX16 format is 0xFFFFF.

Note: The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive. However, when the absolute format load module is converted into the HEX16 format, the data allocated to the addresses 0x100000 and higher are truncated.

When using this command, be careful of the range of addresses in the conversion source.

In the HEX16 format, an extended segment address record is used to represent the addresses 0x10000 and higher.

An extended segment address record in a file is valid until the next extended segment address record appears. If a data record appears without an extended segment address record, then the program calculates addresses assuming that the extended segment address is specified to be 0.

A starting address record is created at the beginning of an HEX16 format file.
CHAPTER 15  LOAD MODULE CONVERTER (f2ms,f2hs,f2ls,f2es)
Chapter 16 describes the conversion formats of format adjuster.

16.1 Outline of the Format Adjuster
16.2 List of Options of the Format Adjuster
16.3 Details of Options of the Format Adjuster
16.1 Outline of the Format Adjuster

The format adjuster sorts data created in the S and HEX formats in the ascending order of addresses, and causes each of the records to contain the specified number of data.

Overview of the Format Adjuster

The format adjuster causes each of the records in one file of the format to contain the specified number of data to unify the format. Figure 16.1-1 shows the concept of the format adjuster.

The portion of an input format file containing no data is padded with 0xFF (default value). The output file is the contents of memory converted into the S and HEX formats.

Use the padding option (-p option) to pad the portion of an input 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 format file has records of varying lengths.

Figure 16.1-2 Example of operation of the Format Adjuster

<table>
<thead>
<tr>
<th>Before conversion</th>
<th>After conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>S007000054455354B8</td>
<td>S00700054455354B8</td>
</tr>
<tr>
<td>S20CFF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S209FF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S20FF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S20FFFF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S20CFF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S209FF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S20FFFF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S20CFF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S209FF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
<tr>
<td>S20FFFF00000109572C160C2D2C6C</td>
<td>S31500FF00000109572C160C2D2C6C</td>
</tr>
</tbody>
</table>

■ Functions of the Format Adjuster

The format adjuster has the following functions:

- The data is sorted in the ascending order of addresses.
- The portion of the specified range of addresses containing no data is padded with the specified data at the time of startup (with 0xff by default).
- As the starting address of a record, specify a value coordinated with the output data length specified at the time of startup.
  - If a remainder exists when the starting address specified for output is coordinated with the data length specified at the time of startup (if the data length in a record is 16 bytes and the starting address is not a multiple of 16), the first record appearing in the output information stores the data from the specified starting address to the address coordinated with the specified data length.
  - The second and later record starting addresses become the one coordinated with the specified length.
- If the input format information contains multiple terminator records, the entry address of the terminator record that appears at last is converted and output.
  Any other terminator records are deleted.
  - If the value of the entry address defined in the conversion source data is not within the range of data after conversion, 0 is set in the terminator record after conversion.
The following section shows an example of converting the address range 0xff0008 through 0xff004a at the record length of 16.

**Figure 16.1-3 Example of Conversion of the Format Adjuster**

<table>
<thead>
<tr>
<th>Before conversion</th>
<th>After conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>S007000054455354B8</td>
<td>S007000054455354B8</td>
</tr>
<tr>
<td>S20CFF0000109572C160C2D2CEC</td>
<td>S30D00FF0008020406080A020406C1</td>
</tr>
<tr>
<td>S20FF000020406080A0C0E1012141660</td>
<td>S31500FF0010080A0C0E101214160109572C160C2D2C5B</td>
</tr>
<tr>
<td>S20FF0000300109572C160C2D2CD4</td>
<td>S31500FF0020020406080A0C0E1012141629</td>
</tr>
<tr>
<td>S20FF000040020406080A0C0E1012141648</td>
<td>S31500FF00300109572C160C2D2CBC</td>
</tr>
<tr>
<td>S20FF000050030020406080A0C0E1012141630</td>
<td>S20FF000060040020406080A0C0E1012141618</td>
</tr>
<tr>
<td>S20FF000060040020406080A0C0E1012141618</td>
<td>S20FF00007000000000FA</td>
</tr>
</tbody>
</table>
16.2 List of Options of the Format Adjuster

Section 16.2 lists the names and functions of options of the format adjuster.

List of Options of the Format Adjuster

Table 16.2-1 lists the options of the format adjuster.

Table 16.2-1 List of Options of the Format Adjuster

<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>Default 16</td>
</tr>
<tr>
<td>Specifying the output range</td>
<td>-ran</td>
<td>need</td>
</tr>
<tr>
<td>Specifying the S1 format output</td>
<td>-S1</td>
<td>Only m2ms</td>
</tr>
<tr>
<td>Specifying the S2 format output</td>
<td>-S2</td>
<td>Only m2ms</td>
</tr>
<tr>
<td>Specifying the S3 format output</td>
<td>-S3</td>
<td>Only m2ms</td>
</tr>
<tr>
<td>Specifying the HEX8 format output</td>
<td>-I16</td>
<td>Only h2hs</td>
</tr>
<tr>
<td>Specifying the HEX16 format output</td>
<td>-I20</td>
<td>Only h2hs</td>
</tr>
<tr>
<td>Specifying the HEX32 format output</td>
<td>-I32</td>
<td>Only h2hs</td>
</tr>
<tr>
<td>Specifying to change the starting address</td>
<td>-ST</td>
<td></td>
</tr>
<tr>
<td>Specifying not to read default option file</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to read option file</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to display of help message</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to output version number and message</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output version number and message</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to output end message</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output end 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>
Section 16.3 describes the options of the format adjuster. For information on the common options of linkage kit, see CHAPTER 3, "COMMON OPTIONS". For information on the common options of a converter, see CHAPTER 14, "COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER".

### Specifying the output record data length (-len)
This option specifies the output record data length. For details, see section 16.3.1 Specifying the Data Length in an Output Record (-len).

### Specifying the output range (-ran)
This option specifies the range of formatting an format. For details, see section 16.3.2 Specifying the Output Range (-ran).

### Specifying an output S format (-S1/-S2/-S3)
This option specifies an record to be used when data is output in the S format using m2ms. For details, see section 16.3.3 Specifying an Output S format (-S1/-S2/-S3).

### Specifying an Output HEX format (-I16/-I20/-I32)
Specifies the record used to output with Hex format using h2hs. For details, see section 16.3.4 Specifying an Output HEX format (-I16/-I20/-I32).

### Specifying to change the starting address of record (-ST)
This changes the starting address of the S record used when outputting in the S format. For details, see section 16.3.5 Specifying Changes to the Starting Address (-ST).
16.3.1 Specifying the Data Length in an Output Record (-len)

Use this option to specify the number of data to be output into a record of an output format.

Specifying the data length in an output record (-len)

[Format]

-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 format is formatted.
Specify 16, 32, 64, or 128 as the data length.
If this option is omitted, 16 is assumed to be specified in the processing.

[Precautions]
This option specifies the number of bytes of data contained in one record, not the record length itself.

[Example]

m2ms sfmtfile.mhx -len 32
sfmtfile.mhx is formatted and 32-byte data is output per record.
m2ms sfmtfile.mhx (An example of omitting the -len specification)
sfmtfile.mhx is formatted and 16-byte data is output per record.
m2ms sfmtfile.mhx -len 96
An error occurs because the specified data length is out of the specifiable range.
m2ms sfmtfile.mhx -len (An example of omitting all the parameters)
An error occurs because the data length specification is omitted.
16.3.2 Specifying the Output Range (-ran)

Use this option to specify the range of formatting using address.

### Specifying the output range (-ran)

**Format**

-ran <Starting address> [ , <Ending address>]

**Parameters**

- **<Starting address>**  
  Starting address
- **<Ending address>**  
  Ending address

**Description**

Use this option to specify the range of formatting using address.

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 sfmfile.mhx (An example of not using the -ran option)
An error occurs because the output range is not specified.
m2ms sfmfile.mhx -ran 0xD000,0xFFFF
The data in sfmfile.mhx at the addresses 0xD000 through 0xFFFF is formatted.
m2ms sfmfile.mhx -ran 0xD000 (An example of omitting the ending address)
The data in sfmfile.mhx as much as 64Kbytes from the addresses 0xD000 (0xD000 through 0x1CFFF) is formatted.
m2ms sfmfile.mhx -ran 0xFFFF,0xD000
An error occurs because the specified ending address is smaller than the starting address.
m2ms sfmfile.mhx -ran (An example of omitting all the parameters)
An error occurs because the starting address is omitted.
```
16.3.3 Specifying an Output S format (-S1/-S2/-S3)

Use this option to specify an S format to be used when data is output. This option is for the S format adjuster (m2ms).

- ■ Specifying an output S format (-S1/-S2/-S3)

  [Format]
  
  -S1
  
  -S2
  
  -S3

  [Parameter]
  
  None

  [Description]
  
  Use this option to specify an record to be used when data contents are output.
  
  The S format adjuster outputs the data contents using one of the S1, S2, and S3 records.
  
  It never outputs data using both the S1 and S2 records.
  
  If more than one of the -S1, -S2, and -S3 options are specified, the one most recently specified is valid.
  
  If none of the -S1, -S2, and -S3 options are specified, the S format adjuster outputs the data contents into the S3 record.

  Note:
  
  If the specification in this option and the output range are not consistent with each other, the S format adjuster reports an error and performs no processing.
  
  Specifying this option changes the terminator record to be used for output (S9, S8, and S7 records). (See Table 16.3-1 )

  Table 16.3-1 List of output record specifications

<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>0x000000-0xFFFFFFFF</td>
<td>S8 record</td>
<td></td>
</tr>
<tr>
<td>-S3</td>
<td>0x00000000-0xFFFFFFFF</td>
<td>S7 record</td>
<td>(Default)</td>
</tr>
</tbody>
</table>
[Example]

m2ms sfmtfile.mhx -ran 0xD000,0x10000 -S1
An error occurs because the output range is up to 0x10000, which cannot be represented in the S1 record.

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

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

m2ms sfmtfile.mhx -ran 0xE000,0xFFFF -S3
The data from 0xE000 through 0xFFFF is formatted and output in the S3 record.
16.3.4 Specifying an Output HEX format (-I16/-I20/-I32)

Use this option to specify a HEX format to be used when data is output. This option is for the HEX format adjuster (h2hs).

- **Specifying an output HEX format (-I16/-I20/-I32)**

  **[Format]**

  - -I16
  
  - -I20
  
  - -I32

  **[Parameter]**

  None

  **[Description]**

  Use this option to specify a record to be used when data contents are output. The HEX format adjuster outputs the data contents using one of the HEX8, HEX16, and HEX32 formats. If more than one of the -I16, -I20, and -I32 options are specified, the one most recently specified is valid. If none of the -I16, -I20, and -I32 options are specified, the HEX format adjuster outputs the data contents into the HEX32 format.

  **[Precautions]**

  If the specification in this option and the output range are not consistent with each other, the HEX format adjuster reports an error and performs no processing.

  **[Example]**

  h2hs hfmtfile.hex -ran 0xD000,0x10000 -I16

  An error occurs because the output range is up to 0x10000, which cannot be represented in the HEX8 format.

  h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I16

  The data from 0xE000 through 0xFFFF is formatted and output in the HEX8 format.

  h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I20

  The data from 0xE000 through 0xFFFF is formatted and output in the HEX16 format.

  h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I32

  The data from 0xE000 through 0xFFFF is formatted and output in the HEX32 format.
16.3.5 Specifying Changes to the Starting Address (-ST)

This specifies the starting address used when outputting data. This is used to change the address of the data.

- Specifying Changes to the Starting Address (-ST)

  **[Format]**

  -ST <Starting address>

  **[Parameter]**

  <Starting address>
  
  Starting address

  **[Description]**

  This specifies the starting address used when outputting data.

  The format adjuster determines the starting address of data normally using the starting address specified by the output range specification (-ran).

  Specifying this option changes the starting address when outputting.

  **[Example]**

  m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST 0x0000
  
  This forms the data in the sfmtfile.mhx from 0xD000 to 0xFFFF address. It outputs this as data from 0x0000 address.

  m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST 0x10000
  
  This forms the data in the sfmtfile.mhx from 0xD000 to 0xFFFF address. It outputs this as data from 0x10000 address.

  m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST
  
  (Example where parameters are omitted.)

  There is an error where the starting address is omitted.
Chapter 17 describes the conversion formats of binary converter.

17.1 Outline of Binary Converter
17.2 List of Options of Binary Converter
17.3 Details on Options of the Binary Converter
17.1 Outline of Binary Converter

The binary converter converts files output by the S format or HEX format into binary data files. It not only simply converts to binary data, but also supports a split mode that separates files into several files for output.

Outline of the Binary Converter

The binary converter converts object files of the S format or HEX format made by the linkage kit into binary data (memory images) and outputs them to files.

Use m2bs to convert the S format into binary data; use h2bs to convert the HEX format into binary data.

It is possible to specify a multiple of input files (S format or Hex format). Also, it is possible to separate converted binary data into a multiple of files of specified byte sizes (hereinafter referred to as the split mode).

Figure 17.1-1 Overview of the Binary Converter

Note: The portion of an input file containing no data is padded with 0xFF (default value). The binary data file to be output is the contents of memory that is output into a file without changes. Use the padding option (-p) to pad with a specified value the portion of an input file containing no data. For details on how to use this option, see Section 14.3, "Padding (-p)."

The default extension of an output binary file is .bin. In the split mode, the extension .bxx (xx is a two digit number (01 through 16)) is unconditionally added.
**Overview of the split mode**

A split mode means that the memory image converted by binary converter is split for specified bytes and output into multiple binary data file.

Figure 17.1-2 shows an overview of the split mode. In Figure 17.1-2, every byte of data is output in turns into two files. In the split mode, every specified byte of data may be output in turns into sixteen files maximum.

Use the -sp option to specify the split mode. For details on how to use the -sp option, see Section 17.3.2, “Specifying the Split Mode (-sp)”.

![Figure 17.1-2 Overview of the split mode](image)
17.2 List of Options of Binary Converter

Section 17.2 lists the option names and functions of binary converter.

Table 17.2-1 lists options of the binary converter.

**Table 17.2-1 List of Options of the 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>need</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 number and messages</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output the version number 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>
17.3 Details on Options of the Binary Converter

Section 17.3 describes the options of the binary converter. For information on the common options for linkage kit, see Chapter 3, "COMMON OPTIONS". For information on the common options for a converter, see Chapter 14, "COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER".

- **Specifying the output range (-ran)**
  This option specifies the range of an S format or HEX format to be converted to a binary image. For details, see Section 17.3.1, "Specifying the Output Range (-ran)".

- **Specifying the split mode (-sp)**
  This option specifies that a binary image is output in the split mode. For details, see Section 17.3.2, "Specifying the Split Mode (-sp)".

- **Specifying the inhibition of the split mode (-Xsp)**
  This option nullifies the split mode specification (-sp). For details, see Section 17.3.3, "Specifying the Inhibition of the Split Mode (-Xsp)".

- **Specifying to create a map list file (-m)**
  This option specifies that conversion information is output in a map list file. For details, see Section 17.3.4, "Specifying to Create a Map List File (-m)".

- **Specifying not to create a map list file (-Xm)**
  This option nullifies the specification to create a map list file (-m). For details, see Section 17.3.5, "Specifying not to Create a Map List File (-Xm)".
17.3.1 Specifying the Output Range (-ran)

Use addresses to specify the range to be converted to a binary image. This option must be specified for binary converter.

- Specifying the output range (-ran)

  [Format]

  - 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 (An example of omitting all the parameters)
  An error occurs because the starting address is omitted.
17.3.2 Specifying the Split Mode (-sp)

Specify this option to output a binary image in the split mode.

- Specifying the split mode (-sp)

  [Format]
  
  - sp < Number of output files > [, < Number of bytes > ]

  [Parameters]
  
  <Number of output files>
  Specifies how many output destination files 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 65Kbytes data into two 32Kbytes files, the last 1Kbytes data is output into the former file.

  If this option is specified, the extension ".bxx" (xx is a two-digit number between 01 and 16 inclusive) is unconditionally added to the output file.

  [Example 1]
  
  m2bs sfmtfile.mhx (Example of not using the -sp option)

  A binary image is output into sfmtfile.bin.

  [Example 2]
  
  m2bs sfmtfile.mhx -sp 2

  Every one byte of a binary image is output in turns into sfmtfile.b01 and sfmtfile.b02.

  [Example 3]
  
  m2bs sfmtfile.mhx -sp 2,2

  Every two bytes of a binary image is output in turns into sfmtfile.b01 and sfmtfile.b02.
17.3.3 Specifying the Inhibition of the Split Mode (-Xsp)

The -Xsp option nullifies the split mode specification (-sp).

Specifying the inhibition of the split mode (-Xsp)

[Format]

- Xsp

[Parameter]
None

[Description]
Specify this option to nullify the -sp specification.
This option needs not be specified in particular because it is the default.

[Example]

m2bs ccp903.mhx -ran 0xE000,0xFFFF
m2bs cpp903.mhx -Xsp -ran 0xE000,0xFFFF

The default processing does not run in the split mode.
The above specifications are equivalent.

m2bs -f option.file ccp903 -Xsp

You may sometimes want to temporarily change the specification in an option file when the option file is used to execute the program.
If the -sp option exists in option.file, there is no need to change the contents of option.file. Simply specify the -Xsp option on a command line to nullify the -sp option.
17.3.4 Specifying to Create a Map List File (-m)

Use this option to output the information at the time of conversion into a map list file.

- Specifying to create a map list file (-m)

[Format]
- m <Map list file name>

[Parameter]
<Map list file name>
Output map list file name

[Description]
Use this option to output information at the time of conversion into a map list file.
The information at the time of conversion is output into a map list file. The following information is output. Item (4) is output only if the -sp option is specified.
1. Input file name information
2. Output file name information
3. Output range information
4. Split unit information
5. Padding data value information
If the <Map list file> has no extension, the default extension ".mp3" is added.

[Example 1]
```
m2bs 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.

**Figure 17.3-1 Example 1 of 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 sfmfile.mhx -ran 0x10000,0x1FFFF -m logfile -sp 2,2
```
The information at the time of conversion is output into logfile.mp3.

**Figure 17.3-2 Example 2 of Contents of logfile.mp3**

<table>
<thead>
<tr>
<th>Input file</th>
<th>sfmfile.mhx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output file</td>
<td>sfmfile.b01</td>
</tr>
<tr>
<td>Output file</td>
<td>sfmfile.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>
17.3.5 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)**

[Format]

```
- Xm
```

[Parameter]
None

[Description]
Specify this option to nullify the -m specification.
There is no need to specify this option because it is the default.

[Example 1]
```
m2bs  ccp903.mhx -ran 0xE000,0xFFFF
m2bs  ccp903.mhx -Xm -ran 0xE000,0xFFFF
```

A map list file is not output by default.
The above specifications are equivalent.

[Example 2]
```
m2bs -f option.file ccp903 -Xm
```

You may sometimes want to temporarily change the specification in an option file when the option file is used to execute the program
If the -m option exists in option.file, you need not change the contents of option.file but simply specify the -Xm option on a command line to nullify the -m option.
Chapter 18 describes the commands of other converters in detail.

18.1 m2is (Converting a S Format File into the HEX8 Format)
18.2 m2es (Converting a S Format File into the HEX16 Format)
18.3 i2ms (Converting a HEX8 Format File into the S Format)
18.4 e2ms (Converting a HEX16 Format File into the S Format)
18.1 m2is (Converting a S Format File into the HEX8 Format)

A S format file is converted into the HEX8 format. Data at the addresses 0 through 0xFFFF is to be converted.

- m2is (Converting a S Format File into the HEX8 Format)

  [Function]

  The m2is command converts a S format file into the HEX8 format.

  ! Figure 18.1-1 m2is (Converting S Format file into HEX8 Format)

  S format file → m2is → HEX8 format file

  (.mhx) → (.ihx)

  Note: The S format can represent the addresses 0 through 0xFFFFFFFF. However, when converted into the HEX8 format, the data allocated at the addresses 0x10000 and higher is truncated. When using this command, be careful of the range of addresses in the conversion source. Since the HEX8 format file consists of data records and a trailer record, the starting address information in the S format will be lost.
18.2  m2es (Converting a S Format File into the HEX16 Format)

A S format file is converted into the HEX16 format.
The data at the addresses 0 through 0xFFFFF is to be converted.

m2es (Converting a S Format File into the HEX16 Format)

[Function]
The m2es command converts a S format file into the HEX16 format.

Figure 18.2-1  m2es (Converting S Format file into HEX16 Format)

Note: The S format can represent the addresses 0 through 0xFFFFFFFF. However, when converted into the HEX16 format, the data allocated at the addresses 0x100000 and higher is truncated. When using this command, be careful of the range of addresses in the conversion source.
A HEX8 format is converted into the S format. The data at the addresses 0 through 0xFFFF is to be converted.

### i2ms (Converting a HEX8 Format File into the S Format)

**[Function]**

The i2ms command converts a HEX8 format file into the S format.

**Figure 18.3-1  i2ms (Converting HEX8 Format file into S Format)**

HEX8 format file → i2ms → S format file

(.ihx) → (.mhx)

Note: Although the S format can represent the addresses 0 through 0xFFFFFFFF, the HEX8 format cannot represent the addresses 0x10000 and higher. A S format file after conversion is created without the S2, S3, S7, and S8 types. Since the HEX8 format does not have starting address information, the starting address after conversion is to be 0.
18.4 **e2ms (Converting a HEX16 Format File into the S Format)**

A HEX16 format is converted into the S format. The data at the addresses 0 through 0xFFFFF is to be converted.

**Figure 18.4-1 e2ms (Converting HEX16 Format file into S Format)**

Note: Although the S format can represent the addresses 0 through 0xFFFFFFFF, the HEX16 format cannot represent the addresses 0x100000 and higher. A S format file after conversion is created without the S7 and S3 types. The starting address information in the HEX16 format is set in the S9 or S8 type.
Chapter 19 describes restrictions and questions and answers on using an object format converter.

19.1 Restrictions on an Object Format Converter
19.1 Questions and Answers on Using an Object Format Converter
19.1 Restrictions on an Object Format Converter

There are several restrictions in the binary converter and format adjuster. No other restrictions have been created for processing when using the other object format converters. It is possible to process using the entire memory that the object format converter can use in execution.

### Restrictions on a object format converter

Table 19.1-1 shows the restrictions when using the object format converter. However, this is not the maximum limit value available for processing. Object format converter performs processing, gaining a memory dynamically. Object format converter outputs an error message with an insufficient memory, when gaining a memory required for processing becomes impossible, and processing is interrupted.

<table>
<thead>
<tr>
<th>Item</th>
<th>Restriction Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option file count</td>
<td>Limitless</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Option file internal line count</td>
<td>Limitless</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Option file internal character count per 1 line</td>
<td>Limitless</td>
<td>Memory dependent</td>
</tr>
<tr>
<td>Option file nest</td>
<td>Not possible</td>
<td></td>
</tr>
<tr>
<td>Input file count (m2bs,m2ms,h2bs,h2hs)</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Input file count (expected above)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I/O file size</td>
<td>Limitless</td>
<td>OS dependent</td>
</tr>
<tr>
<td>I/O file Line Count</td>
<td>Limitless</td>
<td>OS dependent</td>
</tr>
<tr>
<td>File name character count</td>
<td>Limitless</td>
<td>OS dependent</td>
</tr>
<tr>
<td>Maximum Memory Address</td>
<td>0xFFFFFFFF</td>
<td></td>
</tr>
<tr>
<td>Maximum convert size (m2bs,m2ms,h2bs,h2hs)</td>
<td>2GB - 1</td>
<td>Memory dependent</td>
</tr>
</tbody>
</table>

### Cautionary Information Concerning Binary Converter and Format Adjuster

You can specify up to a total of 64 items for the input file.
If you have set a number of input files, they are processed in order their being set.
If there is data for the same address in the input file, the subsequent data overwrites the antecedent data.
You can convert a maximum of 2G-1 bytes at one time.
19.2 Questions and Answers on Using an Object Format Converter

Section 19.2 covers the questions and answers on using an object format converter.

### Questions and answers on using an object format converter

| Q. | There are many converters available. Which one should I use? |
| A. | A converter is used to convert an absolute format load module file that is output by a linker to an object format that can be read by a ROM writer. It is recommended to use f2ms converting to the S format and f2hs to the HEX format because the f2ms and f2hs fully support the 32bits addressing space. Use other conversion tools as required. |
| Example. | f2ms absfile.abs -> Outputs absfile.mhx in the S format |

| Q. | When I use binary converter and format adjuster, an error, "F9001U: Insufficient memory" is output and I cannot convert data to a binary image. What should I do? |
| A. | Binary converter and format adjuster secure as much memory as the area to be converted. If you try to convert a large area to the memory image at once, an error, "F9001U: Insufficient memory" may be output and the processing interrupted. In such a case, split the target area into multiple continuous areas and create a memory image for each of them. Then, merge the files into one binary image. |
| Example. | Converting the binary image area from 0xC00000 through 0xFFFFFE |
| | - If enough memory can be secured (Normal): |
| | 1 m2bs absfile.mhx -ran 0xC00000,0xFFFFFE |
| | - If an error, "F9001U: Insufficient memory" is output: |
| | 1 m2bs absfile.mhx -ran 0xC00000,0xDFFFFF -o absfile1.bin |
| | 2 m2bs absfile.mhx -ran 0xE00000,0xFFFFFE -o absfile2.bin |
| | 3 copy /b absfile1.bin + absfile2.bin absfile.bin |
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  HEX Format
APPENDIX C  S Record Format
APPENDIX D  List of Linker Options
APPENDIX E  List of Librarian Options
APPENDIX F  List of Commands and Options of the Object Format Converter
APPENDIX G  Specification Differences Depending on the OS
APPENDIX 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 have occurred. The causes of errors must be removed before rerun.

● Fatal error

Errors which make it impossible to continue processing. This type of errors results from wrong specifications of the user or problems of the execution environment.
## Linkage Kit Error Message Display Format

Error messages are output by each tool in the following format.

*** File name (line number)  XnnnT:  Message text (supplementary message)

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

[Examples]

*** sample.c(234) E4329L: Value out of range (0xFFFE37D4)
This is an example in which the source file name and line number are also displayed.

*** E4402U: Duplicated module name (date.obj setdate)
This is an example in which neither the source file name nor line number is displayed.

*** F9001U: Insufficient memory
This is an example in which neither the source file name nor line number nor supplementary message is displayed.
## Error Messages of the Linker

<table>
<thead>
<tr>
<th>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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>This is a message which is notified when 2 is specified in the -w option.</td>
</tr>
<tr>
<td></td>
<td>The error is slighter than W1351.</td>
</tr>
<tr>
<td></td>
<td>The message only reports 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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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 &quot;section name&quot; area</td>
</tr>
<tr>
<td></td>
<td>Optimal section allocation was carried out in the area indicated here.</td>
</tr>
<tr>
<td></td>
<td>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

The file indicated here is a module for a different CPU type with downward compatibility.

I0307L  Lower compatible cpu type object  (file name)

Files indicated here are modules generated using the SOFTUNE V5 tools. This is a message which is notified when 2 is specified in the -w option.

I0310L  Softune V5 type object  (file name)

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

W1301L  Writable section located in ROM area  (section name)

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

W1303L  Section with initial data located in RAM area  (section name)

A section exists which is allocated outside the address range specified by using the -ro or -ra option exists. 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.

W1305L  Section  (section name)  located on out of ROM/RAM area  (area name)

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

W1306L  Exceeded maximum address  (section name)

Sections with the same name and with different attributes or types exist in multiple modules.

W1307L  Duplicate section name exist  (section name)
### APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1308L</td>
<td>Overlap located section (section 1, section 2)</td>
</tr>
<tr>
<td>W1314L</td>
<td>Specified address to absolute section (section name)</td>
</tr>
<tr>
<td>W1320L</td>
<td>Not search library at mode-identification mode</td>
</tr>
<tr>
<td>W1321L</td>
<td>Ignore (option) Option at mode-identification mode</td>
</tr>
<tr>
<td>W1325L</td>
<td>Entry point already set</td>
</tr>
<tr>
<td>W1326L</td>
<td>Entry point was changed</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.

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

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

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.

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

The entry point already set has been changed by specifying the `-e` option.
APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1327L</td>
<td>Duplicate symbol definition  (mangle name / symbol name)</td>
</tr>
</tbody>
</table>

The same external definition symbol exists in multiple input modules.
The symbol value defined first is valid.

| W1328L | Mismatch symbol type  (mangle name / 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.

| 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.

| W1367L | Duplicated module name  (file name module name)                       |

The same module name already exists as that intended for linking.
Duplicated module names are not allowed because the problem, such as the correct modules is not specified at the time of creating the absolute assemble list and debugging is generated.
Please, change the module names.
The C/C++ compiler automatically generates the module name from the file name.
Please, change the file names.

| 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.
## Error Messages of the Linkage Kit

**APPENDIX A**

### 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 (mangle name / 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 (mangle name / symbol name)

Because the definition of the external definition symbol could not be found, relocation could not be carried out.

This message is notified if the external definition symbol specified by the `-df` option cannot be found.

It is necessary to combine the modules contained the external definition symbol.

### 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.

### E4326L Different symbol type (mangle name / symbol name)

This message is notified if the external definition symbol specified when an entry point was set using the `-e` option is neither function name nor variable name nor address label name.

### E4327L Illegal RL information

This message is notified if any error in relocation information is found.
### APPENDIX A  Error Messages of the Linkage Kit

#### E4329L
Value out of range  (value)

Overflow occurred in relocation operations.
This message contains the source program name including the description of data for relocation and the line numbers. Use this information to check the program.
If you can use the tag jump of your editor, you can jump to the corresponding source lines immediately.

#### E4330L
Devided by 0

The divisor in the division for relocation operations is 0.

#### E4331L
Indispensable to locate address  (section name)

This message is notified if there is no addressing by the -sc option or a section without any specification is found.

#### E4332L
Not handling group name  (group name)

A group name cannot be specified when a ROM \(\rightarrow\) RAM transfer section is specified using the -sc option.

#### E4333L
Not specified ROM address  (section name)

No ROM address is specified for relocation information to be allocated in the ROM area.
Specify a ROM address using the -sc option.

#### E4351L
Relocatable assemble list not correspond to object file  (file name)

No object corresponding to the relative assemble list file indicated by the supplementary message is found.
The linker stops processing of the list file indicated by the supplementary message, and proceeds with processing of the next list file.
APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4352L</td>
<td>Relocatable assemble list file not found</td>
</tr>
<tr>
<td></td>
<td>(file name)</td>
</tr>
</tbody>
</table>

The relative format assemble list file indicated by the supplementary message is not found. The linker does not create an absolute assemble list file corresponding to the file name indicated by the supplementary message, and continues processing.

| E4354L | Illegal relocatable assemble list file format |
|        | (file name)                                  |

The linker cannot process the format of the relative assemble list file indicated by the supplementary message. The linker stops processing of the list file indicated by the supplementary message, and proceeds with processing of the next list file.

| E4355L | Object data not correspond                   |
|        | (file name)                                  |

Object data of the absolute format load module file and source data of the relative format assemble list indicated by the supplementary message do not match. Rerun the program after reassembling.

| E4357L | Too many array dimension or structure nested |
|        | exceeded 8                                  |

The number of dimensions of an array output to the ARRAY list or that of nesting levels of a structure exceeds 8. Processing continues, but those parts which exceed this limit are not output.

| E4362L | DUMMY section specified                     |
|        | (file name)                                  |

A dummy section is specified in a file indicated by the supplementary message. Do not write any dummy section when creating an absolute format assemble list.

| E4363L | Exceeded maximum section size                |
|        | (section name)                               |

One or more sections which exceed the maximum size of a section exist. Some sections are have the maximum size depending on their section identification. This message is output if such limits are exceeded. Examine such sections.
APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4365L</td>
<td>Not found locatable address in area name (section name)</td>
</tr>
<tr>
<td></td>
<td>No place in the specified allocation area where the applicable section can</td>
</tr>
<tr>
<td></td>
<td>be allocated can be found. Examine the section configuration in the area.</td>
</tr>
<tr>
<td>E4366L</td>
<td>Not found locatable address (section name)</td>
</tr>
<tr>
<td></td>
<td>No place in all address space where the section can be allocated can be</td>
</tr>
<tr>
<td></td>
<td>found. 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.</td>
</tr>
<tr>
<td></td>
<td>Duplicated module names are not allowed. Change the module names.</td>
</tr>
<tr>
<td>E4369L</td>
<td>Invalid module: conflict compile model (file name)</td>
</tr>
<tr>
<td></td>
<td>A module of a different compile model cannot be linked.</td>
</tr>
<tr>
<td>E4370L</td>
<td>CPU information file not found (file name)</td>
</tr>
<tr>
<td></td>
<td>The target CPU information file specified by the -cpu option cannot find.</td>
</tr>
<tr>
<td></td>
<td>This is detected when the file below is not found.</td>
</tr>
<tr>
<td></td>
<td>• %FETOOL%\LIB\935\935.CSV</td>
</tr>
<tr>
<td></td>
<td>• %FETOOL%\LIB\935\cpu_info*.CSV</td>
</tr>
<tr>
<td>E4371L</td>
<td>CPU information not found (file name)</td>
</tr>
<tr>
<td></td>
<td>The target CPU information, in the CPU information file, specified by the</td>
</tr>
<tr>
<td></td>
<td>-cpu option cannot find. This is detected when the target CPU information</td>
</tr>
<tr>
<td></td>
<td>specified in the file below is not found.</td>
</tr>
<tr>
<td></td>
<td>• %FETOOL%\LIB\935\935.CSV</td>
</tr>
<tr>
<td></td>
<td>• %FETOOL%\LIB\935\cpu_info*.CSV</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>E4701L</td>
<td>Symbol is referenced as an explicit specialization and a generated instantiation (symbol name)</td>
</tr>
<tr>
<td>E4702L</td>
<td>Error occurred during symbol name decoding (symbol name)</td>
</tr>
<tr>
<td>F9001L</td>
<td>Insufficient memory</td>
</tr>
<tr>
<td>F9011L</td>
<td>Input file not found (file name)</td>
</tr>
<tr>
<td>F9012L</td>
<td>Library file not found (file name)</td>
</tr>
<tr>
<td>F9015L</td>
<td>File open error (file name)</td>
</tr>
</tbody>
</table>

The same function as the template function to which substance is generated is defined.

The error occurred during the decoding of the symbol name. The symbol name is used as it is.

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.

The specified input file cannot be found.

If the name of the file indicated here is a default library file, the following things are the causes.

When the environment variable LIB935 is set up.
- The library file is not stored in the directory indicated by the environmental variable LIB935.

When the environment variable LIB935 is not set up.
- The library file is not stored in the directory derived from the environmental variable FETOOL.

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.
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 linker. This message is also notified if a write-protected file with the same name exists.

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.

The option cannot be used by the linker. See the -help option or this manual.

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.

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.
### APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9026L</td>
<td>Specified value out of range (value)</td>
</tr>
<tr>
<td>F9027L</td>
<td>Option file nested</td>
</tr>
<tr>
<td>F9030L</td>
<td>Missing input file name</td>
</tr>
<tr>
<td>F9032L</td>
<td>Output file name same as input one (file name)</td>
</tr>
<tr>
<td>F9033L</td>
<td>Illegal file format (file name)</td>
</tr>
<tr>
<td>F9040L</td>
<td>Duplicated file or path name (file name)</td>
</tr>
</tbody>
</table>

A value is specified which is outside the range of allowed values by the `-pl`, `-pw`, or `-w` option is specified.

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

Specify an input file.

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

This message is notified in one of the following cases:
- The library file format is not correct.
- The object module format in a library file is not correct.
- The input file is an absolute format load module.
- Contents of an input module are not correct.
- The format of a relative format assemble list file is not correct.
- CPU Information file format is not correct.

This message is notified in one of the following three cases:
- Input module files of the same name are specified.
- Library files of the same name are specified in the `-l`, or `-el` option.
- The same library path name is specified in the `-L` option more than once.
APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9042L</td>
<td>Duplicated section name (section name)</td>
</tr>
<tr>
<td>F9043L</td>
<td>Duplicated symbol name (mangle name / symbol name)</td>
</tr>
<tr>
<td>F9044L</td>
<td>Duplicated section group name (group name)</td>
</tr>
<tr>
<td>F9047L</td>
<td>No match file name argument</td>
</tr>
<tr>
<td>F9052L</td>
<td>Missing '-cpu' option</td>
</tr>
<tr>
<td>F9053L</td>
<td>Missing '-ro' or '-ra' option</td>
</tr>
<tr>
<td>F9054L</td>
<td>Duplicate section name exist (section name)</td>
</tr>
</tbody>
</table>

The same section name was specified in the -sc or -gr option more than once.

The same external reference symbol was specified in the -el option more than once.

The same group name was used more than once when setting groups using the -gr option.

A file was specified using the wild card, but no corresponding file was found.

No target CPU is specified using the -cpu option.

It is necessary to specify a target CPU by the -cpu option.

The -ro option or -ra option required for automatic allocation is not specified.

Sections with the same name and with different attributes or types exist in multiple modules.

This error is output to "Absolute Assemble List Output".

The error occurred by pre-link process.
The CPU information file is not suited old.
Please obtain the CPU information file of the latest.

The description of the directory in the Instantiation information file is not correct.
Please compile each object again, update the instantiation information file, and execute the linker again.

Linker uses a C/C++ compiler, in order to generate a template function, in case the object file created by C++ is linked.
In the following cases, the message is outputted.
The input file format (.obj) is not correct.
The memories for starting a C/C++ compiler are insufficient.
The C/C++ compiler is outputting the error.

The instantiation of the same symbol is specified with another instantiation information file.
Please compile each input file again, create the instantiation information file, and execute the linker again.

The instantiation information file is not found though the instantiation is necessary for input file (*.obj).
Please compile each input file again, create the instantiation information file, and execute the linker again.
The message files used by the linker could not be opened. Store the error message files in a predetermined directory.

- `lkt935_a.msg` or `lkt935_e.msg`

<table>
<thead>
<tr>
<th>F9998L</th>
<th>File open failed (file name)</th>
</tr>
</thead>
</table>

F9999L Internal error (identification information)

If this error occurs, report it to Fujitsu immediately.
■ Error Messages for the Librarian

| I0401U | Reference to undefined symbol |
---|---|

This is a message reported if the -c option is specified. This message indicates that, after checking external symbols in the library file, external reference symbols that cannot be resolved in the library file are contained. When a library file for which this message is output is used in the linker, care must be taken to know to which module the external definition symbols to be used belong.

| I0402U | Debug information exists |
---|---|

This is a message reported if the -c option is specified. This library file contains a module which includes 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.

| I0410U | Softune V5 type object (file name) |
---|---|

Files indicated here are modules generated using the SOFTUNE V5 tools.

| I0411U | This is an old-format library file. (file name) |
---|---|

The file displayed here is a library made by the SOFTUNE or V5 tool. A backup file is automatically made for the library file that has not been edited yet. The backup file's extension is .bak.

| 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.
### APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1402U</td>
<td>Ignore <code>-pw</code> option</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1403U</td>
<td>Ignore <code>-g</code> option</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1404U</td>
<td>Nothing to operate</td>
</tr>
</tbody>
</table>

There was no library file change nor module extract operation.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1405U</td>
<td>Module not exists to delete (module name)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1406U</td>
<td>Module not exists to extract (module name)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4402U</td>
<td>Duplicated module name (file name module name)</td>
</tr>
</tbody>
</table>

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>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4403U</td>
<td>Duplicated external definition symbol name (file name symbol name)</td>
</tr>
<tr>
<td>E4404U</td>
<td>Invalid module : type (file name)</td>
</tr>
<tr>
<td>E4405U</td>
<td>Invalid module : conflict tool name (file name)</td>
</tr>
<tr>
<td>E4406U</td>
<td>Invalid module : conflict compile model (file name)</td>
</tr>
<tr>
<td>E4407U</td>
<td>Invalid module : conflict CPU type (file name)</td>
</tr>
<tr>
<td>E4470U</td>
<td>CPU information file not found (file 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.

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.

This is not an object file output by the family assembler that can be handled by this librarian.

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

A module of a different CPU type cannot be registered in the same library. Or, the library file which is different from the target CPU is specified.

The target CPU information file specified by the -cpu option cannot find. This is detected when the file below is not found.

- `%FETOOL%\LIB\935\935.CSV`
- `%FETOOL%\LIB\935\cpu_info\*.CSV`
APPENDIX A  Error Messages of the Linkage Kit

The target CPU information, in the CPU information file, specified by the -cpu option cannot find. This is detected when the target CPU information specified in the file below is not found.

- `%FETOOL%\LIB\935\935.CSV`
- `%FETOOL%\LIB\935\cpu_info\*.CSV`

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</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</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>Code</th>
<th>Message</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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9017U</td>
<td>File write error (file name)</td>
</tr>
</tbody>
</table>

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 librarian.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9021U</td>
<td>Too many options</td>
</tr>
</tbody>
</table>

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.
An error is found in the option name specification. Correct the command line and then reactivate.

<table>
<thead>
<tr>
<th>F9022U</th>
<th>Illegal option name (option)</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>F9023U</th>
<th>Illegal option parameter (option)</th>
</tr>
</thead>
</table>

The value cannot be specified in the parameters of the -pl or -pw option. See the -help option or this manual.

<table>
<thead>
<tr>
<th>F9026U</th>
<th>Specified value out of range (value)</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>F9027U</th>
<th>Option file nested</th>
</tr>
</thead>
</table>

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.

<table>
<thead>
<tr>
<th>F9033U</th>
<th>Illegal file format (file name)</th>
</tr>
</thead>
</table>

The library file name is not correctly specified. Specify the library file name correctly.

<table>
<thead>
<tr>
<th>F9035U</th>
<th>Missing library file name</th>
</tr>
</thead>
</table>

Only one library file can be specified. Select either the file name indicated here or a library file name specified before.
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9045U</td>
<td><code>-O</code> option conflict with another option</td>
</tr>
<tr>
<td>F9046U</td>
<td><code>-c</code> option conflict with another option</td>
</tr>
<tr>
<td>F9047U</td>
<td>No match (file name) argument</td>
</tr>
<tr>
<td>F9052U</td>
<td>Missing <code>-cpu</code> option</td>
</tr>
<tr>
<td>F9056U</td>
<td>Mismatch CPU information file version</td>
</tr>
<tr>
<td>F9998U</td>
<td>File open failed (file name)</td>
</tr>
<tr>
<td>F9999U</td>
<td>Internal error (identification information)</td>
</tr>
</tbody>
</table>

If the `-O` option is specified, other options cannot be specified.

Do not combine the `-c` option with any other options.

A file was specified using the wild card, but no corresponding file was found.

No target CPU is specified using the `-cpu` option.
The target CPU must be specified using the `-cpu` option.

The CPU information file is not suited old.
Please obtain the CPU information file of the latest.

The message files used by the librarian file could not be opened.
Store the error message files (lkt_a.msg, lkt_e.msg) in a predetermined directory.

If this error occurs, report it to Fujitsu immediately.
Error Messages of the Object Format Converter

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0501U</td>
<td>Skip start address record</td>
</tr>
<tr>
<td>W1501U</td>
<td>File include WARNING level error (file name)</td>
</tr>
<tr>
<td>W1502U</td>
<td>Unable to convert address (address)</td>
</tr>
<tr>
<td>F1503U</td>
<td>-entry option was specified at the time of -I16 specification</td>
</tr>
<tr>
<td>F1504U</td>
<td>Start address information is not in an input file</td>
</tr>
<tr>
<td>F9001U</td>
<td>Insufficient memory</td>
</tr>
<tr>
<td>F9011U</td>
<td>Input file not found (file name)</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.

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

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.

The start address output specification option is specified at specifying -I16.

The start address information does not exist in the input module file. The f2hs outputs the HEX format without outputting the start address record.

Enough memory is not available to execute the program.

The file specified in the input file cannot be found.
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>F9015U</th>
<th>File open error (file name)</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>F9016U</th>
<th>File read error (file name)</th>
</tr>
</thead>
</table>

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.

<table>
<thead>
<tr>
<th>F9017U</th>
<th>File write error (file name)</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>F9021U</th>
<th>Too many options</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>F9022U</th>
<th>Illegal option name (option)</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>F9023U</th>
<th>Illegal option parameter (option)</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>F9026U</th>
<th>Specified value out of range (value)</th>
</tr>
</thead>
</table>
### APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9027U</td>
<td>Option file nested</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9028U</td>
<td>Specified address too large (option: s=address1 e=address2)</td>
</tr>
</tbody>
</table>

The address specified in parameters of the option cannot be represented in the converted file format. Specify another address.

<table>
<thead>
<tr>
<th>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>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>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>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>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.
APPENDIX A  Error Messages of the Linkage Kit

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9034U</td>
<td>Not absolute load module file (file name)</td>
</tr>
<tr>
<td></td>
<td>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 in the linker and then use it in the converter.</td>
</tr>
<tr>
<td>F9048U</td>
<td>Missing output range</td>
</tr>
<tr>
<td></td>
<td>The output range is not specified. Specify the range of 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 file. Change the output file name.</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 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 opened. Store both lkt_e.msg and lkt_a.msg in the 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>
This appendix explains the following three formats of the HEX formats.

- **HEX8 format:** Format set for the 8-bit
- **HEX16 format:** Format extended for the 16-bit
- **HEX32 format:** Format extended for the 32-bit

B.1 Common Format
B.2 Data Record (HEX8/HEX16/HEX32) Type: 00
B.3 End Record (HEX8/HEX16/HEX32) Type: 01
B.4 Extended Segment Address Record (HEX16/HEX32) Type: 02
B.5 Start Segment Address Record (HEX16/HEX32) Type: 03
B.6 Extended Linear Address Record (HEX32) Type: 04
B.7 Start Linear Address Record (HEX32) Type: 05
B.1 Common Format

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

■ Common format

![Common format diagram]

(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.
I1 indicates the high-order digits and I2 indicates the low-order digits. Values in the range of 0 to 255 can be set.
It is "00" to "FF" in ASCII notation and "0x3030" to "0x4646" in hexadecimal notation.

(c):
Indicates the address allocated to the first data if the contents of (e) are object data.
a1 indicates the high-order digits and a4 indicates the low-order digits. Values in the range of 0 to 65535 can be set.
It is "0000" to "FFFF" in ASCII notation and "0x30303030" to "0x46464646" in hexadecimal notation.

(d):
Indicates the record type.
00: Data record (HEX8/HEX16/HEX32 format)
01: End record (HEX8/HEX16/HEX32 format)
02: Extended segment address record (HEX16/HEX32 format)
03: Start segment address record (HEX16/HEX32 format)
04: Extended linear address record (HEX32 format)
05: Start linear address record (HEX32 format)

(e):
This field depends on the record type of (d). See the explanations of each record in B.2 to B.7.

(f):
Checksum. Each 2-byte data of (b), (c), (d), and (e) represented in ASCII is converted into 1-byte hexadecimal data. Each byte without a sign is added regardless of overflow.
The two's complement of the result is calculated, then set as 2-byte ASCII data. s1 indicates the high-order digits.

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]
: 020000020036C6Extended segment address record
: 0600100090D9226BB4FD43Data record
: 040000035162000541Start address record
: 00000001FFEnd record
B.2 Data Record (HEX8/HEX16/HEX32) Type: 00

d1 and d2 are byte data at the address indicated by (C) and d3 and d4 are byte data of the next address.

Data record (HEX8/HEX16/HEX32)

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 d1 and d2.
Likewise, byte data of the next address is d3 and d4.
B.3  End Record (HEX8/HEX16/HEX32)  Type: 01

The end record is always 0000001FF. Only one end record exists as the last record.

End record (HEX8/HEX16/HEX32)

<table>
<thead>
<tr>
<th>Field</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>F</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>(b)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(c)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(d)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(e)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(g)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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 Segment Address Record (HEX16/HEX32)
Type: 02

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

\[(PA \times 0x10) + (DA + DP) \mod 0x10000\] \mod 0x100000

- **PA**: (e) field value of the record
- **DA**: (c) field value of the data record. Here, this value is a relative address.
- **DP**: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

This is the same method used for calculating the physical address in i8086. With the addition of extended segment address record, up to 20-bit addresses can be represented.

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

---

Figure B.4-1 Extended segment address record (HEX16/HEX32)

<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>2</td>
<td>0</td>
<td>0</td>
<td>p1</td>
<td>p2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>p31</td>
<td>p41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>s11</td>
<td>s21</td>
</tr>
</tbody>
</table>

(e): This is a paragraph address and the actual 2-byte data is represented as 4-byte ASCII code.

In the above figure, p1 indicates the high-order digits.

(c): This field is not used and normally 0 is set.

If extended segment address 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) \mod 0x10000]] \mod 0x100000

- **PA**: (e) field value of the record
- **DA**: (c) field value of the data record. Here, this value is a relative address.
- **DP**: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

This is the same method used for calculating the physical address in i8086. With the addition of extended segment address record, up to 20-bit addresses can be represented.

For the data records which appear before the extended segment address record, the address is calculated with the above PA setting as 0x0000.
B.5 Start Segment Address Record (HEX16/HEX32) Type: 03

This is a record to specify the start address of a program being executed. The start address is calculated using the following formula.

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

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

---

Start segment address record (HEX16/HEX32)

As shown in the above figure, (e) is divided into two fields. The paragraph address is set in (e1), and the offset value is set in (e2).

Both p1 and i1 are high-order digits.

(c) is not used and normally 0 is set.

The start address is calculated using the following formula.

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

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

This record can appear anywhere before the end record.

The appearance count is 0 or 1.
B.6 Extended Linear Address Record (HEX32) Type: 04

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

\[(PA \times 0x10000) + ((DA + DP) \mod 0x10000) \mod 0x100000000\]

- **PA**: (e) field value of the record.
- **DA**: (c) field value of the record. Here, this value is a relative address.
- **DP**: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

Extended linear address record (HEX32)

![Figure B.6-1 Extended linear address record (HEX32)](image)

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

\[(PA \times 0x10000) + ((DA + DP) \mod 0x10000) \mod 0x100000000\]

- **PA**: (e) field value of the record
- **DA**: (c) field value of the record. Here, this value is a relative address.
- **DP**: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

With the addition of extended linear address record, up to 32-bit addresses can be represented.

For the data records which appear before the extended linear address record, the address is calculated with the above PA setting as 0x0000.
### Start Linear Address Record (HEX32) Type: 05

This is a record to specify the start address of a program being executed.

#### Start linear address record (HEX32)

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>e1</td>
</tr>
<tr>
<td></td>
<td>e2</td>
<td>e3</td>
<td>e4</td>
<td>e5</td>
<td>e6</td>
<td>e7</td>
<td>e8</td>
<td>s1</td>
</tr>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td>(e)</td>
<td>(f)</td>
<td>(g)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(e) indicates the 32-bit execution start address is set.

- e1 becomes high-order digits.
- (c) is not used and normally 0 is set.

This record can appear anywhere before the end record.

The appearance count is 0 or 1.
APPENDIX C  S Record Format

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

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)

![Figure C.1-1 S0 type (header record)](image)

This record consists of the above five fields (a) to (e).

The S0 type is called the header type and is placed at the start of a file ahead of each record of S1 to S9. Each field is set in ASCII code.

(a):
Type field. "S0" (0x5330) in ASCII code.

(b):
Indicates the number of bytes in (c), (d), and (e).

Actual 1-byte data is represented as 2-byte ASCII code in this format and the number of characters in these fields divided by 2 is set.

I1 indicates the high-order digits and I2 indicates the low-order digits. Values in the range of 0 to 255 can be set.

It is "00" to "FF" in ASCII notation and "0x3030" to "0x4646" in hexadecimal notation.

(c):
This field is not used and normally "0000" in ASCII notation is set.

(d):
Messages such as version number management information are set.

For the setting method, see the example below.

(e):
Checksum.

Each 2-byte data of (b), (c), and (d) represented in ASCII notation is converted into 1-byte data in hexadecimal notation. Each byte without a sign is added regardless of overflow.

One’s complement of the result is calculated and then set as 2-byte ASCII code.

s1 indicates the high-order digits.

One’s complement: value obtained by reversing each bit

(f):
Generally, control code (such as CR and LF) is added.

Data in this field is skipped until the start character "S" of (a) appears.
[Example]

\[
\begin{array}{c}
\text{S00600004844521B} \\
\text{Indicates ASCII character string "HDR"}
\end{array}
\]
This is a record to store object data that can be represented in two bytes (0x0000 to 0xFFFF).

### S1 Type (Data Record: 2-Byte Address)

The S1 type consists of the above five fields (a) to (e).

(a):  
Type field. "S1" (0x5331) in ASCII code.

(b):  
Indicates the number of bytes in (c), (d), and (e).  
(See the description in Appendix C.1, "S0 Type (Header Record).")

(c):  
Indicates the address allocated to the first data of (d).  
a1 indicates the high-order digits and a4 indicates the low-order digits.  Values in the range of 0 to 65535 can be set.  
It is "0000" to "FFFF" in ASCII notation and "0x30303030" to "0x46464646" in hexadecimal notation.

(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).")

<table>
<thead>
<tr>
<th>S</th>
<th>I1</th>
<th>I2</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d*</th>
<th>d*</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>n bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure C.2-1 S1 type (data record: 2-byte address)](image-url)
C.3  S2 Type (Data Record: 3-Byte Address)

The S2 type differs from the S1 type in (c) field size, and is a record to store object data which requires the 3-byte address.

S2 type (data record: 3-byte address)

The S2 type differs from the S1 type in the above (c) field size, and is a record to store object data which requires the 3-byte address.
C.4  S3 Type (Data Record: 4-Byte Address)

The S3 type differs from the S1 type in (c) field size, and is a record to store object data which requires the 4-byte address.

■ S3 type (data record: 4-byte address)

Figure C.4-1  S3 type (data record: 4-byte address)

| S3 | I1 | I2 | a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | d1 | d2 | d3 | d4 | d* | d* | s1 | s2 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| (a)| (b)| (c)| (d)| (e)| (f)|    |    |    |    |    |    |    |    |    |    |    |    |    |
| 2  | 2  | 4  | n  | 2  |    |    |    |    |    |    |    |    |    |    |    |    |    |

The S3 type differs from the S1 type in the above (c) field size, and is a record to store object data which requires the 4-byte address.
C.5  S5 Type (Record to Manage the Number of Records)

This record sets the number of records contained in a file and may not be specified. It can appear anywhere between S0 and S9.

- S5 type (record to manage the number of records)

Figure C.5-1  S5 type (record to manage the number of records)

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>5</td>
<td>0</td>
<td>n1</td>
<td>n2</td>
<td>s1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(a) (b) (c) (e) (f)
```

2 bytes 2 bytes 4 bytes 2 bytes

The S5 type consists of the above four fields (a) to (e).

(a):
Type field. "S5" (0x5335) in ASCII code.

(b):
Indicates the number of bytes in (c) and (e).
(See the description in Appendix C.1, "S0 Type (Header Record).")

(c):
Indicates the number of data records (S1, S2, S3) in a file.
n1 indicates the high-order digits and n4 indicates the low-order digits. Values in the range of 0 to 65535 can be set.
It is "0000" to "FFFF" in ASCII notation and "0x30303030" to "0x46464646" in hexadecimal notation.

(d):
This field does not exist.

(e):
Checksum.
(See the description in Appendix C.1, "S0 Type (Header Record).")

(f):
Generally, control code (such as CR and LF) is added.
(See the description in Appendix C.1, "S0 Type (Header Record).")
C.6  S7 Type (Terminator Record)

This record indicates the end of a file and contains the start address of execution. This record is placed at the end of a file. This terminator record is used when 4 bytes are required to represent the start address of execution.

### S7 type (terminator record)

![Figure C.6-1 S7 type (terminator record)](image)

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

(b): 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>S8 type (terminator record)</th>
</tr>
</thead>
</table>

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.

Figure C.7-1 S8 type (terminator record)

<table>
<thead>
<tr>
<th>S 1 8</th>
<th>0 1 4</th>
<th>e1 e2</th>
<th>e3 e4</th>
<th>e5 e6</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>
</tr>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>6 bytes</td>
<td>2 bytes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
C.8 S9 Type (Terminator Record)

The S9 type differs from the S7 type in (c) field size, and is a terminator record when 2 bytes are required to represent the start address of execution.

- S9 type (terminator record)

![Figure C.8-1 S9 type (terminator record)](image)

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.
### List of linker options

#### Table D.1-1 List of linker options (1/2)

<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>Specifying padding data</td>
<td>-p</td>
<td>Default 0</td>
</tr>
<tr>
<td>Specification for external symbol output</td>
<td>-symtab</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting the external symbol output</td>
<td>-Xsymtab</td>
<td>Default</td>
</tr>
<tr>
<td>Map list file name 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>Disable Output of Demangled Symbol Name Displayed in List</td>
<td>-Xdemangle</td>
<td></td>
</tr>
<tr>
<td>Enable Output of Demangled Symbol Name Displayed in List</td>
<td>-demangle</td>
<td>Default</td>
</tr>
<tr>
<td>Specification of the number of digits in the list line</td>
<td>-pw</td>
<td>Default 80</td>
</tr>
<tr>
<td>Specification of the number of lines on one list page</td>
<td>-pl</td>
<td>Default 0</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>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>
</tbody>
</table>
## Table D.1-1  List of linker options (2 / 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<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>Target CPU Specification</td>
<td>-cpu</td>
<td>must be specified</td>
</tr>
<tr>
<td>Specifying CPU information file</td>
<td>-cif</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</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>Disable Pre-linking</td>
<td>-XPLNK</td>
<td></td>
</tr>
<tr>
<td>Specification for relative format assemble list input directory</td>
<td>-alin</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute format assemble list output directory</td>
<td>-alout</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute format assemble list output</td>
<td>-als</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute format assemble list output module</td>
<td>-alsf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting absolute format assemble list output</td>
<td>-Xals</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>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 output</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 specification</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>
## List of librarian options

### Table E.1-1  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></td>
</tr>
<tr>
<td>Specification for inhibiting list file output</td>
<td>-Xm</td>
<td>Default</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 80</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>Specifying CPU Information File</td>
<td>-cif</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>
## List of Commands and Options of the Object Format Converter

Attached Table F.1-1 lists the commands of the object format converter and Attached Table F.1-2 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  -&gt;  S format</td>
</tr>
<tr>
<td>f2hs</td>
<td>Absolute format load module  -&gt;  HEX format(HEX8/HEX16/HEX32)</td>
</tr>
<tr>
<td>f2is</td>
<td>Absolute format load module  -&gt;  HEX8 format</td>
</tr>
<tr>
<td>f2es</td>
<td>Absolute format load module  -&gt;  HEX16 format</td>
</tr>
<tr>
<td>m2ms</td>
<td>S format  -&gt;  Adjusted S format</td>
</tr>
<tr>
<td>h2hs</td>
<td>HEX format  -&gt;  Adjusted HEX format</td>
</tr>
<tr>
<td>m2bs</td>
<td>S format  -&gt;  binary data (memory image)</td>
</tr>
<tr>
<td>h2bs</td>
<td>HEX format  -&gt;  binary data (memory image)</td>
</tr>
<tr>
<td>m2is</td>
<td>S format  -&gt;  HEX8 format</td>
</tr>
<tr>
<td>m2es</td>
<td>S format  -&gt;  HEX16 format</td>
</tr>
<tr>
<td>i2ms</td>
<td>HEX8 format  -&gt;  S format</td>
</tr>
<tr>
<td>e2ms</td>
<td>HEX16 format  -&gt;  S format</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>m2ms, h2hs, m2bs, h2bs(need)</td>
</tr>
<tr>
<td>Split mode specification</td>
<td>-sp</td>
<td>m2bs, h2bs only</td>
</tr>
<tr>
<td>Specification for inhibiting split mode</td>
<td>-Xsp</td>
<td>m2bs, h2bs only</td>
</tr>
<tr>
<td>Map list file creation specification</td>
<td>-m</td>
<td>m2bs, h2bs only</td>
</tr>
<tr>
<td>Specification for inhibiting map list file creation</td>
<td>-Xm</td>
<td>m2bs, h2bs only</td>
</tr>
<tr>
<td>Specifying an Output S format</td>
<td>-S1,-S2,-S3</td>
<td>f2ms, m2ms only</td>
</tr>
<tr>
<td>Specifying an Output HEX format</td>
<td>-I16,-I20,-I32</td>
<td>f2hs, h2hs only</td>
</tr>
<tr>
<td>Specifying to output start address record</td>
<td>-entry</td>
<td>f2hs only</td>
</tr>
<tr>
<td>Specifying to not output start address record</td>
<td>-Xentry</td>
<td>f2hs only</td>
</tr>
<tr>
<td>Adjust specification</td>
<td>-adjust</td>
<td>f2ms, f2hs only</td>
</tr>
<tr>
<td>Specifying Changes to the Starting Address</td>
<td>-ST</td>
<td>m2ms, h2hs 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.1-1  Specification differences depending on the OS

<table>
<thead>
<tr>
<th>OS-dependent item</th>
<th>OS type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinction of uppercase/lowercase characters in the file name</td>
<td>No</td>
</tr>
<tr>
<td>Default extension of the file name</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>Current directory</td>
</tr>
<tr>
<td>File specification using the wildcard on the command line</td>
<td>Expanded inside tools</td>
</tr>
</tbody>
</table>

#### Table G.1-2  Differences in wildcard expansion

<table>
<thead>
<tr>
<th>Wildcard pattern</th>
<th>OS type</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Matches the null character or any one character</td>
</tr>
<tr>
<td>*</td>
<td>Matches any character string</td>
</tr>
</tbody>
</table>

#### Table G.1-3  Concrete examples of wildcard expansion

<table>
<thead>
<tr>
<th>Wildcard pattern</th>
<th>OS type</th>
</tr>
</thead>
<tbody>
<tr>
<td>a?.obj</td>
<td>a.obj matches.</td>
</tr>
<tr>
<td></td>
<td>a.obj also matches.</td>
</tr>
<tr>
<td>a*</td>
<td>a.obj, a.obj, and al.abs match.</td>
</tr>
<tr>
<td>*</td>
<td>abc.z and abc.z match.</td>
</tr>
</tbody>
</table>
The index follows on the next page.
This is listed in alphabetic order.
## Index

### A
- Absolute format assemble list .............................................................. 144
- Absolute Format Assemble List File......................................................... 155
- Absolute format assemble list output directory specification (-alout) .......... 130
- Absolute format assemble list output inhibit specification (-Xals) ................ 133
- Absolute format assemble list output module specification (-alsf) ................. 132
- Absolute format assemble list output specification (-als) .................................. 131
- Absolute format load module output specification (-a) ................................... 94
- Adding (registering) a module (-a).......................................................... 184
- Adjust option (-adjust) ............................................................................. 228
- Allocation/link options ............................................................................ 89
- Assemble source list ............................................................................... 157
- Automatic allocation specification (-AL) .................................................. 114
- Automatically Locating Sections .............................................................. 67

### B
- Binary Converter ....................................................................................... 216

### C
- Canceling the omission of names displayed in the list (-dt) ......................... 101
- Cautionary Information Concerning Binary Converter and Format Adjuster ........................................ 268
- Cautionary Information Concerning the Necessary Disk Space ....................... 208
- Cautionary Information Concerning the Specification of Options ..................... 208
- Changing an Output File name (-o) ........................................................... 221
- Changing the format of a list file ................................................................ 55
- Character code of the file name ................................................................... 13
- Checking the contents of a library file ..................................................... 173, 178
- -cmsg option ......................................................................................... 29
- Command line format ............................................................................... 5
- Common format ....................................................................................... 299
- Common options of an object format converter ......................................... 220
- Configuration of a list file ........................................................................ 202
- Configuration of Link List File .................................................................. 145
- Control on combining and locating sections ............................................. 54
- Control on input-output files and messages .............................................. 53
- Control on searching libraries ................................................................... 54
- CPU Information ..................................................................................... 83
- CPU Information File ................................................................................ 83
- CPU Information File Specification (-cif) ................................................. 124
- Creating a group of sections ..................................................................... 57
- Creating a new library file ......................................................................... 173, 174
- -cwno option ......................................................................................... 29
- Data record (HEX8/HEX16/HEX32) ......................................................... 301
- Debug information delete specification (-Xg) ............................................ 93
- Debug information output specification (-g) ............................................ 92
- Default library retrieval inhibit specification (-nd) ..................................... 120
- Default Option File .................................................................................. 46
- Deleting a module (-d) ............................................................................. 186
- Deleting debugging information ................................................................ 173, 177
- Determining location addresses ............................................................... 68, 70
- Disable Output of Demangled Symbol Name Displayed in List (-Xdemangle) .......... 103
- Disable Pre-linking (-XPLNK) ................................................................... 128
- Display of Help Message (-help).............................................................. 33
- Displaying identifier name when outputting list ......................................... 12
- Displaying the contents of a library file .................................................. 173, 178
- Dummy setting of external symbol values (-df) ....................................... 122

### D
- e2ms (Converting a HEX16 Format File into the S Format) ....................... 265
- Editing a library file ................................................................................. 173, 174
- Enable Output of Demangled Symbol Name Displayed in List (-demangle) .......... 104
- End code value and end status ............................................................... 8
- End message ......................................................................................... 10
- End message and -cmsg option ............................................................... 10
- End record (HEX8/HEX16/HEX32) .......................................................... 302
- Entry address specification (-e) ............................................................. 121
- Error Messages for the Librarian ........................................................... 288
- Error messages in the assemble list ...................................................... 155
- Error Messages of the Linker ................................................................ 274
- Error Messages of the Object Format Converter ...................................... 294
- Example Display of the Memory Used Information List Field List ............... 152
- Example of a search when there are multiple library files (1) ...................... 76
- Example of a search when there are multiple library files (2) ...................... 77
INDEX

Example of a search when there is one library file (1) ...................................................... 73
Example of a search when there is one library file (2) .......................................................... 74
Example of a search when there is one library file (3) .......................................................... 75
Example of Describing Option File .................. 45
Example of location when -AL 1 is specified ........ 69
Example of location when -AL 2 is specified .......... 71
Example of location when the order of combining sections is not specified ....................... 64
Example of location when the order of combining sections is specified ......................... 65
Example of location when the section group is specified ................................................. 66
Example of operation ........................................ 241
Example of specifying an option that has a contradictory relation with other options .... 25
Example of specifying an option that has an inclusive relation with other options ........... 25
Example of Specifying Command Lines .......... 26
Executing a command of an object format converter ...................................................... 217
Execution by Specifying Option File ............. 241
Extended linear address record (HEX32) ........ 305
Extended segment address record (HEX16/HEX32) .......................................................... 305
External symbol cross-reference information list ........................................ 144
External Symbol Cross-reference Information List File ............................................... 159
External symbol cross-reference information list file name specification (-xlf) ............. 135
External symbol cross-reference information list output inhibit specification (-Xxl) .... 136
External symbol cross-reference information list output specification (-xl) ................ 134
Extracting a module (-x) ......................... 187
Extracting a module from a library file .......... 173, 176
F
-f option .............................................. 29
f2es (Converting an Absolute Format Load Module into the HEX8 format) .............. 237
f2hs (Converting an Absolute Format Load Module into the HEX format) ................. 235
f2ls (Converting an Absolute Format Load Module into the HEX8 format) ............... 236
f2ms (Converting an Absolute Format Load Module into the S format) .................. 234
FELANG ................................................. 16
FELANG (Message language) ....................... 14
FETOOL .............................................. 14
FETOOL (Installation directory) ................... 14
Forced termination .................................. 7
Format Adjuster ..................................... 216
Function that sets automatically internal ROM/RAM areas (-set_roa) ....................... 126
Functions of the Format Adjuster ................ 241
H
Handling variables with initial values .......... 168
Header format ..................................... 156
Help message ..................................... 11
-help option ...................................... 29
i2ms (Converting a HEX8 Format File into the S Format) ........................................ 264
Indicating identifiers ................................ 12
Inheriting debugging information .................. 55
Inhibiting Check for Presence of Debug Data (-NCI0302LIB) ......................... 125
Inhibiting the search for a library ............ 58
Interleaved Objects on the Linker ................. 84
L
LIB935 (Library file search directory) ........ 14, 18
Library control option ................................ 89
Library File Made by the SOFTUNE V5 Tool .... 180
Library retrieval inhibit specification (-nl) .... 119
Library retrieval path specification (-L) ... 117
Library specification for each symbol (-el) .... 118
Limiting the number of letters for identifiers .... 12
Link list file ........................................ 144
Linkage Kit Error Message Classes .............. 272
Linkage Kit Error Message Display Format .... 273
Linker reservation symbol ......................... 166
Linker restriction .................................. 166
Links of sections .................................. 63
List display example of the control list part .... 147
List display example of the map list part ........ 149
List display example of the symbol list part .... 154
List of commands of the object format converter .... 320
List of common options ......................... 28
List of librarian options ........................ 319
List of linker options .......................... 86, 317
List of Options of a Librarian .................. 182
List of Options of the Binary Converter ... 254
List of Options of the Format Adjuster .......... 243
List of Options of the Load Module Converter .... 227
INDEX

S8 type (terminator record) ........................................ 315
S9 type (terminator record) ........................................ 316
Section Allocation Detailed Information List File .. 163
Section allocation order/address specification (-sc) .................................................. 110
Section combination attribute ................................ 60
Section detail map list ................................................. 144
Section detail map list file name specification (-mlf) .................................................. 141
Section detail map list output inhibit specification (-Xml) ........................................ 142
Section detail map list output specification (-ml) .. 140
Section group specification (-gr) ................................ 113
Section identification ................................................. 61
Section location attribute ......................................... 60
Section name .......................................................... 60
Section types and location destinations .................... 70
Sections to be transferred from ROM to RAM .......... 80
Selecting the warning check level ................................ 55
Selecting whether or not to display a startup message ........................................... 55
Selecting whether or not to display a termination message ...................................... 56
Setting an external symbol value ............................... 59
Setting entry addresses and symbol values ............ 54
Shared combination of sections ................................. 62
Simple connection combination of sections .............. 62
Specification differences depending on the OS ... 322
Specification for external symbol information output (-symtab) .................................. 97
Specification for inhibiting the external symbol information output (-Xsymtab) ........ 98
Specification of the number of digits in the list line (-pw)............................................. 105
Specification of the number of lines on one list page (-pl) .......................................... 106
Specification related to output messages .................. 89
Specification to Continue in the Option File ............ 43
Specifies to prevent the internal ROM/RAM area from being set automatically .......... 83
Specifies to prevent the internal ROM/RAM area from being set automatically (-Xset_rora) ................................................................. 127
Specifying a library file to be searched ................. 58
Specifying a library file to be searched for each symbol 58
Specifying a library to be searched ......................... 72
Specifying a list file name ........................................ 55
Specifying a path to search a library ...................... 58
Specifying an entry address ..................................... 59
Specifying an Output HEX format (-I16/-I20/-I32) 244
Specifying an output HEX format (-I16/-I20/-I32) .249
Specifying an output load module file name .......... 55
Specifying an output S format (-S1/-S2/-S3) 244, 247
Specifying Changes to the Starting Address (-ST) .................................................... 250
Specifying comment in the Option File .................... 44
Specifying Display of End Message (-cmsg) ............ 36
Specifying End Code to 0 When Warning is Issued (-Xowno) ..................................... 39
Specifying End Code to 1 When Warning is Issued (-owno) ..................................... 38
Specifying input object files ..................................... 55
Specifying not to create a map list file (-Xm) 255, 260
Specifying not to Output Start Address Record (-Xentry) .................................... 232
Specifying Reading from Option Files (-f) ............... 31
Specifying ROM and RAM areas ................................. 57
Specifying Suppression of Default Option File (-Xdof) .................................................. 30
Specifying Suppression to Display End Message (-Xcmsg) .................................... 37
Specifying the data length in an output record (-len) .................................................................. 245
Specifying the inhibition of the split mode (-Xsp) ................................................................ 255, 258
Specifying the order of locating sections and the location addresses ......................... 57
Specifying the output format ................................... 55
Specifying the output range (-ran) 244, 246, 255, 256
Specifying the output record data length (-len) .... 244
Specifying the split mode (-sp) ................................. 255, 257
Specifying to Adjust (-adjust) ................................. 233
Specifying to change the starting address of record (-ST) ............................................ 244
Specifying to create a map list file (-m) 255, 259
Specifying to output a list file (-m) ......................... 188
Specifying to Output Start Address Record (-entry) ................................................................ 231
Specifying Version Number Message Output (-V) .. 34
Start address output inhibit option (-Xentry) ............ 228
Start address output option (-entry) ......................... 228
Start linear address record (HEX32) ......................... 306
Start segment address record (HEX16/HEX32) ....... 304
Startup message ....................................................... 9
Startup message and the -V option ......................... 9
Support for creating a ROM ................................. 57
Support Range of Linkage Kit ................................. 4
Suppression of Version Number Message Output (-XV) ................................................ 35
Synopsis .................................................................. 22
INDEX

Synopsis of option .................................................. 22

T
Target CPU specification (-cpu) ......................... 123
TMP (Work directory) ........................................ 14
TMP (work directory) ...................................... 15
Types of characters consisting of identifiers .... 12
Types of load module converters ................. 216
Types of other converters ......................... 216
Types of section contents ............................. 60

U
Using the sections to be transferred from ROM to
RAM ............................................................. 80

Using the wild card ........................................... 167

V
-V option ......................................................... 29

W
Warning message output level specification (-w). 107

X
-Xcmsg option .................................................. 29
-Xcwno option .................................................. 29
-Xdof option .................................................... 29
-XV option .................................................... 29