F\textsuperscript{2}MC\textsuperscript{TM}-16 FAMILY

SOFTUNE\textsuperscript{TM} LINKAGE KIT MANUAL

for V3
PREFACE

■ Objectives and Intended Readership

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

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

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

■ Trademarks

F2MC is a trademark of FUJITSU LIMITED.

SOFTUNE is a trademark of FUJITSU LIMITED.

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

The names of products and systems appearing in this manual are trademarks or registered trademarks of their respective companies.

■ Configuration of this manual

This manual consists of the following five parts:

PART 1 LINKAGE KIT

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

PART 2 LINKER

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

PART 3 LIBRARIAN

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

PART 4 OBJECT FORMAT CONVERTERS

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

Appendices

Explains the error messages of the linkage kit, HEX format and S 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

■ Product name abbreviation

In this manual, product names are abbreviated as follows:

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

### PART 1  LINKAGE KIT .................................................................1

### CHAPTER 1  SPECIFICATIONS OF LINKAGE KIT ........................................3

1.1 Outline of Linkage Kit ............................................................................4
1.2 Startup Procedure ..................................................................................5
1.3 Forced Termination ...............................................................................6
1.4 End Code ..............................................................................................7
1.5 Startup Message ..................................................................................8
1.6 End Message ......................................................................................9
1.7 Help Message ..................................................................................10
1.8 Identifiers ......................................................................................11
1.9 Filename Rules ...............................................................................12
1.10 Environment Variables ..................................................................13

1.10.1 TMP (Work directory) .................................................................14
1.10.2 FELANG ....................................................................................15
1.10.3 FETOOL ....................................................................................16
1.10.4 LIB907 .....................................................................................17
1.10.5 OPT907 ...................................................................................18
1.10.6 OPT .........................................................................................19

### CHAPTER 2  OPTIONS .................................................................21

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

### CHAPTER 3  COMMON OPTIONS ................................................27

3.1 List of Common Options ..................................................................28
3.2 Details of Common Options ............................................................29

3.2.1 Specifying Suppression of Default Option File (-Xdof) ..............30
3.2.2 Specifying Reading from Option Files (-f) ..................................31
3.2.3 Specifying Help Message (-help) ...............................................33
3.2.4 Specifying Version Number/Message Output (-V) .....................34
3.2.5 Suppression of Version Number/Message Output (-XV) ..........35
3.2.6 Specifying Display of End Message (-cmsg) ............................36
3.2.7 Specifying Suppression to Display End Message (-Xcmsg) ......37
3.2.8 Specifying End Code to 1 When Warning is Issued (-cwno) ......38
3.2.9 Specifying End Code to 0 When Warning is Issued (-Xcwno) ....39

### CHAPTER 4  OPTION FILES ..............................................................41

4.1 Option File ..................................................................................42
4.2 Specification to Continue in the Option File .......................................43
4.3 Specifying Comment in the Option File ...........................................44
PART 2  LINKER ................................................................................................................. 49

CHAPTER 5  SPECIFICATIONS OF A LINKER ................................................................ 51
5.1  Overview of a Linker ........................................................................................................... 52
5.2  Functions of a Linker ........................................................................................................... 53
  5.2.1  Control on Input-Output Files and Messages ................................................................. 55
  5.2.2  Control on Combining and Locating Sections ................................................................. 57
  5.2.3  Control on Searching Libraries ...................................................................................... 58
  5.2.4  Setting Entry Addresses and Symbol Values ................................................................. 59
5.3  Types of Sections ............................................................................................................... 60
5.4  Combining Sections ........................................................................................................... 62
5.5  Locating Sections ............................................................................................................... 63
  5.5.1  Example of Location when the Order of Combining Sections is not Specified ............... 64
  5.5.2  Example of Location when the Order of Combining Sections is Specified ....................... 65
  5.5.3  Example of Location when the Section Group is Specified ............................................ 66
5.6  Automatically Locating Sections ....................................................................................... 67
  5.6.1  Automatically Locating Sections when -AL 1 is Specified .............................................. 68
  5.6.2  Automatically Locating Sections when -AL 2 is Specified .............................................. 70
5.7  Searching Libraries ........................................................................................................... 72
  5.7.1  Example of a Search when there is one Library File (1) ................................................... 73
  5.7.2  Example of a Search when there is one Library File (2) ................................................... 74
  5.7.3  Example of a Search when there is one Library File (3) ................................................... 75
  5.7.4  Example of a Search when there are Multiple Library Files (1) ...................................... 76
  5.7.5  Example of a Search when there are Multiple Library Files (2) ...................................... 77
  5.7.6  Processing when Library Files are Individually Specified .............................................. 78
5.8  ROM and RAM Areas ....................................................................................................... 79
5.9  Sections to be Transferred from ROM to RAM ................................................................. 80
5.10 Location of DIR Attribute Section ..................................................................................... 83
5.11 CPU Information File ...................................................................................................... 86
5.12 Mixing of Objects for a Linker ......................................................................................... 87
5.13 Mixing of Objects with Different Function-call Interfaces in Linker ................................... 88

CHAPTER 6  LINKER OPTIONS ......................................................................................... 89
6.1  List of Linker Options ....................................................................................................... 90
6.2  Details of Linker Options .................................................................................................. 93
  6.2.1  Output Load Module File Name Specification (-o) ............................................................ 94
  6.2.2  Debug Information Output Specification (-g) ................................................................. 95
  6.2.3  Debug Information Delete Specification (-Xg) .............................................................. 96
  6.2.4  Absolute Format Load Module Output Specification (-a) .............................................. 97
  6.2.5  Relative Format Load Module Output Specification (-r) ............................................... 98
  6.2.6  Map List File Name Specification (-m) .......................................................................... 99
  6.2.7  Map List Output Inhibit Specification (-Xm) ................................................................. 100
  6.2.8  Canceling the Omission of Names Displayed in the List (-dt) ......................................... 101
  6.2.9  Output Specification of the Map List File Memory Used
6.2.10 Specification of the Number of Digits in the Line (-pw) ............................................................... 103
6.2.11 Specification of the Number of Lines on one List Page (-pl) ........................................................... 104
6.2.12 Warning Message Output Level Specification (-w) ...................................................................... 105
6.2.13 ROM Area Specification (-ro) ....................................................................................................... 106
6.2.14 RAM Area Specification (-ra) ....................................................................................................... 107
6.2.15 Section Allocation Order/Address Specification (-sc) ...................................................................... 108
6.2.16 Section Group Specification (-gr) .................................................................................................... 111
6.2.17 Register Bank Area Specification (-rg) .......................................................................................... 112
6.2.18 Automatic Allocation Specification (-AL) ........................................................................................ 114
6.2.19 Retrieval Library File Specification (-l) .......................................................................................... 116
6.2.20 Library Retrieval Path Specification (-L) ..................................................................................... 117
6.2.21 Library Specification for Each Symbol (-el) .................................................................................. 118
6.2.22 Library Retrieval Inhibit Specification (-nl) .................................................................................. 119
6.2.23 Default Library Retrieval Inhibit Specification (-nd) ..................................................................... 120
6.2.24 Entry Address Specification (-e) .................................................................................................... 121
6.2.25 Dummy Setting of External Symbol Values (-df) ........................................................................ 122
6.2.26 Target CPU Specification (-cpu) ................................................................................................... 123
6.2.27 Specifying CPU Information File (-cif) .......................................................................................... 124
6.2.28 Inhibiting Check for Presence of Debug Data (-NCI0302LIB) .......................................................... 125
6.2.29 Function that sets automatically internal ROM/RAM area (-set_rora) ......................................... 126
6.2.30 Specifies to prevent the internal ROM/RAM area from being set automatically (-Xset_rora) ....... 127
6.2.31 User-specified-area check specification (-check_rora) .................................................................. 128
6.2.32 User-specified-area check suppression specification (-Xcheck_rora) ........................................... 130
6.2.33 Section-placed-area check specification (-check_locate) ............................................................. 131
6.2.34 Section-placed-area check suppression specification (-Xcheck_locate) ....................................... 134
6.2.35 Check User-unspecified Section (-check_section) ......................................................................... 135
6.2.36 Inhibit check for presence of user-unspecified section (-Xcheck_section) .................................. 137
6.2.37 Specification of Standby Mode Transition Instruction (-check_SCF) .............................................. 138
6.2.38 Specification of Prohibiting Standby Mode Transition Instruction Check (-Xcheck_SCF) .......... 140
6.2.39 Specification of Fetch Unit Checks (-Wf) ....................................................................................... 142
6.2.40 Specified Output Load Module for Register Argument Passing (-rp) ........................................... 144
6.2.41 Specified Output Load Module for Stack Argument Passing (-Xrp) ................................................ 145
6.2.42 Specifying that Check on the Total Size of the DIR-attribute Sections (-check_dirsize) .......... 146
6.2.43 Specifying that Check on the Total Size of the DIR-attribute Sections be Inhibited (-Xcheck_dirsize) .......................................................... 147
6.2.44 Relative Assemble List Input Directory Specification (-alin) .......................................................... 148
6.2.45 Absolute Assemble List Output Directory Specification (-alout) .................................................. 149
6.2.46 Absolute Assemble List Output Specification (-als) ....................................................................... 150
6.2.47 Absolute Assemble List Output Module Specification (-alsf) ....................................................... 151
6.2.48 Absolute Assemble List Output Inhibit Specification (-Xals) .......................................................... 152
6.2.49 ROM/RAM and ARRAY List Output Specification (-alr) ................................................................. 153
CHAPTER 7  OUTPUT LIST FILE OF THE LINKER ................................................................. 167
  7.1 Types of List Files Output by the Linker ........................................................................ 168
  7.2 Link List File .................................................................................................................. 169
    7.2.1 Control List ............................................................................................................. 170
    7.2.2 Map List ............................................................................................................... 172
    7.2.3 Memory Used Information List .............................................................................. 174
    7.2.4 Symbol List .......................................................................................................... 177
  7.3 Absolute Assemble List File .......................................................................................... 179
    7.3.1 Header and Information List .................................................................................... 181
    7.3.2 ROM/RAM and ARRAY Lists ................................................................................ 182
    7.3.3 Assemble Source List ............................................................................................ 184
    7.3.4 Section Information List ....................................................................................... 186
    7.3.5 Cross-reference List ............................................................................................. 188
  7.4 External Symbol Cross-reference Information List File ............................................... 190
  7.5 Local Symbol Information List File .............................................................................. 191
  7.6 Section Allocation Detailed Information List File ....................................................... 193

CHAPTER 8  LINKER RESTRICTIONS AND Q&A ............................................................ 195
  8.1 Linker Restrictions ....................................................................................................... 196
  8.2 Q&A for Using the Linker ............................................................................................ 197

PART 3  LIBRARIAN .......................................................................................................... 201

CHAPTER 9  SPECIFICATIONS OF A LIBRARIAN ............................................................ 203
  9.1 Functions of a Librarian ............................................................................................... 204
  9.2 Function Types of a Librarian ..................................................................................... 205
  9.3 Creating and Editing a Library File .............................................................................. 206
  9.4 Extracting a Module from a Library File ...................................................................... 208
  9.5 Deleting Debugging Information of a Library ............................................................. 209
  9.6 Checking and Displaying the Contents of a Library File ............................................. 210
  9.7 Mixing of Objects for a Librarian ................................................................................ 211
CHAPTER 10 OPTIONS A LIBRARIAN .................................................................213

10.1 List of Options of a Librarian ...........................................................................214
10.2 Details of the Options of a Librarian .................................................................216
10.2.1 Adding (Registering) a Module (-a) ..............................................................217
10.2.2 Replacing (Registering) a Module (-r) ...........................................................218
10.2.3 Deleting a Module (-d) .................................................................................219
10.2.4 Extracting a Module (-x) ..............................................................................220
10.2.5 Specifying to Output a List File (-m) ..............................................................221
10.2.6 Specifying not to Output a List File (-Xm) .....................................................222
10.2.7 Specifying to Output Detailed Information of a List File (-dt).......................223
10.2.8 Specifying the Number of Lines Per Page of a List (-pl)...............................224
10.2.9 Specifying the Number of Columns Per Line of a List (-pw).........................225
10.2.10 Creating a Backup File (-b) .........................................................................226
10.2.11 Inhibiting the Creation of a Backup File (-Xb) .............................................227
10.2.12 Checking the Contents of a Library File (-c) ................................................228
10.2.13 Optimizing the Contents of a File (-O) .........................................................229
10.2.14 Specifying to Output Debugging Information (-g) .......................................230
10.2.15 Specifying not to Output Debugging Information (-Xg) ..............................231
10.2.16 Specifying a Target CPU (-cpu) .................................................................232
10.2.17 Specifying CPU Information File (-cif) ......................................................233
10.2.18 Specified Output Library for Register Argument Passing (-rp) ....................234
10.2.19 Specified Output Library for Stack Argument Passing (-Xrp) .....................235

CHAPTER 11 LIST FORMATS OF A LIBRARIAN ................................................237

11.1 Contents of Information in a List File ............................................................238
11.2 List of Module Names .......................................................................................239
11.3 Detailed Information of a Module ....................................................................240
11.4 External Defined and Reference Symbol Information in a Library ....................241

CHAPTER 12 RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN243

12.1 Restrictions on a Librarian ..............................................................................244
12.2 Questions and Answers on Using a Librarian ..................................................245

PART 4 OBJECT FORMAT CONVERTERS .........................................................247

CHAPTER 13 SPECIFICATIONS OF AN OBJECT FORMAT CONVERTER .............249

13.1 Outline of Object Format Converter ...............................................................250
13.2 Types of Converters .......................................................................................252
13.3 Executing an Object Format Converter ..........................................................253

CHAPTER 14 COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER ........255

14.1 List of Common Options of an Object Format Converter ..................................256
14.2 Changing an Output File name (-o) ...............................................................257
14.3 Padding (-p) ..............................................................................................259
Appendixes ...........................................................................................................................................309

APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT ..........................................................310

APPENDIX B  HEX FORMAT ........................................................................................................341
  B.1  Common Format..................................................................................................................342
  B.2  Data Record (HEX8/HEX16/HEX32) Type: 00.............................................................344
  B.3  End Record (HEX8/HEX16/HEX32) Type: 01...............................................................345
  B.4  Extended Segment Address Record (HEX16/HEX32) Type: 02..............................346
  B.5  Start Segment Address Record (HEX16/HEX32) Type: 03........................................347
  B.6  Extended Linear Address Record (HEX32) Type: 04................................................348
  B.7  Start Linear Address Record (HEX32) Type: 05........................................................349

APPENDIX C  S RECORD FORMAT .......................................................................................351
  C.1  S0 Type (Header Record)...................................................................................................352
  C.2  S1 Type (Data Record: 2-Byte Address)..........................................................................353
  C.3  S2 Type (Data Record: 3-Byte Address)..........................................................................354
  C.4  S3 Type (Data Record: 4-Byte Address)..........................................................................355
  C.5  S5 Type (Record to Manage the Number of Records)....................................................356
  C.6  S7 Type (Terminator Record)..........................................................................................357
  C.7  S8 Type (Terminator Record)..........................................................................................358
  C.8  S9 Type (Terminator Record)..........................................................................................359

APPENDIX D  LIST OF LINKER OPTIONS ............................................................................360

APPENDIX E  LIST OF LIBRALIAN OPTIONS ...........................................................................363

APPENDIX F  LIST OF COMMANDS AND OPTIONS OF THE OBJECT FORMAT
CONVERTER ........................................................................................................................364

APPENDIX G  SPECIFICATION DIFFERENCES DEPENDING ON THE OS .......................366

APPENDIX H  F²MC-16LX FAMILY STANDBY MODE TRANSITION INSTRUCTION
CHECK TOOL ..........................................................................................................................367
  H.1  General Description of F²MC-16LX Standby Mode Transition Instruction Check Tool........368
  H.2  How to Operate the Check Tool .......................................................................................369
  H.3  Checking Procedures ........................................................................................................375
  H.4  Warning Messages.............................................................................................................376
  H.5  Check method......................................................................................................................377
  H.6  Detectable Standby Mode Transition Instructions..........................................................380
  H.7  Restrictions........................................................................................................................382
  H.8  Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition....................................................................................................................383

APPENDIX I  DIFFERENCE IN SPECIFICATION OF SOFTUNE LINKER (FLINK907S)
AND OLD LINKER (LINK907A) ..................................................................................................387
PART 1  LINKAGE KIT

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

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

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

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

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

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

- **Command line format**

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

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

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

  - **Linker**

    ```
    flnk907s [ Option ] ... < Filename >
    ```

    Specify the object module filename to be input to `<filename>`.
    Insert a space to specify two or more filenames.
    A wild card such as `*.obj` can also be used. Expanding the wild card of filename depends on the OS. Refer to “APPENDIX G SPECIFICATION DIFFERENCES DEPENDING ON THE OS”.
    In linker, the target CPU must be specified using the `-cpu` option. Be sure to specify the `-cpu` option when executing the link processing.

  - **Librarian**

    ```
    flib907s [ Option ] ... < Filename >
    ```

    Specify the library file that is the target of editing, to `<filename>`.
    In librarian, the target CPU must be specified using the `-cpu` option. Be sure to specify `-cpu` option when executing the library processing.

  - **Object format converter**

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

    Determine the object type filename to `<filename>` based on the functions of each tool.
    Files of the following four types are the target format.
    - Absolute type load module of linker output
    - 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 CTR-C”.) Pressing CTRL-C will suspend a program.

- Forced termination

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

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

End code value and end status

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

Table 1.4   End codes and end status of processing

<table>
<thead>
<tr>
<th>End code</th>
<th>End status of processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>When ended normally or when an error of warning level occurs.</td>
</tr>
<tr>
<td>1</td>
<td>When an error of warning level occurs with the -cwno option specified</td>
</tr>
<tr>
<td>2</td>
<td>When an error occurs making it impossible to achieve the correct output result</td>
</tr>
<tr>
<td>3</td>
<td>When a fatal error occurs making it impossible to continue processing</td>
</tr>
</tbody>
</table>
1.5 Startup Message

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

- Startup message and the -V option

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

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

- Startup message

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

The startup message is shown below.

```
F2MC-16 Family SOFTUNE Linker VxxLxx
ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 1992
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).

```
usage: flnk907s [-option ...] object[ object ...]  *1
----------
options:

------- target CPU option -------
-cpu cpu-name : Specify target CPU (need)

------- linker mode options -------
-a : absolute linking mode (absolute)  *2
-r : relocatable linking mode

------- library options -------
-l filename[,...] : specify library file name

[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”.)

Figure 1.7 Example of help message
1.8 Identifiers

Linkage kit can handle the following seven kinds of identifiers.
• Filename
• Module name
• Option name
• Section name
• Group name
• ROM/RAM area name
• Symbol name

■ Types of characters consisting of identifiers

The following characters can be used as identifiers.
  • Alphabetical letters
  • Numbers
  • Underscore (_)

Numbers cannot be used at the top of letters.
At the same time, types of characters that can be used for the filename depends on the OS being used. The module name that is created from the filename also depends on the OS being used.

■ Indicating identifiers

English uppercase and lowercase are indicated.

■ Limiting the number of letters for identifiers

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

■ Displaying identifier name when outputting list

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

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

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

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

Filename of the input/output files complies with the limited use of characters that are set for the OS. There are cases in which the number of characters and code system must be taken into account because the filename is also set in the object modules.

- Number of characters for the filename

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

- Character code of the filename

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

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

- Filename and characters that can be used for the filename

(Windows Me/Windows 2000/Windows XP Version)

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

\ / : ; , * ? " < > |

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

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

Linkage kit support the following six kinds of environment variable.
- TMP
- FELANG
- FETOOL
- LIB907
- OPT907
- OPT

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

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

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

- **LIB907 (Library search directory)**
  LIB907 specify the directory in which library is stored. Refer to “1.10.4 LIB907” for more details.

- **OPT907 (Default option file storage directory)**
  OPT907 specify directory in which default option files of linker and librarian are stored. Refer to “1.10.5 OPT907” for more details.

- **OPT (Default option file storage directory)**
  OPT specifies a directory in which the default option file of the object tool is stored. Refer to “1.10.6 OPT” for more details.
1.10.1 TMP (Work directory)

TMP (work directory) specifies the work directory that the linkage kit uses during execution.  
The section below gives the description format, an explanation and an example of specification.

TMP (work directory)

[Description format]

```
SET TMP = < Path name >
```

[Explanation]

- It specifies the work directory that the linkage kit uses during execution.
- This environmental variable TMP can also be used in other development tools. (Such as C compiler and assembler)
- When the environmental variable TMP is not specified, the current directory is used.

[Example]

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

FELANG selects and specifies the message language of help message and error message.
The following section gives the description format, an explanation and an example of FELANG.

FELANG

[Description format]

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

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

[Explanation]

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

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

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

[Example]

```
SET FELANG=ASCII
```
1.10.3 FETOOL

FETOOL specifies the root directory in which linkage kit is installed. The following section gives the description format, an explanation and an example of FETOOL.

■ FETOOL

[Description format]

SET FETOOL = < Path name >

Specify the <path name> including drive name.

[Explanation]

Specify the directory in which linkage kit is installed.

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

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

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

[Example]

SET FETOOL=C:\Softune

[Recommended directory structure]

\Softune
  \BIN  Stores load module of linkage kit
  \LIB  Stores message files that does not depend on CPU
        \907  Stores library file for F2MC-16 family and message file

Figure 1.10  Recommended directory structure

[Supplement]

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

The environment variable FETOOL allows linkage kit to notify the directory path of “SOFTUNE”. 
LIB907 (library file search directory) specify the directory in which the library files that linker searches are located. The section below gives the description format, an explanation and an example of LIB907.

### LIB907

**[Description format]**

```plaintext
SET LIB907 = <Path name> [ ; <Path name> ... ]
```

Specify the `<path name>` including the drive name.

**[Explanation]**

It specifies the directory in which the library files that linker searches are located.
Specify the directory in which the C library is stored normally.

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

- Semicolon (`;`)

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

**[Example]**

```
SET LIB907=C:\Softune\LIB\907
```

**[Supplement]**

When the Environment variable FETOOL is specified, the library storage directory of the directory structure, as described in the previous item, is also searched. So the C library is searched even though the environment variable LIB907 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 LIB907.
3) The directory (`%FETOOL%\LIB\907`) that is directed by the environment variable FETOOL.

If the user created the library himself, specify the paths while taking note of the order of searching with the C library.
OPT907 (default option file storage directory) specifies the directory in which the default option files of linker and librarian are stored. The description format, an explanation and an example of OPT907 is given below.

**OPT907**

[Description format]

```
SET OPT907 = < Path name >
```

Specify the <path name> including the drive name.

[Explanation]

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

This environment variable can be omitted.

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

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

- **Linker**
  - `%FETOOL%LIB\907\FLNK907.OPT`
- **Librarian**
  - `%FETOOL%LIB\907\FLIB907.OPT`

[Example]

```
SET OPT907=C:\Softune\LIB\907
```
1.10.6 OPT

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

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

- OPT

  [Description format]

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

  Specify the <path name> including the drive name.

  [Explanation]

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

  This environment variable can be omitted.

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

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

  - Object tools

    - `%FETOOL%\LIB\F2M.OPT`
    - `%FETOOL%\LIB\F2H.OPT`
    - `%FETOOL%\LIB\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:\Softune\LIB
  ```
CHAPTER 2 OPTIONS

This section describes options of the linkage kit.

2.1 Option
2.2 Numeric Expression of Option Parameters
2.3 Notes and Evaluation When Option Is Specified
2.4 Specifying Options that Have Inclusive or Contradictory Relation Each Other
2.5 Example of Specifying Command Lines
Chapter 2  Options

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.

- Optionname [Parameter] ...

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

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

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

Pay attention to the following points when specifying an option.

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

- Parameter

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

[Example]

- a
  
  > gets.obj,puts.obj,getc.obj,putc.obj
  
  -sc CODE=0xC1000,DATA=0x1000
2.2 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 large and small letters can be used for a to f of the hexadecimal notation.

  [Example]
  
  0x100... Hexadecimal notation (= 256)
  100... Decimal notation (= 0x64)
  0xff and 0xFF are the same.
2.3 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=0x100, 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]

```
flnk907s  
flnk907s 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]

```
flib907s sys.lib  -m sys.mp2 ... *1
flib907s -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]

```
flnk907s *.obj -g -o sample.abs
```

Wild card is used to specify two or more input files in this example.
CHAPTER 3 COMMON OPTIONS

Linkage kit has common options that can be used in any tools. These options are also prepared in C compiler and assembler. The options that are unique in this tool are also described in the respective paragraphs.

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

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

- List of common options

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

<table>
<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.

[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]

flnk907s test.obj -Xdof
3.2.2 Specifying Reading from Option Files (-f)

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

- Specifying Reading from Option Files (-f)

[Synopsis]
- f < Option file name >

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

[Description]
Describe options and input files into the file that is specified by < Option file name >. This option issues direction to read contents of the option from the file in which options is described.

Contents of the file specified in the command line and this file are evaluated and processed equally.

Extension of the file name is not determined in default.

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

[Example 1]
f2ms -V -f optfile.f2m

Contents of optfile.f2m

```
#  from  FJ-OMF  to  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]
flib907s 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.
CHAPTER 3  COMMON OPTIONS

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

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

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

```
flib907s -f libfile.opt
```

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

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

Option file name can be specified twice.

```
flib907s 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, strerror.obj, strpbrk.obj,
   strchr.obj, strcmp.obj, strcpy.obj, strlen.obj
```
3.2.3 Specifying Help Message (-help)

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

- Specifying Help Message (-help)

[Synopsis]
- help

[Parameter]
None

[Description]
-help option briefly displays the format to specify the command line and list of options.
Help message is output to the standard output (stdout).
When the command name only is specified, the same help message is output.
When input file name and other options are specified, the help message only is displayed without executing programs when this option is specified.
3.2.4 Specifying Version Number/Message Output (-V)

-V option outputs the message during program startup.

- 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]

flnk907s ccp903

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

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

[Example 2]

f2ms ccp903 -V

SOFTUNE FJ-OMF to S-FORMAT Converter  VxxLxx
ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 1992
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]

flib907s -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]

flnk907s ccp903
flnk907s 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 -flkit.opt ccp903 -XV

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

When lkit.opt has -V option in it, contents of lkit.opt remain unchanged. However, -XV can be specified in order to cancel the -V option.
3.2.6 Specifying Display of End Message (-cmsg)

It displays the end message of the program.

- Specifying Display of End Message (-cmsg)

  [Synopsis]

  - cmsg

  [Parameter]

  Note

  [Description]

  It displays the end message of a program.

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

  [Example 1]

  flnk907s ccp903

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

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

  [Example 2]

  f2ms ccp903 -cmsg
  F2MS COMPLITED FOUND NO ERROR

  At the end of the program, the end message (program name and presence or absence of errors) is displayed.
3.2.7 Specifying Suppression to Display End Message (-Xcmsg)

It suppresses display of end message.

- Specifying Suppression to Display End Message (-Xcmsg)

  [Synopsis]

  - Xcmsg

  [Parameter]

  None

  [Description]

  It suppresses display of end message
  
  The linkage kit does not display the end message at the end of the program in the default setting.
  
  Use this option to cancel the display option (-cmsg) to display the end message of the program.

  [Example 1]

  flnk907s ccp903
  flnk907s 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]**

fink907s ccp903 -cwno

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

**[Example 2]**

fink907s ccp903

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

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

- Specifying End Code to 0 When Warning is Issued (-Xcwno)
  [Synopsis]
  
  - Xcwno

  [Parameter]
  None

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

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

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

  [Example 1]
  
  flnk907s ccp903
  flnk907s ccp903 -Xcwno

  When warning only is issued during execution of program, the end code is 0 in the default setting.

  The two settings as specified above are the same.

  [Example 2]
  
  f2ms -f lkit.opt ccp903 -Xcwno

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

  When the -cwno option is used in lkit.opt, the -cwno option can be canceled by specifying -Xcwno on the command line without changing contents of the lkit.opt.
CHAPTER 4  OPTION FILES

This section describes option files.

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

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

**Option File**

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

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

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

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

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

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

**Execution by Specifying Option File**

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

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

[Example]

```
flib907s -f optfile
```

Content of option file “optfile”

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

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

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

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

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

As described above, all specifications that can be described in command line (excluding -f option and -Xdof option) can be described using the same format.
4.2 Specification to Continue in the Option File

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

- Specification to Continue in the Option File

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

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

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

     -a mod01, mod02, obj03, obj04

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

     -a mod01, mod02, obj03, obj04
          the continued line

  :
4.3 Specifying Comment in the Option File

Comment can be input into option file.

- Specifying comment in the Option File

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

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

The underlined portion is comment.

```
# Example of Library Options
syslib.lib    # INDICATES LIBRARY FILE
-a mod01, mod02, obj03, obj04    # Add Modules
```
The examples below show how to specify option file that can be handled equally as `-a mod01, mod02, obj03, obj04` in command line.

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

- `-a mod01, mod02 ,obj03 , obj04`
  - Inserting a space before and after comma

- `-a mod01,mod02,obj03,obj04 # comment`
  - Adding comment to end of statement

- `# comment line
  -a mod01,mod02,obj03,obj04`
  - Inserting a comment line

- `-a mod01,mod02,obj03, obj04`
  - line feed after comma, continues the parameter

- `-a mod01,mod02,obj03, obj04`
  - line feed before comma, continues the parameter

- `-a mod01,mod02,obj03, # comment obj04`
  - inserting a comment statement continues the parameter

- `-a mod01,mod02,obj03,obj04`
  - line feed after `-a` continues all parameters


4.5 Default Option File

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

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

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

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

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

<table>
<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>flnk907s</td>
<td>flnk907.opt</td>
</tr>
<tr>
<td>Librarian</td>
<td>flib907s</td>
<td>flib907.opt</td>
</tr>
<tr>
<td></td>
<td>f2ms</td>
<td>f2m.opt</td>
</tr>
<tr>
<td></td>
<td>f2hs</td>
<td>f2h.opt</td>
</tr>
<tr>
<td></td>
<td>m2ms</td>
<td>m2m.opt</td>
</tr>
<tr>
<td></td>
<td>h2hs</td>
<td>h2h.opt</td>
</tr>
<tr>
<td></td>
<td>m2bs</td>
<td>m2b.opt</td>
</tr>
<tr>
<td></td>
<td>h2bs</td>
<td>h2b.opt</td>
</tr>
<tr>
<td></td>
<td>f2is</td>
<td>f2i.opt</td>
</tr>
<tr>
<td></td>
<td>f2es</td>
<td>f2e.opt</td>
</tr>
<tr>
<td></td>
<td>m2is</td>
<td>m2i.opt</td>
</tr>
<tr>
<td></td>
<td>m2es</td>
<td>m2e.opt</td>
</tr>
<tr>
<td></td>
<td>i2ms</td>
<td>i2m.opt</td>
</tr>
<tr>
<td></td>
<td>e2ms</td>
<td>e2m.opt</td>
</tr>
</tbody>
</table>

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

- When the environment variable OPT907 or OPT has already been set. The file in the directory that is set by the environment variable is referred to.
  - Linker, librarian
    - %OPT907\default option file
  - Object tool
    - %OPT\default option file
When the environment variable OPT907 or OPT has not been set
The default option file in the development environment directory is referred to.

- Linker, librarian
  
  `%FETOOL%\LIB\907\default option file`

- Object tool
  
  `%FETOOL%\LIB\default option file`

  **Note:** When default option file cannot be found, the linkage kit does not issue error message.
PART 2  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 Overview of a Linker
5.2 Functions of a Linker
5.3 Types of Sections
5.4 Combining Sections
5.5 Locating Sections
5.6 Automatically Locating Sections
5.7 Searching Libraries
5.8 ROM and RAM Areas
5.9 Sections to be Transferred from ROM to RAM
5.10 Location of DIR Attribute Section
5.11 CPU Information File
5.12 Mixing of Objects for a Linker
5.13 Mixing of Objects with Different Function-call Interfaces in Linker
5.1 Overview of a Linker

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

The larger a program to be developed, the more difficult it becomes to describe everything in one source program. Also, if you develop a program using a C compiler, you usually need to import library files in C. A linker is used to combine multiple object modules that are related with each other, then allocate memory location addresses to create a load module in the executable form.

Figure 5.1 shows the relationship between a linker and input-output files.
5.2 Functions of a Linker

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

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

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

- There are the following four types of input files:
  -- Object module file that an assembler outputs
  -- List file
  -- Relative format load module file that a linker outputs
  -- Library file
- Object module files and load module files are processed in the order that they are written on a command line or in an option file.
- There are the following three types of output files:
  -- Absolute format load module file that is the final objective
  -- Relative format load module that can be input again
  -- Link map list file
- The output load module format (absolute or relative) may be specified and the output file name may be changed.
- For a map list, the number of lines per page and the width of a page may be changed.
- Specify an option to output one of the following four types of files.
  To output these files, the link load module must be in the absolute format.
  -- Absolute format assemble list file is created by adding to the absolute format a list file that an assembler has output.
  -- External symbol cross-reference information list that shows the cross-reference between external defined symbols and reference symbols used in modules
  -- Local symbol information list that shows the information on local symbols used in each module
  -- Section detailed map list that shows section location addresses in each module
- Messages consist of startup messages including the program version number, help messages briefly describing how to use the program, and error messages. You can specify whether or not to output a startup message and the level of detecting a warning status.
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 Section 5.2.2, “Control on Combining and Locating Sections”.

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

■ Control on searching libraries

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

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

■ Setting entry addresses and symbol values

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

5.2 Functions of a Linker

5.2.1 Control on Input-Output Files and Messages

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

- **Specifying input object files**
  The input files for a linker include object module files that an assembler outputs and relative format load module files that a linker outputs.
  All the input files must be specified, which can be facilitated by using wildcard.

- **Specifying an output load module file name**
  An output load module file name created after linking is based on the file name of the module that a linker first inputs.
  This function is provided to change the default output file name because it is often inappropriate as a name to represent the entire linking result.
  In particular, the output file name is difficult to understand if a file name is specified using a wildcard. In such a case, you are recommended to specify a file name.

- **Inheriting debugging information**
  Information on symbols and source files is required for debugging.
  If debugging information is specified to be created (-g option) in C or for an assembler, an object module will contain debugging information.
  A linker inherits this debugging information. You can decide either to output it into a load module or to delete it.

- **Specifying the output format**
  Use this function to specify creating an absolute or relative format load module as a result of linking.

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

- **Changing the format of a list file**
  Page control is performed when a list file is created. You may change the number of lines per page and number of characters per line.
  A long symbol name is truncated to fit into one line. You can specify to display the name exactly as you defined it.

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

- **Selecting whether or not to display a startup message**
  You can select whether or not to display the tool name and the copyright at startup.
Selecting whether or not to display a termination message.

You can select whether or not to display a termination message.
5.2 Functions of a Linker

5.2.2 Control on Combining and Locating Sections

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

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

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

- Specifying the order of locating sections and the location addresses
  All the sections may be located in any area in any order. To specify a section name, use a wildcard.

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

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

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

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

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

  This function enables these operations. For details, see Section 5.9, “Sections to be Transferred from ROM to RAM”.
5.2.3 Control on Searching Libraries

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

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

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

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

- **Inhibiting the search for a library**
  
  You can disable the search of a default library or all the libraries.
5.2.4 Setting Entry Addresses and Symbol Values

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

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

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

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

The minimum unit that a linker can combine is a section. Depending on the purpose of using sections in a program, they are located and combined differently. Section 5.3 describes section names, contents types, location attributes, and combination attributes.

- **Section name**
  A section name is used to identify a section.

- **Types of section contents**
  Depending on the purpose of usage, there are the following seven types of section contents:
  An assembler determines the attributes of execution, read, or write. Table 5.3a shows the section types.

  **Table 5.3a Section types**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE</td>
<td>Program code area</td>
<td>Executable, Read</td>
</tr>
<tr>
<td>DATA</td>
<td>Variable area</td>
<td>Read, Write</td>
</tr>
<tr>
<td>CONST</td>
<td>Area of variable with initial value</td>
<td>Read</td>
</tr>
<tr>
<td>STACK</td>
<td>Stack area</td>
<td>Read, Write</td>
</tr>
<tr>
<td>IO</td>
<td>I/O area</td>
<td>Read, Write</td>
</tr>
<tr>
<td>DIR</td>
<td>direst area</td>
<td>Read, Write</td>
</tr>
<tr>
<td>DIRCONST</td>
<td>direct area with initial values</td>
<td>Read</td>
</tr>
</tbody>
</table>

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

  **Table 5.3b Section location attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>Section in which absolute addresses are specified</td>
</tr>
<tr>
<td>REL</td>
<td>Relocatable section</td>
</tr>
</tbody>
</table>

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

  **Table 5.3c Section combination attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC</td>
<td>Sections are combined in succession.</td>
</tr>
<tr>
<td>COMMON</td>
<td>Sections are combined and overlapped at the same address.</td>
</tr>
</tbody>
</table>
5.3 Types of Sections

- Section identification

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

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

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

For a linker, we often write “combining multiple objects” but more accurately we should write “combining sections in objects”.
A section may be combined through a simple connection combination (PUBLIC) and shared combination (COMMON).

- **Simple connection combination of sections**

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

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

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

- **Shared combination of sections**

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

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

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

  The entire size after combination is the larger one of the A-1 and A-2 sizes.
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 Section 5.5.1, “Example of Location when the Order of Combining Sections is not Specified”, 5.5.2, “Example of Location when the Order of Combining Sections is Specified”, and 5.5.3, “Example of Location when the Section Group is Specified”.
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 5.5a.

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

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

Example of location when the order of combining sections is specified

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

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

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

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

Example of location when the section group is specified

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

Note: Sections B and E have the COMMON attribute and other sections the PUBLIC attribute. The order of location is not specified and Sections C, D, and F are put into the same group.

Figure 5.5c Example of location when the section group is specified
5.6 Automatically Locating Sections

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

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

- Automatically Locating Sections
  
  This linker supports automatically locating the following two types of sections:
  
  • Automatically locating sections when -AL 1 is specified
  • Automatically locating sections when -AL 2 is specified

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

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

Determining location addresses

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

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

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

### Table 5.6a Alignment values and sizes of sections

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

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

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

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

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

### Table 5.6b Alignment values and sizes of sections

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

Sections are located in the smallest available area where they can be located.
### Example of location when -AL 1 is specified

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

- *-ro ROM=0xCF000/0xCF8FF*
- *-sc*
- *code1+code2+code3+code4+code5+code6=ROM*
- *-AL 1 :*

**Figure 5.6a  Option specification for a linker**

<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>

**Table 5.6c  Contents of sections**

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

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

**Figure 5.6b  Example of location when -AL 1 is specified**
5.6.2 Automatically Locating Sections when -AL 2 is Specified

If -AL 2 is specified, a linker determines whether sections without any location address specified in the -sc option should be located in the ROM or RAM area, and locates them in an available space of the selected area. Section 5.6.2 describes the location destinations for each section type and the order of determining location addresses. Figure 5.6d 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 5.6d.

Table 5.6d  Section types and location destinations

<table>
<thead>
<tr>
<th>Location destination</th>
<th>Section type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM area</td>
<td>CODE</td>
</tr>
<tr>
<td>(specified in -ro)</td>
<td>CONST</td>
</tr>
<tr>
<td></td>
<td>DIRCONST</td>
</tr>
<tr>
<td>RAM area</td>
<td>IO</td>
</tr>
<tr>
<td>(specified in -ra)</td>
<td>DIR</td>
</tr>
<tr>
<td></td>
<td>DATA</td>
</tr>
<tr>
<td></td>
<td>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 5.6e.

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

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

Table 5.6e  Section location destinations

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

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

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

Figure 5.6c  Linker option specification

Table 5.6f  Sections contained in modules

<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>0x0C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>p4</td>
<td>CODE</td>
<td>REL</td>
<td>0x1000</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>stk</td>
<td>STACK</td>
<td>REL</td>
<td>0x0100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ivect</td>
<td>CONST</td>
<td>REL</td>
<td>0x0100</td>
<td>-</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>0x0040</td>
<td>-</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>0x1000</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>d2</td>
<td>CONST</td>
<td>REL</td>
<td>0x0300</td>
<td>-</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>0x0100</td>
<td>-</td>
</tr>
</tbody>
</table>

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

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

Figure 5.6d  Example of location when -AL 2 is specified
5.7 Searching Libraries

Specify a library file to be searched by a linker in one of the following three ways:
- Setting the default library
- Specification in the -l option
- Specification in the -el option

■ Specifying a library to be searched

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

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

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

○ Setting the default library

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

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

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

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

○ Specification in the -l option

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

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

For details on the -l option, see Section 6.2.19, “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 Section 6.2.21, “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 5.7a.

- 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 named enclosed in () means a reference and that enclosed in {} means a definition.

**Figure 5.7a** Example of a search when there is one library file (1)
There are three examples of a search when there is one library file. Section 5.7.2 describes one of these examples as shown in Figure 5.7b.

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 named enclosed in () means a reference and that enclosed in {} means a definition.

Figure 5.7b  Example of a search when there is one library file (2)
5.7.3 Example of a Search when there is one Library File (3)

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

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 named enclosed in () means a reference and that enclosed in {} means a definition.

Figure 5.7c Example of a search when there is one library file (3)
5.7.4 Example of a Search when there are Multiple Library Files (1)

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

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 specified.

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

Libraries 1 and 2 are searched in this order.

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

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

Then, Library 2 is searched.

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

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

Figure 5.7d Example of a search when there are multiple library files (1)
5.7.5 Example of a Search when there are Multiple Library Files (2)

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

- 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 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 named enclosed in () means a reference and that enclosed in {} means a definition.
5.7.6 Processing when Library Files are Individually Specified

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

■ Processing when library files are individually specified

The external reference symbols for which the library files are individually specified are searched first.
Here, the (S2) symbol is assumed to be imported from Library 2.
Library 2 is searched for (S2) and Module C is linked.
To solve (S1), the program searches Library File 1 and links Module A.
(S2) in Library File 1 is not searched.

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

Be sure that, even if the specified libraries do not contain the target external defined symbol but another search target library does, the symbol is solved.

**Figure 5.7f** Processing when library files are individually specified
When you develop an application to be imported, there often are restrictions on the ROM and RAM sizes and the address range that can be used. By notifying these areas to a linker at the time of linking, you may check for a size exceeding the limit and a section located to an unusable address.

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

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

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

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

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

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

[Example 2]
-ro CodeA=0x1000/0x3FFF
-ra DataA=0x0180/0x57F

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

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

If you specify an address name to locate a section, this program can check whether the section is located in the specified address range.
5.9 Sections to be Transferred from ROM to RAM

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

■ Sections to be transferred from ROM to RAM

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

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

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

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

■ Using the sections to be transferred from ROM to RAM

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

The sections containing variables with initial values shall be INIT.

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

-sc DATA+INIT=0x1000,start+CODE+@INIT=0xC000

As shown in the figure, specify to locate INIT on RAM (0x1000), and also to locate INIT on ROM (xC000) using a section name starting with an at sign, @.

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

■ 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:

```
.programsample
.sectioninit, data
val1:.word0x1234
val2:.word0x5678
.sectionData, data
val3:.res.w0x1
.sectionProg1, code
movw A, val1
movw rw2, A
```
The above program has the init section containing variables with initial values, the Data section containing variable areas, and the Prog1 and Prog2 sections containing program code.

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

<table>
<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>

○...Enabled  ×...Disabled

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

<table>
<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 linkerWrite 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>

○...Enabled  ×...Disabled
5.10 Location of DIR Attribute Section

For F²MC-16 family, two or more direct pages can be used. But, be careful when linking modules that are output by the compiler.

■ Direct page

A direct page is an area where DIR-attribute sections are located.

The size of 1 direct page is 256 bytes.

For F²MC-16 Family, two or more direct pages can be used. For how to use two or more direct pages, refer to the hardware manual for each chip.

■ Location of DIR-attribute sections

DIR-attribute sections must be placed continuously, and they must not be linked across a 256-byte boundary. DIR-attribute sections are located as follows when linked:

● When the total size of the DIR-attribute sections is 256 bytes or less

The linker places the DIR-attribute sections on 1 direct page.

Figure 5.10a  When the total size of the DIR-attribute sections is 256 bytes or less

● When the size of 1 DIR-attribute section exceeds 256 bytes

When the size of 1 DIR-attribute section exceeds 256 bytes, the linker outputs an error (**E4363L: Total size of the DIR attribute sections exceeded 256 bytes**).
When the size of 1 DIR-attribute section exceeds 256 bytes

- When the size of each DIR-attribute section is 256 bytes or less, and the total size of them exceeds 256 bytes

The linker locates the DIR-attribute sections on two or more direct pages in such a manner that each section fits onto each direct page (that is, the sections are not placed across a 256-byte boundary).

When the size of each DIR-attribute section is 256 bytes or less, and the total size of them exceeds 256 bytes

The sections are located on two or more direct pages in such a manner that they are not placed across a 256-byte boundary.
■ Check on the total size of the DIR-attribute sections

The compiler performs processing, based on the assumption that DIR-attribute sections are located direct page.

In the case of Figure 5.10c above, a DIR-attribute section is placed in an area that cannot be accessed. (In the case of Figure 5.10c, DIRINIT cannot be accessed.)

The linker has the function of outputting a warning when the total size of the DIR-attribute sections exceeds 256 bytes. (See Section 6.2.42 Specifying that Check on the Total Size of the DIR-attribute Sections (-check_dirsize).)

If the warning is outputted, modify the source so that the total size of the DIR-attribute sections becomes 256 bytes or less.

Note:

In the following cases, no problem occurs even when the total size of the DIR-attribute sections exceeds 256 bytes:

- When an assembler source uses two or more direct pages
- When each task of Real Time OS uses a different direct page

In a case where no problem occurs even when the total size of the DIR-attribute sections exceeds 256 bytes, check on the total size of the DIR-attribute sections can be inhibited. (See Section 6.2.43 Specifying that Check on the Total Size of the DIR-attribute Sections be Inhibited (-Xcheck_dirsize).)
CHAPTER 5 SPECIFICATIONS OF A LINKER

5.11 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
  - 907.csv

- Search Directory
  - %FETOOL%\LIB\907

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

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

Note: If the CPU information file cannot be found, or the appropriate MB number in the CPU information file does not exist, the Linker will issue a error.
5.12 Mixing of Objects for a Linker

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

- Mixing of objects

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

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

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

Table 5.12a lists the objects and libraries for lower CPUs that may be mixed with the -cpu option.

Table 5.12a  List of CPUs for which objects may be mixed with the -cpu option

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

◎ : Same target specified. This is not mixing.

◯ : Mixing allowed. However, the linker outputs information if 2 is specified in the -w option.

× : Mixing not allowed. The linker outputs an error.
5.13 Mixing of Objects with Different Function-call Interfaces in Linker

Mixture of objects in the linker that have different function-call interface is explained here.

### Mixing of objects with different function-call interfaces in Linker

Mixture of objects in the linker that have different function-call interfaces

F2MC-16 Family supports the following function-call interfaces:

- Stack argument passing (this is the existing function-call interface)
- Register argument passing (RW0, RW1)

Between objects for register argument passing and objects for stack argument passing, the way of passing arguments and the way of using registers are different. So, when objects having different function-call interfaces are mixed and used, programs do not operate normally.

In the linker, objects for stack argument passing and objects for register argument passing must not be mixed even if the target CPU is the same for these two types of objects.

If an object having a different function-call interface is mixed in the linker, the linker outputs the following error and stops processing:

```
(*** E4313L: The module is different calling interface of a function (file name).)
```

When objects have the same function call interface, an object mixture check is performed in the target CPU as usual. For object mixture "check in the target CPU, see 5.12 Mixing of Objects for a Linker".
CHAPTER 6  LINKER OPTIONS

This chapter explains each linker option in more detail.

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

Options are available to specify linker operations in more detail. The following lists the option names and function outlines.

### List of linker options

For details of the parameters and functions required for the options, see the explanations of each option. For explanations of common options, see “CHAPTER 3 COMMON OPTIONS”.

<table>
<thead>
<tr>
<th>Table 6.1 List of linker options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function</strong></td>
</tr>
<tr>
<td>Specification regarding output load modules</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Specification regarding output lists</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Specification regarding output messages</td>
</tr>
<tr>
<td>Allocation/link 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>Library control options</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Table 6.1 List of linker options (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry address specification</td>
<td>-e</td>
<td></td>
</tr>
<tr>
<td>Dummy setting external symbol values</td>
<td>-df</td>
<td></td>
</tr>
<tr>
<td>Target CPU specification</td>
<td>-cpu need</td>
<td></td>
</tr>
<tr>
<td>CPU information file specification</td>
<td>-cif</td>
<td></td>
</tr>
<tr>
<td>Inhibiting Check for Presence of Debug Data</td>
<td>-NC10302LIB</td>
<td></td>
</tr>
<tr>
<td>Function that sets automatically internal ROM/RAM area</td>
<td>-set_rora Default</td>
<td></td>
</tr>
<tr>
<td>Specifies to prevent the internal ROM/RAM area from being set automatically</td>
<td>-Xset_rora</td>
<td></td>
</tr>
<tr>
<td>User-specified-area check specification</td>
<td>-check_rora</td>
<td></td>
</tr>
<tr>
<td>User-specified-area check suppression specification</td>
<td>-Xcheck_rora Default</td>
<td></td>
</tr>
<tr>
<td>Section-placed-area check specification</td>
<td>-check_locate</td>
<td></td>
</tr>
<tr>
<td>Section-placed-area check suppression specification</td>
<td>-Xcheck_locate Default</td>
<td></td>
</tr>
<tr>
<td>Check User-unspecified Section</td>
<td>-check_section</td>
<td></td>
</tr>
<tr>
<td>Inhibit check for presence of user-unspecified section</td>
<td>-Xcheck_section Default</td>
<td></td>
</tr>
<tr>
<td>Checks Standby Mode Transition Instruction</td>
<td>-check_SCF Default</td>
<td></td>
</tr>
<tr>
<td>Prohibits checks on the Standby Mode Transition Instruction</td>
<td>-Xcheck_SCF</td>
<td></td>
</tr>
<tr>
<td>Checks only in fetch units</td>
<td>-Wf</td>
<td></td>
</tr>
<tr>
<td>Specified output load module for register argument passing</td>
<td>-rp</td>
<td></td>
</tr>
<tr>
<td>Specified output load module for stack argument passing</td>
<td>-Xrp Default</td>
<td></td>
</tr>
<tr>
<td>Specified that Check on the Total Size of the DIR-attribute Sections</td>
<td>-check_dirsize Default</td>
<td></td>
</tr>
<tr>
<td>Specified that Check on the Total Size of the DIR-attribute Sections be inhibited</td>
<td>-Xcheck_dirsize</td>
<td></td>
</tr>
</tbody>
</table>
## Table 6.1 List of linker options (Continued)

<table>
<thead>
<tr>
<th>Options regarding the absolute format assemble list output</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification for relative assemble list input directory</td>
<td>-alin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output directory</td>
<td>-alout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output module</td>
<td>-alsf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting absolute assemble list output</td>
<td>-Xals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list output</td>
<td>-alr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list output module</td>
<td>-alrf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting ROM/RAM and ARRAY list output</td>
<td>-Xalr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list symbol and address display position</td>
<td>-na/-an</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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 external symbol cross-reference information list output</td>
<td>-xl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for external symbol cross-reference information list file name</td>
<td>-xlf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting the external symbol crossreference information list output</td>
<td>-Xxl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting the external symbol crossreference information list output module</td>
<td>-sl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for local symbol list output</td>
<td>-slf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specification for local symbol list file name</td>
<td>-Xsl</td>
<td></td>
<td></td>
</tr>
<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></td>
<td>Default</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></td>
<td>Default</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></td>
<td>Default</td>
</tr>
</tbody>
</table>
6.2 Details of Linker Options

This section explains each option of the linker.
For common options in the linkage kit, see “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)” to “6.2.5 Relative Format Load Module Output Specification (-r)”.

■ Options related to the output list
Details of options related to the output list in “6.2.6 Map List File Name Specification (-m)” to “6.2.11 Specification of the Number of Lines on one List Page (-pl)”.

■ Specification related to output messages
Details of options related to output messages in “6.2.12 Warning Message Output Level Specification (-w)”.

■ Allocation/link options
Details of options related to the allocation/link in “6.2.13 ROM Area Specification (-ro)” to “6.2.18 Automatic Allocation Specification (-AL)”.

■ Library control option
Details of options related to the library control in “6.2.19 Retrieval Library File Specification (-l)” to “6.2.23 Default Library Retrieval Inhibit Specification (-nd)”.

■ Other link control options
Details of options related to other link controls in “6.2.24 Entry Address Specification (-e)” to “6.2.43 Specifying that Check on the Total Size of the DIR-attribute Sections be Inhibited (-X check_dirsize)”.

■ Options related to the absolute format assemble list output
Details of options related to the absolute format assemble list output in “6.2.44 Relative Assemble List Input Directory Specification (-alin)” to “6.2.52 ROM/RAM and ARRAY List Symbol and Address Display Position Specification (-na,-an)”.

■ Options related to the object content list output
Details of options related to the object content list output in “6.2.53 External Symbol Cross-reference Information List Output Specification (-xl)” to “6.2.61 Section Detail Map List Output Inhibit Specification (-Xml)”.
6.2.1 Output Load Module File Name Specification (-o)

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

Output load module file name specification (-o)

[FORMAT]

-o <Load module file name>  (Default)

Parameters

<Load module file name>
Output load module file name

Explanation

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

<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

flnk907s putc.obj,getc.obj
A load module file is created with the file name putc.abs.

Example 2

flnk907s *.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

flnk907s *.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

flnk907s *.obj -r -o outfile.rel
A load module file in the relative format is created with the file name outfile.rel.
6.2.2 Debug Information Output Specification (-g)

Object modules created by specifying the output of debug information in the C compiler or assembler contain debug information to be used by the debugger. To use debug information after linking, specify the -g option.

- Debug information output specification (-g)

[FORMAT]

-g

[Parameters]
None

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

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

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

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

[Example]

flnk907s -f rllnk.opt b1 b2

- rllnk.opt # relocatable LM output
- g # debug info.
- pw 100 # page width
- o rel1.rel # output filename
CHAPTER 6  LINKER OPTIONS

6.2.3  Debug Information Delete Specification (-Xg)

Object modules that are created by specifying the output of debug information in the C compiler or assembler contain debug information to be used by the debugger. To remove debug information after linking, specify the -Xg option. Otherwise, do not specify the -g option.

■ Debug information delete specification (-Xg)

[FORMAT]

- Xg  ( Default )

[Parameters]

None

[Explanation]

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

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

[Example]

```
flnk907s -f rllnk.opt a1 a2 a3 -Xg -o a123.rel
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
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

flnk907s 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, 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]

flnk907s a1 a2 a3 -r -o a123.rel

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

Note: If the extension of the input file specified first is “.rel”, the output file name will be the same. Since the contents of the input file are not saved in this case, specify the output file name with -o option to avoid any inconvenience.
6.2.6 Map List File Name Specification (-m)

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

- Map list file name specification (-m)
  
  [FORMAT]

  -m <Map list file name>  (Default)

  [Parameters]

  <Map list file name>
  Output map list file name
  The parameters cannot be omitted.

  [Explanation]

  By default, the linker outputs a map list file. At this point, a file is created with the name of the output load module file whose extension is changed to “.mp1”.

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

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

  [Example]

  flnk907s a1 a2 a3 -r -o a123.rel -m a123.map

  The map list file name output by the linker is changed to a123.map.
6.2.7 Map List Output Inhibit Specification (-Xm)

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

Map list output inhibit specification (-Xm)

[FORMAT]

- Xm

[Parameters]
None

[Explanation]
This option inhibits output of map list files.
If the -Xm option is specified after the -m option, the -m option can be canceled.
By specifying the -Xm option, the -dt -pw, and -pl options can be canceled.

[Example]
flnk907s a1 a2 a3 -r -o a123.rel -Xm
Creating a map list file is inhibited.
6.2.8 Canceling the Omission of Names Displayed in the List (-dt)

Names such as the section names and symbol names are displayed in the map list and object content list of the linker. By default list output, considering legibility of the lists, long names are only partially displayed. This option instructs to output the names without omission. For the display format of the map lists, see “7.2 Link List 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 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]

  flnk907s a1 a2 a3 -o a123.abs -m a123.map -dt

  The symbol names and section names in use are displayed in the list without omission.
6.2.9 Output Specification of the Map List File Memory Used Information List (-mmi)

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

- 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.

- Example
  flnk907s l0mp00 im _lnk -mmi -ra RAM1=0x1000/0x1FFF, RAM2=0x2000/0x23FF
  -ro ROM1=0xBC000/0xBCFFF, ROM2=0xBD000/0xBFFFF -AL 2

- Note: The following will not be output despite -mmi being specified.
  (1) The memory area is not specified. (-ra or -ro is not specified).
  (2) The Map List File output is not valid.
6.2.10 Specification of the Number of Digits in the List Line (-pw)

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

- Specification of the number of digits in the list line (-pw)

[FORMAT]

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

[Parameters]

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

[Explanation]

The length of one line of the link list file and object content list is specified.
If this value is not specified, 132 is set.

[Example]

flnk907s a1 a2 a3 -o a123.abs -m a123.map -dt -pw 80

The number of digits to be displayed in one line is set to 80.
6.2.11 Specification of the Number of Lines on one List Page (-pl)

By default, up to 60 lines are displayed on one page of the map list and object content list output by the linker.
This option is specified to change the number of lines to be displayed on one page.

- Specification of the number of lines on one list page (-pl)

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

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

  [Explanation]
  The number of lines on one page of the link list file and object content list is specified.
  If this value is not specified, 60 is set.
  If 0 is specified, page control is canceled.

  [Example]
  
  flnk907s a1 a2 a3 -o a123.abs -m a123.map -dt -pl 64 -pw 100
  
  The number of lines to be displayed on one page of the list is set to 64.
6.2.12 Warning Message Output Level Specification (-w)

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

- Warning message output level specification (-w)

[FORMAT]

- w < Numeric value >

[Parameters]

<Numeric value>
Specify 0, 1, or 2 as a warning level.

[Explanation]

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

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

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

[Explanation]

flnk907s a1 a2 a3 -o a123.abs -w 2 -Xm

All messages are output.
CHAPTER 6  LINKER OPTIONS

6.2.13  ROM Area Specification (-ro)

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

ROM area specification (-ro)

[Format]

- ro  < Area name > = < Start address > / < End address >

[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. Use the addresses including the bank number (8-bit) to specify the start address and end address. The start address and end address must not extend over a bank.

[Example]

flnk907s *.obj -o ap.abs -ro RomA=0x1000/0x2fff -sc code=RomA ...

Section allocation of the section name code to the address 0x1000-0x2fff is specified. The section is allocated starting with the address 0x1000. A warning message is output if the address 0x2fff is exceeded.
6.2.14 RAM Area Specification (-ra)

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

- RAM area specification (-ra)

[FORMAT]

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

[Parameters]

<Area name>
Name to indicate the address area to be set

<Start address>
Start address of the address area to be set

<End address>
End address of the address area to be set

[Explanation]

The ROM areas are defined. As many areas as required can be defined. Specify the start address and end address, then name the area.

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

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

Use the addresses including the bank number (8-bit) to specify the start address and end address. The start address and end address must not extend over a bank.

[Example]

flnk907s *.obj -o ap.abs -ra RamD=0x0100/0x01ff -sc data=RamD ...

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

The section is allocated starting with the address 0x0100, and a warning message is output if the address 0x01ff is exceeded.
6.2.15 Section Allocation Order/Address Specification (-sc)

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

- Section allocation order/address specification (-sc)

  [Format]

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

  [Parameters]

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

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

  <Alignment>
  The boundary of the next area that can be allocated can be specified using the following keywords.
  - BYTE : Allocates at the byte boundary that can be allocated. If any allocated segment exists, allocation is carried out without a gap.
  - WORD : Allocates at the word boundary of the address that can be allocated.
  - PAGE : Allocates at the page boundary of the address that can be allocated.

  <Address>
  Start address of the allocation

  <Area name>
  Area name specified in the ROM/RAM specification option

  [Explanation]

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

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

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

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

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

[Example 1]

flnk907s *.obj -o ap.abs -sc "*/code"=0xC1000,D=0x1000 ...

Figure 6.2a 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 6.2a Section allocation example 1](image)

[Example 2]

flnk907s *.obj -o ap.abs -ro ROM=0xC1000/0xCFFFF -ra RAM=0x1000/0x13FF -sc P+code+text1=ROM,D=RAM

Figure 6.2b shows the allocation example in this case.

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

![Figure 6.2b 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><code>*</code></td>
<td>-sc &quot;*/code&quot;</td>
<td>Match with all sections whose content type is code</td>
</tr>
<tr>
<td><strong>Match</strong></td>
<td>ab_1, code_1, XXsect, etc</td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><code>_*</code></td>
<td>-sc &quot;ab_*/code&quot;</td>
<td>Match with all sections whose content type is code and whose first three characters are &quot;ab_&quot;</td>
</tr>
<tr>
<td><strong>Match</strong></td>
<td>ab_1, ab_XX, ab_, etc</td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td>aab_XX, ab, etc</td>
<td></td>
</tr>
<tr>
<td><code>_*</code></td>
<td>-sc &quot;*_1/code&quot;</td>
<td>Match with all sections whose content type is code and whose last two characters are &quot;_1&quot;</td>
</tr>
<tr>
<td><strong>Match</strong></td>
<td>ab_1, XX_1, _1, etc</td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td>ab_11, _, etc</td>
<td></td>
</tr>
<tr>
<td><code>_*</code></td>
<td>-sc &quot;ab_*_1/code&quot;</td>
<td>Match with all sections whose content type is code, whose first three characters are &quot;ab_&quot;, and whose last two characters are &quot;_1&quot;</td>
</tr>
<tr>
<td><strong>Match</strong></td>
<td>ab_XX, ab_1, etc</td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td>aab_XX_11, ab_1, etc</td>
<td></td>
</tr>
</tbody>
</table>

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

- `sc *=RAM` -sc `CODE=RAM`

All wild card specifying, in specifying like the one above, agree with sections other than the 'CODE' that can be rearranged (sections arrangement attributes that are REL).
## 6.2.16 Section Group Specification (-gr)

Multiple sections are linked to create a group according to the purpose of the user and a group name is given to the group. By using this group name when specifying the section allocation, multiple sections can be handled as one group.

### Section group specification (-gr)

[Format]

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

[Parameters]

- **<Group name>**
  - Generic name of multiple sections to be grouped
- **<Section name list>**
  - Describes section names to be grouped.
  - The wild card can be used.
  - When specifying multiple names, link them with the + symbol.
- **<Content type>**
  - code, data, stack, const, dir, dirconst, IO

[Explanation]

Sections to be grouped and the order of allocation of sections in the group are specified. The order of section allocation follows the order described in the parameters. Each group name must be a unique name which does not overlap with the section names and other group names. A section that belongs to one group must not belong to another group. When using the wild card, indicate it using double quotation marks. Do not use double quotation marks (" ") in an option file.

[Example 1]

```
flnk907s *.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]

```
flnk907s *.obj -o ap.abs -gr cdgrp="*/code" -sc cdgrp=0xC1000
```

All sections whose content type is code are linked to a group and the group name cdgrp is given to the group. Then the group is allocated to the address 0xC1000 using the -sc option.
6.2.17 Register Bank Area Specification (-rg)

The register bank area used by programs is specified.

- Register bank area specification (-rg)

  [Format]

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

  [Parameters]

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

  [Explanation]

  The register bank area used by programs is specified.

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

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

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

  Table 6.2a shows the areas reserved by the -rg option.
Table 6.2a  Register bank numbers and areas

<table>
<thead>
<tr>
<th>No.</th>
<th>Reserved area</th>
<th>No.</th>
<th>Reserved area</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000180 - 0x00018F</td>
<td>16</td>
<td>0x000280 - 0x00028F</td>
</tr>
<tr>
<td>1</td>
<td>0x000190 - 0x00019F</td>
<td>17</td>
<td>0x000290 - 0x00029F</td>
</tr>
<tr>
<td>2</td>
<td>0x0001A0 - 0x0001AF</td>
<td>18</td>
<td>0x0002A0 - 0x0002AF</td>
</tr>
<tr>
<td>3</td>
<td>0x0001B0 - 0x0001BF</td>
<td>19</td>
<td>0x0002B0 - 0x0002BF</td>
</tr>
<tr>
<td>4</td>
<td>0x0001C0 - 0x0001CF</td>
<td>20</td>
<td>0x0002C0 - 0x0002CF</td>
</tr>
<tr>
<td>5</td>
<td>0x0001D0 - 0x0001DF</td>
<td>21</td>
<td>0x0002D0 - 0x0002DF</td>
</tr>
<tr>
<td>6</td>
<td>0x0001E0 - 0x0001FF</td>
<td>22</td>
<td>0x0002E0 - 0x0002FF</td>
</tr>
<tr>
<td>7</td>
<td>0x0001F0 - 0x0001FF</td>
<td>23</td>
<td>0x0002F0 - 0x0002FF</td>
</tr>
<tr>
<td>8</td>
<td>0x000200 - 0x00020F</td>
<td>24</td>
<td>0x000300 - 0x00030F</td>
</tr>
<tr>
<td>9</td>
<td>0x000210 - 0x00021F</td>
<td>25</td>
<td>0x000310 - 0x00031F</td>
</tr>
<tr>
<td>10</td>
<td>0x000220 - 0x00022F</td>
<td>26</td>
<td>0x000320 - 0x00032F</td>
</tr>
<tr>
<td>11</td>
<td>0x000230 - 0x00023F</td>
<td>27</td>
<td>0x000330 - 0x00033F</td>
</tr>
<tr>
<td>12</td>
<td>0x000240 - 0x00024F</td>
<td>28</td>
<td>0x000340 - 0x00034F</td>
</tr>
<tr>
<td>13</td>
<td>0x000250 - 0x00025F</td>
<td>29</td>
<td>0x000350 - 0x00035F</td>
</tr>
<tr>
<td>14</td>
<td>0x000260 - 0x00026F</td>
<td>30</td>
<td>0x000360 - 0x00036F</td>
</tr>
<tr>
<td>15</td>
<td>0x000270 - 0x00027F</td>
<td>31</td>
<td>0x000370 - 0x00037F</td>
</tr>
</tbody>
</table>

[Example]

```
flnk907s *.obj -o ap.abs -rg 0,1,5-7...
```

The register bank areas 0, 1, 5, 6, and 7 are reserved.
### Automatic Allocation Specification (-AL)

Automatic allocation of sections is specified.

#### Automatic allocation specification (-AL)

**[Format]**

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

**[Parameters]**

- **<0>**
  - No automatic allocation (default)

- **<1>**
  - If an absolute section exists in the area, the order of allocation is changed appropriately to avoid overlapping with the section.

- **<2>**
  - Whether to allocate to the ROM area or RAM area is determined based on the section attributes. Then allocation is carried out to free space of each area.
    
    (link907a automatic allocation compatible)

**[Explanation]**

Automatic allocation of sections is specified.

- If the parameter is 1
  - If an absolute section exists when allocating sections to the areas specified by the -ra or -ro option, relocatable sections are allocated in such a way that allocation addresses do not overlap. In such cases, sections are allocated in descending order of the alignment value and size starting with the section of the largest alignment value and size. In this way, optimal allocation is implemented where free space is minimum.

- If the parameter is 2
  - Whether to allocate sections whose allocation is not specified by the -sc option to the ROM area or RAM area is determined based on the section attributes. Then such sections are allocated to free space of each area.

See also “5.6 Automatically Locating Sections”.

**[Example]**

```
flnk907s -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.
Note: Even if -AL 1 is specified, automatic allocation is not carried out in the following cases.
- No area is set. (-ra or -ro is not set)
- No area name is used in addressing of -sc.
- No absolute section exists to be allocated in the specified area.
  Even if -AL 2 is specified, errors occur and no link processing is performed in the following cases.
- No area is set. (-ra or -ro is not set)
  If -w 2 is specified and the automatic allocation function works, a message is output.
6.2.19 Retrieval Library File Specification (-l)

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

- Retrieval library file specification (-l)

[Format]

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

[Parameters]

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

[Explanation]

Library files are retrieved in the specified order.
Libraries specified here are retrieved before the default library.
If the library files are specified without path names, directories are retrieved in the following order: the directory specified by the -L option, the directory specified in the environmental variable LIB907, and the system library path derived from the environmental variable FETOOL.
The current directory is not retrieved. If you want to retrieve the current directory, specify either the -L option or a period (.) in the environmental variable LIB907.
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]

```
flnk907s *.obj -o ap.abs -l ..\lib\com.lib,libu
flnk907s *.obj -o ap.abs -l "p*.lib"
```

All library files whose file name's first character is “p” are retrieved.
### 6.2.20 Library Retrieval Path Specification (-L)

The path name for retrieving the library file is specified.

<table>
<thead>
<tr>
<th>Library retrieval path specification (-L)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Format</strong></td>
</tr>
<tr>
<td>- L &lt; Library path name &gt; [ , ... ]</td>
</tr>
</tbody>
</table>

**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 variableLIB907 so that this option need not be specified.

The C libraries attached to the C compiler are stored in the path specified by LIB907. 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 LIB907 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**

flnk907s *.obj -o ap.abs -L C:\usr\usrlib -l com.lib,libu
6.2.21 Library Specification for Each Symbol (-el)

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

Library specification for each symbol (-el)

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

[Parameters]
- <Symbol name list>
  Describe the external reference symbol names.
  When specifying multiple symbols, separate them with /.

- <Library file name>
  Library file name to be retrieved.
  Library file names with path names can also be specified. The wild card_ cannot be used.

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

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

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

Examine whether library files can be recreated.

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

[Example]
flnk907s *.obj -o sp.abs -L C:\usr\usrlib -l libu,sublib -el sym1=sublib
6.2.22 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]**
  
  flnk907s -L C:\usr\usr\lib -l libu,sublib *.obj -o ap.abs -nl
  
  The -L, -l, and -el options previously specified are canceled and library retrieval including the default library is inhibited.

  **[Example 2]**
  
  flnk907s -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.
CHAPTER 6 LINKER OPTIONS

6.2.23 Default Library Retrieval Inhibit Specification (-nd)

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

- Default library retrieval inhibit specification (-nd)

[Format]

- nd

[Parameters]

None

[Explanation]

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

[Example]

flnk907s -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.
6.2.24 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]
flnk907s *.obj -o ap.abs -e ProgStart
### Dummy Setting of External Symbol Values (-df)

Undefined symbol values of a user program are forced to be defined.

**[Format]**

- \(-\text{df} <\text{Symbol name}> = \{ <\text{Numeric value}> | <\text{External definition symbol name}> \)\)

**[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]**

```
flnk907s -L \usr\usrlib -l libu,sublib *.obj -o ap.abs -df Sym1=100
```

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

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

### Target CPU specification (-cpu)

**Format**

External definition symbol name

```bash
- cpu < MB number>name >
```

**Parameters**

- `<MB number>`
  - MB number of the target CPU

**Explanation**

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

**Example**

```bash
flnk907s *.obj -o ap.abs -cpu MB90611A
flnk907s *.obj -o ap.abs -cpu MB90704
```

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

The CPU information file used by linker is specified.

- CPU Information File Specification (-cif)

[Format]
- cif < CPU information file name >

[Parameters]
< CPU information file name >
The CPU information file used by linker is specified.

[Explanation]
The CPU information file used by linker is specified.

[Example]
flnk907s *.obj -o ap.abs -cpu MB90704 -cif C:\Softune\lib\907\cpu_info\MB90704.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\907\907.csv

When installing the compiler assembler packs in a different directory and using the compiler, assembler and linkage kit instead of SOFTUNE workbench, specify -cif so that each tool can reference the same CPU information file.
6.2.28 Inhibiting Check for Presence of Debug Data (-NCI0302LIB)

This option inhibits check for presence of debug data in the module extracted from the library file.

Inhibiting Check for Presence of Debug Data (-NCI0302LIB)

[Format]

- NCI0302LIB

[Parameters]
NONE

[Explanation]
When debug data output (-g) and warning level 2 (-w2) are specified to operate the linker, the linker outputs the following data message to the module that has no debug data.

I0302L: Debug information not exist (file name)

When this option is specified, the linker does not output the above information message to the module extracted from the library file.

[Example]
flnk907s -cpu MB90704 -g -w 2 test.obj -l lib907c.lib
*** I0302L: Debug information not exist (C:\Softune\lib\907\lib907c.lib)

flnk907s -cpu MB90704 -g -w 2 test.obj -l lib907c.lib -NCI0302LIB

Information 'I0302L' is outputted.

flnk907s -cpu MB90704 -g -w 2 test.obj -l lib907c.lib -NCI0302LIB
Information 'I0302L' is not outputted.
6.2.29 Function that sets automatically internal ROM/RAM area (-set_rora)

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

- Function that sets automatically internal ROM/RAM area (-set_rora)
  
  [Format]
  
  - set_rora

  [Parameters]
  
  NONE

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

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

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

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

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

  These names are used by the -sc options.

  [Example]
  
  flnk907s *.obj -o ap.abs -cpu MB90704 -set_rora
### 6.2.30 Specifies to prevent the internal ROM/RAM area from being set automatically (-Xset_rora)

Prevents the information from being set regarding the internal ROM/RAM area of the targeted CPU that is referenced to the CPU information file.

<table>
<thead>
<tr>
<th>Specifies to prevent the internal ROM/RAM area from being set automatically (-Xset_rora)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[Format]</strong></td>
</tr>
<tr>
<td>- Xset_rora</td>
</tr>
<tr>
<td><strong>[Parameters]</strong></td>
</tr>
<tr>
<td>NONE</td>
</tr>
<tr>
<td><strong>[Explanation]</strong></td>
</tr>
<tr>
<td>Prevents the information from being set regarding the internal ROM/RAM area of the targeted CPU that is referenced to the CPU information file.</td>
</tr>
<tr>
<td><strong>[Example]</strong></td>
</tr>
<tr>
<td>flnk907s * .obj -o ap.abs -cpu MB90710 -Xset_rora</td>
</tr>
</tbody>
</table>
6.2.31 User-specified-area check specification (-check_rora)

This When the -check_rora option is previously specified, you can check whether the memory map has been changed, by changing only the specified MB number. The -check_rora option checks whether the specified ROM and RAM areas (-ro and -ra options) correspond to the actual addresses of the internal ROM and internal RAM. Use this option when single-chip mode is used.

- User-specified-area check specification (-check_rora)

[Format]

- check_rora

[Parameters]

NONE

[Explanation]

A check is made on whether the specified ROM and RAM areas (-ro and -ra options) exceed the internal ROM and internal RAM.

The following warnings are output when the areas specified for the -ro and -ra options are not within the internal ROM and internal RAM:

W1368L: The area specified for the -ro option is outside the internal-ROM area (area name)
W1369L: The area specified for the -ra option is outside the internal-RAM area (area name)

When the -check_rora option is specified in using single-chip mode, a check is made on whether the specified ROM and RAM areas correspond to the actual addresses of the internal ROM and internal RAM of the product concerned. So, for instance, when a program is ported to a different product, you can check whether the memory map has been changed, by changing only the specified MB number.

Also, when the -check_locate option is specified together with the -check_rora option, you can check whether the program is within the internal ROM and internal RAM.
6.2 Details of Linker Options

[Example]

```
flnk907s -cpu MB90754 -check_rora -ro ROM = 0x00080800/0x000fffff
       -ra RAM1=0x00010000/0x00008fff, RAM2=0x00080000/0x000807ff ...

W1368L: The area specified for the `-ro` option is outside the internal-ROM area (ROM)
W1369L: The area specified for the `-ra` option is outside the internal-RAM area (RAM)
-> The underlined areas specified are not within the ranges of the internal ROM and
   internal RAM; so the warnings are output.
```

```
flnk907s -cpu MB90754 -check_rora -ro ROM = 0x000A0000/0x000fffff
       -ra RAM1=0x00010000/0x00005fff, RAM2=0x00080000/0x000807ff ...

-> The areas specified for the `-ro` and `-ra` options are within the ranges of the internal
   ROM and internal RAM; so no warning is output.
```

**Note:**
1. A warning is output when the `-check_rora` option is specified even if warning output
   suppression (`-w 0`) is specified.
2. The `-check_rora` option is only valid when creating absolute format load modules.
   The `-check_rora` option is ignored when creating relocatable load modules.
3. The `-check_rora` option uses internal-ROM data and internal-RAM data contained in the
   CPU data file. So, no warning is output when the CPU data file contains no data
   concerned. Specify the correct MB number for the `-cpu` option.
6.2.32 User-specified-area check suppression specification (-Xcheck_rora)

The -Xcheck_rora option suppresses the check on the specified ROM area and the specified RAM area (-ro and -ra options) and on the internal-ROM area and the internal-RAM area. Use the -Xcheck_rora option when you want to cancel the -check_rora option.

- User-specified-area check suppression specification (-Xcheck_rora)
  
  [Format]
  
  - Xcheck_rora

  [Parameters]
  
  NONE

  [Explanation]
  
  The -Xcheck_rora option suppresses the check on the specified ROM area and the specified RAM area (-ro and -ra options) and on the addresses of the internal ROM and the internal RAM.

  Use the -Xcheck_rora option when you want to cancel the -check_rora option.

  [Example]
  
  flnk907s -cpu MB90754 -check_rora -ro ROM = 0x00080800/0x000fffff
  -ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...

  W1368L: The area specified for the -ro option is outside the internal-ROM area (ROM)
  W1369L: The area specified for the -ra option is outside the internal-RAM area (RAM)

  -> The underlined areas specified are not within the ranges of the internal ROM and internal RAM; so the warnings are output.

  flnk907s -cpu MB90754 -check_rora -Xcheck_rora -ro ROM = 0x00080800/0x000fffff
  -ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...

  -> Although the underlined areas specified are not within the ranges of the internal ROM and internal RAM, no warning is output because the check is suppressed using the -Xcheck_rora option.
### 6.2.33 Section-placed-area check specification (-check_locate)

A check is made to see that no section is specified outside the memory area. The `-check_locate` option checks the section-placed address, based on the specified ROM area and the specified RAM area (-ro and -ra options) or based on the CPU data file fs internal ROM data and internal RAM data; and outputs a warning if a section is specified outside the memory area.

#### Section-placed-area check specification (-check_locate)

**[Format]**

- `-check_locate`

**[Parameters]**

- `NONE`

**[Explanation]**

The `-check_locate` option checks the section-placed address, based on the specified ROM area and the specified RAM area (-ro and -ra options) or based on the CPU data file fs internal ROM data and internal RAM data; and outputs a warning if a section is specified outside the memory area.

- `W1370L`: The section is placed outside the ROM area (section name)
- `W1371L`: The section is placed outside the RAM area (section name)
- `W1372L`: The section is placed outside the RAM area or the I/O area (section name)
- `W1373L`: The section is placed outside the I/O area (section name)

The section types and the check areas are shown below.
In single-chip mode, when the `-check_locate` option is specified together with the `-check_rora` option, you can check whether the program is placed outside the internal-memory area.

Also, even not in single-chip mode, when the `-check_locate` option is specified together with the ROM area specification and RAM area specification (`-ro` and `-ra` options), you can check whether the program is placed outside the memory area.

<table>
<thead>
<tr>
<th>Section type</th>
<th>Area to be checked Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE</td>
<td>The <code>-check_rora</code> option is specified.</td>
</tr>
<tr>
<td>CONST</td>
<td></td>
</tr>
<tr>
<td>DIRCONST</td>
<td></td>
</tr>
<tr>
<td>ROM/RAM Send-from section</td>
<td>A warning is output when a section is placed outside the area specified for <code>-ro</code> and outside the internal-ROM area.</td>
</tr>
<tr>
<td>SECTION that should be placed within the ROM area</td>
<td></td>
</tr>
<tr>
<td>STACK</td>
<td>A warning is output when a section is placed outside the area specified for <code>-ra</code>.</td>
</tr>
<tr>
<td>DIR</td>
<td></td>
</tr>
<tr>
<td>ROM/RAM Send-to section</td>
<td>A warning is output when a section is placed outside the area specified for <code>-ra</code> and outside the internal-RAM area.</td>
</tr>
<tr>
<td>SECTION that should be placed within the RAM area or the I/O area</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>A warning is output when a section is placed outside the area specified for <code>-ra</code>.</td>
</tr>
<tr>
<td>IO</td>
<td>A warning is output when a section is placed outside the internal-I/O area.</td>
</tr>
</tbody>
</table>
When DATA_A, DATA_B, DATA_C (these are of section type DATA), CODE_D, CODE_E, CODE_F (these are of section type CODE), and STACK_G (this is of section type STACK) are placed in the following memory map, the following check will be made:

```
flnk907s -cpu MB90754 -check_locate -ro ROM = 0x000A0000/0x000fffff
-ra RAM1=0x00001000/0x00005fff, RAM2=0x00080000/0x000807ff ...
```

W1372L: The section is placed outside the ROM area or the I/O area (DATA_A).
W1371L: The section is placed outside the RAM area (STACK_G).
W1372L: The section is placed outside the ROM area or the I/O area (DATA_C).
W1370L: The section is placed outside the ROM area (CODE_D).
W1370L: The section is placed outside the ROM area (CODE_F).

DATA_A is placed within the internal-I/O area; so no warning is output.
DATA_B is placed within the area specified for -ra; so no warning is output.
DATA_C is placed outside the area specified for -ra; so the warning is output.
CODE_D is placed outside the area specified for -ro; so the warning is output.
CODE_E is placed within the area specified for -ro; so no warning is output.
CODE_F is placed outside the area specified for -ro; so the warning is output.
STACK_G is placed outside the area specified for -ra; so the warning is output.

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

The -Xcheck_locate option suppresses the section-placed-area check. Use the -Xcheck_locate option when you want to cancel the -check_locate option.

- Section-placed-area check suppression specification (-Xcheck_locate)

[Format]

- Xcheck_locate

[Parameters]

NONE

[Explanation]

The -Xcheck_locate option suppresses the section-placed-area check. Use the -Xcheck_locate option when you want to cancel the -check_locate option.
6.2.35 Check User-unspecified Section (-check_section)

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

- Check User-unspecified Section (-check_section)

[Format]
- check_section

[Parameters]
NONE

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

If this option is specified, the linker will output the following warning against the section not specified in the specify section layout (-sc) option.

W1375L: This is the section (section name) for which layout is not specified in the -sc option.

If this option is not specified, the linker makes no check.

[Example]
The following operation is performed when the objects having the CODE, INIT, CONST, DATA and STACK sections are linked.

flnk907s -cpu MB90754 -check_section
    -ro ROM=0x000a0000/0x000fffff
    -ra RAM1=0x00001000/0x00008fff,RAM2=0x00008000/0x0000807ff
    -sc CODE+CONST=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2...

W1375L: This is the section (INIT) for which layout is not specified in the -sc option.

->Layout is not specified for the INIT section.
    Since the -check_section option is specified, the warning is output to the INIT section.
flnk907s -cpu MB90754 -check_section
    -ro ROM=0x000a0000/0x000fffff
    -ra RAM1=0x00001000/0x00080000, RAM2=0x00080000/0x000807ff
    -gr ROMGROUP=CODE+CONST -sc DATA=RAM1
    -sc STACK=RAM2 object1 object2...

W1375L: This is the section (CODE) for which layout is not specified in the -sc option.
W1375L: This is the section (INIT) for which layout is not specified in the -sc option.
W1375L: This is the section (CONST) for which layout is not specified in the -sc option.

-> The CODE and CONST sections are grouped in the -gr option.
Since layout is not specified for these sections in the -sc option, however, the warning is output to these sections.

flnk907s -cpu MB90754 -check_section
    -ro ROM=0x000a0000/0x000fffff
    -ra RAM1=0x00001000/0x00080000, RAM2=0x00080000/0x000807ff
    -sc *=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2...

-> Since the section name is specified by the wildcards, layout is specified for all the sections except the DATA and STACK sections.
Since layout is specified for all the sections, no warning is output.

[Notes]
1. No check is made on the sections for which the layout address is specified in the source program.
2. If layout is not specified for the sections in the -sc option even when section group is specified in the -gr option, the warning is output.
3. If the wildcards are used to specify the section name in the -sc option, layout is specified for all the sections that match the wildcards.
   Therefore, no warning is output to the sections that match the wildcards.
   Take care when using the wildcards to specify the section name.
Inhibit check for presence of user-unspecified section (-Xcheck_section)

Inhibit the check for the presence of a section for which the user does not specify layout. This option is used to cancel the -check_section option.

- Inhibit check for presence of user-unspecified section (-Xcheck_section)

  [Format]

  - Xcheck_section

  [Parameters]

  NONE

  [Explanation]

  Inhibit the check for the presence of a section for which the user does not specify layout. This option is used to cancel the -check_section option.

  This option is default.

  [Example]

  The following operation is performed when the objects having the CODE, INIT, CONST, DATA and STACK sections are linked.

  flnk907s -cpu MB90754 -check_section -Xcheck_section
  -ro ROM=0x000a0000/0x000fffff
  -ra RAM1=0x00010000/0x00008000, RAM2=0x00080000/0x000807ff
  -sc CODE+CONST=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2 ...

  ->Layout is not specified for the INIT section.

  Since the check is inhibited in the Xcheck_section option, no warning is output.

  flnk907s -cpu MB90754
  -ro ROM=0x000a0000/0x000fffff
  -ra RAM1=0x00010000/0x00008000, RAM2=0x00080000/0x000807ff
  -sc CODE+CONST=ROM -sc DATA=RAM1 -sc STACK=RAM2 object1 object2 ...

  ->Layout is not specified for the INIT section.

  Since no check is specified in the -check_section option, however, no warning is output. This is the default operation of the linker.
CHAPTER 6 LINKER OPTIONS

6.2.37 Specification of Standby Mode Transition Instruction (-check_SCF)

The linker always checks Standby Mode Transition Instruction.

- Specification of Standby Mode Transition Instruction (-check_SCF)

[Format]
- check_SCF

[Parameters]
None

[Explanation]
When this option is specified, the linker always checks Standby Mode Transition Instruction. With the default operation of the linker, the linker checks the Standby Mode Transition Instruction only when the MB number specified at startup is that of the product corresponding to the Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition. However, if this option is specified, the linker always checks the Standby Mode Transition Instruction regardless of whether it does correspond to the Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition or not.

[Example]
1. When the -check_SCF option is specified:
   flnk907s -cpu MB90F462 main.obj standby.obj -check_SCF
   *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90001 standby.asm:61)
   *** W1379L: The standby mode transition instructions which is not suitable was found by byte check (mov io, #imm8, 00F90003 standby.asm:101)
   *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90007 standby.asm:122)
   *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F9000D standby.asm:183)
   *** W1379L: The standby mode transition instructions which is not suitable was found by byte check (mov io, #imm8, 00F9002F standby.asm:211)
   *** 10312L: Total warning message of S.C.F check:5
   MB90F462 does not correspond to Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition, but by specifying the -check_SCF option, the linker checks Standby Mode Transition Instruction.

2. When the -check_SCF option is not specified:
   flnk907s -cpu MB90F462 main.obj standby.obj
   MB90F462 does not correspond to Cautions for Access to Low Power Consumption Mode
Control Register (LPMCR) for Standby Mode Transition, so the linker does not check the Standby Mode Transition Instruction.
CHAPTER 6 LINKER OPTIONS

6.2.38 Specification of Prohibiting Standby Mode Transition Instruction Check (-Xcheck_SCF)

The linker does not check Standby Mode Transition Instruction.

- Specification of Prohibiting Standby Mode Transition Instruction Check (-Xcheck_SCF)

  [Format]

  - Xcheck_SCF

  [Parameters]

  None

  [Explanation]

  When this option is specified, the linker does not check Standby Mode Transition Instruction. With the default operation of the Linker, the Linker checks the Standby Mode Transition Instruction when the MB number specified at startup is that of the product corresponding to the Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition. However, if this option is specified, the linker does not check the Standby Mode Transition Instruction regardless of whether it does correspond to the Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition or not.

  [Example]

  1. When the -Xcheck_SCF option is specified:

     flnk907s -cpu MB90F387 main.obj standby.obj -Xcheck_SCF

     MB90F387 corresponds to Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition, but by specifying the -Xcheck_SCF option, the linker does not check Standby Mode Transition Instruction.

  2. When the -Xcheck_SCF option is not specified:

     flnk907s -cpu MB90F387 main.obj standby.obj

     *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90001 standby.asm:61)

     *** W1379L: The standby mode transition instructions which is not suitable was found by byte check (mov io, #imm8, 00F90003 standby.asm:101)

     *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90007 standby.asm:122)

     *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90007 standby.asm:183)

     *** W1379L: The standby mode transition instructions which is not suitable was found by byte check (mov io, #imm8, 00F90002F standby.asm:211)

     *** 0312L: Total warning message of S.C.F check:5

     MB90F387 corresponds to Cautions for Access to Low Power Consumption Mode Control
Register (LPMCR) for Standby Mode Transition, so the linker checks the Standby Mode Transition Instruction.
6.2.39 Specification of Fetch Unit Checks (-Wf)

The linker only checks in fetch units when checking Standby Mode Transition Instruction.

- Specification of Fetch Unit Checks (-Wf)

[Format]

- Wf

[Parameters]
None

[Explanation]
When this option is specified, the linker only checks in fetch units when checking Standby Mode Transition Instruction.

With the default operation of the linker, the linker checks using both fetch units and byte units when checking Standby Mode Transition Instruction. However, if this option is specified, the linker checks in only fetch unit.

For details about how to check Standby Mode Transition Instruction, refer to F2MC-16LX family Standby Mode Transition Instruction Check Tool Manual.

[Example]

1. When the -Wf option is specified:

   flnk907s -cpu MB90F387 main.obj standby.obj -Wf

   *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90001 standby.asm:61)

   *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90007 standby.asm:122)

   *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F9000D standby.asm:183)

   *** I0312L: Total warning message of S.C.F check:3

   By specifying the -Wf option, the linker checks in only fetch unit and does not check in byte unit.

2. When the -Wf option is not specified:

   flnk907s -cpu MB90F387 main.obj standby.obj

   *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90001 standby.asm:61)

   *** W1379L: The standby mode transition instructions which is not suitable was found by byte check (mov io, #imm8, 00F90003 standby.asm:101)

   *** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F90007 standby.asm:122)
6.2 Details of Linker Options

*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (mov io, #imm8, 00F9000D standby.asm:183)

*** W1379L: The standby mode transition instructions which is not suitable was found by byte check (mov io, #imm8, 00F9002F standby.asm:211)

*** I0312L: Total warning message of S.C.F check: 5

Checks both in fetch unit and byte unit.
6.2.40 Specifyed Output Load Module for Register Argument Passing (-rp)

The -rp option is an option specified when the function-call interface links objects for register argument passing.
When neither -rp nor -Xrp are specified, objects are linked assuming that the function-call interface outputs load modules for stack argument passing.

- Specifyed Output Load Module for Register argument Passing (-rp)

[Format]

- rp

[Parameters]
NONE

[Explanation]
The -rp option is an option specified when the function-call interface links objects for register argument passing.
Load modules that are output are ones for register argument passing.

[Example]
flnk907s -cpu mb90704 mod1.obj mod2.obj mod3.obj -rp

Note 1:
In the following cases, the linker outputs an error (***E4313L: The module is different calling interface of a function (file name)). If the error is outputted, specify the -rp option or modify the function-call interface of the input object.
- When, with the -rp option specified, an attempt is made to link objects for stack argument passing.
- When, with the -Xrp option specified, an attempt is made to link objects for stack argument passing.

Note 2:
Be sure to use "_rp" for file names of the C library used when linking object for register argument passing.
6.2.41 Specified Output Load Module for Stack Argument Passing (-Xrp)

The -Xrp option is an option specified when the function-call interface links objects for stack argument passing. When neither -rp nor -Xrp are specified, objects are linked assuming that the function-call interface outputs load modules for stack argument passing.

- Specified Output Load Module for Stack argument Passing (-Xrp)

  [Format]

  - Xrp

  (default)

  [Parameters]

  NONE

  [Explanation]

  The -Xrp option is an option specified when the function-call interface links objects for stack argument passing.

  Load modules that are outputted are ones for stack argument passing.

  When the -Xrp option is specified, the -rp option previously specified is disabled.

  [Example]

  flnk907s -cpu mb90704 mod1.obj mod2.obj mod3.obj -rp -Xrp

  [Note:]

  In the following cases, the linker outputs an error (**E4313L: The module is different calling interface of a function (file name)). If the error is outputted, specify the -rp option or modify the function-call interface of the input object.

  • When, with the -rp option specified, an attempt is made to link objects for register argument passing.

  • When, with the -Xrp option specified, an attempt is made to link objects for stack argument passing.
6.2.42 Specifying that Check on the Total Size of the DIR-attribute Sections (-check_dirsize)

The -check_dirsize option is an option specified when checking the total size of the DIR-attribute sections. With the -check_dirsize option specified, when the total size of the DIR-attribute sections exceeds 256 bytes, the linker outputs a warning.

- Specifying that Check on the Total Size of the DIR-attribute Sections (-check_dirsize)
  
  [Format]

  - check_dirsize

  (default)

  [Parameters]

  NONE

  [Explanation]

  The -check_dirsize option is an option specified when checking the total size of the DIR-attribute sections.

  With the -check_dirsize option specified, when the total size of the DIR-attribute sections exceeds 256 bytes, the linker outputs a warning (**W1377L Total size of the DIR attribute sections exceeded 256 bytes).

  If the warning is outputted, modify the source so that the total size of the DIR-attribute sections becomes 256 bytes or less.

  [Example]

  flnk907s -cpu mb90704 mod1.obj mod2.obj mod3.obj -check_dirsize

  [Note:]

  In the following cases, no problem occurs even when the total size of the DIR-attribute sections exceeds 256 bytes: Ignore the warning message.

  • When an assembler source uses two or more direct pages
  • When each task of Real Time OS uses a different direct page

  In a case where no problem occurs even when the total size of the DIR-attribute sections exceeds 256 bytes, the -Xcheck_dirsize option can inhibit check on the total size of the DIR-attribute sections.
6.2.43  Specifying that Check on the Total Size of the DIR-attribute Sections be Inhibited (-Xcheck_dirsize)

The -Xcheck_dirsize option is an option specified when inhibiting check on the total size of the DIR-attribute sections.

- Specifying that Check on the Total Size of the DIR-attribute Sections be inhibited (-Xcheck_dirsize)

  [Format]

  - Xcheck_dirsize

  [Parameters]

  NONE

  [Explanation]

  The -Xcheck_dirsize option is an option specified when inhibiting check on the total size of the DIR-attribute sections.

  When the -Xcheck_dirsize option is specified, the -check_dirsize option previously specified is disabled.

  With the -Xcheck_dirsize option specified, the linker does not output a warning even when the total size of the DIR-attribute sections exceeds 256 bytes.

  [Example]

  flnk907s -check_dirsize -cpu mb90704 mod1.obj mod2.obj mod3.obj -Xcheck_dirsize
CHAPTER 6 LINKER OPTIONS

6.2.44 Relative Assemble List Input Directory Specification (-alin)

The directory in which a relative assemble list file is stored is specified. If this option is not specified, the directory in which the object module is located is specified.

- Relative assemble list input directory specification (-alin)

  [Format]
  `- alin < Path name >`

  [Parameters]
  `<Path name>`
  Directory in which a relative assemble list file is stored

  [Explanation]
  This is an option to be used when an absolute assemble list file should be output.
  The directory in which a relative assemble list file is stored is specified.
  If this option is not specified, the directory in which the object module is located is specified.
  If a relative assemble list file with its path name is specified using the -alf option, the path specified by the -alf option is prioritized.

  [Example]
  `flnk907s *.obj -o ap.abs -alin d:\fmc16 -alf swctr1.lst,mstdef.lst`
  `flnk907s *.obj -o ap.abs -alf d:\fmc16\swctrl.lst,d:\fmc16\mstdef.lst`
  The above two examples share the same meaning.
6.2.45 Absolute Assemble List Output Directory Specification (-alout)

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

- Absolute assemble list output directory specification (-alout)

  [Format]

  - alout < Path name >

  [Parameters]

  <Path name>
  Directory which outputs an absolute assemble list file

  [Explanation]

  The directory which outputs an absolute assemble list file is specified.
  If this option is not specified, the current directory is specified.

  [Example]

  flnk907s *.obj -o ap.abs -alin d:\fmc16 -alf swctrl.lst,mstdef.lst -alout d:\fmc16\als
6.2.46 Absolute Assemble List Output Specification (-als)

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

Absolute assemble list output specification (-als)

[Format]

- als

[Parameters]

None

[Explanation]

All modules are instructed to create absolute assemble lists.
If this option is not specified, absolute assemble lists are not created.
-alsf and -Xals specified before are canceled.

[Example]

flnk907s *.obj -o ap.abs -als
### 6.2.47 Absolute Assemble List Output Module Specification (-alsf)

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

- **Absolute assemble list output module specification (-alsf)**

  **Format**

  ```
  - alsf < Relative assemble list file name > [ , ... ]
  ```

  **Parameters**

  `<Relative assemble list file name>`
  Name of the relative assemble list file which is the source of creating the absolute assemble list.

  **Explanation**

  Modules to create the absolute assemble lists are selected.
  The relative assemble list file names are used to specify the modules. If the extension is omitted, "*.lst" is assumed.
  Any module which is not specified does not create any absolute assemble list.
  This option can be specified in divided multiple parts.
  The previously specified -als and -Xals are canceled.
  Since the store path of a relative assemble list can be specified in the -alin option, the path specification can be omitted when this option is specified.

  **Example**

  ```
  flnk907s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst
  flnk907s *.obj -o ap.abs -alsf swctrl.lst -alsf mstdef.lst
  flnk907s *.obj -o ap.abs -alsf swctrl -alsf mstdef
  ```

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

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

- Absolute assemble list output inhibit specification (-Xals)
  
  [Format]
  
  - Xals ( Default )

  [Parameters]
  
  None

  [Explanation]
  
  The inhibition of creation of absolute assemble lists is instructed for all modules.
  
  This option is a default option and is used to cancel the previously specified -als and -alsf.

  [Example]
  
  flnk907s *.obj -o ap.abs -alsf sectrl.lst,mstdef.lst -Xals
6.2.49 ROM/RAM and ARRAY List Output Specification (-alr)

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

- ROM/RAM and ARRAY list output specification (-alr)

  [Format]

  - alr

  [Parameters]

  None

  [Explanation]

  The ROM/RAM and ARRAY lists are added all absolute assemble lists.

  When using this option, the -als option can be omitted.

  The previously specified -alrf and -Xalr are canceled.

  To output the ROM/RAM and ARRAY lists, specify the debug information output option (-g) when compiling, assembling, and linking.

  [Example]

  flnk907s *.obj -o ap.abs -als -alr -g

  The above two examples share the same meaning.
6.2.50 ROM/RAM and ARRAY List Output Module Specification (-alrf)

Modules to output the ROM/RAM and ARRAY lists are specified. The output of debug information when compiling or assembling is required.

ROM/RAM and ARRAY list output module specification (-alrf)

- alrf <Relative assemble list file name> [ , ... ]

Parameters

<Relative assemble list file name>
Specify the modules to output the ROM/RAM and ARRAY lists using the relative assemble list file names.

Explanation

Modules to output the absolute assemble lists with the added ROM/RAM and ARRAY lists are selected.

Modules are specified using the relative assemble list file names. If the extension is omitted, ".lst" is assumed.

Any module which is not specified does not create any ROM/RAM and ARRAY lists.

If this option is used, the -alsf option can be omitted.

This option can be specified in divided multiple parts.

Previously specified -alr and -Xalr are canceled.

Example

```
flnk907s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst -alrf swctrl.lst,mstdef.lst
flnk907s *.obj -o ap.abs -alsf swctrl.lst -alrf swctrl.lst,mstdef.lst
flnk907s *.obj -o ap.abs -alrf swctrl.lst,mstdef lst
```

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

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

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

  [Format]
  
  - Xalr

  ( Default )

  [Parameters]

  None

  [Explanation]

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

  This option is a default option, therefore, it is not necessary to specify it.

  The previously specified -alr and -alrf are canceled.

  [Example]

  flnk907s *.obj -o ap.abs -als -alr -Xalr

  flnk907s *.obj -o ap.abs -als -Xalr

  The above two examples share the same meaning.
6.2.52 ROM/RAM and ARRAY List Symbol and Address Display Position Specification (-na,-an)

The display positions of the symbols and addresses of the ROM/RAM and ARRAY lists are specified. If -na is specified, symbols (NAME) are output first, then addresses (ADDRESS). If -an is specified, addresses (ADDRESS) are output first, then symbols (NAME).

ROM/RAM and ARRAY list symbol and address display position specification (-na,-an)

[Format]

- na       ( Default )

[Parameters]
None

[Explanation]
Symbols and addresses of the ROM/RAM and ARRAY lists are output in order of NAME and ADDRESS. Symbols are output in alphabetical order.
This option is valid only for modules for which the output of the ROM/RAM and ARRAY lists is specified.
This option is a default option and is specified to cancel the -an option.

[Example]
flnk907s *.obj -o ap.abs -alr -na
flnk907s *.obj -o ap.abs -alr

The above two examples share the same meaning.

[Format]

- an

[Parameters]
None

[Explanation]
Symbols and addresses of the ROM/RAM and ARRAY lists are output in order of ADDRESS and NAME.
Symbols are output in order of address.
This option is used to change the -na option (default).
This option is valid only for modules for which the output of the ROM/RAM and ARRAY lists is specified.

[Example]
flnk907s *.obj -o ap.abs -alr -na -an
flnk907s *.obj -o ap.abs -alr -an

The above two examples share the same meaning.
6.2.53 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]

  flnk907s *.obj -o ap.abs -xl
6.2.54 External Symbol Cross-reference Information List File Name Specification (-xlf)

This option is used to change the output destination directory or file name of the external symbol cross-reference information list file.

- **External symbol cross-reference information list file name specification (-xlf)**

  **[Format]**

  - -xlf <Output file name>

  **[Parameters]**

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

  **[Explanation]**

  The external symbol cross-reference information list file is created with the specified name. When using this option, the -xl option can be omitted.

  If the extension is omitted in the <output file name> specification, the default extension “.mpx” is added.

  If this option is not specified, the absolute format load module file name is used whose extension is changed to “.mpx” as the output file name.

  **[Example]**

  flnk907s *.obj -o ap.abs -xl -xlf ccp903.mpx

  flnk907s *.obj -o ap.abs -xlf ccp903

  The above two examples share the same meaning.
6.2.55 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]

  flnk907s *.obj -o ap.abs -xl -Xxl
  flnk907s *.obj -o ap.abs -xlf
  flnk907s *.obj -o ap.abs

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

The output of the local symbol information list file is specified. The output of debug information when compiling or assembling is required.

[Format]

- sl

[Parameters]
None

[Explanation]
The creation of a local symbol information list file is instructed.
If this option is not specified, no local symbol information list file is created.
To output the local symbol information list file, specify the debug information output option (-g) when compiling, assembling, and linking.

[Example]

flnk907s *.obj -o ap.abs -sl -g
6.2.57 Local Symbol Information List File Name Specification (-slf)

This option is used to change the output destination directory or file name of the local symbol information list file.

- **Local symbol information list file name specification (-slf)**

  [Format]

  ```
  - slf < Output file name >
  ```

  [Parameters]

  - `<Output file name>`

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

  [Explanation]

  The local symbol information list file is created with the specified name.

  When using this option, the -sl option can be omitted.

  If the extension is omitted in the `<output file name>` specification, the default extension “.mps” is added.

  If this option is not specified, the absolute format load module file name is used whose extension is changed to “.mps” as the output file name.

  [Example]

  ```
  flnk907s *.obj -o ap.abs -sl -slf ccp903.mps -g
  ```

  ```
  flnk907s *.obj -o ap.abs -slf ccp903 -g
  ```

  The above two examples share the same meaning.
6.2.58  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]
  
  flnk907s *.obj -o ap.abs -sl -Xsl
  
  flnk907s *.obj -o ap.abs -Xsl
  
  flnk907s *.obj -o ap.abs

  The above three examples share the same meaning.
6.2.59 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]
  
  flnk907s *.obj -o ap.abs -ml
SECTION DETAIL MAP LIST NAME SPECIFICATION (-mlf)

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

Format

Parameters

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

Explanation

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

Example

```
flnk907s *.obj -o ap.abs -ml -mlf ccp903.mpm
flnk907s *.obj -o ap.abs -mlf ccp903
```

The above two examples share the same meaning.
6.2.61 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]
  flnk907s *.obj -o ap.abs -ml -Xml
  flnk907s *.obj -o ap.abs -Xml
  flnk907s *.obj -o ap.abs
  The above two examples share the same meaning.
CHAPTER 7 OUTPUT LIST FILE OF THE LINKER

This chapter explains the formats of each list file output by the linker and how to view the information.

7.1 Types of List Files Output by the Linker
7.2 Link List File
7.3 Absolute Assemble List File
7.4 External Symbol Cross-reference Information List File
7.5 Local Symbol Information List File
7.6 Section Allocation Detailed Information List File
7.1 Types of List Files Output by the Linker

The following five types of list files are output by the linker.

- Link list file
- Absolute format assemble list
- External symbol cross-reference information list
- Local symbol information list
- Section detail map list

Whether to output these files can be selected as options when activating the linker.

- **Link list file**
  The link list file outputs the options for linker activation, input module names, and information about sections and external symbols after module linking.

- **Absolute format assemble list**
  The absolute format assemble list is a list which displays in the absolute format the assemble list output by the assembler in the relative format based on information after module linking.
  This list can be referenced when debugging on the assembler language level, and the addresses in each step of the machine language that are unidentifiable in the link list can be known.

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

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

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

The link list file can be divided into the following 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.

- **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>Option File(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Option file name display area</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) Linker control option display area</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Module(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3) Input module display area</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error(s)</th>
</tr>
</thead>
<tbody>
<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.

Figure 7.2.1a List output format of the control list part
### List display example of the control list part

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

```
F2MC-16 Family SOFTUNE Linker  Control List  2003-08-26 15:18:11  Page: 1

Option File(s)

** no use **

Control(s)

-`g`
-`a`
-`l usrlb.lib`
-`-ro prog=0x8000/0xffff`
-`-ra data=0x0000/0x03ff`
-`-sc P+code=prog,D+data+S=data`

Input Module(s)

1 pca02.obj(pca01)
2 pcasb.obj(pcasb)
3 xccdef.obj(xccdef)

Error(s)

** Nothing **
```

** Figure 7.2.1b List display example of the control list part **
CHAPTER 7  OUTPUT LIST FILE OF THE LINKER

7.2.2 Map List

In the map list, the section names, content types, attributes, and section allocation addresses after linking are displayed.

List output format of the map list part

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

<table>
<thead>
<tr>
<th>S_Addr.</th>
<th>-E_Addr.</th>
<th>Size</th>
<th>Section Type</th>
<th>Al</th>
<th>Sec.(Top**)</th>
</tr>
</thead>
</table>

Page header

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

(1) Map information display area

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

S-Addr: Section start address (hexadecimal)

E-Addr: Section end address (hexadecimal)

Size : Section size (hexadecimal)

Section: Section content type

The section content type is displayed.

- CODE Program section
- DATA Data section
- CONST Data section with initial values
- STACK Stack section
- DIR direct section
- DIRCONST direct section with initial values
- IO IO section

After the section type, the link attribute is displayed.

- P Simple concatenation link
- C Shared link
- A No link

Type : Section attributes

The following attributes are displayed from left.

- R/- Read enabled/disabled
- W/- Write enabled/disabled
- X/- Executable/non-executable
- I/- Initial value Yes/No

Al : Boundary adjustment value for section allocation (hexadecimal)

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

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

(*1) The end address of a section with the section size 0 is displayed by [........].

** Figure 7.2.2a List output format of the map list part

<table>
<thead>
<tr>
<th>List display example of the map list part</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following shows a list display example of the map list part.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S_Addr.</th>
<th>E_Addr.</th>
<th>Size</th>
<th>Section</th>
<th>Type</th>
<th>Al</th>
<th>Sec.(Top 29)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0000001F</td>
<td>00000020</td>
<td>DATA</td>
<td>P</td>
<td>RW -</td>
<td>-</td>
<td>08</td>
</tr>
<tr>
<td>00000020</td>
<td>00000039</td>
<td>0000001A</td>
<td>DATA</td>
<td>P</td>
<td>RW -</td>
<td>-</td>
<td>02</td>
</tr>
<tr>
<td>0000003A</td>
<td>00000053</td>
<td>0000001A</td>
<td>STAK</td>
<td>P</td>
<td>RW -</td>
<td>-</td>
<td>02</td>
</tr>
<tr>
<td>00000054</td>
<td>0000006D</td>
<td>0000001A</td>
<td>DATA</td>
<td>P</td>
<td>RW -</td>
<td>I</td>
<td>02</td>
</tr>
<tr>
<td>00080000</td>
<td>00080039</td>
<td>0000003A</td>
<td>CODE</td>
<td>P</td>
<td>R - X I</td>
<td>02</td>
<td>REL</td>
</tr>
<tr>
<td>00080803A</td>
<td>00080053</td>
<td>0000001A</td>
<td>CODE</td>
<td>P</td>
<td>R - X I</td>
<td>02</td>
<td>REL</td>
</tr>
<tr>
<td>000808054</td>
<td>0008006D</td>
<td>0000001A</td>
<td>DATA</td>
<td>P</td>
<td>R -</td>
<td>-</td>
<td>I</td>
</tr>
</tbody>
</table>

** Figure 7.2.2b List display example of the map list part **
7.2.3 Memory Used Information List

The memory used information list displays the name of the area of the RAM specified area or the ROM specified area and the size of the free space or the remainder range and the header and end address of the specified area.

Memory Used Information List Field List Output Format

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM/RAM Used Information</td>
<td>Display information specified by the -ro or -ra options.</td>
</tr>
<tr>
<td>Area Internal Information</td>
<td>Display 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.</td>
</tr>
<tr>
<td>General Evaluation Value Information</td>
<td></td>
</tr>
</tbody>
</table>

Page Header

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

(1) ROM/RAM Used Information Display Area

- Lines containing a # character at their head among those output to the ROM/RAM Used Information display area indicate the information specified by the -ro or -ra options.

- Lines that do not contain a # character at the head of the lines that were output, display the header address and end address of the sections in the valid area and the size of the area that is used including the gap area and the remainder or insufficient values for the sizes of the used areas.

- Areas specified by the -ro or -ra option, have have the names of their sections that could not be located and the size of the section.

- S_Addr: Start Address (Hex)
- E_Addr: End Address (Hex)
- Size: Area Size (Hex)
- Remain: Size of the area in the memory (Hex)

The following shows the header symbols.

- : Free Area Size
- -: Remainder Area Size
- Space: When free/remainder area is 0.
- Name/State: Memory Area Name and Section Name
(2) Area Internal Information Area

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

- **S_Addr**: Area Start Address (Hex)
- **E_Addr**: Area End Address (Hex)
- **Size**: Area Size (Hex)
- **Name/State**: Area Type
  - **FREE**: Free Area
  - **USED**: Used Area
  - **GAP**: Gap Area

(3) General Evaluation Value Information

General Evaluation Value Information shows the following types of information for the overall ROM and RAM areas.

1. Total value of the specified area (Total)
2. Total value of the used area (Used) (Note: Includes the Gap area.)
3. Total value of the free or remainder area (Remainder)

*Figure 7.2.3a Memory Used Information List Field List Output Format*
CHAPTER 7  OUTPUT LIST FILE OF THE LINKER

Example Display of the Memory Used Information List Field List

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

<table>
<thead>
<tr>
<th>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 00001000</td>
<td>---------</td>
<td>RAM1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001000 - 00001774 00000775 +0000088B</td>
<td></td>
<td>USED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001000 - 00001137 00000137</td>
<td>---------</td>
<td>GAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001136 - 0000152B 000003F4</td>
<td>---------</td>
<td>USED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000152C - 0000152F 00000004</td>
<td>---------</td>
<td>GAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001530 - 00001774 00000245</td>
<td>---------</td>
<td>USED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001775 - 00001FFF 0000088B</td>
<td>---------</td>
<td>FREE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RAM -- Total(00001400) Used(00002194) Remainder(-00000D94) ← All RAM area information

<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>00002000 - 000023FF 00000400</td>
<td>---------</td>
<td>RAM2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00002000 - 00003A1E 00001A1F -0000161F</td>
<td></td>
<td>(+1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>** Not Locate ** 00000B28</td>
<td>data01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>** Not Locate ** 00000EF6</td>
<td>data02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00002000 - 000023FF 00000400</td>
<td>---------</td>
<td>FREE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ROM -- Total(00004000) Used(00002100) Remainder(+00001F00) ← All ROM area information

Note: 1. Memory area information that has sections not located that were to be located in the memory area by the automatic locating option indicates the status with added section size. (When the numerical value expression exceeds 0xFFFFFFFF, the lower part of the 32 bit is displayed of that value.)

2. When Mode 2 of the automatic location option is specified, this displays the memory area specified last for the section not located (for either the ROM area or the RAM area).

3. Sections specified by the user location are included in the memory area which include the section header address.

Figure 7.2.3b Example Display of the Memory Used Information List Field List
7.2.4 Symbol List

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

List output format of the symbol list part

<table>
<thead>
<tr>
<th>Symbol Value</th>
<th>Type</th>
<th>Def.</th>
<th>Symbol Name (Top **)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol Value : Symbol address or symbol value (hexadecimal)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type : Symbol type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One of the following is displayed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Addr. Address level</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EQU EQU defined symbol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit Bit attributes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Def. : Symbol definition</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One of the following is displayed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OM/LM Defined in the input object module or relative format load module.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIB Defined in the linked library</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>user Symbol whose value is temporarily set using the -df option.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol Name: Symbol name</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

(*) If a symbol is referenced, the symbol name is displayed as it is. If it is not referenced, @ is displayed prior to the symbol name.

Figure 7.2.4a List output format of the symbol list part
### List display example of the symbol list part

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

<table>
<thead>
<tr>
<th>Symbol Value</th>
<th>Type</th>
<th>Def.</th>
<th>Symbol Name(Top **)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000004(ABS)</td>
<td>Addr.</td>
<td>OM/LM</td>
<td>Filename</td>
</tr>
<tr>
<td>00000008(ABS)</td>
<td>Addr.</td>
<td>OM/LM</td>
<td>_c1</td>
</tr>
<tr>
<td>000000C4(ABS)</td>
<td>Addr.</td>
<td>OM/LM</td>
<td>_main</td>
</tr>
<tr>
<td>00000018(ABS)</td>
<td>Addr.</td>
<td>OM/LM</td>
<td>_ulval</td>
</tr>
</tbody>
</table>

**Figure 7.2.4b** List display example of the symbol list part
7.3 Absolute Assemble List File

The absolute format assemble list output by the linker consists of the following parts:
- Header
- Information list
- ROM/RAM and ARRAY list
- Assemble source list
- Section information list
- Cross-reference list

We explain here the items output in each list.

- Absolute Assemble List File
  - Header
    Output in the first line of each page.
  - Information list
    The information list output by the assembler is output as it is.
  - ROM/RAM and ARRAY list
    - ROM/RAM list
      Global symbol names allocated in the ROM/RAM area and absolute address information are output.
    - ARRAY list
      Array element names, structure member names, and absolute address information are output.

    If the option-alr is specified, the ROM/RAM and ARRAY lists are displayed. If -Xalr is specified, the ROM/RAM and ARRAY lists are not displayed.
  - Assemble source list
    The assemble source list displays a variety of information about assembling of the source program in units of lines. Error information, locations, object code are displayed.
  - Section information list
    Names and attributes defined in the source program are displayed.
  - Cross-reference list
    Definitions of the symbol names used in the source program and references are displayed using the line numbers.
Figure 7.3 shows the structure of the list.

<table>
<thead>
<tr>
<th>Information list</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ROM/RAM list)</td>
</tr>
<tr>
<td>(ARRAY list)</td>
</tr>
<tr>
<td>Assemble source list</td>
</tr>
<tr>
<td>Section information list</td>
</tr>
<tr>
<td>Cross-reference list</td>
</tr>
</tbody>
</table>

- alr specification

**Figure 7.3  Assemble list structure**

■ Error messages in the assemble list

If errors which occurred when assembling are in the assemble list, the error messages are displayed as they are.
7.3 Absolute Assemble List File

7.3.1 Header and Information List

The header consists of four lines.
On the first page of the list, the information list (settings when activating the assembler, number of errors, number of warnings, source file names, include file names, option file names) is displayed after the header.

■ Header format

The header consists of four lines. It is displayed at the head of each page. The first and second lines have the same format for all assemble lists, and the third line depends on each source program. The following shows the header format.

[1st and 2nd lines]

F2MC-16 Family SOFTUNE Linker  VxxLxx  YYYY-MM-DD  hh:mm:ss  Page: xxxx

| Tool name | Absolute assemble list creation date | Page number |

[3rd line]

- Each list name - (Module name)

The contents of the original assemble list are displayed as they are.

[4th line]

A blank line is output.

Figure 7.3.1 Header format
7.3.2 ROM/RAM and ARRAY Lists

The ROM/RAM and ARRAY lists show information about the symbols described in the C source programs.
To output the ROM/RAM and ARRAY list, the output specification of debug information when compiling or assembling is required.

ROM/RAM and ARRAY lists

The following shows the ROM/RAM and ARRAY list format.

**ROM/RAM list format**

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>VALUE</th>
<th>KIND</th>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>--- name ---</td>
<td>xxxxxxxx</td>
<td>xxxxxxxx</td>
<td>x</td>
<td>xxx</td>
</tr>
</tbody>
</table>

**ARRAY list format**

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>--- name ---</td>
<td>xxxxxxxx</td>
<td>xxxxxxxx</td>
</tr>
</tbody>
</table>

*1: Symbol name

The names of symbols described in programs are displayed.
In the case of an ARRAY list, all elements of the arrays and structures are displayed.

*2: Address

The absolute address is displayed using a hexadecimal value.
The display positions of the symbol name and address can be changed as options. If -na is specified, the symbol name is first displayed and then the address is displayed. If -an is specified, the address is first displayed and then the symbol name is displayed.

For details, see “6.2.52 ROM/RAM and ARRAY List Symbol and Address Display Position Specification (-na,-an)”.

*3: Value

If an initial value is assigned to the symbol, the value is displayed as a decimal number.

*4: Symbol type

The following symbol types are available.

L : Variable value
F : Function name
T : Tag name reference
*5: Memory allocation

The memory area to which a symbol is allocated is displayed.

ROM: ROM area
RAM: RAM area

Figure 7.3.2a ROM/RAM and ARRAY lists format

[Example 1]

- ROM/RAM LISTING - (sample1)

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>VALUE</th>
<th>KIND</th>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Line</td>
<td>000002B8</td>
<td>00000100</td>
<td>L</td>
<td>ROM</td>
</tr>
<tr>
<td>_Nameleng</td>
<td>0000004C</td>
<td>0000016</td>
<td>L</td>
<td>RAM</td>
</tr>
<tr>
<td>_symref</td>
<td>00005592</td>
<td>T</td>
<td>ROM</td>
<td></td>
</tr>
<tr>
<td>_Tflag</td>
<td>00000763</td>
<td>0000002</td>
<td>L</td>
<td>RAM</td>
</tr>
</tbody>
</table>

Figure 7.3.2b Example of ROM/RAM lists

[Example 2]

- ARRAY LISTING - (sample1)

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>_symref[0].val</td>
<td>00005592</td>
<td>00000100</td>
</tr>
<tr>
<td>_symref[0].nam</td>
<td>00005596</td>
<td>00000002</td>
</tr>
<tr>
<td>_symref[0].atr</td>
<td>00005598</td>
<td>00000016</td>
</tr>
<tr>
<td>_symref[1].val</td>
<td>0000559A</td>
<td>00012000</td>
</tr>
<tr>
<td>_symref[1].nam</td>
<td>0000559E</td>
<td>00000002</td>
</tr>
<tr>
<td>_symref[1].atr</td>
<td>000055A0</td>
<td>00000016</td>
</tr>
<tr>
<td>_Xpcr[0][0]</td>
<td>00001066</td>
<td>00000400</td>
</tr>
<tr>
<td>_Xpcr[0][1]</td>
<td>000010A0</td>
<td>00000120</td>
</tr>
</tbody>
</table>

Figure 7.3.2c Example of ARRAY lists
7.3.3 Assemble Source List

The assemble source list is displayed with the location part using the absolute address and the object code part using the determined code after linking.

Assemble source list

The following shows the assemble source list format.

<table>
<thead>
<tr>
<th>SN</th>
<th>LOC</th>
<th>OBJ</th>
<th>LLINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>xxxxxx</td>
<td>xxxxxxxxxxxxxxxxxxxxx</td>
<td>x</td>
<td>xxxxxxxx</td>
</tr>
</tbody>
</table>

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

*1:Section acronym
The first two characters of a section name are displayed.

*2:Location
The 32-bit location value is displayed as a hexadecimal value.

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

*4:Object code type
The attributes of values contained in object code are displayed in the following order of priority.

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

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

*5:Line number
The line number is displayed as a decimal 5-digit number.
7.3 Absolute Assemble List File

*6: Progress display of the preprocessor and optimization code check

**Preprocessor**

X : Line for assembling
& : Macro expansion line

*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.

**Figure 7.3.3a  Assemble source list format**

[Example]

```
<table>
<thead>
<tr>
<th>SN</th>
<th>LOC</th>
<th>OBJ</th>
<th>LLINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>002CE724</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>002CE724</td>
<td>[2]</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>002CE726</td>
<td>0020</td>
<td>0010</td>
<td></td>
</tr>
<tr>
<td>CO</td>
<td>0000A280</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CO</td>
<td>0000A280</td>
<td>AC043C</td>
<td>1025</td>
<td>.DATA.B 2, 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.DATA.H 32,16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.SECTION CODE, CODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MOVW RW4,#LS1</td>
</tr>
</tbody>
</table>
```

**Figure 7.3.3b Example of Assemble source list**
7.3.4 Section Information List

The section information list shows information about the sections defined in a program.
The section information list is displayed on a new page after page break.

- Section information list

The following shows the section information list format.

<table>
<thead>
<tr>
<th>No.</th>
<th>Section name</th>
<th>Size</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Section name</td>
<td>xxx</td>
<td>xx</td>
</tr>
<tr>
<td>2</td>
<td>Section name</td>
<td>xxx</td>
<td>xxx xxx</td>
</tr>
<tr>
<td>3</td>
<td>Section name</td>
<td>xxx</td>
<td>xxx x xxx</td>
</tr>
<tr>
<td>4</td>
<td>Section name</td>
<td>xxx</td>
<td>xxx=xxx</td>
</tr>
</tbody>
</table>

*1: Section occurrence number
   Starts with 0. This number corresponds to the section number in an object file.

*2: Section name
   The defined section names are displayed in order of occurrence.

*3: Section size
   The section size is displayed as a 32-bit hexadecimal value.

*4: Section type
   The type of the sections is displayed. The following types are available.
   - CODE  : Code section
   - DATA  : Data section
   - CONST : Data section with initial values
   - COMMON: Shared section
   - STACK : Stack section
   - DIR   : Direct section
   - DIRCONST: Direct section with initial values
   - IO    : IO section

*5: Section allocation format
   The display of a relative assemble list is output as it is.
   The ALIGN value is displayed for a relative section when assembling. The LOCATE value is displayed for an absolute section.

Figure 7.3.4a Section information list format
### Figure 7.3.4b Example of Section Information List

<table>
<thead>
<tr>
<th>NO</th>
<th>SECTION-NAME</th>
<th>SIZE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DATA</td>
<td>00000004</td>
<td>DATA REL ALIGN=4</td>
</tr>
<tr>
<td>1</td>
<td>INIT</td>
<td>0000001C</td>
<td>DATA REL ALIGN=4</td>
</tr>
<tr>
<td>2</td>
<td>CONST</td>
<td>00000020</td>
<td>CONST REL ALIGN=4</td>
</tr>
<tr>
<td>3</td>
<td>CODE</td>
<td>00000038</td>
<td>CODE REL ALIGN=2</td>
</tr>
</tbody>
</table>
CHAPTER 7  OUTPUT LIST FILE OF THE LINKER

7.3.5 Cross-reference List

The cross-reference list shows information about the names described in a program and the relations between their definition and reference. The cross-reference list is displayed on a new page after page break.

Cross-reference list

The following shows the cross-reference list format.

```
F2MC-16 Family SOFTUNE Linker VxxLxx YYYY-MM-DD hh:mm:ss Page: XXXX

- CROSSREFERENCE LISTING - (Module name)

<table>
<thead>
<tr>
<th>NAME</th>
<th>ATTRIB.</th>
<th>VALUE</th>
<th>DEFINITION/REFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>------ name -------</td>
<td>xxx/xxxx</td>
<td>xxxxxxxx</td>
<td>xx xxx xxx</td>
</tr>
<tr>
<td>*1</td>
<td>*2</td>
<td>*3</td>
<td>*4</td>
</tr>
</tbody>
</table>
```

*1: Name
The symbol names and section names are displayed in order of upper-case letter, lower-case letter and alphabet.

*2: Symbol type
Symbol types are displayed in the following formats.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>Absolute symbol</td>
</tr>
<tr>
<td>REL</td>
<td>Relative symbol</td>
</tr>
<tr>
<td>ABS/EXP</td>
<td>Absolute symbol (external definition)</td>
</tr>
<tr>
<td>REL/EXP</td>
<td>Relative symbol (external definition)</td>
</tr>
<tr>
<td>IMP</td>
<td>External reference symbol</td>
</tr>
<tr>
<td>SECT/ABS</td>
<td>Absolute section</td>
</tr>
<tr>
<td>SECT/REL</td>
<td>Relative section</td>
</tr>
<tr>
<td>UNDEFINED</td>
<td>Undefined symbol</td>
</tr>
<tr>
<td>REGLIST</td>
<td>Register symbol</td>
</tr>
</tbody>
</table>

*3: Value
If the symbol has a value, the value is displayed with a 32-bit absolute address.

*4: definition and reference line number
The line in which the symbol is defined and the lines which are referenced are displayed.
The sharp symbol "#" is added to the end of the line number in which the symbol is defined.

Figure 7.3.5a Cross-reference list format
### 7.3 Absolute Assemble List File

*Example*

<table>
<thead>
<tr>
<th>NAME</th>
<th>ATTRIB.</th>
<th>VALUE</th>
<th>DEFINITION/REFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRSY</td>
<td>SECT/REL</td>
<td>000021C2</td>
<td>266 #</td>
</tr>
<tr>
<td>BINCOL</td>
<td>REL/EXP</td>
<td>00000308</td>
<td>40 # 92 13</td>
</tr>
<tr>
<td>DATAC</td>
<td>REL</td>
<td>00006D58</td>
<td>437 #</td>
</tr>
<tr>
<td>IROAS</td>
<td>ABS</td>
<td>00000101</td>
<td>79 # 10 4</td>
</tr>
</tbody>
</table>

**Figure 7.3.5b Example of Cross-reference list**
CHAPTER 7 OUTPUT LIST FILE OF THE LINKER

7.4 External Symbol Cross-reference Information List File

The symbol cross-reference information list file displays the external definition symbols of each object module after linking and cross-reference information between modules of the external reference symbols.

■ External symbol cross-reference information list file

The following shows the output format of the external symbol cross-reference information list.

<table>
<thead>
<tr>
<th>External Symbol Cross Reference List</th>
<th>YYYY-MM-DD</th>
<th>hh:mm:ss</th>
<th>Page: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Module01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Module02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. Module15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External Symbol Cross Reference List</th>
<th>YYYY-MM-DD</th>
<th>hh:mm:ss</th>
<th>Page: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>--- symbol ---</td>
<td>--- type/value ---</td>
<td>--- module (No.) ---</td>
<td></td>
</tr>
<tr>
<td>extsym1</td>
<td>Addr. 0x000012E8</td>
<td>1# 2 3 4 5 6 8 11 12 14</td>
<td></td>
</tr>
<tr>
<td>extsym2</td>
<td>Addr. 0x0000C3F34</td>
<td>2 3 4# 5 6 8 11 12 14</td>
<td></td>
</tr>
<tr>
<td>extsym3</td>
<td>Addr. 0x000012E6</td>
<td>1# 2 3 4 5 6 8 11 12 13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 15</td>
<td></td>
</tr>
<tr>
<td>operator new(unsigned int)</td>
<td>Addr. 0x0000AAA</td>
<td>1 2 3 4 7# 8</td>
<td></td>
</tr>
<tr>
<td>extsymunresolved</td>
<td>???? 0x00000000</td>
<td>10 15</td>
<td></td>
</tr>
<tr>
<td>n1p</td>
<td>EQU 0x00000001</td>
<td>4 5 6 11 12 14#</td>
<td></td>
</tr>
<tr>
<td>main</td>
<td>Addr. 0x00000314</td>
<td>3# 5</td>
<td></td>
</tr>
</tbody>
</table>

Module(s)

The serial number starting with 1 is added to indicate the module name.

Symbol

The symbol names are displayed. (by default, up to 20 characters)

Type/value

The following types are available.

- Addr.: Address
- EQU : EQU symbol
- Bit : Bit symbol
- ????: Undefined

"value" indicates a value. In the case of a bit symbol, its bit position is displayed in parentheses.

Module (No.)

Modules which are defined/referenced are displayed using their Nos. The # symbol indicates a defined module.

Figure 7.4 External symbol cross-reference information list format
### 7.5 Local Symbol Information List File

The local symbol information list file displays information about the variables and functions which include local symbols for each module constituting an absolute format load module. Since this list is created based on debug information when compiling and assembling, it is necessary to specify the `-g` option.

#### Local symbol information list file

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

| Local Symbol List                           YYYY-MM-DD  hh:mm:ss   Page:    1 |
|---------------------------------------------|-----------------------------|
| Module(s)                                   |                             |
| 1. Module01                                 |                             |
| 2. Module02                                 |                             |
| 15. Module15                                |                             |

| Local Symbol List                           YYYY-MM-DD  hh:mm:ss   Page:    2 |
|---------------------------------------------|-----------------------------|
| === Module No.1(module01) ===               |                             |
| --- symbol ---                             | --- Kind ---               |
| func1(int)                                  | Func. g                    |
| localstatic1                                | Var. s                     |
|                                            | 0x000C3F34                 |
|                                            | 0x000012EE                 |
| === Module No.15(module015) ===             |                             |
| --- symbol ---                             | --- Kind ---               |
| Atable                                      | loc. s                     |
| Extsym                                      | Var. g                     |
|                                            | 0xFFFFFFFC                 |
|                                            | 0x00001342                 |

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

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

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

Up to 20 characters of the symbol name are displayed in one line of the list.
CHAPTER 7 OUTPUT LIST FILE OF THE LINKER

○ Kind
The following symbol types are displayed.

- Var. :Variable (C)
- Func. :Function (C)
- loc. :Local (C)
- Addr. :Address (ASM)
- EQU :EQU symbol (ASM)
- bit :bit symbol (ASM)
- ???? :Undefined
- s :static (C)
- g :global (C)

“value” indicates a value.

○ val
The value of a symbol is indicated.

For loc., an offset value is displayed.

(*) No detailed information about structures (member names) nor the typedef definitions are displayed.

Figure 7.5 Local symbol information list format
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.

<table>
<thead>
<tr>
<th>S.Addr. - E.Addr.</th>
<th>Size</th>
<th>Section</th>
<th>Type</th>
<th>Al</th>
<th>M.No.</th>
<th>Sec.(Top 18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000 - 00000048</td>
<td>00000049</td>
<td>DATA</td>
<td>P</td>
<td>RW-</td>
<td>04</td>
<td>1</td>
</tr>
<tr>
<td>000004A - 000003E1</td>
<td>00000348</td>
<td>DATA</td>
<td>P</td>
<td>RW-</td>
<td>04</td>
<td>2</td>
</tr>
<tr>
<td>000003E4 - .........</td>
<td>00000000</td>
<td>DATA</td>
<td>P</td>
<td>RW-</td>
<td>04</td>
<td>3</td>
</tr>
<tr>
<td>000003E4 - 00000643</td>
<td>00000260</td>
<td>DATA</td>
<td>P</td>
<td>RW-I</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>00000644 - 00000743</td>
<td>00000100</td>
<td>STACK</td>
<td>P</td>
<td>RW-</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>0000C0000 - 0000C0441</td>
<td>00000442</td>
<td>CODE</td>
<td>P</td>
<td>R - X</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>0000C0442 - 0000C148B</td>
<td>0000104A</td>
<td>CODE</td>
<td>P</td>
<td>R - X</td>
<td>02</td>
<td>3</td>
</tr>
<tr>
<td>* 000C148C- 000C201D</td>
<td>00000B92</td>
<td>CODE</td>
<td>P</td>
<td>R - X</td>
<td>02</td>
<td>4</td>
</tr>
<tr>
<td>* 000C2000 - 000C2203</td>
<td>00000204</td>
<td>CODE</td>
<td>P</td>
<td>R - X</td>
<td>02</td>
<td>5</td>
</tr>
<tr>
<td>000C3000 - 000C325F</td>
<td>00000260</td>
<td>DATA</td>
<td>P</td>
<td>R - I</td>
<td>04</td>
<td>4</td>
</tr>
</tbody>
</table>

- Module(s)

The serial number starting with 1 is added to indicate the module name.

- Module(s)

The serial number starting with 1 is added to indicate the module name.

- S.Addr.-E.Addr.

Section's start and end addresses.

A section with overlapping addresses has (*) prior to its section name.

If the size of a section is 0, the end address of it is displayed as “........”.

- Size

A section in which allowed address space is overflowed is displayed with the maximum size + 1.
Section

The following section content types are available.
- CODE : Program section
- DATA : Data section
- CONST : Data section with initial values
- STACK : Stack section
- DIR : Direct section
- DIRCONST : Direct section with initial values
- IO : IO section

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

AI

The boundary adjustment value for section allocation is displayed as hexadecimal values.

M.No.

The module numbers are displayed. Module numbers displayed in Module(s) are displayed.

Sec.(Top xx)

The section names are displayed. xx of (Top xx) has the number of characters that allows the section name to be displayed in one line.

# prior to a section name indicates that data with initial values to be transferred to RAM before execution is allocated in the section.

Figure 7.6 Section Allocation Detailed Information List format
CHAPTER 8    LINKER RESTRICTIONS AND Q&A

This chapter explains about linker restrictions and Q&A for use.

8.1 Linker Restrictions
8.2 Q&A for Using the Linker
8.1 Linker Restrictions

There is no particular restriction on the number of input files and sections that can be processed when using the linker. However there are some restrictions when using the wild card in the input files and library files.

- **Linker restriction**
  
  The linker has no restriction on the number of input files, sections, or symbols that can be processed. If it becomes necessary to register the section names or symbols names for internal processing of the linker, memory is dynamically allocated. And processing is possible until all memory that can be used by the linker is used up.

  If it becomes impossible to continue processing, a message to report insufficient memory is issued and then the linker processing ends.

- **Linker reservation symbol**
  
  In the linker, symbols with the "_ROM_section name", "_RAM_section name" are created automatically for each section using the ROM > RAM transfer function.

  Therefore, if a symbol of the same type is defined in a user program, "W1327L:This symbol is already defined (symbol name)" is generated. The user must not define symbols with the "_ROM_section name" and "_RAM_section name".

- **Restrictions on the object file format**
  
  Restrictions of the linker are listed below. There are following restrictions on the numbers of symbols and others that can be managed in object files.

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

**Note:** The files here mean object files. When using relative format load module files and incremental link files, count them as the total number of input object files when creating them.
## 8.2 Q&A for Using the Linker

Section 8.2 shows the questions and answers on using a linker.

### Using the wild card

<table>
<thead>
<tr>
<th>Q.</th>
<th>There are a large number of input object module files. Can the wild card still be used?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>If you use the wild card for specifying the input file on the command line, the linker expands and executes it. You can specify the input file name in an option file, and the wild card can also be used here. Refer to the following example for using the wild card.</td>
</tr>
</tbody>
</table>

**Example.**
flnk907s *.obj -o outfile.abs  
flnk907s mactrl.obj xz???.obj

<table>
<thead>
<tr>
<th>Q.</th>
<th>The wild card can be used when specifying section allocation, but how can I use the wild card?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>The wild card may be useful when many sections with the same content type should be unified or when programs are created using many section names. It may become necessary to decide characters to become the keywords when naming the section names, considering the use of the wild card.</td>
</tr>
</tbody>
</table>

**Example.**
Section names are defined with the names like DTdata1, DTdata2, DTdata3, DTdata4 ....for the sections whose content type is data, and the names like CDprog1, CDprog2, CDprog3, CDprog4.... for the sections whose content type is code.  
In this case, the following specification method can be selected. (Only the -sc option part is shown)
- `-sc DTdata1+DTdata2+DTdata3+DTdata4=0x1000, CDprog1+CDprog2+CDprog3+CDprog4=0X3000`
- `-sc DT*=0x1000, CD*=0x3000`
- `-sc */data=0x1000, */code=0x3000`
## Handling variables with initial values

<table>
<thead>
<tr>
<th>Q.</th>
<th>When developing embedded programs using C compilers, variables with initial values are created. Since these variables are rewritten during execution of programs, they must be on RAM during execution. Tell us the procedure for creating programs and precautions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>In embedded programs, variables with initial values are on ROM first and they must be on RAM when they are referenced. Therefore, programs become inoperable if the reference address in programs is not set to RAM and a mechanism to transfer initial data from ROM to RAM before application execution is not implemented.</td>
</tr>
</tbody>
</table>
|    | - If the memory model is small or medium  
Variables with initial values generated by the F<sup>2</sup>MC-16 family C compilers are gathered in the INIT section (used for execution) and DCONST section (initial data is stored).  
Therefore, 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.  
The user must write a program to transfer initial value data from DCONST to INIT.  
Example a shows “Program example for transferring initial value data”. |
|    | - If the memory model is compact or large  
Variables with initial values generated by the F<sup>2</sup>MC-16 family C compilers are generated for each module. They are gathered in the “INIT_module name” section to be used for execution and in the “DCONST_module name” section (initial data is stored). The information is set in the DTRANS section.  
Therefore, 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.  
The user must write a program to transfer initial value data based on information set in DTRANS.  
Example b shows “Program example for transferring initial value data”. |

| Example a | [Program example for transferring initial value data: The memory model is either small or medium]  
[section DCONST, const ...... (1)  
DCONST_S  
[section INIT, data ...... (2)  
INIT_S  
.section start, code  
MOV A, #bnksym (DCONST_S)  
MOV ADB, A  
MOV A, #bnksym (INIT_S)  
MOV DTB, A  
MOVW RW0, #sizeof (INIT)  
MOVW A, #INIT_S  
MOVW A, #DCONST_S  
MOVSI DTB, ADB  
(1) DCONST_S is a symbol to indicate the start address of the DCONST section (source) on ROM.  
(2) INIT_S is a symbol to indicate the start address of the INIT section (destination) on RAM.  
These symbols must be defined at the start of the sections, and so the program to transfer initial value data must be linked first. |
### Example

| [Program example for transferring initial value data: The memory model is either compact or large] | .section DTRANS, const  |
| DTRANS_S | .section start, code |
| [MOV A, #bnksym (DTRANS_S)] | MOV DTB, A |
| MOVW RW1, #DTRANS_S | BRA LABEL2 |
| LABEL1: | MOVW A, @RW1+6 |
| MOV USB, A | MOV A, @RW1+2 |
| MOVW A, @RW1+4 | MOVW A, @RW1 |
| MOVW RW0, @RW1+8 | MOVSI SPB, ADB |
| MOVN A, #10 | MOVN A, #10 |
| ADDW RW1, A | ADDW RW1, A |
| LABEL2: | MOVW A, RW1 |
| SUBW A, #DTRANS_S | CMPW A, #sizeof (DTRANS) |
| CMPW A, #sizeof (DTRANS) | BNE LABEL1 |

(1) DTRANS_S is a symbol to indicate the start address of the DTRANS section (transfer information storing table) on ROM. These symbols must be defined at the start of the sections, and so the program to transfer initial value data must be linked first.
Part 3 describes the specifications, options, and output lists of a librarian.

CHAPTER 9 SPECIFICATIONS OF A LIBRARIAN
CHAPTER 10 OPTIONS A LIBRARIAN
CHAPTER 11 LIST FORMATS OF A LIBRARIAN
CHAPTER 12 RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN
CHAPTER 9 SPECIFICATIONS OF A LIBRARIAN

CHAPTER 9 describes the functions and the function types of a librarian. A librarian is a tool used to create a library file.

9.1 Functions of a Librarian
9.2 Function Types of a Librarian
9.3 Creating and Editing a Library File
9.4 Extracting a Module from a Library File
9.5 Deleting Debugging Information of a Library
9.6 Checking and Displaying the Contents of a Library File
9.7 Mixing of Objects for a Librarian
9.8 Mixing of Objects with Different Function-call Interfaces in Librarian
9.1 Functions of a Librarian

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

 Roles of a librarian

To develop a program, divide a source program for each function into modules, each of which you then compile and assemble.

A linker then combines the compiled and assembled modules into one to create the target program.

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

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

Registering the frequently used modules in a batch to create a general-purpose library file enables them to be easily used in other programs. A good example of this is a library in C.

Since a librarian allows you to add, delete, or replace modules of a library file, you can keep them up to date.

![Diagram of the roles of a librarian](image)

**Figure 9.1 Roles 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.
CHAPTER 9 SPECIFICATIONS OF A LIBRARIAN

9.3 Creating and Editing a Library File

Object modules (more than one) that an assembler has output may be united and registered as a library file.
A module may be added to, deleted from, or replaced with the one in an existing library file.

- Creating a new library file
  Object modules (more than one) that an assembler has output may be united and registered as a library file (See Figure 9.3a).

- Editing a library file
  A module may be added to, deleted from, or replaced with the one in an existing library file.
    - Adding a module
      A module may be added to an existing library file (See Figure 9.3b).
9.3 Creating and Editing a Library File

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

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

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

- Extracting a module from a library file
  A module may be extracted from a library file and put back into an object module file (See Figure 9.4).

Figure 9.4 Extracting a module
9.5 Deleting Debugging Information of a Library

An object module registered with a debugging information block in a library may be re-registered without it.

- Deleting debugging information

An object module registered with a debugging information block in a library may be re-registered without it (See Figure 9.5).

Figure 9.5 Deleting debugging information
9.6 Checking and Displaying the Contents of a Library File

The following two items are checked.

• Whether an unsolved external reference symbol exists in a library
• Whether a module with debugging information exists

Additionally, this function provides you with information such as the date and time of creating and updating a library file and registering a module and the names of external definition symbols defined in each module.

■ Checking the contents of a library file

The following two items may be checked.

❖ Whether an unsolved external reference symbol exists in a library

A linker, when a module taken in from a library contains an external reference symbol, first searches for a defined symbol in the same library file.

Therefore, it is recommended that whenever an external reference symbol exists in a module in a library file, the module containing the concerned external definition symbol should be registered in the same library file.

The program checks the correspondence between external reference and defined symbols in a library file. If any undefined reference symbols remains, it outputs a diagnostic message.

❖ Whether a module with debugging information exists

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

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

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

■ Displaying the contents of a library file

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

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

For the contents of display, see CHAPTER 11, “LIST FORMATS OF A LIBRARIAN”.
9.7  Mixing of Objects for a Librarian

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

■ Mixing of objects

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

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

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

Table 9.7a   List of CPUs for which objects may be mixed with the -cpu option

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

☺  Same target specified. This is not mixing.

☺  Mixing allowed. However, the linker outputs information if 2 is specified in the -w option.

×  Mixing not allowed. The linker outputs an error.
9.8 Mixing of Objects with Different Function-call Interfaces in Librarian

Mixture of objects in the librarian that have different function-call interfaces is explained here.

- Mixing of objects with different function-call interfaces in Librarian
  Mixture of objects in the linker that have different function-call interfaces
  F2MC-16 Family supports the following function-call interfaces:

  - Stack argument passing (this is the existing function-call interface)
  - Register argument passing (RW0,RW1)

  Between objects for register argument passing and objects for stack argument passing, the way of passing arguments and the way of using registers are different. So, when objects having different function-call interfaces are mixed and used, programs do not operate normally.

  In the linker, objects for stack argument passing and objects for register argument passing must not be mixed even if the target CPU is the same for these two types of objects.

  If an object having a different function-call interface is mixed in the linker, the linker outputs the following error and stops processing:

  (** E4410U : The module is different calling interface of a function (fire name).)

  When objects have the same function call interface, an object mixture check is performed in the target CPU as usual. For object mixture check in the target CPU, see Section “9.7 Mixing of Objects for a Librarian”.

CHAPTER 10 OPTIONS A LIBRARIAN

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 10.1a List of options for a librarian

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options for creating and editing a library</td>
<td>Adding (registering) a module</td>
<td>-a</td>
</tr>
<tr>
<td></td>
<td>Replacing (registering) a module</td>
<td>-r</td>
</tr>
<tr>
<td></td>
<td>Deleting a module</td>
<td>-d</td>
</tr>
<tr>
<td></td>
<td>Extracting a module</td>
<td>-x</td>
</tr>
<tr>
<td>Options for outputting a list</td>
<td>Specifying to output a list file</td>
<td>-m</td>
</tr>
<tr>
<td></td>
<td>Specifying not to output a list file</td>
<td>-Xm Default</td>
</tr>
<tr>
<td></td>
<td>Specifying to output detailed information of a list file</td>
<td>-dt s, d, r, a</td>
</tr>
<tr>
<td></td>
<td>Specifying the number of lines per page of a list</td>
<td>-pl Default: 60</td>
</tr>
<tr>
<td></td>
<td>Specifying the number of columns per line of a list</td>
<td>-pw Default: 70</td>
</tr>
<tr>
<td>Options for searching and protecting a file</td>
<td>Creating a backup file</td>
<td>-b</td>
</tr>
<tr>
<td></td>
<td>Inhibiting the creation of a backup file</td>
<td>-Xb Default</td>
</tr>
<tr>
<td></td>
<td>Checking the contents of a library file</td>
<td>-c</td>
</tr>
<tr>
<td></td>
<td>Optimizing the contents of a file</td>
<td>-O</td>
</tr>
<tr>
<td>Other options</td>
<td>Specifying to output debugging information</td>
<td>-g</td>
</tr>
<tr>
<td></td>
<td>Specifying not to output debugging information</td>
<td>-Xg</td>
</tr>
<tr>
<td></td>
<td>Specifying a target CPU</td>
<td>-cpu need</td>
</tr>
<tr>
<td></td>
<td>Specified output library for register passing</td>
<td>-rp</td>
</tr>
<tr>
<td></td>
<td>Specified output library for stack passing</td>
<td>-Xrp Default</td>
</tr>
</tbody>
</table>
### Table 10.1a List of options for a librarian (Continued)

<table>
<thead>
<tr>
<th>Common options</th>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifying not to read a default option file</td>
<td>-Xdof</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifying to read an option file</td>
<td>-f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifying to display help messages</td>
<td>-help</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifying to output the version and messages</td>
<td>-V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifying not to output the version and messages</td>
<td>-XV</td>
<td></td>
<td>Default</td>
</tr>
<tr>
<td>Specifying to display a termination message</td>
<td>-cmsg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifying not to display a termination message</td>
<td>-Xcmsg</td>
<td></td>
<td>Default</td>
</tr>
<tr>
<td>Specifying to set the termination code to 1 when a warning occurs</td>
<td>-cwno</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifying to set the termination code to 0 when a warning occurs</td>
<td>-Xcwno</td>
<td></td>
<td>Default</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 Sections 10.2.1, “Adding (Registering) a Module (-a)” through 10.2.4, “Extracting a Module (-x)”.

- Options for outputting a list
  Details of the options for outputting a list are described in Sections 10.2.5, “Specifying to Output a List File (-m)” through 10.2.9, “Specifying the Number of Columns Per Line of a List (-pw)”.

- Options for searching and protecting a file
  Details of the options for searching and protecting a file are described in Sections 10.2.10, “Creating a Backup File (-b)” through 10.2.13, “Optimizing the Contents of a File (-O)”.

- Other options
  Details of other options are described in Sections 10.2.14, “Specifying to Output Debugging Information (-g)” through 10.2.19, “Specifyed Output Library for Stack Argument Passing (-Xrp)”.
10.2.1 Adding (Registering) a Module (-a)

Use the -a option to create a new library file or to add a module to an existing library file.

- Adding (registering) a module(-a)

**[Syntax]**

- a < Object module file name > [ , ... ]

**[Parameter]**

- <Object module file name>
  Object module file name that the assembler has output

**[Description]**

Specify a file name for the module to be registered in a library file.

- If the file name has no extension specified, the “.obj” extension is assigned.
- If an already registered module has the same name as the one to be registered, an error message is output and the latter module is not registered.
- If an external definition symbol with the same name exists, a module is not registered either.

To specify <Object module file name>, you may use a wild card.

**[Example 1]**

flib907s 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]**

flib907s syslib -a "mod*.obj" -a chksw

The object module files with the extension .obj and the first three characters as mod in the current directory and chksw.obj are registered.

**Note:** Up to 65535 modules may be registered.

When using a wildcard, you cannot specify <Object module file name> separated with commas. Specify them in multiple -a options as shown in the following example.

- If a wildcard is specified, you are not supplied with an extension. Be sure to specify an extension.

For information on the expansion of a wildcard in a file name which depends on the OS, see Appendix G, “Differences of Specifications Depending on the OS”.
10.2.2 Replacing (Registering) a Module (-r)

A module in an existing library file is replaced with a new module with the same name.

**Replacing (registering) a module (-r)**

**[Syntax]**

```
-r <Object module file name> [ , ... ]
```

**[Parameter]**

- `<Object module file name>`
  - An object module file name that the assembler has output

**[Description]**

If a module in the library file being edited has the same name as the one in the specified file, the former module is replaced. Otherwise, the specified module is registered.

If the file name has no extension specified, the “.obj” extension is assigned.

To specify `<Object module file name>`, you may use a wild card.

**[Example 1]**

```
flib907s s  syslib.obj -r loadx.obj,loady.obj
```

The two modules in loadx.obj and loady.obj replaces those with the same names in the library file being edited.

If no module with the same name exists in the library file, the modules are added to the library file which is then re-registered.

**[Example 2]**

```
flib907s  syslib.lib -r "load?.obj"
```

The object module files with the extension .obj and the first four characters as load followed by one arbitrary character in the current directory are replaced.

**Note:** When using a wildcard, you cannot specify `<Object module file name>` separated with commas. Specify them in multiple -r options.

If a wildcard is specified, you are not supplied with an extension. Be sure to specify an extension.

For information on the expansion of a wildcard in a file name which depends on the OS, see Appendix G, “Differences of Specifications Depending on the OS”.
10.2.3 Deleting a Module (-d)

An unnecessary module is deleted from a library file.

- Deleting a module (-d)

  [Syntax]
  
  \[- d \ < Module name > \ [ , \ ... \ ]\]

  [Parameter]
  
  <Module name>
  Name of a module to be deleted

  [Description]
  The specified module is deleted from a library file.
  Be sure to specify a module name, not a file name.

  [Example]
  
  flib907s  syslib.lib -d inchar,outchar
  
  Two modules, inchar and outchar are deleted from syslib.lib.

  **Note:** Although you may specify a module name using a pseudo instruction, use the
  same name for an object module file and a module unless absolutely necessary.
  Using different names for them will cause an error when editing a library.
  To specify the same name for a file and a module, use a name consisting only of
  alphanumeric characters and underscores.
  To check the module name, use the list output option (-m) to refer to the module
  name that is output in the beginning of a list file.
10.2.4 Extracting a Module (-x)

A module is extracted from a library file and put back into an object module file before registration.

- Extracting a module (-x)

  [Syntax]

  - x < Module name > [, < Object module file name > ]

  [Parameters]

  <Module name>
  Name of module to be extracted

  <Object module file name>
  Name of output file of extracted module

  [Description]

  The specified module is extracted from a library file.
  The extracted module becomes the same object module file before registration.
  If the <Object module file name> is not specified, a file is created with the <Module name> followed by the “.obj” extension.

  [Example]

  flib907s  syslib -x add
  flib907s  syslib.lib -x add,add.obj
  The module, add is extracted from an existing library file and the add.obj file is created.
  flib907s  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.
  flib907s  syslib -x add -x add,addfunc.obj
10.2.5 Specifying to Output a List File (-m)

Module names and external definition symbol names registered in a library file are output as the information list.

■ Specifying to output a list file (-m)

[Syntax]

```
-m { <List file name> | - }
```

[Parameter]

- `<List file name>`
  Specify the file name of the librarian list to be output.
  Specify a hyphen (-) to output the list in the standard output.

[Description]

Module names and external definition symbol names registered in a library file are output as the information list.

If the `<List file name>` has no extension specified, “.mp2” is added to it.

This option allows you to output only the information of module names registered. To display more detailed information, use the -dt option described later.

The list contents show the status when the librarian is terminated.

If no other option related to editing is provided, the contents of the specified library file is listed.

If you want to check the contents of a library file on the screen without storing it in a list, specify a hyphen in the parameter.

[Example 1]

```
flib907s syslib.lib -m libx.mp2
```

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

[Example 2]

```
flib907s syslib -a obj1.obj2 -m libx.lis
```

The contents of syslib.lib created after the obj1.obj and obj2.obj modules are added are output to libx.lis.

[Example 3]

```
flib907s syslib -m -
```

The module name list registered in syslib.lib is output to the standard output.
10.2.6 Specifying not to Output a List File (-Xm)

This specification inhibits a librarian from outputting a list file.

- Specifying not to output a list file (-Xm)

  [Syntax]

  - Xm

  [Parameter]

  None

  [Description]

  This specification inhibits the output of a list file.
  Specifying the -Xm option after the -m option disables the -m option.

  [Example]

  flib907s syslib.lib -m libx.mp2 -Xm

  The list file is not created.
10.2.7 Specifying to Output Detailed Information of a List File (-dt)

The -m option specifies outputting the list but only displays the list of registered module names.
Use the -dt option to obtain information on sections and external symbols for each module registered in a library, or to obtain information on external definitions and external reference symbols for the entire library.

[Syntax]

```
  - dt <Information type> [ , <Information type> ] ...
```

[Parameter]

```
<Information type>
  s : Outputs a section name and its size for each module.
  d : Outputs external definition symbols for each module.
  r : Outputs external reference symbols for each module.
  a : Outputs for the entire library external definition symbols and external reference symbols yet unsolved in the library.
```

[Description]

If this option is not specified, only the registered module names are output in a list file. This option is used to obtain more detailed information.

The <Information type> must always be specified.

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

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

[Example 1]

```
flib907s syslib.lib -m libx.mp2 -dt r,s
```

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

[Example 2]

```
flib907s 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).

Syntax:

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

Parameter:

<Number of lines>
Specify 0 or between 20 and 255 inclusive.

Description:
Specify the number of lines to be printed per page of a list file.
Specifying 0 disables the page control when the list file is output.
If the -m option is not specified, this option is invalid.

Example 1:
flib907s syslib.lib -m libx.mp2 -pl 40
The number of lines per page of a list is 40.

Example 2:
flib907s syslib.lib -m -dt s -pl 0
A list with section information added is output to the standard output without page ejection.
10.2.9 Specifying the Number of Columns Per Line of a List (-pw)

Use this option to change the number of columns per line of a list from its default (70 characters).

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

[Syntax]

```
-pw <Number of columns>  ( Default: 70 )
```

[Parameter]

<Number of columns>
Specify between 70 and 1023 inclusive.

[Description]
Specify the number of columns to be printed per line of a list file.
Use this option when the default number of columns causes a long symbol name, section name, or module name to extend over two lines, making it difficult to comprehend.

At the default number of columns (70), the number of characters displayable per line are as follows:

- Module name : 22 characters
- Section name : 20 characters
- Symbol name : 29 characters

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

[Example]

```
flib907s syslib.lib -m libx.mp2 -pw 80
```

The number of columns per line of a list is changed to 80.

In this case, the number of characters displayable per line for each name are as follows:

- Module name : 32 characters
- Section name : 30 characters
- Symbol name : 34 characters

**Note:** In a list of a librarian, specifying the number of columns per line in the -pw option changes the number of characters displayed per line for the module name, section name, and symbol name. Since one line of a symbol name is separated on the screen into two fields on the left and right, specifying twice the number of characters in the longest symbol name plus 12 will display an easy-to-understand list.
10.2.10 Creating a Backup File (-b)

Editing a library file causes its contents to be lost. Use the -b option to store a backup of the library file before editing it.

<table>
<thead>
<tr>
<th>Creating a backup file (-b)</th>
</tr>
</thead>
</table>

**[Syntax]**

- b

**[Parameter]**

None

**[Description]**

When a librarian edits a library file by adding or deleting a module, the original contents of the file are changed and lost.

Use this option to create a backup file of the original.

The backup file has the ".bak" extension.

A backup is created for only one generation of a library file. If you edit an important library file, you must create a backup for yourself before using a librarian.

**[Example]**

flib907s  syslib -a putc.obj, getc.obj -b

syslib.lib before editing is stored as syslib.bak after editing.

To syslib.lib after editing, putc.obj and getc.obj are added.
10.2.11 Inhibiting the Creation of a Backup File (-Xb)

Use the -Xb option to cancel the -b option used to obtain a backup.

Inhibiting the creation of a backup file (-Xb)

[Syntax]

- Xb  (Default)

[Parameter]

None

[Description]

By default, a librarian does not create a backup for the library file to be edited. This is the same as specifying the -Xb option.

Specify this option to nullify the -b option when it is specified.

[Example]

All of the following three specifications result in the same processing.

flib907s  syslib    -a putc.obj,getc.obj
flib907s  syslib    -a putc.obj,getc.obj -Xb
flib907s  syslib -b -a putc.obj,getc.obj -Xb

syslib.lib before editing is deleted after editing.

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

Use this option to briefly check the contents of a library file. The following two items are checked.

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

  [Syntax]

  - c

  [Parameter]

  None

  [Description]

  The following two items are checked.

  1. Whether an unsolved external reference symbol exists in the library

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

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

  2. Whether a module with debugging information exists

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

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

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

  [Example]

  flib907s syslib.lib -c

  The contents of syslib.lib are checked.

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

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

- Optimizing the contents of a file (-O)

  [Syntax]

  \[ -O \]

  [Parameter]
  None

  [Description]
  An object module registered with a debugging information block in a library is re-registered without it.

  Since debugging information occupies a very large part of an object module file, you can make a library file significantly smaller by deleting the debugging information.

  [Example]

  \[ flib907s syslib -O \]

  Debugging information is deleted from the syslib.lib file.

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

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

Specifying to output debugging information (-g)

[Syntax]

- g

[Parameter]
None

[Description]
A librarian usually removes any debugging information that may be contained in an object module before registering it in a library file. Use this option to register the specified object without any changes regardless of whether or not it has debugging information.

To delete debugging information after creating a library, use the optimization option-O to recreate a library file.

[Example]

flib907s syslib.lib -a inchar,outchar -g

inchar.obj,outchar.obj is registered with debugging information unremoved, if any are contained, in a library file.
10.2.15 Specifying not to Output Debugging Information (-Xg)

Use the -Xg option to nullify the -g option used to specify not to delete debugging information.

Specifying not to output debugging information (-Xg)

[Syntax]

- Xg                           ( Default )

[Parameter]

None

[Description]

A librarian usually removes any debugging information that may be contained in an object module before registering it in a library file. This is the same as specifying this -Xg option.

Specify this option to nullify the specification of the -g option.

You may use the optimization option-O to delete debugging information in a batch after creating a library.

[Example]

The following three specifications result in the same processing.

flib907s syslib.lib -a inchar,outchar
flib907s syslib.lib -a inchar,outchar -Xg
flib907s syslib.lib -g -a inchar,outchar -Xg

Debugging information contained in inchar.obj,outchar.obj is not registered in a library file.
10.2.16 Specifying a Target CPU (-cpu)

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

### Specifying a Target CPU (-cpu)

**[Syntax]**

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

**[Parameter]**

- `<MB number>`
  - MB number of a target CPU

**[Description]**

A target CPU of the programs to be combined into a library file is specified using an MB number.

**[Example]**

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

**Note:** To create a library, you must specify a target CPU using this option. This option may not be omitted.
10.2.17 Specifying CPU Information File (-cif)

The CPU information file used by librarian is specified.

CPU Information File Specification (-cif)

[Format]
- cif < CPU information file name >

[Parameter]
< CPU information file name >
The CPU information file used by linker is specified.

[Explanation]
The CPU information file used by linker is specified.

[Example]
flib907s syslib.lib -a inchar, outchar -cpu MB90704
-cif C:\Softune\lib\907\cpu_info\MB90704.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\907\907.csv

When installing the compiler assembler packs in a different directory and using the compiler, assembler and linkage kit instead of SOFTUNE workbench, specify -cif so that each tool can reference the same CPU information file.
10.2.18 Specified Output Library for Register Argument Passing (-rp)

The -rp option is an option specified when the function call interface creates/edits libraries whose arguments are passed via registers. When neither -rp nor -Xrp are specified, libraries are created/edited assuming that the function-call interface creates/edits libraries for stack argument passing.

- Specified Output Library for Register Argument Passing (-rp)

  [Format]
  
  - rp

  [Parameter]
  
  NONE

  [Explanation]
  
  The -rp option is an option specified when the function-call interface creates/edits libraries for register argument passing.

  Specify the -rp option in the following cases:
  - When registering objects for register argument passing, in libraries
  - When editing the library in which objects for register argument passing are registered

  [Example]
  
  flib907s sample.lib -cpu mb90704 -a mod1.obj mod2.obj -rp

  Note 1:
  
  In the following cases, the librarian outputs an error (**E4410U The module is different calling interface of a function (file name)). If the error occurs, specify the -rp option or modify the function-call interface of the input object or of the library to be edited.

  - When, with the -rp option specified, an attempt is made to register objects for stack argument passing
  - When, with the -rp option specified, an attempt is made to edit the library where objects for stack argument passing
  - When, with the -Xrp option specified, an attempt is made to register objects for register argument passing
  - When, with the -Xrp option specified, an attempt is made to edit the library where objects for register argument passing

  Note 2:
  
  A mixture of objects for register argument passing and objects stack argument passing cannot be linked.

  So, we recommend that you perform the following when creating/managing libraries:

  Specify library file names that can be identified by the function-call interface of the registered objects.

  (Example) Library for stack argument passing : sample_stk.lib
  Library for register argument passing : sample_rg.lib
10.2.19 Specified Output Library for Stack Argument Passing
(-Xrp)

The -Xrp option is an option specified when the function-call interface creates/edits libraries for stack arguments passing. When neither -rp nor -Xrp are specified, libraries are created/edited assuming that the function-call interface creates/edits libraries for stack argument passing.

- Specified Output Library for Stack Argument Passing (-Xrp)

[Format]

- Xrp

(default)

[Parameter]
NONE

[Explanation]
The -Xrp option is an option specified when the function-call interface creates/edits libraries for stack argument passing.

When the -Xrp option is specified, the -rp option previously specified is disabled.

Specify the -Xrp option in the following cases:

- When registering objects for stack argument passing, in libraries
- When editing the library in which objects for stack argument passing are registered

[Example]
flib907s sample.lib -cpu mb90704 -a mod1.obj mod2.obj -rp -Xrp

Note:
In the following cases, the librarian outputs an error (**E4410U The module is different calling interface of a function (file name)**). If the error occurs, specify the -rp option or modify the function-call interface of the input object or of the library to be edited.

- When, with the -rp option specified, an attempt is made to register objects for stack argument passing
- When, with the -rp option specified, an attempt is made to edit the library where objects for stack argument passing
- When, with the -Xrp option specified, an attempt is made to register objects for register argument passing
- When, with the -Xrp option specified, an attempt is made to edit the library where objects for register argument passing
- When, with the -rp option specified, an attempt is made to edit the library where objects whose arguments are passed via stacks are registered
- When, with the -Xrp option specified, an attempt is made to register objects whose arguments are passed via registers
- When, with the -Xrp option specified, an attempt is made to edit the library where objects whose arguments are passed via registers are registered
CHAPTER 11 LIST FORMATS OF A LIBRARIAN

CHAPTER 11 describes the configuration of a list file of a librarian.

11.1 Contents of Information in a List File
11.2 List of Module Names
11.3 Detailed Information of a Module
11.4 External Defined and Reference Symbol Information in a Library
11.1 Contents of Information in a List File

In a list file of a librarian, the contents of a library file is output in the following five groups
- Module name
- Section information of each module
- External definition symbol information for each module
- External reference symbol information for each module
- External defined and reference symbol information for all the modules

To output a list, you must specify the -m option and the -dt option.

■ Configuration of a list file

Figure 11.1 shows the configuration of a list file.

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

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

<table>
<thead>
<tr>
<th></th>
<th>Library File Name</th>
<th>Number of Modules</th>
<th>Number of Symbols</th>
<th>Librarian Name</th>
<th>Library Creation Date</th>
<th>Library Revision Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[ Module Name ]</td>
<td>[ Entry Date ]</td>
<td>[ Creation Date ]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ModuleA</td>
<td>2003-03-01 14:23:50</td>
<td>2003-03-19 10:03:21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ModuleC</td>
<td>2003-03-01 14:23:50</td>
<td>2003-02-23 15:15:00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*1: Library file name
*2: Number of modules registered in a library file (expressed in decimals)
*3: Number of external definition symbols registered in a library file (expressed in decimals)
*4: Librarian name
*5: Date and time when the library file is first created
*6: Date and time when the library file is last updated.....Same as *5 for a new file
*7: Module name (in alphabetical order)
   For a module name, each line displays as many characters as (Page width -48).
   The default (-pw 70) is 22 characters.

[ Entry Date ] Date and time when the module is registered in a library file
[ Creation Date ] Date and time when the module is created

Figure 11.2a Format of a librarian list (default)
11.3 Detailed Information of a Module

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

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

### List output overview

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

<table>
<thead>
<tr>
<th>Library File Name</th>
<th>Number of Modules</th>
<th>Number of Symbols</th>
<th>Librarian Name</th>
<th>Library Creation Date</th>
<th>Library Revision Date</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Entry Date</th>
<th>Creation Date</th>
</tr>
</thead>
</table>

- **1**: Output by the s parameter in the -dt option.
  
  This is the information of sections in a module.
  
  The section name, section attribute, and size are displayed.
  
  For a section name, each line displays as many characters as (Page width -50).

- **2**: Output by the r parameter in the -dt option.
  
  Each line displays two of the external reference symbol names in a module.

- **3**: Output by the d parameter in the -dt option.
  
  For an external symbol name, each line displays as many characters as ((Page width -12)/2).
  
  Each line displays two of the external definition symbol names in a module.

Figure 11.3  Format of a librarian list (when detailed information is specified)
11.4 External Defined and Reference Symbol Information in a Library

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

List output overview

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

<table>
<thead>
<tr>
<th>Library File Name</th>
<th>Number of Modules</th>
<th>Number of Symbols</th>
<th>Librarian Name</th>
<th>Library Creation Date</th>
<th>Library Revision Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample.lib</td>
<td>3</td>
<td>3</td>
<td>F2MC-16 Family SOFTUNE Librarian</td>
<td>2003-09-09 12:58:30</td>
<td>2003-09-09 12:58:30</td>
</tr>
</tbody>
</table>

[ Module Name ] [ Entry Date ] [ Creation Date ]
ModuleA  2003-09-09 12:58:30  2003-09-09 11:57:34
.
.

*1 [ ALL Ext_Def Symbol(s) ]
  chr1get              p_text
  prtext

*2 [ ALL Ext_Ref Symbol(s) ]
  chr_get              tx_len

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

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

*2: Each line displays two of all the external reference symbols without corresponding external definition symbols in the library file.

For an external symbol name, each line displays as many characters as ((Page width - 12) / 2).

Figure 11.4 Format of a librarian list (when detailed information is specified)
CHAPTER 11  LIST FORMATS OF A LIBRARIAN
CHAPTER 12 RESTRICTIONS AND QUESTIONS AND ANSWERS ON A LIBRARIAN

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

Section 12.1 describes restrictions on using a librarian, such as those on the number of modules and external symbols that can be registered in a library file and other precautions.

- **Restrictions on a librarian**
  
  There are two restrictions on using a librarian:
  
  - Number of modules that can be registered in a library file: 65535 maximum
  - Number of external definition symbols that can be registered in a library file: 65535 maximum

- **Precautions on the required disk space**

  Before editing an existing library file and creating a backup file, check that sufficient disk space is available to store the new library file and the existing library.

- **Precautions on specifying options**

  Specify the options for checking the contents of a library file (-c) and optimizing the contents of a file (-O) alone, respectively.

  These options may not be specified together with other options.
# 12.2 Questions and Answers on Using a Librarian

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

- **Questions and answers on creating a library file**

<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.**
- `fasm907s file1` → `file1.obj` is output.
- `fasm907s file2` → `file2.obj` is output.
- `flib907s libfile -a file1.obj,file2.obj`

<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, you are 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.**
- `fasm907s file1 -g` → `file1.obj` with debugging information is output.
- `flib907s libfile -r file1 -g` → `libfile.lib` with debugging information is output.
- `flib907s libfile -O` → `libfile.lib` without debugging information is output.

<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.**
- `flib907s libfile -a "*.obj"` → All the files with the "*.obj" extension are registered.
<table>
<thead>
<tr>
<th>Q.</th>
<th>I forgot what is contained in a library file that I created a while ago. How can I find what are the modules registered in it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Use the <code>-m</code> option to see what is contained in a library file. Specify <code>&quot;-m File-name&quot;</code> to create a list file with the <code>.mp2</code> extension by default. If the contents output by the <code>-m</code> option is not sufficient for you, use the <code>-dt</code> option together to obtain more detailed information.</td>
</tr>
</tbody>
</table>

**Example.**
- `flib907s libfile -m libdoc` → A list file, `libdoc.mp2` is output.
- `flib907s libfile -m libdoc -dt a,s` → A list file with detailed information, `libdoc.mp2` is output.

<table>
<thead>
<tr>
<th>Q.</th>
<th>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?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Specify a hyphen (<code>-</code>) instead of a file name in the <code>-m</code> option to display the contents in the standard output.</td>
</tr>
</tbody>
</table>

**Example**
- `flib907s libfile -m -`
- `flib907s libfile -m - -dt a,s`
- `flib907s libfile -a file3.obj -m -`

<table>
<thead>
<tr>
<th>Q.</th>
<th>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?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Use the <code>-pw</code> option to increase the number of columns to be displayed per line. It is 70 characters by default. If four characters are in the second line, add twice that number, i.e., eight to specify the number of columns as 78. Then the names will fit in one line.</td>
</tr>
</tbody>
</table>

**Example.**
- `flib907s libfile -m libdoc -pw 80`

<table>
<thead>
<tr>
<th>Q.</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>A librarian allows you to create a backup file for one generation (with the <code>.bak</code> extension) using the <code>-b</code> option. Naturally, you are recommended to create a backup of a library before editing it. However, specify the <code>-b</code> option as required.</td>
</tr>
</tbody>
</table>

**Example.**
- `flib907s libfile -r file1, file2 -d mod4 -b`
PART 4  OBJECT FORMAT CONVERTERS

Part 4 describes the types of object format converters, list of options, functions, and conversions of object formats.

CHAPTER 13  SPECIFICATIONS OF AN OBJECT FORMAT CONVERTER
CHAPTER 14  COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER
CHAPTER 15  LOAD MODULE CONVERTER (f2ms, f2hs, f2ls)
CHAPTER 16  FORMAT ADJUSTER (m2ms, h2hs)
CHAPTER 17  THE BINARY CONVERTER
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 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

Load module converter

This converter converts the absolute format load module of the linker output to a general-purpose format.

Figure 13.1a shows input and output of the load module converter.
13.1 Outline of Object Format 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 13.1b shows input and output of the adjuster and binary converter.

- **Load module converter**

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

  Figure 13.1c shows input and output of the converter.

![Diagram of the adjuster and binary converter](image)

**Figure 13.1b Input and output of the adjuster and binary converter**

![Diagram of the load module converter](image)

**Figure 13.1c Input and output of the converter**
13.2 Types of Converters

The command for an object format converter is x2ys, where the x represents the object format of an input file and the y represents the object format of an output file. An alphabetic character assigned in the x and y format means one of the following formats:
- f: Absolute format load module that a linker outputs
- m: 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 13.2a to convert an object format.

#### Table 13.2a 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>

### S Format Adjuster

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

### S Format Binary Converter

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

### Types of other converters

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

#### Table 13.2b 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 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.

```
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 S Format Binary Converter. For details, see Section "16.3.2 Specifying the Output Range (-ran)."

[Example]

```
f2ms sample
```

The absolute format load module that a linker outputs, `sample.abs`, is input and the `sample.mhx` file in the S format is output.
Chapter 14 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 a converter, the following common options may be used. Table 14.1 lists types of common options of a converter.

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

To display a brief explanation of an option, enter the command name alone or use the -help option.

```
x2ys
x2ys -help
```
14.2 Changing an Output File name (-o)

The directory in which to create an output file after conversion and the file name are changed from the default.

■ Changing an Output File name (-o)

[Syntax]

- o < Object file name >

[Parameter]

<Object file name>  Output file name

[Description]

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

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

- Absolute format load module .abs
- 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, the portion of an file containing no data is padded with data of a specified value.

Padding (-p)

[Syntax]

- p < Value >, < Starting address >, < Ending address >

[Parameters]

<Value> One-byte data

<Starting address> Starting address at which to set <Value>

* When the adjust option (-adjust) of f2ms or f2hs is specified, binary converter and adjuster cannot be used.

<Ending address> Ending address at which to set <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]

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 (Absolute Format Load Module into the HEX8 format), f2es (Absolute Format Load Module into the HEX16 format)
15.1 Outline of Load Module Converter

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

Outline of Load Module Converter

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

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

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

As shown in Figure 15.1, by specifying the adjust option (-adjust), f2ms and f2hs can adjust the output file together with adjuster.

![Figure 15.1 Outline of Load Module Converter](image-url)
15.2 List of Options of the Load Module Converter

Options are available to specify Load Module Converter operations in more detail. The following lists the option names and function outlines.

- List of Options of the Load Module Converter

The table below gives the option names and an overview of their functions.

For details on the parameters and functions required for an option, see the description on each option.

Table 15.2a  List of 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 suppression not to read default option file</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying option file name</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 startup message of program</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Suppression not to output version number and startup message of program</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>Suppression 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>
15.3 Details of Load Module Converter Options

This section describes each of the options for the load module converter. Note that options common to the linkage kit are described in CHAPTER 3. Common Options, and options common to the converter are described in CHAPTER 14. Common Options of the Object Converter.

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

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

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

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

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

This option specifies the format used to output data.

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

[Syntax]

- S1
- S2
- S3

[Parameter]
None

[Description]
Specifies the format used to output data.

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

[Precautions]
If the specification with this option conflicts with the output range, this option outputs an error and performs no processing.
The terminator record (S9 record, S8 record, S7 record) for output varies with the specification of this option. (See Table 15.3.1)

Table 15.3.1: Output S Format Specification List

<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></td>
</tr>
<tr>
<td>-S2</td>
<td>0x00000000-0x00FFFFFFFF</td>
<td>S8 record</td>
<td></td>
</tr>
<tr>
<td>-S3</td>
<td>0x00000000-0xffffffff</td>
<td>S7 record</td>
<td></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)

  [Syntax]
  - -I16
  - -I20
  - -I32

  [Parameter]
  None

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

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

Table 15.3.2: List of Output Hex Format Options

<table>
<thead>
<tr>
<th>Specification</th>
<th>Range of data that can be output</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>-I16</td>
<td>0x00000000-0xFFFFF</td>
<td>HEX8 format</td>
</tr>
<tr>
<td>-I20</td>
<td>0x00000000-0xFFFFFFFF</td>
<td>HEX16 format</td>
</tr>
<tr>
<td>-I32</td>
<td>0x00000000-0xFFFFFFFF</td>
<td>HEX32 format</td>
</tr>
</tbody>
</table>
15.3.3 Specifying to Output Start Address Record (-entry)

Use this option to specify the start segment address record or the start linear address record are output.

- Specifying to Output Start Address Record (-entry)

  [Syntax]
  
  -entry

  [Parameter]
  None

  [Description]
  Use this option to specify the start segment address record or the start linear address record are output.

  If there is no start address information in the input file, a warning will be output (W1504U: Start address information is not in an input file).

  To the Table 15.3.3 described below, the start address record output according to specify Output HEX Format Option (-I16/-I20/-I32) and the data address.

<table>
<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>0x00000000-0x00000000</td>
<td>Start segment address record</td>
</tr>
<tr>
<td>-I32</td>
<td>0x00000000-0x00000000</td>
<td>Start segment address record</td>
</tr>
<tr>
<td></td>
<td>0x00100000-0xFFFF0000</td>
<td>Start linen address record</td>
</tr>
</tbody>
</table>
[Example]

f2hs ccp903.abs -entry -I16

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

f2hs ccp903.abs -entry -I20

The start segment address record output.

f2hs ccp903.abs -entry -I32

The start segment address record is output when the data address size is from 0x0 to 0xFFFFFFFF.

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

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

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

  [Syntax]
  
  -Xentry

  [Parameter]
  
  None

  [Description]
  
  It specifies to suppress the start segment address record and the start Linear address record output.

  Use this option when you want to cancel the -entry option.

  [Example]
  
  f2hs -entry cpp903.abs -I20 -Xentry

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

This option automatically calls the format adjuster for adjustment after converting a format.

- Specifying to Adjust (-adjust)

[Syntax]

- adjust

[Parameter]

None

[Description]

This option automatically calls the format adjuster for adjustment after converting a format. The starting/ending addresses to be adjusted are automatically set.

If the starting/ending address parameters are specified with the padding (-p) option when this option is given, an error occurs.

When this option is specified, the option for the format adjuster can also be specified.

[Example]

f2hs ccp903 -p 0xEF,0x1FE4,0x1FFF -adjust

Because the starting/ending addresses are specified with the padding option, an error occurs.

f2hs ccp903 -p 0xEF -adjust

This option converts the absolute format load module to an adjusted HEX format. In this case, it pads the location where no data exists with data named 0xEF.
15.4 f2ms (Converting an Absolute Format Load Module into the S format)

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

- 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.

```
Absolute format Load module   f2ms   S format
(abs)                        (.mhp)
```

- Figure 15.4 f2ms (Converting an Absolute Format Load Module into the S format)
  [Address]
  The maximum values of addresses that can be represented in the S format are:
  ```
  S1 type  0x0000FFFF
  S2 type  0x00FFFFFF
  S3 type  0xFFFFFFFF
  ```
  The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive.
  Since the S format supports the same range of addresses, conversions can be made without losing any data.
  According to the address allocation, f2ms outputs the following records:
  ```
  0x00000000 to 0x0000FFFF  : S1 type
  0x00100000 to 0x00FFFFFF  : S2 type
  0x10000000 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.

- 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.

  ![Diagram]

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

  [Address]
  The maximum values of addresses that can be represented in the HEX format are:
  - HEX8  0x0000FFFF
  - HEX16 0x0000FFFF
  - 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, f2ms outputs the following records:
  - 0x00000000 to 0x0000FFFF  : HEX8
  - 0x00000000 to 0x0000FFFF  : HEX16
  - 0x01000000 to 0xFFFFFFFF : HEX32

15.6 f2is (Absolute Format Load Module into the HEX8 format), f2es (Absolute Format Load Module into the HEX16 format)

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

The f2is command converts data at 0x0000 to 0xFFFF and the f2es command converts data at 0x00000 to 0xFFFFF.

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

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

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

[Function]

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

![Diagram](image)

Figure 15.6a f2is (Converting an Absolute Format Load Module into the HEX8 format)

[Description]

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

[Address]

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

**Note:** The absolute format load module that is output by a linker can represent the addresses between 0 and 0xFFFFFFFF inclusive. However, when the absolute format load module is converted into the HEX8 format, the data allocated to the addresses 0x10000 and higher are truncated.

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

An HEX8 format file consists of data records and a trailer record.
CHAPTER 15 LOAD MODULE CONVERTER (f2ms, f2hs, f2is)

■ f2es (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.

![Diagram](.abs) f2es [.ehx]

Figure 15.6b f2es (Converting an Absolute Format Load Module into the HEX16 format)

[Description]

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

[Address]

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

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

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

In the HEX16 format, an extended address record is used to represent the addresses 0x100000 and higher.

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

A starting address record is created at the beginning of an HEX16 format file.
Chapter 16 describes the conversion formats of Format Adjuster.

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

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

- Overview of the Format Adjuster

The Format Adjuster causes each of the records in one file of the format to contain the specified number of data to unify the format. Figure 16.1a shows the concept of the Format Adjuster.

![Figure 16.1a Overview of the Format Adjuster](image)

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 format without changes. Use the padding option (-p option) to pad the portion of an format file containing no data with a specified value. For information on using this option, see Section 14.3, "Padding (-p)".
16.1 Specifications of the Format Adjuster

■ Example of operation

Use this option to unify the lengths of data contained in one record if an existing S format file has records of varying lengths.

![Figure 16.1b Example of operation of the Format Adjuster](image)

Before conversion

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>S007000054455354B8</td>
<td>S20CFF0000109572C160C2D2CEC</td>
</tr>
<tr>
<td>S209FF0008020406080AD1</td>
<td>S20FFFF000D020406080A0C0E1012141660</td>
</tr>
<tr>
<td>S20CFF00180109572C160C2D2CD4</td>
<td>S20FFFF0025020406080A0C0E1012141648</td>
</tr>
<tr>
<td>S20CFF0000109572C160C2D2C61</td>
<td>S20FFFF0000109572C160C2D2CA4</td>
</tr>
<tr>
<td>S20FFFF0008020406080A89</td>
<td>S20FFF003D020406080A0C0E1012141630</td>
</tr>
<tr>
<td>S20CFF000480109572C160C2D2CA4</td>
<td>S20FFF000480109572C160C2D2CA4</td>
</tr>
<tr>
<td>S20FFF0000109572C160C2D2CA4</td>
<td>S20FFF0000109572C160C2D2CA4</td>
</tr>
<tr>
<td>S20FFF003D020406080A0C0E1012141630</td>
<td>S20FFF003D020406080A0C0E1012141630</td>
</tr>
<tr>
<td>S20FFF000480109572C160C2D2CA4</td>
<td>S20FFF000480109572C160C2D2CA4</td>
</tr>
<tr>
<td>S20FFF0000109572C160C2D2CA4</td>
<td>S20FFF0000109572C160C2D2CA4</td>
</tr>
<tr>
<td>S20FFF003D020406080A0C0E1012141630</td>
<td>S20FFF003D020406080A0C0E1012141630</td>
</tr>
</tbody>
</table>

After conversion

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>S007000054455354B8</td>
<td>S31500FF00000109572C160C2D2C020406080A020406B9</td>
</tr>
<tr>
<td>S31500FF00010080A0C0E10121416109572C160C2D2C5B</td>
<td>S31500FF00200020406080A020406800A0C0E1012141629</td>
</tr>
<tr>
<td>S31500FF00300109572C160C2D2C020406080A02040689</td>
<td>S31500FF00400080A0C0E10121416109572C160C2D2C2B</td>
</tr>
<tr>
<td>S31500FF00500200406080A020406800A0C0E10121416F9</td>
<td></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.

Before conversion

| S007000054455354B8 |
| S20CFF0000109572C160C2D2CEC |
| S209FF0008020406080AD1 |
| S20FFF000D020406080A0C0E1012141660 |
| S20CFF00180109572C160C2D2CD4 |
| S209FF0020020406080AB9 |
| S20FFF0025020406080A0C0E1012141648 |
| S20CFF00300109572C160C2D2CBC |
| S209FF0038020406080AA1 |
| S20FFF003D020406080A0C0E1012141630 |
| S20CFF00480109572C160C2D2CA4 |
| S209FF0050020406080AB9 |
| S20FFF0055020406080A0C0E1012141618 |
| S804000000FB |

After conversion

| S007000054455354B8 |
| S30D00FF0008020406080A020406 C1 |
| S31500FF0010 080A00CE101214160109572C160C2D2C 5B |
| S31500FF0020 020406080A020406080A0CE10121416 29 |
| S31500FF0030 0109572C160C2D2C020406080A020406 89 |
| S31500FF0040 080A00CE10121416010957 D7 |
| S70500000000FA |

**Figure 16.1c Example of operation of the Format Adjuster**
16.2 List of Options of the Format Adjuster

Section 16.2 lists the names and functions of options of the Format Adjuster.

#### List of Options of the Format Adjuster

Table 16.2 lists the 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 S record starting address</td>
<td>-ST</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying suppression not to read default option file</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying option file name</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 startup message of program</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Suppression not to output version number and startup message of program</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>Suppression 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>
16.3 Details of Options of the Format Adjuster

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 S format file. For details, see section 16.3.2 Specifying the Output Range (-ran).

■ Specifying an output S format (-S1/-S2/-S3)
   This option specifies an S record to be used when data is output in the S format. 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 (-ST)
   This changes the starting address of the record used when outputting in the format. For details, see section 16.3.5 Specifying Changes to the Starting Address (-ST).
16.3.1 Specifying the Data Length in an Output Record (-len)

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

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

  [Syntax]

  -len  <Data length>

  [Parameter]

  <Data length>
  Select 16, 32, 64, or 128.

  [Description]

  Use this option to specify the number of bytes of data to be output into one record when an format file is formatted.

  Specify 16, 32, 64, or 128 as the data length.

  If this option is omitted, 16 is assumed to be specified in the processing.

  [Precautions]

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

  [Example]

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

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

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

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

Use this option to specify the range of formatting an format file using addresses.

- Specifying the output range (-ran)

[Syntax]

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

[Parameters]

<Starting address>
Starting address
<Ending address>
Ending address

[Description]

Use this option to specify the range of formatting an format file using addresses.
You must specify this option in order to convert.
Specify the starting and ending addresses between 0x0 and 0xffffffff inclusive.
The ending address may be omitted. If omitted, data as much as 64Kbytes from the starting
address is formatted.
You cannot specify values that will make the conversion size more than 2Gbytes.

[Example]

m2ms sfmfile.mhx (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 (0x0D000 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 Details of Options of the Format Adjuster

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.

[Syntax]

-S1

-S2

-S3

[Parameter]
None

[Description]
Use this option to specify an record to be used when data contents are output.

The S Format Adjuster outputs the data contents using one of the S1, S2, and S3 records.
It never outputs data using both the S1 and S2 records.

If more than one of the -S1, -S2, and -S3 options are specified, the one most recently specified is valid. If none of the -S1, -S2, and -S3 options are specified, the S Format Adjuster outputs the data contents into the S3 record.

[Precautions]
If the specification in this option and the output range are not consistent with each other, the S Format Adjuster reports an error and performs no processing.

Specifying this option changes the terminator record to be used for output (S9, S8, and S7 records). (See Table 16.3.3:)

Table 16.3.3: List of output S format 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>0x00000000-0x0000FFFF</td>
<td>S9 record</td>
<td></td>
</tr>
<tr>
<td>-S2</td>
<td>0x00000000-0x00FFFFFF</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 sfmfile.mhx -ran 0xD000,0x10000 -S1
An error occurs because the output range is up to 0x10000, which cannot be represented in the S1 record.

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

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

m2ms sfmfile.mhx -ran 0xE000,0xFFFF -S3
The data from 0xE000 through 0xFFFF is formatted and output in the S3 record.
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.

- Specifying an output HEX format (-I16/-I20/-I32)

[Syntax]

- 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 format.

If more than one of the -I16, -I20, and -I32 options are specified, the one most recently specified is valid. If none of the -I16, -I20, and -I32 options are specified, the f2hs outputs the data contents into the HEX32 format.

[Precautions]

If the specification in this option and the output range are not consistent with each other, the HEX Format Adjuster reports an error and performs no processing.

[Example]

h2hs hfmtfile.hex -ran 0xD000,0x10000 -I16
An error occurs because the output range is up to 0x10000, which cannot be represented in the HEX8 format.

h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I16
The data from 0xE000 through 0xFFFF is formatted and output in the HEX8 format.

h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I20
The data from 0xE000 through 0xFFFF is formatted and output in the HEX16 format.

h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I32
The data from 0xE000 through 0xFFFF is formatted and output in the HEX32 format.
16.3.5 Specifying Changes to the Starting Address (-ST)

This specifies the starting address used when outputting data. This is used to change the address of the data.

■ Specifying Changes to the Starting Address (-ST)

[Syntax]

-ST <Starting address>

[Parameters]

<Starting address>
Starting address

[Description]

This specifies the starting address used when outputting data.

The Format Adjuster determines the starting address of data normally using the starting address specified by the output range specification -ran.

Specifying this option changes the starting address when outputting.

[Example]

m2ms sfmfile.mhx -ran 0xD000,0xFFFF -ST 0x0000

This forms the data in the sfmfile.mhx from 0xD000 to 0xFFFF address. It outputs this as data from 0x0000 address.

m2ms sfmfile.mhx -ran 0xD000,0xFFFF -ST 0x10000

This forms the data in the sfmfile.mhx from 0xD000 to 0xFFFF address. It outputs this as data from 0x10000 address.

m2ms sfmfile.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 the Binary Converter.

17.1 Outline of Binary Converter
17.2 List of Options of the Binary Converter
17.3 Details on Options of the Binary Converter
17.1 Outline of Binary Converter

The binary converter converts files output by the S format or Hex format into binary data files. It not only simply converts to binary data, but also supports a split mode that separates files into several files for output.

■ Outline of the Binary Converter

The binary converter converts object files of the S format or Hex format made by the linkage kit into binary data (memory images) and outputs them to files.

Use m2bs to convert the S format into binary data; use h2bs to convert the Hex format into binary data.

It is possible to specify a multiple of input files (S format or Hex format). Also, it is possible to separate converted binary data into a multiple of files of specified byte sizes (hereinafter referred to as the split mode).

Figure 17.1a  Overview of the Binary Converter

Note: The portion of an format file containing no data is padded with 0xFF (default value). The binary data file to be output is the contents of memory that is output into a file without changes.

Use the padding option (-p) to pad with a specified value the portion of an S format file containing no data. For details on how to use this option, see Section 14.3, “Padding (-p)”.

The default extension of an output binary file is .bin. In the split mode, the extension .bxx (xx is a two digit number (01 through 16)) is unconditionally added.

■ Overview of the split mode

A split mode means that the memory image converted by Binary Converter is split for specified bytes and output into multiple binary data file.

Figure 17.1b shows an overview of the split mode. In Figure 17.1b, 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 “17.3.2Specifying the Split Mode (-sp)”.

288
17.1 Outline of Binary Converter

Figure 17.1b Overview of the split mode
17.2 List of Options of the Binary Converter

Section 17.2 lists the option names and functions of the Binary Converter.

- List of Options of the Binary Converter

Table 17.2 lists options of the S Format Binary Converter.

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing an output file name</td>
<td>-o</td>
<td>* Common option for a converter</td>
</tr>
<tr>
<td>Specifying padding data</td>
<td>-p</td>
<td>* Common option for a converter</td>
</tr>
<tr>
<td>Specifying the output range</td>
<td>-ran</td>
<td>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 and messages</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to output the version and messages</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to display a termination message</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying not to display a termination message</td>
<td>-Xcmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to set the termination code to 1 when a warning occurs</td>
<td>-cwno</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specifying to set the termination code to 0 when a warning occurs</td>
<td>-Xcwno</td>
<td>* Common option</td>
</tr>
</tbody>
</table>
17.3 Details on Options of the Binary Converter

Section 17.3 describes the options of the Binary Converter.
For information on the common options for Linkage kit, see CHAPTER 3, "COMMON OPTIONS". For information on the common options for a converter, see CHAPTER 14, "COMMON OPTIONS OF AN OBJECT FORMAT CONVERTER".

■ Specifying the output range (-ran)
  
  This option specifies the range of an S format file or HEX format file to be converted to a binary image. For details, see Section "17.3.1Specifying 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 "17.3.2Specifying the Split Mode (-sp)".

■ Specifying the inhibition of the split mode (-Xsp)
  
  This option nullifies the split mode specification (-sp). For details, see Section "17.3.3Specifying the Inhibition of the Split Mode (-Xsp)".

■ Specifying to create a map list file (-m)
  
  This option specifies that conversion information is output in a list file. For details, see Section "17.3.4Specifying 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 "17.3.5Specifying not to Create a Map List File (-Xm)".
17.3.1 Specifying the Output Range (-ran)

Use addresses to specify the range to be converted to a binary image. This option must be specified for Binary Converter.

- Specifying the output range (-ran)

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

[Parameters]
- Starting address
- Ending address

[Description]
Use addresses to specify the range of an S format file or HEX 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 sfmtfile.mhx (An example of not using the -ran option)
```
An error occurs because the output range is not specified.

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

[Example 3]
```
m2bs sfmtfile.mhx -ran 0xD000 (An example of not using the ending address)
```
The data in sfmtfile.mhx as much as 64Kbytes from the address 0xD000 (0x0D000 through 0x1CFFF) is extracted and output into a binary image file.

[Example 4]
```
m2bs sfmtfile.mhx -ran 0xFFFF,0xD000
```
An error occurs because the specified ending address is smaller than the starting address.

[Example 5]
```
m2bs sfmtfile.mhx -ran (Example of omitting all the parameters)
```
An error occurs because the starting address is omitted.
17.3 Details on Options of the Binary Converter

17.3.2 Specifying the Split Mode (-sp)

Specify this option to output a binary image in the split mode.

- **Specifying the split mode (-sp)**
  
  **[Syntax]**
  
  
  - sp < Number of output files > [ , < Number of bytes > ]

  **[Parameters]**

  <Number of output files>
  Specifies how many output destination files the data should be split into. A value between 2 and 16 inclusive may be specified.

  <Number of bytes>
  Specifies the unit of splitting data in bytes. A value between 0x01 and 0xFFFFFFFF inclusive may be specified.

  **[Description]**

  This option is used to output data in turns into multiple files. Use this option, for example, to output every two bytes of data in turns into two files when data in 32bits units is configured using two ROMs with a 16bits data width.

  In <Number of bytes>, you cannot specify a value that will cause one or more output files to have zero-byte output.

  The <Number of bytes> may be omitted. If so, the data is split in 1byte units by default.

  This option allows you to output the 64Kbytes data into two 32Kbytes binary image files. However, this option is used only to output data in turns into multiple files. If you specify the parameters to split the 64Kbytes data into two 32Kbytes files, the last 1Kbytes data is output into the former file.

  If this option is specified, the extension ".bxx" (xx is a two-digit number between 01 and 16 inclusive) is unconditionally added to the output file.

  **[Example 1]**

  m2bs sfmtfile.mhx (Example of not using the -sp option)

  A binary image is output into sfmtfile.bin.

  **[Example 2]**

  m2bs sfmtfile.mhx -sp 2

  Every one byte of a binary image is output in turns into sfmtfile.b01 and sfmtfile.b02.

  **[Example 3]**

  m2bs sfmtfile.mhx -sp 2,2

  Every two bytes of a binary image is output in turns into sfmtfile.b01 and sfmtfile.b02.
17.3.3 Specifying the Inhibition of the Split Mode (-Xsp)

The -Xsp option nullifies the split mode specification (-sp).

- Specifying the inhibition of the split mode (-Xsp)

  [Syntax]
  
  - Xsp

  [Parameter]
  None

  [Description]
  Specify this option to nullify the -sp specification.
  This option needs not be specified in particular because it is the default.

  [Example]
  
  m2bs  ccp903.mhx -ran 0xE000,0xFFFF
  m2bs  cpp903.mhx -Xsp -ran 0xE000,0xFFFF

  The default processing does not run in the split mode.
  The above specifications are equivalent.

  m2bs  -f option.file ccp903 -Xsp

  You may sometimes want to temporarily change the specification in an option file when the
  option file is used to execute the program.

  If the -sp option exists in option.file, there is no need to change the contents of option.file.
  Simply specify the -Xsp option on a command line to nullify the -sp option.
17.3.4 Specifying to Create a Map List File (-m)

Use this option to output the information at the time of conversion into a map list file.

- Specifying to create a map list file (-m)

  [Syntax]
  
  - m <Map list file name>

  [Parameter]
  
  <Map list file name>
  Output map list file name

  [Description]
  
  Use this option to output information at the time of conversion into a map list file.
  The information at the time of conversion is output into a map list file. The following
  information is output. Item (4) is output only if the -sp option is specified.
  1) Input file name information
  2) Output file name information
  3) Output range information
  4) Split unit information
  5) Padding data value information
  
  If the <Map list file> has no extension, the default extension ".mp3" is added.

  [Example 1]
  
  m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF (Example of not specifying the -m option)
  
  A map list file is not created because the -m option is not specified.

  [Example 2]
  
  m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF -m logfile
  
  The information at the time of conversion is output into logfile.mp3.

  | Input file   | :sfmtfile.mhx |
  | Output file  | :sfmtfile.bin |
  | Convert range| :0x00010000 - 0x0001FFFF |
  | Padding data | :0xFF |

  Figure 17.3a Example 1 of Contents of logfile.mp3
[Example 3]

\[\text{m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF -m logfile -sp 2,2}\]

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

<table>
<thead>
<tr>
<th>Input file</th>
<th>sfmtfile.mhx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output file</td>
<td>sfmtfile.b01</td>
</tr>
<tr>
<td></td>
<td>sfmtfile.b02</td>
</tr>
<tr>
<td>Convert range</td>
<td>0x00010000 - 0x0001FFFF</td>
</tr>
<tr>
<td>Split byte</td>
<td>2</td>
</tr>
<tr>
<td>Padding data</td>
<td>0xFF</td>
</tr>
</tbody>
</table>

**Figure 17.3b Example 2 of Contents of logfile.mp3**
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)

  [Syntax]
  
  - Xm

  [Parameter]
  None

  [Description]
  Specify this option to nullify the -m specification.
  There is no need to specify this option because it is the default.

  [Example 1]
  m2bs ccp903.mhx -ran 0xE000,0xFFFF
  m2bs ccp903.mhx -Xm -ran 0xE000,0xFFFF

  A map list file is not output by default.
  The above specifications are equivalent.

  [Example 2]
  m2bs -f option.file ccp903 -Xm

  You may sometimes want to temporarily change the specification in an option file when the
  option file is used to execute the program
  If the -m option exists in option.file, you need not change the contents of option.file but simply
  specify the -Xm option on a command line to nullify the -m option.
Chapter 18 describes the commands of load module 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 HEX 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.

![Diagram showing S format file to m2is to HEX8 format file]

Figure 18.1 m2is (Converting S Format file into HEX8 Format)

**Note:** The S format can represent the addresses 0 through 0xFFFFFFFF. However, when converted into the HEX8 format, the data allocated at the addresses 0x10000 and higher is truncated.
When using this command, be careful of the range of addresses in the conversion source.
Since a HEX8 format file consists of data records and a trailer record, the starting address information in the S format will be lost.
18.2 m2es (Converting a S Format File into the HEX16 Format)

A S Format file is converted into the HEX16 format.
The data at the addresses 0 through 0xFFFFF is to be converted.

- m2es (Converting a S Format File into the HEX16 Format)

  [Function]
  The m2es command converts a S format file into the HEX16 format.

  ![Diagram](image)

  **Figure 18.2 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.
18.3  i2ms (Converting a HEX8 Format File into the S Format)

A HEX8 format is converted into the S format. The data at the addresses 0 through 0xFFFF is to be converted.

- i2ms (Converting a HEX Format File into the Motola S Format)

  [Function]

  The i2ms command converts a HEX8 format file into the S format.

  ![Figure 18.3  m2is (Converting HEX8 Format file into S Format)]

  Note: Although the S format can represent the addresses 0 through 0xFFFFFFFF, the HEX 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 HEX Format File into the S Format)

A HEX format is converted into the S format. The data at the addresses 0 through 0xFFFFF is to be converted. The e2ms command is used.

■ e2ms (Converting a HEX Format File into the S Format)

[Function]

The e2ms command converts a HEX format file into the S format.

![Diagram](image)

Figure 18.4  m2es (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

19.2 Questions and Answers on Using an Object Format Converter
19.1 Restrictions on an Object Format Converter

There are several restrictions in the binary converter and format adjuster. No other restrictions have been created for processing when using the other object format converters. It is possible to process using the entire memory that the object format converter can use in execution.

Restrictions on a converter

Table 19.1 shows the restrictions when using the Object Format Converter. However, this is not the maximum limit value available for processing. If it is necessary to register the section name or the symbol name with the internal processing of the object format converter, the object format dynamically retrieves memory. When the object format converter reaches the limit of memory for execution, a message is output to indicate that there is not enough memory and the object format converter process ends.

Table 19.1 Object Format Converter Restrictions List

<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>Memory dependent</td>
</tr>
<tr>
<td>Input file count (m2bs,m2ms)</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Input file count (expected m2bs,m2ms)</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)</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.
Questions and Answers on Using an Object Format Converter

Section 19.1 covers the questions and answers on using an object format converter.

Questions and answers on using an object tool

<table>
<thead>
<tr>
<th>Q.</th>
<th>There are many converters available. Which one should I use?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>A converter is used to convert an absolute format load module file that is output by a linker to an object format that can be read by a ROM writer. You are recommended to use f2ms because the S format fully supports the 32bits addressing space. Use other conversion tools as required.</td>
</tr>
<tr>
<td>Example.</td>
<td>f2ms absfile.abs -&gt; Outputs absfile.mhx in the S format</td>
</tr>
</tbody>
</table>

Example.

When I use S Format Binary Converter, an error, "F9001U: Insufficient memory" is output and I cannot convert data to a binary image. What should I do?

S Format Binary Converter secures as much memory as the area to be converted. If you try to convert a large area at once, an error, "F9001U: Insufficient memory" may be output and the processing interrupted. In such a case, split the target area into multiple continuous areas and create a memory image for each of them. Then, merge the files into one binary image.

Example.

Converting the binary image area from 0xC00000 through 0xFFFFFFF
- If enough memory can be secured (Normal):
  1 m2bs absfile.mhx -ran 0xC00000,0xFFFFFFF

- 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, HEX format, and S format.

APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT
APPENDIX B  HEX FORMAT
APPENDIX C  S RECORD FORMAT
APPENDIX D  LIST OF LINKER OPTIONS
APPENDIX E  LIST OF LIBRARIAN OPTIONS
APPENDIX F  LIST OF COMMANDS AND OPTIONS OF THE OBJECT FORMAT CONVERTER
APPENDIX G  SPECIFICATION DIFFERENCES DEPENDING ON THE OS
APPENDIX H  F2MC-16LX FAMILY STANDBY MODE TRANSITION INSTRUCTION CHECK TOOL
APPENDIX I  DIFFERENCE IN SPECIFICATION OF SOFTUNE LINKER (FLINK907S) AND OLD LINKER (LINK907A)
APPENDIX J  DIFFERENCE IN SPECIFICATION OF SOFTUNE LIBRARIAN (FLIB907S) AND OLD LIBRARIAN (LIB907A)
APPENDIX K  DIFFERENCE IN SPECIFICATION OF SOFTUNE LOAD MODULE CONVERTERS (F2MS, F2HS) AND OLD LOAD MODULE CONVERTERS (LMC907A)
APPENDIX L  OBJECT INFORMATION COMPARISON
APPENDIX M  GUIDE TO CHANGING FUNCTION-CALL INTERFACE
This section explains the classification of error messages output by each tool of the linkage kit and the display format.

### Linkage Kit Error Message Classes

Error messages can be classified into the following four levels based on their importance.

- **Information**
  
  Information notifies the user of contents of processing to confirm the processing. Since this is no errors, and the user can obtain correct processing results.

- **Warning**
  
  Warnings are slighter than errors, and output results can be used almost without causing any trouble. It is possible that other processing than the user system is executed. Check the contents of messages before determining whether output results can be used.

- **Error**
  
  Processing continues to be executed, but troubles have occurred which make it impossible to obtain correct results. The causes of errors must be removed before rerun.

- **Fatal error**
  
  Errors which make it impossible to continue processing. This type of errors results from wrong specifications of the user or problems of the execution environment.
### Linkage Kit Error Message Display Format

Error messages are output by each tool in the following format.

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

### Appendix Figure A  Linkage Kit Error Message Display Format

#### [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. This is a message which is notified when 2 is specified in the -w option.</td>
</tr>
<tr>
<td>I0302L</td>
<td>Debug information not exist (file name)</td>
</tr>
<tr>
<td></td>
<td>Debug information is not contained in the input file. This is a message which is notified when 2 is specified in the -w option. The error is slighter than W1351. The message only report that debug information does not exist.</td>
</tr>
<tr>
<td>I0303L</td>
<td>Removed debug information</td>
</tr>
<tr>
<td></td>
<td>Output file was created after removing debug information. This is a message which is notified when 2 is specified in the -w option.</td>
</tr>
<tr>
<td>I0304L</td>
<td>File include WARNING level error (file name)</td>
</tr>
<tr>
<td></td>
<td>The file indicated here was warned in a warning message when linking. This is a message which is notified when 2 is specified in the -w option.</td>
</tr>
<tr>
<td>I0305L</td>
<td>Ignore address alignment</td>
</tr>
<tr>
<td></td>
<td>Because the -pk option was specified, allocation was carried out ignoring boundary alignment when linking. This is a message which is notified when 2 is specified in the -w option.</td>
</tr>
<tr>
<td>I0306L</td>
<td>Section allocated automatically in &quot;section name&quot; area</td>
</tr>
<tr>
<td></td>
<td>Optimal section allocation was carried out in the area indicated here. This is a message which is notified when 2 is specified in the -w option.</td>
</tr>
</tbody>
</table>
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0307L</td>
<td>Lower compatible cpu type object (file name)</td>
</tr>
<tr>
<td></td>
<td>The file indicated here is a module for a different CPU type with downward compatibility. This is a message which is notified when 2 is specified in the -w option.</td>
</tr>
<tr>
<td>I01312L</td>
<td>Total warning message of S.C.F check: &lt;The number of warning messages &gt;</td>
</tr>
<tr>
<td></td>
<td>This message shows the number of the warning message (W1378L and W1379L) which the F²MC-16LX family Standby Mode Transition Instruction Check Tool outputted. This message is surely outputted, when (W1379L and W1379L) is outputted.</td>
</tr>
<tr>
<td>W1301L</td>
<td>Writable section located in ROM area (section name)</td>
</tr>
<tr>
<td></td>
<td>A writable section was allocated in the address range specified as a ROM area. Examine the section allocation specification.</td>
</tr>
<tr>
<td>W1303L</td>
<td>Section with initial data located in RAM area (section name)</td>
</tr>
<tr>
<td></td>
<td>A section with initial data was allocated in the address range specified as a RAM area. Examine the section allocation specification.</td>
</tr>
<tr>
<td>W1305L</td>
<td>Section (section name) located on out of ROM/RAM area (area name)</td>
</tr>
<tr>
<td></td>
<td>A section exists which is allocated outside the address range specified by using the -ro or -ra option. Check the section using the section map. If you cannot get full information from the linker's map list, you can get further information from the section detail map list.</td>
</tr>
<tr>
<td>W1306L</td>
<td>Exceeded maximum address (section name)</td>
</tr>
<tr>
<td></td>
<td>A section exists which was allocated exceeding the maximum address(0xFFFFFFFF).</td>
</tr>
<tr>
<td>W1307L</td>
<td>Duplicate section name exist (section name)</td>
</tr>
<tr>
<td></td>
<td>Sections with the same name and with different attributes or types exist in multiple modules.</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 (symbol name)</td>
</tr>
<tr>
<td></td>
<td>The same external definition symbol exists in multiple input modules.</td>
</tr>
<tr>
<td></td>
<td>The symbol defined first is valid.</td>
</tr>
<tr>
<td>W1328L</td>
<td>Mismatch symbol type (symbol name)</td>
</tr>
<tr>
<td></td>
<td>This message is notified if the external definition symbol specified</td>
</tr>
<tr>
<td></td>
<td>when an entry point was set using the -e option is neither function</td>
</tr>
<tr>
<td></td>
<td>name nor variable name nor address label name.</td>
</tr>
<tr>
<td></td>
<td>This message is notified if the level is lower than E4326.</td>
</tr>
<tr>
<td>W1330L</td>
<td>Too long ROM section name : exceeded 250 character (section name)</td>
</tr>
<tr>
<td></td>
<td>Up to 250 characters are allowed for the section name to be used for</td>
</tr>
<tr>
<td></td>
<td>ROM- RAM transfer.</td>
</tr>
<tr>
<td>W1332L</td>
<td>No match (file name) argument</td>
</tr>
<tr>
<td></td>
<td>A file was specified using the wild card, but no corresponding file</td>
</tr>
<tr>
<td></td>
<td>was found. This specification is ignored.</td>
</tr>
<tr>
<td>W1351L</td>
<td>Debug information not exist</td>
</tr>
<tr>
<td></td>
<td>Part of the list cannot be created due to insufficient debug information.</td>
</tr>
<tr>
<td></td>
<td>Specify the debug information add option -g when compiling, assembling,</td>
</tr>
<tr>
<td></td>
<td>or linking.</td>
</tr>
<tr>
<td>W1367L</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.</td>
</tr>
<tr>
<td></td>
<td>Please, change the module names.</td>
</tr>
<tr>
<td></td>
<td>The C/C++ compiler automatically generates the module name from the file</td>
</tr>
<tr>
<td></td>
<td>name.</td>
</tr>
<tr>
<td></td>
<td>Please, change the file names.</td>
</tr>
</tbody>
</table>
Appendixes

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1368L</td>
<td>The area specified for the -ro option is outside the internal-ROM area (area name)</td>
</tr>
</tbody>
</table>

The area specified for the -ro option is outside the internal-ROM area. Check whether the specified area is correct. This warning is output even if warning output suppression (-w 0) is specified.

| W1369L   | The area specified for the -ra option is outside the internal-RAM area (area name) |

The area specified for the -ra option is outside the internal-RAM area. Check whether the specified area is correct. This warning is output even if warning output suppression (-w 0) is specified.

| W1370L   | The section is placed outside the ROM area (section name)                |

The section that should be placed within the ROM area is placed outside the ROM area. Check the map file to see if the placement is correct. This warning is output even if warning output suppression (-w 0) is specified.

| W1371L   | The section is placed outside the RAM area (section name)                |

The section that should be placed within the RAM area is placed outside the RAM area. Check the map file to see if the placement is correct. This warning is output even if warning output suppression (-w 0) is specified.

| W1372L   | The section is placed outside the RAM area or the I/O area (section name) |

The section that should be placed within the RAM area or the I/O area is placed outside the RAM area or the I/O area. Check the map file to see if the placement is correct. This warning is output even if warning output suppression (-w 0) is specified.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1373L</td>
<td>The section is placed outside the I/O area (section name)</td>
</tr>
<tr>
<td>W1375L</td>
<td>This is the section (section name) for which layout is not specified in the -sc option.</td>
</tr>
<tr>
<td>W1377L</td>
<td>Total size of the DIR attribute sections exceeded 256 bytes</td>
</tr>
<tr>
<td>W1378L</td>
<td>The standby mode transition instructions which is not suitable was found by fetch unit check (Instruction Pattern Absolute Address Source File Name: Line Number)</td>
</tr>
</tbody>
</table>

The section that should be placed within the I/O area is placed outside the I/O area. Check the map file to see if the placement is correct. This warning is output even if warning output suppression (-w 0) is specified.

This is the section for which the user does not specify layout. Use the specify section layout (-sc) option to specify layout for the corresponding section.

This message is outputted when, with the -check_dirsize option specified, the total size of the DIR-attribute sections exceeds 256 bytes. If this message is outputted, modify the source so that the total size of the DIR-attribute sections becomes 256 bytes or less.

**Note:**

In the following cases, no problem occurs even when the total size of the DIR-attribute sections exceeds 256 bytes. Ignore this message.

- When an assembler source uses two or more direct pages
- When each task of Real Time OS uses a different direct page

In a case where no problem occurs even when the total size of the DIR-attribute sections exceeds 256 bytes, this message can be inhibited by specifying the -Xcheck_dirsize option.

This error message is outputted when it detects the standby mode transition instruction strings that do not meet "H.8  Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition". This message is outputted by a Fetch Unit Check. For details, see APPRNDIX "H.4  Warning Messages".
This error message is outputted when it detects the standby mode transition instruction strings that do not meet "H.8 Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition". This message is outputted by a byte Check. For details, see APPRNDIX "H.4 Warning Messages".

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1379L</td>
<td>The standby mode transition instructions which is not suitable was found by byte check (Instruction Pattern Absolute Address Source File Name: Line Number)</td>
</tr>
<tr>
<td>E4302L</td>
<td>Not found section or section group name (section name or group name)</td>
</tr>
<tr>
<td>E4303L</td>
<td>Undefined ROM/RAM area name (area name)</td>
</tr>
<tr>
<td>E4304L</td>
<td>Symbol name is not found (mangle name / symbol name)</td>
</tr>
<tr>
<td>E4305L</td>
<td>Unresolved external symbol (mangle name / symbol name)</td>
</tr>
<tr>
<td>E4312L</td>
<td>Uncompatible cpu type module (file name)</td>
</tr>
</tbody>
</table>

The file displayed here is an incompatible module. The target CPU for input modules must be the same or a compatible CPU.
This error message is outputted under the conditions shown in Appendix Table A-1.

Appendix Table A-1 Relationship between Input Object and -rp Option

<table>
<thead>
<tr>
<th>Specifying of -rp option</th>
<th>Function-call interface for input object</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided</td>
<td>Normal combination for “register argument passing”</td>
<td>The error (E4313L) is outputted.</td>
</tr>
<tr>
<td>Unprovided</td>
<td>The error (E4313L) is outputted.</td>
<td>Normal combination for “stack argument passing”</td>
</tr>
</tbody>
</table>

If these errors occur, correct the specifying of the -rp option or the function-call interface for the input object, referring to Appendix Table A-1.

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.

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.
The divisor in the division for relocation operations is 0.

| E4330L | Divided by 0 |

This message is notified if there is no addressing by the -sc option or a section without any specification is found.

| E4331L | Indispensable to locate address (section name) |

A group name cannot be specified when a ROM- RAM transfer section is specified using the -sc option.

| E4332L | Not handling group name (group name) |

No ROM address is specified for location information to be allocated in the ROM area.
Specify a ROM address using the -sc option.

| E4333L | Not specified ROM address (section 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.

| E4351L | Relocatable assemble list not correspond to object file (file name) |

The relative assemble list file indicated by the supplementary message is not found.
The linker does not create an absolute assemble list file corresponding to the file name indicated by the supplementary message, and continues processing.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4354L</td>
<td>Illegal relocatable assemble list file format (file name)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4355L</td>
<td>Object data not correspond (file name)</td>
</tr>
</tbody>
</table>

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.

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

The number of dimensions of an array output to the ARRAY list or that of nesting levels of a structure exceeds 8. Processing continues, but those parts which exceed this limit are not output.

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

A dummy section is specified in a file indicated by the supplementary message. Do not write any dummy section when creating an absolute assemble list.

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

One or more sections which exceed the maximum size of a section exist. Some sections have the maximum size depending on their section identification. This message is output if such limits are exceeded. Examine such sections.

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

This message is notified if the maximum address of the bank is exceeded when the section is allocated from the specified location. Examine the section configuration in the bank.
### Appendixes

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

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4366L</td>
<td>Not found locatable address (section name)</td>
</tr>
</tbody>
</table>

No place in all address space can be found where the section can be allocated. Examine the program structure.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4367L</td>
<td>Duplicated module name (file name module name)</td>
</tr>
</tbody>
</table>

The same module name already exists as that intended for linking. Duplicated module names are not allowed. Change the module names.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4368L</td>
<td>Referenced other bank symbol (symbol name)</td>
</tr>
</tbody>
</table>

A symbol defined in another bank was referenced in an intra-bank branch instruction. Use a branch instruction to outside banks if the branch to the corresponding location (symbol) is required.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4369L</td>
<td>Invalid module : conflict compile model (file name)</td>
</tr>
</tbody>
</table>

A module of a different compile model cannot be linked.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4370L</td>
<td>CPU information file not found (file name)</td>
</tr>
</tbody>
</table>

Can't find the target CPU information file specified by the -cpu option. This is detected when the file below is not found.

- `%FETOOL%\LIB\907\907.CSV`
APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4371L</td>
<td>CPU information not found (file name)</td>
</tr>
</tbody>
</table>

Can't find the target CPU information, in the CPU information file, specified by the -cpu option. This is detected when the target CPU information specified in the file below is not found.

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9001L</td>
<td>Insufficient memory</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9011L</td>
<td>Input file not found (file name)</td>
</tr>
</tbody>
</table>

The specified input file cannot be found.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9012L</td>
<td>Library file not found (file name)</td>
</tr>
</tbody>
</table>

If the name of the file indicated here is a default library file, the library file is not stored in the directory indicated by the environmental variable LIB907 or derived from the environmental variable FETOOL(...\fetool\lib).
If the file is a library file specified by the -l or -el option, check whether the file name is specified correctly or the retrieval path specified in the -L option is correct.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9015L</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>F9016L</td>
<td>File read error (file name)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9017L</td>
<td>File write error (file name)</td>
</tr>
</tbody>
</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 linker.
This message is also notified if a write-protected file with the same name exists.
### Appendixes

<table>
<thead>
<tr>
<th>F9021L</th>
<th>Too many options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Too many input file names and options (including those in the option file) are specified on the command line. Divide the files and options, and then activate the linker more than once. The option cannot be used by the linker. See the -help option or this manual.</td>
</tr>
<tr>
<td>F9022L</td>
<td>Illegal option name (option)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>F9023L</td>
<td>Illegal option parameter (option)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>F9024L</td>
<td>Illegal character (option)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>F9026L</td>
<td>Specified value out of range (value)</td>
</tr>
<tr>
<td></td>
<td>A value is specified which is outside the range of allowed values by the -pl, -pw, or -w option.</td>
</tr>
<tr>
<td>F9027L</td>
<td>Option file nested</td>
</tr>
<tr>
<td></td>
<td>Nesting of option files is not allowed. Delete the -f option described in the option file.</td>
</tr>
<tr>
<td>F9030L</td>
<td>Missing input file name</td>
</tr>
<tr>
<td></td>
<td>Specify an input file.</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>F9032L</td>
<td>Output file name same as input one (file name)</td>
</tr>
<tr>
<td></td>
<td>Since the output file name indicated here is the same as the input file name, processing cannot continue.</td>
</tr>
<tr>
<td>F9033L</td>
<td>Illegal file format (file name)</td>
</tr>
<tr>
<td></td>
<td>This message is notified in one of the following cases.</td>
</tr>
<tr>
<td></td>
<td>• The library file format is not correct.</td>
</tr>
<tr>
<td></td>
<td>• The object module format in a library file is not correct.</td>
</tr>
<tr>
<td></td>
<td>• The input file is an absolute format load module.</td>
</tr>
<tr>
<td></td>
<td>• Contents of an input module are not correct.</td>
</tr>
<tr>
<td></td>
<td>• The format of a relative format assemble list file is not correct.</td>
</tr>
<tr>
<td></td>
<td>• CPU Information file format is not correct.</td>
</tr>
<tr>
<td>F9040L</td>
<td>Duplicated file or path name (file name)</td>
</tr>
<tr>
<td></td>
<td>This message is notified in one of the following three cases.</td>
</tr>
<tr>
<td></td>
<td>• Input module files of the same name are specified.</td>
</tr>
<tr>
<td></td>
<td>• Library files of the same name are specified in the -l, or -el option.</td>
</tr>
<tr>
<td></td>
<td>• The same library path name is specified in the -L option more than once.</td>
</tr>
<tr>
<td>F9042L</td>
<td>Duplicated section name (section name)</td>
</tr>
<tr>
<td></td>
<td>The same section name was specified in the -sc or -gr option more than once.</td>
</tr>
<tr>
<td>F9043L</td>
<td>Duplicated symbol name (symbol name)</td>
</tr>
<tr>
<td></td>
<td>The same external reference symbol was specified in the -el option more than once.</td>
</tr>
</tbody>
</table>
### Appendixes

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9044L</td>
<td>Duplicated section group name (group name)</td>
</tr>
<tr>
<td></td>
<td>The same group name was used more than once when setting groups using the -gr option.</td>
</tr>
<tr>
<td>F9047L</td>
<td>No match file name argument</td>
</tr>
<tr>
<td></td>
<td>A file was specified using the wild card, but no corresponding file was found.</td>
</tr>
<tr>
<td>F9052L</td>
<td>Missing '-cpu' option</td>
</tr>
<tr>
<td></td>
<td>No target CPU is specified using the -cpu option.</td>
</tr>
<tr>
<td>F9053L</td>
<td>Missing '-ro' or '-ra' option</td>
</tr>
<tr>
<td></td>
<td>The -ro option or -ra option required for automatic allocation is not specified.</td>
</tr>
<tr>
<td>F9054L</td>
<td>Duplicate section name exist (section name)</td>
</tr>
<tr>
<td></td>
<td>Sections with the same name and with different attributes or types exist in multiple modules. This error is output to &quot;Absolute Assemble List Output&quot;.</td>
</tr>
<tr>
<td>F9056L</td>
<td>Mismatch CPU information file version</td>
</tr>
<tr>
<td></td>
<td>The CPU information file is not suited old. Please obtain the CPU information file of the latest.</td>
</tr>
<tr>
<td>F9077L</td>
<td>S.C.F check process returns error</td>
</tr>
<tr>
<td></td>
<td>S.C.F check process returns error.</td>
</tr>
</tbody>
</table>
The message files used by the linker could not be opened. 
Store the error message files in a predetermined directory.

- lkt907_a.msg

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 contains debug information. Debug information in the library file can be removed using the -O option.

I0407U  Lower compatible CPU type object (file name)

The file indicated here is an object file or a library file which contains a module of a different CPU type downwardly compatible with the library file.

W1401U  Ignore '-pl' option

Though the number of lines of the list is specified (-pl options), the output of the target map list is not specified. This specification is ignored.

W1402U  Ignore '-pw' option

Though the number of lines of the list is specified (-pw options), the output of the target map list is not specified. This specification is ignored.

W1403U  Ignore '-g' option

The specification of creation of a library with debug information (-g option) has any meaning only if addition (-a option) or replacement (-r option) of a module is specified. This specification is ignored.
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1404U</td>
<td>Nothing to operate</td>
</tr>
<tr>
<td></td>
<td>There was no library file change nor module extract operation.</td>
</tr>
<tr>
<td>W1405U</td>
<td>Module not exists to delete (module name)</td>
</tr>
<tr>
<td></td>
<td>The module specified by the -d option is not included in the library file.</td>
</tr>
<tr>
<td></td>
<td>Check the registered module names using the -m option.</td>
</tr>
<tr>
<td>W1406U</td>
<td>Module not exists to extract (module name)</td>
</tr>
<tr>
<td></td>
<td>The module specified by the -x option is not included in the library file.</td>
</tr>
<tr>
<td></td>
<td>Check the registered module names using the -m option.</td>
</tr>
<tr>
<td>E4402U</td>
<td>Duplicated module name (file name module name)</td>
</tr>
<tr>
<td></td>
<td>An attempt was made to register a module with the same name as that which has already been</td>
</tr>
<tr>
<td></td>
<td>registered.</td>
</tr>
<tr>
<td></td>
<td>Duplicated names are not allowed in one library and so the module indicated here is not</td>
</tr>
<tr>
<td></td>
<td>registered. Use the -r option for replacement.</td>
</tr>
<tr>
<td>E4403U</td>
<td>Duplicated external definition symbol name (file name symbol name)</td>
</tr>
<tr>
<td></td>
<td>The registered modules contain external definition symbols and the symbol indicated here has</td>
</tr>
<tr>
<td></td>
<td>already been registered in the library.</td>
</tr>
<tr>
<td></td>
<td>Duplicated external definition symbols are not allowed in one library, the module which</td>
</tr>
<tr>
<td></td>
<td>contains the symbol indicated here is not registered.</td>
</tr>
<tr>
<td>E4404U</td>
<td>Invalid module : type (file name)</td>
</tr>
<tr>
<td></td>
<td>Only the object module format output by the assembler can be registered in the library file.</td>
</tr>
<tr>
<td></td>
<td>The absolute format and relative format load modules output by the linker cannot be registered.</td>
</tr>
</tbody>
</table>
### Appendixes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>E4408U</td>
<td>Too many entry modules: exceed 65535</td>
</tr>
<tr>
<td>E4409U</td>
<td>Too many entry external symbols: exceed 65535</td>
</tr>
</tbody>
</table>

This is not an object file output by the family assembler that can be handled by this librarian. Or a library file created for another family librarian is specified.

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.

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

Create another library so that one library contains at most 65535 symbols.
APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

This error message is outputted under the conditions shown in Appendix Table A-2.

Appendix Table A-2 Relationship between Input Object and Library to Be Edited and -rp Option

<table>
<thead>
<tr>
<th>Specifying of -rp option</th>
<th>Function-call interface for input object and library to be edited</th>
<th>“register argument passing”</th>
<th>“stack argument passing”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
</tr>
<tr>
<td></td>
<td>Normal combination for “register argument passing”</td>
<td>No error is outputted.</td>
<td>Normal combination for “stack argument passing” No error is outputted.</td>
</tr>
<tr>
<td>Unprovided</td>
<td>The error (E4410U) is outputted.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If these errors occur, correct the specifying of the -rp option or the function-call interface for the input object and library to be edited, referring to Appendix Table A-2.

E4470U  CPU information file not found (file name)

Can’t find the target CPU information file specified by the -cpu option. This is detected when the file below is not found.

- %FETOOL%\LIB\907\907.CSV
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4471U</td>
<td>CPU information not found (file name)</td>
</tr>
<tr>
<td></td>
<td>Can't find the target CPU information, in the CPU information file, specified by the -cpu option. This is detected when the target CPU information specified in the file below is not found.</td>
</tr>
<tr>
<td></td>
<td>• %FETOOL%\LIB\907\907.CSV</td>
</tr>
<tr>
<td>F9001U</td>
<td>Insufficient memory</td>
</tr>
<tr>
<td></td>
<td>Enough memory is not available to execute the program.</td>
</tr>
<tr>
<td>F9015U</td>
<td>File open error (file name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>F9016U</td>
<td>File read error (file name)</td>
</tr>
<tr>
<td></td>
<td>It is possible that the file is read-protected.</td>
</tr>
<tr>
<td>F9017U</td>
<td>File write error (file name)</td>
</tr>
<tr>
<td></td>
<td>It is possible that a write-protected file with the same name exists. Or it is also possible that not enough free space of the disk is available, so the file cannot be written. Prepare enough free space in the disk before rerunning the librarian.</td>
</tr>
<tr>
<td>F9021U</td>
<td>Too many options</td>
</tr>
<tr>
<td></td>
<td>Too many input file names and options (including those in the option file) are specified on the command line. Divide the files and options, and then activate the librarian more than once.</td>
</tr>
</tbody>
</table>
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9022U</td>
<td>Illegal option name (option)</td>
</tr>
<tr>
<td></td>
<td>An error is found in the option name specification. Correct the command line and then reactivate.</td>
</tr>
<tr>
<td>F9023U</td>
<td>Illegal option parameter (option)</td>
</tr>
<tr>
<td></td>
<td>An error is found in the parameters to be specified in this option.</td>
</tr>
<tr>
<td>F9026U</td>
<td>Specified value out of range (value)</td>
</tr>
<tr>
<td></td>
<td>The value cannot be specified in the parameters of the -pl or -pw option. See the -help option or this manual.</td>
</tr>
<tr>
<td>F9027U</td>
<td>Option file nested</td>
</tr>
<tr>
<td></td>
<td>Nesting of option files is not allowed. Delete the -f option described in the option file.</td>
</tr>
<tr>
<td>F9033U</td>
<td>Illegal file format (file name)</td>
</tr>
<tr>
<td></td>
<td>The CPU information file format is not correct or the library file format is not correct (no library file) or the input file specified for registration or replacement is not in the object format output from the assembler. A file of a different format is entered or the file is damaged.</td>
</tr>
<tr>
<td>F9035U</td>
<td>Missing library file name</td>
</tr>
<tr>
<td></td>
<td>The library name is not correctly specified. Specify the library file name correctly.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>F9036U</td>
<td>Multiple library file name specified (file name)</td>
</tr>
<tr>
<td></td>
<td>Only one library file can be specified. Select either the file name indicated here or a library file name specified before.</td>
</tr>
<tr>
<td>F9045U</td>
<td>'-O' option conflict with another option</td>
</tr>
<tr>
<td></td>
<td>If the -O option is specified, other options cannot be specified.</td>
</tr>
<tr>
<td>F9046U</td>
<td>'-c' option conflict with another option</td>
</tr>
<tr>
<td></td>
<td>Do not combine the -c option with any other options.</td>
</tr>
<tr>
<td>F9047U</td>
<td>No match (file name) argument</td>
</tr>
<tr>
<td></td>
<td>A file was specified using the wild card, but no corresponding file was found.</td>
</tr>
<tr>
<td>F9052U</td>
<td>Missing '-cpu' option</td>
</tr>
<tr>
<td></td>
<td>No target CPU is specified using the -cpu option.</td>
</tr>
<tr>
<td></td>
<td>In flibs, the target CPU must be specified using the -cpu option.</td>
</tr>
<tr>
<td>F9056U</td>
<td>Mismatch CPU information file version</td>
</tr>
<tr>
<td></td>
<td>The CPU information file is not suited old.</td>
</tr>
<tr>
<td></td>
<td>Please obtain the CPU information file of the latest.</td>
</tr>
<tr>
<td>F9998U</td>
<td>File open failed (file name)</td>
</tr>
<tr>
<td></td>
<td>The message files used by the library file could not be opened.</td>
</tr>
<tr>
<td></td>
<td>Store the error message files(lkt_a.msg,lkt_e.msg) in a predetermined directory.</td>
</tr>
</tbody>
</table>
F9999U  Internal error  (identification information)

If this error occurs, report it to Fujitsu immediately.
## Error Messages of the Object Format Converter

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0501U</td>
<td>Skip start address record</td>
</tr>
<tr>
<td></td>
<td>The start address record was contained in the HEX format, but it was skipped</td>
</tr>
<tr>
<td></td>
<td>since it was not required. Converter processing is not affected and</td>
</tr>
<tr>
<td></td>
<td>conversion is carried out correctly.</td>
</tr>
<tr>
<td>W1501U</td>
<td>File include WARNING level error (file name)</td>
</tr>
<tr>
<td></td>
<td>The file specified for input contained an error of the warning level when</td>
</tr>
<tr>
<td></td>
<td>linking. Check the file before using it.</td>
</tr>
<tr>
<td>W1502U</td>
<td>Unable to convert address (address)</td>
</tr>
<tr>
<td></td>
<td>The file to be converted contains address data that cannot be represented</td>
</tr>
<tr>
<td></td>
<td>in the converted format. All data after the address indicated here is</td>
</tr>
<tr>
<td></td>
<td>discarded. Change the converted format.</td>
</tr>
<tr>
<td>W1503U</td>
<td>-entry option was specified at the time of -I16 specification</td>
</tr>
<tr>
<td></td>
<td>A start address output specification option was specified at the time of</td>
</tr>
<tr>
<td></td>
<td>-I16 was specified.</td>
</tr>
<tr>
<td>W1504U</td>
<td>Start address information is not in an input file</td>
</tr>
<tr>
<td></td>
<td>There is no start address information in an input load module file.</td>
</tr>
<tr>
<td></td>
<td>The f2hs outputs IHEX format without outputting the start address record.</td>
</tr>
<tr>
<td>F9001U</td>
<td>Insufficient memory</td>
</tr>
<tr>
<td></td>
<td>Enough memory is not available to execute the program.</td>
</tr>
<tr>
<td>F9011U</td>
<td>Input file not found (file name)</td>
</tr>
<tr>
<td></td>
<td>The file specified in the input file cannot be found.</td>
</tr>
</tbody>
</table>
### APPENDIX A  ERROR MESSAGES OF THE LINKAGE KIT

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9015U</td>
<td>File open error (file name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>F9016U</td>
<td>File read error (file name)</td>
</tr>
<tr>
<td></td>
<td>It is probable that the file is read-protected or hardware fault has occurred.</td>
</tr>
<tr>
<td>F9017U</td>
<td>File write error (file name)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>F9021U</td>
<td>Too many options</td>
</tr>
<tr>
<td></td>
<td>Too many input file names and options (including those in the option file) are specified on the command line.</td>
</tr>
<tr>
<td>F9022U</td>
<td>Illegal option name (option)</td>
</tr>
<tr>
<td></td>
<td>The option cannot be used for the converter. See the -help option or this manual.</td>
</tr>
<tr>
<td>F9023U</td>
<td>Illegal option parameter (option)</td>
</tr>
<tr>
<td></td>
<td>An error is found in parameters to be specified for this option.</td>
</tr>
<tr>
<td>F9026U</td>
<td>Specified value out of range (value)</td>
</tr>
<tr>
<td></td>
<td>A value outside the range of values allowed by the option is specified.</td>
</tr>
</tbody>
</table>
Nesting of option files is not allowed. Delete the -f option described in the option file.

The address specified in parameters of the option cannot be represented in the converted file format. Specify another address.

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

Specify the name of an input file to be converted.

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

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

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.
A file is input which is not an absolute format load module output from the linker. Change the format of the file to the absolute format and then use it in the converter.

The output range is not specified. Specify the range of output (-ran).

The output range exceeds the limit value.

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

The output file name indicated here is too long to be processed. Make the specified file name shorter.

The message files used by the converter could not be opened. Store both lkt_e.msg and lkt_a.msg in the predetermined directory.

If this error occurs, report it to Fujitsu immediately.
Appendixes
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.

<table>
<thead>
<tr>
<th>Common format</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>(a)</td>
</tr>
</tbody>
</table>

(a): Indicates the start of a record. The character ":" (0x3A) is used.

(b): Indicates the number of bytes in the data part of (e).
Since the actual 1-byte data is represented by 2-byte ASCII code in this format, d1 and d2 in the above figure are counted as one.
11 indicates the high-order digits and 12 indicates the low-order digits. Values in the range of 0 - 255 can be set.
It is "00" - "FF" in ASCII notation and "0x3030" - "0x4646" in hexadecimal notation.

(c): Indicates the address allocated to the first data if the contents of (e) are object data.
a1 indicates the high-order digits and a4 indicates the low-order digits. Values in the range of 0 - 65535 can be set.
It is "0000" - "FFFF" in ASCII notation and "0x30303030" - "0x46464646" in hexadecimal notation.

(d): Indicates the record type.
00 : Data record (HEX8/HEX16/HEX32 format)
01 : End record (HEX8/HEX16/HEX32 format)
02 : Extended segment address record (HEX16/HEX32 format)
03 : Start 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.5.
(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.

Appendix Figure B.1 Common format

[Example]
:020000020036C6Extended 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.*

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

**Appendix Figure B.2 Data record (HEX8/HEX16/HEX32)**
B.3 End Record (HEX8/HEX16/HEX32) Type: 01

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

- End record (HEX8/HEX16/HEX32)

<table>
<thead>
<tr>
<th>:</th>
<th>0</th>
<th>0</th>
<th>0</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>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td>(f)</td>
<td>(g)</td>
<td></td>
<td></td>
<td></td>
<td></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.

Appendix Figure B.3 End record (HEX8/HEX16/HEX32)
B.4 Extended Segment Address Record (HEX16/HEX32)
Type: 02

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

\[ ( ( PA \times 0x10 ) + ( ( DA + DP ) \mod 0x10000 ) ) \mod 0x100000 \]

- **PA**: (e) field value of the record
- **DA**: (c) field value of the record. Here, this value is a relative address.
- **DP**: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

---

**Extended segment address record (HEX16/HEX32)**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>2</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td>(e)</td>
<td>(f)</td>
<td>(g)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**That:**

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.

**That:**

This field is not used and normally 0 is set.

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

\[ [( PA \times 0x10 ) + [( DA + DP ) \mod 0x10000]] \mod 0x100000 \]

- **PA**: (e) field value of the record
- **DA**: (c) field value of the record. Here, this value is a relative address.
- **DP**: Value indicating the data position in the (e) field of the data record calculated by setting the position of the first data to 0.

This is the same method used for calculating the physical address in i8086. With the addition of extended address code, up to 20-bit addresses can be represented.

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

**Appendix Figure B.4 Extended segment address record (HEX16/HEX32)**
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

The start segment address record (HEX16/HEX32)

<table>
<thead>
<tr>
<th>:</th>
<th>0</th>
<th>4</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>3</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>i1</th>
<th>i2</th>
<th>i3</th>
<th>i4</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td>(e)</td>
<td></td>
<td></td>
<td></td>
<td>(f)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As shown in the above figure, (e) is divided into two fields. The paragraph address is set in (e1), and the offset value is set in (e2).

Both p1 and i1 are high-order digits.

(c) is not used and normally 0 is set.

The start address is calculated using the following formula.

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

- **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.

Appendix Figure B.5 Start segment address record (HEX16/HEX32)
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)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td>(e)</td>
<td>(f)</td>
<td>(g)</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 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.

This is the same method used for calculating the physical address in i80386. With the addition of extended linear address record, up to 32-bit addresses can be represented.

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

**Appendix Figure B.6** Extended linear address record (HEX32)
B.7 Start Liner Address Record (HEX32) Type: 05

This is a record to specify the start address of a program being executed.

- Start linear address record (HEX32)

<table>
<thead>
<tr>
<th>:</th>
<th>0</th>
<th>4</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>5</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
<th>e6</th>
<th>e7</th>
<th>e8</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td>(e)</td>
<td>(f)</td>
<td>(g)</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>

(e) is where the 32-bit execution start address is set.
Both e1 are 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 Figure B.7 Start linear address record (HEX32)
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.

This record consists of the above five fields (a)-(e).

The S0 type is called the header type and is placed at the start of a file ahead of each record of S1-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.

11 indicates the high-order digits and 12 indicates the low-order digits. Values in the range of 0-255 can be set.

It is "00" - "FF" in ASCII notation and "0x3030" - "0x4646" in hexadecimal notation.

(c): 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.

Appendix Figure C.1 S0 type (header record)

[Example]

S00600004844521B

Indicates ASCII character string "HDR"
C.2 S1 Type (Data Record: 2-Byte Address)

This is a record to store object data that can be represented in two bytes (0x0000-0xFFFF).

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

The S1 type consists of the above five fields (a)-(e).

(a): Type field. "S1" (0x5331) in ASCII code.

(b): Indicates the number of bytes in (c), (d), and (e).

(c): Indicates the address allocated to the first data of (d).

(d): Object data. Actual 1-byte data is represented by 2-byte ASCII code. In the above figure, d1 and d2 are byte data at the address indicated by (c).

(e): Checksum.

(f): Generally, control code (such as CR and LF) is added.

Appendix Figure C.1 S1 type (data record: 2-byte address)
C.3  S2 Type (Data Record: 3-Byte Address)

The S2 type differs from the S1 type in (c) field size, and is a record to store object data which requires the 3-byte address.

| S  | 2 | l1 | l2 | a1 | a2 | a3 | a4 | a5 | a6 | d1 | d2 | d3 | d4 | d’ | d’ | s1 | s2 |
|----|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| (a)| (b)| (c) | (d) | (e) | (f) |
| 2 bytes | 2 bytes | 6 bytes | n bytes | 2 bytes |

The S2 type differs from the S1 type in the above (c) field size, and is a record to store object data which requires the 3-byte address.

Appendix Figure C.3  S2 type (data record: 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.

| S | 3 | l1 | l2 | a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | d1 | d2 | d3 | d4 | d' | d' | s1 | s2 |
|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| (a) | (b) | (c) | (d) | (e) | (f) |
| 2 bytes | 2 bytes | 8 bytes | n bytes | 2 bytes |

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.

Appendix Figure C.4 S3 type (data record: 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.

<table>
<thead>
<tr>
<th>S5 type (record to manage the number of records)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="S5 type diagram" /></td>
</tr>
</tbody>
</table>

The S5 type consists of the above four fields (a)-(e).

(a):
Type field. "S5" (0x5335) in ASCII code.

(b):
Indicates the number of bytes in (c) and (e).

(See the description in Appendix C.1, "S0 Type (Header Record)").

(c):
Indicates the number 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 0-65535 can be set.

It is "0000" - "FFFF" in ASCII notation and "0x30303030" - "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)").

Appendix Figure C.5   S5 type (record to manage the number of records)
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)

<table>
<thead>
<tr>
<th>S</th>
<th>7</th>
<th>0</th>
<th>5</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
<th>e6</th>
<th>e7</th>
<th>e8</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(e)</td>
<td>(f)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>8 bytes</td>
<td>2 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

(a):
Type field. "S7" (0x5337) in ASCII code.

(b):
Indicates the number of bytes in (c) and (e). Always "05".

(c):
Indicates the start address of execution.
e1 indicates the high-order digits and e8 indicates the low-order digits.

(d):
This field does not exist.

(e):
Checksum.

(f):
Generally, control code (such as CR and LF) is added.

Appendix Figure C.6 S7 type (terminator record)
The S8 type differs from the S7 type in (c) field size and is a terminator record when 3 bytes are required to represent the start address of execution.

<table>
<thead>
<tr>
<th>S</th>
<th>8</th>
<th>0</th>
<th>4</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
<th>e6</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(e)</td>
<td>(f)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>6 bytes</td>
<td>2 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</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.

Appendix Figure C.7  S8 type (terminator record)
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.

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>9</th>
<th>0</th>
<th>3</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>s1</th>
<th>s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>2 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>2 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>4 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e)</td>
<td>2 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(f)</td>
<td>2 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The S9 type differs from the S7 type in the above (c) field size and is a terminator record when 2 bytes are required to represent the start address of execution.

Appendix Figure C.8  S9 type (terminator record)
Appendix D lists the linker options.

**List of linker options**

**Attached Table D** List of linker options

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification for outputting load module file name</td>
<td>-o</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for debugging information output</td>
<td>-g</td>
<td></td>
</tr>
<tr>
<td>Specification for deleting debugging information</td>
<td>-Xg</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for outputting absolute format load module</td>
<td>-a</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for outputting relative format load module</td>
<td>-r</td>
<td></td>
</tr>
<tr>
<td>Map list file specification</td>
<td>-m</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for inhibiting map list output</td>
<td>-Xm</td>
<td></td>
</tr>
<tr>
<td>Canceling the omission of names displayed in the list</td>
<td>-dt</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting Memory Used Information list</td>
<td>-mmi</td>
<td></td>
</tr>
<tr>
<td>Specification of the number of digits in the list 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>Register bank area specification</td>
<td>-rg</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>
<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>Function</td>
<td>Option</td>
<td>Remarks</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------------------</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>User-specified-area check specification</td>
<td>-check_rora</td>
<td></td>
</tr>
<tr>
<td>User-specified-area check suppression specification</td>
<td>-Xcheck_rora</td>
<td>Default</td>
</tr>
<tr>
<td>Section-placed-area check specification</td>
<td>-check_locate</td>
<td></td>
</tr>
<tr>
<td>Section-placed-area check suppression specification</td>
<td>-Xcheck_locate</td>
<td>Default</td>
</tr>
<tr>
<td>Check User-unspecified Section</td>
<td>-check_section</td>
<td></td>
</tr>
<tr>
<td>Inhibit check for presence of user-unspecified section</td>
<td>-Xcheck_section</td>
<td>Default</td>
</tr>
<tr>
<td>Checks Standby Mode Transition Instruction</td>
<td>-check_SCF</td>
<td>Default</td>
</tr>
<tr>
<td>Prohibits checks on the Standby Mode Transition Instruction</td>
<td>-Xcheck_SCF</td>
<td></td>
</tr>
<tr>
<td>Checks only in fetch units</td>
<td>-Wf</td>
<td></td>
</tr>
<tr>
<td>Specified output load module for register argument passing</td>
<td>-rp</td>
<td></td>
</tr>
<tr>
<td>Specified output load module for stack argument passing</td>
<td>-Xrp</td>
<td>Default</td>
</tr>
<tr>
<td>Specifying that Check on the Total Size of the DIR-attribute Sections</td>
<td>-check_dirsize</td>
<td>Default</td>
</tr>
<tr>
<td>Specifying that Check on the Total Size of the DIR-attribute Sections be inhibited</td>
<td>-Xcheck_dirsize</td>
<td></td>
</tr>
<tr>
<td>Specification for relative assemble list input directory</td>
<td>-alin</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output directory</td>
<td>-alout</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output</td>
<td>-als</td>
<td></td>
</tr>
<tr>
<td>Specification for absolute assemble list output module</td>
<td>-alsf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting absolute assemble list output</td>
<td>-Xals</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting ROM/RAM and ARRAY list</td>
<td>-alr</td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list output module</td>
<td>-alrf</td>
<td></td>
</tr>
<tr>
<td>Specification for inhibiting ROM/RAM and ARRAY list output module</td>
<td>-Xalr</td>
<td></td>
</tr>
<tr>
<td>Specification for ROM/RAM and ARRAY list symbol and address display position</td>
<td>-na/-an</td>
<td></td>
</tr>
<tr>
<td>Specification for outputting external symbol cross-reference information list</td>
<td>-xl</td>
<td></td>
</tr>
<tr>
<td>Specification for external symbol cross-reference information list file name</td>
<td>-xf</td>
<td></td>
</tr>
</tbody>
</table>
### Attached Table D List of linker options (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<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>* Common option</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>
Attached Table E lists the librarian options.

### List of librarian options

**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>Specified output library for register passing</td>
<td>-rp</td>
<td></td>
</tr>
<tr>
<td>Specified output library for stack passing</td>
<td>-Xrp</td>
<td>Default</td>
</tr>
<tr>
<td>Specification for inhibiting default option file read</td>
<td>-Xdof</td>
<td>* Common option</td>
</tr>
<tr>
<td>Option file read specification</td>
<td>-f</td>
<td>* Common option</td>
</tr>
<tr>
<td>Help message display specification</td>
<td>-help</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for outputting version number/message</td>
<td>-V</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for inhibiting version number/message output</td>
<td>-XV</td>
<td>* Common option</td>
</tr>
<tr>
<td>End message output specification</td>
<td>-cmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification for inhibiting end message output</td>
<td>-Xcmsg</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification to set end code to 1 when warning occurs</td>
<td>-cwno</td>
<td>* Common option</td>
</tr>
<tr>
<td>Specification to set end code to 0 when warning occurs</td>
<td>-Xcwno</td>
<td>* Common option</td>
</tr>
</tbody>
</table>
APPENDIX F  LIST OF COMMANDS AND OPTIONS OF THE OBJECT FORMAT CONVERTER

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

- List of commands of the object format converter

<table>
<thead>
<tr>
<th>Command name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>f2ms</td>
<td>Absolute format load module -&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 F.b  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 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>Specifying an Output HEX format</td>
<td>-l16,-l20,-l32</td>
<td>f2hs, h2hs only</td>
</tr>
<tr>
<td>Specifying to Adjust</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>
### Appendixes

**APPENDIX G SPECIFICATION DIFFERENCES DEPENDING ON THE OS**

Attached Tables G.a - G.c show specification differences depending on the OS.

#### Specification differences depending on the OS

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

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

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

<table>
<thead>
<tr>
<th>Wildcard pattern</th>
<th>OS type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UNIX system OS</td>
</tr>
<tr>
<td>?</td>
<td>Matches any one character</td>
</tr>
<tr>
<td>*</td>
<td>Matches any character string</td>
</tr>
</tbody>
</table>

**Table G.c  Concrete examples of wildcard expansion**

<table>
<thead>
<tr>
<th>Wildcard pattern</th>
<th>OS type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UNIX system OS</td>
</tr>
<tr>
<td>a?.obj</td>
<td>al.obj matches.</td>
</tr>
<tr>
<td></td>
<td>a.obj does not match.</td>
</tr>
<tr>
<td>a*</td>
<td>al.obj, a.obj, and a.abs match.</td>
</tr>
<tr>
<td>*</td>
<td>abc.z and abc.z match.</td>
</tr>
</tbody>
</table>
APPENDIX H describes F\textsuperscript{2}MC-16LX family Standby Mode Transition Instruction Check Tool.

H.1 General Description of F\textsuperscript{2}MC-16LX Standby Mode Transition Instruction Check Tool
H.2 How to Operate the Check Tool
H.3 Checking Procedures
H.4 Warning Messages
H.5 Check method
H.6 Detectable Standby Mode Transition Instructions
H.7 Restrictions
H.8 Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition
H.1 General Description of F²MC-16LX Standby Mode Transition Instruction Check Tool

This section gives a general description of the F²MC-16LX Standby Mode Transition Instruction Check Tool.

■ General Description of Check Tool

This check tool detects standby mode transition instructions in the program developed for the F²MC-16LX family.

The check tool searches the code section in the absolute format load module file (ABS file) made by the SOFTUNE V3 tool and detects an instruction that seems to access the low power consumption mode control register (LPMCR: address A0). The tool checks an instruction string following the detected instruction and outputs a warning message if the string is improper.

Check whether the instruction string for which a warning message is output is applicable to the cautions in Appendix Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition. If applicable, change the instruction string to one given in the cautions. No warning message will be issued for the standby mode transition instructions that meet the cautions.

This check tool statistically checks a customer program. Therefore, it may not completely detect any code and instruction that seem to access the low power consumption mode control register (LPMCR: address A0), if the customer program performs a transition to standby mode using an indirect register at runtime. In such a case, user must check the standby mode transition instruction strings.

For details about the general checking process of this checking tool and detectable standby transition instructions, see section H.5Check method and section H.6Detectable Standby Mode Transition Instructions.

![Appendix Figure H.1 Function of Standby Cancel Fail Check Tool](Image)
H.2 How to Operate the Check Tool

This section explains how to startup the check tool and its options.

■ How to Startup

Use the following procedures to operate the check tool.

[FORMAT]

SCF_Check [< Options >] < Absolute Format Load Module Filename >

The default extension of an absolute format load module file is "ABS."

[Example]

SCF_Check module.abs
SCF_Check -Wf module.abs

[Explanation]

In order to use a check tool, there are the following three methods.

• Use on the Command Line
• Register to the Workbench Customize Build Function
• Register in the Startup of the SOFTUNE Workbench Tools and use

■ Options

The following shows an option for the check tool.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Wf</td>
<td>Checks only in fetch units. Does not check in byte units.</td>
</tr>
</tbody>
</table>

■ Destination of Warning Messages

Check tool warning messages are output to standard outputs.

To record warning messages to files, use the OS redirect function.

[Example]

SCF_Check >check.log module.abs
SCF_Check >check.log -Wf module.abs
### How to Use on the Command Line

1. Copy SCF_Check.exe to any folder.
2. Start up the MS-DOS prompt (command prompt).
3. Move it to the folder that has the absolute format load module of the target for checking, then input the command as follows.

<table>
<thead>
<tr>
<th>Full path name of the folder to where SCF_Check.exe was copied</th>
<th>SCF_Check absolute format load module name</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\Softune\470\ABS&gt;</td>
<td>mb90470.abs</td>
</tr>
</tbody>
</table>

**Example**

- C:\SCF\SCF_Check module.abs
- C:\SCF\SCF_Check -Wf module.abs

**Example of Execution**

```
C:\Softune\470\ABS>scf_check mb90470.abs
*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check
    (mov io,#imm8, 00FC018B, cpu.c:70)
*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check
    (mov io,#imm8, 00FC019C, cpu.c:77)
*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check
    (mov io,#imm8, 00FC01AD, cpu.c:84)
*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check
    (mov io,#imm8, 00FC01ED, cpu.c:106)
*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check
    (mov io,#imm8, 00FC01F8, cpu.c:111)
*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check
    (mov io,#imm8, 00FC0203, cpu.c:116)
*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check
    (mov io,#imm8, 00FC022E, cpu.c:133)
*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check
    (mov io,#imm8, 00FC0239, cpu.c:138)
*** I0312L: Total warning message of S.C.F check : 8
C:\Softune\470\ABS>
```

**Appendix Figure H.2** Example of Execution on the Command Line
How to Register to the Workbench Customize Build Function

If you use a SOFTUNE Workbench in the V30L25 or later version register SCF_Check.exe to the customize build function, so that the automatical check is performed after creating a load module when executing Build or Make, using the following procedures.

1. Copy SCF_Check.exe to any folder.
2. Start up SOFTUNE Workbench and open the check target project.
4. Select [After] in the [Linker], in the [Setup Customize Build] dialog box, and then click [New].

---

**How to Register**

1. Copy SCF_Check.exe to any folder.
2. Start up SOFTUNE Workbench and open the check target project.
4. Select [After] in the [Linker], in the [Setup Customize Build] dialog box, and then click [New].

**Appendix Figure H.3 Setup Customize Build 1**

5. Input the following to [Title], [Execute Filename] and [Option] in the [Setup Customize Build] dialog box, and then check [Enable] and [Use output window].

<table>
<thead>
<tr>
<th>Input Example</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>SCF Check</td>
</tr>
<tr>
<td>Execute Filename</td>
<td>C:\SCF\SCF_Check.exe</td>
</tr>
<tr>
<td></td>
<td>Specify check tools using full path</td>
</tr>
<tr>
<td>Option</td>
<td>%(LOADMODULEFILE)</td>
</tr>
<tr>
<td></td>
<td>Specify load module file.(Specify load module file -&gt; full path name)</td>
</tr>
<tr>
<td>Executing Directory</td>
<td>Not necessary.</td>
</tr>
</tbody>
</table>
Appendix Figure H.4   Setup Customize Build 2

6. Click [OK] to end the [Setup Customize Build].

 adulti

1. Executing Make or Build executes the check tool automatically. If a warning message
is issued, the following message is output to the output window.

[Example of Execution]

This is a warning message in the check tool.

Caution: A warning message is output in the check tool, and if there is “No Error” is output.

Appendix Figure H.5   Example of Execution on Customize Build
How to Register in the Startup of the SOFTUNE Workbench Tools and How to Use

If you are using a SOFTUNE Workbench in the V30L24 or later version, registers SCF_Check.exe in [Start Tool] in [Tool], to use check tools from the Workbench, using the following procedures.

How to Register

1. Copy SCF_Check.exe to any folder.
2. Start up the Softune Workbench.
4. Input the following to [Title], [Execute Filename] and [Option] in the [Setup Tool] dialog box, and then mark [Use output window].

<table>
<thead>
<tr>
<th>Input Example</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCF Check</td>
<td></td>
</tr>
<tr>
<td>C:\SCF/SCF_Check.exe</td>
<td>Specify check tools using full path</td>
</tr>
<tr>
<td>%a</td>
<td>Specify load module file.</td>
</tr>
<tr>
<td>C:\SCF</td>
<td>Set [Execute Filename] for automatical setting.</td>
</tr>
</tbody>
</table>

5. Click [Set] in the [Setup Tool] dialog box to register the tools.
6. Click [OK] to end the [Setup Tool].
Appendixes

How to Use


Appendix Figure H.7  How to use

[Example of Execution]

This is a warning message in the check tool.

Caution: A warning message is output in the check tool, and if there is "No Error" is output.

Appendix Figure H.8  Example of Execution
H.3 Checking Procedures

This section explains how to check using the check tool.

■ Checking Procedures

Use the check tool to check using the following procedures.

Procedure 1  Applying the check tool without specifying the option.

Redundant messages may be output, but checks both in fetch unit and in byte unit to detect all suspicious locations. This provides a complete, no-miss, check. (However, this excludes indirect register detections when executing.)

[Example]

SCF_Check module.abs

Procedure 2  Reviewing locations where warning messages were issued and amending locations that require amendments.

Ignore redundant warning messages.

Procedure 3  Specify the -Wf option and re-check.

Verify that there are no problems.

[Example]

SCF_Check -Wf module.abs

Repeat procedure 2 and 3 until no more warnings are issued.
Appendixes

H.4 Warning Messages

This section explains the warning messages output when the standby mode transition instruction strings are detected that may not meet Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition.

Warning Message

This check tool outputs the following message when it detects the standby mode transition instruction strings that do not meet Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition.

[FORMAT]

Warning Message (Instruction Pattern Absolute Address Source File Name: Line Number)

In the warning message field, of the two check methods of the fetch unit and byte unit checks, either an F or a B is applied after ‘***’ to identify by which check the instruction pattern was found.

[Example of By a Fetch Unit Check]

*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check. (setb io,#imm8 0x00ffffff source filename:line number)

[Example of By a Byte Unit Check]

*** W1379L: The standby mode transition instructions which is not suitable was found by byte check. (setb io,#imm8 0x00ffffff source filename:line number)

[Remarks]

The source filename, line number as auxiliary information in the parenthesis indicates it was received from the debug information.

If there is no debug information, the source filename and line number do not appear.

To show the source filename and line number, rebuild the program targeted for the check with the debug information.
H.5 Check method

This section explains how to check.

- Checking method
  This tool uses the two methods of checking program, one in a fetch unit and the other in a byte unit.

- Fetch unit check
  Fetches instructions and reads each instruction to check.
  With the fetch unit check, it checks on instruction units so there is no possibility for redundant messages to be issued by mis-detection. However, the back side to this is that accurate checks may not be possible when data other than instructions are output to the code section.

[Example]

<table>
<thead>
<tr>
<th>Code Section Byte Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x42 0x6C 0x43 0xA0 ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation Code, Operand (Hex)</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x42, 0x6C</td>
<td>MOV A,#0x6C</td>
</tr>
<tr>
<td>0x43, 0xA0</td>
<td>MOVX A,#0xA0</td>
</tr>
</tbody>
</table>

Checks MOV A,#0x6C.
Checks MOVX A,#0xA0.

When this tool detects the standby mode transition instruction strings that do not meet Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition in a fetch unit check, it will output the following message.

*** W1378L: The standby mode transition instructions which is not suitable was found by fetch unit check (movw io,A, 00F90003 module.c:61)
■ Byte unit check

This checks using single byte data units regardless of the instruction.

Because this checks using byte patterns while updating the address for single bytes, the byte unit check does not allow miss-detection. The back side to this is that if it determines an operand to be an operation code, there is the possibility that it will output redundant warning messages.

Since a redundant warning message may be output, the programmer should check whether the instruction string for which a warning message is output is a real standby mode transition instruction string.

[Example]

<table>
<thead>
<tr>
<th>Code Section Byte Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation Code, Operand (Hex)</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x42, 0x6C</td>
<td>MOV A,#0x6C</td>
</tr>
<tr>
<td>0x43, 0xA0</td>
<td>MOVX A,#0xA0</td>
</tr>
</tbody>
</table>

Checks 0x42.
Checks 0x6C.
Checks 0x43.
Checks 0xA0.

When this tool detects the standby mode transition instruction strings that do not meet Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition in a byte unit check, it will output the following message.

*** W1379L: The standby mode transition instructions which is not suitable was found by byte check (movw io,A, 00F90003 module.c:61)
When Warning Messages Are Not Output Even When the Standby Transition Instruction is Detected

Even if this tool check detects a standby mode transition instruction, it will output no warning message when an instruction following the standby mode transition instruction is the following instruction string.

1. When an instruction in Table 3 in Appendix immediately after a standby transition instruction is described.

   [Example]
   
   MOVLPMCR,#xxh ; Standby transition instruction
   CLRBEIRR:0 ; Instruction in Table 3 in Appendix
   MOVA,#010H ; Any instruction
   .
   .
   .

2. When the measure code is described immediately after a standby transition instruction.

   Measure Code: JMPP, JCTX, RET, RETP, RETI instructions, describing at least two NOP instruction and JMP system instruction immediately after the standby transition instruction are acceptable. Excluding JMP @(@PC+disp16), JMPP @(@PC+disp16).

   [Example]
   
   MOVLPMCR,#xxh ; Standby transition instruction
   NOP ; Measure CodeFirst NOP
   NOP ; Measure CodeSecond NOP
   JMPLABEL ; Measure CodeJMP system instruction
   .
   .
   .
H.6 Detectable Standby Mode Transition Instructions

This section explains detectable standby mode transition instructions.

- **Detectable Standby Mode Transition Instructions**
  
  If any of the following instructions is used as a standby mode transition instruction, this tool checks instructions following the instruction and outputs a warning message.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov io, #imm8</td>
<td>Detects when io, dir, addr16 is 0x00A0 and #imm8, #imm16 bit4 is 1 and bit7 is 1 or bit6 is 1 or bit3 is 0.</td>
</tr>
<tr>
<td>mov dir, #imm8</td>
<td></td>
</tr>
<tr>
<td>movw io, #imm16</td>
<td></td>
</tr>
<tr>
<td>mov addr16, #imm8</td>
<td></td>
</tr>
<tr>
<td>movw addr16, #imm16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>setb io:bp</td>
<td>Detects when io, dir, addr16 is 0x00A0 and bp is 7 or 6.</td>
</tr>
<tr>
<td>setb dir:bp</td>
<td></td>
</tr>
<tr>
<td>setb addr16:bp</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>clrb io:bp</td>
<td>Detects when io, dir, addr16 is 0x00A0 and bp is 3.</td>
</tr>
<tr>
<td>clrb dir:bp</td>
<td></td>
</tr>
<tr>
<td>clrb addr16:bp</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov io, A</td>
<td>Detects when io, dir, addr16 is 0x00A0.</td>
</tr>
<tr>
<td>mov dir, A</td>
<td></td>
</tr>
<tr>
<td>mov addr16, A</td>
<td></td>
</tr>
<tr>
<td>movw io, A</td>
<td></td>
</tr>
<tr>
<td>movw dir, A</td>
<td></td>
</tr>
<tr>
<td>movw addr16, A</td>
<td></td>
</tr>
<tr>
<td>or addr16, A</td>
<td></td>
</tr>
<tr>
<td>and addr16, A</td>
<td></td>
</tr>
<tr>
<td>orw addr16, A</td>
<td></td>
</tr>
<tr>
<td>andw addr16, A</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX H  F^2MC-16LX family Standby Mode Transition Instruction Check Tool

Table H.6e  Standby Transition Instruction 5

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>movb io:bp,A</td>
<td>Detects when io,dir,addr16 is 0x00A0 and bp is 3 or 6 or 7.</td>
</tr>
<tr>
<td>movb dir:bp,A</td>
<td></td>
</tr>
<tr>
<td>movb addr:bp,A</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** This check tool only performs static checks. It detects only instructions that can be clearly regarded as writing to the low power consumption mode control register (LPMCR/address 0x00A0).

In a program that a transition to standby mode is performed by indirect addressing with an enabled register in which the address (0x00A0) of the low power consumption mode control register is set, the check tool cannot detect any standby mode transition instruction.

For details about detectable standby transition instructions, see the explanation on detectable standby transition instructions in section 7 Restrictions.
H.7 Restrictions

This section explains restrictions on the check tool.

- Restrictions

This check tool has the following restrictions.

<table>
<thead>
<tr>
<th>Items</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undetectable Standby</td>
<td>This check tool detects standby mode transition instructions only when a</td>
</tr>
<tr>
<td>Transition Instructions</td>
<td>transition to standby mode is performed by the instructions shown in Section</td>
</tr>
<tr>
<td></td>
<td>6 Detectable Standby mode Transition Instructions. Detection is impossible</td>
</tr>
<tr>
<td></td>
<td>when performing a standby transition using the following instructions.</td>
</tr>
<tr>
<td></td>
<td>MOV eam, #imm8</td>
</tr>
<tr>
<td></td>
<td>MOV eam, Ri</td>
</tr>
<tr>
<td></td>
<td>MOV eam, A</td>
</tr>
<tr>
<td></td>
<td>MOV @RLi+disp8, A</td>
</tr>
<tr>
<td></td>
<td>MOVW eam, RWi</td>
</tr>
<tr>
<td></td>
<td>MOVW eam, A</td>
</tr>
<tr>
<td></td>
<td>MOVW @RLi+disp8, A</td>
</tr>
<tr>
<td>Checkable files</td>
<td>This check tool targets only absolute format load modules created by the</td>
</tr>
<tr>
<td></td>
<td>SOFTUNE V3 tool. This does not target absolute format load modules created</td>
</tr>
<tr>
<td></td>
<td>by the earlier version SOFTUNE V1 tool.</td>
</tr>
</tbody>
</table>
H.8 Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition

This section explains detectable cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition.

■ Devices applicable to the cautions

Devices whose latest manual and its list of corrigenda list Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition.

■ Access to low power consumption mode control register (LPMCR) using assembler language

Use the instructions in Table 1 when setting transition to standby mode in the low power consumption mode control register (LPMCR).

Be sure to place an instruction string in the following enclosure shown below immediately after the standby mode transition instruction in Table H.9a. If standby control macro instructions in Table H.8b are used, the following instruction string is placed. It is therefore recommendable to use standby control macro instructions when developing a program.

When instruction strings other than one in enclosure is placed, no operation performed after the standby mode is canceled will be ensured.

The operation performed after the standby mode is canceled will be ensured only when any instruction in Table H.8c is immediately after the standby mode transition instruction, even if the instruction string in enclosure is not placed.

<table>
<thead>
<tr>
<th>MOV</th>
<th>LPMCR, #H'xx ; Low power consumption mode transition instruction in Table H.9a</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>JMP</td>
<td>$+3 ; Jump to next instruction</td>
</tr>
<tr>
<td>MOV</td>
<td>A, #H'10 ; Any instruction</td>
</tr>
</tbody>
</table>

When instruction strings other than one in enclosure is placed, no operation performed after the standby mode is canceled will be ensured.

The operation performed after the standby mode is canceled will be ensured only when any instruction in Table H.8c is immediately after the standby mode transition instruction, even if the instruction string in enclosure is not placed.

<table>
<thead>
<tr>
<th>MOV</th>
<th>MOV dir, #imm8</th>
<th>MOV eam, #imm8</th>
<th>MOV eam, Ri</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>io, #imm8</td>
<td>Mov dir, #imm8</td>
<td>Mov addr16, A</td>
</tr>
<tr>
<td>MOV</td>
<td>@RLi+disp8,A</td>
<td>Mov dir, A</td>
<td>Mov addr16, A</td>
</tr>
<tr>
<td>MOVW</td>
<td>io, #imm16</td>
<td>MOVW dir, #imm16</td>
<td>MOVW eam, #imm16</td>
</tr>
<tr>
<td>MOVW</td>
<td>io, A</td>
<td>MOVW dir, A</td>
<td>MOVW addr16, A</td>
</tr>
<tr>
<td>MOVW</td>
<td>@RLi+disp8,A</td>
<td>MOVW dir, A</td>
<td>MOVW addr16, A</td>
</tr>
<tr>
<td>SETB</td>
<td>io:bp</td>
<td>SETB dir:bp</td>
<td>SETB addr16:bp</td>
</tr>
<tr>
<td>CLR</td>
<td>io:bp</td>
<td>CLR dir:bp</td>
<td>CLR addr16:bp</td>
</tr>
</tbody>
</table>
### Standby control macro instructions

When the following standby control macro instructions are used, the instruction string meets Cautions for Access to Low Power Consumption Mode Control Register (LPMCR) for Standby Mode Transition. It is therefore recommendable to use standby control macro instructions when developing a program.

#### Table H.8b Standby Control Macro Instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_STOP_HOLD</td>
<td>Sets the SLP bit of the low power consumption mode control register (LPMCR) to 0 and the STP bit to 1. A transition to stop mode or pseudo clock mode is performed. Keeps the level of the external pin.</td>
</tr>
<tr>
<td>IO_STOP_Z</td>
<td>Sets the SLP bit of the low power consumption mode control register (LPMCR) to 1 and the STP bit to 1. A transition to stop mode or pseudo clock mode is performed. Sets the level of the external pin to high-impedance.</td>
</tr>
<tr>
<td>IO_SLEEP</td>
<td>Sets the SLP bit of the low power consumption mode control register (LPMCR) to 1. A transition to sleep mode is performed.</td>
</tr>
<tr>
<td>IO_TMD_HOLD</td>
<td>Sets the SLP bit of the low power consumption mode control register (LPMCR) to 0 and the STP bit to 1. A transition to clock mode is performed. Keeps the level of the external pins.</td>
</tr>
<tr>
<td>IO_TMD_Z</td>
<td>Sets the SLP bit of the low power consumption mode control register (LPMCR) to 1 and the STP bit to 1. A transition to clock mode is performed. Sets the level of the external pin to high-impedance.</td>
</tr>
<tr>
<td>IO_SET_LPMCR IMM</td>
<td>A value specified in IMM is set in the low power consumption mode control register (LPMCR).</td>
</tr>
</tbody>
</table>
APPENDIX H F2MC-16LX family Standby Mode Transition Instruction Check Tool

Table H.8c List of Instructions
ADDC A
ADDDC A
SUBC A
SUBDC A
ADDW A
SUBW A
DIVU A
MULU A
MULUW A
NOT A
ANDW A
ORW A
XORW A
NOTW A
NEG A
NEGW A
ASRW A
LSRW A / SHRW A
LSLW A / SHLW A
SWAP
SWAPW / XCHW A,T
EXTW
MOV A,Ri
MOV A,ear
MOV A,@RWj
MOV A,@RWj+
MOVX A,Ri
MOVX A,ear
MOVX A,@RWj
MOVX A,@RWj+
MOV @RWj,A
MOV @RWj+,A
MOV Ri,@RWj
MOV Ri,@RWj+
MOV @RWj,Ri
MOV @RWj+,Ri
MOV @RWj,#imm8
MOV @RWj+,#imm8
XCH A,@RWj
XCH A,@RWj+
XCH Ri,@RWj
XCH Ri,@RWj+
MOVW A,RWi
MOVW A,ear
MOVW A,@RWj
MOVW A,@RWj+
MOVW @RWj,A
MOVW @RWj+,A
MOVW RWi,@RWj
MOVW RWi,@RWj+
MOVW @RWj,RWi
MOVW @RWj+,RWi
MOVW @RWj,#imm16
MOVW @RWj+,#imm16

XCHW A,@RWj
XCHW A,@RWj+
XCHW RWi,@RWj
XCHW RWi,@RWj+
MOVL A,@RWj
MOVL A,@RWj+
MOVL @RWj,A
MOVL @RWj+,A
ADD A,@RWj
ADD A,@RWj+
ADD @RWj,A
ADD @RWj+,A
ADDC A,@RWj
ADDC A,@RWj+
ADDC A,ear
SUB A,@RWj
SUB A,@RWj+
SUB @RWj,A
SUB @RWj+,A
SUBC A,@RWj
SUBC A,@RWj+
SUBC A,ear
ADDW A,@RWj
ADDW A,@RWj+
ADDW @RWj,A
ADDW @RWj+,A
ADDCW A,@RWj
ADDCW A,@RWj+
ADDCW A,ear
SUBW A,@RWj
SUBW A,@RWj+
SUBW @RWj,A
SUBW @RWj+,A
SUBCW A,@RWj
SUBCW A,@RWj+
SUBCW A,ear
ADDL A,@RWj
ADDL A,@RWj+
SUBL A,@RWj
SUBL A,@RWj+
INC @RWj
INC @RWj+
DEC @RWj
DEC @RWj+
INCW @RWj
INCW @RWj+
DECW @RWj
DECW @RWj+
INCL @RWj
INCL @RWj+
DECL @RWj
DECL @RWj+
CMP A,@RWj
CMP A,@RWj+

CMPW A,@RWj
CMPW A,@RWj+
CMPL A,@RWj
CMPL A,@RWj+
MULU A
MULU A,@RWj
MULU A,@RWj+
MULUW A
MULUW A,@RWj
MULUW A,@RWj+
DIVU A,@RWj
DIVU A,@RWj+
DIVUW A,@RWj
DIVUW A,@RWj+
AND A,@RWj
AND A,@RWj+
AND @RWj,A
AND @RWj+,A
OR A,@RWj
OR A,@RWj+
OR @RWj,A
OR @RWj+,A
XOR A,@RWj
XOR A,@RWj+
XOR @RWj,A
XOR @RWj+,A
NOT @RWj
NOT @RWj+
ANDW A,@RWj
ANDW A,@RWj+
ANDW @RWj,A
ANDW @RWj+,A
ORW A,@RWj
ORW A,@RWj+
ORW @RWj,A
ORW @RWj+,A
XORW A,@RWj
XORW A,@RWj+
XORW @RWj,A
XORW @RWj+,A
NOTW @RWj
NOTW @RWj+
ANDL A,@RWj
ANDL A,@RWj+
ORL A,@RWj
ORL A,@RWj+
XORL A,@RWj
XORL A,@RWj+
NEG @RWj
NEG @RWj+
NEGW A
NEGW @RWj
NEGW @RWj+
NRML A,R0

RORC A
RORC @RWj
RORC @RWj+
ROLC A
ROLC @RWj
ROLC @RWj+
JMP @(@RWj)
JMP @(@RWj+)
JMPP @(@RWj)
JMPP @(@RWj+)
CALL @(@RWj)
CALL @(@RWj+)
CALLP @(@RWj)
CALLP @(@RWj+)
CALLP @ear
DBNZ @RWj,REL
DBNZ @RWj+,REL
DWBNZ @RWj,REL
DWBNZ @RWj+,REL
INT9
PUSHW A
PUSHW AH
PUSHW PS
AND CCR,#imm8
OR CCR,#imm8
MOVEA RWi,@RWj
MOVEA RWi,@RWj+
MOVEA A,@RWj
MOVEA A,@RWj+
SETB io:bp
CLRB io:bp
CBNE @RWj,#imm8,rel
CWBNE
@RWj,#imm16,rel
MUL A,ear
MUL A,@RWj
MUL A,@RWj+
MULW A,RWi
MULW A,@RWj
MULW A,@RWj+
DIV A,@RWj
DIV A,@RWj+
DIVW A,@RWj
DIVW A,@RWj+
MUL A
MULW A
DIV A

385


Access to low power consumption mode control register (LPMCR) using C language, not standby control macro instructions

To set transition to standby mode in the low power consumption mode control register (LPMCR), use any of the following methods:

1. Use an instruction for transition to standby mode as a function, and insert two built-in instructions of _wait_nop() immediately after the standby mode transition instruction. If interrupts other than one for return to standby mode may occur within the function, perform optimization at compiling time and inhibit the occurrence of a LINK/UNLINK instruction.

   Example (Function for transition to clock mode or time-based timer mode)

   ```c
   void enter_watch() {
       IO_LPMCR.byte = 0x10; /* Set TMD bit of LPMCR to 0. */
       __wait_nop();
       __wait_nop();
   }
   ```

2. Write an instruction for transition to standby mode in the _asm statement, and insert two NOP and JMP instructions immediately after the standby mode transition instruction.

   Example (Transition to sleep mode)

   ```c
   __asm("MOVI:_IO_LPMCR, #H'58"); /* Set SLP of LPMCR to 1. */
   __asm("NOP");
   __asm("NOP");
   __asm("JMP$+3"); /* Jump to next instruction. */
   ```

3. Write an instruction for transition to standby mode between #pragma asm and #pragma endasm, and insert two NOP and JMP instructions immediately after the standby mode transition instruction.

   Example (Transition to stop mode)

   ```c
   #pragma asm
   MOVI:_IO_LPMCR, #H'98/* Set STP of LPMCR to 1. */
   NOP
   NOP
   JMP$+3/* Jump to next instruction. */
   #pragma endasm
APPENDIX I DIFFERENCE IN SPECIFICATION OF SOFTUNE LINKER (FLINK907S) AND OLD LINKER (LINK907A)

APPENDIX I describes Difference in Specification of SOFTUNE Linker (flnk907s) and Old Linker (link907a).

■ Preface

This document describes the difference in the specification of the SOFTUNE linker (flnk907s) and the old linker (link907a).

flnk907s is updated from link907a in the following points:

- Command line input
- Absolute assemble list creation function included in linker
- Alleviation of restrictions (input file count, section count, symbol count, etc.)
- Enhancement of environmental variables
- Support for external symbol cross reference information list creation function
- Support for symbol detailed information list creation function
- Support for section map creation function for each module

■ Absolute Assemble List Creation Function

The function of the absolute assemble list creation tool (meal907a), which is another product in the old tool, is included in the linker.

![Appendix Figure I.1  Function of Standby Cancel Fail Check Tool](image)
Appendixes

■ Difference in Command Lines

The difference in the command line specification at activation is given in Table I.a and Table I.b.

Table I.a  Activation Format for Linker

<table>
<thead>
<tr>
<th>flnk907s</th>
<th>link907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>flnk907s [option]... [file name]</td>
<td>link907a file name [option]...</td>
</tr>
<tr>
<td></td>
<td>link907a @option file</td>
</tr>
</tbody>
</table>

Table I.b  Option Specification for Linker

<table>
<thead>
<tr>
<th></th>
<th>flnk907s</th>
<th>link907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where to specify options</td>
<td>Any position allowed</td>
<td>After specification of file name</td>
</tr>
<tr>
<td>Specification of option file</td>
<td>-f option file</td>
<td>@ option file</td>
</tr>
<tr>
<td>Specification of two or more option files</td>
<td>Allowed</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>

■ Environmental Variables

The difference in the environmental variables is given in Table I.c.

Table I.c  Environmental Variables

<table>
<thead>
<tr>
<th></th>
<th>flnk907s</th>
<th>link907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification of working directory</td>
<td>TMP</td>
<td>TMP</td>
</tr>
<tr>
<td>Specification of installation path</td>
<td>FETOOL</td>
<td>-</td>
</tr>
<tr>
<td>Specification of display character code</td>
<td>FELANG</td>
<td>-</td>
</tr>
<tr>
<td>Specification of default option file store path</td>
<td>OPT907</td>
<td>-</td>
</tr>
<tr>
<td>Specification of library retrieval path</td>
<td>LIB907</td>
<td>LIB907</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIB905</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIB902</td>
</tr>
</tbody>
</table>
## Options

The difference in the options is given in Table I.d and Table I.e.

<table>
<thead>
<tr>
<th>Options</th>
<th>flnk907s</th>
<th>link907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input file name specification</td>
<td>Not required</td>
<td>-LINK</td>
</tr>
<tr>
<td>Load module output specification</td>
<td>Not required</td>
<td>-OBJECT</td>
</tr>
<tr>
<td>Load module output suppression</td>
<td>-</td>
<td>-NOOBJECT</td>
</tr>
<tr>
<td>Output load module name specification</td>
<td>-o</td>
<td>-NAME</td>
</tr>
<tr>
<td>Debug information output specification</td>
<td>-g</td>
<td>-DEBUG/-DEV</td>
</tr>
<tr>
<td>Debug information output suppression</td>
<td>-Xg</td>
<td>-NODEBUG</td>
</tr>
<tr>
<td>Absolute format load module output</td>
<td>-a</td>
<td>-FORM A</td>
</tr>
<tr>
<td>Relative format load module output</td>
<td>-r</td>
<td>-FORM R</td>
</tr>
<tr>
<td>Map list file name specification</td>
<td>-m</td>
<td>-LIST</td>
</tr>
<tr>
<td>Map list output suppression</td>
<td>-Xm</td>
<td>-NOLIST</td>
</tr>
<tr>
<td>List display part name omission cancel</td>
<td>-dt</td>
<td>-</td>
</tr>
<tr>
<td>Simple linkage list specification</td>
<td>-</td>
<td>-BRIEF</td>
</tr>
<tr>
<td>List 1 page line count specification</td>
<td>-pl</td>
<td>-PAGELENGTH</td>
</tr>
<tr>
<td>List 1 line digit count specification</td>
<td>-pw</td>
<td>-PAGEWIDTH</td>
</tr>
<tr>
<td>External symbol cross reference list creation</td>
<td>-xl</td>
<td>-CROSS_REFERENCE</td>
</tr>
<tr>
<td>External symbol cross reference list file name specification</td>
<td>-xlf</td>
<td>Not required</td>
</tr>
<tr>
<td>External symbol cross reference list output suppression</td>
<td>-Xxl</td>
<td>-NOCROSS_REFERENCE</td>
</tr>
<tr>
<td>Symbol list output specification</td>
<td>-sl</td>
<td>-LOCAL_SYMBOL</td>
</tr>
<tr>
<td>Symbol list file name specification</td>
<td>-slf</td>
<td>Not required</td>
</tr>
<tr>
<td>Symbol list output suppression specification</td>
<td>-Xsl</td>
<td>-</td>
</tr>
<tr>
<td>Section information map output specification</td>
<td>-ml</td>
<td>-</td>
</tr>
<tr>
<td>Section information map file name specification</td>
<td>-mlf</td>
<td>-</td>
</tr>
<tr>
<td>Section information map output suppression</td>
<td>-Xml</td>
<td>-</td>
</tr>
<tr>
<td>Error information file output allowed/suppressed</td>
<td>-</td>
<td>-(NO)ERRORPRINT</td>
</tr>
<tr>
<td>Warning level setting</td>
<td>-w</td>
<td>-(NO)WARNING</td>
</tr>
<tr>
<td>ROM area specification</td>
<td>-ro</td>
<td>-ROM_BANK</td>
</tr>
<tr>
<td>RAM area specification</td>
<td>-ra</td>
<td>-RAM_BANK</td>
</tr>
<tr>
<td>Section arrangement order (address specification)</td>
<td>-sc</td>
<td>-LOCATE/-SEGMENT</td>
</tr>
<tr>
<td>Section group specification</td>
<td>-gr</td>
<td>-</td>
</tr>
<tr>
<td>Register bank area setting</td>
<td>-rg</td>
<td>-REGISTER_BANK</td>
</tr>
</tbody>
</table>
# Appendixes

## Table I.d  Linker Options (Continued)

<table>
<thead>
<tr>
<th>Option Description</th>
<th>flnk907s</th>
<th>link907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto arrangement specification</td>
<td>-AL</td>
<td>-AUTO_LOCATE</td>
</tr>
<tr>
<td>Retrieval library file specification</td>
<td>-l</td>
<td>-LIBRARY</td>
</tr>
<tr>
<td>Library retrieval path specification</td>
<td>-L</td>
<td>-</td>
</tr>
<tr>
<td>Library retrieval for individual symbols</td>
<td>-el</td>
<td>-</td>
</tr>
<tr>
<td>Library retrieval function suppression</td>
<td>-nl</td>
<td>-</td>
</tr>
<tr>
<td>Default library retrieval suppression</td>
<td>-nd</td>
<td>-</td>
</tr>
<tr>
<td>Input address specification</td>
<td>-e</td>
<td>-</td>
</tr>
<tr>
<td>External symbol value setting</td>
<td>-df</td>
<td>-</td>
</tr>
<tr>
<td>Status code file output</td>
<td>-</td>
<td>-RETCODE</td>
</tr>
<tr>
<td>Termination of the option file</td>
<td>Not required</td>
<td>-END</td>
</tr>
<tr>
<td>Help display</td>
<td>-help</td>
<td>-</td>
</tr>
<tr>
<td>Target CPU specification</td>
<td>-cpu</td>
<td>-CPU</td>
</tr>
<tr>
<td>Default option file input suppression</td>
<td>-Xdof</td>
<td>-</td>
</tr>
<tr>
<td>Activation message display/no display</td>
<td>-V/-XV</td>
<td>-</td>
</tr>
<tr>
<td>Warning-time return value change allowed/suppressed</td>
<td>-cwno/-Xcwno</td>
<td>-</td>
</tr>
<tr>
<td>Termination message display/no display</td>
<td>-cmsg/-Xcmsg</td>
<td>-</td>
</tr>
<tr>
<td>Input termination specification</td>
<td>Not required</td>
<td>-END</td>
</tr>
</tbody>
</table>

## Table I.e  Absolute Format Assemble List Output Options

<table>
<thead>
<tr>
<th>Option Description</th>
<th>flnk907s</th>
<th>meal907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input directory specification</td>
<td>-alin</td>
<td>-INPUT</td>
</tr>
<tr>
<td>Output directory specification</td>
<td>-alout</td>
<td>-OUTPUT</td>
</tr>
<tr>
<td>List output specification</td>
<td>-als</td>
<td>-</td>
</tr>
<tr>
<td>List output module specification</td>
<td>-alsf</td>
<td>-LIST</td>
</tr>
<tr>
<td>List output suppression specification</td>
<td>-Xals</td>
<td>-NOLIST</td>
</tr>
<tr>
<td>ROM/RAM, ARRAY list output specification</td>
<td>-alr</td>
<td>-ROM_RAM_LIST</td>
</tr>
<tr>
<td>ROM/RAM, ARRAY list output module specification</td>
<td>-alrf</td>
<td>-ROM_RAM_LIST</td>
</tr>
<tr>
<td>ROM/RAM, ARRAY list output suppression specification</td>
<td>-Xalr</td>
<td>-NOROM_RAM_LIST</td>
</tr>
<tr>
<td>ROM/RAM, ARRAY list display position specification</td>
<td>-na/-an</td>
<td>-NAME_ADDRESS/-ADDRESS_NAME</td>
</tr>
</tbody>
</table>
Alleviation of Restrictions

The difference in the processing limits is given in Table I.f.

<table>
<thead>
<tr>
<th></th>
<th>flnk907s</th>
<th>link907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input file count</td>
<td>65535</td>
<td>1024</td>
</tr>
<tr>
<td>Linked module count</td>
<td>65535</td>
<td>3000</td>
</tr>
<tr>
<td>Linked section count</td>
<td>65535</td>
<td>8192</td>
</tr>
<tr>
<td>External definition symbol count</td>
<td>65535</td>
<td>65535</td>
</tr>
<tr>
<td>External reference symbol count</td>
<td>65535</td>
<td>65535</td>
</tr>
</tbody>
</table>
APPENDIX J describes Difference in Specification of SOFTUNE Librarian (flib907s) and Old Librarian (lib907a).

■ Preface

This document describes the difference in the specification of the SOFTUNE librarian (flib907s) and the old librarian (lib907a).

■ Difference in Command Lines

The difference in the command line specification at activation is given in Table J.a and Table J.b.

**Table J.a  Activation Format for Librarian**

<table>
<thead>
<tr>
<th></th>
<th>flib907s</th>
<th>lib907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>[option]... [file name]</td>
<td>flib907s [option]... [file name]</td>
<td>lib907a file name [option]...</td>
</tr>
<tr>
<td></td>
<td>lib907a @option file</td>
<td></td>
</tr>
</tbody>
</table>

**Table J.b  Option Specification for Librarian**

<table>
<thead>
<tr>
<th></th>
<th>flib907s</th>
<th>lib907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where to specify options</td>
<td>Any position allowed</td>
<td>After specification of file name</td>
</tr>
<tr>
<td>Specification of option file</td>
<td>-f option file</td>
<td>@ option file</td>
</tr>
<tr>
<td>Specification of two or more option files</td>
<td>Allowed</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>

■ Environmental Variables

The difference in environmental variables is given in Table J.c.

**Table J.c  Environmental Variables**

<table>
<thead>
<tr>
<th></th>
<th>flib907s</th>
<th>lib907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification of working directory</td>
<td>TMP</td>
<td>TMP</td>
</tr>
<tr>
<td>Specification of installation path</td>
<td>FETOOL</td>
<td>-</td>
</tr>
<tr>
<td>Specification of display character code</td>
<td>FELANG</td>
<td>-</td>
</tr>
<tr>
<td>Specification of default option file store path</td>
<td>OPT907</td>
<td>-</td>
</tr>
</tbody>
</table>
### Options

The difference in the options is given in Table J.d.

#### Table J.d Librarian Options

<table>
<thead>
<tr>
<th>Option</th>
<th>flib907s</th>
<th>lib907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library file creation</td>
<td>-a</td>
<td>-CREATE(-C)</td>
</tr>
<tr>
<td>Module addition</td>
<td>-a</td>
<td>-APPEND(-A)</td>
</tr>
<tr>
<td>Module replacement</td>
<td>-r</td>
<td>-REPLACE(-R)</td>
</tr>
<tr>
<td>Module deletion</td>
<td>-d</td>
<td>-DELETE(-D)</td>
</tr>
<tr>
<td>Module selection</td>
<td>-</td>
<td>-SELECT(-S)</td>
</tr>
<tr>
<td>Module extraction</td>
<td>-x</td>
<td>-EXTRACT(-X)</td>
</tr>
<tr>
<td>Backup file creation allowed/suppressed</td>
<td>-b/-Xb</td>
<td>-</td>
</tr>
<tr>
<td>Debug information output allowed/suppressed</td>
<td>-g/-Xg</td>
<td>-</td>
</tr>
<tr>
<td>List file output</td>
<td>-m { file name</td>
<td>-}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Keyword:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Always output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[NO]MODULE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[NO]SEGMENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[NO]PUBLIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[NO]EXTRN</td>
</tr>
<tr>
<td>List file output inhibited</td>
<td>-Xm</td>
<td>-</td>
</tr>
<tr>
<td>External symbol cross reference list creation</td>
<td>-</td>
<td>-CROSS_REFERENCE(-XR)</td>
</tr>
<tr>
<td>List 1 page line count specification</td>
<td>-pl</td>
<td>-</td>
</tr>
<tr>
<td>List 1 line digit count specification</td>
<td>-pw</td>
<td>-</td>
</tr>
<tr>
<td>.LIB file contents check</td>
<td>-c</td>
<td>-</td>
</tr>
<tr>
<td>.LIB file contents optimization</td>
<td>-O</td>
<td>-</td>
</tr>
<tr>
<td>Error information file output enabled/inhibited</td>
<td>-</td>
<td>-ERRORPRINT(-[NO]RP)</td>
</tr>
<tr>
<td>Status code file output</td>
<td>-</td>
<td>RETCODE(-RC)</td>
</tr>
<tr>
<td>Termination of option file Not required</td>
<td>-</td>
<td>END(E)</td>
</tr>
<tr>
<td>Help display</td>
<td>-help</td>
<td>-</td>
</tr>
<tr>
<td>Target CPU specification</td>
<td>-cpu</td>
<td>-CPU</td>
</tr>
<tr>
<td>CPU type check suppression</td>
<td>-</td>
<td>-NOCPUTYPECHECK(-NCTC)</td>
</tr>
<tr>
<td>Default option file input suppression</td>
<td>-Xdof</td>
<td>-</td>
</tr>
<tr>
<td>Activation message display/no display</td>
<td>-V/-XV</td>
<td>-</td>
</tr>
<tr>
<td>Warning-time return value change allowed/suppressed</td>
<td>-cwno/-Xcwno</td>
<td>-</td>
</tr>
<tr>
<td>Termination message display/no display</td>
<td>-cmsg/-Xcmsg</td>
<td>-</td>
</tr>
</tbody>
</table>
## Alleviation of Restrictions

The difference in the processing limits is given in Table J.e.

### Table J.e  Processing Limits

<table>
<thead>
<tr>
<th></th>
<th>flib907s</th>
<th>lib907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input file count</td>
<td>65535</td>
<td>255</td>
</tr>
<tr>
<td>Linked module count</td>
<td>65535</td>
<td>255</td>
</tr>
<tr>
<td>Linked section count</td>
<td>65535</td>
<td>255</td>
</tr>
<tr>
<td>External definition symbol count</td>
<td>65535</td>
<td>1536</td>
</tr>
<tr>
<td>External reference symbol count</td>
<td>65535</td>
<td>1536</td>
</tr>
</tbody>
</table>
APPENDIX K DIFFERENCE IN SPECIFICATION OF SOFTUNE LOAD MODULE CONVERTERS (F2MS, F2HS) AND OLD LOAD MODULE CONVERTERS (LMC907A)

APPENDIX K describes Difference in Specification of SOFTUNE Load Module Converters (f2ms, f2hs) and Old Load Module Converter (lmc907a).

■ Preface

This document describes the difference in the specification of the SOFTUNE load module converters (f2ms, f2hs) and the old load module converter (lmc907a).

■ Difference in Command Lines

The difference in the command line specification at activation are given in Table K.a and Table K.b.

Table K.a Activation Format for Load Module Converter

<table>
<thead>
<tr>
<th></th>
<th>f2ms, f2hs</th>
<th>lmc907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>f2ms [option]... [file name]</td>
<td>lib907a file name [option]...</td>
</tr>
<tr>
<td>HEX</td>
<td>f2hs [option]... [file name]</td>
<td>lib907a @option file</td>
</tr>
</tbody>
</table>

Table K.b Option Specification for Load Module Converter

<table>
<thead>
<tr>
<th></th>
<th>f2ms, f2hs</th>
<th>lmc907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where to specify options</td>
<td>Any position allowed</td>
<td>After specification of file name</td>
</tr>
<tr>
<td>Specification of option file</td>
<td>-f option file</td>
<td>@ option file</td>
</tr>
<tr>
<td>Specification of two or more option files</td>
<td>Allowed</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>

■ Environmental Variables

The difference in environmental variables is given in Table K.c.

Table K.c Environmental Variables

<table>
<thead>
<tr>
<th></th>
<th>f2ms, f2hs</th>
<th>lmc907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification of working directory</td>
<td>TMP</td>
<td>TMP</td>
</tr>
<tr>
<td>Specification of installation path</td>
<td>FETOOL</td>
<td>-</td>
</tr>
<tr>
<td>Specification of display character code</td>
<td>FELANG</td>
<td>-</td>
</tr>
<tr>
<td>Specification of default option file store path</td>
<td>OPT907</td>
<td>-</td>
</tr>
</tbody>
</table>
## Options

The difference in the options is given in Table K.d.

<table>
<thead>
<tr>
<th>Table K.d  Load Module Converter Options</th>
<th>f2ms, f2hs</th>
<th>lmc907a</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-format output</td>
<td>use f2ms command</td>
<td>-FORM(-F) M</td>
</tr>
<tr>
<td>HEX-format output</td>
<td>use f2hs command</td>
<td>-FORM(-F) I</td>
</tr>
<tr>
<td>Change of output file name</td>
<td>-o</td>
<td>-OUTPUT(-O)</td>
</tr>
<tr>
<td>Change of output record length</td>
<td>The other tool (m2ms or h2hs) is used.</td>
<td>-LENGTH(-L)</td>
</tr>
<tr>
<td>Error information file output allowed/suppressed</td>
<td>-</td>
<td>-[NO]ERRORPRINT</td>
</tr>
<tr>
<td>Status code file output</td>
<td>-</td>
<td>-RETCODE(-RC)</td>
</tr>
<tr>
<td>Termination of option file</td>
<td>Not required</td>
<td>-END(E)</td>
</tr>
<tr>
<td>Help display</td>
<td>-help</td>
<td>-</td>
</tr>
<tr>
<td>Default option file input suppression</td>
<td>-Xdof</td>
<td>-</td>
</tr>
<tr>
<td>Activation message display/no display</td>
<td>-V/-XV</td>
<td>-</td>
</tr>
<tr>
<td>Warning-time return value change allowed/suppressed</td>
<td>-cwno/-Xcwno</td>
<td>-</td>
</tr>
<tr>
<td>Termination message display/no display</td>
<td>-cmsg/-Xcmsg</td>
<td>-</td>
</tr>
</tbody>
</table>
### Addition of Conversion Tools

The added SOFTUNE conversion tools are given in Table K.e.

**Table K.e  Load Module Conversion Tool List**

<table>
<thead>
<tr>
<th>SOFTUNE tool</th>
<th>Old 16-related tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute format load module -&gt; S format</td>
<td>f2ms</td>
</tr>
<tr>
<td>Absolute format load module -&gt; HEX format</td>
<td>f2hs</td>
</tr>
<tr>
<td>Absolute format load module -&gt; HEX16 format</td>
<td>f2es</td>
</tr>
<tr>
<td>Absolute format load module -&gt; HEX8 format</td>
<td>f2is</td>
</tr>
<tr>
<td>S format -&gt; S format(*)</td>
<td>m2ms</td>
</tr>
<tr>
<td>HEX format -&gt; HEX format(*)</td>
<td>h2hs</td>
</tr>
<tr>
<td>S format -&gt; Binary</td>
<td>m2bs</td>
</tr>
<tr>
<td>HEX format -&gt; Binary</td>
<td>h2bs</td>
</tr>
<tr>
<td>S format -&gt; HEX16 format</td>
<td>m2es</td>
</tr>
<tr>
<td>S format -&gt; HEX8 format</td>
<td>m2is</td>
</tr>
<tr>
<td>HEX16 format -&gt; S format</td>
<td>e2ms</td>
</tr>
<tr>
<td>HEX8 format -&gt; S format</td>
<td>i2ms</td>
</tr>
</tbody>
</table>

*: The file is sorted in ascending order of addresses, and the length of output data is changed.
APPENDIX L describes Object Information Comparison.

- Preface
  This document describes the procedure to examine whether the new and old absolute format files are exactly the same at the binary level when the source program for the old 16L assembler (asm907a) is ported to the new assembler (fasm907s).

- Outline of Comparison Procedure
  The figure below shows the comparison procedure.

  The absolute format object data created by the new and old tools are converted to the S format.

  ![Diagram of Comparison Procedure]

  The adjuster sorts the file in ascending order of addresses. The adjuster fixes the data count for 1 record.

  Comparison is made based on this list.

Appendix Figure L  Outline of Comparison Procedure
1. The absolute formal object data created with the new tool and the old tool is changed into S format.

2. S record is orthopedically operated using S format adjuster.

3. Operated orthopedically S format file is compared using a file comparison tool.

Note: Ignore the differences, if any, between the S0 record at the first line and the S7 record at the last line.

### Comparison Example

The absolute format file created by the old assembler is "old.abs". The absolute format file created by the new assembler is "new.abs".

```bash
>lmc907a old.abs -o old.mhx ; Converts old absolute format file to S format
>m2ms old.mhx -o old.ahx -ran 0xff00,0xffffff -S3 ; Sorts old S format
>f2ms new.abs -o new.mhx ; Converts new absolute format file to S format
>m2ms new.mhx -o new.ahx -ran 0xff00,0xffffff -S3; Sorts new S format
>fc old.ahx new.ahx; Compares files
```

The comparison results may indicate the following difference, but data with a record beginning with "S0" or "S7" should be ignored.

This record indicates a comment, so a match is not required.

**Output example**

```bash
>fc old.ahx new.ahx
***** old.ahx
S0080000746573743106
***** new.ahx
S00800054553543285
*****
```

Note: When an attempt is made to process data in the entire 16-MB memory space of the F2MC-16 family when using m2ms, the execution speed may slow down. In such a case, use the -ran option to split the file. The file should also be split when data to be compared is large.

### When Data Mismatched

The following shows an FC output example. This information is used to locate mismatched locations.

**Output example**

```bash
>fc old.ahx new.ahx
***** old.ahx
S31500FF0000000102032C37060708090A0B0C0D0E0F19
***** new.ahx
S31500FF0000000102030405060708090A0B0C0D0E0F73
*****
```
Appendixes

Obtaining Addresses from S Format

The S format has the following format:

<table>
<thead>
<tr>
<th>S3</th>
<th>Record length (1 byte)</th>
<th>Address (4 bytes)</th>
<th>Data</th>
<th>CS</th>
</tr>
</thead>
</table>

This indicates a data record. This indicates the data starting address. Checksum

*Note: Ignore records other than "S3" records.

The address of data with a mismatch can be obtained from the above output.

S31500FF0000000102030405060708090A0B0C0D0E0F73

This record data begins at address 0x00ff0000. From this, it is determined that there is mismatched data at addresses 0xff0005, and 0xff0006.

Locating Module Using Link Map

The module where the address obtained in S3 record is written is located using the link map.

At linkage, the -mlf option and the -als option must be specified.

The -mlf option creates detailed map information and the -als option creates the absolute assemble list.

[Output example]

```
>flnk907s main sub1 -AL 2 -mlf main.mpm -als -ro rom=0xff00/0xffff -ra rom=0/0xff
```

Detailed map information (main.mpm)

```
Section Mapping List
Module(s)
  1. main
  2. sub1

 S_Addr. -E_Addr. Size  Section  Type  Al  M.No  Sec.(Top 80)  
00FF0000-00FF000F  00000010  CONST P  R-I  02  1  data
00FF0010-00FF001F  00000010  CONST P  R-I  02  2  data
```

From the detailed map information, it is determined that the appropriate addresses (0xff05, 0xff06) are defined by the "main" module (module No. 1).
Locating Mismatched Locations

Mismatched locations are located from the absolute assemble list created at the time of a link.

Absolute assemble list (main.als)

<table>
<thead>
<tr>
<th>SN</th>
<th>LOC</th>
<th>OBJ</th>
<th>LLINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>.TITLE main</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>.PROGRAM main</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>.SECTION data,CONST,ALIGN=2</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>.DATA.B 0,1,2,3,44,55</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>.6,7,8,9,10,11,12,13,14,15</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>;data ends</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>.END</td>
</tr>
</tbody>
</table>
APPENDIX M  GUIDE TO CHANGING FUNCTION-CALL INTERFACE

If a function-call interface is changed, how to use selected registers and how to pass arguments are also changed. Thus, the program will not operate properly if the existing function-call interface (called "stack argument passing") and a new function-call interface (called "register argument passing") are mixed. Here are the methods and precautions for changing the program to a new function-call interface.

■ New Function-call Interface

The F²MC-16 FAMILY SOFTUNE supports the new function-call interface ("register argument passing") to pass the arguments of the function by registers (RW0 and RW1) as well as the existing function-call interface ("stack argument passing") to pass the arguments by the stack.

Using the function-call interface for "register argument passing", selected arguments are passed by registers, reducing code size and stack usage to improve the program execution speed.

However, the how to pass arguments and to use registers differ between the "register argument passing" and the "stack argument passing." If objects with different function-call interfaces are mixed, the program will not operate properly. Workbench, Assembler, Linker, and Librarian are designed to output warning messages or error messages when detecting a contradiction concerning the function-call interface.

For details about the new function-call interface, refer to the following manuals:

- F²MC-16 FAMILY SOFTUNE C COMPILER MANUAL
  FUNCTION CALL INTERFACE (register argument passing)
  Stack Frame (register argument passing)
  Argument (register argument passing)
  Argument Extension Format (register argument passing)
  Calling Procedure (register argument passing)
  Register (register argument passing)
  Return Value (register argument passing)

- F²MC-16 FAMILY SOFTUNE ASSEMBLER MANUAL
  -rp, -Xrp
  .REG_PASS instruction

- F²MC-16 FAMILY SOFTUNE LINKAGE KIT MANUAL
  Mixing of objects with different function-call interfaces in Linker
  Option (-rp) to specify load module for "register argument passing"
  Option (-Xrp) to specify load module for "stack argument passing"
  Mixing of objects with different function-call interfaces in Librarian
Option (-rp) to specify output of library for "register argument passing"
Option (-Xrp) to specify output of library for "stack argument passing"

Changing of Function-call Interface

To change the function-call interface from "stack argument passing" to "register argument passing," use the following procedure:

Correct the Assembler source (e.g. startup.asm).  (See Correcting assembler source)

Is Workbench used for Build?

Change the function-call interface of the project to "register argument passing" and rebuild. (See Changing by Workbench.)

Specify the -rp option for Compiler, Assembler, Linker and Librarian, and rebuild. (See Changing without using Workbench.)

Appendix Figure M.1  Changing of Function-call Interface

Details of Changing of Function-call Interface

Correcting assembler source

To change the function-call interface from "stack argument passing" to "register argument passing," use the following procedure to correct the assembler source:

1. Description of .REG_PASS pseudo-instructions

When the -rp option is specified, Assembler outputs the object for "register argument passing." However, only with the -rp option, Assembler cannot distinguish between the input assembler source for "register argument passing" and the input assembler source for "stack argument passing."

The .REG_PASS pseudo-instructions indicates the assembler source for "register argument passing."

When the -rp option is specified, if Assembler assembles the assembler source where the .REG_PASS pseudo-instructions is not described, Assembler will output an error.

When changing the assembler source for "stack argument passing" to the assembler source for "register argument passing," describe the .REG_PASS pseudo-instructions.

The .REG_PASS pseudo-instructions can be described anywhere preceding the .END pseudo-instructions.
2. Changing the how to pass arguments

If a function with arguments is defined in the assembler source and is called, change the how to pass arguments to "register argument passing."

The same correction is also required when defining and calling the function with arguments by the assembler description function (asm statement) of C Compiler.

When changing the how to pass arguments to "register argument passing," follow the tables in F2MC-16 FAMILY SOFTUNE C COMPILER Manual "Arguments (register argument passing)."

[Correction example of definitions of functions with arguments]

(stack argument passing)                                            (register argument passing)

_func: _func:
    LINK #0                                                              LINK #0
    MOVW A, @RW3+4                                                      MOVW RW4, @RW3+4
    ADDW A, @RW3+6                                                      MOVW A, RW0
    ADDW A, @RW3+8                                                      ADDW A, RW1
    MOVW _var, A                                                        ADDW A, RW4
    :                                                                  : 

[Correction example of calling functions with arguments]

(stack argument passing)                                            (register argument passing)

:                                                                  :
    MOVW A, #3                                                        MOVW A, #3
    PUSHW A,                                                           PUSHW A,
    MOVW A, #2                                                        MOVW A, #2
    PUSHW A,                                                           MOVW RW1, A
    MOVW A, #1                                                        MOVW A, #1
    PUSHW A,                                                           MOVW RW0, A
    CALL _func                                                         CALL _func
    ADDSP #6                                                           POPW AH
    :                                                                  :
(Supplementary explanation)

Using the assembler source both for "stack argument passing" and for assembler source for "register argument passing"

Using the predefined macro "__REG_PASS__" create an assembler source both for "stack argument passing" and for "register argument passing".

When the predefined macro "__REG_PASS__" is assembled with the -rp option, set 1.
When the predefined macro "__REG_PASS__" is assembled without the -rp option, set 0.

[Example of description of predefined macro "__REG_PASS__"]

```
.PROGRAM_func
.TITLE_func
#if __REG_PASS__
; register argument passing
 .REG_PASS
#endif
```

Note:

With the function-call interface for "register argument passing", return values from the functions of registers RW0 and RW1 are not guaranteed. Do not use these registers across function calls.
Appendixes

- Changing by Workbench

Change the function-call interface of the project to "stack argument passing."

When the function-call interface for "register argument passing" is selected, the -rp option is specified automatically when Compiler, Assembler, Linker, and Librarian start up.

(1) Changing the setting of the current project

Change the function-call interface of the project to "register argument passing" in the [Setup Project] dialog of the project.

(2) When changing the setting of a new project

In the [Create] dialog of the project, "stack argument passing" is selected by default. Change the function-call interface to "register argument passing."

Appendix Figure M.2 Setup Dialog of Project
Note:

SOFTUNE REALOS supports only "stack argument passing." If the REALOS project is selected, only the function-call interface for "stack argument passing" can be selected. There are some restrictions in the operation of the debugger when "register argument passing" is selected as a function call interface. These restrictions become the restrictions when the function to which it is called from the function with the argument, and the LINK/UNLINK instruction is not output by optimizing the compiler is debugged.

- Operation of step-out (GO/RETURN command)
  Since the parent function can no longer be obtained, execution of the program does not stop normally.

- Operation of call stack (SHOW CALLS command)
  Since the parent function can no longer be obtained, the subsequent parent functions cannot be displayed correctly.

- Operation of the UP command
  Since the parent function can no longer be obtained, the scope can not be correctly moved to the subsequent parent functions.

 oasis Changing without using Workbench

(1) Changing without using Workbench
When not using Workbench, specify the -rp option for Compiler, Assembler, Linker, and Librarian.

(2) Changing during compiling and assembling
Declare the prototypes of all functions. If a function whose prototype is not declared is called, the program is not guaranteed to operate.
Appendixes

The area where argument registers are saved is not concatenated with the area where parameters are passed by the stack. A program designed to expect such concatenation does not operate properly. If a program of this type exits, it must be corrected.

Similar to the conventional function-call interface, all arguments of the variable arguments are passed by the stack and need not to be corrected.

(3) Changing during linking and making library

The following C library is added for "register argument passing."
When using "register argument passing," link the following library with ".rp."

- lib/907/lib9*.rp.lib : 48 files

Warning and Error Messages

When a contradiction concerning the function-call interface is detected, Workbench, Assembler, Linker and Librarian output the following warning or error messages.

❖ Workbench warning messages
- When adding of project

| I0227W | Projects with different function-call interfaces exist in the workspace. Objects created by the project with different function-call interfaces cannot be used together. Continue processing? |
|        | YES: Add a project. NO: Do not add a project. |
|        | YES: Set the dependencies between projects. NO: Do not set the dependencies between projects. |
|        | YES: Change the function-call interface. NO: Do not change the function-call interface. |

❖ Assembler error message

| E4718A | Different calling interface of a function |
|        | This error message is outputted under the conditions shown in Appendix Table M-1. |
If these errors occur, correct the specifying of the -rp option or the description of the .RE_PASS pseudo-instructions for the assembler source, referring to Appendix Table M-1.

- **Linker error message**

| E4313L | The module is different calling interface of a function. (file name) |

This error message is outputted under the conditions shown in Table M-2.

### Table M-2 Relationship between Input Object and -rp Option

<table>
<thead>
<tr>
<th>Specification of -rp option</th>
<th>Function-call interface for input object</th>
<th>Provided</th>
<th>Unprovided</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;register argument passing&quot;</td>
<td>Normal combination for &quot;register argument passing&quot;</td>
<td>The error (E4313L) is outputted.</td>
</tr>
<tr>
<td></td>
<td>&quot;stack argument passing&quot;</td>
<td>Normal combination for &quot;stack argument passing&quot;</td>
<td>No error is output.</td>
</tr>
</tbody>
</table>

If these errors occur, correct the specifying of the -rp option or the function-call interface for the input object, referring to Appendix Table M-2.

- **Librarian error message**

| E4410U | The module is different calling interface of a function. (file name) |

This error message is outputted under the conditions shown in Table M-3.
Appendixes

Table M-3  Relationship between Input Object and -rp Option

<table>
<thead>
<tr>
<th>Specifying of -rp option</th>
<th>Function-call interface for input object and library to be edited</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;register argument passing&quot;</td>
</tr>
<tr>
<td>Provided</td>
<td>Normal combination for &quot;register argument passing&quot;</td>
</tr>
<tr>
<td></td>
<td>No error is outputted.</td>
</tr>
<tr>
<td>Unprovided</td>
<td>The error (E4410U) is outputted.</td>
</tr>
<tr>
<td></td>
<td>Normal combination for &quot;stack argument passing&quot;</td>
</tr>
<tr>
<td></td>
<td>No error is outputted.</td>
</tr>
</tbody>
</table>

If these errors occur, correct the specifying of the -rp option or the function-call interface for the input object and library to be edited, referring to Table M-3.

---

Note:

Objects with different function-call interfaces cannot be linked.

Fujitsu provides only the following object function-call interfaces for "stack argument passing".

- Monitor Debugger
- SOFTUNE REALOS
- REALOS debug module (R_D_dbg.obj)
The index follows on the next page.
This is listed in alphabetic order.
Index

A
Absolute Assemble List Creation Function...........387
Absolute Assemble List File............................179
Absolute assemble list output directory specification
(-alout)........................................................149
Absolute assemble list output inhibit specification
(-Xals) .......................................................152
absolute assemble list output module ..................151
Absolute assemble list output specification (-als).................................................................150
Absolute format assemble list ..............................168
Absolute format load module output specification (-a).....................................................................97
Access to low power consumption mode control
register (LPMCR) using assembler language
........................................................................383
Access to low power consumption mode control
register (LPMCR) using C language,
not standby control macro instructions......386
Adding (registering) a module(-a).........................217
Addition of Conversion Tools ...............................397
Adjust option (-adjust) ........................................264
Alleviation of Restrictions ..............................391, 394
Allocation/link options.............................................93
-alsf ......................................................................151
Assemble source list ............................................184
Automatic allocation specification (-AL) ...............114
Automatically Locating Sections ............................67

B
Byte unit check.....................................................378

C
Changing an Output File name (-o)......................257
Changing of Function-call Interface .....................403
Changing the format of a list file.........................55
Character code of the filename..........................12
Check on the total size of the DIR-attribute sections ...........................................................................85
Check User-unspecified Section (-check_section) .................................................................135
Checking method ..................................................377
Checking Procedures ...........................................375
Checking the contents of a library file ............205, 210
Checking the contents of a library file (-c)...........228
-cmsg option ..........................................................29
Command line format..............................................5
Common format ..................................................342
common option of object format converter...........256
Comparison Example..........................................399
Configuration of a list file..................................238
Control on combining and locating sections .........54
Control on searching libraries .............................54
CPU Information ..................................................86
CPU Information File............................................86
CPU Information File Specification (-cif) .......124, 233
Creating a backup file (-b).................................226
Creating a group of sections ...............................57
Creating a new library file ..............................205, 206
Cross-reference list..............................................188
-cwno option.......................................................29

D
Data record (HEX8/HEX16/HEX32)......................344
Debug information delete specification (-Xg).........96
Debug information output specification (-g) ...........95
Default library retrieval inhibit specification (-nd) .............................................................................120
Default Option File ..........................................46
Deleting a module (-d) ........................................219
Deleting debugging information .......................205, 209
Destination of Warning Messages.....................369
Details of Changing of Function-call Interface ....403
Detectable Standby Mode Transition Instructions .............................................................................380
Determining location addresses .......................68, 70
Devices applicable to the cautions...............383
Difference in Command Lines.........................388, 392, 395
Direct page..........................................................83
Displaying identifier name when outputting list......11
Displaying the contents of a library file ............205, 210
-dl.................................................................101
Dummy setting of external symbol values (-df) .................................................................................122

E
e2ms (Converting a HEX Format File into the S
Format) ...............................................................303
Editing a library file...........................................205, 206
End code value and end status.........................7
INDEX

Extracting a module (-x) ........................................ 220
Extracting a module from a library file .......... 205, 208

F
-f option ........................................................ 29
f2es (Absolute Format Load Module into the HEX16 format) ........................................ 274
f2hs (Converting an Absolute Format Load Module into the HEX32 format) .......... 272
f2is (Absolute Format Load Module into the HEX8 format) ........................................ 273
f2ms (Converting an Absolute Format Load Module into the S format) ....................... 271
FELANG .............................................................. 13, 15
Fetch unit check ................................................. 377
FETOOL .............................................................. 13, 16
Forced termination .............................................. 6
Function that sets automatically internal ROM/RAM area (-set_rora) ..................... 126
Functions of a Linker .............................................. 53
Functions of the Format Adjuster ....................... 277

G
General Description of Check Tool ....................... 368

H
Handling variables with initial values .......... 198, 199
Header format .................................................. 181
Help message .................................................... 10
-help option .................................................... 29
How to Register in the Startup of the SOFTUNE Workbench Tools and How to Use .......... 373
How to Register to the Workbench Customize Build Function .................................. 371
How to Startup .................................................. 369
How to Use on the Command Line ....................... 370

I
i2ms (Converting a HEX Format File into the Motrola S Format) .................................. 302
Indicating identifiers ........................................ 11
Inheriting debugging information ..................... 55
Inhibit check for presence of user-unspecified section (-xcheck_section) ............. 137
Inhibiting Check for Presence of Debug Data (-NC10302LIB) ................................. 125
Inhibiting the creation of a backup file (-Xb) ........................................ 227
Inhibiting the search for a library ..................... 58
INDEX

**S**  
S5 type (record to manage the number of records) .................................................. 356  
S7 type (terminator record) .......................................................... 357  
S8 type (terminator record) .......................................................... 358  
S9 type (terminator record) .......................................................... 359  
Section Allocation Detailed Information List File ............................................. 193  
Section allocation order/address specification (-sc) .................................. 108  
Section combination attribute .......................................................... 60  
Section detail map list ......................................................... 168  
Section detail map list name specification (-mlf) .................................................. 164  
Section detail map list output inhibit specification (-Xml) ................................. 165  
Section detail map list output specification (-ml) ............................................. 163  
Section group specification (-gr) ..................................................... 111  
Section identification .............................................................. 61  
Section information list ............................................................ 186  
Section location attribute .......................................................... 60  
Section name .............................................................. 60  
Section types and location destinations .................................................. 70  
Section-placed-area check specification (-check_locate) ................................... 131  
Section-placed-area check suppression specification (-Xcheck_locate) .............. 134  
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 .................................... 366  
Specification of Fetch Unit Checks (-Wf) .................................................. 142  
Specification of Prohibiting Standby Mode Transition Instruction Check (-check_SCF) .................................................. 140  
Specification of Standby Mode Transition Instruction (-check_SCF) ......... .................................................. 138  
Specification of the number of digits in the list line (-pw) .......................................................... 103  
Specification of the number of lines on one list page (-pl) ......................... 104  
Specification related to output messages .............................................. 93  
Specification to Continue in the Option File ..................................... 43  
Specifies to prevent the internal ROM/RAM area from being set automatically .... 86

**P**  
Padding (-p) ........................................................................ 259  
Parameter ............................................................................. 22  
Precautions on specifying options .......................................................... 244  
Precautions on the required disk space .......................................................... 244  
Precautions on the sections to be transferred from ROM to RAM ................. .................................................. 81  
Preface ............................................................................. 387, 392, 395, 398  
Processing when library files are individually specified ............................... 78

**Q**  
question and answer on using object tool .................................................. 307

**R**  
RAM area specification (-ra) ..................................................... 107  
Register bank area specification (-rg) ..................................................... 112  
Relative assemble list input directory specification (-alin) ................................... 145, 147, 148  
Relative format load module output specification (-r) .............................................. 98  
replacing (registering) a module ......................................................... 218  
Restrictions ............................................................................. 382  
Restrictions on a converter ............................................................. 306  
Restrictions on a librarian ................................................................ 244  
Restrictions on the object file format ..................................................... 196  
Retrieval library file specification (-l) ..................................................... 116  
ROM and RAM Areas .................................................................. 79  
ROM area specification (-ro) .................................................................. 106  
ROM/RAM and ARRAY list output inhibit ..................................................... 155  
ROM/RAM and ARRAY list output module specification (-alrf) ......................... 154  
ROM/RAM and ARRAY list output specification (-alr) .................................................. 153  
ROM/RAM and ARRAY list symbol and address display position specification (-na,-an) .................................................. 156  
ROM/RAM and ARRAY lists ............................................................ 182  
ROM/RAM Areas Names .............................................................. 86

**S**  
S Format Adjuster .............................................................. 252  
S Format Binary Converter ............................................................ 252  
S format binary converter ad S format adjuster, cautionary information concerning .................................................. 306  
S0 type (header record) .............................................................. 352  
S1 type (data record 2-byte address) ..................................................... 352  
S2 type (data record 3-byte address) ..................................................... 353  
S3 type (data record 4-byte address) ..................................................... 354
INDEX

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

Specified Output Library for Register argument Passing (-rp)..................................................234

Specified Output Library for Stack Argument Passing (-Xrp)..........................................................235

Specified Output Load Module for Register argument Passing (-rp)..............................................144

Specified Output Load Module for Stack argument Passing (-Xrp)..................................................145

Specifying a library file to be searched.................58

Specifying a library file to be searched for each symbol.........................................................58

Specifying a library file to be searched.............72

Specifying a list file name...........................................55

Specifying a path to search a library...............58

Specifying a Target CPU (-cpu)..........................232

Specifying an entry address..................................59

Specifying an Output HEX format (-I16/-I20/-I32).................................................................280

Specifying an output HEX format (-I16/-I20/-I32)........................................................................285

Specifying an output load module file name........55

Specifying an output S format (-S1/-S2/-S3)....................280, 283

Specifying Changes to the Starting Address (-ST) ........................................................................286

Specifying comment in the Option File.................44

Specifying Display of End Message (-cmgs).............36

Specifying End Code to 0 When Warning is Issued (-cwno)..........................................................39

Specifying End Code to 1 When Warning is Issued (-cwno)..........................................................38

Specifying Help Message (-help)..........................33

Specifying input object files..................................55

Specifying not to create a map list file (-Xm)............................................................................291, 297

Specifying not to output a list file (-Xm)..................222

Specifying not to output debugging information (-Xg) .................................................................231

Specifying not to Output Start Address Record (-Xentry)............................................................269

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 (-Xcmgs) ............................................................37

Specifying that Check on the Total Size of the DIR-attribute Sections (-check_dirsize).............146

Specifying that Check on the Total Size of the DIR-attribute Sections be inhibited (-Xcheck_dirsize).................................................................................................147

Specifying the data length in an output record (-len)......................................................................281

Specifying the inhibition of the split mode (-Xsp)..........................................................................281

Specifying the number of columns per line of a list (-pw).............................................................225

Specifying the number of lines per page of a list (-pl)....................................................................224

Specifying the order of locating sections and the location addresses.................................57

Specifying the output format..................................55

Specifying the output range (-ran).................................................................................................280

Specifying to create a map list file (-m)..................291, 295

Specifying to output a list file (-m)........................221

Specifying to output debugging information (-g)...........................................................................230

Specifying to Output Start Address Record (-entry).................................................................267

Specifying Version Number/Message Output (-V).........................................................................34

Standby control macro instructions..........................384

Start address output inhibit option (-Xentry)........264

Start address output option (-entry)......................264

Start linear address record (HEX32)...............349

Start segment address record (HEX16/HEX32)........347

Startup message.........................................................8

Startup message and the -V option.....................8

Support for creating a ROM......................................57

Support Range of Linkage Kit.................................4

Suppression of Version Linkage Kit......................35

Suppression of Version Number/Message Output (-XV)..............................................................347

symbol list part......................................................178

Synopsis of option.................................................22

T

Target CPU specification (-cpu)..........................123

TMP (Work directory).................................................13

TMP (work directory).................................................14

Types of characters consisting of identifiers........11

Types of load module converters...........................252
Types of other converters ........................................ 252
Types of section contents ........................................ 60

U
User-specified-area check specification  
(-check_rora) .............................................................. 128
User-specified-area check suppression specification  
(-Xcheck_rora) .......................................................... 130
Using the sections to be transferred from ROM to RAM .................................................. 80
Using the wild card ..................................................... 197

V
-V option ................................................................. 29

W
Warning and Error Messages ................................. 408
Warning Message ......................................................... 376
Warning message output level specification (-w)  
................................................................................. 105
When Data Mismatched ........................................... 399
When Warning Messages Are Not Output Even When 
the Standby Transition Instruction is Detected 
................................................................................. 379

X
-Xair ................................................................. 155
-Xcmsg option ............................................................ 29
-Xcwno option ............................................................. 29
-Xdof option .............................................................. 29
-XV option ................................................................. 29