

To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

Notice

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

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

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

RA78K SERIES ASSEMBLER PACKAGE

for OPERATION

USER'S MANUAL

NEC

RA78K SERIES ASSEMBLER PACKAGE

for OPERATION

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.

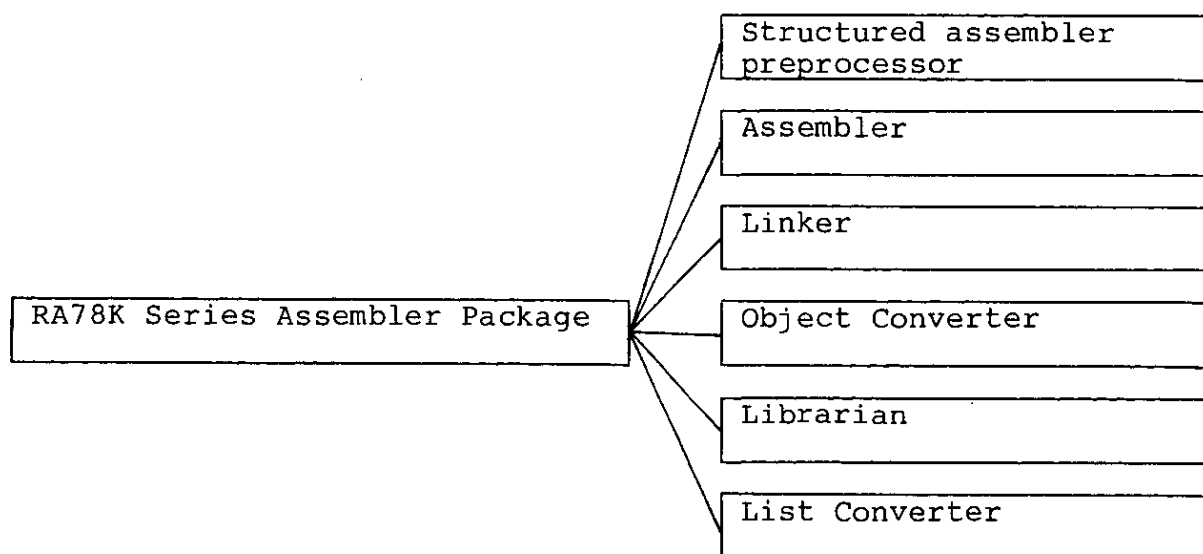
NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or of others.

INTRODUCTION

This manual is designed to facilitate correct understanding of the functions and operation of each program contained in the RA78K Series Assembler Package (hereinafter referred to as "this package or the package").

This manual does not cover the language related to the 78K series such as directives and source program formats. Therefore, before using this manual, read through the RA78K Series Assembler Package User's Manual for Language (hereinafter referred to as "the Language Manual").

Descriptions related to the RA78K/I, RA78K/II, and RA78K/III in this manual are applicable to the version numbers V3.0 and upwards of the RA78K series assembler packages. For application examples in the manual, programs for the 78K/III series microcomputers have been used.



[Target Devices]

The software of the following microcomputers can be developed with this package:

| Package name | Target device |
|--------------|---------------------------------------------------------------------------------------------------|
| RA78K/0 | 78K/0 uPD78012, uPD78014 |
| RA78K/I | 78K/I uPD78112 uPD78134, uPD78134A, uPD78136, uPD78138 |
| RA78K/II | 78K/II uPD78120, uPD78212, uPD78213, uPD78214 uPD78220, uPD78224 uPD78233, uPD78234 |
| RA78K/III | 78K/III uPD78310, uPD78312 uPD78310A, uPD78312A uPD78320, uPD78322 uPD78330, uPD71334 |
| RA78K/VI | 78K/VI uPD78600, uPD78602, uPD78603, uPD78604 |

[Readers of Manual]

Although this manual is intended for those who are familiar with the functions of the microcomputer subject to software development and the method of describing source programs, the manual can also be used by those who use an assembler program for the first time.

[Organization of Manual]

This manual consists of the following 11 chapters and appendixes:

Chapter 1 - General

Outlines the functions of this package including the role of the package in microcomputer development.

Chapter 2 - Product Overview

Describes the program file names offered in this package and the operating environment of each program.

Chapter 3 - Execution of Assembler Package

Describes the program development procedures through execution of the respective programs in this package by using sample programs. Since this chapter places] emphasis on how to actually operate each program, those who wish to operate this package should start reading the manual from this chapter.

Chapters 4 through 8 - Assembler, Linker, Object Converter,
Librarian, and List Converter

Details the functions and operations of the respective programs (Assembler, Linker, Object Converter, Librarian, and List Converter) of this package. These chapters are the most important for you to actually operate each program in this package.

Chapter 9 - Output Lists of Programs

Explains the formats of various lists to be output by the respective programs of this package.

Chapter 10 - Utilization of Assembler Package

Introduces some measures recommended for effective utilization of this package.

Chapter 11 - Error Messages

Describes error messages to be output by the respective programs of this package.

Appendixes - Contain examples of sample program lists, hints on use, lists of respective program options, and a list of librarian subcommands.

[Recommended Usage of Manual]

For those who wish to actually operate this package: First, read Chapter 3, Execution of Assembler Package of this manual.

For those who have a general understanding of assembler programs or those who have read the RA78K Series Assembler Package User's Manual for Language: You may skip Chapter 1, General of this manual.

Utilize various lists in the respective appendixes after you have familiarized yourself with the operation of each program. For quick reference, use the respective program options in Appendix C and the error message lists in Chapter 11.

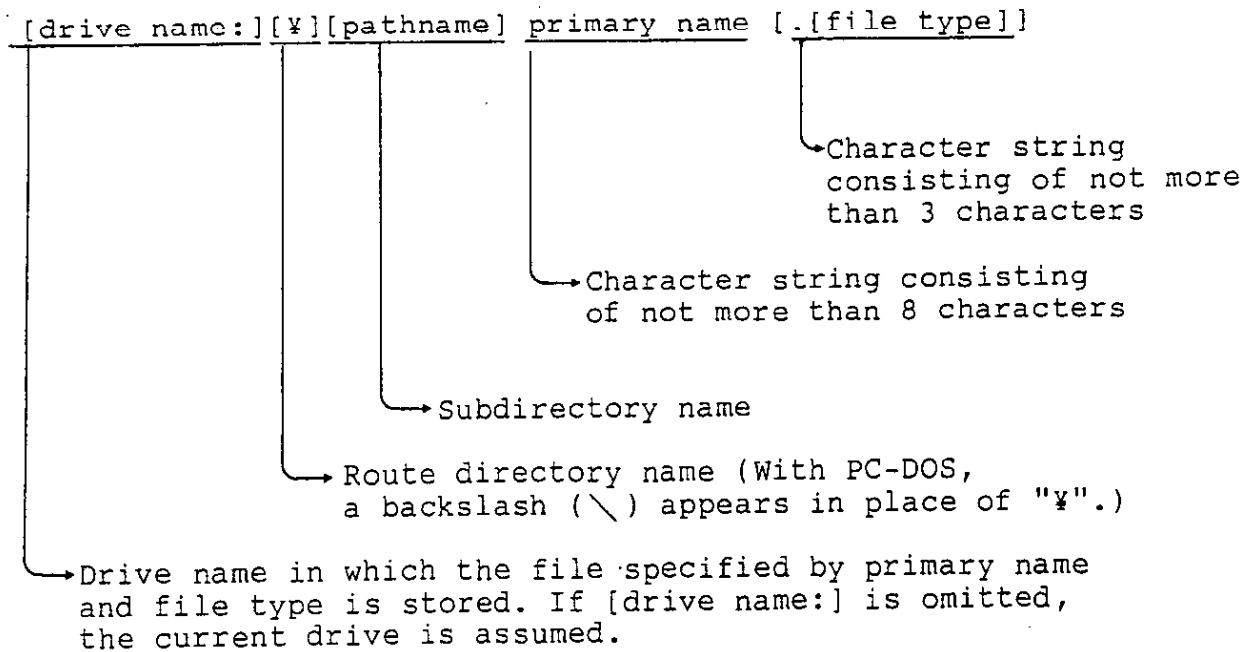
[Symbols and Abbreviations]

The following symbols and abbreviations are used in this manual:

| <u>Symbol</u> | <u>Meaning</u> |
|---------------|------------------------------------------------------------------------|
| ... | Continuation (repetition) of data in the same format |
| [] | Parameter(s) in brackets can be omitted. |
| ' ' | Characters enclosed in ' ' (single quotes) must be input as is. |
| " " | Characters enclosed in " " (double quotes) must be input as is. |
| <u> </u> | Underlined portion of a command description must be input by the user. |
| □ | Only one blank character must be input. |
| △ | One or more blank characters must be input. |
| ▲ | Zero or more blank characters may or may not be input. |
| ⋮ | This part of the program description is omitted. |
| / | Delimiter or separator |
| ␣ | Carriage Return input |
| \ | Backslash |
| | With PC-DOS, a backslash is used in place of "¥". |

[Conventions of Filename Specification]

(1) Disk type file specification



Example:

- Note:
1. No blank character (space) can be specified before or after ":" (colon), "." (period), or "¥".
 2. Uppercase and lowercase letters are not distinguished from each other.
 3. With PC-DOS, a backslash (\) is used in place of "¥".

(2) Device type file specification

Any of the following logical devices can be specified:

| Logical device | Description |
|----------------|----------------------------------------|
| CON | Output to the console |
| PRN | Output to the printer |
| AUX | Output to an auxiliary output device. |
| NUL | Dummy output (Nothing will be output.) |

- * IBM PCTM, IBM PC/XTTM, IBM PC/ATTH, and PC-DOSTM are trademarks of IBM Corp.
- ** MS-DOSTM is a trademark of Microsoft Corp.
- *** V30TM is a trademark of NEC Corporation.
- **** 80286TM and 8086TM are trademarks of Intel Corp.

TABLE OF CONTENTS

| | <u>Page</u> |
|-------------------------------------------------------------------------|-------------|
| CHAPTER 1 GENERAL | 1-1 |
| 1.1 Assembler Overview | 1-1 |
| 1.1.1 What is an assembler? | 1-1 |
| 1.1.2 What is a relocatable assembler? | 1-6 |
| 1.2 Functional Outline of Assembler Package | 1-9 |
| 1.2.1 Creation of source module file with editor | 1-10 |
| 1.2.2 Structured assembler preprocessor | 1-11 |
| 1.2.3 Assembler | 1-12 |
| 1.2.4 Linker | 1-13 |
| 1.2.5 Object converter | 1-14 |
| 1.2.6 Librarian | 1-15 |
| 1.2.7 List converter | 1-16 |
| 1.2.8 Source debugger | 1-17 |
| 1.3 Memory Maps | 1-18 |
| 1.4 Reminders Before Program Development | 1-23 |
| 1.4.1 Number of files that can be input to Linker | 1-23 |
| 1.4.2 Restriction on number of symbols | 1-23 |
| 1.4.3 Maximum performance characteristics of assembler package | 1-24 |
| 1.5 Features of Assembler Package | 1-25 |
| CHAPTER 2 PRODUCT OVERVIEW | 2-1 |
| 2.1 Contents of Product | 2-1 |
| 2.2 Form of Supplied File Medium | 2-3 |
| 2.3 System Configuration | 2-3 |
| CHAPTER 3 EXECUTION OF ASSEMBLER PACKAGE | 3-1 |
| 3.1 Before Executing the Assembler Package | 3-1 |
| 3.1.1 Confirming the contents of the supplied disk | 3-1 |
| 3.1.2 Sample program | 3-1 |
| 3.2 Procedure for Assembler Package Execution | 3-5 |
| 3.3 Summary of Assembler Package Execution Procedure | 3-11 |

| | |
|---------------------------------------------------------------------------------------------------------|------|
| CHAPTER 4 ASSEMBLER | 4-1 |
| 4.1 Input/Output Files of Assembler | 4-1 |
| 4.2 Assembler Functions | 4-3 |
| 4.3 How to Start Up the Assembler | 4-4 |
| 4.3.1 Starting up the assembler | 4-4 |
| 4.3.2 Execution start and end messages | 4-6 |
| 4.4 Assembler Options | 4-8 |
| 4.4.1 Types of assembler options | 4-8 |
| 4.4.2 Priority of assembler options | 4-10 |
| 4.4.3 Description of each assembler option | 4-12 |
| (1) Option for processor type specification (-C) | 4-13 |
| (2) Options for object module file output specification (-O/-NO) | 4-17 |
| (3) Options for forced object module file output specification (-J/-NJ) | 4-19 |
| (4) Options for debug information output specification (-G/-NG) | 4-21 |
| (5) Options for symbol name length specification (-S/-NS) | 4-23 |
| (6) Options for symbol name uppercase/lowercase specification (-CA/-NCA) | 4-25 |
| (7) Option for Include file read path specification (-I) | 4-27 |
| (8) Options for assembly list file output specification (-P/-NP) | 4-29 |
| (9) Options for assembly list file information specification (-KA/-NKA, -KS/-NKS, -KX/-NKX) | 4-31 |
| (10) Options for assembly list file format specification (-LW, -LL, -LH, -LT, -LF/-NLF) | 4-39 |
| (11) Options for error list file output specification (-E/-NE) | 4-57 |
| (12) Option for parameter file specification (-F) | 4-59 |
| (13) Option for temporary file creating path specification (-T) | 4-61 |

| | |
|--------------------------------------------------------------------------|------|
| (14) Options for HELP message display specification (--) | 4-63 |
| CHAPTER 5 LINKER | 5-1 |
| 5.1 Input/Output Files of Linker | 5-1 |
| 5.2 Linker Functions | 5-3 |
| 5.3 Memory Spaces and Memory Areas | 5-4 |
| 5.3.1 Memory spaces | 5-4 |
| 5.3.2 Memory areas | 5-5 |
| 5.4 Merging the Input Segments | 5-8 |
| 5.4.1 Merge types of segments | 5-8 |
| 5.4.2 Rules for determining the merge type | 5-8 |
| 5.4.3 Merging the segments | 5-10 |
| 5.4.4 Segment merging method by merge type | 5-11 |
| 5.5 Determining the Location Addresses of Segments | 5-17 |
| 5.5.1 Location types of segments | 5-17 |
| 5.5.2 Rules for determining the location type | 5-21 |
| 5.5.3 Rules for determining the segment location addresses | 5-23 |
| 5.5.4 Procedure for locating segments | 5-25 |
| 5.6 Link Directives | 5-26 |
| 5.6.1 Directive file | 5-26 |
| 5.6.2 Memory directive | 5-28 |
| 5.6.3 Segment location directive | 5-34 |
| 5.7 How to Start Up the Linker | 5-41 |
| 5.7.1 Starting up the linker | 5-41 |
| 5.7.2 Execution start and end messages | 5-43 |
| 5.8 Linker Options | 5-45 |
| 5.8.1 Types of linker options | 5-45 |
| 5.8.2 Priority of linker options | 5-47 |
| 5.8.3 Description of each linker option | 5-49 |
| (1) Options for load module file output specification (-O/-NO) | 5-50 |
| (2) Options for forced load module file output specification (-J/-NJ) | 5-52 |
| (3) Options for debug information output specification (-G/-NG) | 5-54 |

| | |
|---------------------------------------------------------------------------------------------------------------|------|
| (4) Options for stack-reserving symbol creation specification (-S/-NS) | 5-56 |
| (5) Option for directive file specification (-D) | 5-59 |
| (6) Options for link list file output specification (-P/-NP) | 5-61 |
| (7) Options for link list file information specification (-KM/-NKM, -KD/-NKD, -KP/-NKP, -KL/-NKL) | 5-63 |
| (8) Options for link list file format specification (-LL, -LF/-NLF) | 5-75 |
| (9) Options for error list file output specification (-E/-NE) | 5-80 |
| (10) Option for library file specification (-B) .. | 5-82 |
| (11) Option for library file read path specification (-I) | 5-84 |
| (12) Option for parameter file specification (-F) | 5-86 |
| (13) Option for temporary file creation path specification (-T) | 5-88 |
| (14) Option for HELP message display specification (--) | 5-90 |
| CHAPTER 6. OBJECT CONVERTER | 6-1 |
| 6.1 Input/Output Files of Object Converter | 6-1 |
| 6.2 Object Converter Functions | 6-3 |
| 6.3 How to Start Up the Object Converter | 6-6 |
| 6.3.1 Starting up the object converter | 6-6 |
| 6.3.2 Execution start and end messages | 6-8 |
| 6.4 Object Converter Options | 6-10 |
| 6.4.1 Types of object converter options | 6-10 |
| 6.4.2 Description of each object converter option | 6-11 |
| (1) Options for HEX-format output module file output specification (-O/-NO) | 6-12 |
| (2) Options for symbol table file output specification (-S/-NS) | 6-15 |
| (3) Options for object code output sequence specification (-R/-NR) | 6-18 |

| | |
|--------------------------------------------------|------|
| (4) Option for object code fill specification | |
| (-U) | 6-20 |
| (5) Options for error list file output | |
| specification (-E/-NE) | 6-23 |
| (6) Option for parameter file specification | |
| (-F) | 6-25 |
| (7) Option for HELP message display | |
| specification (--) | 6-27 |
| CHAPTER 7. LIBRARIAN | 7-1 |
| 7.1 Input/Output Files of Librarian | 7-1 |
| 7.2 Librarian Functions | 7-3 |
| 7.3 How to Start Up the Librarian | 7-5 |
| 7.3.1 Starting up the librarian | 7-5 |
| 7.3.2 Execution start and end messages | 7-9 |
| 7.4 Librarian Options | 7-11 |
| 7.4.1 Types of librarian options | 7-11 |
| 7.4.2 Description of each librarian option | 7-12 |
| (1) Options for list file format specification | |
| (-LW, -LL, -LF/-NLF) | 7-13 |
| (2) Option for temporary file creation path | |
| specification (-T) | 7-19 |
| (3) Option for HELP message display | |
| specification (--) | 7-21 |
| 7.5 Subcommands | 7-23 |
| 7.5.1 Types of subcommands | 7-23 |
| 7.5.2 Description of each subcommand | 7-24 |
| (1) CREATE subcommand | 7-25 |
| (2) ADD subcommand | 7-27 |
| (3) DELETE subcommand | 7-29 |
| (4) REPLACE subcommand | 7-31 |
| (5) PICK subcommand | 7-33 |
| (6) LIST subcommand | 7-35 |
| (7) HELP subcommand | 7-37 |
| (8) EXIT subcommand | 7-38 |

| | |
|-------------------------------------------------------------------------------|------|
| CHAPTER 8. LIST CONVERTER | 8-1 |
| 8.1 Input/Output Files of List Converter | 8-1 |
| 8.2 List Converter Functions | 8-3 |
| 8.3 How to Start Up the List Converter | 8-6 |
| 8.3.1 Starting up the list converter | 8-6 |
| 8.3.2 Execution start and end messages | 8-8 |
| 8.4 List Converter Options | 8-10 |
| 8.4.1 Types of list converter options | 8-10 |
| 8.4.2 Description of each list converter option | 8-11 |
| (1) Option for object module file input specification (-R) | 8-12 |
| (2) Option for load module file input specification (-L) | 8-14 |
| (3) Option for absolute assembly list file output specification (-O) | 8-16 |
| (4) Options for error list file output specification (-E/-NE) | 8-18 |
| (5) Option for parameter file specification (-F) | 8-20 |
| (6) Option for HELP message display specification (--) | 8-22 |
| CHAPTER 9. OUTPUT LISTS OF PROGRAMS | 9-1 |
| 9.1 Output Lists of Assembler | 9-1 |
| 9.1.1 Header of assembly list file | 9-2 |
| 9.1.2 Assembly list | 9-3 |
| 9.1.3 Symbol table list | 9-5 |
| 9.1.4 Cross-reference list | 9-6 |
| 9.1.5 Error list | 9-8 |
| 9.2 Output Lists of Linker | 9-9 |
| 9.2.1 Header of link list file | 9-9 |
| 9.2.2 Map list | 9-11 |
| 9.2.3 PUBLIC symbol list | 9-12 |
| 9.2.4 Local symbol list | 9-13 |
| 9.2.5 Error list | 9-14 |
| 9.3 Output List of Object Converter | 9-15 |
| 9.3.1 Error list | 9-15 |

| | |
|-----------------------------------------------------------------------------------------------------|-------|
| 9.4 Output List of Librarian | 9-16 |
| 9.4.1 Library information output list | 9-16 |
| 9.5 Output Lists of List Converter | 9-17 |
| 9.5.1 Absolute assembly list | 9-17 |
| 9.5.2 Error list | 9-17 |
| CHAPTER 10. UTILIZATION OF ASSEMBLER PACKAGE | 10-1 |
| 10.1 How to Execute Each Operation with Efficiency (Use of EXIT Status Function) | 10-1 |
| 10.2 How to Prepare or Complete the Development Environment (Use of Environment Variables) | 10-2 |
| 10.3 How to Interrupt Program Execution | 10-3 |
| 10.4 How to Increase the Readability of Assembly List | 10-3 |
| 10.5 How to Save Yourself Trouble in Program Invocation ... | 10-5 |
| 10.5.1 Describing control instructions in the source program | 10-5 |
| 10.5.2 Creating a parameter file or subcommand file .. | 10-5 |
| 10.6 Creation of Object Module Library File | 10-6 |
| 10.7 How to Change the Option Mark (Use of Environment Variable) | 10-7 |
| CHAPTER 11. ERROR MESSAGES | 11-1 |
| 11.1 Assembler Error Messages | 11-1 |
| 11.2 Linker Error Messages | 11-14 |
| 11.3 Object Converter Error Messages | 11-21 |
| 11.4 Librarian Error Messages | 11-23 |
| 11.5 List Converter Error Messages | 11-27 |

APPENDIXES

| | |
|-----------------------------------------------------|---------|
| APPENDIX A. SAMPLE PROGRAM | A-1 |
| A.1 Source Lists | A-1 |
| A.2 Execution Examples | A-3 |
| A.3 Output Lists | A-4 |
| A.3.1 Assembly Lists | A-4 |
| A.3.2 Symbol Lists | A-7 |
| A.3.3 Cross-reference Lists | A-8 |
| A.3.4 Map List | A-9 |
| A.3.5 PUBLIC Symbol List | A-10 |
| A.3.6 Local Symbol List | A-10 |
| A.3.7 Library Information Output List | A-10 |
| A.3.8 Absolute Assembly List | A-11 |
| APPENDIX B. LIST OF HINTS ON USE | B-1 |
| APPENDIX C. OPTION LISTS | C-1 |
| C.1 List of Assembler Options | C-1 |
| C.2 List of Linker Options | C-6 |
| C.3 List of Object Converter Options | C-11 |
| C.4 List of Librarian Options | C-13 |
| C.5 List of List Converter Options | C-14 |
| APPENDIX D. LIST OF LIBRARIAN SUBCOMMANDS | D-1 |

List of Figures

| | | |
|------------|-------------------------------------------------------------------------------------------|------|
| Fig. 1-1. | Assembler Package | 1-1 |
| Fig. 1-2. | Flow of Assembler | 1-2 |
| Fig. 1-3. | Development Process of Microcomputer-applied Product | 1-3 |
| Fig. 1-4. | Software Development Process | 1-4 |
| Fig. 1-5. | Assembly Phase by This Package | 1-5 |
| Fig. 1-6. | Re-assembly for Debugging | 1-7 |
| Fig. 1-7. | Program Development Utilizing Existing Modules . | 1-8 |
| Fig. 1-8. | Program Development Procedure with This Package | 1-9 |
| Fig. 1-9. | Creation of Source Module File | 1-10 |
| Fig. 1-10. | Function of Structured Assembler Preprocessor .. | 1-11 |
| Fig. 1-11. | Functions of Assembler | 1-12 |
| Fig. 1-12. | Functions of Linker | 1-13 |
| Fig. 1-13. | Functions of Object Converter | 1-14 |
| Fig. 1-14. | Functions of Librarian | 1-15 |
| Fig. 1-15. | Functions of List Converter | 1-16 |
| Fig. 1-16. | Functions of Source Debugger | 1-17 |
| Fig. 3-1. | Structure of Sample Program | 3-2 |
| Fig. 3-2. | Link Directive (1) | 3-7 |
| Fig. 3-3. | Link Directive (2) | 3-7 |
| Fig. 3-4. | Assembler Package Execution Procesure | 3-11 |
| Fig. 4-1. | I/O Files of Assembler | 4-2 |
| Fig. 5-1. | I/O Files of Linker | 5-2 |
| Fig. 5-2. | Memory Spaces | 5-4 |
| Fig. 5-3. | Memory Spaces and Memory Areas | 5-5 |
| Fig. 5-4. | Merging Process (with Same Named Segments) | 5-10 |
| Fig. 5-5. | Merging Process (with No Same-Named Segment) ... | 5-11 |
| Fig. 5-6. | Method of Merging When Merge Type is SEQUENT (with Segment Type Other Than BSEG) | 5-11 |
| Fig. 5-7. | Method of Merging in Bit Units | 5-12 |
| Fig. 5-8. | Method of Merging When Merge Type is 2-byte ALIGN | 5-13 |
| Fig. 5-9 | Method of Merging When Merge Type is 4-byte ALIGN | 5-14 |
| Fig. 5-10. | Locating a Segment by BOND Specification | 5-18 |

| | |
|-----------------------------------------------------------|------|
| Fig. 5-11. Locating a Segment by 2-byte ALIGN | |
| Specification | 5-19 |
| Fig. 5-12. Locating a Segment by 4-byte ALIGN | |
| Specification | 5-19 |
| Fig. 5-13. Locating a Segment by AREA Specification | 5-20 |
| Fig. 5-14. Locating a Segment by Combined AREA/ALIGN | |
| Specification | 5-20 |
| Fig. 5-15. Locating a Segment by FREE Specification | 5-21 |
| Fig. 5-16. Examples of Improper Segment Location | 5-23 |
| Fig. 5-17. Memory Area Name Specification | 5-24 |
| Fig. 5-18. Priority of Address Allocation to Segments | |
| by Location Type | 5-25 |
| Fig. 5-19. Memory Area Name | 5-29 |
| Fig. 5-20. Incorrect Memory Area Definition | 5-32 |
| Fig. 6-1. I/O Files of Object Converter | 6-2 |
| Fig. 7-1. I/O Files of Librarian | 7-2 |
| Fig. 7-2. Procedure for Creating a Library File | 7-4 |
| Fig. 8-1. I/O Files of List Converter | 8-2 |

List of Tables

| | |
|--------------------------------------------------------------------------------------------------------|------|
| Table 1-1. Restrictions on Number of Symbols | 1-23 |
| Table 1-2. Maximum Performance Characteristics of Assembler Package | 1-24 |
| Table 2-1. Program Files Offered | 2-1 |
| Table 2-2. System Configuration | 2-3 |
| Table 4-1. I/O Files of Assembler | 4-1 |
| Table 4-2. Types of Assembler Options | 4-8 |
| Table 4-3. Priority of Assembler Options | 4-10 |
| Table 4-4. List of Target Devices | 4-13 |
| Table 4-5. Characters That Can Be Described as Title | 4-48 |
| Table 5-1. I/O Files of Linker | 5-1 |
| Table 5-2. Areas to Be Allocated to Segments by Default Assumption | 5-7 |
| Table 5-3. Merge Type by Relocation Attribute | 5-9 |
| Table 5-4. List of Location Types of Segments | 5-17 |
| Table 5-5. Location Type Given by Relocation Attribute | 5-22 |
| Table 5-6. Types of Directives | 5-26 |
| Table 5-7. Default Memory Areas and Address Ranges That Can Be Re-defined | 5-30 |
| Table 5-8. Merge Types That Can Be Specified with Directive | 5-36 |
| Table 5-9. Segment Location by Combination of Bind Specification and Memory Space Specification ... | 5-38 |
| Table 5-10. Types of Linker Options | 5-45 |
| Table 5-11. Priority of Linker Options | 5-47 |
| Table 6-1. I/O Files of Object Converter | 6-1 |
| Table 6-2. File Types of Output Files for Each Extension Space | 6-3 |
| Table 6-3. Types of Object Converter Options | 6-10 |
| Table 7-1. I/O Files of Librarian | 7-1 |
| Table 7-2. Types of Librarian Options | 7-11 |
| Table 7-3. Subcommands | 7-23 |
| Table 8-1. I/O Files of List Converter | 8-1 |
| Table 8-2. Types of List Converter Options | 8-10 |

CHAPTER 1. GENERAL

1.1 Assembler Overview

The RA78K Series Assembler Package is a series of programs designed to translate each source program coded in the assembly language for the 78K series microcomputers, into machine language coding.

The assembler package contains six programs: Structured Assembler Preprocessor, Assembler, Linker, Object Converter, Librarian, and List Converter.

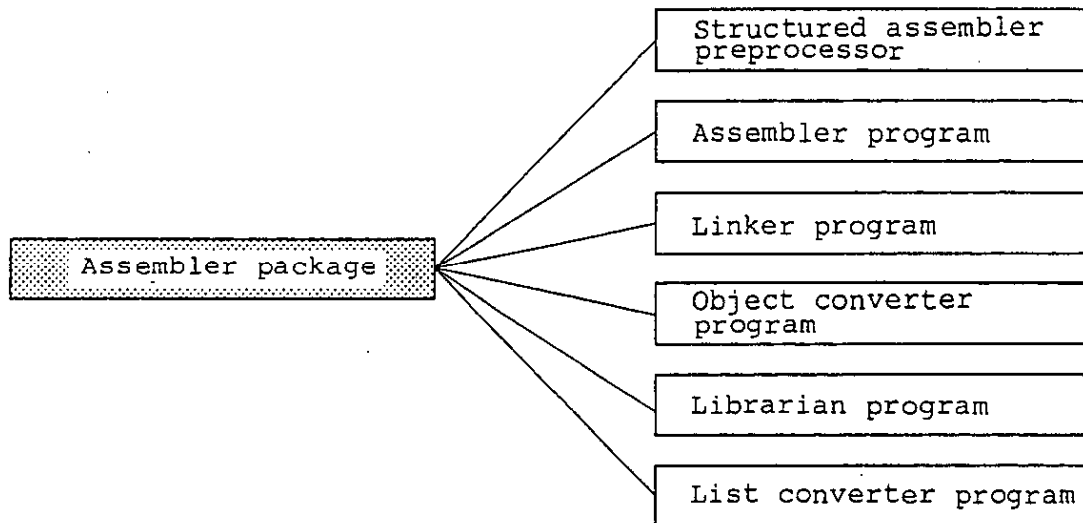


Fig. 1-1. Assembler Package

1.1.1 What is an assembler?

(1) Assembly language and machine language

An assembly language is the most fundamental programming language for microprocessors.

To have a microprocessor in a microcomputer do its job, programs and data are required. These programs and data must be written by a human being (i.e., a programmer) and stored in the memory section of the microcomputer. Programs and data that can be handled by the microcomputer is nothing but a set or combinations of binary numbers which is called machine language (i.e., the language that can be understood or interpreted by the computer).

To create a program in machine language coding, namely, by a set of binary numbers is not an easy job for a human being, because it's difficult for him or her to remember the coding and he or she is likely to make errors in coding. Because assembly language instructions are in one-to-one correspondence with machine language instructions, the assembly language can give the computer a detailed or specific instruction (for example, improving the I/O processing speed). For this reason, there is a method of creating a program using an abbreviated symbol (or mnemonic symbol) which represents the meaning of a machine language instruction to assist the human memory. A programming language system by this symbolic coding is called an assembly language. To translate a program created in the assembly language into a set of binary numbers that can be understood by the micro-processor, another program is required. This program is called an assembler.

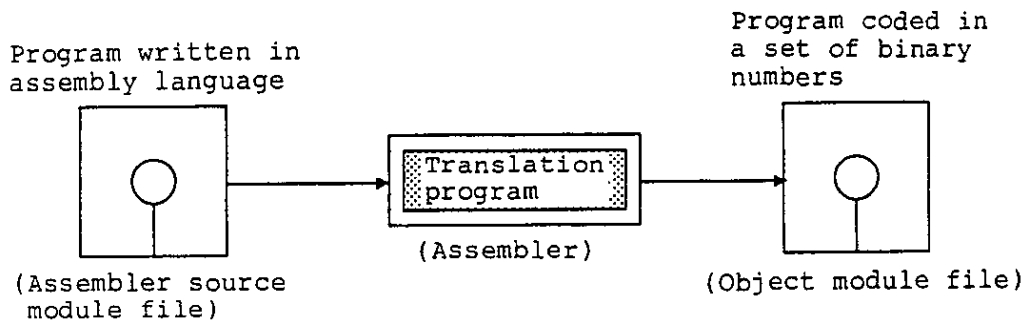


Fig. 1-2. Flow of Assembler

(2) Development of microcomputer-applied products and role of this package

Fig. 1-3 illustrates the standing of the programming in assembly language in the development process of a microcomputer-applied product.

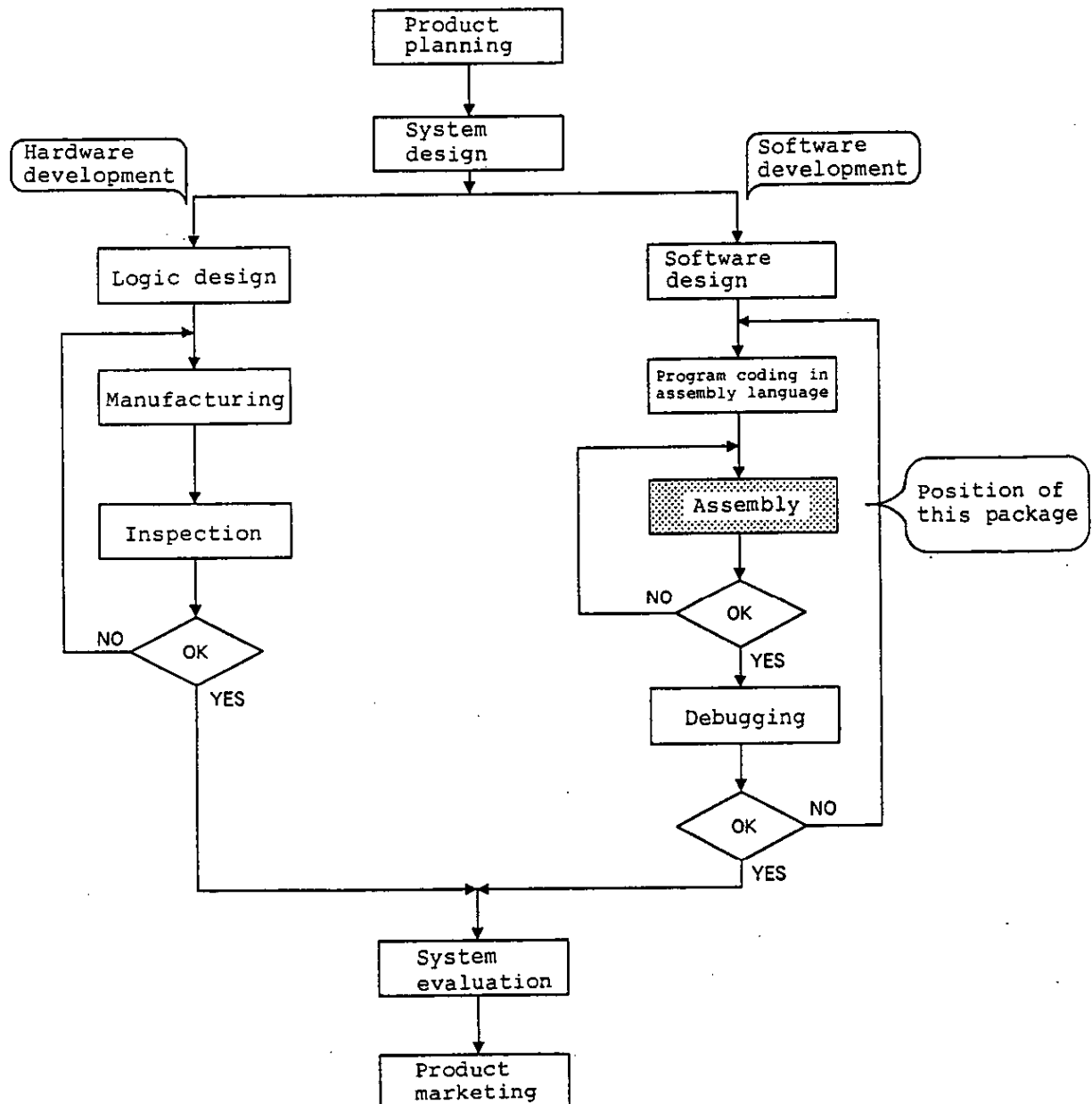


Fig. 1-3. Development Process of Microcomputer-applied Product

The software development process will be further detailed in Fig. 1-4 below.

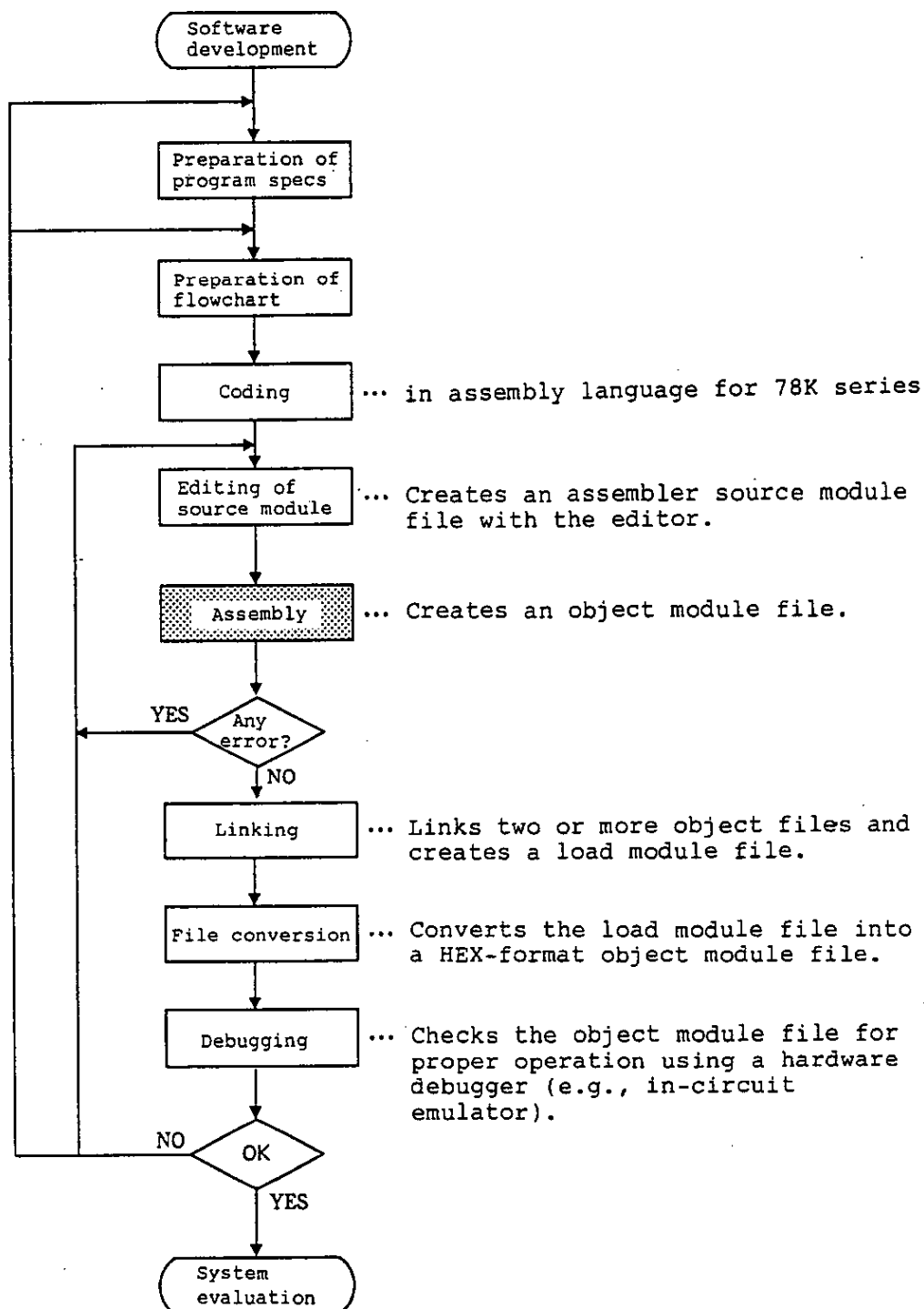


Fig. 1-4. Software Development Process

The assembly phase in the software development process will be reviewed in further detail by giving an example of this package.

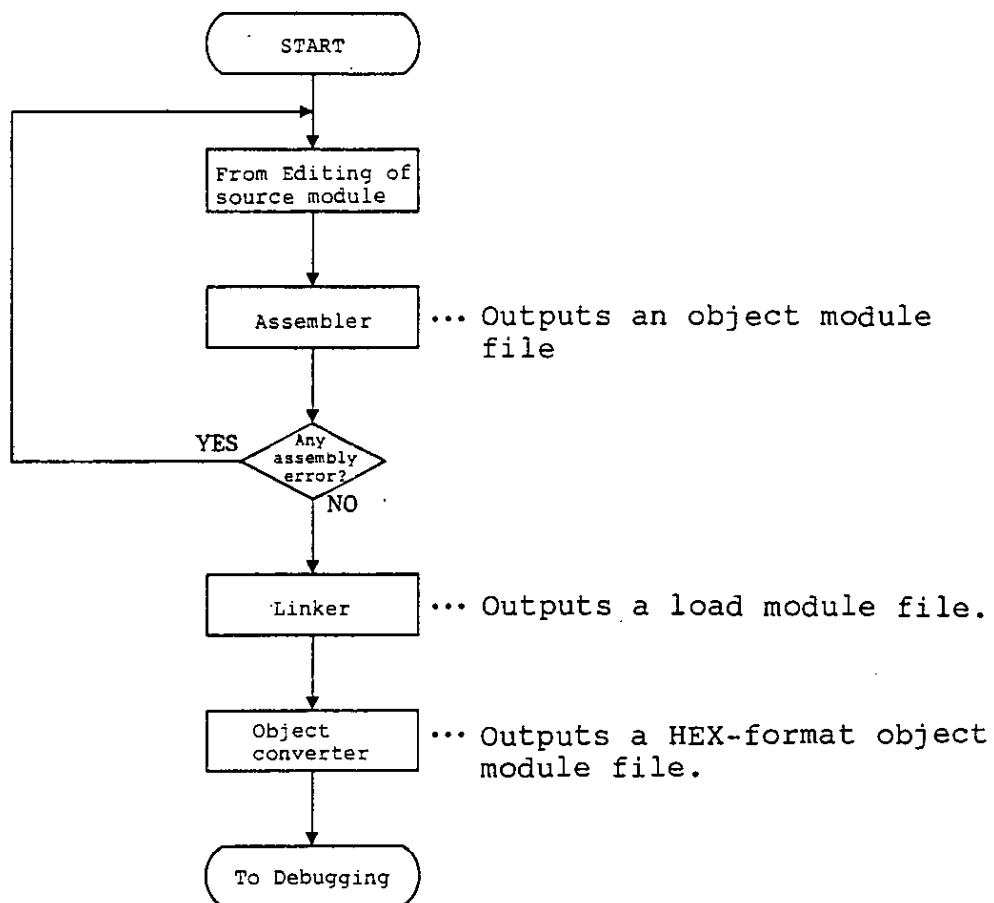


Fig. 1-5. Assembly Phase by This Package

1.1.2 What is a relocatable assembler?

The machine language translated from a source language by the assembler will be stored in the memory of the microcomputer before use. In this case, in which memory location each machine language instruction will be stored must have been determined. Therefore, information on "the allocation of each machine language instruction to a specific address in memory" will be added to the machine language converted (translated) by the assembler.

Depending on the method of allocating addresses to machine language instructions, an assembler can be broadly divided into an absolute assembler and a relocatable assembler.

- o Absolute assembler

The machine language instructions converted (translated) in one-time assembly operation are allocated to absolute addresses.

- o Relocatable assembler

Addresses determined for the machine language instructions converted in one-time assembly operation are tentative. Absolute addresses will be determined by a program called the linker.

In the past, when a program was created with the absolute assembler, programmers had to, as a rule, complete programming at a time. However, if you create a large program at a time, the program becomes complicated, making analysis and maintenance of the program troublesome. To avoid this, such a large program is developed by dividing it into several subprograms (i.e., modules) for each functional unit. This programming technique is called the modular programming.

The relocatable assembler is an assembler suitable for modular programming. The following advantages can be derived from modular programming with the relocatable assembler:

- (1) Increase in development efficiency

It is difficult to write a large program at a time. In such a case, divide the program into modules for each function and the program can be developed with two or more programmers engaged in writing subprograms at the same time. This will certainly increase the development efficiency of the program.

If any bugs are found in the program, you do not need to re-assemble the entire program just to correct part of the program. Only the subprogram (module) requiring correction(s) can be re-assembled. This will help shorten the debugging time.

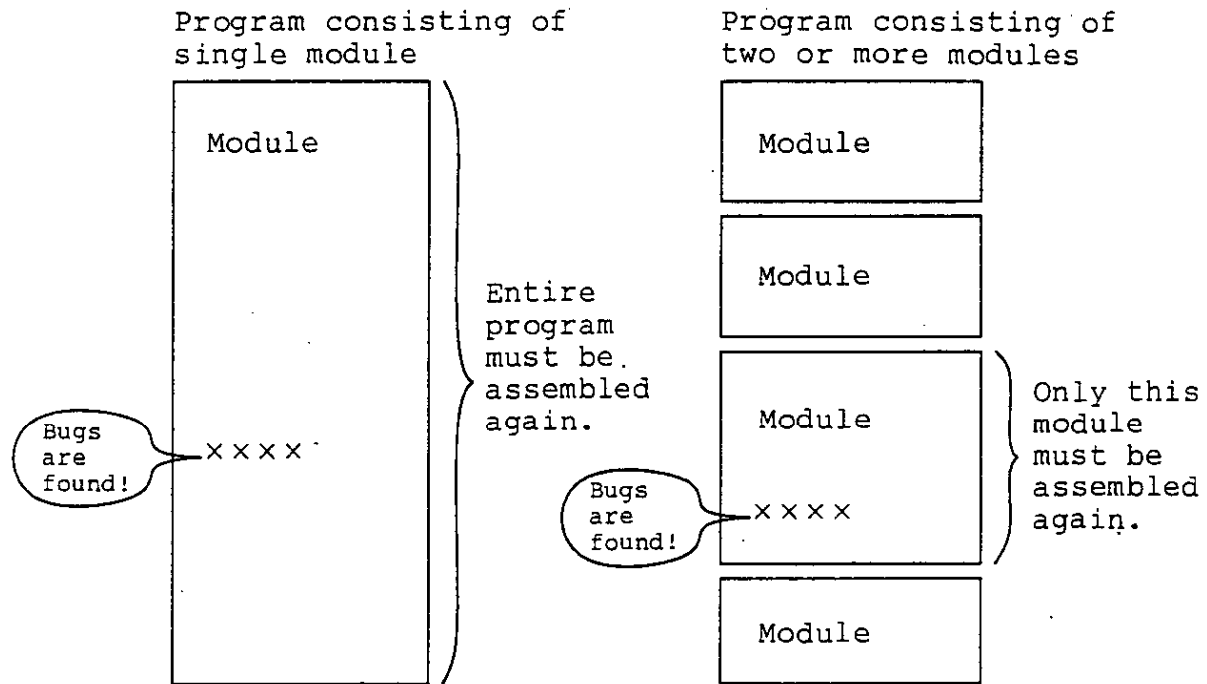


Fig. 1-6. Re-assembly for Debugging

(2) Utilization of resources

Highly reliable, highly versatile modules which have been previously created can be utilized for creation of another program. If you accumulate such high-versatility modules as software resources, you can save time and labor in developing a new program.

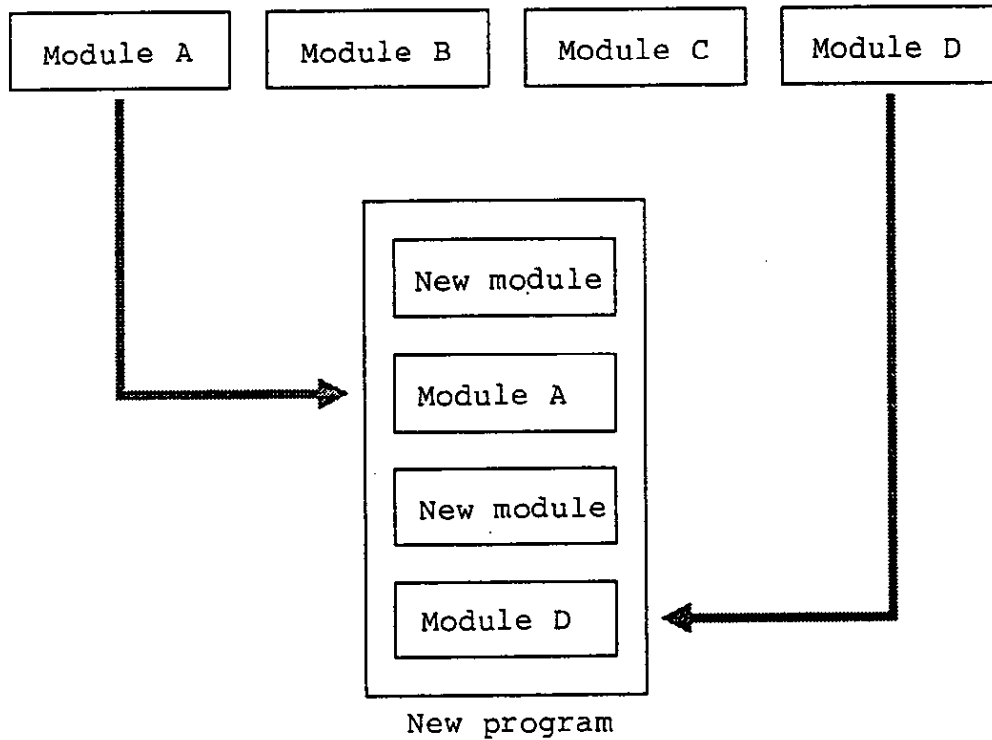


Fig. 1-7. Program Development Utilizing Existing Modules

1.2 Functional Outline of Assembler Package

An ordinary program development procedure with this assembler package is illustrated in Fig. 1-8. The development of a program is basically performed by using assembler, linker, and object converter programs.

This relocatable assembler package also contains programs such as Librarian and List Converter so that programs can be developed with high efficiency.

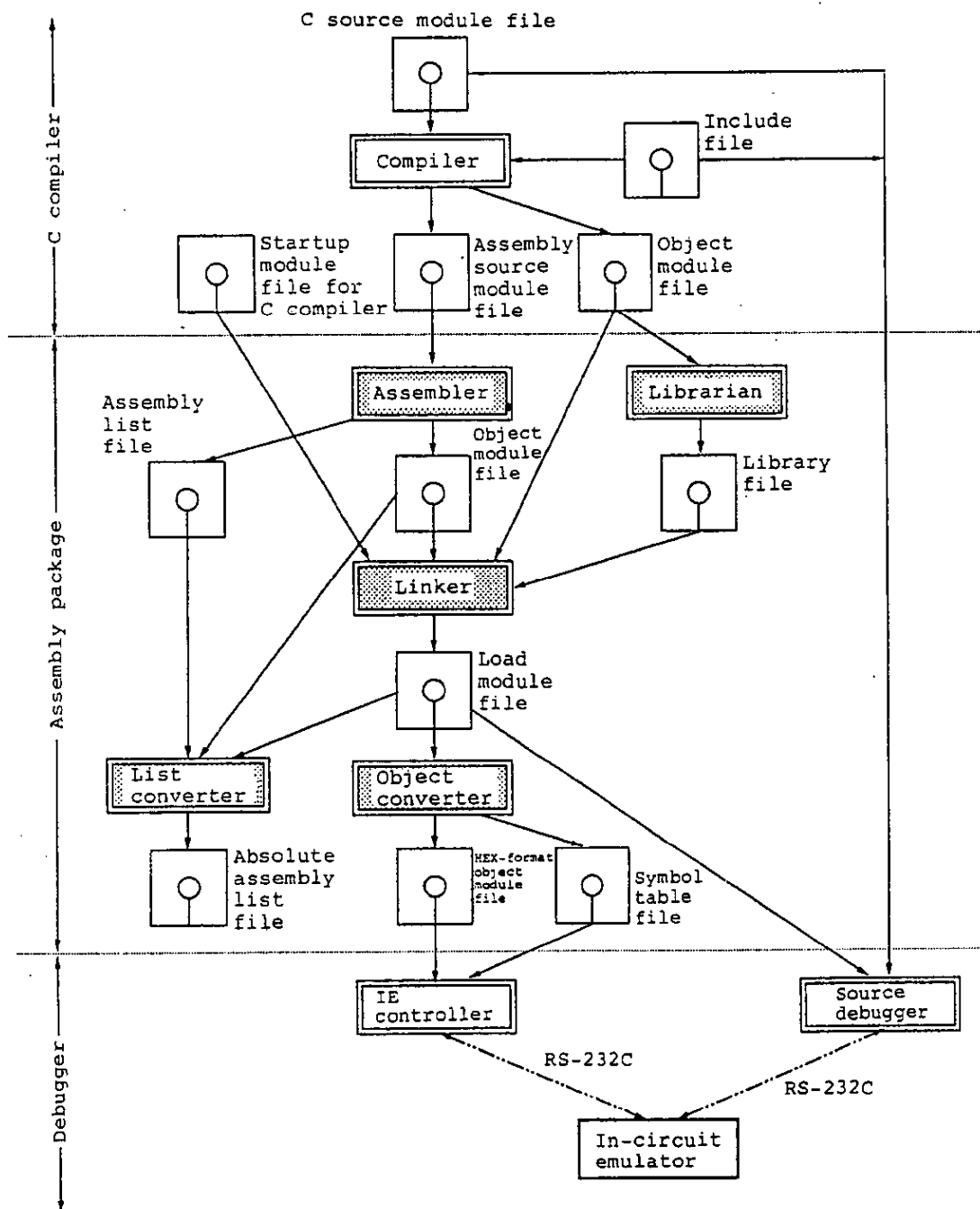


Fig. 1-8. Program Development Procedure with This Package

1.2.1 Creation of source module file with editor

Divide one program into several functional modules.

Each module becomes the unit of coding as well as the unit of input to the assembler. A module serving as the unit of input to the assembler is called a source module.

After coding each source module, the source module is written into a file with the editor. The file thus created is called a source module file.

The source module file becomes an input file to the assembler.

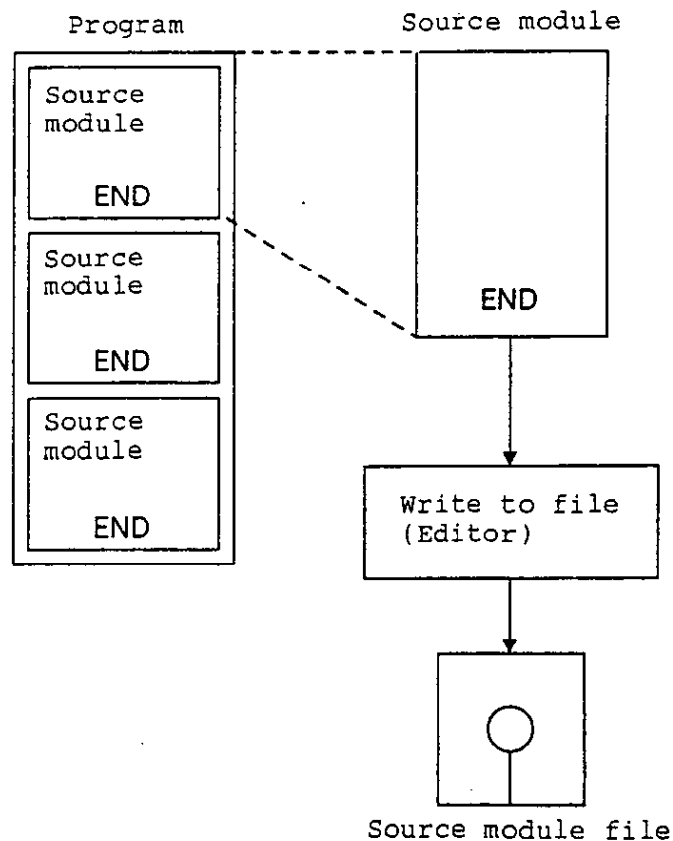


Fig. 1-9. Creation of Source Module File

1.2.2 Structured assembler preprocessor

The structured assembler preprocessor is a program for implementing structured programming in the assembly language. This program accepts a source program written in the structured assembly language as an input file and outputs an assembler source module file.

For details of the structured assembler preprocessor and structured assembly language, see the ST78K Series Structured Assembler Preprocessor User's Manual published separately.

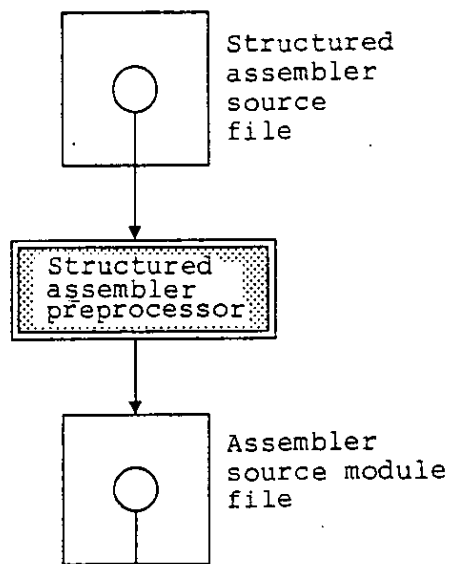


Fig. 1-10. Function of Structured Assembler Preprocessor

1.2.3 Assembler

The assembler accepts assembler source module files as input files and translates assembly language into machine language (a set of binary numbers). If any coding error is found in the input source module, the assembler outputs an assembly error. If no assembly error is found, the assembler outputs an object module file which contains machine language information and relocation information relating to the allocation address of each machine language instruction. The assembler also outputs information at assembly time as an assembly list file.

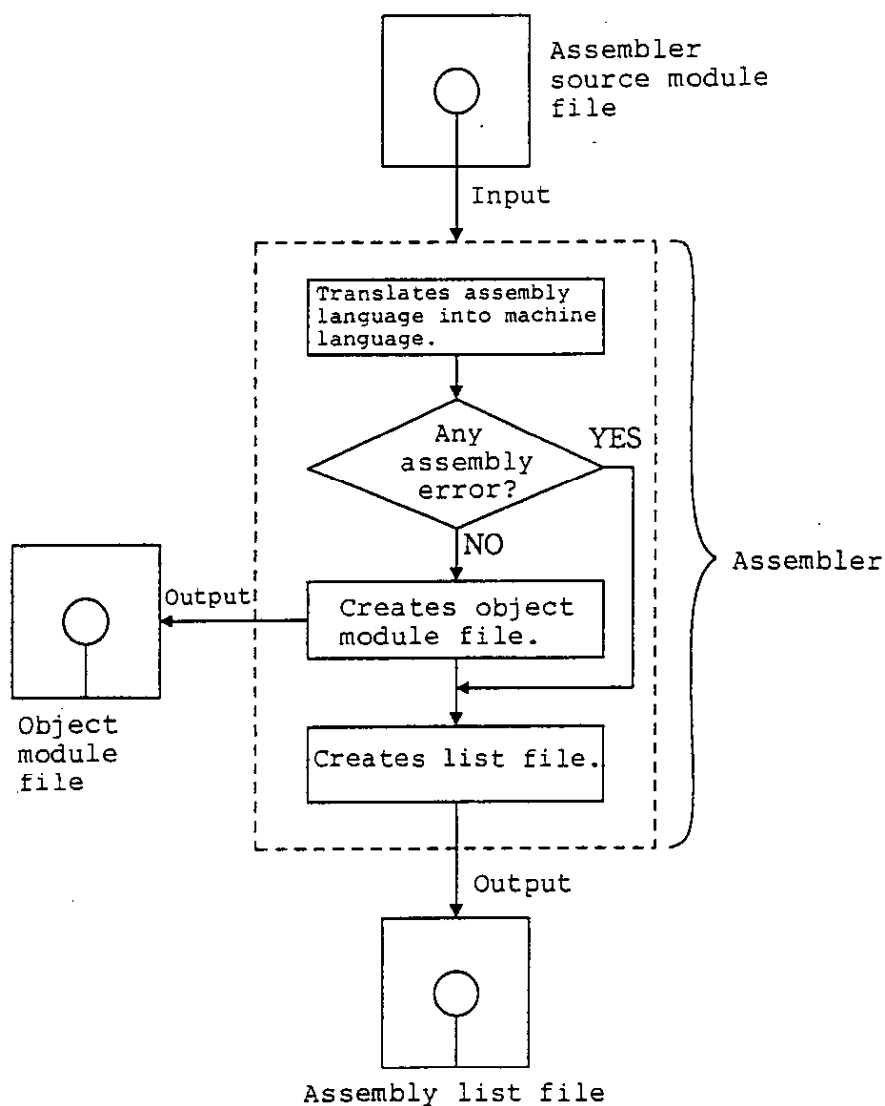


Fig. 1-11. Functions of Assembler

1.2.4 Linker

The linker accepts two or more object module files output by the compiler or assembler as input files and combines them with a library file for output as a single load module file.

The linker also determines addresses to be allocated to each relocatable segment in the input module, whereby the correct values of the respective relocatable symbols and external reference symbols are determined and embedded into the output load module file.

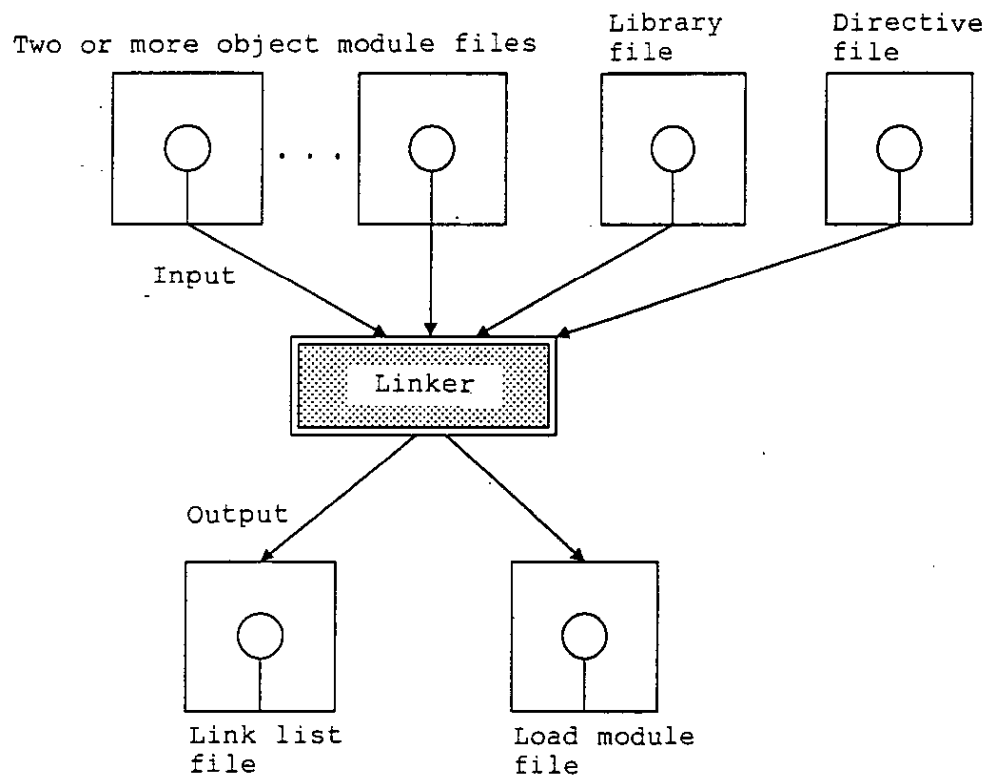


Fig. 1-12. Functions of Linker

1.2.5 Object converter

The object converter accepts the load module file output by the linker as an input file, converts its file format, and outputs the result of the conversion as an HEX-format object module file. The object converter also outputs the symbol information required in symbolic debugging as a symbol table file.

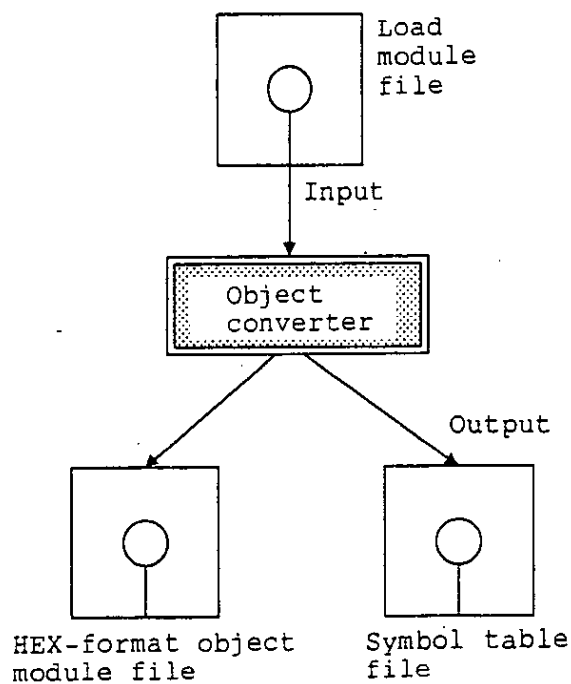


Fig. 1-13. Functions of Object Converter

1.2.6 Librarian

Modules (programs) which have versatility and a definitive interface should be kept in a single library file. By so doing, a number of object module files in a single file can be handled with ease.

The linker has a function to extract only the required modules from the library file and link them with the input object module file(s). Therefore, if you register (store) two or more modules in a single library file, it is unnecessary for you to specify the required module names one by one at linking time.

The librarian is used to create and update a library file.

Object module files output by
Compiler or Assembler

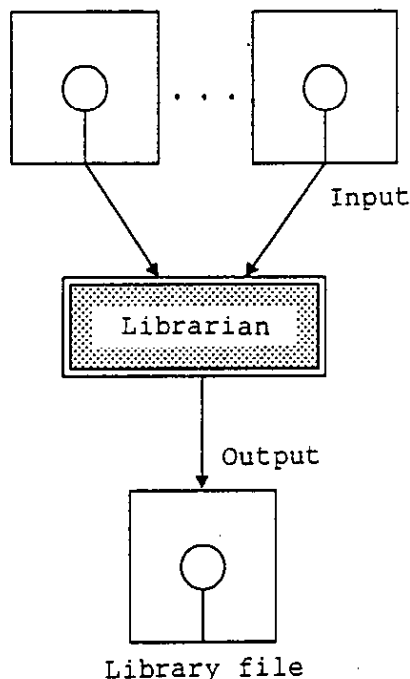


Fig. 1-14. Functions of Librarian

1.2.7 List converter

The list converter accepts the object module file and assembly list file output by the assembler and the load module file output by the linker as input files and outputs an absolute assembly list file.

One drawback of a relocatable assembly list is such that address values and relocatable values in the list differ from the actual values. Because an absolute assembly list has no such drawback, the absolute assembly list output by the list converter will facilitate program debugging as well as program maintenance.

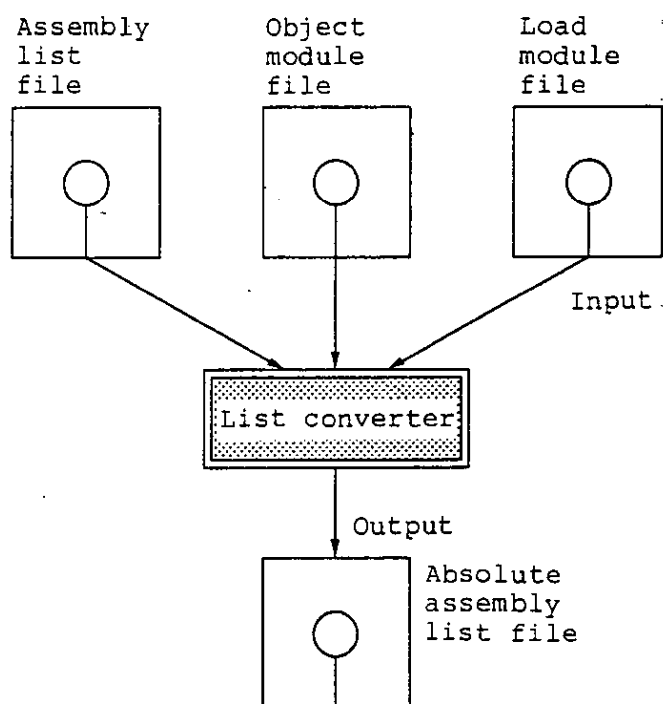


Fig. 1-15. Functions of List Converter

1.2.8 Source debugger

Symbolic debugging (program debugging at the symbolic level) can be performed by downloading the HEX-format object module file output by the object converter into the IE (in-circuit emulator) or EB (evaluation board) of the target system and by reading the symbol table file output by the object converter.

As another way of debugging, you can specify an option which tells the compiler to output debugging information at the time of compiling the source program subject to debugging. By so doing, information on the symbols and line numbers required for debugging is added to the object module so that program debugging can be performed at the source level.

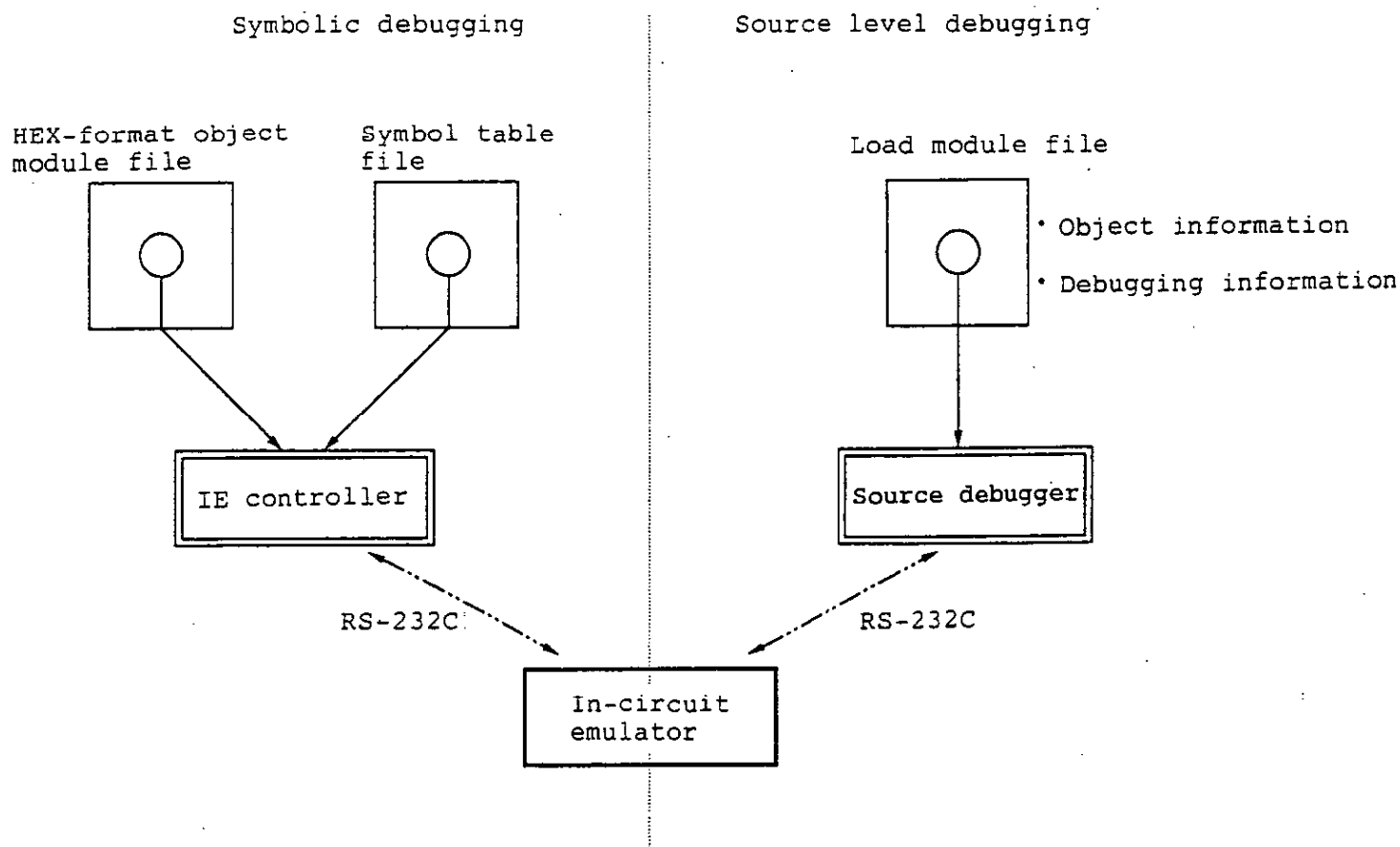
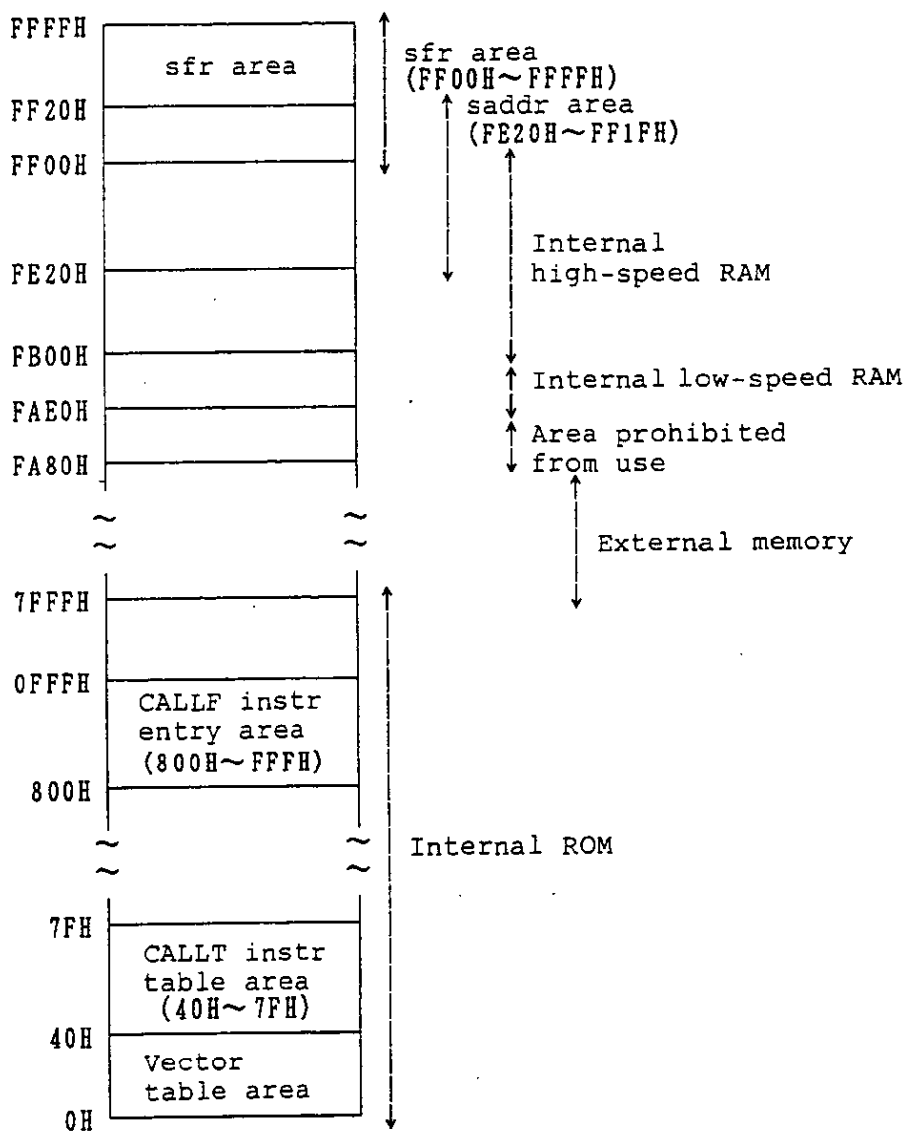


Fig. 1-16. Functions of Source Debugger

1.3 Memory Maps

The memory maps of the respective series in this package are shown in this section.

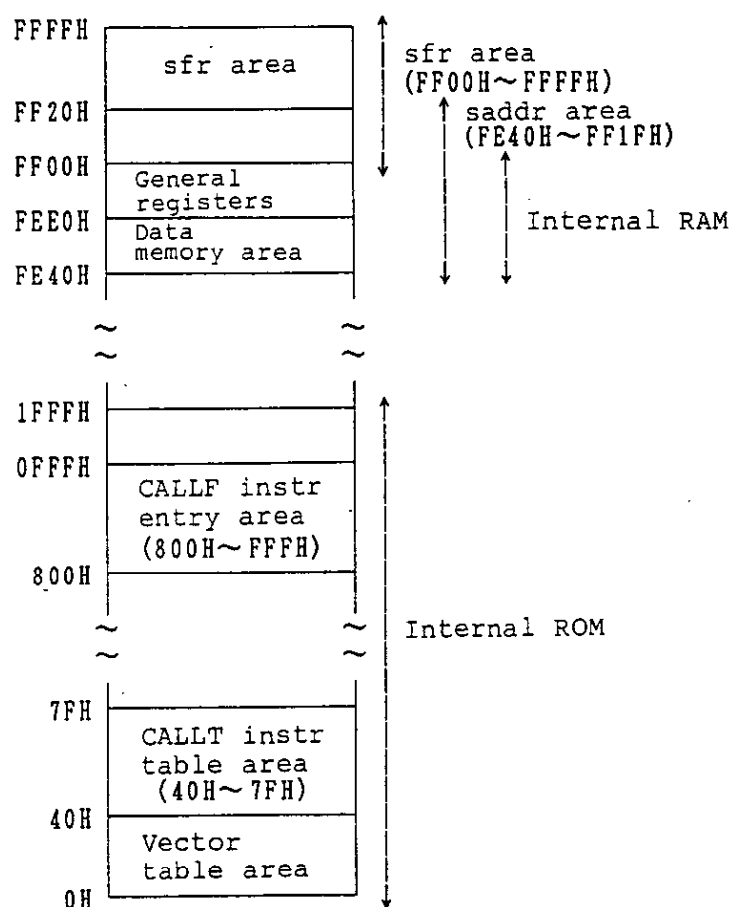
(1) Memory map of 78K/0 (uPD78014)



The internal ROM/RAM areas applicable to each target device in the 78K/0 series are as listed below.

| Target device | Internal ROM area | Internal high-speed RAM area | Internal low-speed RAM area |
|---------------|-------------------|------------------------------|-----------------------------|
| uPD78012 | 0000H to 3FFFH | FD00H to FEFFH | FAE0H to FAFH |
| uPD78014 | 0000H to 7FFFH | FB00H to FEFFH | FAE0H to FAFH |

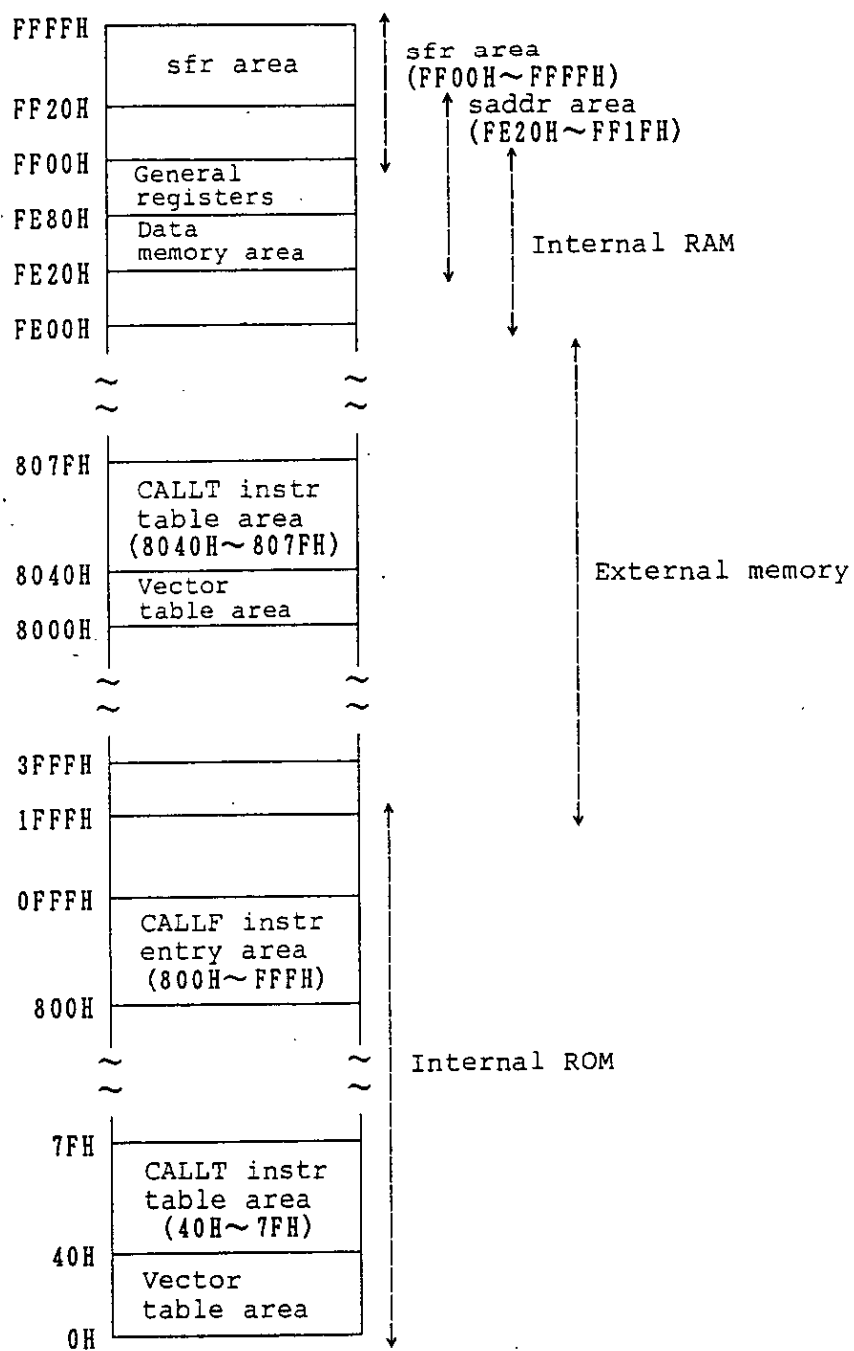
(2) Memory map of 78K/I (uPD78112)



The internal ROM/RAM areas applicable to each target device in the 78K/I series are as listed below.

| Target device | Internal ROM area | Internal RAM area |
|------------------------|-------------------|-------------------|
| uPD78112 | 0000H to 1FFFFH | FE40H to FEFFFH |
| uPD78134/ uPD78134A | 0000H to 3FFFFH | FD80H to FEFFFH |
| uPD78136 | 0000H to 5FFFFH | FD00H to FEFFFH |
| uPD78138 | 0000H to 7FFFFH | FC80H to FEFFFH |

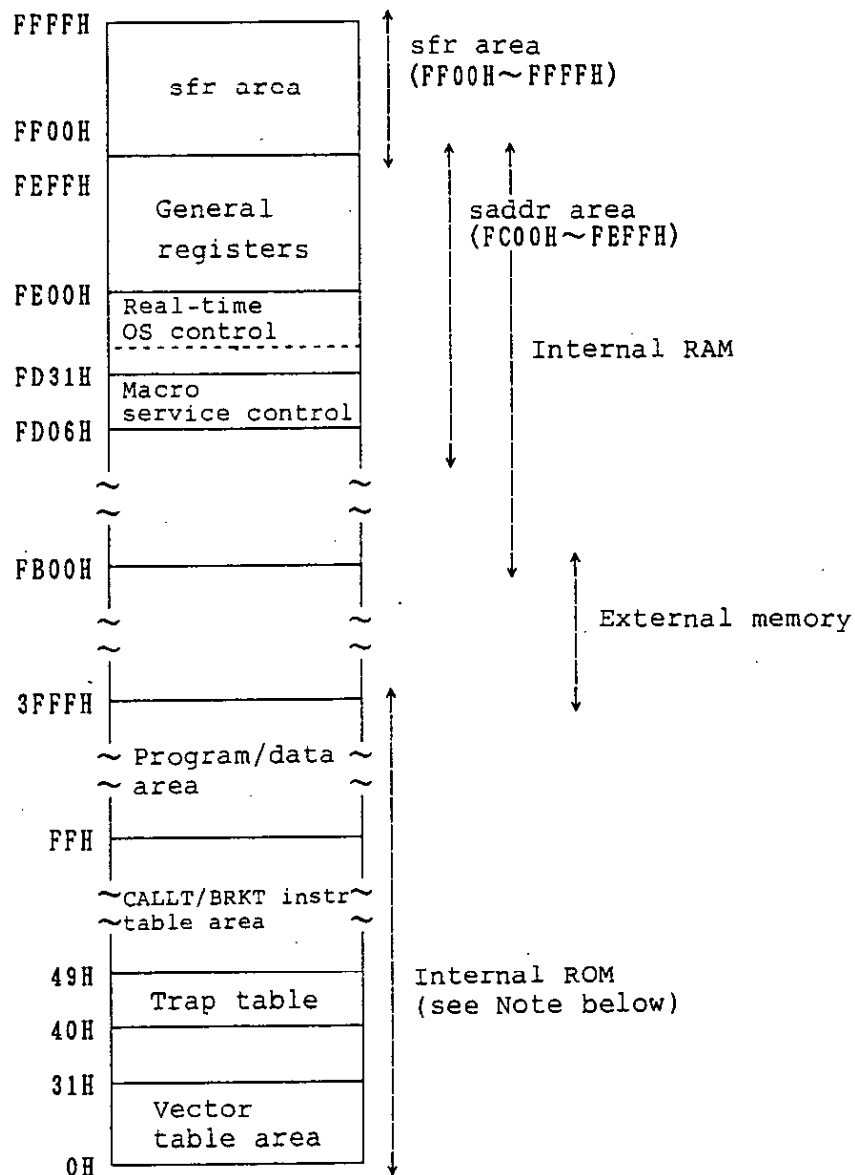
(3) Memory map of 78K/III (uPD78312A)



The internal ROM/RAM areas applicable to each target device in the 78K/III series are as listed below.

| Target device | Internal ROM area | Internal RAM area |
|------------------------|-------------------|-------------------|
| uPD78310/ uPD78310A | None | FE00H to FEFFH |
| uPD78312/ uPD78312A | 0000H to 1FFFFH | FE00H to FEFFH |
| uPD78320 | None . | FC80H to FEFFH |
| uPD78322 | 0000H to 3FFFFH | FC80H to FEFFH |
| uPD78330 | None | FB00H to FEFFH |
| uPD78334 | 0000H to 7FFFFH | FB00H to FEFFH |

(4) Memory map of 78K/VI (uPD78602)



Note: When the EA (External Access) pin of the uPD78600 or uPD78602 is set at a Low level, the internal ROM area becomes an external memory area.

The internal ROM/RAM areas applicable to each target device in the 78K/VI series are as listed below.

| Target device | Internal ROM area | Internal RAM area |
|---------------|-------------------|-------------------|
| uPD78600 | None | FB00H to FEFFH |
| uPD78602 | 0000H to 3FFFH | FB00H to FEFFH |

1.4 Reminders Before Program Development

Before you set your hand to the development of a program, keep in mind the following points:

1.4.1 Number of files than can be input to Linker

The number of object module files that can be input to the linker is as follows:

With 78K/0, 78K/I, 78K/VI: 128 files

With 78K/III . : 64 files

1.4.2 Restriction on number of symbols

The number of local symbols and that of PUBLIC symbols in the assembler and linker, respectively, are restricted as shown in the table below.

Table 1-1. Restrictions on Number of Symbols

| | Number of symbols | |
|-----------|------------------------|-----------------------|
| | No. of local symbols | No. of PUBLIC symbols |
| Assembler | 2,900 (see Note 1) | |
| Linker | 2,900 x No. of modules | 3,000 (see Note 2) |

NOTE: 1. There is no restriction on the number of symbols by symbol type. Undefined symbols will also be counted and included in the total number of symbols.

2. If the number of PUBLIC symbols exceeds 2,000, the execution speed slows down because of the additional time required to access a temporary file.

1.4.3 Maximum performance characteristics of assembler package

The maximum performance characteristics of the assembler package that should be kept in your mind before program development are listed in Table 1-2 below.

Table 1-2. Maximum Performance Characteristics of
Assembler Package

| Program name | Item | | Restriction |
|--------------|----------------------------|------------------|----------------------------------------------------------------|
| Assembler | Symbol length | w/o -S option | 8 characters |
| | | with -S option | 31 characters |
| | No. of characters per line | | 130 characters |
| | No. of segments | ?ASEG | 20 segments |
| | | other than ?ASEG | 80 segments |
| Linker | No. of input module files | | 64 files (with 78K/III) or 128 files (with other than 78K/III) |

1.5 Features of Assembler Package

This package has the following features:

(1) Macro function

When the same group of instructions must be described in a source program over and over again, a macro can be defined by giving a single macro name to the group of instructions. By using this macro function, coding efficiency and readability of the program can be increased.

(2) Optimize function of branch instructions

The assembler package has an assembler directive to automatically select a branch instruction (i.e., BR directive). To create a program with high memory efficiency, a 2-byte branch instruction must be described according to the branch destination range of the branch instruction. However, it is troublesome for the programmer to describe a branch instruction by paying attention to the branch destination range for each branching. If the BR directive is described, the assembler generates the appropriate branch instruction according to the branch destination range. This is called the optimize function of branch instructions.

(3) Conditional assembly function

With this function, part of a source program can be specified for assembly or non-assembly according to a predetermined condition. If a debug statement is described in a source program, whether or not the debug statement should be translated into machine language can be selected by setting a switch for conditional assembly. When the debug statement is no longer required, the source program can be assembled without major modifications to the program.

(4) Directive for general-purpose register selection

As representations for the 78K series general-purpose registers, absolute names (R0, R1, RP0, etc.) and function names (X, A, AX, etc.) are used. When describing a function name in a source program, a general-purpose register select directive must always be used. The RSS directive is provided to allow description of a function name in a source program.

CHAPTER 2. PRODUCT OVERVIEW

2.1 Contents of Product

This product offers the program files listed in Table 2-1 below.

Table 2-1. Program Files Offered

| Program name | Filename (see Note 1) | File type |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| Assembler | RA78Kn.EXE | Command file |
| | RA78Kn.OMx (with 78K/0: x=1 or 2; with 78K/I: x=1, 2, 3, or 4; with 78K/III: x =1, 2, 3, 4, 5, or 6; with 78K/VI: x=1 or 2) | Overlay files |
| | RA78Kn.HLP | HELP file |
| Linker | LK78Kn.EXE | Command file |
| | RA78Kn.OM1 (see Note 2) | Overlay file (shared with Assembler) |
| | LK78Kn.HLP | HELP file |
| Object converter | OC78Kn.EXE | Command file |
| | OC78Kn.HLP | HELP file |
| Librarian | LB78Kn.EXE | Command file |
| | LB78Kn.HLP | HELP file |
| List converter | LCNV78Kn.EXE | Command file |
| | RA78Kn.OMx (x=1,2, or 3) | Overlay files (shared with Assembler) |
| | LCNV78Kn.HLP | HELP file |
| | 78KnMAIN.ASM 78KnSUB.ASM | Sample program files for operation check |

Note: 1. n in the filename indicates one of the numbers 0, 1, 2, 3, and 6 corresponding to 78K series names 78K/0, 78K/I, 78K/II, 78K/III, and 78K/VI, respectively.

2. This overlay file is not used with the 78K/VI.

3. In addition to the above programs, the ST78Kn structured assembler preprocessor (n=0, 1, 2, 3, or 6) is included. For details, refer to the 78K Series Structured Assembler Preprocessor User's Manual.

Remarks: 1. A command file refers to a file which will be the first to be read into memory when each program (i.e., assembler, linker, object converter, librarian, or list converter) is started up.

2. An overlay file refers to a file which will be read into memory only when required during the execution of each program.

2.2 Form of Supplied File Medium

This product is supplied in either of the following two file media:

- o 8-inch 2DD (double-sided double-density) floppy disk
- o 5-inch 2HD (double-sided high-density) floppy disk

2.3 System Configuration

The respective programs contained in this product operate in the environment indicated in Table 2-2 below.

Table 2-2. System Configuration

| Host computer | OS | | Memory size required |
|--------------------------------------------------------------|----------------------------------------------------------------|-----------------------------|----------------------------|
| | CPU | CONFIG.SYS | |
| PC-9800 series | V30 TM 80386 TM 8086 TM | MS-DOS (V3.10) | 384K bytes (see Note 2) |
| IBM PC IBM PC/XT TM IBM PC/AT TM | | PC-DOS (V3.10, V3.30) | |

- NOTE: 1. Each program in the RA78K series relocatable assembler package operates on the MS-DOS for the PC-9800 series offered by NEC. NEC is not responsible for proper operation of these programs on any commercially available MS-DOS other than that supplied by NEC.
2. The memory size required indicates the maximum value of memory space required for each program to operate and does not include the system area.

CHAPTER 3. EXECUTION OF ASSEMBLER PACKAGE

This chapter describes a procedure required when you actually execute this assembler package. By executing each program in the assembler package according to the procedure described in this chapter, you may accustom yourself to the operation of each program in the assembler package.

In all the execution examples contained in this and subsequent chapters, operations on the basis of MS-DOS (PC-9800 series) are shown using programs for the 78K/III series.

3.1 Before Executing the Assembler Package

3.1.1 Confirming the contents of the supplied disk

Set the supplied floppy disk (or its backup disk) containing this assembler package in the disk drive and confirm that the disk contains all the program files listed in Section 2.1, Contents of Product.

3.1.2 Sample program

Of the program files stored in the supplied floppy disk, "78KnMAIN.ASM" and "78KnSUB.ASM" (where n is 0, 1, 2, 3, or 6 indicating each 78K series name) are the files containing a sample program for the operation check of the assembler package. These files will be input as source program files to the assembler in the assembler operation to be described later. Here, the contents of this sample program will be briefly explained.

The sample program is used to convert hexadecimal data into ASCII codes and is divided into two modules: a main routine and a subroutine. The main routine is given a module name "SAMPM" and is stored in a source module file named "78KnMAIN.ASM".

The subroutine is given a module name "SAMPS" and is stored in a source module file named "78KnSUB.ASM".

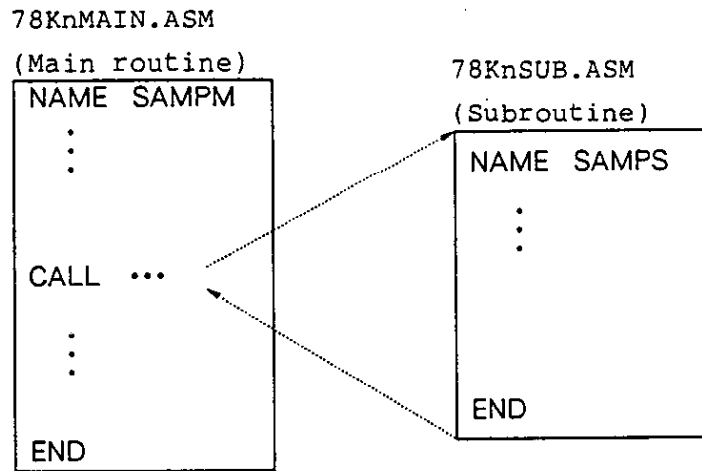


Fig. 3-1. Structure of Sample Program

o 78K3MAIN.ASM (Main routine)

```

$      PROCESSOR(310)

      NAME      SAMPM
;*****
;*
;*      HEX -> ASCII Conversion Program
;*
;*      main-routine
;*
;*****

      PUBLIC    MAIN.START
      EXTRN     CONVAH

DATA    DSEG      AT 0FE20H
HDTSA:  DS        1
STASC:  DS        2

CODE    CSEG      AT 0H
MAIN:   DW        START

START:  CSEG
        MOV       RFM, #00
        MOVW      SP, #0FE80H
        MOV       MM, #00
        MOV       STBC, #08H

        MOV       HDTSA, #1AH
        MOVW      HL, #HDTSA          ;set hex 2-code data in HL register

        CALL      !CONVAH             ;convert ASCII <- HEX
                                       ;output BC-register <- ASCII code
                                       ;set DE <- store ASCII code table
        MOVW      DE, #STASC
        MOV       A, B
        MOV       [DE+], A
        MOV       A, C
        MOV       [DE+], A

        BR        $$

      END

```

o 78K3SUB.ASM (Subroutine)

```

$      PROCESSOR(310)

      NAME      SAMPS
;*****
;
;      HEX -> ASCII Conversion Program
;
;      sub-routine
;
;      input condition : (HL) <- hex 2 code
;
;      output condition : BC-register <-ASCII 2 code
;
;*****

      PUBLIC  CONVAH

CONVAH: CSEG
      MOV     A,#0
      ROL4    [HL]          ;hex upper code load
      CALL    !SASC
      MOV     B,A           ;store result

      MOV     A,#0
      ROL4    [HL]          ;hex lower code load
      CALL    !SASC
      MOV     C,A           ;store result

      RET

;*****
;      subroutine  convert ASCII code
;
;      input      Acc (lower 4bits) <- hex code
;
;      output     Acc      <- ASCII code
;
;*****

SASC:  CMP     A,#0AH        ;check hex code > 9
      BC      SSASC1
      ADD     A,#07H        ;bias(+7)
SASC1: ADD     A,#30H        ;bias(+30)
      RET

      END

```

- Note: 1. This sample program is provided as a reference program which is merely intended for your use in learning the functions and operations of the assembler package. Therefore, it cannot be used as is as an application program.
2. With this sample program, no initial value setting is performed for the Register Set Select (RSS) flag and Register Bank Select (RBS0 to RBS2) flags. Therefore, the register settings used in the sample program are as follows:
- Register bank: 0 (0FEF0H to 0FEFFH)
 - RSS flag : 0

3.2 Procedure for Assembler Package Execution

This section introduces to you the basic procedure for executing this assembler package.

(1) Assemble the sample program "78K3MAIN.ASM".

- o Enter the start-up command line of the assembler as follows:

```
A><u>ra78k3 78k3main.asm -g
```

- o The following message will be output to the console:

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete.      0 error(s) and      0 warning(s) found.
```

(2) Check the contents of drive A.

The assembler has now created two files: "78K3MAIN.REL" (object module file) and "78K3MAIN.PRN" (assembly list file). If you specify the -E option at assembly time, the assembler will output an error list file.

(3) Assemble the sample program "78K3SUB.ASM".

- o Enter the start-up command line of the assembler as follows:

```
A><u>ra78k3 78k3sub.asm -g
```


- o The following message will be output to the console.

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

- (4) Check the contents of drive A.

Two files "78K3SUB.REL" (object module file) and "78K3SUB.PRN" (assembly list file) have been output by the assembler.

If you specify the -E option at assembly time, the assembler will output an error list file.

- (5) Create a directive file.

The directive file is a file to tell the linker how address allocation to segments in the input module should be made. Therefore, if you wish to expand the default ROM/RAM area or define a new memory area for address allocation to segments, a directive file must be created. If a specific address on memory must be allocated to a segment which is not defined as an absolute segment in the source module, a directive file must also be created.

A directive file can be input to the linker by specifying the -D option at linkage time.

Example 1: To expand default RAM area with device (uPD78310 or uPD78310A) which has no internal (on-chip) ROM

Enter the following in the directive file:

```
MEMORY RAM : (DE00H,20FFH)
```

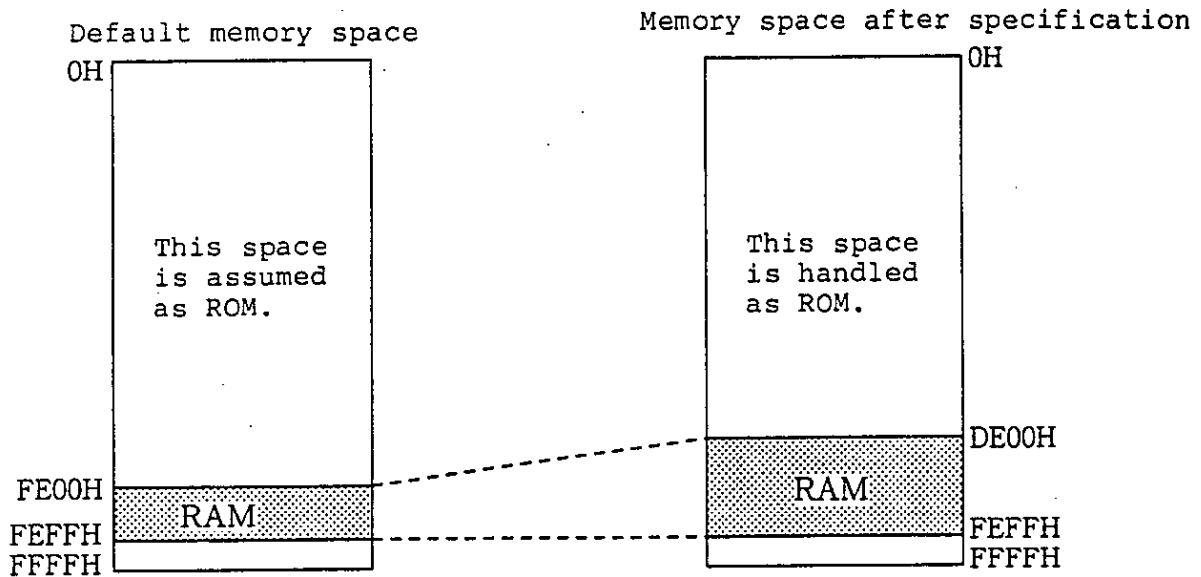


Fig. 3-2. Link Directive (1)

Example 2: To expand default ROM/RAM area with device (uPD78312 or uPD78312A) which has internal ROM and allocate segment CSEG1 to address 2000

Enter the following in the directive file:

```
MEMORY ROM : (0H,3FFFH)
MEMORY RAM : (D000H,2EFFH)
MERGE CSEG1 : AT(2000H)
```

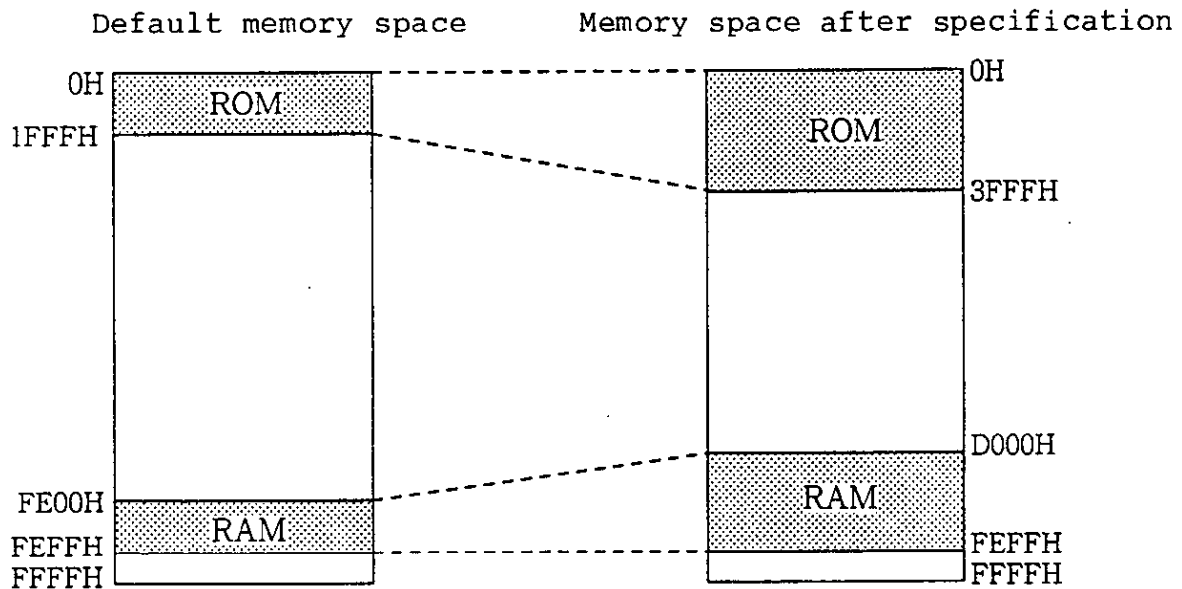


Fig. 3-3. Link Directive (2)

(6) Link the two object module files "78K3MAIN.REL" and "78K3SUB.REL" which have been output by the assembler in steps (1) and (3) above, respectively. Also input the directive file "78K3.DR".

o Enter the start-up command line of the linker as follows:

```
A>lk78k3 78k3main.rel 78k3sub.rel -d78k3.dr -o78k3.lnk -p78k3.map -g
```

└─ This may be omitted if no
directive file is required.

o The following message will be output to the console:

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx  
  
Link complete,      0 error(s) and      0 warning(s) found.
```

(7) Check the contents of drive A.

Two files "78K3.LNK" (load module file) and "78K3.PRN" (link list file) have been output by the linker.
If you specify the -E option at linkage time, the linker will output an error list file.

(8) Convert the load module file which have been output by the linker in step (6) above, into a HEX-format object file.

o Enter the start-up command line of the object converter as follows:

```
A>oc78k3 78k3.lnk
```

o The following message will be output to the console:

```
uCOM-78K/III Object Converter Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx  
  
Object Converter complete,      0 error(s) and      0 warning(s) found.
```

(9) Check the contents of drive A.

Two files "78K3.HEX" (HEX-format object module file) and "78K3.SYM" (symbol table file) have been output by the object converter.

(10) Create a library file.

Register the file "78K3SUB.REL" (object module file) output by the assembler as a library file.

o Enter the start-up command line of the librarian as follows:

```
A>lb78k3 <78k3.job
```

o The following message will be output to the console.

```
uCOM-78K/III Librarian Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx  
*create 78k3.lib  
*add 78k3.lib 78k3sub.rel  
*exit
```

(11) Check the contents of drive A.

A library file named "78K3.LIB" has been output by the librarian.

(12) Create an absolute assembly list file.

To create an absolute assembly list file for the main routine "78K3MAIN.ASM", input three files "78K3MAIN.REL", "78K3MAIN.ASM", and "78K3.LNK" to the list converter.

o Enter the start-up command line of the list converter as follows:

```
A>lcnv78k3 78k3main -l78k3.lnk
```

- o The following message will be output to the console.

```
List Conversion Program for RA78K/III Vx.xx  [xx xxx xx]  
  Copyright (C) NEC Corporation xxxx  USxxxxxxxxxx
```

```
Pass1: start...
```

```
Pass2: start...
```

```
Conversion complete.
```

- (13) Check the contents of drive A.

An absolute assembly list file named "78K3MAIN.P" has been output by the list converter.

3.3 Summary of Assembler Package Execution Procedure

A summary of the procedure for executing the assembler package on the sample program files is as illustrated in Fig. 3-4 below.

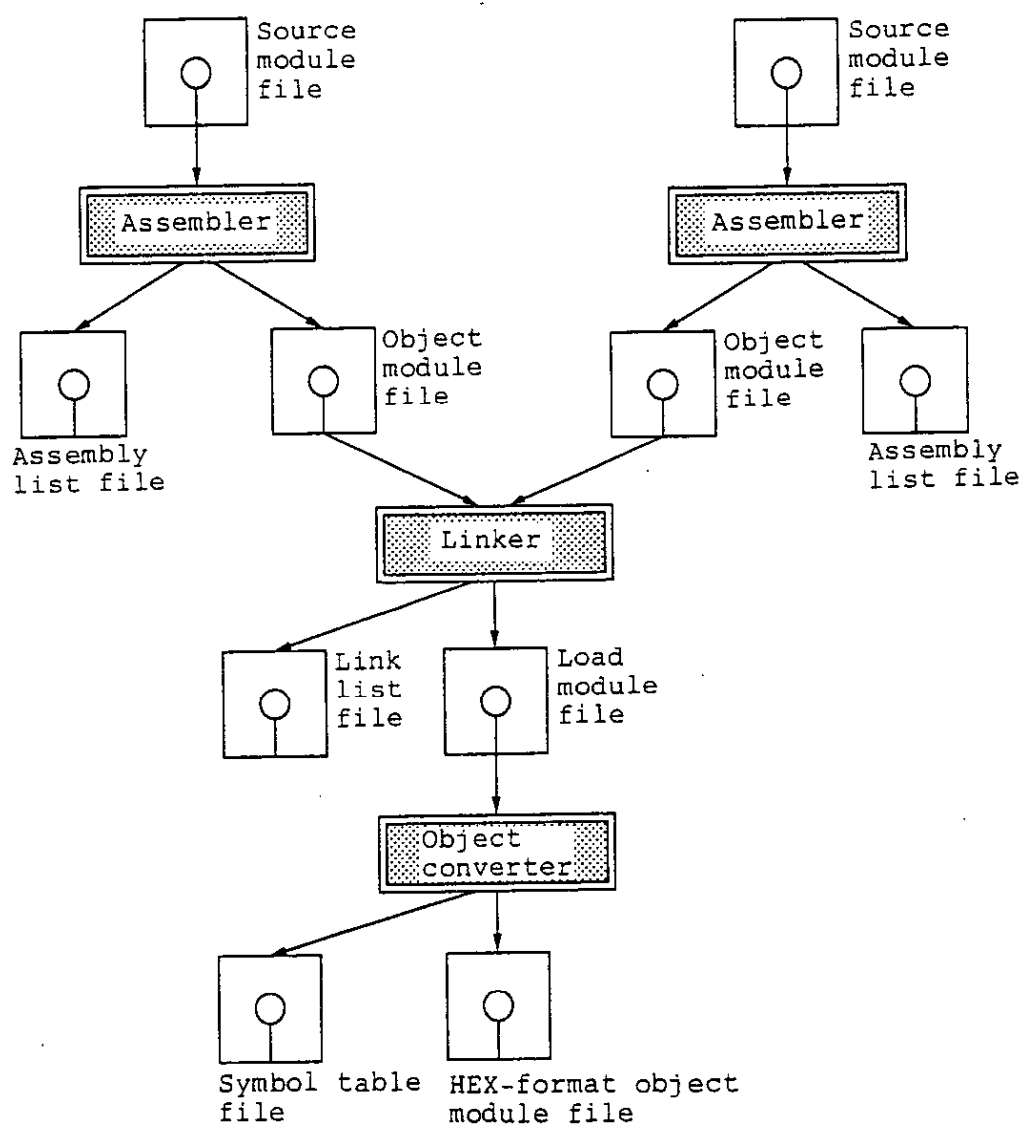


Fig. 3-4 (1). Assembler Package Execution Procedure

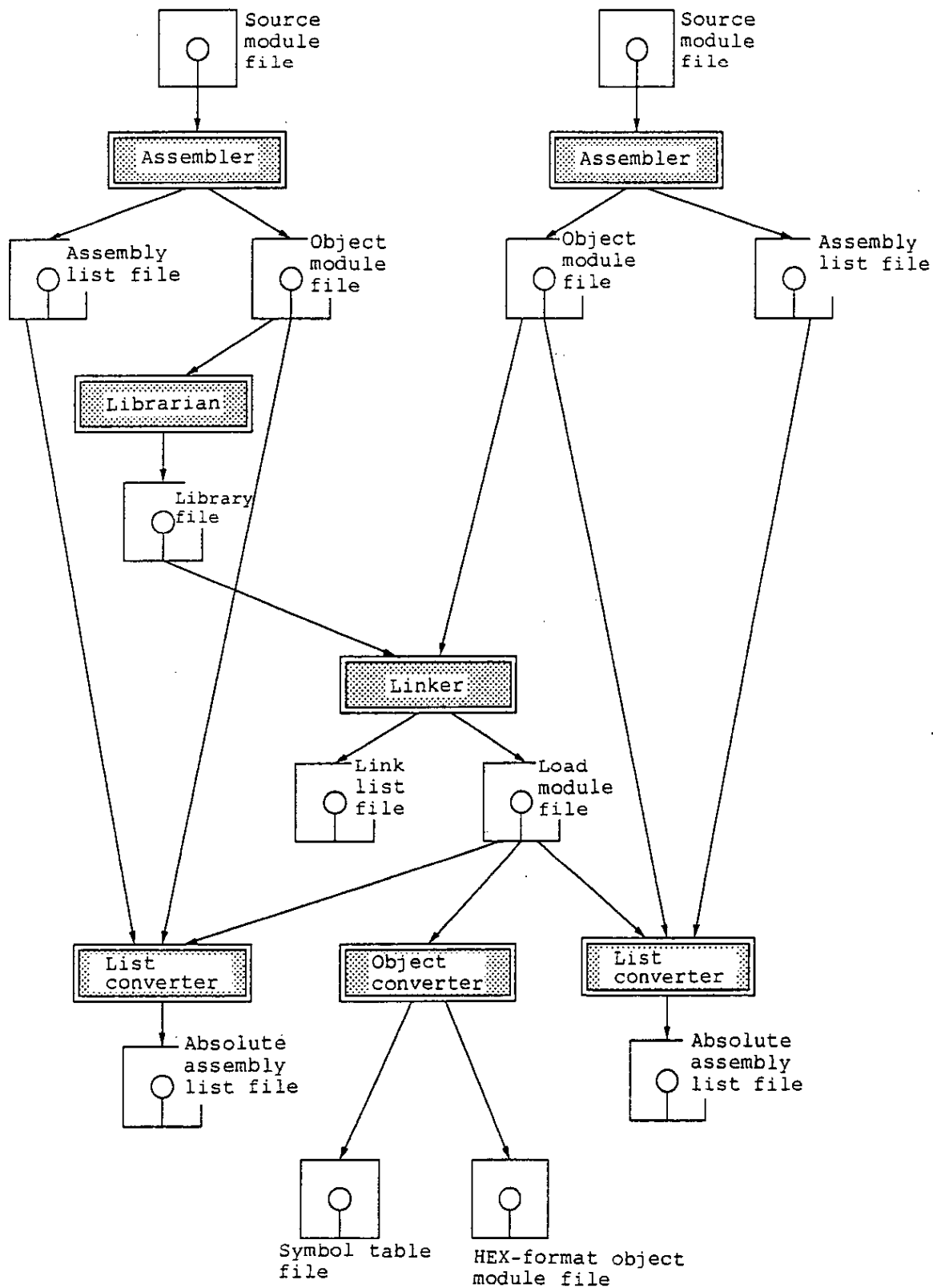


Fig. 3-4 (2). Assembler Package Execution Procedure

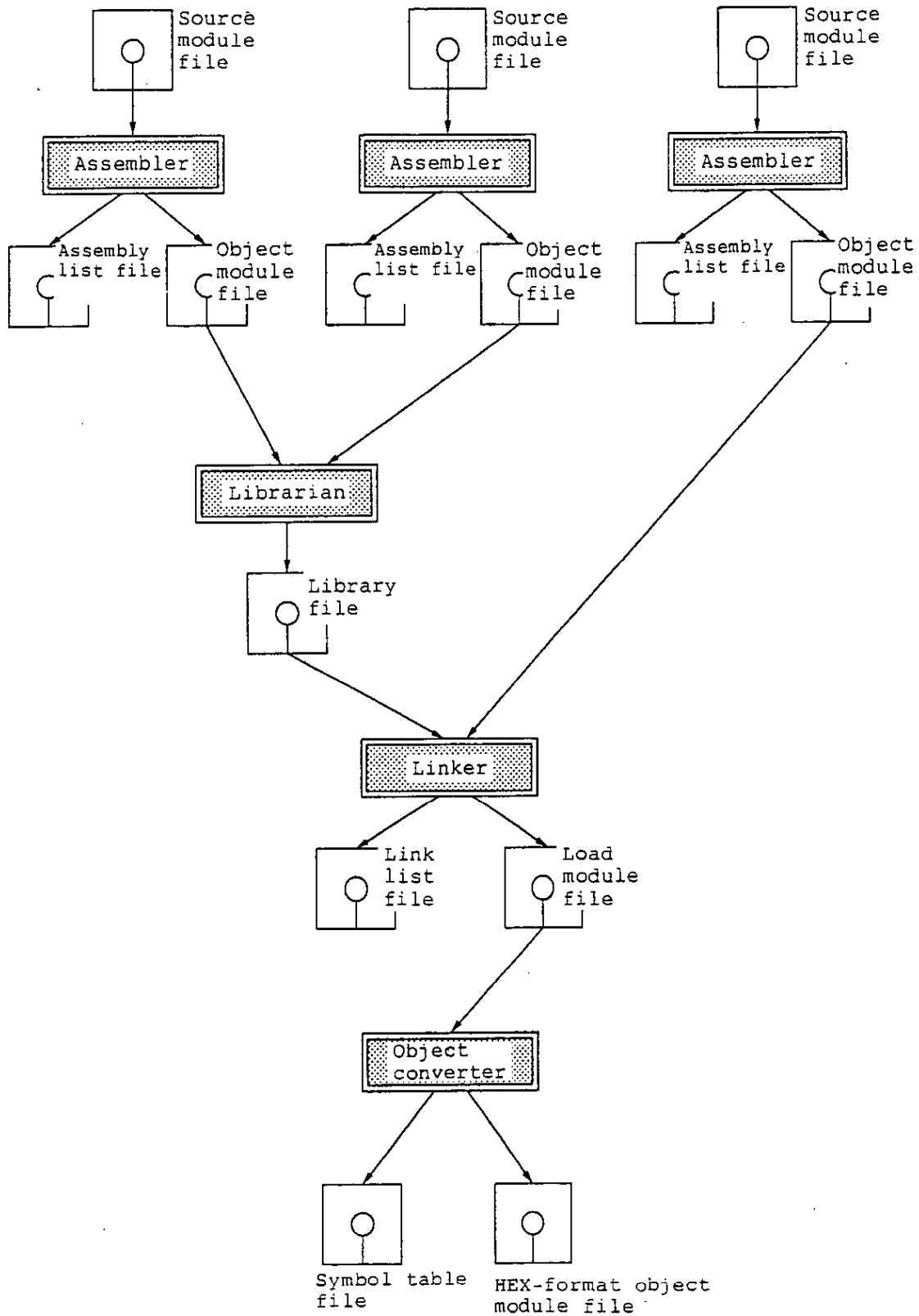


Fig. 3-4 (3). Assembler Package Execution Procedure

CHAPTER 4. ASSEMBLER

The assembler accepts source module files coded in the assembly language for the 78K series microcomputers as input files, translates the assembly language coding of each source module into machine language coding, and outputs each source module file as an object module file.

In addition, the assembler outputs list files such as an assembly list file and an error list file. If any error is found during the assembly of an input source module file, the assembler outputs the error message on an assembly list file or error list file to clearly indicate the cause of the error.

4.1 Input/Output Files of Assembler

The files listed in Table 4-1 below are input and output to and from the assembler.

Table 4-1. I/O Files of Assembler

| Type | Name and description of file | Default file type |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| Input files | <u>Assembler source module file</u> A source module file coded in the assembly language for the 78K series that must be converted (translated) into machine language before use. This file must be created by the user. | .ASM |
| | <u>Include file</u> A file to be referenced in an assembler source file. This file must be created by the user and must have been coded in the assembly language for the 78K series. | None |
| | <u>Parameter file</u> ... A file containing the parameters of the executable program (Assembler). This file must be created by the user. | .PRA |
| Output files | <u>Object module file</u> A binary file which contains machine language information, relocation information on the address of each machine-coded instruction, and symbol information. | .REL |
| | <u>Assembly list file</u> A file containing assembly information such as assembly list and cross-reference list. | .PRN |
| | <u>Error list file</u> A file containing information on errors at assembly time. | .ERA |

Table 4-1. I/O Files of Assembler (contd)

| Type | Name and description of file | Default file type |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| I/O file | Temporary file A file which is automatically generated by the assembler as a work file. This file will be erased on completion of the assembly process. | RA78Kn.\$\$1 or RA78Kn.\$\$2 (fixed to this name) (n = 0, 1, 2, 3, or 6) |

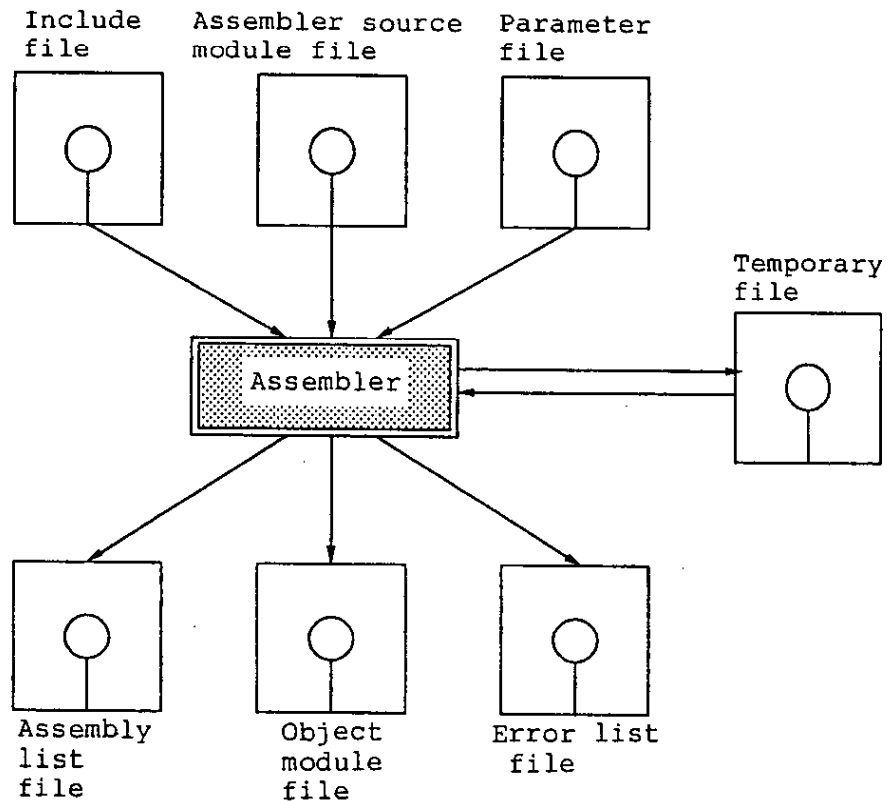


Fig. 4-1. I/O Files of Assembler

4.2 Assembler Functions

- (1) The assembler reads a source module file and translates the assembly language of the source module into machine language.
- (2) If any error related to the system or file I/O is found in the input source module, the assembler outputs an abort error message. If any coding error is found in the input source module, the assembler outputs a fatal error message or warning error message. If an abort error message or fatal error message is output, no object module file will be produced by the assembler. However, if the -J option has been specified at assembly time, the assembler will output an object module file even when a fatal error exists in the input source module file.
- (3) The assembler performs assembly processing according to the assembler option specified in the start-up command line of the assembler. See Section 4.4 for the assembler options.
- (4) The assembler outputs an execution end message when its processing has been completed normally and returns control to the OS.
- (5) The maximum performance characteristics of the assembler are as shown below.

| Item | | Maximum value |
|----------------------------------------------------------------------|------------------|----------------|
| Symbol length | w/o -S option | 8 characters |
| | with -S option | 31 characters |
| Number of characters that can be described per line of source module | | 130 characters |
| Number of segments | ?ASEG | 20 segments |
| | other than ?ASEG | 80 segments |

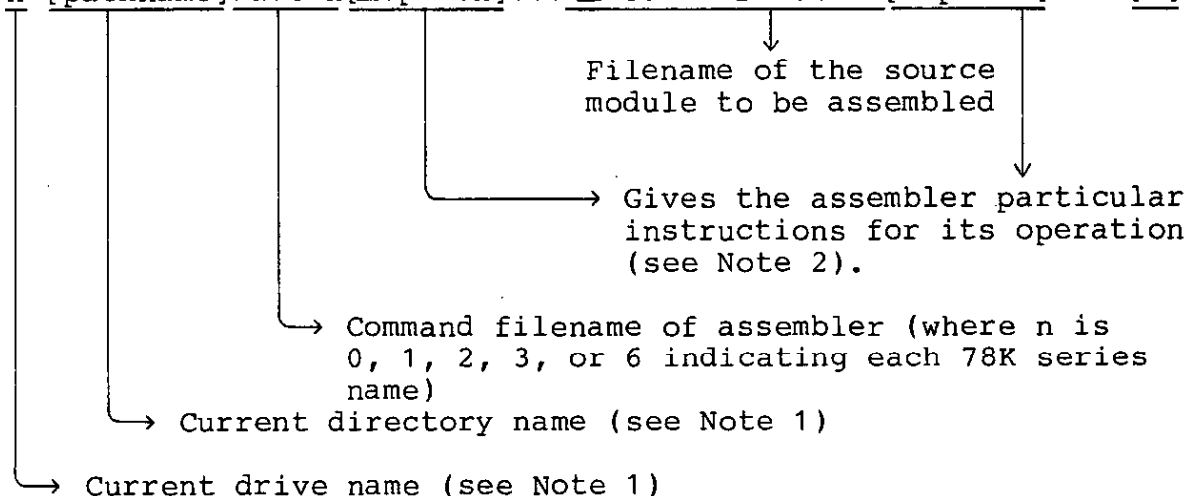
4.3 How to Start Up the Assembler

4.3.1 Starting up the assembler

The assembler can be started up (invoked) in either of the following two ways:

(1) Start-up with the start-up command line of the assembler

X>[pathname]ra78kn[Δoption]... Δsource-filename[Δoption]... [Δ]



Example: A>ra78k3 -g 78k3main.asm -e -np

- NOTE: 1. With MS-DOS V3.10, the command file and overlay files of the assembler must have been stored in the same directory.
2. If two or more assembler options are to be specified, each assembler option must be delimited with a space. See Section 4.4 for details of the assembler options.

(2) Start-up with parameter file

A parameter file is used when all the required information for starting up the assembler cannot be specified in the start-up command line of the assembler or when the same assembler options are to be used repeatedly in each assembly process.

When using this parameter file, the `-F` option must be specified in the start-up command line of the assembler to specify the use of the parameter file.

The assembler can be started up with a parameter file as follows:

X>ra78kn[Δ source-filename] Δ -f parameter-filename

→ File containing information
required to start up the
assembler

→ Option specifying parameter file

- o A parameter file must be created with the editor.
- o The description format of parameters within the parameter file is as shown below.

[[Δ]option[Δ option].. [Δ] Δ]] ...

- o If the source module filename to be input is omitted in the start-up command line of the assembler, only one input source module filename can be described within the parameter file.
- o The input source module filename may be described either before or after an option.
- o In the parameter file, all the assembler options and output filename which should normally be specified in the start-up command line must be described.
For details of the parameter file, see Subsection 4.4.3, "Description of each assembler option".

Example: To create parameter file "78K3MAIN.PRA" with the editor

- o Contents of 78K3MAIN.PRA

```
;parameter file
78k3main.asm -osample.rel -g
-psample.prn
```

- o Start-up of assembler using parameter file "78K3MAIN.PRA"

A>ra78k3 -f78k3main.pra

4.3.2 Execution start and end messages

(1) Execution start message

When the assembler is started up, the following message is output to the console, indicating the start of the assembler execution.

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

(2) Execution end messages

- o If no assembly error is found as a result of an assembly operation, the assembler will output the following message to the console and return control to the OS.

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete.      0 error(s) and      0 warning(s) found.
```

- o If any assembly errors are found as a result of an assembly operation, the assembler will output the following message (the number of errors found) to the console and return control to the OS.

```
Pass1 Start  
78k3MAIN.ASM(20) : E101 Syntax error  
Pass2 Start  
78k3MAIN.ASM(20) : E101 Syntax error
```

```
Assembly complete.      1 error(s) and      0 warning(s) found.
```

- o If any fatal error is found during an assembly operation, the assembler will output the following message to the console, stop its processing, and return control to the OS.

Example 1:

```
A>ra78k3 sample.asm
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
A006 File not found 'SAMPLE.ASM'  
Program aborted.
```

In this example, the assembly operation was discontinued by an abort error resulting from the specification of a source module file which does not exist in drive A.

Example 2:

```
A>ra78k3 78k3main.asm -w
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
  Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
A018 Option is not recognized '-w'  
Program aborted.
```

In this example, the assembly operation was discontinued by an abort error resulting from the input of an assembler option "-W (-w)" which is not recognized by the assembler.

If the assembly program is aborted following the output of an error message, check the cause of the error by referring to Chapter 11, Error Messages and take corrective action(s) as required.

4.4 Assembler Options

4.4.1 Types of assembler options

An assembler option gives the assembler particular instructions for its operation and is broadly divided into the following 14 types:

Table 4-2. Types of Assembler Options

| No. | Classification | Option | Functional description |
|-----|----------------------------------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------|
| 1 | Option for processor type specification | -C | Specifies the processor type indicating the target device subject to assembly. |
| 2 | Options for object module output specification | -O | Specifies the output or non-output of an object module file. |
| | | -NO | |
| 3 | Options for forced object module output specification | -J | Specifies the output or non-output of an object module file by force. |
| | | -NJ | |
| 4 | Options for debug information output specification | -G | Specifies the output or non-output of symbol information for debugging to the object module. |
| | | -NG | |
| 5 | Options for symbol name length specification | -S | Specifies the extension or non-extension of the symbol name length. |
| | | -NS | |
| 6 | Options for symbol name upper-/lower-case specification | -CA | Specifies whether or not the symbol name in lower-case letters is to be distinguished from that in lowercase letters. |
| | | -NCA | |
| 7 | Option for Include file read path specification | -I | Tells the assembler to read Include file(s) from the path(s) specified by this option. |
| 8 | Options for assembly list file output specification | -P | Specifies the output or non-output of an assembly list file. |
| | | -NP | |
| 9 | Options for assembly list file information specification | -KA | Specifies the output or non-output of an assembly list in the assembly list file. |
| | | -NKA | |
| | | -KS | Specifies the output or non-output of a symbol list in the assembly list file. |
| | | -NKS | |
| | | -KX | Specifies the output or non-output of a cross-reference list in the assembly list file. |
| | | -NKX | |

Table 4-2. Types of Assembler Options (contd)

| No. | Classification | Option | Functional description |
|-----|-------------------------------------------------------|--------|------------------------------------------------------------------------------------------------------|
| 10 | Options for assembly list file format specification | -LW | Specifies the number of print columns per line of an assembly list file. |
| | | -LL | Specifies the number of print lines per page of an assembly list file. |
| | | -LH | Specifies the output of a specified character string to the header of an assembly list file. |
| | | -LT | Specifies the number of columns for tabulation. |
| | | -LF | Specifies the addition or non-addition of a form-feed (FF) code to the end of an assembly list file. |
| | | -NLF | |
| 11 | Options for error list file output specification | -E | Specifies the output or non-output of an error list file. |
| | | -NE | |
| 12 | Option for parameter file specification | -F | Specifies the input of input filename and options from the file specified by this option. |
| 13 | Option for temporary file creation path specification | -T | Specifies the creation of a temporary file on the path specified by this option. |
| 14 | Option for HELP message output specification | -- | Specifies the output of HELP message to the console. |

The above table merely introduces all the available assembler options. Each of these assembler options is detailed in Subsection 4.4.3 below. For quick reference, see Appendix C.1, List of Assembler Options in which the description format of each option and the relationship between one option and the other are also outlined.

4.4.2 Priority of assembler options

Table 4-3 shows the precedence order of assembler options when two or more options are specified at the same time.

Table 4-3. Priority of Assembler Options

| | -NO | -NP | -NKA | -NKS | -XX | -NKX | -- |
|-----|-----|-----|------|------|-----|------|----|
| -J | x | | | | | | x |
| -G | x | | | | | | x |
| -P | | | △ | △ | | △ | x |
| -KA | | x | | | | | x |
| -KS | | x | | | x | | x |
| -KX | | x | | | | | x |
| -LW | | x | | | | | x |
| -LL | | x | | | | | x |
| -LH | | x | | | | | x |
| -LT | | x | | | | | x |
| -LF | | x | | | | | x |

In Table 4-3, "X" in the table indicates that the option on the leftmost column becomes invalid if the option on the top column is specified.

Example: A>ra78k3 -c310 78k3main.asm -np -lw80 -lf

In this case, the -LW and -LF options become invalid (because the -NP option has been specified at the same time).

"△" in the table indicates that the option on the leftmost column becomes invalid if all of the options on the top column are specified at the same time.

Example: A>ra78k3 -c310 78k3main.asm -p -nka -nks -nkx

In this case, the -P option becomes invalid, because the -NKA, -NKS, and -NKX have been specified at the same time.

If two options contradicting each other (such as -O and -NO, -P and -NP, etc.) are specified at the same time, whichever you specified later will take precedence over the other.

Example: A>ra78k3 -c310 78k3main.asm -o -no

In this example, the -NO option takes precedence over the -O option which has been specified before the -NO option.

The assembler options not listed in Table 4-3 are not affected by any other assembler options. However, if the "--" option (for HELP message output specification) is specified, all the other options specified at the same time with the "--" option become invalid.

4.4.3 Description of each assembler option

A detailed description of each assembler option is provided in this subsection.

(1) Option for processor type specification (-C)

Description format: -C processor-type

Default assumption: This option cannot be omitted.

Function

The -C option specifies the processor type indicating the target device subject to assembly.

Use

This option must always be specified. The assembler performs assembly on the specified target device and generates object codes corresponding to the target device.

Explanation

Any of the following target devices can be specified with the -C option:

Table 4-4. List of Target Devices

| Series name | Target device name | Processor type |
|-------------|---------------------|----------------|
| 78K/0 | uPD78012 | 012 |
| | uPD78014, uPD78P014 | 014 |
| 78K/I | uPD78112, uPD78P112 | 112 |
| | uPD78134, uPD78P134 | 134 |
| | uPD78136 | 136 |
| | uPD78138, uPD78P138 | 138 |
| 78K/II | uPD78210 | 210 |
| | uPD78212 | 212 |
| | uPD78213 | 213 |
| | uPD78214, uPD78P214 | 213 |
| | uPD78220 | 220 |
| | uPD78224, uPD78P224 | 224 |
| | uPD78233 | 233 |
| | uPD78234, uPD78P234 | 234 |

Table 4-4. List of Target Devices (contd)

| Series name | Target device name | Processor type |
|-------------|-----------------------|----------------|
| 78K/III | uPD78310 | 310 |
| | uPD78312, uPD78P312 | 312 |
| | uPD78310A | 310A |
| | uPD78312A, uPD78P312A | 312A |
| | uPD78320 | 320 |
| | uPD78322, uPD78P322 | 322 |
| | uPD78330 | 330 |
| | uPD78332, uPD78P332 | 332 |
| 78K/VI | uPD78600 | 600 |
| | uPD78602 | 602 |

NOTE

The -C option is an option which cannot normally be omitted from specification. However, by describing a control instruction which has the same function as the -C option in the header of the input source module, this option specification can be omitted in the start-up command line of the assembler. The description formats of this control instruction are as shown below.

| |
|-----------------------------------------------------------------------------------------------------|
| <p>▲\$ ▲PROCESSOR ▲(▲processor-type ▲)</p> <p>▲\$ ▲PC ▲(▲processor-type ▲) ; abbreviated format</p> |
|-----------------------------------------------------------------------------------------------------|

See Chapter 4, Control Instructions in the "RA78K Series Assembler Package User's Manual for Language" for details of this control instruction.

Application Examples

Example 1: To assemble the source program of the uPD78310, specify this option in the command line as follows:

```
A>ra78k3 -c310 78k3main.asm
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

Example 2: To start up the assembler by describing the processor type in the source module header

```
$      PROCESSOR(310)  
  
      NAME      SAMP  
;*****  
;*  
;*      HEX -> ASCII Conversion Program      *  
;*  
;*      main-routine      *  
;*  
;*****  
  
      PUBLIC  MAIN, START  
      EXTRN   CONVAH  
  
      ;
```

In this case, the processor type specification in the command line may be omitted.

A>ra78k3 -c310 78k3main.asm

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

Example 3: To start up the assembler by specifying a target device different from that specified in the source module header

```
$      PROCESSOR(312)

      NAME      SAMPM
;*****
;*
;*      HEX -> ASCII Conversion Program      *
;*
;*              main-routine                  *
;*
;*****

      PUBLIC  MAIN, START
      EXTRN   CONVAH

      :
```

A>ra78k3 -c320 78k3main.asm

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
W321 Duplicate option
Pass2 Start
W321 Duplicate option

Assembly complete, 0 error(s) and 1 warning(s) found.

In this case, the target device specified in the command line takes precedence over that specified in the source module header.

(2) Options for object module file output specification (-O/-NO)

| |
|----------------------------------------------------------------------------------------------------|
| Description format: -O [output-filename] or -NO Default assumption: -O input-filename.REL |
|----------------------------------------------------------------------------------------------------|

Function

- o The -O option specifies the output of an object module file. It also specifies the output destination or output filename of the object module file to be output by the assembler.
- o The -NO option specifies the non-output of an object module file.

Use

- o Use the -O option if you want to change the output destination or output filename of an object module file.
- o If the assembly of a source module is to be performed only to output an assembly list, use the -NO option. This will reduce the assembly time.

Explanation

- o If a fatal error is found during an assembly operation with the -O option specified, no object module file will be output by the assembler.
- o If a drive name is omitted from the -O option specification, the object module file will be output to the current drive.
- o If an output filename is omitted from the -O option specification, "input filename.REL" is assumed as the output filename.
- o If the -O and -NO options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To output an object module file named "SAMPLE.REL"

```
A>ra78k3 -c310 78k3main.asm -osample.rel
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

Example 2: To assemble the source program with both -NO and -O options specified

```
A>ra78k3 -c310 78k3main.asm -no -o
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

In this case, -NO option becomes invalid and -O option is accepted as valid.

-J/-NJ

**Forced object module file
output specification**

(3) Options for forced object module file output specification
(-J/-NJ)

| | |
|---------------------|-----|
| Description format: | -J |
| | or |
| | -NJ |
| Default assumption: | -NJ |

Function

- o The -J option tells the assembler to output an object module file even if a fatal error occurs during an assembly operation.
- o The -NJ option is used to invalidate the -J option.

Use

Normally, the specified object module file will not be output if any fatal error occurs during the assembly operation. Therefore, if you want to execute the program even in case of a fatal error, use the -J option to output an object module file.

Explanation

- o If a fatal error occurs during an assembly operation with the -J option specified, an object module file will be output by the assembler.
- o If the -J and -NJ options are specified at the same time, whichever you specified later will take precedence over the other.

`-J/-NJ`

Forced object module file
output specification

Application Examples

Example 1: To output an object module file even in case of a
fatal error

A>ra78k3 -c310 78k3main.asm -j

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

(4) Options for debug information output specification (-G/-NG)

```

Description format: -G
                   or
                   -NG
Default assumption: -NG

```

Function

- o The `-G` option specifies the addition of debugging information (local symbol information) to an object module file to be output by the assembler.
- o The `-NG` option specifies the non-addition of debugging information to an object module file.

Use

If the -G option is not specified, the line numbers and symbol information required for a symbol table file which becomes an input file to the source debugger will not be added to the output object module file. Therefore, when performing debugging at the source level, assemble all the modules to be linked by specifying the -G option.

Explanation

- o If the -G option is omitted, the -NG option is assumed and thus no debugging information will be output.
- o If the -G and -NG options are specified at the same time,, whichever you specified later will take precedence over the other.

Note

A control instruction which has the same function as the -G or -NG option can be described in the header of the input source module file. The description formats of these control instructions are as shown below.

| | |
|--------------|----------------------|
| ▲\$ ▲DEBUG | |
| ▲\$ ▲DG | ; abbreviated format |
| ▲\$ ▲NODEBUG | |
| ▲\$ ▲NODG | ; abbreviated format |

See Chapter 4 in the "RA78K Series Assembler Package User's Manual for Language" for details of these control instructions.

Application Examples

Example 1: To assemble the source program with -G option specified to add debugging information to an object module file

```
A>ra78k3 -c310.78k3main.asm -g
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

(5) Options for symbol name length specification (-S/-NS)

| |
|----------------------------------------------------------------|
| Description format: -S or -NS Default assumption: -NS |
|----------------------------------------------------------------|

Function

- o The -S option tells the assembler to extend the length of a symbol name that can be recognized by the assembler to a maximum of 31 characters.
- o The -NS option tells the assembler not to extend the symbol name length.

Use

If you want to extend the symbol name length to more than eight characters, use the -S option so that the number of characters that can be recognized as a symbol name may be extended up to 31 characters.

Explanation

- o If the -S option is specified, the assembler will recognize up to 31 characters as a symbol name and output the symbol information to the object.
- o If the -S option is omitted, the -NS option is assumed and the assembler will recognize up to eight characters as a symbol name.
- o If the -S and -NS options are specified at the same time, whichever you specified later will take precedence over the other.
- o When using the source debugger, do not specify the -S option.

Application Examples

Example 1: To assemble the source program with -S option
specified to extend symbol name length to 31
characters max.

A>ra78k3 -c310 78k3main.asm -s

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

`-CA/-NCA`

Symbol name uppercase/lowercase
specification

(6) Options for symbol name uppercase/lowercase specification
(`-CA/-NCA`)

| |
|---------------------------------------------------------------------------------------------------------|
| Description format: <code>-CA</code> or <code>-NCA</code> Default assumption: <code>-CA</code> |
|---------------------------------------------------------------------------------------------------------|

Function

- o The `-CA` option tells the assembler not to distinguish between symbol names described (partly or wholly) in uppercase letters and those described (partly or wholly) in lowercase letters.
- o The `-NCA` option tells the assembler to distinguish symbol names described in uppercase letters from those described in lowercase letters.

Use

Use the `-CA` option if no distinction need to be made between symbol names written in uppercase letters and those in lowercase letters for symbol information output.

Explanation

- o If the `-CA` option is specified, the assembler will output symbol information to the object module file by converting lowercase letters in symbol names to their uppercase equivalents.
- o If the `-NCA` option is specified, the assembler will output symbol information to the object module file without converting lowercase letters in symbol names to their uppercase equivalents.
- o If the `-CA` and `-NCA` options are specified at the same time, whichever you specified later will take precedence over the other.

-CA/-NCA

Symbol name uppercase/lowercase
specification

Application Examples

Example 1: To assemble the source program with -NCA option
specified to distinguish between uppercase and
lowercase letters of symbol names

A>ra78k3 -c310 78k3main.asm -nca

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

(7) Option for Include file read path specification (-I)

Description format: -I pathname[, pathname] ...

Default assumption: Path specified by environment
variable INC78Kn (where n=0, 1, 2,
3, or 6) or current path in which
source file exists

Function

The -I option tells the assembler to input the Include file(s) specified by `$include` in the source module from the path(s) specified by this option.

Use

Use the -I option if you want to search an Include file from a path.

Explanation

- o Two or more path names may be specified with the -I option by delimiting each pathname with "," (comma).
- o No blank (space) is allowed before and after the delimiter ",".
- o If two or more pathnames are input following the -I option or if two or more -I options are specified at the same time, the assembler will search the specified paths in their order of specification, for the file(s) specified by `$include`.
- o If other than pathnames are input following the -I option or if no pathname is specified, an abort error will result.
- o If nine or more -I options are specified at the same time (in the same command line), an abort error will also result.

-I

Include file read path specification

Application Examples

Example 1: To assemble the source program with -I option
specified to read an Include file from directory
"SAMPLE"

A>ra78k3 -c310 78k3main.asm -ib:Ysample

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

(8) Options for assembly list file output specification (-P/-NP)

Description format: -P [output-filename]

or

-NP

Default assumption: -P input-filename.PRN

Function

- o The -P option specifies the output of an assembly list file. It also specifies the output destination or output filename of the assembly list file to be output by the assembler.
- o The -NP option specifies the non-output of an assembly list file.

Use

- o Use the -P option if you want to change the output destination or output filename of an assembly list file.
- o If the assembly of a source module is to be performed only to output an object module file, use the -NP option. This will reduce the assembly time.

Explanation

- o An output filename can be specified with either a disk type filename or a device type filename. Only the following device type filenames can be used with this option: CON, PRN, NUL, and AUX. If "CLOCK" is specified as an output filename, an abort error will result.
- o If an output filename is omitted from the -P option specification, "input filename.PRN" is assumed as the output assembly list filename.
- o If a drive name is omitted from the -P option specification, the assembly list file will be output to the current drive.
- o If the -P and -NP options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To assemble the source program with -P option specified to create an assembly list file named "SAMPLE.PRN"

```
A>ra78k3 -c310 78k3main.asm -psample.prn
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

Example 2: To assemble the source program with -P option specified to output an assembly list file to PRN (printer).

```
A>ra78k3 -c310 78k3main.asm -pprn
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

-KA/-NKA

Assembly list file information
specification

(9) Options for assembly list file information specification
(-KA/-NKA, -KS/-NKS, -KX/-NKX)

(a) Options for assembly list output specification (-KA/-NKA)

| |
|------------------------------------------------------------------|
| Description format: -KA or -NKA Default assumption: -KA |
|------------------------------------------------------------------|

Function

- o The -KA option specifies the output of an assembly list to the assembly list file.
- o The -NKA option specifies the non-output of an assembly list to the assembly list file.

Use

Use the -KA option if you want to have only an assembly list as the contents of an assembly list file.

Explanation

- o If the -KA and -NKA options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -NKA option is specified together with the -NKS and -NKX options, no assembly list file will be output.

-KA/-NKA

Assembly list file information
specification

Application Examples

Example 1: To assemble the source program with -KA option
specified to output an assembly list

A>ra78k3 -c310 78k3main.asm -ka -lw80

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

- o When assembly list file "78K3MAIN.PRN" is referenced, you will find that the following assembly list has been output to the assembly list file.

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c310 78k3main.asm -ka -lw80
Para-file:
In-file: 78K3MAIN.ASM
Obj-file: 78K3MAIN.REL
Prn-file: 78K3MAIN.PRN

Assemble list

| ALNO | STNO | ADRS | OBJECT | M I | SOURCE STATEMENT |
|------|------|---------------|--------|-----|---------------------------------|
| 1 | 1 | | | | \$ PROCESSOR(310) |
| 2 | 2 | | | | |
| 3 | 3 | | | | NAME SAMPM |
| 4 | 4 | | | | ***** |
| 5 | 5 | | | | ; |
| 6 | 6 | | | | HEX -> ASCII Conversion Program |
| 7 | 7 | | | | ; |
| 8 | 8 | | | | main-routine |
| 9 | 9 | | | | ; |
| 10 | 10 | | | | ***** |
| 11 | 11 | | | | |
| 12 | 12 | | | | PUBLIC MAIN, START |
| 13 | 13 | | | | EXTRN CONVAH |
| 14 | 14 | | | | |
| 15 | 15 | ---- | | | DATA DSEG AT 0FE20H |
| 16 | 16 | FE20 | | | HDTSA: DS 1 |
| 17 | 17 | FE21 | | | STASC: DS 2 |
| 18 | 18 | | | | |
| 19 | 19 | ---- | | | CODE CSEG AT 0H |
| 20 | 20 | 0000 R0000 | | | MAIN: DW START |
| 21 | 21 | | | | |
| 22 | 22 | ---- | | | CSEG |
| 23 | 23 | 0000 2B4100 | | | START: MOV RFM, #00 |
| 24 | 24 | 0003 0BFC80FE | | | MOVW SP, #0FE80H |
| 25 | 25 | 0007 2B4000 | | | MOV MM, #00 |
| 26 | 26 | 000A 0944F708 | | | MOV STBC, #08H |
| 27 | 27 | | | | |
| 28 | 28 | 000E 3A201A | | | MOV HDTSA, #1AH |
| 29 | 29 | 0011 6720FE | | | MOVW HL, #HDTSA |

(b) Options for symbol list output specification (-KS/-NKS)

| |
|--------------------------|
| Description format: -KS |
| or |
| -NKS |
| Default assumption: -NKS |

Function

- o The -KS option specifies the output of a symbol list to the assembly list file following the output of an assembly list.
- o The -NKS option specifies the non-output of a symbol list to the assembly list file.

Use

Use the -KS option if you want to have an assembly list and a symbol list as the contents of an assembly list file.

Explanation

- o If the -KS and -NKS options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -KS and -KX options are specified at the same time, the -KS option will be ignored.
- o If the -NKS option is specified together with the -NKA and -NKX options, no assembly list file will be output.

Application Examples

Example 1: To assemble the source program with -KS option
specified to output a symbol list

A>ra78k3 -c310 78k3main.asm -ks -lw80

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

- o When assembly list file "78K3MAIN.PRN" is referenced, you
will find that the following symbol list has been output next
to an assembly list.

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 3

Symbol Table List

| VALUE | ATTR | RTYP | NAME | VALUE | ATTR | RTYP | NAME |
|-------|------|------|--------|-------|------|------|-------|
| | CSEG | | ?CSEG | | CSEG | | CODE |
| ---- | | EXT | CONVAH | | DSEG | | DATA |
| FE20H | ADDR | | HDTSA | 0H | ADDR | PUB | MAIN |
| | MOD | | SAMPM | 0H | ADDR | PUB | START |
| FE21H | ADDR | | STASC | | | | |

-KX/-NKX

Assembly list file information
specification

(c) Options for cross-reference list output specification
(-KX/-NKX)

| |
|-------------------------------------------------------------------|
| Description format: -KX or -NKX Default assumption: -NKX |
|-------------------------------------------------------------------|

Function

- o The -KX option specifies the output of a cross-reference list to the assembly list file following the output of an assembly list.
- o The -NKX option specifies the non-output of a cross-reference list to the assembly list file.

Use

Use the -KX option to output a cross-reference list as the contents of an assembly list file if you want to know how often and where the respective symbols defined in the source module file have been referenced or at which line of the assembly list each symbol has been referenced.

Explanation

- o If the -KX and -NKX options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -KS and -KX options are specified at the same time, the -KS option will be ignored.
- o If the -NKX option is specified together with the -NKA and -NKS options, no assembly list file will be output.

Note

A control instruction which has the same function as the -KX or -NKX option may be specified in the header of the input source module file. The description formats of these control instructions are as shown below.

| | |
|-------------|----------------------|
| ▲\$▲ XREF | |
| ▲\$▲ XR | ; abbreviated format |
| ▲\$▲ NOXREF | |
| ▲\$▲ NOXR | ; abbreviated format |

See Chapter 4 in the "RA78K Series Assembler Package User's Manual for Language" for details of these control instructions.

Application Examples

Example 1: To assemble the source program with -KX option specified to output a cross-reference list

```
A>ra78k3 -c310 78k3main.asm -kx -lw80
uCOM-78K/III Assembler Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

- o When assembly list file "78K3MAIN.PRN" is referenced, you will find that the following cross-reference list has been output next to an assembly list.

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 3

Cross-Reference List

| NAME | VALUE | R | ATTR | RTYP | SEGNAME | XREFS |
|--------|-------|---|------|------|---------|------------|
| ?CSEG | | | CSEG | | ?CSEG | 22# |
| CODE | | | CSEG | | CODE | 19# |
| CONVAH | ----H | E | | EXT | | 130 31 |
| DATA | | | DSEG | | DATA | 15# |
| HDTSA | FE20H | | ADDR | | DATA | 16# 28 29 |
| MAIN | 0H | | ADDR | PUB | CODE | 120 20# |
| SAMPM | | | MOD | | | 3# |
| START | 0H | R | ADDR | PUB | ?CSEG | 120 20 23# |
| STASC | FE21H | | ADDR | | DATA | 17# 33 |

(10) Options for assembly list file format specification

(-LW, -LL, -LH, -LT, -LF/-NLF)

(a) Option for page width specification (-LW)

Description format: -LW [No. of columns per line]

Default assumption: -LW132 (-LW80 with output to console)

Function

- o The -LW option specifies the number of print columns per line of a list file.

Use

Use the -LW option if you want to change the number of print columns per line of any list file.

Explanation

- o The number of print columns per line to be specified with the -LW option must be within the following value range excluding the terminator (CR or LF):

$$72 \leq \text{No. of print columns per line} \leq 132$$

(For output to the console, the maximum value becomes 80 columns.)

If any value beyond this range or other than a value is specified with this option, an abort error will result.

- o If the number of columns per line is omitted, a value of 132 is assumed to have been specified. However, if the output destination of an assembly list file is the console, a value of 80 is assumed.
- o If the -LW option is specified at the same time with the -NP option, the -LW option will be ignored and thus will become invalid.

Application Examples

Example 1: To assemble the source program with -LW option omitted and -P option specified to output an assembly list file to the printer

```
A>ra78k3 -c310 78k3main.asm -pprn
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

- o When assembly list file "78K3MAIN.PRN" is referenced, the output assembly list will look like this.

Assemble list

| ALNO | STNO | ADRS | OBJECT | M I | SOURCE STATEMENT |
|------|------|---------------|--------|-----|---------------------------------|
| 1 | 1 | | | | \$ PC(310) |
| 2 | 2 | | | | |
| 3 | 3 | | | | NAME SAMPN |
| 4 | 4 | | | | |
| 5 | 5 | | | | ; |
| 6 | 6 | | | | HEX -> ASCII Conversion Program |
| 7 | 7 | | | | ; |
| 8 | 8 | | | | main-routine |
| 9 | 9 | | | | ; |
| 10 | 10 | | | | |
| 11 | 11 | | | | |
| 12 | 12 | | | | PUBLIC MAIN,START |
| 13 | 13 | | | | EXTRN CONVAH |
| 14 | 14 | | | | |
| 15 | 15 | ---- | | | DATA DSEG AT OFE20H |
| 16 | 16 | FE20 | | | HDTSA: DS 1 |
| 17 | 17 | FE21 | | | STASC: DS 2 |
| 18 | 18 | | | | |
| 19 | 19 | ---- | | | CODE CSEG AT OH |
| 20 | 20 | 0000 R0000 | | | MAIN: DW START |
| 21 | 21 | | | | |
| 22 | 22 | ---- | | | CSEG |
| 23 | 23 | 0000 2B4100 | | | START: MOV RFM,#00 |
| 24 | 24 | 0003 0BFC80FE | | | MOVW SP,#0FE80H |
| 25 | 25 | 0007 2B4000 | | | MOV MM,#00 |
| 26 | 26 | 000A 0944F708 | | | MOV STBC,#08H |
| 27 | 27 | | | | |
| 28 | 28 | 000E 3A201A | | | MOV HDTSA,#1AH |
| 29 | 29 | 0011 6720FE | | | MOVW HL,#HDTSA |
| 30 | 30 | | | | |
| 31 | 31 | 0014 R280000 | | | CALL !CONVAH |
| 32 | 32 | | | | |
| 33 | 33 | 0017 6521FE | | | MOVW DE,#STASC |
| 34 | 34 | 001A D3 | | | MOV A,B |
| 35 | 35 | 001B 50 | | | MOV [DE+],A |
| 36 | 36 | 001C D2 | | | MOV A,C |
| 37 | 37 | 001D 50 | | | MOV [DE+],A |
| 38 | 38 | | | | |
| 39 | 39 | 001E 14FE | | | BR \$\$ |
| 40 | 40 | | | | |
| 41 | 41 | | | | END |

;set hex 2-code data in HL register
 ;convert ASCII <- HEX
 ;output BC-register <- ASCII code
 ;set DE <- store ASCII code table

Example 2: To assemble the source program with -LW option
specified to set 80 columns as No. of columns per
page of an assembly list

```
A>ra78k3 -c310 78k3main.asm -lw80
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start
```

```
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

o In this case, the output assembly list will look like this.

Assemble list

| ALNO | STNO | ADRS | OBJECT | M I | SOURCE STATEMENT |
|------|------|------|----------|-----|---------------------------------|
| 1 | 1 | | | | \$ PC(310) |
| 2 | 2 | | | | |
| 3 | 3 | | | | NAME SAMPM |
| 4 | 4 | | | | ***** |
| 5 | 5 | | | | ; |
| 6 | 6 | | | | ; |
| 7 | 7 | | | | HEX -> ASCII Conversion Program |
| 8 | 8 | | | | ; |
| 9 | 9 | | | | main-routine |
| 10 | 10 | | | | ***** |
| 11 | 11 | | | | |
| 12 | 12 | | | | PUBLIC MAIN, START |
| 13 | 13 | | | | EXTRN CONVAH |
| 14 | 14 | | | | |
| 15 | 15 | ---- | | | DATA DSEG AT 0FE20H |
| 16 | 16 | FE20 | | | HDTSA: DS 1 |
| 17 | 17 | FE21 | | | STASC: DS 2 |
| 18 | 18 | | | | |
| 19 | 19 | ---- | | | CODE CSEG AT 0H |
| 20 | 20 | 0000 | R0000 | | MAIN: DW START |
| 21 | 21 | | | | |
| 22 | 22 | ---- | | | CSEG |
| 23 | 23 | 0000 | 2B4100 | | START: MOV RFM, #00 |
| 24 | 24 | 0003 | 0BFC80FE | | MOVW SP, #0FE80H |
| 25 | 25 | 0007 | 2B4000 | | MOV MM, #00 |
| 26 | 26 | 000A | 0944F708 | | MOV STBC, #08H |
| 27 | 27 | | | | |
| 28 | 28 | 000E | 3A201A | | MOV HDTSA, #1AH |
| 29 | 29 | 0011 | 6720FE | | MOVW HL, #HDTSA ;set hex |
| 30 | 30 | | | | 2-code data in HL register |

(b) Option for page length specification (-LL)

Description format: -LL [No. of lines per page]

Default assumption: -LL66 (No form-feed operation
with output to console)

Function

- o The -LL option specifies the number of lines per page of a list file.

Use

Use the -LL option if you want to change the number of lines per page of any list file.

Explanation

- o The number of print lines per page to be specified with the -LL option must be within the following value range:
 $20 \leq \text{No. of print lines per page} \leq 32767$
If any value beyond this range or other than a value is specified with this option, an abort error will result.
- o If the number of print lines per page is omitted, a value of 66 is assumed to have been specified.
- o If "0" is specified as the number of print lines per page, no form-feed operation (page ejection) will be carried out.
- o If the -LL option is specified at the same time with the -NP option, the -LL option will be ignored and thus will become invalid.

Application Examples

Example 1: To assemble the source program with -LL option
specified to set 20 lines as No. of lines per page
of an assembly list file

A>ra78k3 -c310 78k3main.asm -ll20 -lw80

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

- o When assembly list file "78K3MAIN.PRN" is referenced, the output assembly list will look like this.

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c310 78k3main.asm -l120 -lw80

Para-file:

In-file: 78K3MAIN.ASM

Obj-file: 78K3MAIN.REL

Prn-file: 78K3MAIN.PRN

Assemble list

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 2

| ALNO | STNO | ADRS | OBJECT | M | I | SOURCE | STATEMENT |
|------|------|------|--------|---|---|--------|-----------------------------------|
| 1 | 1 | | | | | \$ | PROCESSOR(310) |
| 2 | 2 | | | | | | |
| 3 | 3 | | | | | NAME | SAMPM |
| 4 | 4 | | | | | ***** | ***** |
| 5 | 5 | | | | | ; | * |
| 6 | 6 | | | | | ; | HEX -> ASCII Conversion Program * |
| 7 | 7 | | | | | ; | * |
| 8 | 8 | | | | | ; | main-routine * |

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 3

| ALNO | STNO | ADRS | OBJECT | M | I | SOURCE | STATEMENT |
|------|------|------|--------|---|---|-----------|----------------|
| 9 | 9 | | | | | ; | * |
| 10 | 10 | | | | | ***** | ***** |
| 11 | 11 | | | | | | |
| 12 | 12 | | | | | PUBLIC | MAIN, START |
| 13 | 13 | | | | | EXTRN | CONVAH |
| 14 | 14 | | | | | | |
| 15 | 15 | ---- | | | | DATA | DSEG AT 0FE20H |
| 16 | 16 | FE20 | | | | HDTSA: DS | 1 |
| 17 | 17 | FE21 | | | | STASC: DS | 2 |

(c) Option for title character string specification (-LH)

| |
|------------------------------------------|
| Description format: -LH character-string |
| Default assumption: None |

Function

- o The -LH option specifies the character string to be printed in the TITLE column in the header of an assembly list file.

Use

Use the -LH option if you want to give a title to each page of an assembly list file so that the contents of the file can be identified at a glance.

Explanation

- o The number of characters that can be specified as a title is up to 60 characters. However, no blank may be described in the title character string.
- o If more than 60 characters are described as the title character string, the assembler will accept only the first 60 characters of the string as valid and will not output an error message.

<With 78K/0, 78K/III>

However, if the number of characters per line is 117 or less, the effective character string length as a title becomes as follows:

Effective length=(maximum No. of characters per line)-58

<With 78K/I, 78K/VI>

However, if the number of characters per line is 119 or less, the effective character string length as a title becomes as follows:

Effective length=(maximum No. of characters per line)-60

- o If no character string is specified with the -LH option, an abort error will result.

- o If the -LH option is specified at the same time with the -NP option, the -LH option will be ignored and thus will become invalid.
- o If the -LH option is omitted, the TITLE column of the output assembly list file will become blank.
- o The character set that can be described as a title is as shown in Table 4-5 below.

Table 4-5. Characters That Can Be Described as Title

| Character | On Command line | In Parameter File |
|---------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| * ? > < | May be used by enclosing each character in " " (double quotes). See Note below. | May be used. Each character enclosed in " " is interpreted the same as when used on the command line. |
| ; | May be used by enclosing this character in " " (double quotes). | Cannot be used. (Interpreted as comment.) |
| # | May be used. | Cannot be used. (Interpreted as comment.) |
| " (double quote) | Cannot be used as an effective character for a title. | Cannot be used as an effective character for a title. |
| 00H | Cannot be used. | May be used (but the character string is interpreted to have been terminated) |
| 03H,06H,08H 0DH,0EH,10H 15H,17H,18H 1BH,7FH | Cannot be used. | May be used, but each of these characters is output as "!" in an assembly list file. (ODH alone will not be output to the list.) |
| 01H,02H,04H, 05H,07H,08H, 0CH,0FH,11H, 12H,13H,14H, 16H,19H,1CH, 1DH,1EH,1FH | May be used, but each of these characters is output as "!" in an assembly list file. | May be used, but each of these characters is output as "!" in an assembly list file. |
| 1AH | May be used, but this character is displayed as "!" in an assembly list file. | Cannot be used. (End-Of-File code) |
| Alphabetic characters | Each uppercase or lowercase letter is input as is. | Each uppercase or lowercase letter is input as is. |
| Others | May be used. | May be used. |

Note: Character "*" on the start-up command may be described without enclosing it in a pair of double quotes if the character is not subject to expansion as a wild card.

Note

A control instruction which has the same function as the -LH option can be described in the header of the input source module file. The description formats of this control instruction are as shown below.

```
▲$▲TITLE ▲( ▲'character-string'▲ )
▲$▲TT ▲( ▲'character-string'▲ ) ; abbreviated format
```

See Chapter 4 in the "RA78K Series Assembler Package User's Manual for Language" for details of this control instruction.

Application Examples

Example 1: To assemble the source program with -LH option specified to print a title in the header of assembly list file

```
A>ra78k3 -c310 78k3main.asm -lhRA78K3 MAINROUTINE
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

- o When assembly list file "78K3MAIN.PRN" is referenced, the output assembly list will look like this.

uCOM-78K/III Assembler Vx.xx RA78K3_MAINROUTINE Date:xx xxx xxxx Page: 1

Title

Command: -c310 78k3main.asm -lw80 -lhRA78K3_MAINROUTINE

Para-file:

In-file: 78K3MAIN.ASM

Obj-file: 78K3MAIN.REL

Prn-file: 78K3MAIN.PRN

Assemble list

| ALNO | STNO | ADRS | OBJECT | M | I | SOURCE STATEMENT |
|------|------|------|--------|---|---|---------------------------------|
| 1 | 1 | | | | | \$ PROCESSOR(310) |
| 2 | 2 | | | | | |
| 3 | 3 | | | | | NAME SAMP |
| 4 | 4 | | | | | :***** |
| 5 | 5 | | | | | ; |
| 6 | 6 | | | | | HEX -> ASCII Conversion Program |
| 7 | 7 | | | | | ; |
| 8 | 8 | | | | | main-routine |
| 9 | 9 | | | | | ; |
| 10 | 10 | | | | | :***** |
| 11 | 11 | | | | | |
| 12 | 12 | | | | | PUBLIC MAIN, START |
| 13 | 13 | | | | | EXTRN CONVAH |
| 14 | 14 | | | | | |
| 15 | 15 | ---- | | | | DATA DSEG AT 0FE20H |
| 16 | 16 | FE20 | | | | HDTSA: DS 1 |
| 17 | 17 | FE21 | | | | STASC: DS 2 |

(d) Option for title character string specification (-LT)

| |
|----------------------------------------|
| Description format: -LT No. of columns |
|----------------------------------------|

| |
|--------------------------|
| Default assumption: -LT8 |
|--------------------------|

Function

The -LT option specifies the number of columns which becomes the basis of tabulation processing to output an assembly list by replacing a HT (Horizontal Tabulation) code in a source module with several blank characters on any list.

Use

If the number of columns per line of a list is lessened by specifying the -LW option, use the -LT option to lessen the number of blanks by a HT code, thereby saving the number of columns.

Explanation

- o The number of columns to be specified with the -LT option must be within the following value range:

$$0 \leq \text{No. of columns} \leq 8$$

If any value beyond the above range or other than a value is input with this option, an abort error will result.

- o If -LT0 is specified, tabulation processing will not be performed. In this case, a HT code is replaced with one blank character for output.
- o If the -LT option is specified at the same time with the -NP option, the -LT option will become invalid.

Application Examples

Example 1: When assembly list file "78K3MAIN.PRN" output from the source program assembled with -LT option omitted is referenced, the output assembly list will look like this.

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c310 78k3main.asm -lw80
Para-file:
In-file: 78K3MAIN.ASM
Obj-file: 78K3MAIN.REL
Prn-file: 78K3MAIN.PRN

Assemble list

| ALNO | STNO | ADRS | OBJECT | M | I | SOURCE STATEMENT |
|------|------|------|----------|---|---|---------------------------------|
| 1 | 1 | | | | | \$ PROCESSOR(310) |
| 2 | 2 | | | | | |
| 3 | 3 | | | | | NAME SAMPM |
| 4 | 4 | | | | | ***** |
| 5 | 5 | | | | | ; |
| 6 | 6 | | | | | HEX -> ASCII Conversion Program |
| 7 | 7 | | | | | ; |
| 8 | 8 | | | | | main-routine |
| 9 | 9 | | | | | ; |
| 10 | 10 | | | | | ***** |
| 11 | 11 | | | | | |
| 12 | 12 | | | | | PUBLIC MAIN, START |
| 13 | 13 | | | | | EXTRN CONVAH |
| 14 | 14 | | | | | |
| 15 | 15 | ---- | | | | DATA DSEG AT 0FE20H |
| 16 | 16 | FE20 | | | | HDTSA: DS 1 |
| 17 | 17 | FE21 | | | | STASC: DS 2 |
| 18 | 18 | | | | | |
| 19 | 19 | ---- | | | | CODE CSEG AT 0H |
| 20 | 20 | 0000 | R0000 | | | MAIN: DW START |
| 21 | 21 | | | | | |
| 22 | 22 | ---- | | | | CSEG |
| 23 | 23 | 0000 | 2B4100 | | | START: MOV RFM, #00 |
| 24 | 24 | 0003 | 0BFC80FE | | | MOVW SP, #0FE80H |
| 25 | 25 | 0007 | 2B4000 | | | MOV MM, #00 |
| 26 | 26 | 000A | 0944F708 | | | MOV STBC, #08H |
| 27 | 27 | | | | | |
| 28 | 28 | 000E | 3A201A | | | MOV HDTSA, #1AH |
| 29 | 29 | 0011 | 6720FE | | | MOVW HL, #HDTSA |

...

Example 2: To assemble the source program with -LT1 option
specified (number of blanks by HT code=1)

A>ra78k3 -c310 78k3main.asm -lt1

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start
Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

- o When assembly list file "78K3MAIN.PRN" is referenced,
the output assembly list will look like this.

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c310 78k3main.asm -lw80 -ltl
Para-file:
In-file: 78K3MAIN.ASM
Obj-file: 78K3MAIN.REL
Prn-file: 78K3MAIN.PRN

Assemble list

| ALNO | STNO | ADRS | OBJECT | M I | SOURCE STATEMENT |
|------|------|------|----------|-----|--------------------------------------|
| 1 | 1 | | | | \$ PROCESSOR(310) |
| 2 | 2 | | | | |
| 3 | 3 | | | | NAME SAMPM |
| 4 | 4 | | | | ***** |
| 5 | 5 | | | | ; |
| 6 | 6 | | | | ;* HEX -> ASCII Conversion Program * |
| 7 | 7 | | | | ; |
| 8 | 8 | | | | ;* main-routine * |
| 9 | 9 | | | | ; |
| 10 | 10 | | | | ***** |
| 11 | 11 | | | | |
| 12 | 12 | | | | PUBLIC MAIN, START |
| 13 | 13 | | | | EXTRN CONVAH |
| 14 | 14 | | | | |
| 15 | 15 | ---- | | | DATA DSEG AT 0FE20H |
| 16 | 16 | FE20 | | | HDTSA: DS 1 |
| 17 | 17 | FE21 | | | STASC: DS 2 |
| 18 | 18 | | | | |
| 19 | 19 | ---- | | | CODE CSEG AT 0H |
| 20 | 20 | 0000 | R0000 | | MAIN: DW START |
| 21 | 21 | | | | |
| 22 | 22 | ---- | | | CSEG |
| 23 | 23 | 0000 | 2B4100 | | START: MOV RFM, #00 |
| 24 | 24 | 0003 | 0BFC80FE | | MOVW SP, #0FE80H |
| 25 | 25 | 0007 | 2B4000 | | MOV MM, #00 |
| 26 | 26 | 000A | 0944F708 | | MOV STBC, #08H |

The number of blanks by HT code is 1.

- (e) Options for form-feed code addition specification
(-LF/-NLF)

| | |
|---------------------|------|
| Description format: | -LF |
| | or |
| | -NLF |
| Default assumption: | -NLF |

Function

- o The -LF option specifies the addition of form-feed (FF) code to the end of an assembly list file.
- o The -NLF option specifies the non-addition of form-feed (FF) code to the end of an assembly list file.

Use

If you want to have a new page after printing the contents of an assembly list file, add a form-feed (FF) code to the end of the assembly list file by specifying the -LF option.

Explanation

- o If the -LF option is specified at the same time with the -NP option, the -LF option will become invalid.
- o If the -LF and -NLF options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To assemble the source program with -LF option
specified to add FF code to the end of an assembly
list file

A>ra78k3 -c310 78k3main.asm -pprn -lf

uCOM-78K/III Assembler Vx.xx *[xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start

Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

(11) Options for error list file output specification (-E/-NE)

Description format: -E [output-filename]

or

-NE

Default assumption: -NE

Function

- o The -E option specifies the output of an error list file. It also specifies the output destination or output filename of an error list file to be output by the assembler.
- o The -NE option specifies the non-output of an error list file.

Use

- o Use the -E option if you want to save error messages to a file.
- o Also use the -E option if you want to change the output destination or output filename of an error list file.

Explanation

- o An output filename can be specified with either a disk type filename or a device type filename. However, if a device type filename "CLOCK" is specified as an output filename, an abort error will result.
- o If an output filename is omitted from the -E option specification, "input filename.ERA" is assumed as the output error list filename.
- o If a drive name is omitted from the -E option specification, the error list file will be output to the current drive.
- o If the -E and -NE options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To assemble the source program with -E option specified to create an error list file named "SAMPLE.ERA"

```
A>ra78k3 -c310 78k3main.asm -esample.era
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start
```

```
78K3MAIN.ASM(20) : E101 Syntax error
```

```
78K3MAIN.ASM(23) : E101 Syntax error
```

```
Pass2 Start
```

```
78K3MAIN.ASM(12) : E137 Public symbol is undefined
```

```
78K3MAIN.ASM(20) : E101 Syntax error
```

```
78K3MAIN.ASM(23) : E101 Syntax error
```

```
Assembly complete,      3 error(s) and      0 warning(s) found.
```

o When error list file "SAMPLE.ERA" is referenced, the output error list file will look like this.

```
Pass1 Start
```

```
78K3MAIN.ASM(20) : E101 Syntax error
```

```
78K3MAIN.ASM(23) : E101 Syntax error
```

```
Pass2 Start
```

```
78K3MAIN.ASM(12) : E137 Public symbol is undefined
```

```
78K3MAIN.ASM(20) : E101 Syntax error
```

```
78K3MAIN.ASM(23) : E101 Syntax error
```

(12) Option for parameter file specification (-F)

Description format: -F filename

Default assumption: Options and input filenames can be input only from start-up command line.


Function

The -F option tells the assembler that options and input filename(s) will be input from the file specified by this option.

Use

- o Use the -F option if all the required parameters for starting up the assembler cannot be specified in the start-up command line.
- o If you have a set of assembler options which you must specify repeatedly at each assembly operation, describe these assembler options in a parameter file and then specify the -F option in the start-up command line.

Explanation

- o A filename can be specified with only a disk type filename. If any device type filename is specified with this option, an abort error will result.
- o If a filename is omitted from the -F option specification, an abort error will also result.
- o Nesting of parameter files is not allowed. If the -F option is specified in a parameter file, an abort error will result.
- o The number of characters that can be described in a parameter file is not limited.
- o A blank character, Tab character, or "  " must be used as a delimiter between options or input filenames.

- o The options and input filenames described in the parameter file will be expanded to the location on the command line where the parameter file (-F option) has been specified.
- o The assembler will process these expanded options in the order from the last input option.
- o All characters described between ";" or "#" and "□" or EOF code will be interpreted as a comment statement.
- o If two or more -F options are specified at the same time, an abort error will result.

Application Examples

Example 1: To assemble the source program with -F option specified

- o Contents of parameter file "78K3MAIN.PRA"

```
;parameter file
78k3main.asm -osample.rel -g
-psample.prn
```

- o Enter -F option in the start-up command line as follows:

```
A>ra78k3 -f78k3main.pra
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

-T

Temporary file creation
path specification

(13) Option for temporary file creation path specification (-T)

Description format: -T pathname

Default assumption: Temporary file is created on the path
specified by environment variable TMP
or on the current path if no path is
specified by TMP

Function

The -T option tells the assembler to create a temporary file on the path specified by this option.

Use

The -T option can be used to specify where a temporary file is to be created.

Explanation

- o Other than a path cannot be specified as a pathname. If a pathname is omitted from the -T option specification, an abort error will result.
- o If a previously created temporary file exists, the assembler will create a temporary file by overwriting the file unless it is write-protected.
- o If the required memory space for temporary file creation is available, the assembler will create a temporary file in memory. If the memory space is exhausted during the temporary file creation, the assembler will save the temporary file contents in memory to another disk and subsequent accessing to the temporary file will be made to that disk.
- o The temporary file created for an assembly process by the assembler will be erased on completion of the assembly process. The temporary file will also be erased when the assembly process is discontinued by CTRL-C key input.

-T

Temporary file creation
path specification

o A path for temporary file creation is determined in the following order:

- ① Path specified by the -T option
- ② Path specified by environment variable TMP (when the -T option is omitted)
- ③ Current path (when no path is specified by environment variable TMP)

If a temporary file cannot be created on the path specified by ① or ②, an abort error will result.

Application Examples

Example 1: To assemble the source program with -T option specified to create a temporary file on directory TMP

```
A>ra78k3 -c310 78k3main.asm -ttmp
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start  
Pass2 Start
```

```
Assembly complete,      0 error(s) and      0 warning(s) found.
```

(14) Option for HELP message display specification (--)

Description format: --

Default assumption: No HELP message is displayed.

Function

The -- option tells the assembler to display the HELP message on the console.

Use

The HELP message is a list of all assembler options and their functional descriptions. Use the -- option if you want to refer to this message when executing the assembler.

Explanation

- o If the -- option is specified, all the other assembler options specified at the same time will become invalid.
- o If you want to see the next screen for the continuation of the HELP message, type the RETURN (CR) key. If you want to terminate the HELP message display, first type any key other than the RETURN key and then type the RETURN key.

Application Examples

Example 1: Input -- option as shown below and the HELP message will be displayed on the screen.

A>ra78k3 --

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

usage : RA78K3 [option[...]] input-file [option[...]]

The option is as follows ([] means omissible).

-Cx : Select target chip. (x = 310, 312A, etc.) *Must be specified.
-O[file]/-NO : Create the object module file [with the specified name] / Not.
-E[file]/-NE : Create the error list file [with the specified name] / Not.
-P[file]/-NP : Create the print file [with the specified name] / Not.
-KA/-NKA : Output the assemble list to print file / Not.
-KS/-NKS : Output the symbol table list to print file / Not.
-KX/-NKK : Output the cross reference list to print file / Not.
-LW[width] : Specify print file columns per line.
-LL[length] : Specify print file lines per page.
-LF/-NLF : Add Form Feed at end of print file / Not.
-LTn : Expand TAB character for print file(n=1 to 8)/ Not expand(n=0).
-LHstring : Print list header with the specified string.

CHAPTER 5. LINKER

The Linker accepts object module files which are output by the assembler for the 78K series as input files and outputs a load module file.

The linker also outputs list files such as a link list file and an error list file.

If any link error occurs, the linker outputs an error message to an error list file to clearly indicate the cause of the error.

In this case, no load module file will be output by the linker.

5.1 Input/Output Files of Linker

The files listed in Table 5-1 below are input and output to and from the linker.

Table 5-1. I/O Files of Linker

| Type | Name and description of file | Default file type |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| Input files | <u>Object module file</u> A binary file which contains machine language information, relocation information on the address of each machine-coded instruction, and symbol information. This file is output by the Assembler. | .REL |
| | <u>Library file</u> A file in which two or more object module files are registered. This file is output by the Librarian. | .LIB |
| | <u>Directive file</u> A file containing linking instructions to the linker. This file must be created by the user. | .DR |
| | <u>Parameter file</u> ... A file containing the parameters of the executable program (Linker). This file must be created by the user. | .PLK |

Table 5-1. I/O Files of Linker (contd)

| Type | Name and description of file | Default file type |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| Output files | <u>Load module file</u> A binary image file containing all information on the result of the linking process. This file becomes an input file to the Object Converter | .LNK |
| | <u>Link list file</u> A list file containing linkage information such as map list, directive file, and Public symbol list. | .MAP |
| | <u>Error list file</u> A file containing information on errors at linkage time. | .ELK |
| I/O file | <u>Temporary file</u> A file which is automatically generated by the linker as a work file. This file will be erased on completion of the linking process. | LK78Kn.\$\$1, LK78Kn.\$\$2, or LK78Kn.\$\$3 (fixed to this name) |

(n = 0, 1, 2, 3, or 6)

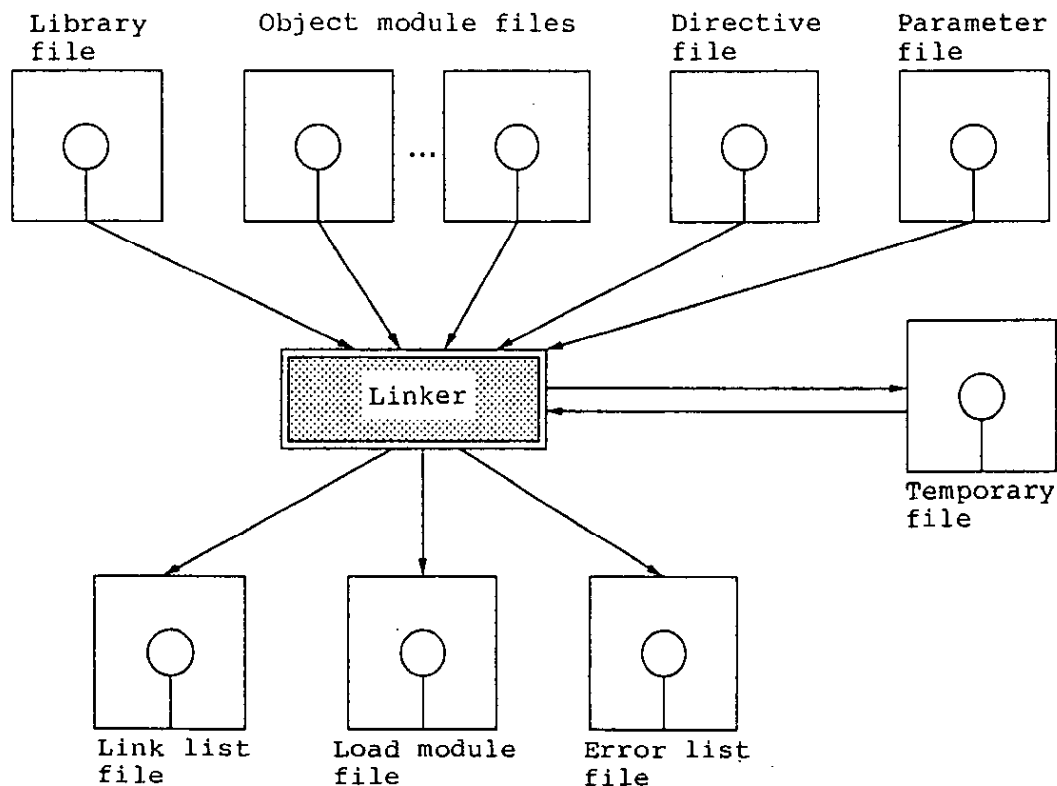


Fig. 5-1. I/O Files of Linker

5.2 Linker Functions

- (1) The linker reads the input object module files and outputs a load module file.
- (2) The linker performs the linking process according to the linker option(s) specified at the start-up of the linker. (See Section 5.8, Linker Options for details of the linker options.)
- (3) The linker merges the input segments and allocates an address to each segment. See Section 5.4, Merging the Input Segments for details of how to merge the input segments and Section 5.5, Determining the Location Addresses of Segments for details of how to allocate an address to each segment.
- (4) After determining the location addresses of the respective segments, the linker determines the value of a label defined in a relocatable segment. Because a label defined in a relocatable segment has an offset from the beginning of a segment defined in an object module file, the linker determines the symbol value of the segment by adding this offset value to the start address of the segment.
- (5) After the symbol value of each segment has been determined, based on the determined symbol value or location address, the linker resolves the relocation of any segment whose value could not be determined at assembly time, because a relocatable symbol, an externally defined symbol, or the location counter in a relocatable segment is referenced.
- (6) On completion of the linking process without any abort or fatal error, the linker outputs a linking end message and returns control to the OS.

5.3 Memory Spaces and Memory Areas

5.3.1 Memory spaces

A memory space refers to a space for defining a memory area. This linker has 16 memory spaces (one regular space and 15 extension spaces), each of which is named as follows:

Regular space = REGULAR

Extension spaces = EX1, EX2, EX3, EX4, EX5, EX6, EX7, EX8,
EX9, EX10, EX11, EX12, EX13, EX14, EX15

(1) Regular space

The regular space is an address space in which ordinary internal ROM and RAM areas and/or "sfr" area exist. Only this regular space is used as a memory space when memory is not to be extended by using the memory bank selection function.

(2) Extension spaces

For the user who wish to expand memory into an area in which an external memory can be located by using the memory bank selection function, the extension spaces are provided to facilitate location of segments to two or more memory banks. By locating segments to any of these extension spaces by memory bank selection in addition to the regular space, object code generation extending over two or more memory spaces may be implemented and symbol references between the regular and extension spaces may be resolved.

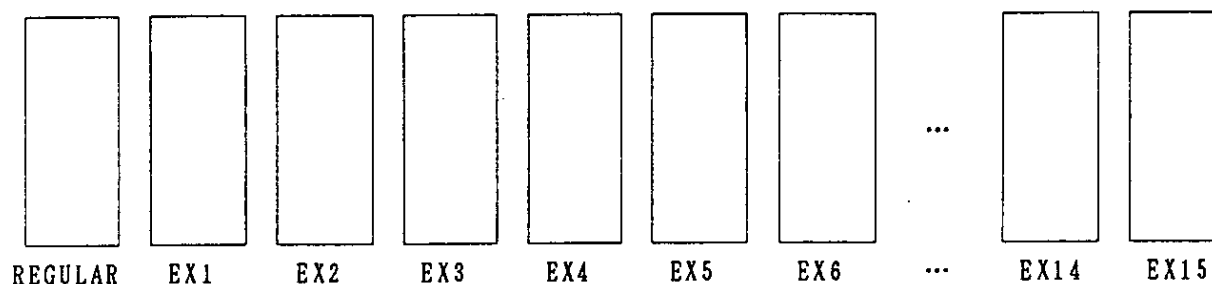


Fig. 5-2. Memory Spaces

5.3.2 Memory areas

A memory area is an area which has been defined in a memory space for locating segments. Memory areas are divided into default memory areas which are to be defined in advance by the linker and memory areas which are to be defined with MEMORY directives by the user.

The linker always locates segments to an address range in which a memory area has been defined, never to an address range in which no memory area has been defined.

(1) Each memory area has a name which has been defined in a memory space (i.e., a memory area name), a start address, and a size. Relations between memory spaces and memory areas are as shown below.

- ① Memory area "ROM" is defined in the REGULAR space with start address 0H and size 8000H.
- ② Memory area "RAM" is defined in the REGULAR space with start address C000H and size 3F00H.
- ③ Memory area "MEM1" is defined in the EX1 space with start address 8000H and size 4000H.
- ④ Memory area "MEM2" is defined in the EX2 space with start address 8000H and size 4000H.

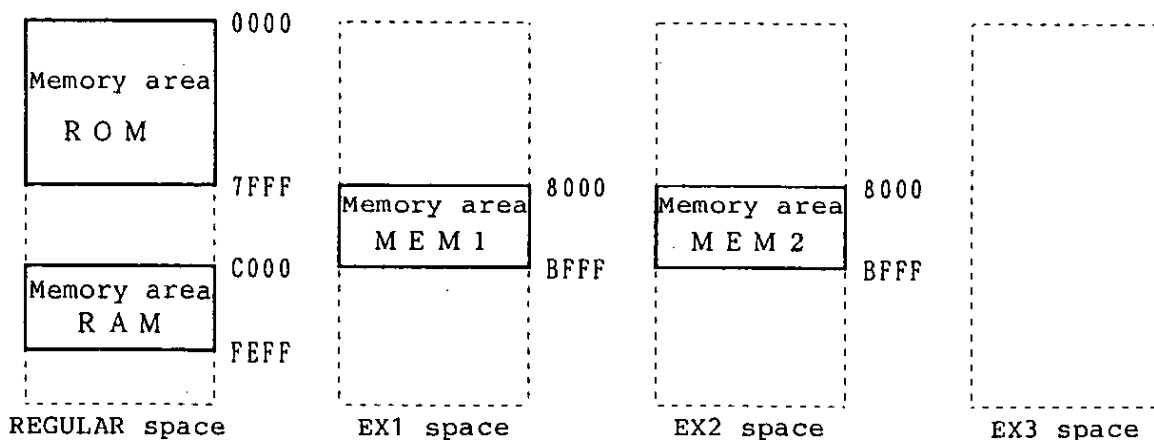


Fig. 5-3. Memory Spaces and Memory Areas

- (2) As mentioned earlier, memory areas are divided into those which are to be defined by the linker by default assumption and those which can be defined freely by the user.

As default memory areas, two memory areas are available: one which has a name "ROM" (in uppercase letters) and the other which has a name "RAM" (in uppercase letters). The start address and size of each of these memory areas differ depending on the target device. With the 78K/0, however, ROM, high-speed RAM (IHRAM), and low-speed RAM (LRAM) areas are available as default memory areas.

The user can use the memory area ROM and memory area RAM as defined by default assumption for locating segments or re-define the start address or size of any of these default memory areas. However, note that the address range that can be re-defined differs depending on the target device. So, be sure to re-define the start address or size of any of these memory areas within the address range established for each target device.

See Section 5.6, Link Directives for the definition and re-definition of a memory area.

- (3) If no directive relating to address allocation to any segment is given during a linking process, the linker determines a memory area to which segments are to be allocated based on the type and relocation attribute of each segment described by the user in the source program. The areas to be allocated to the respective segments by default assumption are shown in Table 5-2.

Table 5-2. Areas to Be Allocated to Segments by Default Assumption

| Applicable target device | | | | Type and relocation attribute of segment | Address to be allocated to each segment by default assumption | Changeability of location address(es) |
|--------------------------|---|-----|----|------------------------------------------|---------------------------------------------------------------------------------------------------------------|---------------------------------------|
| 0 | I | III | VI | | | |
| o | o | o | o | CSEG (UNIT) (see Note 1) | With 78K/0: 80H to FA7FH With 78K/I,78K/III,78K/VI: Any address in ROM area | Yes |
| o | x | x | x | CSEG CALLT | Even address in area from 40H to 7FH | Yes |
| x | o | o | x | CSEG CALLT0 | Even address in area from 40H to 7FH | Yes |
| x | x | o | x | CSEG CALLT1 | Even address in area from 8040H to 807FH | Yes |
| x | x | x | o | CSEG TABLE | Even address in area from 50H to FFH | Yes |
| o | o | o | x | CSEG FIXED | Any address in area from 800H to FFFH | Yes |
| o | o | o | o | CSEG AT (see Note 2) | Specified address | No |
| o | o | o | o | DSEG AT (see Note 2) | Specified address | No |
| o | o | o | o | BSEG AT (see Note 2) | Specified address | No |
| o | o | o | o | DSEG (UNIT) (see Note 1) | With 78K/0: Any address in high-speed RAM area With 78K/I,78K/III,78K/VI: Any address in RAM area | Yes |
| o | x | x | x | DSEG LRAM | Any address in low-speed RAM area | Yes |
| o | o | o | o | DSEG SADDR | With 78K/0,78K/I,78K/III: FE20H to FEFFH (see Note 3) With 78K/VI: FC00H to FEFFH | Yes |
| o | o | o | x | BSEG (UNIT) | FE20H to FEFFH (see Note 3) | Yes |
| x | x | x | o | BSEG SADDR | FC00H to FEFFH | Yes |
| o | x | o | x | DSEG SADDRP | Even address in area from FE20H to FEFFH (see Note 3) | Yes |
| x | x | x | o | DSEG WSADDR | Even address in area from FC00H to FEFFH | Yes |
| x | x | x | o | DSEG DSADDR | Address in multiples of 4 in area from FC00H to FEFFH | Yes |
| o | x | x | x | DSEG IHRAM | Any address in High- speed RAM area | Yes |
| o | x | x | x | DSEG DRAM | Any address in Low- speed RAM area | Yes |

- Note: 1. If no ROM area (or RAM area) exists in the memory area in which a segment whose type and relocation attribute are CSEG (UNIT) or DSEG (UNIT) is to be located, an error will result. When locating a segment in an extension space, a ROM area (or RAM area) can be defined in only one of the memory spaces. Therefore, for a memory space in which no other ROM area (or RAM area) exists, a segment whose type and relocation attribute are CSEG (UNIT) or DSEG (UNIT) will be located in any memory area in the memory space unless a memory area or location address is explicitly specified.
2. Includes the segment defined by the ORG directive.
3. With the uPD78112, an area from FE40H to FEFFH is applicable, in place of an area from FE20H to FEFFH.

5.4 Merging the Input Segments

5.4.1 Merge types of segments

Each segment has a merge type. The merge type of a segment is a type which specifies how the segment must be merged with other segments. The following four merge types are available.

SEQUENT (Sequential), 2-byte ALIGN, 4-byte ALIGN, and COMPLETE

Note: 4-byte ALIGN is the merge type for exclusive use of the 78K/VI series.

5.4.2 Rule for determining the merge type

The merge type of a segment is determined according to the following rule.

The linker determines the merge type of each segment according to the relocation attribute of the segment specified at assembly time. Combinations of relocation attributes and merge types are shown in Table 5-3 below. However, if the merge type of a segment is specified by a link directive at linkage time, the merge type specified by the link directive will take precedence over the merge type given from the relocation attribute of the segment. See Section 5.6, Link Directives for details of each link directive.

Table 5-3. Merge Type by Relocation Attribute

| Applicable target device | | | | Relocation attribute of segment | Merge type to be given |
|--------------------------|---|-----|----|---------------------------------|-------------------------|
| 0 | I | III | VI | | |
| o | o | o | o | CSEG (UNIT) | SEQUENT (Sequential) |
| o | o | o | x | CSEG FIXED | |
| o | o | o | o | DSEG (UNIT) | |
| o | o | o | o | DSEG SADDR | |
| o | x | o | x | DSEG SADDRP | |
| o | x | x | x | DSEG LRAM | |
| o | x | x | x | DSEG IHRAM | |
| o | o | o | o | BSEG (UNIT) | |
| x | x | x | o | BSEG SADDR | |
| o | x | x | x | BSEG LRAM | |
| o | x | x | x | CSEG CALLT | |
| x | o | o | x | CSEG CALLT0 | 2-byte ALIGN |
| x | x | o | x | CSEG CALLT1 | |
| x | x | x | o | CSEG TABLE | |
| o | x | o | x | DSEG SADDRP | |
| x | x | x | o | DSEG WSADDR | |
| x | x | x | o | DSEG DSADDR | 4-byte ALIGN |
| o | o | o | o | CSEG AT (see Note below) | COMPLETE |
| o | o | o | o | DSEG AT (see Note below) | |
| o | o | o | o | BSEG AT | |

Note: Includes the segments defined by the ORG directive.

5.4.3 Merging the segments

The linker performs the merging process of segments according to the following rules.

- (1) Of all the input segments, two or more segments which are identical in all the segment name, segment type (CSEG, DSEG, or BSEG), relocation attribute specified at assembly time, location type, and merge type are merged by the linker into a single segment for output, except when the merge type is COMPLETE.

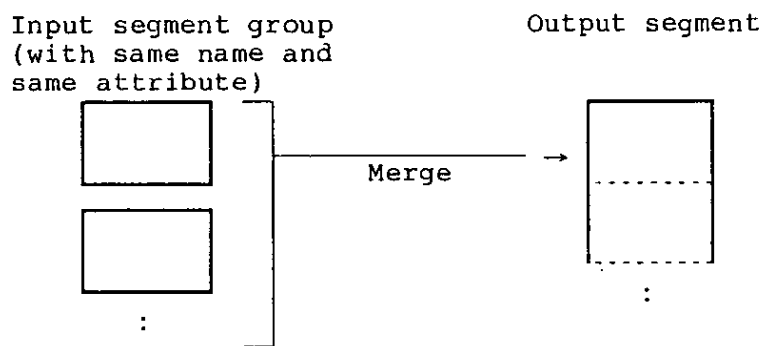


Fig. 5-4. Merging Process (with Same Named Segments)

- (2) If two or more input segments have the same segment name but differ in any of their segment type, relocation attribute, location type, and merge type, an error will result.
- (3) Of all the input segments, a segment which has no other segment identical in segment name will be output as is by the linker. In this case, the segment name, segment type, relocation type, location type, and merge type of the output segment will be the same as those at input time.

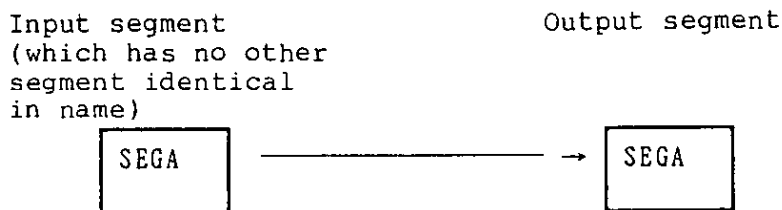


Fig. 5-5. Merging Process (with No Same-Named Segment)

5.4.4 Segment merging method by merge type

The method of merging input segments for each merge type is explained in this subsection.

(1) When merge type is SEQUENT

The method of merging input segments differs between BSEG and any other segment type.

o With segment type other than BSEG

When the merge type is SEQUENT, the input segment data are merged without gap between the two segment data in the order of their appearance in the linker. The value of each origin data indicating the effective part within the segment data will be updated.

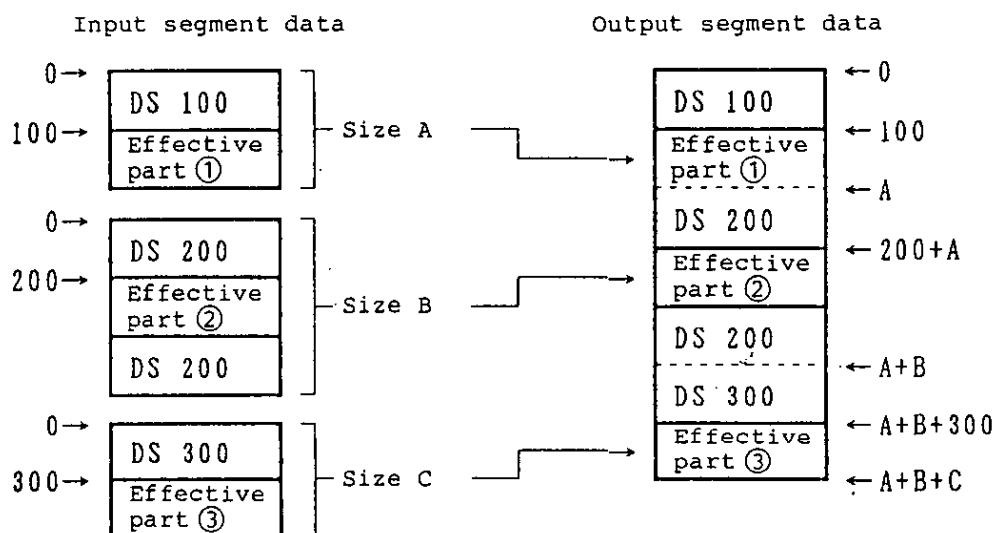


Fig. 5-6. Method of Merging When Merge Type is SEQUENT
(with Segment Type Other Than BSEG)

- o With segment type BSEG

The input segments are merged in a similar manner to the above, except that merging is done in units of bits.

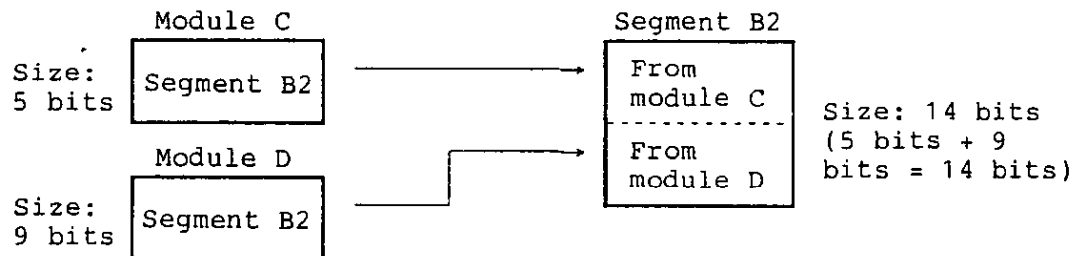


Fig. 5-7. Method of Merging in Bit Units

(2) When merge type is 2-byte ALIGN

When the merge type is 2-byte ALIGN, the input segment data are also merged sequentially in the order of their appearance in the linker, provided that the linker makes alignment so that the start address of each input segment becomes multiples of 2. In other words, if the start address of any input segment becomes an odd-numbered address, the linker will merge the input segments by providing a gap of 1 byte before the start address of the input segment so that the start address becomes multiples of 2.

The value of each origin data indicating the effective part within the segment data will be updated just the same as when the merge type is SEQUENT.

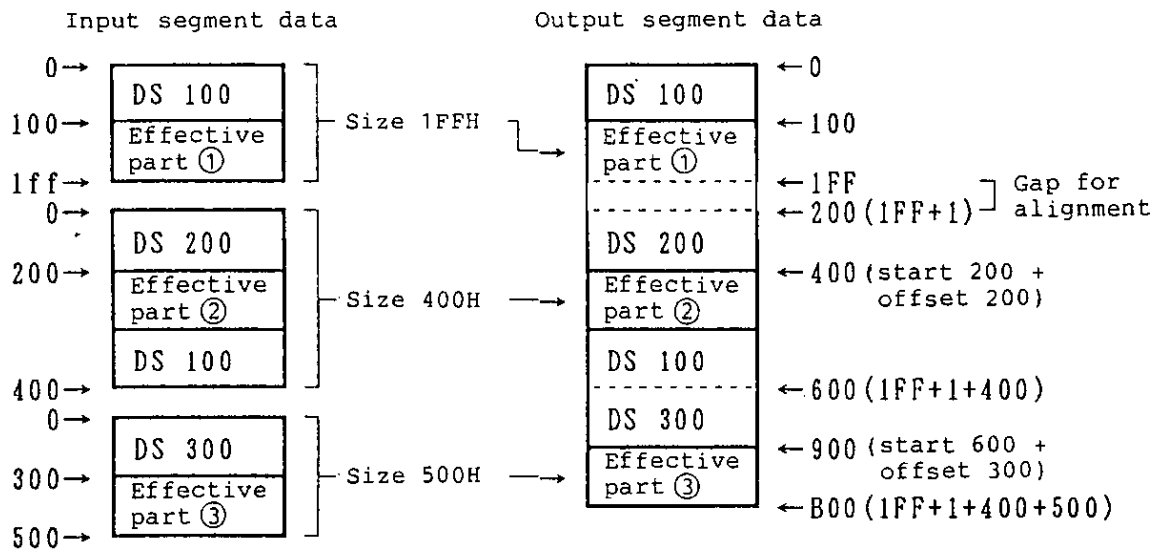


Fig. 5-8. Method of Merging When Merge Type is 2-byte ALIGN

In the above figure, the size of the input segment including the effective part ① is 1FFH. If the input segments are sequentially merged, the start address of the input segment including the effective part ② does not become multiples of 2. Therefore, these two input segments are merged with a gap of 1 byte so that the start address of the input segment including the effective part ② becomes 200H (multiples of 2). As regards the origin data, the values corresponding to the effective data ①, ②, and ③ are updated so as to start from 100H, 400H, and 900H, respectively.

(3) When merge type is 4-byte ALIGN

When the merge type is 4-byte ALIGN, the input segment data are also merged sequentially in the order of their appearance in the linker, provided that the linker makes alignment so that the start address of each input segment becomes multiples of 4. In other words, if the start address of any input segment becomes an odd-numbered address, the linker will merge the input segments by providing a gap of 1 to 3 bytes before the start address of the input segment so that the start address becomes multiples of 4.

The value of each origin data indicating the effective part within the segment data will be updated just the same as when the merge type is SEQUENT.

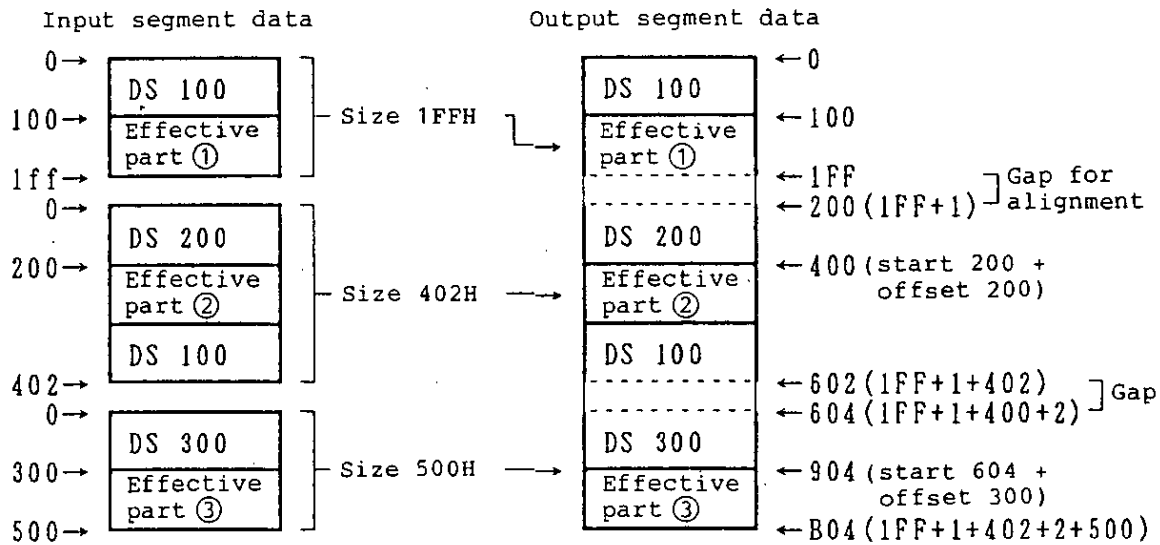


Fig. 5-9. Method of Merging When Merge Type is 4-byte ALIGN

In the above figure, the size of the input segment including the effective part ① is 1FFH. If the input segments are sequentially merged, the start address of the input segment including the effective part ② does not become multiples of 4. Therefore, these two input segments are merged with a gap of 2 bytes so that the imaginary start address of the input segment including the effective part ② becomes 604H (multiples of 4). As regards the origin data, the values corresponding to the effective data ①, ②, and ③ are updated so as to start from 100H, 400H, and 904H, respectively.

Note: 4-byte ALIGN is the merge type applicable only to the 78K/VI series.

(4) When merge type is COMPLETE

If two or more input segments which are identical in the segment name and all have the merge type COMPLETE exist, an error will result. An input segment which has any of the following names will be handled specially by the linker.

?ASEG1, ?ASEG2, ?ASEG3, ?ASEG4, ?ASEG5, ?ASEG6, ?ASEG7,
?ASEG8, ?ASEG9, ?ASEG10, ?ASEG11, ?ASEG12, ?ASEG13, ?ASEG14,
?ASEG15, ?ASEG16, ?ASEG17, ?ASEG18, ?ASEG19, ?ASEG20

- (a) If only one segment which has any of the above segment names is input, the linker will perform the ordinary merging process for the input segment.
- (b) If two or more segments which have one of the above listed names as their identical segment name are input and if the same named segments all have either the merge type SEQUENT or 2-byte ALIGN, the linker will perform the ordinary merging process for the input segments.
- (c) If two or more segments which have one of the above listed names as their identical segment name are input and if the same named segments all have the merge type COMPLETE, the linker will not merge these input segments and locate them as separate segments.
- (d) If two or more segments which have one of the above listed names as their identical segment name are input and if the merge types of the same named segments include both the COMPLETE type and the SEQUENT or 2-byte ALIGN type, the linker will perform the process described in (c) above for the segment(s) which have the merge type COMPLETE and then perform the ordinary merging process for the rest of the segments. (If the remaining segments are identical in all the segment type, relocation attribute, location type, and merge type, these segments will be merged into a single segment. Otherwise, the linker will output an error.)

Note: The above listed 20 segment names will be automatically given by the assembler to segments which have been defined with the ORG directive but whose segment names were omitted from specification at assembly time. Therefore, segments which have these names are normally absolute segments and they all have the merge type COMPLETE. However, because the user may use any of these names explicitly at assembly time and even in such as case the linker performs the merging process for the input segments having any of these names as outlined above, attention must be paid to this point. The assembler also automatically generate such segment names as ?CSEG and ?DSEG. The user is thus prohibited from defining any symbol which begins with "?".

5.5 Determining the Location Addresses of Segments

On completion of the merging process, the linker determines the location address of each segment. This section details how to determine the location address of each segment.

5.5.1 Location types of segments

Table 5-4. List of Location Types of Segments

| Location type | Address information to be added | Meaning |
|--------------------------------------|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BOND | Start address | Assigns the address specified at assembly time for allocation to a segment as the start address of the segment. |
| 2-byte ALIGN | Alignment value | Determines the location addresses of a segment so that the start address of the segment becomes an even-numbered address which can be divided by 2. |
| 4-byte ALIGN (see Note below) | Alignment value | Determines the location addresses of a segment so that the start address of the segment becomes an even-numbered address which can be divided by 4. |
| AREA | Start address End address | Determines the location addresses of a segment so that the entire segment can be located between the start and end addresses (with or without including both). |
| FREE | None | Assign a segment to any address on memory. |

Note: 4-byte ALIGN is the location type applicable only to the 78K/VI series.

Of the above five location types, AREA and 2-byte or 4-byte ALIGN may be specified at the same time. Each location type is detailed below.

(1) BOND

This type locates a segment by using the address specified within a usable memory and determined by the start address specification as the start address of the segment. In this case, the entire segment must be within the range of the usable memory area. Even when the start address of a segment is within a usable memory area, an error will result unless the tail of the segment is within the same continuous memory area.

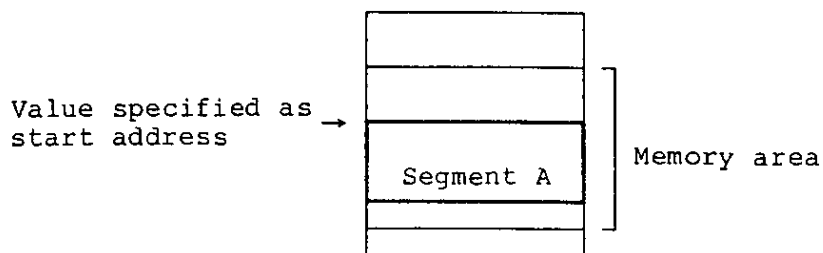


Fig. 5-10. Locating a Segment by BOND Specification

(2) 2-byte ALIGN

This type locates a segment so that the start address of the segment becomes an even-numbered address which can be divided by 2. The entire segment must be located within the address range of a usable memory area (namely, it must not extend over two different memory areas).

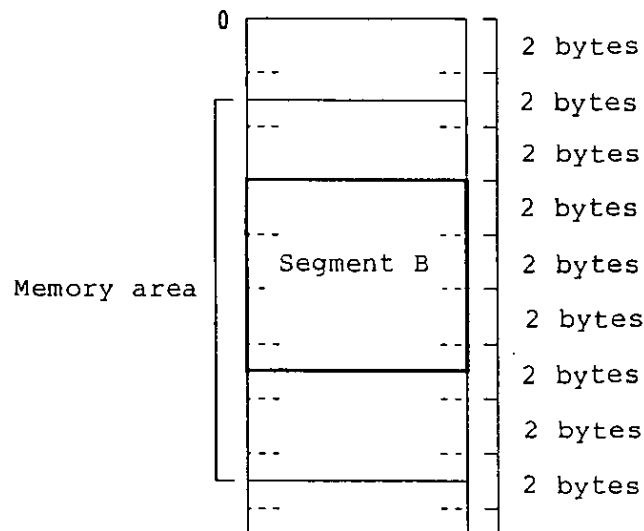


Fig. 5-11. Locating a Segment by 2-byte ALIGN Specification

(3) 4-byte ALIGN

This type locates a segment so that the start address of the segment becomes an even-numbered address which can be divided by 4. The entire segment must be located within the address range of a usable memory area (namely, it must not extend over two different memory areas).

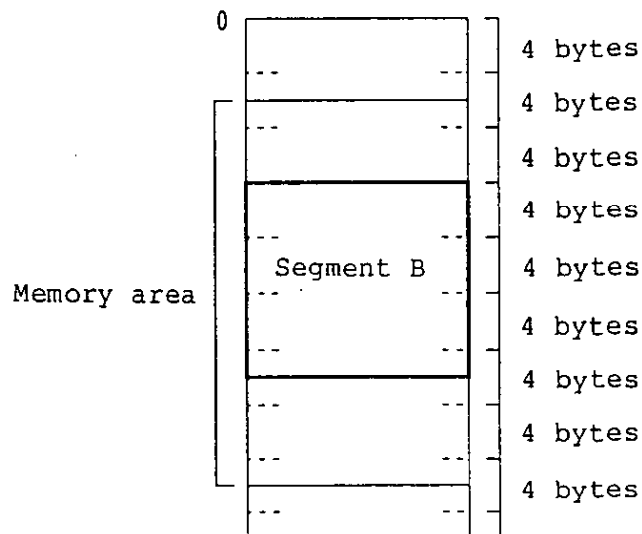


Fig. 5-12. Locating a Segment by 4-byte ALIGN Specification

(4) AREA

This type assigns a segment to anywhere in a usable memory area which is specified by start and end addresses and determines the start address of the segment. In this case, the entire segment must be located within the address range of the usable memory area.

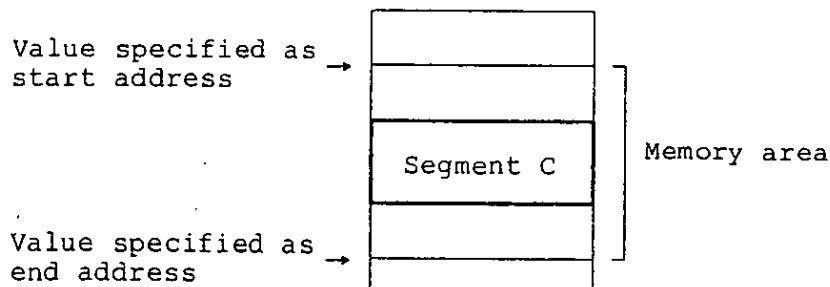


Fig. 5-13. Locating a Segment by AREA Specification

(5) Combination of AREA and 2-byte/4-byte ALIGN

This combination type locates a segment in an area between the start address of AREA and the end address of AREA according to the condition specified by an alignment value. In this case, alignment by the linker begins with address 0, not with the start address of AREA.

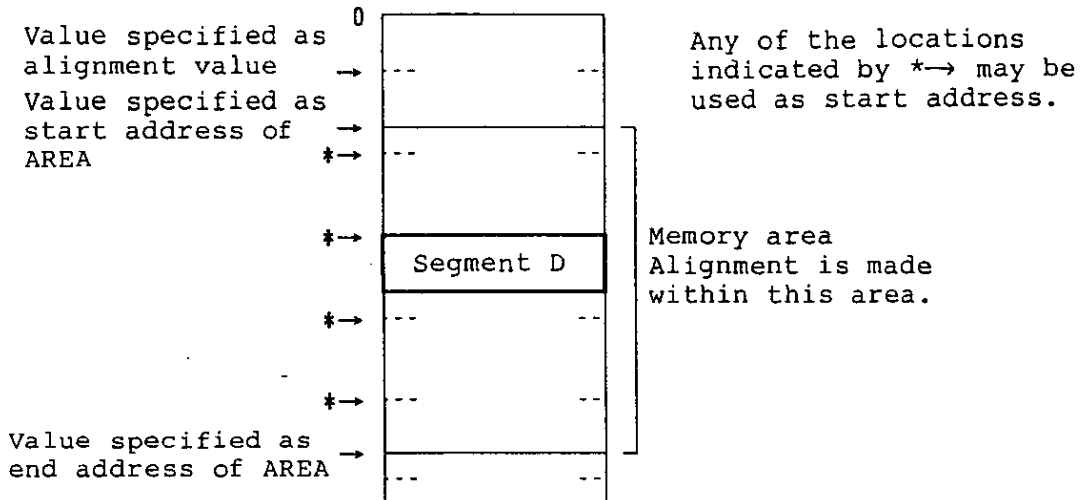


Fig. 5-14. Locating a Segment by Combined AREA/ALIGN Specification

(6) FREE

This type assigns a segment to anywhere in any usable memory area and determines the start address of the segment. In this case, the entire segment must be located within the address range of the usable memory area.

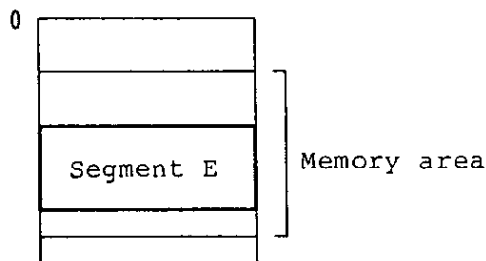


Fig. 5-15. Locating a Segment by FREE Specification

Each location type has a priority level for use when determining the location addresses of a segment. Location addresses are thus determined in the order of the location type with the highest priority.

The precedence order of location types is as shown below.

Highest \leftarrow BOND $>$ AREA + 4-byte ALIGN $>$ AREA + 2-byte ALIGN $>$
AREA $>$ 4-byte ALIGN $>$ 2-byte ALIGN $>$ FREE $>$ \rightarrow Lowest

5.5.2 Rules for determining the location type

The location type of a segment is determined according to the following rules:

- (1) If no location type is specified for a segment by a link directive at linkage time, the location type of the segment will be determined by the relocation attribute of the segment specified at assembly time. The location type which will be given to each segment in this case is shown in Table 5-5. However, if a location type is specified for a segment by a link directive at linkage time, the linker will give priority to the location type specified the link directive. See Section 5.6 for the link directives.

Table 5-5. Location Type Given by Relocation Attribute

| Applicable target device | | | | Relocation attribute of segment | Location type to be given |
|--------------------------|---|-----|----|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | I | III | VI | | |
| o | o | o | o | CSEG (UNIT) | FREE |
| o | o | o | o | DSEG (UNIT) | |
| o | x | x | x | DSEG IHRAM | |
| o | x | x | x | CSEG CALLT | Combination of AREA (start value:40H; end value:7FH) and ALIGN (alignment value: 2 bytes) |
| x | o | o | x | CSEG CALLT0 | |
| x | x | o | x | CSEG CALLT1 | Combination of AREA (start value:8040H; end value:807FH) and ALIGN (alignment value: 2 bytes) |
| o | o | o | x | CSEG FIXED | AREA (start value:800H; end value:FFFH) |
| o | o | o | o | DSEG SADDR | With 78K/0,78K/I,78K/III: AREA (start value:FE20H; end value:FEFFH) (See Note 2.) With 78K/VI: AREA (start value:FC00H; end value:FEFFH) |
| o | o | o | o | BSEG (UNIT) | With 78K/0,78K/I,78K/III: AREA (start value:FE20H; end value:FEFFH) (See Note 2.) With 78K/VI: FREE |
| o | x | o | x | DSEG SADDRP | Combination of AREA (start value:FE20H; end value:FEFFH) and ALIGN (alignment value: 2 bytes) |
| o | x | x | x | DSEG LRAM | AREA (start value: FAE0H; end value: FAFFH) |
| o | x | x | x | BSEG LRAM | |
| x | x | x | o | CSEG TABLE | AREA (start value: 50H; end value: FFH) |
| x | x | x | o | BSEG SADDR | AREA (start value: FC00H; end value: FEFFH) |
| x | x | x | o | DSEG WSADDR | Combination of AREA (start value:FC00H; end value:FEFFH) and ALIGN (alignment value: 2 bytes) |
| x | x | x | o | DSEG DSADDR | Combination of AREA (start value:FC00H; end value:FEFFH) and ALIGN (alignment value: 4 bytes) |
| o | o | o | o | CSEG AT (see Note 1) | BOND (value specified at assembly time) |
| o | o | o | o | DSEG AT (see Note 1) | |
| o | o | o | o | BSEG AT | |

Note: 1. Includes the segment defined by the ORG directive.

2. With the uPD78112, the start value is FE40H.

5.5.3 Rules for determining the segment location addresses

The location addresses of each segment are determined according to the following rules:

- (1) Each segment is located so that the segment in its entirety is within a memory area.

If either of the following two address allocations is made:

- o Allocation which would cause part or all of the segment not to be located within a memory area
- o Allocation which would cause the segment to be located in a noncontinuous address area (by being extended over two or more usable memory areas)

the linker will output an error message and immediately terminate its processing. However, if the -J option has been specified in case of the address allocation which would cause part or all of the segment not to be located within a memory area, the linker will output an error message but will force the segment to be located as specified and continue its processing.

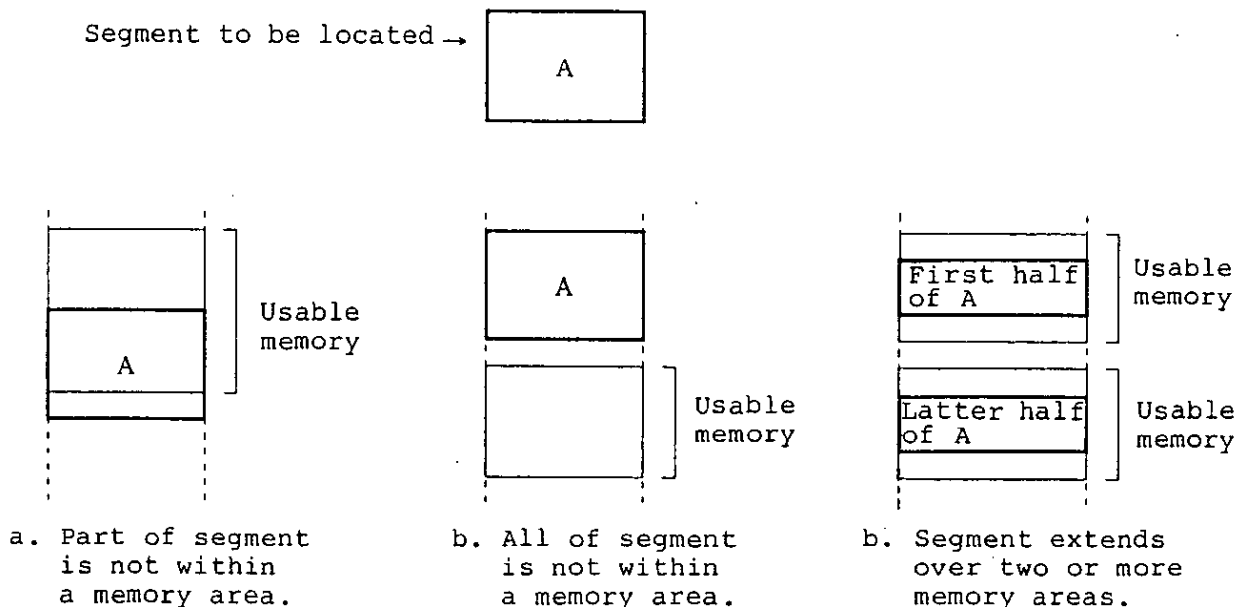


Fig. 5-16. Examples of Improper Segment Location

- (2) If a memory area name is specified for a segment as the Bind specification by a link directive, the segment will be located in the specified memory area.

Likewise, if a memory space name is specified, the segment will be located in a memory area defined in the specified memory space. (Even if a memory space name is specified, no segment will be located in any place not defined as a memory area.)

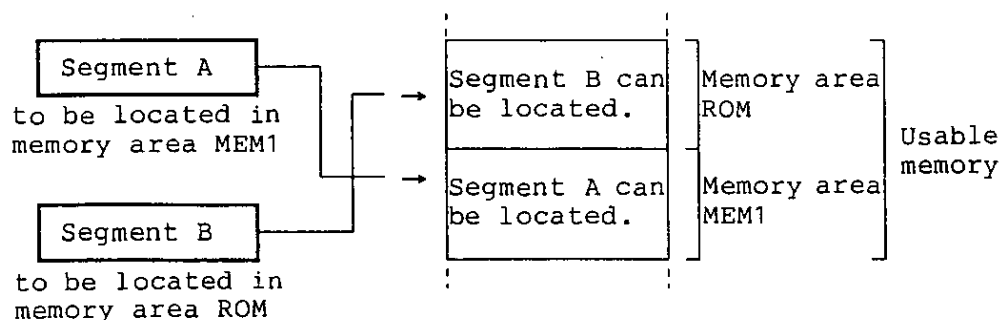


Fig. 5-17. Memory Area Name Specification

- (3) Each segment has a location type and the linker determines the location addresses of the segment according to the location type.
- (4) The location addresses of segments are all determined in units of bytes. With segment type BSEG, the location addresses of the segment are determined after converting its segment size to bytes with 8 bits taken as 1 byte (by rounding up a fractional part to 1 byte).

5.5.4 Procedure for locating segments

This subsection explains a segment locating operation.

- (1) Location address determination for segments is made in the order of the segment which has a location type of the highest priority.
- (2) If two segments have a location type of the same priority, location addresses will be determined in the order of the first input segment.

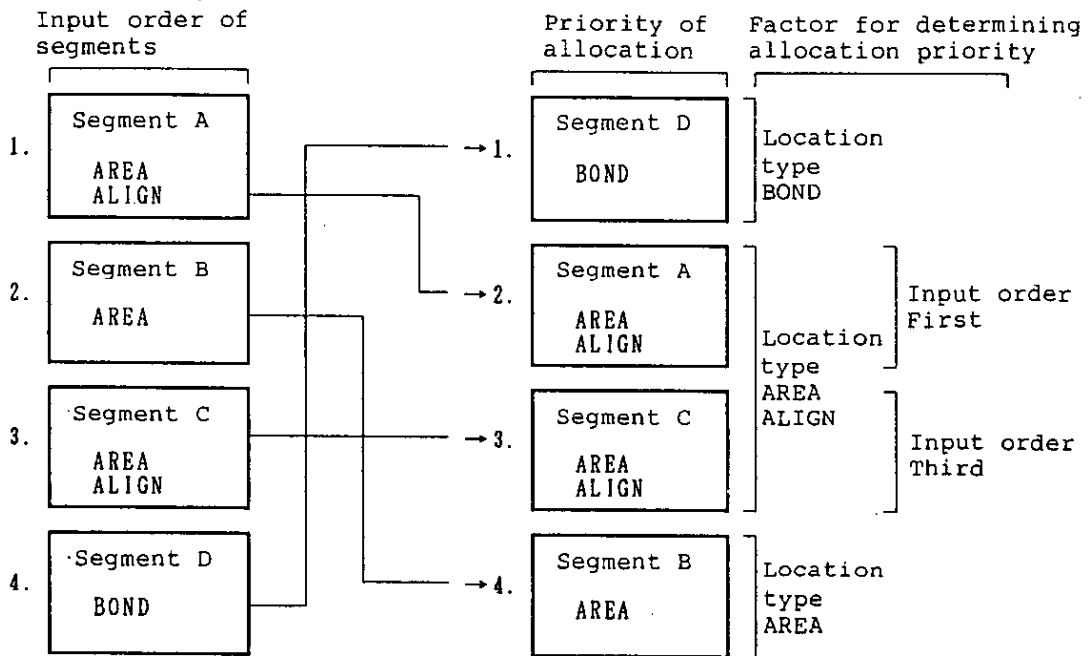


Fig. 5-18. Priority of Address Allocation to Segments by Location Type

- (3) If two or more usable memory areas exist for address allocation to a segment, the segment will be located in a memory area which has the lowest-order address.

5.6 Link Directives

Link directives (hereafter referred to as directives) refer to a group of pseudoinstructions which give the linker particular instructions on such as input files, usable memory areas, and address allocation to segments.

Each directive must be used according to the following procedure: The user must describe directives in a file using the editor (a file containing directives is called a directive file) and must specify the directive file by using the -D option when starting up the linker. The linker reads the directive file and performs a linking process while interpreting each directive in the file. These directives are available in the following two types.

Table 5-6. Types of Directives

| Item | Type of directive | Description |
|------|----------------------------|------------------------------------------------|
| 1 | Memory directive | Defines a usable memory area. |
| 2 | Segment location directive | Specifies the location addresses of a segment. |

5.6.1 Directive file

A directive file is a text file in which link directives have been described. Here, the basic conventions of description in a directive file are explained.

The description format (syntax) of each type of directive in a directive file is as shown below.

① Memory directive

```
MEMORY memory-area-name: (origin value,size)
                           [/memory-space-name]
```

② Segment location directive

```
MERGE segment-name:[location-type-definition]
                    [merge-type-definition][=bind-specification]
                    [/memory-space-name]
```

Two or more directives may be described in a directive file.

(1) Reserved words

In a directive file, the following five words are recognized by the linker as reserved words.

MEMORY MERGE AT SEQUENT COMPLETE

The user cannot use any of these reserved words in a directive file for any other meanings (segment name, memory area name, etc.).

All the reserved words may be described in either uppercase or lowercase letters but both uppercase and lowercase cannot be mixed to describe any of these reserved words.

Examples: MEMORY

memory

Memory ; Not acceptable

(2) Numeric values

When a numeric constant is to be described as the parameter of each directive, the numeric constant may be described in either decimal or hexadecimal numbers.

(3) Comment statement

If ";" or "#" is described in a directive file, characters between the ";" or "#" and an LF character will be handled as a comment statement. If the directive file reaches its end before an LF character appears in the directive file, characters up to the end of the file will be handled as a comment statement.

Example: The underlined part is interpreted as a comment statement.

; DIRECTIVE FILE FOR 78312

MEMORY MEM1:(1000H,1000H) # SECOND MEMORY AREA

5.6.2 Memory directive

The memory directive defines a memory area. The defined memory area may be given a name (memory area name) and referenced by using the segment location directive.

Up to 100 memory areas may be defined including those defined by default assumption.

Syntax

```
MEMORY Δ memory-area-name ▲: ▲(▲ origin value ▲, ▲ size ▲)
                               [ / ▲ memory-space-name ]
```

(1) Memory area name

The parameter "memory area name" specifies the name of the memory area to be defined.

- o Characters that can be used as a memory area name are A to Z, a to z, 0 to 9, -, ?, and @. However, none of the numeric characters 0 to 9 can be used as the first character of a memory area name.
- o Uppercase letters are distinguished as different characters from their lowercase equivalents.
- o Uppercase and lowercase may be mixed to describe a memory area name.
- o The maximum length of a memory area name is 31 characters. If any memory area name is described by exceeding this limit value, an error will result.
- o No two same named memory areas can be defined throughout the entire memory space. Giving the same memory area name to different memory areas is not allowed regardless of whether or not they are in the same memory space.

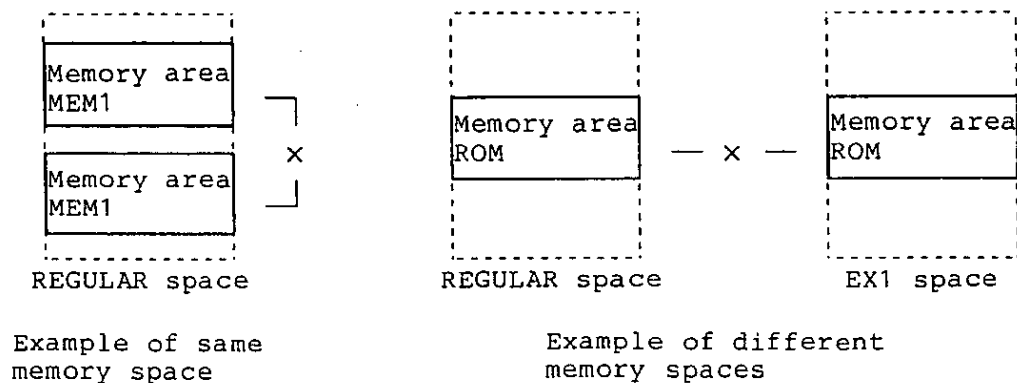


Fig. 5-19. Memory Area Name

(2) Origin value

The parameter "origin value" specifies the start address (first address) of the memory area to be defined.

- o An origin value must be described with a numeric constant in the value range from 0H to FFFFH.

(3) Size

The parameter "size" specifies the size of the memory area to be defined.

- o A numeric constant with a value of 1 or more must be described as the size of the memory area.
- o If you need to re-define the size of the memory area (ROM or RAM) defined by the linker by default assumption, note that there are restrictions on the address ranges that can be re-defined.

The size of each memory area defined by default assumption and its address range that can be re-defined in each target device are shown in Table 5-7 below.

Table 5-7. Default Memory Areas and Address Ranges That Can Be Re-defined

| Series name | Target device | ROM area | | IHRAM area | |
|-------------|---------------|--------------------|-------------------------|--------------------|-------------------------|
| | | Default assumption | Range for re-definition | Default assumption | Range for re-definition |
| 78K/O | 78012 | 0H to 3FFFH | 0H to FA7FH | FD00H to FEFFH | FD00H to FEFFH |
| | 78014 | 0H to 7FFFH | 0H to FA7FH | FB00H to FEFFH | FB00H to FEFFH |

| Series name | Target device | LVRAM area | |
|-------------|---------------|--------------------|-------------------------|
| | | Default assumption | Range for re-definition |
| 78K/O | 78012 | 0H to 3FFFH | 0H to FA7FH |
| | 78014 | 0H to 7FFFH | 0H to FA7FH |

Table 5-7. Default Memory Areas and Address Ranges That
Can Be Re-defined (contd)

| Series name | Target device | ROM area | | IHRAM area | |
|-------------|------------------|--------------------|-------------------------|--------------------|-------------------------|
| | | Default assumption | Range for re-definition | Default assumption | Range for re-definition |
| 78K/I | 78112 | 80H to 1FFFH | 0H to FE3FH | FE40H to FEFFH | 2000H to FEFFH |
| | 78134/ 78134A | 80H to 3FFFH | 0H to FD7FH | FD80H to FEFFH | 4000H to FEFFH |
| | 78136 | 80H to 5FFFH | 0H to FCFFH | FD00H to FEFFH | 6000H to FEFFH |
| | 78138 | 80H to 7FFFH | 0H to FC7FH | FC80H to FEFFH | 8000H to FEFFH |
| 78K/III | 78310/ 78310A | 0H to FDFH | 0H to FDFH | FE00H to FEFFH | 0H to FEFFH |
| | 78312/ 78312A | 0H to 1FFFH | 0H to FDFH | FE00H to FEFFH | 2000H to FEFFH |
| | 78320 | 0H to FC7FH | 0H to FC7FH | FC80H to FEFFH | 0H to FEFFH |
| | 78322 | 0H to 3FFFH | 0H to FC7FH | FC80H to FEFFH | 4000H to FEFFH |
| | 78330 | 0H to 7FFFH | 0H to FAFH | FB00H to FEFFH | 0H to FEFFH |
| | 78330 | 0H to 7FFFH | 0H to FAFH | FB00H to FEFFH | 0H to FEFFH |
| | 78334 | 0H to 7FFFH | 0H to FAFH | FB00H to FEFFH | 8000H to FEFFH |
| 78K/VI | 78600 | 0H to FAFH | 0H to FA7FH | FB00H to FEFFH | 0H to FEFFH |
| | 78602 | 0H to 3FFFH | 0H to FAFH | FB00H to FEFFH | 4000H to FEFFH |
| | 78603 | 0H to F6FFH | 0H to F6FFH | F700H to FEFFH | 0H to FEFFH |
| | 78604 | 0H to 7FFFH | 0H to F6FFH | F700H to FEFFH | 8000H to FEFFH |

With a target device which has an internal ROM, the internal ROM area becomes its default ROM area. With a target device which has no internal ROM, an area from address 0H to an address immediately before the internal RAM becomes its default ROM area. The internal RAM area of each target device serves as its default RAM area.

The address range that can be re-defined for the ROM area is an area from address 0H to an address immediately before the internal RAM, whereas that for the RAM area is an area excluding the internal ROM area and "sfr" area.

Note: With the 78K/0 series, default IHRAM and LRAM areas are the areas of the internal IHRAM and LRAM themselves (provided that for the UNIT type, no allocation to 0H to 7FH is allowed). The range allowed for redefinition of the LRAM area is only within the LRAM area.

- o On a memory space, two or more memory areas cannot be defined in the same address range.

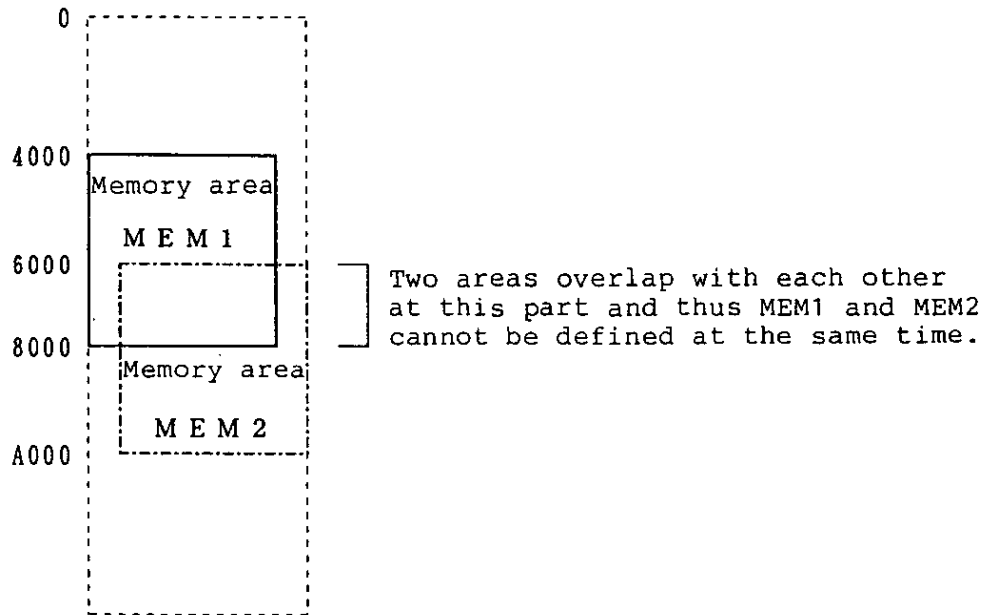


Fig. 5-20. Incorrect Memory Area Definition

(4) Memory space name

The parameter "memory space name" specifies the name of the memory space to which the defined memory area is to be assigned.

- o Any of the following 16 memory spaces may be specified as a memory space name.
REGULAR, EX1, EX2, EX3, EX4, EX5, EX6, EX7, EX8,
EX9, EX10, EX11, EX12, EX13, EX14, EX15
- o A memory space name must be described wholly in uppercase letters.
- o If a memory space name is omitted, the REGULAR space is assumed to have been specified.
- o If a memory space name is omitted following the description of "/", an error will result.

Function

- (1) The MEMORY directive defines a memory area which has the name specified by "memory area name" in the memory space specified by this directive.
- (2) One memory area can be defined per MEMORY directive.
- (3) Two or more MEMORY directives may be specified at the same time. In this case, the memory area names specified by the MEMORY directives are registered in the order of their specification. If the same named memory area name is defined two or more times, an error will result.
- (4) The default memory area (ROM/RAM) is valid unless the same memory area is re-defined with the MEMORY directive. If no MEMORY directive is specified, only the default memory areas (ROM and RAM) of each target device that the linker has are assumed to have been specified.

Application Examples

- (1) To define addresses 0H to 1FFH of default memory space REGULAR as memory area "ROMA"

```
MEMORY   ROMA : (0, 200H)
```

- (2) To define addresses 1F00H to 1FFFH of memory space EX1 as memory area "RAMA"

```
MEMORY   RAMA : (1F00H, 100H) / EX1
```

5.6.3 Segment location directive

The segment location directive locates a specified segment at specific address(es) on a memory area.

Syntax

```

MERGE Δsegment-name ▲: ▲[location-type-definition]
      [▲merge-type-definition] [ /▲memory-space-name]
      [▲=▲bind-specification] [▲/▲memory-space-name]

```

(1) Segment name

The parameter "segment name" specifies a segment name which is included in the object module file to be input to the linker.

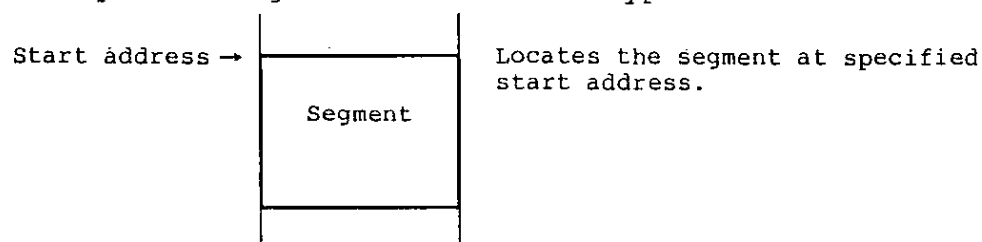
- o Other than an input segment cannot specified as a segment name.
- o If the -NCA option has been specified at assembly time, the segment name to be specified with this directive must be the one which have been described in the assembler source program.
- o If the -NCA option has not been specified at assembly time, the segment name must be described in uppercase letters.

(2) Location type definition

As the parameter "location type definition", only the following type can be specified.

AT ▲(▲start-address▲)

- o The reserved word "AT" must be described in either uppercase or lowercase letters. Both uppercase and lowercase cannot be mixed to describe this reserved word.
- o A start address must be described with a numeric constant.
- o The processing of the location type AT is as follows:



- Note:
- o If a location type definition is omitted, the location address(es) of the segment will be determined according to the relocation attribute specified by the segment directive in the source program.
 - o As a result of locating the segment according to the specified start address, if the segment is located beyond the address range of the memory area in which the segment should be located, an error will result.
 - o If two or more location type definitions are described in one MERGE directive, an error will result.
 - o A location type definition cannot be described for any segment which has been specified as AT with the segment directive or whose location address(es) have been specified with the ORG directive.
 - o A segment with a location type which has a predetermined default memory area in the absence of a location type definition specification can be located to address(es) other than the default area by specifying a location type definition for the segment.

(3) Merge type definition

The parameter "merge type definition" specifies the method of merging the specified segment if two or more same named input segments exist.

- o Only the following merge types can be specified with this directive.

Table 5-8. Merge Types That Can Be Specified with Directive

| Item | Merge type | Description |
|------|------------|-------------------------------------------------------------------------|
| 1 | SEQUENT | Merges segments sequentially in the order of their input to the linker. |
| 2 | COMPLETE | An error will result if same named segments exist. |

- o The reserved words SEQUENT and COMPLETE must be described in either uppercase or lowercase letters. Both uppercase and lowercase cannot be mixed to describe any of these reserved words.

Note: o If a merge type definition is omitted, the merge type SEQUENT is assumed to have been specified. However, if any of the same named input segments has the relocation attribute AT specified at assembly time, the merge type COMPLETE is assumed to have been specified.

- o Two or more merge type definitions cannot be described in one MERGE directive.
- o Only the merge type COMPLETE can be specified for any segment which has been specified as AT with the segment directive or whose location address(es) have been specified with the ORG directive or for which AT (start-address) has been specified in the location type definition.

(4) Bind specification

The parameter "bind specification" determines a memory area in which the segment is to be located. Only a "memory area name" can be specified in this bind specification.

- o If a bind specification is omitted, one of the following memory areas described with the segment directive at assembly time is assumed to have been specified:

CSEG ROM area

DSEG RAM area

BSEG RAM area

- o For a segment for which a bind specification has been described, the relocation attribute of the segment specified at assembly time is effective. Therefore, the memory area specified by a bind specification must allow segment location according to the relocation attribute specified at assembly time. If the segment cannot be located to satisfy conditions by both the relocation attribute and bind specification, an error will result.

For example, if you attempt to locate a segment which has the CALLT attribute in memory area MEM1 by a bind specification, the memory area MEM1 must contain an area in which the segment can be located, within the range from 40H to 7FH.

- o If two or more memory area names are specified as a bind specification, an error will result.

(5) Memory space name

The parameter "memory space name" specifies the name of the memory space in which the segment is to be located.

- o Any of the following 16 memory spaces may be specified as a memory space name.

REGULAR, EX1, EX2, EX3, EX4, EX5, EX6, EX7, EX8,
EX9, EX10, EX11, EX12, EX13, EX14, EX15

- o A memory space name must be described wholly in uppercase letters.
- o If a memory space name is omitted, the REGULAR space is assumed to have been specified.
- o If a memory space name is omitted following the description of "/", an error will result.
- o The linker locates the segment in the specified memory space. The memory area subject to actual segment location differs depending on the combination of the memory space specification and bind specification. The destination (memory area) to which the segment is to be located is shown in the following table.

Table 5-9. Segment Location by Combination of Bind Specification and Memory Space Specification

| Bind specification | Memory space | Destination of segment |
|--------------------|---------------|------------------------------------------------------|
| None | Not specified | Default memory area in the REGULAR space |
| None | Specified | Any memory area in the specified memory space. |
| Memory area name | Not specified | Specified memory area in the REGULAR space. |
| Memory area name | Specified | Specified memory area in the specified memory space. |

The above table merely indicates the significance of memory area definition subject to segment location. In practice, the linker determines the location address(es) of each segment to locate the segment by satisfying all other conditions including the location type definition.

Example: When memory space EX1 is specified but location type definition and bind specification are omitted for segment CSEG with relocation attribute FIXED

- ① By the default assumption of location type definition, the segment must be located within addresses 800H to FFFH.
- ② By the combination of bind specification and memory specification from the above table, the segment may be located in any memory area in memory space EX1.
- ③ To satisfy both conditions ① and ②, the segment is located in a memory area defined in the address range from 800H to FFFH in memory space EX1.

Cautions

- (1) For an input segment for which no MERGE directive is specified, the linker determines the location address(es) of the segment according to the relocation attribute of the segment specified with the segment directive at assembly time.
- (2) If the segment specified by a segment name does not exist, an error will result.
- (3) If the MERGE directive is specified two or more times for the same segment, an error will result.

Application Examples

Examples of allocating addresses to segment SEG1 whose relocation attribute is CSEG FIXED are shown here.

- (1) To allocate address 2000H in ROM area to input segment SEG1

```
MERGE  SEG1 : AT(2000H)
```

- (2) To locate input segment SEG1 in memory area MEM1

```
MERGE  SEG1 : = MEM1
```

- (3) To allocate address 2000H in memory area MEM1 to input segment SEG1

```
MERGE  SEG1 : AT(2000H) = MEM1
```

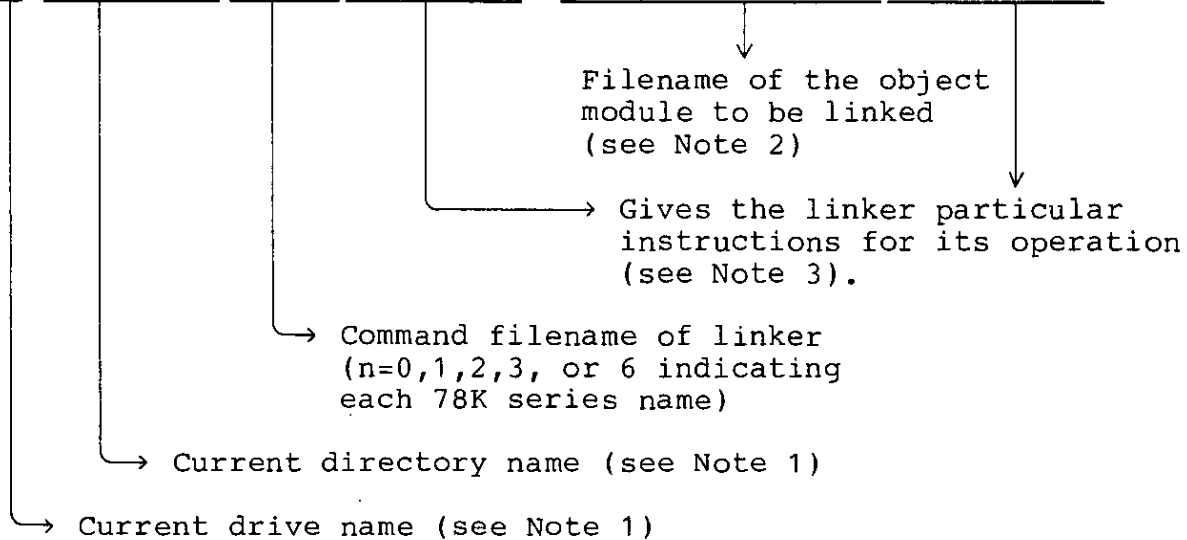
5.7 How to Start Up the Linker

5.7.1 Starting up the linker

The linker can be started up (invoked) in either of the following two ways:

- (1) Start-up with the start-up command line of the linker

X>[pathname]lk78kn[Δoption]...Δobject-filename[Δoption]... [Δ]



Example: A>lk78k3 78k3main.rel 78k3sub.rel -o78k3.lnk -g

- NOTE:
1. With MS-DOS V3.10, the command file and overlay files of the linker must have been stored in the same directory.
 2. The number of object module files that can be input is up to 128 with the 78K/0, 78K/I, or 78K/III, and up to 64 with the 78K/VI.
 3. If two or more linker options are to be specified, each linker option must be delimited with a space. See Section 5.8 for details of the linker options.

(2) Start-up with parameter file

A parameter file is used when all the required information for starting up the linker cannot be specified in the start-up command line of the linker or when the same linker options are to be used repeatedly in each linking process.

When using this parameter file, the -F option must be specified in the start-up command line of the linker to specify the use of the parameter file.

The linker can be started up with a parameter file as follows:

```
X>lk78kn[Δobject-filename] Δ-f parameter-filename
```

→ File containing information required to start up the linker

→ Option specifying parameter file

- o A parameter file must be created with the editor.
- o The description format of parameters within the parameter file is as shown below.

```
[ [ [ Δ ] option [ Δ option ] .. [ Δ ] Δ ] ] ...
```

- o If the object module filename to be input is omitted in the start-up command line of the linker, input object module filenames must be described in the parameter file.
- o The input object module filename may be described either before or after an option.
- o In the parameter file, all the linker options and output filename which should normally be specified in the start-up command line must be described.

Example: To create parameter file "78K3.PLK" with the editor

- o Contents of 78K3.PLK

```
;parameter file
78k3main.rel 78k3sub.rel -o78k3.lnk -p78k3.map -e
-ta:Ytmp -g
```

- o Start-up of linker using parameter file "78K3.PLK"

```
A>lk78k3 -f78k3.plk
```

5.7.2 Execution start and end messages

(1) Execution start message

When the linker is started up, the following message is output to the console, indicating the start of the linker execution.

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

(2) Execution end messages

- o If no linking error is found as a result of a linking operation, the linker will output the following message to the console and return control to the OS.

```
Link complete,      0 error(s) and      0 warning(s) found.
```

- o If any linking errors are found as a result of a linking operation, the linker will output the following message (the number of errors found) to the console and return control to the OS.

```
Object Converter complete,      2 error(s) and      0 warning(s) found.
```

- o If any fatal error is found during a linking operation, the linker will output the following message to the console, stop its processing, and return control to the OS.

Example 1:

```
A>lk78k3_samp1.rel_samp2.rel  
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx  
  
A006 File not found 'SAMP1.REL'  
A006 File not found 'SAMP2.REL'  
Program aborted.
```

In this example, the linking operation was discontinued by an abort error resulting from the specification of object module files which do not exist in drive A.

Example 2:

```
A>lk78k3_78k3main.rel_78k3sub.rel -a  
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx  
  
A018 Option is not recognized '-a'  
Program aborted.
```

In this example, the linking operation was discontinued by an abort error resulting from the input of a linker option "-A (-a)" which is not recognized by the linker.

If the linker is aborted following the output of an error message, check the cause of the error by referring to Chapter 11, Error Messages and take corrective action(s) as required.

5.8 Linker Options

5.8.1 Types of linker options

A linker option gives the linker particular instructions for its operation and is broadly divided into the following 14 types:

Table 5-10. Types of Linker Options

| No. | Classification | Option | Functional description |
|-----|------------------------------------------------------------|--------|-------------------------------------------------------------------------------------------------|
| 1 | Options for load module output specification | -O | Specifies the output or non-output of a load module file. |
| | | -NO | |
| 2 | Options for forced load module output specification | -J | Specifies the output or non-output of a load module file by force. |
| | | -NJ | |
| 3 | Options for debug information output specification | -G | Specifies the output or non-output of symbol information for debugging to the load module file. |
| | | -NG | |
| 4 | Options for stack-reserving symbol creation specification. | -S | Specifies the generation or non-generation of the PUBLIC symbols for stack area reservation. |
| | | -NS | |
| 5 | Option for directive file specification | -D | Specifies the input of the file specified by this option as a directive file. |
| 6 | Options for link list file output specification | -P | Specifies the output or non-output of a link list file. |
| | | -NP | |
| 7 | Options for link list file information specification | -KM | Specifies the output or non-output of a map list list to the link list file. |
| | | -NKM | |
| | | -KD | Specifies the output or non-output of a link directive file to the link list file. |
| | | -NKD | |
| | | -KP | Specifies the output or non-output of a PUBLIC symbol list to the link list file. |
| | | -NKP | |
| | | -KL | Specifies the output or non-output of a local symbol list to the link list file. |
| | | -NKL | |

Table 5-10. Types of Linker Options (contd)

| No. | Classification | Option | Functional description |
|-----|-------------------------------------------------------|--------|--------------------------------------------------------------------------------------------|
| 8 | Options for link list file format specification | -LL | Specifies the number of print lines per page of a list. |
| | | -LF | Specifies the addition or non-addition of a form-feed (FF) code to the end of a list file. |
| | | -NLF | |
| 9 | Options for error list file output specification | -E | Specifies the output or non-output of an error list file. |
| | | -NE | |
| 10 | Option for library file specification | -B | Specifies the input of the file specified by this option as a library file. |
| 11 | Option for library file read path specification | -I | Tells the linker to read library file(s) from the specified path(s). |
| 12 | Option for parameter file specification | -F | Specifies the input of input filename and options from the file specified by this option. |
| 13 | Option for temporary file creation path specification | -T | Specifies the creation of a temporary file on the path specified by this option. |
| 14 | Option for HELP message output specification | -- | Specifies the output of HELP message to the console. |

The above table merely introduces all the available linker options. Each of these linker options is detailed in Subsection 5.8.3 below. For quick reference, see Appendix C.2, List of Linker Options in which the description format of each option and the relationship between one option and the other are also outlined.

5.8.2 Priority of linker options

Table 5-11 shows the precedence order of linker options when two or more options are specified at the same time.

Table 5-11. Priority of Linker Options

| | -NO | -NG | -NP | -NKM | -NKP | -NKL | -- |
|-----|-----|-----|-----|------|------|------|----|
| -J | x | | | | | | x |
| -G | x | | | | | | x |
| -P | | | | △ | △ | △ | x |
| -KM | | | x | | | | x |
| -KD | | | x | x | | | x |
| -KP | | x | x | | | | x |
| -KL | | x | x | | | | x |
| -LL | | | x | | | | x |
| -LF | | | x | | | | x |

In Table 5-11, "X" in the table indicates that the option on the leftmost column becomes invalid if the option on the top column is specified.

Example: A>lk78k3 78k3main.rel 78k3sub.rel -np -km

In this case, the -KM option become invalid (because the -NP option has been specified).

"△" in the table indicates that the option on the leftmost column becomes invalid if all of the options on the top column are specified at the same time.

Example: A>lk78k3 78k3main.rel 78k3sub.rel -p -nkm -nkp -nkl

In this case, the -P option becomes invalid, because the -NKM, -NKP, and -NKL have been specified at the same time.

If two options contradicting each other (such as -O and -NO, -P and -NP, etc.) are specified at the same time, whichever you specified later will take precedence over the other.

Example: A>lk78k3 78k3main.rel 78k3sub.rel -o -no

In this example, the -NO option takes precedence over the -O option which has been specified before the -NO option.

The linker options not listed in Table 5-11 are not affected by any other linker options. However, if the "--" option (for HELP message output specification) is specified, all the other options specified at the same time with the "--" option become invalid.

5.8.3 Description of each linker option

A detailed description of each linker option is provided in this subsection.

(1) Options for load module file output specification (-O/-NO)

Description format: -O [output-filename]
or
-NO
Default assumption: -O input-filename.LNK

Function

- o The -O option specifies the output of a load module file. It also specifies the output destination or output filename of the load module file to be output by the linker.
- o The -NO option specifies the non-output of a load module file.

Use

- o Use the -O option if you want to change the output destination or output filename of a load module file.
- o If the linkage of object module files is to be performed only to output a link list file, use the -NO option. This will reduce the linkage time.

Explanation

- o As an output filename, either a disk type filename or a device type filename (NUL or AUX only) may be specified.
- o If a fatal error is found during a linking operation with the -O option specified, no load module file will be output by the linker.
- o If an output filename is omitted from the -O option specification, "input filename.LNK" is assumed as the output filename and the load module file under that name will be output to the current directory.
- o If only a pathname is specified as the output filename, the load module file named "input-filename.LNK" will be output to the specified path.

- o If the -O and -NO options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To link two object module files with -O option specified to output a load module file named "78K3.LNK"

```
A>lk78k3 78k3main.rel 78k3sub.rel -o78k3.lnk
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

Example 2: To link two object modules with both -NO and -O options specified

```
A>lk78k3 78k3main.rel 78k3sub.rel -no -o
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

In this case, -NO option becomes invalid and -O option is accepted as valid.

-J/-NJ

**Forced load module file output
specification**

(2) Options for forced load module file output specification
(-J/-NJ)

| |
|-------------------------|
| Description format: -J |
| or |
| -NJ |
| Default assumption: -NJ |

Function

- o The -J option specifies the output of a load module file even if a fatal error occurs during a linking operation.
- o The -NJ option is used to invalidate the -J option.

Use

Normally, the specified load module file will not be output if any fatal error occurs during the linking operation. Therefore, if you want to execute the program even in case of a fatal error, use the -J option to output a load module file.

Explanation

- o If a fatal error occurs during a linking operation with the -J option specified, a load module file will be output by the linker.
- o If the -J and -NJ options are specified at the same time, whichever you specified later will take precedence over the other.

-J/-NJ

Forced load module file output
specification

Application Examples

Example 1: To link object module files with -J option
specified to output a load module file even in case
of a fatal error

```
A>lk78k3 78k3main.rel 78k3sub.rel -i
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

(3) Options for debug information output specification (-G/-NG)

```

Description format: -G
                   or
                   -NG
Default assumption: -NG

```

Function

- o The `-G` option specifies the addition of debugging information (local symbol information) to the load module file to be output by the linker.
- o The `-NG` option specifies the non-addition of debugging information to the load module file.

Use

The -G option must always be specified when symbolic debugging is to be performed with the source debugger.

Explanation

- o If the -G option is specified at the same with the -NO option, the -G option will become invalid.
- o If the -G option is omitted, the -NG option is assumed and thus no debugging information will be output.
- o If the -G and -NG options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -NG option is specified at the same time with the -KP or -KL option, neither a PUBLIC symbol list nor a local symbol list will be output.

Application Examples

Example 1: To link two object module files with -G option
specified to add debugging information to a load
module file

A>lk78k3 78k3main.rel 78k3sub.rel -g

uCOM-78K/III Linker Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Link complete, 0 error(s) and 0 warning(s) found.

-S/-NS

Stack-reserving symbol creation
specification

(4) Options for stack-reserving symbol creation specification
(-S/-NS)

Description format: -S [area-name]

or

-NS

Default assumption: -NS

Function

- o The -S option tells the linker to create PUBLIC symbols "_@STBEG" and "_@STEND" for stack area reservation.
- o The -NS option tells the linker not to create the PUBLIC symbols for stack area reservation.

Use

Use the -S option to reserve a stack area.

Explanation

- o As an area name, the memory area defined by the user or memory area name defined by default assumption must be specified.
- o The linker distinguishes an area name described (partly or wholly) in uppercase letters from that described (partly or wholly) in lowercase letters.
- o The linker searches the memory area specified by the -S option for any free area which has the highest start address value and to which no segment is located. If any such free area is found, the linker creates a PUBLIC symbol "_@STBEG" which has as its value the start address of the free area and a PUBLIC symbol "_@STEND" which has as its value the last address of the free area + 1. These symbols will be handled as symbols which have been declared as PUBLIC and have the NUMBER attribute and will be registered at the end of the symbol table of the linker.

When these symbols are output to a link list file, the "Module Name" column of the list file is left blank.

- o If the size of any such free area found is less than 15 bytes, the linker will output a warning message.
- o If no such free area is found, the linker will also output a warning message. In this case, both "@STBEG" and "@STEND" will have "the last address of the memory area specified by -S option + 1" as their value.
- o If no area name is specified, "RAM" is assumed to have been specified.
- o If the -S and -NS options are specified at the same time, whichever you specified later will take precedence over the other.

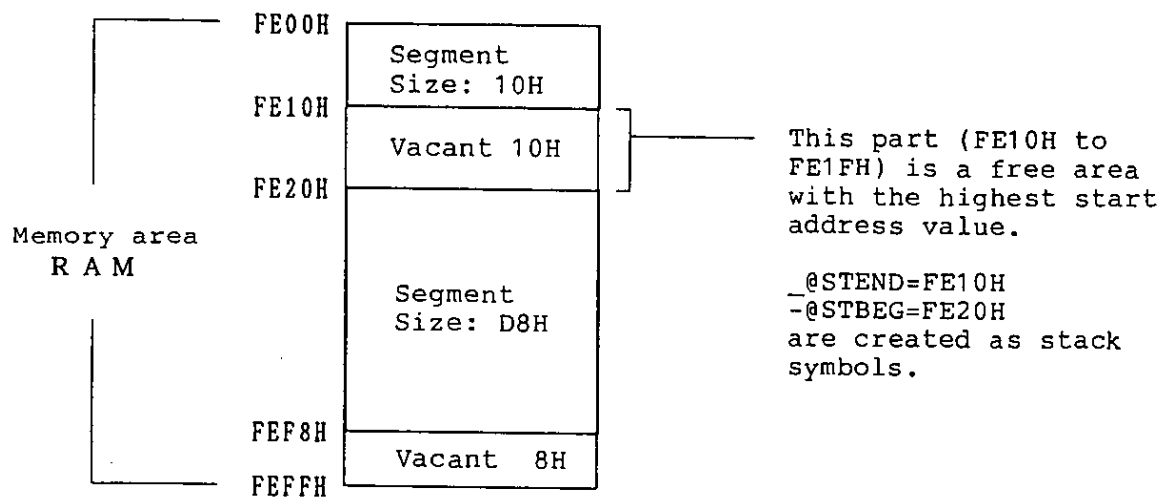
Application Examples

Example 1: To link two object module files with -S option specified to reserve a stack area in RAM (assuming that a segment with size 10H is to be input to RAM area and a segment with size D8H is to be input to saddr area)

```
A>lk78k3 78k3main.rel 78k3sub.rel -s
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```



(5) Option for directive file specification (-D)

Description format: -D filename

Default assumption: None

Function

- o The -D option tells the linker to input the file specified by this option as a directive file.

Use

A directive file is used when a new memory area is to be defined, when the default memory area is to be re-defined, or when a specific address is to be allocated to a segment. To input this directive file to the linker, the -D option must be specified.

Explanation

- o As a filename, only a disk type filename can be specified. If any device type filename is specified, an abort error will result.
- o If a filename is omitted from the -D option specification, an abort error will result.
- o Nesting of directive files is not allowed.
- o The number of characters that can be described in a directive file is not limited.
- o If two or more -D options are specified at the same time or if two or more filenames are specified in the -D option specification, an abort error will result.
- o For details of the directive file, see Section 5.6, Link Directives.

Application Examples

Example 1: To re-define default memory areas ROM and RAM

o Contents of directive file "78K3.DR"

```
memory ROM : (0000h,3FFFh)
memory RAM : (D000h,2EFFh)
```

o To link two object modules with -D option specified to
use directive file 78K3.DR

```
A>lk78k3 78k3main.rel 78k3sub.rel -d78k3.dr
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

(6) Options for link list file output specification (-P/-NP)

```
Description format: -P [output-filename]
                    or
                    -NP
Default assumption: -P input-filename.MAP
```

Function

- o The -P option specifies the output of a link list file. It also specifies the output destination or output filename of the link list file to be output by the linker.
- o The -NP option specifies the non-output of a link list file.

Use

- o Use the -P option if you want to change the output destination or output filename of a link list file.
- o If the linkage of object modules is to be performed only to output a load module file, use the -NP option. This will reduce the linkage time.

Explanation

- o An output filename can be specified with either a disk type filename or a device type filename. Only the following device type filenames can be used with this option: CON, PRN, NUL, and AUX. If "CLOCK" is specified as an output filename, an abort error will result.
- o If an output filename is omitted from the -P option specification, a link list file named "input filename.MAP" will be output to the current directory.
- o If only a pathname is specified as the output filename, a link list file named "input filename.MAP" will be output to the specified path.

- o If the -P and -NP options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To link two object modules files with -P option specified to create a link list file named "78K3.MAP"

```
A>lk78k3 78k3main.rel 78k3sub.rel -p78k3.map
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

Example 2: To link two object module files with -P option specified to output a link list file to PRN (Printer)

```
A>lk78k3 78k3main.rel 78k3sub.rel -pprn
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

-KM/-NKM

Link list file information
specification

(7) Options for link list file information specification

(-KM/-NKM, -KD/-NKD, -KP/-NKP, -KL/-NKL)

(a) Options for map list output specification (-KM/-NKM)

| |
|-------------------------|
| Description format: -KM |
| or |
| -NKM |
| Default assumption: -KM |

Function

- o The -KM option specifies the output of a map list to the link list file.
- o The -NKM option specifies the non-output of a map list to the link list file.

Use

Use the -KM option if you want to have only a map list as the contents of the link list file specified with the -P option.

Explanation

- o If the -KM and -NKM options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -KD option is specified at the same time with the -NKM option, no directive file will be output to the link list file.
- o If the -NKM, -NKP-, and -NKL options are all specified together with the -P option, no link list file will be output.

-KM/-NKM

Link list file information
specification

Application Examples

Example 1: To link two object module files with -P and -KM options specified to output a map list to link list file "78K3.MAP"

```
A>lk78k3 78k3main.rel 78k3sub.rel -p78k3.map -km
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

- o When link list file "78K3.MAP" is referenced, you will find that the following map list has been output to the link list file.

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 1

Command: 78k3main.rel 78k3sub.rel -p78k3.map -km
 Para-file:
 Out-file: 78K3MAIN.LNK
 Map-file: 78K3.MAP
 Direc-file:
 Directive:

*** Link information ***

3 output segment(s)
 40H byte(s) real data
 17 symbol(s) defined

*** Memory map ***

SPACE=REGULAR

MEMORY=ROM

BASE ADDRESS=0000H SIZE=FE00H

| OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS | SIZE | |
|-------------------|------------------|-----------------|-----------------|-------|---------|
| CODE | | | 0000H | 0002H | CSEG AT |
| | CODE | SAMPM | 0000H | 0002H | |
| ?CSEG | | | 0002H | 003CH | CSEG |
| | ?CSEG | SAMPM | 0002H | 0020H | |
| | ?CSEG | SAMPS | 0022H | 001CH | |
| | | | 003EH | FDC2H | |

* gap *

MEMORY=RAM

BASE ADDRESS=FE00H SIZE=0100H

| OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS | SIZE | |
|-------------------|------------------|-----------------|-----------------|-------|---------|
| DATA | | | FE00H | 0020H | |
| | DATA | SAMPM | FE20H | 0003H | DSEG AT |
| | | | FE20H | 0003H | |
| | | | FE23H | 00DDH | |

* gap *

Map
list

(b) Options for directive file output specification (-KD/-NKD)

| |
|------------------------------------------------------------------|
| Description format: -KD or -NKD Default assumption: -KD |
|------------------------------------------------------------------|

Function

- o The -KD option specifies the output of a directive file to the link list file.
- o The -NKD option specifies the non-output of a directive file to the link list file.

Use

Use the -KD option if you want to have a directive file as the contents of the link list file specified with the -P option.

Explanation

- o If the -KD and -NKD options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -KD option is specified at the same time with the -NKM option, no directive file will be output to the link list file.
- o If the -NKM, -NKP-, and -NKL options are all specified together with the -P option, no link list file will be output.

-KD/-NKD

**Link list file information
specification**

Application Examples

Example 1: To link two object module files with -D, -P, and
-KD options specified to output a directive file to
link list file "78K3.MAP"

```
A>lk78k3 78k3main.rel 78k3sub.rel -d78k3.dr -p78k3.map -kd
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

- o When link list file "78K3.MAP" is referenced, you will find that the following directive file has been output to the link list file.

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 1

Command: 78k3main.rel 78k3sub.rel -d78k3.dr -p78k3.map -kd

Para-file:

Out-file: 78K3MAIN.LNK

Map-file: 78K3.MAP

Direc-file:78K3.DR

Directive: memory ROM : (0000H,3FFFH)

memory RAM : (F000H,0EFFH)

← Name of directive file

← Contents of directive file

*** Link information ***

3 output segment(s)
40H byte(s) real data
17 symbol(s) defined

*** Memory map ***

SPACE=REGULAR

MEMORY=ROM

BASE ADDRESS=0000H SIZE=3FFFH

| OUTPUT SEGMENT CODE | INPUT SEGMENT CODE | INPUT MODULE | BASE ADDRESS | SIZE | |
|---------------------------|--------------------------|-----------------|-----------------|-------|---------|
| | | | 0000H | 0002H | CSEG AT |
| | | | 0000H | 0002H | |
| ?CSEG | | SAMPM | 0002H | 003CH | CSEG |
| | ?CSEG | SAMPM | 0002H | 0020H | |
| | ?CSEG | SAMPS | 0022H | 001CH | |
| | | | 003EH | 3FC1H | |

* gap *

MEMORY=RAM

BASE ADDRESS=F000H SIZE=0EFFH

| OUTPUT SEGMENT DATA | INPUT SEGMENT DATA | INPUT MODULE | BASE ADDRESS | SIZE | |
|---------------------------|--------------------------|-----------------|-----------------|-------|---------|
| | | | F000H | 0E20H | |
| | | | FE20H | 0003H | DSEG AT |
| | | | FE20H | 0003H | |
| | | SAMPM | FE23H | 00DCH | |

* gap *

-KP/-NKP

Link list file information
specification

(c) Options for PUBLIC symbol list output specification
(-KP/-NKP)

| |
|-------------------------------------------------------------------|
| Description format: -KP or -NKP Default assumption: -NKP |
|-------------------------------------------------------------------|

Function

- o The -KP option specifies the output of a PUBLIC symbol list to the link list file.
- o The -NKP option specifies the non-output of a PUBLIC symbol list to the link list file.

Use

Use the -KP option to output a PUBLIC symbol list as the contents of the link list file specified with the -P option.

Explanation

- o If the -KP and -NKP options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -NKM, -NKP, and -NKL options are all specified together with the -P option, no link list file will be output.
- o If the -KP and -NG options are specified at the same time, the -KP option will be ignored and no PUBLIC symbol list will be output.

-KP/-NKP

Link list file information
specification

Application Examples

Example 1: To link two object module files with -G, -P, and
-KP options specified to output a PUBLIC symbol
list to link list file "78K3.MAP"

```
A>lk78k3 78k3main.rel 78k3sub.rel -g -p78k3.map -kp
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

- o When link list file "78K3.MAP" is referenced, you will find that the following PUBLIC symbol list has been output to the link list file.

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 1

Command: 78k3main.rel 78k3sub.rel -g -p78k3.map -kp
Para-file:
Out-file: 78K3MAIN.LNK
Map-file: 78K3.MAP
Direc-file:
Directive:

*** Link information ***

3 output segment(s)
40H byte(s) real data
17 symbol(s) defined

*** Memory map ***

SPACE=REGULAR

MEMORY=ROM

BASE ADDRESS=0000H SIZE=FE00H

| | OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS | SIZE | |
|--|-------------------|------------------|-----------------|-----------------|-------|---------|
| | CODE | | | 0000H | 0002H | CSEG AT |
| | | CODE | SAMPM | 0000H | 0002H | |
| | ?CSEG | | | 0002H | 003CH | CSEG |
| | | ?CSEG | SAMPM | 0002H | 0020H | |
| | | ?CSEG | SAMPS | 0022H | 001CH | |
| | | | | 003EH | FDC2H | |

* gap *

MEMORY=RAM

BASE ADDRESS=FE00H SIZE=0100H

| | OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS | SIZE | |
|--|-------------------|------------------|-----------------|-----------------|-------|---------|
| | DATA | | | FE00H | 0020H | |
| | | DATA | SAMPM | FE20H | 0003H | DSEG AT |
| | | | | FE20H | 0003H | |
| | | | | FE23H | 00DDH | |

* gap *

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 2

*** Public symbol list ***

| MODULE | ATTR | VALUE | NAME |
|--------|------|-------|--------|
| SAMPM | ADDR | 0000H | MAIN |
| SAMPM | ADDR | 0002H | START |
| SAMPS | ADDR | 0022H | CONVAH |

PUBLIC symbol
list

-KL/-NKL

Link list file information
specification

(d) Options for local symbol list output specification
(-KL/-NKL)

Description format: -KL

or

-NKL

Default assumption: -NKL

Function

- o The -KL option specifies the output of a local symbol list to the link list file.
- o The -NKL option specifies the non-output of a local symbol list to the link list file.

Use

Use the -KL option to output a local symbol list as the contents of the link list file specified with the -P option.

Explanation

- o If the -KL and -NKL options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -NKM, -NKP, and -NKL options are all specified together with the -P option, no link list file will be output.
- o If the -KL and -NG options are specified at the same time, the -KL option will be ignored and no local symbol list will be output.

-KL/-NKL

**Link list file information
specification**

Application Examples

Example 1: To link two object module files with -G, -P, and
-KL options specified to output a local symbol list
to link list file "78K3.MAP"

```
A>lk78k3 78k3main.rel 78k3sub.rel -g -p78k3.map -kl
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

- o When link list file "78K3.MAP" is referenced, you will find that the following local symbol list has been output to the link list file.

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 1

Command: 78k3main.rel 78k3sub.rel -g -p78k3.map -kl
Para-file:
Out-file: 78K3MAIN.LNK
Map-file: 78K3.MAP
Direc-file:
Directive:

*** Link information ***

3 output segment(s)
40H byte(s) real data
17 symbol(s) defined

*** Memory map ***

SPACE=REGULAR

MEMORY=ROM

BASE ADDRESS=0000H SIZE=FE00H

| OUTPUT SEGMENT CODE | INPUT SEGMENT CODE | INPUT MODULE SAMPM | BASE ADDRESS 0000H | SIZE 0002H | |
|---------------------------|--------------------------|--------------------------|--------------------------|---------------|---------|
| | | | 0000H | 0002H | CSEG AT |
| ?CSEG | | SAMPM | 0002H | 003CH | CSEG |
| | ?CSEG | SAMPM | 0002H | 0020H | |
| | ?CSEG | SAMPS | 0022H | 001CH | |
| | | | 003EH | FDC2H | |

* gap *

MEMORY=RAM

BASE ADDRESS=FE00H SIZE=0100H

| OUTPUT SEGMENT DATA | INPUT SEGMENT DATA | INPUT MODULE SAMPM | BASE ADDRESS FE00H | SIZE 0020H | |
|---------------------------|--------------------------|--------------------------|--------------------------|---------------|---------|
| | | | FE20H | 0003H | DSEG AT |
| | | | FE20H | 0003H | |
| | | | FE23H | 00DDH | |

* gap *

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 2

*** Local symbol list ***

| MODULE | ATTR | VALUE | NAME |
|--------|------|-------|-------|
| SAMPM | MOD | | SAMPM |
| SAMPM | DSEG | | DATA |
| SAMPM | ADDR | FE20H | HDTSA |
| SAMPM | ADDR | FE21H | STASC |
| SAMPM | CSEG | | CODE |
| SAMPM | CSEG | | ?CSEG |
| SAMPS | MOD | | SAMPS |
| SAMPS | CSEG | | ?CSEG |
| SAMPS | ADDR | 0035H | SASC |
| SAMPS | ADDR | 003BH | SASC1 |

Local symbol
list

(8) Options for link list file format specification

(-LL, -LF/-NLF)

(a) Option for page length specification (-LL)

Description format: -LL [No. of lines per page]

Default assumption: -LL66 (No form-feed operation
with console output)

Function

- o The -LL option specifies the number of lines per page of a link list file.

Use

Use the -LL option if you want to change the number of lines per page of a link list file.

Explanation

- o The number of print lines per page to be specified with the -LL option must be within the following value range:
$$20 \leq \text{No. of print lines per page} \leq 32767$$

If any value beyond this range or other than a value is specified with this option, an abort error will result.
- o If the number of print lines per page is omitted, a value of 66 is assumed to have been specified.
- o If "0" is specified as the number of print lines per page, no form-feed operation (page ejection) will be carried out.
- o If the -LL option is specified at the same time with the -NP option, the -LL option will be ignored and thus will become invalid.

Application Examples

Example 1: To link two object module files with -P and -LL options specified to set 20 lines as No. of lines per page of link list file

```
A>lk78k3 78k3main.rel 78k3sub.rel -p78k3.map -l120
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete.      0 error(s) and      0 warning(s) found.
```

- o When file "78K3.MAP" is referenced, the output link list file will look like this.

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 1

Command: 78k3main.rel 78k3sub.rel -p78k3.map -l120
 Para-file:
 Out-file: 78K3MAIN.LNK
 Map-file: 78K3.MAP
 Direc-file:
 Directive:

*** Link information ***

3 output segment(s)
 40H byte(s) real data
 17 symbol(s) defined

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 2

*** Memory map ***

SPACE=REGULAR

MEMORY=ROM

BASE ADDRESS=0000H SIZE=FE00H

| OUTPUT SEGMENT CODE | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS 0000H | SIZE 0002H | CSEG AT |
|---------------------------|------------------|-----------------|--------------------------|---------------|---------|
|---------------------------|------------------|-----------------|--------------------------|---------------|---------|

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 3

| CODE | SAMPM | 0000H | 0002H | |
|-------|-------|-------|-------|------|
| ?CSEG | | 0002H | 003CH | CSEG |
| ?CSEG | SAMPM | 0002H | 0020H | |
| ?CSEG | SAMPS | 0022H | 001CH | |
| | | 003EH | FDC2H | |

* gap *

MEMORY=RAM

BASE ADDRESS=FE00H SIZE=0100H

| OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS FE00H | SIZE 0020H | |
|-------------------|------------------|-----------------|--------------------------|---------------|---------|
| DATA | | | FE20H | 0003H | DSEG AT |

* gap *

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 4

| DATA | SAMPM | FE20H | 0003H |
|------|-------|-------|-------|
| | | FE23H | 00DDH |

* gap *

(b) Options for form-feed code addition specification
(-LF/-NLF)

| | |
|---------------------|------|
| Description format: | -LF |
| | or |
| | -NLF |
| Default assumption: | -NLF |

Function

- o The -LF option specifies the addition of a form-feed (FF) code to the end of a link list file.
- o The -NLF option specifies the non-addition of a form-feed (FF) code to the end of a link list file.

Use

If you want to have a new page after printing the contents of a link list file, add a form-feed (FF) code to the end of the link list file by specifying the -LF option.

Explanation

- o If the -LF option is specified at the same time with the -NP option, the -LF option will become invalid.
- o If the -LF and -NLF options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To link two object module files with -P and -LF options specified to add an FF code to the end of link list file

A>lk78k3 78k3main.rel 78k3sub.rel -p78k3.map -lf

uCOM-78K/III Linker Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Link complete, 0 error(s) and 0 warning(s) found.

(9) Options for error list file output specification (-E/-NE)

Description format: -E [output-filename]

or

-NE

Default assumption: -NE

Function

- o The -E option specifies the output of an error list file. It also specifies the output destination or output filename of the error list file to be output by the linker.
- o The -NE option specifies the non-output of an error list file.

Use

- o Use the -E option if you want to save error messages to a file.
- o Also use the -E option if you want to change the output destination or output filename of an error list file.

Explanation

- o An output filename can be specified with either a disk type filename or a device type filename. However, if a device type filename "CLOCK" is specified as an output filename, an abort error will result.
- o If an output filename is omitted from the -E option specification, "input filename.ELK" is assumed as the output error list filename.
- o If a drive name is omitted from the -E option specification, the error list file will be output to the current drive.
- o If the -E and -NE options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To link two object module files with -D and -E options specified to create an error list file named "78K3.ELK"

```
A>lk78k3 78k3main.rel 78k3sub.rel -dsamp.dr -e78k3.elk
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
SAMP.DR(5) : F204 Segment 'CODE' may not change attribute  
SAMP.DR(5) : F207 Segment 'CODE' may not change arrangement  
*** ERROR A199 Cannot create output file for PASS1 ERROR(S)  
Program Aborted.
```

When error list file "78K3.ELK" is referenced due to an error in the directive file contents, the output error list file will look like this.

```
SAMP.DR(5) : F204 Segment 'CODE' may not change attribute  
SAMP.DR(5) : F207 Segment 'CODE' may not change arrangement  
*** ERROR A199 Cannot create output file for PASS1 ERROR(S)
```

(10) Option for library file specification (-B)

Description format: -B filename

Default assumption: None

Function

The -B option tells the linker to input the file specified by this option as a library file.

Use

Both the assembler and linker create one file for each output module. The more the number of input object modules, the more the number of output module files. Thus, a function to collect two or more modules into a single file is provided. This collection of modules is called a library and a file used to maintain and make available the modules is called a library file. If you create a library file of commonly used modules, files can be efficiently managed and operated. Use the -B option to input a library file to the linker.

Explanation

- o Other than a disk type filename cannot be specified as a filename.
- o A filename cannot be omitted from the -B option specification.
- o If a pathname is included in the filename specification with the -B option, the specified library file will be input from the specified path. If the specified library file does not exist on the specified path, an error will result.
- o If a filename is specified without including a pathname, the linker will search the path specified by the -I option or the default search path for the specified library file.

- o If two or more -B options are specified at the same time, library files will be input in the order of their specification by -B options. Up to 10 -B options may be specified at the same time.
- o See Chapter 7, Librarian for how to create a library file.

Application Examples

Example 1: To link object module file 78K3MAIN.REL with -B option specified to input library file "78K3.LIB" (which contains object module file 78K3SUB.REL)

```
A>lk78k3 78k3main.rel -b78k3.lib
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

(11) Option for library file read path specification (-I)

Description format: -I pathname[, pathname] ...

Default assumption: Path specified by environment
variable LIB78Kn (n=0, 1, 2, 3, or 6)
or current path if not specified by
LIB78Kn

Function

The -I option tells the linker to input library file(s) from the path(s) specified by this option.

Use

Use the -I option if you want to search a library file from a path.

Explanation

- o The -I option is valid only when a library file name which does not include any pathname has been specified with the -B option.
- o Two or more -I options can be specified at the same time. Two or more path names may be specified per -I option by delimiting each pathname with "," (comma). In this case, no blank (space) is allowed before and after the delimiter ",".
- o Up to 10 pathnames may be specified per linking process. If two or more pathnames are specified, the linker will search the specified paths in their order of specification, for the specified library file(s).
- o Even if the specified library file is not found in the specified path(s), the linker will not output an error message.
- o If no pathname is specified, an abort error will result.

- o If a library file is specified with the -B option without including any pathname, the linker will search paths in the following order.
 - ① Path specified by the -I option
 - ② Path specified by environment variable LIB78Kn (when the -I option is omitted)
 - ③ Current path (when no path is specified by environment variable LIB78Kn)

If the specified library filename is not found in any of the above three paths, an error will result.

Application Examples

Example 1: To link two object module files with -B and -I options specified to search directory LIB for library file "78K3.LIB"

```
A>lk78k3 78k3main.rel 78k3sub.rel -b78k3.lib -iVlib
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

(12) Option for parameter file specification (-F)

Description format: -F filename

Default assumption: Options and input filenames can
be input only from start-up
command line.

Function

The -F option tells the linker that options and input file-name(s) will be input from the file specified by this option.

Use

- o Use the -F option if all the required parameters for starting up the linker cannot be specified in the start-up command line.
- o If you have a set of linker options which you must specify repeatedly at each linking operation, describe these linker options in a parameter file and then specify the -F option in the start-up command line.

Explanation

- o A filename can be specified with only a disk type filename. If any device type filename is specified with this option, an abort error will result.
- o If a filename is omitted from the -F option specification, an abort error will also result.
- o Nesting of parameter files is not allowed. If the -F option is specified in a parameter file, an abort error will result.
- o The number of characters that can be described in a parameter file is not limited.
- o A blank character, Tab character, or " " must be used as a delimiter between options or input filenames.

- o The options and input filenames described in the parameter file will be expanded to the location on the command line where the parameter file (-F option) has been specified.
- o The linker will process these expanded options in the order from the last input option.
- o All characters described between ";" and "␣" or an EOF code will be interpreted as a comment statement.
- o If two or more -F options are specified at the same time, an abort error will result.

Application Examples

Example 1: To link two object module files with -F option specified

- o Contents of parameter file "78K3.PLK"

```
;parameter file
78k3main.rel 78k3sub.rel -o78k3.lnk -p78k3.map -e
-ta:Ytmp -g
```

- o Enter -F option in the command line as follows:

```
A>lk78k3 -f78k3.plk
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

-T

Temporary file creation
path specification

(13) Option for temporary file creation path specification (-T)

Description format: -T pathname

Default assumption: Temporary file is created on the path
specified by environment variable TMP
or on the current path if no path is
specified by TMP

Function

The -T option tells the linker to create a temporary file on the path specified by this option.

Use

The -T option can be used to specify where a temporary file is to be created.

Explanation

- o Other than a path cannot be specified as a pathname. If a pathname is omitted from the -T option specification, an abort error will result.
- o Even if a previously created temporary file exists, the linker will create a temporary file by overwriting the file unless it is write-protected when a temporary file creation is requested next time.
- o If the required memory space for temporary file creation is available, the linker will create a temporary file in memory. If no memory space is available for temporary file creation, the linker will save the memory contents to another file and create a temporary file in that space.
- o The temporary file created for a linking process by the linker will be erased on completion of the linking process. The temporary file will also be erased when the linking process is discontinued by CTRL-C key input.

-T

Temporary file creation
path specification

o A path for temporary file creation is determined in the following order:

- ① Path specified by the -T option
- ② Path specified by environment variable TMP (when the -T option is omitted)
- ③ Current path (when no path is specified by environment variable TMP)

If a temporary file cannot be created on the path specified by ① or ②, an abort error will result.

Application Examples

Example 1: To link two object module files with -T option specified to create a temporary file on directory TMP

```
A>lk78k3 78k3main.rel 78k3sub.rel -tYtmp
```

```
uCOM-78K/III Linker Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Link complete,      0 error(s) and      0 warning(s) found.
```

--

HELP message display specification

(14) Option for HELP message display specification (--)

Description format: --

Default assumption: No HELP message is displayed.

Function

The -- option tells the linker to display the HELP message on the console.

Use

The HELP message is a list of all linker options and their functional descriptions. Use the -- option if you want to refer to this message when executing the linker.

Explanation

If the -- option is specified, all the other linker options specified at the same time will become invalid.

Application Examples

Example 1: Input -- option as shown below and the HELP message will be displayed on the screen.

A>lk78K3 --

uCOM-78K/III Linker Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

usage : LK78K3 [option[...]] input-file[...] [option[...]]

option is as follows

-Ffile :Read directive file from specified file.
-O[file]/-NO :Create load module file [with specified name] / Not.
-P[file]/-NP :Create link map file [with specified name] / Not.
-E[file]/-NE :Create error list file [with specified name] / Not.
-KM/-NKM :Output map list to link map file / Not.
-KP/-NKP :Output public symbol list to link map file / Not.
-KL/-NKL :Output local symbol list to link map file / Not.
-LW[page width] :Specify map list file columns.
-LL[page length] :Specify map list file lines per page.
-LF/-NLF :Add Form Feed code at end of link map file / Not.
-G/-NG :Output symbol information to load module file / Not.
-J/-NJ :Create load module file if error occurred / Not.
-Tdirectory :Set temporary directory.
-- :Show this message
DEFAULT ASSIGNMENT: -O -P -NE -KM -NKP -NKL -LW132 -LL66 -LF -G -T.

CHAPTER 6. OBJECT CONVERTER

The Object Converter accepts a load module file output by the linker for the 78K series (in which all reference address information has been resolved) as an input file and outputs it as a HEX-format object module file (or HEX file for short).

The object converter also outputs information on symbols required for symbolic debugging as a symbol table file.

If any object converter error exists, the object converter outputs an error message to clearly indicate the cause of the error.

6.1 Input/Output Files of Object Converter

The files listed in Table 6-1 below are input and output to and from the object converter.

Table 6-1. I/O Files of Object Converter

| Type | Name and description of file | Default file type |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| Input files | <u>Load module file</u> A binary image file containing object code as a result of a linking process. This file is output by the linker. | .LNK |
| | <u>Parameter file</u> A file containing the parameters of an executable program. This file must be created by the user. | .POC |
| Output files | <u>HEX-format object module file</u> .. A file generated by converting an input load module file in HEX object format. | .HEX |
| | <u>Symbol table file</u> ... A file containing information on symbols in each module in the input load module file. | .SYM |
| | <u>Error list file</u> A file containing information on errors during object conversion process. | .EOC |

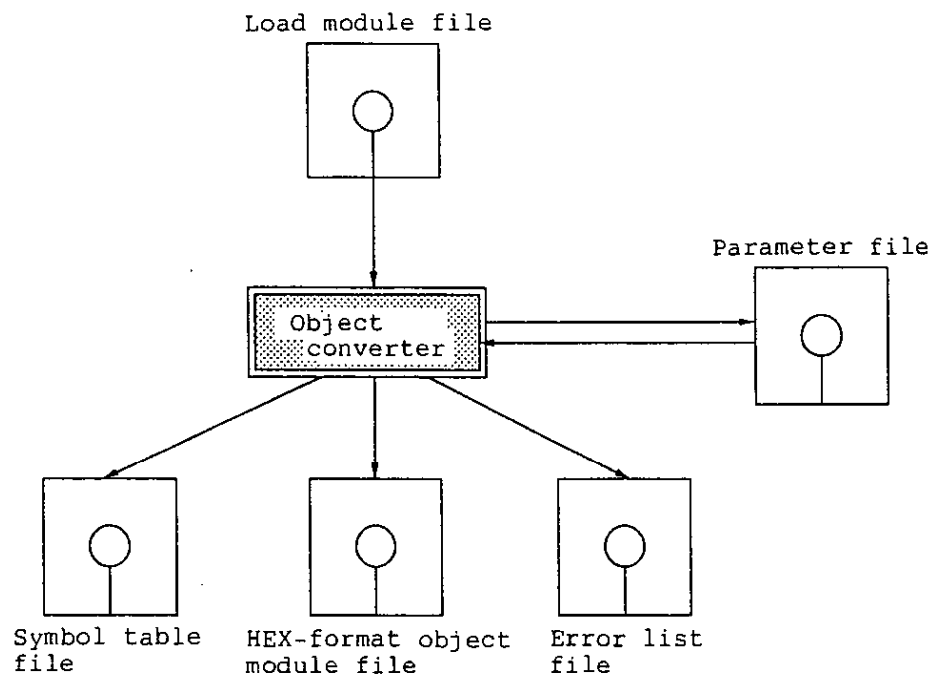


Fig. 6-1. I/O Files of Object Converter

6.2 Object Converter Functions

(1) Converter action taken for extension spaces

When raw data have been output to segments located in extension spaces, the object converter creates a separate HEX-format object module file for each extension space.

When symbols which have the ADDRESS attribute or BIT attribute have been defined in segments located in extension spaces, the object converter creates a separate symbol table file for each extension space. The object converter outputs symbols which have the NUMBER attribute to a symbol table file for the regular space.

The file type of a HEX-format object module file and that of a symbol table file for each extension space are shown in Table 6-2 below.

Table 6-2. File Types of Output Files for Each Extension Space

| File | Regular space | Extension space | | | | | | | |
|--------|---------------|-----------------|-----|-----|-----|-----|------|------|------|
| | REGULAR | EX1 | EX2 | EX3 | EX4 | ... | EX13 | EX14 | EX15 |
| HEX | .HEX | .H1 | .H2 | .H3 | .H4 | ... | .H13 | .H14 | .H15 |
| Symbol | .SYM | .S1 | .S2 | .S3 | .S4 | ... | .S13 | .S14 | .S15 |

(2) HEX-format object module file

The HEX-format object module file output by the object converter can be input to a HEX loader such as a ROM writer or a debugger.

The HEX-format object module file of the sample program is shown below.

```
:020000000200FC
:100002002B41000BFC80FE2B40000944F7083A20EC
:100012001A6720FE2822006521FED350D25014FE1A
:10002200B900059F2835002431B900059F28350005
:0C003200242156AFOA8302A807A830560C
:00000001FF
```

The first to fifth lines constitute an object code record and the last line constitutes an end record.

[Format of HEX-format Object Module File]

| | | | | | | | |
|---|----|------|----|-----------------------|----|----|----|
| : | 10 | 0002 | 00 | 2B41000BFC80F... 3A20 | EC | CR | LF |
| ① | ② | ③ | ④ | ⑤ | ⑥ | | |

- ① Record mark
Indicates the beginning of the record.
- ② Code count (2 digits)
Indicates the number of bytes of the codes to be stored in the data record. The maximum code count is 16 bytes. This field is fixed to "00H" with the end record.
- ③ Location address (4 digits)
Indicates the first address of the codes expressed by the data record.
This field is fixed to "0000H" with the end record.
- ④ Record type (2 digits)
00H ... Indicates the data record.
01H ... Indicates the end record.
- ⑤ Code (32 digits max.)
Object codes are indicated in units of one byte by dividing it into high-order 4 bits and low-order 4 bits. Code can be represented up to 16 bytes.
This field does not exist in the end record.
- ⑥ Check sum (2 digits)
A value obtained by subtracting the data from "Code count" field to "Code" field in sequence, from 0 is entered in this field.

(3) Symbol table file

The symbol table file output by the object converter can be input to a debugger.

The symbol table of the sample program is shown below.

```
#04
;FF      PUBLIC
010022CONVAH
010002START
;FF      SAMPM
;FF      SAMPS
<010035SASC
01003BSASC1
=
```

[Format of Symbol Table File]

| | | | | | | | | | | |
|---------------------------------------------|---|------------------|--------------|----|--------------------|--|----|----|--|--|
| Start of symbol table | # | 04 | CR | LF | | | | | | |
| Start of PUBLIC symbols | ; | FF | Blank | | PUBLIC | | CR | LF | | |
| (See Note 3)→ | | Symbol attribute | Symbol value | | PUBLIC symbol name | | CR | LF | | |
| | | ⋮ | ⋮ | | ⋮ | | ⋮ | ⋮ | | |
| Start of local symbols | ; | FF | Blank | | Module name 1 | | CR | LF | | |
| | < | Symbol attribute | Symbol value | | Local symbol name | | CR | LF | | |
| | | Symbol attribute | Symbol value | | Local symbol name | | CR | LF | | |
| | | ⋮ | ⋮ | | ⋮ | | ⋮ | ⋮ | | |
| | ; | FF | Blank | | Module name 2 | | CR | LF | | |
| (Repeat this format for each object module) | | ⋮ | ⋮ | | ⋮ | | ⋮ | ⋮ | | |
| End mark of symbol table | = | CR | LF | | | | | | | |

↑ ↑
See Note 1. See Note 2.

PUBLIC symbols

Local symbols per module

- NOTE: 1. This field is fixed to four characters.
 2. This field is fixed to six characters. If the symbol value is less than six characters, a blank is inserted in each unfilled character position.
 3. One of the following values is entered as the symbol attribute of each symbol in this field:

| Value | Symbol attribute |
|-------|--------------------------------------------------------------------------------------------------------------------|
| 00 | Constant defined with EQU directive |
| 01 | With 78K/0, 78K/I, or 78K/III: Label within CODE segment With 78K/VI: Label with CODE or DATA segment |
| 02 | With 78K/0, 78K/I, or 78K/III: Label within DATA segment With 78K/VI: Constant defined with EQU directive |
| 03 | With 78K/0, 78K/I, or 78K/III: Bit symbol With 78K/VI: Bit symbol with ABIT attribute |
| 05 | Bit symbol with SFBIT attribute |
| 06 | Bit symbol with RBBIT attribute |
| 07 | Bit symbol with RWBIT attribute |
| FF | Module name |

Note: Symbol values 05, 06, and 07 apply only to the 78K/VI series.

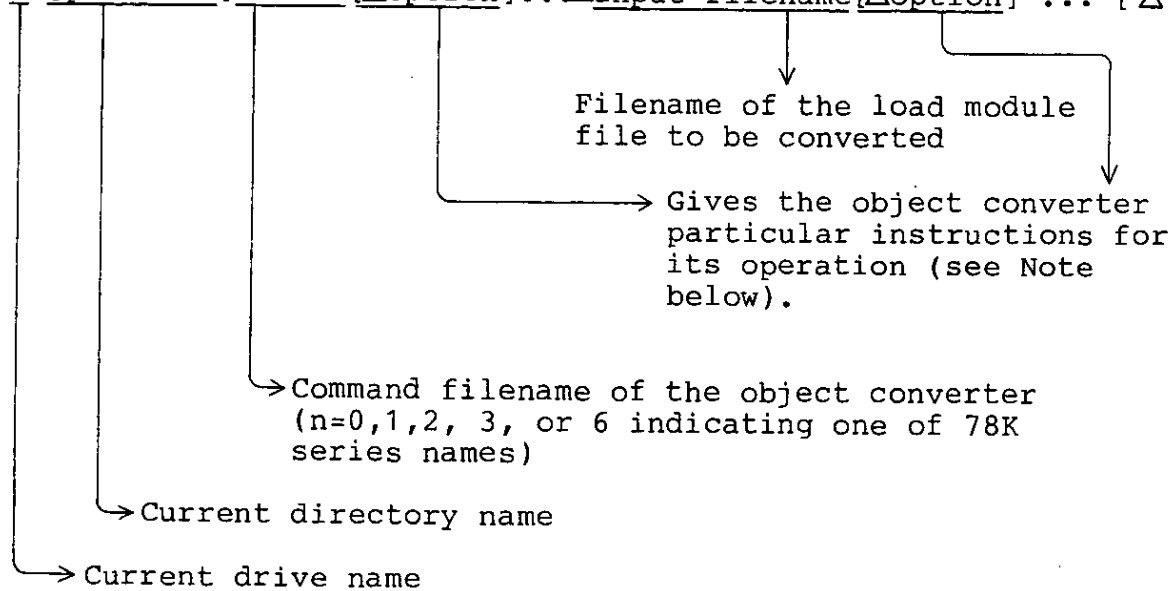
6.3 How to Start Up the Object Converter

6.3.1 Starting up the object converter

The object converter can be started up in the following two ways:

(1) Start-up with command line

X>[pathname]oc78kn[Δoption]...Δinput-filename[Δoption] ... [Δ]



Example: A>oc78k3 78k3.lnk -osample.hex

Note: If two or more object converter options are to be specified, each object converter option must be delimited with a space.
See Section 6.4 for details of each object converter option.

(2) Start-up with parameter file

A parameter file is used when all the required information for starting up the object converter cannot be specified in the start-up command line of the object converter or when the same object converter options are to be used repeatedly in each object conversion process.

When using this parameter file, the `-F` option must be specified in the start-up command line of the object converter to specify the use of the parameter file.

The object converter can be started up with a parameter file as follows:

```
X> oc78kn[Δload-moudle-file]Δ-f parameter-filename
```

→ File containing the information required to start up the object converter

→ Option specifying use of parameter file

- o A parameter file must be created with the editor.
- o The description format of parameters within the parameter file is as shown below.

`[[[Δ]option[Δoption].. [Δ]Δ] ...`

- o If the load module filename to be input is omitted in the start-up command line of the object converter, only one input load module filename may be described within the parameter file.
- o The input load module filename may be described either before or after an option.
- o In the parameter file, all the object converter options and output filename which should normally be specified in the start-up command line must be described.

Example: To create parameter file "78K3.POC" with the editor
o Contents of "78K3.POC"

```
;parameter file
78k3.lnk -osample.hex
-ssample.sym -r
```

o Start-up of object converter by specifying parameter
file "78K3.POC"

```
A>oc78k3 -f78k3.poc
```

6.3.2 Execution start and end messages

(1) Execution start message

When the object converter is started up, the following message is output to the console, indicating the start of the object conversion execution.

```
uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

(2) Execution end messages

o If the object conversion operation is completed normally, the object converter will output the following message to the console and return control to the OS.

```
Object Converter complete,      0 error(s) and      0 warning(s) found.
```

o If any errors are found as a result of an object conversion operation, the object converter will output the following message (the number of errors found) to the console and return control to the OS.

```
Object Converter complete,      3 error(s) and      0 warning(s) found.
```

- o If any fatal error is found during an object conversion operation, the object converter will output the following message, stop its processing, and return control to the OS.

Example 1:

```
A>oc78k3 sample.lnk
```

```
uCOM-78K/III Object Converter Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
A006 File not found 'SAMPLE.LNK'  
Program aborted.
```

In this example, the object conversion operation was discontinued by an abort error resulting from the specification of a load module file which does not exist in drive A.

Example 2:

```
A>oc78k3 78k3.lnk -a
```

```
uCOM-78K/III Object Converter Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
A018 Option is not recognized '-a'  
Program aborted.
```

In this example, the object conversion operation was discontinued by an abort error resulting from the input of an object converter option "-A (-a)" which is not recognized by the object converter.

If the object converter is aborted following the output of an error message, check the cause of the error by referring to Chapter 11, Error Messages and take corrective action(s) as required.

6.4 Object Converter Options

6.4.1 Types of object converter options

An object converter option gives the object converter particular instructions for its operation and is broadly divided into the following seven types:

Table 6-3. Types of Object Converter Options

| No. | Classification | Option | Functional description |
|-----|----------------------------------------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 | Options for HEX-format object module file output specification | -O | Specifies the output or non-output of a HEX-format object module file. |
| | | -NO | |
| 2 | Options for symbol table file output specification | -S | Specifies the output or non-output of a symbol table file. |
| | | -NS | |
| 3 | Options for object code output sequence specification | -R | Specifies the output of HEX-format object codes by sorting in their address sequence or as stored in the input load module file. |
| | | -NR | |
| 4 | Option for object code fill specification | -U | Specifies the output of the fill value specified by this option as object code to an address area to which no HEX-format object code is output. |
| 5 | Options for error list file output specification | -E | Specifies the output or non-output of an error list file. |
| | | -NE | |
| 6 | Option for parameter file specification | -F | Specifies the input of input filename and options from the file specified by this option. |
| 7 | Option for HELP message output specification | -- | Specifies the output of HELP message to the console. |

The above table merely introduces all the available object converter options. Each of these object converter options is detailed in Subsection 6.4.2 below. For quick reference, see Appendix C.3, List of Object Converter Options in which the description format of each option and the relationship between one option and the other are also outlined.

6.4.2 Description of each object converter option

A detailed description of each object converter option is provided in this subsection.

-O/-NO

HEX-format object module file
output specification

(1) Options for HEX-format output module file output specification
(-O/-NO)

| | |
|---------------------|-----------------------------------------------------------------------|
| Description format: | -O (output-filename) |
| | or |
| | -NO |
| Default assumption: | -O input-filename.HEX (file type for extension space: .H1 to .H15) |

Function

- o The -O option specifies the output of a HEX-format object module file. It also specifies the output destination or output filename of the HEX-format object module file to be output by the object converter.
- o The -NO option specifies the non-output of a HEX-format object module file.

Use

- o Use the -O option if you want to change the output destination or output filename of a HEX-format object module file.
- o If the object conversion of a load module file is to be performed only to output a symbol table file, use the -NO option. This will reduce the object conversion time.

Explanation

- o As an output filename, only a disk type filename must be specified. If any device type filename is specified, an abort error will result.
- o If an output filename is omitted from the -O option specification, "input filename.HEX" (file type: ".H1" to ".H15" for a HEX file created for each extension space) is assumed as the output filename and a HEX-format object module file under that name will be output to the current directory.

-O/-NO

HEX-format object module file
output specification

- o If only a pathname is specified as the output filename, a HEX-format object module file named "input-filename.HEX" (file type: ".H1" to ".H15" for a HEX file created for each extension space) will be output to the specified path.
- o If the -O and -NO options are specified at the same time, whichever you specified later will take precedence over the other.
- o When raw data have been output to segments located in extension spaces, the object converter creates a separate HEX-format object module file for each extension space. The file type of a HEX-format object module file for each extension space is as shown below.

| File | Regular space | Extension space | | | | | | | |
|------|---------------|-----------------|-----|-----|-----|-----|------|------|------|
| | REGULAR | EX1 | EX2 | EX3 | EX4 | ... | EX13 | EX14 | EX15 |
| HEX | .HEX | .H1 | .H2 | .H3 | .H4 | ... | .H13 | .H14 | .H15 |

Application Examples

Example 1: To convert load module file "78K3.LNK" with -O option specified to output a HEX file named "SAMPLE.HEX"

```
A>oc78k3 78k3.lnk -osample.hex
```

```
uCOM-78K/III Object Converter Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Object Converter complete,      0 error(s) and      0 warning(s) found.
```

-O/-NO

HEX-format object module file
output specification

Example 2: To convert the load module file with both -NO
and -O options specified

A>oc78k3 78k3.lnk -no -o

uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Object Converter complete, 0 error(s) and 0 warning(s) found.

In this case, -NO option becomes invalid and -O
option is accepted as valid.

-S/-NS

Symbol table file output
specification

(2) Options for symbol table file output specification (-S/-NS)

| | |
|---------------------|-----------------------------------------------------------------------|
| Description format: | -S [output-filename] |
| | or |
| | -NS |
| Default assumption: | -S input-filename.SYM (file type for extension space: .S1 to .S15) |

Function

- o The -S option specifies the output of a symbol table file. It also specifies the output destination or output filename of the symbol table file to be output by the object converter.
- o The -NS option specifies the non-output of a symbol table file.

Use

- o Use the -S option if you want to change the output destination or output filename of a symbol table file.
- o If the object conversion of a load module module file is to be performed only to output a HEX-format object module file, use the -NS option. This will reduce the object conversion time.

Explanation

- o As an output filename, only a disk type filename must be specified. If any device type filename is specified, an abort error will result.
- o If an output filename is omitted from the -S option specification, "input filename.SYM" (file type: ".S1" to ".S15" for a symbol file created for each extension space) is assumed as the output filename and a symbol table file under that name will be output to the current directory.

-S/-NS

Symbol table file output
specification

- o If only a pathname is specified as the output filename, a symbol table file named "input-filename.SYM" (file type: ".S1" to ".S15" for a symbol file created for each extension space) will be output to the specified path.
- o If the -S and -NS options are specified at the same time, whichever you specified later will take precedence over the other.
- o When symbols which have the ADDRESS attribute or BIT attribute have been defined in segments located in extension spaces, the object converter creates a separate symbol table file for each extension space. The object converter outputs symbols which have the NUMBER attribute to a symbol table file for the regular space.

The file type of a symbol table file for each extension space is as shown below.

| File | Regular space | Extension space | | | | | | | |
|--------|---------------|-----------------|-----|-----|-----|-----|------|------|------|
| | REGULAR | EX1 | EX2 | EX3 | EX4 | ... | EX13 | EX14 | EX15 |
| Symbol | .SYM | .S1 | .S2 | .S3 | .S4 | ... | .S13 | .S14 | .S15 |

Application Examples

Example 1: To convert load module file "78K3.LNK" with -S option specified to output a symbol table file named "SAMPLE.SYM"

```
A>oc78k3 78k3.lnk -ssample.sym
```

```
uCOM-78K/III Object Converter Vx.xx [xx xxx xx]  
Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Object Converter complete,      0 error(s) and      0 warning(s) found.
```

-S/-NS

Symbol table file output
specification

Example 2: To convert the load module file with both -NS
and -S options specified

A>oc78k3 78k3.lnk -ns -s

uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Object Converter complete. 0 error(s) and 0 warning(s) found.

In this case, -NS option becomes invalid and -S
option is accepted as valid.

-R/-NR

Object code output sequence
specification

(3) Options for object code output sequence specification (-R/-NR)

| | |
|---------------------|-----|
| Description format: | -R |
| | or |
| | -NR |
| Default assumption: | -NR |

Function

- o The -R option specifies the output of HEX-format object codes by sorting them in the order of their addresses.
- o The -NR option specifies the output of HEX-format object codes in the order of their storage in the input load module file.

Use

Use the -R option if you want to have HEX-format object codes sorted in the order of their addresses.

Explanation

- o If the -R and -NR options are specified at the same time, whichever you specified later will take precedence over the other.
- o If the -R or -NR option is specified together with the -NO option, the -R or -NR option will become invalid.

-R/-NR

**Object code output sequence
specification**

Application Examples

Example 1: To convert load module file "78K3.LNK" with -R
option specified to sort HEX-format object codes in
address sequence

A>oc78k3 78k3.lnk -r

uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Object Converter complete, 0 error(s) and 0 warning(s) found.

(4) Option for object code fill specification (-U)

Description format: -U fill-value[, [start], size]

Default assumption: None

Function

The -U option tells the object converter to output the fill value specified by this option as object codes to an address area to which no HEX-format object code has been output.

Use

Unwanted codes may be written into an address area to which no HEX-format object code has been output. If the program accidentally accesses any address in such address area, proper program operation cannot be expected. To avoid this, the -U option can be used to write some code beforehand into an address area to which no HEX-format object code has been output.

Explanation

- o A fill value may be specified in hexadecimal or decimal numbers but must not exceed the following value range:

$0H \leq \text{fill value} \leq 0FFH$

If a numerical value beyond the above range or other than a numerical value is specified, an abort error will result.

- o As the parameter "start", the start address of an address area into which code is to be filled may be specified in hexadecimal or decimal numbers but must not exceed the following value range:

$0H \leq \text{start} \leq 0FF00H$

If a numerical value beyond the above range or other than a numerical value is specified, an abort error will result.

- o If "start" is omitted from the -U option specification, address 0H is assumed to have been specified.
- o As the parameter "size", the size of the address area into which code is to be filled may be specified in hexadecimal or decimal numbers but must not exceed the following value range:
$$0H \leq \text{size} \leq 0FF00H$$

If a numerical value beyond the above range or other than a numerical value is specified, an abort error will result.
- o If the parameter "start" is specified, the parameter "size" cannot be omitted.
- o If both parameters "start" and "size" are omitted, the object converter will process the input load module file as follows:
 - (a) If the target device subject to assembly has an internal (on-chip) ROM, the object converter will assume that the internal ROM area has been specified.
 - (b) If the target device subject to assembly has no internal ROM, the object converter will output an error message and stops its processing.
- o If the -U option is specified two or more times in the same command line, the last input -U option will take precedence over the others.
- o The "start" and "size" specification formats with the -U option and the interpretation of each format by the object converter are as summarized below.
 - (a) -U fill-value
 - o Internal ROM area if the target device has an internal ROM
 - o Abort error if the target device has no internal ROM
 - (b) -U fill-value, size
 - o Address area from address 0H to address specified by "size"
 - (c) -U fill-value, start, size
 - o Address area from "start" address to address specified by "size"

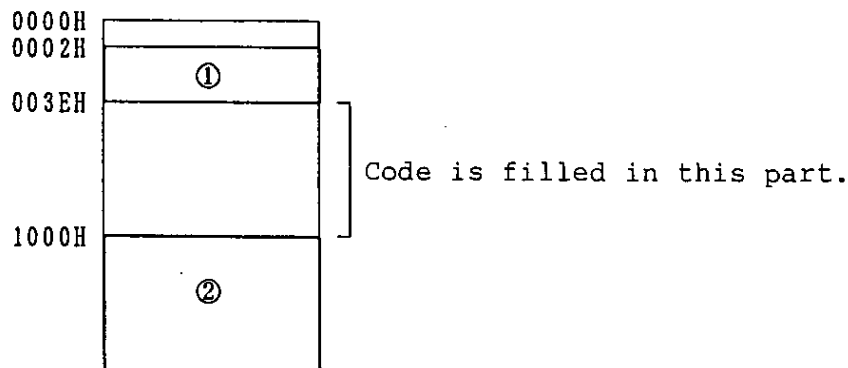
Application Examples

Example 1: To fill a code into an address area to which no HEX-format object code has been output

- o Assume that the following HEX-format object module file exists. In this case, no code has been written into an address area from 003EH to 0FFFH.

```
:020000000200FC
:100002002B41000BFC80FE2B40000944F7083A20EC
:100012001A6720FE2822006521FED350D25014FE1A
:10002200B900059F2835002431B900059F28350005
:0C003200242156AF0A8302A807A830560C
:101000003B5D040026A3 ...
:1010100024A5F622B667 ...
...
:00000001FF
```

①
②



- o Specify -U option to fill code "00H" in address area 003EH to 0FFFH

A>oc78k3 78k3.lnk -u00h,003eh,0fc2h

uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Object Converter complete. 0 error(s) and 0 warning(s) found.

(5) Options for error list file output specification (-E/-NE)

Description format: -E [output-filename]

or

-NE

Default assumption: -NE

Function

- o The -E option specifies the output of an error list file. It also specifies the output destination or output filename of the error list file to be output by the object converter.
- o The -NE option specifies the non-output of an error list file.

Use

- o Use the -E option if you want to save error messages to a file.
- o Also use the -E option if you want to change the output destination or output filename of an error list file.

Explanation

- o An output filename can be specified with either a disk type filename or a device type filename. However, if a device type filename "CLOCK" is specified as an output filename, an abort error will result.
- o If an output filename is omitted from the -E option specification, "input filename.EOC" is assumed as the output error list filename.
- o If a drive name is omitted from the -E option specification, the error list file will be output to the current drive.
- o If the -E and -NE options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To convert load module file "78K3.LNK" with -E option specified to create an error list file named "78K3.EOC"

A>oc78k3 78k3.lnk -e78k3.eoc

uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

F100 Undefined symbol : CONVAH
Object Converter complete, 1 error(s) and 0 warning(s) found.

o When error list file "78K3.EOC" is referenced, the output error list file will look like this.

F100 Undefined symbol : CONVAH

(6) Option for parameter file specification (-F)

Description format: -F filename

Default assumption: Options and input filenames can be input only from start-up command line.

Function

The -F option tells the object converter that options and input filename(s) will be input from the file specified by this option.

Use

- o Use the -F option if all the required parameters for starting up the object converter cannot be specified in the start-up command line.
- o If you have a set of object converter options which you must specify repeatedly at each object conversion operation, describe these object converter options in a parameter file and then specify the -F option in the start-up command line.

Explanation

- o A filename can be specified with only a disk type filename. If any device type filename is specified with this option, an abort error will result.
- o If a filename is omitted from the -F option specification, an abort error will also result.
- o Nesting of parameter files is not allowed. If the -F option is specified in a parameter file, an abort error will result.
- o The number of characters that can be described in a parameter file is not limited.
- o A blank character, Tab character, or "□" must be used as a delimiter between options or input filenames.

- o The options and input filenames described in the parameter file will be expanded to the location on the command line where the parameter file (-F option) has been specified.
- o The object converter will process these expanded options in the order from the last input option.
- o All characters described between ";" or "#" and "␣" or an EOF code will be interpreted as a comment statement.
- o If two or more -F options are specified at the same time, an abort error will result.

Application Examples

Example 1: To convert load module file "78K3.LNK" with -F option specified

- o Contents of parameter file "78K3.POC"

```
;parameter file
78k3.lnk -osample.hex
-ssample.sym -r
```

- o Enter -F option in the command line as follows:

```
A>oc78k3 -f78k3.poc
```

```
uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Object Converter complete,      0 error(s) and      0 warning(s) found.
```

--

HELP message display specification

(7) Option for HELP message display specification (--)

Description format: --

Default assumption: No HELP message is displayed.

Function

The -- option tells the object converter to display the HELP message on the console.

Use

The HELP message is a list of all object converter options and their functional descriptions. Use the -- option if you want to refer to this message when executing the object converter.

Explanation

If the -- option is specified, all the other object converter options specified at the same time will become invalid.

Application Examples

Example 1: Input -- option as shown below and the HELP message will be displayed on the screen.

A>oc78k3 --

uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

usage : OC78K3 [option[...]] input-file [option[...]]

The option is as follows ([] means ommisible).

-Ffile :Input option or input-file name from specified file.
-O[file]/-NO :Create HEX module file [with specified name] / Not.
-S[file]/-NS :Create symbol table file [with specified name] / Not.
-R/-NR :Sort HEX object by address / Not.
-Uvalue[, [start], size] :Fill up HEX object with specified value.
-- :Show this message.

DEFAULT ASSIGNMENT: -O -S -NR

CHAPTER 7. LIBRARIAN

The librarian (or library program) performs the editing of object module files or library files for the 78K series in units of modules. The librarian also outputs a list file.

If any error exists during a library processing, the librarian outputs an error message to the console to clearly indicate the cause of the error.

7.1 Input/Output Files of Librarian

The files listed in Table 7-1 below are input and output to and from the librarian.

Table 7-1. I/O Files of Librarian

| Type | Name and description of file | Default file type |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|
| Input file | <u>Subcommand file</u> A file containing the subcommands and parameters of an executable program. This file must be created by the user. | None |
| Output file | <u>List file</u> A file containing the result of information output from a library file. | .LST |
| I/O files | <u>Object module file</u> An object module file output by the assembler or compiler. | .REL |
| | <u>Library file</u> A library file output by the librarian. This file is input to the librarian for updating the file contents. | .LIB |
| | <u>Temporary file</u> A file which is automatically generated by the librarian as a work file. This file will be erased on completion of the library process. | LB78Kn.\$x x=1 to 6 (fixed to this name) |

(n=0, 1, 2, 3, or 6)

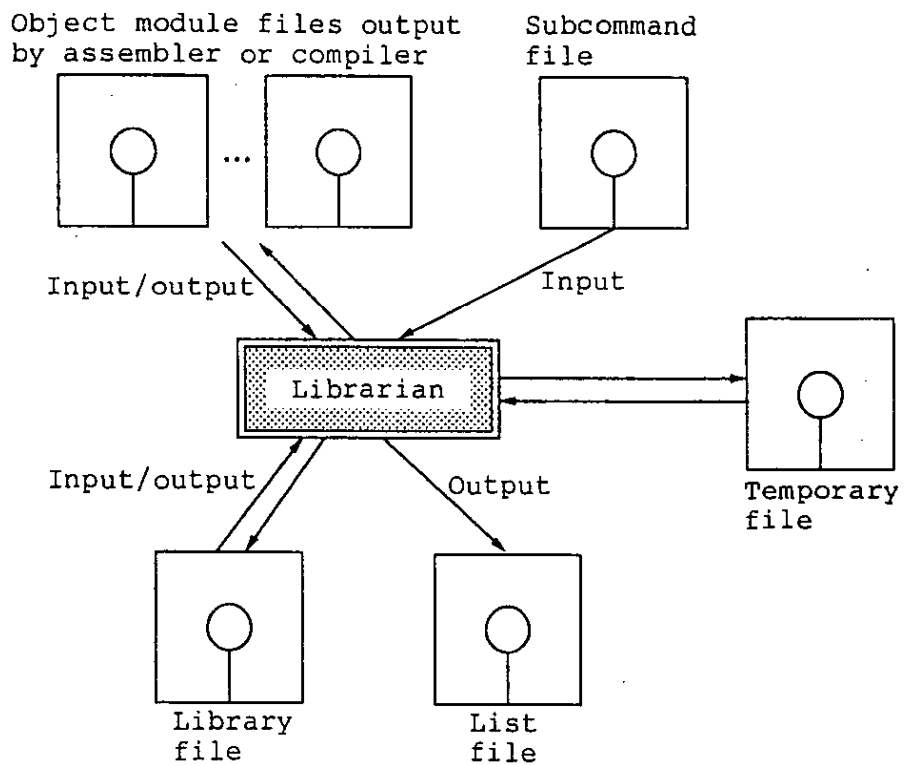


Fig. 7-1. I/O Files of Librarian

7.2 Librarian Functions

(1) Storing modules in library (Creating a library file)

Both the assembler and the linker create one file per output module. The more the object modules are produced, the more the output files are generated. Thus, the function to collect two or more files into a single file is provided. This collection of modules is called a library and a file used to maintain and make available the modules is called a library file.

A library file can be input to the linker. Therefore, if a library file of commonly used modules is created when developing a program using the modular programming technique, files can be efficiently managed and operated.

(2) Editing library files

The librarian has the following editing functions for library files:

- o Adding module(s) to a library file
- o Deleting module(s) from a library file
- o Replacing module(s) in a library file
- o Selecting and copying module(s) from a library file

(These functions are discussed in detail in Section 7.5, Subcommands.)

(3) Listing the library file information

The librarian has the function to edit and output the following items of information stored in a library file:

- o Module name
- o Program created
- o Date of registration
- o Date of file updating
- o PUBLIC symbol information

Note: The librarian functions (2) and (3) above are offered in the librarian as subcommands. The librarian performs its processing by interpreting and executing each subcommand. See Section 7.5, Subcommands for the operation of each subcommand.

(4) Library file creation procedure

Fig. 7-2 illustrates a general procedure for creating a library file.

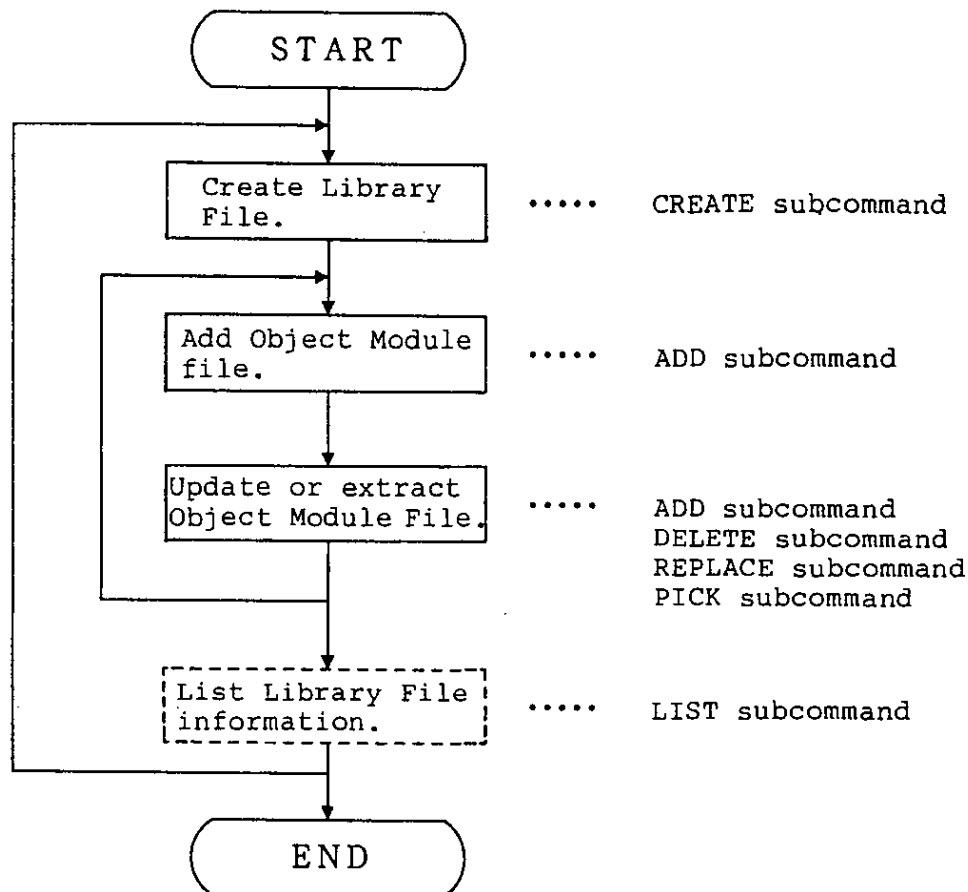


Fig. 7-2. Procedure for Creating a Library File

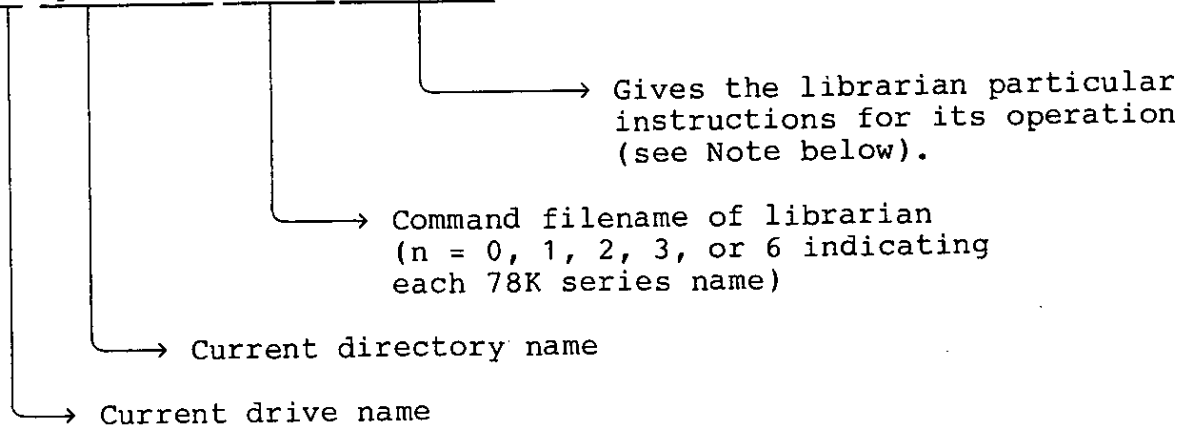
7.3 How to Start Up the Librarian

7.3.1 Starting up the librarian

The librarian can be started up (invoked) in either of the following two ways:

(1) Start-up with the start-up command line of the librarian

X>[pathname]lb78kn[Δoption]...



Example: A>lb78k3 -ll20 -lw80

NOTE: If two or more librarian options are to be specified, each librarian option must be delimited with a space. See Section 7.4, Librarian Options for details of each option.

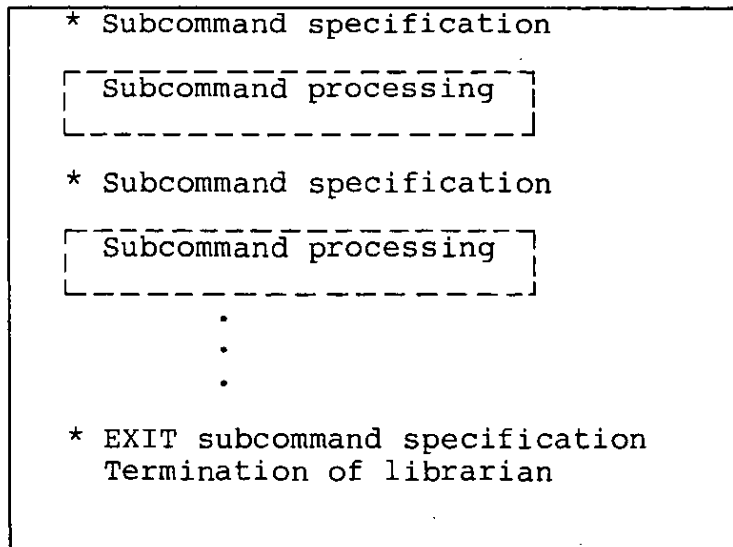
When the librarian is started up, the following message is output to the console:

```
uCOM-78K/III Librarian Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
*
```

Specify a desired librarian subcommand following the prompt
"*.".

```
*create 78k3.lib
*add 78k3.lib 78k3main.rel 78k3sub.rel
*exit
```

On input of the subcommand, the librarian starts processing the subcommand. On completion of the subcommand processing, the prompt "*" appears again, indicating the standby state for your input of the next subcommand. This process will be repeated until you input the EXIT subcommand to terminate the librarian.



Up to 128 characters may be input per line. A continuation line may be specified if all the required operand information cannot be input on a single line. Up to 15 continuation lines are allowed per subcommand.

(2) Start-up with subcommand file

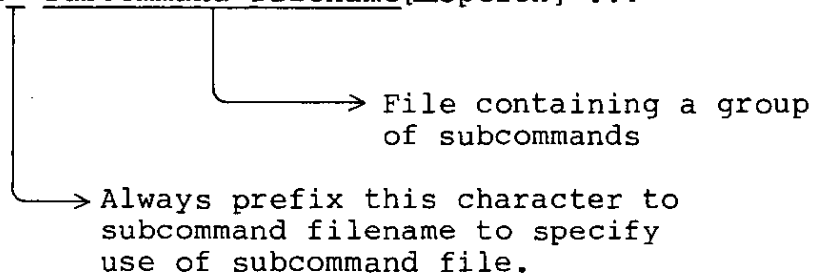
A subcommand file refers to a file in which commands to the librarian are stored. If you do not use a subcommand file when starting up the librarian, you must input two or more subcommands individually following the output of the prompt "*". However, by creating a subcommand file, these subcommands can be processed all at once by the librarian.

You may have to specify the same subcommands repeatedly each time you create a library file. In such a case, use a subcommand file to create a library file.

When using this subcommand file, "<" must be described before the subcommand filename in the start-up command line of the librarian to specify the use of the subcommand file.

The librarian can be started up with a subcommand file as follows:

X>lb78kn Δ< subcommand-filename[Δoption] ...



- o A subcommand file must be created with the editor.
- o The description format of parameters within the subcommand file is as shown below.

| | |
|-----------------|---------------------|
| subcommand-name | operand-information |
| subcommand-name | operand-information |
| | : |
| | : |
| EXIT | |

- o If two or more lines are required for the specification of a subcommand, describe "&" at the end of each line to indicate a continuation line.

- o All characters between ";" (colon) and the end of the line are not interpreted as a librarian subcommand but handled as a comment statement.
- o If the last subcommand in the subcommand file is not the EXIT subcommand, the librarian will assume that the EXIT subcommand has been described in the subcommand file.
- o The librarian processes each subcommand by reading it from the subcommand file. On completion of all the subcommands in the subcommand file, the librarian is terminated.

Example: To create subcommand file "78K3.SLB" with the editor

- o Contents of 78K3.SLB

```

;
;library creation command
;
create 78k3.lib
;
add 78k3.lib 78k3main.rel &
78k3sub.rel
;
exit

```

- o Start-up of librarian using subcommand file "78K3.SLB"

A><lb78k3 <78k3.slb

7.3.2 Execution start and end messages

(1) Execution start message

When the librarian is started up, the following message is output to the console, indicating the start of the librarian execution.

```
uCOM-78K/III Librarian Vx.xx [xx xxx xx]  
  Copyright (C) Corporation xxxx USxxxxxxxxxx  
*
```

(2) Execution end messages

- o The librarian does not output an execution end message. The user must input the EXIT subcommand on completion of each processing. On input of this subcommand, the librarian returns control to the OS.

```
*create 78k3.lib  
*add 78k3.lib 78k3main.rel 78k3sub.rel  
*exit
```

- o If any fatal error is found during a librarian operation, the librarian will output the following message to the console, stop its processing, and return control to the OS.

Example 1:

```
A>lb78k3 78k3.slb
```

```
uCOM-78K/III Librarian Vx.xx [xx xxx xx]  
  Copyright (C) Corporation xxxx USxxxxxxxxxx  
A003 Unrecognized string 'LIB.JOB'  
Usage: LB78K3 [options]
```

In this example, the librarian operation was discontinued by an abort error resulting from the omission of "<" before the subcommand filename specification.

Example 2:

```
A>lb78k3 -a
uCOM-78K/III Librarian Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
A018 Option is not recognized '-a'
Usage: LB78K3 [options]
```

In this example, the librarian operation was discontinued by an abort error resulting from the input of a librarian option "-A (-a)" which is not recognized by the librarian.

If the librarian is aborted following the output of an error message, check the cause of the error by referring to Chapter 11, Error Messages and take corrective action(s) as required.

7.4 Librarian Options

7.4.1 Types of librarian options

A librarian option gives the librarian particular instructions for its operation and is divided into the following three types:

Table 7-2. Types of Librarian Options

| No. | Classification | Option | Functional description |
|-----|-------------------------------------------------------|--------|--------------------------------------------------------------------------------------------|
| 1 | Options for list file format specification | -LW | Specifies the number of print columns per line of a list file. |
| | | -LL | Specifies the number of print lines per page of a list file. |
| | | -LF | Specifies the addition or non-addition of a form-feed (FF) code to the end of a list file. |
| | | -NLF | |
| 2 | Option for temporary file creation path specification | -T | Specifies the creation of a temporary file on the path specified by this option. |
| 3 | Option for HELP message output specification | -- | Specifies the output of HELP message to the console. |

The above table merely introduces all the available librarian options. Each of these librarian options is detailed in Subsection 7.4.2 below. For quick reference, see Appendix C.4, List of Librarian Options in which the description format of each option and the relationship between one option and the other are also outlined.

7.4.2 Description of each librarian option

A detailed description of each librarian option is provided in this subsection.

- (1) Options for list file format specification (-LW, -LL, -LF/-NLF)

- (a) Option for page width specification (-LW)

Description format: -LW [No. of columns per line]

Default assumption: -LW132 (-LW80 with console output)

Function

- o The -LW option specifies the number of print columns per line of a list file.

Use

Use the -LW option if you want to change the number of print columns per line of a list file.

Explanation

- o The number of print columns per line to be specified with the -LW option must be within the following value range excluding the terminator (CR or LF):
$$72 \leq \text{No. of print columns per line} \leq 132$$

(For output to the console, the maximum value becomes 80 columns.)

If any value beyond this range or other than a value is specified with this option, an abort error will result.
- o If the number of columns per line is omitted, a value of 132 is assumed to have been specified. However, if the output destination of a list file is the console, a value of 80 is assumed.
- o If the LIST subcommand is not specified, the -LW option specification will be ignored and thus will become invalid.
- o If two or more -LW options are specified in the same command line, the last specified -LW option will take precedence over the other -LW options.

Application Examples

Example 1: To execute the librarian with -LW option specified
to set 80 columns as No. of columns per line of a
list file

A>lb78k3 -lw80

uCOM-78K/III Librarian Vx.xx [xx xxx xx]

Copyright (C) Corporation xxxx USxxxxxxxxxx

*create 78k3.lib 78k3sub.rel

← Subcommand

*list 78k3.lib

← Subcommand

(b) Option for page length specification (-LL)

Description format: -LL [No. of lines per page]

Default assumption: -LL66 (No form-feed operation
with console output)

Function

- o The -LL option specifies the number of lines per page of a list file.

Use

Use the -LL option if you want to change the number of lines per page of a list file.

Explanation

- o The number of print lines per page to be specified with the -LL option must be within the following value range:
 $20 \leq \text{No. of print lines per page} \leq 32767$
If any value beyond this range or other than a value is specified with this option, an abort error will result.
- o If the number of print lines per page is omitted, a value of 66 is assumed to have been specified.
- o If "0" is specified as the number of print lines per page, no form-feed operation (page ejection) will be carried out.
- o If the LIST subcommand is not specified, the -LL option specification will be ignored and thus will become invalid.
- o If two or more -LL options are specified in the same command line, the last specified -LL option will take precedence over the other -LL options.

Application Examples

Example 1: To execute the librarian with -LL option specified
to set 20 lines as No. of lines per page of a list
file

A>lb78k3 -1120

uCOM-78K/III Librarian Vx.xx [xx xxx xx]

Copyright (C) Corporation xxxx USxxxxxxxxxx

*create 78k3.lib 78k3sub.rel

← Subcommand

*list 78k3.lib

← Subcommand

(c) Options for form-feed code addition specification
(-LF/-NLF)

| | |
|---------------------|------|
| Description format: | -LF |
| | or |
| | -NLF |
| Default assumption: | -NLF |

Function

- o The -LF option specifies the addition of a form-feed (FF) code to the end of a list file.
- o The -NLF option specifies the non-addition of a form-feed (FF) code to the end of a list file.

Use

If you want to have a new page after printing the contents of a list file, add a form-feed (FF) code to the end of the list file by specifying the -LF option.

Explanation

- o If the LIST subcommand is not specified, the -LF option specification will be ignored and thus will become invalid.
- o If the -LF and -NLF options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To execute the librarian with -LF option
specified to add an FF code to the end of a
list file

A>lb78k3 -lf

uCOM-78K/III Librarian Vx.xx [xx xxx xx]

Copyright (C) Corporation xxxx USxxxxxxxxxx

*create 78k3.lib 78k3sub.rel

← Subcommand

*list 78k3.lib

← Subcommand

-T

Temporary file creation
path specification

(2) Option for temporary file creation path specification (-T)

Description format: -T pathname

Default assumption: Temporary file is created on the path
specified by environment variable TMP
or on the current path if no path is
specified by TMP

Function

The -T option tells the librarian to create a temporary file on the path specified by this option.

Use

The -T option can be used to specify where a temporary file is to be created.

Explanation

- o Other than a path cannot be specified as a pathname. If a pathname is omitted from the -T option specification, an abort error will result.
- o If a previously created temporary file exists, the librarian will create a temporary file by overwriting the file unless it is write-protected.
- o If the required memory space for temporary file creation is available, the librarian will create a temporary file in memory. If the memory space is exhausted during the temporary file creation, the librarian will save the temporary file contents in memory to another disk and subsequent accessing to the temporary file will be made to that disk.

- o The temporary file created for a library file process by the librarian will be erased on completion of the library file process. The temporary file will also be erased when the library file process is discontinued by CTRL-C key input.
- o A path for temporary file creation is determined in the following order:
 - ① Path specified by the -T option
 - ② Path specified by environment variable TMP (when the -T option is omitted)
 - ③ Current path (when no path is specified by environment variable TMP)

If a temporary file cannot be created on the path specified by ① or ②, an abort error will result.

Application Examples

Example 1: To execute the librarian with -T option specified to create a temporary file on directory TMP

A>lb78k3 -tYtmp

uCOM-78K/III Librarian Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx
*

(3) Option for HELP message display specification (--)

Description format: --

Default assumption: No HELP message is displayed.

Function

The -- option tells the librarian to display the HELP message on the console.

Use

The HELP message is a list of all librarian subcommands and their functional descriptions. Use the -- option if you want to refer to this message when executing the librarian.

Explanation

If the -- option is specified, all the other librarian options specified at the same time will become invalid.

Application Examples

Example 1: Input -- option as shown below and the HELP message will be displayed on the screen.

A>lb78k3 --

uCOM-78K/III Librarian Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Subcommands : create, add, delete, replace, pick, list, help, exit

Usage : subcommand[option] masterLBF[option] transaction[option]

transaction ::= OMFname
LBFname[(modulename[...])]

<create > : create masterLBF[transaction]

<add > : add masterLBF transaction

<delete > : delete masterLBF(modulename[...])

<replace> : replace masterLBF transaction

<pick > : pick masterLBF (modulename[...])

<list > : list[option] masterLBF[(modulename[...])]

option : -P = output public symbol

-NP = no output public symbol

-O filename = specify output file name

<help > : help

<exit > : exit

7.5 Subcommands

7.5.1 Types of subcommands

A subcommand gives the librarian a detailed instruction for its operation. The librarian has the following eight subcommands:

Table 7-3. Subcommands

| Item No. | Subcommand name | Abbreviated format | Functional description |
|----------|-----------------|--------------------|--------------------------------------------------------------|
| 1 | CREATE | C | Creates a new library file. |
| 2 | ADD | A | Adds a module to a library file. |
| 3 | DELETE | D | Deletes a module from a library file. |
| 4 | REPLACE | R | Replaces a module in a library with another module. |
| 5 | PICK | P | Selects and copies a module from a library file. |
| 6 | LIST | L | Outputs information on the modules stored in a library file. |
| 7 | HELP | H | Outputs HELP message to the console. |
| 8 | EXIT | E | Terminates the librarian. |

The above table merely introduces to you all the subcommands of the librarian. Each of these subcommands is detailed in Subsection 7.5.2 below. For quick reference, see Appendix D, List of Subcommands, in which the description format of each subcommand has been described.

7.5.2 Description of each subcommand

This subsection details each librarian subcommand.

Note: General format of a subcommand file is as follows:

```
*subcommand[Δoption]Δ①library-filename[Δoption] transaction[Δoption]②
```

① The library filename specified immediately before this subcommand specification may be substituted with ".".

② transaction = Δobject-module-filename
 Δlibrary-filename[▲(▲module-name[▲, ...])]

(1) CREATE subcommand

Description format: CREATE library-filename[transaction]

Abbreviated format: C

Function

The CREATE subcommand creates a new library file.

Explanation

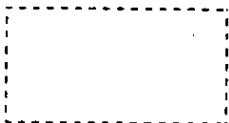
- o The size of the newly created library file becomes 0.
- o If the second operand "transaction" is specified, the object module file(s) specified by the operand will be added to (stored in) the library file upon its creation.
- o If the file specified by the first operand "library-filename" already exists, the librarian will overwrite the existing library file.
- o In the "transaction" specification, neither an object module file which has the same PUBLIC symbol as that of a module in a library file nor an object module which has the same name as a module in a library file can be added to the library file.
- o If any error occurs, the librarian will stop its processing and will not create a library file.

Application Examples

Example 1: To create a new library file named "78K3.LIB"

*create 78k3.lib

<Before creation>



<After creation>

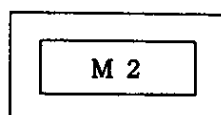
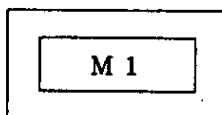


7 8 K 3. L I B

Example 2: To add modules M1 and M2 to a new library file
upon its creation

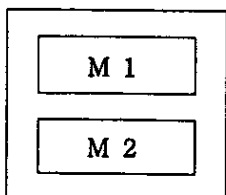
*create 78k3.lib m1.rel m2.rel

<Before creation>



<After creation>

7 8 K 3. L I B



ADD

Adding module(s) to an existing
library file

(2) ADD subcommand

Description format: ADD Δ library-filename[Δ transaction]

Abbreviated format: A

Function

The ADD subcommand adds module(s) to an existing library file.

Explanation

- o The existing library file to which module(s) are to be added may or may not contain any module.
- o If a module which has the same module name as that specified by the second operand "transaction" exists in the existing library file specified by the first operand, an error will result.
- o If the module specified for addition has the same PUBLIC symbol as that of the module in the existing library file, an error will result.

Application Examples

Example 1: To add modules M1 and M2 to library file
"78K3.LIB"

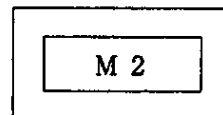
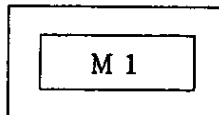
*add 78k3lib m1.rel m2.rel

ADD

Adding module(s) to an existing
library file

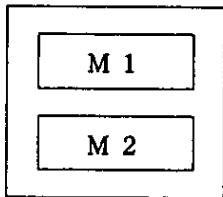
<Before addition>

7 8 K 3 . L I B



<After addition>

7 8 K 3 . L I B

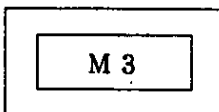
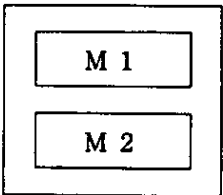


Example 2: To add module M3 to library file "78K3.LIB"

*add 78k3.lib m3.rel

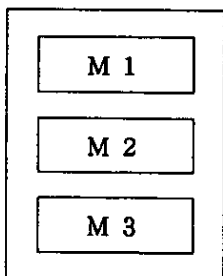
<Before addition>

7 8 K 3 . L I B



<After addition>

7 8 K 3 . L I B

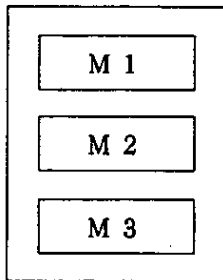


DELETE

Deleting module(s) from an existing
library file

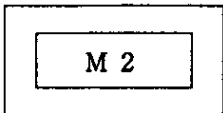
<Before deletion>

7 8 K 3. L I B



<After deletion>

7 8 K 3. L I B



REPLACE

Replacing a module in an existing
library file

(4) REPLACE subcommand

Description format: REPLACE Δ library-filename Δ transaction

Abbreviated format: R

Function

The REPLACE subcommand replaces a module in an existing library file with a module in another object module file.

Explanation

- o If the same named module as that specified for replacement by the second operand does not exist in the library file specified by the first operand, an error will result.
- o If the module specified for replacement has the same PUBLIC symbol as that of the module in the existing library file, an error will result.
- o The object module filename specified for replacement must be the same filename as that when it was added to the library file.
- o If any error occurs, the librarian will stop the Replace operation but the contents of the existing library file will remain unchanged.

REPLACE

Replacing a module in an existing library file

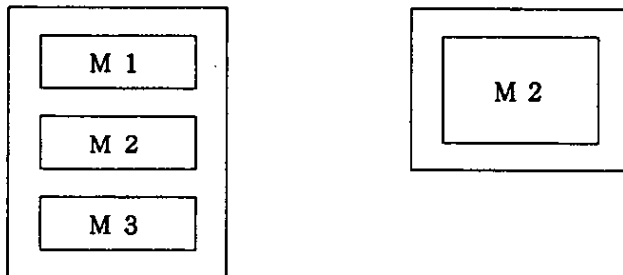
Application Examples

Example 1: To replace module M2 in library file "78K3.LIB" with another (new) module M2

*replace 78k3.lib m2.rel

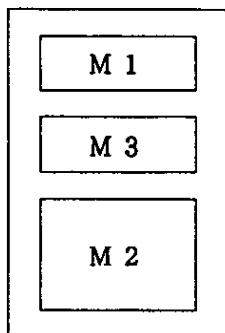
<Before replacement>

7 8 K 3 . L I B



<After replacement>

7 8 K 3 . L I B



Because the librarian first deletes the module M2 in the library file and then adds the new module M2 to the library file, the sequence of the new module M2 becomes the last in the library file.

PICK

Selecting & copying module(s) from
an existing library file

(5) PICK subcommand

| |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Description format: PICK Δlibrary-filename Δ (Δmodule-name [Δ, ...] Δ)</p> <p>Abbreviated format: P</p> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Function

The PICK subcommand picks up (selects) and copies specified module(s) from an existing library file.

Explanation

- o The module picked up from the library file will become an object module file which has the same name as that when it was added to the library file.
- o If the module name specified for selection by the second operand does not exist in the library file specified by the first operand, an error will result.
- o If any error occurs, the librarian will stop the Pick (select & copy) operation. However, if an error occurs during the select and copy process of two or more modules, the module(s) selected and copied before the occurrence of the error will become valid and thus will be saved on the disk.

PICK

Selecting & copying module(s) from
an existing library file

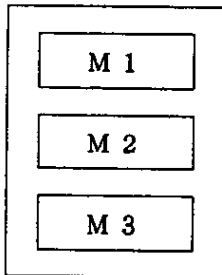
Application Examples

Example 1: To select and copy module M2 from library file
"78K3.LIB"

*pick 78k3.lib m2.rel

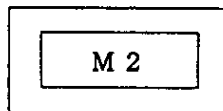
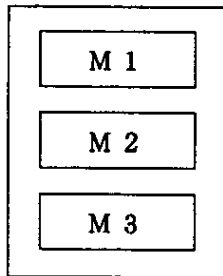
<Before selection>

7 8 K 3 . L I B



<After selection>

7 8 K 3 . L I B



(6) LIST subcommand

Description format: LIST [Δoption]Δlibrary-filename
[Δ(Δmodule-name [Δ, ...]Δ)]

option ::= -PUBLIC/-NOPUBLIC
-O Δfilename

Abbreviated format: L

Function

The LIST subcommand outputs information on the specified module(s) within a specified library file to a list file.

Explanation

- o Two or more options may be specified at the same time.
- o -O option:

A device type filename may also be specified as an output filename. If an output filename is omitted, an error will result. If a file type is omitted from the output filename specification, "input filename.LST" is assumed to have been specified as the output filename.

- o -PUBLIC and -NOPUBLIC options

These options may be input in their abbreviated format (-P or -NP indicated by the underline).

The -PUBLIC option specifies the output of information on PUBLIC symbols only.

The -NOPUBLIC option specifies the non-output of information on PUBLIC symbols.

If both the -PUBLIC and -NOPUBLIC options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To output information on modules in library file "78K3.LIB" to list filename "78K3.LST" with -P option specified for output of PUBLIC symbol information

*list -p -o78k3.lst 78k3.lib

o When list file "78K3.LST" is referenced, the output list file contains the following module information.

```
uCOM-78K/III Librarian Vx.xx      DATE : xx xxx xx          PAGE  1
LIB-FILE NAME : 78K3.LIB          (xx xxx xx)
0001 78K3MAIN.REL      (xx xxx xx)
      MAIN                      START
      NUMBER OF PUBLIC SYMBOLS :  2
0002 78K3SUB.REL      (xx xxx xx)
      CONVAH
      NUMBER OF PUBLIC SYMBOLS :  1
```

(7) HELP subcommand

Description format: HELP

Abbreviated format: H

Function

The HELP subcommand outputs the HELP message to the console.

Explanation

The HELP message is a list of all librarian subcommands and their functional descriptions. Use the HELP subcommand or --option if you want to refer to this message when executing the librarian.

Application Examples

Example 1: Input HELP subcommand as shown below and the HELP message will be displayed on the screen.

*help

```
Subcommands : create, add, delete, replace, pick, list, help, exit

Usage : subcommand[ option] masterLBF[ option] transaction[ option]

           transaction ::= OMFname
                        LBFname[(modulename[...])]

<create > : create masterLBF[ transaction]
<add    > : add masterLBF transaction
<delete > : delete masterLBF(modulename[...])
<replace> : replace masterLBF transaction
<pick   > : pick masterLBF (modulename[...])
<list   > : list[ option] masterLBF[(modulename[...])]
           option : -P = output public symbol
                   -NP = no output public symbol
                   -O filename = specify output file name

<help   > : help
<exit   > : exit
```

(8) EXIT subcommand

| |
|--------------------------|
| Description format: EXIT |
|--------------------------|

| |
|-----------------------|
| Abbreviated format: E |
|-----------------------|

Function

The EXIT subcommand terminates the librarian.

Explanation

Use this subcommand to terminate the librarian.

Application Examples

Example 1: To terminate the librarian.

*exit

CHAPTER 8. LIST CONVERTER

The list converter (or list conversion program) accepts an assembly list file or object module file output by the assembler or a load module file output by the linker as an input file, embeds actual addresses in relocatable addresses or symbols in the input file, and outputs them as an absolute assembly list file. By this output list, you can save yourself a trouble of reading an assembly list while referring to a link map.

8.1 Input/Output Files of List Converter

The files listed in Table 8-1 below are input and output to and from the list converter.

Table 8-1. I/O Files of List Converter

| Type | Name and description of file | Default file type |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| Input files | <u>Object module file</u> A binary file containing machine language information and relocation information and symbol information related to relocation addresses of machine language instructions | .REL |
| | <u>Assembly list file</u> ... A file containing assembly information such as assembly list and cross-reference list. | .PRN |
| | <u>Load module file</u> A binary image file of object codes generated as a result of a linking process. | .LNK |
| | <u>Parameter file</u> A file containing the parameters of the executable program (i.e., list converter). This file must be created by the user. | .PLV |
| Output files | <u>Absolute assembly list file</u> A list file in which actual addresses of relocatable addresses and symbols in the input file are embedded. | .P |
| | <u>Error list file</u> A file containing error information during a list conversion process. | .ELV |

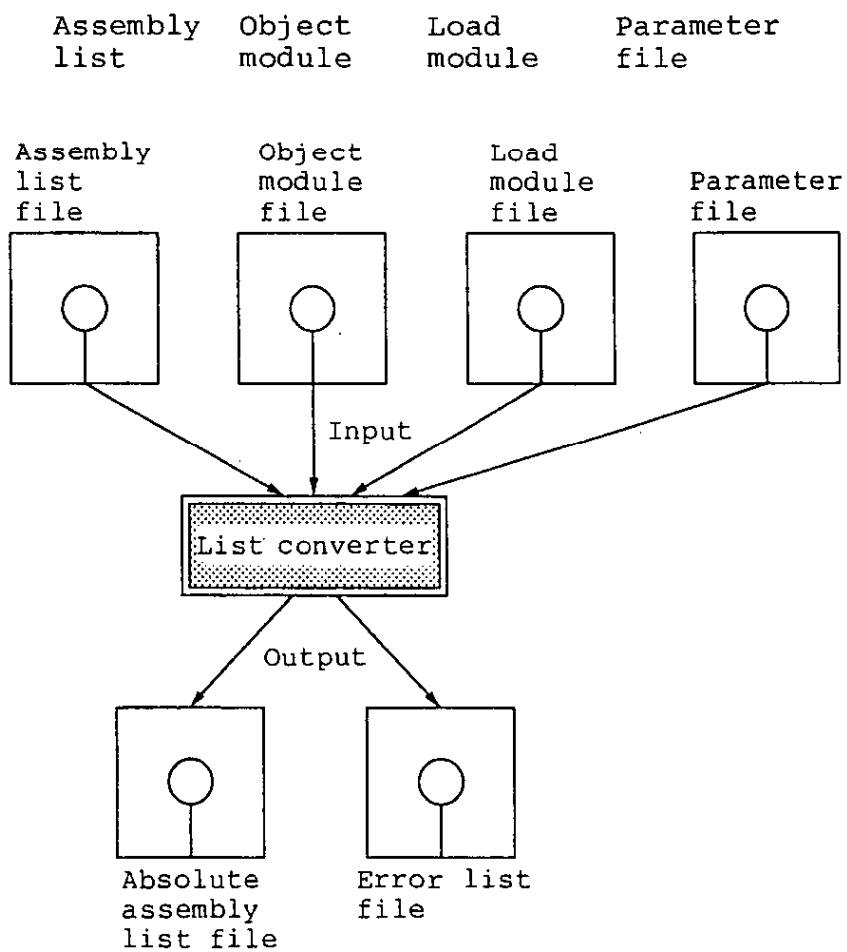


Fig. 8-1. I/O Files of List Converter

8.2 List Converter Functions

As compared with the absolute assembler, the relocatable assembler has the following advantages:

- o It allows program development to be initiated by two or more programmers.
- o It facilitates development and maintenance of each program by dividing it into several modules (subprograms).
- o It allows the library management of programs.
- o It is suitable for development of large-scale programs.

On the other hand, the relocatable assembler also has these disadvantages:

- o Addresses in an assembly list do not coincide with their physical addresses.
- o The value of each external symbol is 0 in an assembly list and thus a link map must be referenced to obtain the actual value of the external symbol.
- o The value of a relocatable segment in an assembly list differs from its actual value.

These disadvantages lead to lowering the productivity of documents particularly for program debugging and program maintenance.

This list converter helps solve the above disadvantages of the relocatable assembler.

- (1) Addresses in an absolute assembly list output by the list converter are in complete agreement with those at program operation time.
- (2) The actual value of each external symbol can be embedded in the absolute assembly list.
- (3) The value of a relocatable segment can be embedded in the absolute assembly list as an actual value.
- (4) An actual value can be embedded even for a symbol value on a symbol table or cross-reference list.

Example 1: Embedding of locations

o Assembly list

| | | | | | |
|----|----|------|----------|------------|-------------|
| 22 | 22 | ---- | | | CSEG |
| 23 | 23 | 0000 | 2B4100 | START: MOV | RFM, #00 |
| 24 | 24 | 0003 | 0BFC80FE | MOVW | SP, #0FE80H |
| 25 | 25 | 0007 | 2B4000 | MOV | MM, #00 |
| 26 | 26 | 000A | 0944F708 | MOV | STBC, #08H |
| 27 | 27 | | | | |
| 28 | 28 | 000E | 3A201A | MOV | HDTSA, #1AH |
| 29 | 29 | 0011 | 6720FE | MOVW | HL, #HDTSA |
| 30 | 30 | | | | |
| 31 | 31 | 0014 | R280000 | CALL | !CONVAH |

o Absolute assembly list

| | | | | | |
|----|----|------|----------|------------|-------------|
| 22 | 22 | ---- | | | CSEG |
| 23 | 23 | 0002 | 2B4100 | START: MOV | RFM, #00 |
| 24 | 24 | 0005 | 0BFC80FE | MOVW | SP, #0FE80H |
| 25 | 25 | 0009 | 2B4000 | MOV | MM, #00 |
| 26 | 26 | 000C | 0944F708 | MOV | STBC, #08H |
| 27 | 27 | | | | |
| 28 | 28 | 0010 | 3A201A | MOV | HDTSA, #1AH |
| 29 | 29 | 0013 | 6720FE | MOVW | HL, #HDTSA |
| 30 | 30 | | | | |
| 31 | 31 | 0016 | R282200 | CALL | !CONVAH |

Example 2: Embedding of object codes

o Assembly list

| | | | | | |
|----|----|------|---------|------|-------------|
| 27 | 27 | | | | |
| 28 | 28 | 000E | 3A201A | MOV | HDTSA, #1AH |
| 29 | 29 | 0011 | 6720FE | MOVW | HL, #HDTSA |
| 30 | 30 | | | | |
| 31 | 31 | 0014 | R280000 | CALL | !CONVAH |
| 32 | 32 | | | | |
| 33 | 33 | 0017 | 6521FE | MOVW | DE, #STASC |
| 34 | 34 | 001A | D3 | MOV | A, B |
| 35 | 35 | 001B | 50 | MOV | [DE+], A |
| 36 | 36 | 001C | D2 | MOV | A, C |
| 37 | 37 | 001D | 50 | MOV | [DE+], A |

o Absolute assembly list

| | | | | | |
|----|----|------|---------|------|-------------|
| 27 | 27 | | | | |
| 28 | 28 | 0010 | 3A201A | MOV | HDTSA, #1AH |
| 29 | 29 | 0013 | 6720FE | MOVW | HL, #HDTSA |
| 30 | 30 | | | | |
| 31 | 31 | 0016 | R282200 | CALL | !CONVAH |
| 32 | 32 | | | | |
| 33 | 33 | 0019 | 6521FE | MOVW | DE, #STASC |
| 34 | 34 | 001C | D3 | MOV | A, B |
| 35 | 35 | 001D | 50 | MOV | [DE+], A |
| 36 | 36 | 001E | D2 | MOV | A, C |
| 37 | 37 | 001F | 50 | MOV | [DE+], A |

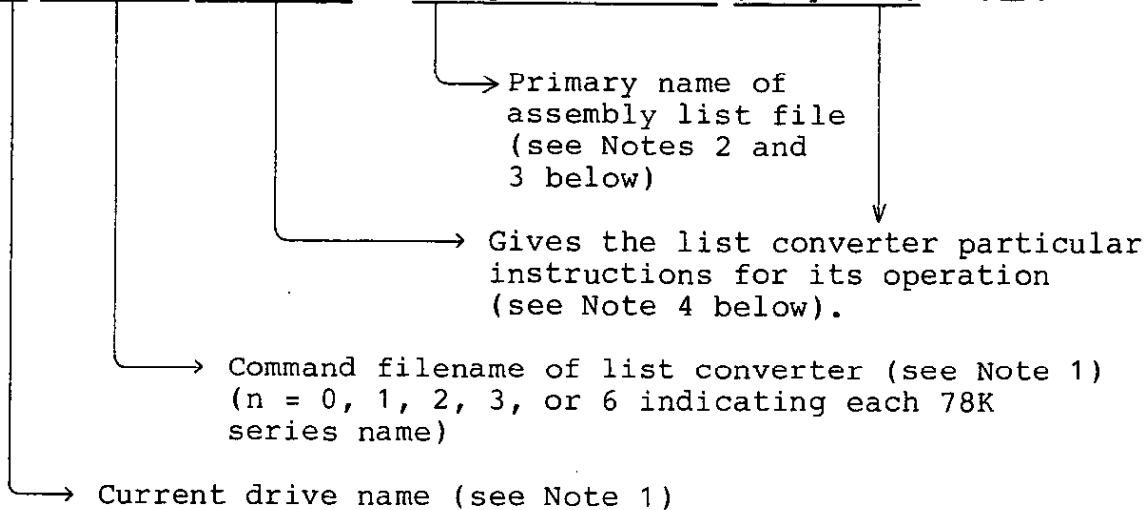
8.3 How to Start Up the List Converter

8.3.1 Starting up the list converter

The list converter can be started up (invoked) in either of the following two ways:

- (1) Start-up with the start-up command line of the list converter

X>lcnv78kn[Δoption]...Δinput-filename [Δoption]...[Δ]



Example: A>lcnv78k3 78k3main -l78k3.lnk

- NOTE: 1. With MS-DOS V3.10, the command file and overlay files of the list converter must have been stored in the same directory.
2. The file type of the input assembly list file must always be ".PRN".
 3. If only the primary name of an assembly list file is to be specified in the command line, the primary name of the input object module file or load module file must be the same as that of the assembly list file. The file type of the object module file or load module file must be as shown below.

| File name | File type |
|--------------------|-----------|
| Object module file | .REL |
| Load module file | .LNK |

- The list converter option (-R or -L) must be used to specify the input of a file which has a primary name different from that of the assembly list file.
4. If two or more list converter options are to be specified, each list converter option must be delimited with a space.
See Section 8.4 for details of the list converter options.

(2) Start-up with parameter file

A parameter file is used when all the required information for starting up the list converter cannot be specified in the start-up command line of the list converter or when the same list converter options are to be used repeatedly in each list conversion process.

When using this parameter file, the -F option must be specified in the start-up command line of the list converter to specify the use of the parameter file.

The list converter can be started up with a parameter file as follows:

```
X>lcnv78kn[  $\Delta$ input-filename]  $\Delta$ -f parameter-filename
```

→File containing information
required to start up the
list converter

→Option specifying parameter file

- o A parameter file must be created with the editor.
- o The description format of parameters within the parameter file is as shown below.

```
[ [  $\Delta$ ]option[ $\Delta$ option].. [  $\Delta$ ]  $\Delta$ ] ] ...
```

- o If no input filename is specified in the start-up command line of the list converter, the input filename(s) must be described within the parameter file.
- o The input filename may be described either before or after an option.
- o In the parameter file, all the list converter options and output filename which should normally be specified in the start-up command line must be described.

Example: To create a parameter file named "78K3.PLV" with the editor

o Contents of 78K3.PLV

```
;parameter file
78k3main -l78k3.lnk
-e78k3.elv
```

o Start-up of list converter using parameter file "78K3.PLV"

```
A>lcnv78k3 -f78k3.plv
```

8.3.2 Execution start and end messages

(1) Execution start message

When the list converter is started up, the following message is output to the console, indicating the start of the list converter execution.

```
List Conversion Program for RA78K/III Vx.xx  [xx xxx xx]
Copyright (C) NEC Corporation xxxx  USxxxxxxxxxx
```

```
Pass1: start...
Pass2: start...
```

(2) Execution end messages

o If no error is found as a result of a list conversion operation, the list converter will output the following message to the console and returns control to the OS.

Conversion complete.

- o If any fatal error is found during a list conversion operation, the list converter will output the following message to the console, stop its processing, and return control to the OS.

Example 1:

```
A>lcnv78k3 sample
```

```
List Conversion Program for RA78K/III Vx.xx  [xx xxx xx]  
Copyright (C) NEC Corporation xxxx  USxxxxxxxxxx
```

```
A006 File not found 'SAMPLE.PRN'  
Program aborted
```

In this example, the list conversion operation was discontinued by an abort error resulting from the specification of an assembly list file which does not exist in drive A.

Example 2:

```
A>lcnv78k3 78k3main -a
```

```
List Conversion Program for RA78K/III Vx.xx  [xx xxx xx]  
Copyright (C) NEC Corporation xxxx  USxxxxxxxxxx
```

```
A018 Option is not recognized '-a'  
Program aborted
```

In this example, the list conversion operation was discontinued by an abort error resulting from the input of a list converter option "-A (-a)" which is not recognized by the list converter.

If the list converter is aborted following the output of an error message, check the cause of the error by referring to Chapter 11, Error Messages and take corrective action(s) as required.

8.4 List Converter Options

8.4.1 Types of list converter options

A list converter option gives the list converter particular instructions for its operation and is divided into the following six types:

Table 8-2. Types of List Converter Options

| No. | Classification | Option | Functional description |
|-----|-------------------------------------------------------------|--------|-------------------------------------------------------------------------------------------|
| 1 | Option for object module file input specification | -R | Specifies the input of an object module file. |
| 2 | Option for load module file input specification | -L | Specifies the input of a load module file. |
| 3 | Option for absolute assembly list file output specification | -O | Specifies the output of an absolute assembly list file. |
| 4 | Options for error list file output specification | -E | Specifies the output or non-output of an error list file |
| | | -NE | |
| 5 | Option for parameter file specification | -F | Specifies the input of input filename and options from the file specified by this option. |
| 6 | Option for HELP message output specification | -- | Specifies the output of HELP message to the console. |

The above table merely introduces to you all the available list converter options. Each of these list converter options is detailed in Subsection 8.4.2 below. For quick reference, see Appendix C.5, List of List Converter Options in which the description format of each option and the relationship between one option and the other are also outlined.

8.4.2 Description of each list converter option

A detailed description of each list converter option is provided in this subsection.

(1) Option for object module file input specification (-R)

Description format: -R input-filename

Default assumption: -R assembly-list-filename.REL

Function

- o The -R option specifies the input of the object module file specified by this option to the list converter.

Use

Use the -R option if the primary name of the object module file to be input to the list converter is different from that of the assembly list file or the file type of the input filename is not ".REL".

Explanation

- o If any fatal error occurs, no absolute assembly list file will be output by the list converter.
- o If only a primary name is described in the input filename specification, the list converter will assume ".REL" as the file type of the input filename.

Application Examples

Example 1: To execute the list converter with -R option
specified to input object module file "SAMPLE.REL"
when the assembly list filename is "78K3MAIN.PRN"

A>lcnv78k3 78k3main -rsample.rel

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
Copyright (C) NEC Corporation xxxx USxxxxxxxxxx

Pass1: start...
Pass2: start...
Conversion complete.

(2) Option for load module file input specification (-L)

Description format: -L input-filename

Default assumption: -L assembly-list-filename.LNK

Function

- o The -L option specifies the input of the load module file specified by this option to the list converter.

Use

Use the -L option if the primary name of the load module file to be input to the list converter is different from that of the assembly list file or the file type of the input filename is not ".LNK".

Explanation

- o If any fatal error occurs, no absolute assembly list file will be output by the list converter.
- o If only a primary name is described in the input filename specification, the list converter will assume ".LNK" as the file type of the input filename.

Application Examples

Example 1: To execute the list converter with -L option
specified to input load module file "SAMPLE.LNK"
when the assembly list filename is "78K3MAIN.PRN"

A>lcnv78k3 78k3main -lsample.lnk

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
Copyright (C) NEC Corporation xxxx USxxxxxxxxxx

Pass1: start...
Pass2: start...
Conversion complete.

-O

Absolute assembly list file
output specification

(3) Option for absolute assembly list file output specification
(-O)

Description format: -O output-filename

Default assumption: -O assembly-list-filename.P

Function

The -O option specifies the output of an absolute assembly list file. It also specifies the output destination or output filename of the absolute assembly list file to be output by the list converter.

Use

Use the -O option if you want to change the output destination or output filename of an absolute assembly list file.

Explanation

- o An output filename can be specified with either a disk type filename or a device type filename. Only the following device type filenames can be used with this option: CON, PRN, NUL, and AUX. If "CLOCK" is specified as an output filename, an abort error will result.
- o If the same device as that of an error list file is specified as a device type output filename, an abort error will also result.
- o If an output filename is omitted from the -O option specification, "assembly-list-filename.P" is assumed as the output absolute assembly list filename.
- o If only a primary name is described in the output filename specification, the list converter will assume ".P" as the file type of the output filename.

-O

Absolute assembly list file
output specification

- o If a drive name is omitted from the -O option specification, the absolute assembly list file will be output to the current drive.

Application Examples

Example 1: To execute the list converter with -O option specified to create an absolute assembly list file named "SAMPLE.P"

A>lcnv78k3 78k3main -osample.p

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
Copyright (C) NEC Corporation xxxx USxxxxxxxxxx

Pass1: start...
Pass2: start...
Conversion complete.

Example 2: To execute the list converter with -O option specified to output an absolute assembly list file to PRN (Printer).

A>lcnv78k3 78k3main -oprn

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
Copyright (C) NEC Corporation xxxx USxxxxxxxxxx

Conversion complete.

(4) Options for error list file output specification (-E/-NE)

Description format: -E [output-filename]

or

-NE

Default assumption: -NE

Function

- o The -E option specifies the output of an error list file. It also specifies the output destination or output filename of an error list file to be output by the list converter.
- o The -NE option specifies the non-output of an error list file.

Use

- o Use the -E option if you want to save error messages to a file.
- o Also use the -E option if you want to change the output destination or output filename of an error list file.

Explanation

- o An output filename can be specified with either a disk type filename or a device type filename. However, if a device type filename "CLOCK" is specified as an output filename, an abort error will result.
- o If the same device as that of an absolute assembly list file is specified as an device type output filename, an abort error will also result.
- o If an output filename is omitted from the -E option specification, "assembly-list-filename.ELV" is assumed as the output error list filename.
- o If only a primary name is described in the output filename specification, the list converter will assume ".ELV" as the file type of the output filename.

- o If a drive name is omitted from the -E option specification, the error list file will be output to the current drive.
- o If the -E and -NE options are specified at the same time, whichever you specified later will take precedence over the other.

Application Examples

Example 1: To execute the list converter with -E option specified to create an error list file named "SAMPLE.ELV"

```
A>lcnv78k3 78k3main -esample.elv
```

```
List Conversion Program for RA78K/III Vx.xx  [xx xxx xx]  
Copyright (C) NEC Corporation xxxx  USxxxxxxxxxx
```

```
Pass1: start...  
Pass2: start...  
Conversion complete.
```

- o When error list file "SAMPLE.ELV" is referenced, the output error list file will look like this.

```
Pass1: start  
*** WARNING W101 Load module file is older than object module file '78K3MAIN.LNK  
, 78K3MAIN.REL'  
*** WARNING W102 Load module file is older than assemble module file '78K3MAIN.L  
NK, 78K3MAIN.PRN'  
  
Pass2: start
```

(5) Option for parameter file specification (-F)

Description format: -F filename

Default assumption: Options and input filenames can
be input only from start-up
command line.

Function

The -F option tells the list converter that options and input name(s) will be input from the file specified by this option.

Use

- o Use the -F option if all the required parameters for starting up the list converter cannot be specified in the start-up command line.
- o If you have a set of list converter options which you must specify repeatedly at each list conversion operation, describe these list converter options in a parameter file and then specify the -F option in the start-up command line.

Explanation

- o A filename can be specified with only a disk type filename. If any device type filename is specified with this option, an abort error will result.
- o If a filename is omitted from the -F option specification, an abort error will also result.
- o If only a primary name is specified in the filename specification, the list converter will assume ".PLV" as the file type of the input filename and then open the file.
- o Nesting of parameter files is not allowed. If the -F option is specified in a parameter file, an abort error will result.

- o The number of characters that can be described in a parameter file is not limited.
- o A blank character, Tab character, or " " must be used as a delimiter between options or input filenames.
- o The options and input filenames described in the parameter file will be expanded to the location on the command line where the parameter file (-F option) has been specified.
- o The list converter will process these expanded options in the order from the last input option.
- o If two or more -F options are specified at the same time, an abort error will result.

Application Examples

Example 1: To execute the list converter with -F option specified

- o Contents of parameter file "78K3.PLV"

```
;parameter file
78k3main -l78k3.lnk
-e78k3.elv
```

- o Enter -F option in the command line as follows:

```
A>lcnv78k3 -f78k3.plv
```

```
List Conversion Program for RA78K/III Vx.xx  [xx xxx xx]
Copyright (C) NEC Corporation xxxx  USxxxxxxxxxxx
```

```
Pass1: start...
Pass2: start...
Conversion complete.
```

--

HELP message display specification

(6) Option for HELP message display specification (--)

Description format: --

Default assumption: No HELP message is displayed.

Function

The -- option tells the list converter to display the HELP message on the console.

Use

The HELP message is a list of all list converter options and their functional descriptions. Use the -- option if you want to refer to this message when executing the list converter.

Explanation

If the -- option is specified, all the other list converter options specified at the same time will become invalid.

Application Examples

Example 1: Input -- option as shown below and the HELP message will be displayed on the screen.

A>lcnv78K3 --

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
Copyright (C) NEC Corporation xxxx USxxxxxxxxxx

usage : LCNV78K3 [option[...]] input-file [option[...]]
The option is as follows([] means omissible).
-Rfile :Specify object module file.
-Lfile :Specify load module file.
-Pfile :Specify assemble list file (relocatable assemble list file).
-Ofile :Specify output list file (absolute assemble list file).
-Ffile :Input option or input-file name from specified file.
-E[file]:Create error list file.
-- :Show this message.

CHAPTER 9. OUTPUT LISTS OF PROGRAMS

9.1 Output Lists of Assembler

The assembler outputs the following lists:

| Output list filename | Output list name |
|----------------------|----------------------|
| Assembly list file | Assembly list |
| | Symbol list |
| | Cross-reference list |
| Error list file | Error list |

9.1.1 Header of assembly list file

The header section is always output at the beginning of an assembly list file.

[Output Format]

uCOM-78K/III Assembler ①Vx.xx ②

Date:③ xxx xxxx Page:④ 1

⑤

Command: ⑥ -c310 78k3main.asm -lw80

Para-file:⑦

In-file: ⑧ 78K3MAIN.ASM

Obj-file: ⑨ 78K3MAIN.REL

Prn-file: ⑩ 78K3MAIN.PRN

[Description of Output Items]

| Item No. | Description |
|----------|--------------------------------------------------------------------------------------------------------------------|
| ① | Vx.xx: Version number of the assembler |
| ② | Title character string (namely, the character string specified by the -LH option or the TITLE control instruction) |
| ③ | Date: Date when the assembly list was created |
| ④ | Page: Page number |
| ⑤ | Subtitle character string (namely, the character string specified by the SUBTITLE control instruction) |
| ⑥ | Command: Image of command line |
| ⑦ | Para-file: Contents of parameter file |
| ⑧ | In-file: Input source module filename |
| ⑨ | Obj-file: Output object module filename |
| ⑩ | Prn-file: Assembly list filename |

9.1.2 Assembly list

The result of an assembly operation and error messages, if any, are output on this list.

[Output Format]

```

      Assemble list
ALNO  STNO  ADRS  OBJECT  M I  SOURCE STATEMENT
      ③④
① 1    ② 1          ⑤ $      PC(310)
① 2    ② 2
① 3    ② 3          ⑤        NAME    SAMPM
① 4    ② 4
5      5          ⑤        PUBLIC  MAIN, START
⑦ ***  ERROR E137, STNO 5 ( 0) Public symbol is undefined
6      6          ⑤        EXTRN   CONVAH
7      7
8      8          ⑤ DATA   DSEG     AT 0FE20H
9      9 ⑥ FE20      HDTSA:  DS      1
10     10 ⑥ FE21     STASC:  DS      2
11     11
12     12 ⑥ -----  CODE     CSEG     AT 0H
13     13          MAIN     DW      START
⑦ ***  ERROR E101, STNO 13 ( 5) Syntax error
14     14
15     15 -----
16     16 0000 ⑧ 2B4100  START:  MOV     RFM, #00
17     17 0003 ⑧ 0BFC80FE  MOVW    SP, #0FE80H
18     18 0007 ⑧ 2B4000     MOV     MM, #00
19     19 000A ⑧ 0944F708   MOV     STBC, #08H
      ...

```

Segment informations:

```

      ADRS    LEN    NAME
⑨ FE20 ⑩ 0003H ⑪ DATA
⑨ 0000 ⑩ 0000H ⑪ CODE
⑨ 0000 ⑩ 0020H ⑪ ?CSEG

```

Target chip: ⑫ uPD78310

Assembly complete, ⑬ 2 error(s) and ⑭ 0 warning(s) found. (⑮ 13)

[Description of Output Items]

| Item No. | Description |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ① | ALNO: Line number of source module image |
| ② | STNO: Line number (includes Include file expansion or macroexpansion) |
| ③ | M: MACRO indicator M : Macro-defined line #n : Macro-expanded line (where "n" indicates nesting level) Blank: Line neither macro-defined nor macro-expanded |
| ④ | I: INCLUDE indicator In : This line is in the INCLUDE file (where "n" indicates nesting level) Blank : INCLUDE file not used. |
| ⑤ | SOURCE STATEMENT: Source program statement |
| ⑥ | ADRS: Location counter value |
| ⑦ | ***: Line at which a fatal or warning error occurred |
| ⑧ | OBJECT: Relocation information R : Object code or symbol value will be changed by the linker. Blank: Object code or symbol value will not be changed. Object code Symbol value set with EQU or SET directive |
| ⑨ | ADRS: Segment address |
| ⑩ | LEN: Segment size |
| ⑪ | NAME: Segment name |
| ⑫ | Target chip: Target device of this assembler |
| ⑬ | Number of fatal errors |
| ⑭ | Number of warning errors |
| ⑮ | Last error line |

9.1.3 Symbol table list

This list contains information of the symbols (including local symbols) defined in the source module.

[Output Format]

Symbol Table List

| VALUE | ATTR | RTYP | NAME | VALUE | ATTR | RTYP | NAME |
|---------|--------|-------|----------|-------|--------|-------|---------|
| ① ----H | ② CSEG | ③ EXT | ④ ?CSEG | | ② CSEG | | ④ CODE |
| ① FE20H | ② ADDR | | ④ CONVAH | | ② DSEG | | ④ DATA |
| | ② MOD | | ④ HDTSA | ① 0H | ② ADDR | ③ PUB | ④ MAIN |
| ① FE21H | ② ADDR | | ④ SAMPM | ① 0H | ② ADDR | ③ PUB | ④ START |
| | | | ④ STASC | | | | |

[Description of Output Items]

| Item No. | Description |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ① | VALUE: Symbol value |
| ② | ATTR: Symbol attribute CSEG : Code segment name DSEG : Data segment name BSEG : Bit segment name MOD : Module name SET : Symbol defined with SET directive NUM : Symbol with NUMBER attribute DNUM : Symbol with DNUMBER attribute ADDR : Symbol with ADDRESS attribute ABIT : Symbol with BIT attribute (addr.bit) SABIT: Symbol with BIT attribute (saddr.bit) SFBIT: Symbol with BIT attribute (sfr.bit) RBIT : Symbol with BIT attribute (A.bit, X.bit, PSW.bit, PSWL.bit, PSWH.bit) RBBIT: Symbol with BIT attribute (br.bit) RWBIT: Symbol with BIT attribute (wr.bit) Blank: Externally referenced symbol declared with EXTRN or EXTBIT directive *****: Undefined symbol |
| ③ | RTYP: Symbol reference format EXT : Externally referenced symbol declared with EXTRN directive EXTB : Externally referenced bit symbol declared with EXTBIT directive PUB : Externally defined symbol declared with PUBLIC directive Blank: This column is blank for a local symbol, segment name, macro name, or module name. *****: Undefined symbol |
| ④ | NAME: Defined symbol name |

9.1.4 Cross-reference list

This list contains information on the line number of a source module at which a symbol defined in the source module is referenced.

[Output Format]

Cross-Reference List

| NAME | VALUE | R | ATTR | RTYP | SEGNAME | XREFS |
|----------|---------|-----|--------|-------|---------|------------|
| ① ?CSEG | | | ④ CSEG | | ⑥ ?CSEG | ⑦ 22# |
| ① CODE | | | ④ CSEG | | ⑥ CODE | ⑦ 19# |
| ① CONVAH | ② ----H | ③ E | | ⑤ EXT | | ⑦ 13# 31 |
| ① DATA | | | ④ DSEG | | ⑥ DATA | ⑦ 15# |
| HDTSA | ② FE20H | | ④ ADDR | | ⑥ DATA | 16# 28 29 |
| MAIN | ② 0H | | ADDR | ⑤ PUB | CODE | 12# 20# |
| SAMPM | | | MOD | | | 3# |
| START | 0H | ③ R | ADDR | ⑤ PUB | ?CSEG | 12# 20 23# |
| STASC | FE21H | | ADDR | | DATA | 17# 33 |

[Description of Output Items]

| Item No. | Description |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ① | NAME: Defined symbol name |
| ② | VALUE: Symbol value |
| ③ | R: Relocation attribute R : Relocatable symbol E : External symbol Blank: Absolute symbol * : Undefined symbol |
| ④ | ATTR: Symbol attribute CSEG : Code segment name DSEG : Data segment name BSEG : Bit segment name MOD : Module name SET : Symbol defined with SET directive NUM : Symbol with NUMBER attribute DNUM : Symbol with DNUMBER attribute ADDR : Symbol with ADDRESS attribute ABIT : Symbol with BIT attribute (addr.bit) SABIT: Symbol with BIT attribute (saddr.bit) SFBIT: Symbol with BIT attribute (sfr.bit) RBIT : Symbol with BIT attribute (A.bit, X.bit, PSW.bit, PSWL.bit, PSWH.bit) RBBIT: Symbol with BIT attribute (br.bit) RWBIT: Symbol with BIT attribute (wr.bit) Blank: Externally referenced symbol declared with EXTRN or EXTBIT directive *****: Undefined symbol |

| Item No. | Description |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ⑤ | RTYP: Symbol reference format EXT : Externally referenced symbol declared with EXTRN directive EXTB : Externally referenced bit symbol declared with EXTBIT directive PUB : Externally defined symbol declared with PUBLIC directive Blank: This column is blank for a local symbol, segment name, macro name, or module name. *****: Undefined symbol |
| ⑥ | SEGNAME: Defined segment name |
| ⑦ | XREFS: Line number at which the symbol indicated in Item 1 has been defined or referenced xxxxx#: Line at which symbol has been defined xxxxx▲: Line at which symbol has been referenced (where ▲ must be only one blank) xxxxx@: Line at which symbol has been declared as EXTRN, EXTBIT, or PUBLIC. |

9.1.5 Error list

This list contains information on each error which has occurred during the start-up processing of the assembler.

[Output Format]

Pass1 Start

① 78KMAIN.ASM(②10) : ③E101 ④Syntax error

① 78KMAIN.ASM(②12) : ③E101 ④Syntax error

Pass2 Start

① 78KMAIN.ASM(②10) : ③E101 ④Syntax error

① 78KMAIN.ASM(②12) : ③E101 ④Syntax error

① 78KMAIN.ASM(②26) : ③E158 ④Undefined symbol reference

① 78KMAIN.ASM(②26) : ③E102 ④Illegal expression

[Description of Output Items]

| Item No. | Description |
|----------|---------------------------------------------------|
| ① | Source module filename in which an error occurred |
| ② | Line number at which the error occurred |
| ③ | Error number |
| ④ | Error message |

9.2 Output Lists of Linker

The linker outputs the following lists:

| Output list filename | Output list name |
|----------------------|--------------------|
| Link list file | Map list |
| | PUBLIC symbol list |
| | Local symbol list |
| Error list file | Error list |

9.2.1 Header of link list file

The header section is always output at the beginning of a link list file.

[Output Format]

uCOM-78K/III LINKER ①Vx.xx

Date:② xxx xxxx Page:③ 1

Command: ④78k3main.rel 78k3sub.rel -g -d78k3.dr -o78k3.lnk -p78k3.map

Para-file: ⑤

Out-file: ⑥78K3.LNK

Map-file: ⑦78K3.MAP

Direc-file:⑧78K3.DR

Directive: ⑨memory ROM : (00000h,03FFFh)
memory RAM : (0F000h,00EFFh)

*** Link information ***

⑩ 3 output segment(s)

⑪ 40H byte(s) real data

⑫ 17 symbol(s) defined

[Description of Output Items]

| Item No. | Description |
|----------|-----------------------------------------------------|
| ① | Vx.xx: Version number of assembler |
| ② | Date: Date when the assembly list was created |
| ③ | Page: Page number |
| ④ | Command: Image of command line |
| ⑤ | Para-file: Contents of parameter file |
| ⑥ | Out-file: Output load module filename |
| ⑦ | Map-file: Link list filename |
| ⑧ | Direc-file: Directive filename |
| ⑨ | Directive: Contents of the directive file |
| ⑩ | Number of segments output to the load module file |
| ⑪ | Size of segment data output to the load module file |
| ⑫ | Number of symbols output to the load module file |

9.2.2 Map list

This list contains information on the location of each segment.

[Output Format]

*** Memory map ***

① SPACE=REGULAR

MEMORY=②ROM

BASE ADDRESS=③0000H SIZE=④3FFFH

| OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS | SIZE | |
|-------------------|------------------|-----------------|-----------------|---------|-----------|
| ⑥ CODE | | | ⑨ 0000H | ⑩ 0002H | ⑪ CSEG AT |
| | ⑦ CODE | ⑧ SAMPM | ⑨ 0000H | ⑩ 0002H | |
| ⑥ ?CSEG | | | ⑨ 0002H | ⑩ 003CH | ⑪ CSEG |
| | ⑦ ?CSEG | ⑧ SAMPM | 0002H | 0020H | |
| | ⑦ ?CSEG | ⑧ SAMPS | 0022H | 001CH | |
| ⑤ * gap * | | | 003EH | 3FC1H | |

MEMORY=②RAM

BASE ADDRESS=③F000H SIZE=④0EFFH

| OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS | SIZE | |
|-------------------|------------------|-----------------|-----------------|---------|-----------|
| ⑤ * gap * | | | ⑨ F000H | ⑩ 0E20H | |
| ⑥ DATA | | | ⑨ FE20H | ⑩ 0003H | ⑪ DSEG AT |
| | ⑦ DATA | ⑧ SAMPM | ⑨ FE20H | ⑩ 0003H | |
| ⑤ * gap * | | | FE23H | 00DCH | |

[Description of Output Items]

| Item No. | Description |
|----------|-------------------------------------------------------------------------------------------|
| ① | SPACE=: Memory space name |
| ② | MEMORY=: Memory area name |
| ③ | BASE ADDRESS=: Start address of the memory area |
| ④ | SIZE=: Size of the memory area |
| ⑤ | Output group name "gap" is displayed for an area in which no segment has been located. |
| ⑥ | OUTPUT SEGMENT: Name of output segment to the load module file |
| ⑦ | INPUT SEGMENT: Name of input segment from the object module file |
| ⑧ | INPUT MODULE: Input module name |
| ⑨ | BASE ADDRESS: Start address of the input or output segment |
| ⑩ | SIZE: Size of the input or output segment |
| ⑪ | Segment type and relocation attribute of the output segment |

9.2.3 PUBLIC symbol list

This list contains information on the PUBLIC symbols defined in each input module.

[Output Format]

*** Public symbol list ***

| MODULE | ATTR | VALUE | NAME |
|---------|--------|---------|----------|
| ① SAMPM | ② ADDR | ③ 0000H | ④ MAIN |
| ① SAMPM | ② ADDR | ③ 0002H | ④ START |
| ① SAMPS | ② ADDR | ③ 0022H | ④ CONVAH |

[Description of Output Items]

| Item No. | Description |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ① | MODULE: Name of module in which the PUBLIC symbol indicated in Item 4 has been defined |
| ② | ATTR: Symbol attribute CSEG : Code segment name DSEG : Data segment name BSEG : Bit segment name MOD : Module name SET : Symbol defined with SET directive NUM : Symbol with NUMBER attribute DNUM : Symbol with DNUMBER attribute ADDR : Symbol with ADDRESS attribute ABIT : Symbol with BIT attribute (addr.bit) SABIT: Symbol with BIT attribute (saddr.bit) SFBIT: Symbol with BIT attribute (sfr.bit) RBIT : Symbol with BIT attribute (A.bit, X.bit, PSW.bit, PSWL.bit, PSWH.bit) RBBIT: Symbol with BIT attribute (br.bit) RWBIT: Symbol with BIT attribute (wr.bit) Blank: Externally referenced symbol declared with EXTRN or EXTBIT directive *****: Undefined symbol |
| ③ | VALUE: Symbol value |
| ④ | NAME: PUBLIC symbol name |

9.2.4 Local symbol list

This list contains information on the local symbols defined in each input module.

[Output Format]

*** Local symbol list ***

| MODULE | ATTR | VALUE | NAME |
|---------|--------|---------|---------|
| ① SAMPM | ② MOD | | ④ SAMPM |
| ① SAMPM | ② DSEG | | ④ DATA |
| ① SAMPM | ② ADDR | ③ FE20H | ④ HDTSA |
| ① SAMPM | ② ADDR | ③ FE21H | ④ STASC |
| SAMPM | CSEG | | CODE |
| SAMPM | CSEG | | ?CSEG |
| SAMPS | MOD | | SAMPS |
| SAMPS | CSEG | | ?CSEG |
| SAMPS | ADDR | ③ 0035H | SASC |
| SAMPS | ADDR | ③ 003BH | SASC1 |

[Description of Output Items]

| Item No. | Description |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ① | MODULE: Name of module in which the local symbol indicated in Item 4 has been defined |
| ② | ATTR: Symbol attribute CSEG : Code segment name DSEG : Data segment name BSEG : Bit segment name MOD : Module name SET : Symbol defined with SET directive NUM : Symbol with NUMBER attribute DNUM : Symbol with DNUMBER attribute ADDR : Symbol with ADDRESS attribute ABIT : Symbol with BIT attribute (addr.bit) SABIT: Symbol with BIT attribute (saddr.bit) SFBIT: Symbol with BIT attribute (sfr.bit) RBIT : Symbol with BIT attribute (A.bit, X.bit, PSW.bit, PSWL.bit, PSWH.bit) RBBIT: Symbol with BIT attribute (br.bit) RWBIT: Symbol with BIT attribute (wr.bit) Blank: Externally referenced symbol declared with EXTRN or EXTBIT directive *****: Undefined symbol |
| ③ | VALUE: Symbol value |
| ④ | NAME: Local symbol name |

9.2.5 Error list

This list contains information on each error which has occurred during the start-up processing of the linker.

[Output Format]

```
*** ERROR ① F405 ② Undefined symbol 'CONVAH' in file '78K3MAIN.REL'  
*** ERROR ① A399 ② Cannot create output file for PASS3 ERROR(S)
```

[Description of Output Items]

| Item No. | Description |
|----------|---------------|
| ① | Error number |
| ② | Error message |

9.3 Output List of Object Converter

The object converter outputs only the following list:

| Output list filename | Output list name |
|----------------------|------------------|
| Error list file | Error list |

9.3.1 Error list

This list contains information on each error which has occurred during the start-up processing of the object converter.

[Output Format & Description of Output Items]

Same as those of the error list output by the linker.

9.4 Output List of Librarian

The librarian outputs only the following list:

| Output list filename | Output list name |
|----------------------|---------------------------------|
| List file | Library information output list |

9.4.1 Library information output list

This list contains information on each module in the library file.

[Output Format]

```
uCOM-78K/III librarian Vx.xx      DATE : ①xx xxx xx                PAGE ②1
LIB-FILE NAME : ③78K3.LIB          (④xx xxx xx)
⑤0001 ⑥78K3MAIN.REL      (⑦xx xxx xx)
    ⑧MAIN                      ⑧START
NUMBER OF PUBLIC SYMBOLS : ⑨2
⑤0002 ⑥78K3SUB.REL      (⑦xx xxx xx)
    ⑧CONVAH
NUMBER OF PUBLIC SYMBOLS : ⑨1
```

[Description of Output Items]

| Item No. | Description |
|----------|---------------------------------------------------------------------------|
| ① | DATE: Date when the list was created |
| ② | PAGE: Page number/number of pages |
| ③ | LIB-FILE NAME: Library filename |
| ④ | (xx xxx xx): Date when the library file was created |
| ⑤ | Module serial number (serial numbering begins with 0001) |
| ⑥ | Module name |
| ⑦ | (xx xxx xx): Date when the module was created |
| ⑧ | PUBLIC symbol name |
| ⑨ | NUMBER OF PUBLIC SYMBOLS: Number of PUBLIC symbols defined in each module |

9.5 Output Lists of List Converter

The list converter outputs the following lists:

| Output list filename | Output list name |
|--------------------------------|------------------------|
| Absolute assembly list file | Absolute assembly list |
| Error list file | Error list |

9.5.1 Absolute assembly list

An absolute assembly list is an assembly list in which absolute values have been embedded.

[Output Format & Description of Output Items]

Same as those of the assembly list output by the assembler.

9.5.2 Error list

This list contains information on each error which has occurred during the start-up processing of the list converter.

[Output Format & Description of Output Items]

Same as those of the error list output by the assembler.

CHAPTER 10. UTILIZATION OF ASSEMBLER PACKAGE

10.1 How to Execute Each Operation with Efficiency (Use of EXIT Status Function)

On completion of the processing by each program in this assembler package, each program returns to the OS the maximum error level which has occurred during the processing as EXIT status.

The following EXIT status (ERROR LEVEL) codes are available:

- o Program terminated normally: 0
- o Program terminated with
WARNING outputs : 0
- o Program terminated with
FATAL ERROR outputs : 1
- o Program aborted : 2

By using these EXIT status codes with a batch file, each operation (assembly, linking, object conversion, etc.) may be carried out efficiently.

[Application Example]

- o Contents of batch file "RA.BAT"

```
ra78k3 -c310 %1.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echoY
echo on
ra78k3 -c310 %2.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echoY
echo on
lk78k3 %1.rel %2.rel -o%3.lnk -g
echo off
IF ERRORLEVEL 1 GOTO ERR
echoY
echo on
oc78k3 %3.lnk
echo off
IF ERRORLEVEL 1 GOTO ERR
GOTO EXIT
:ERR
echo Error has occurred.
echo off
:EXIT
```

- o To execute each program by using batch file "RA.BAT"

A> ra.bat

10.2 How to Prepare or Complete the Development Environment (Use of Environment Variables)

In the development of a program, required operations may be carried out smoothly if you create a directory for two or more related files to put them together in a single file. This can be implemented by specifying an environment variable.

This package supports the following environment variables to prepare or complete the program development environment.

PATH : Search path for executable files
INC78Kn: Search path for INCLUDE files (Assembler only)
LIB78Kn: Search path for library files (Linker only)
TMP : Path for temporary file creation
Note: n in the above INC78Kn and LIB78Kn denotes one of the numbers 0, 1, 2, 3, and 6 corresponding to 78K/0, 78K/I, 78K/II, 78K/III, and 78K/VI series names, respectively.

[Application Example]

- o Contents of AUTOEXEC.BAT file

```
;AUTOEXEC.BAT
verify on
break on
PATH A:¥BIN;A:¥BAT;A:¥RA78K3;      ←①
SET INC78K3=A:¥RA78K3¥INCLUDE      ←②
SET LIB78K3=A:¥RA78K3¥LIB          ←③
SET TMP=A:¥TMP                     ←④
```

[Explanation]

- ① By the PATH specification, executable files A:¥BIN, A:¥BAT, and A:¥RA78K3 are searched in the order named.
- ② By this environment variable specification, the assembler searches A: ¥RA78K3¥INCLUDE for Include file(s).
- ③ By this environment variable specification, the linker searches A: ¥RA78K3¥LIB for library file(s).
- ④ By this environment variable specification, each program creates a temporary file in A:¥TMP.

10.3 How to Interrupt Program Execution

The processing by each program can be interrupted by the control key input CTRL-C. If "break on" is specified in the AUTOEXEC.BAT file, control will be returned to the OS irrespective of the key input timing. If "break off" is specified, control will be returned to the OS only while the screen is on display. In either case, all temporary files and output files being open will be erased.

10.4 How to Increase the Readability of Assembly List

Print the title of an assembly list in its header section by using the -LH option or the TITLE control instruction. Use a title which will plainly indicate the contents of the assembly list. A subtitle may also be printed on the assembly list by using the SUBTITLE control instruction. See Chapter 4 in the "RA78K Series Assembler Package User's Manual for Language" for these control instructions.

[Application Example]

- o To assemble the source program with -LH option specified to print title "RA78K3_MAINROUTINE" in the header of an assembly list

- o When assembly list file "78K3MAIN.PRN" is referenced, the output assembly list will look like this.

```
A>ra78k3 -c310 78k3main.asm -lw80 -lhRA78K3 MAINROUTINE
```

```
uCOM-78K/III Assembler Vx.xx [xx xxx xx]
  Copyright (C) Corporation xxxx USxxxxxxxxxx
```

```
Pass1 Start
Pass2 Start
```

```
Assembly complete.      0 error(s) and      0 warning(s) found.
```

・ 7 8 K 3 M A I N . P R Nを参照します。

```
uCOM-78K/III Assembler Vx.xx RA78K3_MAINROUTINE      Date:xx xxx xxxx Page:  1
                               _____ Title
```

```
Command: -c310 78k3main.asm -lw80 -lhRA78K3_MAINROUTINE
Para-file:
In-file:  78K3MAIN.ASM
Obj-file: 78K3MAIN.REL
Prn-file: 78K3MAIN.PRN
```

Assemble list

| ALNO | STNO | ADRS | OBJECT | M I | SOURCE STATEMENT |
|------|------|------|--------|-----|-----------------------------------|
| 1 | 1 | | | | \$ PROCESSOR(310) |
| 2 | 2 | | | | |
| 3 | 3 | | | | NAME SAMPM |
| 4 | 4 | | | | ***** |
| 5 | 5 | | | | ; |
| 6 | 6 | | | | HEX -> ASCII Conversion Program * |
| 7 | 7 | | | | ; |
| 8 | 8 | | | | main-routine * |
| 9 | 9 | | | | ; |
| 10 | 10 | | | | ***** |
| 11 | 11 | | | | |
| 12 | 12 | | | | PUBLIC MAIN.START |
| 13 | 13 | | | | EXTRN CONVAH |

10.5 How to Save Yourself Trouble in Program Invocation

10.5.1 Describing control instructions in the source program

The control instructions which have the same functions as the options that you always specify at the start-up of the assembler should be specified at the beginning of the source program (i.e., module header). By so doing, you need not specify these assembler options in the command line each time you want to start up the assembler.

[Application Example]

```
$      PROCESSOR(310)
$      DEBUG
$      XREF      ] Control instructions

      NAME      SAMPM
;*****
;*
;*      HEX -> ASCII Conversion Program
;*
;*      main-routine
;*
;*****

      PUBLIC  MAIN, START
      EXTRN   CONVAH
      :
```

10.5.2 Creating a parameter file or subcommand file

A parameter file is used when all the required information for starting up the program (assembler, linker, object converter, or list converter) cannot be specified in the start-up command line of the program or when the same options are to be used repeatedly in each process.

In the librarian, by registering subcommands in a subcommand file and by using the subcommand file, a library file of object modules can be created easily.

[Application Examples]

Example 1: To assemble the source program with -F option specified to use a parameter file

o Contents of parameter file "78K3MAIN.PRA"

```
;parameter file
78k3main.asm -osample.rel -g
-psample.prn
```

o Enter -F option in the command line as follows:

```
A>ra78k3 -f78k3main.pra
```

Example 2: To start up the librarian using a subcommand file

o Contents of subcommand file "78K3.SLB"

```
;
;library creation command
;
create 78k3.lib
;
add 78k3.lib 78k3main.rel &
78k3sub.rel
;
exit
```

o Enter the subcommand file in the command line as follows:

```
A>lb78k3 <78k3.slb
```

10.6 Creation of Object Module Library File

Both the assembler and linker create one file for each output module. The more the number of input object modules, the more the number of output module files. Thus, a function to collect two or more modules into a single file is provided. This collection of modules is called a library and a file used to maintain and make available the modules is called a library file.

A library file can be input to the linker. Therefore, when the modular programming technique is employed for program development, files can be efficiently managed and operated if you create a library file of commonly used modules.

10.7 How to Change the Option Mark (Use of Environment Variable)

This assembler package is provided with an environment variable "OPTMARK" so that the user may change the option mark "-" to any other option mark.

OPTMARK : Search path for executable file

[Application Example]

o Contents of AUTOEXEC.BAT file

```
;AUTOEXEC.BAT
verify on
break on
PATH A:YBIN;A:YBAT;A:YRA78K3;
SET INC78K3=A:YRA78K3\INCLUDE
SET LIB78K3=A:YRA78K3\LIB
SET TMP=A:YTMP
SET OPTMARK=/
```

← Change option mark to "/"

o Enter option mark "/" in the command line as follows:

A>ra78k3 78k3main.asm /g /e

CHAPTER 11. ERROR MESSAGES

11.1 Assembler Error Messages

Error messages related to the Assembler

| | | |
|------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A101 | Message | Source file size 0 ' <u>filename</u> ' |
| | Cause | An attempt was made to input the source file indicated by the displayed filename which has a size of 0 bytes. |
| A102 | Message | Illegal processor type specified |
| | Cause | An error exists in the target device specification. |
| A103 | Message | Syntax error in module header |
| | Cause | A syntax error exists in the description format of a control instruction that can be described in the header of a source module. |
| A104 | Message | Can't use this control outside module header |
| | Cause | This control instruction cannot be described outside the header of a source module. |
| A105 | Message | Duplicate PROCESSOR control |
| | Cause | The PROCEESOR control instruction has been described two or more times in the header of a source module. |
| A106 | Message | Illegal source file name for module name |
| | Cause | Module name creation failed, because the primary name of the source module contains character(s) illegal for symbol name configuration or has already been defined in the source file. |
| A107 | Message | Default segment ?CSEG is already used |
| | Cause | An attempt was made to define the default segment when a segment definition was omitted. |
| A108 | Message | Symbol table overflow ' <u>symbol name</u> ' |
| | Cause | The number of symbols that can be defined exceeded the limit value (2900 symbols). The output symbol name is a symbol when this limit value was exceeded. |

| | | |
|------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A109 | Message | Too many DS |
| | Cause | Too many DS directives have been specified, resulting in an excessive gap between object codes within the segment and thus the required information cannot be output to the object module file. |
| | User action | Divide the source module file into two or more modules or reduce the number of DS directives. |
| A110 | Message | String table overflow |
| | Cause | An overflow has occurred in the string table. |
| | User action | Reduce the number of symbols each consisting of nine or more characters. |
| A111 | Message | Object code more than 128 bytes |
| | Cause | Object code exceeded 128 bytes per line of a source statement. |
| A112 | Message | No processor specified |
| | Cause | The target device name has not been specified either on the start-up command line nor in the source module file (source header). |

| | | |
|------|---------|-------------------------------------------------------------------|
| F201 | Message | Syntax error |
| | Cause | A syntax error exists in the description format of the statement. |
| F202 | Message | Illegal operand |
| | Cause | The operand description is incorrect. |
| F203 | Message | Illegal register |
| | Cause | A register that cannot be described has been specified. |
| F204 | Message | Illegal character |
| | Cause | An illegal character has been described in the source module. |
| F205 | Message | Unexpected LF in string |
| | Cause | An LF code appeared before closing the character string. |
| F206 | Message | Unexpected EOF in string |
| | Cause | The file reached its end before closing the character string. |
| F207 | Message | Unexpected null code in string |
| | Cause | A null code (00H) has been described in the character string. |

| | | |
|------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F301 | Message | Too complex expression |
| | Cause | The expression is too complex to evaluate. |
| F302 | Message | Absolute expression expected |
| | Cause | A relocatable expression has been described instead of an absolute expression. |
| F303 | Message | Illegal expression |
| | Cause | An error exists in the description format of an expression. |
| F304 | Message | Illegal symbol in expression ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name is a symbol that cannot be used in an expression. |
| F305 | Message | Too long string constant |
| | Cause | The string constant exceeded the limit value (2 characters). |
| F306 | Message | Illegal number |
| | Cause | An error exists in the described number. |
| F307 | Message | Division by zero |
| | Cause | An attempt was made to divide by zero. |
| F308 | Message | Too large integer |
| | Cause | The value of the integer constant exceeded 16 bits or 32 bits. |
| F309 | Message | Illegal bit value |
| | Cause | An error exists in the description of a bit value. |
| F310 | Message | Bit value out of range |
| | Cause | The specified bit value exceeded the range of 0 to 7 or 0 to 15. |
| F311 | Message | Operand out of range (n) |
| | Cause | With 78K/0, 78K/I, 78K/III: The specified operand value exceeded the range (0 to 7) for data n. With 78K/VI: The specified operand value exceeded the range (0 to 7) for data n3, (0 to 15) for data 4, or (0 to 31) for data n5. |

| | | |
|------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F312 | Message | Operand out of range (byte) |
| | Cause | The specified operand value exceeded the range (00H to FFH) or the specified byte value in the operand exceeded the range (-128 to +127). |
| F313 | Message | Operand out of range (addr5) |
| | Cause | The specified operand value exceeded the range (40H to 7EH or 8040H to 807EH) for data that can be described as "addr5". |
| F314 | Message | Operand out of range (addr11) |
| | Cause | The specified operand value exceeded the range (800H to FFFH) for data that can be described as "addr11". |
| F315 | Message | Operand out of range (saddr) |
| | Cause | With 78K/0, 78K/I, 78K/III: The specified operand value exceeded the range (0FE20H to 0FF1FH) for data that can be described as "saddr". With 78K/VI: The specified operand value exceeded the range (0FC00H to 0FEFFH) for data that can be described as "bsaddr", "wsaddr", or "dsaddr". |
| F316 | Message | Operand out of range (!addr16) |
| | Cause | The specified operand value exceeded the range (which differs depending on the target device) for data that can be described as "!addr16". |
| F317 | Message | Even expression expected |
| | Cause | With 78K/0, 78K/I, 78K/III: An odd-numbered address has been described as "saddrp" or "sfrp". With 78K/VI: An odd-numbered address has been described as "waddrp" or the operand of the WSFR directive. |
| F318 | Message | Operand out of range (sfr) |
| | Cause | With 78K/0, 78K/I, 78K/III: The specified operand value exceeded the range (FF00H to FFFFH) for data that can be described as the operand of the SFR or SFRP directive or an odd number has been described as the operand of the SFRP directive. With 78K/VI: The specified operand value exceeded the range (FF00H to FFFFH) for data that can be described as the operand of the SFR or SFRP directive or an odd value (number) has been described as the operand of the WFRP directive. |
| F319 | Message | Operand out of range (addr8) |
| | Cause | The specified operand value exceeded the range (00H to 00FFH) for data that can be described as "addr8". |

| | | |
|------|---------|-------------------------------------------------------------------|
| F320 | Message | A multiple of 4 expression expected |
| | Cause | A value other than a multiple of 4 has been described as "addr8". |

| | | |
|------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F401 | Message | Illegal symbol for PUBLIC ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name cannot be declared as a PUBLIC symbol with the PUBLIC directive. |
| F402 | Message | Illegal symbol for EXTRN/EXTBIT ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name cannot be declared as an external reference symbol with the EXTRN or EXTBIT directive. |
| F403 | Message | Can't define PUBLIC symbol ' <u>symbol name</u> ' |
| | Cause | A bit term other than "saddr.bit" has been defined in the PUBLIC symbol indicated by the displayed symbol name. |
| | User action | Because a symbol in which a bit term other than "saddr.bit" has been defined, a SET symbol, a symbol which has already been externally referenced or declared, a segment name, a module name, a macro name, a BSFR/WSFR symbol (user defined symbol), or an EQU symbol cannot be declared as PUBLIC, either cancel the PUBLIC declaration or change the EQU definition. |
| F404 | Message | Public symbol is undefined ' <u>symbol name</u> ' |
| | Cause | The PUBLIC symbol indicated by the displayed symbol name has not been defined. |
| F405 | Message | Illegal bit symbol |
| | Cause | As the bit symbol of an operand in a machine language instruction, a forward-referenced symbol or a symbol not appropriate as a bit symbol has been described. |
| | User action | Use a backward-referenced symbol or a symbol declared with the EXTBIT directive as the bit symbol of the operand. |
| F406 | Message | Can't refer to forward bit symbol ' <u>symbol name</u> ' |
| | Cause | The bit symbol indicated by the displayed symbol name has been either forward-referenced or described in an expression. |
| F407 | Message | Undefined symbol ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name has not been defined. |

| | | |
|------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F408 | Message | Multiple symbol definition ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name has already been defined. (Duplicate symbol definitions) |
| F409 | Message | Too many symbols in operand |
| | Cause | The number of symbols that can be described as operands per line exceeded the limit value. |
| F410 | Message | Phase error |
| | Cause | The value of a symbol has changed during the assembly phase. For example, an EQU symbol defined by using in an operand a label changed by optimization with the BR directive. |
| | User action | Correct the source program by searching the location at which the symbol is referenced and describing an expression there instead of the EQU symbol. |

| | | |
|------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| F501 | Message | Too many default ORG segments |
| | Cause | ORG directives without segment name specification have been described by exceeding the limit value (20 default ORG directives per module). |
| F502 | Message | Illegal segment name |
| | Cause | A symbol illegal as a segment name has been described. |
| F503 | Message | Different segment type ' <u>segment name</u> ' |
| | Cause | The segment indicated by the displayed segment name differs in segment type from another segment defined with the same name. |
| F504 | Message | Too many segments |
| | Cause | The number of segments that can be defined exceeded the limit value (100 segments). |
| F505 | Message | Current segment does not exist |
| | Cause | The ENDS directive has been described before the creation of a segment or before the creation or the next segment after the completion of one segment. |
| F506 | Message | Can't describe DB,DW,DS,label in BSEG |
| | Cause | The DB, DW, or DS directive has been described in a bit segment. |

| | | |
|------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F507 | Message | Can't describe opcodes (,RSS) outside CSEG |
| | Cause | With 78K/0, 78K/I, 78K/VI: A machine language instruction (op code) has been described outside a code segment. With 78K/III: A machine language instruction (op code) or RSS directive has been described outside a code segment. |
| F508 | Message | Can't describe DBIT outside BSEG |
| | Cause | The DBIT directive has been described outside a bit segment. |
| F509 | Message | Illegal address specified |
| | Cause | The address specified for allocation to an absolute segment exceeded the range applicable to the segment. |
| F510 | Message | Location counter overflow |
| | Cause | The location counter value exceeded the range applicable to the segment. |
| F511 | Message | Segment name expected |
| | Cause | A segment name has been omitted from the segment definition directive to define a segment with relocation attribute AT. |

| | | |
|------|---------|-----------------------------------------------------------------------------------------------------------|
| F601 | Message | Nesting over of include |
| | Cause | INCLUDE files have been nested by exceeding the limit value (2 nesting levels). |
| F602 | Message | Must specify switches |
| | Cause | The required switch names have not been specified. |
| F603 | Message | Too many switches described |
| | Cause | The number of switch names that can be described per module exceeded the limit value (five switch names). |
| F604 | Message | Nesting over of IF-clauses |
| | Cause | IF-ENDIF blocks have been nested by exceeding the limit value (eight nesting levels). |
| F605 | Message | Needless ELSE statement exists |
| | Cause | An ELSE statement exists in a location where it is unwanted. |
| F606 | Message | Needless ENDIF statement exists |
| | Cause | An ENDIF statement exists in a location where it is unwanted. |

| | | |
|------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F607 | Message | Missing ELSE or ENDIF |
| | Cause | An IF or _IF statement is not paired with an ELSE or ENDIF statement. |
| F608 | Message | Missing ENDIF |
| | Cause | An IF or _IF statement is not paired with an ENDIF statement. |
| F609 | Message | Illegal ELSEIF statement |
| | Cause | An ELSEIF or _ELSEIF statement has been described after an ELSE statement. |
| F610 | Message | Multiple symbol definition (MACRO) ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name has already been defined as a macro name. |
| F611 | Message | Illegal syntax of parameter |
| | Cause | An error exists in the formal parameter description of a macro. |
| F612 | Message | Too many parameters |
| | Cause | The number of formal parameters per macro-definition exceeded the limit value (16 formal parameters). |
| F613 | Message | Same name parameter described ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name has been specified as a formal parameter with the same name as another formal parameter in a macrodefinition. |
| F614 | Message | Can't nest macro definitions |
| | Cause | An attempt was made to define a macro in another macrodefinition. |
| F615 | Message | Illegal syntax of local symbol |
| | Cause | An error exists in the operand description of the LOCAL directive. |
| F616 | Message | Too many local symbols |
| | Cause | The number of local symbols that can be described per macro body exceeded the limit value (64 symbols). |
| F617 | Message | Missing ENDM |
| | Cause | An ENDM statement is missing from the macro definition directive. |
| F618 | Message | Illegal syntax of ENDM |
| | Cause | An error exists in the ENDM statement description. |
| F619 | Message | Illegal defined macro |
| | Cause | The referenced macro has an error when it was defined. |

| | | |
|------|---------|------------------------------------------------------------------------------------------------|
| F620 | Message | Illegal syntax of actual parameter |
| | Cause | An error exists in the actual parameter description of a macro. |
| F621 | Message | Nesting over of macro references |
| | Cause | Macro references have been nested by exceeding the limit value (eight nesting levels). |
| F622 | Message | Illegal syntax of EXITM |
| | Cause | An error exists in the EXITM statement description. |
| F623 | Message | Illegal operand of REPT |
| | Cause | An expression not allowed as an operand has been described in the REPT directive. |
| F624 | Message | More than ??RAFFFF |
| | Cause | The number of local symbols replaced in a macro-expansion exceeded 65,535 symbols. |
| F625 | Message | Unexpected ENDW |
| | Cause | An excess ENDW statement exists in the macro-definition. |
| F626 | Message | Can't describe LOCAL outside macro definition |
| | Cause | The LOCAL directive has been described in an ordinary source statement (outside a macro body). |

| | | |
|------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| W701 | Message | Too long source line |
| | Cause | The source statement length exceeded 128 characters. |
| | Program action | The assembler ignores 129th and subsequent characters in the source line. |
| W702 | Message | Duplicate PROCESSOR option and control |
| | Cause | Two different target devices have been specified by the -C option in the start-up command line and the PROCESSOR control instruction in the source header. |
| | Program action | The assembler accepts the -C option as valid and ignores the PROCESSOR control instruction in the source header. |
| W703 | Message | Multiple defined module name |
| | Cause | The NAME directive has been defined two or more times. |
| | Program action | The assembler accepts the module name which has already been defined and ignores all the other NAME definitions. |

| | | |
|------|----------------|-----------------------------------------------------------------------------------------------------------------------|
| W704 | Message | Already declared EXTRN symbol ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name has already been declared with the EXTRN directive. |
| | User action | Limit the number of EXTRN declarations for a symbol to one per module. |
| W705 | Message | Already declared EXTBIT symbol ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name has already been declared with the EXTBIT directive. |
| | User action | Limit the number of EXTBIT declarations for a symbol to one per module. |
| W706 | Message | Missing END statement |
| | Cause | An END statement has not been described at the end of the source file. |
| | Program action | The assembler processes the source file by assuming that the END statement has been described at the end of the file. |
| W707 | Message | Illegal statement after END directive |
| | Cause | A statement other than a comment statement, blank, Tab, or LF code has been described following an END statement. |
| | Program action | The assembler ignores the statement (or characters) after the END statement. |
| W708 | Message | Already declared LOCAL symbol ' <u>symbol name</u> ' |
| | Cause | The symbol indicated by the displayed symbol name has already been declared with the LOCAL directive. |
| | User action | Limit the number of LOCAL declarations for a symbol to one per macro. |
| W709 | Message | Few count of actual parameter |
| | Cause | The number of actual parameters has been set less than the number of formal parameters. |
| | Program action | The assembler gives a null string to each actual parameter in excess of the number of actual parameters. |
| W710 | Message | Over count of actual parameter |
| | Cause | The number of actual parameters has been set more than the number of formal parameters. |
| | Program action | The assembler ignores the actual parameters in excess of the number of formal parameters. |
| W711 | Message | Too many errors to report |
| | Cause | Too many errors (six or more errors) exist in this line. |
| | Program action | The assembler does not output an error message for the 6th and subsequent errors and continues its processing. |

| | | |
|------|----------------|------------------------------------------------------------------------------------|
| W712 | Message | Insufficient cross-reference work area |
| | Cause | Memory capacity is not enough for the output processing of a cross-reference list. |
| | Program action | The assembler continues its processing without creating a cross-reference list. |

| | | |
|------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| A901 | Message | Can't open source file ' <u>filename</u> ' |
| | Cause | The source file indicated by the displayed filename cannot be opened. |
| A902 | Message | Can't open parameter file ' <u>filename</u> ' |
| | Cause | The parameter file indicated by the displayed filename cannot be opened. |
| A903 | Message | Can't open include file ' <u>filename</u> ' |
| | Cause | The INCLUDE file indicated by the displayed filename cannot be opened. |
| A904 | Message | Illegal include file ' <u>filename</u> ' |
| | Cause | Only a drive name, pathname, or device type filename has been specified as the filename of the INCLUDE file indicated by the displayed filename. |
| A905 | Message | Can't open overlay file ' <u>filename</u> ' |
| | Cause | The overlay file indicated by the displayed filename cannot be opened. |
| | User action | Check if the overlay file exists in the same directory as the executable (COM) file of the assembler. |
| A906 | Message | Illegal overlay file ' <u>filename</u> ' |
| | Cause | The overlay file indicated by the displayed filename contains incorrect data. |
| A907 | Message | Can't open object file ' <u>filename</u> ' |
| | Cause | The object module file indicated by the displayed filename cannot be opened. |
| | User action | Use a disk which has a free directory area. |
| A908 | Message | Can't open print file ' <u>filename</u> ' |
| | Cause | The assembly list file indicated by the displayed filename cannot be opened. |
| | User action | Use a disk which has a free directory area. |
| A909 | Message | Can't open error list file ' <u>filename</u> ' |
| | Cause | The error list file indicated by the displayed filename cannot be opened. |
| | User action | Use a disk which has a free directory area. |

| | | |
|------|-------------|---------------------------------------------------------------------------------------------------------|
| A910 | Message | Can't open temporary file ' <u>filename</u> ' |
| | Cause | The temporary file indicated by the displayed filename cannot be opened. |
| | User action | Use a disk which has a free directory area. |
| A911 | Message | System error |
| | Cause | A system error has occurred. |
| A912 | Message | Can't set control-C |
| | Cause | Control -C cannot be set to interrupt the assembler execution. |
| A913 | Message | Can't read source file ' <u>filename</u> ' |
| | Cause | A file I/O error has occurred while reading the source file indicated by the displayed filename. |
| A914 | Message | Can't read parameter file ' <u>filename</u> ' |
| | Cause | A file I/O error has occurred while reading the parameter file indicated by the displayed filename. |
| A915 | Message | Can't read include file ' <u>filename</u> ' |
| | Cause | A file I/O error has occurred while reading the INCLUDE file indicated by the displayed filename. |
| A916 | Message | Can't read overlay file ' <u>filename</u> ' |
| | Cause | A file I/O error has occurred while reading the overlay file indicated by the displayed filename. |
| A917 | Message | Can't write object file ' <u>filename</u> ' |
| | Cause | A file I/O error has occurred while writing the object module file indicated by the displayed filename. |
| | User action | Output the object module file to another disk or create a free area in the specified disk. |
| A918 | Message | Can't write print file ' <u>filename</u> ' |
| | Cause | A file I/O error has occurred while writing the assembly list file indicated by the displayed filename. |
| | User action | Output the assembly list file to another disk or create a free area in the specified disk. |
| A919 | Message | Can't write error list file ' <u>filename</u> ' |
| | Cause | A file I/O error has occurred while writing the error list file indicated by the displayed filename. |
| | User action | Output the error list file to another disk or create a free area in the specified disk. |

| | | |
|------|-------------|----------------------------------------------------------------------------------------------------------------|
| A920 | Message | Can't read/write temporary file ' <u>filename</u> ' |
| | Cause | A file I/O error has occurred while reading or writing the temporary file indicated by the displayed filename. |
| | User action | Output the temporary file to another disk or create a free area in the specified disk. |
| A921 | Message | Assembler internal error |
| | Cause | An error has occurred in the assembler itself. |
| | User action | Rerun the assembler. |
| A922 | Message | Insufficient memory in hostmachine |
| | Cause | The system has no sufficient memory to execute the assembler. |
| A923 | Message | Insufficient memory for macro in hostmachine |
| | Cause | The system has become short of the internal memory capacity during the processing of a macro. |
| | User action | Reduce the number of macrodefinitions. |

11.2 Linker Error Messages

Error messages related to the Linker

| | | |
|------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A101 | Message | 'filename' invalid input file (or made by different hostmachine) |
| | Cause | An attempt was made to input a file other than an object module file or to link an object module file created by a host machine which has no compatibility. |
| F102 | Message | Directive syntax error |
| | Cause | An error exists in the directive description. |
| A103 | Message | 'filename' Illegal processor type |
| | Cause | The target device for assembly or compilation is not of the processor type for this linker. |
| | User action | First, confirm that the object module file is correct and that the linker can handle the target device for assembly or compilation. Also confirm that the overlay file is of the correct version. (The linker obtains information peculiar to the target device by referring to part of the overlay file of the assembler.) |
| A104 | Message | 'filename' Different processor type from first input file ' <u>first input filename</u> ' |
| | Cause | The input object module file is different in the target device from that of the first input object module file. |
| W105 | Message | Library file ' <u>filename</u> ' has no public symbol |
| | Cause | No PUBLIC symbol exists in the library file indicated by the displayed filename. For this reason, the object modules in the library file will not be linked. |
| A106 | Message | Can't create temporary file ' <u>filename</u> ' |
| | Cause | The temporary file indicated by the displayed filename cannot be created. |
| F107 | Message | Name ' <u>name</u> ' in directive has already defined |
| | Cause | An attempt was made to define a reserved word or an already defined name as a memory area name in the directive. (The reserved word, memory space name, or memory area name indicated by the displayed name has already been defined.) |
| F108 | Message | Overlapped memory area ' <u>memory area1</u> ' and ' <u>memory area2</u> ' |
| | Cause | In the MEMORY directive, addresses overlap between the two memory areas. |

| | | |
|------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F109 | Message | Memory area ' <u>memory area name</u> ' too long name (up to 31 characters) |
| | Cause | The memory area name indicated by the displayed memory area name and specified in the directive is too long. The memory area name length is limited to 32 characters in the directive. |
| F110 | Message | Memory area ' <u>memory area name</u> ' already defined |
| | Cause | The memory area name indicated by the displayed memory area name and specified in the MEMORY directive has already been defined. |
| F111 | Message | Memory area ' <u>memory area name</u> ' redefinition out of range |
| | Cause | The memory area indicated by the displayed memory area name has been specified in the MEMORY directive by exceeding the range that can be re-defined for the memory area. |
| F112 | Message | Segment ' <u>segment name</u> ' wrong allocation type |
| | Cause | An error exists in the location type specification of the segment indicated by the displayed segment name in the MERGE directive. |
| F113 | Message | linker internal error |
| | Cause | An error has occurred in the Linker itself. |
| | User action | Contact the NEC or NEC's specified agent. |
| F114 | Message | Illegal number |
| | Cause | An error exists in the value of a directive. |
| F115 | Message | Too large value (up to 65535/0FFFFH) |
| | Cause | A value exceeding 65535 (0FFFFH) has been described in a directive. |
| F116 | Message | Memory area ' <u>memory area name</u> ' definition out of range |
| | Cause | In the MEMORY directive, the sum of the start address of the displayed memory area and the memory size exceeded 65535 (0FFFFH). |

| | | |
|------|---------|--------------------------------------------------------------------------------------------------------------------|
| F201 | Message | Multiple segment definition ' <u>segment name</u> ' in merge directive |
| | Cause | The segment indicated by the displayed segment name and specified in the MERGE directive has already been defined. |

| | | |
|------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F202 | Message | Segment type mismatch ' <u>segment1</u> ' in file ' <u>filename</u> ' - ignored |
| | Cause | A segment which has the same name as that indicated by the displayed segment name but is different in segment type and relocation attribute from the displayed segment exists in the file indicated by the displayed filename. |
| F203 | Message | Segment ' <u>segment name</u> ' unknown segment type |
| | Cause | An error exists in the segment information of the input object module file. (An error exists in the merge type specification of the output segment.) |
| F204 | Message | Memory area/space ' <u>name</u> ' not defined |
| | Cause | The memory area or memory space indicated by the displayed name and specified in the MERGE directive has not been defined. |
| F205 | Message | Name ' <u>name</u> ' in directive has bad attribute |
| | Cause | A name which cannot be specified as a segment name, memory area name, or memory space name has been described in the directive. (For example, a memory space name has been specified erroneously in place of a memory area name.) |
| F206 | Message | Segment ' <u>segment name</u> ' can't allocate to memory - ignored |
| | Cause | The segment indicated by the displayed segment name cannot be allocated to any memory area. (A sufficient memory area is not available for locating the segment.) |
| F207 | Message | Segment ' <u>segment name</u> ' has illegal segment type |
| | Cause | The segment type information of the segment indicated by the displayed segment name is illegal. |
| | User action | Re-assemble or re-compile the source program which contains this segment. Check the MERGE directive for correct description. |
| F208 | Message | Segment ' <u>segment name</u> ' may not change attribute |
| | Cause | An attempt was made with a link directive to change the merge type of the displayed segment whose relocation attribute was specified as "AT xxxxH" at assembly time or which has been created with the ORG directive. |
| | User action | For a segment whose merge type is to be specified at linkage time, do not specify any location address for the segment at assembly time. |

| | | |
|------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F209 | Message | Segment ' <u>segment name</u> ' may not change arrangement |
| | Cause | An attempt was made with a link directive to change the location address of the displayed segment whose relocation attribute was specified as "AT xxxxH" at assembly time or which has been created with the ORG directive. |
| | User action | For a segment whose merge type is to be specified at linkage time, do not specify any location address for the segment at assembly time. |
| F210 | Message | Segment ' <u>segment name</u> ' does not exist - ignored |
| | Cause | The displayed segment specified in the directive does not exist. |

| | | |
|------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F301 | Message | Relocatable object code address out of range (file ' <u>filename</u> ', segment ' <u>segment name</u> ', address xxxxH, type ' <u>addressing type</u> ') (See Note 1 below.) |
| | Cause | Information to correct the relocatable object code contained in the input object module file has been output to an address in which no object code exists. (The relocation entry address is outside the range of the origin data.) |
| F302 | Message | Illegal symbol index in line number (file ' <u>filename</u> ', segment ' <u>segment name</u> ' (See Note 1 below.) |
| | Cause | An error exists in the line number information for debugging contained in the input object module file and thus the symbol information is not properly referenced. The line number index does not correspond with the symbol index. |
| F303 | Message | Can't find symbol index in relocatable object code (file ' <u>filename</u> ', segment ' <u>segment name</u> ', address xxxxH, type ' <u>addressing type</u> ') (See Note 1 below.) |
| | Cause | An error exists in the information to correct the relocatable object code contained in the input object module file and thus the symbol information is not properly referenced. The relocation entry index does not correspond with the symbol index. |
| | User action | Confirm that symbols and variables have been referenced in the correct way. |

Note: The address to be displayed as "address xxxxH" is an absolute address after the segment location.

| | | |
|------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F304 | Message | Operand out of range (segment ' <u>segment name</u> ', address xxxxH, type ' <u>addressing type</u> ') (See Note 1 below.) |
| | Cause | The operand value used for resolving relocatable object codes exceeded the range of the operand value applicable to the instruction. |
| | User action | Describe the source program so that the operand value used for resolving relocatable object codes falls within the range of the operand value determined for each addressing type. |
| F305 | Message | Even value expected (segment ' <u>segment name</u> ', address xxxxH, type ' <u>addressing type</u> ') (See Note 1 below.) |
| | Cause | The operand value used for resolving the relocatable object codes of the "callt" or "saddrp" addressing type has become an odd value. The operand value for the "callt" or "saddrp" addressing type must be an even value. |
| F306 | Message | A multiple of 4 value expected (segment ' <u>segment name</u> ', address xxxxH, type ' <u>addressing type</u> ') (See Note 1 below.) |
| | Cause | The operand value used for resolving the relocatable object codes of the "dsaddr" addressing type is not a multiple of 4. |

Note: The address to be displayed as "address xxxxH" is an absolute address after the segment location.

| | | |
|------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A401 | Message | <u>'filename'</u> Bad symbol table |
| | Cause | The symbol information of the input object module file is incorrect. The symbol entry of the input file must begin with ".file". |
| A402 | Message | File ' <u>filename</u> ' has no string table for symbol |
| | Cause | The symbol information of the input object module file is incorrect. |
| | User action | Re-assemble or re-compile the source program. This error may be avoided by limiting the number of characters that can be recognized as a symbol to eight characters with the assembler and to seven characters with the linker. |
| A403 | Message | Symbol ' <u>symbol name</u> ' unmatched type in file ' <u>filename1</u> ' First defined in file ' <u>filename2</u> ' |
| | Cause | A difference exists in the type between the same named externally defined or referenced symbol in the file 1 and that in the file 2. |

| | | |
|------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F404 | Message | Multiple Symbol definition ' <u>symbol name</u> ' in file ' <u>filename1</u> ' First defined in file ' <u>filename2</u> ' |
| | Cause | The PUBLIC symbol defined in the object module file 1 has already been declared as PUBLIC in the object module file 2. (Duplicate PUBLIC symbol declarations) |
| F405 | Message | Undefined symbol ' <u>symbol name</u> ' in file ' <u>filename</u> ' |
| | Cause | The symbol indicated by the displayed symbol name and declared as EXTRN in the file indicated by the displayed filename has not been declared as PUBLIC in another file. |
| W406 | Message | Stack area less than 10 bytes |
| | Cause | The size of the reserved stack area is less than 10 bytes. (The size of the stack area reserved in the memory area specified with the -S option is less than 10 bytes.) |
| W407 | Message | Can't allocate stack area |
| | Cause | The memory area has no free area for stack area reservation. (No stack area can be reserved in the memory area specified with the -S option.) |

| | | |
|------|---------|--------------------------------------------------------------------------|
| A501 | Message | Insufficient memory in hostmachine |
| | Cause | The system has no sufficient memory capacity for the program to operate. |

| | | |
|------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| A901 | Message | Can't open overlay file ' <u>filename</u> ' |
| | Cause | The overlay file indicated by the displayed filename cannot be opened. |
| | User action | Confirm that the overlay file exists in the correct directory (i.e., the same directory as the executable program with MS-DOS version 3.10). |
| A902 | Message | file ' <u>filename</u> ' file not found |
| | Cause | The library file indicated by the displayed filename cannot be opened. |
| A903 | Message | Can't read input file ' <u>filename</u> ' |
| | Cause | The object module file specified as an input file and indicated by the displayed filename cannot be read. |

| | | |
|------|-------------|--------------------------------------------------------------------------------------------|
| A904 | Message | Can't open output file ' <u>filename</u> ' |
| | Cause | The output file indicated by the displayed filename cannot be opened. |
| | User action | Check the disk for output file creation for free area or proper file medium condition. |
| A905 | Message | Can't create temporary file ' <u>filename</u> ' |
| | Cause | The temporary file for symbol entry indicated by the displayed filename cannot be created. |
| | User action | Check the disk for temporary file creation for free area or proper file medium condition. |
| A906 | Message | Can't write map file ' <u>filename</u> ' |
| | Cause | No data can be written into the link list file indicated by the displayed filename. |
| | User action | Check the disk for link list file creation for free area or proper file medium condition. |
| A907 | Message | Can't write output file ' <u>filename</u> ' |
| | Cause | No data can be written into the load module file indicated by the displayed filename. |
| | User action | Check the disk for output file creation for free area or proper file medium condition. |
| A908 | Message | Can't access temporary file ' <u>filename</u> ' |
| | Cause | No data can be written into the temporary file indicated by the displayed filename. |
| | User action | Check the disk for temporary file creation for free area or proper file medium condition. |

11.3 Object Converter Error Messages

Error messages related to the Object Converter

| | | |
|------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A100 | Message | ' <u>filename</u> ' Illegal processor type |
| | Cause | The target device for assembly or compilation is not of the processor type for this program. |
| | User action | First, confirm that the load module file is correct and that the object converter can handle the target device for assembly or compilation. Also confirm that the overlay file is of the correct version. |
| A101 | Message | ' <u>filename</u> ' invalid input file (or made by different hostmachine) |
| | Cause | An attempt was made to input a file other than a load module file or to convert a load module file created by a host machine which has no compatibility. |
| A103 | Message | Symbol ' <u>symbol name</u> ' Illegal attribute |
| | Cause | An error exists in the attribute value of the symbol indicated by the displayed symbol name in the input file. |
| A104 | Message | ' <u>filename</u> ' Illegal input file - not linked |
| | Cause | An attempt was made to input the file not linked (i.e., object module file) and indicated by the displayed filename. |
| A105 | Message | Insufficient memory in hostmachine |
| | Cause | The system has no sufficient memory capacity for the program to operate. |
| A106 | Message | Illegal symbol table |
| | Cause | An error exists in the symbol table of the input load module file. |

| | | |
|------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F200 | Message | Undefined symbol ' <u>symbol name</u> ' |
| | Cause | An address for the symbol indicated by the displayed symbol name has not been resolved. |
| | User action | Define the value of this symbol. If this symbol has been referenced as an external referenced symbol but has not been defined externally, define the symbol externally in the module in which the value of this symbol has been defined. |
| F201 | Message | Out of address range |
| | Cause | The address(es) of object code(s) in the load module file are out of the permissible address range. |

| | | |
|------|---------|----------------------------------------------------------------------------|
| W300 | Message | xxxxH - yyyyH overlapped |
| | Cause | Object codes for addresses xxxxH to yyyyH have been output in duplication. |

| | | |
|------|---------|------------------------------------------------------------------------------------------|
| A900 | Message | Can't open file ' <u>filename</u> ' |
| | Cause | The file indicated by the displayed filename cannot be opened. |
| A901 | Message | Can't close file ' <u>filename</u> ' |
| | Cause | The file indicated by the displayed filename cannot be closed. |
| A902 | Message | Can't read file ' <u>filename</u> ' |
| | Cause | The file indicated by the displayed filename cannot be read properly. |
| A903 | Message | Can't access file ' <u>filename</u> ' |
| | Cause | The file indicated by the displayed filename cannot be read- or write-accessed properly. |
| A904 | Message | Can't write file ' <u>filename</u> ' |
| | Cause | Data cannot be properly written into the file indicated by the displayed filename. |

11.4 Librarian Error Messages

Error Messages related to the Librarian

| | | |
|------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A001 | Message | Missing input file |
| | Cause | Only options have been specified and no input file has been specified. |
| A002 | Message | Too many input files |
| | Cause | The total number of input files has been specified by exceeding the limit value. |
| A003 | Message | Unrecognized string ' <u>whatever specified</u> ' |
| | Cause | A character string other than options has been specified in the command line in the Conversational mode. |
| A004 | Message | Illegal file name ' <u>filename</u> ' |
| | Cause | The filename indicated by the displayed filename contains character(s) not recognized by the OS or exceeded the limit value for the number of characters that can be described as a filename. |
| A005 | Message | Illegal file specification ' <u>filename</u> ' |
| | Cause | The displayed filename is illegal as a filename. |
| A006 | Message | File not found ' <u>filename</u> ' |
| | Cause | The specified input file indicated by the displayed filename does not exist. |
| A007 | Message | Input file specification overlapped ' <u>filename</u> ' |
| | Cause | The input file indicated by the displayed file has been specified in duplication. |
| A008 | Message | File specification conflicted ' <u>filename</u> ' |
| | Cause | The input/output file indicated by the displayed filename has been specified in duplication. |
| A009 | Message | Unable to make file ' <u>filename</u> ' |
| | Cause | The specified file indicated by the displayed filename cannot be created. |
| A010 | Message | Directory not found ' <u>filename</u> ' |
| | Cause | The specification of the output filename indicated by the displayed filename contains a non-existing drive or directory. |
| A011 | Message | Illegal path ' <u>filename</u> ' |
| | Cause | In the specification of an option requiring a pathname as its parameter, other than a pathname has been specified for the file indicated by the displayed filename. |

| | | |
|------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A012 | Message | Missing parameter ' <u>option</u> ' |
| | Cause | The required parameter has not been specified for the option indicated by the displayed option. |
| A013 | Message | Parameter not needed ' <u>option</u> ' |
| | Cause | An unwanted parameter has been specified for the option indicated by the displayed option. |
| A014 | Message | Out of range ' <u>option</u> ' |
| | Cause | A value outside the range has been specified for the option indicated by the displayed option. |
| A015 | Message | Parameter is too long ' <u>option</u> ' |
| | Cause | The parameter of the option indicated by the displayed option has been specified by exceeding the limit value for the number of characters that can be described. |
| A016 | Message | Illegal parameter ' <u>option</u> ' |
| | Cause | A syntax error exists in the parameter description of the option indicated by the displayed option. |
| A017 | Message | Too many parameters ' <u>option</u> ' |
| | Cause | The total number of parameters specified for the option indicated by the displayed option exceeded the limit value. |
| A018 | Message | Option is not recognized ' <u>option</u> ' |
| | Cause | The option indicated by the displayed option is not recognized by the librarian. |
| A019 | Message | Parameter file nested |
| | Cause | The -F option has been specified in the parameter file. |
| A020 | Message | Parameter file read error ' <u>filename</u> ' |
| | Cause | An error has occurred while reading the parameter file indicated by the displayed filename. |
| A021 | Message | Memory allocation failed |
| | Cause | Memory cannot be allocated. |

| | | |
|------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| A100 | Message | Internal error |
| | Cause | An error has occurred in the librarian itself. |
| F101 | Message | Invalid sub command |
| | Cause | The subcommand name is incorrect. |
| F102 | Message | Invalid syntax |
| | Cause | An error exists in the parameter specification of the subcommand. |
| F103 | Message | Illegal input file - different target chip (file: <u>filename</u>) |
| | Cause | An error exists in the target device specification of the input object module file indicated by the displayed filename. |
| F104 | Message | Illegal library file - different target chip (file: <u>filename</u>) |
| | Cause | An error exists in the target device specification of the specified library file indicated by the displayed filename. |
| F105 | Message | Module not found (module: <u>module name</u>) |
| | Cause | The specified module indicated by the displayed module name does not exist in the library file. |
| F106 | Message | Module already exists (module: <u>module name</u>) |
| | Cause | The same module as that indicated by the displayed module name already exists in the update library file or another input file. |
| F107 | Message | Master library file is not specified |
| | Cause | Replacement with "." has been specified when an update library file has not been specified by the previous update operation. |
| F108 | Message | Multiple transaction file (file: <u>filename</u>) |
| | Cause | The input object module file indicated by the displayed filename has been specified in duplication. |
| F109 | Message | Public symbol already exists (symbol: <u>symbol name</u>) |
| | Cause | The same named externally defined symbol as that indicated by the displayed symbol name already exists in the update library file or another input file. |
| F110 | Message | File specification conflicted (file: <u>filename</u>) |
| | Cause | The same input filename as the output filename indicated by the displayed filename has been specified. |

| | | |
|------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| F111 | Message | Illegal file format (file: <u>filename</u>) |
| | Cause | The format of the update library file or another input file indicated by the displayed filename is incorrect. |
| F112 | Message | Library file not found (file: <u>filename</u>) |
| | Cause | The specified library file indicated by the displayed filename cannot be located. |
| F113 | Message | Object module file not found (file: <u>filename</u>) |
| | Cause | The specified object module file indicated by the displayed filename cannot be located. |
| F114 | Message | No free space for temporary file |
| | Cause | A sufficient free area for temporary file creation is not available on the disk. |
| F115 | Message | Not enough memory |
| | Cause | A sufficient memory space for the program to operate cannot be secured. |
| F116 | Message | Subcommand Buffer full |
| | Cause | The continuation line length of the subcommand exceeded the limit value (128 characters x 15 lines). The subcommand line length in the subcommand file exceeded the limit value (128 characters). |

| | | |
|------|---------|--------------------------------------------------------------------------------------------------------------------------|
| A901 | Message | File open error (file: <u>filename</u>) |
| | Cause | A file open error has occurred in the file indicated by the displayed filename or the system is not operating properly. |
| A902 | Message | File read error (file: <u>filename</u>) |
| | Cause | A file read error has occurred in the file indicated by the displayed filename or the system is not operating properly. |
| A903 | Message | File write error (file: <u>filename</u>) |
| | Cause | A file write error has occurred in the file indicated by the displayed filename or the system is not operating properly. |
| A904 | Message | File seek error (file: <u>filename</u>) |
| | Cause | A file seek error has occurred in the file indicated by the displayed filename or the system is not operating properly. |
| A905 | Message | File close error (file: <u>filename</u>) |
| | Cause | A file close error has occurred in the file indicated by the displayed filename or the system is not operating properly. |

11.5 List Converter Error Messages

Error Messages related to the List Converter

| | | |
|------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| A101 | Message | File is not 78K/xxx ' <u>filename</u> ' |
| | Cause | The input file indicated by the displayed filename is not for the 78K/xxx series. |
| W101 | Message | Load module file is older than object module file ' <u>load module filename, object module filename</u> ' |
| | Cause | The load module file older than the object module file has been specified. |
| A102 | Message | Load module file is not executable ' <u>filename</u> ' |
| | Cause | An attempt was made to input a file other than a load module file or to convert a load module file created by a host machine which has no compatibility. |
| W102 | Message | Load module file is older than assemble list file ' <u>load module filename, assembly list filename</u> ' |
| | Cause | The load module file older than the assembly list file has been specified. |
| A103 | Message | Load module file has relocation data ' <u>filename</u> ' |
| | Cause | Addresses for relocatable segments in the load module file indicated by the displayed filename have not been resolved. |
| W103 | Message | Assemble list has error statement ' <u>filename</u> ' |
| | Cause | The assembly list file indicated by the displayed filename contains an error line. |
| A104 | Message | Object module file is executable ' <u>filename</u> ' |
| | Cause | The object module file indicated by the displayed filename is executable. |
| W104 | Message | Segment name is not found in assemble list file ' <u>segment name</u> ' |
| | Cause | The segment indicated by the displayed segment name in the object module file cannot be found in the assembly list file. |
| A105 | Message | Segment name is not found in load module file ' <u>segment name</u> ' |
| | Cause | The segment indicated by the displayed segment name in the object module file cannot be found in the load module file. |

| | | |
|------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| W105 | Message | Segment data length is different ' <u>segment name</u> ' |
| | Cause | The data length of the segment indicated by the segment name in the assembly list file differs from that of the segment in the object module file. |
| | Program action | The list converter executes its processing by ignoring excess segment data. |
| A106 | Message | Segment name is not found in object module file ' <u>segment name</u> ' |
| | Cause | The segment indicated by the displayed segment name in the assembly list file cannot be found in the object module file. |
| A107 | Message | Not enough memory |
| | Cause | A memory area for work is insufficient. |
| A108 | Message | Load module file has no symbol data ' <u>load module name</u> ' |
| | Cause | Because the -NG option has been specified in the linker, no symbol information has been output to the load module file indicated by the displayed load module name. |

| | | |
|------|---------|----------------------------------------------------------------------------------|
| A901 | Message | File open error has occurred ' <u>filename</u> ' |
| | Cause | A file open error has occurred in the file indicated by the displayed filename. |
| A902 | Message | File read error has occurred ' <u>filename</u> ' |
| | Cause | A file read error has occurred in the file indicated by the displayed filename. |
| A903 | Message | File write error has occurred ' <u>filename</u> ' |
| | Cause | A file write error has occurred in the file indicated by the displayed filename. |
| A904 | Message | File seek error has occurred |
| | Cause | A file seek error has occurred. |
| A999 | Message | Internal error |
| | Cause | An error has occurred in the list converter itself. |

APPENDIXES

APPENDIX A. SAMPLE PROGRAM

A.1 Source Lists

(1) 78K3MAIN.ASM

```

$      PC(310)

      NAME      SAMPM
;*****
;*
;*      HEX -> ASCII Conversion Program      *
;*
;*      main-routine      *
;*
;*****
      PUBLIC    MAIN, START
      EXTRN     CONVAH

DATA    DSEG      AT 0FE20H
HDTSA:  DS        1
STASC:  DS        2

CODE    CSEG      AT 0H
MAIN:   DW        START

START:  CSEG
        MOV       RFM, #00
        MOVW      SP, #0FE80H
        MOV       MM, #00
        MOV       STBC, #08H

        MOV       HDTSA, #1AH
        MOVW      HL, #HDTSA      ;set hex 2-code data in HL register

        CALL      !CONVAH        ;convert ASCII <- HEX
                                   ;output BC-register <- ASCII code
                                   ;set DE <- store ASCII code table

        MOVW      DE, #STASC
        MOV       A, B
        MOV       [DE+], A
        MOV       A, C
        MOV       [DE+], A

        BR        $$

      END

```


(2) 78K3SUB.ASM

```

$      PROCESSOR(310)

      NAME      SAMPS
;*****
;*
;*  HEX -> ASCII Conversion Program
;*
;*          sub-routine
;*
;*  input condition : (HL) <- hex 2 code
;*
;*  output condition : BC-register <-ASCII 2 code
;*
;*****

      PUBLIC  CONVAH

CONVAH: CSEG
      MOV     A,#0
      ROL4    [HL]          ;hex upper code load
      CALL    !SASC
      MOV     B,A           ;store result

      MOV     A,#0
      ROL4    [HL]          ;hex lower code load
      CALL    !SASC
      MOV     C,A           ;store result

      RET

;*****
;* subroutine  convert ASCII code
;*
;*  input  Acc (lower 4bits) <- hex code
;*
;*  output Acc          <- ASCII code
;*
;*****

SASC:  CMP     A,#0AH        ;check hex code > 9
      BC      $SASC1
      ADD     A,#07H        ;bias(+7)
SASC1: ADD     A,#30H        ;bias(+30)
      RET

      END

```

A.2 Execution Examples

A>ra78k3 -c310 78k3main.asm -g -kx -lw80

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start

Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

A>ra78k3 -c310 78k3sub.asm -g -kx -lw80

uCOM-78K/III Assembler Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Pass1 Start

Pass2 Start

Assembly complete, 0 error(s) and 0 warning(s) found.

A>lk78k3 78k3main.rel 78k3sub.rel -g -o78k3.lnk -p78k3.map -kp -kl

uCOM-78K/III Linker Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Link complete, 0 error(s) and 0 warning(s) found.

A>oc78k3 78k3.lnk

uCOM-78K/III Object Converter Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx

Object Converter complete, 0 error(s) and 0 warning(s) found.

A>lb78k3

uCOM-78K/III Librarian Vx.xx [xx xxx xx]
Copyright (C) Corporation xxxx USxxxxxxxxxx
*create 78k3.lib 78k3.rel
*list -p -o78k3.lst 78k3.lib
*exit

A>lcnv78k3 78k3main -l78k3.lnk

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
Copyright (C) NEC Corporation xxxx USxxxxxxxxxx

Pass1: start...

Pass2: start...

Conversion complete.

A.3 Output Lists

A.3.1 Assembly Lists

(1) Assembly list of 78K3MAIN.ASM

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c310 78k3main.asm -g -kx -lw80

Para-file:

In-file: 78K3MAIN.ASM

Obj-file: 78K3MAIN.REL

Prn-file: 78K3MAIN.PRN

Assemble list

| ALNO | STNO | ADRS | OBJECT | M I | SOURCE STATEMENT |
|------|------|---------------|--------|-----|--------------------------------------|
| 1 | 1 | | | | \$ PC(310) |
| 2 | 2 | | | | |
| 3 | 3 | | | | NAME SAMPM |
| 4 | 4 | | | | :***** |
| 5 | 5 | | | | :* |
| 6 | 6 | | | | :* HEX -> ASCII Conversion Program * |
| 7 | 7 | | | | :* |
| 8 | 8 | | | | :* main-routine * |
| 9 | 9 | | | | :* |
| 10 | 10 | | | | :***** |
| 11 | 11 | | | | |
| 12 | 12 | | | | PUBLIC MAIN,START |
| 13 | 13 | | | | EXTRN CONVAH |
| 14 | 14 | | | | |
| 15 | 15 | ---- | | | DATA DSEG AT 0FE20H |
| 16 | 16 | FE20 | | | HDTSA: DS 1 |
| 17 | 17 | FE21 | | | STASC: DS 2 |
| 18 | 18 | | | | |
| 19 | 19 | ---- | | | CODE CSEG AT 0H |
| 20 | 20 | 0000 R0000 | | | MAIN: DW START |
| 21 | 21 | | | | |
| 22 | 22 | ---- | | | CSEG |
| 23 | 23 | 0000 2B4100 | | | START: MOV RFM,#00 |
| 24 | 24 | 0003 0BFC80FE | | | MOVW SP,#0FE80H |
| 25 | 25 | 0007 2B4000 | | | MOV MM,#00 |
| 26 | 26 | 000A 0944F708 | | | MOV STBC,#08H |
| 27 | 27 | | | | |
| 28 | 28 | 000E 3A201A | | | MOV HDTSA,#1AH |
| 29 | 29 | 0011 6720FE | | | MOVW HL,#HDTSA |
| 30 | 30 | | | | |
| 31 | 31 | 0014 R280000 | | | CALL !CONVAH |
| 32 | 32 | | | | |
| 33 | 33 | 0017 6521FE | | | MOVW DE,#STASC |
| 34 | 34 | 001A D3 | | | MOV A,B |
| 35 | 35 | 001B 50 | | | MOV [DE+],A |
| 36 | 36 | 001C D2 | | | MOV A,C |
| 37 | 37 | 001D 50 | | | MOV [DE+],A |
| 38 | 38 | | | | |
| 39 | 39 | 001E 14FE | | | BR \$\$ |
| 40 | 40 | | | | |
| 41 | 41 | | | | END |

Segment informations:

| ADRS | LEN | NAME |
|------|-------|-------|
| FE20 | 0003H | DATA |
| 0000 | 0002H | CODE |
| 0000 | 0020H | ?CSEG |

Target chip:uPD78310

Assembly complete, 0 error(s) and 0 warning(s) found. (0)

Note: In this assembly list, the comment statement section has
been deleted after the output of the list.

(2) Assembly list of 78K3SUB.ASM

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c310 78k3sub.asm -g -kx -lw80

Para-file:

In-file: 78K3SUB.ASM

Obj-file: 78K3SUB.REL

Prn-file: 78K3SUB.PRN

Assemble list

| ALNO | STNO | ADRS | OBJECT | M I | SOURCE STATEMENT |
|------|------|------|---------|-----|----------------------------------------------------|
| 1 | 1 | | | | \$ PROCESSOR(310) |
| 2 | 2 | | | | |
| 3 | 3 | | | | NAME SAMPS |
| 4 | 4 | | | | ***** |
| 5 | 5 | | | | ; |
| 6 | 6 | | | | ;* HEX -> ASCII Conversion Program * |
| 7 | 7 | | | | ; |
| 8 | 8 | | | | ;* sub-routine * |
| 9 | 9 | | | | ; |
| 10 | 10 | | | | ;* input condition : (HL) <- hex 2 code * |
| 11 | 11 | | | | ; |
| 12 | 12 | | | | ;* output condition : BC-register<- ASCII 2 code * |
| 13 | 13 | | | | ; |
| 14 | 14 | | | | ***** |
| 15 | 15 | | | | |
| 16 | 16 | | | | PUBLIC CONVAH |
| 17 | 17 | | | | |
| 18 | 18 | ---- | | | CSEG |
| 19 | 19 | 0000 | B900 | | CONVAH: MOV A,#0 |
| 20 | 20 | 0002 | 059F | | ROL4 [HL] |
| 21 | 21 | 0004 | R281300 | | CALL !SASC |
| 22 | 22 | 0007 | 2431 | | MOV B,A |
| 23 | 23 | | | | |
| 24 | 24 | 0009 | B900 | | MOV A,#0 |
| 25 | 25 | 000B | 059F | | ROL4 [HL] |
| 26 | 26 | 000D | R281300 | | CALL !SASC |
| 27 | 27 | 0010 | 2421 | | MOV C,A |
| 28 | 28 | | | | |
| 29 | 29 | 0012 | 56 | | RET |
| 30 | 30 | | | | |
| 31 | 31 | | | | ***** |
| 32 | 32 | | | | ;* subroutine convert ASCII code * |
| 33 | 33 | | | | ;* input Acc (lower 4bits)<- hex code * |
| 34 | 34 | | | | ;* output Acc <- ASCII code * |
| 35 | 35 | | | | ***** |
| 36 | 36 | | | | |
| 37 | 37 | 0013 | AFOA | | SASC: CMP A,#0AH |
| 38 | 38 | 0015 | 8302 | | BC \$\$SASC1 |
| 39 | 39 | 0017 | A807 | | ADD A,#07H |
| 40 | 40 | 0019 | A830 | | SASC1: ADD A,#30H |
| 41 | 41 | 001B | 56 | | RET |
| 42 | 42 | | | | |
| 43 | 43 | | | | END |

Segment informations:

ADRS LEN NAME
0000 001CH ?CSEG

Target chip:uPD78310

Assembly complete, 0 error(s) and 0 warning(s) found. (0)

Note: In this assembly list, the comment statement section has been deleted after the output of the list.

A.3.2 Symbol Lists

(1) Symbol list of 78K3MAIN.ASM

Symbol Table List

| VALUE | ATTR | RTYP | NAME | VALUE | ATTR | RTYP | NAME |
|-------|------|------|--------|-------|------|------|-------|
| | CSEG | | ?CSEG | | CSEG | | CODE |
| ----H | | EXT | CONVAH | | DSEG | | DATA |
| FE20H | ADDR | | HDTSA | 0H | ADDR | PUB | MAIN |
| | MOD | | SAMPM | 0H | ADDR | PUB | START |
| FE21H | ADDR | | STASC | | | | |

(2) Symbol list of 78K3SUB.ASM

Symbol Table List

| VALUE | ATTR | RTYP | NAME | VALUE | ATTR | RTYP | NAME |
|-------|------|------|-------|-------|------|------|--------|
| | CSEG | | ?CSEG | 0H | ADDR | PUB | CONVAH |
| | MOD | | SAMPS | 13H | ADDR | | SASC |
| 19H | ADDR | | SASC1 | | | | |

A.3.3 Cross-reference Lists

(1) Cross-reference list of 78K3MAIN.ASM

| Cross-Reference List | | | | | | | |
|----------------------|-------|---|------|------|---------|-------|--------|
| NAME | VALUE | R | ATTR | RTYP | SEGNAME | XREFS | |
| ?CSEG | | | CSEG | | ?CSEG | 22# | |
| CODE | | | CSEG | | CODE | 19# | |
| CONVAH | ----H | E | | EXT | | 130 | 31 |
| DATA | | | DSEG | | DATA | 15# | |
| HDTSA | FE20H | | ADDR | | DATA | 16# | 28 29 |
| MAIN | 0H | | ADDR | PUB | CODE | 120 | 20# |
| SAMPM | | | MOD | | | 3# | |
| START | 0H | R | ADDR | PUB | ?CSEG | 120 | 20 23# |
| STASC | FE21H | | ADDR | | DATA | 17# | 33 |

(2) Cross-reference list of 78K3SUB.ASM

| Cross-Reference List | | | | | | | |
|----------------------|-------|---|------|------|---------|-------|--------|
| NAME | VALUE | R | ATTR | RTYP | SEGNAME | XREFS | |
| ?CSEG | | | CSEG | | ?CSEG | 18# | |
| CONVAH | 0H | R | ADDR | PUB | ?CSEG | 160 | 19# |
| SAMPS | | | MOD | | | 3# | |
| SASC | 13H | R | ADDR | | ?CSEG | 21 | 26 37# |
| SASC1 | 19H | R | ADDR | | ?CSEG | 38 | 40# |

A.3.4 Map List

uCOM-78K/III LINKER Vx.xx

Date:xx xxx xxxx Page: 1

Command: 78k3main.rel 78k3sub.rel -g -o78k3.lnk -p78k3.map -kp -kl
 Para-file:
 Out-file: 78K3.LNK
 Map-file: 78K3.MAP
 Direc-file:
 Directive:

*** Link information ***

3 output segment(s)
 40H byte(s) real data
 17 symbol(s) defined

*** Memory map ***

SPACE=REGULAR

MEMORY=ROM

BASE ADDRESS=0000H SIZE=FE00H

| OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS | SIZE | |
|-------------------|------------------|-----------------|-----------------|-------|---------|
| CODE | | | 0000H | 0002H | CSEG AT |
| | CODE | SAMPM | 0000H | 0002H | |
| ?CSEG | | | 0002H | 003CH | CSEG |
| | ?CSEG | SAMPM | 0002H | 0020H | |
| | ?CSEG | SAMPS | 0022H | 001CH | |
| | | | 003EH | FDC2H | |

* gap *

MEMORY=RAM

BASE ADDRESS=FE00H SIZE=0100H

| OUTPUT SEGMENT | INPUT SEGMENT | INPUT MODULE | BASE ADDRESS | SIZE | |
|-------------------|------------------|-----------------|-----------------|-------|---------|
| DATA | | | FE00H | 0020H | |
| | DATA | SAMPM | FE20H | 0003H | DSEG AT |
| | | | FE20H | 0003H | |
| | | | FE23H | 00DDH | |

* gap *

A.3.5 PUBLIC Symbol List

*** Public symbol list ***

| MODULE | ATTR | VALUE | NAME |
|--------|------|-------|--------|
| SAMPM | ADDR | 0000H | MAIN |
| SAMPM | ADDR | 0002H | START |
| SAMPS | ADDR | 0022H | CONVAH |

A.3.6 Local Symbol List

*** Local symbol list ***

| MODULE | ATTR | VALUE | NAME |
|--------|------|-------|-------|
| SAMPM | MOD | | SAMPM |
| SAMPM | DSEG | | DATA |
| SAMPM | ADDR | FE20H | HDTSA |
| SAMPM | ADDR | FE21H | STASC |
| SAMPM | CSEG | | CODE |
| SAMPM | CSEG | | ?CSEG |
| SAMPS | MOD | | SAMPS |
| SAMPS | CSEG | | ?CSEG |
| SAMPS | ADDR | 0035H | SASC |
| SAMPS | ADDR | 003BH | SASC1 |

A.3.7 Library Information Output List

uCOM-78K/III librarian Vx.xx DATE : xx xxx xx

PAGE 1

LIB-FILE NAME : 78K3.LIB (xx xxx xx)

0001 78K3SUB.REL (xx xxx xx)

CONVAH

NUMBER OF PUBLIC SYMBOLS : 1

A.3.8 Absolute Assembly List

uCOM-78K/III Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c310 78k3main.asm -g -kx -lw80

Para-file:

In-file: 78K3MAIN.ASM

Obj-file: 78K3MAIN.REL

Prn-file: 78K3MAIN.PRN

Assemble list

| ALNO | STNO | ADRS | OBJECT | M I | SOURCE STATEMENT |
|------|------|---------------|--------|-----|---------------------------------|
| 1 | 1 | | | | \$ PC(310) |
| 2 | 2 | | | | |
| 3 | 3 | | | | NAME SAMPM |
| 4 | 4 | | | | ***** |
| 5 | 5 | | | | ***** |
| 6 | 6 | | | | HEX -> ASCII Conversion Program |
| 7 | 7 | | | | ***** |
| 8 | 8 | | | | main-routine |
| 9 | 9 | | | | ***** |
| 10 | 10 | | | | ***** |
| 11 | 11 | | | | |
| 12 | 12 | | | | PUBLIC MAIN,START |
| 13 | 13 | | | | EXTRN CONVAH |
| 14 | 14 | | | | |
| 15 | 15 | ---- | | | DATA DSEG AT 0FE20H |
| 16 | 16 | FE20 | | | HDTSA: DS 1 |
| 17 | 17 | FE21 | | | STASC: DS 2 |
| 18 | 18 | | | | |
| 19 | 19 | ---- | | | CODE CSEG AT 0H |
| 20 | 20 | 0000 R0200 | | | MAIN: DW START |
| 21 | 21 | | | | |
| 22 | 22 | ---- | | | |
| 23 | 23 | 0002 2B4100 | | | START: CSEG |
| 24 | 24 | 0005 0BFC80FE | | | MOV RFM,#00 |
| 25 | 25 | 0009 2B4000 | | | MOVW SP,#0FE80H |
| 26 | 26 | 000C 0944F708 | | | MOV MM,#00 |
| 27 | 27 | | | | MOV STBC,#08H |
| 28 | 28 | 0010 3A201A | | | MOV HDTSA,#1AH |
| 29 | 29 | 0013 6720FE | | | MOVW HL,#HDTSA |
| 30 | 30 | | | | |
| 31 | 31 | 0016 R282200 | | | CALL !CONVAH |
| 32 | 32 | | | | |
| 33 | 33 | 0019 6521FE | | | MOVW DE,#STASC |
| 34 | 34 | 001C D3 | | | MOV A,B |
| 35 | 35 | 001D 50 | | | MOV [DE+],A |
| 36 | 36 | 001E D2 | | | MOV A,C |
| 37 | 37 | 001F 50 | | | MOV [DE+],A |
| 38 | 38 | | | | |
| 39 | 39 | 0020 14FE | | | BR \$\$ |
| 40 | 40 | | | | |
| 41 | 41 | | | | END |

Segment informations:

| ADRS | LEN | NAME |
|------|-------|-------|
| FE20 | 0003H | DATA |
| 0000 | 0002H | CODE |
| 0002 | 0020H | ?CSEG |

Target chip:uPD78310

Assembly complete, 0 error(s) and 0 warning(s) found. (0)

Note: In this absolute assembly list, the comment statement section has been deleted after the output of the list.

APPENDIX B. LIST OF HINTS ON USE

| Item No. | Point to Bear in Mind | See page |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|
| 1 | About MS-DOS (PC-DOS) Based System: o When starting up each program of this assembler package, the parameter FILES must be set to 13 or more in the environment setting file "CONFIG.SYS" of MS-DOS (PC-DOS). o The required memory size is 384K bytes. | 2-2 |
| 2 | About Overlay Files: o With MS-DOS V3.10, the overlay files of the assembler, linker, or list converter must have been stored in the same directory as the command file when starting up the program. | 4-4, 5-41 8-6 |
| 3 | About Restriction on the Number of Module Files That Can Be Input to the Linker: With 78K/0,78K/I, 78K/VI: o Up to 128 object module files can be input to the linker. With 78K/III: o Up to 64 object module files can be input to the linker. | 1-23 |
| 4 | About Restrictions on the Number of Symbols: o With the assembler, the total number of local and PUBLIC symbols is 2,900. o With the linker, the number of local symbols is 2,900 per module and the number of PUBLIC symbols is 3,000. | 1-23 |
| 5 | About Restrictions on the Number of Segments That Can Be Input to the Assembler: o Up to 20 ?ASEG segments may be input to the assembler. o Up to 80 segments other than ?ASEG may be input to the assembler. | 1-24 |
| 6 | About Option Specification: o If two or more options which are not allowed to be specified with any other options are specified at the same time, whichever you specified last will take precedence over the preceding options. o The -C option cannot be omitted when starting up the assembler. If the -C option is to be omitted from the start-up command line, describe the device model with the \$PC (device model) control instruction in the header of the source module file. o If the -- (HELP message display) option is specified, all other options specified at the same time will become invalid. | 4-10 4-13 4-63 |

APPENDIX C. OPTION LISTS

C.1 List of Assembler Options

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|------------------------------------------------------------|--------------------|------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|---------------------------|----------|
| 1 | Option for processor type specification | -C device model | Specifies target device subject to assembly. | Independent | None | 4-13 |
| 2 | Options for object module file output specification | -O [filename] | Specifies output of object module file. | If -O and -NO are specified at same time, whichever you specified later takes precedence. | -O input filename .REL | 4-17 |
| | | -NO | Specifies suppression of object module file output. | | | |
| 3 | Options for forced object module file output specification | -J | Specifies forced output of object module file even in case of fatal error. | If -J and -NJ are specified at same time, whichever you specified later takes precedence. | -NJ | 4-19 |
| | | -NJ | Specifies non-output of object module file in case of fatal error. | | | |
| 4 | Options for debug information output specification | -G | Specifies output of debugging information to object module file. | If -G and -NG are specified at same time, whichever you specified later takes precedence. | -NG | 4-21 |
| | | -NG | Specifies suppression of debugging information output to object module file. | | | |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|-----------------------------------------------------------|------------------------------|---------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|--------------------------------------------------------------|----------|
| 5 | Options for symbol name length specification | -S | Specifies extension of symbol name length to max. 31 characters. | If -S and -NS are specified at same time, whichever you specified later takes precedence. | -NS | 4-23 |
| | | -NS | Specifies symbol name length as 8 characters max. | | | |
| 6 | Options for symbol name uppercase/lowercase specification | -CA | Specifies that symbol names in uppercase letters are not to be distinguished from those in lowercase letters. | If -CA and -NCA are specified at same time, whichever you later takes precedence. | -CA | 4-25 |
| | | -NCA | Specifies that symbol names in uppercase letters are to be distinguished from those in lowercase letters. | | | |
| 7 | Options for Include file read path specification | -I pathname [, pathname] ... | Specifies input of Include file(s) from path(s) specified by this option. | Independent | Path specified by environment variable INC78Kn (n=0,1,2,3,6) | 4-27 |
| 8 | Options for assembly list file output specification | -P[filename] | Specifies output of assembly list file. | If -P and -NP are specified at same time, whichever you specified later takes precedence. | -P input filename .PRN | 4-29 |
| | | -NP | Specifies suppression of preprocess list file output. | | | |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|----------------------------------------------------------|----------------------|-----------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|----------|
| 9 | Options for assembly list file information specification | -KA | Specifies output of assembly list to assembly list file. | If -KS and -KX are specified at same time, the assembler will ignore -KS option. | -KA | 4-31 |
| | | -NKA | Specifies suppression of assembly list output to assembly list file. | | | |
| | | -KS | Specifies output of symbol list to assembly list file. | If -KA and -NKA, -KS and -NKS, or -KX and -NKX are specified at same time, whichever you specified later takes precedence. -P option will be ignored if -NKA, -NKS, and -NKX are specified at same time. | -NKS | 4-34 |
| | | -NKS | Specifies suppression of symbol list output to assembly list file. | | | |
| | | -KX | Specifies output of cross-reference list to assembly list file. | | -NKX | 4-37 |
| | | -NKX | Specifies suppression of cross-reference list output to assembly list file. | | | |
| 10 | Options for assembly list file format specification | -LW [no. of columns] | Specifies no. of columns per line of assembly list file. | If -NP is specified at same time, -LW will be ignored. | -LW132 (-LW80: console output) | 4-39 |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|-------------------------------------------------------------|----------------------|------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|----------|
| 10 | Options for assembly list file format specification (contd) | -LL [no. of lines] | Specifies no. of lines to be printed per page of assembly list file. | If -NP is specified at same time, -LL will be ignored. | -LL66 (No page ejection: console output) | 4-44 |
| | | -LH character-string | Specifies character string to be printed as title in header of assembly list file. | If -NP is specified at same time, -LH will be ignored. | None | 4-47 |
| | | -LT no. of columns | Specifies no. of columns for tabulation. | If -NP is specified at same time, -LT will be ignored. | -LT8 | 4-51 |
| | | -LF | Specifies addition of formfeed code to the end of assembly list file. | If -LF and -NLF are specified at same time, whichever you specified later takes precedence. If -NP is specified at same time, -LF will be ignored. | -NLF | 4-55 |
| | | -NLF | Specifies non-addition of formfeed code to the end of assembly list file. | | | |
| 11 | Options for error list file output specification | -E[filename] | Specifies output of error list file to file. | If -E and -NE are specified at same time, whichever you specified later takes precedence. | -NE | 4-57 |
| | | -NE | Specifies suppression of error list file output. | | | |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|----------------------------------------------------------------|-----------------------|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|-------------------------------------------------------------------------------|-------------|
| 12 | Option for parameter file specification | -F [filename] | Specifies input of assembler options or input filename(s) from file specified by this option. | Independent | Allows input of options & filenames from only command line. | 4-59 |
| 13 | Option for temporary file creation path specification | -T pathname | Specifies creation of temporary file on path specified by this option. | Independent | Path spec- ified by environment variable TMP. | 4-61 |
| 14 | Option for HELP message display specification | -- | Specifies output of HELP message to console. | All other options specified at same time will be ignored. | No HELP message is displayed. | 4-63 |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|----------------------------------------------------------|--------------------|--------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|----------|
| 1 | Options for load module file output specification | -O [filename] | Specifies output of load module file. | If -O and -NO are specified at same time, whichever you specified later takes precedence. | -O input filename .LNK | 5-50 |
| | | -NO | Specifies non-output of load module file. | | | |
| 2 | Options for forced load module file output specification | -J | Specifies forced output of load module file even in case of fatal error. | If -J and -NJ are specified at same time, whichever you specified later takes precedence. | -NJ | 5-52 |
| | | -NJ | Specifies non-output of load module file in case of fatal error. | | | |
| 3 | Options for debugging information output specification | -G | Specifies addition of debugging information to load module file. | If -G and -NG are specified at same time, whichever you specified later takes precedence. If -NG is specified at same time with -KP or -KL, neither local symbol list nor PUBLIC symbol list will be output. | -NG | 5-54 |
| | | -NG | Specifies non-addition of debugging information to load module file. | | | |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|-----------------------------------------------------------|--------------------|----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|----------|
| 4 | Options for stack reserving symbol creation specification | -S [area-name] | Specifies creation of PUBLIC symbols for stack area reservation. | If -S and -NS are specified at same time, whichever you specified later takes precedence. | -NS | 5-56 |
| | | -NS | Specifies non-creation of PUBLIC symbols for stack area reservation. | | | |
| 5 | Option for directive file specification | -D filename | Specifies input of file specified by this option as directive file. | Independent | None | 5-59 |
| 6 | Options for link list file output specification | -P [filename] | Specifies output of link list file. | If -P and -NP are specified at same time, whichever you specified later takes precedence. | -P input filename .MAP | 5-61 |
| | | -NP | Specifies non-output of link list file. | | | |
| 7 | Options for link list file information specification | -KM | Specifies output of map list to link list file. | If -KM and -NKM are specified at same time, whichever you specified later takes precedence. If -NKM, -NKP, -NKL are all specified at same time, -P will be ignored. | -KM | 5-63 |
| | | -NKM | Specifies non-output of map list to link list file. | | | |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|--------------------------------------------------------------|--------------------|---------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|----------|
| 7 | Options for link list file information specification (contd) | -KD | Specifies output of directive file to link list file. | If -NKM is specified, -KD will be ignored. If -KD and -NKD, -KP and -NKP, or -KL and -NKL are specified at same time, whichever you specified later takes precedence. | -KD | 5-66 |
| | | -NKD | Specifies non-output of directive file to link list file. | | -NKP | 5-69 |
| | | -KP | Specifies output of PUBLIC symbol list to link list file. | | | |
| | | -NKP | Specifies non-output of PUBLIC symbol list to link list file. | If -NG is specified at same time with -KP or KL, neither PUBLIC symbol list nor local symbol list will be output. | -NKL | 5-69 |
| | | -KL | Specifies output of local symbol list to link list file. | | | |
| | | -NKL | Specifies non-output of local symbol list to link list file. | | | |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|--------------------------------------------------|--------------------|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|----------|
| 8 | Options for link list file format specification | -LL [no. of lines] | Specifies no. of print lines per page of list. | If -NP is specified, -LL will be ignored. (output) | -LL66 (No page ejection: console) | 5-75 |
| | | -LF | Specifies addition of a formfeed (FF) code to the end of list file. | If -LF and -NFL are specified at same time, whichever you specified later takes precedence. If -NP is specified, -LF will be ignored. | -NFL | 5-78 |
| | | -NFL | Specifies non-addition of formfeed code to the end of list file. | | | |
| 9 | Options for error list file output specification | -E [filename] | Specifies output of error list file. | If -E and -NE are specified at same time, whichever you specified later takes precedence. | -NE | 5-80 |
| | | -NE | Specifies non-output of error list file. | | | |
| 10 | Option for library file specification | -B filename | Specifies input of file specified by this option as library file. | Independent | None | 5-82 |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|-------------------------------------------------------|-------------------------------|-----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|--------------------------------------------------------------|----------|
| 11 | Option for library file read path specification | -I pathname [,pathname]... | Tells the linker to read library file(s) from path(s) specified by this option. | If library file without pathname is specified with -B option, -I option becomes invalid. | Path specified by environment variable LIB78Kn (n=0,1,2,3,6) | 5-84 |
| 12 | Option for parameter file specification | -F [filename] | Specifies input of assembler options or input filename(s) from file specified by this option. | Independent | Allows input of options & filenames from only command line. | 5-86 |
| 13 | Option for temporary file creation path specification | -T pathname | Specifies creation of temporary file on path specified by this option. | Independent | Path specified by environment variable TMP. | 5-88 |
| 14 | Option for HELP message display specification | -- | Specifies output of HELP message to console. | All other options specified at same time will be ignored. | No HELP message is displayed. | 5-90 |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|----------------------------------------------------------------|--------------------|-----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|----------|
| 1 | Options for HEX-format object module file output specification | -O [filename] | Specifies output of HEX-format object module file. | If -O and -NO are specified at same time, whichever you specified later takes precedence. | -O input filename .HEX (.H1 to .H15 for extension space.) | 6-12 |
| | | -NO | Specifies non-output of HEX-format object module file. | | | |
| 2 | Options for symbol table file output specification | -S [filename] | Specifies output of symbol table file. | If -S and -NS are specified at same time, whichever you specified later takes precedence. | -S input filename .SYM (.S1 to .S15 for extension space) | 6-15 |
| | | -NS | Specifies non-output of symbol table file. | | | |
| 3 | Options for object code output sequence specification | -R | Specifies output of HEX-format object codes by sorting in address sequence. | If -R and -NR are specified at same time, whichever you specified later takes precedence. If -NO is specified, -R will be ignored. | -NR | 6-18 |
| | | -NR | Specifies output of HEX-format object codes in the order as stored in input load module file. | | | |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|--------------------------------------------------|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|-------------------------------------------------------------------|----------|
| 4 | Option for object code fill specification | -U fill-value [, [start], size] | Specifies output of fill value specified by this option as object code to address area to which no HEX-format object code is output. | If -NO is specified, -U will be ignored. | None | 6-20 |
| 5 | Options for error list file output specification | -E [filename] | Specifies output of error list file. | If -E and -NE are specified at same time, whichever you specified later takes precedence. | -NE | 6-23 |
| | | -NE | Specifies non-output of error list file. | | | |
| 6 | Option for parameter file specification | -F [filename] | Specifies input of object converter options or input filename(s) from file specified by this option. | Independent | Allows input of options & input filenames from only command line. | 6-25 |
| 7 | Option for HELP message display specification | -- | Specifies output of HELP message to console. | All other options specified at same time will be ignored. | No HELP message is displayed. | 6-27 |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|-------------------------------------------------------|----------------------|------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|---------------------------------------------|----------|
| 1 | Options for list file format specification | -LW [no. of columns] | Specifies the no. of print columns per line of list file. | If LIST sub-command is not specified, all these options will be ignored. | -LW132 (-LW80: console output) | 7-13 |
| | | -LL [no. of lines] | Specifies the no. of print lines per page of list file. | | -LL66 (no page ejection: console output) | 7-15 |
| | | -LF | Specifies addition of formfeed (FF) code to the end of list file. | If -LF and -NLF are specified at same time, whichever you specified later takes precedence | -NLF | 7-17 |
| | | -NLF | Specifies non-addition of formfeed (FF) code to the end of list file. | | | |
| 2 | Option for temporary file creation path specification | -T pathname | Specifies creation of temporary file on path specified by this option. | Independent | Path specified by environment variable TMP | 7-19 |
| 3 | Option for HELP message output specification | -- | Specifies output of HELP message to console. | All other options specified at same time will be ignored. | No HELP message is output. | 7-21 |

| Item | Classification | Description Format | Function | Relation with Other Options | Default Assumption | See Page |
|------|---------------------------------------------------|--------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|-------------------------------------------------------------|----------|
| 1 | Option for object module file input specification | -R filename | Specifies input of object module file. | Independent | -R assembly list file-name.REL | 8-12 |
| 2 | Option for load module file input specification | -L filename | Specifies input of load module file. | Independent | -L assembly list file-name.LNK | 8-14 |
| 3 | Option for absolute assembly list specification | -O filename | Specifies output of absolute assembly list file. | Independent | -O assembly list file-name.P | 8-16 |
| 4 | Options for error list file output specification | -E [filename] | Specifies output of error list file. | If -E and -NE are specified at same time, whichever you specified later takes precedence. | -NE | 8-18 |
| | | -NE | Specifies non-output of error list file. | | | |
| 5 | Option for parameter file specification | -F filename | Specifies input of input filename and options from file specified by this option. | Independent | Allows input of options & filenames only from command line. | 8-20 |
| 6 | Option for HELP message output specification | -- | Specifies output of HELP message to console. | All other options specified at same time will be ignored. | No HELP message is output. | 8-22 |

| Item | Subcommand Name | Description Format | Function | Abbrev. Format | See Page |
|------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|----------------|----------|
| 1 | CREATE | CREATE Δ library-filename [Δ transaction] | Creates a new library file. | C | 7-25 |
| 2 | ADD | ADD Δ library-filename Δ transaction | Adds a module to a library file. | A | 7-27 |
| 3 | DELETE | DELETE Δ library-filename \blacktriangle (\blacktriangle module-name[\blacktriangle , ...] \blacktriangle) | Deletes a module from a library file. | D | 7-29 |
| 4 | REPLACE | REPLACE Δ library-filename Δ transaction | Replaces a module in a library with another module. | R | 7-31 |
| 5 | PICK | PICK Δ library-filename \blacktriangle (\blacktriangle module-name[\blacktriangle , ...] \blacktriangle) | Selects and copies a module from a library file. | P | 7-33 |
| 6 | LIST | LIST [Δ option] library-filename [\blacktriangle (\blacktriangle module-name[\blacktriangle , ...] \blacktriangle)] option ::= -P/-NP -O \blacktriangle filename | Outputs information on the modules stored in a library file. | L | 7-35 |
| 7 | HELP | HELP | Outputs HELP message to the console. | H | 7-37 |
| 8 | EXIT | EXIT | Terminates the librarian. | E | 7-38 |

NEC