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.

User's Manual

RENESAS

Phase-out/Discontinued

RA78K3

Assembler Package

Operation

RA78K3
Ver. 5.00 or Later

Document No. U10967EJ2V0UM00 (2nd edition)
Date Published January 1998 N CP(K)

[MEMO]

M7A 96.10

Phase-out/Discontinued

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Cumbica-Guarulhos-SP, Brasil
Tel: 011-6465-6810
Fax: 011-6465-6829

J97. 8

[MEMO]

INTRODUCTION

This manual is designed to facilitate correct understanding of the functions of each program in the RA78K3 Series Assembler Package (hereinafter referred to as "the RA78K3") and of the correct methods of using the package, for operators using the RA78K3 to develop software.

This manual does not cover the expressions of directives and source programs or language used in the RA78K3. Therefore, before reading this manual, read the **RA78K3 Assembler Package User's Manual - Language** (hereinafter referred to as "**Language**").

The contents of this manual are intended for use with Ver. 5.00 or later of the RA78K3.

[Target Users]

This manual is intended for users who understand the functions and instructions of the microcontrollers to be developed.

[Target Devices]

The software of the following microcontrollers can be developed with this assembler.

Subseries Name	Target Device
μPD78312	μPD78310[Note 1], 78312[Note 1], 78P312[Note 1]
μPD78312A	μPD78310A, 78312A, 78P312A
μPD78322	μPD78320, 78322, 78P322, 78323, 78324, 78P324
μPD78328	μPD78327, 78328, 78P328
μPD78334	μPD78330, 78334, 78P334
μPD78352A	μPD78350, 78350A, 78352A, 78P352
μPD78356	μPD78355, 78356, 78P356
μPD78366	μPD78365[Note 2], 78366[Note 2], 78P368[Note 2]
μPD78366A	μPD78361A, 78362A, 78P364A, 78363A, 78365A, 78366A, 78368A, 78P368A
μPD78372	μPD78372, 78P372

Notes 1. Discontinued product

2. Maintenance product

[Organization of Manual]

This manual consists of the following eleven chapters and appendixes:

Chapter 1 General

Outlines the role of the RA78K3 in microcontroller software development and the features of the RA78K3.

Chapter 2 Product Overview and Method of Installation

Explains the program file names and operating environment provided by the RA78K3.

Chapter 3 Executing the RA78K3

Explains the procedure for developing software, using a sample program.

The purpose of this chapter is to provide an opportunity for actual use of each program. Those who wish to experience operating the RA78K3 should read this chapter.

Chapter 4	Assembler
Chapter 5	Linker
Chapter 6	Object Converter
Chapter 7	Librarian
Chapter 8	List Converter
	Explain in detail the functions and methods of operation of each program.
	These chapters are important for the actual operation of each program of the RA78K3.
Chapter 9	Program Output List
	Explains the formats of the lists output by each program.
Chapter 10	Getting the most from the RA78K3
	Introduces some measures for optimum utilization of the RA78K3.
Chapter 11	Error Messages
	Explains the error messages output by each program.
Appendixes	Introduce a list of program options, a list of sample programs, and a list of notices on using the RA78K3.

The instruction sets are not detailed in this manual. For these instructions, refer to the user's manual of the microcontroller for which software is being developed.

[How to Read this Masnual]

Those using an assembler for the first time are encouraged to read from Chapter 1, General of this manual. Those who have a general understanding of assembler programs may skip this chapter.

Before using the RA78K3, read Chapter 3, Executing the RA78K3.

After you have become familiar with the operation of each program, you can proceed to utilize the lists in the appendixes.

[Note]

In this manual, it is assumed that the PC-9800 series or IBM PC/AT™ or a compatible machine is used as the host machine. When the HP9000 series 700™, SPARCstation™ family, or RISC NEWS™ is used, there are the following differences depending on the host machine:

- The format of a file name differs.
  - Extension .exe of the execution format is not suffixed with the EWS version of the HP9000 series 700.
  - The extension .bat of a batch file is .sh with the EWS version of the HP series 700.
  - The file name in uppercase letters is in lowercase letters with the EWS version of the HP9000 series 700.
- The execution examples and the method of setting the environments described in the manual differ.

[Legend]

The following symbols and abbreviations are used throughout this manual:

Symbol		Meaning
...	:	Indicates that the same expression is repeated.
[ ]	:	Item(s) in brackets can be omitted.
' '	:	Characters enclosed in ' ' (quotation marks) will be listed as they appear.
< >	:	Characters enclosed in < > (parentheses) will be listed as they appear (mainly titles).
" "	:	Characters enclosed in " " (double quotation marks) are titles of chapters, paragraphs, sections, diagrams or tables to which the reader is asked to refer.
___	:	Indicates an important point, or characters that are to be input in a usage example.
□	:	Indicates one blank space.
Δ	:	Indicates one or more blank or TAB.
∇	:	Indicates zero or more blanks or TABs (i.e. blanks may be omitted).
/	:	Indicates a break between characters.
~	:	Indicates continuity.
[↵]	:	Indicates pressing of the Return key.
Note	:	Indicates a note in the text of this manual.
Caution	:	Indicates information that should be read and noted carefully.
Remark	:	Indicates supplementary information in the text of this manual.

[Related Documents]

The related documents indicated in this publication may include preliminary versions.
However, preliminary versions are not marked as such.

	Document Name		Document No.	
			English	Japanese
Development Tools	RA78K3 Language		To be prepared	U10810J
	RA78K3 Operation		This manual	U10967J
	RA78K3 Structured Assembler Preprocesser		To be prepared	U11136J
	ECC Generator RA78K/III		EEU-1362	EEU-752
	CC78K Series C Compiler Language		EEU-1284	EEU-655
	CC78K Series C Compiler Operation		EEU-1280	EEU-656
	CC78K Series Library Source File		–	U12322J
	ID78K3 Reference		U10441E	U10441J
	IE-78310A-R	Hardware	EEU-1247	EEU-645
		Software	EEU-1248	EEU-637
	IE-78327-R	Hardware	EEU-1358	EEU-718
		Software	EEU-1341	EEU-720
	IE-78330-R	Hardware	EEU-1326	EEU-713
		Software	EEU-1298	EEU-714
	IE-78350-R	Hardware	EEU-1366	EEU-754
		Software	EEU-1376	EEU-753
	IE-78350-R-EM1		EEU-1377	EEU-773
	IE-78355-R-EM1		EEU-1423	EEU-866
	IE-78365-R-EM1		EEU-1454	EEU-924
	IE-78370-R-EM1		EEU-1474	EEU-946
	EP-78320GF-R		EEU-1490	EEU-971
	EP-78320L-R		EEU-1497	EEU-970
	EP-78320GJ-R		EEU-1498	EEU-972
	EP-78327CW-R		EEU-1496	EEU-969
	EP-78327GF-R		EEU-1499	EEU-973
	EP-78330GJ-R		EEU-1478	EEU-958
	EP-78330LQ-R		EEU-1479	EEU-959
	EP-78355GC-R		EEU-1508	EEU-963
	EP-78355GD-R		EEU-1509	EEU-964
	EP-78365GF-R		EEU-1488	EEM-955
Devices	μPD78312A		IEU-1265	IEM-5086
	μPD78322		IEU-1248	IEU-619
	μPD78328		IEU-1268	IEU-693
	μPD78334		IEU-1315	IEU-729
	μPD78352A Hardware		IEU-1327	IEU-781
	μPD78356 Hardware		U10669E	U10669J
	μPD78356 Instruction		U12117E	U12117J
	μPD78362A Hardware		U10745E	U10745J
	μPD78366A Hardware		U10205E	U10205J
	μPD78372 Hardware		U10642E	U10642J

CONTENTS

LIST OF FIGURES

LIST OF TABLES (1/2)

LIST OF TABLES (2/2)

CHAPTER 1 GENERAL

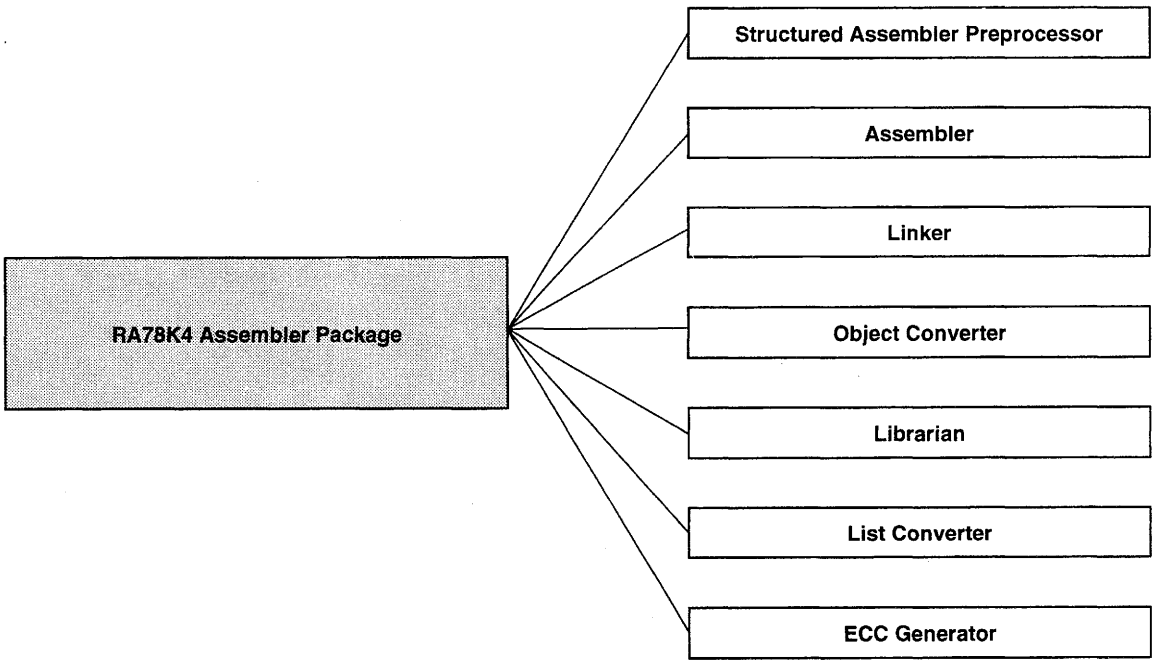
This chapter describes the role of the RA78K3 in microcontroller software development and the features of the RA78K3.

1.1 Assembler Overview

The RA78K3 Assembler Package is a generic term for a series of programs designed to translate source programs coded in the assembly language for 78K/III series microcontrollers into machine language coding.

The RA78K3 contains seven programs: Structured Assembler Preprocessor, Assembler, Linker, Object Converter, Librarian, List Converter and ECC Generator.

Figure 1-1. RA78K3 Assembler Package



1.1.1 What is an assembler?

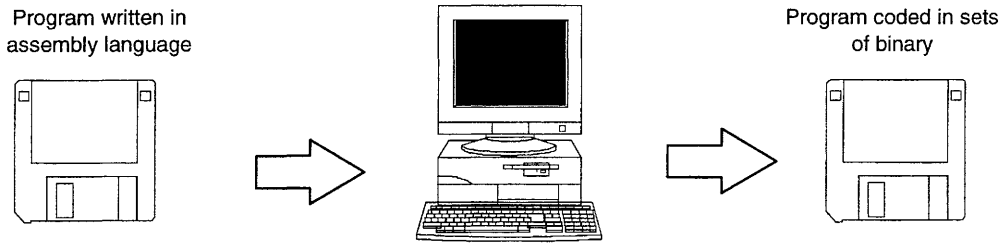
(1) Assembly language and machine language

An assembly language is the most basic programming language for microcontrollers.

For a microcontroller to do its job, programs and data are required. These programs and data must be written by people (i.e., programmers) and stored in the memory section of the microcontroller. Programs and data that can be handled by a microcontroller are written in machine language, which consist of binary numbers. However, programming in machine language, or in binary numbers, is difficult for humans and they may make mistakes. Fortunately, methods exist whereby English abbreviations or mnemonics are used to represent the meanings of the original machine language codes in a way that is easy for people to comprehend. A programming language system that uses this symbolic coding is called an assembly language.

A microcontroller requires a program that translates the program developed in an assembly language into collections of binary numbers the microcontroller can understand. This program is called an assembler.

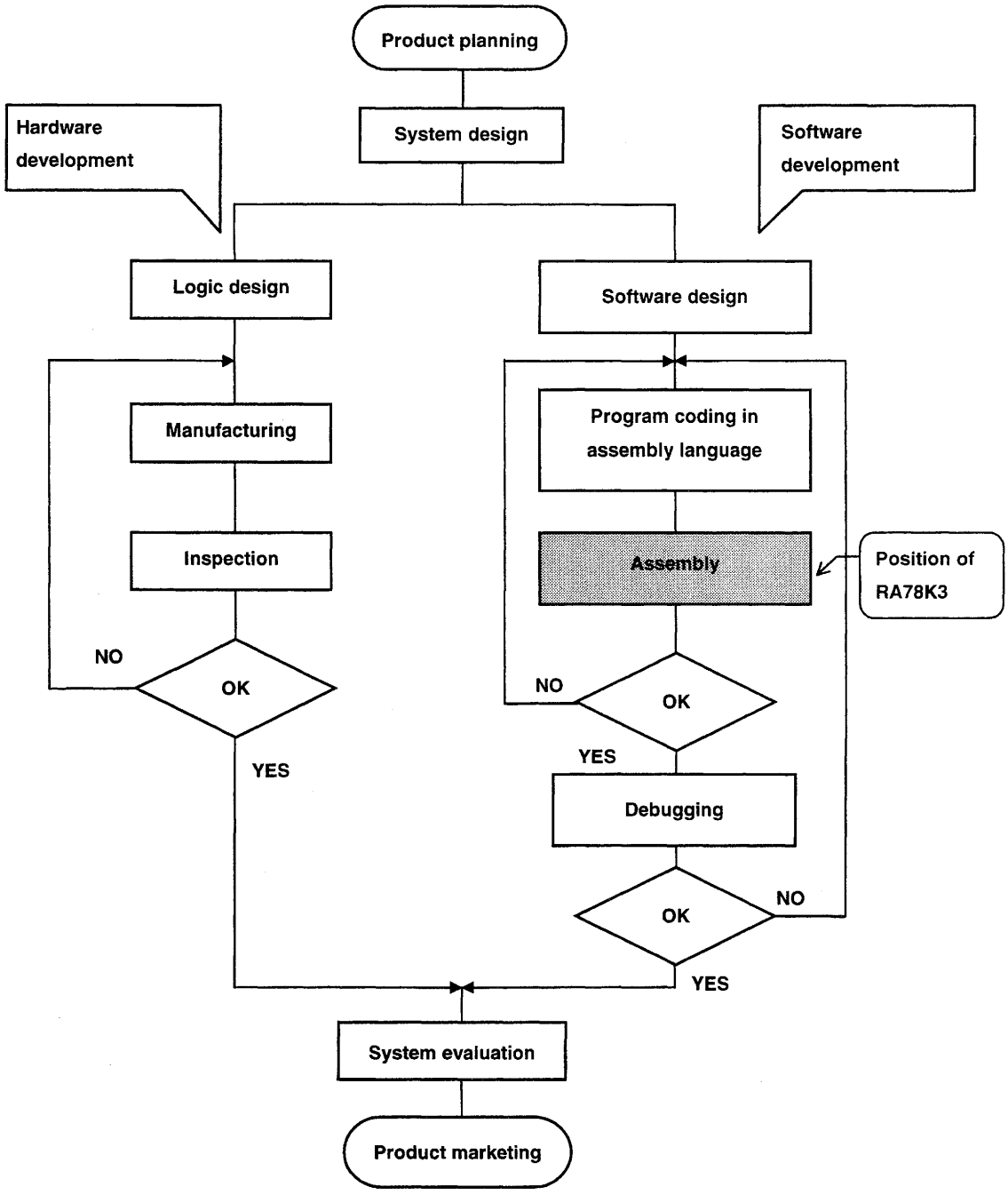
Figure 1-2. Flow of Assembler



(2) Development of microcontroller-related products and the role of RA78K3

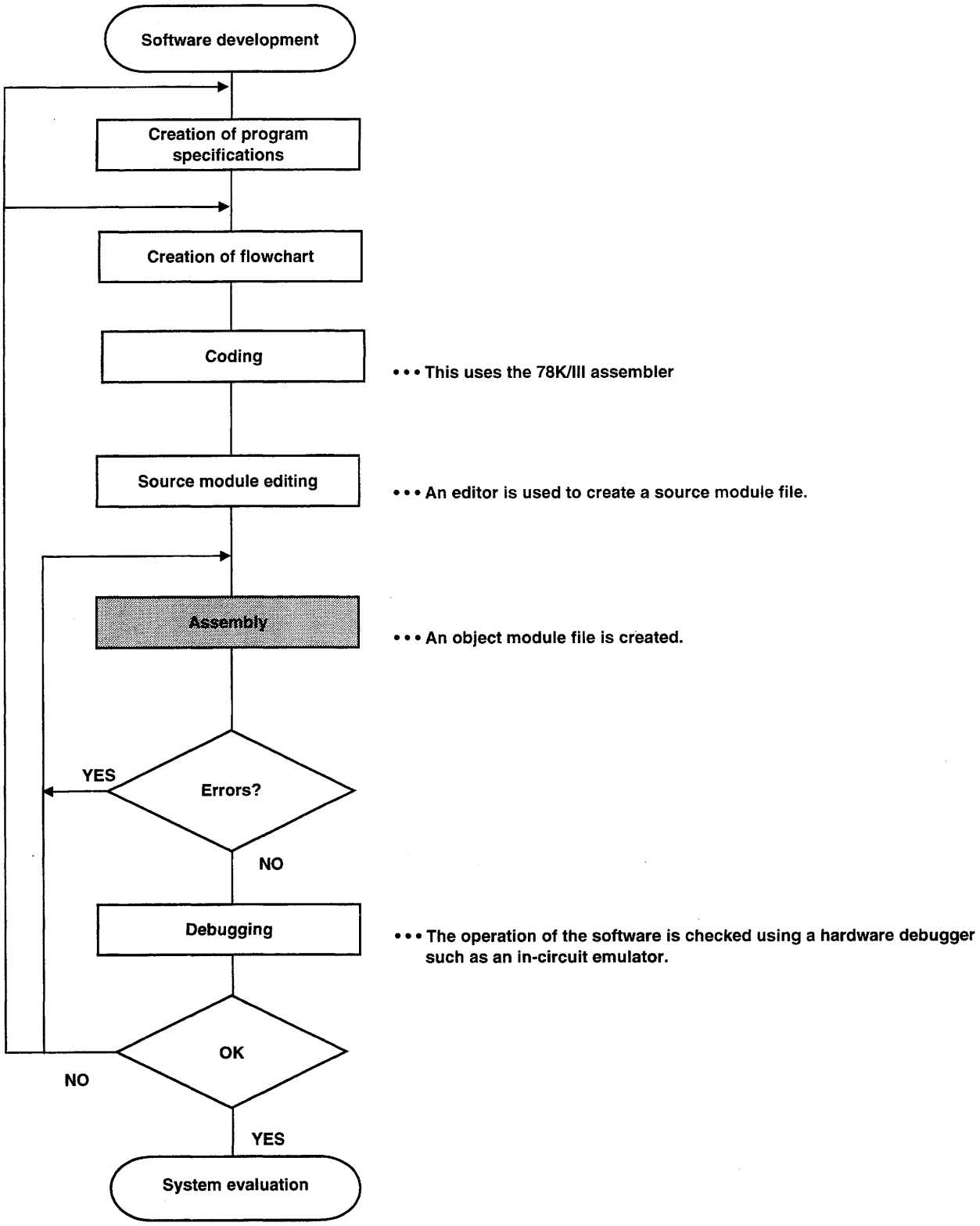
Figure 1-3, "Development Process of Microcontroller-Applied Products," illustrates the position of assembly-language programming in the (software) product development process.

Figure 1-3. Development Process of Microcontroller-Applied Products



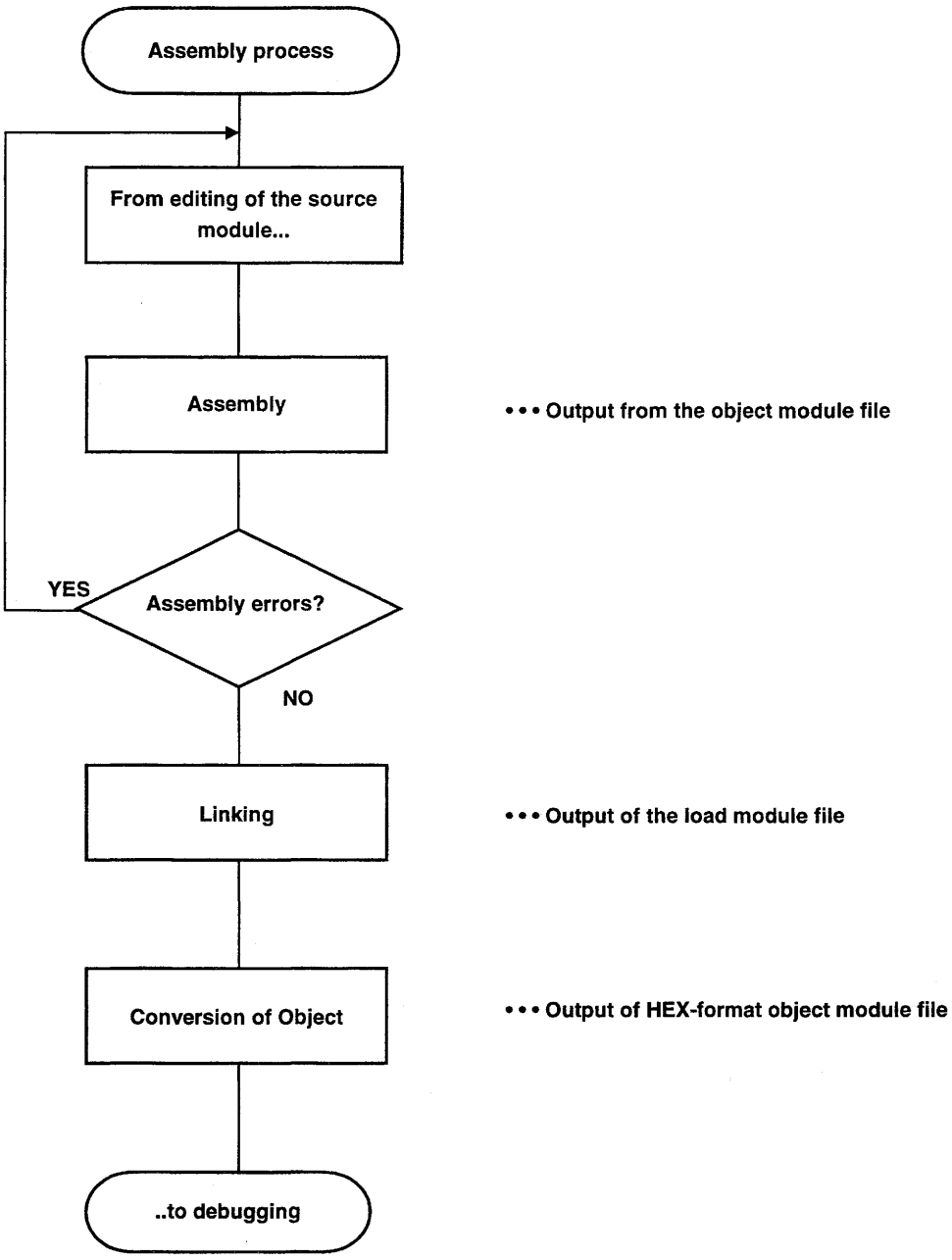
A more detailed explanation of the software development process appears in Figure 1-4, "The Software Development Process."

Figure 1-4. The Software Development Process



The RA78K3 is then applied to the assembly process.

Figure 1-5. The RA78K3 Assembly Process



1.1.2 What is a relocatable assembler?

The machine language translated from a source language by the assembler is stored in the memory of the microcontroller before use. To do this, the location in memory where each machine language instruction is to be stored must already be determined. Therefore, information is added to the machine language assembled by the assembler, stating where in memory each machine language instruction is to be located.

Depending on the method of locating addresses to machine language instructions, assemblers can be broadly divided into absolute assemblers and relocatable assemblers.

- **Absolute assembler**

  An absolute assembler locates machine language instructions assembled from the assembly language to absolute addresses.

- **Relocatable assembler**

  In a relocatable assembler, the addresses determined for the machine language instructions assembled from the assembly language are tentative. Absolute addresses are determined subsequently by a program called the linker.

In the past, when a program was created with an absolute assembler, programmers had to, as a rule, complete programming at the same time. However, if all the components of a large program are created at the same time, the program becomes complicated, making analysis and maintenance of the program troublesome. To avoid this, such large programs are developed by dividing them into several subprograms, called modules, for each functional unit. This programming technique is called modular programming.

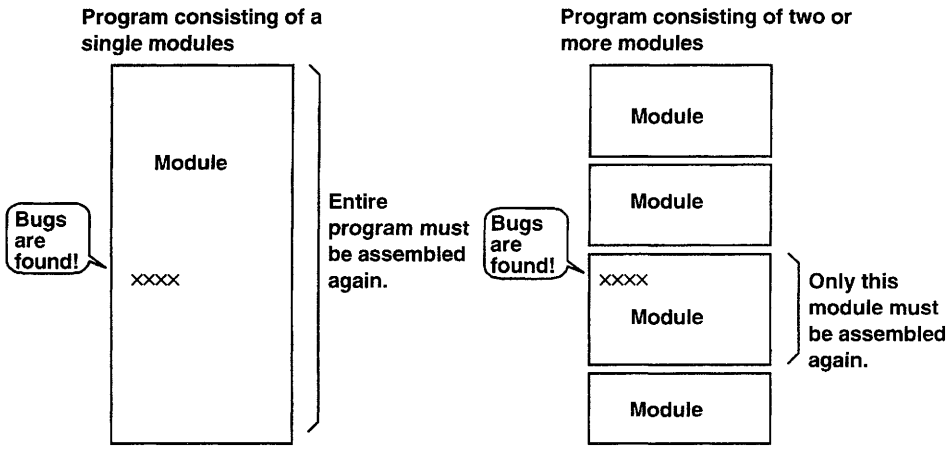
A relocatable assembler is an assembler suitable for modular programming. The following advantages can be derived from modular programming with a relocatable assembler:

(1) Increase in development efficiency

It is difficult to write a large program all at the same time. In such cases, dividing the program into modules for each function enables two or more programmers to develop subprograms in parallel to increase development efficiency.

Furthermore, if any bugs are found in the program, it is not necessary to assemble the entire program just to correct one part of the program, and only a module which must be corrected can be reassembled. This shortens debugging time.

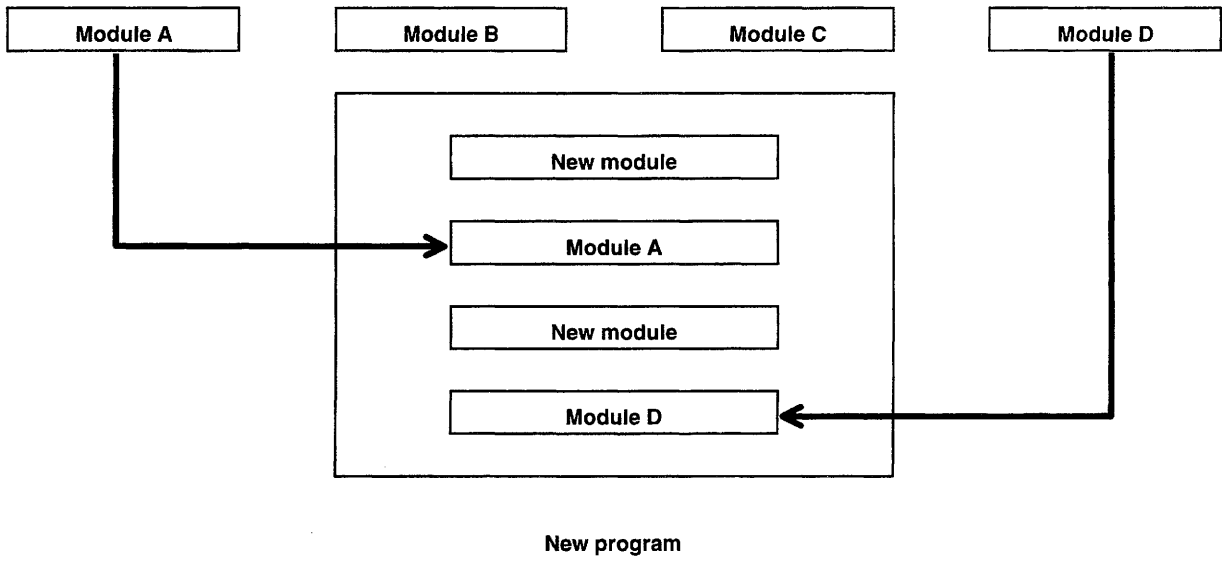
Figure 1-6. Reassembly for Debugging



(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.

Figure 1-7. Program Development Using Existing Module

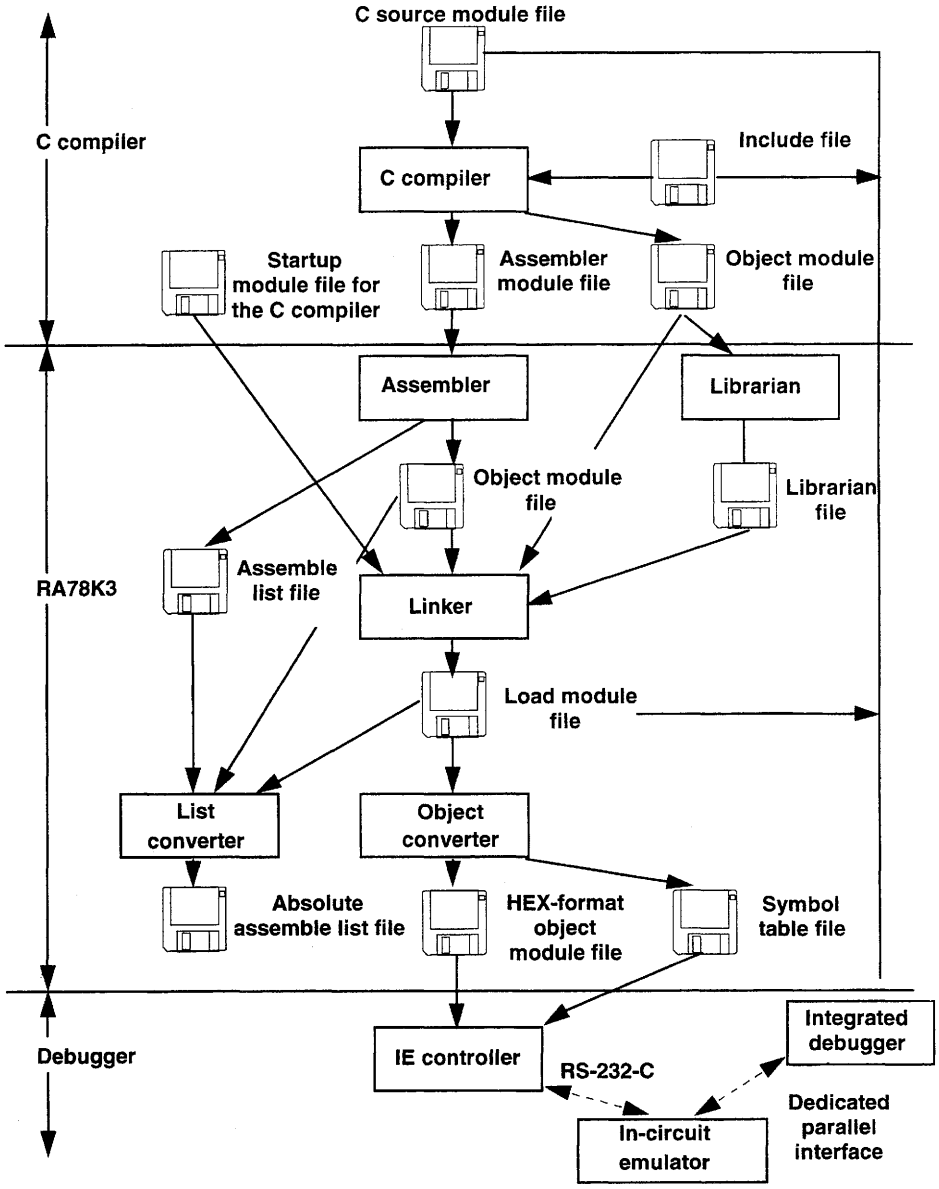


1.2 Overview of Features of the RA78K3

The procedure for developing general programs appears in Figure 1-8, "Procedure for Software Development Using the RA78K3." Program development essentially flows from the assembler to the linker to the object converter.

The assembler, linker, object converter and other programs are generically referred to as the "RA78K3." the assembler program is referred to as the "assembler."

Figure 1-8. Procedure for Software Development Using the RA78K3



1.2.1 Creating a source module file using an editor

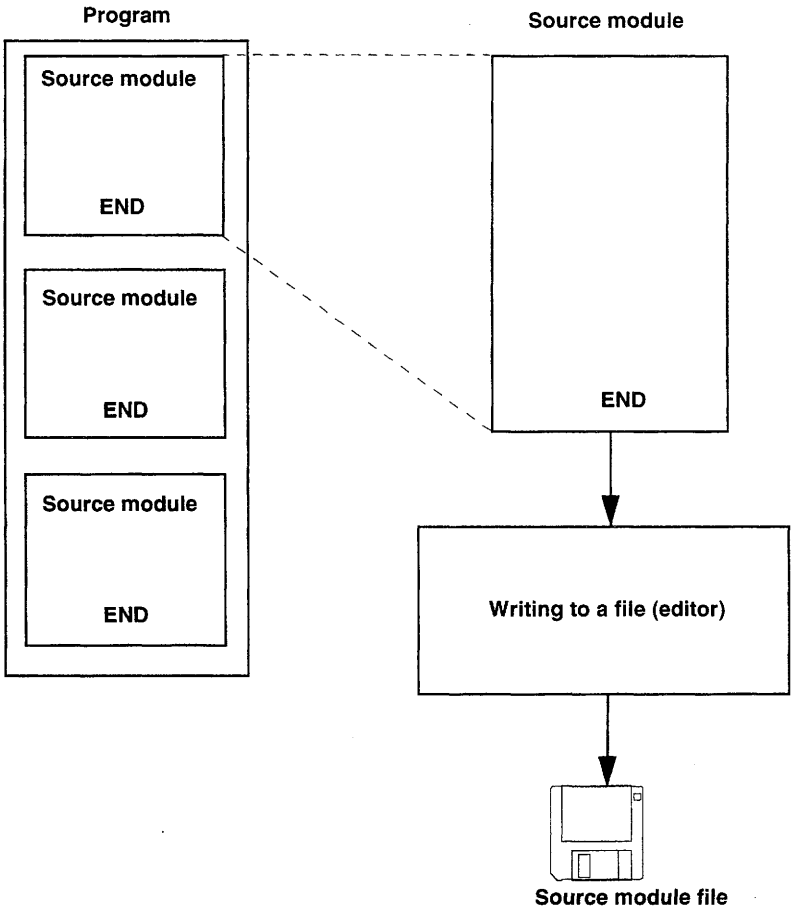
A single program can be divided into two or more modules according to function.

A single module can be used as a coding unit or an assembler input unit.

A module which is used as an input unit for the assembler is called a source module. After the coding of each source module is finished, the source module is written to a file using an editor. The file created in this way is called a source module file.

A source module file is used as an assembler input file.

Figure 1-9. Creating a Source Module file

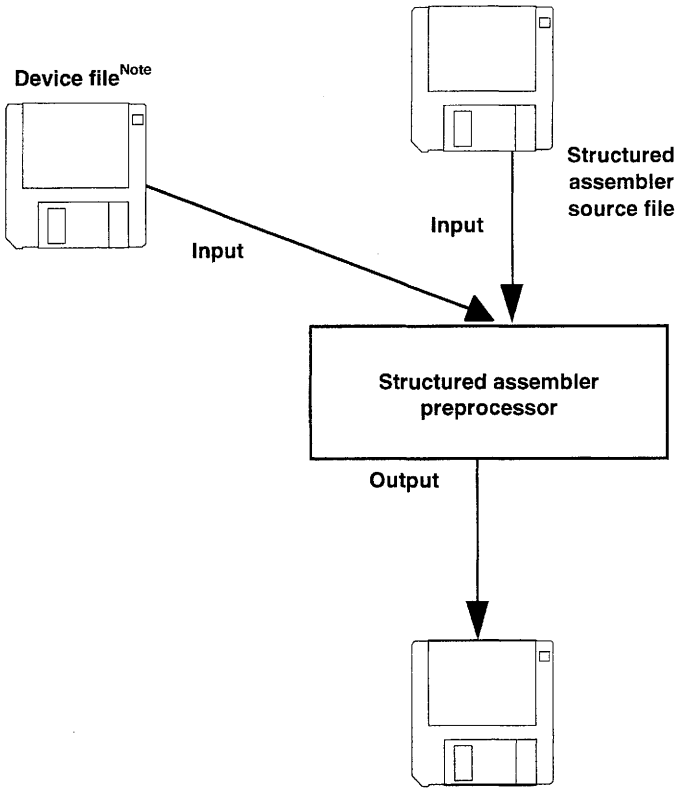


1.2.2 Structured assembler preprocessor

The structured assembler preprocessor is a program whose purpose is to create structured programming using assembly language instructions. The structured assembler preprocessor inputs source programs written in structured assembly language to input the source program for the assembler.

For more information on the structured assembler preprocessor and structured assembly language, refer to the separate "RA78K3 Series Structured Assembler Preprocessor User's Manual."

Figure 1-10. Function of the Structured Assembler Preprocessor



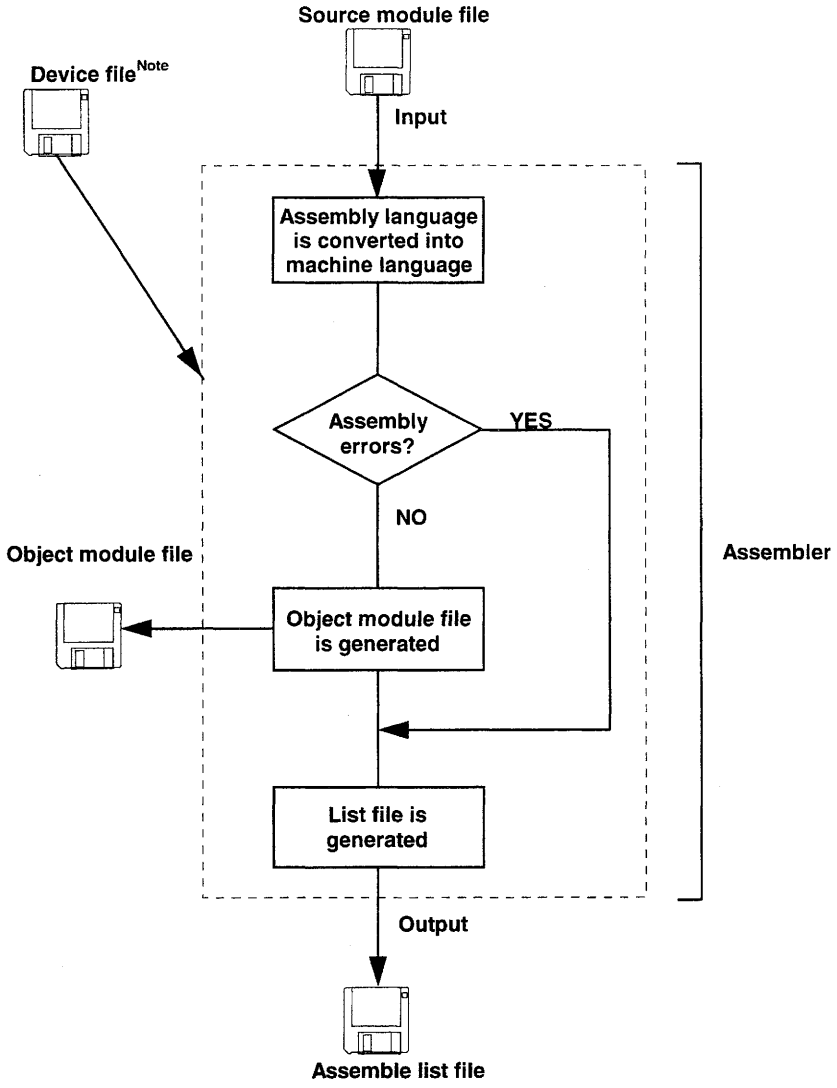
Note The device files of the following subseries are optional.

- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

1.2.3 Assembler

The assembler is a program which inputs the source module file and converts the assembly language into a collection of binary instructions (machine language). If the assembler discovers errors in the descriptions in the source module, it outputs an assembly error. If no assembly errors are found, the assembler outputs an object module file which specifies location data such as where in memory the machine language data and each machine language should be stored. The assembly data is output as an assemble list file.

Figure 1-11. Function of the Assembler



Note The device files of the following subseries are optional.

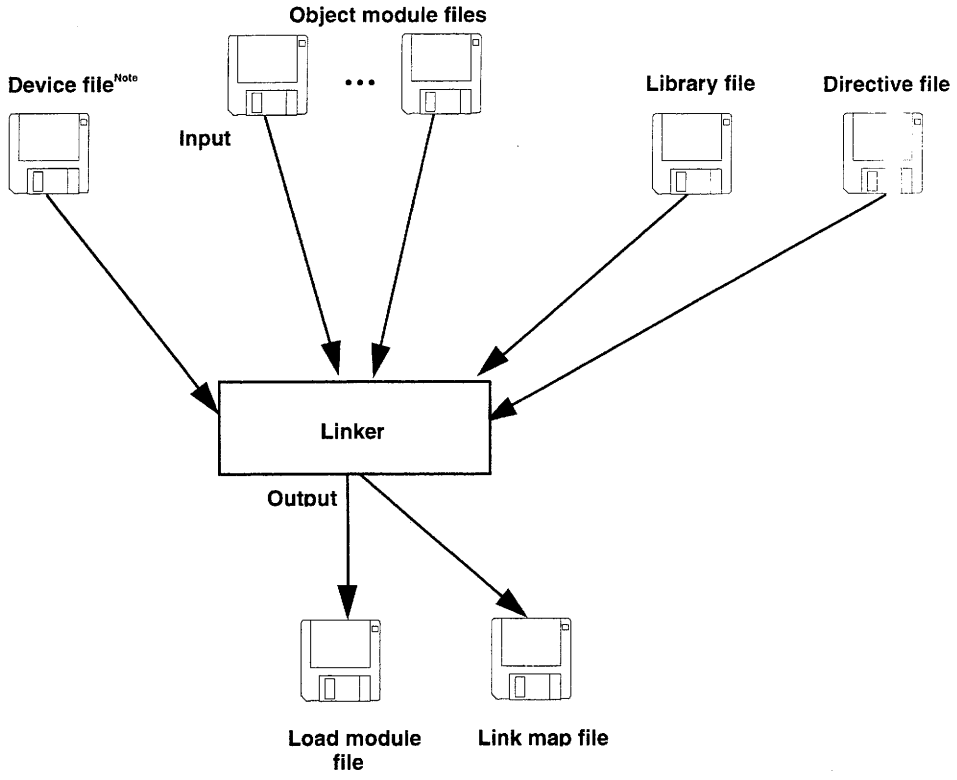
- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

1.2.4 Linker

The linker inputs the multiple object module files output by the compiler and the assembler and links them to output a single load module file (linking must be performed even if only one object module file is input).

The linker determines the location addresses for the relocatable segments in the input modules. This determines the values for the relocatable symbols and external-reference symbols so that the correct values can be embedded in the load module file.

Figure 1-12. Functions of the Linker



Note The device files of the following subseries are optional.

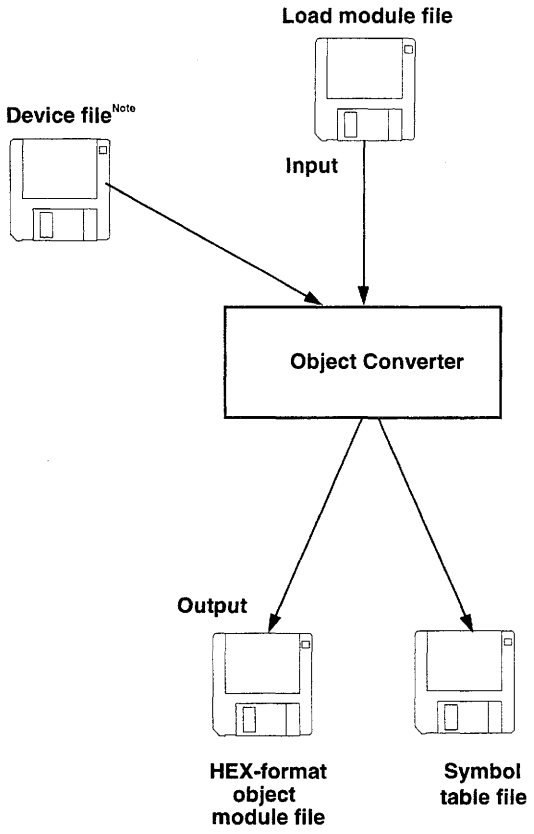
- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

1.2.5 Object converter

The object converter inputs the load module file output by the linker and converts the file format. The resulting file is output as a HEX-format object module file.

The object converter also outputs symbol data necessary for symbolic debugging as a symbol table file.

Figure 1-13. Function of the Object Converter



Note The device files of the following subseries are optional.

- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

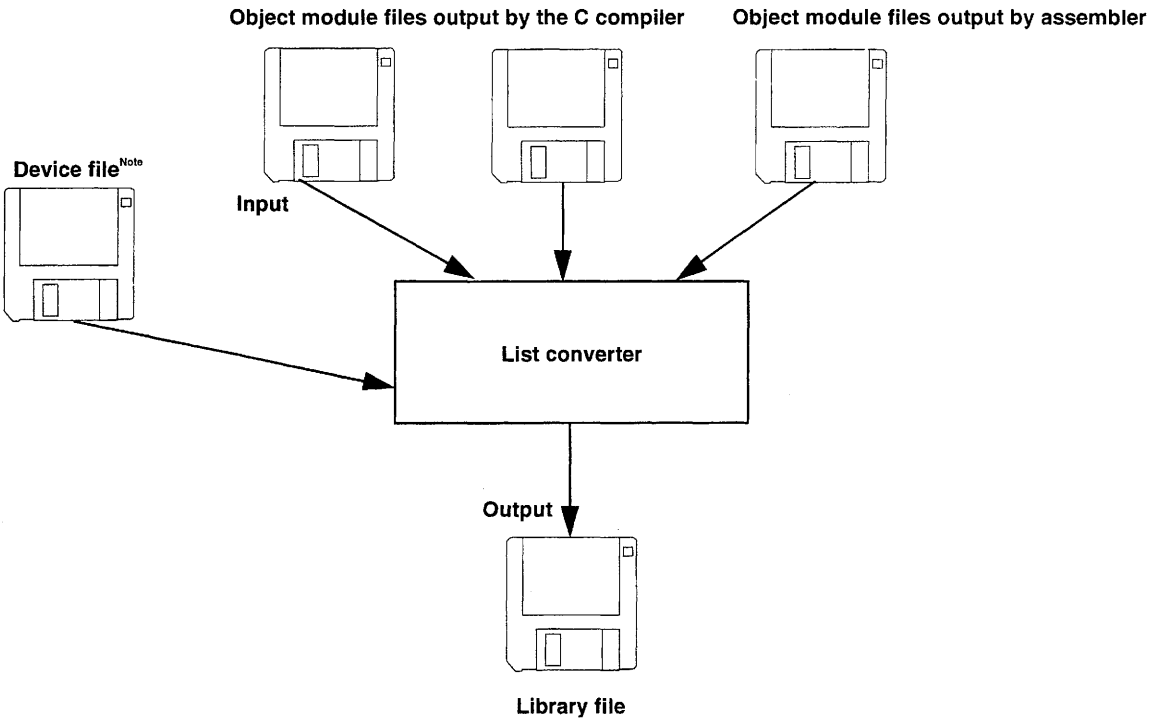
1.2.6 Librarian

For convenience and ease of use, a general-purpose module with a clear interface may be stored in a library. By creating a library, multiple object modules can be stored in a single file, making them easy to handle.

The linker incorporates a function which retrieves from the library file only the modules necessary. When multiple modules are registered in a single library file, the module files can be linked without the need to specify each individual module file name.

The librarian is the program used to create and update the library file.

Figure 1-14. Function of the Librarian



Note The device files of the following subseries are optional.

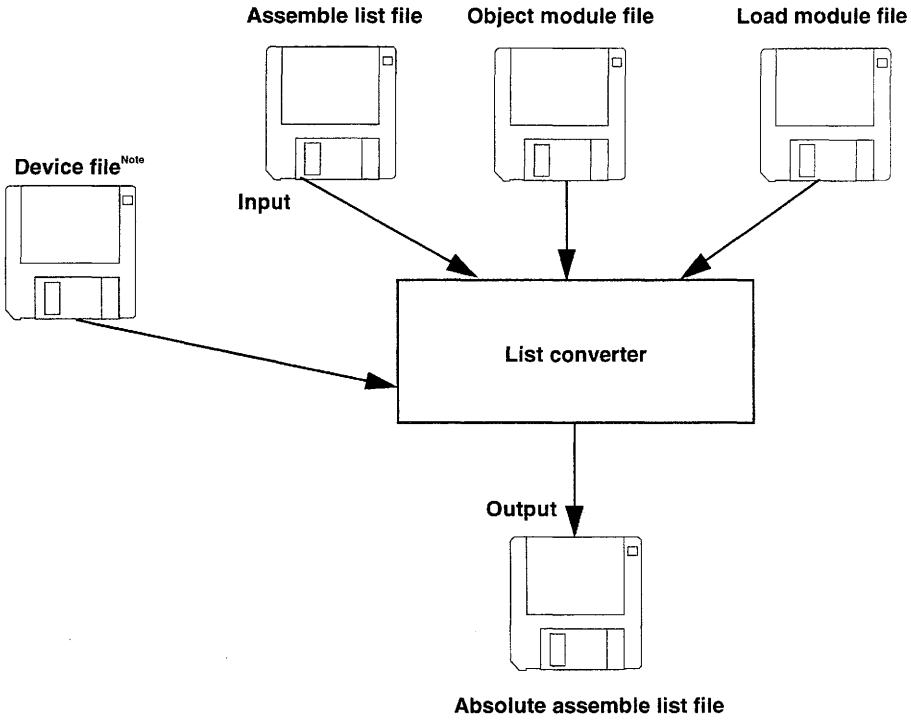
- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

1.2.7 List converter

The list converter inputs the object module files and assemble list file output by the assembler and the load module file output by the linker, and outputs an absolute assemble list file.

Relocatable assemble list files have the disadvantage that addresses and relocatable values in the list may be different from their actual values. An absolute assemble list file determines these values, making debugging and program maintenance easier.

Figure 1-15. Function of the List Converter



Note The device files of the following subseries are optional.

- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

1.2.8 ECC generator

The ECC generator is a tool for error correction code generation which is supplied with the RA78K3 only.

This tool generates and adds data to be written to the ECC ROM area of the μPD78P324, 78P334, 78P356, and 78P372.

For details, refer to the ECC Generator User's Manual (EEU-1362) separately available.

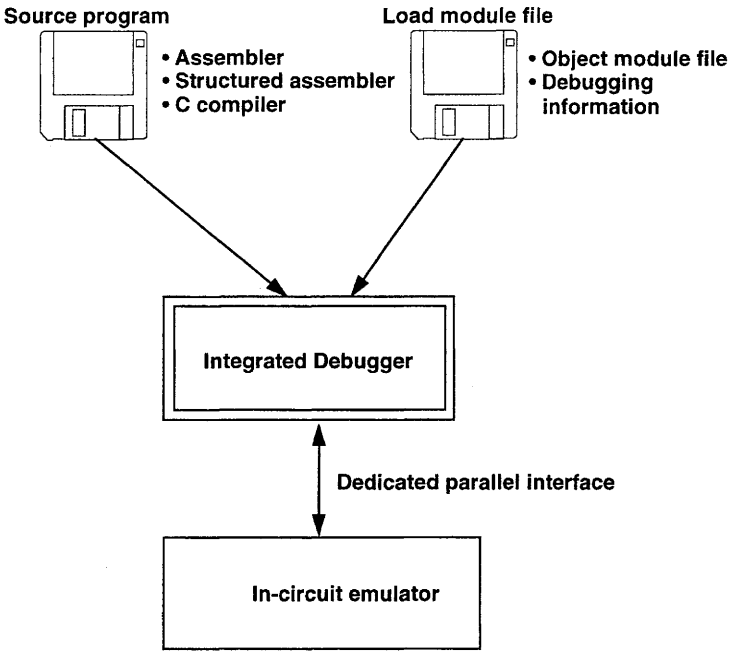
Remark ECC (Error Correcting Code)
Code to correct the data written to the internal PROM if the data includes an error.

1.2.9 Integrated debugger

The integrated debugger for the 78K/IIIseries is a software tool which displays the data from source programs, registers and memories in their respective windows and performs debugging.

The debugger downloads the load module file output by the linker to the in-circuit emulator (IE) of the target system. It can also perform debugging at the source level by reading the source program file.

Figure 1-16. Function of the Integrated Debugger



1.3 Reminders Before Program Development

Before beginning to develop a program, keep the following points in mind.

1.3.1 Number of files that can be input to linker

The number of files that can be input to the linker is 128.

1.3.2 Limits of number of symbols

The number of executable symbols is as follows:

	Number of Local Symbols	Number of PUBLIC Symbols
Assembler	2700 symbols[Note]	
Linker	2700 symbols x number of modules	About 3000 symbols

Note There is no limit depending on the type of a symbol. Undefined symbols are also counted in the number of symbols. Note that the number of symbols shown in this table is the total number of symbols where one symbol consists of eight characters.
If 2001 or more PUBLIC symbols are used, the execution speed slows down because a temporary file is created on the floppy disk.

1.3.3 Maximum performance characteristics of RA78K3

The maximum performance characteristics of the RA78K3 are listed below.

(1) Maximum Performance Characteristics of the Assembler

Table 1-1. Maximum Performance Characteristics of the Assembler

Item		Maximum Performance Characteristics
Number of symbols for which cross-reference list can be output		8000 symbols
Size of macro body for 1 macro reference		32 Kbytes
Number of segments in 1 file		100
Sum of the number of macros that can be referenced, and number of included files that can be specified, in 1 file		250
Sum of the number of macros that can be referenced, and number of included files that can be specified, in 1 included file		100
Relocation data Note 1		65535 items
Number of characters per line		218 characters Note 2
Symbol length	with -NS option	8 characters
	without -NS option	31 characters

Notes
1. "Relocation information" is the information to be passed to the linker if the assembler cannot resolve the symbol value. For example, if an externally referenced symbol is referenced by the MOV instruction, up to two piece of relocation information are created in the .rel file.
2. This does not include the carriage return and feed codes. If 219 characters or more are described on a line, a warning message is output and any characters at or over 219 are ignored.

(2) Maximum Performance Characteristics of Linker

Table 1-2. Maximum Performance Characteristics of Linker

Item	Maximum Performance Characteristics
Number of input modules	128 modules
Number of input libraries (limit of linker)	10 libraries

1.4 Features of RA78K3

The RA78K3 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 RA78K3 has a 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.

By describing the BR directive, the assembler generates the appropriate branch instruction according to the branch destination range. This is called the optimization function of branch instructions.

(3) Conditional assembly function

With this function, a 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

General-purpose registers can be represented by absolute names (R0, R1, RP0, etc.) or by function names (X, A, AX, etc.). With the 78K/III Series, when you describe a function name in a source program, you must always use a general-purpose register-select (RSS) directive. The RSS directive is provided to allow description of a function name as a general register representation in a source program.

[MEMO]

CHAPTER 2 PRODUCT OVERVIEW AND METHOD OF INSTALLATION

This chapter gives an overview of the files provided by the RA78K3 and explains how to install the RA78K3.

2.1 Host Machine and Supply Medium

Table 2-1. Delivery Medium and Recording Format of This Software Package

Host Machine	Operating System	Delivery Medium	Recording Format
PC-9800 series	MS-DOS™	3.5" 2HD FD	MS-DOS
IBM PC/AT and compatibles	PC DOS™	3.5" 2HC FD	PC DOS
HP9000 series 700	HP-UX™	DAT	tar
SPARCstation family	SunOS™	1/4" CGMT	tar
		3.5" 2HC FD	tar
RISC NEWS	NEWS-OS™	3.5" 2HC FD	tar

Be sure to observe the following points in order to execute the programs of the RA78K3 correctly on a PC-9800 series, IBM PC/AT and compatible machine.

Cautions 1. When a PC-9800 series is used as the host machine, each program of the RA78K3 runs normally on MS-DOS of NEC's PC-9800 series.
NEC takes no responsibility for program execution on an MS-DOS version commercially available.

2. Set FILES to 20 or more in the CONFIG.SYS.

3. The minimum memory size necessary for executing the assembler is 400 KB. This memory size increases if macros are used, and assembly may not be performed with 400 KB only in some cases. Therefore, allocate as large a vacant area of conventional memory as possible.

4. If the area of environmental variables runs short, specify the /E option, like SHELL = C:\COMMAND.COM C:\ /P /E:2048, to increase the area of environmental variables.

2.2 Contents of Media

2.2.1 For the PC-9800 series or IBM PC/AT and compatibles

The first supply medium stores the execution format of the assembler package and the second medium stores a device file[Note] in the following directory configuration.

[Outline of supply media]

First medium

\	Executable format of assembler package
	Sample program

Second medium

\DF78310	Device file for μPD78312, 78312A Subseries
\DF78320	Device file for μPD78322, 78328 Subseries
\DF78330	Device file for μPD78334 Subseries
\device\78312	Interrupt and macro service area allocating macro definition file for μPD78312, 78312A Subseries
\device\78322	Interrupt and macro service area allocating macro definition file for μPD78322 Subseries
\device\78328	Interrupt and macro service area allocating macro definition file for μPD78328 Subseries
\device\78334	Interrupt and macro service area allocating macro definition file for μPD78334 Subseries

Note The device files of the following subseries are optional.

- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

2.2.2 For the HP9000 series 700, SPARCstation family, and RISC NEWS

The supply media store the execution format of the assembler package and device file[Note] in the following directory configuration.

[Outline of supply media]

/	Executable format of assembler package Sample program
/df78310	Device file for μPD78312, 78312A Subseries
/df78320	Device file for μPD78322, 78328 Subseries
/df78330	Device file for μPD78334 Subseries
/device/78312	Interrupt and macro service area allocating macro definition file for μPD78312, 78312A Subseries
/device/78322	Interrupt and macro service area allocating macro definition file for μPD78322 Subseries
/device/78328	Interrupt and macro service area allocating macro definition file for μPD78328 Subseries
/device/78334	Interrupt and macro service area allocating macro definition file for μPD78334 Subseries

Note The device files of the following subseries are optional.

- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

2.3 Installation

2.3.1 For PC-9800 series or IBM PC/AT and compatibles

Change the directory from the current directory to the directory to which you wish to install the RA78K3.

Using the MS-DOS or PC DOS copy command, copy the execution format of the assembler and then the device file for the device you wish to use.

The following is an example of the installation procedure when the RA78K3 assembler package and μPD78312, 78312A, Subseries device file are read from drive A: and installed to C:\ra78K3.

X>cd C :\ra78K3	Change the directory from the current directory to the directory to which you wish to install the RA78K3. (Example: C:\ra78K3)
C>copy A:*.*	Copy file from the drive in which it is installed (Example: A: \).
C>xcopy/e A :\	Copy file from the drive in which it is installed.
C>copy C:\ra78k3\df78310*.*	Copy the device file from the subdirectory (C:\ra78k3\df78310) to the directory storing the execution format of the assembler package (C:\ra78k3).

Caution Copy the device file (dxxx.78k) to the directory that stores the executable format of the assembler package (ra78k3.exe, etc.).

2.3.2 For HP9000 series 700, SPARCstation family, RISC NEWS

Change the directory from the current directory to the directory to which you wish to install the RA78K3.

Use the tar command to copy the file for each directory structure. Next, move the device file to the current directory.

The following is an example of the installation procedure when the RA78K3 assembler package is read from tape device /dev/rct/c0 and installed to /ra78K3.

$cd/ra78K3	Change the directory from the current directory to the directory to which you wish to install the RA78K4.
$tar -xfv /dev/rct/c0	Copy the file from the device in which it is installed.
$mv df*/*.	Move the contents of df783* subdirectory to the current directory.

Caution Copy the device file (dxxx.78k) to the directory that stores the executable format of the assembler package (ra78k, etc.).

2.4 File Organization

2.4.1 For the PC-9800series or IBM PC/AT and compatibles

The organization of files in the assembler package after installation is as follows.

__.\	
ra78K3.exe	Executable format of the assembler
st78K3.exe	Executable format of the structured assembler/preprocessor
lk78K3.exe	Executable format of the linker
oc78K3.exe	Executable format of the object converter
lcnv78K3.exe	Executable format of the list converter
lb78K3.exe	Executable format of the librarian
eccgen.exe	Executable format of the ECC generator
ra78k3.hlp	Help file of the assembler
st78k3.hlp	Help file of the structured assembler/preprocessor
lk78k3.hlp	Help file of the linker
oc78k3.hlp	Help file of the object converter
lcnv78k3.hlp	Help file of the list converter
lb78k3.hlp	Help file of the librarian
eccgen.hlp	Help file of the ECC generator
ra78k3.is1	Table file of instruction set definitions used by the assembler
ra78k3.is2	
ra78k3.is3	
ra78k3.is4	
ra78k3.is5	
78k3main.asm	
78k3sub.asm	
test1.s	
test2.s	
testinc.s	
st.bat	
d3xx.78k	Copy the device file[Note] to this directory.
\df78310\d31x.78k	
\df78320\d32x.78k	
\df78330\d33x.78k	
\device\78312\intms.def	
\device\78322\intms.def	
\device\78328\intms.def	
\device\78334\intms.def	

Note The device file of the following subseries are optional.

- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

Cautions 1. Copy the device file (dxxx.78k) to the directory that stores the executable format of the assembler package (ra78k3.exe, etc.).

2. If you intend to use the C compiler, it is recommended that you install the assembler package, screen debugger, integrated debugger, system simulator, and device file in the directory in which the C compiler package is installed in an executable format. If you do not intend to use the C compiler, it is recommended that you install the screen debugger, integrated debugger, system simulator, and device file in the directory in which the executable format of the assembler package is installed.

2.4.2 For the HP9000 series 700, SPARCstation family and RISC NEWS

The organization of files after installation is as follows.

___./	
ra78K3	Execution format of the assembler
st78K3	Execution format of the structured assembler/preprocessor
lk78K3	Execution format of the linker
oc78K3	Execution format of the object converter
lcnv78K3	Execution format of the list converter
lb78K3	Execution format of the librarian
ra78k3.hlp	Help file of the assembler
st78k3.hlp	Help file of the structured assembler/preprocessor
lk78k3.hlp	Help file of the linker
oc78k.hlp	Help file of the object converter
lcnv78k3.hlp	Help file of the list converter
lb78k3.hlp	Help file of the librarian
ra78k3.is1	Table file of instruction set definitions used by the assembler
ra78k3.is2	
ra78k3.is3	
ra78k3.is4	
ra78k3.is5	
78k3main.asm	
78k3sub.asm	
test1.s	
test2.s	
testinc.s	
st.sh	
d3xx.78k	Copy the device file[Note] to this directory.
/df78310/d31x.78k	
/df78320/d32x.78k	
/df78330/d33x.78k	
/device/78312/intms.def	
/device/78322/intms.def	
/device/78328/intms.def	
/device/78334/intms.def	

Note The device file of the following subseries are optional.

- μPD78352A Subseries
- μPD78356 Subseries
- μPD78366 Subseries
- μPD78366A Subseries
- μPD78372 Subseries

Caution Copy the device file (dxxx.78k) to the directory that stores the executable format of the assembler package (ra78k, etc.).

2.5 Environment Setting

2.5.1 Environmental variable

Set the following environmental variables during work.

PATH:	Specifies the directory that stores the executable format of the assembler.
TMP:	Specifies the directory where a temporary file is to be created (this specification is valid only for the PC-9800 series or IBM PC/AT and its compatible machines).
INC78K3:	Specifies the directory from which the include file is to be searched.
LIB78K3:	Specifies the directory from which a library is to be searched if the library is used.

[Example]

For the PC-9800 series, or IBM PC/AT and its compatible machines

```
PATH=%PATH%:C:\ra78k3
set TMP=C:\
set INC78K3=C:\myincs
set LIB78K3=C:\mylibs
```

For the HP9000 series 700, SPARCstation family, or RISC NEWS

```
Example of using csh
    set path= ($path /ra78k3)
    setenv INC78K3 /myincs
    setenv LIB78K3 /mklibs
Example of using sh
    PATH=$PATH:/ra78k3
    INC78K3=/myincs
    LIB78K3=/mylibs
    export PATH INC78K3 LIB78K3
```

[MEMO]

CHAPTER 3 EXECUTING THE RA78K3

This chapter shows the procedure for executing the RA78K3. By actually executing each program of the RA78K3 according to the execution procedure explained in this chapter, you can become accustomed to the operation of the RA78K3.

All examples of operation from this chapter forward are based on use with the PC-9800 series (MS-DOS).

3.1 Before Executing the RA78K3

3.1.1 Verifying the contents of the disk

Verify that the RA78K3 system disk contains all of the files introduced in 2.4, "File Organization."

3.1.2 Sample programs

Among the files stored on the system disk are [78K3MAIN.ASM] and [78K3SUB.ASM]. These files are a sample program for use in verifying the operation of the assembler package.

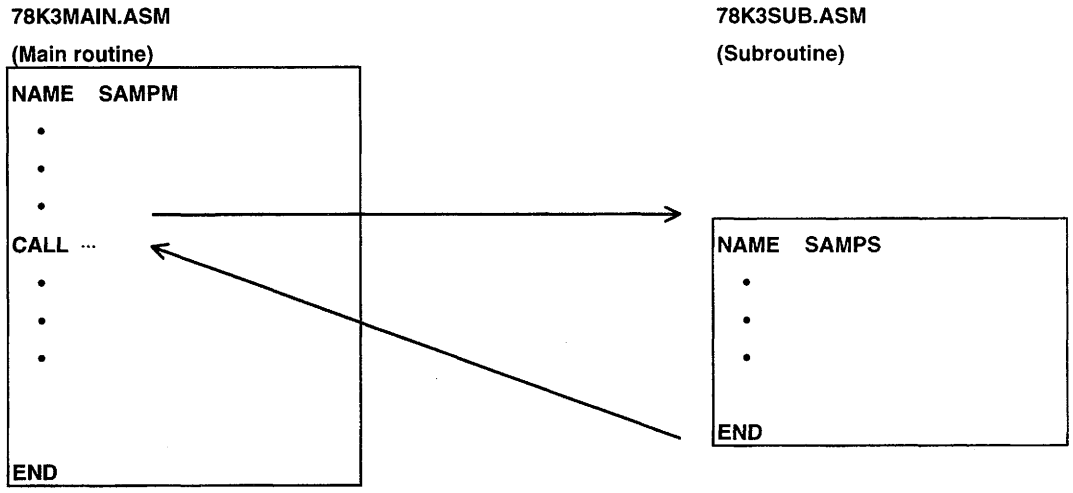
In later assembler operation, these files will be input to the assembler as source program files.

The following is a simple explanation of the contents of the sample programs. These programs consist of hexadecimal data converted to ASCII code. The program consists of two modules, a main routine and a subroutine.

The name of the main routine module is SAMPM, and it is stored in (78K3MAIN.ASM).

The name of the subroutine module is SAMPM, and it is stored in (78K3SUB.ASM).

Figure 3-1. Structure of the Sample Program



■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

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

        MOV     HDTSA,#1AH
        MOVG    HL,#HDTSA           ;set hex 2-code data in HL registor

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

        BR      $$

        END
```

■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

        CSEG
CONVAH: 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
```

Remarks 1. This sample program is a reference program, prepared for the purpose of teaching you about the functions and operation of the RA78K3. It cannot be used as an application program.

2. This sample program does not operate the default settings of the register set selection flag (RSS) or the register bank selection flags (RBS0 to RBS2). The settings for these items are therefore as follows.

Register bank	0 (0FEF0H to 0FEFFH)
RSS flag	0

3.2 Procedure for Executing the RA78K3

This section introduces the basic procedure for executing the RA78K3.

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

Input the following on the command line.

```
C>ra78k3 78k3main.asm
```

The following message is output to the display.

```
78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(2) Check the contents of drive C.

The assembler outputs the object module file (78K3MAIN.REL) and the assemble list file (78K3MAIN.PRN).

If the option -E is specified during assembly, the assembler outputs an error list file (a list of the lines containing assembly errors and the contents of their error messages).

(3) Assemble the sample program 78K3SUB.ASM. Input the following on the command line.

```
C>ra78k3 78k3sub.asm
```

The following message is output to the display.

```
78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(4) Check the contents of drive C.

The assembler outputs the object module file (78K3SUB.REL) and the assemble list file (78K3SUB.PRN).
During assembly, if the option -E is specified, the assembler outputs an error list file.

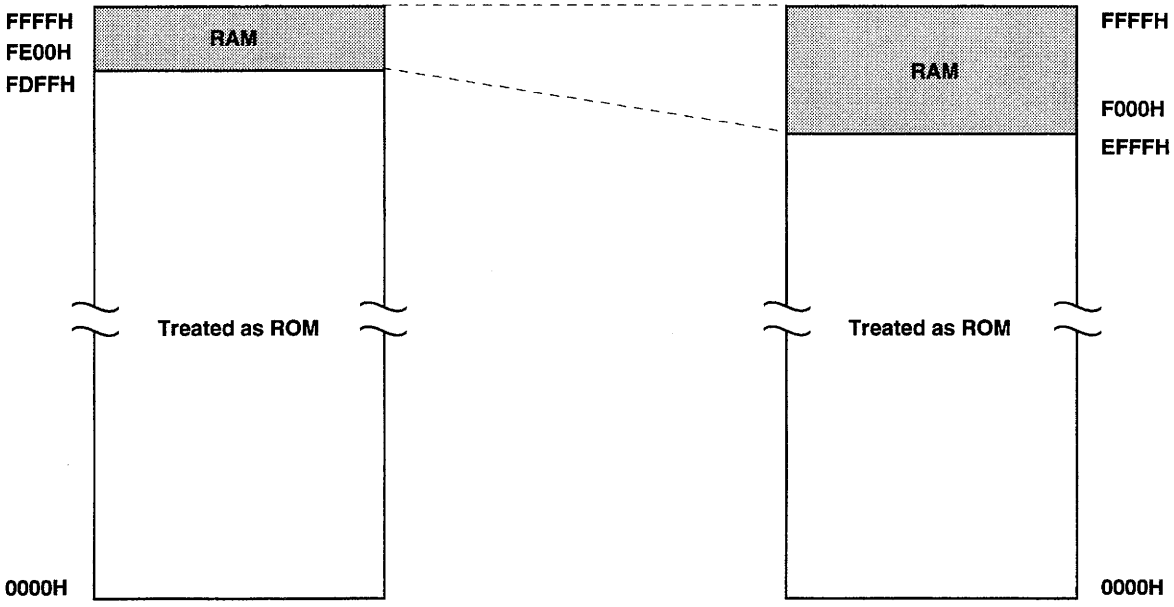
(5) Create a directive file.

A directive file is a file which indicates the location of segments for the linker.
Create a directive file when you need to expand the default ROM/RAM area or define a new memory area.
You will also need to create a directive file when you wish to locate segments not defined as absolute segments within a source module file to a specific address in memory.
During linking, use the -D option to enter the directive file to the linker.

Example 1. The default RAM area is expanded by a device without internal ROM (such as μPD78310A). The following is described in the directive file.

```
MEMORY ROM: (0000H,0F000H)
MEMORY RAM: (0F000H,1000H)
```

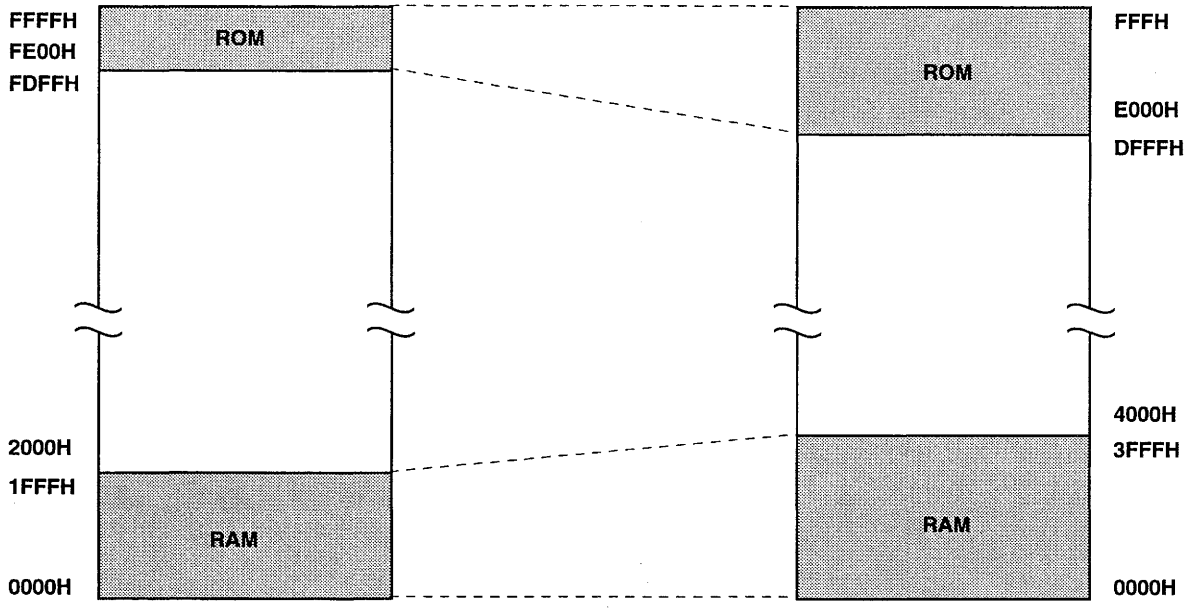
Figure 3-2. Link Directive 1



Example 2. In this example, the ROM/RAM area is expanded by a device with internal ROM such as μPD78312A and segment CSEG1 is located to address 2000H.
The following is described in the directive file.

```
MEMORY ROM: (0H,4000H)
MEMORY RAM: (0E000H,2000H)
MERGE CSEG1: AT (2000H)
```

Figure 3-3. Link Directive 2



(6) As the result of the assembly, the output object module files [78K3MAIN.REL] and [78K3SUB.REL] are linked.

Enter 78K3.DR as the directive file.

Enter the following on the command line.

```
C>lk78K3 78k3main.rel 78k3sub.rel -d78k3.dr -o78k3.lnk -p78k3.map
                                  └──This is not necessary if the directive file is not specified.
```

The following message is output to the display.

```
78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(7) Check the contents of drive C.

The linker outputs the load module file (78K3.LNK) and the link list file (78K3.MAP).

If the option -E is specified during linking, the linker outputs an error list file.

(8) As the result of linking, the output load module file [78K3.LNK] is converted to a HEX-format file.

Enter the following on the command line.

```
C>oc78k3 78k3.lnk
```

The following message is output on the display.

```
78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and      0 warning(s) found.
```

(9) Check the contents of drive C.

The object converter outputs the HEX-format object module file (78K3.HEX) and the symbol table file (78K3.SYM).

(10) Create a library file as follows.

Register the object module file [78K3SUB.REL] output by the assembler as a library file.

Create the file (78k3.job) using an editor.

- Contents of 78k3.job

```
Create 78k3. lib
Add 78k3. lib 78k3sub. rel
exit
```

Enter the following on the command line.

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

The following message is output on the display.

```
78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
*create 78k3.lib
*add 78k3.lib 78k3sub.rel
*exit
```

(11) Check the contents of drive C.

The librarian outputs the library file (78K3.LIB).

(12) Create an absolute assemble list as follows.

To create the absolute assemble list 78K3MAIN.ASM, input [78K3MAIN.REL], [78K3MAIN.ASM] and [78K3.LNK] to the list converter.

Enter the following on the command line.

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

The following message is output on the display.

```
List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
Conversion complete.
```

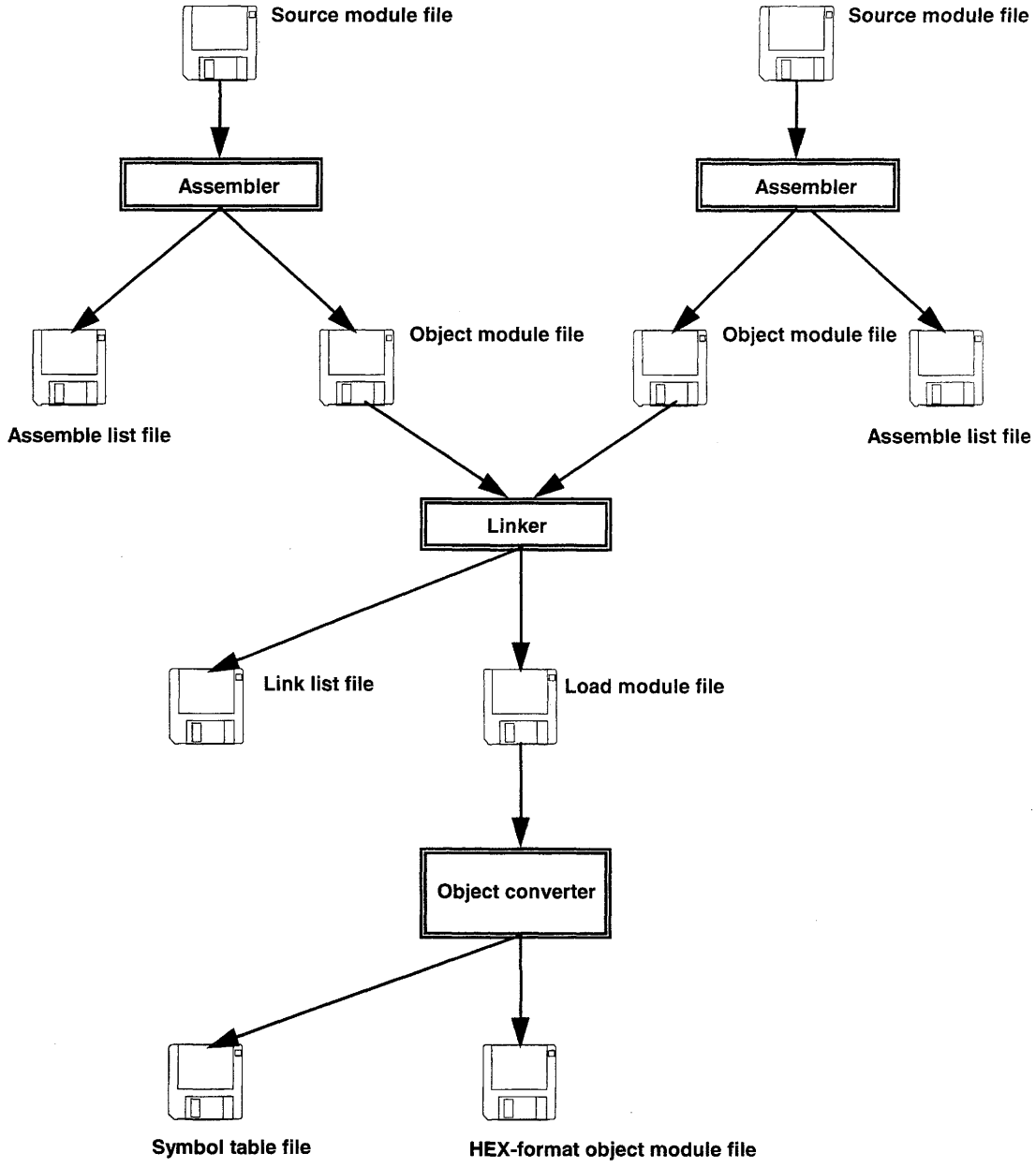
(13) Check the contents of drive C.

The list converter outputs the absolute assemble list file (78K3MAIN.P).

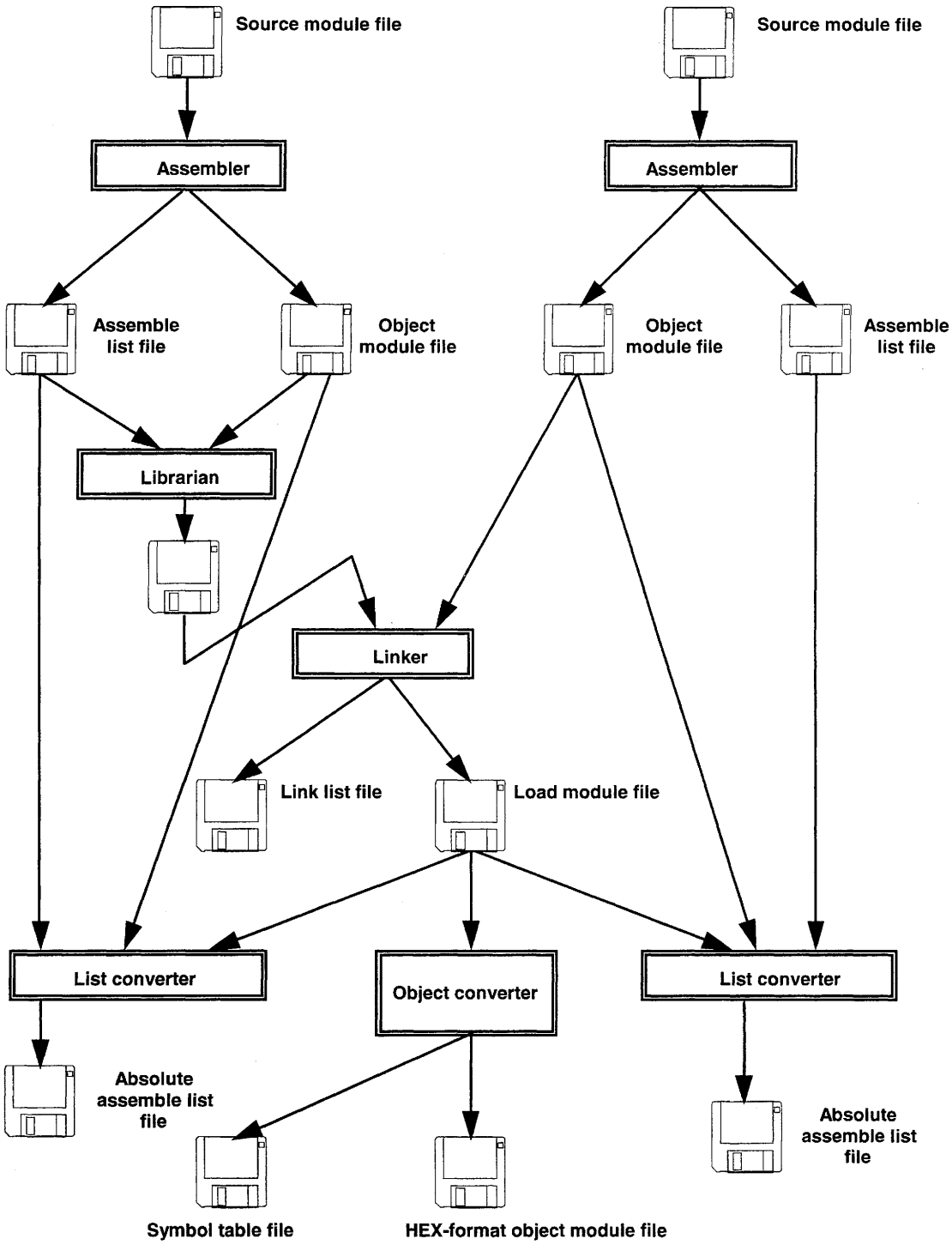
3.3 Summary of the RA78K3 Execution Procedure

The following is a brief summary of section 3.2, "Procedure for Executing the RA78K3."

(1) Execution procedure 1



(2) Execution procedure 2



[MEMO]

CHAPTER 4 ASSEMBLER

The assembler inputs source module files described in the assembly language for 78K/III Series microcontrollers and converts them into machine language coding.

The assembler also outputs list files such as assemble list files and error list files.

If assembly errors occur, an error message is output to the assemble list file and error list file to clarify the cause of the error.

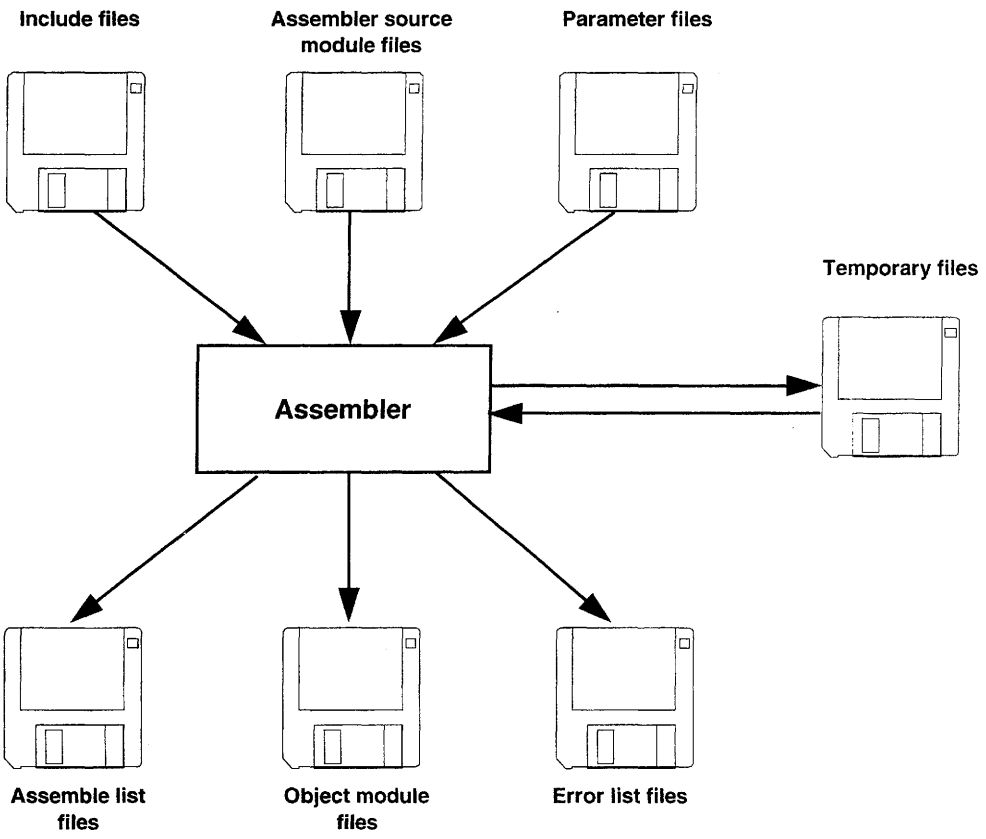
4.1 Assembler Input and Output Files

The following table shows the input/output files of the assembler.

Table 4-1. Assembler Input and Output Files

Type	File Name	Explanation	Default File Type
Input files	Assembler source module files	• These are source module files described in assembly language for 78K/III Series microcontrollers •These files are created by the user.	.ASM
	Include files	• These files are used for reference with assembler source module files. • These are files described in assembly language for 78K/III Series microcontrollers. • These files are created by the user.	—
	Parameter files	• These files contain the parameters for the executed files. • These files are created by the user.	.PRA
Output files	Object module files	• These are binary files including relocation data and symbol data regarding machine language data and machine language location addresses.	.REL
	Assemble list files	• These are files containing assembly data such as assemble lists and cross-reference lists.	.PRN
	Error list files	• These are files containing error data generated during assembly.	.ERA
Input and output files	Temporary files	• These are files created automatically by the assembler for assembly purposes. Temporary files are deleted when assembly ends.	RAxxxxxx.$$n (n=1 to 4)

Figure 4-1. Files Input and Output by the Assembler



4.2 Functions of the Assembler

(1) The assembler reads source module files and converts them from assembly language files into machine language files.

(2) If errors occur, the assembler outputs an abort error. If it finds the described error in the source module, the assembler outputs a "fatal error" or "warning error" message.
If an "abort error" or "fatal error" message is output, the object module file cannot be output normally. However, even if a fatal error has occurred the object module file can be output in case of specifying option -J.

(3) The assembler performs assembly according to the assembler option specified at assembler startup. For a detailed explanation of the assembler options, see 4.4, "Assembler Options."

(4) If assembly is completed correctly the assembler outputs an "Assembly Finished" message and returns control to the operating system.

(5) Maximum performance characteristics of the assembler package are as follows.

Item		Limit
Symbol length	-NS option specified	8 characters
	-NS option not specified	31 characters
Number of characters per line		218 characters[Note 1]
Number of segments	?ASEGn[Note 2]	20
	Other than ?ASEGn	80

Notes 1. This does not include carriage returns and feed codes. If 219 characters or more are described on one line, a warning message is output and all characters after the 218th character are ignored.

2. Absolute segments whose segment name is unspecified will be assigned the default segment name '?ASEGn' (n=1 to 20, source description sequence).

4.3 Assembler Startup

4.3.1 Assembler startup

Two methods can be used to start up the assembler.

(1) Command-line startup

```
X>[path name] RA78K3 [Δoption] ··· Δsource module file name [Δoption] ··· [Δ]
 |      |      |     |                 |                    |
(1)    (2)    (3)   (4)               (5)                  (4)
```

(1) Current drive name
(2) Current directory name
(3) Command file name of the assembler
(4) Enter detailed instructions for the operation of the assembler.
When specifying two or more assembler options, separate the assembler options with a blank space. For a detailed explanation of assembler options, see 4.4, "Assembler Options."
(5) File name of source module to be assembled

Example C>ra78k3 78k3main.asm -e -np

(2) Startup from a parameter file

Use the parameter file when the data required to start up the assembler will not fit on the command line, or when repeating the same assembler option for two or more assembly operations.
To start up the assembler from a parameter file, specify the parameter file option (-F) on the command line.

Start up the assembler from a parameter file as follows.

X>RA78K3 [Δsource module file] Δ-F parameter file name

(1) (2)

(1) A file which includes the data required to start up the assembler
(2) Parameter file (the specified option)
Create the parameter file using an editor.

The rules for describing the contents of a parameter file are as follows.

[[[Δ] option [Δoption] ··· [Δ]Δ]] ···

If the source module file name is omitted from the command line, only 1 source module file name can be specified in the parameter file.
The source module file name can also be described after the option.
Describe in the parameter file all assembler options and output file names specified in the command line.
For a detailed explanation of parameter files, see 4.4.3, "Explanation of assembler options."

Example Create the parameter file (78K3MAIN.PRA) using an editor.

- Contents of 78K3MAIN.PRA

```
;Parameterfile
78k3main.asm -osmple.rel
-psample.prn
```

- Use parameter file (78K3MAIN.PRA) to start up the assembler.

C>ra78k3 -f78k3main.pra

4.3.2 Execution start and end messages

(1) Execution start message

When the assembler is started up, an execution startup message appears on the display.

```
78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
```

(2) Execution end message

If it detects no assembly errors resulting from the assembly, the assembler outputs the following message to the display and returns control to the operating system.

```
Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

If it detects an assembly error resulting from the assembly, the assembler outputs the error number to the display and returns control to the operating system.

```
Pass1 Start
78K3MAIN.ASM(15) : F201 Syntax error
Pass2 Start
78K3MAIN.ASM(15) : F201 Syntax error

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

If the assembler detects a fatal error during assembly which makes it unable to continue assembly processing, the assembler outputs a message to the display, cancels assembly and returns control to the operating system.

Example 1. A nonexistent source module file is specified.

```
C>ra78k3 sample.asm

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

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

In the above example, a nonexistent source module file is specified. An error results and the assembler aborts assembly.

Example 2. A nonexistent assembler option is specified.

```
C>ra78k3 78k3main.asm -b

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

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

In the above example, a nonexistent assembler option is specified. An error results and the assembler aborts assembly.

When an error message is displayed and assembly is aborted, look for the cause in chapter 11, "Error Messages" and take action accordingly.

4.4 Assembler Options

4.4.1 Types of assembler options

The assembler options are detailed instructions for the operation of the assembler. Assembler options are classified into 14 types.

Table 4-2. Assembler Options (1/2)

Number	Classification	Option	Explanation
1	Specify device type	-C	Specifies the device type of the target device.
2	Specify object module file output	-O	Specifies the output of an object module file.
		-NO	
3	Specify forced object module file output	-J	Forces output of an object module file.
		-NJ	
4	Specify debug data output	-G	Outputs debugging data to an object module file.
		-NA	
		-GA	
		-NGA	
5	Specify length of symbol name	-S	Extends length of a symbol name.
		-NS	
6	Specify symbol name case	-CA	Ignores a distinction between upper case and lower case in symbol names.
		-NCA	
7	Specify include file read path	-I	Reads from the path specified in an include file.
8	Specify assemble list file output	-P	Specifies output of an assemble list file.
		-NP	
9	Specify assemble list file data	-KA	Outputs an assemble list into an assemble list file.
		-NKA	
		-KS	Outputs a symbol list into an assemble list file.
		-NKS	
		-KX	Outputs a cross-reference list into an assemble list file.
		-NKX	

Number	Classification	Option	Explanation
10	Specify assemble list file format	-LW	Changes the number of characters that can be printed in 1 line in an assemble list file.
		-LL	Changes the number of lines that can be printed in 1 page in an assemble list file.
		-LH	Outputs the character string specified in the header of an assemble list file
		-LT	Changes the number of spaces in a tab.
		-LF	Inserts a line feed code at the end of an assemble list file.
		-NLF	
11	Specify error list file output	-E	Outputs an error list file.
		-NE	
12	Specify parameter file	-F	Inputs the input file name and assembler options from a specified file.
13	Specify path for temporary file creation	-T	Creates a temporary file in a specified path.
14	Specify help	--	Displays a help message on the display.

This table introduces the assembler options. When actually using the assembler options, refer to Appendix E.1, "List of Assembler Options".

4.4.2 Order of precedence of assembler options

The following table indicates which assembler option takes precedence when two assembler options are specified at the same time.

Table 4-3. Order of Precedence of Assembler Options

	-NO	-NP	-NKA	-NKS	-KX	-NKX	--
-J	×						×
-G	×						×
-P			Δ	Δ		Δ	×
-KA		×					×
-KS		×			×		×
-KX		×					×
-LW		×					×
-LL		×					×
-LH		×					×
-LT		×					×
-LF		×					×

← Horizontal axis

↑ Vertical axis

[Items marked with an X]

When the option in the horizontal axis is specified, the option shown in the vertical axis option is unavailable.

Example C>ra78k3 78k3main.asm -no -lw80 -lf

The options -LW and -LF are unavailable.

[Items marked with a Δ]

When all three of the options in the horizontal axis are specified, the option shown in the vertical axis option is unavailable.

Example C>ra78k3 78k3main.asm -p -nka -nks -nkx

The options -NKA, -NKS and -NKX are all specified at the same time, so option -P is unavailable.

When an option and its 'N' counterpart are specified at the same time (for example, both -O and -NO), only the last of the 2 options is available.

Example C>ra78k3 78k3main.asm -o -no

The option -NO is specified after -O, so option -O is unavailable and -NO is available.

Options not described in Table 4-3 have no particular effect on other options. However, when the help option '--' is specified, all other options become unavailable.

4.4.3 Explanation of assembler options

This section contains detailed explanations of each assembler option.

(1) Specify device type (-C)

Description format: -C device type
Default value: Cannot be omitted

[Function]

Option -C specifies the device type of the target device.

[Application]

Be sure to use the -C option. The assembler performs assembly for the target device and generates an object code for that device.

[Description]

For the correspondence between the -C option and model specification, refer to the following:

1) With μPD78312, 78312A, 78322, 78328, and 78334 Subseries
   Refer to D.2, "Correspondence between Target Device and Device File".
2) With μPD78352A, 78356, 78366, 78366A, and 78372 Subseries
   Refer to the separately available document on the device file (Notes on Using DF783xx Device File).

[Note]

Option -C cannot be omitted. However, if a control instruction with the same function is described at the beginning of the source module, command-line specification can be omitted.

```
▽$▽PROCESSOR▽(▽device type▽)
▽$▽PC▽(▽device type▽)        ;Abbreviated form
```

For information on control instructions, read Chapter 4, "Control Instructions," in the language manual.

[Example]

Example 1. Specify the option -C on the command line as follows.

```
C>ra78k3 –c310 78k3main.asm

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPD78310
 Device file : V1.00

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

Example 2. Specify in the source module and start the assembler.

```
$        PROCESSOR(310)

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

         PUBLIC   MAIN,START
         EXTRN    CONVAH
           .
           .
           .
```

Specifying the target device on the command line may be omitted.

```
C>ra78k3 78k3main.asm

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPD78310
 Device file : V1.00

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

Example 3. Specify different device in the source madule and on command line and start the assembler.

```
$        PROCESSOR(310)

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


         PUBLIC  MAIN,START
         EXTRN   CONVAH
           .
           .
           .
```

C>ra78k3 –c320 78k3main.asm

```
78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
78K3MAIN.ASM(1) : W702 Duplicate PROCESSOR option and control
Pass2 Start
78K3MAIN.ASM(1) : W702 Duplicate PROCESSOR option and control

 Target chip : uPD78320
 Device file : Vx.xx

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

The device specified on the command line takes precedence.

(2) Specify object module file output (-O/-NO)

Description format : -O [output-file-name]
: -NO
Default value : -O (input file name).REL

[Function]

1) Option -O specifies the output of an object module file. It also specifies the location to which it is output and the file name.
2) Option -NO specifies that no object module file is output.

[Application]

Use the option -O to specify the location to which an object module file is output or to change its file name.

Specify the option -NO when performing assembly only to output an assemble list file. This will shorten assembly time.

[Description]

1) Even if the option -O is specified, if a fatal error occurs the object module file cannot be output.
2) If the drive name is omitted when the option -O is specified, the object module file will be output to the current drive.
3) If the output file name is omitted when the option -O is specified, the output file name will be 'input file name.REL'.
4) If both the options -O and -NO are specified at the same time, the option specified last takes precedence.

[Example]

Example 1. Specify output of object module file (SAMPLE.REL).

```
C>ra78k3 78k3main.asm –osample.rel

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Example 2. Specify both options –NO and –O.

```
C>ra78k3 78k3main.asm –no -o

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Option –NO is invalid, and option –O is valid.

(3) Specify forced object module file output (-J/-NJ)

Description format : -J
: -NJ
Default value : -NJ

[Function]

1) Option -J specifies that the object module file can be output even if a fatal error occurs.
2) Option -NJ makes option -J unavailable.

[Application]

Normally, when a fatal error occurs, the object module file cannot be output. When you wish to execute the program with a notice that a fatal error has occurred, specify option -J to output the object module file.

[Description]

1) When option -J is specified, the object module file will be output even if a fatal error occurs.
2) If both options -J and -NJ are specified at the same time, the option specified last takes precedence.

[Example]

Specify output of object module file even if a fatal error occurs.

```
C>ra78k3 78k3main.asm -j

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(4) Specify debug data output (-G/-GA/-NG/-NGA)

This option controls the information created in the object depending on how debugging is to be performed.

Description format : -G
: -GA
: -NG
: -NGA
Default value : -G -GA

[Function]

1) Option -G specifies that debugging data (local symbol data) is to be added to an object module file.
2) Option -GA specifies that source debugging data is to be output to an object module file by the assembly.
3) Option -NG makes option -G unavailable.
4) Option -NGA makes option -GA unavailable.

[Application]

1) Use option -G when performing symbolic debugging of data that includes local symbol data.
2) Use option -GA when performing debugging at the source level of the assembler. To perform debugging at the source level, you will need the integrated debugger.
3) Use option -NG in the following 3 cases.
    1. Symbolic debugging of global symbols only
    2. Debugging without symbols
    3. When only the object is required (evaluation using PROM, etc.)

[Description]

1) If option -G is omitted, the debugging data information is not output.
2) If both options -G and -NG are specified at the same time, the option specified last takes precedence.
3) Option -GA takes precedence over other options regardless of the position in which it is specified.

[Note]

1) A control instruction with the same function as options –G, -GA, and -NG can be described at the beginning of a source module.

```
∇$∇DEBUG
∇$∇DG                    ;Abbreviated form
∇$∇DEBUGA
∇$∇NODEBUG
∇$∇NODG                  ;Abbreviated form
∇$∇NODEBUGA
```

2) When performing debugging at the source level of the compiler or structured assembler, do not use this option. The necessary control instruction is automatically output.

For information on control instructions, read Chapter 4, "Control Instructions," in the language manual.

[Example]

Specify addition of debug data to an object module file.

```
C>ra78k3 78k3main.asm -g

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(5) Specify length of symbol name (-S/-NS)

Description format : -S
: -NS
Default value : -S

[Function]

1) Option -S specifies that the recognizable length of a symbol name is to be extended to a maximum of 31 characters.
2) Option -NS makes option -S unavailable.

[Application]

If a symbol name is longer than 8 characters, the in-circuit emulator cannot load it to a symbol table.
This option is used to perform debugging using the source debugger (Integrated debugger:ID78K3) using a symbol name longer than 8 characters.

[Description]

1) When the option -S is specified, the assembler can recognize symbol names of up to 31 characters. It can also output symbol data to an object.
2) When the -NS option is omitted, the assembler can recognize symbol names of up to 31 characters.
3) If both options -S and -NS are specified at the same time, the option specified last takes precedence.

[Note]

When not using a source debugger, specify option –NS.

[Example]

Extend the recognizable length of a symbol name to 31 characters.

```
C>ra78k3 78k3main.asm -s

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(6) Specify symbol name case (-CA/-NCA)

Description format : -CA
: -NCA
Default value : -NCA

[Function]

1) Option -CA specifies that no distinction is made between uppercase and lowercase characters in a symbol name.
2) Option -NCA specifies that a distinction is made between uppercase and lowercase characters in a symbol name.

[Application]

Use option -CA when you need to ignore the distinction between upper case and lower case.

[Description]

1) When option -CA is specified, the assembler converts lowercase characters in a symbol name to uppercase and outputs them to an object.
2) When the -NCA option is specified, the assembler outputs the symbol name to an object without converting lowercase characters to uppercase.

[Note]

When not using a source debugger, specify option -CA.

[Example]

Specify that a distinction is made between uppercase and lowercase characters in a symbol name.

```
C>ra78k3 78k3main.asm -nca

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(7) Specify include file read path (-I)

Description format : -I path name [, path name] ··· (two or more path names can be specified)
Default value : Path specified by the environmental variable (INC78K3)
: Path contained in the source file when no path is specified.

[Function]

Option -I specifies input of an include file specified by ' $include' in a source module from a specified path.

[Application]

Use option -I to retrieve an include file from a certain path.

[Description]

1) Two or more path names can be specified at once by separating them with ','.
2) A space cannot be entered before or after the ','.
3) When two or more path names are specified following -I, or several -I options are specified, files specified with '$include' will be retrieved in the specified order. Thereafter, files will be retrieved in the default order.
4) If anything other than a path name is specified after -I, or if the path name is omitted, an abort error occurs.
5) If -I is used to specify 9 or more path names, an abort error occurs.

[Example]

Read an include file from SAMPLE in directory.

```
C>ra78k3 78k3main.asm -ib:\sample

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(8) Specify assemble list file output (-P/-NP)

Description format : -P [output-file-name]
: -NP
Default value : -P input file name.PRN

[Function]

1) Option -P specifies output of an assemble list file. It also specifies the destination and file name of the output file.
2) Option -NP makes option -P unavailable.

[Application]

1) Specify option -P to change the output destination or output file name of an assemble list file.
2) Specify option -NP when performing assembly only to output an object module file. This will shorten assembly time.

[Description]

1) A file name can be specified as a disk-type file name or as a device-type file name. However, only CON, PRN, NUL and AUX can be specified as device-type file names. If CLOCK is specified, an abort error will occur.
2) If the output file name is omitted when option -P is specified, the assemble list file name becomes 'input file name.PRN'.
3) If the drive name is omitted when option -P is specified, the assemble list file will be output to the current drive.
4) If both options -P and -NP are specified at the same time, the option specified last takes precedence.

[Example]

Example 1. Create an assemble list file (SAMPLE.PRN).

```
C>ra78k3 78k3main.asm -psample.prn

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Example 2. Output the assemble list file to printer.

```
C>ra78k3 78k3main.asm -pprn

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(9) Specify assemble list file data (-KA/-NKA, -KS/-NKS, -KX/-NKX)

(a) -KA/-NKA

Description format : -KA
: -NKA
Default value : -KA

[Function]

1) Option -KA outputs an assemble list into an assemble list file.
2) Option -NKA makes option -KA unavailable.

[Application]

Specify option -KA to output an assemble list.

[Description]

1) If both options -KA and -NKA are specified at the same time, the option specified last takes precedence.
2) If options -NKA, -NKS and -NKX are all specified, the assemble list file cannot be output.

[Example]

Example 1. Output an assembly list file.

```
C>ra78k3 78k3main.asm –ka –lw80

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Example 2. Reference 78K3MAIN.PRN.

```
78K/III Series Assembler Vx.xx                    Date:xx xxx xxxx Page:    1


Command:  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
```

(b) -KS/-NKS

Description format : -KS
: -NKS
Default value : -NKS

[Function]

1) Option -KS outputs an assemble list followed by a symbol list into an assemble list file.
2) Option -NKS makes option -KS unavailable.

[Application]

Specify option -KS to output a symbol list.

[Description]

1) If both options -KS and -NKS are specified at the same time, the option specified last takes precedence.
2) If options -KS and -KX are specified at the same time, -KS is ignored.
3) If options -NKA, -NKS and -NKX are all specified, the assemble list file cannot be output.

[Example]

Example 1. Output a symbol list.

```
C>ra78k3 78k3main.asm –ks –lw80

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Example 2. Reference 78K3MAIN.PRN. (The assemble list is output, followed by the symbol list.)

```
78K/III Series Assembler Vx.xx                Date:x xxx xxxx Page:    3

          Symbol Table List

 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

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(c) -KX/-NKX

Description format : -KX
: -NKX
Default value : -NKX

[Function]

1) Option -KX outputs an assemble list followed by a cross-reference list into an assemble list file.
2) Option -NKX makes option -KX unavailable.

[Application]

Specify option -KX to output a cross-reference list when you wish to know where and to what degree each symbol defined in a source module file is referenced in the source module, or when you wish to know such information as which line of the assemble list a certain symbol is referenced on.

[Description]

1) If both options -KX and -NKX are specified at the same time, the option specified last takes precedence.
2) If options -KS and -KX are specified at the same time, -KS is ignored.
3) If options -NKA, -NKS and -NKX are all specified, the assemble list file cannot be output.

[Note]

A control instruction with the same function as option -KX/-NKX can also be described at the beginning of a source module.

```
$XREF
$XR                    ;Abbreviated form
$NOXREF
$NOXR                  ;Abbreviated form
```

For information on control instructions, read Chapter 4, "Control Instructions," in the language manual.

[Example]

Example 1. Output a cross-reference list.

```
A>ra78k3 78k3main.asm –kx –lw80

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Example 2. Reference 78K3MAIN.PRN.(The assemble list is output, followed by a cross-reference list.)

```
78K/III Series Assembler Vx.xx                Date:x 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            13@     31
DATA                        DSEG         DATA         15#
HDTSA             FE20H     ADDR         DATA         16#     28      29
MAIN                0H      ADDR   PUB   CODE         12@     20#
SAMPM                       MOD                        3#
START               0H    R ADDR   PUB   ?CSEG        12@     20      23#
STASC             FE21H     ADDR         DATA         17#     33

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(10) Specify assemble list file format (LW, -LL, -LH, -LT, -LF/-NLF)

(a) -LW

Description format : -LW [number-of-characters]
Default value : -LW132 (80 characters in the case of display output)

[Function]

Option -LW changes the number of characters that can be printed in 1 line in a list file.

[Application]

Specify option -LW to change the number of characters that can be printed in 1 line in any type of list file.

[Description]

1) The range of number of characters that can be specified with option -LW is shown below.

   $72 \leq$ number of characters printed on 1 line ≤ 132

   If a numerical value outside this range, or something other than a numerical value, is specifi, an abort error occurs.
2) If the number of characters is omitted, 132 will be specified.
   However, when an assemble list file is output to display, 80 will be specified.
3) The specified number of characters does not include the carriage return and feed codes.
4) If option -NP is specified, option -LW is unavailable.

[Example]

Example 1. Omit the option –LW and output the assemble list to printer.

```
C>ra78k3 78k3main.asm -pprn

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

This references the assemble list.

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
```

Example 2. Specify 80 as the number of characters per line in an assemble list file.

```
C>ra78k3 78k3main.asm –lw80

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

This references the assemble list.

```
     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
```

(b) -LL

Description format : -LL [number-of-lines]
Default value : -LL66 (No page breaks in the case of display output)

[Function]

Option -LL changes the number of lines that can be printed in 1 page in an assemble list file.

[Application]

Specify option -LL to change the number of lines that can be printed in 1 page in an assemble list file.

[Descrription]

1) The range of number of lines that can be specified with option -LL is shown below.

   20 ≤ number of lines printed on 1 page ≤ 32767

   If a numerical value outside this range, or something other than a numerical value, is specified, an abort error occurs.
2) If the number of lines is omitted, 66 will be specified.
3) If the number of lines specified is 0, no page breaks will be made.
4) If option -NP is specified, option -LL is unavailable.

[Example]

Example 1. Specify 20 as the number of lines per page in an assemble list file.

```
C>ra78k3 78k3main.asm -ll20 –lw80

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Example 2. Reference 78K3MAIN.PRN.

```
78K/III Series Assembler Vx.xx                    Date:xx xxx xxxx Page:    1


Command:  78k3main.asm -lw80
Para-file:
In-file:  78K4MAIN.ASM
Obj-file: 78K3MAIN.REL
Prn-file: 78K3MAIN.PRN

      Assemble list

------------------------------------------------------------------------------

78K/III Series 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

------------------------------------------------------------------------------

78K/III Series Assembler Vx.xx                    Date:xx xxx xxxx Page:    3

                               *
    7     7                    ;*
                               *
    8     8                    ;*           main-routine
                               *
    9     9                    ;*
                               *
   10    10                    ;*****************************************
```

(c) -LH

Description format : -LH character string
Default value : None

[Function]

Option -LH specifies the character string printed in the title column of the header of an assemble list file.

[Application]

1) Specify option -LH to display a title that briefly explains the contents of an assemble list file.
2) By printing the title on each page, the contents of the assemble list file can be understood at a glance.

[Description]

1) Up to 60 characters can be specified in the title. The character string cannot include blank spaces.
2) If more than 61 characters are described, the first 60 characters will be recognized and no error message will be output.
   1 Japanese kanji or hiragana character is counted as 2 characters.
   If the maximum number of characters per line is 117 or less, the length of the effective character string changes as follows.

   Effective length = (Max. number of characters per line) - 58

3) If the length of the character string is not specified, an abort error will occur.
4) If option -NP is specified, option -LH is unavailable.
5) If the -LH option is omitted, the title column of the assemble list file will be blank.
6) The character set that can be described in the title column is as follows.

Table 4-4. Characters That Can Be Described as Titles

Character	In Command Line	In Parameter File
*?><|	Can be described if enclosed in " ".	Can be described. Interpreted in the same way as in the command line even if enclosed in " ".
;	Can be described if enclosed in " ".	Cannot be described. (Assumed to be a comment.)
#	Can be described.	Cannot be described. (Assumed to be a comment.)
" (double quotation mark)	Cannot be described as an effective character.	Cannot be described as an effective character.
00H	Cannot be described.	Cannot be described. However, it is interpreted as the end of the character string.
03H, 06H, 08H, 0DH 0EH, 10H, 15H, 17H 18H, 1BH, 7FH	Can be described.	Cannot be described. However, these will appear in the assemble list file as '!' (A single 0DH will not be output to the list.)
01H, 02H, 04H, 05H 07H, 0BH, 0CH, 0FH 11H, 12H, 13H, 14H 16H, 19H, 1CH, 1DH 1EH, 1FH	Can be described. However, these will appear in the assemble list file as '!'	Can be described. However, these will appear in the assemble list file as '!'
1AH	Can be described. However, this will appear in the assemble list file as '!'	Cannot be described. (end of file)
Alphabetic characters	Uppercase and lowercase characters are input as is.	Uppercase and lowercase characters are input as is.
Other	Can be described.	Can be described.

Remark If an asterisk (*) on the startup line is not a target for World Card expansion, it can be described even if it is not enclosed in " ".

[Note]

A control instruction with the same function as option -LH can also be described at the beginning of the startup line.

```
∇$∇TITLE∇(∇'character string'∇)
∇$∇TT∇(∇'character string'∇)          ;Abbreviated form
```

For information on control instructions, read Chapter 4, "Control Instructions," in the language manual.

[Example]

Print the title in the header of an assemble list file.

```
C>ra78k3 78k3main.asm –lhRA78K3 MAINROUTINE

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

This references 78K3MAIN.PRN.

```
78K/III Series Assembler Vx.xx RA78K3_MAINROUTINE     Date:xx xxx xxxx Page:    1
                                        |
                                        └── Title
Command:  78k3main.asm -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
   14    14
   15    15 ----                DATA        DSEG    AT 0FE20H
   16    16 FE20                HDTSA:      DS      1
   17    17 FE21                STASC:      DS      2
```

(d) -LT

Description format : -LT [number-of-characters]
Default value : -LT8

[Function]

Option -LT performs tabulation processing by specifying a number of characters for any type of list for which to substitute and output a number of blank spaces for the HT (horizontal tabulation) code in a source module.

[Application]

When specifying a small number of characters per line for any type of list using option -LW, specify option -LT to insert a tab instead of a series of blank spaces, thus saving on the number of characters used.

[Description]

1) The range of number of characters that can be specified with option -LT is shown below.

   $0 \leq$ number of characters that can be specified ≤ 8

   If a numerical value outside this range, or something other than a numerical value, is specified, an abort error occurs.
2) If -LT0 is specified, tabulation processing will not be performed, and a tabulation code will be output.
3) If option -NP is specified, option -LT is unavailable.

[Examples]

Example 1. Omit option –LT.

```
C>ra78k3 78k3main.asm –lw80

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

This references 78K3MAIN.PRN.

```
78K/III Series Assembler Vx.xx                    Date:xx xxx xxxx Page:   1


Command:  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
```

Example 2. 1 blank is specified using the HT code.

```
C>ra78k3 78k3main.asm –lt1

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

This references 78K3MAIN.PRN.

```
78K/III Series Assembler Vx.xx                    Date:xx xxx xxxx Page:   1


Command:  78k3main.asm -lt1
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  ;set hex 2-code data in HL register
   30    30
   31    31 0014  R280000        CALL !CONVAH                  ;convert ASCII <- HEX
```

Remark The number of blanks entered by the HT code is 1.

(e) –LF/-NLF

Description format : -LF
: -NLF
Default value : -NLF

[Function]

1) Option -LF inserts a form feed (FF) code at the end of an assemble list file.
2) The -NLF option makes the -LF option unavailable.

[Application]

If you wish to add a page break after the contents of an assemble list file are printed, specify option -LF to add a form feed code.

[Description]

1) If option -NP is specified, option -LF is unavailable.
2) If both options -LF and -NLF are specified at the same time, the option specified last takes precedence.

[Example]

Example Add a form feed code at the end of an assemble list file.

```
C>ra78k3 78k3main.asm -pprn -lf

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(11) Specify error list file output (-E/-NE)

Description format : -E [output file name]
: -NE
Default value : -NE

[Function]

1) Option -E outputs an error list file, and specifies the output destination and output file name of the error list file.
2) The -NE option makes the -E option unavailable.

[Application]

1) Specify option -E to save an error message into a file.
2) Specify option -E to change the output destination and output file name of the error list file.

[Description]

1) The error list file can be saved as a disk-type file or as a device-type file. However, if the device-type file name CLOCK is specified, an abort error will occur.
2) When option -E is specified and the output file name is omitted, the error list file name will be 'input file name.ERA'.
3) When option -E is specified and the drive name is omitted, the error list file will be output to the current directory.
4) If both options -E and -NE are specified at the same time, the option specified last takes precedence.

[Example]

Example 1. Create an error list file (SAMPLE.ERA).

```
C>ra78k3 78k3main.asm -esample.era

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
78K3MAIN.ASM(20) : F201 Syntax error
78K3MAIN.ASM(23) : F201 Syntax error
Pass2 Start
78K3MAIN.ASM(12) : F404 Public symbol is undefined 'dSTART'
78K3MAIN.ASM(20) : F201 Syntax error
78K3MAIN.ASM(23) : F201 Syntax error

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

This references the error list file (SAMPLE.ERA).

```
Pass1 Start
78K3MAIN.ASM(20) : F201 Syntax error
78K3MAIN.ASM(23) : F201 Syntax error
Pass2 Start
78K3MAIN.ASM(12) : F404 Public symbol is undefined 'dSTART'
78K3MAIN.ASM(20) : F201 Syntax error
78K3MAIN.ASM(23) : F201 Syntax error
```

(12) Specify parameter file (-F)

Description format : -F [file name]
Default value : This option and the input file name can only be entered on the startup line.

[Function]

Option -F inputs assembler options and the input file name from a specified file.

[Application]

1) Specify option -F when the data required to start up the assembler will not fit on the command line.
2) Specify option -F to repeatedly specify the same options each time assembly is performed and to save those options to a parameter file.

[Description]

1) Only a disk-type file name can be specified as 'file name'. If a device-type file name is specified, an abort error will occur.
2) If the file name is omitted, an abort error will occur.
3) Nesting of parameter files is not permitted. If option -F is specified within a parameter file, an abort error will occur.
4) The number of characters that can be described within a parameter file is unlimited.
5) Separate options or file names with a blank space, a tab or [↵].
6) Parameters and input file names within a parameter file will be expanded at the position specified for the parameter file on the command line.
7) The expanded options specified last will take precedence.
8) All characters entered after ';' or '#' and before [↵] or 'EOF' will be interpreted as comments.
9) If option -F is specified two or more times, an abort error will occur.

[Example]

Example Perform assembly using a parameter file.

• Contents of the parameter file (78K3MAIN.PRA)

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

• Enter the following on the command line.

```
C>ra78k3 -f78k3main.pra

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(13) Specify path for temporary file creation (-T)

Description format : -T Path

Default value : Creates a temporary file in the path specified by the environmental variable TMP. When no path is specified, the temporary file is created in a current path.

[Function]

Option -T specifies a path in which a temporary file is created.

[Application]

Use option -T to specify the location for creation of a temporary file.

[Description]

1) Only a path can be specified as a path name.
2) The path name cannot be omitted.
3) Even if a previously created temporary file exists, if the file is not protected it will be overwritten.
4) As long as the required memory size is available, the temporary file will be expanded in memory. If not enough memory is available, the contents of the temporary file will be written to a disk.
   Such temporary files may be accessed later through the saved disk file.
5) Temporary files are deleted when assembly is finished. They are also deleted when assembly is aborted by pressing (CTRL-C).
6) The path in which the temporary file is to be created is determined according to the following sequence.
   a. The path specified by option -T
   b. The path specified by environmental variable TMP (when option -T is omitted)
   c. The current path (when TMP is not set)

   When a. or b. is specified, if the temporary file cannot be created in the specified path an abort error occurs.

[Example]

Example Specify output of a temporary file to directory TMP.

```
C>ra78k3 78k3main.asm -ttmp

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(14) Specify help (--)

Displays a help message on the display.

Description format : --
Default value : No display

[Function]

Option -- displays a help message.

[Application]

The help message is a list of explanations of the assemble options. Refer to these when executing the assembler.

[Description]

1) When option -- is specified, all other options are unavailable.
2) To read the next part of the help message, press the return key.
   To quit the help display, press any key other than the return key and then press the return key.

[Example]

When option -- is specified, a help message is output on the display.

C>ra78K3 --

```
78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

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/-nkx      :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.
-lt[n]        :Expand TAB character for print file(n=1 to 8)/ Not expand(n=0).
-lhstring     :Print list header with the specified string.
-ca/-nca      :Convert alphabet to capital for symbol / Not.
-g/-ng        :Output debug information to object file / Not.
-j/-nj        :Create object file if fatal error occurred / Not.
-idirectory[,directory..]  :Set include search path.
-tdirectory   :Set temporary directory.
-ffile        :Input option or source module file name from specified file.
-s/-ns        :Expand symbol length up to 31 /or symbol length is 8.
-ga/-nga      :Output assembler source debug information to object file / Not.
--            :Show this message.
DEFAULT ASSIGNMENT:
  -o -ne -p -ka -nks -nkx -lw132 -ll66 -nlf -lt8 -nca -g -nj -s -ga
```

[MEMO]

CHAPTER 5 LINKER

The linker inputs a number of object module files output by the 78K/III assembler, determines a location address and outputs them as a single load module file.

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

If a link error occurs, an error message is output to an error list file to clarify the cause of the error. When an error occurs, the load module file will not be output.

5.1 Files Output by the Linker

The following table shows the input/output files of the linker.

Table 5-1. Files Output by the Linker

Type	File Name	Explanation	Default File Type
Input files	Object module files	• These are binary files which contain relocation and symbol data for machine language data and the location addresses of machine language data. • These files are output by the assembler.	.REL
	Library files	• These are files in which two or more object module files are included. • These files are output by the librarian.	.LIB
	Directive files	• These are files which contain link commands used during linking. • These files are created by the user.	.DR
	Parameter files	• These files contain the parameters for program execution. • These files are created by the user.	.PLK
Output files	Load module files	• These are binary image files which contain all data created as a result of linking. These files are input to the object converter.	.LNK
	Link list files	• These are list files which display the result of linking.	.MAP
	Error list files	• These files contain error data generated during linking.	.ELK
Input/ output files	Temporary files	• These files are automatically generated by the linker for use in linking. They are deleted when assembly is complete.	LKxxxxxx. $$n (n=1 to 3)

5.2 Functions of the Linker

The functions of the linker are as follows.

(1) Joining of input segments

The linker determines and controls the location address of each segment.
The linker identifies identical segments and joins them into a single segment, even if they are in separate object module files.

(2) Determination of input modules

When a library file is specified for input, the module to which an input object module file refers is retrieved from the library and handled as an input module.

(3) Determination of location addresses for input segments

The linker determines location addresses for each segment of an input module. If location attributes for a segment are specified in the source module file, the segment is located according to those attributes. The linker can also specify location attributes in the link directive file of the linker.

(4) Correction of object codes

When location addresses are buried in object codes, the linker corrects the object code according to the location address determined in (3) above.

5.3 Memory Spaces and Memory Areas

A memory space is a space provided for defining memory areas. A memory area is an area defined in memory for the allocation of segments.

Memory space: 64 KB each
Memory area: Each memory space is divided into several memory areas.
The memory area declares the memory addresses for the installed memory.
Segment allocation groups (external ROM, etc.)

Memory area name	Default address	Segments allocated by default
ROM	Internal ROM: Until beginning of RAM if no ROM is installed	CSEG
RAM	Internal RAM	DSEG, BSEG

Remark Use a directive file to change the default address of a memory area or to specify the location of each segment described in a program.

5.4 Link Directives

A link directive (hereinafter referred to as a "directive") is a group of instructions used to perform various directions during linking, such as file input, usable memory area and allocation of segments.

The role of the directive file is to:

(1) Declare addresses in the installed memory.

(2) Divide memory into two or more areas.

Example CALLT area
Internal ROM
External ROM
SADDR
Internal RAM other than SADDR

(3) Segment allocation is specified by the linker.

The following items are specified for each segment.

- Absolute address
- Specification of memory address only

Use an editor to create a directive file (a file which describes directives). When the linker is started up, specify option -D to read the created file.
The linker reads the directives from the file and interprets them to perform linking.
Two types of directives can be used as follows.

Table 5-2. Types of Directives

No.	Directive Type	Explanation
1	Memory directive	• Declares an address in installed memory • Divides memory into two or more areas and specifies a memory area
2	Segment location directive	• Specifies location of a segment

5.4.1 Directive files

The formats for describing directives in a directive file are as follows.
A number of directives can be described in a single directive file.

1) Memory directives
   MEMORY memory area name : (start address value, size) [/memory space name]
2) Segment allocation directives
   MERGE segment name : [ATΔ(Δstart addressΔ)]
   [=memory area name specification] [/memory space name]

(1) Reserved words

The following words are reserved words in a directive file.

MEMORY, MERGE, AT, SEQUENT, COMPLETE

Reserved words cannot be used in a directive file for other meanings (segment name, memory area name, etc.).
Reserved words can be described in uppercase or lowercase characters, but not in a mixture of the two.

Example MEMORY
memory
Memory ; Cannot be used

(2) Symbols

Uppercase and lowercase characters are distinguished when describing segment names, memory area names and memory space names.
Even if a segment name is described in lowercase characters in a source module file, it is possible to handle all characters as uppercase characters by specifying option -CA (do not distinguish uppercase and lowercase characters) during assembly. In this case, use uppercase characters to specify segment names in the directive file.

(3) Numerical values

To describe a numerical constant for each item in a directive, describe the constant in decimal or hexadecimal form.
The description method is the same as for source programs; add "H" at the end for hexadecimals. If A-F appear at the beginning, place "0" first.

Example 23H, 0FC80H

(4) Comments

When a ';' or '#' is described in a directive file, all characters entered from that point to carriage return (LF) are handled as a comment. If the directive file ends before a carriage return, everything before the end of the file is handled as a comment.

Example The underlined portion is a comment.

```
:DIRECTIVE FILE FOR 78312
MEMORY MEM1: (1000H, 1000H) #SECOND MEMORY AREA
```

5.4.2 Memory directives

A memory directive is a directive which defines a memory area (name of an address in the installed memory). The name of a defined memory area (the memory area name) is used to reference a segment location directive. Up to 100 memory areas can be defined, including the default memory area.

[Syntax]

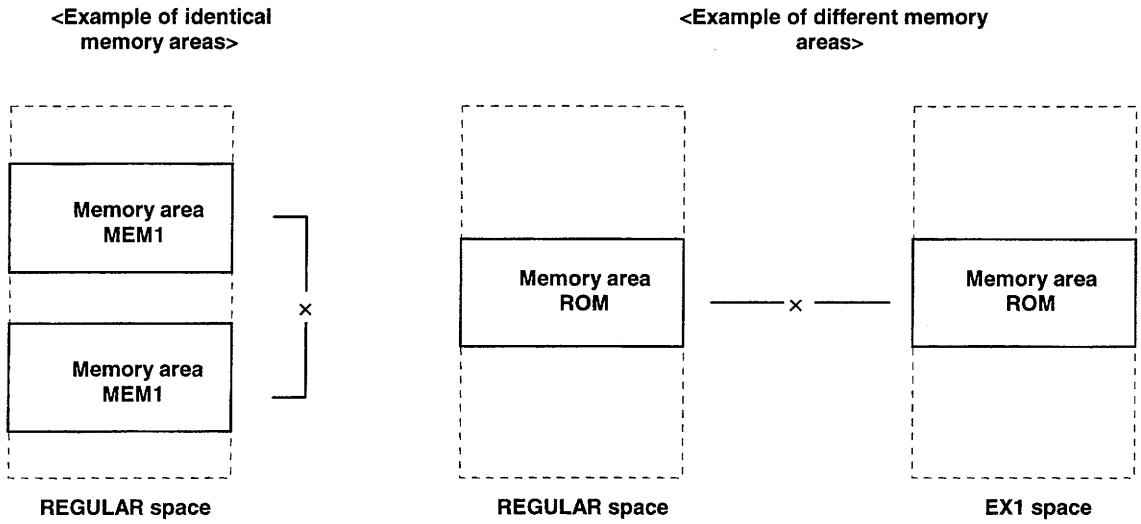
MEMORY∆ memory area name∇: ∇(∇start address value∇, ∇size∇) [/∇memory space name]

(1) Memory area names

Specify a name for the defined memory area. Conditions for specification of memory area names are as follows.

1) The characters which can be used to describe a memory area name are A-Z, a-z, 0-9, _, ?, and @. However, a memory area name cannot begin with 0-9.
2) Uppercase and lowercase characters are interpreted as separate characters.
3) Uppercase and lowercase characters can be mixed together.
4) Maximum length of a memory area name is 31 characters. If 32 or more characters are described, an error results.
5) Each memory area name must exist in only 1 location in the entire memory space. The same memory area name cannot be used for a different memory area, even if they are in different memory spaces.

Figure 5-1. Memory Area Names



(2) Start addresses

Specify the start address of the memory area to be defined.

- Describe a numerical value from 0H to FFFFH.

(3) Size

Specify the size of the memory area to be defined. Specification conditions are as follows.

1) Describe a numerical value of 1 or higher.
2) If the size specification is changed to the default memory area size defined by the linker, limitations on the definable range apply.

For the default memory area size defined for each device and the redefinable range for each device, see the "Considerations on Use" for each device file.

1) With μPD78312, 78312A, 78322, 78328, and 78334 Subseries
   Refer to D.4, "Default Link Directive Information".
2) With μPD78352A, 78356, 78366, 78336A, and 78372 Subseries
   Refer to the separately available document on the device file (Notes on Using DF783xx Device File).

(4) Memory space names

The following 16 memory space names are displayed for 16 memory spaces of 64 KB each.

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

Use memory space names to assign a memory area to a particular memory space. The following conditions on specification of memory space names apply.

1) Memory space names must be entirely in uppercase characters.
2) When a memory space name is omitted, REGULAR is assumed to be specified.
3) If the memory space name is omitted after '/' is described, an error occurs.

[Function]

1) Define a specified memory space for a memory area specified with a memory area name.
2) 1 memory area can be defined with 1 memory directive.
3) A memory directive can be described more than once. However, multiple definitions in the specified order will result in an error.
4) The default memory area is effective as long as the same memory area is not redefined in a memory directive. If the description of a memory directive is omitted, only the default memory area carried by the linker for each device will be specified.

[Example]

1) Define the addresses 0H to 1FFH in the default memory space (REGULAR) as ROMA.

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

2) Define an area as memory area RAMA.

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

5.4.3 Segment location directives

A segment location directive is a directive which locates a specified segment in a specified area of memory or a specific address.

[Syntax]

MERGE∆segment name∇:∇ [AT∇(∇start-address∇)] [∇=∇memory-area-name] [∇/∇memory-space-name]

(1) Segment name

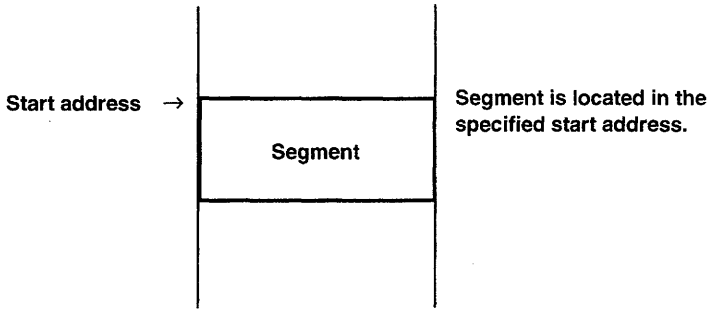
The segment name is the name of a segment included in an object module file input to the linker.

1) Only an input segment can be specified with a segment name.
2) If option -CA is not specified during assembly, the segment name must be specified in the same way as in the source.
3) If option -CA is specified during assembly, the segment name must be specified in uppercase characters.

(2) Start address

The start address allocates a segment to the area specified by "start address."

1) The reserved word AT must be described entirely in either uppercase or lowercase characters. It cannot be described in a mixture of uppercase and lowercase characters.
2) The start address describes a numerical constant.



Cautions 1. When a segment is located in the specified start address, if it exceeds the memory area range for the memory area in which it is located, an error will result.

2. A link directive cannot be used to specify a start address for a segment whose location address is specified by the AT instruction of a segment directive or by an ORG directive.

(3) Memory space names

A memory space name specifies the memory area to which a segment is allocated.

1) Any of the following 16 names can be specified as a memory area name.
   REGULAR, EX1, EX2, EX3, EX4, EX5, EX6, EX7, EX8, EX9, EX10, EX11, EX12, EX13, EX14, EX15
2) Memory space names must be entirely in uppercase characters.
3) When a memory space name is omitted, REGULAR is assumed to be specified.

Segment location destinations are determined as follows.

Table 5-3. Segment Location According to Combination of Memory Area Name Specification and Memory Space Name

Memory area name	Memory space name	Segment location destination
N/A	N/A	Default memory area in the REGULAR space
N/A	Available	A selected memory area in the specified memory space
Available	N/A	Specified memory area in the REGULAR space
Available	Available	Specified memory area in the specified memory space

This table focuses on defining the memory area to which the segment is located. When the actual location address is determined, if [AT (start address)] is specified, the segment is allocated to a location beginning at that address.
For example, if the memory space name 'EX1' is specified for a segment with the relocation characteristic 'CSEG.FIXED', the segment will be located to fit within 800H to FFFH.

[Notes]

1) The location address of an input segment for which no segment location directive is specified will be determined according to the relocation characteristics specified by a segment directive during assembly.
2) If no segment exists for which a segment name has been specified, an error will occur.
3) If more than one segment location directive is specified for the same segment, an error will occur.

[Example]

Allocate an address for a segment SEG1, which has the segment type and relocation characteristic 'CSEG UNIT'. In this example the declared memory area is as follows.

(1) When input segment SEG1 is allocated to 2000H in memory area ROM.

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

(2) When input segment SEG1 is allocated to memory area MEM1.

```
MERGE SEG1: =MEM1
```

(3) When input segment SEG1 is allocated to 2000H in memory area MEM1.

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

5.5 Linker Startup

5.5.1 Linker startup

The following 2 methods can be used to start up the linker.

(1) Startup from the command line

```
X>[path-name] lk78k3 [Δoption]
 |    |        |      |
(1)  (2)      (3)    (4)
                ··· Δobject-module-file-name-[Δoption]···[Δ]
                                 |                    |
                                (5)                  (4)
```

(1) Current drive name
(2) Current directory name
(3) Linker command file name
(4) This contains detailed directions for the action of the linker.
If more than one linker option is specified, separate the options with a space.
(5) This contains detailed directions for the action of the linker.
A maximum of 128 items can be input in an input module.

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

(2) Startup from a parameter file

Use the parameter file when the data required to start up the linker will not fit on the command line, or when repeating the same linker option for two or more assembly operations.
To start up the linker from a parameter file, specify the parameter file specification option (-F) on the command line.
Start up the linker from a parameter file as follows.

X>LK78k3 [Δobject-module-file] Δ-f parameter-file-name
(1) (2)

(1) Parameter file specification option
(2) A file which includes the data required to start up the linker

Remark An editor is used to create the parameter file.

The rules for describing the contents of a parameter file are as follows.

[[[Δ] option [Δoption] ··· [Δ]Δ]] ···

1) If the object module file name is omitted from the command line, specify the object module file name in the parameter file.
2) The object module file name can also be described after the option.
3) Describe in the parameter file all linker options and output file names that should be specified in the command line.

Example Create the parameter file (78K3.PLK) using an editor.
Contents of the parameter file 78K3.PLK:

```
;parameterfile
78k3main.rel 78k3sub.rel -o78k3.lnk -p78k3.map -e
-ta:\tmp -q
```

Use parameter file 78K3.PLK to start up the linker.

C>lk78k3 -f78k3.plk

5.5.2 Execution start and end messages

(1) Execution start message

When the linker is started up, an execution startup message appears on the display.

```
78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx
```

(2) Execution end message

If it detects no link errors resulting from the link, the linker outputs the following message to the display and returns control to the operating system.

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

If it detects a link error resulting from the link, the linker outputs the error number to the display and returns control to the operating system.

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

If the linker detects a fatal error during linking which makes it unable to continue link processing, the linker outputs a message to the display, cancels linking and returns control to the operating system.

Example 1. A nonexistent object module file is specified.

```
C>lk78k3 samp1.rel samp2.rel

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

A006 File not found 'SAMP1.REL'
A006 File not found 'SAMP2.REL'
Program Aborted.
```

In the above example, a nonexistent object module file is specified. An error results and the linker aborts the link.

Example 2. A nonexistent linker option is specified.

```
C>lk78k3 78k3main.rel 78k3sub.rel -z

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

A018 Option is not recognized '-z'
Program Aborted.
```

In the above example, a nonexistent linker option is specified. An error results and the linker aborts the link.

When an error message is displayed and link is aborted, look for the cause in chapter 11, "Error Messages" and take action accordingly.

5.6 Linker Options

5.6.1 Types of linker options

The linker options are detailed instructions for the operation of the linker. Linker options are classified into 15 types.

Table 5-4. Linker Options (1/2)

Number	Classification	Option	Explanation
1	Specify load module file output	-O	Specifies the output of a load module file.
		-NO	
2	Specify forced load module file output	-J	Forces output of a load module file.
		-NJ	
3	Specify debug data output	-G	Outputs debugging data to a load module file.
		-NG	
4	Specify stack decision symbol generation	-S	Automatically generates public symbols for stack decision.
		-NS	
5	Specify directive file	-D	Inputs the specified file as a directive file.
6	Specify link list file output	-P	Specifies output of a link list file.
		-NP	
7	Specify link list file data	-KM	Outputs a map list into a link list file.
		-NKM	
		-KD	Outputs a link directive file into a link list file.
		-NKD	
		-KP	Outputs a public symbol list into a link list file.
		-NKP	
		-KL	Outputs a local symbol list into a link list file.
		-NKL	
8	Specify link list file format	-LL	Changes the number of lines that can be printed in 1 page in a link list file.
		-LF	Inserts a line feed code at the end of a list file.
		-NLF	

Table 5-4. Linker Options (2/2)

Number	Classification	Option	Explanation
9	Specify error list file output	-E	Outputs an error list file.
		-NE	
10	Specify library file	-B	Inputs the specified file as a library file.
11	Specify library file read path	-I	Reads a library file from a specified path.
12	Specify parameter file	-F	Inputs file names and options from a specified file.
13	Specify path for temporary file creation	-T	Creates a temporary file in a specified path.
14	Specify warning message output	-W	Specifies whether or not to output a warning message to the console.
15	Specify help	--	Displays a help message on the display.

This table is presented as a brief introduction to the linker options. When actually using the linker options, see Appendix E.2, "List of Linker Options."

5.6.2 Order of precedence of linker options

The following table indicates which linker option takes precedence when two linker options are specified at the same time.

Table5-5. Order of Precedence of Linker Options

	-NO	-NG	-NP	-NKM	-NKP	-NKL	--
-J	×						×
-G	×						×
-P				Δ	Δ	Δ	×
-KM			×				×
-KD			×	×			×
-KP		×	×				×
-KL		×	×				×
-LL			×				×
-LF			×				×

← Horizontal axis

↑ Vertical axis

[Items marked with an X]

When the option in the horizontal axis is specified, the option shown in the vertical axis option is unavailable.

Example C>lk78k3 78k3main.rel 78k3sub.rel -np -km

The option -KF is unavailable.

[Items marked with a Δ]

When all three of the options in the horizontal axis are specified, the option shown in the vertical axis option is unavailable.

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

The options -NKM, -NKP and -NKL are all specified at the same time, so option -P is unavailable.

When an option and its 'N' counterpart are specified at the same time (for example, both -O and -NO), only the last specified of the 2 options is available.

Example C>lk78k3 78k3main.rel 78k3sub.rel -o -no

The option -NO is specified after -O, so option -O is unavailable and -NO is available.

Options not described in Table 5-5 have no particular effect on other options. However, when the help option '--' is specified, all other options become unavailable.

5.6.3 Explanation of linker options

This section contains detailed explanations of each linker option.

(1) Specify load module file output (-O/-NO)

Description format : -O [output-file-name]
: -NO
Default value : -O input file name.LNK

[Function]

1) Option -O specifies the output of a LOAD module file. It also specifies the location to which it is output and the file name.
2) Option -NO specifies that no LOAD module file is output.

[Application]

1) Use option -O to specify the location to which a load module file is output or to change its file name.
2) Specify option -NO when performing a link only to output a link list file. This will shorten link time.

[Description]

1) The disk type file name and device type file name, NUL and AUX can be specified as output file names.
2) Even if option -O is specified, if a fatal error occurs the load module file cannot be output.
3) If 'output file name' is omitted when option -O is specified, the load module file 'input file name.LNK' will be output to the current directory.
4) If only the path name is specified in 'output file name', 'input file name.LNK' will be output to the specified path.
5) If both options -O and -NO are specified at the same time, the option specified last takes precedence.

[Example]

Example 1. Output a load module file 78K3.LNK.

```
C>lk78k3 78k3main.rel 78k3sub.rel –078k3.lnk

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Example 2. Specify both options -NO and -Oo

```
C>lk78k3 78k3main.rel 78k3sub.rel –no –o

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Option -NO is invalid,and option -O is valid.

(2) Specify forced load module file output (-J/-NJ)

Description format : -J
: -NJ
Default value : -NJ

[Function]

1) Option -J specifies that the load module will be output even if a fatal error occurs.
2) Option -NJ makes option -J unavailable.

[Application]

Normally, when a fatal error occurs, the load module file cannot be output. When you wish to execute the program with a notice that a fatal error has occurred, specify option -J to output the load module file.

[Description]

1) When option -J is specified, the load module will be output even if a fatal error occurs.
2) If both options -J and -NJ are specified at the same time, the option specified last takes precedence.

[Example]

Example Specify output of a load module file even if a fatal error occurs.

```
C>lk78k3 78k3main.rel 78k3sub.rel 78k3sub.rel-j

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(3) Specify debug data output (-G/-NG)

Description format : -G
: -NG
Default value : -G

[Function]

1) Option -G specifies that debugging data (local symbol data) is to be added to a load module file.
2) Option -NG makes option -G unavailable.

[Application]

Be sure to use option -G when performing symbolic debugging with a source debugger.

[Description]

1) If option -NO is specified, option -G is unavailable.
2) If option -G is omitted, debug data cannot be added.
3) If both options -G and -NG are specified at the same time, the option specified last takes precedence.
4) When option -NG is specified, the public symbol list and local symbol list cannot be output regardless of specification of -KP or -KL.

[Example]

Specify addition of debug data to a load module file.

```
C>lk78k3 78k3main.rel 78k3sub.rel –g

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(4) Specify generation of stack decision symbols (-S/-NS)

Description format : -S [Area name]
: -NS
Default value : -NS

[Function]

1) Option -S generates the stack decision public symbols '_@STBEG' and '_@STEND'.
2) Option -NS makes option -S unavailable.

[Application]

Specify option -S to reserve a stack area.

[Description]

1) An 'area name' is a name in which an area memory name defined by the user or an area memory name defined by default is specified.
2) 'Area names' distinguish between uppercase and lowercase characters.
3) The linker searches the memory area specified by option -S for the largest address in which no segment is located. The linker then generates public symbol '_@STEND', which holds the lead address of the largest address area as its value, and public symbol '_@STBEG', which holds the last address +1 as its value.
   These symbols are handled as publicly declared NUMBER attribute symbols, and are registered at the end of the linker's symbol table. When these symbols are output to a link list file, the module name column is left blank.
4) If the largest open area is 10 bytes or smaller, a warning message is output.
5) If no free area exists, a warning message is output and both '_@STEND' and '_@STBEG' hold the last address +1 as their values.
6) If 'area name' is omitted, 'RAM' is specified.
7) If both options -S and -NS are specified at the same time, the option specified last takes precedence.

[Example]

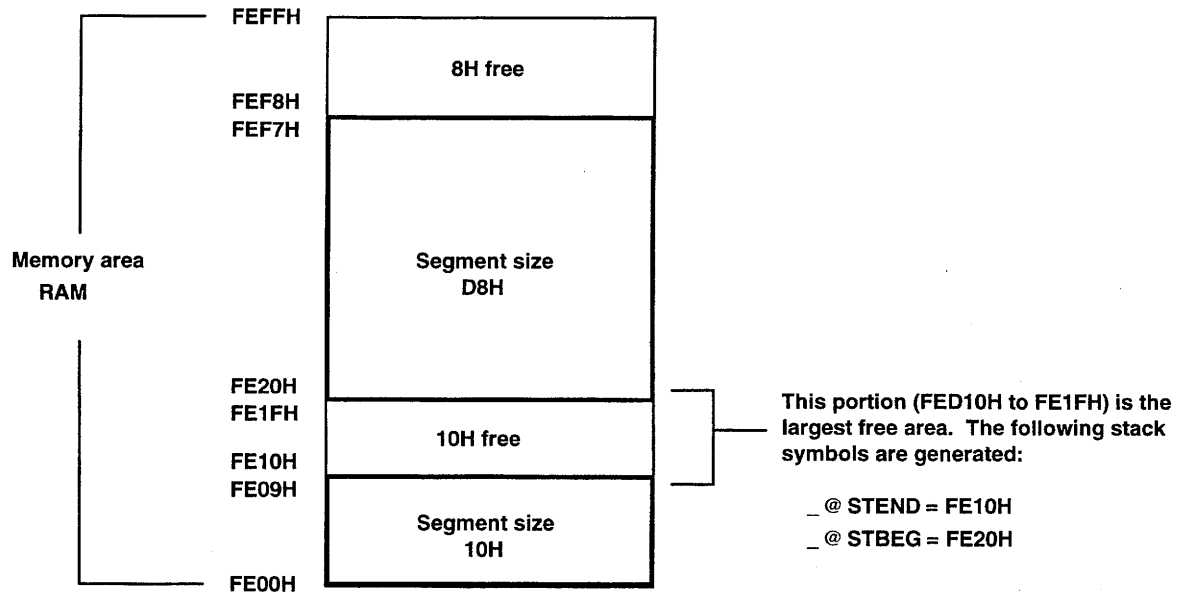
Reserve the stack area in memory area RAM (however, the linker will assume that a segment of size 10H in RAM and a segment of size 08H located in the saddr area are input).

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

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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



(5) Specify directive file (-D)

Description format : -D file name
Default value : None

[Function]

Option -D specifies that a specified file is to be input as a directive file.

[Application]

When you wish to define a new memory area, redefine the default memory area, or locate a segment to a specific address or memory area, you will need to create a directive file. Specify option -D to input this directive file to the linker.

[Description]

1) Only disk-type file names can be specified as a 'file name'. If a device-type file name is specified, an abort error will result.
2) If the file name is omitted, an abort error will result.
3) Nesting of directive files is not permitted.
4) The number of characters that can be described in a directive file is unlimited.
5) If option -D is specified more than once, or if more than one file name is specified, an abort error will occur.
6) For a detailed explanation of directive files, see 5.4, "Link Directives."

[Example]

Example 1. Redefine the default memory area ROM/RAM.

- Contents of the directive file 78K3.DR:

```
MEMORY ROM: (0000H, 4000H)
MEMORY RAM: (0D000H, 2F00H)
```

- Perform link using 78K3.DR.

C>lk78k3 78k3main.rel 78k3sub.rel -d 78k3.dr

```
78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(6) Specify link list file output (-P/-NP)

Description format : -P [output-file-name]
: -NP
Default value : -P input file name.MAP

[Function]

1) Option -P specifies output of a link list file. It also specifies the destination and file name of the output file.
2) Option -NP makes option -P unavailable.

[Application]

1) Specify option -P to change the output destination or output file name of a link list file.
2) Specify option -NP when performing link only to output a load module file. This will shorten link time.

[Description]

1) A file name can be specified as a disk-type file name or as a device-type file name. However, only CON, PRN, NUL and AUX can be specified as device-type file names. If CLOCK is specified, an abort error will occur.
2) If the 'output file name' is omitted when option -P is specified, the link list file name in the current directory becomes 'input file name.MAP'.
3) If only the 'output file name' is specified, 'input file name.MAP' is output to the specified path.
4) If both options -P and -NP are specified at the same time, the option specified last takes precedence.

[Example]

Example 1. Create a link list file (78k3.MAP).

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

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

Example 2. Output the link list file to printer.

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

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(7) Specify link list file data (-KM/-NKM, -KD/-NKD, -KP/-NKP, -KL/-NKL)

(a) -KM/-NKM

Description format : -KM
: -NKM
Default value : -KM

[Function]

1) Option -KM outputs a map list into a link list file.
2) Option -NKM makes option -KM unavailable.

[Application]

Specify option -KM to output a map list to a link list file.

[Description]

1) If both options -KM and -NKM are specified at the same time, the option specified last takes precedence.
2) If option -NKM is specified, the link directive file cannot be output to a link list file even if option -KD is specified.
3) If options -NKM, -NKP and -NKL are all specified, the link list file cannot be output even if option -P is specified.

[Example]

Output a map list into link list file 78K3.MAP.

```
C>lk78k3 78k3main.rel 78k3sub.rel -p78k3.map -km

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

• This references 78K3.MAP.

```
78K/III Series 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)
   3EH byte(s) real data
    23 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=0000H    SIZE=FE00H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
         CODE                         0000H     0002H   CSEG AT
                  CODE     SAMPM      0000H     0002H
         ?CSEG                        0002H     003CH   CSEG
                  ?CSEG    SAMPM      0002H     0020H
                  ?CSEG    SAMPS      0022H     001CH
* gap *                               003EH     FDC2H

  MEMORY=RAM
  BASE ADDRESS=FE00H    SIZE=0200H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
* gap *                               FE00H     0020H
         DATA                         FE20H     0003H   DSEG AT
                  DATA     SAMPM      FE20H     0003H
* gap *                               FE23H     00DDH
* gap (Not Free Area) *               FF00H     0100H

 Target chip : uPDxxxxx
 Device file : Vx.xx
```

Map list

(b) -KD/-NKD

Description format : -KD
: -NKD
Default value : -KD

[Function]

1) Option -KD outputs a link directive file into a link list file.
2) Option -NKD makes option -KD unavailable.

[Application]

Specify option -KD to output a link directive file into a link list file.

[Description]

1) If both options -KD and -NKD are specified at the same time, the option specified last takes precedence.
2) If option -NKM is specified, a link directive file cannot be output into a link list file even if option -KD is specified.
3) If options -NKM, -NKP and -NKL are all specified, a link list file cannot be output even if option -P is specified.

[Example]

Output a link directive file into a link list file (78K3.MAP).

```
C>lk78k3 78k3main.rel 78k3sub.rel -d78k3.dr -p78k3.map -kd

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

• This references 78K3MAIN.PRN.

```
78K/III Series 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 file name
Directive: MEMORY ROM:(0000H,4000H)        ┐    ← Contents of directive file
           MEMORY RAM:(0D000H,2F00H)       ┘

*** Link information ***

     3 output segment(s)
   3EH byte(s) real data
    23 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=0000H    SIZE=4000H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
         CODE                         0000H     0002H   CSEG AT
                  CODE     SAMPM      0000H     0002H
         ?CSEG                        0002H     003CH   CSEG
                  ?CSEG    SAMPM      0002H     0020H
                  ?CSEG    SAMPS      0022H     001CH
* gap *                               003EH     3FC2H

  MEMORY=RAM
  BASE ADDRESS=D000H    SIZE=2F00H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
* gap *                               0000H     2E20H
         DATA                         FE20H     0003H   DSEG AT
                  DATA     SAMPM      FE20H     0003H
* gap *                               FE23H     000DH


 Target chip : uPDxxxxx
 Device file : Vx.xx
```

(c) -KP/-NKP

Description format : -KP
: -NKP
Default value : -NKP

[Function]

1) Option -KP outputs a public symbol list into a link list file.
2) Option -NKP makes option -KP unavailable.

[Application]

Specify option -KP to output a public symbol list into a link list file.

[Description]

1) If both options -KP and -NKP are specified at the same time, the option specified last takes precedence.
2) If options -NKM, -NKP and -NKL are all specified, the link list file cannot be output even if option -P is specified.
3) If options -NG is specified, the public symbol list cannot be output even if option -KP is specified.

[Example]

Example Output a public symbol list into a link list file (78K3.MAP).

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

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

• This references 78K3.MAP.

```
78K/III Series 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)
   3EH byte(s) real data
    23 symbol(s) defined


*** Memory map ***


  SPACE=REGULAR

MEMORY=ROM
  BASE ADDRESS=0000H    SIZE=FE00H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
         CODE                         0000H     0002H   CSEG AT
                  CODE     SAMPM      0000H     0002H
         ?CSEG                        0002H     003CH   CSEG
                  ?CSEG    SAMPM      0002H     0020H
                  ?CSEG    SAMPS      0022H     001CH
* gap *                               003EH     FDC2H

  MEMORY=RAM
  BASE ADDRESS=FE000H    SIZE=0200H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
* gap *                               FE00H     0020H
         DATA                         FE20H     0003H   DSEG AT
                  DATA     SAMPM      FE20H     0003H
* gap *                               FE23H     00DDH
* gap (Not Free Area) *               FE00H     0100H

------------------------------------------------------------------------------
78K/III Series 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


 Target chip : uPDxxxxx
 Device file : Vx.xx
```

Public symbol list

(d) -KL/-NKL

Description format : -KL
: -NKL
Default value : -NKL

[Function]

1) Option -KL outputs a local symbol list into a link list file.
2) Option -NKL makes option -KL unavailable.

[Application]

Specify option -KL to output a local symbol list into a link list file.

[Description]

1) If both options -KL and -NKL are specified at the same time, the option specified last takes precedence.
2) If options -NKM, -NKP and -NKL are all specified, the link list file cannot be output even if option -P is specified.
3) If options -NG is specified, the local symbol list cannot be output even if option -KL is specified.

[Example]

Output a local symbol list into a link list file (78K3.MAP).

```
C>lk78k3 78k3main.rel 78k3sub.rel -g -p78k3.map -kl

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

• This references 78K3.MAP.

```
78K/III Series 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)
   3EH byte(s) real data
    23 symbol(s) defined


*** Memory map ***


  SPACE=REGULAR

MEMORY=ROM
  BASE ADDRESS=0000H    SIZE=FE00H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
         CODE                         0000H     0002H   CSEG AT
                  CODE     SAMPM      0000H     0002H
         ?CSEG                        0002H     003CH   CSEG
                  ?CSEG    SAMPM      0002H     0020H
                  ?CSEG    SAMPS      0022H     001CH
* gap *                               003EH     FDC2H

  MEMORY=RAM
  BASE ADDRESS=FE000H    SIZE=0200H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
* gap *                               FE00H     0020H
         DATA                         FE20H     0003H   DSEG AT
                  DATA     SAMPM      FE20H     0003H
* gap *                               FE23H     00DDH
* gap (Not Free Area) *               FE00H     0100H

--------------------------------------------------------------------------------

78K/III Series 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


 Target chip : uPDxxxxx
 Device file : Vx.xx
```

Local symbol list

(8) Specify link list format (-LL, -LF/-NLF)

(a) -LL

Description format : -LL [number of lines]
Default value : -LL66 (No page breaks in the case of display output)

[Function]

Option -LL changes the number of lines that can be printed in 1 page in a link list file.

[Application]

Specify option -LL to change the number of lines that can be printed in 1 page in a link list file.

[Description]

1) The range of number of lines that can be specified with option -LL is shown below.

   20 ≤ number of lines printed on 1 page ≤ 32767

   If a numerical value outside this range, or something other than a numerical value, is specified, an abort error occurs.
2) If the number of lines is omitted, 66 will be specified.
3) If the number of lines specified is 0, no page breaks will be made.
4) If option -NP is specified, option -LL is unavailable.

[Example]

Specify 20 as the number of lines per page in a link list file.

```
C>lk78k3 78k3main.rel 78k3sub.rel -p78k3.map -ll20

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

• This references 78K3MAP.

```
78K/III Series 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)
   3EH byte(s) real data
--------------------------------------------------------------------------------


78K/III Series Linker Vx.xx                    Date:xx xxx xxxx Page:   2

    23 symbol(s) defined


*** Memory map ***


  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=0000H    SIZE=FE00H
         OUTPUT   INPUT    INPUT      BASE          SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
--------------------------------------------------------------------------------


78K/III Series Linker Vx.xx          Date:xx xxx xxxx Page:   3

         CODE                          0000H     0002H   CSEG AT
                  CODE     SAMPM       0000H     0002H
         ?CSEG                         0002H     003CH   CSEG
                  ?CSEG    SAMPM       0002H     0020H
                  ?CSEG    SAMPS       0022H     001CH
* gap *                                003EH     FDC2H

  MEMORY=ROM
  BASE ADDRESS=FE00H    SIZE=0200H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
* gap *                                FE00H     0020H
--------------------------------------------------------------------------------


78K/III Series Linker Vx.xx          Date:xx xxx xxxx Page:   4

         DATA                          FE20H     0003H   DSEG AT
                  DATA     SAMPM       FE20H     0003H
* gap *                                FE23H     00DDH
* gap (Not Free Area) *                FE00H     0100H

  Target chip : uPDxxxxx
  Device file : Vx.xx
```

(b) -LF/-NLF

Description format : -LF
: -NLF
Default value : -NLF

[Function]

1) Option -LF inserts a form feed (FF) code at the end of a link list file.
2) The option -NLF makes the option -LF unavailable.

[Description]

If you wish to add a page break after the contents of a link list file are printed, specify option -LF to add a form feed code.

[Explanation]

1) If option -NP is specified, option -LF is unavailable.
2) If both options -LF and -NLF are specified at the same time, the option specified last takes precedence.

[Example]

Add a form feed code at the end of a link list file.

```
C>lk78k3 78k3main.rel 78k3sub.rel -p78k3.map -lf

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(9) Specify error list file output (-E/-NE)

Description format : -E [file name]
: -NE
Default value : -NE

[Function]

1) Specify option -E to specify the output destination and file name of an error list file.
2) Option -NE makes option -E unavailable.

[Application]

Specify option -E to change the output destination and output file name of the error list file.

[Explanation]

1) The file name of the error list file can be specified as a disk-type file name or as a device-type file name. However, if the device-type file name CLOCK is specified, an abort error will occur.
2) When option -E is specified and the output file name is omitted, the error list file name will be 'input file name.ELK'.
3) When option -E is specified and the drive name is omitted, the error list file will be output to the current drive.
4) If both options -E and -NE are specified at the same time, the option specified last takes precedence.

[Example of use]

Example Create an error list file (78K3.ELK).

```
C>lk78k3 78k3main.rel 78k3sub.rel -d78k3.elk

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 SAMP.DR (3) : F102 Directive syntax error
 Program Aborted.
```

- An error has occurred in the contents of the directive file. 78K3.ELK is referenced.

```
SAMP.DR(3) : F102 Directive syntax error
```

(10) Specify library file (-B)

Description format : -B file name
Default value : None

[Function]

Option -B specifies a file to be input as a library file.

[Application]

The linker retrieves the module referenced by the input module from a library file and joins only that module to the input module.

The purpose of a library file is to register two or more modules in a single file.

By creating library files that can be used in common with many programs, file management and operation become easier and more efficient. Specify option -B to input a library file to the linker.

[Explanation]

1) Only a disk-type file name can be specified as the file name.
2) The file name cannot be omitted.
3) If a file name which includes a path name is specified, a library file will be input from that path. If no library file exists in the specified path, an error occurs.
4) If a file name which does not include a path name is specified, a library file will be input from a path specified by option -I or from the default search path.
5) If option -B is specified two or more times, a library file will be input in a specified sequence. Up to 10 -B options may be specified.
6) For a detailed explanation of the method of creating library files, read Chapter 7, "Librarian."

[Example]

Input a library file (78K3.LIB).
(78K3SUB.REL is registered in the library file).

```
C>lk78k3 78k3main.rel -b78k3.lib

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(11) Specify library file read path (-I)

Description format : -I path name [, path name] ··· (two or more path names can be specified)
Default value : Path specified by environmental variable 'LIB78K3'
Current path, if no path is specified

[Function]

Option -I specifies input of a library file from a specified path.

[Application]

Use option -I to retrieve a library file from a certain path.

[Description]

1) Option -I is only available when a library file name is specified by option -B without including a path name.
2) Two or more specifications of -I are possible. Two or more paths can be specified by separating them with ','. A blank space cannot be inserted before or after the ','.
3) Up to 10 path names can be specified per link. When two or more path names are specified, the linker searches for library files in the specified order.
4) Even if no library file exists in the specified path, an error will not result.
5) If the path name is omitted, an abort error occurs.
6) If a library file is specified by option -B without including a path name, the linker will search the following paths.
   1. Path specified by option -I
   2. Path specified by environmental variable 'LIB78K3'.
   3. The current path

   If a library file with the specified name is not found in any of these paths, an error will occur.

[Example]

Search for a library file file from path LIB.

```
C>lk78k3 78k3main.rel 78k3sub.rel -b78k3.lib -i\lib

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(12) Specify parameter file (-F)

Description format : -F [file name]
Default value : This option and the input file name can only be entered on the startup line.

[Function]

Option -F specifies input of linker options and the input file name from a specified file.

[Application]

1) Specify option -F when the data required to start up the linker will not fit on the command line.
2) When you wish to repeatedly specify the same options each time assembly is performed, describe those options in a parameter file and specify option -F.

[Description]

1) Only a disk-type file name can be specified as 'file name'. If a device-type file name is specified, an abort error will occur.
2) If the file name is omitted, an abort error will occur.
3) Nesting of parameter files is not permitted. If option -F is specified within a parameter file, an abort error will occur.
4) The number of characters that can be described within a parameter file is unlimited.
5) Separate options or file names with a blank space, a tab or [↵].
6) Options and input file names described in a parameter file will be expanded at the position specified for the parameter file on the command line.
7) The expanded options specified last will take precedence.
8) All characters entered after ';' and before [↵] or 'EOF' will be interpreted as comments.
9) If option -F is specified two or more times, an abort error will occur.

[Example]

Perform link using a parameter file.

- Set the contents of the parameter file (78K3.PLK) as follows.

```
;Parameter file
78k3main.rel 78k3sub.rel -o78k3.lnk -p78k3.map -e
-ta:\tmp -g
```

- Enter the following on the command line.

```
C>lk78k3 -f78k3main.plk

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(13) Specify path for temporary file creation (-T)

Description format : -T path name
Default value : Creates a temporary file in the path specified by the environmental variable TMP.
When no path is specified, the temporary file is created in a current path.

[Function]

Option -T specifies a path in which a temporary file is created.

[Application]

Use option -T to specify the location for creation of a temporary file.

[Description]

1) Only a path can be specified as a path name.
2) The path name cannot be omitted.
3) Even if a previously created temporary file exists, if the file is not protected it will be overwritten.
4) As long as the required memory size is available, the temporary file will be expanded in memory. If not enough memory is available, the contents of the temporary file will be written to a disk. Such temporary files may be accessed later through the saved disk file.
5) Temporary files are deleted when assembly is finished. They are also deleted when assembly is aborted by pressing (CTRL-C).
6) The path in which the temporary file is to be created is determined according to the following sequence.
   1. The path specified by option -T
   2. The path specified by environmental variable TMP (when option -T is omitted)
   3. The current path (when TMP is not set)

   When a. or b. is specified, if the temporary file cannot be created in the specified path an abort error occurs.

[Example]

Specify output of a temporary file to directory 'TMP'.

```
C>lk78k3 78k3main.rel 78k3sub.rel -t\tmp

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(14) Specify warning message output (-W)

Description format : -W [level]
Default value : Outputs an ordinary error message

[Function]

Option -W specifies whether or not a warning message is output to the console.

[Application]

Specify the level at which a warning message will be output

[Description]

1) Only levels 0, 1 and 2 can be specified.
2) The following output levels are available:
   0 ··· No warning message is output.
   1 ··· Normal warning message is output.
   2 ··· Detailed warning message is output.

For a detailed explanation conditions under which warnings are output, see Table 11-2, "Linker Error Messages."

[Example of use]

Example Specify level 2 in option -W.

```
C>lk78k3 sample.rel -w2

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

***ERROR W420 File 'SAMPLE.REL' already has had error (s) / warning (s) by 'RA78K'

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

(15) Specify help (--)

Description format : --
Default value : No display

[Function]

Option -- displays a help message on the display.

[Application]

The help message is a list of explanations of the linker options. Refer to these when executing the linker.

[Description]

When option -- is specified, all other options are unavailable.

[Example]

When option -- is specified, a help message is output on the display.

```
C>lk78k3 --

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

usage : lk78k3 [option[ ...]] input-file[ ...] [option[ ...]]
The option is as follows ([] means omissible).
-ffile            :Input option or input-file name from specified file.
-dfile            :Read directive file from specified file.
-bfile            :Read library file from specified file.
-idirectory[,directory..]  :Set library file search path.
-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.
-tdirectory       :Set temporary directory.
-km/-nkm          :Output map list to link map file / Not.
-kd/-nkd          :Output directive file image 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.
-ll[page length]  :Specify link map file lines per page.
-lf/-nlf          :Add Form Feed at end of the link map file / Not.
-s[memory area]/-ns  :Create stack symbol [in specified memory area] / Not.
-g/-ng            :Output symbol information to load module file / Not.
-j/-nj            :Create load module file if fatal error occurred / Not.
-w                :Change warning level(n=0 to 2).
--                :Show this message.
DEFAULT ASSIGNMENT: -o -p -ne -km -kd -nkp -nkl
-ll66 -nlf -ns -g -nj -w1

directive file usage:
 MEMORY memory-area-name:(origin-value,size)[/memory-space-name]
 MERGE segment-name:[location-type-definition][merge-type-definition]
       [=memory-area-name][/memory-space-name]

 example: MEMORY ROM : (0H,4000H)
          MEMORY RAMA : (0H,100H)/EX1
          MERGE  CSEG1 : =ROM
          MERGE  DSEG1 : AT(80H)COMPLETE=RAMA/EX1
```

CHAPTER 6 OBJECT CONVERTER

The object converter inputs the load module file output by the RA78K3 linker (all reference address data must be determined at this point). It then converts this data into hexadecimal format and outputs it as an object module file.

The object converter also outputs the symbol data required for symbolic debugging as a symbol table file.

When an object converter error occurs, an error message appears on the display to clarify the cause of the error.

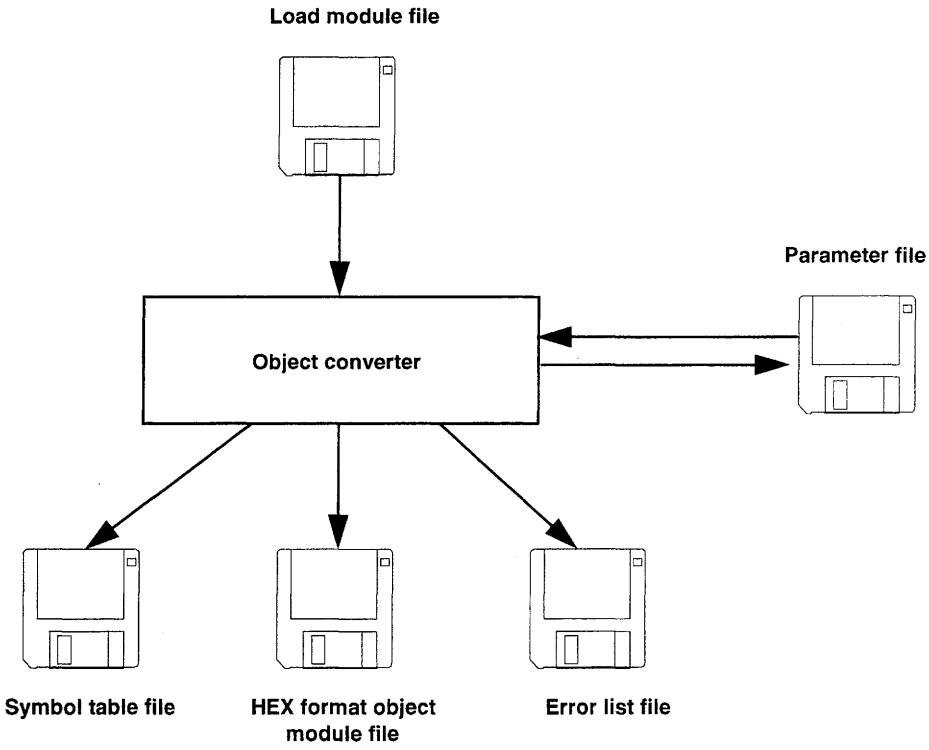
6.1 Object Converter Input and Output Files

Files input and output by the object converter are as shown below.

Table 6-1. Object Converter Input and Output Files

Type	File Name	Explanation	Default File Type
Input files	Load module files	• These are binary image files of the object codes output as a result of linking. • These files are output by the linker.	.LNK
	Parameter files	• These files contain the parameters for the executed programs. • These files are created by the user.	.POC
Output files	HEX format object module files	• These are files created by converting load module files into Intel standard HEX-format object format. • These files are used during mask ROM development and PROM program use. • These are files to be loaded to the in-circuit emulator.	.HEX
	Symbol table files	• These files contain the symbol data included in each module of an input file.	.SYM
	Error list files	• These files contain error data from the object conversion.	.EOC

Figure 6-1. Files Input and Output by the Object Converter



6.2 Functions of the Object Converter

(1) How the Object Converter Handles Extended Space

When a code is output to segments located in extended memory space, the object converter generates a separate HEX-format object module file for each space.
The object converter also generates a symbol table file for each space in extended space when symbols having ADDRESS or BIT attributes are defined for segments located in extended space. All symbols having NUMBER attributes are output to symbol table file generated for normal space.
Table 6-2 shows the file types of the HEX-format object module files and symbol table files generated for extended space.

Table 6-2. Output File Types for Extended Space

File	Normal Space	Extended 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 files

The HEX-format object module file output by the object converter can be input to a HEX loader such as a PROM programmer or a debugger.
The following is a HEX-format object module file of a sample program.

```
:10000000B900059F2813002431B900059F2813006B
:0C001000242156AF018302A807A8305637
:00000001FF
```

Lines 1 and 2 are the record of the object code, and the last line is the last record.

[HEX-format object module file format]

```
: 0C 0010 00 242156AF018302A807A83056 37
|  |  |  |              |               |
(1) (2) (3) (4)          (5)             (6)
```

(1) Record mark
Indicates beginning of record.

(2) Code number (2 digits)
Number of bytes in the code stored in the record. A maximum of 16 bytes can be stored.
The last record is indicated by 00H.

(3) Location address (4 digits)
The start address of the code displayed in the record is shown.
The last record is indicated by 0000H.

(4) Record type (2 digits)
00H indicates a data record, and 01H indicates the last record.

(5) Code (Max. 32 digits)
The object code is shown one byte at a time, with the upper 4 bits and lower 4 bits separated. A maximum of 16 bytes can be expressed in the code.
This field does not exist in the last record.

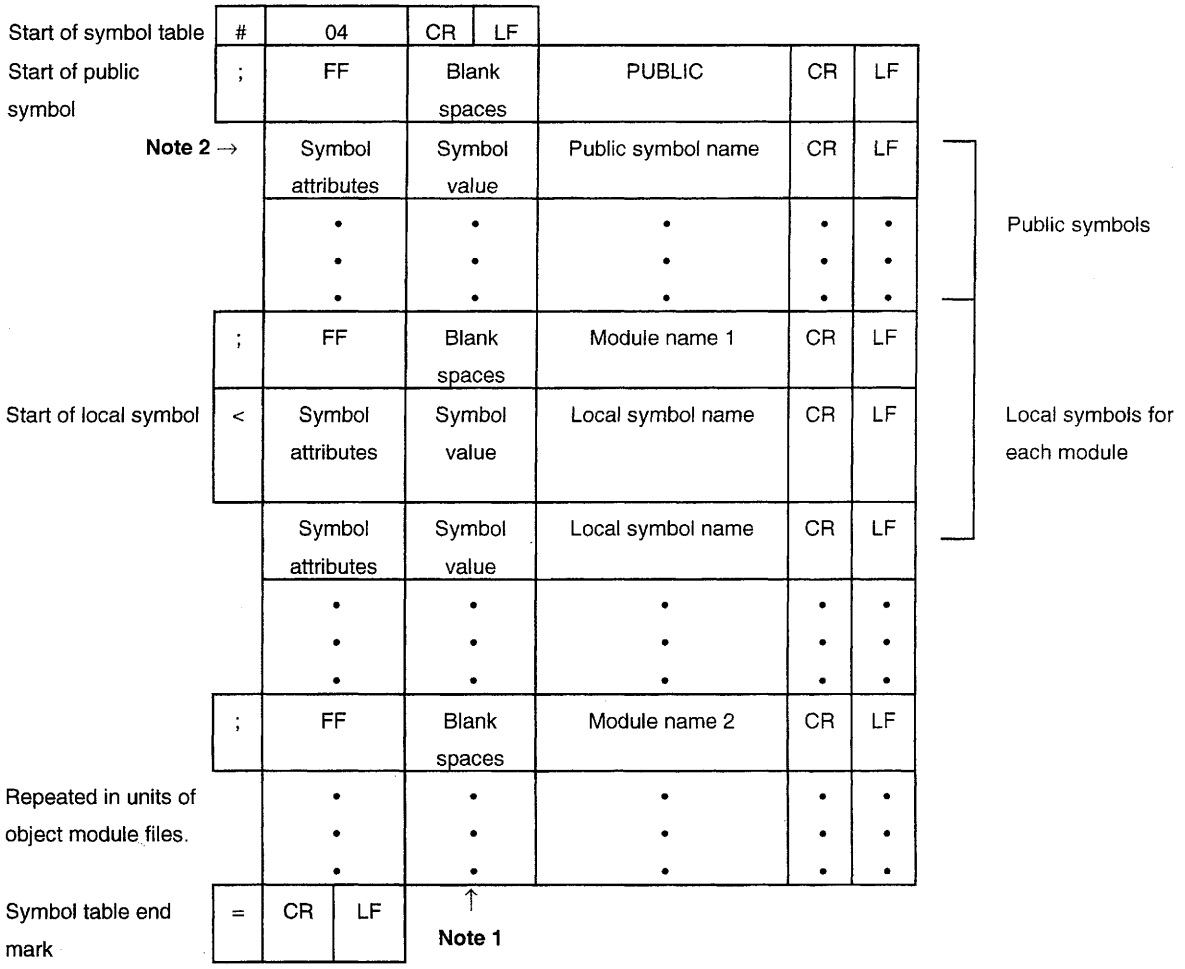
(6) Check sum (2 digits)
A value is input subtracting in order from 0 which counts down the data from the code number to the code.

(3) Symbol table file

The symbol table file output by the object converter is input to a debugger.
The following is the symbol table file of the sample program.

```
#04
;FF     PUBLIC
010000CONVAH
;FF     SAMPS
<010013SASC
010019SASC1
=
```

[Symbol Table File Formats]



Notes **1.** This field is fixed to 4 characters.

2. The symbol attributes take the following values.

Value	Symbol Attribute
00	Constant defined by the EQU directive
01	Label within a code segment
02	Label within a data segment
03	Bit symbol
FF	Module name

6.3 Object Converter Startup

6.3.1 Object converter startup

The following two methods can be used to start up the object converter.

(1) Startup from the command line

X>[path-name] oc78k3 [Δoption]···Δload-module-file-name [Δoption]···[Δ]

(1) (2) (3) (4) (5) (4)

(1) Current drive name
(2) Current directory name
(3) Object converter command file name
(4) This contains detailed directions for the action of the object converter.
(5) File name of the load module to be converted

Example C>oc78k3 78k3.lnk -osamle.hex

Caution If more than one object converter option is specified, separate the options with a space. For a detailed explanation of object converter options, see 6.4, "Object Converter Options."

(2) Startup from a parameter file

Use the parameter file when the data required to start up the object converter will not fit on the command line, or when the same object converter option is specified repeatedly each time object conversion is performed.
To start up the object converter from a parameter file, specify the specify parameter file option (-F) on the command line.

Start up the object converter from a parameter file as follows.

X>oc78k3 [Δload-module-file] Δ-f parameter-file-name
| |
(1) (2)

1) Specify parameter file option
2) A file which includes the data required to start up the object converter

Remark An editor is used to create the parameter file.

The rules for describing the contents of a parameter file are as follows.

[[[Δ] option [Δoption] ··· [Δ] Δ]] ···

Remarks 1. If the load module file name is omitted from the command line, only one load module file name can be specified in the parameter file.

2. The load module file name can also be described after the option.

3. Describe in the parameter file all object converter options and output file names that should be specified in the command line.

Example Create the parameter file (78K3.POC) using an editor.

- Contents of 78K3.POC

```
;parameter file
78k3.lnk -osample.hex
-ssample.sym -r
```

- Use parameter file 78K3.POC to start up the object converter.

```
C>oc78k3 -f78k3.poc
```

6.3.2 Execution start and end messages

(1) Execution start message

When the object converter is started up, an execution startup message appears on the display.

```
78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx
```

(2) Execution end message

If it detects no object conversion errors resulting from the object conversion, the object converter outputs the following message to the display and returns control to the operating system.

```
Object Conversion Complete,       0 error(s) and      0 warning(s) found.
```

If it detects an object conversion errors resulting from the object conversion, the object converter outputs the error number to the display and returns control to the operating system.

```
Object Conversion Complete,       3 error(s) and      0 warning(s) found.
```

If the object converter detects a fatal error during object conversion which makes it unable to continue link processing, the object converter outputs a message to the display, cancels object conversion and returns control to the operating system.

Example 1. A nonexistent load module file name is specified.

```
C>oc78k3 sample.lnk

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

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

In the above example, a nonexistent load module file is specified. An error results and the object converter aborts the object conversion.

Example 2. A nonexistent object converter option is specified.

```
C>oc78k3 78k3.lnk -a

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

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

In the above example, a nonexistent object converter option is specified. An error results and the object converter aborts the object conversion.
When an error message is displayed and object convertsion is aborted, look for the cause in Chapter 11, "Error Messages" and take action accordingly.

6.4 Object Converter Options

6.4.1 Types of object converter options

The object converter options are detailed instructions for the operation of the object converter. Object converter options are classified into 7 types.

The classifications of the object converter options and explanations of each type are shown below.

Table 6-3. Object Converter Options

Number	Classification	Option	Explanation
1	Specify HEX format object module file output	-O	Specifies the output of a HEX format object module file.
		-NO	
2	Specify symbol table file output	-S	Specifies output of a symbol table file.
		-NS	
3	Specify sort by object address order	-R	Sorts HEX format objects in the order of their addresses.
		-NR	
4	Specify object complement	-U	Outputs a specified complement value as an object code for an address area to which no HEX format object is output.
5	Specify error list file output	-E	Outputs an error list file.
		-NE	
6	Specify parameter file	-F	Inputs an input file name and options from a specified file.
7	Specify help	--	Displays a help message on the display.

Remark This table is presented as a brief introduction to the object converter options. When actually using the object converter options, see Appendix E.3, "List of Object Converter Options."

6.4.2 Explanation of object converter options

This section contains detailed explanations of each object converter option.

(1) Specify HEX format object module file output (-O/-NO)

Description format : -O [output file name]
: -NO
Default value : -O input file name.HEX
(The file type for extended space is '.H1 to .H15'.)

[Function]

1) Option -O specifies the output of a HEX format object module file. Option -O also specifies the output destination and output file name.
2) Option -NO specifies that no HEX format object module file is output.

[Application]

1) Specify the option -O to change the output destination and output file name of the HEX format object module file.
2) Specify option -NO when performing an object conversion only to output a symbol table file. This will shorten object conversion time.

[Description]

1) Specify a disk type file name for the 'output file name.'
If a device-type file name is specified, an abort error will result.
2) If the 'output file name' is omitted when option -O is specified, the HEX format object module file 'input file name.HEX' will be output to the current directory.
3) If only the path name is specified in 'output file name', 'input file name.HEX' will be output to the specified path.
4) If both options -O and -NO are specified at the same time, the option specified last takes precedence.

When a code is output to a segment located in extended space, the object converter generates a separate HEX format object module file for each space.
The file types of HEX format object module files generated for extended space are as follows.

File	Normal Space	Extended Space								
	REGULAR	EX1	EX2	EX3	EX4	EX5	···	EX13	EX14	EX15
HEX	.HEX	.H1	.H2	.H3	.H4	.H5	···	.H13	.H14	.H15

[Example]

Example 1. Output a HEX format object module file (SAMPLE.HEX).

```
C>oc78k3 78k3.lnk -osample.hex

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and    0 warning(s) found.
```

Example 2. Specify both options –NO and O.

```
C>oc78k3 78k3.lnk –no -o

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and    0 warning(s) found.
```

Option – NO is invalid, and option –O is valid.

(2) Specify symbol table file output (-S/-NS)

Description format : -S [output file name]
: -NS
Default value : -S input file name.SYM
(The file type for extended space is '.S1 to .S15'.)

[Function]

1) Option -S specifies the output of a symbol table file. Option -S also specifies the output destination and output file name.
2) Option -NS specifies that no symbol table file is output.

[Application]

1) Specify option -S to change the output destination and output file name of the symbol table file.
2) Specify option -NS when performing an object conversion only to output a HEX format object module file. This will shorten object conversion time.

[Description]

1) Specify a disk type file name for the 'output file name. '
If a device-type file name is specified, an abort error will result.
2) If the 'output file name' is omitted when option -S is specified, the symbol table file 'input file name.SYM' will be output to the current directory.
3) If only the path name is specified in 'output file name', 'input file name.SYM' will be output to the specified path.
4) If both options -S and -NS are specified at the same time, the option specified last takes precedence.

When a symbol having an ADDRESS or BIT attribute is defined for a segment located in extended space, the object converter generates a separate symbol table file for each space.
All symbols which have NUMBER attribute are output to a symbol table file in normal space.
The file types of symbol table files generated for extended space are as follows.

File	Normal Space	Extended Space								
	REGULAR	EX1	EX2	EX3	EX4	EX5	···	EX13	EX14	EX15
HEX	.HEX	.S1	.S2	.S3	.S4	.S5	···	.S13	.S14	.S15

[Example of use]

Example 1. Output a symbol table file (SAMPLE.SYM).

```
C>oc78k3 78k3.lnk -ssample.sym

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and    0 warning(s) found.
```

Example 2. Specify both options –NS and -S.

```
C>oc78k3 78k3.lnk -ns -s

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and    0 warning(s) found.
```

Option - NS is invalid, and option -S is valid.

(3) Specify sort by object address order (-R/-NR)

Description format : -R
: -NR
Default value : -NR

[Function]

1) Option -R outputs sorting of HEX format objects in order of address.
2) Option -NR outputs HEX format objects in the order in which they were stored in the load module file.

[Application]

Specify option -R when you need to sort HEX format objects in order of address.

[Description]

1) If both options -R and -NR are specified at the same time, the option specified last takes precedence.
2) If option -NO is specified, option -R/-NR becomes unavailable.

[Note]

When ordering a ROM code, be sure to specify option -R and sort the HEX-format objects in address sequence.

[Example]

Sort HEX format objects in order of address.

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

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and    0 warning(s) found.
```

(4) Specify object complement (-U)

Description format : -U complement value [, [start] , size]
Default value : None

[Function]

Option -U outputs a specified complement value as an object code for an address area to which no HEX format object has been output.

[Application]

Address areas to which no HEX format object has been output may become written with unnecessary code. When such addresses are accessed by the program for any reason, their action may be unpredictable. By specifying option -U, code can be written in advance to address areas to which no HEX format object has been output.

[Description]

1) The range of values that can be specified as complement values is as follows.

   0H ≤ complement value ≤ 0FFH

   Complement values can be specified in decimal or hexadecimal numbers. If a value outside the range or a value other than a numerical value is specified, an abort error occurs.

2) "Start" specifies the start address area for complement to be performed.
   The range of values that can be specified for start is as follows.

   0H ≤ start ≤ 0FF00H

   Start can be specified in decimal or hexadecimal numbers. If a value outside the range or a value other than a numerical value is specified, an abort error occurs. If start is omitted, 0 is assumed to be specified.

3) "Size" specifies the size of the address area for complement to be performed. The range of values that can be specified for size is as follows.

   0H ≤ size ≤ 0FF00H

   Size can be specified in decimal or hexadecimal numbers. If a value outside the range or a value other than a numerical value is specified, an abort error occurs. When start has been specified, size cannot be omitted.

4) If both start and size are omitted, the object converter performs the following processing.
   (a) If the target device for assembly contains internal ROM, the object converter interprets start and size to have the value specified in internal ROM.
   (b) If the target device for assembly does not contain internal ROM, the object converter interprets an error and aborts execution.
5) If option -U is specified two or more times, the item specified last takes precedence.
6) Specification formats for start and size in option -U and their interpretation are as follows.
   (a) -U Complement value
   If the target device for assembly contains internal ROM, the internal ROM range
   If the target device for assembly does not contain internal ROM, abort error
   (b) -U Complement value, size
   From address 0 to the size address
   (c) -U Complement value, start, size
   From start address to size address

[Note]

When ordering a ROM code, be sure to specify the option -U and make sure that the no vacant area exists in the internal ROM area.

[Example]

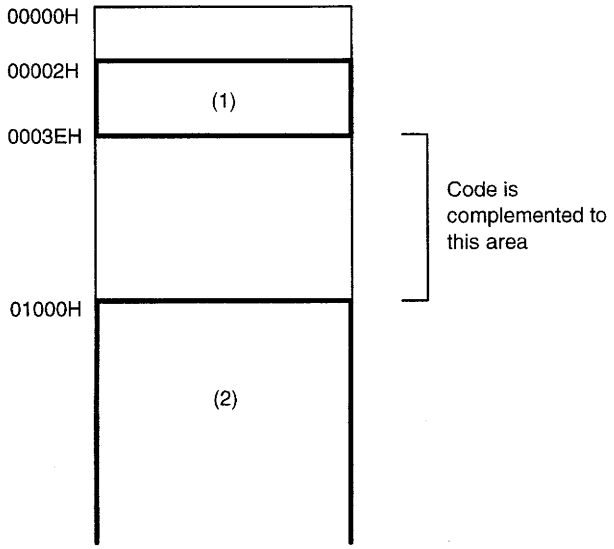
Complement an address area to which a HEX format object has not been output with code.

In the following example, it is supposed that a HEX format object module file exists. In this case, code cannot be written to the address area 003EH-0FFFH.

```
:020000000200FC
:100002002B41000BFC80FE2B40000944F7083A20EC
:100012001A6720FE2822006521FED350D25014FE1A
:10002200B900059F2835002431B900059F28350005
:0C003200242156AF0A8302A807A830560C
:01000003B5D0D0026A3···
:1010100024A5F622B667···
```

•
•
•

```
:00000001FF
```



00H is complemented to the address area 003EH-0FFFH.

```
C>oc78k3 78k3.lnk -u00h, 003eh, 0fffh

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and     0 warning(s) found.
```

(5) Specify error list file output (-E/-NE)

Description format : -E [output file name]
: -NE
Default value : -NE

[Function]

1) Option -E specifies the output of an error list file. Option -E also specifies the output destination and output file name.
2) Option -NE makes option -E unavailable.

[Application]

Specify option -E to change the output destination and output file name of the error list file.

[Explanation]

1) The file name of the error list file can be specified as a disk-type file name or as a device-type file name. However, if the device-type file name CLOCK is specified, an abort error will occur.
2) When option -E is specified and the output file name is omitted, the error list file name will be 'input file name.EOC'.
3) When option -E is specified and the drive name is omitted, the error list file will be output to the current drive.
4) If both options -E and -NE are specified at the same time, the option specified last takes precedence.

[Example]

Create an error list file (78K3.EOC).

```
C>oc78k3 78k3.lnk -e78k3.eoc

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

F100 Undefined symbol : CONVAH
Object Conversion Complete,      1 error(s) and    0 warning(s) found.
```

- This references 78K3.EOC

F100 Underfined symbol : CONVAH

(6) Specify parameter file (-F)

Description format : -F file name
Default value : Options and input file names can only be specified from the startup command line.

[Function]

Option -F specifies input of options and input file names from a specified file.

[Application]

1) Specify option -F when the data required to start up the object converter will not fit on the command line.
2) Specify option -F to repeatedly specify the same options each time object conversion is performed and to save those options to a parameter file.

[Explanation]

1) Only a disk-type file name can be specified as 'file name'. If a device-type file name is specified, an abort error will occur.
2) If the file name is omitted, an abort error will occur.
3) Nesting of parameter files is not permitted. If option -F is specified within a parameter file, an abort error will occur.
4) The number of characters that can be described within a parameter file is unlimited.
5) Separate options or input file names with a blank space, a tab or [↵].
6) Options and input file names described in a parameter file will be expanded at the position specified for the parameter file on the command line.
7) The expanded options specified last will take precedence.
8) All characters entered after ';' or '#' and before [↵] or 'EOF' will be interpreted as comments.
9) If option -F is specified two or more times, an abort error will occur.

[Example]

Perform object conversion using a parameter file.

- Set the contents of parameter file 78K3.POC as follows.

```
;parameter file
78k3.lnk -osample.hex
-ssample.sym -r
```

- Enter the following on the command line.

```
C>oc78k3 -f78k3.poc

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and    0 warning(s) found.
```

(7) Specify help (--)

Description format : --
Default value : No display

[Function]

Option -- displays a help message on the display.

[Application]

The help message is a list of explanations of the object converter options. Refer to these when executing the object converter.

[Description]

When option -- is specified, all other options are unavailable.

[Example]

When option -- is specified, a help message is output on the display.

```
C>oc78k3 --

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

usage : oc78k3 [option[ ...]] input-file [option[ ...]]
The option is as follows ([] means omissible).
-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.
-e[file]/-ne    :Create the error list file [with the 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 edits RA78K3 object module files and library files in units of 1 module.
The librarian also outputs a list file.
If a librarian error occurs, an error message is output to the display indicating the cause of the error.

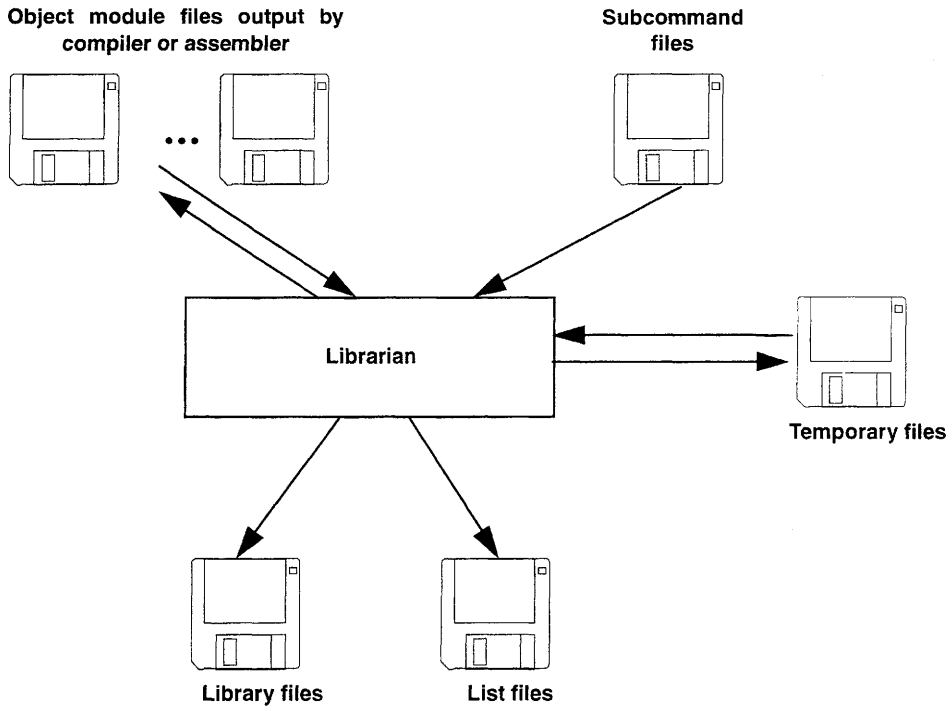
7.1 Files Input and Output by the Librarian

The files input and output by the librarian are as follows.

Table 7-1. Files Input and Output by the Librarian

Type	File Name	Explanation	Default File Type
Input files	Subcommand files	• These files contain the execute program command and the parameters. • These files are created by the user.	None
Output files	List files	• These files are the result of output of library data.	.LST
Input/ output files	Object module files	• These are object module files output by the assembler or compiler.	.REL
	Library files	• These files input the library files output by the librarian and update the contents.	.LIB
	Temporary files	• These files are automatically generated by the librarian when forming a library. They are deleted when execution of the librarian is complete.	Lbxxxxxx.$$y y=1 to 6

Figure 7-1. Files Input and Output by the Librarian



7.2 Functions of the Librarian

(1) Formation of a library of modules

The assembler and linker create 1 file for every module they output.
This means that if a large number of modules are created, the number of files also grows. The RA78K3 therefore includes a function for collecting a number of object modules in a single file. This function is called module library formation, and a file which is organized as a library is called a library file.
A library file can be input to the linker. By creating a library file consisting of modules common to many programs, users can make file management and operation efficient and easy when performing modular programming.

(2) Editing of library files

The librarian incorporates the following editing functions for library files.

1) Addition of modules to library files
2) Deletion of modules from library files
3) Replacement of modules in library files
4) Retrieval of modules from library files
   (For detailed explanations of these functions, see 7.5, "Subcommands.")

(3) Output of library file data

The librarian incorporates functions for the editing and output of the following items of data stored in library files.

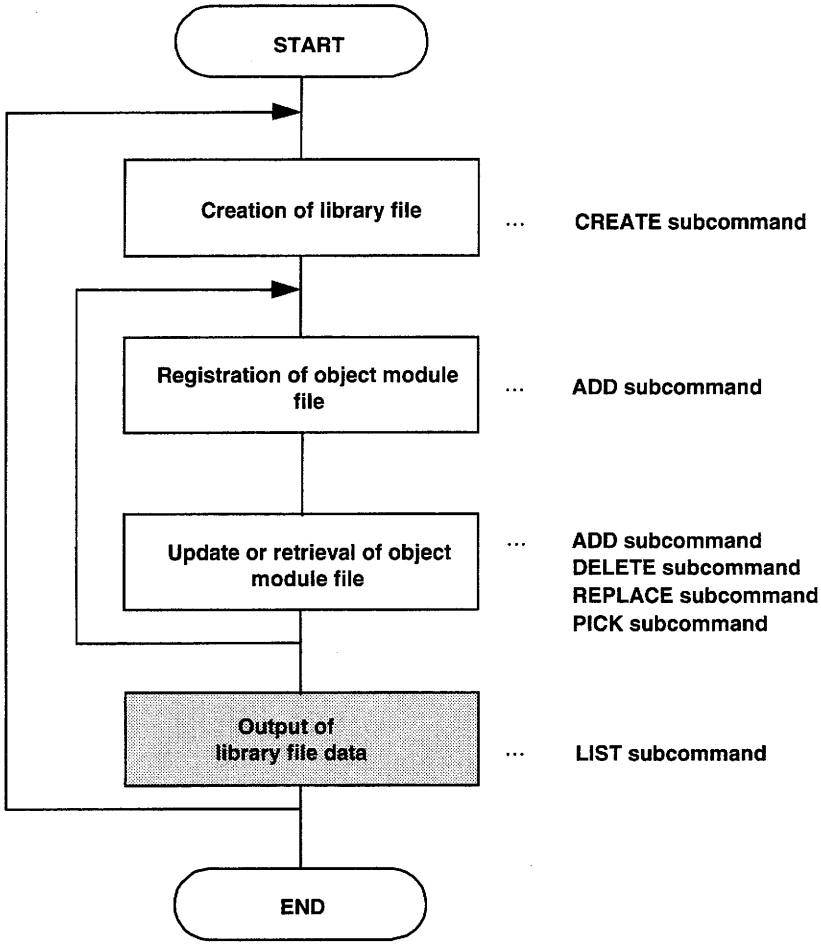
1) Module names
2) Created programs
3) Date of registration
4) Date of update
5) PUBLIC symbol data

Caution The librarian performs functions 2) and 3) listed above using subcommands. The librarian determines each subcommand in order while performing processing. For an explanation of the operation of subcommands, see 7.5, "Subcommands."

(4) Procedure for creating a library file

The general procedure for creating library files is as follows.

Figure 7-2. Procedure for Creating a Library File



7.3 Librarian Startup

7.3.1 Librarian startup

The following two methods can be used to start up the librarian.

(1) Startup from the command line

```
X>[path-name] lb78k3 [Δoption]···
  |       |       |       |
 (1)     (2)     (3)     (4)
```

(1) Current drive name
(2) Current directory name
(3) Librarian command file name
(4) This contains detailed directions for the action of the librarian

Example C>lb78k3 -ll20 -lw80

Caution If more than one librarian option is specified, separate the options with a space. For a detailed explanation of librarian options, see 7.4, "Librarian Options."

Example Startup message

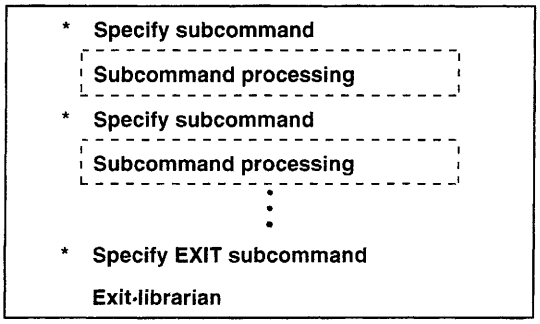
When the librarian is started up, the following startup message appears on the display.

```
78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
*
```

After an asterisk (*), specify a librarian subcommand.

```
*create 78k3.lib
*add 78k3.lib 78k3main.rel 78k3sub.rel
*exit
```

When input of subcommands is finished, processing of each subcommand begins. When processing of one subcommand is complete, '*' appears again on the screen and the librarian waits for the next subcommand to be entered. The librarian repeats this operation until the EXIT subcommand is entered.



Up to 128 characters can be specified in 1 line.
If all the required operand data will not fit on 1 line, use '&' to continue specification on the next line. Specification can be continued up to 15 lines.

(2) Startup from a subcommand file

A subcommand file is a file in which librarian subcommands are stored.
If a subcommand file is not specified when the librarian is started up, multiple subcommands must be specified after the '*' appears. By creating a subcommand file, these multiple subcommand files can all be processed at once.
A subcommand file can also be used when the same subcommand is specified repeatedly each time library formation is performed.
When using a subcommand file, describe '<' before the file name.

Start up the librarian from a subcommand file as follows.

```
X>lb78k3Δ <subcommand file name  [Δoption]···
          |                      |
         (1)                    (2)
```

(1) Be sure to add this when specifying a subcommand file
(2) File in which subcommands are stored

(a) Use an editor to create the subcommand file.
(b) The rules for describing the content of a subcommand file are as follows.

Subcommand name	**operand data**
Subcommand name	**operand data**
⋮	
EXIT	

(c) When repeating one subcommand, describe '&' at the end of each line to indicate continuation.
(d) Everything described from a semicolon (';') to the end of the line will be assumed to be a comment, and will not be interpreted by the librarian command.
(e) If the last subcommand in a subcommand file is not the EXIT subcommand, the librarian will automatically interpret an EXIT subcommand.
(f) The librarian reads subcommands from the subcommand file and processes them. The librarian quits after it completes processing of all subcommands in the subcommand file.

Example Create the subcommand file (78K3.SLB) using an editor.

- Contents of 78K4.SLB

```
;library creation command
create 78k3.lib
add 78k3.lib 78k3main.rel &
78k3sub.rel
exit
```

- Use subcommand file 78K3.SLB to start up the librarian.

```
C>lb78k3 <78k3.slb
```

7.3.2 Execution start and end messages

(1) Execution start message

When the librarian is started up, an execution startup message appears on the display.

```
78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
*
```

(2) Execution end message

The librarian does not output an execution end message. When the user enters the EXIT subcommand after all processing is complete, the librarian returns control to the operating system.

```
*create 78k3.lib
*add 78k3.lib 78k3main.rel 78k3sub.rel
*exit
```

If the librarian detects a fatal error which makes it unable to continue librarian processing, the librarian outputs a message to the display and returns control to the operating system.

Example 1. If '<' is not specified at the beginning of the library file.

```
C>lb78k3 78k3.slb

78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
A003 Unrecognized string '78k3.slb'
Usage: LB78K3 [options]
```

In this example, an error occurs because '<' is not specified at the beginning of the library file, and execution of the librarian is aborted.

Example 2. A nonexistent librarian option is specified.

```
C>lb78k3 -a

78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
A018 Option is not recognized '-a'
Usage: LB78K3 [options]
```

In the above example, a nonexistent librarian option is specified. An error results and the librarian aborts librarian execution.

When an error message is displayed and library formation is aborted, look for the cause in Chapter 11, "Error Messages" and take action accordingly.

7.4 Librarian Options

7.4.1 Types of librarian options

The librarian options are used to specify the format of list files and the file creation path for temporary files. Librarian options are classified into 3 types.

Table 7-2. Librarian Options

Number	Classification	Option	Explanation
1	Specify list file format	-LW	Changes the number of characters that can be printed in 1 line in a list file.
		-LL	Changes the number of lines that can be printed in 1 page in a list file.
		-LF	Inserts a line feed code at the end of a list file.
		-NLF	
2	Specify path for temporary file creation	-T	Creates a temporary file in a specified path.
3	Specify help	--	Displays a help message on the display.

Remark This table is presented as a brief introduction to the librarian options. When actually using the librarian options, see Appendix E.4, "List of Librarian Options."

7.4.2 Explanation of library options

The following is a detailed explanation of the library options.

(1) Specify list file format (LW, -LL, -LF/-NLF)

(a) -LW

Description format : -LW [number of characters]
Default value : -LW132 (80 characters in the case of display output)

[Function]

Option -LW changes the number of characters that can be printed in 1 line in a list file.

[Application]

Specify option -LW to change the number of characters that can be printed in 1 line in a list file.

[Description]

1) The range of number of characters that can be specified with option -LW is shown below.
(In the case of display output, this number is 80)

$$72 \leq \text{number of characters printed on 1 line} \leq 132$$

If a numerical value outside this range, or something other than a numerical value, is specified, an abort error occurs.

2) If the number of characters is omitted, 132 will be specified. If the list file is output to the display, 80 is specified.
3) The specified number of characters does not include the carriage return and feed codes.
4) If the LIST subcommand is not specified, option -LW is ignored.
5) If option -LW is specified 2 or more times, the last specified item will take precedence.

[Example]

Specify 80 as the number of characters per line in a list file.

```
C>lb78k3 -lw80

78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
*create 78k3.lib 78k3sub.rel              ←Subcommand
*list 78k3.lib                            ←Subcommand
```

(b) -LL

Description format : -LL [number of lines]
Default value : -LL66 (No page breaks in the case of display output)

[Function]

Option -LL specifies the number of lines that can be printed in 1 page in a list file.

[Application]

Specify option -LL to change the number of lines that can be printed in 1 page in a list file.

[Description]

1) The range of number of lines that can be specified with option -LL is shown below.

   20 ≤ number of lines printed on 1 page ≤ 32767

   If a numerical value outside this range, or something other than a numerical value, is specified, an abort error occurs.
2) If the number of lines is omitted, 66 will be specified.
3) If the number of lines specified is 0, no page breaks will be made.
4) If the LIST subcommand is not specified, option -LL is ignored.
5) If option -LL is specified 2 or more times, the last specified item will take precedence.

[Example]

Specify 20 as the number of lines per page in a list file.

```
C>lb78k3 -ll20

78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
*create 78k3.lib 78k3sub.rel                ←Subcommand
*list 78k3.lib                              ←Subcommand
```

(c) -LF/-NLF

Description format: -LF
: -NLF
Default value : -NLF

[Function]

1) Option -LF inserts a form feed (FF) code at the end of a list file.
2) The -NLF option makes the -LF option unavailable.

[Application]

If you wish to add a page break after the contents of a list file are printed, specify option -LF to add a form feed code.

[Description]

1) If the LIST subcommand is not specified, option -LF is ignored.
2) If both options -LF and -NLF are specified at the same time, the option specified last takes precedence.

[Example]

Example Add a form feed code to a list file.

```
C>lb78k3 -lf

78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
*create 78k3.lib 78k3sub.rel                ←Subcommand
*list 78k3.lib                              ←Subcommand
```

(2) Specify path for temporary file creation (-T)

Description format : -T path name
Default value : Created in the path specified by the environmental variable TMP.
If no path is specified, the temporary file is created in the current path.

[Function]

Option -T creates a temporary file in a specified path.

[Application]

Use option -T to specify the location for creation of a temporary file.

[Description]

1) Only a path can be specified as a path name.
2) The path name cannot be omitted.
3) Even if a previously created temporary file exists, if the file is not protected it will be overwritten.
4) As long as the required memory size is available, the temporary file will be expanded in memory. If not enough memory is available, the contents of the temporary file being created will be written to disk. Such temporary files may be accessed later through the saved disk file.
5) Temporary files are deleted when library formation is finished. They are also deleted when library formation is aborted by pressing (CTRL-C).
6) The path in which the temporary file is to be created is determined according to the following order.

    1. The path specified by option -T
    2. The path specified by environmental variable TMP (when option -T is omitted)
    3. The current path (when TMP is not set)

    When 1. or 2. is specified, if the temporary file cannot be created in the specified path an abort error occurs.

[Example]

Specify output of a temporary file to directory TMP.

```
C>lb78k3 -t\tmp

78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
*
```

(3) Specify help (--)

Description format : --
Default value : No display

[Function]

Option -- displays a help message on the display.

[Application]

The help message is a list of explanations of the subcommands. Refer to these when executing the librarian.

[Description]

When option -- is specified, all other options are unavailable.

[Example]

When option -- is specified, a help message is output on the display.

```
C>lb78k3 --

78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
+-------------------------------------------------------------------------------+
| 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

The subcommands provide detailed directions for the operation of the librarian. Subcommands are classified into eight types.

Table 7-3. Subcommands

No.	Subcommand Name	Abbrev.	Explanation
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 module in a library file with other modules.
5	PICK	P	Retrieves a module from a library file.
6	LIST	L	Outputs data on modules in a library file.
7	HELP	H	Displays a help message on the display.
8	EXIT	E	Exits librarian.

Remark For a detailed explanation of the subcommands, read Appendix F, "List of Subcommands."

7.5.2 Explanation of subcommands

The following is a detailed explanation of the function and operation of each subcommand.

• General format of command files

*Subcommand [Δoption] Δlibrary-file-name [Δoption] transaction [Δoption]
| |
(1) (2)

(1) The library file name specified immediately before can be replaced with '.'.
(2) Transaction = Δobject-module-file-name Δlibrary-file-name [∇(∇module-name [∇,…])]

(1) CREATE

Description format : CREATEΔlibrary file name [Δtransaction]
Abbreviated format : C

[Function]

The CREATE subcommand creates a new library file.

[Description]

1) The size of the created library file becomes 0.
2) When a transaction is specified, a module is registered at the same time as the library file is created.
3) Library file name: If a file with the same name already exists, it will be overwritten.
4) Transaction: An object module file carrying the same public symbol as the public symbol in the library file cannot be registered.
   A module with the same name as a module in the library file cannot be registered.
5) If an error occurs, processing is interrupted and the library file cannot be created.

[Example]

Example 1. Create a new library file (78K3.LIB)

*create 78k3.lib

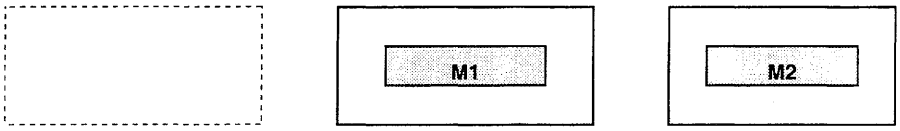
<Before file creation>

<After file creation>

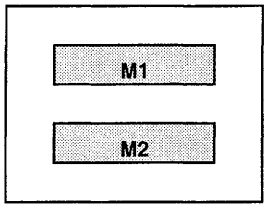
Example 2. Register modules M1 and M2 at the same time as a library file is created.

*create 78k3.lib m1.rel m2.rel

<Before file creation>



<After file creation>
78K3.LIB



(2) ADD

Description format : ADDΔlibrary file name Δtransaction
Abbreviated format : A

[Function]

The ADD subcommand adds a module to a library file.

[Description]

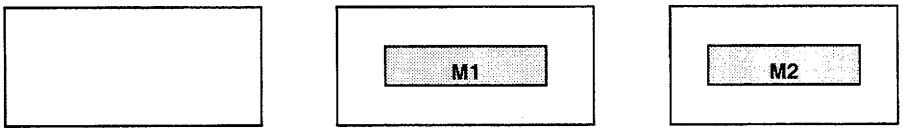
1) A module can be added to a library file even if no modules are currently stored in the library.
2) If a module with the same name as the module to be added already exists in the library file, an error occurs.
3) If the module to be added carries the same public symbol as the public symbol in the library file, an error occurs.

[Example]

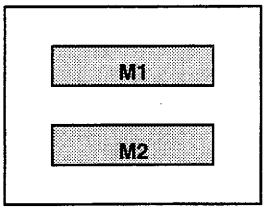
Example 1. Add ,modules (M1 and M2) to a library file (78K3.LIB)

*add 78K3.lib m1.rel m2.rel

<Before addition>
78K3.LIB



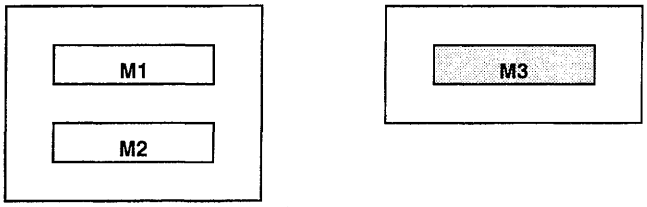
<After addition>
78K3.LIB



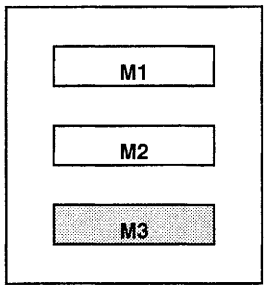
Example 2. Add a module (M3) to a library file (78K3.LIB).

*add 78k3.lib m3.rel

<Before addition>
78K3.LIB



<After addition>
78K3.LIB



(3) DELETE

Description format : DELETE△library file name ∇(∇module name [∇,···]∇)
Abbreviated format : D

[Function]

The DELETE subcommand deletes a module from a library file.

[Description]

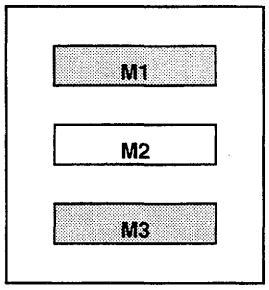
1) If the specified module does not exist in the library file, an error occurs.
2) If an error occurs, processing is interrupted and the condition of the library file will not be changed.

[Example]

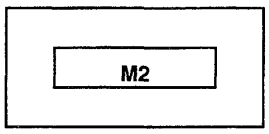
Delete modules (M1, M3) from a library file (78K3.LIB).

```
*delete 78k3.lib m1.rel m3.rel
```

<Before deletion>
78K3.LIB



<After deletion>
78K3.LIB



(4) REPLACE

Description format : REPLACE△library file name△transaction
Abbreviated format: R

[Function]

The REPLACE subcommand replaces module in a library file with the module in other object module files.

[Description]

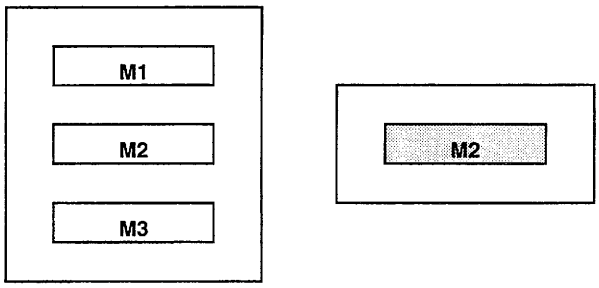
1) If no module in the library file has the same name as the replacement module, an error will result.
2) If a public symbol contained in the replacement module is the same as a public symbol in the library file, an error will occur.
3) The file name of the replacement object module must be the same as the file name used in registration.
4) If an error occurs, processing is interrupted and the condition of the library file will not be changed.

[Example]

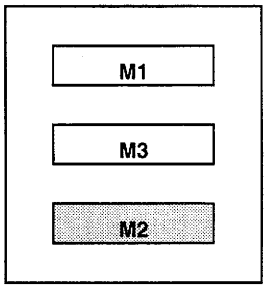
Replace a module (M2) in a library file (78K3.LIB).

*replace 78k3.lib m2.rel

<Before replacement>
78K3.LIB



<After replacement>
78K3.LIB



Because the new module (M2) is registered after the module (M2) in the library file is deleted, M2 is last in order in the library file.

(5) PICK

Description format : PICK△library file name ∇(∇module-name [∇,…]∇)
Abbreviated format : P

[Function]

The PICK subcommand retrieves a specified module from an existing library file.

[Description]

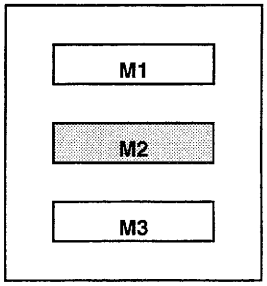
1) The retrieved module becomes an object module file with the file name under which it was registered in the library file.
2) If the specified module name does not exist in the library file, an error will result.
3) If an error occurs, processing is interrupted. However, if an error occurs when two or more modules are specified, the modules retrieved before the module which caused the error become available and are saved onto disk.

[Example]

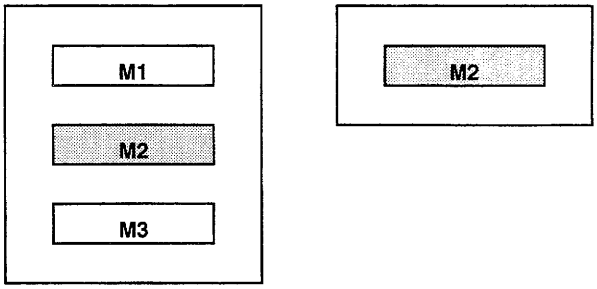
Example Retrieve a module (M2) from a library file (78K3.LIB).

*pick 78k3.lib m2.rel

<Before retrieval>
78K3.LIB



<After retrieval>
78K3.LIB



(6) LIST

Description format: LIST [∆option] ∆library-file-name [∇(∇module-name [∇,···]∇)

- Option : -PUBLIC/-NOPUBLIC
  : - O ∇filename

Abbreviated format: L

[Function]

The LIST subcommand outputs data on modules in a library file.

[Description]

1) Multiple options may be specified.
2) -O:
   A device-type file name can be specified as the output file name.
   If the output file name is omitted, an error occurs.
   If the file type is omitted, the librarian assumes that 'input file name.LST' is entered.
3) -PUBLIC/-NOPUBLIC:
   This option can be selected by specifying only the underlined characters.
   -PUBLIC specifies output of public symbol data.
   -NOPUBLIC makes -PUBLIC unavailable.
   If -PUBLIC and -NOPUBLIC are specified at the same time, the last specified option takes precedence.

[Example]

Output a module data in a library file (78K3.LIB) to a list file (78K3.LST). Specify option -P so that public symbol data will be output.

```
*list -p -o78k3.lst 78k3.lib
```

• List file (78K3.LST) is referenced.

```
78K/III Series 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

Description format : HELP
Abbreviated format : H

[Function]

The HELP command displays a help message on the display.

[Description]

The help message is a list of the subcommands and explanations for each. Specify the HELP command or option -- to refer to this message during librarian execution.

[Example]

Specify the HELP command to output the HELP message.

*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

Description format : EXIT
Abbreviated format : E

[Function]

The EXIT subcommand exits the librarian.

[Description]

Use this subcommand to exit the librarian.

[Example]

Exit the librarian.

```
*exit
```

CHAPTER 8 LIST CONVERTER

The list converter inputs assemble list files and object module files output by the assembler and load module files output by the linker.

The list converter then embeds actual addresses in the relocatable addresses and symbols in the input file and outputs an absolute assembly list. This eliminates the troublesome task of looking at an assemble list while referring to a link map.

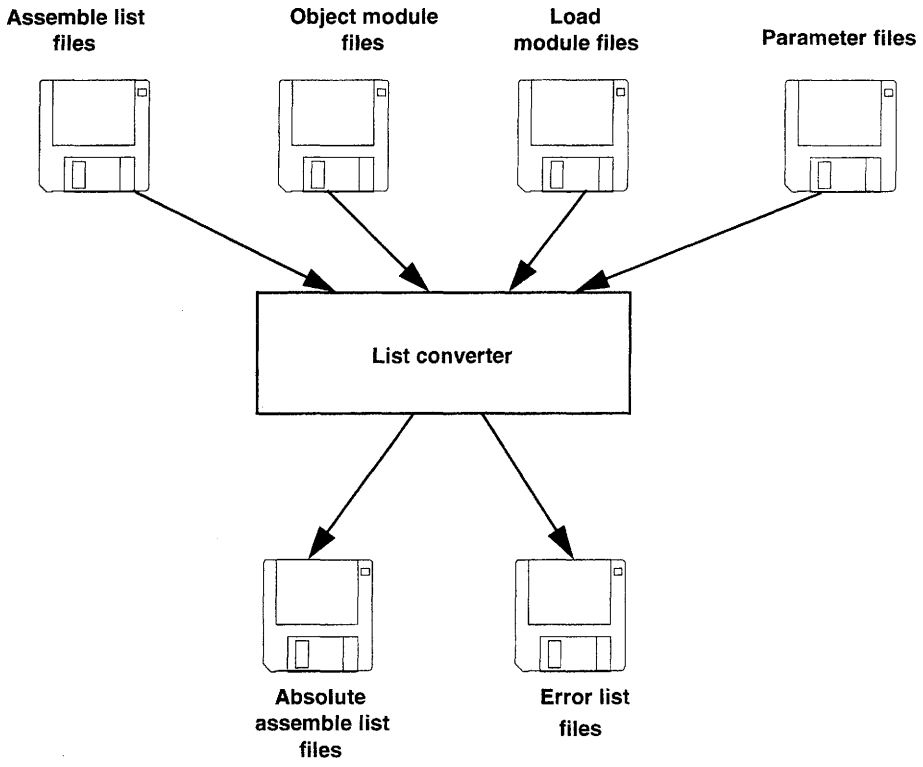
8.1 List Converter Input and Output Files

Files input and output by the list converter are as shown below.

Table 8-1. Assembler Input and Output Files

Type	File Name	Explanation	Default File Type
Input files	Object module files	• These are binary files including relocation data and symbol data regarding machine language data and machine language location addresses.	.REL
	Assemble list files	• These are files containing assembly data such as assemble lists and cross-reference lists.	.PRN
	Load module files	• These are binary image files which contain object code as a result of linking.	.LNK
	Parameter files	• These files contain the parameters for the executed program. • These files are created by the user.	.PLV
Output files	Absolute assemble list files	• This is a list file which embeds actual addresses in relocatable addresses and symbols in the input file.	.P
	Error list files	• These are files containing error data generated during list conversion.	.ELV

Figure 8-1. Files Input and Output by the List Converter



8.2 Functions of the List Converter

The following is a comparison of the advantages and disadvantages of relocatable assemblers with respect to absolute assemblers.

[Advantages]

1) Relocatable assemblers can be developed by a team of several personnel.
2) Relocatable assemblers can be divided into modules for easy development and storage.
3) Relocatable assemblers support library management.
4) Relocatable assemblers are appropriate for development of large-scale programs.

[Disadvantages]

1) The addresses in the assemble lists of relocatable assemblers do not agree with their actual, physical addresses.
2) The values of external symbols become 0 in the assemble lists of relocatable assemblers. To find out the actual values of external symbols, a link map must be referred to.
3) Relocatable values in assemble lists are different from actual values.

The above disadvantages particularly reduce productivity in the areas of debugging and storage because of the considerable documentation they require. The list converter offers a solution to these disadvantages of relocatable assembler packages.

1) The absolute assemble list output by the list converter agrees completely with the addresses used in actual program operation.
2) The actual values of external symbols are embedded in the list.
3) Relocatable values are embedded in the list as actual values.
4) For the symbol values in symbol tables or cross-reference lists, the actual values are embedded in the list.

Example 1. Relocation embedding

- Assemble list

```
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
```

- Absolute assemble list

```
15    15  ----                        CSEG
16    16  0002  B900          CONVAH: MOV     A,#0
17    17  0004  059F                  ROL4    [HL]
18    18  0006  R281500               CALL    !SASC
19    19  0009  2431                  MOV     B,A
20    20
21    21  000B  B900                  MOV     A,#0
22    22  000D  059F                  ROL4    [HL]
23    23  000F  R281500               CALL    !SASC
24    24  0012  2421                  MOV     C,A
25    25
26    26  0014  56                    RET
```

Example 2. Embedding of object codes

- Assemble list

```
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
```

- Absolute assemble list

```
15    15 ----                          CSEG
16    16 0002  B900          CONVAH:  MOV     A,#0
17    17 0004  059F                   ROL4    [HL]
18    18 0006 R281500                 CALL    !SASC
19    19 0009  2431                   MOV     B,A
20    20
21    21 000B  B900                   MOV     A,#0
22    22 000D  059F                   ROL4    [HL]
23    23 000F R281500                 CALL    !SASC
24    24 0012  2421                   MOV     C,A
25    25
26    26 0014  56                     RET
```

8.3 List Converter Startup

8.3.1 List converter startup

Two methods can be used to start up the list converter.

(1) Command-line startup

```
X>lcnv78k3 [Δoption]···Δinput-file-name [Δoption]···[Δ]
 |      |    |          |          |
(1)    (2)  (3)        (4)        (3)
```

(1) Current drive name
(2) Command file name of the list converter
(3) Enter detailed instructions for the operation of the list converter.
(4) Primary name of assemble list

Example C>lcnv78k3 78k3main -I78k3.lnk

Cautions

1. **In (3) above, when specifying two or more list converter options, separate the list converter options with a blank space. For a detailed explanation of list converter options, see 8.4, "List Converter Options."**
2. **Use the extension .PRN for (4) above.**
3. **In (4) above, if only the primary name of the assemble list is specified in the command line, the primary names of the object module file and load module file must be identical to the primary name of the assemble list file.**
   The file types must also be as shown below.

File Name	Type
Object module type	.REL
Load module file	.LNK

Use an option when specifying a file which is different in the primary name.

(2) Startup from a parameter file

Use the parameter file when the data required to start up the list converter will not fit on the command line, or when the same list converter option is specified repeatedly each time list conversion is performed.

To start up the list converter from a parameter file, specify the specify parameter file option (-F) on the command line.

Start up the list converter from a parameter file as follows.

X>lcnv78k3 [Δinput file name] Δ-f parameter file name

(1) (2)

(1) Specify a parameter file option

(2) A file which includes the data required to start up the list converter

Remark Create the parameter file using an editor.

The rules for describing the contents of a parameter file are as follows.

[[[Δ] option [Δoption] ··· [Δ]Δ]]···

1) If the input file name is omitted from the command line, only 1 input file name can be specified in the parameter file.
2) The input file name can also be described after the option.
3) Describe in the parameter file all list converter options and output file names that should be specified in the command line.

Example Create the parameter file (78K3.PLV) using an editor.

• Contents of 78k3.PLV

```
;parameter file
78k3main -l78k3.lnk
-e78k3.elv
```

• Use parameter file (78K3.PLV) to start up the list converter.

C>lcnv78k3 -f78k3.plv

8.3.2 Execution start and end messages

(1) Execution start message

When the list converter is started up, an execution startup message appears on the display.

```
List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
```

(2) Execution end message

If it detects no list conversion errors resulting from the list conversion, the list converter outputs the following message to the display and returns control to the operating system.

```
Conversion complete.
```

If the list converter detects a fatal error during list conversion which makes it unable to continue list conversion processing, the list converter outputs a message to the display, cancels list conversion and returns control to the operating system.

Example 1. A nonexistent assemble list file is specified.

```
C>lcnv78k3 sample

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

A006 File not found '-ISAMPLE. LINK'
Program aborted.
```

In this example, an error occurs because a nonexistent assemble list file is specified, and the processing is aborted.

Example 2. A nonexistent list converter option is specified.

```
C>lcnv78K4 78K4main -a

List Conversion Program for RA78K/IV Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation xxxx
        A018 Option is not recognized '-a'
Program aborted.
```

In this example, an error occurs because a nonexistent list converter option is specified, and the processing is aborted.

When the list converter outputs an error message and aborts list conversion, look for the cause in Chapter 11, "Error Messages" and take action accordingly.

8.4 List Converter Options

8.4.1 Types of list converter options

The list converter options are detailed instructions for the operation of the list converter. List converter options are classified into 7 types.

Table 8-2. List Converter Options

Number	Classification	Option	Explanation
1	Specify object module file input	-R	Inputs an object module file.
2	Specify load module file input	-L	Inputs a load module file.
3	Specify symbol name case	-CA	Does not distinguish uppercase and lowercase in symbol names.
		-NCA	
4	Specify absolute assemble list file output	-O	Specifies output of an absolute assemble list file.
5	Specify error list file output	-E	Outputs an error list file.
		-NE	
6	Specify parameter file	-F	Inputs the input file name and options from a specified file.
7	Specify help	--	Displays a help message on the display.

The above table is provided as an introduction to the list converter options. When actually using the list converter options, read Appendix E.5, "List of List Converter Options."

8.4.2 Explanation of list converter options

This section contains detailed explanations of each list converter option.

(1) Specify object module file input (-R)

Description format : -R input file name
Default value : -R assemble list file name.REL

[Function]

Option -R specifies the input of an object module file.

[Application]

When the primary name of an object module file is different from the primary name in the assemble list file, or if its file type is not ".REL", specify option -R.

[Description]

1) If a fatal error occurs, the absolute assemble list file cannot be output.
2) If only the primary name of the input file name is specified, the list converter will assign the file type '.REL' and input the file.

[Example]

Assemble list file name is 78K3MAIN.PRN, the object module file name is SAMPLE.REL, and the load module file name is 78K3.LNK.

```
C>lcnv78k3 78k3main -rsample.rel

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
Conversion complete.
```

(2) Specify load module file input (-L)

Description format : -L [input-file-name]
Default value : -L assemble-list-file-name.LNK

[Function]

Option -L specifies the input of a load module file.

[Application]

When the primary name of a load module file is different from the primary name in the assemble list file, or if its file type is not ".LNK", specify option -L.

[Description]

1) If a fatal error occurs, the absolute assemble list file cannot be output.
2) If only the primary name of the input file name is specified, the list converter will assign the file type '.LNK' and input the file.

[Example]

Assemble list file name is 78K3MAIN.PRN and the load module file name is SAMPLE.LNK.

```
C>lcnv78k3 78k3main -lsample.lnk

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
Conversion complete.
```

(3) Specify symbol name case (-CA/-NCA)

Description format : -CA
: -NCA
Default value : -CA

[Function]

1) Option -CA specifies that no distinction is made between uppercase and lowercase characters in a symbol name.
2) Option -NCA specifies that a distinction is made between uppercase and lowercase characters in a symbol name.

[Application]

Use option -CA when you need to ignore the distinction between upper case and lower case.

[Description]

1) When option -CA is specified, the assembler converts lowercase characters in a symbol name to uppercase and outputs them to an object.
2) When option -NCA is specified, the assembler outputs the symbol name to an object without converting lowercase characters to uppercase.

[Note]

Match this option with option -CA/-NCA of the assembler.

[Example]

Example Specify that a distinction is made between uppercase and lowercase characters in a symbol name.

```
C>lcnv78k3 78k3main -nca

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
Conversion complete.
```

(4) Specify absolute assemble list file output (-O)

Description format : -O [output-file-name]
Default value : -O assemble list file name.P

[Function]

Option -O specifies the output of an absolute assemble list file. Option -O also specifies the output destination and output file name.

[Application]

Use option -O to change the output destination and output file name of the absolute assemble list file.

[Description]

1) A file name can be specified as a disk-type file name or as a device-type file name. However, only CON, PRN, NUL and AUX can be specified as device-type file names. If CLOCK is specified, an abort error will occur.
2) If the same device is specified for the file name as for the error file, an abort error will occur.
3) If the output file name is omitted when option -O is specified, the absolute assemble list file name will become 'assemble list file name.P'.
4) If only the primary name of the output file name is specified, the list converter will assign the file type '.P' and output the file.
5) If the drive name is omitted when option -O is specified, the absolute assemble list file will be output to the current drive.

[Example]

Example1. Create an absolute assemble list file (SAMPLE.P)

```
C>lcnv78k3 78k3main -osample.p

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
Conversion complete.
```

Example2. Output the absolute assemble list file to printer

```
C>lcnv78k3 78k3main -oprn

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
Conversion complete.
```

(5) Specify error list file output (-E/-NE)

Description format : -E [output file name]
: -NE
Default value : -NE

[Function]

1) Specify option -E to specify the output of an error list file. This option also specifies the output destination and output file name.
2) Option -NE makes option -E unavailable.

[Application]

Specify option -E to save error messages in a file.

[Description]

1) The file name of the error list file can be specified as a disk-type file name or as a device-type file name. However, if the device-type file name CLOCK is specified, an abort error will occur.
2) If the device specified in the file name is the same as that specified in the absolute assemble list file, an abort error will occur.
3) If option -E is specified and the output file name is omitted, the error list file name will be 'assemble list file name.ELV'.
4) If only the primary name of the output file name is specified, the list converter will assign the file type '.ELV' and output the file.
5) If the drive name is omitted when option -E is specified, the error list file will be output to the current drive.
6) If both options -E and -NE are specified at the same time, the option specified last takes precedence.

[Example]

Create an error list file (SAMPLE.ELV).

```
C>lcnv78k3 78k3main -esample.elv

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
*** WARNING A105 Segment name is not found in load module file 'DATA'
Program aborted.
```

• The error list file (SAMPLE.ELV) is referenced.

```
Pass1: start
*** ERROR A105 Segment name is not found in load module file 'DATA'
```

(6) Specify parameter file (-F)

Description format : -F file name
Default value : Options and input file names can only be entered on the startup line.

[Function]

Option -F specifies input of options and the input file name from a specified file.

[Application]

1) Specify option -F when the data required to start up the list converter will not fit on the command line.
2) When you wish to repeatedly specify the same options each time list conversion is performed, describe those options in a parameter file and specify option -F.

[Description]

1) Only a disk-type file name can be specified as 'file name'. If a device-type file name is specified, an abort error will occur.
2) If the file name is omitted, an abort error will occur.
3) If only the primary name of the file name is specified, the list converter will assign the file type '.PLV' and open the file.
4) Nesting of parameter files is not permitted. if option -F is specified within a parameter file, an abort error will occur.
5) The number of characters that can be described within a parameter file is unlimited.
6) Separate options or input file names with a blank space, a tab or [↵].
7) Options and input file names described in a parameter file will be expanded at the position specified for the parameter file on the command line.
8) The expanded options specified last will take precedence.
9) If option -F is specified two or more times, an abort error will occur.

[Example]

Start up list converter using a parameter file.

The contents of the parameter file (78K3.PLV) are as follows.

```
;parameter file
78k3main -l78k3.lnk
-e78k3.elv
```

Enter the following on the command line.

```
C>lcnv78k3 -f78k3.plv

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
Conversion complete.
```

(7) Specify help (--)

Description format : --
Default value : No display

[Function]

Option -- displays a help message on the display.

[Application]

The help message is a list of explanations of the list converter options. Refer to these when executing the list converter.

[Description]

When option -- is specified, all other options are unavailable.

[Example]

When option -- is specified, a help message is output on the display.

```
C>lcnv78k3 --

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

usage : lcnv78k3 [option[...]] input-file [option[...]]
The option is as follows([] means omissible).
-r[file]:Specify object module file.
-l[file]:Specify load module file.
-o[file]:Specify output list file (absolute assemble list file).
-ffile  :Input option or input-file name from specified file.
-e[file]:Create error list file.
-ca/-nca:Convert alphabet to capital for symbol / Not.
--      :Show this message.
DEFAULT ASSIGNMENT: -ca
```

[MEMO]

CHAPTER 9 PROGRAM OUTPUT LIST

The following is an explanation of the formats and other information for the lists output by each program.

- Lists Output by the Assembler
  - Assemble list file header
  - Assemble list
  - Symbol list
  - Cross-reference list
  - Error list

- Lists Output by the Linker
  - Link list file header
  - Map list
  - Public symbol list
  - Local symbol list
  - Error list

- List Output by the Object Converter
  - Error list

- List Output by the Librarian
  - Library data output list

- List Output by the List Converter
  - Absolute assemble list
  - Error list

9.1 Lists Output by the Assembler

The assembler outputs the following lists.

Output List File Name	Output List Name
Assemble list file	Assemble list
	Symbol list
	Cross-reference list
Error list file	Error list

9.1.1 Assemble list file headers

The header is always output at the beginning of an assemble list file.

[Output format]

```
78K/III Series Assembler (1)Vx.xx (2)                    Date:(3)xx xxx xxxx Page:(4)  1
(5)
Command:  (6)-c310 78k3main.asm -o -p -e
Para-file:(7)
In-file:  (8)78K3MAIN.ASM
Obj-file: (9)78K3MAIN.REL
Prn-file:(10)78K3MAIN.PRN
```

[Explanation of output items]

Item	Details
(1)	Assembler version no.
(2)	Title character string Character string specified by option -LH or TITLE control instruction
(3)	Date of assemble list creation
(4)	Page no.
(5)	Subtitle character string Character string specified by SUBTITLE control instruction
(6)	Command-line image
(7)	Contents of parameter file
(8)	Input source module file name
(9)	Output object module file name
(10)	Assemble list file name

9.1.2 Assemble list

The assemble list outputs the results of the assemble with error messages (if errors occur).

[Output format]

```
      Assemble list

 ALNO  STNO ADRS   OBJECT    M I  SOURCE STATEMENT
 (1)   (2)      (3)(4)       (5)
 (1)1  (2)1                  (5)$       PROCESSOR(310)
 (1)2  (2)2                  (5)
 (1)3  (2)3                  (5)        NAME    SAMPM
 (1)4  (2)4                  (5);*******************************************
 (1)5  (2)5                  (5);*                                         *
 (1)6  (2)6                  (5);*      HEX -> ASCII Conversion Program    *
 (1)7  (2)7                  (5);*                                         *
 (1)8  (2)8                  (5);*            main-routine                 *
 (1)9  (2)9                  (5);*                                         *
(1)10 (2)10                  (5);*******************************************
(1)11 (2)11                  (5)
(1)12 (2)12                  (5)        PUBLIC  MAIN,START
(1)13 (2)13                  (5)        EXTRN   CONVAH
(1)14 (2)14
        •
        •
        •
   33    33 (6)0016 (8)6521EF             MOVW   DE,#STASC  ;set DE <- store ASCII
code table
   34    34                               MOV
(7)***  ERROR F201, STNO    34 (   31) Syntax error
   35    35 (6)0019  (8)50                MOV    [DE+],A
        •
        •
        •
Segment informations:

ADRS   LEN      NAME

(9)FE20  (10)0003H  (11)DATA
(9)0000  (10)0002H  (11)CODE
(9)0000  (10)001EH  (11)?CSEG

 Target chip : (12)uPDxxxxx
 Device file : (13)Vx.xx
Assembly complete,     (14)3 error(s) and     (15)0 warning(s) found. (   (16)34)
```

[Explanation of output items]

Item	Details
(1)	Line no. of source module image
(2)	Line no. (including expansion of INCLUDE files and macros)
(3)	Macro display M: This is a macro definition line. #n: This is a macro expansion line. n is the nest level. Blank: This is not a macro definition or expansion line.
(4)	INCLUDE display In : Within an INCLUDE file. n is the nest level. Blank : INCLUDE file is not used.
(5)	Source program statement
(6)	Location counter value
(7)	Fatal error/warning occurrence line
(8)	Relocation data R: Object code or symbol value is changed by the linker. Blank: Object code or symbol value is not changed by the linker.
(9)	Segment address
(10)	Segment size
(11)	Segment name
(12)	RA78K3 target device
(13)	Device file version no.
(14)	Number of fatal errors
(15)	Number of warnings
(16)	Final error line

9.1.3 Symbol list

A symbol list outputs the symbols (including local symbols) defined in a source module.

[Output format]

```
                Symbol Table List

    VALUE      ATTR    RTYP  NAME                    VALUE     ATTR     RTYP     NAME
           (2)CSEG           (4)?CSEG                       (2)CSEG           (4)CODE
(1)   ------H         (3)EXT (4)CONVAH                      (2)DSEG           (4)DATA
(1)  FE20H   (2)ADDR         (4)HDTSA              (1)0H    (2)ADDR  (3)PUB   (4)MAIN
           (2)MOD            (4)SAMPM              (1)0H    (2)ADDR  (3)PUB   (4)START
(1)  FE21H   (2)ADDR         (4)STASC
```

[Explanation of output items]

Item	Details		
(1)	Symbol value		
(2)	Symbol attributes	SABIT	: BIT attribute symbol (saddr.bit)
	CSEG : Code segment name	SFBIT	: BIT attribute symbol (sfr.bit)
	DSEG : Data segment name	RBIT	: BIT attribute symbol (A.bit, X.bit,
	BSEG : Bit segment name		PSW.bit, PSWL.bit, PSWH.bit)
	MOD : Module name	RBBIT	: BIT attribute symbol (br.bit)
	SET : Symbol defined by SET directive	RWBIT	: BIT attribute symbol (wr.bit)
	NUM : NUMBER attribute symbol	Blank	: External reference symbol declared
	DNUM : DNUMBER attribute symbol		by EXTRN or EXTBIT
	ABIT : BIT attribute symbol (addr.bit)	*****	: Undefined symbol
(3)	Symbol reference format		
	EXT : External reference symbol declared by EXTRN		
	EXTB : External reference symbol declared by EXTBIT		
	PUB : External reference symbol declared by PUBLIC		
	Blank : Local symbol, segment name, macro name, module name		
	***** : Undefined symbol		
(4)	Defined symbol name		

9.1.4 Cross-reference list

A cross-reference list outputs data indicating where (on what line) symbols are defined in a source module.

[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#
(1)HDTSA          (2) FE20H         (4)ADDR                 (6)DATA     (7)16#     28     29
(1)MAIN           (2)    0H         (4)ADDR     (5)PUB      (6)CODE     (7)12@     20#
(1)SAMPM                            (4)MOD                              (7)3#
(1)START          (2)    0H    (3) R (4)ADDR     (5)PUB      (6)?CSEG    (7)12@     20     23#
(1)STASC          (2) FE21H         (4)ADDR                 (6)DATA     (7)17#     33
```

[Explanation of output items] (1/2)

Item	Details		
(1)	Defined symbol name		
(2)	Symbol value		
(3)	Relocation attributes R: Relocatable symbol E: External symbol Blank: Absolute symbol *: Undefined symbol		
(4)	Symbol attributes	SABIT	: BIT attribute symbol (saddr.bit)
	CSEG : Code segment name	SFBIT	: BIT attribute symbol (sfr.bit)
	DSEG : Data segment name	RBIT	: BIT attribute symbol (A.bit, X.bit,
	BSEG : Bit segment name		PSW.bit, PSWL.bit, PSWH.bit)
	MOD : Module name	RBBIT	: BIT attribute symbol (br.bit)
	SET : Symbol defined by SET directive	RWBIT	: BIT atttribute symbol (wr.bit)
	NUM : NUMBER attribute symbol	Blank	: External reference symbol declared
	DNUM : DNUMBER attribute symbol		by EXTRN or EXTBIT
	ADDER : ADDRESS attribute symbol	*****	: Undefined symbol
	ABIT : BIT attribute symbol (addr.bit)		

[Explanation of output items] (2/2)

Item	Details
(5)	Symbol reference format EXT : External reference symbol declared by EXTRN EXTB : External reference symbol declared by EXTBIT PUB : External reference symbol declared by PUBLIC Blank : Local symbol, segment name, macro name, module name ***** : Undefined symbol
(6)	Defined symbol name
(7)	Definition/reference line no. Definition line: XXXXX# Reference line: XXXXX∇ (∇= 1 blank) EXTRN declaration, EXTBIT declaration, PUBLIC declaration: XXXXX@

9.1.5 Error list

An error list stores the error messages output when the assembler is started up.

[Output format]

```
  Pass1 Start
(1)ERAMAIN.ASM((2)25) : (3)F202 (4)Illegal operand
(1)ERAMAIN.ASM((2)31) : (3)F201 (4)Syntax error
(1)ERAMAIN.ASM((2)34) : (3)F201 (4)Syntax error
  Pass2 Start
(1)ERAMAIN.ASM((2)25) : (3)F202 (4)Illegal operand
(1)ERAMAIN.ASM((2)31) : (3)F201 (4)Syntax error
(1)ERAMAIN.ASM((2)34) : (3)F201 (4)Syntax error
```

[Explanation of output items]

Item	Details
(1)	Name of source module file in which error occurred
(2)	Line on which error occurred
(3)	Error no.
(4)	Error message

9.2 Lists Output by the Linker

The linker outputs the following lists.

Output List File Name	Output List Name
Link list file	Map list
	Public symbol list
	Local symbol list

9.2.1 Link list file headers

The header is always output at the beginning of a link list file.

[Output format]

```
78K/III Series Linker (1)Vx.xx                    Date:(2)xx xxx xxxx Page:(3) 1

Command:   (4) 78k3main.rel 78k3sub.rel -g -o78k3.map -d78k3.dr
Para-file: (5)
Out-file:  (6)78K3.MAP
Map-file:  (7)78K3MAIN.MAP
Direc-file:(8)78K3.DR
Directive: (9)MEMORY ROM:(0000H,3FFFH)
              MEMORY RAM:(0D000H,2EFFH)



*** Link information ***

(10)   3 output segment(s)
(11)   3EH byte(s) real data
(12)   23 symbol(s) defined
```

[Explanation of output items]

Item	Details
(1)	Linker version no.
(2)	Date of link list file creation
(3)	Page no.
(4)	Command-line image
(5)	Contents of parameter file
(6)	Output load module file name
(7)	Link list file name
(8)	Directive file name
(9)	Directive file contents
(10)	Number of segments output to load module file
(11)	Size of data output to load module file
(12)	Number of symbols output to load module file

9.2.2 Map list

The map list outputs data on the location of segments.

[Output format]

```
*** Memory map ***

(1)  SPACE=REGULAR

    MEMORY=(2)ROM
    BASE ADDRESS=(3)0000H    SIZE=(4)3FFFH
           OUTPUT   INPUT    INPUT      BASE          SIZE
           SEGMENT  SEGMENT  MODULE     ADDRESS
         (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   (9)0002H    (10)0020H
                  (7)?CSEG  (8)SAMPS   (9)0022H    (10)001CH
(5)* gap *                            (9)003EH    (10)3FC1H

    MEMORY=(2)RAM
    BASE ADDRESS=(3)D000H    SIZE=(4)2EFFH
           OUTPUT   INPUT    INPUT      BASE          SIZE
           SEGMENT  SEGMENT  MODULE     ADDRESS
(5)* gap *                            (9)D000H    (10)2E20H
         (6)DATA                       (9)FE20H    (10)0003H   (11)DSEG AT
                  (7)DATA   (8)SAMPM   (9)FE20H    (10)0003H
(5)* gap *                            (9)FE23H    (10)00DCH

   Target chip : (12)uPDxxxxx
   Device file : (13)Vx.xx
```

[Explanation of output items]

Item	Details
(1)	Memory space name
(2)	Memory area name
(3)	Memory area start address
(4)	Memory area size
(5)	Output group Displays 'gap' for areas where nothing is located.
(6)	Segment names output to load module file
(7)	Segment names read from object module file
(8)	Input module name
(9)	Segment start address
(10)	Output/input segment size
(11)	Segment type and reallocation attributes
(12)	Target device for this assemble
(13)	Device file version no.

9.2.3 Public symbol list

A public symbol list outputs data on public symbols defined in an 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
```

[Explanation of output items]

Item	Details			
(1)	Name of module in which public symbols are defined			
(2)	Symbol attributes		SABIT	: BIT attribute symbol (saddr.bit)
	CSEG	: Code segment name	SFBIT	: BIT attribute symbol (sfr.bit)
	DSEG	: Data segment name	RBIT	: BIT attribute symbol (A.bit, X.bit,
	BSEG	: Bit segment name		PSW.bit, PSWL.bit, PSWH.bit)
	MOD	: Module name	RBBIT	: BIT attribute symbol (br.bit)
	SET	: Symbol defined by SET directive	RWBIT	: BIT attribute symbol (wr.bit)
	NUM	: NUMBER attribute symbol	Blank	: External reference symbol declared
	DNUM	: DNUMBER attribute symbol		by EXTRN or EXTBIT
	ADDER	: ADDRESS attribute symbol	*****	: Undefined symbol
	ABIT	: BIT attribute symbol (addr.bit)		
(3)	Symbol value			
(4)	Public symbol name			

9.2.4 Local symbol list

A local symbol list outputs data on local symbols defined in an 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
(1)SAMPM     (2)CSEG                         (4)CODE
(1)SAMPM     (2)CSEG                         (4)?CSEG
(1)SAMPS     (2)MOD                          (4)SAMPS
(1)SAMPS     (2)CSEG                         (4)?CSEG
(1)SAMPS     (2)ADDR        (3)0035H         (4)SASC
(1)SAMPS     (2)ADDR        (3)003BH         (4)SASC1
```

[Explanation of output items]

Item	Details			
(1)	Name of module in which local symbols are defined			
(2)	Symbol attributes		SABIT	: BIT attribute symbol (saddr.bit)
	CSEG	: Code segment name	SFBIT	: BIT attribute symbol (sfr.bit)
	DSEG	: Data segment name	RBIT	: BIT attribute symbol (A.bit, X.bit,
	BSEG	: Bit segment name		PSW.bit, PSWL.bit, PSWH.bit)
	MOD	: Module name	RBBIT	: BIT attribute symbol (br.bit)
	SET	: Symbol defined by SET directive	RWBIT	: BIT attribute symbol (wr.bit)
	NUM	: NUMBER attribute symbol	Blank	: External reference symbol declared
	DNUM	: DNUMBER attribute symbol		by EXTRN or EXTBIT
	ADDER	: ADDRESS attribute symbol	*****	: Undefined symbol
	ABIT	: BIT attribute symbol (addr.bit)		
(3)	Symbol value			
(4)	Local symbol name			

9.2.5 Error list

An error list stores the error messages output when the linker is started up.

[Output format]

```
*** ERROR (1)F405 (2)Undefined symbol 'CONVAH' in file '78K3MAIN.REL'
```

[Explanation of output items]

Item	Details
(1)	Error no.
(2)	Error message

9.3 List Output by the Object Converter

The object converter outputs the following list.

Output List File Name	Output List Name
Error list file	Error list

9.3.1 Error list

Error messages output when the object converter is started up are stored in an error list.

[Output format]

Same as error list output by the linker.

9.4 List Output by the Librarian

The librarian outputs the following list.

Output List File Name	Output List Name
List file	Library data output list

9.4.1 Library data output list

The library data output list outputs data on the modules in a library file.

[Output format]

```
  78K/III Series librarian Vx.xx     DATE :(1) xx xxx xx                    PAGE (2)1
  LIB-FILE NAME : (3)78K3.LIB         ((4)xx xxx xx)
(5)0001  (6)78K3MAIN.REL     ((7)xx xxx xx)
    (8)MAIN                                (8)START
    NUMBER OF PUBLIC SYMBOLS :   (9)2
  (5)0002  (6)78K3SUB.REL      ((7)xx xxx xx)
    (8)CONVAH
    NUMBER OF PUBLIC SYMBOLS :   (9)1
```

[Explanation of output items]

Item	Details
(1)	Date of list creation
(2)	Number of pages
(3)	Library file name
(4)	Date of library file creation
(5)	Module serial no. (beginning from 0001)
(6)	Module name
(7)	Date of module creation
(8)	Public symbol name
(9)	Number of public symbols defined in module

9.5 Lists Output by the List Converter

The list converter outputs the following lists.

Output List File Name	Output List Name
Absolute assemble list file	Absolute assemble list
Error list file	Error list

9.5.1 Absolute assemble list

The absolute assemble list embeds absolute values in the assemble list and outputs the list.

[Output format]

Same as for the assemble list output by the assembler.

9.5.2 Error list

Error messages output when the list converter is started up are stored in an error list.

[Output format]

Same as for the error list output by the assembler.

[MEMO]

CHAPTER 10 GETTING THE MOST FROM THE RA78K3

This chapter introduces some methods that will help you to use the RA78K3 efficiently.

10.1 Improving Operating Efficiency (EXIT Status Function)

When any of the programs of the RA78K3 finishes processing, the program stores the maximum level of errors occurring during processing as the "EXIT status," and returns control to the operating system.

The EXIT statuses are as follows:

- Normal operation: 0
- WARNING occurs: 0
- FATAL ERROR occurs: 1
- ABORT: 2

The exit status can be used to create a batch file, making operation more efficient.

[Example]

Contents of the batch file (RA.BAT)

```
ra78k3 %1.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
ra78k3 %2.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
lk78k3 %1.rel %2.rel -o%3.lnk -g
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
oc78k3 %3.lnk
echo off
IF ERRORLEVEL 1 GOTO ERR
GOTO EXIT
:ERR
echo Error occurred
:EXIT
```

- Perform processing using batch file (RA.BAT).

A>ra.bat

10.2 Preparing the Development Environment (Environmental Variables)

The RA78K3 supports the following environmental variables for preparing the software development environment.

PATH : Search path for execution format
INC78K3 : Search path for include file (assembler only)
LIB78K3 : Search path for library file (linker only)
TMP : Path for creating temporary files

When developing programs, it is a good idea to create a subdirectory in which to collect all related files. This will make program development easier and more convenient.

[Example]

Example Contents of AUTOEXEC.BAT

```
;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)
```

(1) Because this path is specified, execution format files are retrieved from directories in the order A:\BIN, A:\BAT, A:\RA78K3.
(2) The assembler retrieves include files from the directory A:\RA78K3\INCLUDE.
(3) The linker retrieves library files from A:\RA78K3\LIB.
(4) Each program creates a temporary file in A:\TMP.

10.3 Interrupting Program Execution

Execution of each program can be interrupted by entering CTRL-C from the keyboard.

If 'break on' is specified during execution of AUTOEXEC.BAT, control is returned to the operating system regardless of the timing of the key input. When 'break off' is specified, control is only returned to the operating system during screen display. In this case, all open temporary files and output files are deleted.

10.4 Making the Assemble List Easy to Read

Display a title in the header of an assemble list using option -LH or the TITLE control instruction. By displaying a title that briefly indicates the contents of the assemble list, the contents of the assemble list can be made easy to see at a glance.

When the SUBTITLE control instruction is used, a subtitle can also be displayed. For information on control instructions, see Chapter 4, "Control Instructions" in the language manual.

[Example]

Print a title in the header of an assemble list file.

```
C>ra78k3 78k3main.asm –lw90 -lhRA78k3_MAINROUTINE

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 start
Pass2 start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

This references 78K3MAIN.PRN.

```
78K/III Series Assembler Vx.xx RA78K3_MAINROUTINE  Date:xx xxx xxxx Page: 1
                                                    └── Title

Command: 78k3main.asm -lw90 -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
      ┆
```

10.5 Reducing Program Startup Time

10.5.1 Describing a control instruction in the source program

Control instructions which have the same functions as the options normally specified in assembler startup can be specified in advance in the source program. This eliminates the need to specify options every time the assembler is started up.

[Example of use]

```
$       PROCESSOR(310)

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

        PUBLIC  MAIN,START
        EXTRN   CONVAH
                ┆
```

10.5.2 Creating parameter files and subcommand files

When executing any of the RA78K3's programs (assembler, linker, object converter and list converter), if all the necessary data will not fit on the command line, or if the same options are specified every time the program is executed, create a parameter file.

Also, subcommands can be registered in a subcommand file in the librarian. This makes object module library formation easy.

[Example of use 1]

Example 1. Create a parameter file and perform assembly.
Contents of parameter file 78K3MAIN.PRA

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

• Contents of parameter file 78K3.SLB

```
;
;library creation command
;
create 78k3.lib
;
add 78k3.lib 78K3main.rel &
78k3sub.rel
;
exit
```

• Enter the following on the command line.

```
C>lb78k3 <78k3.slb
```

10.6 Object Module Library

The assembler and linker create 1 file for every 1 output module. When there are many object modules, therefore, the number of files also increases. The RA78K3 incorporates a function for collecting a number of object modules in a single file. This function is called module library formation. A file which forms such a library is called a library file.

Library files can be input to the linker. Therefore, when performing modular programming, library files containing common modules can be created, enabling efficient file management and operation.

[MEMO]

CHAPTER 11 ERROR MESSAGES

This chapter explains the causes of error messages output by the RA78K3's programs (assembler, linker, object converter and librarian), and the action to be taken by the user.

11.1 Overview of Error Messages

Error messages output by the RA78K3 are divided into the following 3 levels.

(1) Abort errors (Axxx)

An error has occurred which makes the program unable to continue processing. The program quits (interrupts) immediately.

If the abort error is found on the startup line, processing ends when another startup-line error is found.

(2) Fatal errors (Fxxx)

An execution error has occurred. When another error is found, the program quits (interrupts) without generating an output object.

When a fatal error occurs, to clarify that an output object is not generated, if an object with the same name exists, that object is deleted.

(3) Warning errors (Wxxxx)

Compiler: Creates an output object that may be different from that expected by the user but that runs normally.

Assembler: Creates an output object as the user expected though an error occurs at the position independent of code generation.

Remark In a program executed in conversational format, the execution ends normally unless an abort error occurs.

Assembler error messages are classified as follows.

Each assembler error message is explained beginning on the next page.

- A0xx --- Command line analysis error
- A9xx --- File or system error
- A1xx --- Other abort error
- F2xx --- Statement description error
- F3xx --- Expression error
- F4xx --- Symbol error
- F5xx --- Segment error
- F6xx --- Control instruction or macro error
- W7xx --- Any type of warning error

11.2 Assembler Error Messages

Table 11-1. Assembler Error Messages (1/12)

A101	Message	Source file size 0 'file name'
	Cause	A source module with file size 0 has been input.
A102	Message	Illegal processor type specified
	Cause	A mistake was made in the specification of the target device.
A103	Message	Syntax error in module header
	Cause	A mistake was made in description format for a control instruction that can be described in a source module header.
A104	Message	Can't use this control outside module header
	Cause	A control instruction for description in a source module header is described in an ordinary source.
A105	Message	Duplicate PROCESSOR control
	Cause	A PROCESSOR control instruction is described more than once in a source module header.
A106	Message	Illegal source file name for module name
	Cause	Module name cannot be created because the primary name for the source file name has a character that is not a legal symbol structure character.
A107	Message	Default segment ?CSEG is already used
	Cause	Attempted to define an undefined segment with a default segment.
A108	Message	Symbol table overflow 'symbol name'
	Cause	The number of symbols exceeds the limit (2700 symbols) that can be defined.
A109	Message	Too many DS
	Cause	Too many gaps have opened between object codes in a segment because too many DS directives are used, so data cannot be output to the object file.
A110	Message	String table overflow
	Cause	Limits of the string table are exceeded.
	Action by user	Reduce number of symbols to 9 characters or less.
A111	Message	Object code more than 128 bytes
	Cause	Object code exceeds 128 bytes per line in a source statement.
A112	Message	No processor specified
	Cause	Target device is not specified in the command line or in the source module file.

Table 11-1. Assembler Error Messages (2/12)

F201	Message	Syntax error
	Cause	An incorrect statement description format was used.
F202	Message	Illegal operand
	Cause	The described operand is illegal.
F203	Message	Illegal register
	Cause	A register that cannot be described was specified.
F204	Message	Illegal character
	Cause	An illegal character is described in the source module.
F205	Message	Unexpected LF in string
	Cause	A carriage return code appears in a character string before the string is closed.
F206	Message	Unexpected EOF in string
	Cause	An end-of-file code appears in a character string before the string is closed.
F207	Message	Unexpected null code in string
	Cause	A null code (00H) is described in a character string.

Table 11-1. Assembler Error Messages (3/12)

F301	Message	Too complex expression
	Cause	Expression is too complex.
F302	Message	Absolute expression expected
	Cause	A relocatable expression is described.
F303	Message	Illegal expression
	Cause	Incorrect description format for expression is used.
F304	Message	Illegal symbol in expression 'symbol name'
	Cause	An unusable symbol is described in an expression.
F305	Message	Too long string constant
	Cause	Limit on string constant length (2 characters) is exceeded.
F306	Message	Illegal number
	Cause	Incorrect numerical value is described.
F307	Message	Division by zero
	Cause	A value is divided by zero.
F308	Message	Too large integer
	Cause	The value of a constant exceeds 16 bits or 32 bits.
F309	Message	Illegal bit value
	Cause	Incorrect bit value is described.
F310	Message	Bit value out of range
	Cause	Bit value exceeds the range 0 to 7 or 0 to 15.
F311	Message	Operand out of range (n)
	Cause	Specified value exceeds the range n (0 to 7).
F312	Message	Operand out of range (byte)
	Cause	Value of an operand exceeds the range (00H to FFH), or the value of the byte in an operand is outside the range (-128 to +127).
F313	Message	Operand out of range (addr5)
	Cause	Operand is outside the describable range (40H to 7EH or 8040H to 807EH) for addr5.
F314	Message	Operand out of range (addr11)
	Cause	Operand is outside the describable range (800H to FFFH) for addr11.
F315	Message	Operand out of range (saddr)
	Cause	Operand is outside the describable range (0FE20H to 0FF1FH) for saddr.

Table 11-1. Assembler Error Messages (4/12)

F316	Message	Operand out of range (addr16)
	Cause	Operand is outside the describable range (varies according to target device) for addr16.
F317	Message	Even expression expected
	Cause	Odd-number address is described for word access.
F318	Message	Operand out of range (sfr)
	Cause	The description range (0FF00H to 0FFFFH) of the operand of the SFR/SFRP directive is exceeded, or an odd number is specified as the operand of the SFRP directive.
F326	Message	Illegal SFR access in operand
	Cause	An SFR symbol which cannot be accessed is described.

Table 11-1. Assembler Error Messages (5/12)

F401	Message	Illegal symbol for PUBLIC 'symbol name'
	Cause	This symbol cannot be declared PUBLIC.
F402	Message	Illegal symbol for EXTRN/EXTBIT 'symbol name'
	Cause	This symbol cannot be declared EXTRN/EXTBIT.
F403	Message	Can't define PUBLIC symbol 'symbol name'
	Cause	This symbol already has a PUBLIC declaration and cannot be defined with a PUBLIC declaration.
	Action by user	Because the symbol defining a bit parameter other than saddr.bit, SET symbol, a symbol already declared to be externally referenced, segment name, module name, macro name, BSFR/WSFR symbol (user-defined symbol), and EQUD symbol cannot be declared as PUBLIC, either cancel the PUBLIC declaration, or change the EQU definition.
F404	Message	Public symbol is undefined 'symbol name'
	Cause	A symbol with a PUBLIC declaration is undefined.
F405	Message	Illegal bit symbol
	Cause	An illegal symbol is used as a forward-reference symbol or bit symbol for the bit symbol of an operand in a machine-language instruction.
	Action by user	Describe backward reference or EXTBIT declaration for the bit symbol.
F406	Message	Can't refer to forward bit symbol 'symbol name'
	Cause	Description refers forward to a bit symbol or refers to a bit symbol in an expression.
F407	Message	Undefined symbol reference 'symbol name'
	Cause	An undefined symbol is used.
F408	Message	Multiple symbol definition 'symbol name'
	Cause	Symbol name is defined more than once.
F409	Message	Too many symbols in operand
	Cause	The number of symbols described in an operand exceeds the number that can be described in 1 line.
F410	Message	Phase error
	Cause	The value of the symbol changed during assemble (for example, an EQU symbol label changed by optimum processing of BR directive is defined in an operand).

Table 11-1. Assembler Error Messages (6/12)

F501	Message	Too many default ORG segment
	Cause	The number of ORG directives without segment name specification exceeds the limit (20 directives in one module).
F502	Message	Illegal segment name
	Cause	Symbol is described with an illegal segment name.
F503	Message	Different segment type 'segment name'
	Cause	Two or more segments are defined with the same name but types are different.
F504	Message	Too many segment
	Cause	Number of segments defined exceeds limit (100).
F505	Message	Current segment is not exist
	Cause	The ENDS directive is described before a segment is created or before the next segment is created after a segment has been once completed.
F506	Message	Can't describe DB, DW, DS, ORG, label in BSEG
	Cause	DB, DW, DS, ORG directives are defined in a bit segment.
F507	Message	Can't describe opcodes (,RSS) outside CSEG
	Cause	Machine language instruction or RSS directive is described in something other than a code segment.
F508	Message	Can't describe DBIT outside BSEG
	Cause	DBIT directive is described in something other than a bit segment.
F509	Message	Illegal address specified
	Cause	An address allocated to an absolute segment is outside the range for that segment.
F510	Message	Location counter overflow
	Cause	Location counter is outside the range for a segment.
F511	Message	Segment name expected
	Cause	Segment name is not specified for segment definition directive for reallocation attribute is AT.
F512	Message	Segment size is odd numbers 'segment name'
	Cause	Size of reallocation attribute callt0 or callt1 segment is described in an odd number.

Table 11-1. Assembler Error Messages (7/12)

F601	Message	Nesting over include
	Cause	Nesting of include file exceeds limit (2 levels).
F602	Message	Must be specified switches
	Cause	Switch name not specified.
F603	Message	Too many switches described
	Cause	Switch name description exceeds limit (5 per module).
F604	Message	Nesting over of IF-classes
	Cause	Nesting of IF/_IF clauses exceeds limit (8 levels).
F605	Message	Needless ELSE statement exists
	Cause	An ELSE statement exists where it is not necessary.
F606	Message	Needless ENDIF statement exists
	Cause	An ENDIF statement exists where it is not necessary.
F607	Message	Missing ELSE or ENDIF
	Cause	An ELSE or ENDIF statement required by IF/_IF clause is missing.
F608	Message	Missing ENDIF
	Cause	An ENDIF statement required by IF/_IF clause is missing.
F609	Message	Illegal ELSEIF statement
	Cause	An ELSEIF or _ELSEIF statement is described after an ELSE statement.
F610	Message	Multiple symbol definition (MACRO) 'symbol name'
	Cause	Symbol used to define a macro name is already defined.
F611	Message	Illegal syntax of parameter
	Cause	Formal parameter of a macro is incorrect.
F612	Message	Too many parameter
	Cause	Number of formal parameters for a macro definition exceeds limit (16).
F613	Message	Same nameparameter described 'symbol name'
	Cause	Symbol is specified with same name as a formal parameter for a macro definition.

Table 11-1. Assembler Error Messages (8/12)

F614	Message	Can't nest macro definition
	Cause	Macro definition cannot be nested in another macro definition.
F615	Message	Illegal syntax of local symbol
	Cause	Description of operand in a LOCAL directive is incorrect.
F616	Message	Too many local symbols
	Cause	Number of local symbols that can be described in 1 macro body (64) is exceeded.
F617	Message	Missing ENDM
	Cause	ENDM statement required by macro definition directive is missing.
F618	Message	Illegal syntax of ENDM
	Cause	ENDM statement description is incorrect.
F619	Message	Illegal definition macro
	Cause	Referenced macro is incorrectly defined.
F620	Message	Illegal syntax of actual parameter
	Cause	Description of actual parameter of macro is incorrect.
F621	Message	Nesting over of macro reference
	Cause	The limit on nesting in a macro reference (8 levels) is exceeded.
F622	Message	Illegal syntax of EXITM
	Cause	EXITM statement is incorrect.
F623	Message	Illegal operand of REPT
	Cause	An unpermitted expression is described in the operand of a REPT directive.
F624	Message	More than ??RAFFFF
	Cause	More than 65535 local symbols are replaced during macro development.
F625	Message	Unexpected ENDM
	Cause	An unexpected ENDM is found.
F626	Message	Can't describe LOCAL outside macro definition
	Cause	LOCAL directive is described in a normal source statement other than a macro body.
F627	Message	More than two segments in this include / macro
	Cause	2 or more segments are found in an include file, macro body, rept-endm block, or irp-endm block.
F628	Message	Illegal REPT/IRP block
	Cause	<1> REPT/IRP is described without segment definition, or none of the CSEG directive, an instruction that creates a label or object, and the DS directive is described in the REPT/IRP block. <2> REPT/IRP is described without segment definition, or the DSEG/BSEG directive is described in the REPT/IRP block.

Table 11-1. Assembler Error Messages (9/12)

W701	Message	Too long source line
	Cause	Over 218 characters are described on 1 line of a source statement.
	Program processing	219th and subsequent characters are ignored.
W702	Message	Duplicate PROCESSOR option and control
	Cause	Command-line specification option for target device (-C) and PROCESSOR directive in source header are both specified.
	Program processing	Command-line specification option for target device (-C) is available, and PROCESSOR directive in source header is ignored.
W703	Message	Multiple defined module name
	Cause	NAME directive is defined 2 or more times.
	Program processing	NAME directive is unavailable and the already defined module name is available.
W704	Message	Already declared EXTRN symbol 'symbol name'
	Cause	This symbol is already declared EXTRN.
	Action by user	Specify EXTRN declaration once in 1 module.
W705	Message	Already declared EXTBIT symbol 'symbol name'
	Cause	This symbol is already declared EXTBIT.
	Action by user	Specify EXTBIT declaration once in 1 module.
W706	Message	Missing END statement
	Cause	END statement is not described at end of source file.
	Program processing	Assumes that END statement is described at end of source file.

Table 11-1. Assembler Error Messages (10/12)

W707	Message	Illegal statement after END directive
	Cause	Item other than comment, space, tab, or CR code is described after END statement.
	Program processing	Ignores everything after END statement.
W708	Message	Already declared LOCAL symbol 'symbol name'
	Cause	This symbol is already declared LOCAL.
	Action by user	Declare 1 symbol LOCAL only once per macro.
W709	Message	Few count of actual parameter
	Cause	Fewer actual parameters are set than formal parameters.
	Program processing	Formal parameters are handled as null strings where actual parameters are insufficient.
W710	Message	Over count of actual parameter
	Cause	More actual parameters are set than formal parameters.
	Program processing	Surplus actual parameters are ignored.
W711	Message	Too many errors to report
	Cause	Too many errors exist to report in a single line (i.e. 6 or more errors)
	Program processing	6th and subsequent error messages are not output but processing continues.
W712	Message	Insufficient cross-reference work area
	Cause	Memory is insufficient to process output of cross-reference list.
	Program processing	Cross-reference list is not output but processing continues.
F801	Message	Illegal Debug Information
	Cause	Macro reference or include file speccification exceeds the limit.

Table 11-1. Assembler Error Messages (11/12)

A901	Message	Can't open source file 'file name'
	Cause	Source file cannot be opened.
A902	Message	Can't open parameter file 'file name'
	Cause	Parameter file cannot be opened.
A903	Message	Can't open include file 'file name'
	Cause	Include file cannot be opened.
A904	Message	Illegal include file 'file name'
	Cause	A drive name only, path name only or a device-type file name is specified as an include file name.
A905	Message	Can't open overlay file 'file name'
	Cause	Overlay file cannot be opened.
	Action by user	Make sure the overlay file is in the same directory as the assembler execution format.
A906	Message	Illegal overlay file 'file name'
	Cause	Contents of overlay file are illegal.
A907	Message	Can't open object file 'file name'
	Cause	Object file cannot be opened.
	Action by user	Use a disk with an open area in its directory.
A908	Message	Can't open print file 'file name'
	Cause	Assemble list file cannot be opened.
	Action by user	Use a disk with an open area in its directory.
A909	Message	Can't open error list file 'file name'
	Cause	Error list file cannot be opened.
	Action by user	Use a disk with an open area in its directory.
A910	Message	Can't open temporary file 'file name'
	Cause	Temporary file cannot be opened.
	Action by user	Use a disk with an open area in its directory.
A911	Message	System error
	Cause	A system error has occurred.
A912	Message	Can't set Control-C
	Cause	CTRL+C key cannot be input because assemble execution has been stopped.
A913	Message	Can't read source file 'file name'
	Cause	A file input/output error has occurred in the source file.

Table 11-1. Assembler Error Messages (12/12)

A914	Message	Can't read parameter file 'file name'
	Cause	A file input/output error has occurred in the parameter file.
A915	Message	Can't read include file 'file name'
	Cause	A file input/output error has occurred in the include file.
A916	Message	Can't read overlay file 'file name'
	Cause	A file input/output error has occurred in the overlay file.
A917	Message	Can't write object file 'file name'
	Cause	A file input/output error has occurred in the object file.
	Action by user	Output object file to another directory or create an open area in the specified disk.
A918	Message	Can't write print file 'file name'
	Cause	A file input/output error has occurred in the assemble list file.
	Action by user	Output assemble list file to another directory or create an open area in the specified disk.
A919	Message	Can't write error list file 'file name'
	Cause	A file input/output error has occurred in the error list file.
	Action by user	Output error list file to another directory or create an open area in the specified disk.
A920	Message	Can't read / write temporary file 'file name'
	Cause	A file input/output error has occurred in the temporary file.
	Action by user	Output temporary file to another directory or create an open area in the specified disk.
A921	Message	Assembler internal error
	Cause	An assembler-internal error has occurred.
	Action by user	Execute assemble again.
A922	Message	Insufficient memory in hostmachine
	Cause	System does not have sufficient memory to execute assembler.
A923	Message	Insufficient memory for macro in hostmachine
	Cause	Memory for macro became insufficient in the middle of macro processing.
	Action by user	Reduce number of macros defined.

11.3 Linker Error Messages

Table 11-2. Linker Error Messages (1/8)

A101	Message	'File name' invalid input file (or made by different hostmachine)
	Cause	File other than object module file was input, or link was attempted with object module file created on an incompatible host machine.
F102	Message	Directive syntax error
	Cause	Description of directive is incorrect.
A103	Message	'File name' Illegal processor type
	Cause	Target device of assemble or compile is not a target device of this linker.
	Action by user	Check to ensure that the object module file is correct. Check to ensure that the target device for the assemble or compile can be handled by the linker. Also check that the overlay file is the correct version. (The linker references part of the overlay file of the assembler to obtain characteristic data on the target device.)
A104	Message	'File name' Different processor type from first input file 'first input file name'
	Cause	An object module file is input whose target device is different from that of the first input object module file.
W105	Message	Library file 'file name' has no public symbol
	Cause	Library file has no public symbol. Therefore, an object module included in the library file cannot be linked. (If the option –w is 1 or move, the message is displayed.)
A106	Message	Can't create temporary file 'file name'
	Cause	Cannot create temporary file.

Table 11-2. Linker Error Messages (2/8)

F107	Message	Name 'name' in directive already defined
	Cause	Attempted to define a reserved word or a previously defined name as the memory area of a directive. This name (reserved word, memory space name, memory area name) is already defined.
F108	Message	Overlapped memory area 'Memory area 1' and 'Memory area 2'
	Cause	The memory area addresses defined in the memory directive are overlapped.
F109	Message	Memory area 'Memory area name' too long name (up to 31 characters)
	Cause	The memory area name specified in the directive is too long. The memory area name specified in the directive is 32 characters or longer.
F110	Message	Memory area 'Memory area name' already defined
	Cause	The memory area specified in the memory directive is already registered.
F111	Message	Memory area 'Memory area name' redefinition out of range
	Cause	The range of the memory area specified in the memory directive is outside the redefinable range.
F112	Message	Segment 'segment name' wrong allocation type
	Cause	Wrong allocation type is specified for the segment in the merge directive.
A113	Message	Linker internal error
	Cause	Internal error in the linker
	Action by user	Contact an authorized representative or NEC.
F114	Message	Illegal number
	Cause	Description of a numerical value in a directive is incorrect.
F115	Message	Too large value (up to 65535/0FFFFH)
	Cause	A value greater than 65535 (0FFFFH) is described in the directive.
F116	Message	Memory area 'Memory area name' definition out of range
	Cause	The sum of the start address and size of the memory area specified in the memory directive exceeds 65535 (0FFFFH).
F117	Message	Too many line number data in 'file name'
	Cause	Input line number data (debugging data) again and continue processing. An object file will only be output if option -J is specified.

Table 11-2. Linker Error Messages (3/8)

F201	Message	Multiple segment definition 'segment name' in merge directive
	Cause	Segment specified in the merge directive is already registered (the same segment is attempted to specify allocation using multiple merge directives).
F202	Message	Segment type mismatch 'segment 1' in file 'segment 2' -ignored
	Cause	A segment with the same name as this segment but having the reallocation attributes of a different segment type is found.
A203	Message	Segment 'segment name' unknown segment type
	Cause	An error exists in the segment data of the input object module file (specification of link of output segments is incorrect).
F204	Message	Memory area/space 'name' not defined
	Cause	Memory area/space name specified in merge directive is not defined.
F205	Message	Name 'name' in directive has bad attribute
	Cause	An item that cannot be described in a segment name, memory area name or memory space name is described in the directive (for example, a memory space name is described where a memory area name is required).
F206	Message	Segment 'segment name' can't allocate to memory - ignored
	Cause	Segment cannot be allocated to memory (not enough memory area exists to allocate segment).
F207	Message	Segment 'segment name' has illegal segment type
	Cause	This segment type data is illegal.
F208	Message	Segment 'segment name' may not change attribute
	Cause	Attempted to change the link type in the directive for a segment created with the reallocation attribute 'AT xxxxH' specified during assemble, or created using the ORG directive.
F209	Message	Segment 'segment name' may not change arrangement
	Cause	Attempted to change the allocation address in the directive for a segment created with the reallocation attribute 'AT xxxxH' specified during assemble, or created using the ORG directive.
	Action by user	Do not specify the allocation address in the assembler for a segment whose link type is to be specified during link.
F210	Message	Segment 'segment name' does not exist - ignored
	Cause	Segment specified in the directive does not exist.

Table 11-2. Linker Error Messages (4/8)

F301	Message	Relocatable object code address out of range (file 'file name', segment 'segment name', address xxxxH, type 'addressing type')
	Cause	Correction data of relocatable object code included in the input object module file is output to an address where no object code exists (relocation entry address is out of range of origin data).
	Action by user	Check that symbol reference is correct.
F302	Message	Illegal symbol index in line number (file 'file name', segment 'segment name')
	Cause	Line number data for debugging included in the input object module file is incorrect, and does not correctly reference the symbol data. Line number index and symbol index do not correspond.
F303	Message	Can't find symbol index in relocatable object code (file 'file name', segment 'segment name', address xxxxH, type 'addressing type')
	Cause	Correction data of relocatable code included in the input object module file is incorrect, and does not correctly reference the symbol data. Relocation entry and symbol index do not correspond.
	Action by user	Check that reference method of symbols and variables is correct.
F304	Message	Operand out of range (segment 'segment name', address xxxxH, type 'addressing type')
	Cause	Operand value used in decision of relocatable object code is out of range for operand values corresponding to the instruction.
	Action by user	Describe the value for the operand in the source program that fits within the range determined for each addressing type.
F305	Message	Even value expected (segment 'segment name', address xxxxH, type 'addressing type')
	Cause	The operand value used to determine the callt or saddrp addressing relocatable object code is an odd number (callt and saddrp addressing operands must be even numbers).

Caution The address shown in 'address xxxxH' in the messages in F301 to F305 are absolute addresses after segment allocation.

Table 11-2. Linker Error Messages (5/8)

A401	Message	'File name' Bad symbol table
	Cause	Symbol data of input object module file is illegal. Symbol entry of input file does not begin with '.file'.
A402	Message	File 'file name' has no string table for symbol
	Cause	Symbol data of input object module file is illegal.
	Action by user	Perform assemble or compile again. This may be avoidable by making the recognizable number of characters 8 for the assembler and 7 for the compiler.
A403	Message	Symbol 'symbol name' unmatched type in file 'file-name1' First defined in file 'file-name2'
	Cause	Externally defined/referenced symbol type with same name is different in file 1 and file 2.
F404	Message	Multiple Symbol definition 'symbol name' in file 'file-name1' First defined in file 'file-name2'
	Cause	Public symbol defined in object module file 1 is already declared PUBLIC in object module file 2.

Table 11-2. Linker Error Messages (6/8)

F405	Message	Undefined symbol 'symbol-name' in file 'file-name'
	Cause	Symbol declared EXTRN in the file is not declared PUBLIC in another file.
W406	Message	Stack area less than 10 bytes
	Cause	Size of protected stack area is 10 bytes or less (size of stack area protected in memory area specified with -S option is 10 bytes or less). (Displayed if -W option is 1 or more.)
W407	Message	Can't allocate stack area
	Cause	No free area is available in memory area in which stack area is protected (stack area cannot be protected in memory area specified with -S option).
W411	Message	Different REL type in file 'file name'
	Cause	The version of the type of OMF differs (displayed if the -W option is 2 or more).
F412	Message	Multiple CHGSFR in file 'file name' First defined in file 'file name'
	Cause	CHGSFR specification made for all input OMF differs.
F413	Message	Multiple LOCATION in file 'file name' First defined in file 'file name'
	Cause	This is output if 2 or more LOCATION instructions for all input OMF are found.
F414	Message	'LOCATION' operation not found in all modules
	Cause	This is output if no LOCATION instructions for all input OMF are found.
F415	Message	-QD/QF/etc. and Not -QD/QF/etc. REL are mixed
	Cause	The compile option for all input OMF, except CC_DC, does not match.

Table 11-2. Linker Error Messages (7/8)

W416	Message	Multiple CAP/NOCAP are in file 'file-name (option)' First defined in file 'file-name (option)'
	Cause	CAP/NOCAP assemble or compile options are not identical for all input OMF. (Displayed if -W option is 2 or more.)
W417	Message	The version of tool name in file 'file-name' are more than one Used the first one in file 'file-name'
	Cause	A discrepancy exists between each tool (CC78K3, ST78K3, RA78K3) used until the link stage for all input OMF and the DF version. (Displayed if –W option is 2 or more.)
W418	Message	File 'file name' is old. Can't find TOOL infomation
	Cause	This is output when TOOL information is not found in input OMF. Normally, this is always output when link is performed with an old (DF-incompatible) object module file. (Display is –W option is 2 or more.)
F420	Message	File 'file name' has already had error(s)/warning(s) by 'tool name'
	Cause	An error message or warning message for each tool (CC78K3, ST78K3, RA78K3) used until the link stage is output.

A501	Message	Insufficient memory in hostmachine
	Cause	The system does not have sufficient memory to operate the program.

Table 11-2. Linker Error Messages (8/8)

A901	Message	Can't open overlay file 'file name'
	Cause	Overlay file cannot be opened.
	Action by user	Make sure the overlay file is in the correct directory (a directory containing an execution program).
A902	Message	File 'file name' not found
	Cause	The specified library file cannot be opened.
A903	Message	Can't read input file 'file name'
	Cause	Object module file specified as an input file cannot be read.
A904	Message	Can't open output file 'file name'
	Cause	Output file cannot be opened.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to create output file.
A905	Message	Can't create temporary file 'file name'
	Cause	Temporary file for symbol entry cannot be created.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to attempt to create temporary file.
A906	Message	Can't write map file 'file name'
	Cause	Data cannot be written to the link list file.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to attempt to create link list file.
A907	Message	Can't write output file 'file name'
	Cause	Data cannot be written to the load module file.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to attempt to create output file.
A908	Message	Can't access temporary file 'file name'
	Cause	Temporary file cannot be written.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to attempt to create temporary file.
A909	Message	Can't read device file 'device file name'
	Cause	DF cannot be read because no TOOL information exists for all input OMF.

11.4 Object Converter Error Messages

Table 11-3. Object Converter Error Messages (1/2)

A100	Message	'File name' Illegal processor type
	Cause	Target device of the assembler or compiler is different from the target device of this program.
	Action by user	Check whether the load module file is correct and check target device of the assemble or compile. Also, check whether the version of the device file is correct.
A101	Message	'File name' invalid input file (or made by different hostmachine)
	Cause	Attempted to input a file other than a load module file, or to convert a load module file created on an incompatible host machine.
A103	Message	Symbol 'symbol name' Illegal attribute
	Cause	A mistake exists in the symbol attribute of the input file.
A104	Message	'File name' Illegal input file - not linked
	Cause	Attempted to input an object module file.
A105	Message	Insufficient memory in hostmachine
	Cause	Memory is not sufficient to operate the program.
A106	Message	Illegal symbol table
	Cause	A mistake exists in the symbol table of the input load module file.

F200	Message	Undefined symbol 'symbol name'
	Cause	A symbol whose address is undetermined has been found.
	Action by user	Define the symbol's value. This symbol is referenced as an external reference symbol. If it is not externally defined, specify an external definition outside the module in which the value of the symbol is defined.
F201	Message	Out of address range
	Cause	The address of an object in a load module file is out of range.

Table 11-3. Object Converter Error Messages (2/2)

W300	Message	xxxxH - yyyyH overlapped
	Cause	Objects overlapped in the address from xxxxH to yyyyH are output.

A900	Message	Can't open file 'file name'
	Cause	File cannot be opened.
A901	Message	Can't close file 'file name'
	Cause	File cannot be closed.
A902	Message	Can't read file 'file name'
	Cause	File cannot be correctly read.
A903	Message	Can't access file 'file name'
	Cause	File cannot be correctly read or written to.
A904	Message	Can't write file 'file name'
	Cause	Data cannot be correctly written to an output file.

11.5 Librarian Error Messages

Table 11-4. Librarian Error Messages (1/4)

A001	Message	Missing input file
	Cause	Only options are specified. No input files are specified.
A002	Message	Too many input files
	Cause	Total number of input files exceeds the limit.
A003	Message	Unrecognized string '???'
	Cause	Something other than an option is specified on a conversational-format command line.
A004	Message	Illegal file name 'file name'
	Cause	File name includes character(s) not permitted by the operating system, or exceeds the limit for number of characters.
A005	Message	Illegal file specification 'file name'
	Cause	An illegal item is specified in the file name.
A006	Message	File not found 'file name'
	Cause	Specified input file does not exist.
A007	Message	Input file specification overlapped 'file name'
	Cause	Input file name specification is overlapped.
A008	Message	File specification conflicted 'file name'
	Cause	Input or output file name specifications overlap.
A009	Message	Unable to make file 'file name'
	Cause	Specified output file cannot be created.
A010	Message	Directory not found 'file name'
	Cause	A drive or directory which does not exist is included in the output file name.
A011	Message	Illegal path 'file name'
	Cause	An item other than a path name is specified in an option specifying the path name for a parameter.
A012	Message	Missing parameter 'option'
	Cause	Required parameter is not specified.
A013	Message	Parameter not needed 'option'
	Cause	An unnecessary parameter is specified.

Table 11-4. Librarian Error Messages (2/4)

A014	Message	Out of range 'option'
	Cause	Specified value is out of range.
A015	Message	Parameter is too long 'option'
	Cause	Number of characters specified in parameter exceeds limit.
A016	Message	Illegal parameter 'option'
	Cause	A mistake exists in the syntax of the parameter.
A017	Message	Too many parameters 'option'
	Cause	Total number of parameters exceeds limit.
A018	Message	Option is not recognized 'option'
	Cause	An incorrect option is specified.
A019	Message	Parameter file nested
	Cause	-F option is specified in a parameter.
A020	Message	Parameter file read error 'file name'
	Cause	An error occurred in reading a parameter file.
A021	Message	Memory allocation failed
	Cause	An error occurred in memory allocation.

Table 11-4. Librarian Error Messages (3/4)

A100	Message	Internal error
	Cause	An internal error has occurred.
F101	Message	Invalid sub command
	Cause	Subcommand name is incorrect.
F102	Message	Invalid syntax
	Cause	Parameter specification in subcommand is incorrect.
F103	Message	Illegal input file - different target chip (file: file name)
	Cause	Specification of target device in input object module file is incorrect.
F104	Message	Illegal library file - different target chip (file: file name)
	Cause	Specification of target device in library file is incorrect.
F105	Message	Module not found (module: file name)
	Cause	Specified module does not exist in library file.
F106	Message	Module already exists (module: file name)
	Cause	A module of the same name already exists in the updated library file or another input file.
F107	Message	Master library file is not specify
	Cause	Updated library file is not specified in a previous operation, but the library file name is replaced with '. '.
F108	Message	Multiple transaction file (file: file name)
	Cause	Input object module files overlap.
F109	Message	Public symbol already exists (symol: symbol name)
	Cause	An externally defined symbol name already exists in an updated library file or other input file.
F110	Message	File specification conflicted (file: file name)
	Cause	Specified input file name is same as output file name.
F111	Message	Illegal file format (file: file name)
	Cause	Format of an updated library file or other input file is incorrect.
F112	Message	Library file not found (file: file name)
	Cause	Specified library file is not found.

Table 11-4. Librarian Error Messages (4/4)

F113	Message	Object module file not found (file: file name)
	Cause	Specified object module file is not found.
F114	Message	No free space for temporary file
	Cause	Sufficient space does not exist in the disk to create a temporary file.
F115	Message	Not enough memory
	Cause	Sufficient memory is not available to operate the program.
F116	Message	Sub command Buffer full
	Cause	Limit for continuous line length in a subcommand (128 x 15 characters) is exceeded. Limit for length of 1 line in a subcommand (128 characters) is exceeded.

A901	Message	File open error (file: file name)
	Cause	An error exists in the file, or the system is not operating properly.
F902	Message	File read error (file: file name)
	Cause	An error exists in the file, or the system is not operating properly.
A903	Message	File write error (file: file name)
	Cause	An error exists in the file, or the system is not operating properly.
A904	Message	File seek error (file: file name)
	Cause	An error exists in the file, or the system is not operating properly.
A905	Message	File close error (file: file name)
	Cause	An error exists in the file, or the system is not operating properly.

11.6 List Converter Error Messages

Table 11-5. List Converter Error Messages (1/2)

A101	Message	File is not 78K/III 'file name'
	Cause	Input file name is not a 78K/III file name.
W101	Message	Load module file is older than object module file 'load module file name, object module file name'
	Cause	A load module file is specified which is older than the object module file.
A102	Message	Load module file is not executable 'file name'
	Cause	Attempted to input a file other than a load module file, or attempted to convert a load module file created on an incompatible host machine.
W102	Message	Load module file is older than assemble module file 'load module file name, assemble list file name'
	Cause	A load module file is specified which is older than the assemble list file.
A103	Message	Load module file has relocation data 'file name'
	Cause	Address of load module file is not determined.
W103	Message	Assemble list has error statement 'file name'
	Cause	An error exists in the assemble list.
A104	Message	Object module file is executable 'file name'
	Cause	Object module file is in an executable format.
W104	Message	Segment name is not found in assemble list file 'segment name'
	Cause	Segment name of object module file is not found in assemble list.
A105	Message	Segment name is not found in load list file 'segment name'
	Cause	Segment name of object module file is not found in load module file.

Table 11-5. List Converter Error Messages (2/2)

W105	Message	Segment data length is different 'segment name'
	Cause	Length of segment data in assemble list file is different from length of segment data in object module file.
	Program processing	Surplus segment data is ignored and processing continues.
A106	Message	Segment name is not found in object module file 'file name'
	Cause	Segment name of assemble list file is not found in object module file.
A107	Message	Not enough memory
	Cause	Memory is not sufficient for program operation.
A108	Message	Load module file has no symbol data 'load module name'
	Cause	Option -NG is specified in linker, so symbol data in load module file cannot be output.

A901	Message	File open error has occurred 'file name'
	Cause	File cannot be opened.
A902	Message	File read error has occurred 'file name'
	Cause	File cannot be correctly read.
A903	Message	File write error has occurred 'file name'
	Cause	Data cannot be correctly written to file.
A904	Message	File seek error has occurred 'file name'
	Cause	File seek error has occurred.
A999	Message	Internal error
	Cause	Program-internal error

APPENDIX A SAMPLE PROGRAMS

The following is an introduction to the sample lists of each program used in the RA78K3.

- Source lists
- Execution example
- Output lists

  Assemble lists

  Symbol lists

  Cross-reference lists

  Map list

  Public symbol lists

  Local symbol lists

  Library data output lists

  Absolute assemble lists

A.1 Source Lists

(1) 78K3MAIN.ASM

```
$       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

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

        MOV     HDTSA,#1AH
        MOVG    HL,#HDTSA               ;set hex 2-code data in HL registor

        CALL    CONVAH                  ;convert ASCII <- HEX
                                        ;output BC-register <- ASCII code
        MOVW    DE,#STASC               ;set DE <- store ASCII code table
        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

        CSEG
CONVAH: 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 Example

```
C>ra78k3 78k3main.asm -g -kx -lw90

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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


C>ra78k3 78k3sub.asm -g -kx -lw90

78K/III Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1 Start
Pass2 Start

 Target chip : uPDxxxxx
 Device file : Vx.xx

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


C>lk78k3 78k3main.rel 78k3sub.rel -g -o78k3.lnk -p78k3.map -kp -kl

78K/III Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

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

```
C>oc78k3 78k3.lnk

78K/III Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

 Target chip : uPDxxxxx
 Device file : Vx.xx

Object Conversion Complete,      0 error(s) and     0 warning(s) found.


C>lb78k3

78K/III Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx
*create 78k3.lib
*add 78k3.lst 78k3sub.rel
*exit


C>lcnv78k3 78k3main -l78k3.lnk

List Conversion Program for RA78K/III Vx.xx [xx xxx xx]
   Copyright (C) NEC Corporation 1989,19xx

Pass1: start...
Pass2: start...
Conversion complete.
```

A.3 Output Lists

A.3.1 Assemble lists

(1) 78K3MAIN.ASM assemble list

```
78K/III Series Assembler Vx.xx                          Date:xx xxx xxxx Page:    1


Command:  78k3main.asm -g -kx -lw90
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 0FFD20H
   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  OBFC80FE              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 2-code da
                                        ta in HL registor
   30    30
   31    31 0014  R280000               CALL    !CONVAH            ;convert ASCII <-
                                        HEX
   32    32                                                        ;output BC-registe
                                        r <- ASCII code
   33    33 0017  6521FE                MOVW    DE,#STASC          ;set DE <-store A
                                        SCII code table
   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
 Device file : Vx.xx
Assembly complete,     0 error(s) and    0 warning(s) found. (   0)
```

(2) 78K3SUB.ASM assemble list

```
78K/III Series Assembler Vx.xx                          Date:xx xxx xxxx Page:   1


Command:  78k3sub.asm -g -kx -lw90
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]            ;hex upper code load
   21    21 0004 R281300             CALL    !SASC
   22    22 0007  2431               MOV     B,A             ;store result
   23    23
   24    24 0009  B900               MOV     A,#0
   25    25 000B  059F               ROL4    [HL]            ;hex lower code load
   26    26 000D R281300             CALL    !SASC
   27    27 0010  2421               MOV     C,A             ;store result
   28    28
   29    29 0012  56                 RET
   30    30
   31    31                  ;*************************************************
   32    32                  ;* subroutine   convert ASCII code                *
   33    33                  ;*      input   Acc(lower 4 bits) <- hex code     *
   34    34                  ;*      output  Acc               <- ASCII code   *
   35    35                  ;*************************************************
   36    36
   37    37 0013  AF0A       SASC:   CMP     A,#0AH          ;check hex code > 9
   38    38 0015  8302               BC      $SASC1
   39    39 0017  A807               ADD     A,#07H          ;bias(+7)
   40    40 0019  A830       SASC1:  ADD     A,#30H          ;bias(+30)
   41    41 001B  56                 RET
   42    42
   43    43                          END
```

```
Segment informations:

ADRS  LEN      NAME

0000  001CH    ?CSEG

 Target chip : uPDxxxxx
 Device file : Vx.xx
Assembly complete,     0 error(s) and    0 warning(s) found. (   0)
```

A.3.2 Symbol lists

(1) 78K3MAIN.ASM symbol list

```
          Symbol Table List

 VALUE   ATTR    RTYP   NAME                VALUE   ATTR   RTYP NAME

         CSEG           ?CSEG                       CSEG          CODE
----H            EXT    CONVAH                      DSEG          DATA
FF20H    ADDR           HDTSA                 0H    ADDR   PUB   MAIN
         MOD            SAMPM                 0H    ADDR   PUB   START
FE21H    ADDR           STASC
```

(2) 78K3SUB.ASM symbol list

```
          Symbol Table List

VALUE   ATTR    RTYP   NAME                VALUE   ATTR   RTYP NAME

         CSEG           ?CSEG                 0H    ADDR   PUB   CONVAH
         MOD            SAMPM                13H    ADDR         SASC
 19H     ADDR           SASC1
```

A.3.3 Cross-reference lists

(1) 78K3MAIN.ASM cross-reference list

```
          Cross-Reference List

NAME              VALUE    R ATTR    RTYP   SEGNAME   XREFS

?CSEG                        CSEG             ?CSEG       22#
CODE                         CSEG             CODE        19#
CONVAH            ----H      E              EXT           13@     31
DATA                         DSEG             DATA        15#
HDTSA             FE20H      ADDR             DATA        16#     28      29
MAIN                0H       ADDR     PUB     CODE        12@     20#
SAMPM                        MOD                           3#
START               0H     R ADDR     PUB     ?CSEG       12@     20      23#
STASC             FE21H      ADDR             DATA        17#     33
```

(2) 78K3SUB.ASM cross-reference list

```
          Cross-Reference List

NAME              VALUE    R ATTR    RTYP   SEGNAME   XREFS

?CSEG                        CSEG             ?CSEG       18#
CONVAH              0H     R ADDR     PUB     ?CSEG       16@     19#
SAMPS                        MOD                           3#
SASC               13H     R ADDR             ?CSEG       21      26      37#
SASC1              19H     R ADDR             ?CSEG       38      40#
```

A.3.4 Map list

```
78K/IV Series 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)
   3EH byte(s) real data
    23 symbol(s) defined


*** Memory map ***


  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=0000H    SIZE=FE00H
         OUTPUT   INPUT    INPUT      BASE          SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
         CODE                         0000H         0002H   CSEG AT
                  CODE     SAMPM      0000H         0002H
         ?CSEG                        0002H         003CH   CSEG
                  ?CSEG    SAMPM      0002H         0020H
                  ?CSEG    SAMPS      0022H         001CH
* gap *                               003EH         FDC2H

  MEMORY=RAM
  BASE ADDRESS=FE00H    SIZE=0200H
         OUTPUT   INPUT    INPUT      BASE          SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
* gap *                               FE00H         0020H
         DATA                         FE20H         0003H   DSEG AT
                  DATA     SAMPM      FE20H         0003H
* gap *                               FE23H         00DDH
* gap (Not Free Area) *               FF00H         0100H
```

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 data output list

```
78K/III Series 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
```

A.3.8 Absolute assemble lists

```
78K/III Series Assembler Vx.xx                          Date:xx xxx xxxx Page:   1


Command:  78k3main.asm -g -kx -lw90
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 0FF20H
   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 ----                          CSEG
   23    23 0002  2B4100          START:  MOV     RFM,#00
   24    24 0005  OBFC80FE                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          ;set hex 2-code da
                                 ta in HL registor
   30    30
   31    31 0016 R282200                  CALL    !CONVAH            ;convert ASCII <-
                                 HEX
   32    32                                                          ;output BC-registe
                                 r <- ASCII code
   33    33 0019  6521FE                  MOVW    DE,#STASC          ;set DE <- store A
                                 SCII code table
   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
 Device file : Vx.xx
Assembly complete,     0 error(s) and    0 warning(s) found. (   0)
```

[MEMO]

APPENDIX B LIST OF CAUTIONS ON USE

The following is a list of items to note carefully when using the RA78K3.

B.1 Cautions

B.1.1 Handling device file

Information dependent on each target device is separated from the RA78K3 Ver. 5.00 and is included in a device file[Note 1] separately available. The RA78K3 Ver. 5.00 or later, therefore, cannot be used unless a device file corresponding to the device used is purchased.

As for the subseries[Note 2] whose device file is not available, a device file is supplied with RA78K3 Ver. 5.00 or later. However, the supplied device file is for assembler and C compiler only (file with extension ".78k"). For each device file supplied, refer to **Appendix D Notes on Using Device File.**

Notes 1. Device files for the μPD78352A, 78356, 78366, 78366A, and 78372 Subseries are optional.

2. μPD78312, 78312A, 78322, 78328, and 78334 Subseries

B.1.2 Memory necessary for execution (with PC-9800 Series, IBM PC/AT, and compatible machine)

The minimum memory size necessary for executing the assembler is 400 KB. This memory size increases if macros are used, and assembly may not be performed with 400 KB only in some cases. Therefore, allocate as large a vacant area of the conventional memory as possible.

B.1.3 Notes on list converter

The default symbol name case specification option of the list converter is -CA. If -CA is not specified by the assembler, specify -NCA with the list converter.

B.1.4 Notes on debug option

When performing compiling or structured assembly by specifying debug data output by the C compiler or structured assembler preprocessor, and when assembling the output assembler source, do not specify the debug data output option. The option necessary during assembly is output in the assembler source as a control statement by the C compiler or structured assembler preprocessor.

B.1.5 Notes on C compiler

Several points must be noted when assembling the assembler source output by the C compiler and performing C source level debugging.

For details, refer to the document supplied with the C compiler package (Notes on Use).

B.1.6 Notes on using network

If a directory that creates a temporary file is placed in a file system shared on a network, an abnormal operation may take place because files conflict. Avoid this conflict by setting an option or environmental variable.

B.1.7 Notes on ordering ROM code

Be sure to make the following specification by using the object converter when ordering a ROM code.

- Specify the -U option and make sure that there is no vacant area in the internal ROM area.
- Specify the -R option and sort the HEX-format objects in the order of address.

B.2 Limitations

The RA78K3 has the following limitations.

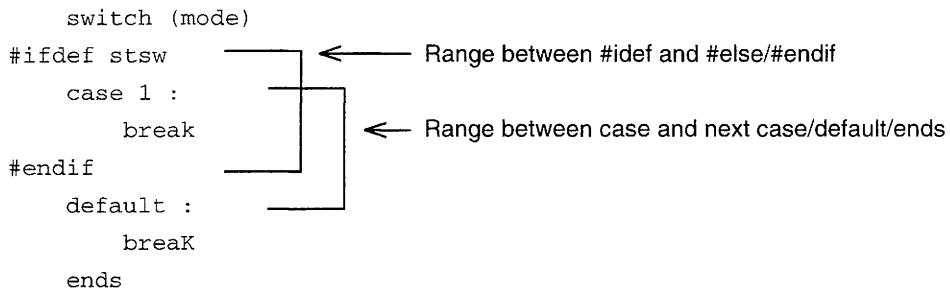
B.2.1 Limitations of structured assembler

(1) Control statements with crossing range causes an error.

[Description]

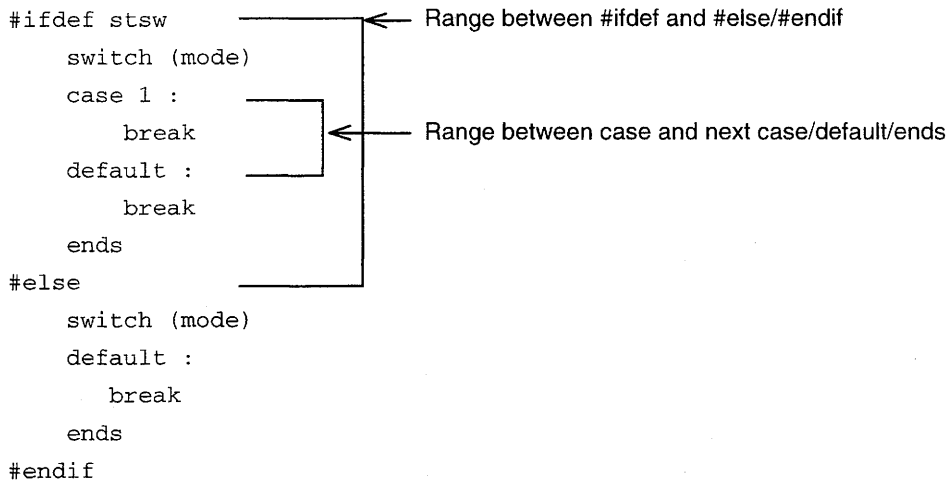
If a control statement is enclosed so that the statement is divided or crossed in between #ifdef and #endif, the control statement causes an error if #ifdef is true.

[Example]



[Preventive measures]

Nesting is correctly processed. Rewrite the source so that the range of the control statement is not crossed. The above example should be rewritten as follows:



B.2.2 Limitations of assembler

(1) When a label that is influenced by optimization is described in the portion of saddr when a bit symbol having the value of saddr.bit is defined as EQU, an error may be illegal.

[Description]

A label that is influenced by optimization is described in the portion of saddr when a bit symbol having the value of saddr.bit is defined as EQU. Therefore, an error may be illegal or the object code may be illegal in the following two cases:

- If saddr is 0FD20H and is outside an area in path 1 and inside an area in path 2, path 1 outputs an error to the EQU definition line, but path 2 does not output an error <illegal>.
- If saddr is 0FF1FH and is inside an area in path 1 and outside an area in path 2, path 1 does not output an error to the EQU definition line but path 2 outputs an error <normal>.

An error message (F410 Phase error) is output to the label that is defined after this EQU symbol has been referenced. If this label is referenced, the object is illegal.

[Preventive measures]

None

(2) The number of characters that can be written on one line is 256.

[Description]

If more than 256 characters are written on one line, an error message (W701 Too long source line) is output, and the 257th character and those that follow are assumed as the next line and assembled.

The 218th character from the beginning and those that follow are ignored. A 2-byte character such as Kanji is counted as 2 characters.

[Preventive measures]

None. Write 256 characters or less (including carriage return/line feed) on one line.

(3) If include files that ends in an incomplete form (carriage return/line feed is missing on the last line) are nested, an error may occur.

[Description]

If include files that ends in an incomplete form (carriage return/line feed is missing on the last line) are nested, one line in the include file of nest level 2 disappears, and the code is output to the next line. At this time, an error message (F201 Syntax error) is output.

[Example]

```
s1    equ    label1*2
s2    equ    label2*2
s3    equ    label3*2 [EOF]  ; No carriage return code before [EOF]
```

[Preventive measures]

None.

Be sure to end the last line of an include file with carriage return or line feed.

Input a carriage return code before EOF.

(4) The location counter cannot be referenced in a bit segment.

[Description]

The location counter ($) is meaningless in a bit segment.

If a symbol is defined by referencing the location counter in a bit segment, assembly and link is terminated without an error, but the object converter outputs an error message (A108 Symbol 'symbol name' illegal attribute) and is aborted.

[Example]

```
bs    bseg    at 0fe20H
b1    dbit
ba    equ     $
b2    dbit
bb    equ     $
b3    dbit
end
```

[Preventive measures]

None

(5) If an include file name is written in Kanji, the include file may not be correctly recognized.

[Description]

If an include file is written in Kanji, the include file may not be correctly recognized.

[Preventive measures]

None. Do not write an include file name in Kanji.

(6) If BSEG of the same segment name exists and if the BITPOS or MASK operator (with a name defined by BSEG (relocatable) specified as an operand) is specified, the object code of an instruction is illegal.

[Description]

If the name defined by BSEG (relocatable) is specified as the operand of the BITPOS or MASK operator, and if BSEG with the same segment name exists, the object code of the instruction for which the BITPOS or MASK operator is specified is illegal.

[Example]

```
----------test1.asm ----------
b1      bseg                    ; Segment of same name exists.
        dbit
lab     dbit
        dbit

        cseg
        mov     a,#bitpos lab   ; Object code is illegal.
        end

----------test2.asm ----------
b1      bseg                    ; Segment of same name exists.
        dbit
lab1    dbit
        dbit

        cseg
        mov     a,#bitpos lab1  ; Object code is illegal.
        end
```

[Preventive measures]

None

(7) (ampersand) in conditional assemble control instruction processing is not interpreted.

[Description]

"&" in IRP at the side of the processing that is made false by a conditional assemble control instruction is not interpreted as a character string coupling symbol "& (ampersand)", and an error message (F204 Illegal character) is output.

[Example]

```
$_IF (AAA = BBB)
IRP     ZZZ,<1,2,3>
LAB$ZZZ;        DW      0       ; An error (F204) occurs because of "&".
ENDM
$ENDIF
```

[Preventive measures]

Nest macros.

```
$_IF (AAA = BBB)
M1      MACRO para1             ; ←Addition
IRP     ZZZ,<1,2,3>
Para1:  DW    0                 ; ←Change
ENDM
ENDM                            ; ←Addition
M1      "LAB&ZZZ"               ; ←Addition
$ENDIF
```

(8) "^Z" is always necessary at the end of an include file.

[Description]

"^Z" is always necessary at the end of an include file.

[Preventive measures]

None. Make sure that an include file ends with a blank line.

(9) A hang-up occurs if the last line of an include file is macro reference and if "^Z" is missing before EOF.

[Description]

A hang-up occurs if the last line of an include file is macro reference and if "^Z" is missing before EOF.

[Preventive measures]

None. Make sure that an include file ends with a blank line.

(10) If a source file with a function of a long function name is compiled, and an assembler source is output and assembled when a compiler is used, an error may occur.

[Description]

When the compiler is used, and if a source including a function is compiled and an assembler source is output, the function name of $DGS or label with up to seven characters such as ?? are appended is generated. If the function name exceeds 24 characters as a result, the symbol name exceeds 31 characters, and an error occurs before assembly.

[Preventive measures]

None. Do not use a function name of more than 24 characters.

(11) When a compiler is used and if a C source with symbol definition in the #asm block is compiled and its assembler source is assembled, the debug data is illegal.

[Description]

When a compiler is used and a C source with symbol definition in the #asm block is compiled, and if the output assembler source is assembled, the debug information is illegal.

[Example]

```
int i;
void func (void)  {
   #asm
   syma:
                MOV A,#_i
                MOV X,A
   #endasm
}
```

[Preventive measures]

None. To debug a source, do not define a symbol in the #asm block.

(12) When a compiler is used and if a source including the CALLF function is compiled and assembled, an illegal line number data is output.

[Description]

When a compiler is used and if a source including the CALLF function is compiled and the output assembler source is assembled, an illegal line number data is output.

[Preventive measures]

None

B.2.3 Limitations of linker

(1) An error message is output if the object module file of the source file exceeding 8128 lines is linked.

[Description]

An error message (F117 Too Many line number data in 'file name') is output if the object module file of the source file exceeding 8128 lines is linked.

[Preventive measures]

None. Keep the source file to within 8128 lines.

(2) An error message is output if the object file created with an assembler lower than RA78K3 Ver. 5.00 is input.

[Description]

An error message (A909 Can't read DEVICE_FILE file 'file name') is output if the object file created with an assembler lower than RA78K3 Ver. 5.00 (not supporting device file) is input.

[Preventive measures]

None. Assemble by using an assembler of RA78K3 Ver. 5.00 or later.

B.2.4 Limitations of ECC generator

Unless the -U option (complement option) is specified by the object converter when ECCGEN is used, the ECC error correcting code of the last code may be illegal.

[Preventive measures]

When using the ECC generator, specify the -U option with the object converter (oc78k3), and specify appropriate values for the addresses of the internal ROM.

[MEMO]

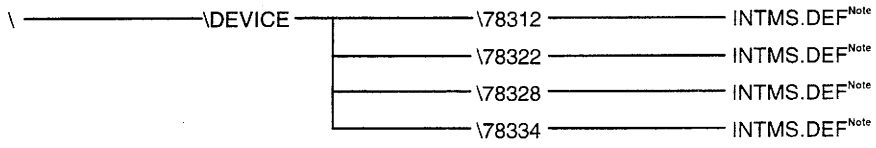
APPENDIX C USING SUPPLIED FILE (INTMS.DEF)

This appendix describes the supplied file (INTMS.DEF) that defines macros to allocate an area used for interrupt vector tables and macro services that are expected to be often used when the RA78K3 is used.

C.1 Overview

The INTMS.DEF file defines a macro that is used to allocate an area or define the name of an area.

The INTMS.DEF file for each subseries exists in the following directory.



Note For the μPD78312 and 78312A Subseries

Caution The INTMS.DEF file is a file supplied with the RA78K3 of the old version (earlier than Ver. 5.00). Therefore, it is supplied only to maintain compatibility with the RA78K3 earlier than Ver. 5.00. Therefore, the file for the μPD78352A, 78356, 78366, 78366A, and 78372 Subseries is not supplied.

C.2 Using Macro for Vector Table Setting

A vector table used for interrupts is set by writing the name of a macro prepared for each interrupt. The first address of an interrupt servicing routine must be defined by the name of an interrupt (such as INTP0).

This macro generates a segment, and the segment name of that segment is determined for each interrupt. Make sure that the segment name does not overlap the other symbol names.

Table C-1 shows the interrupt names, macro names, and segment names.

[Example]

```
        ADVENT  ; Setting of vector table (macro reference)
                ; Interrupt vector of INTAD is set in segment name I_ADVT
INTAD:
        Interrupt servicing routine
```

Caution This macro does not support (use of external memory addresses 8000H through 807FH as a vector table) when TPF = 1.

Table C-1. Interrupt Names, Macro Names, and Segment Names (1/2)

Interrupt Name	Macro Name	Segment Name	Target Subseries			
			μPD78312 μPD78312A	μPD78322	μPD78328	μPD78334
RST[Note]	RSTVENT	RSTVT	○	○	○	○
NMI	NMIVENT	NMIVT	○	○	○	○
INTWDT	WDTVENT	WDTVT	○	○	○	○
INTCR00	CR00VENT	I_CR00VT	○			
INTCR01	CR01VENT	I_CR01VT	○			
INTCR10	CR10VENT	I_CR10VT	○			
INTCR11	CR11VENT	I_CR11VT	○			
INTE0	E0VENT	I_E0VT	○			
INTE1	E1VENT	I_E1VT	○			
INTE2	E2VENT	I_E2VT	○			
INTTM0	TM0VENT	I_TM0VT	○			
INTTM1	TM1VENT	I_TM1VT	○			
INTTM2	TM2VENT	I_TM2VT	○			
INTOV	OVVENT	I_OVVT		○		○
INTOV0	OV0VENT	I_OV0VT			○	
INTP0	P0VENT	I_P0VT		○	○	○
INTP1	P1VENT	I_P1VT		○	○	○
INTP2	P2VENT	I_P2VT		○	○	○
INTP3	P3VENT	I_P3VT		○		○
INTCC00R	C00RVENT	I_C00RVT				○
INTP4	P4VENT	I_P4VT		○		○
INTCCX0	CCX0VENT	I_CCX0VT		○		
INTCC01R	C01RVENT	I_C01RVT				○
INTP5	P5VENT	I_P5VT		○		○
INTCC01	CC01VENT	I_CC01VT		○		

Note Reset vector

Table C-1. Interrupt Names, Macro Names, and Segment Names (2/2)

Interrupt Name	Macro Name	Segment Name	Target Subseries			
			μPD78312 μPD78312A	μPD78322	μPD78328	μPD78334
INTP6	P6VENT	I_P6VT		○		○
INTOV1	OV1VENT	I_OV1VT			○	
INTCM00	CM00VENT	I_CM00VT		○	○	
INTCM01	CM01VENT	I_CM01VT		○	○	
INTCM02	CM02VENT	I_CM02VT		○	○	
INTCM03	CM03VENT	I_CM03VT		○	○	
INTCM04	CM04VENT	I_CM04VT			○	
INTCM05	CM05VENT	I_CM05VT			○	
INTCM06	CM06VENT	I_CM06VT			○	
INTCC10	CC10VENT	I_CC10VT				○
INTCMX0	CMX0VENT	I_CMX0VT		○		
INTCM10	CM10VENT	I_CM10VT		○		○
INTCM11	CM11VENT	I_CM11VT				○
INTCM12	CM12VENT	I_CM12VT			○	○
INTCM20	CM20VENT	I_CM20VT				○
INTCM21	CM21VENT	I_CM21VT				○
INTCM30	CM30VEMT	I_CM30VT	○	○	○	○
INTSER	SERVENT	I_SERVT	○	○	○	○
INTSR	SRVENT	I_SRVT	○	○	○	○
VINTST	STVENT	I_STVT		○	○	○
INTCSI	CSIVENT	I_CSIVT	○	○	○	○
INTAD	ADVENT	I_ADVT	○			
INTTB	TBVENT	I_TBVT				
BRK_I[Note 1]	BRKVENT	BRKVT	○	○	○	○
TRAP0[Note 2]	TR0TENT	T_TR0TT		○	○	○

Notes **1.** Break instruction

2. Op code trap

C.3 Using Macro Service Control Word Area Allocating Macro (except μPD78312 and 78312A Subseries)

By describing a macro name prepared for each macro service, a macro service control word area is allocated and a label is created.

This macro creates the label of a macro service control word and the label of each byte in the macro service control word. It also creates a segment name.

The interrupt name and macro name using a macro service, and the created label name and segment name are in compliance with the following convention.

Label of macro service control word:	"MCW" + abbreviated interrupt name + "P"
Label of macro service control register:	"MSM" + abbreviated interrupt name
Label of macro service channel pointer:	"CHP" + abbreviated interrupt name
Label of macro service counter (counter mode):	"MAC" + abbreviated interrupt name
Label of compare byte data storage area (data compare mode):	"DTC" + abbreviated interrupt name
Label of SFR pointer (data shift mode):	"SPRP" + abbreviated interrupt name
Segment name:	Macro name + "_S" or macro name + "S"

For the interrupt name, macro name, and abbreviated interrupt name, refer to Table C-2.

Table C-2. Interrupt Names, Macro Names, and Abbreviated Interrupt Names

Interrupt Name	Macro Name	Abbreviated Interrupt Name	Target Subseries		
			μPD78322	μPD78328	μPD78334
INTOV	OVMCW	OV	○		○
INTOV0	OV0MCW	OV0		○	
INTP0	P0MCW	P0	○	○	○
INTP1	P1MCW	P1	○	○	○
INTP2	P2MCW	P2	○	○	○
INTP3	P3MCW	P3	○		○
INTCC00R	C00RMCW	C00R			○
INTP4	P4MCW	P4	○		○
INTCCX0	CCX0MCW	CCX0	○		
INTCC01R	C01RMCW	C01R			○
INTP5	P5MCW	P5	○		○
INTCC01	CC01MCW	CC01	○		
INTP6	P6MCW	P6	○		○
INTOV1	OV1MCW	OV1		○	
INTCM00	CM00MCW	CM00	○	○	
INTCM01	CM01MCW	CM01	○	○	
INTCM02	CM02MCW	CM02	○	○	
INTCM03	CM03MCW	CM03	○	○	
INTCM04	CM04MCW	CM04		○	
INTCM05	CM05MCW	CM05		○	
INTCM06	CM06MCW	CM06		○	
INTCC10	CC10MCW	CC10		○	
INTCMX0	CMX0MCW	CMX0			○
INTCM10	CM10MCW	CM10	○		
INTCM11	CM11MCW	CM11	○		○
INTCM12	CM12MCW	CM12			○
INTCM20	CM20MCW	CM20		○	○
INTCM21	CM21MCW	CM21			○
INTCM30	CM30MCW	CM30			○
INTSR	SRMCW	SR	○	○	○
INTST	STMCW	ST	○	○	○
INTCSI	CSIMCW	CSI	○	○	○
INTAD	ADMCW	AD	○	○	○

[Example]

```
CSIMCW
MOVW AX,#0××××H
MOVW MCWCSIP,AX

Or

MOV  A,#0××H
MOV  MSMCSI,A
MOV  A,#0××H
MOV  CHPCSI,A
```

C.4 Using Macro Service Channel Area Allocating Macro (μPD78312 and 78312A Subseries only)

By describing a macro name prepared for each macro service channel, the area of the macro service channel is allocated and a label is created.

This macro creates each label and segment name in a macro service channel.

The interrupt name and macro name using the macro service, and the related label name and segment name are in compliance with the following convention.

Label of macro service pointer:	"MSP" + channel number + "P"
Label of macro service counter:	"MSC" + channel number
Label of SFR pointer:	"SFR" + channel number
Segment name:	Macro name + "_S"

For the channel name, macro name, and channel number, refer to Table C-3.

Table C-3. Channel Name, Macro Name, and Channel Number

Channel Name	Macro Name	Channel Number
Channel 0	MCH0	0
Channel 1	MCH1	1
Channel 2	MCH2	2
Channel 3	MCH3	3
Channel 4	MCH4	4
Channel 5	MCH5	5
Channel 6	MCH6	6
Channel 7	MCH7	7

[Example]

```
MCH0

MOVW MSP0P,#MEMORY
MOV  MSC0,#10H
MOV  SFRP0,#LOW P0
```

C.5 Using Macro Service Channel Area Allocating Macro (except μPD78312 and 78312A Subseries)

The 78K/III Series is provided with nine types of macro services. Table C-4 lists the name of each service and the name of the macro service channel area allocating macro corresponding to that macro service.

Table C-4. Macro Service Names and Macro Names

Macro Service Name		Macro Name	Target Subseries		
			μPD78322	μPD78328	μPD78334
Counter mode		–	○	○	○
Data compare mode		–	○	○	○
Data shift mode		–	○	○	○
PTOIVL (CC = 01)[Note 1]		–		○	
Bit pattern operation mode		DS_BL	○	○	○
A/D conversion result transfer mode	Byte transfer	DS_ADB	○	○	
	Word transfer	DS_ADW	○	○	
Block transfer mode	Byte transfer	DS_BTB	○	○	○
	Word transfer	DS_BTW	○	○	○
Data differential mode	Byte transfer	DS_DDB	○	○	○
	Word transfer	DS_DDW	○	○	○
Data differential mode (with memory pointer)	Byte transfer	DS_DDB_P	○		○
	Word transfer	DS_DDW_P	○		○
Data addition mode	Byte transfer	DS_DAB	○		
	Word transfer	DS_DAW	○		
PTOIVL (CC = 00, 10)[Note 1]		DS_PTOI		○	
PTODTR (CC = 00,01)[Note 2]		DS_PTDT0		○	
PTODTR (CC = 10)[Note 2]		DS_PTDT1		○	
Sequential pulse output control mode 1		DS_POSEQ			○
Sequential pulse output control mode 2		DS_POPRL			○

Notes 1. PTOIVL: Programmable timer output interval mode
2. PTODTR: Programmable timer output data transfer mode

(1) Bit pattern operation mode

[Description format]

DS_BL ch_name,'relocation'

[Parameter]

ch_name: Name of macro service channel (within 4 alphanumeric characters)

'relocation': Relocation attribute of macro service channel.
Be sure to enclose in ' ' (normally, 'SADDR').

[Created symbol name]

"MCH" + ch_name: Segment name of macro service channel

"BP" + ch_name: Address of bit pattern

"CHA" + ch_name: Address to be set to channel pointer of macro service control word

"SFRP" + ch_name: Address of SFRP

[Example]

```
DS_BL   P0,'SADDR'

MOV     A,#LOW CHAP0
MOV     !CHPPO,A

MOV     BPP0,#10101010B
MOV     SFRPP0,#LOW P0
```

(2) A/D conversion result transfer mode

[Description format]

DS_ADB ch_name,buff_size,'relocation'
DS_ADW ch_name,buff_size,'relocation'

[Parameter]

ch_name:	Name of macro service channel (within 4 alphanumeric characters)
buff_size:	Size of buffer (number of buffers)
'relocaton':	Relocation attribute of macro service channel Be sure to enclosed in ' '. Normally, this is 'SADDR', but specify 'SADDRP' for a word buffer.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"BA" + ch_name:	First address of macro service buffer (DS_ADB)
"BA" + ch_name + "P":	First address of macro service buffer (DS_ADW)
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"MSC" + ch_name:	Address of macro service counter

[Example]

```
DS_ADW      AD,03H,'SADDRP'

MOV         A,#LOW CHAAD
MOV         !CHPAD,A

MOV         MSCAD,#03H
```

(3) Block transfer mode

[Description format]

DS_BTB ch_name,'relocation'
DS_BTW ch_name,'relocation'

[Parameter]

ch_name: Name of macro service channel (within 4 alphanumeric characters)
'relocation': Relocation attribute of macro service channel
Be sure to enclosed in ' '.
Normally, specify 'SADDRP' because the created segment must be located at an even address of FE00H to FEFFH.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"MP" + ch_name + "P":	Address of memory pointer
"MPL" + ch_name:	Address of low-order byte of memory pointer
"MPH" + ch_name:	Address of high-order byte of memory pointer
"MSC" + ch_name:	Address of macro service counter
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"SFRP" + ch_name:	Address of SFR pointer

[Example]

```
DS_BTB      SR,'SADDRP'

MOV         A,#LOW CHASR
MOV         !CHPSR,A

MOV         SFRPSR,#LOW RXB
MOVW        MPSRP,#BUFF0
MOV         MSCSR,#03H
```

(4) Data differential mode

[Description format]

DS_DDB ch_name,buff_size,'relocation'
DS_DDW ch_name,buff_size,'relocation'

[Parameter]

ch_name:	Name of macro service channel (within 4 alphanumeric characters)
buff_size:	Size of buffer (number of buffers)
'relocation':	Relocation attribute of macro service channel Be sure to enclosed in ' '. Because LDB must be always located at an even address of FE00H to FEFFH, specify an address (odd or even) according to the size of the buffer (number of buffers).

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"BA" + ch_name:	First address of macro service buffer (DS_DDB)
"BA" + ch_name + "P":	First address of macro service buffer (DS_DDW)
"LDB" + ch_name + "P":	Address of LDB
"LDBL" + ch_name:	Address of low-order byte of LDB
"LDBH" + ch_name:	Address of high-order byte of LDB
"MSC" + ch_name:	Address of macro service counter
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"SFRP" + ch_name:	Address of SFR pointer

[Example]

```
DS_DOW      RPU,03H,'SADDRP'

MOV         A,#LOW CHARPU
MOV         !CHPP4,A

MOV         SFRPRPU,#LOW CCXOUW
MOVW        LDBRPUP,#DUMY0
MOV         MSCRPU,#03H
```

(5) Data differential mode (with memory pointer)

[Description format]

DS_DDB_P ch_name,'relocation'
DS_DDW_P ch_name,'relocation'

[Parameter]

ch_name: Name of macro service channel (within 4 alphanumeric characters)
'relocation': Relocation attribute of macro service channel
Be sure to enclosed in ' '.
Normally, specify 'SADDRP' because the created segment must be located at an even address of FE00H to FEFFH.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"MP" + ch_name + "P":	Address of memory pointer
"MPL" + ch_name:	Address of low-order byte of memory pointer
"MPH" + ch_name:	Address of high-order byte of memory pointer
"LDB" + ch_name + "P":	Address of LDB
"LDBL" + ch_name:	Address of low-order byte of LDB
"LDBH" + ch_name:	Address of high-order byte of LDB
"MSC" + ch_name:	Address of macro service counter
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"SFRP" + ch_name:	Address of SFR pointer

[Example]

```
DS_DDW_P    RPU,'SADDRP'

MOV         A,#LOW CHARPU
MOV         !CHPCCX0,A

MOV         SFRPRPU,#LOW CCXOUW
MOVW        MPRPUP,#BUFF2
MOVW        LDBRPUP,#DUMY1
MOV         MSCRPU,#03H
```

(6) Data addition mode

[Description format]

DS_DAB ch_name,'relocation'
DS_DAW ch_name,'relocation'

[Parameter]

ch_name:	Name of macro service channel (within 4 alphanumeric characters)
'relocation':	Relocation attribute of macro service channel Be sure to enclosed in ' '. Normally, this is 'SADDR' but specify 'SADDRP' for a word buffer.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"ADD" + ch_name:	Address of addition data storage area (DS_DAB)
"ADD" + ch_name + "P":	Address of addition data storage area (DS_DAW)
"ADDL" + ch_name:	Address of low-order byte of addition data storage area (DS_DAW)
"ADDH" + ch_name:	Address of high-order byte of addition data storage area (DS_DAW)
"SFR2" + ch_name:	Address of transfer destination SFR pointer
"SFR1" + ch_name:	Address of transfer source SFR pointer
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"MSC" + ch_name:	Address of macro service counter

[Example]

```
DS_DAW      CM00,'SADDRP'
MOV         A,#LOW CHACM00
MOV         !CHPCM00,A

MOV         SFR2CM00,#LOW CM00
MOV         SFR1CM00,#LOW CM00
MOVW        ADDCM00P,#DATA0
MOV         MSCM00,#01H
```

(7) Programmable timer output interval mode (when CC = 00, 10)

[Description format]

DS_PTOI ch_name,'relocation'

[Parameter]

ch_name: Name of macro service channel (within 4 alphanumeric characters)

'relocation': Relocation attribute of macro service channel
Be sure to enclosed in ' '.
Normally, specify 'SADDRP' because the created segment must be located at an even address of FE00H to FEFFH.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"WD" + ch_name + "P":	Address of word data
"WDL" + ch_name:	Address of low-order byte of word data
"WDH" + ch_name:	Address of high-order byte of word data

[Example]

```
DS_PT01     CM06,'SADDRP'

MOV         A,#LOW CHACM06
MOV         !CHPCM06,A

MOVW        WDCM06P,#W_DTA
```

(8) Programmable timer output data transfer mode (mode 0: when CC = 00, 01)

[Description format]

DS_PTDT0 ch_name,'relocation'

[Parameter]

ch_name: Name of macro service channel (within 4 alphanumeric characters)

'relocation': Relocation attribute of macro service channel
Be sure to enclosed in ' '.
Normally, specify 'SADDRP' because the created segment must be located at an even address of FE00H to FEFFH.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"WD0" + ch_name + "P":	Address of word data 0
"WD0L" + ch_name:	Address of low-order byte of word data 0
"WD0H" + ch_name:	Address of high-order byte of word data 0
"WD1" + ch_name + "P":	Address of word data 1
"WD1L" + ch_name:	Address of low-order byte of word data 1
"WD1H" + ch_name:	Address of high-order byte of word data 1

[Example]

```
DS_PTDT0    CM00,'SADDRP'

MOV         A,#LOW CHACM00
MOV         !CHPCM00,A

MOVW        WD0CM00P,#W_DTA0
MOVW        WD1CM00P,#W_DTA1
```

(9) Programmable timer output data transfer mode (mode 1: when CC = 10)

[Description format]

DS_PTDT1 ch_name,'relocation'

[Parameter]

ch_name: Name of macro service channel (within 4 alphanumeric characters)

'relocation': Relocation attribute of macro service channel
Be sure to enclosed in ' '.
Normally, specify 'SADDRP' because the created segment must be located at an even address of FE00H to FEFFH.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"WD0" + ch_name + "P":	Address of word data 0
"WD0L" + ch_name:	Address of low-order byte of word data 0
"WD0H" + ch_name:	Address of high-order byte of word data 0
"WD1" + ch_name + "P":	Address of word data 1
"WD1L" + ch_name:	Address of low-order byte of word data 1
"WD1H" + ch_name:	Address of high-order byte of word data 1
"WD2" + ch_name + "P":	Address of word data 2
"WD2L" + ch_name:	Address of low-order byte of word data 2
"WD2H" + ch_name:	Address of high-order byte of word data 2

[Example]

```
DS_PTDT1    CM01,'SADDRP'

MOV         A,#LOW CHACM01
MOV         !CHPCM01,A

MOVW        WD0CM01P,#W_DTA0
MOVW        WD1CM01P,#W_DTA1
MOVW        WD2CM01P,#W_DTA2
```

(10) Successive pulse output control mode 1

[Description format]

DS_POSEQ ch_name,'relocation'

[Parameter]

ch_name: Name of macro service channel (within 4 alphanumeric characters)

'relocation': Relocation attribute of macro service channel
Be sure to enclosed in ' '.
Normally, specify 'SADDRP' because the created segment must be located at an even address of FE00H to FEFFH.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"WDn" + ch_name + "P":	Address of word data n (n = 0 to 5)
"WDnL" + ch_name:	Address of low-order byte of word data n (n = 0 to 5)
"WDnH" + ch_name:	Address of high-order byte of word data n (n = 0 to 5)

[Example]

```
DS_POSEQ    RPU,'SADDRP'

MOV         A,#LOW CHARPU
MOV         !CHPCMX0,A

MOVW        WD0RPUP,#W_DTA0
            •
            •
            •
MOVW        WD5RPUP,#W_DTA5
```

(11) Successive pulse output control mode 2

[Description format]

DS_POPRL ch_name,'relocation'

[Parameter]

ch_name: Name of macro service channel (within 4 alphanumeric characters)

'relocation': Relocation attribute of macro service channel
Be sure to enclosed in ' '.
Normally, specify 'SADDRP' because the created segment must be located at an even address of FE00H to FEFFH.

[Created symbol name]

"MCH" + ch_name:	Segment name of macro service channel
"CHA" + ch_name:	Address to be set to channel pointer of macro service control word
"WDn" + ch_name + "P":	Address of word data n (n = 0 to 5)
"WDnL" + ch_name:	Address of low-order byte of word data n (n = 0 to 5)
"WDnH" + ch_name:	Address of high-order byte of word data n (n = 0 to 5)
"T" + ch_name + "P":	Address of word data T
"TL" + ch_name:	Address of low-order byte of word data T
"TH" + ch_name:	Address of high-order byte of word data T
"MSC" + ch_name:	Address of macro service counter

[Example]

```
DS_POPRL    RPU,'SADDRP'

MOV         A,#LOW CHARPU
MOV         !CHPCMX0,A

MOVW        WD0RPUP,#W_DTA0
            •
            •
            •
MOVW        WD5RPUP,#W_DTA5
MOVW        TRPUP,#T_DTA
MOV         MSCRPU,#10H
```

C.6 Including File

The INTMS.DEF file is included by the include control instruction.
Include the INTMS.DEF file at the beginning of a module body (immediately after the control instruction described at the beginning of the source program).

Example

```
$       PC (312)
$       IC (C :\DEVICE\78312\INTMS.DEF)
        NAME TEST
$       IC (C :\DEVICE\78312\SFRBIT.DEF)
```

The assembler can specify the search path of the include file on a start line or by an environmental variable. By using this function, therefore, only a file name can be specified in a source file without a path specified.

Example With MS-DOS

Specifying search path

- To specify on start line of assembler

  RA78K3 TEST -IA : \RA78K3\DEVICE\78312

- To specify with environmental variable

  Input as follows on the command line of MS-DOS.

  SET INC78K3 = C : \RA78K3\DEVICE\78312

In these cases, describe as follows in the source file.

```
$       PC (3212)
$       IC (INTMS.DEF)
        NAME  TEST
$       IC (SFRBIT.DEF)
```

[MEMO]

APPENDIX D NOTES ON USING DEVICE FILE

This appendix describes the points to be noted when using the device file for the μPD78312, 78312A, 78322, 78328, and 78334 Subseries supplied with the RA78K3.

For the μPD78352A, 78356, 78366, 78366A, and 78372 Subseries, refer to the document supplied with an optional device file (Notes on Using DF783xx Device File).

D.1 Device File

A device file is a binary file having model-dependent information prepared for each model of the target device.

The device file includes the following information:

- SFR names (special function register names), SFR bit names
- Default link directive data
- Interrupt request names

D.2 Correspondence between Target Devices and Device Files

The correspondence between each target device and the model specified when assemble or compile is performed is as shown in Tables D-1 through D-6.

Immediately after installation, all device files to which each subseries correspond, and the file necessary for development tools are copied. Basically, it is recommended to use these files as is. If you wish to delete unnecessary files to save the disk capacity, do so by referring to Tables D-1 through D-6.

(1) μPD78312 Subseries

Table D-1. Required Device Files (μPD78312 Subseries)

Target Device	Model Specification	Required Device File
μPD78310	-c310	d310.78k
μPD78312	-c312	d312.78k
μPD78P312	-cp312	dp312.78k

(2) μPD78312A Subseries

Table D-2. Required Device Files (μPD78312A Subseries)

Target Device	Model Specification	Required Device File
μPD78310A	-c310A	d310a.78k
μPD78312A	-c312A	d312a.78k
μPD78P312A	-cp312A	dp312a.78k

(3) μPD78322 Subseries

Table D-3. Required Device Files (μPD78322 Subseries)

Target Device	Model Specification	Required Device File
μPD78320	-c320	d320.78k
μPD78322	-c322	d322.78k
μPD78P322	-cp322	dp322.78k
μPD78323	-c323	d323.78k
μPD78324	-c324	d324.78k
μPD78P324	-cp324	dp324.78k

(4) μPD78328 Subseries

Table D-4. Required Device Files (μPD78328 Subseries)

Target Device	Model Specification	Required Device File
μPD78327	-c327	d327.78k
μPD78328	-c328	d328.78k
μPD78P328	-cp328	dp328.78k

(5) μPD78334 Subseries

Table D-5. Required Device Files (μPD78334 Subseries)

Target Device	Model Specification	Required Device File
μPD78330	-c330	d330.78k
μPD78334	-c334	d334.78k
μPD78P334	-cp334	dp334.78k

(6) Development tools

Table D-6. Required Device Files (development tools)

Development Tools	Required Device File
RA78K3	d*.78k[Note]
CC78K3	d*.78k[Note]

Note * indicates an alphanumeric character. Use the device file required for the target device by referring to Tables D-1 through D-5.

D.3 SFR Name and SFR Bit Name

The name of a special function register (SFR) and its bit can be specified by using a predetermined symbol when each development tool is used. This symbol is called an SFR name or SFR bit name. This symbol is treated as a reserved word of the assembler package or C compiler package.

When the C compiler package is used, the SFR names and SFR bit names are recognized if the #pragma sfr command is used. The other tools recognize them as standard. The assembler package and C compiler package outputs alarm and error messages in response to inappropriate access based on this information.

(1) For the SFR names and SFR bit names that can be used with the μPD78312 and 78312A Subseries, refer to the following document.

Document name: μPD78312A Special Function Register List
Document number: IEM-5118

(2) For the SFR names and SFR bit names that can be used with the μPD78322 Subseries, refer to the following document.

Document name: μPD78322 Special Function Register List
Document number: IEM-5501

(3) For the SFR names and SFR bit names that can be used with the μPD78328 Subseries, refer to the following document.

Document name: μPD78328 Special Function Register List
Document number: IEM-5514

(4) For the SFR names and SFR bit names that can be used with the μPD78334 Subseries, refer to the following document.

Document name: μPD78334 Special Function Register List
Document number: IEM-5518

D.4 Default Link Directive Information

Each device in the 78K/III Series has different internal ROM and RAM capacities. Each device file includes default link directive data necessary for the assembler package to relocate user program and data according to the internal ROM and RAM capacities of each device.

The user should change this default setting by creating a link directive for each target system according to the memory configuration of the actual target system and giving instructions to the assembler package (linker). Note that the user program, data, and stack are not appropriately located with the default link directive data. This means that, for example, the user data or stack may be located overlapping the register bank area or reserved area of the C compiler package.

Tables D-7 through D-10 shows this information of each model.

Note that the area name ROM and area name RAM are essential area names. Unless explicitly specified by the MERGE statement, all code segments (CSEG) are relocated to the area of the area name ROM, and all data segments (DSEG) and bit segments (BSEG) are relocated to the area of the area name RAM.

(1) μPD78312 and 78312A Subseries

Table D-7. Default Link Directive Data (μPD78312 and 78312A Subseries)

Model	Default Link Directive Data
μPD78310 μPD78310A	MEMORY ROM: (00000H, 0FE00H) MEMORY RAM: (0FE00H, 00200H)
μPD78312 μPD78312A μPD78P312A	MEMORY ROM: (00000H, 02000H) MEMORY RAM: (0FE00H, 00200H)

(2) μPD78322 Subseries

Table D-8. Default Link Directive Data (μPD78322 Subseries)

Model	Default Link Directive Data
μPD78320	MEMORY ROM: (00000H, 0FC80H) MEMORY RAM: (0FC80H, 00380H)
μPD78322 μPD78P322	MEMORY ROM: (00000H, 04000H) MEMORY RAM: (0FC80H, 00380H)
μPD78323	MEMORY ROM: (00000H, 0FB00H) MEMORY RAM: (0FB00H, 00500H)
μPD78324 μPD78P324	MEMORY ROM: (00000H, 08000H) MEMORY RAM: (0FB00H, 00500H)

(3) μPD78328 Subseries

Table D-9. Default Link Directive Data (μPD78328 Subseries)

Model	Default Link Directive Data
μPD78327	MEMORY ROM: (00000H, 0FD00H) MEMORY RAM: (0FD00H, 00300H)
μPD78328 μPD78P328	MEMORY ROM: (00000H, 04000H) MEMORY RAM: (0FD00H, 00300H)

(4) μPD78334 Subseries

Table D-10. Default Link Directive Data (μPD78334 Subseries)

Model	Default Link Directive Data
μPD78330	MEMORY ROM: (00000H, 0FB00H) MEMORY RAM: (0FB00H, 00500H)
μPD78334 μPD78P334	MEMORY ROM: (00000H, 08000H) MEMORY RAM: (0FB00H, 00500H)

Caution As the default specification, the area name RAM includes the SFR area (0FF00H through 0FFFFH). However, because the SFR area is treated by the linker as a reserved area, no segment (user data and stack) is relocated in this area. For example, even if MEMORY RAM: (0FE00H, 00100H) is specified with the μPD78310A, the result is the same as the default setting of MEMORY RAM: (0FE00H, 00200H).

D.5 Interrupt Request Name

When describing an interrupt routine (interrupt function) in C language, the function described in C language is specified by the #pragma vect or #pragma interrupt command. At this time, the type of the interrupt is given by a symbol as a parameter. This symbol is called an interrupt request name. The interrupt request name is used by the C compiler package. The C compiler package creates an appropriate interrupt vector from the specified interrupt request name and interrupt function name.

This information is as shown in Tables D-11 through D-14.

As the interrupt request name of the maskable interrupt, the symbol of the corresponding interrupt request signal is assigned. To the other special interrupt sources, specific symbols are given.

(1) μPD78312 and 78312A Subseries

Table D-11. Interrupt Request Name (μPD78312 and 78312A Subseries)

Interrupt Request Name	Interrupt Vector Table Address	Interrupt Request Name	Interrupt Vector Table Address
RST	000H	INTCR00	01AH
NMI	002H	INTCR01	01CH
INTE0	004H	INTCR10	01EH
INTE1	006H	INTCR11	020H
INTE2	008H	INTSER	022H
INTWDT	00AH	INTSR	024H
INTTB	00CH	INTST	026H
INTTM0	00EH	INTAD	028H
INTTM1	010H	BRK_I	03EH
INTTM2	012H		

Remark RST: Reset, NMI, INTWDT: Non-maskable interrupt, BRK_I: Software interrupt

(2) μPD78322 Subseries

Table D-12. Interrupt Request Name (μPD78322 Subseries)

Interrupt Request Name	Interrupt Vector Table Address	Interrupt Request Name	Interrupt Vector Table Address
RST	000H	INTCM01	018H
NMI	002H	INTCM02	01AH
INTWDT	004H	INTCM03	01CH
INTOV	006H	INTCM10	01EH
INTP0	008H	INTCM11	020H
INTP1	00AH	INTSER	022H
INTP2	00CH	INTSR	024H
INTP3	00EH	INTST	025H
INTP4/INTCCX0	010H	INTCSI	028H
INTP5/INTCC01	012H	INTAD	02AH
INTP6	014H	TRAP0	03CH
INTCM00	016H	BRK_I	03EH

Remark RST: Reset, NMI, INTWDT: Non-maskable interrupt,
BRK_I, TRAP0: Software interrupt

(3) μPD78328 Subseries

Table D-13. Interrupt Request Name (μPD78328 Subseries)

Interrupt Request Name	Interrupt Vector Table Address	Interrupt Request Name	Interrupt Vector Table Address
RST	000H	INTCM04	018H
NMI	002H	INTCM05	01AH
INTWDT	004H	INTCM06	01CH
INTOV0	006H	INTCC10	01EH
INTP0	008H	INTCM20	020H
INTP1	00AH	INTSER	022H
INTP2	00CH	INTSR	024H
INTOV1	00EH	INTST	025H
INTCM00	010H	INTCSI	028H
INTCM01	012H	INTAD	02AH
INTCM02	014H	TRAP0	03CH
INTCM03	016H	BRK_I	03EH

Remark RST: Reset, NMI, INTWDT: Non-maskable interrupt,
BRK_I, TRAP0: Software interrupt

(4) μPD78334 Subseries

Table D-14. Interrupt Request Name (μPD78334 Subseries)

Interrupt Request Name	Interrupt Vector Table Address	Interrupt Request Name	Interrupt Vector Table Address
RST	000H	INTCM11	018H
NMI	002H	INTCM12	01AH
INTWDT	004H	INTCM20	01CH
INTOV	006H	INTCM21	01EH
INTP0	008H	INTCM30	020H
INTP1	00AH	INTSER	022H
INTP2	00CH	INTSR	024H
INTP3/INTCC00R	00EH	INTST	026H
INTP4/INTCC01R	010H	INTCSI	028H
INTP5	012H	INTAD	03AH
INTP6	014H	TRAP0	03CH
INTCMX0	016H	BRK_I	03EH

Remark RST: Reset, NMI, INTWDT: Non-maskable interrupt,
BRK_I: Software interrupt, TRAP: Exception

[MEMO]

APPENDIX E LIST OF OPTIONS

In this appendix, the program options are summarized in table form.
Please refer to these when developing programs.
This list of options can also be used as an index.

E. 1 List of Assembler Options

No.	Classification	Description format	Function	Relation to other options	Interpretation when omitted	Ref. page
1	Specify device type	-C [device type]	Specifies the device type of the target device.	Independent	Cannot be omitted	53
2	Specify object module file output	-O [output file name]	Specifies the output of an object module file.	If both options -O and -NO are specified at the same time, the option specified last takes precedence.	-O [input file name.REL]	54
		-NO	Specifies that no object module file is output.			
3	Specify forced object module file output	-J	Specifies that the object module file can be output even if a fatal error occurs.	If both options -J and -NJ are specified at the same time, the option specified last takes precedence.	-NJ	55
		-NJ	Makes option -J unavailable.			
4	Specify debug data output	-G	Specifies that debug data is to be added to an object module file.	If both options -G and -NG are specified at the same time, the option specified last takes precedence.	-G	56
		-NG	Makes option -G unavailable.			
		-GA	Specifies that assembler source debugging data is to be added to an object module file by the structured assembler.	If both options -GA and -NGA are specified at the same time, the option specified last takes precedence.	-GA	58
		-NGA	Makes option -GA unavailable.			
5	Specify length of symbol name	-S	Specifies that the recognizable length of a symbol name is to be extended to a maximum of 31 characters.	If both options -S and -NS are specified at the same time, the option specified last takes precedence.	-S	60
		-NS	Makes option -S unavailable.			
6	Specify symbol name case	-CA	Specifies that no distinction is made between uppercase and lowercase characters in a symbol name.	If both options -CA and -NCA are specified at the same time, the option specified last takes precedence.	-NCA	61
		-NCA	Specifies that a distinction is made between uppercase and lowercase characters in a symbol name.			
7	Specify include file read path	-I path name [, path name] ... (two or more path names can be specied)	Specifies input of an include file from a specified path.	Independent	Path specified by the environmental variable (INC78K3)	62
8	Specify assemble list file output	-P [output file name]	Specifies output of an assemble list file. It also specifies the destination and file name of the output file.	If both options -P and -NP are specified at the same time, the option specified last takes precedence.	-P [input file name.PRN]	63
		-NP	Makes option -P unavailable.			
9	Specify assemble list file data	-KA	Outputs an assemble list into an assemble list file.	If -KS and _KX are specified at the same time, -KS is ignored.	-KA	64
		-NKA	Makes option -KA unavailable.			
		-KS	Outputs an assemble list followed by a symbol list into an assemble list file.	If both options -KA and -NKA, both options -KS and -NKS, or both options -KX and -NKX are specified at the same time, the option specified last takes precedence.	-NKS	66
		-NKS	Makes option -KS unavailable.			
		-KX	Outputs an assemble list followed by a cross-reference list into an assemble list file.	If options -NKA, -NKS and -NKX are all specified, the assemble list file cannot be output.	-NKX	67
		-NKX	Makes option -KX unavailable.			

No.	Classification	Description format	Function	Relation to other options	Interpretation when omitted	Ref. page
10	Specify assemble list file format	-LW [number of characters]	Changes the number of characters that can be printed in 1 line in a list file.	If option -NP is specified, option -LW is unavailable.	-LW 132 (80 characters for display)	69
		-LL [number of lines]	Changes the number of lines that can be printed in 1 page in an assemble list file.	If option -NP is specified, option -LL is unavailable.	-LL 66 (page feed is not performed in the case of display output)	71
		-LH [character string]	Specifies the character string printed in the title column of the header of an assemble list file.	If option -NP is specified, option -LH is unavailable.	None	73
		-LT [number of characters]	Specifies a number of characters to be developed in a tab.	If option -NP is specified, option -LT is unavailable.	-LT8	76
		-LF	Inserts a form feed (FF) code at the end of an assemble list file.	If both options -LF and -NLF are specified at the same time, the option specified last takes precedence. If option -NP is specified, option -LF is unavailable.	-NLF	79
		-NLF	Makes the -LF option unavailable.			
11	Specify error list file output	-E [output file name]	Outputs an error list file.	If both options -E and -NE are specified at the same time, the option specified last takes precedence.	-NE	80
		-NE	Makes the -E option unavailable.			
12	Specify parameter file	-F File name	Inputs assembler options and the input file name from a specified file.	Independent	Options and input files can only be specified on the execution line.	82
13	Specify path for temporary file creation	-T Path name	Creates a temporary file in a specified path.	Independent	Path specified by environmental variable TMP	84
14	Specify help	--	Displays a help message on the display. Description format: --	When option -- is specified, all other options are unavailable.	No display	88

E. 2 List of Linker Options

No.	Classification	Description format	Function	Relation to other options	Interpretation when omitted	Ref. page
1	Specify load module file output	-O [output file name]	Outputs a load module file.	If both options -O and -NO are specified at the same time, the option specified last takes precedence.	-O (input file name).LNK	112
		-NO	Does not output a load module file.			
2	Specify forced load module file output	-J	Forces output of a load module file.	If both options -J and -NJ are specified at the same time, the option specified last takes precedence.	-NJ	113
		-NJ	Makes option -J unavailable.			
3	Specify debug data output	-G	Outputs debugging data to a load module file.	If both options -G and -NG are specified at the same time, the option specified last takes precedence.	-G	114
		-NG	Makes option -G unavailable.	When option -NG is specified, the public symbol list and local symbol list cannot be output regardless of specification of -KP or -KL.		
4	Specify generation of stack decision symbols	-S [area name]	Automatically generates stack decision public symbols.	If both options -S and -NS are specified at the same time, the option specified last takes precedence.	-NS	115
		-NS	Makes option -S unavailable.			
5	Specify directive file	-D file name	Specifies a particular file to be input as a directive file.	Independent	—	117
6	Specify link list file output	-P [output file name]	Specifies output of a link list file.	If both options -P and -NP are specified at the same time, the option specified last takes precedence.	-P [input file name.MAP]	118
		-NP	Makes option -P unavailable.			
7	Specify link list file data	-KM	Outputs a map list into a link list file.	If both options -KM and -NKM are specified at the same time, the option specified last takes precedence.	-KM	119
		-NKM	Makes option -KM unavailable.	If options -NKM, -NKP and -NKL are all specified, the link list file cannot be output even if option -P is specified.		
		-KD	Outputs a link directive file into a link list file.	If option -NKM is specified, option -KD becomes unavailable.	-KD	121
		-NKD	Makes option -KD unavailable.	If both options -KD and -NKD, both -KP and -NKP, or both -KL and -NKL are specified at the same time, the option specified last takes precedence.		
		-KP	Outputs a public symbol list into a link list file.		-NKP	123
		-NKP	Makes option -KP unavailable.	If option -NG is specified, the public symbol list and local symbol list cannot be output even if option -KP or -KL is specified.		
		-KL	Output a local symbol list into a link list file.		-NKL	125
		-NKL	Makes option -KL unavailable.			

No.	Classification	Description format	Function	Relation to other options	Interpretation when omitted	Ref. page
8	Specify link list format	-LL [number of lines]	Specifies number of lines that can be printed in 1 page in a link list file.	If option -NP is specified, option -LL is unavailable.	-LL66 (page feed is not performed in the case of display output)	127
		-LF	Inserts a form feed (FF) code at the end of a link list file.	If both options -LF and -NLF are specified at the same time, the option specified last takes precedence. If option -NP is specified, the option -LF is unavailable.	-NLF	129
		-NLF	Makes the -LF option unavailable.			
9	Specify error list file output	-E [file name]	Outputs error list file.	If both options -E and -NE are specified at the same time, the option specified last takes precedence.	-NE	130
		-NE	Default value: -NE Makes option -E unavailable.			
10	Specifies library file	-B file name	Inputs a specific file as a library file.	Independent	—	131
11	Specify library file read path	-I path name [, path name] ... (two or more path names can be specified)	Reads a library file from a specified path.	If a library file without a path name is specified by option -B, option -I is unavailable.	Path specified by environmental variable 'LIB78K3'	132
12	Specify parameter file	-F file name	Inputs linker options and the input file name from a specified file.	Independent	This option and the input file name can only be entered on the startup line.	133
13	Specify path for temporary file creation	-T path name	Creates a temporary file in a specified path.	Independent	Path specified by the environmental variable TMP. Current path, if no path is specified	134
14	Specify warning message output	-W [level]	Specifies whether or not a warning message is output to the console.	Independent	Outputs an ordinary error message	136
15	Specify help	--	Displays a help message on the display.	All other options are unavailable.	No display	137

E. 3 List of Object Converter Options

No.	Classification	Description format	Function	Relation to other options	Interpretation when omitted	Ref. page
1	Specify HEX format object module file output	-O [output file name]	Outputs a HEX format object module file.	If both options -O and -NO are specified at the same time, the option specified last takes precedence.	-O (input file name).HEX (file type H1 to H15 for extended space)	167
		-NO	No HEX format object module file is output.			
2	Specify symbol table file output	-S [output file name]	Outputs a symbol table file.	If both options -S and -NS are specified at the same time, the option specified last takes precedence.	-S [input file name].SYM (file type S1 to S15 for extended space)	169
		-NS	Does not output a symbol table file.			
3	Specify sort by object address order	-R	Sorts HEX format objects in order of address.	If both options -S and -NS are specified at the same time, the option specified last takes precedence. If option -NO is specified, option -R/-NR becomes unavailable.	-NR	171
		-NR	Makes option -R unavailable.			
4	Specify object complement	-U complement value [, [start] , size]	Outputs a specified complement value as an object code for an address area to which no HEX format object has been output.	If option -NO is specified, -U becomes unavailable.	—	172
5	Specify error list file output	-E [output file name]	Outputs an error list file.	If both options -E and -NE are specified at the same time, the option specified last takes precedence.	-NE	175
		-NE	Makes option -E unavailable.			
6	Specify parameter file	-F file name	Inputs options and input file names from a specified file.	Independent	Options and input file names can only be specified from the startup command line.	176
7	Specify help	--	Displays a help message on the display (console).	All other options are unavailable.	No display	180

E. 4 List of Librarian Options

No.	Classification	Description format	Function	Relation to other options	Interpretation when omitted	Ref. page
1	Specify list file format	-LW [number of characters]	Changes the number of characters that can be printed in 1 line in a list file.	Unavailable if the LIST subcommand is not specified.	-LW132 (80 characters for display output)	192
		-LL [number of lines]	Changes the number of lines that can be printed in 1 page in a list file.		-LL66 (page feed is not performed in the case of display output)	193
		-LF	Inserts a form feed (FF) code at the end of a list file.	If both options -LF and -NLF are specified at the same time, the option specified last takes precedence.	-NLF	194
		-NFL	Makes the -LF option unavailable.			
2	Specify path for temporary file creation	-T [path name]	Creates a temporary file in a specified path.	Independent	Created in the path specified by the environmental variable TMP.	195
3	Specify help	--	Displays a help message on the display.	All other options are unavailable.	No display	197

E. 5 List of List Converter Options

No.	Classification	Description format	Function	Relation to other options	Interpretation when omitted	Ref. page
1	Specify object module file input	-R [input file name]	Specifies the input of an object module file.	Independent	-R [assemble list file name.REL]	221
2	Specify load module file input	-L [input file name]	Inputs a load module file.	Independent	-L [assemble list file name.LNK]	222
3	Specify symbol name case	-CA	Specifies that no distinction is made between uppercase and lowercase characters in a symbol name.	If both options -CA and -NCA are specified at the same time, the option specified last takes precedence.	-CA	
		-NCA	Specifies that a distinction is made between uppercase and lowercase characters in a symbol name.			
4	Specify absolute assemble list file output	-O [output file name]	Outputs an absolute assemble list file.	Independent	-O [assemble list file name .P]	223
5	Specify error list file output	-E [output file name]	Outputs an error list file.	If both options -E and -NE are specified at the same time, the option specified last takes precedence.	-NE	224
		-NE	Makes option -E unavailable.			
6	Specify parameter file	-F file name	Inputs options and input file name from a specified file.	Independent	Options and input file names can only be input from the execution line.	225
7	Specify help	--	Displays a help message on the display (console).	All other options are unavailable.	No display	227

APPENDIX F LIST OF SUBCOMMANDS

This appendix is a summary of the subcommands in list form.
It will be helpful to refer to this list when developing software programs.
This list of subcommands can also serve as an index.

No.	Classification	Description Format	Function	Abbrev. format	Ref. Page
1	CREATE	CREATE△library file name [△transaction]	Creates a new library file.	C	200
2	ADD	ADD△library file name △transaction	Adds a module to a library file.	A	201
3	DELETE	DELETE△library file name∇ (∇module name [∇, ···]∇)	Deletes a module from a library file.	D	202
4	REPLACE	REPLACE△library file name △transaction	Replaces one module with another in a library file.	R	203
5	PICK	PICK△library file name∇ (∇module name [∇, ···]∇)	Retrieves a specified module from an existing library file.	P	205
6	LIST	LIST[△option]library file name [∇(∇module name)]	Outputs data on modules in a library file.	L	207
7	HELP	HELP	Displays a help message on the display (console).	H	209
8	EXIT	EXIT	Exits the librarian.	E	210

Phase-out/Discontinued

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6465-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑