

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

On April 1<sup>st</sup>, 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 1<sup>st</sup>, 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**

# **RA78K4 Ver. 1.60 or Later**

## **Assembler Package**

## **Operation**

---

## **Target Devices** **78K4 Series**

Document No. U16708EJ1V0UM00 (1st edition)

Date Published September 2003 N CP(K)

© NEC Electronics Corporation 2003

Printed in Japan

[MEMO]

**Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.**

**PC/AT is a trademark of International Business Machines Corporation.**

**HP9000 Series 700 and HP-UX are trademarks of Hewlett-Packard Company.**

**SPARCstation is a trademark of SPARC International, Inc.**

**Solaris and SunOS are trademarks of Sun Microsystems, Inc.**

• **The information in this document is current as of June, 2003. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**

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

• NEC 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 NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.

• Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

• While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

• NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

"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.

"Special": 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, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.

(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

# Regional Information

Some information contained in this document may vary from country to country. Before using any NEC Electronics product in your application, please contact the NEC Electronics 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.

## [GLOBAL SUPPORT]

<http://www.necel.com/en/support/support.html>

### **NEC Electronics America, Inc. (U.S.)**

Santa Clara, California  
Tel: 408-588-6000  
800-366-9782

### **NEC Electronics (Europe) GmbH**

Duesseldorf, Germany  
Tel: 0211-65 03 01

### **NEC Electronics Hong Kong Ltd.**

Hong Kong  
Tel: 2886-9318

- **Sucursal en España**

Madrid, Spain  
Tel: 091-504 27 87

- **Succursale Française**

Vélizy-Villacoublay, France  
Tel: 01-30-67 58 00

- **Filiale Italiana**

Milano, Italy  
Tel: 02-66 75 41

- **Branch The Netherlands**

Eindhoven, The Netherlands  
Tel: 040-244 58 45

- **Tyskland Filial**

Taeby, Sweden  
Tel: 08-63 80 820

- **United Kingdom Branch**

Milton Keynes, UK  
Tel: 01908-691-133

### **NEC Electronics Hong Kong Ltd.**

Seoul Branch  
Seoul, Korea  
Tel: 02-558-3737

### **NEC Electronics Shanghai, Ltd.**

Shanghai, P.R. China  
Tel: 021-6841-1138

### **NEC Electronics Taiwan Ltd.**

Taipei, Taiwan  
Tel: 02-2719-2377

### **NEC Electronics Singapore Pte. Ltd.**

Novena Square, Singapore  
Tel: 6253-8311

J03.4

## INTRODUCTION

This manual is intended to give users an understanding of the functions of each program in the RA78K4 Assembler Package (hereafter referred to as "the RA78K4") and the operation methods for developing software using the RA78K4.

This manual does not cover the expressions of directives and source programs or language used in the RA78K4. Therefore, before reading this manual, read the **RA78K4 Series Assembler Package Language User's Manual (U15255E)** (hereafter referred to as "the Language Manual").

The contents of this manual are intended for use with Ver. 1.60 or later of the RA78K4.

### [Target Readers]

This manual is intended for users who understand the functions and instructions of the microcontroller for which software is being developed (78K4 Series).

### [Organization]

This manual consists of the following eleven chapters and appendices:

#### **CHAPTER 1 GENERAL**

Outlines the role of the RA78K4 in microcontroller software development and the features of the RA78K4.

#### **CHAPTER 2 PRODUCT OVERVIEW AND INSTALLATION**

Explains the program file names and operating environment provided by the RA78K4.

#### **CHAPTER 3 EXECUTING RA78K4**

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 RA78K4 should read this chapter.

#### **CHAPTER 4 STRUCTURED ASSEMBLER**

#### **CHAPTER 5 ASSEMBLER**

#### **CHAPTER 6 LINKER**

#### **CHAPTER 7 OBJECT CONVERTER**

#### **CHAPTER 8 LIBRARIAN**

#### **CHAPTER 9 LIST CONVERTER**

#### **CHAPTER 10 ECC GENERATOR**

#### **CHAPTER 11 PROGRAM OUTPUT LIST**

Explains the formats of the lists output by each program.

#### **CHAPTER 12 EFFICIENT USE OF RA78K4**

Introduces some measures for optimum utilization of the RA78K4.

#### **CHAPTER 13 ERROR MESSAGES**

Explains the error messages output by each program.

**APPENDICES** Introduce a list of program options, a list of sample programs, and a list of notices on using the RA78K4.

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 Manual]

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 RA78K4, read **CHAPTER 3 EXECUTION PROCEDURE OF RA78K4**.

After becoming familiar with the operation of each program, utilize the lists in the **APPENDICES**.

### [Caution]

In this manual, it is assumed that a PC-9800 Series or an IBM PC/AT™ compatible is used as the host machine.

When the HP9000 Series 700™ or SPARCstation™ family is used, keep the following differences in mind.

- File name format is different.
  - Extension .exe of an executable file is not suffixed with an EWS version such as the HP9000 Series 700.
  - Extension .bat of a batch file is rendered .sh with an EWS version such as the HP9000 Series 700.
  - The file names shown in uppercase are in lowercase with an EWS version such as the HP9000 Series 700.
- The execution examples and the environment setup indicated in this manual differ.

### [Conventions]

The following symbols and abbreviations are used throughout this manual:

∴	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.
< >:	Names of dialog boxes and Windows
" ":	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.
<b>Note:</b>	Footnote for item marked with <b>Note</b> in the text
<b>Caution:</b>	Information requiring particular attention
<b>Remark:</b>	Supplementary information

**[Related Documents]**

The documents related to this manual are listed below.

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

**Document related to development tools (user's manuals)**

Document Name		Document No.
RA78K4 Assembler Package	Operation	This manual
	Language	U15255E
	Structured Assembler Preprocessor	U11743E
CC78K4 C Compiler	Operation	U16707E
	Language	U11571E
SM78K4 System Simulator Ver. 1.40 or Later	Reference (Windows™ Based)	U10093E
SM78K Series System Simulator Ver. 1.40 or Later	External Part User Open Interface Specifications	U10092E
ID78K Series Integrated Debugger Ver. 2.30 or Later	Operation (Windows Based)	U15185E
ID78K4 Integrated Debugger	Reference (Windows Based)	U10440E
RX78K4 Real-Time OS	Fundamental	U10603E
	Installation	U10604E

**Caution** The related documents listed above are subject to change without notice. Be sure to use the latest version of each document for designing.

## CONTENTS

<b>CHAPTER 1 GENERAL</b> .....	<b>13</b>
<b>1.1 Assembler Overview</b> .....	<b>13</b>
1.1.1 What is an assembler? .....	14
1.1.2 What is a relocatable assembler? .....	18
<b>1.2 Overview of Features of RA78K4</b> .....	<b>20</b>
1.2.1 Creating a source module file using an editor .....	21
1.2.2 Structured assembler preprocessor .....	22
1.2.3 Assembler .....	23
1.2.4 Linker .....	24
1.2.5 Object converter.....	25
1.2.6 Librarian .....	26
1.2.7 List converter .....	27
1.2.8 Debugger .....	28
<b>1.3 Reminders Before Program Development</b> .....	<b>29</b>
1.3.1 Maximum performance of RA78K4 .....	29
<b>1.4 Features of RA78K4</b> .....	<b>30</b>
<b>CHAPTER 2 PRODUCT OUTLINE AND INSTALLATION</b> .....	<b>31</b>
<b>2.1 Host Machine and Supply Medium</b> .....	<b>31</b>
<b>2.2 Installation</b> .....	<b>32</b>
2.2.1 Installation of Windows version .....	32
2.2.2 Installation of UNIX version .....	32
<b>2.3 Installation of Device Files</b> .....	<b>33</b>
2.3.1 Installation of Windows version .....	33
2.3.2 Installation of UNIX version .....	33
2.3.3 Registry registration of device files .....	33
<b>2.4 Directory Configuration</b> .....	<b>34</b>
2.4.1 Windows version directory configuration.....	34
2.4.2 UNIX version directory configuration.....	35
<b>2.5 Uninstallation Procedure</b> .....	<b>36</b>
2.5.1 Uninstallation of Windows version .....	36
2.5.2 Uninstallation of UNIX version .....	36
<b>2.6 Environment Settings</b> .....	<b>37</b>
2.6.1 Environmental variables.....	37
2.6.2 Kanji code in source file .....	37
<b>CHAPTER 3 EXECUTION PROCEDURE OF RA78K4</b> .....	<b>38</b>
<b>3.1 Before Executing RA78K4</b> .....	<b>39</b>
3.1.1 Sample programs.....	39
3.1.2 Configuration of sample program .....	42
<b>3.2 Execution Procedure of RA78K4</b> .....	<b>43</b>
<b>3.3 Execution Procedure of ST78K4</b> .....	<b>49</b>
<b>3.4 Assembling, Linking, and Object Conversion from Command Line (DOS Prompt, EWS)</b> .....	<b>54</b>
<b>3.5 Using Parameter File</b> .....	<b>58</b>

<b>CHAPTER 4 STRUCTURED ASSEMBLER .....</b>	<b>59</b>
<b>4.1 I/O Files of Structured Assembler .....</b>	<b>59</b>
<b>4.2 Functions of Structured Assembler .....</b>	<b>60</b>
<b>4.3 Structured Assembler Startup .....</b>	<b>61</b>
4.3.1 Structured assembler startup .....	61
4.3.2 Execution start and end messages .....	63
<b>4.4 Structured Assembler Options .....</b>	<b>65</b>
4.4.1 Types of structured assembler options .....	65
4.4.2 Explanation of structured assembler options .....	66
<b>4.5 Setting Options from PM plus .....</b>	<b>80</b>
4.5.1 Setting options .....	80
4.5.2 Options .....	82
<b>CHAPTER 5 ASSEMBLER .....</b>	<b>83</b>
<b>5.1 I/O Files of Assembler .....</b>	<b>83</b>
<b>5.2 Functions of Assembler .....</b>	<b>85</b>
<b>5.3 Assembler Startup .....</b>	<b>86</b>
5.3.1 Assembler startup .....	86
5.3.2 Execution start and end messages .....	88
<b>5.4 Assembler Options .....</b>	<b>90</b>
5.4.1 Types of assembler options .....	90
5.4.2 Order of precedence of assembler options .....	92
5.4.3 Explanation of assembler options .....	93
<b>5.5 Options Settings in PM plus .....</b>	<b>125</b>
5.5.1 Option setting method .....	125
5.5.2 Option settings .....	127
<b>CHAPTER 6 LINKER .....</b>	<b>129</b>
<b>6.1 I/O Files of Linker .....</b>	<b>129</b>
<b>6.2 Functions of Linker .....</b>	<b>130</b>
<b>6.3 Memory Spaces and Memory Areas .....</b>	<b>130</b>
<b>6.4 Link Directives .....</b>	<b>131</b>
6.4.1 Directive files .....	131
6.4.2 Memory directives .....	133
6.4.3 Segment location directives .....	135
<b>6.5 Linker Startup .....</b>	<b>138</b>
6.5.1 Linker startup .....	138
6.5.2 Execution start and end messages .....	140
<b>6.6 Linker Options .....</b>	<b>142</b>
6.6.1 Types of linker options .....	142
6.6.2 Order of precedence of linker options .....	144
6.6.3 Explanation of linker options .....	145
<b>6.7 Option Settings in PM plus .....</b>	<b>173</b>
6.7.1 Option setting method .....	173
6.7.2 Option settings .....	176

<b>CHAPTER 7 OBJECT CONVERTER</b> .....	<b>178</b>
<b>7.1 I/O Files of Object Converter</b> .....	<b>179</b>
<b>7.2 Functions of Object Converter</b> .....	<b>180</b>
<b>7.3 Object Converter Startup</b> .....	<b>194</b>
7.3.1 Object converter startup.....	194
7.3.2 Execution start and end messages .....	196
<b>7.4 Object Converter Options</b> .....	<b>198</b>
7.4.1 Types of object converter options .....	198
7.4.2 Explanation of object converter options .....	199
<b>7.5 Option Settings in PM plus</b> .....	<b>211</b>
7.5.1 Option setting method.....	211
7.5.2 Option settings.....	214
<b>CHAPTER 8 LIBRARIAN</b> .....	<b>215</b>
<b>8.1 I/O Files of Librarian</b> .....	<b>215</b>
<b>8.2 Functions of Librarian</b> .....	<b>217</b>
<b>8.3 Librarian Startup</b> .....	<b>219</b>
8.3.1 Librarian startup.....	219
8.3.2 Execution start and end messages .....	221
<b>8.4 Librarian Options</b> .....	<b>222</b>
8.4.1 Types of librarian options .....	222
8.4.2 Explanation of librarian options .....	223
<b>8.5 Subcommands</b> .....	<b>229</b>
8.5.1 Types of subcommands .....	229
8.5.2 Explanation of subcommands .....	229
<b>8.6 Option Settings in PM plus</b> .....	<b>238</b>
8.6.1 Option setting method.....	238
8.6.2 Option settings.....	240
<b>CHAPTER 9 LIST CONVERTER</b> .....	<b>241</b>
<b>9.1 I/O Files of List Converter</b> .....	<b>242</b>
<b>9.2 Functions of List Converter</b> .....	<b>243</b>
<b>9.3 List Converter Startup</b> .....	<b>246</b>
9.3.1 List converter startup .....	246
9.3.2 Execution start and end messages .....	248
<b>9.4 List Converter Options</b> .....	<b>249</b>
9.4.1 Types of list converter options .....	249
9.4.2 Explanation of list converter options .....	250
<b>9.5 Option Settings in PM plus</b> .....	<b>256</b>
9.5.1 Option setting method.....	256
9.5.2 Option settings.....	258
<b>CHAPTER 10 ECC GENERATOR</b> .....	<b>259</b>
<b>10.1 Position of ECC Generator</b> .....	<b>260</b>
<b>10.2 Product Contents</b> .....	<b>261</b>
<b>10.3 I/O Files of ECC Generator</b> .....	<b>262</b>
10.3.1 ECC-supported HEX-format object module file.....	263
10.3.2 ECC generator startup .....	264

10.3.3 Execution start and end messages.....	266
<b>10.4 ECC Generator Options.....</b>	<b>267</b>
10.4.1 Types of ECC generator options.....	267
10.4.2 Explanation of ECC generator options.....	268
<b>10.5 Utilization of ECC Generator .....</b>	<b>273</b>
10.5.1 How to execute operations with efficiency (use of EXIT status function) .....	273
10.5.2 How to interrupt program execution.....	273
10.5.3 How to start ECC generator efficiently.....	273
<b>CHAPTER 11 PROGRAM OUTPUT LIST .....</b>	<b>274</b>
<b>11.1 Lists Output by Assembler .....</b>	<b>275</b>
11.1.1 Assemble list file headers .....	275
11.1.2 Assemble list .....	276
11.1.3 Symbol list .....	278
11.1.4 Cross-reference list.....	279
11.1.5 Error list .....	280
<b>11.2 Lists Output by Linker.....</b>	<b>281</b>
11.2.1 Link list file headers .....	281
11.2.2 Map list .....	282
11.2.3 Public symbol list.....	283
11.2.4 Local symbol list .....	284
11.2.5 Error list .....	284
<b>11.3 List Output by Object Converter .....</b>	<b>285</b>
11.3.1 Error list .....	285
<b>11.4 List Output by Librarian .....</b>	<b>286</b>
11.4.1 Library data output list .....	286
<b>11.5 Lists Output by List Converter .....</b>	<b>287</b>
11.5.1 Absolute assemble list.....	287
11.5.2 Error list .....	287
<b>CHAPTER 12 EFFICIENT USE OF RA78K4.....</b>	<b>288</b>
<b>12.1 Improving Operating Efficiency (EXIT Status Function).....</b>	<b>289</b>
<b>12.2 Preparing Development Environment (Environmental Variables).....</b>	<b>290</b>
<b>12.3 Interrupting Program Execution.....</b>	<b>290</b>
<b>12.4 Making Assemble List Easy to Read .....</b>	<b>291</b>
<b>12.5 Reducing Program Startup Time.....</b>	<b>292</b>
12.5.1 Specifying control instruction in source program .....	292
12.5.2 Using PM plus.....	292
12.5.3 Creating parameter files and subcommand files.....	293
<b>12.6 Object Module Library Formation .....</b>	<b>294</b>
<b>CHAPTER 13 ERROR MESSAGES .....</b>	<b>295</b>
<b>13.1 Overview of Error Messages .....</b>	<b>295</b>
<b>13.2 Structured Assembler Error Messages .....</b>	<b>296</b>
<b>13.3 Assembler Error Messages .....</b>	<b>301</b>
<b>13.4 Linker Error Messages .....</b>	<b>310</b>
<b>13.5 Object Converter Error Messages.....</b>	<b>316</b>
<b>13.6 Librarian Error Messages.....</b>	<b>318</b>

13.7 List Converter Error Messages .....	321
13.8 ECC Generator Error Messages .....	323
13.9 PM plus Error Messages .....	325
<b>APPENDIX A SAMPLE PROGRAMS .....</b>	<b>327</b>
<b>A.1 Source Lists .....</b>	<b>327</b>
<b>A.2 Execution Example .....</b>	<b>329</b>
<b>A.3 Output Lists .....</b>	<b>331</b>
A.3.1 Assemble lists .....	331
A.3.2 Symbol lists .....	335
A.3.3 Cross-reference lists .....	336
A.3.4 Map list .....	337
A.3.5 Public symbol list .....	338
A.3.6 Local symbol list .....	338
A.3.7 Library data output list .....	338
A.3.8 Absolute assemble lists .....	339
<b>APPENDIX B NOTES ON USE .....</b>	<b>343</b>
<b>APPENDIX C LIST OF OPTIONS .....</b>	<b>345</b>
<b>C.1 List of Structured Assembler Options .....</b>	<b>345</b>
<b>C.2 List of Assembler Options .....</b>	<b>347</b>
<b>C.3 List of Linker Options .....</b>	<b>350</b>
<b>C.4 List of Object Converter Options .....</b>	<b>352</b>
<b>C.5 List of Librarian Options .....</b>	<b>354</b>
<b>C.6 List of List Converter Options .....</b>	<b>355</b>
<b>C.7 List of ECC Generator Options .....</b>	<b>356</b>
<b>APPENDIX D LIST OF SUBCOMMANDS .....</b>	<b>357</b>
<b>APPENDIX E INDEX .....</b>	<b>358</b>

## CHAPTER 1 GENERAL

This chapter describes the role of the RA78K4 in microcontroller software development and the features of the RA78K4.

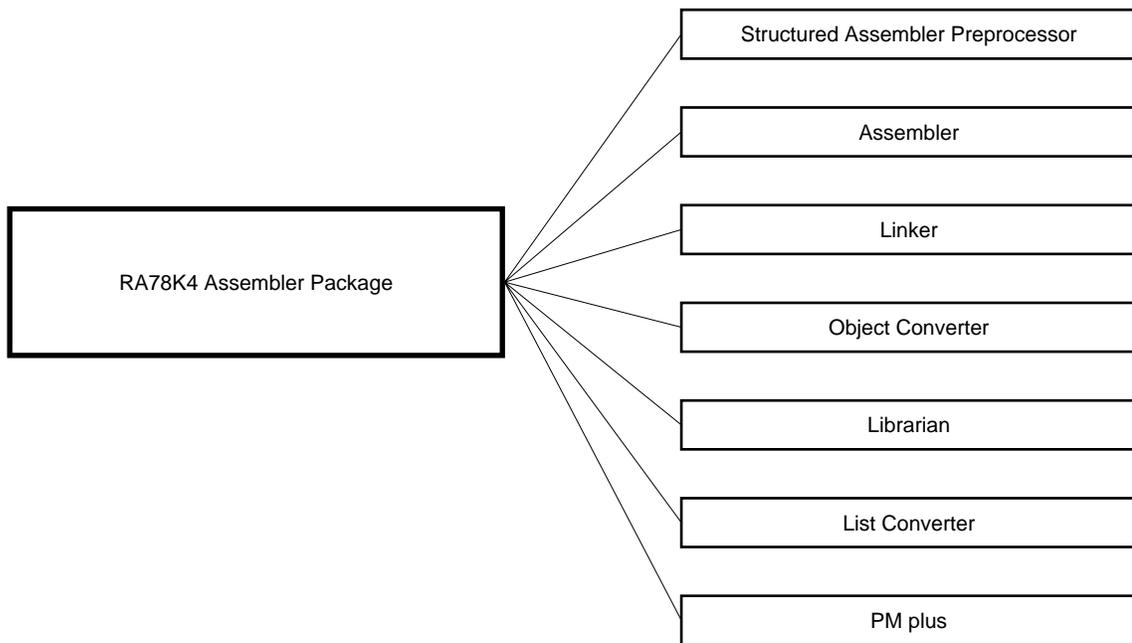
### 1.1 Assembler Overview

The RA78K4 Assembler Package is a generic term for a series of programs designed to translate source programs coded in the assembly language for 78K4 Series microcontrollers into machine language coding.

The RA78K4 contains six programs: Structured Assembler Preprocessor, Assembler, Linker, Object Converter, Librarian, and List Converter.

In addition, PM plus that makes a series of operations, including editing, compiling/assembling, linking, and debugging on Windows, easy is supplied with the RA78K4.

**Figure 1-1. RA78K4 Assembler Package**



### 1.1.1 What is an assembler?

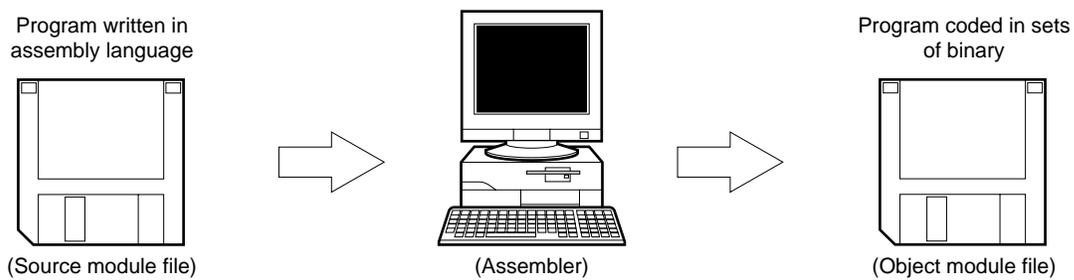
#### (1) Assembly language and machine language

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

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. The programs and data handled by the microcontroller are collections of binary numbers called machine language. For programmers, however, machine language code is difficult to remember, causing errors to occur frequently. 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.

Since the microcontroller must handle programs in machine language form, another program is required that translates programs created in assembly language into machine language. This program is called an assembler.

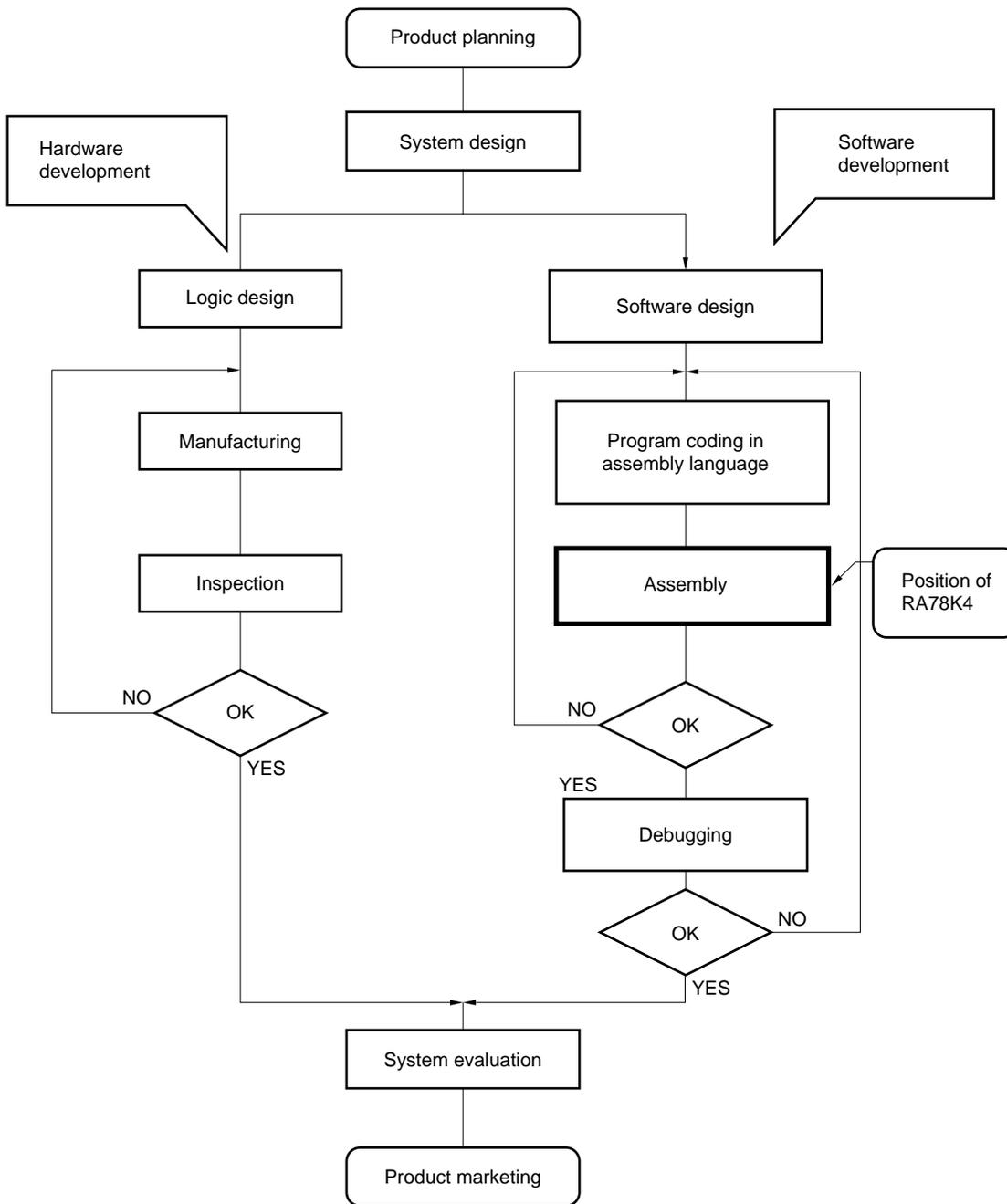
**Figure 1-2. Flow of Assembler**



(2) Development of microcontroller-related products and the role of RA78K4

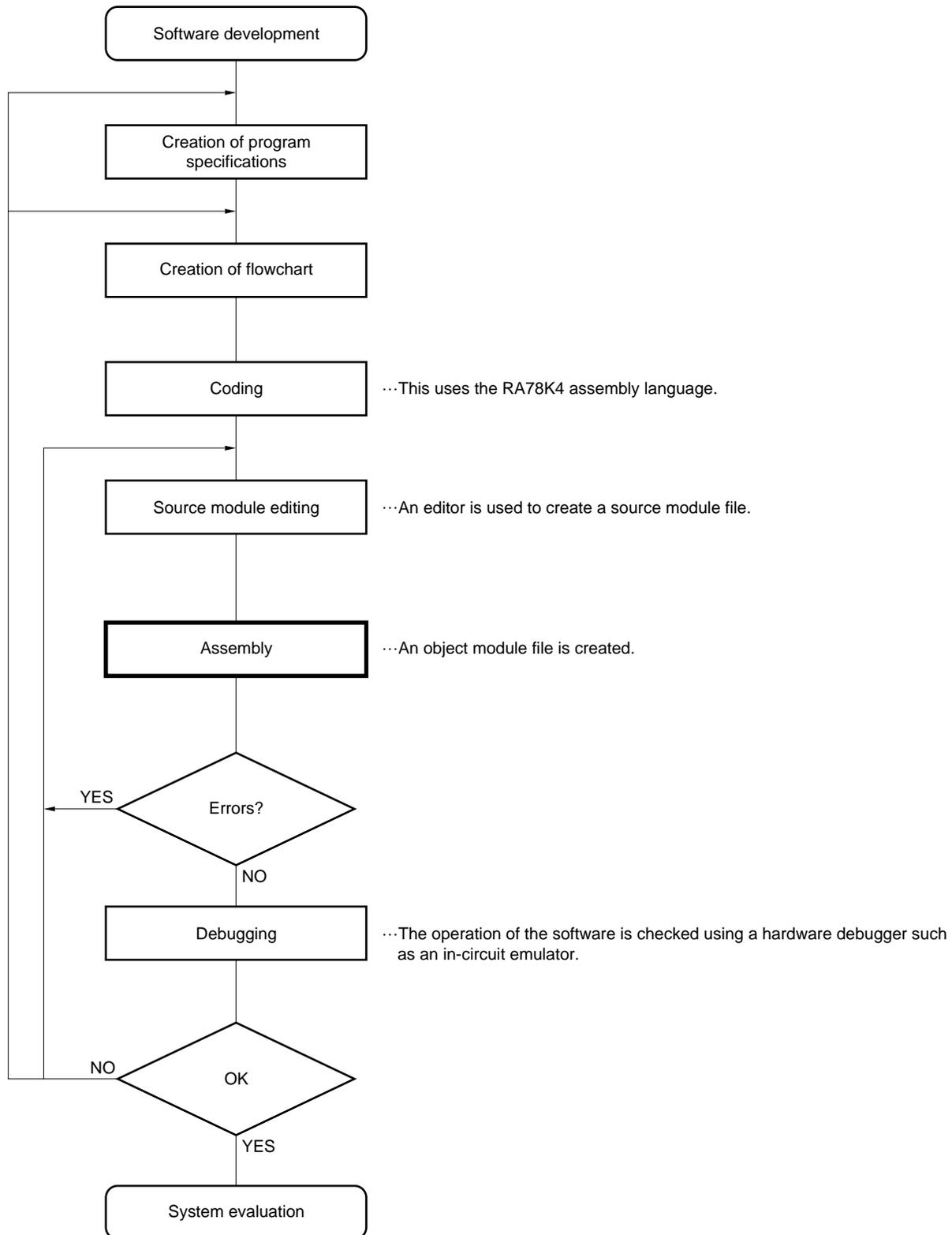
Figure 1-3 Development Process of Microcontroller-Applied Products illustrates the position of assembly-language programming in the (software) product development process.

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



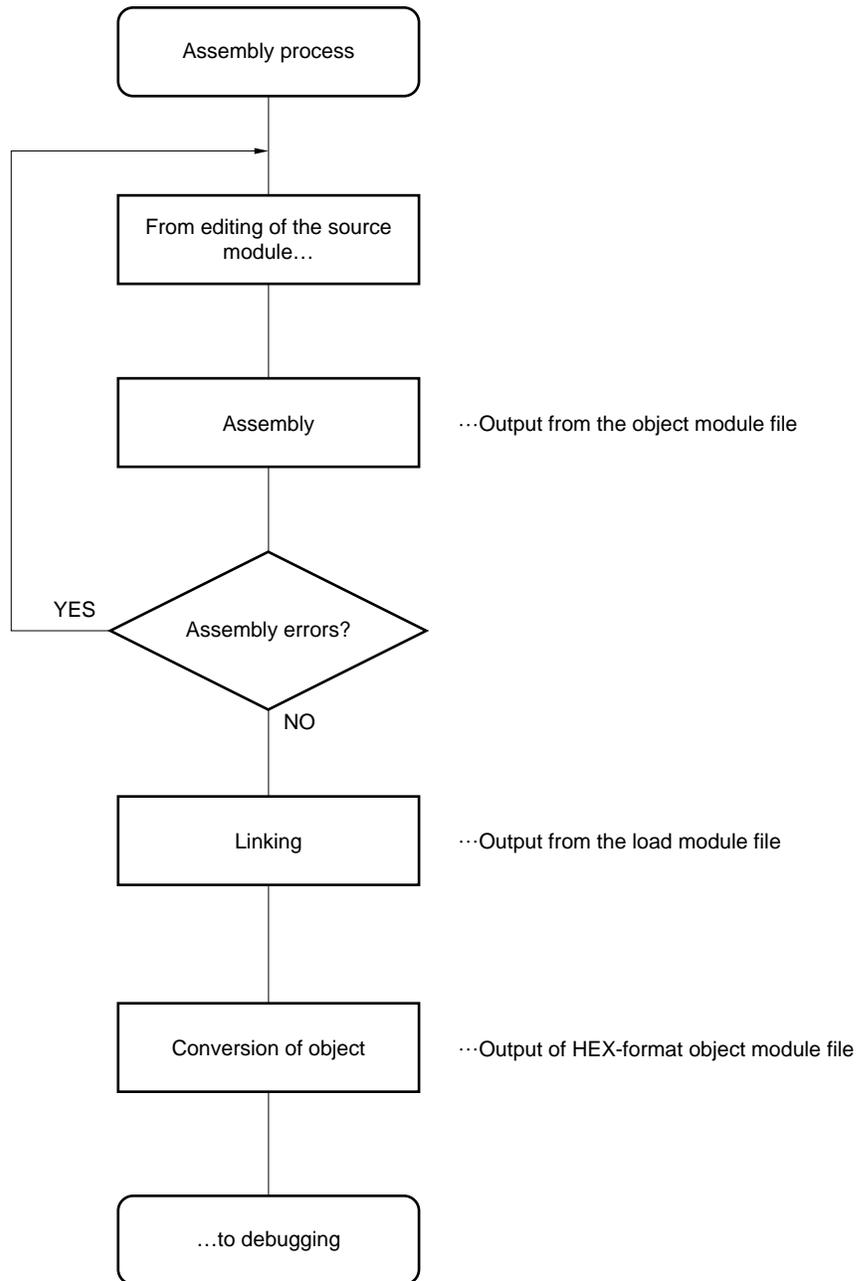
A more detailed explanation of the software development process appears in **Figure 1-4 Software Development Process**.

**Figure 1-4. Software Development Process**



The RA78K4 is then applied to the assembly process.

**Figure 1-5. RA78K4 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.

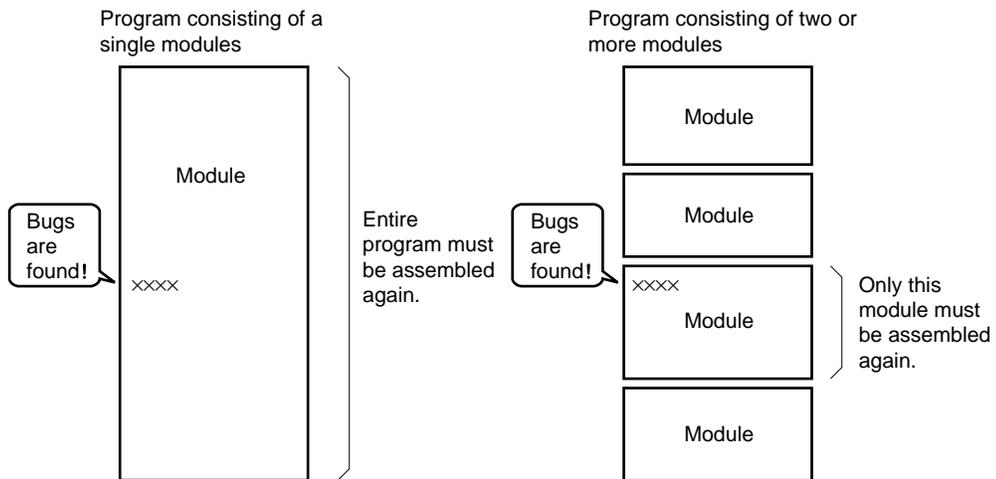
A relocatable assembler is an assembler suitable for modular programming. The following advantages can be derived from modular programming with a relocatable assembler:

**(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.

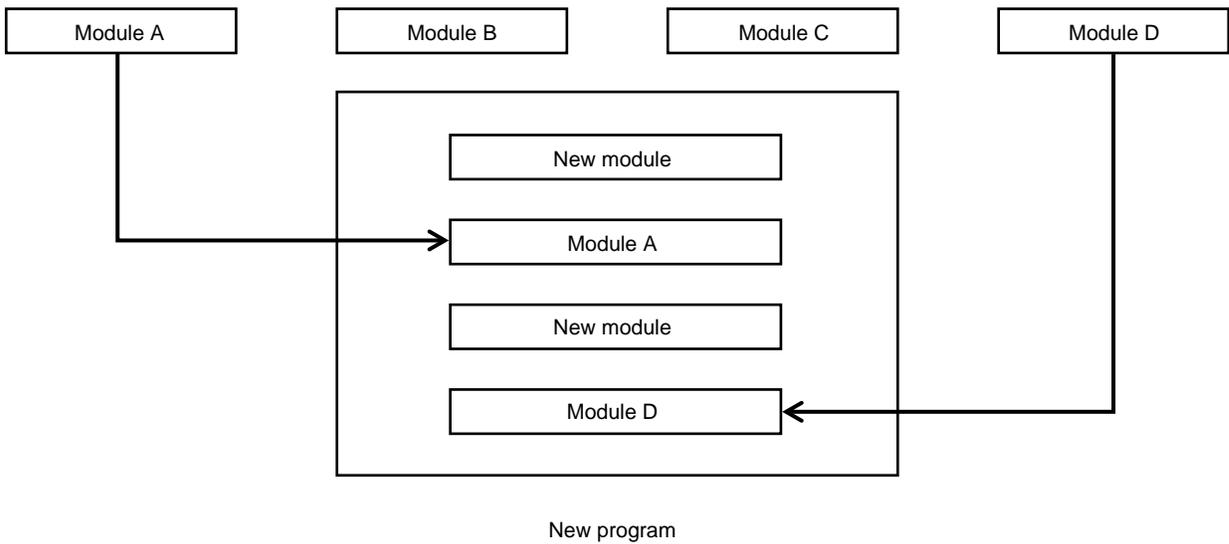
**Figure 1-6. Reassembly for Debugging**



**(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**

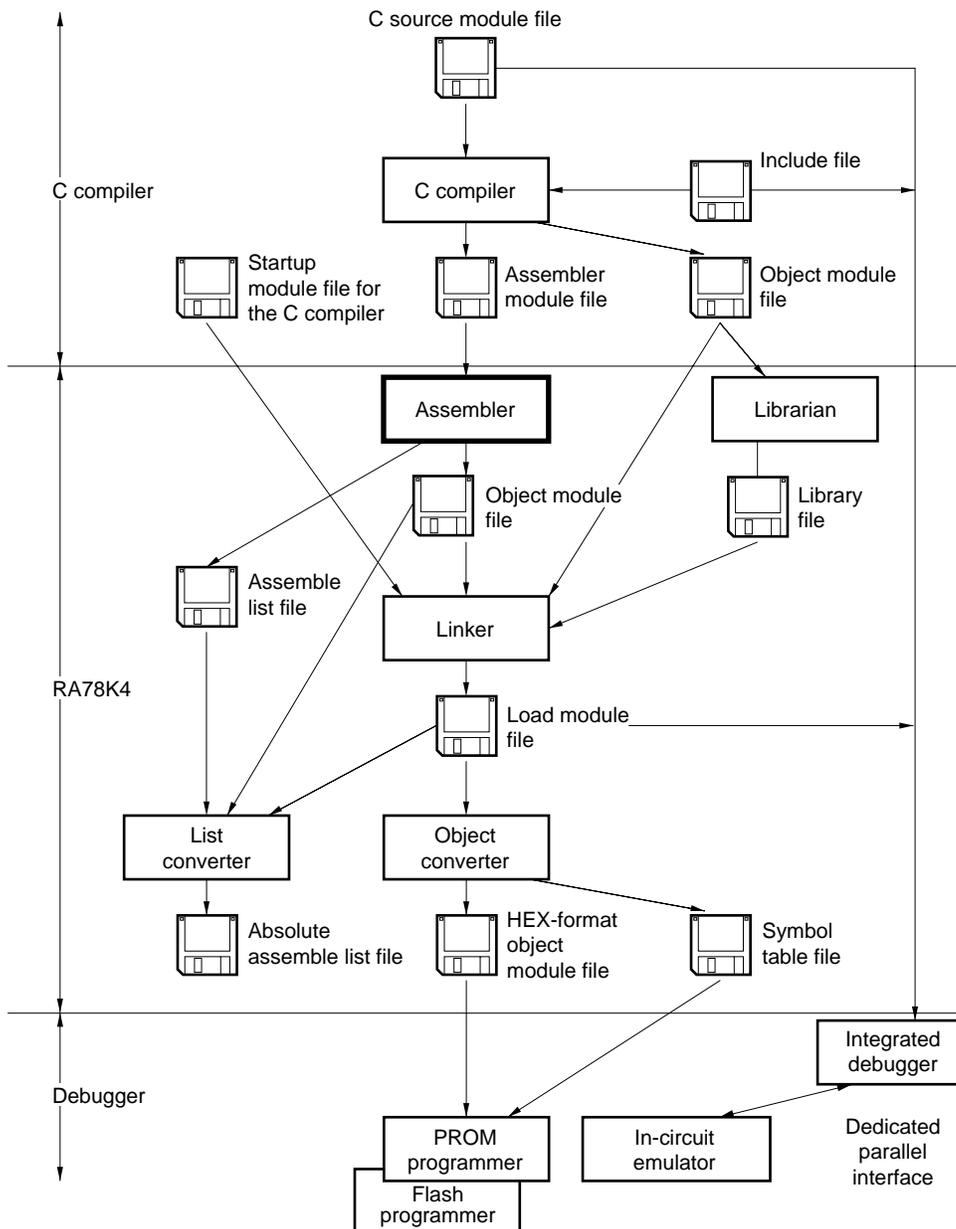


## 1.2 Overview of Features of RA78K4

The procedure for developing general programs appears in **Figure 1-8 Procedure for Program Development Using RA78K4**. 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 "RA78K4." the assembler program is referred to as the "assembler."

**Figure 1-8. Procedure for Program Development Using RA78K4**



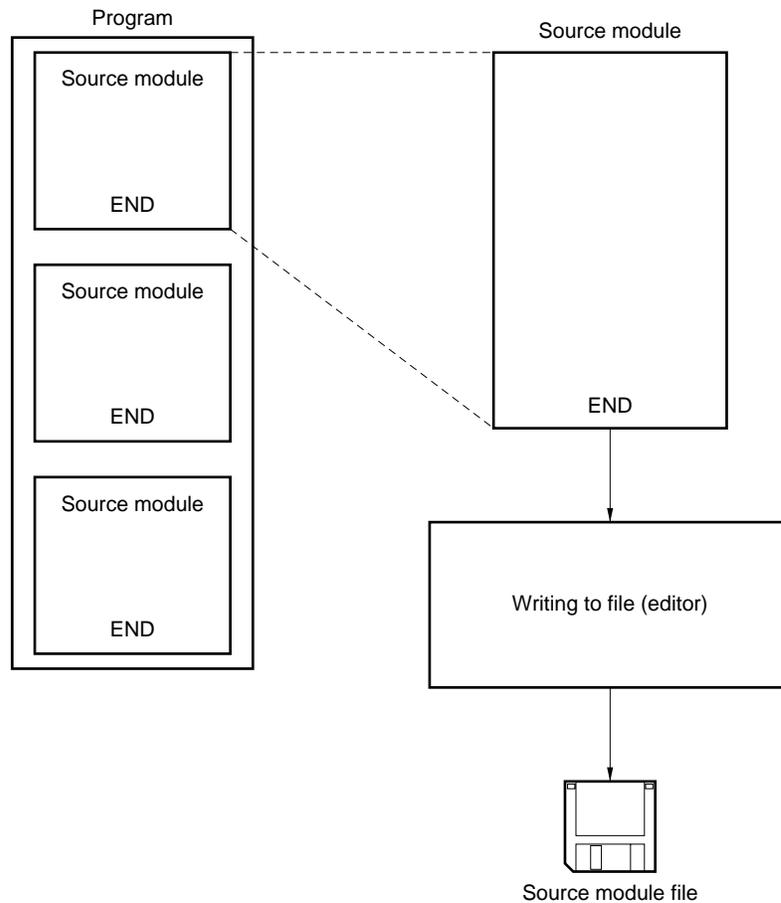
### 1.2.1 Creating a source module file using an editor

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 Source Module File**

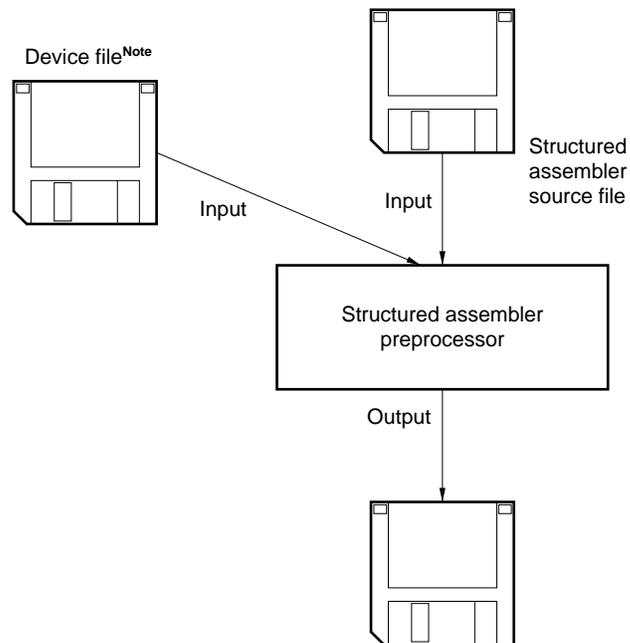


### 1.2.2 Structured assembler preprocessor

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 **RA78K4 Structured Assembler Preprocessor User's Manual (U11743E)**.

**Figure 1-10. Function of Structured Assembler Preprocessor**

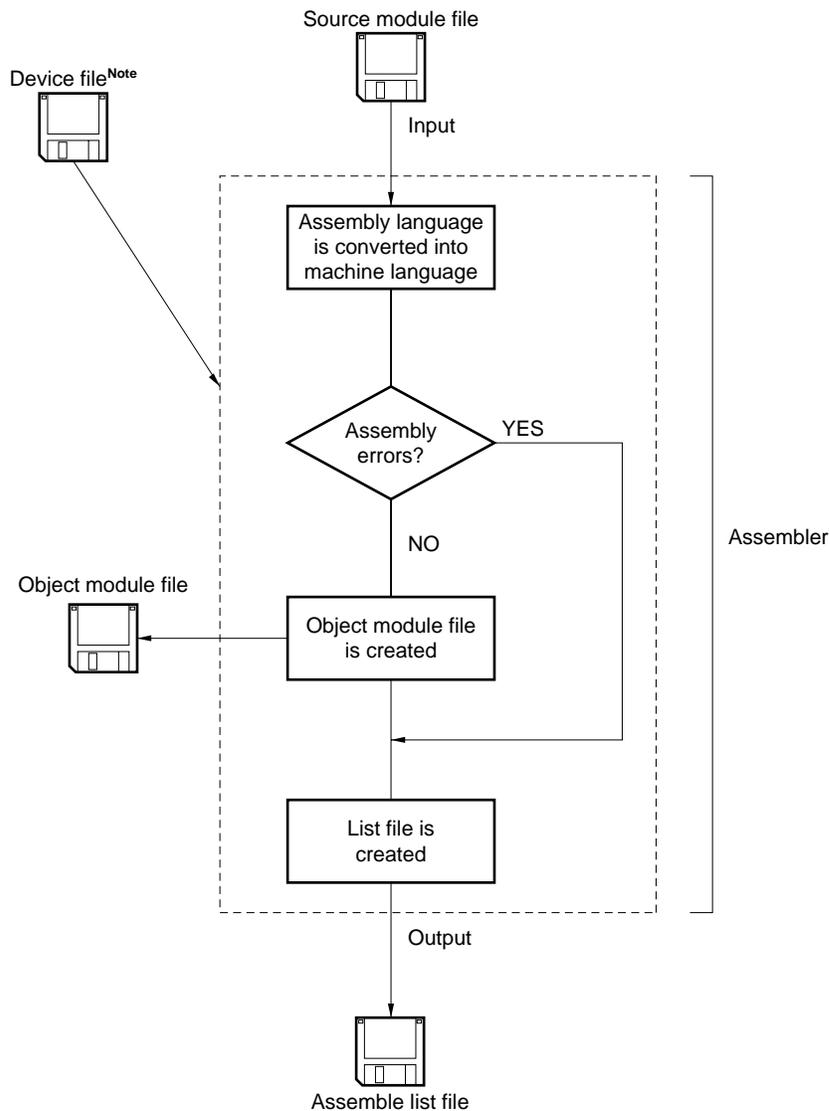


**Note** Obtain the device file separately.

### 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 Assembler**



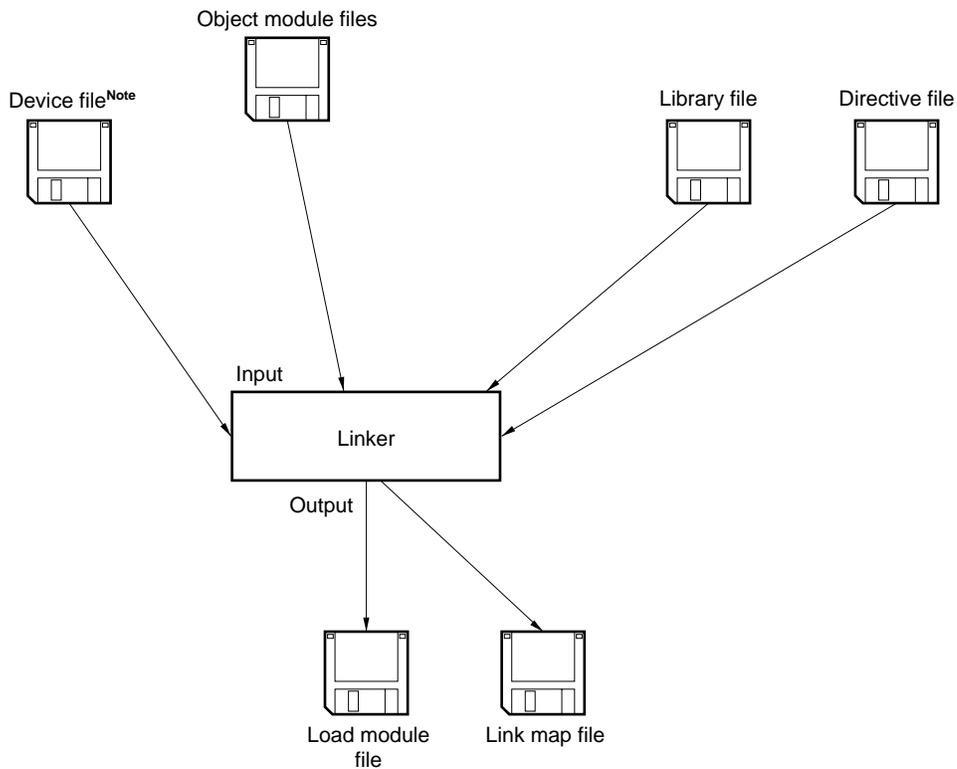
**Note** Obtain the device file separately.

### 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. Function of Linker**



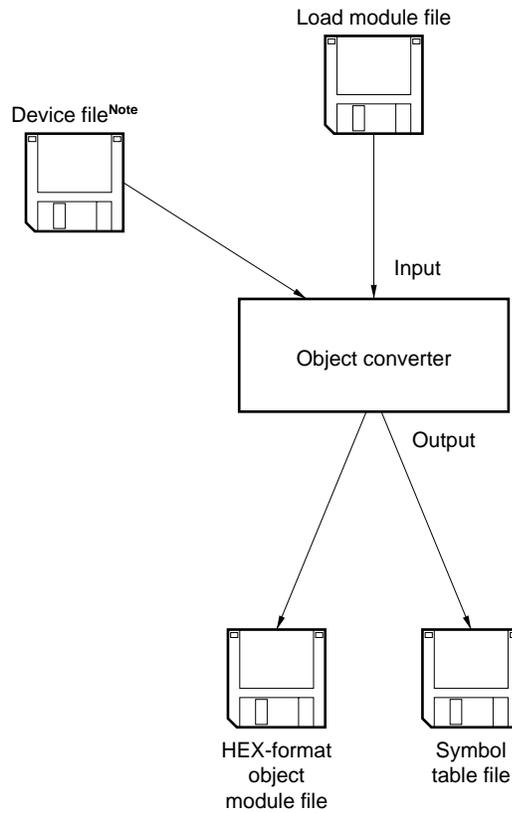
**Note** Obtain the device file separately.

### 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 Object Converter**



**Note** Obtain the device file separately.

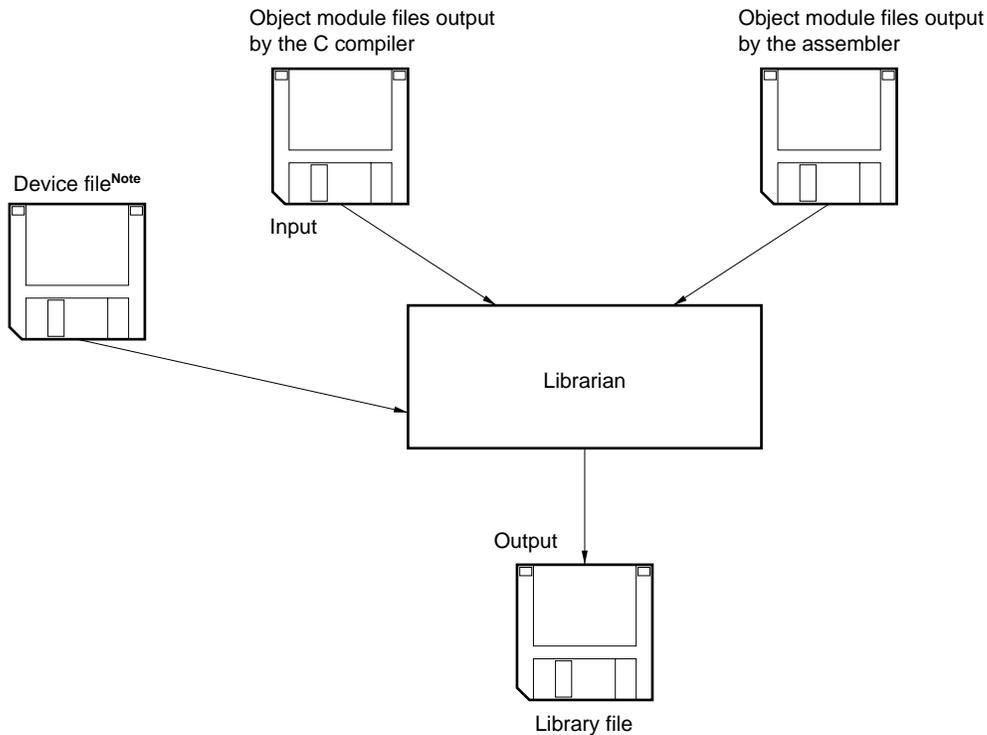
### 1.2.6 Librarian

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 Librarian**



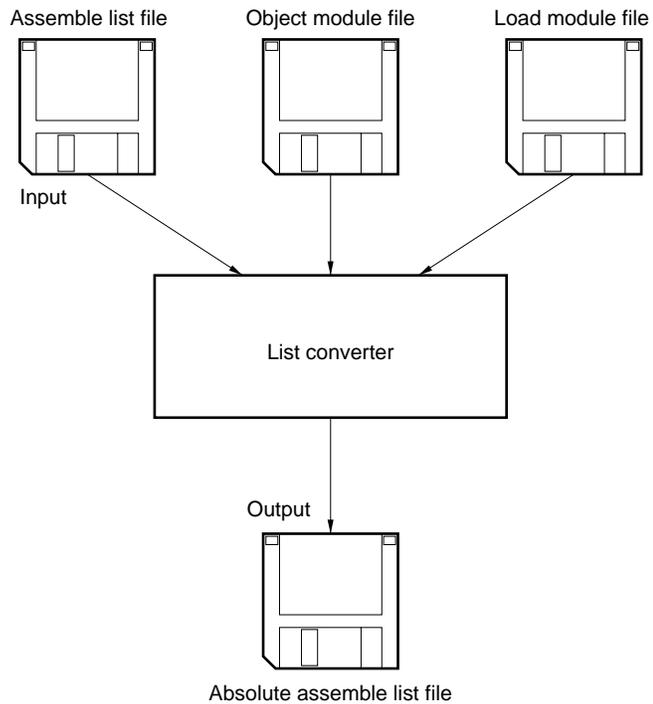
**Note** Obtain the device file separately.

### 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 List Converter**



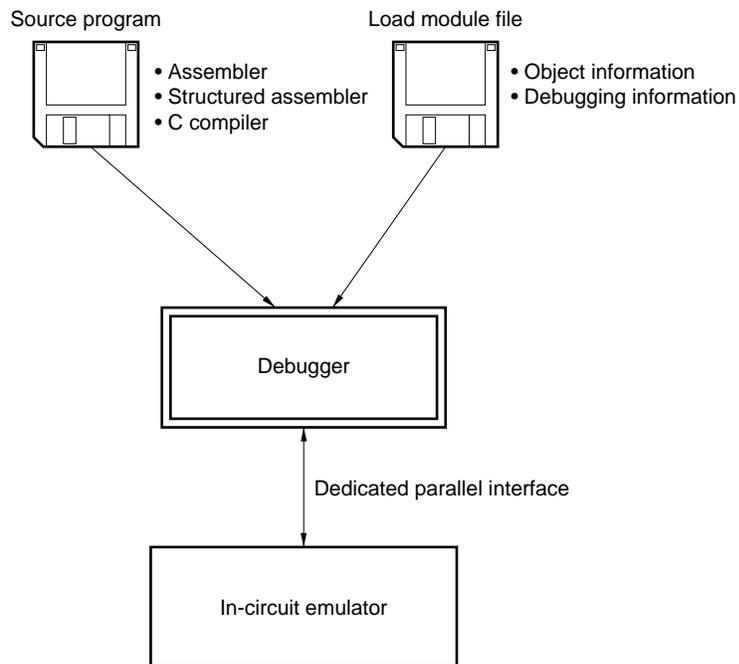
### 1.2.8 Debugger

The debugger for the 78K4 Series 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.

The debugger and IE are separate packages and are sold separately from the RA78K4.

**Figure 1-16. Function of Debugger**



### 1.3 Reminders Before Program Development

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

#### 1.3.1 Maximum performance of RA78K4

##### (1) Maximum performance of assembler

**Table 1-1. Maximum Performance of Assembler**

Item	Maximum Performance	
	PC Version	EWS Version
Number of symbols (local + public)	65535 <sup>Note 1</sup>	65535 <sup>Note 1</sup>
Number of symbols that can be output to cross reference list	65534 <sup>Note 2</sup>	65534 <sup>Note 2</sup>
Maximum macro body size referenced by one macro	1 MB	1 MB
Total size of all macro bodies	10 MB	10 MB
Number of segments in 1 file	256	256
Macros and include specifications in 1 file	10000	10000
Macros and include specifications in 1 include file	10000	10000
Relocation information <sup>Note 3</sup>	65535	65535
Line number information	65535	65535
BR/CALL directive in 1 file	32767	32767
Number of characters on 1 line of source code	2048 <sup>Note 4</sup>	2048 <sup>Note 4</sup>
Symbol length	256 characters	256 characters
Number of switch name definitions	1000	1000
Character length of switch name	31	31
Include file nesting levels in 1 file	8 levels	8 levels

- Notes**
1. If more than 2001 public symbols are used, the speed slows down because the temporary file is accessed.
  2. Excluding the number of module names and section names.
  3. Information to be passed to the linker if the symbol value cannot be resolved by the assembler.  
For example, if an externally referenced symbol is to be referenced by the MOV instruction, two pieces of relocation information are generated in a .rel file.
  4. Including CR and LF codes. If more than 2048 characters are written on one line, a warning message is output and the 2049th character and those that follow are ignored.

##### (2) Maximum performance of linker

**Table 1-2. Maximum Performance of Linker**

Item	Maximum Performance	
	PC Version	EWS Version
Number of symbols (local + public)	65535	65535
Line number information of the same segment	65535	65535
Number of segments	65535	65535
Input modules	1024	1024

## 1.4 Features of RA78K4

The RA78K4 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 RA78K4 has a directive to automatically select a branch instruction (i.e., BR directive, CALL directive).

To create a program with high memory efficiency, a 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 or CALL 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) General-purpose register selection function

General-purpose registers can be represented by absolute names (R0, R1, RP0, etc.) or by function names (X, A, AX, etc.). When user describes a function name in a source program, always describe a general-purpose register-select directive (RSS directive). The RSS directive is provided to allow description of a function name as a general-purpose register representation in a source program.

## CHAPTER 2 PRODUCT OUTLINE AND INSTALLATION

This chapter explains the procedure used to install the files stored in the supply media of the RA78K4 in the user development environment (host machine) and the procedure to uninstall these files from the user development environment.

### 2.1 Host Machine and Supply Medium

The assembler package supports the development environments shown in Table 2-1. The supply medium differs depending on the host machine.

**Table 2-1. Supply Medium and Recording Format of Assembler Package**

Host Machine	Corresponding OS (Version)	Supply Medium	Recording Format
PC-9800 Series	Japanese Windows (98/Me/2000/XP/NT™ 4.0) <sup>Note</sup>	CD-ROM	Supports Windows standard installer
IBM PC/AT compatible	Japanese Windows (98/Me/2000/XP/NT 4.0) <sup>Note</sup> English Windows (98/Me/2000/XP/NT 4.0) <sup>Note</sup>		
HP9000 Series 700	HP-UX™ (Rel. 10.10 or later)	CD-ROM	cp command
SPARCstation family	SunOS™ (Rel. 4.1.4 or later) Solaris™ (Rel. 2.5.1 or later)		

**Note** To use the assembler in Windows, PM plus is necessary.

If PM plus is not used, each tool included in the assembler package can be used from the DOS prompt (Windows 98/Me) or command prompt (Windows 2000/XP/NT 4.0)

## 2.2 Installation

### 2.2.1 Installation of Windows version

The procedure for installing to the host machine the files provided in the RA78K4's supply media is described below.

<1> Starting up Windows

Power on the host machine and peripherals and start Windows.

<2> Set supply media

Set the RA78K4's supply media in the appropriate drive (CD-ROM drive) of the host machine. The setup programs will start automatically. Perform the installation by following the messages displayed in the monitor screen.

**Caution** If the setup program does not start automatically, execute **SETUP.EXE** in the **RA78K4\DISK1** folder.

<3> Confirmation of files

Using Windows Explorer, etc., check that the files contained in the RA78K4's supply media have been installed to the host machine.

For the details of each folder, refer to **2.4 Directory Configuration**.

### 2.2.2 Installation of UNIX version

Install the UNIX version with the following procedure. Installation to /necools/bin is assumed here.

<1> Login

Log in to the host machine.

<2> Directory selection

Go to the install directory.

```
%cd /necools/bin
```

<3> Setting of supply media

Set the CD-ROM in the CD-ROM drive.

<4> Copying of files

Execute the cp command to copy the files from the CD-ROM (copy the files after checking that the CD-ROM has been set in the CD-ROM drive).

<5> Setting of environmental variable PATH

Add /necools/bin to the environmental variable PATH.

## 2.3 Installation of Device Files

### 2.3.1 Installation of Windows version

Use the device file installer to install the device files. The device file installer is installed at the same time as the RA78K4.

### 2.3.2 Installation of UNIX version

Either specify the directory for device files with the `-y` option, or specify the directory (example: `-y/nectools/dev`), and copy the device files to a directory with the assembler execution format (example: `/nectools/bin`).

### 2.3.3 Registry registration of device files

If the device files are already installed, a message prompting you to perform registry registration of the device files may be displayed during RA78K4 installation.

If currently using a 32-bit environment, register the device file used for the RA78K4 (Ver. 1.30 or earlier, 16-bit environment) to a registry (32-bit environment).

Registry registration can also be done using the device file installer after RA78K4 installation has been completed.

The registry registration procedure is as follows.

<1> Startup of device file installer

<2> Source selection

Click the [Browse...] button and select "NECDEV.INI" used in the 16-bit environment.

Select a file registered to a registry from the device file displayed in the source list box.

<3> Move

Register the file to the registry (32-bit environment) by clicking the [Move] button.

## 2.4 Directory Configuration

### 2.4.1 Windows version directory configuration

The standard directory displayed during installation is "\NECTools32" on the drive where Windows is installed. The configuration of the install directory is as shown below. The drive and install directory may be changed during installation.

**Figure 2-1. Directory Configuration**

— NECTools32\	
bin\	
ra78k4.exe	Execution format of assembler
st78k4.exe	Execution format of structured assembler preprocessor
lk78k4.exe	Execution format of linker
oc78k4.exe	Execution format of object converter
lcnv78k4.exe	Execution format of list converter
lb78k4.exe	Execution format of librarian
eccgen.exe	Execution format of ECC GENERATOR
lb78k4e.exe	Interface tool between library and DLL of PM plus environment
lb78k4p.exe	Standalone startup library
ra78k4.is*	File used by assembler
*78k4p.dll	DLL tool for PM plus
*78k4.hlp	Help file for starting command line
doc\	User's manual and supplementary explanations
help\	Online manual
setup\	Data files for installation and uninstallation
smp78k4\ra78k4\	
k4main.asm	Assembler sample program
k4sub.asm	Assembler sample program
ra.bat	Batch file for assembler sample program
readme.doc	Explanation of sample program and batch file (text file)
test1.s	Structured assembler sample program
test2.s	Structured assembler sample program
testinc.s	Structured assembler sample program
st.bat	Batch file for structured assembler sample program

**Remark** The explanations in this manual assume installation to the standard directory with the default program folder name "NECTools32" according to the default directions of the setup program.

### 2.4.2 UNIX version directory configuration

The file configuration after installation is as follows. The following assumes installation in /necools/bin.

**Figure 2-2. UNIX Version Directory Configuration**

— necools/	
bin/	
ra78k4	Execution format of assembler
st78k4	Execution format of structured assembler preprocessor
lk78k4	Execution format of linker
oc78k4	Execution format of object converter
lcnv78k4	Execution format of list converter
lb78k4	Execution format of librarian
eccgen	Execution format of ECC GENERATOR
*.hlp	Help file corresponding to each program (text file)
ra78k4.is*	Table file defining instruction set used by assembler
*.asm, *.s	Sample program for installation confirmation
*.sh	Batch file for installation confirmation
readme.doc	Explanation of use of install confirmation shell file (text file)

It is recommended to install the C compiler, integrated debugger, system simulator, and device file to the directory to which the assembler package has been installed.

## 2.5 Uninstallation Procedure

### 2.5.1 Uninstallation of Windows version

The procedure for uninstalling the files installed to the host machine is described below.

- (1) Windows startup  
Power on the host machine and peripherals and start Windows.
- (2) Opening [Control Panel] window  
Press the [Start] button and select [Settings]-[Control Panel] to open the <Control Panel> window.
- (3) Opening of <Add/Remove Programs Properties> window  
Double-click the [Add/Remove Programs] icon in the <Control Panel> window to open the <Add/Remove Programs Properties> window.
- (4) Removing RA78K4  
After selecting "NEC RA78K/4 Assembler Vx.xx" from the list of installed software displayed in the <<Install/Uninstall>> tab in the <Add/Remove Programs Properties> window, click the [Add/Remove...] button.  
When the <System Settings Change> window is opened, click the [Yes ] button.
- (5) Confirmation of files  
Using Windows Explorer, etc., check that the files installed to the host machine have been uninstalled. For the details of each folder, refer to **2.4 Directory Configuration**.

### 2.5.2 Uninstallation of UNIX version

Delete the files copied in **2.2.2 Installation of UNIX version** with the rm command.

## 2.6 Environment Settings

### 2.6.1 Environmental variables

Set the following environmental variables. If the assembler package has been installed using the Windows installer, the necessary environmental variables are automatically set.

PATH	Specifies the directory to which the executable format of the assembler is stored.
TMP	Specifies a directory where a temporary file is to be created (valid only with the PC-9800 Series and IBM PC/AT compatibles)
INC78K4	Specifies a directory where the include file is searched.
LIB78K4	Specifies the directory where a library is searched, if the library is used.
LANG78K	Specifies the kanji code (2-byte code) described in the comment.

#### [Example]

With PC-9800 Series or IBM PC/AT compatibles

```

PATH = %PATH%; C:\NECTools32\bin
set TMP = C:\tmp
set INC78K4 = C:\NECTools32\inc78k4
set LIB78K4 = C:\NECTools32\lib78k4
set LANG78K = SJIS

```

With HP9000 Series 700 or SPARCstation family

```

• Example when csh is used
  set path = ($path /ra78k4)
  setenv INC78K4 /ra78k4
  setenv LIB78K4 /ra78k4
  setenv LANG78K EUC

• Example when sh is used
  PATH = $PATH:/ra78k4
  INC78K4 = /ra78k4
  LIB78K4 = /ra78k4
  export PATH INC78K4 LIB78K4
  setenv LANG78K EUC

```

**Remark** This is an example when files are installed in the directory “/ra78k4”.

If files are installed in “/nectools/bin” as shown in **2.4.2 UNIX version directory configuration**, read “/ra78k4” as “/nectools/bin”.

### 2.6.2 Kanji code in source file

- Kanji (2-byte characters) can be used in specific places (comments, etc.) in the source file.
- Specify the kanji code type using an environmental variable (LANG78K), kanji code control instruction (KANJI CODE), or kanji code specification option (-ZE/-ZS/-ZN).

## CHAPTER 3 EXECUTION PROCEDURE OF RA78K4

This chapter explains the procedures for using the assembler package RA78K4, from assembling to object conversion.

Sample programs 'k4main.asm' and 'k4sub.asm' are assembled, linked, and converted into objects in accordance with the execution procedures explained in this section.

With the PC-9800 Series or IBM PC/AT compatibles, how to run the assembler package on PM plus and the command line is explained.

### 3.1 Before Executing RA78K4

#### 3.1.1 Sample programs

Among the files stored on the system disk are [K4MAIN.ASM] and [K4SUB.ASM]. These files are a sample program for use in verifying the operation of the assembler package.

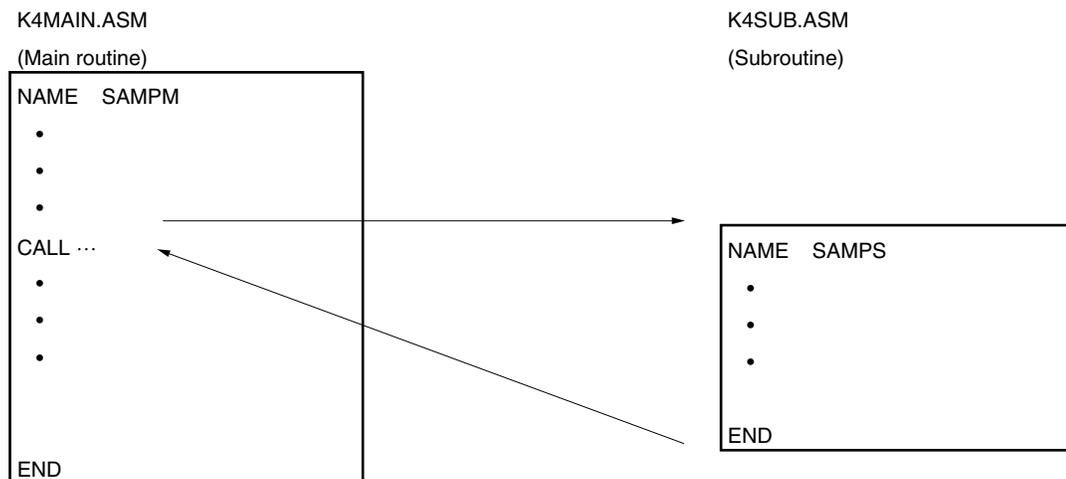
In later assembler operation, these files will be input to the assembler as source program files.

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 (K4MAIN.ASM).

The name of the subroutine module is SAMPS, and it is stored in (K4SUB.ASM).

**Figure 3-1. Structure of Sample Program**



## ■K4MAIN.ASM (Main routine)

```

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

PUBLIC MAIN, START
EXTRN  CONVAH

DATA   DSEG   AT 0FFD20H
HDTSA: DS     1
STASC: DS     2

CODE   CSEG   AT 0H
MAIN:  DW     START

CSEG
LOCATION          15
START: MOV     REM, #00
      MOVG    SP, #0FFE00H
      MOV     MM, #00
      MOV     STBC, #08H

      MOV     HDTSA, #1AH
      MOVG    WHL, #HDTSA           ;set hex 2-code data in WHL register

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

      BR     $$

      END

```

## ■K4SUB.ASM (Subroutine)

```

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: XOR    A, #0
        ROL4   [WHL]          ;hex upper code load
        CALL   $!SASC
        MOV    B,A            ;store result

        MOV    A, #0
        ROL4   [WHL]          ;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 RA78K4. 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) and the register bank selection flags (RBS0, RBS1). The settings for these items are therefore as follows.  
Register bank    0 (0FEF0H to 0FEFFH or 0FFE0H to 0FFEFH)  
RSS flag         0

### 3.1.2 Configuration of sample program

The following describes the sample program that is used as an example for the operations described below.

K4main.asm   Main module  
K4sub.asm    Submodule  
mylib.lib    Library file (this is not used here.)  
sample.dr    Directive file

### 3.2 Execution Procedure of RA78K4

The batch files (ra.bat) in the system disk are used for the RA78K4 operation.

The assembler, linker, object converter, and list converter are executed in this order using "k4main.asm" and "k4sub.asm", which are written in assembly language in ra.bat, as source files. If an error occurs, a message is output and the batch file terminates.

Specification of the type of device to be used as the target is input to this batch file (obtain the device file separately).

The following explanation uses the  $\mu$ PD784026 as the target device.

■ ra.bat (batch program for verifying RA78K4 operation)

```

echo off
cls
set     LEVEL=0

if "%1" == "" goto ERR_BAT

ra78k4 -c%1 k4main.asm
if errorlevel 1 set LEVEL=1
ra78k4 -C%1 k4sub.asm
if errorlevel 1 set LEVEL=1
if %LEVEL% == 1 echo Assemble error !!
if %LEVEL% == 1 goto END

cls
lk78k4 k4main.rel k4sub.rel -s -orasample.lmf -prasample.map
if errorlevel 1 echo Link error !!
if errorlevel 1 goto END

cls
oc78k4 rasample
if errorlevel 1 echo Object conversion error !!
If errorlevel 1 goto END

cls
Set LEVEL=0
lcnv78k4 -lrasample.lmf -rk4main.rel k4main.prn
if errorlevel 1 set LEVEL=1
lcn78k4 -lrasample.lmf -rk4sub.rel k4sub.prn
if errorlevel 1 set LEVEL=1
if #LEVEL% == 1 echo List conversion error !!
if #LEVEL% == 1 goto END

cls
echo No error.
goto END

```

:ERR\_BAT

echo Usage : ra.bat chiptype

:ERR

echo on

**(1) Execute the batch file.**

Specify the target device type and execute the RA78K4-operation verification batch program.

```
C>ra.bat 4026
```

The following message is output to the display.

```
78K/IV Series Assembler Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Pass1 Start
```

```
Pass2 Start
```

```
Target chip : uPD784026
```

```
Device file : Vx.xx
```

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

```
78K/IV Series Assembler Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Pass1 Start
```

```
Pass2 Start
```

```
Target chip : uPD784026
```

```
Device file : Vx.xx
```

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

Clear the screen.

```
78K/IV Series Linker Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Target chip : uPD784026
```

```
Device file : Vx.xx
```

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

Clear the screen.

```
78K/IV Series Object Converter Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Target chip : uPD784026
Device file : Vx.xx
```

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

Clear the screen.

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

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

```
List Conversion Program or RA78K/IV Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

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

Clear the screen.

No error.

**(2) Check the contents of drive C.**

The following files are output.

K4main.rel:	Object module file
K4main.prn:	Assemble list file
K4sub.rel:	Object module file
K4sub.prn:	Assemble list file
rasample.lmf:	Load module file
rasample.map:	Link list file
rasample.hex:	HEX format object module file
rasample.sym:	Symbol table file
k4main.p:	Absolute assemble list file
k4sub.p:	Absolute assemble list file

**(3) Summary of RA78K4 execution procedure**

The following is a brief summary of the execution procedure of the RA78K4.

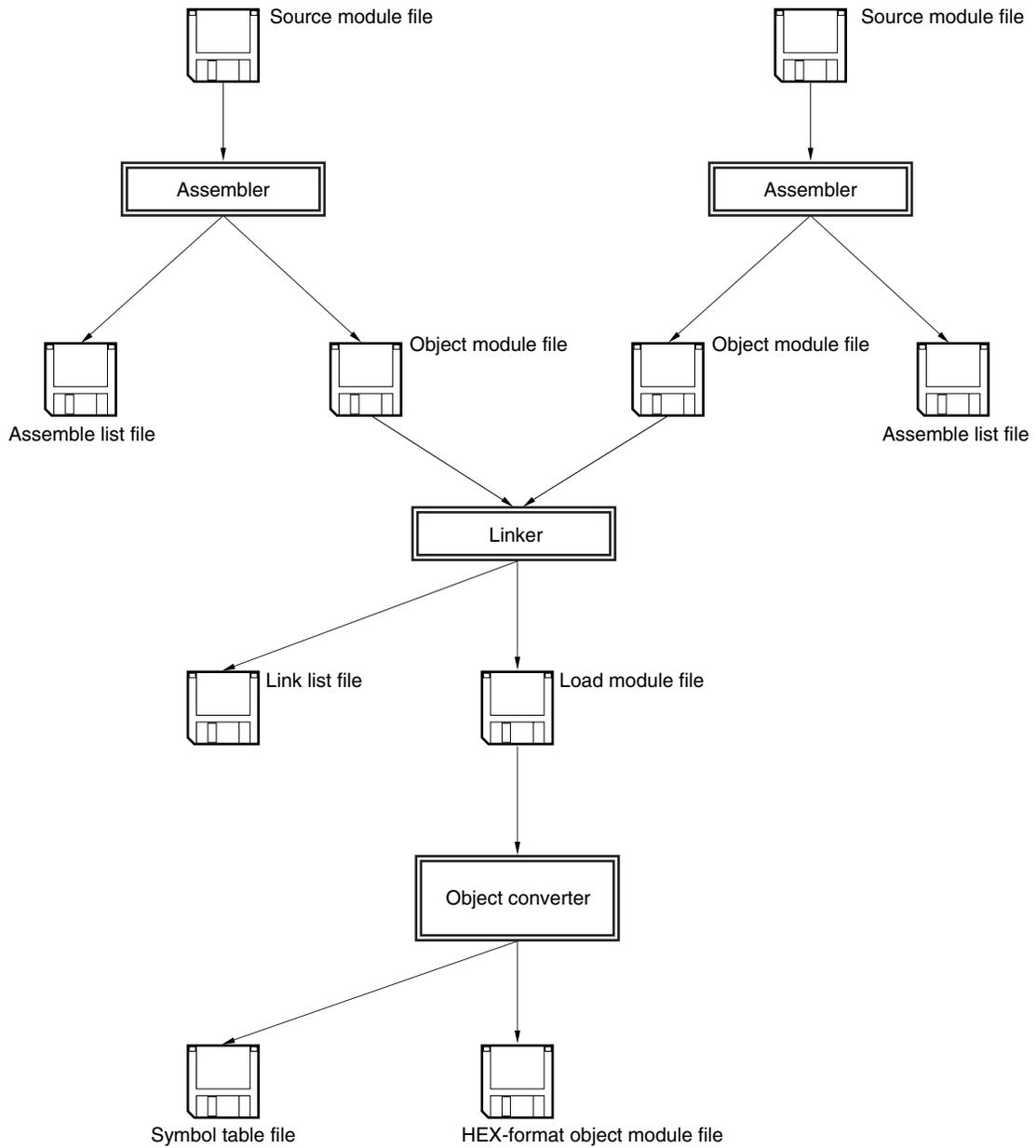
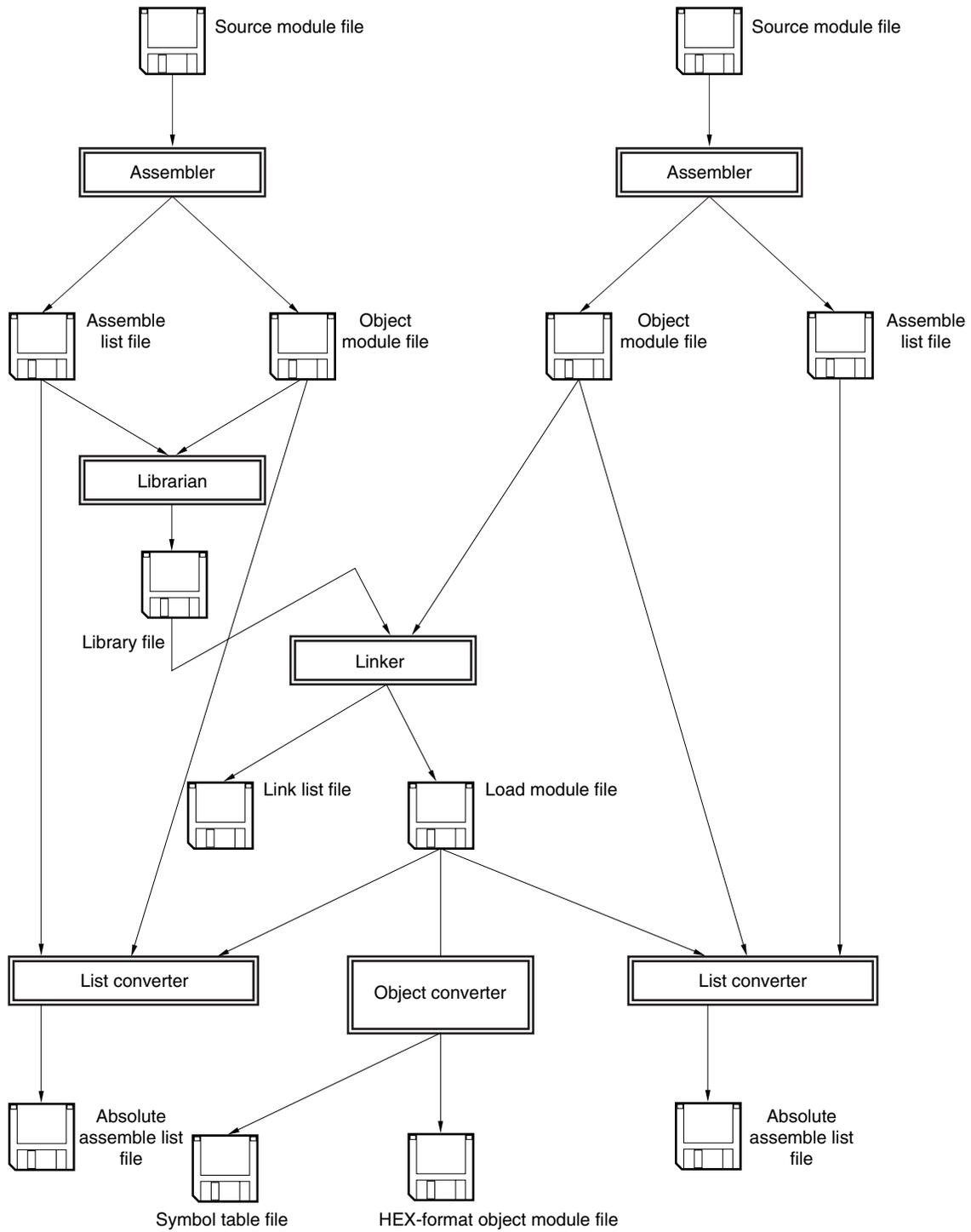
**Figure 3-2. RA78K4 Execution Procedure 1**

Figure 3-3. RA78K4 Execution Procedure 2



### 3.3 Execution Procedure of ST78K4

To verify the operation of the ST78K4, use the batch file (st.bat) stored on the system disk.

The structured assembler, assembler, linker, object converter, and list converter are executed in order using the sample programs "test1.s" and "test2.s", which are written in structured assembly language in st.bat, as source files. The batch file then terminates following output of any error messages.

Specification of the type of the target device can be input to this batch file (obtain the device file separately).

The following explanation uses the  $\mu$ PD784026 as the target device.

#### ■st.bat (ST78K4-operation verification batch program)

```

echo off
cls
set      LEVEL=0

if "%1" == "" goto ERR_BAT

st78k4 -C%1 test1.s
ra78k4 test1.asm
if errorlevel 1 set LEVEL=1
st78k4 -C%1 test2.s
ra78k4 test2.asm
if errorlevel 1 set LEVEL=1
if %LEVEL% == 1 echo Assemble error !!
if %LEVEL% == 1 goto END

cls
lk78k4 test1.rel test2.rel -s -ostsample.lmf -pstsampl.map
if errorlevel 1 echo Link error !!
if errorlevel 1 goto END

cls
oc78k4 test
if errorlevel 1 echo Object conversion error !!
if errorlevel 1 goto END

cls
set LEVEL=0
lcnv78k4 -lstsample.lmf -rtest1.rel test1.prn
if errorlevel 1 set LEVEL=1
lcnv78k4 -lstsample.lmf -rtest2.rel test2.prn
if errorlevel 1 set LEVEL=1
if %LEVEL% == 1 echo List conversion error !!
if %LEVEL% == 1 goto END

cls
echo No error.
goto END

:ERR_BAT

echo Usage : st.bat chiptype

:END

echo on

```

**(1) Executing the batch file**

Specify the target device type and execute the ST78K4-operation verification batch program.

```
C>st.bat 4026
```

The following message will be output.

```
Structured assembler preprocessor for RA78K/IV Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
start
```

```
Target chip : uPD784026
Device file : Vx.xx
```

```
Conversion complete, 0 error(s) found.
```

```
78K/IV Series Assembler Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Pass1 Start
```

```
Pass2 Start
```

```
Target chip : uPD784026
Device file : Vx.xx
```

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

```
Structured assembler preprocessor for RA78K/IV Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
start
```

```
Target chip : uPD784026
Device file : Vx.xx
```

```
Conversion complete, 0 error(s) found.
```

```
78K/IV Series Assembler Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Pass1 Start
```

```
Pass2 Start
```

```
Target chip : uPD784026
Device file : Vx.xx
```

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

Clear the screen.

```
78K/IV Series Linker Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Target chip : uPD784026
Device file : Vx.xx
```

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

Clear the screen.

```
78K/IV Series Object Converter Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Target chip : uPD784026
Device file : Vx.xx
```

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

Clear the screen.

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

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

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

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

Clear the screen.

No error.

**(2) Check the contents of drive C.**

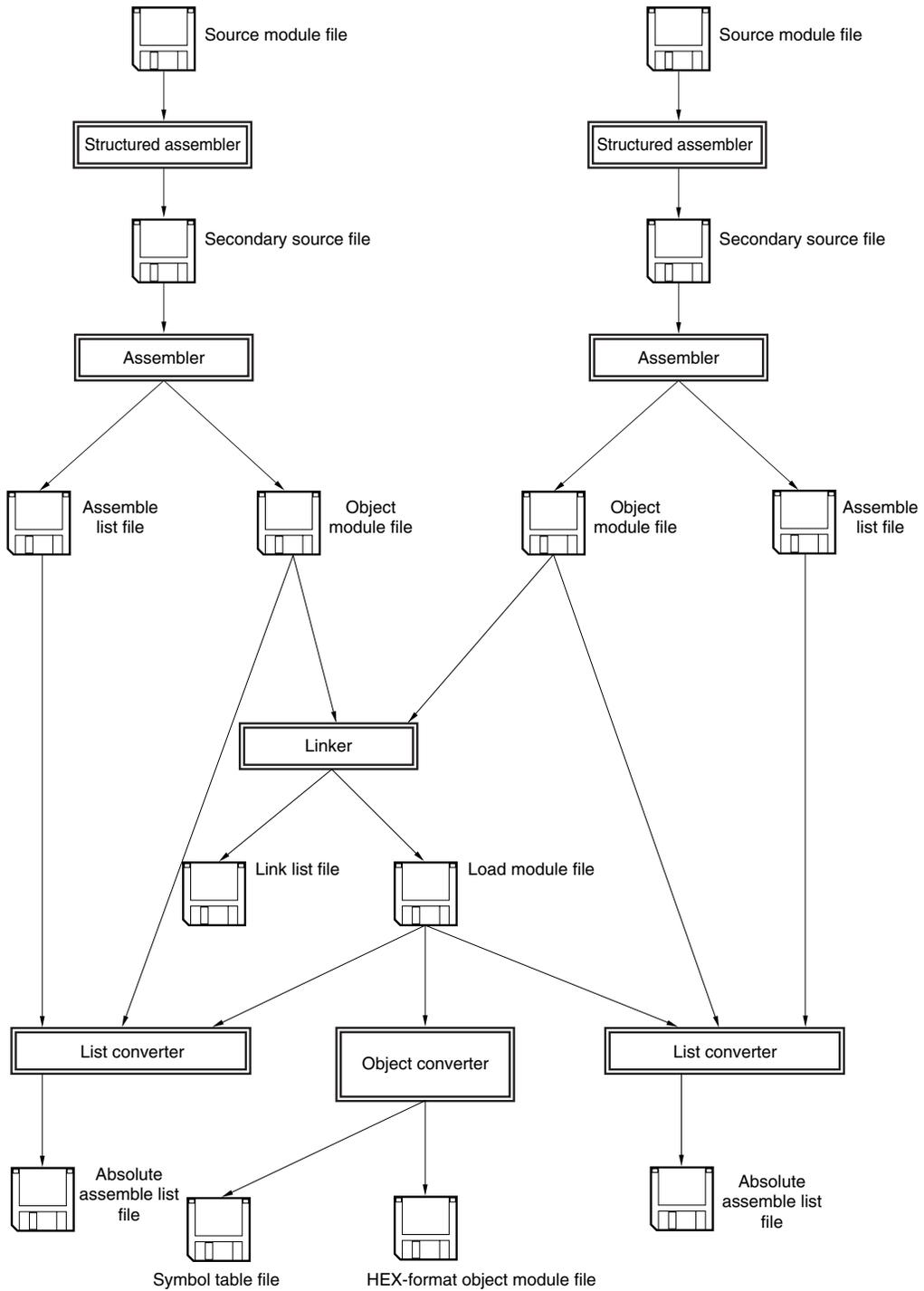
The following files will be output.

test1.asm: Secondary source file  
test1.rel: Object module file  
test1.prn: Assemble list file  
test2.asm: Secondary source file  
test2.rel: Object module file  
test2.prn: Assemble list file  
stsample.lmf: Load module file  
stsample.map: Link list file  
stsample.hex: HEX-format object module file  
stsample.sym: Symbol table file  
test1.p: Absolute assemble list file  
test2.p: Absolute assemble list file

(3) Summary of ST78K4 execution procedure

The following is a summary of the execution procedure of the ST78K4.

Figure 3-4. ST78K4 Execution Procedure



### 3.4 Assembling, Linking, and Object Conversion from Command Line (DOS Prompt, EWS)

This section explains how to execute assembly and object conversion from the command line.

#### (1) Assemble the sample program K4MAIN.ASM.

Input the following on the command line.

```
C><ra78k4 -c4026 k4main.asm
```

The following message is output to the display.

```
78K/IV Series Assembler Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx, xxxx

Pass1 Start
Pass2 Start

Target chip : uPD784xxx
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 (K4MAIN.REL) and the assemble list file (K4MAIN.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 K4SUB.ASM. Input the following on the command line.

```
C><ra78k4 -c4026 k4sub.asm
```

The following message is output to the display.

```
78K/IV Series Assembler Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx, xxxx

Pass1 Start
Pass2 Start

Target chip : uPD784xxx
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 (K4SUB.REL) and the assemble list file (K4SUB.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.

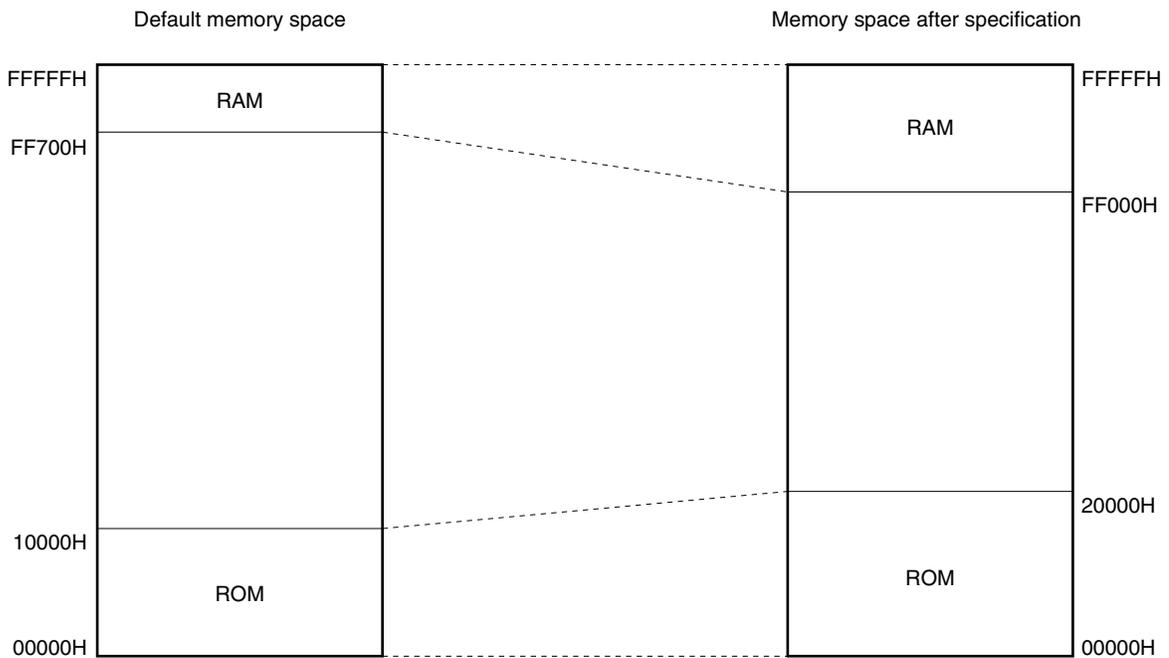
During linking, use the -D option to enter the directive file to the linker.

**Example** In this example, the default ROM/RAM area is expanded by the  $\mu$ PD784026 and segment CSEG1 is located to address 2000H.

The following is described in the directive file (when LOCATION 0FH is executed)

```
MEMORY ROM: (0H, 20000H)
MEMORY RAM: (0FF000H, 10000H)
MERGE CSEG1: AT (2000H)
```

**Figure 3-5. Link Directive**



**(6) As the result of the assembly, the output object module files [K4MAIN.REL] and [K4SUB.REL] are linked.**

Enter K4.DR as the directive file.

Enter the following on the command line.

```
C>lk78k4 k4main.rel k4sub.rel -dk4.drNote 1 -ok4.lmf -pk4.map -SNote 2
```

**Notes 1.** Not necessary if a directive file is not specified.

**2.** Stack resolution symbol (\_@STBEG) creation option

The following message is output to the display.

```
78K/IV Series Linker Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
  Target chip : uPD784xxx
  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 (K4.lmf) and the link list file (K4.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 (K4.lmf) is converted to a HEX-format file.**

Enter the following on the command line.

```
C>oc78k4 k4.lmf
```

The following message is output on the display.

```
78K/IV Series Object Converter Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

  Target chip : uPD784xxx
  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 (K4.HEX) and the symbol table file (K4.SYM).

**(10) Create a library file as follows.**

Register the object module file (K4SUB.REL) output by the assembler as a library file.

Enter the following on the command line.

```
C><lb78k4 < k4.job
```

The following message is output on the display.

```
78K/IV Series Librarian Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
*create k4.lib
*add k4.lib k4sub.rel
*exit
```

**Remark** The parts underlined above are the contents of "k4.job."

**(11) Check the contents of drive C.**

The librarian outputs the library file (K4.LIB).

**(12) Create an absolute assemble list as follows.**

To create the absolute assemble list K4MAIN.ASM, input [K4MAIN.REL], [K4MAIN.ASM] and [K4.lmf] to the list converter.

Enter the following on the command line.

```
C><lcnv78k4 k4main -lk4.lmf
```

The following message is output on the display.

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

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

**(13) Check the contents of drive C.**

The list converter outputs the absolute assemble list file (K4MAIN.P).

### 3.5 Using Parameter File

If two or more options are input when the assembler or linker is started, the information necessary for starting cannot be completely specified on the command line, or the same specification may be repeatedly made. In this case, the parameter file is used.

To use the parameter file, specify the parameter file specification option on command line.

**Caution** The parameter file cannot be specified by the PM plus option setting.

Assembler or linker is started by the parameter file as follows:

```
> [path-name] RA78K4 Δ -F parameter-file-name
> [path-name] LK78K4 Δ -F parameter-file-name
> [path-name] OC78K4 Δ -F parameter-file-name
```

Here is an example of its use.

```
Example C>ra78k4 -Fpara.pra
          C>lk78k4 -Fpara.plk
          C>oc78k4 -Fpara.poc
```

The parameter file is created with an editor. All the options and output file names that should be specified on the command line can be written in the parameter file.

Here is an example of creating a parameter file with an editor.

(Contents of para1.pra)

```
-c4038 k4main.asm -e
```

(Contents of para1.plk)

```
k4main.rel k4sub.rel -bmylib.lib -osample.Imf -s
```

(Contents of para1.poc)

```
sample.lmf -u0FFH -osample.hex -r
```

## CHAPTER 4 STRUCTURED ASSEMBLER

The structured assembler inputs source module files written in the structured assembly language of 78K4 Series microcontrollers, converts them into assembly language, and outputs them as secondary source module files.

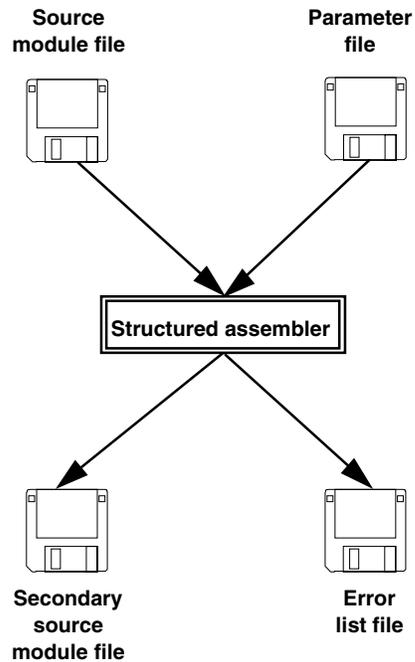
### 4.1 I/O Files of Structured Assembler

The I/O files of the structured assembler are as shown below.

**Table 4-1. I/O Files of Structured Assembler**

Type	File Name	Explanation	Default File Type
Input files	Source module files	These are source module files written in structured assembly language. These files are created by the user.	None
	Parameter files	These are files that contain the options for specifying the structured assembler options from the file. These files are created by the user.	.PST
Output files	Secondary source module files	These are source module files written in assembly language.	.ASM
	Error list file	These are files containing structured assembler error data.	.EST

**Figure 4-1. I/O Files of Structured Assembler**



**Caution** A parameter file cannot be specified when PM plus is used.

## 4.2 Functions of Structured Assembler

- (1) The structured assembler reads source module files, converts them into assembler input source files, and outputs them as assembler source files.
- (2) If an error related to the file or system occurs, the structured assembler outputs an abort error, and if a write error is found in the source module, it outputs a fatal error or warning error.  
If an abort error or fatal error occurs, the secondary source file cannot be output normally. However, when the -J option has been specified, the secondary source file can be output even if a fatal error has occurred.
- (3) Structured assembly processing is performed in accordance with the option specified at startup. For a detailed explanation of the structured assembler options, refer to **4.4 Structured Assembler Options**.
- (4) If the structured assembly processing has been completed correctly, the structured assembler outputs a “completed” message, and returns control to the OS.

## 4.3 Structured Assembler Startup

### 4.3.1 Structured assembler startup

Two methods can be used to start up the structured assembler.

#### (1) Command-line startup

Start up the structured assembler by inputting the following command.

```
X> [path-name] st78k4 [ $\Delta$ option] . . .  $\Delta$ source-module-file-name [ $\Delta$ option] . . .  
|           |           |           |           |           |  
(1)        (2)        (3)        (4)        (5)        (4)
```

- (1) Current drive name
- (2) Current directory name
- (3) Structured assembler command file name
- (4) Enter detailed instructions for the operation of the structured assembler. When specifying more than one option, delimit the options with a space. For a detailed explanation of the structured assembler options, refer to **4.4 Structured Assembler Options**.
- (5) File name of source module to undergo structured assembly.

**Example** C>st78k4 -c4038 test1.s -e

**(2) Startup from a parameter file**

Use a parameter file to avoid the inconvenience involved when repeating the same structured assembler option at startup for two or more structured assembly operations.

To start up the structured assembler from a parameter file, specify the parameter file specification option (-F) on the command line.

The procedure for starting up the structured assembler from a parameter file is shown below.

```
X> [path-name] st78k4 [Δsource-module-file-name] Δ-F parameter-file-name
|         |                               |         |
(1)      (2)                             (3)      (4)
```

- (1) Current drive name
- (2) Current directory name
- (3) Parameter file specification option
- (4) Parameter file name

The rules for writing 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 one source module file name can be specified in the parameter file.

The source file name can also be written after the option.

Write in the parameter file all options and output file names specified in the command line.

**Example** Creating a parameter file (test1.pst) using the editor.

- Contents of test1.pst

```
;Parameterfile
test1.s -osample.asm
-esample.est -c4038
```

- The parameter file (test1.pst) is used to start up the structured assembler.

```
C>st78k4 -ftest1.pst
```

### 4.3.2 Execution start and end messages

#### (1) Execution start message

When the structured assembler is activated, an execution start message appears on the display.

```
Structured assembler preprocessor for RA78K/IV Vx.xx [xx xxx xx]
Copyright (C) NEC Electronics Corporation xxxx
```

#### (2) Processing display message

“.” is displayed every 100 lines of the structured assembler processing.

```
Start.....
```

#### (3) Execution end message

If no errors are detected, the following message is output to the display, and control is returned to the OS.

```
Target chip:uPD78xxxx
Device file:Vx.xx
```

```
Conversion complete, 0 error (s) found.
```

If errors are detected, the number of detected errors is output to the display, and control is returned to the OS.

```
TEST1.S (8) :F209 Syntax error
```

```
Target chip:uPD78xxxx
Device file:Vx.xx
```

```
Conversion complete, 1 error (s) found.
```

If a fatal error is detected during structured assembly which makes it impossible to continue structured assembly processing, the structured assembler outputs a message to the display, cancels the structured assembly processing, and returns control to the OS.

#### Example 1. When a non-existent source module file is specified

```
C>>st78k4 sample.s
```

```
Structured assembler preprocessor for RA78K/IV Vx.xx [xx xxx xx]
Copyright (C) NEC Electronics Corporation xxxx
```

```
A006 File not found 'SAMPLE.S'
```

```
Program aborted.
```

In the above example, the specification of a non-existent source module file results in an error, and assembly is stopped.

**Example 2.** When a non-existent option is specified

```
C>st78k4 test1.s -z
```

```
Structured assembler preprocessor for RA78K/IV Vx.xx [xx xxx xx]  
Copyright (C) NEC Electronics Corporation xxxx
```

```
A012 Missing parameter '-z'  
Please enter 'ST78K4 --', if you want help messages.
```

```
Program aborted.
```

In the above example, the specification of a non-existent option results in an error, and assembly is stopped.

When an error message is displayed and assembly is stopped, search for the cause of the error in **CHAPTER 13 ERROR MESSAGES** and take action accordingly.

## 4.4 Structured Assembler Options

### 4.4.1 Types of structured assembler options

The structured assembler options give detailed instructions for the operation of the structured assembler.

The structured assembler options are classified into the following 13 types.

**Table 4-2. Structured Assembler Options**

Number	Classification	Option	Description
1	Device type specification	-C	Specifies the type of the target device
2	Word symbol character specification	-SC	Specifies the final character of the word symbol name
3	Symbol definition	-D	Specifies the symbol given to the #IFDEF directive, etc.
4	Tab number specification	-WT	Specifies the output position of the converted instruction.
5	Include file path specification	-I	Specifies the drive and directory of the include file.
6	Secondary source file specification	-O	Specifies the secondary source file name.
7	Error list file specification	-E	Specifies the error list file name.
8	Parameter file specification	-F	Specifies the parameter file name.
9	Debug data output specification	-GS	Specifies the output of structured-assembler source-level debug data.
		-NGS	
10	Secondary source file forcible output specification	-J	Specifies the forcible output of the secondary source file.
11	Kanji code (2-byte code) specification	-ZS	Specifies the kanji code type to be described in the comment statement.
		-ZE	
		-ZN	
12	Device file search path specification	-Y	Specifies the path via which the device file will be searched
13	Help specification	--	Outputs the help message to the display.

#### 4.4.2 Explanation of structured assembler options

This section contains a detailed explanation of each structured assembler option.

##### (1) Device type specification (-C)

Syntax:               -C device-type

Default assumption: Cannot be omitted

##### [Function]

Specifies the device that is the target of structured assembly.

##### [Explanation]

- 1) Always specify the -C option. The structured assembler performs preprocessing on the specified target device and generates the assembler source code.  
Note that if the -C option is omitted, an error will occur.
- 2) If the device types specified by the -C option and by the processor device type specification control instruction differ, a warning is issued. In this case, the structured assembler will prioritize the device type specified by the -C option.
- 3) The device type specified by the -C option is output to the secondary source file as a processor device type specification control instruction. However, this does not occur if a device type with the same name as a processor device type specification control instruction is specified.

##### [Example of use]

The  $\mu$ PD784038 is specified as the target device.

```
C>st78k4 test.s -c4038
```

##### [Notice]

The -C option cannot be omitted. However, if a processor device type specification control instruction (\$PROCESSOR) is written at the top of the source file, specification in the command line can be omitted. Refer to **Notes on Use** in the device file of each device for details concerning device types.

**(2) Word symbol character specification (-SC)**

Syntax: -SC character

Default assumption: P or p

**[Function]**

Specifies the final character of the symbol that is the target of judgment in cases when bytes/words must be differentiated a symbol name.

**[Explanation]**

- 1) The structured assembler generates different instructions depending on whether the data to be handled is a byte or a word.  
If it is a substitution, the MOV instruction is generated for a byte and the MOVW for a word.  
If it is a word symbol reserved word, a word instruction is generated.
- 2) If a symbol that is not a reserved word is specified, the symbol is judged to be either a byte symbol or a word symbol based on its final character, and an instruction is generated.
- 3) If the -SC option is not specified, a symbol ending with 'P' or 'p' is judged to be a word symbol.
- 4) Characters to be judged are alphabet-equivalent characters only. Note that alphabet letters are not case sensitive.
- 5) If more than one specification is made, the item specified last is valid.

**[Example of use]**

A symbol ending with @ is specified as a word symbol.

```
C>st78k4 test.s -sc@
```

```
<test.s>
```

```
A = #3
AX = #3
SYM = #3
SYM@ = #3
```

```
<test.asm>
```

```
MOV A, #3 ;A = #3
MOVW AX, #3 ;AX = #3
MOV SYM, #3 ;SYM = #3
MOVW SYM@, #3 ;SYM@ = #3
```

**(3) Symbol definition specification (-D)**

Syntax:                -D symbol-name [=numerical-value] [, symbol-name [=numerical-value]...]

**[Function]**

Defines the symbol.

**[Explanation]**

- 1) The numerical value given to a symbol can be binary, octal, decimal, or hexadecimal.  
If the numerical value specification is omitted, the value becomes 1.
- 2) Defining a symbol using this option is identical to defining a symbol using the #define directive.
- 3) Up to 30 items can be defined in the command line by using commas as delimiters.
- 4) This option is usually used in combination with the #ifdef directive.
- 5) If more than one specification is made, the item specified last is valid.
- 6) If this option is specified together with the #define directive, a warning message is output and the #define directive is taken as valid.
- 7) Alphabet letters are not case sensitive.

**[Example of use]**

The symbol "TRUE" is defined as 1.

```
C>>st78k4 test.s -dTRUE=1
```

**(4) Tab number specification (-WT)**

Syntax:                -WT numerical-value-1  
                          -WT [numerical-value-1], numerical-value-2  
                          -WT [numerical-value-1], [numerical-value-2], numerical-value-3  
Default assumption: -WT2, 3, 4

**[Function]**

Specifies the number of tabs of the converted assembly language.

**[Explanation]**

- 1) The -WT option allows the output position of the assembler source instructions to be freely adjusted, thus improving the program's readability.
- 2) Numerical value 1 specifies the number of tabs until the instruction is output.  
Numerical value 2 specifies the number of tabs until the instruction operand is output.  
Numerical value 3 specifies the number of tabs until the instruction command is output.
- 3) Specify the numerical value as a decimal number from within the following ranges.  
Numerical value 1: 0 to 97  
Numerical value 2: 1 to 98  
Numerical value 3: 2 to 99  
Numerical value 1 < numerical value 2 < numerical value 3
- 4) If more than one specification is made, the item specified last is valid.

**[Example of use]**

"3" is specified for numerical value 1, "4" for numerical value 2, and "5" for numerical value 3.

```
C>st78k4 test.s -wt3,4,5
```

**(5) Include file path specification (-I)**

Syntax:                -I [drive:] directory

Default assumption: Current directory

**[Function]**

Specifies the name of the include file path to be input to the structured assembler.

**[Explanation]**

- 1) Specify a drive and directory in which the include file exists.
- 2) If the -I option is omitted, the include file will be assumed to be in the current drive and current directory.
- 3) If more than one specification is made, the item specified last is valid.

**[Example of use]**

The directory with the include file is specified as c:\include.

```
C>st78k4 test.s -ic:\include
```

**(6) Secondary source file specification (-O)**

Syntax:               -O [[[drive:] directory] file-name]

Default assumption: -O input-file-name.ASM

**[Function]**

Specifies the output destination of the post-conversion secondary source file and the file name.

**[Explanation]**

- 1) Specify the output drive, directory, and file name of the post-conversion secondary source file.
- 2) If the -O option is omitted, the output file is created in the current directory by replacing the file type of the input file with .ASM.
- 3) Either "NUL" or "AUX" can be specified as the file name.
- 4) The secondary source file is not output when processing is stopped due to a fatal error.
- 5) If more than one specification is made, the item specified last is valid.

**[Example of use]**

"sample.asm" is specified as the secondary source file.

```
C>st78k4 test.s -osample.asm
```

**(7) Error list file specification (-E)**

Syntax: -E [[drive:][directory] file-name]

Default assumption: -E input-file-name.EST

**[Function]**

Specifies the output destination of the error list file and the file name.

**[Explanation]**

- 1) Specify the output drive, directory, and file name of the error list file.
- 2) If the -E option is omitted, the error list file is created in the current directory by replacing the file type of the input file with .EST.
- 3) Either "NUL" or "AUX" can be specified as the file name.
- 4) If more than one specification is made, the item specified last is valid.

**[Example of use]**

"sample.est" is specified as the error list file.

```
C>st78k4 test.s -esample.est
```

**(8) Parameter file specification (-F)**

Syntax:               -F [[drive:] directory] file-name

**[Function]**

Specifies the file name of the parameter file.

**[Explanation]**

- 1) Specify the input drive, directory, and file name of the parameter file.
- 2) The file name cannot be omitted. If the file type is omitted, the type is assumed to be “.PST”.
- 3) This option is effective when a large number of symbols are defined in the command line using option -D.
- 4) Multiple specification of this option results in an error.
- 5) Parameter-file nests are prohibited, and their specification results in an error.
- 6) The characters following “;” or “#” in a parameter file are all assumed to be comments, up to LH or EOF.

**[Example of use]**

“sample.pst” is specified as a parameter file.

```
C>st78k4 -fsample.pst
```

**(9) Debug data output specification (-GS/-NGS)**

Syntax:                -GS  
                          -NGS

Default assumption: -GS

**[Function]**

Specifies the output of structured-assembler source-level debug data.

**[Explanation]**

- 1) The -GS option specifies the output of debug data to the secondary source file.
- 2) The -NGS option disables the -GS option setting.
- 3) If there is compiler debug data in the input source file, the -GS option replaces “\$” with “;” at the top of the file.
- 4) If the -GS and -NGS options are both specified, the option specified later is taken as valid.
- 5) If omitted, the -GS option is assumed to have been specified.

**[Notice]**

When debugging at the structured assembler source level, be sure to specify the debug data output specification (-GS/-NGS). When assembling the secondary source file, be sure to specify the debug data output specification option (-G/-GA). The structured assembler outputs the required option to the secondary source file as a control instruction.

**[Example of use]**

The output of debug data to the secondary source file is specified.

```
C>st78k4 test.s -gs
```

**(10) Secondary source file forcible output specification (-J)**

Syntax:                -J

**[Function]**

Forcibly outputs the secondary source file when processing is stopped due to a fatal error.

**[Explanation]**

- 1) The secondary source file is forcibly output when processing is stopped due to a fatal error.
- 2) The fatal error line outputs the image of the input source file as is to the secondary source file.

**[Example of use]**

Forcible output of the secondary source file is specified.

```
C>>st78k4 test.s -j
```

**(11) Kanji code specification (-ZS/-ZE/-ZN)**

Syntax: -ZS  
          -ZE  
          -ZN

Default assumption: Interpreted as follows depending on the OS.

Windows, HP-UX: -ZS  
SunOS, Solaris: -ZE

**[Function]**

Specifies the type of the kanji code to be described in the comment.

**[Explanation]**

- 1) Specify the kanji code as follows
  - ZS: Interpreted as shift JIS code.
  - ZE: Interpreted as EUC code.
  - ZN: Not interpreted as kanji.
- 2) These options correspond to the kanji code specification control instructions as follows.
  - ZS: \$KANJI CODE SJIS
  - ZE: \$KANJI CODE EUC
  - ZN: \$KANJI CODE NOTE
- 3) The priority order of specifications of the kanji code is as follows.
  - (1) Specification of -ZS/-ZE/-ZN option
  - (2) Specification of the kanji code specification control instruction (\$KANJI CODE)
  - (3) Specification of the environmental variable LANG78K
  - (4) Specification of default settings of each OS

**[Example of use]**

It is specified that the kanji is interpreted as shift JIS code.

```
A><u>st78k4 test.s -zs
```

**(12) Device file search path specification (-Y)**

Syntax: -Y [drive:] directory

Default assumption: Default assumption: The device file will be searched in the following order.

- 1) <..\dev> (for the path that activated st78k4.exe)
- 2) The path that started up st78k4.exe
- 3) The current path
- 4) The path that was specified by the environment variable PATH

**[Function]**

Specifies the path via which a device file will be searched.

**[Explanation]**

- 1) The device file is read from the specified path
- 2) If other than a path name is specified, an error will occur.
- 3) Even if the directory specification symbol<sup>Note</sup> has not been written on the end of the directory, the assumption will be that it has.
- 4) The device file will be searched in the following order.
  1. The path specified by the -Y option
  2. <..\dev> (for the path that activated st78k4.exe)
  3. The path that activated st78k4.exe
  4. The current path
  5. The path that was specified by the environment variable PATH

**Note** “¥” in the PC-9800 Series and “\” in IBM PC/AT and compatibles.

**[Example of use]**

It is specified that the device file be read from the directory c: \nectools32\dev.

```
C>st78k4 test.s -yc:\nectools32\dev
```

**(13) Help specification (--)**

Syntax:                --

Default assumption: No display

**[Function]**

The -- option displays the help message on the screen.

**[Application]**

The help message is a list of structured assembler options and their definitions. Refer to this when executing the structured assembler.

**[Explanation]**

When the -- option is specified, all other structured assembler options become invalid.

**Caution** This option cannot be specified from PM plus.

To reference the PM plus help, click the Help button in the <Structured Assembler Options> dialog box.

**[Example of use]**

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

```
C>st78k4 --
```

```
Structured assembler preprocessor for RA78K/IV Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx
```

```
Usage:st78k4 [option[...]] input-file [option[...]]
```

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

```
-cx          :Select target chip. (x = 000, 012, etc.) *Must be specified.
-o[file]    :Create the assembler source file[with the specified name].
-e[file]    :Create the error list file[with the specified name].
-ffile     :Input options or source file name from specified file.
-idirectory :Set include search path.
-sc[character]:Specify the last character of word symbol.
-wtn1/-wt[n1], n2/-wt[n1], [n2], n3
            :Specify the number of tabs up to output position of each field.
            n1:Output position mnemonic field.
            n2:Output position operand field. *Must be
            n3:Output position comment field. 0<= n1<n2<n3<100.
-dname[=data][,name[=data][...]]
            :Define name[with data].
-gs/-ngs   :Output the structured assembler source debug information to
            assembler source file / Not.
-j         :Create the assembler source file if fatal error occurred.
```

Press RETURN to continue...

```
-ydirectory :Set device file search path.
--         :Show this message.
```

```
DEFAULT ASSIGNMENT:-o -e -scp -wt2,3,4 -gs
```

## 4.5 Setting Options from PM plus

This section will explain how to set up the structured assembler from PM plus.

### 4.5.1 Setting options

Selecting [Structured Assembler Options...] from the [Tools] menu in PM plus will cause the <Structured Assembler Options> dialog box to appear.

By entering the required options in this dialog box, the structured assembler options can be set.

**Figure 4-2. <Structured Assembler Options> Dialog Box (When <<Output>> Tab Is Selected)**

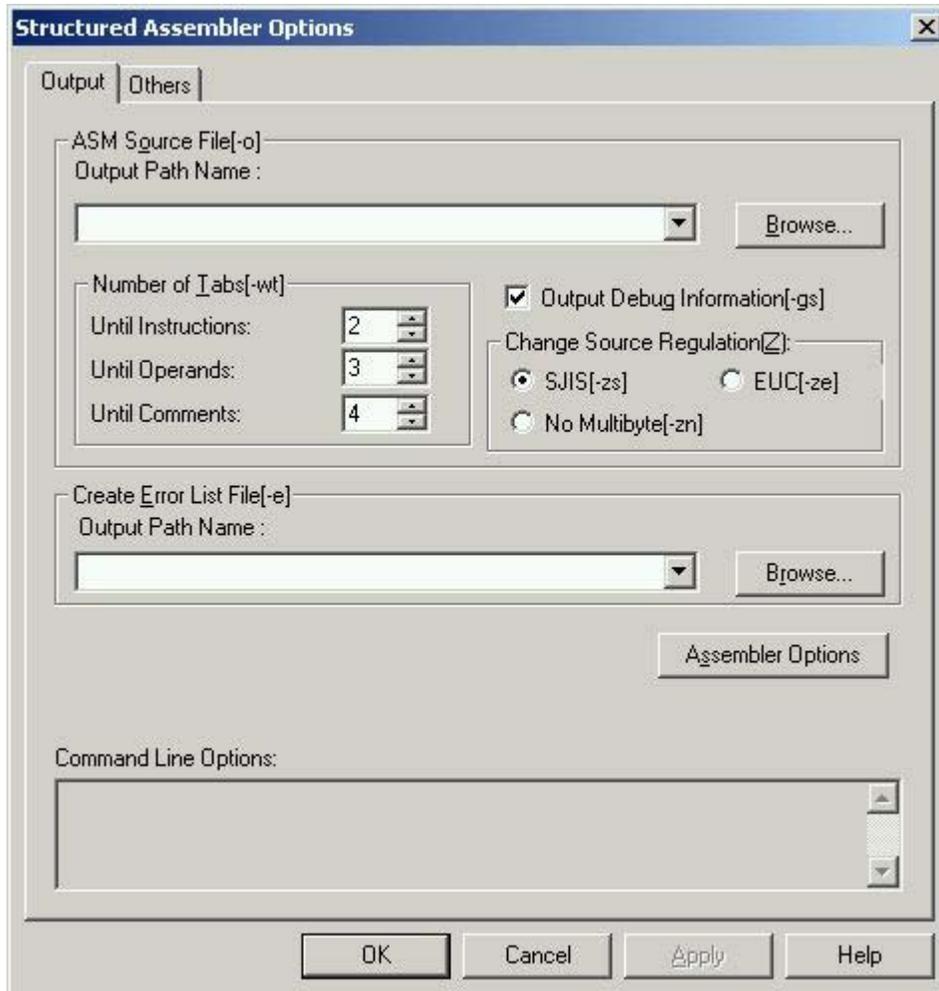


Figure 4-3. &lt; Assembler Source Options&gt; Dialog Box

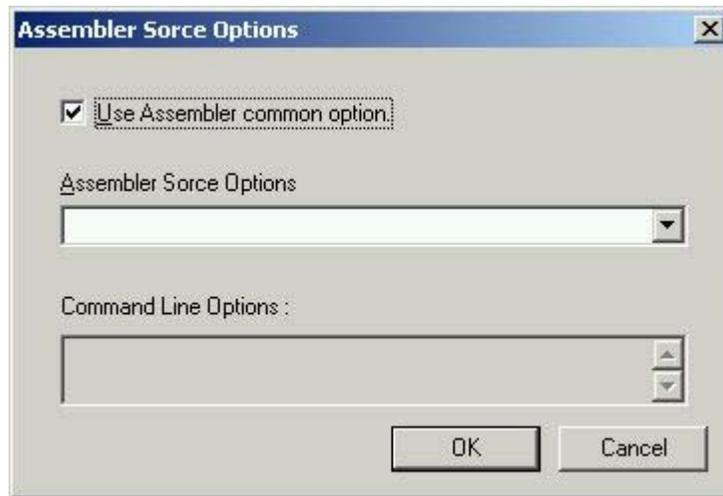
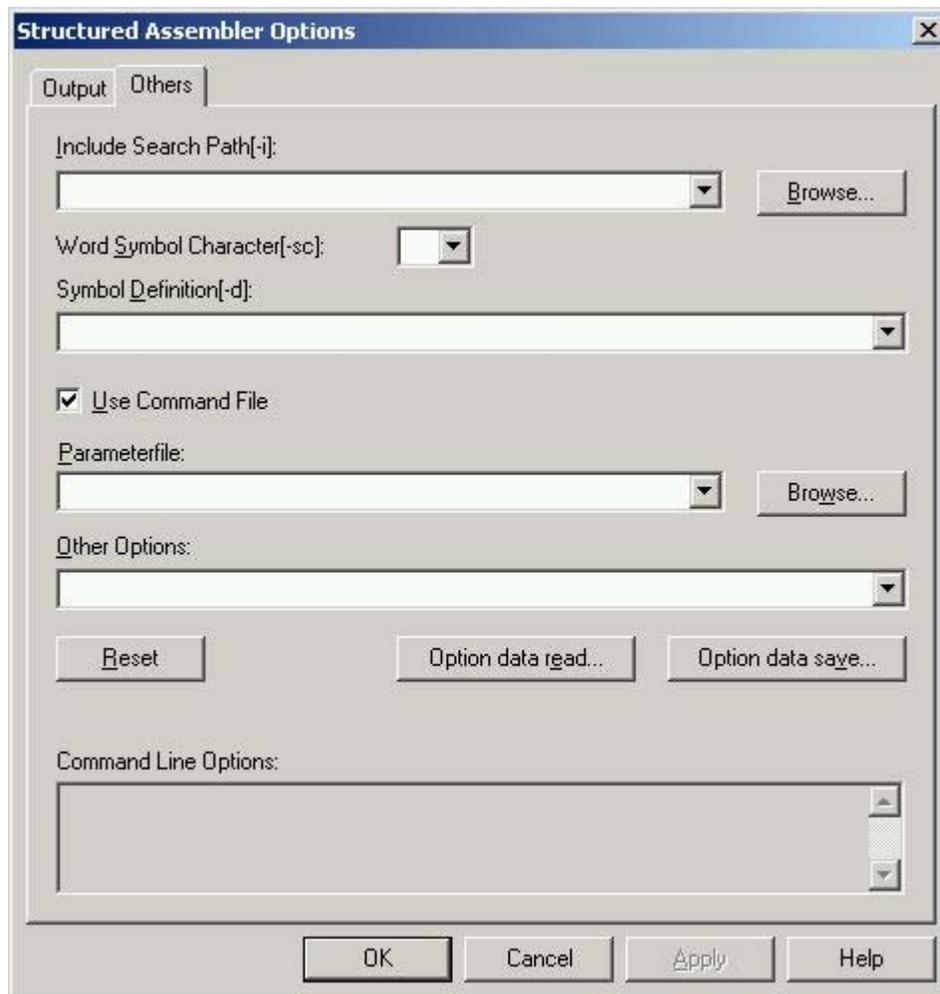


Figure 4-4. &lt;Structured Assembler Options&gt; Dialog Box (When &lt;&lt;Others&gt;&gt; Tab Is Selected)



### 4.5.2 Options

The following is an explanation of each option in the <Structured Assembler Options> dialog box.

- **ASM Source File [-o]**  
Output Path Name  
Specify the ASM source output path either using the [Browse...] button or by directly inputting it.
- **Number of Tabs [-wt]**  
Specify the number of tabs until the converted assembly language is output.  
The number of tabs before an instruction, before an operand, and before a comment can be specified separately.
- **Output Debug Information [-gs]**  
Output the debug information to a secondary source file.
- **Change Source Regulation [Z]**  
Select the multibyte code type (SJIS [-zs], EUC [-ze], No Multibyte [-zn]) to be used in source comments.
- **Create Error List File [-e]**  
Output Path Name  
To output a error list file, input the file name in the text box.  
Specify the path using the [Browse...] button.
- **Assembler Options**  
Specify the assembler options for the assembler source module file.
  - **Use Assembler common option**  
Enable the assembler common option set in the <Structured Assembler Options> dialog box.
  - **Assembler Source Options**  
Input the character string including the option name to enable the option for the output assembler source.
- **Include Search Path [-i]**  
Specify the include file path by using the [Browse...] button or by directly inputting it.
- **Word Symbol Character [-sc]**  
Specify the last character of the symbol that is to be defined as a word. Thereafter, generate instructions determining whether it is a word symbol or byte symbol using the last character.
- **Symbol Definition [-d]**  
Input the numerical value to be defined as symbol.
- **Use Command File**  
Select this check box to create a command file.
- **Parameterfile**  
Read a user-defined parameter file selected either using the [Browse...] button or by directly inputting it.
- **Other Options**  
To select options other than those selectable by check box or radio button, type the option in the input box.
- **Reset**  
Resets the input contents.
- **Option data read...**  
Opens the <Option Data Read> dialog box and after the option data file has been specified, reads this file.
- **Option data save...**  
Opens the <Option Data Save> dialog box and saves the option data to the option data file under the specified name.
- **Command Line Options**  
This edit box is read-only. The currently set option character string is displayed.

## CHAPTER 5 ASSEMBLER

The assembler inputs source module files written in the assembly language for 78K4 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.

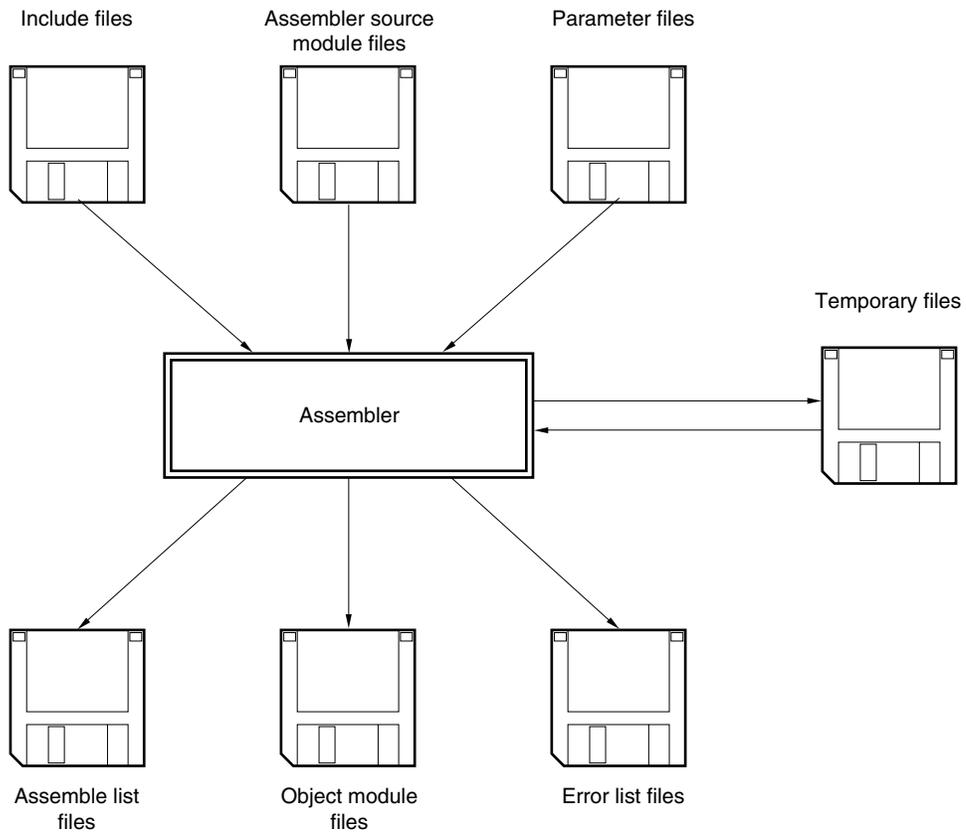
### 5.1 I/O Files of Assembler

The I/O files of the assembler are as shown below.

**Table 5-1. I/O Files of Assembler**

Type	File Name	Explanation	Default File Type
Input files	Assembler source module files	<ul style="list-style-type: none"><li>• These are source module files written in assembly language for 78K4 Series microcontrollers</li><li>• These files are created by the user.</li></ul>	.ASM
	Include files	<ul style="list-style-type: none"><li>• These files are used for reference with assembler source module files.</li><li>• These are files written in assembly language for 78K4 Series microcontrollers.</li><li>• These files are created by the user.</li></ul>	—
	Parameter files	<ul style="list-style-type: none"><li>• These files contain the parameters for the executed files.</li><li>• These files are created by the user.</li></ul>	.PRA
Output files	Object module files	<ul style="list-style-type: none"><li>• These are binary files including relocation data and symbol data regarding machine language data and machine language location addresses.</li></ul>	.REL
	Assemble list files	<ul style="list-style-type: none"><li>• These are files containing assembly data such as assemble lists and cross-reference lists.</li></ul>	.PRN
	Error list files	<ul style="list-style-type: none"><li>• These are files containing error data generated during assembly.</li></ul>	.ERA
I/O files	Temporary files	<ul style="list-style-type: none"><li>• These are files created automatically by the assembler for assembly purposes. Temporary files are deleted when assembly ends.</li></ul>	RAxxxx.\$\$n (n = 1 to 4)

Figure 5-1. I/O Files of Assembler



## 5.2 Functions of 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 write 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, refer to **5.4 Assembler Options**.
- (4) If assembly is completed correctly the assembler outputs an "Assembly Finished" message and returns control to the operating system.

## 5.3 Assembler Startup

### 5.3.1 Assembler startup

Two methods can be used to start up the assembler.

#### (1) Command-line startup

```
x > [path-name] RA78K4 [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, refer to **5.4 Assembler Options**.

- (5) File name of source module to be assembled

**Example** `C>ra78k4 -c4038 k4main.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 > RA78K4 [Δsource-module-file] Δ-F parameter-file-name
```

```
          |           |
          (2)        (1)
```

(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 writing 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 written after the option.

Write in the parameter file all assembler options and output file names specified in the command line.

For a detailed explanation of parameter files, refer to **5.4.3 Explanation of assembler options**.

**Example** Create the parameter file (K4MAIN.PRA) using an editor.

- Contents of K4MAIN.PRA

```
;Parameter file
k4main.asm -osample.rel
-psample.prn
```

- Use parameter file (K4MAIN.PRA) to start up the assembler.

```
A>ra78k4 -fk4main.pra
```

### 5.3.2 Execution start and end messages

#### (1) Execution start message

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

```
78K/IV Series Assembler Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

#### (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 : uPD78xxx
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
K4MAIN.ASM(12) : F201 Syntax error
Pass2 Start
K4MAIN.ASM(12) : F201 Syntax error
K4MAIN.ASM(29) : F407 Undefined symbol reference 'CONVAH'
K4MAIN.ASM(29) : F303 Illegal expression

Target chip : uPD78xxx
Device file : Vx.xx

Assembly complete,      3 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 non-existent source module file is specified.

```
C>ra78k4 sample.asm

78K/IV Series Assembler Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

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

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

**Example 2.** A non-existent assembler option is specified.

```
C>ra78k4 k4main.asm -z

78K/IV Series Assembler Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

A012 Missing parameter '-z'
Please enter 'RA78K4--', If you want help messages.
Program aborted.
```

In the above example, a non-existent 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 13 ERROR MESSAGES** and take action accordingly.

## 5.4 Assembler Options

### 5.4.1 Types of assembler options

The assembler options are detailed instructions for the operation of the assembler. Assembler options are classified into 16 types.

**Table 5-2. Assembler Options (1/2)**

Number	Classification	Option	Explanation
1	Device type specification	-C	Specifies the device type of the target device.
2	Object module file output specification	-O	Specifies the output of an object module file.
		-NO	
3	Forced object module file output specification	-J	Forces output of an object module file.
		-NJ	
4	Debug data output specification	-G	Outputs debugging data (local symbol data) to an object module file.
		-NG	
		-GA	Outputs assembler source debugging data to an object module file.
		-NGA	
5	Include file read path specification	-I	Reads from the path specified in an include file.
6	Assemble list file output specification	-P	Specifies output of an assemble list file.
		-NP	
7	Assemble list file data specification	-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	
8	Assemble list file format specification	-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	
9	Error list file output specification	-E	Outputs an error list file.
		-NE	
10	Parameter file specification	-F	Inputs the input file name and assembler options from a specified file.

Table 5-2. Assembler Options (2/2)

Number	Classification	Option	Explanation
11	Specification of path for temporary file creation	-T	Creates a temporary file in a specified path.
12	Specify change of SFR area	-CS	Changes the range of the SFR area from the specified value.
		-CSA	Checks all internal RAM areas and SFR peripheral areas that can be changed for absolute settings and outputs an error message if an absolute description is found.
13	Kanji code specification	-ZS	Kanji described in the comment is interpreted as shift JIS code.
		-ZE	Kanji described in the comment is interpreted as EUC code.
		-ZN	Characters described in the comment are not interpreted as kanji.
14	Device file search path specification	-Y	Reads a device file from a specified path.
15	Symbol definition specification	-D	Defines a symbol.
16	Help specification	--	Displays a help message on the display.

### 5.4.2 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 5-3. Order of Precedence of Assembler Options**

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

↑ Vertical axis

#### [Items marked with an ×]

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

**Example** `C>ra78k4 -c4038 k4main.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>ra78k4 -c4038 k4main.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>ra78k4 -c4038 k4main.asm -o -no`

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

Options not described in Table 5-3 have no particular effect on other options. However, when the help option '-?' is specified, all other options become unavailable.

### 5.4.3 Explanation of assembler options

This section contains detailed explanations of each assembler option.

#### (1) Device type specification (-C)

Syntax:                -C device-type

Default assumption: Cannot be omitted

#### [Function]

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

#### [Application]

Use option -C sparingly. The assembler performs assembly for the target device and generates an object code for that device.

#### [Explanation]

For the target devices that can be specified by option -C, refer to "Considerations when using device files."

#### [Notice]

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.

<pre> ▽\$▽PROCESSOR▽ (▽device-type▽) ▽\$▽PC▽ (▽device-type▽)      ; Abbreviated form </pre>
---

For information on control instructions, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

#### [Example of use]

Specify option -C on the command line as follows.

```
C>>ra78k4 -c4038 k4main.asm
```

**(2) Object module file output specification (-O/-NO)**

Syntax:               -O [output-file-name]  
                      -NO

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

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

**[Explanation]**

- 1) The disk type file name, device type file names NUL and AUX, and the path name can be specified in [output file name]. When the device type file names CON, PRN, and CLOCK are specified, an abort error results.
- 2) Even if option -O is specified, if a fatal error occurs the object module file cannot be output.
- 3) If the drive name is omitted when option -O is specified, the object module file will be output to the current drive.
- 4) If the output file name is omitted when option -O is specified, the output file name will be 'input-file-name.REL'.
- 5) If both options -O and -NO are specified at the same time, the option specified last takes precedence.

**[Example of use]**

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

```
C>ra78k4 -c4038 k4main.asm -osample.rel
```

**(3) Forced object module file output specification (-J/-NJ)**

Syntax:                -J  
                          -NJ

Default assumption: -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.

**[Explanation]**

- 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 of use]**

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

```
C><u>ra78k4 -c4038 k4main.asm -j
```

**(4) Debug data output specification (-G/-NG, -GA/-NGA)****(a) -G/-NG**

Syntax:            -G  
                      -NG

Default assumption: -G

**[Function]**

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

**[Application]**

- 1) Use option -G when performing symbolic debugging of data that includes local symbol data.
- 2) 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.)

**[Explanation]**

- 1) If both options -G and -NG are specified at the same time, the option specified last takes precedence.

**[Notice]**

A control instruction with the same function as options -G and -NG can be written at the beginning of a source module.

<pre> ∇\$∇DEBUG ∇\$∇DG           ; abbreviated form ∇\$∇NODEBUG ∇\$∇NODG         ; abbreviated form           </pre>
--

For information on control instructions, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

**[Example of use]**

Specify addition of debug data to an object module file.

```
C>ra78k4 -c4038 k4main.asm -g
```

**(b) -GA/-NGA**

Syntax:                -GA  
                          -NGA

Default assumption: -GA

**[Function]**

- 1) Option -GA specifies that source debugging data is to be added to an object module file by the structured assembler.
- 2) Option -NGA makes option -GA unavailable.

**[Application]**

- 1) Use option -GA when performing debugging at the source level of the assembler or structured assembler. To perform debugging at the source level, you will need the separately available integrated debugger.
- 2) Use option -NGA in the following 2 cases.
  1. Debugging without an assembler source
  2. When only the object is required (evaluation using PROM, etc.)

**[Explanation]**

- 1) If both options -GA and -NGA are specified at the same time, the option specified last takes precedence.
- 2) Option -GA takes precedence over other options regardless of the position in which it is specified.

**[Notice]**

A control instruction with the same function as options -GA and -NGA can be written at the beginning of a source module.

<pre> ▽\$▽DEBUGA ▽\$▽NODEBUGA </pre>
--------------------------------------

For information on control instructions, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

**[Example of use]**

Specify addition of assembler source debug data to an object module file.

```
C>ra78k4 -c4038 k4main.asm -ga
```

**(5) Include file read path specification (-I)**

Syntax: -I path-name [, path-name] ... (two or more path names can be specified)

Default assumption: Path contained in the source file

Path specified by the environmental variable (INC78K4)

**[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.

**[Explanation]**

- 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 of use]**

Read an include file from directory C:\SAMPLE

```
C>ra78k4 -c4038 k4main.asm -ic:\sample
```

**(6) Assemble list file output specification (-P/-NP)**

Syntax:                -P [output-file-name]  
                      -NP

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Create an assemble list file (SAMPLE.PRN).

```
C>ra78k4 -c4038 k4main.asm -psample.prn
```

**(7) Assemble list file data specification (-KA/-NKA, -KS/-NKS, -KX/-NKX)****(a) -KA/-NKA**

Syntax:            -KA  
                      -NKA

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Output an assembly list file.

```
C>>ra78k4 -c4038 k4main.asm -ka
```

Reference K4MAIN.PRN.

Assemble list

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE STATEMENT
1	1					
2	2					NAME SAMPM
3	3					;*****
4	4					;*
5	5					;* HEX -> ASCII Conversion Program *
6	6					;*
7	7					;* main-routine *
8	8					;*
9	9					;*****
10	10					
11	11					PUBLIC MAIN, START
12	12					EXTRN CONVAH
13	13					
14	14	-----				DATA DSEG AT 0FFD20H
15	15	0FFD20				HDTSA: DS 1
16	16	0FFD21				STASC: DS 2
17	17					
18	18	-----				CODE CSEG AT 0H
19	19	000000	R0400			MAIN: DW START
20	20					
21	21	-----				CSEG
22	22	000000	09C1FF00			LOCATION 15
23	23	000004				START: MOV RFM, #00
24	24	000004	092000FE0F			MOVG SP, #0FFE00H
25	25	000009	2BC400			MOV MM, #00
26	26	00000C	09C0F708			MOV STBC, #08H
27	27					
						.
						.
						.

**(b) -KS/-NKS**

Syntax:            -KS  
                      -NKS

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Output a symbol list.

```
C>ra78k4 -c4038 k4main.asm -ks
```

Reference K4MAIN.PRN.

(The assemble list is output, followed by the symbol list.)

## Symbol Table List

VALUE	ATTR	RTYP	NAME	VALUE	ATTR	RTYP	NAME
	CSEG		?CSEG		CSEG		CODE
----H		EXT	CONVAH		DSEG		DATA
FE20H	ADDR		HD TSA	0H	ADDR	PUB	MAIN
	MOD		SAMP M	0H	ADDR	PUB	START
FE21H	ADDR		STASC				

**(c) -KX/-NKX**

Syntax:                -KX  
                           -NKX

Default assumption: -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.

**[Explanation]**

- 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.

**[Notice]**

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

<pre> V\$VXREF V\$VXR           ; abbreviated form V\$VNOXREF V\$VNOXR        ; abbreviated form </pre>
---

For information on control instructions, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

**[Example of use]**

Output a cross-reference list.

```
C>ra78k4 -c4038 k4main.asm -kx
```

Reference K4MAIN.PRN.

The assemble list is output, followed by a cross-reference list.

Cross-Reference List

NAME	VALUE	R	ATTR	RTYP	SEGNAME	XREFS
?CSEG			CSEG		?CSEG	21#
CODE			CSEG		CODE	18#
CONVAH	----H	E		EXT		12@ 29
DATA			DSEG		DATA	14#
HDTSA	FF20H		ADDR		DATA	15# 26 27
MAIN	0H		ADDR	PUB	CODE	11@ 19
SAMPM			MOD			2#
START	0H	R	ADDR	PUB	?CSEG	11@ 19 22#
STASC	FE21H		ADDR		DATA	16# 31

**(8) Assemble list file format specification (-LW, -LL, -LH, -LT, -LF/NLF)****(a) -LW**

Syntax:               -LW [number-of-characters]

Default assumption: -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.

**[Explanation]**

- 1) The range of number of characters that can be specified with option -LW is shown below.  
(80 characters in the case of display output)

$$72 \leq \text{number of characters printed on 1 line} \leq 2046$$

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.  
However, when an assemble list file is output to display, 80 will be specified.
- 3) The specified number of characters does not include the terminator (CR, LF).
- 4) If option -NP is specified, option -LW is unavailable.

**[Notice]**

A control instruction with the same function as option -LW can also be written at the beginning of a source module.

$\nabla \$ \nabla \text{WIDTH}$

For information on control instructions, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

**[Example of use]**

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

```
C>ra78k4 -c4038 k4main.asm -lw80
```

This references the assemble list.

Assemble list

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE STATEMENT
1	1					
2	2					NAME SAMPM
3	3					;*****
4	4					;*
5	5					;* HEX -> ASCII Conversion Program *
6	6					;*
7	7					;* main-routine *
8	8					;*
9	9					;*****
10	10					
11	11					PUBLIC MAIN, START
12	12					EXTRN CONVAH
13	13					
14	14	-----				DATA DSEG AT 0FFD20H
15	15	0FFD20				HDTSA: DS 1
16	16	0FFD21				STASC: DS 2
17	17					
18	18	-----				CODE CSEG AT 0H
19	19	000000	R0400			MAIN: DW START
20	20					
21	21	-----				CSEG
22	22	000000	09C1FF00			LOCATION 15
23	23	000004				START: MOV RFM, #00
24	24	000004	092000FE0F			MOVG SP, #0FFE00H
25	25	000009	2BC400			MOV MM, #00
26	26	00000C	09C0F708			MOV STBC, #08H
27	27					
						.
						.
						.

**(b) -LL**

Syntax:               -LL [number-of-lines]

Default assumption: -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.

**[Explanation]**

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

$$20 \leq \text{number of lines printed on 1 page} \leq 32767$$

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.

**[Notice]**

A control instruction with the same function as option -LL can also be written at the beginning of a source module.

$\nabla\$\nabla$ LENGTH

For information on control instructions, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

**[Example of use]**

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

```
C>>ra78k4 -c4038 k4main.asm -ll20
```

This references K4MAIN.PRN.

78K/IV Series Assembler Vx.xx

Date:xx xxx xxxxx Page: 1

Command: -c4038 k4main.asm -l120  
 Para-file:  
 In-file: K4MAIN.ASM  
 Obj-file: K4MAIN.REL  
 Prn-file: K4MAIN.PRN

Assemble list

78K/IV Series Assembler Vx.xx

Date:xx xxx xxxxx Page: 2

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE STATEMENT
1	1					
2	2					NAME SAMPM
3	3					;*****
4	4					;*
5	5					;* HEX -> ASCII Conversion Program *
6	6					;*
7	7					;* main-routine *

78K/IV Series Assembler Vx.xx

Date:xx xxx xxxxx Page: 3

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE STATEMENT
8	8					;*
9	9					;*****
10	10					PUBLIC MAIN, START
11	11					:
						:

**(c) -LH**

Syntax:               -LH character-string

Default assumption: 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.

**[Explanation]**

- 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 written, the first 60 characters will be recognized and no error message will be output.

A 2-byte character is calculated as two characters.

If the maximum number of characters per line is 119 or less, the length of the effective character string changes as follows.

$$\text{Effective length} = (\text{Max. number of characters per line}) - 60$$

- 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 option -LH is omitted, the title column of the assemble list file will be blank.
- 6) The character set that can be written in the title column is as follows.

Table 5-4. Characters That Can Be Written as Titles

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

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

**[Notice]** A control instruction with the same function as option -LH can also be written at the beginning of the startup line.

$\nabla$ \$ $\nabla$ TITLE $\nabla$ ( $\nabla$ 'character-string' $\nabla$ ) $\nabla$ \$ $\nabla$ TT $\nabla$ ( $\nabla$ 'character-string' $\nabla$ ) ; abbreviated form
--

For information on control instructions, read **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.



**(d) -LT**

Syntax: -LT [number-of-characters]

Default assumption: -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.

**[Explanation]**

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

$$0 \leq \text{number of characters that can be specified} \leq 8$$

If a numerical value outside this range, or something other than a numerical value, is specified, an abort error occurs.

- 2) If -LTO 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.

**[Notice]**

A control instruction with the same function as option -LT can also be written at the beginning of a source module.

$\nabla \$ \nabla_{\text{TAB}} \nabla \text{number-of-tabs}$
--

For information on control instructions, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

**[Example of use]**

**Example 1.** sample.prn is referenced when option -LT is omitted.

Assemble list

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE	STATEMENT
1	1					NAME	SAMPLE
2	2						
3	3	----				CODE	CSEG
4	4	0000	63			MOV	A, B
5	5	0001	619A			SET1	A, 1
6	6					END	

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

```
C>ra78k4 -c4038 sample.asm -lt1
```

This references sample.prn.

Assemble list

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE	STATEMENT
1	1					NAME	SAMPLE
2	2						
3	3	----				CODE	CSEG
4	4	0000	63			MOV	A, B
5	5	0001	619A			SET1	A, 1
6	6					END	

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

**(e) -LF/-NLF**

Syntax:           -LF  
                  -NLF

Default assumption: -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 option -LF 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.

**[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.

**[Notice]**

A control instruction with the same function as option -LF/-NLF can also be written at the beginning of a source module.

<pre> ▽\$▽FORMFEED ▽\$▽NOFORMFEED </pre>
--

For information on control instructions, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

**[Example of use]**

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

```
C>ra78k4 -c4038 k4main.asm -pprn -lf
```

**(9) Error list file output specification (-E/-NE)**

Syntax:                -E [output-file-name]  
                       -NE

Default assumption: -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 option -NE makes the option -E 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.

**[Explanation]**

- 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 of use]**

Create an error list file (k4main.era).

```
C>ra78k4 -c4038 k4main.asm -ek4main.era
```

```
78K/IV Series Assembler Vx.xx [xx xxx xx]  
Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Pass1 Start  
K4MAIN.ASM(31) : F202 Illegal operand  
Pass2 Start  
K4MAIN.ASM(26) : F312 Operand out of range (byte)  
K4MAIN.ASM(31) : F202 Illegal operand
```

```
Target chip : uPD784xxx  
Device file : Vx.xx
```

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

This references the error list file (k4main.era).

```
Pass1 Start  
K4MAIN.ASM(31) : F202 Illegal operand  
Pass2 Start  
K4MAIN.ASM(26) : F312 Operand out of range (byte)  
K4MAIN.ASM(31) : F202 Illegal operand
```

**(10) Parameter file specification (-F)**

Syntax:               -F file-name

Default assumption: This option and the input file name can only be entered on the command 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.

**[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 written 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 of use]**

Perform assembly using a parameter file.

Set the contents of the parameter file (K4MAIN.PRA) as follows.

```
;parameter file
k4main.asm -osample.rel -g -c4038
-psample.prn
```

Enter the following on the command line.

```
C>ra78k4 -fk4main.pra
```

**(11) Specification of path for temporary file creation (-T)**

Syntax:                 -T path-name

Default assumption: 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.

**[Explanation]**

- 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 of use]**

Specify output of a temporary file to directory C:\TMP.

```
C>>ra78k4 -c4038 k4main.asm -tc:\tmp
```

**(12) SFR area change specification (-CS/-CSA)**

Syntax:                -CS [absolute expression n]  
                          -CSA

Default assumption: -CS0FH

**[Function]**

- 1) Option -CS [absolute expression n] uses an absolute expression n to change the range of the SFR area from the specified value.  
0XFFD00H to 0XFFFFFFH (X = 0, F)
- 2) Option -CSA checks to ensure that no LOCATION instructions are described and that no addresses located with absolute segments are found in all internal RAM areas and SFR areas that can be set.  
Do not use this option.

**[Application]**

Describe option -CS to change the SFR area.

**[Explanation]**

- 1) If both options -CS [absolute expression n] and -CSA are specified at the same time, the option specified last takes precedence.
- 2) During linking, the specified values in all modules specified by the -CS option and CHGSFR directive must be identical.  
And its value must be identical to the value specified by LOCATION instructions.
- 3) A control directive with the same function as options -CS/-CSA can also be described at the beginning of a source module.  
The description format is as follows.

```

∇$∇CHGSFR∇(∇absolute expression n∇)
∇$∇CHGSFRA

```

For information on control directives, read **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

**[Example of use]**

Set the SFR area to 0XFFD00H to 0XFFFFFFH.

```
C>>ra78k4 -c4038 k4main.asm -cs0fh
```

**(13) Kanji code specification (-ZS/-ZE/-ZN)**

Syntax:            -ZS  
                       -ZE  
                       -ZN

Default assumption: Interpreted as follows depending on the OS.

-ZS (Windows/HP-UX)  
 -ZE (SunOS/Solaris)

**[Function]**

- 1) Kanji described in the comment is interpreted as the specified kanji code.
- 2) Kanji code is interpreted as follows depending on the option.
  - ZS: Shift JIS code
  - ZE: EUC code
  - ZN: Not interpreted as kanji.

**[Application]**

These options are used to specify the interpretation of the kanji code of the kanji in the comment line.

**[Explanation]**

- 1) If the -ZS, -ZE, and -ZN options are specified at the same time, the one specified later takes priority.
- 2) The control instruction that functions as the -ZS, -ZE, and -ZN options can be described at the start of the source module.

The syntax is shown below.

▲\$▲KANJICODEΔSJIS ▲\$▲KANJICODEΔEUC ▲\$▲KANJICODEΔNONE
---

For details of the control instruction, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language manual.

- 3) Kanji code can also be specified by using the environmental variable LANG78K. For details of the environmental variables, refer to **12.2 Preparing Development Environment (Environmental Variables)**.

**[Example of use]**

The kanji code is interpreted as EUC code

```
A>ra78k4 k4main.asm -c4038 -ze
```

**(14) Device file search path specification (-Y)**

Syntax:                -Y path-name

Default assumption: Device files will be read from the path determined in the following order.

- 1) <..\dev> (for the ra78k4.exe startup path)
- 2) Path by which RA78K4 was started up
- 3) Current directory
- 4) The environmental variable PATH

**[Function]**

Reads a device file from the specified path.

**[Application]**

Specify a path where a device file exists.

**[Explanation]**

- 1) If anything other than a path name is specified after option -Y, an abort error occurs.
- 2) If the path name is omitted after option -Y, an abort error occurs.
- 3) The path from which the device file is read in the order determined as follows.
  - a. Path specified by option -Y
  - b. <..\dev> (for the ra78k4.exe startup path)
  - c. Path by which RA78K4 was started up
  - d. Current directory
  - e. The environmental variable PATH

**[Example of use]**

Specify the path for the device file as directory c:\78k4\dev

```
C>ra78k4 -k4main.asm -c4038 -yc:\78k4\dev
```

**(15) Symbol definition specification (-D)**

Syntax: -D symbol-name [= value][, symbol-name [= value]...]

Default assumption: None

**[Function]**

Option -D defines symbols.

**[Application]**

Specify option -D when defining symbols.

**[Explanation]**

- 1) The value given to a symbol is binary, octal, decimal, or hexadecimal. When value specification is omitted, 1 will be specified.
- 2) Up to 30 symbols can be specified by using a comma as a delimiter.
- 3) Up to 30 characters can be described for a symbol name.
- 4) When duplicate names are specified, the latest one specified is valid.
- 5) Symbol names are case sensitive.
- 6) Symbols defined with -D are used instead of EQU/\$SET/\$RESET. If a symbol name specified for -D was also defined in the source, an error results

**[Example of use]**

When specifying 2 as the symbol definition:

```
C>ra78k4 k4main.asm -c4038 -dSYM=2
```

**(16) Help specification (--)**

Syntax:                --

Default assumption: 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.

**[Explanation]**

- 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.

**Caution** This option cannot be specified on PM plus.

To reference PM plus help, click the help button in the <Assembler Options> dialog box.

**[Example of use]**

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

```
C>ra78k4 --
78K/IV Series Assembler Vx.xx  [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

usage : ra78k4 [option[...]] input-file [option[...]]
The option is as follows ([ ] means omissible).
-cx          :Select target chip. ( x = 012,014 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.
-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.
-ydirectory :Set device file search path.
-ffile     :Input option or source module file name from specified file.
-ga/-nga   :Output assembler source debug information to object file / Not.
-cs[address] :Specify SFR area address.
-csa       :Check exist of an absolute segment in the SFR memory space.
-dname[=data][,name[=data][...]] :Define name [with data].
-zs/-ze/-zn :Change source regulation.
           -zs:SJIS code usable in comment.
           -ze:EUC code usable in comment.
           -zn:no multibyte code in comment.
--         :Show this message.
DEFAULT ASSIGNMENT:
           -o -ne -p -ka -nks -nkx -lw132 -ll66 -nlf -lt8 -g -nj -ga
```

## 5.5 Options Settings in PM plus

This section describes the method for setting assembler options from PM plus.

### 5.5.1 Option setting method

Select [A]ssembler Options... from the [T]ools menu of PM plus or click  to display the <Assembler Options> dialog box.

Assembler options can be set by inputting the required options in this dialog box.

**Figure 5-2. <Assembler Options> Dialog Box (When <<Output1>> Tab Is Selected)**

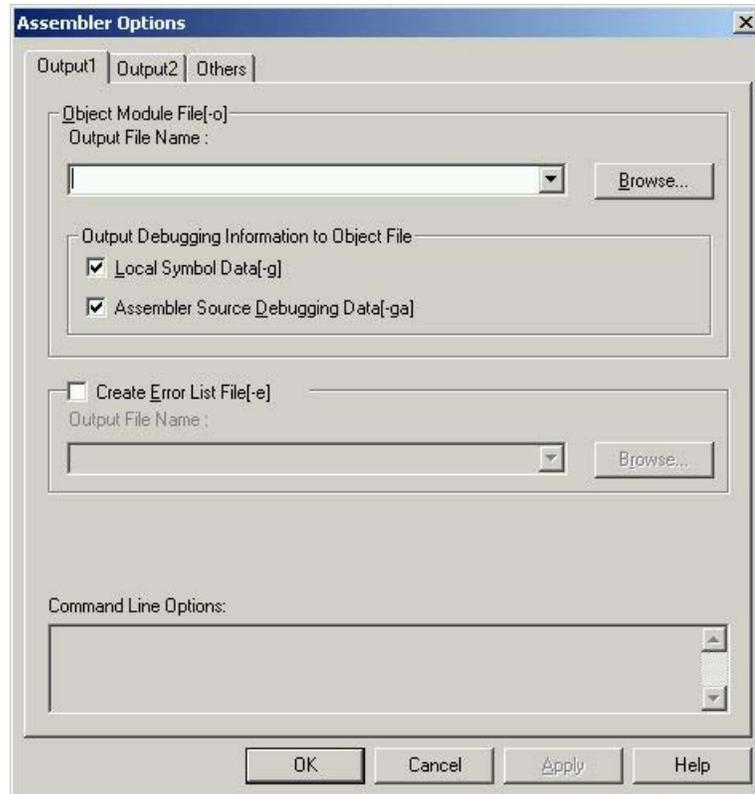


Figure 5-3. &lt;Assembler Options&gt; Dialog Box (When &lt;&lt;Output2&gt;&gt; Tab Is Selected)

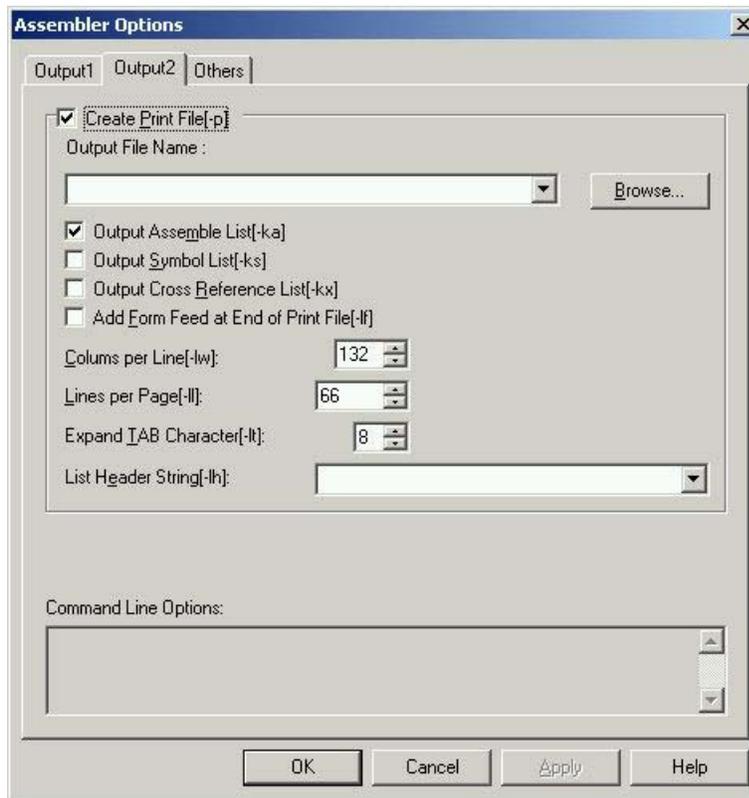
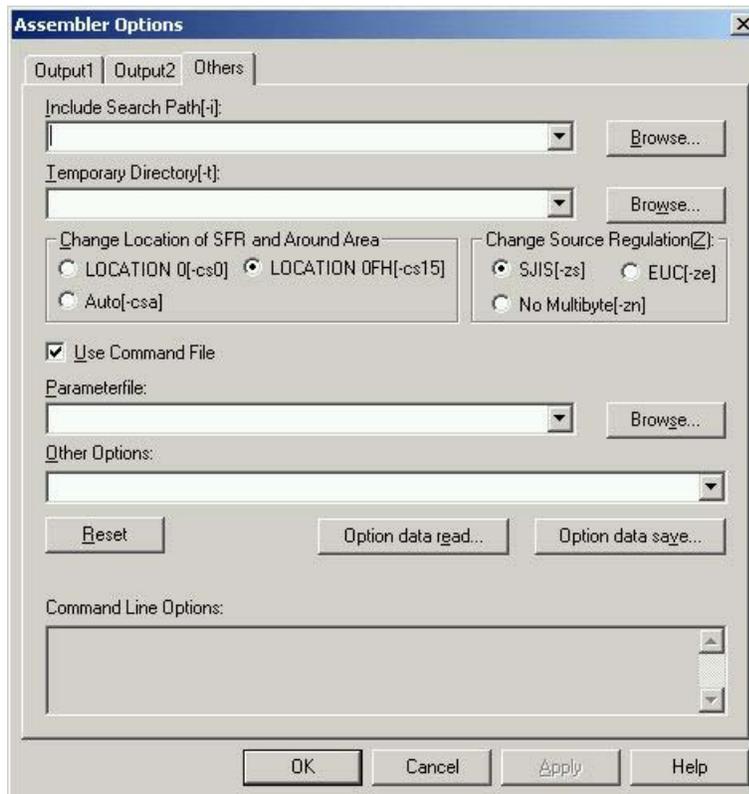


Figure 5-4. &lt;Assembler Options&gt; Dialog Box (When &lt;&lt;Others&gt;&gt; Tab Is Selected)



### 5.5.2 Option settings

The various options in the <Assembler Options> dialog box are described below.

- **O**bject Module File [-o]  
Output File Name:  
Specify the object module file output path either using the [Browse...] button or by directly inputting it.
- **L**ocal Symbol Data [-g]  
Perform symbolic debugging including local symbols.
- **A**ssembler Source **D**ebugging Data [-ga]  
Add the assembler source debug data.
- **C**reate **E**rror List File [-e]  
Output File Name:  
To output an error list file, input the file name in the input box.  
To specify the path, use the [Browse...] button.
- **C**reate Print File [-p]  
Output File Name:  
Specify the assemble list file output path either using the [Browse...] button or by directly inputting it.
- **O**utput Assemble List [-ka]  
Output the assemble list in the assemble list file.
- **O**utput **S**ymbol List [-ks]  
Output symbol list after assemble list.
- **O**utput **C**ross **R**eference List [-kx]  
Output cross-reference list after assemble list.
- **A**dd Form Feed at End of Print File [-lf]  
Append form feed code after printing contents of assemble list file.
- **C**olumns per Line [-lw]  
Specify the number of characters per line in assemble list file (selectable from 72 to 2046 characters).
- **L**ines per Page [-ll]  
Specify the number of lines per page in assemble list file (selectable from 20 to 32767 characters).
- **E**xpand **T**AB character [-lt]  
Specify the tab character length (selectable from 0 to 8 characters)
- **L**ist **H**ead String [-lh]  
Specify the character string to be printed in the title box of the assemble list file header (up to 60 characters).
- **I**nclude Search Path [-i]  
Specify the path for reading include file either using the [Browse...] button or by directly inputting it.
- **T**emporary Directory [-t]  
Specify the location where the temporary file is to be created, either using the [Browse...] button or by directly inputting it.
- **C**hange Location of SFR and Around Area  
Change the SFR area according to the specified value by selecting LOCATION 0[-cs0] or LOCATION 0FH [-cs15].  
If Auto[-csa] is selected, whether an absolute description exists or not is checked in all the internal RAM and SFR areas that can be changed.
- **C**hange Source Regulation [Z]  
Select the multibyte code type (SJIS[-zs], EUC[-ze], No Multibyte[-zn]) to be used in source comments.
- **U**se Command File  
Select this check box to create a command file.

- **Parameterfile**  
Read a user-defined parameter file selected either using the [Browse...] button or by directly inputting it.
- **Other Options**  
If wishing to specify an option other than the options that can be selected with a check box or radio button, input the option in the input box.
- **Reset**  
Resets the input contents.
- **Option data read...**  
Opens the <Option Data Read> dialog box and after the option data file has been specified, reads this file.
- **Option data save ...**  
Opens the <Option Data Save> dialog box and save the option data to the option data file with a name.
- **Command Line Options**  
This edit box is read-only. The currently set option character string is displayed.

## CHAPTER 6 LINKER

The linker inputs a number of object module files output by the 78K4 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.

### 6.1 I/O Files of Linker

The I/O files of the linker are as follows.

**Table 6-1. I/O Files of Linker**

Type	File Name	Explanation	Default File Type
Input files	Object module files	<ul style="list-style-type: none"> <li>• These are binary files which contain relocation and symbol data for machine language data and the location addresses of machine language data.</li> <li>• These files are output by the assembler.</li> </ul>	.REL
	Library files	<ul style="list-style-type: none"> <li>• These are files in which two or more object module files are included.</li> <li>• These files are output by the librarian.</li> </ul>	.LIB
	Directive files	<ul style="list-style-type: none"> <li>• These are files which contain link commands used during linking.</li> <li>• These files are created by the user.</li> </ul>	.DR
	Parameter files	<ul style="list-style-type: none"> <li>• These files contain the parameters for program execution.</li> <li>• These files are created by the user.</li> </ul>	.PLK
Output files	Load module files	<ul style="list-style-type: none"> <li>• These are binary image files which contain all data created as a result of linking. These files are input to the object converter.</li> </ul>	.LMF
	Link list files	<ul style="list-style-type: none"> <li>• These are list files which display the result of linking.</li> </ul>	.MAP
	Error list files	<ul style="list-style-type: none"> <li>• These files contain error data generated during linking.</li> </ul>	.ELK
I/O files	Temporary files	<ul style="list-style-type: none"> <li>• These files are automatically generated by the linker for use in linking. They are deleted when assembly is complete.</li> </ul>	LKxxxxx.\$\$n (n = 1 to 3)

## 6.2 Functions of 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.

## 6.3 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.

**Table 6-2. 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

- Remarks**
1. Use a directive file to change the default address of a memory area or to specify the location of each segment written in a program.
  2. For specific examples, refer to **3.5 (5) Create a directive file.**

## 6.4 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 specifies 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 6-3. Types of Directives**

No.	Directive Type	Explanation
1	Memory directive	<ul style="list-style-type: none"> <li>• Declares an address in installed memory</li> <li>• Divides memory into two or more areas and specifies a memory area</li> </ul>
2	Segment location directive	<ul style="list-style-type: none"> <li>• Specifies location of a segment</li> </ul>

### 6.4.1 Directive files

The formats for specifying directives in a directive file are as follows.

A number of directives can be specified 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 written in uppercase or lowercase characters, but not in a mixture of the two.

**Example** MEMORY  
memory  
Memory: Cannot be used

If two or more segments with the same name exist in the source, specify "COMPLETE" in order not to merge the segments, and generate an error. To merge the segments, specify "SEQUENT (default)" in the directive.

SEQUENT: Merges the segments in the order in which they appear, so that no gaps are created. BSEG merges the segments in bit units in the order in which they appear.

COMPLETE: An error occurs if two or more segments with the same name exist.

**Example** MERGE DSEG1: COMPLETE=RAM

**(2) Symbols**

Uppercase and lowercase characters are distinguished when specifying segment names, memory area names and memory space names.

**(3) Numerical values**

To specify a numerical constant for each item in a directive, write the constant in decimal or hexadecimal form. The 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 written 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 784038  
MEMORY MEM1: (01000H, 1000H) #SECOND MEMORY AREA
```

### 6.4.2 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]

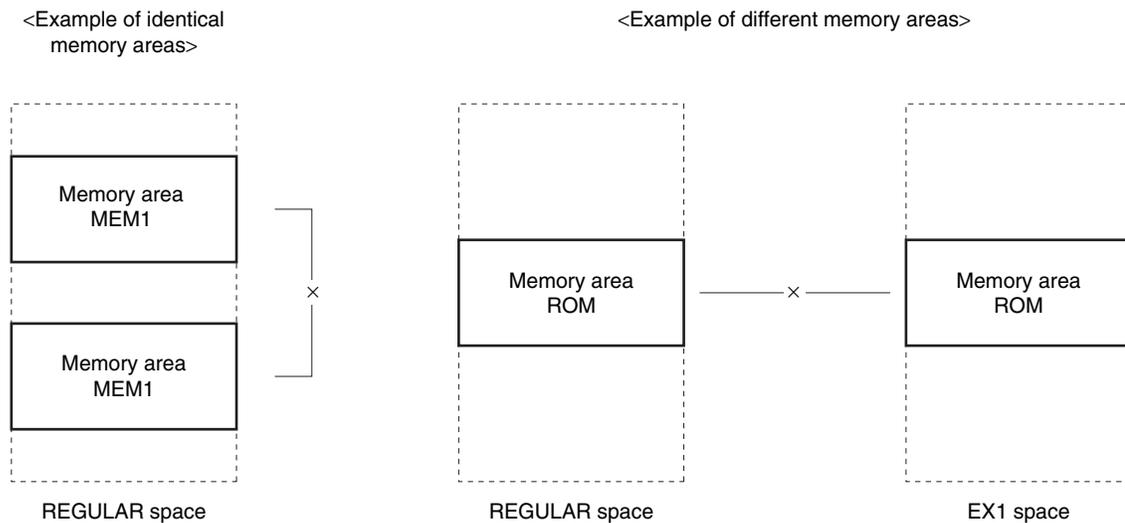
```
MEMORYΔ memory-area-name∇:∇ (∇start-address-value∇, ∇size∇) [ /∇memory-space-name ]
```

#### (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 6-1. Memory Area Names**



#### (2) Start addresses

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

Describe a numerical value from 0H to FFFFFFFH.

**(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 "Notes on Use" for each 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 specified 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 written, 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 specified 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 specification of a memory directive is omitted, only the default memory area carried by the linker for each device will be specified.
- 5) If you wish to use a different memory area without using the default memory space, specify the size of the default area name as "0".

**[Example of Use]**

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

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

- 2) Define an area in memory space EX1 as memory area RAMA.

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

### 6.4.3 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

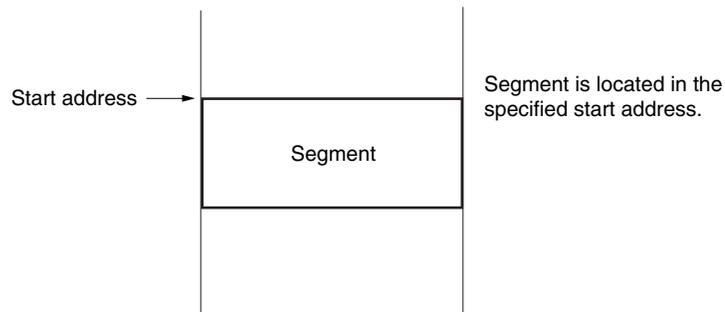
The segment name is the name of a segment included in an object module file input to the linker.

- 1) Only an input segment can be specified with a segment name.
- 2) The segment name must be specified in the same way as in the source.

#### (2) Start address

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

- 1) The reserved word AT must be specified entirely in either uppercase or lowercase characters. It cannot be specified in a mixture of uppercase and lowercase characters.
- 2) The start address specifies 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 specified 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 6-4. Segment Location According to Combination of Memory Area Name Specification and Memory Space Name**

Memory Area Name	Memory Space Name	Segment Location Destination
×	×	Default memory area in the REGULAR space
×	○	A selected memory area in the specified memory space
Memory area name	×	Specified memory area in the REGULAR space
Memory area name	○	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.

**[Notice]**

- 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 definition 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 of Use]**

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.

```
MEMORY ROM: (00000H, 20000H)
MEMORY RAM: (0FF000H, 1000H)
MEMORY MEM1: (30000H, 10000H)
```

(1) When input segment SEG1 is allocated to 2000H in memory area ROM (refer to **Figure 6-2 (1)**).

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

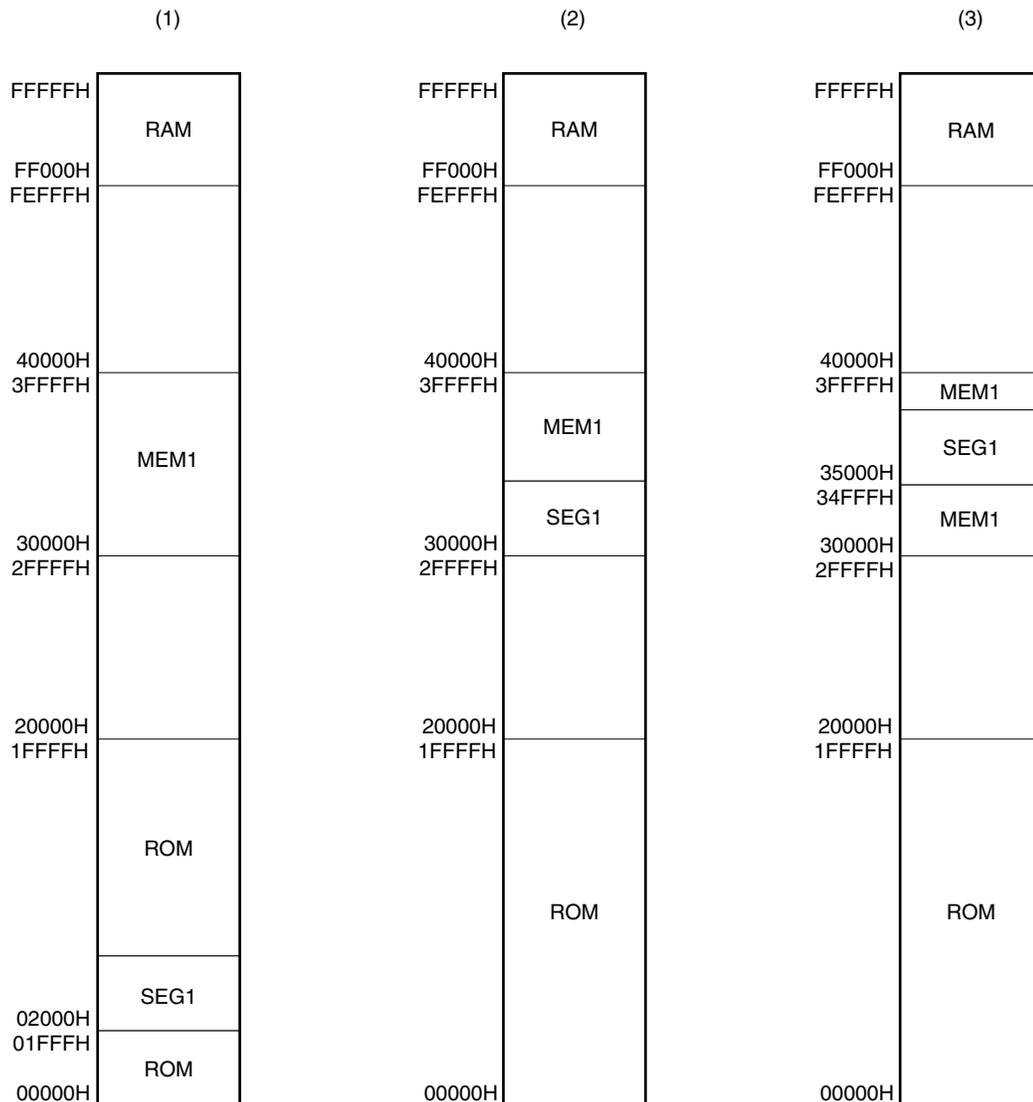
(2) When input segment SEG1 is allocated to memory area MEM1 (refer to **Figure 6-2 (2)**).

```
MERGE SEG1: =MEM1
```

(3) When input segment SEG1 is allocated to 35000H in memory area MEM1 (refer to **Figure 6-2 (3)**).

```
MERGE SEG1: AT (35000H)=MEM1
```

**Figure 6-2. Specific Examples of Segment Allocation**



## 6.5 Linker Startup

### 6.5.1 Linker startup

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

#### (1) Startup from the command line

```
x> [path-name] lk78k4 [ $\Delta$ option] ...  $\Delta$ object-module-file-name [ $\Delta$ option] ... [ $\Delta$ ]
|   |           |           |           |           |
(1) (2)        (3)        (4)        (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 256 items can be input in an input module.

**Example** C>lk78k4 k4main.rel k4sub.rel -ok4.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>LK78K4 [ $\Delta$ object-module-file]  $\Delta$ -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 writing the contents of a parameter file are as follows.

```
[[ [ $\Delta$ ] option [ $\Delta$ option] ... [ $\Delta$ ] $\Delta$ ] ...
```

- 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 written after the option.
- 3) Write in the parameter file all linker options and output file names that should be specified in the command line.

**Example** Create the parameter file (K4.PLK) using an editor.  
Contents of the parameter file K4.PLK:

```
;parameter file
k4main.rel k4sub.rel -ok4.lmf -pk4.map -e
-ta:\tmp
```

Use parameter file K4.PLK to start up the linker.

```
C>lk78k4 -fk4.plk
```

## 6.5.2 Execution start and end messages

### (1) Execution start message

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

```
78K/IV Series Linker Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

### (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.

```
Target chip : uPD784xxx
Device file : Vx.xx
```

```
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.

```
Target chip : uPD784xxx
Device file : Vx.xx
```

```
Link complete,      1 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 non-existent object module file is specified.

```
C>>lk78k4 samp1.rel samp2.rel

78K/IV Series Linker Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxxx,xxxxx

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

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

**Example 2.** A non-existent linker option is specified.

```
C>lk78k4 k4main.rel k4sub.rel -z

78K/IV Series Linker Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

A018 Option is not recognized '-z'
Please enter 'LK78K4 --', if you want help messages.
Program Aborted.
```

In the above example, a non-existent 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 13 ERROR MESSAGES** and take action accordingly.

## 6.6 Linker Options

### 6.6.1 Types of linker options

The linker options are detailed instructions for the operation of the linker. Linker options are classified into 17 types.

**Table 6-5. Linker Options (1/2)**

Number	Classification	Option	Explanation
1	Load module file output specification	-O	Specifies the output of a load module file.
		-NO	
2	Forced load module file output specification	-J	Forces output of a load module file.
		-NJ	
3	Debug data output specification	-G	Outputs debugging data to a load module file.
		-NG	
4	Stack symbol generation specification	-S	Automatically generates public symbols for stack decision.
		-NS	
5	Directive file specification	-D	Inputs the specified file as a directive file.
6	Link list file output specification	-P	Specifies output of a link list file.
		-NP	
7	Link list file data specification	-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	Link list file format specification	-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 6-5. Linker Options (2/2)

Number	Classification	Option	Explanation
9	Error list file output specification	-E	Outputs an error list file.
		-NE	
10	Library file specification	-B	Inputs the specified file as a library file.
11	Library file read path specification	-I	Reads a library file from a specified path.
12	Parameter file specification	-F	Inputs file names and options from a specified file.
13	Specification of path for temporary file creation	-T	Creates a temporary file in a specified path.
14	Device file search path specification	-Y	Reads a device file from a specified path.
15	Warning message output specification	-W	Specifies whether or not to output a warning message to the console.
16	Link specification of boot area program of flash ROM models	-ZB	Specifies the first address of the flash ROM area.
17	Help specification	--	Displays a help message on the display.

**Remark** For details of the linker option, refer to **C.3 List of Linker Options**.

### 6.6.2 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.

**Table 6-6. Order of Precedence of Linker Options**

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

↑  
Vertical  
axis

←  
Horizontal  
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>lk78k4 k4main.rel k4sub.rel -np -km`

The option -KM 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>lk78k4 k4main.rel k4sub.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>lk78k4 k4main.rel k4sub.rel -o -no`

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

Options not specified in Table 6-6 have no particular effect on other options. However, when the help option '--' is specified, all other options become unavailable.

### 6.6.3 Explanation of linker options

This section contains detailed explanations of each linker option.

#### (1) Load module file output specification (-O/-NO)

Syntax:               -O [output-file-name]  
                      -NO

Default assumption: -O input-file-name.lmf

#### [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.

#### [Explanation]

- 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.lmf' will be output to the current directory.
- 4) If only the path name is specified in 'output-file-name', 'input-file-name.lmf' 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 of use]

Output a load module file k4.lmf.

```
C>lk78k4 k4main.rel k4sub.rel -ok4.lmf
```

**(2) Forced load module file output specification (-J/-NJ)**

Syntax:                -J  
                          -NJ

Default assumption: -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.

**[Explanation]**

- 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 of use]**

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

```
C><u>lk78k4 k4main.rel k4sub.rel -j
```

**(3) Debug data output specification (-G/-NG)**

Syntax:                -G  
                          -NG

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Specify addition of debug data to a load module file.

```
C>>lk78k4 k4main.rel k4sub.rel -g
```

**(4) Generation of stack decision symbols specification (-S/-NS)**

Syntax:               -S [area-name]  
                      -NS

Default assumption: -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.

**[Explanation]**

- 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.

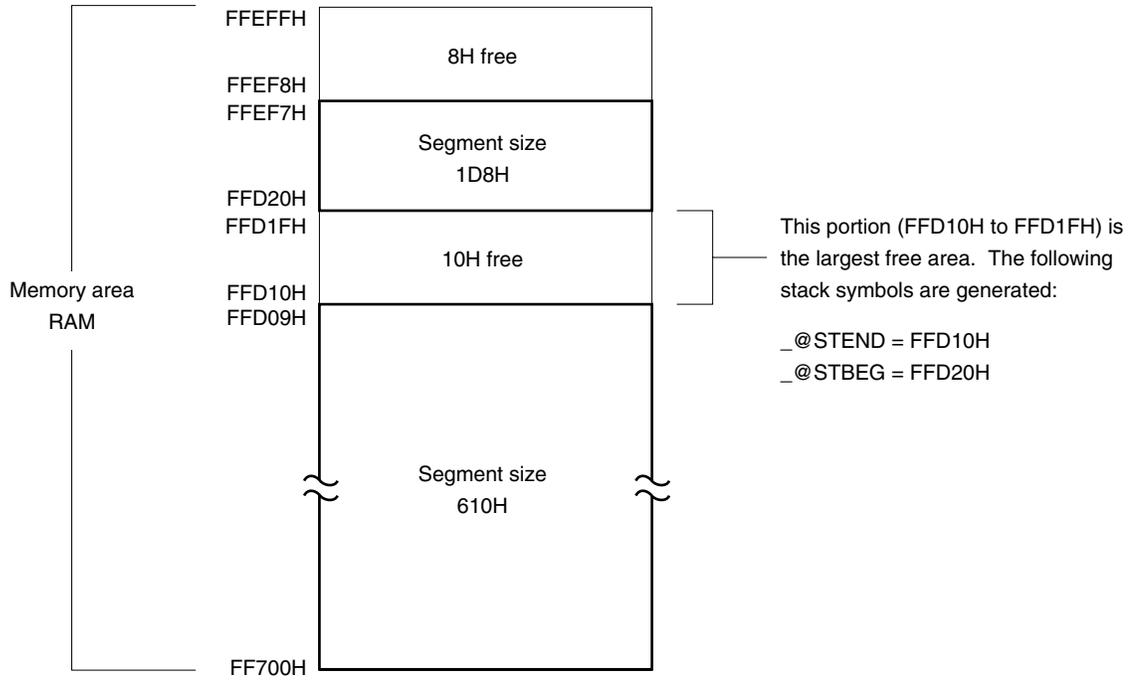
These symbols are handled as publicly declared NUMBER attribute symbols, and are registered at the end of the linker's symbol table. When these symbols are output to a link list file, the module name column is left blank.

- 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 of use]**

Reserve the stack area in memory area RAM (however, the linker will assume that a segment of size 610H in RAM and a segment of size 1D8H located in the saddr area are input).

```
C>lk78k4 k4main.rel k4sub.rel -s
```



**(5) Directive file specification (-D)**

Syntax:               -D file-name

Default assumption: 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.

**[Explanation]**

- 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 specified 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, refer to **6.4 Link Directives**.

**[Example of use]**

Redefine the default memory area ROM/RAM.

Contents of the directive file K4.DR:

```
memory ROM: (0000h, 4000h)
memory RAM: (0FF000h, 1000h)
```

Perform link using K4.DR.

```
C><u>lk78k4 k4main.rel k4sub.rel -dk4.dr
```

**(6) Link list file output specification (-P/-NP)**

Syntax:                -P [output-file-name]  
                          -NP

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Create a link list file (K4.MAP).

```
C>>lk78k4 k4main.rel k4sub.rel -pk4.map
```

**(7) Link list file data specification (-KM/-NKM, -KD/-NKD, -KP/-NKP, -KL/-NKL)****(a) -KM/-NKM**

Syntax:            -KM  
                      -NKM

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Output a map list into link list file K4.MAP.

```
C>lk78k4 k4main.rel k4sub.rel -pk4.map -km
```

This references K4.MAP.

78K/IV Series Linker Vx.xx Date:xx xxx xxxx Page: 1

Command: k4main.rel k4sub.rel -pk4.map -km  
 Para-file:  
 Out-file: K4MAIN.LMF  
 Map-file: K4.MAP  
 Direc-file:  
 Directive:

\*\*\* Link information \*\*\*

3 output segment(s)  
 48H byte(s) real data  
 23 symbol(s) defined

\*\*\* Memory map \*\*\*

SPACE = REGULAR

MEMORY = ROM

BASE ADDRESS = 00000000H SIZE = 00010000H

	OUTPUT SEGMENT	INPUT SEGMENT	INPUT MODULE	BASE ADDRESS	SIZE	
	CODE			00000000H	00000002H	CSEG AT
	CODE	SAMPM		00000000H	00000002H	
* gap *				00000002H	00000007EH	
	?CSEG			00000080H	00000046H	CSEG
		?CSEG	SAMPM	00000080H	0000002AH	
		?CSEG	SAMPS	000000AAH	0000001CH	
* gap *				000000C6H	0000FF3AH	

MEMORY = RAM

BASE ADDRESS = 000FF700H SIZE = 00000900H

	OUTPUT SEGMENT	INPUT SEGMENT	INPUT MODULE	BASE ADDRESS	SIZE	
* gap *				000FF700H	00000620H	
	DATA			000FFD20H	00000003H	DSEG AT
		DATA	SAMPM	000FFD20H	00000003H	
* gap *				000FFD23H	000001DDH	
* gap (Not Free Area) *				000FFF00H	00000100H	

Map list

Target chip : uPD784026  
 Device file : Vx.xx

**(b) -KD/-NKD**

Syntax:                -KD  
                          -NKD

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Output a link directive file into a link list file (K4.MAP).

```
C>lk78k4 k4main.rel k4sub.rel -dk4.dr -pk4.map -kd
```

This references K4.MAP.

78K/IV Series Linker Vx.xx Date:xx xxx xxxx Page: 1

```
Command: k4main.rel k4sub.rel -dk4.dr -pk4.map -kd
Para-file:
Out-file: K4MAIN.LMF
Map-file: K4.MAP
Direc-file:K4.DR ← Directive file name
Directive: memory ROM:(0000h,4000h) ← Contents of directive file
           memory RAM:(0ff000h,1000h)
```

\*\*\* Link information \*\*\*

```
3 output segment(s)
48H byte(s) real data
23 symbol(s) defined
```

\*\*\* Memory map \*\*\*

SPACE = REGULAR

```
MEMORY = ROM
BASE ADDRESS = 00000000H SIZE = 00004000H
  OUTPUT  INPUT  INPUT  BASE  SIZE
  SEGMENT SEGMENT MODULE ADDRESS
CODE      CODE   SAMPM  00000000H 00000002H CSEG AT
* gap *    ?CSEG  00000000H 00000002H
          ?CSEG  00000002H 00000007EH
          ?CSEG  00000080H 00000046H CSEG
          ?CSEG  00000080H 0000002AH
* gap *    ?CSEG  000000AAH 0000001CH
          000000C6H 00003F3AH

MEMORY = RAM
BASE ADDRESS = 000FF000H SIZE = 00001000H
  OUTPUT  INPUT  INPUT  BASE  SIZE
  SEGMENT SEGMENT MODULE ADDRESS
* gap *    DATA  000FF000H 00000D20H
          DATA  000FFD20H 00000003H DSEG AT
          DATA  000FFD20H 00000003H
* gap *    000FFD23H 000001DDH
* gap (Not Free Area) * 000FFF00H 00000100H
```

```
Target chip : uPD784xxx
Device file : Vx.xx
```

**(c) -KP/-NKP**

Syntax:               -KP  
                          -NKP

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Output a public symbol list into a link list file (K4.MAP).

```
C>lk78k4 k4main.rel k4sub.rel -g -pk4.map -kp
```

This references K4.MAP.

78K/IV Series Linker Vx.xx Date:xx xxx xxxx Page: 1

Command: k4main.rel k4sub.rel -g -pk4.map -kp  
 Para-file:  
 Out-file: K4MAIN.LMF  
 Map-file: K4.MAP  
 Direc-file:  
 Directive:

\*\*\* Link information \*\*\*

3 output segment(s)  
 48H byte(s) real data  
 23 symbol(s) defined

\*\*\* Memory map \*\*\*

SPACE = REGULAR

.  
 .  
 .

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

\*\*\* Public symbol list \*\*\*

MODULE	ATTR	VALUE	NAME
SAMPM	ADDR	00000000H	MAIN
SAMPM	ADDR	00000084H	START
SAMPS	ADDR	000000AAH	CONVAH

Public symbol list

Target chip : uPD784026  
 Device file : Vx.xx

**(d) -KL/-NKL**

Syntax:                -KL  
                          -NKL

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Output a local symbol list into a link list file (K4.MAP).

```
C>lk78k4 k4main.rel k4sub.rel -g -pk4.map -kl
```

This references K4.MAP.

78K/IV Series Linker Vx.xx Date:xx xxx xxxx Page: 1

Command: k4main.rel k4sub.rel -g -pk4.map -kl  
 Para-file:  
 Out-file: K4MAIN.LMF  
 Map-file: K4.MAP  
 Direc-file:  
 Directive:

\*\*\* Link information \*\*\*

3 output segment(s)  
 48H byte(s) real data  
 23 symbol(s) defined

\*\*\* Memory map \*\*\*

SPACE = REGULAR

.  
 .  
 .

78K/IV 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	000FFD20H	HDTSA
SAMPM	ADDR	000FFD21H	STASC
SAMPM	CSEG		CODE
SAMPM	CSEG		?CSEG
SAMPS	MOD		SAMPS
SAMPS	CSEG		?CSEG
SAMPS	ADDR	000000BDH	SASC
SAMPS	ADDR	000000C3H	SASC1

Local symbol list

Target chip : uPD784xxx  
 Device file : Vx.xx

**(8) Link list format specification (-LL, -LF/-NLF)****(a) -LL**

Syntax:               -LL [number-of-lines]

Default assumption: -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.

**[Explanation]**

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

$$20 \leq \text{number of lines printed on 1 page} \leq 32767$$

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 of use]**

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

```
C>lk78k4 k4main.rel k4sub.rel -pk4.map -ll20
```

This references K4.MAP.

78K/IV Series Linker Vx.xx Date:xx xxx xxxx Page: 1

Command: k4main.rel k4sub.rel -pk4.map -l120  
 Para-file:  
 Out-file: K4MAIN.LMF  
 Map-file: K4.MAP  
 Direc-file:  
 Directive:

\*\*\* Link information \*\*\*

3 output segment(s)  
 48H byte(s) real data

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

23 symbol(s) defined

\*\*\* Memory map \*\*\*

SPACE = REGULAR

MEMORY = ROM

BASE ADDRESS = 00000000H SIZE = 00010000H  
 OUTPUT INPUT INPUT BASE SIZE  
 SEGMENT SEGMENT MODULE ADDRESS

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

	CODE			00000000H	00000002H	CSEG AT
		CODE	SAMPM	00000000H	00000002H	
* gap *				00000002H	0000007EH	
	?CSEG			00000080H	00000046H	CSEG
		?CSEG	SAMPM	00000080H	0000002AH	
		?CSEG	SAMPS	000000AAH	0000001CH	
* gap *				000000C6H	0000FF3AH	
	.					
	.					
	.					

**(b) -LF/-NLF**

Syntax:           -LF  
                  -NLF

Default assumption: -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.

**[Application]**

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 of use]**

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

```
C><u>lk78k4 k4main.rel k4sub.rel -pk4.map -lf
```

**(9) Error list file output specification (-E/-NE)**

Syntax:                -E [file-name]  
                          -NE

Default assumption: -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]**

Create an error list file (K4.ELK).

```
C>>lk78k4 k4main.rel k4sub.rel -dk4.dr -ek4.elk
```

An error has occurred in the contents of the directive file. K4.ELK is referenced.

```
K4.DR(3) : F102 Directive syntax error
```

**(10) Library file specification (-B)**

Syntax: -B file-name

Default assumption: 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.

**Caution** When specifying two or more libraries in the <Linker Options> dialog box in PM plus, delimit them with commas (.).

- 6) For a detailed explanation of the method of creating library files, refer to **CHAPTER 8 LIBRARIAN**.

**[Example of use]**

Input a library file (K4.LIB).

(K4SUB.REL is registered in the library file).

```
C>lk78k4 k4main.rel -bk4.lib
```

**(11) Library file read path specification (-I)**

Syntax:                   -I path-name [, path-name] ... (two or more path names can be specified)

Default assumption: Path specified by environmental variable 'LIB78K4'

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.

**[Explanation]**

- 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 paths in the following sequence.
  1. Path specified by option -I
  2. Path specified by environmental variable 'LIB78K4'.
  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 of use]**

Search for a library file file from path C:\LIB.

```
C>>lk78k4 k4main.rel k4sub.rel -bk4.lib -ic:\lib
```

**(12) Parameter file specification (-F)**

Syntax:                -F file-name

Default assumption: This option and the input file name can only be entered on the command line.

**[Function]**

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

**[Application]**

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

**[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 written 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 written 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 of use]**

Perform link using a parameter file.

Set the contents of the parameter file (K4.PLK) as follows.

```
;Parameter file
k4main.rel k4sub.rel -ok4.lmf -pk4.map -e
-ta:\tmp -g
```

Enter the following on the command line.

```
C>lk78k4 -fk4.plk
```

**(13) Specification of path for temporary file creation (-T)**

Syntax:               -T path-name

Default assumption: 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.

**[Explanation]**

- 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 of use]**

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

```
C>lk78k4 k4main.rel k4sub.rel -t\tmp
```

**(14) Device file search path specification (-Y)**

Syntax:                -Y path-name

Default assumption: Device files will be read from the path determined in the following order.

- 1) <..\dev> (for the lk78k4.exe startup path)
- 2) Path by which LK78K4 was started up
- 3) Current directory
- 4) The environmental variable PATH

**[Function]**

Reads a device file from the specified path.

**[Application]**

Specify a path where a device file exists.

**[Explanation]**

- 1) If anything other than a path name is specified after option -Y, an abort error occurs.
- 2) If the path name is omitted after option -Y, an abort error occurs.
- 3) The path from which the device file is read in the order determined as follows.
  - a. Path specified by option -Y
  - b. <..\dev> (for the lk78k4.exe startup path)
  - c. Path by which LK78K4 was started up
  - d. Current directory
  - e. The environmental variable PATH

**[Example of use]**

Specify the path for the device file as directory c:\78k4\dev

```
C>lk78k4 k4main.rel k4sub.rel -yc:\78k4\dev
```

**(15) Warning message output specification (-W)**

Syntax:                -W [level]

Default assumption: 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

**[Explanation]**

- 1) If anything other than a path name is specified after option -W, an abort error occurs.
- 2) Only levels 0, 1 and 2 can be specified.
- 3) 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, refer **Table 13-3 Linker Error Messages**.

**[Example of use]**

Specify level 2 in option -W.

```
C>>lk78k4 k4main.rel k4sub.rel -w2
```

**(16) Link specification of boot area ROM program of flash ROM model (-ZB)**

Syntax: -ZB

Default assumption: No link specification

**[Function]**

Specifies the first address of the flash ROM area.

**[Explanation]**

- 1) Specifies linking of the boot area ROM program of a flash ROM model and the first address of the flash ROM area.
- 2) If no address is specified, an error occurs.

**[Example of use]**

```
C><u>lk78k4 k4main.asm -zbxxxxx</u>
```

**(17) Help specification (--)**

Syntax:                --

Default assumption: 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.

**[Explanation]**

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

**Caution** This option cannot be specified on PM plus.

To reference the PM plus help, click the help button in the <Linker Options> dialog box.

**[Example of use]**

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

```
C>lk78k4 --

78K/IV Series Linker Vx.xx [xx xxx xx]
Copyright (C) NEC Electronics Corporation xxxx,xxxx

usage : lk78k4 [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.
-ydirectory    :Set device file search path.
-j/-nj        :Create load module file if fatal error occurred / Not.
-w            :Change warning level(n=0 to 2).
-zbaddress     :Create Boot file (address:flash start address)
--           :Show this message.

Press RETURN to continue ...
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,0FFFFFFH)
          MEMORY RAMA : (100H,200H)/EX1
          MERGE CSEG1 : =ROM
          MERGE DSEG1 : AT(100H) COMPLETE = RAMA/EX1
```

## 6.7 Option Settings in PM plus

This section describes the method for setting linker options from PM plus.

### 6.7.1 Option setting method

Select [Linker Options...] from the [Tools] menu of PM plus or click  to display the <Linker Options> dialog box. Linker options can be set by inputting the required options in this dialog box.

Figure 6-3. &lt;Linker Options&gt; Dialog Box (When &lt;&lt;Output1&gt;&gt; Tab Is Selected)

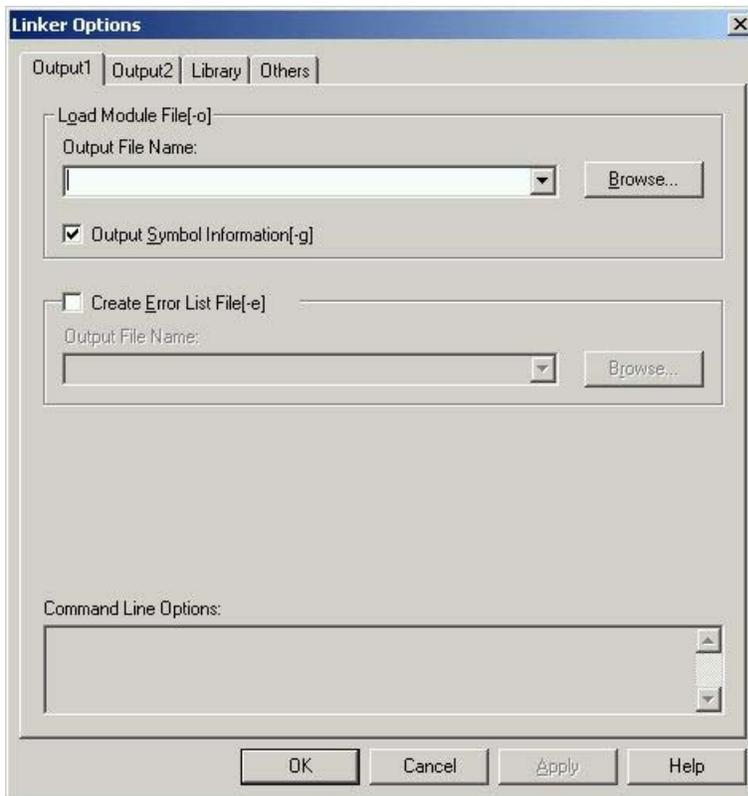


Figure 6-4. &lt;Linker Options&gt; Dialog Box (When &lt;&lt;Output2&gt;&gt; Tab Is Selected)

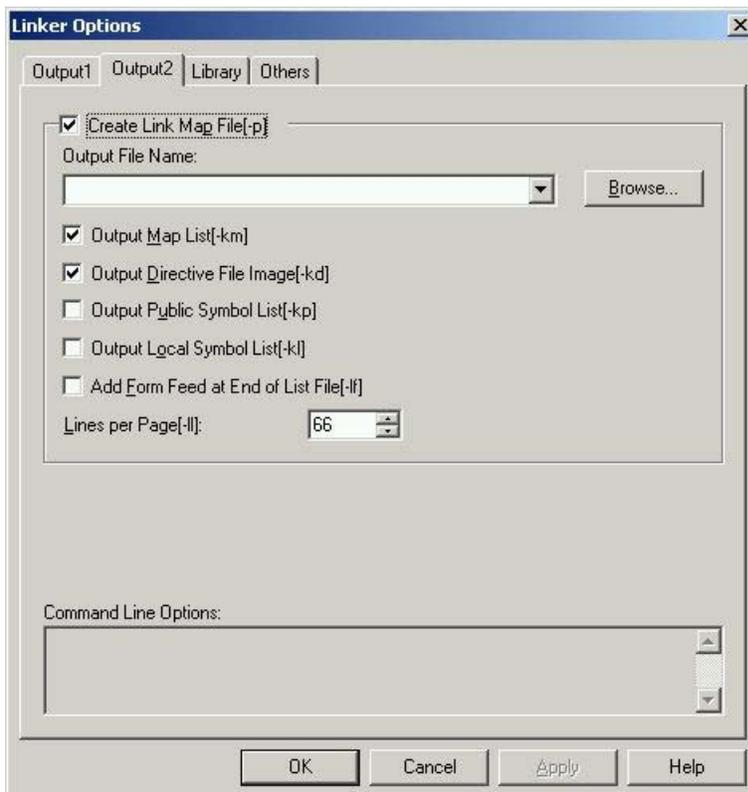


Figure 6-5. &lt;Linker Options&gt; Dialog Box (When &lt;&lt;Library&gt;&gt; Tab Is Selected)

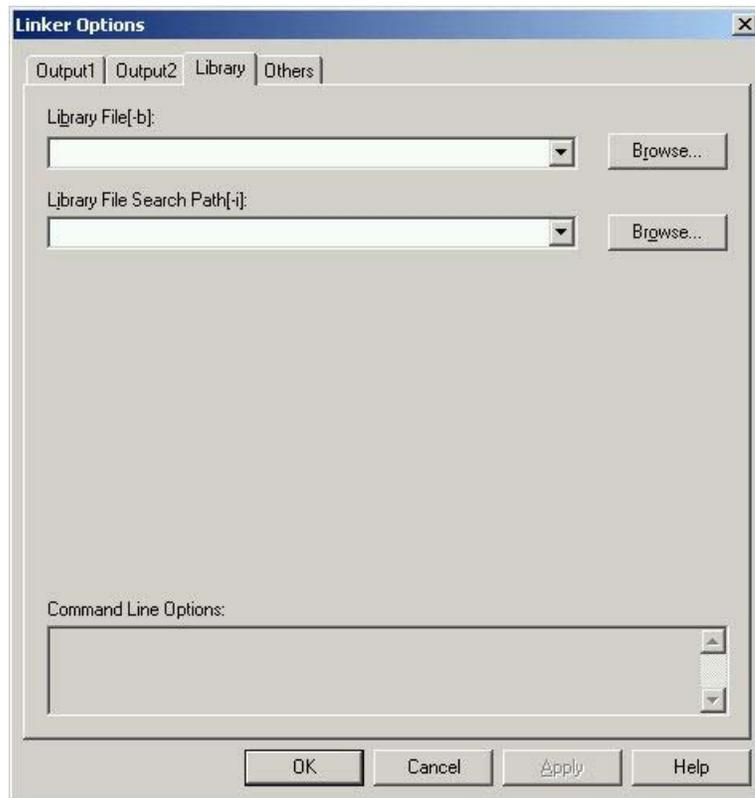
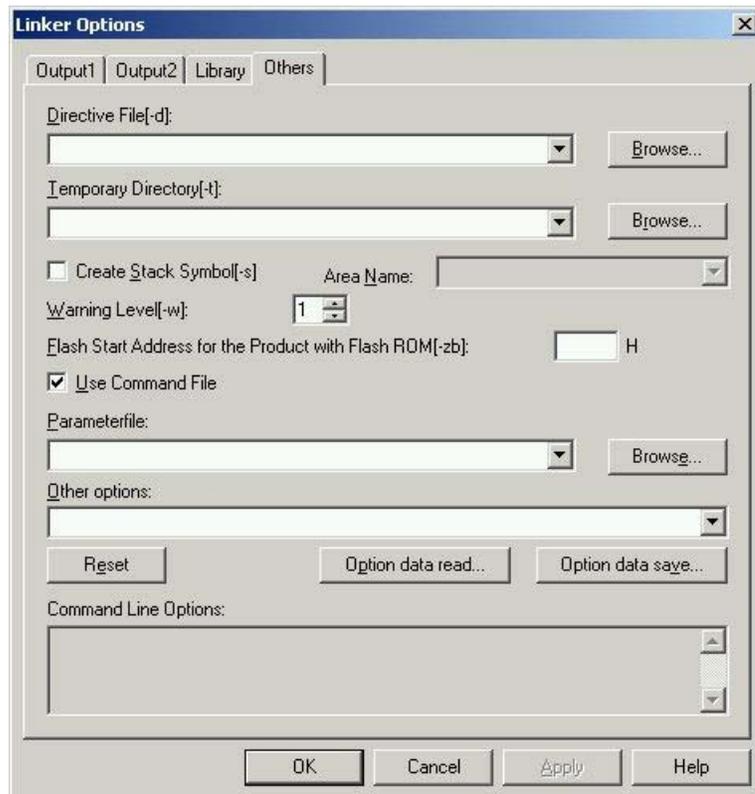


Figure 6-6. &lt;Linker Options&gt; Dialog Box (When &lt;&lt;Others&gt;&gt; Tab Is Selected)



### 6.7.2 Option settings

The various options in the <Linker Options> dialog box are described below.

- **L**oad Module File [-o]  
Output File Name:  
Specify the load module file output path either using the [Browse...] button or by directly inputting it.
- Output **S**ymbol Information [-g]  
Append debug data (local symbol information) in load module file.
- Create **E**rror List File [-e]  
Output File Name:  
To output an error list file, input the file name in the input box.  
To specify the path, use the [Browse...] button.
- Create Link **M**ap File [-p]  
Select this check box to output a link list file.  
Specify the link list file output path either using the [Browse...] button or by directly inputting it.
- Output **M**ap List [-km]  
Output the map file in the link list file.
- Output Directive File Image [-kd]  
Output the link directive file in the link list file.
- Output Public Symbol List [-kp]  
Output the public symbol list in the link list file.
- Output **L**ocal Symbol List [-kl]  
Output the local symbol list in the link list file.
- Add Form Feed at End of List File [-lf]  
Append page break code after printing contents of link list file.
- Lines per Page [-ll]  
Specify the number of lines per page in link list file (selectable from 20 to 32767 characters).
- **L**ibrary File [-b]  
Specify input of the specified file as a library file either using the [Browse...] button or by directly inputting it.
- **L**ibrary File Search Path [-i]  
Specify input of the library file from the specified path, either using the [Browse...] button or by directly inputting it.
- **D**irective File [-d]  
Specify input of the specified file as a directive file, either using the [Browse...] button or by directly inputting it.
- **T**emporary Directory [-t]  
Specify the path for creating a temporary file, either using the [Browse...] button or by directly inputting it.
- Create **S**tack Symbol [-s]  
When this option is checked, the maximum empty area in the memory area is secured as the stack area.
- Area **N**ame  
Specify the user-defined memory area name or default memory area name.
- **W**arning Level [-w]  
Specify the warning message output level.  
0: Don't output warning message.  
1: Output normal warning message.  
2: Output detailed warning message.
- **F**lash Start Address for the Product with Flash ROM [-zb]  
Specify the boot area start address for internal flash ROM versions.

- **U**se Command File  
Select this check box to create a command file.
- **P**arameterfile  
Read the user-defined parameter file either using the [Browse...] button or by directly inputting it.
- **O**ther options  
If wishing to specify an option other than the options that can be selected with a check box or radio button, input the option in the input box.
- **R**eset  
Resets the input contents.
- **O**ption data read...  
Opens <Option Data Read> dialog box opens and the option data file is specified, that file is read.
- **O**ption data save...  
After the <Option Data Save> dialog box opens, save the option data file to the option data file with a name.
- **C**ommand Line Options  
This edit box is read-only. The currently set option character string is displayed.

## CHAPTER 7 OBJECT CONVERTER

The object converter inputs the load module file output by the RA78K4 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 used 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.

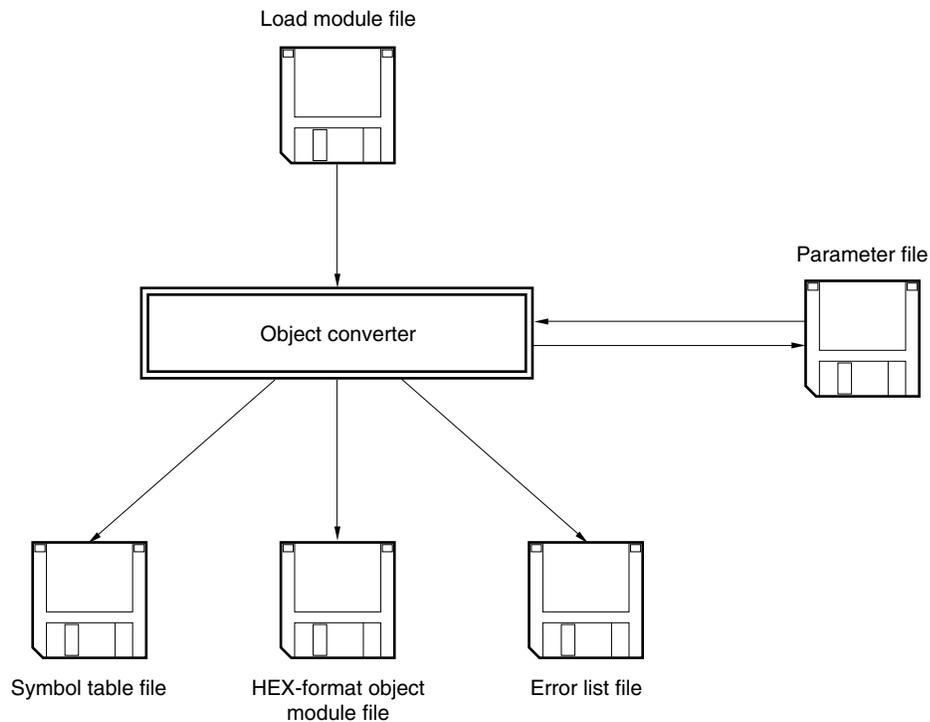
## 7.1 I/O Files of Object Converter

The I/O files of the object converter are as shown below.

**Table 7-1. I/O Files of Object Converter**

Type	File Name	Explanation	Default File Type
Input files	Load module files	<ul style="list-style-type: none"> <li>• These are binary image files of the object codes output as a result of linking.</li> <li>• These files are output by the linker.</li> </ul>	.LMF
	Parameter files	<ul style="list-style-type: none"> <li>• These files contain the parameters for the executed programs.</li> <li>• These files are created by the user.</li> </ul>	.POC
Output files	HEX format object module files	<ul style="list-style-type: none"> <li>• These are files created by converting load module files into hexadecimal object format.</li> <li>• These files are used during mask ROM development and PROM program use.</li> </ul>	.HEX
	Symbol table files	<ul style="list-style-type: none"> <li>• These files contain the symbol data included in each module of an input file.</li> </ul>	.SYM
	Error list files	<ul style="list-style-type: none"> <li>• These files contain error data from the object conversion.</li> </ul>	.EOC

**Figure 7-1. I/O Files of Object Converter**



## 7.2 Functions of 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.

To output a separate HEX file to each space, specify the space for both memory and merge directives in the link directive file. For the link directive, refer to **6.4.1 Directive files**.

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 7-2 shows the file types of the HEX-format object module files and symbol table files generated for extended space.

**Table 7-2. 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) Flash ROM self-rewriting mode support

The object converter can create separate HEX object module files in the boot area and flash area for the code located in the flash ROM when the self-rewriting mode of the flash ROM is used. To output separate HEX files, specify object converter option -ZF. The file type is as follows:

**Table 7-3. File Type When -ZF Option Is Specified**

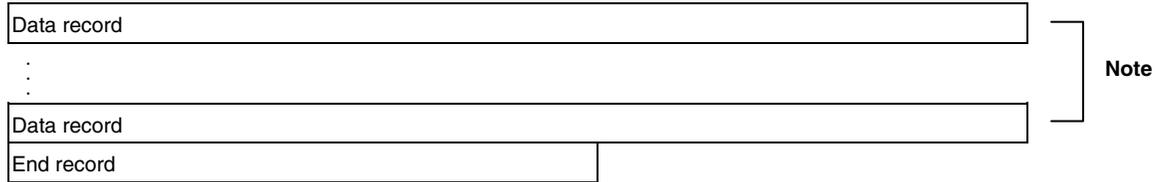
File	File Type
Output file at boot area ROM program side	.HXB
Output file at program side other than boot area ROM	.HXF

### (3) 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.

```
:020000020000FC
:0200000084007A
:020000020000FC
:1000800009C1FF002BCC00092000FE0F2BC4000982
:10009000C0F7083A201A38FB20FD0F09F0AA0038F3
:1000A000DB21FD0FD350D25014FEB900059F3F0C49
:1000B000002431B900059F3F0300242156AF0A8375
:0600C00002A807A830565B
:00000001FF
```

**[Intel standard HEX-format object module file format]****Figure 7-2. Intel Standard Format**

**Note** The data record is repeated here.

**(a) Data record**

```

: XX XXXX 00 DD...DD SS
| | | | | |
(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.
- (3) Location address (offset)  
The start address (offset) of the code displayed in the record is shown as a 4-digit hexadecimal.
- (4) Record type (2 digits)  
Fixed at 00.
- (5) Code (Max. 32 digits)  
The object code is shown one byte at a time, with the higher 4 bits and lower 4 bits separated. A maximum of 16 bytes can be expressed in the code.
- (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.

**(b) End record**

```

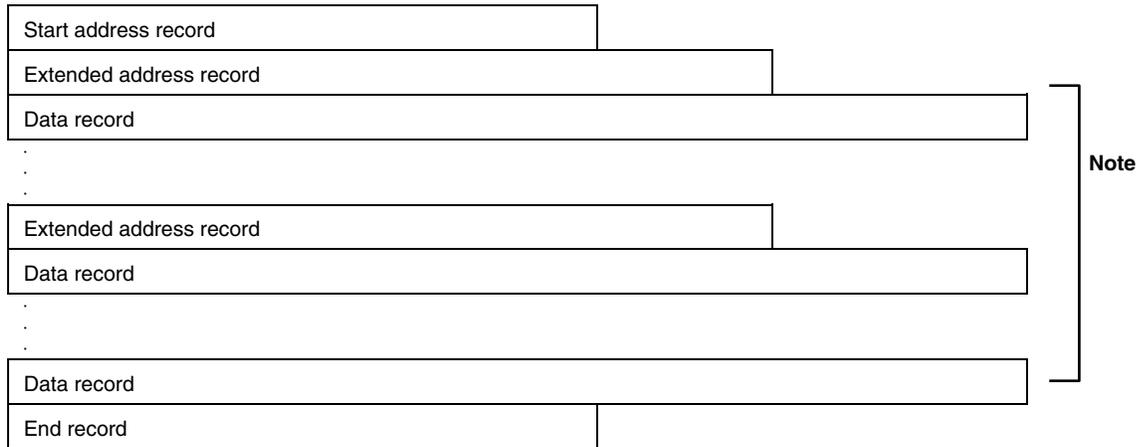
: 00 0000 01 FF
| | | | |
(1) (2) (3) (4) (5)

```

- (1) Record mark
- (2) Code number, fixed to 00
- (3) Fixed to 0000
- (4) Record type, fixed to 01
- (5) Check sum, fixed to FF

[Intel extended HEX-format object module file format]

Figure 7-3. Intel Extended Format



**Note** The extended address record and data record are repeated here.

(a) Extended address record

```

: 02 0000 02 XXXX SS
| | | | | |
(1) (2) (3) (4) (5) (6)
    
```

- (1) Record mark  
Indicates beginning of record.
- (2) Code number, fixed to 02
- (3) Fixed to 0000
- (4) Record type, fixed to 02
- (5) Higher 8-bit value of address  
The paragraph value of the segment is shown as a 4-digit hexadecimal.
- (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 higher 8-bit value of the address.

**(b) Data record**

```

: XX XXXX 00  DD...DD SS
| | | | | |
(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.
- (3) Location address (offset)  
The start address (offset) of the code displayed in the record is shown as a 4-digit hexadecimal.
- (4) Record type (2 digits)  
Fixed to 00H.
- (5) Code (Max. 32 digits)  
The object code is shown one byte at a time, with the higher 4 bits and lower 4 bits separated. A maximum of 16 bytes can be expressed in the code.
- (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.

**(c) Start address code**

```

: 04 0000 03 0000 0000 F9
| | | | | | |
(1) (2) (3) (4) (5) (6) (7)
    
```

- (1) Record mark
- (2) Fixed to 04
- (3) Fixed to 0000
- (4) Fixed to 03
- (5) Fixed to 0000
- (6) Fixed to 0000
- (7) Fixed to F9

**(d) End record**

```

: 00 0000 01 FF
| | | | |
(1) (2) (3) (4) (5)
    
```

- (1) Record mark
- (2) Fixed to 00
- (3) Fixed to 0000
- (4) Fixed to 01
- (5) Fixed to FF

**[Extended tech HEX-format object module file format]**

HEX files are composed of the following 3 types of block.

- 1) Data block
- 2) Symbol block (This is an unused block. Symbol data uses the symbol table file.)
- 3) Termination block

Each block starts with a header field composed of a common 6 characters, and ends with the string end-of-line.

Maximum length of each block is 255, not including the start character % and end-of-line.

The format for the common header field is shown in **Table 7-4 Extended Tech Header Field**.

**Table 7-4. Extended Tech Header Field**

Item	No. of ASCII Characters	Explanation
%	1	The percent symbol specifies that the block is in extended tech format.
Block length	2	This is a 2-digit hexadecimal which indicates the number of characters in the block. This number of characters does not include the start character % and end-of-line.
Block type	1	6 = Data block 3 = Symbol block 8 = Termination block
Check sum	2	This is a 2-digit hexadecimal which indicates the remainder produced when the total value of the characters in the block (except the start character %, the check sum, and end-of-line) is divided by 256. The total value of the characters is shown in <b>Table 7-5 Character Values for Check Sum Evaluation</b> .

**Table 7-5. Character Values for Check Sum Evaluation**

Character	Value (Decimal)
0 to 9	0 to 9
A to Z	10 to 35
\$	36
%	37
. (period)	38
_ (underscore)	39
a to z	40 to 65

**(a) Data block**

The format for the data block is shown in **Table 7-6 Data Block Format for Extended Tech.**

**Table 7-6. Data Block Format for Extended Tech**

Field	No. of ASCII Characters	Explanation
Header	6	Standard header field Block type = 6
Load address	2 to 17	Address from which the object code is loaded. Number of characters is variable.
Object code	2n	Number of bytes n, displayed as a 2-digit hexadecimal

**Caution** In extended Tech, the number of characters in a specific field is variable within 2 to 17 (1 to 16 characters of actual data). The first character in this variable field is a hexadecimal which indicates the length of the field. The numerical zero indicates that a character line consists of 16 characters. The length of the character string is therefore 1 to 16 characters, and the length of the variable-length field including the character string length indicator is 2 to 17.

**[Example]**

```
% 15 6 1C 3 100 020202020202
| | | | | | |
(1) (2) (3) (4) (5) (6) (7)
```

- (1) Header character
- (2) Block length: 15H = 21
- (3) Block type: 6
- (4) Check sum: 1CH
- (5) Number of digits in load address
- (6) Load address: 100H
- (7) Object code: 6 bytes

**(b) Termination block**

The format for the termination block is shown in **Table 7-7 Termination Block Format for Extended Tech.**

**Table 7-7. Termination Block Format for Extended Tech**

Field	No. of ASCII Characters	Explanation
Header	6	Standard header field Block type = 8
Load address	2 to 17	Start address for program execution. Number of characters is variable.

**[Example]**

```
% 08 8 1A 2 80
| | | | | |
(1) (2) (3) (4) (5) (6)
```

- (1) Header character
- (2) Block length: 8H
- (3) Block type: 8
- (4) Check sum: 1AH
- (5) Number of digits in load address
- (6) Load address: 80H

**(c) Symbol block (unused)**

The extended Tech symbol block is data used for symbolic debugging. It may be assumed to have the following characteristics.

- 1) Symbol: 1 to 16 uppercase and lowercase alphabets, numerals, period and underscore.  
Numerals are not permitted for the start character.
- 2) Value: Up to 64 bits (16 hexadecimal digits) possible.
- 3) Type: Address or scalar (a scalar indicates any numerical value other than an address).  
Addresses are divided into code addresses (instruction addresses) and data addresses (addresses of data items).
- 4) Global/local specification: Indicates whether a symbol is global (external reference enabled) or local.
- 5) Section membership: A section may be considered a range to which a memory name is given.  
Each address in a program belongs to at least 1 section. A scalar does not belong to any section.

The format for the symbol block is shown in **Table 7-8 Symbol Block Format for Extended Tech.**

**Table 7-8. Symbol Block Format for Extended Tech**

Field	No. of ASCII Characters	Explanation
Header	6	Standard header field Block type = 3
Section name	2 to 17	Name of the section which includes the symbols defined in the block. Number of characters is variable.
Section definition	5 to 35	Each symbol block must have 1 of this type of field. This field may be placed before or after any number of symbol definition fields. This format is shown in Table 7-9.
Symbol definition	5 to 35 each	This is a symbol definition field greater than 0 as shown in Table 7-10.

The symbols contained in a program are transferred as a symbol block. Each symbol block includes a section name and a list of the symbols that belong to that section. If necessary, a scalar can also be included in any section.

A symbol in the same section can be placed in one or more blocks.

The formats for the section definition field and the symbol definition field in the symbol block are shown in Table 7-9 and Table 7-10.

**Table 7-9. Symbol Block Section Definition Fields for Extended Tech**

Field	No. of ASCII Characters	Explanation
0	1	0 specifies that the field is a section definition field.
Base address	2 to 17	This is a section start address. Number of characters is variable.
Length	2 to 17	Indicates the section length. Number of characters is variable and is calculated by the following equation: $1 - (\text{higher address} - \text{base address})$

**Table 7-10. Symbol Block Symbol Definition Fields for Extended Tech**

Field	No. of ASCII Characters	Explanation
Type	1	1-digit hexadecimal indicating global/local symbol specification and type of value displayed. 1 = Global address 2 = Global scalar 3 = Global code address 4 = Global data address 5 = Local address 6 = Local scalar 7 = Local code address 8 = Local data address
Symbol	2 to 17	Indicates the symbol length. Variable.
Numerical value	2 to 17	Value corresponding to a symbol. Number of characters is variable.

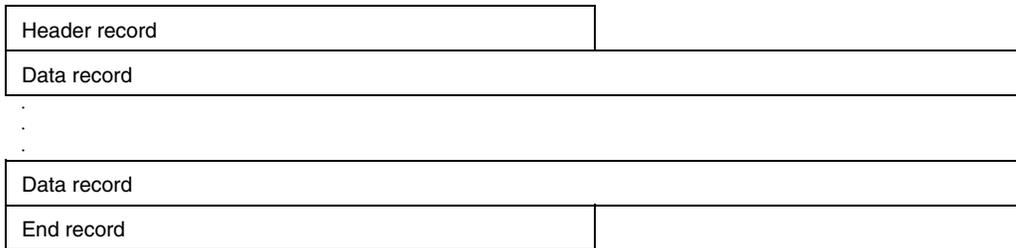
**[Motorola S-type format]**

Motorola S-type format files are converted from 5 records into 3 types. The composition of the entire file is shown in **Figure 7-4 Motorola S-Type Format**. Types of records are shown in **Table 7-11 Motorola HEX File Record Types**.

**Table 7-11. Motorola HEX File Record Types**

Header record (optional)	S0
Data record	S2 (Standard 24 bits) S3 (32 bits)
End record	S8 (Standard 24 bits) S7 (32 bits)

**Figure 7-4. Motorola S-Type Format**



Motorola HEX format files are divided into standard 24-bit addresses and 32-bit addresses. Standard addresses are composed of records S0, S2, and S8. The 32-bit addresses are composed of records S0, S3 and S7. Header record S0 is optional and is not output. A CR character is placed at the end of each S record.

The general formats and their meanings for each field in each record are shown in **Table 7-12 General Format for Each Record** and **Table 7-13 Meanings of Fields**.

**Table 7-12. General Format for Each Record**

Record Type	General Format
S0	S0XXYY...YYZZZZ
S2	S2XXWWWWWWWDD...DDZZ
S3	S3XXWWWWWWWDD...DDZZ
S7	S7XXWWWWWWWZZ
S8	S8XXWWWWWWWZZ

Table 7-13. Meanings of Fields

Field	Meaning
Sn	Record type
XX	Length of data record Number of bytes in the address, hexadecimal data and check sum
YY...YY	File name ASCII code for the input file name expressed as a hexadecimal
WWWWW[WW]	24th [32th] bit address
DD...DD	Hexadecimal data 1 byte of data is expressed as a 2-digit hexadecimal.
ZZ	Check sum The lower 1 byte of complement 1 for the sum for each byte of the record length, address and the hexadecimal data, expressed as a 2-digit hexadecimal

**[Example]**

S2 08 00FF11 D4520A20 A0  
 | | | | |  
 (1) (2) (3) (4) (5)

- (1) Record type: S2
- (2) Record length: 8 bytes
- (3) Load addresses (24-bit address)
- (4) Hexadecimal data
- (5) Check sum

**(a) S0 record**

S0 XX YYYYYYYY ZZ  
 | | | |  
 (1) (2) (3) (4)

- (1) Record type
- (2) Record length  
This is the number of bytes in (3) plus the number of bytes in (4).
- (3) File name
- (4) Check sum

**(b) S2 record**

<u>S2</u>	<u>XX</u>	<u>WWWWWWW</u>	<u>DD...DD</u>	<u>ZZ</u>
(1)	(2)	(3)	(4)	(5)

(1) Record type

(2) Record length

This is the number of bytes in (3) plus the number of bytes in (4) plus the number of bytes in (5).

(3) Load address

This is the 24-bit load address of the data in (4) within the range 0H to FFFFFFFH.

(4) Data

This is the loaded data itself.

(5) Check sum

**(c) S3 record**

<u>S3</u>	<u>XX</u>	<u>WWWWWWW</u>	<u>DD...DD</u>	<u>ZZ</u>
(1)	(2)	(3)	(4)	(5)

(1) Record type

(2) Record length

This is the number of bytes in (3) plus the number of bytes in (4) plus the number of bytes in (5).

(3) Load address

This is the 24-bit load address of the data in (4) within the range 0H to FFFFFFFH.

(4) Data

This is the loaded data itself.

(5) Check sum

**(d) S7 record**

<u>S7</u>	<u>XX</u>	<u>WWWWWWW</u>	<u>ZZ</u>
(1)	(2)	(3)	(4)

(1) Record type

(2) Record length

This is the number of bytes in (3) plus the number of bytes in (4).

(3) Entry address

This is the 32-bit entry address within the range 0H to FFFFFFFH.

(4) Check sum

**(e) S8 record**

```

S8 XX WWWWWWW ZZ
| | | |
(1) (2) (3) (4)

```

- (1) Record type
- (2) Record length  
This is the number of bytes in (3) plus the number of bytes in (4).
- (3) Entry address  
This is the 32-bit entry address within the range 0H to FFFFFFFH.
- (4) Check sum

**(4) 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.

```

#08
;FF          PUBLIC
01000000AAACONVAH
0100000000MAIN
0100000084START
;FF          SAMPM
<02000FFD20HDTSA
02000FFD21STASC
;FF          SAMPS
<01000000BDSASC
01000000C3SASC1
=

```

[Symbol Table File Formats]

Start of symbol table	#	04	CR	LF			
Start of public symbol	;	FF	4 blank spaces	PUBLIC	CR	LF	
<b>Note 1</b> →		Symbol attributes	Symbol value	Public symbol name	CR	LF	Public symbols
		•	•	•	•	•	
		•	•	•	•	•	
	;	FF	4 blank spaces	Module name 1	CR	LF	Local symbols for each module
Start of local symbol	<	Symbol attributes	Symbol value	Local symbol name	CR	LF	
		Symbol attributes	Symbol value	Local symbol name	CR	LF	
		•	•	•	•	•	
		•	•	•	•	•	
		•	•	•	•	•	
Repeated in units of object modules.		•	•	•	•	•	
		•	•	•	•	•	
		•	•	•	•	•	
Symbol table end mark	=	CR	LF				

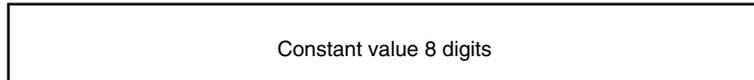
- Notes**
1. Symbol attributes are the values shown in **Table 7-14**.
  2. For symbol values, refer to **Figure 7-5**.

**Table 7-14. Values of Symbol Attributes**

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 for ABBIT attribute
04	Bit symbol for AWBIT attribute
05	Bit symbol for saddr.bit/bsaddr.bit3
06	Bit symbol for wsaddr.bit4
07	Bit symbol for sfr.bit/bsfr.bit3
08	Bit symbol for wsfr.bit4
09	Bit symbol for RBBIT attribute
10	Bit symbol for RWBIT attribute
11	Bit symbol for byte register.bit
FF	Module name

**Figure 7-5. Symbol Value Formats**

- When the symbol attribute is NUMBER



- When the symbol attribute is LABEL



- When the symbol attribute is a bit symbol



↑  
n: Bit position (0 to 7)

## 7.3 Object Converter Startup

### 7.3.1 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] oc78K4 [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>oc78K4 k4.lmf -osample.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, refer to 7.4 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>oc78K4 [-load-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 writing 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 specified after the option.
  3. Write 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 (K4.POC) using an editor.

Contents of K4.POC

```
;parameter file
K4.lmf -osample.hex
-ssample.sym -r
```

Use parameter file K4.POC to start up the object converter.

```
C>oc78K4 -fK4.poc
```

### 7.3.2 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/IV Series Object Converter Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

#### (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.

```
Target chip : uPD7804xxx
Device file : Vx.xx

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.

```
Target chip : uPD784xxx
Device file : Vx.xx

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 non-existent load module file name is specified.

```
C>>oc78K4 sample.lmf

78K/IV Series Object Converter Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

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

In the above example, a non-existent load module file is specified. An error results and the object converter aborts the object conversion.

**Example 2.** A non-existent object converter option is specified.

```
C>>oc78K4 k4.lmf -a

78K/IV Series Object Converter Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

A018 Option is not recognized '-a'
Please enter 'OC78K4 --', if you want help messages.
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 conversion is aborted, look for the cause in **CHAPTER 13 ERROR MESSAGES** and take action accordingly.

## 7.4 Object Converter Options

### 7.4.1 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 10 types.

The classifications of the object converter options and explanations of each type are shown below.

**Table 7-15. Object Converter Options**

Number	Classification	Option	Explanation
1	HEX format object module file output specification	-O	Specifies the output of a HEX-format object module file.
		-NO	
2	Symbol table file output specification	-S	Specifies output of a symbol table file.
		-NS	
3	Specification of sort by object address order	-R	Sorts HEX-format objects in the order of their addresses.
		-NR	
4	Object complement specification	-U	Outputs a specified complement value as an object code for an address area to which no HEX-format object is output.
5	Error list file output specification	-E	Outputs an error list file.
		-NE	
6	Parameter file specification	-F	Inputs an input file name and options from a specified file.
7	Specify HEX format	-KI	Outputs Intel standard HEX format object.
		-KIE	Outputs Intel extended HEX format object.
		-KT	Outputs extended Tech format object.
		-KM	Outputs Motorola S-type format (standard address) object.
		-KME	Outputs Motorola S-type format (32-bit address) object.
8	Device file search path	-Y	Reads a device file from a specified path.
9	File division output specification for flash ROM model	-ZF	<ul style="list-style-type: none"> <li>• Adds an option that outputs the boot area and other areas to separate HEX-format files when linking of the boot area ROM program of a flash ROM model is specified.</li> <li>• If the -ZF option is specified, the output file type at the boot area ROM program side is "HXB", and the output file type at the side of the other programs is "HXF".</li> </ul>
10	Help specification	--	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, refer to **C.4 List of Object Converter Options**.

### 7.4.2 Explanation of object converter options

This section contains detailed explanations of each object converter option.

#### (1) HEX format object module file output specification (-O/-NO)

Syntax:               -O [output-file-name]  
                      -NO

Default assumption: -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.

#### [Explanation]

- 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.
- 5) When -ZF option is specified, the file type is as follows:

**Table 7-16. File Type When -ZF Option Is Specified**

File	File Type
Output file at boot area ROM program side	.HXB
Output file at program side other than boot area ROM	.HXF

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.

**Table 7-17. Type of HEX Format Object Module File for Extended Space**

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 of use]**

Output a HEX-format object module file (SAMPLE.HEX).

```
C>oc78K4 k4.lmf -osample.hex
```

**(2) Symbol table file output specification (-S/-NS)**

Syntax:               -S [output-file-name]  
                      -NS

Default assumption: -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.

**[Explanation]**

- 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.

**Table 7-18. Types of Symbol Table File for Extended Space**

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]**

Output a symbol table file (SAMPLE.SYM).

```
C>oc78K4 k4.lmf -ssample.sym
```

**(3) Specification of sort by object address order (-R/-NR)**

Syntax:                -R  
                          -NR

Default assumption: -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.

**[Explanation]**

- 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.

**[Example of use]**

Sort HEX-format objects in order of address.

```
C>>oc78K4 k4.lmf -r
```

**(4) Object complement specification (-U)**

Syntax: -U complement-value [, [start] , size]

Default assumption: 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.

**[Explanation]**

- 1) The range of values that can be specified as complement values is as follows.

$$0H \leq \text{complement value} \leq 0FFH$$

Complement values can be specified in binary, octal, 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.

$$0H \leq \text{start} \leq \text{Largest address in program space other than SFR area}$$

Start can be specified in binary, octal, 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.

$$1H \leq \text{size} \leq \text{Largest address in program space other than SFR area}$$

Size can be specified in binary, octal, 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) Two or more address areas cannot be specified with option -U.

- 7) 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

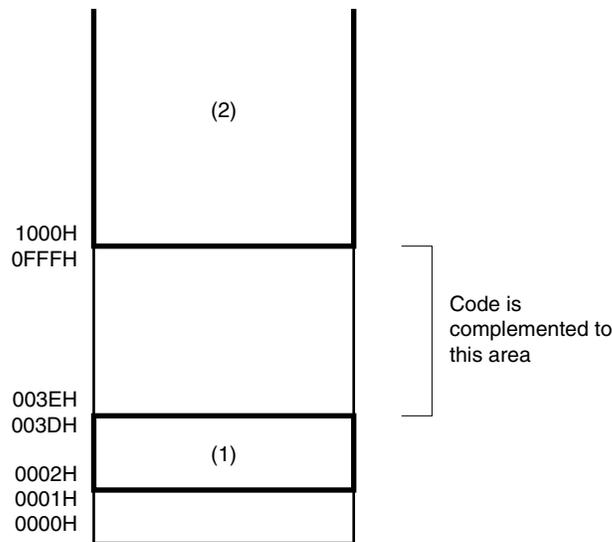
**[Example of use]**

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 to 0FFFH.

```
C>oc78K4 k4.lmf -u00h,003eh,0fc2h
```

**(5) Error list file output specification (-E/-NE)**

Syntax:                -E [output-file-name]  
                          -NE

Default assumption: -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 of use]**

Create an error list file (K4.EOC).

```
C>>oc78K4 k4.lmf -eK4.eoc
```

**(6) Parameter file specification (-F)**

Syntax:               -F file-name

Default assumption: 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 written 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 written 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 of use]**

Perform object conversion using a parameter file.

Set the contents of parameter file 78K4.POC as follows.

```
;parameter file
K4.lmf -osample.hex
-ssample.sym -r
```

Enter the following on the command line.

```
C>>oc78K4 k4.lmf -f78K4.poc
```

**(7) HEX format specification (-KI/-KIE/-KT/-KM/-KME)**

Description format: -KI  
                          -KIE  
                          -KT  
                          -KM  
                          -KME  
Default value:       -KIE

**[Function]**

Specifies format of a HEX-format object module file to be output.

**[Application]**

Use these options to specify the format of a HEX-format object module file to be output from among [Intel standard HEX format], [Intel extended HEX format], [Extended Tech format], [Motorola S-type format (standard address)] and [Motorola S-type format (32-bit address)].

**[Explanation]**

1) The address ranges specified by each of these options are as follows.

-KI:	Intel standard HEX format 0H to FFFFH (up to 64 KB)
-KIE:	Intel extended HEX format 0H to FFFFFH (up to 1 MB)
-KT:	Extended Tech format 0H to FFFFFFFFH (up to 4 GB)
-KM:	Motorola S-type format (standard address) 0H to FFFFFFFH (up to 16 MB)
-KME:	Motorola S-type format (32-bit address) 0H to FFFFFFFFH (up to 4 GB)

**[Example of use]**

Specify a Motorola S-type format (standard address) object.

```
C>>oc78k4 k4.lmf -km
```

**(8) Device file search path specification (-Y)**

Syntax:                -Y path-name

Default assumption: Device files will be read from the path determined in the following order.

- 1) <..\dev> (for the oc78K4.exe startup path)
- 2) Path by which OC78K4 was started up
- 3) Current directory
- 4) The environmental variable PATH

**[Function]**

Reads a device file from the specified path.

**[Application]**

Specify a path where a device file exists.

**[Explanation]**

- 1) If anything other than a path name is specified after option -Y, an abort error occurs.
- 2) If the path name is omitted after option -Y, an abort error occurs.
- 3) The path from which the device file is read in the order determined as follows.
  - a. Path specified by option -Y
  - b. <..\dev> (for the oc78K4.exe startup path)
  - c. Path by which OC78K4 was started up
  - d. Current directory
  - e. The environmental variable PATH

**[Example of use]**

Specify the path for the device file as directory c:\78K4\dev

```
C>>oc78K4 k4.lmf -yc:\78K4\dev
```

**(9) File separate output specification for flash ROM model (-ZF)**

Syntax: -ZF

Default assumption: Not separately output.

**[Function]**

Specifies the first address of the flash ROM area.

**[Application]**

Specify a path where a device file exists.

**[Explanation]**

- 1) Adds an option that outputs the boot area and other areas to separate HEX-format files when linking the boot area ROM program of a flash ROM model is specified.
- 2) If the -ZF option is specified, the output file type at the boot area ROM program side is "HXB", and the output file type at the side of the other programs is "HXF".

**[Example of use]**

```
C>>oc78K4 k4.lmf -zf
```

**(10) Help specification (--)**

Syntax:                --

Default assumption: 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.

**[Explanation]**

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

**[Example of use]**

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

```
C>>oc78k4 --
```

```
78K/IV Series Object Converter Vx.xx [xx xxx xx]
```

```
Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
usage : oc78k4 [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.
-kkind                :Select hex format. I;intel format IE;intel extend format
                      T;tex format M;s format ME;s-32bit format
-ydirectory         :Set device file search path.
--                    :Show this message.
```

```
DEFAULT ASSIGNMENT: -o -s -nr -kie
```

## 7.5 Option Settings in PM plus

This section describes the method for setting object converter options from PM plus.

### 7.5.1 Option setting method

Select [Object Converter Options...] from the [Tools] menu of PM plus or click  to display the <Object Converter Options> dialog box.

Object converter options can be set by inputting the required options in this dialog box.

Figure 7-6. &lt;Object Converter Options&gt; Dialog Box (&lt;&lt;Output1&gt;&gt; Tab Is Selected)

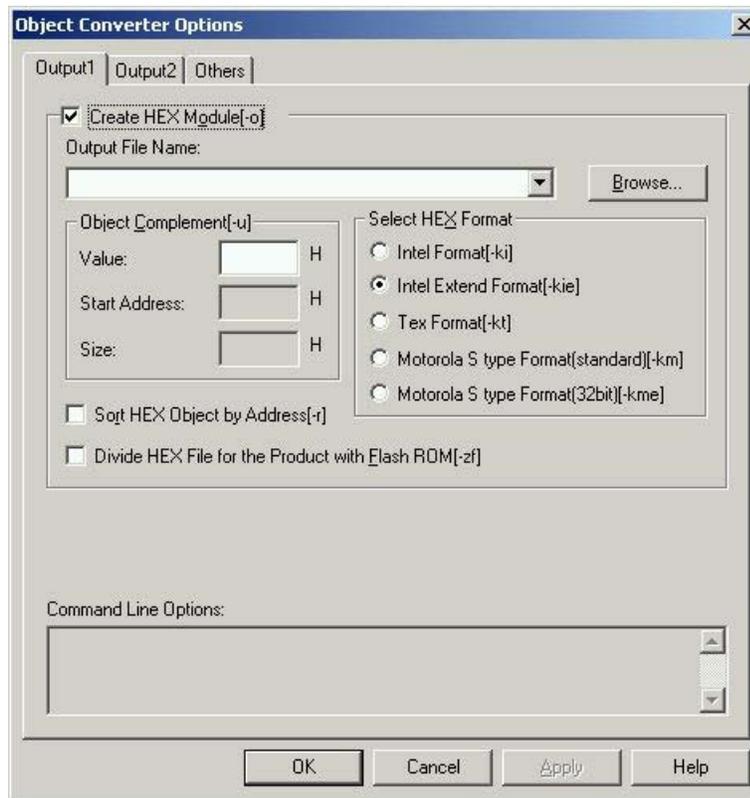
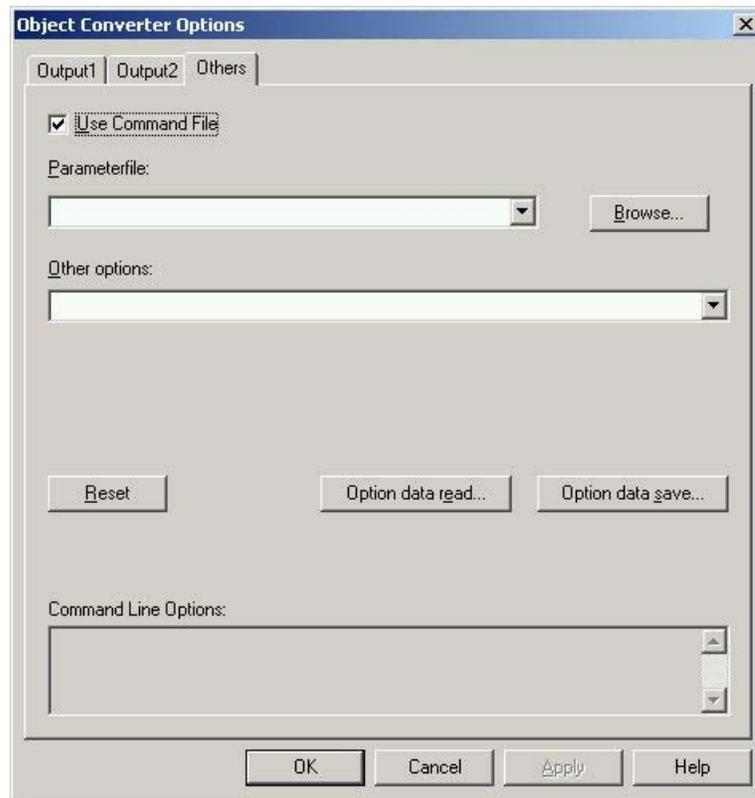


Figure 7-7. &lt;Object Converter Options&gt; Dialog Box (&lt;&lt;Output2&gt;&gt; Tab Is Selected)



Figure 7-8. &lt;Object Converter Options&gt; Dialog Box (&lt;&lt;Others&gt;&gt; Tab Is Selected)



## 7.5.2 Option settings

The various options in the <Object Converter Options> dialog box are described below.

- **Create HEX module [-o]**  
Output File Name:  
Specify the HEX format object module file output path either using the [Browse...] button or by directly inputting it.
- **Object Complement [-u]**  
Writes code beforehand in order to prevent illegal code from getting written to addresses to which no HEX format object is output and program loop from occurring.
- **Select HEX Format**  
Select the HEX format for the object to be output.
- **Sort HEX Object by Address [-r]**  
Specify this option if HEX format objects need to be sorted by address.
- **Divide HEX Files for the product with Flash ROM [-zf]**  
Divides boot area and other areas into different HEX format files for flash ROM versions and outputs these files.
- **Create Symbol Table File [-s]**  
Output File Name:  
Specify the symbol table file output path either using the [Browse...] button or by directly inputting it.
- **Create Error List File [-e]**  
Output File Name:  
To output an error list file, input the file name in the input box.  
To specify the path, use the [Browse...] button.
- **Use Command File**  
Select this check box to create a command file.
- **Parameterfile**  
Read a user-defined parameter file either using the [Browse...] button or by directly inputting it.
- **Other options**  
If wishing to specify an option other than the options that can be selected with a check box or radio button, input the option in the input box.
- **Reset**  
Resets the input contents.
- **Option data read...**  
Opens the <Option Data Read> dialog box and after the option data file has been specified, reads this file.
- **Option data save...**  
Opens the <Option Data Save> dialog box and save the option data to the option data file with a name.
- **Command Line Options**  
This edit box is read-only. The currently set option character string is displayed.

## CHAPTER 8 LIBRARIAN

The librarian edits RA78K4 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.

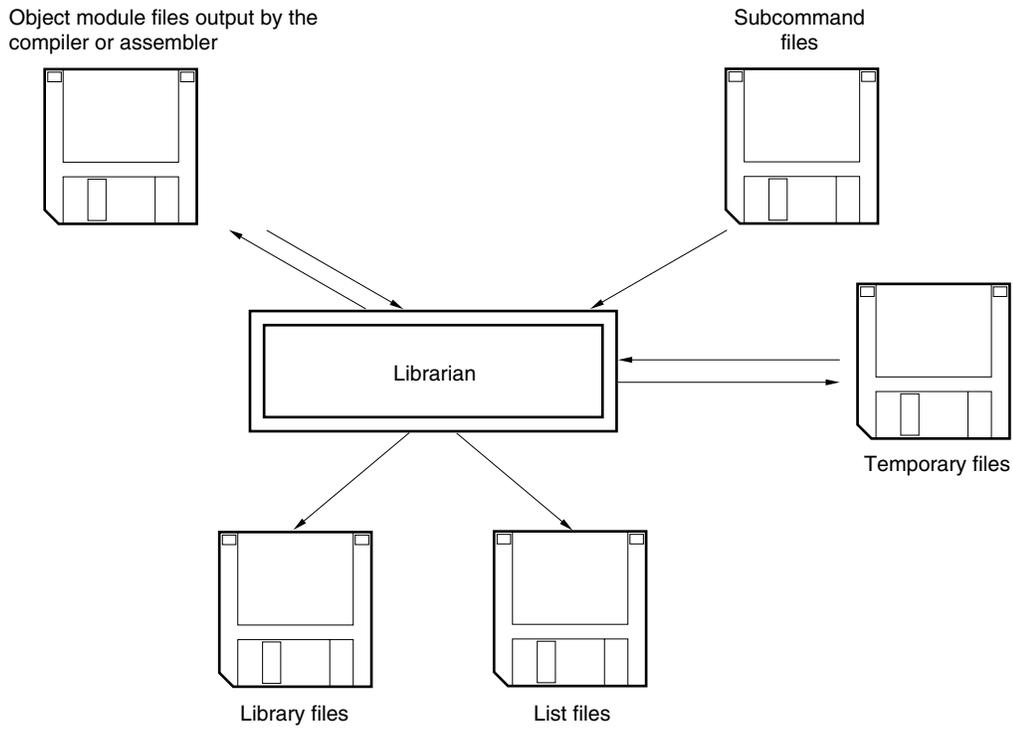
### 8.1 I/O Files of Librarian

The I/O files of the librarian are as follows.

**Table 8-1. I/O Files of Librarian**

Type	File Name	Explanation	Default File Type
Input files	Subcommand files	<ul style="list-style-type: none"><li>• These files contain the execute program command and the parameters.</li><li>• These files are created by the user.</li></ul>	None
Output files	List files	<ul style="list-style-type: none"><li>• These files are the result of output of library data.</li></ul>	.LST
I/O files	Object module files	<ul style="list-style-type: none"><li>• These are object module files output by the assembler or compiler.</li></ul>	.REL
	Library files	<ul style="list-style-type: none"><li>• These files input the library files output by the librarian and update the contents.</li></ul>	.LIB
	Temporary files	<ul style="list-style-type: none"><li>• These files are automatically generated by the librarian when forming a library. They are deleted when execution of the librarian is complete.</li></ul>	Lbxxxxx.\$\$y (y = 1 to 6)

Figure 8-1. I/O Files of Librarian



## 8.2 Functions of 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 RA78K4 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, refer to **8.5 Subcommands**.)

### (3) Output of library file data

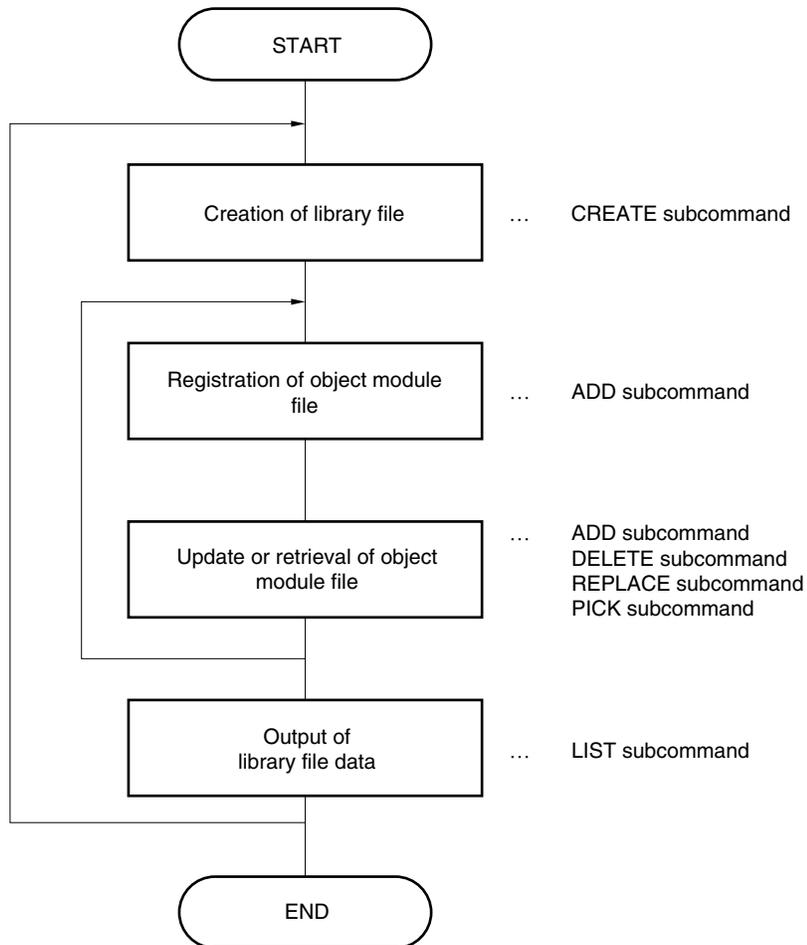
The librarian incorporates functions for the editing and output of the following items of data stored in library files.

- 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) explained above using subcommands. The librarian determines each subcommand in order while performing processing. For an explanation of the operation of subcommands, refer to **8.5 Subcommands**.

**(4) Procedure for creating a library file**

The general procedure for creating library files is as follows.

**Figure 8-2. Procedure for Creating Library File**

## 8.3 Librarian Startup

### 8.3.1 Librarian startup

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

#### (1) Startup from the command line

```
X> [path-name] lb78k4 [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>lb78k4 -l120 -lw80

**Caution** If more than one librarian option is specified, separate the options with a space. For a detailed explanation of librarian options, refer to 8.4 Librarian Options.

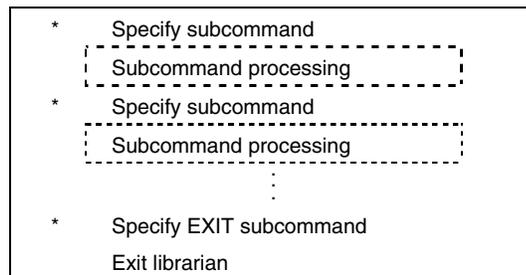
When the librarian is started up, the following startup message appears on the display.

```
78K/IV Series Librarian Vx.xx [xx xxx xx]
Copyright (C) NEC Electronics Corporation xxxx,xxxx
*
```

After an asterisk (\*), specify a librarian subcommand.

```
*create k4.lib
*add k4.lib k4main.rel k4sub.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>lb78k4Δ < 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 writing 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 (K4.SLB) using an editor.

Contents of K4.SLB

```
;library creation command
create k4.lib
add k4.lib k4main.rel &
k4sub.rel
;
exit
```

Use subcommand file K4.SLB to start up the librarian.

```
C>lb78k4 <k4.slb
```

### 8.3.2 Execution start and end messages

#### (1) Execution start message

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

```
78K/IV Series Librarian Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
*
```

#### (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 k4.lib
*add k4.lib k4main.rel k4sub.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** A non-existent librarian option is specified.

```
C>>lb78k4 -a

78K/IV Series Librarian Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
A018 Option is not recognized '-z'
Usage: LB78K4 [options]
```

In the above example, a non-existent 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 13 ERROR MESSAGES** and take action accordingly.

## 8.4 Librarian Options

### 8.4.1 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 4 types.

**Table 8-2. Librarian Options**

Number	Classification	Option	Explanation
1	List file format specification	-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	Specification of path for temporary file creation	-T	Creates a temporary file in a specified path.
3	Device file search path specification	-Y	Reads a device file from a specified path.
4	Help specification	--	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, refer to **C.5 List of Librarian Options**.

### 8.4.2 Explanation of librarian options

The following is a detailed explanation of the library options.

#### (1) List file format specification (-LW, -LL, -LF/-NLF)

##### (a) -LW

Syntax: -LW [number-of-characters]

Default assumption: -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.

##### [Explanation]

- 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)

$$72 \leq \text{number of characters printed on 1 line} \leq 2046$$

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 terminator (CR, LF).
- 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 of use]

Specify 80 as the number of characters per line in a list file.

```
C>>lb78k4 -lw80
```

**(b) -LL**

Syntax: -LL [number-of-lines]

Default assumption: -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.

**[Explanation]**

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

$$20 \leq \text{number of lines printed on 1 page} \leq 32767$$

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 of use]**

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

```
C>lb78k4 -ll20
```

**(c) -LF/-NLF**

Syntax:               -LF  
                          -NLF

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Add a form feed code to a list file.

```
C>>1b78k4 -lf
```

**(2) Specification of path for temporary file creation (-T)**

Syntax: -T path-name

Default assumption: 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.

**[Explanation]**

- 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 of use]**

Specify output of a temporary file to directory c:\TMP.

```
C>>lb78k4 -tc:\tmp
```

**(3) Device file search path specification (-Y)**

Syntax: -Y path-name

Default assumption: Device files will be read from the path determined in the following order.

- 1) <..\dev> (for the lb78k4.exe startup path)
- 2) Path by which LB78K4 was started up
- 3) Current directory
- 4) The environmental variable PATH

**[Function]**

Reads a device file from the specified path.

**[Application]**

Specify a path where a device file exists.

**[Explanation]**

- 1) If anything other than a path name is specified after option -Y, an abort error occurs.
- 2) If the path name is omitted after option -Y, an abort error occurs.
- 3) The path from which the device file is read in the order determined as follows.
  - a. Path specified by option -Y
  - b. <..\dev> (for the lb78k4.exe startup path)
  - c. Path by which LB78K4 was started up
  - d. Current directory
  - e. The environmental variable PATH

**[Example of use]**

Specify the path for the device file as directory c:\78k4\dev

```
C>>lb78k4 -yc:\78k4\dev
```



## 8.5 Subcommands

### 8.5.1 Types of subcommands

The subcommands provide detailed directions for the operation of the librarian. Subcommands are classified into eight types.

**Table 8-3. 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, refer to **APPENDIX D LIST OF SUBCOMMANDS**.

### 8.5.2 Explanation of subcommands

The following is a detailed explanation of the function and operation of each subcommand.

#### [General format of command files]

\*Subcommand [ $\Delta$ option]  $\Delta$ library-file-name [ $\Delta$ option] transaction [ $\Delta$ option]

|  
(1)

|  
(2)

(1) The library file name specified immediately before can be replaced with '.'.

(2) Transaction =  $\Delta$ object-module-file-name  $\Delta$ library-file-name [ $\nabla$ ( $\nabla$ module-name [ $\nabla$ ,...])]

**(1) CREATE**

Syntax: CREATE $\Delta$ library-file-name [ $\Delta$ transaction]

Abbreviated format: C

**[Function]**

The CREATE subcommand creates a new library file.

**[Explanation]**

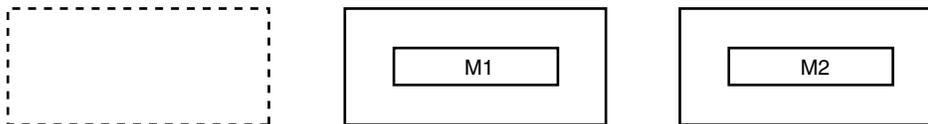
- 1) The size of the created library file becomes 0.
- 2) When a transaction is specified, a module is registered at the same time as the library file is created.
- 3) Library file name: If a file with the same name already exists, it will be overwritten.
- 4) Transaction: An object module file carrying 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 of use]**

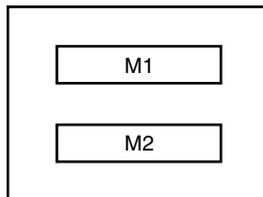
Register modules M1 and M2 at the same time as a library file is created.

```
*create k4.lib m1.rel m2.rel
```

<Before file creation>



<After file creation>  
K4.LIB



**(2) ADD**

Syntax:           ADDΔlibrary-file-name Δtransaction

Abbreviated format: A

**[Function]**

The ADD subcommand adds a module to a library file.

**[Explanation]**

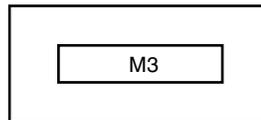
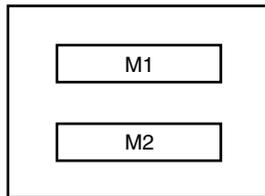
- 1) A module can be added to a library file even if no modules are currently stored in the library.
- 2) If a module with the same name as the module to be added already exists in the library file, an error occurs.
- 3) If the module to be added carries the same public symbol as the public symbol in the library file, an error occurs.

**[Example of use]**

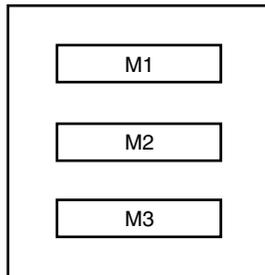
Add a module (M3) to a library file (K4.LIB).

```
*add k4.lib m3.rel
```

<Before addition>  
K4.LIB



<After addition>  
K4.LIB



**(3) DELETE**

Syntax: DELETEΔlibrary-file-name ▽(▽module-name [▽,...]▽ )

Abbreviated format: D

**[Function]**

The DELETE subcommand deletes a module from a library file.

**[Explanation]**

- 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.

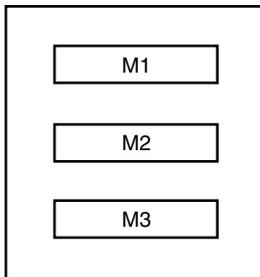
**[Example of use]**

Delete modules (M1, M3) from a library file (K4.LIB).

```
*delete k4.lib (m1.rel m3.rel)
```

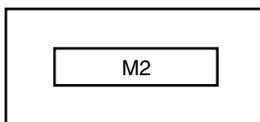
<Before deletion>

K4.LIB



<After deletion>

K4.LIB



**(4) REPLACE**

Syntax: REPLACEΔlibrary-file-nameΔtransaction

Abbreviated format: R

**[Function]**

The REPLACE subcommand replaces module in a library file with the module in other object module files.

**[Explanation]**

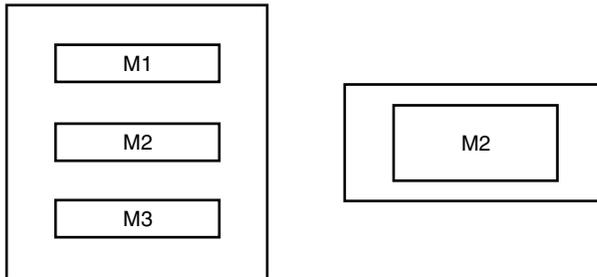
- 1) If no module in the library file has the same name as the replacement module, an error will result.
- 2) If a public symbol contained in the replacement module is the same as a public symbol in the library file, an error will occur.
- 3) The file name of the replacement object module must be the 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 of use]**

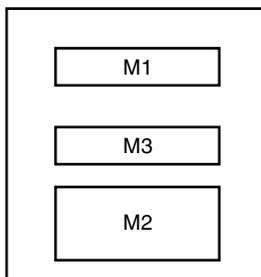
Replace a module (M2) in a library file (K4.LIB).

```
*replace k4.lib m2.rel
```

<Before replacement>  
K4.LIB



<After replacement>  
K4.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**

Syntax: PICKΔlibrary-file-name ∇(∇module-name [∇,...]∇)

Abbreviated format: P

**[Function]**

The PICK subcommand retrieves a specified module from an existing library file.

**[Explanation]**

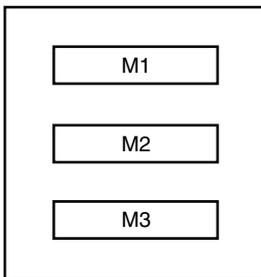
- 1) The retrieved module becomes an object module file with the file name under which it was registered in the library file.
- 2) If the specified module name does not exist in the library file, an error will result.
- 3) If an error occurs, processing is interrupted. However, if an error occurs when two or more modules are specified, the modules retrieved before the module which caused the error become available and are saved onto disk.

**[Example of use]**

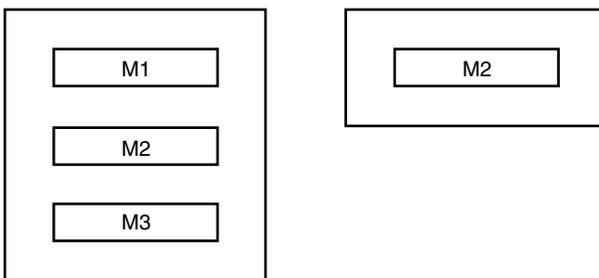
Retrieve a module (M2) from a library file (K4.LIB).

```
*pick k4.lib (m2.rel)
```

<Before pickup>  
K4.LIB



<After pickup>  
K4.LIB



**(6) LIST**

Syntax: LIST [ $\Delta$ option]  $\Delta$ library-file-name [ $\nabla$ ( $\nabla$ module-name [ $\nabla$ ,...] $\nabla$ )]

Option : -PUBLIC/-NOPUBLIC  
-O  $\nabla$ file-name

Abbreviated format: L

**[Function]**

The LIST subcommand outputs data on modules in a library file.

**[Explanation]**

- 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 of use]**

Output a module data in a library file (K4.LIB) to a list file (K4.LST). Specify option -P so that public symbol data will be output.

```
*list -p -ok4.lst k4.lib
```

List file (K4.LST) is referenced.

```
78K/IV Series librarian Vx.xx      DATE : xx xxx xx                PAGE  1
```

```
LIB-FILE NAME : K4.LIB          (xx xxx xx)
```

```
0001  K4MAIN.REL          (xx xxx xx)
```

```
MAIN                                START
```

```
NUMBER OF PUBLIC SYMBOLS :  2
```

```
0002  K4SUB.REL          (xx xxx xx)
```

```
CONVAH
```

```
NUMBER OF PUBLIC SYMBOLS :  1
```

**(7) HELP**

Syntax: HELP

Abbreviated format: H

**[Function]**

The HELP command displays a help message on the display.

**[Explanation]**

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 of use]**

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**

Syntax: EXIT

Abbreviated format: E

**[Function]**

The EXIT subcommand exits the librarian.

**[Explanation]**

Use this subcommand to exit the librarian.

**[Example of use]**

Exit the librarian.

\*exit

## 8.6 Option Settings in PM plus

This section describes the method for specifying library files from PM plus.

### 8.6.1 Option setting method

Select [l lb78k4p] from [Start external tool] from the [Tools] menu of PM plus or click  to display the <Library File Name> dialog box. After specifying the path and the file name, press [Next] to display the <Subcommand> dialog box.

The various librarian options can be set by inputting the required option in this dialog box.

Figure 8-3. &lt;Library File Name&gt; Dialog Box

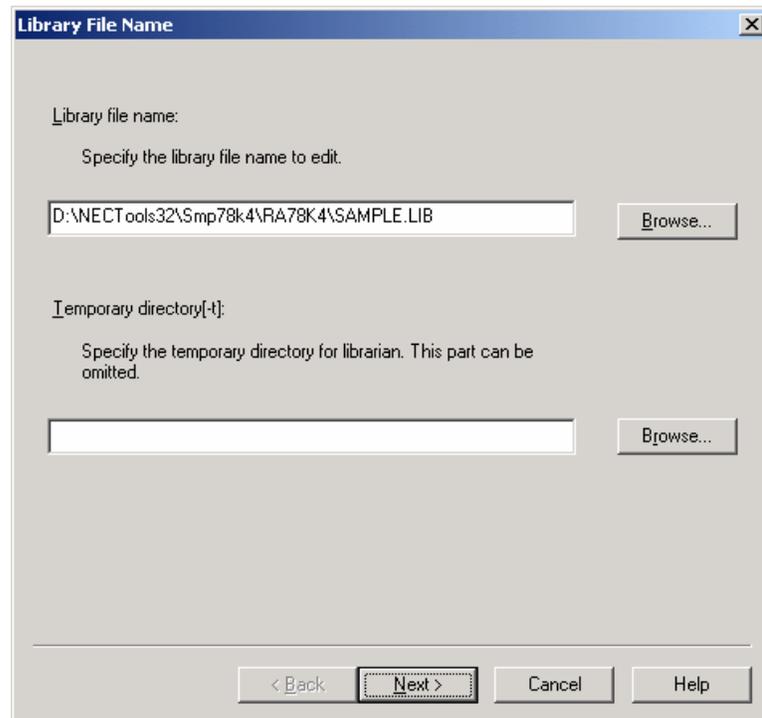
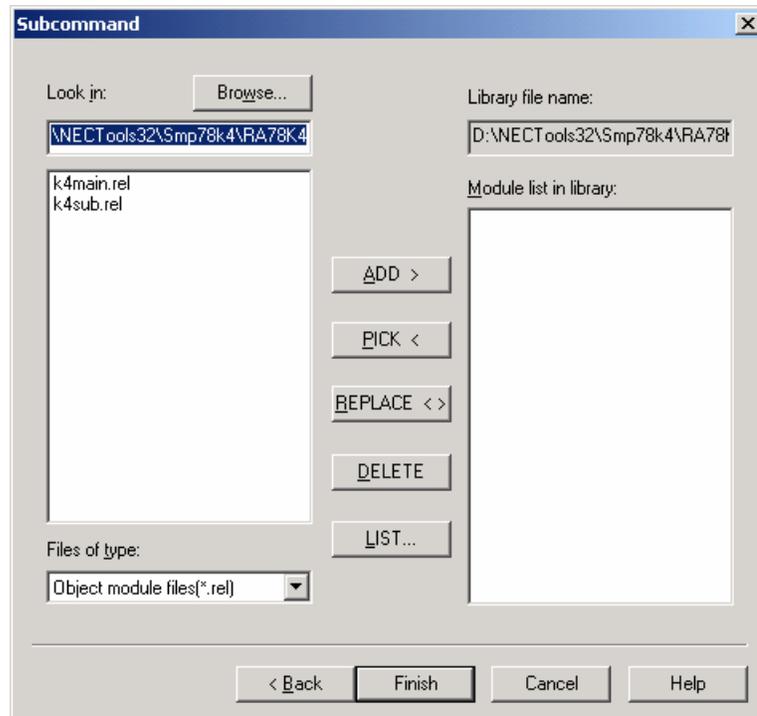


Figure 8-4. &lt;Subcommand&gt; Dialog Box



### 8.6.2 Option settings

The various options in the <Librarian> dialog box are described below.

- **L**ibrarian file name  
Specify the library file path for editing either using the [Browse...] button or by directly inputting it.
- **T**emporary directory [-t]  
Specify the path of the temporary file to be created either using the [Browse...] button or by directly inputting it.
- **L**ook **i**n  
Select the object module files to be made into a library either using the [Browse...] button or by directly inputting it.
- **A**DD  
Add a module to an existing library file.
- **P**ICK  
Retrieve the specified module from an existing library file.
- **R**EPLACE  
Replace an existing library file module with the module of another object module file.
- **D**ELETE  
Delete a module from an existing library file.
- **L**IST  
Outputs data on modules in a library file.

## CHAPTER 9 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.

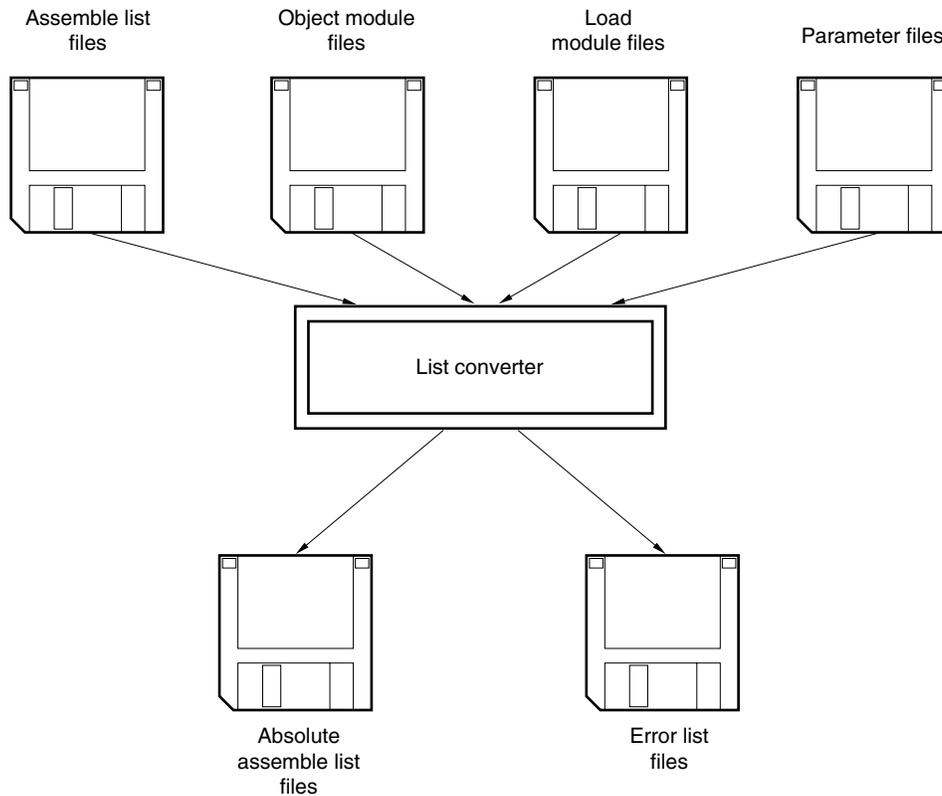
### 9.1 I/O Files of List Converter

The I/O files of the list converter are as shown below.

**Table 9-1. I/O Files of List Converter**

Type	File Name	Explanation	Default File Type
Input files	Object module files	<ul style="list-style-type: none"> <li>These are binary files including relocation data and symbol data regarding machine language data and machine language location addresses.</li> </ul>	.REL
	Assemble list files	<ul style="list-style-type: none"> <li>These are files containing assembly data such as assemble lists and cross-reference lists.</li> </ul>	.PRN
	Load module files	<ul style="list-style-type: none"> <li>These are binary image files which contain object code as a result of linking.</li> </ul>	.LMF
	Parameter files	<ul style="list-style-type: none"> <li>These files contain the parameters for the executed program.</li> <li>These files are created by the user.</li> </ul>	.PLV
Output files	Absolute assemble list files	<ul style="list-style-type: none"> <li>This is a list file which embeds actual addresses in relocatable addresses and symbols in the input file.</li> </ul>	.P
	Error list files	<ul style="list-style-type: none"> <li>These are files containing error data generated during list conversion.</li> </ul>	.ELV

**Figure 9-1. I/O Files of List Converter**



## 9.2 Functions of 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

```

16 16 0FFD20          HDTSA: DS      1
17 17 0FFD21          STASC: DS      2
18 18
19 19 -----          CODE   CSEG   AT 0H
20 20 000000 R0400      MAIN:   DW      START
21 21
22 22 -----          CSEG
23 23 000000 09C1FF00      LOCATION      15
24 24 000004 2BCC00      START:  MOV      RFM, #00
25 25 000007 092000FE0F      MOVG   SP, #0FFE00H
26 26 00000C 2BC400      MOV    MM, #00

```

## • Absolute assemble list

```

16 16 0FFD20          HDTSA: DS      1
17 17 0FFD21          STASC: DS      2
18 18
19 19 -----          CODE   CSEG   AT 0H
20 20 000000 R8400      MAIN:   DW      START
21 21
22 22 -----          CSEG
23 23 000080 09C1FF00      LOCATION      15
24 24 000084 2BCC00      START:  MOV      RFM, #00
25 25 000087 092000FE0F      MOVG   SP, #0FFE00H
26 26 00008C 2BC400      MOV    MM, #00

```

**Example 2.** Embedding of object codes

- Assemble list

```

16    16 0FFD20                HDTSA: DS      1
17    17 0FFD21                STASC: DS      2
18    18
19    19 -----                CODE   CSEG    AT 0H
20    20 000000 R0400          MAIN:   DW      START
21    21
22    22 -----                CSEG
23    23 000000 09C1FF00          LOCATION      15
24    24 000004 2BCC00          START:  MOV     RFM,#00
25    25 000007 092000FE0F      MOVG    SP,#0FFE00H
26    26 00000C 2BC400          MOV     MM,#00
    
```

- Absolute assemble list

```

16    16 0FFD20                HDTSA: DS      1
17    17 0FFD21                STASC: DS      2
18    18
19    19 -----                CODE   CSEG    AT 0H
20    20 000000 R8400          MAIN:   DW      START
21    21
22    22 -----                CSEG
23    23 000080 09C1FF00          LOCATION      15
24    24 000084 2BCC00          START:  MOV     RFM,#00
25    25 000087 092000FE0F      MOVG    SP,#0FFE00H
26    26 00008C 2BC400          MOV     MM,#00
    
```

## 9.3 List Converter Startup

### 9.3.1 List converter startup

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

#### (1) Command-line startup

```
X>lcnv78k4 [ $\Delta$ option]... $\Delta$ input-file-name [ $\Delta$ option]... [ $\Delta$ ]
|         |         |         |         |
(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>lcnv78k4 k4main -lk4.lmf

- 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, refer to 9.4 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.

**Table 9-2. Type of Specification File When List Converter Is Started**

File Name	Type
Object module type	.REL
Load module file	.LMF

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.

**Caution** Option -F cannot be specified when PM plus is used.

```
X>lcnv78k4  [ $\Delta$ input-file-name]  $\Delta$ -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 writing the contents of a parameter file are as follows.

```
[[ [ $\Delta$ ] option [ $\Delta$ option] ... [ $\Delta$ ] $\Delta$ ]]...
```

- 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 written after the option.
- 3) Write 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 (K4.PLV) using an editor.

Contents of K4.PLV

```
;parameter file
k4main -lk4.lmf
-ek4.elv
```

Use parameter file (K4.PLV) to start up the list converter.

```
C>lcnv78k4 -fk4.plv
```

### 9.3.2 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/IV Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx

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** A non-existent list converter option is specified.

```
C><u>lcnv78k4 k4main -a
```

When the list converter outputs an error message and aborts list conversion, look for the cause in **CHAPTER 13 ERROR MESSAGES** and take action accordingly.

## 9.4 List Converter Options

### 9.4.1 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 6 types.

**Table 9-3. List Converter Options**

Number	Classification	Option	Explanation
1	Object module file input specification	-R	Inputs an object module file.
2	Load module file input specification	-L	Inputs a load module file.
3	Absolute assemble list file output specification	-O	Specifies output of an absolute assemble list file.
4	Error list file output specification	-E	Outputs an error list file.
5	Parameter file specification	-F	Inputs the input file name and options from a specified file.
6	Help specification	--	Displays a help message on the display.

**Remark** For details of the list converter options, refer to **C.6 List of List Converter Options**.

### 9.4.2 Explanation of list converter options

This section contains detailed explanations of each list converter option.

#### (1) Object module file input specification (-R)

Syntax: -R [input-file-name]

Default assumption: -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.

#### [Explanation]

- 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 of use]

Assemble list file name is K4MAIN.PRN, the object module file name is SAMPLE.REL, and the load module file name is K4.LMF.

```
C><u>lcnv78k4 k4main -rsample.rel -lk4.lmf
```

**(2) Load module file input specification (-L)**

Syntax:               -L [input-file-name]

Default assumption: -L assemble-list-file-name.LMF

**[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 ".LMF", specify option -L.

**[Explanation]**

- 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 '.LMF' and input the file.

**[Example of use]**

Assemble list file name is K4MAIN.PRN and the load module file name is SAMPLE.LMF.

```
C>>lcnv78k4 k4main -lsample.lmf
```

**(3) Absolute assemble list file output specification (-O)**

Syntax: -O [output-file-name]

Default assumption: -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.

**[Explanation]**

- 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 of use]**

Create an absolute assemble list file (SAMPLE.P).

```
C>lcnv78k4 k4main -osample.p -lk4.lmf
```

**(4) Error list file output specification (-E/-NE)**

Syntax:                -E [output-file-name]  
                          -NE

Default assumption: -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.

**[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) 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 of use]**

**Example** Create an error list file (SAMPLE.ELV).

```
C>>lcnv78k4 k4main -esample.elv
```

The error list file (SAMPLE.ELV) is referenced.

```
*** WARNING W101 Load module file is older than object module file 'K4MAIN.LMF,  
K4MAIN.REL'  
Pass1: start  
*** ERROR A105 Segment name is not found in load module file 'DATA'
```

**(5) Parameter file specification (-F)**

Syntax:                -F file-name

Default assumption: Options and input file names can only be entered on the command 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.

**[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) 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 written 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 written 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 of use]**

Start up list converter using a parameter file.

The contents of the parameter file (K4.PLV) are as follows.

```
;parameter file
k4main -lk4.lmf
-ek4.elv
```

Enter the following on the command line.

```
C>>lcnv78k4 -fk4.plv
```

**(6) Help specification (--)**

Syntax:            --  
Default assumption: 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.

**[Explanation]**

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

**[Example of use]**

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

```
C>>lcnv78k4 --
```

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

```
usage : LCNV78K4 [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.  
--      :Show this message.
```

## 9.5 Option Settings in PM plus

This section describes the method for setting list converter options from PM plus.

### 9.5.1 Option setting method

Select [List converter options...] from the [Tools] menu of PM plus or click  to display the <List Converter Options> dialog box.

List converter options can be set by inputting the required options in this dialog box.

Figure 9-2. &lt;List Converter Options&gt; Dialog Box (When &lt;&lt;Output&gt;&gt; Tab Is Selected)

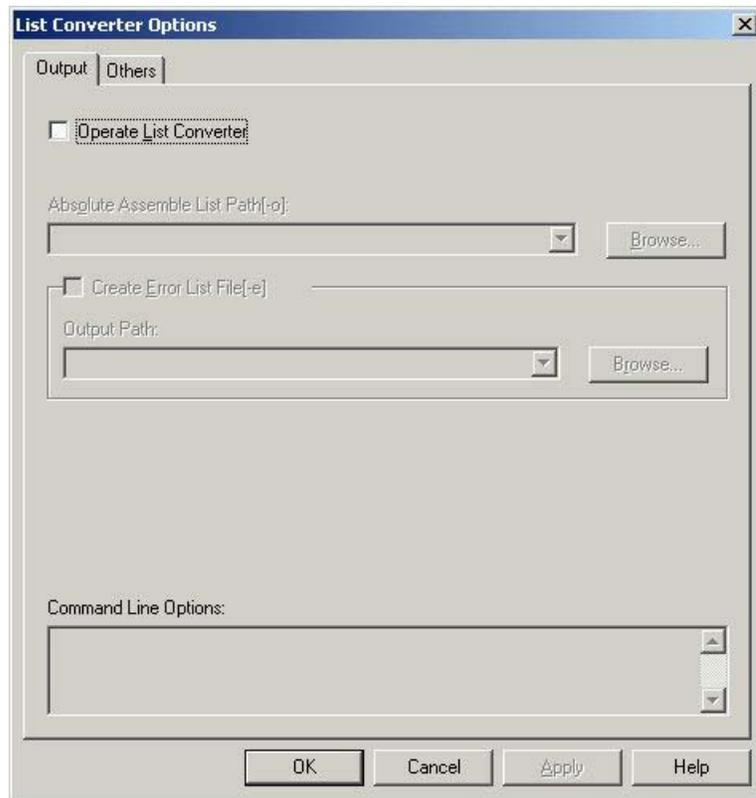
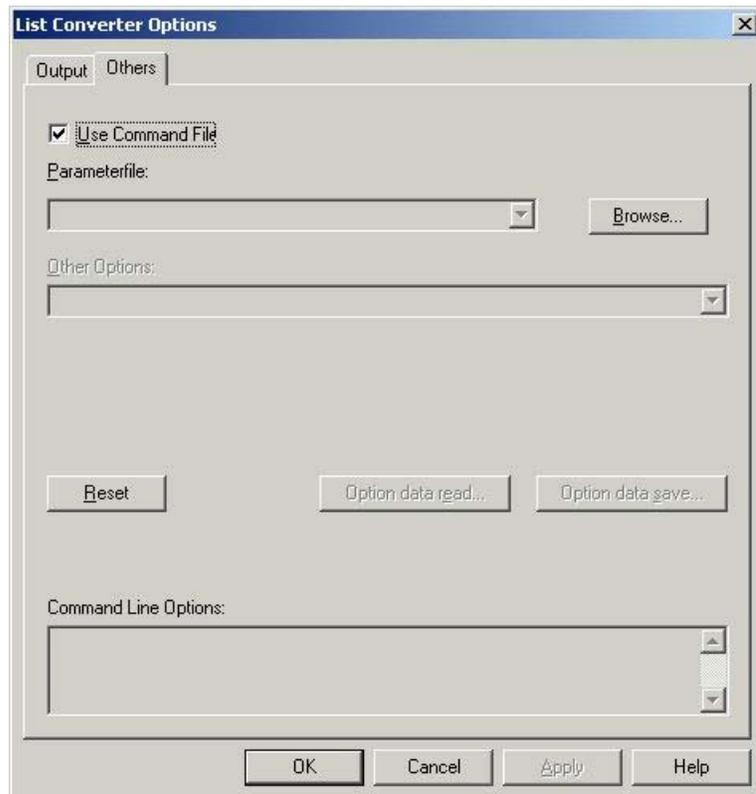


Figure 9-3. &lt;List Converter Options&gt; Dialog Box (When &lt;&lt;Others&gt;&gt; Tab Is Selected)



### 9.5.2 Option settings

The various options in the <List Converter Options> dialog box are described below.

- **Operate List Converter**  
Select this check box to start the list converter.
- **Absolute Assemble List Path [-o]**  
Specify the absolute assemble list path either using the [Browse...] button or by directly inputting it.
- **Create Error List File [-e]**  
**Output Path**  
To output an error list file, input the file name in the input box.  
To specify the path, use the [Browse...] button.
- **Use Command File**  
Select this check box to create a command file.
- **Parameterfile**  
Read a user-defined parameter file selected either using the [Browse...] button or by directly inputting it.
- **Other Options**  
If wishing to specify an option other than the options that can be selected with a check box or radio button, input the option in the input box.
- **Reset**  
Resets the input contents.
- **Option data read...**  
Opens the <Option Data Read> dialog box and after the option data file has been specified, reads this file.
- **Option data save...**  
Opens the <Option Data Save> dialog box and save the option data to the option data file with a name.
- **Command Line Options**  
This edit box is read-only. The currently set option character string is displayed.

## CHAPTER 10 ECC GENERATOR

The ECC generator inputs the HEX-format object module file that is output by the object converter and outputs a HEX-format object module file to which an ECC (Error Correcting Code) is added. This output file is referred to as the ECC-supported HEX-format object module file.

The ECC-supported HEX-format object module file is a HEX object for writing data to a PROM-type microcontroller using a PROM programmer or to a flash memory-type microcontroller using a flash programmer. It cannot be used for an in-circuit emulator.

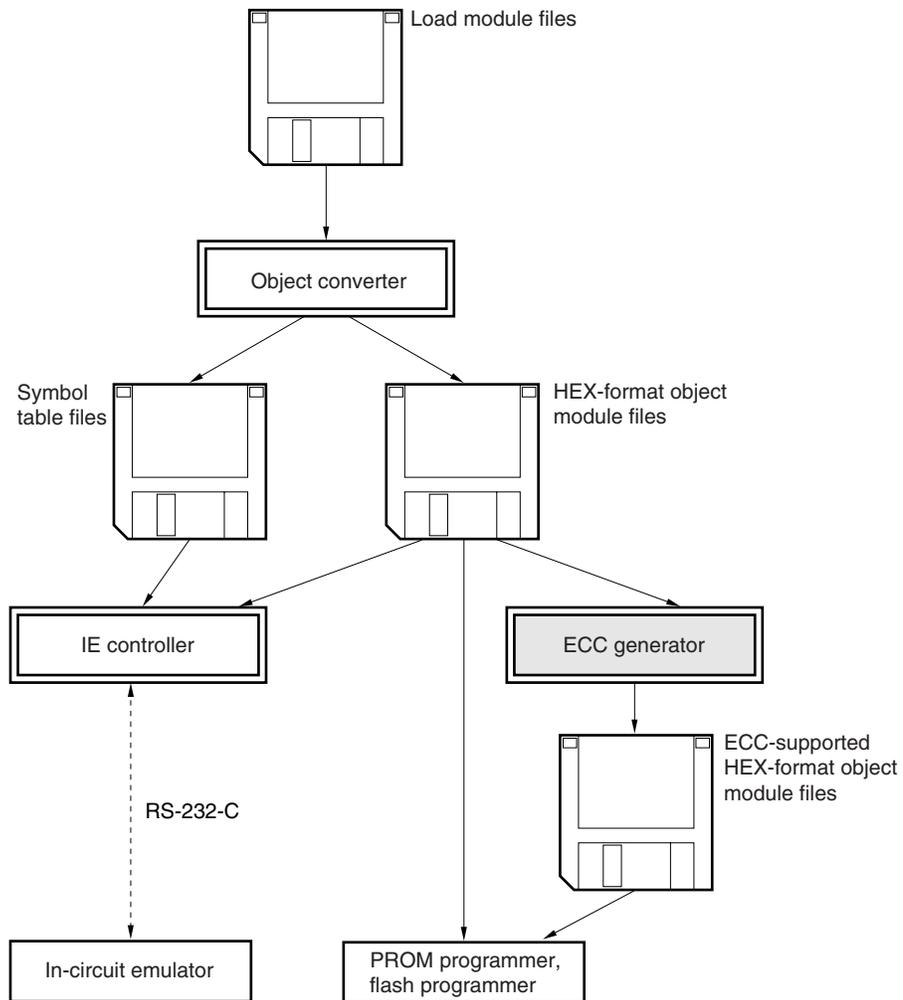
**Remark** ECC is an abbreviation of Error Correcting Code.

The ECC improves the reliability of the data by correcting code errors that are written in the microcontroller. The ECC is written in a specified address in the microcontroller.

10.1 Position of ECC Generator

Figure 10-1 Position of ECC Generator shows the position of the ECC generator.

Figure 10-1. Position of ECC Generator



## 10.2 Product Contents

**Table 10-1 Supplied Files** shows the files that are supplied in the ECC generator.

The ECC generator is included in the assembler package, which is for the devices that support the ECC.

**Table 10-1. Supplied Files**

Program Name	File Name	File Type
ECC generator	ECCGEN.EXE	Command file
	ECCGEN.HLP	Help file

**Remark** The command file is the file that is read in the memory first when programs are activated.

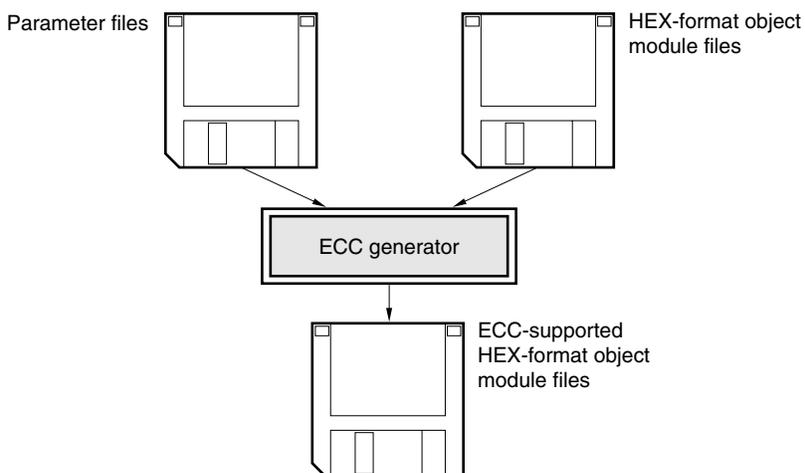
### 10.3 I/O Files of ECC Generator

The I/O files of the ECC generator are shown in **Table 10-2 I/O Files of ECC Generator**.

**Table 10-2. I/O Files of ECC Generator**

Type	File Name	Explanation	Default File Type
Input files	HEX-format object module files	These files are output by the object converter.	.HEX
	Parameter files	These files contain the parameters for the executed program. These files are created by the user.	.PEC
Output files	ECC-supported HEX-format object module files	These are HEX-format files created by adding the ECC and the ECW code to the HEX-format object module files.	.ECG

**Figure 10-2. I/O Files of ECC Generator**



**10.3.1 ECC-supported HEX-format object module file**

The ECC-supported HEX-format object module file consists of the following data and codes:

- Input HEX data
- ECC
- ECW

**(1) Input HEX data**

Input HEX data refers to the hexadecimal data contained in the input HEX-format object module file. The ECC generator outputs HEX data as is the range between the start and end addresses (the 32 KB area of 0000H to 7FFFH) of the input HEX data.

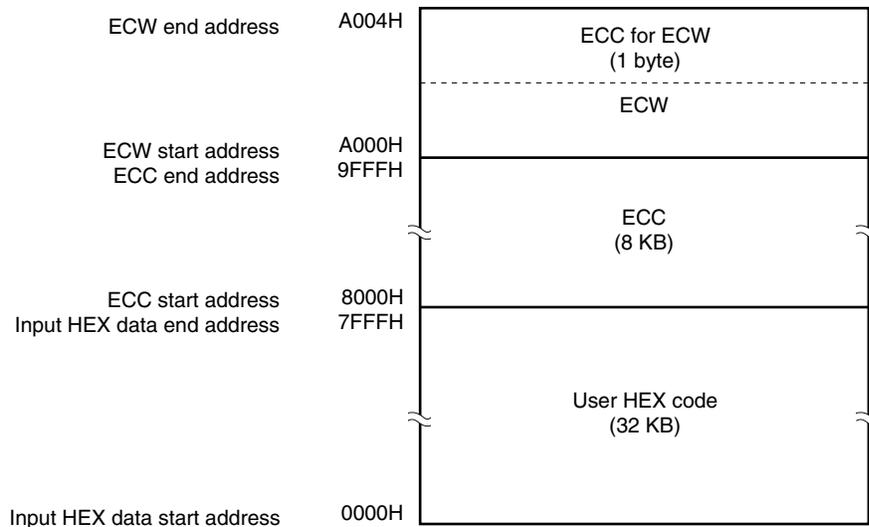
**(2) ECC**

The ECC is output to addresses 8000H to 9FFFH.

**(3) ECW**

The ECW is information indicating whether or not the ECC is to be used and is output to addresses A000H to A004H. The ECC for ECW is output to address A004H.

**Figure 10-3. Address Allocation**



### 10.3.2 ECC generator startup

The ECC generator may be started up in either of the following two ways.

#### (1) Startup from the command line

```
X> [path name] eccgen [[Δoption] ...] Δinput file name [[Δoption]...] [Δ]
|       |           |       |           |           |
<1>  <2>       <3>   <4>           <5>       <4>
```

- <1> Current drive name
- <2> Current directory name
- <3> ECC generator command file name
- <4> This contains detailed directions for the operation of the ECC generator.
- <5> HEX-format object module file

**Example** `C>eccgen k4.hex -osample.ecg`

- Remarks**
1. If two or more ECC generator options are to be specified, separate the options with a space. For a detailed explanation of ECC generator options, see **10.4 ECC Generator Options**.
  2. The startup command must be specified on a single line. Up to 127 characters may be described in a command line.
  3. Only a disk-type file name can be specified as "input file name".

**(2) Startup from a parameter file**

Use the parameter file when the data required to start up the ECC generator will not fit on the command line, or when the same ECC generator option is specified repeatedly each time ECC generator is executed.

To start up the ECC generator from a parameter file, specify the parameter file specification option (-F) on the command line.

Start up the ECC generator from a parameter file as follows.

```
X> [path name] eccgen [ $\Delta$ input file name] $\Delta$ -F parameter file name
                        |           |
                        (1)        (2)
```

<1> Parameter file specification option

<2> A file which includes the data required to start up the ECC generator

- Remarks**
1. An editor is used to create the parameter file.
  2. The rules for describing the contents of a parameter file are as follows.

```
[[ $\Delta$ ] option [ $\Delta$ option] ... [ $\Delta$ ] $\Delta$ ] ...
```

If “input file name” is omitted from the command line, only one input file name can be specified in the parameter file.

The “input file name” can also be described after the option.

Describe in the parameter file all ECC generator options and output file names that should be specified in the command line.

For details of parameter files, see the explanation in **10.4 ECC Generator Options**.

**Example** Create the parameter file SAMPLE.PEC using an editor.

- Contents of SAMPLE.PEC

```
;parameter file
k4.hex -osample.ecg
```

- Use parameter file SAMPLE.PEC to start up the ECC generator.

```
C>eccgen -fsample.pec
```

### 10.3.3 Execution start and end messages

#### (1) Execution start message

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

```
ECC Generator Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx
```

#### (2) Execution end message

If no error is found as a result of an ECC generation operation, the ECC generator outputs the following message to the display and returns control to the operating system.

```
ECC Generation complete.
```

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

**Example** The following message is output to the display when input file name is omitted.

```
C><u>eccgen

ECC Generator Vx.xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx

Usage : ECCGEN option input-file option
Please enter ' ECCGEN -' , if you want help messages.
Program aborted.
```

When the ECC generator outputs an error message and aborts ECC generation, look for the cause in **13.8 ECC Generator Error Messages** and take action accordingly.

## 10.4 ECC Generator Options

### 10.4.1 Types of ECC generator options

The ECC generator options are detailed instructions for the operation of the ECC generator. ECC generator options are classified into 5 types.

**Table 10-3. ECC Generator Options**

Number	Classification	Option	Explanation
1	Specify ECC-supported HEX-format object module file name	-O	Specifies the name of an ECC-supported HEX-format object module file and/or its output destination.
2	Specify parameter file	-F	Inputs the input file name and options from a specified file.
3	Specify address	-A	Specifies the data addresses to be used in the ECC generator.
4	Specify forced ECG file output	-J	Specifies whether to output the ECG file or not.
5	Specify help	--	Outputs a help message on the display.

The above table is provided as an introduction to the ECC generator options. When actually using the ECC generator options, see **C.7 List of ECC Generator Options**. In this list, the description format of each option and the relationships between options are outlined.

**Remark** Option names used in the ECC generator are not case-sensitive.

### 10.4.2 Explanation of ECC generator options

This section contains detailed explanations of each ECC generator option.

#### (1) ECC-supported HEX-format object module file name specification (-O)

Syntax:                   -O [output file name]  
Default assumption:   -O input file name.ECG

##### [Application]

Option -O is used to change the output file name and/or output destination of an ECC-supported HEX-format object module file.

##### [Explanation]

- 1) Either a disk-type or device-type file name can be specified as "output file name". However, the device CLOCK cannot be specified.
- 2) If only a path name is specified as "output file name", the ECC-supported HEX-format object module file will be output to the specified path under the name "input file name.ECG".
- 3) If "output file name" is omitted from the -O option specification or if the -O option is omitted, "input file name.ECG" is assumed to be the output file name and the ECC-supported HEX-format object module file will be output to the current directory.

##### [Example of use]

Output an ECC-supported HEX-format object module file "SAMPLE.ECG".

```
C>eccgen k4.hex -osample.ecg
```

```
ECC Generator Vx. xx [xx xxx xx]  
Copyright (C) NEC Electronics Corporation xxxx
```

```
ECC Generation complete.
```

**(2) Parameter file specification (-F)**

Syntax:                -F file name

Default assumption: Options and input file names can only be entered on the command line.

**[Application]**

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

**[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) "file name" cannot be omitted. If only the primary name of the file name is specified, the file type "PEC" is assumed to have been omitted.
- 3) Nesting of parameter files is not allowed. 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 or input file names described in a parameter file will be expanded at the position specified for the parameter file on the command line.
- 7) All characters entered after ';' and before [↵] or EOF will be interpreted as comments.
- 8) More than one -F option cannot be specified.

**[Example of use]**

Start up the ECC generator using a parameter file.

- Contents of parameter file "SAMPLE.PEC"

```
;parameter file
k4.hex -osample.ecg
```

- Enter the following on the command line.

```
C>eccgen -fsample.pec
```

```
ECC Generator Vx. xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx
```

```
ECC Generation complete.
```

**(3) Address specification (-A)**

Syntax: -A [address specification 1][,[address specification 2][,address specification 3[address specification 4]]]

Default assumption: The assumption as follows.

- Input HEX data end address is 7FFFH
- ECC error correction code start address is 8000H
- ECW data start address is A000H
- ECW data error correction code start address is A004H

**[Application]**

The following addresses used in the ECC generator should be specified.

- Input HEX data end address
- ECC error correction code start address
- ECW data start address
- ECW data error correction code start address

**[Explanation]**

- 1) "Address specification 1" is a specification to change the input HEX data end address.
- 2) "Address specification 2" is a specification to change the ECC error correction code start address.
- 3) "Address specification 3" is a specification to change the ECW data start address.
- 4) "Address specification 4" is a specification to change the ECW data error correction code start address.
- 5) When only -A is specified or when an illegal character string is specified, an abort error (A012) will occur.
- 6) An abort error (A051) will occur when:
  - Address specification 1 is set to 0
  - Address specification 1 is not a multiple of 4
- 7) An abort error (A052) will occur when:
  - Address specification 1  $\geq$  Address specification 2
- 8) An abort error (A053) will occur when:
  - Address specification 2  $\geq$  Address specification 3
- 9) An abort error (A054) will occur when:
  - Address specification 3  $\geq$  Address specification 4

**[Relations to other options]**

None

**(4) Forced ECG file output specification (-J/-NJ)**

Syntax:                -J  
                          -NJ

Default assumption: -NJ

**[Application]**

Specifies whether to forcibly output the ECG file or not.

**[Explanation]**

- 1) Option -J specifies the output of the ECG file when there is a fatal error.
- 2) Option -NJ makes option -J unavailable.

**(5) Help specification (--)**

Syntax:                --

Default assumption: No display

**[Application]**

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

**[Explanation]**

- 1) When option -- is specified, all other ECC generator options are unavailable.
- 2) If the help file is not on the same path as the executable program, no help message will be displayed.

**[Example of use]**

The help message is output on the display.

SR\_data indicates the ECW data.

```
C>>eccgen --
```

```
ECC Generator Vx.xx [xx xxx xx]
```

```
  Copyright (C) NEC Electronics Corporation xxxx
```

```
Usage: X>ECCGEN [ [option] ...] input-file [ [option] ...] [  ]
```

Options are as follows:

```
-o [file]               Specify the OUTPUT HEX-format object module file
  Example:    -oOUT.HEX
  Default:    -o
-ffile                 @ Input option or input-file name from specified file
  Example:    -fPARAMNAME.PEC
  Default:    None
-a [Hex_end_address]  [, [ECC_start_address]
  [,SR_data_start_address [,SR_data_ECC_address] ] ]
                      Specify Hex_end_address/ECC_start_address
                      /SR_data_start_address/SR_data_ECC_address.
  Default:    -a7fffh, 8000h, 0a000h, 0a004h
-j/-nj                : Create object file if fatal error occurred/Not.
--                     Show this message.
```

## 10.5 Utilization of ECC Generator

### 10.5.1 How to execute operations with efficiency (use of EXIT status function)

On completion of ECC generation processing, the ECC generator passes error levels of errors that occurred during the processing to the operating system as EXIT statuses.

The following two EXIT statuses are available:

EXIT Status	Error
0	Program terminated normally
2	An abort error has occurred

By using these EXIT statuses in a batch file, the ECC generation operation may be carried out efficiently.

### 10.5.2 How to interrupt program execution

Processing by the ECC generator can be interrupted by inputting CTRL-C. The timing of CTRL-C input differs depending on the setting of AUTOEXEC.BAT. If "break = on" is specified in the AUTOEXEC.BAT file, control will be returned to the operating system irrespective of the CTRL-C input timing. If "break = off" is specified, control will be returned to the operating system only while being displayed.

When the ECC generation program is interrupted, the ECC-supported HEX-format object module file will not be output.

### 10.5.3 How to start ECC generator efficiently

When the command line cannot include all the specification information necessary for ECC generator startup, and when the same information must be specified repeatedly in each startup, a parameter file should be used.

Refer to **10.4.2 Explanation of ECC generator options** for the details of parameter files.

#### [Application example]

Start up the ECC generator using a parameter file.

- Contents of parameter file SAMPLE.PEC

```
;parameter file
k4.hex -osample.ecg
```

- Enter the following on the command line.

```
C>eccgen -fsample.pec

ECC Generator Vx. xx [xx xxx xx]
Copyright (C) NEC Electronics Corporation xxxx

ECC Generation complete.
```

## CHAPTER 11 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
  
- Lists output by the list converter
  - Absolute assemble list
  - Error list

## 11.1 Lists Output by 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

### 11.1.1 Assemble list file headers

The header is always output at the beginning of an assemble list file.

#### [Output format]

```

78K/IV Series Assembler (1)Vx.xx (2)           Date:(3)xx xxx xxxx Page:(4)  1
(5)
Command: (6)-c4038 k4main.asm -o -p -e
Para-file:(7)
In-file: (8)K4MAIN.ASM
Obj-file: (9)K4MAIN.REL
Prn-file:(10)K4MAIN.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

11.1.2 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
      (3) (4)
(1)1  (2)1                (5)
(1)2  (2)2                (5)      NAME      SAMPM
(1)3  (2)3                ;*****
      4      4            ;*
      5      5            ;*      HEX -> ASCII Conversion Program
      6      6            ;*
      7      7            ;*      main-routine
      8      8            ;*
      9      9            ;*****
      .
      .
      .
      25  25 (6)00000C (8)0000000000      MOVW      MM,#00
      000011 0000
(7)*** ERROR F202, STNO 25 ( 0) Illegal operand
      26  26 (6)000013 (8)09C0F708      MOV      STBC,#08H
      .
      .
      .
    
```

Segment informations:

```

      ADRS      LEN      NAME
(9)0FFD20 (10)000003H (11)DATA
(9)000000 (10)000002H (11)CODE
(9)000000 (10)00002EH (11)?CSEG
    
```

Target chip : (12)uPD784026

Device file : (13)Vx.xx

Assembly complete, (14)1 error(s) and (15)0 warning(s) found. ( (16)25)

**[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)	RA78K4 target device
(13)	Device file version no.
(14)	Number of fatal errors
(15)	Number of warnings
(16)	Final error line

11.1.3 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) FFD20H	ADDR		(4) HDTSA	(1) 0H	(2) ADDR	(3) PUB	(4) MAIN
	MOD		(4) SAMPM	(1) 4H	(2) ADDR	(3) PUB	(4) START
(1) FFD21H	ADDR		(4) STASC				

[Explanation of output items]

Item	Details																																								
(1)	Symbol value																																								
(2)	<p>Symbol attributes</p> <table border="0"> <tr> <td>CSEG:</td> <td>Code segment name</td> <td>BIT:</td> <td>BIT attribute symbol (addr.bit)</td> </tr> <tr> <td>DSEG:</td> <td>Data segment name</td> <td>SABIT:</td> <td>BIT attribute symbol (saddr.bit)</td> </tr> <tr> <td>BSEG:</td> <td>Bit segment name</td> <td>SABIT2:</td> <td>BIT attribute symbol (saddr2.bit)</td> </tr> <tr> <td>MAC:</td> <td>Macro name</td> <td>SABITA:</td> <td>BIT attribute symbol (saddr.a.bit)</td> </tr> <tr> <td>MOD:</td> <td>Module name</td> <td>SFBIT:</td> <td>BIT attribute symbol (sfr.bit)</td> </tr> <tr> <td>SET:</td> <td>Symbol defined by SET directive</td> <td>RBIT:</td> <td>BIT attribute symbol (A.bit, X.bit, PSW.bit, PSWL.bit, PSWH.bit)</td> </tr> <tr> <td>NUM:</td> <td>NUMBER attribute symbol</td> <td>Blank:</td> <td>External reference symbol declared by EXTRN or EXTBIT</td> </tr> <tr> <td>ADDER:</td> <td>ADDRESS attribute symbol</td> <td>****:</td> <td>Undefined symbol</td> </tr> <tr> <td>SADDR2:</td> <td>SADDR2 attribute symbol</td> <td></td> <td></td> </tr> <tr> <td>SADDRA:</td> <td>SADDRA attribute symbol</td> <td></td> <td></td> </tr> </table>	CSEG:	Code segment name	BIT:	BIT attribute symbol (addr.bit)	DSEG:	Data segment name	SABIT:	BIT attribute symbol (saddr.bit)	BSEG:	Bit segment name	SABIT2:	BIT attribute symbol (saddr2.bit)	MAC:	Macro name	SABITA:	BIT attribute symbol (saddr.a.bit)	MOD:	Module name	SFBIT:	BIT attribute symbol (sfr.bit)	SET:	Symbol defined by SET directive	RBIT:	BIT attribute symbol (A.bit, X.bit, PSW.bit, PSWL.bit, PSWH.bit)	NUM:	NUMBER attribute symbol	Blank:	External reference symbol declared by EXTRN or EXTBIT	ADDER:	ADDRESS attribute symbol	****:	Undefined symbol	SADDR2:	SADDR2 attribute symbol			SADDRA:	SADDRA attribute symbol		
CSEG:	Code segment name	BIT:	BIT attribute symbol (addr.bit)																																						
DSEG:	Data segment name	SABIT:	BIT attribute symbol (saddr.bit)																																						
BSEG:	Bit segment name	SABIT2:	BIT attribute symbol (saddr2.bit)																																						
MAC:	Macro name	SABITA:	BIT attribute symbol (saddr.a.bit)																																						
MOD:	Module name	SFBIT:	BIT attribute symbol (sfr.bit)																																						
SET:	Symbol defined by SET directive	RBIT:	BIT attribute symbol (A.bit, X.bit, PSW.bit, PSWL.bit, PSWH.bit)																																						
NUM:	NUMBER attribute symbol	Blank:	External reference symbol declared by EXTRN or EXTBIT																																						
ADDER:	ADDRESS attribute symbol	****:	Undefined symbol																																						
SADDR2:	SADDR2 attribute symbol																																								
SADDRA:	SADDRA attribute symbol																																								
(3)	<p>Symbol reference format</p> <table border="0"> <tr> <td>EXT:</td> <td>External reference symbol declared by EXTRN (other than SADDR2, SADDRA attributes)</td> </tr> <tr> <td>EXT2:</td> <td>External reference symbol declared by EXTRN (SADDR2 attribute)</td> </tr> <tr> <td>EXTA:</td> <td>External reference symbol declared by EXTRN (SADDRA attribute)</td> </tr> <tr> <td>EXTB:</td> <td>External reference symbol declared by EXTBIT (saddr.bit)</td> </tr> <tr> <td>EXTBA:</td> <td>External reference symbol declared by EXTBIT(saddr.a.bit)</td> </tr> <tr> <td>EXTB2:</td> <td>External reference symbol declared by EXTBIT(saddr2.bit)</td> </tr> <tr> <td>PUB:</td> <td>External definition symbol declared by PUBLIC</td> </tr> <tr> <td>Blank:</td> <td>Local symbol, segment name, macro name, module name</td> </tr> <tr> <td>****:</td> <td>Undefined symbol</td> </tr> </table>	EXT:	External reference symbol declared by EXTRN (other than SADDR2, SADDRA attributes)	EXT2:	External reference symbol declared by EXTRN (SADDR2 attribute)	EXTA:	External reference symbol declared by EXTRN (SADDRA attribute)	EXTB:	External reference symbol declared by EXTBIT (saddr.bit)	EXTBA:	External reference symbol declared by EXTBIT(saddr.a.bit)	EXTB2:	External reference symbol declared by EXTBIT(saddr2.bit)	PUB:	External definition symbol declared by PUBLIC	Blank:	Local symbol, segment name, macro name, module name	****:	Undefined symbol																						
EXT:	External reference symbol declared by EXTRN (other than SADDR2, SADDRA attributes)																																								
EXT2:	External reference symbol declared by EXTRN (SADDR2 attribute)																																								
EXTA:	External reference symbol declared by EXTRN (SADDRA attribute)																																								
EXTB:	External reference symbol declared by EXTBIT (saddr.bit)																																								
EXTBA:	External reference symbol declared by EXTBIT(saddr.a.bit)																																								
EXTB2:	External reference symbol declared by EXTBIT(saddr2.bit)																																								
PUB:	External definition symbol declared by PUBLIC																																								
Blank:	Local symbol, segment name, macro name, module name																																								
****:	Undefined symbol																																								
(4)	Defined symbol name																																								



[Explanation of output items] (2/2)

Item	Details
(5)	Symbol reference format EXT: External reference symbol declared by EXTRN (other than SADDR2, SADDRA attributes) EXT2: External reference symbol declared by EXTRN (SADDR2 attribute) EXTA: External reference symbol declared by EXTRN (SADDRA attribute) EXTB: External reference symbol declared by EXTBIT (saddr.bit) EXTBA: External reference symbol declared by EXTBIT (saddra.bit) EXTB2: External reference symbol declared by EXTBIT (saddr2.bit) PUB: External definition 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@

11.1.5 Error list

An error list stores the error messages output when the assembler is started up.

[Output format]

```

Pass1 Start
(1)ERROR.ASM((2)26) : (3)F202 (4)Illegal operand
(1)ERROR.ASM((2)32) : (3)F202 (4)Illegal operand
Pass2 Start
(1)ERROR.ASM((2)26) : (3)F202 (4)Illegal operand
(1)ERROR.ASM((2)29) : (3)F407 (4)Undefined symbol reference 'DTSA'
(1)ERROR.ASM((2)29) : (3)F303 (4)Illegal expression
(1)ERROR.ASM((2)32) : (3)F202 (4)Illegal operand
(1)ERROR.ASM((2)37) : (3)F407 (4)Undefined symbol reference 'F'
(1)ERROR.ASM((2)37) : (3)F303 (4)Illegal expression
    
```

[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

## 11.2 Lists Output by 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

### 11.2.1 Link list file headers

The header is always output at the beginning of a link list file.

#### [Output format]

```
78K/IV Series Linker (1)Vx.xx                               Date: (2)xx xxx xxxx Page: (3) 1
```

```
Command: (4) k4main.rel k4sub.rel -ok4.map -dk4.dr
```

```
Para-file: (5)
```

```
Out-file: (6)K4.MAP
```

```
Map-file: (7)K4MAIN.MAP
```

```
Direc-File: (8)K4.DR
```

```
Directive: (9)memory ROM: (0000h, 4000h)
           memory RAM: (0ff000h, 1000h)
```

```
*** Link information ***
```

```
(10) 3 output segment(s)
```

```
(11) 37H 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

### 11.2.2 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) 00000000H    SIZE = (4) 00004000H
      OUTPUT   INPUT   INPUT   BASE      SIZE
      SEGMENT  SEGMENT  MODULE  ADDRESS
(6) CODE
      (7) CODE   (8) SAMPM (9) 00000000H (10) 00000002H (11) CSEG AT
(5) * gap *
      (6) ?CSEG (7) ?CSEG (8) SAMPM (9) 00000002H (10) 00000007EH
      (7) ?CSEG (8) SAMPS (9) 00000080H (10) 00000046H (11) CSEG
(5) * gap *
      (7) ?CSEG (8) SAMPS (9) 000000AAH (10) 0000001CH
      (9) 000000C6H (10) 00003F3AH

MEMORY = RAM
BASE ADDRESS = (3) 000FF000H    SIZE = (4) 00001000H
      OUTPUT   INPUT   INPUT   BASE      SIZE
      SEGMENT  SEGMENT  MODULE  ADDRESS
(5) * gap *
      (6) DATA (7) DATA (8) SAMPM (9) 000FF000H (10) 00000D20H
      (9) 000FFD20H (10) 00000003H (11) DSEG AT
      (9) 000FFD20H (10) 00000003H
(5) * gap *
      (9) 000FFD23H (10) 000001DDH
(5) * gap (Not Free Area) *
      (9) 000FFF00H (10) 00000100H

Target chip : (12) uPD784026
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.



### 11.2.4 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) 000FFD20H	(4) HDTSA
(1) SAMPM	(2) ADDR	(3) 000FFD21H	(4) STASC
(1) SAMPM	(2) CSEG		(4) CODE
(1) SAMPM	(2) CSEG		(4) ?CSEG
(1) SAMPM	(2) MOD		(4) SAMPM
(1) SAMPM	(2) CSEG		(4) ?CSEG
(1) SAMPM	(2) ADDR	(3) 000000BDH	(4) SASC
(1) SAMPM	(2) ADDR	(3) 000000C3H	(4) SASC1

**[Explanation of output items]**

Item	Details
(1)	Name of module in which local symbols are defined
(2)	Symbol attributes CSEG: Code segment name      BIT: BIT attribute symbol (addr.bit) DSEG: Data segment name      SABIT: BIT attribute symbol (saddr.bit) BSEG: Bit segment name      SABIT2: BIT attribute symbol (saddr2.bit) MAC: Macro name      SABITA: BIT attribute symbol (saddr.a.bit) MOD: Module name      SFBIT: BIT attribute symbol (sfr.bit) SET: Symbol defined by SET directive      RBIT: BIT attribute symbol (A.bit, X.bit, PSW.bit, PSWL.bit, PSWH.bit) NUM: NUMBER attribute symbol      Blank: External reference symbol declared by EXTRN or EXTBIT ADDR: ADDRESS attribute symbol SADDR2: SADDR2 attribute symbol SADDRA: SADDRA attribute symbol      *****: Undefined symbol
(3)	Symbol value
(4)	Local symbol name

### 11.2.5 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 'K4MAIN.REL'

**[Explanation of output items]**

Item	Details
(1)	Error no.
(2)	Error message

### 11.3 List Output by Object Converter

The object converter outputs the following list.

Output List File Name	Output List Name
Error list file	Error list

#### 11.3.1 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.

## 11.4 List Output by Librarian

The librarian outputs the following list.

Output List File Name	Output List Name
List file	Library data output list

### 11.4.1 Library data output list

The library data output list outputs data on the modules in a library file.

#### [Output format]

```

78K/IV Series librarian Vx.xx      DATE : (1) xx xxx xx                PAGE (2) 1
LIB-FILE NAME : (3) K4.LIB         ((4) xx xxx xx)
(5) 0001 (6) K4MAIN.REL           ((7) xx xxx xx)
      (8) MAIN                      (8) START
NUMBER OF PUBLIC SYMBOLS : (9) 2
(5) 0002 (6) K4SUB.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

## 11.5 Lists Output by 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

### 11.5.1 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.

### 11.5.2 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.

## CHAPTER 12 EFFICIENT USE OF RA78K4

This chapter introduces some methods that will help you to use the RA78K4 efficiently.

## 12.1 Improving Operating Efficiency (EXIT Status Function)

When any of the programs of the RA78K4 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 of use]

Contents of the batch file (RA.BAT)

```

ra78K4 -c4038 k4main. -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
ra78K4 -c4038 k4sub.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
lk78K4 k4main.rel k4sub.rel -ok4.lmf -g
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
oc78K4 k4.lmf
echo off
IF ERRORLEVEL 1 GOTO ERR
GOTO EXIT
:ERR
echo Error occurred
:EXIT

```

- Perform processing using batch file (RA.BAT).

```
C>>ra.bat
```

## 12.2 Preparing Development Environment (Environmental Variables)

The RA78K4 supports the following environmental variables for preparing the software development environment.

PATH: Search path for execution format  
 INC78K4: Search path for include file (assembler only)  
 LIB78K4: Search path for library file (linker only)  
 TMP: Path for creating temporary files  
 LANG78K: Kanji (2-byte character) type specification

### [Example of use]

Contents of AUTOEXEC.BAT

```
;AUTOEXEC.BAT
Verify on
break on
PATH C:\BIN;C:\BAT;C:\RA78K4; ← (1)
SET INC78K4=C:\RA78K4\INCLUDE ← (2)
SET LIB78K4=C:\RA78K4\LIB ← (3)
SET TMP=C:\TMP ← (4)
SET LANG78K = SJIS ← (5)
```

- (1) Because this path is specified, execution format files are retrieved from directories in the order C:\BIN, C:\BAT, C:\RA78K4.
- (2) The assembler retrieves include files from the directory C:\RA78K4\INCLUDE.
- (3) The linker retrieves library files from C:\RA78K4\LIB.
- (4) Each program creates a temporary file in C:\TMP.
- (5) Kanji in the comment statement is interpreted as shift JIS code.

**Caution** If the assembler package has been installed using the Windows installer, the necessary environmental variables are automatically set.

## 12.3 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 a batch file, 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.

## 12.4 Making 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, refer to **CHAPTER 4 CONTROL INSTRUCTIONS** in the Language Manual.

### [Example of use]

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

```
C>ra78K4 -c4038 k4main.asm -lhRA78K4_MAINROUTINE
```

This references K4MAIN.PRN.

```
78K/IV Series Assembler Vx.xx RA78K4_MAINROUTINE Date:xx xxx xxxx Page: 1
```

Title

```
Command: -c4038 k4main.asm -lhRA78K4_MAINROUTINE
Para-file:
In-file: K4MAIN.ASM
Obj-file: K4MAIN.REL
Prn-file: K4MAIN.PRN
```

Assemble list

ALNO	STNO	ADRS	OBJECT	M I	SOURCE STATEMENT
1	1				
2	2				NAME SAMPM
3	3				;*****
4	4				;* *
5	5				;* HEX -> ASCII Conversion Program *
6	6				;* *
7	7				;* main-routine *
					:
					:
					:

## 12.5 Reducing Program Startup Time

### 12.5.1 Specifying control instruction in 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(4038) ] Control instructions
$ XREF

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

```

### 12.5.2 Using PM plus

The options of each program of the RA78K4 are automatically saved to the project file (.RPJ) in PM plus. The saved options will be used for subsequent builds (MAKE). It is not therefore necessary to specify the options each time.

### 12.5.3 Creating parameter files and subcommand files

When executing any of the RA78K4's programs (structured assembler, 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]

Create a parameter file and perform assembly.

Contents of parameter file K4MAIN.PRA

```
;parameter file  
k4main.asm -osample.rel -g  
-psample.prn
```

Enter the following on the command line.

```
C><u>ra78k4 -fk4main.pra
```

#### [Example of use 2]

Create a parameter file and perform assembly.

Contents of parameter file K4.SLB

```
;  
;library creation command  
;  
create k4.lib  
;  
add k4.lib k4main.rel &  
k4sub.rel  
;  
exit
```

Enter the following on the command line.

```
C><u>lb78k4 <k4.slb
```

## 12.6 Object Module Library Formation

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 RA78K4 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.

## CHAPTER 13 ERROR MESSAGES

This chapter explains the causes of error messages output by the RA78K4's programs (structured assembler, assembler, linker, object converter, librarian, list converter, ECC generator, and PM plus), and the action to be taken by the user.

### 13.1 Overview of Error Messages

Error messages output by the RA78K4 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 command line, processing ends when another command 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 (Wxxx)**

An output object is generated which may not be the result the user intended.

**Remark** In a program executed in conversational format, the execution ends normally unless an abort error occurs.

Assembler error messages are classified as follows.

Error messages of the RA78K4 are explained beginning on the next page.

- A0xx --- Command line analysis error
- A9xx --- File or system error
- A1xx --- Other abort error
- F2xx --- Statement specification error
- F3xx --- Expression error
- F4xx --- Symbol error
- F5xx --- Segment error
- F6xx --- Control instruction or macro error
- W7xx --- Any type of warning error

## 13.2 Structured Assembler Error Messages

Table 13-1. Structured Assembler Error Messages (1/5)

A001	Message	Missing input file
	Cause	An input file has not been specified.
	Action by user	Specify an input file.
A002	Message	Too many input files
	Cause	Two or more input files have been specified.
	Action by user	Specify only one input file.
A004	Message	Illegal file name 'file name'
	Cause	Either there are illegal characters in the file name, or the number of characters exceeds the limit.
	Action by user	Input a file name that has legal characters and is within the character number limit.
A005	Message	Illegal file specification 'file name'
	Cause	An illegal file has been specified.
	Action by user	Specify a legal file.
A006	Message	File not found 'file name'
	Cause	The specified file does not exist.
	Action by user	Specify an existent file.
A008	Message	File specification conflicted 'file name'
	Cause	An I/O file name has been specified in duplicate.
	Action by user	Specify different I/O file names.
A009	Message	Unable to make file 'file name'
	Cause	The specified file is write-protected.
	Action by user	Release the write protection on the specified file.
A010	Message	Directory not found 'file name'
	Cause	A non-existent drive and/or directory has been included in the output file name.
	Action by user	Specify an existent drive and/or directory.
A011	Message	Illegal path 'option'
	Cause	Other than a path name has been specified in the option that specifies the path for the parameter.
	Action by user	Specify a correct path name.
A012	Message	Missing parameter 'option'
	Cause	A necessary parameter has not been specified.
	Action by user	Specify the parameter.
A014	Message	Out of range 'option'
	Cause	The specified numerical value is outside the range.
	Action by user	Specify a correct numerical value.
A015	Message	Parameter is too long 'option'
	Cause	The number of characters in the parameter exceeds the limit.
	Action by user	Specify a parameter whose character number is within the limit.

Table 13-1. Structured Assembler Error Messages (2/5)

A016	Message	Illegal parameter 'option'
	Cause	The syntax of the parameter is incorrect.
	Action by user	Specify a correct parameter.
A017	Message	Too many parameter 'option'
	Cause	The total number of parameters exceeds the limit.
	Action by user	Specify parameters within the number limit.
A018	Message	Option is not recognized 'option'
	Cause	The option name is incorrect.
	Action by user	Specify a correct option name.
A019	Message	Parameter file nested
	Cause	The -F option has been specified inside a parameter file.
	Action by user	Do not specify the -F option inside a parameter file.
A020	Message	Parameter file read error 'file name'
	Cause	The parameter file cannot be read.
	Action by user	Specify a correct parameter file.
A021	Message	Memory allocation failed
	Cause	There is insufficient memory.
	Action by user	Secure the necessary memory.
A101	Message	Open/read/write/close error on 'file name'
	Cause	Due to a file I/O error, the file cannot be opened, read written to, or closed normally.
	Action by user	Specify a correct file name.
A102	Message	Can't find 'file name'
	Cause	Either the include file does not exist, or the include file name has been specified together with an input file name or output file name.
	Action by user	Specify a correct path, directory, and file name.
A103	Message	Illegal include file 'file name'
	Cause	An illegal name has been specified for an include file.
	Action by user	Specify a correct file.
A104	Message	Illegal (-sc) character
	Cause	A character that cannot be used as a symbol has been specified in the -SC option.
	Action by user	Specify a correct character.
A105	Message	Can't define the reserved symbol
	Cause	A reserved word has been specified in the -D option.
	Action by user	Do not specify a reserved word in the -D option.
A106	Message	Duplicate PROCESSOR control
	Cause	The PROCESSOR control instruction has been specified more than once in the source file. A product type different to that of the -C option has been specified.
	Action by user	Specify the PROCESSOR control instruction once only. Correct the product type name.

Table 13-1. Structured Assembler Error Messages (3/5)

A107	Message	No processor specified
	Cause	The device type has not been specified.
	Action by user	Specify the device type.
A108	Message	Illegal processor type specified
	Cause	The device type specification is incorrect in the PROCESSOR control instruction in the source file.
	Action by user	Specify a correct device type.
A109	Message	Illegal processor type specified -C
	Cause	The device type specification is incorrect in the -C option.
	Action by user	Specify a correct device type.
A110	Message	Can't use this control outside module header
	Cause	An instruction that should have been written in the source module header has been written in a normal source line.
	Action by user	Write the instruction in the source module header.
A111	Message	Syntax error in module header
	Cause	The syntax of the instruction written in the source module header is incorrect.
	Action by user	Write the instruction using the correct syntax.
A112	Message	Structured assembler preprocessor internal error
	Cause	An error has occurred inside the structured assembler preprocessor.
	Action by user	Contact NEC Electronics.

F201	Message	Illegal #ELSE/#ENDIF
	Cause	#ELSE and #ENDIF statements have been written in an incorrect place.
	Action by user	Write the #ELSE and #ENDIF statements in the correct place.
F202	Message	Illegal #ENDCALLT
	Cause	An #ENDCALLT statement has been written in an incorrect place.
	Action by user	Write the #ENDCALLT statement in the correct place.
F203	Message	Missing #ENDIF
	Cause	The #ENDIF statement is missing.
	Action by user	Write the #ENDIF statement in the correct place.
F204	Message	Missing #ENDCALLT
	Cause	The #ENDCALLT statement is missing.
	Action by user	Write the #ENDCALLT statement in the correct place.
F205	Message	Too many #DEFCALLT definition
	Cause	The registered number of callt instruction conversion patterns exceeds the limit.
	Action by user	Reduce the number of registered #defcallt instructions.
F206	Message	Too many CALL instruction
	Cause	There are too many instructions to be defined by #DEFCALLT to #ENDCALLT.
	Action by user	Specify only one instruction to be defined by #DEFCALLT to #ENDCALLT.

**Table 13-1. Structured Assembler Error Messages (4/5)**

F207	Message	Duplicate definition
	Cause	The same conversion pattern has been defined a second time.
	Action by user	Correct the #DEFCALLT registration.
F208	Message	Symbol table overflow
	Cause	The number of symbols exceeds the limit.
	Action by user	Reduce the number of symbols.
F209	Message	Syntax error
	Cause	The syntax of the written statement is incorrect.
	Action by user	Use correct syntax.
F210	Message	Nest level error
	Cause	There is an error in the nesting (overflow, nesting level, etc.)
	Action by user	Use a correct control statement.
F211	Message	Too many characters in a line
	Cause	The length of one line has been exceeded.
	Action by user	Specify 218 or fewer characters on one line.
F212	Message	Too many include files
	Cause	There is an include quasi-directive in the include file.
	Action by user	Do not specify an include quasi-directive in the include file.
F214	Message	Illegal BREAK
	Cause	A BREAK statement has been written in an incorrect place.
	Action by user	Write the BREAK statement in the correct place.
F215	Message	Illegal CONTINUE
	Cause	A CONTINUE statement has been written in an incorrect place.
	Action by user	Write the CONTINUE statement in the correct place.
F216	Message	Illegal CASE/DEFAULT/ENDS
	Cause	A CASE/DEFAULT/ENDS statement has been written in an incorrect place.
	Action by user	Write the CASE/DEFAULT/ENDS statement in the correct place.
F217	Message	Illegal ELSEIF/ELSE/ENDIF
	Cause	An ELSEIF/ELSE/ENDIF statement has been written in an incorrect place.
	Action by user	Write the ELSEIF/ELSE/ENDIF statement in the correct place.
F218	Message	Illegal NEXT
	Cause	A NEXT statement has been written in an incorrect place.
	Action by user	Write the NEXT statement in the correct place.
F219	Message	Illegal ENDW
	Cause	An ENDW statement has been written in an incorrect place.
	Action by user	Write the ENDW statement in the correct place.
F220	Message	Illegal UNTIL/UNTIL_BIT
	Cause	UNTIL and UNTIL_BIT statements have been written in an incorrect place.
	Action by user	Write the UNTIL and UNTIL_BIT statements in the correct place.

Table 13-1. Structured Assembler Error Messages (5/5)

F221	Message	Missing ENDIF
	Cause	The ENDIF statement is missing.
	Action by user	Write the ENDIF statement in the correct place.
F222	Message	Missing ENDS
	Cause	The ENDS statement is missing.
	Action by user	Write the ENDS statement in the correct place.
F223	Message	Missing ENDW
	Cause	The ENDW statement is missing.
	Action by user	Write the ENDW statement in the correct place.
F224	Message	Missing NEXT
	Cause	The NEXT statement is missing.
	Action by user	Write the NEXT statement in the correct place.
F225	Message	Missing UNTIL/UNTIL_BIT
	Cause	The UNTIL and UNTIL_BIT statements are missing.
	Action by user	Write the UNTIL and UNTIL_BIT statements in the correct place.
F226	Message	Illegal character in a line
	Cause	An incorrect character has been written in the source line.
	Action by user	Delete the incorrect character written in the source line.
F227	Message	Illegal operand in a line
	Cause	The data size of the substitution and comparative condition formats is incorrect.
	Action by user	Specify the correct data size.
F228	Message	Illegal SFR access in operand
	Cause	An sfr symbol that is unable to access the substitution format has been written.
	Action by user	Check the access status of the sfr symbols and write a correct sfr symbol.
F229	Message	This symbol is reserved "symbol name"
	Cause	The symbol used is a reserved word.
	Action by user	Change the symbol name.

W301	Message	Symbol redefinition
	Cause	The symbol has been defined more than once by the #define statement.
	Program action	The most recently defined symbol is valid.
	Action by user	To validate the symbol first defined, correct the syntax.
W302	Message	Duplicate PROCESSOR option and control
	Cause	The device type specified in the -C option is different to that specified in the \$PROCESSOR control instruction.
	Program action	The device type specified in the -C option is valid, and the device type specified in the \$PROCESSOR control instruction is ignored.
	Action by user	Check that the device type specified in the -C option is correct.

## 13.3 Assembler Error Messages

Table 13-2. Assembler Error Messages (1/9)

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 format for a control instruction that can be written in a source module header.
A104	Message	Can't use this control outside module header
	Cause	A control instruction for specification in a source module header is written in an ordinary source.
A105	Message	Duplicate PROCESSOR control
	Cause	A PROCESSOR control instruction is written 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 definable symbols has been exceeded.
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 128bytes
	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.
A113	Message	Chgsfr option out of range
	Cause	An incorrect value was specified for option -CS.
A114	Message	Local symbol name of asm statement must begin with '?L' in C source.
	Cause	A local symbol which begins with other than '?L' is described in the #asm of the C source.

F201	Message	Syntax error
	Cause	An incorrect statement format was used.
F202	Message	Illegal operand
	Cause	The specified operand is illegal.
F203	Message	Illegal register
	Cause	A register that cannot be specified was specified.

Table 13-2 Assembler Error Messages (2/9)

F204	Message	Illegal character
	Cause	An illegal character is specified 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 written in a character string.
F208	Message	Absolute statement
	Cause	When option -CSA/CHGSFR directive is specified, an absolute description is made for a describable internal RAM area and SFR area.

F301	Message	Too complex expression
	Cause	Expression is too complex.
F302	Message	Absolute expression expected
	Cause	A relocatable expression is specified.
F303	Message	Illegal expression
	Cause	Incorrect format for expression is used.
F304	Message	Illegal symbol in expression 'file name'
	Cause	An unusable symbol is used in an expression.
F305	Message	Too long string as constant
	Cause	Limit on string constant length (4 characters) is exceeded.
F306	Message	Illegal number
	Cause	Incorrect numerical value is specified.
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.
F309	Message	Illegal bit value
	Cause	Incorrect bit value is specified.
F310	Message	Bit value out of range
	Cause	Bit value exceeds the range 0 to 7.
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 specifiable range (40H to 7EH) for addr5.
F314	Message	Operand out of range (addr11)
	Cause	Operand is outside the specifiable range (800H to FFFH) for addr11.
F315	Message	Operand out of range (saddr)
	Cause	Operand is outside the specifiable range (0FE20H to 0FF1FH) for saddr.

Table 13-2. Assembler Error Messages (3/9)

F316	Message	Operand out of range (addr16)
	Cause	Operand is outside the specifiable range (varies according to target device) for addr16.
F317	Message	Even expression expected
	Cause	Odd-number address is specified for word access.
F318	Message	Operand out of range (sfr)
	Cause	Operands for the SFR/SFRP directives are specified exceeding the limit, or an odd value is specified for the operand of the SFR directive.
F319	Message	Operand out of range (addr20)
	Cause	Operand is outside the describable range (0H to FFFFFH except main RAM area) for addr20.
F320	Message	Operand out of range (addr24)
	Cause	Operand is outside the describable range (0H to FFFFFH) for addr24.
F321	Message	Operand out of range (saddr1)
	Cause	Operand is outside the describable range (FE00H to FEFFH and FFE00H to FFEFFH) for saddr1.
F322	Message	Operand out of range (saddr2)
	Cause	Operand is outside the describable range (FD20H to FDFFH and FFD20H to FFDFH) for saddr2.
F323	Message	Operand out of range (word)
	Cause	Operand is outside the describable range (0H to FFFFH) for word.
F324	Message	Operand out of range (locaddr)
	Cause	A value other than 0 or 15 was described for locaddr.
F325	Message	LOCATION parameter mismatch
	Cause	Operands in option -CS/CHGSFR directive and LOCATION instruction do not match.
F326	Message	Illegal SFR access in operand
	Cause	An SFR symbol which cannot be accessed is described.
F328	Message	Operand out of range (tbyte)
	Cause	Operand is outside the describable range (0H to FFH) for tbyte.

F401	Message	Illegal symbol for PUBLIC 'symbol name'
	Cause	This symbol cannot be declared PUBLIC.
F402	Message	Illegal symbol for EXTRN/EXBIT '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	A symbol defined with bit items other than saddr1.bit or saddr2.bit cannot have a PUBLIC declaration. Cancel PUBLIC declaration or change 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	Specify backward reference or EXTBIT declaration for the bit symbol.

**Table 13-2. Assembler Error Messages (4/9)**

F406	Message	Can't refer to forward bit symbol 'symbol name'
	Cause	Specification 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 written 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).
F411	Message	This symbol is reserved 'symbol name'
	Cause	The defined symbol is a reserved word.

F502	Message	Illegal segment name
	Cause	Symbol is written 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).
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 outside CSEG
	Cause	Machine language instruction or BR directive is defined in something other than a code segment.
F508	Message	Can't describe DBIT outside BSEG
	Cause	DBIT directive is defined 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 corresponding 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 segment is described in an odd number.
F513	Message	Too many line number data
	Cause	The number of items of line information for debugging exceeds the limit.

F601	Message	Nesting over of 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 exceeds limit (5 per module).

Table 13-2. Assembler Error Messages (5/9)

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 written 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 name parameter described 'symbol name'
	Cause	Symbol is specified with same name as a formal parameter for a macro definition.
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	Specification 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 is incorrect.
F619	Message	Illegal defined macro
	Cause	Referenced macro is incorrectly defined.
F620	Message	Illegal syntax of actual parameter
	Cause	Specification 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 specified 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.

Table 13-2. Assembler Error Messages (6/9)

F626	Message	Can't describe LOCAL outside macro definition
	Cause	LOCAL directive is specified 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.
W701	Message	Too long source line
	Cause	Over 2048 characters are described on 1 line of a source statement.
	Program processing	2049th 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 written at end of source file.
	Program processing	Assumes that END statement is described at end of source file.
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.

Table 13-2. Assembler Error Messages (7/9)

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.
W713	Message	'SP' operand in 16bit instruction
	Cause	SP is described in operand in 16-bit processing instruction.
	Program processing	Outputs a 24-bit processing instruction object code.
W714	Message	Can't analyze operand (saddr1 or saddr2)
	Cause	Assembler cannot judge whether symbol described in an operand is SADDR1 or SADDR2 attribute.
	Program processing	If the operand has SADDR1 attribute, the 1-byte object code '00' (NOP) is added and the operand is output.
	Action by user	Define the operand symbol before reference.
W715	Message	Cause over flow in expression result
	Cause	Result of calculation exceeds FFFFFFFH (24 bits).
	Program processing	Last 24-bit value is extracted as the result of the calculation of the expression.
W716	Message	Multiple define LOCATION directive
	Cause	2 or more LOCATION instructions are defined.
	Program processing	LOCATION instructions are ignored and processing continues.
W717	Message	Normal, callt and callf functions must be described together respectively.
	Cause	Debugging information may be illegal because normal, callt, and callf functions are not described together.
	Program processing	Describe normal and callt functions together.
F801	Message	Illegal debug information
	Cause	<1> The number of items of information for debugging exceeds the limit. <2> An internal error occurred in the assembler itself.
	Program processing	That line is ignored.

Table 13-2. Assembler Error Messages (8/9)

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.
A913	Message	Can't read source file 'file name'
	Cause	A file input/output error has occurred in the source file.
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.

**Table 13-2. Assembler Error Messages (9/9)**

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.
A924	Message	Too large cross references list
	Cause	Internal memory was insufficient during cross reference list processing.
	Action by user	Divide the source file.
A925	Message	Too many relocation date
	Cause	Internal memory was insufficient during relocatable symbol processing.
	Action by user	Divide the source file.

## 13.4 Linker Error Messages

Table 13-3. Linker Error Messages (1/6)

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	Specification 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.
A106	Message	Can't create temporary file 'file name'
	Cause	Cannot create temporary file.
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.
F113	Message	Linker internal error
	Cause	Internal error in the linker
	Action by user	Contact an authorized representative or NEC.
F114	Message	Illegal number
	Cause	Specification of a numerical value in a directive is incorrect.
F115	Message	Too large value (up to 16M/0FFFFFFH)
	Cause	A value greater than 16MB (0FFFFFFH) is described in the directive.

Table 13-3. Linker Error Messages (2/6)

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 (up to 65535/0FFFFH) in the same name segment 'file name'
	Cause	The amount of line number data (debugging data) exceeds the limit. Input line number data (debugging data) again and continue processing. An object file will only be output if option -J is specified.
A118	Message	Incorrect version input file 'file name'
	Cause	The version of the input file is not compatible with the linker
A119	Message	Different SFR area (input file 'file name')
	Cause	The SFR change directive and specified SFR area are different in the input file.
A120	Message	Unsuitable for processor type from first input file 'first input file name'
	Cause	The first input object module file contains no device data.
W121	Message	Different processor type from next input file 'next input file name'
	Cause	The device data in the first input object module file is different from the device data in the next input object module file.
	Program processing	The device data in the first input object module file is effective, and processing continues.

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.

Table 13-3. Linker Error Messages (3/6)

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 xxxxxxH' 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' is not exist - ignored
	Cause	Segment specified in the directive does not exist.

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).
F306	Message	A multiple of 4 value expected (segment 'segment name', address xxxxH, type 'addressing type')
	Cause	The value of the operand used for resolving the relocatable object code for saddr addressing is not a multiple of 4.

**Caution** The address shown in 'address xxxxH' in the messages in F301 to F306 are absolute addresses after segment allocation.

Table 13-3. Linker Error Messages (4/6)

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.
F403	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.
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).
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).
F408	Message	Can't find -A symbol
	Cause	The symbol written after -A for program entry address specification of a linker option is not a public symbol.
F409	Message	-A symbol 'symbol name' is unmatched type
	Cause	The type of the symbol searched by -A in program entry address specification of a linker option is incorrect.
	Action by user	Use the permitted type of the symbol that is to be searched by program entry address specification.
F410	Message	Multiple module name definition 'module name' in file 'file 1' First defined in file 'file 2'
	Cause	Name of module in object module file 1 is the same as name of module in object module file 2.
W411	Message	Different REL type in 'file name'
	Cause	The type version of object module file is different.
	Action by user	Re-assemble or re-compile with the newest version.
F412	Message	Multiple CHGSFR in file 'file 1' First defined in file 'file 2'
	Cause	Value specified in CHGSFR directive or option -CS is different in files 1 and 2. Value specified in CHGSFR directive or option -CS must be the same in the entire program.
F413	Message	Multiple LOCATION in file 'file name 1' First defined in file 'file name 2'
	Cause	This is output if 2 or more LOCATION instructions for all input object module files are found.
F414	Message	'LOCATION' operation not found in all modules
	Cause	This is output if no LOCATION instructions for all input object module files are found.
F415	Message	-QD/QF/etc. and Not -QD/QF/etc. REL are mixed
	Cause	An input object module file has a different specification for a compiler optimization option which must be the same for the entire program. Compile using the same value as in the rest of the program.

Table 13-3. Linker Error Messages (5/6)

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 object module files.
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 (CC78K4, ST78K4, RA78K4) used until the link stage for all input object module files and the device file version.
W418	Message	File 'file name' is old. Can't find TOOL information
	Cause	This is output when TOOL information is not found in input object module file. Normally, this is always output when link is performed with an old (DF-incompatible) object module file.
F424	Message	-ZF REL and no -ZF REL are mixed in file 'file name'
	Cause	When linking load module of the boot area ROM program of a flash ROM model with object module of the flash area program, some object module do not specify the -ZF option during compilation.
F425	Message	There are different function ID in same name 'function name' (file 'file name')
	Cause	A function of the same name declared as EXT_FUNC by the compiler has a different ID value.
F426	Message	Multiple input BOOT file 'file name'
	Cause	Two or more load module of the boot area ROM program are input when load module of the boot area ROM program of a flash ROM model is linked with object module of the flash area program.
F427	Message	BOOT REL and -ZF REL are mixed in file 'file name'
	Cause	Object module specified by the -ZF option is input during compilation for linking with the -ZF option specified.
F428	Message	FLASH start address larger than ROM max address
	Cause	The first address of the flash ROM area is greater than the ROM end address of the target device.
F429	Message	BOOT segment 'segment name' are found in FLASH file 'file name'
	Cause	When load module of the boot area ROM program of a flash ROM model is linked with object module of the flash area ROM program, a segment with a location address less than the first address of the flash ROM area exists in the object module.
F430	Message	Different FLASH address in file 'file name'
	Cause	All the first addresses of the flash ROM area of the input files are not the same.
F431	Message	There are different function name in same ID (function name) (file 'file name')
	Cause	Two or more functions declared as EXT_FUNC by the compiler have the same ID value.
F432	Message	Illegal allocation of an EXT_FUNC function 'function name' (file 'file name')
	Cause	The entity of the function declared as EXT_FUNC by the compiler exists when linking is performed with the -ZB option specified.
A501	Message	Insufficient memory in hostmachine
	Cause	The system does not have sufficient memory to operate the program.

Table 13-3. Linker Error Messages (6/6)

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.

## 13.5 Object Converter Error Messages

Table 13-4. 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.
	Action by user	If the source is written in C, make sure that the assembler code in the C source satisfies the following conditions: <Conditions> <ul style="list-style-type: none"> <li>• If a local symbol is used, use a symbol that starts with string ?L (such as ?L@01 or ?L@sym). However, keep this symbol to within 8 characters. Do not define this symbol externally (PUBLIC declaration).</li> </ul>

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.

W300	Message	xxxxxxH - yyyyyyH overlapped
	Cause	Objects overlapped in the address from xxxxxxH to yyyyyyH are output.
W301	Message	Can't initialize RAM area 'address' - 'address'
	Cause	Initial value data is output to the RAM area.
	Action by user	If DB/DW is written in DSEG of the assembly source, either change the object to DS or write the DB/DW instruction in CSEG.

**Table 13-4. Object Converter Error Messages (2/2)**

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.

## 13.6 Librarian Error Messages

Table 13-5. Librarian Error Messages (1/3)

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.
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.

Table 13-5. Librarian Error Messages (2/3)

A021	Message	Memory allocation failed
	Cause	An error occurred in memory allocation.
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 (symbol: 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.
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.

Table 13-5. Librarian Error Messages (3/3)

F117	Message	Can not use device file
	Cause	A device-type file is specified in the input file. CLOCK is specified in the list command of an input or output file. PRN, CON, or CLOCK is specified in an output object module file or output library file.
F118	Message	Illegal path (file:file name)
	Cause	Path name in the specified file is incorrect.
A901	Message	File open error (file: file name)
	Cause	File cannot be opened.
F902	Message	File read error (file: file name)
	Cause	File cannot be correctly read.
A903	Message	File write error (file: file name)
	Cause	Data cannot be correctly written to file.
A904	Message	File seek error (file: file name)
	Cause	File seek error has occurred.
A905	Message	File close error (file: file name)
	Cause	File cannot be closed.

## 13.7 List Converter Error Messages

Table 13-6. List Converter Error Messages (1/2)

A101	Message	File is not 78K4 'file name'
	Cause	Input file name is not a 78K4 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 module file 'segment name'
	Cause	Segment name of object module file is not found in load module file.
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.
A109	Message	Overlay file can not open 'path name'
	Cause	Assembler overlay file cannot be opened.
A110	Message	Illegal assembler list file 'file name'
	Cause	Input assemble list is a file type other than an assemble list file.

**Table 13-6. List Converter Error Messages (2/2)**

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

## 13.8 ECC Generator Error Messages

Table 13-7. ECC Generator Error Messages (1/2)

A001	Message	Missing input file
	Cause	No input file has been specified for options other than -F and --, or no help file exists when the ECC generator was activated by specifying only its executable program name.
A002	Message	Too many input files
	Cause	The total number of input files exceeds the limit.
A003	Message	Unrecognized string 'whatever specified'
	Cause	Something other than an option is specified on a conversational-format command line.
A004	Message	Illegal file name 'file name'
	Cause	The file name displayed contains an error in syntax, characters or number of characters that can be described as a file name.
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	The specified input file does not exist.
A007	Message	Input file specification overlapped 'file name'
	Cause	The input file name specification is overlapped.
A008	Message	File specification conflicted 'file name'
	Cause	The input or output file name specification is overlapped.
A009	Message	Unable to make file 'file name'
	Cause	The specified output file cannot be created because it already exists as a read-only file.
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 'option'
	Cause	An item other than a path name is specified in an option specifying the path name for an option parameter.
A012	Message	Missing parameter 'option'
	Cause	The required option parameter is not specified.
A013	Message	Parameter not needed 'option'
	Cause	An unnecessary option parameter is specified.
A014	Message	Out of range 'option'
	Cause	A value outside the range has been specified for the option parameter.
A015	Message	Parameter is too long 'option'
	Cause	The number of characters specified in the option parameter exceeds the limit.
A016	Message	Illegal parameter 'option'
	Cause	A mistake exists in the syntax of the option parameter.
A017	Message	Too many parameter 'option'
	Cause	The total number of parameters specified for the option exceeds the limit.
A018	Message	Option is not recognized 'option'
	Cause	An incorrect option is specified.
A019	Message	Parameter file nested
	Cause	The -F option is specified in the parameter file.

Table 13-7. ECC Generator Error Messages (2/2)

A020	Message	Parameter file read error 'file name'
	Cause	An error occurred in reading the parameter file.
A021	Message	Memory allocation failed
	Cause	An error occurred in memory allocation.
A051	Message	Missing end address of input HEX data
	Cause	The specification of parameter 1 of the -A option (address specification 1) is incorrect.
A052	Message	Missing start address of ECC generation
	Cause	The specification of parameter 2 of the -A option (address specification 2) is incorrect.
A053	Message	Missing start address of SR generation
	Cause	The specification of parameter 3 of the -A option (address specification 3) is incorrect.
A054	Message	Missing start address of SR ECC generation
	Cause	The specification of parameter 4 of the -A option (address specification 4) is incorrect.

A101	Message	Illegal file 'file name'
	Cause	The input file (file shown in displayed file name) is not the HEX-format object module file.
A102	Message	Check sum error at address - 'address'
	Cause	A check sum error exists in the input file.
A103	Message	Address out of range
	Cause	The address of an input HEX data exceeds the end address of the input HEX data.
A104	Message	Code exist at same address in 'address'
	Cause	Codes for input HEX data exist at the same address.

F301	Message	No generated ecc code for code 'address' - 'address'
	Cause	An object code that does not generate the ECC code exists.

W701	Message	Empty area exist in 'address' - 'address'
	Cause	An empty area exists in the input code space.
W702	Message	Replace Intel expand format
	Cause	Although the input file is an Intel HEX file, the generated code in the address space exceeds 64 K due to the generation of ECC code, ECW data, and ECC code for ECW data. Because of this, the file is being output based on the Intel expansion HEX format.

A911	Message	Read error 'file name'
	Cause	An error occurred during disk read.
A912	Message	Write error
	Cause	There is no empty area in the data area or an error occurred during disk write.
A913	Message	Can't open 'file name'
	Cause	The file cannot be opened or there is no empty area in the directory.

### 13.9 PM plus Error Messages

This section explains error messages that are not contained in the online help of PM plus. For information on other PM plus error messages, please refer to the online help of PM plus.

#### (1) Error messages displayed by the structured assembler (ST78K4) DLL

!	Message	Cannot find ST78K4.EXE shown in environment variable PATH.
	Cause	ST78K4.EXE execution format is not registered in the directory shown in the environment variable PATH
	Solution	Enter ST78K4.EXE and other ST78K4 related files in the directory shown in the environment variable PATH.
	Button	Click OK to close the message box.
×	Message	Not enough memory.
	Cause	Memory is not enough.
	Solution	Try again after shutting down other applications.
	Button	Click OK to close the message box.
×	Message	Cannot lock the memory.
	Cause	Insufficient memory or Windows System operation error
	Solution	Shut down other applications or try again after re-starting Windows.
	Button	Click OK to close the message box.

#### (2) Error messages displayed by the assembler (RA78K4) DLL

!	Message	Cannot find RA78K4.EXE shown in environment variable PATH.
	Cause	RA78K4.EXE execution format is not registered in the directory shown in the environment variable PATH
	Solution	Enter RA78K4.EXE and other RA78K4 related files in the directory shown in the environment variable PATH.
	Button	Click OK to close the message box.
×	Message	Not enough memory.
	Cause	Memory is not enough.
	Solution	Try again after shutting down other applications.
	Button	Click OK to close the message box.
×	Message	Cannot lock the memory.
	Cause	Insufficient memory or Windows System operation error
	Solution	Shut down other applications or try again after re-starting Windows.
	Button	Click OK to close the message box.

**(3) Error messages displayed by the linker (LK78K4) DLL**

!	Message	Cannot find LK78K4.EXE shown in environment variable PATH.
	Cause	LK78K4.EXE execution format is not registered in the directory shown in the environment variable PATH.
	Solution	Enter LK78K4.EXE and other LK78K4 related files in the directory shown in the environment variable PATH.
	Button	Click OK to close the message box.
×	Message	Not enough memory.
	Cause	Memory is not enough.
	Solution	Try again after shutting down other applications.
	Button	Click OK to close the message box.
×	Message	Cannot lock the memory.
	Cause	Insufficient memory or Windows System operation error
	Solution	Shut down other applications or try again after re-starting Windows.
	Button	Click OK to close the message box.

**(4) Error messages displayed by the object converter (OC78K4) DLL**

!	Message	Cannot find OC78K4.EXE shown in environment variable PATH.
	Cause	OC78K4.EXE execution format is not registered in the directory shown in the environment variable PATH.
	Solution	Enter OC78K4.EXE and other OC78K4 related files in the directory shown in the environment variable PATH.
	Button	Click OK to close the message box.
×	Message	Not enough memory.
	Cause	Memory is not enough.
	Solution	Try again after shutting down other applications.
	Button	Click OK to close the message box.
×	Message	Cannot lock the memory.
	Cause	Insufficient memory or Windows System operation error
	Solution	Shut down other applications or try again after re-starting Windows.
	Button	Click OK to close the message box.

## APPENDIX A SAMPLE PROGRAMS

The following is an introduction to the sample lists attached to the RA78K4.

### A.1 Source Lists

#### (1) K4MAIN.ASM

```

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

                PUBLIC   MAIN, START
                EXTRN    CONVAH

DATA          DSEG      AT 0FFD20H
HDTSA:        DS         1
STASC:        DS         2

CODE          CSEG      AT 0H
MAIN:         DW         START

                CSEG
                LOCATION 15
START:        MOV        RFM, #00
                MOVG     SP, #0FFE00H
                MOV      MM, #00
                MOV      STBC, #08H

                MOV      HDTSA, #1AH
                MOVG     WHL, #HDTSA           ;set hex 2-code data in WHL register

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

                BR       $$

                END
```

## (2) K4SUB.ASM

```

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    [WHL]      ;hex upper code load
        CALL    $!SASC
        MOV     B,A        ;store result

        MOV     A,#0
        ROL4    [WHL]      ;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

```
A><u>ra78k4 -c4026 k4main.asm -kx -lw90
```

```
78K/IV Series Assembler Vx.xx [ xx xxx xx]  
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Pass1 Start
```

```
Pass2 Start
```

```
Target chip : uPD784026  
Device file : Vx.xx
```

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

```
A><u>ra78k4 -c4026 k4sub.asm -kx -lw90
```

```
78K/IV Series Assembler Vx.xx [ xx xxx xx]  
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Pass1 Start
```

```
Pass2 Start
```

```
Target chip : uPD784026  
Device file : Vx.xx
```

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

```
A><u>lk78k4 k4main.rel k4sub.rel -g -o78k4.lnk -p78k4.map -kp -kl
```

```
78K/IV Series Linker Vx.xx [ xx xxx xx]  
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
Target chip : uPD784026  
Device file : Vx.xx
```

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

A><u>oc78k4 78k4.lmf

78K/IV Series Object Converter Vx.xx [xx xxx xx]

Copyright (C) NEC Electronics Corporation xxxx,xxxx

Target chip : uPD784026

Device file : Vx.xx

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

A><u>lb78k4

78K/IV Series Librarian Vx.xx [xx xxx xx]

Copyright (C) NEC Electronics Corporation xxxx,xxxx

\*create k4.lib k4sub.rel

\*list -p -o78k4.lst k4.lib

\*exit

A><u>lcnv78k4 k4main -l78k4.lmf

List Conversion Program for RA78K/IV Vx.xx [xx xxx xx]

Copyright (C) NEC Electronics Corporation xxxx

Pass1: start...

Pass2: start...

Conversion complete.

## A.3 Output Lists

### A.3.1 Assemble lists

#### (1) K4MAIN.ASM assemble list

78K/IV Series Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c4026 k4main.asm -kx -lw90

Para-file:

In-file: K4MAIN.ASM

Obj-file: K4MAIN.REL

Prn-file: K4MAIN.PRN

#### Assemble list

ALNO	STNO	ADRS	OBJECT	M I	SOURCE STATEMENT
1	1				
2	2				NAME SAMPM
3	3				;*****
4	4				;*
5	5				;* HEX -> ASCII Conversion Program *
6	6				;*
7	7				;* main-routine *
8	8				;*
9	9				;*****
10	10				
11	11				PUBLIC MAIN,START
12	12				EXTRN CONVAH
13	13				
14	14	-----			DATA DSEG AT 0FFD20H
15	15	0FFD20			HDTSA: DS 1
16	16	0FFD21			STASC: DS 2
17	17				
18	18	-----			CODE CSEG AT 0H
19	19	000000	R0400		MAIN: DW START
20	20				
21	21	-----			CSEG
22	22	000000	09C1FF00		LOCATION 15
23	23	000004	2BCC00		START: MOV RFM,#00
24	24	000007	092000FE0F		MOVG SP,#0FFE00H
25	25	00000C	2BC400		MOV MM,#00
26	26	00000F	09C0F708		MOV STBC,#08H
27	27				

```

28 28 000013 3A201A      MOV    HDTSA,#1AH
29 29 000016 38FB20FD0F    MOVG  WHL,#HDTSA ;set hex 2-code data in WHL register
30 30
31 31 00001B R09F00000    CALL  CONVAH      ;convert ASCII <- HEX
32 32                                ;output BC-register <- ASCII code
33 33 00001F 38DB21FD0F    MOVG  TDE,#STASC ;set DE <- store ASCII code table
34 34 000024 D3            MOV    A,B
35 35 000025 50            MOV    [TDE+],A
36 36 000026 D2            MOV    A,C
37 37 000027 50            MOV    [TDE+],A
38 38
39 39 000028 14FE      BR     $$
40 40
41 41                                END

```

Segment informations

ADRS	LEN	NAME
0FFD20	000003H	DATA
000000	000002H	CODE
000000	00002AH	?CSEG

(2) 78K4SUB.ASM assemble list

78K/IV Series Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: -c4026 k4sub.asm -kx -lw90

Para-file:

In-file: K4SUB.ASM

Obj-file: K4SUB.REL

Prn-file: K4SUB.PRN

Assemble list

ALNO	STNO	ADRS	OBJECT	M I	SOURCE STATEMENT
1	1				
2	2				NAME SAMPS
3	3				;*****
4	4				;*
5	5				;* HEX -> ASCII Conversion Program *
6	6				;*
7	7				;* sub-routine *
8	8				;*
9	9				;* input condition : (HL) <- hex 2 code *
10	10				;*
11	11				;* output condition : BC-register <-ASCII 2 code *
12	12				;*
13	13				;*****
14	14				
15	15				PUBLIC CONVAH
16	16				
17	17	-----			CSEG
18	18	000000	B900		CONVAH: MOV A,#0
19	19	000002	059F		ROL4 [WHL] ;hex upper code load
20	20	000004	3F0C00		CALL \$!SASC
21	21	000007	2431		MOV B,A ;store result
22	22				
23	23	000009	B900		MOV A,#0
24	24	00000B	059F		ROL4 [WHL] ;hex lower code load
25	25	00000D	3F0300		CALL \$!SASC
26	26	000010	2421		MOV C,A ;store result
27	27				
28	28	000012	56		RET
29	29				

```

30 30 ;*****
31 31 ;* subroutine convert ASCII code *
32 32 ;* input Acc (lower 4bits) <- hex code *
33 33 ;* output Acc <- ASCII code *
34 34 ;*****
35 35
36 36 000013 AF0A SASC: CMP A,#0AH ;check hex code > 9
37 37 000015 8302 BC $SASC1
38 38 000017 A807 ADD A,#07H ;bias(+7)
39 39 000019 A830 SASC1: ADD A,#30H ;bias(+30)
40 40 00001B 56 RET

```

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE STATEMENT
41	41					
42	42					END

Segment informations

ADRS	LEN	NAME
000000	00001CH	?CSEG

## A.3.2 Symbol lists

## (1) K4MAIN.ASM symbol list

## Symbol Table List

VALUE	ATTR	RTYP	NAME
	CSEG		?CSEG
	CSEG		CODE
-----H		EXT	CONVAH
	DSEG		DATA
FFD20H	ADDR		HDTSA
0H	ADDR	PUB	MAIN
	MOD		SAMPM
4H	ADDR	PUB	START
FFD21H	ADDR		STASC

## (2) K4SUB.ASM symbol list

## Symbol Table List

VALUE	ATTR	RTYP	NAME
	CSEG		?CSEG
0H	ADDR	PUB	CONVAH
	MOD		SAMPS
13H	ADDR		SASC
19H	ADDR		SASC1

**A.3.3 Cross-reference lists**

**(1) K4MAIN.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@ 32
DATA			DSEG		DATA	15#
HDTSA	FFD20H		ADDR		DATA	16# 29 30
MAIN	0H		ADDR	PUB	CODE	12@ 20#
SAMPM			MOD			3#
START	4H	R	ADDR	PUB	?CSEG	12@ 20 24#
STASC	FFD21H		ADDR		DATA	17# 34

**(2) K4SUB.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: k4main.rel k4sub.rel -g -o78k4.lmf -p78k4.map -kp -kl  
 Para-file:  
 Out-file: 78K4.LMF  
 Map-file: 78K4.MAP  
 Direc-file:  
 Directive:

\*\*\* Link information \*\*\*

3 output segment(s)  
 48H byte(s) real data  
 23 symbol(s) defined

\*\*\* Memory map \*\*\*

SPACE = REGULAR

MEMORY = ROM

BASE ADDRESS = 00000000H SIZE = 00010000H

OUTPUT SEGMENT CODE	INPUT SEGMENT CODE	INPUT MODULE SAMPM	BASE ADDRESS	SIZE	
			00000000H	00000002H	CSEG AT
			00000000H	00000002H	
* gap *			00000002H	0000007EH	
?CSEG			00000080H	00000046H	CSEG
	?CSEG	SAMPM	00000080H	0000002AH	
	?CSEG	SAMPS	000000AAH	0000001CH	
* gap *			000000C6H	0000FF3AH	

MEMORY = RAM

BASE ADDRESS = 000FF700H SIZE = 00000900H

OUTPUT SEGMENT DATA	INPUT SEGMENT DATA	INPUT MODULE SAMPM	BASE ADDRESS	SIZE	
* gap *			000FF700H	00000620H	
			000FFD20H	00000003H	DSEG AT
			000FFD20H	00000003H	
* gap *			000FFD23H	000001DDH	
* gap (Not Free Area) *			000FFF00H	00000100H	

Target chip : uPD784026  
 Device file : Vx.xx

**A.3.5 Public symbol list**

\*\*\* Public symbol list \*\*\*

MODULE	ATTR	VALUE	NAME
SAMPM	ADDR	00000000H	MAIN
SAMPM	ADDR	00000084H	START
SAMPS	ADDR	000000AAH	CONVAH

**A.3.6 Local symbol list**

\*\*\* Local symbol list \*\*\*

MODULE	ATTR	VALUE	NAME
SAMPM	MOD		SAMPM
SAMPM	DSEG		DATA
SAMPM	ADDR	000FFD20H	HDTSA
SAMPM	ADDR	000FFD21H	STASC
SAMPM	CSEG		CODE
SAMPM	CSEG		?CSEG
SAMPS	MOD		SAMPS
SAMPS	CSEG		?CSEG
SAMPS	ADDR	000000BDH	SASC
SAMPS	ADDR	000000C3H	SASC1

**A.3.7 Library data output list**

78K/IV Series Librarian Vx.xx      DATE : xx xxx xx      PAGE 1

LIB-FILE NAME : K4.LIB      (xx xxx xxx)

0001 K4SUB.REL      (xx xxx xx)

CONVAH

NUMBER OF PUBLIC SYMBOLS : 1

## A.3.8 Absolute assemble lists

## (1) K4MAIN.ASM absolute assemble list

78K/IV Series Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: k4main.asm -kx -lw90

Para-file:

In-file: K4MAIN.ASM

Obj-file: K4MAIN.REL

Prn-file: K4MAIN.PRN

## Assemble list

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE STATEMENT
1	1					
2	2					NAME SAMPM
3	3					;*****
4	4					;*
5	5					;* HEX -> ASCII Conversion Program *
6	6					;*
7	7					;* main-routine *
8	8					;*
9	9					;*****
10	10					
11	11					PUBLIC MAIN,START
12	12					EXTRN CONVAH
13	13					
14	14	-----				DATA DSEG AT 0FFD20H
15	15	0FFD20				HDTSA: DS 1
16	16	0FFD21				STASC: DS 2
17	17					
18	18	-----				CODE CSEG AT 0H
19	19	000000	R8400			MAIN: DW START
20	20					
21	21	-----				CSEG
22	22	000080	09C1FF00			LOCATION 15
23	23	000084	2BCC00			START: MOV RFM,#00
24	24	000087	092000FE0F			MOVG SP,#0FFE00H
25	25	00008C	2BC400			MOV MM,#00
26	26	00008F	09C0F708			MOV STBC,#08H
27	27					

```
28 28 000093 3A201A      MOV   HDTSA,#1AH
29 29 000096 38FB20FD0F    MOVG  WHL,#HDTSA ;set hex 2-code data in WHL register
30 30
31 31 00009B R09F0AA00     CALL  CONVAH ;convert ASCII <- HEX
32 32 ;output BC-register <- ASCII code
33 33 00009F 38DB21FD0F    MOVG  TDE,#STASC ;set DE <- store ASCII code table
34 34 0000A4 D3           MOV   A,B
35 35 0000A5 50           MOV   [TDE+],A
36 36 0000A6 D2           MOV   A,C
37 37 0000A7 50           MOV   [TDE+],A
38 38
39 39 0000A8 14FE      BR    $$
40 40
41 41                END
```

## Segment informations:

ADRS	LEN	NAME
0FFD20	000003H	DATA
000000	000002H	CODE
000080	00002AH	?CSEG

(2) K4SUB.ASM absolute assemble list

78K/IV Series Assembler Vx.xx

Date:xx xxx xxxx Page: 1

Command: k4sub.asm -kx -lw90  
 Para-file:  
 In-file: K4SUB.ASM  
 Obj-file: K4SUB.REL  
 Prn-file: K4SUB.PRN

Assemble list

ALNO	STNO	ADRS	OBJECT	M	I	SOURCE STATEMENT
1	1					
2	2					NAME SAMPS
3	3					;*****
4	4					;*
5	5					;* HEX -> ASCII Conversion Program *
6	6					;*
7	7					;* sub-routine *
8	8					;*
9	9					;* input condition : (HL) <- hex 2 code *
10	10					;*
11	11					;* output condition : BC-register <-ASCII 2 code *
12	12					;*
13	13					;*****
14	14					
15	15					PUBLIC CONVAH
16	16					
17	17	-----				CSEG
18	18	0000AA	B900			CONVAH: MOV A,#0
19	19	0000AC	059F			ROL4 [WHL] ;hex upper code load
20	20	0000AE	3F0C00			CALL \$!SASC
21	21	0000B1	2431			MOV B,A ;store result
22	22					
23	23	0000B3	B900			MOV A,#0
24	24	0000B5	059F			ROL4 [WHL] ;hex lower code load
25	25	0000B7	3F0300			CALL \$!SASC
26	26	0000BA	2421			MOV C,A ;store result
27	27					
28	28	0000BC	56			RET
29	29					

```

30 30 ;*****
31 31 ;* subroutine convert ASCII code *
32 32 ;* input Acc (lower 4bits) <- hex code *
33 33 ;* output Acc <- ASCII code *
34 34 ;*****
35 35
36 36 0000BD AF0A SASC: CMP A,#0AH ;check hex code > 9
37 37 0000BF 8302 BC $SASC1
38 38 0000C1 A807 ADD A,#07H ;bias(+7)
39 39 0000C3 A830 SASC1: ADD A,#30H ;bias(+30)
40 40 0000C5 56 RET

```

ALNO STNO ADRS OBJECT M I SOURCE STATEMENT

```

41 41
42 42 END

```

Segment informations:

ADRS	LEN	NAME
0000AA	00001CH	?CSEG

## APPENDIX B NOTES ON USE

When using the RA78K4, note the following points:

**(1) Device file**

A device file is necessary to execute the RA78K4. A device file is not included in the RA78K4 package, so obtain the device file separately.

**(2) Memory directive**

The default memory area name of each device cannot be erased.

The size of the default memory area that is not used must be 0.

For the default memory area name, refer to the notes on use of the device file of each device.

Note that some segments are allocated to the default area. Exercise care when hanging an area name.

**(3) Debug options**

If compilation or structured assembly has been executed with debugging information output by a C compiler or structured assembler pre-processor, debugging information should not be output (use the -NGA option) for assembling the output assembly source. Otherwise, it is not possible to perform a C compiler or structured assembler source-level debug.

**(4) Memory initialization directives**

Codes are output even if a memory initialization directive (DW or DB) is written in data segment (DSEG).

When ordering a ROM code, an error occurs if a code exists in an address other than those of the internal ROM.

**(5) EQU definitions of SFR names**

Although SFR names can be specified for EQU quasi-directive operands, an assembly error will occur if an SFR name outside the saddr area is specified as PUBLIC.

**(6) CC78K4**

Several points must be noted when executing C source level debugging by assembling the assembler source output by the CC78K4.

For details, refer to the document supplied with the C compiler package (Notes on Use).

**(7) ID78K4/ID78K4-NS and SM78K4**

When performing debugging with ID78K4/ID78K4-NS and SM78K4, use the number of symbols and number of source lines within the limitation of the ID78K4/ID78K4-NS and SM78K4.

For details, refer to the document supplied with the debugger/simulator (Notes on Use).

**(8) When using a network**

If a directory that creates temporary files is placed in the file system that is shared on a network, a file conflict occurs, causing abnormal operations. Avoid this conflict by setting appropriate options and environmental variables.

**(9) Object converter specifications**

If Start is specified using the object converter option -U, filling will begin from the start address or the address to which the code is allocated, whichever is smaller.

The SFR area (FF00H to FFFFH) will not be filled.

Code writing format: -U filling value [, [Start], Size]

Values in the square brackets, [ ], can be omitted.

## APPENDIX C 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.

### C.1 List of Structured Assembler Options

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
1	Device type specification	-C device-type	Specifies the type of the target device.	Independent	Cannot be omitted.	p.66
2	Word symbol character specification	-SC character	Specifies the final character of a word symbol name.	Independent	-SCP	p.67
3	Symbol definition specification	-D symbol [numerical-value]	Specifies the symbol given to the #IFDEF directive, etc.	Independent	None	p.68
4	Tab number specification	-WT numerical-value, numerical-value, numerical-value	Specifies the position of output of the converted instruction.	Independent	-WT2, 3, 4	p.69
5	Include file path specification	-I path-name ... (multiple specifications OK)	Reads the include file from the specified path.	Independent	Path specified by environment variable 'INC78K4'	p.70
6	Secondary source file specification	-O [file-name]	Specifies the secondary source.	Independent	-O[input-file-name.ASM]	p.71
7	Error list file specification	-E [file-name]	Outputs the error list file.	Independent	-E[input-file-name.EST]	p.72
8	Parameter file specification	-F file-name	Inputs the input file name and option from the specified file.	Independent	Input of option and file name only possible from command line	p.73
9	Debug data output specification	-GS	Specifies the output of structured assembler source level debug data.	If the options -GS and -NGS are both specified, the one specified later is valid.	-GS	p.74
		-NGS	Invalidates the option -GS.			

**APPENDIX C LIST OF OPTIONS**

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
10	Secondary source file forcible output specification	-J	Forcibly outputs the secondary source file.	Independent	No forcible output	p.75
11	Kanji code specification	-ZS	Kanji described in the comment is interpreted as shift JIS code.	If the -ZS, -ZE, and -ZN options are specified at the same time, the one specified later takes priority.	In Windows/HP-UX: -ZS In SunOS, Solaris: -ZE	p.76
		-ZE	Kanji described in the comment is interpreted as EUC code.			
		-ZN	Characters described in the comment are not interpreted as kanji.			
12	Device file search path specification	-Y path-name	Reads the device file form the specified path.	Independent	<..\dev> for the path that activated the ST78K4.	p.77
13	Help specification	--	Outputs the help message to the display (console).	Invalidates all other options	Not displayed	p.78

**C.2 List of Assembler Options**

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
1	Device type specification	-C device-type	Specifies the device type of the target device.	Independent	Cannot be omitted	p.93
2	Object module file output specification	-O [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]	p.94
		-NO	Specifies that no object module file is output.			
3	Forced object module file output specification	-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	p.95
		-NJ	Makes option -J unavailable.			
4	Debug data output specification	-G	Specifies that local symbol 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	p.96
		-NG	Makes option -G unavailable.			
		-GA	Specifies that 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	p.97
		-NGA	Makes option -GA unavailable.			
5	Include file read path specification	-I path-name [, path-name] ... (two or more path names can be specified)	Specifies input of an include file from a specified path.	Independent	Path specified by the environmental variable 'INC78K4'	p.98
6	Assemble list file output specification	-P [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.REL]	p.99
		-NP	Makes option -P unavailable.			
7	Assemble list file data specification	-KA	Outputs an assemble list into an assemble list file.	If -KS and _KX are specified at the same time, -KS is ignored.	-KA	p.100
		-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	p.102
		-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	p.103
		-NKX	Makes option -KX unavailable.			

**APPENDIX C LIST OF OPTIONS**

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
8	Assemble list file format specification	-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.	-LW132	p.105
		-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.	-LL66	p.107
		-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	p.109
		-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	p.112
		-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	p.114
		-NLF	Makes the option -LF unavailable.			
9	Error list file output specification	-E [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	p.115
		-NE	Makes the option -E unavailable.			
10	Parameter file specification	-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 command line.	p.117
11	Specification of path for temporary file creation	-T path-name	Creates a temporary file in a specified path.	Independent	Path specified by environmental variable TMP	p.118
12	SFR area change specification	-CS [specified value]	Changes the range of the SFR area from the specified value.	If both options -CS and -CSA are specified at the same time, the option specified last takes precedence.	-CS15	p.119
		-CSA	Checks to ensure that no absolute settings are described.			
13	Kanji code specification	-ZS	Kanji described in the comment is interpreted as shift JIS code.	If the -ZS, -ZE, and -ZN options are specified at the same time, the one specified later takes priority.	In Windows/HP-UX: -ZS In SunOS: -ZE	p. 120
		-ZE	Kanji described in the comment is interpreted as EUC code.			
		-ZN	Characters described in the comment are not interpreted as kanji.			
14	Device file search path specification	-Y path-name	Reads a device file from the specified path.	Independent	<.\dev> (for the RA78K4 startup path)	p.121

APPENDIX C LIST OF OPTIONS

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
15	Symbol definition specification	-D symbol-name [= value] [, symbol-name [= value]...]	Defines a symbol.	Independent	None	p.122
16	Help specification	--	Displays a help message on the display. Description format: --	When option -- is specified, all other options are unavailable.	No display	p.123

### C.3 List of Linker Options

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
1	Load module file output specification	-O [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).LMF	p.145
		-NO	Does not output a load module file.			
2	Forced load module file output specification	-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	p.146
		-NJ	Makes option -J unavailable.			
3	Debug data output specification	-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. When option -NG is specified, the public symbol list and local symbol list cannot be output regardless of specification of -KP or -KL.	-G	p.147
		-NG	Makes option -G unavailable.			
4	Generation of stack decision symbols specification	-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	p.148
		-NS	Makes option -S unavailable.			
5	Directive file specification	-D file-name	Specifies a particular file to be input as a directive file.	Independent	—	p.150
6	Link list file output specification	-P [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 [file-name.MAP]	p.151
		-NP	Makes option -P unavailable.			
7	Link list file data specification	-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	p.152
		-NKM	Makes option -KM unavailable.			
		-KD	Outputs a link directive file into a link list file.	If options -NKM, -NKP and -NKL are all specified, the link list file cannot be output even if option -P is specified.	-KD	p.154
		-NKD	Makes option -KD unavailable.			
		-KP	Outputs a public symbol list into a link list file.	If option -NKM is specified, option -KD becomes unavailable.	-NKP	p.156
		-NKP	Makes option -KP unavailable.			
		-KL	Output a local symbol list into a link list file.	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.	-NKL	p.158
		-NKL	Makes option -KL 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.		

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
8	Link list format specification	-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	p.160
		-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, option -LF is unavailable.	-NLF	p.162
		-NLF	Makes the -LF option unavailable.			
9	Error list file output specification	-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	p.163
		-NE	Default value: -NE Makes option -E unavailable.			
10	Library file specification	-B file-name	Inputs a specific file as a library file.	Independent	—	p.164
11	Library file read path specification	-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 'LIB78K4'	p.165
12	Parameter file specification	-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 command line.	p.166
13	Specification of 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	p.167
14	Device file search path specification	-Y path-name	Reads a device file from the specified path.	Independent	Reads device files from <..\dev> (for the lk78k4.exe startup path)	p.168
15	Warning message output specification	-W [level]	Specifies whether or not a warning message is output to the console.	Independent	Outputs an ordinary error message	p.169
16	Link specification of boot area ROM program of flash ROM model	-ZB	Specifies the first address of the flash ROM area.	Independent	None	p.170
17	Help specification	--	Displays a help message on the display.	All other options are unavailable.	No display	p.171

**C.4 List of Object Converter Options**

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
1	HEX format object module file output specification	-O [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)	p.199
		-NO	No HEX format object module file is output.			
2	Symbol table file output specification	-S [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)	p.201
		-NS	Does not output a symbol table file.			
3	Specification of 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 becomes unavailable.	-NR	p.202
		-NR	Makes option -R unavailable.			
4	Object complement specification	-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.	—	p.203
5	Error list file output specification	-E [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	p.205
		-NE	Makes option -E unavailable.			
6	Parameter file specification	-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 command line.	p.206
7	HEX format specification	-KI	Specifies Intel standard HEX format.	When any two or more of -KI, -KIE, -KT, -KM and -KME are specified at the same time, the last specified option takes precedence.	-KIE	p.207
		-KIE	Specifies Intel extended HEX format.			
		-KT	Specifies Extended Tech format.			
		-KM	Specifies Motorola S-type format (standard address).			
		-KME	Specifies Motorola S-type format (32-bit address).			

**APPENDIX C LIST OF OPTIONS**

---

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
8	Device file search path specification	-Y path-name	Reads a device file from the specified path.	Independent	<..\dev> for path by which OC78K4 was started up	p.208
9	File separate output specification for flash ROM model	-ZF	Adds an option that separately outputs the boot area and other areas to separate HEX format files when linking of the boot area ROM program of a flash memory model is specified.	Independent	None	p.209
10	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.210

### C.5 List of Librarian Options

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
1	List file format specification	-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	p.223
		-LL [number-of-lines]	Changes the number of lines that can be printed in 1 page in a list file.		-LL66	p.224
		-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	p.225
		-NLF	Makes the option -LF unavailable.			
2	Specification of 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.	p.226
3	Device file search path specification	-Y path-name	Reads a device file from the specified path.	Independent	<..\dev> (for the lb78k4 startup path)	p.227
4	Help specification	--	Displays a help message on the display.	All other options are unavailable.	No display	p.228

**C.6 List of List Converter Options**

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
1	Object module file input specification	-R [file-name]	Specifies the input of an object module file.	Independent	-R [assemble-list-file-name.REL]	p.250
2	Load module file input specification	-L [file-name]	Inputs a load module file.	Independent	-L [assemble-list-file-name.LMF]	p.251
3	Absolute assemble list file output specification	-O [file-name]	Outputs an absolute assemble list file.	Independent	-O [assemble-list-file-name.P]	p.252
4	Error list file output specification	-E [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	p.253
		-NE	Makes option -E unavailable.			
5	Parameter file specification	-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 command line.	p.254
6	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.255

## C.7 List of ECC Generator Options

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
1	ECC-supported HEX-format object module file name specification	-O [output file name]	Specifies the name and output destination of the ECC-supported HEX-format object module.	Independent	-O input file name .ECG	p.268
2	Parameter file specification	-F file name	Inputs the input file name and options from the specified file.	Independent	Options and input file names can only be input from the command line.	p.269
3	Address specification	-A [address specification 1] [, address specification 2] [, address specification 3] [, address specification 4]]	Specifies the address of each type of data used by the ECC generator.	Independent	<ul style="list-style-type: none"> <li>• The input HEX data end address is 7FFFH.</li> <li>• The ECC error correction code start address is 8000H.</li> <li>• The ECW data start address is A000H.</li> <li>• The ECW data error correction code start address is A004H.</li> </ul>	p.270
4	Forced ECG file output specification	-J	Specifies whether or not to output the ECG file.	Independent	-NJ	p.271
5	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.272

## APPENDIX D 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	Format	Function	Abbrev. Format	Ref. Page
1	CREATE	CREATE $\Delta$ library-file-name [ $\Delta$ transaction]	Creates a new library file.	C	p.230
2	ADD	ADD $\Delta$ library-file-name $\Delta$ transaction	Adds a module to a library file.	A	p.231
3	DELETE	DELETE $\Delta$ library-file-name $\nabla$ ( $\nabla$ module-name [ $\nabla$ , ...] $\nabla$ )	Deletes a module from a library file.	D	p.232
4	REPLACE	REPLACE $\Delta$ library-file-name $\Delta$ transaction	Replaces one module with another in a library file.	R	p.233
5	PICK	PICK $\Delta$ library-file-name $\nabla$ ( $\nabla$ module-name [ $\nabla$ , ...] $\nabla$ )	Retrieves a specified module from an existing library file.	P	p.234
6	LIST	LIST[ $\Delta$ option] $\Delta$ library-file-name [ $\nabla$ ( $\nabla$ module-name [ $\nabla$ , ...] $\nabla$ )]	Outputs data on modules in a library file.	L	p.235
7	HELP	HELP	Displays a help message on the display (console).	H	p.236
8	EXIT	EXIT	Exits the librarian.	E	p.237

## APPENDIX E INDEX

### [A]

-A (ECCGEN).....	270
Abort error.....	295
Absolute assemble list.....	243
Absolute assemble list file.....	242
ADD.....	231
.ASM.....	59, 83
Assemble list.....	243, 276, 291
Assemble list file.....	242
Assembler.....	13, 23, 83
AT.....	131

### [B]

-B (LK78K4).....	164
------------------	-----

### [C]

-C (RA78K4).....	93
-C (ST78K4).....	66
COMPLETE.....	132
CREATE.....	230
Cross-reference list.....	279
-CS (RA78K4).....	119
-CSA (RA78K4).....	119

### [D]

-D (LK78K4).....	150
-D (RA78K4).....	122
-D (ST78K4).....	68
DELETE.....	232
.DR (LK78K4).....	129

### [E]

-E (LCNV78K4).....	253
-E (LK78K4).....	163
-E (OC78K4).....	205
-E (RA78K4).....	115
-E (ST78K4).....	72
.ECG.....	262
.ELK.....	129
.ELV.....	242
Environmental variable.....	37
.EOC.....	179
.ERA.....	83
Error list.....	280, 284, 285, 287
.EST.....	59

Execution procedure.....	38, 43, 49
EXIT.....	237

### [F]

-F (ECCGEN).....	269
-F (LCNV78K4).....	254
-F (LK78K4).....	166
-F (OC78K4).....	206
-F (RA78K4).....	117
-F (ST78K4).....	73
Fatal error.....	295

### [G]

-G (LK78K4).....	147
-G (RA78K4).....	96
-GA (RA78K4).....	97
-GS (ST78K4).....	74

### [H]

HELP.....	236
.HEX.....	179, 262

### [I]

-I (LK78K4).....	165
-I (RA78K4).....	98
-I (ST78K4).....	70
INC78K4.....	290
Installation.....	34
Intel standard HEX format.....	181

### [J]

-J (ECCGEN).....	271
-J (LK78K4).....	146
-J (RA78K4).....	95
-J (ST78K4).....	75

### [K]

-KA (RA78K4).....	100
Kanji code.....	37, 76, 120
-KD (LK78K4).....	154
-KI (OC78K4).....	207
-KIE (OC78K4).....	207
-KL (LK78K4).....	158
-KM (LK78K4).....	152
-KM (OC78K4).....	207
-KME (OC78K4).....	207

-KP (LK78K4).....	156	-NGA (RA78K4).....	97
-KS (RA78K4).....	102	-NGS (ST78K4).....	74
-KT (OC78K4).....	207	-NJ (ECCGEN).....	271
-KX (RA78K4).....	103	-NJ (LK78K4).....	146
<b>[L]</b>		-NJ (RA78K4).....	95
-L (LCNV78K4).....	251	-NKA (RA78K4).....	100
LANG78K.....	290	-NKD (LK78K4).....	154
-LF (LB78K4).....	225	-NKL (LK78K4).....	158
-LF (LK78K4).....	162	-NKM (LK78K4).....	152
-LF (RA78K4).....	114	-NKP (LK78K4).....	156
-LH (RA78K4).....	109	-NKS (RA78K4).....	102
.LIB.....	129, 215	-NKX (RA78K4).....	103
LIB78K4.....	290	-NLF (LB78K4).....	225
Librarian.....	29, 215	-NLF (LK78K4).....	162
Link directive.....	55, 131	-NLF (RA78K4).....	114
Link list file.....	56, 129	-NO (LK78K4).....	145
Linker.....	24, 129	-NO (OC78K4).....	199
LIST.....	235	-NO (RA78K4).....	94
List converter.....	27, 241	-NP (LK78K4).....	151
-LL (LB78K4).....	224	-NP (RA78K4).....	99
-LL (LK78K4).....	160	-NR (OC78K4).....	202
-LL (RA78K4).....	107	-NS (LK78K4).....	148
.LMF.....	129, 179, 242	-NS (OC78K4).....	201
Load module file.....	56, 129, 179, 242	<b>[O]</b>	
Local symbol list.....	284	-O (ECCGEN).....	268
.LST.....	215	-O (LCNV78K4).....	252
-LT (RA78K4).....	112	-O (LK78K4).....	145
-LW (LB78K4).....	223	-O (OC78K4).....	199
-LW (RA78K4).....	105	-O (RA78K4).....	94
<b>[M]</b>		-O (ST78K4).....	71
.MAP.....	129	Object complement specification.....	203
Map list.....	282	Object converter.....	25, 178
Maximum performance.....	29	Option setting from PM plus.....	80, 125, 173, 211, 238, 256
MEMORY.....	132	<b>[P]</b>	
Memory area.....	130	.P.....	242
Memory directive.....	131	-P (LK78K4).....	151
Memory space.....	130	-P (RA78K4).....	99
MERGE.....	132	Parameter file.....	59, 62, 83, 87, 129, 139, 179, 195, 242, 247
<b>[N]</b>		PATH.....	290
-NE (LCNV78K4).....	253	.PEC.....	262
-NE (LK78K4).....	163	PICK.....	234
-NE (OC78K4).....	205	.PLK.....	129
-NE (RA78K4).....	115	.PLV.....	242
-NG (LK78K4).....	147	PM plus use.....	292
-NG (RA78K4).....	96		

.POC ..... 179  
 .PRA ..... 83  
 .PRN ..... 83, 242  
 .PST ..... 59  
 Public symbol list ..... 283

**[R]**

-R (LCNV78K4) ..... 250  
 -R (OC78K4) ..... 202  
 RAM ..... 130  
 REGULAR ..... 134, 136  
 .REL ..... 83, 129, 215, 242  
 REPLACE ..... 233  
 ROM ..... 130

**[S]**

-S (LK78K4) ..... 148  
 -S (OC78K4) ..... 201  
 Sample program ..... 327  
 -SC (ST78K4) ..... 67  
 Segment location directive ..... 131, 135  
 SEQUENT ..... 132  
 Structured assembler ..... 22, 59  
 Subcommand ..... 229  
 .SYM ..... 179

**[T]**

-T (LB78K4) ..... 226  
 -T (LK78K4) ..... 167  
 -T (RA78K4) ..... 118  
 TMP ..... 290

**[U]**

-U (OC78K4) ..... 203

**[W]**

-W (LK78K4) ..... 169  
 Warning error ..... 295  
 -WT (ST78K4) ..... 69

**[Y]**

-Y (LB78K4) ..... 227  
 -Y (LK78K4) ..... 168  
 -Y (OC78K4) ..... 208  
 -Y (RA78K4) ..... 121  
 -Y (ST78K4) ..... 77

**[Z]**

-ZB (LK78K4) ..... 170  
 -ZE (RA78K4) ..... 120  
 -ZE (ST78K4) ..... 76  
 -ZF (OC78K4) ..... 209  
 -ZN (RA78K4) ..... 120  
 -ZN (ST78K4) ..... 76  
 -ZS (RA78K4) ..... 120  
 -ZS (ST78K4) ..... 76

**[Symbol]**

-- (ECCGEN) ..... 272  
 -- (LB78K4) ..... 228  
 -- (LCNV78K4) ..... 255  
 -- (LK78K4) ..... 171  
 -- (OC78K4) ..... 210  
 -- (RA78K4) ..... 123  
 -- (ST78K4) ..... 78