To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

User's Manual

RA78K0S

Assembler Package Ver. 1.40 or Later

Operation

Target Device
78K0S Series

Document No. U16656EJ1V0UM00 (1st edition)
Date Published July 2003 N CP(K)

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.

Unix is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

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.

M8E 02. 11-1

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

- **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.
Hong Kong
Tel: 2886-9318

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 who wish to develop software using the RA78K0S an understanding of the functions of each program in the RA78K0S Series Assembler Package (hereafter referred to as "the RA78K0S") and of the correct methods of using the package.

This manual does not cover the language, such as the expressions of assembler directives and source programs used in the RA78K0S. Therefore, before reading this manual, read the **RA78K0S Assembler Package Language User's Manual (U16657E)** (hereafter referred to as "Language").

The contents of this manual are intended for use with Ver. 1.40 or later of the RA78K0S.

[Target Readers]

The RA78K0S is intended for users who understand the functions and instructions of the microcontroller to be developed (78K0S Series).

[Organization]

This manual consists of the following chapters and appendixes:

CHAPTER 1 GENERAL

Outlines the role of the RA78K0S in microcontroller software development and the features of the RA78K0S.

CHAPTER 2 PRODUCT OUTLINE AND INSTALLATION

Explains the program file names and operating environment provided by the RA78K0S.

CHAPTER 3 EXECUTING RA78K0S

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 RA78K0S 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 PROGRAM OUTPUT LIST

Explains the formats of the lists output by each program.

CHAPTER 11 EFFICIENT USE OF RA78K0S

Introduces some measures for optimum utilization of the RA78K0S.

CHAPTER 12 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 RA78K0S.

The instruction sets are not detailed in this manual.

For these instructions, refer to the user's manual of the microcontroller to be 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 RA78K0S, read **CHAPTER 3 EXECUTING RA78K0S**.

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

[Caution]

In this manual, it is assumed that a PC-9800 Series personal computer 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 set-up 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.
< >:	Indicates window names or dialog names
" ":	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.
∆:	Indicates one or more blank or TAB.
∇:	Indicates zero or more blanks or TABs (i.e. blanks may be omitted).
/:	Indicates a break between characters.
~:	Indicates continuity.
[↵]:	Indicates pressing of the Return key.
Note:	Footnote for item marked with **Note** in the text
Caution:	Information requiring particular attention
Remark:	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 Number
CC78K0S C Compiler Ver. 1.50 or Later	Operation	U16654E
	Language	U16655E
RA78K0S Assembler Package Ver. 1.40 or Later	Operation	This manual
	Language	U16657E
	Structured Assembly Language	U11623E
SM78K0S System Simulator	Operation	To be prepared
ID78K0S-NS Integrated Debugger Ver. 2.52 or Later	Operation	U16584E
78K0S Series OS MX78K0S	Fundamental	U12938E
PM plus Ver. 5.10		To be prepared

Caution The contents of the above related documents are subject to change without notice. Be sure to use the latest edition of each document when designing your system.

CONTENTS

CHAPTER 1 GENERAL

This chapter describes the role of the RA78K0S in microcontroller software development and the features of the RA78K0S.

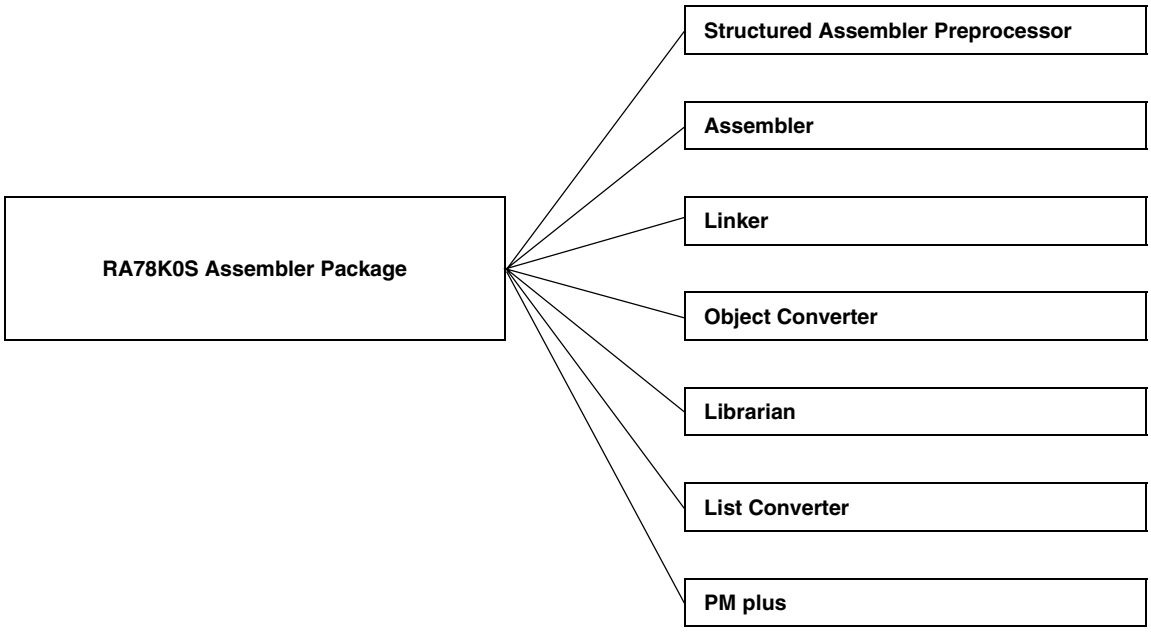
1.1 Assembler Overview

The RA78K0S Assembler Package is a generic term for a series of programs designed to translate source programs coded in the assembly language for 78K0S Series microcontrollers into machine language coding.

The RA78K0S contains six programs: Structured Assembler Preprocessor, Assembler, Linker, Object Converter, Librarian, and List Converter.

In addition, PM plus that allows users to easily perform a series of operations, including editing, compiling/assembling, linking, and debugging on Windows™ is supplied with the RA78K0S.

Figure 1-1. RA78K0S Assembler Package



1.1.1 What is an assembler?

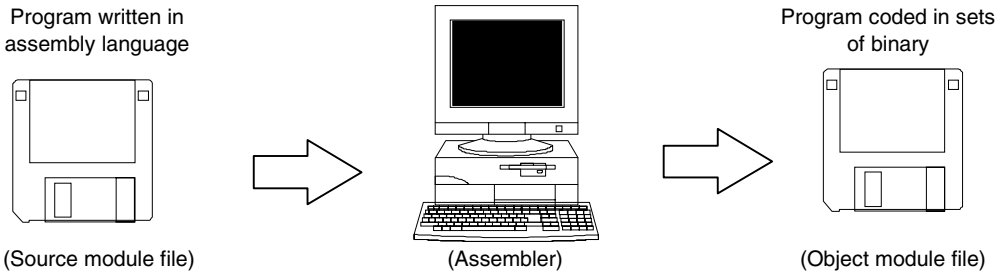
(1) Assembly language and machine language

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.

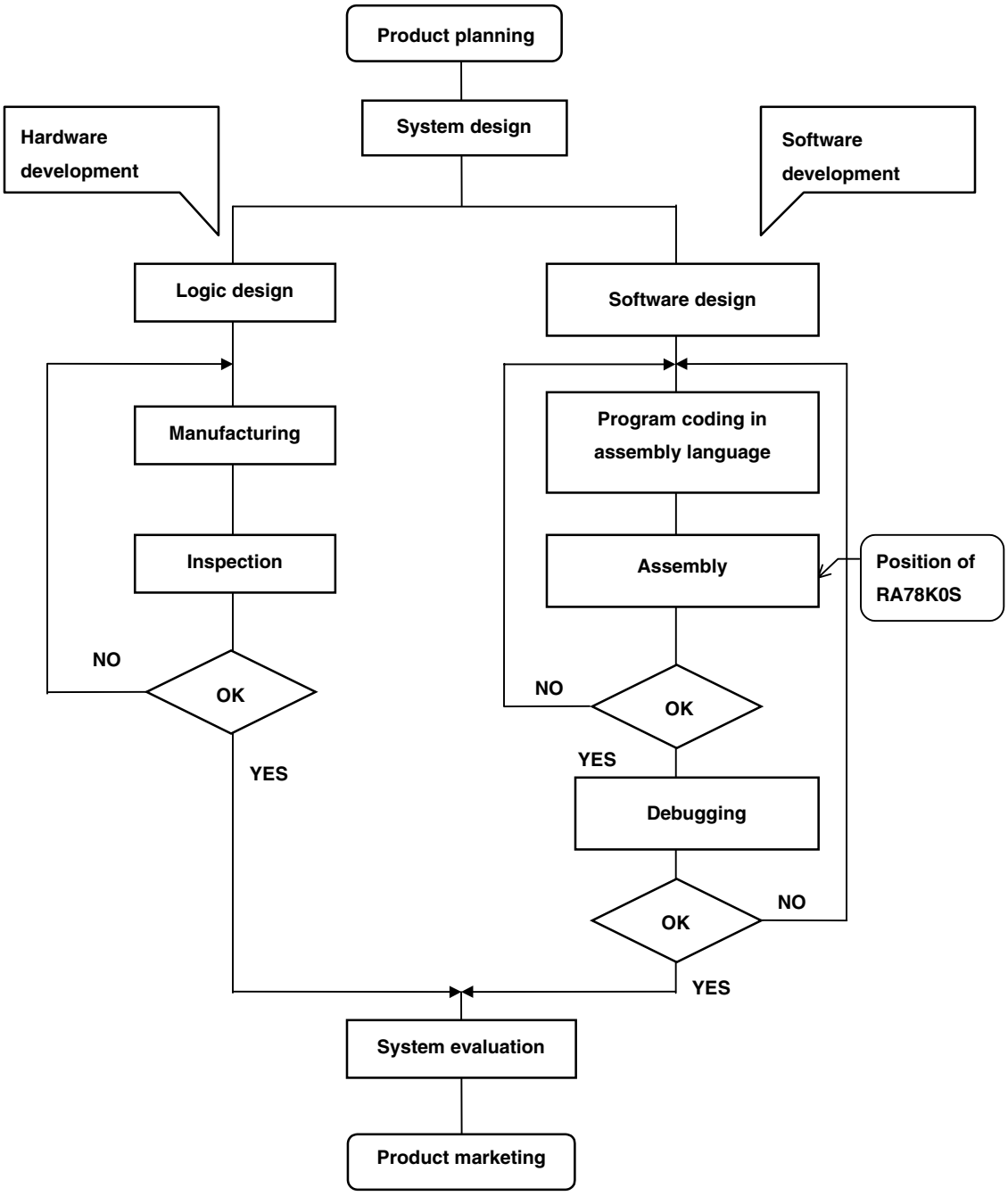
Figure 1-2. Flow of Assembler



(2) Development of microcontroller-related products and the role of RA78K0S

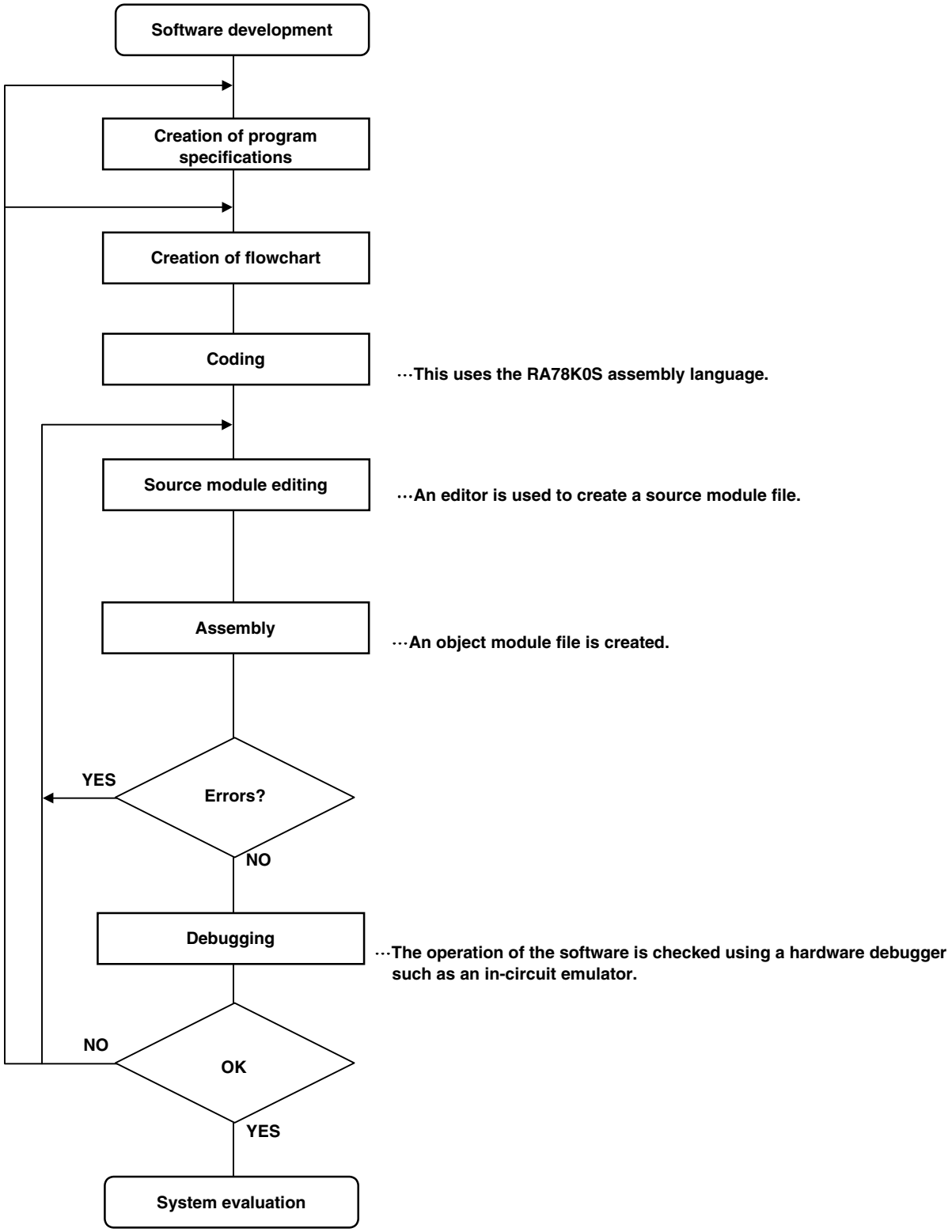
Figure 1-3 Development Process of Microcontroller-Applied Products illustrates the position of assembly-language programming in the (software) product development process.

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



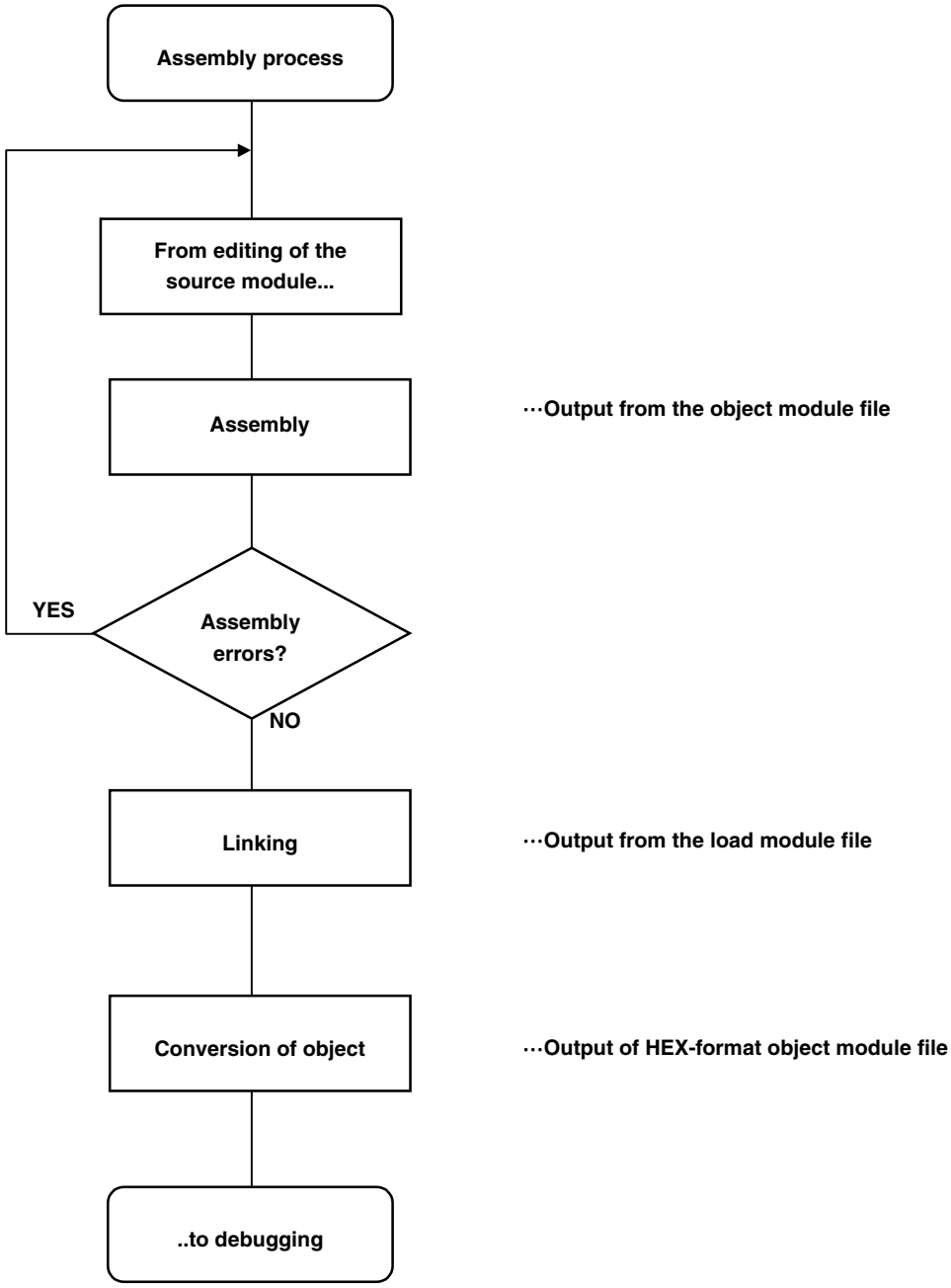
A more detailed explanation of the software development process appears in **Figure 1-4 Software Development Process.**

Figure 1-4. Software Development Process



The RA78K0S is then applied to the assembly process.

Figure 1-5. RA78K0S 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.

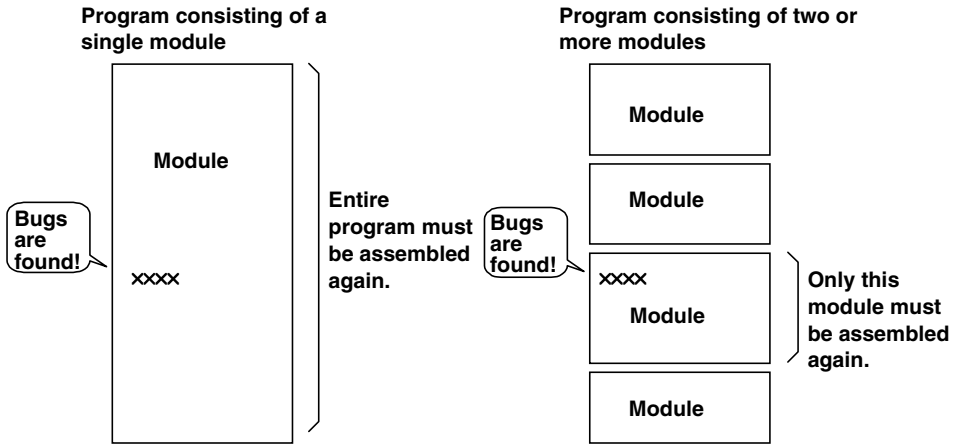
A relocatable assembler is an assembler suitable for modular programming. The following advantages can be derived from modular programming with a relocatable assembler:

(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.

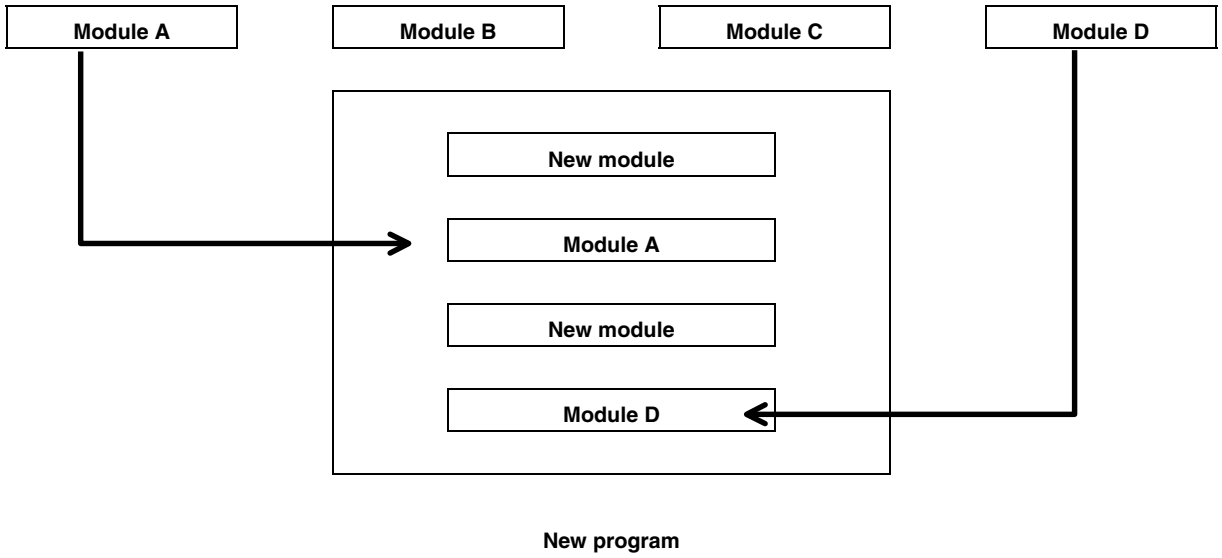
Figure 1-6. Reassembly for Debugging



(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

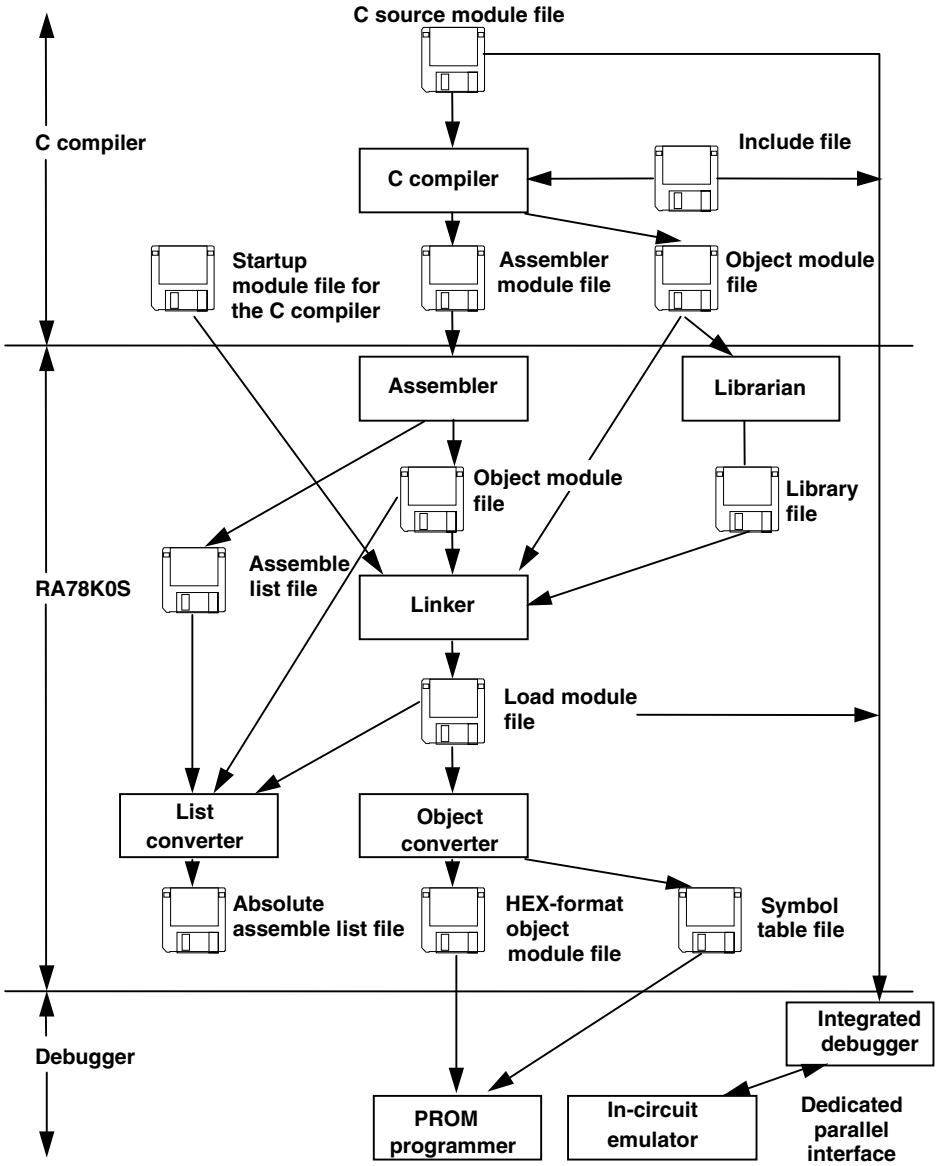


1.2 Overview of Features of RA78K0S

The procedure for developing general programs appears in **Figure 1-8 Procedure for Program Development Using RA78K0S**. 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 "RA78K0S," the assembler program is referred to as the "assembler."

Figure 1-8. Procedure for Program Development Using RA78K0S



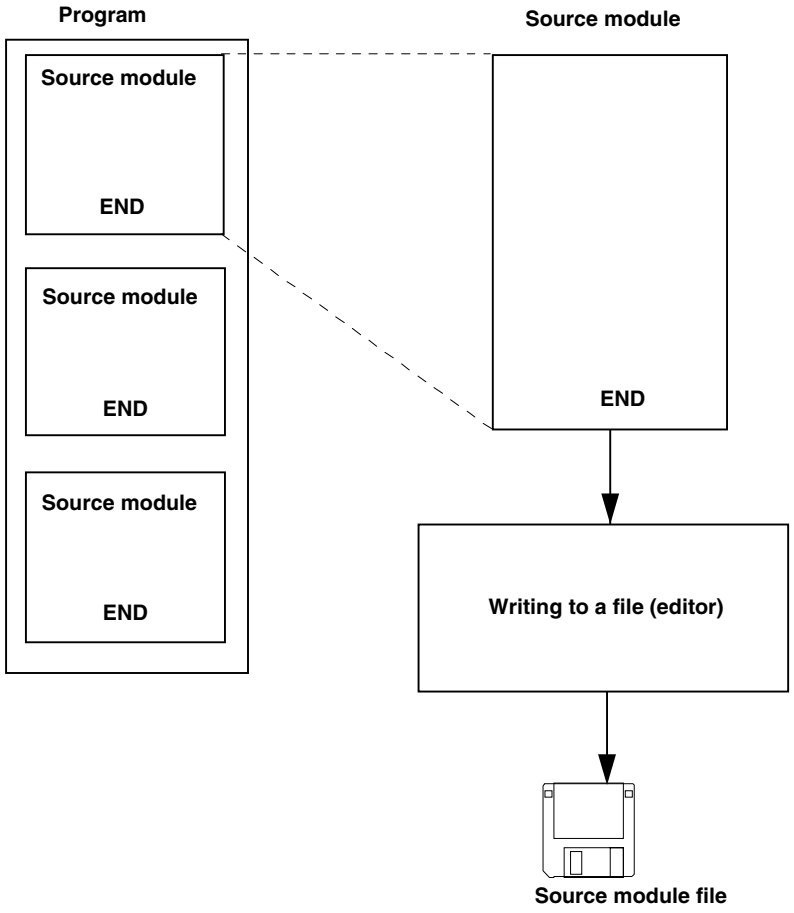
1.2.1 Creating a source module file using an editor

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

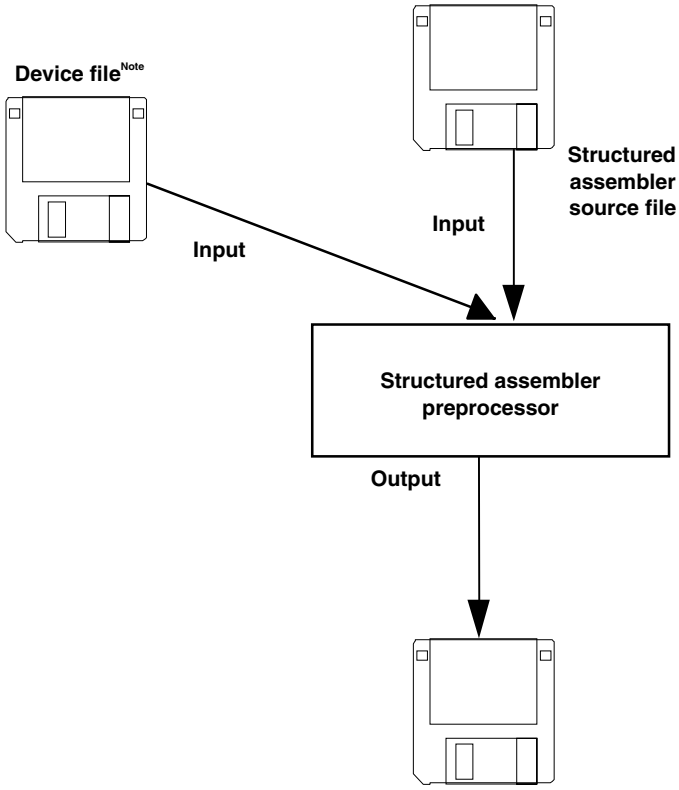


1.2.2 Structured assembler preprocessor

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 output the source program for the assembler.

For the structured assembler preprocessor and structured assembly language, refer to the separate document **RA78K0S Structured Assembly Language User's Manual (U11623E)**.

Figure 1-10. Function of Structured Assembler Preprocessor

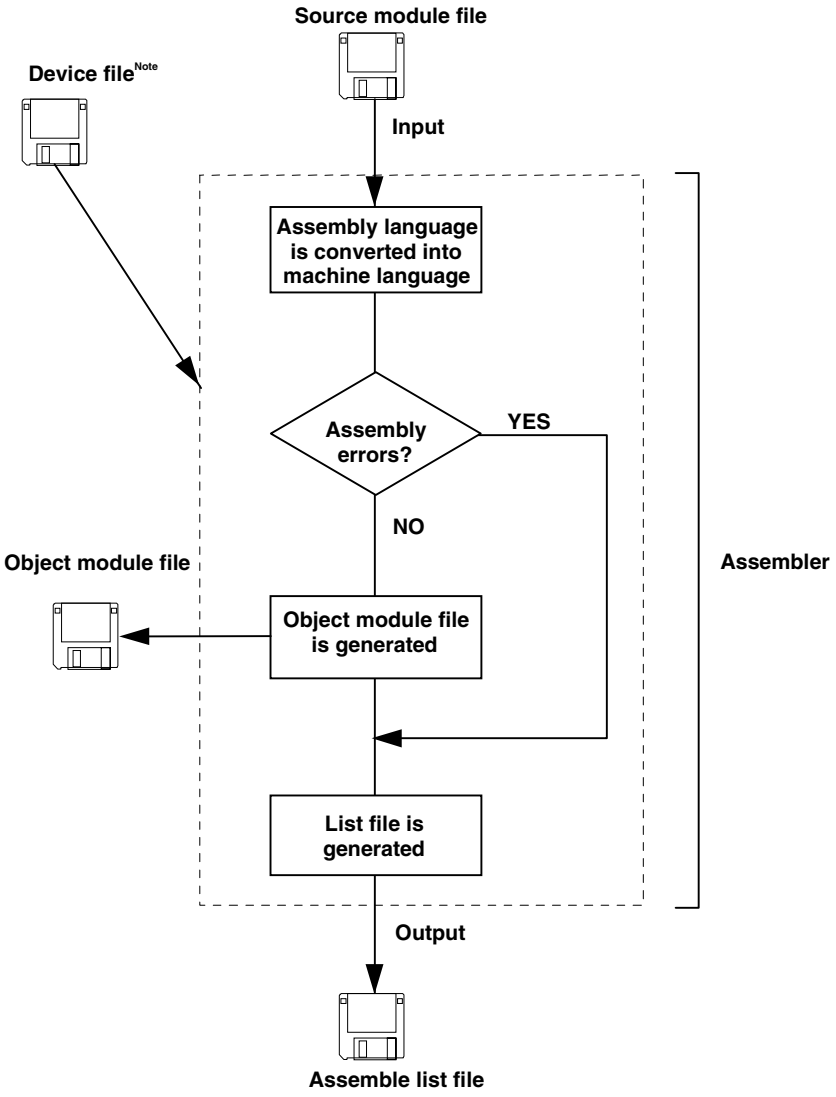


Note Obtain the device file separately.

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



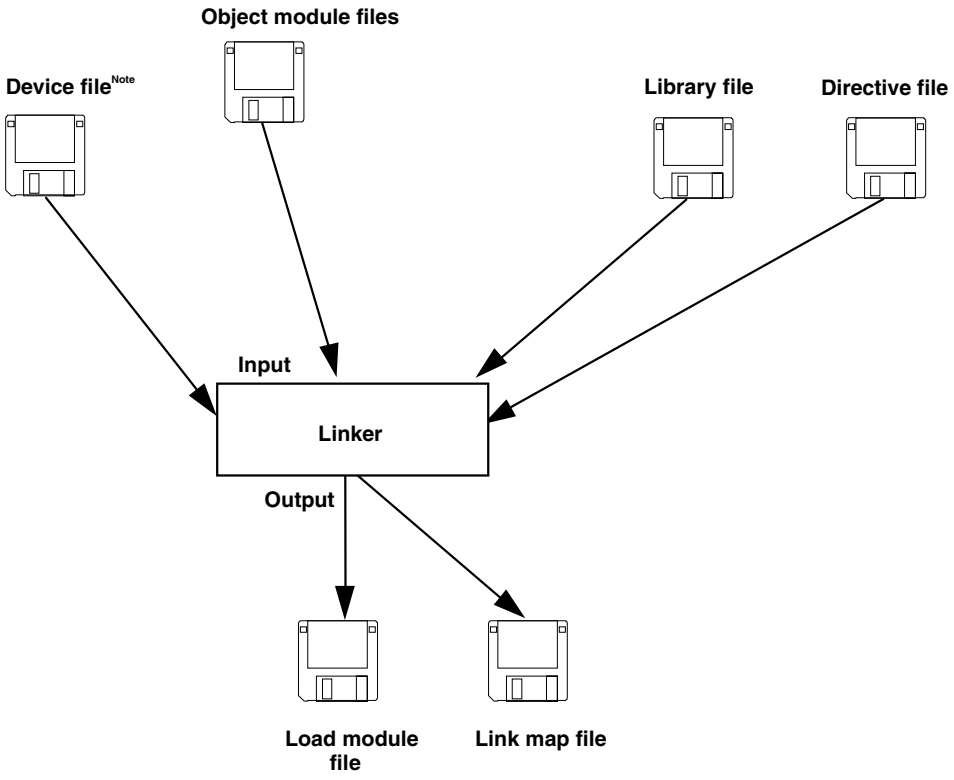
Note Obtain the device file separately.

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. Functions of Linker



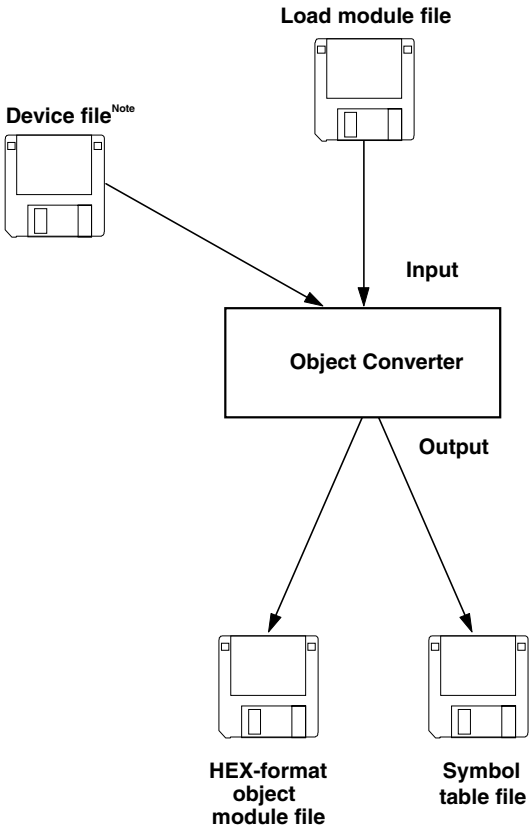
Note Obtain the device file separately.

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.

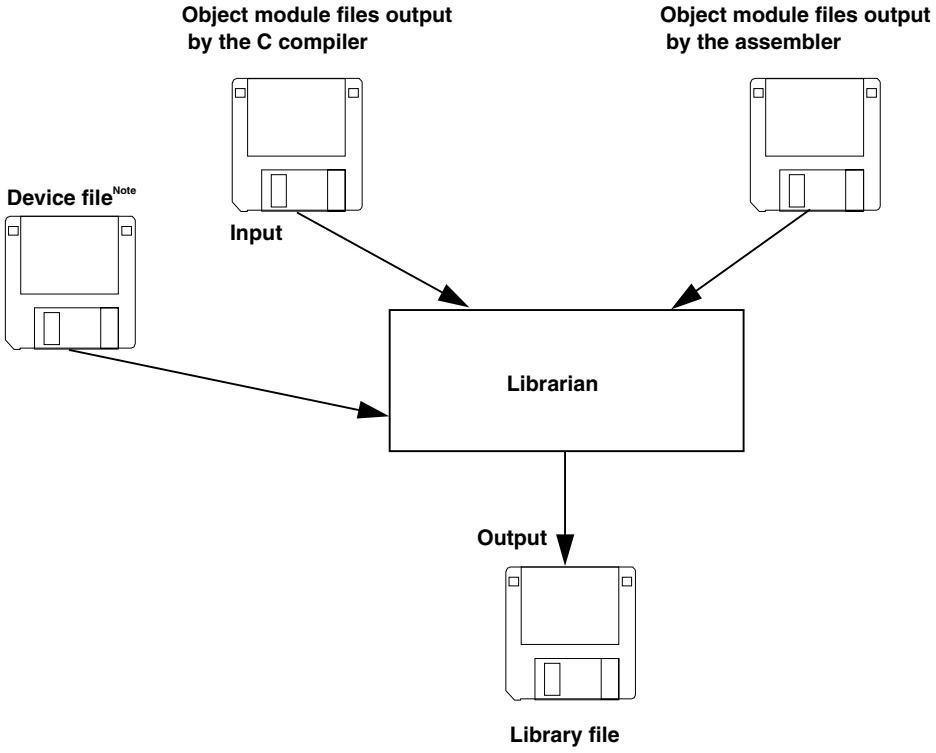
1.2.6 Librarian

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



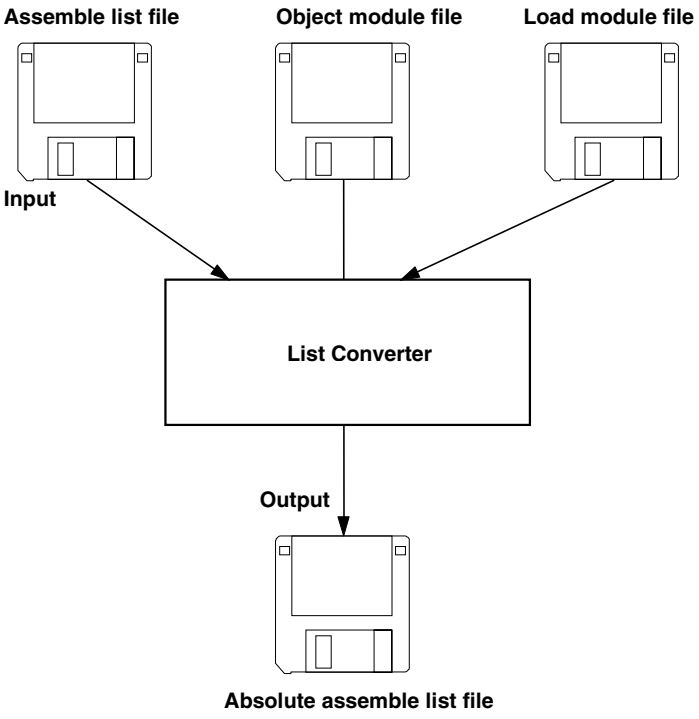
Note Obtain the device file separately.

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



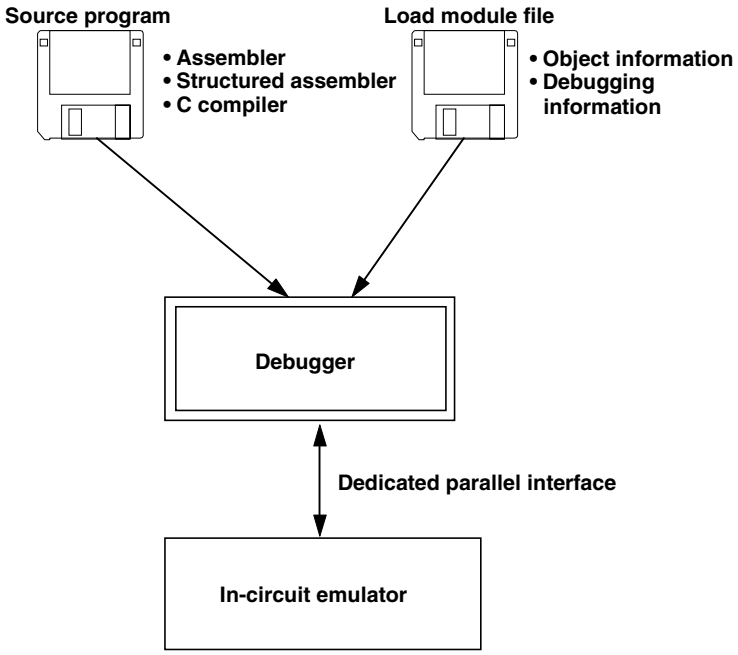
1.2.8 Debugger

The debugger for the 78K0S 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 RA78K0S.

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 characteristics of RA78K0S

(1) Maximum performance of assembler

Table 1-1. Maximum Performance of Assembler

Item	Maximum Performance	
	PC Version	EWS Version
Number of symbols (local + public)	65,535[Note 1]	65,535[Note 1]
Number of symbols that can be output to cross reference list	65,534[Note 2]	65,534[Note 2]
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	100	100
Macros and include specifications in 1 file	10,000	10,000
Macros and include specifications in 1 include file	10,000	10,000
Relocation information[Note 3]	65,535	65,535
Line number information	65,535	65,535
BR/CALL directive in 1 file	32,767	32,767
Number of characters on 1 line of source code	2,048[Note 4]	2,048[Note 4]
Symbol length	256 characters	256 characters
Number of switch name definitions	100	100
Character length of switch name	31	31
Include file nesting levels in 1 file	8 levels	8 levels

Notes 1. If more than 2,001 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 2,048 characters are written on one line, an error message is output and processing ends.

(2) Maximum performance of linker

Table 1-2. Maximum Performance of Linker

Item	Maximum Performance	
	PC Version	EWS Version
Number of symbols (local + public)	65,535	65,535
Line number information of the same segment	65,535	65,535
Number of segments	65,535	65,535
Input modules	1,024	1,024

1.4 Features of RA78K0S

The RA78K0S 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 RA78K0S 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.

CHAPTER 2 PRODUCT OUTLINE AND INSTALLATION

This chapter explains the procedure used to install the files stored in the supply media of the RA78K0S 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)Note	CD-ROM	Supports Windows standard installer
IBM PC/AT compatible	Japanese Windows (98/Me/2000/XP/NT 4.0)Note English Windows (98/Me/2000/XP/NT 4.0)Note		
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 RA78K0S 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 RA78K0S 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 RA78K0S\DISK1 folder.

<3> Confirmation of files

Using Windows Explorer, etc., check that the files contained in the RA78K0S 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 /nectools/bin is assumed here.

<1> Login

Log in to the host machine.

<2> Directory selection

Go to the install directory.

```
%cd /nectools/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 /nectools/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 RA78K0S.

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 RA78K0S installation.

If currently using a 32-bit environment, register the device file used for the RA78K0S (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 RA78K0S 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. Install PM plus and the RA78K0S in the same directory.

Figure 2-1. Directory Configuration

NECtools32\	
bin\	
ra78k0s.exe	Execution format of assembler
st78k0s.exe	Execution format of structured assembler preprocessor
lk78k0s.exe	Linker
oc78k0s.exe	Object converter
lc78k0s.exe	List converter
lb78k0s.exe	Librarian
lb78k0se.exe	Interface tool between library and DLL of PM plus environment
lb78k0sp.exe	Standalone startup library
ra78k0s.is	File used by assembler
*78k0sp.dll	DLL tool for PM plus
*78k0s.hlp	Help file for starting command line
doc\	User's manual and supplementary explanations
hlp\	Online manual
setup\	Data files for installation and uninstallation
smp78k0s\ra78k0s\	
k0smain.asm	Assembler sample program
k0ssub.asm	Assembler sample program
ra.bat	Batch file for assembler sample program
readme.doc	Explanation of for 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 /nectools/bin.

```
——nectools/
        bin/
          ra78k0s          Executable format of assembler
          st78k0s          Executable format of structured assembler preprocessor
          lk78k0s          Executable format of linker
          oc78k0s          Executable format of object converter
          lc78k0s          Executable format of list converter
          lb78k0s          Executable format of librarian
          *.hlp            Help file corresponding to each program (text file)
          ra78k0s.is*      Table file defining instruction set used by assembler
          *.asm,*.s        Sample program for installation confirmation
          *.sh             Batch file 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 RA78K0S
After selecting “NEC RA78K0S 78K/0 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

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)
INC78K0S	Specifies a directory where the include file is searched.
LIB78K0S	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 INC78K0S = C:\NECTools32\inc78k0s
set LIB78K0S = C:\NECTools32\lib78k0s
set LANG78K = SJIS
```

With HP9000 Series 700 or SPARCstation family

- Example when `csh` is used

```
set path = ($path /ra78k0s)
setenv INC78K0S /ra78k0s
setenv LIB78K0S /ra78k0s
setenv LANG78K EUC
```

- Example when `sh` is used

```
PATH = $PATH:/ra78k0s
INC78K0S = /ra78k0s
LIB78K0S = /ra78k0s
export PATH INC78K0S LIB78K0S
setenv LANG78K EUC
```

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 (KANJICODE), or kanji code specification option (-ZE/-ZS/-ZN).

CHAPTER 3 EXECUTING RA78K0S

This chapter explains the procedures for using the assembler package RA78K0S, from assembling to object conversion.

Sample programs "k0smain.asm" and "k0ssub.asm" are assembled, linked, and converted into objects in accordance with the execution procedures explained in this section. In the execution procedures from the structured assembler to the object conversion, sample programs "test1.s" and "test2.s" are used.

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 RA78K0S

3.1.1 Sample programs

Among the files stored on the system disk are [K0SMAIN.ASM] and [K0SSUB.ASM]. These files are a sample program for use in verifying the operation of the assembler package.

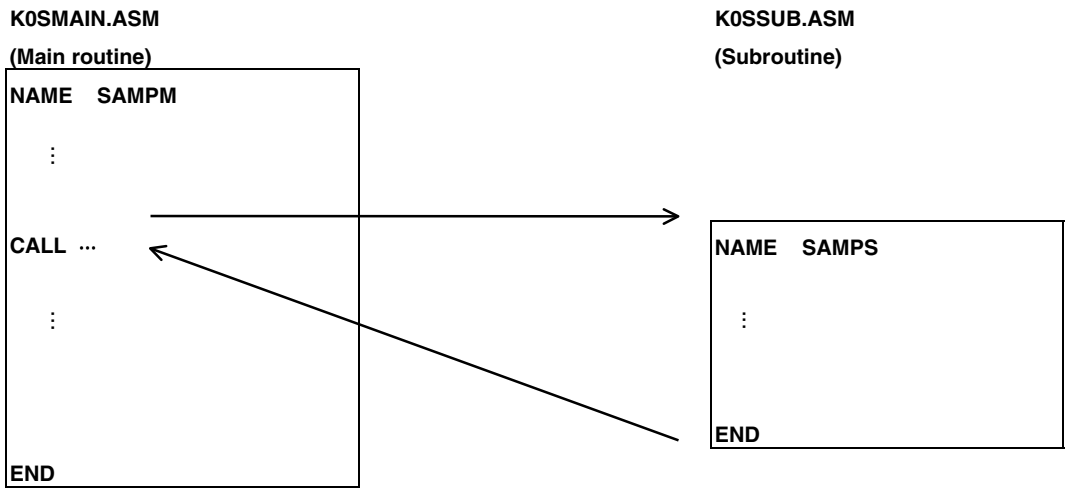
In later assembler operation, these files will be input to the assembler as source program files.

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 source module file (K0SMAIN.ASM).

The name of the subroutine module is SAMPS, and it is stored in source module file (K0SSUB.ASM).

Figure 3-1. Structure of Sample Program



■**K0SMAIN.ASM (Main routine)**

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

        PUBLIC MAIN, START
        EXTRN  CONVAH
        EXTRN  _@STBEG

DATA    DSEG   saddr
HDTSA:  DS     1
STASC:  DS     2

CODE    CSEG   AT 0H
MAIN:   DW     START

        CSEG
START:
        ;chip initialize
        MOVW   AX, #_@STBEG
        MOVW   SP, AX

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

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

        BR     $$

        END
```

■**K0SSUB.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:
        MOV     A, [HL]
        ROR     A, 1
        ROR     A, 1
        ROR     A, 1
        ROR     A, 1
        AND     A, #0FH         ;hex upper code load
        CALL    !SASC
        MOV     B, A            ;store result

        XOR     A, A
        XCH     A, [HL]
        AND     A, #0FH         ;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 (+7H)
SASC1:  ADD     A, #30H         ;bias (+30H)
        RET

        END
```

Remark This sample program is a reference program, prepared for the purpose of teaching you about the functions and operation of the RA78K0S. It cannot be used as an application program.

3.1.2 Configuration of sample program

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

K0smain.asm	Main module
K0ssub.asm	Submodule
mylib.lib	Library file (this is not used here.)
sample.dr	Directive file

3.2 Execution Procedure of RA78K0S

The batch files (ra.bat) in the system disk are used for the RA78K0S operation.

The assembler, linker, object converter, and list converter are executed in this order using “k0smain.asm” and “k0ssub.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 μPD789024 as the target device.

■ ra.bat (batch program for verifying RA78K0S operation)

```
echo off
cls
set      LEVEL=0

if "%1" == "" goto ERR_BAT

ra78k0s -C%1 k0smain.asm
if errorlevel 1 set LEVEL=1
ra78k0s -C%1 k0ssub.asm
if errorlevel 1 set LEVEL=1
if %LEVEL% == 1 echo Assemble error !!
if %LEVEL% == 1 goto END

cls
lk78k0s k0smain.rel k0ssub.rel -s -orasample.lmf -prasample.map
if errorlevel 1 echo Link error !!
if errorlevel 1 goto END

cls
oc78k0s rasample
if errorlevel 1 echo Object conversion error !!
if errorlevel 1 goto END

cls
set LEVEL=0
lc78k0s -ltest.lmf -rk0smain.rel k0smain.prn
if errorlevel 1 set LEVEL=1
lc78k0s -lrasample.lmf -k0ssub.rel k0ssub.pn
```

```
    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) Execute the batch file.

Specify the target device type and execute the RA78K0S-operation verification batch program.

```
A>ra.bat 9024
```

The following message is output to the display.

```
78K/0S Series Assembler Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

PASS_PARSE Start
PASS_OUTOBJ Start

 Target chip : uPD789024
 Device file : Vx.xx

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

78K/0S Series Assembler Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

PASS/PARSE Start
PASS/OUTOBJ Start

 Target chip : uPD789024
 Device file : Vx.xx

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

Clear the screen.

```
78K/0S Series Linker Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx
```

```
 Target chip : uPD789024
 Device file : Vx.xx

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

Clear the screen.

```
78K/0S Series Object Converter Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

 Target chip : uPD789024
 Device file : Vx.xx

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

Clear the screen.

```
List Conversion Program for RA78K/0S Vx.xx [xx xxx xxxx]
  Copyright (C) NEC Electronics Corporation xxxx,xxxx

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

List Conversion Program for RA78K/0S x.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.

k0smain.rel:	Object module file
k0smain.prm:	Assemble list file
k0ssub.rel:	Object module file
k0ssub.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
k0smain.p:	Absolute assemble list file
k0ssub.p:	Absolute assemble list file

(3) Summary of RA78K0S execution procedure

The following is a brief summary of the execution procedure of the RA78K0S.

Figure 3-2. RA78K0S Execution Procedure 1

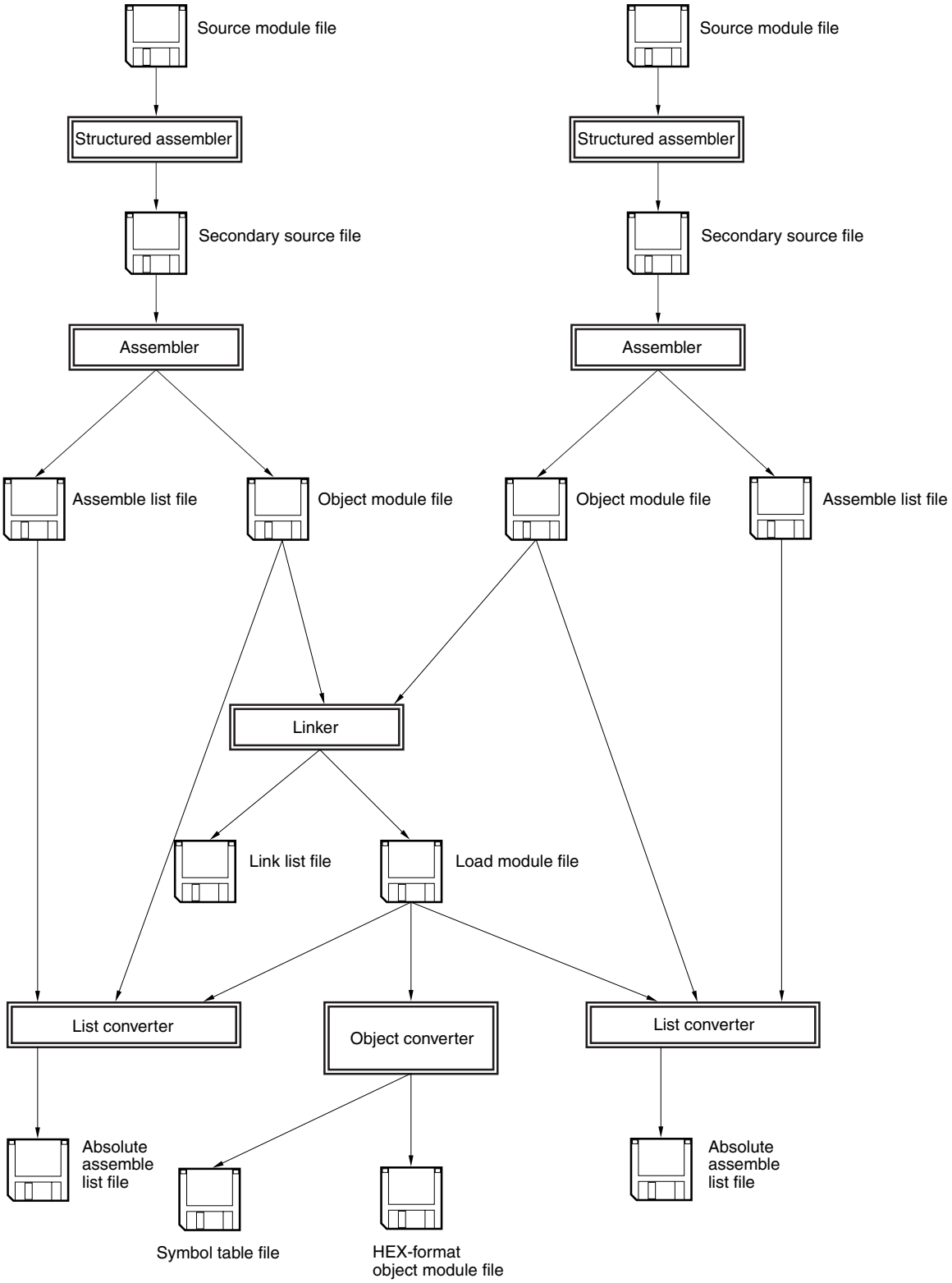
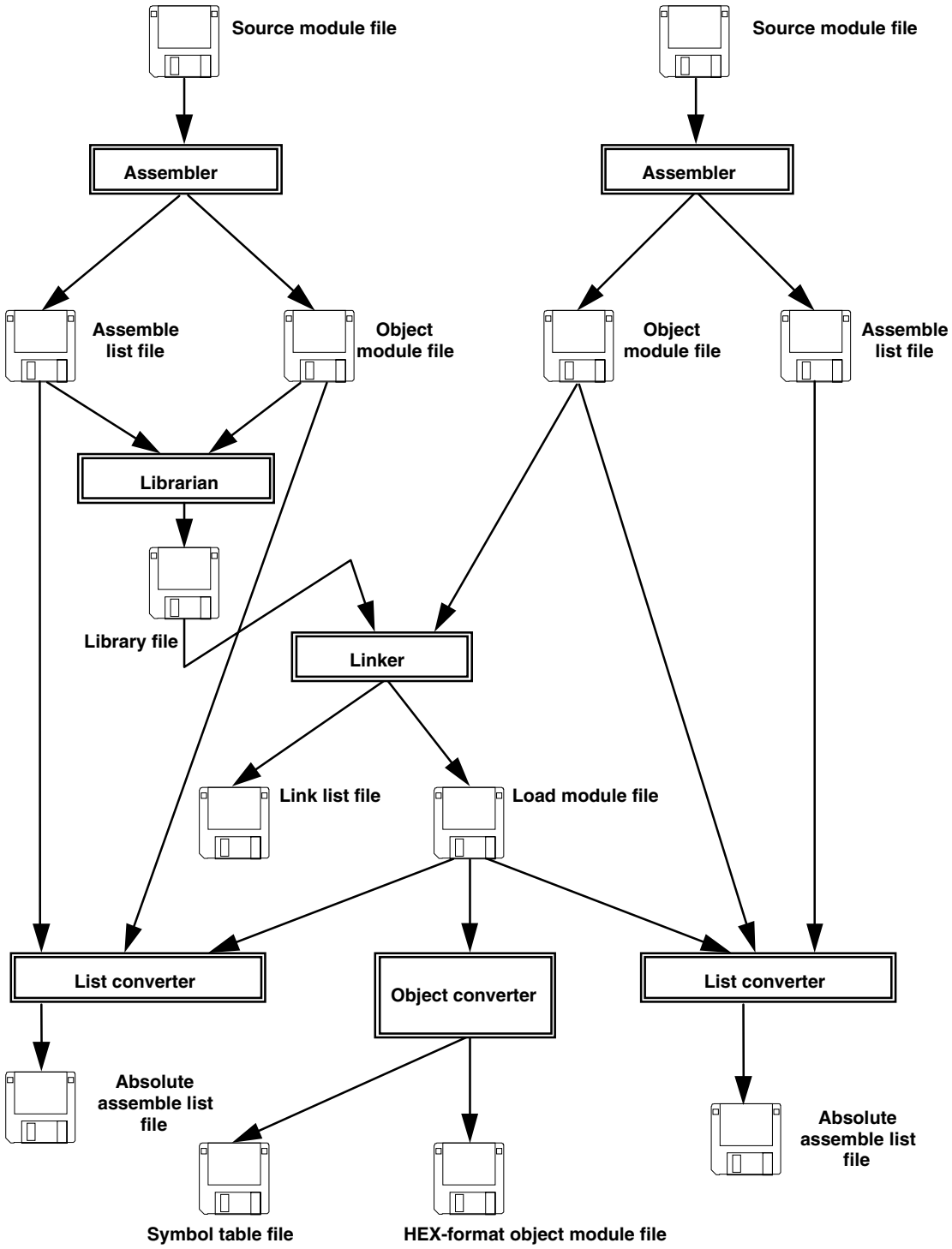


Figure 3-3. RA78K0S Execution Procedure 2



3.3 Execution Procedure of ST78K0S

To verify the operation of the ST78K0S, 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 (the device file is obtained separately).

The following explanation uses the μPD789024 as the target device.

■st.bat (ST78K0S-operation verification batch program)

```
echo off
cls
set  LEVEL=0

if "%1" == "" goto ERR_BAT

st78k0s -C%1 test1.s
ra78k0s test1.asm
if errorlevel 1 set LEVEL=1
st78k0s -C%1 test2.s
ra78k0s test2.asm
if errorlevel 1 set LEVEL=1
if %LEVEL% == 1 echo Assemble error !!
if %LEVEL% == 1 goto END

cls
lk78k0s test1.rel test2.rel -s -otest.lmf -ptest.map
if errorlevel 1 echo Link error !!
if errorlevel 1 goto END

cls
oc78k0s stsample
if errorlevel 1 echo Object conversion error !!
if errorlevel 1 goto END

cls
set LEVEL=0
lc78k0s -1stsample.lmf -rtest1.rel test1.prn
if errorlevel 1 set LEVEL=1
lc78k0s -ltest.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 ST78K0S-operation verification batch program.

```
C>st.bat 9024
```

The following message will be output.

```
Structured assembler preprocessor for RA78K/0S Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

start

 Target chip:uPD789024
 Device file:Vx.xx

Conversion complete, 0 error (s) found.

78K/0S Series Assembler Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation,xxxx,xxxx

PASS_PARSE Start
PASS_OUTOBJ Start

 Target chip:uPD789024
 Device file:Vx.xx

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

Structured assembler preprocessor for RA78K/0S Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

start

 Target chip:uPD789024
 Device file:Vx.xx

Conversion complete, 0 error (s) found.

78K/0S Series Assembler Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

PASS_PARSE Start
PASS_OUTOBJ Start

 Target chip:uPD789024
 Device file:Vx.xx

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

Clear the screen.

```
78K/0S Series Linker Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

 Target chip:uPD789024
 Device file:Vx.xx

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

Clear the screen.

```
78K/0S Series Object Converter Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

 Target chip:uPD789024
 Device file:Vx.xx

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

Clear the screen.

```
List Conversion Program for RA78K/0S Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

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

List Conversion Program for RA78K/0S Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

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

Clear the screen.

```
No errors.
```

(2) Check the contents of drive C.

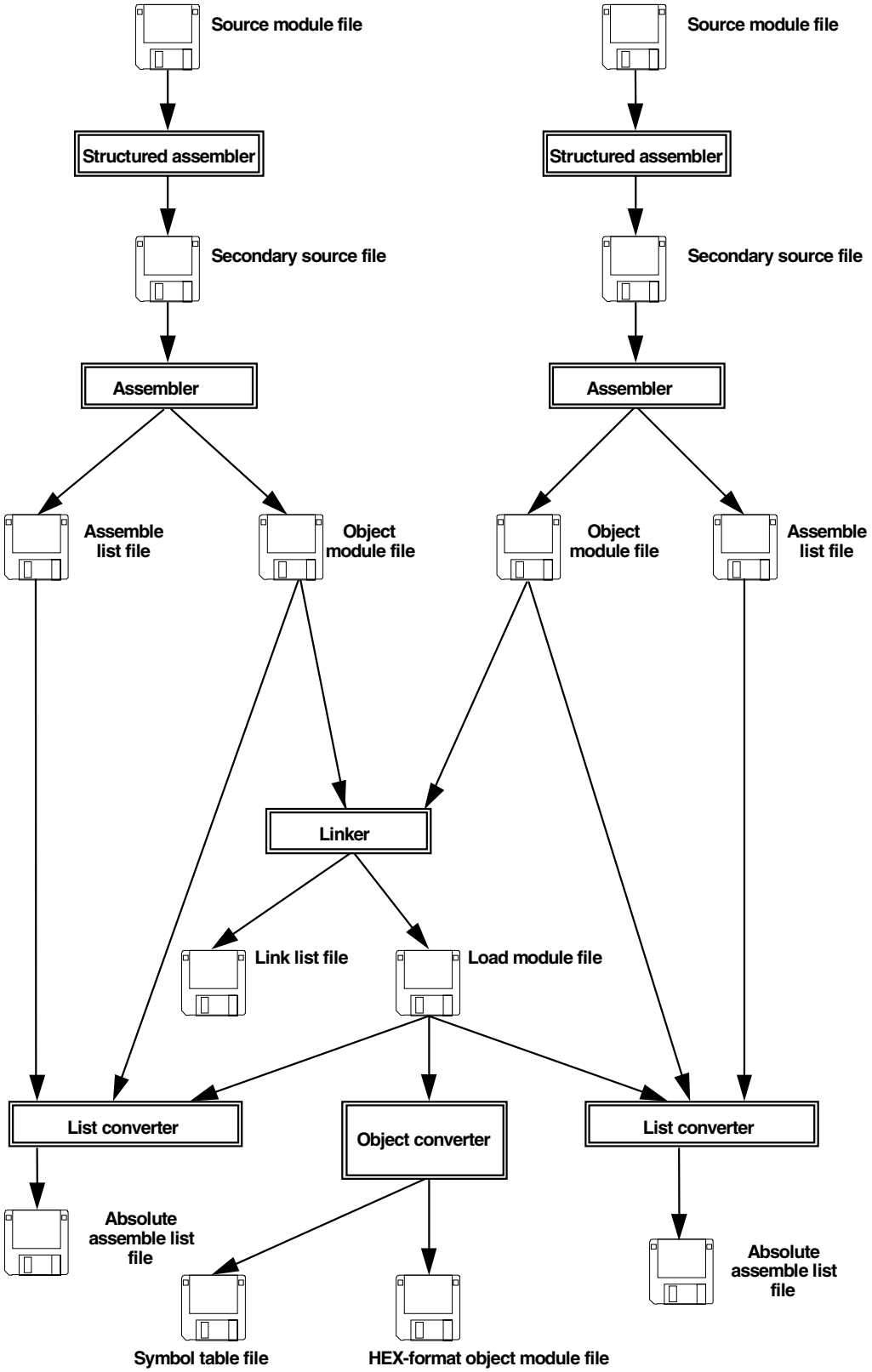
The following files will be output.

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 ST78K0S execution procedure

The following is a summary of the execution procedure of the ST78K0S.

Figure 3-4. ST78K0S Execution Procedure 1



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 K0SMAIN.ASM.

Input the following on the command line.

```
C>ra78k0s -c9176 k0smain.asm
```

The following message is output to the display.

```
78K/0S Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD789176
 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 (K0SMAIN.REL) and the assemble list file (K0SMAIN.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 K0SSUB.ASM. Input the following on the command line.

```
C>ra78k0s -c9176 k0ssub.asm
```

The following message is output to the display.

```
78K/0S Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD789176
 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 (K0SSUB.REL) and the assemble list file (K0SSUB.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 that 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.

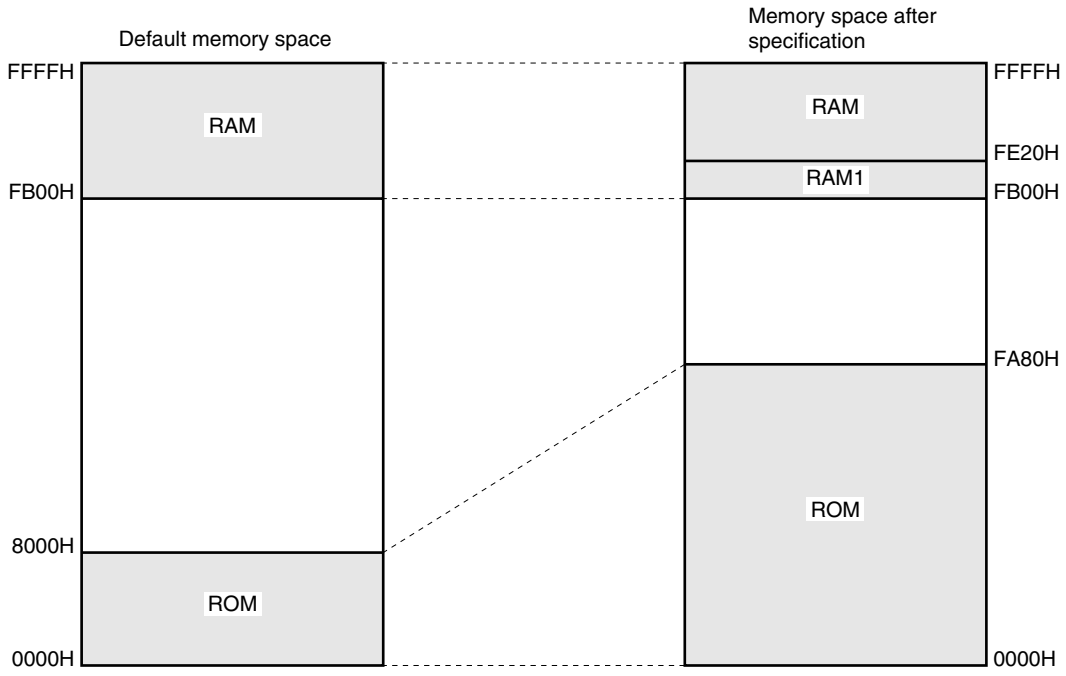
You will also need to create a directive file when you wish to locate segments not defined as absolute segments within a source module file to a specific address in memory.

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

Example To extend the ROM area (0H to 7FFFH) to (0H to FA7FH), and extend the RAM area to (FE20H to FFFFH) and RAM1 (FB00H to FE1FH), write the following to the directive file.

```
MEMORY ROM: (0H, 0FA80H)
MEMORY RAM1: (0FB00H, 320H)
MEMORY RAM: (0FE20H, 1E0H)
```

Figure 3-5. Link Directive



(6) As the result of the assembly, the output object module files [K0SMAIN.REL] and [K0SSUB.REL] are linked.

Enter K0S.DR as the directive file.

Enter the following on the command line.

```
C>lk78k0s k0main.rel k0ssub.rel -dk0s.dr[Note 1] -ok0s.lmf -pk0s.map -S[Note 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/0S Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

 Target chip : uPD789176
 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 (K0S.lmf) and the link list file (K0S.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 (K0S.lmf) is converted to a HEX-format file.

Enter the following on the command line.

```
C>oc78k0s k0s.lmf -r -u0FFH
```

The following message is output on the display.

```
78K/0S Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

 Target chip : uPD789176
 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 (K0S.HEX) and the symbol table file (K0S.SYM).

(10) Create a library file as follows.

Register the object module file (K0SSUB.REL) output by the assembler as a library file.
Enter the following on the command line.

```
C>lb78k0s < k0s.job
```

The following message is output on the display.

```
78K/0S Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

*create k0s.lib
*add k0s.lib k0sub.rel
*exit
```

Remark The parts underlined above are the contents of “k0s.job.”

(11) Check the contents of drive C.

The librarian outputs the library file (K0S.LIB).

(12) Create an absolute assemble list as follows.

To create the absolute assemble list K0SMAIN.ASM, input [K0SMAIN.REL], [K0SMAIN.ASM] and [K0S.lmf] to the list converter.
Enter the following on the command line.

```
C>lc78k0s k0smain -lk0s.lmf
```

The following message is output on the display.

```
List Conversion Program for RA78K0S 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 (K0SMAIN.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 PM plus option setting.

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

```
>[path-name] RA78K0SΔ-F parameter-file-name
>[path-name] LK78K0SΔ-F parameter-file-name
>[path-name] OC78K0SΔ-F parameter-file-name
```

Here is an example of its use.

Example
```
C>ra78k0S -Fpara.pra
C>lk78k0S -Fpara.plk
C>oc78k0S -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)

```
-c9024 k0smain.asm -e
```

(Contents of para.plk)

```
k0smain.rel k0ssub.rel -bmylib.lib -osample.Imf -s
```

(Contents of para.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 78K0S Series microcontrollers, converts them into assembly language, and outputs them as secondary source module files.

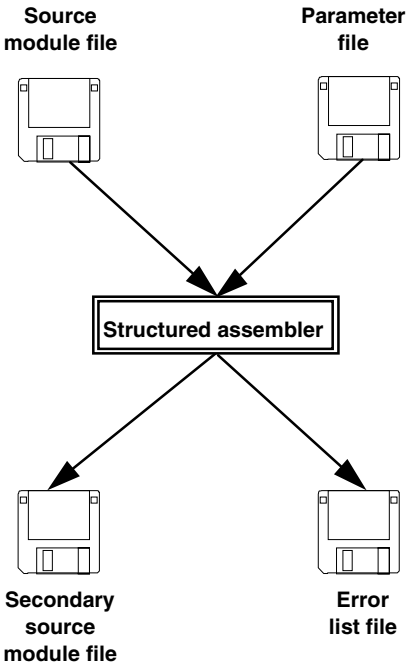
4.1 I/O Files of Structured Assembler

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 files	• These are files containing structured assembler error data.	.EST

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



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

The following two methods can be used to start up the structured assembler.

(1) Startup from the command line

Start up the structured assembler by inputting the following command.

```
X> [path-name] st78k0s [Δoption] ... Δsource-module-file-name [Δoption]...
|       |         |         |                    |              |
(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>st78k0s –c9024 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.

Start up the structured assembler from a parameter file as follows.

```
X>[path-name] st78k0s [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 module 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 Create the parameter file (test1.pst) using an editor.

- Contents of test1.pst

```
;Parameterfile
test1.s –osample.asm
-esample.est –c9024
```

- Use parameter file (test1.pst) to start up the structured assembler.

```
C>st78k0s –ftest1.pst
```

4.3.2 Execution start and end messages

(1) Execution start message

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

```
Structured assembler preprocessor for RA78K/0S 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 operating system.

```
 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 operating system.

```
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 unable to continue structured assembly processing, the structured assembler outputs a message to the display, cancels the structured assembly processing, and returns control to the operating system.

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

```
C>st78k0s sample.s

Structured assembler preprocessor for RA78K/0S Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 1996

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>st78k0s test1.s -z

Structured assembler preprocessor for RA78K/0S Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

A012 Missing parameter '-z'
Please enter 'ST78K0S --', 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 12 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 assembler.

[Explanation]

1) Always specify option -C. The structured assembler performs preprocessing on the specified target device and generates the assembler source code.
   Note that if option -C is omitted, an error will occur.
2) If the device types specified by option -C 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 option -C.
3) The device type specified by option -C 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 μPD789024 is specified as the target device.

```
C>st78k0s test.s -c9024
```

[Notice]

Option -C 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 option -SC 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>st78k0s 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>st78k0s 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 until the converted assembly language is output.

[Explanation]

1) Option -WT 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 comment 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>st78k0s 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 option -I 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 b:\include.

```
C>st78k0s test.s -ib:\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 option -O 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>st78k0s 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 option -E 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>st78k0s 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>st78k0s -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) Option -GS specifies the output of debug data to the secondary source file.
2) Option -NGS makes option -GS unavailable.
3) If there is compiler debug data in the input source file, option -GS replaces “$” with “;” at the top of the file.
4) If both options -GS and -NGS are specified, the option specified last takes precedence.
5) If omitted, option -GS 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 not 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>st78k0s 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>st78k0s 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: $KANJICODE SJIS
   -ZE: $KANJICODE EUC
   -ZN: $KANJI CODE NOTE
3) The priority order of specifications of the kanji code is as follows.
   (1) Specification of -ZS/-ZE/-ZN option
   (2) Specification of the kanji code specification control instruction ($KANJICODE)
   (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>st78k0s test.s -zs
```

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

Syntax: -Y [drive:] directory

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

1) <..\dev> (for the path that started up st78k0s.exe)
2) The path that started up st78k0s.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 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 started up st78k0s.exe)
   3. The path that started up st78k0s.exe
   4. The current path
   5. The path that was specified by the environment variable PATH

[Example of use]

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

```
C>st78k0s test.s -ya:\nectools\dev
```

(13) Help specification (--)

Syntax: --

Default assumption: No display

[Function]

Option -- displays a help message.

[Application]

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

[Explanation]

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

Caution This option cannot be specified on PM plus.
To reference PM plus help, click the help button in the <Structured Assembler Options Setup> dialog box.

[Example of use]

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

```
C>st78k0s --

Structured assembler preprocessor for RA78K/0S Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

Usage:st78k0s [option[...]] input-file [option[...]]

The option is as follows ([]means omissible, ...means repetition).
 -cx            :Select target chip. (x = 9002, 9014, 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.
 -zs/-ze/-zn    :Change source regulation.
           -zs:SJIS code usable in comment.
           -ze:EUC code usable in comment.
Press RETURN to continue...
           -zn:no multibyte code in comment.
 -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 option 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)

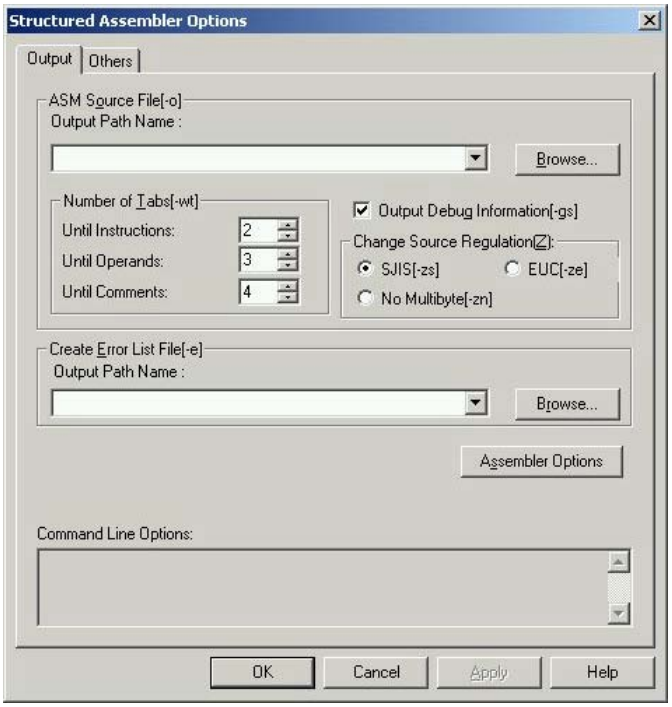


Figure 4-3. < Assembler Source Options> Dialog Box

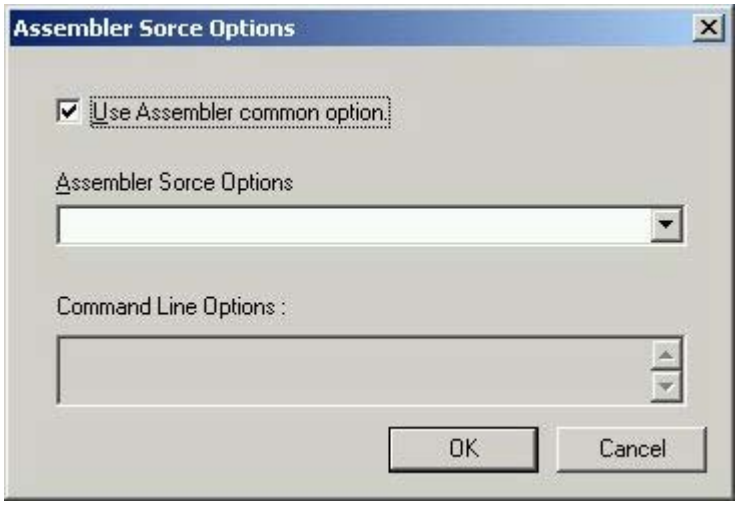
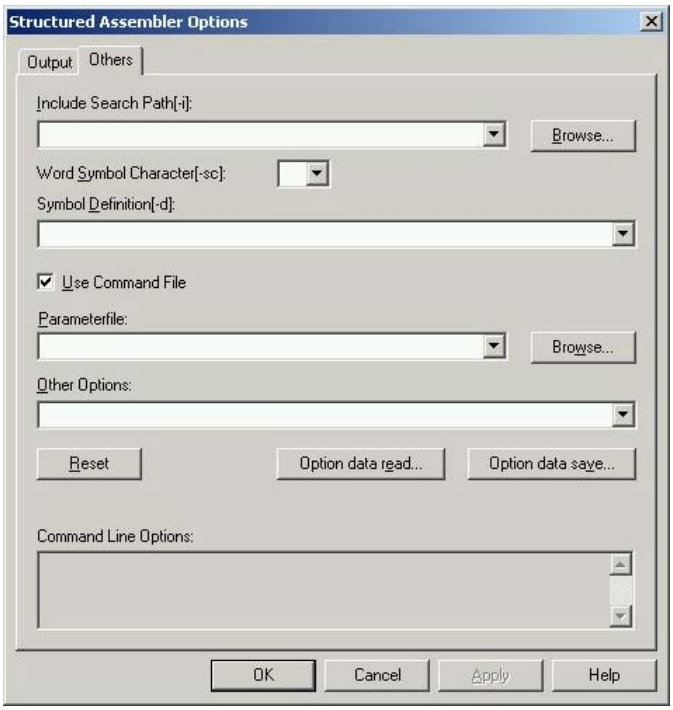


Figure 4-4. <Structured Assembler Options> Dialog Box (When <<Others>> Tab Is Selected)



4.5.2 Options

The following is an explanation of each option in the <Structured Assembler Options Setup> 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 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 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 78K0S 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.

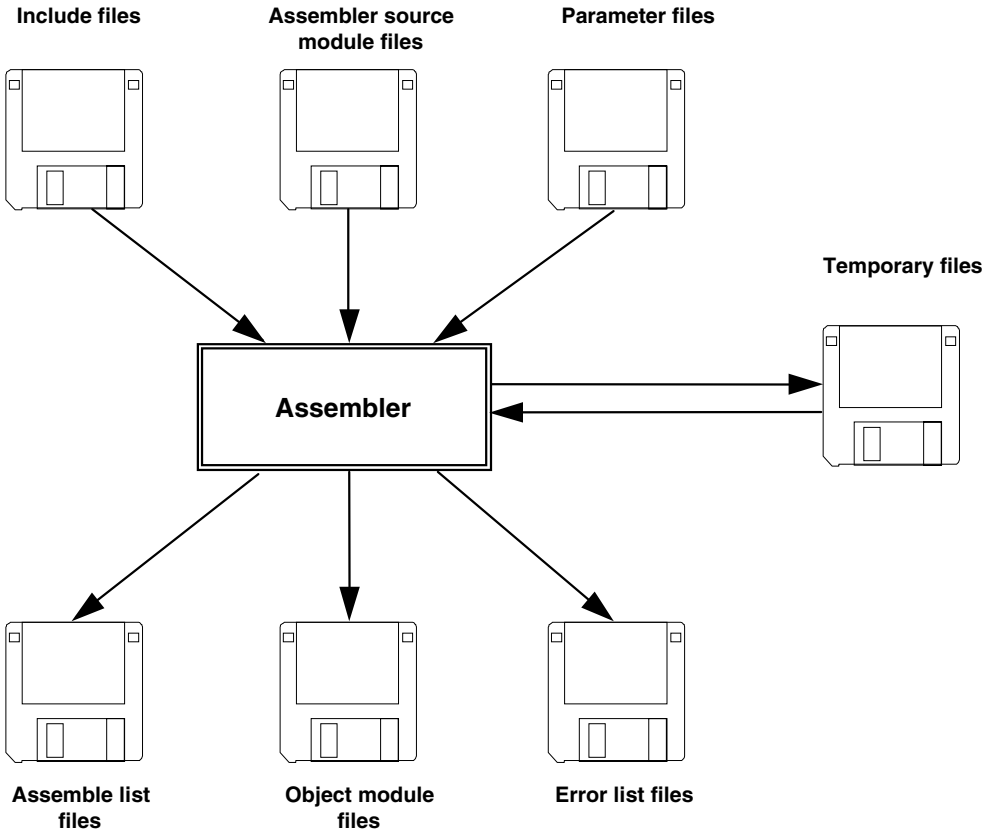
5.1 I/O Files of Assembler

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	• These are source module files written in assembly language for 78K0S Series microcontrollers. • These files are created by the user.	.ASM
	Include files	• These files are used for reference with assembler source module files. • These are files written in assembly language for 78K0S Series microcontrollers. • These files are created by the user.	None
	Parameter files	• These files contain the parameters for the executed files. • These files are created by the user.	.PRA
Output files	Object module files	• These are binary files including relocation data and symbol data regarding machine language data and machine language location addresses.	.REL
	Assemble list files	• These are files containing assembly data such as assemble lists and cross-reference lists.	.PRN
	Error list files	• These are files containing error data generated during assembly.	.ERA
I/O files	Temporary files	• These are files created automatically by the assembler for assembly purposes. Temporary files are deleted when assembly ends.	RAxxxxx.$$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

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

(1) Startup from the command line

```
X > [path-name] RA78K0S [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 an individual option 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>ra78k0s –c9024 k0smain.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 the same assembler option is specified repeatedly each time assembly is performed.

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

Start up the assembler from a parameter file as follows.

```
X > RA78K0S [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 (specification 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 (K0SMAIN.PRA) using an editor.

- Contents of K0SMAIN.PRA

```
;parameter file
k0smain.asm -osample.rel
-psample.prn
```

- Use parameter file (K0SMAIN.PRA) to start up the assembler.

```
C>ra78k0s -fk0smain.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/0S 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
K0SMAIN.ASM(12) :F201 Syntax error
Pass2 Start
K0SMAIN.ASM(12) :F201 Syntax error
K0SMAIN.ASM(29) :F407 Undefined symbol reference 'CONVAH'
K0SMAIN.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>ra78k0s sample.asm

78K/0S 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>ra78k0s k0smain.asm -z

78K/0S Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

A012 Missing parameter '-z'
Please enter 'RA78K0S--', 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 12 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 15 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	

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

Number	Classification	Option	Explanation
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.
11	Specification of path for temporary file creation	-T	Creates a temporary file in a specified path.
12	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.
13	Device file search path specification	-Y	Reads a device file from a specified path.
14	Symbol definition specification	-D	Defines a symbol.
15	Help specification	--	Displays a help message on the display.

5.4.2 Order of precedence of assembler options

Table 5-3 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	--
-J	×						×
-G	×						×
-P			Δ	Δ		Δ	×
-KA		×					×
-KS		×			×		×
-KX		×					×
-LW		×					×
-LL		×					×
-LH		×					×
-LT		×					×
-LF		×					×

← Horizontal axis

↑ 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>ra78k0s –c9024 k0smain.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>ra78k0s –c9024 k0smain.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>ra78k0s –c9024 k0smain.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 Order of Precedence of Assembler Options** 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]

Always specify option -C. 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.

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

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

[Example of use]

Specify option -C on the command line as follows.

```
C>ra78k0s -c9024 k0smain.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>ra78k0s -c9024 k0smain.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>ra78k0s –c9024 k0smain.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]

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.

```
∇$∇DEBUG
∇$∇DG           ; abbreviated form
∇$∇NODEBUG
∇$∇NODG         ; abbreviated form
```

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

[Example of use]

Specify addition of debug data to an object module file.

```
C>ra78k0s –c9024 k0smain.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.

```
∇$∇DEBUGA
∇$∇NODEBUGA
```

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

[Example of use]

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

```
C>ra78k0s –c9024 k0smain.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 (INC78K0S)

[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 B:\SAMPLE.

```
C>ra78k0s –c9024 k0smain.asm -ib:\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>ra78k0s –c9024 k0smain.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 specifies to output 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.

```
C>ra78k0s –c9024 k0smain.asm -ka
```

Reference K0SMAIN.PRN.

```
   Assemble list

 ALNO  STNO ADRS OBJECT M I SOURCE STATEMENT

    1     1                          NAME    SAMPM
    2     2                  ;*************************************
    3     3                  ;
    4     4                  ;   HEX -> ASCII Conversion Program
    5     5                  ;
    6     6                  ;        main-routine
    7     7                  ;
    8     8                  ;*************************************
    9     9
   10    10                          PUBLIC MAIN, START
   11    11                          EXTRN  CONVAH
   12    12                          EXTRN  _@STBEG
   13    13
   14    14  ----            DATA    DSEG  saddr
   15    15  0000            HDTSA: DS 1
   16    16  0001            STASC: DS 2
   17    17
   18    18  ----            CODE    CSEG AT 0H
   19    19  0000 R0000      MAIN:  DW START
   20    20
   21    21  ----                    CSEG
   22    22  0000            START:
   23    23
   24    24                          ;chip initialize
   25    25  0000 RF00000             MOVW   AX, #_@STBEG
   26    26  0003 E61C                MOVW   SP, AX
   27    27
   28    28  0005 RF5001A             MOV    HDTSA, #1AH
   29    29  0008 RFC0000             MOVW   HL, #HDTSA      ;set hex 2-code data
in HL register
   30    30
   31    31  000B R220000             CALL   !CONVAH         ;convert ASCII<- HE
X
   32    32                                                 ;output BC-register
<- ASCII code
   33    33  000E RF80100             MOVW   DE, #STASC      ;set DE <- store ASC
II code table
   34    34  0011 0A27                MOV    A, B
   35    35  0013 EB                  MOV    [DE], A
   36    36  0014 88                  INCW   DE
   37    37  0015 0A25                MOV    A, C
   38    38  0017 EB                  MOV    [DE], A
   39    39
   40    40  0018 30FE                BR     $$
   41    41
   42    42                          END
```

(b) -KS/-NKS

Syntax: -KS
: -NKS
Default assumption: -NKS

[Function]

1) Option -KS specifies to output 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>ra78k0s –c9024 k0smain.asm -ks
```

Reference K0SMAIN.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         HDTSA                           0H      ADDR   PUB   MAIN
          MOD          SAMPM                           0H      ADDR   PUB   START
FE21H     ADDR         STASC
```

(c) -KX/-NKX

Syntax: -KX
: -NKX
Default assumption: -NKX

[Function]

1) Option -KX specifies to output 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.

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

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

[Example of use]

Output a cross-reference list.

```
C>ra78k0s –c9024 k0smain.asm -kx
```

Reference K0SMAIN.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             FE20H       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.
(up to 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]

The control instruction which is equivalent to the function of the -LW option can be described at the first of the source module.
The synopsis is as follows:

```
∇$∇WIDTH
```

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

[Example of use]

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

```
C>ra78k0s -c9024 k0smain.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                            EXTRN   _@STBEG
  14    14 ----                    DATA    DSEG    AT 0FE20H
  15    15 FE20                    HDTSA:  DS      1
  16    16 FE21                    STASC:  DS      2
  17    17
  18    18 ----                    CODE    CSEG    AT 0H
  19    19 0000 R0000              MAIN:   DW      START
  20    20
  21    21 ----                            CSEG
  22    22 0000                    START:
  23    23
  24    24
  25    25
        :
        :
```

(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 ≤ number of lines printed on 1 page ≤ 32767

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

[Notice]

The control instruction which is equivalent to the function of the -LL option can be described at the first of the source module.
The synopsis is as follows:

```
∇$∇LENGTH
```

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

[Example of use]

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

```
C>ra78k0s –c9024 k0smain.asm -ll20
```

This references K0SMAIN.PRN.

```
78K/0S Series Assembler Vx.xx                                   Date:xx xxx xxxx Page:   1

Command:-c9024 k0smain.asm -ll20
Para-file:
In-file:K0SMAIN.ASM
Obj-file:K0SMAIN.REL
Prn-file:K0SMAIN.PRN

      Assemble list

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



78K/0S Series Assembler Vx.xx                                   Date:xx xxx xxxx 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/0S Series Assembler Vx.xx                                   Date:xx xxx xxxx 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.
   1 Japanese kanji or hiragana character is counted as 2 characters.
   If the maximum number of characters per line is 119 or less, the length of the effective character string changes as follows.

   Effective length = (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 the -LH option 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 command 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 a source module.

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

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

[Example of use]

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

```
C>ra78k0s –c9024 k0smain.asm -lhRA78K0S_MAINROUTINE
```

This references K0SMAIN.PRN.

```
78K/0S Series Assembler Vx.xx  RA78K0S_MAINROUTINE      Date:xx xxx xxxx Page:   1
                                        └──Title


Command:  -c9024 k0smain.asm -lhRA78K0S_MAINROUTINE
Para-file:
In-file:K0SMAIN.ASM
Obj-file:K0SMAIN.REL
Prn-file:K0SMAIN.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                   *
          .
          .
          .
```

(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 ≤ number of characters that can be specified ≤ 8

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

[Notice]

The control instruction which is equivalent to the function of the -LT option can be described at the first of the source module.
The synopsis is as follows:

∇$∇TAB∇ tab-count

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

[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    0A27                 MOV       A, B
   5     5 0002    0A12                 SET1      A, 1
   6     6                              END
```

[Example 2]

1 blank is specified using the HT code.

```
  C>ra78k0s –c9024 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    0A27           MOV A, B
   5     5 0002    0A12           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 -LF option unavailable.

[Application]

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

[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]

The control instruction which is equivalent to the function of the -LF/-NLF option can be described at the first of the source module.
The synopsis is as follows:

```
∇$∇FORMFEED
∇$∇NOFOMFEED
```

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

[Example of use]

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

```
C>ra78k0s –c9024 k0smain.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 -NE option makes the -E option unavailable.

[Application]

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

[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 drive.
4) If both options -E and -NE are specified at the same time, the option specified last takes precedence.

[Example of use]

An error list file (sample.era) is created.

```
C>ra78k0s -c9024 k0smain.asm -esample.era

78K/0S Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

PASS_PARSE Start
K0SMAIN.ASM(26) :F202 Illegal operand
PASS_OUTOBJ Start
K0SMAIN.ASM(32) :F407 Undefined symbol reference 'F'
K0SMAIN.ASM(41) :F407 Undefined symbol reference 'F'

 Target chip:uPD78xxxx
 Device file:Vx.xx

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

The error list file (sample.era) is referenced.

```
PASS_PARSE Start
K0SMAIN.ASM(26) :F202 Illegal operand
PASS_OUTOBJ Start
K0SMAIN.ASM(32) :F407 Undefined symbol reference 'F'
K0SMAIN.ASM(41) :F407 Undefined symbol reference 'F'
Pass1 Start
```

(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 (K0SMAIN.PRA) as follows.

```
;parameter file
k0smain.asm -osample.rel –g –c9024
-psample.prn
```

Enter the following on the command line.

```
C>ra78k0s -fk0smain.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 A:\TMP.

```
C>ra78k0s -c9024 k0smain.asm -ta:\tmp
```

(12) 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.

```
▲$▲KANJICODE△SJIS
▲$▲KANJICODE△EUC
▲$▲KANJICODE△NONE
```

   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 **11.2 Preparing Development Environment (Environmental Variables).**

[Example of use]

The kanji code is interpreted as EUC code

```
A>ra78k0s k0smain.asm –c9024 -ze
```

(13) 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 ra78k0s.exe startup path)
2) Path by which RA78K0S 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 ra78k0s.exe startup path)
   c. Path by which RA78K0S was started up
   d. Current directory
   e. The environmental variable PATH

[Example of use]

Specify the path for the device file as directory a:\78k0s\dev.

```
C>ra78k0s k0smain.asm –c9024 -ya:\78k0s\dev
```

(14) Symbol definition specification (-D)

Syntax: -D symbol-name [=numerical-value][, symbol-name[=numerical-value]…]
Default assumption: None

[Function]

Defines a symbol.

[Application]

Specify option -D when wanting to define a symbol.

[Explanation]

1) The numerical values assigned to symbols must be binary, octal, decimal, or hexadecimal. If specification of the numerical value is omitted, is it assumed that 1 was specified.
2) Up to 30 symbols can be specified, delimited with commas.
3) Up to 30 characters can be described for the symbol name.
4) If the same name is specified more than once, the name specified last is valid.
5) The letters in the symbol name 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]

Specify 2 as the symbol definition.

```
C>ra78k0s k0smain.asm –c9024 –dSYM=2
```

(15) 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 assembler 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.

```
78K/0S Series Assembler Vx.xx  [xx xx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx,xxxx

usage : ra78k0s [option[...]] input-file [option[...]]
The option is as follows ([] means omissible).
-cx          :Select target chip. ( x = 9012,p9014 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.
-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 [Assembler Options...] from the [Tools] 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)

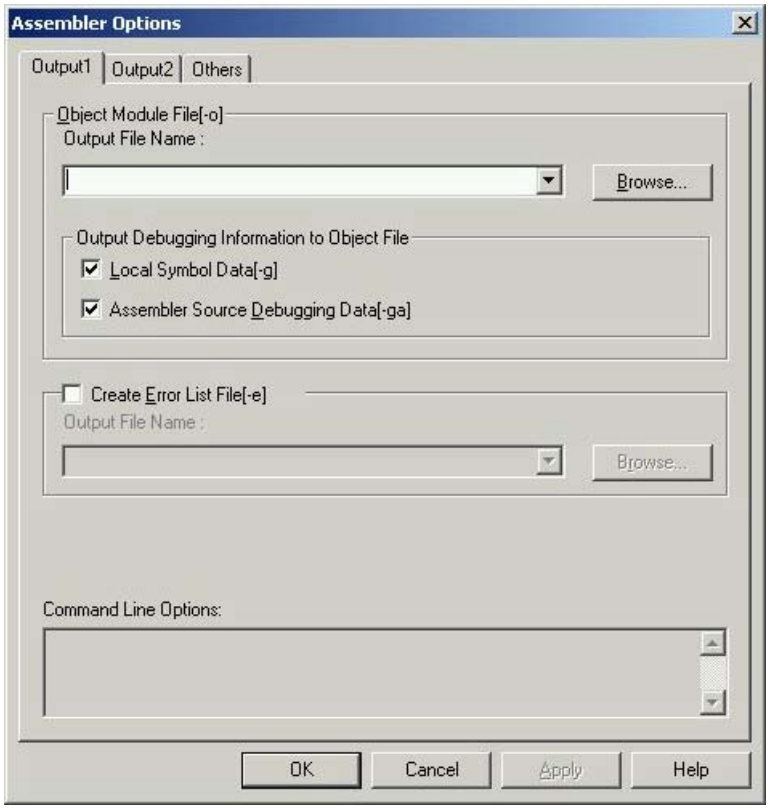


Figure 5-3. <Assembler Options> Dialog Box (When <<Output2>> Tab Is Selected)

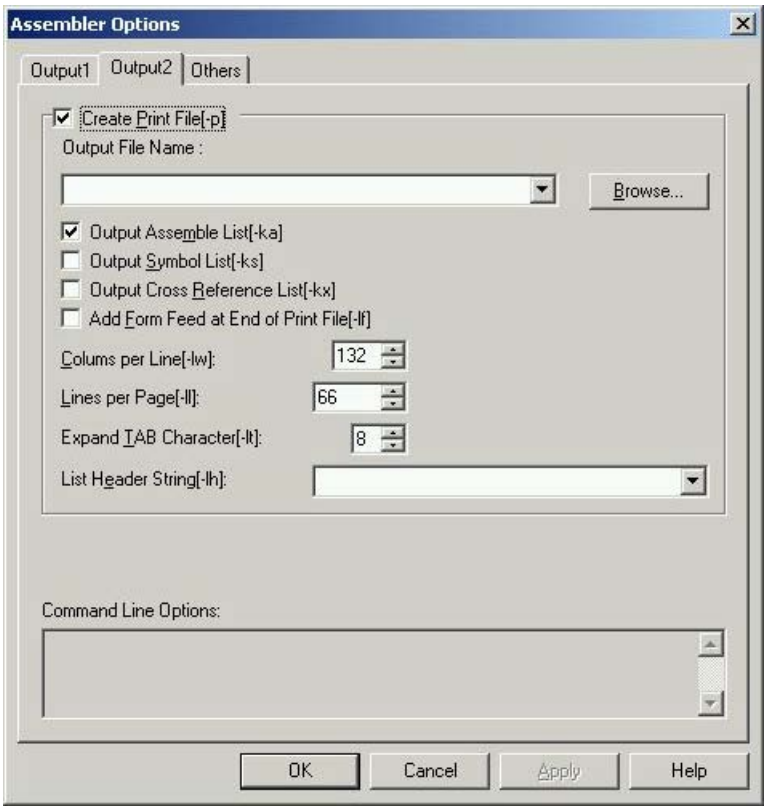
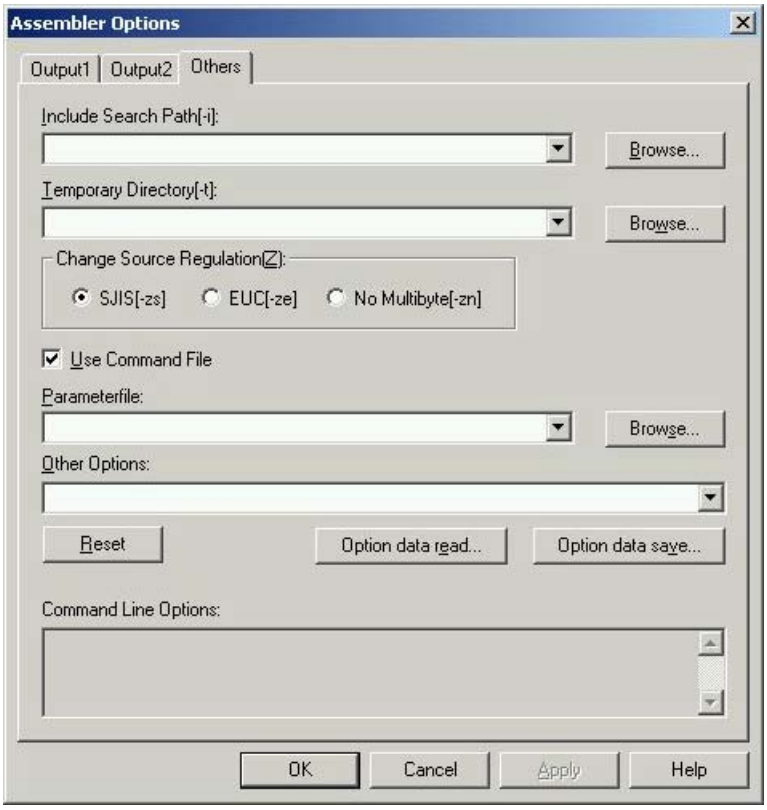


Figure 5-4. <Assembler Options> Dialog Box (When <<Others>> Tab Is Selected)



5.5.2 Option settings

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

- Object Module File [-o]
  Output File Name:
  Specify the object module file output path either using the [Browse...] button or by directly inputting it.
- Local Symbol Data [-g]
  Perform symbolic debugging including local symbols.
- Assembler Source Debugging Data [-ga]
  Add the assembler source debug data.
- 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.
- Create Print File [-p]
  Output File Name:
  Specify the assemble list file output path either using the [Browse...] button or by directly inputting it.
- Output Assemble List [-ka]
  Output the assemble list in the assemble list file.
- Output Symbol List [-ks]
  Output symbol list after assemble list.
- Output Cross Reference List [-kx]
  Output cross-reference list after assemble list.
- Add Form Feed at End of Print File [-lf]
  Append form feed code after printing contents of assemble list file.
- Columns per Line [-lw]
  Specify the number of characters per line in assemble list file (selectable from 72 to 2,046 characters).
- Lines per page [-ll]
  Specify the number of lines per page in assemble list file (selectable from 20 to 32,767 characters).
- Expand TAB character [-lt]
  Specify the tab character length (selectable from 0 to 8 characters)
- List Header String [-lh]
  Specify the character string to be printed in the title box of the assemble list file header (up to 60 characters).
- Include Search Path [-i]
  Specify the path for reading include file either using the [Browse...] button or by directly inputting it.
- Temporary Directory [-t]
  Specify the location where the temporary file is to be created, either using the [Browse...] button or by directly inputting it.
- Change Source Regulation [Z]
  Select the multibyte code type (SJIS[-zs], EUC[-ze], No Multibyte[-zn]) to be used in source comments.
- 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 6 LINKER

The linker inputs a number of object module files output by the 78K0S Series 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 shown below.

Table 6-1. I/O Files of Linker

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

Caution Spaces other than the /REGULAR space cannot be specified because only one memory space is available.

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 **Figure 3-5 Contents of Link Directive File (sample.dr).**

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 area
Internal RAM other than SADDR area

(3) Segment location 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	• Declares an address in installed memory • Divides memory into two or more areas and specifies a memory area
2	Segment location directive	• Specifies location of a segment

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][Note]

Note The memory space name does not need to be described actually because the default is regarded as /REGULAR.

2) Segment location directives

MERGE segment name: [AT∆(∆start address∆)]

[=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

In cases when there are two or more segments with the same name in the source file, specifying “COMPLETE” in the directive will ensure that an error occurs and the segments are not integrated, whereas specifying “SEQUENT (default)” will ensure that the segments are integrated.

SEQUENT: Segments are merged without a space between in the order of appearance. BSEG is merged in 1-bit units in order of appearance.

COMPLETE: When there are two or more segments with the same name, an error occurs.

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 to 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 789024
MEMORY MEM1: (01000H, 1000H) #SECOND MEMORY AREA
```

6.4.2 Memory directives

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

[Syntax]

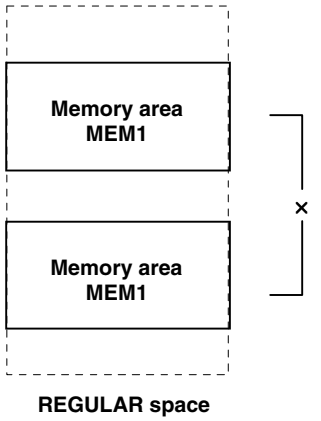
MEMORY∆ memory-area-name∇:∇ (∇start-address-value∇, ∇size∇) [/∇memory-space-name]

(1) Memory area names

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

1) The characters which can be used to describe a memory area name are A to Z, a to z, 0 to 9, _, ?, and @. However, a memory area name cannot begin with 0 to 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 FFFFH.

(3) Size

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

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

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

(4) Memory space name

The memory space name is displayed in the 64 KB space REGULAR.

Conditions on specification are as follows.

1) Memory space name must be specified 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 area, specify the size of the default area name as "0".

[Example of Use]

Define the addresses 0H to 1FFH in the memory space as ROMA.

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

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

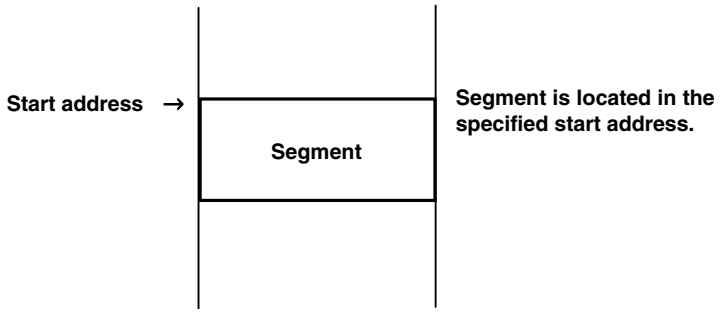
The segment name is the name of a segment included in an object module file input to the linker.

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 assemble 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 specification of a segment definition 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) Only REGULAR can be specified as a memory area name.
2) Memory space names must be specified 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
Memory area name	×	Specified memory area in the REGULAR 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.

[Notice]

1) The location address of an input segment for which no segment location directive is specified will be determined according to the relocation attributes 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 attributes 'CSEG UNIT'. In this example the declared memory area is as follows.

```
MEMORY ROM:  (0000H, 1000H)
MEMORY MEM1: (1000H, 2000H)
MEMORY RAM:  (0FE00H, 200H)
```

(1) When input segment SEG1 is allocated to 500H in ROM area.

```
MERGE SEG1: AT (500H)
```

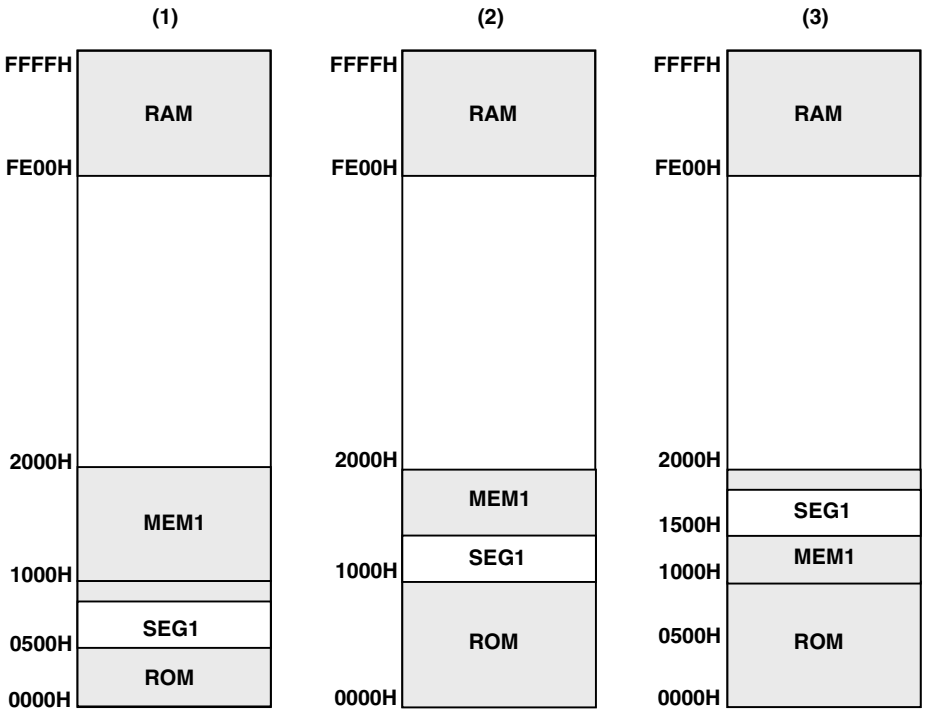
(2) When input segment SEG1 is allocated to memory area MEM1.

```
MERGE SEG1:= MEM1
```

(3) When input segment SEG1 is allocated to 1500H in memory area MEM1.

```
MERGE SEG1: AT (1500H) = MEM1
```

Figure 6-2. Specific Examples of Segment Allocation



6.5 Linker Startup

6.5.1 Linker startup

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

(1) Startup from the command line

```
X>[path-name] lk78k0s [Δoption] ··· Δobject-module-file-name [Δoption]···[Δ]
 |       |          |         |                  |                |
(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>lk78k0s k0smain.rel k0ssub.rel –ok0s.lmf -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 the same linker option is specified repeatedly each time link is performed.

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>LK78K0S [Δobject-module-file] Δ-f parameter-file-name
                                  |            |
                                 (1)          (2)
```

(1) Parameter file specification option

(2) A file which includes the data required to start up the linker

Remark An editor is used to create the parameter file.

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

```
[[[Δ] option [Δoption] ··· [Δ]Δ]] ···
```

1) If the object module file name is omitted from the command line, specify the object module file name in the parameter file.
2) The object module file name can also be 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 (K0S.PLK) using an editor.

Contents of the parameter file K0S.PLK:

```
;parameter file
k0smain.rel k0ssub.rel –ok0s.lmf –pk0s.map -e
-ta:\tmp
```

Use parameter file K0S.PLK to start up the linker.

```
C>lk78k0s –fk0s.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/0S 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:uPD78xxx
 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:uPD78xxx
 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>lk78k0s samp1.rel samp2.rel

78K/0S Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx

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>lk78k0s k0smain.rel k0ssub.rel -z

78K/0S Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx

A018 Option is not recognized '-z'
Please enter 'LK78K0S --', 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 12 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 16 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 decision 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	Help specification	--	Displays a help message on the display.

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

6.6.2 Order of precedence of linker options

Table 6-6 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	×						×
-G	×						×
-P				Δ	Δ	Δ	×
-KM			×				×
-KD			×	×			×
-KP		×	×				×
-KL		×	×				×
-LL			×				×
-LF			×				×

← Horizontal axis

↑ 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>lk78k0s k0smain.rel k0ssub.rel -np –km -s`

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>lk78k0s k0smain.rel k0ssub.rel -p -nkm -nkp –nkl -s`

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>lk78k0s k0smain.rel k0ssub.rel -o –no -s`

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

Options not specified in **Table 6-6 Order of Precedence of Linker Options** 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 k0s.lmf.

```
C>lk78k0s k0smain.rel k0ssub.rel -ok0s.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>lk78k0s k0smain.rel k0ssub.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 level.

[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>lk78k0s k0smain.rel k0ssub.rel -g
```

(4) Stack decision symbol generation 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]

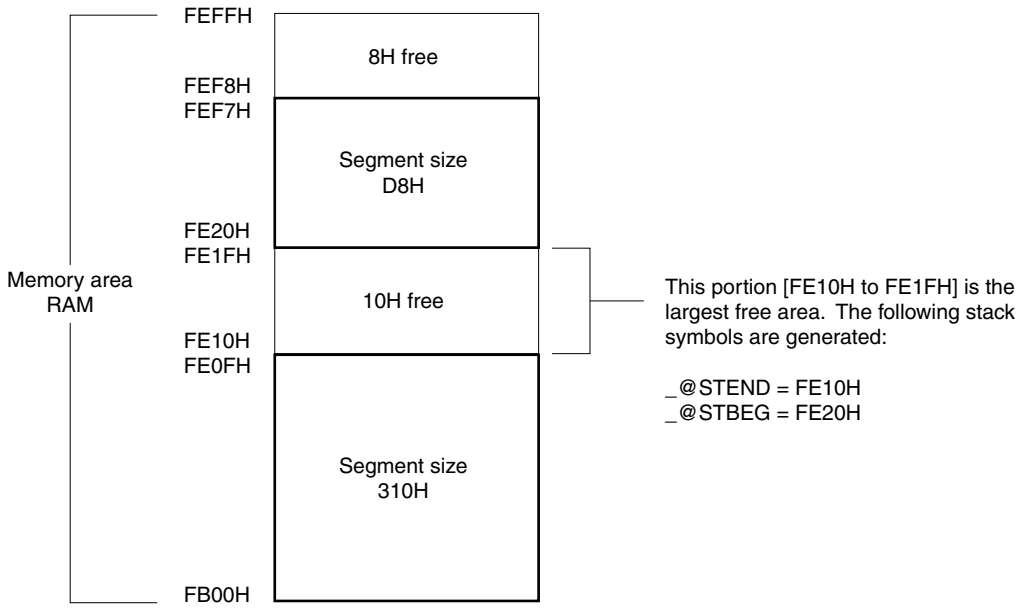
1) An 'area name' is a name in which a memory area name defined by the user or a memory area name defined by default is specified.
2) 'Area names' distinguish between uppercase and lowercase characters.
3) The linker searches the memory area specified by option -S for the largest address in which no 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 310H in RAM and a segment of size D8H located in the saddr area are input).

```
C>lk78k0s k0smain.rel k0ssub.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 K0S.DR:

```
memory ROM: (0000h, 1000h)
memory RAM: (0FE20h, 1E0h)
```

Perform link using K0S.DR.

```
C>lk78k0s k0smain.rel k0ssub.rel -dk0s.dr -s
```

(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 path name is specified in 'output file name', '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 (K0S.MAP).

```
C>lk78k0s k0smain.rel k0ssub.rel -pk0s.map -s
```

(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 K0S.MAP.

```
C>lk78k0s k0smain.rel k0ssub.rel -pk0s.map -km -s
```

Reference K0S.MAP.

```
78K/0S Series Linker Vx.xx            Date:xx Dec xxxx Page:  1

Command: k0smain.rel k0ssub.rel -km -pk0s.map -s
Para-file:
Out-file: K0SMAIN.LMF
Map-file: K0S.MAP
Direc-file:
Directive:

*** Link information ***

     3 output segment(s)
   37H byte(s) real data
    25 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=0000H  SIZE=4000H
         OUTPUT    INPUT     INPUT     BASE      SIZE
         SEGMENT   SEGMENT   MODULE    ADDRESS
         CODE                          0000H     0002H     CSEG AT
                   CODE      SAMPM     0000H     0002H
* gap *                                0002H     007EH
         ?CSEG                         0080H     0035H     CSEG
                   ?CSEG     SAMPM     0080H     0015H
                   ?CSEG     SAMPS     0095H     0020H
* gap *                                00B5H     3F4BH

  MEMORY=RAM
  BASE ADDRESS=FD00H  SIZE=0300H
         OUTPUT    INPUT     INPUT     BASE      SIZE
         SEGMENT   SEGMENT   MODULE    ADDRESS
* gap *                                FD00H     0120H
         DATA                          FE20H     0003H     DSEG AT
                   DATA      SAMPM     FE20H     0003H
* gap *                                FE23H     00DDH
* gap (Not Free Area) *                FF00H     0100H

  Target chip:uPD789026
  Device file:Vx.xx
```

Map list

(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 (K0S.MAP).

```
C>lk78k0s k0smain.rel k0ssub.rel -dk0s.dr -pk0s.map –kd -s
```

This references K0S.MAP.

```
78K/0S Series Linker Vx.xx            Date:xx xxx xxxx Page:   1

Command:k0smain.rel k0ssub.rel –dk0s.dr –pk0s.map –kd -s
Para-file:
Out-file:K0SMAIN.LMF
Map-file:K0S.MAP
Direc-file:K0S.DR                                  ← Directive file name
Directive:memory ROM: (0h, 1000h)                  ← Contents of directive file
          memory RAM: (0fe20h, 1e0h)

*** Link information ***

     3 output segment(s)
   37H byte(s) real data
    23 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 0000H    SIZE = 1000H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
         CODE                         0000H     0002H   CSEG AT
         ⋮
```

(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 option -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 (K0S.MAP).

```
C>lk78k0s k0smain.rel k0ssub.rel -g -pk0s.map –kp -s
```

This references K0S.MAP.

```
78K/0S Series Linker Vx.xx            Date:xx xxx xxxx Page:   1

Command:k0smain.rel k0ssub.rel -g -pk0s.map -kp -s
Para-file:
Out-file:K0SMAIN.LMF
Map-file:K0S.MAP
Direc-file:
Directive:


*** Link information ***

     3 output segment(s)
   37H byte(s) real data
    23 symbol(s) defined


*** Memory map ***


  SPACE = REGULAR
        :

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

78K/0S Series Linker Vx.xx            Date:xx xxx xxxx Page:   2

*** Public symbol list ***

MODULE    ATTR       VALUE    NAME

SAMPM     ADDR       0000H    MAIN
SAMPM     ADDR       0080H    START
SAMPS     ADDR       009AH    CONVAH
          NUM        FE20H    _@STBEG
          NUM        FD00H    _@STEND

 Target chip:uPD78xxx
 Device file:Vx.xx
```

Public symbol list

(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 option -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 (K0S.MAP).

```
C>lk78k0s k0smain.rel k0ssub.rel -g -pk0s.map –kl -s
```

This references K0S.MAP.

```
78K/0S Series Linker Vx.xx                Date:xx xxx xxxx Page:   1

Command:k0smain.rel k0ssub.rel -g -pk0s.map -kl -s
Para-file:
Out-file:K0SMAIN.LMF
Map-file:K0S.MAP
Direc-file:
Directive:


*** Link information ***

     3 output segment(s)
   37H byte(s) real data
    23 symbol(s) defined


*** Memory map ***


  SPACE = REGULAR

        :

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

78K/0S Series Linker Vx.xx                Date:xx xxx xxxx Page:   2

*** Local symbol list ***

MODULE  ATTR     VALUE     NAME

SAMPM   MOD                SAMPM
SAMPM   DSEG               DATA
SAMPM   ADDR     FE20H     HDTSA
SAMPM   ADDR     FE21H     STASC
SAMPM   CSEG               CODE
SAMPM   CSEG               ?CSEG
SAMPS   MOD                SAMPS
SAMPS   CSEG               ?CSEG
SAMPS   ADDR     00ACH     SASC
SAMPS   ADDR     00B2H     SASC1


 Target chip:uPD78xxx
 Device file:Vx.xx
```

Local symbol list

(8) Link list file 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 ≤ number of lines printed on 1 page ≤ 32767

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

[Example of use]

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

```
C>lk78k0s k0smain.rel k0ssub.rel -pk0s.map -ll20 -s
```

This references K0S.MAP.

```
78K/0S Series Linker Vx.xx               Date:xx xxx xxxx Page:   1

Command:k0smain.rel k0ssub.rel -pk0s.map -ll20 -s
Para-file:
Out-file:K0SMAIN.LMF
Map-file:K0S.MAP
Direc-file:
Directive:


*** Link information ***

     3 output segment(s)
   37H byte(s) real data
--------------------------------------------------------------------------------


78K/0S Series Linker Vx.xx               Date:xx xxx xxxx Page:   2

    23 symbol(s) defined


*** Memory map ***


  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 0000H    SIZE = 2000H
         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
--------------------------------------------------------------------------------



78K/0S Series Linker Vx.xx               Date:xx xxx xxxx Page:   3

         CODE                          0000H     0002H  CSEG AT
                  CODE     SAMPM       0000H     0002H
* gap *                                0002H     007EH
         ?CSEG                         0080H     0035H  CSEG
                  ?CSEG    SAMPM       0080H     0015H
                  ?CSEG    SAMPS       0095H     0020H
* gap *                                00B5H     1F4BH

  MEMORY = RAM
  BASE ADDRESS = FE00H   SIZE = 0200H
        OUTPUT     INPUT    INPUT     BASE      SIZE
        :
```

(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>lk78k0s k0smain.rel k0ssub.rel -pk0s.map –lf -s
```

(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 (K0S.ELK).

```
C>lk78k0s k0smain.rel k0ssub.rel -dk0s.dr -ek0s.elk -s
```

An error has occurred in the contents of the directive file. K0S.ELK is referenced.

```
K0S.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.
6) For a detailed explanation of the method of creating library files, refer to **CHAPTER 8 LIBRARIAN**.

[Example of use]

Input a library file (K0S.LIB).

(K0SSUB.REL is registered in the library file).

```
C>lk78k0s k0smain.rel -bk0s.lib
```

(11) Library file read path specification (-l)

Syntax:	-l path-name [, path-name] ··· (two or more path names can be specified)
Default assumption:	Path specified by environmental variable 'LIB78K0S'
	Current path, if no path is specified

[Function]

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

[Application]

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

[Explanation]

1) Option -l is only available when a library file name is specified by option -B without including a path name.
2) Two or more specifications of -l are possible. Two or more paths can be specified by separating them with ','. A blank space cannot be inserted before or after the ','.
3) Up to 10 path names can be specified per link. When two or more path names are specified, the linker searches for library files in the specified order.
4) Even if no library file exists in the specified path, an error will not result.
5) If the path name is omitted, an abort error occurs.
6) If a library file is specified by option -B without including a path name, the linker will search the paths in the following sequence.
   a. Path specified by option -l
   b. Path specified by environmental variable 'LIB78K0S'.
   c. 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 from directory A:\LIB.

```
C>lk78k0s k0smain.rel k0ssub.rel -bk0s.lib -ia:\lib -s
```

(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 link 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 (K0S.PLK) as follows.

```
;parameter file
k0smain.rel k0ssub.rel –ok0s.lmf –pk0s.map –e -s
-ta:\tmp -g
```

Enter the following on the command line.

```
C>lk78k0s -fk0s.plk -s
```

(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 link is finished. They are also deleted when link 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>lk78k0s k0smain.rel k0ssub.rel -t\tmp -s
```

(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 lk78k0s.exe startup path)
2) Path by which LK78K0S 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 lk78k0s.exe startup path)
   c. Path by which LK78K0S was started up
   d. Current directory
   e. The environmental variable PATH

[Example of use]

Specify the path for the device file as directory a:\78k0s\dev.

```
C>lk78k0s k0smain.rel k0ssub.rel -ya:\78k0s\dev -s
```

(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 to **Table 12-3 Linker Error Messages.**

[Example of use]

Specify level 2 in option -W.

```
C>lk78k0s k0smain.rel k0ssub.rel -w2 -s
```

(16) 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 linker options are unavailable.

Caution This option cannot be specified on PM plus.
To reference PM plus help, click the help button in the <Linker Options Setup> dialog box.

[Example of use]

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

```
    C>lk78k0s --

78K/0S Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx

usage:lk78k0s [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).
--                 :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,4000H)
         MEMORY RAMA:(0H,100H)
         MERGE CSEG1:= ROM
         MERGE DSEG1:AT(80H)
```

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. <Linker Options> Dialog Box (When <<Output1>> Tab Is Selected)

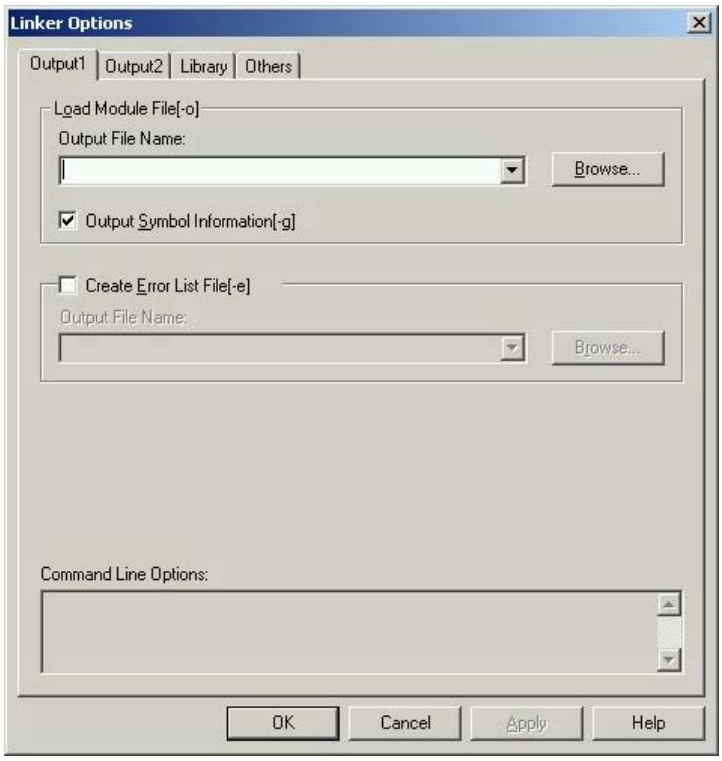


Figure 6-4. <Linker Options> Dialog Box (When <<Output2>> Tab Is Selected)

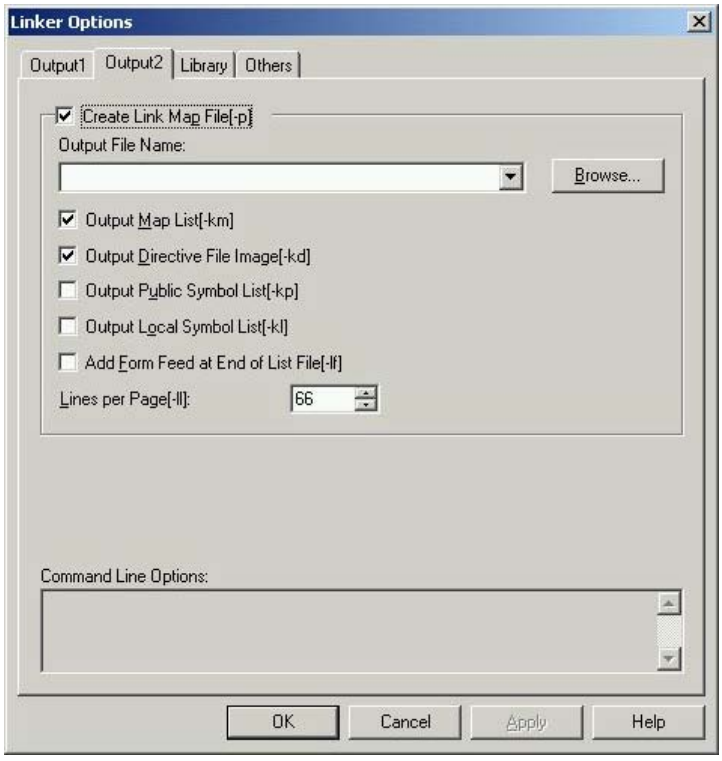


Figure 6-5. <Linker Options> Dialog Box (When <<Library>> Tab Is Selected)

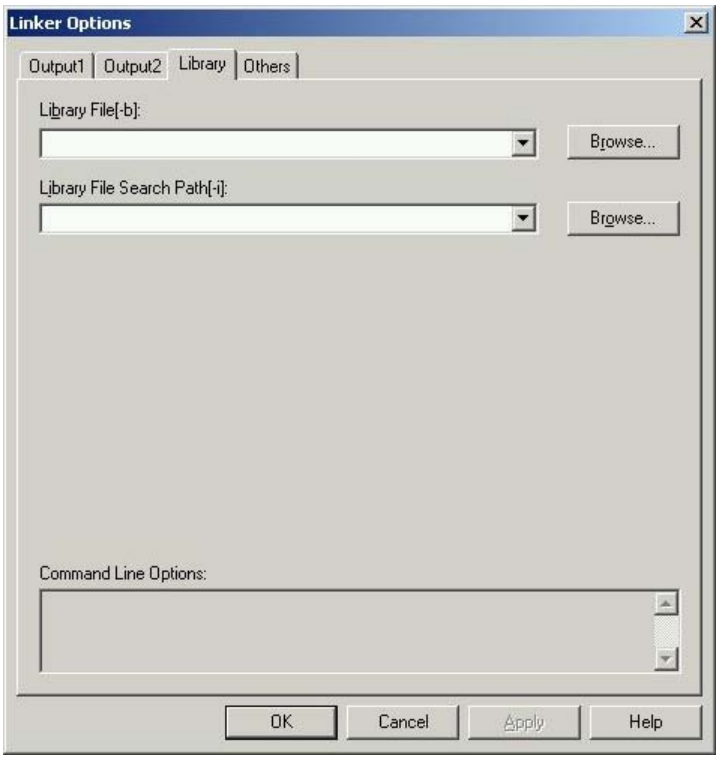
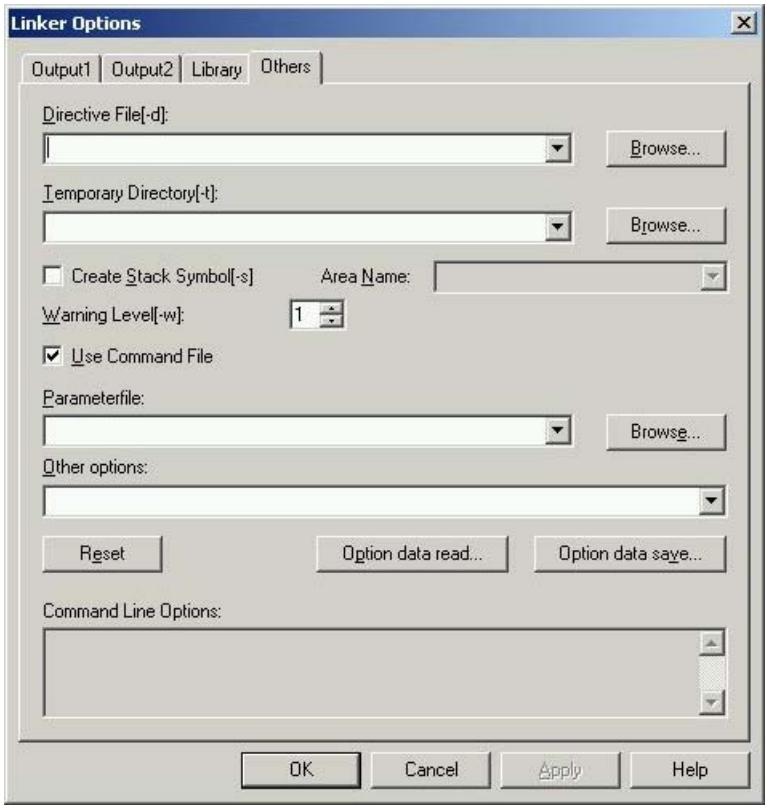


Figure 6-6. <Linker Options> Dialog Box (When <<Others>> Tab Is Selected)



6.7.2 Option settings

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

- Load Module File [-o]
  Output File Name:
  Specify the load module file output path either using the [Browse...] button or by directly inputting it.
- Output Symbol Information [-g]
  Append debug data (local symbol information) in load module file.
- 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.
- Create Link Map 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 Map 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 Local 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 32,767 characters).
- Library File [-b]
  Specify input of the specified file as a library file either using the [Browse...] button or by directly inputting it.
- Library File Search Path [-i]
  Specify input of the library file from the specified path, either using the [Browse…] button or by directly inputting it.
- Directive File [-d]
  Specify input of the specified file as a directive file, either using the [Browse...] button or by directly inputting it.
- Temporary Directory [-t]
  Specify the path for creating a temporary file, either using the [Browse…] button or by directly inputting it.
- Create Stack Symbol [-s]
  When this option is checked, the maximum empty area in the memory area is secured as the stack area.
- Area Name
  Specify the user-defined memory area name or default memory area name.
- Warning Level [-w]
  Specify the warning message output level.
  0: Don’t output warning message.
  1: Output normal warning message.
  2: Output detailed warning message.
- Use Command File
  Select this check box to create a command file.

- Parameterfile

  Read the 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 <Option Data Read> dialog box opens and the option data file is specified, that file is read.
- Option data save…

  After the <Option Data Save> dialog box opens, save the option data file 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 7 OBJECT CONVERTER

The object converter inputs the load module file output by the RA78K0S 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.

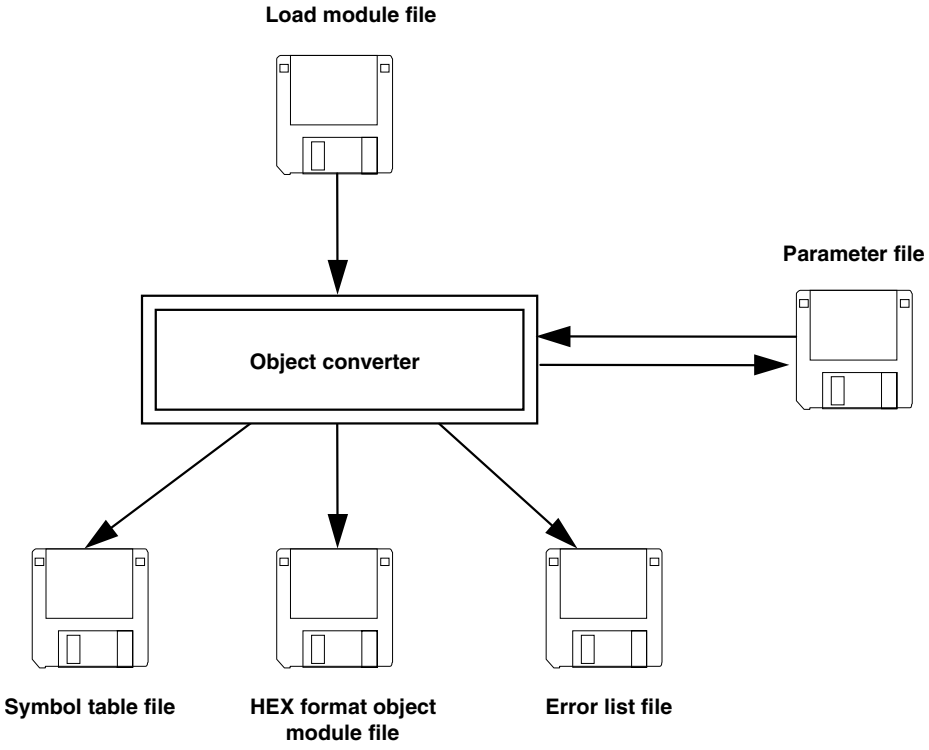
7.1 I/O Files of Object Converter

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	• These are binary image files of the object codes output as a result of linking. • These files are output by the linker.	.LMF
	Parameter files	• These files contain the parameters for the executed programs. • These files are created by the user.	.POC
Output files	HEX format object module files	• These are files created by converting load module files into hexadecimal object format. • These files are used during mask ROM development and PROM program use.	.HEX
	Symbol table files	• These files contain the symbol data included in each module of an input file.	.SYM
	Error list files	• These files contain error data from the object conversion.	.EOC

Figure 7-1. I/O Files of Object Converter



7.2 Functions of Object Converter

(1) HEX-format object module files

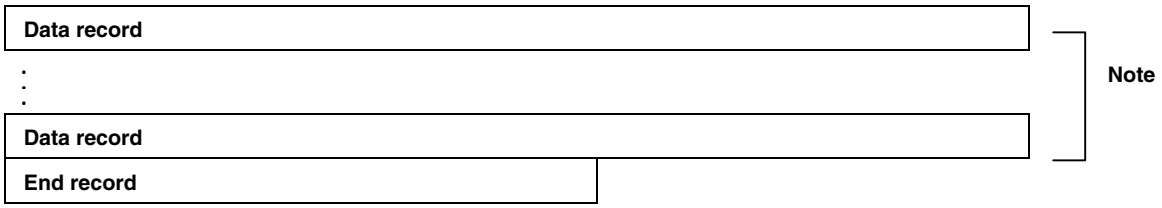
The HEX-format object module file output by the object converter can be input to a HEX loader such as a PROM programmer or a debugger.

The following is a HEX-format object module file of a sample program.

```
:0200000080007E
:10008000F5201AFC20FE229500F821FE0A27EB88B5
:100090000A25EB30FE2F00000000630F22AC000A9F
:1000A000E70A430F630F22AC000AE520130A380267
:0500B0008307833020EE
: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 at 00
(3) Fixed at 0000
(4) Record type, fixed at 01
(5) Check sum, fixed at FF

(2) Symbol table file

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

```
#04
;FF    PUBLIC
010095CONVAH
010000MAIN
010080START
;FF    SAMPM
<02FE20HDTSA
02FE21STASC
;FF    SAMPS
<0100ACSASC
0100B2SASC1
=
```

[Symbol Table File Formats]

Start of symbol table	#	04	CR	LF			
Start of public symbol	;	FF	4 blank spaces		PUBLIC	CR	LF
Note 1→		Symbol attributes	Symbol value		Public symbol name	CR	LF
		• • •	• • •		• • •	• • •	• • •
	;	FF	4 blank spaces		Module name 1	CR	LF
Start of local symbol	<	Symbol attributes	Symbol value **Note 2**		Local symbol name	CR	LF
		Symbol attributes	Symbol value		Local symbol name	CR	LF
		• • •	• • •		• • •	• • •	• • •
	;	FF	4 blank spaces		Module name 2	CR	LF
Repeated in units of object modules.		• • •	• • •		• • •	• • •	• • •
Symbol table end mark	=	CR	LF				

Public symbols

Local symbols for each module

Notes 1. Symbol attributes are the values shown in **Table 7-2 Values of Symbol Attributes**.

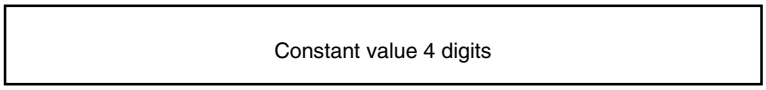
2. For symbol values, refer to **Figure 7-3 Symbol Value Formats**.

Table 7-2. Values of Symbol Attributes

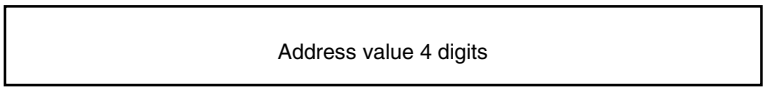
Value	Symbol Attribute
00	Constant defined by the EQU directive
01	Label within a code segment
02	Label within a data segment
03	Bit symbol
FF	Module name

Figure 7-3. Symbol Value Formats

- When the symbol attribute is NUMBER



- When the symbol attribute is ADDRESS



- 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] oc78k0s [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>oc78k0s k0s.lmf -osample.hex`

Caution If more than one object converter option is specified, separate an individual object converter option 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 parameter file specification option (-F) on the command line.

Start up the object converter from a parameter file as follows.

```
X>oc78k0s [Δload-module-file] Δ-f parameter-file-name
                                  |              |
                                 (1)            (2)
```

(1) Parameter file specification option
(2) A file which includes the data required to start up the 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 (K0S.POC) using an editor.

Contents of K0S.POC

```
;parameter file
k0s.lmf -osample.hex
-ssample.sym -r
```

Use parameter file K0S.POC to start up the object converter.

```
C>oc78k0s -fk0s.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/0S 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:uPD789xxx
 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:uPD789xxx
 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 object conversion 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>oc78k0s sample.lmf

78K/0S 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>oc78k0s k0s.lmf -a

78K/0S Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx

A018 Option is not recognized '-a'
Please enter 'OC78K0S --', if you want help messages.
Program aborted.
```

In the above example, a non-existent 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 12 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 8 types.

The classifications of the object converter options and explanations of each type are shown below.

Table 7-3. 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	Device file search path specification	-Y	Reads a device file from a specified path.
8	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

[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.

[Example of use]

Output a HEX format object module file (SAMPLE.HEX).

```
C>oc78k0s k0s.lmf -osample.hex
```

(2) Symbol table file output specification (-S/-NS)

Syntax: -S [output-file-name]
: -NS
Default assumption: -S input-file-name.SYM

[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.

[Example of use]

Output a symbol table file (SAMPLE.SYM).

```
C>oc78k0s k0s.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>oc78k0s k0s.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 ≤ complement value ≤ 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 ≤ start ≤ 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 ≤ size ≤ 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 contains internal ROM, the internal ROM range
   If the target device 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]

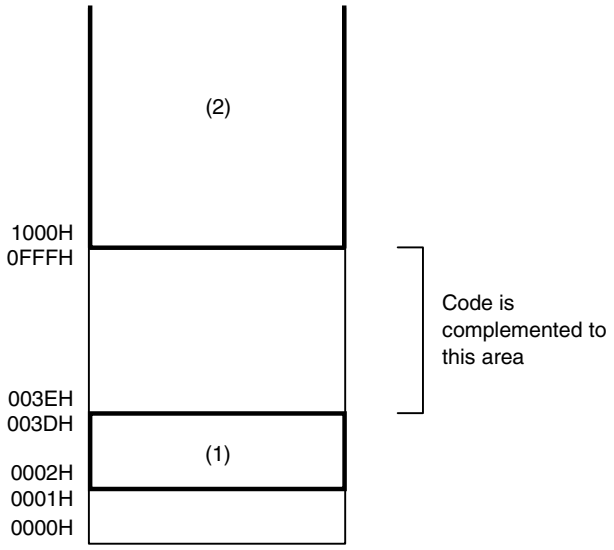
Complement an address area to which a HEX format object has not been output with code.

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 to 0FFFH.

```
:020000000200FC
:100002002B41000BFC80FE2B40000944F7083A20EC
:100012001A6720FE2822006521FED350D25014FE1A      (1)
:10002200B900059F2835002431B900059F28350005
:0C003200242156AF0A8302A807A830560C
:01000003B5D0d0026A3…                            (2)
:1010100024A5F622B667…

   ⋮

:00000001FF
```



00H is complemented to the address area 003EH to 0FFFH.

```
C>oc78k0s k0s.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 (K0S.EOC).

```
C>oc78k0s k0s.lmf -ek0s.eoc
```

(6) Parameter file specification (-F)

Syntax: -F file-name
Default assumption: Options and input file names can only be specified from the 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 K0S.POC as follows.

```
;parameter file
k0s.lmf -osample.hex
-ssample.sym -r
```

Enter the following on the command line.

```
C>oc78k0s k0s.lmf -fk0s.poc
```

(7) 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 oc78k0s.exe startup path)
2) Path by which OC78K0S 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 oc78k0s.exe startup path)
   c. Path by which OC78K0S was started up
   d. Current directory
   e. The environmental variable PATH

[Example of use]

Specify the path for the device file as directory a:\78k0s\dev.

```
C>oc78k0s k0s.lmf -ya:\78k0s\dev
```

(8) 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 object converter options are unavailable.

[Example of use]

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

```
C>oc78k0s --

78K/0S Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx

usage : oc78k0s [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 specified name] / Not.
-r/-nr          :Sort HEX object by address / Not.
-uvalue [,[start], size] :Fill up HEX object with specified value.
-ydirectory     :Set device file search path.
--              :Show this message.
DEFAULT ASSIGNMENT: -o -s -nr
```

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 7-4. <Object Converter Options> Dialog Box (When <<Output1>> Tab Is Selected)

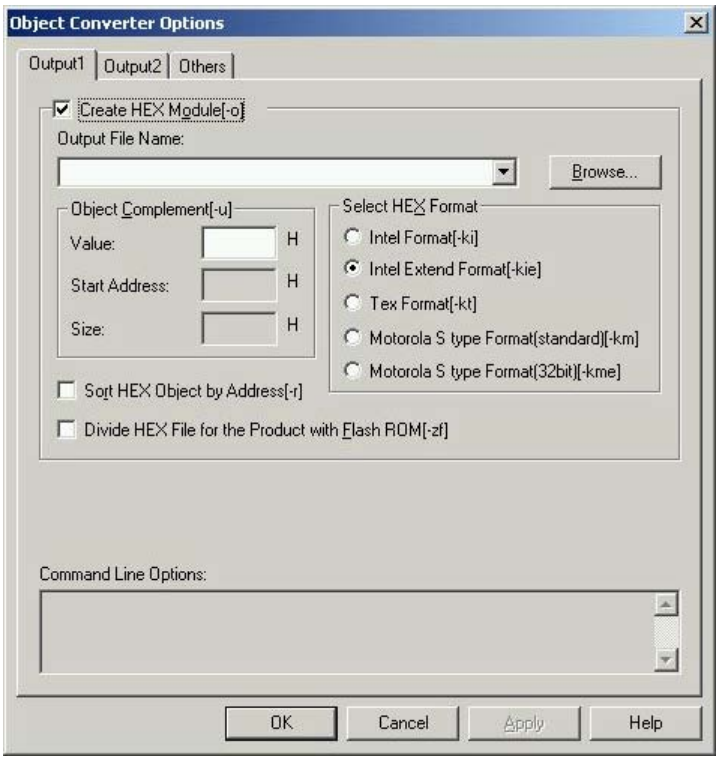


Figure 7-5. <Object Converter Options> Dialog Box (When <<Output2>> Tab Is Selected)

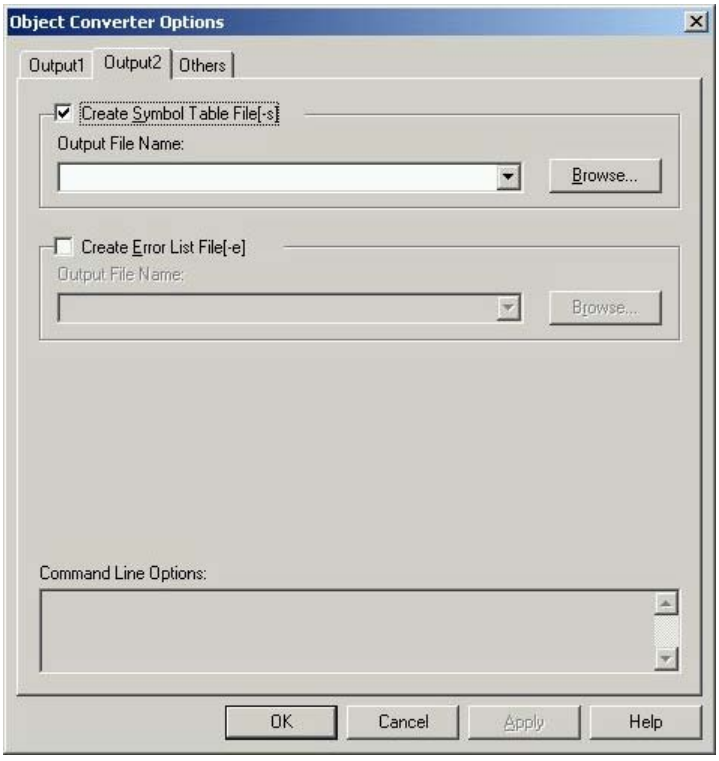
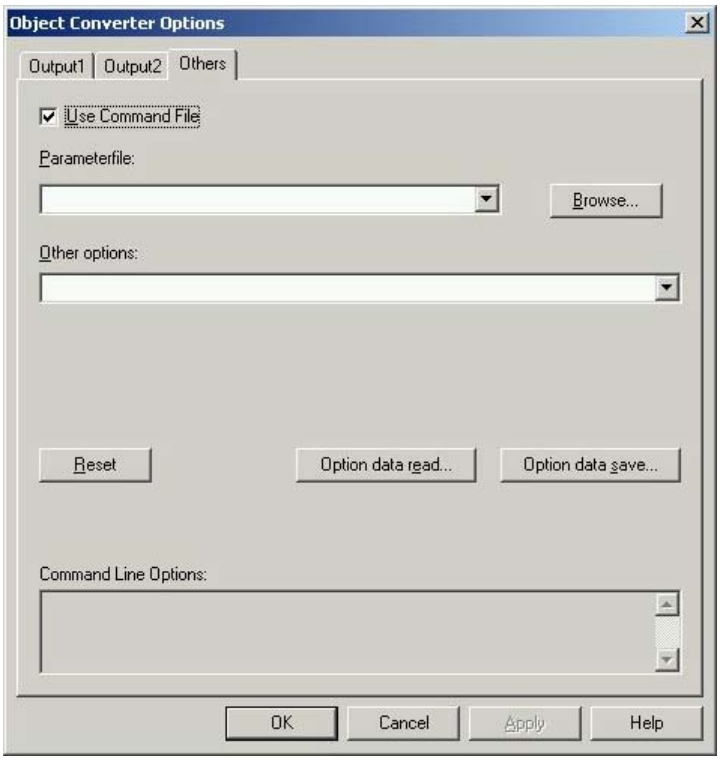


Figure 7-6. <Object Converter Options> Dialog Box (When <<Others>> Tab Is Selected)



7.5.2 Option settings

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

- Create HEX module [-o]
  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.
- Sort HEX Object by Address [-r]
  Specify this option if HEX format objects need to be sorted by address.
- 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 RA78K0S 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.

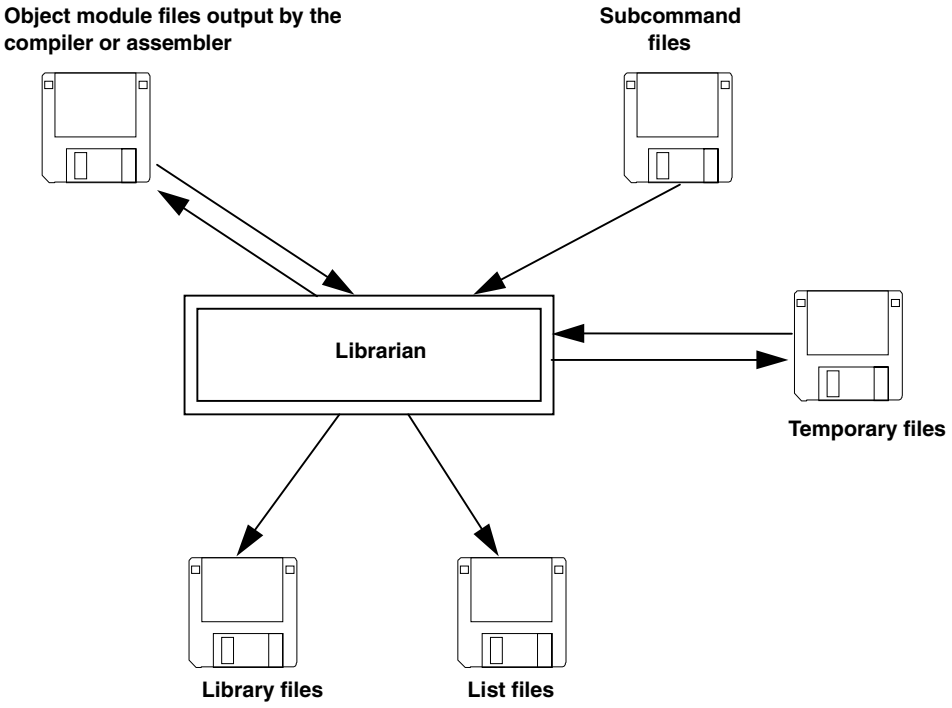
8.1 I/O Files of Librarian

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

Table 8-1. I/O Files of Librarian

Type	File Name	Explanation	Default File Type
Input files	Subcommand files	• These files contain the execute program command and the parameters. • These files are created by the user.	None
Output files	List files	• These files are the result of output of library file data.	.LST
I/O files	Object module files	• These are object module files output by the assembler or compiler.	.REL
	Library files	• These files input the library files output by the librarian and update the contents.	.LIB
	Temporary files	• These files are automatically generated by the librarian when forming a library. They are deleted when execution of the librarian is complete.	Lbxxxxxx.$$y (y = 1 to 6)

Figure 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 RA78K0S 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.

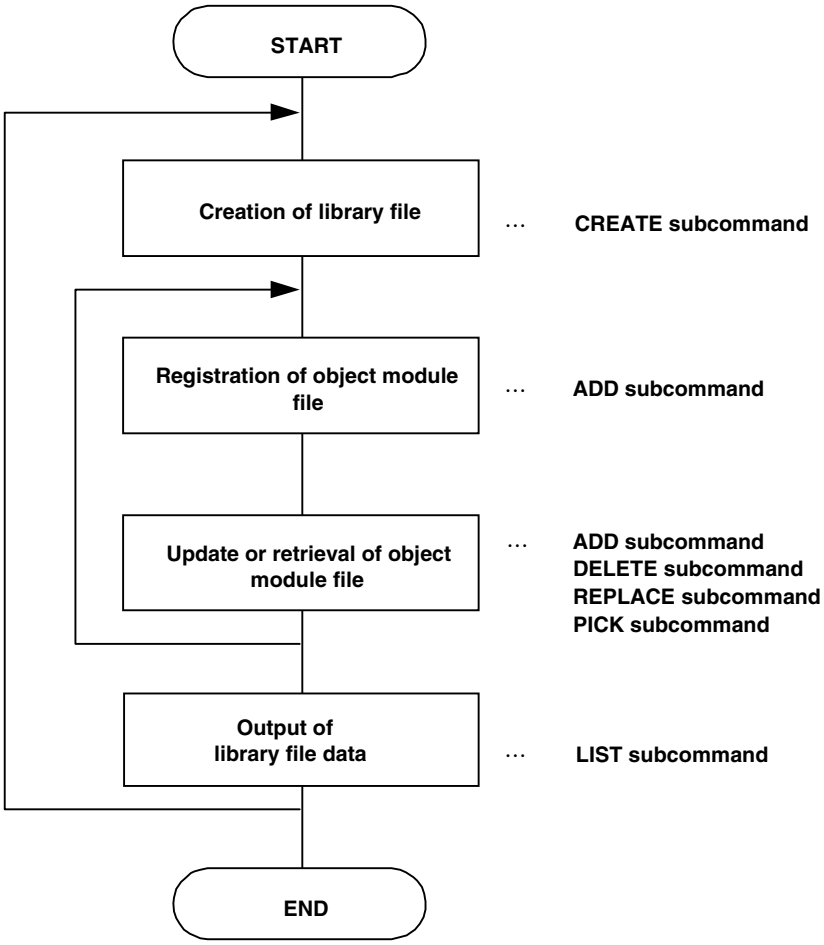
1) Module names
2) Created programs
3) Date of registration
4) Date of update
5) PUBLIC symbol data

Caution The librarian performs functions 2) and 3) listed 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] lb78k0s [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>lb78k0s -ll20 -lw80`

Caution If more than one librarian option is specified, separate an individual librarian option with a space. For a detailed explanation of librarian options, refer to 8.4 Librarian Options.

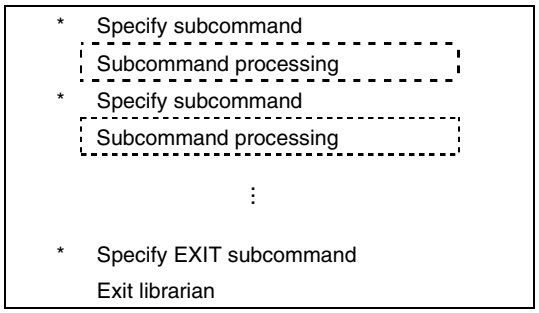
When the librarian is started up, the following startup message appears on the display.

```
78K/0S Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx
*
```

After an asterisk (*), specify a librarian subcommand.

```
*create k0s.lib
*add k0s.lib k0smain.rel k0ssub.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>lb78k0sΔ < 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 contents 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 (K0S.SLB) using an editor.

Contents of K0S.SLB

```
;library creation command
create k0s.lib
add k0s.lib k0smain.rel &
k0ssub.rel
;
exit
```

Use subcommand file K0S.SLB to start up the librarian.

```
C>lb78k0s <k0s.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/0S 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 k0s.lib
*add k0s.lib k0smain.rel k0ssub.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>lb78k0s -a

   78K/0S Series Librarian Vx.xx [xx xxx xx]
      Copyright (C) NEC Electronics Corporation xxxx, xxxx
   A018 Option is not recognized '-z'
   Usage: LB78K0S [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 12 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 library 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 specifies the number of characters in one line in a list file.

[Application]

Specify option -LW to change the number of characters in one line in a list file.

[Explanation]

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

$$72 \leq \text{number of characters printed on 1 line} \leq 260$$

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>lb78k0s -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 ≤ number of lines printed on 1 page ≤ 32767

   If a numerical value outside this range, or something other than a numerical value, is specified, an abort error occurs.
2) If the number of lines is omitted, 66 will be specified.
3) If the number of lines specified is 0, no page breaks will be made.
4) If the LIST subcommand is not specified, option -LL is ignored.
5) If option -LL is specified 2 or more times, the last specified item will take precedence.

[Example of use]

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

```
C>lb78k0s -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>lb78k0s -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 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 b:\TMP.

```
C>lb78k0s -tb:\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 lb78k0s.exe startup path)
2) Path by which LB78K0S 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 lb78k0s.exe startup path)
   c. Path by which LB78K0S was started up
   d. Current directory
   e. The environmental variable PATH

[Example of use]

Specify the path for the device file as directory a:\78k0s\dev.

```
C>lb78k0s -ya:\78k0s\dev
```

(4) 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 subcommands. Refer to these when executing the librarian.

[Explanation]

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

[Example of use]

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

```
   C>lb78k0s --

78K/0S Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx
+-------------------------------------------------------------------------+
| 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                                                      |
+-------------------------------------------------------------------------+
```

8.5 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 [Δoption] Δlibrary-file-name [Δoption] transaction [Δoption]
                          |                          |
                         (1)                        (2)
```

(1) The library file name specified immediately before can be replaced with '.'.

(2) Transaction = Δobject-module-file-name Δlibrary-file-name [∇(∇module-name [∇, ···])]

(1) CREATE

Syntax: CREATE∆library-file-name [∆transaction]
Abbreviated format: C

[Function]

The CREATE subcommand creates a new library file.

[Explanation]

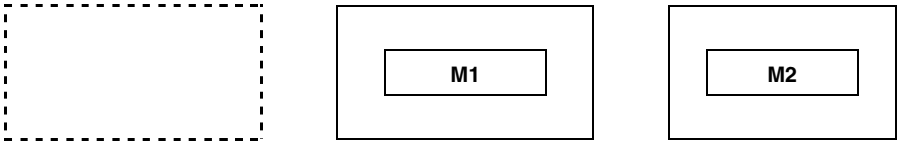
1) The size of the created library file becomes 0.
2) When a transaction is specified, a module is registered at the same time as the library file is created.
3) Library file name: If a file with the same name already exists, it will be overwritten.
4) Transaction: An object module file carrying the 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]

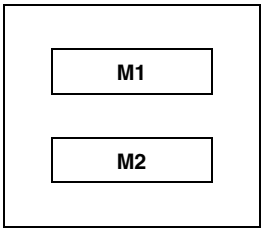
Register modules M1 and M2 at the same time as a library file is created.

```
*create k0s.lib m1.rel m2.rel
```

<Before file creation>



<After file creation>
K0S.LIB



(2) ADD

Syntax: ADD∆library-file-name ∆transaction

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.

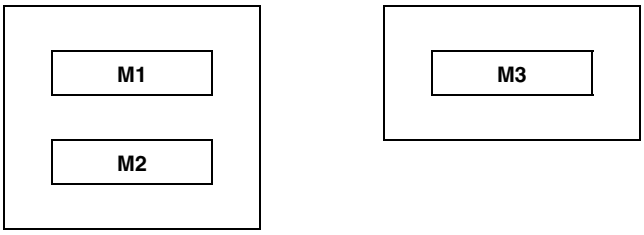
[Example of use]

Add a module (M3) to a library file (K0S.LIB).

```
*add k0s.lib m3.rel
```

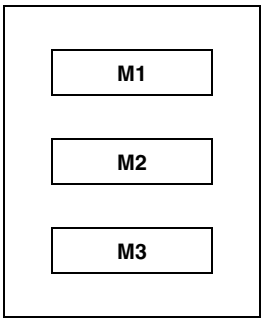
<Before addition>

K0S.LIB



<After addition>

K0S.LIB



(3) DELETE

Syntax: DELETE∆library-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.

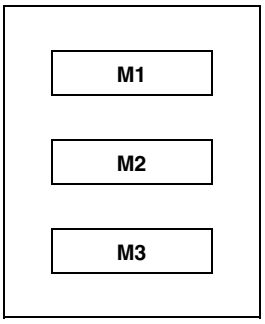
[Example of use]

Delete modules (M1, M3) from a library file (K0S.LIB).

```
*delete k0s.lib (m1.rel m3.rel)
```

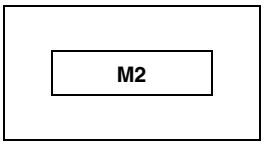
<Before deletion>

K0S.LIB



<After deletion>

K0S.LIB



(4) REPLACE

Syntax: REPLACE∆library-file-name∆transaction
Abbreviated format: R

[Function]

The REPLACE subcommand replaces module in a library file with the module in other object module files.

[Explanation]

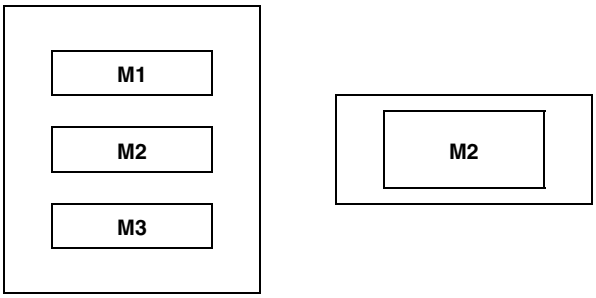
1) If no module in the library file has the same name as the replacement module, an error will result.
2) If a public symbol contained in the replacement module is the same as a public symbol in the library file, an error will occur.
3) The file name of the replacement object module must be the same 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]

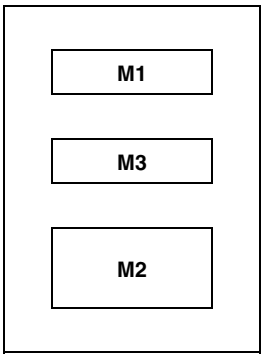
Replace a module (M2) in a library file (K0S.LIB).

```
*replace k0s.lib m2.rel
```

<Before replacement>
K0S.LIB



<After replacement>
K0S.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: PICK∆library-file-name ∇(∇module-name [∇, · · ·]∇)

Abbreviated format: P

[Function]

The PICK subcommand retrieves a specified module from an existing library file.

[Explanation]

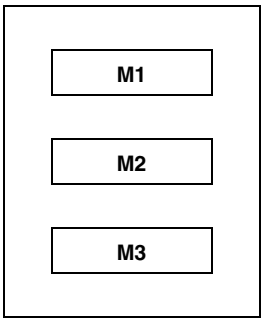
1) The retrieved module becomes an object module file with the file name under which it was registered in the library file.
2) If the specified module name does not exist in the library file, an error will result.
3) If an error occurs, processing is interrupted. However, if an error occurs when two 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]

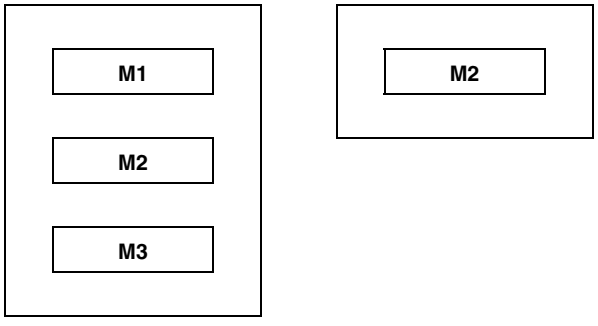
Retrieve a module (M2) from a library file (K0S.LIB).

```
*pick k0s.lib (m2.rel)
```

<Before pickup>
K0S.LIB



<After pickup>
K0S.LIB



(6) LIST

Syntax: LIST [Δoption] Δlibrary-file-name [∇(∇module-name [∇, ···]∇)]

Option: -PUBLIC/-NOPUBLIC
: -O ∇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 (K0S.LIB) to a list file (K0S.LST). Specify option -P so that public symbol data will be output.

```
*list -p -ok0s.lst k0s.lib
```

List file (K0S.LST) is referenced.

```
78K/0S Series librarian Vx.xx    DATE : xx xxx xx            PAGE   1
LIB-FILE NAME : K0S.LIB     (xx xxx xx)
0001  M1.REL    (xx xxx xx)
  sym1   sym2   sym3
  NUMBER OF PUBLIC SYMBOLS :   3
0002  M3.REL    (xx xxx xx)
  NUMBER OF PUBLIC SYMBOLS :   0
0003  M2.REL    (xx xxx xx)
  bit1   bit2
  NUMBER OF PUBLIC SYMBOLS :   2
```

(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 flies from PM plus.

8.6.1 Option setting method

Select [1 lb78k0sp] 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. <Library File Name> Dialog Box

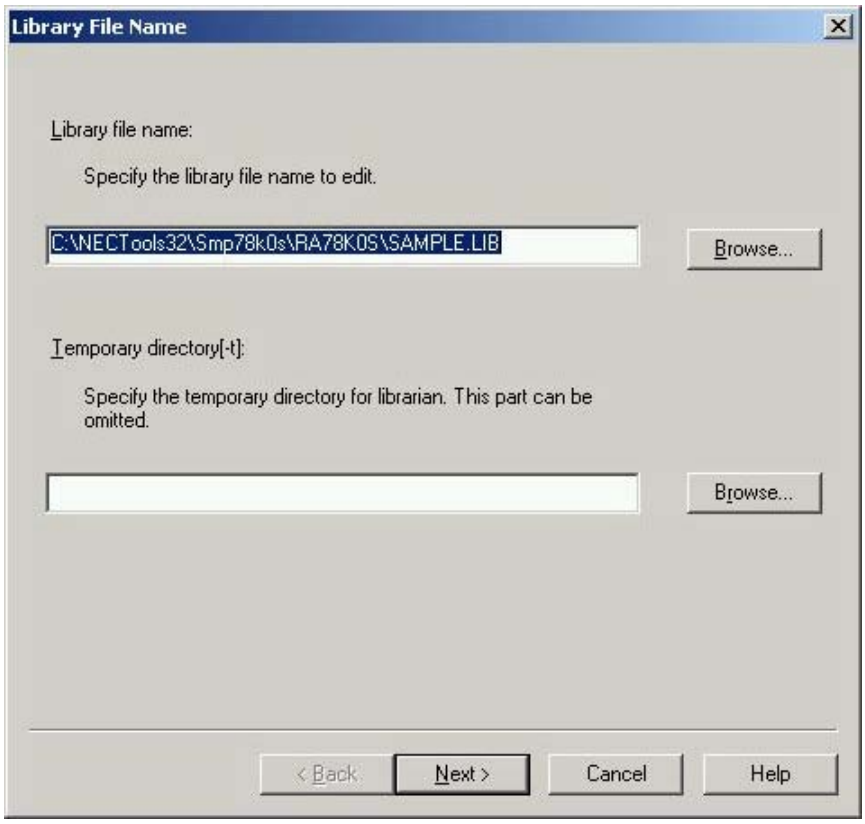
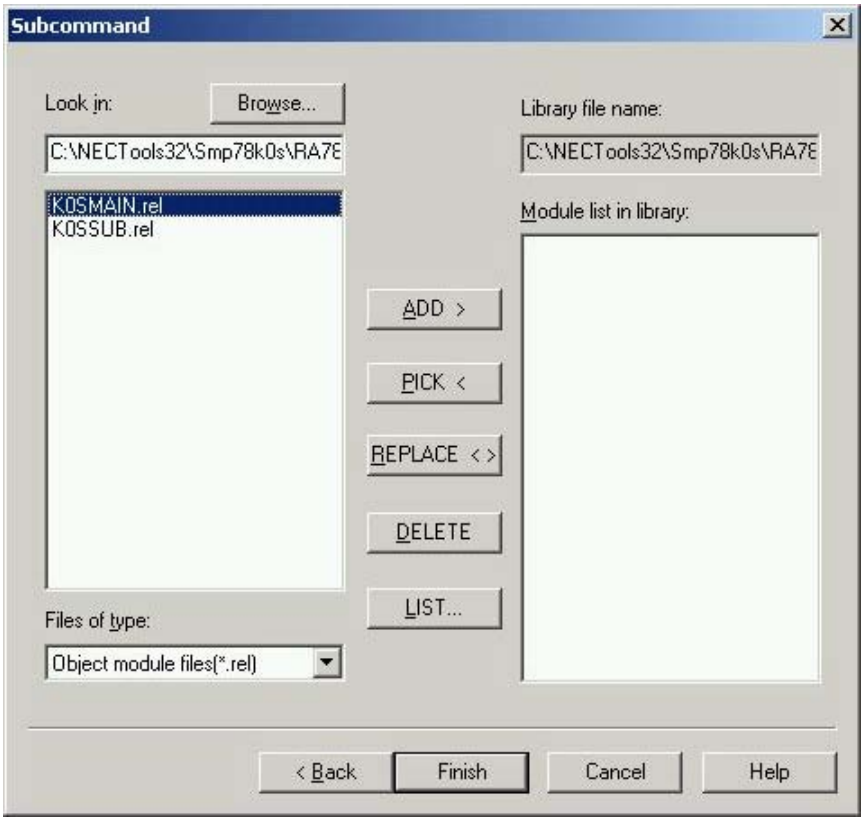


Figure 8-4. <Subcommand> Dialog Box



8.6.2 Option settings

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

- Librarian file name

  Specify the library file path for editing either using the [Browse...] button or by directly inputting it.
- Temporary directory [-t]

  Specify the path of the temporary file to be created either using the [Browse...] button or by directly inputting it.
- Look in

  Select the object module files to be made into a library either using the [Browse...] button or by directly inputting it.
- ADD

  Add a module to an existing library file.
- PICK

  Retrieve the specified module from an existing library file.
- REPLACE

  Replace an existing library file module with the module of another object module file.
- DELETE

  Delete a module from an existing library file.
- LIST

  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 assemble list file. This eliminates the troublesome task of looking at an assemble list while referring to a link map.

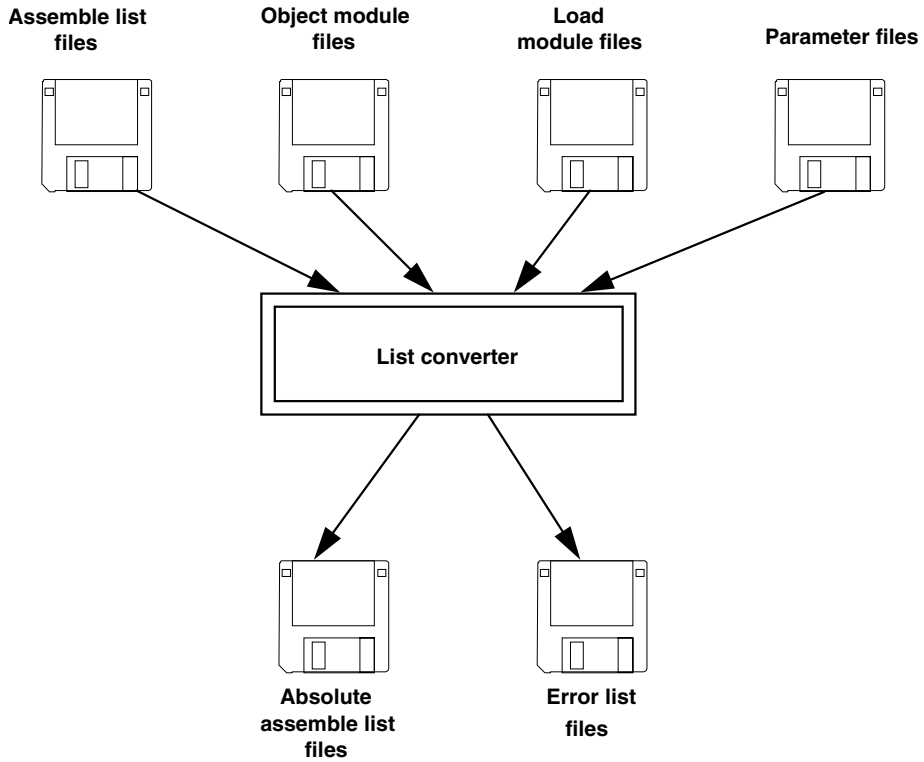
9.1 I/O Files of List Converter

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	• These are binary files including relocation data and symbol data regarding machine language data and machine language location addresses.	.REL
	Assemble list files	• These are files containing assembly data such as assemble lists and cross-reference lists.	.PRN
	Load module files	• These are binary image files which contain object code as a result of linking.	.LMF
	Parameter files	• These files contain the parameters for the executed program. • These files are created by the user.	.PLV
Output files	Absolute assemble list files	• This is a list file which embeds actual addresses in relocatable addresses and symbols in the input file.	.P
	Error list files	• These are files containing error data generated during list conversion.	.ELV

Figure 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

```
21  21 ----             CSEG
22  22 0000         START:
23  23
24  24              ;chip initialize
25  25
26  26 0000  F5201A MOV   HDTSA, #1AH
27  27 0003  FC20FE MOVW  HL, #HDTSA ;set hex 2-code data in HL register
28  28
29  29 0006 R220000 CALL  !CONVAH    ;convert ASCII<- HEX
30  30                               ;output BC-register<- ASCII code
31  31 0009  F821FE MOVW  DE, #STASC ;set DE<- store ASCII code table
32  32 000C  0A27   MOV   A, B
33  33 000E  EB     MOV   [DE], A
34  34 000F  88     INCW  DE
35  35 0010  0A25   MOV   A, C
36  36 0012  EB     MOV   [DE], A
```

- Absolute assemble list

```
21  21 ----             CSEG
22  22 0080         START:
23  23
24  24              ;chip initialize
25  25
26  26 0080  F5201A MOV   HDTSA, #1AH
27  27 0083  FC20FE MOVW  HL, #HDTSA ;set hex 2-code data in HL register
28  28
29  29 0086 R229500 CALL  !CONVAH    ;convert ASCII<- HEX
30  30                               ;output BC-register<- ASCII code
31  31 0089  F821FE MOVW  DE, #STASC ;set DE<- store ASCII code table
32  32 008C  0A27   MOV   A, B
33  33 008E  EB     MOV   [DE], A
34  34 008F  88     INCW  DE
35  35 0090  0A25   MOV   A, C
36  36 0092  EB     MOV   [DE], A
```

Example 2. Embedding of object codes

- Assemble list

```
21  21 ----          CSEG
22  22 0000        START:
23  23
24  24              ;chip initialize
25  25
26  26 0000  F5201A MOV   HDTSA, #1AH
27  27 0003  FC20FE MOVW  HL, #HDTSA ;set hex 2-code data in HL register
28  28
29  29 0006 R220000 CALL  !CONVAH    ;convert ASCII<- HEX
30  30                                ;output BC-register<- ASCII code
31  31 0009  F821FE MOVW  DE, #STASC ;set DE<- store ASCII code table
32  32 000C  0A27   MOV   A, B
33  33 000E  EB     MOV   [DE], A
34  34 000F  88     INCW  DE
35  35 0010  0A25   MOV   A, C
36  36 0012  EB     MOV   [DE], A
```

- Absolute assemble list

```
21  21 ----          CSEG
22  22 0080        START:
23  23
24  24              ;chip initialize
25  25
26  26 0080  F5201A MOV   HDTSA, #1AH
27  27 0083  FC20FE MOVW  HL, #HDTSA ;set hex 2-code data in HL register
28  28
29  29 0086 R229500 CALL  !CONVAH    ;convert ASCII<- HEX
30  30                                ;output BC-register<- ASCII code
31  31 0089  F821FE MOVW  DE, #STASC ;set DE<- store ASCII code table
32  32 008C  0A27   MOV   A, B
33  33 008E  EB     MOV   [DE], A
34  34 008F  88     INCW  DE
35  35 0090  0A25   MOV   A, C
36  36 0092  EB     MOV   [DE], A
```

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) Startup from the command line

```
X>lc78k0s [Δoption]···Δinput-file-name [Δoption]···[Δ]
 |      |          |               |              |
(1)    (2)        (3)             (4)            (3)
```

(1) Current drive name
(2) Command file name of the list converter
(3) Enter detailed instructions for the operation of the list converter.
(4) Primary name of assemble list

Example `C>lc78k0s k0smain -lk0s.lmf`

Cautions 1. In (3) above, when specifying two or more list converter options, separate an individual list converter option 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 parameter file specification option (-F) on the command line.

Start up the list converter from a parameter file as follows.

```
X>lc78k0s [Δinput-file-name] Δ-f parameter-file-name
                                  |          |
                                 (1)        (2)
```

(1) Parameter file specification option

(2) A file which includes the data required to start up the list converter

Remark Create the parameter file using an editor.

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

```
[[[Δ] option [Δoption] ··· [Δ]Δ]]···
```

1) If the input file name is omitted from the command line, only 1 input file name can be specified in the parameter file.
2) The input file name can also be 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 (K0S.PLV) using an editor.

Contents of K0S.PLV

```
;parameter file
k0smain -lk0s.lmf
-ek0s.elv
```

Use parameter file (K0S.PLV) to start up the list converter.

```
C>lc78k0s -fk0s.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/0S 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>lc78k0s k0smain -a

List Conversion Program for RA78K/0S Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

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

When the list converter outputs an error message and aborts list conversion, look for the cause in **CHAPTER 12 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	Outputs 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 the 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 K0SMAIN.PRN, the object module file name is SAMPLE.REL, and the load module file name is K0S.LMF.

```
C>lc78k0s k0smain -rsample.rel -lk0s.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 K0SMAIN.PRN and the load module file name is SAMPLE.LMF.

```
C>lc78k0s k0smain -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>lc78k0s k0smain -osample.p -lk0s.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>lc78k0s k0smain -esample.elv
```

The error list file (SAMPLE.ELV) is referenced.

```
*** WARNING W101 Load module file is older than object module file'K0SMAIN.LMF,
K0SMAIN.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 (K0S.PLV) are as follows.

```
;parameter file
k0smain -lk0s.lmf
-ek0s.elv
```

Enter the following on the command line.

```
C>lc78k0s -fk0s.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 list converter options are unavailable.

[Example of use]

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

```
C>lc78k0s --

List Conversion Program for RA78K/0S Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx

usage : LC78K0S [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).
-f [file]    :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. <List Converter Options> Dialog Box (When <<Output>> Tab Is Selected)

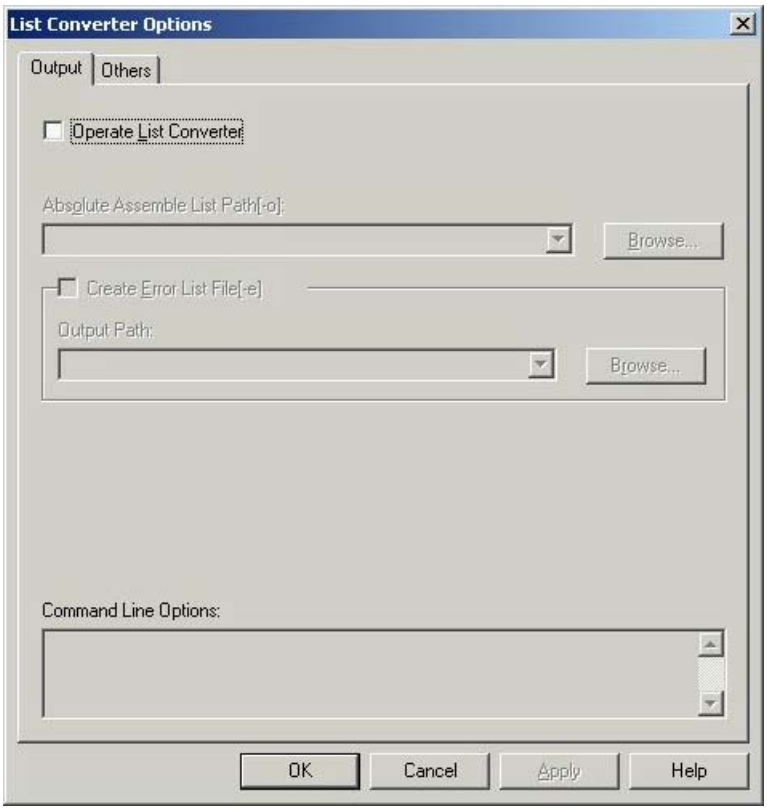
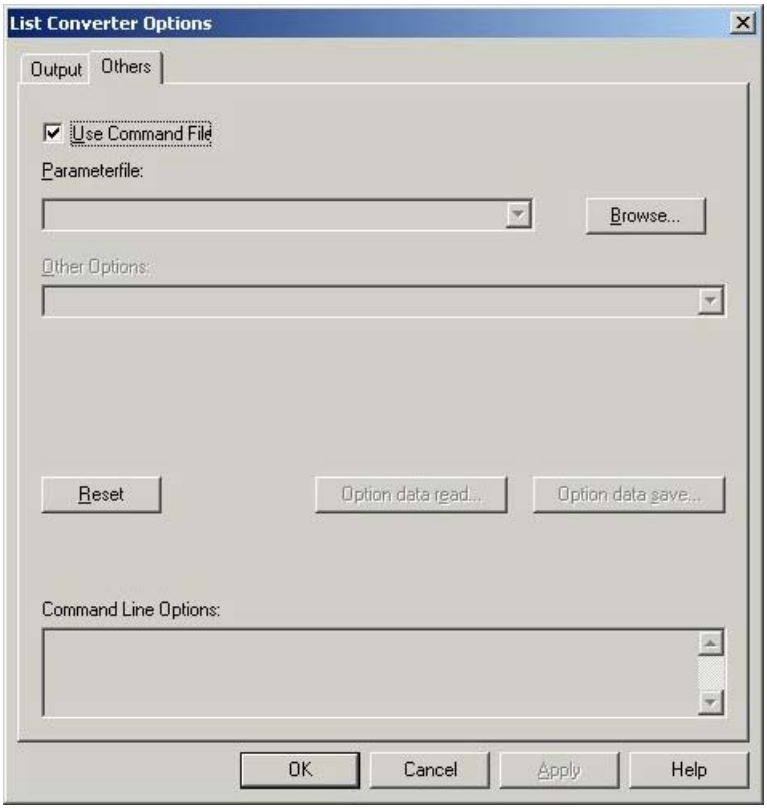


Figure 9-3. <List Converter Options> Dialog Box (When <<Others>> Tab Is Selected)



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 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

10.1 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

10.1.1 Assemble list file headers

The header is always output at the beginning of an assemble list file.

[Output format]

```
78K/0S Series Assembler (1)Vx.xx (2)                Date:(3)xx xxx xxxx Page:(4)  1
(5)
Command:  (6)k0smain.asm -c9024
Para-file:(7)
In-file:  (8)K0SMAIN.ASM
Obj-file: (9)K0SMAIN.REL
Prn-file:(10)K0SMAIN.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

10.1.2 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
 (2)2  (2)2                 (5)   NAME     SAMPM
     ⋮
  28    28
  29    29 (6)0006 (8)R220000 (5)  CALL  !CONVAH  ;convert ASCII<- HEX
  30    30                    (5)                 ;output BC-register<-ASCII code
  31    31 (6)0009  00000000        MOV  DE, #STASC;set DE<-store ASCII code table
(7)**   ERROR  F202,  STNO    31 (  0) Illegal operand
           (6)000D  00
  32    32 (6)000E (8)0A27    (5)   MOV   A, B
  33    33 (6)0010 (8)EB      (5)   MOV   [DE], A
     ⋮
  Segment informations:

  ADRS      LEN        NAME

 (9)FE20  (10)0003H  (11)DATA
 (9)0000  (10)0002H  (11)CODE
 (9)0000  (10)0017H  (11)?CSEG

  Target chip:(12)uPD78xxxx
  Device file:(13)Vx.xx
  Assembly complete,      (14)1 error(s) and   (15)0 warning(s) found. (  (16)31)
```

[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)	RA78K0S target device
(13)	Device file version no.
(14)	Number of fatal errors
(15)	Number of warnings
(16)	Final error line

10.1.3 Symbol list

A symbol list outputs the symbols (including local symbols) defined in a source module.

[Output format]

```
Symbol Table List

     VALUE      ATTR      RTYP     NAME                VALUE      ATTR      RTYP     NAME
                (2)CSEG             (4)?CSEG                       (2)CSEG             (4)CODE
(1)  ----H              (3)EXT     (4)CONVAH                      (2)DSEG             (4)DATA
(1)  FE20H      ADDR               (4)HDTSA          (1)  0H     (2)ADDR   (3)PUB     (4)MAIN
                MOD                (4)SAMPM          (1)  0H     (2)ADDR   (3)PUB     (4)START
(1)  FE21H      ADDR               (4)STASC
```

[Explanation of output items]

Item	Details
(1)	Symbol value
(2)	Symbol attributes CSEG: Code segment name DSEG: Data segment name BSEG: Bit segment name MAC: Macro name MOD: Module name SET: Symbol defined by SET directive NUM: NUMBER attribute symbol ADDR: ADDRESS attribute symbol SABIT: BIT attribute symbol (saddr.bit) SFBIT: BIT attribute symbol (sfr.bit) RBIT: BIT attribute symbol (A.bit, X.bit) Blank: External reference symbol declared by EXTRN or EXTBIT *****: Undefined symbol
(3)	Symbol reference format EXT: External reference symbol declared by EXTRN (SADDR attribute) EXTB: External reference symbol declared by EXTBIT (saddr.bit) PUB: External definition symbol declared by PUBLIC Blank: Local symbol, segment name, macro name, module name *****: Undefined symbol
(4)	Defined symbol name

10.1.4 Cross-reference list

A cross-reference list outputs data indicating where (on what line) symbols are defined in a source module.

[Output format]

```
    Cross-Reference List

   NAME         VALUE       R      ATTR       RTYP       SEGNAME     XREFS
(1)?CSEG                        (4)CSEG                 (6)?CSEG    (7)21#
(1)CODE                         (4)CSEG                 (6)CODE     (7)18#
(1)CONVAH    (2)----H    (3)E               (5)EXT                  (7)12     29
(1)DATA                         (4)DSEG                 (6)DATA     (7)14#
(1)HDTSA     (2)FE20H           (4)ADDR                 (6)DATA     (7)15#    26    27
(1)MAIN      (2)0H              (4)ADDR    (5)PUB       (6)CODE     (7)11@    19#
(1)SAMPM                        (4)MOD                              (7)2#
(1)START     (2)0H       (3)R   (4)ADDR     (5)PUB     (6)?CSEG    (7)11@    19    22#
(1)STASC     (2)FE21H           (4)ADDR                 (6)DATA     (7)16#    31
```

[Explanation of output items]

Item	Details
(1)	Defined symbol name
(2)	Symbol value
(3)	Relocation attributes R: Relocatable symbol E: External symbol Blank: Absolute symbol *: Undefined symbol
(4)	Symbol attributes CSEG: Code segment name DSEG: Data segment name BSEG: Bit segment name MAC: Macro name MOD: Module name SET: Symbol defined by SET directive NUM: NUMBER attribute symbol ADDR: ADDRESS attribute symbol SABIT: BIT attribute symbol (saddr.bit) SFBIT: BIT attribute symbol (sfr.bit) RBIT: BIT attribute symbol (A.bit, X.bit) Blank: External reference symbol declared by EXTRN or EXTBIT *****: Undefined symbol
(5)	Symbol reference format EXT: External reference symbol declared by EXTRN (SADDR attribute) EXTB: External reference symbol declared by EXTBIT (saddr.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@

10.1.5 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

10.2 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

10.2.1 Link list file headers

The header is always output at the beginning of a link list file.

[Output format]

```
78K/0S Series Linker (1)Vx.xx       Date:(2)xx xxx xxxx Page:(3)1

Command:   (4)k0smain.rel k0ssub.rel -ok0s.map -dk0s.dr
Para-file: (5)
Out-file:  (6)K0S.LMF
Map-file:  (7)K0SMAIN.MAP
Direc-file:(8)
Directive: (9)

*** 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

10.2.2 Map list

The map list outputs data on the location of segments.

[Output format]

```
     *** Memory map ***

(1)  SPACE = REGULAR

     MEMORY =(2)ROM
     BASE ADDRESS =(3)0000H  SIZE =(4)2000H
          OUTPUT  INPUT       INPUT    BASE         SIZE
          SEGMENT             SEGMENT              MODULE       ADDRESS
       (6)CODE                       (9)0000H    (10)0002H    (11)CSEG AT
               (7)CODE     (8)SAMPM  (9)0000H    (10)0002H
(5)* gap *                           (9)0002H    (10)007EH
       (6)?CSEG                      (9)0080H    (10)0035H    (11)CSEG
               (7)?CSEG    (8)SAMPM  (9)0080H    (10)0015H
               (7)?CSEG    (8)SAMPS  (9)0095H    (10)0020H
(5)* gap *                           (9)00B5H    (10)1F4BH

     MEMORY = RAM
     BASE ADDRESS = (3)FE00H      SIZE =(4)0200H
          OUTPUT INPUT        INPUT    BASE         SIZE
          SEGMENT           SEGMENT    MODULE       ADDRESS
(5)* gap *                           (9)FE00H    (10)0020H
       (6)DATA                       (9)FE20H    (10)0003H    (11)DSEG AT
               (7)DATA     (8)SAMPM  (9)FE20H    (10)0003H
(5)* gap *                           (9)FE23H    (10)00DDH
(5)* gap (Not Free Area) *           (9)FE00H    (10)0100H

   Target chip:(12)uPD78xxx
   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.

10.2.3 Public symbol list

A public symbol list outputs data on public symbols defined in an input module.

[Output format]

```
*** Public symbol list ***

MODULE      ATTR       VALUE      NAME

(1)SAMPM    (2)ADDR    (3)0000H   (4)MAIN
(1)SAMPM    (2)ADDR    (3)0080H   (4)START
(1)SAMPS    (2)ADDR    (3)0095H   (4)CONVAH
```

[Explanation of output items]

Item	Details			
(1)	Name of module in which public symbols are defined			
(2)	Symbol attributes			
	CSEG:	Code segment name	NUM:	NUMBER attribute symbol
	DSEG:	Data segment name	ADDR:	ADDRESS attribute symbol
	BSEG:	Bit segment name	SABIT:	BIT attribute symbol (saddr.bit)
	MAC:	Macro name	SFBIT:	BIT attribute symbol (sfr.bit)
	MOD:	Module name	RBIT:	BIT attribute symbol (A.bit, X.bit)
	SET:	Symbol defined by SET directive	Blank:	External reference symbol declared by EXTRN or EXTBIT
			*****:	Undefined symbol
(3)	Symbol value			
(4)	Public symbol name			

10.2.4 Local symbol list

A local symbol list outputs data on local symbols defined in an input module.

[Output format]

```
     *** Local symbol list ***

   MODULE      ATTR       VALUE       NAME

(1)SAMPM    (2)MOD                 (4)SAMPM
(1)SAMPM    (2)DSEG                (4)DATA
(1)SAMPM    (2)ADDR    (3)FE20H    (4)HDTSA
(1)SAMPM    (2)ADDR    (3)FE21H    (4)STASC
(1)SAMPM    (2)CSEG                (4)CODE
(1)SAMPM    (2)CSEG                (4)?CSEG
(1)SAMPS    (2)MOD                 (4)SAMPS
(1)SAMPS    (2)CSEG                (4)?CSEG
(1)SAMPS    (2)ADDR    (3)00ACH    (4)SASC
(1)SAMPS    (2)ADDR    (3)00B2H    (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	NUM:	NUMBER attribute symbol
	DSEG:	Data segment name	ADDR:	ADDRESS attribute symbol
	BSEG:	Bit segment name	SABIT:	BIT attribute symbol (saddr.bit)
	MAC:	Macro name	SFBIT:	BIT attribute symbol (sfr.bit)
	MOD:	Module name	RBIT:	BIT attribute symbol (A.bit, X.bit)
	SET:	Symbol defined by SET directive	Blank:	External reference symbol declared by EXTRN or EXTBIT
			*****:	Undefined symbol
(3)	Symbol value			
(4)	Local symbol name			

10.2.5 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 'K0SMAIN.REL'
```

[Explanation of output items]

Item	Details
(1)	Error no.
(2)	Error message

10.3 List Output by Object Converter

The object converter outputs the following list.

Output List File Name	Output List Name
Error list file	Error list

10.3.1 Error list

An error list stores the error messages output when the object converter is started up.

[Output format]

Same as for the error list output by the linker.

10.4 List Output by Librarian

The librarian outputs the following list.

Output List File Name	Output List Name
List file	Library data output list

10.4.1 Library data output list

The library data output list outputs data on the modules in a library file.

[Output format]

```
   78K/0S Series librarian Vx.xx       DATE:(1) xx xxx xx          PAGE(2)1
   LIB-FILE NAME:(3)K0S.LIB      ((4)xx xxx xx)
(5)0001  (6)K0SMAIN.REL    ((7)xx xxx xx)
   (8)MAIN                         (8)START
    NUMBER OF PUBLIC SYMBOLS:  (9)2
(5)0002  (6)K0SSUB.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

10.5 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

10.5.1 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.

10.5.2 Error list

An error list stores the error messages output when the list converter is started up.

[Output format]

Same as for the error list output by the assembler.

CHAPTER 11 EFFICIENT USE OF RA78K0S

This chapter describes how to use the RA78K0S efficiently.

11.1 Improving Operating Efficiency (EXIT Status Function)

When any of the programs of the RA78K0S 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)

```
ra78K0s –c9024 k0smain. -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
ra78K0s –c9024 k0ssub.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
lk78K0s k0smain.rel k0ssub.rel –ok0s.lmf -g
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
oc78K0s k0s.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
```

11.2 Preparing Development Environment (Environmental Variables)

The RA78K0S supports the following environmental variables for preparing the software development environment.

PATH:	Search path for execution format
INC78K0S:	Search path for include file (assembler only)
LIB78K0S:	Search path for library file (linker only)
TMP:	Path for creating temporary files
LANG78K;	Kanji (2-byte character) type specification

When developing programs, it is a good idea to create a subdirectory in which to collect all related files. This will make program development easier and more convenient.

[Example of use]

Contents of AUTOEXEC.BAT

```
;AUTOEXEC.BAT
Verify on
break on
PATH C:\BIN;C:\BAT;C:\RA78K0S;          ← (1)
SET INC78K0S = C:\RA78K0S\INCLUDE       ← (2)
SET LIB78K0S = C:\RA78K0S\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:\RA78K0S.
(2) The assembler retrieves include files from the directory C:\RA78K0S\INCLUDE.
(3) The linker retrieves library files from C:\RA78K0S\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 Windows installer was used for installation, the required environment variables are automatically set.

11.3 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.

11.4 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**.

[Example of use]

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

```
C>ra78k0s -c9024 k0smain.asm -lhRA78K0S_MAINROUTINE
```

This references K0SMAIN.PRN.

```
                                                    Date:xx xxx xxxx Page: 1
78K/0S Series Assembler Ex.xx  RA78K0S_MAINROUTINE
                                          |
                                          └── Title

Command: -c9024 k0smain.asm -lhRA78K0S_MAINROUTINE
Para-file:
In-file: K0SMAIN.ASM
Obj-file:K0SMAIN.REL
Prn-file:K0SMAIN.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                  *
       ⋮
```

11.5 Reducing Program Startup Time

11.5.1 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 (9024)          ] Control instructions
$ XREF

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

11.5.2 Using PM plus

Each program option in the RA78K0S is automatically stored in a project file (.PRJ) in PM plus. Because the stored options are used for a second or subsequent build (MAKE) operation, the necessity of specifying the options each time is removed.

11.5.3 Creating parameter files and subcommand files

When executing any of the RA78K0S's programs (assembler, linker, object converter and list converter), if all the necessary data will not fit on the command line, or if the same options are specified every time the program is executed, create a parameter file.

Also, subcommands can be registered in a subcommand file in the librarian. This makes object module library formation easy.

[Example of use 1]

Create a parameter file and perform assembly.

Contents of parameter file K0SMAIN.PRA

```
;parameter file
k0smain.asm -osample.rel -g
-psample.prn
```

Enter the following on the command line.

```
C>ra78k0s -fk0smain.pra
```

[Example of use 2]

Create a parameter file and perform assembly.

Contents of parameter file K0S.SLB

```
;
;library creation command
;
create k0s.lib
;
add k0s.lib k0smain.rel &
k0ssub.rel
;
exit
```

Enter the following on the command line.

```
C>lb78k0s <k0s.slb
```

11.6 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 RA78K0S 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 12 ERROR MESSAGES

This chapter explains the causes of error messages output by the assembler package (structured assembler, assembler, linker, object converter and librarian), and the action to be taken by the user.

12.1 Overview of Error Messages

Error messages output by the RA78K0S 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.

RA78K0S error messages are classified as follows.

Each RA78K0S error message is explained beginning on the next page.

- A0xx --- Command line analysis error
- A9xx --- File or system error
- A1xx --- Other abort error
- F2xx --- Statement specification error
- F3xx --- Expression error
- F4xx --- Symbol error
- F5xx --- Segment error
- F6xx --- Control instruction or macro error
- W7xx --- Any type of warning error

12.2 Structured Assembler Error Messages

Table 12-1. 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 12-1. 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 parameters '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 12-1. 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 instructions
	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 12-1. 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 12-1. 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.

12.3 Assembler Error Messages

Table 12-2. Assembler Error Messages (1/8)

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 128 bytes
	Cause	Object code exceeds 128 bytes per line in a source statement.
A112	Message	No processor specified
	Cause	Target device is not specified in the command line or in the source module file.
A114	Message	Local symbol name of asm statement must begin with ‘?L’ in C source
	Cause	A local symbol that did not start with ‘?L’ was described in #asm in the C source.
A115	Message	Too long source line
	Cause	The limit on the length of one line (2048 characters) has been exceeded.

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.
F204	Message	Illegal character
	Cause	An illegal character is specified in the source module.

Table 12.2. Assembler Error Messages (2/8)

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.

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 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 number
	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.
F316	Message	Operand out of range (addr16)
	Cause	Operand is outside the specifiable range (varies according to target device) for addr16.

Table 12-2. Assembler Error Messages (3/8)

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.

F401	Message	Illegal symbol for PUBLIC 'symbol name'
	Cause	This symbol cannot be declared PUBLIC.
F402	Message	Illegal symbol for EXTRN/EXTBIT 'symbol name'
	Cause	This symbol cannot be declared EXTRN/EXTBIT.
F403	Message	Can’t define PUBLIC symbol 'symbol name'
	Cause	This symbol already has a PUBLIC declaration and cannot be defined with a PUBLIC declaration.
	Action by user	A symbol defined with bit items other than saddr.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.
F406	Message	Can’t refer 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 specified 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 segments
	Cause	Number of segments defined exceeds limit (100).

Table 12-2. Assembler Error Messages (4/8)

F505	Message	Current segment is not exist
	Cause	The ENDS directive was written before the segment was made, or before the next segment was made immediately after the end of the segment.
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 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.

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).
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.

Table 12-2. Assembler Error Messages (5/8)

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 65,535 local symbols are replaced during macro development.
F625	Message	Unexpected ENDM
	Cause	An unexpected ENDM is found.
F626	Message	Can’t describe LOCAL 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.

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.

Table 12-2. Assembler Error Messages (6/8)

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.
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.

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.

Table 12-2. Assembler Error Messages (7/8)

A905	Message	Can't open overlay file 'file name'
	Cause	Overlay file cannot be opened.
	Action by user	Make sure the overlay file is in the same directory as the assembler execution format.
A906	Message	Illegal overlay file 'file name'
	Cause	Contents of overlay file are illegal.
A907	Message	Can't open object file 'file name'
	Cause	Object file cannot be opened.
	Action by user	Use a disk with an open area in its directory.
A908	Message	Can't open print file 'file name'
	Cause	Assemble list file cannot be opened.
	Action by user	Use a disk with an open area in its directory.
A909	Message	Can't open error list file 'file name'
	Cause	Error list file cannot be opened.
	Action by user	Use a disk with an open area in its directory.
A910	Message	Can't open temporary file 'file name'
	Cause	Temporary file cannot be opened.
	Action by user	Use a disk with an open area in its directory.
A911	Message	System error
	Cause	A system error has occurred.
	Action by user	Confirm the execution environment and execute assemble again.
A912	Message	Can’t set Control+C
	Cause	CTRL+C cannot be set because assemble execution has been aborted.
	Action by user	Confirm the execution environment and execute assemble again.
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 12-2. Assembler Error Messages (8/8)

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.

12.4 Linker Error Messages

Table 12-3. Linker Error Messages (1/5)

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 has 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 65,535/0FFFFH)
	Cause	A value greater than 65,535 (0FFFFH) is described in the directive.

Table 12-3. Linker Error Messages (2/5)

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 65,535 (0FFFFH).

F201	Message	Multiple segment definition 'segment name' in merge directive
	Cause	Segment specified in the merge directive is already registered (the same segment is attempted to specify allocation using multiple merge directives).
F202	Message	Segment type mismatch 'segment 1' in file 'segment 2' -ignored
	Cause	A segment with the same name as this segment but having the reallocation attributes of a different segment type is found.
A203	Message	Segment 'segment name' unknown segment type
	Cause	An error exists in the segment data of the input object module file (specification of link of output segments is incorrect).
F204	Message	Memory area/space 'name' not defined
	Cause	Memory area/space name specified in merge directive is not defined.
F205	Message	Name 'name' in directive has bad attribute
	Cause	An item that cannot be described in a segment name, memory area name or memory space name is described in the directive (for example, a memory space name is described where a memory area name is required).
F206	Message	Segment 'segment name' can't allocate to memory - ignored
	Cause	Segment cannot be allocated to memory (not enough memory area exists to allocate segment).
F207	Message	Segment 'segment name' has illegal segment type
	Cause	This segment type data is illegal.
F208	Message	Segment 'segment name' may not change attribute
	Cause	Attempted to change the link type in the directive for a segment created with the reallocation attribute 'AT xxxxH' specified during assemble, or created using the ORG directive.
F209	Message	Segment 'segment name' may not change arrangement
	Cause	Attempted to change the allocation address in the directive for a segment created with the reallocation attribute 'AT xxxxH' specified during assemble, or created using the ORG directive.
	Action by user	Do not specify the allocation address in the assembler for a segment whose link type is to be specified during link.
F210	Message	Segment 'segment name' is not exist - ignored
	Cause	Segment specified in the directive does not exist.
F211	Message	Bank type mismatch 'symbol name' in file 'file name' - ignored
	Cause	A mismatch exists in the specified symbol bank number.
	Action by user	Confirm that the bank number of the symbol is correct.

Table 12-3. Linker Error Messages (3/5)

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 or saddrp addressing operand must be even numbers).
F306	Message	A multiple of 4 value expected (segment ‘segment name’, address xxxxH type ‘addressing type’)
	Cause	The operand value used to determine the relocatable object code of the 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.

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.

Table 12-3. Linker Error Messages (4/5)

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 symbols written subsequent to the program entry address specification -A of the linker option do not exist in the public symbols.
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.
W410	Message	Multiple module name definition ‘module name’ in file ‘file 1’. First defined in file ‘file 2’
	Cause	The module name of object module file 1 and the module name of object module file 2 are the same.
W411	Message	Different REL type in file 'file name'
	Cause	The type version of the object module file is different.
	Action by user	Assemble or recompile with the latest version.
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.
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 (CC78K0S, ST78K0S, RA78K0S) 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.
F420	Message	File 'file name' has already had error(s)/warning(s) by 'tool name'
	Cause	An error message or warning message for each tool (CC78K0S, ST78K0S, RA78K0S) used until the link stage is output.
F425	Message	There are different function ID in same name “function name” (file ‘file name’)
	Cause	A function with the same name as the function declared as EXT-FUNC by the compiler has a different ID value.
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.

Table 12-3. Linker Error Messages (5/5)

A901	Message	Can’t open overlay file 'file name'
	Cause	Overlay file cannot be opened.
	Action by user	Make sure the overlay file is in the correct directory (a directory containing an execution program).
A902	Message	File 'file name' not found
	Cause	The specified library file cannot be opened.
A903	Message	Can't read input file 'file name'
	Cause	Object module file specified as an input file cannot be read.
A904	Message	Can't open output file 'file name'
	Cause	Output file cannot be opened.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to create output file.
A905	Message	Can’t create temporary file 'file name'
	Cause	Temporary file for symbol entry cannot be created.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to attempt to create temporary file.
A906	Message	Can’t write map file 'file name'
	Cause	Data cannot be written to the link list file.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to attempt to create link list file.
A907	Message	Can’t write output file 'file name'
	Cause	Data cannot be written to the load module file.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to attempt to create output file.
A908	Message	Can’t access temporary file 'file name'
	Cause	Temporary file cannot be written.
	Action by user	Check condition (open capacity, condition of media, etc.) of the disk used to attempt to create temporary file.
A909	Message	Can't read device file 'device file name'
	Cause	Device file corresponding to device specified by option -C or $PROCESSOR control instruction cannot be read.

12.5 Object Converter Error Messages

Table 12-4. Object Converter Error Messages (1/2)

A006	Message	File not found ‘file name’
	Cause	The specified input file does not exist.
	Action by user	When linked to the startup routine of the C compiler, files are output as [“Startup routine name”.lmf]. In this case, specify the output file name as [-o*.lmf] using a linker option.
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	When the source is written in C language, check whether the following cautions are applicable. <Cautions> • When using a local symbol, use a symbol that starts with the character string ?L (?L@01, ?L@sym, etc.), but make sure the symbol has 8 or less characters. Also, be sure not to externally define this symbol (PUBLIC declaration).
A107	Message	Can’t specify –U option for ROMless device
	Cause	The object complement option (-U) has been specified for a device with no internal ROM.

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 CSEG of the assembly source, either change the object to DS or write the DB/DW instruction in DSEG.

Table 12-4. 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.

12.6 Librarian Error Messages

Table 12-5. 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 file
	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 parameter '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 file.
A020	Message	Parameter file read error 'file name'
	Cause	An error occurred in reading a parameter file.

Table 12-5. Librarian Error Messages (2/3)

A021	Message	Memory allocation failed
	Cause	An error occurred in memory allocation.
A022	Message	Memory allocation failed
	Cause	Memory allocation has failed.
A023	Message	Illegal character ‘,’ before file name
	Cause	Necessary ',' exists before the input file.
A024	Message	Illegal character
	Cause	An illegal character or character string is found.
A025	Message	Qualifier is not unique.
	Cause	The abbreviation type of the modifier is not unique.
A026	Message	Ambigous input redirect.
	Cause	No file name is specified after '<', or '< Δ file name' is specified more than once.

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 file names overlap.
F109	Message	Public symbol already exists (symol: symbol name)
	Cause	An externally defined symbol name of the same name already exists in an updated library file or other input file.
F110	Message	File specification conflicted (file: file name)
	Cause	Specified input file name is same as output file name.
F111	Message	Illegal file format (file: file name)
	Cause	Format of an updated library file or other input file is incorrect.
F112	Message	Library file not found (file: file name)
	Cause	Specified library file is not found.

Table 12-5. Librarian Error Messages (3/3)

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 × 15 characters) is exceeded. Limit for length of 1 line in a subcommand (128 characters) is exceeded.
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.
A902	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.

12.7 List Converter Error Messages

Table 12-6. List Converter Error Messages (1/2)

A101	Message	File is not 78K/0S 'file name'
	Cause	Input file name is not a 78K0S file name.
W101	Message	Load module file is older than object module file 'load module file name, object module file name'
	Cause	A load module file is specified which is older than the object module file.
A102	Message	Load module file is not executable 'file name'
	Cause	Attempted to input a file other than a load module file, or attempted to convert a load module file created on an incompatible host machine.
W102	Message	Load module file is older than assemble module file 'load module file name, assemble list file name'
	Cause	A load module file is specified which is older than the assemble list file.
A103	Message	Load module file has relocation data 'file name'
	Cause	Address of load module file is not determined.
W103	Message	Assemble list has error statement 'file name'
	Cause	An error exists in the assemble list.
A104	Message	Object module file is executable 'file name'
	Cause	Object module file is in an executable format.
W104	Message	Segment name is not found in assemble list file 'segment name'
	Cause	Segment name of object module file is not found in assemble list.
A105	Message	Segment name is not found in load list file 'segment name'
	Cause	Segment name of object module file is not found in load module file.
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 12-6. 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

12.8 PM plus Error Messages

This section explains the error messages not described in PM plus's help. For details of other PM plus error messages, refer to PM plus's online help.

<1> Error messages displayed in DLL for structured assembler (ST78K0S)

!	Message	Cannot find ST78K0S.EXE shown in environment variable PATH.
	Cause	The ST78K0S.EXE execution format is not registered in the directory indicated by the PATH environment variable.
	Action by user	Put ST78K0S.EXE and other files related to the ST78K0S in the directory indicated by the PATH environment variable.
	Button	Click OK to close the message box.
×	Message	Not enough memory.
	Cause	There is insufficient memory.
	Action by user	Try again after closing other applications.
	Button	Click OK to close the message box.
×	Message	Cannot lock the memory.
	Cause	There may be insufficient memory, or the Windows system may be malfunctioning.
	Action by user	Try again after either closing other applications or restarting Windows.
	Button	Click OK to close the message box.

<2> Error messages displayed in DLL for assembler (RA78K0S)

!	Message	Cannot find RA78K0S.EXE shown in environment variable PATH.
	Cause	The RA78K0S.EXE execution format is not registered in the directory indicated by the PATH environment variable.
	Action by user	Put RA78K0S.EXE and other files related to the RA78K0S in the directory indicated by the PATH environment variable.
	Button	Click OK to close the message box.
×	Message	Not enough memory.
	Cause	There is insufficient memory.
	Action by user	Try again after closing other applications.
	Button	Click OK to close the message box.
×	Message	Cannot lock the memory.
	Cause	There may be insufficient memory, or the Windows system may be malfunctioning.
	Action by user	Try again after either closing other applications or restarting Windows.
	Button	Click OK to close the message box.

<3> Error messages displayed in DLL for linker (LK78K0S)

!	Message	Cannot find LK78K0S.EXE shown in environment variable PATH.
	Cause	The LK78K0S.EXE execution format is not registered in the directory indicated by the PATH environment variable.
	Action by user	Put LK78K0S.EXE and other files related to the LK78K0S in the directory indicated by the PATH environment variable.
	Button	Click OK to close the message box.
×	Message	Not enough memory.
	Cause	There is insufficient memory.
	Action by user	Try again after closing other applications.
	Button	Click OK to close the message box.
×	Message	Cannot lock the memory.
	Cause	There may be insufficient memory, or the Windows system may be malfunctioning.
	Action by user	Try again after either closing other applications or restarting Windows.
	Button	Click OK to close the message box.

<4> Error messages displayed in DLL for object converter (OC78K0S)

!	Message	Cannot find OC78K0S.EXE shown in environment variable PATH.
	Cause	The OC78K0S.EXE execution format is not registered in the directory indicated by the PATH environment variable.
	Action by user	Put OC78K0S.EXE and other files related to the OC78K0S in the directory indicated by the PATH environment variable.
	Button	Click OK to close the message box.
×	Message	Not enough memory.
	Cause	There is insufficient memory.
	Action by user	Try again after closing other applications.
	Button	Click OK to close the message box.
×	Message	Cannot lock the memory.
	Cause	There may be insufficient memory, or the Windows system may be malfunctioning.
	Action by user	Try again after either closing other applications or restarting Windows.
	Button	Click OK to close the message box.

APPENDIX A SAMPLE PROGRAMS

The following describes the sample program lists attached to the RA78K0S.

A.1 K0smain.asm

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

PUBLIC  MAIN, START
EXTRN   CONVAH
EXTRN   _@STBEG

DATA    DSEG    saddr
HDTSA:  DS  1
STASC:  DS  2

CODE    CSEG    AT 0H
MAIN:   DW  START

        CSEG
START:

    ;chip initialize
    MOVW    AX, #_@STBEG
    MOVW    SP, AX

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

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

    BR      $$

    END
```

A.2 K0ssub.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, [HL]
    ROR     A, 1
    ROR     A, 1
    ROR     A, 1
    ROR     A, 1
    AND     A, #0FH         ;hex upper code load
    CALL    !SASC
    MOV     B, A            ;store result

    XOR     A, A
    XCH     A, [HL]
    AND     A, #0FH         ;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(+7H)
SASC1:
    ADD     A, #30H         ;bias(+30H)
    RET

    END
```

A.3 test1.s

```
EXTRN   SEARCH, STABLE
EXTRN   _@STBEG
PUBLIC  MAIN, START
;*****************************************
;           String data search
;*****************************************
SDATA:
    DB  04,12H,34H,56H,78H
;
;
CODE    CSEG    AT 0H
START:  DW      MAIN

MAIN:
    MOVW    AX, #_@STBEG
    MOVW    SP, AX
    DE = #STABLE
    HL = #SDATA
    CALL  !SEARCH
    if_bit (!CY)
SLI:
        repeat
        until (forever)
    else
SERR:
        repeat
        until (forever)
    endif
    END
```

A.4 test2.s

```
#include   "testinc.s"

PUBLIC SEARCH

    CSEG
;*******************************************************
;*  Data search                                        *
;*    input    HL     search data address             *
;*             DE     table top address               *
;*    output   CY = 1 not find                        *
;*             CY = 0 find (DE<-table address)        *
;*******************************************************

SEARCH:
    while ([DE]!= #0) (A)
       BC = #0
       A = [DE]
       C = A
       PUSH    HL
       PUSH    DE
       while  ([DE]== [HL]) (A)
          DE++
          HL++
          if (C == #0) (A)

             POP DE
             POP HL
             CLR1  CY
             RET
          endif
          C--
       endw
       POP DE
       POP HL
       A =[DE]
       A += E
       E = A
       if (CY)
       D++
       endif
        endw
    SET1   CY
    RET
    END
```

A.5 testinc.s

```
PUBLIC  STABLE

;*********************************
;         Data table
;*********************************

    CSEG
STABLE:
    DB  03, 12H, 34H, 78H
    DB  04, 55H, 66H, 77H, 88H
    DB  05, 12H, 34H, 56H, 78H, 10H
    DB  03, 12H, 34H, 56H
    DB  04, 12H, 34H, 0AH, 78H
    DB  04, 12H, 34H, 56H, 70H
    DB  04, 12H, 34H, 56H, 78H
    DB  01, 0ABH
    DB  02, 34H, 78H
    DB  00
```

A.6 st.bat

```
        echo off
        cls
        set      LEVEL=0

        if "%1" == "" goto ERR_BAT

        st78k0s -C%1 test1.s
        ra78k0s test1.asm
        if errorlevel 1 set LEVEL=1
        st78k0s -C%1 test2.s
        ra78k0s test2.asm
        if errorlevel 1 set LEVEL=1
        if %LEVEL% == 1 echo Assemble error !!
        if %LEVEL% == 1 goto END

        cls
        lk78k0s test1.rel test2.rel -s -otest.lmf -ptest.map
        if errorlevel 1 echo Link error !!
        if errorlevel 1 goto END

        cls
        oc78k0s test
        if errorlevel 1 echo Object conversion error !!
        if errorlevel 1 goto END

        cls
        set LEVEL=0
        lc78k0s -ltest.lmf -rtest1.rel test1.prn
        if errorlevel 1 set LEVEL=1
        lc78k0s -ltest.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
```

APPENDIX B NOTES ON USE

The device file is necessary to execute the RA78K0S. The device file is not included in the RA78K0S package and must be obtained separately.

(1) Device file

The device file is necessary to execute the RA78K0S. The device file is not included in the RA78K0S package, and must be obtained separately.

(2) Memory directive

The default memory area name of each device cannot be erased. The size of the unused default memory area must be 0.

For the default memory area name refer to the **Notes on Use** of the device file of each device.

Note that because some segments are allocated to the default area, care is needed when changing an area name.

(3) Debug operation

When debug data is output by the C compiler and structured assembler preprocessor, and then compiled or undergoes structured assembly, when assembling the output assemble source, specify the -NGA option to avoid outputting debug data. If debug data is output, debugging may not be able to be performed at the C compiler or structured assembler source level.

(4) Memory initialization directives

Codes are output even if a memory initialization directive (DW or DB) is written in a data segment (DSEG).

When ordering ROM code, an error will occur if code exists in an address other than an internal ROM address.

(5) SFR names and EQU definition

Although an SFR name can be specified for the operand of the EQU quasi-directive, when an SFR name outside of the saddrr area is specified as PUBLIC, an assemble error will occur.

(6) CC78K0S

Several points must be noted when executing C-source level debugging by using the assembled assembler source output by the CC78K0S.

For details, refer to the **document supplied with the C compiler package (Notes on Use)**.

(7) ID78K0S-NS and SM78K0S

When performing debugging with the ID78K0S-NS and SM78K0S, use the number of symbols and source lines that are within the limits of the ID78K0S-NS and SM78K0S.

For details, refer to the document supplied with the debugger/simulator package (Notes on Use).

(8) When using a network

If a directory that creates temporary files is placed in a file system that is shared in a network, a file conflict may occur, causing abnormal operations. Avoid this conflict by setting appropriate options and environment variables.

(9) Object converter operation specifications

When start is specified using the object converter option –U, addresses are filled starting from the smallest address where the start address or code is located. Filling is not carried out in the SFR area (FF00H to FFFFH).

Syntax: -U complement-value [, [start], size]
Values in 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.65
2	Word symbol character specification	-SC character	Specifies the final character of a word symbol name.	Independent	-SCP	p.66
3	Symbol definition specification	-D symbol [numerical-value]	Specifies the symbol given to the #IFDEF directive, etc.	Independent	None	p.67
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.68
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 'INC78K0S'	p.69
6	Secondary source file specification	-O [file-name]	Specifies the secondary source file name.	Independent	-O [input-file-name.ASM]	p.70
7	Error list file specification	-E [file-name]	Outputs the error list file.	Independent	-E [input-file-name.EST]	p.71
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.72
9	Debug data output specification	-GS	Specifies the output of structured assembler source level debug data.	If both options -GS and -NGS are specified at the same time, the option specified last takes precedence.	-GS	p.73
		-NGS	Makes option -GS unavailable.			

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.74
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. 75
		-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 started up the ST78K0S.	p.76
13	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.77

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.91
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.92
		-NO	Specifies that no object module file is output.			
3	Forced object module file output specification	-J	Forcibly outputs the object module file.	If both options -J and -NJ are specified at the same time, the option specified last takes precedence.	-NJ	p.93
		-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.94
		-NG	Makes option -G unavailable.			
		-GA	Specifies that source debugging data is to be added to an object module file.	If both options -GA and -NGA are specified at the same time, the option specified last takes precedence.	-GA	p.95
		-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 (INC78K0S)	p.96
6	Assemble list file output specification	-P [file-name]	Outputs an assemble list 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.97
		-NP	Does not output an assemble list file.			
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. 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. If options -NKA, -NKS and -NKX are all specified, option -P is unavailable.	-KA	p.98
		-NKA	Does not output an assemble list file.			
		-KS	Outputs an assemble list followed by a symbol list into an assemble list file.		-NKS	p.100
		-NKS	Makes option -KS unavailable.			
		-KX	Outputs an assemble list followed by a cross-reference list into an assemble list file.		-NKX	p.101
		-NKX	Makes option -KX unavailable.			

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 an assemble list file.	If option -NP is specified, option -LW is unavailable.	-LW132	p.103
		-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.105
		-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.106
		-LT [number-of-characters]	Specifies a number of characters to be developed in a tab.	If option -NP is specified, option -LT is unavailable. 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.	-LT8	p.110
		-LF	Inserts a form feed (FF) code at the end of an assemble list file.		-NLF	p.112
		-NLF	Makes 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.113
		-NE	Makes 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.114
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.115
12	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.116
		-ZE	Kanji described in the comment is interpreted as EUC code.			
		-ZN	Characters described in the comment are not interpreted as kanji.			
13	Device file search path specification	-Y path-name	Reads a device file from the specified path.	Independent	<..\dev> (for the RA78K0S startup path)	p.117
14	Symbol definition specification	-D symbol-name [=numerical-value] [, symbol-name [=numerical-value] …]	Defines a symbol.	Independent	None	p.118
15	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.118

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.140
		-NO	Does not output a load module file.			
2	Forced load module file output specification	-J	Forcibly outputs a load module file.	If both options -J and -NJ are specified at the same time, the option specified last takes precedence.	-NJ	p.141
		-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.142
		-NG	Makes option -G unavailable.			
4	Specification of generation of stack decision symbols	-S [area-name]	Automatically generates stack decision public symbols.	If both options -S and -NS are specified at the same time, the option specified last takes precedence.	-NS	p.143
		-NS	Makes option -S unavailable.			
5	Directive file specification	-D file-name	Specifies a particular file to be input as a directive file.	Independent	None	p.145
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.146
		-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. If options -NKM, -NKP and -NKL are all specified, option -P is unavailable. If option -NKM is specified, option -KD becomes unavailable. If both options -KD and -NKD, both -KP and -NKP, or both -KL and -NKL are specified at the same time, the option specified last takes precedence. If option -NG is specified, the public symbol list and local symbol list cannot be output even if option -KP or -KL is specified.	-KM	p.147
		-NKM	Makes option -KM unavailable.			
		-KD	Outputs a link directive file into a link list file.		-KD	p.149
		-NKD	Makes option -KD unavailable.			
		-KP	Outputs a public symbol list into a link list file.		-NKP	p.150
		-NKP	Makes option -KP unavailable.			
		-KL	Output a local symbol list into a link list file.		-NKL	p.152
		-NKL	Makes option -KL unavailable.			

No.	Classification	Format	Function	Relation to Other Options	Interpretation When Omitted	Ref. Page
8	Link list file 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.154
		-LF	Inserts a form feed (FF) code at the end of a link list file.	If both options -LF and -NLF are specified at the same time, the option specified last takes precedence. If option -NP is specified, the option specified last takes precedence.	-NLF	p.156
		-NLF	Makes option -LF 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.157
		-NE	Makes option -E unavailable.			
10	Library file specification	-B file-name	Inputs a specific file as a library file.	Independent	None	p.158
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 'LIB78K0S'	p.159
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.160
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	p.161
14	Device file search path specification	-Y path-name	Reads a device file from the specified path.	Independent	<..\dev> for the LK78K0S startup path	p.162
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.163
16	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.164

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.182
		-NO	Does not output a HEX format object module file.			
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 H1 to H15 for extended space)	p.183
		-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 -R and -NR are specified at the same time, the option specified last takes precedence. If option -NO is specified, option -R becomes unavailable.	-NR	p.184
		-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, option -U becomes unavailable.	None	p.185
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.187
		-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.188
7	Device file search path specification	-Y path-name	Reads a device file from the specified path.	Independent	<..\dev> for the OC78K0S startup path	p.189
8	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.190

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. If both options -LF and -NLF are specified at the same time, the option specified last takes precedence.	-LW132	p.203
		-LL [number-of-lines]	Changes the number of lines that can be printed in 1 page in a list file.		-LL66	p.204
		-LF	Inserts a form feed (FF) code at the end of a list file.		-NLF	p.205
		-NFL	Makes option -LF unavailable.			
2	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	p.206
3	Device file search path specification	-Y path-name	Reads a device file from the specified path.	Independent	<..\dev> (for the LB78K0S startup path)	p.207
4	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.208

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]	Inputs an object module file.	Independent	-R [assemble-list-file-name.REL]	p.230
2	Load module file input specification	-L [file-name]	Inputs a load module file.	Independent	-L [assemble-list-file-name.LMF]	p.231
3	Absolute assemble list file output specification	-O [file-name]	Outputs an absolute assemble list file.	Independent	-O [assemble-list-file-name.P]	p.232
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.233
		-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.234
6	Help specification	--	Displays a help message on the display (console).	All other options are unavailable.	No display	p.235

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∆library-file-name [∆transaction]	Creates a new library file.	C	p.210
2	ADD	ADD∆library-file-name ∆transaction	Adds a module to a library file.	A	p.211
3	DELETE	DELETE∆library-file-name∇ (∇module-name [∇, ···]∇)	Deletes a module from a library file.	D	p.212
4	REPLACE	REPLACE∆library-file-name ∆transaction	Replaces one module with another in a library file.	R	p.213
5	PICK	PICK∆library-file-name∇ (∇module-name [∇, ···]∇)	Retrieves a specified module from an existing library file.	P	p.214
6	LIST	LIST[∆option]∆library-file-name [∇(∇module-name [∇, ···]∇)]	Outputs data on modules in a library file.	L	p.215
7	HELP	HELP	Displays a help message on the display (console).	H	p.216
8	EXIT	EXIT	Exits the librarian.	E	p.217

APPENDIX E INDEX

[M]

[N]

[O]

[P]

[R]

[S]

[T]

[U]

[W]

[Y]

[Z]

[Symbol]