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

RA78K0R Ver. 1.20

Assembler Package

Operation

Target Devices
78K0R Microcontrollers

Document No. U18547EJ1V0UM00 (1st edition)
Date Published October 2007

[MEMO]

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

- **The information in this document is current as of October, 2007. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".
  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.
  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.
  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).
  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

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

(Note)
(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02. 11-1

[MEMO]

INTRODUCTION

This manual is intended to give users who wish to develop software using the RA78K0R an understanding of the functions of each program in the RA78K0R Series Assembler Package (hereafter referred to as "the RA78K0R") and of the correct methods of using the package.
This manual does not cover the expressions of directives and source programs or language used in the RA78K0R. Therefore, before reading this manual, read the **RA78K0R Ver. 1.20 Assembler Package Language User's Manual (U18546E)** (hereinafter referred to as "the Language Manual").
The contents of this manual are intended for use with Ver. 1.20 or later of the RA78K0R.

[Target Readers]
The RA78K0R is intended for users who understand the functions and instructions of the microcontroller for which software is being developed (78K0R Microcontroller).

[Organization]
This manual consists of the following eleven chapters and appendixes:

CHAPTER 1 GENERAL
Outlines the role of the RA78K0R in microcontroller software development and the features of the RA78K0R.

CHAPTER 2 PRODUCT OUTLINE AND INSTALLATION
Explains the program file names and operating environment provided by the RA78K0R.

CHAPTER 3 EXECUTION PROCEDURE OF RA78K0R
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 RA78K0R should read this chapter.

CHAPTER 4 ASSEMBLER

CHAPTER 5 LINKER

CHAPTER 6 OBJECT CONVERTER

CHAPTER 7 LIBRARIAN

CHAPTER 8 LIST CONVERTER

CHAPTER 9 PROGRAM OUTPUT LIST
Explains the formats of the lists output by each program.

CHAPTER 10 EFFICIENT USE OF RA78K0R
Introduces some measures for optimum utilization of the RA78K0R.

CHAPTER 11 ERROR MESSAGES
Explains the error messages output by each program.

APPENDIXES Introduce a list of program options, a list of sample programs, and a list of notices on using the RA78K0R.

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

[How to Read This Manual]

Those using an assembler for the first time are encouraged to read from **CHAPTER 1 GENERAL** of this manual.
Those who have a general understanding of assembler programs may skip this chapter.
Before using the RA78K0R, read **CHAPTER 3 EXECUTION PROCEDURE OF RA78K0R**.
After you have become familiar with the operation of each program, you can proceed to utilize the lists in the **APPENDIXES**.

[Conventions]

The following symbols and abbreviations are used throughout this manual:

⋮:	Indicates that the same expression is repeated.
[ ]:	Item(s) in brackets can be omitted.
' ':	Characters enclosed in ' ' (quotation marks) will be listed as they appear.
< >:	Names of dialog boxes and Windows
" ":	Characters enclosed in " " (double quotation marks) are titles of chapters, paragraphs, sections, diagrams or tables to which the reader is asked to refer.
___:	Indicates an important point, or characters that are to be input in a usage example.
□:	Indicates one blank space.
∆:	Indicates one or more blank or TAB.
∇:	Indicates zero or more blanks or TABs (i.e. blanks may be omitted).
/:	Indicates a break between characters.
~:	Indicates continuity.
[↵]:	Indicates pressing of the Return key.
Note:	Indicates a footnote for item marked with **Note** in the text.
Caution:	Indicates information requiring particular attention.
Remark:	Indicates supplementary information.

[Related Documents]

The documents related to this manual are listed below.

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

Document related to development tools (user's manuals)

Document Name		Document No.
RA78K0R Ver. 1.00 Assembler Package	Operation	This manual
	Language	U18546E
CC78K0R Ver. 1.00 C Compiler	Operation	U18549E
	Language	U18548E
SM+ System Simulator	Operation	U18010E
PM+ Ver. 6.30 Project Manager		U18416E
ID78K0R-QB Ver. 3.20 Integrated Debugger	Operation	U17839E

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

[MEMO]

CONTENTS

LIST OF FIGURES

Figure No. Title , Page

LIST OF TABLES

Table No. Title , Page

CHAPTER 1 GENERAL

This chapter describes the role of the RA78K0R in microcontroller software development and the features of the RA78K0R.

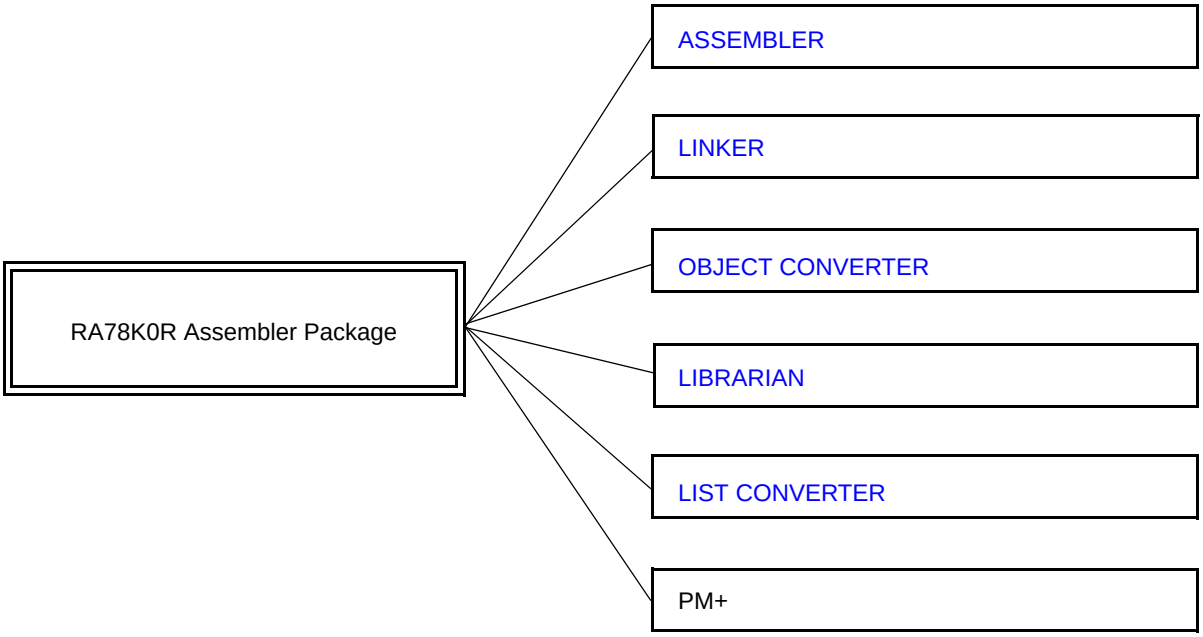
1.1 Assembler Overview

The RA78K0R Assembler Package (hereafter referred to as "the RA78K0R") is a generic term for a series of programs designed to translate source programs coded in the assembly language for 78K0R microcontrollers into machine language coding.

The RA78K0R contains 5 programs: Assembler, Linker, Object Converter, Librarian, and List Converter.

In addition, a PM+ that helps you perform a series of operations including editing, compiling/assembling, linking, and debugging your program on Windows® is also supplied with the RA78K0R.

Figure 1-1 RA78K0R Assembler Package



1.1.1 Assembler

An assembly language is the most fundamental programming language for a microcontroller.

Programs and data are required for the microprocessor in a microcontroller to do its job. These programs and data must be written by users to the memory of the microcontroller.

The programs and data handled by the microcontroller are collections of binary numbers called machine language.

For users, 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 users to comprehend. The basic programming language system that uses this symbolic coding is called an assembly language.

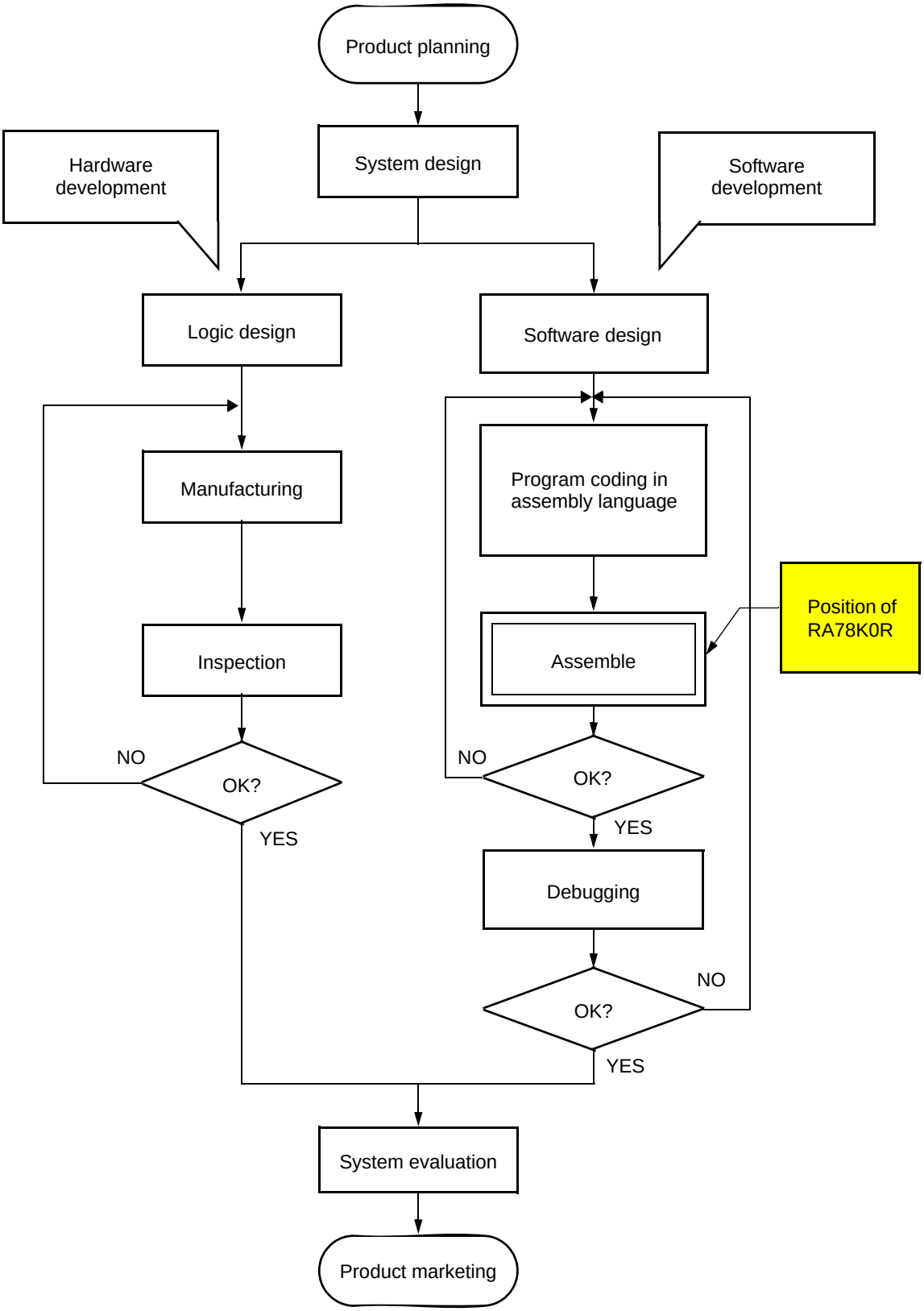
Since machine language is the only programming language in which a microcontroller can handle programs, however, another program is required that translates programs created in assembly language into machine language. This program is called an assembler.

Figure 1-2 Flow of Assembler

1.1.2 Development of microcontroller-related products and the role of RA78K0R

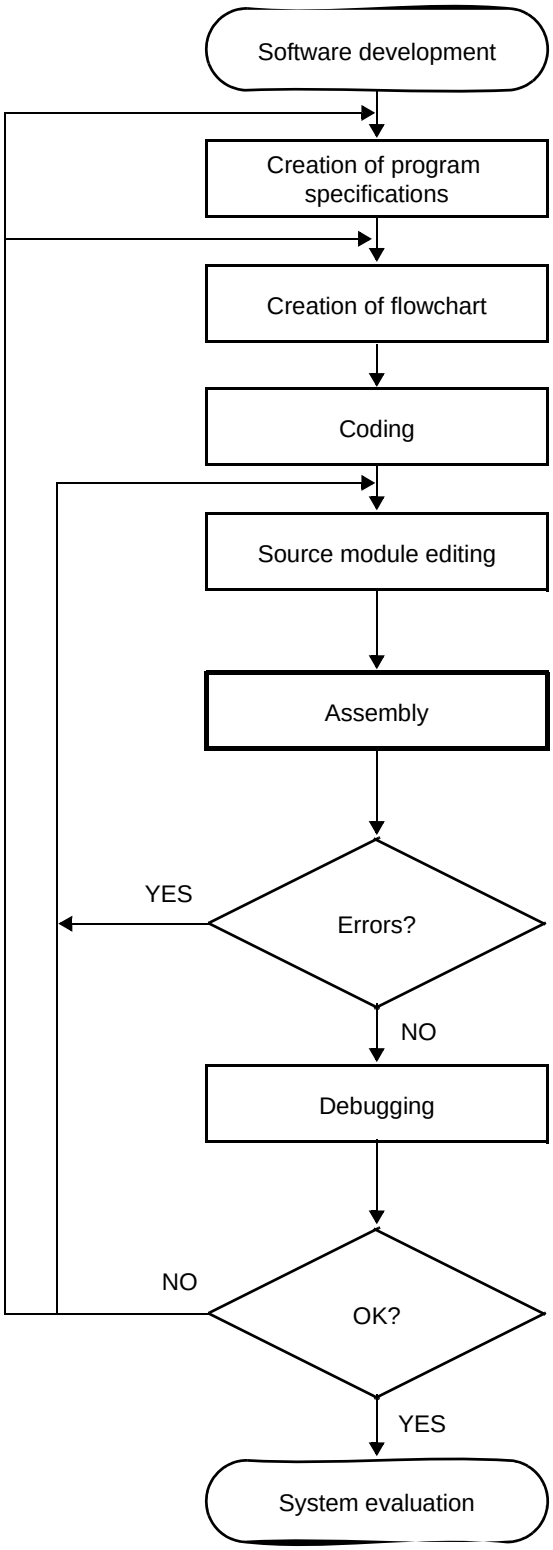
The following figure illustrates the position of "assemble" in the product development process.

Figure 1-3 Development Process of Microcontroller-Applied Products



A more detailed explanation of the software development process appears in the following figure.

Figure 1-4 Software Development Process



... This uses the RA78K0R assembly language.

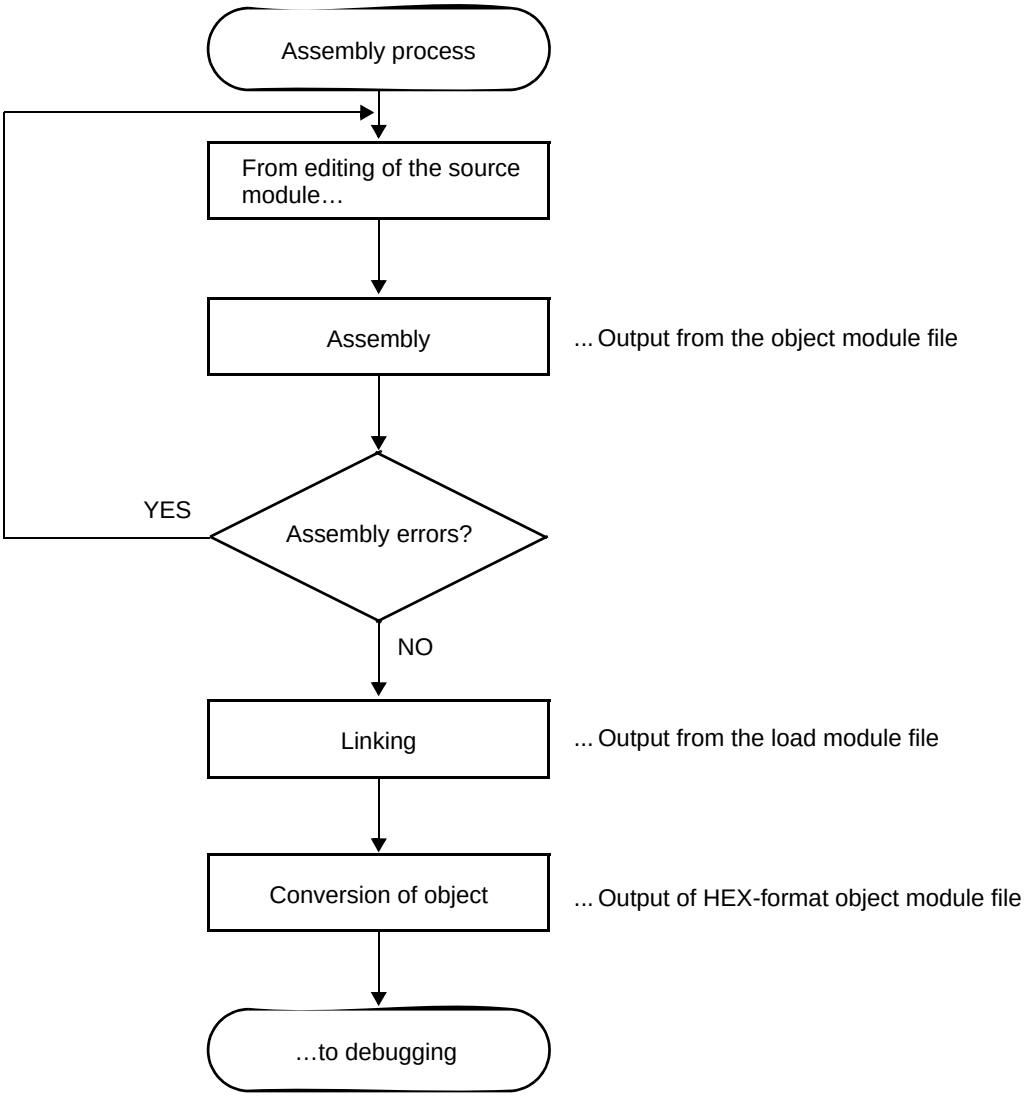
... An editor is used to create a source module file.

... An object module file is created.

... The operation of the software is checked using a hardware debugger such as an in-circuit emulator.

The RA78K0R is then applied to the assembly process.

Figure 1-5 RA78K0R Assembly Process



1.1.3 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.

Assemblers are broadly classified into "absolute assemblers" and "relocatable assemblers" depending on the method for determining to which addresses machine language instructions are relocated. The RA78K0R employs a relocatable assembler.

- 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 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 as a single entity, the program becomes complicated, making analysis and maintenance of the program difficult. 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.

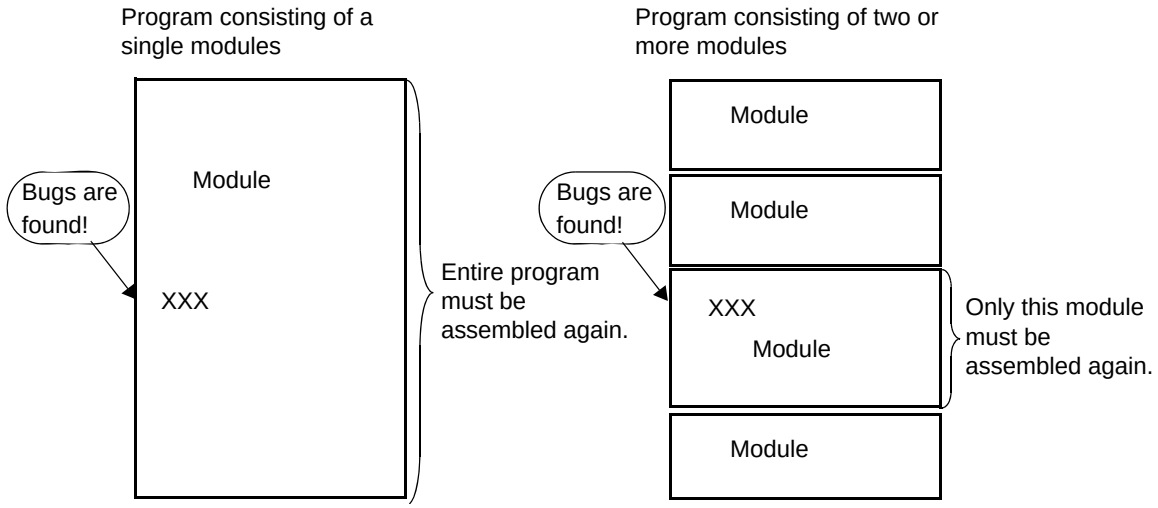
A relocatable assembler is an assembler suitable for modular programming, which has the following advantages:

(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.

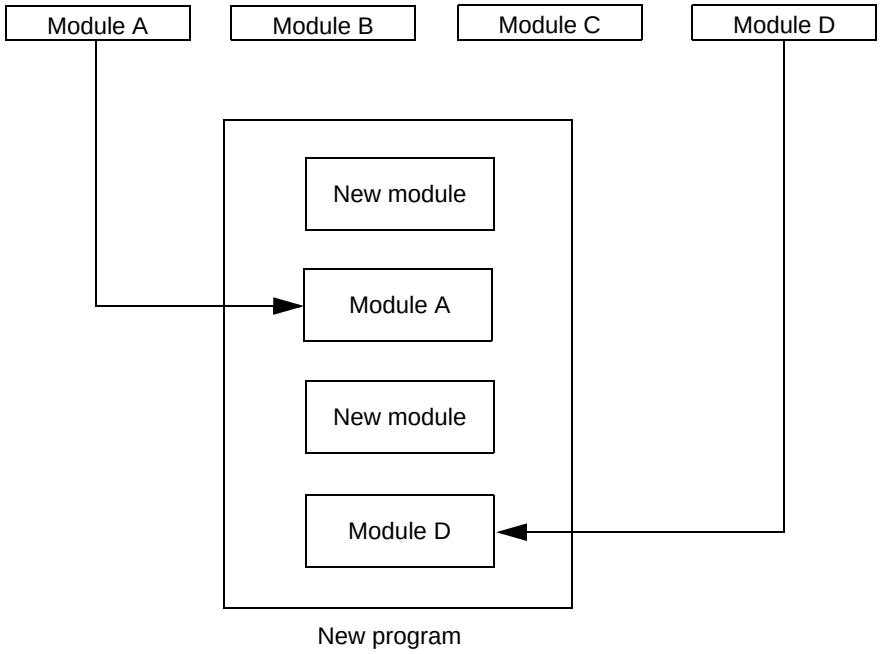
Figure 1-6 Reassembly for Debugging



(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

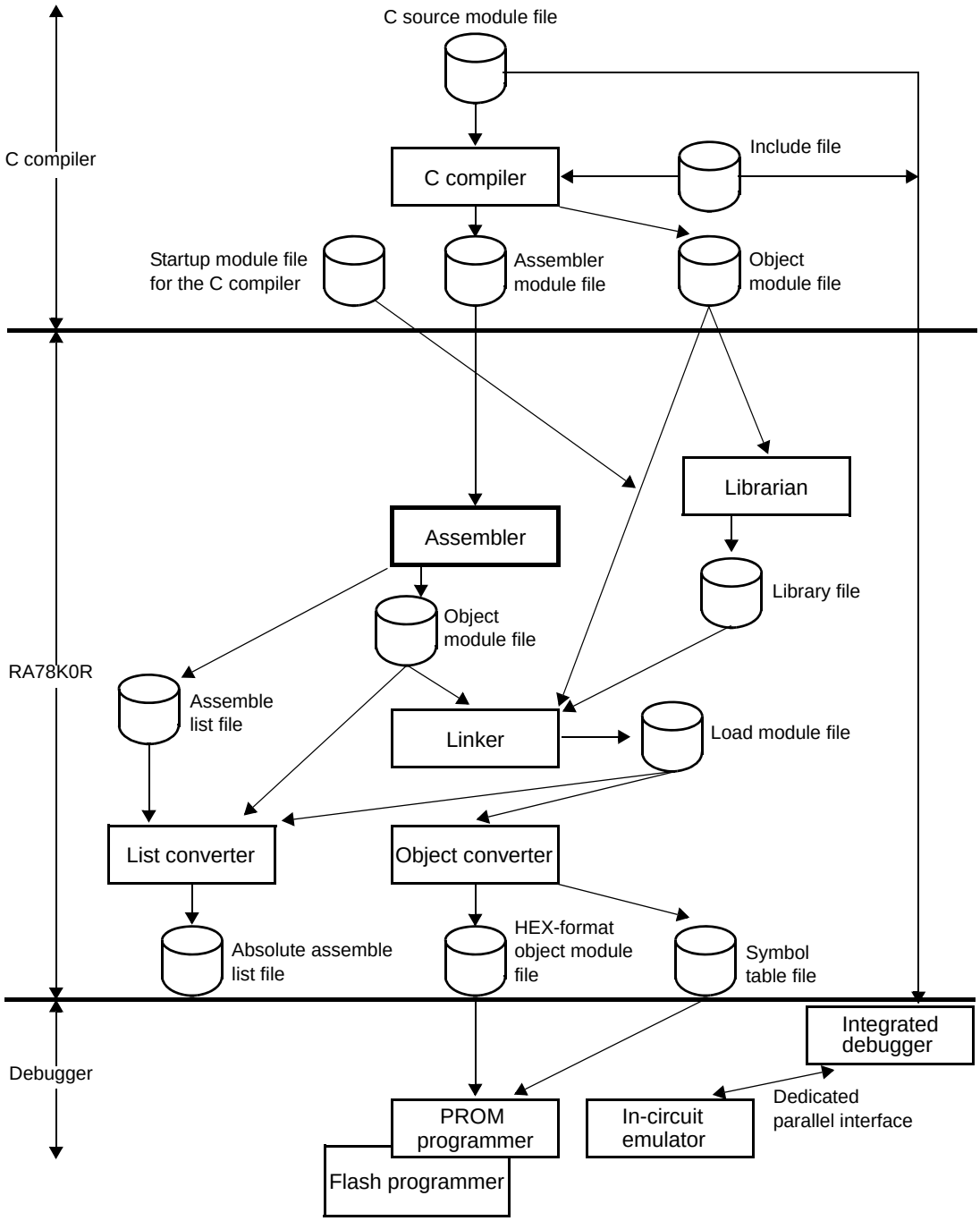


1.2 Overview of Features of RA78K0R

The procedure for developing general programs appears in the following table. 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 "RA78K0R". the assembler program is referred to as the "assembler".

Figure 1-8 Procedure for Program Development Using RA78K0R



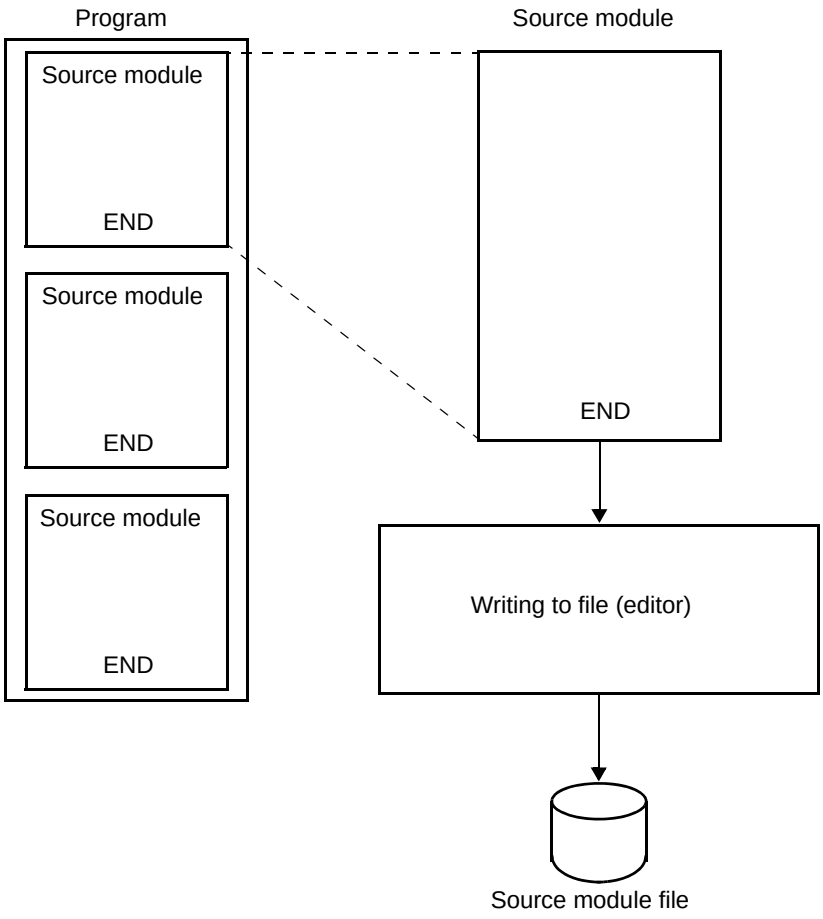
1.2.1 Creating a source module file using an editor

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.

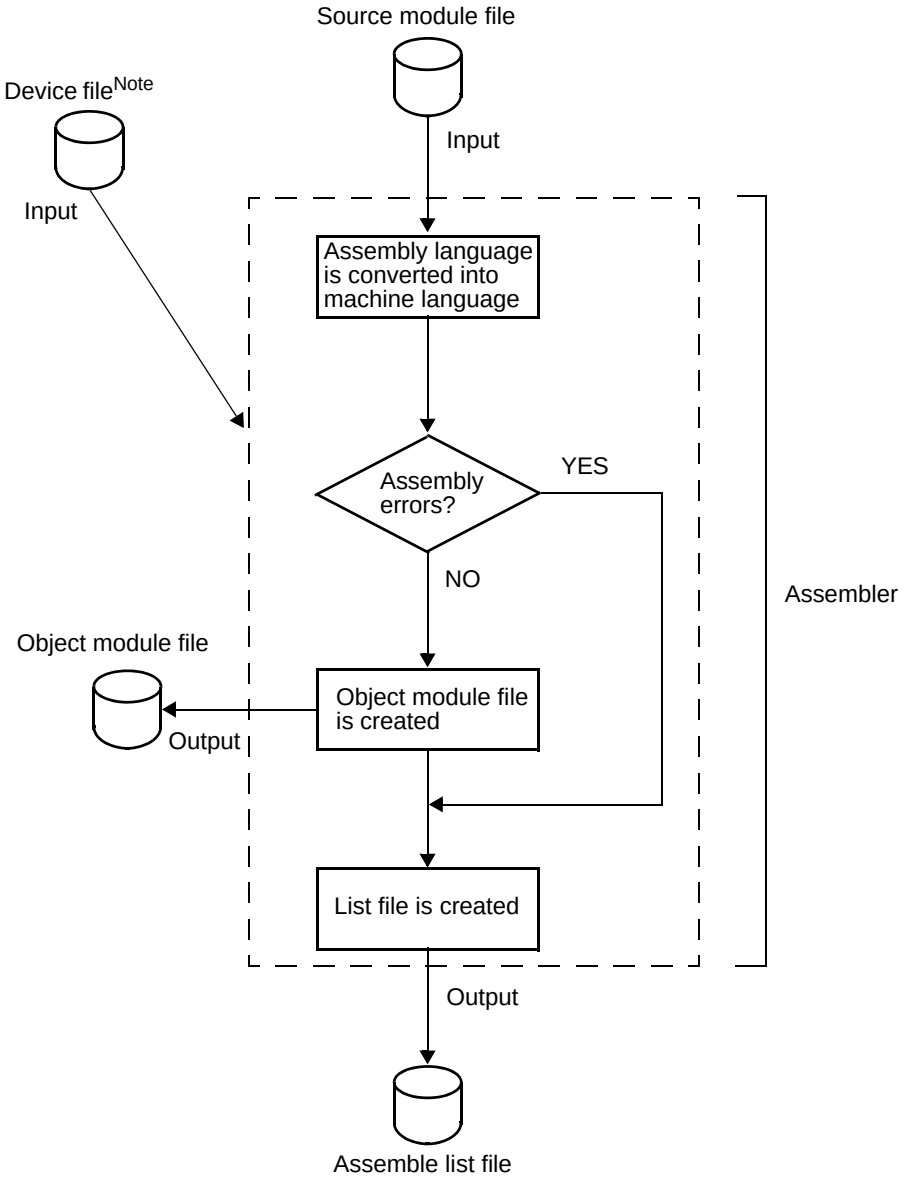
Figure 1-9 Creating Source Module File



1.2.2 Assembler

The assembler is a program which inputs the device file and 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 1-10 Function of Assembler



Note Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

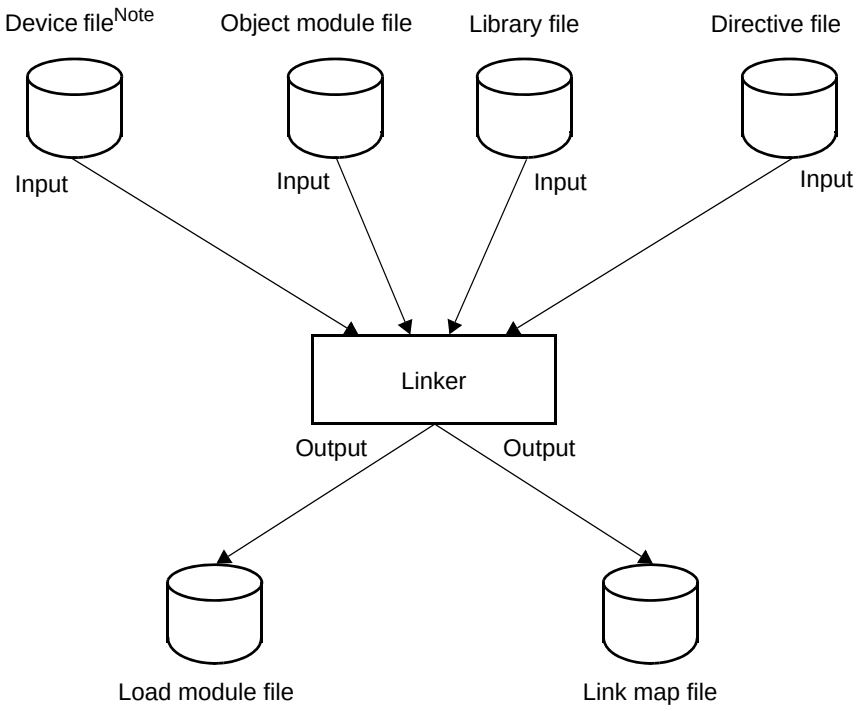
http://www.necel.com/micro/ods/eng/

1.2.3 Linker

The linker inputs the device file and 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 1-11 Function of Linker



Note Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

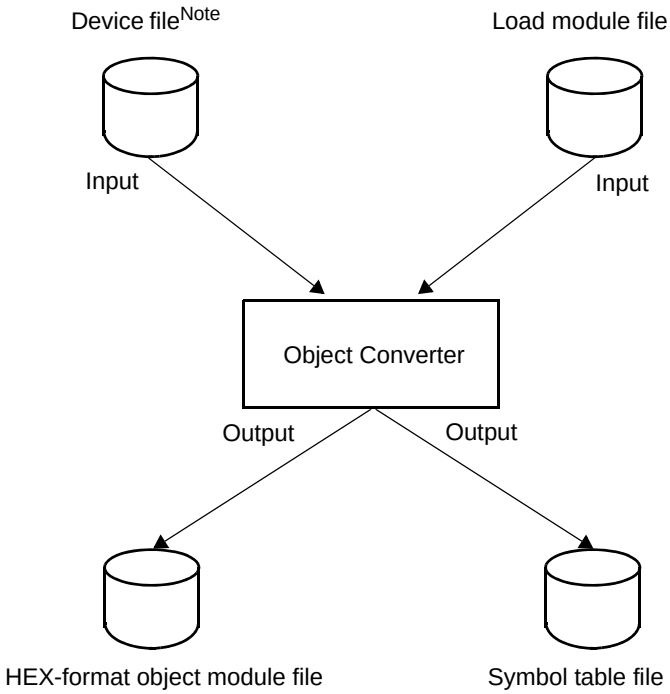
http://www.necel.com/micro/ods/eng/

1.2.4 Object converter

The object converter inputs the device file and 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 1-12 Function of Object Converter



Note Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

http://www.necel.com/micro/ods/eng/

1.2.5 Librarian

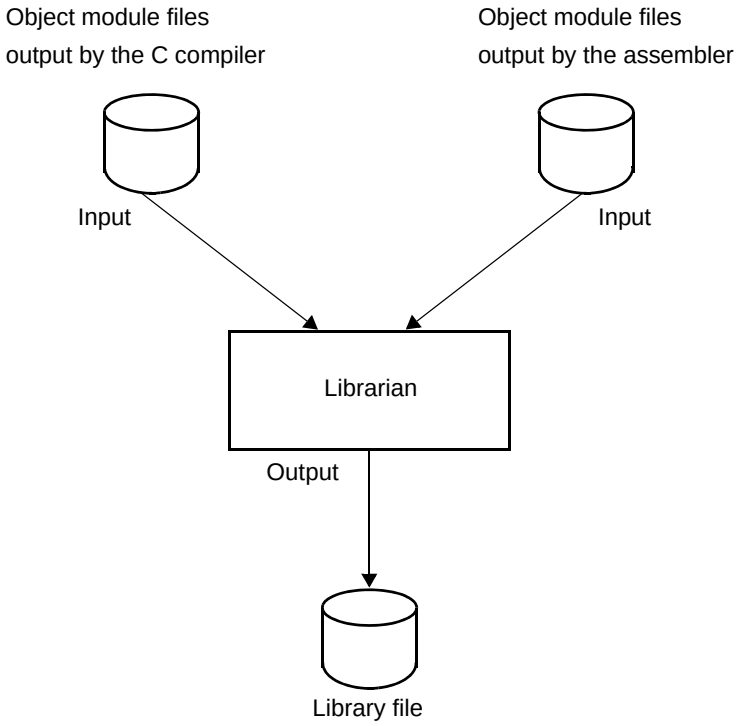
For convenience and ease of use, a general-purpose module with a clear interface may be stored in a library. By creating a library, multiple object modules can be stored in a single file, making them easy to handle.

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.

The librarian inputs the object module file output by the compiler and the assembler and converts the library file.

Figure 1-13 Function of Librarian

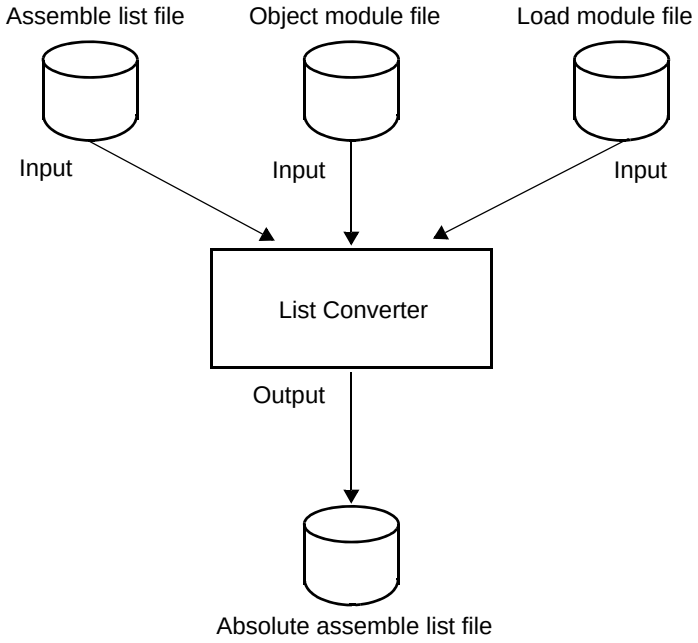


1.2.6 List converter

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 1-14 Function of List Converter



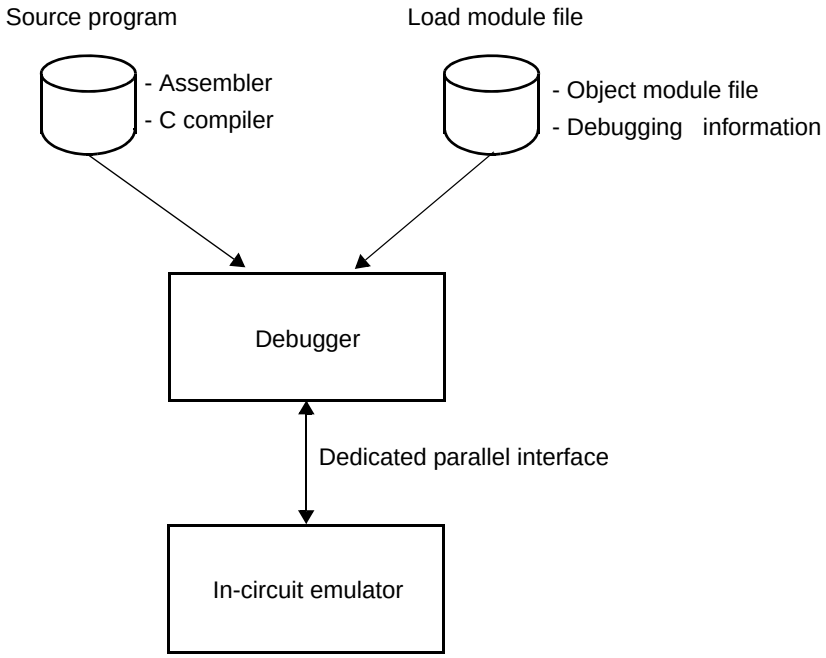
1.2.7 Debugger

The debugger for the 78K0R 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 RA78K0R.

Figure 1-15 Function of Debugger



1.3 Reminders Before Program Development

Refer to the following before beginning program development.

1.3.1 Quantitative limits for RA78K0R

(1) Quantitative limits for assembler

Item	Quantitative Limits
Number of symbols (local + public)	65,535 symbols
Number of symbols that can be output to cross reference list	65,534 symbols[Note 1]
Maximum marco body size referenced by one marco	1 M bytes
Total size of all macro bodies	10 M bytes
Number of segments in 1 file	256 segments
Macros and include specifications in 1 file	10,000
Macros and include specifications in 1 include file	10,000
Relocation information[Note 2]	65,535 items
Line number information	65,535 items
BR/CALL quasi directives in 1 file	32,767 directives
Numbers of characters on 1 line of source code	2,048 characters[Note 3]
Symbol length	256 characters
Number of switch name definitions[Note 4]	1,000
Character length of switch name[Note 4]	31 characters
Character length of segment name	8 characters
Character length of module name (NAME quasi directive)	256 characters
Number of virtual parameters in MACRO quasi directive	16 parameters
Number of actual parameters in macro reference	16 parameters
Number of actual parameters in IRP quasi directive	16 parameters
Number of local symbols in macro body	64 symbols
Total number of local symbols in expanded macro	65,535 symbols
Nesting levels in macro (macro reference, REPT quasi directive, IRP quasi directive)	8 levels
Number of characters specifiable by TITLE control instruction, the -lh option	60 characters[Note 5]
Number of characters specifiable by SUBTITLE control instruction	72 characters
Include file nesting levels in 1 file	8 levels
Conditional assembly nesting levels	8 levels

Item	Quantitative Limits
Number of include file paths specifiable by the -i option	64 paths
Number of symbols definable by the -d option	30 symbols

Note 1 Excluding the number of module names and section names.
Memory is used. If there is no memory, a file is used.

Note 2 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.

Note 3 Including CR and LF codes. If more than 2048 characters are written on one line, a warning message is output and the 2049th character and those that follow are ignored.

Note 4 The switch name is set as true/false by the SET/RESET quasi directive and is used by $IF, etc.

Note 5 If the maximum number of characters that can be specified in one line of the assemble list file ("X") is 119, this figure will be "X - 60" or less.

(2) Quantitative limits for linker

Item	Quantitative Limits
Number of symbols (local + public)	65,535 symbols
Line number information of the same segment	65,535 items
Number of segments	65,535 segments[Note 1]
Input modules	1,024 modules
Character length of memory area name	256 characters
Number of memory areas	100 areas[Note]
Number of library files specifiable by the -b option	64 files
Number of include file paths specifiable by the -i option	64 paths

Note Including those defined by default.

1.4 Features of RA78K0R

The RA78K0R 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

"BR" and "CALL" are available as automatic branch instruction selection directives.

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/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 RA78K0R 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 below.

Table 2-1 Supply Medium of Assembler Package

OS	Supply Medium
Windows 2000/XP Professional /XP Home Edition	CD-ROM

2.2 Installation

The procedure for installing to the host machine the files provided in the RA78K0R's supply media is described below.

(1) Starting up Windows

Power on the host machine and peripherals and start Windows.

(2) Set supply media

Set the RA78K0R's supply media in the appropriate drive (CD-ROM drive) of the host machine. The setup programs will start automatically. Perform the installation by following the messages displayed in the monitor screen.

Caution If the setup program does not start automatically, execute SETUP.EXE in the RA78K0R\DISK1 folder.

(3) Confirmation of files

Using Windows Explorer, etc., check that the files contained in the RA78K0R's supply media have been installed to the host machine.

For the details of each folder, refer to "2.4 Folder Configuration".

2.3 Installation of Device Files

Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

http://www.necel.com/micro/ods/eng/

Use the "device file installer" to install the device files. The "device file installer" is installed at the same time as the RA78K0R.

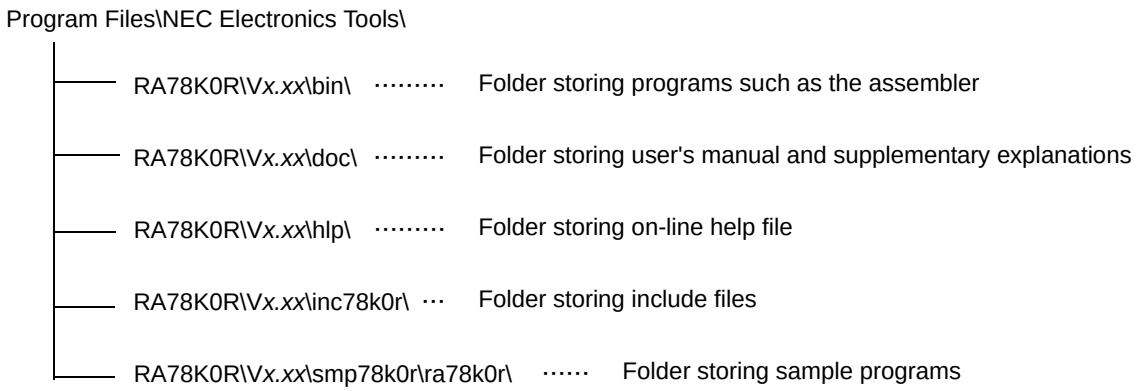
2.4 Folder Configuration

The standard folder displayed during installation is "Program Files\NEC Electronics Tools\" on the drive where Windows is installed. The configuration of the install folder is as shown below. The drive and install folder may be changed during installation.

Install PM+ and the RA78K0R in the same folder.

The explanations in this manual assume installation to the standard folder "Program Files\NEC Electronics Tools\" according to the installer.

Figure 2-1 Folder Configuration



2.5 File Organization

The table below lists the contents of each folder.

The folder structure and file organization are the ones obtained when the installer was used.

Table 2-2 File Organization

Folder Name	File Name	Description
RA78K0R\V*x.xx*\bin\	ra78k0r.exe	Assembler
	lk78k0r.exe	Linker
	oc78k0r.exe	Object converter
	lb78k0r.exe	Librarian
	lc78k0r.exe	Librarian
	lb78k0re.exe	Interface tool between library and DLL of PM+ environment
	lb78k0rp.exe	Standalone start up library
	ra78k0r.is*	File used by assembler
	*78k0rp.dll	DLL tool for PM+
	*78k0r.hlp	Help file for starting command line (text file)
RA78K0R\V*x.xx*\doc\	*.pdf	User's manual and supplementary explanations
RA78K0R\V*x.xx*\hlp\	78k0rp.chm	on-line help file
RA78K0R\V*x.xx*\inc78k0r\	compati.inc	Include file for assembler option (-compati)
RA78K0R\V*x.xx*\smp78k0r\ra78k0r\	k0rmain.asm	Assembler sample program (main routine)
	k0rsub.asm	Assembler sample program (subroutine)
	ra.bat	Batch file for assembler sample program
	readme.txt	Explanation of sample program and batch file (text file)

Remark *: Alphanumeric symbols

2.6 Uninstallation

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) Removing RA78K0R

Select "NEC EL RA78K0R Vx.xx" in [Add or Remove Programs] on the Control Panel.

(3) 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 Folder Configuration".

2.7 Environment Settings

2.7.1 Host machine

The assembler package is designed to run on the following OSs:

- Command prompt of Windows 2000/XP

2.7.2 Environmental variables

Set the following environmental variables.

Table 2-3 Environmental Variables

Environmental Variables	Explaination
PATH	Specifies the folder to which the executable format of the assembler is stored.
TMP	Specifies a folder where a temporary file is to be created
INC78K0R	Specifies a folder where the include file is searched.
LIB78K0R	Specifies the folder where a library is searched, if the library is used.
LANG78K	Specifies the kanji code (2-byte code) described in the comment.

[Example]

```
PATH=%PATH%; C:\Program Files\NEC Electronics Tools\RA78K0R\Vx.xx\bin
set TMP=C:\tmp
set INC78K0R=C:\Program Files\NEC Electronics Tools\RA78K0R\Vx.xx\inc78k0r
set LIB78K0R=C:\Program Files\NEC Electronics Tools\XXXXXX\Vx.xx\lib78k0r
set LANG78K=SJIS
```

2.7.3 Kanji code in source file

- Kanji (2-byte character) 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 (-zs/-ze/-zn).

CHAPTER 3 EXECUTION PROCEDURE OF RA78K0R

This chapter explains the procedures for using the assembler package RA78K0R, from assembling to object conversion.

Sample programs "k0rmain.asm" and "k0rsub.asm" are assembled, linked, and converted into objects in accordance with the execution procedures explained.

In this section, how to run the assembler package on command line is explained.

3.1 Before Executing RA78K0R

3.1.1 Sample programs

Among the files stored on the system disk are "k0rmain.asm" and "k0rsub.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.

k0rmain.asm: Main module

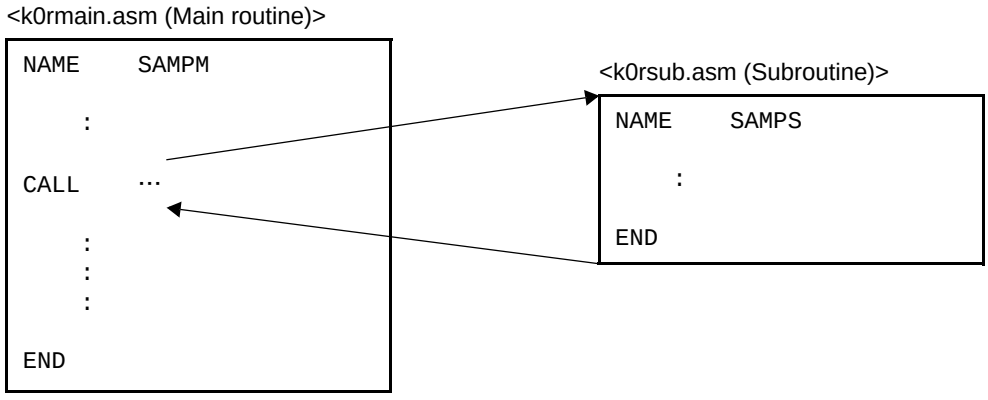
k0rsub.asm: Sub module

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 (k0rmain.asm).

The name of the subroutine module is SAMPS, and it is stored in (k0rsub.asm).

Figure 3-1 Structure of Sample Program



<k0rmain.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    SP , #_STBEG

        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
```

<k0rsub.asm (Subroutine)>

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

                PUBLIC  CONVAH

                CSEG
CONVAH :        XOR     A , A
                ROL4    [ HL ]          ; hex upper code load (a)
                CALL    !SASC
                MOV     B , A           ; store result

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

                RET

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

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

                END
```

Caution 1 This sample program is a reference program, prepared for the purpose of teaching you about the functions and operation of the RA78K0R. It cannot be used as an application program.

Caution 2 This sample program does not operate the default settings of the register bank selection flags (RBS0, RBS1). The settings for these items are therefore as follows.

Register bank 0 (FFEF8H to FFEFFH)

Caution 3 Since the ROL4 instruction described on line marked (a) in the above sample program is an instruction for the 78K0 but not supported by the 78K0R, specification of an assembler option (-compati) is required.

For details on the assembler option (-compati), refer to "4.4 Assembler Options".

3.1.2 Configuration of sample program

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

k0rmain.asm:	Main module
k0rsub.asm:	Sub module
mylib.lib:	Library file (this is not used here.)
sample.dr:	Directive file

3.2 Execution Procedure of RA78K0R

The batch files (ra.bat) in the system disk are used for the RA78K0R operation.

The assembler, linker, object converter, and list converter are executed in this order using "k0rmain.asm" and "k0rsub.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 by downloading it from the Version-up Service).

The following explanation uses the "uPD78F1166_A0" as the target device.

<ra.bat (batch program for verifying RA78K0R operation)>

```
        echo off
        cls
        set LEVEL = 0

        if "%1" == "" goto ERR_BAT

        ra78k0r -C%1 k0rmain.asm
        if errorlevel 1 set LEVEL = 1
        ra78k0r -C%1 k0rsub.asm
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo Assemble error !!
        if %LEVEL% == 1 goto END

        cls
        lk78k0r k0rmain.rel k0rsub.rel -s -orasample.lmf -prasample.map
        if errorlevel 1 echo Link error !!
        if errorlevel 1 goto END

        cls
        oc78k0r rasample
        if errorlevel 1 echo Object conversion error !!
        if errorlevel 1 goto END
        cls
        set LEVEL = 0
        lc78k0r -lrasample.lmf -rk0rmain.rel k0rmain.prn
        if errorlevel 1 set LEVEL = 1
        lc78k0r -lrasample.lmf -rk0rsub.rel k0rsub.prn
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo List conversion error !!
        if %LEVEL% == 1 goto END

        cls
        echo No error.
        goto END

: ERR_BAT

        echo Usage : ra.bat chiptype

: END
        echo on
```

(1) Execute the batch file.

Specify the target device specification name and execute the verification batch program for checking the RA78K0R operation.

```
C>ra.bat f1166a0
```

The following message is output to the display.

```
78K0R Assembler Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78f1166a0
 Device file : Vx.xx

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

78K0R Assembler Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78f1166a0
 Device file : Vx.xx

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

Clear the screen.

```
78K0R Linker Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

 Target chip : uPD78f1166a0
 Device file : Vx.xx

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

Clear the screen.

```
78K0R Object Converter Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

 Target chip : uPD78f1166a0
 Device file : Vx.xx

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

Clear the screen.

```
List Conversion Program for RA78K0R Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

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

List Conversion Program for RA78K0R Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

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

Clear the screen.

```
No error.
```

(2) Check the contents of drive C.

The following files are output.

k0rmain.rel:	Object module file
k0rmain.prn:	Assemble list file
k0rsub.rel:	Object module file
k0rsub.prm:	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
k0rmain.p:	Absolute assemble list file
k0rsu.p:	Absolute assemble list file

(3) Summary of RA78K0R execution procedure

The following is a brief summary of the execution procedure of RA78K0R.

Figure 3-2 RA78K0R Execution Procedure 1

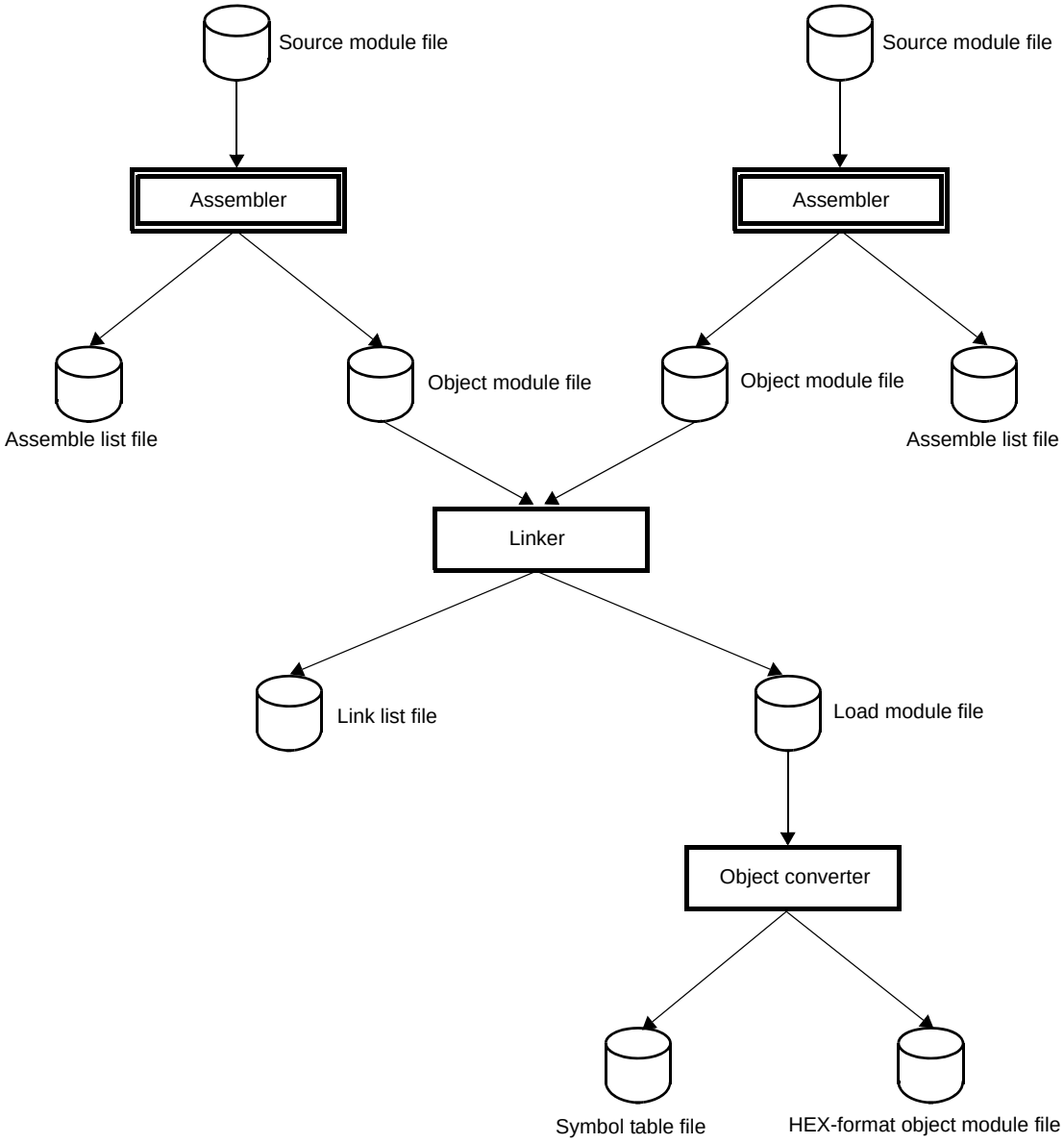
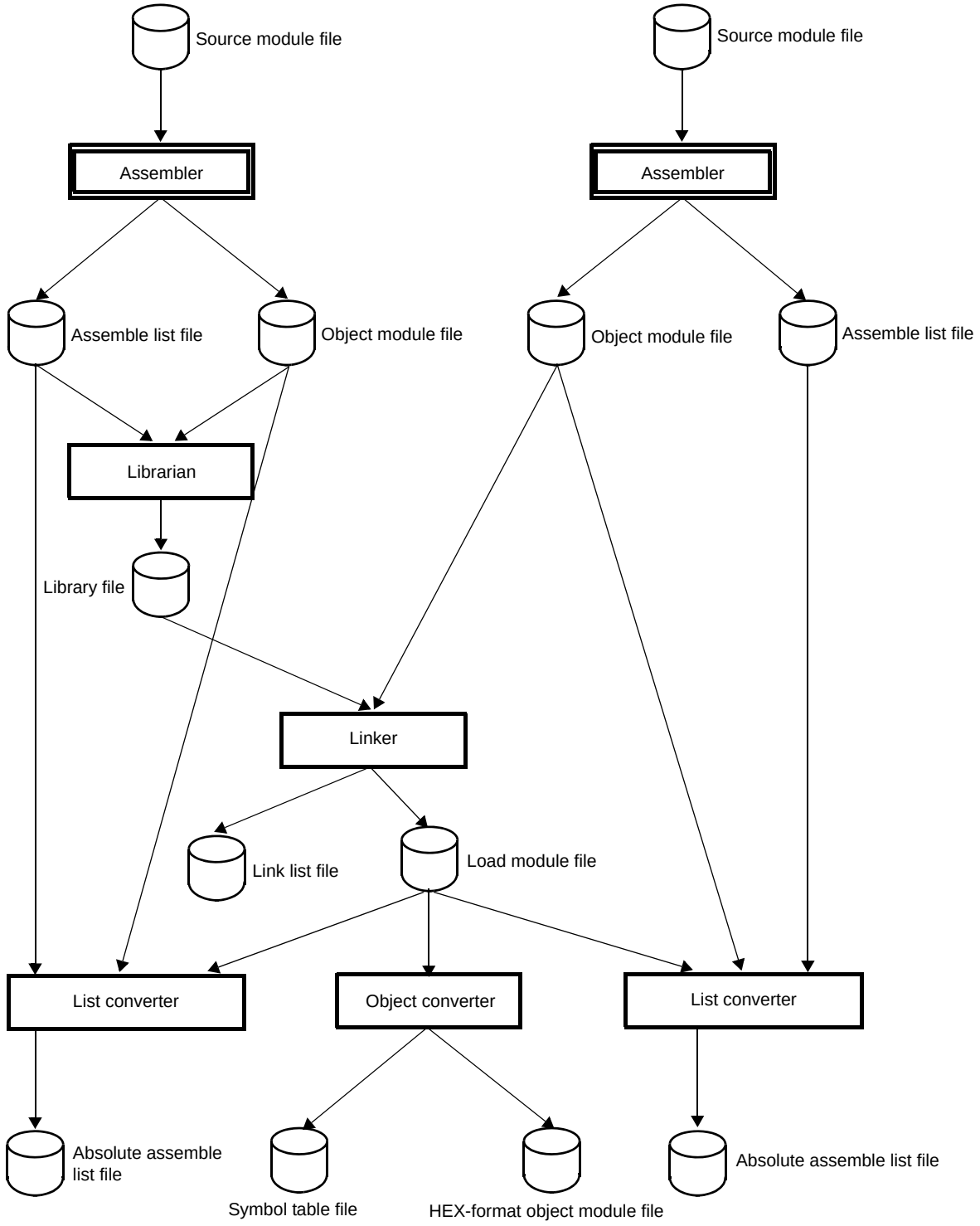


Figure 3-3 RA78K0R Execution Procedure 2



3.3 Execution Procedure from Command Line

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

(1) Assemble the sample program k0rmain.asm.

Input the following on the command line.

```
C>ra78k0r -cf1166a0 k0rmain.asm
```

The following message is output to the display.

```
78K0R Assembler Vx.xx [xx xxx xx]
    Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78F1166_A0
 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 (k0rmain.rel) and the assemble list file (k0rmain.prn).

If the -e option 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 k0rsub.asm.

Input the following on the command line.

```
C>ra78k0r -cf1166a0 k0rsub.asm
```

The following message is output to the display.

```
78K0R Assembler  Vx.xx [xx xxx xx]
    Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78F1166_A0
 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 (k0rsub.rel) and the assemble list file (k0rsub.prn).

During assembly, if the -e option is specified, the assembler outputs an error list file.

(5) Create a directive file.

A directive file is a file which indicates the location of segments for the linker.

Create a directive file when you need to expand the default ROM/RAM area or define a new memory area.

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

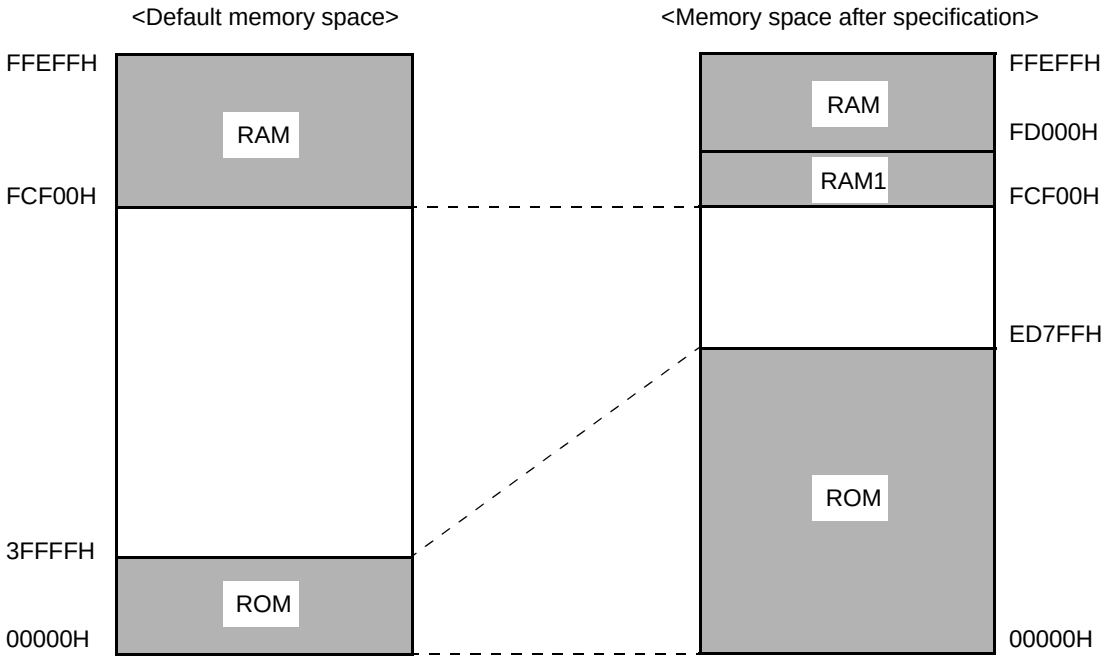
During linking, use the -d option to enter the directive file to the linker.

[Example]

- To extend the ROM area (0H to 3FFFFH) to (0H to ED7FFH), and extend the RAM area to (FCF00H to FFEFFH) and RAM1 (FCF00H to FCFFFH).

  Write the following to the directive file.

```
MEMORY ROM : ( 0H , 0ED800H )
MEMORY RAM1 : ( 0FCF00H , 100H )
MEMORY RAM : ( 0FD000H , 2F00H )
```



(6) As the result of the assembly, the output object module files "k0rmain.rel" and "k0rsub.rel" are linked.

Enter k0r.dr as the directive file.

Enter the following on the command line.

```
C>lk78k0r k0rmain.rel k0rsub.rel -dk0r.dr(Note 1) -ok0r.lmf -pk0r.map -s(Note 2)
```

Note 1 Not necessary if a directive file is not specified.

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

The following message is output to the display.

```
78K0R Linker Vx.xx [xx xxx xx]
    Copyright (C) NEC Electronics Corporation xxxx

 Target chip : uPD78F1166_A0
 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 (k0r.lmf) and the link list file (k0r.map).

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

(8) As the result of linking, the output load module file (k0r.lmf) is converted to a HEX-format file.

Enter the following on the command line.

```
C>oc78k0r k0r.lmf -r -u0FFH
```

The following message is output on the display.

```
78K0R Object Converter Vx.xx [xx xxx xx]
    Copyright (C) NEC Electronics Corporation xxxx

 Target chip : uPD78F1166_A0
 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 (k0r.hex) and the symbol table file (k0r.sym).

(10) Create a library file as follows.

Register the object module file (k0rsub.rel) output by the assembler as a library file.

Enter the following on the command line.

```
C>lb78k0r < k0r.job
```

The following message is output on the display.

```
78K0R Librarian Vx.xx [xx xxx xx]
    Copyright (C) NEC Electronics Corporation xxxx
*create k0r.lib                         ; Contents of "k0r.job"
*add k0r.lib k0rsub.rel                 ; Contents of "k0r.job"
*exit
```

(11) Check the contents of drive C.

The librarian outputs the library file (k0r.lib).

(12) Create an absolute assemble list as follows.

To create the absolute assemble list k0rmain.asm, input "k0rmain.rel", "k0rmain.asm" and "k0r.lmf" to the list converter.

Enter the following on the command line.

```
C>lc78k0r main -l.lmf
```

The following message is output on the display.

```
List Conversion Program for RA78K0R Vx.xx [xx xxx xx]
    Copyright (C) NEC Electronics Corporation xxxx

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

(13) Check the contents of drive C.

The list converter outputs the absolute assemble list file (k0rmain.p).

3.4 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 on PM+.

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

```
>[path-name]ra78k0r△-fparameter-file-name
>[path-name]lk78k0r△-fparameter-file-name
>[path-name]oc78k0r△-fparameter-file-name
```

Here is an example of its use.

```
C>ra78k0r -fpara.pra
C>lk78k0r -fpara.plk
C>oc78k0r -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>

```
-cf1166a0 k0rmain.asm -e
```

<Contents of para1.plk>

```
k0rmain.rel k0rsub.rel -bmylib.lib -osample.lmf -s
```

<Contents of para1.poc>

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

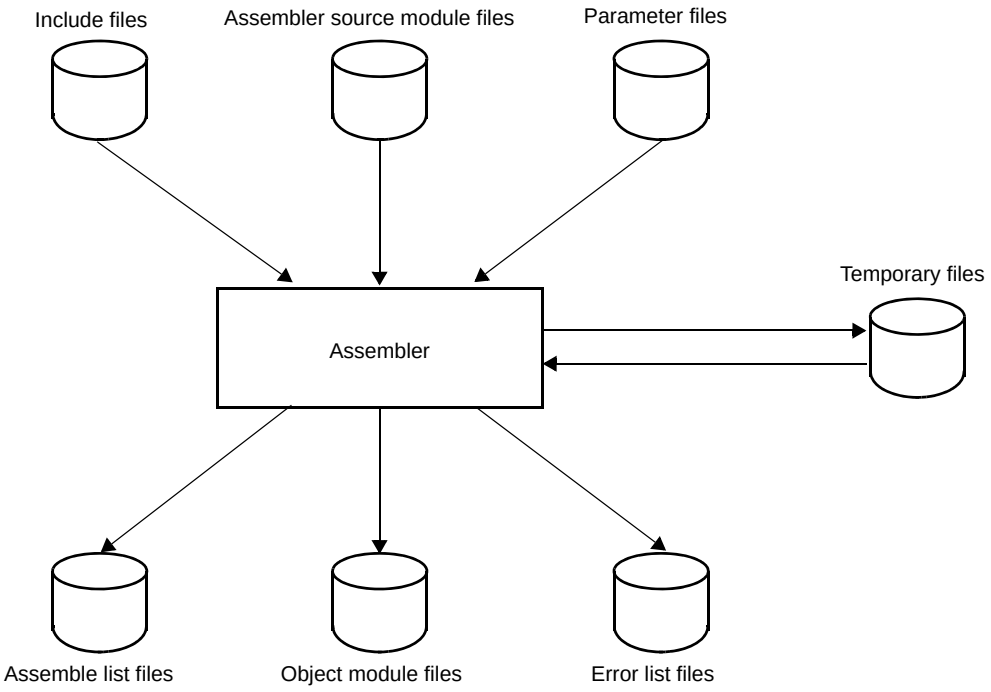
CHAPTER 4 ASSEMBLER

The assembler inputs source module files written in the assembly language for 78K0R 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.

Figure 4-1 I/O Files of Assembler



4.1 I/O Files of Assembler

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

Table 4-1 I/O Files of Assembler

Type	File Name	Explanation	Default File Type
Input files	Assembler source module files	- Source module files written in assembly language for 78K0R microcontrollers (user-created files)	.asm
	Include files	- Files referenced from assembler source module files - Source module files written in assembly language for 78K0R microcontrollers (user-created files)	-
	Parameter files	- Files containing the parameters for the executed programs (user-created files)	.pra
Output files	Object module files	- Binary files containing relocation data and symbol data regarding machine language data and machine language location addresses	.rel
	Assemble list files	- Files containing assembly data such as assemble lists and cross-reference lists	.prn
	Error list files	- Files containing error data generated during assembly	.era
I/O files	Temporary files	- Files created automatically by the assembler for assembly purposes Temporary files are deleted when assembly ends.	RAxxxxx.$$n (n = 1 to 4)

4.2 Functions of Assembler

(1) Conversion of assembly language into machine language

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

4.3 Assembler Startup

4.3.1 Methods to start assembler

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

(1) Startup from the command line

```
 X>[path-name]ra78k0r[Δoption]...source-module-file-name[Δoption]...[Δ]
  |       |       |       |                    |               |
 (a)     (b)     (c)     (d)                  (e)             (d)
```

(a) Current drive name

(b) Current folder name

(c) Command file name of the assembler

(d) Enter detailed instructions for the operation of the assembler.

When specifying two or more assembler options, separate the options with a blank space. Uppercase characters and lowercase characters are not distinguished for the assembler options. For a detailed explanation of assembler options, refer to "4.4 Assembler Options".

Enclose a path that includes a space in a pair of double quotation marks (" ").

(e) File name of source module to be assembled

Specify the file name of a path that includes a space by enclosing it in a pair of double quotation marks (" ").

[Example]

- To output an error list file k0rmain.era, describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -e -np
```

(2) Startup from a parameter file

Use the parameter file when the data required to start up the assembler will not fit on the command line, or when repeating the same assembler option for two or more assembly operations.

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

Start up the assembler from a parameter file as follows.

```
X>ra78k0r[Δsource-module-file]Δ-fparameter-file-name
                                |           |
                               (a)         (b)
```

(a) Parameter file specification option

(b) A file which includes the data required to start up the assembler

Remark 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 "3.4 Using Parameter File".

[Example]

(1) Create a parameter file k0rmain.pra using an editor.

```
; parameter file
k0rmain.asm -osample.rel
-psample.prn
```

(2) Use the parameter file k0rmain.pra to start up the assembler.

```
C>ra78k0r -fk0rmain.pra
```

4.3.2 Execution start and end messages

(1) Execution start message

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

```
78K0R Assembler Vx.xx [xx xxx xx]
     Copyright (C) NEC Electronics Corporation 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
k0rmain.asm ( 12 ) : RA78K0R error E2201 : Syntax error
PASS2 Start
k0rmain.asm ( 12 ) : RA78K0R error E2201 : Syntax error
k0rmain.asm ( 29 ) : RA78K0R error E2407 : Undefined symbol reference 'CON-
VAH'
k0rmain.asm ( 29 ) : RA78K0R error E2303 : 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]

<A non-existent source module file is specified.>

```
C>ra78k0r sample.asm
```

```
78K0R Assembler Vx.xx [xx xxx xx]
    Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F2006 : File not found 'sample.asm'
Program aborted.
```

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

<A non-existent assembler option is specified.>

```
C>ra78k0r k0rmain.asm -z
```

```
78K0R Assembler Vx.xx [xx xxx xx]
    Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F2012 : Missing parameter '-z'
Please enter 'RA78K0R--', if you want help messages.
Program aborted.
```

In the above example, a non-existent assembler option is specified. An error occurs and the assembler aborts assembly.

When an error message is displayed and assembly is aborted, look for the cause in "CHAPTER 11 ERROR MESSAGES" and take action accordingly.

4.4 Assembler Options

4.4.1 Types of assembler options

The assembler options are detailed instructions for the operation of the assembler.

The types and explanations for assembler options are shown below.

Table 4-2 Assembler Options

Classification	Option	Explanation
Device type specification	-c	Specifies the device type of the target device.
Object module file output specification	-o	Specifies the output of an object module file.
	-no	
Forced object module file output specification	-j	Forces output of an object module file.
	-nj	
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	
Include file read path specification	-i	Reads from the path specified in an include file.
Assemble list file output specification	-p	Specifies output of an assemble list file.
	-np	
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	
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	
Error list file output specification	-e	Outputs an error list file.
	-ne	

Classification	Option	Explanation
Parameter file specification	-f	Inputs the input file name and assembler options from a specified file.
Specification of path for temporary file creation	-t	Creates a temporary file in a specified path.
Kanji code (2-byte 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.
Device file search path specification	-y	Reads a device file from a specified path.
Symbol definition specification	-d	Defines a symbol.
78K0R common object specification	-common	Specifies output of an object module file common to the 78K0R.
Self programming specification	-self	Specifies when using self programming.
78K0 compatible macro	-compati	Enables assembly of assembler source files generated by the 78K0 assembler.
	-ncompati	
Help specification	- -	Displays a help message on the display.

4.4.2 Order of precedence of assembler options

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

Table 4-3 Order of Precedence of Assembler Options

	-no	-np	-nka	-nks	-kx	-nkx	--
-j	NG						NG
-g	NG						NG
-p			Δ	Δ		Δ	NG
-ka		NG					NG
-ks		NG			NG		NG
-kx		NG					NG
-lw		NG					NG
-ll		NG					NG
-lh		NG					NG
-lt		NG					NG
-lf		NG					NG

[Items marked with an NG]

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

<Example>

```
C>ra78k0r -cf1166a0 k0rmain.asm -no -lw80 -lf
```

The -lw and -lf options 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>ra78k0r -cf1166a0 k0rmain.asm -p -nka -nks -nkx
```

The -nka, -nks and -nkx options are all specified at the same time, so the -p option 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>ra78k0r -cf1166a0 k0rmain.asm -o -no
```

The -no option is specified after -o, so the -o option is unavailable and -no is available.

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

Device type specification

(1) -c

[Syntax]

```
-cdevice-type
```

- Default assumption

  Cannot be omitted

[Function]

- The -c option specifies the device type of the target device.

[Application]

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

[Explanation]

- For the target devices that can be specified by the -c option, refer to the user's manual of the device used or "Device Files Operating Precautions".

[Notice]

- The -c option cannot be omitted. However, if a control instruction ($PROCESSOR) with the same function is described at the beginning of the source module, command-line specification can be omitted.

```
Δ$ΔPROCESSORΔ(Δdevice-typeΔ)
Δ$ΔPCΔ(Δdevice-typeΔ)          ; Abbreviated form
```

  For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To specify the uPD78F1166_A0 as the target device, describe as:

```
C>ra78k0r -cf1166a0 main.asm
```

Object module file output specification

(1) -o/-no

[Syntax]

```
-o[output-file-name]
-no
```

- Default assumption

  -o*input-file-name*.rel

[Function]

- The -o option specifies the output of an object module file. It also specifies the location to which it is output and the file name.
- The -no option makes the -o, -j, -g, and -ga option unavailable.

[Application]

- Use the -o option to specify the location to which an object module file is output or to change its file name. Specify the -no option when performing assembly only to output an assemble list file. This will shorten assembly time.

[Explanation]

- The disk type file name, device type file names NUL and AUX, and the path name can be specified in output file name. An abort error occurs when the device type file names CON, and PRN are specified.
- Even if the -o option is specified, if a fatal error occurs the object module file cannot be output.
- If the drive name is omitted when the -o option is specified, the object module file will be output to the current drive.
- If the output file name is omitted when the -o option is specified, the output file name will be "input-file-name.rel".
- If both the -o and -no options are specified at the same time, the option specified last takes precedence.

[Example of use]

- To output an object module file (sample.rel), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -osample.rel
```

Forced object module file output specification

(1) -j/-nj

[Syntax]

```
-j
-nj
```

- Default assumption

  -nj

[Function]

- The -j option specifies that the object module file can be output even if a fatal error occurs.
- The -nj option makes the -j option 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 the -j option to output the object module file.

[Explanation]

- When the -j option is specified, the object module file will be output even if a fatal error occurs.
- If both the -j and -nj options are specified at the same time, the option specified last takes precedence.
- If the -no option is specified, the -j option is unavailable.

[Example of use]

- To output an object module file (k0rmain.rel) even if a fatal error occurs, describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -j
```

Debug data output specification

(1) -g/-ng

[Syntax]

```
-g
-ng
```

- Default assumption

  -g

[Function]

- The -g option specifies that debugging data (local symbol data) is to be added to an object module file.
- The -ng option makes the -g option unavailable.

[Application]

- Use the -g option when performing symbolic debugging of data that includes local symbol data.
- Use the -ng option in the following 3 cases.

  (i) Symbolic debugging of global symbols only

  (ii) Debugging without symbols

  (iii) When only the object is required (evaluation using PROM, etc.)

[Explanation]

- If both the -g and -ng options are specified at the same time, the option specified last takes precedence.
- The -ga option takes precedence over other options regardless of the position in which it is specified.
- If the -no option is specified, the -g option is unavailable.

[Notice]

- A control instruction (DEBUG/NODEBUG or DG/NODG) with the same function as the -g/-ng options can be written at the beginning of a source module.

```
Δ$ΔDEBUG
Δ$ΔDG           ; abbreviated form
Δ$ΔNODEBUG
Δ$ΔNODG         ; abbreviated form
```

  For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To add debug data (local symbol data) to an object module file (k0rmain.rel), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -g
```

(2) -ga/-nga

[Syntax]

```
-ga
-nga
```

- Default assumption

  -ga

[Function]

- The -ga option specifies that source debugging data is to be added to an object module file by the structured assembler.
- The -nga option makes the -ga option unavailable.

[Application]

- Use the -ga option 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.
- Use the -nga option in the following 2 cases.

  (i) Debugging without an assembler source

  (ii) When only the object is required (evaluation using PROM, etc.)

  (iii) Debugging at the source level of the C compiler

[Explanation]

- If both the -ga and -nga options are specified at the same time, the option specified last takes precedence.
- The -ga option takes precedence over other options regardless of the position in which it is specified.
- If the -no option is specified, the -ga option is unavailable.

[Notice]

- A control instruction (DEBUGA/NODEBUGA) with the same function as the -ga and -nga options can be written at the beginning of a source module.

```
∆$∆DEBUGA
∆$∆NODEBUGA
```

For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To add assembler source debug data to an object module file (k0rmain.rel), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -ga
```

Include file read path specification

(1) -i

[Syntax]

```
-ipath-name[,path-name]... (two or more path names can be specified)
```

- Default assumption

  The include file is searched in the following sequence.

  (i) Path where the source file exists

  (ii) Path specified by environmental variable (INC78K0R)

[Function]

- The -i option specifies input of an include file specified by "$include" in a source module from a specified path.

[Application]

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

[Explanation]

- Two or more path names can be specified at once by separating them with ",".
- A space cannot be entered before or after the ",".
- The include file specified by "$include" is searched in the following sequence.

  (i) If two or more path names are specified following the -i option, the include file is searched in the specified order.

  (ii) If two or more -i options are specified, the include file is searched with the option specified later taking precedence.

  (iii) After the path specified by the -i option is searched, the include file is searched in the same order as the default assumption.
- An abort error occurs if anything other than a path name is specified after -i, or if the path name is omitted.
- An abort error occurs if -i is used to specify 65 or more path names.

[Example of use]

- To search and read an include file from folders C:\sample1 and C:\sample2 in that order, describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -iC:\sample,C:\sample2
```

- To read an include file from folder C:\Program Files\NEC Electronics Tools\include files, describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -i"C:\Program Files\NEC Electronics
Tools\include files"
```

Assemble list file output specification

(1) -p/-np

[Syntax]

```
-p[output-file-name]
-np
```

- Default assumption

  -p*input-file-name*.prn

[Function]

- The -p option specifies output of an assemble list file. It also specifies the destination and file name of the output file.
- The -np option makes the -p, -ka, -ks, -kx, -lw, -ll, -lh, -lt, and -lf option unavailable.

[Application]

- Specify the -p option to change the output destination or output file name of an assemble list file.
- Specify the -np option when performing assembly only to output an object module file. This will shorten assembly time.

[Explanation]

- 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 the output file name is omitted when the -p option is specified, the assemble list file name becomes *input-file-name*.prn.
- If the drive name is omitted when the -p option is specified, the assemble list file will be output to the current drive.
- If both the -p and -np options are specified at the same time, the option specified last takes precedence.

[Example of use]

- To output an assemble list file (sample.prn), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -psample.prn
```

Assemble list file data specification

(1) -ka/-nka

[Syntax]

```
-ka
-nka
```

- Default assumption

  -ka

[Function]

- The -ka option outputs an assemble list into an assemble list file.
- The -nka option makes the -ka option unavailable.

[Application]

- Specify the -ka option to output an assemble list.

[Explanation]

- If both the -ka and -nka options are specified at the same time, the option specified last takes precedence.
- If the -nka, -nks, and -nkx options are all specified, the assemble list file cannot be output.
- If the -np option is specified, the -ka option is unavailable.

[Example of use]

- To output an assembly list file into an assemble list file (k0rmain.prn), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -ka
```

<Contens of k0rmain.prn>

```
        Assemble list

ALNO  STNO  ADRS  OBJECT  M I   SOURCE STATEMENT
 1     1
 2     2                              NAME  SAMPM
 3     3                           ; ****************************************
 4     4                           ; *                                      *
 5     5                           ; *     HEX -> ASCII Conversion Program  *
 6     6                           ; *                                      *
 7     7                           ; *             main-routine             *
 8     8                           ; *                                      *
 9     9                           ; ****************************************
10    10
11    11                           PUBLIC  MAIN , START
12    12                           EXTRN   CONVAH
13    13                           EXTRN   _@STBEG
14    14
15    15    -----                  DATA    DSEG    AT      0FFE20H
16    16    FFE20                  HDTSA : DS      1
17    17    FFE21                  STASC : DS      2
18    18
19    19    -----                  CODE    CSEG    AT      0H
20    20    00000 R0000            MAIN :  DW      START
21    21
22    22    -----                          CSEG
23    23    00000                  START :
24    24
25    25                                   ; chip initialize
26    26    00000 RCBF80000                MOVW    SP , #_@STBEG
27    27
28    28    00004 CD201A                   MOV     HDTSA , #1AH
29    29    00007 3620FE                   MOVW    HL , #LOWW ( HDTSA ) ; set hex
2-code data in HL registor
        :
```

(2) -ks/-nks

[Syntax]

```
-ks
-nks
```

- Default assumption

  -nks

[Function]

- The -ks option outputs an assemble list followed by a symbol list into an assemble list file.
- The -nks option makes the -ks option unavailable.

[Application]

- Specify the -ks option to output a symbol list.

[Explanation]

- If both the -ks and -nks options are specified at the same time, the option specified last takes precedence.
- If the -ks and -kx options are specified at the same time, -ks is ignored.
- If the -nka, -nks, and -nkx options are all specified, the assemble list file cannot be output.
- If the -np option is specified, the -ks option is unavailable.

[Example of use]

- To output an assemble list followed by a symbol list into an assemble list file (k0rmain.prn), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -ks
```

<Contents of k0rmain.prn>

```
        Symbol Table List

VALUE   ATTR    RTYP    NAME            VALUE   ATTR    RTYP    NAME

        CSEG            ?CSEG                   CSEG            CODE
-----H          EXT     CONVAH                  DSEG            DATA
FFE20H  ADDR            HDTSA               0H  ADDR    PUB     MAIN
        MOD             SAMPM               0H  ADDR    PUB     START
FFE21H  ADDR            STASC           -----H          EXT     _@STBEG
```

(3) -kx/-nkx

[Syntax]

```
-kx
-nkx
```

- Default assumption

  -nkx

[Function]

- The -kx option outputs an assemble list followed by a cross-reference list into an assemble list file.
- The -nkx option makes the -kx option unavailable.

[Application]

- Specify the -kx option 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]

- If both the -kx and -nkx options are specified at the same time, the option specified last takes precedence.
- If the -ks and -kx options are specified at the same time, -ks is ignored.
- If the -nka, -nks, and -nkx options are all specified, the assemble list file cannot be output.
- If the -np option is specified, the -kx option is unavailable.

[Notice]

- A control instruction (XREF/NOXREF, or XR/NOXR) with the same function as the -kx/-nkx option can also be written at the beginning of a source module.

```
Δ$ΔXREF
Δ$ΔXR           ; abbreviated form
Δ$ΔNOXREF
Δ$ΔNOXR         ; abbreviated form
```

  For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To output an assemble list followed by a cross-reference list into an assemble list file (k0rmain.prn), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -kx
```

<Contents of k0rmain.prn>

```
        Cross-Reference List

NAME    VALUE   R       ATTR    RTYP    SEGNAME XREFS

?CSEG                   CSEG            ?CSEG   22#
CODE                    CSEG            CODE    19#
CONVAH  -----H  E               EXT             12@     31
DATA                    DSEG            DATA    15#
HDTSA   FFE20H          ADDR            DATA    16#     28      29
MAIN        0H          ADDR    PUB     CODE    11@     20#
SAMPM                   MOD              2#
START       0H  R       ADDR    PUB     ?CSEG   11@     20      23#
STASC   FFE21H          ADDR            DATA    17#     33
_@STBEG -----H  E               EXT             13@     26
```

Assemble list file format specification

(1) -lw

[Syntax]

```
-lw[number-of-characters]
```

- Default assumption

  -lw132 (80 characters in the case of display output)

[Function]

- The -lw option changes the number of characters that can be printed in 1 line in a list file.

[Application]

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

[Explanation]

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

  72 ≤ number of characters printed on 1 line ≤ 2046

  An abort error occurs if a numerical value outside this range, or something other than a numerical value, is specified.
- 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.
- The specified number of characters does not include the terminator (CR, LF).
- If the -np option is specified, the -lw option is unavailable.

[Notice]

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

  ```
  ΔΔ$ΔWIDTH
  ```

  For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To specify 80 as the number of characters per line in an assemble list file (k0rmain.prn), describe as:

  ```
  C>ra78k0r -cf1166a0 k0rmain.asm -lw80
  ```

<Contents of k0rmain.prn>

```
        Assemble list

ALNO  STNO  ADRS  OBJECT  M I   SOURCE STATEMENT

 1     1
 2     2                              NAME  SAMPM
 3     3                          ; ************************************
 4     4                          ; *                                  *
 5     5                          ; * HEX -> ASCII Conversion Program  *
 6     6                          ; *                                  *
 7     7                          ; *           main-routine           *
 8     8                          ; *                                  *
 9     9                          ; ************************************
10    10
11    11                          PUBLIC  MAIN , START
12    12                          EXTRN   CONVAH
13    13                                  EXTRN   _@STBEG
14    14
15    15    -----                 DATA    DSEG    AT      0FFE20H
16    16    FFE20                 HDTSA : DS      1
17    17    FFE21                 STASC : DS      2
18    18
19    19    -----                 CODE    CSEG    AT      0H
20    20    00000   R0000         MAIN :  DW      START
21    21
22    22    -----                         CSEG
23    23    00000                 START :
24    24
25    25                                  ; chip initialize
26    26    00000   RCBF80000            MOVW    SP , #_@STBEG
27    27
28    28    00004   CD201A               MOV     HDTSA , #1AH
29    29    00007   3620FE               MOVW    HL , #LOWW ( HDTSA )  ; set
hex 2-code data in HL registor
        :
```

(2) -ll

[Syntax]

```
-ll[number-of-lines]
```

- Default assumption

  -ll0 (No page breaks)

[Function]

- The -ll option changes the number of lines that can be printed in 1 page in an assemble list file.

[Application]

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

[Explanation]

- The range of number of lines that can be specified with the -ll option is shown below.

  20 ≤ number of lines printed on 1 page ≤ 32,767

  An abort error occurs if a numerical value outside this range, or something other than a numerical value, is specified.
- If the number of lines is omitted, 0 will be specified.
- If the number of lines specified is 0, no page breaks will be made.
- If the -np option is specified, the -ll option is unavailable.

[Notice]

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

  ```
  ∆$∆LENGTH
  ```

  For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To specify 20 as the number of lines per page in an assemble list file (k0rmain.prn), describe as:

  ```
  C>ra78k0r -cf1166a0 k0rmain.asm -ll20
  ```

<Contents of k0rmain.prn>

```
78K0R Assembler Vx.xx           Date:xx xxx xx  Page:1

Command : -cf1166a0 k0rmain.asm -ll20
Para-file :
In-file : k0rmain.asm
Obj-file : k0rmain.rel
Prn-file : k0rmain.prn

        Assemble list

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


78K0R Assembler Vx.xx           Date:xx xxx xx  Page:2

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                          ;

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


78K0R Assembler Vx.xx           Date:xx xxx xx  Page:3


ALNO  STNO  ADRS  OBJECT   M I  SOURCE  STATEMENT

 9     9                          ; *********************************
10    10
11    11                          PUBLIC  MAIN , START
12    12                          EXTRN   CONVAH
13    13                          EXTRN   _@STBEG
14    14
15    15    -----                 DATA    DSEG    AT      0FFE20H
16    16    FFE20                 HDTSA : DS      1
       :
```

(3) -lh

[Syntax]

```
-lhcharacter-string
```

- Default assumption

  None

[Function]

- The -lh option specifies the character string printed in the title column of the header of an assemble list file.

[Application]

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

[Explanation]

- Up to 60 characters can be specified in the title. The character string cannot include blank spaces.
- If more than 61 characters are written, the first 60 characters will be recognized and no error message will be output.

  A kanji (2-byte character) is calculated as two characters.

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

  Effective length = (Max. number of characters per line) - 60
- An abort error occurs if the length of the character string is not specified.
- If the -np option is specified, the -lh option is unavailable.
- If the -lh option is omitted, the title column of the assemble list file will be blank.

- The character set that can be written in the title column is as follows.

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

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

[Notice]

- A control instruction (TITLE, or TT) with the same function as the -lh option can also be written at the beginning of the startup line.

```
Δ$ΔTITLEΔ(Δ'character-string'Δ)
Δ$ΔTTΔ(Δ'character-string'Δ)            ; abbreviated form
```

For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To print the title “RA78K0R_MAINROUTINE“ in the header of an assemble list file (k0rmain.prn), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
```

<Contents of k0rmain.prn>

```
78K0R Assembler Vx.xx   RA78K0R_MAINROUTINE     Date:xx xxx xx  Page:1
                                  |
                                Title

Command : -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
Para-file :
In-file : k0rmain.asm
Obj-file : k0rmain.rel
Prn-file : k0rmain.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               *
        :
```

(4) -lt

[Syntax]

```
-lt[number-of-characters]
```

- Default assumption

  -lt8

[Function]

- The -lt option 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 the -lw option, specify the -lt option to insert a tab instead of a series of blank spaces, thus saving on the number of characters used.

[Explanation]

- The range of number of characters that can be specified with the -lt option is shown below.

  0 ≤ number of characters that can be specified ≤ 8

  An abort error occurs if a numerical value outside this range, or something other than a numerical value, is specified.
- If the number of characters is omitted, 8 will be specified.
- If -lt0 is specified, tabulation processing will not be performed, and a tabulation code will be output.
- If the -np option is specified, the -lt option is unavailable.

[Notice]

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

  ```
  ΔΔ$ΔTABΔnumber-of-tabs
  ```

  For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To reference an assemble list file (sample.prn) when the -lt option is omitted, describe as:

  ```
  C>ra78k0r -cf1166a0 sample.asm
  ```

<Contents of sample.prn>

```
       Assemble list

ALNO  STNO  ADRS  OBJECT  M I   SOURCE STATEMENT

 1     1
 2     2                              NAME  SAMPM
 3     3                          ; **********************************
 4     4                          ;
 5     5                          ;       HEX -> ASCII Conversion Program
 6     6                          ;
 7     7                          ;             main-routine
 8     8                          ;
 9                                ; **********************************
10    10
11    11                          PUBLIC  MAIN , START
12    12                          EXTRN   CONVAH
13    13                          EXTRN   _@STBEG
14    14
15    15    -----                 DATA    DSEG    AT      0FFE20H
16    16    FFE20                 HDTSA : DS      1
17    17    FFE21                 STASC : DS      2
18    18
19    19    -----                 CODE    CSEG    AT      0H
20    20    00000 R0000           MAIN :  DW      START
        :
```

- To specify 1 blank entered by the HT code, describe as:

```
C>ra78k0r -cf1166a0 sample.asm -lt1
```

<Contents of sample.prn>

```
       Assemble list

ALNO  STNO  ADRS  OBJECT  M I   SOURCE STATEMENT

 1     1
 2     2                              NAME  SAMPM
 3     3                          ; **********************************
 4     4                          ;
 5     5                          ;       HEX -> ASCII Conversion Program
 6     6                          ;
 7     7                          ;             main-routine
 8     8                          ;
 9     9                          ; **********************************
10    10
11    11                          PUBLIC MAIN , START
12    12                          EXTRN CONVAH
13    13                          EXTRN _@STBEG
14    14
15    15    -----                 DATA DSEG AT 0FFE20H
16    16    FFE20                 HDTSA : DS 1
17    17    FFE21                 STASC : DS 2
18    18
19    19    -----                 CODE CSEG AT 0H
20    20    00000 R0000           MAIN : DW START
        :
```

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

(5) -lf/-nlf

[Syntax]

```
-lf
-nlf
```

- Default assumption

  -nlf

[Function]

- The -lf option inserts a form feed (FF) code at the end of an assemble list file.
- The -nlf option makes the the -lf option unavailable.

[Application]

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

[Explanation]

- If the -np option is specified, the -lf option is unavailable.
- If both the -lf and -nlf options are specified at the same time, the option specified last takes precedence.

[Notice]

- A control instruction (FORMFEED/NOFORMFEED) with the same function as the -lf/-nlf option can also be written at the beginning of a source module.

```
∆$∆FORMFEED
∆$∆NOFORMFEED
```

  For information on control instructions, refer to RA78K0R Assembler Package Language User's Manual.

[Example of use]

- To add a form feed code at the end of an assemble list file (k0rmain.prn), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -p -lf
```

Error list file output specification

(1) -e/-ne

[Syntax]

```
-e[output-file-name]
-ne
```

- Default assumption

  -ne

[Function]

- The -e option outputs an error list file, and specifies the output destination and output file name of the error list file.
- The option -ne makes the the -e option unavailable.

[Application]

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

[Explanation]

- The error list file can be saved as a disk-type file or as a device-type file.
- When the -e option is specified and the output file name is omitted, the error list file name will be *input-file-name*.era.
- When the -e option is specified and the drive name is omitted, the error list file will be output to the current folder.
- If both the -e and -ne options are specified at the same time, the option specified last takes precedence.

[Example of use]

- To create an error list file (k0rmain.era), describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -ek0rmain.era
```

```
78K0R Assembler Vx.xx [xx xxx xx]
     Copyright (C) NEC Electronics Corporation xxxx

PASS1 Start
k0rmain.asm ( 31 ) : RA78K0R error E2202 : lllegal operand
PASS2 Start
k0rmain.asm ( 26 ) : RA78K0R error E2312 : Operand out of range (byte)
k0rmain.asm ( 31 ) : RA78K0R error E2202 : lllegal operand

 Target chip : uPD78F1166_A0
 Device file : Vx.xx

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

<Contents of k0rmain.era>

```
PASS1 Start
k0rmain.asm ( 31 ) : RA78K0R error E2202 : lllegal operand
PASS2 Start
k0rmain.asm ( 26 ) : RA78K0R error E2312 : Operand out of range (byte)
k0rmain.asm ( 31 ) : RA78K0R error E2202 : lllegal operand
```

Parameter file specification

(1) -f

[Syntax]

```
-ffile-name
```

- Default assumption

  Options and input file names can only be input from the command line.

[Function]

- The -f option inputs assembler options and the input file name from a specified file.

[Application]

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

[Explanation]

- Only a disk-type file name can be specified as *file-name*. If a device-type file name is specified, an abort error occurs
- If the file name is omitted, an abort error occurs.
- Nesting of parameter files is not permitted. If the -f option is specified within a parameter file, an abort error occurs.
- The number of characters that can be written within a parameter file is unlimited.
- Separate options or file names with a blank space, a tab or the line feed code (LF).
- Parameters and input file names within a parameter file will be expanded at the position specified for the parameter file on the command line.
- The expanded options specified last will take precedence.
- The characters following ";" or "#" in a parameter file are all assumed to be comments, up to the line feed code (LF) or EOF.
- If the -f option is specified two or more times, an abort error occurs.

[Example of use]

- Perform assembly using a parameter file.
  Set the contents of the parameter file (k0rmain.pra) as follows.

```
; parameter file
k0rmain.asm -osample.rel -g -cf1166a0
-psample.prn
```

Enter the following on the command line.

```
C>ra78k0r -fk0rmain.pra
```

Specification of path for temporary file creation

(1) -t

[Syntax]

```
-tpath-name
```

- Default assumption

  Path specified by environmental variable TMP

  Current path, if no path is specified.

[Function]

- The -t option specifies a path in which a temporary file is created.

[Application]

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

[Explanation]

- Only a path can be specified as a path name.
- The path name cannot be omitted.
- Even if a previously created temporary file exists, if the file is not protected it will be overwritten.
- 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.
- Temporary files are deleted when assembly is finished. They are also deleted when assembly is aborted by pressing (CTRL+C).
- The path in which the temporary file is to be created is determined according to the following sequence.

  (i) The path specified by the -t option

  (ii) The path specified by environmental variable TMP (when the -t option is omitted)

  (iii) The current path (when TMP is not set)

  When (i) or (ii) is specified, if the temporary file cannot be created in the specified path, an abort error occurs.

[Example of use]

- To output a temporary file to folder C:\tmp, describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -tC:\tmp
```

- To output a temporary file to folder C:\Program Files\NEC Electoronics Tools\temporary files, describe as:

```
C>ra78k0r -cf1166a0 k0rmain.asm -t”C:\Program Files\NEC Electronics
Tools\temporary files”
```

Kanji code (2-byte code) specification

(1) -zs/-ze/-zn

[Syntax]

```
-zs
-ze
-zn
```

- Default assumption

  -zs

[Function]

- Kanji (2-byte character) described in the comment is interpreted as the specified kanji code (2-byte code).
- 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]

- If the -zs, -ze, and -zn options are specified at the same time, the one specified later takes priority.
- The control instruction (KANJICODE) that functions as the -zs, -ze, and -zn options can be described at the start of the source module.

  The syntax is shown below.

```
Δ$ΔKANJICODEΔSJIS
Δ$ΔKANJICODEΔEUC
Δ$ΔKANJICODEΔNONE
```

  For details of the control instruction, refer to RA78K0R Assembler Package Language User's Manual.
- Kanji code can also be specified by using the environmental variable LANF78K. For details of the environmental variables, refer to "10.2 Preparing Development Environment (Environmental Variables)".

[Example of use]

- To interprete the kanji code as EUC code, describe as:

```
C>ra78k0r k0rmain.asm -cf1166a0 -ze
```

Device file search path specification

(1) -y

[Syntax]

```
-ypath-name
```

- Default assumption

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

  (i) Path registered in the device file installer

  (ii) Path by which RA78K0R was started up

  (iii) Current folder

  (iv) The environmental variable PATH

[Function]

- The -y option reads a device file from the specified path.

[Application]

- Specify a path where a device file exists.

[Explanation]

- An abort error occurs if anything other than a path name is specified after the -y option.
- An abort error occurs if the path name is omitted after the -y option.
- The path from which the device file is read in the order determined as follows.

  (i) Path specified by the -y option

  (ii) Path registered in the device file installer

  (iii) Path by which RA78K0R was started up

  (iv) Current folder

  (v) The environmental variable PATH

[Example of use]

- To specify the path for the device file as folder C:\78k0r\dev, describe as:

```
C>ra78k0r k0rmain.asm -cf1166a0 -yC:\78k0r\dev
```

- To specify the path for the device file as folder C:\Program Files\NEC Electronics Tools\device files, describe as:

```
C>ra78k0r k0rmain.asm -cf1166a0 -y"C:\Program Files\NEC Electronics
Tools\device files"
```

Symbol definition specification

(1) -d

[Syntax]

```
-dsymbol-name[=value][,symbol-name[=value]...]
```

- Default assumption

  None

[Function]

- The -d option defines symbols.

[Application]

- Specify the -d option when defining symbols.

[Explanation]

- The value given to a symbol is binary, octal, decimal, or hexadecimal. When value specification is omitted, 1 will be specified.
- Up to 30 symbols can be specified by using a comma as a delimiter.
- Up to 31 characters can be described for a symbol name.
- When duplicate names are specified, the latest one specified is valid.
- Symbol names are case sensitive.
  Symbols defined with -d are used instead of EQU/$SET/$RESET. An abort error occurs if a symbol name specified for -d was also defined in the source.

[Example of use]

- To specify 2 as the symbol definition, describe as:

```
C>ra78k0r k0rmain.asm -cf1166a0 -dSYM=2
```

78K0R common object specification

(1) -common

[Syntax]

```
-common
```

- Default assumption

  The object file supporting the specified device is output.

[Function]

- The -common option specifies output of an object module file common to the 78K0R.

[Application]

- This option generates an object code that can be used commonly in the 78K0R, regardless of the device type specification option (-c).

  The output object module file can be linked with an object file for which a different device in the 78K0R is specified.

[Explanation]

- Specify this option to generate an object code that can be used commonly in the 78K0R.

[Notice]

- Even when the -common option is specified, the device type specification option (-c) or control instruction of the same function must not be omitted.

  An abort error occurs if the 78K0R common object specification option (-common) is specified for all the input object module files to be linked.

[Example of use]

- To generate an object code that can be used commonly in the 78K0R, describe as:

```
C>ra78k0r k0rsub.c -cf1166a0 -common
```

Self programming specification

(1) -self

[Syntax]

```
-self
```

- Default assumption

  None

[Function]

- The -self option stops an error occurring when "CALL! xxxxxH" is described even if address xxxxxH is outside the access range (i.e., there is no internal ROM).

  For address xxxxxH, refer to the user's manual of the device used.

[Application]

- Specify the -self option when using self programming.

[Explanation]

- Specify this option if an error occurs when "CALL! xxxxxH" is described during self programming.

[Example of use]

- In order not to output errors for the description of "CALL ! xxxxxH" during self programming, describe as:

```
C>ra78k0r k0rsub.asm -cf1166a0 -self
```

78K0 compatible macro

(1) -compati/-ncompati

[Syntax]

```
-compati
-ncompati
```

- Default assumption

  -ncompati

[Function]

- The -compati option enables assembly of assembler source files generated by the 78K0 assembler.
- Option -ncompati makes the -compati option unavailable.

[Application]

- A fatal error (E2337) will result if an attempt is made to assemble an assembler source that includes instructions for the 78K0, which normally cannot be used for the 78K0R.

  Specify the -compati option to assemble assembler sources without changing the following 78K0 instructions that cannot be used for the 78K0R.

  78K0 instructions that cannot be used for 78K0R:

  DIVUW, ROR4, ROL4, ADJBA, ADJBS, CALLF, DBNZ

[Explanation]

- When the -compati option is specified, the RA78K0R includes file "..\inc78k0r\compati.inc" (for the path through which ra78k0r.exe is activated) and performs macro conversion for 78K0 instructions that cannot be used for 78K0R.

[Example of use]

- To perform macro conversion for 78K0 instructions that are removed from the 78K0R, describe as:

```
C>ra78k0r k0rsub.asm -cf1166a0 -compati
```

Help specification

(1) --

[Syntax]

```
--
```

- Default assumption

  No display

[Function]

- The -- 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]

- When the -- option is specified, all other options are unavailable.
- 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+.
  To reference PM+ help, click the [Help] button in the [Assembler Options] dialog box.

[Example of use]

- To output a help message on the display, describe as:

```
C>ra78k0r --
```

```
78K0R Assembler Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

usage : ra78k0r [option [...]] input-file [option [...]]
The option is as follows ([] means omissible).
-cx           : Select target chip. (x = f1166a0 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].
-common       : Create the common object module file for 78K0R.
-self         : Use Self-programming.
-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 -ll0 -nlf -lt8 -g -nj -ga
```

4.5 Option Settings in PM+

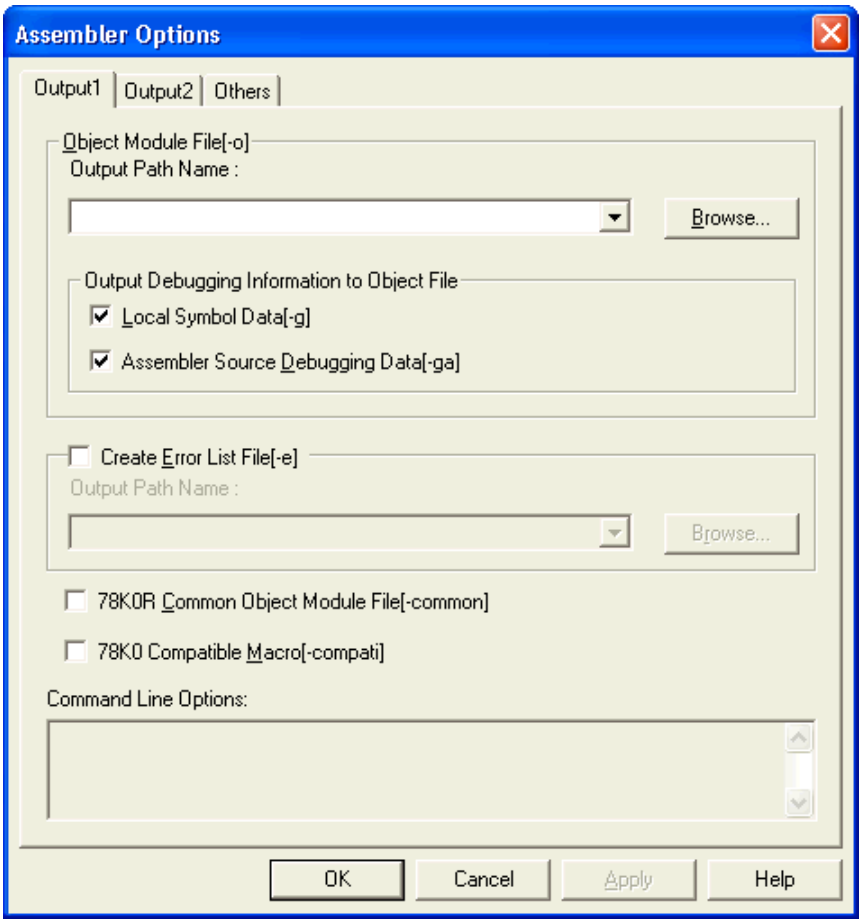
This section describes the method for setting assembler options from PM+.

4.5.1 Option setting method

The [Assembler Options] dialog box is opened if [Assembler Options] is selected from the [Tools] menu of PM+ or if the [RA] button on the toolbar is clicked.

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

Figure 4-2 [Assembler Options] Dialog Box

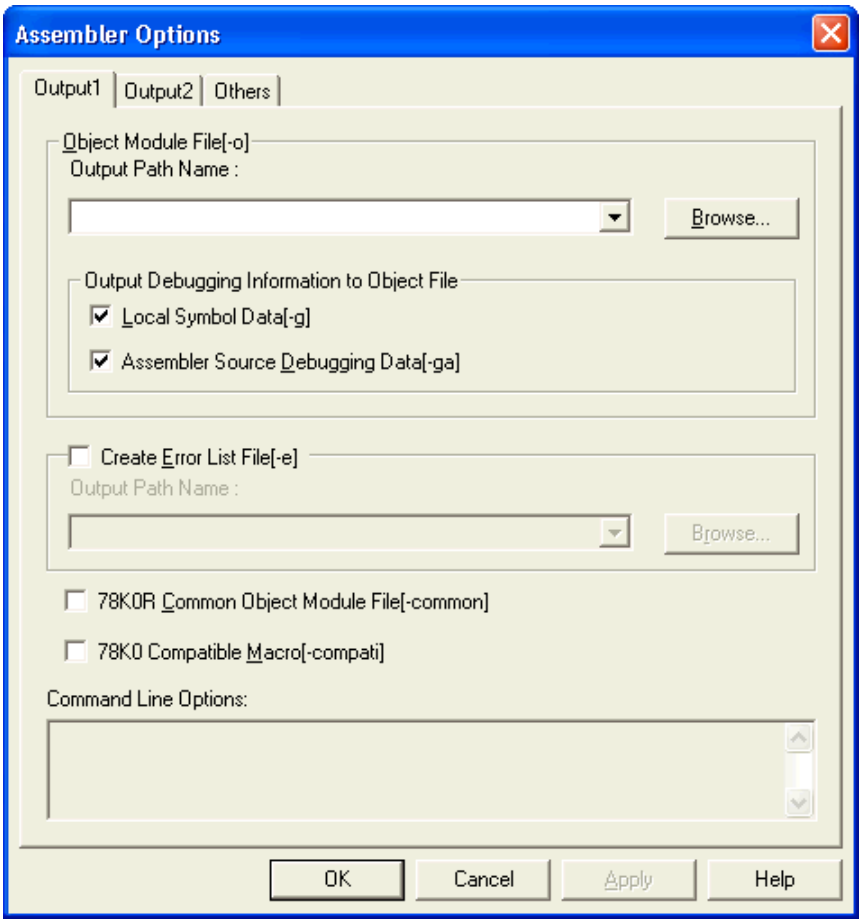


4.5.2 Explanation of dialog box

The various tabs in the [Assembler Options] dialog box are described below.

(1) [Output1] tab

Figure 4-3 [Assembler Options] Dialog Box (When [Output1] Tab Is Selected)



- Object Module File[-o]

(When specified by common option) Output Path Name	Specify the path of the object module file by using the [Browse...] button or directly inputting a path name. The number of characters that can be input is up to 259.
(When specified by individual option) Output File Name	Specify the path and file name of the object module file by using the [Browse...] button or directly inputting a path and file name.

- Output Debugging Information to Object File

Local Symbol Data[-g]	Select this option to add debug data (local symbol data) to the object module file (by default).
Assembler Source Debugging Data[-ga]	Select this option to add source debug data to the object module file (by default).

- Create Error List File[-e]

  Select this option to output an error list file.

(When specified by common option) Output Path Name	Specify the path of the error list file by using the [Browse...] button or directly inputting a path name. The number of characters that can be input is up to 259.
(When specified by individual option) Output File Name	Specify the path and file name of the error list file by using the [Browse...] button or directly inputting a path and file name. The number of characters that can be input is up to 259.

- 78K0R Common Object Module File[-common]

  Select this option to output an object module file common to the 78K0R.

- 78K0 Compatible Macro[-compati]

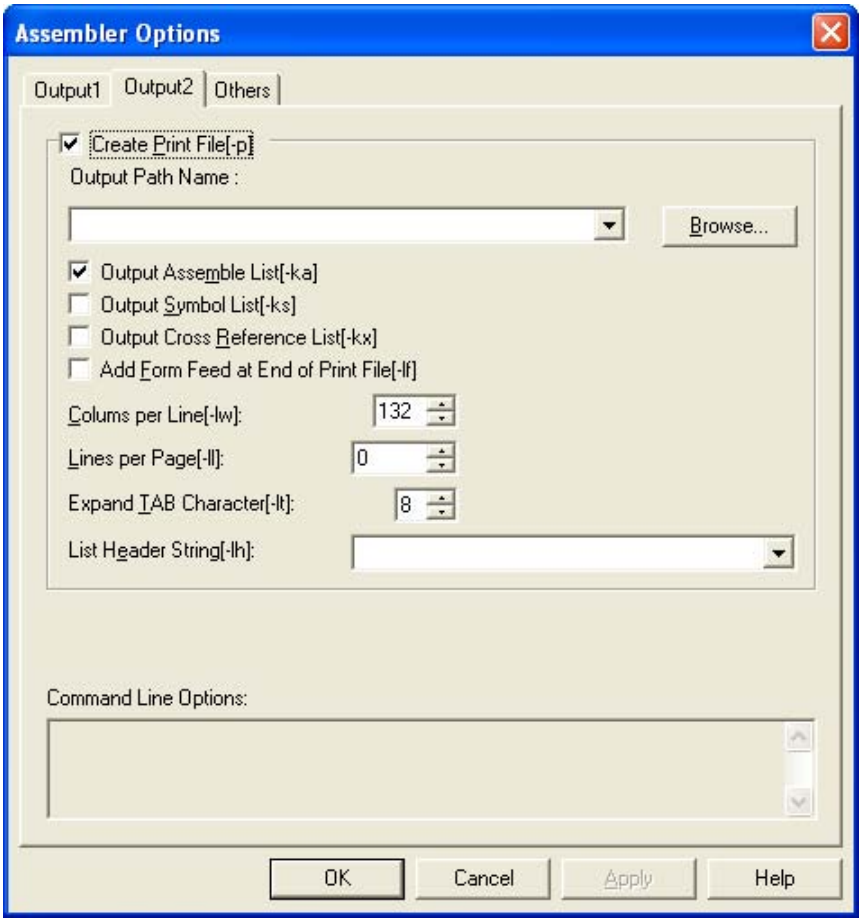
  Select this option to enable assembly of assembler source files generated by the 78K0 assembler.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

(2) [Output2] tab

Figure 4-4 [Assembler Options] Dialog Box (When [Output2] Tab Is Selected)



- Create Print File[-p]

  Select this option to output an assemble list file (by default).

(When specified by common option) Output Path Name	Specify the path of the assemble list file by using the [Browse...] button or directly inputting a path name. The number of characters that can be input is up to 259.
(When specified by individual option) Output File Name	Specify the path and file name of the assemble list file by using the [Browse...] button or directly inputting a path and file name. The number of characters that can be input is up to 259.

- Output Assemble List[-ka]

  Select this option to output an assemble list in the assemble list file (by default).

- Output Symbol List[-ks]

  Select this option to output a symbol list file, following the assemble list, in the assemble list file.

- Output Cross Reference List[-kx]

  Select this option to output a cross reference list, following the assemble list, in the assemble list file.

- Add Form Feed at End of Print File[-lf]

  Select this option to suffix a form feed code (FF) to the assemble list file.

- Colums per Line[-lw]

  Specify the number of characters on one line of the assemble list file, in a range of 72 to 2,046. 132 is specified by default.

- Lines per Page[-ll]

  Specify the number of lines on one page of the assemble list file, 0 or in a range of 20 to 32,767. 0 is specified by default.

- Expand TAB Character[-lt]

  Specify the length of the tab character, in a range of 0 to 8. 8 is specified by default.

- List Header String[-lh]

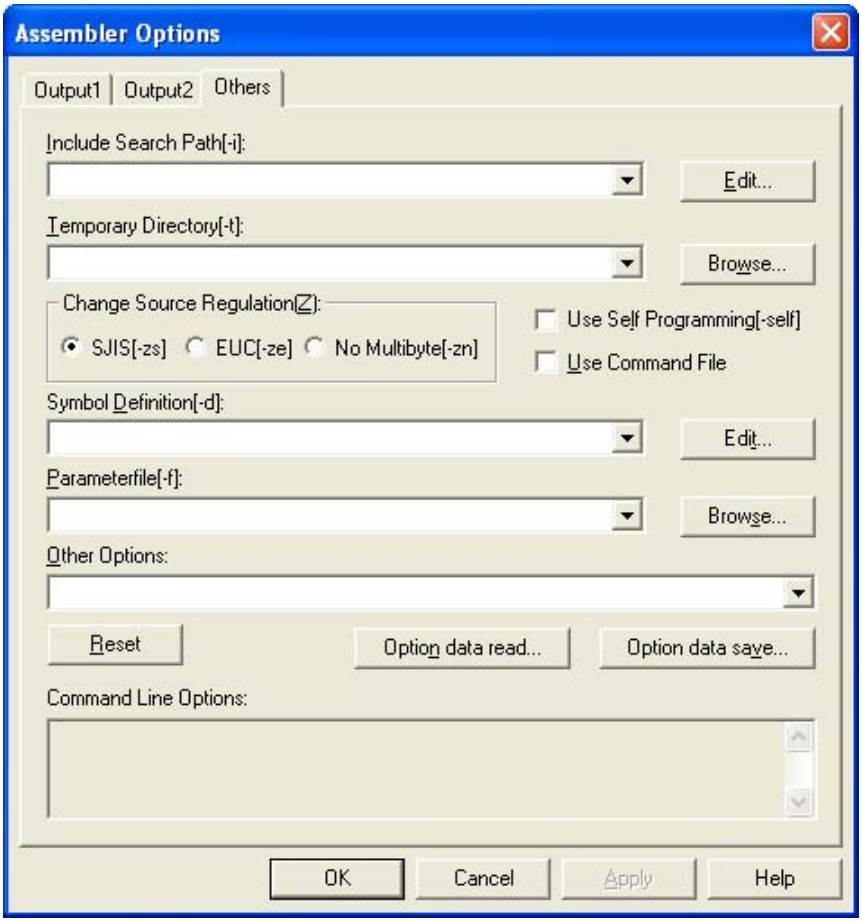
  Specify the character string to be printed in the title column of the header of the assemble list file.
  The number of characters that can be input is up to 60.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

(3) [Others] tab

Figure 4-5 [Assembler Options] Dialog Box (When [Others] Tab Is Selected)



- Include Search Path[-i]

  Specify the path via which the include file is to be read by using the [Edit...] button (opens the [Edit Option] dialog box) or directly inputting a path name.

  Up to 64 items can be specified by using commas as delimiters.

  The number of characters that can be input is up to 259.

  Remark If "Link STDIO Library for SM+[S]" is selected in the [SM+ Options] dialog box of SM+ for 78K0R, the standard I/O library dedicated to SM+ is displayed in the Command Line Options area. In the same manner, if a library has been specified with the RX78K0R, the library specified by the RX78K0R is displayed in the Command Line Options area.

  A value from which the number of libraries required by the SM+ for 78K0R and RX78K0R is subtracted can be specified as the number of files.

- Temporary Directory[-t]

  Specify the path where a temporary file is to be created by using the [Browse...] button or directly inputting a path name.

- Change Source Regulation[Z]

  This option selects the type of kanji code (2-byte code) (SJIS[-zs], EUC[-ze], or No Multibyte [-zn]) to be used in the comment of the source.

  “SJIS[-zs]“ is selected by default.

- Use Self Programming[-self]

  Select this option to use the self programming function.

- Use Command File

  Select this option to create a command file.

- Symbol Definition[-d]

  Input a numeric value to be defined as a symbol by using the [Edit...] button (opens the [Edit Option] dialog box) or directly inputting a value.

  Up to 30 items can be specified by using commas as delimiters. Up to 31 characters can be used for specifying a symbol.

- Parameterfile[-f]

  Specify the file to be input as a user-defined parameter file by using the [Browse...] button or directly inputting a file name.

  The number of characters that can be input is up to 259.

- Other Options

  To specify an option other than the options that can be set in this dialog box, enter the option in the input box.

  The number of characters that can be input is up to 259.

  Caution The help specification option (--) cannot be specified on PM+.

- [Reset] button

  Resets the input contents.

- [Option data read....] button

  Opens the [Read Option Data] dialog box and after the option data file has been specified, reads this file.

- [Option data save...] button

  Opens the [Save Option Data] dialog box and save the option data to the option data file with a name.

- Command Line Options

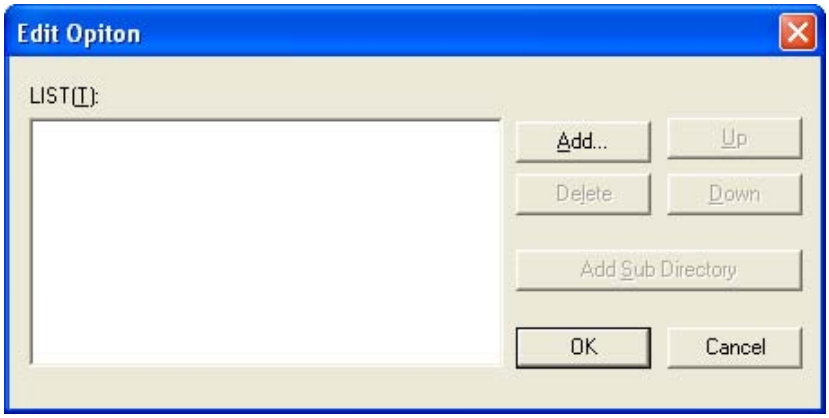
  This edit box is read-only. The currently set option character string is displayed.

4.5.3 [Edit Option] dialog box

Items are edited in list format in the [Edit Option] dialog box.

The [Edit Option] dialog box is described below.

Figure 4-6 [Edit Option] Dialog Box



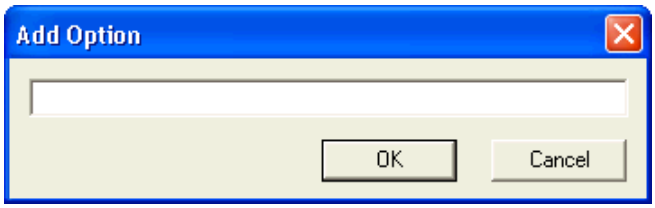
- [Add...] button

  Adds a list item.

  If the item to be added is a file or folder, the corresponding [Browse for Folder] dialog box opens.

  In all other cases, the [Add Option] dialog box opens. Specify details of the item to be added in this box.

Figure 4-7 [Add Option] Dialog Box



- [Delete] button

  Deletes the selected list item.

- [Up] button

  Moves the selected list item up.

- [Down] button

  Moves the selected list item down.

- [Add Sub Directory] button

  A subfolder can be added to the selected list item when the item is specified as “Include Search Path [-i]” on the [Others] tab.

CHAPTER 5 LINKER

The linker inputs a number of object module files output by the 78K0R assembler, determines a location address and outputs them as a single load module file.

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

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

5.1 I/O Files of Linker

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

Table 5-1 I/O Files of Linker

Type	File Name	Explanation	Default File Type
Input files	Object module files	- Binary files containing relocation data and symbol data regarding machine language data and machine language location addresses - Files output by the assembler	.rel
	Library files	- Files in which two or more object module files are included - Files output by the librarian	.lib
	Directive files	- Files which contain link directives used during linking (user-created files)	.dr
	Parameter files	- Files containing the parameters for the executed programs (user-created files)	.plk
Output files	Load module files	- 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	- List files which display the result of linking	.map
	Error list files	- Files containing error data generated during linking	.elk
I/O files	Temporary files	- Files created automatically by the linker for linking purposes Temporary files are deleted when linking ends.	LKxxxxx.$$n (n = 1 to 3)

5.2 Functions of Linker

(1) Joining of input segments

The linker determines and controls the location address of each segment.

The linker identifies identical segments and joins them into a single segment, even if they are in separate object module files.

(2) Determination of input modules

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

(3) Determination of location addresses for input segments

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

(4) Correction of object codes

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

5.3 Memory Spaces and Memory Areas

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

Memory area: Each memory space is divided into several memory areas.
The memory area declares the memory addresses for the installed memory.

Caution Only one memory space can be defined. No name other than "REGULAR" can be specified as the memory space.

Table 5-2 Segment Allocation Groups (External ROM, etc.)

Memory Area Name	Default Address	Segments Allocated by Default
ROM	Internal ROM (up to the start address of the RAM area in the case of ROMless products)	CSEG
RAM	Internal RAM	DSEG, BSEG

Remark 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.

Remark 2 For specific examples, refer to "3.3 Execution Procedure from Command Line (5)".

5.4 Link Directives

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

Two types of directives can be used as follows.

The role of the directives is to:

Directive Type	Roles
Memory directives	- Declare addresses in the installed memory - Divide memory into two or more areas [Example] CALLT area Internal ROM External ROM SADDR Internal RAM other than SADDR
Segment location directives	- Segment allocation is specified by the linker The following items are specified for each segment. - Absolute address - Specification of memory address only

Create a directive file (a file which describes directives) using an editor, and then specify the -d option when the linker is started.

The linker then reads the directives from the file and interprets them to perform linking.

5.4.1 Directive files

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

- Memory directives

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

- Segment allocation directives

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

  MERGE *segment-name* : [*merge-attribute*][=*memory-area-name-specification*][/*memory-space-name*]

A number of directives can be specified in a single directive file.

For details of each directive, refer to “5.4.2 Memory directives“ and “5.4.3 Segment location directives“.

(1) Reserved words

The following words are reserved words in a directive file.

Reserved words cannot be used in a directive file for other meanings (segment name, memory area name, etc.).

Reserved Words	Explanation
MEMORY	Specifies a memory directive.
MERGE	Specifies a segment location directive.
AT	Specifies a relocation attribute of a segment location directive (start address).
SEQUENT	Specifies a merge attribute of a segment location directive (segments are merged).
COMPLETE	Specifies a merge attribute of a segment location directive (segments are not merged).

Caution Reserved words can be written in uppercase or lowercase characters, but not in a mixture of the two.

[Example]

MEMORY: Can be used

memory: Can be used

Memory: Cannot be used

(2) Symbols

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

(3) Numerical values

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

[Example]

```
23H, 0FC80H
```

(4) Comments

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

[Example]

The underlined portion is a comment.

```
; DIRECTIVE FILE FOR 78F1166_A0
MEMORY MEM1 : ( 40000H , 10000H ) #SECOND MEMORY AREA
```

5.4.2 Memory directives

A memory directive is a directive which defines a memory area (name of an address in the installed memory).

The name of a defined memory area (the memory area name) is used to reference a segment location directive.

Up to 100 memory areas can be defined, including the default memory area.

[Syntax]

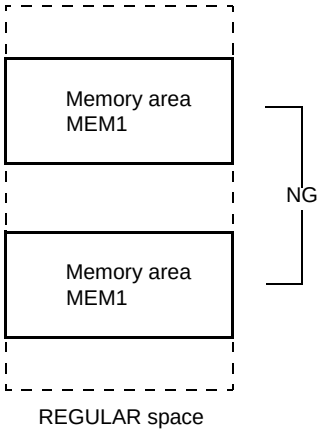
```
MEMORY∆memory-area-name∆:∆(∆start-address-value∆,∆size∆)[/∆memory-space-name]
```

(1) Memory area names

Specify a name for the defined memory area.

Conditions for specification of memory area names are as follows.

- The characters which can be used to describe a memory area name are A-Z, a-z, 0-9, _, ?, and @.
  However, a memory area name cannot begin with 0-9.
- Uppercase and lowercase characters are interpreted as separate characters.
- Uppercase and lowercase characters can be mixed together.
- Maximum length of a memory area name is 256 characters.
  If 257 or more characters are described, an error occurs.
- 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.



(2) Start addresses

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

Describe a numerical value from 0H to FFFFFH.

(3) Size

Specify the size of the memory area to be defined.

Specification conditions are as follows.

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

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

(4) Memory space names

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

The following conditions on specification of memory space names apply.

- Memory space names must be specified entirely in uppercase characters.
- When a memory space name is omitted, REGULAR is assumed to be specified.
- If the memory space name is omitted after "/" is written, an error occurs.

[Function]

- Define a specified memory space for a memory area specified with a memory area name.
- 1 memory area can be defined with 1 memory directive.
- A memory directive can be specified more than once. However, multiple definitions in the specified order will result in an error.
- 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.
- If you wish to use a different memory area without using the default memory space, specify the size of the default area name as "0".

[Example of Use]

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

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

5.4.3 Segment location directives

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

[Syntax]

```
MERGE Δsegment-nameΔ:Δ[ATΔ(Δstart-addressΔ)][Δ=Δmemory-area-name][Δ/Δmemory-space-name]
MERGE Δsegment-nameΔ:Δ[merge-attribute][Δ=Δmemory-area-name][Δ/Δmemory-space-name]
```

(1) Segment name

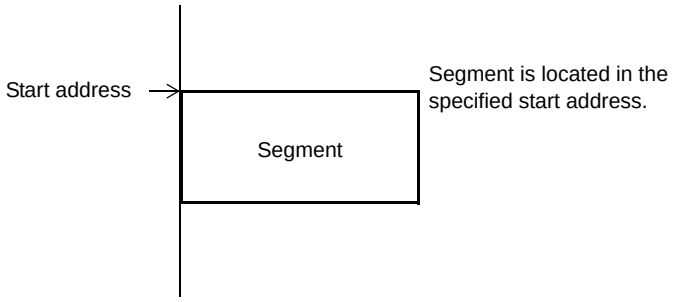
The segment name is the name of a segment included in an object module file input to the linker.

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

(2) Start address

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

- 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.
- The start address specifies a numerical constant.



Caution 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 occurs.

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

(3) Merge attribute

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

SEQUENT	Merges the segments in the order in which they appear, so that no gaps are created. BSEG merges the segments in bit units in the order in which they appear.
COMPLETE	An error occurs if two or more segments with the same name exist.

[Example]

```
MERGE DSEG1 : COMPLETE=RAM
```

(4) Memory space names

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

- Only REGULAR can be specified as a memory area name.
- Memory space names must be specified entirely in uppercase characters.
- When a memory space name is omitted, REGULAR is assumed to be specified.

Segment location destinations are determined as follows.

Memory Area	Memory Space	Segment Location Destination
No specification	No specification	Default memory area in the REGULAR space
Memory area name	No specification	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.

For example, if the memory space name "REGULAR" is specified for a segment with the relocation characteristic "CSEG BASE", the segment will be located to fit within 0000H to FFFFH.

[Notice]

- The location address of an input segment for which no segment location directive is specified will be determined according to the relocation characteristics specified by a segment definition directive during assembly.
- An abort error occurs if no segment exists for which a segment name has been specified.
- An abort error occurs if more than one segment location directive is specified for the same segment.

[Example of Use]

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

```
MEMORY ROM : ( 0000H , 1000H )
MEMORY MEM1 : ( 1000H , 2000H )
```

- When input segment SEG1 is allocated to 500H in memory area ROM (refer to the following figure (1)).

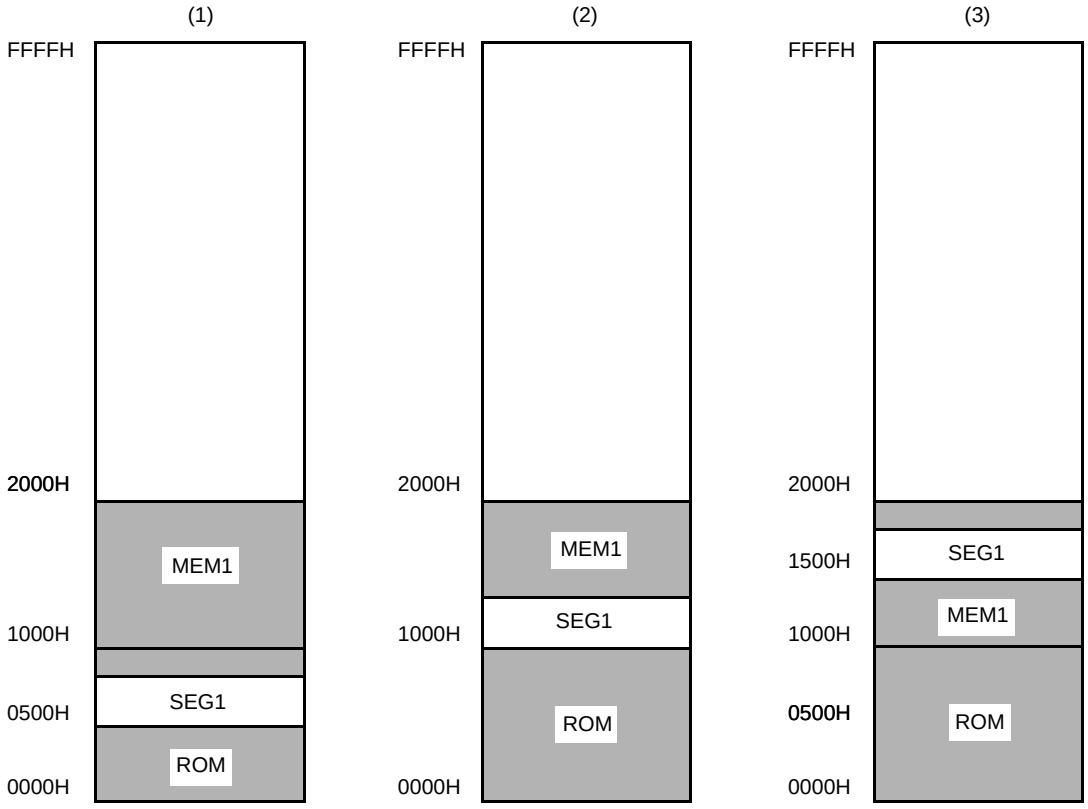
```
MERGE SEG1 : AT (500H)
```

- When input segment SEG1 is allocated to memory area MEM1 (refer to the following figure (2)).

```
MERGE SEG1 : =MEM1
```

- When input segment SEG1 is allocated to 1500H in memory area MEM1 (refer to the following figure (3)).

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



5.5 Linker Startup

5.5.1 Methods to start linker

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

(1) Startup from the command line

```
 X>[path-name]lk78k0r[Δoption]...object-module-file-name[Δoption]...[Δ]
  |       |       |       |                    |               |
 (a)     (b)     (c)     (d)                  (e)             (d)
```

(a) Current drive name

(b) Current folder name

(c) Linker command file name

(d) This contains detailed directions for the action of the linker.

If more than one linker option is specified, separate the options with a space. Uppercase characters and lowercase characters are not distinguished for the linker options. For a detailed explanation of the linker options, refer to "5.6 Linker Options".

Enclose a path that includes a space in a pair of double quotation marks (" ").

(e) Object module file to be linked

A maximum of 1,024 items can be input in an input module.

Specify the file name of a path that includes a space by enclosing it in a pair of double quotation marks (" ").

[Example]

- To add debug data to a load module file (k0r.lmf), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -ok0r.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 repeating the same linker option for two or more assembly operations.

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

Start up the linker from a parameter file as follows.

```
 X>lk78k0r[Δobject-module-file]Δ-fparameter-file-name
 |       |                        |          |
(a)     (b)                      (c)        (d)
```

(a) Current drive name

(b) Current folder name

(c) Parameter file specification option

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

Remark 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 object module file name is omitted from the command line, specify the object module file name in the parameter file.
- The object module file name can also be written after the option.
- Write in the parameter file all linker options and output file names that should be specified in the command line. For a detailed explanation of parameter files, refer to "3.4 Using Parameter File".

[Example]

(1) Create the parameter file (k0r.plk) using an editor.

```
; parameter file
k0rmain.rel k0rsub.rel -ok0r.lmf -pk0r.map -e
-tC:\tmp
```

(2) Use parameter file k0r.plk to start up the linker.

```
C>lk78k0r -fk0r.plk
```

5.5.2 Execution start and end messages

(1) Execution start message

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

```
78K0R Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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]

<A non-existent object module file is specified.>

```
C>lk78k0r samp1.rel samp2.rel
```

```
78K0R Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F3006 : File not found 'samp1.REL'
RA78K0R error F3006 : File not found 'samp2.REL'
Program Aborted.
```

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

<A non-existent linker option is specified.>

```
C>lk78k0r k0rmain.rel k0rsub.rel -z
```

```
78K0R Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F3018 : Option is not recognized '-z'
Please enter 'LK78K0R --', if you want help messages.
Program Aborted.
```

In the above example, a non-existent linker option is specified. An error occurs and the linker aborts the link.

When an error message is displayed and link is aborted, look for the cause in "CHAPTER 11 ERROR MESSAGES" and take action accordingly.

5.6 Linker Options

5.6.1 Types of linker options

The linker options are detailed instructions for the operation of the linker.

The types and explanations for linker options are shown below.

Table 5-3 Linker Options

Classification	Option	Explanation
Load module file output specification	-o	Specifies the output of a load module file.
	-no	
Forced load module file output specification	-j	Forces output of a load module file.
	-nj	
Debug data output specification	-g	Outputs debugging data to a load module file.
	-ng	
Generation of stack decision symbols specification	-s	Automatically generates public symbols for stack decision.
	-ns	
Directive file specification	-d	Inputs the specified file as a directive file.
Link list file output specification	-p	Specifies output of a link list file.
	-np	
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	
Link list format specification	-ll	Changes the number of lines that can be printed in 1 page in a link list file.
	-lf	Inserts a page feed code at the end of a list file.
	-nlf	
Error list file output specification	-e	Outputs an error list file.
	-ne	
Library file specification	-b	Inputs the specified file as a library file.
Library file read path specification	-i	Reads a library file from a specified path.
Parameter file specification	-f	Inputs file names and options from a specified file.

Classification	Option	Explanation
Specification of path for temporary file creation	-t	Creates a temporary file in a specified path.
Device file search path specification	-y	Reads a device file from a specified path.
Warning message output specification	-w	Specifies whether or not to output a warning message to the console.
Link specification of boot area ROM program of flash memory model	-zb	Specifies the first address of the flash memory area.
On-chip debug option byte specification	-go	Specifies the on-chip debug option byte.
Security ID specification	-gi	Specifies a security ID.
User option byte specification	-gb	Specifies the value set for the user option byte.
Mirror area specification	-mi	Specifies the location destinations of segments in the mirrored area.
64 KB boundary location specification	-ccza	Specifies whether to locate a segment to the last byte of each 64 KB boundary area.
Help specification	--	Displays a help message on the display.

5.6.2 Order of precedence of linker options

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

Table 5-4 Order of Precedence of Linker Options

	-no	-ng	-np	-nkm	-nkp	-nkl	--
-j	NG						NG
-g	NG						NG
-p				Δ	Δ	Δ	NG
-km			NG				NG
-kd			NG	NG			NG
-kp		NG	NG				NG
-kl		NG	NG				NG
-ll			NG				NG
-lf			NG				NG

[Items marked with an NG]

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

<Example>

```
C>lk78k0r k0rmain.rel k0rsub.rel -np -km
```

The -km option 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>lk78k0r k0rmain.rel k0rsub.rel -p -nkm -nkp -nkl
```

The -nkm, -nkp, and -nkl options are all specified at the same time, so -p option 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>lk78k0r k0rmain.rel k0rsub.rel -o -no
```

The -no option is specified after -o, so -o option is unavailable and -no is available.

Options not specified in Table 5-4 have no particular effect on other options. However, when the help option (--) is specified, all other options become unavailable.

Load module file output specification

(1) -o/-no

[Syntax]

```
-o[output-file-name]
-no
```

- Default assumption

  -o*input-file-name*.lmf

[Function]

- The -o ption specifies the output of a load module file. It also specifies the location to which it is output and the file name.
- The -no option makes the -o, -j, and -g option unavailable.

[Application]

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

[Explanation]

- The disk type file name and device type file name, NUL and AUX can be specified as output file names.
- Even if the -o option is specified, if a fatal error occurs the load module file cannot be output.
- If *output-file-name* is omitted when the -o option is specified, the load module file *input-file-name*.lmf will be output to the current folder.
- If only the path name is specified in *output-file-name*, *input-file-name*.lmf will be output to the specified path.
- If both options -o and -no are specified at the same time, the option specified last takes precedence.

[Example of use]

- To output a load module file (k0r.lmf), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -ok0r.lmf
```

Forced load module file output specification

(1) -j/-nj

[Syntax]

```
-j
-nj
```

- Default assumption

  -nj

[Function]

- The -j option specifies that the load module will be output even if a fatal error occurs.
- The -nj option makes the -j option 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 the -j option to output the load module file.

[Explanation]

- When the -j option is specified, the load module will be output even if a fatal error occurs.
- If both options -j and -nj are specified at the same time, the option specified last takes precedence.
- If the -no option is specified, the -j option is unavailable.

[Example of use]

- To output a load module file (k0rsub.lmf) even if a fatal error occurs, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -j
```

Debug data output specification

(1) -g/-ng

[Syntax]

```
-g
-ng
```

- Default assumption

  -g

[Function]

- The -g option specifies that debugging data (local symbol data) is to be added to a load module file.
- The -ng option makes the -g, -kp, and -kl option unavailable.

[Application]

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

[Explanation]

- When the -ng option is specified, the public symbol list and local symbol list cannot be output.
- If both options -g and -ng are specified at the same time, the option specified last takes precedence.
- If the -no option is specified, the -g option is unavailable.

[Example of use]

- To add debug data to a load module file (k0rsub.lmf), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -g
```

Generation of stack decision symbols specification

(1) -s/-ns

[Syntax]

```
-s[area-name]
-ns
```

- Default assumption

  -ns

[Function]

- The -s option generates the stack decision public symbols "_@STBEG" and "_@STEND".
- The -ns option makes the -s option unavailable.

[Application]

- Specify the -s option to reserve a stack area.

[Explanation]

- *area-name* is a name in which an area memory name defined by the user or an area memory name defined by default is specified.
- Area-names distinguish between uppercase and lowercase characters.
- The linker searches the memory area specified by the -s option 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.

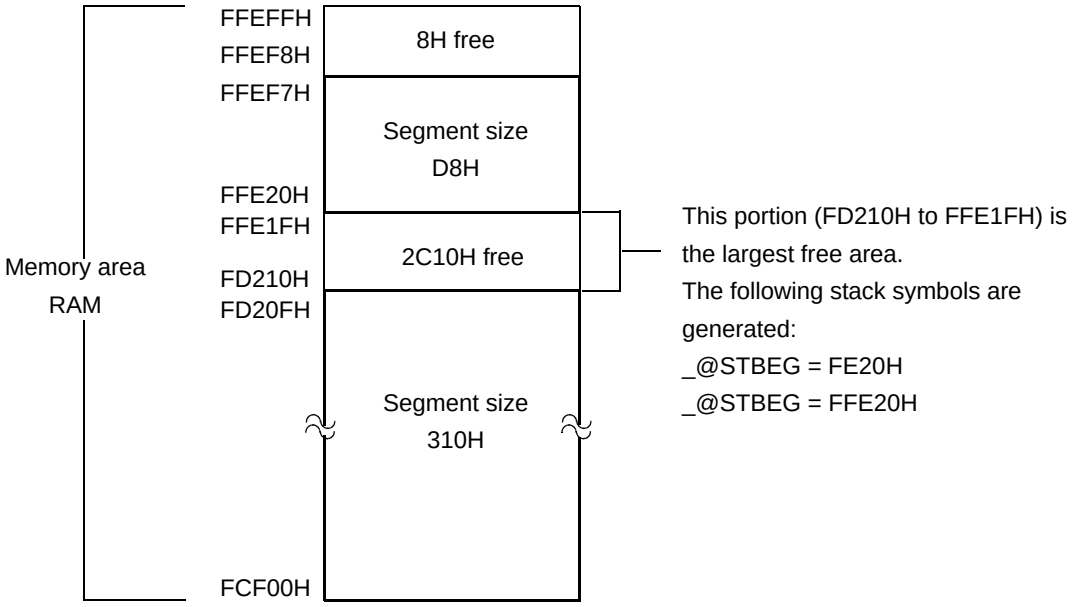
  These symbols are handled as publicly declared NUMBER attribute symbols, and are registered at the end of the linker's symbol table. When these symbols are output to a link list file, the module name column is left blank.
- If the largest open area is 10 bytes or smaller, a warning message is output.
- If no free area exists, a warning message is output and both "_@STEND" and "_@STBEG" hold the last address +1 as their values.
- If an area name is omitted, "RAM" is specified.
- If both options -s and -ns are specified at the same time, the option specified last takes precedence.

[Example of use]

- To reserve the stack area in memory area RAM, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -s
```

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.



Directive file specification

(1) -d

[Syntax]

```
-dfile-name
```

- Default assumption

  None

[Function]

- The -d option 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 the -d option to input this directive file to the linker.

[Explanation]

- Only disk-type file names can be specified as a file name. If a device-type file name is specified, an abort error occurs.
- If *file-name* is omitted, an abort error occurs.
- Nesting of directive files is not permitted.
- The number of characters that can be specified in a directive file is unlimited.
- An abort error occurs if the -d option is specified more than once, or if more than one file name is specified.
- For a detailed explanation of directive files, refer to "5.4 Link Directives".

[Example of use]

- Redefine the default memory area ROM/RAM.

  <Contents of the directive file k0r.dr>

```
memory ROM : ( 0H , 40000H )
memory RAM : ( 0FCF00H , 3000H )
```

  To link the directive file (k0r.dr), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -dk0r.dr
```

Link list file output specification

(1) -p/-np

[Syntax]

```
-p[output-file-name]
-np
```

- Default assumption

  -p*input-file-name*.map

[Function]

- The -p option specifies output of a link list file. It also specifies the destination and file name of the output file.
- The -np option makes the -p, -km, -kd, -kp, -kl, -ll, and -lf option unavailable.

[Application]

- Specify the -p option to change the output destination or output file name of a link list file.
- Specify the -np option when performing link only to output a load module file. This will shorten link time.

[Explanation]

- 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 *output-file-name* is omitted when the -p option is specified, the link list file name in the current folder becomes *input-file-name*.map.
- If only *output-file-name* is specified, *input-file-name*.map is output to the specified path.
- If both options -p and -np are specified at the same time, the option specified last takes precedence.

[Example of use]

- To create a link list file (k0r.map), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -pk0r.map
```

Link list file data specification

(1) -km/-nkm

[Syntax]

```
-km
-nkm
```

- Default assumption

  -km

[Function]

- The -km option outputs a map list into a link list file.
- The -nkm option makes the -km option unavailable.

[Application]

- Specify the -km option to output a map list to a link list file.

[Explanation]

- If the -nkm, -nkp, and -nkl options are all specified, the link list file cannot be output.
- If the -nkm option is specified, the link directive file cannot be output to a link list file.
- If both the -km and -nkm options are specified at the same time, the option specified last takes precedence.
- If the -np option is specified, the -km option is unavailable.

[Example of use]

- To output a map list into a link list file (k0r.map), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -s -pk0r.map -km
```

<Contents of k0r.map>

```
78K0R Linker Vx.xx                      Date:xx xxx xx  Page:1

Command : k0rmain.rel k0rsub.rel -s -pk0r.map -km
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file :
Directive :

*** Link information ***

    4 output segment(s)
  5FH byte(s) real data
   41 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 00000H   SIZE = 40000H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
        CODE                    00000H  00002H  CSEG    AT
                CODE    SAMPM   00000H  00002H
* gap *                          00002H  000BEH
        ?CSEGOB0                000C0H  00004H  CSEG    OPT_BYTE
        ?CSEG                   000C4H  00059H  CSEG
                ?CSEG   SAMPM   000C4H  00017H
                ?CSEG   SAMPS   000DBH  00042H
* gap *                          0011DH  3FEE3H

  MEMORY = LRAM
  BASE ADDRESS = FCF00H   SIZE = 03100H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
* gap *                          FCF00H  02F20H
        DATA                    FFE20H  00003H  DSEG    AT
                DATA    SAMPM   FFE20H  00003H
* gap *                          FFE23H  000DDH
* gap ( Not Free Area ) *        FFF00H  00100H
```

Map list

(2) -kd/-nkd

[Syntax]

```
-kd
-nkd
```

- Default assumption

  -kd

[Function]

- The -kd option outputs a link directive file into a link list file.
- The -nkd option makes the -kd option unavailable.

[Application]

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

[Explanation]

- If the -nkm, -nkp, and -nkl options are all specified, a link list file cannot be output.
- If the -nkm option is specified, a link directive file cannot be output into a link list file.
- If both options -kd and -nkd are specified at the same time, the option specified last takes precedence.
- If the -np option is specified, the -kd option is unavailable.

[Example of use]

- To output a link directive file into a link list file (k0r.map), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -s -dk0r.dr -pk0r.map -kd
```

<Contents of k0r.map>

```
78K0R Linker Vx.xx                    Date:xx xxx xx  Page:1

Command : k0rmain.rel k0rsub.rel -s -dk0r.dr -pk0r.map -kd
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file : k0r.dr                          <-- Directive file name
Directive : MEMORY ROM : ( 0H , 0ED800H )   <-- Contents of directive file
            MEMORY RAM : ( 0FCF00H , 1100H )
            MEMORY RAM : ( 0FE000H , 1F00H )

*** Link information ***

    6 output segment(s)
  9DH byte(s) real data
   40 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 00000H   SIZE = ED800H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
        CODE                    00000H  00002H  CSEG    AT
          :
```

(3) -kp/-nkp

[Syntax]

```
-kp
-nkp
```

- Default assumption

  -nkp

[Function]

- The -kp option outputs a public symbol list into a link list file.
- The -nkp option makes the -kp option unavailable.

[Application]

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

[Explanation]

- If -nkm, -nkp, and -nkl options are all specified, the link list file cannot be output.
- If -ng options is specified, the public symbol list cannot be output.
- If both options -kp and -nkp are specified at the same time, the option specified last takes precedence.
- If the -np option is specified, the -kp option is unavailable.

[Example of use]

- To output a public symbol list into a link list file (k0r.map), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -s -g -pk0r.map -kp
```

<Contents of k0r.map>

```
78K0R Linker Vx.xx                      Date:xx xxx xx  Page:1

Command : k0rmain.rel k0rsub.rel -s -g -pk0r.map -kp
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file :
Directive :

*** Link information ***

    6 output segment(s)
  9DH byte(s) real data
   40 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 00000H   SIZE = 40000H
    :

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

78K0R Linker Vx.xx                      Date:xx xxx xx  Page:2

*** Public symbol list ***

MODULE  ATTR    VALUE   NAME

SAMPM
        ADDR    00000H  MAIN
        ADDR    000D2H  START
SAMPS
        ADDR    000E9H  CONVAH
        NUM     FFE20H  _@STBEG
        NUM     FCF00H  _@STEND

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

Public symbol list

(4) -kl/-nkl

[Syntax]

```
-kl
-nkl
```

- Default assumption

  -nkl

[Function]

- The -kl option outputs a local symbol list into a link list file.
- The -nkl option makes the -kl option unavailable.

[Application]

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

[Explanation]

- If the -nkm, -nkp, and -nkl options are all specified, the link list file cannot be output.
- If -ng option is specified, the local symbol list cannot be output.
- If both options -kl and -nkl are specified at the same time, the option specified last takes precedence.
- If the -np option is specified, the -kl option is unavailable.

[Example of use]

- To output a local symbol list into a link list file (k0r.map), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -s -g -pk0r.map -kl
```

<Contents of k0r.map>

```
78K0R Linker Vx.xx                      Date:xx xxx xx  Page:1

Command : k0rmain.rel k0rsub.rel -s -g -pk0r.map -kl
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file :
Directive :

*** Link information ***

    6 output segment(s)
  9DH byte(s) real data
   40 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR
    :
------------------------------------------------------------------------

78K0R Linker Vx.xx                      Date:xx xxx xx  Page:2

*** Local symbol list ***

MODULE  ATTR    VALUE   NAME

SAMPM
        MOD             SAMPM
        DSEG            DATA
        ADDR    FFE20H  HDTSA
        ADDR    FFE21H  STASC
        CSEG            CODE
        CSEG            ?CSEG
SAMPS
        MOD             SAMPS
        CSEG            ?CSEG
        ADDR    0015CH  SASC
        ADDR    00162H  SASC1

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

Local symbol list

Link list format specification

(1) -ll

[Syntax]

```
-ll[number-of-lines]
```

- Default assumption

  -ll0 (No page breaks)

[Function]

- The -ll option changes the number of lines that can be printed in 1 page in a link list file.

[Application]

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

[Explanation]

- The range of number of lines that can be specified with the -ll option is shown below.

  0, 20 ≤ number of lines printed on 1 page ≤ 32,767

  An abort error occurs if a numerical value outside this range, or something other than a numerical value, is specified.
- If *number-of-lines* is omitted, 0 will be specified.
- If the number of lines specified is 0, no page breaks will be made.
- If the -np option is specified, the -ll option is unavailable.

[Example of use]

- To specify 20 as the number of lines per page in a link list file (k0r.map), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -s -pk0r.map -ll20
```

<Contents of k0r.map>

```
78K0R Linker Vx.xx                      Date:xx xxx xx  Page:1

Command : k0rmain.rel k0rsub.rel -s -pk0r.map -km -ll20
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file :
Directive :

*** Link information ***

    4 output segment(s)
  5FH byte(s) real data
   41 symbol(s) defined

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

78K0R Linker Vx.xx                      Date:xx xxx xx  Page:2

*** Memory map ***

 SPACE = REGULAR

 MEMORY = ROM
 BASE ADDRESS = 00000H  SIZE = 40000H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
        CODE                    00000H  00002H  CSEG    AT
                CODE    SAMPM   00000H  00002H
* gap *                          00002H  000BEH
        ?CSEGOB0                000C0H  00004H  CSEG    OPT_BYTE
        ?CSEG                   000C4H  00059H  CSEG
                ?CSEG   SAMPM   000C4H  00017H
--------------------------------------------------------------------------------

78K0R Linker Vx.xx                      Date:xx xxx xx  Page:3

                ?CSEG   SAMPS   000DBH  00042H
* gap *                          0011DH  3FEE3H

 MEMORY = RAM
 BASE ADDRESS = FCF00H  SIZE = 03100H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
* gap *                          FCF00H  02F20H
        DATA                    FFE20H  00003H  DSEG    AT
                DATA    SAMPM   FFE20H  00003H
* gap *                          FFE23H  000DDH
* gap (Not Free Area) *          FFF00H  00100H


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

(2) -lf/-nlf

[Syntax]

```
-lf
-nlf
```

- Default assumption

  -nlf

[Function]

- The -lf option inserts a form feed (FF) code at the end of a link list file.
- The -nlf option makes the the -lf option unavailable.

[Application]

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

[Explanation]

- If the -np option is specified, the -lf option is unavailable.
- If both options -lf and -nlf are specified at the same time, the option specified last takes precedence.

[Example of use]

- To add a form feed code at the end of a link list file (k0r.map), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -pk0r.map -lf
```

Error list file output specification

(1) -e/-ne

[Syntax]

```
-e[file-name]
-ne
```

- Default assumption

  -ne

[Function]

- Specify the -e option to specify the output destination and file name of an error list file.
- The -ne option makes the -e option unavailable.

[Application]

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

[Explanation]

- The file name of the error list file can be specified as a disk-type file name or as a device-type file name.
- When the -e option is specified and the output file name is omitted, the error list file name will be *input-file-name*.elk.
- When the -e option is specified and the drive name is omitted, the error list file will be output to the current drive.
- If both options -e and -ne are specified at the same time, the option specified last takes precedence.

[Example of use]

- To create an error list file (k0r.elk), describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -dk0r.dr -ek0r.elk
```

An error has occurred in the contents of the directive file.

<Contents ofk0r.elk>

```
k0r.dr ( 3 ) : RA78K0R error E3102 : Directive syntax error
```

Library file specification

(1) -b

[Syntax]

```
-bfile-name
```

- Default assumption

  None

[Function]

- The -b option 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 the -b option to input a library file to the linker.

[Explanation]

- Only a disk-type file name can be specified as the file name.
- *file-name* cannot be omitted.
- 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.
- If a file name which does not include a path name is specified, a library file will be input from a path specified by the -i option or from the default search path.
- If the -b option is specified two or more times, a library file will be input in a specified sequence. Up to 64 -b options may be specified.

  Caution When specifying two or more libraries in the [Linker Options] dialog box in PM+, delimit them with commas (,).
- For a detailed explanation of the method of creating library files, refer to "CHAPTER 7 LIBRARIAN".

[Example of use]

- To input a library file (k0r.lib), describe as:

```
C>lk78k0r k0rmain.rel -bk0r.lib
```

  k0rsub.rel is registered in the library file.

Library file read path specification

(1) -i

[Syntax]

```
-ipath-name[,path-name]... (two or more path names can be specified)
```

- Default assumption

  Path specified by environmental variable LIB78K0R

  Current path, if no path is specified.

[Function]

- The -i option specifies input of a library file from a specified path.

[Application]

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

[Explanation]

- The -i option is only available when a library file name is specified by the -b option without including a path name.
- Two or more specifications of -i are possible. Two or more paths can be specified by separating them with ",". A blank space cannot be inserted before or after the ",".
- 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.
- Even if no library file exists in the specified path, an error will not occur.
- If *path-name* is omitted, an abort error occurs.
- If a library file is specified by the -b option without including a path name, the linker will search paths in the following sequence.

  (i) Path specified by the -i option

  (ii) Path specified by environmental variable "LIB78K0R".

  (iii) The current path

  An abort error occurs if a library file with the specified name is not found in any of these paths.

[Example of use]

- To search and read a library file from folders C:\lib1 and C:\lib2 in that order, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -bk0r.lib -iC:\lib1,C:\lib2
```

- To search for a library file from path C:\Program Files\NEC Electronics Tools\library files, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -bk0r.lib -i”C:\Program Files\NEC
Electronics Tools\library files”
```

Parameter file specification

(1) -f

[Syntax]

```
-ffile-name
```

- Default assumption

  Options and input file names can only be input from the command line.

[Function]

- The -f option specifies input of linker options and the input file name from a specified file.

[Application]

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

[Explanation]

- Only a disk-type file name can be specified as file-name. If a device-type file name is specified, an abort error occurs.
- If *file-name* is omitted, an abort error occurs.
- Nesting of parameter files is not permitted. If the -f option is specified within a parameter file, an abort error occurs.
- The number of characters that can be written within a parameter file is unlimited.
- Separate options or file names with a blank space, a tab or a line feed code (LF).
- 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.
- The expanded options specified last will take precedence.
- The characters following ";" or "#" in a parameter file are all assumed to be comments, up to the line feed code (LF) or EOF.
- If the -f option is specified two or more times, an abort error occurs.

[Example of use]

- Perform link using a parameter file (k0r.plk).

  <Contents of the parameter file (k0r.plk)>

```
; parameter file
k0rmain.rel k0rsub.rel -ok0r.lmf -pk0r.map -e
-tC:\tmp -g
```

Enter the following on the command line.

```
C>lk78k0r -fk0r.plk
```

Specification of path for temporary file creation

(1) -t

[Syntax]

```
-tpath-name
```

- Default assumption

  Path specified by environmental variable TMP

  Current path, if no path is specified.

[Function]

- The -t option specifies a path in which a temporary file is created.

[Application]

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

[Explanation]

- Only a path can be specified as a path name.
- *path-name* cannot be omitted.
- Even if a previously created temporary file exists, if the file is not protected it will be overwritten.
- 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.
- Temporary files are deleted when assembly is finished. They are also deleted when assembly is aborted by pressing (CTRL+C).
- The path in which the temporary file is to be created is determined according to the following sequence.

  (i) The path specified by the -t option

  (ii) The path specified by environmental variable TMP (when the -t option is omitted)

  (iii) The current path (when TMP is not set)

  When (i) or (ii) is specified, if the temporary file cannot be created in the specified path an abort error occurs.

[Example of use]

- To output a temporary file to folder C:\tmp, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -tC:\tmp
```

- To output a temporary file to folder C:\Program Files\NEC Electronics Tools\temporary files, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -t”C:\Program Files\NEC Electronics
Tools\temporary files”
```

Device file search path specification

(1) -y

[Syntax]

```
-ypath-name
```

- Default assumption

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

  (i) Path registered in the device file installer
  (ii) Path by which LK78K0R was started up
  (iii) Current folder
  (iv) The environmental variable PATH

[Function]

- The -y option reads a device file from the specified path.

[Application]

- Specify a path where a device file exists.

[Explanation]

- If anything other than a path name is specified after the -y option, an abort error occurs.
- If the path name is omitted after the -y option, an abort error occurs.
- The path from which the device file is read in the order determined as follows.

  (i) Path specified by the -y option
  (ii) Path registered in the device file installer
  (iii) Path by which LK78K0R was started up
  (iv) Current folder
  (v) The environmental variable PATH

[Example of use]

- To specify the path for the device file as folder C:\78k0r\dev, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -yC:\78k0r\dev
```

- To specify the path for the device file as folder C:\Program Files\NEC Electronics Tools\device files, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -y"C:\Program Files\NEC Electronics
Tools\device files"
```

Warning message output specification

(1) -w

[Syntax]

```
-w[level]
```

- Default assumption

  -w1

[Function]

- The -w option 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]

- If anything other than a level is specified following the -w option, an abort error occurs.
- Only levels 0, 1 and 2 can be specified.
- 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 "11.3 Linker Error Messages".

[Example of use]

- To output a detailed warning message, describe as:

```
C>lk78k0r k0rmain.rel k0rsub.rel -w2
```

Link specification of boot area ROM program of flash memory model

(1) -zb

[Syntax]

```
-zb
```

- Default assumption

  No limitation for the location range

[Function]

- The -zb option specifies the first address of the flash memory area.

[Explanation]

- Specifies linking of the boot area ROM program of a flash memory model and the first address of the flash memory area.

  The specifiable value range is 0H to 0FFFFH.
- If no address is specified, an error occurs.
- No codes can be located at addresses higher than the specified address.

  Caution Do not specify this option for a device that does not have a flash memory area self programming function.

[Example of use]

- To specify 2000h as the start address of flash memory area, describe as:

```
C>lk78k0r k0rmain.asm -zb2000h
```

On-chip debug option byte specification

(1) -go

[Syntax]

```
-go control-value,start-address[,size]
```

- Default assumption

  On-chip debug is not used.

  Address C3H is the initial value specified the device file.

[Function]

- Specifies whether on-chip debug is used or not.

[Application]

- Use the -go option to specify the control value, the start address, and the program size for on-chip debug operation.

[Explanation]

- For the control value, specify the control value for on-chip debug operation.
  An abort error occurs if a value that cannot be specified for the control value is specified.
  For details on the control value, refer to the document supplied with the ID78K0R.
- For the start address, specify the location start address of the on-chip debug program.
  The range of values that can be specified for the start address is shown below.

  0 ≤ start address ≤ 0FFFFFH

  If the start address is omitted, D8H will be specified.
  For details on the start address, refer to the “QB-MINI2 On-Chip Debug Emulator with Programming Function“ (U18371EJ).
- For the size, specify the size of the on-chip debug program.
  The range of values that can be specified for the size is shown below.

  88 ≤ size ≤ 1,024

  If the size is omitted, 88 will be specified.
  For details on the program size, refer to the document supplied with the ID78K0R.
- An abort error occurs if anything other than a numeric value is specified for the control value, start address, or size.
- If the -go option is specified, the control value will be located at address C3H.
  No segments can be located at addresses 2H, 3H, and CEH to D7H, nor an area of the program size starting from an address specified with the -go option, because these areas will be filled with FFH.
  Addresses C0 to C2H are secured as the user option byte area by specifying the -gb option.

- If the -go option is not specified, no user codes can be located at addresses C3H because these addresses are reserved.
- The control value to be located at address C3H can also be specified by defining the segment with relocation attributes shown below, in the assembler source. Define the segment with 4 bytes in total, including the user option byte starting from address C0H.

```
[Any segment name]      CSEG    OPT_BYTE
                        DB      11H
                        DB      22H
                        DB      33H
                        DB      44H
```

If specification of the assembler source and specification of this option are made in duplicate, this option takes precedence.

[Example of use]

- Embed 0FFH at address C3H as a control value.
- Reserve the area starting from the start address (address 12345H) up to "256" bytes as the program area.

```
C>lk78k0r k0rmain.rel -go0FFH,12345H,256
```

Security ID specification

(1) -gi

[Syntax]

```
-gisecurity-ID
```

- Default assumption

  A security ID is not set.

[Function]

- Specifies a security ID.

[Application]

- Specify the -gi option to set a security ID.

[Explanation]

- Specify a hexadecimal value that ends with "H". If any other value is specified, an abort error occurs.
- Specify a security ID within 10 bytes. If the specified security ID falls short of 10 bytes, the higher bits are filled with 0.
- The security ID is set at addresses C4H to CDH. If a security ID is set, no segment can be located at addresses C4H to CDH.
- An abort error occurs if this option is specified for a device that does not have a security ID function.
- A security ID can also be specified by defining the following relocation attribute segment in the assembler source. However, be sure to specify SECUR_ID as the relocation attribute of the segment.

```
[Any segment name]  CSEG    SECUR_ID
                    DB      11H
                    DB      22H
                    DB      33H
                    DB      44H
                    DB      55H
                    DB      66H
                    DB      77H
                    DB      88H
                    DB      99H
                    DB      0AAH
```

If specification of the assembler source and specification of this option are made in duplicate, this option takes precedence.

[Caution]

- If this option is not specified for a device that has a security ID function, any code may be allocated.

[Example of use]

- To specify the same "112233445566778899AA" as the specification of the above assembler source, describe as:

```
C>lk78k0r k0rmain.rel -gi1122334455667788 99aah
```

User option byte specification

(1) -gb

[Syntax]

```
-gbuser-option-byte-value
```

- Default assumption

  Initial value set in the device file.

[Function]

- The -gb option specifies the value set for the user option byte.

[Application]

- Use the -gb option to specify the user option byte value.

[Explanation]

- The range of values that can be specified for the user option byte is shown below.

  0 ≤ user option byte value ≤ 0FFFFFFH

- An abort error occurs if a value that cannot be specified as the user option byte value is specified.
- Specify a hexadecimal value that ends with "H". If any other value is specified, an abort error occurs.
- The user option byte is specified at addresses C0H to C2H.
- If the -gb option is not specified, no user codes can be located at addresses C0 to C2H because these addresses are reserved.
- Specify the option byte value within 3 bytes. If the specified option byte value falls short of 3 bytes, the higher bits are filled with 0.
- The control value to be located at addresses C0H to C2H can also be specified by defining the segment with relocation attributes shown below, in the assembler source. Define the segment with 4 bytes in total, including the user option byte starting from address C3H.

```
[Any segment name]      CSEG    OPT_BYTE
                        DB      11H
                        DB      22H
                        DB      33H
                        DB      44H
```

If specification of the assembler source and specification of this option are made in duplicate, this option takes precedence.

[Example of use]

- To specify A1H at address C0H, B2H at address C1H, and C3H at address C2H as the user option byte value, describe as:

```
C>lk78k0r k0rmain.rel -gb0A1B2C3H
```

Mirror area specification

(1) -mi

[Syntax]

```
-mi0 or -mi1
```

- Default assumption

  -mi0

[Function]

- The -mi option specifies the location destinations of segments in the mirrored area.

[Application]

- Use the -mi option to specify the location destinations of segments in the mirrored area.

[Explanation]

- The location destinations of segments with relocation attribute CSEG MIRRORP are specified by the linker.
- If -mi0 is specified, the segment is located in the mirror area when MAA = 0. If -mi1 is specified, the segment is located in the mirror area when MAA = 1.
  For details on the mirror area, refer to the user's manual of the device.
- Public symbol "_@MAA" will be generated. This is a NUMBER attribute symbol that holds "0" when -mi0 is specified, and holds "1" when -mi1 is specified.

[Example of use]

- To locate the segment in the mirror area when MAA = 1, describe as:

```
C>lk78k0r k0rmain.rel -mi1
```

With a device in which a mirror area is secured at F1000H or higher address, segments with CSEG MIRRORP are located at address 11000H and higher.

<Example of use with startup routine provided by the compiler>

```
MOVW    PMC , #_@MAA
```

In this case, "1" is stored in PMC.

64 KB boundary location specification

(1) -ccza

[Syntax]

```
-ccza
-nccza
```

- Default assumption

  -ccza (when input files are assembler output files only)

  -nccza (when compiler output file is included in input files)

[Function]

- The -ccza option specifies whether to locate a segment to the last byte (xFFFFH[Note]) of each 64 KB boundary area.

  Note x: 0H to EH

[Application]

- Use the -ccza option to specify whether to locate a segment to the last byte of each 64 KB boundary area.

[Explanation]

- If development is performed only with the assembler, specification of this option is not necessary because a segment is automatically located to the last byte of each 64 KB boundary area.
- If an object module file output from the compiler is input to the linker, the linker automatically assumes that the -nccza option is specified, so no segment is located to the last byte of each 64 KB boundary area.
- If the -za option is specified in the compiler, relocation of a segment to the last byte of each 64 KB boundary area is enabled, so specify the -ccza option.
- For details on segment relocation to the last byte of each 64 KB boundary area, refer to CC78K0R C Compiler Language User's Manual.

Help specification

(1) --

[Syntax]

```
--
```

- Default assumption

  No display

[Function]

- The -- 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 the -- option is specified, all other options are unavailable.

  Caution This option cannot be specified on PM+.

  To reference PM+ help, click the [Help] button in the [Linker Options] dialog box.

[Example of use]

- To output a help message on the display, describe as:

```
C>lk78k0r --
```

```
78K0R Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

usage : lk78k0r [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[n]            : Change warning level (n = 0 to 2).
-zbaddress       : Create Boot file (address : flash start address).
-godata,address[,size] : Change On-Chip Debug Option Bytes, start address,
size (size = 88 to 1024).
-giid            : Set Security ID.
-gbdata          : Set User Option Bytes.
-mi[0 or 1]      : Select allocation for MIRRORP segment.
-ccza/-nccza     : Allocate user code to nFFFFH/Not.
--               : Show this message.

DEFAULT ASSIGNMENT : -o -p -ne -km -kd -nkp -nkl -ll0 -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 : ( 0FEF00H , 100H )
           MERGE CSEG1 : = ROM
           MERGE DSEG1 : AT ( 0FF000H )
```

5.7 Option Settings in PM+

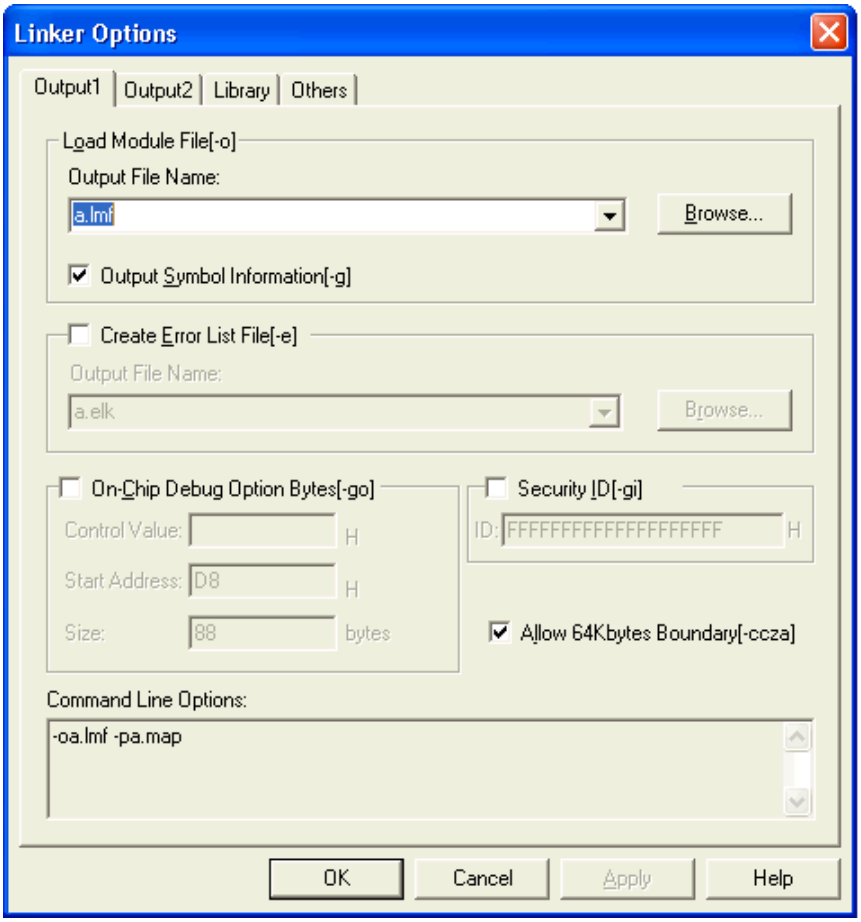
This section describes the method for setting linker options from PM+.

5.7.1 Option setting method

The [Linker Options] dialog box is opened if [Linker Options] is selected from the [Tools] menu of PM+ or if the [LK] button on the toolbar is clicked.

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

Figure 5-1 [Linker Options] Dialog Box

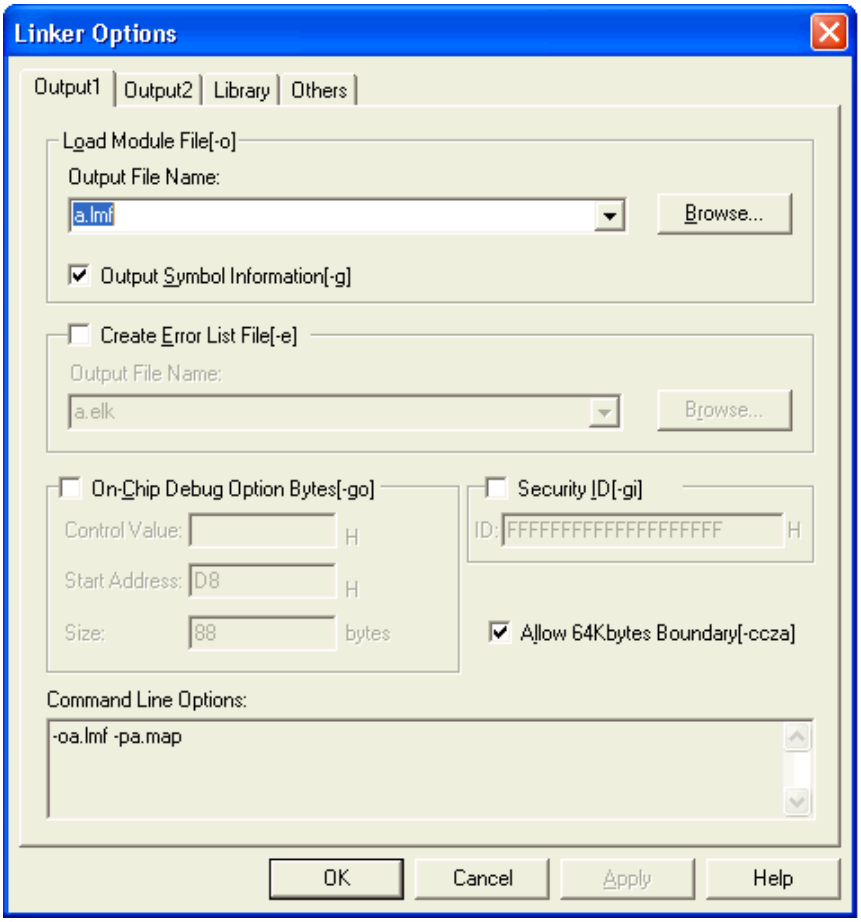


5.7.2 Explanation of dialog box

The various tabs in the [Linker Options] dialog box are described below.

(1) [Output1] tab

Figure 5-2 [Linker Options] Dialog Box (When [Output1] Tab Is Selected)



- Load Module File[-o]

Output File Name	Specify the path and file name of the load module file by using the [Browse...] button or directly inputting a path and file name. The number of characters that can be input is up to 259. a.lmf is specified by default.
Output Symbol Information[-g]	Select this option to add debug information (local symbol information) to the load module file (by default).

- Create Error List File[-e]

  Select this option to output an error list file.

Output File Name	Specify the path name and file name of the error list file by using the [Browse...] button or directly inputting a path and file name. The number of characters that can be input is up to 259.

- On-Chip Debug Option Bytes[-go]

  Select this option to use the on-chip debug option byte function.

Control Value	Specify the control value for on-chip debug operation. For details on the control value, refer to the document supplied with the ID78K0R.
Start Address	Specify the location start address of the on-chip debug program. The specifiable value range is as follows. 0 ≤ start address ≤ 0FFFFFH D8H is specified by default. If the start address is omitted, D8H will be specified.
Size	Specify the on-chip debug program size. The specifiable value range is as follows. 88 ≤ size ≤ 1,024 88 is specified by default. If the size is omitted, 88 bytes will be specified.

  Remark If the on-chip debug option byte setting is changed using Applilet, the changed contents are reflected in this dialog box.

  Caution This option cannot be specified for a device that does not have an on-chip debug function.

- Security ID[-gi]

  Select this option to set a security ID.

  Up to hexadecimal 20 characters can be input. FFFFFFFFFFFFFFFFFFFF is specified by default.

ID	Specify a security ID.

  Remark If the security ID setting is changed using Applilet, the changed contents are reflected in this dialog box.

  Caution This option cannot be specified for a device that does not have a security ID function.

- Allow 64Kbytes Boundary[-ccza]

  Select this option if segment relocation to the last byte of a 64 KB boundary area (xFFFFH[Note]) is specified (Default setting when all of the input files are the assembler outputs).

  Note x: 0H to EH

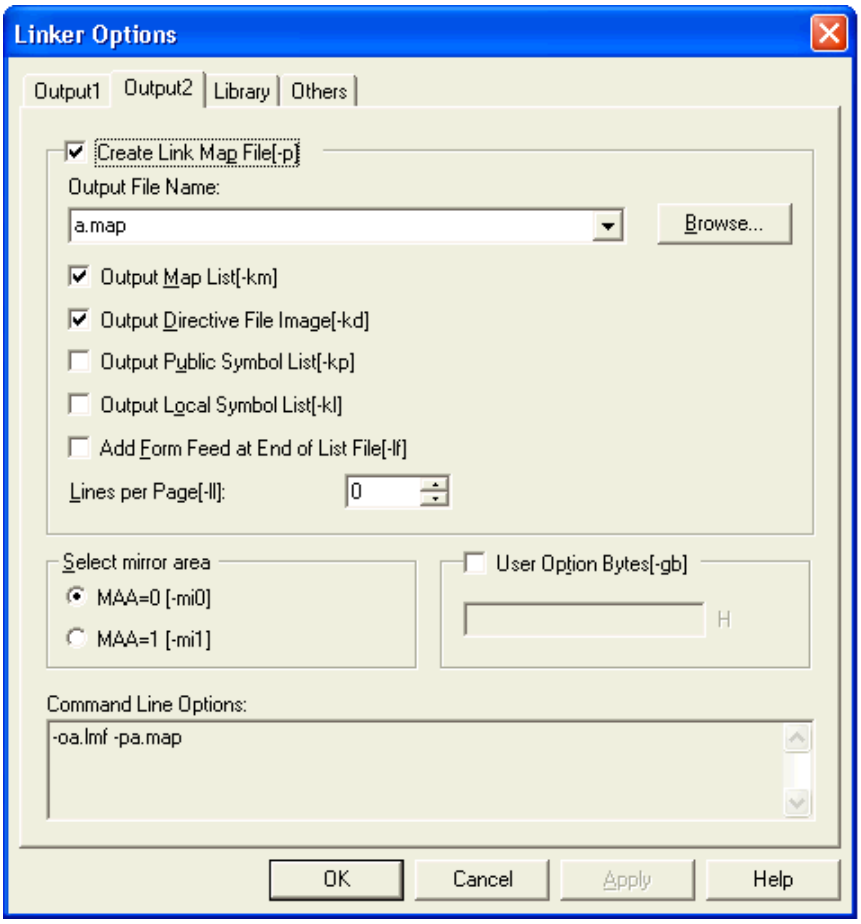
  Caution This option cannot be specified for a project to which C sources are registered if "Disable Extensions (ANSI Standard Only)[-za]" is not selected on the [Extend] tab in the [Compiler Options] dialog box.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

(2) [Output2] tab

Figure 5-3 [Linker Options] Dialog Box (When [Output2] Tab Is Selected)



- Create Link Map File[-p]

  Select this option to output a link list file.

Output File Name	Specify the path and file name of the link list file by using the [Browse...] button or directly inputting a path and file name. The number of characters that can be input is up to 259. a.map is specified by default.
Output Map List[-km]	Select this option to output a map file in the link list file (by default).
Output Directive File Image[-kd]	Select this option to output a link directive file in the link list file (by default).
Output Public Symbol List[-kp]	Select this option to output a public symbol list in the link list file.
Output Local Symbol List[-kl]	Select this option to output a local symbol list in the link list file.
Add Form Feed at End of List File[-lf]	Select this option to add a form feed code (FF) to the link list file.

Lines per Page[-ll]	Specify the number of lines on one page of the link list file. The specifiable number of lines is 0 and from 20 to 32,767. 0 is specified by default.

- Select mirror area

  Select the area in which segments that mirrored to the RAM space are to be located.

  MAA=0 [-mi0]: Locate the segment in the area to be mirrored when MAA = 0.

  MAA=1 [-mi1]: Locate the segment in the area to be mirrored when MAA = 1.

  “MAA=0 [-mi0]“ is specified by default.

  For details on the mirror area, refer to the user's manual of the device.

- User Option Bytes[-gb]

  Select this option to specify the value to set a user option byte value.

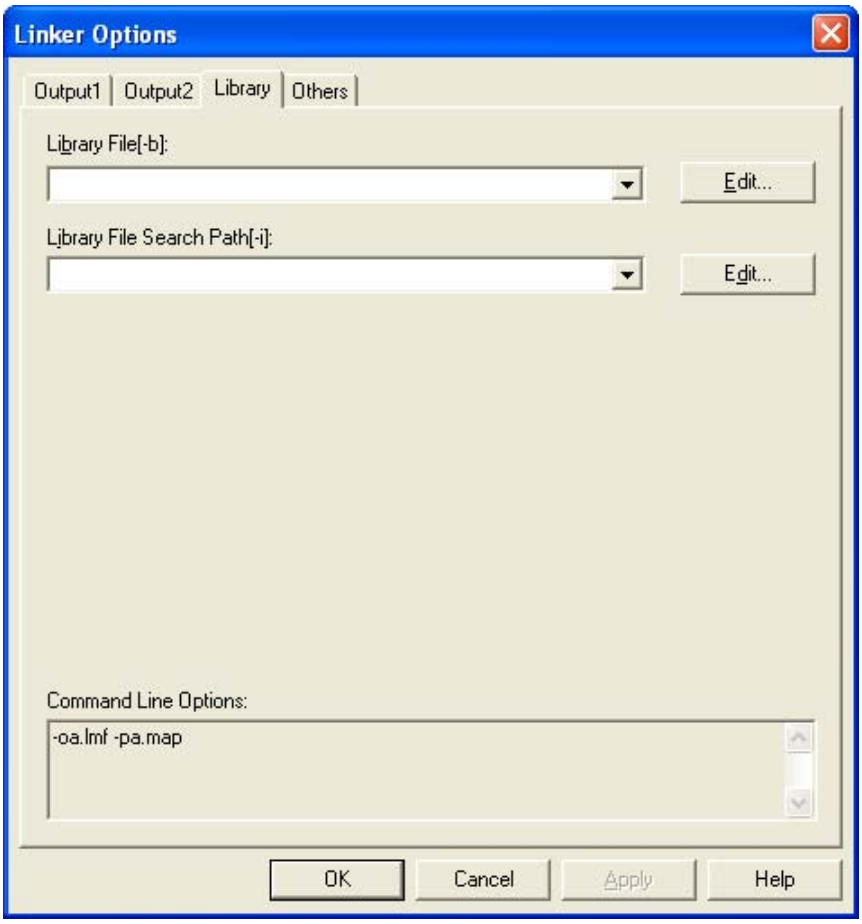
  The specifiable value range is 0H to 0FFFFFFH.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

(3) [Library] tab

Figure 5-4 [Linker Options] Dialog Box (When [Library] Tab Is Selected)



- Library File[-b]

  Specify a file to be input as a library file by using the [Edit...] button (opens the [Edit Option] dialog box) or directly inputting a path and file names.

  Up to 64 items can be specified by using commas as delimiters.

  Up to 259 characters can be input to each file.

  Remark When the library file name required for a project has been output to the project file by the CC78K0R, the library file is displayed in the Command Line Options area. In addition, if "Link STDIO Library for SM+[S]" is selected in the [SM+ Options] dialog box of SM+ for 78K0R, the standard I/O library dedicated to SM+ is displayed in the Command Line Options area. In the same manner, if a library has been specified with the RX78K0R, the library specified by the RX78K0R is displayed in the Command Line Options area.

  A value from which the number of libraries required by the CC78K0R, SM+ for 78K0R and RX78K0R is subtracted can be specified as the number of files.

- Library File Search Path[-i]

  Specify the path via which the library file is to be read, by using the [Edit...] button (opens the [Edit Option] dialog box) or directly inputting a path and file name.

  Up to 64 items can be specified by using commas as delimiters.

  Remark For a project to which C sources are registered, the install path of the standard library is displayed in the Command Line Options area.

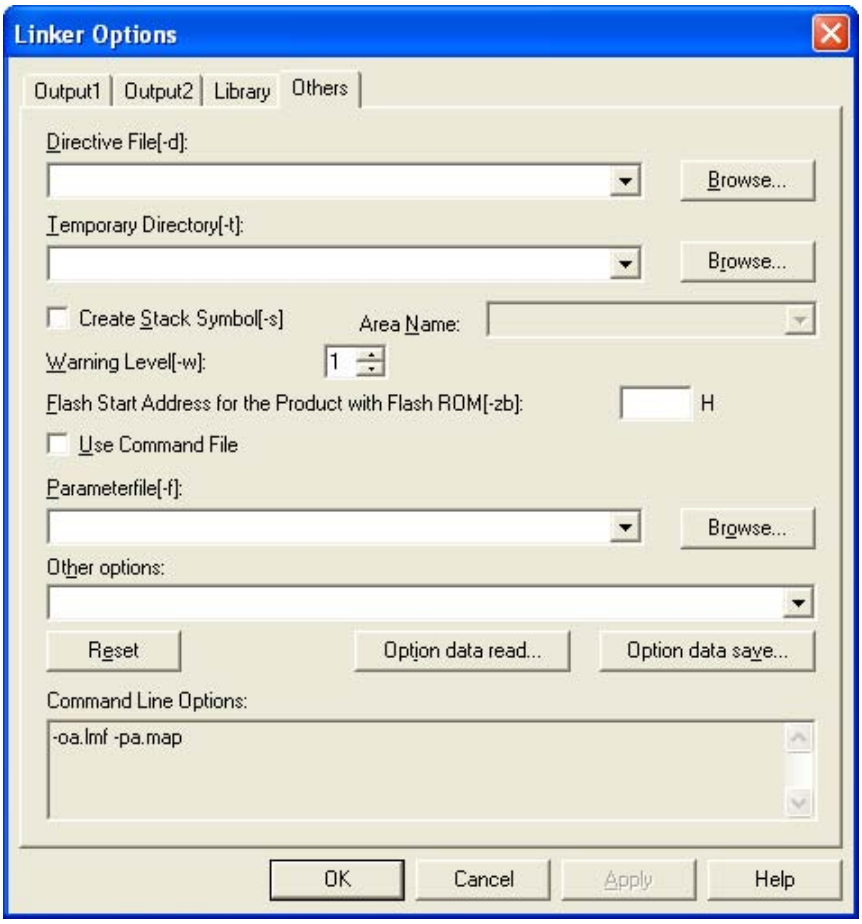
  A value from which the number of standard library paths is subtracted can be specified as the install path.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

(4) [Others] tab

Figure 5-5 [Linker Options] Dialog Box (When [Others] Tab Is Selected)



- Directive File[-d]

  Specify the file to be input as a directive file by using the [Browse...] button or directly inputting a path and file name.

  The number of characters that can be input is up to 259.

  Remark If the directive file is changed using Applilet, the changed contents are reflected in this dialog box.

- Temporary Directory[-t]

  Specify the file where a temporary file is to be created by using the [Browse...] button or directly inputting a path and file name.

  The number of characters that can be input is up to 259.

- Create Stack Symbol[-s]

  Select this option to allocate the maximum vacant area of the memory area as a stack area.

Area Name	Specify a memory area name defined by the user or the memory area name defined by default. The number of characters that can be input is up to 256.

- 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.

  1 is specified by default.

- Flash Start Address for the Product with Flash ROM[-zb]

  Specify the boot area start address for products with flash memory.

  The specifiable value range is as follows.

  0H ≤ zb ≤ 0FFFFH

  Caution Do not specify this option for a device that does not have a flash memory area self programming function.

- Use Command File

  Select this option to create a command file.

- Parameterfile[-f]

  Specify the file to be input as a user-defined parameter file by using the [Browse...] button or directly inputting a file name.

  The number of characters that can be input is up to 259.

- Other options

  To specify an option other than those that can be set in this dialog box, enter the option in the input box.

  The number of characters that can be input is up to 259.

  Caution The help specification option (--) cannot be specified on PM+.

- [Reset] button

  Resets the input contents.

- [Option data read...] button

  Opens [Read Option Data] dialog box opens and the option data file is specified, that file is read.

- [Option data save...] button

  After the [Save Option Data] dialog box opens, save the option data file to the option data file with a name.

- Command Line Options

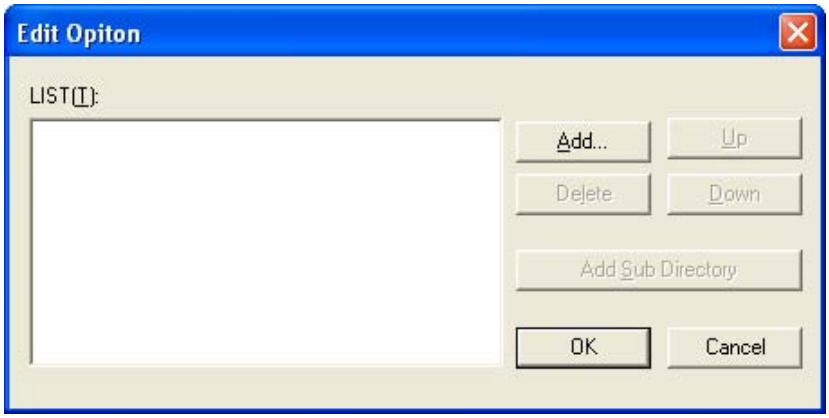
  This edit box is read-only. The currently set option character string is displayed.

5.7.3 [Edit Option] dialog box

Items are edited in list format in the [Edit Option] dialog box.

The [Edit Option] dialog box is described below.

Figure 5-6 [Edit Option] Dialog Box



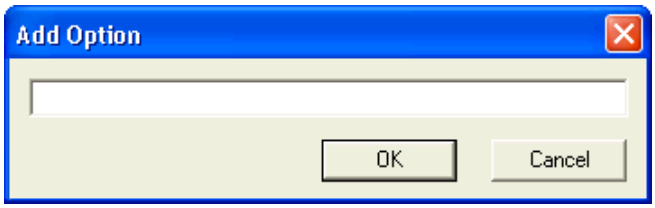
- [Add...] button

  Adds a list item.

  If the item to be added is a file or folder, the corresponding [Browse for Folder] dialog box opens.

  In all other cases, the [Add Option] dialog box opens. Specify details of the item to be added in this box.

Figure 5-7 [Add Option] Dialog Box



- [Delete] button

  Deletes the selected list item.

- [Up] button

  Moves the selected list item up.

- [Down] button

  Moves the selected list item down.

- [Add Sub Directory] button

  A subfolder can be added to the selected list item when the item is specified as “Library File Search Path [-i]” on the [Library] tab.

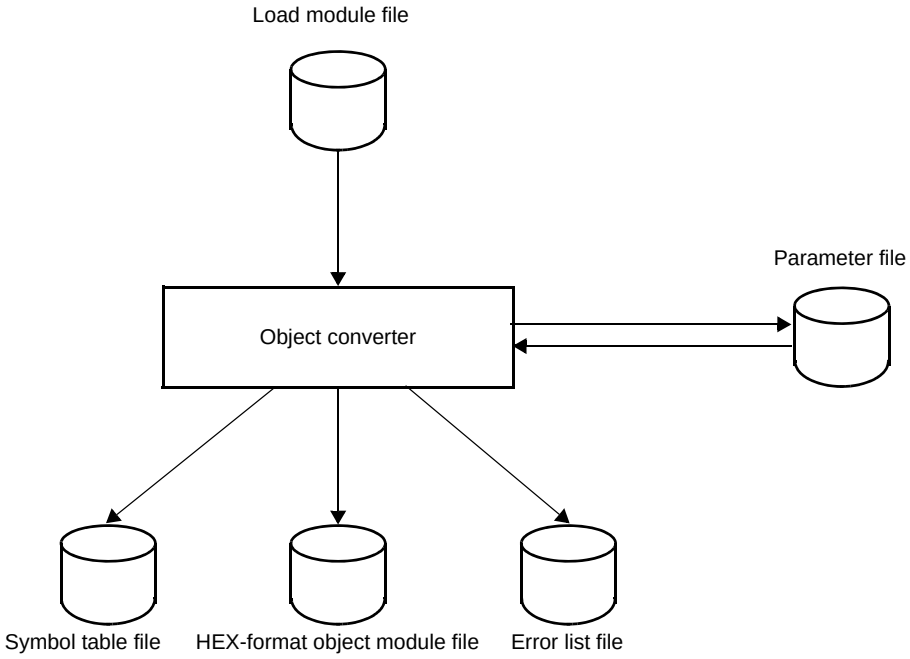
CHAPTER 6 OBJECT CONVERTER

The object converter inputs the load module file output by the RA78K0R 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.

Figure 6-1 I/O Files of Object Converter



6.1 I/O Files of Object Converter

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

Table 6-1 I/O Files of Object Converter

Type	File Name	Explanation	Default File Type
Input files	Load module files	- Binary image files of the object codes output as a result of linking - Files output by the linker	.lmf
	Parameter files	- Files containing the parameters for the executed programs (user-created files)	.poc
Output files	HEX format object module files	- 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	- Files containing the symbol data included in each module of an input file.	.sym
	Error list files	- Files containing error data generated during converting objects	.eoc

6.2 Functions of Object Converter

6.2.1 Flash memory self-rewriting mode support

The object converter can create separate HEX object module files in the boot area and flash area for the code located in the flash memory when the self-rewriting mode of the flash memory is used. To output separate HEX files, specify the object converter option (-zf). The file type is as follows:

Table 6-2 File Type When -zf Option Is Specified

File	File Type
Output file at boot area ROM program side	.hxb
Output file at program side other than boot area ROM	.hxf

6.2.2 HEX-format object module files

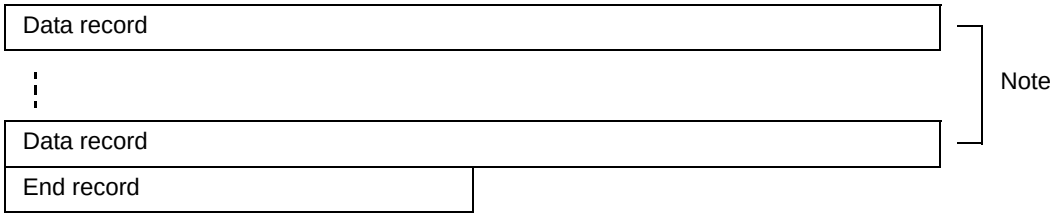
The HEX-format object module file output by the object converter can be input to a HEX loader such as a PROM programmer or a debugger.

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

```
: 0200000080007E
: 1000800011201A1620FE9A93001421FE63958462B3
: 1000900095FAFE617131809AA40073617131809A82
: 0D00A000A40072AF4D8D020D070D30AFA8
: 00000001FF
```

[Intel standard HEX-format object module file format]

Figure 6-2 Intel Standard Format



Note The data record is repeated here.

(1) Data record

:	02	0000	00	8000	7E
(i)	(ii)	(iii)	(iv)	(v)	(vi)

(i) Record mark

Indicates beginning of record.

(ii) Code number (2 digits)

Number of bytes in the code stored in the record. A maximum of 16 bytes can be stored.

(iii) Location address (offset)

The start address (offset) of the code displayed in the record is shown as a 4-digit hexadecimal.

(iv) Record type (2 digits)

Fixed at 00.

(v) Code (Max. 32 digits)

The object code is shown one byte at a time, with the upper 4 bits and lower 4 bits separated. A maximum of 16 bytes can be expressed in the code.

(vi) 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.

(2) End record

:	00	0000	01	FF
(i)	(ii)	(iii)	(iv)	(v)

(i) Record mark

Indicates beginning of record.

(ii) Code number

Fixed at 00.

(iii) Fixed at 0000

(iv) Record type

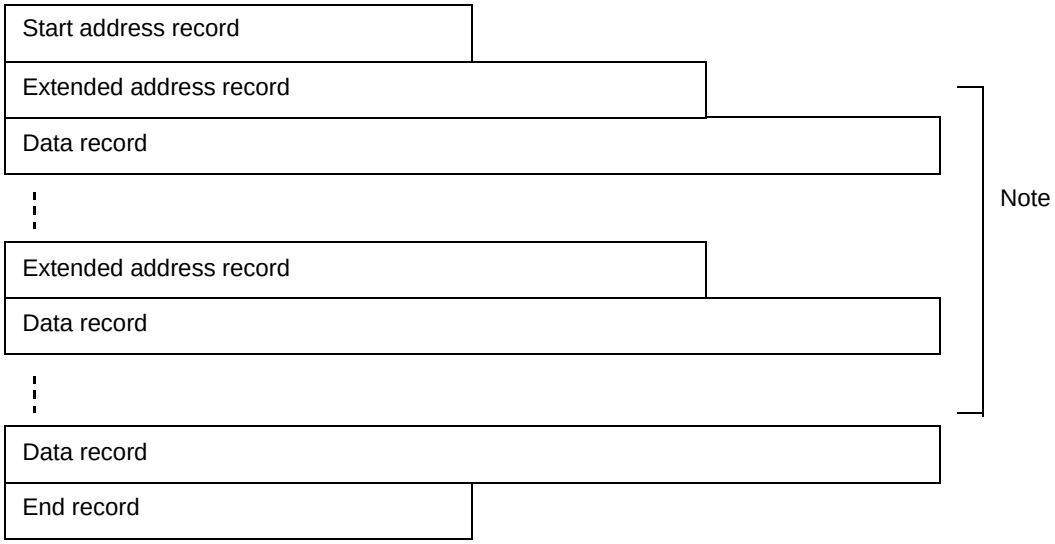
Fixed at 01.

(v) Check sum

Fixed at FF.

[Intel extended HEX-format object module file format]

Figure 6-3 Intel Extended Format



Note The extended address record and data record are repeated here.

(1) Extended address record

:	02	0000	02	XXXX	SS
(i)	(ii)	(iii)	(iv)	(v)	(vi)

(i) Record mark

Indicates beginning of record.

(ii) Code number

Fixed to 02.

(iii) 0Fixed to 0000

(iv) Record type

Fixed to 02.

(v) The paragraph value of the segment

The paragraph value of the segment is shown as a 4-digit hexadecimal.

(vi) Check sum (2 digits)

A value is input subtracting in order from 0 which counts down the data from the code number to the higher 8-bit value of the address.

(2) Data record

```
:       XX      XXXX    00      DD ... DD       SS
(i)     (ii)    (iii)   (iv)        (v)         (vi)
```

(i) Record mark

Indicates beginning of record.

(ii) Code number (2 digits)

Number of bytes in the code stored in the record. A maximum of 16 bytes can be stored.

(iii) Location address (offset)

The start address (offset) of the code displayed in the record is shown as a 4-digit hexadecimal.

(iv) Record type (2 digits)

Fixed to 00H.

(v) 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.

(vi) Check sum (2 digits)

A value is input subtracting in order from 0 which counts down the data from the code number to the code.

(3) Start address code

```
:       04      0000    03      0000    0000    F9
(i)     (ii)    (iii)   (iv)    (v)     (vi)    (vii)
```

(i) Record mark

Indicates beginning of record.

(ii) Fixed to 04

(iii) Fixed to 0000

(iv) Fixed to 03

(v) Fixed to 0000

(vi) Fixed to 0000

(vii) Fixed to F9

(4) End record

```
:       00      0000    01      FF
(i)     (ii)    (iii)   (iv)    (v)
```

(i) Record mark

(ii) Fixed to 00

(iii) Fixed to 0000

(iv) Fixed to 01

(v) Fixed to FF

[Extended tech HEX-format object module file format]

HEX files are composed of the following 3 types of block.

- Data block
- Symbol block (This is an unused block. Symbol data uses the symbol table file.)
- Termination block

Each block starts with a header field composed of a common 6 characters, and ends with the string end-ofline. Maximum length of each block is 255, not including the start character % and end-of-line.

The format for the common header field is shown below.

Table 6-3 Extended Tech Header Field

Item	No. of ASCII Characters	Explanation
%	1	The percent symbol specifies that the block is in extended tech format.
Block length	2	This is a 2-digit hexadecimal which indicates the number of characters in the block. This number of characters does not include the start character % and end-of-line.
Block type	1	6 = Data block 3 = Symbol block 8 = Termination block
Check sum	2	This is a 2-digit hexadecimal which indicates the remainder produced when the total value of the characters in the block (except the start character %, the check sum, and end-of-line) is divided by 256. The total value of the characters is shown in Table 6-4.

Table 6-4 Character Values for Check Sum Evaluation

Character	Value (Decimal)
0 to 9	0 to 9
A to Z	10 to 35
$	36
%	37
. (period)	38
_ (underscore)	39
a-z	40 to 65

(1) Data block

The format for the data block is shown below.

Table 6-5 Data Block Format for Extended Tech

Field	No. of ASCII Characters	Explanation
Header	6	Standard header field Block type = 6
Load address	2 to 17	Address from which the object code is loaded. Number of characters is variable.
Object code	2n	Number of bytes n, displayed as a 2-digit hexadecimal

Caution In extended Tech, the number of characters in a specific field is variable within 2 to 17 (1 to 16 characters of actual data). The first character in this variable field is a hexadecimal which indicates the length of the field. The numerical zero indicates that a character line consists of 16 characters. The length of the character string is therefore 1 to 16 characters, and the length of the variable-length field including the character string length indicator is 2 to 17.

```
%      15     6      1C     3      100    020202020202
(i)    (ii)   (iii)  (iv)   (v)    (vi)      (vii)
```

(i) Header character

(ii) Block length

15H = 21

(iii) Block type

6

(iv) Check sum

1CH

(v) Number of digits in load address

(vi) Load address

100H

(vii) Object code

6 bytes

(2) Termination block

The format for the termination block is shown below.

Table 6-6 Termination Block Format for Extended Tech

Field	No. of ASCII Characters	Explanation
Header	6	Standard header field Block type = 8
Load address	2 to 17	Start address for program execution. Number of characters is variable.

```
%       08      8       1A      2       80
(i)    (ii)   (iii)    (iv)    (v)     (vi)
```

(i) Header character

(ii) Block length

8H

(iii) Block type

8

(iv) Check sum

1AH

(v) Number of digits in load address

(vi) Load address

80H

(3) Symbol block (unused)

The extended Tech symbol block is data used for symbolic debugging. It may be assumed to have the following characteristics.

Table 6-7 Symbol Block Format for Extended Tech

Items	Characteristics
Symbol	1 to 16 uppercase and lowercase alphabets, numerals, period and underscore. Numerals are not permitted for the start character.
Value	Up to 64 bits (16 hexadecimal digits) possible.
Type	Address or scalar (a scalar indicates any numerical value other than an address). Addresses are divided into code addresses (instruction addresses) and data addresses (addresses of data items).
Global/local specification	Indicates whether a symbol is global (external reference enabled) or local.
Section membership	A section may be considered a range to which a memory name is given. Each address in a program belongs to at least 1 section. A scalar does not belong to any section.

The format for the symbol block is shown below.

Table 6-8 Symbol Block Format for Extended Tech

Field	No. of ASCII Characters	Explanation
Header	6	Standard header field Block type = 3
Section name	2 to 17	Section name 2 to 17 Name of the section which includes the symbols defined in the block. Number of characters is variable.
Section definition	5 to 35	Each symbol block must have 1 of this type of field. This field may be placed before or after any number of symbol definition fields. This format is shown in Table 6-9.
Symbol definition	5 to 35 each	This is a symbol definition field greater than 0 as shown in Table 6-10.

The symbols contained in a program are transferred as a symbol block. Each symbol block includes a section name and a list of the symbols that belong to that section. If necessary, a scalar can also be included in any section.

A symbol in the same section can be placed in one or more blocks.

The formats for the section definition field and the symbol definition field in the symbol block are shown below.

Table 6-9 Symbol Block Section Definition Fields for Extended

Field	No. of ASCII Characters	Explanation
0	1	0 specifies that the field is a section definition field.
Base address	2 to 17	This is a section start address. Number of characters is variable.
Length	2 to 17	Indicates the section length. Number of characters is variable and is calculated by the following 1 - (higher address - base address)

Table 6-10 Symbol Block Symbol Definition Fields for Extended Tech

Field	No. of ASCII Characters	Explanation
Type	1	1-digit hexadecimal indicating global/local symbol specification and type of value displayed. 1 = Global address 2 = Global scalar 3 = Global code address 4 = Global data address 5 = Local address 6 = Local scalar 7 = Local code address 8 = Local data address
Symbol	2 to 17	Indicates the symbol length. Variable.
Numerical value	2 to 17	Value corresponding to a symbol. Number of characters is variable.

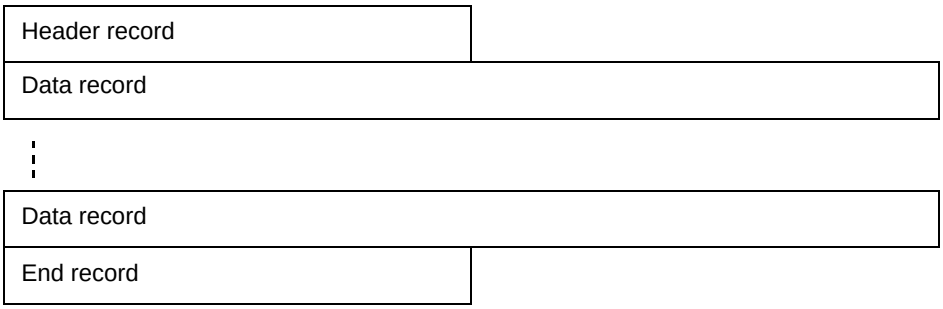
[Motorola S-type format]

Motorola S-type format files are converted from 5 records into 3 types. The composition of the entire file is shown in Figure 6-4. Types of records are shown below.

Table 6-11 Motorola HEX File Record Types

Item	Record Type
Header record (optional)	S0
Data record	S2 (Standard 24 bits) S3 (32 bits)
End record	S8 (Standard 24 bits) S7 (32 bits)

Figure 6-4 Motorola S-Type Format



Motorola HEX format files are divided into standard 24-bit addresses and 32-bit addresses. Standard addresses are composed of records S0, S2, and S8. The 32-bit addresses are composed of records S0, S3 and S7. Header record S0 is optional and is not output. A CR character is placed at the end of each S record.

The general formats and their meanings for each field in each record are shown below.

Table 6-12 General Format for Each Record

Record Type	General Format
S0	S0XXYY ... YYZZZZ
S2	S2XXWWWWWWDD ... DDZZ
S3	S3XXWWWWWWWWDD ... DDZZ
S7	S7XXWWWWWWWWZZ
S8	S8XXWWWWWWZZ

Table 6-13 Meanings of Fields

Field	Meaning
Sn	Record type
XX	Length of data record Number of bytes in the address, hexadecimal data and check sum
YY ... YY	File name ASCII code for the input file name expressed as a hexadecimal
WWWWWW [WW]	24th [32th] bit address
DD ... DD	Hexadecimal data 1 byte of data is expressed as a 2-digit hexadecimal.
ZZ	Check sum The lower 1 byte of complement 1 for the sum for each byte of the record length, address and the hexadecimal data, expressed as a 2-digit hexadecimal

```
S2       08       00FF11   D4520A20          A0

(i)     (ii)      (iii)      (iv)           (v)
```

(i) Record type

S2

(ii) Record length

8 bytes

(iii) Load addresses (24-bit address)

(iv) Hexadecimal data

(v) Check sum

(1) S0 record

```
S0       XX       YYYYYYYY          ZZ

(i)     (ii)       (iii)           (iv)
```

(i) Record type

(ii) Record length

This is the number of bytes in (iii) plus the number of bytes in (iv).

(iii) File name

(iv) Check sum

(2) S2 record

```
S2      XX      WWWWWW  DD ... DD       ZZ
(i)    (ii)      (iii)     (iv)        (v)
```

(i) Record type

(ii) Record length

This is the number of bytes in (iii) plus the number of bytes in (iv) plus the number of bytes in (v).

(iii) Load address

This is the 24-bit load address of the data in (iv) within the range 0H to FFFFFFH.

(iv) Data

This is the loaded data itself.

(v) Check sum

(3) S3 record

```
S3      XX      WWWWWWWW        DD ... DD       ZZ
(i)    (ii)      (iii)             (iv)        (v)
```

(i) Record type

(ii) Record length

This is the number of bytes in (iii) plus the number of bytes in (iv) plus the number of bytes in (v).

(iii) Load address

This is the 24-bit load address of the data in (iv) within the range 0H to FFFFFFFH.

(iv) Data

This is the loaded data itself.

(v) Check sum

(4) S7 record

```
S7      XX      WWWWWWWW        ZZ
(i)    (ii)      (iii)         (iv)
```

(i) Record type

(ii) Record length

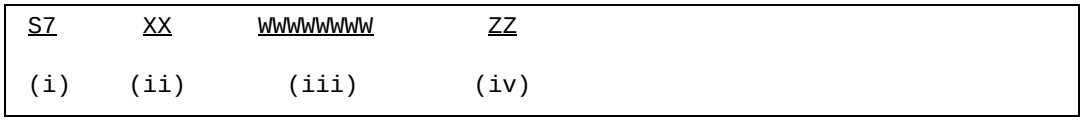
This is the number of bytes in (iii) plus the number of bytes in (iv).

(iii) Entry address

This is the 32-bit entry address within the range 0H to FFFFFFFH.

(iv) Check sum

(5) S8 record



(i) Record type

(ii) Record length

This is the number of bytes in (iii) plus the number of bytes in (iv).

(iii) Entry address

This is the 24-bit entry address within the range 0H to FFFFFFH.

(iv) Check sum

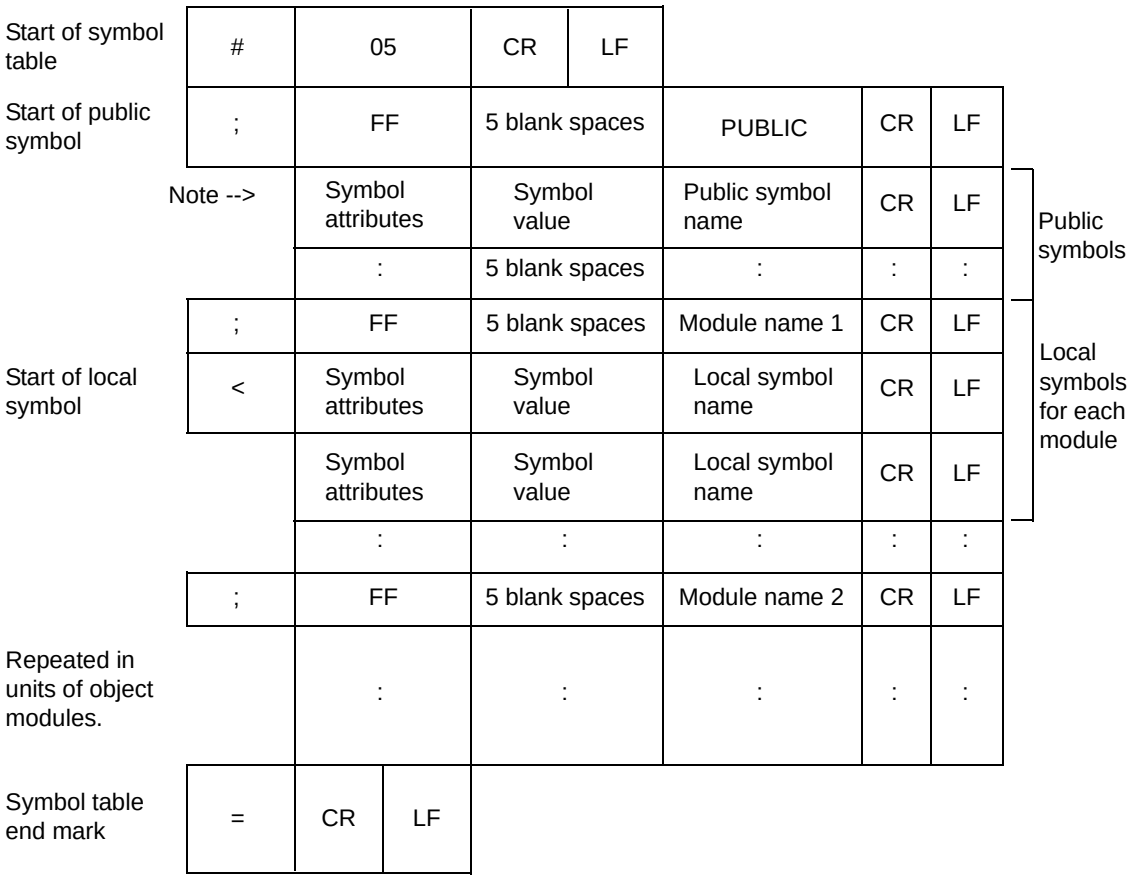
6.2.3 Symbol table file

The symbol table file output by the object converter is input to a debugger.

The following is the symbol table file of the sample program.

```
#05
; FF    PUBLIC
01000E9CONVAH
0100000MAIN
01000D2START
00FFE20_@STBEG
00FCF00_@STEND
; FF    SAMPM
<02FFE20HDTSA
02FFE21STASC
; FF    SAMPS
<010015CSASC
0100162SASC1
=
```

Figure 6-5 Symbol Table File Formats



Note Symbol attributes are the values shown below.

For symbol values, refer to Figure 6-6.

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 6-6 Symbol Value Formats

[When the symbol attribute is NUMBER]

Constant value 4 digits

[When the symbol attribute is LABEL]

Address value 4 digits

[When the symbol attribute is a bit symbol]

Upper 13 bits	Lower 3 bits

Upper 13 bits: The relative address from 0FE00H

Lower 3 bits: Bit position (0 to 7)

6.3 Object Converter Startup

6.3.1 Methods to start object converter

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

(1) Startup from the command line

```
 X>[path-name]oc78k0r[Δoption]...load-module-file-name[Δoption]...[Δ]
  |       |       |       |                    |             |
 (a)     (b)     (c)     (d)                  (e)           (d)
```

(a) Current drive name

(b) Current folder name

(c) Object converter command file name

(d) This contains detailed directions for the action of the object converter.

Enclose a path that includes a space in a pair of double quotation marks (" ").

(e) File name of the load module to be converted.

If more than one object converter option is specified, separate the options with a space. Uppercase characters and lowercase characters are not distinguished for the object converter options. For a detailed explanation of object converter options, refer to "6.4 Object Converter Options".

Specify the file name of a path that includes a space by enclosing it in a pair of double quotation marks (" ").

[Example]

- To output a HEX-format object module file (sample.hex), describe as:

```
C>oc78k0r k0r.lmf -osample.hex
```

(2) Startup from a parameter file

Use the parameter file when the data required to start up the object converter will not fit on the command line, or when the same object converter option is specified repeatedly each time object conversion is performed. To start up the object converter from a parameter file, specify the specify parameter file option (-f) on the command line.

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

```
 X>oc78k0r[Δload-module-file]Δ-fparameter-file-name
  |   |                          |           |
(a) (b)                        (c)         (d)
```

(a) Current drive name

(b) Current folder name

(c) Specify parameter file option

(d) A file which includes the data required to start up the object converter

Remark 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 load module file name is omitted from the command line, only one load module file name can be specified in the parameter file.
- The load module file name can also be specified after the option.
- Write in the parameter file all object converter options and output file names that should be specified in the command line. For a detailed explanation of parameter files, refer to "3.4 Using Parameter File".

[Example]

(1) Create the parameter file (k0r.poc) using an editor.

```
; parameter file
k0r.lmf -osample.hex
-ssample.sym -r
```

(2) Use parameter file k0r.poc to start up the object converter.

```
C>oc78k0r -fk0r.poc
```

6.3.2 Execution start and end messages

(1) Execution start message

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

```
78K0R Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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 : uPD78xxx
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 : uPD78xxx
Device file : Vx.xx

Object Conversion Complete, 3 error(s) and 0 warning(s) found.
```

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

[Example]

<A non-existent load module file name is specified.>

```
C>oc78k0r sample.lmf
```

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

RA78K0R error F4006 : File not found 'sample.lmf'
Program aborted.
```

In the above example, a non-existent load module file is specified. An error occurs and the object converter aborts the object conversion.

<A non-existent object converter option is specified.>

```
C>oc78k0r k0r.lmf -a
```

```
78K0R Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F4018 : Option is not recognized '-a'
Please enter 'OC78K0R --', if you want help messages.
Program aborted.
```

In the above example, a nonexistent object converter option is specified. An error occurs 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 11 ERROR MESSAGES" and take action accordingly.

6.4 Object Converter Options

6.4.1 Types of object converter options

The object converter options are detailed instructions for the operation of the object converter.

The types and explanations for object converter options are shown below.

Table 6-14 Object Converter Options

Classification	Option	Explanation
HEX format object module file output specification	-o	Specifies the output of a HEX-format object module file.
	-no	
Symbol table file output specification	-s	Specifies output of a symbol table file.
	-ns	
Specification of sort by object address order	-r	Sorts HEX-format objects in the order of their addresses.
	-nr	
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.
	-nu	
Error list file output specification	-e	Outputs an error list file.
	-ne	
Parameter file specification	-f	Inputs an input file name and options from a specified file.
HEX format specification	-ki	Intel standard HEX format
	-kie	Intel extended HEX format
	-kt	Extended Tech format
	-km	Motorola S-type format (standard address)
	-kme	Motorola S-type format (32-bit address)
Device file search path specification	-y	Reads a device file from a specified path.
File separate output specification for flash memory model	-zf	Outputs the boot area and other areas to separate HEX-format files.
Help specification	--	Displays a help message on the display.

HEX format object module file output specification

(1) -o/-no

[Syntax]

```
-o[output-file-name]
-no
```

- Default assumption

  -o*input-file-name*.HEX

[Function]

- The -o option specifies the output of a HEX-format object module file. The -o option also specifies the output destination and output file name.
- The -no option specifies that no HEX-format object module file is output.

[Application]

- Specify the -o option to change the output destination and output file name of the HEX-format object module file.
- Specify the -no option when performing an object conversion only to output a symbol table file. This will shorten object conversion time.

[Explanation]

- Specify a disk type file name for *output-file-name*.
  If a device-type file name is specified, an abort error occurs.
- If *output-file-name* is omitted when the -o option is specified, the HEX-format object module file *input-file-name*.hex will be output to the current folder.
- If only the path name is specified in *output-file-name*, *input-file-name*.hex will be output to the specified path.
- If both options -o and -no are specified at the same time, the option specified last takes precedence.
- When the -zf option is specified, the file type is as follows:

File	File Type
Output file at boot area ROM program side	.hxb
Output file at program side other than boot area ROM	.hxf

[Example of use]

- To output a HEX-format object module file (sample.hex), describe as:

```
C>oc78k0r k0r.lmf -osample.hex
```

Symbol table file output specification

(1) -s/-ns

[Syntax]

```
-s[output-file-name]
-ns
```

- Default assumption

  -s*input-file-name*.sym

[Function]

- The -s option specifies the output of a symbol table file. The -s option also specifies the output destination and output file name.
- The -ns option specifies that no symbol table file is output.

[Application]

- Specify the -s option to change the output destination and output file name of the symbol table file.
- Specify the -ns option when performing an object conversion only to output a HEX-format object module file. This will shorten object conversion time.

[Explanation]

- Specify a disk type file name for *output-file-name*.
  If a device-type file name is specified, an abort error occurs.
- If *output-file-name* is omitted when the -s option is specified, the symbol table file *input-file-name*.sym will be output to the current folder.
- If only the path name is specified in *output-file-name*, *input-file-name*.sym will be output to the specified path.
- If both options -s and -ns are specified at the same time, the option specified last takes precedence.

[Example of use]

- To output a symbol table file (sample.sym), describe as:

```
C>oc78k0r k0r.lmf -ssample.sym
```

Specification of sort by object address order

(1) -r/-nr

[Syntax]

```
-r
-nr
```

- Default assumption

  -r

[Function]

- The -r option outputs sorting of HEX-format objects in order of address.
- The -nr option outputs HEX-format objects in the order in which they were stored in the load module file.

[Application]

- Specify the -nr option if the HEX-format objects do not have to be sorted in address order.

[Explanation]

- If both options -r and -nr are specified at the same time, the option specified last takes precedence.
- If the -no option is specified, the -r/-nr option becomes unavailable.

[Example of use]

- To sort HEX-format objects in order of address, describe as:

```
C>oc78k0r k0r.lmf -r
```

Object complement specification

(1) -u/-nu

[Syntax]

```
-ucomplement-value[,[start-address],size]
-nu
```

- Default assumption

  -u0FFH (filled with 0FFH)

[Function]

- The -u option outputs a specified complement value as an object code for an address area to which no HEX-format object has been output.
- The -nu option makes the -u option unavailable.

[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 the -u option, code can be written in advance to address areas to which no HEX-format object has been output.

[Explanation]

- The range of values that can be specified as complement values is as follows.

  0H ≤ complement values ≤ 0FFH

  complement-value can be specified in binary, octal, decimal or hexadecimal numbers. An abort error occurs if a value outside the range or a value other than a numerical value is specified.
- *start-address* specifies the start address area for complement to be performed.
  The range of values that can be specified for start address is as follows.

  0H ≤ start address ≤ 0FFEFFH

  start-address can be specified in binary, octal, decimal or hexadecimal numbers. An abort error occurs if a value outside the range or a value other than a numerical value is specified. If *start-address* is omitted, 0 is assumed to be specified.
- *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 ≤ 0FFF00H

  size can be specified in binary, octal, decimal or hexadecimal numbers. An abort error occurs if a value outside the range or a value other than a numerical value is specified. When *start-address* has been specified, *size* cannot be omitted.

- If neither a start address nor size is specified, the object converter performs processing assuming that the range of the internal ROM is specified.
- If both the -u and -nu options are specified at the same time, the one specified later takes precedence.
- If the -no option is specified, the -u/-nu option is invalid.
- Two or more address ranges cannot be specified by using the -u option.
- Specification formats for start address and size in the -u option and their interpretation are as follows.

  (a) -u*complement-value*

  If the target device for assembly contains internal ROM, the internal ROM range

  (b) -u*complement-value,size*

  From address 0 to the size - 1 address

  (c) -u*complement-value,start-address,size*

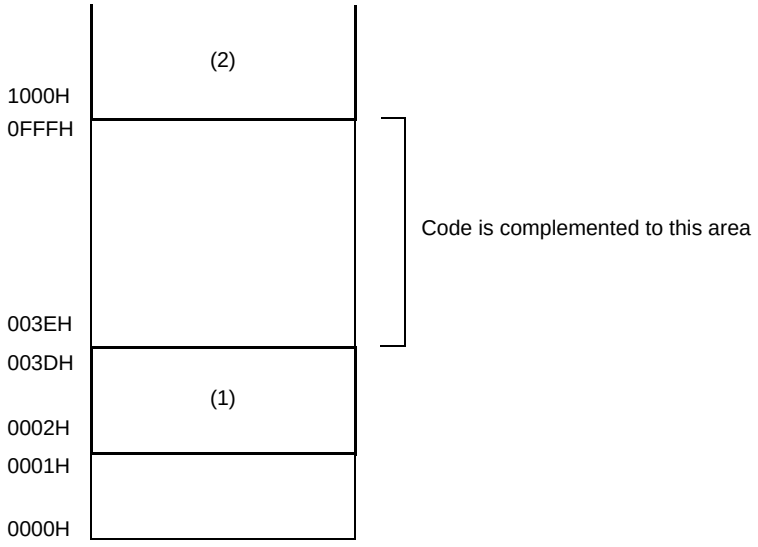
  From start address to start address + size - 1 address

[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     ; (1)
: 100012001A6720FE2822006521FED350D25014FE1A     ; (1)
: 10002200B900059F2835002431B900059F28350005     ; (1)
: 0C003200242156AF0A8302A807A830560C
: 01000003B5D0d0026A3...                         ; (2)
: 1010100024A5F622B667...                        ; (2)
    :
: 00000001FF
```



To complemente 00H to the address area 003EH to 0FFFH, describe as:

```
C>oc78k0r k0r.lmf -u00h, 003eh, 0fc2h
```

Error list file output specification

(1) -e/-ne

[Syntax]

```
-e[output-file-name]
-ne
```

- Default assumption

  -ne

[Function]

- The -e option specifies the output of an error list file. The -e option also specifies the output destination and output file name.
- The -ne option makes the -e option unavailable.

[Application]

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

[Explanation]

- The file name of the error list file can be specified as a disk-type file name or as a device-type file name.
- When the -e option is specified and *output-file-name* is omitted, the error list file name will be *input-file-name*.eoc.
- When the -e option is specified and the drive name is omitted, the error list file will be output to the current drive.
- If both options -e and -ne are specified at the same time, the option specified last takes precedence.

[Example of use]

- To create an error list file (k0r.eoc), describe as:

```
C>oc78k0r k0r.lmf -ek0r.eoc
```

Parameter file specification

(1) -f

[Syntax]

```
-ffile-name
```

- Default assumption

  Options and input file names can only be input from the command line.

[Function]

- The -f option specifies input of options and input file names from a specified file.

[Application]

- Specify the -f option when the data required to start up the object converter will not fit on the command line.
- Specify the -f option to repeatedly specify the same options each time object conversion is performed and to save those options to a parameter file.

[Explanation]

- Only a disk-type file name can be specified as file-name. If a device-type file name is specified, an abort error occurs.
- If *file-name* is omitted, an abort error occurs.
- Nesting of parameter files is not permitted. If the -f option is specified within a parameter file, an abort error occurs.
- The number of characters that can be written within a parameter file is unlimited.
- Separate options or input file names with a blank space, a tab or a line feed code (LF).
- 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.
- The expanded options specified last will take precedence.
- All characters entered after ";" or "#" and before a line feed code (LF) or "EOF" will be interpreted as comments.
- If the -f option is specified two or more times, an abort error occurs.

[Example of use]

- Perform object conversion using a parameter file (78k0r.poc).

  <Contents of 78k0r.poc>.

```
; parameter file
k0r.lmf -osample.hex
-ssample.sym -r
```

Enter the following on the command line.

```
C>oc78k0r k0r.lmf -f78k0r.poc
```

HEX format specification

(1) -ki/-kie/-kt/-km/-kme

[Syntax]

```
-ki
-kie
-kt
-km
-kme
```

- Default assumption

  -kie

[Function]

- Specifies format of a HEX-format object module file to be output.

[Application]

- Use these options to specify the format of a HEX-format object module file to be output from among [Intel standard HEX format], [Intel extended HEX format], [Extended Tech format], [Motorola S-type format (standard address)] and [Motorola S-type format (32-bit address)].

[Explanation]

- The address ranges specified by each of these options are as follows.

Options	Output Format	Range
-ki	Intel standard HEX format	0H to FFFFH (up to 64 KB)
-kie	Intel extended HEX format	0H to FFFFFH (up to 1 MB)
-kt	Extended Tech format	0H to FFFFFFFFH (up to 4 GB)
-km	Motorola S-type format (standard address)	0H to FFFFFFH (up to 16 MB)
-kme	Motorola S-type format (32-bit address)	0H to FFFFFFFFH (up to 4 GB)

[Example of use]

- To specify a HEX-format object module file to be output as the Motorola S format (standard address), describe as:

```
C>oc78k0r k0r.lmf -km
```

Device file search path specification

(1) -y

[Syntax]

```
-ypath-name
```

- Default assumption

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

  (i) Path registered in the device file installer
  (ii) Path by which OC78K0R was started up
  (iii) Current folder
  (iv) The environmental variable PATH

[Function]

- The -y option reads a device file from the specified path.

[Application]

- Specify a path where a device file exists.

[Explanation]

- If anything other than a path name is specified after the -y option, an abort error occurs.
- If *path-name* is omitted after the -y option, an abort error occurs.
- The path from which the device file is read in the order determined as follows.

  (i) Path specified by the -y option
  (ii) Path registered in the device file installer
  (iii) Path by which OC78K0R was started up
  (iv) Current folder
  (v) The environmental variable PATH

[Example of use]

- To specify the path for the device file as folder C:\78k0r\dev, describe as:

```
C>oc78k0r k0r.lmf -yC:\78k0r\dev
```

- To specify the path for the device file as folder C:\Program Files\NEC Electronics Tools\device files, describe as:

```
C>oc78k0r k0r.lmf -y"C:\Program Files\NEC Electronics Tools\device files"
```

File separate output specification for flash memory model

(1) -zf

[Syntax]

```
-zf
```

- Default assumption

  Not separately output.

[Function]

- The -zf option outputs the boot area and other areas to separate HEX-format files.

[Explanation]

- Adds an option that outputs the boot area and other areas to separate HEX-format files when linking the boot area ROM program of a flash memory model is specified.
- If the -zf option is specified, the output file type at the boot area ROM program side is "hxb", and the output file type at the side of the other programs is "hxf".

  Caution Do not specify this option for a device that does not have a flash memory area self programming function.

[Example of use]

- To outputs the boot area and other areas to separate HEX-format files (k0r.hxb, k0r.hxf), describe as:

```
C>oc78k0r k0rlmf -zf
```

Help specification

(1) --

[Syntax]

```
--
```

- Default assumption

  No display

[Function]

- The -- 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 the -- option is specified, all other options are unavailable.

  Caution This option cannot be specified from PM+.

  To reference PM+ help, click the [Help] button in the [Object Converter Options] dialog box.

[Example of use]

- To output a help message on the display, describe as:

```
C>oc78k0r --
```

```
78K0R Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

usage : oc78k0r [option[...]] input-file [option[...]]
The option is as follows ([ ] means omissible).
-ffile       : Input option or input-file name from specified file.
-o[file]/-no : Create HEX module file [with specified name]/Not.
-s[file]/-ns : Create symbol table file [with specified name]/Not.
-e[file]/-ne : Create the error list file [with the specified name]/Not.
-r/-nr       : Sort HEX object by address/Not.
-uvalue[,[start],size]/-nu : Fill up HEX object with specified value/Not.
-kkind       : Select hex format.  I;intel format  IE;intel extend format
                                   T;tex format  M;s format  ME;s-32bit format
-ydirectory  : Set device file search path.
-zf     : Create boot hex module file (HXB), and flash hex module file(HXF).
--           : Show this message.
DEFAULT ASSIGNMENT  : -o -s -r -u0ffh
```

6.5 Option Settings in PM+

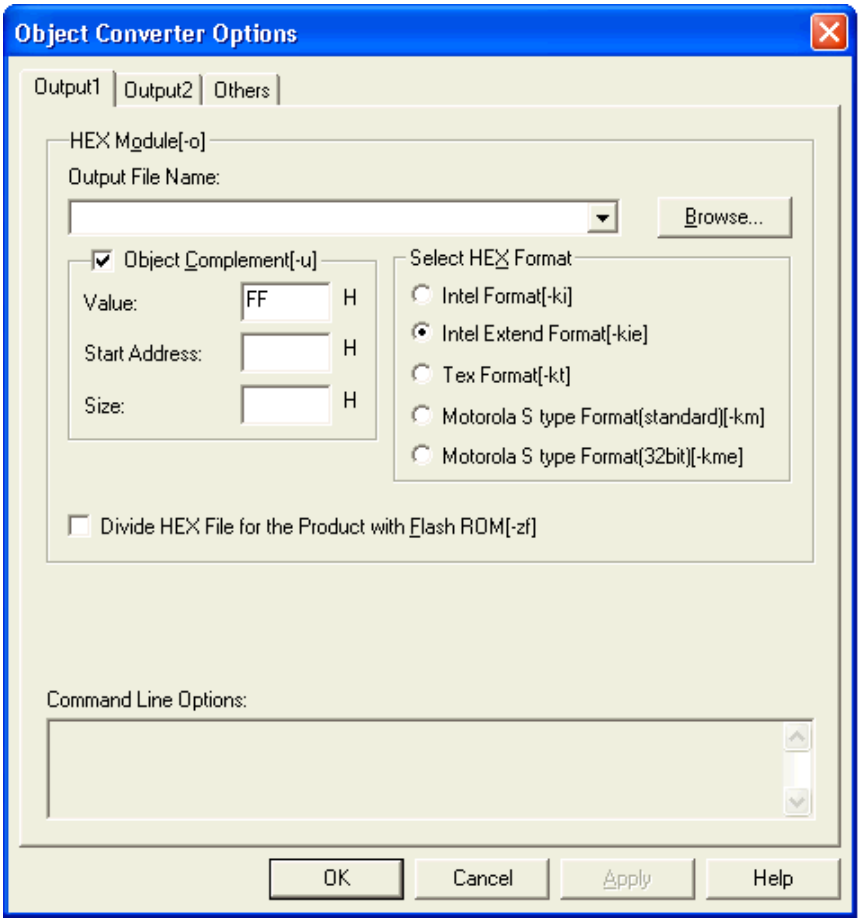
This section describes the method for setting object converter options from PM+.

6.5.1 Option setting method

The [Object Converter Options] dialog box is opened if [Object Converter Options] is selected from the [Tools] menu of PM+ or if the [OC] button on the toolbar is clicked.

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

Figure 6-7 [Object Converter Options] Dialog Box

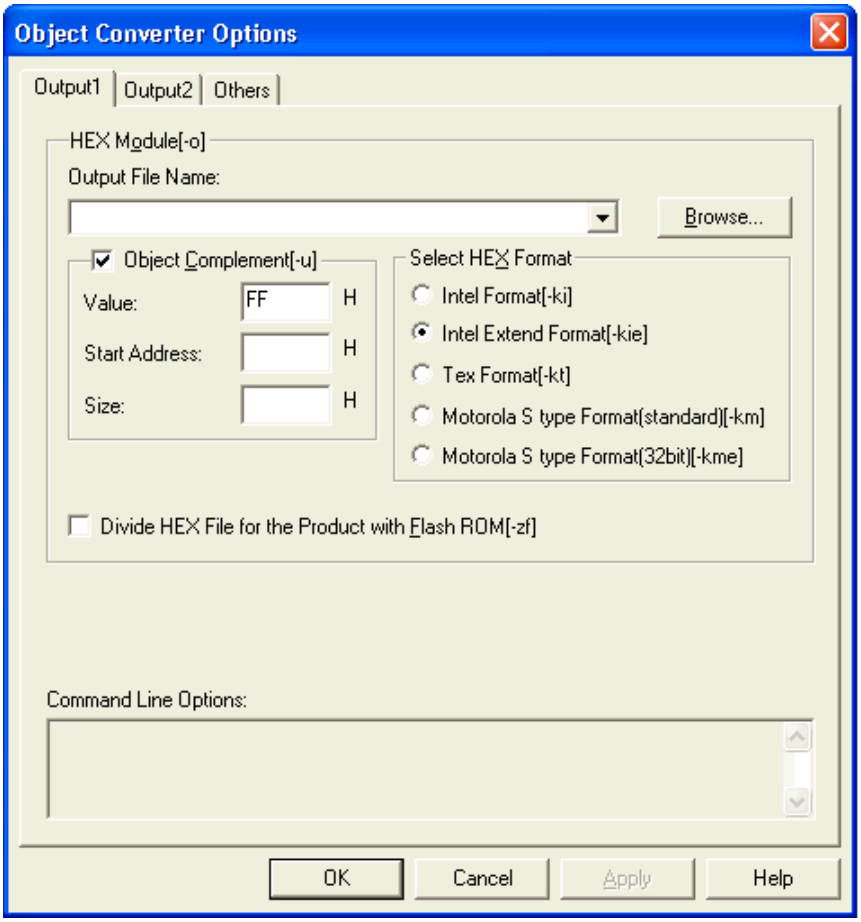


6.5.2 Explanation of dialog box

The various tabs in the [Object Converter Options] dialog box are described below.

(1) [Output1] tab

Figure 6-8 [Object Converter Options] Dialog Box (When [Output1] Tab Is Selected)



- HEX Module[-o]

Output File Name	Specify the path and file name of the object module file by using the [Browse...] button or directly inputting a path and file name. The number of characters that can be input is up to 259.

- Object Complement[-u]

  Specify this object to write a code in advance to addresses to which a HEX-format object is not output, to prevent unwanted codes from being written to those addresses and the program from hanging up (by default).

Value	Specifies the value for complement to be performed. The range of values that can be specified for complement value is as follows. 0H ≤ complement value ≤ 0FFH FF is specified by default.
Start Address	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 address ≤ 0FFEFFH
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 ≤ 0FFF00H

- Select HEX Format

  Select the HEX format (Intel standard HEX format [-ki], Intel extended HEX format [-kie], Extended Tech format [-kt], Motorola S-type format (standard address) [-km], Motorola S-type format (32-bit address) [-kme]) of the object to be output.

  “Intel extended HEX format [-kie]“ is selected by default.

  Caution This option cannot be specified for a device that does not have a memory bank function.

- Divide HEX File for the Product with Flash ROM[-zf]

  Select this option to output the boot area and other areas of a product that contains flash memory to separate HEX-format files.

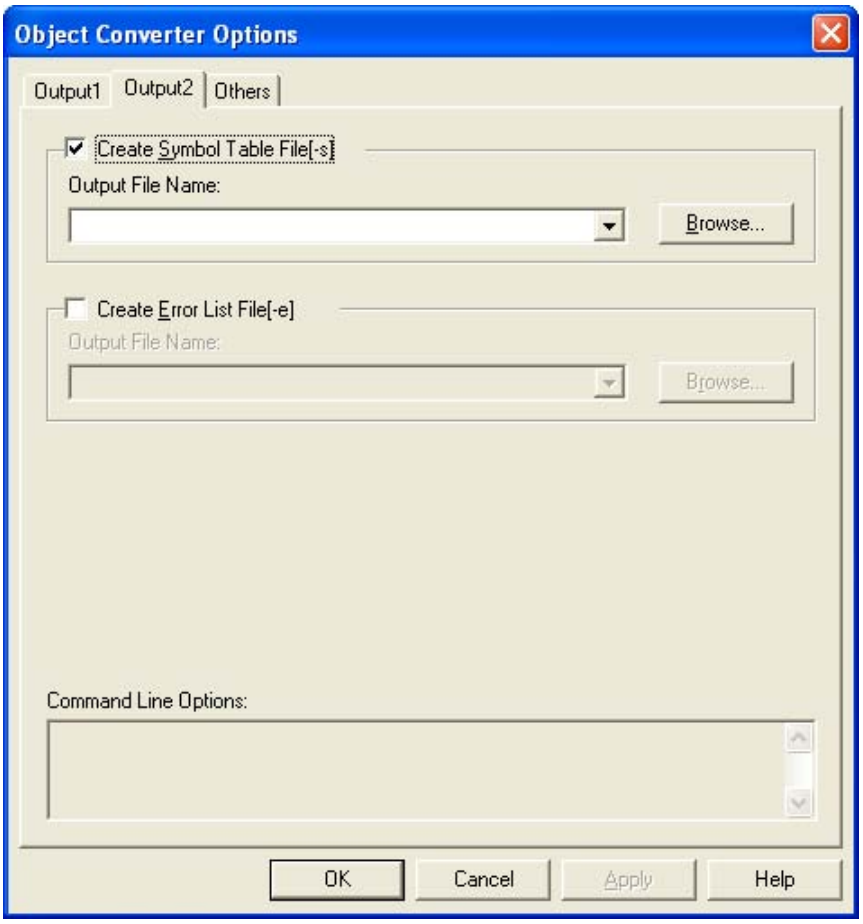
  Caution Do not specify this option for a device that does not have a flash memory area self programming function.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

(2) [Output2] tab

Figure 6-9 [Object Converter Options] Dialog Box (When [Output2] Tab Is Selected)



- Create Symbol Table File[-s]

  Select this option to output a symbol table file (by default).

Output File Name	Specify the path and file name of a symbol table file by using the [Browse...] button or directly inputting a path and file name. The number of characters that can be input is up to 259.

- Create Error List File[-e]

  Select this option to output an error list file.

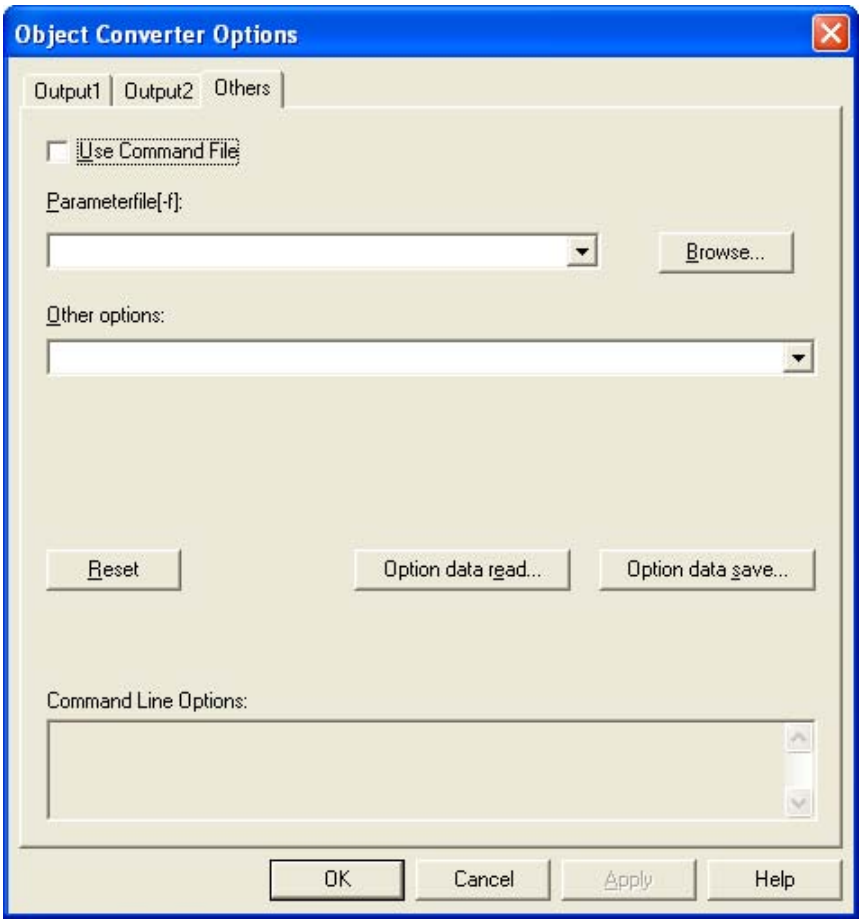
Output File Name	Specify the path and file name of the error list file by using the [Browse...] button or directly inputting a path and file name. The number of characters that can be input is up to 259.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed (by default).

(3) [Others] tab

Figure 6-10 [Object Converter Options] Dialog Box (When [Others] Tab Is Selected)



- Use Command File

  Select this option to create a command file.

- Parameterfile[-f]

  Specify the file to be input as a user-defined parameter file, by using the [Browse...] button or directly inputting a file name.

- Other options

  To specify an option other than those that can be set in this dialog box, enter the option in the input box.

  Caution The help specification (--) option cannot be specified on PM+.

- [Reset] button

  Resets the input contents.

- [Option data read...] button

  Opens the [Read Option Data] dialog box and after the option data file has been specified, reads this file.

- [Option data save...] button

  Opens the [Save Option Data] 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.

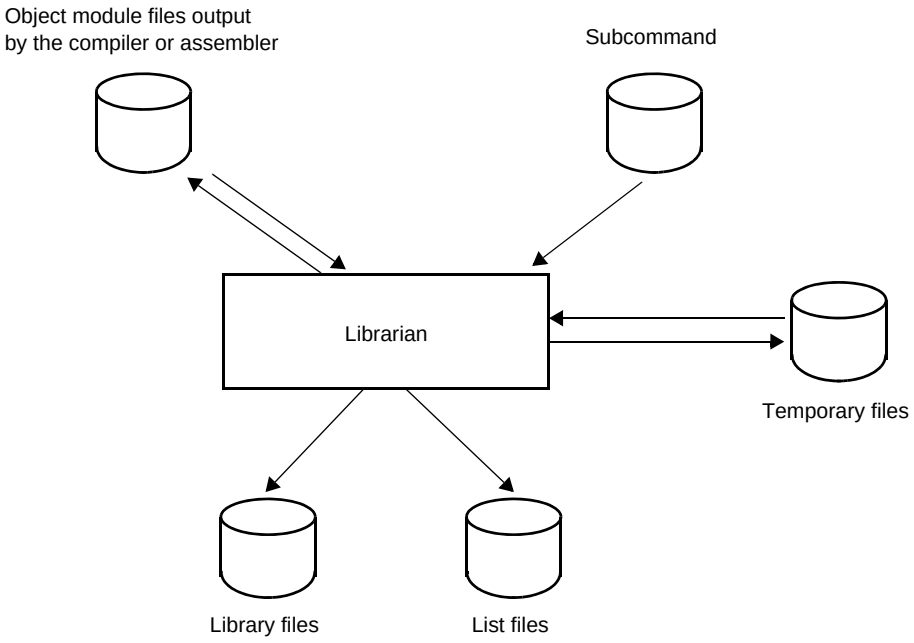
CHAPTER 7 LIBRARIAN

The librarian edits RA78K0R 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.

Figure 7-1 I/O Files of Librarian



7.1 I/O Files of Librarian

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

Table 7-1 I/O Files of Librarian

Type	File Name	Explanation	Default File Type
Input files	Subcommand files	- Files containing the parameters for the executed programs (user-created files)	None
Output files	List files	- Files containing the result of library file data output.	.lst
I/O files	Object module files	- Object module files output by the compiler or assembler.	.rel
	Library files	- Files used to input the library files output by the librarian and update the contents	.lib
	Temporary files	- Files created automatically by the librarian when forming a library Temporary files are deleted when execution of the librarian ends.	Lbxxxxxx.$$y (y = 1-6)

7.2 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 RA78K0R 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.

(a) Addition of modules to library files

(b) Deletion of modules from library files

(c) Replacement of modules in library files

(d) Retrieval of modules from library files

For detailed explanations of these functions, refer to "7.5 Subcommands".

(3) Output of library file data

The librarian incorporates functions for the editing and output of the following items of data stored in library files.

(a) Module names

(b) Created programs

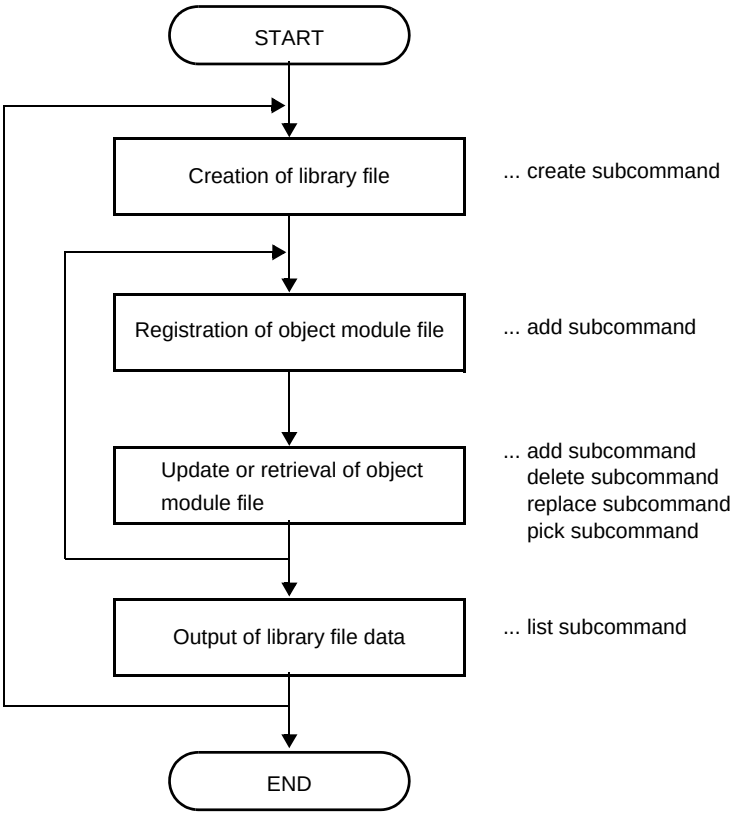
(c) Date of registration

(d) Date of update

(e) PUBLIC symbol data

Caution The librarian performs functions (2) and (3) explained above using subcommands. The librarian determines each subcommand in order while performing processing. For an explanation of the operation of subcommands, refer to "7.5 Subcommands".

The general procedure for creating library files is as follows.



7.3 Librarian Startup

7.3.1 Methods to start librarian

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

(1) Startup from the command line

```
 X>[path-name]lb78k0r[Δoption] ...
 |       |       |       |
(a)     (b)     (c)     (d)
```

(a) Current drive name

(b) Current folder name

(c) Librarian command file name

(d) This contains detailed directions for the action of the librarian

If more than one librarian option is specified, separate the options with a space. Uppercase characters and lowercase characters are not distinguished for the librarian options. For a detailed explanation of librarian options, refer to "7.4 Librarian Options".

Enclose a path that includes a space in a pair of double quotation marks (" ").

[Example]

```
C>lb78k0r -ll20 -lw80
```

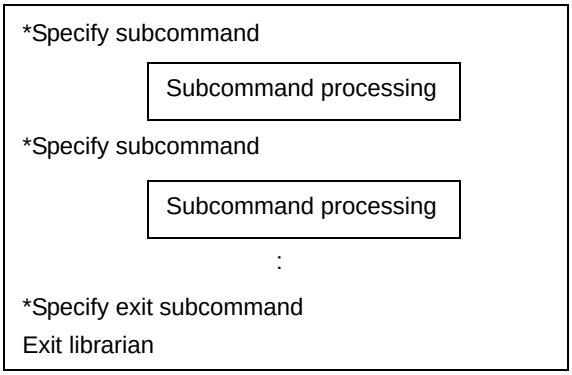
When the librarian is started up, the following startup message appears on the display.

```
78K0R Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx
*
```

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

```
*create k0r.lib
*add k0r.lib k0rmain.rel k0rsub.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>lb78k0r∆<subcommand-file-name[∆option]...
          |           |
         (a)         (b)
```

(a) Be sure to add this when specifying a subcommand file

(b) File in which subcommands are stored

Remark Create the parameter file using an editor.

The rules for writing the content of a subcommand file are as follows.

```
Subcommand-name operand-data
Subcommand-name operand-data
   :
EXIT
```

- When repeating one subcommand, describe "&" at the end of each line to indicate continuation.
- 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.
- If the last subcommand in a subcommand file is not the exit subcommand, the librarian will automatically interpret an exit subcommand.
- 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]

(1) Create the subcommand file (k0r.slb) using an editor.

```
; library creation command
create k0r.lib
add k0r.lib k0rmain.rel &
k0rsub.rel
;
exit
```

(2) Use subcommand file k0r.slb to start up the librarian.

```
C>lb78k0r <k0r.slb
```

7.3.2 Execution start and end messages

(1) Execution start message

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

```
78K0R 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 k0r.lib
*add k0r.lib k0rmain.rel k0rsub.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>lb78k0r -a

78K0R Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx
RA78K0R error F5018 : Option is not recognized '-z'
Usage : LB78K0R [options]
```

In the above example, a non-existent librarian option is specified. An error occurs and the librarian aborts librarian execution.

When an error message is displayed and library formation is aborted, look for the cause in "CHAPTER 11 ERROR MESSAGES" and take action accordingly.

7.4 Librarian Options

7.4.1 Types of librarian options

The librarian options provide detailed directions for the operation of the librarian.

The types and explanations for librarian options are shown below.

Table 7-2 Librarian Options

Classification	Option	Explanation
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 page feed code at the end of a list file.
	-nlf	
Specification of path for temporary file creation	-t	Creates a temporary file in a specified path.
Help specification	- -	Displays a help message on the display.

List file format specification

(1) -lw

[Syntax]

```
-lw[number-of-characters]
```

- Default assumption

  -lw132 (80 characters in the case of display output)

[Function]

- The -lw option changes the number of characters that can be printed in 1 line in a list file.

[Application]

- Specify the -lw option to change the number of characters that can be printed in 1 line in a list file.

[Explanation]

- The range of number of characters that can be specified with the -lw option is shown below.
  (In the case of display output, this number is 80)

  72 ≤ number of characters printed on 1 line ≤ 260

  An abort error occurs if a numerical value outside this range, or something other than a numerical value, is specified.
- If *number-of-characters* is omitted, 132 will be specified. If the list file is output to the display, 80 is specified.
- The specified number of characters does not include the terminator (CR, LF).
- If the list subcommand is not specified, the -lw option is ignored.
- If the -lw option is specified 2 or more times, the last specified item will take precedence.

[Example of use]

- To specify 80 as the number of characters per line in a list file, describe as:

```
C>lb78k0r -lw80
```

(2) -ll

[Syntax]

```
-ll[number-of-lines]
```

- Default assumption

  -ll0 (No page breaks)

[Function]

- The -ll option specifies the number of lines that can be printed in 1 page in a list file.

[Application]

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

[Explanation]

- The range of number of lines that can be specified with the -ll option is shown below.

  20 ≤ number of lines printed on 1 page ≤ 32,767

- An abort error occurs if a numerical value outside this range, or something other than a numerical value, is specified.
- If *number-of-lines* is omitted, 0 will be specified.
- If the number of lines specified is 0, no page breaks will be made.
- If the list subcommand is not specified, the -ll option is ignored.
- If the -ll option is specified 2 or more times, the last specified item will take precedence.

[Example of use]

- To specify 20 as the number of lines per page in a list file, describe as:

```
C>lb78k0r -ll20
```

(3) -lf/-nlf

[Syntax]

```
-lf
-nlf
```

- Default assumption

  -nlf

[Function]

- The -lf option inserts a form feed (FF) code at the end of a list file.
- 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 the -lf option to add a form feed code.

[Explanation]

- If the list subcommand is not specified, the -lf option is ignored.
- If both options -lf and -nlf are specified at the same time, the option specified last takes precedence.

[Example of use]

- To add a page feed code to a list file, describe as:

```
C>lb78k0r -lf
```

Specification of path for temporary file creation

(1) -t

[Syntax]

```
-tpath-name
```

- Default assumption

  Path specified by environmental variable TMP

  Current path, if no path is specified.

[Function]

- The -t option creates a temporary file in a specified path.

[Application]

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

[Explanation]

- Only a path can be specified as a path name.
- *path-name* cannot be omitted.
- Even if a previously created temporary file exists, if the file is not protected it will be overwritten.
- 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.
- Temporary files are deleted when library formation is finished. They are also deleted when library formation is aborted by pressing (CTRL+C).
- The path in which the temporary file is to be created is determined according to the following order.

  (i) The path specified by the -t option

  (ii) The path specified by environmental variable TMP (when the -t option is omitted)

  (iii) The current path (when TMP is not set)

  When (i) or (ii) is specified, if the temporary file cannot be created in the specified path an abort error occurs.

[Example of use]

- To output a temporary file to folder C:\tmp, describe as:

```
C>lb78k0r -tC:\tmp
```

- To output a temporary file to folder C:\Program Files\NEC Electronics Tools\temporary files, describe as:

```
C>lb78k0r -t"C:\Program Files\NEC Electronics Tools\temporary files"
```

Help specification

(1) --

[Syntax]

```
--
```

- Default assumption

  No display

[Function]

- The -- 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 the -- option is specified, all other options are unavailable.

  Caution This option cannot be specified from PM+.

  To reference PM+ help, click the [Help] button in the [Library File Name] dialog box.

[Example of use]

- To output a help message on the display, describe as:

```
C>lb78k0r --
```

```
78K0R Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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                                                      |
+-------------------------------------------------------------------------+
```

7.5 Subcommands

7.5.1 Types of subcommands

The subcommands provide detailed directions for the operation of the librarian.

The explanations for subcommands are shown below.

Table 7-3 Subcommands

Subcommand Name	Abbrev.	Explanation
create	c	Creates a new library file.
add	a	Adds a module to a library file.
delete	d	Deletes a module from a library file.
replace	r	Replaces module in a library file with other modules.
pick	p	Retrieves a module from a library file.
list	l	Outputs data on modules in a library file.
help	h	Displays a help message on the display.
exit	e	Exits librarian.

7.5.2 Explanation of subcommands

The following is a detailed explanation of the function and operation of each subcommand.

[General format of command files]

```
*Subcommand[Δoption]Δlibrary-file-name[Δoption]transaction[Δoption]
                              |                        |
                             (a)                      (b)
```

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

(b) *Transaction* = Δ*object-module-file-name*Δ*library-file-name*[Δ(Δ*module-name*[Δ, ...])]

Remark Uppercase characters and lowercase characters are not distinguished for the subcommands and options.

create

[Syntax]

```
create∆library-file-name[∆transaction]
```

- Abbreviated form

  c

[Function]

- The create subcommand creates a new library file.

[Explanation]

- The size of the created library file becomes 0.
- When a transaction is specified, a module is registered at the same time as the library file is created.
- *library-file-name*: If a file with the same name already exists, it will be overwritten.
- *transaction*: An object module file carrying the same public symbol as the public symbol in the library file cannot be registered.

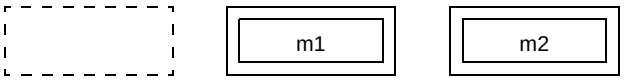
  A module with the same name as a module in the library file cannot be registered.
- If an error occurs, processing is interrupted and the library file cannot be created.

[Example of use]

- To register modules m1 and m2 at the same time as a library file (k0r.lib) is created, describe as:

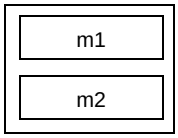
```
*create k0r.lib m1.rel m2.rel
```

<Before file creation>



<After file creation>

k0r.lib



add

[Syntax]

```
add∆library-file-name∆transaction
```

- Abbreviated form

  a

[Function]

- The add subcommand adds a module to a library file.

[Explanation]

- A module can be added to a library file even if no modules are currently stored in the library.
- An abort error occurs if a module with the same name as the module to be added already exists in the library file.
- An abort error occurs if the module to be added carries the same public symbol as the public symbol in the library file.

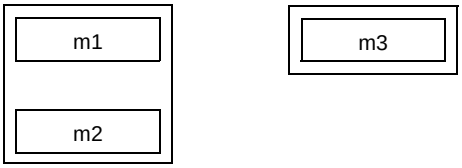
[Example of use]

- To add a module (m3) to a library file (k0r.lib), describe as:

```
*add k0r.lib m3.rel
```

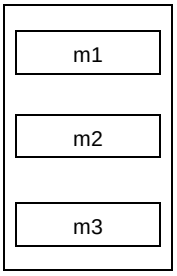
<Before addition>

k0r.lib



<After addition>

k0r.lib



delete

[Syntax]

```
delete∆library-file-name∆(∆module-name[∆, ...]∆)
```

- Abbreviated form

  d

[Function]

- The delete subcommand deletes a module from a library file.

[Explanation]

- If the specified module does not exist in the library file, an error occurs.
- If an error occurs, processing is interrupted and the condition of the library file will not be changed.

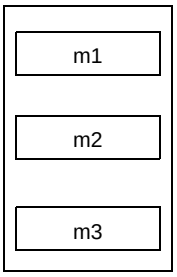
[Example of use]

- To delete modules (m1, m3) from a library file (k0r.lib), describe as:

```
*delete k0r.lib ( m1.rel , m3.rel )
```

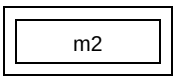
<Before deletion>

k0r.lib



<After deletion>

k0r.lib



replace

[Syntax]

```
replaceΔlibrary-file-nameΔtransaction
```

- Abbreviated form

  r

[Function]

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

[Explanation]

- An abort error occurs if no module in the library file has the same name as the replacement module.
- An abort error occurs if a public symbol contained in the replacement module is the same as a public symbol in the library file.
- The file name of the replacement object module must be the same as the file name used in registration.
- If an error occurs, processing is interrupted and the condition of the library file will not be changed.

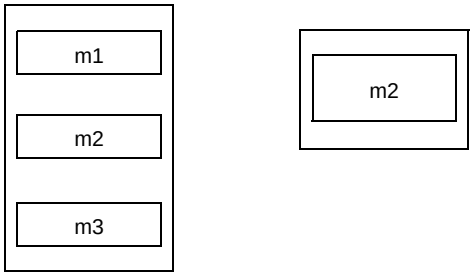
[Example of use]

- To replace a module (m2) in a library file (k0r.lib), describe as:

```
*replace k0r.lib m2.rel
```

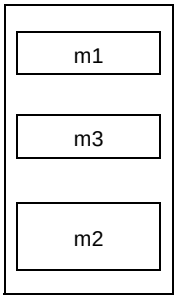
<Before replacement>

k0r.lib



<After replacement>

k0r.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.

pick

[Syntax]

```
pick∆library-file-name∆(∆module-name[∆, ...]∆)
```

- Abbreviated form

  p

[Function]

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

[Explanation]

- The retrieved module becomes an object module file with the file name under which it was registered in the library file.
- If the specified module name does not exist in the library file, an error occurs.
- 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.

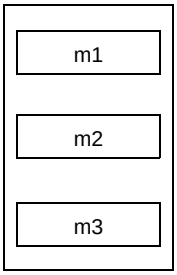
[Example of use]

- To retrieve a module (m2) from a library file (k0r.lib), describe as:

```
*pick k0r.lib ( m2.rel )
```

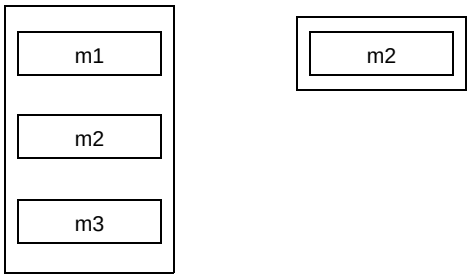
<Before pickup>

k0r.lib



<After pickup>

k0r.lib



list

[Syntax]

```
list[Δoption]Δlibrary-file-name[Δ(Δmodule-name[Δ, ...]Δ)]
        Option: -public/-nopublic
              : -oΔfile-name
```

- Abbreviated form

  l

[Function]

- The list subcommand outputs data on modules in a library file.

[Explanation]

- Multiple options may be specified. Uppercase characters and lowercase characters are not distinguished for the options.
- -o:

  A device-type file name can be specified as the output file name.

  If *output-file-name* is omitted, an error occurs.

  If the file type is omitted, the librarian assumes that "*input-file-name*.lst" is entered.
- -public/-nopublic:

  It can also be specified as -p/-np.

  -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]

- To output a module information in a library file (k0r.lib) to a list file (k0r.lst), describe as:

```
*list -p -ok0r.lst k0r.lib
```

  At this time, specify the -p option so as to output public symbol information.

<Contents of k0r.lst>

```
78K0R librarian Vx.xx                    DATE : xx xxx xx PAGE : 1

LIB-FILE NAME : k0r.lib          ( xx xxx xx )

0001  k0rmain.rel        ( xx xxx xx )

  MAIN                           START

  NUMBER OF PUBLIC SYMBOLS :     2

0002    k0rsub.rel       ( xx xxx xx )

  CONVAH

  NUMBER OF PUBLIC SYMBOLS :     1
```

help

[Syntax]

```
help
```

- Abbreviated form

  ```
  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 the -- option to refer to this message during librarian execution.

[Example of use]

- To specify the help command to output the help message, describe as:

```
*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                                                        |
+-------------------------------------------------------------------------+
```

exit

[Syntax]

```
exit
```

- Abbreviated form

```
e
```

[Function]

- The exit subcommand exits the librarian.

[Explanation]

- Use this subcommand to exit the librarian.

[Example of use]

- To exit the librarian, describe as:

```
*exit
```

7.6 Option Settings in PM+

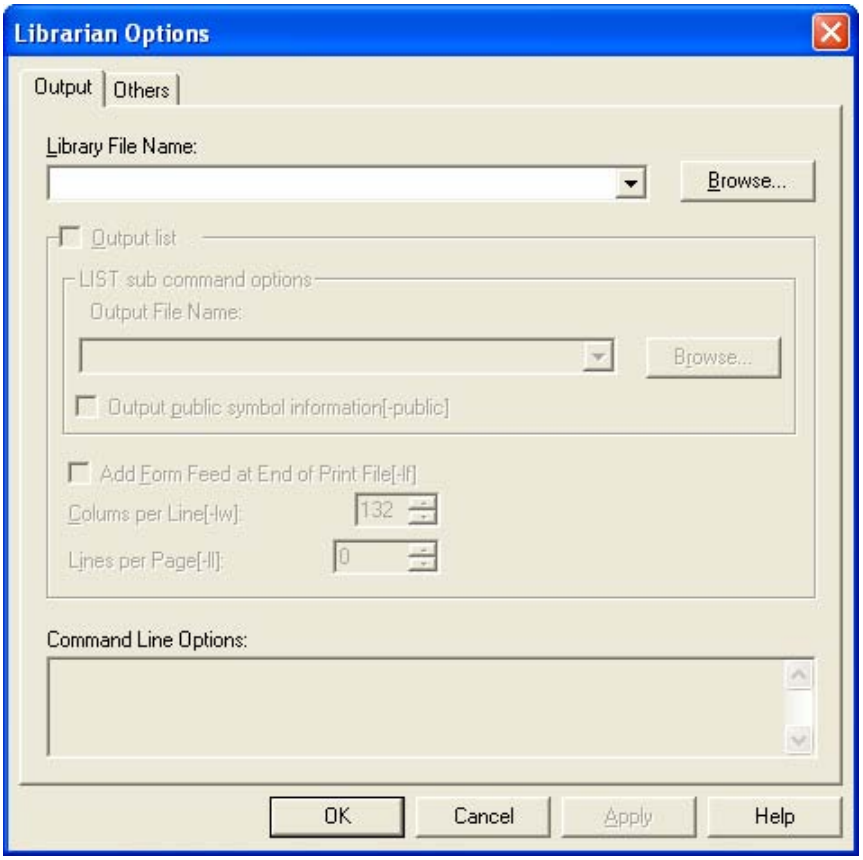
This section describes the method for specifying library flies from PM+.

7.6.1 Option setting method

Select [Librarian Options] from the [Tools] menu of PM+ or the [LB] button on the toolbar is clicked to display the [Librarian Options] dialog box.

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

Figure 7-2 [Librarian Options] Dialog Box

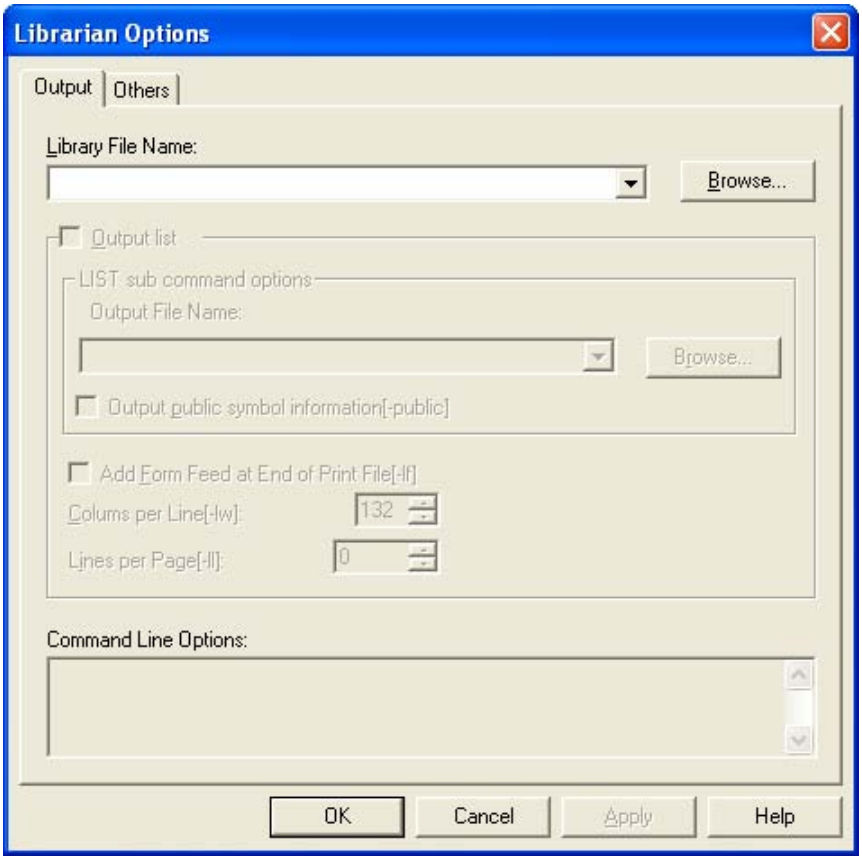


7.6.2 Explanation of dialog box

The various tabs in the [Librarian Options] dialog box are described below.

(1) [Output] tab

Figure 7-3 [Librarian Options] Dialog Box (When [Output] Tab Is Selected)



- Library File Name

  Specify the name of the library file by using the [Browse...] button or by directly inputting the file name.

  The number of characters that can be input is up to 259.

- Output list

  Select this option to output a list file.

- LIST subcommand options

Output File Name	Specify the name and path of the list file by using the [Browse...] button or by directly inputting the file name. The number of characters that can be input is up to 259.
Output public symbol information[-public]	Select this option to add public symbol information to the list file to be output.

- Add Form Feed at End of Print File[-lf]

  Select this option to add a page feed code (FF) to the end of a library file.

- Colums per Line[-lw]

  Specify the number of characters on one line of a library file.

  Between 76 and 260 characters can be specified. 132 is specified by default.

- Lines per Page[-ll]

  Specify the number of lines on one page of a library file.

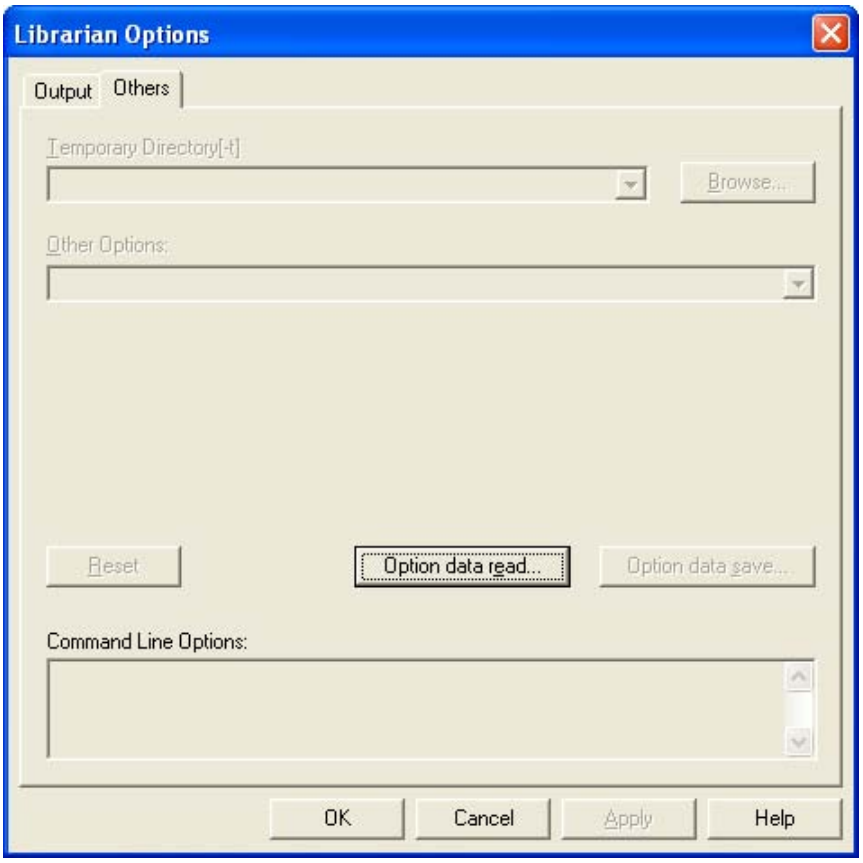
  0, and between 20 and 32,767 lines can be specified. 0 is specified by default.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

(2) [Others] tab

Figure 7-4 [Librarian Options] Dialog Box (When [Others] Tab Is Selected)



- Temporary Directory[-t]

  Specify the path where a temporary file is to be created, by using the [Browse...] button or directly inputting a path name.

- Other Options

  To specify an option other than those that can be set in the dialog box, enter it in the input box.

  Caution The help specification (--) option cannot be specified on PM+.

- [Reset] button

  Resets the input contents.

- [Option data read...] button

  Opens the [Read Option Data] dialog box and after the option file has been specified, reads this file.

- [Option data save...] button

  Opens the [Save Option Data] dialog box and saves the option file under the specified name.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

7.7 Method for Manipulating Library Files from PM+

This section describes the method for manipulating library files from PM+.The [Edit Option] dialog box is described below.

7.7.1 Method for manipulating

Register the execution format for starting LB in standalone mode by selecting [Register Ex-tool] from the [Tools] menu of PM+.

Selecting the registered icon then opens the [Library File Name] dialog box.

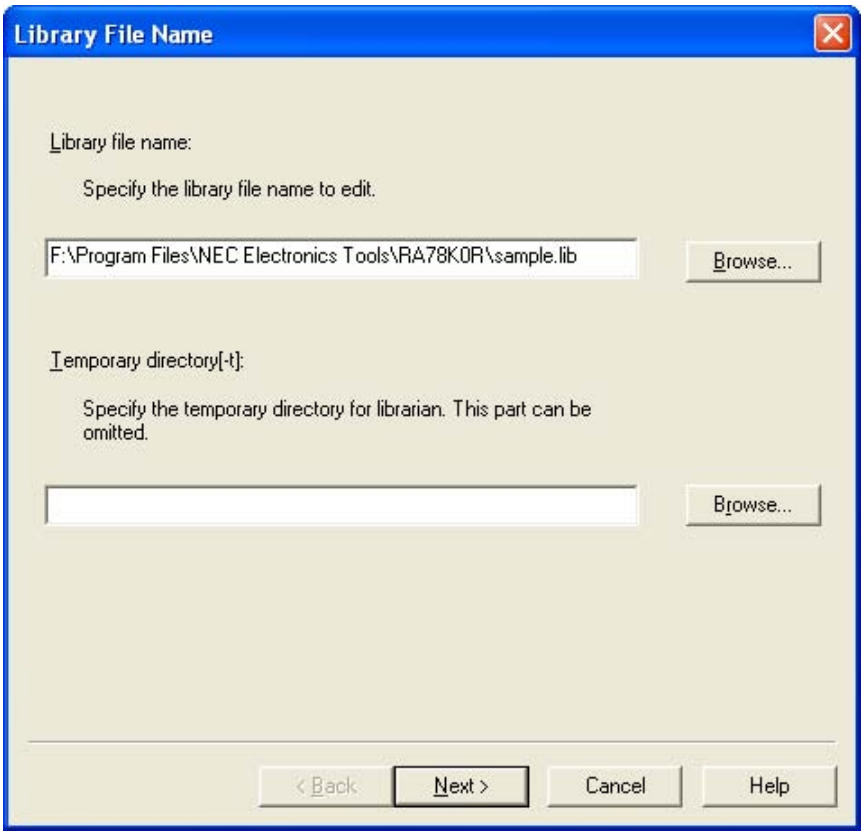
After specifying the path and the file name, click the [Next] button to display the [Subcommand] dialog box.

7.7.2 Explanation of dialog boxes

The various items in the [Library File Name] dialog box and [Subcommand] dialog box are described below.

(1) [Library File Name] Dialog Box

Figure 7-5 [Library File Name] Dialog Box



- Library file name

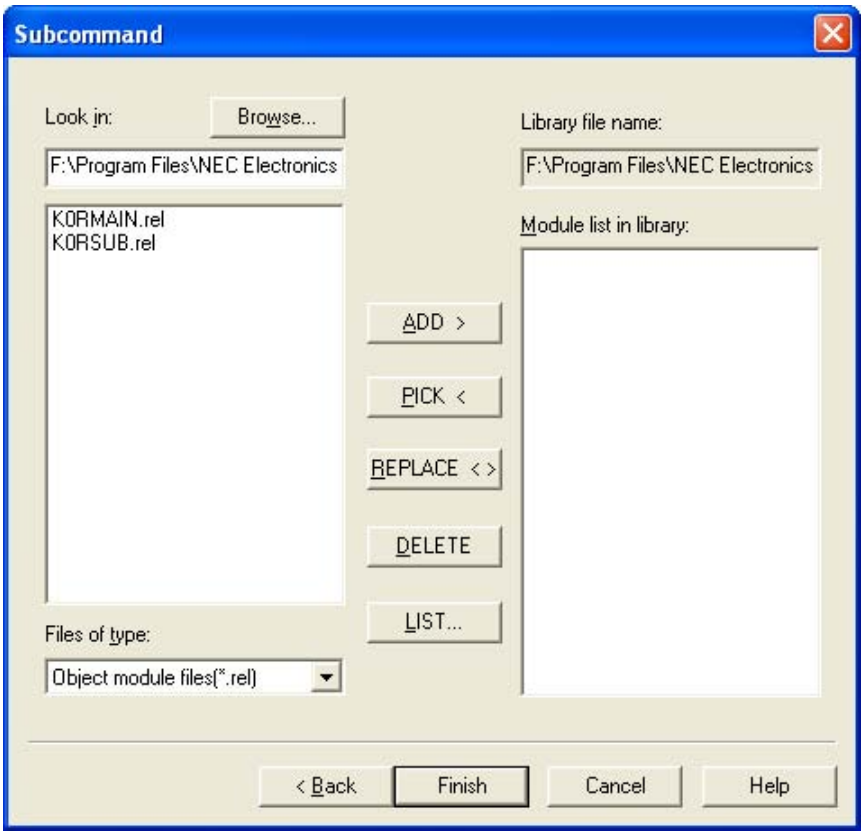
  Specify the name of the library file by using the [Browse...] button or by directly inputting the file name.

- Temporary directory[-t]

  Specify the path where a temporary file is to be created, by using the [Browse...] button or directly inputting a path name.

(2) [Subcommand] Dialog Box

Figure 7-6 [Subcommand] Dialog Box



- Look in

  Specify the path where the object module file to be used as a library exists by using the [Browse...] button or directly inputting a path.

  If a path is specified, a file list is displayed.

- Files of type

  Specify the type of the file to be displayed on the list of files.

- Library file name

  This edit box is read-only. It displays the file name of the library currently specified.

- Module list in library

  A list of the object module files in the library currently specified is displayed.

- [ADD >] button

  Add a module to an existing library file.

- [PICK <] button

  Retrieve the specified module from an existing library file.

- [REPLACE < >] button

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

- [DELETE] button

  Delete a module from an existing library file.

- [LIST...] button

  Outputs data on modules in a library file.

  The [LIST subcommand] dialog box opens.

Table 7-4 [LIST subcommand] Dialog Box

- Output file

Output file name[-o]	Specify the path and name of the output file by using the [Browse...] button or by directly inputting the file name. The number of characters that can be input is up to 259.
Add Form Feed at end of print file[-lf]	Select this option to add a page feed code (FF) to the end of an output file.
Colums per Line[-lw]	Specify the number of characters on one line of an output file. Between 76 and 260 characters can be specified. 132 is specified by default.
Lines per Page[-ll]	Specify the number of lines on one page of an output file. 0, and between 20 and 32,767 lines can be specified. 0 is specified by default.

- Output public symbol information[-public]

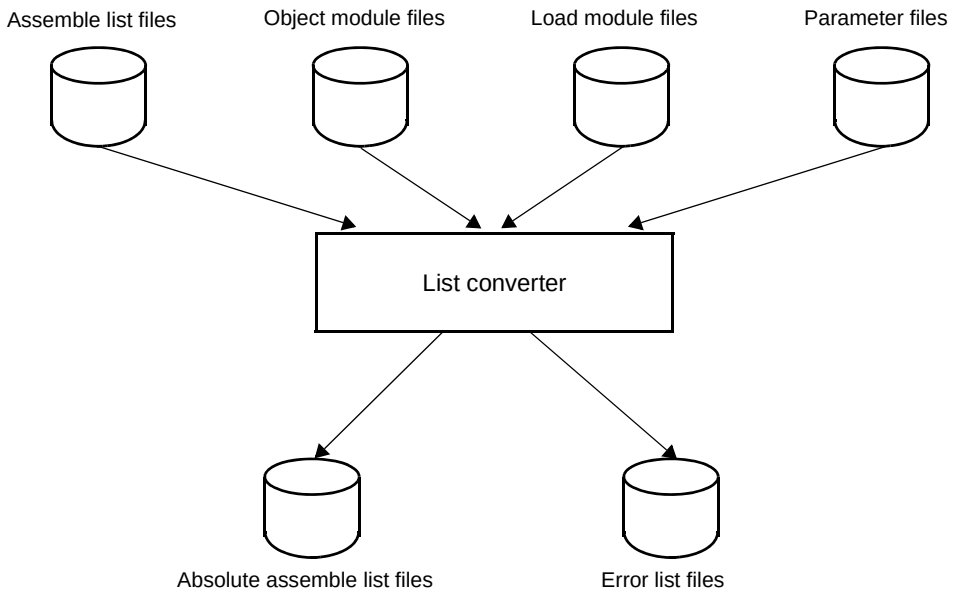
  Select this option to add public symbol information to the output file to be output.

CHAPTER 8 LIST CONVERTER

The list converter inputs assemble list files and object module files output by the assembler and load module files output by the linker.

The list converter embeds actual addresses in the relocatable addresses and symbols in the input file and outputs an absolute assembly list. This eliminates the troublesome task of looking at an assemble list while referring to a link map.

Figure 8-1 I/O Files of List Converter



8.1 I/O Files of List Converter

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

Table 8-1 I/O Files of List Converter

Type	File Name	Explanation	Default File Type
Input files	Object module files	- Binary files containing relocation data and symbol data regarding machine language data and machine language location addresses	.rel
	Assemble list files	- Files containing assembly data such as assemble lists and cross-reference lists	.prn
	Load module files	- Binary image files of the object codes output as a result of linking	.lmf
	Parameter files	- Files containing the parameters for the executed programs (user-created files)	.plv
Output files	Absolute assemble list files	- A list file which embeds actual addresses in relocatable addresses and symbols in the input file	.p
	Error list files	- Files containing error data generated during converting lists	.elv

8.2 Functions of List Converter

(1) Resolving disadvantages of assemblers (relocatable assemblers)

The list converter offers a solution to disadvantages of relocatable assembler packages by embedding the location and object codes in the assemble list file.

- The absolute assemble list output by the list converter agrees completely with the addresses used in actual program operation.
- The actual values of external symbols are embedded in the list.
- Relocatable values are embedded in the list as actual values.
- For the symbol values in symbol tables or cross-reference lists, the actual values are embedded in the list.

[Example 1] and [Example 2] are examples of the absolute assemble list file that can be acquired by the list converter.

[Example 1]

Relocation data is embedded as shown below.

<Assemble list>

```
22  22  -----           CSEG
23  23  00000       START :
24  24
25  25                  ; chip initialize
26  26  00000 RCBF80000 MOVW    SP , #_@STBEG
27  27
28  28  00004 CD201A    MOV     HDTSA , #1AH
29  29  00007 3620FE    MOVW    HL , #LOWW ( HDTSA )   ; set hex 2-code data in
HL registor
30  30
31  31  0000A RFD0000   CALL    !CONVAH     ; convert ASCII <- HEX
32  32                                      ; output BC-register <- ASCII code
33  33  0000D 3421FE    MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII
code table
34  34  00010 63        MOV     A , B
35  35  00011 99        MOV     [ DE ] , A
36  36  00012 A5        INCW    DE
37  37  00013 62        MOV     A , C
38  38  00014 99        MOV     [ DE ] , A
39  39
40  40  00015 EFFE      BR      $$
41  41
42  42                  END
```

<Absolute assemble list>

```
22  22  -----           CSEG
23  23  000D2       START :
24  24
25  25                  ; chip initialize
26  26  000D2 RCBF820FE MOVW    SP , #_@STBEG
27  27
28  28  000D6 CD201A    MOV     HDTSA , #1AH
29  29  000D9 3620FE    MOVW    HL , #LOWW ( HDTSA )   ; set hex 2-code data in
HL registor
30  30
31  31  000DC RFDE900   CALL    !CONVAH     ; convert ASCII <- HEX
32  32                                      ; output BC-register <- ASCII code
33  33  000DF 3421FE    MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII
code table
34  34  000E2 63        MOV     A , B
35  35  000E3 99        MOV     [ DE ] , A
36  36  000E4 A5        INCW    DE
37  37  000E5 62        MOV     A , C
38  38  000E6 99        MOV     [ DE ] , A
39  39
40  40  000E7 EFFE      BR      $$
41  41
42  42                  END
```

[Example 2]

The object codes are embedded as shown below.

<Assemble list>

```
22  22  -----           CSEG
23  23  00000       START :
24  24
25  25                  ; chip initialize
26  26  00000 RCBF80000 MOVW    SP , #_@STBEG
27  27
28  28  00004 CD201A    MOV     HDTSA , #1AH
29  29  00007 3620FE    MOVW    HL , #LOWW ( HDTSA )   ; set hex 2-code data in
HL registor
30  30
31  31  0000A RFD0000   CALL    !CONVAH     ; convert ASCII <- HEX
32  32                                      ; output BC-register <- ASCII code
33  33  0000D 3421FE    MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII
code table
34  34  00010 63        MOV     A , B
35  35  00011 99        MOV     [ DE ] , A
36  36  00012 A5        INCW    DE
37  37  00013 62        MOV     A , C
38  38  00014 99        MOV     [ DE ] , A
39  39
40  40  00015 EFFE      BR      $$
41  41
42  42                  END
```

<Absolute assemble list>

```
22  22  -----           CSEG
23  23  000D2       START :
24  24
25  25                  ; chip initialize
26  26  000D2 RCBF820FE MOVW    SP , #_@STBEG
27  27
28  28  000D6 CD201A    MOV     HDTSA , #1AH
29  29  000D9 3620FE    MOVW    HL , #LOWW ( HDTSA )   ; set hex 2-code data in
HL registor
30  30
31  31  000DC RFDE900   CALL    !CONVAH     ; convert ASCII <- HEX
32  32                                      ; output BC-register <- ASCII code
33  33  000DF 3421FE    MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII
code table
34  34  000E2 63        MOV     A , B
35  35  000E3 99        MOV     [ DE ] , A
36  36  000E4 A5        INCW    DE
37  37  000E5 62        MOV     A , C
38  38  000E6 99        MOV     [ DE ] , A
39  39
40  40  000E7 EFFE      BR      $$
41  41
42  42                  END
```

8.3 List Converter Startup

8.3.1 Methods to start list converter

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

(1) Startup from the command line

```
 X>lc78k0r[Δoption]...input-file-name[Δoption]...[Δ]
 |   |       |               |           |
(a) (b)     (c)             (d)         (c)
```

(a) Current drive name

(b) Command file name of the list converter

(c) Enter detailed instructions for the operation of the list converter.

When specifying two or more list converter options, separate the list converter options with a blank space. Uppercase characters and lowercase characters are not distinguished for the list converter options. For a detailed explanation of list converter options, refer to "8.4 List Converter Options".

Enclose a path that includes a space in a pair of double quotation marks (" ").

(d) Primary name of assemble list

Specify the file name of a path that includes a space by enclosing it in a pair of double quotation marks (" ").

Use the extension .prn.

If only the primary name of the assemble list is specified in the command line, the primary names of the object module file and load module file must be identical to the primary name of the assemble list file.

The file types must also be as shown below.

File Name	Type
Object module type	.rel
Load module file	.lmf

[Example]

- If the primary name is different between an assemble list file (k0rmain.prn) and a load module file (sample.lmf), describe as follows so as to specify the input of a load module file (sample.lmf).

```
C>lc78k0r k0rmain.prn -lsample.lmf
```

(2) Startup from a parameter file

Use the parameter file when the data required to start up the list converter will not fit on the command line, or when the same list converter option is specified repeatedly each time list conversion is performed.

To start up the list converter from a parameter file, specify the specify parameter file option (-f) on the command line.

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

```
C>lc78k0r[Δinput-file-name]Δ-fparameter-file-name
                              |           |
                             (a)         (b)
```

(a) Parameter file specification option

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

Remark Create the parameter file using an editor.

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

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

- If the input file name is omitted from the command line, only 1 input file name can be specified in the parameter file.
- The input file name can also be written after the option.
- Write in the parameter file all list converter options and output file names that should be specified in the command line.

[Example]

(1) Create the parameter file (k0r.plv) using an editor.

```
; parameter file
k0rmain -lk0r.lmf
-ek0r.elv
```

(2) Use parameter file (k0r.plv) to start up the list converter.

```
C>lc78k0r -fk0r.plv
```

8.3.2 Execution start and end messages

(1) Execution start message

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

```
List Conversion Program for RA78K0R 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>lc78k0r k0rmain.prn -a
```

```
List Conversion Program for RA78K0R Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F6018 : Option is not recognized '-a'
Please enter 'LC78K0R --', if you want help messages.
Program aborted.
```

When the list converter outputs an error message and aborts list conversion, look for the cause in "CHAPTER 11 ERROR MESSAGES" and take action accordingly.

8.4 List Converter Options

8.4.1 Types of list converter options

The list converter options are detailed instructions for the operation of the list converter.

The types and explanations for list converter options are shown below.

Table 8-2 List Converter Options

Classification	Option	Explanation
Object module file input specification	-r	Inputs an object module file.
Load module file input specification	-l	Inputs a load module file.
Absolute assemble list file output specification	-o	Specifies output of an absolute assemble list file.
Error list file output specification	-e	Outputs an error list file.
	-ne	
Parameter file specification	-f	Inputs the input file name and options from a specified file.
Help specification	- -	Displays a help message on the display.

Object module file input specification

(1) -r

[Syntax]

```
-r[input-file-name]
```

- Default assumption

  -r*assemble-list-file-name*.rel

[Function]

- The -r option 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 the -r option.

[Explanation]

- If a fatal error occurs, the absolute assemble list file cannot be output.
- 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]

- If the primary name is different between an assemble list file (k0rmain.prn) and an object module file (sample.rel), describe as follows so as to specify the input of an object module file (sample.rel).

```
C>lc78k0r k0rmain.prn -lsample.rel
```

Load module file input specification

(1) -l

[Syntax]

```
-l[input-file-name]
```

- Default assumption

  -l*assemble-list-file-name*.lmf

[Function]

- The -l option 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 the -l option.

[Explanation]

- If a fatal error occurs, the absolute assemble list file cannot be output.
- 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]

- If the primary name is different between an assemble list file (k0rmain.prn) and a load module file (sample.lmf), describe as follows so as to specify the input of a load module file (sample.lmf).

```
C>lc78k0r k0rmain.prn -lsample.lmf
```

Absolute assemble list file output specification

(1) -o

[Syntax]

```
-o[output-file-name]
```

- Default assumption

  -o*assemble-list-file-name*.p

[Function]

- The -o option specifies the output of an absolute assemble list file.
  The -o option also specifies the output destination and output file name.

[Application]

- Use the -o option to change the output destination and output file name of the absolute assemble list file.

[Explanation]

- 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 the same device is specified for the file name as for the error file, an abort error occurs.
- If the output file name is omitted when the -o option is specified, the absolute assemble list file name will become "assemble-list-file-name.p".
- 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.
- If the drive name is omitted when the -o option is specified, the absolute assemble list file will be output to the current drive.

[Example of use]

- To create an absolute assemble list file (sample.p), describe as:

```
C>lc78k0r k0rmain.prn -osample.p -lk0r.lmf
```

Error list file output specification

(1) -e/-ne

[Syntax]

```
-e[output-file-name]
-ne
```

- Default assumption

  -ne

[Function]

- Specify the -e option to specify the output of an error list file.
  This option also specifies the output destination and output file name.
- The -ne option makes the -e option unavailable.

[Application]

- Specify the -e option to save error messages in a file.

[Explanation]

- The file name of the error list file can be specified as a disk-type file name or as a device-type file name.
- If the device specified in the file name is the same as that specified in the absolute assemble list file, an abort error occurs.
- If the -e option is specified and the output file name is omitted, the error list file name will be "assemble-list-file-name.elv".
- 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.
- If the drive name is omitted when the -e option is specified, the error list file will be output to the current drive.
- If both options -e and -ne are specified at the same time, the option specified last takes precedence.

[Example of use]

- To create an error list file (sample.elv), describe as:

```
C>lc78k0r k0rmain.prn -esample.elv
```

<Contents of sample.elv>

```
RA78K0R warning W6701 : Load module file is older than object module file
'k0rmain.lmf, k0rmain.rel'
Pass1 : start

RA78K0R warning W6702 : Load module file is older than assemble module file
'k0rmain.lmf, k0rmain.prn'

Pass2 : start
```

Parameter file specification

(1) -f

[Syntax]

```
-ffile-name
```

- Default assumption

  Options and input file names can only be input from the command line.

[Function]

- The -f option specifies input of options and the input file name from a specified file.

[Application]

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

[Explanation]

- Only a disk-type file name can be specified as file-name. If a device-type file name is specified, an abort error occurs.
- If the file name is omitted, an abort error occurs.
- If only the primary name of the file name is specified, the list converter will assign the file type ".plv" and open the file.
- Nesting of parameter files is not permitted. If the -f option is specified within a parameter file, an abort error occurs.
- The number of characters that can be written within a parameter file is unlimited.
- Separate options or input file names with a blank space, a tab or a line feed code (LF).
- 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.
- The expanded options specified last will take precedence.
- If the -f option is specified two or more times, an abort error occurs.
- All characters entered after ";" or "#" and before a line feed code (LF) or "EOF" will be interpreted as comments.

[Example of use]

- Start up list converter using a parameter file (k0r.plv).

  <Contents of the k0r.plk>

```
: parameter file
k0rmain -lk0r.lmf
-ek0r.elv
```

Enter the following on the command line.

```
C>lc78k0r -fk0r.plv
```

Help specification

(1) --

[Syntax]

```
--
```

- Default assumption

  No display

[Function]

- The -- 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 the -- option is specified, all other options are unavailable.

  Caution This option cannot be specified on PM+.

  To reference PM+ help, click the [Help] button in the [List Converter Options] dialog box.

[Example of use]

- To output a help message on the display, describe as:

```
C>lc78k0r --
```

```
List Conversion Program for RA78K0R Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

usage : LC78K0R [option [...]] input-file [option [...]]
The option is as follows ([] means omissible).
-R [file] : Specify object module file.
-L [file] : Specify load module file.
-O [file] : Specify output list file (absolute assemble list file).
-Ffile    : Input option or input-file name from specified file.
-E [file] : Create error list file.
--        : Show this message.
```

8.5 Option Settings in PM+

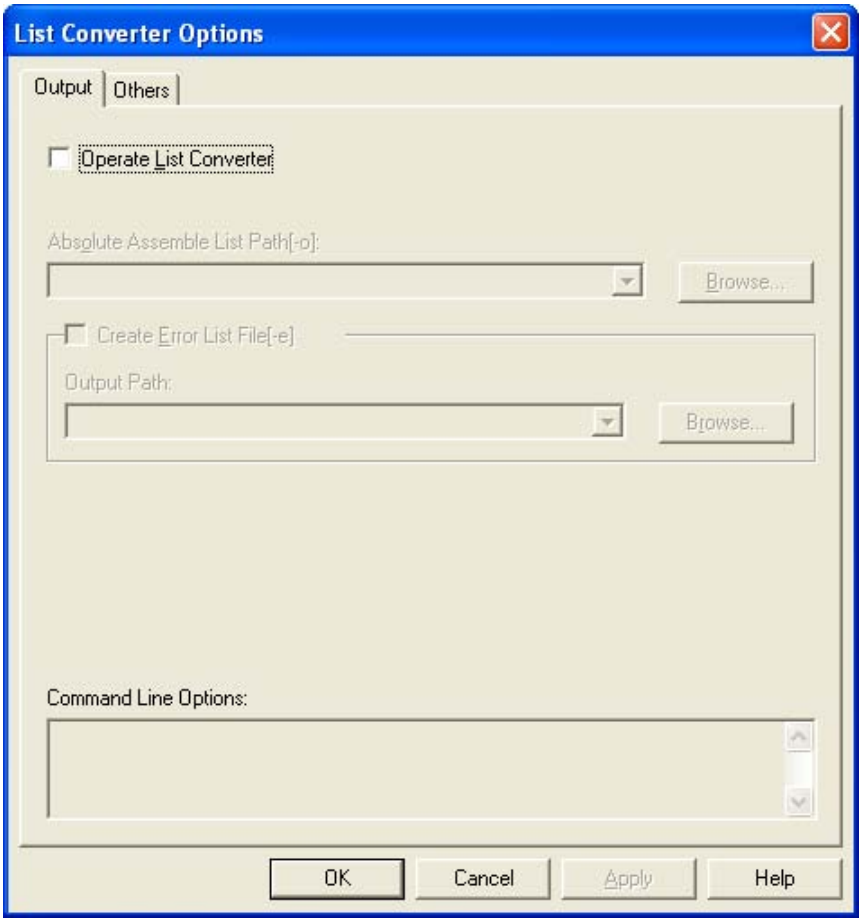
This section describes the method for setting list converter options from PM+.

8.5.1 Option setting method

The [List Converter Options] dialog box is opened if [List converter Options] is selected from the [Tools] menu of PM+ or if the [LC] button on the toolbar is clicked.

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

Figure 8-2 [List Converter Options] Dialog Box

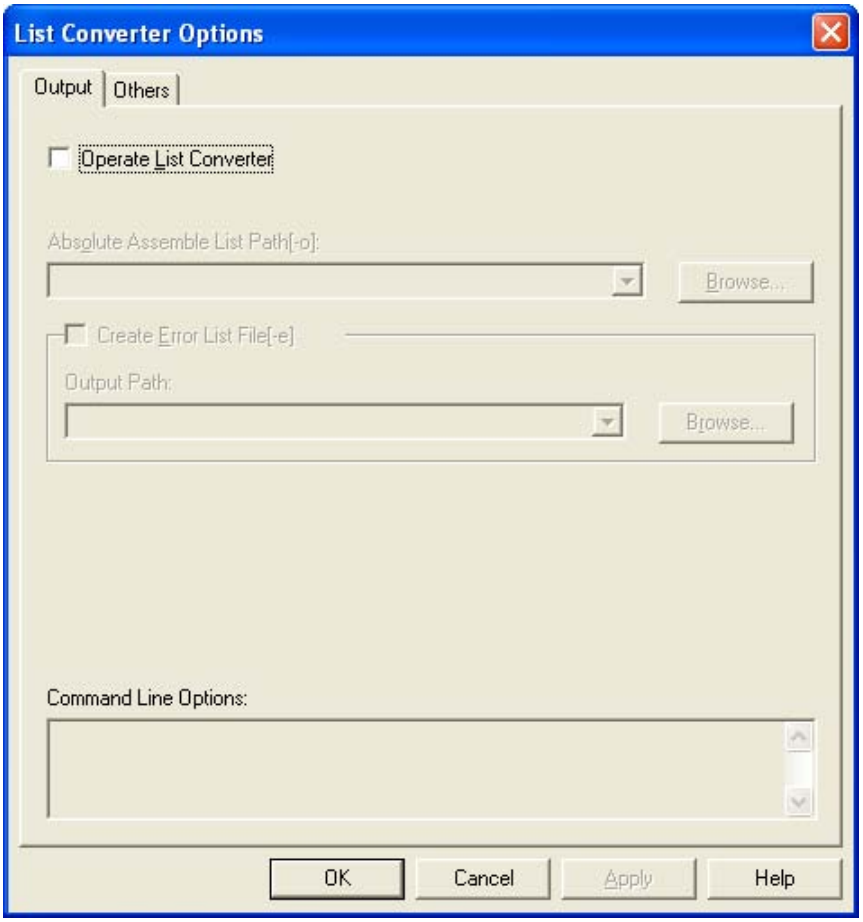


8.5.2 Explanation of dialog box

The various tabs in the [List Converter Options] dialog box are described below.

(1) [Output] tab

Figure 8-3 [List Converter Options] Dialog Box (When [Output] Tab Is Selected)



- Operate List Converter

  Select this option to start the list converter.

- Absolute Assemble List Path[-o]

  Specify the path of the absolute assemble list by using the [Browse...] button or directly inputting a path.

  The number of characters that can be input is up to 259.

- Create Error List File[-e]

  Select this option to output an error list file.

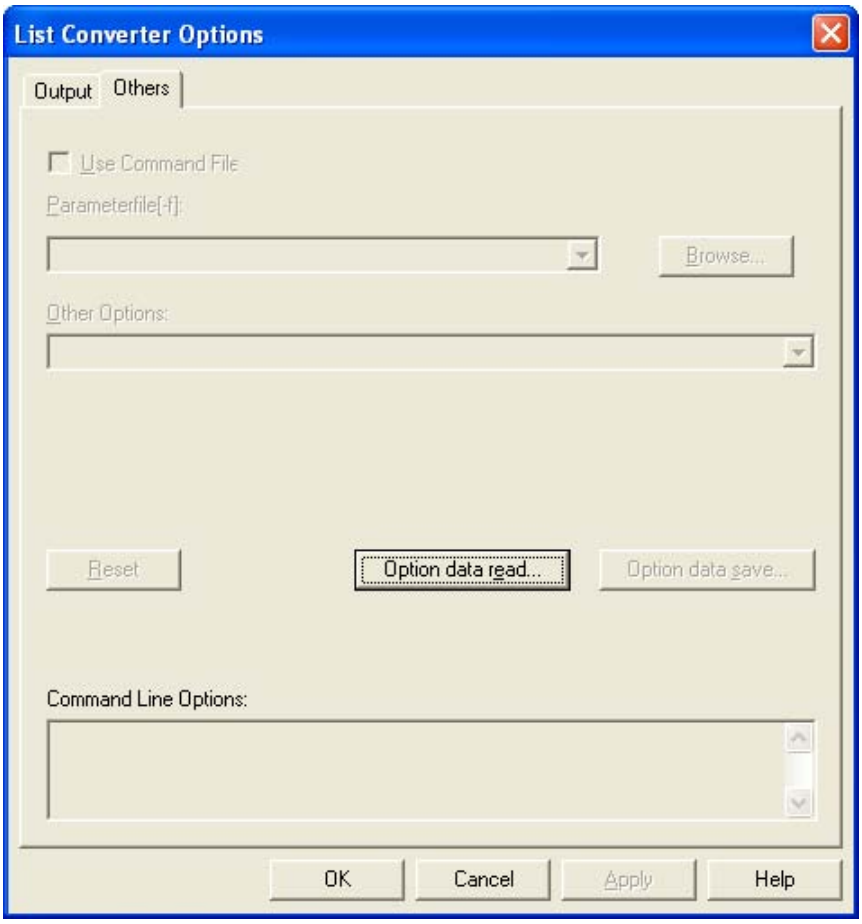
Output Path	Specify the path of the error list file by using the [Browse...] button or directly inputting a path. The number of characters that can be input is up to 259.

- Command Line Options

  This edit box is read-only. The currently set option character string is displayed.

(2) [Others] tab

Figure 8-4 [List Converter Options] Dialog Box (When [Others] Tab Is Selected)



- Use Command File

  Select this option to create a command file.

- Parameterfile[-f]

  Specify the file to be input as a user-defined parameter file by using the [Browse...] button or directly inputting a file name.

- Other Options

  To specify an option other than those that can be set in this dialog box, enter the option in the input box.

  Caution The help specification (--) option cannot be specified on PM+.

- [Reset] button

  Resets the input contents.

- [Option data read...] button

  Opens the [Read Option Data] dialog box and after the option data file has been specified, reads this file.

- [Option data save...] button

  Opens the [Save Option Data] 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 9 PROGRAM OUTPUT LIST

The following is an explanation of the formats and other information for the lists output by each program.

- Lists Output by Assembler
- Lists Output by Linker
- List Output by Object Converter
- List Output by Librarian
- Lists Output by List Converter

9.1 Lists Output by Assembler

The assembler outputs the following lists.

Table 9-1 Lists Output by Assembler

Output List File Name	Output List Name
Assemble list file	Assemble list file headers
	Assemble list
	Symbol list
	Cross-reference list
Error list file	Error list

9.1.1 Assemble list file headers

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

[Output format]

```
78K0R Assembler (1)Vx.xx (2)SAMPLE_TITLE    Date:(3)xx xxx xxxx Page:(4)1
(5)SAMPLE_SUBTITLE
Command:    (6) k0rmain.asm -cf1166a0
Para-file:  (7)  -ks -kx
In-fine:    (8)  k0rmain.asm
Obj-file:   (9)  k0rmain.rel
Prn-file:   (10) k0rmain.prn
```

Item	Details
(1)	Assembler version no.
(2)	Title character string Character string specified by the -lh option or TITLE control instruction
(3)	Date of assemble list creation
(4)	Page no.
(5)	Subtitle character string Character string specified by SUBTITLE control instruction
(6)	Command-line image
(7)	Contents of parameter file
(8)	Input source module file name
(9)	Output object module file name
(10)	Assemble list file name

9.1.2 Assemble list

The assemble list outputs the results of the assemble with error messages (if errors occur).

[Output format]

```
        Assemble list

(1)ALNO (2)STNO (6)ADRS (8)OBJECT (3)M (4)I (5)SOURCE STATEMENT
    1        1
    2        2                                      NAME  SAMPM
    :
   31       31       0000A   RFD0000                CALL  !CONVAH
                                                ; convert ASCII <- HEX
   32       32                                  ; output BC-register <- ASCII code
   33       33       0000D   00000000                MOV   DE , #LOWW ( STASC )
                                                ; set DE <- store ASCII code table
                     00011   00
(7) ** ERROR E2202, STNO 33 ( 33 ) Illegal operand
   34       34       00012   63                     MOV   A , B
   35       35       00013   99                     MOV   [ DE ] , A
    :
Segment informations :

(9)ADRS (10)LEN  (11) NAME

   FFE20    00003H   DATA
   00000    00002H   CODE
   00000    00019H   ?CSEG

Target chip : (12) uPD78xxx
Device file : (13) Vx.xx
Assembly complete, (14)1 error(s) and (15)0 warning(s) found. ((16)33)
```

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 (4 or 5 digits)
(7)	Error 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 (5 digits)

Item	Details
(10)	Segment size (8 digits)
(11)	Segment name
(12)	RA78K0R target device
(13)	Device file version no.
(14)	Number of fatal errors
(15)	Number of warnings
(16)	Final error line

9.1.3 Symbol list

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

[Output format]

```
Symbol Table List

(1)VALUE (2)ATTR  (3)RTYP  (4)NAME     (1)VALUE (2)ATTR  (3)RTYP (4)NAME
           CSEG             ?CSEG                CSEG             CODE
   ----H            EXT     CONVAH               DSEG             DATA
   FFE20H  ADDR             HDTSA          0H    ADDR     PUB     MAIN
           MOD              SAMPM          0H    ADDR     PUB     START
   FFE21H  ADDR             STASC       -----H           EXT     _@STBEG
```

Item	Details
(1)	Symbol value (8 digits)
(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 BIT: BIT attribute symbol (addr.bit) SABIT: BIT attribute symbol (saddr.bit) SFBIT: BIT attribute symbol (sfr.bit) RBIT: BIT attribute symbol (A.bit, X.bit, PSW.bit) SFR: Names defined as SFRs by EQU directive SFRP: Names defined as SFRPs by EQU directive 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 reference symbol declared by PUBLIC Blank: Local symbol, segment name, macro name, module name *****: Undefined symbol
(4)	Defined symbol name

9.1.4 Cross-reference list

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

[Output format]

```
        Cross-Reference List

(1)NAME   (2)VALUE  (3)R     (4)ATTR   (5)RTYP   (6)SEGNAME    (7)XREFS
   ?CSEG                       CSEG                ?CSEG          22#
   CODE                        CSEG                CODE           19#
   CONVAH    -----H    E                   EXT                    12@    31
   DATA                        DSEG                DATA           15#
   HDTSA     FFE20H            ADDR                DATA           16#    28     29
   MAIN          0H            ADDR      PUB       CODE           11@    20#
   SAMPM                       MOD                                 2#
   START         0H    R       ADDR      PUB       ?CSEG          11@    20     23#
   STASC     FFE21H            ADDR                DATA           17#    33
   _@STBEG   -----H    E                   EXT                    13@    26
```

Item	Details
(1)	Defined symbol name
(2)	Symbol value (8 digits)
(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 BIT: BIT attribute symbol (addr.bit) SABIT: BIT attribute symbol (saddr.bit) SFBIT: BIT attribute symbol (sfr.bit) RBIT: BIT attribute symbol (A.bit, X.bit) SFR: Names defined as SFRs by EQU directive SFRP: Names defined as SFRPs by EQU directive 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 reference symbol declared by PUBLIC Blank: Local symbol, segment name, macro name, module name *****: Undefined symbol
(6)	Defined symbol name

Item	Details
(7)	Definition/reference line no. Definition line: xxxxx# Reference line: xxxxx Δ (Δ = 1 blank) EXTRN declaration, EXTBIT declaration, PUBLIC declaration: xxxxx@

9.1.5 Error list

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

[Output format]

```
PASS1 Start
(1)ERROR.ASM ( (2)26 ) : RA78K0R (3)error (4)E2202 : (5)Illegal operand
(1)ERROR.ASM ( (2)32 ) : RA78K0R (3)error (4)E2202 : (5)Illegal operand
PASS2 Start
(1)ERROR.ASM ( (2)26 ) : RA78K0R (3)error (4)E2202 : (5)Illegal operand
(1)ERROR.ASM ( (2)29 ) : RA78K0R (3)error (4)E2407 : (5)Undefined symbol
reference 'DTSA'
(1)ERROR.ASM ( (2)29 ) : RA78K0R (3)error (4)E2303 : (5)Illegal expression
(1)ERROR.ASM ( (2)32 ) : RA78K0R (3)error (4)E2202 : (5)Illegal operand
(1)ERROR.ASM ( (2)37 ) : RA78K0R (3)error (4)E2407 : (5)Undefined symbol
reference 'F'
(1)ERROR.ASM ( (2)37 ) : RA78K0R (3)error (4)E2303 : (5)Illegal expression
```

Item	Details
(1)	Name of source module file in which error occurred
(2)	Line on which error occurred
(3)	Type of error
(4)	Error no.
(5)	Error message

Caution The file name and the line where the error occurred may not be displayed.

9.2 Lists Output by Linker

The linker outputs the following lists.

Table 9-2 Lists Output by Linker

Output List File Name	Output List Name
Link list file	Link list file headers
	Map list
	Public symbol list
	Local symbol list
Error list file	Error list

9.2.1 Link list file headers

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

[Output format]

```
78K0R Linker (1)Vx.xx                Date : (2)xx xxx xxxx Page : (3)1

Command :       (4) k0rmain.rel k0rsub.rel -s -ok0r.map -dk0r.dr
Para-file :     (5)
Out-file :      (6) k0rmain.lmf
Map-File :      (7) k0r.map
Direc-File :    (8) k0r.dr
Directive :     (9) MEMORY ROM  : ( 0H , 0ED800H )
                (9) MEMORY RAM1 : ( 0FCF00H , 1100H )
                (9) MEMORY RAM  : ( 0FE000H , 1F00H )

*** Link information ***

(10)   6 output segment(s)
(11) 9DH byte(s) real data
(12)  40 symbol(s) defined
```

Item	Details
(1)	Linker version no.
(2)	Date of link list file creation
(3)	Page no. (4 or 5 digits)
(4)	Command-line image (4 or 5 digits)
(5)	Contents of parameter file
(6)	Output load module file name
(7)	Link list file name
(8)	Directive file name
(9)	Directive file contents
(10)	Number of segments output to load module file
(11)	Size of data output to load module file
(12)	Number of symbols output to load module file

9.2.2 Map list

The map list outputs data on the location of segments.

[Output format]

```
*** Memory map ***

(1)SPACE = REGULAR

  MEMORY = (2)ROM
  BASE ADDRESS = (3)00000H    SIZE = (4)ED800H
      (6)OUTPUT   (7)INPUT    (8)INPUT  (9)BASE     (10)SIZE
         SEGMENT     SEGMENT     MODULE    ADDRESS
         CODE                            00000H      00002H  (11)CSEG  AT
                     CODE        SAMPM     00000H      00002H
(5)* gap *                               00002H      000BEH
         ?CSEGOB0                        000C0H      00004H  (11)CSEG  OPT_BYTE
         ?CSEG                           000C4H      00059H  (11)CSEG
                     ?CSEG       SAMPM     000C4H      00017H
                     ?CSEG       SAMPS     000DBH      00042H
(5)* gap *                               0011DH      ED6E3H

  MEMORY = RAM1
  BASE ADDRESS = (3)FCF00H    SIZE = (4)01100H
      (6)OUTPUT   (7)INPUT    (8)INPUT  (9)BASE     (10)SIZE
         SEGMENT     SEGMENT     MODULE    ADDRESS
(5)* gap *                               FCF00H      01100H

  MEMORY = RAM
  BASE ADDRESS = (3)FE000H    SIZE = (4)01F00H
      (6)OUTPUT   (7)INPUT    (8)INPUT  (9)BASE     (10)SIZE
         SEGMENT     SEGMENT     MODULE    ADDRESS
(5)* gap *                               FE000H      01E20H
         DATA                            FFE20H      00003H  (11)DSEG  AT
                     DATA        SAMPM     FFE20H      00003H
(5)* gap *                               FFE23H      000DDH

 Target chip : (12)uPD78xxx
 Device File : (13)Vx.xx
```

Item	Details
(1)	Memory space name
(2)	Memory area name
(3)	Memory area start address (5 digits)
(4)	Memory area size (5 digits)
(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

Item	Details
(9)	Segment start address (8 digits)
(10)	Output/input segment size (8 digits)
(11)	Segment type and reallocation attributes
(12)	Target device for this assemble
(13)	Device file version no.

9.2.3 Public symbol list

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

[Output format]

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

(1)MODULE    (2)ATTR      (3)VALUE     (4)NAME

   SAMPM
                ADDR         00000H       MAIN
                ADDR         000D2H       START
   SAMPS
                ADDR         000E9H       CONVAH

                NUM          FFE20H       _@STBEG

                NUM          FE000H       _@STEND
```

Item	Details
(1)	Name of module in which public symbols are defined
(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 BIT: BIT attribute symbol (addr.bit) SABIT: BIT attribute symbol (saddr.bit) SFBIT: BIT attribute symbol (sfr.bit) RBIT: BIT attribute symbol (A.bit, X.bit PSW.bit) SFR: Names defined as SFRs by EQU directive SFRP: Names defined as SFRPs by EQU directive Blank: External reference symbol declared by EXTRN or EXTBIT *****: Undefined symbol
(3)	Symbol value (8 digits)
(4)	Public symbol name

9.2.4 Local symbol list

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

[Output format]

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

(1)MODULE    (2)ATTR      (3)VALUE     (4)NAME

   SAMPM
                MOD                       SAMPM
                DSEG                      DATA
                ADDR         FFE20H       HDTSA
                ADDR         FFE21H       STASC
                CSEG                      CODE
                CSEG                      ?CSEG
   SAMPS
                MOD                       SAMPS
                CSEG                      ?CSEG
                ADDR         00114H       SASC
                ADDR         0011AH       SASC1
```

Item	Details
(1)	Name of module in which local symbols are defined
(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 BIT: BIT attribute symbol (addr.bit) SABIT: BIT attribute symbol (saddr.bit) SFBIT: BIT attribute symbol (sfr.bit) RBIT: BIT attribute symbol (A.bit, X.bit, PSW.bit) SFR: Names defined as SFRs by EQU directive SFRP: Names defined as SFRPs by EQU directive Blank: External reference symbol declared by EXTRN or EXTBIT *****: Undefined symbol
(3)	Symbol value (8 digits)
(4)	Local symbol name

9.2.5 Error list

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

[Output format]

```
RA78K0R (1) error (2) E3405 : (3) Undefined symbol 'CONVAH' in file
'k0rmain.rel'
```

Item	Details
(1)	Type of error
(2)	Error no.
(3)	Error message

9.3 List Output by Object Converter

The object converter outputs the following list.

Table 9-3 Explanation of Object Converter Output Items

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

9.3.1 Error list

Error messages output when the object converter is started up are stored in an error list.

[Output format]

Same as error list output by the linker.

9.4 List Output by Librarian

The librarian outputs the following list.

Table 9-4 List Output by Librarian

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

9.4.1 Library data output list

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

[Output format]

```
78K0R librarian (1)Vx.xx                DATE:(2)xx xxx xx    PAGE:(3)1

LIB-FILE NAME : (4)k0r.lib        ( (5)xx xxx xx )

(6)0001   (7)k0rmain.rel  ( (8)xx xxx xx )

  (9)MAIN                         (9)START

  NUMBER OF PUBLIC SYMBOLS : (10)2

(6)0002   (7)k0rsub.rel   ( (8)xx xxx xx )

  (9)CONVAH

  NUMBER OF PUBLIC SYMBOLS : (10)1
```

Item	Details
(1)	Librarian version no.
(2)	Date of list creation
(3)	Number of pages
(4)	Library file name
(5)	Date of library file creation
(6)	Module serial no. (beginning from 0001)
(7)	Module name
(8)	Date of module creation
(9)	Public symbol name
(10)	Number of public symbols defined in module

9.5 Lists Output by List Converter

The list converter outputs the following lists.

Table 9-5 Lists Output by List Converter

Output List File Name	Output List Name
Absolute assemble list file	Absolute assemble list
Error list file	Error list

9.5.1 Absolute assemble list

The absolute assemble list embeds absolute values in the assemble list and outputs the list.

[Output format]

Same as for the assemble list output by the assembler.

9.5.2 Error list

Error messages output when the list converter is started up are stored in an error list.

[Output format]

Same as for the error list output by the assembler.

CHAPTER 10 EFFICIENT USE OF RA78K0R

This chapter introduces some methods that will help you to use the RA78K0R efficiently.

10.1 Improving Operating Efficiency (EXIT Status Function)

When any of the programs of the RA78K0R 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:

Table 10-1 EXIT Statuses

Processing	EXIT Statuses
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)>

```
ra78K0R -cf1166a0 k0rmain.-g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
ra78K0R -cf1166a0 k0rsub.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
lk78K0R k0rmain.rel k0rsub.rel -ok0r.lmf -g
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
oc78K0R k0r.lmf
echo off
IF ERRORLEVEL 1 GOTO ERR
GOTO EXIT
:ERR
echo  Error occurred
: EXIT
```

To perform processing using batch file (ra.bat), describe as:

```
C>ra.bat
```

10.2 Preparing Development Environment (Environmental Variables)

The RA78K0R supports the following environmental variables for preparing the software development environment.

Table 10-2 Environmental Variables

Environmental Variables	Explaination
PATH	Search path for execution format
INC78K0R	Search path for include file (assembler)
LIB78K0R	Search path for library file (linker only)
TMP	Path for creating temporary files
LANG78K	Kanji (2-byte character) type specification

[Example of use]

<Contents of autoexec.bat>

```
; autoexec.bat
Verify on
break on
PATH C:\bin;C:\bat;C:\ra78k0r;      <-- (1)
SET INC78K0R=C:\ra78k0r\include     <-- (2)
SET LIB78K0R=C:\ra78k0r\lib         <-- (3)
SET TMP=C:\tmp                      <-- (4)
SET LANG78K=SJIS                    <-- (5)
```

(1) Because this path is specified, execution format files are retrieved from folders in the order C:\bin, C:\bat, C:\ra78k0r.

(2) The assembler retrieves include files from the folder C:\ra78k0r\include.

(3) The linker retrieves library files from C:\ra78k0r\lib.

(4) Each program creates a temporary file in C:\tmp.

(5) Kanji (2-byte character) in the comment statement is interpreted as shift JIS code.

10.3 Interrupting Program Execution

Execution of each program can be interrupted by entering CTRL+C from the keyboard.

If "break on" is specified during execution of 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.

10.4 Making Assemble List Easy to Read

Display a title in the header of an assemble list using the -lh option 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 RA78K0R Assembler Package Language User's Manual.

[Example of use]

To print a title in the header of an assemble list file (k0rmain.prn).

```
C>ra78k0r -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
```

<Contents of k0rmain.prn>

```
78K0R Assembler Vx.xx RA78K0R_MAINROUTINE       Date:xx xxx xx  Page:1
                            |
                          Title

Command : -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
Para-file :
In-file : k0rmain.asm
Obj-file : k0rmain.rel
Prn-file : k0rmain.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                       ; *
      :
```

10.5 Reducing Program Startup Time

10.5.1 Specifying control instruction in the source program

Control instructions which have the same functions as the options normally specified in assembler startup can be specified in advance in the source program. This eliminates the need to specify options every time the assembler is started up.

[Example of use]

```
$ PROCESSOR ( f1166a0 )     ; Control instructions
$ XREF                      ; Control instructions

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

10.5.2 Using PM+

The options of each program of the RA78K0R are automatically saved to the project file (.PRJ) in PM+. The saved options will be used for subsequent builds (MAKE). It is not therefore necessary to specify the options each time.

10.5.3 Creating parameter files and subcommand files

When executing any of the RA78K0R'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 k0rmain.pra>

```
; parameter file
k0rmain.asm -osample.rel -g
-psample.prn
```

Enter the following on the command line.

```
C>ra78k0r -fk0rmain.pra
```

[Example of use 2]

Create a parameter file and perform assembly.

<Contents of parameter file k0r.slb>

```
;
; library creation command
;
create k0r.lib
;
add k0r.lib k0rmain.rel &
k0rsub.rel
;
exit
```

Enter the following on the command line.

```
C>lb78k0r <k0r.slb
```

10.6 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 RA78K0R 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 11 ERROR MESSAGES

This chapter explains the causes of error messages output by the RA78K0R's programs (assembler, linker, object converter and librarian), and the action to be taken by the user.

11.1 Overview of Error Messages

Error messages output by the RA78K0R are divided into the following 4 levels.

(1) Abort error (F*xxxx*)

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 error (E*xxxx*)

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) Internal error (C*xxxx*)

Processing is immediately terminated (aborted) because an internal error has occurred.

(4) Warning (W*xxxx*)

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.

The error messages of the assembler package are classified as follows.

Type	Explanation
Fn0*xx*	Command line analysis error
Fn9*xx*	File or system error
Fn1*xx*	Other abort error
Cn*xxx*	Internal error
En2*xx*	Statement specification error
En3*xx*	Expression error
En4*xx*	Symbol error
En5*xx*	Segment error
En6*xx*	Control instruction or macro error
Wn*xxx*	Any type of warning

Remark n = 2 to 6

2: Assembler

3: Linker

4: Object Converter

5: Librarian

6: List Converter

11.2 Assembler Error Messages

Table 11-1 Assembler Error Messages

Error Number	Error Message	
F2001	Message	Missing input file
	Cause	An input file has not been specified.
	Action by User	Specify an input file.
F2002	Message	Too many input files
	Cause	Two or more input files have been specified.
	Action by User	Specify only one input file.
F2004	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.
F2005	Message	Illegal file specification '*file name*'
	Cause	An illegal file has been specified.
	Action by User	Specify a legal file.
F2006	Message	File not found '*file name*'
	Cause	The specified file does not exist.
	Action by User	Specify an existent file.
F2008	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.
F2009	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.
F2010	Message	Directory not found '*file name*'
	Cause	A non-existent drive and/or folder has been included in the output file name.
	Action by User	Specify an existent drive and/or folder.
F2011	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.
F2012	Message	Missing parameter '*option*'
	Cause	A necessary parameter has not been specified.
	Action by User	Specify the parameter.

Error Number	Error Message	
F2013	Message	Parameter not needed '*option*'
	Cause	An unnecessary parameter has been specified.
	Action by User	Delete the unnecessary parameter.
F2014	Message	Out of range '*option*'
	Cause	The specified numerical value is outside the range.
	Action by User	Specify a correct numerical value.
F2015	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.
F2016	Message	Illegal parameter '*option*'
	Cause	The syntax of the parameter is incorrect.
	Action by User	Specify a correct parameter.
F2017	Message	Too many parameters '*option*'
	Cause	The total number of parameters exceeds the limit.
	Action by User	Specify parameters within the number limit.
F2018	Message	Option is not recognized '*option*'
	Cause	The option name is incorrect.
	Action by User	Specify a correct option name.
F2019	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.
F2020	Message	Parameter file read error '*file name*'
	Cause	The parameter file cannot be read.
	Action by User	Specify a correct parameter file.
F2021	Message	Memory allocation failed
	Cause	There is insufficient memory.
	Action by User	Secure the necessary memory.
F2101	Message	Source file size 0 '*file name*'
	Cause	A source module with file size 0 has been input.
F2102	Message	Illegal processor type specified
	Cause	A mistake was made in the specification of the target device.
F2103	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.

Error Number	Error Message	
F2104	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.
F2105	Message	Duplicate PROCESSOR control
	Cause	A PROCESSOR control instruction is written more than once in a source module header.
F2106	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.
F2107	Message	Default segment ?CSEG is already used
	Cause	Attempted to define an undefined segment with a default segment.
F2108	Message	Symbol table overflow '*symbol name*'
	Cause	The number of symbols exceeds the definable limit.
F2109	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.
F2110	Message	String table overflow
	Cause	Limits of the string table are exceeded.
	Action by User	Reduce number of symbols to 9 characters or less.
F2111	Message	Object code more than 128bytes
	Cause	Object code exceeds 128 bytes per line in a source statement.
F2112	Message	No processor specified
	Cause	Target device is not specified in the command line or in the source module file.
F2114	Message	Local symbol name of asm statement must begin with '?L' in C source.
	Cause	A local symbol which begins with other than '?L' is described in the #asm of the C source.
F2115	Message	Too long source line
	Cause	The limit on the length of one line (2048 characters) has been exceeded.
E2201	Message	Syntax error
	Cause	An incorrect statement format was used.
E2202	Message	Illegal operand
	Cause	The specified operand is illegal.
E2203	Message	Illegal register
	Cause	A register that cannot be specified was specified.
E2204	Message	Illegal character
	Cause	An illegal character is specified in the source module.

Error Number	Error Message	
E2205	Message	Unexpected LF in string
	Cause	A carriage return code appears in a character string before the string is closed.
E2206	Message	Unexpected EOF in string
	Cause	An end-of-file code appears in a character string before the string is closed.
E2207	Message	Unexpected null code in string
	Cause	A null code (00H) is written in a character string.
E2209	Message	Too many line number
	Cause	The number of lines described in one file has exceeded the limit.
E2301	Message	Too complex expression
	Cause	Expression is too complex.
E2302	Message	Absolute expression expected
	Cause	A relocatable expression is specified.
E2303	Message	Illegal expression
	Cause	Incorrect format for expression is used.
E2304	Message	Illegal symbol in expression '*file name*'
	Cause	An unusable symbol is used in an expression.
E2305	Message	Too long string as constant
	Cause	Limit on string constant length (4 characters) is exceeded.
E2306	Message	Illegal number
	Cause	Incorrect numerical value is specified.
E2307	Message	Division by zero
	Cause	A value is divided by zero.
E2308	Message	Too large integer
	Cause	The value of a constant exceeds 16 bits.
E2309	Message	Illegal bit value
	Cause	Incorrect bit value is specified.
E2310	Message	Bit value out of range
	Cause	Bit value exceeds the range 0 to 7.
E2311	Message	Operand out of range (n)
	Cause	Specified value exceeds the range n (0 to 7).
E2312	Message	Operand out of range (byte)
	Cause	Operand is outside the specifiable range (00H to 0FFH) for byte.
	Action by User	Specify the operand in a specifiable range.
E2313	Message	Operand out of range (addr5)
	Cause	Operand is outside the specifiable range (80H to BFH) for addr5.

Error Number	Error Message	
E2315	Message	Operand out of range (saddr)
	Cause	Operand is outside the specifiable range (0FFE20H to 0FFF1FH) for saddr.
E2316	Message	Operand out of range ($!addr20)
	Cause	The specified operand is out of the specifiable range for addr16 (00000H to 0FFFFFH), or out of the range "-32768 to +32,767" as a result of calculation of its relative displacement from the address next to the branch directive.
E2317	Message	Even expression expected
	Cause	Odd-number address is specified for word access.
E2318	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 SFRP directive.
E2319	Message	Operand out of range (word)
	Cause	Operand is outside the specifiable range (0000H to 0FFFFH) for word.
	Action by User	Specify the operand in a specifiable range.
E2320	Message	Operand out of range (20bit)
	Cause	Operand is outside the specifiable range (00000H to 0FFFFFH) for 20bit.
	Action by User	Specify the operand in a specifiable range.
E2321	Message	Operand out of range (addr20)
	Cause	Operand is outside the specifiable range (0000H to 0FFFFFH) for addr20.
	Action by User	Specify the operand in a specifiable range.
E2322	Message	Illegal operand, 2ndSFR is used as addr16
	Cause	The specified operand is illegal. 2ndSFR is used as addr16.
	Action by User	Modify the operand in the same manner as when addr16 is described.
E2323	Message	Illegal operand, 2ndSFR.bit is used as addr16.bit
	Cause	The specified operand is illegal. 2ndSFR.bit is used as addr16.bit.
	Action by User	Modify the operand in the same manner as when addr16 is described.
E2324	Message	Illegal operand, SFR can't be used as addr16
	Cause	The specified operand is illegal. SFR cannot be used as addr16.
	Action by User	Describe the operand with SFR.
E2325	Message	Illegal operand, SFR.bit can't be used as addr16.bit
	Cause	The specified operand is illegal. SFR.bit cannot be used as addr16.bit.
	Action by User	Describe the operand with SFR.bit.

Error Number	Error Message	
E2326	Message	Illegal SFR access in operand
	Cause	An SFR symbol that cannot access an operand is described.
E2327	Message	Operand out of range (addr20)
	Cause	The specified operand is out of the specifiable range for addr20 (00000H to 0FFFFFH), or out of the range "-128 to +127" as a result of calculation of its relative displacement from the address next to the branch directive.
	Action by User	Specify the operand in a specifiable range.
E2328	Message	Operand out of range (n)
	Cause	The specified operand is out of the range of n (1 to 7).
	Action by User	Specify the operand in a specifiable range.
E2329	Message	Operand out of range (n)
	Cause	The specified operand is out of the range of n (1 to 15).
	Action by User	Specify the operand in a specifiable range.
E2330	Message	Operand out of range (addr16 / BR or CALL)
	Cause	Operand is outside the specifiable range (0H to FFFFH) for addr16.
	Action by User	Specify the operand in a specifiable range.
E2331	Message	Operand out of range (addr16 / NUMBER)
	Cause	The specified operand is out of the specifiable range (0H to FFFFH) for addr16 (numeric constant, and NUMBER attribute symbol).
	Action by User	Specify the operand in a specifiable range.
E2332	Message	Operand out of range (!addr16 / ADDRESS)
	Cause	The specified operand is out of the specifiable range for addr16 (ADDRESS attribute symbol).
	Action by User	Change the operand to one of the following, within the specifiable area. (1) F0000H to FFFFFH (2) Area mirrored when MAA = 0, or area mirrored when MAA = 1 For details on the mirror area, refer to the user's manual of the device.
E2333	Message	Operand out of range (ES:!addr16 / ADDRESS)
	Cause	The specified operand is out of the specifiable range (0H to FFFFFH) for ES:!addr16 (ADDRESS attribute symbol).
	Action by User	Specify the operand in a specifiable range.
E2334	Message	Operand out of range (!addr16.bit / ADDRESS)
	Cause	Operand is outside the specifiable range for !addr16.bit.
	Action by User	Change the operand to one of the following, within the specifiable area. (1) F0000H to FFFFFH (2) Area mirrored when MAA = 0, or area mirrored when MAA = 1 For details on the mirror area, refer to the user's manual of the device.

Error Number	Error Message	
E2335	Message	Operand out of range (ES:!addr16.bit / ADDRESS)
	Cause	The specified operand is out of the specifiable range (0H to FFFFFH) for ES:!addr16.bit.
	Action by User	Specify the operand in a specifiable range.
E2336	Message	Operand out of range (addr / BR or CALL)
	Cause	The specified operand is out of the specifiable range for operand "addr" of directives BR/CALL.
	Action by User	Specify the operand in a specifiable range.
E2337	Message	Illegal mnemonic, use another mnemonic or option -COMPATI
	Cause	A 78K0 instruction that cannot be used for the 78K0R is used.
	Action by User	Describe another instruction or use the -compati option.
E2338	Message	Operand out of range (EQU operand)
	Cause	The specified operand is out of the specifiable range (0H to 0FFFFFH) for operand "addr" of directives EQU.
	Action by User	Specify the operand in a specifiable range.
E2339	Message	Operand out of range (word / ADDRESS)
	Cause	The specified operand is out of the specifiable range for word (ADDRESS attribute symbol).
	Action by User	Change the operand to one of the following, within the specifiable area. (1) F0000H to FFFFFH (2) Area mirrored when MAA = 0, or area mirrored when MAA = 1 For details on the mirror area, refer to the user's manual of the device.
E2340	Message	Operand out of range (ES:word / ADDRESS)
	Cause	The specified operand is out of the specifiable range for ES:word (ADDRESS attribute symbol).
	Action by User	Specify the operand in a specifiable range.
E2341	Message	Illegal size for Option Bytes
	Cause	The segment that specifies the user option byte and on-chip debug option byte is not specified with 4 bytes.
	Action by User	Specify with 4 bytes the segment that specifies these bytes.
E2342	Message	Illegal value for Option Bytes
	Cause	An illegal value was located to the segment that specifies the user option byte and on-chip debug option byte.
	Action by User	Locate a correct value. For the locatable values, refer to the user's manual for the device.
E2343	Message	Illegal Option Bytes segment
	Cause	Two or more segments were specified as the segment that specifies the user option byte and on-chip debug option byte.
	Action by User	Specify only one segment to specify the user option byte and on-chip debug option byte.

Error Number	Error Message	
E2401	Message	Illegal symbol for PUBLIC '*symbol name*'
	Cause	This symbol cannot be declared PUBLIC.
E2402	Message	Illegal symbol for EXTRN/EXBIT '*symbol name*'
	Cause	This symbol cannot be declared EXTRN/EXTBIT.
E2403	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.
E2404	Message	Public symbol is undefined '*symbol name*'
	Cause	A symbol with a PUBLIC declaration is undefined.
E2405	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.
E2406	Message	Can't refer to forward bit symbol '*symbol name*'
	Cause	Specification refers forward to a bit symbol or refers to a bit symbol in an expression.
E2407	Message	Undefined symbol reference '*symbol name*'
	Cause	An undefined symbol is used.
E2408	Message	Multiple symbol definition '*symbol name*'
	Cause	Symbol name is defined more than once.
E2409	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.
E2410	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).
E2411	Message	This symbol is reserved '*symbol name*'
	Cause	The defined symbol is a reserved word.
E2502	Message	Illegal segment name
	Cause	Symbol is written with an illegal segment name.
E2503	Message	Different segment type '*symbol name*'
	Cause	Two or more segments are defined with the same name but types are different.
E2504	Message	Too many segment
	Cause	Number of segments defined exceeds limit (256).

Error Number	Error Message	
E2505	Message	Current segment is not exist
	Cause	An ENDS directive was written before a segment was created, or before a subsequent segment was created after the previous segment had ended.
E2506	Message	Can't describe DB, DW, DS, ORG, label in BSEG
	Cause	DB, DW, DS, ORG directives are defined in a bit segment.
E2507	Message	Can't describe opcodes outside CSEG
	Cause	Machine language instruction or BR directive is defined in something other than a code segment.
E2508	Message	Can't describe DBIT outside BSEG
	Cause	DBIT directive is defined in something other than a bit segment.
E2509	Message	Illegal address specified
	Cause	An address allocated to an absolute segment is outside the range for that segment.
E2510	Message	Location counter overflow
	Cause	Location counter is outside the range for a corresponding segment.
E2511	Message	Segment name expected
	Cause	Segment name is not specified for segment definition directive for reallocation attribute is AT.
E2512	Message	Segment size is odd numbers '*symbol name*'
	Cause	Size of reallocation attribute callt0 segment is described in an odd number.
E2515	Message	Security ID is not supported for this device
	Cause	A security ID cannot be used with the specified device.
E2516	Message	Option Bytes is not supported for this device
	Cause	Option byte cannot be used with the specified device.
E2601	Message	Nesting over of include
	Cause	Nesting of include file exceeds limit (2 levels).
E2602	Message	Must be specified switches
	Cause	Switch name not specified.
E2603	Message	Too many switches described
	Cause	Switch name exceeds limit (5 per module).
E2604	Message	Nesting over of IF-classes
	Cause	Nesting of IF/_IF clauses exceeds limit (8 levels).
E2605	Message	Needless ELSE statement exists
	Cause	An ELSE statement exists where it is not necessary.
E2606	Message	Needless ENDIF statement exists
	Cause	An ENDIF statement exists where it is not necessary.

Error Number	Error Message	
E2607	Message	Missing ELSE or ENDIF
	Cause	An ELSE or ENDIF statement required by IF/_IF clause is missing.
E2608	Message	Missing ENDIF
	Cause	An ENDIF statement required by IF/_IF clause is missing.
E2609	Message	Illegal ELSEIF statement
	Cause	An ELSEIF or _ELSEIF statement is written after an ELSE statement.
E2610	Message	Multiple symbol definition (MACRO) '*symbol name*'
	Cause	Symbol used to define a macro name is already defined.
E2611	Message	Illegal syntax of parameter
	Cause	Formal parameter of a macro is incorrect.
E2612	Message	Too many parameter
	Cause	Number of formal parameters for a macro definition exceeds limit (16).
E2613	Message	Same name parameter described '*symbol name*'
	Cause	Symbol is specified with same name as a formal parameter for a macro definition.
E2614	Message	Can't nest macro definition
	Cause	Macro definition cannot be nested in another macro definition.
E2615	Message	Illegal syntax of local symbol
	Cause	Specification of operand in a LOCAL directive is incorrect.
E2616	Message	Too many local symbols
	Cause	Number of local symbols that can be described in 1 macro body (64) is exceeded.
E2617	Message	Missing ENDM
	Cause	ENDM statement required by macro definition directive is missing.
E2618	Message	Illegal syntax of ENDM
	Cause	ENDM statement is incorrect.
E2619	Message	Illegal defined macro
	Cause	Referenced macro is incorrectly defined.
E2620	Message	Illegal syntax of actual parameter
	Cause	Specification of actual parameter of macro is incorrect.
E2621	Message	Nesting over of macro reference
	Cause	The limit on nesting in a macro reference (8 levels) is exceeded.
E2622	Message	Illegal syntax of EXITM
	Cause	EXITM statement is incorrect.
E2623	Message	Illegal operand of REPT
	Cause	An unpermitted expression is specified in the operand of a REPT directive.

Error Number	Error Message	
E2624	Message	More than ??RAFFFF
	Cause	More than 65535 local symbols are replaced during macro development.
E2625	Message	Unexpected ENDM
	Cause	An unexpected ENDM is found.
E2626	Message	Can't describe LOCAL outside macro definition
	Cause	LOCAL directive is specified in a normal source statement other than a macro body.
E2627	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.
W2701	Message	Too long source line
	Cause	Over 2048 characters are described on 1 line of a source statement.
	Program processing	2049th and subsequent characters are ignored.
W2702	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.
W2703	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.
W2704	Message	Already declared EXTRN symbol '*symbol name*'
	Cause	This symbol is already declared EXTRN.
	Action by User	Specify EXTRN declaration once in 1 module.
W2705	Message	Already declared EXTBIT symbol '*symbol name*'
	Cause	This symbol is already declared EXTBIT.
	Action by User	Specify EXTBIT declaration once in 1 module.
W2706	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.
W2707	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.

Error Number	Error Message	
W2708	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.
W2709	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.
W2710	Message	Over count of actual parameter
	Cause	More actual parameters are set than formal parameters.
	Program processing	Surplus actual parameters are ignored.
W2711	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.
W2712	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.
W2717	Message	Normal, callt and callf functions must be described together respectively.
	Cause	Debugging information may be illegal because normal, callt, and callf functions are not described together.
	Program processing	Describe normal and callt functions together.
E2801	Message	Illegal debug information
	Cause	The debug information in the source file is illegal.
	Action by User	Execute the compiler once again.
F2901	Message	Can't open source file '*file name*'
	Cause	Source file cannot be opened.
F2902	Message	Can't open parameter file '*file name*'
	Cause	Parameter file cannot be opened.
F2903	Message	Can't open include file '*file name*'
	Cause	Include file cannot be opened.
F2904	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.

Error Number	Error Message	
F2905	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 folder as the assembler execution format.
F2906	Message	Illegal overlay file '*file name*'
	Cause	Contents of overlay file are illegal.
F2907	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 folder.
F2908	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 folder.
F2909	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 folder.
F2910	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 folder.
F2913	Message	Can't read source file '*file name*'
	Cause	A file input/output error has occurred in the source file.
F2914	Message	Can't read parameter file '*file name*'
	Cause	A file input/output error has occurred in the parameter file.
F2915	Message	Can't read include file '*file name*'
	Cause	A file input/output error has occurred in the include file.
F2916	Message	Can't read overlay file '*file name*'
	Cause	A file input/output error has occurred in the overlay file.
F2917	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 folder or create an open area in the specified disk.
F2918	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 folder or create an open area in the specified disk.

Error Number	Error Message	
F2919	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 folder or create an open area in the specified disk.
F2920	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 folder or create an open area in the specified disk.
C2921	Message	Assembler internal error
	Cause	An assembler-internal error has occurred.
	Action by User	If the error cannot be resolved, contact NEC Electronics or an NEC Electronics distributor.
F2922	Message	Insufficient memory in hostmachine
	Cause	System does not have sufficient memory to execute assembler.
F2923	Message	Insufficient memory for macro in hostmachine
	Cause	Memory for macro became insufficient in the middle of macro processing.
	Action by User	Reduce number of macros defined.

11.3 Linker Error Messages

Table 11-2 Linker Error Messages

Error Number	Error Message	
F3001	Message	Missing input file
	Cause	An input file has not been specified.
	Action by User	Specify an input file.
F3002	Message	Too many input files
	Cause	Two or more input files have been specified.
	Action by User	Specify only one input file.
F3004	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.
F3005	Message	Illegal file specification '*file name*'
	Cause	An illegal file has been specified.
	Action by User	Specify a legal file.
F3006	Message	File not found '*file name*'
	Cause	The specified file does not exist.
	Action by User	Specify an existent file.
F3007	Message	Input file specification overlapped '*file name*'
	Cause	The input file name has already been specified elsewhere.
	Action by User	Input a unique file name.
F3008	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.
F3009	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.
F3010	Message	Directory not found '*file name*'
	Cause	A non-existent drive and/or folder has been included in the output file name.
	Action by User	Specify an existent drive and/or folder.
F3011	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.

Error Number	Error Message	
F3012	Message	Missing parameter '*option*'
	Cause	A necessary parameter has not been specified.
	Action by User	Specify the parameter.
F3013	Message	Parameter not needed '*option*'
	Cause	An unnecessary parameter has been specified.
	Action by User	Delete the unnecessary parameter.
F3014	Message	Out of range '*option*'
	Cause	The specified numerical value is outside the range.
	Action by User	Specify a correct numerical value.
F3015	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.
F3016	Message	Illegal parameter '*option*'
	Cause	The syntax of the parameter is incorrect.
	Action by User	Specify a correct parameter.
F3017	Message	Too many parameters '*option*'
	Cause	The total number of parameters exceeds the limit.
	Action by User	Specify parameters within the number limit.
F3018	Message	Option is not recognized '*option*'
	Cause	The option name is incorrect.
	Action by User	Specify a correct option name.
F3019	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.
F3020	Message	Parameter file read error '*file name*'
	Cause	The parameter file cannot be read.
	Action by User	Specify a correct parameter file.
F3021	Message	Memory allocation failed
	Cause	There is insufficient memory.
	Action by User	Secure the necessary memory.
F3031	Message	Security ID is not supported for this device
	Cause	A Security ID cannot be used with the specified device.
	Action by User	Do not specify a Security ID.

Error Number	Error Message	
F3101	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.
E3102	Message	Directive syntax error
	Cause	Specification of directive is incorrect.
F3103	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.)
F3104	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.
W3105	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.
F3106	Message	Can't create temporary file '*file name*'
	Cause	Cannot create temporary file.
E3107	Message	Name '*name*' in directive already defined
	Cause	Attempted to define a reserved word or a previously defined name as the memory area of a directive. This name (reserved word, memory space name, memory area name) is already defined.
E3108	Message	Overlapped memory area '*Memory area 1*' and '*Memory area 2*'
	Cause	The memory area addresses defined in the memory directive are overlapped.
E3109	Message	Memory area '*Memory area name*' too long name (up to 256 characters)
	Cause	The memory area name specified in the directive is too long. The memory area name specified in the directive is 256 characters or longer.
E3110	Message	Memory area '*Memory area name*' already defined
	Cause	The memory area specified in the memory directive is already registered.
E3111	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.
E3112	Message	Segment '*symbol name*' wrong allocation type
	Cause	Wrong allocation type is specified for the segment in the merge directive.

Error Number	Error Message	
C3113	Message	Linker internal error
	Cause	Internal error.
	Action by User	Contact NEC Electronics or an NEC Electronics distributor.
E3114	Message	Illegal number
	Cause	Specification of a numerical value in a directive is incorrect.
E3115	Message	Too large value (up to 1048575/0FFFFFH)
	Cause	A value greater than 1048575 (0FFFFFH) is described in the directive.
E3116	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 1048575 (0FFFFFH).
E3117	Message	Too Many line number data (up to 65535/0FFFFH) in the same name segment 'segment'
	Cause	The maximum number of line number entries per section, 65535, is exceeded.
E3118	Message	Can't find target chip in all modules
	Cause	The target device cannot be identified because the 78K0R common object specification option (-common) is specified for all the input object module files.
	Action by User	Remove the unnecessary 78K0R common object specification options (-common).
E3201	Message	Multiple segment definition '*symbol 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).
E3202	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.
F3203	Message	Segment '*symbol 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).
E3204	Message	Memory area/space '*name*' not defined
	Cause	Memory area/space name specified in merge directive is not defined.
E3205	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).
E3206	Message	Segment '*symbol name*' can't allocate to memory - ignored
	Cause	Segment cannot be allocated to memory (not enough memory area exists to allocate segment).
E3207	Message	Segment '*symbol name*' has illegal segment type
	Cause	This segment type data is illegal.

Error Number	Error Message	
E3208	Message	Segment '*symbol name*' may not change attribute
	Cause	Attempted to change the link type in the directive for a segment created with the reallocation attribute 'AT xxxxxH' specified during assemble, or created using the ORG directive.
E3209	Message	Segment '*symbol name*' may not change arrangement
	Cause	Attempted to change the allocation address in the directive for a segment created with the reallocation attribute 'AT xxxxxH' 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.
E3210	Message	Segment '*symbol name*' is not exist - ignored
	Cause	Segment specified in the directive does not exist.
E3212	Message	Default segment can’t allocate to memory-ignored
	Cause	The default segment cannot be allocated to the memory area.
	Action by User	Check if data of -gb, -gi, or -go can be allocated to ROM area ranges.
W3213	Message	Segment '*segment name*' allocated on General-purpose registers
	Cause	The segment is allocated to a general-purpose register.
	Action by User	An unexpected overwrite to the general-purpose register may occur, so reallocate it as necessary.
E3301	Message	Relocatable object code address out of range (file '*file name*', segment '*symbol name*', address xxxxxH, 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.
E3302	Message	Illegal symbol index in line number (file '*file name*', segment '*symbol 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.
E3303	Message	Can't find symbol index in relocatable object code (file '*file name*', segment '*symbol name*', address xxxxxH, 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.
E3304	Message	Operand out of range (file '*file name*', segment '*symbol name*', symbol '*symbol name*', address xxxxxH, 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.

Error Number	Error Message	
E3305	Message	Even value expected (file '*file name*', segment '*symbol name*', symbol '*symbol name*', address xxxxxH, type '*addressing type*')
	Cause	The operand value used to determine the callt or saddrp addressing relocatable object code is an odd number (callt and saddrp addressing operands must be even numbers).
E3306	Message	A multiple of 4 value expected (segment '*symbol name*', address xxxxxH, type '*addressing type*')
	Cause	The value of the operand used for resolving the relocatable object code for saddr addressing is not a multiple of 4.
F3401	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'.
F3402	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.
E3403	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.
E3404	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.
E3405	Message	Undefined symbol '*symbol name*' in file '*file name*'
	Cause	Symbol declared EXTRN in the file is not declared PUBLIC in another file.
W3406	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 the -s option is 10 bytes or less).
W3407	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 the -s option).
E3410	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.
W3411	Message	Different REL type in '*file name*'
	Cause	The type version of object module file is different.
	Action by User	Re-assemble or re-compile with the newest version.

Error Number	Error Message	
E3415	Message	Compiler options are mixed in file '*file name1*' First defined in file '*file name2*'
	Cause	The same compiler optimization option should have been specified throughout the entire program, but an object file with a different optimization specification was input. Recompile the program with the same specification.
W3416	Message	Multiple CAP/NOCAP are in file '*file name (option)*' Defined first one in file '*file name (option)*'
	Cause	CAP/NOCAP assemble or compile options are not identical for all input object module files.
W3417	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 (CC78K0R, RA78K0R) used until the link stage for all input object module files and the device file version.
W3418	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.
W3420	Message	File '*file name*' has already had error(s)/warning(s) by '*tool name*'
	Cause	An error message or warning message for each tool (CC78K0R, RA78K0R) used until the link stage is output.
E3424	Message	-ZF REL and no -ZF REL are mixed in file '*file name*'
	Cause	When linking load module of the boot area ROM program of a flash memory model with object module of the flash area program, some object module do not specify the -zf option during compilation.
E3425	Message	There are different function ID in same name '*function name*' (file '*file name*')
	Cause	A function of the same name declared as EXT_FUNC by the compiler has a different ID value.
F3426	Message	Multiple input BOOT file '*file name 1*'. First input file '*file name 2*'
	Cause	Two or more load modules in the boot area ROM program were input when a load module in the boot area ROM program of a flash memory model was to be linked with an object module in the flash area program.
	Action by User	Specify only one load module file for the boot area ROM program.
E3427	Message	BOOT REL and -ZF REL are mixed in file '*file name*'
	Cause	Object module specified by the -zf option is input during compilation for linking with the -zf option specified.
E3428	Message	FLASH start address larger than ROM max address
	Cause	The first address of the flash memory area is greater than the ROM end address of the target device.
E3429	Message	BOOT segment '*symbol name*' are found in FLASH file '*file name*'
	Cause	When load module of the boot area ROM program of a flash memory model is linked with object module of the flash area ROM program, a segment with a location address less than the first address of the flash memory area exists in the object module.

Error Number	Error Message	
F3430	Cause	Different FLASH address in file '*file name 1*'. First specified in file '*file name 2*'
	Action by User	Several values are specified as the start address of the flash memory area in the input file.
	Cause	Specify the same value for all of the -zb option description.
E3431	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.
E3432	Message	Illegal allocation of an EXT_FUNC function '*function name*' (file '*file name*')
	Cause	The entity of the function declared as EXT_FUNC by the compiler exists when linking is performed with the -zb option specified.
E3433	Message	Can't find FLASH start address in file '*file name*'
	Cause	The start address of the flash memory is missing.
	Action by User	Input the LMF file for which -zb is specified.
W3434	Message	Can't specify User Option Bytes/On-Chip Debug Option Bytes/Security ID with LMF
	Cause	The -gb, -go, or -gi option cannot be specified if a load module file is specified for the input file.
	Action by User	Do not specify the -gb, -go, or -gi option when a load module file is specified for the input file to be relinked.
F3435	Message	ext_table address in file '%s'. First specified in file '%s'
	Cause	The value of #pragma ext_table specified in the C source is illegal.
	Action by User	Specify the same value for "ITBLTOP" in the startup routine and #pragma ext_table in the C source file.
F3502	Message	Too many segment (up to 65535/0FFFFH)
	Cause	Total number of input segments exceeds 65,535.
F3901	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 folder (a folder containing an execution program).
F3902	Message	File '*file name*' not found
	Cause	The specified library file cannot be opened.
F3903	Message	Can't read input file '*file name*'
	Cause	Object module file specified as an input file cannot be read.
F3904	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.

Error Number	Error Message	
F3905	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.
F3906	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.
F3907	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.
F3908	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.
F3909	Message	Can't read DEVICE_FILE file '*device file name'*
	Cause	Device file corresponding to device specified by each tool (CC78K0R, RA78K0R) used until the link stage cannot be read.

Caution The address shown in 'address xxxxH' in the messages in E3301 to E3306 are absolute addresses after segment allocation.

11.4 Object Converter Error Messages

Table 11-3 Object Converter Error Messages

Error Number	Error Message	
F4001	Message	Missing input file
	Cause	An input file has not been specified.
	Action by User	Specify an input file.
F4002	Message	Too many input files
	Cause	Two or more input files have been specified.
	Action by User	Specify only one input file.
F4004	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.
F4005	Message	Illegal file specification '*file name*'
	Cause	An illegal file has been specified.
	Action by User	Specify a legal file.
F4006	Message	File not found '*file name*'
	Cause	The specified input file does not exist.
	Action by User	The file is output as "startup routine name.lmf" if the startup routine of the C compiler is linked. In this case, specify an output file name with a linker option, like "-o*.lmf".
F4008	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.
F4009	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.
F4010	Message	Directory not found '*file name*'
	Cause	A non-existent drive and/or folder has been included in the output file name.
	Action by User	Specify an existent drive and/or folder.
F4011	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.

Error Number	Error Message	
F4012	Message	Missing parameter '*option*'
	Cause	A necessary parameter has not been specified.
	Action by User	Specify the parameter.
F4013	Message	Parameter not needed '*option*'
	Cause	An unnecessary parameter has been specified.
	Action by User	Delete the unnecessary parameter.
F4014	Message	Out of range '*option*'
	Cause	The specified numerical value is outside the range.
	Action by User	Specify a correct numerical value.
F4015	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.
F4016	Message	Illegal parameter '*option*'
	Cause	The syntax of the parameter is incorrect.
	Action by User	Specify a correct parameter.
F4017	Message	Too many parameters '*option*'
	Cause	The total number of parameters exceeds the limit.
	Action by User	Specify parameters within the number limit.
F4018	Message	Option is not recognized '*option*'
	Cause	The option name is incorrect.
	Action by User	Specify a correct option name.
F4019	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.
F4020	Message	Parameter file read error '*file name*'
	Cause	The parameter file cannot be read.
	Action by User	Specify a correct parameter file.
F4021	Message	Memory allocation failed
	Cause	There is insufficient memory.
	Action by User	Secure the necessary memory.
F4100	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.

Error Number	Error Message	
F4101	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.
F4103	Message	Symbol '*symbol name*' Illegal attribute
	Cause	A mistake exists in the symbol attribute of the input file.
F4104	Message	'*file name*' Illegal input file - not linked
	Cause	Attempted to input an object module file.
F4105	Message	Insufficient memory in hostmachine
	Cause	Memory is not sufficient to operate the program.
F4106	Message	Illegal symbol table
	Cause	A mistake exists in the symbol table of the input load module file.
	Action by User	If the source is written in C, make sure that the assembler code in the C source satisfies the following conditions: (Conditions) If a local symbol is used, use a symbol that starts with string ?L (such as ?L@01 or ?L@sym). However, keep this symbol to within 8 characters. Do not define this symbol externally (PUBLIC declaration).
F4107	Message	Can't specify -U option for ROMless device
	Cause	The object complement specification option (-u) is specified for a model without internal ROM.
E4200	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.
E4201	Message	Out of address range
	Cause	The address of an object in a load module file is out of range.
W4300	Message	xxxxxH - yyyyyH overlapped
	Cause	Objects overlapped in the address from xxxxxH to yyyyyH are output.
W4301	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 described in DSEG of the assembly source, change it to DS or describe the DB/DW instruction in CSEG.
F4900	Message	Can't open file '*file name*'
	Cause	File cannot be opened.
F4901	Message	Can't close file '*file name*'
	Cause	File cannot be closed.

Error Number	Error Message	
F4902	Message	Can't read file '*file name*'
	Cause	File cannot be correctly read.
F4903	Message	Can't access file '*file name*'
	Cause	File cannot be correctly read or written to.
F4904	Message	Can't write file '*file name*'
	Cause	Data cannot be correctly written to an output file.
F4905	Message	Can't open overlay file '*file name*'
	Cause	The overlay file cannot be opened.
	Action by User	Check if the overlay file is in the same folder as the execution format.
C4999	Message	Object Converter internal error
	Cause	This is an internal error.
	Action by User	Contact NEC Electronics or an NEC Electronics distributor.

11.5 Librarian Error Messages

Table 11-4 Librarian Error Messages

Error Number	Error Message	
F5001	Message	Missing input file
	Cause	Only options are specified. No input files are specified.
F5002	Message	Too many input files
	Cause	Total number of input files exceeds the limit.
F5003	Message	Unrecognized string '???'
	Cause	Something other than an option is specified on a conversational-format command line.
F5004	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.
F5005	Message	Illegal file specification '*file name*'
	Cause	An illegal item is specified in the file name.
F5006	Message	File not found '*file name*'
	Cause	Specified input file does not exist.
F5007	Message	Input file specification overlapped '*file name*'
	Cause	Input file name specification is overlapped.
F5008	Message	File specification conflicted '*file name*'
	Cause	Input or output file name specifications overlap.
F5009	Message	Unable to make file '*file name*'
	Cause	Specified output file cannot be created.
F5010	Message	Directory not found '*file name*'
	Cause	A drive or folder which does not exist is included in the output file name.
F5011	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.
F5012	Message	Missing parameter '*option*'
	Cause	Required parameter is not specified.
F5013	Message	Parameter not needed '*option*'
	Cause	An unnecessary parameter is specified.
F5014	Message	Out of range '*option*'
	Cause	Specified value is out of range.
F5015	Message	Parameter is too long '*option*'
	Cause	Number of characters specified in parameter exceeds limit.

Error Number	Error Message	
F5016	Message	Illegal parameter '*option*'
	Cause	A mistake exists in the syntax of the parameter.
F5017	Message	Too many parameters '*option*'
	Cause	Total number of parameters exceeds limit.
F5018	Message	Option is not recognized '*option*'
	Cause	An incorrect option is specified.
F5019	Message	Parameter file nested
	Cause	-f option is specified in a parameter file.
F5020	Message	Parameter file read error '*file name*'
	Cause	An error occurred in reading a parameter file.
F5021	Message	Memory allocation failed
	Cause	Memory allocation has failed.
F5022	Message	Memory allocation failed
	Cause	Memory allocation has failed.
F5023	Message	Illegal character ',' before file name
	Cause	Necessary ',' exists before the input file.
F5024	Message	Illegal character
	Cause	An illegal character or character string is found.
F5025	Message	Qualifier is not unique.
	Cause	The abbreviation type of the modifier is not unique.
F5026	Message	Umbigous input redirect.
	Cause	No file name is specified after '<', or '< file name' is specified more than once.
C5100	Message	Internal error
	Cause	An internal error has occurred.
E5101	Message	Invalid sub command
	Cause	Subcommand name is incorrect.
E5102	Message	Invalid syntax
	Cause	Parameter specification in subcommand is incorrect.
E5103	Message	Illegal input file - different target chip (file: *file name*)
	Cause	Specification of target device in input object module file is incorrect.
E5104	Message	Illegal library file - different target chip (file: *file name*)
	Cause	Specification of target device in library file is incorrect.

Error Number	Error Message	
E5105	Message	Module not found (module: *file name*)
	Cause	Specified module does not exist in library file.
E5106	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.
E5107	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 '. '.
E5108	Message	Multiple transaction file (file: *file name*)
	Cause	Input object module files overlap.
E5109	Message	Public symbol already exists (symbol: *symbol name*)
	Cause	An externally defined symbol name already exists in an updated library file or other input file.
E5110	Message	File specification conflicted (file: *file name*)
	Cause	Specified input file name is same as output file name.
E5111	Message	Illegal file format (file: *file name*)
	Cause	Format of an updated library file or other input file is incorrect.
E5112	Message	Library file not found (file: *file name*)
	Cause	Specified library file is not found.
E5113	Message	Object module file not found (file: *file name*)
	Cause	Specified object module file is not found.
E5114	Message	No free space for temporary file
	Cause	Sufficient space does not exist in the disk to create a temporary file.
E5115	Message	Not enough memory
	Cause	Sufficient memory is not available to operate the program.
E5116	Message	Sub command Buffer full
	Cause	Limit for continuous line length in a subcommand (128 x 15 characters) is exceeded. Limit for length of 1 line in a subcommand (128 characters) is exceeded.
E5117	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.
E5118	Message	Illegal path (file: *file name*)
	Cause	Path name in the specified file is incorrect.
W5201	Message	Module not found (module: *file name*)
	Cause	The module for which replace is specified is not in the library file.

Error Number	Error Message	
F5901	Message	File open error (file: *file name*)
	Cause	File cannot be opened.
F5902	Message	File read error (file: *file name*)
	Cause	File cannot be correctly read.
F5903	Message	File write error (file: *file name*)
	Cause	Data cannot be correctly written to file.
F5904	Message	File seek error (file: *file name*)
	Cause	File seek error has occurred.
F5905	Message	File close error (file: *file name*)
	Cause	File cannot be closed.

11.6 List Converter Error Messages

Table 11-5 List Converter Error Messages

Error Number	Error Message	
F6001	Message	Missing input file
	Cause	An input file has not been specified.
	Action by User	Specify an input file.
F6002	Message	Too many input files
	Cause	Two or more input files have been specified.
	Action by User	Specify only one input file.
F6004	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.
F6005	Message	Illegal file specification '*file name*'
	Cause	An illegal file has been specified.
	Action by User	Specify a legal file.
F6006	Message	File not found '*file name*'
	Cause	The specified file does not exist.
	Action by User	Specify an existent file.
F6008	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.
F6009	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.
F6010	Message	Directory not found '*file name*'
	Cause	A non-existent drive and/or folder has been included in the output file name.
	Action by User	Specify an existent drive and/or folder.
F6011	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.
F6012	Message	Missing parameter '*option*'
	Cause	A necessary parameter has not been specified.
	Action by User	Specify the parameter.

Error Number	Error Message	
F6013	Message	Parameter not needed '*option*'
	Cause	An unnecessary parameter has been specified.
	Action by User	Delete the unnecessary parameter.
F6014	Message	Out of range '*option*'
	Cause	The specified numerical value is outside the range.
	Action by User	Specify a correct numerical value.
F6015	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.
F6016	Message	Illegal parameter '*option*'
	Cause	The syntax of the parameter is incorrect.
	Action by User	Specify a correct parameter.
F6017	Message	Too many parameters '*option*'
	Cause	The total number of parameters exceeds the limit.
	Action by User	Specify parameters within the number limit.
F6018	Message	Option is not recognized '*option*'
	Cause	The option name is incorrect.
	Action by User	Specify a correct option name.
F6019	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.
F6020	Message	Parameter file read error '*file name*'
	Cause	The parameter file cannot be read.
	Action by User	Specify a correct parameter file.
F6021	Message	Memory allocation failed
	Cause	There is insufficient memory.
	Action by User	Secure the necessary memory.
F6101	Message	File is not 78K0R '*file name*'
	Cause	Input file name is not a 78K0R file name.
F6102	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.
F6103	Message	Load module file has relocation data '*file name*'
	Cause	Address of load module file is not determined.

Error Number	Error Message	
F6104	Message	Object module file is executable '*file name*'
	Cause	Object module file is in an executable format.
F6105	Message	Segment name is not found in module file '*symbol name*'
	Cause	Segment name of object module file is not found in load module file.
F6106	Message	Segment name is not found in object module file '*symbol name*'
	Cause	Segment name of assemble list file is not found in object module file.
F6107	Message	Not enough memory
	Cause	Memory is not sufficient for program operation.
F6108	Message	Load module file has no symbol data '*load module name*'
	Cause	The -ng option is specified in linker, so symbol data in load module file cannot be output.
F6109	Message	Overlay file can not open 'path name'
	Cause	Assembler overlay file cannot be opened.
F6110	Message	Illegal assembler list file '*file name*'
	Cause	Input assemble list is a file type other than an assemble list file.
W6701	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.
W6702	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.
W6703	Message	Assemble list has error statement '*file name*'
	Cause	An error exists in the assemble list.
W6704	Message	Segment name is not found in assemble list file '*symbol name*'
	Cause	Segment name of object module file is not found in assemble list.
W6705	Message	Segment data length is different '*symbol 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.
F6901	Message	File open error has occurred '*file name*'
	Cause	File cannot be opened.
F6902	Message	File read error has occurred '*file name*'
	Cause	File cannot be correctly read.
F6903	Message	File write error has occurred '*file name*'
	Cause	Data cannot be correctly written to file.

Error Number	Error Message	
F6904	Message	File seek error has occurred '*file name*'
	Cause	File seek error has occurred.
C6999	Message	Internal error
	Cause	Program-internal error

11.7 PM+ Error Messages

This section explains error messages that are not contained in the online help of PM+. For information on other PM+ error messages, please refer to PM+ Online Help.

11.7.1 Assembler (RA78K0R)

Table 11-6 Error Messages for DLL for Assembler (RA78K0R)

Error Type	Error Message	
x	Message	Cannot find RA78K0R.EXE shown in environmental variable PATH.
	Cause	The RA78K0R.EXE execution format is not in the specified folder.
	Action by User	Re-install the RA78K0R assembler package.
	Button	[OK] … Closes the message box.
!	Message	Cannot find folder. Will you create?
	Cause	The specified folder does not exist.
	Action by User	Create a folder or select another folder.
	Button	[OK] …Creates a folder and closes the message box. [Cancel] …Closes the message box.
!	Message	Not make folder.
	Cause	The specified folder could not be created.
	Action by User	Specify another folder.
	Button	[OK] … Closes the message box.
!	Message	Invalid value. The range of columns per line is from 72 to 2046.
	Cause	A value exceeding the limit range is specified as the number of characters on one line.
	Action by User	Specify a value within the range.
	Button	[OK] … Closes the message box.
!	Message	Invalid value. The range of line per page is from 0, or 20 to 32767.
	Cause	A value outside the limit range is specified as the number of lines on one page.
	Action by User	Specify a value within the limit range.
	Button	[OK] … Closes the message box.

Error Type	Error Message	
!	Message	Invalid value. The range of TAB character is from 0 to 8.
	Cause	A value outside the input limit range is described as the number of tabs up to an operand.
	Action by User	Specify a value within the limit range.
	Button	[OK] … Closes the message box.
!	Message	Too many Include Search Path. Up to 64 can be specified for Include Search Path.
	Cause	A number of include file paths outside the input limit range is specified.
	Action by User	Specify a number within the limit range.
	Button	[OK] … Closes the message box.
!	Message	Too many characters for Include Search Path.
	Cause	The length of the include file path is specified as a number of characters exceeding the input range.
	Action by User	Specify a number of characters in the input range.
	Button	[OK] … Closes the message box.
!	Message	Multiple Include Search Path definition. (Include file path)
	Cause	Include file paths are described in duplicate.
	Action by User	Delete the duplicate path(s) and retry.
	Button	[OK] … Closes the message box.
!	Message	Too many symbols for Symbol Definition. Up to 30 symbols can be specified.
	Cause	A number of symbols exceeding the input range is described for the symbol definition.
	Action by User	Keep the number of symbols to within the usable range and retry.
	Button	[OK] … Closes the message box.
!	Message	Too many characters for Symbol Definition. Up to 31 characters can be described for symbol name.
	Cause	The length of the defined symbol is greater than the number of characters in the input range.
	Action by User	Keep the number of characters of the symbol to within the range and retry.
	Button	[OK] … Closes the message box.
!	Message	Multiple symbol definition. (symbol)
	Cause	A symbol is described in duplicate for the symbol definition.
	Action by User	Delete the duplicate symbol and retry.
	Button	[OK] … Closes the message box.

Error Type	Error Message	
!	Message	Invalid word symbol definition. (symbol)
	Cause	An illegal character is used for symbol definition.
	Action by User	Define the symbol using an appropriate character.
	Button	[OK] … Closes the message box.
!	Message	Can't set options to (source file name).
	Cause	If a common option is reflected on an individual option, specification of the individual option becomes illegal.
	Action by User	Check the specification of the individual option and retry.
	Button	[OK] … Closes the message box.

11.7.2 Linker (LK78K0R)

Table 11-7 Error Messages for DLL for Linker (LK78K0R)

Error Type	Error Message	
x	Message	Cannot find LK78K0R.EXE shown in environmental variable PATH.
	Cause	The LK78K0R.EXE execution format is not in the specified folder.
	Action by User	Re-install the RA78K0R assembler package.
	Button	[OK] … Closes the message box.
!	Message	Invalid value. The range of warning level is from 0 to 2.
	Cause	A value outside the limit input range is specified as the warning level.
	Action by User	Specify a value within the limit range.
	Button	[OK] … Closes the message box.
!	Message	Invalid value. The range of flash start address is from 0h to 0FFFFFh.
	Cause	A value outside the input limit range is specified as the flash start address.
	Action by User	Specify a value within the limit range.
	Button	[OK] … Closes the message box.
!	Message	Cannot find path or file. Make sure path or filename.
	Cause	The specified path or file cannot be found.
	Action by User	Specify a correct path or file name.
	Button	[OK] … Closes the message box.
!	Message	Cannot find folder. Will you create?
	Cause	The specified folder does not exist.
	Action by User	Create a folder or select another folder.
	Button	[OK] … Creates a folder and closes the message box. [Cancel] … Closes the message box.
!	Message	Not make folder.
	Cause	The specified folder cannot be created.
	Action by User	Specify another folder.
	Button	[OK] … Closes the message box.
!	Message	Out of range. The range of Control Value for On-Chip Debug Option Bytes is from 0h to 0FFh.
	Cause	A value that cannot be specified for the target device is specified for the control value for on-chip debug option byte.
	Action by User	Refer to the user's manual of the device used to check the specifiable value.
	Button	[OK] … Closes the message box.

Error Type	Error Message	
!	Message	Invalid Control Value for On-Chip Debug Option Bytes. Missing parameter.
	Cause	The control value for on-chip debug option byte has not been input.
	Action by User	Input the control value for on-chip debug option byte.
	Button	[OK] … Closes the message box.
!	Message	Invalid Control Value for On-Chip Debug Option Bytes. Control Value for On-Chip Debug Option Bytes is specified in hexadecimal numbers.
	Cause	The specification format of the control value specified for on-chip debug option byte is illegal.
	Action by User	Specify the control value in the correct format.
	Button	[OK] … Closes the message box.
!	Message	Out of range. The range of Start Address for On-Chip Debug Option Bytes is from 0h to 0FFFFFh.
	Cause	The start address specified for on-chip debug option byte is out of the specifiable range.
	Action by User	Specify the start address within the specifiable range.
	Button	[OK] … Closes the message box.
!	Message	Invalid Start Address for On-Chip Debug Option Bytes. Missing parameter.
	Cause	The start address for on-chip debug option byte has not been input.
	Action by User	Input the start address for on-chip debug option byte.
	Button	[OK] … Closes the message box.
!	Message	Invalid Start Address for On-Chip Debug Option Bytes. Start Address for On-Chip Debug Option Bytes is specified in hexadecimal numbers.
	Cause	The specification format of the start address specified for on-chip debug option byte is illegal.
	Action by User	Specify the start address in the correct format.
	Button	[OK] … Closes the message box.
!	Message	Invalid value. The range of Size for On-Chip Debug Option Bytes is from 256 to 1024.
	Cause	The size specified for on-chip debug option byte is outside the input limit range.
	Action by User	Specify a value within the limit range and retry.
	Button	[OK] … Closes the message box.

Error Type	Error Message	
!	Message	Invalid Start Address for On-Chip Debug Option Bytes. Size for On-Chip Debug Option Bytes is specified in decimal numbers.
	Cause	The specification format of the size specified for on-chip debug option byte is illegal.
	Action by User	Specify the size in the correct format.
	Button	[OK] … Closes the message box.
!	Message	Invalid Security ID. Security ID is specified in hexadecimal numbers.
	Cause	The specified Security ID is not in the correct format.
	Action by User	Specify the Security ID in the correct format.
	Button	[OK] … Closes the message box.
!	Message	Invalid Security ID. Missing parameter.
	Cause	A Security ID has not been input.
	Action by User	Input a Security ID.
	Button	[OK] … Closes the message box.
!	Message	Too many figures for Security ID. Up to 20 can be specified for Security ID.
	Cause	The specification format of the specified Security ID is incorrect.
	Action by User	Specify the Security ID in the correct format and retry.
	Button	[OK] … Closes the message box.
!	Message	Out of range. The range of User Option Byte is from 0h to 0FFFFFFFFh.
	Cause	The size specified for user option byte is outside the input limit range.
	Action by User	Specify a value within the limit range and retry.
	Button	[OK] … Closes the message box.
!	Message	Invalid User Option Byte. Missing parameter.
	Cause	The user option byte value has not been input.
	Action by User	Input the user option byte value.
	Button	[OK] … Closes the message box.
!	Message	Invalid User Option Byte. Option Bytes is specified in hexadecimal numbers.
	Cause	The specification format of the User Option Byte is illegal.
	Action by User	Specify the value in the correct format.
	Button	[OK] … Closes the message box.

Error Type	Error Message	
!	Message	Too many files for Library File. Up to 64 can be specified for Library File.
	Cause	A number of library files outside the input limit range is specified.
	Action by User	Specify a number of library files within the limit range.
	Button	[OK] … Closes the message box.
!	Message	(library file) Too long file name for Library File.
	Cause	The file name specified as a library file has a number of characters outside the limit range.
	Action by User	Retry with a number of characters within the limit range.
	Button	[OK] … Closes the message box.
!	Message	(library file) Multiple Library File definition.
	Cause	Library files are described in duplicate.
	Action by User	Delete the duplicate library file(s) and retry.
	Button	[OK] … Closes the message box.
!	Message	Too many path for Library File Search Path. Up to 64 can be specified.
	Cause	A number of paths for reading a library file is described outside the limit input range.
	Action by User	Specify a number of paths within the limit range and retry.
	Button	[OK] … Closes the message box.
!	Message	(library file read path) Too many characters for Library File Search Path.
	Cause	The library file read path is described with a number of characters outside the limit input range.
	Action by User	Specify the library file read path with a number of characters within the limit range.
	Button	[OK] … Closes the message box.
!	Message	(library file read path) Multiple Library File Search Path definition.
	Cause	Library file read paths are described in duplicate.
	Action by User	Delete the duplicate path(s) and retry.
	Button	[OK] … Closes the message box.

11.7.3 Object Converter (OC78K0R)

Table 11-8 Error Messages for DLL for Object Converter (OC78K0R)

Error Type	Error Message	
x	Message	Cannot find OC78K0R.EXE shown in environmental variable PATH.
	Cause	The OC78K0R.EXE execution format is not in the specified folder.
	Action by User	Re-install the RA78K0R assembler package.
	Button	[OK] … Closes the message box.
!	Message	Invalid description format with object complement.
	Cause	The object complement value is specified in an illegal format.
	Action by User	Specify a format that can be described.
	Button	[OK] … Closes the message box.
!	Message	Invalid value. The range of complement value is from 0h to 0FFh.
	Cause	A value outside the input limit range is specified as the complement value.
	Action by User	Specify a value within the limit range.
	Button	[OK] … Closes the message box.
!	Message	Out of range. The range of start address is from 0h to 0FFEFFH.
	Cause	A value outside the input limit range is specified as the start address.
	Action by User	Specify a value within the limit range.
	Button	[OK] … Closes the message box.
!	Message	Out of range. The range of size is from 1h to 0FFF00H.
	Cause	A value outside the limit input range is specified as the size.
	Action by User	Specify a value within the limit range.
	Button	[OK] … Closes the message box.
!	Message	Cannot find folder. Will you create?
	Cause	The specified folder does not exist.
	Action by User	Create a folder or select another folder.
	Button	[OK] … Creates a folder and closes the message box. [Cancel] … Closes the message box.
!	Message	Not make folder.
	Cause	The specified folder could not be created.
	Action by User	Specify another folder.
	Button	[OK] … Closes the message box.

11.7.4 Librarian (LB78K0R)

Table 11-9 Error Messages for DLL for Librarian (LB78K0R)

Error Type	Error Message	
x	Message	Cannot find LB78K0R.EXE shown in environmental variable PATH.
	Cause	The LB78K0R.EXE execution format is not in the specified folder.
	Action by User	Re-install the RA78K0R assembler package.
	Button	[OK] … Closes the message box.
!	Message	Cannot find folder. Will you create?
	Cause	The specified folder does not exist.
	Action by User	Create a folder or select another folder.
	Button	[OK] … Creates a folder and closes the message box. [Cancel] … Closes the message box.
!	Message	Not make folder.
	Cause	The specified folder could not be created.
	Action by User	Specify another folder.
	Button	[OK] … Closes the message box.

Table 11-10 Error Messages Displayed by Execution Format DLL (lb78k0rp.exe) for Starting Librarian in Standalone Mode

Error Type	Error Message	
!	Message	Do not be specified library file name.
	Cause	No file name has been specified in the edit box for library file name specification.
	Action by User	Specify the correct file name.
	Button	[OK] … Closes the message box.
!	Message	Cannot find folder. Will you create ?
	Cause	The specified output path does not exist.
	Action by User	Create a folder or select another folder.
	Button	[OK] … Closes the message box.
!	Message	Not make folder.
	Cause	An error has occurred during folder creation in the case that creation of folders is selected.
	Action by User	Check if there are problems in the system folder structure.
	Button	[OK] … Closes the message box.
?	Message	LB execute error. Display log ?
	Cause	An error has occurred during LB execution.
	Action by User	Select either of the following actions by clicking the [OK] button (displays LB execution log) or [Cancel] button (does not display the LB execution log).
	Button	[OK] … Displays the LB execution log. [Cancel] … Closes the message box.
!	Message	No file selected in left list box.
	Cause	No file is selected in the left-hand list box for execution of the add or replace subcommand.
	Action by User	Select at least one file.
	Button	[OK] … Closes the message box.
!	Message	No file selected in right list box.
	Cause	No file is selected in the right-hand list box for execution of the pick or delete subcommand.
	Action by User	Select at least one file.
	Button	[OK] … Closes the message box.
?	Message	Would you like to delete the selected modules ?
	Cause	This is the message for confirming deletion of the selected module in the library.
	Action by User	Select either of the following actions by clicking the [OK] button (delete) or [Cancel] button (do not delete).
	Button	[OK] … Deletes the module selected in the library. [Cancel] … Closes the message box.

Error Type	Error Message	
!	Message	Invalid output file name.
	Cause	An illegal file was specified in the field for specifying the list output file name.
	Action by User	Specify the correct output file name.
	Button	[OK] … Closes the message box.
!	Message	Invalid value. The range of columns per line is from 72 to 260.
	Cause	A value out of the specifiable range was specified for the number of characters per line in the output list file.
	Action by User	Specify the value within the specifiable range.
	Button	[OK] … Closes the message box.
!	Message	Invalid value. The range of lines per page is from 0, 20 to 32767.
	Cause	A value out of the specifiable range was specified for the number of lines per page in the output list file.
	Action by User	Specify the value within the specifiable range.
	Button	[OK] … Closes the message box.

11.7.5 List Converter (LC78K0R)

Table 11-11 Error Messages for DLL for List Converter (LC78K0R)

Error Type	Error Message	
x	Message	Cannot find LC78K0R.EXE shown in environmental variable PATH.
	Cause	The LC78K0R.EXE execution format is not in the specified folder.
	Action by User	Re-install the RA78K0R assembler package.
	Button	[OK] … Closes the message box.
!	Message	Cannot find folder. Will you create?
	Cause	The specified folder does not exist.
	Action by User	Create a folder or select another folder.
	Button	[OK] … Creates a folder and closes the message box. [Cancel] … Closes the message box.
!	Message	Not make folder.
	Cause	The specified folder could not be created.
	Action by User	Specify another folder.
	Button	[OK] … Closes the message box.

APPENDIX A SAMPLE PROGRAMS

This chapter is an introduction to the sample lists attached to the RA78K0R.

A.1 k0rmain.asm

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

PUBLIC  MAIN , START
EXTRN   CONVAH
EXTRN   _@STBEG

DATA    DSEG    AT      0FFE20H
HDTSA : DS      1
STASC : DS      2

CODE    CSEG    AT      0H
MAIN :  DW      START

        CSEG
START :

        ; chip initialize
        MOVW    SP , #_@STBEG

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

        CALL    !CONVAH                 ; convert ASCII <- HEX
                                        ; output BC-register <- ASCII code
        MOVW    DE , #LOWW ( 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 k0rsub.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 :
        XOR     A , A
        ROL4    [ HL ]          ; hex lower code load
        CALL    !SASC
        MOV     B , A           ; store result

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

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

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

        END
```

APPENDIX B NOTES ON USE

In this chapter, when using the RA78K0R, note the following points:

(1) Device file

A device file is required to execute the RA78K0R. The device file is not included in the RA78K0R package, and must be obtained separately.

Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

http://www.necel.com/micro/ods/eng/

(2) Object converter

Use the object converter by specifying the -r (address source of object) and -u (complement value specification) options.

These options are specified by default.

An abort error occurs if a ROM code is ordered (work known as "across processing" or "tape out") when the addresses of the objects are not sorted. Therefore, be sure to specify -r (do not cancel the specification).

(3) Memory initialization quasi directives

If memory initialization quasi directives DB, DW, and DG are described in a data segment (DSEG), object codes are output but the object converter outputs the warning message W4301. This is because a code exists at an address other than one in the ROM area (code area).

An abort error occurs if a ROM code is ordered (work known as "across processing" or "tape out") in this status.

(4) Manipulation specification of object converter

If starting address is specified by the object converter the -u option, complement is started from the start address or the address where the code is located, whichever is lower. complement is not performed for the internal RAM area (ED800H to FFFFFH).

Description format: -u*complement-value*[,[*start-address*],*size*]

Remark [ ] may be omitted.

(5) Memory directives

The default memory area name of each device cannot be erased.

Set the size of the default memory area name not used to 0.

Some segments, however, are allocated to the default area. Bear this in mind when changing the area name.

For the default memory area name, refer to the User's Manual of the device to be used.

(6) Debug option

If debug information is output by the C compiler and the compiler is executed, do not output the debug information when the output assemble source is assembled (specify this by using the -nga option). If the debug information is output, debugging may not be executed at the C compiler source level.

(7) CC78K0R

Several points must be noted when C source level debugging is executed by assembling the assembler source output by the CC78K0R.

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

(8) ID78K0R-QB, SM+ for 78K0R

When debugging is executed with the ID78K0R-QB, SM+ for 78K0R, keep the number of symbols and the number of source lines to within the limit of the ID78K0R-QB, SM+ for 78K0R.

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

(9) Segment name

When describing a segment name, do not describe the same name as the primary name of the source file name. Otherwise, the abort error F2106 will occur as a result of assembly.

(10) EQU definition of SFR name

An SFR name may be specified as an operand of the EQU quasi directive. If the name of an SFR outside the saddr area is specified as PUBLIC, an assembly error occurs.

(11) When using a network

If a folder where a temporary file is to be created is on a file system that is shared on a network, a file conflict may occur and an abnormal operation may be performed. Avoid this conflict by using options and environmental variables.

(12) When using self programming

The following describes notes on the tool options for self programming.

(a) Assembler

If firmware for self programming is allocated outside the internal ROM area specified in the device file, the RA78K0 may output an error for the description of "CALL !8100H". This error is avoided by specifying the -self option.

With the RA78K0R, describe "CALL !!addr20" to call firmware for self programming because “CALL !xxxxxH” can be described in the entire space.

(b) Linker

Use the -zb option to create a load module file for the boot area.

Following the -zb option, specify the start address in the flash memory area.

To create a load module file for the flash area, input the load module file for the boot area and the object module file for the flash area and relink them.

Allocate the object module file for the flash area at an address later than the one specified with the -zb option.

(c) Object Converter

Input a load module file for the boot area and an object module file for the flash area, relink them, and input the output load module file to the OC78K0R.

By specifying the -zf option at this time, a HEX-format file (*.hxb) for the boot area and a HEX-format file (*.hxf) for the flash area can be separately output.

APPENDIX C LIST OF OPTIONS

In this chapter, the program options are listed.

Please refer to these when developing programs.

This list of options can also be used as an index.

C.1 Assembler Options

Table C-1 Assembler Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
Device type specification	`-c`*device-type*	Specifies the device type of the target device.	Independent	Cannot be omitted
Object module file output specification	`-o[`*output-file-name*`]`	Specifies the output of an object module file.	If both options -o and -no are specified at the same time, the option specified last takes precedence.	-o*input-file-name*.rel
	`-no`	Specifies that no object module file is output.		
Forced object module file output specification	`-j`	Specifies that the object module file can be output even if a fatal error occurs.	If both options -j and -nj are specified at the same time, the option specified last takes precedence.	-nj
	`-nj`	Makes the -j option unavailable.		
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
	`-ng`	Makes the -g option unavailable.		
	`-ga`	Specifies that source debugging data is to be added to an object module file by the structured assembler.	If both options -ga and -nga are specified at the same time, the option specified last takes precedence.	-ga
	`-nga`	Makes the -ga option unavailable.		
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 where the source file exists. Path specified by the environmental variable “INC78K0R”
Assemble list file output specification	`-p[`*output-file-name*`]`	Specifies output of an assemble list file. It also specifies the destination and file name of the output file.	If both options -p and -np are specified at the same time, the option specified last takes precedence.	-p*input-file-name*.prn
	`-np`	Makes the -p option unavailable.		

Table C-1 Assembler Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
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 the -nka, -nks and -nkx options are all specified, the -p option is unavailable.	-ka
	`-nka`	Makes the -ka option unavailable.		
	`-ks`	Outputs an assemble list followed by a symbol list into an assemble list file.		-nks
	`-nks`	Makes the -ks option unavailable.		
	`-kx`	Outputs an assemble list followed by a cross-reference list into an assemble list file.		-nkx
	`-nkx`	Makes the -kx option unavailable.		
Assemble list file format specification	`-lw[`*number-of-characters*`]`	Changes the number of characters that can be printed in 1 line in a list file.	If the -np option is specified, the -lw option is unavailable.	-lw132
	`-ll[`*number-of-lines*`]`	Changes the number of lines that can be printed in 1 page in an assemble list file.	If the -np option is specified, the -ll option is unavailable.	-ll0
	`-lh`*character-string*	Specifies the character string printed in the title column of the header of an assemble list file.	If the -np option is specified, the -lh option is unavailable.	None
	`-lt[`*number-of-characters*`]`	Specifies a number of characters to be developed in a tab.	If the -np option is specified, the -lt option is unavailable.	-lt8
	`-lf`	Inserts a form feed (FF) code at the end of an assemble list file.	If both options -lf and -nlf are specified at the same time, the option specified last takes precedence. If the -np option is specified, the -lf option is unavailable.	-nlf
	`-nlf`	Makes the the -lf option unavailable.		
Error list file output specification	`-e[`*output-file-name*`]`	Outputs an error list file.	If both options -e and -ne are specified at the same time, the option specified last takes precedence.	-ne
	`-ne`	Makes the the -e option unavailable.		

Table C-1 Assembler Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
Parameter file specification	`-f`*file-name*	Inputs assembler options and the input file name from a specified file.	Independent	Options and input file names can only be input from the command line.
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'
Kanji code (2-byte 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.	-zs
	`-ze`	Kanji described in the comment is interpreted as EUC code.		
	`-zn`	Characters described in the comment are not interpreted as kanji.		
Device file search path specification	`-y`*path-name*	Reads a device file from the specified path.	Independent	Note
Symbol definition specification	`-d`*symbol-name*`[=`*value*`][,`*symbol-name*`[=`*value*`]...]`	Defines a symbol.	Independent	None
78K0R common object specification	`-common`	Specifies output of an object module file common to the 78K0R.	Independent	The object file supporting the specified device is output.
Self programming specification	`-self`	Specifies when using self programming	Independent	None
78K0 compatible macro	`-compati`	Enables assembly of assembler source files generated by the 78K0 assembler.	Independent	-ncompati
Help specification	`--`	Displays a help message on the display.	When the `--` option is specified, all other options are unavailable.	No display

Note Device files will be read from the path determined in the following order.

(1) Path registered in the device file installer

(2) Path by which RA78K0R was started up

(3) Current folder

(4) The environmental variable PATH

C.2 List of Linker Options

Table C-2 List of Linker Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
Load module file output specification	-o[*output-file-name*]	Outputs a load module file.	If both options -o and -no are specified at the same time, the option specified last takes precedence.	-o*input-file-name*.lmf
	-no	Does not output a load module file.		
Forced load module file output specification	-j	Forces output of a load module file.	If both options -j and -nj are specified at the same time, the option specified last takes precedence.	-nj
	-nj	Makes the -j option unavailable.		
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 the -ng option is specified, the public symbol list and local symbol list cannot be output regardless of specification of -kp or -kl.	-g
	-ng	Makes the -g option unavailable.		
Generation of stack decision symbols specification	-s[*area-name*]	Automatically generates stack decision public symbols.	If both options -s and -ns are specified at the same time, the option specified last takes precedence.	-ns
	-ns	Makes the -s option unavailable.		
Directive file specification	-d*file-name*	Specifies a particular file to be input as a directive file.	Independent	None
Link list file output specification	-p[*output-file-name*]	Specifies output of a link list file.	If both options -p and -np are specified at the same time, the option specified last takes precedence.	-p*input-file-name*.map
	-np	Makes the -p option unavailable.		

Table C-2 List of Linker Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
Link list file data specification	-km	Outputs a map list into a link list file.	If both options -km and -nkm are specified at the same time, the option specified last takes precedence.	-km
	-nkm	Makes the -km option unavailable.	If the -nkm, -nkp and -nkl options are all specified, the link list file cannot be output even if the -p option is specified.	
	-kd	Outputs a link directive file into a link list file.	If the -nkm option is specified, the -kd option 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 the -ng option is specified, the public symbol list and local symbol list cannot be output even if the -kp or -kl option is specified.	-kd
	-nkd	Makes the -kd option unavailable.		
	-kp	Outputs a public symbol list into a link list file.		-nkp
	-nkp	Makes the -kp option unavailable.		
	-kl	Output a local symbol list into a link list file.		-nkl
	-nkl	Makes the -kl option unavailable.		
Link list format specification	-ll[*number-of-lines*]	Specifies number of lines that can be printed in 1 page in a link list file.	If the -np option is specified, the -ll option is unavailable.	-ll0
	-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 the -np option is specified, the option specified last takes precedence.	-nlf
	-nlf	Makes the -lf option unavailable.		
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
	-ne	Default value: -ne Makes the -e option unavailable.		
Library file specification	-b*file-name*	Inputs a specific file as a library file.	Independent	None

Table C-2 List of Linker Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
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 the -b option, the -i option is unavailable.	Path specified by environmental variable 'LIB78K0R'
Parameter file specification	-f*file-name*	Inputs linker options and the input file name from a specified file.	Independent	Options and input file names can only be input from the command line.
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'.
Device file search path specification	-y*path-name*	Reads a device file from the specified path.	Independent	Note
Warning message output specification	-w[*level*]	Specifies whether or not a warning message is output to the console.	Independent	-w1
Link specification of boot area ROM program of flash memory model	-zb	Specifies the first address of the flash memory area.	None	No limitation for the location range
On-chip debug option byte specification	-go*control-value*,*start-address*[,*size*]	Specifies the on-chip debug option byte.	None	Address C3H is the initial value specified the device file.
Security ID specification	-gi*security-ID*	Specifies a Security ID.	None	A security ID is not set.
User option byte specification	-gb*user-option-byte-value*	Specifies the value set for the user option byte.	None	Initial value set in the device file.
Mirror area specification	-mi0 or -mi1	Specifies the location destinations of segments in the mirrored area.	None	-mi0

Table C-2 List of Linker Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
64 KB boundary location specification	-ccza	Specifies whether to locate a segment to the last byte of each 64 KB boundary area.	None	-ccza (when input files are assembler output files only) -nccza (when compiler output file is included in input files)
	-nccza	Makes the -ccza option unavailable.	None	
Help specification	--	Displays a help message on the display.	All other options are unavailable.	No display

Note Device files will be read from the path determined in the following order.

(1) Path registered in the device file installer

(2) Path by which LK78K0R was started up

(3) Current folder

(4) The environmental variable PATH

C.3 List of Object Converter Options

Table C-3 List of Object Converter Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
HEX format object module file output specification	-o[*output-file-name*]	Outputs a HEX format object module file.	If both options -o and -no are specified at the same time, the option specified last takes precedence.	-o*input-file-name*.HEX
	-no	No HEX format object module file is output.		
Symbol table file output specification	-s[*output-file-name*]	Outputs a symbol table file.	If both options -s and -ns are specified at the same time, the option specified last takes precedence.	-s*input-file-name*.sym
	-ns	Does not output a symbol table file.		
Specification of sort by object address order	-r	Sorts HEX format objects in order of address.	If both options -s and -ns are specified at the same time, the option specified last takes precedence. If the -no option is specified, the -r option becomes unavailable.	-r
	-nr	Makes the -r option unavailable.		
Object complement specification	-u*complement-value*[,[*start-address*],*size*]	Outputs a specified complement value as an object code for an address area to which no HEX format object has been output.	If -u and -nu are specified at the same time, the one specified later takes precedence. If the -no option is specified, -u becomes unavailable.	-u0FFH
	-nu			
Error list file output specification	-e[*output-file-name*]	Outputs an error list file.	If both options -e and -ne are specified at the same time, the option specified last takes precedence.	-ne
	-ne	Makes the -e option unavailable.		
Parameter file specification	-f*file-name*	Inputs options and input file names from a specified file.	Independent	Options or input file names can only be input from the command line.
HEX format specification	-ki	Intel standard HEX format	Independent	-kie
	-kie	Intel extended HEX format		
	-kt	Extended Tech format		
	-km	Motorola S-type format (standard address)		
	-kme	Motorola S-type format (32-bit address)		

Table C-3 List of Object Converter Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
Device file search path specification	`-ypath-name`	Reads a device file from the specified path.	Independent	Note
File separate output specification for flash memory model	`-zf`	Adds an option that separately outputs the boot area and other areas to separate HEX format files when linking of the boot area ROM program of a flash memory model is specified.	Independent	Not separately output.
Help specification	`--`	Displays a help message on the display.	All other options are unavailable.	No display

Note Device files will be read from the path determined in the following order.

(1) Path registered in the device file installer

(2) Path by which OC78K0R was started up

(3) Current folder

(4) The environmental variable PATH

C.4 List of Librarian Options

Table C-4 List of Librarian Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
List file format specification	-lw[*number-of-characters*]	Changes the number of characters that can be printed in 1 line in a list file.	Unavailable if the list subcommand is not specified.	-lw132
	-ll[*number-of-lines*]	Changes the number of lines that can be printed in 1 page in a list file.		-ll0
	-lf	Inserts a form feed (FF) code at the end of a list file.	If both options -lf and -nlf are specified at the same time, the option specified last takes precedence.	-nlf
	-nlf	Makes the the -lf option unavailable.		
Specification of path for temporary file creation	-t*path-name*	Creates a temporary file in a specified path.	Independent	Created in the path specified by the environmental variable 'TMP'.
Help specification	--	Displays a help message on the display.	All other options are unavailable.	No display

Note Device files will be read from the path determined in the following order.

(1) Path registered in the device file installer

(2) Path by which LB78K0R was started up

(3) Current folder

(4) The environmental variable PATH

C.5 List of List Converter Options

Table C-5 List of List Converter Options

Classification	Format	Function	Relation to Other Options	Interpretation When Omitted
Object module file input specification	`-r[`*`input-file-name`*`]`	Specifies the input of an object module file.	Independent	-r*assemble-list-file-name*.rel
Load module file input specification	`-l[`*`input-file-name`*`]`	Inputs a load module file.	Independent	-l*assemble-list-file-name*.lmf
Absolute assemble list file output specification	`-o[`*`output-file-name`*`]`	Outputs an absolute assemble list file.	Independent	-o*assemble-list-file-name*.p
Error list file output specification	`-e[`*`output-file-name`*`]`	Outputs an error list file.	If both options -e and -ne are specified at the same time, the option specified last takes precedence.	-ne
	`-ne`	Makes the -e option unavailable.		
Parameter file specification	`-f`*`file-name`*	Inputs options and input file name from a specified file.	Independent	Options or input file names can only be input from the command line.
Help specification	`--`	Displays a help message on the display (console).	All other options are unavailable.	No display

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.

Table D-1 List of Subcommands

Classification	Format	Function	Abbrev. Format
create	`create`Δ*`library-file-name`*[Δ*`transaction`*]	Creates a new library file.	c
add	`add`Δ*`library-file-name`*Δ*`transaction`*	Adds a module to a library file.	a
delete	`delete`Δ*`library-file-name`*Δ(Δ*`module-name`*[Δ, ...]Δ)	Deletes a module from a library file.	d
replace	`replace`Δ*`library-file-name`*Δ*`transaction`*	Replaces one module with another in a library file.	r
pick	`pick`Δ*`library-file-name`*Δ(Δ*`module-name`*[Δ, ...]Δ)	Retrieves a specified module from an existing library file.	p
list	`list`[Δ*`option`*]Δ*`library-file-name`*[Δ(Δ*`module-name`*[Δ, ...]Δ)]	Outputs data on modules in a library file.	l
help	`help`	Displays a help message on the display.	h
exit	`exit`	Exits the librarian.	e

INDEX

Y

Z

For further information,
please contact:

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

Hanover Office
Podbielskistrasse 166 B
30177 Hannover
Tel: 0 511 33 40 2-0

Munich Office
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

Shanghai Branch
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai, P.R.China P.C:200120
Tel:021-5888-5400
http://www.cn.necel.com/

Shenzhen Branch
Unit 01, 39/F, Excellence Times Square Building,
No. 4068 Yi Tian Road, Futian District, Shenzhen,
P.R.China P.C:518048
Tel:0755-8282-9800
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
Unit 1601-1613, 16/F., Tower 2, Grand Century Place,
193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737
http://www.kr.necel.com/

G0706