NEC

User's Manual

RA75X ASSEMBLER PACKAGE

PC-9800 Series (MS-DOS™) Based
IBM PC/AT™ (PC DOS™) Based
Version 5.xx Operation

Target Devices
75X Series
75XL Series

Document No. U12622EJ6V0UM00 (6th edition)
(O.D.No. EEU-1346C)
Date Published September 1997 N

[MEMO]

Regional Information

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

J96. 8

Major Revisions in This Edition

Page	Description
General	RA75X Assembler Package Version 4.5X changed to Version 5.XX
General	Programs included in the RA75X Assembler Package were changed.
	• A macro function added to the assembler program. Accordingly, the macro processor was eliminated. • A library converter program was added.
General	Target devices were added: μPD750064, 750066, 750068, 75P0076, 750104, 750106, 750108, 75P0116, 753012A, 753016A, 753017A, 75P3018A, 753036, 75P3036, 753204, 753206, 753208, 75P3216, 753304[Note], 754202, 754144, 754244, 754264, 75F4264[Note], 754302, 754304, 75P4308
General	Development of target devices which were under development was completed. μPD750004, 750006, 750008, 75P0016, 753012, 753016, 753017, 75P3018, 753104, 753106, 753108, 75P3116
p.36	**1.2.8 Library converter** was added.
p.38	**1.3.4 Other limitation items** was added.
p.44	**Table 2-2 List of Host Machine Applicable Models (PC-9800 Series)** was changed.
p.46	Applicable OS were added and versions have been changed: **2.2.1 (2) PC-9800 series OS** **2.2.2 (2) IBM PC/AT OS**
p.46	Assembler Package 5 inch FD was eliminated. **2.2.1 (3)** and **2.2.2 (3) Assembler package supply media**
p.48	**3.1.1 Assembler package installation procedure** was changed.
p.94, 102-197, 117	Assembler option types were added: **4.4.4 (6) -GA/-NGA, (11) -CA/-NCA, (12) -S/-NS, (13) -D/-ND, (19) -Y**
p.180	A linker option type was added: **5.4.4 (15) -Y**
p.194, 197-200	Object Converter option types was added: **6.4.4 (2) -R/-NR, (4) -O/-NO, (5) -E/-NE, (6) -F, (7) -Y**
p.249-251	List Converter option types was added: **8.4.2 (3) -R, (4) -E/-NE, (5) -F**
p.253	**CHAPTER 9 LIBRARY CONVERTER** was added.
p.259	**CHAPTER 10 OPTION SETTINGS FROM PROJECT MANAGER** was added.
p.273	**11.1.1 Assembly list** was changed.
p.322	**13.6 Library Converter Error Messages** was added.
p.327	**A.6 List of Library Converter Options** was added.

Note Under development

The mark ★ shows major revised points.

[MEMO]

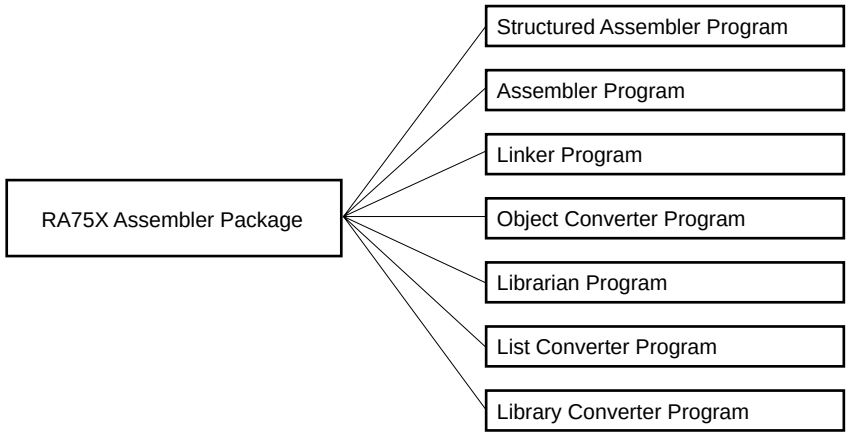
PREFACE

The purpose of this manual is to give users a clear understanding of the function and operation method of each of the programs comprising the RA75X Assembler Package (referred to in the text simply as “assembler package”).

This manual does not cover source program writing: the **RA75X Assembler Package User’s Manual — Language** (**Document No. U12385E**, referred to in the text simply as “**language**”) should therefore be read this manual.

This manual applies to assembler package version 5.XX products.

★



[Readers]

This manual is intended for users who are familiar with the functions of and method of writing source programs for the microcomputer (75X series and 75XL series) subject to development.

[Target devices]

The assembler package can be used for software development on the microcomputers shown below (75X Series).

<75X Series>

Series	Title	Target Devices
—	Evaluation chip	μPD75000, 75000A
General-purpose series	General purpose	μPD75004, 75006, 75008, 75P008
	General purpose, + A/D converter	μPD75028, 75036, 75P036, 75064, PD75066, 75068, 75P068
	General purpose + A/D converter + EEPROM™	μPD75048, 75P048
Control series	Control	μPD75104, 75106, 75108, 75112, 75116, 75104A, μPD75108A, 75P108, 75P108B,75P116
	Low-voltage high-speed control	μPD75108F, 75112F, 75116F
	F product + low voltage capability	μPD75116H, 75117H, 75P117H
FIP drive series	FIP™ drive	μPD75206, 75208, 75212A, 75216A, 75217, 75218, μPD75P216A, 75P218, 75268, 75CG208, 75CG216A
	FIP drive + A/D converter	μPD75236, 75237, 75238, 75P238
LCD drive series	LCD drive	μPD75304, 75306, 75308, 75304B, 75306B, 75308B, μPD75312, 75316, PD75312B, 75316B, 75P308, μPD75P316, 75P316A, 75P316B
	LCD drive + A/D converter	μPD75238, 75P328
	LCD drive + A/D converter + high-level functions	μPD75336, 75P336
Slave series		μPD75402A, 75P402
Control (on-chip A/D converter) series	Control (on-chip A/D converter)	μPD75512, 75516, 75P516
	Control (on-chip A/D converter) + high speed	μPD75517, 75518, 75P518
Telephone series	LCD drive + DTMF + D/A converter	μPD75352A
	LCD drive + DTMF + D/A converter + A/D converter	μPD75617A

★ **<75XL Series>**

Series	Title	Target Devices
General-purpose series	General purpose	μPD750004, 750006, 750008, 75P0016
	General purpose, + RC oscillator circuit	μPD750104, 750106, 750108, 75P0116
	General purpose, + A/D converter	μPD750064, 750066, 750068, 75P0076
LCD drive series	LCD drive	μPD753012, 753012A, 753106, 753016A, 753017, 753017A, 75P3018, 75P3018A
	LCD drive + A/D converter	μPD753036, 75P3036
	For driving a LCD (small)	μPD753104, 753106, 753108, 75P3116
		μPD753204, 753206, 753208, 75P3216
	LCD drive + RC oscillator circuit (small)	μPD753304Note
Keyless entry		μPD754202
		μPD754144, 754244, 754264 μPD75F4264Note
General purpose small		μPD74302, 754304, 75P4308

[Organization]

The configuration of this manual is shown below.

CHAPTER 1. GENERAL DESCRIPTION

Describes the role of the assembler package in microcomputer development, etc., together with a general outline of its functions.

CHAPTER 2. PRODUCT SUMMARY

Gives the file names of the programs provided in the assembler package, and describes the program operating environment, etc.

CHAPTER 3. ASSEMBLER PACKAGE EXECUTION

Describes the actual execution procedure for each of the programs in the assembler package, using sample programs .

CHAPTER 4. ASSEMBLER, CHAPTER 5 LINKER, CHAPTER 6 OBJECT CONVERTER, CHAPTER 7 LIBRARIAN

★ **CHAPTER 8. LIST CONVERTER, CHAPTER 9 LIBRARY CONVERTER**

Explain in detail the functions of each program in the assembler package (assembler, linker, object converter, librarian, list converter, library converter) and how to run them.

★ **CHAPTER 10. SETTING OPTIONS FROM THE PROJECT MANAGER**

Explains setting of options from the Project Manager, used when running the Assembler Package in Windows™.

CHAPTER 11. PROGRAM OUTPUT LISTS

Describes the format of the various lists output by the assembler package programs.

CHAPTER 12. EFFECTIVE USE OF THE ASSEMBLER PACKAGE

Introduces way of using the assembler package effectively.

CHAPTER 13. ERROR MESSAGES

Describes the error messages output by the assembler package programs.

APPENDIX

The appendices include a list of program options, sample program list, maximum performance table, and list of points for attention.

[Reading the manual]

Readers wishing to actually use the assembler package should read **Chapter 3 "Assembler Package Execution".** Readers who have a general understanding of assemblers of who have already read the **RA75X Assembler Package User's Manual Language Volume can skip Chapter 1.**

Appendix A "List of Options" can be used once the reader is familiar with the operation of each program.

[Legend]

The meanings of symbols and abbreviations used throughout this manual are shown below.

… : Continuation of same format
[ ] : Item in square brackets may be omitted
‘ ’ : Actual characters enclosed in ‘ ’
“ ” : Actual characters enclosed in “ ”

Characters in heavy type: The characters themselves

— : Important item (Underlining in examples indicates input characters)
␣ : One or more spaces
: Abbreviated form of program description
( ) : Actual characters enclosed in ( )
/ : Delimiter symbol
CR : Carriage Return
LF : Line Feed
↵ : Line feed key input
● - ▲ : From ● to ▲

[File Name Rules]

(1) Disk type file specification

[Drive name:] [\] [directory name\] primary name [.file type]

- Max. 3 characters
- Max. 8 characters
- Specifies name of directory containing file. If omitted, the current directory is assumed to be specified.
- Specifies name of drive containing file. If omitted, the current drive is assumed to be specified.

Remarks 1. Spaces cannot be specified before or after ':', '.', and '\'.

2. No distinction is made between upper- and lower-case characters.

3. If the file type is omitted, the following default values are used for each type of file.

File	Default File Type
Source program file	.ASM
Object module file	.REL
Load module file	.LNK
HEX format object file	.HEX
Assemble list file	.PRN
Link map file	.MAP
Symbol table file	.SYM
Error list file	.ERA
Library file	.LIB
★ Library Converter Output Library File	.CNV
Library list file	.LST
Absolute assembly list file	.P
Assembler parameters file	.PRA
Linker parameter file	.PLK

4. When a file without a file type is specified, 'file name.'

Example:
```
B>RA75X ABC ; In this time, the file name is interpreted 'ABC.ASM'.
B>RA75X ABC.; In this time, the file name is interpreted 'ABC'.
```

5. Not only an absolute path [Note 1] but also a relative path [Note 2] can be used in the directory name.

Notes 1. Using the root directory ('\') as the starting point

Example: '\YBIN\75X', etc.

2. Using the current directory or parent directory ('. .') as the starting point

Example: 'SRC\HDR', '.. \DOC\75X', etc.

6. Directories are not supported for libraries.

(2) Device type file specification

The following can be specified as logical devices .

CON ... Console (output: CRT, input : Keyboard)
PRN ... Line printer (output only)
AUX ... RS-232-C
NUL ... Null device

However, in some cases these cannot be specified or a meaningless result is obtained if specified. This is explained in the text.

Remark A ‘CLOCK’ device cannot be specified.

[Related documents]

Documentation related to this manual is shown below.

Document Title	Document No.
RA75X Assembler Package User’s Manual Version 4.5X - Language Volume	U12385E
RA75X Structured Assembler Preprocessor User’s Manual	U12598E
RA75X Structured Assembler Preprocessor	EEA-1203

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. GENERAL DESCRIPTION

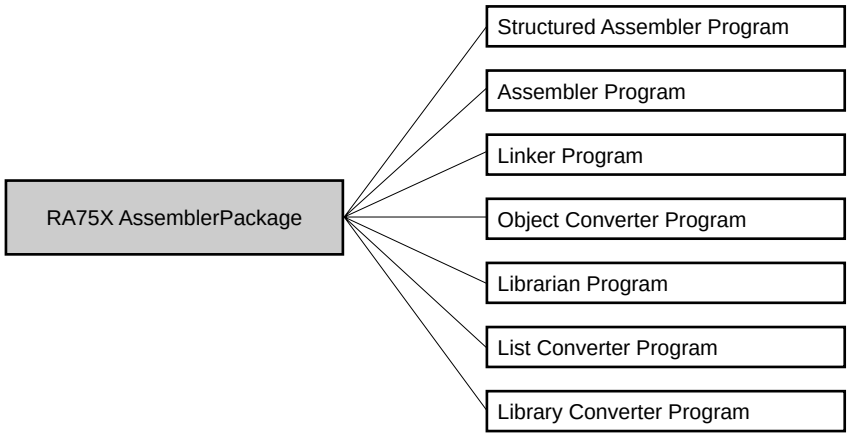
Describes the role of the assembler package in 75X Series and 75XL Series development, etc., together with a general outline of its functions.

1.1 Outline of Assembler

The RA75X Assembler Package (referred to in the following text simply as “assembler package”) is the generic name for a series of programs for converting source programs written in 75X Series and 75XL Series assembly language into machine language.

The Assembler Package includes 7 programs, Structured Assembler, Assembler, Linker, Object Converter, Librarian, List converter and Library Converter.

Figure 1-1 RA75X Assembler Package



1.1.1 The function of an assembler

(1) Assembly language and machine language

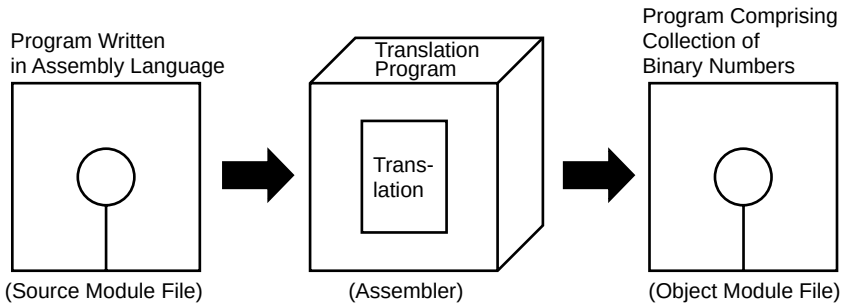
Assembler language is the most basic programming language for microprocessors.

To operate a microprocessor, a program and data are necessary. This is programmed by a human being and memorized in the memory of the microcomputer. Programs and data which a microcomputer can handle are collections of binary data; this is known as machine language (computer-comprehensible language) .

Programming in machine language, that is in binary code, is difficult for a human mind to learn and prone to errors: hence the method of representing machine language in easily understood English symbols and writing programs using these symbols. The language system for programming using these symbols is called assembly language.

A program is needed to translate a program written in assembly language into a collection of binary numbers intelligible to a microprocessor. This program is called an assembler.

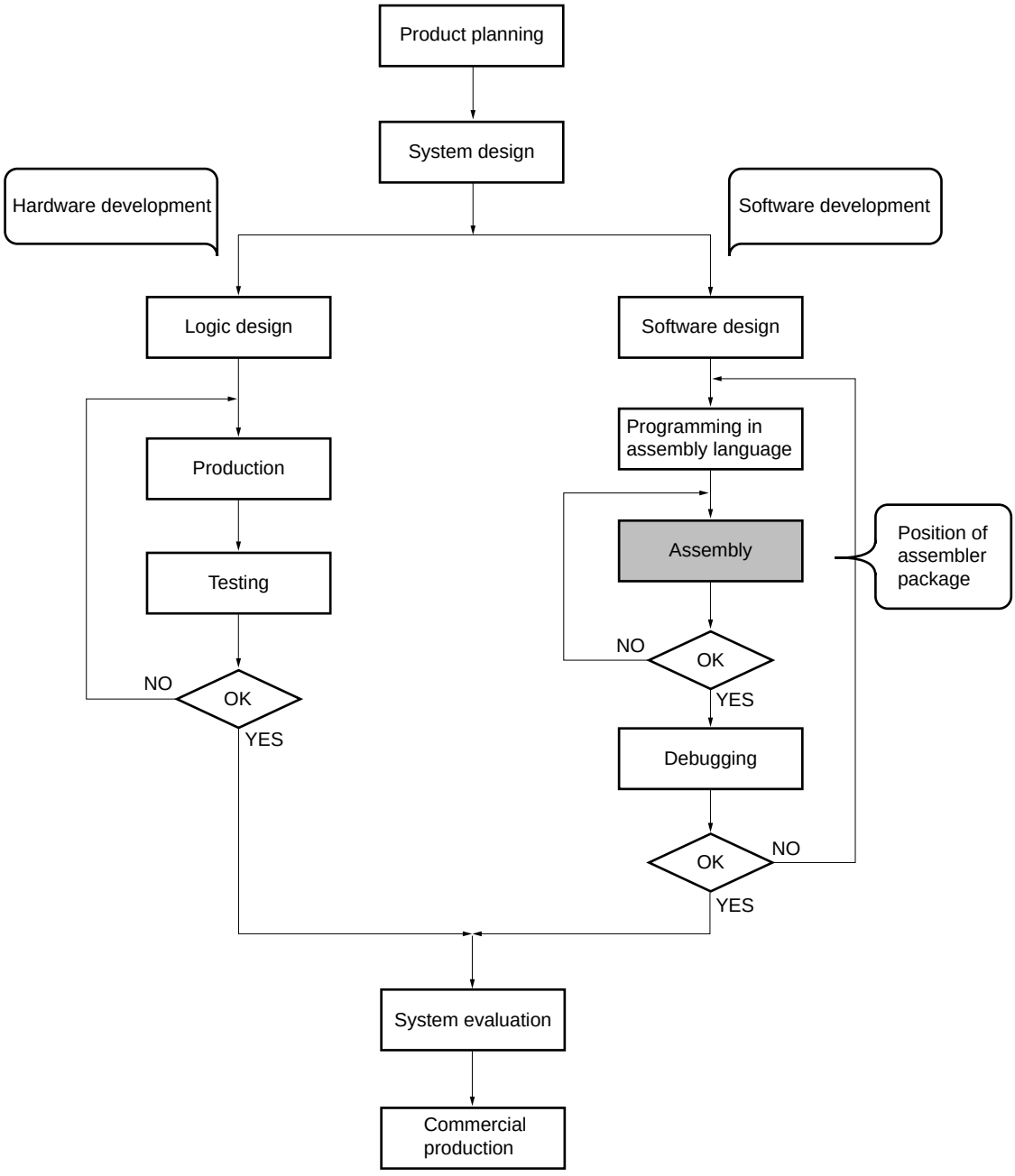
Figure 1-2 Assembler Flow



(2) The role of this package in developing products using microcomputers

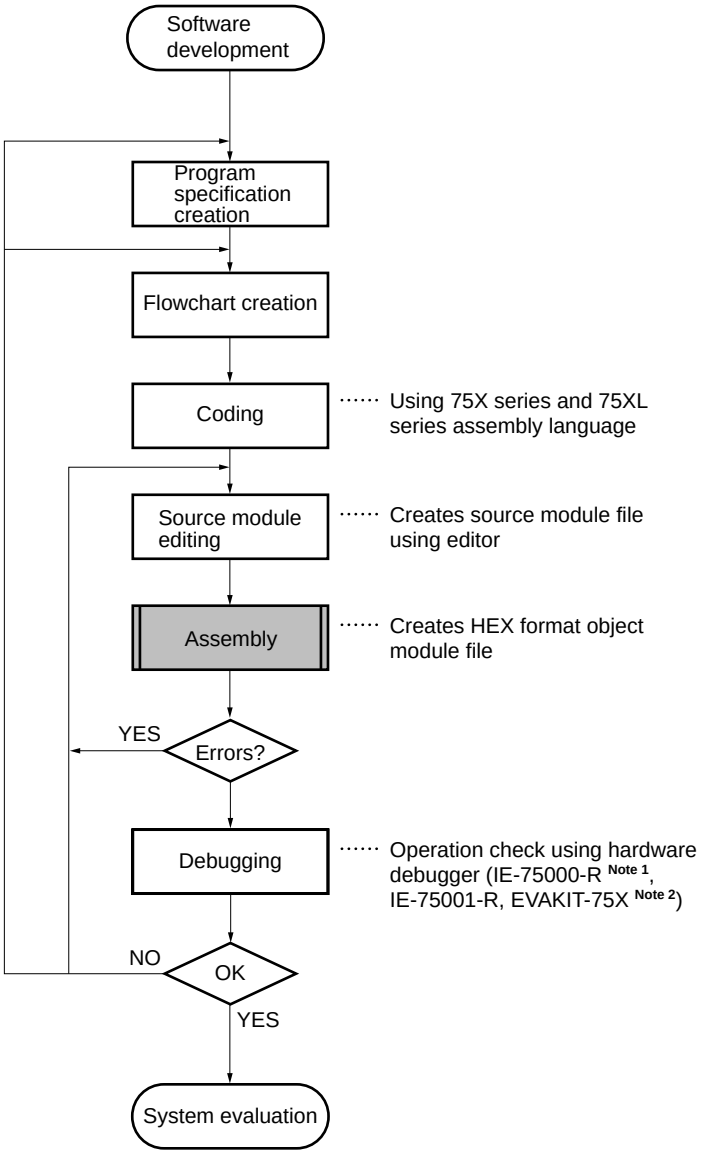
Where assembly language programming fits into product development is outlined in **Figure 1-3 "Development Process for Products Using Microcomputers"**.

Figure 1-3 Development Process for Products Using Microcomputers



The software development process is described in somewhat more detail by **Figure 1-4 “Software Development Process”.**

Figure 1-4 Software Development Process

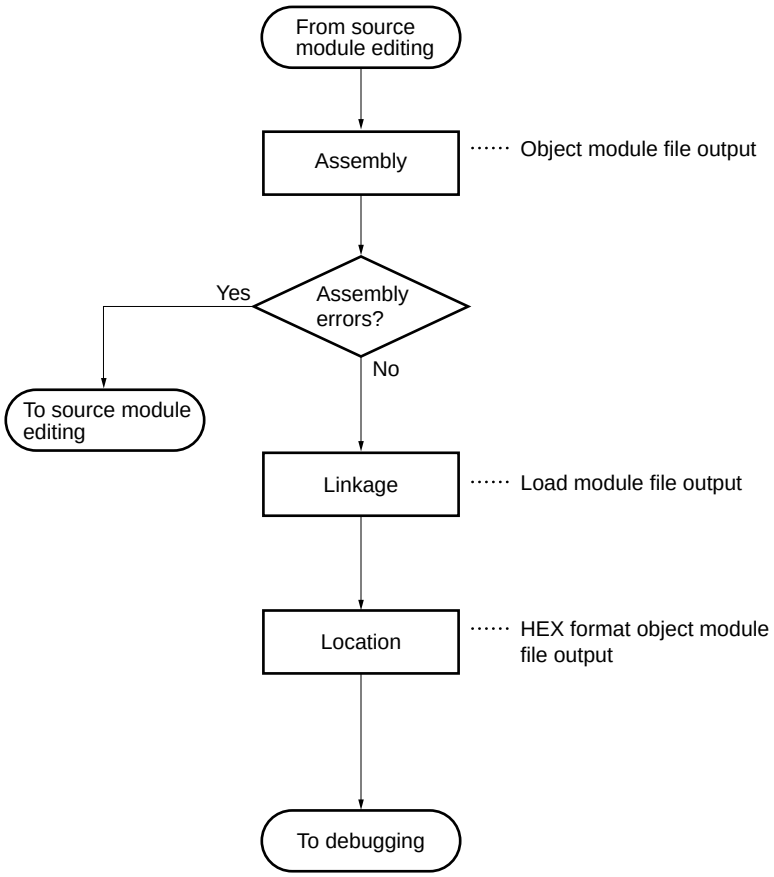


Notes 1. Maintenance product (not available for purchase)

2. Discontinued (not available for purchase)

We shall now apply the assembler package to the assembly process.

Figure 1-5 Assembly Process of This Package



Remarks As necessary, use the structured assembler, preprocessor, librarian, list converter, or library converter.

1. Structured assembler preprocessor
Program for implementing structured programming in assembly language.

2. Librarian
It is convenient to “librarize” general-purpose modules with a clear interface by means of the librarian. Librarization enables a large number of object modules to be handled easily as a single file.

3. List converter
The list converter is used to create an assembly list which incorporates absolute values for debugging purposes.

★ **4.** Library Converter
Library files which can be input to the Linker and Librarian in Assembler Package Version 5.00 or later can be created from library files in the object program module format output by versions of Librarian before Version 5.00.

1.1.2 Function of a relocatable assembler

The machine language resulting from the assembler conversion process is written into the memory of the microcomputer before being used. To do this, it is first necessary to determine where in memory the machine language is to be written. Therefore the machine language produced by the assembler's conversion has, attached to it information indicating in which address in memory it should be located.

Assemblers can be broadly classified into two kinds - absolute assemblers and relocatable assemblers - according to the method used to place machine language in memory address.

- Absolute assembler
  An absolute assembler locates machine language converted in a single assembly operation in absolute addresses.

- Relocatable assembler
  With a relocatable assembler, the addresses converted by the assembly process are only determined temporarily. Absolute address determination is performed by the program known as the linker.

When a program is created with an absolute assembler, programming must, in principle, be performed as a one-time operation. However, writing a large program at one time as a single entity results in a complex program and makes program analysis difficult when maintenance is required . In view of this, program development is carried out by dividing the program into a number of subprograms (modules), each with a specific function. This is known as modular programming.

A relocatable assembler is one which is suited to modular programming. The advantages of modular programming using a relocatable assembler are described below.

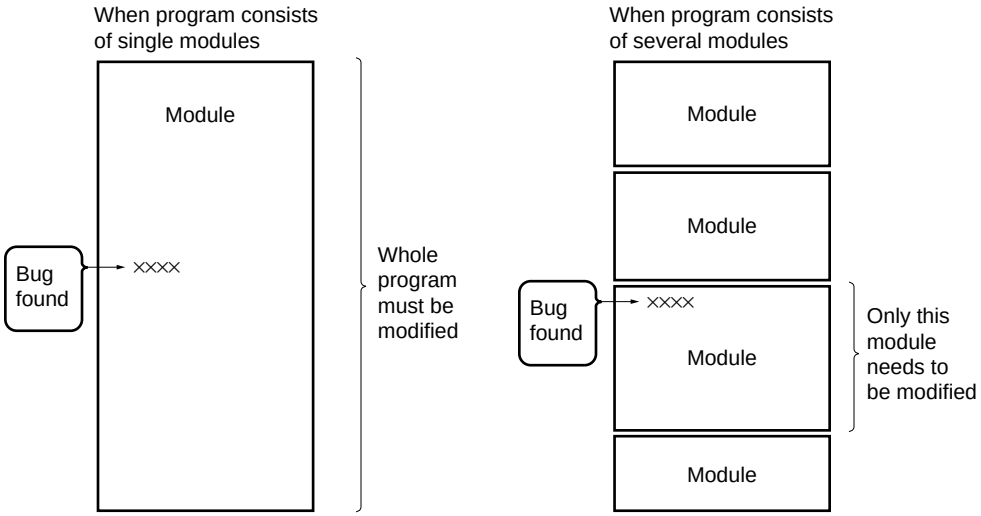
(1) Development is made more efficient

Writing a large program as a single unit is difficult.

Dividing a large program into modules allows a number of programs to be developed in parallel, making the process more efficient.

Also, when a bug is found it is not necessary to reassemble the whole program just to amend one party; only the module requiring correction needs to be reassembled. This enables the time required for debugging to be reduced.

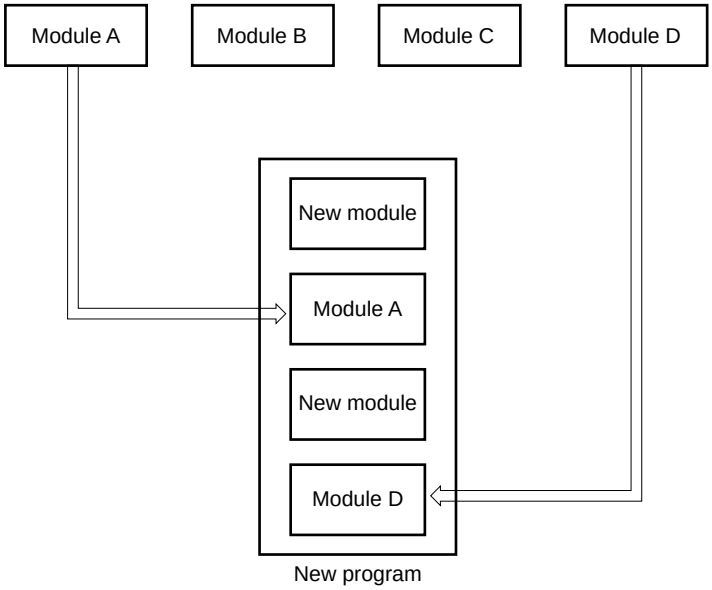
Figure 1-6 Reassembly Process



(2) Resources can be fully used

Highly reliable and generally applicable modules previously created can be used in the development of other programs. Accumulating a number of such generally applicable modules means that the proportion of new work needed on program development can be reduced .

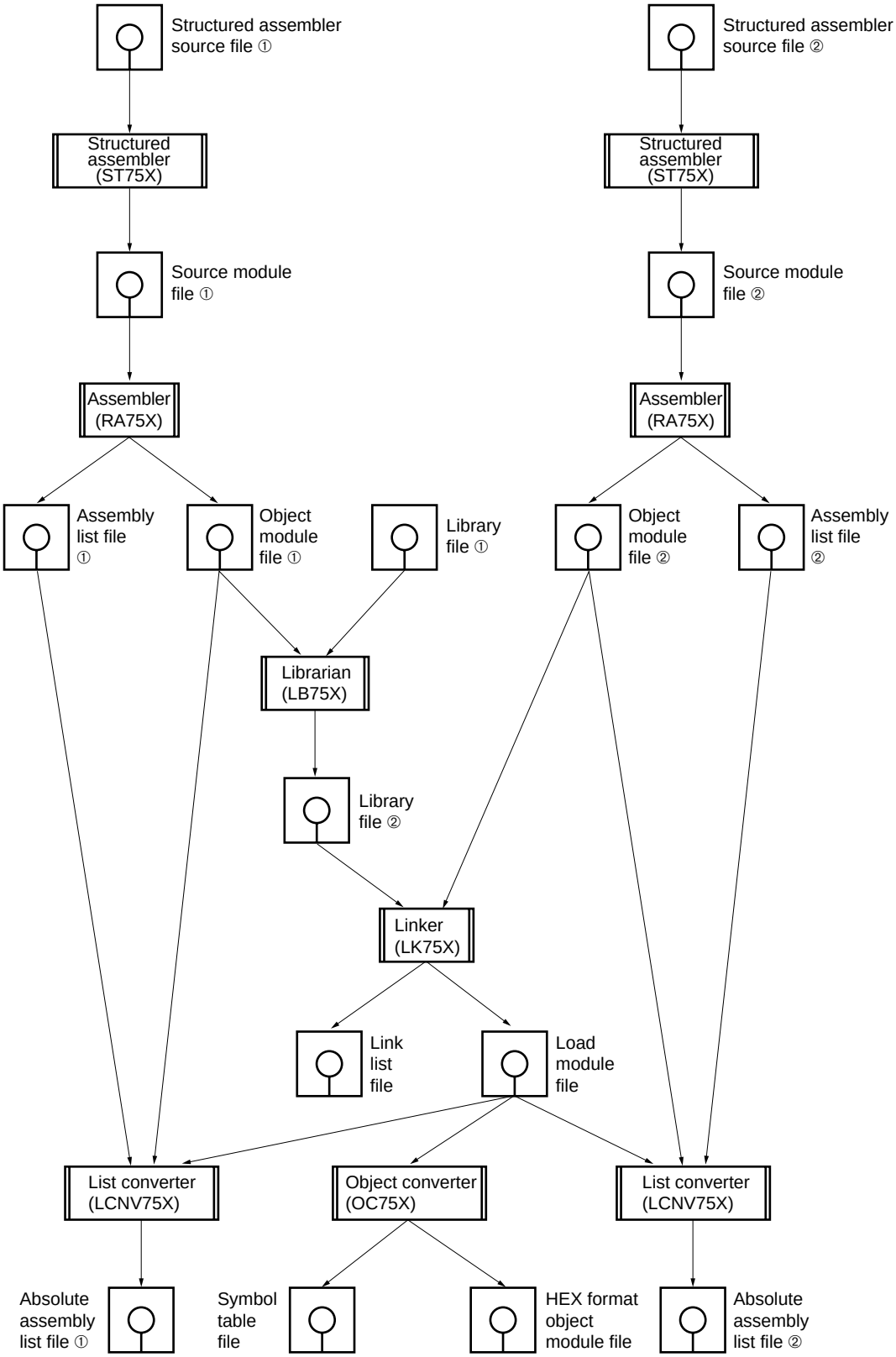
Figure 1-7 Program Creation Using Previously Written Modules



1.2 Outline of Assembler Package Functions

The general program development procedure using this package is shown in **Figure 1-8 "Program Development Procedure Using Assembler Package".** The basic procedure used in program development is: assembler → linker → object converter. Use the Structured Assembler, Librarian, List Converter or Library Converter as necessary.

Figure 1-8 Program Development Procedure Using Assembler Package



1.2.1 Source module file creation using editor

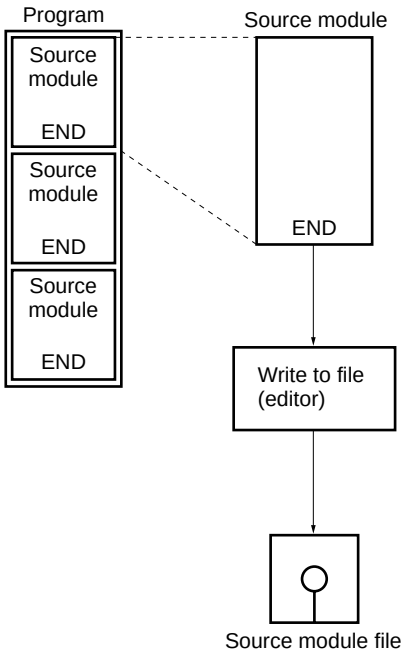
A single program is functionally divided into a number of modules.

A module is the unit of coding and the unit of input to the assembler. A module which is an assembler input unit is called a source module.

After coding of each source module is completed, the source module is input using the editor and written to a file. The file created in this way is called the source module file.

The source module file is the input file for the assembler (RA75X).

Figure 1-9 Source Module File Creation

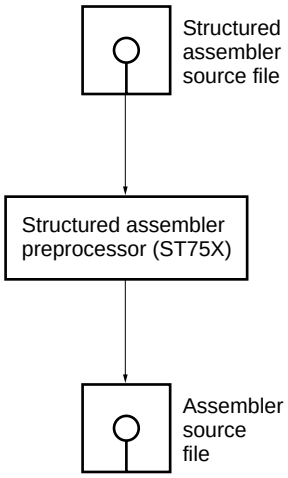


1.2.2 Structured assembler preprocessor

The structured assembler preprocessor is a program for implementing structured programming in assembly language. A source program written in assembly language is input, and an assembler source program is output.

Please refer to the separate volume **"RA75X Assembler Package Structured Assembler Preprocessor User's Manual"** (Document No. U12598E) for details of the structured assembler preprocessor and structured assembly language.

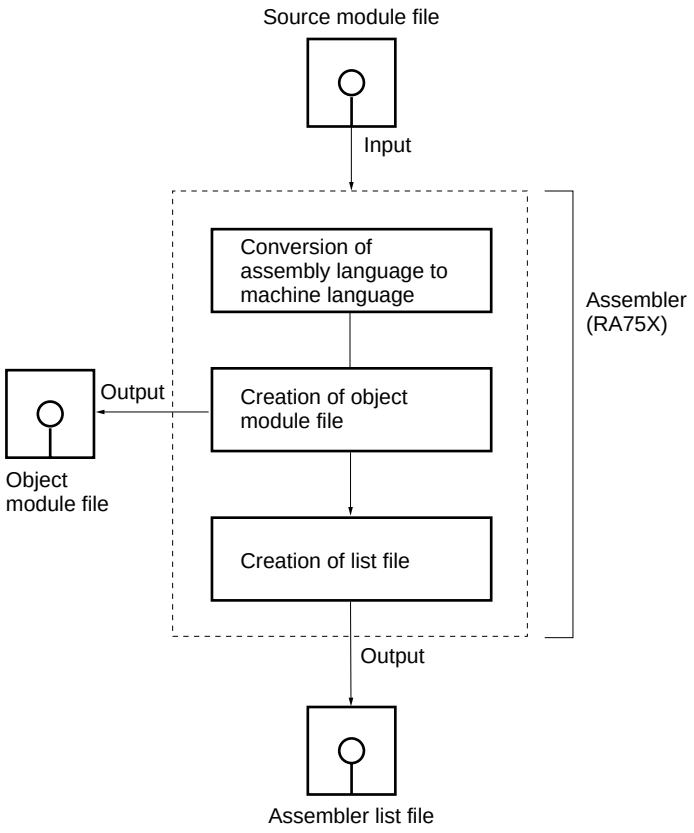
Figure 1-10 Structured Assembler Preprocessor Functions



1.2.3 Assembler

The assembler has a source module file as input, and converts the assembly language into machine language. If coding errors are found in the source module file, assembly errors are output. If there are no assembly errors, an object module file is output containing machine language information and relocation information relating to the relocatable machine language addresses. In addition, assembly-time information is output as an assembly list file.

Figure 1-11 Assembler Function

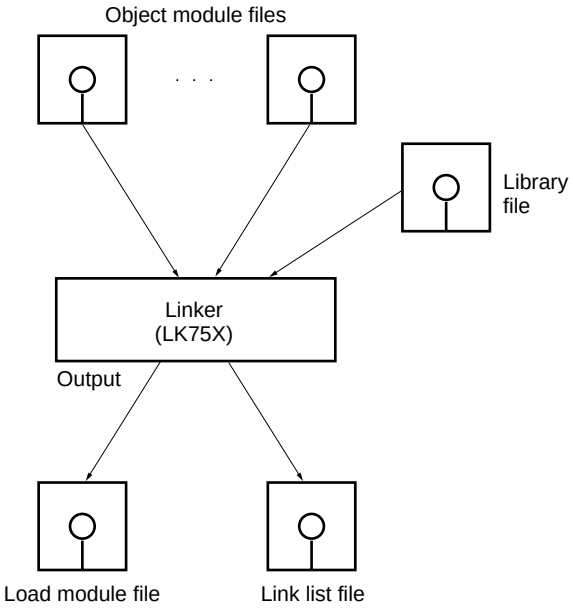


1.2.4 Linker

The linker has as input a number of object module files output by the assembler or a library file output by the librarian, links them, and outputs a single load module file. It also outputs link-time information as a link list file.

If a library file is specified, one only should be specified at the end of the input file names.

Figure 1-12 Linker Function



1.2.5 Object converter

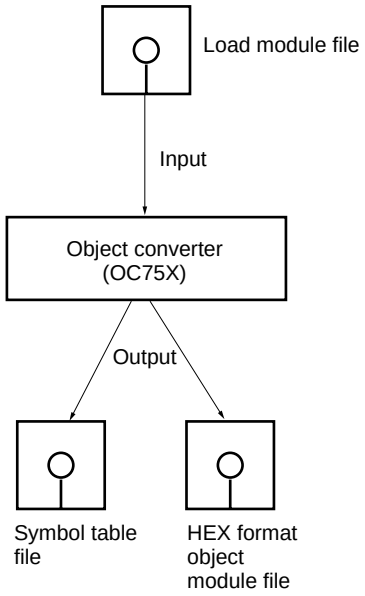
The object converter has an input the load module file output by the linker, and output the results as a HEX format object module file.

An object module file output by the assembler cannot be used as input to the object converter.

A HEX format object module file is necessary for ROM ordering and debugging input.

In addition, symbol information required for symbolic debugging by the debugger is output as a symbol table file.

Figure 1-13 Object Converter Function



The basic processing of the assembler package ends when processing has been completed normally as far as the object converter .

In addition, program development can be made more efficient by using the librarian and list converter.

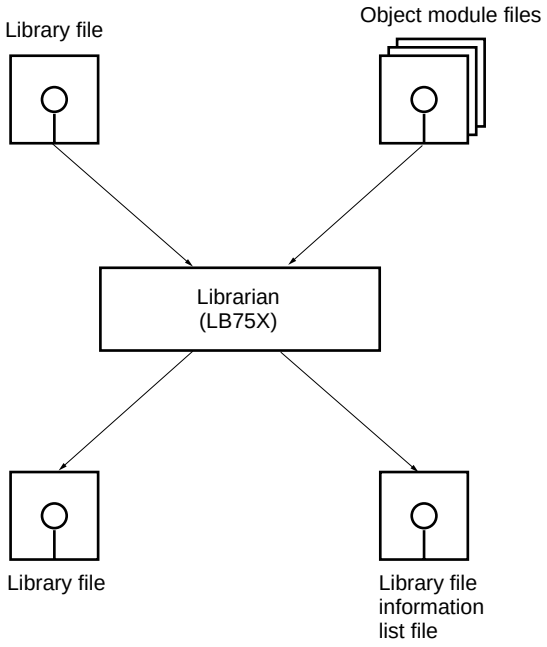
1.2.6 Librarian

The librarian is used to create or update library files.

It is convenient to “librarize” general-purpose modules with a clear interface, as this enables a large number of object modules to be handled easily as a single file.

The linker includes a function for extracting only the needed modules from the library file. Therefore, if a number of modules are recorded in a single library file, it is not necessary to specify the file name of each module required for linkage .

Figure 1-14 Librarian Function

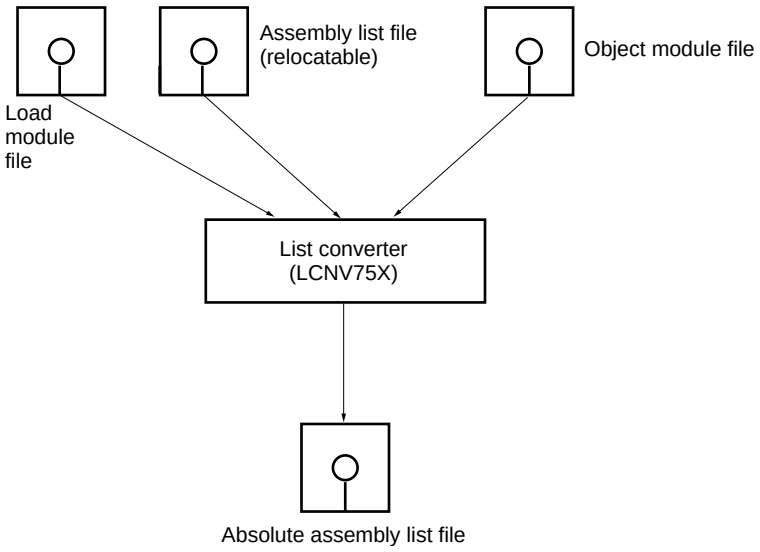


1.2.7 List converter

The list converter has as input the assembly list file of relocatably assembled modules, the object module file, and the load module file output by the linker, and incorporates actual address values and object code in a relocatable assembly list. It then outputs the results as an absolute assembly list.

The list converter is designed to enable relocatable assembled programs to be debugged more efficiently.

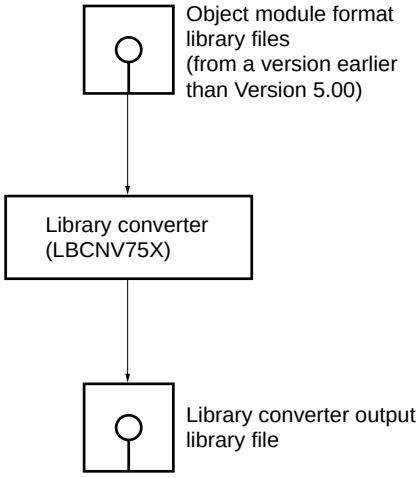
Figure 1-15 List Converter Function



★ 1.2.8 Library converter

The Library Converter outputs library files which can be input to Version 5.00 or subsequent versions of Linker and Librarian when object program module format library files output by the Librarian in a version of RA75X Assembler Package earlier than Version 5.00 are input.

Figure 1-16 Library Converter Functions



1.3 Before Beginning Program Development

The following restrictions should be noted before beginning actual program development.

1.3.1 Assembly-time source statements

Source statements up to 220 characters in length (including CR and LF) can be handled by the assembler.

1.3.2 Limitation on number of symbols

★ **(1) During assembly** **Approx. 3,000**

(2) During linkage

Limits to the number of symbols differs as follows depending on the type of symbol.

- Local symbols : No limit
- External definition (PUBLIC) symbols : Approx. 3,000 for all input modules
- External reference (EXTRN) symbols : Approx. 500 per module

1.3.3 Limitations on number of segments

(1) During assembly

For one source program, a total of approximately 120 for (a) to (c) below can be written:

(a) Number of segment definition pseudo-instructions
(b) Number of ORG pseudo-instructions
(c) 2 × number of VENT pseudo-instructions

(2) During Linkage

For all input modules, a total of approximately 250 (a) to (d) below can be input:

(a) Number of input modules × 2
(b) Number of segments
(c) Number of ORG pseudo-instructions in source program
(d) 2 × number of VENT pseudo-instructions

★ **1.3.4 Other limitations**

(1) During assembly

- Number of local symbols in 1 macro — 100 (including temporary parameters)
- Nest Levels — Approx. 64 Kbytes
- Macro Body Area Size — 32 Levels
  (Macro instructions, $IF instructions, $SWITCH instructions and $INCLUDE instructions together)
- Maximum number of times repeating macros can be repeated — 1023 Times

1.3.5 Number of linker branch tables

Approximately 1,000 branch tables can be created by the linker.

1.3.6 Caution on homonymous segments

If there are two or more segments with the same name in a single source module, list conversion may not be performed correctly. Ensure that all segments in a source module have different names when using the list converter.

1.4 Assembler Package Features

The assembler package features the following functions.

(1) Branch instruction optimization function (BR)

An automatic branch instruction selection pseudo-instruction (BR pseudo-instruction) is provided. To create a program which makes efficient use of memory, it is necessary to use a 1-byte or 2-byte branch instruction according to the branch destination range. However, it is very tedious to take account of the branch destination range in writing each branch instruction. When the BR pseudo-instruction is used, the assembler generates the appropriate branch instruction code for the branch range concerned.
This process is referred to as optimization.

(2) VENTn pseudo-instruction

75X series and 75XL series products have an interrupt vector table in the area from address 0H to address 0FH as a maximum (depending on the device). The start address of each interrupt service routine and the value of MBE (memory bank enable flag) and RBE (register bank enable flag) during interrupt servicing are set in this interrupt vector area.
The VENTn pseudo-instruction is provided to facilitate the setting of values in this vector table.

(3) TBR and TCALL pseudo-instructions

If it is wished to execute a 2-byte or 3-byte branch instruction or call instruction as a 1-byte instruction by means of the GETI instruction special data must be set in the GETI instruction reference table (20H to 7FH). The TBR and TCALL pseudo-instructions are provided to facilitate the setting of this data.

★ **(4) Macros**

A macro is a symbolic name which is assigned to a string of commands and is used in the source program in place of that string of commands, so that those commands are executed each time that name is called. When the same string of commands is used repeatedly, if it is made into a macro, the volume of the source program can be made smaller. Also, when a single function includes a number of commands, if a name indicating the function of that string of commands is assigned to it, the source program becomes easy to write and easy to read.

(5) Structured assembler preprocessor (ST75X)

The structured assembler preprocessor is a program for implementing structured programming in assembly language. A source program written in assembly language is input, and an assembler source program is output. Use of structured assembly language enables a program with good coding characteristics to be written.

(6) Librarian (LB75X)

A librarian function is provided. This enables a number of object modules to be collected together in a single library file.

Collecting generally applicable modules as a library file enable modules to be used more effectively, and also offers improved efficiency in terms of file management and operability.

(7) List converter (LCNV75X)

The list converter is designed to improve debugging efficiency using IE-75000-R[Note 1], IE-75001-R, EVAKIT-75X[Note 2] of programs assembled by the assembler.

In an ordinary assembly list, addresses in relocatable segments and object code in which relocatable symbols are referenced are different from the final values.

For this reason, when debugging is performed with absolute addresses specified, it is not possible to find the absolute addresses simply by referring to the assembly list, and the link map list must also be consulted.

The list converter is a program which generates an absolute assembly list in which the final absolute addresses are incorporated in the relocatable addresses and object code in the assembly list output by the assembler.

Notes 1. Maintenance product (not available for purchase)

2. Discontinued (not available for purchase)

1.5 Cautions on 75XL Series Development

The following points must be noted when using a 75XL Series device as a development device.

(1) For 75XL Series development, a 75XL Series device file (sold separately) is needed as well as the RA75X assembler package.

[MEMO]

CHAPTER 2. PRODUCT SUMMARY

This chapter outlines the file names and operating environment for each of the programs provided in the assembler package.

2.1 Product Contents

The RA75X assembler package consists of the files shown in Table 2-1.

★ **Table 2-1 Provided Files (1/2)**

Program Name	File Name	File Contents
Structured Assembler	st75x.exe	Structured assembler preprocessor execution format command file
	st75x.hlp	Execution format help file.
	st75xp.dll	Tools DLL for Project Manager
	st75xp.hlp	Tools DLL help file.
	st75x.om1	Overlay File
Assembler	ra75x.exe	Assembler execution format command file.
	ra75x.hlp	Execution format help file.
	ra75xp.dll	Tools DLL for Project Manager
	ra75xp.hlp	Tools DLL help file.
	ra75x.om1[Note]	Assembler Information File
	ra75x.om*	75X Device File
Linker	lk75x.exe	Linker Execution Format Command File
	lk75xp.dll	Tools DLL for Project Manager
	lk75xp.hlp	Tools DLL help file.
Object Converter	oc75x.exe	Object converter execution format command file.
	oc75x.hlp	Execution format help file.
	oc75x.dll	Tools DLL for Project Manager.
	oc75xp.hlp	Tools DLL help file.
Librarian	lb75x.exe	Librarian execution format command file.
	lb75x.hlp	Execution format help file.
List Converter	lcnv75x.exe	List converter execution format command file.
	lcnv75x.hlp	Execution format help file.
Library Converter	lbcnv75x.exe	Library Converter execution format command file.
Project Manager	prjtman.exe	Project Manager execution format command file.
	prjtman.hlp	Project Manager help file.
	prjtmake.exe	Project Manager make execution format command file.
	prjtedit.exe	Standard editor execution format command file.
	prjtedit.htp	Standard editor help file.
	prjtlog.exe	File used by the Project Manager.
	prjtmsg.dll	File used by the Program Manager.
	prjtspin.dll	File used by the Project Manager.
	prjtedit.dll	File used by the standard Editor.
	prjtman.txt	Project Manager supplementary explanations.
	ddummy.75x	75X Dummy Device File

Note This file is also necessary when starting the Linker and Object Converter.

★ **Table 2-1 Provided Files (2/2)**

Program Name	File Name	File Contents
	ra75x.log	Setup Log File
	necdev.ini	Tool Information File
	prjlog.pif	PIF file for Project Manager ‘prjtlog.exe’
	prjpipe.386	File used by the Project Manager
	75xtest1.asm,75xtest2.asm	Sample program file for the Assembler
	stest1.src, stest2.src	Sample program file for the Structured Assembler
	sra75x.bat	Batch file for the Structured Assembler
	cd_chg.lib, exam0000.idm, exam0001.idm,example1.idl, example2.idl,example3.asm pick_up.fnc,sub_pu.fnc	Sample program for the Project Manager’s standard editor

Remarks 1. A command file is the first file read into memory when a program is started.

2. An overlay file is read into memory if required during program execution.

3. A sample program file is used to check the operation of the assembler package.

2.2 Host Machine Models and Operating Systems

Host machines and OSs that can run RA75X are described below.

2.2.1 PC-9800 series

(1) PC-9800 series models

★ **Table 2-2 Host Machine Models (PC-9800 Series) (1/2)**

CPU / Supported Model	8086/V30™	80286	80386	80486
PC-9801	PC-9801 E F1/2/3 M2/3 VF2 VM0/2/4/21/11 U2 UV2/21/11 CV21 UR/20 UF XLmodel 1/2/4 VX0/2/4/01/21/41 UX21/41 RX2/4/21/51 EX2/4 XL2 RL2/5/21/51 RA2/5/21/51 ES2/5 RS21/51 Tmodel W2/W5/W7/S5/F5/F51/F7 LV21/22 LX2/4/5/5C LS2/5 N NV NL	XAmodel 1/2/3/11/21/31 XLmodel 1/2/4 VX0/2/4/01/21/41 UX21/41 RX2/4/21/51 EX2/4 DX2/4/U2/U5 LX2/4/5/5C	XL2 RL2/5/21/51 RA2/5/21/51 ES2/5 RS21/51 Tmodel W2/W5/W7/S5/F5/F51/F71 DS2/5/U2/U5 DA2/5/7/U2/U5/U7 CS2/5/5W US/40/80 FS2/5/7/U2/U5/U7 FX2/5/U2/5 LS2/5 NS/20 NS/E/20/40 NS/T/20/40 NS/L/40 NC/40	FA2/5/7/U2/U5/U7 BXU2/U6/M2 BAU2/U6/M2 BS2U2/U7/M2 BX2U2/U7/M2 BA2U2/U7/M2 BA3U2 BX3U2 NA/40/120/C/C40/120C NS/R/40/120 NX/C/120 Ne/120/W/340/W NS/A/120/340 NL/R NL/A120/260A

Remark Models in the table that have a high resolution mode can also be used in high resolution mode.

Caution At least 640K bytes of internal memory is required.

★ **Table 2-2 Host Machine Models (PC-9800 Series) (2/2)**

CPU / Supported Model	8086/V30	80286	80386	80486
PC-H98			model70-002/100 model60-002/040/100 modelU60-002/040/100	Smodel8-002/040/100 SmodelU8-002/040/100 model80-002/040/100 modelU80-002/040/100 model90-002/040/100 modelU90-002/040/100 model100 modelU100 model105-100/300 modelU105-100/300 Tmodel1/2/2C

CPU / Supported Model	80486	Pentium™
PC-9821	Ae U2/U7/U7W/M2/M7/M7W As U2/U7/U7W/U8/U8W/M2/M7/M7W Ap U2/U7/U7W/U9/U9W/M2/M7/M7W Be U7W Bs U7W Bp U7W/U8W As2 U2/U7W/U8W/M2 Ap2 U2/U8W/C9W/C9T/M2 Xp U8W/C8W Xs U7W/C8W Xe U7W Ap3 C8W/C9W/U2/M2 As3 C8W/U2/M2 Xe10 4/C4 PC-9821 model S1/S2 Ce model S1/S2 Ce2 model S1/S2/S2D/T2/T2D Cs2 model S2/S3 Cb model 2F/2D/2 Cx model S2/S3 Cb2 B/T/M/A Ts/120/W Ne/120/W/340/W Np 340W/540W/810W Ns 340W/540W/810W Ne2/340W Nd/340W Ld 260/350A/350A2 Lt 260/350A/540A Nm/340 Ne3/3 Nd2/3	Af U9W/M9W Bf U9W/M9W An U8W/C9T/U2/M2 Xt C10W Xa U8W/C9W/C10W/U1 Xn U8W/C9W Xf U8W/C9W/U1 Xa7 C4/C8 Xa9 C4/C8 Xa10 C4/O12 Cx2 S15B/S17B/S15T/S17T Cf model S3 Nf 340W/810W

Remark Models in the table that have a high resolution mode can also be used in high resolution mode.

Caution At least 640K bytes of internal memory is required.

★ **(2) PC-9800 series OS**

OS	Version
MS-DOS™	Ver. 3.30-Ver5.00[Note 1]/5.00A[Note 1]
Windows	Ver. 3.1[Note 2]

Notes 1. A task swapping function is provided in Ver. 5.00/5.00A, but the task swapping function cannot be used with the RA75X assembler package.

2. If the Assembler is used in Windows, Project Manager is necessary. If Project Manager is not used, run the Assembler under MS-DOS.

★ **(3) Assembly package supply medium**

3.5-inch FD (2HD)

2.2.2 IBM PC/AT™

(1) Models

IBM PC/AT

Caution At least 384K bytes of internal memory is required (not including the system area).

★ **(2) IBM PC/AT OS**

OS	Version
PC DOS™[Note1]	Ver. 5.0 to Ver. 6.0
MS-DOS[Note 1]	Ver. 5.0 to Ver. 6.0 5.0/V[Note 2]
IBM DOS™[Note 1]	J5.02/V[Note 2]
Windows	Ver. 3.1[Note 3]

Notes 1. A task swapping function is provided from Ver. 5.0 onward, but the task swapping function cannot be used with the RA75X assembler package.

2. Only the English-language mode is supported.

3. If the Assembler is used in Windows, Project Manager is necessary. If Project Manager is not used, run the Assembler under MS-DOS PC DOS or IBM DOS.

★ **(3) Assembler package supply media**

3.5-inch FD (2HC)

CHAPTER 3. ASSEMBLER PACKAGE EXECUTION

This chapter describes the assembler package installation procedure and execution procedures.

Executing each program in accordance with the execution procedures described here will enable the user to become familiar with the operation of the assembler package.

3.1 Before Running Assembler Package

★ 3.1.1 Assembler package installation procedure

Install the Assembler Package and the Project Manager. The Project Manager is necessary if the Assembler is to be used in Windows.

This software is supplied on four 3.5-inch floppy disks.

Installation can be done by one of the following three methods.

Table 3-1 Installation Methods

Installation Method	Programs which can be installed
Execute 'setupj.exe' in Windows 3.1.	Assembler Package Project Manager (Japanese Edition)
Execute 'setupe.exe' in Windows 3.1.	Assembler Package Project Manager (English Edition)
Execute 'dosint.bat' under DOS.	Assembler Package

Cautions 1. Close all currently running applications before installation.

2. Windows 3.1 may become unstable if installation is cancelled, so restart the host machine.

3. The information file used by each tool, 'necdev.ini,' is changed only when 'setupj.exe' or 'setupe.exe' is executed and the software is installed. Accordingly, if you use Project Manager, install it using 'setupj.exe' or 'setupe.exe.'

4. If your OS is the Japanese Windows 3.1, execute 'setupj.exe', if English Windows 3.1, execute 'setupe.exe', and if DOS, execute 'dosinst.bat' to install the Assembler Package and the Project Manager.

(1) In installing by running 'setupj.exe' or 'setupe.exe' under Windows 3.1

The following installation example is for the case that the host machine is a PC-9800 series model, the Assembler Package and Project Manager are being read from Drive 'C:', the execution format is being installed in A:\nectools\bin and the sample program is being installed in A:\nectools\smp75x. It is assumed that Windows has already been started.

[Execution Example]

<1> Running the Installer

(a) Insert the 'RA75X SETUP DISK#1' in the floppy disk drive.

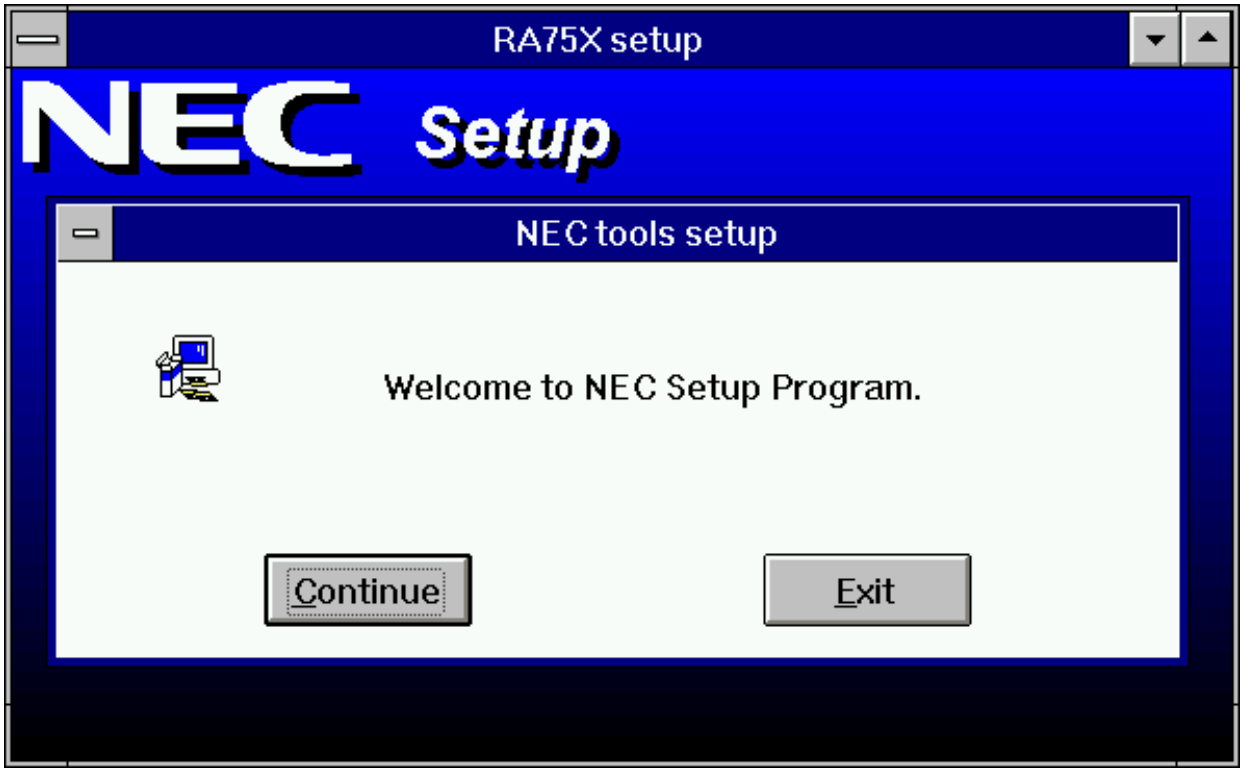
(b) Select 'Run' from the File menu.

(c) Input the following in the command line input box.

Input 'setupj.exe' in the command line input box in Japanese Windows 3.1 and 'setupe.exe' in English Windows 3.1.

(d) Select 'OK'.

(e) After setup initialization, the installer will start.

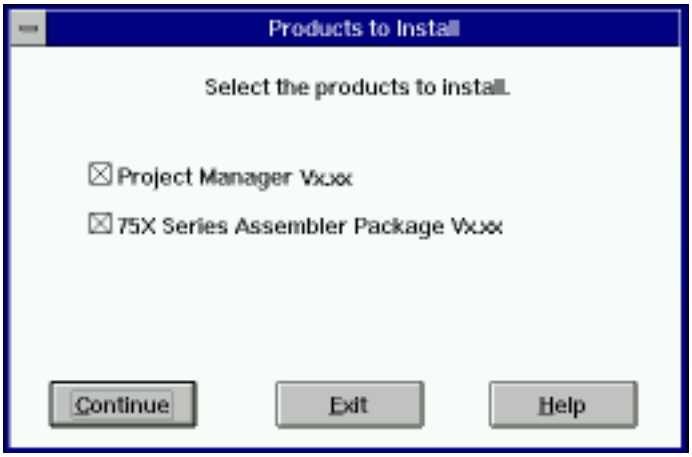


(f) To continue the installation, select ‘Continue (C).’
To terminate installation, select ‘End (E).’

<2> Select the item to be installed.

(a) Select the items to be installed by clicking the appropriate check boxes.
In the default installation, the Assembler Package and Project Manager are selected for installation.

(b) After selecting the items to be installed, select 'Continue (C).'
If you are installing the Assembler, it is necessary that Project Manager be installed already or that it be installed at the same time that the Assembler is installed.

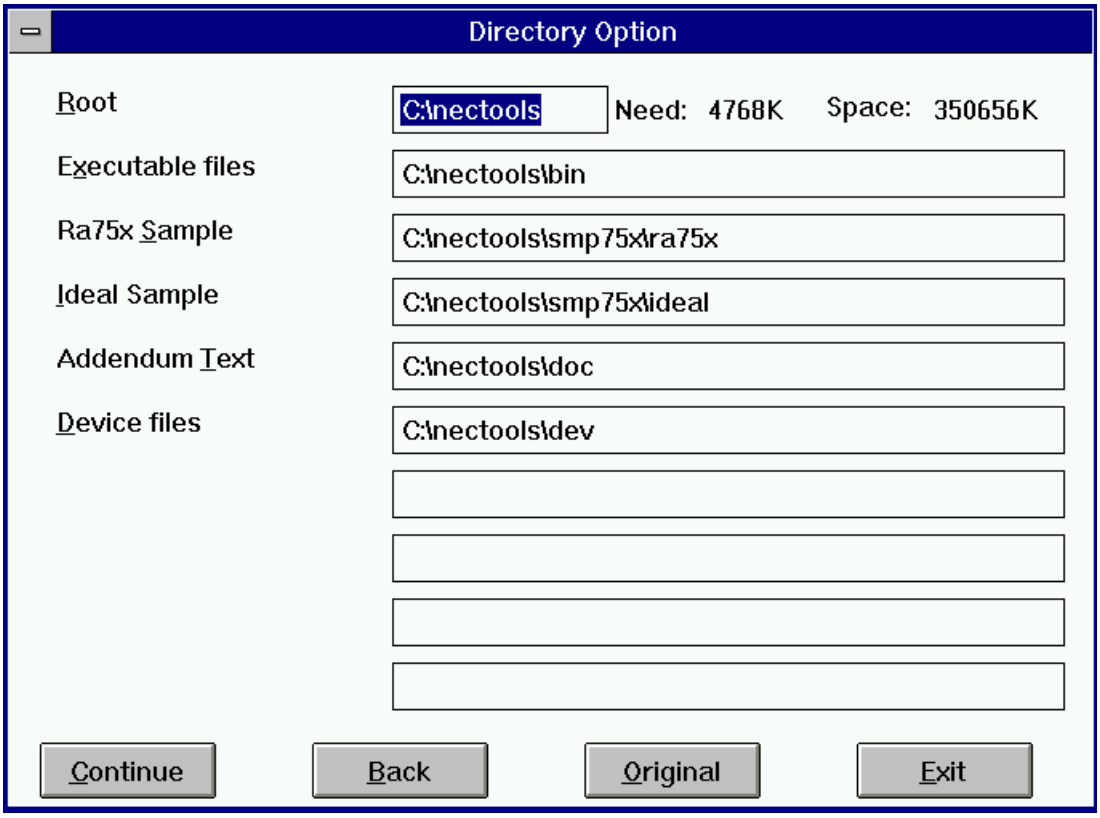


(c) To end the installation, select 'End (E).'

(d) Items which cannot be installed will be displayed in gray.

<3> Specify the installation directory.

(a) The dialog box for specifying the installation destination directories is then displayed. After inputting the installation directories as shown below, select 'Continue (C).'



(b) To return to the installation item selection dialog box, select 'Back (B).'

(c) To specify the default directory, select 'Initialize (O).' The default installation destination directory thus becomes '\nectools' on the drive where Windows is installed. If the tools have already been installed using the installer, that root directory is used. To edit the root directory, the directories which are related to the root directory and linked to it are changed.

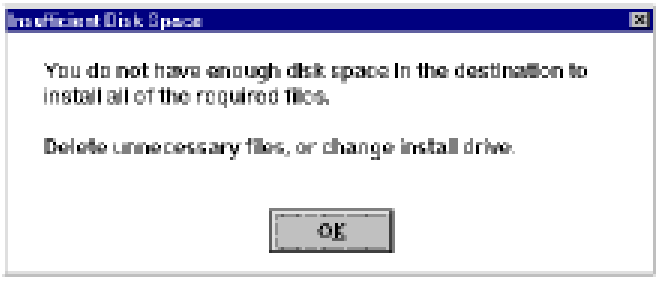
(d) To terminate the installation, select 'End (E).'

(e) Items which cannot be installed are displayed in gray.

(f) If there are no supplementary explanations, the supplementary explanation directory is displayed in gray. If there are supplementary explanations, icons are registered after installation is completed, so read their contents.

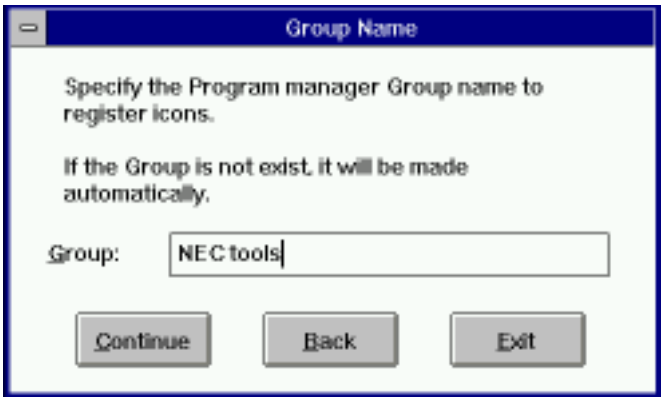
(g) If the specified directory is incorrect, it will result in an error and the following message will be displayed.

(h) If the capacity is insufficient, it will result in an error and the following message will be displayed.



<4> Specify the group to be registered in Program Manager.

(a) The dialog box for specifying the registered group name is next displayed. After inputting the group name to be registered as shown below, select 'Continue (C).' If the specified group does not exist, that group is newly created.
Also, if the specified group has already been registered using the Installer, that group is used.



(b) To return to the dialog box for specifying the installation destination, select 'Back (B).'

(c) To terminate installation, select 'Exit (E).'

(d) If you are not installing Project Manager, the dialog box for specifying the registered group name is not displayed.

<5> Start copying files.

(a) The dialog box for starting copying of files is displayed. If you select 'Continue (C),' copying of files will start.

(b) To return to the dialog box for specifying the registered group name, select 'Back (B).'

(c) To terminate installation, select 'Exit (E).'

<6> Copying Files

(a) When 'Cancel (C)' is selected, the following message is displayed.

(b) To terminate installation, select 'Yes (Y).'
To resume copying of files, select 'No (N).'

<7> Replace the Output Media

(a) When the following message is displayed, insert 'RA75X SETUP DISK #2' in the floppy disk drive.

(b) When the same message is repeated, insert 'RA75X SETUP DISK #3,' then next 'RA75X SETUP DISK #4' in the floppy disk drive.

<8> The registered group and icons are created.

The Assembler cannot be operated in Windows, so no icon is created for it.

<9> 'AUTOEXEC.BAT' is changed.

(a) Using the radio button, select whether to modify the 'AUTOEXEC.BAT' file by selecting 'Installer will modify automatically' or 'Modify it manually later.'

(b) If 'Installer will modify automatically' is selected, the 'AUTOEXEC.BAT' file on the disk where the Windows directory is located will be rewritten and the original file will be saved with the name 'AUTOEXEC.OLD.'

(c) If ‘Modify it manually later’ is selected, ‘AUTOEXEC.SMP’ will be created in the root directory as a sample to help you modify the AUTOEXEC.BAT file.
If ‘AUTOEXEC.SMP’ already exists, the following contents will be added to it.

```
REM --- nec tools installer 0xx/xx/xx xx:xx:xx ---
PATH a:\nectools\bin;%PATH%
```

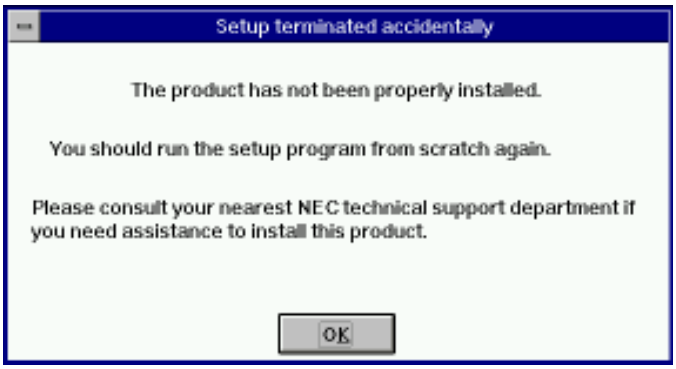
<10> Finish installation.

(a) The following message is displayed. Select ‘OK’ to terminate the installation.

(b) Restarting the Host Machine

Remark If you cancelled the installation in the middle, the following message will be displayed. Press ‘Continue (C)’ to return to the dialog box where you selected “Exit (E).’

Select ‘Exit (E)’ to display the next message.
Select ‘OK’ to close the installer.



(2) If you are installing by running 'dosinst.bat' under DOS.

The example is shown of installation in the case where a PC-9800 series host machine is used, the Assembler Package is read from drive C:, the execution format is installed in A:\nectools\bin and the sample program is installed in A:\nectools\smp75x.

[Execution Example]

<1> Execute the batch file 'dosinst.bat.'

The description format is as shown below.

```
x>dosinst.bat △ Installation source drive △ Installation destination drive △ Installation destination directory △
```

△ : indicates 1 or more blank spaces.

<Example of Description>

```
A>mkdir nectools              ; Creates an installation destination directory.
A>C:                          ; Changes from the current drive to the installation source drive.
C>dosinst.bat c: a: nectools  ; Executes the batch file.
```

<2> To end the installation, input the CTRL key + C key or the ALT key + C key when the message 'Press any key to continue." is displayed.

The following message will be displayed.

```
End batch file execution?  (Y/N) ?
```

Input 'Y' to terminate execution of the batch file.

Press 'N' to continue execution of the batch file.

<3> If you terminate execution of the batch file, please refer to the contents of 'nectools\ra75x.add' and modify 'autoexec.bat.'

<Contents of 'ra75x.add'>

```
REM PLEASE ADD TO THE BOTTOM OF YOUR AUTOEXEC.BAT.
PATH a:\nectools\BIN;%PATH%
```

(3) File Configuration after Installation

The Assembler Package file configuration after installation is as follows.

```
nectools\
    bin\
        ra75x.exe       Assembler execution format command file.
        st75x.exe       Structured Assembler preprocessor execution format command file.
        lk75x.exe       Linker execution format command file.
        oc75x.exe       Object converter execution format command file.
        lb75x.exe       Librarian execution format command file.
        lbcnv75x.exe    Library converter execution format command file.
        lcnv75x.exe     List converter execution format command file.
        *.hlp           Execution format help file.
        *p.dll          Project Manager tools DLL
        *p.hlp          Tools DLL help file.
        prjtman.exe     Project Manager execution format command file.
        prjtmake.exe    Project Manager make execution format command file.
        prjtlog.exe     File used by the Project Manager.
        prjtedit.exe    Standard editor execution format command file.
        prjtmsg.dll     File used by the Program Manager.
        prjtspin.dll    File used by the Project Manager.
        prjtedit.dll    File used by the standard editor.
        prjtman.hlp     Project Manager help file.
        prjtedit.htp    Standard editor help file.
        ra75x.om1       Assembler Information file
        ra75x.om*       75X device file.
        st75x.om1       Overlay file.
    doc\
        prjtman.txt     Project Manager supplementary explanations.
    dev\ Note
        ddummy.75X      75X dummy device file (use Project Manager)
    setup\
        ra75x.log       Setup log file.
    smp75x\
        ideal\
            example\
                *.*     idea-L light sample program (for training)
            org\
                *.*     idea-L light sample program (for saving)
        ra75x\
          *.asm         Assembler sample file.
          *.s           Structured Assembler sample file
        sra75x.bat      Structured Assembler sample batch file
windows\
        necdev.ini      Tools information file
        prjlog.pif      Project Manager 'prjtlog.exe' PIF file
    system\
        prjpipe.386     File used by the Project Manager.
```

Note This is an empty directory. Please copy the 75XL device file (sold separately) to this directory.

(4) Adjustment of 'CONFIG.SYS' file contents

Create a file 'CONFIG.SYS' in the root directory of the disk used to start up the host machine, and include the following in it:

FILES=n

'n' is a number indicating the maximum number of files that can be opened by one program. '20' or more should be written in order to use the assembler package. However, a larger number than this may be necessary depending on what other programs (editor, etc.) are to be used concurrently with the assembler package. Please check the manual, etc., for each program.
The following should also be specified in order to increase the program execution speed:

BUFFERS=n

'n' is the number of system input/output buffers to L be used by the entire system. To a certain degree, the greater this value, the faster is the disk input/output speed. However, this value is directly related to memory consumption. If there is adequate installed memory capacity, it is as well to specify a larger value. If 512 to 640 KB of memory is installed, a value of around '20' should be specified.

(5) Setting environment variable

The following environment variable is supported by the assembler package.

'INC75X'..... Specifies the include file search path. This environment variable is used when using an include file which defines constant values etc., specific to the product subject to assembly. See the section on the assembler -I option for details.

Caution

The assembler package assembler and linker are divided into an executable command file and overlay files. When these program files are in the current path (current directory of current drive), the program files can be loaded simply by specified 'RA75X' in the command line. However, in some cases source module files and the associated object files, etc., are located in the current path and other executable file, etc., are located in a different path.
The method for accomplishing this is as follows.

Example:
If the assembler program files ('RA75X.EXE' and 'RA75X.OM1', 'RA75X.OM2', 'RA75X.OM3', & 'RA75X.OM4') are in the subdirectory '\BIN\75X' and the current path is other than '\BIN\75X' there are two possible start methods as shown below.

(1) Specify the drive and directory in which the program files are located in full, as follows:

```
A>C:\BIN\75X\RA75X 75XTEST1.ASM -C106
```

(2) Include the name of the path in which the program files are located in a list of command search paths supported by the OS.
For example:

```
A>SET PATH=A:\BIN; C :\BIN; C:\BINY75X
```

If 'AUTOEXEC.BAT', etc., is used for this setting the assembler can subsequently be started simply by specifying 'RA75X' in the command line:

```
A>RA75X 75XTEST1.ASM -C106
```

(In either case, the command file and overlay files must be in the same path.)

3.1.2 Sample programs

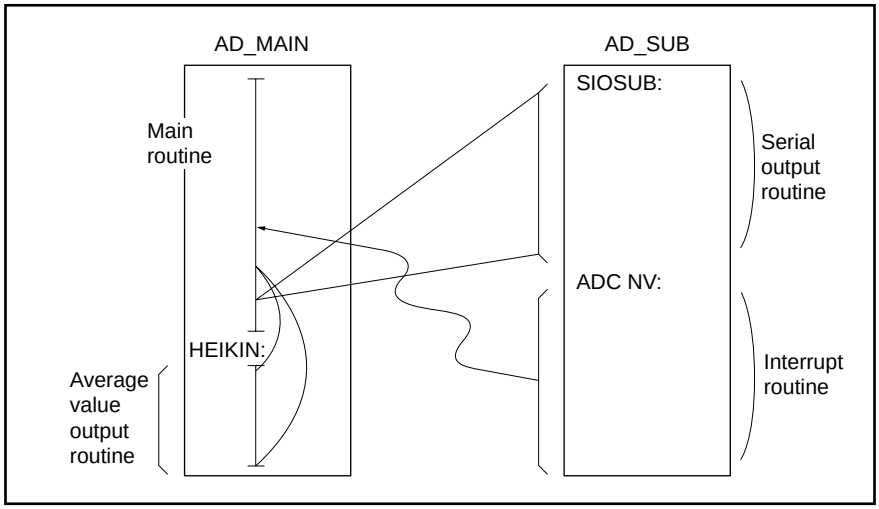
Of the files stored on disk, **“75XTESTI.ASM”** and **“75XTEST2.ASM”** are sample program for confirming operation. These files will be the input to the assembler as source program files in subsequent assembler operations.

A brief description of the contents of this sample program is given below.

This sample program is an A/D conversion program which samples an analog source (PTH00H pin input signal) 8 times using μPD75106 on-chip hardware (programmable threshold port and serial interface) and outputs the average value from the serial output pin.

The sample program is divided into two modules: one is stored in the source module file **“75XTEST1.ASM”** under the module name ‘AD_MAIN’, and the other is stored in the source module file **“75XTEST2 .ASM”** under the module name ‘AD_SUB’.

Figure 3-1 Sample Program Construction



Caution

This sample program has been provided as reference software to help the user learn the functions and operating procedures of the assembler package, and cannot be used directly as an application program.

3.2 Assembler Package Execution Procedure

3.2.1 Assembler, Linker, Object Converter

(1) Changes to 'RA75X' under 'SMP75X' which is under 'NECTOOLS' in the current directory in Drive A.

```
A>
A>CD\NECTOOLS\SMP75X\RA75X
A:\NECTOOLS\SMP75X\RA75X>
```

(2) Assemble the sample program "75XTEST1.ASM".

- The following is input in the command line.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106
```

- The following message is output at the console.

```
75X Series Assembler VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation l985 ,XXXX

ASSEMBLY START

 TARGET CHIP        :  UPD75106
 STACK SIZE = 000AH

 ASSEMBLY COMPLETE, NO ERROR FOUND
```

(3) Check the contents of drive B.

The assembler has output **"75XTEST1.REL"** (object module file) and **"75XTEST1.PRN"** (assembly list file).

(4) Assemble the sample program “75XTEST2.ASM”.

- The following is input in the command line.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST2.ASM -C106
```

- The following message is output at the console.

```
75X Series Assembler VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1985 ,XXXX

ASSEMBLY START

 TARGET CHIP : UPD75106
 STACK SIZE = 0002H

 ASSEMBLY COMPLETE, NO ERROR FOUND
```

(5) Check the contents of drive B.

The assembler has output **“75XTEST2.REL”** (object module file) and **“75XTEST2.PRN”** (assembly list file).

(6) Link object module files “75XTEST1.REL” and “75XTEST2.REL” output by the assembly.

- The following is input in the command line.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK
```

- The following message is output at the console.

```
75X Series Linker VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation l985

LINK COMPLETE, NO ERROR FOUND
```

(7) Check the contents of drive B.

The linker has output **“75XTEST.LNK”** (load module file) and **“75XTEST1.MAP”** (link list file).

(8) Input the load module file “75XTEST.LNK” output as the result of linkage to the object converter.

```
A:\NECTOOLS\SMP75X\RA75X>OC75X 75XTEST.LNK
```

- The following message is output at the console.

```
75X Series Object Converter VX.XX [XX Xxx XX]
Copyright (C) NEC Corporation 1985 ,XXXX

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

(9) Check the contents of drive B.

The object converter has output **“75XTEST.HEX”** (HEX format object module file) and **“75XTEST.SYM”** (symbol table file).

3.2.2 Librarian

(1) Execute the librarian.

- Execute the librarian.

```
A>\NECTOOLS\SMP75X\RA75X>LB75X
```

- The following message is output to the console and the librarian prompt is displayed.

```
75X Series Librarian VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1984, XXXX

*
```

- Input the librarian subcommand as shown below (the EXIT command returns control to the OS).

```
*CREATE 75XTEST.LIB
*ADD 75XTEST1.REL, 75XTEST2.REL TO 75XTEST.LIB
*LIST 75XTEST.LIB TO 75XTEST.LST PUBLICS
*EXIT

B>
```

(2) Check the contents of drive B.

The librarian has output **"75XTEST.LIB"** (library file) and **"75XTEST.LST"** (list file).

(3) The library file created in this way can be input to the linker as follows:

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST.LIB -075XTEST.LNK
```

3.2.3 List converter

(1) Execute the list converter.

- The following is input in the command line.

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -L75XTEST.LNK 75XTEST1
```

- The following message is output at the console.

```
List Conversion Program for RA75X VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1986, 1997

Pass 1: start     .............
Pass 2: start     .........
Conversion complete
```

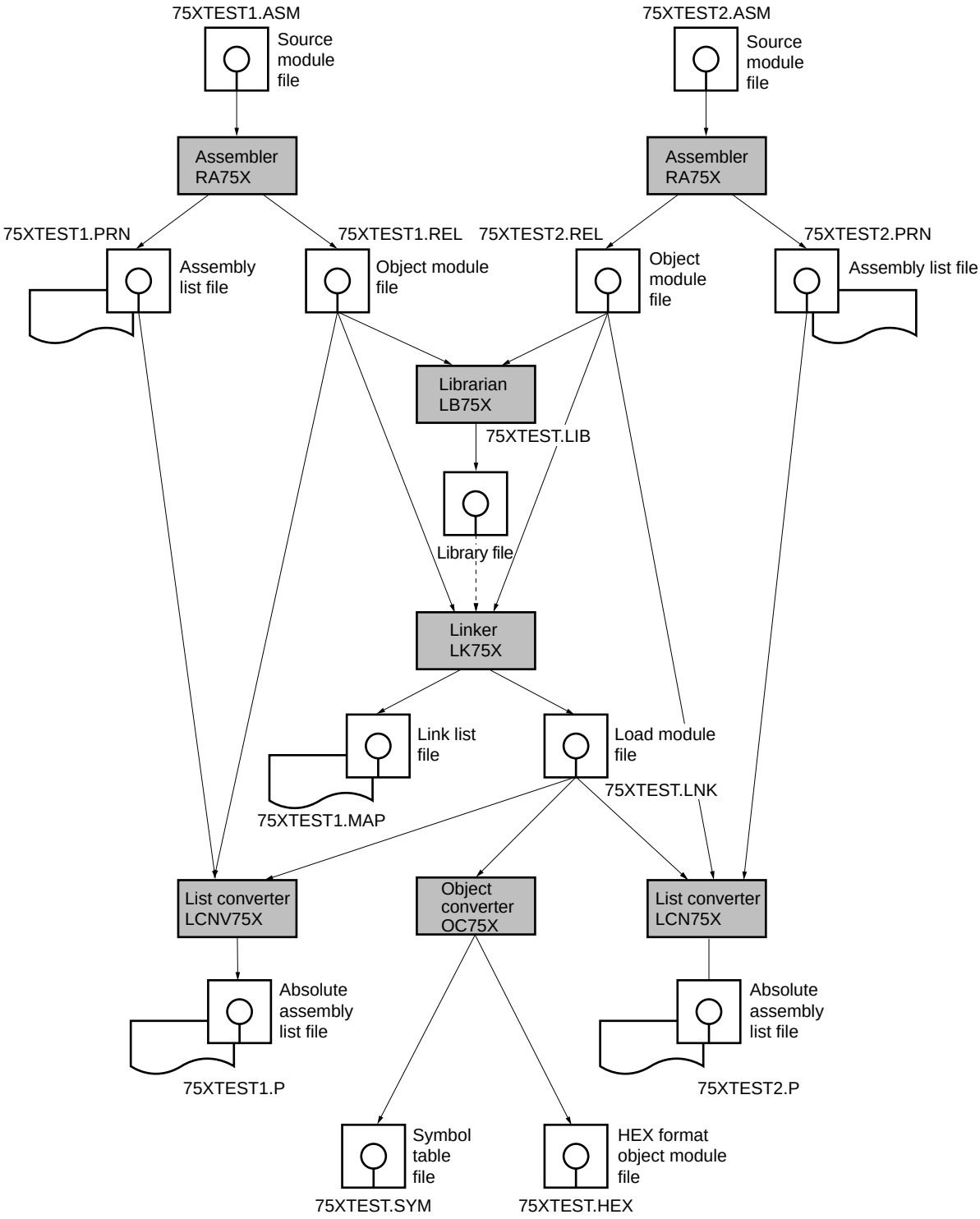
(2) Check the contents of drive B.

The list converter has output **"75XTEST1.P"** (absolute assembly list).

3.3 Summary of Assembler Package Execution Procedure

The assembler package execution procedure for the sample program is summarized in the figure below.

Figure 3-2 Assembler Package Execution Procedure



CHAPTER 4. ASSEMBLER

The assembler (RA75X) has as input a source module file written in 75X Series and 75XL Series assembly language, converts this into machine language, and outputs the result as an object module file.

In addition, the assembler outputs assembly list files such as an assembly list, symbol table list, cross-reference list, etc.

If there are assembly errors, the assembler outputs error messages in the assembly list, error list, etc., indicating the cause of the errors.

4.1 Assembler Input/Output Files

Assembler (RA75X) input/output files are shown in Table 4-1.

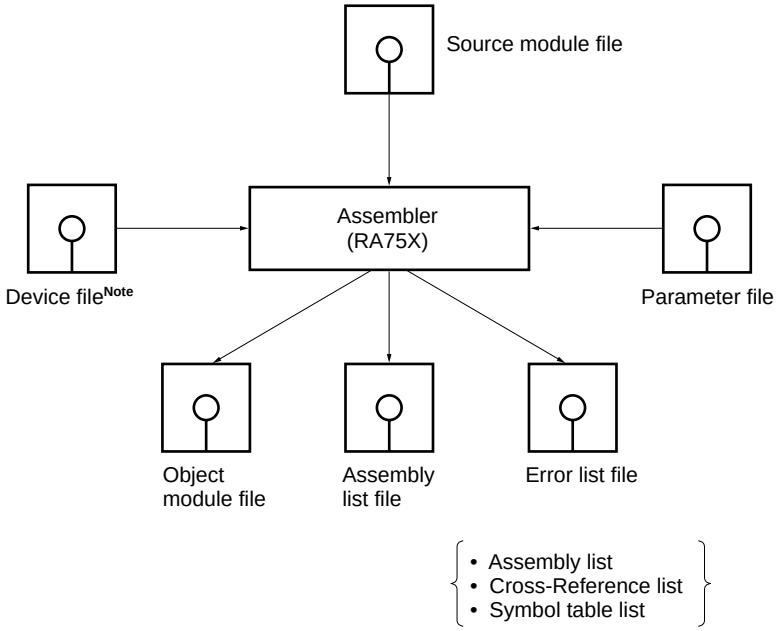
Table 4-1 Assembler Input/Output Files

Type of File		Default File Type
Input file	**Source module files** Source module files written in 75X Series and 75XL Series assembly language	.ASM
	Parameter files Note 1 When it is not wished to specify option one by one in the command line when the assembler is started, the options to be specified, etc., are generated as a parameter file using the editor etc.	.PRA
	Device file (sold separately) File containing 75XL Series product SFR information, etc.	.75X
Output file	**Object module file** Binary file containing machine language information, machine language location address related information and symbol formation.	.REL
	Assemble list file Note 2 File containing assembly information such as assembly list, symbol table list, cross-reference list, etc.	.PRN
	Error list file File containing the assembly-time error information.	.ERA

Notes 1. For detail, see **“4.4.4 Description of assembler options (18) -F”**.

2. The address of the Assemble List File created by the Assembler is a virtual address, so if you are referring to the actual address, refer to the Absolute Assemble List File address created by the list converter.

Figure 4-1 Assembler Input/Output Files



Note A 75XL Series device file (sold separately) is needed for 75XL Series development.

4.2 Assembler Functions

- The assembler reads source module files and converts the assembler language into machine language.
- If a coding error is found in the source module, an error message is output in the assembly list and error list.
- The assembler performs assembly processing in accordance with the assembler options specified when the assembler is started (or in the source module header). See **4.4 "Assembler Options"** for assembler options.
- When this processing terminates normally, the assembler output a termination message and returns control to the OS.
- Maximum assembler capabilities are shown below.

	Item	Maximum Value
	Number of characters that can be written in one source line (including C_R and L_F)	220 characters
★	Characters used in symbol	The first 31 characters (8 characters when the -NS option is specified) are valid.
★	Number of symbols handled by assembler	Approx. 3000
	Number of segments that can be handled by assembler Total of ↓ 1) Number of segment definition pseudo-instructions 2) Number of ORG pseudo-instructions 3) 2 × number of VENT pseudo-instructions	Approx. 120

4.3 Assembler Start Method

4.3.1 Starting the assembler

The assembler is started by inputting the following command in the format shown in the OS command line.

```
X>RA75X[␣option...]␣input file name [␣option...]
```

- X indicates the current drive.
- “Input file name” is the name of the source module file to be assembled. Only one input file name can be specified. Therefore, it is not possible to assemble multiple source module files in one run of the assembler. The drive name, directory name, etc., can be added to the input file name.

  Example RA75X -C106 B : 75XTEST1.ASM
  RA75X -C106 C : \USER\NEC\75XTEST1.ASM

- “option” is a string of 1 to 3 characters beginning with the “ - “ symbol, and may be followed by parameters. Options can be written before and after the input file, and if there are multiple options, they can be written in any order. However, if multiple identical options or options of the same kind are written, in some cases an error is generated, and in some cases the last output specified is valid for details. See **4.4 “Assembler Options”** for details.
- One or more blanks (spaces or TAB) should be used to separate options and the input file name.
- The input file name and options can be written in the parameter file. For the use of the parameter file, see the item on the -F option in **4.4.4 “Description of assembler options”**.
- As a default operation, a file with the same name as the input file but with the file type changed to ‘.REL’ is created in the current directory. This can be changed by means of the ‘-O’ option.

4.3.2 Execution start and end messages

(1) Execution start message

When the assembler is started, an execution start message is displayed on the console.

```
75X Series Assembler VX. XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1985, XXXX

ASSEMBLY START
```

(2) Execution end message

- If no assembly errors are detected during assembly, the assembler outputs the following message and returns control to the OS.

```
ASSEMBLY COMPLETE, NO ERROR FOUND
```

- If assembly errors are detected during assembly, the assembler displays the number of errors on the console and returns control to the OS.

```
ASSEMBLY COMPLETE, 5 ERRORS FOUND( 10)
```

- If a fatal error is detected during assembly which prevents assembly from continuing, the assembler outputs a message to the console, stops execution, and returns control to the OS.

Example 1. When a source file which does not exist in drive B is specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X SAMPLE.ASM -C106
75X Series Assembler VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1985 ,XXXX

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

In this example, an error is generated since a source file which does not exist in drive is specified, and assembly is aborted.

Example 2. When the -C option is not specified

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM
75X Series Assembler VXX.XX [XX Xxx xx]
   Copyright (C) NEC Corporation 1985, XXX

***ERROR A099 CHIP IS NOT SELECTED
   –C000A/000/004/006/008/028/036/048/064/066/068/
     104/106/108/P108/112/116/116H/117H
     206/208/CG208/212A/216A/CG216A/217/218/236/237/238/268/
     304/306/308/312/312B/316/316A/316B/328/336/352A
     402/
     512/516/517/518/
     617A

     0004/0006/0008/P0016/                                  } Note
     3012/3016/3017/P3018/3104/3106/3108/P3116              }

Program aborted
```

Note The 75XL Series device file in the same directory as the assembler main program is displayed.

In this example, an error is generated since the -C option for specification of the target product has not been specified, and assembly is aborted.
When the assembler outputs an error message and aborts assembly, the cause of the error message should be found in **13.1 "Assembler's Error Messages"**, and appropriate action taken.

4.3.3 Assembler error handling

If the assembler detects an error during execution, it performs one of the following three kinds of processing according to the severity of the error.

(1) Abort error

If the assembler is generated which prevents program execution from continuing, the program displays a 'Program aborted' message, and the program is aborted immediately.

(2) Fatal error

If an error is generated which would result in generation of object code different from that intended by the user, the program nevertheless continues processing to the end, then outputs the message "ASSEMBLY COMPLETE, X ERRORS FOUND" (where X is the number of errors).

(3) Normal termination

If the program terminates normally, it outputs the message "ASSEMBLY COMPLETE NO ERROR FOUND".

- Error related to start line (only output to standard output)

```
Error number␣error message
```

- Error unrelated to start line

Output format for assembly list and standard output

```
***␣ERROR␣#error number,␣STNO  #nnnn␣(mmmm), ␣error message
```

nnnn : Error line number
mmmm : Previous error line number

4.3.4 Assembler termination status

When the assembler terminates and returns control to the OS, one of the following error status codes is returned to the OS.

Termination Condition	Termination Status
Normal termination	0
Fatal error	1
Abort error	2

When the assembler is started from a batch file under MS-DOS (PC DOS, IBM DOS), it is possible to determine whether there are any assembly errors automatically using these values.

4.4 Assembler Options

4.4.1 Types of assembler options

Assembler options are used to give the assembler detailed directions concerning its operation. There are 19 different options as shown below.

Table 4-2 Assembler Options (1/2)

No .	Description Format	Function/Category	Default Interpretation
1	-C product	Specification of assembler target product	Cannot be omitted
2	-M mode	75XL Series CPU mode switching Cannot be specified when a 75X Series device is used.	Cannot be omitted when a 75XL Series device is used.
3	-O[file name] -NO	Object module file specification	'Input file name.REL' is create in current path.
4	-J -NJ	Object module file forced output output specification	-NJ
5	-G -NG	Specification of output to object module of of symbol information for debugging	-G
★ 6	-GA -NGA	Specifies output of source debugging information object module files.	-GA
7	-P[file name] -NP	Assembly list file specification	'Input file name.PRN' is create in current path.
8	-E[file name] -NE	Error list file specification	-NE
9	-KS -NKS	Symbol table list output specification	-NKS
10	-KX -NKX	Cross-reference list output specification	-NKX
★ 11	-CA -NCA	Specifies distinguishing between upper/ lower case letters. -CA : Do not distinguish between upper/ lower case letters. -NCA: Distinguish between upper/lower case letters.	-NCA
★ 12	-S -NS	Sets the symbol name length. -S : Sets a maximum of 31 characters. -NS : Sets a maximum of 8 characters.	-S
★ 13	-Dsymbol name [= numerical value] [,Symbol Name [= numerical value]…] -NDsymbol Name	Symbol Name	None
14	-LL[number of lines] -LW[number of characters]	Number of lines and columns per page of assembly list file	-LL66 -LW132
15	-LT[number of characters]	Specification of number of TAB expansion columns in assembly list table	-LT8
16	-KA -NKA	Assembly list output specification	-KA
17	-I path name [, path name...]	Include file search path specification	Search the path specified by the 'IN-CLUDE' pseudo command, the source module file path and the path set in environmental variable 'INC75X, ' in that order.
18	-F file name	Parameter file specification	Read all the options and file names from the command line.

Table 4-2 Assembler Options (2/2)

No .	Description Format	Function/Category	Default Interpretation
★ 19	-Ypath name	Specifies the device file search path.	It searches in the sequence of the '..\DEV' path with respect to the RA75X starting path, the RA75X starting path, the current directory, and the path set in the environment variable 'PATH.'

Remark Options can be written in either upper- or lower-case characters.

4.4.2 Assembler options specification method

Assembler options are specified in the command line when the assembler is started or in a parameter file.

In addition, certain assembler options can be specified by a control instruction which corresponds to the module header of the source module (until a comment, mnemonic or pseudo-instruction appears in the source module).

The assembler options shown below can be specified by using a control instruction in ( ) in the module header. It is convenient to use these options to specify in the module header those items which must always be specified each time assembly is performed.

★ • -C (PROCESSOR) or (PC)
★ • -CA (CAP) or (CA), -NCA (NOCAP) or (NOCA)
• -G (DEBUG) or (DB), -NG (NODEBUG) or (NODB)
★ • -GA (DEBUGA) or (DA), -NGA (NODEBUGA) or (NODA)
★ • -J (GENERATE) or (GEN), -NJ (NOGENERATE) or (NOGEN)
• -KS (SYMBOLS) or (SB), -NKS (NOSYMBOLS) or (NOSB)
• -KX ‘(XIREF) or (XR), -NKX (NOXREF) or (NOXR)
• -LL (PAGELENGTH) or (PL), -LW (PAGEWIDTH) or (PW)
★ • -LT (TAB) or (TB)
• -M (MODE) or (MD)
★ • -S(SYMLEN) or (SL), -NS(NOSYMLEN) or (NOSL)

The control instructions is specified in the module header of the source module as shown below.

$ [␣] control instruction
⋮
One option can be specified on one line.
(Multiple control instructions cannot be specified on one line.)
‘$’ is specified in the first column.

Example To specify a control instruction in the module header of a source module.

```
$       SYMBOLS
$       XREF
$       TITLE='A-D CONVERTER
;*******************************************************
;***     A-D CONVERT PROGRAM          ***
;*******************************************************
```

4.4.3 Assembler options priority order

(1) If multiple identical options or options of the same kind are specified in the command line, the option specified last is valid.
(2) If the same or same kind of option is specified in the parameter file and in the command line, the command line option is valid.
(3) If another special assembler option is specified among assembler options, it may be meaningless.
The priority order for these assembler options is shown in **Table 4-3**. A cross (x) in the table means that when the option under “A” is specified the option under “B” is invalid.

Table 4-3 Assembler Options Priority

★

B \ A	-NO	-NP
-GA	∴	—
-G	∴	—
-J	∴	—
-KA	—	∴
-KS	—	∴
-KX	—	∴
-LL	—	∴
-LW	—	∴
-LT	—	∴

∴ : The side B options are invalid.
— : There is no relationship between side A and side B.

4.4.4 Description of assembler options

Each off the assembler options is described in detail in the following pages.

-C chip

(1) -C

Description Format	-C device
Default Interpretation	Cannot be omitted

[Function]

- -C option specifies the product subject to assembly.

[Use]

- -C option must always be specified. The assembler performs assembly appropriate to the product specified by -C option.

[Description]

- The ROM range, RAM range, instruction set, reserved words (specified address name symbols), etc., vary according to the 75X Series and 75XL Series product (device). Products which can be specified by the -C option and the ROM and RAM ranges of each product are shown in Table 4-4.

Table 4-4 Assemble Object Device List

(1) 75X Series

Assembly Target Product	Product Specification	ROM Range	RAM Range
μPD75000	000	0H to 3FFFH	0H to 0F7FH
μPD75000A	000A	0H to 0F7FH	0H to 0F7FH
μPD75004	004	0H to 0FFFH	0H to 01FFH
μPD75006	006	0H to 177FH	0H to 01FFH
μPD75008, 75P008	008	0H to 1F7FH	0H to 01FFH
μPD75028	028	0H to 1F7FH	0H to 01FFH
μPD75036, 75P036	036	0H to 3F7FH	0H to 03FFH
μPD75048, 75P048	048	0H to 1F7FH	0H to 01FFH, 400H to 07FFH Note
μPD75064	064	0H to 0FFFH	0H to 01FFH
μPD75066	066	0H to 177FH	0H to 01FFH
μPD75068, 75P068	068	0H to 1F7FH	0H to 01FFH
μPD75104, 75104A	104	0H to 0FFFH	0H to 013FH
μPD75106	106	0H to 177FH	0H to 013FH
μPD75108, 75108A, μPD75P108B, 75108F	108	0H to 1F7FH	0H to 01FFH
μPD75P108	P108	0H to 1FFFH	0H to 01FFH
μPD75112, 75112F	112	0H to 2F7FH	0H to 01FFH
μPD75116, 75P116, 75116F	116	0H to 3F7FH	0H to 01FFH
μPD75116H	116H	0H to 3F7FH	0H to 02FFH
μPD75117H, 75P117H	117H	0H to 5F7FH	0H to 02FFH
μPD75206	206	0H to 177FH	0H to 013FH Note 1
μPD75208	208	0H to 1F7FH	0H to 01BFH Note 1
μPD75CG208	CG208	0H to 1FFFH	0H to 01BFH Note 1
μPD75212A	212A	0H to 2F7FH	0H to 01FFH

Note Addresses 0400H to 07FFH are allocated by EEPROM.

-C chip

Assembly Target Product	Product Specification	ROM Range	RAM Range
μPD75216A, 75P216A	216A	0H to 3F7FH	0H to 01FFH
μPD75CG216A	CG216A	0H to 3 FFFH	0H to 01FFH
μPD75217	217	0H to 5F7FH	0H to 02FFH
μPD75218, 75P218	218	0H to 7F7FH	0H to 03FFH
μPD75236	236	0H to 3F7FH	0H to 02FFH
μPD75237	237	0H to 5F7FH	0H to 03FFH
μPD75238, 75P238	238	0H to 7F7FH	0H to 03FFH
μPD75268	268	0H to 1F7FH	0H to 01FFH
μPD75304, 75304B	304	0H to 0FFFH	0H to 01FFH Note 2
μPD75306, 75306B	306	0H to 177FH	0H to 01FFH Note 2
μPD75308, 75P308, 75308B	308	0H to 1F7FH	0H to 01FFH Note 2
μPD75312	312	0H to 2F7FH	0H to 01FFH Note 2
μPD75312B	312B	0H to 2F7FH	0H to 03FFH Note 2
μPD75316, 75P316	316	0H to 3F7FH	0H to 01FFH Note 2
μPD75P316A	316A	0H to 3F7FH	0H to 03FFH Note 2
μPD75316B, 75P316B	316B	0H to 3F7FH	0H to 03FFH Note 2
μPD75328, 75P328	328	0H to 1F7FH	0H to 01FFH Note 3
μPD75336. 75P336	336	0H to 3F7FH	0H to 02FFH Note 3
μPD75352A	352A	0H to 2F7FH	0H to 03FFH Note 4
μPD75402, 75P402	402	0H to 077FH	0H to 003FH
μPD75512	512	0H to 2F7FH	0H to 01FFH
μPD75516, 75P516	516	0H to 3F7FH	0H to 01FFH
μPD75517	517	0H to 5F7FH	0H to 03FFH
μPD75518, 75P518	518	0H to 7F7FH	0H to 03FFH
μPD75617A	617A	0H to 5F7FH	0H to 05FFH Note 4

Notes 1. Display memory means an area comprising a total of 196 bits, consisting of RAM addresses 1C0H to 1FFH with the exception of 1C3H, 1C7H, 1CBH, 1CFH, 1D3H, 1D7H, 1DBH, 1DFH, 1E3H, 1E7H, 1EBH, 1EFH, 1F3H, 1F7H and 1FBH.

2. 8-bit data transfer instructions (MOV XA, mem / MOV mem, XA / XCH XA mem) cannot be use on addresses in the range 01E0H to 01FFH.

3. 8-bit data transfer instructions (MOV XA, mem / MOV mem, XA / XCH XA. mem) cannot be used on addresses in the range 01E8H to 01FFH.

4. 8-bit data transfer instructions (MOV XA, mem / MOV mem, XA / XCH XA, mem) cannot be used on addresses in the range 0100H to 0126H.

-C chip

(2) 75XL Series

Assembly Target Product	Product Specification	ROM Range	RAM Range
μPD750004	0004	0H to 0FFFH	0H to 1FFH
μPD750006	0006	0H to 17FFH	0H to 1FFH
μPD750008	0008	0H to 1FFFH	0H to 1FFH
μPD75P0016	P0016	0H to 3FFFH	0H to 1FFH
μPD750104	0104	0H to 0FFFH	0H to 1FFH
μPD750106	0106	0H to 17FFH	0H to 1FFH
μPD750108	0108	0H to 1FFFH	0H to 1FFH
μPD75P0116	P0116	0H to 3FFFH	0H to 1FFH
μPD750064	0064	0H to 0FFFH	0H to 1FFH
μPD750066	0066	0H to 17FFH	0H to 1FFH
μPD750068	0068	0H to 1FFFH	0H to 1FFH
μPD75P0076	P0076	0H to 3FFFH	0H to 1FFH
μPD753012	3012	0H to 2FFFH	0H to 3FFH Note 2
μPD753016	3016	0H to 3FFFH	0H to 3FFH Note 2
μPD753017	3017	0H to 5FFFH	0H to 3FFH Note 2
μPD75P3018	P3018	0H to 7FFFH	0H to 3FFH Note 2
μPD753012A	3012A	0H to 2FFFH	0H to 3FFH Note 2
μPD753016A	3016A	0H to 3FFFH	0H to 3FFH Note 2
μPD753017A	3017A	0H to 5FFFH	0H to 3FFH Note 2
μPD75P3018A	P3018A	0H to 7FFFH	0H to 3FFH Note 2
μPD753036	3036	0H to 3FFFH	0H to 2FFH Note 3
μPD75P3036	P3036	0H to 3FFFH	0H to 2FFH Note 3
μPD753104	3104	0H to 0FFFH	0H to 1FFH Note 4
μPD753106	3106	0H to 17FFH	0H to 1FFH Note 4
μPD753108	3108	0H to 1FFFH	0H to 1FFH Note 4
μPD75P3116	P3116	0H to 3FFFH	0H to 1FFH Note 4
μPD753204	3204	0H to 0FFFH	0H to 1FFH Note 5
μPD753206	3206	0H to 17FFH	0H to 1FFH Note 5
μPD753208	3208	0H to 1FFFH	0H to 1FFH Note 5
μPD75P3216	P3216	0H to 3FFFH	0H to 1FFH Note 5
μPD753304 Note 1	3304	0H to 0FFFH	0H to 0FFH 1E0H to 1F7H Note 6
μPD754144	4144	0H to 07FFH	0H to 07FH 0400H to 041FH Note 7
μPD754244	4244	0H to 0FFFH	0H to 07FH 0400H to 041FH Note 7
μPD754264	4264	0H to 0FFFH	0H to 07FH 0400H to 041FH Note 7
μPD75F4264 Note 1	F4264	0H to 0FFFH	0H to 07FH 0400H to 041FH Note 7
μPD754302	4302	0H to 07FFH	0H to 0FFH
μPD754304	4304	0H to 0FFFH	0H to 0FFH
μPD75P4308	P4308	0H to 01FFFH	0H to 0FFH

-C chip

Notes 1. Under development.

2. The addresses 1F0H-1FFH are allocated by the display memory.

3. The addresses 1ECH-1FFH are allocated by the display memory.

4. The addresses 1E0H-1F7H are allocated by the display memory.

5. The addresses 1ECH-1F7H are allocated by the display memory.

6. The addresses 1E0H-1F7H are allocated by the display memory.

7. Addresses 0400H-041FH are allocated by EEPROM.

[Examples]

Example 1. When the -C option is omitted.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM
75X Series Assembler VXX.XX [XX Xxx xx]
   Copyright (C) NEC Corporation 1985, XXXX

***ERROR A099 CHIP IS NOT SELECTED
   –C000A/000/004/006/008/028/036/048/064/066/068/
     104/106/108/P108/112/116/116H/117H
     206/208/CG208/212A/216A/CG216A/217/218/236/237/238/268/
     304/306/308/312/312B/316/316A/316B/328/336/352A
     402/
     512/516/517/518/
     617A

     0004/0006/0008/P0016/
     3012/3016/3017/P3018/3104/3106/3108/P3116   } Note

Program aborted
```

Note The installed 75XL Series device files are displayed.
In this example, an error is generated since the -C option is omitted, and program execution is aborted.

Example 2. When a μPD75104/75104A source program is assembled.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C104
```

(2) -M

Description Format	-M mode
★ Default Interpretation	Omission Impossible (75XL Series)

[Function]

- The -M option specifies the 75XL Series CPU mode.

[Use]

- Assembly is performed in accordance with the CPU mode specified by -M.
- An error will be flagged if this option is not specified when a 75XL Series device is used.
- ★ In the 75XL Series, the Mark1 Mode cannot be specified for products with more than 16 Kbytes of ROM.
- This option cannot be specified when a 75X Series device is used.

[Description]

- ★ The modes that can be selected with this option are shown in the table below.

Option	Specified Mode	Model		
		75X Series	75XL Series (ROM is less than 16 Kbytes)	75XL Series (ROM is 16 Kbytes or more)
-M2	Mark2	∴	○	○
-M1	Mark1	∴	○	∴
-M0	Common	∴	○	○

○ : Can be specified.

∴ : Cannot be specified.

- In the Mark1 mode, The BRA and CALLA instructions result in an error.
- In the common mode, an instruction in which there is a change in the stack results in an error.

-M MODE

[Examples]

Example 1. If **"75XTEST1.ASM"** is assembled with the -M2 option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -0004 -M2
```

Example 2. If **"75XTEST1.ASM"** is assembled without the -M option being specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C0004
75X Series Assembler VXX. XX [XX Xxx xx]
   Copyright (C) NEC Corporation 1985 ,XXXX

*** ERROR A099 MODE IS NOT SELECTED
Program aborted
```

→ An assembly error is flagged.

(3) -O/-NO

Description Format	-O [output file name] -NO
Default Interpretation	'Input file name.REL' is created in current path

[Function]

- The -O option specifies the output destination and file name of the object module file output by the assembler.
- The -NO option specifies that no object module file is to be created.

[Use]

- The -O option is specified when it is wished to change the object module file output destination or file name.
- The -NO option is specified when assembly is to be performed only in order to output an assembly list, etc. (the assembly time is reduced).

[Description]

- When the -O option is specified and the output file name is omitted, the output file name 'source module file name.REL' is taken as being specified.
- If the path name is omitted from the file name specification, the current path is taken as being specified.

-O/-NO	object/no object

[Examples]

Example 1. If **"75XTEST1.ASM"** is assembled with the -NO option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -NO
```

→ An object module file is not output.
The Assemble List File **"75XTEST1.PRN"** only is output.

Example 2. If **"75XTEST1.ASM"** is assembled with the -O option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -OSAMPLE.REL
```

→ The object module file **"SAMPLE.REL"** and the Assemble List file **"75XTEST1.PRN"** are output.

(4) -J/-NJ

Description Format	-J
	-NJ
Default Interpretation	-NJ

[Function]

- The -J option specifies that an object module file is to be created even if there is an assembly error.
- The -NJ option specifies that an object module file is not to be created if there is an assembly error.

[Use]

- When generating an object file even when there is an error in the source file, specify the -J option.

[Explanation]

★ • When the -NO option is specified, the -J option is invalid.

[Examples]

- If **"75XTEST1.ASM"** is assembled with the -J option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -J
```

→ The object module file **"75XTEST1.REL"** is output even when there is an assembly error.

(5) -G/-NG

Description Format	-G
	-NG
Default Interpretation	-G

[Function]

- The -G option specifies that symbol information is to be output to the object module file output by the assembler.
- The -NG option specifies that symbol information is not to be output to the object module file.

[Use]

- When the -NG option is specified, the necessary symbol information is not output at the link list file output by the linker or the symbol table file which is input to the debugger (IE-75000-R Note 1, IE-75001-R, EVAKIT-75X Note 2). Therefore, when symbolic debugging is to be performed, all modules to be linked should be assembled with the -G option specified.
- If symbol information is not required and it is wished to shorten the assembly time if only by a little, the -NG option should be specified.

Notes 1. Maintenance product (not available for purchase)

2. Discontinued (not available for purchase)

[Description]

- When the -NO option was specified, the -G option is invalid.
- When the -NG option is specified, symbol information is not output at the object module file output by the assembler. Therefore, when the object module files output at this time is linked, symbol information is not output to the link list file output by the linker or the symbol table list output by the object converter either.

-G/-NG debug/no debug

[Examples]

- Assembly of **“75XTEST1.ASM”** with the -G option specified

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -G
```

- Assembly of **“75XTEST2.ASM”** with the -NG option specified .

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST2.ASM -C106 -NG
```

To link **“75XTEST1.REL”** and **“75XTEST2.REL”**

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2. REL
```

-G/-NG debug/no debug

→Symbols are not displayed in the symbol list in the link list file for **“75XTEST2.ASM”** for which the -NG option was specified.

```
SYMBOL LIST FOR 75XTEST1.LNK

TYPE      VALUE      ATTRIBUTE NAME
-------   ---------  ----------  --------
--------  ---------  MODULE    AD_MAIN
CODE      0046H      SYMBOL    HEIKIN
PBIT      0FBCH.1    SYMBOL    IET0
CODE      0060H      SYMBOL    LOOP1
CODE      0066H      SYMBOL    LOOP2
CODE      0079H      SYMBOL    LOOP3
CODE      007DH      SYMBOL    LOOP4
CODE      0048H      SYMBOL    LOOP5
CODE      0050H      SYMBOL    MAIN
PBIT      0FB0H.1    SYMBOL    MBE
DATA      0FB3H      SYMBOL    PCC
PBIT      0PB0H.0    SYMBOL    RBE
DATA      0110H      PUBLIC    SEG0
CODE      0020H      PUBLIC    SEG1.
CODE      0050E      PUBLIC    SEG2
CODE      0046H      PUBLIC    SEG3
CODE      0020H      PUBLIC    SEL15
DATA      0F80H      SYMBOL    SP
DATA      0110H      PUBLIC    TDATA
DATA      0FA0H      SYMBOL    TMO
DATA      0FA6H      SYMBOL    TMOD0

LINK COMPLETE, NO ERROR FOUND
```

★ **(6) -GA/-NGA**

Description format	-GA
	-NGA
Default Interpretation	-GA

[Function]

- The -GA option instructs to output object module files, output by the Assembler, with source debugging information added.
- The -NGA option instructs to output object module files, output by the Assembler, without source debugging information added.

[Use]

- Specify the -NGA option when desiring to generate object module files without source debugging information added.

[Description]

- When the -NO option is specified, the -GA option becomes invalid.

[Example]

- Assemble **"75XTEST1.ASM"** with the -NGA option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -NGA
```

→The object module file **"75XTEST1.REL"** is output without source debugging information added.

Caution IE-75000-R and IE-75001-R do not support source debugging, so specify the -NGA option.

(7) -P/-NP

Description Format	-P [output file name] -NP
Default Interpretation	'input file name.PRN' is created in current path

[Function]

- The -P option specifies the output destination and file name of the assembly list file output by the assembler.
- The -NP option specifies that no assembly list file is to be created.

[Use]

- The -P option is specified when it is wished to change the assembly list file output destination or file name.
- The -NP option is specified when assembly is to be performed only in order to output an object module file, etc. (the assembly time is reduced).

[Description]

- If the drive name is omitted from the file name specification, the current path name is taken as being specified.
- The following can be specified as the device type output destination:
  - -PPRN ..... Assembly list is output to line printer.
  - -PCON ..... Assembly list is output to console.
  - -PAUX ..... Assembly list is output to RS-232-C.
  - -PNUL ..... Assembly list is not output.
- An error list file can be output separately by means of the -E option.
- If the -NP option is specified, the following options are invalid.
  - -KS, -KX, -LL, -LW, -KA, -LT

-P/-NP print/no print

[Examples]

Example 1. If **"75XTEST1.ASM"** is assembled with the -NP option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -NP
```

→Assemble List file is not output.
Only the object module file **"75XTEST1.REL"** is output.

Example 2. If **"75XTEST1.ASM"** is assembled with the -P option specified (the file name is **"SAMPLE.PRN"**).

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -PSAMPLE.PRN
```

→The Assemble List File **"SAMPLE.PRN"** and the object module file **"75XTEST1.REL"** are output.

Example 3. To output the list to the printer

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -PPRN
```

→The assembly list file is output to the printer.

-E/-NE error print/no error print

(8) -E/-NE

Description Format	-E [output file name]
	-NE
Default Interpretation	-NE

[Function]

- The -E option specifies error list file output, and the output destination and filename.
- The -NE option specifies that no error list file is to be output.

[Use]

- When the assembly list is very long, it is difficult to find error lines in the list. In this case, the -E option can be specified to extract only assembly error information.

[Description]

- If the output file name is omitted when the -E option is specified, “source module file name.ERA” is taken as being specified as the output file name.
- If the drive name is omitted from the file name specification, the current path name is taken as being specified.
- If the same output file name as the specified by the -P option is specified, an error list is not output.
- The following can be specified as the device type file output destination.
  -EPRN ..... Error list is output tp line printer.
  -ECON ..... Error list is output to console.
  -EAUX ..... Error list is output to RS-232-C.
  -ENUL ..... Error list is not output.

[Example]

- If **“75XTEST1.ASM”** is assembled with the -E option specified (the file name is **“75XTEST.ERA”**).

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -E75XTEST.ERA
```

→If there was an error, the Error List File **“75XTEXT.ERA”** is output.

(9) -KS/-NKS

Description Format	-KS
	-NKS
Default Interpretation	-NKS

[Function]

- The -KS option specifies that a symbol table list is to be output to the assembly list file.
- The -NKS option specifies that a symbol table list is not to be output.

[Use]

- The -KS option is specified when it is wished to list the symbol name, symbol attribute, value, etc., of all symbols defined in the source module.

[Description]

- When the -KS option is specified, a symbol table list is output after the assembly list in the assembly list file.
- If the -NP option is specified, the -KS option is invalid and a symbol table list is not output.

[Example]

- If **"75XTEST1.ASM"** is assembled with the -KS option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -KS
```

→The following symbol table list is output in the assembly list file.

```
75X SERIES ASSEMBLER VX.XX                                    XX/XX/XX XX:XX:XX PAGE : X

   ** A-D CONVERTER VX.XX                                **

   SYMBOL TABLE LIST

OFFSET   TYPE         SYMBOL    OFFSET TYPE         SYMBOL   OFFSET  TYPE         SYMBOL   OFFSET  TYPE         SYMBOL

---------- ---------  AD_MAIN   --------- CODE  EXT ADCONV   0000H   CODE         HEIKIN   0FBCH.1 PBIT         IETO
0010H    CODE         LOOP1     0016H    CODE        LOOP2    0029H   CODE         LOOP3    002DH   CODE         LOOP4
0002H    CODE         LOOP5     0000H    CODE        MAIN     0FB0H.1 PBIT         MBE      0FB3H   DATA         PCC
0FB0H.0  PBIT         RBE       0002H    DATA  PUB   SEG0     0002H   CODE  PUB    SEG1     0039H   CODE  PUB    SEG2
000AH    CODE  PUB    SEG3      0000H    CODE  PUB   SEL15    -------  CODE  EXT   SIOSUB   0F80H   DATA         SP
---------- STACK EXT  STACK     0110H    DATA  PUB   TDATA    0FA0H   DATA         TM0      0FA6H   DATA         TMOD0

TARGET CHIP: UPD75106
STACK SIZE = 000AH

ASSEMBLY COMPLETE, NO ERROR FOUND
```

(10) -KX/-NKX

Description Format	-KX
	-NKX
Default Interpretation	-NKX

[Function]

- The -KX option specifies that a cross-reference list is to be output to the assembly list file.
- The -NKX option specifies that a cross-reference list is not to be output.

[Use]

- The -KX option is specified when it is wished to ascertain such information as how often a symbol defined in the source module file list is referenced in the source module, in which lines of coding in the assembly list that symbol has been referenced, and so forth.
- For example, if the location at which a symbol which defines a subroutine entry address is known, it is possible to find immediately where that subroutine was called.

[Description]

- If the -NP option is specified, the -KX option is invalid.
- The cross-reference list is output at the end of the assembly list file (a file containing only the cross-reference list is not output).

[Example]

- If **“75XTEST1.ASM”** is assembled with the -KX option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -KX
```

-KX/-NKX cross-reference/no cross-reference

→The following type of cross reference list is output in the Assemble List File.

```
75X SERIES ASSEMBLER VX.XX                                XX/XX/XX XX:XX:XX PAGE:   X

      ** A-D CONVERTER VX. XX                                    **

      CROSS REFERENCE LIST

SYMBOL    TYPE       VALUE                          ATTRIBUTES XREF LIST

AD_MAIN   ----------  ----------                     1
ADCONV    CODE       ----------      EXT            6, 10
HEIKIN    CODE       0000H       R      SEG = SEG3  60, #68
IET0      PBIT       0FBCH.1                        53
LOOP1     CODE       0010H       R      SEG = SEG2  #37, 39
LOOP2     CODE       0016H       R      SEG = SEG2  #41, 43
LOOP3     CODE       0029H       R      SEG = SEG2  #56, 63
LOOP4     CODE       002DH       R      SEG = SEC2  #58, 59
LOOP5     CODE       0002H       R      SEG = SEG3  #69, 75
MAIN      CODE       0000H       R      SEG = SEG2  9, #23
MBE       PBIT       0FB0H.1                        9, 10
PCC       DATA       0FB3H                          30
RBE       PBIT       0FB0H.0                        9, 10
SEG0      DATA       0002H         PUB ABS          #12
SEG1      CODE       0002H         PUB REL = IENT   #17
SEG2      CODE       0039H         PUB REL = INBLOCK #22
SEG3      CODE       000AH         PUB REL = SENT   #67
SEL15     CODE       0000H       R PUB SEG = SEG1   7, #18, 25, 47
SIOSUB    CODE       ---------     EXT              6, 62
SP        DATA       0F80H                          27
STACK     STACK      ---------     EXT              26
TDATA     DATA       0110H         PUB ABS          7, #13, 61
TM0       DATA       0FA0H                          51
TMOD0     DATA       0FA6H                          49

 TARGET CHIP : UPD75106
 STACK SIZE = 000AH

ASSEMBLY COMPLETE, NO ERROR FOUND
```

-CA/-NCA cap/no cap

★ **(11) -CA/-NCA**

Description Format	-CA
	-NCA
Default Interpretation	-NCA

[Function]

- The -CA option specifies not to distinguish the upper and lower case letters of symbol name.
- The -NCA option specifies to distinguish the upper and lower case letters of symbol name.

[Use]

- Use the -CA option in cases where upper and lower case letters are not distinguished and specify the -NCA option when distinguishing upper and lower case letters.

[Explanation]

- If neither the -CA option or the -NCA option is specified, processing is the same as when the -NCA option is specified.

[Description Example]

- Assemble “75XTEST1.ASM” with the -CA option specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -CA
```

→The object module file **“75XTEST1.REL”** is output without upper and lower case letters in symbol names distinguished.

Caution Upper and lower case letters in symbol names cannot be judged by the IE-75000-R and IE-75001-R, so specify the -CA option.

★ **(12) -S/-NS**

Description Format	-S
	-NS
Default Interpretation	-S

[Function]

- The -S option instructs to expand the length of recognizable symbol names to a maximum of 31 characters.
- The -NS option instructs to invalidate the -S option and allow the length of recognizable symbol names to be a maximum of 8 characters.

[Use]

- Specify the -S option when making the length of recognizable symbol names a maximum of 31 characters and specify the -NS option when making the length of recognizable symbol names a maximum of 8 characters.

[Explanation]

- If neither the -S or -NS option is specified, processing is the same as when the -S option is specified.

[Description Example]

- Assemble **"75XTEST1.ASM"** specifying the -NS option.

```
A:\NECTOOLS\SMP75X\RA75X\RA75X 75XTEST1.ASM -C106 -NS
```

→The object module file **"75XTEST1.REL"** is output with the symbol names a maximum of 8 characters in length.

Caution The IE-75000-R and IE-75001-R can recognize only symbol names with a length of 8 characters, so specify the -NS option.

★ **(13) -D/-ND**

Description Format	-D Symbol Name [= Numerical Value][, Symbol Name [= Numerical value][…]] -ND Symbol Name [, Symbol Name […]]
Default Interpretation	A symbol is not defined.

[Function]

- The -D option instructs that the specified symbol be defined with the value of the specified numerical value.
- The -ND option invalidates the specified symbol definition.

[Use]

- If you are defining a specified symbol with the value of the specified numerical value, specify the -D option. If you are invalidating the definition of the specified symbol, specify the -ND option.

[Explanation]

- If specification of a numerical value is omitted, the symbol value becomes 1.

[Description Example]

- If defining 1 in the symbol ‘TRUE.’

```
A:\NECTOOL\SMP75X\RA75X\RA75X>RA75X 75XTEST1.ASM -DTRUE=1
```

(14) -LL/-LW

Description Format	-LL number of lines printed on one page
	-LW number of columns printed on one line
Default Interpretation	-LL66
	-LW132

[Function]

- The -LL option specifies the number of lines per page in the assembly list.
- The -LW option specifies the number of columns per line in the assembly list.

[Use]

- The -LL and -LW option are specified when it is wished to change the number of lines to be printed on one page or the number of columns to be printed in one line of the assembly list.

[Description]

- The number of lines and columns which can be specified by these options are as follows:
  20 - number of print lines per page - 65535
  72 - number of print columns per line - 256
- If the number of characters of per line for which output to the assembly list file is attempted exceeds the value specified by the -LW option, the assembler truncates the characters exceeding the specified number of columns before outputting the line to the list file.
- The number of lines actually printed on one page of the assembly list file is (number of lines specified by -LL option - 6), as a 3-line margin is left at the top and bottom of each page of the assembly list.

-LL/-LW page length/page width

[Example]

Example 1. If **"75XTEST1.ASM"** is assembled with printing of 40 lines per page and 80 characters per line specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -LL40 -LW80
```

→The following type of assembly list is output.

80

```
75X SERIES ASSEMBLER VX.XX                                  XX/XX/XX XX:XX:XX  PAGE :  X
   ** A-D CONVERTER VX.XX                                                          **
COMMAND          : 75XTEST1.ASM -C106 -LL40 -LW80

STN0  ADRS  R   OBJECT      IC  MAC    SOURCE STATEMENT

   1                                    $         TITLE='A-D CONVERTER VX. XX'
   2                                    ;************************************************************
   3                                    ;***      A-D CONVERT PROGRAM                  ***
   4                                    ;************************************************************
   5                                              NAME   AD_MAIN
   6                                              EXTRN  CODE(ADCONV),CODE(SIOSUB)
   7                                              PUBLIC TDATA,SEL15
   8                                              STKLN  10
   9   0000    R   C000                           VENT0  MBE=1,RBE=1,MAIN
  10   0008    E   8000                           VENT4  MBE=1,RBE=0,ADCONV
  11
  12   ----                         SEG0    DSEG   1 AT 10H
  13   0110                         TDATA:  DS     2
  14
  15                                   ;***      GETI TABLE      ***
  16
  17   ----                         SEG1    CSEG   IENT
  18   0000        991F             SEL15:  SEL    MB15
  19
  20                                   ;***      MAIN ROUTINE  ***
  21
  22   ----                         SEG2    CSEG   INBLOCK
  23   0000        9921             MAIN:   SEL    RB1
  24
  25   0002    R   00                         GETI   SEL15            ;STACK POINTER SET
```

40

-LL/-LW page length/page width

```
75X SERIES ASSEMBLER VX.XX                                XX/XX/XX  XX:XX:XX  PAGE :  X

   ** A-D CONVERTER VX. XX                                                        **

STN0  ADRS  R   OBJECT      IC  MAC    SOURCE STATEXENT

  26   0003   E  8900                          MOV     XA,#STACK       ;
  27   0005      9280                          MOV     SP,XA           ;
  28
  29   0007      73                            MOV     A,#0011B
  30   0008      93B3                          MOV     PCC,A           ;PCC ← 0011B
  31
  32                                ;**       DATA RAM 0H-13FH ZERO CLEAR**
  33
  34   000A      9911                          SEL     MB1
  35   000C      8B3F                          MOV     HL,#3FH
  36   000E      8900                          MOV     XA,#00H
  37   0010      E8                  LOOP1:    MOV     @HL,A           ;100H-13FH
  38   0011      AA6A                          DECS    HL
  39   0013      FC                            BR      LOOP1
  40   0014      9910                          SEL     MB0
  41   0016      E8                  LOOP2:    MOV     @HL,A           ;0H-FFH
  42   0017      AA6A                          DECS    HL
  43   0019      FC                            BR      LOOP2
  44
  45                                ; **      TIMER SET(SAMPLING TIME = 30MSEC, FXX=4
```

Example 2. The -LL and -LW options are omitted.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTESTI.ASM -C106
```

→The assembly list is output as follows.

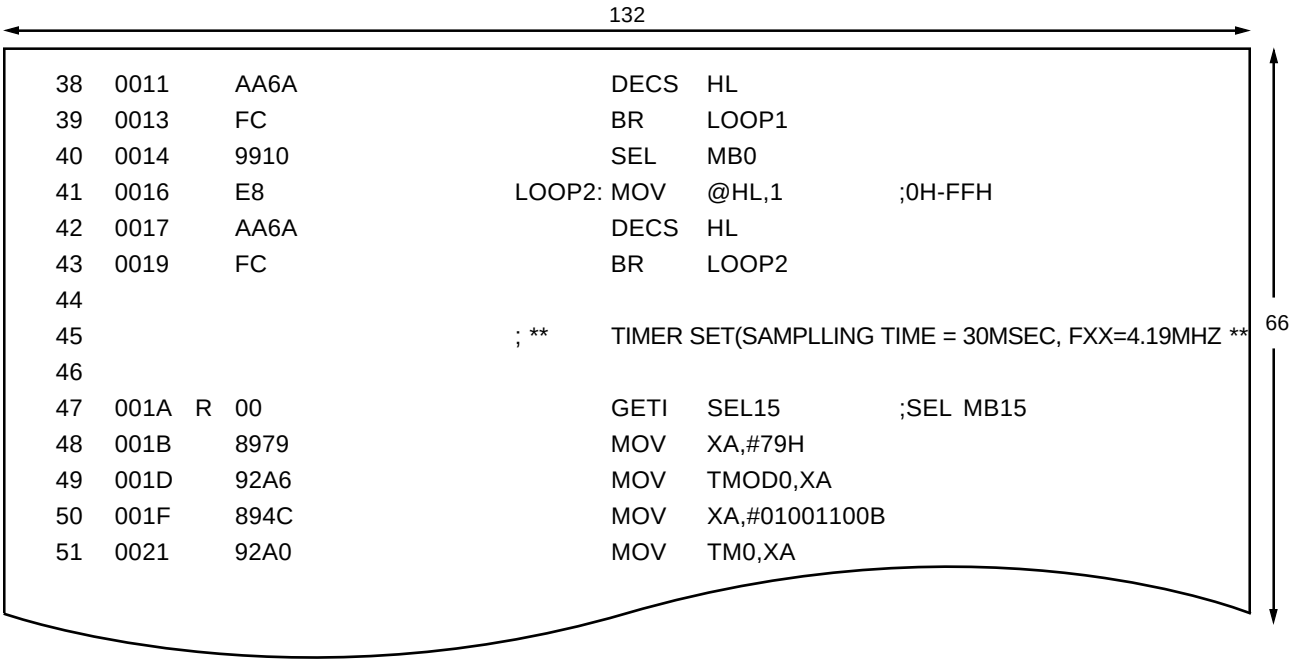
-LL/-LW page length/page width

132

66

```
75X SERIES ASSEMBLER VX.XX                                  XX/XX/XX XX:XX:XX  PAGE :  X
   ** A-D CONVERTER VX.XX                                                          **
COMMAND          : 75XTEST1.ASM -C106

STN0  ADRS  R   OBJECT      IC  MAC   SOURCE STATEMENT

   1                                    $        TITLE='A-D CONVERTER VX. XX'
   2                                    ;************************************************************
   3                                    ;***      A-D CONVERT PROGRAM                  ***
   4                                    ;************************************************************
   5                                             NAME   AD_MAIN
   6                                             EXTRN  CODE(ADCONV),CODE(SIOSUB)
   7                                             PUBLIC TDATA,SEL15
   8                                             STKLN  10
   9   0000    R   C000                          VENT0  MBE=1,RBE=1,MAIN
  10   0008    E   8000                          VENT4  MBE=1,RBE=0,ADCONV
  11
  12   ----                                 SEG0   DSEG   1 AT 10H
  13   0110                                 TDATA: DS     2
  14
  15                                        ;***   GETI TABLE     ***
  16
  17   ----                                 SEG1   CSEG   IENT
  18   0000        991F                     SEL15: SEL    MB15
  19
  20                                        ;***   MAIN ROUTINE  ***
  21
  22   ----                                 SEG2   CSEG   INBLOCK
  23   0000        9921                     MAIN:  SEL    RB1
  24
  25   0002    R   00                            GETI   SEL15          ;STACK POINTER SET
  26   0003    E   8900                          MOV    XA,#STACK       ;
  27   0005        9280                          MOV    SP,XA           ;
  28
  29   0007        73                            MOV    A,#0011B
  30   0008        93B3                          MOV    PCC,A          ;PCC ← 0011B
  31
  32                                        ;**    DATA RAM 0H-13FH ZERO CLEAR**
  33
  34   000A        9911                          SEL    MB1
  35   000C        8B3F                          MOV    HL,#3FH
  36   000E        8900                          MOV    XA,#00H
  37   0010        E8                       LOOP1: MOV    @HL,A          ;100H-13FH
```

-LL/-LW page length/page width




```
 38  0011      AA6A                    DECS    HL
 39  0013      FC                      BR      LOOP1
 40  0014      9910                    SEL     MB0
 41  0016      E8            LOOP2:    MOV     @HL,1           ;0H-FFH
 42  0017      AA6A                    DECS    HL
 43  0019      FC                      BR      LOOP2
 44
 45                          ; **      TIMER SET(SAMPLLING TIME = 30MSEC, FXX=4.19MHZ **
 46
 47  001A  R   00                      GETI    SEL15           ;SEL  MB15
 48  001B      8979                    MOV     XA,#79H
 49  001D      92A6                    MOV     TMOD0,XA
 50  001F      894C                    MOV     XA,#01001100B
 51  0021      92A0                    MOV     TM0,XA
```

-LL/-LW page length/page width

```
75X SERIES ASSEMBLER VX.XX                                XX/XX/XX XX:XX:XX  PAGE :  X

   ** A-D CONVERTER VX.XX                                                          **

STN0  ADRS  R  OBJECT     IC  MAC    SOURCE STATEMENT

  52   0023      9DB2                        EI
  53   0025      9D9C                        EI        IET0
  54
  55   0027      9911                        SEL       MB1
  56   0029      8900                LOOP3:  MOV       XA,#00H
  57   002B      9A0F                        MOV       B,#0H
  58   002D      9A87                LOOP4:  SKE       B,#08H
  59   002F      FD                          BR        LOOP4
  60   0030   R  AB4000                      CALL      !HEIKIN
  61   0033      9210                        MOV       TDATA,XA
  62   0035   E  AB4000                      CALL      !SIOSUB
  63   0038      F0                          BR        LOOP3
  64
  65                                ;***      HEIKIN    (SAMPLE NUMBERS = 8)       ***
  66
  67   ----                SEG3    CSEG      SENT
  68   0000      9A2E      HEIKIN:   MOV       C,#2H
  69   0002      D9        LOOP5:    XCH       A,X
  70   0003      E6                  CLR1      CY
  71   0004      98                  RORC      A
  72   0005      D9                  XCH       A,X
  73   0006      98                  RORC      A
  74   0007      CE                  DECS      C
  75   0008      F9                  BR        LOOP5
  76   0009      EE                  RET
  77
  78                                          END

TARGET CHIP : UPD75106
STACK SIZE = 000AH

ASSEMBLY COMPLETE, NO ERROR FOUND
```

(15) -LT

Description Format	-LT [number of characters]
Default Interpretation	-LT8

[Function]

- This option informs the assembler of the number of space characters (20H) to which TAB codes (09H) in the source module are to be expanded when the assembly list is generated.

[Use]

- TAB codes in the source module are expanded to a number of space characters when output to the assembly list. This option is used to set the maximum number of space characters corresponding to one TAB code at this time.

[Description]

- The number of characters can be specified as a decimal number between 0 and 8. An error will be generated if a number outside this range or a non-numeric value is specified.
- If the parameter is omitted, 8 is taken as being specified.
- If the -NP is specified, the -LT option is ignored.

(16) -KA/-NKA

Description Format	-KA
	-NKA
Default Interpretation	-KA

[Function]

- -KA option specifies assembly list to be output to print file.
- -NKA option specifies assembly list not to be output to print file.

[Use]

- When assembly list is not required, specify -NKA option.
- When assembly list is required while parameter file including -NKA option is being used, specify -KA option after -F option.

[Description]

- The -KA, -NKA options correspond to the 'LIST' control instruction, 'NOLIST' control instruction of assemblers. However, the -KA and -NKA options have an effect on whole source module.
- If the -KA and -NKA options are specified simultaneously, the option specified later is effective.
- The -NKA option takes priority over the 'LIST' control instruction.
- 'NOLIST' control instruction takes priority over the -KA option.
- When the -NP option is specified, the -KA option becomes invalid and no assembly list is output.

-I include path

(17) -I

Description Format	-I path name [, path name...]
Default Interpretation	Search is executed in stipulated search order [Note].

[Function]

- The -I option specifies the search path of an include file specified by ‘INCLUDE’ control instruction.

[Use]

- When an include file is in a different path from the source module file, this option is used to specify that path.

[Description]

- Multiple paths can be specified, separated by commas. In this case, space cannot be inserted before or after the commas.
- The path name cannot be omitted. Also, an error will result if an item other than a path name is specified.
- Up to eight -I options can be specified at one time.
- If multiple paths are specified by an -I option, include files are searched for in the specified order.
- The search path can be set by the environment variable ‘INC75X’ as well as by the -I option. [Note]

Example

```
A>SET INC75X=A:\SRC\HDR\106
```

Note The include file search is conducted in the following order:

(1) When file name specified by ‘INCLUDE’ control instruction does not include a path name
  <1> path in which source module file exists
  <2> path specified by -I option
  <3> path specified by environment variable ‘INC75X’

(2) When the file name specified by ‘INCLUDE’ control instruction includes an absolute path name (beginning with the drive name or ‘\’)
  <1> Path specified by ‘INCLUDE’ control instruction

(3) When file name specified by ‘INCLUDE’ control instruction includes relative path name (beginning with “. .” or beginning with except the drive name or ‘\’)
  <1> Path whose name comprises the name of the path in which the source module file exists followed by the path name specified by the ‘INCLUDE’ control instruction.
  <2> Path whose name comprises the path name specified by the -I option followed by the path name specified by the ‘INCLUDE’ control instruction
  <3> Path whose name comprises the path name specified by the environment variable ‘INC75X’ followed by the path name specified by the ‘INCLUDE’ control instruction

[Examples]

Next the following conditions:

- Source module file name: A:\SRC\UTILS.ASM
- 'INCLUDE' control instruction specification: $INCLUDE PATH\INCFILE.H
- -I option specification: -IB:\WORK
- Environment variable 'INC75X' specification: B:\INC

The include file search order is as follows:

<1> A:\SRC\PATH\INCFILE.H

<2> B:\WORK\PATH\INCFILE.H

<3> B:\INC\PATH\INCFILE.H

That is, the assembler searches these files in order and read the first one found as part of the source module file.

(18) -F

Description Format	-F parameter file name
Default Interpretation	Parameter file not used.

[Function]

- The -F option specifies that the assembler option and input file name are to be read from the file specified by the option parameter. This file is called the parameter file.

[Use]

- Writing options and input file names to be specified for the assembler in a parameter file in advance also reduces the amount of typing required.
- Options and input file names can still be specified in the command line even if a parameter file is used. It is thus possible to write only frequently used options in the parameter file.

[Description]

- The parameter file is a text file, and can be created with an editor, etc. There are no particular restriction on the length of the parameter file.
- The parameter file name cannot be omitted . However, if the file type is omitted, '.PRA' is taken as being specified.
- A logical device name ('CON', 'AUX', etc.) cannot be specified as the parameter file name. Use of such names will result in an error.
- The contents of the parameter file are expanded at the position at which the -F option is written in the assembler start line. It is therefore possible to change the parameter file contents or add other option specifications with options written after the -F option.
- Parameter files cannot be nested. If an -F option is written in the parameter file, an error will result.
- It is not possible to use more than one parameter at one time. If multiple -F options are specified, an error will result.
- Individual options and input file names should be separated by spaces, TABs or Line Feed characters. A parameter file description cannot be split over a number of lines.
- The ';' and '#' symbols are treated as comment marks in the parameter file. Characters from these characters to the end of the line are regarded as a comment.

-F parameter file name

[Examples]

Consider a parameter file [ASM.PRA] with the following contents.

```
75XTEST1.ASM      ; Input file name
-C106             ; Chip is UPD75106
-KS               ; Keep symbol list
```

Example 1. The assembler is started with parameter file **[ASM.PRA]** specified.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X -FASM.PRA
```

Example 2. The contents specified by the parameter file **[ASM.PRA]** are changed and added to in the command line.

```
A:\NECTOOLS\SMP75X\RA75X>RA75X -FASM.PRA -C000 -PPRN
```

★ **(19) -Y**

Description Format	-Y Path Name
Default Interpretation	Executes a search in accordance with the specified search sequence **((2) to (5) of the [Explanation])**.

[Function]

- The -Y option specifies the device file search path.

[Use]

- Specify the -Y option when searching from the specified path first.

[Explanation]

- A device file is searched for by the following sequence.
  (1) Path specified by the -Y option.
  (2) '.. \DEV' path with respect to the RA75X starting path.
  (3) RA75X starting path.
  (4) Current Path
  (5) Environment Variable 'PATH'

[MEMO]

CHAPTER 5. LINKER

The linker (LK75X) has as its input object module files output by the assembler and a library file created by the librarian, and outputs a load module file and link list file.

If a link error occurs, an error message is output to the link list file and the console.

[Main Linker Processing]

<1> Linkage of the object modules in the input file
<2> Determination of segment location addresses
<3> Resolution of relocatable object code
<4> Automatic branch table creation

5.1 Linker Input/Output Files

Linker (LK75X) input/output files are shown in Table 5-1.

Table 5-1 Linker Input/Output Files

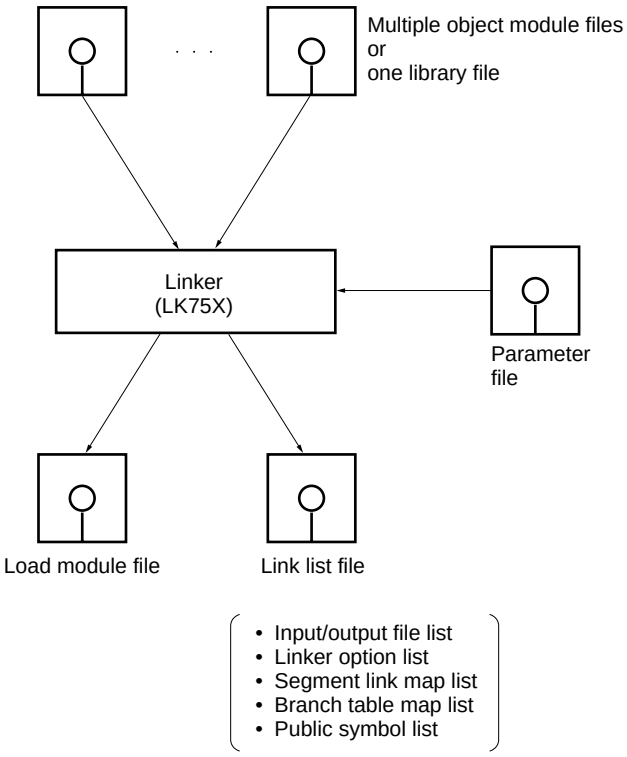
Type of File		Default File Type
Input file	**Object module file** [Note 1] Object module file output by assembler	.REL
	Library file [Notes 1, 2] File created by librarian in which multiple object modules are recorded	.LIB
	Parameter file [Note 3] This file is created with an editor when it is wished to specify a large number of files which cannot be specified in the command line as linker input files.	.PLK
Output file	**Load module file** [Note 1] File containing all information resulting from linkage. The load module file is used as the input file for the object converter (OC75X).	.LNK
	Link list file List file containing linkage information such as input/output file list, linker option list, segment list, etc.	.MAP

Notes 1. Binary file.

2. For details, see **CHAPTER 7 “LIBRARIAN”**.

3. For details, see **(14) -F under 5.4.4 “Description of linker options”**.

Figure 5-1 Linker Input/Output Files



5.2 Linker Functions

- The main functions of the linker are as follows:

1. Linkage of object modules in the input file
2. Determination of segment location addresses
3. Resolution of relocatable object code
4. Automatic branch table creation

- If an error is found during linkage (symbol reference cannot be resolved, segment location not possible, etc.), an error message is output in the segment link map list and to the console.
- The linker performs linkage processing in accordance with the linker options specified when the linker is started. See **5.4 "Linker Options"** for linker options.
- When this processing terminates normally, the linker outputs a termination message and returns control to the OS.
- Maximum linker capabilities are shown below.

Item		Maximum Value
Number of symbols that can be handled by linker	Number of local symbols	No limit
	Number of external definition (PUBLIC) symbols	Approx. 3,000 for all input modules
	Number of external reference (EXTRN) symbols	Approx. 500 per module
Number of segments that can be handled by linker For all input modules: Total of { 1) 2 × number of input 2) Number of segments 3) Number of ORG pseudo-instructions in source program 4) 2 × number of VENT pseudo-instructions		Approx. 250
Number of branch tables that can be created		Approx. 1,000
Maximum number of input files		62

5.2.1 Linkage of object modules in input files

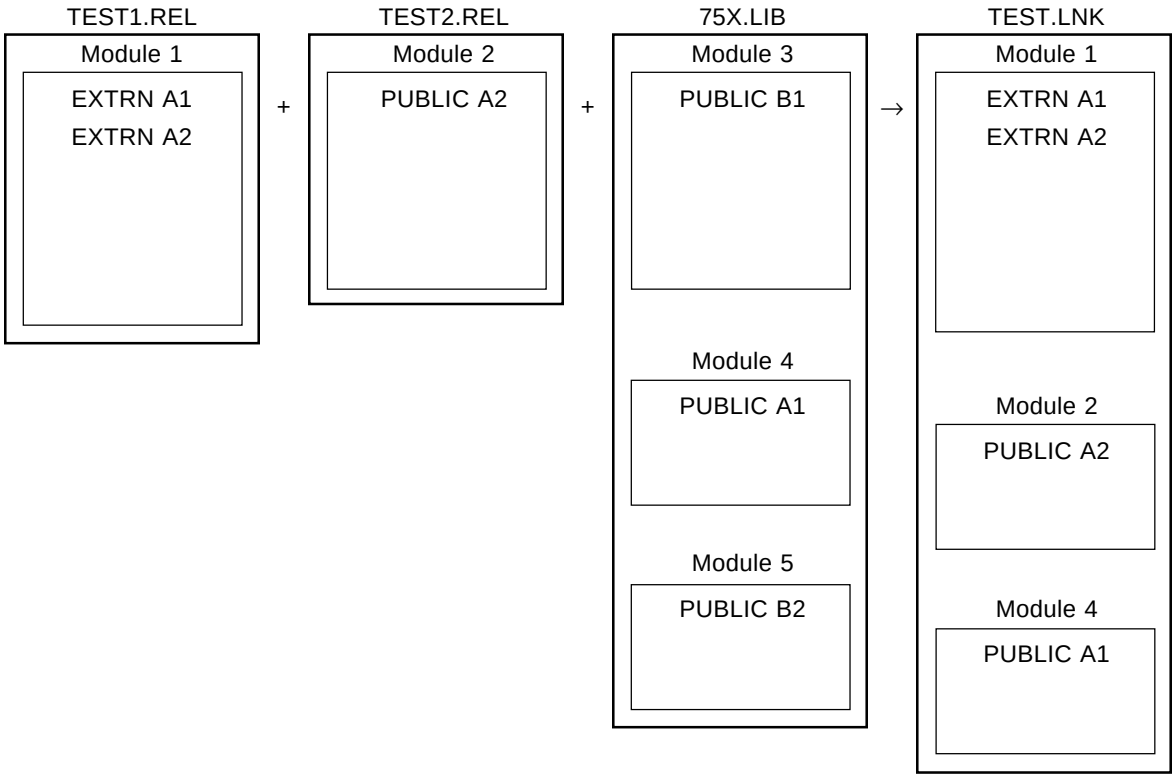
The linker has multiple object module files as input. In addition, one library file can be input together with the object module files.

If there are unresolvable symbol reference relations (by EXTRN and PUBLIC pseudo-instructions) between input object module files, the linker checks the object modules in the library file specified as an input file. Then, if there are object modules in which unresolvable symbols are externally defined in the library file, those modules are extracted from the library file automatically, an linked together with the modules in the object module files specified for input.

Example The linker is started as shown below.

```
A>LK75X B:TEST1.REL B:TEST2.REL B:75X.LIB -OB:TEST.LNK
                    ↑               ↑          ↑
                Object module file  Library file  Load module file
```

In this case, the object modules are linked as shown below.



5.2.2 Determination of segment location address

The linker determines the location addresses of segments in the input object modules.

(1) Order of priority for determining segment location addresses.
<1> Determination of absolute segment location addresses (data segments are all absolute segments)
<2> Determination of relocatable segment location addresses
(a) Determination of the location addresses of segments for which address specification is performed by the linker -CD option (see **(6) -CD 5.4.4 "Description of linker options"** for the -CD options)
(b) Determination of the location addresses of other relocatable segments
The order of priority for determining the location addresses of other relocatable segments specified at assembly time, as shown below.
1) Location address determination for segments with IENT attribute
2) Location address determination for segments with SENT PAGE attribute
3) Location address determination for segments with SENT attribute
4) Location address determination for segment with INBLOCK PAGE attribute
5) Location address determination for segments with XBLOCK PAGE attribute
6) Location address determination for segments with INBLOCK attribute
7) Location address determination for segments with XBLOCK attribute
8) Location address determination for segments with INBLOCKA PAGE attribute
9) Location address determination for segments with XBLOCKA PAGE attribute
10) Location address determination for segments with INBLOCKA attribute
11) Location address determination for segments with XBLOCKA attribute

Location addresses are determined starting with segments with the highest priority shown above (this priority order does not show the order of location addresses).
The location adjustment shown in **Table 5-2 "Segment Relocation Attributes and Location Adjustment"** is performed for segments with the relocation attributes shown above when location address determination is performed.

Table 5-2 Segment Relocation Attributes and Location Adjustment

Relocation Attribute	Location
IENT segment	Located so that the entire segment is within the area 0020H to 007FH. Also, located so that the segment starts at an even address.
SENT segment	Located so that the entire segment is within the area 0000H to 07FFH.
PAGE segment	Located so that the start of the segment is at the start of a page (XX00H).
INBLOCK segment	Located so that the entire segment is within the range OH to 3FFFH, and in the same block.
INBLOCKA segment	Located so that the entire segment is in the same block, as with INBLOCK. However, location is possible within the entire ROM area.
XBLOCK segment	Located so that the entire segment is within the range 0H to 3FFFH. Segment boundary adjustment is not performed as in the case of INBLOCK.
XBLOCKA segment	Located in the same way as for XBLOCK, with location possible within the entire ROM area.
AT absolute expression	The segment is located in the addresses specified by the absolute expression. This segment is only valid for program memory. This segment is also called an ‘absolute segment’.

Segment location prohibited areas can be specified by means of the -RS option.

(2) Segment location address determination method

Absolute segments and segments whose location addresses are specified by the -CD option are located unconditionally starting at the specified address. The area specified by the -RS option is reserved in advance as a segment location prohibited area . There are three methods of determining the location addresses of other relocatable segments, as follows:

- Sequential linkage mode (specified by the -SQ option)
- Random link mode (-RN option specification or mode specification omitted)
- Order specification mode (specified by -CD option)

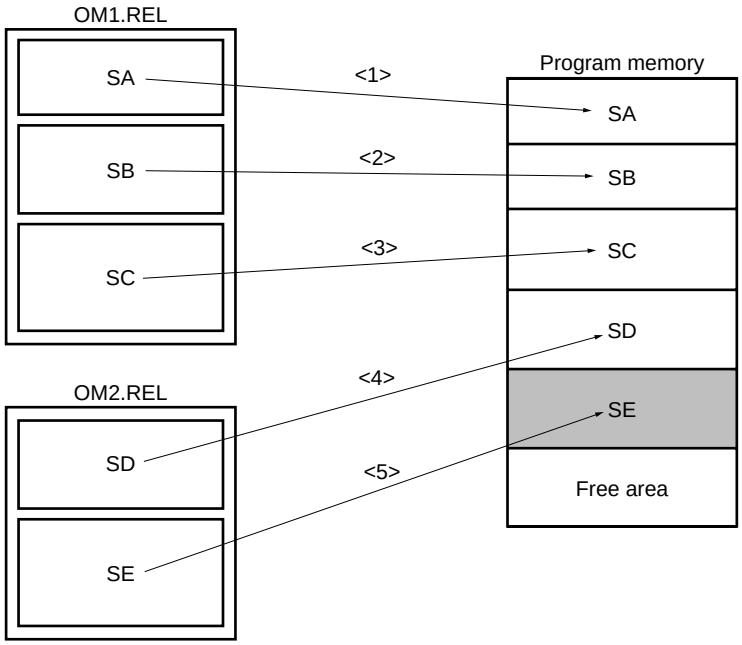
<1> Sequential linkage mode

Location is performed from the low address of the free memory area in the segment input order. The segment input order is the order of the object modules specified for input or, in an object module, the order in which the segments are written in the source.

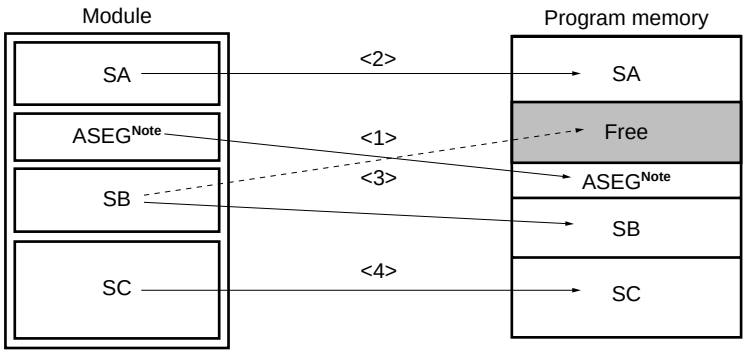
Example 1. The linker is started as shown below.

```
A>LK75X OM1.REL OM2.REL -00M.LNK -SQ
```

In this case the input segments are located as shown below.



Example 2. If there is an absolute segment, performing linkage with the -SQ option specified may result in the creation of a free area in memory as shown below.



Note ASEG is an absolute segment.

In the above case, the location address of the absolute segment is determined first. Therefore, when location of segment SB is attempted after segment SA, if segment SB is larger than the free area between segment SA and ASEG, segment SB is located after ASEG.
In this case, the area between segment SA and ASEG is left as a free area.

<2> Random linkage mode

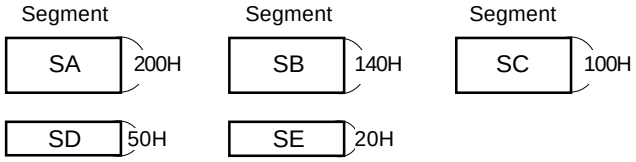
Location is performed so as to avoid the creation of free areas in memory as far as possible, without regard to the segment input order.

Starting with the largest of the input segments, location is performed in the lowest address in which location is possible in the free area left after absolute segment location.

Example Assume that the following segments are input.

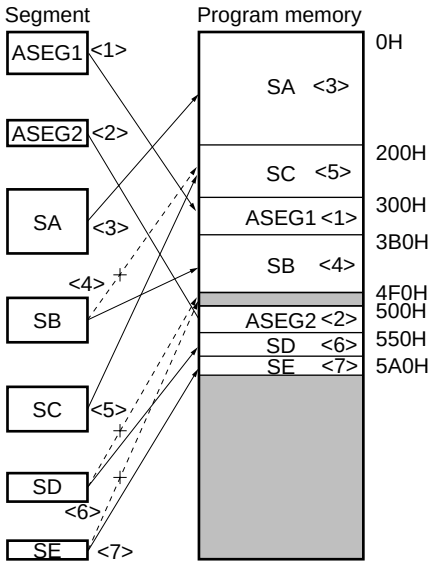
- Absolute segment

- Relocatable segments



(Numbers: segment sizes)

In this case the input segments are located as shown on the next page.



<1> First, the absolute segments are located at the specified addresses (ASEG1: address 300H, ASEG2: address 500H).

<2> The largest of the relocatable segments, segment SA, is located starting at the lowest address of the free area.

<3> The next largest segment, SB, is located. As segment SB is 140H in size, it cannot be located in the first free area (with a size of 100H, from address 200H to address 2FFH). Segment SB is therefore located starting at address 3B0H of the next free area.

<4> Next, segment SC is located. As segment SC is 100H in size, it is located in the first free area (addresses 200H to 2FFH).

<5> Next, segment SD is located. As segment SB is 150H in size, it cannot be located in the first free area (4F0H to 4FFH). Segment SD is therefore located starting at address 550H of the next free area.

<6> Next, segment SE is located. As segment SE is 20H in size, it cannot be located in the first free area (4F0H to 4FFH), and is therefore located starting at address 5A0H of the next free area.

<7> Finally, the shaded areas shown in the above diagram are left as free areas.

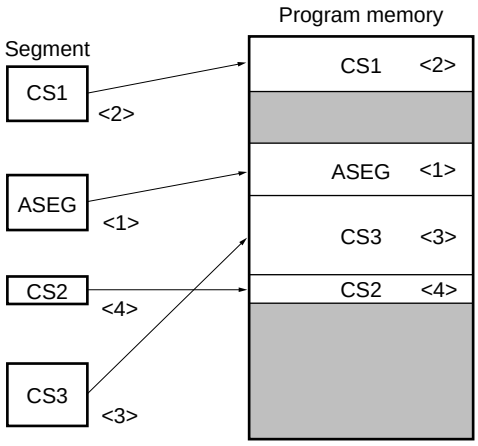
<3> Order specification mode

Location is performed in the segment order specified by the -CD option.

Example The following -CD option is specified when linkage is started.

-CD (CS1, CS3, CS2)

Of the segments input at this time, CS1, CS2, and CS3 are located in the specified order starting at the free area in memory (however, they are not necessarily located in consecutive areas).



Caution

As the order specification mode is specified by the -CD option, segment address specification can be performed at the same time.

Example -CD (C1, C4, C3'300H, C6, C6)

When this specification is made, the order specification mode is interpreted individually before the segment for which address specification has been performed. That is, it is equivalent to the following specification:

-CD (C1, C4) -CD (C3'300H, C6, C5)

However, if there are a large number of segments, and object code is generated to the point of filling the ROM capacity of the product on which assembly is being performed, the processing time will be extremely long. If the correct location method is not found after trying relocation a certain number of times, an error message is output and the operation is aborted.
The following 3 processing methods can be used at this time.

1. **Link the object modules which could not be linked after specifying the -SQ option (an error will of course be generated as a result), decide the location method manually based on the size of the segments in the output map list, then perform linkage again with the location order specified by the -CD option.**
2. **Reduce the number of segments.**
3. **Amend the program so that INBLOCK (INBLOCKA) segments fit exactly within block boundaries.**

(3) Stack segment location address determination

A stack segment is a segment which is reserved as a stack area by the STKLN pseudo-instruction in a source module.

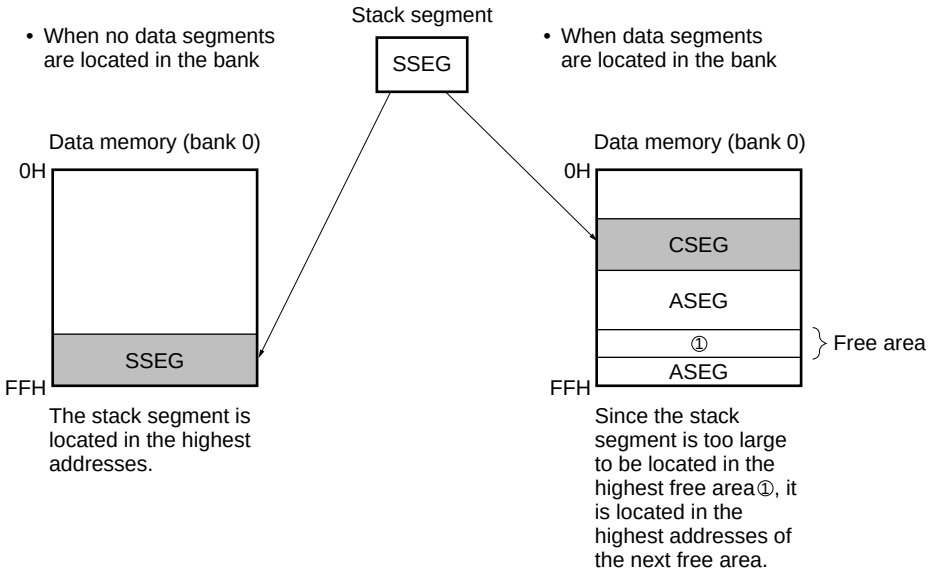
<1> Stack segment linkage

If there are stack segments in the input module, the linker links all the stack modules. Thus the stack segment size after linkage is the sum of the sizes of all the stack segments.
However, the stack segment size after linkage can be changed by the linker -SZ option (see **(10) -SZ under 5.4.4 "Description of linker options "**).

<2> Stack segment location

The stack segment is located starting in the highest addresses in the area comprising addresses 0H to 0FFH (bank 0) of the data memory. If data segments are located in bank 0, the stack segment is located in the highest free area at which location is possible.

Example

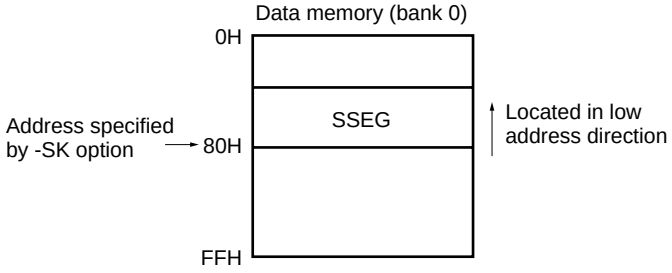


The stack segment location address can also be specified when linkage is performed by means of the -SK option.

Example The following option is specified at linkage time.

-SK80H

In this case the stack segment is located starting at address (address specified by -SK option -1) in the low address direction.



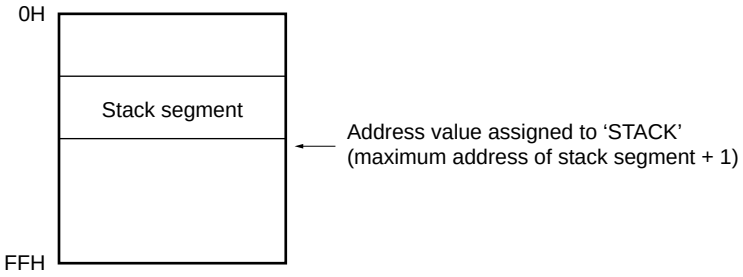
5.2.3 Resolution of relocatable object code

During assembly, temporary values are incorporated in the object code of instructions which reference relocatable symbols or external reference symbols.

The linker amends this object code to the correct values.

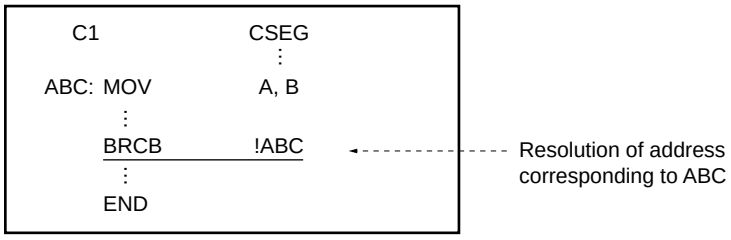
Also, if a source module is written using the reserved word ‘STACK’, the following values are assigned to ‘STACK’ by the linker.

1. **When linker -SK option specified**
   The address value specified by the -SK option is assigned to the reserved word ‘STACK’.
2. **When linker -SK option is not specified**
   The address value (maximum address of stack segment + 1) is assigned to the reserved word ‘STACK’.

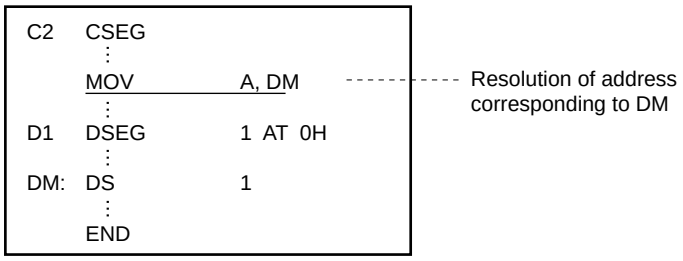


The correct value is also assigned to ‘STACK’ at the time of linkage in the object code of a ‘MOV XA, #STACK’ instruction which sets a value in the stack pointer by using the reserved word ‘STACK’.

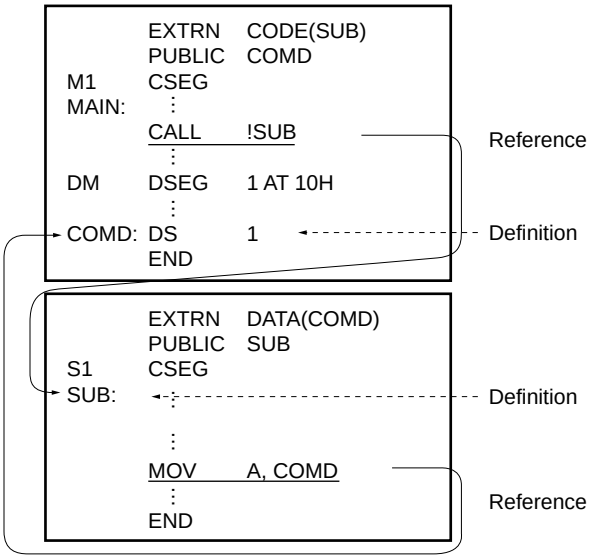
Example 1. When a relocatable item is referenced in the segment



Example 2. Symbol reference in another segment



Example 3. When an external reference name is referenced



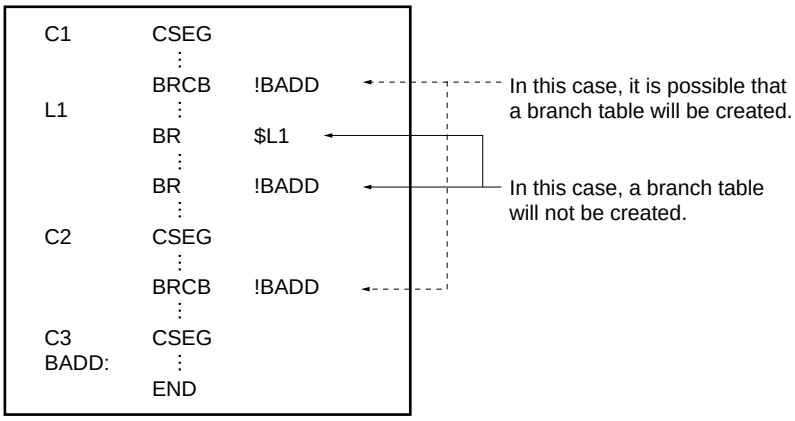
5.2.4 Automatic branch table creation

When a symbol which has an address value in another block (4K bytes from x000H to xFFFH) is referenced by a 2-byte branch instruction (BRCB instruction), the linker automatically creates a 3-byte [Note 2] branch instruction (called a branch table) in the original block[Note 1], enabling the other block to be referenced by the 2-byte branch instruction.

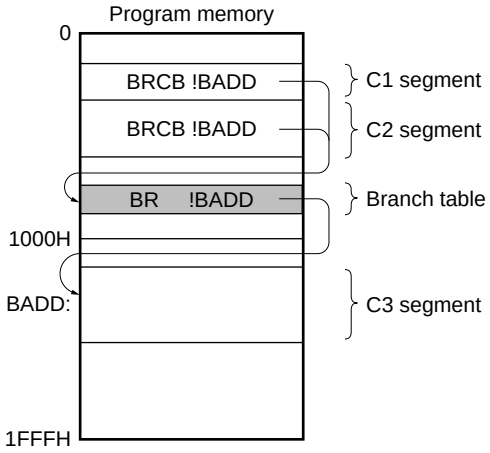
Notes 1. To be exact , the branch table is created in the block to which the address two addresses ahead of the address specified by the BRCB instruction belongs .
Therefore, if the BRCB instruction is written on a block boundary (XFFEH, XFFFH), the corresponding branch table will be created in the next block. Writing a BRCB instruction at the end of a block should therefore be avoided.

2. The kind of 3-byte branch instruction created depends on the size of on-chip ROM in the device concerned, as shown below.
ROM size up to 16 Kbytes .....BR !addr instruction.
ROM size exceeding to 16 Kbytes .....BRA !addr1 instruction

Example 1. The branch table is created in the case shown below.



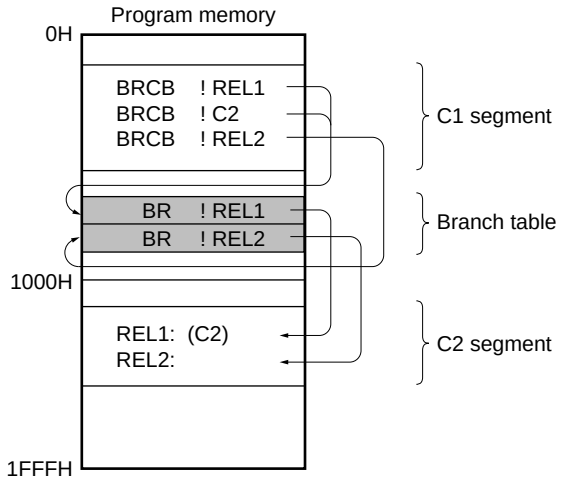
When the segments shown above are located as shown below, a branch table is created.



Example 2. If there are 2-byte branch instructions which reference differently named symbols which have the same value in the same block, one common branch table is created in the same block.

```
C1          CSEG
              ⋮
            BRCB    !REL1   ←──┐  These symbols have
              ⋮                 │  different names but a
            BRCB    !C2     ←──┘  common address value
              ⋮
            BRCB    !REL2
              ⋮
C2          CSEG
REL1:
              ⋮
REL2:
            END
```

When the segments shown above are located as shown below, a branch table is created as shown below.



The created branch table information is shown in the branch table map list in the link list file output by the linker.

Example 3. Two modules are linked as shown below.

Module 1

```
C1      CSEG
        DS      10H

ABS1    EQU     30H
ABS2    CSEG    AT 10H
ABS3:

REL1    CSEG
REL2:   DS      0FF0H

C2      CSEG
        EXTRN   EXT1, EXT2, EXT3

        BRCB    !ABS1
        BRCB    !ABS2
        BRCB    !ABS3

        BRCB    !ABS1
        BRCB    !REL1
        BRCB    !REL2

        BRCB    !ABS1
        BRCB    !EXT1
        BRCB    !EXT2
        BRCB    !EXT3

        END
```

Module 2

```
        PUBLIC  EXT1,EXT3
EXT1    EQU     40H
EXT2    CSEG    AT 10H
EXT3:
        END
```

The following branch table map list is output to the link list file output by the linker.

SEGMENT LINK MAP FOR BR.LNK (BR1)

MAP OF ROM AREA:

BASE	LENGTH	MODULE NAME	SEGMENT NAME (TYPE)	
0000H	0010H	DOS22	C1	(INBLOCK)
0010H	0000H	DOS22	ABS2	(ABSOLUTE)
0010H	0000H	DOS23	EXT2	(ABSOLUTE)
0010H	0FF0H	DOS22	REL1	(INBLOCK)
1000H	0014H	DOS22	C2	(INBLOCK)
1014H	0009H			(Branch table)
101DH	0763H		** GAP **	

MAP OF ROM AREA:

TYPE	BASE	LENGTH	MODULE NAME	SEGMENT NAME
	0000H	0100H		** GAP **
STACK	0100H	0000H	BR1	SSEG
	0100H	0040H		** GAP **

BRANCH TABLE MAP FOR BR.LNK:

BLOCK NO	LOCATED ADDRESS		REFERENCE EXPRESSION	REFERENCE ADDRESS	REFERENCE SEGMENT
01	1014H	BR	!REL1	0010H	REL1
01	1017H	BR	!0030H	0030H	
01	101AH	BR	!EXT1	0040H	

Three branch tables have been created.

Caution

In order to perform automatic branch table creation processing, the linker records provisional branch tables uniformly for all BRCB instructions other than those for which the object code was determined during assembly (which branch from an absolute address to an absolute address). Then only those branch tables which are really necessary are created when segment location addresses are determined.

Therefore, it may happen that the number of branch tables recorded temporarily during linker processing exceeds the maximum number of approximately 1,000 even if branch tables are not actually created, with the result that the linker aborts.

If this happens, linkage should be performed again with the -NTB linker option specified (see 5.4.4 “Description of linker options” for the -NTB option).

However, if linkage is performed with the -NTB option specified, no branch tables will be created, and thus a BRCB instruction which branches to another block will cause an error.

5.3 Linker Start Method

5.3.1 Starting the linker

The linker is started by inputting the following command in the format shown in the OS command line.

```
X>LK75X [␣option…]␣input file name[␣input file name…] [␣option…]
```

- X indicates the current drive.
- “input file name” is the name of the object module file to be linked. The drive name, directory name, etc., can be added to the input file name.

  Examples LK75X B: 75XTEST1.REL C: 75XTEST2.REL
  LK75X C: \USER\NEC\75XTEST1.REL C: \USER\NEC\75XTEST2.REL

- “option” is a string of 1 to 3 characters beginning with the “-” symbol, and may be followed by parameters. Options can be written before and after the input file, and if there are multiple options, they can be written in any order. However, if multiple identical options or options of the same kind are written, in some cases an error is generated, and in some cases the last output specified is valid . See **5.4 “Linker Options”** for details.
- One or more blanks (spaces or TAB) should be used to separate options and the input file name.
- The input file name and options can be written in a parameter file. For the use of the parameter file, see the item on the **(14) ‘-F’ option in 5.4.4 “Description of linker options”**.
- As the default output destination, a file with the same name a the first file specified but with the file type changed to ‘.LNK’ is created in the current path. This can be changed by means of the ‘-O’ option.
- ‘RA75X.OM1’ is necessary to start the linker.

5.3.2 Execution start and end messages

(1) Execution start message

When the linker is started an execution start message is displayed on the console

```
75X Series Linker VX.XX [XX Xxx XX]
  Copyright (C) NEC Corporation 1985
```

(2) Execution end message

- If linkage terminates normally, the linker outputs the following message to the console and returns control to the OS.

```
LINK COMPLETE, NO ERROR FOUND
```

- If linkage errors are detected during linkage, the linker displays an error message on the console and returns control to the OS.

```
B>LK75X -FLINK.PLK
75X Series Linker VX. XX [XX Xxx XX]
  Copyright (C) NEC Corporation 1985

*** W300 CHIP TYPE MISMATCH(MODULE: AD_SUB)

LINK COMPLETE, 1 ERROR FOUND
```

- If a fatal error is detected during linkage which prevents linkage from continuing, the linker outputs a message to the console, stops execution, and returns control to the OS.
  An example of an error message is shown on the next page.

Example When a source module file is specified as an input file.

```
B>LK75X 75XTEST1.REL 75XTEST2.ASM
75X Series Linker VX. XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1985

*** ERROR F304 INVALID FILE SYNTAX(FILE= 75XTEST2.ASM)
Program aborted
```

In this example, an error is generated since a source module file is specified as an input file, and linkage is aborted.

When the linker outputs an error message and aborts linkage, the cause of the error message should be found in **13.2 “Linker’s Error Messages”**, and appropriate action taken.

5.3.3 Linker error handling

If the linker detects an error during execution, it performs one of the following three kinds of processing according to the severity of the error.

(1) Abort error

If an error is generated which prevents program execution from continuing, the program displays a ‘Program aborted’ message and the program is aborted immediately .

(2) Fatal error

If an error is generated which would result in generation of object code different from that intended by the user, the program nevertheless continues processing to the end, then outputs the message “LINK COMPLETE, X ERRORS FOUND” (where X is the number of errors).

(3) Normal termination

If the program terminates normally, it outputs the message “ LINK COMPLETE, NO ERROR FOUND”.

In the (1) and (2) cases, the error message is output in the following format (the destinations are standard output and the map file).

```
***␣ERROR␣error number␣error message
```

5.3.4 Linker termination status

When the linker terminates and returns control to the OS, one of the following error status codes is returned to the OS.

Termination Condition	Termination Status
Normal termination	0
Fatal error	1
Abort error	2

When the linker is started from a batch file under MS-DOS (PC DOS, IBM DOS), it is possible to determine whether there are any linkage errors automatically using these values.

5.4 Linker Options

5.4.1 Types of linker options

Linker options are used to five the linker detailed directions concerning its operation. There are 15 different options as shown below.

Table 5-3 Kinds of Linker Options

No.	Description Format	Function/Category	Default Interpretation
1	-M[module name]	Output module name specification	Object module name of first file input
2	-P[file name] -NP	Link list file specification	First input 'input file .MAP' is output to current path
3	-KM -NKM	Map list output specification	-KM
4	-KP -NKP	Public symbol list output specification	-KP
5	-KL -NKL	Local symbol list output specification	-KL
6	-CD([segment name ['address] [, ...])	Code segment relocation address location order specification (multiple specifications possible)	Automatically located by linker
7	-RS (start address, end address [. ..., ...])	Code segment allocation prohibited area specification (multiple specifications possible)	ROM area not incorporated in target device
8	-SQ -RN	Segment location order specification	-RN
9	-SK address	Sets stack address in assembler reserved word 'STACK'.	Set automatically by linker.
10	-SZ[+]size	Stack size change specification	None
11	-NTB	Specifies suppression of automatic branch table creation	Created automatically
12	-O[file name] -NO	Load module file specification	First input ' input file .LNK' is output to current path
13	-J -NJ	Load module file forced output specification	-NJ
14	-F file name	Parameter file specification	All options and file names are read from command line
★ 15	-Y path name	Specifies the device file search path.	For the LK75X run path, the path specified in '..\DEV' path, LK75X run path, current directory and environmental variable 'PATH' are searched in that order.

Remark Options can be written in either upper- or lower-case characters.

Caution

- **Link list definition**

  There are 6 kinds of link lists.

  Link lists
  - **Linker option list**
  - **Input/output module list**
  - **Segment link map list**
  - **Branch table map list**
  - **Public symbol list**
  - **Symbol list**

5.4.2 Linker option specification method

Linker options are specified in the command line when the linker is started or in a parameter file. See **5.3 "Linker Start Method"** for the method of specifying linker options in the command line and in a parameter file.

5.4.3 Linker option priority order

(1) If multiple identical options or options of the same kind are specified in the command line, the option specified last is valid.
(2) If the same or same kind of option is specified in the parameter file and in the command line, the command line option is valid.
(3) With the -CD and -RS options, all the specified options are valid. However, an error will be generated if multiple different specifications are made for the same segment with the -CD option,.

5.4.4 Description of linker options

Each of the linker options is described in detail in the following pages.

-M name

(1) -M

Description Format	-M load module name
Default Interpretation	-M first input file object module name

[Function]

- The -M option specifies the output load module name.

[Use]

- The -M option specifies the output load module name.

[Description]

- The load module name should be specified as a string of up to 8 characters.
- The load module name is printed in the link list.

[Example]

- ‘AAA’ is specified as the output load module name.

```
A:\NEC\TOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -MAAA
```

-M name

→The link list is printed out as follows.

```
75X SERIES LINKER VX.XX                                    XX/XX/XX XX:XX:XX  PAGE :  X

COMMAND          : 75XTEST1.REL 75XTEST2.RBL -075XTEST.LNK -MAAA

INPUT MODULE LIST :
 75XTEST1.REL        (AD_MAIN)
 75XTEST2.REL        (AD_SUB)

LOAD MODULE LIST :
 75XTEST.LNK         (AAA)

SEGMENT LINK MAP FOR 75XTEST.LNK (AAA)

MAP OF ROM AREA :
```

(2) -P/-NP

Description Format	-P[output file name] -NP
Default Interpretation	The object module 'file name.MAP' specified initially is output to the current path.

[Function]

- The -P option specifies the output destination and file name of the link list file output by the linker.
- The -NP option specifies that no link list file is to be created.

[Use]

- The -P option is specified when it is wished to change the link list file output destination or file name.
- The -NP option is specified when linkage is to be performed only in order to output a link module file, etc. (the linkage time is reduced).

[Description]

- In addition to a file name, the following can be specified as the file output destination:
  - -PPRN Link list is output to line printer.
  - -PCON Link list is output to console.
  - -PAUX Link list is output to RS-232-C.
- If the drive name is omitted from the output name specification, the current path is used. If the extension is omitted, 'MAP' is used.

[Examples]

Example 1. When the -P option is specified. The file name is **"SAMPLE.MAP"**.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -PSAMPLE.MAP
```

→The link list file **"SAMPLE.MAP"** and load module file **"75XTEXT.LNK"** are output.

Example 2. To output the link list **"75XTEST.LNK"** to the printer

```
A:\NECTOOLS\SMP75X\RA75X>LX75X 75XTESTl.REL 75XTEST2.REL -075XTEST.LNK -PPRN
```

Example 3. The -NP option is specified.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -NP
```

→The link list file is not output.
The Load Module File **"75XTEST.LNK"** only is output.

(3) -KM/-NKM

Description Format	-KM
	-NKM
Default Interpretation	-KM

[Function]

- The -KM option specifies that map lists (segment map list and branch table map list) are to be output to the link list file.
- The -NKM option specifies that map lists are not to be output to the link list file.

[Use]

- The -NKM option is specified when linkage is to be performed only in order to output a load module file, etc. (the linkage time is reduced).

[Description]

- The link list file output destination is specified by the -P option.
- If the -NP is specified the -KM option is invalid and map lists are not output.

[Examples]

Example 1. When the -NKM option is specified

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -NKM
```

→The link list is as shown below.

```
75X SERIES LINKER VX.XX                                  XX/XX/XX XX:XX:XX  PAGE :  X

COMMAND          : 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -NKM        Linker Option List


INPUT MODULE LIST:
 75XTEST1.REL        (AD_MAIN)
 75XTEST2.REL        (AD_SUB)                                          Input/Output Module List

LOAD MODULE LIST:
 75XTEST.LNK         (AD_MAIN)


PUBLIC SYMBOL LIST FOR 75XTEST.LNK

TYPE      VALUE    MODULE     SYMBOL NAME                             Public Symbol List
--------  -------- ---------- ----------------------
CODE      0022H    AD_SUB     ADCONV
```

Example 2. When the -KM option is specified (the same result is produced if the -KM option is omitted)

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1. REL 75XTEST2.REL -075XTEST.LNK -KM
```

-KM/-NKM map/no map

→The link list is as shown below.

```
75X SERIES LINKER VX.XX                                  XX/XX/XX XX:XX:XX  PAGE :  X

COMMAND         : 75XTEST1.REL 75XTEST2.REL -075XTEST.LNX -KM        Linker Option List


INPUT MODULE LIST:
 75XTEST1.REL       (AD_MAIN)
 75XTEST2.REL       (AD_SUB)                                  Input/Output Module List

LOAD MODULE LIST:
 75XTEST.LNK        (AD_MAIN)

SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA :

                  BASE    LENGTH  MODULE NAME          SEGMENT NAME (TYPE)
                  ------  ------  -------------------  -------------------------------------------
                  0000H   0002H   AD_MAIN                                (ABSOLUTE)
                  0002H   0006H                        ** GAP **
                  0008H   0002H   AD_MAIN                                (ABSOLUTE)
                  000AH   0012H   AD_SUB               SEG4             (SENT)
                  001CH   0004H                        ** GAP **
                  0020H   0002H   AD_MAIN              SEG1             (IENT)
                  0022H   0024H   AD_SUB               SEG5             (SENT)
                  0046H   000AH   AD_MAIN              SEG3             (SENT)
                  0050H   0039H   AD_MAIN              SEG2             (INBLOCK)
                  0089H   16F7H                        ** GAP **

MAP OF RAM AREA :
                  TYPE            BASE   LENGTH  MODULE NAME          SEGMENT NAME
                  -------         -----  ------  -------------------  ------------------------
                                  0000H  00F4H                        ** GAP **
                  STACK           00F4H  000CH   AD_MAIN              SSEG
                                  0100H  0010H                        ** GAP **
                  DATA            0110H  0002H   AD_MAIN              SEG0
                                  0112H  002EH                        ** GAP **

                                                                      Segment Map List

PUBLIC SYMBOL LIST POR 75XTEST.LNK
```

Caution

In this example no branch table is created and therefore a branch table map list is not output even though the -KM option is specified.

(4) -KP/-NKP

Description Format	-KP
	-NKP
Default Interpretation	-KP

[Function]

- The -KP option specifies that a public symbol list is to be output to the link list file.
- The -NKP option specifies that a public symbol list is not to be output.

[Use]

- The -KP option is specified when it is wished to ascertain information such as the symbol names and values of symbols defined by a PUBLIC pseudo-instruction in an input segment, the names of defined modules, etc.
- If the -NKP option is specified, the linkage processing time is shortened somewhat.

[Description]

- The public symbol list output destination is specified by the -P option.
- If the -NP is specified the -KP option is invalid and a public symbol list is not output.

[Examples]

Example 1. When the -KP option is specified

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -KP
```

-KP/-NKP publics/no publics

→The public symbol list is output.

```
75X SERIES LINKER VX.XX                                    XX/XX/XX XX:XX:XX  PAGE :  X

COMMAND          : 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -KP
                                                    ⋮
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
              BASE     LENGTH    MODULE NAME      SEGMENT NAME (TYPE)
              ---------  ----------  ----------------------  ----------------------------------------------
              0000H    0002H     AD_MAIN                        (ABSOLUTE)
              0002H    0006H                      ** GAP **
                                                    ⋮
MAP OF RAM AREA:
              TYPE               BASE    LENGTH   MODULE NAME   SEGMENT NAME
              -------            -------  ----------  ----------------------  ----------------------
                                 0000H    00F4H                  ** GAP **
              STACK              00F4H    000CH    AD_MAIN       SSEG
                                                    ⋮
PUBLIC SYMBOL LIST POR 75XTEST.LNK

TYPE     VALUE    MODULE     SYMBOL NAME
-------  ---------  ----------  ----------------------
CODE     0022H    AD_SUB     ADCONV
DATA     0110H    AD_MAIN    SEG0
CODE     0020H    AD_MAIN    SEG1
CODE     0050H    AD_MAIN    SEG2
CODE     0046H    AD_MAIN    SBG3
CODE     000AH    AD_SUB     SEG4
CODE     0022H    AD_SUB     SEG5
CODE     0020H    AD_MAIN    SEL15
CODE     000AH    AD_SUB     SIOSUB
DATA     0100H               STACK
DATA     0110H    AD_MAIN    TDATA
                                                    ⋮
SYMBOL LIST FOR 75XTEST.LNK

TYPE     VALUE    ATTRIBUTE NAME
-------  ---------  ----------------  --------
--------  ---------  MODULE    AD_MAIN
CODE     0046H    SYMBOL    HEIKIN
```

Example 2. When the -NKP output is specified

```
A:\NECTOOLS\SMP75\RA75X>LX75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -NKP
```

→The public symbol list is output.

```
75X SERIES LINKER VX.XX                                    XX/XX/XX XX:XX:XX  PAGE :  X

COMMAND          : 75XTESTl.REL 75XTEST2.REL -075XTEST.LNK -NKP
                                                    ⋮
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
              BASE     LENGTH    MODULE NAME      SEGMENT NAME (TYPE)
              ---------  ----------  ----------------------  ----------------------------------------------
              0000H    0002H     AD_MAIN                        (ABSOLUTE)
              0002H    0006H                      ** GAP **
                                                    ⋮
MAP OF RAM AREA:
              TYPE               BASE    LENGTH   MODULE NAME   SEGMENT NAME
              -------            --------  ----------  ----------------------  -------------------------
                                 0000H   00F4H                  ** GAP **
              STACK              00F4H   000CH    AD_MAIN       SSEG
                                 0100H   0010H                  ** GAP **
                                                    ⋮
SYMBOL LIST FOR 75XTEST.LNK

TYPE     VALUE    ATTRIBUTE NAME
-------  ---------  ---------------- ---------
-------  ---------  MODULE     AD_MAIN
CODE     0046H    SYMBOL     HEIKIN
PBIT     0FBCH.1  SYMBOL     IET0
CODE     0060H    SYMBOL     LOOP1
CODE     0066H    SYMBOL     LOOP2
```

(5) -KL/-NKL

Description Format	-KL
	-NKL
Default Interpretation	-KL

[Function]

- The -KL option specifies that a local symbol list is to be output to the link list file.
- The -NKL option specifies that a local symbol list is not to be output.

[Use]

- The -KL option is specified when it is wished to ascertain information such as the symbol attribute, value and type of all the symbols defined in an input segment.
- If the -NKL option is specified, the linkage time is shortened somewhat.

[Description]

- The symbol list output destination is specified by the -P option.
- If the -NP is specified the -KL option is invalid and therefore a symbol list is not output.

[Examples]

Example 1. When the -KL option is specified

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -KL
```

-KL/-NKL local symbols/no local symbols

→The local symbol list is output.

```
75X SERIES LINKER VX.XX                                    XX/XX/XX XX:XX:XX  PAGE :  X

COMMAND          : 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -KL
                                                    ⋮
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
              BASE     LENGTH    MODULE NAME     SEGMENT NAME (TYPE)
              ---------  ----------  ----------------------  ----------------------------------------------
              0000H    0002H     AD_MAIN                     (ABSOLUTE)
              0002H    0006H                     ** GAP **
                                                    ⋮
MAP OF RAM AREA:
              TYPE               BASE    LENGTH   MODULE NAME   SEGMENT NAME
              -------            -------  ----------  ----------------------  ------------------------
                                 0000H    00F4H                     ** GAP **
              STACK              00F4H    000CH    AD_MAIN          SSEG
                                                    ⋮
PUBLIC SYMBOL LIST POR 75XTEST.LNX

TYPE     VALUE    MODULE     SYMBOL NAME
-------  ---------  ----------  ----------------------
CODE     0022H    AD_SUB     ADCONV
DATA     0110H    AD_MAIN    SEG0
CODE     0020H    AD_MAIN    SEG1

SYMBOL LIST FOR 75XTEST.LNK
                                                    ⋮
TYPE     VALUE    ATTRIBUTE NAME
-------  ---------  ----------------- --------
--------  ---------  MODULE     AD_MAIN
CODE     0046H    SYMBOL     HEIKIN
PBIT     0FBCH.1  SYMBOL     IET0
CODE     0060H    SYMBOL     LOOP1
CODE     0066H    SYMBOL     LOOP2
CODE     0079H    SYMBOL     LOOP3
CODE     007DH    SYMBOL     LOOP4
CODE     0048H    SYMBOL     LOOP5
CODE     0050H    SYMBOL     MAIN
PBIT     0FB0H.1  SYMBOL     MBE
DATA     0FB3H    SYMBOL     PCC
```

Example 2. When the -NKL option is specified

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1. REL 75XTEST2. REL -075XTEST. LNX -NKL
```

→The local symbol list is output.

```
75X SERIES LINKER VX.XX                                    XX/XX/XX XX:XX:XX  PAGE :  X

COMMAND         : 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -KL
                                                           ⋮
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
              BASE    LENGTH    MODULE NAME     SEGMENT NAME (TYPE)
              ---------  ----------  ----------------------  ----------------------------------------------
              0000H   0002H     AD_MAIN                     (ABSOLUTE)
              0002H   0006H                     ** GAP **
                                                           ⋮
MAP OF RAM AREA:
              TYPE              BASE    LENGTH   MODULE NAME   SEGMENT NAME
              -------           --------  ------------  ----------------------  -------------------------
                                0000H   00F4H                  ** GAP **
              STACK             00F4H   000CH    AD_MAIN       SSEG
                                                           ⋮
PUBLIC SYMBOL LIST FOR 75XTEST.LNK

TYPE     VALUE    MODULE     SYMBOL NAME
--------  ---------  ------------  ----------------------
CODE     0022H    AD_SUB     ADCONV
DATA     0110H    AD_MAIN    SEG0
CODE     0020H    AD_MAIN    SEG1
                                                           ⋮
LINK COMPLETE, NO ERROR FOUND
```

(6) -CD

Description Format	-CD (segment name [‘ address] [. ...])
Default Interpretation	Relocatable code segment is located automatically by linker

[Function]

- The -CD option specifies the location address of a relocatable code segment, or specifies the location order for multiple code segments.

[Use]

- The -CD option is specified when it is wished to specify a location address when linkage is performed on segments defined as relocatable segments in the assembly stage.
- It is specified when it is wished to determine in relative terms the location order of multiple relocatable segments.

[Description]

- The segment name specified is a segment name defined by a CSEG pseudo-instruction in a source module.
- The segment location address can be specified in binary, octal, decimal or hexadecimal notation. The address range is within the ROM area of the target device.
- An error will result if multiple specifications are made for the same segment.

Caution

Address specification and order specifications can be made simultaneously with the -CD option.

Example -CD (C1, C4, C3’300H, C6, C5)

When this kind of specification is made, order specifications are interpreted individually before segments for which an address specification is made. In other words, this case is equivalent to the following specifications.

-CD (C1, C4) -CD (C3’300H, C6, C5)

-CD code

[Examples]

Example 1. When the -CD option is omitted

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK
```

```
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
        BASE    LENGTH  MODULE NAME     SEGMENT NAME (TYPE)
        ------  ------  --------------  ------------------------------------------
        0000H   0002H   AD_MAIN                      (ABSOLUTE)
        0002H   0006H                   ** GAP **
        0008H   0002H   AD_MAIN                      (ABSOLUTE)
        000AH   0012H   AD_SUB          SEG4         (SENT)
        001CH   0004H                   ** GAP **
        0020H   0002H   AD_MAIN         SEG1         (IENT)
        0022H   0024H   AD_SUB          SEG5         (SENT)
        0046H   000AH   AD_MAIN         SEG3         (SENT)
        0050H   0039H   AD_MAIN         SEG2         (INBLOCK)
        0089H   16F7H                   ** GAP **
```

-CD code

Example 2. When the -CD option is specified for segments SEG2 and SEG3

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -CD(SEG2'100H, SEG3'200H
```

→The segments are located at the addresses specified by the -CD option.

```
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
        BASE    LENGTH  MODULE NAME         SEGMENT NAME (TYPE)
        ------  ------  ------------------  ----------------------------------------
        0000H   0002H   AD_MAIN                           (ABSOLUTE)
        0002H   0006H                       ** GAP **
        0008H   0002H   AD_MAIN                           (ABSOLUTE)
        000AH   0012H   AD_SUB              SEG4          (SENT)
        001CH   0004H                       ** GAP **
        0020H   0002H   AD_MAIN             SEG1          (IENT)
        0022H   0024H   AD_SUB              SEG5          (SENT)
        0046H   00BAH                       ** GAP **
        0100H   0039H   AD_MAIN             SEG2          (INBLOCK)
        0139H   00C7H                       ** GAP **
        0200H   000AH   AD_MAIN             SEG3          (SENT)
        020AH   1576H                       ** GAP **
```

(7) -RS

Description Format	-RS (start address, end address [, …, …])
Default Interpretation	-RS (ROM area not incorporated in target device)

[Function]

- The -RS option specifies the area of program memory (ROM) in which code segment location is prohibited, using the start and end addresses.

[Use]

- The -RS option is specified when there is an area in the ROM area in segments are not to be located.

[Description]

- The linker does not locate code segments in the area specified by the -RS option.
- The start address and end address can be specified in binary, octal, decimal or hexadecimal notation.
- The following condition must apply: start address - end address

Caution

When numeric specification is omitted by -RS option. It is assumed that OH has been specified for the omitted part. Therefore, a numeric specification should not be omitted during -RS option specification.

Example

```
A>LK75X TEST -RS (, 100H, , 200H)
```

When described as above, address 0H to 200H become location disabled area.

[Examples]

Example 1. When the -RS option is omitted.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK
```

```
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
        BASE    LENGTH  MODULE NAME     SEGMENT NAME (TYPE)
        ------  ------  --------------  ------------------------------------------
        0000H   0002H   AD_MAIN                      (ABSOLUTE)
        0002H   0006H                   ** GAP **
        0008H   0002H   AD_MAIN                      (ABSOLUTE)
        000AH   0012H   AD_SUB          SEG4         (SENT)
        001CH   0004H                   ** GAP **
        0020H   0002H   AD_MAIN         SEG1         (IENT)
        0022H   0024H   AD_SUB          SEG5         (SENT)
        0046H   000AH   AD_MAIN         SEG3         (SENT)
        0050H   0039H   AD_MAIN         SEG2         (INBLOCK)
        0089H   16F7H                   ** GAP **
```

Example 2. When the -RS option is specified

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -RS(40H, 60H)
```

```
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
        BASE    LENGTH  MODULE NAME     SEGMENT NAME (TYPE)
        ------  ------  --------------  ------------------------------------------
        0000H   0002H   AD_MAIN                      (ABSOLUTE)
        0002H   0006H                   ** GAP **
        0008H   0002H   AD_MAIN                      (ABSOLUTE)
        000AH   0012H   AD_SUB          SEG4         (SENT)
        001CH   0004H                   ** GAP **
        0020H   0002H   AD_MAIN         SEG1         (IENT)
        0022H   000AH   AD_MAIN         SEG3         (SENT)
        002CH   0014H                   ** GAP **
        0040H   0021H                                (Reserve area)
        0061H   0024H   AD_SUB          SEG5         (SENT)
        0085H   0039H   AD_MAIN         SEG2         (INBLOCK)
        00BEH   16C2H                   ** GAP **
```

(8) -SQ/-RN

Description Format	-SQ
	-RN
Default Interpretation	-RN

[Function]

- The -SQ option specifies that segments are to be located in the order in which they are written in the input modules or in each input module.
- The -RN segment specifies that segments are to be located in memory in an efficient fashion without regard to the module input order.

[Use]

- The -RN option is specified when it is wished to avoid creating free areas as far as possible, without regard to the segment input order.
- When desiring to locate segments in the description order in the source module and in the order specified in the input module, specify the -SQ option.

[Description]

- The -SQ and -RN options are used to indicate to the linker the method to be used for determining the location addresses of relocatable segments.
  See **5.2.2. (2) "Determination of segment location addresses"** for details of the segment location method when each option is specified.

Caution

However, if there are a large number of segments, and object code is generated to the point of filling the ROM capacity of the product on which assembly is being performed, the processing time will be extremely long. Since there is danger that the user will mistake it for running out of control, if a correct location method is not discovered after a fixed number of relocation attempts (3000 times), an error message is output and the operation is aborted.
There are three possible countermeasures in this case, as follows:

1. **Link the object modules which could not be linked after specifying the -SQ option (anerror will of course be generated as a result), decide the location method manually based on the size of the segments in the output map list, then perform linkage again with the location order specified by the -CD option.**
2. **Reduce the number of segments.**
3. **Amend the program so that INBLOCK (INBLOCKA) segments fit exactly within block boundaries.**

-SQ/-RN sequential/random

[Examples]

Example 1. When the -RN option is specified (the result is the same if the -RN option is omitted)

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -RN
```

→Segments are located as follows.

```
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
        BASE    LENGTH  MODULE NAME     SEGMENT NAME (TYPE)
        ------  ------  --------------  ------------------------------------------
        0000H   0002H   AD_MAIN                       (ABSOLUTE)
        0002H   0006H                   ** GAP **
        0008H   0002H   AD_MAIN                       (ABSOLUTE)
        000AH   0012H   AD_SUB          SEG4          (SENT)
        001CH   0004H                   ** GAP **
        0020H   0002H   AD_MAIN         SEG1          (IENT)
        0022H   0024H   AD_SUB          SEG5          (SENT)
        0046H   000AH   AD_MAIN         SEG3          (SENT)
        0050H   0039H   AD_MAIN         SEG2          (INBLOCK)
        0089H   16F7H                   ** GAP **
```

-SQ/-RN sequential/random

Example 2. When the -SQ option is specified

```
A:\NECTOOLS\SMP75X>LX75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -SQ
```

→Segments in 75XTEST1.ASM and 75XTEST2.ASM are located in the order of segment description and when there are links, in the input module order.

```
SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA:
        BASE    LENGTH  MODULE NAME     SEGMENT NAME (TYPE)
        ---------  ----------  ----------------------  ----------------------------------------------
        0000H   0002H   AD_MAIN                       (ABSOLUTE)
        0002H   0006H                   ** GAP **
        0008H   0002H   AD_MAIN                       (ABSOLUTE)
        000AH   0016H                   ** GAP **
        0020H   0002H   AD_MAIN         SEG1          (IENT)
        0022H   0039H   AD_MAIN         SEG2          (INBLOCK)
        005BH   000AH   AD_MAIN         SEG3          (SENT)
        0065H   0012H   AD_SUB          SEG4          (SENT)
        0077H   0024H   AD_SUB          SEG5          (SENT)
        009BH   16E5H                   ** GAP **
```

(9) -SK

Description Format	-SK address
Default Interpretation	Value is assigned to reserved word STACK by linker

[Function]

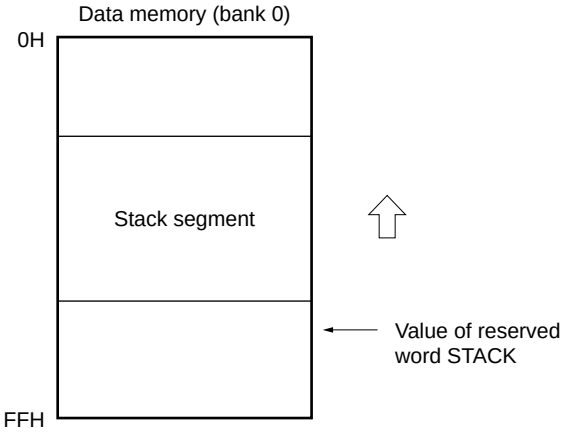
- The -SK option sets a value in the reserved word STACK.
  The linker locates the stack segment in the low address direction starting at the address (address specified by -SK option -1).

[Use]

- The -SK option is specified when it is wished to specify the stack segment location address (it is only meaningful when the stack pointer value has been specified using the reserved word 'STACK' in the source program).
  See **4.5 (3) STKLN for details**.

[Description]

- The reserved word STACK has a value which indicates the address (maximum address of stack segment + 1).
- When the stack segment is located automatically in the linkage processing, the reserved word 'STACK' is assigned the address value (maximum address of stack segment +1) by the linker.
- If the value of STACK is specified by the -SK option when linkage is performed, the stack segment is located in the low address direction starting at the address (address specified by -SK option -1).
  (See **5.2.2 (3) "Determination of stack segment location address"** for details of stack segment location.)



-SK- stack

- The specified address must be an even address in the range 0H to 100H, and can be specified in binary, octal, decimal or hexadecimal notation.
- The -SK option can only be specified once.
- The stack segment can only be located in bank 0.

[Examples]

Example 1. When the -SK option is omitted

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK
```

→The stack segment is located automatically.

```
MAP OF ROM AREA:
        TYPE        BASE        LENGTH      MODULE NAME         SEGMENT NAME
        ---------   ----------  ----------  ------------------  ------------------------
                    0000H       00F4H                           ** GAP **
        STACK       00F4H       000CH       AD_MAIN             SSEG
                    0100H       0010H                           ** GAP **
        DATA        0110H       0002H       AD_MAIN             SEG0
                    0112H       002EH                           ** GAP **
```

Example 2. When the -SK option is specified

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -SK0A0H
```

→The stack segment is located starting as address (address specified by -SK option - 1).

```
MAP OF ROM AREA:
        TYPE            BASE        LENGTH      MODULE NAME          SEGMENT NAME
        ---------       ----------  ----------  -------------------- ------------------------
                        0000H       0094H                            ** GAP **
        STACK           0094H       000CH       AD_MAIN              SSEG
                        00A0H       0070H                            ** GAP **
        DATA            0110H       0002H       AD_MAIN              SEG0
                        0112H       002EH                            ** GAP **
```

(10) -SZ

Description Format	-SZ [±] size
Default Interpretation	No change of stack area is assumed

[Function]

- The -SZ option specifies the size of the stack segment to be changed to in nibbles.

[Use]

- The -SZ option is specified when it is wished to change the size of the stack segment linked when linkage is performed (it is only meaningful when the stack pointer value has been set using the reserved word ‘STACK’ in the source program, and the stack size has been reserved with the STKLN pseudo-instruction).

[Description]

- The stack segment size is increased or decreased by the amount specified by the -SZ option.
- The size can be specified in binary, octal, decimal or hexadecimal notation.
- If multiple -SZ options are specified, the last one specified is valid.

[Example]

Example The -SZ option is specified as shown below.

```
A:\NECTOOLS\SMP75X\RA75X>75X 75XTEST1.REL 75XTEST2.REL -075XTEST.LNK -SZ+1CH
```

→Since the size of the original stack segment is 0CH, the stack segment size is 0CH + 1CH = 28H.

```
MAP OF ROM AREA:
        TYPE        BASE        LENGTH      MODULE NAME         SEGMENT NAME
        ---------   ----------  ----------  ------------------  ------------------------
                    0000H       00D8H                           ** GAP **
        STACK       00D8H       0028H       AD_MAIN             SSEG
                    0100H       0010H                           ** GAP **
        DATA        0110H       0002H       AD_MAIN             SEG0
                    0112H       002EH                           ** GAP **
```

(11) -NTB

Description Format	-NTB
Default Interpretation	Linker creates a branch tables automatically

[Function]

- The -NTB option notifies the linker that automatic creation of branch tables is not to be performed.

[Use]

- If linkage is aborted with an “F012 BRANCH TABLE OVERFLOW” error, linkage should be performed again with the -NTB option specified.

[Description]

- In order to perform automatic branch table creation processing, the linker records provisional branch table uniformly for all BRCB instructions other than those for which the object code was determined during assembly (which branch from an absolute address to an absolute address). Then only those branch tables which are really necessary are created when segment location addresses are determined.
  Therefore, it may happen that the number of branch tables recorded temporarily during linker processing exceeds the maximum number of approximately 1,000, even if branch tables are not actually created, with the result that the linker aborts.
- If the -NTB option is specified the linker does not create branch tables automatically, and therefore linkage will not be aborted with an “F012 BRANCH TABLE OVERFLOW” error.
- Also, as the number of branch tables approaches the maximum of approximately 1,000, the generated object efficiency becomes extremely low.
- If linkage is performed with the -NTB option specified, an error will result if the branch destination of a BRCB instruction is not in the same block.
  In this case, the relevant BRCB instructions should be rewritten as 3-byte branch instructions in the source module.

- NTB no table

[Example]

Example Assume that the following error was output when linkage was performed.

```
A:\NECTOOLS\SMP75X\RA75X>LX75X BR1.REL BR2.REL -OBR.LNK
75X Series Linker VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1985

*** ERROR F012 BRANCH TABLE OVERFLOW
Program aborted
```

→Linkage is performed with the -NTB option specified.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X BR1.REL BR2.REL -OBR.LNK -NTB
75X Series Linker VX. XX [XX Xxx XX]
   Copyright (C) NBC Corporation 1985

*** W502 EVALUATED VALUE EXCEEDS THE RANGE (AT XXXXH IN XXXX)

LINK COMPLETE, 1 ERROR FOUND
```

→Link processing terminates, but the **"W502"** error is output for commands branching to other blocks in the BRCB command.

(12) -O/-NO

Description Format	-O [load module file name] -NO
Default Interpretation	'First specified object file name.LNK' is created in current path.

[Function]

- The -O option specifies the name of the load module file to be created by the linker.
- The -NO option informs the linker that a load module file is not to be created.

[Use]

- The -O option is specified when it is wished to change the load module file name from the default name.
- The -NO option is specified when the linker is to be started only in order to output a link list file, etc.

[Description]

- If neither the -O option nor the -NO option is specified, the linker creates a load module file in the current path using the first object module file name specified, but with the file type changed to '.LNK'.
  This fives the same result as when the -O option is specified with the file name omitted.
- 'NUL' or 'AUX' can be specified for the file name as a logical device name. If 'NUL' is specified, the same result is obtained as when the -NO option is specified.
- It is possible to specify the path in which the load module file is to be created by including the path name in the file name. In this case, a file which has the name of the first object module file specified, but with the file type changed to '.LNK', is created in the specified path.
- If -O and -NO are specified at the same time, the latter specification is valid.

[Examples]

Example 1. When the linker is started with **"75XTEST.LNK"** specified as the output file name.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X -075XTEST.LNK 75XTEST1.REL 75XTEST2.REL
```

→The link list file **"75XTEST1.MAP"** and load module file **"75XTEXT.LNK"** are output.

Example 2. When the linker is started with the -NO option specified

```
A:\NECTOOLS\SMP75X\RA75X>LK75X -NO -P75XTEST.MAP 75XTEST1.REL 75XTEST2.REL
```

→In this case, the link list file **"75XTEXT.MAP"** only is output.

(13) -J/-NJ

Description Format	-J
	-NJ
Default Interpretation	-NJ

[Function]

- The -J option specifies that a load module file is to be created even if there is a linkage error.
- The -NJ option specifies that a load module file is not to be created if there is a linkage error.

[Use]

- When desiring to generate a load module file even if there are errors during linking, specify the -J option.

★ **[Explanation]**

- When the -NO option is specified, the -J option is invalid.

[Examples]

- When the -J option is specified.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1.REL -J
75X Series Linker VXX.XX [XX Xxx xx]
   Copyright (C) NEC Corporation 1985

*** ERROR W310 UNRESOLVED SYMBOL(MODULE: AD_MAIN SYMBOL: ADCONV)
*** ERROR W310 UNRESOLVED SYMBOL(MODULE: AD_MAIN SYMBOL: SIOSUB)
*** ERROR W503 REFERENCE SYMBOL IS UNRESOLVED (AT 0008H IN)
*** ERROR W503 REFERENCE SYMBOL IS UNRESOLVED (AT 0035H IN SEG2)

LINK COMPLETE, 4 ERRORS FOUND
```

→An load module file **‘75XTEST1.LNK’** has been output even though there is an assembly error.

(14) -F

Description Format	-F parameter file name
Default Interpretation	Parameter file is not used

[Function]

- The -F option specifies that linker options and input/output file names are to be read from the file specified by the option parameter. This file is called the parameter file.

[Use]

- Writing options and input/output file names to be specified for the linker in a parameter file in advance also reduces the amount of typing required.
- Options and input file names can still be specified in the command line even if a parameter file is used. It is thus possible to write only frequently used options in the parameter file.

[Description]

- The parameter file is a text file, and can be created with an editor, etc. There are no particular restrictions on the length of the parameter file.
- The parameter file name cannot be omitted. However, if the file type is omitted, '.PLK' is taken as being specified.
- A logical device name ('CON', 'AUX', etc.) cannot be specified as the parameter file name. Use of such names will result in an error.
- The contents of the parameter file are expanded at the point at which the -F option is written in the linker start line. It is therefore possible to change the parameter file contents or add other option specifications with options written after the -F option.
- Parameter files cannot be nested. If an -F option is written in the parameter file, an error will result.
- It is not possible to use, more than one parameter at one time. If multiple -F options are specified, an error will result.
- Individual options and input file names should be separated by spaces, TABs or Line Feed characters. A parameter file description cannot be split over a number of lines.
- The ';' and '#' symbols are treated as comment marks in the parameter file. Characters from these characters to the end of the line are regarded as a comment.

-F parameter file name

[Examples]

Consider a parameter file **‘75XTEST.PLK’** with the following contents.

```
75XTEST1.REL      ; Input file name
75XTEST2.REL      ; Input file name
-075XTEST.LNK     ; Output file name
-P75XTEST.MAP     ; Link list file name
-KM               ; Obtain map list on link list file
```

Example 1. The linker is started with parameter file **‘75XTEST.PLK’** specified.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X F75XTEST.PLK
```

Example 2. The contents specified by the parameter file **’75XTEST.PLK’** are changed and added to in the command line.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X F75XTEST.PLK -PPRN -SQ
```

★ **(15) -Y**

Description Format	-Y Path name
Default Interpretation	Executes a search in accordance with the specified search sequence **((2) ~ (5) of the [Explanation])**.

[Function]

- The -Y option specifies the device file search path.

[Use]

- Specify the -Y option when searching from the specified path first.

[Explanation]

- A device file is searched for by the following sequence.
  (1) Path specified by the -Y option.
  (2) '..\DEV' path with respect to the LK75X starting path.
  (3) LK75X starting path.
  (4) Current Path
  (5) Environment Variable 'PATH'

CHAPTER 6. OBJECT CONVERTER

The object converter (OC75X) has as its input the load module file [Note 1] output by the linker, and outputs a HEX format object module file which can be input to a HEX loader [Note 2].

The object converter also outputs the symbol table file required for symbolic debugging using a debugger control program.

Notes 1. All reference address information must have been resolved.

2. PG-1500, IE-75000-R (maintenance product), IE-75001-R, EVAKIT-75X (discontinued)

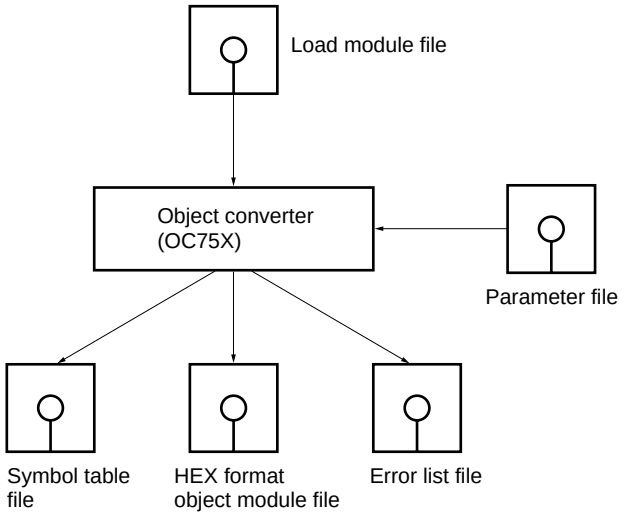
6.1 Object Converter Input/Output Files

The object converter (OC75X) input/output files are shown in Table 6-1.

Table 6-1 Object Converter Input/Output Files

	Type of File		Default File Type
★	Input file	**Load module file** Load module file output by the linker	.LNK
		Parameter file This is a file created by the editor when desiring to specify a large number of files which cannot be defined on the command line as object converter input files.	.POC
	Output file	**HEX format object module file** HEX format file which can be input to HEX loader.	.HEX
		Symbol table file File containing information on symbols included in each module of the input file	.SYM
★		**Error list file** This is a file which contains error information when running the object converter.	.EOC

★ **Figure 6-1 Object Converter Input/Output Files**



Caution An object module file (.REL) output by the assembler cannot be input.

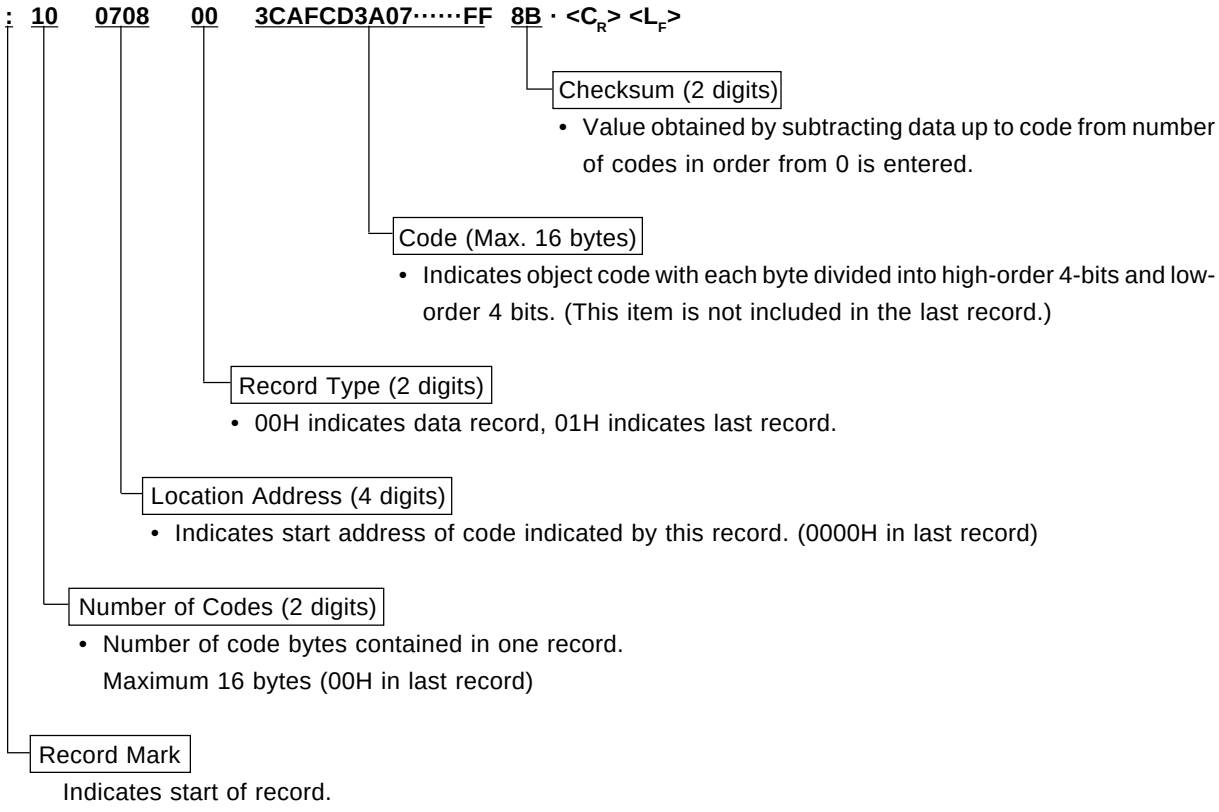
6.2 Object Converter Functions

- The object converter converts the information in a load module file to a HEX format object module which can be input by a HEX loader (PG-1500, IE-75000-R[Note 1], IE-75001-R, EVAKIT-75X[Note 2]).
- If an error is found during object conversion, an error message is displayed on the console.
- The object converter performs processing in accordance with the object converter options specified when it is started. See **6.4 "Object Converter Options"** for object converter options.
- When this processing terminates normally, the object converter outputs a termination message and returns control to the OS.
- If the -NG option is specified when assembly is performed, module symbol information is not output to the symbol table file.

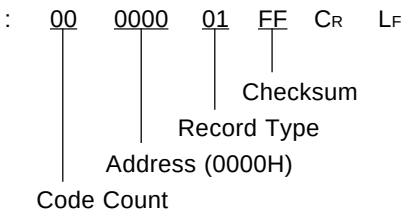
Notes 1. Maintenance product (not available for purchase)
2. Discontinued (not available for purchase)

6.2.1 HEX format object module file format

The HEX format object module file is output in HEX format.



Remark The last record indicating the end of the object code is as shown below.



Example

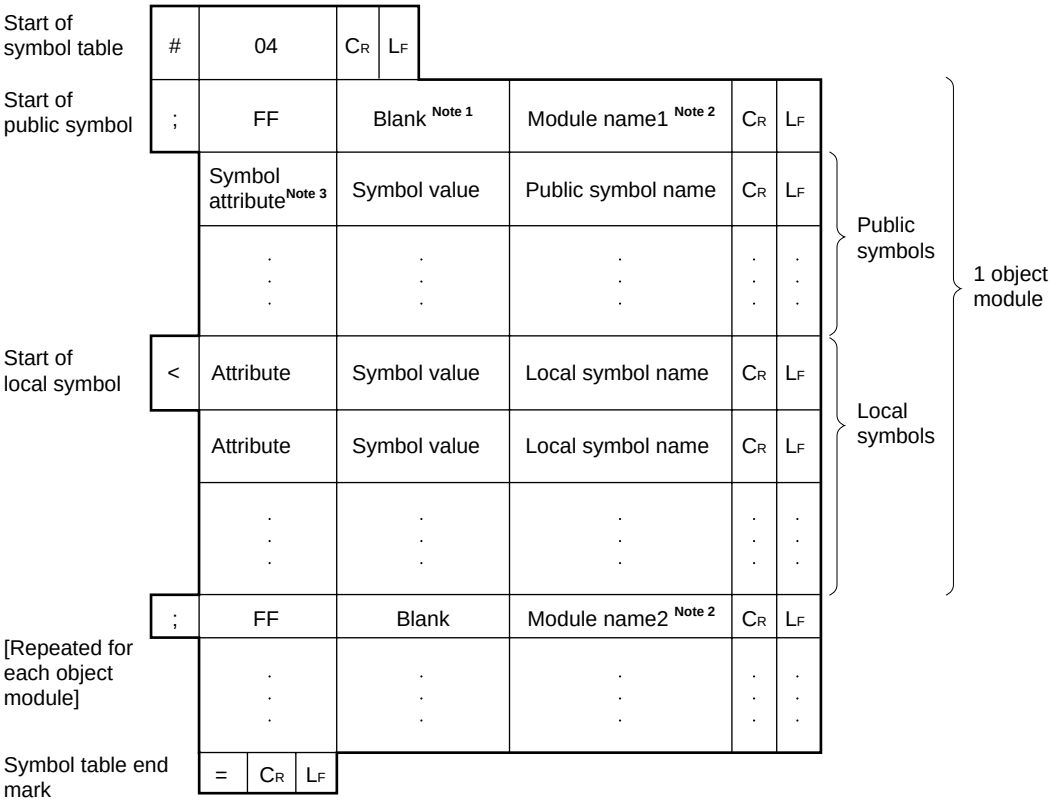
```
:02000000C050EE
:1000080080229907992299 11A2101092E489EE9200
:04001800E09906EE77
:10002000991F9907108BD389C093C09D40A3C0929C
:10003000D67AC0FEBD049B40CAF19A09A3C0AAC1EA
:100040009D809906C7EF9A2ED9E698D998CEF9EEF9
:1000500099211089009 2807393B399118B3F890085
:10006000E8AA6AFC9910E8AA6AFC10897992A68924
:100070004C92A09DB29D9C991189009A0F9A87FD80
:09008000AB40469210AB400AF0BF
:00000001FF
```

Data Records

Last Record

6.2.2 Symbol table file format

The symbol table file is output in the following format.



Notes 1. This column is fixed at 4 characters.

2. Up to 8 characters are entered in this column.

3. The symbol attributes are indicated by the following values.

Value	Symbol Attribute
00	NUMBER
01	CODE
02	DATA
03	STACK
05	BIT
08	PBIT
FF	Module name

Example

```
#04
;FF       AD_MAIN
020110SEG0
010020SEG1
010050SEG2
010046SEG3
010020SEL15
020110TDATA
<010046HEIKIN
083EF1IET0
010060LOOP1
010066LOOP2
010079LOOP3
01007DLOOP4
010048LOOP5
010050MAIN
083EC1MBE
020FB3PCC
083ECORBE
020F80SP
020FA0TM0
020FA6TMOD0
;FF       AD_SUB
010022ADCONV
01000ASEG4
010022SEG5
01000ASI0SUB
<020FCOBSB0
01002BLOOP
020FD4PTH0
020FD6PTHM
083ECORBE
020FE4SI0
020FE0SI0M
010032WAIT
=
```

6.3 Object Converter Initiation Method

6.3.1 Starting the object converter

The object converter is started by inputting the following command in the format shown in the OS command line.

```
X>OC75X [␣option,...] ␣input file name [␣option...]
```

- X indicates the current drive .
- “input file name” is the name of the load module file to be converted. The drive name, directory name, etc., can be added to the input file name.

Example

```
OC75X  B:75XTEST.LNK
OC75X  C:\USER\NEC\75XTEST.LNK
```

- “option” is a string of 1 to 2 letters beginning with the “-” symbol, and may be followed by parameters. Options can be written before and after the input file, and if there are multiple options, they can be written in any order. However, if multiple identical options or options of the same kind are written, in some cases an error is generated, and in some cases the last output specified is valid. See **6.4 “Object Converter Options”** for details.
- One or more blanks (spaces or TAB) should be used to separate options and the input file name.
- As the default output destination, a file with the same name as the input file but with the file type changed to ‘.HEX’ is created in the current path. This can be changed by means of the ‘-O’ option.
- **‘RA75X.OM1’** is necessary to start the object converter.

6.3.2 Execution start and end messages

(1) Execution start message

When the object converter is started an execution start message is displayed on the console.

```
75X Series Object Converter VX.XX [XX Xxx XX]
  Copyright (C) NEC Corporation 1985 ,XXXX
```

(2) Execution end message

If the object converter does not find a fatal error, it outputs the following message to the console and returns control to the OS.

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

When the object converter outputs an error message and aborts linkage, the cause of the error message should be found in **13.3 "Object Converter Error Messages"**, and appropriate action taken.

6.3.3 Object converter error handling

If the object converter detects an error during execution, it performs one of the following four kinds of processing according to the severity of the error .

(1) Abort error

If an error is generated which prevents program execution from continuing, the program displays a 'Program aborted' message, and the program is aborted immediately.

★ However, if this type of error is discovered on the object converter start line, execution ends after the check continues for the remainder of the start line.

(2) Fatal error

★ An error in which a HEX format object module file or symbol file which has been generated differs from the user's design is treated as a fatal error. Processing is continued to the extent that other errors can be detected, but a HEX format object module file or a symbol file is not created. (If a file with the same name as the one you are trying to create already exists, the existing file is deleted.)

★ **(3) Warning Error**

If some error is discovered in a portion which is not related to the generation of a HEX format object module file or symbol file, it is counted as a warning error. This is a comparatively minor error, and does not effect the file which is output. Program execution can continue to the very end.

(4) Normal termination

★ If the program terminates normally, it outputs the message "Object Conversion Complete, 0 error(s) and 0 warning(s) found".

In case (1), (2) and (3) above, the error message is output in the following format.

```
***_ERROR_error number_error message
```

★ The error number is expressed as one alphabet character and 3 digits. The header alphabet characters are one of the following letters.

- W (Warning Error)
- F (Fatal Error)
- A (Abort Error)

6.3.4 Object converter termination status

When the object converter terminates and returns control to the OS, one of the following error status codes is returned to the OS.

Termination Conditions	Termination Status
Normal termination	0
★ Warning error	0
Fatal error	1
Abort error	2

When the object converter is started from a batch file under MS-DOS (PC DOS, IBM DOS), it is possible to determine whether there are any errors automatically using these values.

6.4 Object Converter Options

6.4.1 Types of object converter options

Object converter options are used to give the object converter detailed directions concerning its operation. There are seven different options as shown below.

Table 6-2 Object Converter Option Types

No.	Description Format	Function/Category	Default Interpretation
1	-S[file name] -NS	Symbol table file output	'Input file name.SYM' is created in current path
★ 2	-R -NR	Specifies the HEX format object output sequence.	-NR
3	-U fill value	Mask ROM ordering object output specification	None
4	-O[file name] ★ -NO	HEX format object module file specification	'Input file name.HEX' is created in current path
★ 5	-E[File name] -NE	Specifies the error list file.	-NE
★ 6	-Ffile name	Specifies the parameter file.	Reads all the options and file names from the command line.
★ 7	-Ypath name	Specifies the device file search path.	It searches in the sequence of the '··\DEV' path with respect to the OC75X starting path, the OC75X starting path, the current directory, and the path set in the environment variable 'PATH.'

Remark Options can be written in either upper- or lower-case characters

6.4.2 Object converter option specification method

Object converter options are specified in the command line when the object converter is started. See **6.3 "Object Converter Start Method"** for the method of specifying object converter options in the command line.

6.4.3 Object converter option priority order

(1) If multiple identical options or options of the same kind are specified in the command line, the option specified last is valid.
(2) If the same or same kind of option is specified in the parameter file and in the command line, the command line option is valid.

6.4.4 Description of object converter options

Each of the object converter options is described in detail in the following pages.

-S/NS symbols/no symbols

(1) -S/-NS

Description Format	-S[output file name] -NS
Default Interpretation	'Input file name.SYM' is created in current path

[Function]

- The -S option informs the object converter that a symbol table file is to be output.
- The -NS option informs the object converter that a symbol table is not to be output.

[Use]

- The -NS option is specified when symbolic debugging is not to be performed in debugging.

★ **[Description]**

- If the -S option or the -NS option is not specified, the object converter creates a symbol table file on the current path with the initially specified load module file name, but with the file type changed to 'SYM.' It is the same when the file name has been omitted and the -S option has been specified.
- A logical device name (such as 'COM' or 'AUX') cannot be specified as a file name. Describing a file name with a logical device name will result in an error.
- A path name can be described and the path where a symbol table file is created can be specified in a file name. In this case, a file with the load module file name, but with the file type changed to 'SYM' is created on the specified path.
- If -S and -NS are specified at the same time, the option specified last is valid.

-S/-NS symbols/no symbols

[Example]

Example 1. When the -NS option is specified

```
A:\NECTOOLS\SMP75\RA75X>OC75X 75XTEST.LNK -NS
```

→A symbol table file is not output.
The HEX format object module file **“75XTEST.HEX”** only is output.

Example 2. When the -S option is specified

```
A:\NECTOOLS\SMP75\RA75X>OC75X 75XTEST.LNK -S75XTEST.SYM
```

→The symbol table file **“75XTEXT.SYM”** and HEX format object module file **“75XTEXT.HEX”** are output.

(2) -R/-NR

Description format	-R
	-NR
Default Interpretation	-NR

[Function]

- The -R option instructs the arrangement of HEX format objects in address order and outputting them.
- The -NR option instructs the outputting of HEX format objects in the order in which they are stored in load module files.

[Use]

- Use these options when desiring to specify concerning the output order of HEX format objects.

[Explanation]

- If the -R option and -NR option are specified at the same time, the last option specified becomes valid.
- When the -NO option is specified, the -R option and -NR option are invalidated.

[Example of Use]

Example Specify the -R option.

```
A:\NECTOOLS\SMP75X\RA75X>OC75X 75XTEST.LNK -R
```

→Arranges HEX format objects in address order and outputs them.

-U fill up

(3) -U

Description Format	-U filler value [,[start], size]
Default Interpretation	None

[Function]

- The -U option indicates that the specified decimal code is to be placed in all addresses other than those used in the program description and the object code of all the addresses of the target device are to be output.

[Use]

- The -U option is specified when object code for mask ROM ordering is to be output with free ROM area filled with decimal codes.

★ **[Explanation]**

- For filler values, specify a value from 0 to 255 in binary, octal, decimal or hexadecimal notation. If a numerical value outside this range is specified, or something other than a numerical value is specified, or if a filler value is not specified, it will result in an error.
- At the start, use a header address in the address range where you intend to carry out filling, specifying a number between 0H and 0FEFFH in binary, octal, decimal or hexadecimal notation. If a numerical value outside this range is specified, or if something other than a numerical value is specified, it will result in an error. When the start specification is omitted, it is regarded as if 0 was specified.
- In the size, specify a number from 1H to 0FF00H in binary, octal, decimal or hexadecimal notation to be the size of the address area where you intend to carry out filling. If a numerical value outside this range is specified, if something other than a numerical value is specified or if the size was omitted when specifying the start, it will result in an error. If specification of both the start and the size is omitted, it is regarded as if the internal ROM area has been specified.
- The description format and address range are as follows.
  - -U Filler Value : Internal ROM Area
  - -U Filler Value size : From address 0 to the address specified by the size.
  - -U Filler Value, start, size : From the address specified by start to the address specified by the size.
- The final address of the address range where you intend to carry out filling is a value from 1H to 0FEFFH. Specify a size value which does not exceed this range. If this range is exceeded, it will result in an error.
- When the -U option is specified more than once, the last option specified becomes valid. More than one address range cannot be specified.

[Example]

Example The -U option is specified.

```
A:\NECTOOLS\SMP75X\RA75X>OC75X 75XTEST.LNK -U00
```

-U fill up

When -U output is specified

```
:10000000C0500000000000008022990799229911 39
:10001000A2101092E489EE92E09906EE0000000032
:10002000991F9907108BD389C093C09D40A3C0929C
:10003000D67AC0FEBD049B40CAF19A09A3C0AAC1EA
:100040009D809906C7EF9A2ED9E698D998CEF9EEF9
:10005000992110890092807393B399118B3F890085
:10006000E8AA6AFC9910E8AA6AFC10897992A68924
:100070004C92A09DB29D9C991189009A0F9A87FD80
:10008000AB40469210AB400AF000000000000000B8
:1000900000000000000000000000000000000000060
:1000A00000000000000000000000000000000000050
:1000B00000000000000000000000000000000000040
:1000C00000000000000000000000000000000000030
:1000D00000000000000000000000000000000000020
:1000E00000000000000000000000000000000000010
:1000F00000000000000000000000000000000000000
:10010000000000000000000000000000000000000EF
:10011000000000000000000000000000000000000DF
:10012000000000000000000000000000000000000CF
:10013000000000000000000000000000000000000BF
:10014000000000000000000000000000000000000AF
:100150000000000000000000000000000000000009F
:100160000000000000000000000000000000000008F
:100170000000000000000000000000000000000007F
:100180000000000000000000000000000000000006F
```

★ All addresses not used for program description have been filled with the specified decimal code (00).

(4) -O/-NO

Description Format	-O[HEX format object module file name]
★	-NO
Default Interpretation	“Input file name.HEX” is created in current path

[Function]

- The -O option specifies the name of the HEX format object module file to be created by the object converter.
- ★ The -NO option instructs the object converter not to generate a HEX format object module file.

[Use]

- The -O option is specified when it is wished to change the HEX format object module file name from the default name.
- ★ In cases where the object converter is run solely for the purpose of outputting a symbol table file, etc., specify the -NO option.

[Description]

- If the -O operation or the -NO option is not specified, the object converter creates a HEX format object module file with the initially specified load module file name, but with the file type changed to ‘HEX,’ on the current path. This gives the same result as when the -O option is specified with the file name omitted.
- ★ A logical device name cannot be specified as a file name. If a logical device name is specified, it will result in an error.
- It is possible to specify the path in which the HEX format object module file is to be created by including the path name in the file name. In this case, a file with the same name as the load module file. but with the file type changed to ‘.LNK’, is created in the specified path.
- ★ If -O and -NO are specified at the same time, the option specified last is valid.

[Example]

Example 1. When the converter is started with **‘TEST.HEX’** specified as the output file name

```
A:\NECTOOLS\SMP75X\RA75X>OC75X -OTEST.HEX 75XTEST.LNK
```

(5) -E/-NE

Description Format	-E [output file name] -NE
Default Interpretation	-NE

[Function]

- The -E option specifies error list file output, and the output destination and filename.
- The -NE option specifies that no error list file is to be output.

[Use]

- Specify the -E option when desiring to change the error list file output destination or output file name.

[Description]

- When specifying the -E option, if the output file name is omitted, it is regarded as if the output file name ‘Load Module File Name.EOC’ was specified.
- If the drive name is omitted from the file name specification, the current path name is taken as being specified.
- The path name can be described in the file name and the path for generating an error list file can be specified. In this case, a file with the load module file name, but with the file type changed to ‘EOC’ is generated in the specified path.
- The following can be specified as the device type file output destination.
  -EPRN ..... Error list is output tp line printer.
  -ECON ..... Error list is output to console.
  -EAUX ..... Error list is output to RS-232-C.
  -ENUL ..... Error list is not output.

If the -E option and the -NE option are instructed at the same time, the last option specified becomes valid.

(6) -F

Description Format	-F parameter file name
Default Interpretation	Parameter file is not used

[Function]

- The -F option specifies that object converter options and input/output file names are to be read from the file specified by the option parameter. This file is called the parameter file.

[Use]

- Writing options and input/output file names to be specified for the object converter in a parameter file in advance also reduces the amount of typing required.
- Options and input file names can still be specified in the command line even if a parameter file is used. It is thus possible to write only frequently used options in the parameter file.

[Description]

- The parameter file is a text file, and can be created with an editor, etc. There are no particular restrictions on the length of the parameter file.
- The parameter file name cannot be omitted. However, if the file type is omitted' '.POC' is taken as being specified.
- A logical device name ('CON', 'AUX', etc.) and a path name cannot be specified as the parameter file name. Use of such names will result in an error.
- The contents of the parameter file are expanded at the point at which the -F option is written in the object converter start line. It is therefore possible to change the parameter file contents or add other option specifications with options written after the -F option.
- Parameter files cannot be nested. If an -F option is written in the parameter file, an error will result.
- It is not possible to use, more than one parameter at one time. If multiple -F options are specified, an error will result.
- Individual options and input file names should be separated by spaces, TABs or Line Feed characters. A parameter file description cannot be split over a number of lines.
- The ';' and '#' symbols are treated as comment marks in the parameter file. Characters from these characters to the end of the line are regarded as a comment.

★ -Y device file search path

(7) -Y

Description Format	-Y Path name
Default Interpretation	Executes a search in accordance with the specified search sequence **((2) ~ (5) of the [Explanation])**.

[Function]

- The -Y option specifies the device file search path.

[Use]

- Specify the -Y option when searching from the specified path first.

[Explanation]

- A device file is searched for by the following sequence.
  (1) Path specified by the -Y option.
  (2) '··\DEV' path with respect to the OC75X starting path.
  (3) OC75X starting path.
  (4) Current Path
  (5) Environment Variable 'PATH'
- If a name other than a path name is specified, or if the path name is omitted, it will result in an error.

CHAPTER 7. LIBRARIAN

The librarian is a tool for collecting together modules which are of general applicability and have a clear interface in a single file (“librarization”).

Once multiple modules have been collected together in a single file, the library file can be specified as an input file when linkage is performed in addition to the object module file.

Only the necessary modules in the library file are linked.

Creating a library file in this way facilitates object module management and administration.

7.1 Librarian Input/Output Files

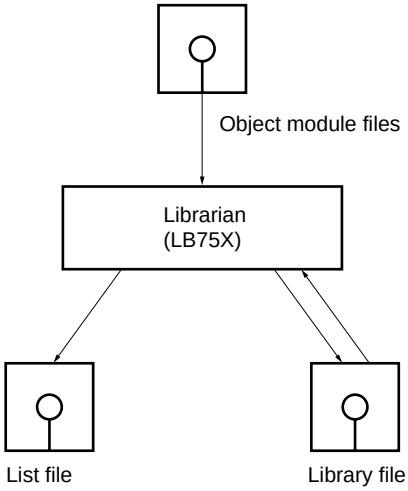
Librarian input/output files are shown in Table 7-1.

Table 7-1 Librarian Input/Output Files

Type of File		Default File Type
Input file	**Object module file** [Note 1] Object module file output by the assembler.	.REL
	Library file File created by the librarian. Specific modules or the all modules in the file can be input.	.LIB
	Subcommand file [Note 2] A series of subcommands for the librarian are created in advance using the editor in the form of a subcommand file.	
Output file	**Library file** File which has been updated by the librarian (additions/deletions/replacements).	.LIB
	List file List file to which information on each module in the library file is output.	.PRN

Notes 1. Binary file.
2. See **7.3.3 “Subcommand file”**.

Figure 7-1. Librarian Input/Output Files



Caution ‘__’ cannot be used as a file name in Librarian. Use ‘-’ instead.

7.2 Librarian Functions

- The main functions of the librarian are as follows:

  1. Librarization of modules
  2. Library file editing
  3. Printing of library file information

- If an error is found during processing, the librarian outputs an error message to the console.
- The librarian performs processing in accordance with the subcommands specified after the librarian is started. See **7.4 “Description of Subcommands”** for the subcommands.
- When the EXIT subcommand is input, the librarian returns control to the OS.

7.2.1 Module librarization

The assembler creates one object module in one file. If there are a large number of object modules, therefore, the number of files also increases. Consequently, a function is provided for collecting together a number of modules in a single file. This is called “module librarization”, and the librarized file is called a “library file”.

A library file can also be input to the linker. Therefore, if a library file is created from generally applicable modules when modular programming is used, efficiency can be improved in terms of both file management and operability.

7.2.2 Library file editing

The librarian has the following functions for editing a library file.

- Addition of modules to library file
- Deletion of modules from library file
- Replacement of modules in library file
  (See **7.4 “Description of Subcommands”** for details of these functions.)

7.2.3 Printing of library file information

The librarian has functions for editing and printing the following information held in a library file.

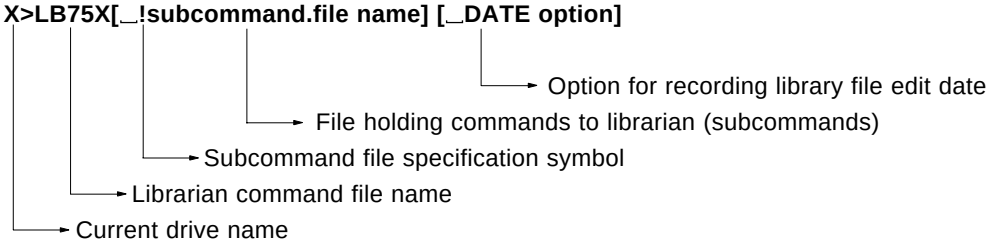
- Module name
- Creating program
- Recording date
- Update date
- PUBLIC symbol information
  (See **7.4 “Description of Subcommands”** for details of these functions.)

7.3 Librarian Start Method

7.3.1 Starting the librarian

The librarian is initiated as shown below.

X>LB75X[␣!subcommand.file name] [␣DATE option]



Cautions 1. The librarian command files “LB75X.COM” and “LB75X.OMO” must be stored in the current directory.

2. If the librarian is started without specification of a subcommand file, the librarian waits for subcommand input. The librarian does not support directories.

7.3.2 Subcommand input in conversational mode

- If a subcommand file is not specified when the librarian is started, the librarian waits for subcommand input (with the ‘*’ prompt displayed) after displaying the start message.

```
A:\NECTOOLS\SMP75X\RA75X>LB75X
75X Series Librarian VX.XX [XX Xxx XX]
Copyright (C) NEC Corporation 1984, XXXX

*
```

- A subcommand is used to give instructions to the librarian, and uses the following format (see **7.4 “Description of Subcommands”** for details of sub-commands).

```
*Subcommand name  Operand information
```

Prompt output by librarian

7.3.3 Subcommand file

- If a series of subcommands to be given to the librarian is decided in advance, these subcommands are compiled into a subcommand file using the editor.
- If a subcommand file is specified when the librarian is started, the librarian reads the subcommands from the subcommand file and performs processing accordingly, and when all the subcommands in the subcommand file have been executed, returns control to the OS.
- The subcommand is created in the following format:

```
Subcommand name    Operand information
Subcommand name    Operand information
                   ⋮
EXIT
```

- Up to 132 characters can be written on one line of the subcommand file.
- If a subcommand does not fit on one line, an ampersand (‘&’) symbol is written at the end of the line to indicate that the subcommand is continued on the next line.
- Characters from a semicolon (‘;’) up to the end of a line are regarded as a comment, and are not interpreted as part of a librarian subcommand.
- Even if there is not EXIT subcommand at the end of the subcommand file, the librarian infers the presence of an EXIT subcommand and terminates the processing.
- When subcommand input ends, processing of each subcommand begins. When processing of one subcommand is completed, ‘*’ is displayed again and the librarian waits for input of the next subcommand. This operation is repeated until the termination subcommand (EXIT subcommand) is input.

```
* Subcommand specification
  Subcommand processing
* Subcommand specification
  Subcommand processing
          ⋮
          ⋮
* Termination subcommand specification
  Librarian termination
```

- Up to 80 subcommand characters can be written on one line.
- If the subcommand operand information specification does not fit on one line, the specification can be continued by using '&'.
- If '&' is specified at the end of a line, the operand input request symbol '-' is printed on the next line, and input of operand information can be continued from the beginning of the next line.

```
*;LIBRARY CREATION COMMAND
*CREATE 75XTEST.LIB
*ADD 75XTEST1.REL 75XTEST2.REL &
-TO 75XTEST.LIB
*LIST 75XTEST.LIB TO SAMPL.LST PUBLICS
*EXIT
```

- An example of a subcommand file is shown below.

```
;
;LIBRARY CREATION COMMAND
;
CREATE 75XTEST.LIB
;
ADD 75XTEST1.REL, 75XTEST2.REL &
      TO 75XTEST.LIB
;
LIST 75XTEST.LIB TO SAMPL.LST PUBLICS
;
EXIT
```

7.3.4 Execution start and end messages

(1) Execution start message

When the librarian is started an execution start message is displayed on the console.

```
75X Series Librarian VX.XX [XX Xxx XX]
  Copyright (C) NEC Corporation 1984, XXXX
```

(2) Execution end message

If no fatal error is detected, the librarian returns control to the OS without outputting a message.

If the librarian outputs an error message and aborts processing, the cause of the error message should be found in **13.4 “Librarian Error Messages”**, and appropriate action taken.

7.3.5 Date option

The librarian has only one option: the DATE option

The DATE option is used to record the date on which the library file was edited as information in the library file, and to print this information as a header in the output list.

DATE date

Description Format	DATE (character string)
Abbreviated Format	DA (character string)
Default Interpretation	DATE (system time)

[Function]

- The DATE record the specified string (date) as the library file edit date in the library file.
- The specified string is also printed in the list file header.

[Use]

- The DATE option is specified in order to record when editing was performed on individual modules in the library file.

[Description]

- A string of up to 12 characters should be specified.

[Examples]

Example 1. The librarian is started with the DATE option specified.
A subcommand file is not specified.

```
A:\NECTOOLS\SMP75X\RA75X>LB75X DATE (XX/XX/XX)
75X Series Librarian VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1984,XXXX

*CREATE 75XTEST. LIB
*ADD 75XTEST1 TO 75XTEST. LIB
*LIST 75XTEST.LIB
```

```
75X Series Librarian VX.XX      DATE(XX/XX/XX                )    PAGE :      X

   LIB-FILE NAME : 75XTEST.LIB          (XX/XX/XX       XX/XX/XX          )

   1  AD_MAIN              (XX/XX/XX                          )
      UPDATE :      0      RA75X VX. XX UPD75106

         NUMBER OF MODULES : 1
```

→The processing date is recorded in the library file.

DATE date

Example 2. The library file **"75XTEST.LIB"** is updated with the DATE option specified.

```
A:\NECTOOLS\SMP75X\RA75X>LB75X DATE XX/XX/XX
75X Series Librarian VX. XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1984, XXXX

*ADD 75XTEST2 TO 75XTEST.LIB
*LIST 75XTEST.LIB
```

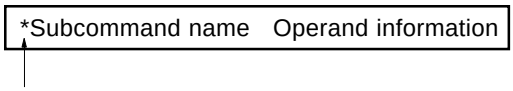
The update date is recorded in the library file.

```
75X Series Librarian VX.XX      DATE(XX/XX/XX              )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB          (XX/XX/XX       XX/XX/XX          )

  1  AD_MAIN              (XX/XX/XX                          )
     UPDATE :      0      RA75X VX. XX UPD75106

  2  AD_SUB               (XX/XX/XX                          )
     UPDATE :      0      RA75X VX. XX UPD75106

        NUMBER OF MODULES : 2
```

In 1, the date on which library file **"75XTEST.LIB"** was edited last remains.
In 2, the update date is recorded.

7.4 Description of Subcommands

Details of each subcommand are given in the following pages.
The description format is as follows:



(1) CREATE

Description Format	CREATE Library file name
Abbreviated Format	C Library file name
Default Interpretation	None

[Function]

- The CREATE subcommand creates a new library file.

[Use]

- The assembler and linker create one output module in one file. If there are a large number of modules, therefore, the number of files also increases.
  Consequently, a function is required for collecting together a number of modules in a single file. This function is called “module librarization”, and the librarized file is called a “library file”. If there are a large number of modules, a library file can be created by specifying the CREATE subcommand.

[Description]

- The name of the library file to be created should be specified as the operand.
- There are no modules in the initially created library file.

[Examples]

Example 1. Consider an initially created library file named **“75XTEST.LIB”**.

```
*CREATE 75XTEST.LIB
```

→The library file **“75XTEST.LIB”** is created.

Example 2. The library file name is omitted.

```
*CREATE
*** ERROR W212 ILLEGAL FILE SPECIFICATION :
*
```

→Omission of the library file name results in an error.

Example 3. Multiple library files are specified for initialization.

```
*CREATE CLIB.LIB DLIB.LIB
*** ERROR W209 PARAMETER OVER
*
```

→If multiple library files are specified for initialization, as shown here, an error results. However, the first library file specified is created.

(2) ADD

Description Format

*ADD	Object module file name Library file name [(Object module name [, ...])]	[, ...] TO	Update library file name

Abbreviated Format

*A	Object module file name Library file name [(Object module name [, ...])]	[, ...] TO	Update library file name

[Function]

- The ADD subcommand specifies that one or more modules in a different file are to be added to an existing library file.

[Use]

- The ADD subcommand is specified when it is wished to newly record one or more modules in a library file initially created with the CREATE subcommand.

[Description]

- The object module file or library file containing the modules to be added to the library file is specified as the input file.
- If the file type is omitted from the input file name, it is taken to be '.REL'.
- If the input file is a library file, the name of the module in the library file to be recorded is specified in parentheses ( ).
- If the module name is omitted, all the modules in the library file are recorded.
- The update library file name is the name of the library file to which the addition is made.

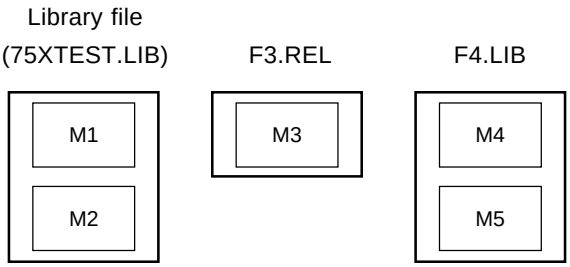
Caution

The update library file must not contain a module with the same name as that of the module to be added.

[Example]

Example 1. Modules M3 and M4 are to be added to the library file **“75XTEST.LIB”.**

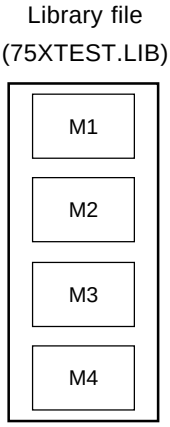
<Before addition>



Description format

*ADD F3, F4, (M4) TO 75XTEST.LIB

<After recording>



Example 2. Module **“75XTEST1”** is to be added to the existing library file **“75XTEST.LIB”**.

```
*ADD 75XTEST1 TO 75XTEST.LIB
*
```

→The contents of the library file **“75XTEST.LIB”** are checked.

```
*LIST 75XTEST.LIB

75X Series Librarian VX.XX      DATE(                    )    PAGE :      X

   LIB-FILE NAME : 75XTEST.LIB          (                              )

   1  AD_MAIN              (                                  )
      UPDATE :       0      RA75X VX. XX UPD75106

         NUMBER OF MODULES : 1
```

(3) DELETE

Description Format	*DELETE Library file name (Object module name [, …])
Abbreviated Format	*D Library file name (Object module name [, …])

[Function]

- The DELETE subcommand specifies that one or more modules in an existing library file are to be deleted.

[Use]

- The DELETE subcommand is specified when it is wished to delete one or more modules which are no longer needed from a library file.

[Description]

- The library file name specified is the name of the library file containing the module(s) to be deleted.
- The module name specified is the name of the module to be deleted from the library file.

Caution

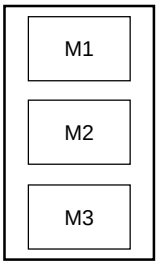
The specified module must exist in the library file.

[Example]

Example 1. Modules M1 and M3 are to be deleted from the library file **“75XTEST.LIB”**.

<Before deletion>

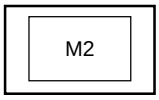
75XTEST.LIB



DELETE delete

- Description format
  *DELETE 75XTEST.LIB (M1, M3)

<After deletion>
75XTEST.LIB



Only module M2 remains in the library file.

Example 2. Module **“AD_SUB”** is to be deleted from existing library file **“75XTEST.LIB”**.

<1> First, the contents of the library file are checked.

```
*LIST 75XTEST.LIB

75X Series Librarian VX.XX      DATE(                    )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB        (                    )

  1  AD_MAIN            (                    )
     UPDATE :      0     RA75X VX. XX UPD75106

  2  AD_SUB             (                    )
     UPDATE :      0     RA75X VX. XX UPD75106

       NUMBER OF MODULES : 2
```

<2> Module **“AD_SUB”** is deleted.

```
*DELETE 75XTEST.LIB (AD_SUB)
```

<3> The contents of library file **“75XTEST.LIB”** are checked.

```
*LIST 75XTEST.LIB

75X Series Librarian VX.XX      DATE(                    )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB          (                    )

  1  AD_MAIN              (                    )
     UPDATE :      0      RA75X VX. XX UPD75106

        NUMBER OF MODULES : 1
```

→The module **“AD_SUB”** is deleted and **“AD_MAIN”** only remains.

(4) REPLACE

Description Format

```
                  { Object file name                          }
*REPLACE          { Library file name                         }  FROM  Update library file name
                  { [(Object module name [, ...] ) ]          }

                  { Object file name                          }
*R                { Library file name                         }  FROM  Update library file name
                  { [(Object module name [, ...] ) ]          }
```

[Function]

- The REPLACE subcommand specifies that one or more modules in an existing library file are to be replaced with modules from another object module file or library file.

[Use]

- The REPLACE subcommand is specified when it is wished to update the recorded module contents.

[Description]

- The object module file or library file containing the module(s) is specified as the input file.
- If the input file is a library file, the name of the module in the library file to be replaced is specified in parentheses ( ).
- If a module name is not specified, all the modules in the library file are replaced.
- The update library file name is the name of the library file from which the replacement is made.

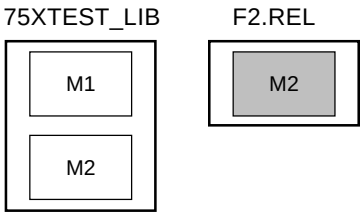
Caution

- **The update library file must contain a module with the same name as that of the module to be replaced.**
- **The input library file name and the update library file name must be different.**

[Example]

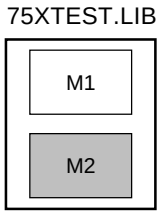
Example 1. Module M2 in the library file **“75XTEST.LIB”** is to be replaced.

<Before replacement>



- Description format
  *REPLACE F2 (M2) FROM 75XTEST.LIB

<After replacement>



M2 in library file ’75XTEST.LIB’ has been replaced.

Example 2

<1> The contents of public symbols in the library file **“75XTEST.LIB”** are to be checked.

```
*LIST 75XTEST.LIB PUBLICS

75X Series Librarian VX.XX      DATE(                              )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB          (                                  )

  1  AD_MAIN              (                                  )
     UPDATE :      0      RA75X VX. XX UPD75106

     SEG0
     SEG1
     SEG2
     SEG3
     SEG15
     TDATA

        NUMBER OF PUBLIC SYMBOLS : 6

  2  AD_SUB               (                                  )
     UPDATE :      0      RA75X VX. XX UPD75106

     ADCONV
     SEG4
     SEG5
     SIOSUB

        NUMBER OF PUBLIC SYMBOLS : 4

        NUMBER OF MODULES : 2
```

REPLACE replace

<2> “75XTEST1.ASM” is assembled after rewriting as shown below.

```
$      TITLE=' A-D CONVERTER VX.XX'
;*****************************************************
;***   A-D CONVERT PROGRAM            ***
;*****************************************************
          NAME      AD_MAIN
          EXTRN     CODE(ADCONV), CODE(SIOSUB), CODE(HEIKIN)
          PUBLIC    TDATA,SEL15
          STKLN     10
          VENT0     MBE=1, RBE=1, MAIN
          VENT4     MBE=1, RBE=0, ADCONV

SEG0      DSEG      1 AT 10H
TDATA:    DS        2

;***      GETI      TABLE          ***

SEG1      CSEG      IENT
SEL15:    SEL       MB15

;***      MAIN ROUTINE             ***

SEG2      CSEG      INBLOCK
MAIN :    SEL       RB1

          GETI      SEL15                ;STACK POINTER SET
          MOV       XA,#STACK              ;
          MOV       SP,XA                  ;

          MOV       A,#0011B
          MOV       PCC,A                ;PCC ← 0011B

;**       DATA RAM 0H-13FH ZERO CLEAR                **

          SEL       MB1
          MOV       HL,#3FH
          MOV       XA,#00H
LOOP1:    MOV       @HL,A                ;100H-13FH
          DECS      HL
          BR        LOOP1
          SEL       MB0
LOOP2:    MOV       @HL,A                ;0H-FFH
          DECS      HL
          BR        LOOP2
```

REPLACE replace

```
;**     TIMER SET (SAMPLING TIME = 30 MSEC, FXX=4.19 MHZ)     **

        GETI    SEL15       ;SEL    MB15
        MOV     XA,#79H
        MOV     TMOD0,XA
        MOV     XA,#01001100B
        MOV     TM0,XA
        EI
        EI      IET0

        SEL     MB1
LOOP3:  MOV     XA,#00H
        MOV     B,#0H
LOOP4:  SKE     B,#08H
        BR      LOOP4
        CALL    !HEIKIN
        MOV     TDATA,XA
        CALL    !SIOSUB
        BR      LOOP3

        END
```

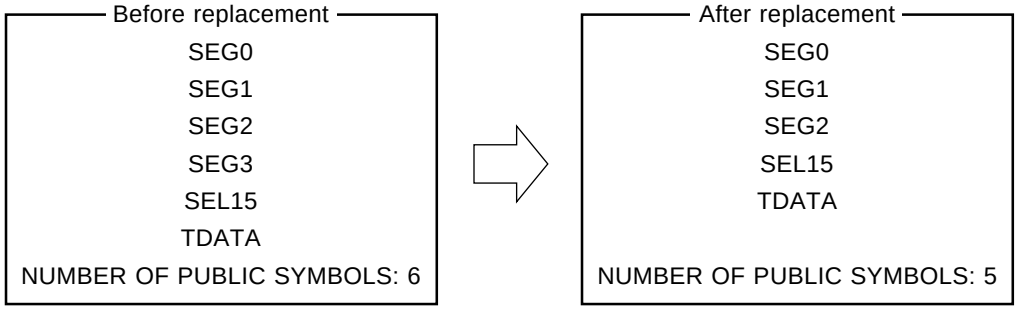
<3> Modules in library file “75XTEST.LIB” are replaced.

```
*REPLACE 75XTEST1 FROM 75XTEST.LIB
```

<4> Check the contents of the public symbols in the library file **"75XTEST.LIB"** after replacement.

```
*LIST 75XTEST.LIB PUBLICS

75X Series Librarian VX.XX      DATE(                    )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB        (                    )

  1  AD_MAIN             (                    )
     UPDATE :      0      RA75X VX. XX UPD75106

     SEG0
     SEG1
     SEG2
     SEG15
     TDATA

        NUMBER OF PUBLIC SYMBOLS : 5

  2  AD_SUB              (                    )
     UPDATE :      0      RA75X VX. XX UPD75106

     ADCONV
     SEG4
     SEG5
     SIOSUB

        NUMBER OF PUBLIC SYMBOLS : 4

        NUMBER OF MODULES : 2
```

→The 'AD_MAIN' public symbols in the library file **"75XTEST.LIB"** before replacement are changed in **"75XTEST1"** after replacement as follows.



Example 3.

<1> The contents of the library file **"75XTEST.LIB"** are to be checked.

```
*LIST 75XTEST.LIB PUBLICS

75X Series Librarian VX.XX      DATE(                              )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB          (                                  )

  1  AD_MAIN              (                                  )
     UPDATE :      0      RA75X VX. XX UPD75106

     SEG0
     SEG1
     SEG2
     SEG3
     SEG15
     TDATA

        NUMBER OF PUBLIC SYMBOLS : 6

  2  AD_SUB               (                                  )
     UPDATE :      0      RA75X VX. XX UPD75106

     ADCONV
     SEG4
     SEG5
     SIOSUB

        NUMBER OF PUBLIC SYMBOLS : 4

        NUMBER OF MODULES : 2
```

REPLACE replace

<2> **“75XTEST1.ASM”** is assembled after rewriting as shown below.

```
$      TITLE=' A-D CONVERTER VX.XX'
;*****************************************************
;***   A-D CONVERT PROGRAM            ***
;*****************************************************
          NAME      AD_MAIN
          EXTRN     CODE(ADCONV), CODE(SIOSUB), CODE(HEIKIN)
          PUBLIC    TDATA,SEL15
          STKLN     10
          VENT0     MBE=1, RBE=1, MAIN
          VENT4     MBE=1, RBE=0, ADCONV

SEG0      DSEG      1 AT 10H
TDATA:    DS        2

;***      GETI      TABLE        ***

SEG1      CSEG      IENT
SEL15:    SEL       MB15

;***      MAIN ROUTINE           ***

SEG2      CSEG      INBLOCK
MAIN :    SEL       RB1

          GETI      SEL15               ;STACK POINTER SET
          MOV       XA,#STACK             ;
          MOV       SP,XA                 ;

          MOV       A,#0011B
          MOV       PCC,A               ;PCC ← 0011B

;**       DATA RAM 0H-13FH ZERO CLEAR              **

          SEL       MB1
          MOV       HL,#3FH
          MOV       XA,#00H
LOOP1:    MOV       @HL,A               ;100H-13FH
          DECS      HL
          BR        LOOP1
          SEL       MB0
LOOP2:    MOV       @HL,A               ;0H-FFH
          DECS      HL
          BR        LOOP2
```

REPLACE replace

```
;**     TIMER SET (SAMPLING TIME = 30 MSEC, FXX=4.19 MHZ)     **

        GETI    SEL15       ;SEL    MB15
        MOV     XA,#79H
        MOV     TMOD0,XA
        MOV     XA,#01001100B
        MOV     TM0,XA
        EI
        EI      IET0

        SEL     MB1
LOOP3:  MOV     XA,#00H
        MOV     B,#0H
LOOP4:  SKE     B,#08H
        BR      LOOP4
        CALL    !HEIKIN
        MOV     TDATA,XA
        CALL    !SIOSUB
        BR      LOOP3

        END
```

<3> A new file **“CLIB.LIB”** is created by LB75X, and its contents are checked.

```
*LIST CLIB.LIB PUBLICS

75X Series Librarian VX.XX      DATE(                    )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB        (                    )

  1  AD_MAIN             (                    )
     UPDATE :      0      RA75X VX. XX UPD75106

     SEG0
     SEG1
     SEG2
     SEL15
     TDATA

        NUMBER OF PUBLIC SYMBOLS : 5

  2  AD_SUB              (                    )
     UPDATE :      0      RA75X VX. XX UPD75106

     ADCONV
     SEG4
     SEG5
     SIOSUB

        NUMBER OF PUBLIC SYMBOLS : 4

        NUMBER OF MODULES : 2
```

<4> **“CLIB.LIB”** is replaced with **“75XTEST.LIB”**.

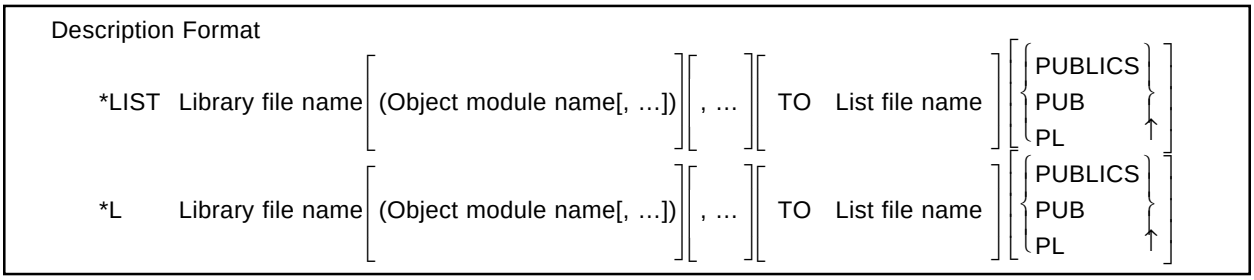
```
*REPLACE CLIB.LIB FROM 75XTEST.LIB
```

→The public symbol information in **“75XTEST.LIB”** after replacement is as follows.

```
*LIST 75XTEST.LIB PUBLICS

75X Series Librarian VX.XX      DATE(                              )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB          (                                  )

  1  AD_MAIN              (                                  )
     UPDATE :      1      RA75X VX. XX UPD75106

     SEG0
     SEG1
     SEG2
     SEL15
     TDATA

        NUMBER OF PUBLIC SYMBOLS : 5

  2  AD_SUB               (                                  )
     UPDATE :      1      RA75X VX. XX UPD75106

     ADCONV
     SEG4
     SEG5
     SIOSUB

        NUMBER OF PUBLIC SYMBOLS : 4

        NUMBER OF MODULES : 2
```

(5) LIST

Description Format



[Function]

- The LIST subcommand specifies that information on modules in the library file are to be output to the list file.

[Use]

- The LIST subcommand is specified when it is wished to obtain information on modules recorded in a library file.

[Description]

- The name of the library file for which information is to be printed is specified as the library file name.
- If only information for a specific module is to be printed, the relevant module name is specified in parentheses ( ).
- If a module name is not specified, information on all the modules in the library file is printed out.
- The name of the file to which the print information is to be output is specified as the list file name.
- The following can be specified as the list file name:
  PRN .... Output to printer
  CON .... Output to console
- If information on PUBLIC symbols defined in the module is to be output, ‘PUBLICS’ is specified.
  PUBLIC symbols are symbols declared by the assembler PUBLIC pseudo-instruction.

- Output information is as shown below.

Library information	• Creation date, update date • Number of recorded modules
Module information	• Module name • Creation program name • Recording date, update date • Number of updates • PUBLIC symbols defined in module • Number of PUBLIC symbols

Caution

If the list file name is omitted, the information is output to the console.

[Example]

Example 1. To output information on all modules in **“75XTEST.LIB”**.

```
*LIST 75XTEST.LIB
75X Series Librarian VX.XX      DATE(                    )    PAGE :      X

  LIB-FILE NAME : 75XTEST.LIB        (                      )

  1  AD_MAIN             (                          )
     UPDATE :      0      RA75X VX. XX UPD75106

  2  AD_SUB              (                          )
     UPDATE :      0      RA75X VX. XX UPD75106

        NUMBER OF MODULES : 2
```

Example 2. To output information on the specific module ‘AD_MAIN’ in **“75XTEST.LIB”**.

```
*LIST 75XTEST.LIB(AD_MAIN)
75X Series Librarian VX.XX      DATE(                    )    PAGE :      X

   LIB-FILE NAME : 75XTEST.LIB          (                      )

   1  AD_MAIN              (                                  )
      UPDATE :      0      RA75X VX. XX UPD75106

         NUMBER OF MODULES : 1
```

Example 3. To output information on PUBLIC symbols in the specific module ‘AD_MAIN’ in **“75XTEST.LIB”**.

```
*LIST 75XTEST.LIB (AD_MAIN) PUBLICS

75X Series Librarian VX.XX      DATE(                    )    PAGE :      X

   LIB-FILE NAME : 75XTEST.LIB          (                      )

   1  AD_MAIN              (                                  )
      UPDATE :      0      RA75X VX. XX UPD75106

      SEG0
      SEG1
      SEG2
      SEG3
      SEG15
      TDATA

         NUMBER OF PUBLIC SYMBOLS : 6

         NUMBER OF MODULES : 1
```

(6) EXIT

Description Format	*EXIT
Abbreviated Format	*E

[Function]

- The EXIT subcommand specifies that the librarian is to be terminated.

[Use]

- The EXIT subcommand is specified to terminate the librarian.

[Example]

- Terminates the librarian.

```
*EXIT
A:\NECTOOLS\SMP75X\RA75X>
```

[MEMO]

CHAPTER 8. LIST CONVERTER

The list converter (LCNV75X) has as input an assembly list file and object module file output by the assembler and a load module file output by the linker, and outputs an absolute assembly list in which actual values are incorporated in the relocatable addresses and object code in the assembly list file.

Debugging efficiency can be improved by performing program debugging (using an IE-75000-R [Note 1], IE-75001-R or EVAKIT-75X [Note 2]) while referring to the absolute assembly list.

Notes 1. Maintenance product (not available for purchase)

2. Discontinued (not available for purchase)

8.1 List Converter Input/Output Files

List converter (LCNV75X) input/output files are shown in Table 8-1.

Table 8-1 List Converter Input/Output Files

Type of File		Default File Type
Input file	**Assembly list file** Note 1 Assembly list file output by the assembler.	.PRN
	Object module file Notes 1, 2 Object module file output by the assembler.	.REL
	Load module file Note 2 Load module file output by the linker. The list converter calculates the actual values from this file.	.LNK
★	**Parameter file** File for creating execution program parameters User-created file	.PLV
Output file	**Absolute assembly list file** List file in which with the relocatable values in the assembly list output by the assembler are replaced with actual values determined by the linker.	.P
★	**Error list file** This is file which contains error information when running the list converter.	.ELV

Notes 1. The input assembly list file and object module file must have been output as the result of assembly of the same source program.

Also, since addresses in the assembly list file created by the assembler are virtual addresses, the addresses in the absolute assembly list file created by the list converter should be referred to.

2. Binary file

★ **Figure 8-1 List Converter Input/Output Files**

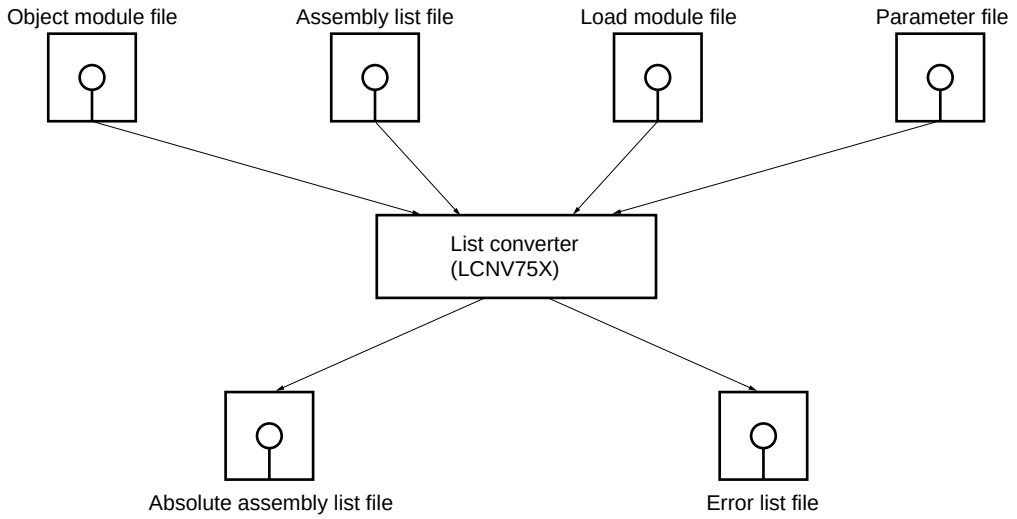
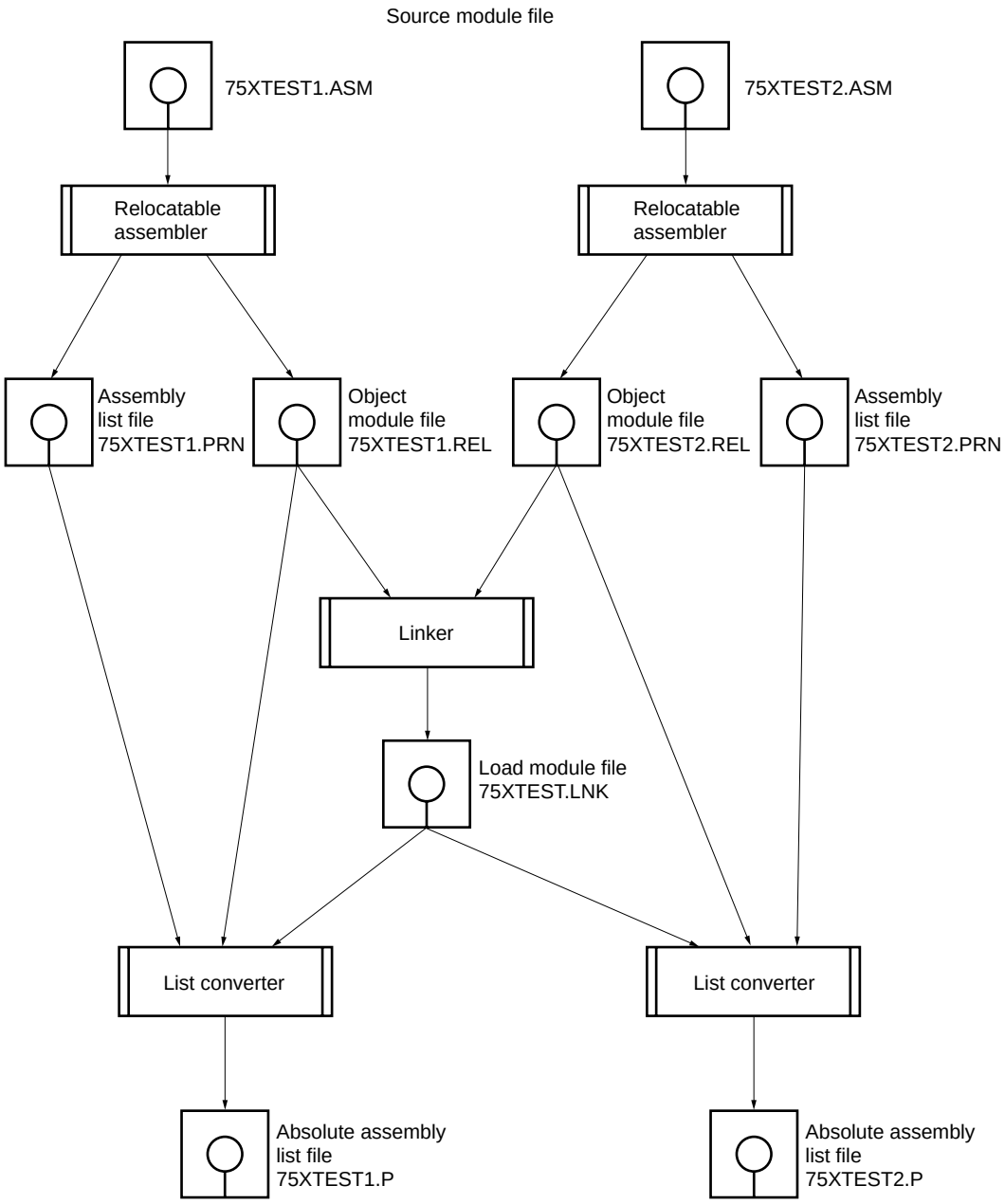


Figure 8-2 Example of List Converter Input/Output Files



8.2 List Converter Functions

The list converter searches among the modules in the input load module file (.LNK) for modules in the object module file (.REL) input at the same time. The list converter reads from the relevant module information values determined during linkage for the relocatable addresses and temporary object code which were not determined during assembly. It then inserts these values in the appropriate places in the input assembly list and outputs the result as an absolute assembly list file.

8.2.1 Incorporation of location addresses

Offset addresses with 0000H as the start of the segment are incorporated as relocatable segment addresses in the assembly list output by the assembler. The list converter incorporates absolute location addresses determined during linkage in place of these relocatable location addresses.

Example Following assembly list is input to the list converter.

<Assembly list>

```
19   ----                          SEG2   CSEG   INBLOCK
20   0000      9921                MAIN:  SEL    RB1
21
22   0002  R   00                        GETI   SEL15          ;STACK POINTER SET
23   0003  E   8900                      MOV    XA,#STACK       ;
24   0005      9280                      MOV    SP,XA           ;
25
26   0007      73                        MOV    A,#0011B
27   0008      93B3                      MOV    PCC,A          ;PCC ← 0011B
```

Temporary values are incorporated as the location addresses.

Absolute addresses are incorporated during linkage in the location addresses, and output as an absolute assembly list.

<Absolute assembly list>

```
19   ----                          SEG2   CSEG   INBLOCK
20   0050      9921                MAIN:  SEL    RB1
21
22   0052  R   10                        GETI   SEL15          ;STACK POINTER SET
23   0053  E   8900                      MOV    XA,#STACK       ;
24   0055      9280                      MOV    SP,XA           ;
25
26   0057      73                        MOV    A,#0011B
27   0058      93B3                      MOV    PCC,A          ;PCC ← 0011B
```

Absolute addresses are incorporated.

8.2.2 Incorporation of object code

Temporary values (with 00H used as the data value indicated by relocatable symbols) are incorporated in the object code of instructions which reference relocatable symbols (including external reference symbols) in the assembly list output by the assembler. The list converter replaces this temporary object code with the correct object code determined during linkage.

Example Following assembly list is input to the list converter.

<Assembly list>

```
54   002B      9A0F                    MOV    B,#00H
55   002D      9A87          LOOP4:    SKE    B,#08H
56   002F      FD                      BR     LOOP4
57   0030  R   AB4000                  CALL   !HEIKIN
58   0033      9210                    MOV    TDATA,XA
59   0035  E   AB4000                  CALL   !SIOSUB
```

Temporary values are incorporated in the object code.

An absolute assembly list is output in which the object code of the CALL instructions which reference the relocatable symbol ‘HEIKIN’ and the external reference symbol ‘SIOSUB’ is replaced with the correct values determined during linkage.

<Absolute assembly list>

```
54   007B      9A0F                    MOV    B,#00H
55   007D      9A87          LOOP4:    SKE    B,#08H
56   007F      FD                      BR     LOOP4
57   0080  R   AB4046                  CALL   !HEIKIN
58   0083      9210                    MOV    TDATA,XA
59   0085  E   AB400A                  CALL   !SIOSUB
```

Correct object code is incorporated.

Caution The code of the VENTn pseudo-instruction is not converted by the list converter.

8.2.3 List converter processing method

List converter processing consists of two stages, called ‘passes’.

In ‘pass 1’ the list converter checks the contents of the input object module file and load module file, and searches the load module file for modules, corresponding to the object module file. In ‘pass 2’, the list, converter incorporates absolute values in the assembly list, based on the module information found, and outputs an absolute assembly file.

As a guide to the processing stage, the list converter outputs the following images to the console during processing. Each dot output represents one segment. In ‘pass 2’, the remaining segments are skipped when all the segments in the relevant module have been processed, so that fewer dots may be output in this pass than in ‘pass 1’.

```
pass 1: start .....
pass 2: start .....
```

8.2.4 Points to note when using the list converter

When using the list converter, the following points should be noted when writing the source program.

- The VENTn and ORG pseudo-instructions should be written in upper-case characters starting in column 9 of the source program.
- The NOLIST control instruction should not be used.
- Segments with the same name should not be used in the same module.
- A segment definition pseudo-instruction or ORG pseudo-instruction must be written before writing an instruction which generates object code.

Also, the list converter does not correct the symbol values of a cross-reference list or symbol list included in the assembly list.

All files input to the list converter must be free of errors.

8.3 List Converter Start Method

8.3.1 List starting the list converter

The list converter is started as shown below.

★

```
X>LCNV75X [␣List converter option[...] ]␣file specification
          |                |
          |                Specifies list converter options. Note
          List converter command file name
Current drive name
```

Note If multiple options are specified, they are separated by spaces. See **8.4.2 "List converter options"** for details.

(1) Command line specification

★

```
X>LCNV75XL␣-L file name␣-A file specification [␣-O output file name]
          |                |                  |
          |                |                  Output file name Note 3
          |                Primary name of assembly list file and object module file output by assembler Note 2
          Name of load module file output by linker Note 1
```

Notes 1. The file type (.LNK) should also be specified.

2. The file type of the assembly list file and the object module file must be ' . PRN' and ' '.REL' respectively.

3. The name of the file to which the absolute assembly list created by the list converter is to be output should be specified.
If omitted, a specified by file is output with the primary name file type '.P'.

4. -L, and -O can be specified in any order.

Caution Spaces may not be inserted between options and file names.

8.3.2 Execution start and end messages

(1) Execution start message

When the list converter is started, a start message is displayed on the console.

```
List Conversion Program for RA75X VX.XX [XX Xxx XX]
  Copyright (C) NEC Corporation 1986, 1997
```

After that, when the list converter then asks for input file names in conversational mode, specify the appropriate file.

(2) Execution end message

- If processing terminates normally, the list converter outputs the following message to the console and returns control to the OS.

```
Conversion complete
```

- If a fatal error is detected during processing which prevents the conversion processing from continuing, the list converter outputs a message to the console, stops execution, and returns control to the OS.

Example If there is an error in the input/output file specification method, the list converter outputs the following message to the console and returns control to the OS.

★

```
List Conversion Program for RA75X VX.XX [XX Xxx XX]
  Copyright (C) NEC Corporation 1986, 1997

Usage: LCNV75X option input-file option
Please enter 'LCNV75X --', if you want help messages.
```

- If the list converter outputs an error message (error number) and aborts processing, the cause of the error message should be found in **13.5 List Converter Error Messages**, and appropriate action taken.

Caution

If there is an error in the contents of the input assembly list file, the list converter will create an output file in which conversion is incomplete. You should check for an error message on the console even if an output file is produced.

8.3.3 List converter error handling

If the list converter detects an error during execution, it performs one of the following three kinds of processing according to the severity of the error.

(1) Abort error

If an error is generated which prevents program execution from continuing, the program displays a 'Program aborted' message, and the program is aborted immediately.

(2) Fatal error

If an error is generated which would result in generation of object code different from that intended by the user, the program nevertheless continues processing to the end, then outputs the message "X ERRORS FOUND" (when X is the number of errors) .

(3) Normal termination

If the program terminates normally, it outputs the message "NO ERROR FOUND"

In cases (1) and (2) above, the error message is output in the following format.

```
***␣ERROR␣error number␣error message
```

The error number consists of a letter followed by a 3-digit number. The initial letter is one of the following:

W (Warning error)
F (Fatal error)
A (Abort error)

8.3.4 List converter termination status

When the list converter terminates the execution and returns control to the OS, one of the following error status codes is returned to the OS.

Termination Condition	Termination Status
Normal termination	0
Fatal error	1
Abort error	2

When the list converter is started from a batch file under MS-DOS (PC DOS, IBM DOS), it is possible to determine whether there are any errors automatically using these values.

8.4 List Converter Options

8.4.1 Types of list converter options

List converter options are used to specify the input/output files to be used by the list converter. There are five options as shown in **Table 8-2** below. Options can be written in either upper- or lower-case characters.

★ **Table 8-2 List Converter Option Types**

Item No.	Description Format	Function, Classification	Interpretation when Omitted
1	-L [file name]	Specifies the input load module file name.	Input assembly list file primary name.LNK.
2	-O [file name]	Specifies the output absolute assembly list file name.	Input assembly list file primary name.P
3	-R [file name]	Specifies the input object module file name.	Input assembly list file primary name.REL
4	-E [file name] -NE	Specifies the error list file name.	-NE
5	-F file name	Specifies the parameter file.	Read all options and file names from the command line.

8.4.2 List converter options

Each of the list converter options is described in detail in the following pages.

-L link file name

(1) -L

Description Format	-L [file name]
★ Default	Input assembly list file primary name.LNK

[Function]

- The -L option specifies the name of the load module file to be input.

[Use]

- The -L option is specified when the input files are specified in the command line.

[Description]

- If there is a -L option is not specified, the file input assembly list file, primary name file type '.LNK'.
- If the only the primary name of the input file is specified, input a file with the same name, but with the file type '.LNK' added

[Examples]

Example 1. When the -L option is specified

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -L75XTEST.LNK 75XTEST1
List Conversion Program for RA75X VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1986, 1987

Pass 1: start .....................
Pass 2: start ...........
Conversion complete
```

-L link file name

Example 2. When the -L option is not specified

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -75XTEST1

List Conversion Program for RA75X VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1986, 1997

 pass 1: start .....................
 pass 2: start .....
conversion complete
```

→The list converter asks for the load module file name, and this should therefore be input. When the file name has been input, the converter processing is started.
If the specified file does not exist or the file type is omitted, the following message is output and control is returned to the OS.

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X 75XTEST1

List Conversion Program for RA75X VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1986, 1997

A006 File not found '-I75XTEST1.LNK'
Program aborted
```

Example 3. Insert a space between -L and the primary name

```
A\NECTOOLS\SMP75X\RA75X>LCNV75X -L 75XTEST.LNK

List Conversion Program for RA75X V1.1 [19 Mar 88]
   Copyright (C) NEC Corporation 1986, 1997

A002 Too many input file
Please enter 'LCNV75X--', if you want help messages.
Program aborted
```

Caution

A space cannot be inserted between -L and the file name. If a space is inserted, the list converter will abort as shown in Example 3.

-O output file name

(2) -O

Description Format	-O [file name]
Default	The file is output with the file extension '.P' added to the input assembly list file primary name.

[Function]

- The -O option specifies the name of the file to which the absolute assembly list is to be output.

[Use]

- The -O option is specified when the absolute assembly list is to be output directly to the printer or the disk or file name is to be changed from the default specification.

[Description]

- If the absolute assembly list is to be output directly to the printer, '.PRN' should be specified as the file name.
- If the -O option is not specified, a file with the same primary name as the input assembly list file and file type '.P' is output to the disk containing the input assembly list file.

[Example]

Example 1. When the -O option is specified

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -L75XTEST.LNK 75XTEST1 -OSAMPL

List Conversion Program for RA75X VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1986, 1997

Pass 1: start .....
Pass 2: start .....
Conversion complete
```

→The absolute assembly list **"SAMPL.P"** is output.

-O output file name

Example 2. When the file name **"SAMPLE.LST"** is specified by the -O option

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -L75XTEST.LNK 75XTEST1 -OSAMPL.LST

List Conversion Program for RA75X VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1986
Pass 1: start .............
Pass 2: start ........
Conversion complete
```

→The absolute assembly list **"SAMPL.LST"** is output.

Example 3. When the -O option is not specified

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -L75XTEST.LNK 75XTEST1

List Conversion Program for RA75X VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1986
Pass 1: start .............
Pass 2: start ........
Conversion complete
```

In this example the input and output files are as shown below.

• Input load module file	75XTEST.LNK
• Input assembly list file	75XTEST1.PRN
• Input object module file	75XTEST1.REL
• Output absolute assemble list file	75XTEST1.P

Caution

A space cannot be inserted between -O and the file name.
If a space is inserted, an error will be output.

-R relocatable file name

★ **(3) -R**

Description Format	-R [file name]
Default	Input assembly list file primary name.REL

[Function]

- The -R option specifies the name of the object module file to be input.

[Use]

- If the primary name of the object module file differs from the primary name of the assembly list file, or if the file type is not ".REL", specify the -R option.

[Description]

- If there is a fatal error, the absolute assembly list is not output.
- If the only the primary name of the input file is specified, input a file with the same name, but with the file type '.REL' added.

[Example]

- If the -O option is specified (load module file **"75XTEST.LNK"**, assembly list file **"75XTEST1.PRN"**, object module file **"SAMPLE.REL"**)

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -L75XTEST.LNK 75XTEST1 -RSAMPLE.REL
```

→The absolute assembly list file **"75XTEST1.P"** is output.

-E/-NE error print/no error print

★ **(4) -E/-NE**

Description Format	-E [output file name]
	-NE
Default Interpretation	-NE

[Function]

- The -E option specifies error list file output, and the output destination and filename.
- The -NE option specifies that no error list file is to be output.

[Use]

- When the assembly list is very long, it is difficult to find error lines in the list. In this case, the -E option can be specified to extract only assembly error information.

[Description]

- If the output file name is omitted when the -E option is specified, “source module file name.ERA” is taken as being specified as the output file name.
- If the drive name is omitted from the file name specification, the current path name is taken as being specified.
- If the same output file name as the specified by the -P option is specified, an error list is not output.
- The following can be specified as the device type file output destination.
  -EPRN ..... Error list is output to line printer.
  -ECON ..... Error list is output to console.
  -EAUX ..... Error list is output to RS-232-C.
  -ENUL ..... Error list is not output.
- If the same device as the absolute assembly list file is specified in the file name, it will result in an abort error.

[Example]

- If **“75XTEST1.ASM”** is assembled with the -E option specified (the file name is **“75XTEST.ELV”**).

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X 75XTEST1 -L75XTEST.LNK -E75XTEST.ELV
```

→If there was an error, the error list file **“75XTEXT.ELV”** is output.

-F parameter file name

★ **(5) -F**

Description Format	-F parameter file name
Default Interpretation	Parameter file not used.

[Function]

- The -F option specifies that the list convert option and input file name are to be read from the file specified by the option parameter. This file is called the parameter file.

[Use]

- Writing options and input file names to be specified for the list convert in a parameter file in advance also reduces the amount of typing required.
- Options and input file names can still be specified in the command line even if a parameter file is used. Therefore, it is possible to write only frequently used options in the parameter file.

[Description]

- The parameter file is a text file, and can be created with an editor, etc. There are no particular restriction on the length of the parameter file.
- The parameter file name cannot be omitted. However, if the file type is omitted, '.PLV' is taken as being specified.
- A logical device name ('CON', 'AUX', etc.) cannot be specified as the parameter file name. Use of such names will result in an error.
- The contents of the parameter file are expanded at the position at which the -F option is written in the assembler start line. It is therefore possible to change the parameter file contents or add other option specifications with options written after the -F option.
- Parameter files cannot be nested. If an -F option is written in the parameter file, an error will result.
- It is not possible to use more than one parameter at one time. If multiple -F options are specified, an error will result.
- Individual options and input file names should be separated by spaces, TABs or Line Feed characters. A parameter file description cannot be split over a number of lines.
- The ';' and '#' symbols are treated as comment marks in the parameter file. Characters from these characters to the end of the line are regarded as a comment.

-F parameter file name

[Examples]

Consider a parameter file **“75XTEST.PLV”** with the following contents.

```
;PARAMETER FILE
75XTEST1 -L75XTEST.LNK
-E75XTEST.ELV
```

Example The assembler is started with parameter file **“75XTEST.PLV”** specified.

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -F75XTEST.PLV
```

★ CHAPTER 9 LIBRARY CONVERTER

The library converter (LBCNV75X) outputs library files which can be input to version 5.00 or subsequent versions of Linker (LK75X) and Librarian (LB75X) when object program module format library files output by a version of Librarian (LB75X) in a RA75X Assembler Package which is earlier than version 5.00 are input.

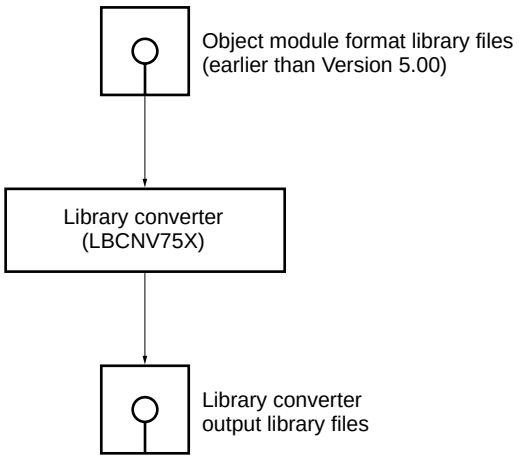
9.1 Library Converter Input/Output Files

Library converter (LBCNV75X) input/output files are the files shown in **Table 9-1 Library Converter Input/Output Files**.

Table 9-1 Library Converter Input/Output Files

Type of File		Default File Type
Input File	**Object program module format library files** Library files output by the Librarian (LB75X) in a RA75X Assembler Package version earlier than Version 5.00.	.LIB
Output File	**Library converter output library files** Library files which can be input to the Linker (LK75X) and Librarian (LB75X) in Version 5.00 or subsequent versions of the RA75X Assembly Package.	.CNV

Figure 9-1 Library Converter Input/Output Files



9.2 Library Converter Functions

- The library converter converts object program module format library files output by a version of Librarian (LB75X) in an Assembler Package which is earlier than Version 5.00 to library files which can be input to the Linker and Librarian in Version 5.00 or subsequent versions.
- If an error is discovered while the library converter is running, an error message is displayed on the console.
- The library converter carries out processing in accordance with the library converter options which are specified at startup time. See **9.4 Library Converter Options**, concerning the options in library converter.
- When processing is completed normally, the library converter outputs an end message and control returns to the OS.

Caution The object modules included in library files converted by the library converter cannot be debugged.

9.3 Starting the Library Converter

9.3.1 Starting the library converter

In order to start the library converter, input a command on the OS command line with the following format.

```
X>LBCNV75X[␣Option] ␣Input file [␣Option…]
```

- X indicates the current drive.
- The input file name is the name of the library file which you want to convert. The drive name and directory name, etc. can be included with the input file name.
- Please input 1 or more spaces between each option or input file (space or TAB) to demarcate between them.
- The default output destination is the originally specified file name with the file type changed to '.CNV,' created on the current path. This can be changed using the -O option.

9.3.2 Execution start and end message

(1) Execution Start Message

The execution start message is displayed on the console when the library converter starts.

```
75X Series  Library Converter Vx.xx [XX xxx XX]
  Copyright (C) NEC Corporation 1996
```

(2) Execution End Message

- If the library converter has not detected any fatal error, it outputs the following message to the console, then control is returned to the OS.

```
Library Conversion Complete, 0 error(s) found.
```

- If the library converter outputs an error message and aborts processing, the cause of the error message should be found in **13.6 Library Converter Error Messages,** and appropriate action taken.

9.3.3 Library converter error processing

The library converter outputs an Abort Error and aborts processing immediately if an error occurs which makes it impossible to continue program execution.

Error messages are output with the following format.

```
***␣Error No.␣Error message
```

9.3.4 Library converter end status

The library converter returns the following error status code to the OS when it ends execution and returns control to the OS.

End Conditions	End Status
End normally	0
Abort error	2

If the Library Converter is started from a batch file in MS-DOS (PC DOS, IBM DOS), it can be judged automatically using these values whether there were any link errors.

9.4 Library Converter Options

9.4.1 Types of library converter option

The following type of option is provided.

Table 9-2 Types of Library Converter Option

Item No.	Description Format	Function, Classification	Interpretation when Omitted
1	-O [file name]	Specifies a library file output by the Library Converter.	Generates 'Input file name.cnv' in the current path.

9.4.2 Specifying the library converter option

The library converter option is specified on the command line when starting the library converter. Please refer to **9.3 Starting the Library Converter** concerning specification of the library converter option on the command line.

9.4.3 Priority order of library converter options

If the same option is specified more than once on the command line of the library converter, the option specified last becomes valid.

9.4.4 Library converter option explanation

Details of the -O option are explained below.

-O output file name

(1) -O

Description Format	-O [File name]
Format when Omitted	'Input file name.CNV' is generated on the current path.

[Function]

- The -O option specifies the library file name output by the library converter.

[Use]

- Specify the -O option when desiring to change the library file name output by the library converter from the default file name.

[Explanation]

- If the -O option is not specified, a library file which has the same name as the input file name first specified, but with the file type changed to '.CNV,' is generated and output by the library converter. This is the same in the case when the file name is omitted and the -O option is specified.
- A logical device name (such as 'COM' or 'AUX') cannot be specified as a file name. If such a name is used to describe the file name, it will result in an error.
- The path name can be included in the file name and the path where the library file output by the library converter is to be generated can be specified. In this case, a file with the name of the input file, but with the file type changed to '.CNV' is generated in the specified path.

[Example]

Start the converter with the output file name **"TEST.CNV"** specified.

```
A:\NECTOOLS\SMP75X\RA75X>LBCNV75X 75XTEST.LIB -OTEST.CNV
```

→The library file **"TEST.CNV"** is output by the library converter.

★ CHAPTER 10 SETTING OPTIONS FROM THE PROJECT MANAGER

In this chapter, setting of options from the project manager is explained.
Please refer to the **Project Manager User's Manual** concerning the project manager.
The programs in which the options can be set from the project manager are as follows.

- Assembler
- Linker
- Object converter
- Structured assembler preprocessor

10.1 Setting Options from the Project Manager

The programs in which the options can be set from the project manager are as follows.

- Assembler
- Linker
- Object converter
- Structured assembler preprocessor (see **RA75X Structured Assembler Preprocessor User's Manual** concerning setting of options from this program's project manager.)

Cautions 1. In the check for the existence of an include file when creating the assembler's make file, clearing of commands and character strings only is performed and $if, $_if and similar conditions are disregarded.

Description Examples

```
$if SYM
$include (func1.inc)
$else
$include(func2.inc)
$endif
```

In the description examples, the $if, $else and $endif lines are disregarded, so both files are interpreted as include files. For that reason, regardless of whether or not there is an actual reference, if these files do not exist, it will result in an error during the build.

2. **When setting options from the project manager, the -C and -Y options are added in the assembler. At this time, the user cannot add the -C option.**
3. **When setting options from the project manager, the -Y and -F options are added in the linker (-F project file primary name.PLK). At this time, the user cannot add the -F option.**
4. **When setting options from the project manager, the -Y option is added in the object converter.**
5. **The structured assembler preprocessor always starts together with the assembler. When setting options from the project manager at this time, the -C and -Y options are added in the assembler. At this time, the user cannot add the -C option.**
6. **The project manager's [Option] → [Debug] menus are disregarded. For debugging information, set the -GS option in the [Set Structured Assembler Options] menu or the -GA option in the [Set Assembler Options] menu.**

10.1.1 Assembler

(1) Options Menu Items

Options Menu items set from the project manager are as follows.

Figure 10-1 Options Setting Menu (Assembler)

(2) Options Setting Dialog Box

Following is an explanation of the dialog box for setting assembler options.

From the menu items, select [Option] → [Set Assembler Options] menu, or select [Options] ([Source List] menu. Select the “Options” button in the dialog box to open the assembler options setting dialog box.

Figure 10-2 Options Setting Dialog Box (If the source file has not been selected) (Assembler)

Sets option for all source files

Figure 10-3 Options Setting Dialog Box (If the source file has been selected) (Assembler)

Sets option for the selected source file

Table 10-1 Option Setting Dialog Box Functions (Assembler)

Button	Function
“OK” button	If a source file is not selected, option setting of all source files (source files for which individual options are not set) is done, then the dialog box closes. If a source file is selected, option setting is done for the selected source file, then the dialog box closes. If the return key is pressed when the focus is on the option input combo box, it is regarded as if the “OK” button was pressed.
“Source option delete” button	This becomes valid when a source file is selected. If this button is selected, the individual options set in source file units are cleared. Overall options become valid for a source file with individual options cleared.
“Cancel” button	This button cancels the settings in this dialog box and closes the dialog box. If the ESC key is pressed, it is regarded as if the “Cancel” button was pressed.
“Help” button	This opens the help file for this dialog box.
Option character string display area (*1)	This displays option character strings which are currently set. It also displays option character strings which cannot be reduced to one line. Option character strings input in the option input combo box are displayed here in real time.
Option input combo box (*2)	Input option character strings here. The number of characters [Note] can be up to 127 characters. Device file specification is done in the project manager settings, so it is not necessary here.
[↓]	A history of past inputs drops down. Up to 10 past inputs are stored. (If there is no source, there are 10 options and also 10 options for special sources. If there is a character string in the history which is the same as the option character string, the previously input history is cleared and the newly input character string is added.

Note Includes the name of the source file set automatically by the project manager and the number of option characters.

Caution When setting options, no check is made for errors in the option description. Errors in the option description become errors during build.

(3) Source File Option Setting Dialog Box

If [Options] → [Source List] menu is selected from the menu items, this dialog box opens (see **Figure 10-4**).

Figure 10-4. [Source List] Dialog Box

If a source file is selected and the [Options] button pressed, the assembler options setting dialog box opens (see **Figure 10-2**). If options are input in the options input combo box (*2) and the “OK” button pressed, the source file options setting dialog box opens (see **Figure 10-5**).

Figure 10-5 Source File Options Setting Dialog Box

Table 10-2 Source File Options Setting Dialog Box Functions

Button	Function
“OK” button	Sets options for the selected source file and closes the dialog box.
“Set overall options” button	Sets options for all source files (source files for which individual options are not set).
“Cancel” button	This cancels the settings in this dialog box and closes it.

10.1.2 Linker

(1) Options Menu Items

Options menu items set from the project manager are as follows.

Figure 10-6 Options Setting Menu (Linker)

(2) Options Setting Dialog Box

Following is an explanation of the set linker options dialog box.

From the menu items, select [Options] → [Set Linker Options] menu to open the set linker options dialog box.

Figure 10-7 Options Setting Dialog Box (Linker)

Table 10-3 Options Setting Dialog Box Functions (Linker)

Button	Function
"OK" button	Sets the options and closes the dialog box. If the return key is pressed when the focus is on the options input combo box, it is regarded as if the "OK" button was pressed.
"Cancel" button	This cancels the settings in this dialog box and closes it. If the ESC key is pressed, it is regarded as if the "Cancel" button was pressed.
"Help" button	This opens the help files for this dialog box.
Option character string display area (*1)	Displays the character strings for the currently set options. Option character strings which cannot be fit on one line are also displayed. Option character strings input in the options input combo box are displayed here in real time.
Option input combo box (*2)	Input option character strings here. The number of characters [Note] can be up to 127 characters. Device file specification is done in the project manager settings, so it is not necessary here.
[↓]	A history of past inputs drops down. Up to 10 past inputs are stored. If there is a character string in the history which is the same as the option character string, the previously input history is cleared and the newly input character string is added.

Note Includes the name of the source file set automatically by the project manager and the number of option characters.

Cautions 1. The "Source Options Delete" button is displayed in reverse video and cannot be selected.
2. When setting options, no check is made for errors in the option description. Errors in the option description become errors during build.

10.1.3 Object converter

(1) Options Menu Items

Options menu items set from the project manager are shown below.

Figure 10-8 Options Setting Menu (Object Converter)

(2) Options Setting Dialog Box

Following is an explanation of the set object converter options dialog box.

From the Menu items, select [Options] → [Set Object Converter Options] menu to open the set object converter options dialog box.

Figure 10-9 Options Setting Dialog Box (Object Converter)

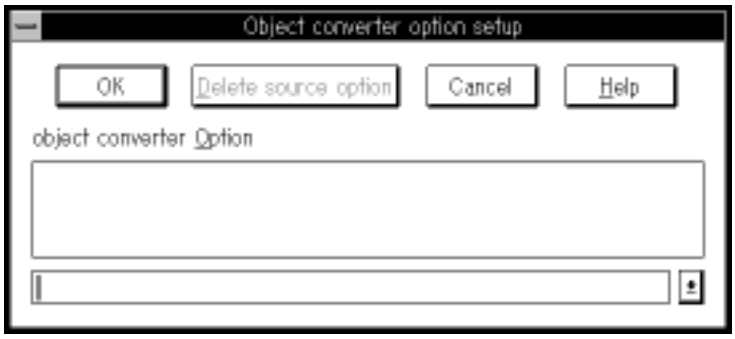


Table 10-4 Options Setting Dialog Box Functions (Object Converter)

Button	Function
“OK” button	Sets the options and closes the dialog box. If the return key is pressed when the focus is on the options input combo box, it is regarded as if the “OK” button was pressed.
“Cancel” button	This cancels the settings in this dialog box and closes it. If the ESC key is pressed, it is regarded as if the “Cancel” button was pressed.
“Help” button	This opens the help files for this dialog box.
Option character string display area (*1)	Displays the character strings for the currently set options. Option character strings which cannot be fit on one line are also displayed. Option character strings input in the options input combo box are displayed here in real time.
Option input combo box (*2)	Input option character strings here. The number of characters [Note] can be up to 127 characters. Device file specification is done in the project manager settings, so it is not necessary here.
[↓]	A history of past inputs drops down. Up to 10 past inputs are stored. If there is a character string in the history which is the same as the option character string, the previously input history is cleared and the newly input character string is added.

Note Includes the name of the source file set automatically by the project manager and the number of option characters.

Cautions 1. The “Source Options Delete” button is displayed in reverse video and cannot be selected.

2. When setting options, no check is made for errors in the option description. Errors in the option description become errors during build.

[MEMO]

CHAPTER 11. PROGRAM OUTPUT LISTS

This chapter shows the format, etc., of the various lists output by each program in the assembler package.

11.1 Assembler Output Lists

The assembler outputs the following lists.

- Assembly list
- Symbol table list
- Cross-reference list
- Error list

The assembly list, symbol table list and cross-reference list are output to the assembly list file. The error list is output to the error list file.

11.1.1 Assembly list

The assembly results are output in this list together with any error messages (only if there are errors).

★ **[Output Format]**

```
75X SERIES ASSEMBLER Vx.xx                                          <1>   PAGE: <2>
  **                          ... ... ... ...  <3>  ... ... ... ...                **

COMMAND       : cccc ..... <5>
COMMAND FILE: <4>

IC   LINE  ADDR R  OBJECT    ASSEMBLER SOURCE              ;ORIGINAL SOURCE Note 1
     0001  0000  R  12345678  lebel:  mnem    operand      ;lebel:  mnem    operand
   +0001  0000  R  12345678  lebel:  mnem    operand
     0002  0000     12345678  lebel:  mnem    operand, operand
                                                          ;lebel:  mnem  operand,operand
     0003                                                 ;$INCLUDE=A1.INC
1    0001                                                 ;$INCLUDE=A2.INC
2  +0001                                                  ;$INCLUDE=A3.INC
3    0001
a.asm(1)  :   #001 SYNTAX ERROR
4  +0001
4    0001
ffnnnn  hhhh  a   xxxxxxxx  sssssssssss……..sssssssssss;uuuu...uuuu
 ⋮  ⋮     ⋮   ⋮       ⋮                              ⋮          ⋮
<6><7>   <8> <9>    <10>                            <11>       <12>

 SOURCE FILE: A.ASM
INCLUDE FILE : A1.INC(1)
               A2.INC(2)
               A3.INC(3)
MACRO FILE   : EXMAC.M(4)
TARGET CHIP : UPD75000
DEVICE FILE   : VX.XX Note 2
STACK SIZE   =   0000H

ASSEMBLY COMPLETE, NO ERROR FOUND
```

Notes 1. The source input to the structured assembler preprocessor is output.

2. When a device file is used, that file's version is output.

★ **[Descriptions of Output Items]**

No.	Description
<1>	System date
<2>	Output list page number in decimal notation
<3>	Title (value specified by TITLE control instruction) (Blank if there is no TITLE control instruction specification)
<4>	Parameter file command image (Blank if there is no parameter file input specification)
<5>	Command image
<6>	File number
<7>	Displays the line number of a source described by the user in 4 digit decimal notation.
<8>	Displays the location counter value in 4-digit hexadecimal.
<9>	Operand Attributes E ... Reference to external reference symbol. R ... Reference to a relocatable symbol △ ... Absolute value
<10>	Object code is displayed in hexadecimal.
<11>	Source statement which includes the Assembler generation line. (The source (32 column) input to the structured assembler preprocessor is output.)
<12>	User described source statement. (If ⑪ exceeds the heading of the line, it is output after carriage return.)

11.1.2 Symbol table list

This name, attributes and symbol values of symbols defined in the source program are output in this list.

[Output Format]

```
75X SERIES ASSEMBLER Vx.xx                                          <1>  PAGE: <2>
  **                          ... ... ... ...  <3>  ... ... ... ...                **
  SYMBOL TABLE LIST

OFFSET  TYPE      SYMBOL              OFFSET  TYPE      SYMBOL
xxxxH   tttttt    rrr  SS.....  ...  xxxxH   tttttt    rrr  SS.....  ...
   :      :       :    :                 :       :       :    :
  <4>    <5>     <6>  <7>               <4>     <5>     <6>  <7>
```

[Descriptions of Output Items]

No.	Description
<1>	System date
<2>	Output list page number in decimal notation
<3>	Title (value specified by TITLE control instruction) (Blank if there is not TITLE control instruction specification)
<4>	Symbol value shown as 4 hexadecimal digits (In case of segment name, segment size shown as 4 hexadecimal digits)
<5>	Symbol attribute { BIT .... Bit symbol CODE .... Code symbol DATA .... Data symbol NUMBER .... Constant symbol PBIT .... Port bit symbol MACRO .... Macro name
<6>	Symbol reference format { EXT ... External reference symbol name PUB ... External definition symbol name △ ... Local symbol name
<7>	Symbol name

11.1.3 Cross-reference list

The locations (line numbers) at which symbols are referenced in the source program are output in this list.

[Output Format]

```
75X SERIES ASSEMBLER Vx.xx                                    <1>  PAGE: <2>
  **                          ... ... ... ...  <3>  ... ... ... ...          **
  CROSS REFERENCE LIST

SYMBOL   TYPE   VALUE                                  ATTRIBUTES XREF LIST

SS.....  TT..... xxxx   A   RRR AA                       nn...nn...nn.....
  :        :       :     :    :   :                        :
 <4>      <5>     <6>   <7>  <8> <9>                      <10>
```

[Descriptions of Output Items]

No.	Description
<1>	System date
<2>	Output list page number in decimal notation
<3>	Title (value specified by TITLE control instruction) (Blank if there is no TITLE control instruction)
<4>	Symbol name
<5>	Symbol attributes { BIT .... Bit symbol CODE .... Code symbol DATA .... Data symbol NUMBER .... Constant symbol PBIT .... Port bit symbol MACRO .... Macro name
<6>	Symbol value shown as 4 hexadecimal digits
<7>	Symbol relocation attributes { A ... Absolute symbol R ... Relocatable symbol
<8>	Symbol reference format { EXT ... External reference symbol name PUB ... External definition symbol name △ ... Local symbol name
<9>	Additional symbol information { Label symbol .. Definition segment name .. (Blank for absolute segment) Name symbol .. Definition segment name .. (Blank for constant symbol) Segment name .. Link attribute and location attribute Others .. Blank
<10>	Definition/reference statement number in decimal notation "#" on the left of the number indicates a definition statement number

11.1.4 Error list

Only error line and error messages resulting from assembly are output in the error list.

[Output Format]

```
a.asm(1):#001 SYNTAX ERROR
 <1> <2>  <3> <4>
```

[Descriptions of Output Items]

No.	Description
<1>	Input file name
<2>	Input file line number
<3>	Error message number shown as 3 decimal digits (See 13.1 for the meaning of the message numbers)
<4>	Error message

11.2 Linker Output List

The file name creates the following lists in the link list file.

Link List File

- Linker option list
- Input-output module list
- Segment link map list
- Branch table map list
- Public symbol list
- Symbol table list

11.2.1 Linker option list

The image of the options input from the console is output in this list.

[Output Format]

```
75X SERIES LINKER Vx.xx .                                    <1>  PAGE: <2>

COMMAND : cccc ……..……..……..…….. <4>
COMMAND FILE :  <3>
```

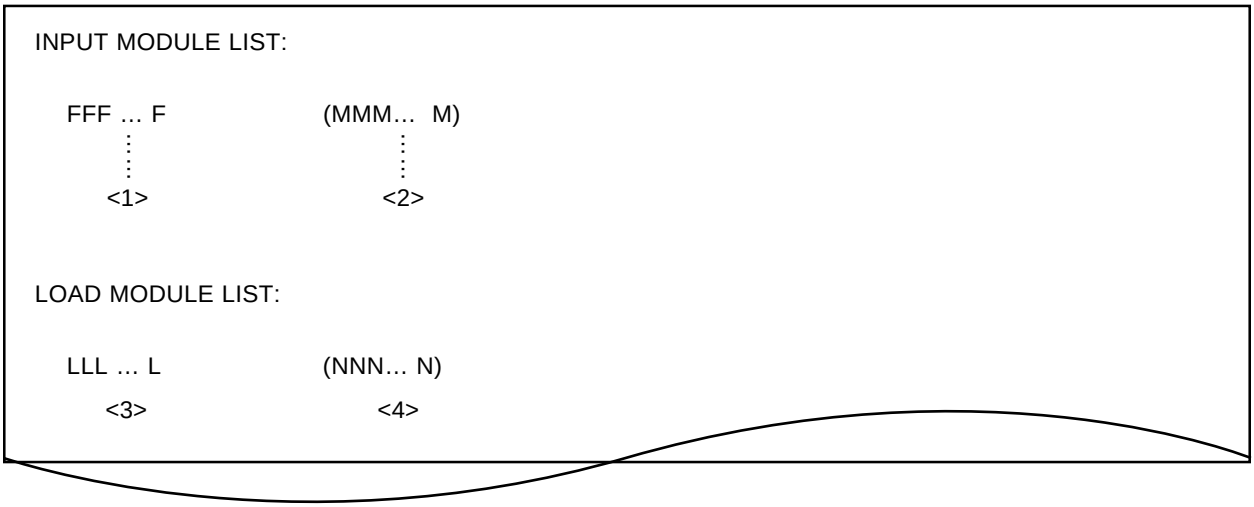
[Descriptions of Output Items]

No.	Description
<1>	System date
<2>	Output list page number in decimal notation
<3>	Parameter file command image (Blank if there is no parameter file input specification)
<4>	Command image

11.2.2 Input - output module list

The input/output file names and module names are output in this list.

[Output Format]



[Descriptions of Output Items]

No.	Description
<1>	Input file name (object module file, library file)
<2>	Input object module name
<3>	Output file name (load module file)
<4>	Output module name (value specified by -M option) (If there is no -M option, primary name of first input file name)

11.2.3 Segment link map list

Information on located segments is output in this list in address order separately for the program memory and data memory.

[Output Format]

```
SEGMENT LINK MAP FOR    <1>          ( <2> )

MAP OF ROM AREA:
             BASE   LENGTH   MODULE NAME    SEGMENT NAME (TYPE)
             -------- ------------ ---------------------- ------------------------------------------
   XXX...    BBBB   LLLL         MMM...                NNN...              (RRR... )
     ⋮         ⋮      ⋮             ⋮                    ⋮                   ⋮
    <3>       <4>    <5>           <6>                  <7>                 <8>

   MAP OF RAM AREA:
             TYPE                BASE    LENGTH   MODULE NAME    SEGMENT NAME
             --------            --------  ------------ ---------------------- ------------------------
   XXX       TTTT   XXX...       BBBB    LLLL         MMM...                 NNN...
     ⋮         ⋮       ⋮          ⋮       ⋮              ⋮                     ⋮
    <3>       <9>     <3>        <4>     <5>            <6>                   <7>
```

[Descriptions of Output Items]

No.	Description
<1>	Output file name (load module file)
<2>	Output module name (value specified by -M option) (If there is no -M option, primary name of output file name)
<3>	Segment overlap indication (‘shown as ‘** OVERLAP **’) (Blank if there is no segment overlap)
<4>	Segment start address (shown as 4 hexadecimal digits in ascending address order)
<5>	Segment size (4 hexadecimal digits) (End address - Start address + 1)
<6>	Module name (Segment definition module name)
<7>	Segment name (left-justified) (‘** GAP **’ is shown for a free area) (Blank in case of default absolute segment or VENT area)
<8>	Segment relocation attributes { ‘ABSOLUTE’ absolute segment ‘IENT’ ‘SENT’ ‘INBLOCK’, ‘INBLOCKA’ ‘XBLOCK’, XBLOCKA’ ‘INBLOCK PAGE’, ‘INBLOCKA PAGE’ ‘SENT PAGE’ ‘XBLOCK PAGE’, XBLOCKA PAGE’
<9>	Segment attributes (left-justified) { ‘CODE’ ... Code segment ‘DATA’ ... Data segment ‘STACK’ ... Stack segment

11.2.4 Branch table map list

Information on branch tables generated in order to branch to another block is output in this list.

[Output Format]

```
BRANCH TABLE MAP FOR   <1>
BLOCK   LOCATED          REFERENCE   EXPRESSION    REFERENCE   REFERENCE

NO      ADDRESS                                    ADDRESS     ADDRESS
----    --------------   ----------------------------------   ----------
BB      XXXX       BR    !NNN...       + AAA...     RRRR        SSS...
 ⋮       ⋮         ⋮       ⋮             ⋮           ⋮           ⋮
<2>     <3>       <4>     <5>           <6>         <7>         <8>
```

[Descriptions of Output Items]

No.	Description
<1>	Output file name (load module file)
<2>	Branch table location bank value (2 hexadecimal digits)
<3>	Address in block (4 hexadecimal digits)
<4>	Target device ROM size Up to 16K: ‘BR’ Over 16K: ‘BRA’
<5>	Reference symbol name (left- justified) (Definition segment name when an intra-module local symbol is referenced. Reference address value in case of absolute address reference)
<6>	Reference symbol modification value (4 hexadecimal digits) (Blank in case of absolute address reference)
<7>	Reference start address (4 hexadecimal digits)
<8>	Reference start segment name (left-justified) (‘.....’ in case of absolute address reference)

11.2.5 Public symbol list, symbol table list

Information on public symbols and local symbols is output in these lists.

[Output Format]

```
PUBLIC SYMBOL LIST FOR  <1>

TYPE    VALUE    MODULE     SYMBOL NAME
-------  ---------  ------------  ----------------------
TTTT    XXX...   MMM...     NNN...
  ⋮       ⋮         ⋮          ⋮
<2>     <3>       <4>        <5>

SYMBOL LIST FOR ①

TYPE    VALUE    ATTRIBUTE NAME
-------  ---------  ------------  ---------
TTTT    XXX...   AAA...     NNN...
  ⋮       ⋮         ⋮          ⋮
 <2>     <3>       <4>        <5>
```

[Descriptions of Output Items]

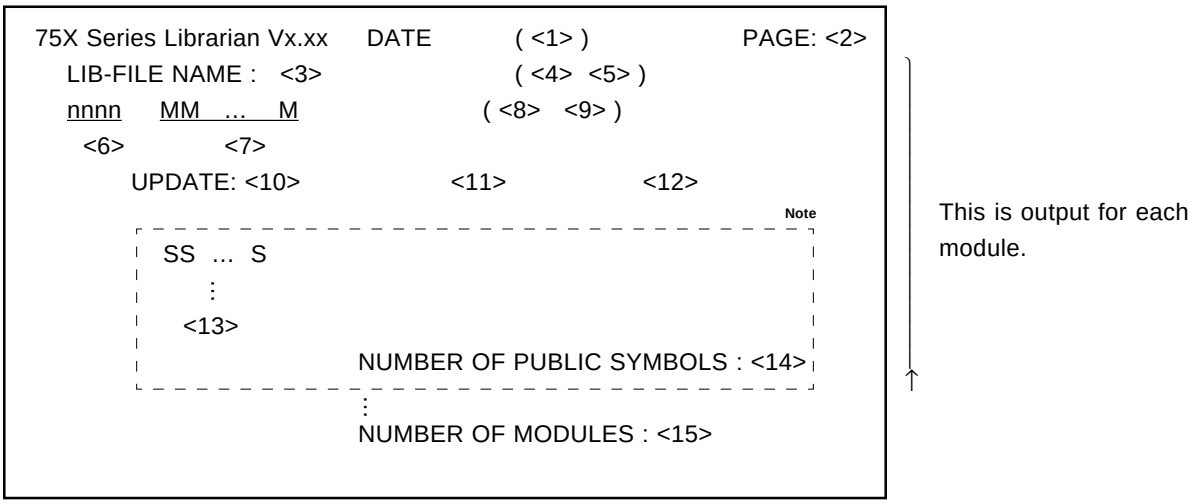
No.	Description
<1>	Output file name (load module file)
<2>	Symbol attribute names (left-justified): 'BIT' ... Bit symbol; 'CODE' ... Code symbol; 'DATA' ... Data symbol; 'NUMBER' ... Constant symbol; 'PBIT' ... Port bit symbol; 'STACK' ... Stack symbol
<3>	Symbol value (4 hexadecimal digits)
<4>	Name of defined symbol (left-justified)
<5>	Name of symbol defined in module (left-justified)
<6>	Symbol type (left- justified): 'MODULE' .. In case of module name; 'PUBLIC' .... In case of symbol; 'SYMBOL' ... In case of intra-module symbol

11.3 Librarian Output List

The librarian outputs a list of library file information by means of the LIST subcommand. See **7.4 (5) LIST** for the list output destination.

11.3.1 Library file information list

[Output Format]



Note The area enclosed by a broken line is only printed when there is a PUBLICS specification.

[Descriptions of Output Items]

No.	Description
<1>	System date
<2>	Output list page number (decimal notation)
<3>	Library file name
<4>	Library file creation date
<5>	Library file update date
<6>	Module serial number (decimal notation)
<7>	Module name
<8>	Module creation date
<9>	Module update date
<10>	Number of module updates (decimal notation)
<11>	Name of the software which created module
<12>	Assembly target product
<13>	PUBLIC symbol name
<14>	Total number of PUBLIC symbols defined in the module
<15>	Total number of modules in library file

11.4 List Converter Output List

The list converter outputs the absolute assembly list.

11.4.1 Absolute assembly list

In this list, the actual values determined by the linker are incorporated in the assembly list part (excluding the symbol list and cross-reference list) in the assembly list file output by the assembler.

[Output Format]

```
75X SERIES ASSEMBLER Vx.xx                                          <1>  PAGE: <2>
  **                          ... ... ... ...  <3>  ... ... ... ...                **
COMMAND : cccc ..... <5>
COMMAND FILE :  <4>

  STNO  ADRS  R  OBJECT  IC   MAC  SOURCE  STATEMENT
  nnnn  hhhh  a  xxxxxxxx = i  +mm  sss  .....
    :     :     :     :     :     :              :
    :     :     :     :     :     :              :
  <6>   <7>  <8>  <9>  <10> <11>            <12>
-------------------------------------------------------------------------------------------------------------------
TARGET CHIP:      <14>
STACK SIZE=xxxx   <13>
ASSEMBLY COMPLETE,                ERROR       FOUND
```

[Descriptions of Output Items]

No.	Description
<1>	System date
<2>	Output list page number in decimal notation
<3>	Title (value specified by TITLE control instruction) (Blank if there is no TITLE control instruction specification)
<4>	Parameter file command image (Blank if there is no parameter file input specification)
<5>	Command image
<6>	Source statement number shown as 4 decimal digits
<7>	Location counter value shown as 4 hexadecimal digits
<8>	Operand attributes E.... Reference of external reference symbol (symbol declared by the EXTRN pseudo-instruction) R ... Reference of relocatable symbols in the same module △... Absolute value
<9>	Object code in hexadecimal notation In case of a symbol definition pseudo-instruction, the operand evaluation value is shown as 4 hexadecimal digits.
<10>	Include nesting level (1) (Blank if not nested)
<11>	Always blank (for function expansion)
<12>	Source statement
<13>	Stack area size shown as 4 hexadecimal digits
<14>	Assembly target product name

Items <7> and <9> are subject to amendment.

[MEMO]

CHAPTER 12. EFFECTIVE USE OF THE ASSEMBLER PACKAGE

This chapter suggests some methods of making efficient use of the assembler package.

12.1 How to Utilize Parameter File

It is convenient to use a parameter file when starting the assembler, linker or object converter.

A parameter file is one in which specifications of input files, options, etc., needed when starting a program are recorded beforehand using an editor.

Use of a parameter file is particularly recommended when starting the linker, which uses a large number of input files.

The contents of options specified in a parameter file can be added to or changed in the command line when starting the program.

Example 1. A parameter file 'LINK.PLK' is created with the editor.

- Contents of LINK.PLK

```
75XTEST1.REL 75XTEST2.REL
-075X.LNK -PSAMP.MAP -KM
```

- The linker is started with parameter file 'LINK.PLK' specified.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X -FLINK.PLK
75X Series Linker VX.XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1985

LINK COMPLETE, NO ERROR FOUND
```

Example 2. The options specified in the parameter file are changed and added to in the command line.

```
A:\NECTOOLS\SMP75X\RA75X>LK75X -FLINK.PLK -PTEST.LNK -SQ
```

12.2 Use of the List Converter

In the assembly list output by the relocatable assembler, temporary values are used for addresses and object code which were not determined during assembly.

In order to perform debugging with the in-circuit emulator, etc., using this assembly list, the actual values of addresses and object code must be calculated by referring to the segment map list, etc., output by the linker. (This is not so necessary if symbolic debugging is performed effectively using the OC75X).

This problem is common to relocatable assemblers .

The list converter is provided to solve this problem. This list converter creates an absolute assembly list, which enables debugging to be performed more efficiently.

If the absolute assembly list is to be output to the printer, specifying the printer as the output destination in the list converter (LCNV75X) enables the list to be output directly to the printer.

Example To output the absolute assembly list to the printer

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X 75XTEST1 -L75XTEST.LNX -OPRN

List Conversion Program for RA75X VX. XX [XX Xxx XX]
   Copyright (C) NEC Corporation 1986, 1997

Pass 1: start .....................
Pass 2: start .........
Conversion complete
```

12.3 Finding Error Lines

To find error lines in the assembly list in order to eliminate assembly errors, the error line back no, is used. When errors are detected as a result of assembly, the assembler outputs the following message.

Example

```
ASSEMBLY COMPLETE, 5 ERRORS FOUND ( 10)
```

Number of errors — Line number of last error line

In this example, the l0th line of the assembly list is the last error line. In the l0th line of the assembly list, the line number of the preceding error line is shown together with the error message. The error line back number, in the assembly list is indicated as shown below.

```
*** ERROR #xxx, STNO #1111    (0000)          , ERROR MESSAGE ..... <1>
*** ERROR #xxx, STNO #mmmm    (1111)          , ERROR MESSAGE ..... <2>
*** ERROR #xxx, STNO #nnnn    (mmmm)          , ERROR MESSAGE ..... <3>
ASSEMBLY COMPLETE, t t t t ERROR S FOUND (nnnn)                ..... <4>
```

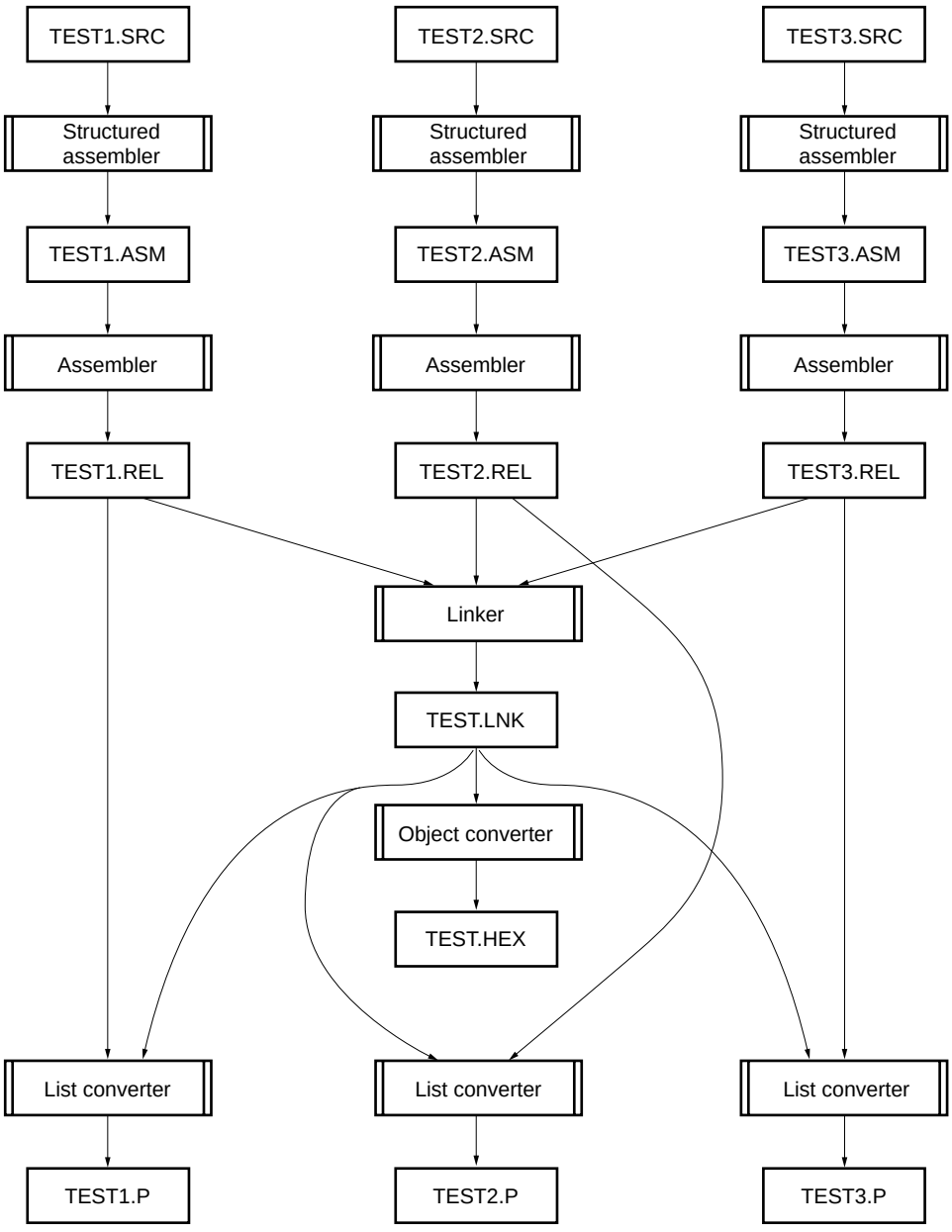
<1> First error line
<2> Next error line
<3> Next error line
<4> Last error line

Using the error line back number, indication enables lines in which errors are generated to be added in the order nnnn → mmmm → llll from the end of the list file.

12.4 Example of Use of Batch File

In the assembly package, an error status code is returned according to the error level. If an error status code is used, assembly can be performed efficiently using a batch file. An example is given here of performance of the following processing.

(1) Batch file (BAT.BAT) processing flow



(2) Sample program

BAT.BAT

```
ECHO OFF
SET LEVEL=0
ST75X TEST1.SRC
RA75X TEST1.ASM -C106
IF ERRORLEVEL 1 SET LEVEL=1
ST75X TEST2.SRC
RA75X TEST2.ASM -C106
IF ERRORLEVEL 1 SET LEVEL=1
ST75X TEST3.SRC
RA75X TEST3.ASM -C106
IF ERRORLEVEL 1 SET LEVEL=1
CLS
IF %LEVEL%=1 ECHO       Error generated during assembly
IF %LEVEL%=1 GOTO END
SET LEVEL=0
CLS
LK75X TEST1 TEST2 TEST3-0TEST
IF ERRORLEVEL 1 ECHO   Error generated during linkage
IF %LEVEL%=1 GOTO END
CLS
OC75X TEST
IF ERRORLEVEL 1 ECHO   Error generated during object conversion
IF %LEVEL%=1 GOTO END
SET LEVEL=0
LCNV75X -LTEST.LNK -ATEST1
IF ERRORLEVEL 1 SET LEVEL=1
LCNV75X -LTEST.LNK -ATEST2
IF ERRORLEVEL 1 SET LEVEL=1
LCNV75X -LTEST.LNK -ATEST3
IF ERRORLEVEL 1 SET LEVEL=1
CLS
IF %LEVEL%=1 ECHO       Error generated during list conversion
IF %LEVEL%=1 GOTO END
ECHO            No errors
:END
```

(3) Example of executed result

(a) Normal termination

```
No error
```

(b) When there is an assembly error

```
Error in assembly
```

The error should be corrected with reference to the assembly print file, and assembly performed again.

CHAPTER 13. ERROR MESSAGES

This chapter describes the error messages output by each program in the assembler package, the cause of each error, action to be taken by the user, and so forth.

13.1 Assembler's Error Messages

Format: *** ERROR[Note] error number error message
PROGRAM ABORTED
This error cause program execution to be aborted.

Note This part is not output in error messages A001 to A021.

Error No. A001	Message	Missing input file
	Cause	Input file has not been specified in start line.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A002	Message	Too many input files
	Cause	Specified number of input files exceeds limit.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A003	Message	Unrecognized string 'specified string'
	Cause	Specified string cannot be interpreted.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A004	Message	Illegal file name 'file name'
	Cause	Type or length of characters in file name is illegal
	Program processing	Checks start line syntax , then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A005	Message	Illegal file specification 'file name'
	Cause	File name format is incorrect.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A006	Message	File not found 'file name'
	Cause	Specified file does not exist.
	Program	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A007	Message	Input file specification overlapped 'file name'
	Cause	Overlapping input file specification
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A008	Message	File specification conflicted 'file name'
	Cause	The input or output file is specified overlaps.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A009	Message	Unable to make file 'file name'
	Cause	Specified output file cannot be created.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check remaining disk capacity, etc.
Error No. A010	Message	Directory not found 'file name'
	Cause	Nonexistent drive or directory is included in output file name.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.

Assembler's Error Message (cont'd)

Error No. A011	Message	Illegal path 'option'
	Cause	Item other than path name is specified in option in which path name should be specified as parameter.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name and option name, then re-execute.
Error No. A012	Message	Missing parameter 'option'
	Cause	Necessary parameter has not been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A013	Message	Parameter not needed 'option'
	Cause	Unnecessary parameters has been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A014	Message	Out of range 'option'
	Cause	Number specified as parameter exceeds permitted range.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A015	Message	Parameter is too long 'option'
	Cause	Parameter is too long.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A016	Message	Illegal parameter 'option'
	Cause	Parameter syntax is incorrect.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A017	Message	Too many parameters 'option'
	Cause	Number of parameters exceeds limit.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A018	Message	Option is not recognized 'option'
	Cause	Incorrect option has been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute,
Error No. A019	Message	Parameter file nested
	Cause	-F option is included in parameter file.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct parameter file, then re-execute.
Error No. A020	Message	Parameter file read error 'file name'
	Cause	Parameter file read has failed.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check the disk status.
Error No. A021	Message	Memory allocation failed
	Cause	Memory block acquisition has failed.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check mounted memory capacity and whether there are resident programs, etc.
Error No. A099	Message	CHIP OR MODE IS NOT SELECTED
	Cause	-C option or -M option has not been specified.
	Program processing	Program execution is halted.
	User action	Specify -C option or -M option.

Assembler's Error Message (cont'd)

Error No. A100	Message	NO OVERLAY FILE - overlay file name
	Cause	Overlay file has not been found.
	Program processing	Program execution is halted.
	User action	Check that overlay file is in executable format and is in same directory and same drive.
Error No. A101	Message	ILLEGAL OVERLAY FILE VERSION - overlay file name
	Cause	Overlay file version is incorrect.
	Program processing	Program execution is halted.
	User action	Check that overlay file version is correct.
Error No. A102	Message	MACRO MEMORY OVERFLOW, CANNOT DEFINE MACRO 'MACRO_name'
	Cause	Insufficient memory to record macro
	Program processing	Program execution is halted.
	User action	Partition the program and re-execute
Error No. A103	Message	PARAMETER REPLACEMENT OVERFLOW
	Cause	More than 128 formal parameter substitutions.
	Program processing	Program execution is halted.
	User action	Check that there is no more than 128 formal parameter substitutions.
Error No. A901	Message	WORKING TABLE SPACE EXHAUSTED
	Cause	Working area in memory is insufficient.
	Program processing	Program execution is halted.
	User action	Reduce number of segments, and PUBLIC & EXTRN symbols or reduce number of unresolved branch instructions.
Error No. A902	Message	BRANCH TABLE OVERFLOW
	Cause	BR pseudo-instruction optimization work area is full.
	Program processing	Program execution is halted.
	User action	Reduce number of BR pseudo-instructions.
Error No. A903	Message	OPEN ERROR
	Cause	Input/output file cannot be opened.
	Program processing	Program execution is halted.
	User action	Check status of specified disk.
Error No. A904	Message	CLOSE ERROR
	Cause	Input/output file cannot be closed.
	Program processing	Program execution is halted.
	User action	Check status of specified disk.
Error No. A905	Message	DUPLICATE INCLUDE OR MACRO FILE 'file_name'
	Cause	Include file name specified in source program, or external macro file name, is same as file name specified in start line.
	Program processing	Program execution is halted.
	User action	Check file name.
Error No. A909	Message	DISK FULL
	Cause	Capacity of disk specified for output is insufficient.
	Program processing	Program execution is halted.
	User action	Delete unnecessary files, or use a new disk.
Error No. A999	Message	PROGRAM ERROR
	Cause	Memory contents have been overwritten by another program.
	Program processing	Program execution is halted.
	User action	Check the multi-user method of use.

Format: *** ERROR #error number, STNO # ~ (~), error message
These errors are printed in the assembly list.

Assembler's Error Message

#1	Message	SYNTAX ERROR
	Cause	Error in statement format.
	Program processing	Mnemonic operand value is regarded as 0 and processing is continued. With pseudo-instruction, line is regarded as invalid and processing continues.
	User action	Amend to correct description format.
#2	Message	FORMAT ERROR
	Cause	Number of operands is insufficient.
	Program processing	Mnemonic operand value is regarded as 0 and processing is continued. With pseudo-instruction, line is regarded as invalid and processing continues.
	User action	Amend to correct description format.
#3	Message	ILLEGAL PARAMETER TO CONTROL
	Cause	Error in specified parameter option description.
	Program processing	The option is ignored. In case of INCLUDE option, however, fatal error results and program execution is aborted.
	User action	Correct this parameter description.
#4	Message	CONTROL COMMAND IS NOT RECOGNIZED
	Cause	Error in written option name.
	Program processing	The option is ignored.
	User action	Amend to correct option name.
#5	Message	MISPLACED PRIMARY CONTROL
	Cause	Specified option can only be written in start command line or at start of source program.
	Program processing	This option is ignored.
	User action	Amend to correct description location.
#6	Message	NO TITLE FOR TITLE CONTROL
	Cause	No parameter specification in TITLE option.
	Program processing	This option is ignored.
	User action	Specify the parameter.
#7	Message	PAGEWIDTH WITHIN THE LIMIT FROM 72 TO 132
	Cause	PAGEWIDTH option parameter value is not in range 72 to 132.
	Program processing	This option is ignored (default value 132 is used).
	User action	Correct the parameter.
#8	Message	PAGELENGTH MORE THAN 20
	Cause	PAGELENGTH option parameter value is 20 or less.
	Program processing	This option is ignored (default value 66 is used).
	User action	Correct the parameter.
#9	Message	INCLUDE FILE NEST OVERFLOW
	Cause	INCLUDE control instruction nesting in include file is too deep.
	Program processing	INCLUDE option in which error was detected is ignored.
	User action	Reduce include file nesting (to 1 level).
#10	Message	NO PARAMETER TO CONTROL
	Cause	There is no parameter specification for option requiring parameter.
	Program processing	This option is ignored.
	User action	Specify the parameter.

Assembler's Error Message (cont'd)

#11	Message	NO PARAMETER IS ALLOWED TO THIS CONTROL
	Cause	Parameter has been specified for option which does not require parameter.
	Program processing	Specified parameter is ignored.
	User action	Delete the parameter.
#14	Message	LABEL WITHOUT COLON IS REQUIRED
	Cause	There is no name or segment name in symbol field.
	Program processing	This statement is ignored as illegal statement.
	User action	Write name or segment name in symbol field.
#15	Message	LABEL WITH COLON IS NOT ALLOWED
	Cause	Delimiter between symbol field and mnemonic field is space, not colon ':'
	Program processing	This statement is ignored as illegal statement.
	User action	Replace unnecessary colon ':' delimiter with space.
#16	Message	LABEL WITHOUT COLON IS NOT ALLOWED
	Cause	Colon ':' is required as delimiter between symbol field and mnemonic field.
	Program processing	Processing is continued with label undefined.
	User action	Write colon ':'
#17	Message	NAME IS NOT ALLOWED TO THIS STATEMENT
	Cause	Name cannot be written in symbol field in this statement.
	Program processing	Processing is continued with name undefined.
	User action	Define name in appropriate place other than this statement, and re-assemble.
#18	Message	UNDEFINED SYMBOL
	Cause	Undefined symbol has been referenced.
	Program processing	Processing is continued with 0 as undefined symbol value.
	User action	Amend to correct symbol or define symbol.
#19	Message	ABSOLUTE EXPRESSION EXPECTED
	Cause	Relocatable expression cannot be written as operand of this statement.
	Program processing	With SET pseudo-instruction, symbol is not defined. In other cases, processing is performed with expression regarded as absolute expression.
	User action	Use absolute expression as operand.
#20	Message	NUMERIC EXPRESSION IS REQUIRED
	Cause	Inappropriate item has been written as operator term.
	Program processing	With symbol definition pseudo-instruction, symbol is not defined. In other cases, processing is continued with 0 as operand value.
	User action	Use appropriate item as operator term.
#21	Message	EXPRESSION STACK OVERFLOW
	Cause	Expression description is too complex.
	Program processing	Processing is continued with 0 as expression value.
	User action	Rewrite expression in simpler form.
#22	Message	ATTRIBUTE OF EXPRESSION MISMATCHED
	Cause	Impermissible symbol attribute is used in expression symbol attribute combination.
	Program processing	With symbol definition pseudo-instruction, symbol is not defined. In other cases, processing is continued with 0 as expression value.
	User action	Amend to correct description format.
#23	Message	SEGMENT TYPE MISMATCHED
	Cause	Impermissible item is written as operand symbol attribute.
	Program processing	With symbol definition pseudo-instruction, symbol is not defined. In other cases, processing is continued with 0 as expression value.
	User action	Amend to correct description format.

Assembler's Error Message (cont'd)

#24	Message	OPERAND TYPE MISMATCHED
	Cause	Mismatch with operand description format permitted in mnemonic or pseudo-instruction.
	Program processing	With mnemonic, NOPs are generated equivalent to longest code of target device. In other cases, this statement is ignored as illegal statement.
	User action	Amend operand to correct description format.
#25	Message	BIT ADDRESS EXPRESSION ERROR
	Cause	Bit specification exceeds range 0 to 3.
	Program processing	With symbol definition pseudo-instruction, symbol is not defined. In other cases, processing is continued with 0 as bit value.
	User action	Ensure that bit specification is in range 0 to 3.
#26	Message	STRING LONGER THAN 2 CHARACTERS IS NOT ALLOWED
	Cause	String exceeding 2 characters cannot be used in expression.
	Program processing	Processing is continued using first 2 characters.
	User action	Ensure that string length is no more than 2 characters.
#28	Message	EXPRESSION "(" NEST OVERFLOW
	Cause	Nesting of parentheses "(" is too deep.
	Program processing	Processing is continued with 0 as operand value.
	User action	Ensure that parenthesis "(" nesting is maximum of 8 levels.
#30	Message	MORE THAN 80 CHARACTERS STRING IS NOT ALLOWED
	Cause	String exceeding 80 characters in length has been written.
	Program processing	Excess part of string is ignored. First 80 characters are valid.
	User action	Divide string into sections of up to 80 characters.
#32	Message	ALREADY DEFINED SYMBOL - IGNORE
	Cause	Symbol this statement attempts to define has already been defined.
	Program processing	Processing is continued with this symbol undefined.
	User action	Change symbol name and re-assemble.
#34	Message	ARITHMETIC OVERFLOW
	Cause	Expression description is too complex.
	Program processing	Processing is continued with 0 as expression value.
	User action	Rewrite expression in simpler form.
#35	Message	SUBSTITUTE OVERFLOW
	Cause	Operand value exceeds permissible object code substitution value range.
	Program processing	Processing is continued with 0 as object code substitution value.
	User action	Ensure that operand value is in this range.
#36	Message	OPERAND MUST BE CODE SEGMENT ADDRESS
	Cause	Items other than code or constant symbol is used as operand symbol attribute.
	Program processing	Processing is continued with constant symbol as operand symbol attribute.
	User action	Ensure that operand symbol attribute is code or constant symbol.
#37	Message	OPERAND MUST BE DATA (DATA) SEGMENT ADDRESS
	Cause	Items other than data or constant symbol is used as operand symbol attribute.
	Program processing	Processing is continued with constant symbol as operand symbol attribute.
	User action	Ensure that operand symbol attribute is data or constant symbol.
#38	Message	OPERAND MUST BE DATA (BIT) SEGMENT ADDRESS
	Cause	Items other than bit or constant symbol is used as operand symbol attribute.
	Program processing	Processing is continued with constant symbol as operand symbol attribute.
	User action	Ensure that operand symbol attribute is bit or constant symbol.

Assembler's Error Message (cont'd)

#40	Message	BALANCE ERROR
	Cause	',' not balanced in expression.
	Program processing	Processing is continued with supplementary ','.
	User action	Amend so that ',' balance is obtained.
#42	Message	ILLEGAL CHARACTER
	Cause	Illegal character(s) used in source program.
	Program processing	Processing is continued with illegal character (s) replaced with '!'.
	User action	Eliminate illegal character(s) and re-assemble.
#43	Message	SAME SEGMENT MUST HAVE SAME ATTRIBUTE TYPE
	Cause	Different relocation attributes are used for same segment name.
	Program processing	Processing is continued with relocation attribute specified first as valid.
	User action	Make all relocation attributes the same, or do not specify 2nd and subsequent relocation attributes.
#49	Message	TEXT EXISTS BEHIND END STATEMENT - IGNORE
	Cause	Source exists after END statement.
	Program processing	Source after END statement is ignored.
	User action	No action required (object code in which this error occurs is not affected) .
#50	Message	FORWARD REFERENCE IS NOT ALLOWED
	Cause	Symbol forward reference is not allowed in this operand expression.
	Program processing	Processing is continued with 0 as value of forward reference symbol.
	User action	Amend source program description to avoid forward reference.
#51	Message	PUBLIC SYMBOL IS OF ILLEGAL TYPE
	Cause	Impermissible symbol name is written in operand field of PUBLIC pseudo-instruction.
	Program processing	This statement is ignored as illegal statement.
	User action	Amend to correct description.
#52	Message	NO END STATEMENT IN SOURCE TEXT
	Cause	There is no END statement in source module.
	Program processing	Processing is continued with END statement assumed to be present at end of source module.
	User action	Write END statement at end of source module.
#54	Message	LOCATION COUNTER OVERFLOW
	Cause	Location counter value exceeds ROM size maximum value of target device.
	Program processing	Processing is continued with counter overflow.
	User action	Ensure that maximum value is not exceeded.
#56	Message	OPERAND MUST BE ABSOLUTE OR IN THIS SEGMENT
	Cause	Expression other than absolute expression has been written in operand, or symbol not defined in same segment has been referenced.
	Program processing	With ORG/CSEG/DSEG/DS pseudo-instructions, this statement is ignored as illegal statement.
	User action	Use absolute expression for operand expression and symbols in same segment.
#57	Message	ONLY ONE NAME STATEMENT IS ALLOWED
	Cause	Module name specification has been performed more than once.
	Program processing	Processing is continued with second and subsequent specifications regarded as illegal.
	User action	Do not use more than one module name specification.
#58	Message	INVALID MNEMONIC CODE
	Cause	Impermissible mnemonic has been written.
	Program processing	NOP codes are generated equivalent to longest code of target device as illegal statement.
	User action	Use permitted mnemonic.

Assembler's Error Message (cont'd)

#59	Message	SEGMENT SIZE OVERFLOW TO RELOCATION TYPE (INBLOCK)
	Cause	Segment with INBLOCK or INBLOCKA attribute exceeds one block range.
	Program processing	Processing is continued without further action.
	User action	Ensure that segment size is in one block range.
#60	Message	SEGMENT SIZE OVERFLOW TO RELOCATION TYPE (XBLOCK)
	Cause	Segment with XBLOCK or XBLOCKA attribute exceeds ROM range permitted for target device.
	Program processing	Processing is continued without further action.
	User action	Ensure that segment size is in ROM range of target device.
#61	Message	SEGMENT SIZE OVERFLOW TO RELOCATION TYPE (IENT)
	Cause	Segment with IENT attribute exceeds range 20H to 7FH.
	Program processing	Processing is continued without further action.
	User action	Ensure that segment size is in range 20H to 7FH.
#62	Message	SEGMENT SIZE OVERFLOW TO RELOCATION TYPE (SENT)
	Cause	Segment with SENT attribute exceeds range 0 to 7FFH.
	Program processing	Processing is continued without further action.
	User action	Ensure that segment size is in range 0 to 7FFH.
#64	Message	THIS PREDEFINE SYMBOL IS NOT ALLOWED
	Cause	Inappropriate specific address name symbol has been written as operand.
	Program processing	Processing is continued with this statement regarded as illegal statement.
	User action	Write correct operand.
#65	Message	EXCHANGE ANOTHER BRANCH OPERATION
	Cause	Illegal branch instruction has been written.
	Program processing	Processing is continued with operand value taken as "0".
	User action	Change to branch instruction of appropriate number of bytes.
#66	Message	STACK OVERFLOW BY STKLN - TOO COMPLICATED SYNTAX
	Cause	Value of STKLN pseudo-instruction operand exceeds range 0 to 100H.
	Program processing	Processing is continued with this statement regarded as illegal statement.
	User action	Ensure that value is in range 0 to 100H.
#67	Message	THIS STATEMENT IS NOT ALLOWED FOR DATA (BIT) SEGMENT
	Cause	Statement which cannot be written in data segment has been written.
	Program processing	Processing is continued with this statement regarded as illegal statement.
	User action	Delete this statement, or write it in correct segment.
#68	Message	THIS STATEMENT IS NOT ALLOWED FOR CODE OR DATA (DATA) SEGMENT
	Cause	Statement which cannot be written in code segment or data segment has been written.
	Program processing	Processing is continued with this statement regarded as illegal statement.
	User action	Delete this statement, or write it in correct segment.
#70	Message	OPERAND MUST BE IN A BYTE
	Cause	Operand value exceeds range 0 to FFH.
	Program processing	Processing is continued with operand value taken as 0.
	User action	Ensure that value is in range 0 to FFH.
#72	Message	ODD ADDRESS IS NOT ALLOWED FOR THIS OPERAND
	Cause	Odd address has been written as operand.
	Program processing	Processing is continued with operand value taken as 0.
	User action	Change operand value to appropriate even address.

Assembler's Error Message (cont'd)

#73	Message	THIS INSTRUCTION IS NOT ALLOWED AT ODD ADDRESS
	Cause	TBR and TCALL pseudo-instructions have been allocated to odd address.
	Program processing	Processing is continued with this statement regarded as illegal statement.
	User action	Allocate TBR/TCALL pseudo-instructions to appropriate even address.
#74	Message	TOO MANY ERRORS TO REPORT
	Cause	There are too many errors in this statement. (9 or more errors).
	Program processing	Processing is continued without outputting 9 or more error messages.
	User action	No action required.
#100	Message	PHASE ERROR
	Cause	The symbol value has changed during assembly. • When a BR pseudo-instruction is located on a block boundary, correct location addresses cannot be obtained in the subsequent assembly, and therefore an error is output. • When there is an error in a statement which contains a symbol reference , the values of symbols defined in lines after that line are also different, and therefore this error is output. • In addition, this error may be generated due to the influence of an error in another statement.
	Program processing	Processing is continued without further action.
	User action	• When a BR pseudo-instruction is located on a block boundary, replace that BR pseudo-instruction with an ordinary branch instruction. • Check other error statement.
#101	Message	ZERO DIVIDE ERROR
	Cause	Division by 0 has occurred during evaluation of expression.
	Program processing	Mnemonic processing is continued with value of expression taken to be 0. Processing is continued with this pseudo-instruction line is invalidated.
	User action	Write expression correctly.
#102	Message	SYMBOL QUANTITY IS OVERFLOW
	Cause	Number of symbols exceeds processable number (permissible number of symbols per source module is approx. 3,000).
	Program processing	Statements following location of this error are not processed.
	User action	Reduce number of symbols and re-assemble, or divide source program and perform split assembly.
#103	Message	SEGMENT QUANTITY IS OVERFLOW
	Cause	Number of segments exceeds processable number.
	Program processing	Statements following location of this error are not processed.
	User action	Reduce number of segments to re-assemble, or divide source program and perform split assembly.
#105	Message	CROSS REFERENCE TABLE OVERFLOW
	Cause	The table for creation of cross-reference list is full.
	Program processing	Cross-reference table creation is discontinued.
	User action	Assemble without creating cross-reference list, or divide source program and perform split assembly.
#311	Message	SEGMENT MUST BE ABSOLUTE
	Cause	ORG pseudo-instruction is written in relocatable segment (ORG pseudo-instruction can only be used in absolute segment).
	Program processing	This ORG pseudo-instruction is ignored.
	User action	Do not use the ORG pseudo-instruction in a relocatable segment. When object code allocated to an absolute address is generated, write that part as an absolute segment.

Assembler's Error Message (cont'd)

#401	Message	'macro' not found
	Cause	Macro definition is illegal.
	Program processing	That macro definition is invalidated, and processing is continued.
	User action	Describe correctly.
#402	Message	Illegal macro name
	Cause	Error in macro name.
	Program processing	That macro definition is invalidated, and processing is continued.
	User action	Describe the correct macro name.
#403	Message	Macro formal parameter error, parameter 'parameter' in macro 'macro_name'
	Cause	Error in macro formal parameter.
	Program processing	That macro definition is invalidated, and processing is continued.
	User action	Describe correctly.
#404	Message	'include' is not allowed in macro definition 'macro_nane'
	Cause	include has been described in macro definition.
	Program processing	include is invalidated, and processing is continued.
	User action	Describe include contents directly.
#405	Message	Missing 'endm', macro 'macro_name'
	Cause	There is no 'endm' description.
	Program processing	That macro is invalidated, and processing is continued.
	User action	Describe end correctly.
#406	Message	Illegal external macro 'macro_name'
	Cause	Illegal external macro has been specified.
	Program processing	Illegal external macro definition is invalidated, and processing is continued.
	User action	Describe correct external macro.
#407	Message	Illegal local symbol, symbol 'local_symbol' in macro 'macro_name'
	Cause	Error in local symbol description.
	Program processing	That local symbol is invalidated, and processing is continued.
	User action	Describe correct local symbol.
#409	Message	Input string too long
	Cause	Length of one line exceeds the limit.
	Program processing	That line is invalidated, and processing is continued.
	User action	Ensure that one line contains no more than 128 characters.
#412	Message	Illegal global symbol, symbol 'global_symbol'
	Cause	Error in global symbol.
	Program processing	That line is invalidated, and processing is continued.
	User action	Describe correct global symbol.
#414	Message	Nest level overflow
	Cause	Nest level exceeds the limit.
	Program processing	Block exceeding nest level is skipped, and processing is continued.
	User action	Set within the limit.
#415	Message	'switch' not found
	Cause	There is no switch description.
	Program processing	That line is invalidated, and processing is continued.
	User action	Amend to the correct description.
#416	Message	Missing '$ ends'
	Cause	There is no $ends description.
	Program processing	That case block is invalidated, and processing is continued.
	User action	Describe the correct endcase.

Assembler's Error Message (cont'd)

#417	Message	'$ if' not found
	Cause	There is no $ if description.
	Program processing	That line is invalidated, and processing is continued.
	User action	Amend to the correct description.
#418	Message	Missing '& endif'
	Cause	There is no $ endif description.
	Program processing	That if block is invalidated, and processing is continued.
	User action	Describe the correct endif.
#419	Message	Zero divide error
	Cause	Division by 0 has occurred during expression evaluation.
	Program processing	That line or block is invalidated, and processing is continued.
	User action	Describe the expression correctly.
#420	Message	Too many local symbols, symbol 'local_symbol' in macro 'macro_name'
	Cause	Number of local symbols in one macro exceeds the limit.
	Program processing	That macro is invalidated, and processing is continued.
	User action	Describe the number of local symbols within the limit.
#421	Message	Syntax error 'macro' statement, macro 'macro_name'
	Cause	Error in macro definition statement description.
	Program processing	Processing is continued with that macro definition invalidated.
	User action	Describe correctly.
#422	Message	'macro' statement too long, macro 'macro_name'
	Cause	Length of one macro definition statement line exceeds the limit.
	Program processing	That macro definition is invalidated, and processing is continued.
	User action	Ensure that one line contains no more than 128 characters.
#423	Message	Expression value is out of range
	Cause	Value of repeat macro expression exceeds 1023.
	Program processing	Processing is continued with expression as 1023.
	User action	In the case of expansion exceeding 1024, nest the repeat macro definitions.
#425	Mess age	Operand syntax error
	Cause	Error in definition statement operand description.
	Program processing	Processing is continued with that definition invalidated.
	User action	Describe the operand setting correctly.
#428	Message	'$ else' is after '$ else'
	Cause	There is an else statement after a $ else statement.
	Program processing	That definition is invalidated, and processing is continued.
	User action	Describe correctly.
#429	Message	Illegal $ case label
	Cause	$ case label is wrongly described.
	Program processing	Processing is continued with that $ case label invalidated.
	User action	Describe correctly.
#435	Message	Duplicate, case label definition
	Cause	Duplicate label definition.
	Program processing	Processing is continued with this definition invalidated.
	User action	Describe the $ case definition correctly.
#436	Message	$ case label after '$ default'
	Cause	There is a $ case label after $ default
	Program processing	Processing is continued with this definition invalidated.
	User action	Describe the $ case label correctly.

Assembler's Error Message (cont'd)

#439	Message	'macro' is not allowed in macro definition 'macro_name' 'lodm' is not allowed in macro definition 'macro_name'
	Cause	Macro definition statement in macro definition.
	Program processing	Processing is continued with that macro name macro definition invalidated.
	User action	Describe correctly.
#498	Message	Undefine symbol
	Cause	Undefined symbol in SET expression outside macro definition
	Program processing	Processing is continued without performing SET symbol registration.
	User action	Check whether symbol is assembler pseudo-instruction SET symbol
#499	Message	Unsuitable expression format
	Cause	Set expression outside macro definition is incompatible with this macro processor's SET expression format.
	Program processing	Processing is continued without performing SET symbol registration.
	User action	Check whether symbol is assembler pseudo-instruction SET symbol.

13.2 Linker's Error Messages

Error No. A001	Message	Missing input file
	Cause	Input file has not been specified in start line.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A002	Message	Too many input file
	Cause	Specified number of input files exceeds limit.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A003	Message	Unrecognized string 'specified string'
	Cause	Specified string cannot be interpreted.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A004	Message	Illegal file name 'file name'
	Cause	Type or length of characters in file name is illegal.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A005	Message	Illegal file specification 'file name'
	Cause	File name format is incorrect.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A006	Message	File not found 'file name'
	Cause	Specified file does not exist.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name then re-execute.
Error No. A007	Message	Input file specification overlapped 'file name'
	Cause	Overlapping input file specification.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A008	Message	File specification conflicted 'file name'
	Cause	Input or output file specification overlaps.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A009	Message	Unable to make file 'file name'
	Cause	Specified output file cannot be created.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check remaining disk capacity, etc.
Error No. A010	Message	Directory not found 'file name'
	Cause	Nonexistent drive or directory is included in output file name.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A011	Message	Illegal path, 'option'
	Cause	Item other than path name is specified in option in which path name should be specified as parameter.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name and option name, then re- execute.

Linker's Error Messages (cont'd)

Error No. A012	Message	Missing parameter 'option'
	Cause	Necessary parameter has not been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A013	Message	Parameter not needed 'option'
	Cause	Unnecessary parameter has been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A014	Message	Out of range 'option'
	Cause	Number specified as parameter exceeds permitted range.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A015	Message	Parameter is too long 'option'
	Cause	Parameter is too long.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A016	Message	Illegal parameter 'option'
	Cause	Parameter syntax is incorrect.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A017	Message	Too many parameters 'option'
	Cause	Number of parameters exceeds limit.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A018	Message	Option is not recognized 'option'
	Cause	Incorrect option has been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A019	Message	Parameter file nested
	Cause	-F option is included in parameter file.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct parameter file, then re-execute.
Error No. A020	Message	Parameter file read error 'file name'
	Cause	Parameter file read has failed.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check disk status.
Error No. A021	Message	Memory allocation failed
	Cause	Memory block acquisition has failed.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check mounted memory capacity and whether there are resident programs, etc.
Error No. F004	Message	BALANCE ERROR
	Cause	No correspondence between first and last ' (single quotation mark) used in option specification.
	Program processing	Program execution is halted.
	User action	Specify option correctly.

Linker's Error Messages (cont'd)

Error No. F006	Message	ILLEGAL FILE - file name
	Cause	Input file contents are illegal.
	Program processing	Program execution is halted.
	User action	Specify correct input file.
Error No. F012	Message	BRANCH TABLE OVERFLOW
	Cause	Number of branch tables exceeds processable number (approx. 1,000 can be created).
	Program processing	Program execution is halted.
	User action	Reduce number of BRCB instruction (note that branch tables may also be created by branch instruction optimization).
Error No. F013	Message	EXTERNAL SYMBOL OVERFLOW (MODULE: module name)
	Cause	Too many input module EXTRN symbols (approx. 500 EXTRN symbols can be used in one input module).
	Program processing	Program execution is halted.
	User action	Reduce number of EXTRN symbols.
Error No. F014	Message	SYMBOL TABLE OVERFLOW (MODULE: module name)
	Cause	Too many PUBLIC symbols (total of approx. 2,000 PUBLIC symbols can be used).
	Program processing	Program execution is halted.
	User action	Reduce number of PUBLIC symbols.
Error No. F015	Message	SEGMENT TABLE OVERFLOW (MODULE: module name)
	Cause	Too many segments (total of approx. 120 segments can be handled, including number of ORG pseudo-instructions).
	Program processing	Program execution is halted.
	User action	Reduce number of segments.
Error No. F016	Message	MODULE TABLE OVERFLOW (FILE: file name)
	Cause	Too many input modules (approx. 120 modules can be input).
	Program processing	Program execution is halted.
	User action	Reduce number of input modules.
Error No. A901	Message	WORKING TABLE SPACE EXHAUSTED
	Cause	Working area in memory is insufficient.
	Program processing	Program execution is halted.
	User action	Reduce number of segments, and PUBLIC & EXTRN symbols, or reduce number of unresolved branch instructions.
Error No. A903	Message	OPEN ERROR
	Cause	Input/output file cannot be opened.
	Program processing	Program execution is halted.
	User action	Check status of specified disk.
Error No. A904	Message	CLOSE ERROR
	Cause	Input/output file cannot be closed.
	Program processing	Program execution is halted.
	User action	Check status of specified disk.
Error No. A909	Message	DISK FULL
	Cause	Capacity of disk specified for output is insufficient.
	Program processing	Program execution is halted.
	User action	Delete unnecessary files , or use a new disk.

Linker's Error Messages (cont'd)

Error No. A999	Message	PROGRAM ERROR
	Cause	Memory contents have been overwritten by another program.
	Program processing	Program execution is halted.
	User action	Check method of use in multi-user mode.
Error No. F100	Message	INVALID OBJECT (MODULE: module name)
	Cause	File other than object module or load module file has been input.
	Program processing	Program execution is halted.
	User action	Check input file.
Error No. F102	Message	ILLEGAL CHARACTER (CHARACTER: character)
	Cause	Impermissible character has been specified.
	Program processing	Program execution is halted.
	User action	Do not use illegal characters.
Error No. F106	Message	RETRY COUNTER OVER (SEGMENT: segment name)
	Cause	Location has been attempted a reasonable number of times, changing the relocatable segment order, but without success (the linker cannot find a location method as there are a large number of segments).
	Program processing	Program execution is halted.
	User action	Input segments in the order in which they are to be located, and specify the -SQ option. (To find the size of each segment, specify the -SQ option without considering the segment order. Linkage will generate an error in most cases, but the size of each segment is shown in the map file).
Error No. F134	Message	MODULE NOT FOUND (MODULE: module name)
	Cause	Specified module is not in library file.
	Program processing	Program execution is halted.
	User action	Specify correct module name.
Error No. F135	Message	MODULE COUNT OVERFLOW (FILE: file name)
	Cause	There are too many module name specifications for library file.
	Program processing	Program execution is halted.
	User action	Reduce number of module name specifications.
Error No. A505	Message	ILLEGAL SEGMENT CLASS (SEGMENT: Segment name, CLASS: segment type)
	Cause	Illegal segment type in input file.
	Program processing	Program execution is halted.
	User action	Create input file again.
Error No. F301	Message	THIS MODULE IS DIFFERENT VER/REV (MODULE: module name)
	Cause	Input file version is not compatible with this linker version.
	Program processing	Program execution is halted.
	User action	Input file output by same version of assembler.
Error No. F302	Message	THIS MODULE IS OBJECT DIFFERENT CHIP OR SERIES (MODULE: module name)
	Cause	Modules output by non-75X assembler has been input.
	Program processing	Program execution is halted.
	User action	Input modules output by 75X assembler.
Error No. F303	Message	THIS MODULE IS OBJECT NOT LINKABLE (MODULE : module name)
	Cause	Modules output by non-75X assembler has been input.
	Program processing	Program execution is halted.
	User action	Input file output by 75X assembler.

Linker's Error Messages (cont'd)

Error No. F304	Message	INVALID FILE SYNTAX (FILE: file name)
	Cause	Module output by the non-75X assembler has been input.
	Program processing	Program execution is halted.
	User action	Input files output by 75X assembler.
Error No. F307	Message	SAME NAME SEGMENT IN DIFFERENT CLASS (SEGMENT: segment name)
	Cause	Segments with same name but different segment type exist.
	Program processing	Processed as different segments.
	User action	Change segment name or segment type and re-link.
Error No. W300	Message	CHIP TYPE MISMATCH (MODULE: module name)
	Cause	Input module chip type is different from first input module chip type.
	Program processing	First input module chip type is valid.
	User action	Standardize chip type for all input modules and re-assemble.
Error No. W308	Message	MULTIPLE DEFINED (MODULE: module name SYMBOL: symbol name)
	Cause	Same external definition symbol name has been declared more than once.
	Program processing	Only first external definition symbol name input is valid.
	User action	Change to different external definition symbol name.
Error No. W309	Message	SAME NAME SEGMENTS OF DIFFERENT ALIGNMENT TYPE (MODULE: module name SEGMENT: segment name)
	Cause	Different attributes have been specified for segments with same name.
	Program processing	These sane name segments are not linked, and are processed as different segments.
	User action	Specify same attribute for segments with same name.
Error No. W310	Message	UNRESOLVED SYMBOL (MODULE: module name SYMBOL: symbol name)
	Cause	External reference symbol information has not been resolved. There is no external reference symbol corresponding to the external definition symbol.
	Program processing	Part of text corresponding to this external reference symbol is not corrected.
	User action	Link to module containing corresponding external reference symbol.
Error No. W311	Message	TYPE MISMATCH (MODULE: module name SYMBOL: symbol name)
	Cause	External definition symbol and external reference symbol segment types do not match.
	Program processing	External definition symbol segment type is valid.
	User action	Ensure that external definition symbol and external reference symbol segment types match.
Error No. F402	Message	INVALID STACK SIZE APPOINTED
	Cause	Increment or decrement size has been specified for size 0 stack in -SZ option.
	Program processing	Program execution is halted.
	User action	Do not specify increment or decrement size in -SZ option.
Error No. W403	Message	EVALUATED STACK SIZE IS INVALID
	Cause	Stack size exceeds 100H.
	Program processing	Processing is continued with 0 as stack size.
	User action	Use -SZ option specification of 100H or less, or amend so that accumulated value of stack size for each module is 100H or less.
Error No. F405	Message	SPECIFIED SEGMENT NOT FOUND IN INPUT MODULES (SEGMENT: segment name)
	Cause	Code segment name specified by order or address option does not exist in input file.
	Program processing	Program execution is halted.
	User action	Specify correct segment name.

Linker's Error Messages (cont'd)

Error No. F406	Message	SPECIFIED SEGMENT NOT FOUND IN SPECIFIED CLASS (CLASS: segment type, SEGMENT: segment name)
	Cause	Segment name specified by order or address option does not exist in specified code segment.
	Program processing	Program execution is halted.
	User action	Specify correct segment name.
Error No. F407	Message	SPECIFIED SEGMENT IS ABSOLUTE (SEGMENT: segment name)
	Cause	Allocation address has been specified in absolute segment by address option.
	Program processing	Address option is ignored.
	User action	Re-link without specifying allocation address in absolute segment.
Error No. F408	Message	ADDRESS FOR SEGMENT SPECIFIED MORE THAN ONCE (SEGMENT: segment name)
	Cause	Multiple address specifications have been made for same segment.
	Program processing	Program execution is halted.
	User action	Eliminate duplicate specifications.
Error No. W409	Message	STACK OVER THE DATA MEMORY
	Cause	Stack size specified by STKLN pseudo-instruction and -SZ option exceeds range specifiable for stack.
	Program processing	Processing is continued with 0 as stack size.
	User action	Ensure that total stack size specified by STKLN pseudo-instruction and -SZ option does not exceed range specifiable for stack.
Error No. W411	Message	ALIGNMENT NOT COMPATIBLE WITH ASSIGNED ADDRESS (SEGMENT: segment name)
	Cause	Relocation attribute of address-specified segment is not compatible with specified address.
	Program processing	Segment is located with relocation attribute taken as valid.
	User action	Ensure that specification does not involve incompatibility between address specification and relocation attribute.
Error No. F412	Message	CAN NOT CREATE BRANCH TABLE (SEGMENT: segment name)
	Cause	Branch tables required for absolute or address-specified segment cannot be created in same block.
	Program processing	Processing is continued with branch instruction unresolved.
	User action	Move segment specification address to another block, or reduce number of unresolved branch instructions.
Error No. F415	Message	LOCATION OVERFLOW AT CODE MEMORY (SEGMENT: segment name)
	Cause	Attempt has been made to locate code segment exceeding maximum target chip ROM value.
	Program processing	Segment is located with maximum target chip ROM value exceeded.
	User action	Amend so that code segment fits in ROM range of target device.
Error No. F416	Message	LOCATION OVERFLOW AT DATA MEMORY (SEGMENT: segment name)
	Cause	Attempt has been made to locate data segment exceeding maximum target chip RAM value.
	Program processing	Segment is located with maximum target chip RAM value exceeded.
	User action	Amend so that data segment fits in RAM range of target device.
Error No. F418	Message	SEGMENT SIZE OVER (ALIGN: relocation attribute SEGMENT: segment name)
	Cause	Maximum segment size stipulated by relocation attribute has been exceeded.
	Program processing	Performs location ignoring maximum segment size stipulated by relocation attribute.
	User action	Divide segment so that it is within maximum segment size range stipulated by relocation attribute.

Linker's Error Messages (cont'd)

Error No. F419	Message	CAN NOT ALLOCATE IN FIXED AREA (ALIGN: relocation attribute SEGMENT: segment name)
	Cause	This segment cannot be located in location area in ROM corresponding to specified attribute (IENT: 20H to 7FH, SENT: 0H to 7FFH, INBLOCK and INBLOCKA: area not running over 4K-byte block, XBLOCK: 0 to 3FFFH).
	Program processing	Perform location exceeding location area stipulated by relocation attribute.
	User action	Perform absolute specification or correct address option so that segment can be located in range corresponding to reallocation attribute specified for this segment.
Error No. W422	Message	SEGMENTS OVERLAP (SEGMENT: segment name AND segment name)
	Cause	An attempt has been made to locate multiple overlapping segments in same area.
	Program processing	Processing is continued without further action (however, linkage results are not assured).
	User action	Specify overlapping segments with address specification or absolute specification so that they do not overlap.
Error No. F423	Message	SEGMENT IN RESERVED SPACE (SEGMENT: segment name)
	Cause	An attempt has been made to locate segment in area in which location of segment for which absolute specification or address specification has been made is prohibited.
	Program processing	Segment is located in location prohibited area.
	User action	Ensure that area specified by -RS option action and area for which absolute specification or address specification is made do not overlap.
Error No. W501	Message	REFERENCE TYPE ERROR (AT address IN segment name)
	Cause	Impermissible item has been used in reference symbol segment type.
	Program processing	Processing is continued with "no segment type" as symbol type.
	User action	Correct so that external reference symbol with correct segment type is referenced, and re-assemble .
Error No. W502	Message	EVALUATED VALUE EXCEEDS THE RANGE (AT address IN segment name)
	Cause	Calculated operand value exceeds permissible range.
	Program processing	Text correction is performed with 0 as operand value.
	User action	Correct so that operand value is in permissible range, and re-assemble
Error No. W503	Message	REFERENCE SYMBOL IS UNRESOLVED
	Cause	Information on external reference symbol referenced in text has not been resolved.
	Program processing	Part of text corresponding to this external reference symbol is not corrected .
	User measure	Link to module containing corresponding external reference symbol.

13.3 Object Converter Error Message

Error No. A001	Message	Missing input file
	Cause	Input file has not been specified in start line.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A002	Message	Too many input file
	Cause	Specified number of input files exceeds limit.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A003	Message	Unrecognized string ‘specified string’
	Cause	Specified string cannot be interpreted.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Confirm program start method, then re-execute.
Error No. A004	Message	Illegal file name ‘file name’
	Cause	Type or length of characters in file name is illegal.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Checks/correct file name, then re-execute.
Error No. A005	Message	Illegal file specification ‘file name’
	Cause	File name format is incorrect.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A006	Message	File not found ‘file name’
	Cause	Specified file does not exist.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute .
Error No. A007	Message	Input file specification overlapped ‘file name’
	Cause	Overlapping input file specification.
	Program processing	Checks start line syntax, then halts processing and returns control to OS .
	User action	Check/correct file name, then re-execute.
Error No. A008	Message	File specification conflicted ‘file name’
	Cause	Input or output file specification overlaps.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A009	Message	Unable to make file ‘file name’
	Cause	Specified output file cannot be created.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check remaining disk capacity, etc.
Error No. A010	Message	Directory not found ‘file name’
	Cause	Nonexistent drive or directory is included in output file name.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name, then re-execute.
Error No. A011	Message	Illegal path ‘option’
	Cause	Item other than path name is specified in option in which path name should be specified as parameter.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct file name and option name, then re-execute.

Object Converter Error Messages (cont'd)

Error No. A012	Message	Missing parameter 'option'
	Cause	Necessary parameter has not been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A013	Message	Parameter not needed 'option'
	Cause	Unnecessary parameter has been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A014	Message	Out of range 'option'
	Cause	Number specified as parameter exceeds permitted range.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A015	Message	Parameter is too long 'option'
	Cause	Parameter is too long.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A016	Message	Illegal parameter 'option'
	Cause	Parameter syntax is incorrect.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A017	Message	Too many parameters 'option'
	Cause	Number of parameters exceeds limit.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A018	Message	Option is not recognized 'option'.
	Cause	Incorrect option has been specified.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct option syntax, then re-execute.
Error No. A019	Message	Parameter file nested
	Cause	-F option is included in parameter file.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check/correct parameter file, then re-execute.
Error No. A020	Message	Parameter file read error 'file name'
	Cause	Parameter file read has failed.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check the disk status.
Error No. A021	Message	Memory allocation failed
	Cause	Memory block acquisition has failed.
	Program processing	Checks start line syntax, then halts processing and returns control to OS.
	User action	Check mounted memory capacity and whether there are resident programs, etc.
★ Error No. A100	Message	'File name' Illegal processor type
	Cause	The target assembly or compilation product is not covered by this program.
	Program Processing	Program execution is halted.
	User Action	Check if the load module file is correct. Confirm the target product to be assembled or compiled. Check if the overlay file is the correct version.

Object Converter Error Messages (cont'd)

★	Error No. A101	Message	'File name' Invalid input file (or made by different host machine)
		Cause	A file other than a load module file was input, or a load module file created on an incompatible host machine was input.
		Program Processing	Program execution is halted.
		User action	Check if someone didn't attempt to input a file other than a load module file. When input of a load module file created on an incompatible host machine is attempted, use a hot machine which is compatible.
★	Error No. A103	Message	Symbol 'symbol name' Illegal attribute
		Cause	There are errors in the symbol attribute values of input symbols.
		Program Processing	Program execution is halted.
		User action	Execute again from assemble or from compile.
★	Error No. 104	Message	'File name' Illegal input file - not linked.
		Cause	Input of an object module file (an unlinked object file) was attempted.
		Program Processing	Program execution is halted.
		User action	Input this file to the HEX converter after it has been linked.
★	Error No. A105	Message	Insufficient memory in host machine.
		Cause	There is not sufficient memory in the system to run the program.
		Program Processing	Program execution is halted.
		User action	When it is possible to add more memory to the host machine, add more memory. Add to the amount of memory which can be used by other application programs. When it is not possible to add more memory, linking cannot be done with this host machine.
★	Error No. A106	Message	Illegal symbol table
		Cause	There are errors in the symbol table in the input load module file.
		Program Processing	Program execution is halted.
		User action	Relink and create a correct load module file.
★	Error No. A900	Message	Can't open file 'file name'
		Cause	The file cannot be opened.
		Program Processing	Program execution is halted.
		User action	Specify the correct file name. Check the disk condition (available space, media condition, etc.). Prepare a correct file (particularly in the case of an overlay file).
★	Error No. A901	Message	Can't close file 'file name'
		Cause	The file cannot be closed.
		Program Processing	Program execution is halted.
		User action	Check the disk condition (available space, media condition, etc.).
★	Error No. A902	Message	Can't read file 'file name.'
		Cause	The file could not be read correctly.
		Program Processing	Program execution is halted.
		User action	Check the disk condition where the file exists (available space, media condition, etc.). Check if the file where there was an error is the correct file.
★	Error No. A903	Message	Can't access file 'file name'
		Cause	The file could not be read correctly or it could not be written correctly.
		Program Processing	Program execution is halted.
		User action	Check the disk condition where the file exists (available space, media condition, etc.). Check if the file where there was an error is the correct file.
★	Error No. A904	Message	Can't write file 'file name'
		Cause	Data could not be written correctly to the output file.
		Program Processing	Program execution is halted.
		User action	Check the disk condition where the file exists (available space, media condition, etc.).

Object Converter Error Messages (cont'd)

★ Error No. F200	Message	Undefined symbol 'symbol name'
	Cause	There is a symbol with an unresolved address.
	Program Processing	Set the value at 0, then output the symbol table and continue program execution.
	User action	Please define the value of this symbol. This symbol is referred to as an external reference symbol, but when no external definition is provided, please define it externally in the module defined by the symbol value.
★ Error No. F201	Message	Out of address range
	Cause	The load module file's object address exceeds the limit.
	Program Processing	Execution of the program continues as is.
	User action	Specify the output object area correctly.
★ Error No. W300	Message	xxxxH - yyyyH overlapped
	Cause	The object is overlapping with respect to the addresses between xxxx and yyyy.
	Program Processing	Output the object as is and continue processing.
	User action	Using expansion space, etc., link and correct so that no object is output that is overlapping with respect to the same address in the same address space.

13.4 Librarian Error Messages

(1) Error message for fatal file I/O errors

Format: FATAL I/O ERROR
DEVICE : device name
FILE NAME: file name
ERROR : error message
PROGRAM ABORTED

Message	READ ERROR
Cause	Disk read error has occurred.
Program processing	Control is returned to OS.
User action	Check disk status.
Message	WRITE ERROR
Cause	Write error has occurred on disk.
Program processing	Control is returned to OS.
User action	Check disk status.
Message	NO DIRECTORY EXIST
Cause	There is insufficient directory area for recording files on disk.
Program processing	Control is returned to OS.
User action	Use a new floppy disk.
Message	END OF VOLUME
Cause	There is insufficient directory area for writing files on disk.
Program processing	Control is returned to OS.
User action	Increase disk area.
Message	OPEN UNSUCCESSFUL
Cause	Specify correct file name.
Program processing	Control is returned to OS.
User action	Correct file name.
Message	SELECT ERROR
Cause	Nonexistent device name has been specified.
Program processing	Control is returned to OS.
User action	Specify correct device name.

(2) Error message for start command

Format: *** ERROR error number error message
PROGRAM ABORTED

Librarian's Error Messages

Error No. F001	Message	MISSING FILE SPECIFICATION
	Cause	Parameter file name has not been specified.
	Program processing	Control is returned to OS.
	User action	Specify parameter file name and re-execute.
Error No. F002	Message	ILLEGAL FILE SPECIFICATION - file name
	Cause	File name is not correct.
	Program processing	Control is returned to OS.
	User action	Specify correct file name and re-execute.
Error No. F005	Message	FILE NOT FOUND - file name
	Cause	Specified file does not exist.
	Program processing	Control is returned to OS.
	User action	Specify correct file name and re-execute.
Error No. F007	Message	ILLEGAL OR MISSING PARAMETER: parameter
	Cause	Parameter has not been specified for control which requires parameter, or illegal parameter has been specified.
	Program processing	Control is returned to OS.
	User action	Specify correct parameter and re-execute.
Error No. F008	Message	CONTROL IS NOT RECOGNIZED: string
	Cause	Illegal string has been specified as control.
	Program processing	Control is returned to OS.
	User action	Specify correct control and re-execute.
Error No. F009	Message	INVALID SYNTAX: X
	Cause	Syntax error in start command.
	Program processing	Control is returned to OS.
	User action	Specify command file name and control correctly, then re-execute.

(3) Error messages in librarian processing

Format: *** ERROR error number error message

Librarian's Error Messages

Error No. W201	Message	INVALID COMMAND
	Cause	Subcommand name is incorrect.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W202	Message	INVALID SYNTAX
	Cause	Subcommand parameter specification is incorrect.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W203	Message	FILE NOT FOUND
	Cause	Specified file does not exist.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W204	Message	MODULE NOT FOUND - file name (module name)
	Cause	Specified file does not exist.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W205	Message	NOT LIBRARY FILE - file name
	Cause	File specified as library file is not library file.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W206	Message	NOT OBJECT FILE - file name
	Cause	File specified as object module file is not object module file.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W207	Message	FILE ALREADY EXISTS - file name
	Cause	File specified in CREATE command already exists.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W208	Message	MODULE ALREADY EXISTS - file name (module name)
	Cause	In output library file, the module of which name is same as that in input file exists.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W209	Message	PARAMETER OVER
	Cause	Too many parameters
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W210	Message	PUBLIC SYMBOL symbol name IN file name (module name)
	Cause	External definition symbol defined in module in file specified in ADD or REPLACE subcommand already exists in output library file.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.

Librarian's Error Messages (cont'd)

Error No. W211	Message	MISSING FILE SPECIFICATION
	Cause	There is no file name specification in subcommand parameter.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W212	Message	ILLEGAL FILE SPECIFICATION - file name
	Cause	Illegal file name has been specified in subcommand parameter.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W213	Message	FILE SPECIFICATION CONFLICTED - file name
	Cause	Input file name and output file name specified by subcommand parameter do not match.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand .
Error No. W214	Message	ILLEGAL FILE - file name
	Cause	File specified by subcommand parameter is not object module file, load module file or library file or, write-protected file has been specified as output file.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-execute with correct subcommand.
Error No. W215	Message	CHECK SUM ERROR - file name
	Cause	Check sum error in file.
	Program processing	Program ignores this subcommand and waits for input of next subcommand.
	User action	Re-create file in which checksum error occurred , then re-execute.
Error No. W216	Message	EXIT SUBCOMMAND NOT FOUND - file name
	Cause	There is no EXIT subcommand in parameter file specified at start time.
	Program processing	Processing is performed assuming presence of EXIT subcommand at end of parameter file (control is returned to OS).
	User action	Insert EXIT subcommand in parameter file.
Error No. A901	Message	WORKING TABLE SPACE EXHAUSTED
	Cause	Work area (memory) is insufficient.
	Program processing	Control is returned to OS.
	User action	Increase memory.
Error No. A902	Message	INVALID FILE SYNTAX - file name
	Cause	File format is incorrect.
	Program processing	Control is returned to OS.
	User action	Re-execute with correct subcommand.
Error No. A999	Message	PROGRAM ERROR: error code
	Cause	Program bug has been detected.
	Program processing	Control is returned to OS.
	User action	Inform NEC of program name, version and error code.

13.5 List Converter Error Messages

Message	-l <L file> -a<A file> [-o<O file>] L: Linker output file A: Assembler output file O: Output file name
Cause	Start command option name is incorrect.
Program processing	Execution is halted and control is returned to OS.
User action	Check option name and start again.
Message	File name file open error
Cause	<1> Input file has not been found. <2> There is a space between the option and file name.
Program processing	Execution is halted and control is returned to OS.
User action	<1> Check the input file name. With the -L output, the file type must also be specified. With the -A option, check that there is both '.REL' and '.PRN'. <2> Start again with no space between option and file name.
Message	Error #n (n = 20 to 27, 35, 36, 40 to 47) program aborted
Cause	Error in input load module file or input object module file.
Program processing	Execution is halted and control is returned to OS.
User action	Check the input file name. If there is no error, the file is damaged and must be created again with the linker or assembler.
Message	Error #30 program aborted
Cause	Error in input assembly list file.
Program processing	Execution is halted and control is returned to OS.
User action	Check the input file name. If there is no error, the file is damaged and must be created again with the assembler.
Message	Error #37 program aborted
Cause	<1> Error in input assembly list. <2> ORG pseudo-instruction is not written in upper-case characters starting at column 9 in the source module. <3> There is not ORG pseudo-instruction or CSEG pseudo-instruction in first segment of source module.
Program processing	Execution is halted and control is returned to OS.
User action	Check the input file name. Write the ORG pseudo-instruction in upper-case characters starting at column 9 in the source module. Write an ORG pseudo-instruction or CSEG pseudo-instruction at the start of the first segment of the source module.

★ 13.6 Library Converter Error Messages

Error No. A001	Message	Illegal input file
	Cause	The input file contents are incorrect.
	Program Processing	Program execution is halted.
	User Action	Specify the correct input file.
Error No. A002	Message	Memory allocation failed
	Cause	Allocation of memory blocks failed.
	Program Processing	Program execution is halted.
	User Action	Check the installed memory capacity or check if there are any resident programs running.

APPENDIX A. LIST OF OPTIONS

Options for each of the programs in the assembler package are shown here in tabular form. Please refer to these when carrying out program development.

A.1 List of Assembler Options

No.	Description Format	Function/Category	Default Interpretation	Pages
1	-C product	Specification of assembler target product	Cannot be omitted	p.82
2	-M mode	75XL Series CPU mode switching Cannot be omitted when a 75XL Series devices is used.	Cannot be specified when a 75XL Series devices is used.	p.86
3	-O[file name] -NO	Object module file specification	'Input file name.REL' is created on the current path.	p.88
4	-J -NJ	Object module file forced output specification	-NJ	p.100
5	-G -NG	Specification of output to the object module of symbol information for debugging	-G	p.101
★ 6	-GA -NGA	Specifies output of source debugging information object module files.	-GA	p.94
7	-P[file name] -NP	Assembly list file specification	'Input file name.PRN' is created on the current path.	p.95
8	-E[file name] -NE	Error list file specification	-NE	p.97
9	-KS -NKS	Symbol table list output specification	-NKS	p.98
10	-KX -NKX	Cross-reference list output specification	-NKX	p.100
★ 11	-CA -NCA	Specifies distinguishing between upper/lower case letters. -CA: Do not distinguish between upper/lower case letters. -NCA: Distinguish between upper/lower case letters.	-NCA	p.102
★ 12	-S -NS	Sets the symbol name length -S: Sets a maximum of 31 characters. -NS: Sets a maximum of 8 characters.	-S	p.103
★ 13	-D symbol Name [= numerical value] [,symbol Name [= numerical value] -ND symbol Name	Sets the symbol definition	None	p.104
14	-LL[number of lines] -LW[number of columns	Number of lines and columns per page of assembly list file	-LL66 -LW132	p.105
15	-LT[number of characters]	Specification of number of TAB expansion columns in assembly list file	-LT8	p.111
16	-KA -NKA	Assembly list output specification	-KA	p.112
17	-I path name [, path name...]	Include file search path specification	Searching is executed on the path specified by the 'INCLUDE' control command, the path of the source module file and the path specified by environmental variable 'INC75X.'	p.113
18	-F file name	Parameter file specification	All operation and file names are read from command line	p.115
★ 19	-Y path name	Specifies the device file search path.	It searches in the sequence of the ' .. \DEV' path with respect to the RA75X starting path, the RA75X starting path, the current directory, and the path set in the environment variable 'PATH.'	p.117

A.2 List of Linker Options

No.	Description Format	Function/Category	Default Interpretation	Pages
1	-M[module name]	Output module name specification	Object module name of first file input	p.146
2	-P [file name] -NP	Link list file specification	First input ‘input file.MAP’ is output to current path	p.148
3	-KM -NKM	Map list output specification	-KM	p.149
4	-KP -NKP	Local symbol list output specification	-KP	p.153
5	-KL -NKL	Specification of the local symbol list	-KL	p.156
6	-CD (segment name [‘address] [,...])	Code segment relocation address location order specification (multiple specifications possible)	Automatically located by linker	p.159
7	-RS (start address, end address[, ..., ...])	Code segment allocation prohibited area specification (multiple specifications possible)	ROM area not incorporated in target device	p.162
8	-SQ -RN	Segment location order specification	-RN	p.165
9	-SK address	Sets stack address in assembler reserved word ‘STACK’	Set automatically by linker	p.168
10	-SZ[(] size	Stack size change specification	None	p.171
11	-NTB	Specifies suppression of automatic branch table creation	Created automatically	p.172
12	-O[file name] -NO	Load module file specification	The initially input ‘Input file.LNK’ is output to the current path.	p.174
13	-J -NJ	Load module file forced output specification	-NJ	p.176
14	-F file name	Parameter file specification	All options and file name are read from command line	p.178
★ 15	-Y path name	Specifies the device file search path.	Searching is executed on the ‘..\DEV’ path, the LK75X run path, the current directory, and the environmental variable ‘PATH,’ in that order, for the LK75X run path.	p.180

A.3 List of Object Converter Options

No.	Description Format	Function/Category	Default Interpretation	Pages
1	-S[file name] -NS	Symbol table file specification	'input file name .SYM' is created in current path	p.192
★ 2	-R -NR	Specifies the HEX format object output sequence.	-NR	p.194
3	-U fill value	Mask ROM ordering object output specification	None	p.195
4	-O[file name]	HEX format object module file specification	'Input file name.HEX' is created in current path.	p.197
★ 5	-E[file name] -NE	Specifies the error list file.	-NE	p.198
★ 6	-F file name	Specifies the parameter file.	Reads all the options and file names from the command line.	p.199
★ 7	-Y path name	Specifies the device file search path.	It searches in the sequence of the ' .. \DEV' path with respect to the RA75X starting path, the OC75X starting path, the current directory, and the path set in the environment variable 'PATH.'	p.200

A.4 List of Librarian Subcommands

No.	Description	Function	Abbreviated Format	Pages
1	CREATE library file name	Library file creation	C	p.211
2	ADD { object module file name / library file name / [(object module name[, ...])] ↑ } [, ...] TO update library file name	Module recording	A	p.213
3	DELETE library file name (object module name [, ...])	Module deletion	D	p.216
4	REPLACE { object module file name / library file name / [(object module name[, ...])] ↑ } FROM update library file name	Module replacement	R	p.219
5	LIST library file nane [(object module name [, ...])] [. ...] [TO list file name][PUBLICS]	Library information output	L	p.230
6	EXIT	Librarian termination	E	p.233

A.5 List of Converter Options

No.	Description Format	Function/Category	Default Interpretation	Pages
★ 1	-L file name	Input load module file name specification.	Input assembly list file name.LNK	p.245
2	-O file name	Output absolute assembly list file name specification.	Input assembly list file primary name .P	p.247
★ 3	-R [file name]	Input object module file name specification	Input assembly list file primary name .REL	p.249
★ 4	-E [file name] -NE	Error list file specification	-NE	p.250
★ 5	-F file name	Parameter file specification	All option and filename are read from command line	p.251

★ A.6 List of Librarian Converter Options

No.	Description Format	Function/Category	Default Interpretation	Pages
1	-O [file name]	Librarian converter output librarian file specification	'input file name,CNV' is created in current path	p.258

[MEMO]

APPENDIX B. MAXIMUM CAPABILITIES

The maximum capabilities of the assembler package are shown for the next after:

- Source statement length
- Number of symbols that can be written
- Number of segments that can be written
- Number of branch tables that can be created

(1) Source statement length

Program Name	Maximum Capability
Assembler	220 characters (including C_R and L_F)

(2) Number of symbols that can be written

Program Name	Maximum Capability	
★ Assembler	• In assembly	Approx. 3,000
Linker	• Local symbols	No limit
	• External definition (PUBLIC)	approx. 3,000/all modules
	• External reference (EXTRN) symbols	Approx. 500/module

(3) Number of segments that can be written

Program Name	Maximum Capability
Assembler	Total of approximately 120 for (a) to (c) below per module: (a) Number of segment definition pseudo-instructions (b) Number of ORG pseudo-instructions (c) 2 × Number of VENT pseudo-instructions
Linker	Total of approximately 250 for (a) to (d) below for all modules: (a) 2 × Number of input modules (b) Number of segments (c) Number of ORG pseudo-instructions (d) 2 × Number of VENT pseudo-instructions

(4) Number nch table that can be treated

Program Name	Maximum Capability
Linker	Approx. 1,000

★ (5) Other

Program Name	Maximum Capability	
Assembler	• Number of local symbols in 1 macro	100 (including temporary parameters)
	• Nest level	Approx. 64Kbytes
	• Macro body area size	32 Levels
	• macro instructions $IF instruction, $SWITCH instructions, $INCLUDE instruction)	
	• Number of repetitions of a repeating macro	1023 times

APPENDIX C. POINT FOR ATTENTION

Points requiring attention when using the assembler package are shown here.

No.	Point for Attention	Remedy/Action	Reference
1	Caution on memory bit operations: mem.bit object code is generated even if immediate data in the range 0FB0H.0 to 0FBFH.3 or 0FF0H.0 to 0FFFH.3 is specified.	If mem.bit object code is to be generated, a reserved word must be specified in the ranges shown on the left.	Language Volume 3.5 Operand Characteristics
2	Caution on segments with same name: If segment with the same name are written in one source module, the list converter may not function correctly.	Do not write modules with the same name in one source module when using the list converter.	Language Volume 4.2 Segment Definition Pseudo-Instructions
3	Caution on source program description: If the assembly list of a source program which does not follow the rules shown on the right is input, the list converter may abort due to an error with the result that list is not converted correctly.	The following rules should be followed: <1> Write VENTn and ORG pseudo-instructions in upper-case characters starting at column 9 in the source program. <2> Do not use a NOLIST control instruction. <3> Do not use segments with the same name in the same module. <4> Be sure to write a segment definition pseudo-instruction before writing an instruction which generates object code.	Language Volume Chapter 4. Pseudo-Instructions
4	Restriction on input files: All files input to the list converter must be free of errors.	Check that the following files are error-free: Assembly list file (.PRN) Object module file (.REL) Load module file (.LNK)	

No.	Point for Attention	Remedy/Action	Reference
5	Cautions on input file name to be used with debugger: For input file to be used with debugger in-circuit emulator., a file name is created as a module name in the process of assembly. Therefore, if the first letter of the file name is written in number, an error results on debugger side upon loading.	<1> The first letter of the file name should be written in letters other than numeric. <2> Change existing file name with 'name' pseudo-instruction.	
6	Bug concerning BRCB instruction: <1> When jump destination address of BRCB instruction is BLOCK external reference as 'number of label-constant', output is not performed in branch table map address ascending order. <2> When jump destination address of BRCB instruction is BLOCK external reference in the form of 'label-constant' and there is a BRCB instruction of different description format at the same jump destination address in the same block, an extra branch table is created.		
★ 7	Caution when using byte separation operators (HIGH, LOW): If the item is a relocatable item or an external reference item, nesting cannot be done. However, if used in combination with the BRCB, EQU and SET commands, absolute items only can be used.		Language Volume Table 3-10 Combination of items and operators (except external reference items) according to relocation attributes. Table 3-11 Combination of items and operators (external reference items) according to relocation attributes.
★ 8	Cautions concerning library converter options: Object modules included in library files converted by the Library Converter cannot be debugged.		Chapter 9 Library Converter
★ 9	Cautions concerning the assembler options: IE-75000-R and IE-75001-R do not respond to source debugging. Also, there is no distinguishing between upper case and lower case letters in symbol names. Only symbol names with lengths of 8 characters will be recognized.	Set options as shown below . ~ NGA, -CA -NS	4.4.4 (6) -GA/-NGA (11) -CA/-NCA (12) -S/-NS

APPENDIX D. SAMPLE PROGRAMS

This chapter describes some of sample lists of programs etc., that are used in assembler package.

D.1 Source Lists

(1) 75XTEST1.ASM

```
$  TITLE='A-D CONVERT'
;*********************************************
;***   A-D CONVERT PROGRAM       ***
;*********************************************
            NAME      AD_MAIN
            EXTRN     CODE(SIOSUB,ADVONC)
            PUBLIC    TDATA, SEL15
            STKLN     10
            VENT      MBE=1, RBE=1, MAIN
            VENT 4    MBE=1, RBE=0, ADCONV
SEG0        DESG      1 AT 10H
TDATA:      DS        2

;***        GET1 TABLE            ***
SEG1        CSEG      IENT
SEL15:      SEL       MB15

;***        MAIN ROUTINE          ***
SEG2        CSEG      INBLOCK
MAIN:       SEL       RB1

            GETI      SEL15             ;STACK POINTER SET
            MOV       XA, #STACK        ;
            MOV       SP, XA            ;

            MOV       A, #0011B
            MOV       PCC, A            ;PCC ← 0011B

;**         DATA RAM 0H-13FH ZERO CLEAR   **

            SEL       MB1
            MOV       HL, #3FH
            MOV       XA, #00
LOOP1:      MOV       @HL, A            ;100H-13FH
            DECS      HL
            BR        LOOP1
            SEL       MB0
LOOP2:      MOV       @HL, A            ;0H-FFH
            DECS      HL
            BR        LOOP2
```

```
;**       TIMER SET (SAMPLING TIME = 30MSEC, FXX=4.19MHz)   **

          GETI     SEL15          ;SEL     MB15
          MOV      XA, #79H
          MOV      TMOD0, XA
          MOV      XA, #01001100B
          MOV      TM0, XA
          EI
          EI       IET0
          SEL      MB1
LOOP3:    MOV      XA, #0H
          MOV      B, #00H
LOOP4:    SKE      B, #08H
          BR       LOOP4
          CALL     !HEIKIN
          MOV      TDATA, XA
          CALL     !SIOSUB
          BR       LOOP3

;***      HEIKIN   (SAMPLE NUMBERS = 8)    ***
SEG3      CSEG     SENT
HEIKIN:   MOV      C, #2H
LOOP5:    XCH      A, X
          CLR1     CY
          RORC     A
          XCH      A, X
          RORC     A
          DECS     C
          BR       LOOP5
          RET

          END
```

(2) 75XTEST2.ASM

```
$  TITLE='A-D CONVERT'
;***********************************************
;***   A-D CONVERT PROGRAM       ***
;***********************************************
            NAME      AD_SUB
            EXTRN     DATA(TDATA), CODE (SEL15)
            PUBLIC    SIOSUB, ADCONV
            STKLN     2
;***         SIO SUB-ROUTINE      ***
SEG4        CSEG      SENT
SIOSUB:     PUSH      BS
            SEL       RB2
            SEL       MB1
            MOV       XA, TDATA
            GETI      SEL15           ;SEL     MB15
            MOV       SIO, XA
            MOV       XA, #11101110B
            MOV       SIOM, XA        ;CLOCK=262kHz, MSB
            POP       BS
            RET

;***         ANALOG INPUT    (RBE=0)    ***
SEG5        CSEG      SENT
ADCONV:     PUSH      BS
            GETI      SEL15           ;SEL     MB15
            MOV       HL, #0D3H
            MOV       XA, #0C0H
            MOV       BSB0, A         ;BSB0 ← 0H
LOOP:       SET1      BSB0.@L
            MOV       A, BSB0
            MOV       PTHM, XA        ;COMP.  START
            MOV       A, #0AH         ;18 MACHINE
WAIT:       INCS      A               ;CIRCLE WAIT
            BR        WAIT
            MOV1      CY, #H+PTH0.0
            MOV1      BSB0.@L, CY
            DECS      L
            BR        LOOP
            MOV       X, #0H
            MOV       A, BSB0
            ADDS      XA', XA         ;ADD DATA
            SET1      RBE
            POP       BS
            INCS      B               ;SAMPLE COUNT INC.
            RETI

            END
```

D.2 Execution Examples

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST1.ASM -C106 -KS -KX
75X Series Assembler VX.XX [XX Xxx XX]
    Copyright (C) NEC Corporatton 1985, XXXX


ASSEMBLY START


TARGET CHIP : UPD75106
STACK SIZE = 000AH

ASSEMBLY COMPLETE,   NO ERROR FOUND
```

```
A:\NECTOOLS\SMP75X\RA75X>RA75X 75XTEST2.ASM -C106 -KS -KX
75X Series Assembler VX.XX [XX Xxx XX]
    Copyright (C) NEC Corporation 1985, XXXX


ASSEMBLY START


TARGET CHIP = UPD75106
STACK SIZE = 000AH

ASSEMBLY COMPLETE, NO ERROR FOUND
```

```
A:\NECTOOLS\SMP75X\RA75X>LK75X 75XTEST1. REL 75XTEST2. REL -075XTEST. LNK
75X Series Linker VX.XX [XX Xxx XX]
    Copyright (C) NEC Corporation 1985


LINK COMPLETE, NO ERROR FOUND
```

```
A:\NECTOOLS\SMP75X\RA75X>0C75X 75XTEST. LNK
75X Series Object Converter VX.XX [XX Xxx XX]
    Copyright (C) NEC Corporation 1985, XXXX


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

```
A:\NECTOOLS\SMP75X\RA75X>LB75X
75X Series Librarian VX.XX [XX Xxx XX]
    Copyright (C) NEC Corporation 1984, XXXX

*CREATE 75XTEST.LIB
*ADD 75XTEST1.REL, 75XTEST2.REL TO 75XTEST.LIB
*LIST 75XTEST.LIB TO 75XTEST.LST PUBLICS
*EXIT
```

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -L75XTEST.LNK -A75XTEST1

List Conversion Program for RA75X VX.XX [XX Xxx XX]
    Copyright (C) NEC Corporation 1986, 1997

pass 1: start .....
pass 2: start .....
Conversion complete
```

```
A:\NECTOOLS\SMP75X\RA75X>LCNV75X -L75XTEST.LNK 75XTEST2

List Conversion Program for RA75X VX. XX [XX Xxx XX]
    Copyright (C) NEC Corporation 1986, 1997

pass 1: start .....
pass 2: start .....
Conversion complete
```

D.3 Output List

(1) Assembly list

(a) 75XTEST1.ASM assembly list

(Output to 75XTEST1.PRN.)

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :     X

    ** A-D CONVERTER VX.XX                              **

COMMAND       : 75XTEST1.ASM -C106 -KS -KX

STNO ADRS R OBJECT    IC MAC    SOURCE STATEMENT

     1                          $          TITLE='A-D CONVERTER VX.XX'
     2                          ;************************************************************
     3                          ;***       A-D CONVERT PROGRAM                  ***
     4                          ;************************************************************
     5                                     NAME     AD-MAIN
     6                                     EXTRN    CODE(ADCONV), CODE (SIOSUB)
     7                                     PUBLIC   TDATA, SEL15
     8                                     STKLN    10
     9  0000 R C000                        VENT0    MBE=1, RBE=1, MAIN
    10  0008 E 8000                        VENT4    MBE=1, RBE=0, ADCONV
    11
    12  ----                   SEG0       DSEG     1 AT 10H
    13  0110                   TDATA:     DS       2
    14
    15                         ;***       GETI TABLE             ***
    16
    17  ----                   SEG1       CSEG     IENT
    18  0000    991F           SEL15:     SEL      MB15
    19
    20                         ;***       MAIN ROUTINE           ***
    21
    22  ----                   SEG2       CSEG     INBLOCK
    23  0000    9921           MAIN:      SEL      RB1
    24
    25  0002 R 00                         GETI     SEL15          ;STACK POINTER SET
    26  0003 E 8900                       MOV      XA, #STACK      ;
    27  0005    9280                      MOV      SP, XA          ;
    28
    29  0007    73                        MOV      A, #0011B
    30  0008    93B3                      MOV      PCC, A         ;PCC ← 0011B
    31
```

```
32                          ;**     DATA RAM 0H-13FH ZERO CLEAR      **
33
34  000A   9911                    SEL     MB1
35  000C   8B3F                    MOV     HL, #3FH
36  000E   8900                    MOV     XA, #00H
37  0010   E8              LOOP1:  MOV     @HL, A          ; 100H-13FH
38  0011   AA6A                    DECS    HL
39  0013   FC                      BR      LOOP1
40  0014   9910                    SEL     MB0
41  0016   E8              LOOP2:  MOV     @HL, A          ;0H-FFH
42  0017   AA6A                    DECS    HL
43  0019   FC                      BR      LOOP2
44
45                          ;**     TIMER SET(SAMPLING TIME = 30MSEC, FXX=4.19MHZ) **
46
47  001A R 00                      GETI    SEL15           ;SEL    MB15
48  001B   8979                    MOV     XA, #79H
49  001D   92A6                    MOV     TMOD0, XA
50  001F   894C                    MOV     XA, #01001100B
51  0021   92A0                    MOV     TM0, XA
```

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :     X

    ** A-D CONVERTER VX.XX                              **

STNO ADRS R OBJECT    IC MAC    SOURCE STATEMENT

    52 0023    9DB2                      EI
    53 0025    9D9C                      EI          IET0
    54
    55 0027    9911                      SEL         MB1
    56 0029    8900              LOOP3:  MOV         XA, #00H
    57 002B    9A0F                      MOV         B, #0H
    58 002D    9A87              LOOP4:  SKE         B, #08H
    59 002F    FD                        BR          LOOP4
    60 0030 R  AB4000                    CALL        !HEIKIN
    61 0033    9210                      MOV         TDATA, XA
    62 0035 E  AB4000                    CALL        !SIOSUB
    63 0038    F0                        BR          LOOP3
    64
    65                           ;***    HEIKIN      (SAMPLE NUMBERS = 8)      ***
    66
    67 ----                      SEG3    CSEG        SENT
    68 0000    9A2E              HEIKIN: MOV         C, #2H
    69 0002    D9                LOOP5:  XCH         A, X
    70 0003    E6                        CLR1        CY
    71 0004    98                        RORC        A
    72 0005    D9                        XCH         A, X
    73 0006    98                        RORC        A
    74 0007    CE                        DECS        C
    75 0008    F9                        BR          LOOP5
    76 0009    EE                        RET
    77
    78                                   END
```

(b) 75XTEST2.ASM assembly list

(Output to 75XTEST2.PRN)

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :     X

    ** A-D CONVERT VX.XXSUB)                                          **

COMMAND       : 75XTEST2.ASM -C106 -KS -KX

STNO ADRS R OBJECT     IC MAC    SOURCE STATEMENT

     1                            $          TITLE='A-D CONVERTER VX.XX(SUB)'
     2                            ;*************************************************************
     3                            ;***       A-D CONVERT PROGRAM                  ***
     4                            ;*************************************************************
     5                                       NAME     AD-SUB
     6                                       EXTRN    CODE(SEL15)
     7                                       EXTRN    DATA(TDATA)
     8                                       PUBLIC   SIOSUB, ADCONV
     9                                       STKLN    2
    10                            ;***       SIO SUB-ROUTINE      ***
    11
    12  ----                      SEG4       CSEG     SENT
    13  0000    9907              SIOSUB:    PUSH     BS
    14  0002    9922                       SEL      RB2
    15  0004    9911                       SEL      MB1
    16  0006 E A200                        MOV      XA, TDATA
    17  0008 E 00                          GETI     SEL15        ;SEL     MB15
    18  0009    92E4                       MOV      SIO, XA
    19  000B    89EE                       MOV      XA, #11101110B
    20  000D    92E0                       MOV      SIOM, XA     ;CLOCK=262KHZ, MSB
    21  000F    9906                       POP      BS
    22  0011    EE                         RET
    23
    24                            ;***       ANALOG INPUT    (RBE=0)      ***
    25
    26  ----                      SEG5       CSEG     SENT
    27  0000    9907              ADCONV:    PUSH     BS
    28  0002 E 00                          GETI     SEL15        ;SEL     MB15
    29  0003    8BD3                       MOV      HL, #0D3H
    30  0005    89C0                       MOV      XA, #0C0H
    31  0007    93C0                       MOV      BSB0, A      ;BSB0 ← 0H
    32  0009    9D40              LOOP:    SET1     BSB0.@L
    33  000B    A3C0                       MOV      A, BSB0
    34  000D    92D6                       MOV      PTHM, XA     ;COMP. START
    35  000F    7A                         MOV      A, #0AH      ;18 MACHINE
    36  0010    C0                WAIT:    INCS     A            ;CIRCLE WAIT
```

```
37  0011   FE          BR      WAIT
38  0012   BD04        MOV1    CY, @H+PTH0.0
39  0014   9B40        MOV1    BSB0. @L, CY
40  0016   CA          DECS    L
41  0017   F1          BR      LOOP
42  0018   9A09        MOV     X, #0H
43  001A   A3C0        MOV     A, BSB0
44  001C   AAC1        ADDS    XA', XXA        ;ADD DATA
45  001E   9D80        SET1    RBE
46  0020   9906        POP     BS
47  0022   C7          INCS    B               ;SAMPLE COUNT INC.
48  0023   EF          RETI
49
50                     END
```

(2) Symbol table list

(a) 75XTEST1.ASM symbol table list

(Output to 75XTEST1.PRN.)

```
75X SERIES ASSEMBLER VX.XX                                    XX/XX/XX XX:XX:XX PAGE : X

** A-D CONVERTER VX.XX                                  **

SYMBOL TABLE LIST

OFFSET   TYPE         SYMBOL    OFFSET TYPE         SYMBOL    OFFSET  TYPE         SYMBOL    OFFSET  TYPE         SYMBOL

---------- ---------  AD_MAIN   ---------  CODE EXT  ADCONV    0000H   CODE         HEIKIN    0FBCH.1 PBIT         IETO
0010H    CODE         LOOP1     0016H    CODE      LOOP2     0029H   CODE         LOOP3     002DH   CODE         LOOP4
0002H    CODE         LOOP5     0000H    CODE      MAIN      0FB0H.1 PBIT         MBE       0FB3H   DATA         PCC
0FB0H.0  PBIT         RBE       0002H    DATA  PUB SEG0      0002H   CODE PUB     SEG1      0039H   CODE PUB     SEG2
000AH    CODE  PUB    SEG3      0000H    CODE  PUB SEL15     -------  CODE EXT    SIOSUB    0F80H   DATA         SP
---------- STACK EXT  STACK     0110H    DATA  PUB DATA      0FA0H   DATA         TM0       0FA6H   DATA         TMOD0

TARGET CHIP: UPD75106
STACK SIZE = 000AH

ASSEMBLY COMPLETE, NO ERROR FOUND
```

(b) 75XTEST2.ASM symbol table list

(Output to 75XTEST2.PRN.)

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX XX:XX:XX PAGE : X

** A-D CONVERTER VX.XX (SUB)                                  **

SYMBOL TABLE LIST

OFFSET   TYPE      SYMBOL       OFFSET   TYPE    SYMBOL          OFFSET TYPE    SYMBOL   OFFSETTYPE    SYMBOL

---------- ---------  AD_SUB       0000H    CODE    PUB ADCONV      0FC0H  DATA    BSB0     0009H  CODE   LOOP
0FD4H    DATA      PTH0         0FD6H    DATA          PTHM      0FB0H.0 PBIT    RBE      0012H  CODE   PUB SEG4
0024H    CODE      PUB SEG5     -------  CODE    EXT SEL15       0FE4H  DATA    SIO      0FE0H  DATA   SIOM
0000H    CODE      PUB SIOSUB   -------  DATA    EXT TDATA       0010H  CODE    WAIT
```

(3) Cross-reference lists

(a) 75XTEST1.ASM cross-reference list

(Output to 75XTEST1.PRN)

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :    X

    ** A-D CONVERTER VX.XX                              **

CROSS REFERENCE LIST

SYMBOL    TYPE     VALUE                                ATTRIBUTES XREF LIST

AD_MAIN   ------   -------                              1
ADCONV    CODE     -------      EXT                     6, 10
HEIKIN    CODE     0000H    R       SEG = SEG3          60, #68
IET0      PBIT     0FBCH.1                              53
LOOP1     CODE     0010H    R       SEG = SEG2          #37, 39
LOOP2     CODE     0016H    R       SEG = SEG2          #41, 43
LOOP3     CODE     0029H    R       SEG = SEG2          #56, 63
LOOP4     CODE     002DH    R       SEG = SEG2          #58, 59
LOOP5     CODE     0002H    R       SEG = SEG3          #69, 75
MAIN      CODE     0000H    R       SEG = SEG2          9, #23
MBE       PBIT     0FB0H.1                              9,10
PCC       DATA     0FB3H                                30
RBE       PBIT     0FB0H.0                              9, 10
SEG0      DATA     0002H        PUB ABS                 #12
SEG1      CODE     0002H        PUB REL  = IENT         #17
SEG2      CODE     0039H        PUB REL  = INBLOCK      #22
SEG3      CODE     000AH        PUB REL  = SENT         #67
SEL15     CODE     0000H    R   PUB SEG  = SEG1         7, #18, 25, 47
SIOSUB    CODE     -------      EXT                     6, 62
SP        DATA     0F80H                                27
STACK     STACK    ------       EXT                     26
TDATA     DATA     0110H        PUB ABS                 7, #13, 61
TM0       DATA     0FA0H                                51
TM0D0     DATA     0FA6H                                49

TARGET CHIP : UPD75106
STACK SIZE = 000AH
```

ASSEMBLY COMPLETE, NO ERROR FOUND

(b) 75XTEST2.ASM cross-reference list

(Output to 75XTEST2.PRN)

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :     X

    ** A-D CONVERT VX.XX(SUB)                                  **

CROSS REFERENCE LIST

SYMBOL    TYPE     VALUE                                  ATTRIBUTES XREF LIST

AD_SUB    ------   -------                                1
ADCONV    CODE     0000H    R  PUB SEG = SEG5             8, #27
BSB0      DATA     0FC0H                                  31, 32, 33, 39, 43
LOOP      CODE     0009H    R      SEG = SEG6             #32, 41
PTH0      DATA     0FD4H                                  38
PTHM      DATA     0FD6H                                  34
RBE       PBIT     0FB0H.0                                45
SEG4      CODE     0012H      PUB REL = SENT             #12
SEG5      CODE     0024H      PUB REL = SENT             #26
SEL15     CODE     -------    EXT                        6, 17, 28
SIO       DATA     0FE4H                                  18
SIOM      DATA     0FE0H                                  20
SIOSUB    CODE     0000H    R  PUB SEG = SEG4             8, #13
TDATA     DATA     -------    EXT                        7, 16
WAIT      CODE     0010H    R      SEG = SEG5             #36, 37

TARGET CHIP : UPD75106
STACK SIZE = 0002H

ASSEMBLY COMPLETE, NO ERROR FOUND
```

(4) Link list (linker control list, input/output module list, segment list, and symbol table list)

(Output to 75XTEST1.MAP)

```
75X SERIES LINKER VX.XX                    XX/XX/XX  XX:XX:XX                PAGE     X

COMMAND     :  75XTEST1.REL 75XTEST2.REL-075XTEST.LNK

INPUT MODULE LIST:
  75XTEST1.REL       (AD_MAIN)
  75XTEST2.REL       (AD_SUB)

LOAD MODULE LIST :
  75XTEST.LNK        (AD_MAIN)

SEGMENT LINK MAP FOR 75XTEST.LNK (AD_MAIN)

MAP OF ROM AREA :

                BASE      LENGTH    MODULE NAME          SEGMENT NAME    (TYPE)
                ---------  ----------  ----------------------  ------------------------------------------------
                0000H     0002H     AD_MAIN                              (ABSOLUTE)
                0002H     0006H                          ** GAP **
                0008H     0002H     AD_MAIN                              (ABSOLUTE)
                000AH     0012H     AD_SUB               SEG4            (SENT)
                001CH     0004H                          ** GAP **
                0020H     0002H     AD_MAIN              SEG1            (IENT)
                0022H     0024H     AD_SUB               SEG5            (SENT)
                0046H     000AH     AD_MAIN              SEG3            (SENT)
                0050H     0039H     AD_MAIN              SEG2            (INBLOCK)
                0089H     16F7H                          ** GAP **

MAP OF RAM AREA:

                TYPE              BASE     LENGTH     MODULE NAME          SEGMENT NAME
                --------          --------  ----------  ----------------------  -------------------------
                                  0000H    00F4H                          ** GAP **
                STACK             00F4H    000CH      AD_MAIN             SSEG
                                  0100H    0010H                          ** GAP **
                DATA              0110H    0002H      AD_MAIN             SEG0
                                  0112H    002EH                          ** GAP **
```

PUBLIC SYMBOL LIST FOR 75XTEST.LNK

TYPE	VALUE	MODULE	SYMBOL NAME
CODE	0022H	AD_SUB	ADCONV
DATA	0110H	AD_MAIN	SEG0
CODE	0020H	AD_MAIN	SEG1
CODE	0050H	AD_MAIN	SEG2
CODE	0046H	AD_MAIN	SEG3
CODE	000AH	AD_SUB	SEG4
CODE	0022H	AD_SUB	SEG5
CODE	0020H	AD_MAIN	SEL15
CODE	000AH	AD_SUB	SIOSUB

```
75X SERIES LINKER  VX.XX                          XX/XX/XX  XX:XX:XX  PAGE :    X

SYMBOL LIST FOR 75XTEST.LNK

TYPE      VALUE      ATTRIBUTE    NAME
---------  ----------  ---------    ---------

---------  ----------  MODULE       AD_MAIN
CODE      0046H      SYMBOL       HEIKIN
PBIT      0FBCH.1    SYMBOL       IET0
CODE      0060H      SYMBOL       LOOP1
CODE      0066H      SYMBOL       LOOP2
CODE      0079H      SYMBOL       LOOP3
CODE      007DH      SYMBOL       LOOP4
CODE      0048H      SYMBOL       LOOP5
CODE      0050H      SYMBOL       MAIN
PBIT      0FB0H.1    SYMBOL       MBE
DATA      0FB3H      SYMBOL       PCC
PBIT      0FB0H.0    SYMBOL       RBE
DATA      0110H      PUBLIC       SEG0
CODE      0020H      PUBLIC       SEG1
CODE      0050H      PUBLIC       SEG2
CODE      0046H      PUBLIC       SEG3
CODE      0020H      PUBLIC       SEL15
DATA      0F80H      SYMBOL       SP
DATA      0110H      PUBLIC       TDATA
DATA      0FA0H      SYMBOL       TM0
DATA      0FA6H      SYMBOL       TM0D0

---------  ----------  MODULE       AD_SUB
CODE      0022H      PUBLIC       ADCONV
DATA      0FC0H      SYMBOL       BSB0
CODE      002BH      SYMBOL       LOOP
DATA      0FD4H      SYMBOL       PTH0
DATA      0FD6H      SYMBOL       PTHM
PBIT      0FB0H.0    SYMBOL       RBE
CODE      000AH      PUBLIC       SEG4
CODE      0022H      PUBLIC       SEG5
DATA      0FE4H      SYMBOL       SIO
DATA      0FE0H      SYMBOL       SIOM
CODE      000AH      PUBLIC       SIOSUB
CODE      0032H      SYMBOL       WAIT

LINK COMPLETE, NO ERROR FOUND
```

(5) HEX format object module file

(Output to 75XTEST1.HEX)

```
:02000000C050EE
:1000080080229907992299 11A2101092E489EE9200
:04001800E09906EE77
:10002000991F9907108BD389C093C09D40A3C0929C
:10003000D67AC0FEBD049B40CAF19A09A3C0AAC1EA
:100040009D809906C7EF9A2ED9E698D998CEF9EEF9
:1000500099211089009280739 3B399118B3F890085
:10006000E8AA6AFC9910E8AA6AFC10897992A68924
:100070004C92A09DB29D9C991189009A0F9A87FFD80
:09008000AB40469210AB400AF0BF
:00000001FF
```

(6) Symbol table file

(Output to 75XTEST.SYM)

```
#04
;FF   AD_MAIN
020110SEG0
010020SEG1
010050SEG2
010046SEG3
010020SEL15
030100STACK
020110TDATA
<010046HEIKIN
083EF1IET0
010060LOOP1
010066LOOP2
010079LOOP3
01007DLOOP4
010048LOOP5
010050MAIN
083EC1MBE
020FB3PCC
083C0RBE
020F80SP
020FA0TM0
020FA6TM0D0
;FF   AD_SUB
010022ADCONV
01000ASEG4
010022SEG5
01000ASIOSUB
<020FC0BSB0
0100SBLOOP
020FD4PTH0
020FD6PTHM
083EC0RBE
020FE4SIO
020FE0SIOM
010032WAIT
=
```

(7) Library file information list

(Output to 75XTEST.LST)

```
75X Series Librarian VX.XX                DATE(                    )    PAGE :    X

   LIB-FILE NAME : 75XTEST.LIB      (                                          )

   1    AD_MAIN              (                                        )
        UPDATA :    0        RA75X VX.XX          UPD75106

        SEG0
        SEG1
        SEG2
        SEG3
        SEL15
        TDATA

                NUMBER OF PUBLIC SYMBOLS :     6

   2    AD_SUB               (                                        )
        UPDATE :    0        RA75X VX.XX          UPD75106

        ADCONV
        SEG4
        SEG5
        SIOSUB

                NUMBER OF PUBLIC SYMBOLS :     4


                NUMBER OF MODULES :     2
```

(8) Absolute assembly list

(a) 75XTEST1.PRN absolute assembly list

(Output to 75XTEST1.P)

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :     X

    ** A-D CONVERTER VX.XX                              **

COMMAND       : 75XTEST1.ASM -C106        -KS      -KX

STNO ADRS R OBJECT     IC MAC    SOURCE STATEMENT

     1                              $          TITLE='A-D CONVERTER VX.XX'
     2                              ;************************************************************
     3                              ;***       A-D CONVERT PROGRAM                    ***
     4                              ;************************************************************
     5                                         NAME     AD-MAIN
     6                                         EXTRN    CODE(ADCONV), CODE (SIOSUB)
     7                                         PUBLIC   TDATA, SEL15
     8                                         STKLN    10
     9  0000 R C000                            VENT0    MBE=1, RBE=1, MAIN
    10  0008 E 8000                            VENT4    MBE=1, RBE=0, ADCONV
    11
    12  ----                       SEG0        DSEG     1 AT 10H
    13  0110                       TDATA:      DS       2
    14
    15                             ;***        GETI TABLE              ***
    16
    17  ----                       SEG1        CSEG     IENT
    18  0020    991F               SEL15:      SEL      MB15
    19
    20                             ;***        MAIN ROUTINE            ***
    21
    22  ----                       SEG2        CSEG     INBLOCK
    23  0050    9921               MAIN:       SEL      RB1
    24
    25  0052 R 10                              GETI     SEL15           ;STACK POINTER SET
    26  0053 E 8900                            MOV      XA, #STACK       ;
    27  0055    9280                           MOV      SP, XA           ;
    28
    29  0057    73                             MOV      A, #0011B
    30  0058    93B3                           MOV      PCC, A          ;PCC ← 0011B
    31
    32                             ;**         DATA RAM 0H-13FH ZERO CLEAR      **
    33
    34  005A    9911                           SEL      MB1
```

```
35  005C    8B3F                    MOV     HL, #3FH
36  005E    8900                    MOV     XA, #00H
37  0060    E8              LOOP1:  MOV     @HL, A          ; 100H-13FH
38  0061    AA6A                    DECS    HL
39  0063    FC                      BR      LOOP1
40  0064    9910                    SEL     MB0
41  0066    E8              LOOP2:  MOV     @HL, A          ;0H-FFH
42  0067    AA6A                    DECS    HL
43  0069    FC                      BR      LOOP2
44
45                          ;**     TIMER SET(SAMPLING TIME = 30MSEC, FXX=4.19MHZ) **
46
47  006A R 00                       GETI    SEL15           ;SEL    MB15
48  006B    8979                    MOV     XA, #79H
49  006D    92A6                    MOV     TM0D0, XA
50  006F    894C                    MOV     XA, #01001100B
51  0071    92A0                    MOV     TM0, XA
```

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :     X

    ** A-D CONVERTER VX.XX                              **

STNO ADRS R OBJECT    IC MAC    SOURCE STATEMENT

    52  0073    9DB2                      EI
    53  0075    9D9C                      EI          IET0
    54
    55  0077    9911                      SEL         MB1
    56  0079    8900              LOOP3:  MOV         XA, #00H
    57  007B    9A0F                      MOV         B, #0H
    58  007D    9A87              LOOP4:  SKE         B, #08H
    59  007F    FD                        BR          LOOP4
    60  0080 R  AB4046                    CALL        !HEIKIN
    61  0083    9210                      MOV         TDATA, XA
    62  0085 E  AB400A                    CALL        !SIOSUB
    63  0088    F0                        BR          LOOP3
    64
    65                            ;***    HEIKIN      (SAMPLE NUMBERS = 8)      ***
    66
    67  ----                      SEG3    CSEG        SENT
    68  0046    9A2E              HEIKIN: MOV         C, #2H
    69  0048    D9                LOOP5:  XCH         A, X
    70  0049    E6                        CLR1        CY
    71  004A    98                        RORC        A
    72  004B    D9                        XCH         A, X
    73  004C    98                        RORC        A
    74  004D    CE                        DECS        C
    75  004E    F9                        BR          LOOP5
    76  004F    EE                        RET
    77
    78                                    END
```

```
75X SERIES ASSEMBLER VX.XX                          XX/XX/XX XX:XX:XX PAGE : X

** A-D CONVERTER VX.XX                                        **

SYMBOL TABLE LIST

OFFSET   TYPE         SYMBOL    OFFSET  TYPE         SYMBOL   OFFSET  TYPE         SYMBOL   OFFSET  TYPE         SYMBOL

---------- ---------  AD_MAIN   ---------CODE  EXT  ADCONV   0000H   CODE         HEIKIN   0FBCH.1 PBIT         IETO
0010H    CODE         LOOP1     0016H   CODE         LOOP2    0029H   CODE         LOOP3    002DH   CODE         LOOP4
0002H    CODE         LOOP5     0000H   CODE         MAIN     0FB0H.1 PBIT         MBE      0FB3H   DATA         PCC
0FB0H.0  PBIT         RBE       0002H   DATA  PUB    SEG0     0002H   CODE PUB     SEG1     0039H   CODE PUB     SEG2
000AH    CODE  PUB    SEG3      0000H   CODE  PUB    SEL15    -------  CODE EXT    SIOSUB   0F80H   DATA         SP
---------- STACK EXT  STACK     0110H   DATA  PUB    TDATA    0FA0H   DATA         TM0      0FA6H   DATA         TMOD0
```

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :     X

     ** A-D CONVERTER VX.XX                                  **

CROSS REFERENCE LIST

SYMBOL    TYPE     VALUE                                    ATTRIBUTES XREF LIST

AD_MAIN   ------   -------                                  1
ADCONV    CODE     -------      EXT                         6, 10
HEIKIN    CODE     0000H    R       SEG = SEG3              60, #68
IET0      PBIT     0FBCH.1                                  53
LOOP1     CODE     0010H    R       SEG = SEG2              #37, 39
LOOP2     CODE     0016H    R       SEG = SEG2              #41, 43
LOOP3     CODE     0029H    R       SEG = SEG2              #56, 63
LOOP4     CODE     002DH    R       SEG = SEG2              #58, 59
LOOP5     CODE     0002H    R       SEG = SEG3              #69, 75
MAIN      CODE     0000H    R       SEG = SEG2              9, #23
MBE       PBIT     0FB0H.1                                  9,10
PCC       DATA     0FB3H                                    30
RBE       PBIT     0FB0H.0                                  9, 10
SEG0      DATA     0002H        PUB ABS                     #12
SEG1      CODE     0002H        PUB REL  = IENT             #17
SEG2      CODE     0039H        PUB REL  = INBLOCK          #22
SEG3      CODE     000AH        PUB REL  = SENT             #67
SEL15     CODE     0000H    R   PUB SEG  = SEG1             7, #18, 25, 47
SIOSUB    CODE     -------      EXT                         6, 62
SP        DATA     0F80H                                    27
STACK     STACK    ------       EXT                         26
TDATA     DATA     0110H        PUB ABS                     7, #13, 61
TM0       DATA     0FA0H                                    51
TM0D0     DATA     0FA6H                                    49

TARGET CHIP : UPD75106
STACK SIZE = 000AH

ASSEMBLY COMPLETE, NO ERROR FOUND
```

(b) 75XTEST2.PRN absolute assembly list

(Output to 75XTEST2.P)

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :    X

    ** A-D CONVERT VX.XX (SUB)                                        **

COMMAND       : 75XTEST2.ASM -C106 -KS -KX

STNO ADRS R OBJECT    IC MAC    SOURCE STATEMENT

     1                          $          TITLE='A-D CONVERTER VX.XX(SUB)'
     2                          ;*************************************************************
     3                          ;***       A-D CONVERT PROGRAM                  ***
     4                          ;*************************************************************
     5                                     NAME     AD-SUB
     6                                     EXTRN    CODE(SEL15)
     7                                     EXTRN    DATA(TDATA)
     8                                     PUBLIC   SIOSUB, ADCONV
     9                                     STKLN    2
    10                          ;***       SIO SUB-ROUTINE      ***
    11
    12  ----                   SEG4       CSEG     SENT
    13  000A    9907           SIOSUB:    PUSH     BS
    14  000C    9922                       SEL      RB2
    15  000E    9911                       SEL      MB1
    16  0010 E A210                        MOV      XA, TDATA
    17  0012 E 10                          GETI     SEL15        ;SEL     MB15
    18  0013    92E4                       MOV      SIO, XA
    19  0015    89EE                       MOV      XA, #11101110B
    20  0017    92E0                       MOV      SIOM, XA     ;CLOCK=262KHZ, MSB
    21  0019    9906                       POP      BS
    22  001B    EE                         RET
    23
    24                          ;***       ANALOG INPUT    (RBE=0)      ***
    25
    26  ----                   SEG5       CSEG     SENT
    27  0022    9907           ADCONV:    PUSH     BS
    28  0024 E 10                          GETI     SEL15        ;SEL     MB15
    29  0025    8BD3                       MOV      HL, #0D3H
    30  0027    89C0                       MOV      XA, #0C0H
    31  0029    93C0                       MOV      BSB0, A      ;BSBO ← 0H
    32  002B    9D40           LOOP:      SET1     BSB0.@L
    33  002D    A3C0                       MOV      A, BSB0
    34  002F    92D6                       MOV      PTHM, XA     ;COMP. START
    35  0031    7A                         MOV      A, #0AH      ;18 MACHINE
    36  0032    C0             WAIT:      INCS     A            ;CIRCLE WAIT
    37  0033    FE                         BR       WAIT
```

```
38  0034   BD04          MOV1    CY, @H+PTH0,0
39  0036   9B40          MOV1    BSB0, @L, CY
40  0038   CA            DECS    L
41  0039   F1            BR      LOOP
42  003A   9A09          MOV     X, #0H
43  003C   A3C0          MOV     A, BSB0
44  003E   AAC1          ADDS    XA', XXA        ;ADD DATA
45  0040   9D80          SET1    RBE
46  0042   9906          POP     BS
47  0044   C7            INCS    B               ;SAMPLE COUNT INC.
48  0045   EF            RETI
49
50                        END
```

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX XX:XX:XX PAGE : X

** A-D CONVERTER VX.XX (SUB)                                **

SYMBOL TABLE LIST

OFFSET   TYPE     SYMBOL        OFFSET   TYPE        SYMBOL      OFFSET TYPE  SYMBOL   OFFSET TYPE        SYMBOL

------------ --------- AD_MAIN      0000H    CODE    PUB ADCONV  0FC0H  DATA  BSB0     0009H  CODE        LOOP
0FD4H    DATA     PTH0          0FD6H    DATA    PTHM        0FB0H.0 PBIT  RBE      0012H  CODE  PUB SEG4
0024H    CODE     PUB SEG5      -------  CODE    EXT SEL15   0FE4H  DATA  SIO      0FE0H  DATA        SIOM
0000H    CODE     PUB SIOSUB    -------  DATA    EXT TDATA   0010H  CODE  WAIT
```

```
75X SERIES ASSEMBLER VX.XX                    XX/XX/XX  XX:XX:XX  PAGE :     X

     ** A-D CONVERTER VX.XX(SUB)                          **

CROSS REFERENCE LIST

SYMBOL    TYPE     VALUE                                  ATTRIBUTES XREF LIST

AD_SUB    ------   -------                                1
ADCONV    CODE     0000H    R  PUB SEG = SEG5             8, #27
BSB0      DATA     0FC0H                                  31, 32, 33, 39, 43
LOOP      CODE     0009H    R      SEG = SEG5             #32, 41
PTH0      DATA     0FD4H                                  38
PTHM      DATA     0FD6H                                  34
RBE       PBIT     0FB0H.0                                45
SEG4      CODE     0012H       PUB REL = SENT             #12
SEG5      CODE     0024H       PUB REL = SENT             #26
SEL15     CODE     -------     EXT                        6, 17, 28
SIO       DATA     0FE4H                                  18
SIOM      DATA     0FE0H                                  20
SIOSUB    CODE     0000H    R  PUB SEG = SEG4             8, #13
TDATA     DATA     -------     EXT                        7, 16
WAIT      CODE     0010H    R      SEG = SEG5             #36, 37

TARGET CHIP : UPD75106
STACK SIZE = 0002H

ASSEMBLY COMPLETE, NO ERROR FOUND
```

APPENDIX E. INDEX

E.1 Index

[MEMO]